YU ISSN 0350-5596

informatica

Časopis izdaja Slovensko društvo INFORMATIKA, 61000 Ljubljana, Jamova 39, Jugoslavija

Uredništvo in uprava: 61000 Ljubljana, Institut "Jožef Štefan", Jamova 39, telef. (061)263-261, telegram JOSTIN, telex: 31 296 YU JOSTIN.

Letna naročnina za delovne organizacije je 350,00 din, za posameznika 120,00 din, prodaja posamezne številke 60,00 din.
Žiro račun št.: 50101-678-51841
Stališče uredništva se lahko razlikuje od mnenja avtorjev.

Pri financiranju revije sodeluje tudi Raziskovalna skupnost Slovenije.

Na podlagi mnenja Republiškega sekretariata za prosveto in kulturo št. 4210-44/79 z dne 1.2.1979, je časopis oproščen temeljnega davka od prometa proizvodov.

Tisk: Tiskarna KRESIJA, Ljubljana

Grafična oprema: Rasto Kirn

ČASOPIS ZA TEHNOLOGIJO RAČUNALNIŠTVA IN PROBLEME INFORMATIKE
ČASOPIS ZA RAČUNARSKO TEHNOLOGIJU I PROBLEME INFORMATIKE
SPISANIE ZA TEHNOLOGIJA NA SMETANJEJO I PROBLEMI OD OBLASTA NA INFORMATIKATA

YU ISSN 0350 – 5596

LETNIK 4, 1980 – št. 3

V S E B I N A

informatica

Published by INFORMATIKA, Slovene Society for Informatics, 61000 Ljubljana, Jamova 39, Yugoslavia

Headquarters: 61000 Ljubljana, Institut "Jožef Stefan", Jamova 39, Phone: (061)263 261, Cable: JOSTIN Ljubljana, Telex: 31 296 YU JOSTIN.

Annual subscription rate for abroad is US $ 22 for companies, and US $ 7,5 for individuals.

Printed by: Tiskarna KRESIJA, Ljubljana

DESIGN: Rasto Kirn

JOURNAL OF COMPUTING AND INFORMATICS

YU ISSN 0350 – 5596

VOLUME 4, 1980 – No. 3

CONTENTS

SISTEMI Z VEČ PROCESORJI

DRAGO NOVAK

UDK: 681.3.012

ELEKTROTEHNA, DO DELTA, LJUBLJANA

Članek opisuje problematiko računalniških sistemov z več procesorji. Začenja s cilji, ki jih hočemo doseči s takimi sistemi. Temu sledi klasifikacija sistemov glede na stopnjo povezanosti elementov sistema in glede na topološko strukturo. Podrobneje so obravnavani tesno povezani sistemi. Naštete so prednosti in slabosti posameznih struktur. Načeta je tudi problematika sistemske kontrole s poudarkom na sinhronizaciji. Pri tem je bilo največ pozornosti posvečeno distribuirani kontroli. Na koncu sta dodana dva zgleda sistemov z več procesorji.

MULTIPLE PROCESSOR SYSTEMS. This article describes computer systems with multiple processors. It begins with goals that we want to achieve with such system. This is followed by classification of systems based on degree of coupling between elements and on topological structure. Close coupled systems are discused more in detail. Advantages and disadvantages of particular structures are quoted next. We also touch the problems of system control where problems of synchronization are stressed. The main attention was laid on distributed control. At the end two examples of multiple processor systems are presented.

1. UVOD

V literaturi vse češče srečujemo izraze kot so: multiprocesor, multiračunalnik, računalniška mreža, distribuirano procesiranje, ipd. Dejstvo je, da postajajo sistemi oz.koncepti, ki se skrivajo za temi izrazi, vse popularnejši. V članku bomo poskusili opredeliti cilje, ki jih hočemo doseči s takimi sistemi; klasificirali bomo sisteme glede na posamezne parametre kot so : stopnja povezanosti, topologija, vrsta povezave, ipd; seznanili se bomo z različnimi pristopi h kontroli takih sistemov, pri čemer se bomo najdalj zadržali pri t.im. distribuirani kontroli, ki trenutno najbolj buri duhove med strokovnjaki iz tega področja; in na koncu bomo predstavili in analizirali dva obstoječa sistema z več procesorji.

Definicij pomena posameznih, zgoraj naštetih izrazov se bomo lotili šele pri klasifikaciji sistemov. Zato bomo do takrat uporabljali izraz sistem z več procesorji (multiple processor system), s katerim bomo zajeli vso palato sistemov, ki vsebujejo več procesorjev - od zelo tesno povezanih multiprocesorskih sistemov pa do računalniških mrež.

2. CILJI PRI NAČRTOVANJU

Sisteme z več procesorji gradimo zato, da povečamo zmožnosti sistema v primerjavi z zmožnostmi enega samega procesorja. Gre predvsem za povečanje fleksibilnosti, integritete in zmogljivosti. Pri načrtovanju sistemov postavljamo za cilj izboljšanje vseh treh lastnosti, vendar so te utežene z različnimi utežmi. Želimo imeti n.pr. zelo fleksibilen sistem, pri čemer žrtvujemo zmogljivost (n.pr. hitrost).

2.1. Fleksibilnost

Fleksibilnost, ki je tipična lastnost vsakega sistema z množicami enakih elementov, je tudi odlika sistemov z več procesorji. V literaturi srečamo za to lastnost še izraze kot so: modularnost, razširljivost, prilagodljivost, ipd.

Sistem je fleksibilen, če ga je mogoče na enostaven način modificirati in če ga je mogoče na enostaven način širiti. V prvem primeru gre za spremembo določenih funkcij sistema, ki zahteva zamenjavo nekaterih hardwarskih ali programskih elementov. Če tak poseg ne potegne za seboj celega plazu sprememb na ostalih elementih sistema (ponovno načrtovanje), pravimo, da smo dosegli modifikacijo na enostaven način. V splošnem so sistemi z več procesorji manj občutljivi na spremembe kot monoprocesorski sistemi. Seveda pa zavisi fleksibilnost v smislu modificiranja od zgradbe sistema (stopnja povezanosti, vrsta kontrole, ipd.).

Pri širjenju sistema gre za dodajanje novih elementov, ki ima za posledico povečanje zmogljivosti sistema pri istem naboru funkcij ali povečanje števila funkcij. Pri tem stremimo, da je razpon med minimalno in maksimalno konfiguracijo čim večji ter da je mogoče znotraj teh meja večati sistem v čim manjših korakih. Pri monoprocesorskih sistemih je večanje sistema omejeno na večanje vhodno-izhodnih kapacitet in na širjenje programske opreme. Povečanje zmogljivosti pri istem naboru funkcij je praktično nemogoče. Sisteme z več procesorji je mogoče enostavje širiti v obeh pogledih. Tudi tu lahko dodamo, da igra pri tem važno vlogo zgradba oz.vrsta sistema (mreža, multiprocesor, porazdeljeni sistem, ipd.).

2.2. Integriteta

Z integriteto označujemo stopnjo neobčutljivosti na napake in izpade. To lastnost imenujejo nekateri tudi zanesljivost (reliability) ali tolerance napak (fault tolerance). Napaka, ki se pojavi v hardwarskem ali programskem delu sistema povzroči izpad dela sistema, ki se lahko razširi do izpada celega sistema. Integriteta obsega detektiranje napak, ograjevanje (preprečevanje širjenja posledic napake), diagnosticiranje, vzpostavitev konsistentnega stanja sistema, popravljanje napake in ponovno vključitev popravljenega elementa v sistem. Osnovni pogoj za naštete dejavnosti pa je redundantnost. Ta je prisotna v mnogo večji meri v sistemih z več procesorji - v monoprocesorskih sistemih ni redundantnih procesorjev. Ponovno lahko torej trdimo, da je integriteta sistemov z več procesorji boljša kot integriteta monoprocesorskih sistemov.

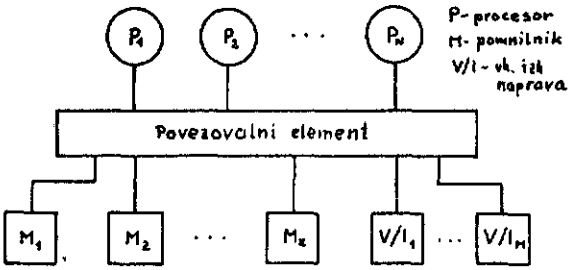
2.3. Zmogljivost

Zmogljivost sistema merimo z odzivnim časom ali z *propustnostjo (troughput)*. Zmogljivost vsakega monoprocesorskega sistema ima svojo mejo, ki jo je mogoče preseči samo z arhitekturo z več procesorji. Odzivni čas, ki je tako zelo kritičen v nekaterih aplikacijah, lahko v sistemih z več procesorji bistveno znižamo. To storimo tako, da določeno število procesorjev namenimo izključno za obravnavo časovno kritičnih signalov. S tem, da so ti procesorji večino časa brez dela (zaradi potrebe po hitrem odzivu si ne smemo privoščiti "overheada" vezanega na preklop konteksta), smo seveda zmanjšali propustnost sistema, ki pač ni bila odločujoči faktor. Povečanje zmogljivosti ne poteka linearno s številom procesorjev v sistemu. Vzrok temu je dejstvo, da mora vsak procesor nameniti del svojih zmogljivosti (časa) za sinhronizacijo in komunikacijo. Sinhronizacija je potrebna pri uporabi skupnih virov sistema. Komunikacije pa je pogojena z logično odvisnostjo procesov, ki se odvijajo na posameznih procesorjih. V vsakem sistemu z več procesorji doseže krivulja *zmogljivosti sistema proti številu procesorjev* prej ali slej zasičenje, ko zmogljivosti ne moremo več povečati, čeprav dodajamo nove procesorje. Kako hitro pridemo do te točke pa zavisi predvsem od tega kako dobro se "prilega" arhitektura sistema dani aplikaciji in kako učinkovito so implementirane sinhronizacijske in komunikacijske operacije.

3. KLASIFIKACIJA SISTEMOV

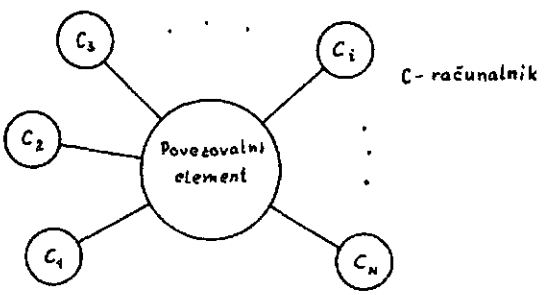
S isteme z več procesorji je mogoče klasificirati po več kriterijih. V nadaljevanju si bomo ogledali klasifikacijo glede na stopnjo povezanosti, glede na topologijo in glede na vrsto povezav. Še prej pa bomo poskusili definirati nekaj pojmov.

- Multiprocesor. Multiprocesor je sistem z *več procesorji, katerih delo nadzoruje centralni* operacijski sistem. Vsak procesor ima dostop do skupnega pomnilnika in skupnih V/I naprav.
Centralni operacijski sistem skrbi za razporejanje in posredovanje nalog - procesov posameznim procesorjem. Vsebuje tudi sinhronizacijske in komunikacijske operacije, ki omogočajo koordinacijo procesov, ki se odvijajo na posameznih procesorjih. Povezovalni element med procesorji, pomnilniškimi moduli in V/I krmilniki mora preprečevati konfliktne situacije, ki bi se pojavile pri hkratni uporabi istega vira.



Slika 3.0.1. Struktura multiprocesorskega sistema

- Računalniška mreža. Računalniška mreža je sistem z več računalniki, ki so med seboj povezani preko standardnih komunikacijskih kanalov. Z računalnikom označujemo sistem, ki vsebuje procesor, pomnilnik in V/I naprave ter operacijski sistem, ki povezuje omenjene elemente v celoto.



Slika 3.0.2. Struktura računalniške mreže.

Tudi v mreži imamo globalni operacijski sistem, ki se nahaja hierarhično nad lokalnim operacijskim sistemom in običajno ni centraliziran. Sodelovanje procesov je mogoče preko sporočil. Pošiljanje in sprejemanje sporočil se odvija po določenem komunikacijskem protokolu.

- Multiračunalnik. Multiračunalnik je sistem z več računalniki, ki so na specifičen način povezani. Čeprav je ta izraz zadnje čase zelo popularen, vidimo, da se po svoji definiciji ne razlikuje od računalniške mreže. Poglejmo kje leži razlog za dve različni imeni za isto strukturo sistema. Multiprocesorji imajo zaradi centraliziranega operacijskega sistema in specifičnega povezovalnega elementa precej slabih lastnosti (integriteta). *Prednosti multi*procesorjev ponavadi ne odtehtajo cene in napora vloženega v načrtovanje. Poleg tega večina aplikacij ne zahteva tako tesno povezanega sistema. Zato so mnogi razvijalci ubrali drugo pot. Procesorjem so dodali privatni pomnilnik, operacijski sistem so razbili na več lokalnih in prešli na koordinacijo s pomočjo sporočil. S tem so prišli do strukture mreže, ki pa je niso tako poimenovali, ker je bil sistem še vedno prostorsko zelo cenraliziran. Računalniško mrežo pa si intuitivno predstavljamo kot nekaj velikega: razdalje 100 kilometrov in več, velike računske centre z ogromnimi računalniki, ipd.

- Distribuirani sistem (distributed system). Za razliko od prvih treh terminov, ki skrivajo za seboj posamezne organizacije - zgradbe si-

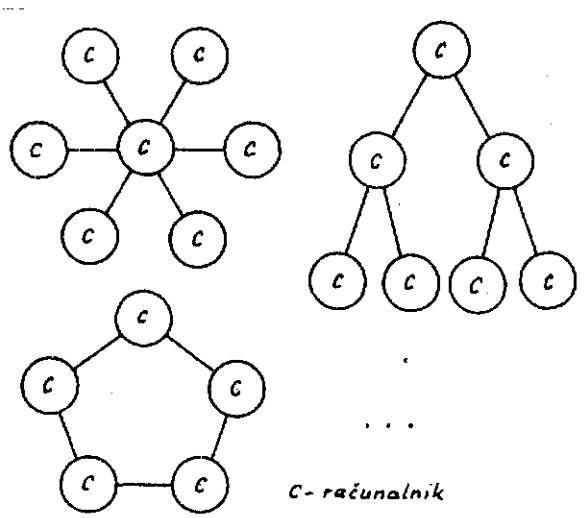
stemov, pomeni pojem distribuirani sistem popolnoma nov koncept, ki obsega organizacijo - arhitekturo sistema, organizacijo podatkov, način obdelave in vrsto kontrole. Ti sistemi bazirajo običajno na t.im. multiračunalnikih. Globalni podatki so porazdeljeni po sistemu. Tudi operacijski sistem je razpršen po sistemu. Noben proces nima ažurnih podatkov o trenutnem globalnem stanju sistema. Vse odločitve se sprejemajo na podlagi "subjektivnih" mnenj o globalnem stanju sistema.

3.1. Stopnja povezanosti

Glede na stopnjo povezanosti delimo sisteme z več procesorji v multiprocesorje in v računalniške mreže. V prvem sistemu so procesorji tesno povezani, v drugem pa je povezava bolj ohlapna. Stopnjo povezanosti lahko merimo oz.ocenjujemo s kapaciteto komunikacijskih kanalov in časom dostopa. Ena od najbolj konkretnih definicij je sledeča (FULLER78): stopnja povezanosti je definirana z minimalnim časom dostopa do globalne podatkovne strukture v sistemu pri najneugodnejših razmerah. V multiprocesorskem sistemu ima vsak procesor direkten dostop do globalnih podatkov, ki so shranjeni v primarnem - delovnem pomniniku. Tu se gibljejo dostopni časi v mejah 1 - 50 mikrosekund. Najneugodnejši trenutek je takrat, ko vsi hkrati zahtevajo dostop do skupnega pomnilnika. Merimo čas od zahteve pa do omogočitve dostopa zadnjemu procesorju. V računalniških mrežah so ti časi mnogo večji. Gibljajo se v mejah 0,1 - 1 sekunde. Tu je v najneugodnejšem položaju računalnik, ki je najbolj oddaljen od globalnih podatkov (razdaljo merimo s številom vmesnih tranzitnih računalnikov).

3.2. Topologija

V prejšnjem poglavju smo ocenjevali stopnjo povezanosti procesorjev v sistemu. Sedaj pa si poglejmo zgradbo oz. organizacijo teh povezav, ki določajo topologijo sistema. V računalniških mrežah so elementi, ki jih povezujemo v sistem računalniki. Povezave pa dajo sistemu karakteristično topologijo zvezde, drevesa, obroča, popolno povezanega grafa, itd. Zgradba samih računalnikov nas ne zanima, saj bazira sistem na pošiljanju in sprejemanju sporočil.

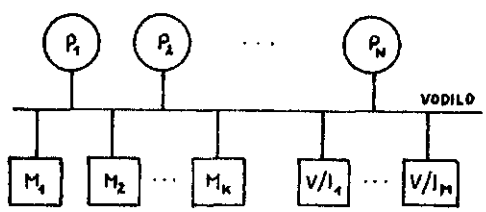


Slika 3.2. Topologije mrež.

Pri multiprocesorskih sistemih nimamo več opravka s sporočili temveč s signali. Gre za tesno povezane sisteme - procesorji so povezani preko sistemskih vodil (naslovno, podatkovno in kontrolno vodilo). Tipične topološke strukture so: skupno časovno deljeno vodilo; matrika stikal, ki preklapljajo vodila (crossbar) in pomniniki z več vrati na katera lahko priključujemo vodila.

3.2.1. Skupno vodilo

Vodilo predstavlja interni povezovalni element vsakega računalnika. Običajno ga sestavljajo: podatkovno vodilo, naslovno vodilo in kontrolno vodilo. V multiprocesorskem sistemu je na vodilo priključenih več procesorjev. S tem je postalo vodilo in z njim vsi pasivni moduli priključeni nanj, skupna last vseh procesorjev.

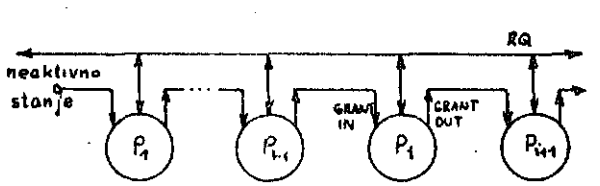


Slika 3.2.1. Skupno časovno deljeno vodilo.

Ves pretok podatkov gre izključno preko skupnega vodila. Zato je potrebno preprečiti, da bi v danem trenutku uporabljalo vodilo več procesorjev. V ta namen je potrebno kontrolno vezje, ki na zahtevo dodeljuje vodilo po vnaprej definirani politiki (fiksne različne prioritete, ciklično, ipd). Ko procesor pridobi vodilo zase, ga lahko uporablja le v času enega pomnilniškega cikla. Gre torej za časovni multipleks. Dodajanje procesorjev pomeni s stališča posameznega procesorja zmanjševanje propustnosti skupnega vodila oz. navidezno upočasnevanje pomnilnikov. Poleg razreševanja omenjenih konfliktov mora obstojati še zapora (lock), ki omogoča implementacijo nedeljive operacije testiranja in setiranja (test and set) določenih spremenljivk ali zastavic, ki jih uporabljamo pri vzajemnem izključevanju doseganja skupnih podatkovnih struktur in sinhronizacijskih operacij.

Za razreševanje konfliktnih situacij pri doseganju vodila je na razpolago več metod: serijska (daisy chain), s centralnim koordinatorjem, samoizbirna (self select), idr.

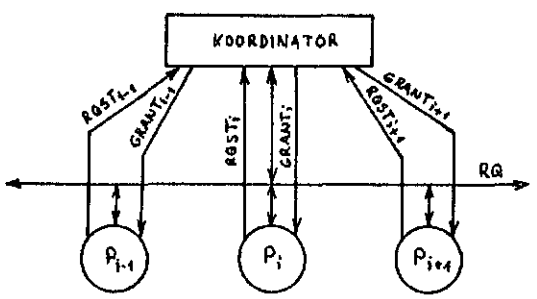
a) Serijska metoda (daisy chain). Aktivni moduli, priključeni na vodilo, so serijsko povezani s potrditveno linijo BUSGRANT (slika 3.2.1.a) preko GRANTIN vhodov in GRANTOUT izhodov.



Slika 3.2.1.a. Serijska metoda.

Preko te linije izve modul ali zahteva vodilo kakšen modul z višjo prioriteto. Dvosmerna linija za zaseganje in sproščanje vodila je paralelno priključena na vse module. Uporabnik vodila drži to linijo v aktivnem stanju. Vsi potencialni uporabniki pa jo testirajo. Doseganje vodila gre po naslednjem protokolu. Modul aktivira GRANTOUT izhod in s tem informira vse module nižje prioritete (procesorji so povezani po padajoči prioriteti), da se ne smejo udeleževati tekme za vodilo. Nato testira GRANTIN vhod in status linije za zaseganje vodila (RQ). Ko preide vhod GRANTIN v neaktivno stanje in linija RQ ni aktivna, zasede vodilo z aktiviranjem linije RQ. Obenem pa deaktivira GRANTOUT izhod. Moduli, ki ne potrebujejo vodila samo preslikujejo stanje GRANTIN vhoda na GRANTOUT izhod.

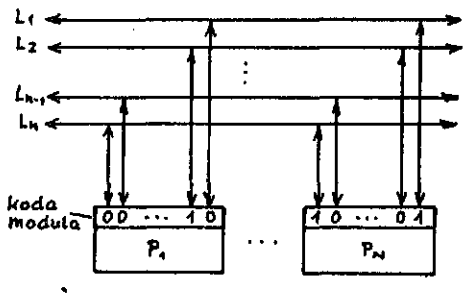
b) Centralni koordinator. V tem primeru ima vsak modul po dve privatni liniji: eno za zahtevo po vodilu (RQST) in eno za odobritev uporabe vodila (GRANT). S tem se je podrl koncept čistega vodila (vsi moduli priključeni na vse linije). Za zasedanje in sproščanje vodila je na razpolago dvosmerna linija RQ, podobno kot pri prejšnji metodi.(Slika 3.2.1.b).



Slika 3.2.1.b. Centralni koordinator.

Modul obvesti koordinator preko RQST linije, da želi uporabljati vodilo. Koordinator nadzira zasedenost vodila (RQ linija) in pregleduje zahteve. Ko se vodilo sprosti, izbere po določenem algoritmu (zavisi od razvrščanja) enega od modulov in mu pošlje signal GRANT. Modul, ki sprejme potrditev aktivira linijo RQ in se priključi na vodilo. Centralni koordinator ima pregled nad vsemi aktivnimi moduli. Zato je mogoče uveljaviti poljuben režim razvrščanja v primeru več hkratnih zahtev po vodilu. Poleg tega je mogoče slediti rezultate razvrščanja v daljših intervalih (merjenje čakalnih časov, merjenje časa uporabe vodila za posamezne module) in v skladu z globalno politiko korigirati razvrščanje (dinamično spreminjanje prioritete).

c) Samoizbirna metoda (self select method). Vodilo je razširjeno z n dodatnimi dvosmernimi linijami. Po teh linijah je mogoče prenašati 2^n različnih uporabnih kod. Vsakemu modulu pripada ena od teh kod, ki skriva v sebi tudi prioriteto modula. V času uporabe vodila je na teh dodatnih linijah koda modula, ki zaseda vodilo. Pri iskanju novega uporabnika vodila se tudi uporabljajo te linije (slika 3.2.1.c).
Ko se pojavi signal, ki napoveduje začetek izbiranja novega uporabnika vodila, postavijo vsi moduli, ki potrebujejo vodilo svoje kode na vodilo za samoizbiro (LI-LN). To vodilo izvrši ALI funkcijo (wired OR). Sedaj pogleda vsak modul najvišji bit na vodilu in ga primerja s svojo kodo. Če se bita ne ujemata, umakne svojo kodo z vodila. To se izvrši bit za bitom do zadnjega bita kode. Na vodilu ostane koda modula, ki je bil izbran za naslednjega uporabnika vodila. Serijsko primerjanje lastne kode s tisto na vodilu LI-LN poteka relativno hitro, saj ga lahko izvedemo s hitrimi TTL vezji.

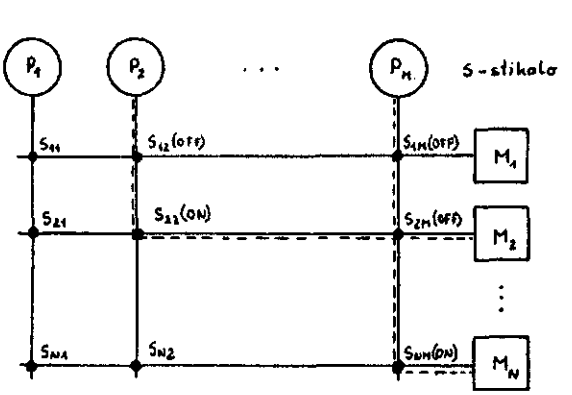


Slika 3.2.1.c. Samoizbirni mehanizem.

Skupno vodilo ima nekaj prednosti pred ostalimi vrstami povezovalnih elementov v multiprocesorskih sistemih. Prva je enostavnost in s tem nizka cena. Omogoča tudi dobro fleksibilnost sistema, ki se odraža v enostavnosti širjenja. Po drugi strani pa ravno vodilo predstavlja potencialno ozko grlo sistema. Z dodajanjem novih procesorjev se veča zmogljivost celega sistema (povečana procesna moč), zmanjša pa se zmogljivost vsakega posameznega procesorja (hitrost dostopa do pomnilnika in s tem hitrost izvajanja programa se zmanjša). Že sama arhitektura nam prinaša pojav zasičenosti (minimalno povečanje ali celo znižanje zmogljivosti sistema z dodajanjem novih procesorjev). S stališča integritete pa je ta struktura skrajno neugodna, ker ravno vodilo, ki je srce sistema ni redundantno. Vsak izpad vodila vodi v katastrofo - izpad celega sistema. Nekaj tolažbe nam daje dejstvo, da je vodilo zaradi enostavnosti zelo zanesljivo.

3.2.2. Matrika stikal - "crossbar".

Največja hiba opisane konfiguracije s skupnim vodilom je upadanje propustnosti vodila z dodajanjem novih procesorjev. Vodilo je namreč predstavljalo edini komunikacijski kanal v sistemu. To hibo odpravimo tako, da aktivne module (procesorje) in pasivne module (pomnilniki, V/I naprave) povežemo z matriko stikal. Matrika ima dimenzije MxN, če je M število aktivnih in N število pasivnih modulov (slika 3.2.2).



Slika 3.2.2.1.Matrika stikal.

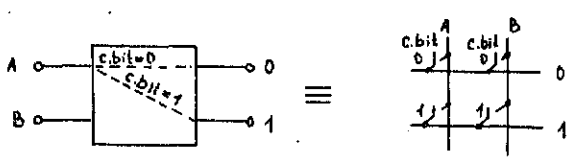
Pri taki strukturi multiprocesorskega sistema lahko poteka več komunikacij istočasno, če se le med seboj izključujejo (vsak pasivni modul dosega samo en procesor). Vsaka linija na gornji sliki predstavlja vodilo in stikalo preklaplja celo vodilo. Stikala morajo tudi preprečevati hkratne dostope do istega pasivnega modula (v vsakem trenutku je lahko vključeno samo eno stikalo) in v primeru več hkratnih zahtev morajo uveljaviti določeno politiko razvrščanja. Iz nalog stikal in strukture povezav vidimo, da je matrika stikal zelo kompleksen hardwarski modul. Za ilustracijo vzemimo primer 24 32bitnih procesorjev in 32 pomnilniških modulov (citiran v (ENSLOW77)). Za tak sistem vsebuje matrika stikal trikrat toliko integriranih vezij kot jih vsebuje računalnik IBM 360/75.

Sistemi z matriko stikal se odlikujejo po veliki propustnosti, saj je dovoljeno več hkratnih komunikacij. Sistem se da širiti do dimenzij matrike na zelo enostaven način. S stališča integritete so taki sistemi boljši od sistemov s skupnim vodilom. Med slabe strani pa štejemo visoko ceno, ki izvira iz kompleksnosti. Širjenje sistema preko dimenzij matrike je praktično nemogoče.

Zelo zmogljiva LSI in VLSI vezja nam lahko vlivajo malo optimizma, da bo s primernimi vezji omenjena kompleksnosti mogoče graditi "crossbar" stikala na enostavnejši način. Precej napora se vlaga tudi v raziskave modularne gradnje teh stikal, ki bi odpravila visoko začetno ceno in nepremagljivo mejo dimenzij matrike. Poglejmo si eno tako alternativno rešitev, ki se odlikuje z modularnostjo (PATEL79).

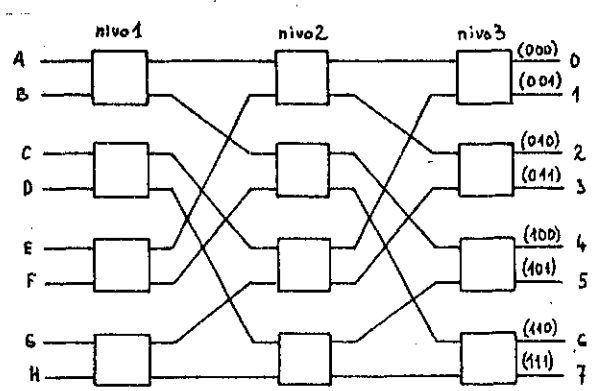
Delta mreže

Osnovna ideja je v tem, da z ustrezno povezavo majhnih matrik stikal ("crossbar" 2x2) gradimo poljubno velika povezovalna mreže, ki omogočajo več khratnih neizključujočih se povezav. Vzemimo za osnovo matriko 2x2 (slika 3.2.2.2). Kontrolni bit na vhodu pove na kateri izhod naj se vhod poveže.



Slika 3.2.2.2."Crossbar" vezje 2x2.

S pomočjo te osnovne matrike lahko zgradimo večjo povezovalno mrežo drevesaste oblike. Povezava se formira na podlagi kontrolnih bitov za vsak nivo en bit.

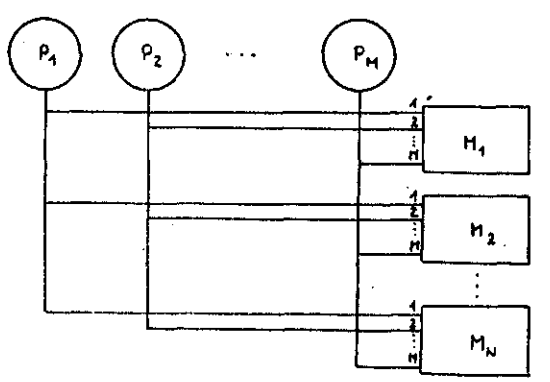


Slika 3.2.2.3.Delta mreža 8x8.

Za delta mrežo 8x8, prikazano na sliki 3.2.2.3 so potrebni trije nivoji in s tem trije kontrolni biti. Propustnost take mreže je nekoliko nižja kot pri "crossbar" matriki. Če je npr. A povezan z izhodom 4, potem: B ne more biti istočasno povezan na nobenega od izhodov od št.4 naprej in F ne more biti povezan na izhoda 4 in 5. Pri "crossbar" matriki je edina omejitev, da dva vhoda ne smeta biti istočasno povezana z istim izhodom. Delta mreže prekašajo "crossbar" matriko po ceni šele od dimenzije 16x16 naprej. V (PATEL79) je poleg opisa, kako konstruiramo delta mreže, še primerjalna analiza delta mrež in "crossbar" matrike glede na ceno in propustnost.

3.2.3. Pomnilniki z več vrati

Pri prejšnji strukturi smo videli, da smo za doseženo paralelnost komunikacij drago plačali s precejšno kompleksnostjo povezovalne matrike in slabo fleksibilnostjo. Če hočemo poenostaviti povezovalni element, moramo ustrezno modificirati ostali del sistema. Gre predvsem za modifikacijo pomnilniških modulov, saj ravno ti predstavljajo deljene vira sistema. Pomnilnike opremimo z več vrati - vhodi na katere lahko priključimo vodila. Krmilno vezje skrbi za to, da lahko v vsakem trenutku samo en vhod dosega pomnilnik in določa kateri vhod bo to, če se pojavi zahteva na več vhodih hkrati. Vso kontrolno logiko smo torej potisnili od povezovalnega elementa na pomnilniške module. Povezovalni element se sestoji sedaj samo še iz množice vodil (slika 3.2.3.1).



Slika 3.2.3.1.Sistem s pomnilniki z več vrati.

S primernim povezovanjem lahko postanejo nekateri pomnilniki privatni za posamezen procesor ali za posamezno skupno procesorjev. Opisana struktura ima to dobro lastnost, da je možnih več paralelnih komunikacij. Širjenje sistema je enostavno, dokler ne zasedemo vseh vrat na pomnilniških modulih. Med slabe strani pa lahko ponovno štejemo precejšno kompleksnost (tokrat na pomnilniških modulih), slabo fleksibilnost in veliko število povezav ter konektorjev.

3.3. Vrsta povezav

V sistemih z več procesorji se pojavljata dve vrsti prenosa podatkov in s tem dve vrsti povezav: serijske in paralelne. Običajno zasledimo paralelni prenos podatkov v tesno povezanih sistemih, serijski pa v ohlapno povezanih sistemih. Vrsto povezav določa cena, narava aplikacije (v letalih je zelo važna teža), ipd.

4. SISTEMSKA KONTROLA

Sistemsko kontrolo sestavljajo programi in hardwarske komponente, ki povezujejo aktivne in pasivne module v smiseln sistem. Ponavadi imenujemo programski del sistemske kontrole operacijski sistem v ožjem smislu besede. Hardwarski del (arbiter zaseganja vodila, implementacijo nedeljive operacije "test and set", implementacijo komunikacijskih protokolov, ipd) pa predstavlja bolj ali manj uporabno osnovo za implementacijo operacijskega sistema. Ta najnižji nivo pojava sistemske kontrole označujemo kot arhitektonske poteze sistema. Nekaj smo o teh potezah že slišali v prejšnjem poglavju, zato se bomo osredotočili na t.im. operacijski sistem oz. izvršnik (executive).

Izvršnik uporablja osnovne operacije jedra in omogoča vodenje izvajanja - izvrševanja procesov na sistemu. Operacijski sistem pa vsebuje še druge funkcije, ki niso vezane izključno na izvajanje - "run time". Razliko med njima bi lahko bolj drastično označili na naslednji način. Izvršnik predstavlja potreben pogoj, da sistem sploh funkcionira, operacijski sistem pa omogoča uporabniku, da sistem bolj ali manj udobno in učinkovito uporablja. Ko smo ravno pri definiciji, si poglejmo še definicijo jedra. Jedro je množica primitivnih operacij, ki predstavljajo implementacijo mehanizmov potrebnih pri uporabi skupnih virov, komunikaciji ter kreiranju in brisanju procesov (npr. operacije: wait, send, P, V, ipd). Izvršnik uporablja te mehanizme za oblikovanje politike, ki je za konkretno zgradbo sistema in za konkretno aplikacijo najbolj primerna. Tu mislimo na politiko uporabe skupnih virov, razvrščanja procesov po procesorjih, itd.

Izvršnik mora zagotavljati:

- regularno rabo skupnih virov
- medprocesno komunikacijo
- preprečevanje mrtvih točk (dead lock)
- odkrivanje, lokalizacijo in odpravljanje napak

Naštete cilje je mogoče doseči na dva načina:

- s centralizirano kontrolo
- z distribuirano kontrolo

Centralizirana kontrola je takšna kontrola, pri kateri odloča o vsakem skupnem viru en sam objekt ("test-and-set" zastavica, semafor, monitor). Status sistema je dostopen vsem procesorjem in je vedno ažuren. Glede na fizično razpršenost globalne baze podatkov in način doseganja (pomnilniška referenca, poziv monitorske procedure, sporočilo), klasificiramo kontrolo kot bolj ali manj centralizirano oz.decentralizirano.

Distribuirana kontrola predstavlja ravno obraten pristop. Vsak skupni vir kontrolira več objektov - kontrolerjev. To pravilo mora veljati na vseh nivojih abstrakcije. V sistemu nimamo globalne baze podatkov. Na več mestih se formirajo "privatne" baze, ki so bolj ali manj neažurne. Zato prihaja do kontradiktornih delnih odločitev. Končne odločitve se formirajo po večinskem ali kakšnem drugem principu. LeLann (LELANN77) definira distribuirane sisteme kot sisteme, v katerih si sodelujoče komponente ne delijo fizičnega prostora (skupni pomnilnik) in/ali nimajo skupne časovne reference.

Centralizirana kontrola predstavlja klasični pristop, ki je bil uporabljen že v monoprocesorskih sistemih. To vrsto kontrole je mogoče podrobneje deliti še na tri primere:

- relacija gospodar-suženj (master-slave)
- ločeni izvršniki (skupne sistemske tabele)
- simetrična organizacija

Gospodar - suženj. Rutine izvršnika se izvajajo vedno na istem procesorju - gospodarju. Če potrebuje suženj kakšno uslugo, kliče gospodarja in čaka dokler gospodar ne prekine trenutne dejavnosti in začne izvajati ustrezno rutino. Če dodelimo procesor samo za izvajanje izvršnika, se relacija obrne. Sedaj predstavlja ta procesor sužnja, ki čaka, da mu bo eden od potencialnih gospodarjev posredoval nalogo. Take vrste sisteme je mogoče najenostavneje implementirati. Procesor, ki izvaja izvršnik, mora biti dovolj zmogljiv, da ne zavira ostalega sistema (BUHR78). S stališča integritete sistema je ta primer najneugodnejši, saj predstavlja najbolj centralistični primer (nimamo nobene redundantnosti). Izpad procesorja, ki izvaja izvršnik povzroči izpad celega sistema.

Ločeni izvršniki. Vsak procesor ima svoj izvršnik, ki servisira lokalne zahteve. Globalne sistemske tabele omogočajo koordinacijo lokalnih izvršnih sistemov. Sistem je razbit na bolj ali manj avtonomne enote (procesor, pomnilnik, V/I naprave). Vsak lokalni izvršnik skrbi le za množico lokalnih virov. Rekonfiguracija V/I naprav zahteva intervencijo operaterja. Ker ni globalnega izvršnika, ki bi bdel nad vsemi viri sistema, je potrebno npr. V/I napravo fizično preklopiti na modul oz. lokalni izvršnik, kjer se je pojavila zahteva po taki napravi. V tem primeru gre za bolj decentralizirano kontrolo, ki je tipična za multiračunalnike. Sistem s tako kontrolo je manj občutljiv na izpade, ker je kontrola razpršena po sistemu.

Simetrična organizacija. Procesorji in vsi ostali viri sistema predstavljajo anonimno množico virov. Izvršnik ni fizično razpršen po sistemu niti ga ne izvaja en sam vnaprej določen procesor. Predstavlja množico deljenih rutin. Procesorji si med seboj podajajo vlogo gospodarja, ki jo prevzamejo, ko izvajajo kakšno rutino izvršnika. Večina rutin mora omogočati ponoven vstop (reentry) tako, da jo lahko izvaja več procesorjev hkrati. Ta organizacija vsebuje prednosti obeh prej naštetih. S tem, da nadzoruje vse vire sistema, podobno kot pri organizaciji "gospodar-suženj" ni problemov pri rekonfiguraciji sistema. Po drugi strani pa je zelo ugodna s stališča integritete, ker ob izpadu posameznega procesorja ostane sistem še živ, saj lahko poljuben procesor izvaja funkcije izvršnika - omogočena je uporaba vseh virov sistema razen izpadlega procesorja.

4.1. Sinhronizacijski mehanizmi

Doslej smo omenili dve osnovni vrsti sistemske kontrole v sistemih z več procesorji: centralizirano in distribuirano. Ti se razlikujeta že v osnovnih mehanizmih. Zato si bomo podrobneje ogledali te mehanizme, ki so za centralizirano kontrolo že dokaj popularni, za distribuirano pa še manj znani.

4.1.1. Centralizirana kontrola

Mehanizmi, na katerih sloni centralizirana kontrola, se poslužujejo centralne podatkovne baze sistema. Operacije s katerimi so ti mehanizmi implementirani spreminjajo to podatkovno bazo. To so s stališča sistema najbolj kritične operacije, zato morajo biti nedeljive in med seboj izključujoče. Sinhronizacijski mehanizmi so

običajno implementirani v več nivojih (monitorske procedure uporabljajo za interno sinhronizacijo signale, medsebojno izključenost monitorskih procedur implementiramo s semaforom, itd). Temelji za tako nivojsko strukturo morajo biti implementirani že v hardwaru (npr. nedeljiva "test and set" operacija). Sinhronizacija je potrebna pri doseganju skupnih virov (podatki, procedure, V/I naprave, ipd) in pri komunikaciji med procesi. V prvem primeru gre za dosego vzajemnega izključevanja (mutual exclusion), oz. za serializacijo paralelnih procesov pri uporabi skupnih - deljenih virov. Sinhronizacija je torej pogojena z logično (komunikacija) in fizično (uporaba skupnih virov) odvisnostjo procesov. Procesi morajo testirati globalne pogoje (npr. zasedenost naprave), ki jih predstavlja globalna podatkovna struktura (semafor, lokalne monitorske spremenljivke, itd.). Procesi so organizirani v vrste: vrsta procesov pripravljenih na izvajanje, vrste blokiranih procesov. Tudi te vrste sestavljajo globalno podatkovno bazo sistema (v skupnem pomniniku). Dostop do teh podatkov pa mora biti omejen le na sinhronizacijske operacije in na druge operacije jedra. V nadaljevanju si bomo ogledali signale, semafore in monitorje, kot tipične predstavnike centralizirane kontrole.

Signali

Signali so predstavniki vrst procesov. Deklariramo jih kot spremenljivke. Nad temi vrstami so definirane tri osnovne operacije: wait(signal), send(signal) in awaited(signal). Te operacije so implementirane v jedru. Z njimi se procesi sinhronizirajo.

Operacija wait(s) postavi klicujoči proces v vrsto na signal s. Proces se ustavi.

Operacija send(s) odstrani en proces iz čakalne vrste na signal s in ga naniza v vrsto procesov pripravljenih na izvajanje. Če je vrsta na signal s prazna, operacija send ne stori ničesar.

Operacija awaited(s) nam pove ali je vrsta na signal s prazna ali ne.

Operaciji wait in send nam torej omogočata ustavitev in nadaljevanje procesa. Signal nima pomnilne lastnosti. Če pošljemo signal prej, preden ga je ciljni proces pripravljen sprejeti (wait), se bo signal izgubil. Ta lastnost predstavlja potencialno nevarnost mrtve točke. Zato običajno navežemo pošiljanje signala na predhodno testiranje obstoja vrste (awaited). Signal pošljemo šele, ko vrsta ni prazna.

Politika razvrščanja procesov zavisi od implementacije. Operacija send(s) lahko npr. aktivira celo vrsto blokiranih procesov, ki so čakali na signal s, lahko pa seveda jemlje enega po enega iz vrste po predpisanem algoritmu. Poglejmo, kako dosežemo s signali vzajemno izključevanje pri uporabi skupnih virov (slika 4.1.1.1).

```
1       while busy do wait(s);
2       busy:=true;
            :
        uporaba vira
            :
I       busy:=false;
I+1     send(s);
```

Slika 4.1.1.1. Vzajemno izključevanje pri uporabi skupnih virov.

Rešitev na sliki 4.1.1.1 velja le za monoprocesorski sistem, kjer je v vsakem trenutku aktiven (se izvaja) samo en proces. Skupna spremenljivka "busy" predstavlja status skupnega vira (zaseden ali prost). Dokler vir ni prost, se vsi procesi, ki želijo uporabljati vir, blokirajo in nanizajo v čakalno vrsto na signal s. Ko proces, ki uporablja vir, sprosti vir, omogoči enemu od čakajočih procesov uporabo vira. V multiprocesorskem okolju bi prišlo do neregularne uporabe vira. Preprečimo jo lahko tako, da implementiramo stavka 1 in 2 ter stavka I in I+1 kot nedeljivi in izključujoči se operaciji. Kot bomo videli kasneje sta to ravno operaciji P in V nad binarnim semaforom.

Semafori

Za razliko od signalov imajo semafori pomnilno lastost. Signale lahko smatramo za nekoliko enostavnejše konstrukte, saj lahko implementiramo semafore s signali. Nad semaforom sta definirani dve operaciji:

- V(sem): poveča vrednost semafora sem za 1.
- P(sem): zmanjša vrednost semafora sem za 1, če bo rezultat nenegativen. Sicer se operacija ne more zaključiti.

Obe operaciji sta nedeljivi. P operacija omogoča podobno kot prej opisana wait operacija ustavitev oz.zakasnitev procesa. Tudi na semafore se navezujejo vrste. Če se P operacija ne more zaključiti (vrednost semafora bi postala negativna), gre proces v čakalno vrsto, vezano na ta semafor. Šele V operacija nad istim semaforom omogoča nadaljevanje tega procesa. Semafor si zapomni vsako V operacijo. Semafor, ki lahko zavzame samo dve vrednost ("true", "false"), imenujemo binarni semafor. Običajno ga uporabljamo za dosego vzajemnega izključevanja uporabe skupnih virov (slika 4.1.1.2).

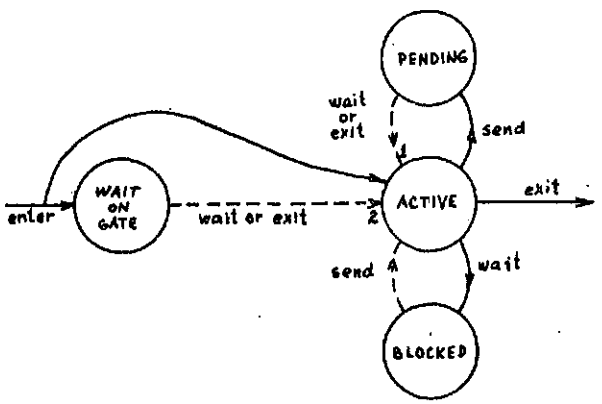
```
P(sem);
   :
uporaba vira
   :
V(sem);
```

Slika 4.1.1.2. Vzajemno izključevanje implementirano z uporabo semafora.

Monitori

Monitor je zbirka podatkov in procedur. V vsakem trenutku sme izvajati monitorsko proceduro same en proces. Pravimo, da se sme samo en proces nahajati v monitorju. Vstop v monitor (doseganje lokalnih podatkov) je možen le preko monitorskih procedur, zato jih imenujemo tudi vstopne procedure (entry procedure). V vsakem trenutku je lahko aktivna samo ena vstopna procedura (vzajemno izključevanje vstopov). Monitorji omogočajo boljše strukturiranje programov oz. procesov. Sinhronizacija, ki je potrebna pri doseganju skupnih virov - podatkov, je skrita oz. omejena na monitorske procedure. Ta sinhronizacija je običajno realizirana s signali. V monitorskih procedurah bosta torej operaciji wait in send skrbeli za pravilno odvijanje teh procedur. Tu se bo pojavila vrsta procesov, ki bodo obtičali v monitorju (z wait operacijo v monitorski proceduri). Vsak proces, ki se zablokira, sprosti monitor. Vzajemno izključevanje vstopnih procedur je implementirano z vhodnim semaforom (monitor gate semaphore). Tu se pojavi vrsta procesov, ki čakajo na vstop v monitor. Monitorji se razlikujejo

med seboj po načinu razvrščanja procesov. Podroben opis je mogoče najti v (EXEL80).



——→ *prehod iz aktivnega stanja, ki ga povzroči klic aktivnega procesa.*

– – → *prehod iz stanja čakanja, ki ga povzroči klic drugega - aktivnega procesa.*

Slika 4.1.1.3. Hoarov tip monitorja.

Na sliki 4.1.1.3 je podana politika razvrščanja za Hoarov monitor v obliki končnega avtomata. Proces, ki vstopi v monitor, ostane aktiven, če je vhodna vrsta prazna, sicer pa se uvrsti v vhodno vrsto. Z operacijo wait v monitorski proceduri preide proces v neaktivno stanje (vrsta blokiranih procesov). Z operacijo send preide proces v posebno vrsto procesov, ki so "izviseli" v monitorju (PENDING - stanje). S tem sprosti monitor in omogoči procesu, ki je čakal na signal (vrsta blokiranih procesov), da konča monitorsko proceduro. Ob vsakem izstopu iz monitorja in ob vsaki wait operaciji se monitor sprosti. Pri izbiri novega procesa imajo prednost procesi iz nujne vrste (so v PENDING stanju) pred tistimi iz vhodne vrste. Monitor za dosego vzajemnega izključevanja pri uporabi skupnega vira izgleda tako:

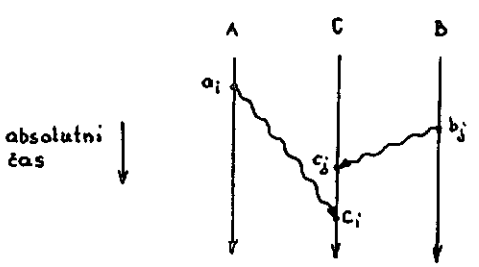
```
skupni_vir monitor
begin busy: Boolean;
      nonbusy: signal;
   procedure zasedi
     begin
       if busy then wait(nonbusy);
       busy:=true
     end
   procedure sprosti
     begin
       busy:=false; signal(nonbusy)
     end
   busy:=false;   začetna vrednost
end skupni_vir
```

Slika 4.1.1.4. Monitor, ki omogoča regularno uporabo skupnega vira.

4.1.2. Distribuirana kontrola

Distribuirani sistem se sestoji iz zbirke procesov, ki so prostorsko ločeni in ki komunicirajo z izmenjavo sporočil (LAMPORT78). Ta Lamportova definicija je nekoliko ožja od LeLannove (LELANN77), ki eksplicitno poudarja še možnost odsotnosti skupne časovne reference. Dejansko so distribuirani sistemi tudi brez skupne časovne reference. To referenco pa moramo vpeljati v sistem, če hočemo, da bo sistem pravilno deloval. V distribuiranih sistemih čas, ki je potreben za prenos sporočila, ni zanemarljivo majhen. V večini primerov tudi ne poznamo maksimalnega časa, ki ga lahko sporočilo porabi za svojo pot (težave pri detekciji izgube sporočila). Procesi, ki se odvijajo na posameznih procesnih elementih (procesni element je par procesor - pomnilnik) so sestavljeni iz dogodkov. Ti dogodki so urejeni (relacija "prej" oz. "pozneje"). Za vsak par dogodkov je mogoče povedati kateri se je oz. se bo zgodil prej. Vseh dogodkov sistema pa ne moremo urediti s to relacijo, ker ni enotne časovne reference. Cilj sinhronizacijskih mehanizmov je totalna ureditev dogodkov sistema, ki v vsakem trenutku zagotavlja konsistentno stanje sistema.

Ilustrirajmo problem z zgledom. Imejmo tri procese A, B in C. Prva dva naj uporabljata datoteko X. Vse operacije nad datoteko X izvaja proces C. Ker v sistemu ni enotne časovne reference in ker ne vemo koliko časa potujejo sporočila, lahko pride do naslednjega primera. Dogodek a_i (proces A je poslal sporočilo procesu C, s katerim zahteva operacijo nad datoteko X) se je zgodil pred dogodkom b_j (proces B je poslal sporočilo procesu C, s katerim zahteva operacijo nad datoteko X). Zaradi različnih časov prenosa lahko pride v procesu C prej do dogodka c_j (sprejem sporočila iz procesa B) kot do dogodka c_i (sprejem sporočila iz procesa A). Če torej proces C servisira zahteve po vrstnem redu prispetja, lahko pride do nekonsistentnega stanja sistema. Grafično je ta primer nakazan na sliki 4.1.2.1.



Slika 4.1.2.1

Zahteve se ne servisirajo po vrstnem redu, kot se pojavljajo v sistemu. Če vzamemo, da pomeni dogodek a_i zahtevo po čitanju datoteke in dogodek b_j brisanje datoteke, potem je jasno, da je vrstni red servisiranja zelo važen. Vpeljati moramo mehanizme, ki nam bodo omogočili, da bomo lahko sporočila ob pošiljanju opremili z absolutnim časom all zaporedno številko. Sinhronizacijski mehanizmi bazirajo torej na logičnih all fizičnih urah, ki žigosajo sporočila ob oddaji. To omogoča sprejemniku urejevanje sporočil po času oddaje oz. urejevanje zahtev po servisiranju po času pojava teh zahtev v matičnih procesih.

LeLann se pri karakterizaciji distribuiranih sistemov poslužuje nekoliko drugačne terminologije. Namesto pošiljanja in sprejemanja sporočil govori o produkciji in opazovanju dogodkov. Dogodki so posledica posameznih operacij procesa. V distribuiranih sistemih sta s stališča opazovalca produkcija in materializacija (manifestacija) dogodka časovno ločeni. Potreben je določen čas, da se produkcija manifestira kot materializacija v opazovalčevem procesu (sprejem sporočila). V konvencionalnih sistemih pa razlike med produkcijo in materializacijo ni (zapis v skupni pomnilnik).

Sinhronizacijske mehanizme je mogoče deliti v dve skupini: centralizirane in decentralizirane. Centralizirani so tisti pri katerih ima sinhronizacijski element enolično ime, ki ga poznajo vsi procesi, ki se med seboj sinhronizirajo in ki imajo ob vsakem času dostop do tega elementa. Centralizirani mehanizmi so: fizična ura, števci dogodkov (eventcount) in statični zaporednik (static sequencer). Decentralizirani mehanizmi pa so: več fizičnih ali logičnih ur, paroma deljene skupne spremenljivke, krožeči znak (circulating token) in krožeči zporednik (circulating sequencer).

Centralizirani mehanizmi

Pri vseh mehanizmih gre za to, da imamo centralno mesto, ki daje absoluten čas ali dell zaporedne številke. Ti mehanizmi zelo spominjajo na sinhronizacijske mehanizme v klasičnih sistemih (semafori, signali). Imajo tudi vse slabosti omenjenih mehanizmov. Komunikacijski sistem po katerem dobivajo procesi absolutni čas mora biti zelo zanesljiv. Poleg tega moramo dobro poznati zakasnitve, ki jih prinaša potovanje sporočil. V primeru fizične ure, posamezni procesi čitajo vrednost ure in z dobljeno vrednostjo žigosajo dogodke (timestamp).

Drugi mehanizem temelji na števcih, ki štejejo dogodke posameznih vrst. Nad temi števci so definirane elementarne operacije:

- advance(E): povečaj vrednost E-ja za 1. (E je števec dogodkov)
- read(E): povej vrednost E-ja. Dobimo spodnjo vrednost E-ja po "read" operaciji in zgornjo vrednost E-ja pred "read" operacijo.
- await(E,v): suspendiraj kličujoči proces, dokler ni vrednost E-ja vsaj enaka vrednosti v.

Te operacije se lahko izvajajo paralelno brez zahteve po vzajemnem izključevanju. Omogočajo nam oštevilčevanje sorodnih dogodkov v različnih procesih (zahteve po določeni vrsti uslug). Tako bo servisni proces sprejemal oštevilčene zahteve, ki pa seveda ne bodo prihajale po vrstnem redu. Z operacijo "await" bo lahko počakal na novo zahtevo in odbijal zahteve, ki želijo prehitevati.

Tretji mehanizem temelji na zaporedniku, ki je implementiran kot spremenljivka naravnih nepadajočih vrednosti. Definirana je elementarna operacija:

ticket(S): vrni vrednost zaporednika S in mu povečaj vrednost za 1.

Ta operacija mora biti implementirana tako, da je nedeljiva in da je zagotovljeno vzajemno izključevanje procesov pri izvajanju te operacije. Ta mehanizem je podoben prvemu. Dogodkom delimo žetone, ki omogočajo globalno urejevanje dogodkov.

Decentralizirani mehanizmi

Fizične ure. V sistem želimo vpeljati čas, ki nam bo omogočal kronološko urejevanje dogodkov oz.akcij. Posamezne ure morajo biti točne in med seboj morajo biti sinhronizirane. Če označimo uro s C in zvezni čas s t potem morata veljati pogoja:

1. Za vsak i mora obstajati $\varkappa \ll 1$ tako da velja:

$$\left| \frac{dC_i(t)}{dt} - 1 \right| < \varkappa$$

2. Za vse i,j mora veljati:

$$|C_i(t) - C_j(t)| < \varepsilon$$

Ker je jasno, da ni dveh ur, ki bi šle enako, se s časom razlike večajo. Za izpolnitev drugega pogoja je potrebno občasno popravljati vrednost ur. Ure vedno pomikamo naprej, nikoli nazaj. V (LAMPORT78) je na modelu podana rešitev, kako je potrebno sinhronizirati ure.

Logične ure. Logična ura C_i je funkcija, ki priredi vsakemu dogodku procesa P_i eno številko $C_i(a)$ (a je dogodek). Funkcija C naj predstavlja cel sistem logičnih ur. Ta funkcija mora zadoščati posebnemu urnemu pogoju, ki omogoča *urejevanje dogodkov*. Ta pogoj je sledeč: Če se je zgodil dogodek a pred dogodkom b ($a \rightarrow b$), potem je $C(a) < C(b)$. Pri čemer relacija $a \rightarrow b$ pomeni:

1. a se je zgodil pred b. a in b sta dogodka istega procesa.
2. a pomeni pošiljanje, b pa sprejemanje istega sporočila.

Torej lahko urni pogoj preplšemo v novo obliko:

I. Če a in b dogodka procesa P_i in je a pred b potem sledi $C_i(a) < C_i(b)$.

II. Če pomeni a pošiljanje sporočila iz procesa P_i in b sprejem istega sporočila v P_j sledi: $C_i(a) < C_j(b)$.

Pogoj I izpolnimo tako, da vsak proces P_i poveča svojo uro C_i za 1 med dvema zaporednima dogodkoma. Pogoj II pa izpolnimo na naslednji način. Vsako sporočilo mora vsebovati časovni žig (timestamp) $T_m = C_i(a)$, ki ga "udari" dogodek a v procesu P_i. Po sprejemu tega sporočila mora postaviti proces P_j svojo uro C_j na vrednost, ki je enaka ali večja od trenutne in večja od T_m.

S takšnim sistemom logičnih ur je mogoče razširiti delno (lokalno) urejevanje dogodkov na totalno urejevanje, ki ga definira relacija $\Longrightarrow$. Definiramo jo tako:

za $a \in P_i$ in $b \in P_j$ velja $a \Longrightarrow b$, če velja:
1. $C_i(a) < C_j(b)$ ali
2. $C_i(a) = C_j(b)$ in $P_i < P_j$ ($<$ je relacija, ki urejuje procese med seboj).

Poglejmo si uporabo opisanega mehanizma pri reševanju problema vzajemnega izključevanja pri uporabi skupnih virov (primer je vzet iz (LAMPORT78)). Iščemo algoritem, ki bo zadovoljil naslednje tri pogoje:

a) proces, ki je dobil vir, ga mora sprostiti preden ga dodelimo drugemu procesu.

b) skupni vir moramo dodeljevati po istem vrstnem redu kot se javljajo zahteve zanj.

c) če vsak proces, ki mu je bil dodeljen vir, sprosti ta vir, potem bo vsaki zahtevi ugodeno.

Rešitev iščemo v sistemu ur, ki zadoščajo pogojema I in II (varianta urnega pogoja). Tako definiramo totalno ureditev ($\Longrightarrow$) vseh dogodkov (operacij sproščanja in zaseganja skupnega vira). Predpostavimo, da lahko vsak proces pošlje sporočilo vsem ostalim procesom, ki uporabljajo skupni vir. Poleg tega predpostavimo, da sprejemajo procesi sporočila po istem zaporedju kot so bila oddana in da se nobeno sporočilo ne izgubi. Ta predpostavka se nanaša na vsak proces

posebej in ne na cel sistem. Gre za to, da se sporočila istega proces ne smejo prehitevati. Vsak proces ima svojo vrsto zahtev, ki je popolnoma privatna. V vsaki je začetno sporočilo "T_0 : P_0 zahteva vir" (T_0 je manjši od katerekoli ure v sistemu: P_0 je proces, ki mu prvemu dovolimo uporabo vira). Algoritem predstavlja naslednjih pet pravil:

1. Ko proces P_i zahteva vir, pošlje vsem procesom sporočilo "T_m: P_i zahteva vir" in dá to sporočilo tudi v svojo vrsto zahtev (T_m je časovni žig sporočila).

2. Ko proces P_j prejme sporočilo "T_m: P_i zahteva vir", ga da v svojo vrsto zahtev in odgovori procesu P_i s potrditvenim sporočilom, ki ima tudi časovni žig.

3. Pri sprostitvi vira umakne P_i vsa sporočila "T_m: P_i zahteva vir" iz svoje vrste zahtev in pošlje vsem procesom žigosano sporočilo "P_i sprošča vir".

4. Če prejme proces P_j sporočilo "P_i sprošča vir", odstrani vsa sporočila "T_m: P_i zahteva vir" iz svoje vrste zahtev.

5. Procesu P_i je odobrena uporaba skupnega vira, če sta izpolnjena naslednja pogoja:
   - v vrsti zahtev je sporočilo "T_m: P_i zahteva vir" na prvem mestu. Vrsta je urejena z relacijo $\Longrightarrow$ (važen je čas oddaje sporočila, ne pa čas sprejema).
   - P_i je sprejel od vsakega procesa, ki sodeluje v tej igri sporočilo, ki ima kasnejši žig kot T_m.

Opisani algoritem zadošča pogojem a,b in c ter zagotavlja vzajemno izključevanje pri uporabi skupnega vira. Vsak proces lokalno izvaja opisani algoritem pri tem pa ne uporablja nobenih centralnih podatkov ali centralnega sinhronizacijskega elementa. Slaba stran algoritma pa je v tem, da morajo vsi procesi aktivno sodelovati pri vsaki zahtevi in sprostitvi vira. Izpad enega od procesov onemogoči nadaljno uporabo skupnega vira in predstavlja potencialno nevarnost za blokiranje sistema. Pojem izpada ima smisel le v kontekstu fizičnega časa, kajti sicer ne moremo ločiti izpada od presledka med dvema dogodkoma.

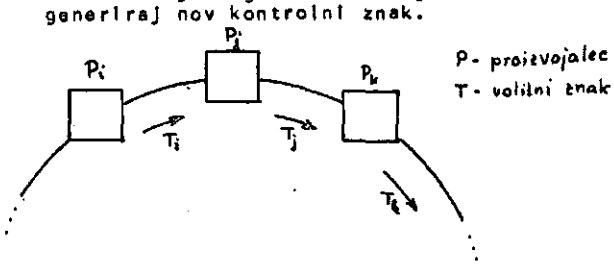
Kroženje privilegija

Sistem sestavljajo procesi, ki startajo operacije. Operacije so implementirane z množico akcij na nižjem nivoju. Procese, ki startajo operacije oz.akcije imenujemo proizvajalce; procese, ki izvajajo akcije pa imenujemo porabnike (consumer). Proizvajalci imajo enolične in stalna imena. Zato jih je mogoče poljubno urediti. Sistem sinhroniziramo tako, da dovolimo samo enemu proizvajalcu in sicer tistemu, ki je v privilegiranem stanju, startanje nove operacije. Vsako startanje operacije pomeni namreč sprožitev niza akcij, ki jim je treba dati zaporedno številko. Proizvajalce povežemo v logični obroč s tem, da vsakemu določimo oba soseda in da ima zadnji za soseda prvega. Privilegij kroži po tem navideznem obroču in zagotavlja vzajemno izključevanje pri startanju novih operacij.

Paroma deljene spremenljivke. Vsak proizvajalec deli s sosedom po eno skupno spremenljivko. Preko teh dveh izve ali je kateri od sosedov v privilegiranem stanju. Na podlagi opazovanja sosedov se vsak proizvajalec samoiniciativno odloči, da bo prešel v privilegirano stanje. Lahko se seveda zgodi, da je takih proizvajalcev več. Ta problem je podrobneje obdelan v (DIJKSTRA74).

Krožeči znak. Po navideznem obroču proizvajalcev kroži kontrolni znak (control token). Proces, ki trenutno poseduje ta znak lahko starta operacijo. Za pravilno delovanje je potrebno zagotoviti, da bo v obroču v vsakem trenutku samo en kontrolni znak in da je mogoče v primeru izpada enega od proizvajalcev rekonfigurirati obroč. Če izpade ravno proizvajalec, ki ima kontrolni znak, je potrebno delegirati proizvajalca, ki bo generiral novi kontrolni znak. V sistemu morajo biti časovniki, ki povedo, kdaj lahko smatramo kontrolni znak za izgubljen. Vsak proizvajalec ima tak časovnik, ki se resetira, ko prejme proizvajalec kontrolni znak. Poglejmo si protokol, ki zagotavlja detekcijo izgube kontrolnega znaka in njegovo restavracijo.

- kadar se časovnik izteče, generira proizvajalec volilni znak s svojim imenom. Proizvajalec je kandidat za generiranje novega kontrolnega znaka.
- če prejme kandidat kontrolni znak, preden je prišel njegov volilni znak naokrog po obroču, umakne svoj znak in prekliče volilno fazo. Ta del protokola imenujemo precedenčno pravilo.
- kadarkoli prejme kandidat volilni znak, si zapiše ime proizvajalca, ki ga je generiral (S(i) - lista kandidatov, ki jo pozna kandidat i).
- ko prejme kandidat lastni volilni znak, ga umakne iz obroča, resetira časovnik in izvede naslednji algoritem: če je i=min S(i), generiraj nov kontrolni znak.



Slika 4.1.2.2. Faza volitev

Dokaz, da dá opisano pravilo v končnem času en *sam kontrolni znak je podan v* (LELANN80).

Krožeči zaporednik. Po navideznem obroču kroži zaporednik z uporabo mehanizma krožečega znaka. Ko prejme proizvajalec kontrolni znak, lahko izvede poljubno število "ticket" operacij. S temi operacijami priskrbi žetone vsem akcijam, ki nanje čakajo. Nato pošlje kontrolni znak sosedu. Z metodo krožečega znaka smo dosegli vzajemno izključevanje operacij "ticket" nad zaporednikom. Žetoni so seveda označevalci s trenutno vrednostjo zaporednika, ki poveča svojo vrednost pri vsaki operaciji "ticket". Torej nosi vsak žeton drugo številko. Porabniki lahko v skladu z definirano strategijo vpeljujejo totalno urejevanje na podlagi žetonov, ki so jih prinesle s seboj posamezne akcije (akcije določene operacije imajo zaporedne številke). Konfliktne situacije se rešujejo tako, da se daje prednost akcijam z žetoni, ki imajo nižjo vrednost.

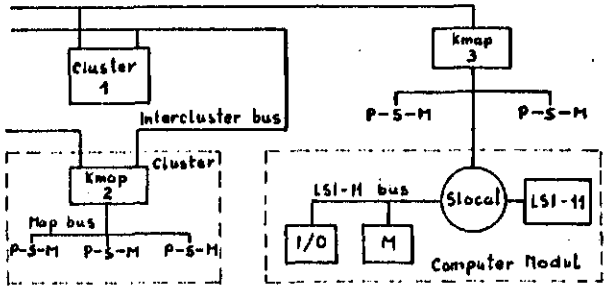
Iz tega kratkega pregleda mehanizmov za sinhronizacijo v distribuiranih sistemih vidimo, da so s stališča integritete sistema najbolj ugodni decentralizirani mehanizmi. Ti predstavljajo dejansko nove koncepte, ki niso na silo privlečeni iz klasičnih centraliziranih sistemov.

5. DVA PRIMERA SISTEMOV Z VEČ PROCESORJI

5.1.Cm*

Zgradba

Multiprocesorski sistem Cm* so razvili na Carnegie-Mellon univerzi. Zgradili so ga predvsem za raziskovalne namene. Gre za tesno povezan sistem s skupnim pomnilnikom. Skupni pomnilnik ni realiziran kot posebni modul. V vsakem modulu je poleg procesorja še pomnilnik. Vsak procesor lahko dosega vsak pomnilniški modul v sistemu. Struktura sistema je prikazana na sliki 5.1.1.

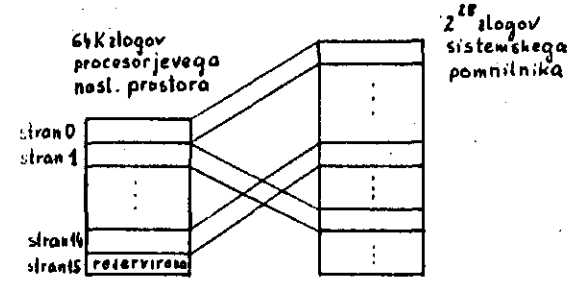


Slika 5.1.1.Struktura Cm* sistema

Osnova sistema je računalniški modul (Computer Module - Cm). Sestavljajo ga: procesor, pomnilniki in V/I naprave ter lokalno stikalo (Slocal). To stikalo preklaplja vodilo procesorja na interno vodilo all pa na preslikovalno vodilo (Map bus). Preslikovalno vodilo povezuje računalniške module v grupe (cluster). V grupo spada še Kmap modul. Ta je sestavljen iz treh delov: kontrolerja preslikovalnega vodila, mikroprogramiranega procesorja (cikel 150 nanosekund) in vmesnika za dve vodili (intercluster bus). Grupe so povezane med seboj z vodili. Struktura sistema ima torej tri nivoje: grupe, računalniške module in posamezne elemente (procesor, pomnilnik, V/I naprava). V skladu s tem imamo tri vrste referenc (doseganje pomnilnika): lokalne, v okviru grupe in izven lastne grupe. Lokalne reference se izvajajo na klasični način - vzpostavi se fizična povezava in nato sledi prenos podatka (circuit switching). Pri vseh ostalih vrstah referenc pa se sistem poslužuje drugega mehanizma (packet switching). Vodila niso dodeljena za ves čas trajanja reference, temveč samo za čas, ki je potreben, da se prenese paket od enega vozlišča (Kmap, Slocal) do drugega. Paket vsebuje naslov in/all podatek. V vsakem trenutku je angažiran torej le del poti med procesorjem in naslovljenim pomnilnikom. Tak način omogoča boljšo izkoriščenost vodil (lokalna referenca traja 3,5 mikrosekund, znotraj grupe 9,5 mikrosekund in med grupama 26 mikrosekund). Vsi procesorji si delijo skupni virtualni pomnilnik velikosti 2^{28} zlogov. Preslikovalni mehanizmi, ki podpirajo koncept virtualnega pomnilnika so koncentrirani v "Slocal" in "Kmap" modulih. Pomnilnik je razdeljen na segmente z največ 4Kzlogi. V "Slocal" modulu je relokacijska tabela, preko katere se preslikujejo vse lokalne reference. Če naslov ni lokalen, se posreduje "Kmap" modulu, ki kontrolira preslikovalno vodilo. Procesor, ki je generiral naslov, se odklopi od internega vodila. Če je referenca znotraj grupe, generira "Kmap" fizični naslov in ga pošlje ustreznemu "Slocal" modulu. V primeru referenc izven lastne grupe se vključijo še vodila, ki povezujejo "Kmap" module.

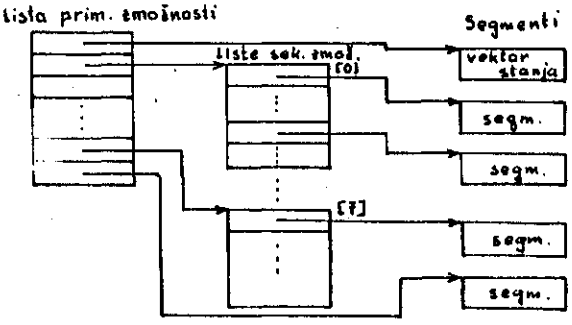
Koncept virtualnega pomnilnika

Vsak procesor lahko naslavlja največ 64Kzlogov (16bitni naslov). Ta prostor je razdeljen na 16 strani po 4Kzlogov. Vsaka stran pomeni okno v sistemski virtualni pomnilnik (2^{28} zlogov) (slika 5.1.2). Najvišja stran procesorjevega naslovnega prostora je rezervirana. Vsebuje med drugim 15 registrov (okenski registri), ki definirajo povezavo med stranmi procesorja in segmenti virtualnega pomnilnika. Povezava je indirektna preko "zmožnosti" (capability).



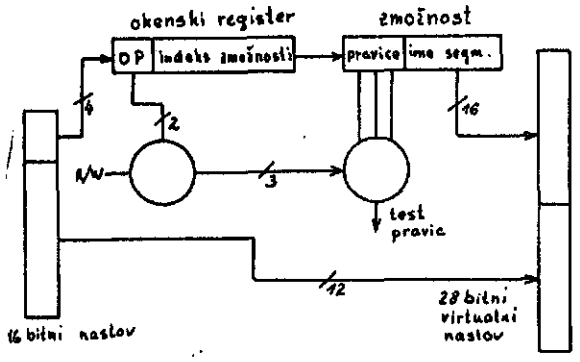
Slika 5.1.2.Okna iz procesorjevega naslovnega prostora v sistemski pomnilnik.

Vsak od teh 15 registrov vsebuje indeks, ki se nanaša na strukturo zmožnosti procesa, ki se trenutno odvija na procesorju. Zmožnost je predstavljena z dvema besedama, ki vsebujeta ime segmenta in pravice t.j. operacije, ki so dovoljene nad tem segmentom. Vsak proces je predstavljen z okoljem (environment) (slika 5.1.3). To je trinivojska struktura. Na prvem nivoju je lista primarnih zmožnosti. Prvi element te liste je kazalec na vektor stanja procesa kadar se ne izvaja. Ostali elementi pa so kazalci na liste sekundarnih zmožnosti. Teh je lahko osem. Vsi ostali elementi primarne liste in elementi sekundarnih list so zmožnosti za segmente, ki jih lahko proces direktno naslavlja. Tretji nivo pa sestavljajo segmenti. Vsak segment je definiran z deskriptorjem segmenta. Proces lahko dosega samo segmente za katere ima zmožnosti. Nad temi segmenti lahko izvaja samo operacije dovoljene z zmožnostmi.



Slika 5.1.3.Struktura okolja procesa

Virtualni naslov se torej generira na naslednji način (slika 5.1.4). Štirje najvišji biti 16bitnega naslova izberejo enega od 15 okenskih registrov. Ta vsebuje indeks za zmožnost v strukturi procesovega okolja. Ta zmožnost vsebuje ime segmenta (16 bitov), ki z 12 biti originalnega fizičnega naslova formira 28bitni virtualni naslov. Pri tvorbi virtualnega naslova se opravi tudi test med želeno operacijo nad segmentom in pravico definirano z zmožnostjo. Če je test negativen, se sproži past, ki servisira napake. Če hočemo spremeniti preslikavo strani v segment virtualnega pomnilnika, moramo zapisati v ustrezni okenski register indeks, ki kaže na novo zmožnost.



Slika 5.1.4. Generiranje virtualnega naslova.

Segment je definiran z deskriptorjem. Ta določa začetni naslov in dolžino. Segment lahko ima lastnost sklada, vrste ali drugih podatkovnih struktur. Za vsak segment je mogoče definirati do osem dovoljenih operacij. Pri običajnih segmentih sta to operaciji čitanja in zapisovanja. Segment je lahko tudi prazen v pomnilniškem smislu. V tem primeru sproži referenca kakšno kontrolno operacijo.

Vzajemno izključevanje procesov

V Cm* so implementirane vse kontrolne operacije na osnovi pomnilniških referenc. Tako npr. sproži procesor prekinitev na drugem procesorju z navedbo (referenco) določenega naslova. To nam omogoča, da se tudi pri kontrolnih operacijah poslužujemo zaščitnih in preslikovalnih mehanizmov. V Cm* ni implementirana nedeljiva "test and set" operacija nad nelokalnim pomnilnikom. Zato pa lahko "Kmap" blokira določen segment za določeno serijo referenc. S tem je zagotovljena nedeljivost kompleksnejših operacij. Dve taki operaciji, ki omogočata vzajemno izključevanje sta:

- prečitaj vsebino naslovljene besede, če je večja od nič, zmanjšaj njeno vrednost za ena. Vrni originalno vrednost.
- povečaj vrednost vsebine naslovljene besede za ena. Vrni originalno vrednost.

Komunikacija

Pri komunikaciji uporabljamo posebne segmente imenovane poštni nabiralniki (mail box). Procesi oddajajo sporočila v nabiralnike iz katerih jih naslovniki pobirajo. Sporočila so lahko cell segmenti ali posamezne besede. V prvem primeru se prenese preko nabiralnika samo zmožnost za dostop do segmenta, ki predstavlja sporočilo. Nad nabiralniki sta torej definirani sledeči operaciji:

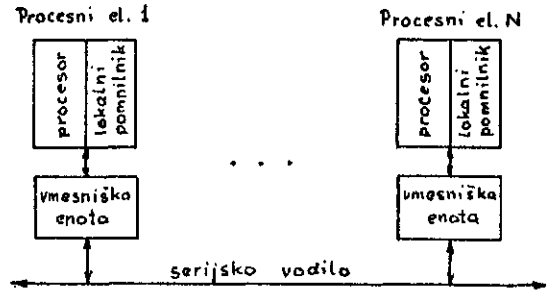
- Send(Message, ReplyMBox, Mailbox). Ta operacija prenese zmožnosti za sporočilo in za nabiralnik za odgovor iz liste zmožnosti klicujočega procesa v nabiralnik (Mailbox). Če je nabiralnik poln se proces uvrsti v čakalno vrsto.
- Receive(Mailbox). Če je v nabiralniku sporočilo oz. zmožnost za doseganje segmenta, ki predstavlja sporočilo, potem se preneseta zmožnosti za sporočilo in nabiralnik za odgovor v listo zmožnosti procesa. Sicer pa se proces postavi v čakalno vrsto.

Cm* spada po naši klasifikaciji med tesno povezane sisteme. Po zgradbi bi ga težje uvrstili v katero od naštetih vrst. Gre za sistem z več vodili, ki jih je mogoče dokaj svobodno povezovati. Osnovni elementi - grupe pa predstavljajo poznano strukturo s skupnim vodilom in centralnim koordinatorjem. Sistem je torej dokaj fleksibilen. Zanimiv koncept doseganja pomnilnika, ki vsebuje možnosti preslikavanja in zaščite omogoča uporabo obeh vrst kontrol - centralizirane in distribuirane. Z ustreznim razporejanjem zmožnosti je mogoče doseči, da je omogočeno doseganje skupnega pomnilnika ali pa ne. Ta izredno dobra fleksibilnost je bila poglavitni cilj pri načrtovanju sistema, saj smo že omenili, da je bil Cm* sistem zgrajen za raziskovalne namene.

5.2. HXDP - Honeywell Experimental Distributed Processor

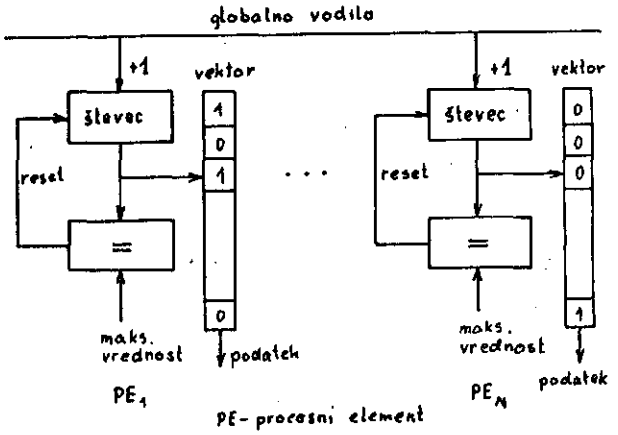
Podobno kot Cm* je bil HXDP zgrajen za raziskave na področju distribuirane kontrole, procesorskih povezav, itd. Namenjen je predvsem za sprotno vodenje (real time). Posebnost sistema je v nekonvencionalni delitvi kontrolnih funkcij med programsko opremo in hardwarom. Glavni cilj pa je popolna distribuirana kontrola. Ta bazira na medprocesni in s tem na medprocesorski komunikaciji.

HXDP sestavlja 64 šibko povezanih procesnih elementov. Povezani so s serijskim vodilom (hitrost 1,25 Mbit/sek). Taka struktura omogoča, da vsi procesni elementi opazujejo dogajanja v sistemu. Procesni elementi so priključeni na vodilo preko vmesniških enot (slika 5.2.1).



Slika 5.2.1. Zgradba HXDP sistema.

Pri medprocesni komunikaciji gre za dejansko pošiljanje sporočil. Vsi procesi so med seboj enako oddaljeni. To pomeni, da gredo tudi sporočila med dvema procesoma na istem procesnem elementu preko globalnega vodila. To sicer pomeni dodatno obremenjevanje vodila, po drugi strani pa olajšuje distribuirano kontrolo. Sporočila so naslovljena na imena in ne na fizične naslove hardwarskih procesnih elementov. Vsaka vmesniška enota razpoznava do osem različnih imen. Vsak procesni element ima osem vhodnih in eno izhodno vrsto za sporočila. Vse so tipa FIFO (first in first out). Realizirane so v hardwaru. Program samo vloži sporočilo v izhodno vrsto, če je v njej še prostor. Podobno prestreza vmesniška enota sporočila in jih spravlja v ustrezne vhodne vrste. Zastavice in kazalci omogočajo programsko sprejemanje teh sporočil. Za dodeljevanje vodila uporablja sistem posebno tehniko imenovano VDPA (Vector Driven Proportional Access). Na sliki 5.2.2 je ta tehnika prikazana.



Slika 5.2.2.VDPA tehtka

V vsaki vmesniški enoti je 256bitni vektor in indeks, ki kaže na en element vektorja. Ob konfiguraciji sistema se v vektor zapiše vzorec enic, ki določa kako pogosto bo lahko procesni element uporabljal vodilo. Ob inicializaciji se vsi indeksi postavijo na začetek vektorja in se sinhrono pomikajo vzdolž vektorja. Vrednost vseh indeksov poveča za ena poseben signal na koncu vsakega sporočila. Pri vsaki poziciji indeksa lahko ima samo en vektor vrednost 1. Vrednost 1 daje pravico na uporabo vodila. Odda se eno sporočilo iz izhodne vrste.

Opisani sistem je manj fleksibilen kot Cm*. Fleksibilnost se manifestira samo v obliki razširljivosti (dodajanje procesnih elementov je enostavno). Rekonfiguracija osnovne topološke strukture pa je nemogoča. S stališča integritete je sistem dokaj dober. To nam je dala slutiti že tendenca po distribuirani kontroli. S tem, da je komunikacija popolnoma javna, je relativno lahko odkriti napako v sistemu. Kritični del sistema so vmesniški moduli. Pravilno dodeljevanje vodila zavisi od normalnega delovanje VDPA algoritma. Nobeba enota ne sme zgrešiti signala, ki poveča vrednost indeksa. Zato je potrebno vpeljati zaščitne mehanizme proti izgubi tega signala.

6. SKLEP

V tem članku smo podali samo pregled problematike vezane na sisteme z več procesorji. Pri tem smo poskusili zajeti vso širino, ki je v literaturi običajno premalo poudarjena. Nekaj problematike najdemo opisane pod naslovi vezanimi na operacijske sisteme, nekaj na področju arhitektur sistemov, nekaj pod zanesljivostjo oz. integriteto sistemov itd. Naš cilj je bil torej prej pogledati kako obširen je gozd v katerega se podajamo, kot pa postajanje pri vsakem drevesu. Seznam uporabljene literature bo bralcu v pomoč, da bo lahko sam dovolj podrobno proučil problematiko, ki ga zanima.

Na začetku članka smo videli kateri cilji nas pripeljejo do sistemov z več procesorji. Potem smo sisteme klasificirali po različnih kriterijih. Nekatere vrste tesno povezanih sistemov smo analizirali bolj podrobno. V naslednjem delu smo spregovorili o kontroli takih sistemov. Najdalj smo se pomudili pri sinhronizacijskih mehanizmih, ki so osnova za vzajemno izključevanje in komunikacijo. Poudarek je bil na distribuiranih mehanizmih. Za konec pa smo si podrobneje ogledali dva multiprocesorska sistema, ki služita za raziskave na tem še premalo raziskanem področju.

7. LITERATURA

1.(FULLER78) S.H.Fuller,et al:"Multi-Microprocessors: An Overview and Working Example" Proc.of the IEEE,Vol.66,No.2,pp.216-228.

2.(MAHR78) W.Mahr,R.Patzelt:"Improvments of the Multiprocessing Capabilities of Microprocessor Busses", Euromicro Journal,Vol.4, pp.2o7-219.

3.(ENSLOW77) P.H.Enslow Jr:"Multiprocessor Organization - A Survey", Computing Surveys,Vol.9,No.1,March 1977,pp.1o3-129.

4.(SEARLE75) B.C.Searle,D.E.Freberg:"Tutorial: Microprocessor Applications in Multiple Processor Systems",Computer,Oct.1975, pp.22-3o.

5.(PATEL79) J.H.Patel:"Processor - Memory Interconnections for Multiprocessors", 6th Ann.Symp.on Computer Architecture, April 1979,pp.168-177.

6.(NOVAK8o) D.Novak idr:"Jedro za podporo implementacije sprotnih sistemov", časopis Informatica,letnik 4,št.2,str.3-7.

7.(BUHR78) R.J.A.Buhr:"Toward Transparent Coordination Mechanisms for Multiple Microprocessor Systems",Collected Technical Papers of the Microprocessor System Dev. Lab.,C,June 1977-May 1978, Dept.of Systems Engineering and Computing Science, Carleton University, Ottawa.

8.(LELANN77) G.Le Lann:"Distributed Systems - Toward a Formal Approach", Proceedings of IFIP Congress 77, Toronto,Aug.1977, North Holland,pp.155-16o.

9.(LAMPORT78) L.Lamport:"Time,Clocks and the Ordering of Events in a Distributed System",CACM,Vol.21,No.7,pp.558-565.

1o.(LELANN8o) G.Le Lann:"Synchronization in Distributed Systems",Materials from NATO ASI Multiple Processor Computers,Maratea, Italy,15-28 June 198o.

11.(DIJKSTRA74) E.W.Dijkstra:"Self-stabilizing Systems in Spite of Distributed Control", Communications of the ACM,Vol.17,No.11, November 1974,pp.643-644.

12.(WIRTH77) N.Wirth:"Toward a Discipline of Real-Time Programming",CACM,Vol.2o,No.8, August 1977,pp.577-583.

13.(NWIRTH77) N.Wirth:"Modula"(trije članki), Software-Practice and Experience,Vol.7, pp.3-84.

14.(NOVAK79) D.Novak idr:"Modula - programski jezik prihodnosti za mikroračunalniške aplikacije?",Informatica,let.3,št.2,18-24.

15.(DIJKSTRA68) E.W.Dijkstra:"Co-operating Seqvential Processes",in Genuys (ed.):"Programming Languages",Academic Press 1968, New York,pp.43-112.

16.(HOARE74) C.A.R.Hoare:"Monitors: An Operating System Structuring Concept",CACM,Vol.17, No.1o,pp.549-557.

17.(EXEL8o) M.Exel:"Monitorji",Delovno poročilo IJS,198o,DP2o87.

18.(SWAN77) Swan,R.J.et al:"Cm*- A modular,multi microprocessor",AFIPS Conf.Proc.,Vol.46, 1977 National Computer Conference.

19.(JENSEN78) E.D.Jensen:"The Honeywell Experimental Distributed Processor - An Overview",Computer,1978,No.1,pp.28-38.

2o.(RBUHR78) R.J.A.Buhr:"Concurrent High Level Language Models Applied to Multimicroprocessor System Development, glej 7.

ARHITEKTURA RAČUNARA ZA OBRADU DVODIMENZIONALNIH SLIKA

SLOBODAN RIBARIĆ

TEHNIČKA VOJNA AKADEMIJA KoV JNA, ZAGREB

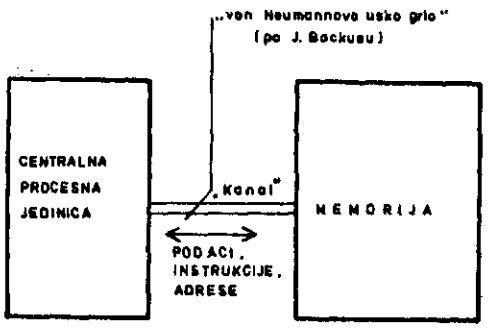
UDK: 681.3.06

U radu je prikazan razred računarskih sistema i procesora koji se po arhitekturi razlikuju od konvencionalnog von Neumannovog sekvencijalnog računara.
Opisani su sistemi za paralelnu obradu dvodimenzionalnih slika (SPAC, SOLOMON, ILLIAC III, Glucksmanov parapropagacijski procesor, UCPR 1, CLIP, STARAN, PMM, AP-120B) i dane su ocjene njihovih performansi. Razvoj LSI tehnologije utječe na arhitekturu računara za obradu slika (MPP procesor, procesor na bazi dvodimenzionalnog polja mikroprocesora) i omogućio je primjenu procesora na bazi 4-bitnih ili 8-bitnih odrezaka, te primjenu procesora i računara dinamičke arhitekture.

COMPUTER ARCHITECTURE FOR TWO-DIMENSIONAL PICTURE PROCESSING: Computer systems and processors that differs in their architecture from von Neumann conventional sequential computer are presented. Systems for two-dimensional picture parallel processing are discussed (SPAC, SOLOMON, ILLIAC III, Glucksman's parapropagation processor, UCPR 1, CLIP, STARAN, PMM, AP-120 B) and their performances evaluation is given. LSI technology development impacts on computer architecture for picture processing (MPP processor, processor based on two-dimensional field of microprocessors) and pave the way for application of processors based on 4- or 8-bit slices, and usage of processors and computers with dinamical architecture.

1. UVOD

Većina postupaka primjenjenih u digitalnoj obradi i raspoznavanju slika je sekvencijalnog tipa, iako je stvarni proces u svojoj osnovi paralelan. Razlozi serijske prirode tih postupaka leže u činjenici da je naše kvantitativno opažanje i opisivanje svijeta (koji je sam inherentno paralelan) već preko 300 godina podvrgnuto utjecaju sekvencijalne matematike, preko 50 godina je pod teretom sekvencijalnih algoritama i preko 20 godina pod utjecajem sekvencijalnog programiranja u FORTRAN-u [1], [2]. Drugi razlog, koji je implicitno sadržan u prvom, je u tome da je većina računarskih sistema, koji se primijenjuju u postupku obrade i raspoznavanja slika, sekvencijalnog tipa. Neumannov model sekvencijalnog računara [3] sa četiri funkcionalne komponente, sa koncepcijom pohranjivanja instrukcija i podataka u istoj memoriji, te sa koncepcijom sekvencijalnog izvođenja instrukcija, suvereno vlada već u tri generacije računara. Usprkos tome, model doživljava i kritike: J.Backus* ironično naziva vezu memorije i centralne procesne jedinice " " u tom modelu "von Neumannovim uskim grlom" (sl. 1).



Sl. 1. Pojednostavljeni funkcionalni model von Neumannovog računara

Zadatak je programa da mijenja sadržaj memorije, a to se u von Neumannovom modelu izvodi stalnim slanjem riječ po riječ naprijed-natrag preko "kanala" (sl. 1) koji povezuje memoriju i centralnu procesnu jedinicu. Veliki dio tog "prometa" riječi u tom "kanalu" ("von Neumannovom uskom grlu") nisu korisni podaci već instrukcije i adrese. J.Backus tvrdi da sigurno mora postojati "... manje primitivan način ..." stvaranja željenih promjena u memoriji od ovog slanja ogromnog broja riječi naprijed-natrag kroz "kanal" . Ne samo to, koncept von Neumanna, J.Backus naziva i "intelektualnim uskim grlom", jer upućuje na način razmišljanja koji isključuje paralelnost [4].

* tvorac Fortrana

Razvojem tehnologije visokog (LSI) i vrlo visokog stupnja integracije (VLSI) postale su paralelne elektroničke i računarske strukture dostupne i ekonomski opravdane. Takva pojava omogućila je rad na računarskim strukturama čije su koncepcije bile u prošlosti potisnute upravo zbog tehnoloških ograničenja. Kako iskoristiti te nove mogućnosti na području obrade slika? Dvojako:

a) traženjem i iskorištavanjem paralelizma u postojećim algoritmima i postupcima obrade slika, te njihovim prilagođavanjem paralelnim računarskim strukturama [5], [6],

b) vraćanjem na same probleme, odnosno prirodne fenomene i uvođenjem novih matematičkih i programskih modela u postupku obrade i raspoznavanja slika [7], [8], [9].

U ovom članku dat je pregled arhitekture paralelnih računarskih struktura koje se primjenjuju u digitalnoj obradi slika i njihov utjecaj na brzinu izvođenja obrade.

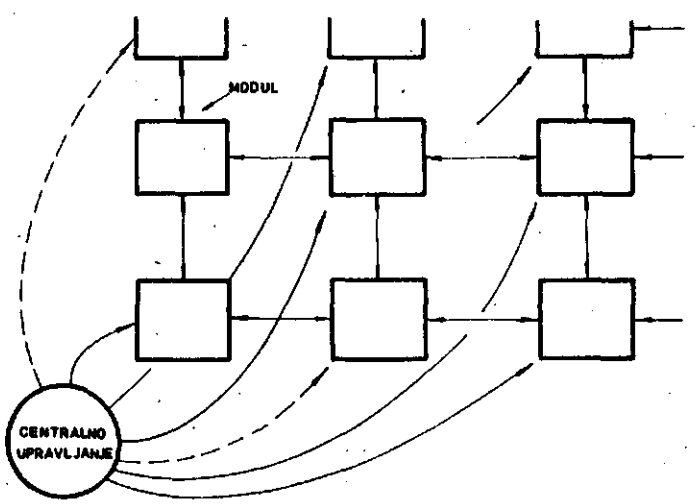
2. PARALELNE RAČUNARSKE STRUKTURE U OBRADI SLIKA

U drugoj polovini ovog desetljeća (70-80) iskristalizirala su se dva smjera razvoja sistema za obradu i raspoznavanje slika [10]:

a) razvoj velikih sistema za teorijska istraživanja, sistema za razvoj novih aplikacija, te za egzaktnija rješenja znanih problema (primjerice, sistem sa CDC 6600 računarom, asocijativni procesor STARAN, ili više računarski sistem na bazi PDP 11/40 povezan sa ARPA mrežom [11]),

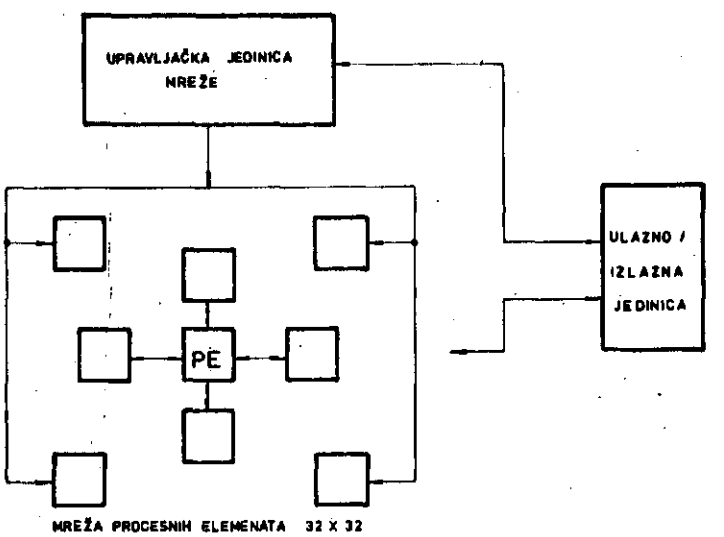
b) razvoj manjih sistema za rješavanje skupa specifičnih problema obrade i raspoznavanja slika (primjerice, sistemi opisani u [1] i [7]).

Paralelizam u postupku obrade i raspoznavanja slika, te razvoj paralelnih algoritama [12] - [15] za obradu slika snažno utječu na arhitekturu oba tipa sistema.
S.H. Unger [16], [17] je još 1958.godine predložio model računara za obradu dvodimenzionalnih slika imajući u vidu neprilagođenost arhitekture konvencionalnih računara.



Sl. 2. Model računara S.H. Ungera (Spatical computer)

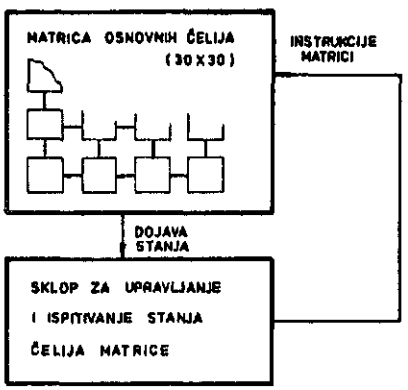
Slika 2. prikazuje predloženu organizaciju računara SPAC (spatial computer). Model računara se sastoji iz polja logičkih modula. Modul je, u stvari, jednobitni "računar" sa jednim jednobitnim akumulatorom, memorijom sa izravnim pristupom koja se sastoji od šest jednobitnih riječi, i sa sklopovima za osnovne logičke operacije, te operacije pohranjivanja i čitanja iz memorije. Svaki modul saobraća sa četiri susjeda, a instrukcije prima od sklopa centralnog upravljanja. Sklop centralnog upravljanja izdaje instrukcije cijelom polju modula i nema mogućnost adresiranja pojedinog modula. Svi moduli izvršavaju istu instrukciju. Kasnije će takva arhitektura računara (M.J.Flynn 1972. [18]) biti klasificirana kao SIMD (Single Instruction stream Multiple Data Stream) - jednostruki instrukcijski tok višestruki tok podataka. S. Unger je predvidio i paralelni unos slike preko ulaznog uređaja koji čine fotodiode pridružene svakom modulu u polju. Zbog tehnoloških ograničenja predloženi model računara SPAC nije realiziran, već je polje dimenzija 36 x 36 modula simulirano na računaru IBM 704. Simulirani su programi za izlučivanje šuma (gla đenje slike), detekciju rubova, stanjivanje, te raspoznavanje slova [17].



Sl. 3. Organizacija SOLOMON računara

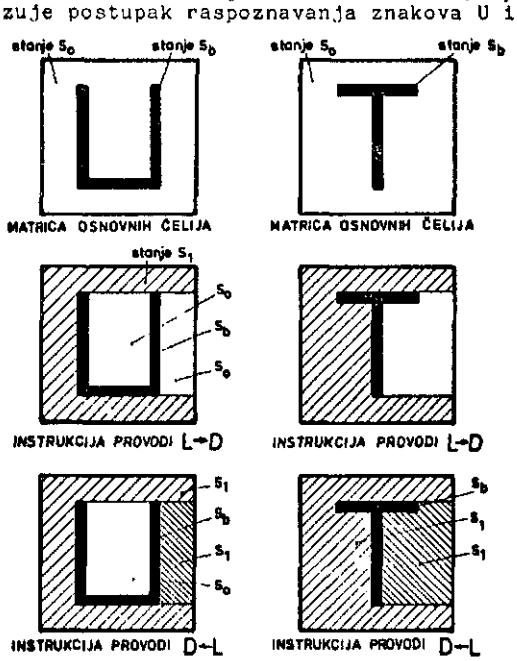
Slika 3. prikazuje računar SOLOMON [20] u kojem je mreža procesnih elemenata (PE) organizirana poput one kod SPAC računara. Svaki PE je povezan sa svoja četiri susjeda. U SOLOMON-u, također, cijela mreža simultano izvršava instrukciju. Upravljanje, međutim, omogućava da neki od PE iz mreže ignoriraju izvođenje dane instrukcije.
Osnovni sistem sastoji se od mreže procesnih elemenata (32 x 32), upravljačke jedinice mreže, te ulazno/izlazne jedinice.
Godine 1963. razvijen je sistem ILLIAC III (Digital Computer Laboratory University of Illinois) [19] za obradu dvodimenzionalnih slika. Sistem se sastoji iz pet funkcionalnih jedinica: skanera velike brzine, PAU (Pattern Articulation Unit) jedinice za lokalno pretprocesiranje slika, TU (Taxicrinic Unit) jedinice za raspoznavanje, aritmetičke jedinice za izvođenje matematičke analize i statističke obrade, te izlaznih jedinica (štampača i prikazne jedinice). Jezgro sistema je PAU jedinica koja se sastoji od 1024 (32x32) identičnih međusobno povezanih procesnih modula. Oni izvode Boolove funkcije, operacije sa vrijednostima praga, stanjivanje, gla đenje, detekciju linijskih elemenata i druge postupke pretprocesiranja slika. ILLIAC III se koristi u obradi i analizi fotografija tragova čestica.

H.A. Glucksman (1965.godine) predložio je paralelni sistem [21] koji se sastoji iz matrice od 30x30 osnovnih ćelija i sklopa za ispitivanje stanja matrice (sl.4).



Sl. 4. Blok-shema Glucksmanovog paralelnog procesora

Osnovna ćelija se sastoji iz dva bistabila i desetak I i ILI vrata. Zbog jednostavnosti i pravilnosti strukture sistem je pogodan za izvedbu u integriranoj tehnici. Sistem primjenjuje princip sprečavanja provođenja (parapropagacija [21]), i jednostavnim paralelnim operacijama promjene stanja ćelija vrši klasifikaciju tipiziranih slova i brojeva. Svaka ćelija matrice može biti u jednom od tri stanja: S_0, S_1 i S_b. Sve ćelije su u početnom trenutku u stanju S_0. Projicirano slovo ili znak na matricu postavlja ćelije pod konturom u stanje S_b. Svaka ćelija ima svojstvo provođenja u četiri smjera, odnosno promjene stanja S_0 u S_1, ali sve dok ne naiđe na barijeru - ćeliju sa stanjem S_b. Tada nastupa prekid provođenja (izmjene stanja), odnosno parapropagacija. Kombinacijom provođenja u različitim smjerovima i detekcijom konačnog stanja matrice (postojanje dijela matrice sa stanjem S_0) moguće je izvršiti klasifikaciju znakova. Slika 5. prikazuje postupak raspoznavanja znakova U i T.

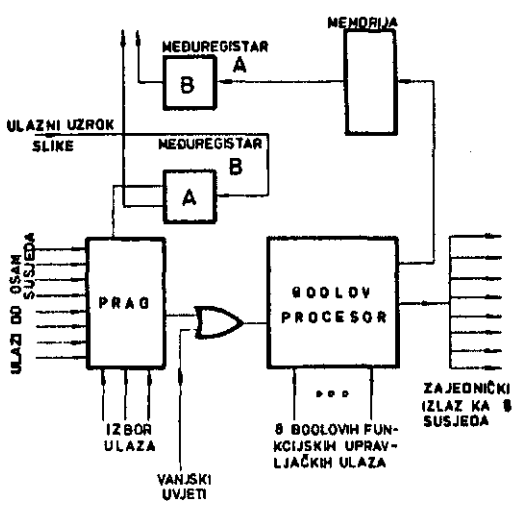


Sl. 5. Postupak raspoznavanja znakova U i T

Ako nakon sekvencije paralelnog provođenja L→D, D→L matrica sadrži ćelije sa stanjem S_0, nepoznati znak je U. Simulacija Glucksmanovog sistema na mikroračunaru (sa matricom 8x8 ćelija) zahtijevala je 1,5 k bajta programske memorije i dala je zadovoljavajuće rezultate raspoznavanja za tipizirane znakove (99% ispravno klasificiranih).
Godine 1967. M.J.B. Duff i B.M.Jones konstruirali su sistem UCPR 1 [1] za automatsku lokalizaciju slijedova na fotografijama tragova čestica. Prilikom konstrukcije sistema pridržavali su se tri osnovna principa [1] :

a) sistem mora imati paralelni unos i paralelnu obradu slika,
b) komponente sistema moraju biti takve da su pogodne za mikrominijaturizaciju,
c) primjeniti tehnike i metode, koliko je god to moguće, koje počivaju na iskustvima dobijenim analizom fizioloških sistema.

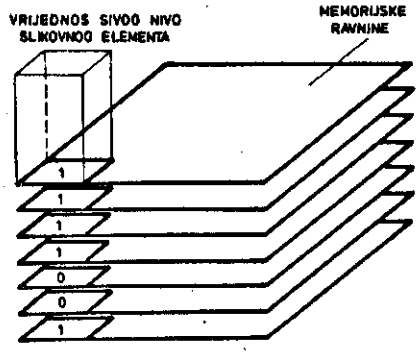
M.J.B.Duff i D.M.Watson su 1973.godine razvili prototip paralelnog procesora CLIP izvedenog u TTL tehnologiji. Godine 1978. su M.J.B. Duff, L.P. Cordella i S.Levialdi implementirali CLIP procesor u MOS tehnologiji. Procesor se sastoji od polja 96x96 procesnih elemenata [22]. CLIP procesor pripada SIMD arhitekturi.



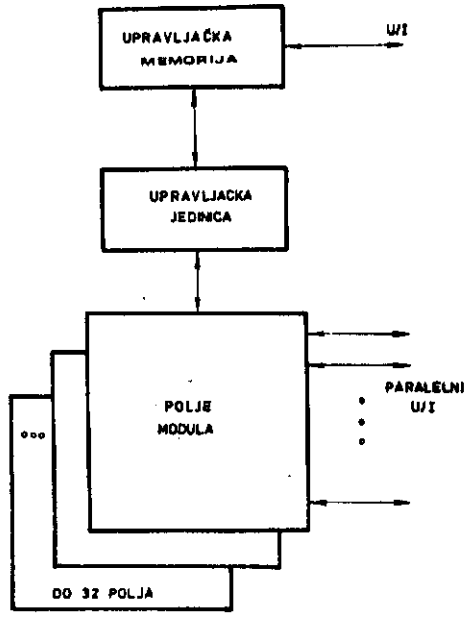
Sl. 6. Procesni element CLIP procesora

Procesni element (sl.6) sličan je procesnom modulu PAU procesnog polja sistema ILLIAC III. Slika se u CLIP procesoru pohranjuje u memorijske ravnine - svaki bit vrijednosti sivog nivoa točke na slici pohranjen je u memorijskoj ravnini tako da stupac tvori vrijednost sivog nivoa slike (sl.7). Binarna slika se pohranjuje u jednoj ravnini. Primjena procesnog polja i takve organizacije memorije omogućuje efikasno paralelno izvođenje postupaka pretprocesiranja slika (gla đenje, izlučivanje konture, sužavanje, modifikacija, histograma i sl.). Na primjer, koeficijent povećanja brzine S_p za operaciju gla đenja (u odnosu na sekvencijalni miniračunar HP 2116B) iznosi $\omega^2/6$, gdje je ω dimenzija slike.

Slika 8. prikazuje organizaciju paralelnog računara STARAN [23],[24]. Iako STARAN nije konstruiran sa namjenom da se koristi u obradi slika upravo njegova paralelna arhitektura omogućava efikasnu primjenu na tom području.



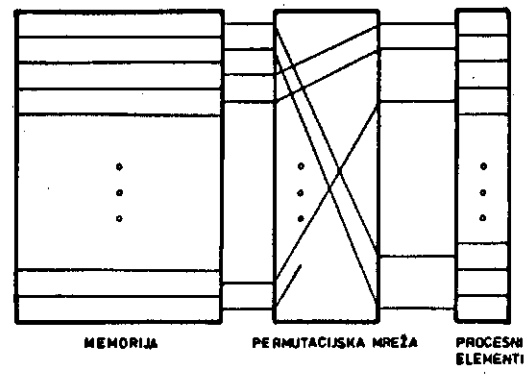
Sl. 7. Organizacija memorije CLIP procesora



Sl. 8. Organizacija paralelnog računara STARAN

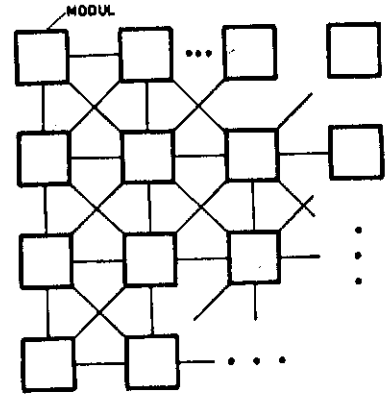
Osnovna arhitektura računara STARAN sastoji se od konvencionalne upravljačke memorije, upravljačke jedinice i više polja modula. Polje modula se sastoji od više-dimenzionalne pristupne* memorije, permutacijske mreže i 256 procesnih elemenata (sl.9). Kod arhitekture paralelnog računara STARAN izbjegnuto je "von Neumannovo usko grlo". Svakoj riječi u memoriji dodijeljena je veza (preko permutacijske mreže) sa jednim procesorom. Primjena STARANA u obradi i klasifikaciji slika dala je dobre rezultate. Na primjer, klasifikacija po metodi najveće vjerojatnosti izvodi se za faktor 4÷5 puta brže u odnosu na računar IBM 360/75, konvolucija za faktor 80 u odnosu na računar IBM 360/195, te filtriranje slike za faktor 150 u odnosu na računar HP 3000 [23].

* pod programskim upravljanjem riječi u memorijskom polju mogu biti razdijeljene na riječi proizvoljne dužine.



Sl. 9. Polje modula računara STARAN

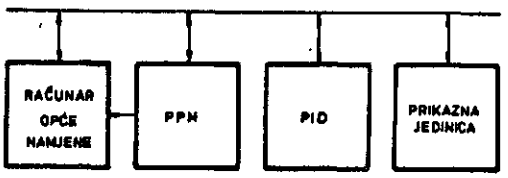
Odličan pregled arhitekture paralelnih i asocijativnih računara - dao je K.J.Thurber [24]. B.Kruse je opisao PPM (Parallel Picture Processing Machine) [25] koji je jezgro sistema za obradu slika.
PPM se sastoji od polja identičnih modula, svaki modul dodijeljen je za obradu jedne slikovne točke. Svaki modul u polju je povezan sa osam susjednih modula (sl.10). Modul se sasto-



Sl. 10. Polje modula PPM procesora

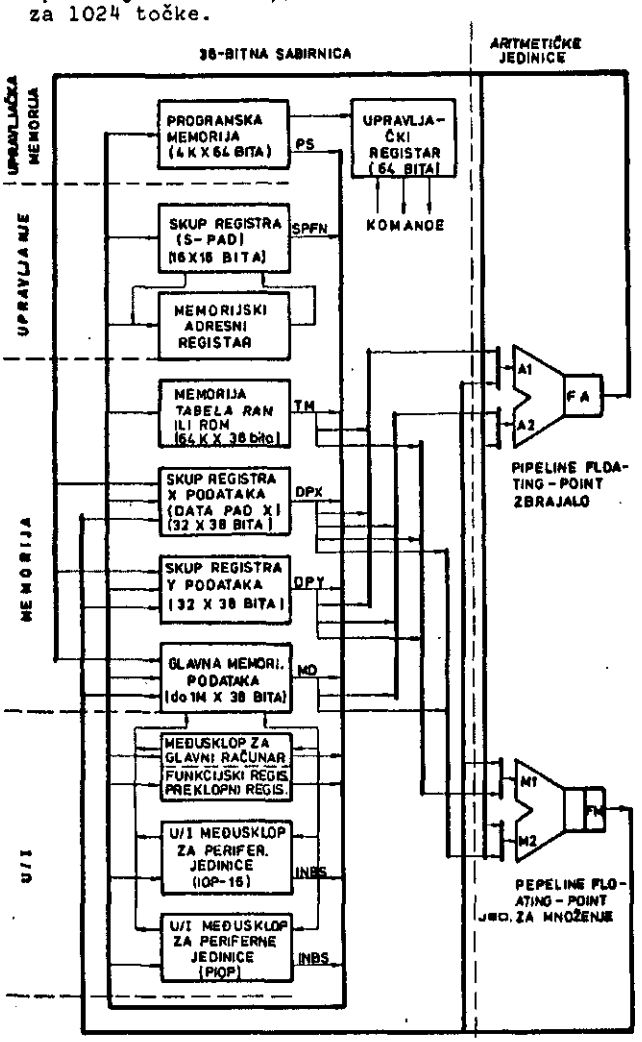
ji od kombinacionog sklopa, upravljačke jedinice i registra stanja. PPM procesor je namijenjen za pretprocesiranje slika, a svojom organizacijom prilagoden je za efikasno izvođenje lokalnih slikovnih operacija. Sistem za obradu slika čine PPM procesor, jedinica za unos slike PID, prikazna jedinica i računar opće namjene koji upravlja komponentama sistema (sl. 11).

U obradi slika često se primjenjuju sistemi koji predstavljaju spregu računara ili miniračunara i procesora specijalne namjene.



Sl. 11. Sistem za obradu slika sa PPM procesorom

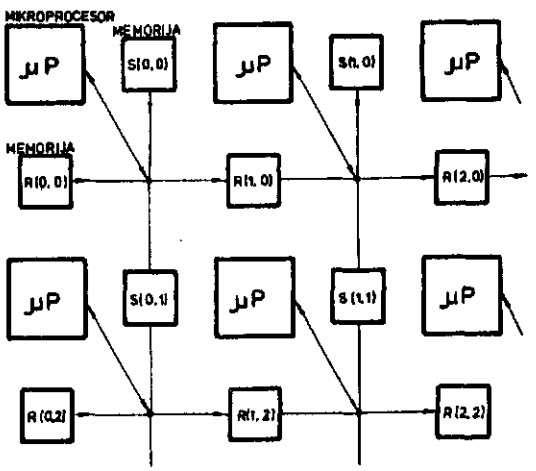
Slika 12. prikazuje arhitekturu aritmetičkog procesora AP-120B (Floating Point Systems) [26].
Višesabirnička arhitektura, interna memorija sa vremenom pristupa 167 ns, te pipeline tehnika primjenjena u 38-bitnim floating point aritmetičkim jedinicama dozvoljavaju veliku prilagodljivost u simultanom prijenosu operanada i rezultata, te veliku propustnost, tako da AP-120B izvršava 12 miliona floating point operacija ili za 2,7 milisekunde izvodi FFT za 1024 točke.



Sl. 12. Arhitektura procesora AP-120B

AP-120B djeluje u sprezi sa računarom opće namjene (proizvođač isporučuje opremu za proizvodnju AP-120B sa preko dvadeset tipova računara opće namjene [27]).
Proizvođač isporučuje i bogatu programsku biblioteku, te programsku podršku za razvoj programa (Assembler, Linker, Debugger, Simulator [28]).
AP-120B namijenjen je za obradu masivnih podataka, na primjer, za analizu i obradu podataka seizmičkih istraživanja, za analizu i obradu slika, govora i sl.
Primjer primjene takvog procesora u sprezi sa miniračunarom PDP 11/40 prikazan je u [29].

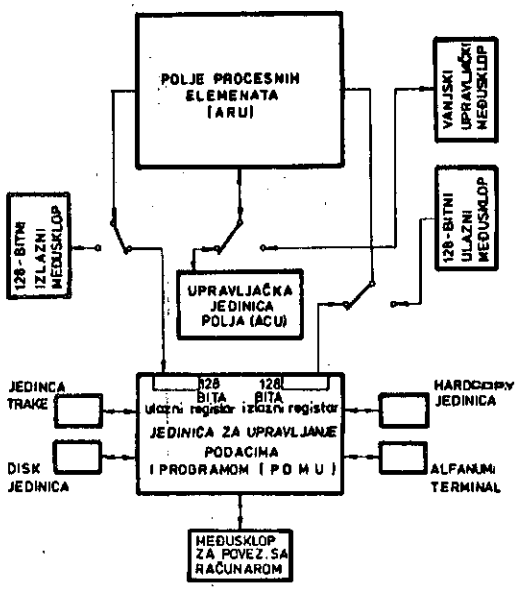
Razvoj LSI tehnologije i pojava visoko prilagodljivog, pouzdanog elementa niske cijene-mikroprocesora omogućila je njegovu primjenu u obradi slika.
Slika 13. prikazuje dvodimenzionalno polje mikroprocesora za obradu slika (R.P.Roesser [30]).



Sl. 13. Dvodimenzionalno polje mikroprocesora za obradu slika

U slučaju slika sa visokom rezolucijom (npr. 512x512 slikovnih točaka) nemoguće je svakoj točki slike dodijeliti procesor i memoriju. To se rješava tako da se ulazna slika razdijeli na sektore i svaki se sektor obraduje sukcesivno sa istim poljem mikroprocesora. Eksperimentalna obrada slike dimenzija 1024x1024 sa poljem mikroprocesora veličine 16x16 pokazala je da je broj potrebnih ciklusa za obradu približno jednak omjeru broja slikovnih točaka sa brojem mikroprocesora ($1024^2/16^2 = 4096$ ciklusa) [30].

Slika 14. prikazuje sistem MPP (Massively Parallel Processor) [31], razvijen je u NASA Goddard Space Flight Center. Sistem namijenjen za obradu slika i dvodimenzionalnih podataka. Moć takvog sistema najbolje ilustrira podatak da sistem izvodi 6G (giga) operacija zbrajanja i oduzimanja u sekundu ili 2G množenja u sekundi.
MPP (sl.14) se sastoji od polja procesnih elemenata ARU (Array unit), upravljačke jedinice polja ACU (Array control unit), te jedinice za upravljanje podacima i programom PDMU (Program & Data management unit). Upravljačka jedinica izdaje identične upravljačke signale i adrese svim procesnim elementima u polju. MPP ima međusklop za povezivanje sa računarom i dva međusklopa za izravni pristup podataka polju procesnih elemenata. Polje procesnih elemenata sastoji se iz 16384 procesna elementa (128x128).



Sl. 14. MPP sistem za obradu slika

Svaki procesni element je LSI čip koji ima aritmetičku i logičku jedinicu, jedinicu za usmjeravanje podataka (svaki procesni element saobraća sa svoja četiri susjeda), memorijsku jedinicu sa izravnim pristupom (RAM) kapaciteta 256 bita, te U/I jedinicu za saobraćanje sa susjedima. Komponente unutar procesnog elementa saobraćaju preko dvosmjerne sabirnice podataka. Upravljačka jedinica polja ACU izdaje instrukcije polju sa frekvencijom 10 MHz. ACU jedinica je mikroprogramirana i primjenjuje tehniku prekrivanja izvodenja operacija u cilju postizanja veće brzine djelovanja.

MPP se primjenjuje u obradi satelitskih snimaka.

Razvoj LSI tehnologije omogućio je primjenu bit-odrezaka (bit-slice) veličine 4 (M 10800 Motorola, SBP 0400 TI, 2900 AMD) ili 8 bita (F 100220 Fairchild) kao osnove za izgradnju mikroprogramibilnih procesora specijalne namjene (procesori za realizaciju algoritama za obradu slika, procesori za brzu Fourierovu transformaciju, korelaciju i sl.) [32].

Daljnji razvoj računarske tehnike i arhitekture omogućit će primjenu univerzalnih modula sa programibilnim funkcijama i pojavu dinamičke arhitekture računara [33], gdje se rekonfiguracija računarskog sistema odvija pod isključivo programskim upravljanjem (npr. 64-bitni računar rekonfigurira u jedan 32-bitni i dva 16-bitna [33]). Takva dinamička arhitektura će zasigurno naći primjenu i u obradi i raspoznavanju slika, jer se u različitim fazama postupka obrade i klasifikacije slika zahtijeva različita konstrukcija i struktura procesora [34].

3. ZAKLJUČAK

U radu je prikazan razred računarskih sistema i procesora koji se po arhitekturi razlikuju od konvencionalnog von Neumannovog sekvencijalnog računara.
Opisani su sistemi za obradu dvodimenzionalnih slika i dane su njihove performanse.

Očito je, da razvoj arhitekture računara utječe na efikasnost obrade slika, međutim, postoji i povratni utjecaj paralelizma, prisutnog u svim fazama obrade i raspoznavanja (pretprocesiranje, izlučivanje karakteristika, klasifikacija), na arhitekturu računarskog sistema. Taj povratni utjecaj biti će sve izrazitiji daljnjim razvojem LSI tehnologije i sve nižom cijenom procesnih elemenata. Jedino što nedostaje su analize načina utjecaja paralelizma na model procesora te kriteriji njegovog izbora.

4. LITERATURA

1 M.J.B.Duff, Parallel Computation in Pattern Recognition, Methodologies of Pattern Recognition, Ed. S. Watanabe, Academic Press, N.York 1969.

2 K.J.Thurber, P.C.Patton, The Future of Parallel Processing, IEEE Transactions on Computers, Vol.C-22, No.12., decembar 1973. str. 1140-1143.

3 A.W.Burks, H.H.Goldstine, J.Neumann, Preliminary Discussion of the Logical Design of an Electronic Computing Instrument, J.Neumann; Collected Works, Vol.5, ed.A.H. Taub, Pergamon Press, Ltd. Oxford, 1961., str. 34-79.

4 J.Backus, Can Programming be Liberated From Neumman Style and Its Algebra of Programs, Communications of ACM, Vol. 21.No.8, august 1978. str. 613-641.

5 D.J.Kuck: A Survey of Parallel Machine Organization and Programming, ACM Computing Surveys, Vol.9.,No.1 mart 1977., str. 29-59.

6 C.V.Ramamoorthy, M.J.Gonzalez, A Survey of Techniques for Recognizing Parallel Processable Streams in Computer Programs, Fall Joint Computer Conf., 1969, str.1-15.

7 S.Levialdi, Paralllel Pattern Processing, IEEE Transactions on Systems, Man, Cybern., Vol. SMC-1, jul 1971, str. 292-296.

8 J.E.Hawkins: Network Properties for Pattern Recognition, Methodologies of Pattern Recognition, ed. S. Watanabe Academic Press 1969, str. 275-286.

9 S.Ribarić, Model pretprocesiranja i paralelizam na nivou slikovnih operacija, Informatica, Bled 1979. (3204).

10 E.C.Lyons, Digital Image Processing: An Overview, Computer, Vol.10., No.8., august 1977., st.12-14.

11 H.C.Andrews, An Educational Digital Image Processing Facility, ibid.str. 48-53.

12 R.Stefanelli, A.Rosenfeld, Some Parallel Thinning Algorithms for Digital Pictures, Journal of ACM, Vol. 18. No.2. april 1971.,str. 255-264.

13 M.C.Pease, An Adaptation of the Fast Fourier Transform for Parallel Processing, Journal of ACM, Vol.15. No.2, april 1968.str. 252-264.

14 M.J.B.Duff, Parallel Processing Techniques, Pattern Recognition, Ideas in Practice, Ed. B.G. Batchelor, Plenum Press, New York 1978.

15 S.K.Chang, On the Parallel Computation of Local Operations, Proc. 3rd ACM Symp. Theory of Computing, 1971., maj, str. 101-115.

16 S.H.Unger, A Computer Oriented Towards Spatial Problems, Proc. IRE, Vol. 46., oktobar 1958., str. 1744--1950.

17 S.H.Unger, Pattern Detection and Recognition, Proc. IRE, Vol.47, 1959. str. 1737-1752.

18 M.J.Flynn, Some Computer Organizations and Their Effectivenes, IEEE Transactionson Computeres, Vol. C-21., No.9., septembar 1972., str. 948-960.

19 B.H.McCormick, The Illinois Pattern Recognition Computer-ILLIAC III, IEEE Transactions on Electronic Computers, decembar 1963, str. 791-809.

20 J.Gregory, R.Mc Reynolds, The SOLOMON Computer, IEEE Transactions on Electronic Computers, december 1963, str. 774-780.

21 H.A.Glucksman, A Parapropagation Pattern Classifier, IEEE Transactions on Electronic Computers, jun 1965, str. 434-443.

22 L.P.Cordella, M.J.B.Duff, S.Levialdi, An Analysis of Computational Cost in Image Processing: A Case Study, IEEE Transactions on Computers, Vol. C-27, No.10, oktobar 1978, str. 904-910.

23 D.Rohrbacher, J.L.Potter, Image Processing with Staran Parallel Computer, Computer, Vol.10, No.8. august 1977., str. 54-59.

24 K.J.Thurber, Associative and Parallel Processors ACM Computing Surveys, Vol.7., No.4. decembar 1975., str. 215-255.

25 B.Kruse, A Parallel Picture Processing Machine, IEEE Transactions on Computers, Vol. C-22., No.12, decembar 1973, str.1075-1087.

26 Floating Point Systems, Inc.The Age of Array Processing is Here, FPS Inc 1978.

27 R.Bodmer, Floating Point Systems, S.A. LTD, (privatna komunikacija) 1978.

28 AP-120B, Array Transform Processor, Software Develepment Package Manuals, Floating Point Systems, Inc. 1976.

29 M.Čarapić, M.Kunc, S.Stanković, Computer System for Imate Data Processing, Informatica 79, Bled 1979. 6113.

30 R.P.Roesser, Two-Dimensional Microprocessor Pipelines for Image Processing, IEEE Transactions on Computers, Vol. C-27.No.2, februar 1978, str. 144-156.

31 Digital Technology Review, Parallel Processor Will Be Capable of Performing 6G Additions/s, Computer Design, mart 1979, str. 55-56.

32 N.A.Alexandridis, Bit-Sliced Microprocessor Architecture, Computer, jun 1978., str. 56-80.

33 S.I.Kartashev, S.P.Kartashev, L SI Modular Computers, Systems and Networks, Computer, Vol.11. No.7. juli 1978. str. 7-15.

34 L.Uhr, R.Douglass, A.Parallel - Serial Recognition Cone System for Perception: Some Test Results, Pattern Recognition, 1979, Vol. 11. str. 29-39.

ON A SUBROUTINE FOR ROOT FINDING

D. B. POPOVSKI

DEPARTMENT OF ENGINEERING UNIVERSITY OF BITOLA, BITOLA

UDK: 681.3.51

Some scientific subroutine packages contain a root finding subroutine, call it RTMI, *in which the method of equidistant inverse parabolic interpolation is used. It is mentioned that this method is a particular case of the method of inverse parabolic approximation. Subroutine* P *is presented in which the hybrid inverse parabolic approximation – – bisection method is used. It is shown that subroutine* P *is more effective than subroutine* RTMI.

O JEDNOM POTPROGRAMU ZA NALAŽENJE KORENA. *Neki paketi naučnih potprograma sadrže jednog potprograma za nalaženje korena, nazvanog* RTMI, *u kome se koristi metoda ekvidistantne inverzne paraboličke interpolacije. Napomenuto je da je ova metoda specijalni slučaj metode inverzne paraboličke aproksimacije. Prezentiran je potprogram* P *u kome se koristi hibridna metoda inverzna paraboličke aproksimacija – bisekcija. Pokazano je da je potprogram* P *efektivniji od potprograma* RTMI.

Some scientific subroutine packages[1-4] contain a root finding subroutine, call it RTMI, in which the method of equidistant inverse parabolic interpolation is used. This method is quadratically convergent[5]. It requires two function evaluations per iteration and has an efficiency index[6] 1.414... .

In this paper a subroutine is presented which is more effective than subroutine RTMI . The method of equidistant inverse parabolic interpolation is replaced by the method of inverse parabolic approximation[7]. The first method is a particular case of the second : if we use the method of inverse parabolic approximation as a "pure" interpolation method (omitting the extrapolation steps) , we obtain the method of inverse parabolic interpolation , and if the interpolation in the last method is equidistant , we obtain the method of equidistant inverse parabolic interpolation. The method of inverse parabolic approximation requires only one function evaluation per iteration. Its asymptotic convergence rate , as well as its efficiency index , is 1.839... . In the general case , as with all superlinear methods , it does not have a guaranteed convergence. To make an always-convergent algorithm it is combined with the bisection method . Thus , a hybrid algorithm is obtained which may be described in the following manner:

Suppose that a real root of

$$y(x)=0 \tag{1}$$

has been bracketed by initial approximations x_1 *and* x_2 . *Calculate* y_1 *and* y_2 , *and test if*

$$sign y_1 \neq sign y_2 \quad . \tag{2}$$

Set $x_0=x_1$. *Calculate* x_3 *by the bisection step*

$$x_3=0.5(x_1+x_2) \quad . \tag{3}$$

(a) *Calculate* y_3 . *If*

$$sign y_3 \neq sign y_2 \ , \tag{4}$$

set $x_0=x_2$. *Find* x_4 *by the inverse parabolic approximation step*

$$x_4=x_3+\left[\left(\frac{x_3-x_2}{y_3-y_2}-\frac{x_2-x_1}{y_2-y_1}\right)\frac{y_2}{(y_3-y_2)+(y_2-y_1)}-\frac{x_3-x_2}{y_3-y_2}\right]y_3 \tag{5}$$

provided that this point is between x_3 *and* x_0 . *Otherwise get* x_4 *by the bisection step*

$$x_4=0.5(x_0+x_3) \quad . \tag{6}$$

Replace (x_1,y_1) *by* (x_2,y_2) , (x_2,y_2) *by* (x_3,y_3) *and* x_3 *by* x_4 . *Return to* (a).

This means that method (5) is used whenever possible (as interpolating or extrapolating) , and only the latest three iterates are used even though they may be all on the same side of the root[8-10]. The opposite bracketing point x_0 and the current point x_3 are the latest iterates whose function values do not have the same sign. They are used in the auxiliary bisection step (6) only when (5) breaks down i.e. when it gives a result which is not in the interval bounded by x_3 and x_0 .

The following is a FORTRAN IV realization of this algorithm.

Description of parameters:

R - *Resultant root of the equation* F(X)=0.
Y - *Resultant function value at root* R.
F - *Name of the external function subprogram used.*
A - *Input value which specifies the initial approximation* x_1 .
B - *Input value which specifies the initial approximation* x_2 .
E - *Input value which specifies the upper bound of the relative error for successive iterates.*
M - *Maximum number of iteration steps permitted.*
I - *Resultant error parameter coded as follows:*
  I=1 - *Basic assumption* (2) *is not satisfied.*
  I=2,3,...,M - *No error.* I *is the number of iteration steps i.e. calls of subprogram* F.
  I=M+1 - *No convergence after* M *iteration steps.*

To show that subroutine P is more effective than subroutine RTMI a considerable number of examples were run on an IBM 1130 computer using floating-point arithmetic with 32-bit mantissa (extended precision). The results are given in table 1. A special subprogram for time measuring on IBM 1130 is used[11]. Subroutine P consumes 380 16-bit words in memory which is less than 552 of RTMI.

REFERENCES

1. *1130 Scientific Subroutine Package* (1130-CM-02X) , Programmer's Manual, IBM Corp., New York, 1968, 117--119.

2. *System/360 Scientific Subroutine Package* (360A-CM--03X), Version III, Programmer's Manual, IBM Corp., New York, 1968, 217-219.

3. *Scientific Subroutines Package* , Reference Manual , DEC-11-SSPMA-A-D , Digital Equipment Corp., Maynard, Massachusetts, 1973, 3/44.

4. *Scientific Subroutines*, Reference Manual , AA-1101B--TC , Digital Equipment Corporation , Maynard, Massachusetts, November 1978, 3/45.

5. Ll.G.Chambers, *A Quadratic Formula for Finding the Root of an Equation* , Mathematics of Computation, 25 (1971), 305-307.

6. A.M.Ostrowski, *Solution of Equations in Euclidean and Banach Spaces* , third edition of Solution of Equations and Systems of Equations , Academic Press, New York and London, 1973, 32.

7. J.F.Traub, *Iterative Methods for the Solution of Equations* , Prentice-Hall , Englewood-Cliffs, New Jersey, 1964, 233.

8. R.F.King, *Methods without Secant Steps for Finding a Bracketed Root*, Computing, 17(1976), 49-57.

9. D.B.Popovski, *A Fortran IV Subroutine for Finding a Bracketed Root*, Informatica, 4/1979, 23-24.

10. D.B.Popovski, *A Root Finding Algorithm* , Prilozi (Association for Science and Art - Bitola) , 30-31 (1979), 289-293.

11. Computing Laboratory , Department of Electrical Engineering , University of Belgrade : personal communication.


```
      SUBROUTINE P(R,Y,F,A,B,E,M,I)
      I=2
      C=A
      D=F(C)
      G=B
      H=F(G)
      IF(D)1,18,2
    1 IF(H)19,20,4
    2 IF(H)4,20,19
    3 G=Q
      H=U
    4 Q=H-D
      Q=(C+G)*0.5
      S=Q-G
      T=(G-C)/Q
      GO TO 7
    5 Q=(C+G)*0.5
      S=Q-G
    6 Q=V
      H=U
    7 U=F(Q)
      I=I+1
      IF(I-M)8,21,21
    8 V=U-H
      IF(U)9,24,10
    9 IF(H)11,24,12
   10 IF(H)12,24,11
   11 IF(V)13,3,13
   12 C=G
      D=H
   13 G=Q
      Q=T
      T=S/V
      Q=V+Q
      IF(Q)14,5,14
   14 S=((T-Q)*H/Q-T)*U
      Q=G+S
      IF(Q-C)15,5,16
   15 IF(S)5,5,17
   16 IF(S)17,5,5
   17 IF(ABS(Q)*E-ABS(S))6,22,22
   18 R=C
      Y=D
      RETURN
   19 I=1
   20 R=G
      Y=H
      RETURN
   21 IF(U)23,24,23
   22 U=F(Q)
   23 I=I+1
   24 R=Q
      Y=U
      RETURN
      END
```

Table 1. Numerical examples: E=(EPS)=1.E-5, M=(IEND)=30, (IER)=0
(In parenthesis are the parameters of subroutine RTMI).

F (FCT)	A (XLI)	B (XRI)	R (X)	Y	(F)	I	Execution time (s) RTMI	P
((X-14.)*X+57.)*X-65.	0.	3.	1.92...	0.59E-7	0.00E 0	9	0.170	0.144
	3.	5.	4.39...	0.59E-7	-0.11E-6	7	0.176	0.103
	5.	8.	7.68...	0.29E-6	0.17E-6	9	0.198	0.144
(((X-20.)*X+133.)*X-330.)*X+236.	0.	2.	1.18...	0.23E-6	0.00E 0	8	0.188	0.137
	2.	4.	3.42...	-0.23E-6	0.47E-6	8	0.193	0.137
	4.	8.	6.57...	0.14E-5	0.21E-5	9	0.217	0.153
	8.	9.	8.81...	-0.21E-5	-0.81E-5	8	0.196	0.139
((((X-4.)*X-66.)*X+196.)*X+713.)*X-1000.	-9.	-5.	-6.97...	-0.12E-4	-0.31E-4	7	0.165	0.125
	-5.	-3.	-3.11...	0.95E-6	-0.21E-4	7	0.208	0.125
	-3.	3.	1.17...	0.00E 0	-0.95E-6	8	0.231	0.148
	3.	5.	4.85...	0.57E-5	-0.28E-5	8	0.232	0.149
	5.	9.	8.04...	-0.33E-4	-0.33E-4	8	0.187	0.149
EXP(1.-X)*(X-1.)*10.-1.	0.	2.	1.11...	0.00E 0	-0.74E-8	9	0.235	0.191
	2.	5.	4.57...	0.18E-8	-0.27E-8	8	0.283	0.166
ALOG((0.25*X-2.)*X+4.5)-1.	0.	2.	1.02...	0.93E-9	0.00E 0	6	0.146	0.133
	2.	9.	6.97...	0.93E-9	-0.27E-8	8	0.237	0.191
(0.04*X-0.4)*X+0.5-SIN(X)	-1.	1.	0.36...	-0.23E-9	-0.93E-9	8	0.227	0.164
	1.	5.	3.57...	0.93E-9	0.93E-9	9	0.241	0.197
	5.	6.	5.78...	0.00E 0	-0.32E-8	7	0.241	0.146
	6.	10.	9.21...	-0.90E-8	0.25E-8	8	0.267	0.171
						Total	4.238	3.012
						Index	141	100

PRIMERJALNA ANALIZA POUKA RAČUNALNIŠTVA NA SREDNJI STOPNJI IZOBRAŽEVANJA

J. BENKOVIČ,
A. KORNHAUSER
in
V. RAJKOVIČ

FAKULTETA ZA NARAVOSLOVJE IN TEHNOLOGIJO – RCPU, LJUBLJANA
VISOKA ŠOLA ZA ORGANIZACIJO DELA, KRANJ

UDK: 373 : 681.3

Probleme pri uvajanju pouka računalništva v srednjem izobraževanju v Sloveniji skuša reševati interdisciplinarno sestavljena raziskovalna skupina v okviru večletnega razvojno-raziskovalnega projekta v ta namen. Cilj tega projekta je izdelava okvirnega vzgojnoizobraževalnega modela za pouk računalništva na srednji stopnji usmerjenega izobraževanja, njegova evaluacija v praksi ter priprava predloga sistema izobraževanja učiteljev računalništva za ta program.
Delo, o katerem je bilo poročano vsako leto (J.Benkovič, A.Kornhauser, V.Rajkovič et al., 1977, 1978 in 1979), je bilo zaključeno s komparativno analizo s pomočjo vprašalnika, intervjuji učiteljev na seminarjih in z neposrednim sodelovanjem raziskovalcev v delu na šolah. Ta analiza je zajela doseganje vzojnoizobraževalnih smotrov, izbor tem, njih obseg in časovno razporeditev vsebine, zahtevnost programa za učence in učitelje ter podrobno analizo posameznih tem po teh kriterijih. Končno je opredelila še ustreznost sistema izobraževanja učiteljev računalništva. Služila naj bi kot izhodišče za programiranje specifičnega pouka računalništva na različnih smereh in usmeritvah srednje stopnje usmerjenega izobraževanja.

COMPARATIVE ANALYSIS OF COMPUTER TEACHING AT SECONDARY LEVEL: Problems encountered in the introduction of computer teaching into secondary education were investigated by an interdisciplinary research team within a developmental research project over several years. The aim of the project was to design a broad educational model for computer teaching at the secondary level of vacationally oriented education, to evaluate it in practice and to prepare a proposal for a system of in-service training of computer science teachers for this programme.
The project, on which annual reports were made (J.Benkovič, A.Kornhauser, V.Rajkovič et al., 1977, 1978,1979), also includes a comparative analysis using a questionnaire, interviews of teachers during seminars and direct involvement of researchers in computer teaching in schools. The analysis evaluates the achievement of educational objectives, the selection of topics, their scope, a work schedule of the content of the programme, the level of the programme vis á vis students and teachers, and a detailed analysis of individual topics according to the above criteria. Finally, the adequacy of the system of in-service training of computer science teachers was also assessed. The aim of the analysis was to provide a starting point for setting up a programme for specific computer teaching in the different types of vocationally oriented secondary education.

Računalništvo je postalo neogibni pogoj pretežnega števila dejavnosti, zato ga je nujno uvajati tudi na nižje ravni izobraževanja. V Sloveniji smo začeli s poukom računalništva na srednji stopnji izobraževanja že pred desetimi leti,vendar postopno in v šole, ki so se za to prijavile. V tem desetletju lahko zabeležimo nekaj uspehov: v skoraj vse štiriletne srednje šole je bilo vpeljano računalništvo, najsi bo kot posebni predmet s tem imenom, ali pa kot program v okviru praktičnih znanj.

Uvajanje pouka računalništva v srednje šole pa je seveda odprlo vrsto problemov, zato je bil osnovan razvojno-raziskovalni projekt z namenom reševati te probleme v povezavi s prakso in pripravljati temelj za računalništvo v usmerjenem izobraževanju.

Smotri projekta so bili tako izobraževalni kot vzgojni, saj računalništvo odpira široke možnosti za doseganje obojih. Končni cilj pa je bil izdelava vzgojnoizobraževalnega modela za pouk računalništva na srednji stopnji izobraževanja učiteljev za ta program.

Raziskovalna skupina je bila sestavljena tako, da je zagotavljala interdisciplinarno povezavo strokovnjakov na področju računalništva, matematike, naravoslovja in tehnologije, zajela pa je tudi uporabnike. Ker je bil projekt razvojno-raziskovalne narave, so v njem kot partnerji sodelovali številni učitelji srednjih šol - vseh tistih, ki so bile pripravljene ne le uvajati računalništvo v svoje programe, temveč tudi intenzivno sodelovati pri zasnovi, izvedbi in evaluaciji programov. To so bili: M.Andolšek, Šolski center Radovljica, M.Budna, Gimnazija Velenje, M.Cokan, Šola za oblikovanje Ljubljana, V.Elvič-Volk, Ekonomsko-administrativna šola Maribor, A.Kališnik, Rudarski šolski center Velenje, J.Karčnik, Šolski center Idrija, M.Koren, Ekonomsko-administrativna šola Maribor, M.Kovač, Tehniška šola za strojništvo Ljubljana, A.Kramer, Gimnazija Ivan Cankar Ljubljana, J.Pavlišič, Center srednjih šol Črnomelj, E.Skočir, Gimnazija-ekonomska šola Trbovlje, N.Šilc, Gimnazija Celje, M.Žigman, Ekonomska srednja šola Ljubljana, F.Žigon, Gozdarski šolski center Postojna. Poleg njih je občasno sodelovala pri delu še vrsta učiteljev računalništva na drugih šolah, tako da skupno število teh sodelavcev dosega 70.

Zaključek dela predstavlja komparativna analiza številnih parametrov pouka računalništva na srednji stopnji izobraževanja, o kateri poročamo v nadaljnjem.

1. CILJ IN METODA ANALIZE

Cilj analize je bil primerjati pouk računalništva na srednji stopnji izobraževanja pred uvajanjem rezultatov projekta za razvoj pouka računalništva v usmerjenem izobraževanju in po njihovem uvajanju. Pri tem gre zlasti za naslednje rezultate projekta:
- novi učni načrt,
- zasnova didaktičnega kompleta,
- zasnova modela izobraževanja učiteljev,
- izdelava navodil za učitelje računalništva za izvajanje pouka in
- priporočila v zvezi z učnimi pripomočki.

Metoda dela je obsegala zlasti naslednje elemente:
- vprašalnik za učitelje računalništva,
- poročila sodelavcev - učiteljev na šolah, ki zajemajo časovno razporeditev učne snovi, sekvenco primerov algoritmov in vaj ter opis didaktičnih problemov s posebnim poudarkom na praktičnem delu,
- načrtno zasnovane razgovore z učitelji računalništva pri neposrednem delu na šolah, na aktivih in seminarjih.

Zgoraj navedeni elementi metode dela niso uporabljeni le enkratno, ampak predstavljajo permanenten in povezan proces. Učitelji kontinuirano sodelujejo že tri do štiri leta. Posamezni elementi se med seboj vsebinsko prepletajo in dopolnjujejo. Tako npr. na seminarjih v skupinskem sodelovanju raziskovalcev in učiteljev tolmačimo odgovore na vprašanja v vprašalniku in poročila učiteljev. Pri tem delu običajno sodeluje le 10 do 30 učiteljev, dasiravno je bil npr. vprašalnik razposlan na 60 naslovov. Slabostim, ki jih prinaša relativno majhen vzorec, se skušamo tako izogniti s poglobljeno razpravo, ki bistveno prispeva k jasnosti stališč. Zlasti pomembni so intervjuji učiteljev v neposrednih situacijah pri pouku.

2. VPRAŠALNIK IN ODGOVORI

Vprašalnik je bil razdeljen v štiri sklope, ki so zajeli:
1. oceno vrednosti teme za doseganje učnovzgojnih smotrov,
2. čas za obravnavo teme,
3. zahtevnost programa (teme) za učence,
4. zahtevnost programa (teme) za učitelje.

2.1. Ocena vrednosti teme za doseganje učnovzgojnih smotrov

Posamezne teme iz učnega načrta za predmet računalništvo je bilo treba ovrednotiti glede na pomembnost za doseganje naslednjih smotrov:

A. Učenci se seznanjajo z osnovnimi principi delovanja računalnika in dobivajo vpogled

Tema \ Smotri		A	B	C	D	E	F	agregirana ocena pomem.
1.	Pojem računalnika, algoritma in programa	●	●	◒			F2	2.5
2.	Kratek pregled zgodovine računalnikov	◒			●	○	F1 F2	1.75
3.	Smotri pouka računalništva	○			○	◒		1.0
4.	Pomen kibernetike, organizacije, upravljanja in računalništva v sodobnem času	◒			●	◒	F1	2.0
5.	Pojem informacije	◒			●			1.5
6.	Predstavitev informacije	●		○	●			2.25
7.	Analogni in digitalni računalniki	●		○	○	○		1.75
8.	Binarna predstavitev informacije	●	○	○	○			1.75
9.	Zapis števil in alfanumeričnih znakov v računalništvu	●	◒	○	○			2.0
10.	Osnovne enote računalnika	●			○	◒		1.75
11.	Matematično tehnične osnove računalnika	●	◒	○	○	○		2.25
12.	Preprost model računalnika in zbirni jezik	●	●	◒	○			2.75
13.	Odvijanje programa v računalniku	●	◒	◒	○			2.25
14.	Osnovni pojmi o računalniških sistemih	●	○	○	◒	○	F2	2.25
15.	Pojem jezika	●	○	◒	○			2.0
16.	Pomen, vloga in vrste programskih jezikov	◒	○	◒	◒	◒	F2 F3	2.25
17.	Prevajanje programskih jezikov	●		○	○		F2	1.5
18.	Sistematični pristop k reševanju problema	◒	●	●	○			2.75
19.	Razvoj in zapis algoritma za reševanje problema	◒	●	●		○	F1	2.75
20.	Programski jezik	◒	●	●	○	○	F1	3.0
21.	Zgledi uporabe	●	●	●	●	◒	F3	4.5
22.	Mesto in vloga računalniškega znanja v poklicih	◒			○	◒	F3	1.75

Tabela 1: OCENA VREDNOSTI TEME ZA DOSEGANJE UČNOVZGOJNIH SMOTROV

v številne možnosti njegove uporabe s posebnim poudarkom na izbrani smeri.

B. Učenci razvijajo algoritmično mišljenje ter natančnost in sistematičnost pri reševanju problemov.

C. Učenci se usposabljajo in spodbujajo za reševanje preprostih problemov z računalnikom.

D. Učenci pridobivajo vpogled v mesto in vlogo informatike, kibernetike, organizacije in upravljanja pri človekovih aktivnostih.

E. Učenci se seznanjajo z mestom in vlogo računalniškega znanja v računalniških poklicih.

Poleg zgornjih petih smotrov iz učnega načrta so učitelji lahko dodali tudi specialne smotre glede na svoje potrebe. Dodane smotre bi lahko strnili zlasti v tri:

F1. Učenci krepijo samozavest.

F2. Učenci spoznavajo mesto računalniških znanj v razvoju znanosti in tehnike.

F3. Učenci razvijajo sposobnosti za sprejemanje novih dosežkov v zvezi s hitrim razvojem disciplin.

Ocena vrednosti teme za doseganje učno-vzgojnih smotrov je podana v tabeli 1. Ocena lahko zavzame štiri vrednosti:

● zelo pomembno

◕ pomembno

○ manj pomembno

– ni v povezavi s smotrom

Številke ob desnem robu tabele, ki pomenijo združeno oceno pomembnosti teme, smo dobili tako, da smo izračunali vsoto parcialnih pomembnosti, kjer poln krog okvirno pomeni 1, polovičen 0.5 in prazen 0,25.

2.2. Čas za obravnavo teme

K posameznim temam je bilo treba vpisati dejanski porabljeni čas za izvajanje teme in predlog števila ur za morebitni novi učni načrt.

Srednja vrednost dejansko porabljenih in predlaganih ur skupaj s planom po učnem načrtu je podana v tabeli 2.

Tema \ Ure	PLAN 54	DEJ. 58	PRED. 68.5
1. Pojem računalnika, algoritma in programa	2.0	2.5	3.0
2. Kratek pregled zgodovine računalnikov	1.0	1.0	1.5
3. Smotri pouka računalništva	1.0	1.0	1.0
4. Pomen kibernetike, organizacije, upravljanja in računalništva v sodobnem času	2.0	1.5	1.5
5. Pojem informacije	0.5	1.0	1.0
6. Predstavitev informacije	0.5	1.0	1.0
7. Analogni in digitalni računalnik	0.5	0.5	1.0
8. Binarna predstavitev informacije	1.0	1.5	2.0
9. Zapis števil in alfanumeričnih znakov v računalništvu	1.5	1.0	1.5
10. Osnovne enote računalnika	3.0	3.0	3.5
11. Matematično tehnične osnove računalnika	2.0	2.0	*
12. Preprost model računalnika in zbirni jezik	1.0	1.0	1.5
13. Odvijanje programa v računalniku	1.0	1.5	2.0
14. Osnovni pojmi o računalniških sistemih	1.0	1.0	1.0
15. Pojem jezika	0.5	1.0	1.0
16. Pomen, vloga in vrste programskih jezikov	0.5	1.0	1.0
17. Prevajanje programskih jezikov	1.0	1.0	1.0
18. Sistematični pristop k reševanju problema	1.0	2.0	2.5
19. Razvoj in zapis algoritma za reševanje problema	6.0	4.5	5.0
20. Programski jezik	25.0	26.0	29.0
21. Zgledi uporabe	1.0	2.0	3.0
22. Mesto in vloga računalniškega znanja v poklicih	1.0	1.0	2.0

Tabela 2: ČAS ZA OBRAVNAVO TEME

* družbeno-ekonomska usmeritev: 0 ur,
naravoslovno-matematična usmeritev: 1 ura,
proizvodno-tehnična usmeritev: 4 ure.

2.3. Zahtevnost programa za učence

Zahtevnost posamezne teme za učence smo skušali opredeliti na osnovi kognitivne taksonomije po Bloomu (Bloom,B.S. 1970), za katero smo uporabili okrajšave:

1. SZ spominsko znanje,kot je poznavanje podatkov, definicij in podobno, brez globljega razumevanja ali sposobnosti uporabe - nekakšno naučeno znanje;
2. R razumevanje, torej znanje, ki ga učenec dojema, globlje razume;
3. U uporaba znanja, kar pomeni, da je učenec znanje osvojil, ga razume in zna uporabljati;
4. VK višje kategorije znanja, kamor spada analitično in sintetično mišljenje, kjer učenec zna analizirati določeno znanje na njegove sestavine in to povezati v večje celote, pa tudi sposobnost evaluacije znanja.

Uvedli pa smo še kognitivno kategorijo po Gagneju (Gagne,R.M. 1970):

5. RP reševanje problema, ki je opredeljeno na osnovi učenčevega dela, kadar se sreča s problemi, ki jih mora reševati s povezovanjem zgornjih štirih spoznavnih kategorij.

Rezultati so strnjeno prikazani v tabeli 3, kjer pomeni:

- ● zelo zahtevno
- ◪ zahtevno
- ○ manj zahtevno
- — stopnje zahtevnosti ni mogoče izraziti (npr. zaradi nezadostne globine teme, ki je bila glede na čas in navodila v učnem načrtu dosežena).

2.4. Zahtevnost programa za učitelje

Za posamezno temo je bilo treba opredeliti zahtevnost glede na:

- operacionalizacijo ciljev (OC),
- učno snov (SNOV),
- terminologijo (TER),
- metode (MET),
- opremo, učila (UČILA),
- evaluacijo (EVAL).

Rezultati so strnjeno prikazani v tabeli 4, kjer pomeni:

- ● zelo zahtevno
- ◪ zahtevno
- ○ manj zahtevno

Tema		SZ	R	U	VK	RP
1.	Pojem računalnika, algoritma in programa	○	◪	◪		
2.	Kratek pregled zgodovine računalništva	◪	○			
3.	Smotri pouka računalništva		○	○		
4.	Pomen kibernetike, organizacije, upravljanja in računalništva v sodobnem času	○	◪	●	●	
5.	Pojem informacije	○	◪	◪	●	
6.	Predstavitev informacije	○	◪	◪	◪	◪
7.	Analogni in digitalni računalnik	○	◪	○		
8.	Binarna predstavitev informacije	○	◪	○	◪	
9.	Zapis števil in alfanumeričnih znakov v računalništvu	○	●	◪	◪	
10.	Osnovne enote računalnika	○	○	○		
11.	Matematično tehnične osnove računalnika	○	●	○	◪	
12.	Preprost model računalnika in zbirni jezik	◪	●	◪	○	
13.	Odvijanje programa v računalniku	○	●	◪	◪	◪
14.	Osnovni pojmi o računalniških sistemih	○	◪	○		
15.	Pojem jezika	○	◪	○		
16.	Pomen, vloga in vrste programskih jezikov	○	○	○		
17.	Prevajanje programskih jezikov	○	◪	○		
18.	Sistematični pristop k reševanju problema	○	●	●	●	●
19.	Razvoj in zapis algoritma za reševanje problema	○	●	●	●	●
20.	Programski jezik	◪	◪	●	●	●
21.	Zgledi uporabe	○	◪	◪		
22.	Mesto in vloga računalniškega znanja v poklicih	○	◪	◪		

Tabela 3: ZAHTEVNOST PROGRAMA ZA UČENCE

Tema		OC	SMOV	TER	MET	UČILA	EVAL
1.	Pojem računalnika, algoritma in programa	◕	○	○	◕	◕	◕
2.	Kratek pregled zgodovine računalništva	○	○	○	○	◕	○
3.	Smotri pouka računalništva	◕	○	○	◕	○	○
4.	Pomen kibernetike, organizacije, upravljanja in računalništva v sodobnem času	●	●	○	◕	○	●
5.	Pojem informacije	●	◕	◕	◕	○	◕
6.	Predstavitev informacije	◕	◕	◕	○	○	○
7.	Analogni in digitalni računalnik	○	◕	◕	○	○	○
8.	Binarna predstavitev informacije	◕	○	○	○	○	○
9.	Zapis števil in alfanumeričnih znakov v računalništvu	●	◕	○	◕	○	○
10.	Osnovne enote računalnika	○	○	○	○	○	○
11.	Matematično tehnične osnove računalnika	◕	●	○	◕	◕	○
12.	Preprost model računalnika in zbirni jezik	◕	●	◕	◕	◕	○
13.	Odvijanje programa v računalniku	◕	●	○	◕	◕	○
14.	Osnovni pojmi o računalniških sistemih	○	◕	○	○	○	○
15.	Pojem jezika	○	○	○	○	○	○
16.	Pomen, vloga in vrste programskih jezikov	○	◕	○	○	○	○
17.	Prevajanje programskih jezikov	◕	●	○	◕	○	○
18.	Sistematični pristop k reševanju problema	●	◕	○	●	○	◕
19.	Razvoj in zapis algoritma za reševanje problema	●	●	○	●	○	◕
20.	Programski jezik	●	●	◕	●	◕	◕
21.	Zgledi uporabe	◕	◕	○	○	◕	○
22.	Mesto in vloga računalniškega znanja v poklicih	◕	◕	○	○	○	◕

Tabela 4: ZAHTEVNOST PROGRAMA ZA UČITELJE

3. REZULTATI EVALUACIJE

Rezultati evaluacije so podani po temah za vsa štiri področja: doseganje učnovzgojnih smotrov, čas za obravnavo teme, zahtevnost programa za učence ter zahtevnost programa za učitelje.

3.1. Pojem računalnika, algoritma in programa

Ta tema zajema prikaz računalnika v praksi, pojem algoritma in programa s primeri diagramov poteka ter definicijo računalnika s pojmi računalnik, vhod, obdelava, izhod.

Anketiranci so ocenili, da je to pomembna tema in ji prisodili oceno 2.5. Potrebna je tako za doseganje smotrov pri obravnavi splošnih računalniških znanj kot za uvajanje v reševanje problemov s pomočjo računalnika. Menijo, da je odmerjeni čas za to uvodno temo preskromen, posebej še zaradi zahtevnosti prvega uvajanja v metode in učila. Tema vključuje strukturiranje operacij, torej logiko, to pa se še poglobi v temah 18,19 in 20. Predlagano je povečanje odmerjenega časa z dveh na tri ure.

3.2. Razvoj računalnikov

Tema podaja kratek pregled razvoja računalnikov. V splošnem je manj zahtevna, vendar dokaj privlačna. Ocena je 1.75. Prispeva predvsem k razumevanju dinamike razvoja računalništva in s tem k realni oceni današnjega stanja, ki je le člen v verigi razvoja. Iz tega izhaja tudi potreba po stalnem strokovnem izpopolnjevanju.

Učni pripomočki so tu vrzel. Potrebno je predvsem ustrezno slikovno gradivo, zlasti na prosojnicah. Predlagani čas za to temo je 1.5 ure.

3.3. Smotri pouka računalništva

Smotri za okvir te učne enote so navedeni pod 2.1. Ocena te teme v vrednostjo 1.0 kaže, da učitelji še zelo malo poznajo pomen smotrov oz. njihovo operacionalizacijo skozi vsebino, metode in tehnike vzgojnoizobraževalnega dela.

Gre za poskus, da bi učitelji skupaj z učenci našli primere uresničevanja smotrov, navedenih pod 2.1. Pokazalo pa se je, da je ta tema uvrščena predvsem prezgodaj, ko učenci in učitelji še nimajo dovolj gradiva za take primere, pa tudi, da ne prvi ne drugi niso navajeni razmišljati o smotrih svojega pouka in učenja in se uče le vsebine ter v manjši meri tudi metod in tehnik.

Bistvena pomanjkljivost je tudi v učbeniku in drugih gradivih v tem, da je premalo življenjskih primerov, v katerih učenci ne bi le uporabljali vsebine znanja, temveč tudi metode in tehnike ter tako uresničevali procesne cilje vzgojnoizobraževalnega programa v realnih

situacijah, ki bi najbolj prispevale k doživljanju realizacije smotrov. Gre zlasti za metodo reševanja realnih problemov ter razvijanje raziskovalno-izobraževalnega pristopa. Te metode pa tudi učitelji še slabo poznajo, zato je treba načrtovati permanentno izobraževanje učiteljev zlasti v tej smeri.

3.4. Pomen kibernetike, organizacije, upravljanja in računalništva v procesih dela in življenja

Ta tema je nova in še ni zajeta v učbeniku. Pokazalo se je, da brez podpore učbenika učitelji zanjo niso mogli najti primerne vsebine in metod. To tudi razloži nizko oceno 2.0. Spet bi morali prikazati realne primere po posameznih navedenih področjih, vendar jim le-ti očitno manjkajo, zato so tudi namenili temi le 1.5 ure, kar je manj, kot je bilo predvideno (2 uri). Temu ustrezne so tudi težave učencev. Krivdo pripisujejo pomanjkanju gradiv in pripomočkov (ocena 3.6), ki bi jim pomagali temo ustrezno uresničiti. Nujno potrebujejo realne, kvalitetne primere. Analiza tudi v tem primeru jasno kaže, da učitelji le malo samostojno sledijo razvoju računalništva ter se predvsem zanašajo na učbenik. Gotovo pa je temu krivo tudi dejstvo, da imamo v domačih jezikih le malo literature, ki bi jim lahko učinkovito pomagala pri samoizobraževanju.

3.5. Pojem informacije

Ta uvodna tema za sklop naslednjih štirih zajema osnovne pojme iz teorije informacij. Po oceni iz primerjalne analize je ta tema premalo konkretna. To se negativno odraža v operacionalizaciji ciljev, saj učitelj nima realnih situacij, na katere bi gradil uporabo definicij. Predlog je, da bi temo močneje povezali z naslednjo, ki daje take možnosti. Za ločeno temo so učitelji dali oceno vsega 1.5 in jo omejili na čas 1 ure.

3.6. Predstavitev informacij

Kot je omenjeno zgoraj, je tema nadaljevanje prejšnje ter obravnava vrste predstavitve in načine zapisa informacije. Njen pomen presega okvire predmeta računalništvo. Ocena je 2.25. Zanimivo je, da pri tej temi ni opaziti večjih težav ne pri učiteljih, ne pri učencih, verjetno zato, ker je dobro strukturirana. Predlagani čas je 1 ura.

3.7. Analogni in digitalni računalniki

Tema opisuje analogne in digitalne računalnike in zlasti poudarja skupne značilnosti in razlike. Gre predvsem za informacije o računalnikih, torej za vsebinske pristope. Zato je tema tudi dobila nizko oceno 1.75. Opozoriti kaže, da bi ob ustrezni predelavi tema lahko služila za doseganje višjih kognitivnih kategorij, zlasti na ravni komparativne analize. Potrebni čas je 1 ura.

3.8. Binarna predstavitev informacij

Tema zajema predstavitev informacij v binarnem sistemu in predstavlja uvod v naslednjo temo. Nizka ocena 1.75 kaže na težave, ki verjetno znova izvirajo iz dejstva, da je premalo razumljivih praktičnih primerov, čeprav je nekaj podanih. V tem smislu lahko razumemo tudi predlog, da je treba temi posvetiti 2 uri.

3.9. Zapis števil in alfanumeričnih znakov v računalniku

Ta sklop obravnava predstavitev numerične in zatem nenumerične informacije v računalniku in je ključna tema za razumevanje delovanja računalnika kot stroja za avtomatsko obdelavo podatkov, ne pa le kot računskega stroja. Dobljena ocena 2.0 kaže, da tema še ni ustrezno izvajana. Gre zlasti za to, da učenci to snov teže razumejo, kar je razumljivo, saj gre za večjo stopnjo abstrakcije. Predvsem je nujno usvojeno znanje tudi razumeti, sicer učenci tema ne bodo mogli uporabljati, kar je zahteva naslednjih tem. Težave učencev pa so seveda pogojene s težavami učiteljev, ki jim manjka metodika za operacionalizacijo ciljev, zlasti takrat, kadar gre za premik snovi od konkretne proti abstraktni. Predlagani čas je 1.5 ure.

3.10. Osnovne enote računalnika

To je dokaj obsežna tema, ki vključuje shematični prikaz zgradbe računalnika, splošne pojme o vhodnih in izhodnih enotah (elektronski pisalni stroj, tiskalnik, čitalnik in luknjalnik kartic, katodni zaslon in svetlobno pisalo, čitalnik in luknjalnik traku, risalnik, optični čitalnik), dalje različne izvedbe pomnilnika, aritmetično enoto in krmilno enoto. Kljub obsežnosti je ta tema za učence privlačna, predvsem zato, ker podaja pretežno dejstva in pojme in ne terja višjih kognitivnih dosežkov. Predvsem je konkretne narave in zato učenci nimajo težav z abstrakcijo. Ocena je 1.75 in čas 3.5 ure. Bistveno lahko prispevajo k večji kvaliteti zlasti ogledi računalniških naprav ob njihovem delovanju ali pri vzdrževalnih delih.

3.11. Matematično-tehnične osnove računalnika

Ta tema vključuje pojem Boolove algebre, definicije konjunkcije, disjunkcije in negacije ter osnovnih pet aksiomov. Zatem preide na preklopno algebro in končno na logična vezja.

Komparativna analiza je pokazala, da učitelji na družboslovno-ekonomski usmeritvi to temo izpuščajo, kar kaže obžalovati, saj so zlasti logična vezja lahko tudi osnova strukturiranja znanja v družboslovnih in še posebej v ekonomskih disciplinah. Ugotoviti pa je treba, da je tudi na naravoslovnem področju tema bolj površno obdelana, ocena 1.25. Le proizvodno-tehnična usmeritev ji posveča več pozornosti (ocena 2.75). Povprečna ocena je 2.25.

Tema je zlasti primerna za preverjanje, ali smo dosegli stopnjo razumevanja, omogoča pa tudi doseganje prej omenjenih višjih kognitivnih kategorij. Učitelji ugotavljajo zahtevnost snovi. Predlagani čas je za proizvodno-tehnično usmeritev 4 ure, za naravoslovno-matematično 1 ura, za družboslovno-ekonomsko pa nič. To kaže na potrebo po poglobitvi dela z učitelji, zlasti v smeri strukturiranja znanja.

3.12. Preprost model računalnika in zbirni jezik

Tema skuša prikazati, kako računalnik deluje in vzroke za tako delovanje. Obenem nudi primer programa v zbirnem jeziku. Tema je visoko ocenjena z 2.5. Ugotovljeno pa je, da je snov zahtevna tako za učitelja kot za učence. Pri tem gre posebej za težave z razumevanjem zapisa števil in alfanumeričnih znakov. Pogoj za uspeh pri tej tematiki je razumevanje snovi. Predlagani čas je 2 uri.

3.13. Odvijanje programa v računalniku

Tema obsega vnašanje programa v računalnik ter izvajanje programa v računalniku s shematskimi prikazi posameznih stopenj. Pri tem še posegamo na stopnjo reševanja problemov, kar poveča zahtevnost obravnavane snovi. Ocena je 2.25. Po učnem načrtu je bila predvidena 1 ura, učitelji pa predlagajo 2 uri, saj sedaj porabijo za to najmanj uro in pol. Terjano povečanje časa ute-

meljujejo s potrebo po doseganju razumevanja in tudi višjih kognitivnih kategorij, vsaj stopnje analize. Celo za učitelje je po njihovem mnenju snov zelo zahtevna.

3.14. Osnovni pojmi o računalniških sistemih

Tema podaja pojem sistema in konfiguracije, kapacitete pomnilnika, možnost konfiguracij računalnika ter pojem računalniške mreže. Sledе opisi načinov delovanja računalnika ter osnove operacijskih sistemov. Tema zaključuje z obravnavo možnosti napak v računalniku s stališča zanesljivosti njegovega delovanja.

Povprečna ocena je 2.25, predlagani čas 1 ura ustreza. Učenci imajo le delne težave z razumevanjem snovi in to velja tudi za učitelje. Ugotavljamo visoko stopnjo motivacije, ki jo zlasti dosegamo s praktičnim delom na računalniških sistemih.

3.15. Pojem jezika

Tema najprej opredeli naravne jezike, sintakso in semantiko. Zatem preide na umetne jezike in poudari zlasti programske. Ker gre predvsem za definicije, je to spominsko zahtevna tema, ki pa večini ne predstavlja večjih težav - najbrž zato, ker se veže na znanje iz slovenskega in tujih jezikov. Predlagani čas je 1 ura, povprečna ocena pa 2.0.

3.16. Pomen, vloga in vrste programskih jezikov

V tem sklopu obravnavamo vlogo programskega jezika in odnosa: programski jeziki in človek ter programski jeziki in računalnik. Obdela hierarhijo jezikov. Ocena je 2.25, predlagani čas je 1 ura. To spet kaže, da tema - morda prav zaradi poznavanja naravnih jezikov - ne predstavlja večjih težav.

3.17. Prevajanje programskih jezikov

Tema pokaže zaenkrat še omejene možnosti računalnika pri komuniciranju s človekom in opredeli prevajalni program kot prevajalnik ali kompilator. Zatem to podkrepi s primerom prehoda od programa v višjem programskem jeziku preko strojnega jezika do terjanih rezultatov.

Pomembnost teme je ocenjena sorazmerno nizko z 1.5. Po mnenju učiteljev 1 ura v ta namen ustreza, vendar ugotavljajo veliko zahtevnost snovi. Ta ugotovitev pa se manj odraža pri učencih, ki imajo le delne težave z razumevanjem.

3.18. Sistematični pristop k reševanju problema

Tema skuša razložiti, kako poteka reševanje problemov z računalnikom in postavi šest stopenj v tem reševanju: 1. definicijo problema, 2. grobo zamisel ustreznega algoritma, 3. podrobno izdelavo algoritma, 4. pisanje programa, 5. testiranje programa in 6. končno izvršitev programa. Tema se nadaljuje v naslednjih štirih sklopih.

Ta tema je bistvenega pomena in je dobila oceno 2.75. Kaže pa, da je zelo trd oreh za učence in učitelje. Namesto planirane 1 ure predlagajo učitelji 2.5 ure, že sedaj pa porabijo zanjo najmanj 2 uri. To kaže, da učitelji nimajo vajeni metode reševanja problemov pri svojem delu oziroma, da je ta v šolah le malo v rabi. Težko je dobro definirati probleme, kaj šele nadaljevati delo po naslednjih stopnjah. Pri definiciji je treba ugotoviti, v čem je bistvo problema in nato začrtati njegovo reševanje tako, da bi lahko načrtno zbirali in urejali podatke. Naslednja težava je postavljanje hipotez o medsebojnih odvisnostih urejenih podatkov, ki bi jih zatem skušali preverjati.

Kljub takemu stanju je na temi treba vztrajati, saj morajo učitelji računalništva tu popravljati ne le, kar je bilo pri drugih predmetih zanemarjenega, temveč tudi svoje lastno izobraževanje, ki je očitno redko temeljilo na metodi reševanja problemov - ne glede na to, da vsak delavec v praksi mora reševati probleme.

Preverjanje hipotez kot zadnja, a ključna faza metode reševanja problemov pa je mnogokrat kar izpuščeno. Tako učenci kot učitelji se preveč radi zadovoljijo s hipotezo kot končnim rezultatom, kar je seveda v nasprotju s potrebami usmerjenega izobraževanja za realne razmere dela in vsakdanjega življenja. Ta tema potrebuje vsaj 2.5 ure. Sicer pa mora kot metoda prepletati vse naslednje teme.

3.19. Razvoj in zapis algoritma za reševanje problema z računalnikom

Temo sestavljajo primeri računalniško usmerjenih algoritmov. Predstavljeni so splošni pojmi računalniških algoritmov, spremenljivka, vhod-izhod, kretnica, zanka, modul itd. Primerjalno so podani tudi načini zapisa algoritma v procesu razvoja algoritma: naravni jezik, diagram poteka, programski jezik.

Ocena teme je 2.75. Učitelji ugotavljajo, da so tako operacionalizacija ciljev kot vsebina in metoda zelo zahtevne, učenci imajo težave z razumevanjem, uporabo znanja in doseganjem višjih kognitivnih kategorij.

Vse te težave izvirajo predvsem iz dejstva, da je - kot je bilo že ugotovljeno v prejšnji temi - metoda reševanja problemov v naših šolah redkost in učenci niso vajeni svojega znanja logično strukturirati. Podobno velja seveda tudi za učitelje, ki so bili tudi učenci takih šol na vseh stopnjah.

3.20. Programski jezik

Učitelji lahko izbirajo med fortranom in pascalom. Obsežna tema zajema razvoj enega od jezikov, vlogo prevajalnika, programske ukaze, aritmetične izraze, vhodno izhodne stavke, krmilne stavke, polja spremenljivk in podprograme. Struktura je podobna za oba jezika.

Tema je dobila visoko oceno 3. Predvidenih 25 ur žele učitelji povečati na 29 ur. Učitelji ugotavljajo težave pri operacionalizaciji ciljev, tj. ne vedo natančno, kako v tej temi doseгati navedene cilje. Snov in metoda se jim zdita zelo zahtevni. Dokaj znanja terjajo tudi ostali trije - terminologija, učila in evaluacija.

Za učence pa ugotavljajo, da le stežka dosegajo višje kognitivne kategorije od stopnje uporabe znanja dalje. Celo več: še samo spoznavanje in razumevanje snovi jim delata težave. Te pa še povečajo nižji programski jeziki (na primer asemblerji) in manj udobni računalniški sistemi (na primer mikroračunalniki).
Težave predstavlja najprej aparaturna oprema, ki ni vselej na razpolago ali celo ni dosegljiva. Tem pa takoj sledijo problemi diferenciacije programov po posameznih usmeritvah, ki zavise zlasti od učiteljeve usposobljenosti in učnih pripomočkov.

Ugotoviti moramo, da je tema, ki predstavlja ne le preverjanje dosežkov celotnega pouka računalništva, temveč tudi sintezo pridobljenega znanja, bistvena, zato kaže o njej še nose-

bej razmišljati pri nadaljnjem delu in ji zagotoviti potrebno vsebino, ustrezno metodo in primerni čas.

3.21. Zgledi uporabe

Povsem razumljivo je, da je ta tema dobila največjo oceno po pomembnosti, tj. 4.5, saj gre za doseganje vseh vzgojnoizobraževalnih smotrov. Učitelji terjajo namesto predvidene 1 ure povečanje na 3 ure. Poudariti pa je treba, da gre pri tej enoti v bistvu le za zaključno utrjevanje snovi, saj so primeri uporabe sestavni del vsake učne enote. Pri tem je pomembna specifičnost posameznih usmeritev.

3.22. Mesto in vloga računalniškega znanja v poklicih

To naj bi bila informativna tema za poklicno usmerjanje učencev. Velika težava pri tem je pomanjkanje znanja, saj učitelj ne more biti strokovnjak za vsa področja. Tema je ocenjena z 1.75, torej manj, kot bi zaslužila. To je verjetno zato, ker jo je moč realizirati le z vabljenimi strokovnjaki z različnih področij dejavnosti, v katerih ima računalnik pomembno vlogo. Še najbolje bi bilo, če bi le-to učenci spoznavali kar neposredno v njihovi delovni organizaciji. Potreben čas je vsaj dve uri.

4. SINTEZA REZULTATOV

Iz zgornje analize vprašalnika, poročil učiteljev na šolah in razgovorov z učitelji računalništva sledijo zlasti naslednji zaključki:

4.1. Učni načrt

- Teme sorazmerno enakomerno pokrivajo učnovzgojne smotre. Izjemi sta le smotra uvodni seznanjanje z osnovami in zaključni smoter z vlogo računalniških znanj v poklicih.

- Glede na pomembnost za doseganje smotrov predmeta močno izstopa tema 21, sledijo pa ji teme 20, 12, 18 in 19. Glede na to kaže prestrukturirati vsebino predmeta tako, da bi omenjene teme z ustrezno sekvenco zgledov, primerov in vaj prestavljale hrbtenico učnega načrta, ki bi ga na ustreznih mestih dopolnili z znanji ostalih tem.

- Dodane smotre (F1, F2, F3) je moč zajeti v osnovnih smotrih, če bi jih jasneje opredelili.

- Časovno kaže okrepiti temo 21. Glede na težave s fondom ur in različno pomembnostjo tem je treba izdelati predlog minimalnega učnega načrta oz. ključnih znanj, ki naj jih dosežemo v okviru tega predmeta.

- Predmet ne zahteva od učencev preveč spominskega znanja. Razumevanje pa je sorazmerno zahtevno in predstavlja oviro za uporabo znanja in doseganje višjih spoznavnih kategorij, ki so vselej zasnovane na razumevanju. Skok od poznavanja do kategorije razumevanja je pri marsikateri temi višji kot od razumevanja do uporabe znanja.

- Predmet je usmerjen k reševanju problema z računalnikom, kar je velika obveza in spodbuda tako učencem kot učiteljem. Računalnik spodbuja, obenem pa podpira postopno doseganje višjih kategorij znanja. Zato je treba učencem in učiteljem pomagati z ustrezno opremo, učili in drugimi učnimi pripomočki.

4.2. Zasnova didaktičnega kompleta

- Sedanji učbenik (Bratko,I., Rajkovič,V. 1980) v glavnem pokriva navedene teme. Izjeme so le teme 3,4 in 22. Ta problem vsaj deloma rešuje učbenik "Osnove tehnike in proizvodnje" s poglavjem "Informatika in računalništvo" (Bratko,I., Rajkovič,V. 1980). Ob tem pa nas to poglavje razbremeni nekaterih uvodnih tem, tako da se bo novi učbenik za predmet računalništvo lažje posvetil že omenjenim ključnim temam za razvijanje sposobnosti reševanja problemov.

- Zbirka nalog (Benkovič,J.,Cokan,A., Martinec, M., Reinhardt,R., Roblek,B. 1980) je v pomoč tako učencem kot učiteljem, zlasti pri naporih za doseganje višjih kategorij znanja v poglavjih 18,19 in 20, ki izstopajo s svojo zahtevnostjo.

- Priročnik za učitelje (v pripravi) je glede na sorazmerno zahtevno snov, operacionalizacijo ciljev ter evaluacijo znanja nujno pripraviti v najkrajšem času. Pri pripravi je seveda treba upoštevati rezultate te analize.

- Komplet prosojnic (v pripravi). Večina učiteljev je pri obravnavi učil in učnih pripomočkov omenila potrebo po slikovnem gradivu v obliki kompleta prosojnic. Ustrezne prosojnice učitelji le težko pripravijo (npr. za temo 2), pomemben pa je tudi enoten komplet za teme 12, 13 in 21.

4.3. Zasnova modela izobraževanja učiteljev

- Zahtevnost ključnih tem (glede na smotre) za učence in učitelje priča, da smo s tečaji za učitelje lahko pokrili le najnujnejše potrebe (Roblek,B. 1972). Čimprej je treba pričeti izvajati ustrezni pedagoški modul v okviru visokošolskega študija, ki ga zaenkrat kot dodiplomski študij načrtujejo v sodelovanju FNT - matematika, Fakulteta za elektrotehniko in za metodološki sklop FNT - RCPU (Računalniški center za programirano učenje).

- Seminarji za učitelje (dvo do tridnevni), ki so že postali redna praksa, so ne le pomemben način dopolnilnega izobraževanja, ampak predstavljajo permanenten stik med učitelji in sodelavci projekta ter seminarjev. V bodočih temah seminarjev bo treba več pozornosti posvetiti metodičnim problemom v zvezi z reševanjem problemov in evaluacijo znanja, kajti dejstvo je, da tistega, česar učitelj sam ni bil deležen, ne more posredovati drugim.

4.4. Izdelava navodil za učitelje za izvajanje pouka računalništva

- Po mnenju učiteljev so sedanja navodila (Rajkovič,V. 1977) pripomogla k boljšemu doseganju zahtevnosti programa predvsem v okviru metod podajanja računalniških konceptov in reševanja problemov z računalnikom.

- Navodila je potrebno dopolniti glede na teme 4 in 9 ter v odviru tem 18, 19 in 20 izvleči programske in algoritmične koncepte ter jih povezati s sekvenco primerov in vaj.

4.5. Učni pripomočki

- Osnovni učni pripomoček je računalnik z visokim programskim jezikom (pascal, fortran). Zaželen je čimbolj neposreden stik z računalnikom, ki ga omogoča interaktivno delo.

- Pri poglavjih 12 in 13 nekateri učitelji uporabljajo tudi programiran kalkulator.

- Uporaba grafoskopa je pogosta, vendar je vezana na ustrezno pripravljene prosojnice, s čimer imajo učitelji vsaj pri nekaterih poglavjih (2,10,12,13,14) precej težav. To narekuje čimprejšnjo pripravo kompleta prosojnic.

5. ZAKLJUČEK

Komparativna analiza realizacije vzgojnoizobraževalnega programa računalništva v srednjih šolah je pokazala zlasti:

1. Težave so v zvezi z operacionalizacijo ciljev. Te ne izvirajo le iz vrzeli v metodah in premajhnega števila primerov v okviru posameznih učnih enot, temveč zlasti iz dejstva, da so učitelji vajeni razmišljati predvsem o vsebinskih ciljih pouka in bistveno manj o procesnih. Tako stanje pogojuje njihovo lastno izobraževanje, saj je hiba naše celotne vzgojnoizobraževalne prakse, da terja predvsem poznavanje podatkov o dejstvih in pojmih, bistveno manj pa njihovo globje razumevanje in sposobnost uporabe. Prav tako vse prepogosto napišemo cilje za začetek učnovzgojnega programa, ker to pač terjajo navodila, pri obravnavi tem pa jih uspešno ignoriramo.

   Rešitev bi bila zahteva, da celotni sistem evaluacije pridobljenega znanja postavimo na preverjanje doseganja deklariranih smotrov. V tem smislu bi bilo treba evaluirati ne le učenčeve dosežke, temveč tudi delo in pogoje za delo učitelja in šole ter ne nazadnje sam vzgojnoizobraževalni program.

2. Dokaj visoka stopnja abstraktnosti v učni snovi predmeta računalništvo je seveda zahteva, ki jo težko premoščamo, saj je ugotovljeno, da je prehod iz konkretne v abstraktno sfero težak. Zato je treba sposobnost abstrakcije načrtno gojiti v sodelovanju z vsemi drugimi predmeti.

3. Metoda reševanja problemov kot ključna metoda s pomenom za znanost in prakso je še pravi pastorek v vzgojnoizobraževalni praksi. Vzrok za to pa ni le v nevajenosti učiteljev v uporabi te metode, temveč tudi v pomanjkanju vrste dobrih primerov reševanja realnih problemov ob istočasni ohranitvi potrebne sistematike in logične hrbtenice predmeta. Zato bo treba terjati, da postane metoda reševanja problemov neogibni pogoj gradnje učnih enot.

4. Za interdisciplinarno povezovanje bi kazalo iskati primere, ki bi pomagali učiteljem računalništva, da svoje delo povezujejo z delom učiteljev drugih strok na šoli.

5. Povezava s prakso je nujna ne le pri zagotavljanju aparaturne opreme, temveč tudi za usmerjanje. Več bo treba storiti na šolah, da bi bila ta povezava tesnejša. Še zlasti pa je treba učencem in učiteljem odpreti vrata delovnih organizacij in jim nuditi proste kapacitete ter strokovno pomoč, sicer šole svojega programa ne bodo mogle uresničiti, kar se bo otepalo praksi. Že sedaj je nekaj svetlih zgledov takega sodelovanja, ki jih je treba poudariti kot primer za druge.

6. Projekt pouka računalništva v usmerjenem izobraževanju mora stalno dopolnjevati učbenik, graditi priročnik za učitelje, pripraviti komplet prosojnic, teste, razmišljati o drugih avdiovizualnih pripomočkih in učilih. Na učitelje lahko prenesemo odgovornost za individualizacijo pouka, ne pa tudi za celotno gradnjo programa. Ta mora biti skupinsko delo najboljših strokovnjakov, ki ga ocenijo ustrezni strokovni in družbenopolitični organi, da bi bilo zatem lahko osnova delu na šolah.

7. Sistem izobraževanja učiteljev nasploh in posebej učiteljev računalništva mora biti tesno vezan na naloge njihovega dela. Vsa praksa kaže, da "česar se učitelj ni učil, tudi sam ne zna učiti drugih". Če je bil vzgajan k poslušanju predavanj, bo tudi sam predaval. Če ni srečal metode reševanja problemov in povezovanja teorije s prakso v svojem študiju, tega tudi v šoli ne bo delal.

6. LITERATURA

Benkovič,J., Korbar,R., Rajkovič,V., Roblek, B., Sirnik,I., Kritična ocena stanja računalništva na srednjih šolah v SR Sloveniji, Informatica 77, 7 203.

Benkovič,J., Bratko,I., Kornhauser,A., Rajkovič,V., Roblek,B., Sirnik,I., Introducing fundamentals of cybernetics with automatic control at secondary level, IFAC Symposium Trends in Automatic Control Education, Barcelona,1977.

Benkovič,J., Cokan,A., Martinec,M., Reinhardt, R., Roblek,B., Računalništvo: Zbirka nalog I, Državna založba Slovenije, Ljubljana, 1980.

Bloom,B.S., Taksonomija ili klasifikacija obrazovnih i odgojnih ciljeva, Jugoslovenski zavod za proučavanje školskih i prosvetnih pitanja, Beograd, 1970.

Bratko,I., Rajkovič,V., Roblek,B., Some aspects of secondary school computer education, Modern trends in cybernetics and systems, vol.3, Springer-Verlag, Berlin, 1976, 35-40.

Bratko,I., Rajkovič,V., Uvod v računalništvo, Državna založba Slovenije, Ljubljana,1980(ponatis).

Bratko,I., Rajkovič,V., Informatika in računalništvo, poglavje v učbeniku Osnove tehnike in proizvodnje, Tehniška založba Slovenije, Ljubljana, 1980.

Gagne,R.M., The conditions of learning, 2nd edition, Holt, Rinehart&Winston, New York,1970.

Roblek,B.(urednik), Računalništvo: Gradivo s tečaja za učitelje, Zavod SRS za šolstvo, Ljubljana, 1972.

Rajkovič,V., Metodični problemi pri pouku računalništva, gradivo za učitelje računalništva, 1977.

Rajkovič,V., On the role of computer science subjects at the secondary school level, 3rd International Conference on Information Technology, Jerusalem, 1978.

Rajkovič,V., Vrednotenje računalniške opreme za potrebe reformirane srednje šole, simpozij Informatica 78, Bled, 7 107.

Reihardt,R.,Rajkovič,V., Lajovic,I., Vrečko, J., Kriteriji za izbiro programskega jezika za pouk računalništva v srednji šoli, simpozij Informatica 77, Bled, 7 107.

Reihardt,R., Republiška tekmovanja srednješolcev iz računalništva, Informatica, 2(1), 1978, 59-83.

Benkovič,J., Kornhauser,A., Rajkovič,V., Poročilo o delu, raziskovalni projekt Pouk računalništva v usmerjenem izobraževanju, 1.faza 1978, 2. faza 1979, 3. faza 1980.

Kornhauser,A., Uporaba računalnika v kemijskem izobraževanju, Vzgoja in izobraževanje, 6(5), 1975, 3-24.

GENERISANJE IMENIČKIH OBLIKA U SRPSKOHRVATSKOM JEZIKU

D. VITAS

MATEMATIČKI INSTITUT, KNEZ MIHAILOVA 35, BEOGRAD

UDK: 681.3.06 : 808.61

U radu je opisan postupak automatske morfonološke sinteze imeničkih oblika u savremenom srpskohrvatskom jeziku. Postupak je zasnovan na formalizaciji pravila alternacija i identifikaciji segmenta u reči na koji, eventualno, deluju alternacije. Opisani su potrebni ulazni podaci, kao i njihov odnos sa ugradjenim "znanjem o oblicima". Istaknute su karakteristike programske realizacije, kao i pravci daljih istraživanja.

GENERATION OF NOUN´S FORMS IN SERBOCROATIAN. In this paper, the procedure of automatic morphonological synthesis of noun´s forms in contemporary serbocroatian is described. The procedure is based on formalization of alternation rules and identification of word´s segments which may be liable to alternation. Needed input data and their rapport with implemented "knowledge of forms" are described. Characteristics of program realization and the directions for further investigations are outlined.

UVOD

1. Proces generisanja iskaza u nekom prirodnom jeziku, u okviru date generativno-transformacione gramatike i nad zadatim rečnikom, se sastoji u osnovi iz dve faze (Vauquois/75/):
-generisanja površinske sintaksičke strukture, i
-morfološkog generisanja oblika reči.
U prvoj fazi se drvo izvodjenja dubinske strukture, koja je, u izvesnom smislu, kanonična reprezentacija značenja, primenom transformacionih pravila prevodi u drvo izvodjenja površinske strukture. Završni simboli (terminali) drveta izvodjenja su elementi rečnika, koji se još nisu realizovali. Pravila koja odredjuju morfološku realizaciju se proučavaju u okviru morfonologije i morfosintakse. Pri tome, prema (Dubois/73/), morfonologija je opis svih operacija pomoću kojih niz završnih simbola površinske strukture dobija fonološku i fonetsku interpretaciju, kako bi postao realizovani iskaz. Morfosintaksa je opis pravila kombinovanja morfema u formiranju reči, kao i fleksivnih afiksa.
2. Da bismo u daljem izlaganju izbegli dvosmislenosti, uvodimo sledeća terminološka razlikovanja:

Element leksike ćemo nazivati leksem. Realizaciju leksema u iskazu ćemo nazivati oblik. Bilo kakav niz karaktera (grafičkih simbola) izmedju dva blanka, koji je realizovan u nekom tekstu, zvaćemo reč.

Za kvantitativnu lingvistiku, posebno za leksičku statistiku, je karakteristično ispitivanje odnosa reč/oblik, pri čemu je broj reči (N) odredjen dužinom teksta a broj oblika (V) dužinom rečnika tog teksta
3. Poslednjih godina je bilo pokušaja automatskog generisanja iskaza srpskohrvatskoga jezika (dalje, SH-jezika). Ovakvi eksperimenti su bili zasnovani na parcijalnom gramatičkom opisu površinske strukture iskaza a kao rečnik je upotrebljavan rečnik oblika leksema, podesno označen za potrebe ovakvog pristupa generatornom procesu. Ovakav pristup je razumljiv kada se zna da ne postoji teoretsko-lingvistički model savremenog SH-jezika u obliku podesnom za automatsku obradu. Problemi automatskog morfonološkog generisanja odn. problemi morfonološke sinteze oblika, do

sada, takodje, nisu obradjivani.
4. Cilj ovoga rada je da se opiše postupak automatskog morfonološkog generisanja imeničkih oblika u savremenom SH-jeziku, kao i elementi programske implementacije ovog postupka.

U razradi podesnog formalizma za opis morfonoloških relacija, korišćene su različite gramatike, kao i Pravopis SH-jezika iz 1954. godine.

Imenički oblici se, u sistemu koji ćemo opisati, generišu polazeći od nominativa jednine (skraćeno, NJ), odn. nominativa množine (NM) ako jednina ne postoji, umesto iz osnove, kako je uobičajeno. Opredeljivanje za NJ je bilo odredjeno praktičnim razlozima: Za datu imenicu, NJ, ako postoji, je jedinstven, za razliku od osnove koja to ne mora biti. Takodje, nominativom je uobičajeno prikazivati imenice u rečniku, te nam ovakav pristup obezbedjuje eventualnu upotrebu postojećih rečnika u daljim istraživanjima.
5. Primer. Problem morfonološkog generisanja, kao i uvedene pojmove, prikazaćemo na sledećem iskazu - teoremi Euklidske geometrije: U svakom uglu postoji n-1 poluprava, pomoću kojih je taj ugao podeljen na n jednakih uglova.
Ovaj iskaz sadrži 16 reči. Leksem (ugao) se realizovao tri puta, u oblicima uglu, ugao, uglova. Zadatak morfonolške sinteze se sastoji u opisivanju skupa transformacija koje leksem (ugao) prevode u navedene oblike.
Da se ovaj problem ne sastoji u dopisivanju nastavaka na osnovu može se videti iz sledećih primera: Kada se na osnovu (vojnik) dopiše nastavak -e, onda se u vokativu jednine realizuje oblik "vojniče" a u akuzativu množine oblik "vojnike". Dativ jednine za leksem (fuga) je "fugi" a za leksem (tuga) "tuzi", tj. u istom padežu, nastavak -i proizvodi različite osnove.
6. Program za generisanje imeničkih oblika je realizovan u RC Matematičkog instituta, na mašini IBM 360/44. Program je pisan na programskom jeziku FORTRAN IV. Nepogodnost primene FORTRAN-a na istraživanja u ovoj oblasti su delimično otklonjene upotrebom jednog broja rutina opisanih u (Vitas/79/), namenjenih pojednostavljivanju manipulacije karakterima. Zadržan je i način kodiranja nekih slova SH-alfabeta (npr, š je SX, ć je CX, č je CY, ž je ZX, itd.). Program zahteva oko 60 KB.

MEHANIZAM ALTERNACIJA

1. U sistemu za generisanje imeničkih oblika, oblici imenica se, kao što je ranije naznačeno, generišu iz NJ, pri čemu ćemo NJ smatrati reprezentantom odgovarajućeg leksema. Realizacija postupka generisanja podrazumeva opis pravila transformacije leksema (predstavljenog nominativom jednine) u oblike, mehanizam primene tih pravila, kao i odredjivanje one podreči u datoj reči nad kojom ova pravila treba primeniti. Pomenuta pravila transformacije leksema se, obično, nazivaju pravilima alternacija, ili, u starijoj literaturi, glasovnim zakonima. Pri tome, pojam alternacije obuhvata i pravila koja se primenjuju pri gradjenju reči, no ova pravila nisu od značaja u procesu generisanja oblika.

2. Oblik pravila alternacija, u opštem slučaju, je sledeći:
Ako u datoj reči postoji podniz $\bar{x}$ sa svojstvom S i zadovoljen je uslov R, onda $\bar{x}$ zameniti podnizom $\bar{y}$.

Nizovi $\bar{x}$ i $\bar{y}$ mogu biti prazni ali ne istovremeno. Prazan niz obeležavamo sa $\emptyset$.
Svojstvo S je skup akustičkih i artikulacijskih osobina glasova, takvih da jednoznačno odredjuju elemente niza $\bar{x}$. Npr. svojstvo "biti velarni konsonant" odredjuje glasove k, g, h, pa je niz od jednog elementa sa ovim svojstvom ili -k, ili -g, ili -h. U odnosu na uslov R, razlikujemo dva tipa alternacija. Alternacije kod kojih je uslov R uvek zadovoljen, nazivaju se fonološki uslovljene alternacije. Alternacije kod kojih uslov R zavisi od morfonoloških osobina oblika koji se generiše, se nazivaju morfološki uslovljene alternacije. Ispunjenost uslova R kod morfološki uslovljenih alternacija moguće je u nekim slučajevima utvrditi automatski (npr. palatalizacija kod imenica muškog roda ili jotovanje), dok u drugim slučajevima to, za sada, nije moguće (npr. prisustvo nepostojanog a ili prelazak o u l u NJ), te je tada imenici neophodno pridružiti informacije koje ukazuju na prisustvo alternacija.
Elementi niza $\bar{y}$ su jednoznačno odredjeni elementima niza $\bar{x}$, ako su ispunjeni uslovi S i R. Npr. u slučaju palatalizacije, za $\bar{x}$=k, $\bar{y}$=č.
Spisak alternacija ugradjenih u sistem dat je u Prilogu 1.

3. Pitanje redosleda primene alternacija nije razmatrano u okviru SH-morfonologije. U sistem su ugradjene alternacije sa prioritetom definisanim redosledom navodjenja alternacija u Prilogu 1, što je dalo zadovoljavajuće rezultate. Mogućnost eksperimentisanja sa drukčijim prioritetima je, takodje, ugradjena u sistem.

4. Sledeći problem koji se javlja u postupku generisanja oblika se sastoji u ograničavanju dejstva alternacija na podreč zadate reči. U lingvističkoj teoriji je uobičajeno shvatanje da se

oblici promenljivih reči mogu rastaviti na osnovu i nastavak, pri čemu je osnova nosilac leksičkog značenja a nastavak nosilac gramatičkog značenja reči. Osnova date reči, u načelu, nije jedinstvena - jedan leksem može imati više različitih osnova (koje nazivamo alomorfima). Nastavak se formira od karakterističnog fleksivnog nastavka i, eventualno, od infiksa. Infiksi, opisani u Prilogu 3, pripadaju osnovi, ali se u ovom radu, posmatraju kao zasebni morfemi. Pri tome, jasno je da ako su za dati leksem poznate sve njegove osnove, infiksi i nastavci, onda se generisanje odredjenog oblika svodi na korektno dopisivanje odgovarajućih morfema.
Ukoliko je, pak, poznata samo jedna od osnova, ili jedan unapred utvrdjeni oblik, onda je neophodno mehanizmom alternacija transformisati tu osnovu u njene alomorfe, da bi,u sledećem koraku, bili korektno generisani oblici. Alomorfi osnove, a time i generisani oblici, mogu biti veoma udaljeni od polazne osnove, odn. oblika,s obzirom na raspored glasova. Npr. kod imenice (otac), oblicima NJ/otac/, DJ/ocu/,VJ/oče/ je zajednički samo prvi glas.
S druge strane sve osnove jedne reči sadrže jedan zajednički podniz, koji ostaje neizmenjen u toku generisanja oblika. Ovaj podniz smo nazvali osnovni segment. Osnovni segment je uvek sadržan u osnovi, kao pravi podniz. Osnovni segment je jedinstven za datu imenicu, što bi moglo biti od značaja pri konstruisanju rečnika osnova u raznim računarskim primenama /automatsko prevodjenje, pronalaženje dokumenata, itd./. Dopunu osnovnog segmenta date reči do bilo kojeg njenog oblika ili osnove nazvali smo sufiks-segment, ili kraće s-segment. Pod medjuoblikom ćemo podrazumevati bilo koju reč nastalu dopisivanjem sufiks-segmenta na osnovni segment. s-segment nekog oblika, osnove ili medjuoblika je onaj deo reči u kome eventualno deluju alternacije.

5. Dekompozicija oblika na osnovni i s-segment je, sa stanovišta automatske obrade, podesnija od uobičajenog rastavljanja na osnovu i nastavak, jer se polazeći od unapred utvrdjenog oblika deo u s-segmentu na koji eventualno deluju alternacije može automatski odrediti.
Ako sa v označimo proizvoljan vokal (uključujući i r, kada se pojavljuje kao vokal) a sa c bilo koji konsonant, onda podniz u s-segmentu koji je izložen dejstvu alternacija, može imati jedan od dva sledeća oblika
/a/ vc^nv $(0 \leq n \leq 4)$ /b/ vc^n $(1 \leq n \leq 3)$
(Granice za n su postavljene prema statističkim istraživanjima o slogovnoj strukturi SH-jezika (Kokovič /79/)).

6. Programski je mehanizam alternacija realizovan rutinom ALTER, kojom se vrši morfonološka transformacija s-segmenata. Ključni deo ove rutine je rutina IDENT, koja u s-segmentu identifikuje podnizove sa svojstvom S alternacija. Realizacija alternacija tipičnih za NJ (alternacije 1, 3, 4, 5) se vrši rutinom OULA. Alternacija 2 (Ø→a) je realizovana u okviru rutine NEPA, zbog problema identifikacije prazne reči u s-segmentu. Prepoznavanje s-segmenata se vrši rutinom SUFIX, tako što se u reči identifikuje jedna od struktura /a/, /b/ s-segmenta.

7. Primer. Kao ilustraciju opisanog postupka, navodimo tok generisanja oblika nominativa množine imenice (podatak), pretpostavljajući da su potrebna morfološka obeležja, kao i nastavak za NM: - i, odredjeni. U primeru je prikazano odredjivanje sufiks-segmenata, sufiks-segmenti izmenjeni dejstvom alternacija i medjuoblici, nastali dopisivanjem izmenjenih sufiks-segmenata na osnovni segment. Brojevi alternacija se odnose na spisak alternacija dat u Prilogu 1.

Medjuoblik	s-segment	alternacija ili nastavak	izmenjeni s-segment
NJ:podatak	ak	1.	k
podatk	atk	+i	atki
podatki	atki	6.	atci
podatci	atci	11.	aci
NM:podaci			

STRUKTURA PODATAKA

1. Pre nego što se pristupi generisanju imeničkih oblika na gore opisani način, potrebno je odrediti morfonološke osobine traženog oblika. Osobine koje treba utvrditi odnose se na ispitivanje egzistencije traženog oblika, na odredjivanje odgovarajućeg infiksa i padežnog nastavka, na definisanje skupa alternacija kojim se NJ prevodi u traženi oblik, kao i na ispitivanje izvesnih svojstava, koja nisu u neposrednoj vezi sa algoritmom generisanja ali su neophodne u radu na tipologiji s-segmenata.

2. Razlikujemo dve grupe morfonoloških osobina. Prvu grupu čine one osobine koje se moraju pridružiti imenici kao ulazni podatak. U realizovanom sistemu, ovu grupu čine tri podgrupe podataka:
-niz MORF morfoloških osobina date imenice.Elementi niza MORF su opisani u Prilogu 2.
-niz MFMARK takav da je MFMARK(i)=tačno, ako se u postupku generisanja primenjuje alternacija i; inače je= netačno.
-oblik koji treba generisati i koji se zadaje brojem (jednina,množina) i padežom. Npr.sa (2,1)

je označen NM, sa (1,5) vokativ jednine.(0,0) označava da treba generisati celu paradigmu.

3. Drugoj grupi podataka pripadaju one osobine traženog oblika čije je prisustvo moguće automatski utvrditi. Ove osobine ćemo pomenuti navodjenjem rutina koje ih ispituju.
Rutina PADEZ ispituje uslove egzistencije oblika (5.-8. elementi niza MORF) i ako oblik postoji, odredjuje indeks padežnog nastavka u listi nastavaka (Prilog 4). Rutinom INFIX se ispituje da li traženi oblik dopušta infiks i ako je tako, odredjuje se indeks infiksa u listi infiksa (Prilog 3). PSJ ispituje svojstva završnog konsonanta u s-segmentu i generiše uslove palatalizacije, sibilarizacije, jotovanja i inverznih alternacija za neke vrste imenica. Rutina KONGRP ispituje da li u s-segmentu postoji konsonantska grupa i, ako postoji, odredjuje njene osobine a posebno, osetljivost na alternaciju 2. DPADEZ vrši korekciju nastavaka pod dejstvom podataka generisanih rutinama PSJ i KONGRP.

4. Tokom rada opisanih rutina, konstruiše se niz MARK-1, čijih je prvih 16 elemenata istovetno sa nizom MARK. Ovaj niz sadrži sve informacije potrebne za korektno generisanje bilo kog oblika. Generisani indikatori svojstava u nizu MARK-1 su posledica "znanja o oblicima", ugradjenih u sistem posredstvom navedenih rutina. (Niz MARK-1 je prikazan u primerima na kraju rada.)

5. Istraživanje odnosa "znanja" i ulaznih podataka se može predstaviti kao problem minimalizacije skupa ulaznih podataka. Naime, nizovi MORF i MFMARK nisu minimalni u smislu da se neka od, u njima prikazanih, svojstava mogu generisati. Takav je slučaj npr sa nepostojanim a u GM (MFMARK(2)) ili egzistencijom infiksa kod imenica srednjeg roda (MORF(12)). U takvim slučajevima, odgovarajući ulazni podaci imaju samo kontrolnu ulogu. S druge strane, neki ulazni podaci, npr. semantički marker "živo-neživo" (MORF(4)) se, za sada, ne mogu generisati. Problem odredjivanja minimalnih ulaznih podataka će biti moguće rešiti, po našem mišljenju, tek kada se izvrši tipološka analiza s-segmenata, kao i statistička analiza nizova tipa MARK-1, pridruženih ovim s-segmentima.
Nizovi ulaznih podataka MORF i MFMARK predstavljaju definiciju morfonološkog ponašanja imenica kojoj su pridruženi s obzirom da je za zadatu imenicu i nizove MORF i MFMARK jednoznačno odredjen bilo koji njen oblik.

6. Opisani programski sistem je iscrpno testiran na oko dve stotine imenica. Izbor podataka za testiranje je izvršen prema karakterističnim primerima, navedenim u različitim gramatikama SH-jezika. Za ovako izabran skup test-podataka, testiranje je obavljeno uspešno u smislu da generisane paradigme u potpunosti odgovaraju svom gramatičkom opisu. Na kraju ovog rada su navedeni rezultati generisanja oblika nekih "komplikovanijih" imenica. U prvom pravougaoniku su uokvireni ulazni podaci a u drugom generisani oblik.

Izvestan broj imenica, ipak, nije bio obuhvaćen testiranjem, i to iz sledećih razloga:
- Neke komponente sistema, kao npr. rutina za ispitivanje disimilacije ili rutina za adaptaciju atipčnih sufiks-segmenata imenica stranog porekla na morfonološki sistem SH-jezika, još nisu realizovane.
- Kod nekih imenica se javljaju odstupanja od ugradjenih morfonoloških pravila. Takve su npr. (mozak) čiji je GJ=mozga, umesto moska (nepostojano a + jednačenje po zvučnosti), (doba) -jedina imenica srednjeg roda na -a, itd. Moguće je, naravno, sistem tako dopuniti da obuhvati i ove izuzetke, ali smatramo da takav postupak nema smisla kada se ima u vidu sporadičnost ovakvih pojava. Stoga će ovakve imenice biti predstavljene u posebnom rečniku izuzetaka, navodjenjem osobenosti njihove promene.
- Neke reči, koje se po svojoj funkciji ponašaju kao imenice, u morfonološkom smislu pripadaju pridevima te ih treba generisati po pridevskoj promeni. Takve su npr. reči Bačka, Turska, draga, itd.

DALJA ISTRAŽIVANJA

1. Rezultati opisanog istraživanja ohrabruju na proširivanje eksperimenta. Osim već pomenutih radova na tipologiji sufiks-segmenata, neposredno predstoji proširivanje sistema na druge vrste promenljivih reči. Smatramo da je promenu pridева, kao i drugih promenljivih reči, izuzimajući glagole, moguće ugraditi, uz neznatne dopune u opisani sistem, dok će za ugradnju glagola biti potrebno dodati sistemu neke nove mehanizme morfonološke analize. Bilo bi od značaja ispitati i mogućnost primene opisanog postupka na druge fleksivne jezike a posebno na južnoslovenske jezike.

2. Kao posebno zanimljiv problem se može postaviti pitanje upotrebljivosti klasičnih rečnika SH-jezika u automatskoj morfonološkoj analizi i sintezi. Ovaj problem se može formulisati na sledeći način: Da li je, na osnovu uobičajenog načina opisivanja leksema u rečniku, moguće automatski konstruisati nizove MORF i MFMARK, za potrebe sistema za generisanje oblika? Pozitivno rešenje bi omogućilo da se polazeći od postoje-

ćih rečnika konstruišu razne vrste rečnika, neophodne u daljim istraživanjima i primenama u oblasti računarske lingvistike, čime bi bile ostvarene značajne materijalne i vremenske uštede.

3. Povezivanjem sistema za generisanje oblika sa nekim sistemom za izradu konkordanci (npr.Vitas /79/), bi bila ostvarena mogućnost automatskog pronalaženja svih pojavljivanja oblika jednog ili više leksema u nekom skupu tekstova. Kako se uz svako pojavljivanje nekog oblika nalaze reference na tekstove ili delove tekstova u kojima se taj oblik pojavio, bila bi dakle omogućena ekstrakcija svih tekstova koji sadrže zadate lekseme. Ovakvom tehnikom bi se, dakle, mogao realizovati sistem za komunikaciju sa bazom tekstova na SH-jeziku, upotrebom prirodnih ključeva -reprezentantima leksema.

4. Pomenuti problemi, će biti rešavani u okviru projekta "Matematička i računarska lingvistika", koji bi trebalo da se počev od sledeće godine realizuje u Matematičkom institutu.

BIBLIOGRAFIJA

1. Dubois J. et al. /75/: Dictionnaire de linguistique, Larousse, Paris, 1973.
2.Ković M. /79/: Slogovna struktura srpskohrvatskog jezika, Izveštaj sa Seminara za matematičku i računarsku lingvistiku Matematičkog instituta, Beograd, decembar 1979.
3. Vauquois B. /75/: La traduction automatique à Grenoble, Documents de linguistique quantitative n° 24, Dunod (Ass.Jean-Favard),Paris, 1975.
4. Vitas D./79/: Prikaz jednog sistema za automatsku analizu teksta, INFORMATICA 79, Bled, oktobar 1979, 7 101.


```
PODATAK     1 7
MORF  = 1 1 1 0 1 1 1 0 0 0 0 0 0 1 1 1
MFMARK = T T F F F T T F T T T F
BROJ= 2 PADEZ= 1
NASTAVAK=    2   1   1   6
IZMENJENO K=PODATK
IZMENJENO K=PODATK
MARK-1 1110111C0000011100000000000C0000000000000
       0000000C0000000000000001C001101111000000000
PSJ-MARKERI  1 0 0 1 0 0 0 0 1 0
MARK-1 1110111C0000011100000000000C0000000000001
       0010000001001000000001000010111000000000
IZMENJENO K=PODATCI
IZMENJENO K=PODACI
IZMENJENO K=PODACI
```

Primer 1.NM(podatak)=podaci

```
TURCYIN     1 7
MORF  = 1 1 1 1 1 1 1 0 1 0 0 0 0 1 1 1
MFMARK = F T F F F T T T T T T F
BROJ= 2 PADEZ= 2
NASTAVAK=    2   2   1   2
MARK-1 1111111C1000011100000000C0CCC0C000000C000C0
       0000000C0000000000000001000111111000C000000
PSJ-MARKERI  1 0 0 0 0 0 0 0 0 0
MARK-1 1111111C10C0011100000000C0C000C000000000C01
       00C000000C0000000000001000000111000000000
PSJ-MARKERI  1 0 0 1 0 0 0 0 1 0
MARK-1 1111111C0000011100000000000C0000000000001
       0010000001001000000001000010111000000000
IZMENJENO K=TURAKA
```

Primer 2. GM(Turčin)=Turaka

```
MISAO       1 5
MORF  = 1 3 2 0 1 1 1 0 0 0 0 0 0 1 1 1
MFMARK = T T T F F T T T T T F T F
BROJ= 1 PADEZ= 7
NASTAVAK=    1   7   3   6
IZMENJENO K=MISC
IZMENJENO K=MISL
U REZIMU  1 ZA ALTERNACIJU BROJ  2 MFMARK=F
IZMENJENO K=MISL
MARK-1 1320111C0000011100000000C000C000000000000
       0000000000000000000001C0011111010000000C00
PSJ-MARKERI  1 0 0 0 0 0 0 0 0 1
MARK-1 1320111CC0C0011100000C0C000C000000000C01
       0000000C01C0000000000C000011010000000000
IZMENJENO K=MISLX
IZMENJENO K=MISXLX
IZMENJENO K=MISXLX
IZMENJENO K=MISXLXU
```

Primer 3.IJ(misao)=mišlju

```
DRVC        1 4
MORF  = 1 1 3 1 1 1 1 0 0 0 0 1 4 1 1 1
MFMARK = F F F F F F F F F F F F
BROJ= 1 PADEZ= 2
NASTAVAK=    1   2   1   2
MARK-1 1131111C000141110000000000C0C000C000000
       0000000CC000C000000000000000000000000000C00
PSJ-MARKERI  1 0 0 0 0 0 3 0 0 0
MARK-1 1131111C000141110000000000C0C000000000001
       0000030CC0C0C000000000C000CCC000000000000
INFIX=    4  22   7
IZMENJENO K=CRVE
IZMENJENO K=CRVET
IZMENJENO K=CRVETA
```

Primer 4(a).GJ(drvo)=drveta - "živo"

```
PISAC       1 5
MORF  = 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1
MFMARK = T T F F F T T F T T T F
BROJ= 1 PADEZ= 5
NASTAVAK=    1    5    1    3
IZMENJENO K=PISC
IZMENJENO K=PISC
MARK-1 1111111000000111000000C000000000000C000000
       000000000000000000000001C001101110000000000
PSJ-MARKERI   1 0 1 0 0 0 0 1 0 0
MARK-1 1111111000000111000000C0C000000000000CC1
       010000000100100000000001C001001110000000C00
IZMENJENO K=PISCYE
IZMENJENO K=PISXCYE
IZMENJENO K=PISXCYE
```

Primer 6.VJ(pisac)=pišče

```
UGAO        1 4
MORF  = 1 1 1 0 1 1 1 0 0 1 1 0 0 1 1 1
MFMARK = T T T F F F F F T T T F
BROJ= 2 PADEZ= 2
NASTAVAK=    2    2    1    2
IZMENJENO K=UGC
IZMENJENO K=UGL
IZMENJENO K=UGL
MARK-1 1110111001100111000000000000000000C000000
       000000000000000000000001C000001110000000000
PSJ-MARKERI   1 0 0 0 0 0 0 0 0 0
MARK-1 1110111001100111000000000000000000000CC1
       000000000100100000000001C000000111000000000
INFIX=    1    3   18
IZMENJENO K=UGLO
IZMENJENO K=UGLOV
IZMENJENO K=UGLOVA
```

Primer 7.GM(ugao)=uglova

Prilog 1. Ugradjene alternacije

(Alternacije su navedene ili prikazivanjem nizova $\bar{x}$ i $\bar{y}$ u obliku $\bar{x}\rightarrow\bar{y}$, ili svojim uobičajenim gramatičkim nazivom).

Ulazne alternacije:

1. a→Ø
2. Ø→a
3. o→l
4. Ø→l
5. l→O
6. palatalizacija
7. sibilarizacija
8. jotovanje
9. jednačenje po mestu tvorbe
10. jednačenje po zvučnosti
11. gubljenje t,d u konsonantskoj grupi
12. s→š

Ugradjene alternacije:

13. Gubljenje udvostručenog konsonanta
14. Inverzna palatalizacija
15. Inverzna sibilarizacija
16. Inverzna jednačenja

Prilog 2. Niz MORF

1. Vrsta reči (imenica,...)
2. Vrsta promene (A,E,I - prema nastavku za GJ)
3. Rod imenice (muški, ženski, srednji)
4. Semantički marker "živo-neživo"

```
DRVO        1 4
MORF  = 1 1 3 0 1 1 1 0 0 0 0 0 0 1 1 1
MFMARK = F F F F F F F F F F F F
BROJ= 1 PADEZ= 2
NASTAVAK=    1    2    1    2
MARK-1 1130111C000001110000000000C000000000000
       00C000000000000000000000000000000000C0C00
PSJ-MARKERI   1 0 0 0 0 0 3 0 0 0
MARK-1 1130111C000001110000000C0C000000000000CC1
       000003000000000000000000C000000000000C00000
IZMENJENO K=DRVA
```

Primer 4(b).GJ(drvo)=drva - "neživo"

```
SLIKA       1 5
MORF  = 1 2 2 0 1 1 1 0 0 0 0 0 0 1 1 1
MFMARK = F F F F F F T T T T T F
BROJ= 1 PADEZ= 3
NASTAVAK=    1    3    2    6
MARK-1 1220111000000111000000C000000000000C000000
       000000000000C000000000C001111110000C00000
PSJ-MARKERI   1 0 0 1 0 0 1 0 1 0
MARK-1 1220111000000111000000000C0C000000000CC1
       001001000C000000000000000C0101110000000C0
IZMENJENO K=SLICI
IZMENJENO K=SLICI
```

Primer 5.DJ(slika)=slici

Markeri egzistencije

5. Postoji jednina?
6. Postoji množina?
7. Rod jednine = rodu množine?
8. Ako nije 7, kako se menja množina?

Opis infiksa

9. Produžena jednina?
10. Produžena množina?
11. Broj različitih produžavanja
12. Nejednakosložnost imenica?
13. Tip infiksa za 12.

Atipični nastavci

14. atipičan nastavak za VJ
15. atipičan nastavak za IJ
16. atipičan nastavak za GM

Prilog 3.
Lista infiksa

1. -ov-
2. -ev-
3. -in-
4. -et-
5. -es-
6. -t-
7. -v-
8. -r-
9. -n-

Prilog 4.
Lista nastavaka padežnih

1. -Ø
2. -a
3. -e
4. -u
5. -O
6. -i
7. -$\bar{a}$
8. -om
9. -em
10. -ju
11. -ama
12. -ima
13. -iju

NOVOSTI KOJE MIKRORAČUNALA UNOSE U UPRAVLJANJE I NADZOR PROCESA

GABRO SMILJANIĆ

ELEKTROTEHNIČKI FAKULTET, ZAGREB

UDK: 681.3.019

Mikroračunala unose više novosti u upravljanje i praćenje rada procesa, koje ranije nisu postojale. U članku se neke od tih novosti razmatraju i iznose primjeri.

NEW POSSIBILITIES INTRODUCED BY MICROPROCESSORS IN PROCESS CONTROL AND SUPERVISION: The microprocessors introduce several new possibilities in process control and supervision not existing earlier. In the article some of these new possibilities are considered and examples given.

O novim mogućnostima koje se otvaraju na području upravljanja procesima, pojavom mikroračunala onakvih kakva se danas mogu kupiti na tržištu, bilo je već govora u ranijim radovima.[1,2,3,4,] U ovom članku pokušava se neke od tih mogućnosti detaljnije istražiti i prikazati njihovu primjenu u praksi. Dakako sve te primjene nije bilo moguće do kraja razraditi-to će trajati koliko i mikroračunala - ali neke od njih su realizirane i nalaze se u primjeni već duže vremena, dok se druge nalaze u fazi studija, pa će tek kasnije moći biti realizirane, ako se za to nadje snage i interesa.

Radovi prikazani u ovom članku ostvareni su na Zavodu za regulacionu i signalnu tehniku (RST) Elektrotehničkog fakulteta u Zagrebu (ETF) u okviru istraživačkih tema financiranih od SIZ-a za znanost i u suradnji s različitim poduzećima (TRS, INA, ATM). Pri tome se u okviru tema SIZ-a i različitih drugih naučnih poduhvata načelno razmatraju i istražuju nove mogućnosti na Zavodu za RST, a zatim se tako dobivena saznanja pokušavaju realizirati u suradnji sa zainteresiranim poduzećima, pa i pojedincima. Dokle se u realizaciji uspjelo otići ovisi u velikoj mjeri o privrednim poduzećima i uopće o našim jugoslavenskim mogućnostima, sa svim onim što to znači. Naime, postavljeni zadaci daleko prelaze mogućnosti grupe okupljene oko Zavoda za RST ETF-a u svakom pogledu. Ovo je trebalo reći na početku da bi se shvatio karakter ovog članka.

Prije nego što se prijedje na daljnja razmatranja pogledajmo koja su to osnovna svojstva mikroračunala koja omogućavaju da se realiziraju novosti u upravljanju procesima. (Ta su svojstva naširoko razmatrana u spomenutim radovima.[1,2,3,4])

To je u prvom redu niska cijena današnjih komercijalno raspoloživih mikroračunala, koja se spušta sve do 2-3 $ po jednom mikroprocesoru ili nekom drugom LSI sklopu. Zbog toga se s mikroračunalima ne mora štediti, barem što se tiče sastavnih djelova. Upotreba nekoliko ili čak nekoliko desetaka mikroračunala za upravljanje samo jednog procesa ne predstavlja ozbiljan ekonomski problem, dok je s većim računalima to bilo nezamislivo. Zbog istog razloga upravljanje mikroračunalima dolazi u obzir i kod najmanjih procesa i naprava. ("Proces" se u ovom članku pretpostavlja u najširem smislu tj. proces je sve ono gdje se nešto dogdja i s čime se može upravljati).

Dalje uz tako nisku cijenu mikroračunala imaju značajnu moć obrade podataka, tako da se za mnoge poslove mogu usporedjivati s većim računalima. Uz sve to dimenzije su im neznatne, tako da u tome pogledu mikroračunala mogu izgledati kao nevažan dodatak senzorima i izvršnim organima, koji predstavljaju veći dio procesnog upravljačkog sistema i po cijeni i po fizičkim dimenzijama. Najveći dio cijene svakako predstavlja rad potreban da se upravljački sistemi koncipiraju, razrade upravljački algoritmi, da se oni implementiraju u programe, te

da se na kraju složе i puste u pogon upravljački sistemi kao cjelina. Sve to skupa ne bi znatno pojeftinilo ni onda kad bi cijelina mikroračunala pala na nulu.

To su eto novosti koje mikroračunala unose u upravljanje i nadzor procesa. Razmotrimo sada kako se te osnovne novosti mogu odraziti na različite vrste upravljanja procesima. Načini na koje su se procesi upravljali prije pojave mikroračunala mogli bi se svesti na ove osnovne tipove:

- Upravljanje različitim digitalnim hardverskim sklopovima pravljenim "po mjeri" za svaki proces
- Elektroničko analogno upravljanje (mehaničko, pneumatsko itd.)
- Upravljanje upotrebom većih računala
- Nije se ranije upravljalo ni nadziralo

(Pod nadzorom se ovdje podrazumijeva mogućnost praćenja rada nekog procesa kao cjeline ili njegovih djelova).

ZAMJENA DIGITALNE HARDVERSKE LOGIKE

Ovo područje stavljamo na početak iz više razloga. Prvi je taj što je prelaz na upotrebu mikroračunala na ovom području za inženjere-automehaničare vjerovatno najjednostavniji. U osnovi mnogo toga ostaje isto ili slično, samo se sklopovska logika djelomično zamjenjuje programskom logikom. Mali programi pisani u strojnom jeziku ili heksadecimalno ne predstavljaju neki značajniji korak za inženjera "hardveraša", a mikroračunalo se ugradjuje kao jedan dio u hardverski sistem. Ono s lakoćom realizira složene digitalne procedure koje predstavljaju značajan problem za digitalne hardverske sklopove. Zbog svega toga je upravo na ovom području kod nas dosta napravljeno i najdalje se otišlo s realizacijom.

No najprije razmotrimo koja su svojstva digitalne hardverske logike i koje dodatne mogućnosti daje upotreba mikroračunala. Svaki proces ili naprava zahtijeva da se za njegovo upravljanje realizira niz funkcija. Te se funkcije mogu dakako ostvariti sa digitalnim logičkim sklopovima, gradjenim upravo za taj proces "po mjeri". Da bi se ta gradnja olakšala često se industrijski proizvode cijeli logički moduli koji obavljaju odredjene funkcije. Za rješenje nekog upravljačkog problema treba iskrojiti upravljački sistem upotrebom i povezivanjem takvih modula, što je dakako lakše napraviti, nego sve graditi od početka od osnovnih sklopova, diskretnih ili integriranih. No bez obzira na način gradnje takvih upravljačkih sistema očito je da realizacija sve složenijih upravljačkih funkcija zahtijeva sve više različitih sklopova, kartica, ormara u koje je sve to smješteno, izvora napajanja, ventilatora itd. Takva se upravljačka logika može napraviti, ali ona postaje sve glomaznija i skuplja, što se od nje traži da obavlja složenije zadatke [5,6]. Svaka i najmanja promjena se veoma teško izvodi, a još se teže može na nju odlučiti, jer to može onemogućiti, makar i na najkraće vrijeme, rad velikih i skupih postrojenja. Kolikogod to izgledalo naučnotehnički opravdano malo se tko može odlučiti da na upravljačko-mjernom sistemu napr. jednog velikog naftnog polja doda jedan bistabil, promjeni univibrator ili nešto slično.

S druge strane mikroračunala izvode s lakoćom složene programe, nedostižne hardverskoj logici, a da pri tome imaju neznatne dimenzije, malu potrošnju i disipaciju, a uz sve to im se programi mogu mijenjati i prelaziti na novi način rada samo zamjenom ROM memorija s novim programima koji se mogu razviti i provjeriti bez direktne veze sa upravljanim procesom. Pitanje koje se sada neizbježno nameće je a zašto se onda ipak ponekad upotrebljava upravljanje samo hardverskom logikom. Najčešći razlog je taj da su konstruktori ovladali s upotrebom hardverske logike, dok još ne vladaju s upotrebom mikroračunala. No ponekad se hardverska logika može upotrijebiti i onda kad se vlada mikroračunarskom tehnikom. To će se najbolje objasniti realiziranim primjerima upravljanja i mjerenja na naftnim poljima. U jednom slučaju [5,6] upravljačke funkcije su veoma jednostavne, ali ima velik broj vanjskih ulaznih i izlaznih mjesta (pedesetak bušotina i spojni naftovod), pa je dobivanje podataka s velikog broja mjesta i slanje podataka na velik broj mjesta osnovni problem za koji mora postojati odgovarajući hardver. Sama obrada dobivenih podataka je relativno jednostavna, pa ne predstavlja neki ozbiljniji problem ni za hardversku logiku. S druge strane manji broj različitih upotrebljenih tehnologija je prikladniji za održavanje na terenu od manje kvalificirane radne snage. Zbog toga je u tome slučaju upotrebljena isključivo hardverska logika.

Prema tome moglo bi se zaključiti da će se mikroračunalo upotrijebiti tamo gdje su funkcije nešto složenije, te gdje se zahtijeva određena fleksibilnost funkcija. Tamo gdje su funkcije jednostavne, ali treba, zbog drugih namjena, mnogo hardverskih sklopova, uporeba mikroračunala ne mora predstavljati prednost. Ovo posljednje vrijedi za standardna mikroračunala opće namjene.

No baš za takve namjene pojavila su se specijalizirana mikroračunarska rješenja tj. za obavljanje jednostavnih upravljačkih funkcija na velikom broju mjesta. Takvo je napr. jednostavno mikroračunalo PLC-700, namjenjeno za "on-off" upravljanje procesima. Ono ima svega nekoliko naredbi prilagodjenih osnovnim logičkim operacijama (AND, STO, OR, COMPL itd.) i veoma se jednostavno programira. Ulazi i izlazi su mu jednobitni tj. podatke prima samo iz izvora koji mogu biti u stanju 0 ili 1 (otvoreni ili zatvoreni kontakt), a isto tako može davati naredbe samo za otvaranje i zatvaranje kontakata. No takvo računalo može uspješno upravljati i primati podatke sa stotina različitih mjesta. Zbog toga je upravo takvo mikroračunalo upotrebljeno u drugom slučaju naftnog polja, koje je kasnije realizirano[7,8], a ima oko 700 vanjskih veza. No napomenimo još jednom da je to specijalizirano mikroračunalo, gradjeno upravo za takve namjene. Ono inače za svoj interni rad upotrebljava 12.bitnu riječ, a sastoji se od uobičajenih elemenata, procesora, PROM-a itd. i može raditi sa 1024 ulazno-izlazna signala.

Značajna moć obrade podataka omogućava da se s mikroračunalima realiziraju složene funkcije koje bi u principu bilo moguće rješavati i s hardverskom logikom, ali bi rezultiralo krutim i glomaznim sklopovima. Upotreba mikroračunala smanjuje količinu upotrebljenih sklopova, dimenzije, disipaciju itd. Dakako za ovakve zadatke će se upotrijebiti standardno mikroračunalo, koje pri tome dolazi do izražaja upravo sa svojom moći obrade podataka i prema tome mogućnostima za izvodjenje složenijih funkcija. Broj vanjskih mjesta s kojima se komunicira igra pri tome manju ulogu.

Kao primjer takve upotrebe mikroračunala može se navesti upravljanje Uredjajem[9,10] za magnetsku registraciju digitalnih podataka TRS-103. Uredjaj kojim se upravlja može se sastojati od 1 do 4 kasete s magnetskim trakama za zapis podataka, kao i svim potrebnim mehaničkim i elektromagnetskim sklopovima za zapis podataka na magnetsku traku. Ovakav uredjaj služi kao vanjska jedinica u računarskim sistemima, te s njome upravlja "glavno" računalo s kojim je uredjaj povezan. Upravljanje s takvim uredjajem je relativno složeno, a može se u principu napraviti na dva osnovna načina. S radom sklopova za zapis podataka upravlja u većoj ili manjoj mjeri digitalna sklopovska logika koja se pridruži uredjaju s trakama. Što više poslova obavlja ta logika ona je složenija sa svim drugim navedenim nedostacima. Može dakako ta logika obavljati samo najnužnije funkcije, a glavni dio posla oko upravljanja preuzima "glavno" računalo. No time se glavno računalo značajno opterećuje sa sporednim poslovima za periferijsku jedinicu, a na uštrb svojeg osnovnog zadatka. Posebno mikroračunalo koje upravlja svim funkcijama potrebnim jedinicama magnetskih traka, predstavlja gotovo idealno rješenje. Snaga upotrebljenog mikroračunala ZILOG Z80 je dovoljna da obavi sve potrebne funkcije upravljanja, a da mu pri tome ostane dovoljno raspoloživog vremena za provjeru ispravnosti prenosa podataka metodom dijeljenja s polinomom[10] (CRCC) i prenošenjem dobivenog ostatka.

Glavno računalo komunicira s uredjajem magnetskih traka upotrebom desetak makroinstrukcija, pisanih u obliku koda, koje upravljačko mikroračunalo interpretira i izvede. Za izvodjenje svake takve makroinstrukcije odigrava se dosta složen program u upravljačkom mikroračunalu. Takve makroinstrukcije su napr.: Čitaj blok, Piši blok, Piši oznaku polja, Izbaci kasetu, Izbriši kanal itd. Glavno računalo prenese svoje zahtjeve u posebnu memoriju vanjske jedinice, pa nastavlja sa svojim radom ne čekajući vanjsku jedinicu da obavi svoj posao. Mehanički sklopovi su dakako znatno sporiji, pa ne bi imalo smisla, da glavno računalo čeka vanjsku jedinicu. Čak što više glavno računalo može jedinici magnetskih kaseta uputiti i nove naredbe prije nego što se predhodne izvrše i one se čuvaju u memoriji. Ovako složeno i ekonomično upravljanje bi bilo teško realizirati na bilo koji drugi način. Uz to se mogu napraviti i promjene načina rada modifikacijama programa zapisanih u PROM memorijama. Treba naglasiti da je ova jedinica pripremljena za serijsku proizvodnju u TRS-u.

Još složeniji način rada predstavlja tzv. "inteligentno" ponašanje upravljačkog sistema. Takvo ponašanje znači da sam upravljački sistem

odlučuje o načinu upravljanja koje će biti primjenjeno u zavisnosti o stanju upravljanog procesa. Samo takvo upravljanje ne predstavlja neku naročitu novost. Ono se upotrebljava već dosta dugo, no za takvo upravljanje su ranije bila potrebna u najmanju ruku miniračunala ili ona veća od njih. Hardverska logika je nepraktična za ostvarivanje tako složenih zadataka. S mikroračunalima takvo upravljanje mogu imati i najmanji procesi. Takav način upravljanja će biti zbog toga prikazan kasnije kao zamjena velikih računala, na primjeru upravljanja semaforima na raskršću.[11]

MIKRORAČUNALO KAO ZAMJENA ANALOGNOG REGULATORA

Osnovni analogni sklop za reguliranje procesa je analogni regulator, koji usporedjuje stvarnu vrijednost neke veličine iz reguliranog procesa sa željenom veličinom. Dobivena razlika se nakon pojačanja vodi kao korektiv s odgovarajućim predznakom u proces. To znači da analogni sklopovi obavljaju funkcijju reguliranja veličine. Standardno upotrebljavani sklop u industrijskim poduzećima je tzv. PID regulator, koji ima proporcionalni, integralni i derivacioni član u regulacionom izrazu.

Već su se i ranije digitalna računala upotrebljavala da svojim programom izvode algoritme koji u procesu obavljaju istu ili veoma sličnu funkciju kao što je izvode analogni regulatori. No veliko računalo stoji neusporedivo više nego analogni regulator, a s druge strane je njegova moć znatno veća nego što je potrebno za oponašanje rada samo jednog regulatora. Zbog toga je iz ekonomskih razloga moglo doći u obzir samo to da digitalno računalo zamjenjuje više desetaka ili čak stotina analognih regulatora. To drugim riječima znači da upravljanje cijelog jednog velikog procesa ili barem njegovog značajnog dijela ovisi samo o jednom računalu. U slučaju kvara ili pogrešnog rada tog računala morao bi se prekinuti rad cijele tvornice ili nekog sličnog procesa. Može se dakako uvesti redundantni sistem s računalom ili bez njega, ali to opet poskupljuje cijelu izvedbu.

Pri upotrebi mikroračunala i na ovom području dolaze do izražaja mala cijena i velika moć obrade podataka. Mikroračunalo može bez velikih teškoća zamijeniti rad desetak analognih regulatora izvodeći svojim programom odgovarajuće upravljačke algoritme u vremenskom multipleksu tj. posvećujući svoju pažnju prvoj, drugoj itd, regulacionoj petlji. Proizvodnja jednog takvog univerzalnog regulatora koji bi realizirao 8 regulacionih petlji pomoću mikroračunala MOTOROLA 6800 pripremljena je za tvornicu ATM Zagreb.[12] Pri tome mikroračunalo ni iz daleka ne radi svojom maksimalnom snagom samo na realizaciji regulacionih algoritama za upravljačke petlje, već ima dovoljne radne mogućnosti da se može posvetiti i drugim poslovima, kao što je komuniciranje s operatorom-tehnologom, koji upravlja s odvijanjem procesa s odgovarajućeg upravljačkog stola. Nadalje kao još jedna neiskorištena mogućnost postoji i to da se za upravljanje izvode i složeniji upravljački algoritmi, nego što su oni koje izvode analogni regulatori. U petlji se napr. ne mora raditi s konstantnim pojačanjem kao kod analognih regulatora, već se pojačanje u petlji može napraviti takvo da ovisi o nekom parametru kao što je veličina ulaznog signala ili slično.

Ovakav univerzalni regulator može upravljati dijelom nekog procesa uz veoma prihvatljivu cijenu. U tome slučaju, osim toga, ne ovisi cijeli proces samo o jednom računalu, pa je pouzdanost rada veća. Upravljanje se može uvoditi postepeno, dio po dio, a ne odjednom kao kod upotrebe velikog računala.

UPOTREBA MIKRORAČUNALA ZA UPRAVLJANJE UMJESTO VELIKIH RAČUNALA

Jedna od mogućnosti upravljanja s mikroračunalima koju je ranije bilo moguće ostvariti samo s većim računalima već je bila navedena. To je "inteligentno" upravljanje malih procesa za koje se ne bi isplatilo upotrijebiti veća računala, a drugim načinima, zbog ograničenih računskih mogućnosti, to nije moguće napraviti. Takvo upravljanje pomoću mikroračunala će biti prikazano na primjeru relativno malog i jednostavnog sistema upravljanja cestovnim saobraćajem pomoću semafora.[11] To upravljanje ima nekoliko elemenata koji doprinose "inteligentnom" ponašanju tj. snalaženju u novoj situaciji. Ovdje će biti navedena tri takva elementa.

Osnovni senzor u upravljanju semaforom predstavljaju induktivne magnetske petlje ugradjene u kolovoz i spojene s mikroračunalom. One registriraju prolaz vozila signalom dobivenim promjenom magnetskog polja zbog prolaza vozila iznad petlje. Upotrebom takvih senzora može se ustanoviti broj vozila, kao i njihova brzina.

Može se napr. predpostaviti da će se na prilazu semaforu vozila kretati brzinom od 10 do 70 km/h. Brzina od napr. 250 km/h ili 0 km/h bila bi nerazumna i mogla bi dolaziti zbog kvara na senzoru. Zbog toga računalo prije uzimanja u razmatranje podataka dobivenih od senzora provjerava da li su oni "razumni". Ako se brzine nalaze u području od 10 do 70 km/h, onda su dobiveni podaci razumni i uzimaju se u obzir za daljnje razmatranje. Ako je naprotiv brzina vozila 0 km/h smatra se nerazumnom i ne uzima se u obzir neko vrijeme. Nakon recimo 10 min. ponovo se uzima podatak od senzora s predhodno nerazumnom brzinom od 0 km/h. Ako se od njega i dalje dobivaju isti rezultati oni se i nadalje ignoriraju, ali se i nadalje povremeno (recimo nakon svakih 10 min.) provjeravaju podaci sa tog senzora. Kad se konačno konstatira da inkriminirani senzor daje opet razumne podatke, oni se počimaju uzimati u obzir za daljnju obradu (napr. vozilo u kvaru iznad induktivne petlje je uklonjeno, te ona opet daje razumne podatke).

Na sličan način se može provjeravati da li su naredbe izdane izvršnim organima izvršene ili ne. Dade se napr. naredba za paljenje crvenog svjetla. Nakon toga se pomoću odgovarajućeg senzora provjerava da li je crveno svjetlo zaista upaljeno ili ne. Ako nije izdaje se još nekoliko uzastopnih naredaba za njegovo paljenje, dakako svaki put uz provjeru. Ako se ni nakon 4-5 takvih naredbi crveno svjetlo ne upali, konstatira se da je semafor u kvaru i prelazi se na upravljanje paljenjem žutog žmigavog svjetla tj. rad se i dalje odvija, ali na degradiran način (havarijski), koji doduše ne omogućava najefikasniji prolaz vozila kroz raskršće kao u slučaju normalnog funkcioniranja, ali se posao (prolaz vozila) ipak nekako odigrava.

Daljnji elemenat toga inteligentnog ponašanja može biti upotreba odgovarajućih "signalnih planova", zapisanih u memoriji računala. Ti planovi predstavljaju neku vrstu najbolje izmjene svjetala na semaforu za različite slučajeve (ujutro: polazak na posao, prije podne, popodne, povratak s posla, veče, noć, petak popodne itd.). Svaki se takav signalni plan na osnovu proučavanja prometa, upotrebljava kao norma upravljanja, koja u datim uvjetima daje najbolje rezultate. Ipak se provjerava da li u stvarnoj situaciji baš taj signalni plan omogućava maksimalni protok vozila. Ako ne osigurava uključuje se drugi plan koji više odgovara (napr. nešto se izvanredno dogodilo na prilazu k semaforu). Tvrdoglava upotreba uvijek istih signala planova bez obzira na realnu situaciju, je primjer složenog ali neinteligentnog upravljanja.

Naglasimo to još jednom da inteligentno ponašanje zahtijeva značajnu moć obrade podataka kakvu su ranije mogla pružiti samo veća računala, ali ne i upotreba hardverske logike ili drugih starijih načina (analogne tehnike) sa skromnijim mogućnostima manipuliranja podacima. Sada eto mikroračunala omogućavaju takvo ponašanje i za tako jednostavan sistem kao što je to raskršće, a da to pretjerano ne opterećuje troškovima svaki semafor. Štoviše i mnogima još jeftinijim sistemima nego što je raskršće se može omogućiti inteligentan rad, ako se može proizvesti velika serija, tako da se razvoj sistema, koji stoji najviše, raspodjeli na veliki broj proizvedenih komada. Spomenimo još usput i to da se signalni planovi mogu praviti simuliranjem raskršća na velikom računalu na "offline" način. Ovaj sistem još za sada nije realiziran već je samo napravljena studija, koja predstavlja osnovu za prelaz na realizaciju.

Drugi primjer zamjene rada velikih računala mikroračunalima je upotreba više mikroračunala za upravljanje jednog procesa. Naime cijena mikroračunala je tako niska da upotreba većeg broja računala za upravljanje samo jednog procesa ne predstavlja problem što se tiče cijene samih računala. Štoviše i za obavljanje samo jedne funkcije može se upotrijebiti više mikroračunala, a da to ne bude preskupo. Pitanje je samo kako će se to učiniti.

Već je bilo istaknuto da upotreba jednog računala koje upravlja procesom kao cjelinom može uskladiti rad procesa tako da se dobiju optimalni rezultati, gledajući globalno. S druge strane je loše što cijeli proces ovisi o radu samo jednog računala. Uz upotrebu mikroračunala upravljanje se može decentralizirati, tako da svako mikroračunalo upravlja jednim djelom procesa malo ovisno ili potpuno neovisno od drugih djelova procesa. U tome slučaju ono može sasvim uspješno obavljati svoju funkciju lokalnog upravljanja, ali ne postoji koordiniranje rada na nivou procesa kao cjeline.

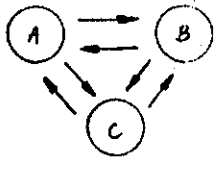
Može se doduše postaviti hierarhijska organizacija kod koje je jedno računalo "glavno" i ono se brine o radu procesa kao cjeline, a ostala obavljaju lokalne funkcije nižeg ranga. No tada opet rad procesa ovisi o jednom računalu. Ako ono ispadne iz pogona rad procesa se ipak može nastaviti samo bez koordiniranja na nivou procesa. Lokalno distribuirano upravljanje nastavlja svoj rad. Način rada je time doduše degradiran, ali u nekim slučajevima može biti prihvatljiv. Već i to predstavlja poboljšanje koje omogućavaju mikroračunala. Daljnja prednost koju pruža takav sistem sastoji se u tome da se upravljanje može uvoditi postepeno, uvodjenjem pojedinih lokalnih upravljačkih funkcija, koje se odnose samo na dio procesa. Tek kad lokalne funkcije uspješno rade može se preći na instaliranje globalnog upravljanja procesom.

No da li se može realizirati distribuirano upravljanje bez hierarhijske organizacije s "glavnim" računalom? Distribuirano upravljanje procesom sa većim brojem mikroračunala koja koordiniraju rad na nivou procesa medjusobnim "dogovaranjem" tj. izmjenom poruka " od medjusobnog interesa", a bez "glavnog" računala bio bi korak dalje u decentraliziranom upravljanju.

Kod upotrebe manjeg broja lokalnih mikroračunala upravljanje se može organizirati tako da svako računalo saobraća direktno sa svakim. Za 2 i 3 mikroračunala to je prikazano na sl.1. Tako bi se mogao koordinirati rad na upravljanju manjih procesa, mada i kod tako malih sistema može dolaziti do konfliktnih situacija, koje ne bi doprinosile kvaliteti upravljanja. Uz upotrebu većeg broja lokalnih računala (10 do 20) očito je da situacija postaje mnogo složenija, jer je potreban velik broj komunikacionih kanala, a osim toga još lakše dolazi do konfliktnih situacija. Uz sve to lokalna mikroračunala bi trošila najveći dio svoga rada na medjusobno dogovaranje, tako da bi jedva mogla išta raditi na stvarno upravljačkim poslovima.

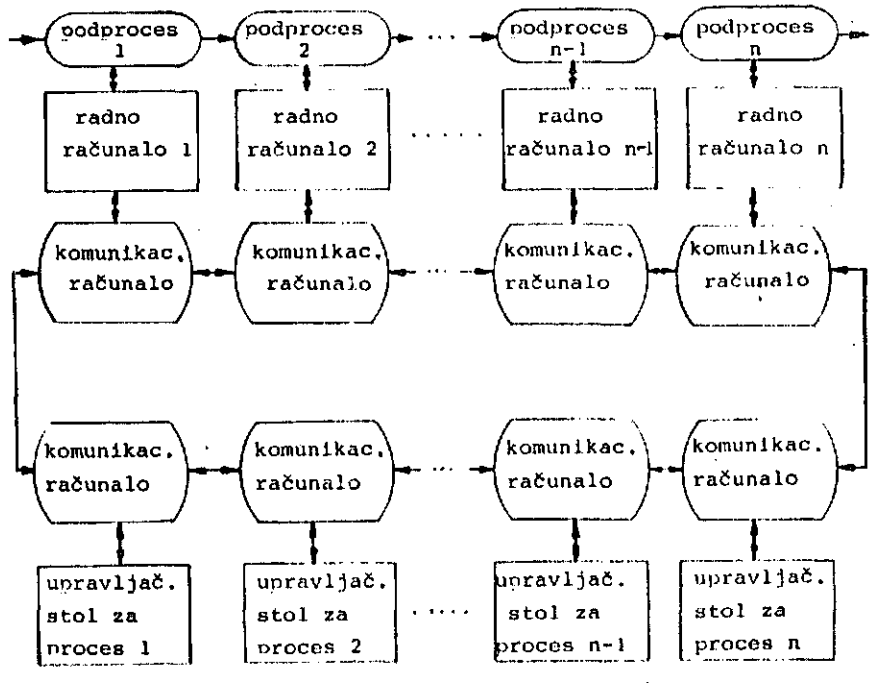
Jedna analiza rada programera,[13] koji rade na sličan način, pokazala je da za isti zadatak za koji jedan programer treba 1 godinu rada, dva programera troše 26% više vremena, radi potrebe za dogovaranjem i uskladjivanjem. Tri

a) komuniciranje izmedju dvaju računala A i B



b) komuniciranje izmedju triju računala A,B i C

slika 1



slika 2

programera bi već trošila 30 do 40 % više vremena itd. Ako bi taj isti posao radilo 10 programera, oni, teoretski gledano, ne bi napravili ništa korisnog, već bi se samo medjusobno dogovarali i uskladjivali. Ovaj primjer je iz drugog područja, ali je način rada veoma sličan. I kvantitativni podaci se mogu uzimati s rezervom, no osnovni zaključak da uz povećani broj sudionika u dogovaranju "svaki sa svakim", sve više vremena se troši na dogovaranje, a sve manje ga preostaje za stvarno koristan rad, sigurno vrijedi. Prema tome sistem dogovaranja "svaki sa svakim" kod većeg broja sudionika ne dolazi u obzir.

Da bi se ipak omogućilo uskladjeno decentralizirano upravljanje procesa, a da se pri tome izbjegne hierarhijska upotreba "glavnog" računala, predlaže se tzv. prstenasta organizacija povezivanja lokalnih mikroračunala[14]. Ta se organizacija zasniva na ovim osnovnim pretpostavkama: Svako lokalno mikroračunalo je do maksimuma orjentirano na upravljanje svojeg dijela procesa, pa je izmjena poruka s drugim isto takvim računalima svedena na minimum, ali je ipak tolika da se može uskladiti rad procesa kao cjeline. Dalje, uz svako radno mikroračunalo tj. ono koje upravlja lokalnim djelom procesa pridruženo je još jedno komunikacijsko mikroračunalo, koje se brine o komuniciranju izmedju "njegovog" radnog mikroračunala i ostalih radnih mikroračunala. Komunikacijska mikroračunala su standardizirana, pa treba razviti samo jedno. Ostala se samo multipliciraju. Time se štedi na radu potrebnom za razvoj sistema. Sva su računala spojena u prsten (sl.2.), a poruke s adresom se šalju samo susjednom računalu, koje ih prima i izvršava sve potrebno u vezi s njima, ako su namjenjene njemu. Ako nisu namjenjene njemu poruke se u slijedećem koraku šalju susjednom računalu, koje postupa na isti način itd. Kad poruka dodje na svoje odredište, tamo se prihvaća i zadržava. Poruke se šalju od njihova izvora na desnu i na lijevu stranu, tako da postoji kraći i duži put slanja poruka. Ako je jedan put u kvaru poruka ipak stiže dužim putem, tako da je redundantnošću povećana pouzdanost. Svaki proces ima svoje upravljačko mjesto s kojeg operator može pratiti odvijanje procesa i utjecati na njega slanjem poruka preko komunikacijskog računala na prsten, kao što to rade i radna računala. Jedan takav stol može služiti za upravljanje procesa kao cjeline.

Realizacija ovakvog načina upravljanja na nekom konkretnom sistemu je dakako značajan poduhvat, koji se ne može napraviti bez znatnih ulaganja, tako da ovo za sada predstavlja samo predhodnu studiju, koja još čeka na mogućnost realizacije i provjere u praksi.

UPRAVLJANJE S MIKRORAČUNALIMA ONOGA ŠTO SE RANIJE NIJE UPRAVLJALO

Jedno od najznačajnijih područja za primjenu mikroračunala u upravljanju predstavljaju nove primjene tj. upravljanje onoga što se ranije nije upravljalo klasičnim metodama. To su najčešće tzv. "male" primjene u proizvodnji i širokoj potrošnji, za koje su stariji načini upravljanja bili ili preskupi ili neprikladni iz bilo kojeg drugog razloga, pa nisu dolazili u obzir. Naprotiv mikroračunala svojom niskom cijenom i malim dimenzijama omogućavaju da se upravljaju i oni najmanji procesi i naprave. Kad kažemo najmanji onda u prvom redu mislimo na cijenu. S mikroračunalima se mogu inteligentno upravljati i tako male naprave koje stoje nekoliko hiljada novih dinara, a da pri tome cijena tih naprava ne bude znatno povećana. To mogu biti strojevi i naprave koje služe u proizvodnji (različiti mali strojevi i mjerni aparati) ili naprave u širokoj potrošnji, najčešće u "domaćinstvu" (pećnice, šivaći strojevi, grijanje i zaključavanje stana, paljenje automobilskog motora itd.). Jedan primjer takvog malog upravljačkog sistema je tzv. "kućno" računalo zasnovano na mikroprocesoru. Ono obavlja , osim mnoštva drugih poslova, još i razne poslove oko upravljanja i zaštite stana.[15] Ono napr. programirano uključuje i isključuje električna trošila.Svako trošilo ima svoju adresu, pa se prema odabranom programu uključuje i isključuje bojler, televizor, pećnica, radio, grijanje pojedinih prostorija itd. Može se napr. isprogramirati uključivanje pećnice za kuhanje ili podgrijavanje jela pola sata prije povratka kući s posla, uključenje grijanja stana ili samo jedne sobe nekoliko sati prije povratka s vikenda (ili grijanje vikendice nekoliko sati prije dolaska u nju), televizora kada dodje vrijeme odredjene emisije itd. Svako od kućnih trošila ima svoju adresu (maksimalno se može upravljati s 64 trošila) kojom računalo aktivira tu jedinicu. Adresa se šalje VF signalima po električnoj mreži. Uz sve to postoji i elektronički sat koji pokazuje vrijeme i brine se o tome da se programirani poslovi naprave kad to

treba. Čovjek može pomoću posebnog prenosivog odašiljača preuzeti upravljanje trošila od računala.

Na kućno računalo mogu se priključiti i različiti senzori, koji osiguravaju stan od nepoželjnih pojava kao što je povišena temperatura, pojava dima ili prisutnost plina, vlažnost itd. signalizira se i pokušaj otvaranja vrata ili prozora na neuobičajen način itd. itd,

Osim ovakvih upravljačkih funkcija, računalo obavlja i niz drugih poslova koji ne spadaju u upravljanje, ali mogu biti veoma korisni u kući kao što su različite igre, pomoć u učenju, dobivanje različitih informacija iz nekog centra (*prognoza vremena, stanje na cestama, program* kazališta, kina i sportskih priredaba itd.). Takvo računalo uz sve to nije preskupo (računalo KAG A2). Ono stoji oko 1100 $, a može se nabaviti i za dinare ("Velebit", Zagreb, OOUR kompjutera). Ovo je dakako samo jedan primjer, koji može ilustrirati ovo područje, a moguće su i nebrojene druge slične primjene.

LITERATURA

1. G. Smiljanić — Mikroprocesori i mogućnosti njihove proizvodnje i primjene u Jugoslaviji, Automatika 1977,br.1-2,str.3

2. G. Smiljanić — Praćenje i upravljanje procesa s mikroračunalima, JUREMA 1979, 1 svezak str.101

3. G. Smiljanić — Specifičnosti upotrebe mikroračunala za praćenje i upravljanje procesa, Informatica, 1978, br.4,str.26

4. G. Smiljanić — Masovna upotreba digitalnih računala za upravljanje u proizvodnji i širokoj potrošnji i njezin odraz na ekonomske i društvene tokove, Automatika (u štampi)

5. M. Petrinović, G. Smiljanić, B. Jeren, Ž. Šipek, M. Vuković — Elektroničko mjerenje i upravljanje na naftnom polju, JUREMA 1977, svezak I, C-18, Teorija i primjena automatike

6. M. Vuković, G. Smiljanić, M. Petrinović — Upravljanje i mjerenje na sabirno-transportnom sistemu naftnog polja "Bilo", JUREMA 1979, 1 svezak str.105

7. M. Petrinović — Privatno saopćenje

8. Ž. Šipek, M. Živković, M. Petrinović, D. Čorak, T. Kadić — Elektroničko upravljanje i mjerenje na mjernoj i otpornoj stanici naftnog polja, JUREMA 25 (1980), 1 svezak, str. 143.

9. M. Žagar, N. Rendić — Razvoj uredjaja za magnetsku registraciju digitalnih podataka TRS 103, ETF Zagreb 1980, Izvještaj za TRS

10. M. Žagar — Upravljanje kompjuterskih periferijskih jedinica s mikroprocesorima, ETF Zagreb 1978, Magistarski rad

11. A. Jergović — Analiza upravljanja cestovnog saobraćaja pomoću kompjutera, ETF Zagreb, 1979, Magistarski rad

12. Ž. Žuvela — Univerzalni mikroprocesorski sistem regulatora ETF Zagreb, 1980, Magistarski rad

13. — Accelerated microcomputer development with the vew HP 6400 System, Hewlet-Packard Seminar, Zagreb 15.IV.1980.Hotel Interkontinental

14. M. Kukrika — Problemi organiziranja sistema s više mikroračunala za vodjenje procesa, ETF Zagreb, 1980, Magistarski rad

15. J. Makanec — Privatno saopćenje

OSNUTEK BIPOLARNEGA MIKROPROCESORJA

G. BREZNIK,
M. GERKEŠ,
M. DRUŽOVEC,
V. ŽUMER

VISOKA TEHNIŠKA ŠOLA MARIBOR, VTO ELEKTROTEHNIKA
62000 MARIBOR, Smetanova 17

UDK: 681.3.06

V članku opisujemo izvedbo 16-bitnega bipolarnega mikroprocesorja z elementi družine 2900. Procesor ima paralelno 2 x 8 bitno zgradbo in je s tem univerzalen za vse standardne formate, ki so kodirani v dvojiškem komplementu. Sistem je izveden mikroprogramsko s horizontalno zgradbo mikroinstrukcije. Pri razvoju in testiranju mikroprogramov je kot vhodno-izhodna enota uporabljen host-računalnik. Razvit je simbolični zapis za mikroprograme. Izdelani so mikroprogrami za izvajanje osnovnih matematičnih funkcij za stalno in za premično vejico. Kot zgled dajemo množenje dveh 16-bitnih števil.

THE DESIGN OF A BIPOLAR MICROPROCESSOR: The article describes a 16-bit bipolar microprocessor with 2900 family elements. The processor has a parallel 2 x 8-bit construction and is thus universal for all standard-format data coded in the two's complement. The system has a horizontal microinstruction. By development and testing procedures an host-computer was used. Developed were microprogrammes for basic mathematical functions with fixed- and floating-point using a symbol record. The example shows multiplication of two 16-bit numbers.

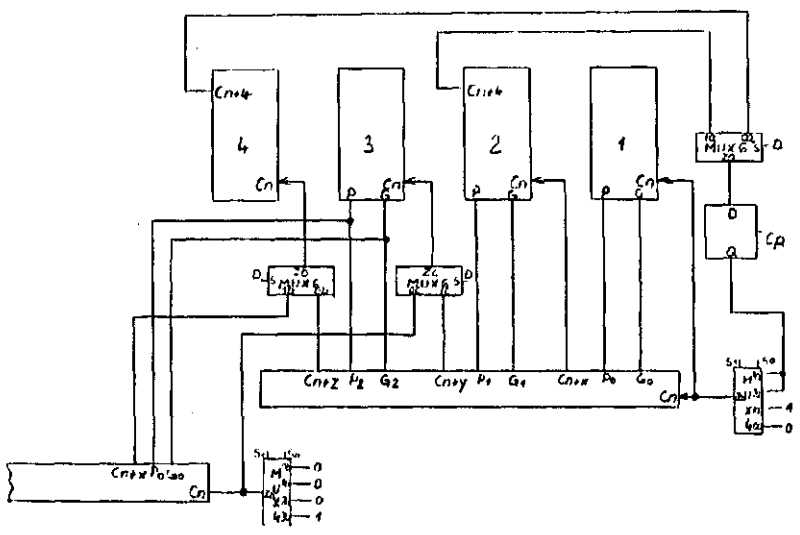
UVOD

Glavna in bistvena razlika med običajnimi mikroprocesorji in bipolarnimi (mikroprogramiranimi) mikroprocesorji je v arhitekturi. Pri običajnih mikroprocesorjih so funkcije za procesiranje podatkov in za krmiljenje v enem čipu, medtem ko je to pri bipolarnem mikroprocesorju izvedeno z več čipi. Razlika je tudi v tem, da imajo običajni mikroprocesorji vnaprej definirano in stalno dolžino besede, prav tako arhitekturo in nabor ukazov. Bipolarni mikroprocesorji omogočajo poljubno arhitekturo in dolžino besede. Projektant mora pri delu z bipolarnimi mikroprocesorji poznati tako hardware kot software, kar ima za posledico težje in dolgotrajnejšo delo. Tudi mikroukazi, ki se tu uporabljajo, so bolj zamotani kot ukazi na običajnem mikroračunalniškem nivoju ali na nivoju zbirnega jezika.

Bipolarni mikroprocesorji uspešno nadomeščajo klasična digitalna vezja, posebno če naj le-ta opravljajo zahtevnejše naloge. Z njimi učinkovito rešujemo procesiranje podatkov tudi po zapletenih algoritmih. Tako n.pr. lahko s 16-bitno verzijo mikroprogramiranega sistema dobimo rezultat množenja, deljenja, seštevanja ali odštevanja v plavajoči vejici (kjer je mantisa 24 bitna in eksponent 8 biten) v manj kot 15 μs, elementarne funkcije pa hitreje od 100 μs.

ZGRADBA

Za procesno enoto smo uporabili štiri bipolarne mikroprocesorje 2901. Ti vsebujejo aritmetično in logično eno-



Slika 1

to, enoto za pomik, pomikalni register in 16 naslovni RAM (delovni registri) z dvema izhodoma A in B.

Procesna enota je zgrajena, da lahko izvaja operacije s šestnajst bitnimi in z dvaintrideset bitnimi operandi, kakor tudi z operandi, ki so kodirani v premični vejici in imajo štirindvajset bitov za mantiso in osem bitov za eksponent.

Vidimo, da lahko operacije s šestnajst in z dvaintrideset bitnimi operandi izvajamo brez večjih težav. Organizacija v premični vejici pa zahteva izvajanje v dveh delih. Najprej izvajamo operacije s šestnajstimi biti operandov z nižjo utežjo, nato pa še z osmimi biti operandov z višjo utežjo ter z eksponentom. Zato moramo definirati kdaj uporabimo šestnajst bitno konfiguracijo procesne enote dvakrat po osem bitov ali šestnajstbitno. To definiramo s posebnim krmilnim bitom. Slika 1 prikazuje povezavo procesorjev za izvajanje aritmetičnih operacij. Povezava prenosnih bitov med procesorji je izvedena z dvema "carry look ahead" generatorjema.

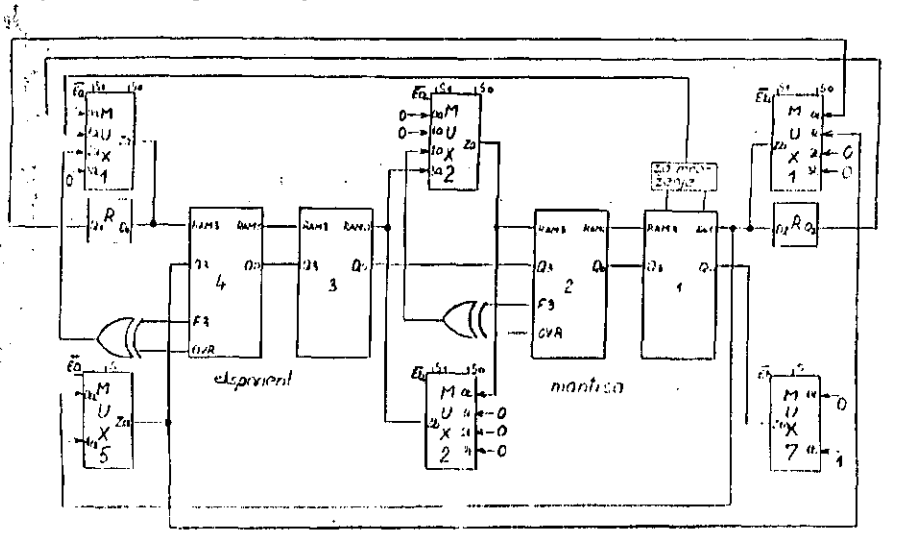
Kadar sistem ni razdeljen na dva dela, uporabljamo prvi "carry look ahead" generator za vse štiri mikroprocesorje. Kadar pa uporabljamo obe procesni enoti ločeni, služi prvi "carry look ahead" generator prvemu in drugemu mikroprocesorju (mantisa), drugi "carry look ahead" generator pa tretjemu in četrtemu mikroprocesorju (eksponent). Prenos v mikroprocesor 1 je izveden preko multiplekserja (MUX 4), ki nam daje možnost, da izbiramo prenosni bit, kot je podano v tabeli I.

S_1	S_0	A	B
0	0	0	0
0	1	1	0
1	0	C_n	0
1	1	C_n	1

Tabela I

C_n je prenosni bit pri aritmetičnih operacijah, kjer je operand večji od 16 bitov.

Na sliki 2 je narisana povezava procesorjev, ki je potrebna za izvajanje pomikov. Ta povezava omogoča pomike 2 x 8 bitnih ali 16 bitnih operandov. Na izbiro imamo logični pomik, aritmetični pomik in poseben pomik, ki poenostavlja množenje.



Slika 2

KRMILNA ENOTA

Krmilno enoto (slika 3) sestavljajo sekvencer za mikroprograme 2909, krmilnik pogojnih skokov 29803, multiplekser za izbiro pogojnih bitov, števec iteracij in mikroprogramski pomnilnik z vmesnim registrom. Navedena vezja omogočajo izvajanje naslednjih funkcij:

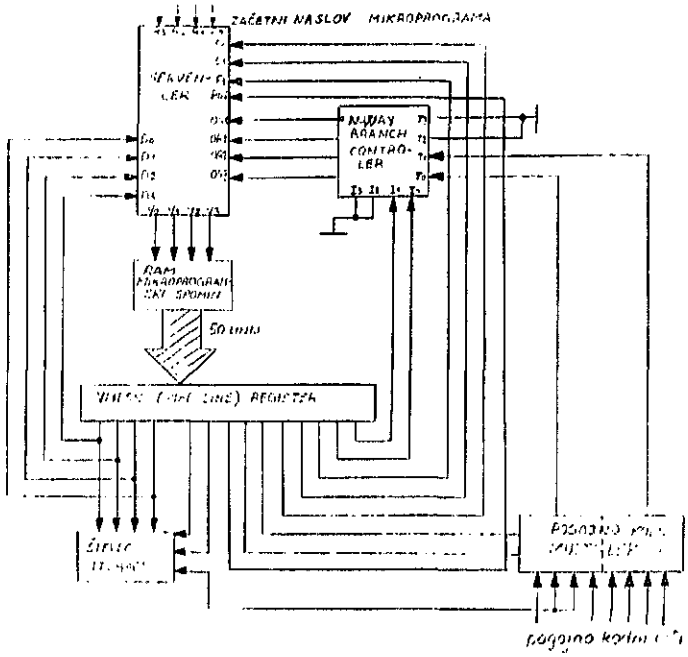
1. povečevanje mikroprogramskega števca za ena
2. skok na poljubni naslov (nepogojni skok)
3. izvajanje pogojnih skokov
4. krmiljenje zanke
5. skok v mikrosubroutino (možna je vgnezditev do globine 4).

POVEZAVA MIKROPROGRAMIRANEGA SISTEMA Z OKOLICO

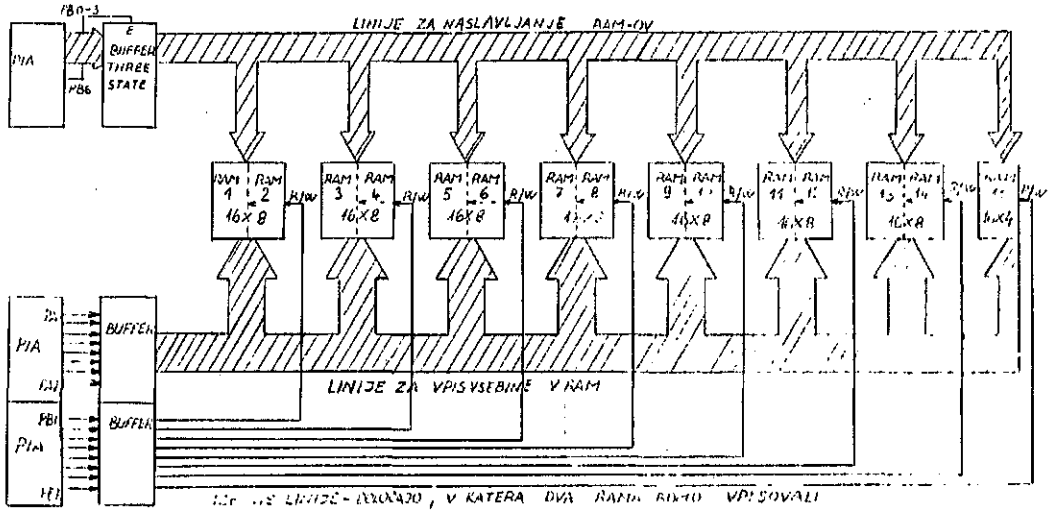
Do sedaj opisani mikroprogramirani mikroprocesor je neprimeren za komunikacijo z okolico. V našem primeru smo si omogočili dostop vanj s pomočjo mikroračunalnika ISKRA DATA 1680.

Narejeni sta dve povezavi. Prva povezava (slika 4) omogoča preko dveh perifernih povezovalnikov (MC 6820) vnašanje mikroprogramov v mikroprogramski pomnilnik. Preko druge povezave, ki je shematsko prikazana na sliki 5, lahko mikroračunalnik prenaša operande z mikroprogramiranim sistemom in zahteva izvajanje določenega mikroprograma. Na mikroračunalniku je narejena programska podpora, ki navidezno neposredno povezuje uporabnika ob teleprinterju z mikroprogramiranim sistemom. Tu imamo pet monitorskih ukazov:

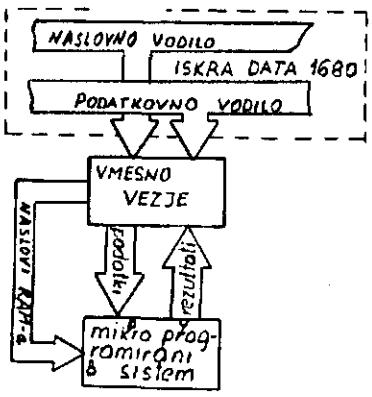
- R postavitev mikroprogramiranega mikroprocesorja v začetno stanje
- Vx omogoča vpis mikro ukaza na naslov x v mikroprogramskem pomnilniku
- Sx sproži izvajanje mikroprograma, ki ima začetni naslov na x
- Nx omogoča vpis operandov v registre z začetnim naslovom x



Slika 3



Slika 4



Slika 5

- l povzroči izpis vseh 16 registrov mikroprogramiranega mikroprocesorja

IZDELAVA MIKROPROGRAMOV

Mikroprogrami so sestavljeni iz mikro ukazov. Format mikro ukaza je naveden v tabeli II, kjer dajemo pomen posameznih skupin bitov.

biti v mikro ukazu	določajo operacije
1 - 9	izbira funkcije mikroprocesorjev 3 in 4
10 - 18	izbira funkcije mikroprocesorjev 1 in 2
19 - 22	naslavljanje A izhoda RAM-a v mikroprocesorju mikroprog.sist.
23 - 26	naslavljanje B izhoda RAM-a v mikroprocesorju

biti v mikro ukazu	določajo operacije
27 - 31	izbira vrste pomika
32 - 33	določanje prenosnega bita za aritmetične operacije
34 - 35	povezava mikroprocesorjev (2 x 8 bitov ali 16 bitov)
36 - 43	naslov za skok
44 - 48	izbira funkcije krmilne enote
50 - 55	izbira testa za pogojni skok
56 - 57	krmiljenje števca iteracij
58	zaustavljanje ure

Tabela II

V tabeli III je prikazan primer mikroprograma za množenje dveh šestnajstbitnih števil.

```
*V5 S MNOZENJE DVEH 16 BITNIH STEVIL
0000000111000000011000000011111000000000000000000100000000001
*V6
1001000111001000111000000101111000000000000000000100000000101
*V7
1000000111000000111000000101111000000000000000011010000011111
*V8
1000000111000000111000000101111000000000100001101101000011111
*V9
1000000111000000111000000100111000000011000011001000000000101
*VA
1000000011000000001000000100111000000001000011011010000011111
*VB
1000010011000010010000001100111001000011000011001000000000101
*VC
0111000100110000100000001111100000000000000011001000000000101
```

Tabela III

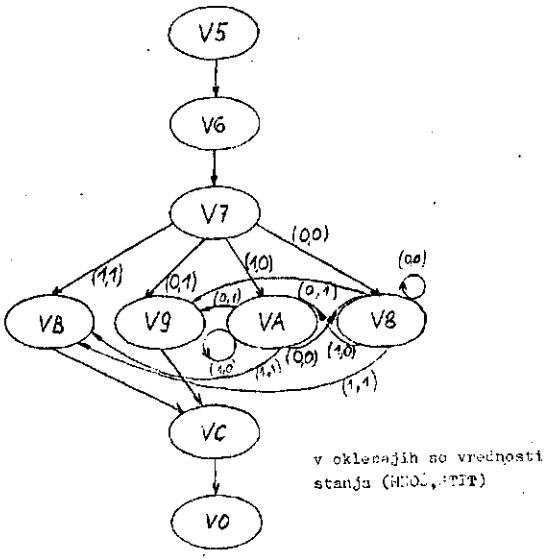
ZAKLJUČEK

V članku smo zapisali osnovne značilnosti bipolarnega mikroprocesorja, ki smo ga realizirali z namenom, da ga priključimo kot pomožni - periferni procesor k mikroračunalniku. Materialna oprema je narejena dovolj univerzalno, da lahko tak mikroprocesor uporabimo tudi kot samostojen procesor. Glede na namen takšnega procesorja je potrebno razviti ustrezne mikroprograme.

Ker je tak način podajanja mikroprograma nepregleden, smo izdelali zapis (tabela IV), ki je v pomoč programerju. Takšen zapis skupaj z diagramom poteka (slika 6) nazorno prikazuje algoritem mikroprograma. Preslikava s simboličnega zapisa v binarni zapis je enolična in jo je mogoče enostavno opraviti ročno ali z računalnikom.

```
V5-      RAM I → Qreg , 0 → ŠTIT ,
V6-     (0 + RAM 2)/2 , Qreg /2 ,
V7-     TEST (MNOŽ,ŠTIT), ŠTIT+1 ,
V8-(RAM 2 + 0 → RAM 2)/2 , Qreg/2 , ŠTIT+1, TEST(MNOŽ,ŠTIT),
V9-(RAM 2 + 0 → RAM 2)/2, skok na VC,
VA-(RAM 2 + RAM I → RAM 2)/2, Qreg/2 ,ŠTIT+1 , TEST(MNOŽ,ŠTIT),
VB-(RAM 2 - RAM I → RAM 2)/2 , skok na VC
VC- Qreg → RAM I , skok na 0,
```

Tabela IV



Slika 6

LITERATURA

1. M6800 MICROPROCESSOR APPLICATIN MANUAL

2. IDM 2900 FAMILY MICROPROCESSOR DATABOOK (NATIONAL SEMICONDUCTOR)

3. THE TTL DATA BOOK FOR DESIGN ENGINEERS (TEXAS INSTRUMENTS)

4. LOW POWER SCHOTTKY DATA BOOK (FAIRCHAILD)

5. SUPPLEMENT TO THE TTL DATA BOOK FOR DESIGN ENGINEERS (TEXAS INSTRUMENTS)

6. NIKITAS A.ALEXANDRIDIS-BIT-SLICED MICRO-PROCESSOR ARCHITECTURE , Computer VI. 1978

7. NAVODILA ZA UPORABO IN PRIPRAVO SISTEMA ISKRA DATA 1680

8. M. Družovec: PERIFERNA ARITMETIČNA ENOTA - predlog komunikacije in krmiljenja, ISEMEC 1980, str. C/1 - 1

9. M. Gerkeš: PROBLEMSKO ORIENTIRANE STRUKTURE PROCESNIH ENOT, raz. nal. VTŠ 1980

10. V. Žumer: RAČUNALNIK, VTŠ 1980

MIKROPROCESORSKI IN PARALELNI SISTEMI

A. PRAPROTNIK

UDK: 681.3.012

ISKRA RAČUNALNIKI, LJUBLJANA

Članek opisuje osnove multiprocesorskih struktur. Poudarek je predvsem na arhitekturi materialne opreme multiprocesorskih organizacij. Kratko so opisane tudi osnove multiprocesorskih operacijskih sistemov.

Microprocessor and Parallel Systems

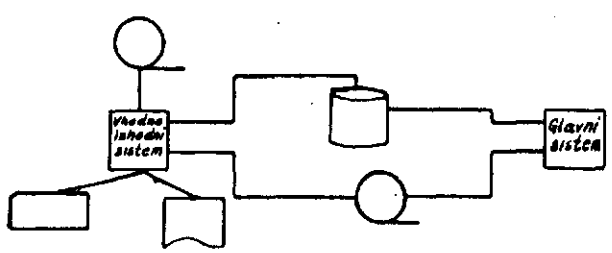
This article is an overwiew of multiprocessors systems; specially is pointed out an architecture of multiprocessor systems. Shortly are discribed also basis of multiprocessors.

UVOD

Na tržišču se pojavljajo in vedno bolj uveljavljajo multiprocesorski sistemi , ki se pred drugimi sistemi odlikujejo s svojo propustnostjo, prilagodljivostjo, zanesljivostjo in razpoložljivostjo. Te lastnosti dajejo motivačijo še večjemu razvoju teh sistemov.

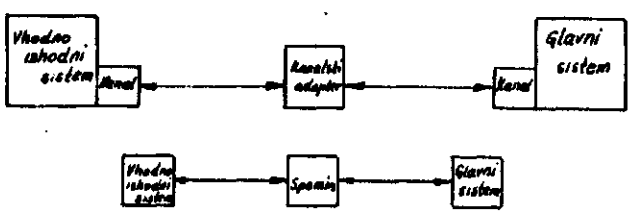
Zametek paralelnih sistemov tvorita enoprocesorska organizacija sistema z direktnim dostopom vhodno - izhodnih enot (I/O) v spomin (DMA) in organizacija sistema s kanali. Pri teh dveh arhitekturah se je del opravil, ki jih je opravljal procesor, prenesel na druge enote.

Pred obravnavanjem multiprocesorskih organizacij bi še omenil primer sistema dveh računalnikov,v katerem deluje en računalnik kot glavni procesor, drugi računalnik pa kot samostojni vhodno-izhodni procesor. Prvotna povezava med njima je bil magnetni trak, ki se je fizično prenašal, pozneje so sisteme medsebojno povezovali preko magnetno tračnih enot ali enot masovnega spomina (diskov) z mehanskimi in kasneje elektronskim stikalom. Na ta način smo dobili nek paralelni sistem, ki ga imenujemo ohlapno ali indirektno zvezani sistem (slika 1).
V vsakem sistemu deluje samostojni operacijski sistem, ki gleda na drug sistem kot na vhodno-izhodno enoto.



sl.1

Oba sistema lahko povežemo tudi preko posebnega kanalskega vmesnika ali preko skupnega spomina (slika 2).



sl.2

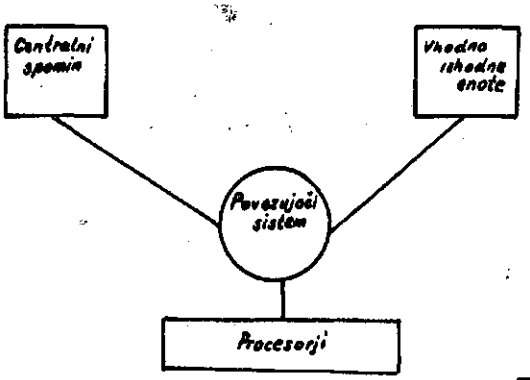
Na sliki 2, ki nam prikazuje takšno povezavo,takoj lahko vidimo, da vsak sistem obravnava drug sistem kot vhodno - izhodno enoto.Sam prenos pa zahteva sodelavo obeh sistemov hkrati. En sistem oddaja podatke , drugi pa jih istočasno sprejema. Vmesniki poskrbe za potrebne signale med sistemoma. Vsak sistem ima svoj operacijski sistem.

OSNOVE MULTIPROCESORJEV

Ameriški standard (ANSI) definira multiprocesor kot računalniški sistem, ki je sestavljen iz dveh ali več procesorskih enot, katere upravlja skupna kontrola (procesorji imajo skupni enojni operacijski sistem). Vendar v tej definiciji nista zajeta koncepta medsebojnega delovanja posameznih enot in skupne uporabe posameznih struktur, ki predstavljata bistvo multiprocesiranja.

Računalniški sistem mora imeti glede na materialno opremo možnost skupne uporabe centralnega spomina in možnost dosega vhodno - izhodnih enot ne glede na katerokoli razvrstitev procesorjev in spomina . (slika 3)

Važen je nivo medsebojnega delovanja sistemov. V resničnih multiprocesorskih sistemih mora sodelovanje med procesorji preiti na najnižji nivo, torej na nivo datotek, podatkovnih nizov in celo podatkovnih elementov.Glede na nivo obdelav pa mora obstojati možnost medsebojnega delovanja na nivoju kompletnih poslov podprogramov in posameznih postopkov.



sl.3

ORGANIZACIJA MATERIALNE OPREME

V osnovi obstajajo samo tri različne organizacije materialne opreme v multiprocesorskih sistemih.

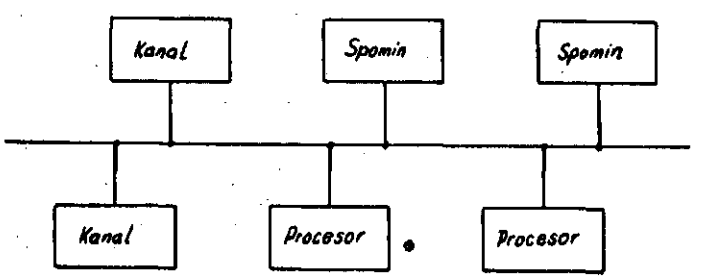
a)Časovno razdeljeno skupno vodilo (BUS)
b)Crossbarsko stikalo
c)Spomin z več vrati

Pri nekaterih sistemih pa je paralelnost dosežena z drugačnimi metodami, ki se jih bomo kasneje na hitro dotaknili :

- Nesimetrične ali nehomogene organizacije
- Vektorski procesorji in sistemi za računanje večkratnih vrednosti (polj)
- Pipeline procesorji (Pipeline processors)
- Zanesljivo tekoči sistemi (Fault tolerant systems)
- Asociativni procesorji

A. SISTEMI S ČASOVNO RAZDELJENIM VODILOM

Najpreprostejšo organizacijo za katerikoli sistem dobimo s postavitvijo vodila ter priključitvijo vseh posameznih enot na to vodilo (slika 4) Na ta način dobimo preprost multiprocesorski sistem.



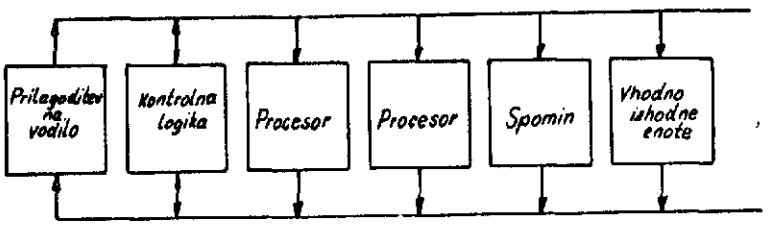
sl.4

Prenos preko pasivnega povezovalnega sistema popolnoma nadzorujejo oddajne in sprejemne enote. Enota, ki želi prenašati, mora najprej ugotoviti stanje vodila (status), naslov enote s katero želi kommunicirati, njeno zmožnost kommuniciranja (status) ter sporočiti nalogo te enote. Nato enota lahko začne prenos. Enota, s katero se kommunicira, pa mora spoznati na vodilu svoj naslov in odgovarjati na kontrolne signale.

V primeru, da takemu vodilu želimo dodati ali odvzeti funkcionalno enoto, ni potrebno napraviti v materialni opremi skoro nobene spremembe, ampak samo v programski opremi z določitvijo enot. Tak sistem je lahko cenen.

Te prednosti pa nastopajo na račun drugih omejitev. Največja omejitev je v zmogljivosti sistema, kajti obstaja samo eno vodilo za vse prenose. Pri izboljšavi pa se takoj poveča kompleksnost sistema.

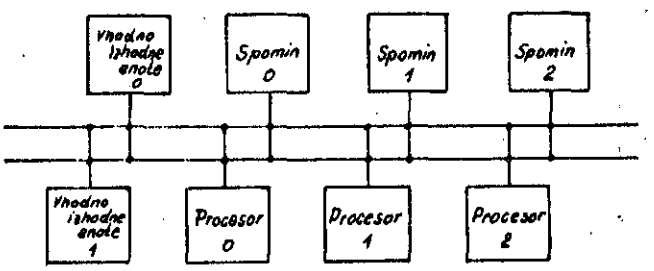
Prva stopnja izboljšav bi bila v dveh enosmernih vodilih. (slika 5)



sl.5

Pri tem se sama kompleksnost sistema ne poveča preveč, vendar tudi ne pridobimo s tem veliko. Že preprosti prenos normalno zahteva uporabo obeh vodil.

V naslednji stopnji izboljšanja dodamo dvosmerna vodila. (slika 6)



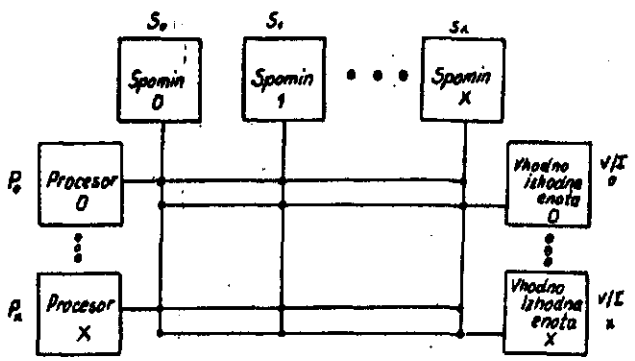
sl.6

S tem dosežemo večkratni hkraten prenos. V povezavah niso več pasivni elementi, ampak morajo biti inteligentni (logika, stikala) v vseh točkah, kjer so funkcionalne enote vezane na skupno vodilo.

B. SISTEMI S CROSSBARSKIM STIKALOM

V primeru, da povečujemo število vodil, pridemo do stanja, kjer ima vsaka spominska enota na razpolago svoje vodilo (slika 7). Povezovalni podsistem je "neblokirajoče Crossbarsko stikalo". Maksimalni prenos se lahko izvaja istočasno in je omejen s številom spominskih enot in ne s stikalom. Od tu izhaja tudi pridevnik neblokirajoč. Običajno ga izpuščamo.

Karakteristika sistemov s Crossbarskimi stikali je preprostost spajanja in podpiranja hkratnega prenosa v vse spominske enote ali iz njih. V samem stikalu pa je potrebna velika zmoznost materialne opreme. Vsaka točka priključitve mora dovoljevati paralelni prenos in obvladovati vrstni red pri večkratnih istočasnih zahtevah za dosego istega spominskega modula. S tem postane materialna oprema v stikalu kompleksna. Prednost je v tem, da pri dodajanju posameznih funkcionalnih enot do-

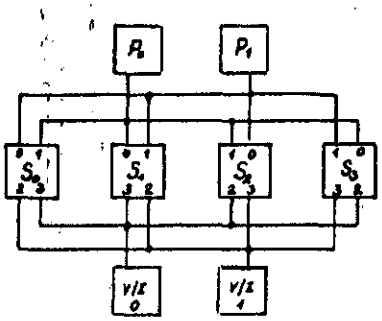


sl. 7

dajamo samo module priključitvenih točk. Obenem pa je potrebno spremeniti operacijski sistem, kar je običajno trd oreh.

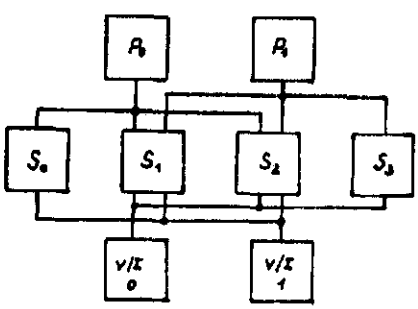
C. SISTEMI S SPOMINI Z VEČ VRATI

Nadzorno in krmilno logiko iz Crossbarskih stikal lahko preneseno na spominsko enoto. Na ta način dobimo spominske module z več vrati (multiport memory modules). (slika 8)



sl. 8

Konflikte pri doseganju spomina razrešujemo s tem, da določimo vsakemu vhodu v spominsko enoto določeno prioriteto. Na sl. 8 je to prikazano s številkami vrstnega reda vhoda v spominski modul. Vezavo lahko tudi nekoliko predrugačimo in dobimo nekatere vhode popolnoma namenjene samo nekaterim enotam. Označimo jih lahko kot "privatne" za določene procesorje ali vhodno - izhodne enote (slika 9). Sama ta organizacija ima prednost v povečanju varnosti pred neželjenim dosegom spomina.



sl. 9

Dovoljuje shranitev nujnih programov pri obnovitvi sistema, ki ga nekateri procesorji ne morejo doseči. V tem pa je lahko tudi slabost : v primeru okvare enega procesorja, drugi procesorji ne morejo dobiti informacije o stanju in nadzoru, katera se nahaja v spominu, ki ga je lahko dosegel procesor, ki ima napako.

PRIMERJAVE OSNOVNIH SISTEMSKIH ORGANIZACIJ

Oglejmo si primerjavo opisanih sistemskih organizacij. Upoštevali bomo naslednje faktorje, ki dajejo določene podatke o sistemu : ceno, prilagodljivost, možnost nadgradnje in propustnost sistema. Obenem bomo pregledali tudi nekatere slabosti.

A) SISTEM S ČASOVNO RAZDELJENIM SKUPNIM VODILOM

1) Najnižja cena materialne opreme.
2) Enostavnost; vodilo je lahko popolnoma pasivno.
3) Fizično je lahko spreminjati materialno opremo.
4) Kapaciteta sistema je omejena z zmožnostjo vodila. Ta lahko omeji celotno zmogljivost sistema.
5) Napaka na vodilu je lahko usodna v delovanju sistema.
6) Dodajanje nove funkcijske enote lahko zmanjša celotno zmogljivost sistema.
7) Efektivnost sistema je najnižja.
8) Te vrste organizacija je primerna samo za manjše sisteme.

B) SISTEMI S CROSSBARSKIMI STIKALI

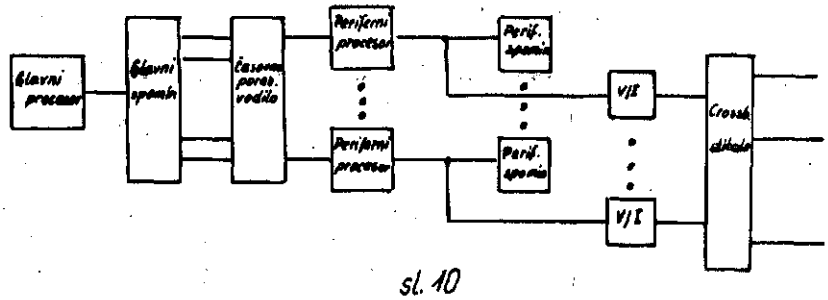
1) Sistem povezav je najbolj zahteven.
2) Funkcijske enote so lahko najcenejše in najbolj enostavne.
3) Te vrste organizacija se splača samo za multiprocesorske sisteme.
4) Omogoča največji možni prenos in največjo hitrost pri prenosu.
5) Dodajanje funkcijskih enot povečuje zmogljivost celotnega sistema.
6) Doseči je mogoče največjo efektivnost sistema.
7) Razširitev sistema je možno izvesti brez sprememb operacijskega sistema.
8) Razširitev sistema je omejena z matriko stikala. Te pa se da v določenih mejah modularno povečati.

C) SISTEMI S SPOMINI Z VEČ VRATI

1) Zahtevajo najdražji spomin. Ves nadzor in stikalna oprema sta v spominskih modulih.
2) Obstaja možnost velike prenosne hitrosti v celotnem sistemu.
3) Velikost in konfiguracija sta omejena s številom vhodov v spominske module. To število se določi ob zasnovi sistema.
4) Potrebno je veliko število kablov in konektorjev.

D) OSTALE ORGANIZACIJE
NESIMETRIČNI ALI NEHOMOGENI SISTEMI

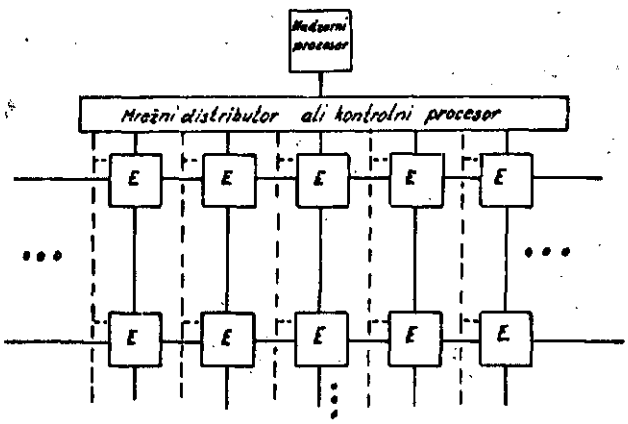
Te vrste organizacija uporablja takozvane periferne procesorje, ki komunicirajo na eni strani s spominskimi moduli, na drugi strani pa z vhodno - izhodnimi enotami (slika 10). Verjetno bo ta organizacija postala v prihodnosti še pogostejša, če ne standardna. Motivacija je v ceni, saj se kot periferni procesorji lahko uporabljajo mikroračunalniki.



sl. 10

VEKTORSKI SISTEMI IN SISTEMI ZA RAČUNANJE VEČKRATNIH VREDNOSTI - POLJ (array)

Običajno se vršijo opracije na vseh komponentah večkratnih vrednosti z medsebojnim vplivanjem vmesnih rezultatov. V primeru, da se izvaja te vrste proces na klasičnem računalniku, je potrebno pretvoriti naravno paralelni proces v vrsto posameznih stopenj na individualnih podatkovnih elementih. Torej je cilj pri vektorskem računalniku možnost hkratne obdelave identičnih operacij na različnih podatkovnih elementih.
Shematično je tak sistem prikazan matrično (slika 11). Vsak procesni element (E) je zase popoln sistem, ki vključuje tudi spomin. Medsebojno odvisnost med podatki in posameznimi procesnimi elementi vzpostavimo pri napolnitvi lokalnega spomina določenega elementa s podatki. Potrebne krmilne signale za vse sisteme generira in delegira kontrolni procesor.



sl. 11

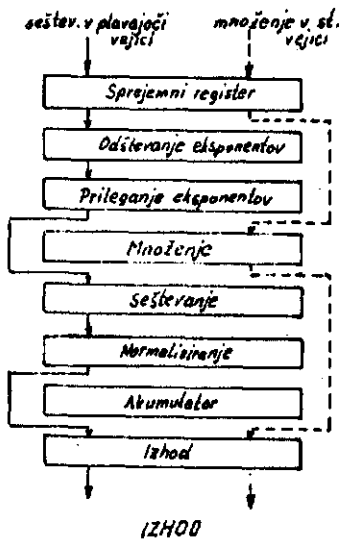
Sistemi, ki imajo samo eno kolono procesorskih elementov, se imenujejo vektorski procesor. Posebno ime "ortogonalni procesor' ima tudi sistem, ki ima eno kolono in eno vrstico procesnih elementov.

Te vrste sistem bi lahko uporabili tudi za normalne obdelave masovnih podatkov, čeprav se na prvi pogled zdi, da so namenjeni znanstvenim obdelavam.

PIPELINE SISTEMI

Pri teh vrstah sistemov se uporablja za paralelnost procesiranja druga tehnika, kot smo jo obravnavali do sedaj. Oglejmo si koncept teh sistemov, ki se ne sklada z našimi predpostavkami o multiprocesiranju.

V določeni liniji posamezne procesne enote izvršujejo določena opravila tako, da na koncu linije dobimo razultat (slika 12).



sl. 12

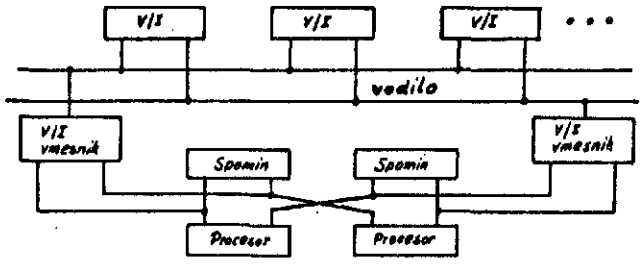
Prednost tega sistema je v tekočem izvajanju operacij ; operacijo za operacijo. Kot primer bi navedli seštevanje v plavajoči vejici.

Operacija	Čas trajanja
odštevanje eksponentov	40 nsek
poravnavanje	70 nsek
seštevanje	50 nsek
normaliziranje	80 nsek
skupaj	240 nsek

Skupno traja operacija seštevanja v plavajoči vejici 240 nsek. Števila pa lahko seštevamo in dobimo rezultate vsakih 80 nsek, kolikor traja najdaljša faza v tej operaciji.

ZANESLJIVO TEKOČI SISTEMI (Fault tolerant systems)

Vodilo pri teh sistemih je želja, da je na razpolago za delovanje vsaj en kompleten sistem. Ta želja prekrije vse ostale kriterije. Realizacija pa je v multiprocesorskem sistemu, ki je povezan na poseben način.
(slika 13)



sl. 13

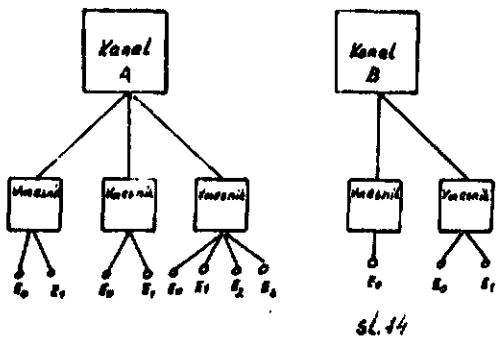
Te vrste procesorji morajo imeti robustno materialno opremo (uporabljajo se v telekomunikacijah in zračnih plovilih), ki mora biti obenem podprtana z zanesljivo programsko opremo, ki zmore hitre prekonfiguracije sistema.

ASOCIATIVNI PROCESORJI

Po zamisli in izvedbi je asociativno procesiranje resnično paralelna operacija. Osnovni koncept je iskanje lokacij v nekem polju, ki vsebuje enake podatke, kot je podatek v osnovnem primerjalnem registru. Polje, v katerem iščemo te lokacije, je spominsko polje in ga tudi imenujemo asociativni spomin ali vsebinsko naslovljeni spomin (CAM). Iščemo namreč naslove po vsebini. Uporabnost tega sistema zvečamo še z maskovnimi registri, ki jih dodamo k osnovnemu primerjalnemu registru. Z njim določimo bite, ki jih iščemo. Običajno je te vrste spomin samo del nekega večjega sistema.

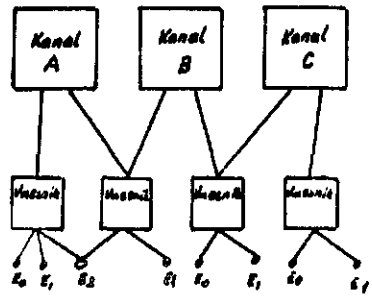
POVEZAVA IN SKUPNA UPORABA VODIL

Do sedaj smo obravnavali v glavnem povezave spominskih enot s procesorji in kanali in njihovo hkratno delovanje. V multiprocesorskih sistemih sodelujejo tudi vhodno - izhodne enote. Možnost povezav le-teh v sistemu je veliko. Poglejmo si eno na sliki 14.

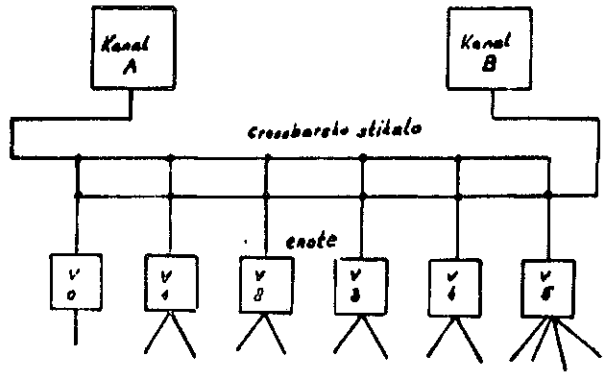


sl. 14

V tej povezavi obstaja k vsaki enoti samo ena pot. Če je kanal zaseden, se do določene vhodno-izhodne enote ne more priti. Temu se izognemo do neke mere s povezavo po sliki 15. Z uporabo crossbarskega stikala v povezavi pa prilagodljivost sistema še povečamo. (slika 16)



sl. 15



sl. 16

POVEZAVA VEČJEGA ŠTEVILA MINIPROCESORSKIH IN MIKROPROCESORSKIH SISTEMOV

To področje postaja vedno bolj zanimivo in se tudi obkljuje vedno bolj in bolj, predvsem zaradi nizke cene mikroprocesorjev.

Sistemi z enojnim vodilom bi bili zelo primerni in enostavni za povezavo v multiprocesorski sistem. Nastopijo pa težave že v naslavljanju istega spominskega modula iz dveh procesorjev.

Procejšnji razvoj gre v smeri medprocesne in medprocesorske povezave. Potrebno je najti zadovoljiv odgovor na vprašanje : "Na kakšen način zahtevati in nadzorovati izvajanje odvisnih poslov takrat, kadar je dodeljen tak posel procesorju, ki je različen od tistega, ki je izvajanje zahteval?". Drug zanimiv problem zadeva materialno in programsko opremo. Avtomatično je potrebno odkriti napako v materialni ali programski opremi in najti izhod iz te napake.

Končni cilj bi bila povezava velikega števila mikroprocesorjev v sistem, ki bi zmogel procesirati posle, ki jih zmorejo veliki računalniki. Potrebno bo dodati procej sistemske kontrole, ki bi jo izvedli spet s cenenimi mikroprocesorji. Omejitev pa je spet v časovno razdelje - nem vodilu.

PROGRAMSKA OPREMA

Lahko bi rekli, da v programski opremi, ki je potrebna za multiprocesorske sisteme, ni velike razlike od one opre -

me, ki jo zahtevajo veliki sistemi pri multiprogramiranju. Bolj nam postane jasno, če pogledamo kakšne funkcije naj opravlja operacijski sistem v multiprocesorkih sistemih :

a) Dodeljevanje naprav in njihovo upravljanje
b) Zavarovanje tabel in podatkov
c) Varovanje pred prekinitvijo delovanja sistema
d) Obravnavati nenormalne prekinitve
e) Uravnoteženje vhoda - izhoda
f) Komuniciranje med procesorji
g) Uravnoteženje obremenitev posameznih procesorjev
h) Rekonfiguracija

Od naštetih funkcij samo zadnje tri pripadajo multiprocesorkim sistemom. Efektivnost operacijskega sistema postane zelo važna v multiprocesorskem sistemu. Slabo delovanje operacijskega sistema lahko izniči vse prednosti multiprocesorskega sistema.

ORGANIZACIJA MULTIPROCESORSKEGA SISTEMA

Obstajajo tri osnovne organizacije v razvoju operacijskega sistema za multiprocesorje.
- nadzornemu sistemu (monitorju) sledilni način
- nazorni sistem - monitor v vsakem procesorju
- enakovredno obravnavanje vseh procesorjev

V večini multiprocesorjev je prva organizacija operacijskega sistema najpogostnejša. Ta tip organizacije je najenostavnejši in jo je mogoče izvesti iz operacijskega sistema enojnih procesorjev, ki imajo zmožnost multiprogramiranja. Sistem je običajno neefektiven v nadzorovanju in uporabi vse opreme.

I. Operacijski sistem, ki dela v monitorju/sledilnem načinu ima naslednje značilnosti :

- monitor je vedno v istem procesorju. Če drug procesor, ki dela pod nadzorom monitorja , želi strežbo, mora to zahtevati in počakati, da postane monitor prost. Ker je samo en procesor monitor,se na ta način izognemo konfliktom v tabelah in nadzorovanju tabel.

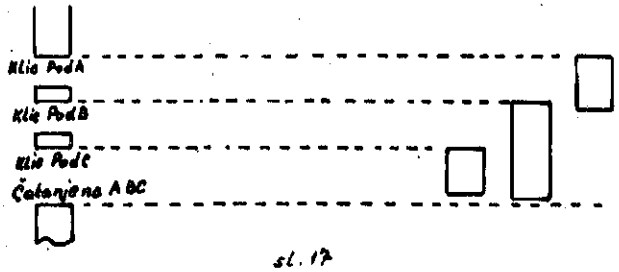
-Celotni sistem se podre ali vsaj zahteva intervencijo operaterja, če pride do napake v monitorju

- Celotni sistem je precej neprilagodljiv.

- Prazni tek procesorjev, ki čakajo na strežbo monitorja, lahko postane onemben vreden.,

- Ta tip operacijskega sistema zahteva relativno prepro - sto materialno in programsko opremo.

- Operacijski sistem te vrste je najbolj efektiven v posebnih aplikacijah, kjer je obremenitev sistema poznana, in v primerih, da imajo ostali procesorji manjše zmožnosti kot procesorji, ki imajo nadzor.



sl. 17

Od ostalih dveh organizacij operacijskega sistema ni jasno glede zmogljivosti sistema, katera je najboljša. Izgleda, da sta obe organizaciji boljši od prve.

II. Pri drugem tipu organizacije se nahaja v vsakem procesorju kopija nadzornega sistema (monitorja) in s tem dobi organizacija naslednje karakteristike :

-vsak procesor obdeluje sam svoje potrebe .

- Vsak procesor ima svoje privatne tabele. Nekatere tabele morajo biti skupne celotnemu sistemu , kar sproži težave pri nadzoru dosega teh tabel.

- Sistem ni tako občutljiv za napake kot prejšnji tip, vendar je ponovni zagon procesorja , ki ima napako,vprašljiv.

- Vsak procesor ima lastne vhodno - izhodne enote,itd. Sprememba povezave vhodno - izhodnih enot zahteva večji poseg.

III. Najtežji tip organizacije je enakovredno obravnavanje vseh procesorjev oziroma ostalih enot. Rezultati, ki jih daje takšna organizacija,pa odtehtajo težave te organizacije.

Značilnosti te organizacije bi bile :

- monitor se seli iz procesorja v procesor; nekateri procesorji lahko izvajajo monitorne programe istočasno.

- Operacijski sistem lahko porazdeli obremenitev sistema .

- Konflikti v zahtevah po strežbi se lahko rešujejo na osnovi prioritete, ki je lahko postavljena statično ali dinamično.

- Nadzorni programi morajo biti ponovno vzpostavljeni , saj lahko različni procesorji izvajajo iste programe v istem času.

- Pri uporabi tabel prihaja do konfliktov in s tem do zakasnitev, vendar se temu ne da izogniti. Moramo pa jih popolnoma nadzorovati.

Prednosti te organizacije bi torej bile :

a) sistem lahko degradiramo
b) sistem daje neko popolnost
c) sistem najbolj efektivno izrabi enote, ki so mu na razpolago.

PROGRAMIRANJE

Potencialna moč materialne opreme multiprocesorjev oz. paralelnih procesorjev se lahko razgubi, če jo aplikacijski programi ne izrabljajo. Do sedaj je bilo napravljenih precej študij na razvoju jezikov, ki bi specifično podpirali paralelne operacije.

Najbolj enostavna in splošna je situacija,pri kateri program, ki se izvaja,vsebuje različne neodvisne podprograme. Z enim procesorjem se ti programi izvajajo sorijsko,pri nekaterih multiprogramskih sistemih pa lahko tudi asinhrono. Čas obdelave je neglede na vrstni red obdelave enak vsoti časov izvajanja podprogramov. Če so na razpolago še drugi procesorji,se lahko podprogrami izvajajo paralelno. Čas izvajanja bo krajši oz. enak času izvajanja najdaljšega programa (slika 17).

Sistem mora imeti pri paralelnem izvajanju posebne stavke, ki označujejo neodvisne podprograme, označbe, kdaj se lahko prične izvajanje,in označbe,kje so potrebni rezultati za nadaljevanje glavnega programa.

Literatura :

Enslov : Multiprocessors and other parallel systems
Georgia Institute of Technology Atlanta USA

Brinch Hansen : Operating System Principles,
Prentice - Hall, Englewood Cliffs.

Hoare, C.A.R. Monitors :
An operating system structuring concept.

Hoare,C.A.R. Towards a theory of parallel programming
Academic Press, New York 1972.

SLIJED TOKA PROGRAMA ZA MALA RAČUNALA

DRAGAN GAMBERGER

UDK: 681.3 : 519.682

Ukazuje se na mogućnost razvoja sistemskog programa za slijed toka korisničkog programa (trace program) za razna mala računala koji omogućuje da se bez dodatnih sklopova na istom sistemu simulira rad korisničkog programa u produženom vremenu uz dobivanje potrebnih podataka o njegovom toku i rezultatima na ispisnoj jedinici. Za mala računala prikladno je u trace program uvesti i mogućnost simulacije ulazno-izlaznih komunikacija kao i odredjivanje stvarnog vremena izvodjenja. Oblik ispisa takvog trace programa prikazan je za mikroračunalo Intel8080.

TRACE PROGRAM FOR SMALL COMPUTERS: A possibility of system program development for various small computers which would enable off-line simulation of the user programs is described. The presented method uses the existing prototype hardware and displays the information of the application program flow on a terminal. For small computers it is convenient to include in the trace program the possibility for input-output simulation and determination of the real execution time. Terminal listing for such a trace program for Intel 8080 microcomputer is presented.

UVOD

Trace program je sistemski program za mala računala koji izvodi korisnički program instrukciju po instrukciju uz istovremeno prikazivanje na ispisnoj jedinici bitnih promjena u računalu. Pri tome za mala računala poteškoću predstavlja izvodjenje programa instrukciju po instrukciju na korisničkom računalu bez posebnih dodatnih sklopova. Problem je rješen primjenom simulacije instrukcija a novost je u tome što se za većinu instrukcija koristi simulacija promjenom mjesta izvodjenja dok se mali broj instrukcija simulira programski. Tim je proces simulacije instrukcije znatno pojednostavljen što omogućuje da se u trace program uključe i druge funkcije, kao što je simulacija ulazno-izlaznih komunikacija i odredjivanje stvarnog vremena izvodjenja, te da time on postane cjelovito pomoćno sredstvo za razvoj korisničke programske podrške.

NAČIN SIMULACIJE PROGRAMA

Trace program simulira korisnički program instrukciju po instrukciju počevši sa zadanom početnom adresom jednakim redoslijedom kao i pri izvodjenju u stvarnom vremenu. Simulacija instrukcija koristi se zato jer općenito nije moguće izvoditi korisnički program koji se nalazi bilo u ROM-u ili RAM-u instrukciju po instrukciju bez ugradnje posebnih sklopova. Instrukcija korisničkog programa simulira se tako da se prebaci u odredjeni dio RAM-a sistemskog programa, iza nje postavi instrukcija bezuslovnog skoka koja kontrolu vraća u sistemski program a zatim ovdje stvarno izvodi. Ovim načinom se medjutim ne mogu simulirati instrukcije čije izvodjenje ovisi o mjestu na kojem se nalaze kao i instrukcije koje mijenjaju sadržaj programskog brojila ili bi na bilo koji drugi način, recimo zaustavljanjem rada računala ili omogućavanjem prekida programa, mogle učiniti da se iza instrukcije koja se simulira ne izvede instrukcija skoka koja kontrolu vraća sistemskom programu. Zbog toga je unaprijed potrebno odrediti tip instrukcije koja se simulira te u slučaju da se radi o instrukciji koja ne dozvoljava ovakav način simulacije, potrebno je preći na njemu programsku simulaciju koja će u korisničko stanje računala unijeti iste promjene kao da je instrukcija stvarno izvršena. Na programsku simulaciju može se preći i kod instrukcija kod kojih je normalno moguća simulacija izvodjenjem na drugom mjestu ali samo kada zato postoje posebni razlozi, na primjer za simulaciju ulazno-izlaznih instrukcija, jer je ovakva simulacija složenija i zahtijeva posebni programski dio za svaku instrukciju. Diagram toka prikazuje osnovne funkcije trace programa. Pod korisničkim stanjem

računala smatra se onaj sadržaj registara i memorije te stanje flagova koje bi računalo imalo u danom trenutku da se korisnički program normalno izvodi. Pri izvodjenju u produženom vremenu korisničko stanje CPU-a pamti se u posebnom dijelu sistemskog RAM-a.

OBLIK ISPISA

Trace program svaku instrukciju korisničkog programa prikazuje posebnim retkom na jedinici za ispis. Na početku se prvo ispisuje adresa instrukcije a zatim njen mnemonički oblik. Po slijedu adresa instrukcija mogu se pratiti promjene u korisničkom programskom brojilu dok mnemonički oblik instrukcije znatno olakšava razumjevanje i snalaženje u programu.

U nastavku istog retka ispisuju se promjene u sadržajima registara i stanja flagova koje je izazvala instrukcija, kao i promjene u sadržaju memorijskih lokacija na koje utječe instrukcija. Pri tome se ispisuje adresa memorijske lokacije i njen sadržaj koji se dobiva njenim stvarnim čitanjem nakon što je izvršena simulacija instrukcije. Time se lako uočavaju nelogičnosti rada programa koje nastaju zbog neispravnog rada memorije ili zbog pokušaja upisa sadržaja u ROM ili u nepostojeću memoriju. Na kraju retka ispisuje se ukupno potrebno stvarno vrijeme za izvršenje programa počevši od mjesta početka simulacije koje se izračunava tako da se za svaku instrukciju pribroji njeno vrijeme izvodjenja. Korisnik može signalom sa tastature privremeno zaustaviti simulaciju radi analize rezultata, i nakon toga ponovo je pokrenuti. Nekim drugim signalom, kada se trace program koristi kao dio monitora, kontrola sistemskog programa se može prenijeti u komandni mod monitora i uz njegovu pomoć ispitati ili promjeniti korisničko stanje računala.

SIMULACIJA ULAZNO-IZLAZNIH NAREDBI

Pri simulaciji ulazno-izlaznih naredbi se kod nailaska na izlaznu naredbu podatak stvarno ne prenosi na vanjsku jedinicu već samo posebno ispisuje iza mnemoničkog oblika instrukcije. Za ulazne instrukcije se ispis prekida na tom mjestu i čeka dok korisnik preko tastature ne unese podatak koji će se shvatiti kao da je stigao od vanjske jedinice. Simulacija ulazno-izlaznih instrukcija prikladna je za ispitivanje programa kada ne raspolažemo sa vanjskim jedinicama a obavezno ju je koristiti pri simulaciji programa koji komuniciraju sa vanjskim jedinicama koje zahtjevaju posluživanje u stvarnom vremenu kao i programa koji rade sa jedinicom za ispis i tastaturom sa kojima radi sistemski program.

UPRAVLJANJE RADOM TRACE PROGRAMA

Na početku rada trace programa korisnik preko tastature odredjuje početnu adresu svog programa i stanje CPU-a prije početka izvršavanja programa, što mu omogućuje ispitivanje programa za razne početne uvjete i praktično pokretanje simulacije od bilo koje druge točke sa pretpostavljenim dostignutim stanjem računala. U toku početne konverzacije korisnik bira i uvjete simulacije kao što su izvodjenje ili simulacija ulazno-izlaznih instrukcija, brzina simulacije i druge. Nakon toga vrši se ispis početnog korisničkog stanja CPU-a i prelazi na simulaciju pojedinih instrukcija programa.

SIMULACIJA PREKIDA PROGRAMA

Trace program pretpostavlja da je pri početku korisničkog programa zabranjeno prihvaćanje prekida programa. Ukoliko se u toku izvodjenja naidje na instrukciju koja omogućava prekid programa ona se ne izvodi ali se njena pojava pamti pa se svaki put, do eventualne pojave instrukcije koja zabranjuje prihvaćanje prekida programa, na kraju simulacije pojedine instrukcije kratko omogući prihvat prekida programa. Ukoliko je u tom trenutku zahtjev za prekid prisutan, doći će do njegova posluživanja. Ako se želi i nakon prekida programa nastaviti sa simulacijom, potrebno je na mjestu prve instrukcije korisničkog programa koji poslužuje prekid programa, dodati instrukciju skoka u potprogram sa adresom posebnog dijela trace programa kako bi se kontrola vratila sistemskom programu.

OBLIK ISPISA ZA INTEL 8080

Table prikazuju početni oblik odredjivanja funkcija trace programa i ispis kratkog programa za Intel 8080, pri čemu je simuliran isti program ali je na tabeli 1 prikazan slučaj sa izvodjenjem ulazno-izlaznih instrukcija, bez stanja čekanja pri prenosu podataka izmedju CPU-a i memorije te sa prihvatom prekida programa dok tabela 2 prikazuje simulaciju ulazno-izlaznih instrukcija, pristup memoriji sa po jednim stanjem čekanja te promjenjenim početnim stanjem CPU-a.

Posebne karakteristike računala Intel 8080 je da ulazno-izlazna komunikacija može ići i preko instrukcija koje adresiraju memoriju te da se umjesto potrebnog vremena za izvodjenje instrukcija broje potrebna stanja CPU-a.

ZAKLJUČAK

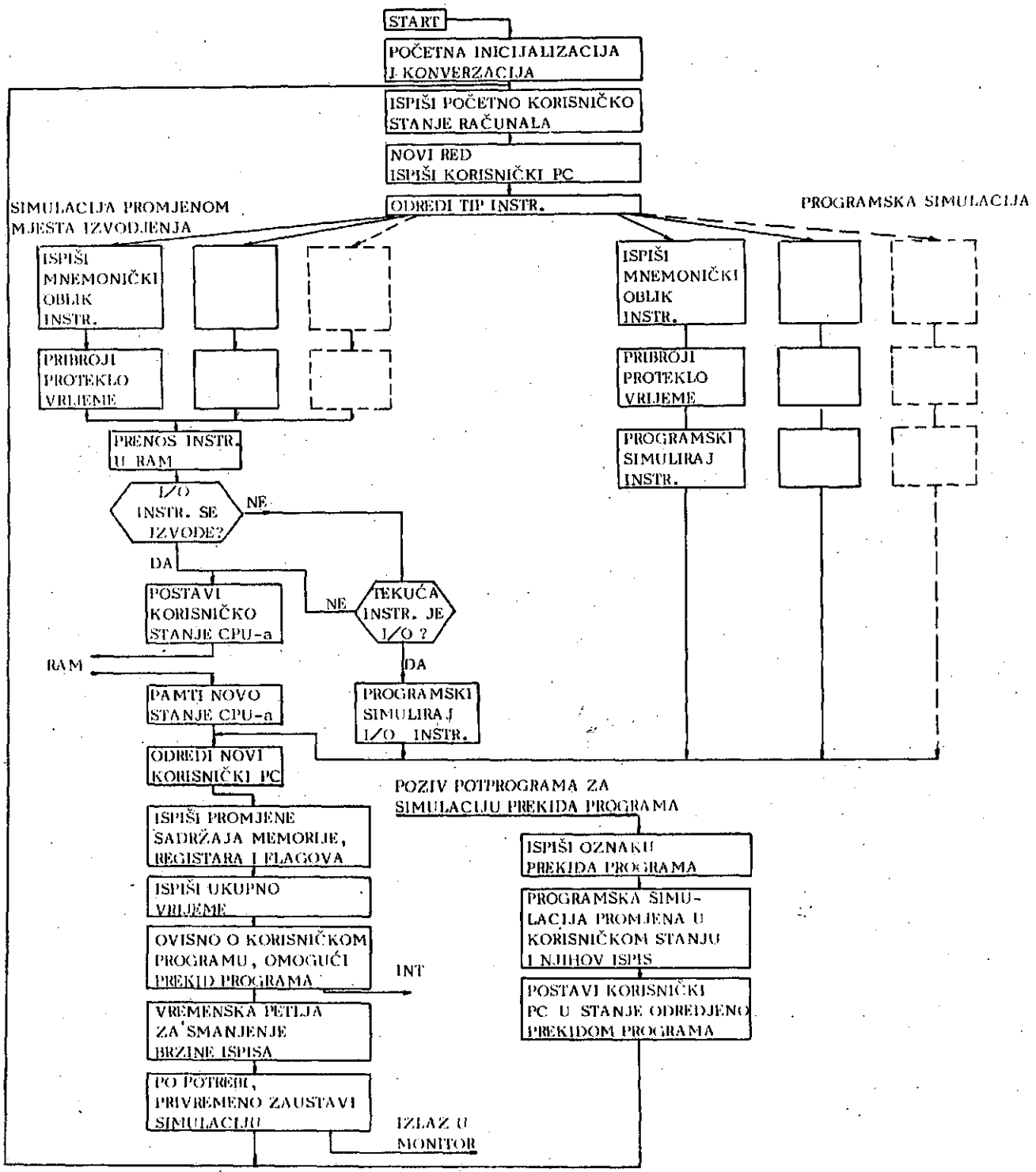
Simulator programa je relativno velik i složen sistemski program (za Intel 8080. on zauzima 3,5 K ROM-a i 256 by-

ta RAM-a) koji je potrebno posebno realizirati za svaku vrstu računala ali se njegova primjena ipak isplati jer omogućuje, pružajući uvid u tok i posljedice odvijanja korisničkog programa, znatno skraćenje vremena razvoja programske podrške. Ovo je od posebne važnosti jer je cijena razvoja programa sve veći dio cijene izgradnje računalskih sistema.

REFERENCE

1. Breakpoint and Single Step a New Approach, SGS Ates Planar News, p. 3-5,
2. Encyclopedia of Computer Science, Petrocelli, Charter, New York 1976, 1427-1428.

DIJAGRAM TOKA



```
PC=3000
APSW=3000
BC=3000
DE=3000
HL=3000
SP=3700
SIMULATION SPEED (0-2) 0
EXECUTE I/O INSTR. (Y,N) Y
NUMBER OF WAIT STATES PER 4E4. ACCESS (0-9) 0


PC=3000 A=00 C=00 B=00 E=00 D=00 L=00 H=00 SP=0700 Cy=0 Z=0
3000 CALL 3010H              4E4 36FE=03,00 SP=36FE
3010 XTHL                    4E4 36FE=00,00 L=00 H=00
3011 PUSH D                  4E4 36FC=00,00 SP=36FC
3012 MOV E,A                 E=00
3013 INR A                   A=01
3014 MOV D,A                 D=FF
3015 INR A                   A=00
3016 XCHG                    E=00 D=00 L=00 H=FF
3017 CALL 3A89H              4E4 36FA=1A,30 SP=36FA
3A89C INR A                  A=01
3A9D MOV A,A                 A=97
3A9E ANI 02H                 A=02
3AA0 JZ  3A89DH
3AA3 DCR A                   A=00
3AA4 MOV A,A                 A=00
3AA5 RET                     SP=36FC
3018 PUSH PSW                4E4 36FA=02,30 SP=36FA
3019 INR A                   A=01
301C MOV A,A                 A=95
301D RAR                     A=4A Cy=1
301E JNC 3018H
3021 POP PSW                 A=30 SP=36FC Cy=0
3022 DCR A                   A=00
3023 MOV A,A                 4E4 FF20=30
3024 XCHG                    E=00 D=FF L=05 H=30
3025 POP D                   E=30 D=30 SP=36FE
3026 XTHL                    4E4 36FE=05,30 L=00 H=30
3027 RET                     SP=3700
3005 CMA                     A=CF
3006 ANI 07H                 A=07
3008 STA 0FF00H              4E4 FF00=FF
300B EI
300C NOP
     RST 1                   4E4 36FE=00,30 SP=36FE
2100 LXI H,2130H
2103 RET                     SP=3700
3000 JMP 3000H
```

Tabela 1

```
PC=3000
APSW=1122
BC=3344
DE=5566
HL=7700
SP=3200
SIMULATION SPEED (0-2) 1
EXECUTE I/O INSTR. (Y,N) N
4E4DRY MAPPED I/O ON 0FFXXH (Y,N) Y
NUMBER OF WAIT STATES PER 4E4. ACCESS (0-9) 1

PC=3000 A=11 C=44 B=33 E=66 D=55 L=33 H=77 SP=3200 Cy=0 Z=0
3000 CALL 3010H              4E4 31FE=03,30 SP=31FE
3010 XTHL                    4E4 31FE=33,77 L=03 H=33
3011 PUSH D                  4E4 31FC=66,55 SP=31FC
3012 MOV E,A                 E=00
3013 INR A                   L=04
3014 MOV D,A                 D=FF
3015 INR A                   L=05
3016 XCHG                    E=05 D=30 L=00 H=FF
3017 CALL 3A89H              4E4 31FA=1A,30 SP=31FA
A89C INR A                   L=21
A89D MOV A,A       IN FF21=97 A=97
A89E ANI 02H                 A=02
A8A0 JZ  3A89DH
A8A3 DCR A                   L=20
A8A4 MOV A,A       IN FF20=30 A=30
A8A5 RET                     SP=31FC
3018 PUSH PSW                4E4 31FA=02,30 SP=31FA
3019 INR A                   L=21
301C MOV A,A       IN FF21=95 A=95
301D RAR                     A=4A Cy=1
301E JNC 3018H
3021 POP PSW                 A=30 SP=31FC Cy=0
3022 DCR A                   L=20
3023 MOV A,A       OUT FF20=30
3024 XCHG                    E=00 D=FF L=05 H=30
3025 POP D                   E=66 D=55 SP=31FE
3026 XTHL                    4E4 31FE=05,30 L=66 H=77
3027 RET                     SP=3300
3005 CMA                     A=CF
3006 ANI 07H                 A=07
3008 STA 0FF00H    OUT FF00=07
300B EI
300C NOP
300D JMP 3003H
```

Tabela 2

RAZVOJ DIGITALNEGA TIPALA IN MIKRORAČUNALNIŠKEGA KONTROLNEGA SISTEMA ZA OBLOČNO VARJENJE

S. PREŠERN,
I. OZIMEK,
M. ŠPEGEL

ODSEK ZA RAČUNALNIŠTVO IN INFORMATIKO,
INSTITUT JOŽEF STEFAN, LJUBLJANA

UDK: 681.3 : 007.52

Članek opisuje razvoj senzorskega sistema, ki omogoča robotu za obločno varjenje zelo natančno delo in prilagodljivost spremenljivemu okolju. Razvili smo digitalni senzor položaja tipalnega trna z dvema prostostnima stopnjama. Kontrolni sistem je sestavljen iz mikroračunalnika Z 80 s 4 K byt i bralnega pomnilnika (ROM) in 16 K byt i dinamičnega pomnilnika z naključnim dostopom (RAM). Senzorski sistem za pozicioniranje je opremljen s povratno zanko. Digitalno tipalo in spremljajoč mikroračunalniški kontrolni sistem je splošen in ga lahko uporabimo kot senzorski sistem za različne robote. Posebno pozornost pa smo posvetili uporabi tega sistema pri obločnem varjenju.

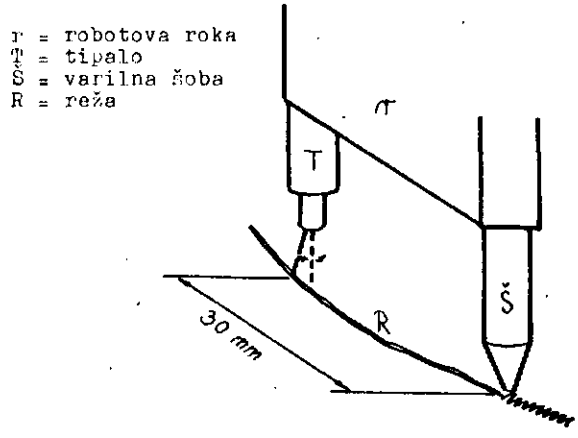
DEVELOPMENT OF A DIGITAL SENSOR AND MICROPROCESSOR CONTROL SYSTEM FOR ARC WELDING

The article describes the development of a sensing and control system for increasing capabilities of robots for high precision work and adaptivity to a changing environment conditions. We developed a digital tactile sensing system with two degrees of freedom. The sensing system is controled by the microcomputer Z 80 microprocessor with 4 K byte of read only memory (ROM) and 16 K byts of dynamic random access memory (RAM), and is equipped with feedback positioning control. This digital tactile sensing system and the corresponding micro computer control system is general and suitable for different robots. Special attention has been given to application of this sensor for seam tracking by arc welding robots.

1. UVOD

Za uspešno robotizacijo proizvodnih postopkov so bistvenega pomena tipala. S pomočjo tipala dobi robot potrebno informacijo o svoji neposredni okolici. V mikroprocesorju se ta informacija obdela in s pomočjo kontrolnega sistema omogoča inteligentno vodenje robotovih akcij. Roboti za varjenje zahtevnejših konstrukcij morajo imeti vpogled v stanje okolice, saj morajo poznati trajektorijo varjenja z natančnostjo tudi do milimetra. Le s primerno konstruiranim tipalom in z mikroračunalniškim kontrolnim sistemom, opremljenim s povratno zanko za pozicioniranje, lahko doseže robot tako prilagodljivost okolju in zagotovi potrebno preciznost varjenja, da lahko pri tem delu nadomesti človeka.

Robot tipa režo zvara med procesom varjenja. Na ta način ne izgubimo dodatnega časa, ki bi ga robot potreboval, če bi pred vsakim varjenjem najprej otipal režo zvara in ne potrebujemo dodatnega spomina kamor bi morali to informacijo shraniti. Tipalo je togo vpeto približno 3 cm pred varilno šobo (Slika 1). Varilna šoba s tipalom se glede na varjenec giblje s konstantno hitrostjo v. Trn tipala potuje po reži zvara in se s tem odklanja v smer reže (Slika 1). Ta odklon merimo z digitalnimi senzorji položaja trna, ki so v ohišju tipala. Ker je tirnica reže dvodimenzionalna, potrebujemo senzor položaja z dvema prostostnima stopnjama. Maksimalni odklon trna v vsako smer je -10 mm do +10 mm. Če upoštevamo, da je razdalja med tipalom in šobo 30 mm, vidimo, da to tipalo omogoča varjenje v prostorskem kotu 18 stopinj.



Slika 1: Princip delovanja tipala

2. KONSTRUKCIJA OPTIČNIH SENZORJEV

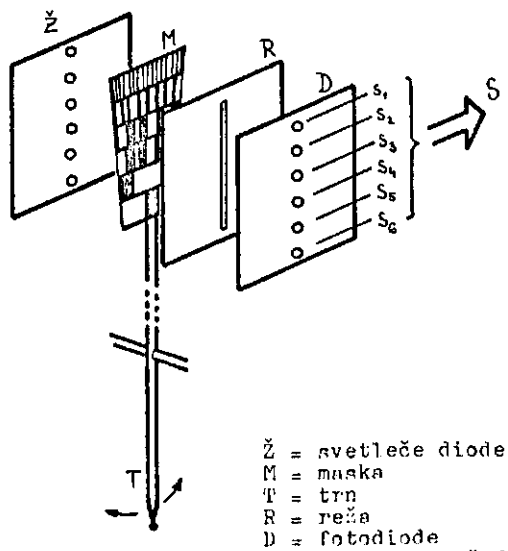
Pri konstrukciji senzorjev položaja tipalnega trna imamo na razpolago:

- potenciometrične senzorje (strain gauge, potenciometer)
- indukcijske senzorje
- optične senzorje.

Ker želimo, da bi bilo delovanje senzorja či

manj občutljivo tako na mehansko drsenje, ki se pojavlja pri potenciometrih in hkrati povzroča dodatno silo trenja pri premikanju trna, kot tudi na elektromagnetne motnje, ki nastajajo zaradi močnih sunkov toka (do 200 A) med varjenjem, smo se odločili za uporabo optičnih senzorjev.

Vsak merilec položaja tipalnega trna za posamezno prostostno stopnjo je sestavljen iz šestih infrardečih svetlečih diod (LED) s premerom 1,5 mm, ki so razvrščene ena nad drugo v oddaljenosti 4 mm. Za senzorje smo uporabili ustrezne fotodiode enake velikosti. Na gibljiv trn je pritrjena maska z grayevo kodo, ki ima to lastnost, da se pri prehodu iz vsake vrednosti v naslednjo vrednost spremeni samo en bit. Maska se premika med infrardečimi svetlečimi diodami in fotodiodami. Kombinacija osvetljenih fotodiod tvori signal S, ki vsebuje informacijo o položaju trna. S šestimi fotodiodami lahko ločimo 64 različnih položajev trna. Da dobimo dovolj ozek snop infrardeče svetlobe, uporabimo kovinsko režo debeline 0,2 mm (Slika 2).

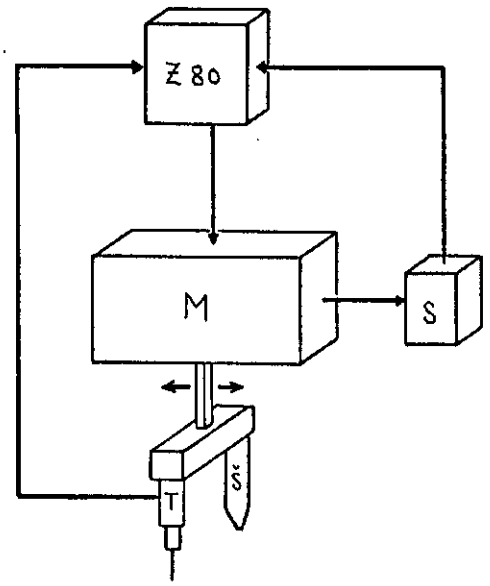


Ž = svetleče diode
M = maska
T = trn
R = reža
D = fotodiode
S = signal položaja trna

Slika 2: Konstrukcija optičnega senzorja

3. POVRATNA ZANKA POZICIONIRANJA

Poleg precizne informacije o trajektoriji reže zvara je za natančno pozicioniranje varilne šobe potrebna povratna zanka na motorju, ki premika šobo (Slika 3). Signal o položaju trna obdelamo v mikroračunalniku, in dobimo velikost in smer premika motorja. Premik motorja lahko najenostavneje in precizno merimo optično s prekinjanjem žarka s pomočjo reže, ki je pritrjena na osi motorja. Informacija o velikosti premika motorja je torej digitalna in s štetjem svetlobnih impulzov s fotodiodo na osi motorja smo zelo enostavno izvedli povratno zanko in rešili pozicioniranje šobe.



T = tipalo
µC = Z80
M = motor za pozicioniranje
Š = senzor položaja motorja
Š = varilna šoba

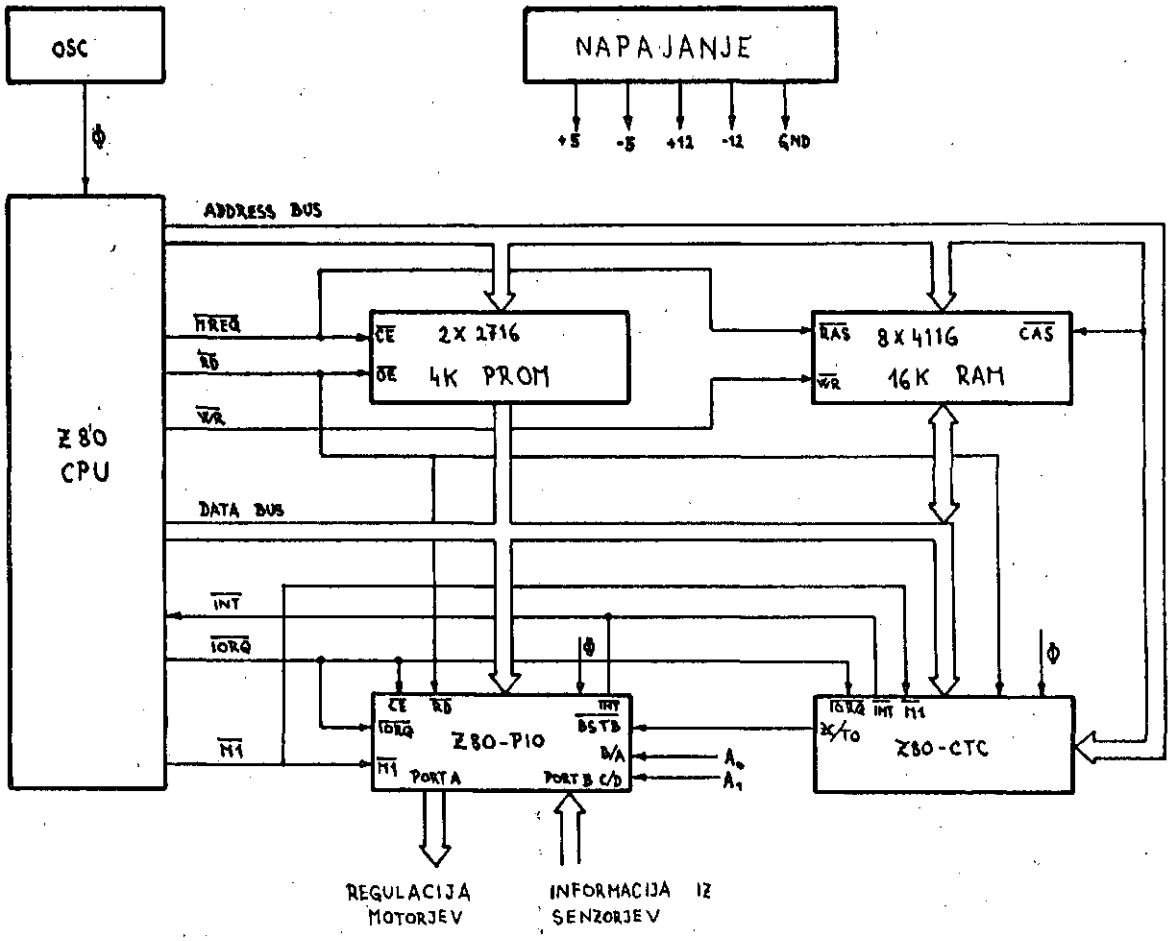
Slika 3: Povratna zanka pri pozicioniranju varilne šobe

4. ARHITEKTURA MIKRORAČUNALNIŠKEGA KONTROLNEGA SISTEMA

Zaradi relativno obsežnega programa in številnih dodatnih nalog kontrolnega sistema ter zaradi možnosti razširitve obstoječega koncepta nalog mikroračunalnika, smo se odločili za uporabo mikroračunalnika Z80. Ta mikroračunalnik je primeren tudi zaradi nekaterih ukazov, ki jih pri drugih mikroračunalnikih nimamo. S tem postane programiranje s sistemom Z80 zelo elegantno in hitrejše. Taki ukazi so na primer aviomatično ponavljanje določenih ukazov dokler so izpolnjeni ustrezni pogoji, pogojni relativni skoki, neposredna manipulacija z biti in drugi.

Sistem ima poleg ožjih članov družine Z80 (Z80-CPU, Z80-PIO, Z80-CTC) in potrebnih vmesnih komponent (kot so invertorji, NAND-vrata, multiplekserji) tudi dva EPROM-a s po 2 K bytov spomina, ki vsebujata program za Z80, nekaj tabel (kot na primer tabelo za prepoznavanje grayeve kode, tabelo za različne hitrosti varjenja) in potrebne konstante (o številu korakov med meritvijo položaja trna in uporabo tega podatka v programu za pozicioniranje šobe). Sistem ima še 8 dinamičnih RAM-ov s po 16 K bitl spomina (Slika 4). Konfiguracija na sliki 4 kaže osnovni sistem, ki ga sestavljajo:

1 x Z80-CPU ; 8 bitni mikroprocesor
1 x Z80-PIO ; 8 bitna paralelna vhodno-izhodna enota
1 x Z80-CTC ; kontrolni programirljivi časovnik
2 x 2716 ; 2K x 8 bitni PROM
8 x 4116 ; 16K x 1 bitni dinamični RAM
2 x 74LS157 ; izbirno vezje 8:4 (multipleksor)
2 x 7430 ; 8-vhodna NAND vrata



Slika 4: Bločni diagram mikroračunalniškega kontrolnega sistema

in še nekatera vmesna TTL vezja. Konfiguracija vsebuje kristalni oscilator z nižjo frekvenco (2 MHz) kot je maksimalna dovoljena od proizvajalca (4 MHz) in s tem zadostuje potrebam, ki jih zahtevamo od sistema. Vezje je ožičeno na standardni dvojni evropski kartici.

5. KONTROLNI PROGRAM

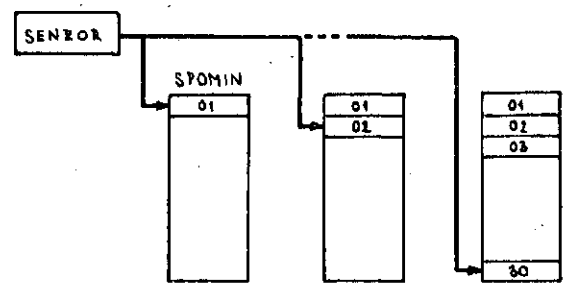
Kontrolni program mora poskrbeti za čitanje signala iz senzorjev in kontrolirati motor za pozicioniranje varilne šobe. Ker je tipalo nameščeno 30 mm pred šobo, pride do zakasnitve med čitanjem signala in trenutkom, ko se nahaja šoba na mestu, kjer je bil ta odmik izmerjen. Vrednost odmika je za ta čas spravljena v spominu (RAM-u). Čitanje položaja tipalnega trna je na vsak milimeter v smeri reže. Ker je razdalja med tipalom in šobo 30 mm imamo ves čas spravljenih v spominu 30 meritev in imamo zanko z zakasnitvijo 30 korakov. Program poskrbi, da se signal (Sx, Sy) položaja trna prečita ob pravem trenutku, ki ga meri Z80-CTC. Premik šobe se izračuna glede na položaj trna pri prejšnji meritvi

$$(\Delta x,\Delta y)_i = k_x\, k_y\, [(S_x, S_y)_i - (S_x,S_y)_{i-1}]$$

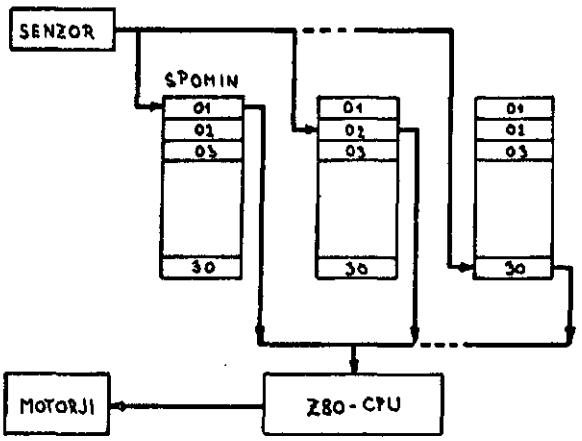
Na začetku reže, ko tipalo že sledi reži, šoba pa zaradi zaostanka še ni dosegla reže so vse vrednosti (Sx,Sy) enake nič in je tudi $(\Delta x,\Delta y)_i = 0$ $\quad i = 1,2,\ldots,30.$

Na koncu varjenja pa imamo obratni položaj, ko je tipalo že izven reže, šoba pa še vari. V tem primeru je čitanje signala iz tipala končano in črpamo le še vrednosti (Sx,Sy) iz spomina dokler ne obdelamo vseh 30 vrednosti.

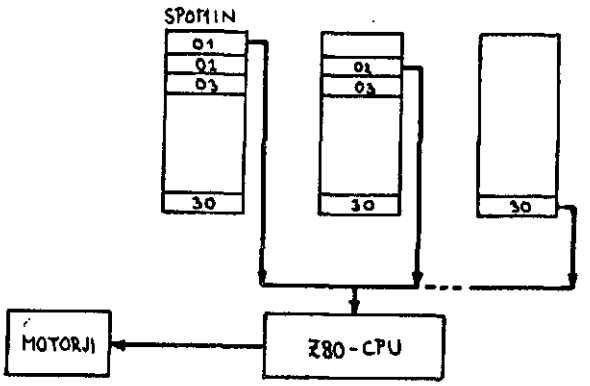
Organizacija signalov v pomnilniku



1 faza: POLNENJE - tipalo daje signale o položaju trna, šoba še ne vari. V tej fazi nimamo čitanja ampak le polnenje pomnilnika.



II faza: OBRATOVANJE - v mikroračunalniku Z80 obdelamo signale, ki jih dobivamo iz pomnilnika in hkrati na izpraznjeno mesto zapisujemo vrednosti signalov iz tipala.



III faza: PRAZNENJE - tipalo je že izven reže, Z80 čita vrednosti iz pomnilnika in ga postopoma prazni.

Po vsakem izračunu premika šobe (Δx,Δy) i dobimo iz Z80 digitalno informacijo o velikosti translacije šobe. Na osi vsakega motorja, ki premika šobo skupaj s tipalom v x oziroma y smeri imamo zaslonko z režo , ki ob vrtenju prekinja svetlobni žarek. S štetjem prekinitev žarka in ob upoštevanju prenosa motorja kontroliramo velikost premika šobe. Premik je končan, ko je število prekinitev žarka enako digitalni vrednosti Δx oz. Δy, za kar poskrbi program v Z80. Povratna zanka na ta način omogoča precizno pozicioniranje šobe.

6. SKLEP

Digitalno tipalo ob mikroračunalniški podpori pomeni nov korak k avtomatizaciji varjenja in k večji preciznosti delovanja robotov za varjenje. Hkrati omogoča večjo fleksibilnost avtomatskih naprav za varjenje in varilskih robotov ter s tem njihov boljši izkoristek. Opisan senzorski sistem še ni bil preizkušen v industrijskem okolju ki je polno prahu, kjer prasketajo kapljice železa okrog varilne šobe in kjer imamo zaradi obločnega varjenja zelo močne elektromagnetne motnje. Zato bo potrebno v industrijskem okolju celoten senzorski sistem primerno zaščititi proti vsem tem motnjam.

Prednost obstoječega mikroračunalniškega kontrolnega sistema je tudi možnost razširitve programa in nalog sistema Z80. Pri varjenju konstrukcij, ki vsebujejo kot 90 stopinj potrebujemo na primer tipalo, ki se je sposobno odkloniti za 3 cm oziroma toliko, kolikor je razdalja med tipalom in šobo. Če varimo pod večjim kotom od 18 stopinj je treba upoštevati spremembo hitrosti varjenja, saj je hitrost varjenja nastavljena v smeri naprej pod kotom 0 stopinj. Želimo tudi, da bi robot lahko sam poiskal začetek reže in ne bi bilo treba pred varjenjem tipala ročno vsaviti v režo. Vse te razširitve in izpopolnitve obstoječi sistem omogoča in je bil s tem namenom tako zgrajen.

7. BIBLIOGRAFIJA

Zilog Data Book

MOSTEK 1978 , Memory Data Book & Designers Guide

TESTNO USMERJEN JEZIK TESTOL

J. ŠILC,
P. KOLBEZEN

UDK: 681.3.06

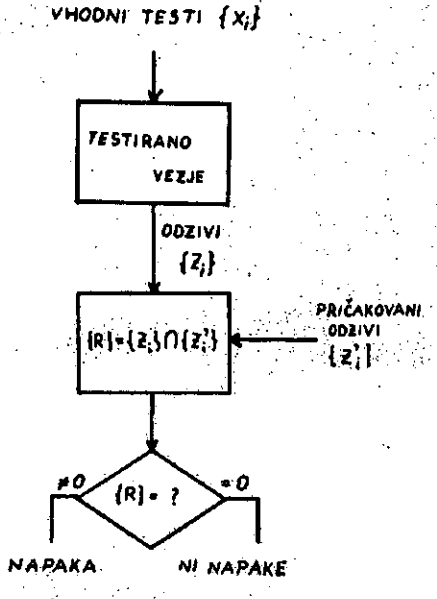
INSTITUT JOŽEF STEFAN, LJUBLJANA

V delu je podan pregleden opis programskega jezika TESTOL, ki omogoča bolj ali manj učinkovito statično, dinamično in funkcionalno testiranje digitalnih vezij in sistemov.

TEST ORIENTED LANGUAGE TESTOL. In this paper a review of programming language TESTOL is presented which enables efficient static, dynamic and functional testing of digital circuits and systems.

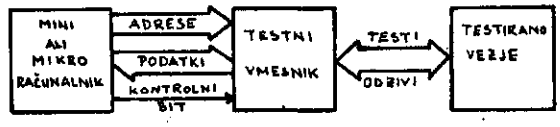
1. UVOD

Testiranje digitalnih vezij se v splošnem sestoji iz zaporednega dodajanja testnih naborov X_i, $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$, na vhode testiranega vezja in opazovanja ustreznih odzivov Z_i, $Z_i = (z_{i1}, z_{i2}, \ldots, z_{im})$ ter primerjanje le-teh z že v naprej izračunanimi oz. pričakovanimi odzivi Z_i'. Vsako protislovje oz. neujemanje določa napako (slika 1.).



Slika 1.

Takšno testiranje je izvedeno s pomočjo mini ali mikro računalnika in ustreznega vmesnika, ki omogoča komuniciranje računalnika s testiranim vezjem (slika 2.). Vmesnik nam omogoča hkratno vpisovanje oz. branje z vseh vhodnih oz. izhodnih priključkov testiranega vezja.



Slika 2.

Predno začnemo s testiranjem, je potrebno podati računalniku informacijo o testiranem vezju. Definirati moramo vhodne in izhodne priključke testiranega vezja, določiti obliko in časovni potek vhodnih testov X_i in pričakovane odzive Z_i'. V ta nemen je bil determiniran uporabniško usmerjen jezik TESTOL (TEST Oriented Language).

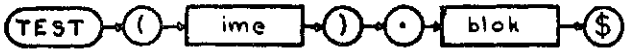
2. TESTNO USMERJEN VHODNI JEZIK TESTOL

TESTOL nam omogoča enostaven in učinkovit opis vhodnih in izhodnih priključkov testiranega vezja, kakor tudi enostavno določitev vhodnih testov in ustreznih pričakovanih odzivov. S pomočjo TESTOL-a je mogoče določiti:

- vhodne priključke,
- izhodne priključke,
- inicializacijske vhodne nabore,
- vhodne teste,
- pričakovene odzive in
- časovni potek testiranja, to je zakasnitve

med posameznimi vhodnimi testi, kakor tudi zakasnitve med testi in ustreznimi odzivi.

Sintaksa TESTOL-a je opisana s sintaktičnimi diagrami. Program napisan v TESTOL-u je sestavljen iz dveh delov, ki ju imenujemo "glava programa" in "blok" (slika 3.).

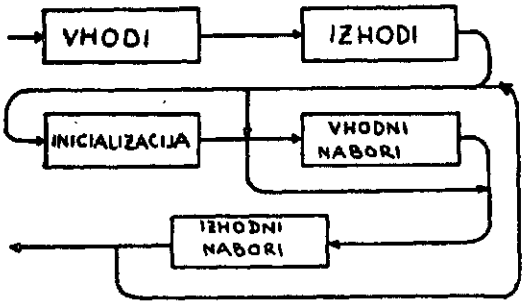


Slika 3. Program

Glava programa ima to nalogo, da da testnemu programu ime. Blok je sestavljen iz petih stavkov, ki si morajo slediti v predpisanem vrstnem redu, kot je prikazano na sliki 4. Pripomniti velja, da lahko presledke in prehode v novo vrstico uporabljamo kjerkoli v programu, pač glede na preglednost programa.

STAVEK VHODI

Vhodne priključke testiranega vezja opišemo s stavkom VHODI. Množici priključkov (oznake priključkov so identične oznakam na evropa konektorju, preko katerega je testirano vezje priključeno na vmesnik) lahko priredimo simbolično ime, ki ima lahko največ 10 alfanumeričnih znakov in se začenja s črko (slika 5.). Število



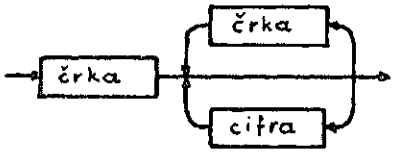
Slika 4. Blok

STAVEK IZHODI

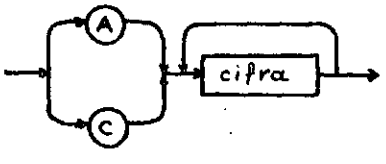
S stavkom IZHODI opišemo izhodne priključke testiranega vezja. Enako kot v stavku VHODI tudi tu množici priključkov priredimo simbolično ime. Vrstni red navedenih priključkov ima enak smisel kot pri navajanju vhodnih priključkov. Velja pripomniti, da isti priključek ne more obenem pripadati množici vhodnih in množici izhodnih priključkov. Sintaksa stavka IZHODI je prikazana na sliki 9.

STAVEK VHODNI NABORI

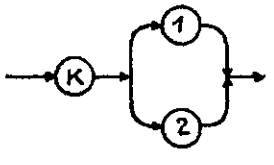
S stavkom VHODNI NABORI določimo število, obliko in časovni potek vhodnih testov, kakor tudi informacijo o tem, ali in po kolikem času opa-



Slika 5. Ime



Slika 6. Oznaka priključka



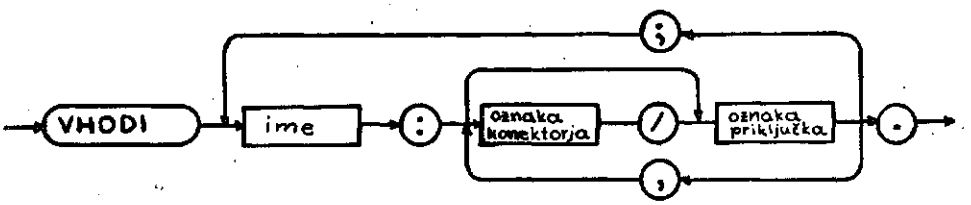
Slika 7. Oznaka konektorja

priključkov, ki pripadajo imenu ni omejeno. Pri navajanju priključkov je pomembno, da si sledijo v pravem vrstnem redu. Ta podrobnost je pomembna zato, ker pri opisovanju vhodnih oz. inicializacijskih naborov prirejamo posameznim imenom določene vrednosti in je s pozicijo določenega priključka v imenu, določena njegova binarna vrednost (slika 6.). Pri navajanju priključkov na vezja realizirana na tiskanini dvojnega evropa formata, moramo navesti kateremu konektorju priključki pripadajo (slika 7.). Če oznake konektorja ne navedemo se smatra, da gre za konektor K1. Sintaksa stavka VHODI je prikazana na sliki 8.

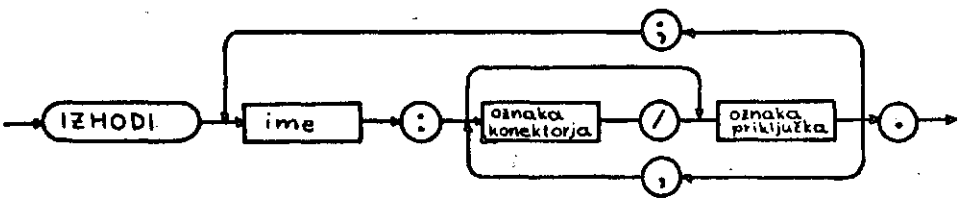
zujemo ustrezne odzive. Število vhodnih testov je podano s parametrom "število korakov" (slika 10.). Z enim stavkom VHODNI NABORI definira-

Slika 10. Število korakov

mo m, $1 \leq m \leq 1030$, vhodnih testov. Številu korakov sledijo imena, ki so določena v stavku VHODI, katerim priredimo določene vrednosti, ki jih navajamo v desetiškem številskem sestavu. Imenom, ki jih eksplicitno ne navedemo, se priredi vrednost 0. Vrednosti imen se pretvorijo v dvojiški številski sestav in posamezni biti



Slika 8. Stavek VHODI



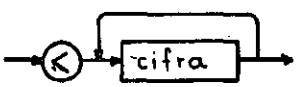
Slika 9. Stavek IZHODI

predstavljajo, v skladu z vrstnim redom priključkov v sklopu imena, binarno vrednost vhodnega priključka. V vsakem koraku se nad vrednostjo imena lahko izvrši ena od naslednjih aritmetičnih operacij:

1. INC (*inkrement*) - *vrednosti imena se* v vsakem koraku prišteje 1
2. DEC (dekrement) - vrednosti imena se v vsakem koraku odšteje 1
3. NEG (negacija) - vrednost imena se v vsakem koraku negira
4. SUBn (odštevanje) - vrednosti imena se v vsakem koraku odšteje n
5. ADDn (prištevanje) - k vrednosti imena se v vsakem koraku prišteje n
6. MULn (množenje) - vrednost imena se v vsakem koraku množi z n
7. DIVn (deljenje) - vrednost imena se v vsakem koraku integer delji z n.

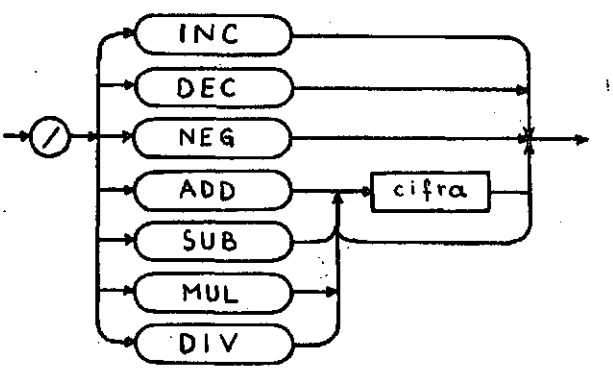
Sintaksa aritmetične operacije je podana na sliki 11. Nadalje je s stavkom VHODNI NABORI možno podati časovni potek vhodnih testov. To nam omogoča parameter "vhodna zakasnitev". (slika 12.). Za znakom > navedemo faktor vhodne zakasnitve N, ki nam pove, kolikokrat bo T_{vn} (čas med vhodnimi nabori) večji od minimalne zakasnitve τ; $T_{vn} = N\tau$. Minimalna zakasnitev τ je

Slika 12. Vhodna zakasnitev



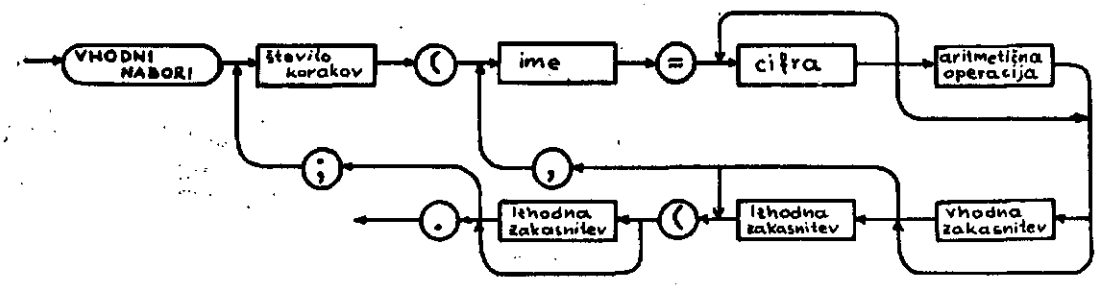
Slika 13. Izhodna zakasnitev

odvisna od računalnika, ki testiranje krmili, maksimalni T_{vn} je omejen z maksimalnim celim številom N_{max}, ki ga ta računalnik lahko procesira; $1 \leq N \leq N_{max}$. Če parameter "vhodna zakasnitev" izpustimo velja N = 1. Stavek VHODNI NABORI nam omogoča določiti tudi časovni potek opazovanja odzivov vezja. To nam omogoča parameter "izhodna zakasnitev" (slika 13.). Za znakom < navedemo faktor izhodne zakasnitve M, ki



Slika 11. Aritmetična operacija

nam pove, kolikokrat bo T_{izh} (čas med vhodnim testom in opazovanjem ustreznega odziva) večji od minimalne zakasnitve τ_0; $T_{izh} = M\tau_0$. Parameter je mogoče pisati pred ali za okrogli zaklepaj (slika 14.). V prvem primeru, ko je izhodna zakasnitev znotraj okroglega oklepaja, se izhodna zakasnitev *nanaša na m odzivov*, ki pripadajo m vhodnim testom. Takšna uporaba je smiselna le v primeru, ko je $T_{izh} \leq T_{vn}$. V drugem primeru, ko je izhodna zakasnitev podana za okroglim zaklepajem, opazujemo odziv testiranega vezja, ki je posledica vhodnega testa oz. množice vhodnih testov, po času T_{izh} za zadnjim vhodnim testom. Če parametra "izhodna zakasnitev" ne pišemo, to ne pomeni, da je M = 1 (to moramo eksplicitno napisati), temveč, da je čas T_{izh} neskončno oz., da ustreznega odziva ne opazujemo. Sintaksa stavka VHODNI NABORI je prikazana na sliki 14.



Slika 14. Stavek VHODNI NABORI

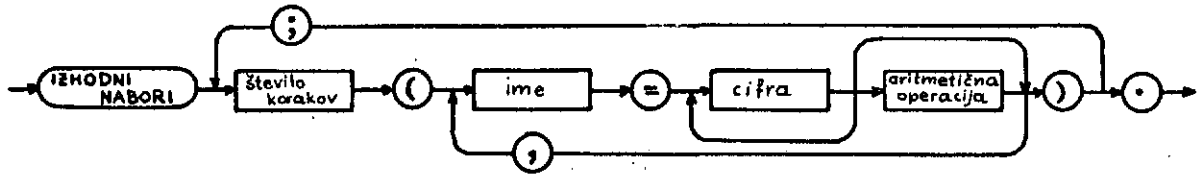
STAVEK IZHODNI NABORI

S stavkom IZHODNI NABORI definiramo pričakovane odzive testiranega vezja. Pri navajanju pričakovanih odzivov moramo paziti, da se njihovo zaporedje ujema z zaporedjem ustreznih vhodnih testov. Tudi tu velja, da se imenom, ki jih eksplicitno ne navedemo priredi vrednost 0. Sintaksa stavka IZHODNI NABORI je prikazana na sliki 15.

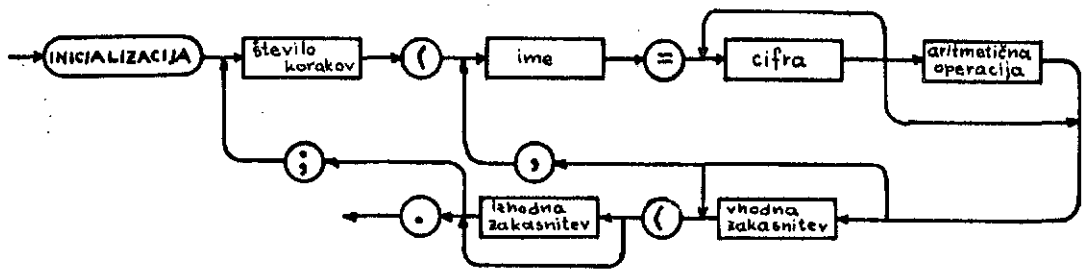
čakovan odziv testiranega vezja na inicializacijske nabore. V primeru, ko se dejanski in pričakovan odziv razlikujeta inicializacijo ponovimo.

3. ZAKLJUČEK

Namen pričujočega članka je predvsem seznaniti bralca z osnovnimi značilnostmi testno usmerjenega jezika TESTOL-a. S pomočjo jezika TESTOL



Slika 15. Stavek IZHODNI NABORI



Slika 16. Stavek INICIALIZACIJA

STAVEK INICIALIZACIJA

Sekvenčna vezja je pred pričetkom testiranja koristno inicializirati, saj se po priključitvi na napajalno napetost postavijo v nedefinirana stanja. Če sekvenčna vezja nimajo posebnih sinhronizacijskih vhodov, jih je potrebno s pomočjo normalnih vhodnih signalov, ki jih določimo s stavkom INICIALIZACIJA, postaviti v znano stanje. Stavek INICIALIZACIJA (slika 16.) je sintaktično podoben stavku VHODNI NABORI. Z njim generiramo inicializacijski nabor oz. množico inicializacijskih naborov, ki nam postavi sekvenčno vezje v znano stanje, enako kot vhodne teste. Stavku INICIALIZACIJA lahko sledi stavek IZHODNI NABORI, s katerim določimo pri-

je mogoče izvajati statično, dinamično in (predvsem) funkcionalno testiranje. Prevajalnik za jezik TESTOL je napisan v visokem programskem jeziku PASCAL.

4. LITERATURA

(1) A. KRIŽNIK: Diplomska naloga, Ljubljana 1979

(2) K. JENSEN, N. WIRTH: PASCAL (User Manual and Report) Springer - Verlag, New-York Heidelberg Berlin, 1974

MEHURČNI POMNILNIKI – I. DEL

B. MIHOVILOVIĆ,
J. ŠILC,
P. KOLBEZEN

UDK: 681.327.664.4

INSTITUT JOŽEF STEFAN, LJUBLJANA

Prvi v seriji člankov o mehurčnih pomnilnikih podaja širši vpogled v mesto mehurčnih pomnilnikov v hierarhiji pomnilnikov. Smo v času, ko se takšen pomnilniški medij šele uveljavlja, zato so v tem delu nekoliko širše predstavljeni proizvajalci mehurčnih pomnilniških sistemov, ki jih ni malo in so dokaz, da so mehurčni pomnilniki danes realnost. Podano je današnje tehnološko in ekonomsko stanje v svetu in pogled v bližnjo prihodnost. Na koncu je posvečeno nekaj besed osnovnim karakteristikam in možnostim uporabe mehurčnih pomnilnikov.

MAGNETIC BUBBLE MEMORIES - PART 1. The first in a series of articles on magnetic bubble memories (MBM) reviews the place of bubble in the memory hierachy. We live in the time when such memory medium haven't been brought forward yet. Therefore the emphasis of presentation is mainly on the producers of magnetic bubble devices (MBD), the number of which is rather large what proves that the bubble memories are a reality nowadays. It gives the today technological and economic state in the world and the sight into near future. At the end few words are given on the basic characteristics and the possibilities of the use of bubble memories.

1. UVOD

V sedemdesetih letih[1)] so se pojavili mehurčni pomnilniki (z uporabnimi lastnostmi masovnih perifernih pomnilnikov), ki imajo nekaj svojskih lastnosti, kot so: prirojena zanesljivost, neobčutljivost na okolico in visoka gostota.

Kljub naštetim dobrim lastnostim, pa se načrtovalci in uporabniki mehurčnih pomnilniških sistemov soočajo z nekaterimi problemi, ki spremljajo realno uporabo mehurčnih pomnilnikov, kot so kompleksna nadzorna logika, ojačevanje razmeroma slabotnih signalov in generiranje visokih tokovnih impulzov. Danes so nekateri proizvajalci naštete tehnološke probleme že uspešno rešili, tako da so mehurčni pomnilniški sistemi danes že realnost. Zaradi njihove majhnosti, učinkovitosti, majhne porabe moči in ostalih prirojenih lastnosti, so mehurčni pomnilniški sistemi primerljivi z diskovnimi in tračnimi pomnilniškimi sistemi.

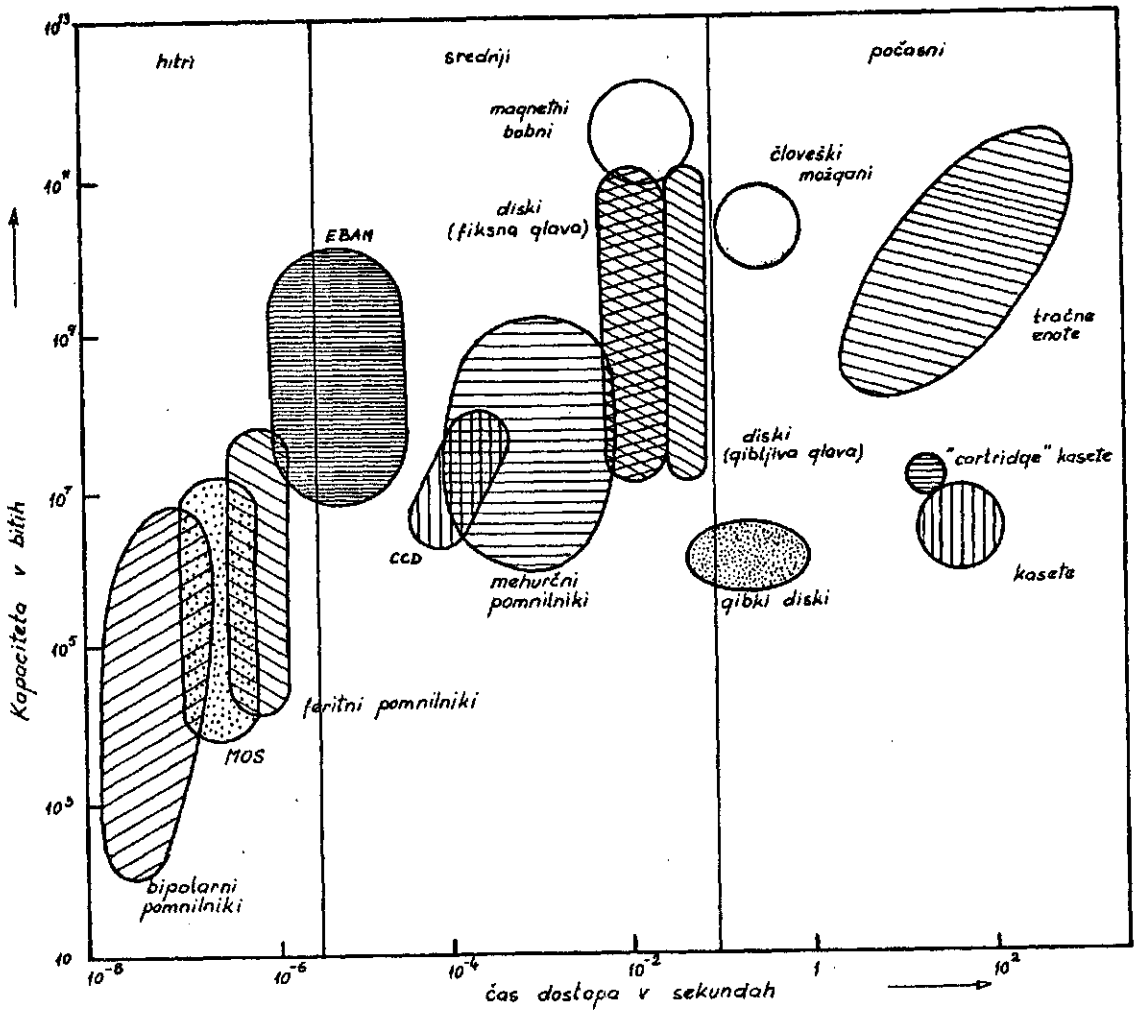
1) Čeprav je raziskovalna skupina A. H. Bobecka v Bell Telephone Laboratories že v začetku leta 1967 naredila prve eksperimente [1] na področju tvorbe cilindričnih oblik magnetnih domen v feromagnetikih, so se rezultati teh raziskav realizirali v obliki mehurčnih pomnilnikov šele v sedemdesetih letih.

2. MESTO MEHURČNIH POMNILNIKOV V HIERARHIJI POMNILNIKOV

Da bi prikazali položaj mehurčnih pomnilnikov v množici elektronskih in neelektronskih pomnilnikov, je potrebna medsebojna primerjava v večih dimenzijah. Ena od dimenzij je vsekakor kapaciteta pomnilnika in druga je čas dostopa. Tretja dimenzija, ki je zelo pomembna z ekonomskega stališča, je cena na bit vgrajenega pomnilnika. Nenazadnje je tu še četrta dimenzija, ki ponazarja časovni tehnološki razvoj pomnilnika (otroštvo, adolescenca, zrela doba in starost). Z množico naštetih dimenzij, bi lahko zgradili najmanj tri diagrame, s katerimi bi primerjali posamezne pomnilnike med seboj.

Na sliki 1. je podan diagram, ki prikazuje odvisnost med kapaciteto pomnilnika in časom dostopa za posamezne tehnologije pomnilnikov.

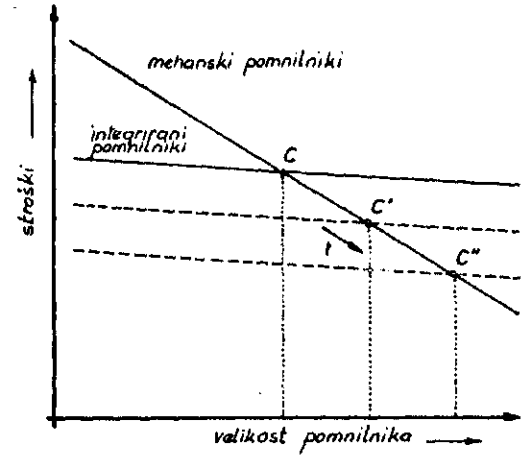
Če pogledamo na diagram s stališča pomnilnikov, katerih tehnološki razvoj je že v t.i. zreli dobi ali starosti, opazimo glede na čas dostopa, veliko praznino med feritnimi pomnilniki na eni in magnetnimi diski s fiksno glavo na drugi strani. To vrzel poizskušajo zapolniti tri tehnologije (ki pa so še v otroški oz. adolescentni dobi) in sicer CCD (charge coupled devices), mehurčni pomnilniki



Slika 1.

(MBM) in EBAM (electron-beam accessed memories). EBAM pomnilniki naj bi predvidoma imeli nizko ceno na bit vgrajenega pomnilnika, toda imajo to slabo lastnost, da vsebujejo krhke vakuumske komponente, kar ta pomnilnik loči od ostalih dveh nemehanskih pomnilnikov. CCD pomnilniki so glede na čas dostopa enakovredni mehurčnim pomnilnikom, vendar je njihova kapaciteta manjša (približno enaka kot pri MOS pomnilnikih), kar ne zagotavlja njihove uspešnosti. Lahko rečemo, da bodo obstoječo vrzel uspešno zapolnili le mehurčni pomnilniki.

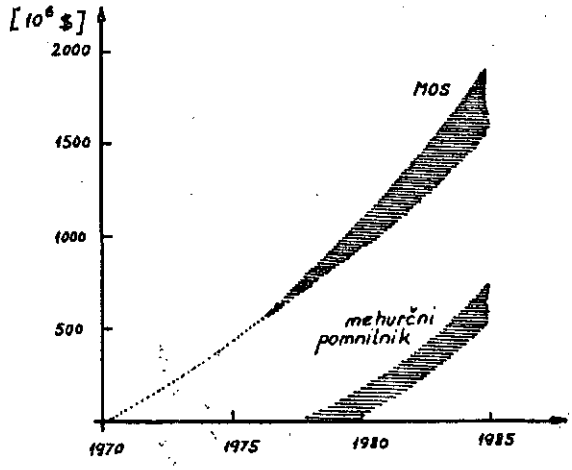
Če grupiramo pomnilnike v dve skupini, to je mehanske in integrirane (solid state) pomnilnike, ter jih opazujemo v prostoru stroški/velikost pomnilnika, vidimo, da je uporaba prvih upravičena na področju velikih pomnilnikov. Za integrirane pomnilnike pa ugotovimo, da so stroški tako pri majhnih kot pri velikih pomnilnikih praktično izenačeni. Kot je razvidno iz slike 2. je uporaba integriranih pomnilnikov (med katere sodijo tudi mehurčni pomnilniki) bolj upravičena pri manjših pomnilnikih, vendar zagotovo lahko trdimo, da bo točka C, v kateri se stroški obeh tipov pomnilnikov izenačijo, s časom prešla v C' oziroma C", itd. Ali drugače povedano, v prihodnje bodo integrirani pomnilniki vse bolj izpodrivali mehanske pomnilnike tudi na področju velikih pomnilnikov.



Slika 2.

Čeprav mehurčni pomnilniki po času dostopa niso primerljivi z MOS pomnilniki, bomo na naslednjem diagramu

(slika 3.) prikazali predvideno produkcijo obeh tehnologij v naslednjih letih. Vidimo, da je v letu 1980 razmerje med produkcijo MOS (dinamičnih) pomnilnikov in mehurčnih pomnilnikov še 10:1, že v letu 1985 pa se bo to razmerje predvidoma zmanjšalo na 10:3.



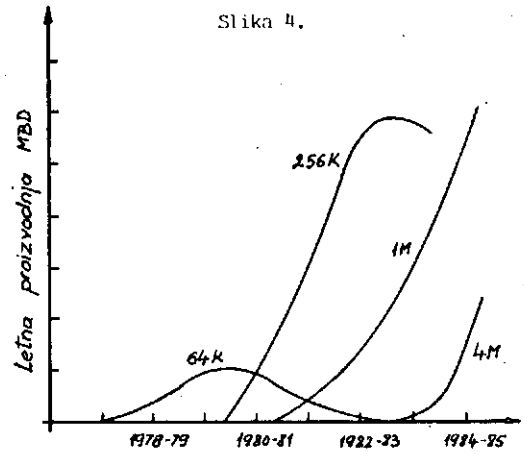
Slika 3.

3. ALI PREDSTAVLJAJO MEHURČNI POMNILNIKI DANES REALNOST

Vsekakor! Od prvih poizkusov na področju tvorbe cilindričnih oblik magnetnih domen v letu 1967 je minilo pet let do izdelave osnovne pomnilniške strukture, v kateri je bilo možno izvesti vpis, branje, pomik ter brisanje informacije. Kapacitete prvih mehurčnih pomnilnikov (MBM) so bile manj kot 1024 bitov (1 Kbit). Po letu 1972 pa je kapaciteta MBD rapidno naraščala, tako da je bila v letu 1979 256 Kbitov že standardna velikost (danes je kapaciteta MBM že 1 Mbit in več), to pa pomeni v povprečju trikratno povečevanje kapacitete MBM letno. Zanimivo je, da se je kapaciteta MOS pomnilnikov v času njihovega tehnološkega razvoja povečevala s faktorjem 2 letno. Takšno progresivno naraščanje kapacitete MBD lahko realno pričakujemo tudi v osemdesetih letih, saj se že danes kažejo rezultati raziskav na področju uvajanja novih materialov in postopkov v tehnologijo izdelave mehurčnih pomnilnikov. Če pa le obstajajo kakšne omejitve, seveda naravne oblike, pa te ne preprečujejo realizacije MBM gostote vsaj 64 Mbitov/cm^2 (morda pa celo 256 Mbitov/cm^2). Slika 4. prikazuje zgodovinski in bodoči pogled na letno proizvodnjo MBM različnih kapacitet v letih 1978-1985. 1 Mbitni MBM, ki so danes že realizirani v obliki čipa, shranjujejo informacijo, ki je enaka več kot 30-tim tipkanim stranem podatkov ali preko 300 m luknjanega traku. Npr. današnji dinamični RAM pomnilniki imajo kapaciteto 64 Kbitov, ROM pomnilniki 128 Kbitov in mini gibki diski z enostranskim zapisom 720 Kbitov.

Naštejmo še nekaj lastnosti, ki opravičujejo vse večjo uporabnost tega pomnilniškega medija. Prva je vsekakor preprost fotolitografski postopek obdelave substrata (potrebna sta dva koraka, medtem ko je pri MOS tehnologiji teh korakov sedem). Druga dobra lastnost je v večjem izkoristku substrata pri izdelavi MBM čipov, kot v drugih tehnologijah. Naslednja dobra lastnost MBM pa je, da so ob izdelavi čipa vnešene t.i. redundančne spominske zanke (redundancy storage loops), ki omogočajo obnovitev določenega števila (do nekaj 10 %) defektnih spominskih zank. Tu je še ena od bistvenih lastnosti MBM in sicer njegova nezbrisljivost (non volatility), ki se odraža v tem, da se ob izpadu napajanja pozicije magnetnih mehurčkov (informacija) ohranjajo.

Slika 4.



4. TEHNOLOŠKO IN EKONOMSKO STANJE V SVETU

V zadnjih letih se je pojavila množica proizvajalcev MBM tako v ZDA, na Japonskem, kot v Evropi. Oglejmo si tehnološko in ekonomsko stanje v svetu.

ZDA:

Bell Telephone Laboratories

Kljub temu, da so v teh laboratorijih realizirali prvi uporabljiv MBM, sedaj razpolagajo le z 64 Kbitnimi MBM, ki jim zadoščajo za pokrivanje standardnih elementov v telefonskih sistemih. Njihov nadaljni razvoj je usmerjen v izboljšanje postopka ionske implantacije propagacijskih vzorcev (I2P2 - ion implanted propagation patterns) z namenom zmanjšanja premera magnetnih mehurčkov, to je povečanja gostote MBM čipov.

IBM

IBM je osvojil tri tehnologije izdelave MBM in sicer: postopek permalojnih vzorcev (permalloy patterns), tehnologijo I2P2 in njihovo lastno odkritje "urejena mreža mehurčkov" (Bubbles Lattice File)[2].

Hewlett-Packard

Po nekajletnih raziskavah je pri HP zanimanje za MBM nekoliko popustilo. Kmalu pa so delo na tem področju obnovili in zgradili so 1 Mbitni MBM, ki pa je specializiran

2) Vobče so vse do tedaj znane tehnologije "shranjevale" informacijo tako, da je prisotnost magnetnega mehurčka predstavljala logično 1 in njegova odsotnost logično 0. Pri BLF tehnologiji pa govorimo o zaprti gomilasti grupi magnetnih mehurčkov in informacija je shranjena v obliki različnih stanj stene, ki obkroža mehurčno domeno [2].

za njihove lastne potrebe. Perioda propagacijskega vzorca [3] je 8 μ, propagacijska frekvenca pa je 200 KHz, ima pa tudi zelo visoko redundanco 40 %.

Intel

Prve poizkuse so pri Intlu izvedli šele po letu 1977 in kmalu plasirali na tržišče 1 Mbitni MBM s štirimi podpornimi čipi in sicer kontrolerjem, generatorjem tokovnih impulzov, ojačevalnikom in driverjem navitij. Našteti elementi so grajeni tako, da se enostavno povezujejo z mikroračunalniškimi sistemi istega proizvajalca. Propagacijski elementi imajo obliko nesimetričnih škarnic (asymmetric chevron) s periodo 10 + 12 μ pri premeru magnetnega mehurčka 2,7 μ. Ta MBM vsebuje 48 redundančnih zank pri 320 pomnilniških zankah in njegova propagacijska frekvenca je 100 KHz.

National Semiconductors

Pri Nationalu so pričeli s proizvodnjo MBM v zadnjem četrtletju leta 1977. Njihov prvi proizvod je 256 Kbitni pomnilnik s periodo 16 μ in 7,25 % redundanco.

Rockwell International

Tudi tu so začeli s 256 Kbitnimi MBM, da bi koncem leta 1980 morda uspeli razviti 4 Mbitni MBM [5]. Za 256 Kbitno in 1 Mbitno strukturo velja, da uporablja nesimetrične škarnične permalojne vzorce. Čipi so izdelani s konvencionalnimi fotolitografskimi postopki, vendar so uspeli zmanjšati medvzorčno razdaljo (gap) na 0,5 μ. Za tvorbo magnetnega polja uporabljajo tuljavice, tako da dosežejo frekvenco 100 + 150 KHz. Čip dimenzije 1,5 x 1,5 cm ima redundanco 7,8 %. Pri Rockwellu so v želji za povečanjem kapacitete (4 Mbitov) uporabili novo tehnologijo "ionska implementacija stičnih diskastih vzorcev" (ion-implementation contiguous disk propagation patterns). Prednost tega novega postopka je v tem, da med vzorci, ki so manjši, ni presledkov in s tem dosežejo veliko večje gostote in višje propagacijske frekvence.

Texas Instruments

V letu 1977 so začeli pri TI izdelovati prve MBM z nekoliko svojsko organizacijo (90 Kbitov). Redundančnih zank je bilo 8,3 %, uporabili pa so T obliko propagacijskih permalojnih vzorcev. Takšna oblika vzorcev je dopuščala le propagacijske frekvence 50 KHz, zato so pri naslednjem 248,75 Kbitnem MBM uporabili nesimetrične škarnične vzorce (perioda 16 μ), tako da so dvignili frekvenco na 100 KHz. Leta 1979 so izdelali MBM kapacitete 1 Mbit, z gostoto 10^6 bitov/cm^2 [6].

Japonska:

Fujitsu, Kawasaki

64 Kbitni MBM je predstavljal začetek pri tem vodilnem japonskem prolizvajalcu in je specifičen v tem, da vsebuje nesimetrične polmesečne oblike vzorcev (half-disks or crescents). Razvili so tudi že 256 Kbitni MBM. Kot posebnost pa naj omenimo tudi to, da so pri Fujitsuju usmerili razvoj v izgradnjo specifičnega pomnilniškega medija, tako imenovane mehurčne pomnilniške kasete (magnetic bubble cassette memory) totalne kapacitete 74 Kbitov [3]. Celoten kasetni pomnilniški sistem sestavljajo nosilec kasete, kaseta in kontrolna enota. Kaseta je s 24 pinskim priključkom povezana s kasetnimi nosilcem, je hitro zamenljiva, sorazmerno majhna (60x45x20 mm), proizvajalec pa zagotavlja njeno praktično uporabnost.

Hitachi, Kokobunji

Prvi Hitchijev proizvod je imel kapaciteto 64 Kbitov in je uporabljal T obliko propagacijskih elementov s periodo 20 μ. Ta MBM je imel od 131 spominskih zank 6 redundadančnih. Kasneje so tudi pri Hitachiju prešli na škarnične oz. polmesečne oblike propagacijskih vzorcev. Naslednji proizvod je bil 256 Kbitni pomnilnik z 7,9 % redundanco in 100 KHz propagacijsko frekvenco. Prednost tega proizvajalca je, da je razvil mehurčni pomnilniški sistem (MBD), to je tiskanino, ki vsebuje štiri 256 Kbitne MBM in podporne čipe. Napajanje MBD je enotno (+5V), vse ostale potrebne napetosti (-5V, +20V) so zgenerirane s DC-DC pretvornikom, ki je tudi na tej plošči. MBD je zgrajen tako, da je direktno priključljiv na mikroračunalniške sisteme s procesorjem I8080 ali M6800 [4].

NEC, Kawasaki

Propagacijski vzorci pri tem proizvajalcu so nekoliko svojski, imajo Y-Y obliko. Tako so zgradili 64 Kbitni MBM z redundanco 15 % (paket dveh čipov). Poizkušali so z relativno visoko propagacijsko frekvenco 300 KHz (pri 32V napajanju). Nadaljni razvoj je potekal na 132 Kbitnem čipu z 8,3 % redundanco.

Evropa:

V Evropi se je z razvojem MBM ukvarjalo nekaj proizvajalcev, ki pa so večinoma delali v sodelovanju z ameriškimi proizvajalci (Phillips, Siemens). Samostojni razvoj je potekal le pri angleškem proizvajalcu Plessey Microsystems. Začetek je bil že v letu 1969, v letu 1977 pa so izdelali 64 Kbitni MBM, katerega značilnost je ta, da je čip majhnih dimenzij (2x2 cm). Uporabljajo konvencionalne tehnologije, škarnične propagacijske vzorce s periodo 16 μ. Uspelo pa jim je izdelati tudi 256 Kbitni pomnilnik.

leto	78/79	80/81	82/83	84/85
ZDA	2,2	28	100	260
Japonska	1,0	13	48	130
Evropa	0,8	7	32	90
Skupaj milj. $	4,0	48	180	480

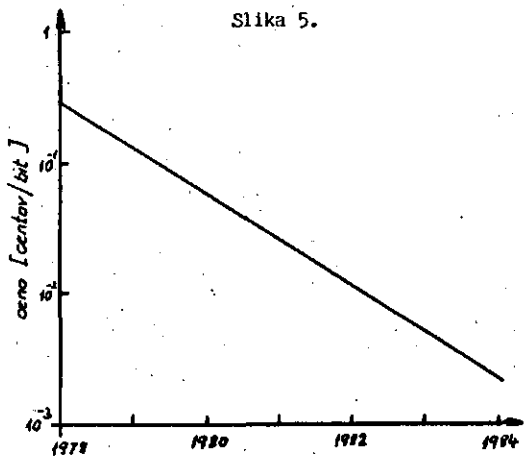
Tabela 1. svetovna proizvodnja MBM v milijon dolarjih

Pričakovati je, da se bo v naslednjih letih proizvodnja MBM močno povečala (tabela 1) in da bo v letu 1985 dosegla vrednost 500 milijonov dolarjev. Pričakovati je tudi, da bodo stroški za izdelavo MBM močno padli (iz

3) Perioda propagacijskega vzorca je določena z geometrijskimi lastnostmi vzorca in medvzorčno razdaljo (gap).

125 ^{mc}/bit v letu 1978 na 1 ^{mc}/bit leta 1985), kar je razvidno iz slike 5.

Slika 5.



5. NAMESTO ZAKLJUČKA

Dejstvo je, da MBD v sebi združujejo zelo bogato strukturo in kompleksno delovanje. Proizvajalci težijo k temu, da bi postala tako struktura MBM, kot celotnega MBD bolj integrirana in enostavnejša za vključevanje v mikroračunalniški sistem. To pa zahteva od načrtovalca mehurčnega pomnilniškega sistema, da je soočen tako s fizikalno tehnološkimi lastnostmi MBM, kakor tudi z načinom vključitve MBM v mehurčni pomnilniški sistem. Zagotovo lahko trdimo, da je današnja podporna oprema, ki spremlja MBM, več ali manj specifična za določene mikroračunalniške sisteme, vendar se proizvajalci trudijo pri načrtovanju enotne minimalne podporne opreme. Vmesniki naj bi bili standardizirani in naj bi imeli možnost korigiranja napak in veliko stopnjo redundance.

Zanesljivost delovanja tega pomnilniškega medija lahko štejemo v njegovo dodatno odliko, saj je maksimalna verjetnost nastopa napake v pomnilniškem sistemu 10^{-12}, v samem MBM pa celo 10^{-40}. Eventuelne napake se najčešče pojavijo v MBD, bodisi v napajalnem modulu ali v detektorjih signalov in jih je mogoče zmanjšati z ustreznim načrtovanjem tiskane plošče, na kateri je ta podporna oprema instalirana. Efekt staranja skorajda nima vpliva na zanesljivost delovanja mehurčnega pomnilnika. Temperatura okolice ne vpliva dosti na delovanje, saj je energijska poraba sistema reda 1 W. Sam mehurčni pomnilnik brez posledic prenese zunanje magnetno polje jakosti do 1600 ^{A}/m (čipi so obdani s feromagnetno oblogo).

Učinkovitost pomnilniških sistemov ovrednostimo s časom dostopa (access time), ta pa je pri mehurčnih pomnilnikih odvisna od organizacije pomnilniških celic[4] in propagacijske frekvence. Tipični časi dostopa za 256 Kbitni MBM so 3 + 10 ms.

K proizvodnji MBD sodi tudi predhodno statično, dinamično in parametrično testiranje komponent, ki je običajno združeno v ATE (automatic test equipment). Zgrajeni testni sistemi so še razmeroma dragi (150 + 200 tisoč dolarjev), vendar pa je enostavnejše testiranje (ki navadno zadošča) dosti cenejše.

Nazadnje lahko poudarimo, da je spekter aplikacij MBD zelo širok. Namen mehurčnih pomnilnikov ni, da bi popolnoma zamenjali obstoječe masovne pomnilnike (diski, bobni, trakovi, kasete, itd.), temveč, da zapolnijo vrzel v hierarhiji pomnilnikov, ocenjeno glede na razmerji kapaciteta/cena in učinkovitost/cena (slika 1.). To pa jim zaradi njihovih svojskih lastnosti (majhna energijska poraba, nezbrisljivost, velika zanesljivost, majhna teža in volumen, kompaktnost, modularnost, robustnost, itd.) tudi uspeva. Naštejmo nekaj področij na katerih se bodo MBD uveljavili zaradi naštetih svojskih lastnosti:

- nadzor industrijskih procesov (zanesljivost, robustnost, minimalno vzdrževanje),
- prenosne mikroračunalniške naprave (modularnost, majhna teža in poraba moči, kompaktnost),
- poslovni računalniški sistemi (zanesljivost, minimalno vzdrževanje, modularnost),
- v neugodnih delovnih pogojih (temperaturna neobčutljivost, zanesljivost, robustnost),
- kot pomožni μP pomnilniki (modularnost, zanesljivost),
- zamenjava diskov s fiksno glavo (velika učinkovitost, zanesljivost).

Nizka cena in nezbrisljivost MBM sta karakteristiki, ki spremljata vsa področja uporabe mehurčnih pomnilnikov, medtem ko ostale karakteristike lahko uspešno zadostimo s skrbno in premišljeno izbrano podporno opremo.

Razvoj mehurčnih pomnilniških sistemov je zelo dinamičen in bo še naprej potekal v dveh smereh: iskanju novih tehnoloških možnosti pri izdelavi samih mehurčnih pomnilnikov v smislu povečanja gostote in učinkovitosti (za leto 1990 napovedujejo celo 256 Mbitne čipe), ter v načrtovanju univerzalnejše in bolj integrirane podporne opreme.

4) Velikosti minorskih zank, ki pomnijo informacijo

6. LITERATURA

[1] A.H.BOBECK: Bell Syst.Tech.J., 46, pp 1901-1925, 1967
[2] C.P.HO&H.CHANG: IEEE Trans.Magnetics,13,pp 945-952, 1977
[3] T.MORIKAWA: Electronic Engineering, pp 95-101, Feb. 1980
[4] Y.KITA, N.YAMAGUCHI, M.SUGIE & S.YOSHIZAWA: IEEE Trans.on Computers, Vol. C-29, No.2, pp 89-96 February 1980
[5] Electronic, pp 41-42, May 1980.
[6] R.E.FONTANA, D.C.BULLOCK & S.K.SINGH: IEEE Trans. on Magnetics, Vol. MAG-16, No. 5, Sept. 1980
[7] P.NEWMAN: Electronic Engineering, pp. 43-58, Sept. 1979
[8] D.J.HUNTER: Electronic Engineering, pp. 39-51, July 1979
[9] L.WALLES: Electronic, pp 80-81, March 1979

UNIVERZITETNI POUK RAČUNALNIŠTVA II

ANTON P. ŽELEZNIKAR

SOZD ELEKTROTEHNA, DO DELTA, LJUBLJANA

UDK: 378.681.3

Nadaljevanje članka prinaša vsebino ostalih dvajsetih modulov univerzitetnega pouka računalništva, kot je bilo prikazano na sliki 1 v članku (1). Ti moduli zajemajo pretežno matematične in teoretične objekte računalniških znanosti; izjemi sta le modula 5.5 (podatkovne baze) in 5.6 (izdelava programov). V tretjem delu (nadaljevanju)bo podana primerjava domačih načrtov in tega predloga z ustrezno analizo stanja.

University Curriculum in Computing II. This conclusion shows the contents of twenty modules of university curriculum in computing as it was graphically presented in Fig. 1 of article (1). These modules include topics of predominant mathematical and theoretical subjects with exception of modules 5.5 (data bases) and 5.6 (implementation of programms). In the next part (conclusion following) a comparition of domestic curricula and IFIP proposal will be given accompanied by the state analysis.

1. Uvod

V drugem delu članka (1) bomo opisali vsebino preostalih 20 modulov iz slike 1 v članku (1). Ker je snov tega dela dokaj obširna, bomo razpravo o primerjavi naših učnih načrtov z načrtom UNESCO/IFIP preložili na tretji del članka.

Omenimo takoj na začetku, da vsebuje opis modulov tega drugega paketa še dva modula, ki sodita v ožji izbor predmetov iz računalništva (in informatike), in sicer: modul 5.5: Oblikovanje in upravljanje podatkovnih baz ter modul 5.6: Oblikovanje in razvoj programov. Preostali moduli so matematične, informacijskoteoretične in družboslovne narave.

2. Nadaljni moduli izobraževalnega paketa

V tem poglavju bomo opisali nadaljnih 20 modulov izobraževalnega paketa.

MODUL 1.2

Linearna algebra (3-0-2)

Cilji

Linearna algebra in diferencialne enačbe oblikujejo telo tistega znanja, ki je za računalniško znanost dovolj širokega pomena. Študenti računalništva naj bi bili s to snovjo dobro podkovani.

Poudarek modula

Predavanja naj bodo uravnotežena med algebraičnim in računalniškim pristopom, ko se začne z linearnimi prostori n-terk in Gaussovimi elementarnimi vrstičnimi operacijami, kar pripelje do predstavitve matrik in k bolj abstraktnim pristopom, ko so matrike predstavitve linearnih upodobitev.

Vsebina modula

1. Linearne enačbe in matrike

sistemi linearnih enačb
ekvivalenca pri elementarnih vrstičnih operacijah
Gaussova eliminacija
matrika koeficientov
vrstično reducirana matrika
izračunavanje, rešitve
matrično multipliciranje
obrnljive matrike
izračunavanje inverzov z elementarnimi vrstičnimi operacijami

--- 10 % ---

2. Determinante

opredelitev in osnovne lastnosti
Cramerjevo pravilo
inverzna matrika
determinanta transponirane matrike in produkta dveh matrik
kriterij obrnljivosti kvadratne matrike

--- 10 % ---

3. Vektorski prostori

abstraktna opredelitev vektorskih prostorov
primeri
linearna odvisnost in neodvisnost
baze in podprostori
razsežnost linearnega prostora

--- 10 % ---

4. Linearna preslikava

linearne upodobitve, jedro in slika upodobitve
linearne upodobitve kot vektorski prostor

izbira baze v vektorski bazi
predstavitev linearnih upodobitev z matrikami
podobne matrike
kompozicija upodobitev in množenje matrik
algebra linearnih upodobitev in matrik
relacija med linearnimi upodobitvami in linearnimi enačbami
obstoj rešitve linearnih enačb v odvisnosti od pridružene linearne upodobitve

--- 20 % ---

5. Linearne neenačbe in konveksne množice

linearne neenačbe in polprostori
simpliciji in konveksne linearne kombinacije
konveksni poliedri in konveksne množice
razločevanje hiperravnin
simpleksna metoda linearnega programiranja

--- 10 % ---

6. Notranji produkt in norme

notranji produkt
dolžina, kot, usmerjeni cosinus
uporaba v linearni in ravninski geometriji
norma
uvod v ortogonalne baze
Gram - Schmidtov postopek ortogonalizacije
ortogonalna razširitev in Fourierevo pravilo

--- 10 % ---

7. Kvadratne oblike

simetrične matrike in kvadratne oblike
kvadratne površine
učinki linearnih transformacij
racionalna redukcija na diagonalo
invarianca indeksa
pozitivna opredeljivost
ortogonalna redukcija kvadratnih oblik 2 x 2
uporaba v ravninski koniki
splošni primer: ortogonalna redukcija, karakteristični koreni in vektorji
Cayley - Hamiltonov izrek
sled, diskriminanta
uporaba v analitični geometriji

--- 20 % ---

8. Fourierjeve vrste

vektorski prostor kvadratno integrabilnih funkcij
ortogonalne množice
aproksimacija s končnimi vsotami
pojem polne ortogonalne množice
splošne Fourierjeve vrste
trigonometrične funkcije kot poseben primer
dokaz polnosti

--- 10 % ---

MODUL 1.3

Matematična analiza (3-0-2)

Cilji

Doseže naj se delovno znanje iz osnov teorije funkcij, integralov in diferencialnih enačb pri funkcijah ene in več spremenljivk

Poudarek modula

Modul je osnoven, ker temeljijo na njem ostali matematični moduli. Snov naj vsebuje veliko število primerov v obliki domačih nalog. Na koncu modula naj se znanje preizkusi tako, da se opravljajo navadne operacije integriranja, odvajanja in reševanje diferencialnih enačb.

Vsebina modula

1. Realne funkcije z več spremenljivkami

limite in zveznost
diferencial funkcije več spremenljivk
verižno pravilo
delni odvodi
gradient
zakon povprečja, maksima in minima
implicitne funkcije
Lagrangeovi multiplikatorji
Taylorjeve vrste
aproksimacije
uporaba, primeri

--- 10 % ---

2. Limite zaporedij

iterativni postopki
Newtonova metoda
konvergenca monotonih zaporedij
neskončna zaporedja realnih in kompleksnih števil
absolutna konvergenca
potenčne vrste realnih in kompleksnih števil
funkcijske razširitve
rešitve vrst
nedeterminirane oblike

--- 20 % ---

3. Večkratna integracija

definicija integrala, eksistence kot ploščine, volumna, mase, povprečja
numerični izračun
centroidi, momenti
izračun s ponavljanimi enostavnimi integrali
cilindrične in sferične koordinate

--- 10 % ---

4. Metode za diferenciranje in integriranje

zapleteni problemi diferenciranja z implicitnimi funkcijami itn.
integracija s substitucijo, s parcialnimi ulomki in z uporabo tabel
inverzne trigonometrične funkcije
nepravi integrali
maksimi in minimi funkcij ene in več spremenljivk
Lagrangeovi multiplikatorji

5. Linearne diferencialne enačbe

enačbe oblike Mdx + Ndx = 0
natančni diferenciali
integrirni faktorji
linearne diferencialne enačbe s konstantnimi koeficienti
enačbe reda n
sistemi diferencialnih enačb
linearne diferencialne enačbe s spremenljivimi koeficienti
posebni problemi
zakoni gibanja, ohranitve energije
problem dveh teles

--- 20 % ---

6. Diferencialne enačbe

tangentna polja
pomen rešitve krivulj
Picardova metoda določitve eksistence
numerična koračna rešitev
sistemi $y'_1 = f_i(x, y_1 \ldots, y_n)$ in numerična rešitev
posebni primer $y' = f(x)$
posebne numerične metode
trapecoidno pravilo
Simpsonovo pravilo
enačbe oblike $Mdx + Ndy = 0$
ločitev spremenljivk
natančni diferenciali
integrirni faktorji

--- 20 % ---

MODUL 2.1

Človekovo in organizacijsko obnašanje (2-1-2)

Poudarek modula

Uvod v principe upravljanja človekovega obnašanja, še posebej v povezavi z organizacijami, kjer se bodo uvajali računalniško podprti informacijski sistemi.

MODUL 2.3

Teorija informacije in kodiranje (2-0-1)

Cilji

Pokažejo naj se pristopi k problemom zanesljivega prenosa sporočil skozi nezanesljive kanale in preučujejo naj se najbolj uporabljani kodi za zaznavanje in popravljanje napak.

Poudarek modula

Osnove teorije informacije so pretežno matematične narave. V predavanjih naj se ostaja pri osnovnih, formalnih definicijah in na praktični uporabi linearnih in cikličnih kodov. V modulu niso poudarjene sodobne metode signalne analize in obdelave, ker bi to zahtevalo višjo stopnjo matematike in ker sega ta problematika bolj v področje telekomunikacij kot informatike.

Vsebina modula

1. Informacijski viri

definicija informacije
informacijski vir brez pomnilnika
entropija in njene lastnosti
razširitev brezpomnilnega vira
informacijski vir Markova
povezani vir
razširitve Markovljevih virov
zgradba jezika

--- 20 % ---

2. Lastnosti kodov

enolično dekodljivi kodi
večlični kodi
Kraftova neenačba
McMillanova neenačba
Shannon - Fanovi kodi

--- 10 % ---

3. Kodiranje informacijskih virov

povprečna dolžina koda
prvi Shannonov izrek
primer Markovljevega vira
Huffmanovi kodi
r - narno kompaktni kodi
kodna učinkovitost in redundanca

--- 20 % ---

4. Kanali in vmesna informacija

informacijski kanali
verjetnostne relacije v kanalu
apriorne in posteriorne entropije
vmesna informacija
brezšumni in deterministični kanali
kaskadni kanali
aditivnost medsebojne informacije
zmogljivost kanala
pogojna vmesna informacija

--- 20 % ---

5. Zanesljiva sporočila pri nezanesljivih kanalih

verjetnost napak in odločitvena pravila
Fanov šop
zaznavanje in popravljanje napak
Hammingova razdalja
drugi Shannonov izrek

--- 20 % ---

6. Ciklični kodi

definicija cikličnega koda
generiranje polinomov
divizorji za $x^n + 1$
relacija med Hammingovimi in cikličnimi kodi
kodiranje in dekodiranje z linearnimi pomikalnimi registri

--- 10 % ---

MODUL 2.5

Verjetnost in statistika (3-0-2)

Cilji

Študent naj pridobi delovno znanje o konceptih, ki se uporabljajo v verjetnostnem računu in statistiki. To znanje ima osnovni pomen v teoriji in praksi računalniških znanosti (operacijske raziskave, simulacija, modeliranje itd.).

Poudarek modula

Ker je snov modula težavna, jo je potrebno obravnavati rigorozno. To pa zahteva abstraktno predstavitev s praktičnimi primeri povsod tam, kjer je to mogoče. Obdela naj se veliko število vaj v obliki domačih nalog s praktičnimi in realnimi podatki.

Vsebina modula

1. Verjetnost kot matematični sistem

vzorčni prostori
dogodki in podmnožice
verjetnostni aksiomi
vzorčni izreki
končni vzorčni prostori in mere
binomski koeficienti in števne metode v verjetnostnih problemih
pogojna verjetnost
neodvisni dogodki
Bayesova formula

--- 20 % ---

2. Naključne spremenljivke in njihova porazdelitev

naključne spremenljivke (diskretne in zvezne)
verjetnostne funkcije
gostotne in porazdelitvene funkcije
posebne porazdelitve (binomska, hipergeometrična, Poissonova, uniformna, eksponenčna, normalna itd.)
povprečje in varianca
Čebiševljeva neenačba
neodvisne naključne spremenljivke
funkcije naključnih spremenljivk in njihove porazdelitve

--- 20 % ---

3. Limitni izreki

Poissonova in normalna aproksimacija
središčnolimitni izrek
zakon velikih števil
statistične aplikacije

--- 10 % ---

4. Statistična interferenca

ocenjevanje in vzorčevanje
točkovne in intervalne ocene
hipotezno preizkušanje
stopnja preizkusa
regresija
primeri neparametričnih metod

--- 20 % ---

5. Verige Markova

prehodne verjetnosti in matrika
klasifikacija stanj
ergodične lastnosti
problem naključnega gibanja
primeri iz fizike, biologije in vedenja

--- 20 % ---

6. Stohastični procesi

tipi procesov
Markovljevi procesi
uporaba v teoriji vrst

--- 10 % ---

MODUL 2.7

Numerične metode I (3-3-2)

Cilji

Študent naj bi spoznal osnovne algoritme numeričnih izračunov, teoretične osnove algoritmov in probleme, ki so povezani z implementacijo algoritmov.

Poudarek modula

Snov je povezana z razpravo o algoritmih, s pripadajočo teorijo ter z obravnavo prednosti in pomanjkljivosti metod. Rešitve realnih problemov naj bodo ponazorjene s programi, tako da nimamo zgolj kodiranja raznih algoritmov. Konvergenca in analiza napak za določene algoritme naj se obravnavata teoretično. V oba tečaja naj bo vključena aritmetika s pomično vejico in uporaba matematičnih subrutinskih paketov v povezavi s specifičnimi problemi. Ostala snov predmeta naj se pokrije zaporedno. Globina podajanja snovi se lahko spreminja, obravnava pa naj se vsa ali večina predložene snovi.

Vsebina modula

1. Aritmetika s pomično vejico

koncepti številskega sistema s pomično vejico
posledice zaradi končne natančnosti
prikaz napak zaradi zaokrožitve

--- 15 % ---

2. Uporaba paketa matematičnih subrutin

--- 10 % ---

3. Interpolacija

račun končnih razlik
polinomska interpolacija
inverzna interpolacija
rezinasta interpolacija

--- 15 % ---

4. Aproksimacija

uniforma
z diskretnimi najmanjšimi kvadrati
polinomska
Fourierjeva aproksimacija
Čebiševljeva ekonomizacija

--- 10 % ---

5. Numerična integracija in diferenciacija

interpolacijska numerična integracija
Euler - McLaurenova formula
Gaussova kvadratura
adaptivna integracija
hitri Fourierjev transform
Richardsonova ekstrapolacija in numerična diferenciacija

--- 15 % ---

6. Reševanje nelinearnih enačb

bisekcija
iteracija s fiksno točko
Newtonova metoda
Aitkenov postopek
konvergenčne stopnje
učinkovit izračun polinomov
Bairstowova metoda

--- 15 % ---

7. Reševanje navadnih diferencialnih enačb

metode s Taylorjevimi vrstami
Eulerjeva metoda z lokalno in globalno analizo napak
metode Runge - Kutta
napovedovalne / korekturne metode
avtomatično ugotavljanje napake:
sprememba koraka in stopnje
stabilnost

--- 20 % ---

MODUL 3.1

Sistemski koncepti in implikacije (2-0-2)

Cilji

a) priprava študenta na sistemski pristop in organizacijo
b) poučevanje osnovnih konceptov, ki se uporabljajo pri modeliranju sistemov

Poudarek modula

Modul naj se izvaja kot zaporedje predavanj, ki so prepletena s primeri in študijem primerov. Ti primeri so lahko iz področja ekologije, medicinskih uslug, trans-

porta, proizvodnih sistemov itn. Cilj modula so praktične izkušnje in preveliko poglabljanje v teorijo formalnih sistemov ni priporočljivo.

Vsebina modula

1. Koncept sistema

stanja
preslikava, vhodi, izhodi
hierarhična zgradba
sistemi s kompleksnimi, nasprotujočimi, različnimi cilji: metode rešitev
sistemske omejitve
odprti in zaprti sistemi
odprti sistemi: lastnosti, negativna entropija, prilagajanje
elementi: podsistemi, podenote, sestavni deli
povezave
podsistemi: neodvisnost/odvisnost, metode odvezovanja
podoptimizacija, stranski učinki
deterministični sistemi: verjetnosti sistemi
zamisel povratne povezave: maksimiranje, zadovoljevanje, prilagojevanje, nastavitev na spremembe
krmiljenje sistema: standard kot napoved izhoda, povratna povezava (odprta in zaprta zanka)
cena krmiljenja
splošna teorija sistemov

--- 20 % ---

2. Definiranje sistema

modeli kot predstavitve sistemov
zapleteni in enostavni modeli
formalni in neformalni sistemi: interakcija med njimi
sistemska zgradba: alternativna zgradba
problem identifikacije
pripomočki: blokovni diagrami, grafi poteka, odločitvene tabele
stopnje sestavljanja sistemov

--- 10 % ---

3. Sistemska analiza

izbira najboljše akcijske smeri med več možnimi: prednosti in pomanjkljivosti, dobiček in cena
določitev ustreznih elementov, povezav in postopkov za doseganje ciljev
cilji, alternative, cena/dobiček, kriteriji
modeliranje
sistemsko načrtovanje: izboljšava obstoječega sistema, razvoj novega sistema
postopek sistemskega načrtovanja:
  identifikacija problemov in definicije alternativne rešitve
    izbira rešitve
    sinteza predloženega sistema
    testiranje sistema
    čiščenje sistema
sistemska optimizacija

--- 10 % ---

4. Upravljalski sistemi

hierarhična zgradba
interakcije med podenotami in v podenotah
človek kot element sistema
upravljalski sistem:
  operacijski sistem
  odločitveni sistem
  krmilni sistem
  časovne odvisnosti
  informacijski pretoki
informacijski sistemi za upravljalske sisteme (kot podsistemi v upravljalskem sistemu)
elementi informacijskega sistema:
  upravniki
  računalniška materialna in programska oprema
  komunikacijske mreže
  baze podatkov itn.
funkcionalni sistemi:
  računovodska pooblastila
  inventura itn.

--- 30 % ---

5. Upravljalski informacijski sistemi

pomen informacijskega sistema v organizaciji
razlike v informacijskih potrebah glede na klasično zgradbo organizacije
povezava med človekom in sistemom:
  sistemi človek/stroj
operacijski in proizvajalni postopki v informacijskih sistemih
razlika med logičnimi in fizičnimi sistemi
načrtovanje informacijskih sistemov
metode za razvoj informacijskih sistemov

--- 30 % ---

MODUL 3.6

Operacijske raziskave (2-0-2)

Cilji

Študent naj pridobi znanje o glavnih metodah operacijskega raziskovanja. Te metode so daljnosežne, saj so operacijske raziskave sistematičen in racionalen pristop k fundamentalnim problemom, kot je upravljanje sistemov z odločanjem, kjer se dosegajo najboljši rezultati v odvisnosti od razpoložljive informacije.

Poudarek modula

Večina snovi o operacijskih raziskavah obravnava analizo matematičnih modelov pa tudi analizo problemov. Modul naj uvede določeno ravnovesje, ker modeli sicer so uporabni, predstavljajo pa še vedno le poenostavljeno sliko realnih razmer, kjer se pojavlja največja težavnost reševanja problemov, ki ne sme biti podcenjevana.

Vsebina modula

1. Narava operacijskih raziskav

opredelitev problema (narava problema, opredelitveni cilji, sistemska analiza)
tipi problemov
  položaji tveganja
  maksimiranje učinkovitosti
  maksimum itn.
zgradba modelov
modelni približki
sekvenčni odločitveni modeli
izpeljevanje rešitev iz modelov
simulacija
procedure vzorčevanja in vrednotenja

--- 20 % ---

2. Problemi dodeljevanja

transportni problem
prireditveni problem
simpleksna metoda
dualnost
parametrično programiranje

praktični primeri

--- 20 % ---

3. Inventurni problemi

inventurni problemi (narava, kontekst in zgradba)
deterministični problem enega kosa (posameznosti), ene ravnine
večkosovni deterministični problem, ena ravnina
verjetnostni problemi
praktični primeri

--- 20 % ---

4. Problemi zamene, vzdrževanja in zanesljivosti

kapitalne naprave
stroški popusta
zamena ob pričakovanju napake
skupinska zamena
splošen postopek obnove
zanesljivost

--- 20 % ---

5. Dinamično programiranje

zamisel optimalnosti
odločitvena drevesa
deterministični problemi

--- 10 % ---

6. Sekvenciranje in koordinacija

PERT
kritične poti

--- 10 % ---

MODUL 3.7

Teorija razvrščanja (2-0-1)

Cilji

Uvajajo se osnovne metode in rezultati teorije razvrščanja.

Poudarek modula

Teorija razvrščanja je del operacijskih raziskav. Zaradi svoje pomembnosti pri modeliranju in simulaciji najnovejših sistemov z delitvijo časa, multiprocesiranjem, sistemov za obdelavo podatkov z več posli (nalogami), kot pri modeliranju računalniških mrež se ta snov obravnava v okviru posebnega modula. Snov naj dobi svojo matematično predstavitev z uporabo v realnih sistemih, z vrsto domačih nalog za študenta, saj imajo že enostavni problemi razvrščanja dokaj zapletene rešitve.

Vsebina modula

1. Uvod

opredelitev in mere sistemov razvrščanja
zgradba osnovnih sistemov razvrščanja
definicija in klasifikacija stohastičnih procesov
Markovljeve verige diskretnega časa
zvezne verige Markova
procesi rojstva/smrti

--- 10 % ---

2. Sistemi razvrščanja tipa rojstvo/smrt

klasični sistemi razvrščanja
prestrašeni prihajajoči
neomejeno število strežnikov (M/M/oo)
primer m strežnikov (M/M/m)
končni pomnilnik (M/M/1/K)
končna uporabniška populacija z enim strežnikom (M/M/1/M)
končna uporabniška populacija z neomejenim številom strežnikov (M/M/oo/M)
končna uporabniška populacija, m-strežnik, končni pomnilnik (M/M/m/K/M)

--- 10 % ---

3. Markovljeve vrste

enačbe ravnotežja
metoda ogrodij
Erlangova porazdelitev
vrsta $M/E_r/1$
vrsta $E_r/M/1$
obsegovni prihajajoči sistem
obsegovni uslužnostni sistem
mreža Markovljevih vrst

--- 30 % ---

4. Vrsta M/G/1

prehodne verjetnosti
povprečna dolžina vrste
porazdelitev števila v sistemu
porazdelitev čakalnega časa
delovna perioda in njeno trajanje
število, ki je postreženo v delovni periodi

--- 10 % ---

5. Meje, neenačbe in približki

približek pri močnem prometu
gornje in spodnje meje za povprečno čakanje
diskretni približki
tekoči približek
približek v napadalnem času

--- 10 % ---

MODUL 3.8

Numerične metode II (2-3-2)

Cilji in poudarek modula so opisani v modulu 2.7 (numerične metode I).

Vsebina modula

1. Aritmetika plavajoče vejice

osnovne zamisli številskih sistemov s plavajočo vejico
posledice končne natančnosti
prikaz napak zaradi zaokroževanja

--- 15 % ---

2. Uporaba paketov matematičnih subrutin

--- 10 % ---

3. Neposredne metode za linearne sisteme enačb

Gaussova izločitev
operacijski račun
implementacija (tečaji in lestvice)
neposredne metode s faktoriranjem

--- 20 % ---

4. Analiza napak in norme

vektorske in matrične norme
pogojna števila in ocenitve napak

iterativne izboljšave

--- 15 % ---

5. Iterativne metode

Jacobijeva metoda
Gauss-Seidelova metoda
pospešitev iterativnih metod
nadsprostitev

--- 15 % ---

6. Izračun lastnih vrednosti in lastnih vektorjev

osnovni izreki
ocenitve napak
potenčna metoda
Jacobijeva metoda
Householderjeva metoda

--- 15 % ---

7. Sorodna problematika

numerična rešitev problemov mejnih vrednosti za navadne diferencialne enačbe
rešitev nelinearnih sistemov algebrajskih enačb
rešitev naddeterminiranih sistemov z metodo najmanjših kvadratov

--- 10 % ---

MODUL 4.2

Simulacija in modeliranje
(modul še ni bil definiran)

MODUL 4.6

Matematična logika (2-0-2)

Cilji

Študent se uvede v zamisli izračunljivosti in odločitvenosti.

Poudarek modula

Čeprav je teorija izračunljivosti veja čiste matematike, je pa tudi del teorije o digitalnih računalnikih. Zaradi tega naj se uvedejo Turingovi stroji kot modeli univerzalnih računalnikov, ki jim sledi teorija rekurzivnih funkcij. V nadaljevanju modula naj se pokaže obstoj absolutno nerešljivih problemov.

Vsebina modula

1. Splošna teorija izračunljivosti izračunljive funkcije

Turingovi stroji
izračunljive in delno izračunljive funkcije
relativno izračunljive funkcije
operacije na izračunljivih funkcijah
kompozicija in minimalizacija
rekurzivne funkcije
primitivne rekurzivne funkcije
Goedelovo preštevanje
splošne rekurzivne funkcije
Goedelov izrek
parcialno rekurzivne funkcije
Churcheva teza
nerešljivi odločitveni problemi
polizračunljivi predikati
odločitveni problemi
rekurzivno preštevne množice

--- 60 % ---

2. Uporaba splošne teorije

kombinatorni problemi
kombinatorni sistemi
Turingovi stroji
semithuevi sistemi
Thuevi sistemi
besedni problem za polgrupe
normalni in Postovi sistemi
matematična logika
logika
izreki nepolnosti in nerešljivosti logike
aritmetična logika
logika prvega reda
delno izjavni računi

--- 40 % ---

MODUL 5.1

Družbeno in kulturno okolje

Cilji

Preučuje naj se, kako lahko družbeno in kulturno okolje vplivata na način računalniške uporabe v dani družbi in kakšen je možni družbeni in kulturni učinek na splošno rabo računalnikov v dani družbi.

MODUL 5.2

Oblikovanje in realizacija sistemov (3-1-3)

Cilji

Poudarja naj se informacijska analiza ter logično in fizično oblikovanje sistemov.

Poudarek modula

Oba dela tega modula naj se poučujeta hkrati in ne zaporedno. Poudarek bodi na iterativni naravi analize in oblikovalnega postopka.

Vsebina modula

1. Informacijska analiza

a) Uvod v sistemski življenski cikel
pregled posameznih faz sistemskega razvoja in njihovih relacij
zamisel, informacijska analiza
oblikovanje sistema
programiranje, dokumentacija
instalacija, prevrednotenje

--- 2,5 % ---

b) upravljanje sistemskega življenskega cikla
vođenje projekta za sistemski razvoj
ravnine zapletenosti pri sistemskem oblikovanju
odgovornosti sistemskih analitikov, sistemskih načrtovalcev, programerjev, operaterjev in vodstva obdelave podatkov
organizacijski in vedenjski učinki sistemskega oblikovanja in realizacijskih pristopov

--- 7,5 % ---

c) osnovni analitični pripomočki
analizni koraki
predhodne raziskave
študija splošne primernosti
splošni sistemski predlog
podrobna analiza
analitične metode
dogodkovno usmerjeni organizacijski diagrami poteka
odločitvene tabele
prednostna mrežna analiza

--- 10 % ---

č) opredelitev sistemskih alternativ
- ročni sistemi
- primerjava ročnih z avtomatičnimi deli sistemov
- interaktivne zahteve upravljalskega računalnika, odziv, zmogljivost, jezik, vključno naravni
- splošni in prirejeni izhod (grafični, tekstni, slušni, avtomatično poročanje, razpoznavanje podatkov itn.)
- gradnja in razgradnja podatkov in materialne opreme
- določitev elementov skupnih podatkovnih baz
- alternative podatkovnega upravljanja
- odzivne potrebe glede na ekonomične, materialne/programske in organizacijske zahteve
- programirano odločanje

--- 10 % ---

d) opredelitev sistemske gospodarnosti
- stroški in vrednost informacije
- mere sistemske zmogljivosti, cene, odziva, natančnosti, zanesljivosti, prožnosti, varnosti, kvalitete, učinkovitosti, kapacitivnosti
- identifikacija in kvantifikacija sistemskih stroškov ter stroškov osebja, naprav, konverzije in instaliranja
- identifikacija, kvantifikacija in meritve sistemskih prednosti (posredne in neposredne koristi)
- analiza izboljšane kakovosti informacije
- dodelitev stroškov in cenitev računalniških uslug

--- 10 % ---

e) določitev potreb logičnega sistema
- format stavka za sistemske potrebe
- ločitev logičnega (sistemskega) in fizičnega načrtovanja (zbirke, programi, procedure)
- izhodne sistemske potrebe, operacijska ravnina, vodenje na prvi ravnini, srednje upravljanje, izvršno upravljanje
- informacija za strateško in taktično planiranje in odločanje
- specifikacija izhodnih metod in formatov
- sistemska dokumentacija
- metode sistemske specifikacije, ročne in polavtomatske metode

--- 10 % ---

2. S i s t e m s k o o b l i k o v a n j e

a) osnovni oblikovalni pripomočki in cilji
- pregled sistemskega življenjskega cikla
- dokumentacija za različne ravnine oblikovanja
- cilji sistemskega oblikovanja
- sistemska celovitost
- tipi sistemskega oblikovanja (paketno, interaktivno)
- proračun in upravljanje projekta

--- 5 % ---

b) izbira materialne in programske opreme in vrednotenje
- izbira naprav
- vrednotenje materialnih in programskih zahtev
- metode avtomatskega vrednotenja
- simulacija, analitični modeli
- podrobna analiza stroškov (za osebje, materialno in programsko opremo)
- tekmovalni poudarek

--- 7,5 % ---

c) oblikovanje in tehnika programske opreme
- oblikovalna modularnost
- oblikovanje uporabniških vmesnikov z avtomatiziranimi procedurami
- standarizacija podsistemskega oblikovanja
- urejevanje, obdelava in razpoznavanje podatkovnih zbirk
- oblikovanje podsistemskih vmesnikov
- podatkovno in proizvodno krmiljenje
- zunanje in notranje obračunavanje v sistemu
- konverzijski podsistemi
- človeški inženiring

--- 10 % ---

č) razvoj podatkovnih baz
- zgradba podatkovne baze
- njeno oblikovanje, vzdrževanje in uporaba
- celovitost podatkovne baze
- sistemi upravljanja podatkovnih baz

--- 7,5 % ---

d) razvoj programov
- izbira jezikov
- uporaba standardnih gradbenih blokov in skupnih programov
- popravljanje napak
- trdnost programov
- programirni standardi in dokumentacija

--- 7,5 % ---

e) izdelava sistema
- preizkušanje in popravljanje napak
- planiranje in izvajanje konverzije
- vodenje programiranja
- preizkušanje in instalacija
- koordinacija med ročnimi in avtomatiziranimi procedurami
- načrt za izdelavo

--- 7,5 % ---

f) naknadna analiza izdelave
- preverjanje sistemskih zmogljivosti, stroškov razvoja in naporov
- ovrednotenje materialnih in programskih zmogljivosti
- časovniški sistemi
- cikel preoblikovanja, modifikacije sistema
- celovite spremembe v sistemu, ki obratuje

--- 5 % ---

MODUL 5.5

Oblikovanje in upravljanje podatkovnih baz (3-3-2)

Cilji

a) prikaže naj se potreba po podatkovnih bazah
b) poudarijo naj se zamisli in zgradbe, ki so potrebne pri oblikovanju in izdelavi upravljalskega sistema podatkovnih baz

Poudarek modula

Poudari naj se razumevanje odnosov med organizacijo fizične zbirke in metodami podatkovnega strukturiranja. Koncepti podatkovnih modelov naj pokažejo tudi mreže ter relacijske in hierarhične modele. Obravnavajo naj se primeri specifičnih upravljalskih sistemov podatkovnih baz v primerjavi s prikazanimi podatkovnimi modeli. Preučujejo naj se metode podatkovne celovitosti in varnosti. Glavna izkušnja tega modula naj bo oblikovanje in izdelava enostavnega upravljalskega sistema podatkovne baze, ki naj vsebuje varnost zbirk in določeno obliko vpraševanja v sistem.

Vsebina modula

1. Upravljalski sistemi podatkovnih baz

problem: velike zbirke podatkov, ki so deljene med več različnih uporabniških programov
specifični problemi podatkovne neodvisnosti, zanesljivosti, prilagodljivosti, celovitosti, obnavljanja, zmogljivosti in preglednosti
zgradba podatkovnega upravljalskega sistema
povezava med aplikativnimi programi in podatkovnim upravljalskim sistemom
procedurne in neprocedurne povezave
vase zaključeni sistemi in sistemi, razširjeni z gostiteljskim jezikom
uporabniški podatkovni sistemi: hierarhične mreže, relacijski modeli
pomen administracije podatkovnih baz
podrobnosti enega ali dveh primerov sistemov podatkovnih baz

--- 20 % ---

2. Seznam primerov tipičnih sistemov podatkovnih baz

--- 5 % ---

3. Uvod: pregled osnovnih zamisli

sistemi za obdelavo informacij
shranjevanje podatkov in podatkovni sistemi za razpoznavanje
podatkovni upravljalski sistemi
upravljalski sistemi podatkovnih baz
upravljalski informacijski sistemi
računalniška organizacija
  CPE in centralni pomnilnik
  pomnilne naprave z vrtenjem
  masovni pomnilniki
definicije
  zapisi, zbirka
  podatkovna baza
  podatkovni model in podmodel
  ključi
zgodovina upravljalskih sistemov podatkovnih baz (USPB)
  zapis, tekstualni, numerični primer

--- 20 % ---

4. Funkcije in sestavni deli USPB

okolica
  materialna in programska oprema
  uporabniki
  vodenje in dogovori
generiranje podatkovne baze
  informacija in dekodiranje
  surovi podatki
  podatkovna definicija
  podatkovna zgradba
  pomnilniška zgradba
  reorganizacija podatkovne baze
povpraševanje
  zaključene možnosti
  možnosti gostiteljskih jezikov
  knjižnice opravil
obnavljanje
  dodajanje, brisanje, spreminjanje
podatkovni model
  mreža, hierarhija
  relacija
politika varnosti
  zasebnost
  kvaliteta in celovitost
  zaščitni mehanizmi
glavne dejavnosti v USPB
  upravljanje podatkovnih baz
  povpraševanje
  obnavljanje
  pomnilniški prostor
  prožnost
  vidiki zaščite in varnosti

--- 25 % ---

5. Oblikovanje zbirk in poti dostopa

zaporedne (sekvenčne) zbirke
indeksno zaporedne zbirke
liste in obroči
večkratne liste
sekljanje
imeniki in obrnjene liste
informacijski razpoznavni sistemi
  celične, obrnjene zgradbe
  prepletene zgradbe

--- 15 % ---

6. Primerjava s komercialnimi sistemi

TOTAL
sistem 2000
IMS/VS
IDS, IDMS
ADABAS
DMS II

--- 10 % ---

7. Prihodnje usmeritve USPB

zelo velike podatkovne baze
geografska in organizacijska porazdelitev logičnih naprav in komponent
integracija asociativnih procesorjev in pomnilnikov
računalniki za upravljanje podatkovnih baz
migracija območij
umetna inteligenca in zapletene sheme indeksiranja

--- 5 % ---

MODUL 5.6

Oblikovanje in razvoj programov (3-3-2)

Cilji

Modul uvaja pregled postopkov in metod za opredelitev in upravljanje velikih programirnih projektov.

Poudarek modula

Ta modul predstavlja formalni pristop k izkustvenim postopkom oblikovanja in razvoja programske opreme in razlaga študentu pripomočke za uporabo. Študentje naj delajo v skupinah, ko se praktično seznanjajo z organizacijo, upravljanjem in razvojem velikih programirnih projektov. Vidiki skupinskih projektov naj predvidijo delo v ločenih laboratorijih in določen čas za razprave o skupinskih projektih.

Vsebina modula

1. Standardi

zgradba
kje in kako začeti
problem priročnika za standarde
navzdolnji razvoj paketa standardov

--- 5 % ---

2. Organizacija

izdelava in kontrola standardov
minimizacija obveznih standardov
definiranje pregleda in politike standardov
planiranje in nadzor izdelovalnega postopka (programa)
odvisnost prednosti standardov od časa in kakovosti
metode uveljavljanja
funkcija koordinatorja knjižnice in standardov za projektni urad in podporo
pregled in vzdrževanje standardov
posebni problemi velikih, zapletenih, naprednih in mednarodnih projektov

--- 20 % ---

3. Standardi projektnega nadzora

kontrolne točke
kontrola s končnim dogodkom
ocenjevalni pregledi vmesne in končne stopnje
dokumentacija projektnih pogodb

--- 15 % ---

4. Standardi projektnih skupin

organizacija skupine
cilji
število in odgovornosti
opisi opravil
zahteve po šolanju
pomožni projektni standardi
varnost
ponavljanje in obnavljanje
specifikacija okolice
materialna in programska oprema
zunanje usluge
standardi za preglede in poročanje
definicija vloge projektnega knjižničarja oziroma koordinatorja za oblikovanje in vzdrževanje dokumentacije

--- 20 % ---

5. Izdelava dokumentacije

poteze projektne dokumentacije, ki se prilagaja obliki projekta
prilagajanje kreiranja dokumentacije razvojnim in izdelovalnim aktivnostim
minimizacija dokumentacijskih zahtev
optimizacija kakovosti
spreminjanje metod skladno z obstoječimi nivoji spretnosti

--- 20 % ---

6. Projektni tehnični standardi

metodologija in tehnični standardi za aktivnosti v velikih projektih
vodenje standardov

--- 10 % ---

7. Sistemski priročnik projekta

projektni standardi in dogovori
specifikacija sistema
podrobne sistemske definicije
oblikovanje in uporabniške procedure
instalacijski in operacijski priročniki
vzgojne zahteve
pregled (seznam)

--- 15 % ---

MODUL 5.7

Teorija formalnih jezikov in avtomatov (2-0-2)

Cilji

Študent naj si pridobi znanje na področju teoretičnih računalniških znanosti in spozna njihovo hierarhično prepletenost. Ta modul daje tudi teoretične dokaze metod, ki se uporabljajo v jezikovnih prevajalnikih.

Poudarek modula

Uvajajo se osnovni rezultati formalnih modelov računalništva. Poudarek je na razvijanju študentskih veščin ter v razumevanju strogih definicij in v opredeljevanju njihovih logičnih posledic. V tem pomenu je tudi poudarek na določenih problemih.

Vsebina modula

1. Formalne gramatike in avtomati

produkcijski sistemi in jeziki
regularne, kontekstnosvobodne, kontekstno občutljive in rekurzivne gramatike
sprejemniki: končni avtomati, odlagalni avtomati, linearno omejeni avtomati, Turingovi stroji
sintaksna in izvajalna drevesa
Church - Rosserjev izrek
osnovni problemi jezikovne teorije:
opis, razpoznavanje
odločitveni problemi
frazno strukturirane gramatike in naravni jeziki
urejene gramatike

--- 20 % ---

2. Regularni jeziki

regularne množice in izrazi
deterministični in nedeterministični končni avtomati
ekvivalenca deterministčnih in nedeterminističnih končnih razpoznavnikov
Kleenejeva karakterizacijska teorija za množice, ki jih sprejemajo končni avtomati
odvodi regularnih izrazov
dvosmerni končni avtomati
končni avtomati, stroji s končnim številom stanj in njihove relacije s kombinatornimi preklopnimi vezji
kompleksnost vezij
določeni dogodki in stroji s končnim pomnilnikom
ekvivalenca stanj in minimizacija stanj v končnih avtomatih
algebrska dekompozicija in teorija strukture
Hartmans-Stearnsova strukturalna teorija
polgrupe in Krohn-Rhodesova dekompozicija

--- 20 % ---

3. Kontekstno svobodni jeziki

Chomskyjevi in Griesbachovi izreki normalnih form
samovgnezdenje, neenoumnost
ekvivalenca kontekstno svobodnih jezikov in množic, ki jih sprejmejo nedeterministični odlagalni avtomati
lastnosti zaprtja kontekstno svobodnih jezikov
hitro razpoznavanje kontekstno svobodnih jezikov
Earlyjevi algoritmi
Valiantov $n^{2,8}$ razpoznavni algoritem
deterministični odlagalni avtomati in stavčna analiza
deterministični kontekstno svobodni jeziki
omejitve odlagalnih avtomatov

--- 20 % ---

4. Deterministična analiza kontekstno svobodnih jezikov

pomikalno redukcijska analiza
navzdolnja analiza
LR (k) gramatika
LL (k) gramatika
prednostne gramatike
enostavna in operatorska prednost
šibka in mešano strategijska prednost
kontekstno omejene gramatike
relacije med determinističnimi kontekstno svobodnimi jeziki

--- 30 % ---

5. Rekurzivni jeziki

večtračni Turingovi stroji in pripadajoči formalizmi za razpoznavanje
nerešljivost ustavitvenega problema
redukcija Postovega korespondenčnega problema na ustavitveni problem
neodločljive lastnosti gramatik

--- 10 % ---

MODUL 6.1

Funkcionalnost in vrednotenje (2-0-1)

Cilji

Študent naj se dobro seznani z modeliranjem računalniških sistemov tako, da razume pristope optimalne uporabe teh sistemov. Slovenska beseda za funkcionalnost je tudi zmogljivost.

Poudarek modula

Po uvedbi zamisli modela sistema za obdelavo podatkov naj se uvedejo metode gradnje modelov. Ti modeli se uporabljajo za optimiziranje in napoved obnašanja sistema v različnih okoljih. Prikaže naj se več primerov, ki temeljijo na danih napravah.

Vsebina modula

1. Osnovne zamisli gradnje modelov

uporaba modela za vrednotenje in napovedovanje
kaj je gradnja modela
specifični problemi modeliranja računalniških sistemov
monitorji za materialno in programsko opremo
metode modelne analize:
  operacijska analiza
  modeli z enostavnimi vrstami
  mreže vrst
  simulacija diskretnih dogodkov
uporaba:
  oblikovanje arhitekture
  izbira konfiguracije
  napoved zmogljivosti

--- 20 % ---

2. Gradnja modelov z operacijsko analizo

zamisli operacijske analize
pojem sistema in zahtev
osnovni kriteriji
osnovne operacijske relacije
interpretacije merjenih rezultatov
analiza pojava zasičenosti
meje operacijske analize
verjetnostni modeli

3. Modeli, ki temeljijo na razvrščanju

mreže vrst
osnovni rezultati
analiza zasičenja
hierarhični modeli
približne metode
uporaba v multiprogramskih in multiprocesorskih sistemih

--- 30 % ---

4. Diskretni simulacijski modeli

zamisel
specifični problemi
vhodno generiranje
natančnost rezultatov in interval zaupanja
primeri

--- 20 % ---

MODUL 6.2

Prevajalniki in njihovo oblikovanje (3-3-2)

Cilji

Pojasnjuje se metode, ki so povezane z analizo izvirnih jezikov in z generiranjem učinkovitega objektnega koda.

Poudarek modula

Čeprav mora biti obravnavana tudi teoretična snov, naj modul poučuje študenta, kako so lahko zgrajeni prevajalniki. Programiranje naj obsega izdelavo prevajalniških komponent ter oblikovanje preprostega, toda popolnega prevajalnika, ki ima obliko skupinskega projekta.

Vsebina modula

Modul vsebuje več materiala kot ga je moč v predavanjih smiselno uporabiti, tako da je določena izbira snovi še potrebna.

1. Pregled nekaterih metod

pregled zbirnih metod, metod oblikovanja simbolnih tabel in makrojev
pregled sintaksne analize in drugih oblik programskega razpoznavanja
pregled prevajanja, nalaganja in izvajanja s poudarkom na predstavitvi programov v nalagalnem jeziku

--- 10 % ---

2. ___ ___ ___

metode enoprehodnega prevajanja
prevajanje aritmetičnih izrazov iz postfiksne oblike v računalniški jezik
učinkovita uporaba registrov in začasnega pomnilnika

--- 5 % ---

3. ___ ___ ___

dodeljevanje pomnilnika konstantam, začasnim spremenljivkam, poljem
začasni pomnilnik
funkcijske in stavčne procedure
neodvisna blokovna struktura
vgnezdena blokovna struktura
dinamično dodeljevanje pomnilnika

--- 10 % ---

4. ___ ___ ___

objektni kod za indeksirane spremenljivke
funkcije pomnilniških preslikav

obveščevalni vektorji
prevajanje s sekvencioniranjem stavkov

--- 5 % ---

5. ___ ___ ___

podrobna organizacija enostavnega popolnega prevajalnika
simbolne tabele
leksikalno pregledovanje vhoda (razpoznavnik)
sintaksno pregledovanje (analizator)
generatorji objektnega koda
operatorski in operandni skladi
izhodne subrutine
diagnostika napak

--- 15 % ---

6. ___ ___ ___

podatkovni tipi
prehodne funkcije
izrazi in stavki mešanih oblik

--- 5 % ---

7. ___ ___ ___

prevajanje subrutin in funkcij
pozivi parametrov z naslovom, imenom in vrednostjo
subrutine s stranskimi učinki
omejitve pri enoprehodnem izvajanju
objektni kod za prenos parametrov
objektni kod za subrutinsko telo

--- 15 % ---

8. ___ ___ ___

navezovalne metode
razprava o metaprevajalniku v njegovem lastnem jeziku

--- 5 % ---

9. ___ ___ ___

optimizacijske metode
frekvenčna analiza uporabe programskih struktur za ugotavljanje najpomembnejših lastnosti optimizacije

--- 5 % ---

10. ___ ___ ___

lokalna optimizacija s prednostjo posebnih ukazov
nalaganje registrov s konstantami
shranjevanje ničel
spreminjanje predznaka
seštevanje v pomnilnik
množenje in deljenje z 2
zamena deljenja z množenjem s konstanto
kvadriranje
potenciranje s celimi števili
primerjanje z ničlo
optimizacija indeksov

--- 10 % ---

11. ___ ___ ___

optimizacija izrazov
identitete, povezane z negativnim predznakom
izračun skupnih podizrazov in druge metode
minimizacija začasnega pomnilnika in števila aritmetičnih registrov v strojih z več registri

--- 5 % ---

12. ___ ___ ___

optimizacija zank
tipične zanke, kodirane na več načinov
optimizacija uporabe indeksnih registrov v notranjih zankah
klasifikacija zank za primerne optimizacije

--- 5 % ---

13. ___ ___ ___

problemi globalne optimizacije
opredelitev programskega grafa
analiza programskega grafa
preureditev programa tako, da se čim manj opravlja v notranjih zankah
faktoriranje invariantnih podizrazov
objektni kod za povezavo med posameznimi bloki (grafi)

--- 5 % ---

MODUL 6.3

Izračunljivost in kompleksnost (3-0-2)

Cilji

Preučujejo se najnovejši rezultati teoretičnih računalniških znanosti kot dopolnilo k modulu 5.7.

Poudarek modula

Poudarek naj bo na strogem prikazu matematičnih definicij in dokazov.

Vsebina modula

1. A l g o r i t m i

algoritmi
efektivne procedure
efektivni izračuni
algoritmični jeziki
zgodovina
definicije in primeri

--- 5 % ---

2. P r o g r a m i r n i j e z i k i

natančna definicija programirnih jezikov z enostavnim (programirnim) algoritmičnim jezikom (visok jezik)
primer: zančni jezik (PL) (strojni jezik ni priporočljiv)
programi za enostavne funkcije in lastnosti jezikovnega zaprtja
splošnost jezika
primer: izdelava jezikovnih konstruktov (WHILE, IF, REPEAT) in zamisli (rekurzija)
algoritmična ekvivalenca različnih lastnosti programirnih jezikov
Churcheva teza in diagonalna metoda, uporabljeni za prikaz obstoja neodločitvenih problemov
neodločitvenost ustavitvenega problema

--- 15 % ---

3. R e k u r z i v n e f u n k c i j e

uporaba jezika delno rekurzivnih funkcij za opredeljevanje funkcij
odvisnosti med operacijami p.r. funkcij (kompozicija, ponovitev, minimizacija, eksponiranje, primitivna rekurzija, kombinacija) in konstrukti programirnih jezikov
različne opredelitve rekurzije
deginicija rekurzivne funkcije, ki ni primitivno rekurzivna
metode kodiranja za predstavitev nizov kot števil ali odvisnih nizov nad različnimi abecedami
primeri: Goedelova števila, uporaba dvočrkovne

abecede

--- 15 % ---

4. Strojni jeziki

Turingovi stroji
neomejeni registrski stroji
stroji z naključnim dostopom

--- 5 % ---

5. Jezik za manipulacijo nizov

primeri: SNOBOL, algoritmi A.A. Markova, označeni Markovljevi algoritmi (programirane gramatike)
univerzalni program in funkcija
indeksiranje p.r. funkcij in dokaz neodvisnosti ustavitvenega problema brez Churcheve teze
indeksni in rekurzijski izreki in njihova uporaba

--- 15 % ---

6. Rekurzivno preštevne (r.p.) množice

ekvivalenca r.p. množic in jezikov, ki jih generirajo polthuevi in Postovi produkcijski sistemi
neodločljivost Postovega korespondenčnega problema
gramatike (kontekstno občutljive, kontekstno svobodne) kot omejitve polthuevih sistemov
odločitvenost in neodločitvenost različnih gramatičnih lastnosti

--- 10 % --

3. Sklep k drugemu delu

Opisani moduli, ki so pretežno matematične in teoretične narave, kažejo visoko stopnjo raznovrstnosti snovi, ki bi se naj poučevala. Ob skorajda klasično oblikovanih matematičnih modulih (1.2. Linearna algebra, 1.3. Matematična analiza, 2.5. Verjetnost in statistika, 2.7. Numerične metode I, 3.8. Numerične metode II, 5.7. Teorija formalnih jezikov in avtomatov ter 6.3. Izračunljivost in kompleksnost) najdemo vsaj za naše razume tudi bolj "obrobno" problematiko (2.3. Teorija informacije in kodiranje, 3.7. Teorija razvrščanja ter 4.6. Matematična logika), ki sodijo v fundament računalniških znanosti. Pri tem velja opozoriti na določena razhajanja med snovjo v modulu in naslovom modula. Npr. matematična logika obsega v glavnem teorijo izračunljivosti in njeno uporabo, sami logiki pa je odmerjen le manjši del modula.

Posebej omenimo tudi nekatere "sistemske" module (3.1. Sistemski koncepti in implikacije, 3.6. Operacijske raziskave, 4.2. Simulacija in modeliranje ter 5.2. Oblikovanje in realizacija sistemov), ki se vsebinsko toliko razlikujejo od naših programov, da so prav zaradi tega zanimivi. Ta zadnja ugotovitev - razlikovanje v primerjavi z našimi programi -, bi lahko pripeljala do polemičnih in novih kvalitetnih premikov tudi v domačih učnih programih.

Literatura

(1) A.P. Železnikar, Univerzitetni pouk računalništva, Informatica 4 (1980), št. 2, 32-42.

(2) Tečaj 2. stopnje organizacijsko-računalniške usmeritve, VŠOD, Kranj, 1980.

(3) A Proposal of a Modular Curriculum in Computer Science for Developing Countries (W.F.Atchinson, W. Brauer, R.A. Buckingham, J. Hebenstreit, Y. Parker, UNESCO-IFIP (first draft), Paris, March 1980.

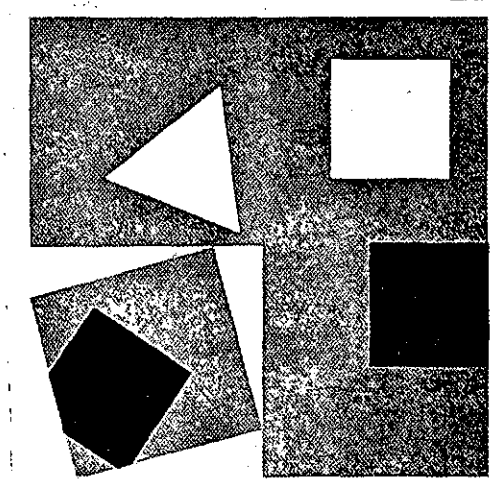
(4) Objave o študiji na FE (1980-81), Univerza E. Kardelja, Fakulteta za elektrotehniko, Ljubljana, avgust 1980.

(5) Predlog učnega načrta in opis predmetov VTO Elektrotehnika, Univerza v Mariboru, VTŠ, Maribor, april 1980.

J. Benkovič
A. Cokan
M. Martinec
R. Reinhardt
B. Roblek

RAČUNALNIŠTVO

Zbirka nalog 1



Zbirka nalog iz računalništva

Skupina avtorjev: Janez Benkovič, Aleksander Cokan, Mark Martinec, Robert Reinhardt in Branko Roblek je pripravila knjigo "Računalništvo: Zbirka vaj 1" (Državna založba Slovenije, 1980, 134 strani, cena 196.- din). Zbirka nalog dopolnjuje učbenik za predmet Računalništvo v srednjem usmerjenem izobraževanju. Razdeljena je na štiri poglavja: (1) Uvodni zgledi, (2) Problemi s pogoji in ponavljanji, (3) Podprogrami in (4) Tabele.

Posamezno poglavje sestavljajo skrbno izbrane rešene naloge - zgledi, ki večinoma rastejo drug iz drugega. Poglavje zaključujejo vaje - nerešene naloge, ki so namenjene utrjevanju snovi ob samostojnem delu učencev na računalniku. Zaporedje poglavij in zgledov sistematično vodi bralca od preprostejših k zahtevnejšim algoritmičnim in jezikovnim gradnikom.

Posebej velja omeniti strukturo posameznega zgleda. Tekstu naloge sledi namen zgleda, ki vsebuje tiste elemente jezika in programske prijeme, ki se v zgledu prvič pojavijo. Zatem je podan načrt rešitve, ki pri večini zgledov obsega pripravo in opis rešitve ter zapis algoritma s seznamom spremenljivk. Zgledi so do podrobnosti izdelani, zaključuje pa jih program v pascalu in fortranu ter rezultati. Upoštevane so realne možnosti uporabe računalniške opreme za izvajanje pouka računalništva. Vaje učencev potekajo na različnih računalnikih v pascalu oz. fortranu in to skoraj brez izjeme s paketnim načinom obdelave. To je verjetno tudi razlog, da v zgledih in vajah ne zasledimo interaktivnega dela z računalnikom ali uporabe grafičnega terminala ipd.

Problem vzporedne uporabe dveh programskih jezikov je težak. Z razdelitvijo in obliko zgledov je ta problem dobro rešen, tako da učenci, ki se praviloma ukvarjajo le z enim jezikom, ne bi smeli imeti težav.

Zbirki bi lahko očitali preveliko število nalog matematične narave in premalo nenumeričnih nalog, nalog iz poslovne informatike in drugih področij uporabe računalnika. Res pa je, da je težko pokriti razna področja z zgledi na tem nivoju, ki bi bili vsaj do neke mere realni. Več lahko v tem pogledu pričakujemo od napovedanega drugega dela zbirke, ki bo moral prisluhniti različnim usmeritvam in smerem usmerjenega izobraževanja. Odprto ostaja tudi vprašanje tistih nalog iz računalništva, ki niso vezane na programiranje. Morebiti kaže ob naslednji izdaji zbirke razmisliti o dodatnem poglavju s takimi nalogami.

V splošnem pa lahko zaključimo, da nam bo pričujoča zbirka v veliko pomoč pri poučevanju in učenju. Priporočamo jo vsem tistim, ki se želijo ob praktičnem delu na računalniku naučiti osnov dobrega programiranja v pascalu oz. fortranu in ne le učencem na srednji stopnji izobraževanja.

Vladislav Rajkovič

SLOVENSKO DRUŠTVO INFORMATIKA
LJUBLJANA, Parmova 41

Ljubljana, 16. 12. 1980

ZAPISNIK SEJE IO SDI, Z DNE 12.12.1980

Prisotni: S. Divjak, B. Džonova, A. Gorup, J. Grad, A. Jerman-Blažič, M. Krisper, M. Mekinda, R. Murn, L. Rinkwitz, K. Seršen, F. Žerdin, A.P. Železnikar

Dnevni red:

1. Potrditev in pregled zapisnika zadnje seje (9.5.80)
2. Obravnava predlogov za organizacijo simpozijev Informatica 81 in Informatica 82
3. Poročilo o stanju časopisa Informatica
4. Imenovanje Uredniškega odbora za izdajo monografij
5. Predlog nabave nove opreme
6. Predlog organizacijskih in kadrovskih sprememb
7. Razno.

Sejo je otvoril in vodil predsednik društva prof. dr. A. Železnikar.

Ad 1) Sklepi iz zapisnika zadnje seje z dne 9.5.1980 so bili izvršeni.

Ad 2) V zvezi z organizacijo simpozija Informatica 81 in 82 je imel prof. dr. Železnikar že več razgovorov s predstavniki Gospodarskega razstavišča in Elektrotehniške zveze Slovenije.

Možnosti, ki se ob tem pojavljata, sta tile:

- simpozij z razstavo v Ljubljani
- alternativa Ljubljana - Bled

Tov. Mekinda je pripomnil, da je potrebno poskrbeti za vsebino simpozija, da ta ostane na mednarodnem nivoju. Prednost simpozija v Ljubljani je v tem, da se dopolnjujeta sejem in simpozij.

Tov. A. Jerman-Blažič pa predlaga, naj bi simpozij Informatica 81 še bil na Bledu, kajti v Ljubljani je velik problem preskrbeti prenočišča s polnim penzionom za približno 400 udeležencev. Poleg tega je tudi cena bivanja in ostalih stroškov v Ljubljani precej večja kot na Bledu.

Tov. A. Jerman-Blažič je seznanil IO s predlogom pogodbe IJS in dal svoje naslednje predloge:

1. da se ugotovi interes IJS
2. da se zadrži mednarodni status (približno tretjino tujih referatov)
3. za l. 1981 predlaga, da sprejmemo pogodbo IJS in naj bo simpozij še na Bledu (IJS prevzame izvajanje organizacije in zato tudi odgovarja - vse pa naj bo naravnano v korist SDI)
4. za l. 1982 pa SDI sklene pogodbo z Gospodarskim razstaviščem za organiziranje simpozija v Ljubljani.

Predsednik društva naj je opozoril, da neglede na to, kje bo simpozij Informatica 81, po potekala organizacija Informatica 81 in Informatica 82 vzporedno.

Tov. Krisper je dodal, da naj bo z IJS sodelovanje na vsebinskem področju.

Sklep 1: Informatica 81 bo v pristojnosti SDI - vsi navzoči se strinjajo.

Sklep 2: Informatica 81 bo v Ljubljani.

Sklep 3: Simpozija bosta imela kolektivno vodstvo z naslednjimi odbori:

- programski odbor (D. Novak, M. Exel, M. Mekinda)
- organizacijski odbor (V. Rajković, M. Krisper, F. Žerdin)
- finančni odbor (V. Herbst, D. Šalehar, A. Gorup)
- recenzentski odbor (A.P. Železnikar, B. Džonova, Bavec)
- odbor za lokacije in prenočišča (K. Seršen, (L. Rinkwitz, ...)

Sklep 4: Programski odbor da vsebinska izhodišča za naslednji sestanek IO SDI.

Ad 3) Ena od naslednjih številk Informatice bo namenjena robotiki. Članke pričakujemo iz Gorenja in IJS. Številka 3 Informatice 80 gre v kratkem v tisk.

Ad 4) Uredniški odbor za izdajo monografij ima naslednje naloge:
- izdela poslovnik, ki ureja pogodbe z avtorji in zagotavlja rentabilnost poslovanja na področju izdajanja monografij. Odbor sestavljajo: Leskovar, Mekinda, D. Novak, A. Jerman-Blažič, Železnikar.

Ad 5) Predlog nabave nove opreme.

Predlaga se nakup manjšega sistema za vodenje in podporo administracije SDI. Zahtevali smo ponudbe od Altos Computer Systems in Cromemco oz. njegovega zastopnika Agromarketing, Zagreb. Na tem sistemu bi potekala glavna obdelava poslov za oba simpozija. Potreben je mobilen sistem, ki pa ga ni moč dobiti pri domačih proizvajalcih.

Tov. Krisper je pripomnil, da ima nabava tega sistema veliko funkcijo v okviru modernizacije administracije.

Sklep: soglasno sprejeto, da gre SDI v to nabavo.

Ad 6) Dosedanja tajnica, tov. Kuželicki Staša, je dala pismeno izjavo, da ne utegne več opravljati dosedanjih zadolžitev iz subjektivnih vzrokov. Na njeno mesto je bila izvoljena tov. Katarina Seršen kot v.d. tajnika SDI.

Tov. A. Jerman-Blažič je predlagal, da se zaposli tehnično tajnico s polnim delovnih časom.

Predlog članov je, da se opravi prenos sedeža društva SDI iz IJS ter podpiše samoupravni sporazum z DO Delta za dobo dveh let glede uporabe prostorov in drugih uslug.

Do morca 1981 bo računovodstvo SDI še po starem (IJS), potem pa naj se angažira računovodja, ki bo bližji novemu sedežu SDI.

SDI naj bi v bodoče stremelo za pridobitev lastnih prostorov, morda v okviru univerze ali republiških upravnih organov.

Ad 7) Republiško tekmovanje srednješolcev iz raču-

nalništva se lahko razširi v zvezno tekmovanje, če bo za to akcijo pripravljena ekipa (Dorn. Reinhardt, Hafner, Batagelj, Pisanski, Martinec itn.)

Naslednja seja IO SDI bo 26.12.1980 ob 12,00 uri na Parmovi 41.

Zapisnikar: K. Seršen, l.r.

Predsednik SDI: A.P. Železnikar

SLOVENSKO DRUŠTVO INFORMATIKA

LJUBLJANA, Jamova 39

Ljubljana, 6. 1. 1981

ZAPISNIK SEJE IO SDI, Z DNE 26.12.1980

Prisotni: Bratko, Čadež, Divjak, Hodžar, A. Jerman-Blažič, Krisper, Kanonenko, Mekinda, Rinkwitz, Seršen, Slatinek, Žerdin, Železnikar

Dnevni red:

1. Pregled zapisnika zadnje seje (12.12.1980)
2. Simpozija Informatica 81 in 82
3. Razno

Sejo je otvoril in vodil predsednik SDI tov. Železnikar.

Ad 1)

1.1 V zapisniku z dne 12.12.1980 smo pozabili vnesti sklep o povišanju honorarja tov. Murnu od 4.000,- din na 5.000,-- din na številko, kot tehničnemu uredniku revije Informatica.

1.2 Tov. A. Jerman-Blažič je pripomnil, da diskusija prejšnje seje ni bila natančno povzeta, izrazil je precejšen dvom, da bi tako hitro prenesli simpozij Informatica iz Bleda v Ljubljano.
Tov. A. Jerman-Blažič je ponovno poudaril ugodnosti oz. prednosti, da se simpozij Informatica 81 odvija na Bledu.
V zvezi s pogodbo IJS pa je tov. A. Jerman-Blažič povedal, da nas ni on seznanil z omenjeno pogodbo, temveč jo je le kritično pojasnil.

1.3 Pripomba tov. Divjaka - sklep št. 3 zadnje seje (12.12.80), naj bi bil samo v informacijo in še ni sprejet, ker to ni predlog IO.
Od GR zahtevamo pismeno ponudbo.

1.4 Dr. Hodžar je predlagal, da sprejmemo zapisnik s popravki na predlagani dnevni red in dodamo točko: formiranje dnevnega reda. Če je sklep po statutu nezakonit je treba to ugotoviti.

Razprava o prvi točki je bila zaključena s pripombami.

Ad 2) O programu simpozija za leto 81 in 82 mora razpravljati skupščina, gradivo pa pripravi IO.

Tov. Divjak je poudaril, da je potrebno sodelovati z vsemi gospodarskimi in družbenopolitičnimi organizacijami v Sloveniji, da bi s tem dobil simpozij nevtralni značaj.

Tov. Divjak predlaga za simpozij Informatica 81, da se odvija znanstveni oz. teoretični del na Bledu na GR pa paneli in razstava.

Tov. Mekinda je opozoril, da prinaša predlog tov. Divjaka novo kvaliteto.

Predlog tov. Žerdina je, da se zadolži nekaj članov, ki bodo konkretno razdelali predloge.

Pri tem so ponudili sodelovanje: Žerdin, Divjak, Krisper, Mekinda in Železnikar.

Naloga zgoraj omenjene komisije je:

1. Zbere vse pismene ponudbe
2. Dilema Bled - Ljubljana
3. Alternative in predlogi morajo biti dokumentirani: nosilec akcije je SDI
4. Simpozij mora imeti jugoslovanski in mednarodni nivo
5. Dilema simpozija Informatica 81 in 82 naj vključuje programski aspekt

Tov. A. Jerman-Blažič bo preskrbel vse podatke o jugoslovanskih simpozijih s področja informatike (terminsko in vsebinsko).

Ad 3) Razno.

Priprave na skupščino SDI:

3.1.1 Evidentiranje članstva (Železnikar, Seršen, Krisper)
(akcija, ki jo je treba takoj izpeljat).
Natisnemo 300 formularjev prijavnic.

3.1.2 Predlog članarine - pravilnik o članarini in olajšavah (pripravi tov. Divjak)
Skupščina naj odloči, kolikšna bo članarina za člane in študente.

3.1.3 Disciplinski pravilnik (pripravi DO - Žerdin, Grad, Mandelc)

3.1.4 Pravilnik o finančno materialnem poslovanju (pripravi NO - A. Jerman-Blažič)

3.1.5 Pravilnik o poslovanju uredniških odborov - pripravi tov. Murn

3.1.6 Pravilnik o organizacijskih in programskih odborih (Divjak)

3.1.7 Komisija za program dela (Banovec, Železnikar Mekinda, A. Jerman-Blažič)

3.1.8 Komisija za pripravo kandidatne liste - kandidacijska komisija (odprta kandidacijska lista)

3.1.9 Nadzorni odbor mora napravit poročilo za občni zbor

3.1.10 Poročilo disciplinskega odbora

3.1.11 Poročilo o zaključnem računu

3.1.12 Spremembe statuta - ugotovi se, če je statut SDI usklajen z ustavnimi in drugimi spremembami (to nalogo prevzame tov. A. Jerman-Blažič)

Sklep: na naslednji seji bomo obravnavali kandidate za kandidacijsko listo.

O internih spremembah se bo razpravljalo prav tako na naslednji seji, ki bo 16.1.1981.

NOVICE IN ZANIMIVOSTI

NOVI KRMILNIKI ZA DINAMIČNE POMNILNIKE

Tvrdka Texas Instruments bo v začetku leta 1981 dala na tržišče novo družino krmilnikov za dinamične pomnilnike. Gre za integrirana vezja popularne serije 74LS in sicer 74LS600, 74LS601, 74LS602 in 74LS603. Osnovna razlika med vezji je v tem, da so namenjena za različne konfiguracije pomnilnikov in za različne načine osveževanja.

Tip	Konfiguracija	Osveževanje
74LS600	4K/16K	prikrito in/ali masovno osveževanje
74LS601	64K	"
74LS602	4K/16K	kraja cikla in/ali masovno osveževanje
74LS603	64K	"

Vsa vezja bodo v standardnem 20 pinskem ohišju.

PD

2K X 8

Ena od vodilnih tvrdk na področju spominskih elementov MOSTEK, je za sredino leta 1981 najavila nov 16K statični pomnilnik z časom dostopa 90 ns. Vezje je kompatibilno s popularnim eprom-om 2716. Zaradi svoje izrazito uporabniško orientirane konfiguracije, bo vezje zagotovo naletelo na izredno ugoden odmev na tržišču.

PD

REGULATOR ZA TOKOVE 10A IN VEČ

National Semiconductor je razvil nov regulator, imenovan LM 196, katerega osnovna odlika je velik izhodni tok in disipacija moči 70 W. Vezje daje po zaslugi novega tehnološkega procesa točno definirano izhodno napetost v mejah ± 0.8%

PD

TRIDIMENZIONALNA SLIKA NA DOMAČIH TELEVIZORJIH

Produkt razvojnih laboratorijev tvrdke TI je eno najbolj kompliciranih integriranih vezij - TMS 9918 A - videlo display processor, ki je zasnovano tako, da se prilega večini popularnih mikroprocesorjev. S pomočjo vezja dosežemo visoko ločljivost /256x192/. Vsebuje 15 osnovnih barv, ki jim lahko primešamo tudi zunanji video signal. Tridimenzionalna prevara tiči v dejstvu, da je slika na ekranu razdeljena na 32 ravnin, paralelnih z ekranom. Če se na ekranu srečata dva objekta, potem tisti iz ravnine z višjo prioriteto pokrije tistega iz ravnine z nižjo. Vezje je možno priključiti brez posebnih dodatkov na vodilo računalnika.

PD

NOV KRMILNIK ZA GIBKE DISKE

Tudi TI se je pridružil proizvajalcem krmilnikov za gibke diske. Izdelek ima to lepo lastnost, da po sprejetju ukaza računalnika preko vgrajenega mikrokontrolerja opravi nalogo in prenese iz/v spomin podatek s pomočjo internega DMA procesorja. Na krmilnik je mogoča neposredna priključitev štirih pogonov. In še oznaka elementa:TMS 9909.

PD

DIGITALNI KORELATOR

TDC 1023 J je ime prvega digitalnega korelatorja v obliki samostojnega integriranega vezja. Vezje je sposobno primerjati 64 bitno besedo s serijskim signalom frekvence 20 MHz. Vezje uporablja princip korelacije za ugotavljanje specifičnih vzorcev v prihajajočem signalu. Kot dodatek lahko vezje uporabimo tudi za merjenje zakasnitev skozi različne medije. Osnovni del vezja je 64 bitni pomikalni register, ki ga poganja takt frekvence 40 MHz. Vezje je TTL kompatibilno. Cena pa je 85 $.

PD

PROGRAMLJIVI ŠTEVCI

8650 in 8651 sta nova programljiva generatorja in to za frekvence od 0,0005 Hz do 60 kHZ in od 0,00083 Hz do 100kHz. Oba lahko generirata do 64 različnih frekvenc s pomočjo vgrajenega kristalnega oscilatorja. Proizvajalec je EPSON AMERICA INC. CA 90505.

PD

SREČANJA

1981 leto

3-5 februar, Emeryville, ZDA

Fifth Berkeley Workshop on Distributed Data Management and Computer Network

Organizator:L.Berkeley Lab. in U.S.Dept. of Energy
Informacije:R.R.Johnson,L.Berkeley Lab.University of California, Berkeley, Ca, 9472o

23-25 februar, Julich, ZR Nemčija

Technical Conference on Measurement, Modeling and Evaluation of Computer Systems

Organizator: Gesellshaft fur Informatik, Nachrichtentehnishe Gesellshaft
Informacije: B.Mertens, KFA - Julich -ZAM, Postfach 1913, D-5170 Julich 1,F.R.Germany

24-26 februar, St.Lois, ZDA

ACM Computer Science Conference

Organizator: ACM
Informacije: 1981 Computer Science Conference Computer Science Departement, University of Missouri-Rolla, Rolla, MO 64401, USA

27-28 februar, Lausanne, Švica

International Conference on Aspects of Document Preparation Systems

Organizator: ACM Swiss Chapter, IEEE, AFCET, INRIA, GESO
Informacije: J.D.Nicoud, Calculatrice Digital, E.P.F.L. Bellerive 16, CH-1007 Lausanne, Switzerland

27 februar - 1 marec, New Delhi, India

Informatics 81, International Symposium on Informatics for Development

Organizator: Computer Society of India, IFIP Committee on Informatics for Development
Informacije:Informatics 81, NCSDCT,TIFR,Colaba, Bombay 400 00 , India

8-12 marec, San Diego, ZDA

5th International Conference on Software Engineering

Organizator: ACM, IEEE-CS
Informacije: L.Stucki, Boeing Computer Services Company, PO Box 24 346, Seattle, Washington 98 124, USA

9-13 marec, Dublin, Irska

Informatics and Industrial Development, International Conference

Organizator: IBI, UNIDO, National Board for Science and Technology of Ireland
Informacije:Conference Office, National Board for Science and Technology, Shelbourne Road, Dublin 4, Ireland

12-13 marec, Pittsburgh, ZDA

Computer Science and Statistics: 13th Symposium on the Interface

Organizator:Carnegie-Mellon University
Informacije:W.F.Eddy, Dept. of Statistics, Carnegie-Mellon University, Pittsburgh, PA 15213

18-20 marec,Tampa,ZDA

14th Annual Simulation Symposium

Organizator: ACM SIGSIM, IEEE-CS,SCS
Informacije:Philip N. Adams, Armco Inc. 24 North Main St. Middle-Town, OH 45043

23-25 marec, Houston, ZDA

Office Automation Conference

Organizator:AFIPS
Informacije:Carol Sturgeon, AFIPS OAC,1815 N.Lynn St.Suite 800, Arlington, VA 22209,ZDA

23-27 marec, Jahorina, Jugoslavija

V Bosansko-Hercegovački Simposium iz Informatike

Organizator: Elektrotehnički fakultet Sarajevo ETAN, SIZ Nauke SR B i H
Informacije: Elektrotehnički fakultet Sarajevo, Odsjek za Informatiku, 71 000 Sarajevo, Toplička bb., Jugoslavija

25-27 marec,St.Louis, Missouri, ZDA

8th Annual ACM SIGUCC Computer Center Management Symposium

Organizator: ACM SIGUCC
Informacije:R.E.Lee, Director of Computer Center University of Missouri-Rolla, Rolla, MO 65401 USA

25-27 marec, Baltimore, ZDA

Conference on Information Sciences and Syste Systems

Organizator:Johns Hopkins University
Informacije:G.L.Meyer and W.J.Rugh, EE Dept., Johns Hopkins University, Baltimore, MD 21218, USA

26-27 marec, St.Louis, Missouri, ZDA

ACM SIGCSE Technical Symposium on Computer Science Education

Organizator: ACM SIGCSE
Informacije: K.Magel , Computer Science Dept., University of Missouri at Rolla, Rolla MO 65 401 USA

30 marec - 1 april, London, Velika Britanija

ICS 1981, International Computing Symposium on Sytem Architecture

Organizator: Microprocessors and Microsystems in cooperation with ACM European Region
Informacije: I.S.Torsun, Dept. of Computer Science, Brunel University, Uxbridge, Middlesex UB 8 PH England

1-3 april, Paris, Francija

2nd International Symposium on Distributed Computing Systems

Organizator: IFIP
Informacije:T. Bricheteau, IRIA, B.P. 105, 78 78150, Le Chesnay, France

6-10 aprilFirence, Italija

International Congress on Logic, Informatics and Law

Organizator:Instituto per la Documentazione giuridica of Consiglio Nazionale delle Re Richerche
Informacije: Instituto per la documentazione giuridica, Via Panciatichi, 56/16-50127 Florence, Italy

13-16 april, Mexico City, Mehiko

15th International Symposium on Mini and Microcomputers

Organizator: International Society for Mini and Microcomputers
Informacije: ing. Jorge Gil, Academic Secretary, MIMI, IIMASUNAM, Apardo Postal 20-726, Mexico 20 D.F.

19-25 april, Peniscola, Španija

Formalization of Programming Concepts

Oragnizator: European Association for Theoretical Computer Science
Informacije: ICFPC, Facultat dInformatica, Universitat Politecnica de Barcelona, Jordi Girona Salgado 31, Barcelona 34, Spain

20-25 april, Zagreb, Jugoslavija

JUREMA

Organizator: JUREMA
Informacije: Tajništvo JUREMA, ul. Djure Salaja 5 /IV pp 398 41000 Zagreb, Jugoslavija

26-29 april, Annapolis, Md., ZDA

9th Annual Telecomunications Policy Research Conference

Organizator: National Science Fondation, Federal Communications Commission, Markle Fondation
Informacije: TPRC Organizing Commettee, c/o William Taylor, Bell Laboratoires 2C-258, 600 Mountain Avenue, Murray Hill, NJ 07974

29 april- 1 maj, Ann Arbor, ZDA

'ACM International Conference on Management of Data

Organizator: ACM SIGMOD
Informacije: Toby J. Teorey, Data base Systems Research Group, Graduate School of Business Administration, The Universtity of Michigan, Ann Arbor, MI 48109, USA

30 april-1 maj, Pittsburg, Pa., ZDA

12th Annual Pittsburg Conference on Modeling and Simulation

Organizator: University of Pittsburg in cooperation with Pittsburg section of IEEE, Systems, Man and Cybernetics Society
Informacije: William Vogt or Marlin Mickle, Modeling and Simulation Conference, 348 Benedum Engineering Hall, University of Pittsburg, Pittsburg, PA 15261

4-6 maj, New York, ZDA

11th Conference on Computer Audit, Control and Security

Organizator: EDP Auditors Foundation, Automation Training Center
Informacije: Harold Weiss, Automation Training Center, 11250 Roger Bacon Drive Suite 17 Reston VA 22090, USA

11-13 maj, Milwaukee, Wis., ZDA

13th Annual ACM Symposium on Theory of Computing

Organizator:ACM, SIGACT, University of Wisconsin, Milwaukee
Informacije:George Davida, Dept. of EECS, University of Wisconsin, Milwaukee, WI 53201 USA

12-14 maj, Minneapolis,ZDA

8th International Symposium on Computer Architecture

Organizator: ACM, SIGART
Informacije:V.R.Franta, Computer Science Dept.,Minneapolis, University of Minesota, 143 Space Center, MN 55455, USA

17-20 maj, Ann Arbor, Mich., ZDA

5th International Conference on Computers and the Humanities

Organizator: Assoc. for Computers and the Humanities, Assoc.,for Literary and Linguistic Computing, University of Michigan
Informacije: Gregory A. A. Marks, Institute for Social Research, University of Michigan, Ann Arbor, MI 48 104, USA

25-28 maj, Dubrovnik, Jugoslavija

III International Symposium "Computer at the University"

Organizator:University Computing Center Zagreb
Informacije: SRCE, for Symposium, Engelsova bb 41000 Zagreb, Jugoslavija

27-30 maj, Rousse, Bugarija

Organization and Automation of the Experimental Research

Organizator: State Committee of Science and Technical Progress and Central V Council of Scientific and Technical Unions in Bulgaria
Informacije: The Organizing Commmettee of the third Conference "Organization and Automation of the Experimental Research", looo Sofia, Rakovski str. 108, Bulgaria

3-5 juni, Amsterdam, Nizozemska

Second Seminar on Distributed Data Sharing Systems

Organizator: Vrije Universiteit, Vakgroep Informatica, INRIA, IGDD
Informacije:Prof. Dr.R.P.van de Riet, Vrije Universiteit, Vakgroep Informatica,Wiskundig Seminarium, De Boelelaan 1081, HV Amsterdam, the Nitherlands

8-10 juni,Portland, Ore., ZDA

ACM SIGPLAN Symposium on Text Manipulation

Organizator: ACM SIGPLAN, SIGOA
Informacije:P.Abrahams, 214 River Road, Deerfield, MA 01342, USA

9-11 juni, Pingree, Colo., ZDA

ACM Software Engineering Symposium on Tool and Methodology Evaluation

Organizator: ACM SIGSOFT
Informacije:William E.Riddle, Cray Labs., 5311 Western Ave., Boulder, CO 80301, USA

11-15 maj, Budapest, Madžarska

International Symposium COMNET 81, Networks from the User's Points of View

Organizator: IFIP, UNESCO
Informacije: COMNET 81 Secretariat, John V. Meumann, Society for Computer Sciences P.O.B, 240, H-1368, Budapest Hungary

10-12 juni, Waterloo, Ontario, Kanada

7th Conference of the Canadian Man Computer Communication Society

Organizator: Canadian Man Computer Communication Society
Informacije: Marceli Wein, Computer Graphic Section, Division of EE, National Research Council, Ottawa, Ontario, Canada, K1A OR8

10-12 juni, Nürnberg, Z R Nemčija

CONPAR 81, Conference on Analyzing Problem Classes and Programming for Parallel Computing

Organizator: Gesellschaft für Informatik and Institut für Mathematische Maschinen und Datenverarbeitung
Informacije:W.Handler, IMMD, Universitat Erlangen-Nürnberg, Martenstrasse 3 D 8520 Erlangen, F.R.Germany

24-26 juni, Portland, ZDA

11th International Symposium on Fault Tolerant Computing

Organizator:IEEE-CS
Informacije: Chung-Jen Tan, IBM T.J. Watson Research Center, Box 218, Yorktown Heights, NY 10598, USA

21-25 juni, Los Angeles, ZDA

Computers in Education Program at Annual Conference of the American Society for Engineering Education

Organizator: American Society for Engineeering Education
Informacije: Dean K. Frederick, Control Theory and System Program, Corporate Research and Development Center, General Electric Company, River Road, Schenectady, NY 12345, USA

15-17 juni, Portorož, Jugoslavija

Second Yugoslav Symposium on Applied Robotics

Organizator: Yugoslav Commettee for ETAN, Institut Mihailo Pupin, Institut Jožef Stefan Institut za Automatiku in Računarske nauke Energoinvest
Informacije: Yugoslav Committee for ETAN Symposium on Applied Robotics, POB 356, 11001 Beograd, Jugoslavija

29 juni- 1 juli, Nashville, ZDA

18th Design Automation Conference

Organizator:ACM SIGDA, IEEE-CS
Informacije:R.J.Smith II, V-R Information Systems Inc., 5758 Balcones Drive, Suite 205, Austin, TX 78731, USA

13-17 juli, Haifa, Izrael

8th International Colloquium on Automata, Languages and Programming

Organizator: European Association for Theoretical Computer Science
Informacije: S.Even (ICALP 81), Computer Science Dept., The Technion, Haifa, Israel

27-31 juli,Lasanne, Švica

3rd World Conference on Computers in Education

Organizator:IFIP TC 3 and its Working Groups 3.1, 3.3,3.4.
Informacije:P.Immer, Ecole Polytechnique Federale de Lausanne, Switzerland

5-7 august, Snowbird, Utah, ZDA

ACM Symposium on Symbolic and Algebraic Computation

Organizator:ACM SIGSAM, European SEAS, SMC, Nordic Interest Group for Symbolic and Algebraic Manipulation
Informacije: B.F.Caviness, Dept. of Mathematical Sciences, Rensseler Polytechnic Institute, Troy NY 12181, USA

24-28 avgust, Kyoto, Japonska

8th IFAC World Conference

Organizator:IFAC
Informacije: IFAC Secretariat, A.2361 Laxemburg Schlossplatz 12, Austria

25-28 avgust, Bangkok, Tailand

International Conference on Computing for Development

Organizator: Asian Institute for Technology and Carl Duisberg Gesellschaft in cooperation with ACM
Informacije:M.Nawaz Sharif, Director, Regional Computer Center, Div. of Computer Applications Asian Institute of Technology, P.O.B. 2754, Bangkok, Thailand

31 avgust-4 september, Strbskè Pleso, ČSSR

10th International Symposium on Mathematical Foundations of Computer Science
Organizator: Computing Research Center, Bratislava
Informacije:Jozef Gruska, Computing Research Center, Dubravska 3, 885 31 Bratislava, Czechoslovakia

7-9 september, Kaiserslautern, ZR Nemčija

International Conference on Computer Hardware Description Languages and Their Applications

Organizator:IFIP Tech.Comm. TC 10 and its Working Group W.G 10.2 in cooperation with ACM SIGARCH and SIGDA, IEEE-CS?,GI,NTG
Informacije:Reiner Hartenstein, Universitat Kaiserslautern, Fachbereich Informatik Posfach 3049, D-6750 Kaisenslautern, F.R.Germany

9-11 september, Darmstadt, ZR Nemčija

Eurographics 81, Annual Conference of the Eurographics Society

Organizator:German Computer Society
Informacije:J.Encarnaco, Eurographics 81,
Technische Hochschule Darmstadt, FG Graphisch Interactive Systeme, Steubenplatz 12,
D-6100 Darmstadt, Federal Republic of Germany

14-16 september, Las Vegas, Nev., ZDA

ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems
Or
Oragnizator: ACM Special Interest Group on Measurement and Evaluation
Informacije: Herbert Schwetman, Dept of Computer Science, Purdue University, West Lafayette IN 47907, USA

28-30 september, Brno, ČSSR

International Conference on Fault - Tolerant Systems and Diagnostics

Organizator: Czechoslovak Scientific and Technical Society, Czech. Central Commettee for Electrotechnics, Central Professional Group for Diagnostics in Electronics, Hause of Technology ČSVTS in Češke Budejovice
Informacije: DUM Techniky ČSVTS, Trida 5 Kvetna 42, 37 021, Ceske Budejovice, ČSSR

14-15 oktober Orlando, ZDA

Workshop on Data Bases d for Small Systems

Organizator: ACM SIGMOD, SIGSMALL
Informacije:Rob Gerritsen, President International Data Base Systems, Inc., 2300 Walnut Street, Suite 701, Philadelphia, PA 19103, USA

21-23 oktober, San Francisco, ZDA

APL 81

Organizator:ACM
Informacije: Eugene Mannacio, 428 Alameda dela Loma, Novato, CA 94947, USA

9-11 november, Los Angeles, ZDA

ACM 81

Oragnizator: ACM
Informacije:A.C. Tony Shelter, Xerox Business Systems, A1-39, 201 South Aviation Blvd. El Segundo, CA 90245, USA

1982 leto

9-11 februar, Indianapolis, ZDA

ACM Annual Computer Science Conference

Organizator:ACM
Informacije:Marshall Yovits, Indiana University at Indianapolis, 1125 East 38th St. Indianapolis IN 46205 USA

15-17 februar, Orlando, ZDA

Annual Computer Science Conference

Organizator:ACM
Informacije:Terry Frederick, Dept. of Computer Science, University of Central Florida Orlando, FL 305 275, USA

STROKOVNE RAZSTAVE

V Palais des Congrè v Parisu bo v času od 18-22 maja 1981 l. mednarodna razstava "Bureautique - AFCET- SICOB"
Paralelno z razstavo bo potekal kongres "Bureautique 81"Vse informacije v zvezi z razstavo in kongresom dobite če pišete na naslov:
AFCET, 156 , Boulevard Pereire ,
75017 Paris France, ali

SICOB, 4-6, Place de Valois,
75001 Paris , France

SEMINARJI

Iz programa SZAMOKa (International Computer Education and Information Centre) za leto 1981 smo izbrali nekatere zanimljive seminarje s področja računalništva in informatike:

5-9 maj
Software Development for Microcomputers

12-16 maj
Advanced Microcomputer and Microprocessor Applications

11-15 maj
Structured Program Design by Warnier`s Method (workshop)

18-22 maj
Structured Program Design by Jackson`s Method (workshop)

26-30 maj
Applied Statistics

9-13 juni
Distributed systems

5-9 oktober
Structured Systems Design (workshop)

12-16 oktober
Management of Computer Information System Development Projects

2o-24 oktober
Structured Program Design

27-1 31 oktober
Management of Programming Teams

Vse informacije v zvezi s seminarji dobite na naslov:
SZAMOK, Budapest 112 POB 146, H-1502,Hungary

Vabilo k sodelovanju

Univerzitetni Računski center v Zagrebu je organizator Mednarodnega simpozija o računalniku na univerzi. Prispevke lahko pošljete najkasneje do 15-4-1981 l. na naslov: SRCE, Engelsova bb, Zagreb. Program simpozija zajema naslednja področja:

-vloga in perspektive razvoja univerzitetnih računskih centrov
-informacijski sistemi in računalniške mreže
-organizacija in na analiza podatkov.

RAZISKOVALNE NALOGE, PRIJAVLJENE NA RSS V LETU 1980

Naslov naloge: Regulacija vrtilne hitrosti enosmernih motorjev s spreminjanjem polja, 3. faza, leto 1980.

Projekt: Krmilni in regulacijski sistemi

Nosilec naloge: Rafael Cajhen, EF, Ljubljana

Program raziskave:

- analiza metod za kombinirano regulacijo;
- matematična opredelitev sistema;
- osvojitev koncepta regulacije;
- optimiranje regulatorjev;
- zgraditev fizikalnega modela;
- meritve in preizkusi na modelu;
- analiza rezultatov in definicija smernic za izboljšave;
- problematika nelinearnosti in ukrepi za izboljšanje dinamike regulacijskega sistema;
- zaključki.

Naslov naloge: Podsestavi avtomatskih industrijskih merilnih sistemov V.

Projekt: Avtomatski industrijski modularni merilni sistemi

Nosilec naloge: Peter Šuhel, EF, Ljubljana

Program raziskave:

Kompleksna študija industrijskih merilnih sistemov:

- študija za izdelavo avtomatskih testirnikov podsestavov v analogni tehniki;
- študija problematike podsestavov digitalne tehnike.

Modularnost in programiranje merilnega sistema:

- študij možnosti vpeljave IEEE 488 vmesnika v avtomatske merilne sisteme za potrebe industrije;
- proučitev CAMAC vmesnika za večje industrijske sisteme kontrole kvalitete.

Sistemi merilnih robotov:

- študija mikroračunalniškega krmiljenja aktuatorjev.

Naslov naloge: Zasnova in strukturiranje operacijskih sistemov.

Projekt: Računalniška tehnika in proizvodnja

Nosilec naloge: Matija Exel, UIS, Ljubljana

Program raziskave:

a) raziskave konceptov softwarskih sistemov:
   - načini strukturiranja;
   - modularizacija s procesnim konceptom oz. konceptom modula;

b) sistemski jeziki:
   - študij in razvoj jezika Modula;
   - konceptualne osnove novih jezikov (Modula - 2, Ada);
   - metode dokazovanja pravilnosti;

c) uporaba izsledkov raziskav na področju zasnove manjših specializiranih (uporabniško usmerjenih) operacijskih sistemov z uporabo jezika Modula:
   - poskus restrukturiranja in poenostavitve nekaterih delov sistema Unix.

Naslov naloge: Informacijski sistemi in baze podatkov

Projekt: Informacijski sistemi

Nosilec naloge: Peter Tancig, UIS, Ljubljana

Program raziskave:

- izdelava smernic za nadaljnje delo na področjih projekta "Informacijski sistemi", ki se z letom 1980 izteka;
- nadaljnje preizkušanje in ovrednotenje moderne švedske metodologije (ISAC) razvoja IS in po potrebi njena prilagoditev našim pogojem;
- nadaljevanje dela pri razvoju participativne metodologije analize in načrtovanja IS s posebnim poudarkom na sodelovanju vseh prizadetih v številnih odločitvenih procesih:
  - utemeljitev mesta in vloge ustreznega odločitvenega procesa v procesu analize in sinteze IS;
  - razvoj participativnega večparameterskega odločitvenega modela na osnovi mehkih množic in predstavitve kompleksnega znanja;
  - preizkus in vrednotenje modela ter podporne programske opreme na konkretnih primerih s posebnim poudarkom na operativnosti odločitvene tehnike;
- Začetek dela na področju računalniške podpore pri realizaciji:
  - pregled stanja področja
  - izdelava nekaterih eksperimentalnih programskih orodij;
- dokončanje izdelave formalnega modela analize in sinteze infološke strukture IS;
- izdelava koncepta enovite (organizacije) poslovne baze podatkov v okviru obstoječega paketa za vodenje baze podatkov TOTAL ter njegova realizacija;

- Instaliranje sistema INGRES za obravnavanje relacijskih baz podatkov na računalnik PDP-11 in njegov prenos na manjši bazi podatkov;
- Izdelava eksperimentalnega LISP-ovega sistema za obravnavanje baz podatkov, na katerega bo priključen vmesnik za komuniciranje v naravnem jeziku;
- Nadaljevanje dela na problematiki uporabe bolj naravnih jezikov za komuniciranje z bazami podatkov:
  - pregled stanja tega področja;
  - osnovna formalna modeliranja sintakse slovenskega jezika;
  - ATN analizator za sintaktično-semantično analizo naravnega jezika;
  - izdelava računalniško vodene bibliogragije tega področja;
- Nadaljevanje dela na področjih uporabe matematične logike pri delu z bazami podatkov (modeliranje sveta)
  - formalizem podatkovnih baz in spraševalnih jezikov;
  - logično modeliranje naravnega jezika;
  - logika kot programski jezik;
  - mehanično dokazovanje teoremov in umetna inteligenca.

Naslov naloge: Sodobni koncepti upravljanja in vodenje poslov.

Projekt: Računalniška avtomatizacija industrijskih procesov

Nosilec naloge: Anton Čižman, IJS, Ljubljana

Program raziskave:
- nekatere matematične metode in postopki za računalniško identifikacijo multivariabilnih dinamičnih sistemov in parametrov; primerjava ter uporabnost metod;
- razvoj rekurzivnih metod za identifikacijo zveznih multivariabilnih sistemov z diskretnimi metodami;
- verifikacija razvitih identifikacijskih in transformacijskih algoritmov s pomočjo povezave analognega in digitalnega računalnika;
- razvoj laboratorijske naprave - modela multivariabilnega sistema z regulacijo temperatur in nivojev - za analizo, sintezo in testiranje zaprtih zančnih postopkov računalniškega vodenja.

Naslov naloge: Diagnostika delovanja sistemov.

Projekt: Računalniška tehnika in proizvodnja

Nosilec naloge: Janez Korenini, IJS, Ljubljana

Program raziskave:
- raziskava zakonitosti v sistemu glede na firmwarsko diagnostiko;
- izdelava matematičnega modela sistema glede na firmwarsko diagnostiko;
- raziskave časovnih parametrov;
- strategija določitve diagnostičnih linij;
- sinteza diagnostičnih postopkov na firmwarskem nivoju;
- praktični primeri.

Naslov naloge: Simulacija in zanesljivost digitalnih sistemov.

Projekt: Računalniška tehnika in proizvodnja

Nosilec naloge: Rudi Murn, IJS, Ljubljana

Program raziskave:
- metodologija načrtovanja opisa digitalnih elementov najvišje integracije;
- študij vplivov konfliktnih situacij pri funkcionalni simulaciji časovno krmiljenih elementov;
- ocena uporabnosti že zgrajenega simulatorja za razvoj novega funkcionalnega simulatorja;
- razširitev Petri mrež na nivo opisnega in modelirnega jezika s posebnim poudarkom na zahteve kot so: paralelizmi, sinhronizacija in konfliktna situacija (zlasti v zvezi s poz. 1 in 2);
- raziskave mikroračunalniških sistemov neobčutljivih na napake (Fault-Tolleraut):
  - verifikacija in lokalizacija napak s pomočjo kodnih in samotestnih postopkov;
  - določevanje kriterijev zanesljivosti.

Naslov naloge: Načrtovanje organizacij digitalnih sistemov III. faza

Projekt: Računalniška tehnika in proizvodnja

Nosilec naloge: Peter Kolbezen, IJS, Ljubljana

Program raziskave:
Raziskave v zvezi z razvojem sistema za mikroračunalniške aplikativne sisteme.
- uvajanje prevajalnika POLMP v programsko prakso (prenos paketa POLMP iz računalnika Syber 72 na računalnik PDP-11, razširitev jezika POLMP na makro ukaze mikroračunalniškega zbirnega jezika);
- razvoj uporabniško usmerjenega jezika za opisovanje testnih naborov za dinamično testiranje modulov digitalnih vezij (razvoj koncepta tesnega postopka, specifikacija materialne in programske opreme, razvoj koncepta vhodnega jezika, prevajanje v notranjo računalniško interpretacijo, prevajanje notranje interpretacije v izhodni jezik (tesni nabori), prevajanje odzivnih signalov vezja v notranjo interpretacijo izhodnega jezika);
- specifikacija in razvoj specifične materialne opreme za dinamično testiranje modulov digitalnih vezij;
- raziskave v zvezi z razvojem harwarskega simulatorja (Razvoj na osnovi že izdelanega koncepta, specifikacije komponent, primeri realizacije);

Raziskave multimikroračunalniških sistemov.
- študija in po možnosti tudi razvoj nekaterih za nas zanimivejših komponent sodobnejših mikroračunalniških sistemov (programska in materialna oprema);
- raziskave koordinacije procesov z mikroračunalniki (formalizacija opisovanja sočasnih procesov, optimizacija, razvoj univerzalne koordinacijske programske opreme - monitorja).

NAVODILO ZA PRIPRAVO ČLANKA

Avtorje prosimo, da pošljejo uredništvu naslov in kratek povzetek članka ter navedejo približen obseg članka (število strani A 4 formata). Uredništvo bo nato poslalo avtorjem ustrezno število formularjev z navodilom.

Članek tipkajte na priložene dvokolonske formularje. Če potrebujete dodatne formularje, lahko uporabite bel papir istih dimenzij. Pri tem pa se morate držati predpisanega formata, vendar pa ga ne vrišite na papir.

Bodite natančni pri tipkanju in temeljiti pri korigiranju. Vaš članek bo s foto postopkom pomanjšan in pripravljen za tisk brez kakršnihkoli dodatnik korektur.

Uporabljajte kvaliteten pisalni stroj. Če le tekst dopušča uporabljajte enojni presledek. Črni trak je obvezen.

Članek tipkajte v prostor obrobljen z modrimi črtami. Tipkajte do črt - ne preko njih. Odstavek ločite z dvojnim presledkom in brez zamikanja prve vrstice novega odstavka.

Prva stran članka:
a) v sredino zgornjega okvira na prvi strani napišite naslov članka z velikimi črkami;
b) v sredino pod naslov članka napišite imena avtorjev, ime podjetja, mesto, državo;
c) na označenem mestu čez oba stolpca napišite povzetek članka v jeziku, v katerem je napisan članek. Povzetek naj ne bo daljši od 10 vrst.
d) če članek ni v angleščini, ampak v katerem od jugoslovanskih jezikov izpustite 2 cm in napišite povzetek tudi v angleščini. Pred povzetkom napišite angleški naslov članka z velikimi črkami. Povzetek naj ne bo daljši od 10 vrst. Če je članek v tujem jeziku napišite povzetek tudi v enem od jugoslovanskih jezikov;
e) izpustite 2 cm in pričnite v levo kolono pisati članek.

Druga in naslednje strani članka:
Kot je označeno na formularju začnite tipkati tekst druge in naslednjih strani v zgornjem levem kotu,

Naslovi poglavij:
naslove ločuje od ostalega teksta dvojni presledek.

Če nekaterih znakov ne morete vpisati s strojem jih čitljivo vpišite s črnim črnilom ali svinčnikom. Ne uporabljajte modrega črnila, ker se z njim napisani znaki ne bodo preslikali.

Ilustracije morajo biti ostre, jasne in črno bele. Če jih vključite v tekst, se morajo skladati s predpisanim formatom. Lahko pa jih vstavite tudi na konec članka, vendar morajo v tem primeru ostati v mejah skupnega dvokolonskega formata. Vse ilustracije morate (nalepiti) vstaviti sami na ustrezno mesto.

Napake pri tipkanju se lahko popravljajo s korekcijsko folijo ali belim tušem. Napačne besede, stavke ali odstavke pa lahko ponovno natipkate na neprozoren papir in ga pazljivo nalepite na mesto napake.

V zgornjem desnem kotu izven modro označenega roba oštevilčite strani članka s svinčnikom, tako da jih je mogoče zbrisati.

Časopis INFORMATICA
Uredništvo, Institut Jožef Stefan, Jamova 39, Ljubljana

Naročam se na časopis INFORMATICA. Predplačilo bom izvršil po prejemu vaše položnice.

Cenik: letna naročnina za delovne organizacije 350,00 din, za posameznika 120,00 din.

Časopis mi pošiljajte na naslov ☐ stanovanja ☐ delovne organizacije.

Priimek.....

Ime.....

Naslov stanovanja

Ulica.....

Poštna številka_____ Kraj.....

Naslov delovne organizacije

Delovna organizacija.....

.....

Ulica.....

Poštna številka_____ Kraj.....

Datum..... Podpis:

.....

INSTRUCTIONS FOR PREPARATION OF A MANUSCRIPT

Authors are invited to send in the address and short summary of their articles and indicate the opproximate size of their contributions (in terms of A 4 paper). Subsequently they will receive the outor's kits.

Type your manuscript on the enclosed two-column-format manuscript paper. If you require additional manuscript paper you can use similar-size white paper and keep the proposed format but in that case please do not draw the format limits on the paper.

Be accurate in your typing and through in your proof reading. This manuscript will be photographically reduced for reproduction without any proof reading or corrections before printing.

Use a good typewriter. If the text allows it, use single spacing. Use a black ribbon only.

Keep your copy within the blue margin lines on the paper, typing to the lines, but not beyond them. Double space between paragraphs.

First page manuscript:

a) Give title of the paper in the upper box on the first page. Use block letters.
b) Under the title give author's names, company name, city and state - all centered.
c) As it is marked, begin the abstract of the paper. Type over both the columns. The abstract should be written in the language of the paper and should not excesed 10 lines.
d) If the paper is not in English, drop 2 cm after having written the abstract in the language of the paper and write the abstract in English as well. In front of the abstract put the English title of the paper. Use block letters for the title. The lenght of the abstract should not be greater than 10 lines.
e) Drop 2 cm and begin the text of the paper in the left column.

Second and succeeding pages of the manuscript:
As it is marked on the paper, begin the text of the second and succeeding pages in the left upper corner.

Format of the subject headings:
Headings are separated from text by double spacing.

If some characters are not available on your typwriter write them legibly in black ink or with a pencil. Do not use blue ink, because it shows poorly.

Illustrations must be black and white, sharp and clear. If you incorporate your illustrations into the text keep the proposed format. Illustration can also be placed at the end of all text material provided, however, that they are kept within the margin lines of the full size two-column format. All illustrations must be placed into appropriate positons in the text by the author.

Typing errors may be corrected by using white correction paint or by retyping the word, sentence or paragraph on a piece of apaque, white paper and pasting it nearly over errors

Use pencil to number each page on the upper-right-hand corner of the manuscript, outside the blue margin lines so that the numbers may be erased.

Časopis INFORMATICA
Uredništvo, Institut Jožef Stefan, Jamova 39, Ljubljana

Please enter my subscription to INFORMATICA and send me the bill.

Annual subscription price: companies 350,00 din (for abroad US $ 22), individuals 120,00 din (for abroad US $ 7,5).

Send journal to my ☐ home address ☐ company's address.

Surname.....

Name.....

Home address

Street.....

Postal code _____ City.....

Company address

Company.....

.....

Street.....

Postal code _____ City.....

Date..... Signature

.....

delta računalniški sistemi

SOZD ELEKTROTEHNA, o.o., DO DELTA, proizvodnja računalniških sistemov in inženiring, p.o.
61000 Ljubljana, Parmova 41
Telefon: 061/310-393
Telex: 31 578 YU ELDEC

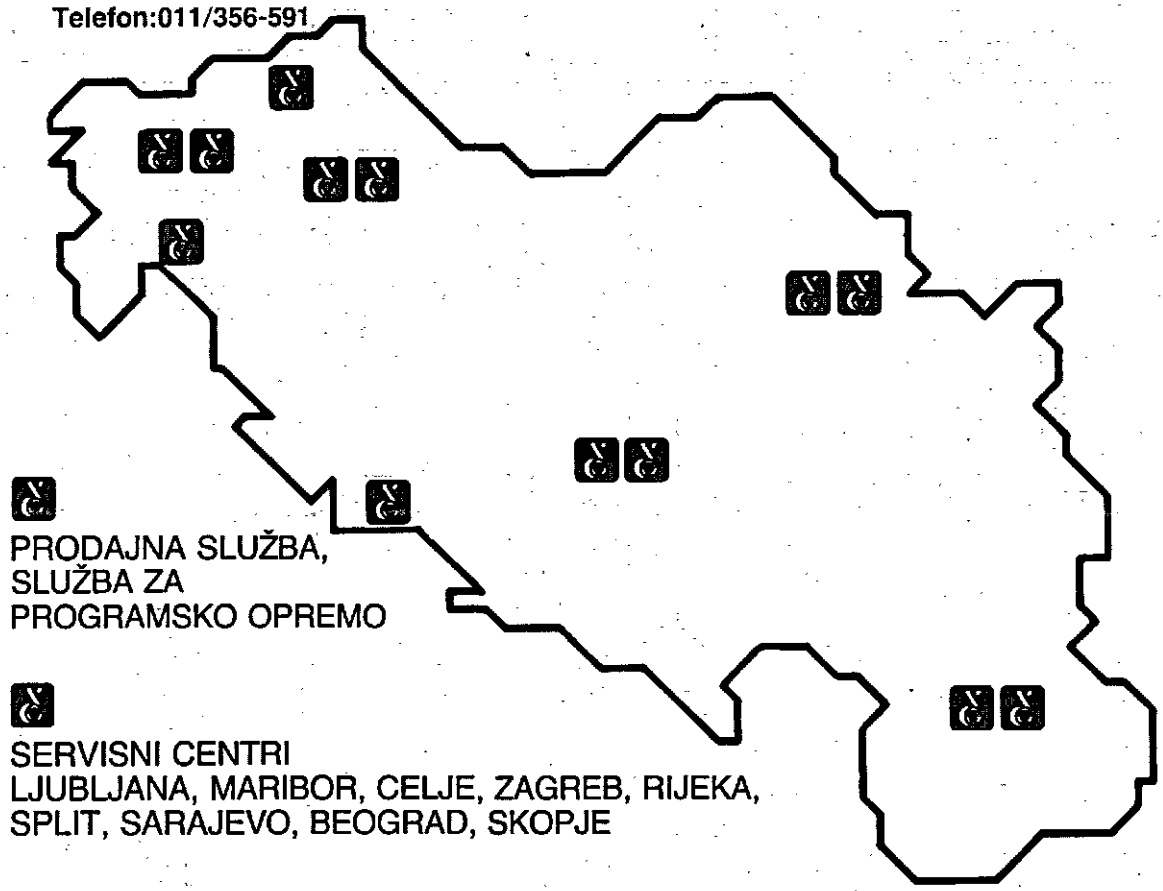
POSLOVNA ENOTA ZAGREB

ZAGREBAČKI VELESAJAM, II. UPRAVNA ZGRADA
ALEJA BORISA KIDRIČA 2, 41000 ZAGREB
Telefon: 041/520-003, 516-690

— PROIZVODNJA RAČUNALNIŠKE OPREME
61000 LJUBLJANA, LINHARTOVA 62a
Telefon: 061/323-585, 326-661
— VZDRŽEVALNA SLUŽBA
61000 LJUBLJANA, LINHARTOVA 62a
Telefon: 061/323-585, 326-661
— SLUŽBA ZA PROGRAMSKO OPREMO
TITOVA 51, 61000 LJUBLJANA, Telefon: 061/327-654
— DELTA IZOBRAŽEVALNI CENTER
Telefon: 061/345-673
— PRODAJNA SLUŽBA
TITOVA 51, 61000 LJUBLJANA
Telefon: 061/320-241, int. 397, 420
— DELTA INŽENIRING, PARMOVA 41, 61000 LJUBLJANA
Telefon: 061/314-394
— SLUŽBA ZA RAZVOJ STROJNE OPREME
Telefon: 061/23-251, 21-874
— SLUŽBA ZA RAZVOJ PROGRAMSKE OPREME
Telefon: 061/28-216

POSLOVNA ENOTA BEOGRAD

— VZDRŽEVALNA SLUŽBA — KARAĐORĐEV TRG 13, 11080 ZEMUN
Telefon: 011/694-537, 695-604
— PRODAJNA SLUŽBA, »SAVA CENTAR«
MILENTIJE POPOVIĆA 9, 11070 NOVI BEOGRAD
Telefon: 011/453-885
— SLUŽBA ZA PROGRAMSKO OPREMO — »SAVA CENTAR«
Telefon:011/356-591



PRODAJNA SLUŽBA,
SLUŽBA ZA
PROGRAMSKO OPREMO

SERVISNI CENTRI
LJUBLJANA, MARIBOR, CELJE, ZAGREB, RIJEKA,
SPLIT, SARAJEVO, BEOGRAD, SKOPJE