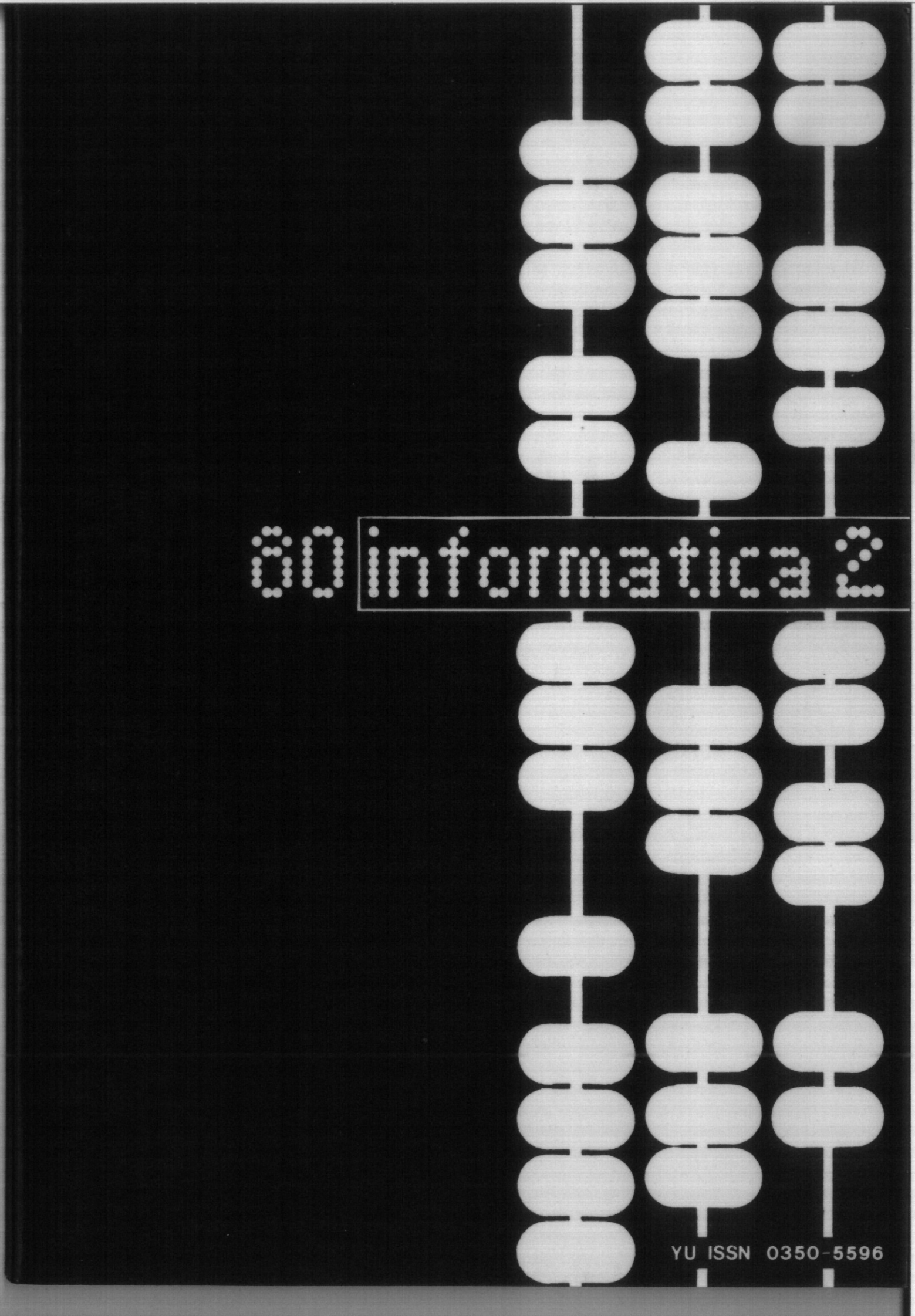


YU ISSN 0350-5596

Iskradata

ZA VEČJO PRODUKTIVNOST

RAČUNALNIK ISKRADATA C 18.

MIKRORAČUNALNIK ISKRADATA 1680.

ELEKTRONSKI PISALNIK ISKRADATA 80.

Področje uporabe:

avtomatska obdelava podatkov
spremljanje proizvodnje
vnos podatkov
prenos podatkov
krmiljenje procesov
grafične aplikacije
meritve
raziskave
izobraževanje
spremljanje rezultatov športnih tekmovanj
priprava teksta

ISKRA

informatica

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

Uredništvo in uprava: 61000 Ljubljana, Institut "Jožef Stefan", 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 RAČUNARSKU TEHNOLOGIJU I PROBLEME INFORMATIKE
SPISANIE ZA TEHNOLOGIJA NA SMETANJETO I PROBLEMI OD OBLASTA NA INFORMATIKATA

YU ISSN 0350 – 5596

LETNIK 4, 1980 – št. 2

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. 2

CONTENTS

JEDRO ZA PODPORO IMPLEMENTACIJE SPROTNIH SISTEMOV

D. NOVAK
M. EXEL
M. KOVAČEVIĆ
B. KASTELIC

UDK: 681.3.013/.014

IJS LJUBLJANA

Članek opisuje koncept jedra, ki omogoča učinkovito implementacijo sprotnega (real time) vodenja z mikroračunalnikom. Tak koncept omogoča dekomposicijo programskega paketa v logično zaključene dele - procese, ki med seboj komunicirajo.

V jedru so predvidene operacije, ki omogočajo preklapljanje procesov, sinhronizacijo in zaščito skupnih virov. Okolica sistema (predmet vodenja, informacijski viri - senzorji) vpliva na obnašanje sistema preko prekinitvenih vhodov. Prekinitve obravnavamo na enak način kot "navadne" signale v smislu Modula.

KERNEL FOR THE IMPLEMENTATION OF REAL TIME SYSTEMS. The paper describes a kernel for an efficient implementation of real time control systems using microcomputers. The kernel concept supports the decomposition of application software into logically well delimited modules - a set of cooperating processes.

Kernel provides operations for process switching and synchronisation and for implementation of mutual exclusion. The system behaviour is determined mainly by interrupts from the environment of the system: the kernel treats interrupts as "normal" signals in the sense of Modula.

1. UVOD

Sprotno vodenje procesov je ponavadi kompleksen problem. Kompleksnost izvira iz narave samih procesov, ki jih želimo voditi. Informacija o stanju procesa, ki jo generirajo razni senzorji je prisotna le kratek čas in se lahko pojavlja v neregularnih presledkih. Sistem, ki vodi proces, mora biti torej dovolj hiter, da lahko sprejme informacijo kadarkoli je na voljo. Ta informacija je lahko za sistem bolj ali manj pomembna. V prvem primeru je potrebno aktivnost sistema nasilno prekiniti, v drugem primeru pa sistem občasno pregleda svoje informacijske vire. Če med periodičnim pregledovanjem izgubi kakšno informacijo, to za sistem ni kritično.

Sistem za sprotno vodenje procesov mora torej zbirati informacije o stanju procesa in ga ustrezno voditi. Pri tem ne sme izgubiti relevantnih informacij o trenutnem stanju procesa. Seveda pa se tudi ne sme zamuditi s identificiranjem stanja procesa toliko časa, da bi proces "ušel". Sistem mora biti dovolj zmogljiv - hiter. To pa pomeni, da mora biti vedno zaposlen in da ne sme brezplodno čakati na kakšno sporočilo ali signal (busy wait).

Že iz teh splošnih ugotovitev je razvidno, da je sistem sestavljen iz treh delov - procesov: zbiranje podatkov in formiranje slike stanja vodenega procesa, odločanje - vodenje (izračunajo se nove vrednosti kontrolnih parametrov) in formiranje časovnih signalov. Omenjeni procesi se lahko odvijajo paralelno. Če imamo na razpolago samo en procesor pa se lahko izvajanje teh procesov prepleta.

Ugotovili smo torej, da je mogoče razbiti sprotni sistem na logične enote - procese, ki se lahko odvijajo paralelno. Torej je priporočljivo uporabiti metode multiprogramiranja, če imamo v sistemu en sam procesor. V ta namen moramo definirati pojem procesa, točke v katerih lahko pride do preklopa ter politiko dodeljevanja procesorja posameznim procesom. Elementarne operacije s katerimi implementiramo preklapljanje procesov so zbrane v jedru. Te operacije so nedeljive in jih implementiramo tako, da se izvajajo v privilegiranem režimu delovanja (prepovedane prekinitve) in jih ponavadi implementiramo kot sistemske funkcije.

V nadaljevanju se bomo najprej pobliže seznanili s pojmom procesa. Nato pa si bomo pogledali kako je s kreiranjem in brisanjem, s sinhronizacijo ter s preklapljanjem in razvrščanjem procesov v konkretnem jedru, ki je bilo implementirano na procesorju 8800. V zadnjih dveh poglavjih pa bo govora o startanju celega sistema procesov in o nekaterih implementacijskih podrobnostih.

2. PROCESI

Proces bomo imenovali vsak zaključen izvajajoči se program z lastnim skladom in deskriptorjem; s stališča sistema je proces enota za razvrščanje (scheduling). Če ne bo dvoumno, bomo v nadaljevanju imenovali program procesa kar proces. Deskriptor procesa je predstavnik procesa v jedru. Elementarne operacije jedra so definirane nad deskriptorji procesov.

Deskriptorji procesov, ki se ne izvajajo, sestavljajo čakalne vrste na procesor in na razne logične pogoje. Deskriptor sestavljajo torej kazalci, ki omogočajo formiranje vrst, indikator stanja procesa (ali je proces pripravljen na izvajanje ali čaka na izpolnitev kakšnega logičnega pogoja) in informacija potrebna za nadaljevanje procesa (vrednosti vseh registrov procesorja in naslov nadaljevanja ali vrednost kazalca na vrh sklada, če so vrednosti registrov spravljene v lastnem skladu).

Nov proces kreiramo tako, da naložimo ustrezni program v pomnilnik, formiramo deskriptor in poženemo proces. Če pa so programi za vse potencialne procese stalno prisotni v pomnilniku, se kreiranje zreducira na formiranje deskriptorja. Šele z deskriptorjem se lahko proces udeležuje tekme za procesor.

Brisanje procesa pomeni dejansko brisanje njegovega deskriptorja. Sam proces pa ostane nespremenjen v pomnilniku, dokler se prostor, ki ga zaseda, ne dodeli drugemu procesu.

Na monoprocesorskem sistemu je aktiven oziroma se izvaja samo en proces. Ostali so nanizani v čakalne vrste na izpolnitev logičnih pogojev, ki jih običajno signaliziramo s signali in v čakalno vrsto na procesor. Tekoči proces zapusti procesor prostovoljno, ko pride do točke, kjer določeni logični pogoji niso izpolnjeni (za izpolnitev bo poskrbel drug proces - sodelovanje procesov) ali neprostovoljno, ko ga prekine kakšen nujen zunanji dogodek.

3. PRIMER JEDRA ZA MULTIPROGRAMIRANJE

Jedro sestavljajo elementarne operacije za kreiranje in brisanje procesov ter za sinhronizacijo procesov. Poleg tega je v jedru program, ki dodeljuje procesor posameznim procesom, oziroma izbere izmed procesov, ki so pripravljeni na izvajanje enega in ga aktivira. Pri tem se lahko poslužuje poljubne politike izbiranja (glede na prioriteto, glede na dolžino čakanja v vrsti ipd). Poglejmo si pobliže nekaj takih elementarnih operacij jedra, ki je bilo implementirano na Motoroli 6800.

3.1. Sočasna izključitev procesov

V uvodu smo ugotovili, da pridemo do procesov z dekompozicijo problema spretnega vodenja. Zato je jasno, da so procesi med seboj odvisni - uporabljajo skupne vire (spremenljivke, V/I naprave, ipd). Prvi pogoj za pravilno delovanje sistema procesov je sočasna izključitev procesov pri uporabi skupnih virov. Vsak proces, ki želi uporabljati določen skupni vir, gre skozi naslednje faze: zasedba vira, uporaba vira in sprostitev vira. Uporabo skupnega vira imenujemo kritični del procesa. Samo en proces se sme nahajati v svojem kritičnem delu za določen vir. Zato potrebujemo na začetku in na koncu kritičnega dela določene sinhronizacijske operacije. V našem primeru sta to operaciji P(sem) in V(sem) nad binarnim semaforom sem. Ti operaciji sta implementirani s operacijama wait in send, ki bosta podrobneje opisani pri sinhronizaciji procesov. Operaciji P in V sta neprekinljivi.

```
P: if sem.value then wait(sem.signal)
   else sem.value:=true;
```

```
V: if queue on sem.signal=empty then
     sem.value:=false
   else send(sem.signal);
```

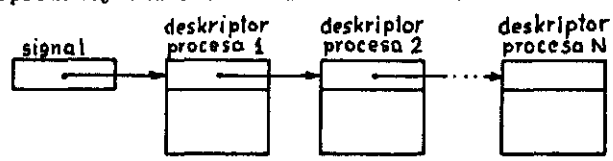
Sočasno izključenost procesov dosežemo torej na sledeč način:

```
P(busy):
uporaba vira: (*kritični del*)
V(busy):
```

Slika 1. Sekvenca, ki zagotavlja sočasno izključenost pri uporabi skupnih virov.

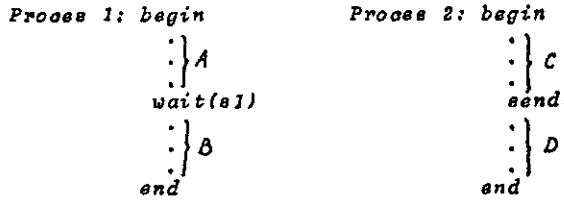
3.2. Sinhronizacija procesov

Za sinhronizacijo procesov uporabljamo signale (kot so definirani v Moduli). Signal je spremenljivka os. kazalec na vrsto.



Slika 2. Signal z vrsto procesov

V jedru sta definirani operaciji wait(signal) in send(signal). Prva postavi proces na konec vrste na signal, druga pa odstrani prvi proces iz vrste na signal in ga nanizа na konec vrste procesov, ki čakajo na procesor (vrsta procesov pripravljenih na izvajanje). Ti operaciji omogočata sinhronizacijo izvajanja procesov. Proces se z wait operacijo ustavi in je blokiran, dokler ga ne reši nek drug proces s send operacijo nad istim signalom. Tako smo na primer na sliki 3 s signalom s1 dosegli, da se lahko B v procesu 1 izvrši šele po izvršitvi C-ja v procesu 2.



Slika 3. Sinhronizacija procesov 1 in 2

Pri tem predpostavljamo, da bo proces 1 prej izvršil wait operacijo kot proces 2 send operacijo.

Opisani zgled na sliki 3 predstavlja enosmerno (unilateralno) sinhronizacijo. Signal "potuje" samo v eni smeri. Z dvosmerno (bilateralno) sinhronizacijo dosežemo enakopravnost procesov. V tem primeru imamo dva signala, ki "potujeta" v nasprotnih smereh. Tipičen primer dvosmerne sinhronizacije je skupen bufer z operacijama jemanja iz njega in vlaganja vanj, ki sta v različnih procesih (slika 4).

```
Vzemi_el: begin
            if št.el=0 then wait(ni_prazen);
            vzemi element iz buferja;
            dec(št.el);
            send(ni_poln);
          end.
```

```
Vstavi_el: begin
             if št.el=N then wait(ni_poln);
             vstavi element v bufer;
             ino(št.el);
             send(ni_prazen);
           end.
```

Slika 4. Dvosmerna sinhronizacija (N je dolžina buferja).

Opisana rešitev velja le za dva procesa. Če imamo več procesov, ki jemljejo elemente iz buferja, oziroma vstavljajo vanj nove elemente, moramo zagotoviti sočasno izključevanje izvajanja opisanih procedur.

Že na teh kratkih zgledih vidimo, da je osnovni pogoj za pravilno izvajanje sistema procesov sledeč: vsak signal mora biti pričakovan (proces, ki mu pošiljamo signal - send operacija, je že moral izvršiti wait operacijo nad istim signalom). Če ne zagotovimo izpolnitve tega pogoja, bodo procesi, ki so zamudili signal blokirani za vedno.

Tak način sinhronizacije pride torej v poštev samo v primeru, ko poznamo hitrosti posameznih procesov in je število procesov v sistemu dovolj majhno, da lahko predvidimo vse možne situacije. To pa je po Wirthu ravno razlika med multiprogramiranjem in sprotnim programiranjem (real time programming). Pri multiprogramiranju mora biti namreč zagotovljeno pravilno izvajanje sistema procesov ne glede na absolutne in relativne hitrosti procesov. Seveda pa morajo biti te hitrosti večje od nič.

Problem izgubljanja signalov in s tem povezanega možnega blokiranja sistema (dead lock) lahko olajšamo z dodatno operacijo awaited(signal) (Modula), ki pogleda ali je vrsta na signal prazna ali ne. Sedaj moramo pred vsakim pošiljanjem signala (operacija send) pogledati ali kdo pričakuje signal ali ne oz. ali je vrsta na signal prazna ali ne (operacija awaited) in v skladu s tem pošljemo signal ali ne. V našem sistemu operacija awaited(signal) ni implementirana. Zato se moramo zavedati opisanih nevarnosti.

Doslej smo se omejevali le na sinhronizacijo procesov med seboj. Pri sprotnih sistemih pa morajo biti ti sinhronizirani tudi z dogodki v okolici - eksternimi procesi. Zunanji dogodki se manifestirajo v obliki prekinitev. Ker smo želeli ohraniti enostavno strukturo sinhronizacije tudi za zunanje procese, smo tudi prekinitve klasificirali kot signale. To dosežemo tako, da damo v prekinitveno rutino za vsak prekinitveni vhod samo send operacijo nad priredjenim signalom (slika 5).

```
int_serv_rutine: begin
                   send(sig)
                 end
```

Slika 5. Prekinitvi smo priredili signal sig.

Z izenačitvijo "navadnih" signalov in prekinitev postane wait operacija bolj univerzalna, saj vključuje sedaj tudi čakanje na prekinitev, ki je v Moduli eksplicitno s doio operacijo. Tako ima npr. krmilnik naprave sledečo obliko:

```
key_dev_driver: begin

                  omogočitev prekinitve;
                  wait(prekinitev);
                  onemogočitev prekinitve;
                  čitanje znaka;
                end
```

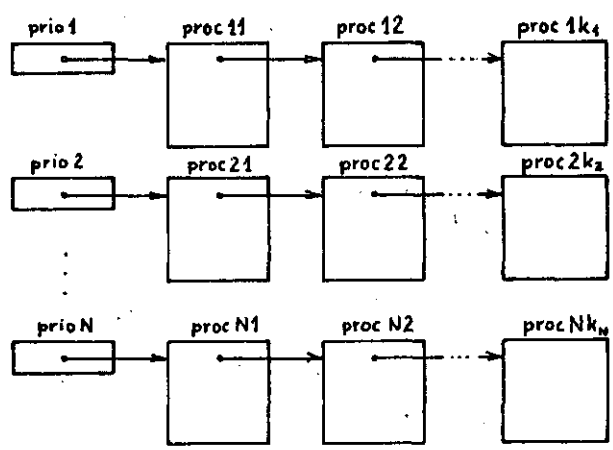
Slika 6. Krmilnik za sprejem znakov iz tastature

Pri inicializaciji sistema onemogočimo prekinitve na vseh napravah. Šele v krmilniku naprave se omogoči prekinitev - vzpostavi se prekinitvena pot od naprave do procesorja.

3.3. Kreiranje in brisanje procesov

Kreiranje procesov je omejeno na formiranje deskriptorjev procesov. Torej morajo biti programi procesov že naloženi v delovnem pomnilniku. Analogno pomeni brisanje samo brisanje deskriptorja. Ker lahko vsak proces briše samo svoj deskriptor, imenujemo operacijo brisanja samomor (suicide). Operaciji create in suicide sta prav tako implemetirani v jedru.

Deskriptorji vseh procesov sistema, razen tistega, ki se izvaja, so nanizani v vrste. Za vsako prioriteto obstoja svoja vrsta. Deskriptor procesa, ki se izvaja ni vključen v nobeno vrsto. Nanj kaže poseben kazalec.



Slika 7. Organizacija vrst

Znotraj vrst za posamezne prioritete se prepletajo še vrste na posamezne signale. Deskriptor procesa, ki čaka na signal, se nahaja fizično v dveh vrstah: v vrsti ustrezne prioritete in v vrsti na signal. Logično pa je seveda vedno samo v eni vrsti. Ko čaka na signal ima namreč status "wait on signal", ki pove izbiralnemu programu - "schedulerju", da naj ga pri izbiranju naslednjega procesa preskoči. Dokler ne prejme signala se ne udeležuje tekme za procesor.

Iz opisane strukture vrst je razvidno, da operacija create vsebuje: dodeljevanje prostora za deskriptor, formiranje deskriptorja in vključitev v ustrezno vrsto s stanjem "pripravljen na izvajanje". Operacija create v bistvu predstavlja operacijo startanja procesa, katerega program je že naložen.

Če proces ni ciklièen, izvrši na svojem logičnem koncu operacijo suicide. S tem se je izložil iz čakalne vrste in ne more več dobiti procesorja. Kot smo omenili pa ostane proces v delovnem pomnilniku in ga je mogoče ponovno startati z operacijo create.

3.4. Preklapljanje in razvrščanje

Preklop pomeni ustavitev tekočega procesa, izbiro in aktiviranje novega. Naštete operacije izvrši poseben program v jedru, ki ga imenujemo razvrščevalnik (scheduler).

Proces ustavimo tako, da shranimo stanje vseh registrov procesorja in nanizamo deskriptor tekočega procesa v eno od čakalnih vrst. Izbira se vrši po vnaprej določenem algoritmu. V našem primeru po vrsti (začenši z najvišjo prioriteto) pregledujemo čakalne vrste na posameznih prioritetnih nivojih in iščemo proces, ki je pripravljen na izvajanje. Ko naletimo na tak proces, restavriramo stanje registrov procesorja z vrednostmi kot so bile pri ustavitvi tega procesa in sprožimo izvajanje. Še prej pa izložimo deskriptor izbranega procesa iz vrste. Če izbiralni program ne najde nobenega procesa, ki bi bil pripravljen na izvajanje, starta slepi proces (dummy process). To je neskončna zanka. Slepi proces ne bo več odstopil procesorja prostovoljno. Za preklop je potrebna zunanja prekinitev. Če so vsi procesi blokirani, je jasno, da lahko le zunanji dogodek "reši" vsaj en proces in s tem sistem ponovno steče.

Preklop se izvrši v naslednjih primerih:

a) proces se je izvršil do mesta kjer mora čakati na izpolnitev določenih logičnih pogojev (wait(s) operacija),

b) proces je prekinjen. Prekinitev je ekvivalentna send(s) operaciji. Lahko se zgodi, da postane ob prekinitvi kakšen proces z visoko prioriteto pripravljen na izvajanje. S tem, da se izvrši preklop takoj po prekinitvi, se zmanjša odzivni čas, saj ima proces, ki čaka to prekinitev več upanja, da bo prišel na vrsto. Prekinitev se razlikuje od navadne send operacije le po tem, da ji vedno sledi preklop (na koncu prekinitvene rutine pride vedno do preklopa),

c) proces prostovoljno odstopi procesor. V jedru je implementirana posebna operacija sleep. Ta postavi proces, ki jo je klical, v čakalno vrsto, čeprav je še vedno pripravljen na izvajanje. Ta operacija pride v poštev, kadar določenemu procesu dinamično spreminjamo prioriteto. Ob vsakem znižanju prioritete je priporočljiv preklop. Če nima nobeden od procesov, ki so pripravljeni na izvajanje višje ali enake prioritete kot proces, ki je klical operacijo sleep, se isti proces nadaljuje - ni preklopa.

Opisana shema preklapljanja se nekoliko razlikuje od preklapljanja v Moduli. Tam pride do preklopa po vsaki send operaciji, razen v primeru, ko pošiljamo signal procesu nižje prioritete (send v procesu naprave - "device process"). V našem primeru pa po send operaciji ni preklopa. Razlika je tudi pri prekinitvah. V Moduli so te pričakovane z doio operacijo. Po vsaki prekinitvi pride do preklopa in aktivira se proces, ki je čakal na to prekinitev. Pri nas pa pričakujemo prekinitev z navadno wait operacijo. Po vsaki prekinitvi pride do preklopa, pri čemer ni rečeno, da bo stekel proces, ki je čakal na prekinitev. Aktiven postane proces z najvišjo prioriteto. Procesi, ki so vezani na okolico (prekinitve) morajo imeti visoke prioritete, tako da je zagotovljen zahtevani odzivni čas.

4. STARTANJE SISTEMA PROCESOV

Doslej smo govorili pretežno o sinhronizaciji procesov, kreiranju in brisanju ter o preklapljanju in razvrščanju. Niemo pa se vprašali kako bomo sistem procesov sploh startali. Pri opisu jedra smo omenili, da operaciji create in suicide samo formirata oz.brišeta deskriptor procesa. Sedaj pa si poglejmo kako je z nalaganjem.

Jedro sistema in vsi procesi so zapisani na gibkem disku (floppy disk). Poleg tega je na disku še poseben proces imenovan proces 0, katerega naloga bo zgraditev sistema in tabela, ki vsebuje vse začetne pogoje za vse procese v sistemu. Vsak proces ima svoj sklad v katerem so definirane začetne vrednosti registrov in začetni naslov izvajanja. V tabeli pa je za vsak proces vrednost kazalca na vrh sklada in prioriteta procesa. V jedru je vnaprej formiran samo en deskriptor in ta pripada procesu 0. Inicialni nalagalnik (spravljen v EPROMu) naloži cel paket iz diska v delovni pomnilnik in sproži proces 0. Proces 0 se odvija v neprekinljivem režimu. Proces 0 kreira v skladu s podatki v sistemski tabeli deskriptorje vseh procesov. Procesi sami so že naloženi v delovnem pomnilniku. Na koncu izvrši proces 0 samomor in s tem starta cel sistem. Operacija suicide se namreč nadaljuje s posivom razvrščevalnika, ki išče nov proces, ki je pripravljen na izvajanje in ga aktivira.

5. IMPLEMENTACIJA

Operacije jedra so implementirane v obliki sistemskih funkcij, ki se odvijajo v privilegiranem režimu (prepovedane prekinitve). Aktiviramo oz.kličemo jih s sistemskimi posivi, ki so implementirani s programsko prekinitvijo:

```
SWI
FCB   #x      x - številka sistemske
                  funkcije
```

Ob vsakem sistemskem posivu se shrani stanje registrov v sklad procesa, ki teče. S tem je opravljena že polovica dela, ki je potrebno ob preklopu. V deskriptor moramo shraniti samo še vrednost kazalca na vrh sklada. Deskriptor procesa obsega 8 zlogov (bytov): status in prioriteta, dva kazalca za uvrščanje v čakalne vrste, kazalec na vrh sklada in zlog za dodeljevanje prostora za deskriptor.

Poleg naštetih operacij je bila implementirana še operacija za spreminjanje prioritete procesa. Kompletno jedro obsega nekaj manj kot 500 zlogov pomnilnika. Za vrste in za sistemske semafore in signale pa je rezerviranih še 280 zlogov.

6. SKLEP

Opisano jedro je bilo implementirano v okviru izdelave specifičnega operacijskega sistema. Ta operacijski sistem bo tekel na enem od računalnikov v multimikroračunalniškem sistemu za vodenje telefonske centrale. Samo jedro pa bo prisotno v vseh mikroračunalnikih sistema. Prvi testi so dali dobre resultate. Podobno kot v Moduli so postali programski krmilniki naprav mnogo preglednejši. Jasno je, da moramo plačati določeno ceno za pridobljeno modularnost in s tem enostavnost aplikacijske programske opreme. Ta cena je v "overheadu" jedra, ki podaljšuje odzivne čase sistema. Zato je bil naš cilj, da zasnujemo čimbolj enostavno jedro, ki pa bo vseeno omogočilo učinkovito sinhronizacijo in "pošteno" politiko dodeljevanja procesorja. Končno oceno ali nam je to uspelo ali ne bo mogoče podati šele, ko bo zaživel cel sistem.

7. LITERATURA

1.N.Wirth: Modula (trije članki), Software - Practice and Experience, Vol.7(1977),3-84.

2.N.Wirth: Toward a Discipline of Real-Time Programming, Communications of the ACM, Vol.20(1977), No.8, 577-583.

3.T.M.Raleigh: Introduction to Scheduling and Switching under UNIX,Proceedings of the Digital Equipment Computer Society, Atlanta, Georgia - May 1976, 867-877.

4.D.Novak, M.Exel, M.Kovačević, B.Kastelic: Modula - programski jezik prihodnosti za mikroračunalniške aplikacije? Informatica, letnik 3(1979), št.2, 18-24.

5.M.Exel: Komunikacija med sekvenčnimi procesi - pregled, del I in del II, Informatica, letnik 1(1977), št.1 in 2.

PRAVLJICA O MULTIMIKROJIH

Prosti prevod iz revije Elektronische Rechenanlagen, 22.Jahrgang (1980), Heft 2 - H.N.Lipp: Das Maerchen von den Multimikros.

Nekoč je živel sistem, v katerem so se pojavile manjše upravne naloge. Ker so cene mikroprocesorjev ravno močno padle, je zadolžil za to dejavnost en 8 bitni mikroprocesor, čeprav bi moral sam za to dodatno dejavnost porabiti le delček svojih zmogljivosti. Upravni mikroprocesor je prevzel omenjene naloge. Kadar je moral čakati, ni ostal križem rok, temveč je začel z razvojem protokolov, po katerih naj bi mu sistem posredoval naloge. Ker ga je motila neposredna povezava s sistemom, si je uredil predsobo v obliki ojačevalnikov treh stanj (tri-state buffer). Poleg tega si je priskrbel še lokalni delovni prostor. Njegovo navdušenje nad nadaljnjim razvojem čistih upravnih aspektov je bolj in bolj rastlo. Končno je zahteval pomožni mikroprocesor, ki bi ga razbremenil pri samem delu. Tega je tudi dobil. Kot je bilo za predvidevati, je prišlo kmalu do konfliktov pri razdeljevanju nalog in pri uporabi skupnih pripomočkov. Nujna je postala vpeljava prioritet in poslovnika v obliki shem oddajanja. Odločilni kriterij je bila, kot običajno, služben a doba.

Zahtevam po nadaljnjih procesorjih se kmalu ni dalo več oporekati. Staremu mikroprocesorju je uspelo uveljaviti hierarhično upravno strukturo s samim na vrhu te strukture. Sedaj je samo še delegiral naloge, sprejemal rezultate in vzdrževal komunikacijo s sistemom. Tudi to mu je kmalu postalo nadležno, ker je s tem izgubljal dragoceni čas, ki bi ga lahko uporabil za teoretiziranje o upravljanju. Zato je vpeljal posebne vhodno-izhodne enote, kar mu je omogočilo, da je dvignil celo hierarhično piramido za en nivo. Končno je padel učinek na tako nizko vrednost, da je sistem podvzel ukrepe s katerimi naj bi ponovno dvignil učinkovitost svojega uslužnostnega podsistema. Zavoljo širšega pregleda je naložil zunanjemu 16 bitnemu mikroprocesorju nalogo, naj izdela strokovno mnenje. Preiskovalni mikroprocesor je izvedel izčrpne meritve in zbral vzorce posameznih aktivnosti ter jih celo ocenil. Bilanca njegovega truda je bila identifikacija nekaj mikroprocesorjev, ki so izvajali izključno NOPe. Veliko število dela potrebnih mikroprocesorjev pa mu je ušlo, ker jih zaradi zvito razporejenih DO zank in zaradi zaposlenosti pri čakanju na vodilo ni opazil.

Za pomoč pri konicah je našel naš sistem naslednjo rešitev: paralelno k obstoječi, bo zgradil še eno upravno strukturo, za kar bo potrebnih precej novih mikroprocesorjev. Stara hierarhija pa ni hotela brez nadaljnjega prepustiti neljubi konkurenci svojega področja in je reagirala z ustreznimi protiukrepi, kar je učinek še zmanjšalo. Vhodno-izhodne naprave so morale z vpeljavo časovnikov in s tem povezano vpeljavo uradnih ur, reducirati možnost zunanjega dostopa s strani sistema, ker so bile obremenjene z izdajo lastnih upravnih protokolov in z obrambo pred tujimi. Končno so popolnoma prenehale občevati z okolico.

Obe upravni strukturi sta bili sedaj zaposleni z generiranjem vzajemnih predpisov in z interpretacijo navodil. Ko so se nakopičili nerešljivi konflikti,sta se zedinili za uvedbo novega koordinacijskega mikroprocesorja, ki sta ga lahko obe strukturi klicali. In če sistem ni prekinil napajalnih linij, potem vsi mikroprocesorji veselo mikroprocesirajo še danes.

D.Novak

UVODJENJE PARALELIZMA U OBRADI PROGRAMA ZA MULTI–MIKROPROCESORSKE SISTEME

NIKOLA HADJINA

SVEUČILIŠNI RAČUNSKI CENTAR, ZAGREB

UDK: 681.3.012

U radu su dane mogućnosti uvodjenja paralelizma u obradi programa za multiprocesorske sisteme, kako sa stanovišta definiranja posebnih jezika, tako i sa stanovišta detekcije paralelnih procesa, dijelova programa, u već postojećim sekvencijalnim programima. Posebno su razradjeni uvjeti na osnovu kojih se odredjuje sekvencijalnost i paralelnost medju blokovima. Dan je postupak implementiranja na već postojećem kompilatoru programskog jezika.

INTRODUCING OF PARALLEL PROCESSING IN PROGRAM EXECUTION FOR MULTI-MICROPROCESSOR SISTEMS: Possibilities of parallel processing in program execution for multi-microprocessing systems, by definition of a new programming languages and by detection of parallel processable code in sequential programms are given. Conditions for sequentiality and parallelity between blocks of sequential program are specially analysed. Implementation of this method in already existing compilers is also given.

1. UVOD

Problem organiziranja i pisanja programa za višeprocesorske sisteme igra jednu od važnijih uloga u programskom inženjerstvu. Ovdje je potrebno načiniti razliku izmedju sistemske i aplikativne programske podrške. Sistemski programi (operacioni sistemi, kompilatori i dr.) su tradicionalno napisani od strane eksperata, sistem programera, dok je aplikacijska programska podrška napisana od strane programera sa općim znanjem o programiranju i primjeni računala.

Na nivou sistemskog programiranja postoje razradjeni postupci za upravljanje i kontrolu paralelizma u obradi na višeprocesorskim sistemima, za izbjegavanje konfliktnih situacija izmedju asinhronih procesa, za sinhronizaciju i kreiranje paralelnih procesa i dr.

Osnovni zadatak je dakle sistemskih programa da aplikativno programiranje učine što jednostavnijim i efikasnijim.

S tog aspekta najveći teret pada na programske jezike i njihove procesore (kompilatore).

Glavni alat za uvodjenje paralelizma, funkcijska dekompozicija programa, se dakle može provesti na dva nivoa:

1. Uvodjenjem viših programskih jezika koji omogućavaju programiranje paralelnih procesa;
2. Izmjenama i dopunama standardnih kompilatora viših programskih jezika kako bi se mogli detektirati paralelni procesi unutar sekvencijalnih programa.

Kod uvodjenja paralelizma prvom metodom potrebno je prvo, veliki zahvat na sistemskom nivou (definiranje sintakse i semantike jezika, te pisanje novog kompilatora) i dobro poznavanje od strane programera kako programskog jezika tako i samog problema koji se rješava.

Kod korištenja druge metode nije potreban zahvat na definiranju samog jezika već je potrebna samo dopuna već postojećeg kompilatora, što će biti pokazano u ovom radu.

2. PROGRAMSKI JEZICI ZA PROGRAMIRANJE PARALELNIH PROCESA

Ovdje će biti dano nekoliko ideja za uvodjenje viših programskih jezika za paralelno programiranje, te kao primjer biti će dana definicija postojećeg jezika za paralelno programiranje (CONCURRENT PASCAL).

DEFINIRANJE PROGRAMSKOG JEZIKA

Da bi jezik bio pogodan za paralelno programiranje potrebno je da sadrži neke naredbe kao npr:

NAREDBA	*ZNAČENJE*
1. DO ALL	Poredak izvodjenja slijedećih naredbi nije važan.
2. DO SEQUENTIALLY	Izvodjenje slijedećih naredbi se mora provesti u sekvenci. (Sekvencijalna obrada se podrazumjeva ako se ne specificira DO ALL).
3. DO BOTH	Izvodjenje slijedeće dvije naredbe se provodi bez obzira na poredak.

Posebno efikasni aplikacijski sistemi su oni u kojima su jezici tako definirani da omogućavaju operacije na posebno definiranim strukturama podataka. Npr. kompilator za jedan procesor polja (array processor) može proizvesti efikasniji kod ukoliko programski jezik dozvoljava operacije nad cijelim poljem (PL/1, APL i dr.). Raspoloživost ove informacije u vrijeme kompilacije, a ne u vrijeme izvodjenja (pozivi procedura) doprinosi generiranju efikasnijeg koda.

Posebna pažnja mora biti posvećena ulozi rekurzije u programiranju više mikroprocesorskih sistema. Po tradiciji se je na rekurziju gledalo kao na tehniku programiranja uglavnom sa teoretskog i pedagoškog aspekta, posebno zbog velikog utroška vremena izvodjenja, ali u sistemima sa više procesora više rekurzivnih poziva se može izvoditi paralelno.

CONCURRENT PASCAL (CO-PASCAL)

CO-PASCAL predstavlja proširenje sekvencijalnog *PASCAL-a* uvodjenjem novih sistemskih komponenti, procesa, monitora i klasa, te novih naredbi jezika.

PROCES predstavlja jednu aktivnost programa, a sastoji se od područja vlastitih podataka (ulazno/izlaznih) i sekvencijalnog programa, čije se naredbe izvode striktno jedna za drugom.

MONITOR omogućava programu da specificira kompilatoru kako će se koristiti neki djeljivi podatak od strane više procesa istovremeno. Monitor sinhronizira paralelne procese i prenosi podatke medju njima. On može kontrolirati raspored korištenja zajedničkih sistemskih sredstava (programskih i sklopovskih). Monitor se sastoji od djeljivih podataka, operacija za sinhronizaciju te od početne operacije.

KLASE definiraju strukturu podataka i moguće operacije nad njima.

Tako se program u *CO-PASCAL-u* sastoji od procesa, klasa i monitora koji su medjusobno povezani slijedećim naredbama *CO-PASCAL-a:*

NAREDBA	*ZNAČENJE*
COBEGIN S1, S2,..., Sn COEND	Naredbe od S1,.., Sn se mogu izvoditi paralelno.
BEGIN S1, S2,..., Sn END	Naredbe S1 do Sn se izvode sekvencijalno, jedna za drugom.

Ipak ova metoda paralelnog programiranja donosi sa sobom tri glavna problema:

- Odgovornost programera da detektira i izrazi sam sve moguće paralelizme u programu;
- Male promjene u programu mogu zahtjevati od programera da potpuno reorganizira uvedene paralelizme u programu;
- Programi koji su već napisani za sekvencijalnu obradu moraju biti ponovo napisani.

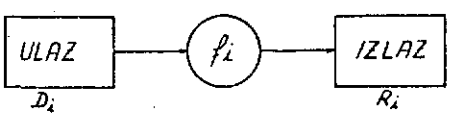
Ta je metoda zato još uvijek pogodna samo na sistemskom nivou.

Stoga je pogodno uvesti postupke automatskog uvodjenja paralelizma, preseljenjem tog zadatka sa aplikacionog programera na kompilator programskog jezika.

3. DETEKCIJA PARALELNIH PROCESA UNUTAR SEKVENCIJALNOG PROGRAMA

Ovdje će biti razradjeni postupci funkcijske dekompozicije programa na skup dijelova programa (procesa) koji se izvode paralelno ili sekvencijalno već prema postojećim odnosima koji su ovdje definirani.

Program koji se sastoji od jedne ili više naredbi može se rastaviti na jedan ili više dijelova (blokova), elementarnih procesa, kao na slici 1. Svaki elementarni proces provodi zapravo operaciju preslikavanja ulaznog skupa podataka u izlazni skup.

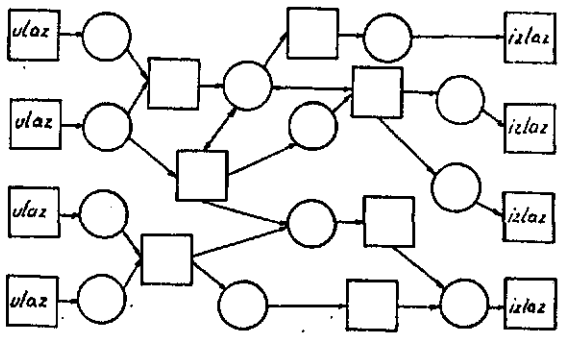


Slika 1. Elementarni proces Pi

Zapravo za svaki elementarni proces ili kraći proces Pi definira se domena Di (skup ulaznih podataka) i područje Ri (skup izlaznih podataka) te frekvencija preslikavanja fi.

Zadatak je funkcijske dekompozicije da se pronadju elementarni procesi, te da se odrede odnosi medju njima, s

time da se program može prikazati dijagramom elementarnih procesa kao na slici 2., od kojih se neki mogu izvoditi i paralelno.



Slika 2. Mreža elementarnih procesa - dekomponirani program

Ovdje će biti razradjene neke klase tih odnosa, postupci uvodjenja i verifikacije paralelizma u obradi programa u multi-mikroprocesorskom sistemu.

Kako je u multi-mikroprocesorskom sistemu, u obradi paralelnih procesa, moguće korištenje privatne i djeljive memorije potrebno je razmotriti dva slučaja:

a) Svi procesori koriste istu glavnu memoriju (djeljiva memorija);
b) Svaki procesor ima dodjeljenu privatnu memoriju u kojoj je smještena informacija koja se trenutno obradjuje od strane tog procesora. Kada je obrada završena informacija se obično vraća u glavnu memoriju.

Razlika se uočava ukoliko se dva procesa A i B izvode paralelno i A mijenja lokaciju i prije nego je dohvati proces B. Tada će u slučaju korištenja djeljive memorije proces B dohvatiti izmjenjenu vrijednost od strane procesa A, a u slučaju privatne memorije proces će dohvatiti vrijednost lokacije koja je tamo već prethodno bila smještena.

Zbog detekcije paralelizma u programu potrebno je razmotriti odnose unutar programa.

3.1. ODNOSI IZMEDJU DIJELOVA PROGRAMA

Prema načinu na koji procesi (instrukcije, dijelovi programa, programski blokovi, programi) koriste memorijske lokacije postoje četiri kategorije (W, X, Y i Z) lokacija:

W - Lokacije koja se samo dohvaća za vrijeme izvodjenja P;
X - Lokacija u koju se samo spremaju podaci za vrijeme izvodjenja P;
Y - Prva operacija nad tom lokacijom je operacija dohvata. Jedna od narednih operacija procesa P smješta podatak u tu lokaciju;
Z - Prva operacija procesa P nad tom lokacijom je operacija spremanja. Jedna od slijedećih operacija procesa P je operacija dohvata te lokacije.

Dijelove programa, radi lakšeg razumjevanja, nazvati ćemo u daljnjem tekstu blokovima. Uz pretpostavku da se program sastoji od N blokova, postoji 5 mogućih odnosa izmedju dva susjedna bloka B_i i B_{i+1} $(1 \leq i < N)$.

1. Preduvjetni blok B_i mora dohvatiti sve što zahtijeva prije nego B_{i+1} spremi svoje rezultate.
2. Konzervativni blok B_i mora spremiti svoje rezultate prije nego to učini blok B_{i+1}.
3. Komutativni blok B_i smije biti izveden prije ili poslije B_{i+1} ali ne u isto vrijeme.
4. Paralelni blokovi B_i i B_{i+1} mogu se izvoditi u isto vrijeme, a sve lokacije se mogu koristiti u proizvoljnom poretku.
5. Sekvencijalni blok B_i mora spremiti svoje rezultate prije nego B_{i+1} dohvati ono što zahtijeva.

Za svaki blok B_i sada možemo definirati skupove lokacija prema uvedenim definicijama:

W_i - Predstavlja skup svih lokacija koje se samo dohvaćaju za vrijeme izvodjenja bloka B_i.

X_i - Predstavlja skup svih lokacija u koje se samo spremaju podaci za vrijeme izvodjenja bloka B_i.

Y_i - Predstavlja skup svih lokacija za koje je prva operacija dohvata, a jedna od slijedećih operacija bloka B_i je operacija spremanja.

Z_i - Predstavlja skup svih lokacija za koje je prva operacija operacija spremanja, a jedna od slijedećih operacija bloka B_i je operacija dohvata lokacije.

Kako bi se umanjila kompleksnost posla uvedene su kratice:

WY_i - Predstavlja skup svih lokacija za koje se provodi samo operacija dohvata, a ne i operacija postaljanja u bloku B_i, domena D_i, dakle

$$WY_i = W_i \cup Y_i = D_i$$

XYZ_i - Predstavlja skup svih lokacija koje poprimaju novu vrijednost u B_i (područje R_i) dakle

$$XYZ_i = X_i \cup Y_i \cup Z_i = R_i$$

Takodjer je potrebno uvesti skup lokacija V koji predstavlja sve lokacije koje su dohvaćene prije nego su bile prepisane izvodjenjem blokova B_i i B_{i+1}.

Da bi se provela dekompozicija programa prema slici 2. dovoljno je razmotriti uvjete paralelnosti i sekvencijalnosti za slučaj korištenja djeljive i vlastite memorije u višeprocesorskom sistemu.

3.1.1. KORIŠTENJE VLASTITE MEMORIJE

1. UVJET PARALELNOSTI

Blokovi B_i i B_{i+1} mogu se izvoditi u isto vrijeme, a sadržaji lokacija se mogu dohvaćati i mijenjati u proizvoljnom poretku. Stoga ne postoji zavisnost između ulaza i izlaza od B_i i B_{i+1}, tj.

$$WY_i \cap XYZ_{i+1} = 0 \qquad (1.1)$$

$$XYZ_i \cap WY_{i+1} = 0 \qquad (1.2)$$

(0-nulti skup)

Lokacije koje se mijenjaju od oba bloka B_i i B_{i+1} ne smiju se koristiti nigdje dok se ne postave na svoju vrijednost, budući je vrijednost tih lokacija nedefinirana tj.

$$XYZ_i \cap XYZ_{i+1} \cap V = 0 .$$

Kako je

$$XYZ_i \cap XYZ_{i+1} = (X_i \cup Y_i \cup Z_i) \cap (X_{i+1} \cup Y_{i+1} \cup Z_{i+1}) = (X_i \cup Z_i) \cap (X_{i+1} \cup Z_{i+1})$$

slijedi dodatni uvjet paralelnosti dva bloka,

$$(X_i \cup Z_i) \cap (X_{i+1} \cup Z_{i+1}) \cap V = 0 \qquad (1.3)$$

2. UVJET SEKVENCIJALNOSTI

Blok B_i mora spremiti svoje rezultate prije nego B_{i+1} dohvati ono što zahtijeva, tj.

$$XYZ_i \cap WY_{i+1} \neq 0 \qquad (1.4)$$

3.1.2. KORIŠTENJE DJELJIVE MEMORIJE

1. UVJET PARALELNOSTI

Isti uvjet kao pod (1.1) i (1.2)

Parcijalni rezultati proizvedeni od B_i (B_{i+1}) ne smiju biti prepisani od B_{i+1} (B_i) tj.

$$Z_i \cap XYZ_{i+1} = 0 \qquad (1.5)$$

$$XYZ_i \cap Z_{i+1} = 0 \qquad (1.6)$$

Kombiniranjem (1.1) i (1.5)

$$(WY_i \cup Z_i) \cap XYZ_{i+1} = 0 \qquad (1.7)$$

Kombiniranjem (1.2) i (1.6)

$$XYZ_i \cap (WY_{i+1} \cup Z_{i+1}) = 0 \qquad (1.8)$$

Lokacije koje se mijenjaju u blokovima B_i i B_{i+1} ne smiju se koristiti bez prethodnog restauriranja,

$$XYZ_i \cap XYZ_{i+1} \cap V = 0 . \qquad (1.9)$$

Kako je,

$$XYZ_i \cap XYZ_{i+1} = (X_i \cup Y_i \cup Z_i) \cap (X_{i+1} \cup Y_{i+1} \cup Z_{i+1}) = X_i \cap X_{i+1}$$

slijedi

$$X_i \cap X_{i+1} \cap V = 0 . \qquad (1.10)$$

Uvjeti za paralelnost za ovaj slučaj su dakle (1.7), (1.8) i (1.10).

2. UVJET SEKVENCIJALNOSTI

Blok B_i mora spremiti svoje rezultate prije nego B_{i+1} dohvati što zahtijeva, tj.

$$XYZ_i \cap WY_{i+1} = 0$$

3.1.3. POOPĆENJE UVJETA

Moguće je proširiti ove odnose definirane za dva bloka i na n blokova koji su u nizu

$$\{B_1, B_2, \ldots\ldots, B_n\}$$

UVJET PARALELNOSTI

Blokovi $\{B_1, B_2, \ldots\ldots, B_n\}$ mogu se izvoditi u isto vrijeme. Poredak dohvata i spremanja podataka nema važnosti.

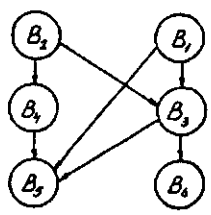
UVJET SEKVENCIJALNOSTI

Blok B_k mora biti završen prije izvođenja bloka B_{k+1} za svaki k $(1 \leq k < n)$.

3.2. GRAF PRVENSTVA

Nakon što je provedena dekompozicija programa ispitivanjem uvjeta paralelnosti i sekvencijalnosti, potrebno je iz dijagrama sa slike 2. proizvesti graf prvenstva. On

se dobiva tako da čvorovi tog grafa objedinjavaju funkciju preslikavanja, ulazno izlazne podatke i kontrolnu informaciju, tj. oni predstavljaju blokove programa B_i. Tako se dobije graf prvenstva sa slike 3.



Slika 3. Graf prvenstva

Potrebno je napomenuti da strelica usmjerena od čvora B_i prema B_{i+1} znači da se blok programa B_i mora izvesti prije bloka B_{i+1} (sekvencijalno). Nepostojanje strelice medju čvorovima znači, ukoliko ne postoji zahtjev sekvencijalnosti, da se blokovi mogu izvoditi paralelno. Kontrolna informacija u čvorovima potrebna je za održavanje sinhronizacije u izvedjenju medju blokovima, što uz uvjet sekvencijalnosti znači da izvodjenje nekog bloka B_i ne može započeti dok ne završe svi blokovi iz kojih izlaze strelice, a koje ulaze u čvor B_i.

Da bi postupak uvodjenja paralelizma bio potpuno ispravan potrebno je provesti ispitivanje odredjenosti ovakovog sistema procesa (blokova). Sistem procesa (blokova) naziva se odredjenim ako rezultat koji taj skup blokova proizvodi ne ovisi o brzini i redosljedu izvodjenja procesa. Verifikacija odredjenosti provodi se postupcima opisanim u [2] i [6]. Odredjenost sistema procesa postiže se sinhronizacijom izvodjenja blokova na osnovu ispitivanja uvjeta sekvencijalnosti i paralelnosti, te na osnovi ispitivanja odredjenosti sistema procesa.

3.3. IMPLEMENTACIJA

Kako bi se mogli realizirati postupci za uvodjenje paralelizma u sekvencijalnim programima moguće je proširiti višeprolazne kompilatore sa još dvije faze, analizator i detektor. Medjusobne odnose dijelova programa potrebno je na osnovu uvedenih uvjeta posebno analizirati za pojedinačne višeprogramske jezike (ALGOL, FORTRAN, PASCAL, ...) i to posebno za one naredbe koje već implicitno po svojoj sintaksi uključuju blokove programa (petlje, uvjetne naredbe, procedure,...).

ANALIZATOR

Ta faza kompilacije dijeli programe na blokove. Analizator proizvoljno limitira veličinu bloka na odredjenu programsku konstrukciju (npr. petlju) ili skupinu naredbi koje ne koriste više od odredjenog broja različitih naredbi. Varijable unutar blokova se klasificiraju u skupove W, X, Y i Z ovisno o načinu njihovog korištenja.

DETEKTOR

U toj fazi kompilacije uzimaju se blokovi koje je proizveo analizator, te se nad njima provode testovi koji odredjuju paralelnost i sekvencijalnost medju blokovima. Tu se ugradjuje i kontrolna informacija neophodna za sinhronizaciju izvodjenja blokova. U ovoj fazi je također potrebno provesti verifikaciju odredjenosti sistema procesa (blokova) postupcima uvedenim u [2] i [6].

Tako se za slučaj primjene ovog postupka na sekvencijalni program u PASCAL-u kao rezultat dobije graf prvenstva koji u biti odgovara transformaciji sekvencijalnog PASCAL programa u program napisan u CO-PASCAL-u, što je i bio zadatak ovog postupka.

4. ZAKLJUČAK

Jasno je da obje metode uvodjenja paralelizma zahtjevaju značajan razvoj i unapredjenje sistemske programske podrške, tj. kompilatora. Upotrebom postojeće tehnologije ti zahtjevi nisu nesavladivi. Proširenjem mogućnosti kompilatora programskih jezika oslobadja se aplikacioni inženjer od teškog zadatka prilagodjavanja programa računarskom sistemu, a samim time multi-mikroprocesorski sistemi postaju moćniji i jednostavniji za korištenje.

REFERENCE:

[1] K. Jackson, C. I. Molr; Parallel processing in software and hardware, Sagamore Computer Conference of Parallel Processing, 1975.

[2] I. Margana, A. Raffaele, M. Zacchi; A new scheme for analyzing parallel processing systems, Sagamore Conference on Parallel Processing, 1975.

[3] D. J. Evans, S. A. Williams; Analysis and detection of parallel processable code. The Computer Journal Vol 23, No 1.

[4] J. P. Bañatre, J. P. Ronteau, L. Trilling; An Event-Driven Compiling Technique, Communications of the ACM, Vol 22, No 1.

[5] D. A. Prener; Some Software Considerations for large multi-processor systems, International Microcomputers, Minicomputers, Microprocessors 79, Conference Proceedings.

[6] A. Shaw; Logic Design of Operating Systems, Prentice-Hall, New Jersey, 1974.

[7] N. Hadjina; Princip izgradnje sistemskih programskih komponenti u multi-mikroprocesorskim sistemima, Informatica 1979., br. 4.

POSTOPKI ZA POVEČANJE ZANESLJIVOSTI DIGITALNIH SISTEMOV

R. MURN

INSTITUT J. STEFAN, LJUBLJANA

UDK: 681.3.519.718

Članek obranava osnovne postopke za povečanje zanesljivosti digitalnih sistemov, ki so osnovani na načelu dopuščanja napak.Opisani so pomembnejši postopki kot so redundantni postopki,detekcijski in korekcijski kodi ter vezja,ki omogočajo lastno preizkušanje.Učinkovitost detekcijskih in korekcijskih kodov je prikazana na primeru podsistemov kot so polprevodniški pomnilniki.Obravnavana so tudi sodobna integrirana vezja,ki imajo zmožnost lastnega testiranja in vezja,ki so posebej namenjena za diagnostiko napak.

METHODS FOR IMPROVING RELIABILITY OF DIGITAL SYSTEMS. Basic concepts and Tecniques of fault tolerance are discussed in this paper.In particular,the paper covers redundancy techniques,coding techniqes and the design of self-checking code checkers; and the design of fault-tolerant computer subsystems such as semiconductor memory. The paper also surveys recent techniques for incorporating self-test and fault-tolerant features into LSI and VLSI components.

1.UVOD

Digitalne sisteme s sodobnimi polprevodniškimi elementi in mikroprocesorji najdemo v vedno večji meri na takih mestih,kjer je zanesljivost delovanja ključnega pomena.Nekontrolirani učinki napak in različne odpovedi lahko povzroče nezaželen izpad določenega opravila.Zanesljivost,pripravljenost sistema ter vzdrževanje sistema predstavljajo izredno pomembne lastnosti.Napak v sistemu v splošnem ne moremo popolnoma odpraviti,lahko pa izboljšamo omenjene lastnosti.Zasnova dopuščanja napak(Fault-Tolerance)v digitalnih sistemih združuje različne postopke kot so:detekcija in korekcija napak,vezja,ki omogočajo lastno testiranje,redundantnost in obnovitvene postopke.Izbira postopkov je odvisna predvsem od arhitekture sistema,kompleksnosti sistema,dopustne cene,stopnje sistemove zanesljivosti in časa,ki je potreben za ponovno vzpostavitev v normalno stanje.

Sodobni polprevodniški elementi,zlasti integrirani elementi zelo visoke stopnje integracije narekujejo veliko bolj dinamično obravnavo postopkov za povečanje zanesljivosti kot je bilo to v preteklosti.

V članku želimo osvetliti nekatere pomembnejše pojme in področja zanesljivosti digitalnih sistemov.Ogledali si bomo tudi osnovne postopke kot so redundantni postopki,postopke na osnovi detekcijskih in korekcijskih kodov,skupaj z vezji,ki imajo zmožnost testiranja samega sebe in pa nekatere tendence pri razvoju sodobnih integriranih vezij zelo visoke integracije.

2. OSNOVNE PREDPOSTAVKE

Sistem deluje brez napak določen čas, ki ga imenujemo srednji čas med napakami(izpadi) -MTBF.Ko se napaka pojavi,največkrat nima takojšni učinek na pravilnost delovanja sistema. Prav tako se vpliv napake zazna(detektira)šele po določenem času.Srednji čas med zaznavo vpliva napake in detekcijo imenujemo srednji čas do detekcije-MTD.Ko napako detektiramo jo lokaliziramo in popravimo.Čas za to opravilo označimo s srednjim časom do popravila-MTR.Običajno je mišljeno popravilo kot zamenjava določenega zamenljivega elementa ali modula.Čas od popra-

vila oziroma zamenjave pa do vpliva naslednje napake imenujemo pripravljenost sistema.Na pripravljenost sistema torej vplivata poleg MTBF tudi MTD inMTR.V večina sistemih dodajamo komponente,module in programsko opremo,da izboljšamo tako MTD kot MTR.

Osnovni način izboljšanja MTBF sistema bi bilo kar načelo zmanjšanja števila komponent(manj komponent-manjše število možnih napak) in pa vpeljava zanesljivih komponent.Naslednja pot za nadaljno izboljšanje zanesljivosti je vpeljava raznih oblik redundanc(rezerviranosti).

Kot pomembna se izkaže vpeljava pojma "varen za napake".Pravimo,da je sistem varen za napake,če se pojavi zaradi napake napačen izhod samo v primeru,ko se istočasno zazna prisotnost napake.Detekcijski in korekcijski kodi in vezja,ki omogočajo lastno testiranje predstavljajo uspešna sredstva za izboljšanje MTD,MTR in varnost za napake digitalnih sistemov.

Vrednotenje vpeljave vseh postopkov v celoten sistem izvedemo s pomočjo analitičnega modeliranja,simulacije ali pa s kombinacijo obeh načinov.Običajno se primerja ocena zanesljivosti z začetnim sistemom,kjer še nismo vpeljali postopkov za povečanje zanesljivosti. Vrednotenje zanesljivosti posameznih pod enot in postopkov je najti v veliki meri v literaturi.

Pri razvoju moramo v vsakem primeru upoštevati model in značaj napake,ki jo dopuščamo. Če si odmislimo človeške napake in razvojne napake,nas predvsem zanimajo fizične napake.Pod pojmom napaka bomo smatrali kako odstopanje zahtevane vrednosti logičnih spremenljivk v sistemu.Napake se lahko širijo po sistemu in povzročajo odpoved sistema.

Napake so lahko stalne(v času obravnave) t.j.Stalno v 1 ali Stalno v 0(Stuck-at) in nestalne t.j. take,ki povzroče vmesne logične vrednosti.

Razlikujemo enojne in večkratne napake. Slednje vpljivajo na več logičnih spremenljivk. Večkratne napake imajo lahko vse enako logično vrednost-enakovrstne napake.

V sistemih se lahko pojavijo tudi trenutne napake.Naslednji trenutek napake ni več,vendar je njena pojavitev lahko povzročila odpoved sistema.

3. REDUNDANTNI POSTOPKI

Na splošno lahko delimo redudantne (rezervirne)postopke na dve različni kategoriji:
a)na hardwarske(dodajanje komponent) in
b)na programske (posebni programi).

a) Hardwarske redundance:

Na splošno lahko to kategorijo redundanc delimo na dve skupini in sicer na statične redundance in na dinamične redundance.

Statične redundance imenujemo tudi maskirne redundance,ker se uvedejo rezervne komponente za maskiranje učinkov hardwarskih napak. Rezervni elementi so stalno priključeni in aktivni ter izvršijo maskiranje v trenutku in avtomatično.Posebnih detekcijskih in korekcijskih korakov ne potrebujemo.

Maskirne redundance vpeljujemo v sistem na različne načine,od nivoja posameznih komponent do nivoja sistema.Na nivoju komponent je splošno uporabljena tehnika"štirih elementov". Npr.diodo nadomestimo s štirimi diodami(dve vsporedni veji s po dvema diodama),pri tem prekinitev ali kratek stik posamezne diode ne vpliva na funkcionalnost vezja.

Sodobne mikroračunalniške komponente(mikro procesorji,nizka cena)so ponovno spodbudile večjo uporabo redundanc na nivoju sistema.Sistemi z dubliranjem,trojno modularne redundance z majoritetnim rezerviranjem-TMR,/1/ in splošne N-modularne redundance-NMR so pogosto v uporabi.

Vmesni nivo predstavljajo korekcijski kodi,le-ti maskirajo napake na posameznih bitih brez večkratnih rezervirnih modulov.

Dinamične redundance vsebujejo rezervne module,ki jih vključimo v sistem z namenom,da zamenjajo aktivne module,ko se ti pokvarijo. Pri tem se običajno zahtevajo neposredni koraki kot so:detekcija napak,lokalizacija napak in popravilo.Vpeljava dinamičnih redundanc zahteva številne odločitve v fazi funkcionalnega razvoja digitalnega sistema,ozir.računalnikov. Glede na oceno primernosti,lahko uvedemo kombinacijo statičnih in dinamičnih redundanc.

Redundančni postopki so učinkoviti le toliko časa,dokler ni prekoračena stopnja rezerviranosti modulov,kar ugotavljamo s posebnimi detekcijskimi postopki.

b) Programske redundance:Vsebujejo dodatne programe,navodila in mikro instrukcije,ki jih sicer ne vsebujejo običajni računalniški sistemi. Ti podatki služijo ali za detekcijo napak ali pa za obnovitveni postopek,največkrat skupaj z dinamičnimi redundancami.Uporabljajo se tri glavne oblike programskih redundanc:1)večkratno shranjevanje pomembnejših programov in podatkov,2)testni programi ali mikroprogrami,3) deli operacijskega sistema,ki vzajemno delujejo s hardwarskimi redundancami in omogočajo restart programa.

Pri večini modernih računalniških(fault-

tolerant) sistemih zasledimo kombinacijo vseh treh oblik.

V literaturi pogosto zasledimo ocene in izračune zanesljivosti posameznih komponent in enot.V članku/2/ je npr.prikazan pristop izgradnje programa za izračun zanesljivosti različnih modelov pomnilniških sistemov.

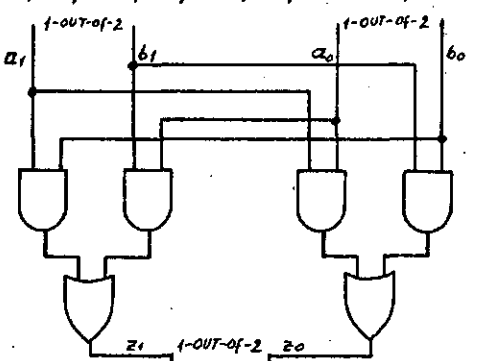
4. DETEKCIJSKI KODI IN VEZJE ZA LASTNO TESTIRANJE

Detekcijski kodi in vezja za lastno testiranje predstavljajo učinkovito sredstvo za izboljšanje kriterijev zanesljivosti digitalnih sistemov.Detekcijski kod predstavlja redundantno kodiranje podatkov,ki omogoča detekcijo napak.Vezje za lastno testiranje je logično vezje,ki omogoča obravnavo informacij zakodiranih v detekcijskem kodu.S pravilnim razvojem vezja lahko dosežemo detekcijo večine možnih napak na kodiranem izhodu vezja.Informacija o detekciji napake je torej neločljiv del izhoda vezja.V vsakem trenutku,ko dobimo na izhodu novo informacijo je možno ugotoviti ali je izhod pravilen ali pa zaradi napak modificiran.Izhodu vezja za lastno testiranje je dodano še vezje,ki javi prisotnost nepravilnega izhoda.

Lastnost kodov za detekcijo in korekcijo napak so poznane že dalj časa,cilj sodobnih raziskav pa je razviti postopke ,ki bi bili učinkoviti pri sedanjih in bodočih digitalnih sistemih in tehnologijah.Pripadajoča testirna vezja naj bi bila za velikostni razred bolj zanesljiva od funkcionalnega sistema ali pa tako izvedena,da se v celoten detekcijski mehanizem vključujejo tudi napake v testnem vezju samem.

Izbira ustreznih kodov je pogojena s specifičnimi lastnostmi digitalnih sistemov in z značajem napak.Pionirsko delo v razvoju kodnih sistemov predstavlja publikacija Hamminga v letu 1950 /3/,ki je predstavil razred korekcijskih kodov za enojne napake in so dobili po njemu tudi ime.Enostavni eno-bitni kodi za preizkušanje parnosti so tudi danes pogosto v uporabi.Kodi s konstantno utežjo ali m-out-of-n kodi so zelo popularni/4/.(v besedah z n bitii,mora biti natančno m bitov z vrednostjo 1;kod je nesimetričen).Kod omogoča detekcijo tako enojnih napak kot enakovrstnih večkratnih napak(enakovrstne večkratne napake imajo lastnost,da so vse stalno 1 ali stalno 0). Podobno zmožnost kot gornji imajo Bergerjevi kodi, s to prednostjo,da so tudi separacijski kodi,to pomeni,da ne potrebujemo posebnega vezja za ločitev podatkovnih bitov od preizkuševalnih.Seveda obstaja še vrsta namensko usmerjenih kodov kot so npr.ciklični kodi,aritmetični kodi itd.

Na sliki 1 imamo primer vezja za ugotavljanje enakosti(komparator) dveh skupin bitov,pri tem je zagotovljeno,da se kaka enojna ali enakovrstna napaka v samem vezju vedno detektira na izhodu.Vezje/5/ preslika oba vhodna para v en izhodni par tako,da je izhodni par 1-out-of-2 kod samo,če sta oba vhodna para tudi 1-out-of-2 koda(pravilni izhod ima vrednost 01 in 10 in napačen 00 in 11).Normalni vhodi so:0101/01,0110/10,1001/10, in 1010/01.

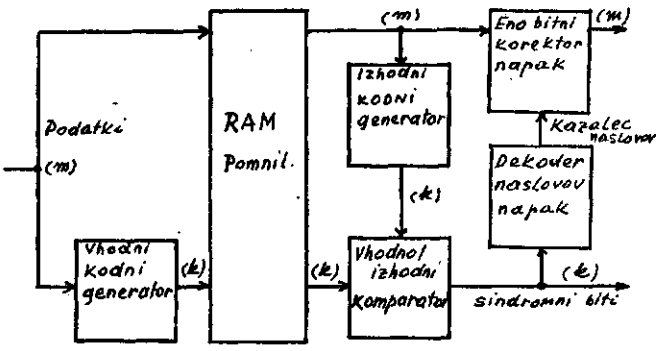


Slika 1.Komparatorsko vezje z zmožnostjo lastnega testiranja

5.KOREKCIJA NAPAK PRI POMNILNIŠKIH SISTEMIH

Detekcijski in korekcijski kodi se uspešno uporabljajo pri sodobnih polprevodniških pomnilnikih,zlasti pri večjih modulih.Največ se uporabljajo kodi z zmožnostjo korekcije enojne napake in detekcije dvojne napake(SEC / DED).Omenjeni kodni postopki so osnovani na Hammingovem kodu za korekcijo enojne napake /3/.Kot se vidi na sliki 2 se na vhodu vkodira m podatkovnim bitom polje k parnostnih bitov in shrani celotno število n=m+k bitov.Pri čitanju iz pomnilnika se ponovno generira polje k parnostnih bitov,ki se primerjajo s prvotno shranjenem poljem k parnostnih bitov.Če se polji ujemata ni napake,sindromni biti so enaki nič.Če pa se polji razlikujeta,zaznamo napako. Korekcijsko logično vezje potem obrne napačen bit,tako da dobimo zopet pravilno podatkovno besedo.

Pomembna pa je tudi oranizacija pomnilniških enot.Željeno je ,da pripada posamezna RAM enota enemu bitu v pomnilniški besedi tako, da vpliva napaka v RAM enoti samo na pripadajoči bit pomnilniške besede.Z opisanim postopkom detektiramo napake v celotnem pomnilniku.



Slika 2.Blokovna shema korekcije in detekcije napak

Vpeljava detekcijskih in korekcijskih kodov ima neposreden vpliv na nekatere pomembne lastnosti kot so:cena pomnilnika,hitrost in zanesljivost.

Minimalno število redundantnih bitov k,ki jih dodamo za korekcijo enojne napake določimo s pomočjo enačbe

$$2^k \geq m + k + 1 \quad .$$

Za detekcijo dvojne napake pa še dodamo en bit. (minimalna Hammingova razdalja je 4).Iz spodnje razpredelnice lahko razvidimo,da je postopek vedno bolj ekonomičen,če povečujemo število podatkovnih bitov na besedo.

Korekcija enojne/detekcija dvojne napake		
Dolžina podatkovnih bitov m	Preizkuševalni biti k	Povečanje %
4	4	1oo.o
8	5	62.5
10	5	5o.o
12	6	5o.o
16	6	37.5
18	6	33.3
20	6	3o.o
24	6	25.o
32	7	21.9
64	8	12.5
96	8	8.3

Seveda pa zahteva postopek dodatna vezja kar vpliva na čas pomnilniškega cikla,ki je odvisen tudi od uporabljene tehnologije elektronskih komponent.

Oceno izboljšanja zanesljivosti v pomnilniku z vpeljavo korekcije napake lahko podamo z razmerjem $R = P_1 / P_2$, kjer je P_1 verjetnost enobitne napake brez korekcije in P_2 verjetnost dvojne napake s korekcijo(k bitov dodanih).

$$P_1 = mR^{(m-1)}(1-R), \qquad \text{po /6/}$$

kjer je $R = e^{-\lambda t}$ in λ- pogostost izpadov v čipu.

$$P_2 = \frac{(m+k)(m+k-1)}{2} R^{(m+k-2)} (1-R)^2$$

$$R = \frac{2m}{(m+k)(m+k-1)} \cdot \frac{1}{R^{(k-1)}(1-R)}$$

Za zgled vzemimo naslednje podatke:

m=32 bitov t= 1ooo ur

k= 7 bitov $\lambda = 10^{-6}$ napak na uro

Po vstavitvi podatkov v izraz R,dobimo,da je pomnilniški sistem z enojno korekcijo 43 krat bolj zanesljiv od sistema brez korekcije.

Bistvena prednost avtomatične korekcije enojne napake pa se kaže pri trenutnih napakah,kar je še posebno pomembno pri dinamičnih pomnilnikih.

6. SODOBNA TESTIRNA INTEGRIRANA VEZJA

Vedno pogostejša uporaba integriranih vezij zelo visoke integracije (VLSI) v računalniške sisteme otežuje učinkovito uvajanje klasičnih postopkov za povečanje zanesljivosti. Rešitve se kažejo v razvoju modificiranih postopkov ter v razvoju VLSI vezij,ki imajo vgrajene diagnostične postopke in v posebnih diagnostičnih integriranih vezjih.

Na področju razvoja vezij z vgrajenimi postopki za izboljšanje zanesljivosti omenjamo zamisel,ki sta jo predlagala Sedmak in Liebergot /7/. Predlagana arhitektura vsebuje funkcionalno in dublirano vezje s komparatorjem. Dublirano vezje se razlikuje od običajnih dubliranih vezij v tem,da je le-to logični komplement funkcionalnega vezja.Zaradi različnosti obeh razvojev se izognemo identičnim napakam,ki enako vplivajo na obe vezji.Komparator ima zmožnost lastnega testiranja in je lahko enak vezju iz slike 1.Izkaže pa se,da je težko izdelati komplementarno sekvenčno vezje, pa tudi raznim nezaželjenim časovnim in sinhronizacijskim problemom se s težavo obranimo.

Proizvodnja posebnih diagnostičnih integriranih vezij je v zadnjem času v stalnem porastu.Na tržišču se pojavljajo novi proizvodi,ki največkrat predstavljajo kompleksen sistem in v veliki meri nadomestijo številne

komponente nižje integracije.Omenili bomo le nekatere od teh: Podjetje Signetics je izdelalo večnamensko integrirano vezje 2653,ki nam lahko uspešno služi kot polinomski preizkuševalni generator za vertikalno,longitudinalno in CRC ugotavljanje napak.Pri podjetju Advanced Micro Devices Inc. so izdelali v n-MOS tehnologiji procesorsko vezje AmZ 8065 za diagnostične namene pri magnetnih diskih. Vezje detektira tako enojne kot skupinske napake(12-bitne skupinske napake) v serijskem pretoku podatkov pri pogostosti do 20 miljonov bitov na sekundo.Isto podjetje je razvilo tudi novo vezje v enem čipu za korekcijo enojnih in detekcijo dvojnih napak za 16-bitno podatkovno polje.Integrirano vezje 2960 uporablja modificiran Hammingov kod,realizirano pa je v EC tehnologiji s TTL-kompatibilnimi vhodi in izhodi.

7. SKLEP

Problematika zanesljivosti digitalnih sistemih ni nova,stara je kot prvi digitalni računalniki.Res pa je,da so se zahteve zanesljivosti znatno povečale v začetku 1960 leta, ko so se začeli izvajati vesoljski programi.V preteklosti je bilo to področje nekoliko odmaknjeno od širše strokovne javnosti,kar je imelo za posledico pomankljivo obravnavo v literaturi.

Z razvojem nove tehnologije,zlasti mikroprocesorjev in cene teh so se razširile aplikacije na vsa mogoča področja človekove dejavnosti. Zaradi kompleksnosti digitalnih sistemov in sodobnih elementov je potrebno razviti nove in učinkovite postopke za povečanje zanesljivosti. Neizbežni problematiki dagnostike napak se moramo posvetiti že pri razvoju sistemov s ciljem obvladovanja razvojnih,konstrukcijskih in operacijskih napak.Veliko razvojnega napora pa bo potrebno vložiti tudi v problematiko razvoja zanesljive programske opreme,ki je v literaturi dokaj slabo obdelana.

Namen članka je seznaniti bralca s tem vedno bolj aktualnim področjem ter pokazati na rešitve za povečanje zanesljivosti nekaterih podsistemov.

LITERATURA

/1/ - A.E.Cooper in W.T.Chow;Development of on board space computer systems,IBM J.Res. Develop.,vol.20,April 1976.

/2/ - L.Budin,Ž.Nožica,V.Peruško,D.Vuković; Program za proračun pouzdanosti različitih memorijskih sistema,Informatica,Št.2 1979.

/3/ - R.W.Hamming;Error detecting and error correcting codes,The Bell System Tech. Journal,No.2,April 1950

/4/ - D.A.Anderson in G.Meze;Design of totally self-checking check circuits for m-out-of-n codes,IEEE Tr.,March 1973.

/5/ - D.A Anderson;Design of self-checking digital networks using coding tecniques, Tech.Rpt. R-527,Urbana Univers.Illinois.

/6/ - L.Levine,W.Meyers;Semiconductor memory reability with error detecting and correcting codes,Computer,October,1976.

/7/ - R.M.Sedmak in H.L.Liebergot;Fault-tolerance of a general purpose computer implemented by very large scale integration,FTCS - 8,conference,1978.

DELJENJE SLOVENSKIH BESED V PROFESORJIH TEKSTA

B. KASTELIC
D. NOVAK

UDK: 681.3.01 : 806.63

IJS LJUBLJANA

Članek opisuje program za deljenje slovenskih besed, ki je lahko vključen v procesor teksta. Opisano je delovanje programa, važnejša pravila, ki jih program upošteva in napake, ki se jih ne da odpraviti. Program je realiziran na mikroračunalniku s procesorjem M6800 in pravilno deli slovenske besede z 99% zanesljivostjo.

DIVISION OF SLOVENE WORDS ON TEXT PROCESSORS. The article describes a programme on division of Slovene words, which can be included in a text processor, then, the activity of programme, outstanding rules, considered by the programme and mistakes that can not be dispatched. The programme is realised on the microcomputer with processor M6800 and divides the words correctly wtih a 99% certitude.

1. UVOD

Uporaba procesorjev teksta (Text processor) se je pri nas že dobro razširila v vsakdanje življenje. Bistvena lastnost procesorja teksta je med drugim tudi oblikovanje prosto vnešenega teksta v stolpce, ki so na obeh straneh poravnani. Ker procesor teksta ne zna deliti slovenskih besed, poravna obe strani stolpca tako, da besede ustrezno razmakne med seboj z vnašanjem dodatnih presledkov. Tako pride v primerih, ko je beseda, ki ne gre več v vrstico, daljša, pri ožjih stolpcih do nenormalno dolgih presledkov. Besedilo je sicer levo in desno poravnano, vendar je branje teksta s široko razmaknjenimi besedami naporno. Procesor teksta z deljenjem besed oblikuje estetsko lepše stolpce brez pretiranih presledkov med besedami. V članku je opisano delovanje programa za deljenje slovenskih besed. Program lahko dokaj enostavno vključimo v procesor teksta.

2. DELJENJE SLOVENSKIH BESED

Po Slovenskem pravopisu delimo nesestavljene besede takole:

a) Soglasnik ali svočnik med samoglasnikoma jemljemo k prihodnjemu slogu: de-lo-va-ti, do-mo-vi-na, kr-ti-na, ve-tr-ni-ca, no-soj-en, bol-ha.

b) Soglasniški sklop, ki se da brez sile izgovoriti, lahko ves vzamemo k prihodnjemu slogu, lahko pa tudi pridržimo prvi soglasnik pri prejšnjemu slogu: do-bra, pre-kla, go-dla, mi-šljen, še-plja, pa tudi lahko: dob-ra, prek-lja, god-lja, miš-ljen, češ-plja.

c) Od težko izgovorljivega soglasniškega sklopa jemljemo izgovorljivi del k naslednjemu slogu: jer-bas, var-stvo, stol-ni, valj-čast, gum-bi, gan-ljiv.

Uporaba procesorjev teksta (Text processor) se je pri nas že dobro razširila v vsakdanje življenje. Bistvena lastnost procesorja teksta je med drugim tudi oblikovanje prosto vnešenega teksta v stolpce, ki so na obeh straneh poravnani. Ker procesor teksta ne zna deliti slovenskih besed, poravna obe strani stolpca tako, da besede ustrezno razmakne med seboj z vnašanjem dodatnih presledkov. Tako pride v primerih, ko je beseda, ki ne gre več v vrstico, daljša, pri ožjih stolpcih do nenormalno dolgih presledkov.

Uporaba procesorjev teksta (Text processor) se je pri nas že dobro razširila v vsakdanje življenje. Bistvena lastnost procesorja teksta je med drugim tudi oblikovanje prosto vnešenega teksta v stolpce, ki so na obeh straneh poravnani. Ker procesor teksta ne zna deliti slovenskih besed, poravna obe strani stolpca tako, da besede ustrezno razmakne med seboj z vnašanjem dodatnih presledkov. Tako pride v primerih, ko je beseda, ki ne gre več v vrstico, daljša, pri ožjih stolpcih do nenormalno dolgih presledkov.

SLIKA 1: Razlika med tekstom, oblikovanim s procesorjem teksta z deljenjem in tekstom, oblikovanim s procesorjem teksta brez deljenja je očitna.

č) Zaporedna samoglasnika lahko delimo: Le-on, Di-ana, ide-al, te-ori-ja.

V sestavljenkah delimo predpono od osnovne besede: bres-glav, med-vojen, is-daja, do-stopen, čez-meren, na-brati, nad-moški.

Zloženke najbolje delimo po sestavinah: vi-no-grad, kolo-vos, trdo-glav, vroče-kriven.

3. OPIS PROGRAMA

Važnejša pravila, ki so vnešena v program za deljenje slovenskih besed:

- soglasnik med dvema samoglasnikoma jemlje k naslednjemu zlogu.
- deli med dvema samoglasnikoma.
- deli med dvema enakima soglasnikoma.
- ne deli lahko izgovorljivih soglasniških sklopov: lj, nj, ks, pr, tr, kr,...
- ne deli samoglasnik na začetku besede.
- zaporedje samo-, so-, so- in samoglasnik deli med soglasnikoma, ko je to težko izgovorljivi soglasniški sklop.
- upošteva sestavljenke, ki se začnejo z: na, nad, naj, po, pod, za in ne. Predlogi, ki se končujejo s soglasnikom, v sestavljenkah niso kritični.
- soglasnik R smatra kot samoglasnik, ko se nahaja med dvema soglasnikoma.

Program za deljenje slovenskih besed ne upošteva izjem, ki jih zgornja pravila ne zajemajo oziroma so v nasprotju z njimi.

Program uporablja dva vmesna pomnilnika: v prvega zapišemo besedo, ki jo želimo deliti, drugi je namenjen za shranjevanje informacij o posameznih znakih besede. Za besedo smatra vse znake med dvema presledkoma. V besedo je lahko vključeno tudi ločilo, katerega pri deljenju ne upošteva. Računalnik najprej označi vse črke, ki se pojavljajo v besedi, nato označi samoglasnike. Soglasnik R označi kot samoglasnik, ko se nahaja med dvema soglasnikoma.

N A S L E D N J I .

1. vm. pom.	4E	41	53	4C	45	44	4E	4A	49	2E	00	..
2. vm. pom.	00	00	00	00	00	00	00	00	00	00	00	..

SLIKA 2: Slika prikazuje oba vmesna pomnilnika, s pomočjo katerih deli računalnik slovensko besedo. V zgornjem vmesnem pomnilniku je zapisana v ASCII znakih beseda, ki jo hočemo deliti. Na vseh ostalih lokacijah so ničle. To je stanje pred deljenjem.

N A S L E D N J I .

1. vm. pom.	4E	41	53	4C	45	44	4E	4A	49	2E	00	..
2. vm. pom.	08	88	08	08	88	08	08	08	88	00	00	..

SLIKA 3: V drugi vmesni pomnilnik se zapišejo najprej informacije o posameznih znakih. S tretjim bitom so označene črke, s sedmim bitom samoglasniki.

N A S L E D N J I .

1. vm. pom.	4E	41	53	4C	45	44	4E	4A	49	2E	00	..
2. vm. pom.	08	89	08	08	88	09	08	08	88	00	00	..

SLIKA 4: Po deljenju je rezultat deljenja vpisan v drugem vmesnem pomnilniku (bit nič). Besedo "NASLEDNJI." torej delimo "NA-SLED-NJI.".

Deljenje je razdeljeno na dva dela:

- delitev prvega zloga z upoštevanjem sestavljenk z že omenjenimi predlogi;
- delitev ostalih zlogov.

Rezultati deljenja se zapišejo v drugi vmesni pomnilnik. V kolikor je ta program vključen v procesor teksta, je potrebno razdeljeno besedo še pravilno prepisati v ustrezne vmesne pomnilnike.

4. VKLJUČITEV V PROCESOR TEKSTA

Z vključitvijo programa za deljenje slovenskih besed v procesor teksta, ki je opisan v članku (2), ostane njegovo delovanje nespremenjeno. Dodana sta le dva ukaza:

.DE ukaz ne lomi vrstice in omogoča v nadaljnjem tekstu deljenje besed

.ND ukaz ne lomi vrstice in prepoveduje v nadaljnjem tekstu deljenje besed

Prepoved deljenja uporabimo pri kraticah, besedah, ki jih procesor teksta napačno deli itd.

5. NAPAKE

Napake se pojavljajo pri sestavljenkah in izjemah, kjer se pri sklopu dveh smiselnih sestavin v eno besedo pojavljajo najrazličnejše kombinacije črk. Pravopis zahteva, da jih delimo po sestavinah, kar pa ni mogoče zajeti v splošna pravila.

Za primer vzamimo besedi "podvoz" in "podvojen". Računalnik pravilno deli:

pod-voz,

toda napačno:

pod-vo-jen.

Prav tako beseda "podrobiti". Računalnik jo deli:

pod-ro-bi-ti.

To je pravilno v smislu, ko je beseda sestavljena iz predloga "pod" in glagola "robiti", vendar napačno, ko je beseda sestavljena iz predloga "po" in glagola "drobiti".

Tudi izjemo "arheolog" deli napačno

ar-he-olog,

ker upošteva pravilo, da vedno deli med dvema samoglasnikoma.

To so napake, ki se jim z računalnikom ne moremo izogniti. Te napake zahtevajo korekturo operaterja. Korekcije omogoča procesor teksta s prepovedjo deljenja kritične besede ali z ročnim deljenjem.

Opisani program je po prvih ocenah sposoben pravilno deliti slovenske besede z vsaj 99% zanesljivostjo, kar je vsekakor, vsaj s tehnične plati, dovolj dobro. Zanesljivost bi se verjetno lahko z redno uporabo in analizo napak, ob manjši korekciji programa, še povečala.

6. LITERATURA

1. Slovenski pravopis, DZS, Ljubljana 1962

2. A.P. Železnikar: Procesiranje teksta z mikroračunalniki, INFORMATICA, 3/79, str. 48-57

UNIVERZALNI PROGRAMATOR ZA PROGRAMIRANJE BIPOLARNIH PROM–OV IN PLA–JEV

F. ŠTRAVS
M. DRUŽOVEC
M. GERKEŠ
V. ŽUMER

UDK: 681.327.28 – 503.55

VISOKA TEHNIŠKA ŠOLA MARIBOR, JUGOSLAVIJA

Programiranje bipolarnih vezij je sorazmerno zahtevno, ker so predpisani programski impulzi dokaj različnih oblik. Postopki programiranja se ne dajo poenotiti, ker uporablja vsak proizvajalec nekoliko drugačno tehnologijo. Zato mora biti programator sposoben generirati več različnih vrst impulzov, ki se naj v odvisnosti od zahtev ustrezno oblikujejo. V članku je dana izvedba univerzalnega in modularno zgrajenega programatorja s programskim generiranjem impulzov. Za različne vrste vezij je potrebno pognati posamezne programske module, ki se izvajajo z mikroračunalnikom družine 6800. Kot zgled dodajmo programiranje PROM-a 74 S 288 tako, da izvaja funkcijo naslovnega krmilnika IDM 29811.

UNIVERSAL PROGRAMMER FOR PROGRAMMING OF BIPOLAR PROMS AND PLAS: The programming of bipolar circuits is a relatively pretentious task since the required pulses have to be shaped differently in accordance with technology differencies of the manufacturers and can not be unified. The programmer must be able to generate pulses in accordance with requirements. The article describes an universal and modulary built programming board for pulse generation. For different circuits the programming mudules of the 6800 family microcomputer have to be known. As an example the programming of 74 S 288 PROM, which performs the function of IDM 29811 Next-Address Controler, is described.

UVOD

Prednost, ki jo imajo bipolarna vezja pred vezji v MOS tehnologiji, je predvsem v manjših zakasnilnih časih. Za delo v realnem času lahko v nekaterih primerih, kot so n.pr. hitre regulacije, učinkovito uporabimo bipolarne mikroprocesorje. Za uspešno delovanje sistema morajo tudi pomnilniška in ostala logična vezja zadostiti časovnim zahtevam mikroprocesorja, da ne vnašajo prevelikih zakasnitev oziroma, da ne znižujejo frekvenco ure. Bipolarni PROM-i izpolnjujejo vse zahteve, da lahko sodelujejo v takšnem sistemu, saj je njihov zakasnilni čas pod 50 nsec. Uporabljajo se lahko za spominske elemente, za generiranje zapletenih logičnih signalov, za programirano logiko itd. Tako so lahko n.pr. v PROM-ih vpisani algoritmi oz. podatki, ki jih naj ima mikroprocesor na voljo v najkrajšem času. Posebno mesto zavzemajo programirana logična polja - PLA, ki jih odlikuje poleg hitrih odzivnih časov še velika fleksibilnost. Predstavljajo programirano logiko, ki s programiranjem dobi dokončno obliko, ki je pozneje ni mogoče več spremeniti. Programirana logična polja poleg omenjenih lastnosti še močno zmanjšuje število integriranih vezij. S programiranimi logičnimi polji se dajo realizirati različne logične funkcije, pa tudi enostavnejše aritmetične operacije kot so seštevanje ali množenje.

Programiranje bipolarnih vezij je sorazmerno zahtevno, ker so predpisani programski impulzi dokaj različnih oblik. Zaradi nezanesljive dobave in hitrega razvoja se ni dobro nasloniti le na enega proizvajalca in zato je smiselno, da je programator univerzalen. Ker ima vsak proizvajalec svojo tehnologijo, so tudi postopki programiranja, ki jih podajajo proizvajalci, med sabo različni. Zato mora biti programator sposoben generirati več različnih vrst impulzov, ki se naj v odvisnosti od zahtev ustrezno oblikujejo. Zato je najbolj primeren programator s programskim generiranjem impulzov, ki je fleksibilen, glavni del sprememb pa pade na programsko opremo.

Za vse bipolarne elemente te vrste velja isto načelo programiranja, ki temelji na prežiganju (prekinjanju) specialnih povezav v strukturi integriranega vezja. Pri PROM-u predstavlja pomnilniški del matrika posebnih celic, od katerih vsaka vsebuje po eno povezavo (spoj), ki se lahko prežge ali pa ostane nepoškodovana. Stanje spoja vpliva na logično stanje izhoda. Spoj predstavlja tanka metalna plast, ki je po sredini zožena in ki se pri programiranju zaradi povečanega toka segreje in pregori. Pri PLA-ju pa podobni spoji predstavljajo povezave med logičnimi elementi v polju AND oz. OR vrat.

Pri programiranju se neustrezne povezave prekinejo in ostanejo le tiste, ki jih zahteva željena logična funkcija. Ker se pri prežiganju sprošča toplota, se lahko istočasno programira le en spoj, med posameznimi prežigi pa mora biti zagotovljen čas za hlajenje.

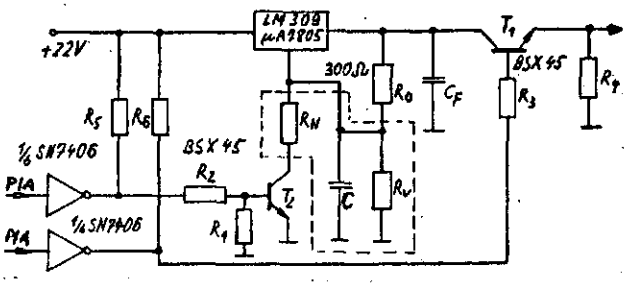
OPIS UNIVERZALNEGA PROGRAMATORJA

Prototip programatorja je v prvi fazi prirejen za programiranje bipolarnih PROM-ov proizvajalcev TEXAS INSTRUMENTS in NATIONAL SEMICONDUCTOR. Med PLA-ji pa se lahko programirajo vezja serije PAL proizvajalca MONOLITHIC MEMORIES in FPLA proizvajalca SIGNETICS. Programator je izveden v modularni izvedbi, tako programska oprema kot vezje. Aparaturni del programatorja predstavlja matična kartica in več manjših pomožnih kartic. Za vsako družino PROM-ov oz. PLA-jev, ki imajo skupne programske lastnosti, je predvidena posebna pomožna kartica z ustreznimi podnožji. S priključitvijo pomožne kartice se avtomatsko vzpostavijo delovni pogoji (napetosti), ki ustrezajo tipu PROM-a oz. PLA. Prav tako je predviden za vsako pomožno kartico svoj podprogram, medtem ko je glavni program skupen. Programator je predviden za priključitev na mikroračunalniški sistem ISKRA DATA 1680 tako, da se matična kartica vstavi v mikroračunalnik, ustrezna pomožna kartica pa se preko konektorja ali podaljška iz ploščatega kabla spoji z matično kartico . Programator se napaja preko zunanjega vira 22 V, 1,5 A in 5 V iz mikroračunalnika za napajanje integriranih vezij.

Kot je razvidno iz sheme na sliki 1, sta za krmiljenje programatorja uporabljeni dve PIA-i. Obe sta priključeni preko bufferjev in naslovnega dekodirnika na mikroračunalnik ISKRA DATA 1680. A - stran PIA-e 1 krmili troje vezij, za oblikovanje napajalne napetosti, oblikovanje CS (chip select) impulzov in oblikovanje OUTPUT impulzov. B - stran PIA-e 1 pa služi za naslavljanje PROM-ov, oziroma krmili vezje za naslavljanje produktnih linij pri PAL. Za PROM-e, ki imajo več kot 8 naslovnih linij, B - stran PIA-e 1 ne zadostuje več in zato lahko za krmiljenje nadaljnih 8 naslovnih linij uporabimo B - stran PIA-e 2, pri čemer ustreza linija PB0 devetemu bitu. Za naslavljanje vhodnih linij pri PAL je uporabljena PIA 2 v celoti.

VEZJE ZA OBLIKOVANJE IMPULZOV

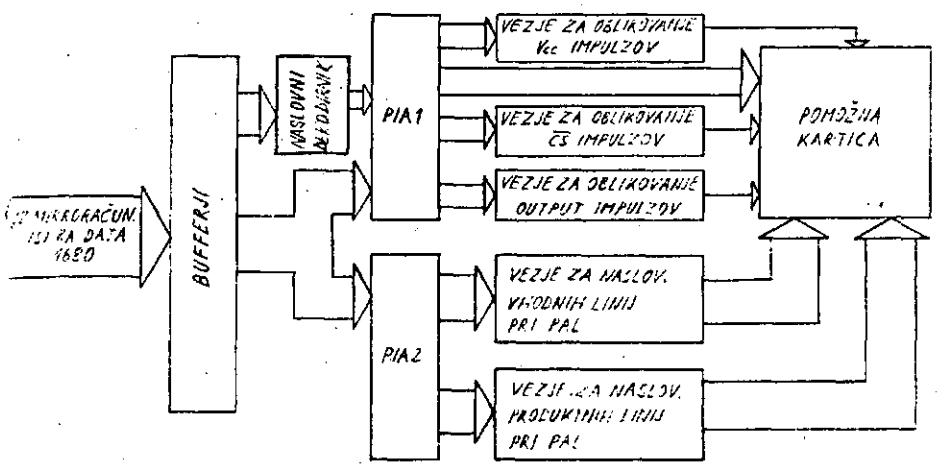
Pri opazovanju impulzov, ki so predpisani za različna programiranja, lahko ugotovimo, da so vsi skoki napetosti med 0 in 20 V. Ker se pri prežiganju pojavljajo tokovi do 750 mA, mora vezje zadostiti tudi tokovni zahtevi. Na sliki 2 je prikazano uporabljeno vezje, ki omogoča skoke napetosti z možnostjo nastavitve časov vzpona, ki so predpisani. Čas vzpona napetosti je odvisen od vrednosti kondenzatorja C.



Slika 2

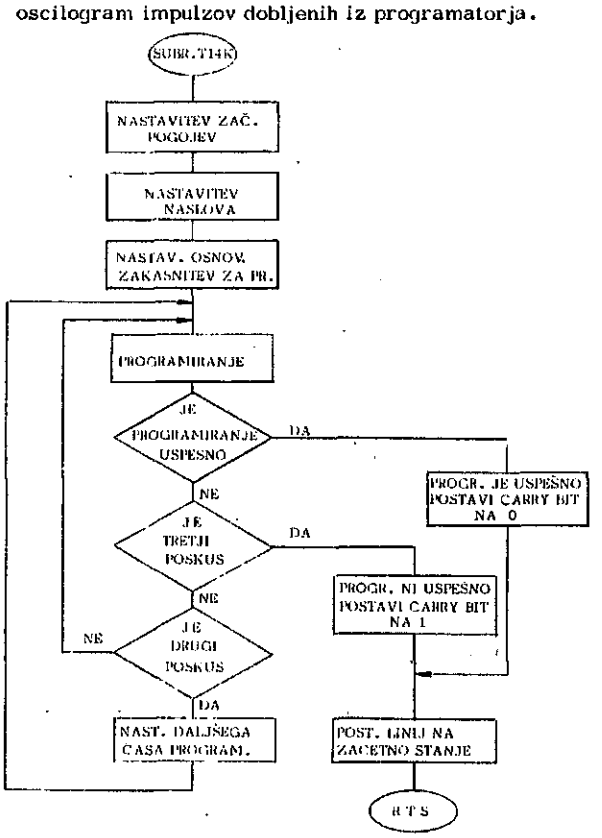
PROGRAMIRANJE PROM-OV (TEXAS INSTRUMENTS)

Programiramo lahko 3 - state PROM-e ali PROM-e z odprtim kolektorjem. Na ustrezni pomožni kartici je predvideno posebno mikrostikalo S_2, s katerim se nastavi režim za programiranje enih ali drugih. Mikrostikalo S_1 je predvideno za nastavitev izhoda (stolpca), ki naj se programira. Potek programiranja je zamišljen tako, da se najprej sklene prvi kontakt stikala S_1, s čemer je omogočeno programiranje vseh bitov, ki so dosegljivi na izhodu 0_1. Nato poženemo glavni program in mikroračunalnik postopoma naslavlja naslove, kjer se naj vpiše logična 0 in sproti programira. Po vsakem programiranju spoj tudi testira, če je uspešno prežgan. Ko so programirani vsi zahtevani biti v prvem stolpcu, se program ustavi, da lahko sklenemo drugi kontakt na mikrostikalu S_1. Seveda prvi kontakt sedaj razklenemo in program znova poženemo. Postopek se ponavlja za vse izhode. Slika 3 prikazuje diagram poteka podprograma. Naslovi spojev, ki se naj prežgejo, so v obliki tabele vpisani v enem izmed RAM-ov mikroračunalnika.

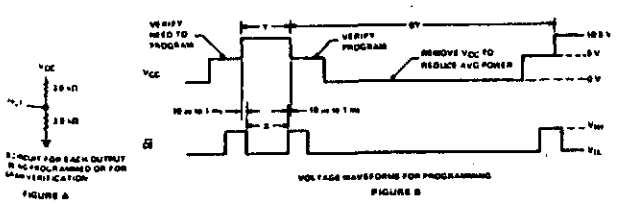


Slika 1

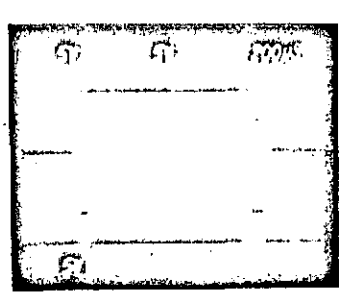
Tabela se s pomočjo programiranja po stolpcih preslika v PROM. Na sliki 4 je časovni diagram predpisanih programskih impulzov, na sliki 5 pa je fotografiran oscilogram impulzov dobljenih iz programatorja.



Slika 3



Slika 4

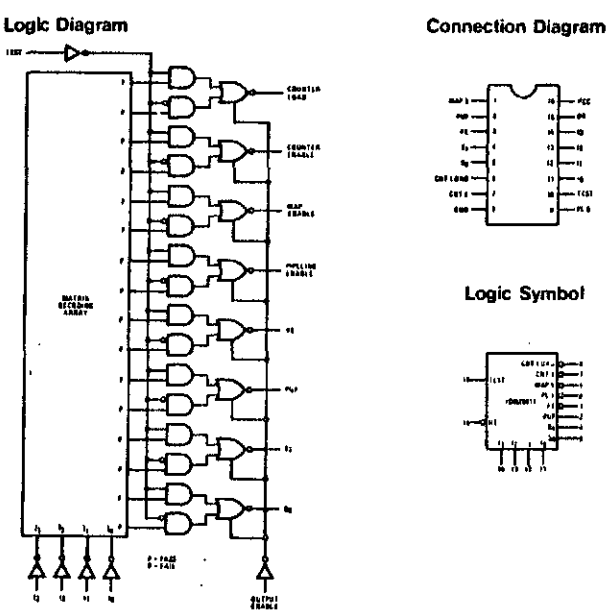


Slika 5

Na sliki 6 je prikazan program za vpisovanje podatkov v PROM 74 S 288. PROM ima 32 8-bitnih besed. Tabela, ki jo vanj preslikamo, je podana na naslovih od 300 - 31F. Na naslovu STL je vpisana številka stolpca (1 - 8), ki se iz tabele preslika v PROM in mora ustrezati številki sklenjenega kontakta na stikalu S_1. Za primer kako lahko PROM uporabimo n.pr. za nadomestitev logičnih vezij, smo vanj vpisali pravilnostno tabelo naslovnega krmilnika IDM 29 811 (glej sliko 7 in tabelo 1). Z ustrezno razmestitvijo stolpcev iz tabele 1 smo celo dosegli, da je razporeditev priključkov na PROM-u enaka ustreznim priključkom na vezju IDM 29 811.

```
                        *PROGRAMIRANJE PROM-A (TEXAS
                        *
                0330    TAB EQU $330
                0331    TA1 EQU $331
                0340    STL EQU $340
                0200    T14K EQU $200
                0400     ORG $400
0400  CE 0300           LDX #$300
0403  FF 0330           STX TAB
0406  FE 0330          ZAC LDX TAB
0409  A6   00           LDAA 0,X
040B  F6 0340           LDAB STL
040E  44               PON LSRA
040F  5A                DECB
0410  26   FC            BNE PON
0412  24   06            BCC NAP
0414  F6 0331           LDAB TA1
0417  BD 0200           JSR T14K
041A  7C 0331          NAP INC TA1
041D  B6 0331           LDAA TA1
0420  81   20           CMPA #$20
0422  26   E2            BNE ZAC
0424  3F                SWI
0425  80                FCB $80
                0000     END
```

Slika 6



Slika 7

Tabela 1

Truth Table

Mnemonic	Function	Fig. No.	14	13	12	11	10	4	5	3	2	6	7	1	0
			Inputs							Outputs					
			I_3	I_2	I_1	I_0	TEST	Next ADDR Source S_1	S_0	File FE	PUP	Counter LOAD	EN	MAPE	PLE
JZ	JUMP ZERO		L	L	L	L	L	H	H	H	H	L	L	H	L
			L	L	L	L	H	H	H	H	H	L	L	H	L
CJS	COND JSB PL		L	L	L	H	L	L	L	H	H	H	H	H	L
			L	L	L	H	H	H	H	L	H	H	H	H	L
JMAP	JUMP MAP		L	L	H	L	L	H	H	H	H	H	H	L	H
			L	L	H	L	H	H	H	H	H	H	H	L	H
CJP	COND JUMP PL		L	L	H	H	L	L	L	H	H	H	H	H	L
			L	L	H	H	H	H	H	H	H	H	H	H	L
PUSH	PUSH/COND LD CNTR		L	H	L	L	L	L	L	L	H	H	H	H	L
			L	H	L	L	H	L	L	L	H	L	H	H	L
JSRP	COND JSB R/PL		L	H	L	H	L	L	H	L	H	H	H	H	L
			L	H	L	H	H	H	H	L	H	H	H	H	L
CJV	COND JUMP VECTOR		L	H	H	L	L	L	L	H	H	H	H	H	H
			L	H	H	L	H	H	H	H	H	H	H	H	H
JRP	COND JUMP R/PL		L	H	H	H	L	L	H	H	H	H	H	H	L
			L	H	H	H	H	H	H	H	H	H	H	H	L
RFCT	REPEAT LOOP, CTR ≠ 0		H	L	L	L	L	H	L	H	L	H	L	H	L
			H	L	L	L	H	L	L	L	L	H	H	H	L
RPCT	REPEAT PL, CTR ≠ 0		H	L	L	H	L	H	H	H	H	H	L	H	L
			H	L	L	H	H	L	L	H	H	H	H	H	L
CRTN	COND RTN		H	L	H	L	L	L	L	H	L	H	H	H	L
			H	L	H	L	H	H	L	L	L	H	H	H	L
CJPP	COND JUMP PL & POP		H	L	H	H	L	L	L	H	L	H	H	H	L
			H	L	H	H	H	H	H	L	L	H	H	H	L
LDCT	LD CNTR & CONTINUE		H	H	L	L	L	L	L	H	H	L	H	H	L
			H	H	L	L	H	L	L	H	H	L	H	H	L
LOOP	TEST END LOOP		H	H	L	H	L	H	L	H	L	H	H	H	L
			H	H	L	H	H	L	L	L	L	H	H	H	L
CONT	CONTINUE		H	H	H	L	L	L	L	H	H	H	H	H	L
			H	H	H	L	H	L	L	H	H	H	H	H	L
JP	JUMP PL		H	H	H	H	L	H	H	H	H	H	H	H	L
			H	H	H	H	H	H	H	H	H	H	H	H	L

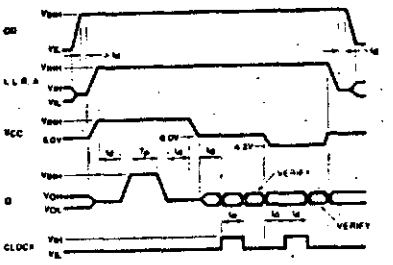
L = Low, H = High

DM2901

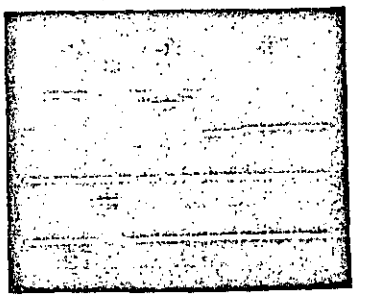
PROGRAMIRANJE PAL-OV (MONOLITHIC MEMORIES)

S tem programatorjem je možno programirati vseh 15 tipov PAL-ov MONOLITHIC MEMORIES. Pomožna kartica vsebuje dve podnožji z 20 priključki, stikalo S_1 in potrebne elemente za predhodno nastavitev pravilnih napetosti. Ker poteka v prvi fazi programiranje spojev na produktnih linijah od 0 do 31, mora biti PAL vstavljen v podnožje 1 na pomožni kartici. Odvisno od tipa PAL je potrebno nastaviti položaj stikala S1. V primeru programiranja takšnega PAL-a, ki daje v aktivnem stanju visok nivo (1) na izhodu, je potrebno postaviti S_1 v položaj ACTIVE HIGH. Obratno velja za PAL-e z nizkim aktivnim nivojem položaj ACTIVE LOW. Programiranje je zamišljeno tako, da so v enem izmed RAM-ov vpisane vse koordinate spojev (številka vhodne linije, številka produktne linije), ki naj se programirajo. Ker so koordinate v RAM-u podane binarno, je v podprogramu narejen programski dekodirnik, ki postavi vse naslovne linije na pravilne nivoje in ugotovi izhod, s pomočjo katerega se bo spoj prežigal. Ko poženemo glavni program, ta prečita koordinate za prvi spoj in skoči v podprogram, ki opravi programiranje in testira prekinitev spoja. Nato glavni program prečita nove koordinate in postopek se ponovi. Postopek se ponavlja za vse spoje na produktnih linijah od 0 do 31, ki so predvideni za prežiganje. Po prežigu zadnjega spoja se program ustavi, da lahko PAL vstavimo v podnožje 2 in s tem preide programiranje v drugo fazo.

V drugi fazi je možno programirati vse spoje na produktnih linijah od 32 do 63. Znova poženemo glavni program, ki postopoma čita nove koordinate spojev in s pomočjo podprograma opravi programiranje. Ko se prežgejo vsi potrebni spoji, se program ustavi in programiranje je končano. Slika 8 prikazuje časovni diagram predpisanih impulzov, na sliki 9 pa je posnet oscilogram impulzov, dobljenih iz programatorja.



Slika 8



Slika 9

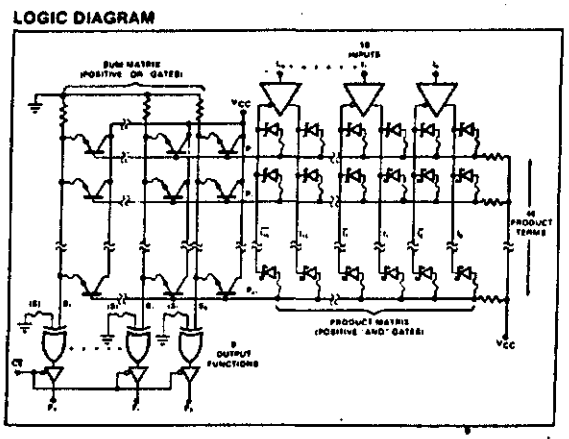
PROGRAMIRANJE FPLA (SIGNETICS)

Programiranje poteka v treh fazah in sicer v naslednjem zaporedju:

1. Programiranje izhodne polaritete (aktiven nivo izhoda)
2. Programiranje produktnih termov
3. Programiranje OR termov.

Za vsako fazo postopka je predvideno svoje podnožje na pomožni kartici, ker vsaka faza zahteva svojo obliko

programskih impulzov. Za vsako fazo je predviden svoj podprogram. Na sliki 10 je prikazana notranja povezava FPLA.



Slika 10

Postopek programiranja izhodne polaritete se prične tako, da se na stikalu S_1 sklene eden izmed kontaktov, ki omogoči dostop programskim impulzom na ustrezen izhod. Nato poženemo program, ki prežge spoj S na ustreznem izhodu. Postopek se ponovi na vseh tistih izhodih, kjer funkcija to zahteva.

Pri programiranju produktnih termov postopamo tako, da najprej sklenemo prvi kontakt stikala S_2, s čemer je omogočeno prežiganje vseh spojev vezanih na vhod I_o. Nato poženemo glavni program, ki prečita ustrezen naslov produktnega terma z informacijo o programiranju spojev. Informacija o programiranju spojev pove, kateri od spojev I_o ali $\overline{I_o}$ se naj prežge oziroma pove, da se naj prežgeta oba spoja, če funkcija zahteva, da vhod I_o nima vpliva na ta produktni term. Glavni program zatem s pomočjo podprograma opravi programiranje enega ali obeh spojev. Nato glavni program prečita naslednji naslov produktnega terma in spet skoči v podprogram, ki opravi ustrezno programiranje. Postopek se ponovi še za vse ostale produktne terme. Ko so prežgani vsi ustrezni spoji v stolpcih I_o in $\overline{I_o}$, se program ustavi, da lahko sklenemo drugi kontakt stikala S_2. S tem je omogočeno prežiganje spojev v stolpcih I_1 in $\overline{I_1}$, ki določajo vpliv vhoda I_1 na posamezni produktni term. Nato znova poženemo glavni program in postopek se ponovi. Postopek je enak za vseh 16 vhodnih spremenljivk.

Pri programiranju OR termov postopamo tako, da najprej sklenemo prvi kontakt stikala S_1, s čemer je omogočeno programiranje vseh spojev na OR termu za izhod F_o. Nato poženemo glavni program, ki postopoma nastavlja naslove ustreznih termov in s pomočjo podprograma prežiga spoje. Po končanem programiranju prvega OR terma se program ustavi, da lahko sklenemo drugi kontakt na stikalu S_1, s čemer je omogočeno programiranje drugega OR terma. Program znova poženemo in postopek se ponovi. Postopek je enak za vseh 8 OR termov.

ZAKLJUČEK

Programator je narejen tako, da se lahko priredi za programiranje še ostalih PROM-ov oz. PLA-jev, ki v tej fazi niso zajeti. V principu je mogoče programiranje kateregakoli bipolarnega elementa, ki ima možnost programiranja. Matična kartica ostane v vseh primerih enaka, pomožne kartice pa se dodatno naredijo. Prav tako je potrebno za vsako novo kartico narediti poseben podprogram, medtem ko je glavni program lahko za večino primerov skupen.

LITERATURA

1. Bipolar LSI Databook, Monolithic Memories, 1979

2. PAL - Programable Array Logic Handbook, Monolithic Memories, 1978

3. TTL Data Book for Design Engineers, TEXAS INSTRUMENTS, 1977

4. Suplement to the TTL Data Book, TEXAS INSTRUMENTS

5. Memory Databook, National Semiconductor, 1977

6. IDM 2900 Family Microprocessor Databook, National Semicondustor, 1978

7. Memory products, MOTOROLA 1979

8. Bipolar and MOS microprocessors, SIGNETICS, 1978

9. Logic TTL, SIGNETICS, 1978

10. Bipolar microprocessor familiy 2900, AMD

11. M 6820 Data Sheet, MOTOROLA

12. Linear integrated circuits, FAIRCHILD, 1976

13. Katalog ISKRA DATA 1680

MICROCOMPUTERS ROLE IN DISTRIBUTED DATA PROCESSING DEVELOPMENT

FJODOR RUŽIĆ

UDK: 681.3

INSTITUTE FOR COMPUTER UTILIZATION, ZAGREB

Microcomputers are revolutionary and are changing users to do things that only a few could make a few years ago. This way in data processing field made micros to be ideal media for the data processing development towards distributed processing implementation. As micros developed and advanced they have many good objectives to develop distributed processing allows several microcomputers to work together to equal the performance of larger computer systems.
The micros technology development makes the distributed processing systems going to implementation of infosystems for public data communications which are much more powerful systems since they can access a very much larger store of information and provide true interactive facilities. Information systems for public data communications open the way to the universal computer service in any location that can access the public communication facilities.

ULOGA MIKROKOMPJUTORA U RAZVOJU DISTRIBUIRANE OBRADE INFORMACIJA
Mikrokompjutori su unjeli revolucionarne promjene na području obrade informacija, omogućujući korisnicima pristup onim obradama koje su prije nekoliko godina mogli izvoditi samo njih nekolicina. Tako, mikrokompjutori postaju idealan medij za razvoj organizacije obrade informacija usmjeren ka uvodjenju sistema distribuirane obrade. Razvoj mikrokompjutora donosi mogučnost medjusobnog povezivanja više mikrokompjutora koji, povezani tako u mrežu, mogu izvršavati obrade kao i veliki kompjutorski sistemi.
Istovremeno mikrokompjutorski sistemi omogućuju razvoj sistema distribuirane obrade i uvodjenje sistema za javno komuniciranje podacima i informacijama, koji su znatno razvijeniji budući omogućuju rad sa velikim bankama podataka i interaktivan rad. Ovi sistemi otvaraju put ka kategoriji sveobuhvatne informatičke usluge na svim onim mjestima gdje je moguć pristup javnim komunikacionim sredstvima.

1. Introduction

During periods of rapid change, timely accurate information became essential to organizational success. The rate of change we will face during the 1980's is unparalleled. To master this change and improve their productivity, organizations will need to dramatically increase their use of information systems technology.
This need arises at a time when information systems are becoming cheaper and more flexible. Advances in microelectronics and telecommunications technology are driving hardware costs downward. Distributed data processing and telecommunications networks are proliferating and organizations can now tailor the flow of information to meet their specific needs.
The computer as general purpose tool is applied by all of the organizational activities as fast as change can be managed. The technology and machine capabilities far out strip user ability to utilize them in general sense. As microcomputers become more easily available and usable the applications will grow. The micros are getting into all sorts of products but their rate of change and implications of their productions are not widely appreciated.
The microcomputer has become upgradable into

an inquiry station, a word processing station, an intelligent terminal-in general, they are indicative of the end users machine of the future.
As microtechnology advanced, the micro was able to accomplish tasks only a minicomputer could do before. In fact, some micros have advanced so much that they are called minis. The term `microcomputer`is closely allied to the term microprocessor. The devices are similar although there is a technical difference between them. The term`micro` is used very often from the reason of simplicity. This abbreviation is common today and makes sense because most of the time the users will be interested in the fact that a product contains a micro.
The current computer technology moves toward distributed processing (DDP) and DDP network so the DDP is the latest phase of networking evolution that began with the first interconnection of the two computer systems and communication implemented at the user software level. As soon as the microcomponents become available the transition from central systems to distributed networks begins. As the issues of decentralized controls via micros is becoming available total packages are available to users who see advantages in moving functions away from the central system to the distributed systems` members.
As micros developed and advanced they have many good objectives to develop distibuted processing allows several microcomputers to work together to equal performance of larger computer systems.
Distributed processing spreads the computing work over several smaller computers rather than concentrating all the work in one large computer. For the user in DDP system microcomputers mean that he can economically have access to the same kind of operating and accounting information previously only to his large competitors.

2. Data Processing and Microtechnology Development

When computers first became commercially available large companies saw the benefits of using them in their organizations. Today microcomputers are affordable, so small business educators and scientists think they can foresee those some kinds of benefits on a smaller scale. As large organizations changed, computer systems could not easily adapt to meet new requirements. So today`small business are finding it difficult to modify previously accepted great software and hardware packages because software built to meet immediate needs may not be usable in the future developments due to change.
At 1975 the microcomputer appeared on the scene. This was a complete computer-on-a-chip of silicon. This chip was mounted on a board which plugged into the computer. Now the new minicomputers (as well as the large mainframes) in the distributed systems actually use microprocessor chips as components, so that that the difference between minis and micros has disappeared.
And what to say about microcomputer? A microcomputer is a very small but versatile, law cost and powerfull general purpose machine that can be programmed on a language designed by humans to perform specific tasks. A microcomputer is just another computer, except that it is smaller and less expensive than a mini or maxi. A microprocessor is basically the central processing unit of computer shrunk. The new microprocessors were developed using technology called Large Scale Integration (LSI and VLSI) in which thousands of elements are connected on a substrate of silicon. Microprocessor is generally designed for a specific purpose. The size of the program is limited by the amount of memory supplied and in many cases is not easily changed. When one takes a microprocessor and adds certain electronic parts, such as memory and input/output circuit a microcomputer is the result. With additional memory the microcomputer can handle larger, more complex tasks. Input/output circuits permit the microcomputer to work with other equipment such as keyboard, printers, displays, etc. Although the microcomputer offers little competition to larger computers on computing power and speed, designers everywhere have capitalized on the micro`s strength of versatility, small size and low cost to produce variety of products. In many cases these products wouldn`t exist today without the micro, because the design using other methods would be too large or too expensive.

At current rate of microtechnology development the microutilization may be grouped into one of three categories:
- a micro is inside the piece of equipment that controls some procedures
- a single micro controls an entire system
- several micros are interconnected to the larger computer - the technique very close

to the term of distributed processing.

One of the main objectives that end users in DDP systems demand considering micros utilization is the amount of memory to be used. In this term the memory is always spoken of in terms of read only memory type, for storing and developing dedicating programs. With the new memory development the users will be able to decide both the amount and the type of own memory. The recent computer technology based on semiconductor memories will take the central role in future data processing - in distributed processing systems and infosystems for public data communications.

The newcomer in memory technology are magnetic bubble devices and charged-coupled devices, too. The potential market for magnetic bubble memories lies in the replacement of tape and disk memories with a capacity of between one and ten million bits. Magnetic bubble memories are natural for applications where permanence of storage and portability are desired so the devices in DDP system environments will ask this new memory technology in very strong and urgent way. The charge-coupled memories are both smaller and less expensive than random access semiconductor-based memories but the great disadvantage of such devices is that the information stored is volatile. However there are many DDP applications where serial access for stored information is entirely satisfactory.

Laser, hologram and high-resolution microfilm also hold promise for the memories application in the future DDP systems. It will be years before these memories become available to the users but when they do they will give the memory revolution such as the magnetic bubbles and charge-coupled memories are doing now.

For the mass storage technology that will be the central point of each infosystem of public data communication archival memory would be of the main interest. Archival memory is high capacity secondary memory that can store up to trillion bits with access time of about three to five seconds.

The characteristic of the micro-computer based systems used in DDP vary. One can find a wide variety of memory capacity, memory types and bit size or word length. Word length are the factor in determining how much processing a user can get done in a day. Many of the microcomputer systems available now have a 16-bit central processor meaning that the user is able to address 8 K of memory directly. A smaller word length, such as 8 bits, requires indirect addressing or more than one step and thus slower processing. As the word length/ size increases the user can do a lot more in one command.

Most microcomputer systems are based on MOS (metal-oxid semiconductor) technology. Memory capacities ranged from 4K all the way up to the 256 K. Many are offered with floppy disk for random access storage.

Software support has not been as well developed for microcomputer systems as it has been for mini and maxi computer systems. The smaller memory size currently available on micros has required high efficiency from the programming language. Therefore, many programs have had to be written in machine language instead of high level language.

Writing programs directly into machine language is tedious and time consuming. To solve this problem such programs are written in high level languages on larger computers, usually on minis and a device called cross-assembler is used to produce a machine language that can be run on the micro.

As memory sizes for micros expand it is anticipated that high level languages will become available. Today only basic language has been widely used, mainly because of its simplicity. But the languages such as FORTRAN, COBOL and the excellent PASCAL are slowly begun to be run and used on the microcomputers. These developments will make programs for the microcomputer systems easier to write and problem solving and should bring wider use in DDP systems.

3. Micros in DDP Systems

The rate of the technology change in the next years will be new storage techniques, microfiles, very large scale-integration, new types of networks and distributed systems and the wide spread use of microcomputers which provide terminals with processing power and which provide new minicomputer architecture will radically change traditional data processing.

The current explosion of the micro and minicomputer technologies into the distributed intelligence presents a frequently bewildering array of possibilities to the computer user. The micros and minis seem to be approaching the capabilities of the medium and large scale mainframes while the large mainframes are shrinking to the physical size of the micros. The resulting distributed intelligence appears in terminals, modems, multiplexors,

concentrators and other network oriented devices in addition to the host computers. Adding to the complexities of the environment is the fact that it is possible to acquire all of the source. The new multisource hardware is a very real element of any DDP network planning and implementation. The only thing that is constant in the network environment is the change. Change is an integral part of any successful cooperation including the use of computers either as a management tool or a method. Change and the ability to adapt it must be an integral part of any network activity centralized and/or distributed. Most network design activities began with a very specific detailed definition of a very specific set of applications to be accomodated. A workable approach to the definition of a logically structured frame of references involves the definition and identification of the basic function sets required for the configuration of DDP network.
The three basic functions in the DDP systems are:
- distributed data processing (a)
- distributed network processing (b)
- distributed data base processing (c)

a) - the manipulation of information to produce the desired result
b) - information transmission among network nodes
c) - the storage of information in wide variety of forms appropriate for access by the DDP network and its users.

In the DDP environment all of these functions are distributed at more locations in the network.
The activities that are carried out on DDP systems very often you will see below:
- joint works on document sharing and exchange of files, papers and other documentation
- distribution of programs
- using program libraries of shared programs
- access to data bases
- natural languge processing and text editing

A large number of important applications of computing require that a geographically diverse group of users have access to a single program or data base through the network.

For the end users in DDP systems mean that he can ecconomically have access to the same kind of operating and accounting information previously available only to his larger competitors. Within large companies that take the DDP systems design the micro has opened up new capabilities for department heads and for managers at their individual locations.

There are no programers and no operators to go through to obtain computer stored information. Managers themselves can get the information they need and whenever they need it.

The ultimate user in DDP system uses information applied by a microcomputer to perform his job. So, the Data Processing Departments and large using groups must interface with the new body of users. The end users have a range of options which could satisfy information processing needs. Among them there are some specific as follow:
- remote computing services supplied by data processing department
- shared small business or minicomputer
- personal (home) computer systems
- programmable terminal (intelligent)

In the distributed processing systems the end user benefits are:
- understand the demand for local processing of information which can be met through microcomputers local, small computers
- learn the optimum approaches to providing microcomputers to the end users demand
- understand the support, communication, software, training and installation requirements of microcomputers.

DDP networks represent a rapidly expanding alternative to the conventional centralized network. The introduction of large scale integrated circuits has brought about lower cost terminals and computers. A trend towards making data communications more compatible with data processing began a few years ago and is continuing. The communication supplies have data equipment which can perform data processing functions as well as data communications. The apparent result is that all aspect of information system are becoming more highly integrated in the DDP network. The micros technology development has reduced the costs associated with computing. The typical computer user no longer confines the data processing functions to the computer-specialized site with the host computer. Today, with the mini and microcomputers utilization many users in DDP system input and process data at sites remote from the central computer. The results of data processing at the central location are distributed to distant terminal sites. DDP permits the performance optimization of network components by enabling tasks to be asigned

to those elements in the system which are best suited to perform them. A complex distributed data network consists of a large number of communication links connecting several network nodes. The new communication equipment with the built in processor based on microelectronics components exists today in general DDP network elements:

- communication processor - a today's microprocessor-based computer combined with integrated communication devices
- front/end processor - small stand-alone, also microprocessor-based computer which is connected to the central computer and performs the major communication task for the central computer allowing it to be dedicated primarily to application functions
- concentrator - microprocessor-based system located in DDP network so that it can control a number of terminal links and concentrate the data traffic.

Micros on the market today using technological advances provide the benefits of the multiplexing in the network operations. There are single processor microcomputers in use, operating with semiconductor RAM for buffer storage and ROM to contain control program firmware, as Data Concentrator in DDP networking. The arrival of the microcomputer based concentrator represents a major step in closing the gap between the minicomputer and the large computers in DDP systems. The microelectronics and microcomputers are coming to the aid of the minicomputer as well.

4. Micros and Terminal Devices

Terminals are devices through which users can communicate with computers. They contain typewriter-like keyboards and displays. Some may also contain printers which provide a hard copy of the data displayed on the screen.

Over the years terminals have been getting increasingly more intelligent. There are two primary reasons for this. The first cause relates to the development of plug-compatible terminal devices. The second reason for increasing intelligence on terminals is the use of microprocessors (or programmable microelectronics).

Before videoterminals became available the main user-computer dialog was via paper-tape, punched cards, teleprinters. Modern video terminals are silent, quick and easy to understand as the computer input/output is given on letters and numbers. Early terminals were very dumb and had no way of manipulating data to the computer or interpreting the computer's special commands. With the current development of LSI and microcomputers the electronics inside the terminals became more sophisticated. As a result we have intelligent terminals that can manipulate the data and generally make work easier for the computer and human. More and more terminals are being produced with microprocessors built into them, with a resultant flexibility that was unheard of a few years ago. Intelligent terminals contain microprocessors and in many cases microcomputers. When connected these terminals can form a distributed network. Bubble memories and the additional power they can give intelligent terminals are a possible first step towards distributed systems in many working areas. These terminals are able to do their own processing and pass results to a central processors in distributed processing.

5. Micros and DDP toward Infosystems for Public Data Communications

The microprocessor technology development is the important milestone in the informatics field. It comes the period of microcomputer based systems and with distributed data processing systems development we can realize the new information systems design as necessary to meet the end users demands in the full way.

Distributed data processing is end-user oriented and hence it must be connected data processing and data communications to create flexible, modular designed distributed processing systems. With the marriage of data processing and data communications started the new phase in data processing decentralization which first sophisticated phase was bounded with the distributed systems implementation.

The new phase in the data processing decentralization based on the current microprocessors development is the phase of the infosystems for public data communication (INCO).

In the INCO systems every user can communicate with the computer which is storing the information and such information network is intended almost entirely for information communications with the minimum of computation involved. At this point of view the basic principles

that must be incorporated into INCO system are:
- simplicity
- reliability
- mass accessivity

The INCO system is a development of distributed processing systems but is much more powerful since it can access a very much larger data bases and store of the information and provides true interactive facilities.

INCO systems will provide computer based information service in almost every home. It offers frequently updated information to a mass lay audience on textual and visual form to extend the prime use of the stand-all one home microcomputers. It is based on existing technology - the telephone system as data transmitter, a modest adaptation of the home TV set to display selected data and graphics and the host computer with multiuser access, large data storage and rapid retrieval.

It provides for mass dissemination but the information is not under the central control - just alike the telephone service and rather than TV or newspapers. Further, the INCO system is cheap, easy and inexpensive to put up information as an information supplier - and the category - information power is extended with the potentially. So the INCO systems will have a great role in the successive revolutions of literacy and printing. Finally, it will have an effect on existing forms of information provision-book, newspapers, magazines, etc. Few existing communications media will disappear but their role will be changed.

Some current developments on data base management system field provide INCO systems utilization to the wide variety of end users. So each system user would have to access not one computer system where everything can be fraud but several. INCO systems use the telephone network to gain access to the computerized data bank. An alternative approach is to transmit the information over the air with the normal television broadcast signals. As mentioned the key element in the INCO systems is the intelligent visual terminal and there are three main types of communication applications for such terminals:
- terminal to database
- terminal to computer
- terminal to terminal

One database might contain a large amount of information enetered by a number of individual information suppliers. However, a large number of separate data bases owned and operated by each information supplier is quite likely. The technilogical advances in stand-alone products with ever-increasing intelligence is the new help to INCO systems improvement. Interactive work requires a variety of input devices that would make a single, universal terminal, difficult to devise. In such work users require a very fast response and this is difficult to accomplish when a number of remote users are all accessing the same central computer. The solution to these problems is to use intelligent terminal with its own microprocessor computer capability and to transmit programs to this terminal from a central program data base. However, the prospects for storing the end user's programs on cassette and single solid state chips appear to eliminate the need for transmitting programs over a communication links to the terminals.

The local micro can take desirable data input from the great information banks available to INCO system. The local microcomputer can also give the local user a flexibility of handling INCO system, both as to speed of access to the required information and the cost of obtaining which is available to the ordinary user who has a single, simple terminal device. The INCO system's aim is to be user friendly. To persuade the user a data base is very easy to use. INCO system should have to structure its own data bases in a variety of ingenious ways. It has to ensure that any user is anticipated so that he can be routed in all manner of directions. Although some people may describe a typical INCO system data base having a tree structure in fact such is the amount of looping back and forth. These factors suggest the need for intelligent terminal. The introduction for the INCO systems and dedicated intelligent terminal makes it possible for organizations to store and exchange electronic information inside their and within other organizations. The introduction of enclosed INCO systems would do more than introduce a new machine-tool into the office. The first thing it does is to enable information to be captured at source and hence to remove the necessity for transcription to bring it into the office data base. Thus computers are brought nearer to the mass population and so the way is made for further, wider computerisation.

The linking of microcomputers with INCO systems could be the first step towards producing the cheapest small computers yet. At the same time it makes possible a new form of software distribution. The recent developments of micro-

computer-based system cover VDU screen, processor, memory and cassette or diskette, backing store with large scale integration processor. INCO systems micro-based inteligent terminals have the potential for changing the cost equations radically.
The new micro technology which turns a home TV set into an INCO system terminal is now microprocessor controlled. That gives a greater flexibility than the TTL logic technology used in earlier electronic equipments and gives the possibility, among others, of integrating a complete microcomputer systems with INCO system terminal, to produce a new category in the informatics field - the intelligent terminals programmable as a computer system.
The next potential change to these small computers is related to the use of software in the way in which it is distributed.
The process of transmitting software via communications medium from a different computer is known as telesoftware. One of its effects is that micro-based end user terminal in INCO system has no need for its own software medium diskette or cassette.

6. Conclusions

As microtechnology advanced the micro was able to accomplish tasks a microcomputer could do before. In fact some micros have advanced so much that they are called mini. So the micros and minis are approaching the capabulities of the medium and large scale mainframes while the large mainframes are taking physical size of the minis.
The micros and distributed system are making new contributions to competitions by both increasing internal efficiency and improving custormer service. Business people have really just begun to discover wide variety of uses for microcomputers. Unlike the larger business computers which require a specially designed room environment and a data processing staff, a microcompuetr is designed to be used at an individual level. It is relatively inexpensive and doesn't need a data processing professional to operate it. But most important is that micros are easy to use. Nearly any person can purchase it, install and operate his own. Thus in distributed processing systems the computer devices can do all the functions processor-based devices do and more. They are intelligent devices and this means the cost of programming them is higher than that of a processor-based devices, but the advantages are larger. They can run programs in stand-alone mode also.

Since the control of distributed systems processors is local turn round and processing priorities are in the hands of local management. The response time between data input and information output is often faster than when data is stored at a central site. The introduction of the infosystem for public data communications as the distributed system development makes it possible for organizations to store and exchange electronic information inside themselves and with other organizations. Where INCO systems come personal computing through the use of microcomputers will not flourish.
But at the same time the use of microcomputers in business, government and education will grow strongly. The users will benifit yet from the sociological impact of having computing power available at home. As people become more accustomed to interacting with data display devices they will lose their gear of the technology and look for more services outside their offices, institutions and homes.

References

1. R.H.Eckhouse, J.A.Stankovic, A.Van Dam: "Issues in Distributed Processing", Computer, January 1978.
2. Distributed Processing-Proceedings, National Computer Conference, Anaheim, California, June 1978.
3. C.Weitzman:"Distributed Micro/Minicomputer Systems", Prantice Hall, 198o.
4. R.J.Thierauf: "Distributed Processing Systems", Prantice Hall, 1978.
5. Distributed Processing-Current Practice and Future Developments, Q.E.D., 1978.
6. F.Ruzic:"Distributed Data Processing Systems", Informatica, Bled, 1978.
7. F.Ruzic: "Infosystems for Public Data Communications", ACM Computer Science Conference, Kansas, 198o.

UNIVERZITETNI POUK RAČUNALNIŠTVA I

ANTON P.ŽELEZNIKAR

UDK: 378 : 681.3

SOZD ELEKTROTEHNA, DO DELTA, LJUBLJANA

Kvaliteta univerzitetnega pouka računalništva (in informatike) postaja bistveni element nadaljnega razvoja proizvodnje in uporabe računalnikov pri nas. Članek prinaša v dveh delih celotni načrt pouka za razvijajoče dežele, kot je bil predložen v okviru posebnega odbora IFIP v povezavi z UNESCO. V drugem članka bo opisan polemičen prispevek glede na IFIPov predlog in glede na stanje izobraževalnega postopka na slovenskih univerzah. Primerjava načrta in stanja kaže obojestranski deficit in hkrati opozarja na konkretno neodgovornost (nezainteresiranost, nesposobnost) posameznikov in ustanov pri oblikovanju in spremljanju sodobnih računalniškoizobraževalnih procesov doma in po svetu.

University Curriculum in Computing I. The quality of a university curriculum in computing is one of the essential factors influencing further development of computer production and usage. This article deals (in two parts) with the entire proposal of curriculum for developing countries being elaborated within the Technical Committee for Education (TC 3) of IFIP under a contract of UNESCO. In the second part of the article the differences are pointed out when the IFIP proposal is compared with the status of curricula in computing on universities in Slovenia. This comparision shows a twodirectional deficit and calls for corrections and inovation of the present educational efforts on domestic universities.

1. Uvod

Kakšno naj bi bilo izobraževanje iz računalništva (in informatike) na univerzah razvijajočih dežel, je vprašanje, ki ne zanima samo prizadetih, temveč se z njim ukvarjajo predvsem v razvitih državah. V marcu 1980 je bil izdelan dokument (1) (UNESCO-IFIP), ki opisuje modularni tečaj iz računalništva za dežele v razvoju. Ta dokument je nastajal v okviru posebne delovne skupine tehničnega odbora za vzgojo (TC 3 / IFIP) sporazumno z UNESCO. Delovno skupino so sestavili iz učiteljev iz ZDA, ZR Nemčije, Velike Britanije, Francije in UNESCO (Atchinson, Brauer, Buckingham, Hebenstreit in Paker). Dokument (1) predpostavlja, da se v razvijajočih deželah vsaj zaenkrat ne bo razvila domača računalniška industrija, zato v njem ne najdemo modulov, ki se neposredno nanašajo na računalniško proizvodnjo. Dokument (1) ostaja tako slej ko prej zanimivo dopolnilo za naše razmere.

2. Moduli izobraževalnega paketa

Dokument (1) opisuje 34 modulov, ki jih je moč povezati v drevo (glej sliko 1). To drevo predstavlja 4 prekrivajoče smeri, katerih vsaka naj bi trajala več kot 3 leta. Glede na sliko 1 (številke modulov, npr. 5.3, v desnem spodnjem kotu) so te smeri naslednje:

1. Informatika za upravljavce in administratorje (moduli 1.1, 2.1, 2.2, 2.4, 3.1, 3.2, 4.1, 4.3 in 5.1).

2. Iformatika za inženirje drugih področij (moduli 1.1, 2.2, 2.3, 2.4, 2.7, 3.2, 3.3, 3.5, 3.8 in 4.3).

3. Šolanje sistemskih inženirjev (vsi moduli z izuemo modulov 2.7, 3.8, 4.6, 5.7 in 6.3).

4. Šolanje programirnih inženirjev, tj. poklicnih programerjev (moduli 1.1, 1.2, 2.2, 2.3, 2.4, 2.6, 3.1, 3.2, 3.4, 3.5, 4.1, 4.3, 4.4, 4.5, 5.3, 5.5, 5.6 in 6.1).

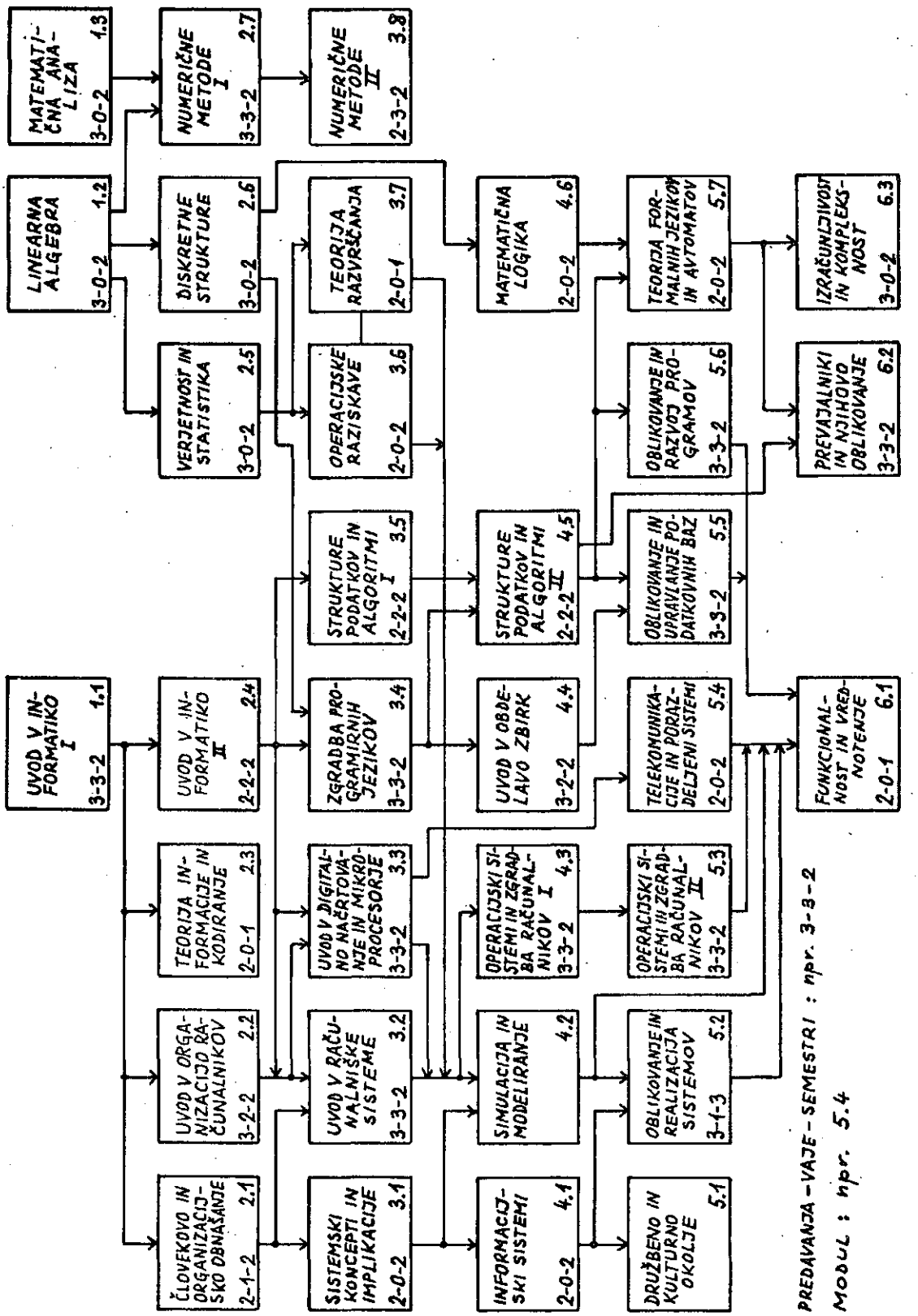
K tem smerem je moč dodati še smer za tiste študente, ki kažejo razvit smisel za abstraktne vidike informatike in za raziskovanje (moduli 1.2, 2.6, 4.6, 5.7 in 6.3).

Tri številke v zaporedju A-B-C pomenijo tedenske ure predavanj (A), vaj (B) in število semestrov (C).

3. Vsebina nekaterih modulov

Opis vsebine vseh modulov s slike 1 bi zahteval več prostora, zato si oglejmo vsebinsko le nekatere. Vsebine posameznih modulov so bile oblikovane na osnovi obstoječih uradnih dokumentov ZDA, Velike Britanije, Francije in Zapadne Nemčije ter odražajo tako tudi splošno stanje v razvitih državah. Pri tem velja še posebej omeniti, da so v predlogu izpuščeni tisti moduli, ki se nanašajo na informatično tehnologijo, to je na računalniško industrijo. Vrsta tehnološko pomembnih modulov se namreč poučuje v okviru elektronske tehnologije in takšen študij zahteva drago dodatno laboratorijsko opremo za izpeljavo vaj in diplomskih del.

V danem predlogu UNESCO bomo morali pri oblikovanju domačih smeri upoštevati pomanjkanje tehnoloških modulov v njem oziroma pogleda-



Slika 1. Predlog študijskega programa računalništva (in informatike) za univerze dežel v razvoju kot ga je predložila posebna delovna skupina UNESCO.IFIP

ti, kje se ustrezni predmeti že poučujejo in kako bi jih bilo mogoče vključiti v ustrezne tehnološke smeri.

Oglejmo si vsebino nekaterih modulov, ki so za nas glede na trenutno stanje še posebej zanimivi.

MODUL 1.1

Uvod v informatiko I (3-3-2)

Cilji

Po končanem modulu naj bi študent pridobil

a) metodologijo razvoja algoritmov in reševanja problemov

b) znanje iz uporabe programirnih jezikov in

c) sposobnost kodiranja, iskanja napak in dokumetacije kratkih programov ter stiliziranega pisanja (sestavljanja) programov

Deli modula

Snov o programiranju v visokih jezikih in o razvoju algoritmov naj bi bila obravnavana kot celota. Poudarki v učni snovi pa so tile:

a) splošna zgradba računalnikov

b) razvoj algoritmov

c) programiranje v določenem stilu

Vsebina modula

1. O r g a n i z a c i j a r a č u n a l n i k o v :

vhodne/izhodne enote
pomnilniki (dostop, lokacija, naslov, vsebina, pisanje, čitanje)
aritmetične in logočne enote
krmilne enota (strojni ukaz, ukazni števnik)
strojni cikel (jemanje in izvajanje ukazov)
notranja predstavitev podatkov)binarni sistem, biti in zlogi, oktalni in heksadecimalni sistem)

--- 20 % ---

2. P r o g r a m i r a n j e : predstavitev celih in realnih števil, karakterjev, ukazov
tipi podatkov, konstante, spremenljivke
aritmetični izrazi
prireditveni stavki
logični izrazi
sekvenciranje, izmenjava in iteracija
polja
podprogrami in parametri
enostavni V/I
tehnika odkrivanja napak

--- 40 % ---

3. R a z v o j a l g o r i t m o v :

metode reševanja problemov
uporaba diagramov poteka
postopno določevanje
enostavni numerični primeri
algoritmi iskanja (linearni, binarni), sortiranja (izmenjava, vstavljanje), povezovanja urejenih seznamov
primeri iz poslovne uporabe (manipulacija podatkov), simulacije in iger

--- 40 % ---

Vaje

Študenti dobivajo krajše probleme za domačo nalogo. V laboratorijo potekajo razprave po skupinah, ko se analizirajo algoritmi in programi študentov s poudarkom na pravilnost in obliko (stil).

MODUL 2.2

Uvod v organizacijo računalnikov (3-2-2)

Cilji

a) priučitev osnov logičnega načrtovanja

b) razumevanje mehanike prenosa informacij in krmiljenja v digitalnem računalniku

c) pojasnevanje organizacije in zgradbe materialnih komponent računalnika

Poudarek v modulu

Trije glavni deli modula (logika, zgradba in arhitektura) naj se prepletajo in naj se ne poučujejo ločeno zaporedoma. Logično načrtovanje naj se poučuje funkcionalno do stopnje, ki je potrebna za razumevanje zgradbe bistvenih delov računalnika. Tehnološke podrobnosti naj se izpuščajo.

Vsebina modula

1. O s n o v e l o g i č n e g a n a č r t o v a n j a :

predstavitev podatkovne in krmilne informacije z binarnimi signali
logične lastnosti vrat in bistabilnih vezij
tabele pravilnosti, Boolove funkcije in časovni diagrami
kombinatorna vezja z uporabo AND, OR, NOT, NAND in NOR vrat
paralelni in serijski registri
sinhroni krmilni mehanizmi

--- 20 % ---

2. P r e d s t a v i t e v p o d a t k o v :

nekateri kodi (EBCDIC, ASCII, Baudot)
generiranje in detekcija parnosti
kodirniki, dekodirniki, kodni pretvorniki
številski sistemi (celo število, plavajoča vejica, negativna števila)

--- 10 % ---

3. U k a z n i f o r m a t i :

operacijski kod in modifikacijski biti
ukazi za navajanje pomnilnika
ukazi za navajanje registrov
nepomnilniški ukazi
tipi naslavljanja (takojšnje, indirektno, razširjeno, indeksno, relativno)
izvrševanje (izvajanje) ukazov

--- 20 % ---

4. A r h i t e k t u r a r a č u n a l n i k a :

funkcije in komunikacija med komponentami računalniškega sistema
materialna realizacija jemanja ukazov, zgradbe naslova in izvrševanja ukaza
pretok podatkov in diagram krmiljenja enostavnega procesorja

zamisel mikroprogramiranja in podobnost s programsko opremo
lastnosti enostavnih V/I naprav ter njihovi krmilniki, sinhrono krmiljenje in prekinjevanje
načini komuniciranja s procesorji

--- 30 % ---

5. P r i m e r i :

študij dejanskega, enostavnega mikro ali miniračunalnika

--- 20 % ---

Vaje

Vaje se opravljajo na mini ali mikroračunalnikih.

MODUL 2.4

Uvod v informatiko II (2-2-2)

Cilji

a) razvoj discipline oblikovanja programov stilno in izrazno, odkrivanja napak in preizkušanja večjih programov

b) uvajanje v algoritmično analizo

c) uvajanje osnovnih vidikov obdelave nizov, rekurzije, notranjih iskalnih/sortirnih metod in enostavnih struktur podatkov

Poudarek modula

Snov tega modula naj se uvaja v povezavi z enim ali več projekti velikih programov. Učitelj naj začne z opredeljevanjem obsežnejšega projekta, z uporabo metodologije strukturiranega programiranja in nadalje z razvojem vrste malih projektov, ki zajemajo obdelavo nizov, rekurzijo, iskanje in sortiranje in/ali podatkovne strukture. Poudarek naj bo na dobrem programirnem stilu, izražavi in dokumentaciji. Včasih je potrebno vpeljati tudi še dodaten programirni jezik (npr. Fortran), da se pokažejo razlike. Podrobnosti programirnih jezikov naj bodo v tem primeru vključene v modul. Uvede naj se analiza algoritmov, ki jo učitelj sproti prenaša na študente.

Prikazana naj bodo izhodišča za izdelavo programirnih projektov, ko se študentje organizirajo v programirne skupine. Ta pristop je bistven zlasti v tečajih na višji ravnini in naj se ga uvede čimbolj zgodaj. Tudi pri velikem številu študentov naj bo delo vsakega študenta podvrženo kritiki drugih študentov.

Vsebina modula

1. P r e g l e d :

osnove dobrega programirnega stila, izražave in dokumentacije
podrobnosti dodatnega programirnega jezika
--- 15 % ---

2. O s n o v e s t r u k t u r i r a n e g a p r o g r a m i r a n j a

krmilni potek (pretok)
invariantna relacija zanke
postopna izboljšava stavkov in podatkovnih struktur
navzdolnje programiranje --- 40 % ---

3. P r e i z k u š a n j e i n o d k r i v a n j e n a p a k

--- 10 % ---

4. P r o c e s i r a n j e n i z o v

stik
podnizi
prilegojevanje
--- 5 % ---

5. N o t r a n j e i s k a n j e i n s o r t i r a n j e :

metode: binarna, z bazo, Shelova, hitro sortiranje, združevalno sortiranje
kodiranje z izračunanim vstopom
--- 10 % ---

6. S t r u k t u r e p o d a t k o v

linearno dodeljevanje (skladi, vrste) in povezano dodeljevanje (enostavno povezane liste)
--- 10 % ---

7. R e k u r z i j a

--- 10 % ---

MODUL 2.6

Diskretne strukture (3-0-2)

Cilji

a) pokaže se poenoten teoretičen pristop k vrsti zamisli, ki se pojavljajo v naslednjih modulih

b) študent naj se usposobi za spremljanje sodobne znanstvene in strokovne literature

Poudarek modula

Uvaja se teoretična snov v matematični in natančni obliki. Zamisli in rezultati naj bodo podprti s prikazi primerov iz računalniških znanosti.

Snov je lahko razdeljena na dva dela: moderno algebro in njeno uporabo pri sekvenčnih strojih in računalniških sistemih ter na grafe in drevesa, uporabljene na podatkovnih strukturah in algoritmih. Študent naj opravlja izdatne domače naloge, tako teoretične kot programirne, ki pospešujejo razumevanje uporabe danih zamisli v okviru računalniških znanosti.

Vsebina modula

1. G r u p e i n p o d g r u p e

grupni aksiomi
permutacijske grupe
podgrupe, komnožice
normalne podgrupe
polgrupe, proste polgrupe
monoidi
uporaba (sekvenčni stroji, kodi za popravljanje napak, modularna aritmetika, gramatike, jeziki)
--- 20 % ---

2. G r a f i

usmerjeni in neusmerjeni grafi
podgrafi, verige, vezja, poti
cikli, povezljivost
relacije delne urejenosti
prileżne in incidenčne matrike
minimalne poti
elementi transportnih vezij
uporaba (diagrami poteka in prehodnih stanj, primeri iz teorije kodiranja, algoritmi za določanje ciklov in minimalnih poti, listne strukture, odločitvena drevesa, poljski zapis in drevesa, pretoki (poteki) in vezja)

3. M r e ž e i n B o o l o v e a l g e b r e :

aksiomatična opredelitev Boolovih algeber kot algebraičnih struktur z dvema operacijama
dualnost, osnovni rezultati
predikati in predikatne funkcije

logične povezave
pravilnostne vrednosti in tabele
algebra predikatnih funkcij
Boolova algebra pravilnostnih vrednosti
uporaba (preklopna vezja, osnovne računalniške komponente, odločitvene tabele)
--- 25 % ---

4. K o n č n a p o l j a

opredelitev, predstavitev, struktura
minimalni polinomi
neokrajšljivi polinomi
primitivni elementi
polinomski koreni
uporaba (kodi za korekcijo napak, sekvenčni generatorji)
--- 15 % ---

MODUL 3.2

Uvod v računalniške sisteme (3-3-2)

Cilji

a) osnovne zamisli računalniških sistemov

b) uvajanje v računalniško arhitekturo

c) poučevanje zbirnega jezika

Poudarek modula

Urejenost snovi je odvisna od razpoložljivih sredstev. Študent naj spozna podrobnosti zbirnega jezika, dela na projektih in si pridobi programirne izkušnje na specifičnem računalniku. Poudarek naj bo na dovolj širokem območju konceptov in metod. Programirne metode iz modulov 1.1 in 2.4 naj se uporabljajo tudi tu.

Vsebina modula

1. Zgradba računalnika in strojni jezik

pomnilnik, krmiljenje, procesiranje in V/I enote
registri, tipi strojnih ukazov in njihovi formati
programsko krmiljenje
cikel vzetja in izvršitve ukaza
generiranje taktov
V/I operacije

--- 15 % ---

2. Zbirni jezik

mnemonične operacije
simbolični naslovi
koncepti zbirnika in ukazni format
opredelitev podatka-besede
literali
lokacijski števnik
zastavice in sporočila napak
konstrukti visokih programirnih jezikov

--- 30 % ---

3. Načini naslavljanja

indeksiranje
indirektno naslavljanje
absolutno in relativno naslavljanje

--- 5 % ---

4. Makroji

opredelitev, poziv
parametri, razširitev
vgnezdenje
pogojno zbiranje

--- 10 % ---

5. V/I za zbirke

osnovne fizične značilnosti V/I in pomožnih pomnilnih naprav
sistem za krmiljenje zbirk
V/I specifikacijski stavki in periferni (programski) vmesniki
obdelava podatkov (vmesna in blokirajoča)

6. Delitev in povezava programa

subrutine, korutine
rekurzivne rutine in rutine s ponavljajočimi vstopi

--- 20 % ---

7. Zgradba zbirnika

eno in dvoprehodni zbirniki
premeščanje
nalagalnik s premeščanjem

--- 5 % ---

8. Interpretivne rutine:

simulatorji, zasledovanje

--- 5 % ---

MODUL 3.3

Uvod v digitalno načrtovanje in mikroprocesorje (3-3-2)

Cilji

Študent naj pridobi znanje iz delovanja računalniške materialne opreme:

a) na ravnini vrat (vezij)

b) na ravnini registrov

c) na ravnini aritmetične in logične enote

d) na ravnini krmiljenja

e) na ravnini delovanja celote

Poudarek modula

Poudarek naj bo na osnovnih mehanizmih in ne toliko na tehnoloških podrobnostih. Študent naj se usposobi za samostojno analizo logičnega diagrama in njegovega delovanja. Priporočljivo je delo v laboratoriju na problematiki logičnega načrtovanja in pridobivanja izkušenj zlasti na mikroprocesorskem področju.

Vsebina modula

1. Kombinatorna logika

Boolova algebra (operatorji AND, OR, INVERT)
binarne spremenljivke in funkcije
pravilnostne tabele
vrata (vezja) in operatorji
vezja tipa NOR, NAND in EOR
dve ravnini realizacije
iterativna realizacija (naslov, primerjalniki, množilniki, generatorji in preizkuševalniki parnosti, aritmetične in logične enote)
multipleksorji, kodirniki
pomnilna vezja

--- 25 % ---

2. Sekvenčna logika

potreba po sekvenčnih vezjih
pojem stanja
vhodno (asinhrono) in taktno (sinhrono) prožena vezja
bistabilna vezja (taktna in brez takta)
števniki, registri
podatkovne in krmilne pote

--- 25 % ---

3. Mikroprocesorji in mikroračunalniki

zgradba mikroračunalnika (procesor, pomnilnik in V/I)
bitne rezine ter 8- in 16-bitni mikroprocesorji
mikroprocesorska arhitektura (registri, indeksni in skladni kazalec, naslavljalni načini)
V/I vmesniki (paralelni in serijski vmesniki, sistemski takt, taktne faze in bitne hitrosti)
pomnilnik (naključni, čitalni pomnilniki, pomnilniške preslikave V/I)
prekinitve (prekinitveni tipi, obdelava prekinitev, zaporedne in vektorske prekinitve)
načini direktnega pomnilniškega dostopa
programska oprema in pripomočki za razvoj programov (urejevalniki in zbirniki, povezovalniki in makrozbirniki, prečna programska oprema)
aparaturni pripomočki (ocenjevalni in razvojni sistemi, logični analizator, emulacija procesorjev)

--- 50 % ---

MODUL 3.4

Zgradba programirnih jezikov (3-3-2)

Cilji

a) oblikovanje razumevanja organizacije programirnih jezikov, še posebej obnašanja programov v času njihovega izvajanja

b) uvajanje formalnega študija specifikacij in analize programirnih jezikov

c) razvijanje reševanja problemov ter programirnih navad (pristopov) ter njihovo uvajanje v učno snov na elementaren način

Poudarek modula

To je uporabniški tečaj elementov programirnega jezika s poudarkom na obnašanju programov v času izvanja. Tečaj naj bi zagotovil ustrezno podlago za kasnejše zahtevnejše module, ki obravnavajo formalne in teoretične vidike programirnih jezikov in/ali postopke njihovega prevajanja.

Ni potrebno, da je snov modula pokrita zaporedno, tako kot je navedeno kasneje. Programirni jeziki se lahko opredelijo in analizirajo hkrati v odvisnosti od njihovih lastnosti in omejitev z upoštevanjem okolja v času izvajanja. Uvede se razprava o osnovnih (nujnih) specifikacijah programirnih jezikov s poudarkom na njihovi implementaciji. Modul naj vsebuje obilico programirnih vaj v vseh obravnavanih jezikih, s poudarkom na lastnostih posameznih jezikov.

Vsebina modula

1. Zgradba jezikovne definicije:

koncepti formalnega jezika s sintakso
osnovne značilnosti gramatik
regularne in kontekstno svobodne gramatike
Backus-Naurova oblika
dvoumnost
primer Algola

--- 15 % ---

2. Podatkovni tipi in strukture

pregled osnovnih podatkovnih tipov
seznami in drevesa
pripomočki za opis in obdelavo podatkovnih tipov
jezikovni pripomočki za statično in dinamično upravljanje pomnilnika za podatke

--- 15 % ---

3. Krmilne strukture in pretok podatkov

konstrukti programirnega jezika za specifikacijo programskega krmiljenja in prenos podatkov
stavko DO...FOR, DO...WHILE, REPEAT... UNTIL, BREAK
subrutine in procedure
bločne strukture
prekinitve
odločitvene tabele
rekurzija
povezava z dobrim programirnim stilom

--- 15 % ---

4. Razmere v času izvajanja

vpliv okolice v času izvajanja programov
časovni vplivi zaradi različnih lastnosti programirnih jezikov

--- 25 % ---

5. Interpretivni jeziki

primerjava prevajanja in interpretiranja
procesiranje nizov z jeziki, kot je npr. Snobol 4
vektorsko procesiranje z jeziki, kot je npr. APL

--- 20 % ---

6. Leksikalna in gramatična analiza

uvod v leksikalno analizo
pregledovanje
pregledovalniki s končnimi avtomati
simbolne tabele
uvod v gramatično analizo in prevajalnike
navzdolnji razpoznavniki
navzdolnja in navzgornja analiza

--- 10 % ---

MODUL 3.5

Strukture podatkov in algoritmi I (2-2-2)

Cilji

a) uvajanje navadnih podatkovnih struktur
b) alternativne metode za organizacijo in predstavitev struktur podatkov

Poudarek modula

Modul naj ima veliko število primerov, s poudarkom na prostorskih in časovnih zahtevah pri različnih metodah.

Študent naj opravi vrsto nalog, kot so binarna predstavitev znakov in kazalcev, pisanje programov za procesiranje v skladih, numerični problemi z uporabo polj ter obdelava polinomov.

Vsebina modula

1. Uvod

biti, zlogi, besede
predstavitev numeričnih podatkov in kodiranje znakov
kazalec, podatek in informacija
zbirke podatkov kot vozlišča
polja v okviru vozlišč
osnovni registri
združevanje in ločevanje podatkov
primarni in sekundarni pomnilniki in procesiranje
zaporedna in indeksnozaporedna organizacija zbirk

--- 25 % ---

2. Linearne in seznamske strukture

nizi bitov in znakov
skladi in vrste
fizične in logične strukture podatkov
organizacija zbirk
fizično zaporedne in povezane strukture
enostavni seznami (liste)
skupinski seznami
krožni seznami
slovarski seznami
povezani seznami
pleksi
dvojno povezani seznami
obrnjeni seznami
operacije nad naštetimi strukturami
uporaba: skladi in tabelna analiza

--- 30 % ---

3. Polja

večrazsežnostna polja
ortogonalni seznami
zaporedno dodeljevanje
izračunavanje naslovov
raztresena polja

--- 15 % ---

4. Drevesne strukture, množice in relacije

drevesa, binarna drevesa
algoritmi za gibanje po drevesih
povezana drevesa
drevesa in binarno iskanje
drevesa v iskalnih algoritmih

--- 15 % ---

5. Podatkovni tipi visokih jezikov in obdelava podatkov

podatkovni tipi
pripomočki za naslavljanje in za operacije nad polji
strukture
pripomočki za gradnjo zapletenih struktur
dodeljevanje pomnilnika

--- 15 % ---

MODUL 4.1

Informacijski sistemi (2-0-2)

Cilji

Opredelitev informacijskega sistema in organizacijske funkcije; modifikacija teh funkcij z informacijskim sistemom.

Poudarek modula

Poučevanje naj temelji na študiju kodov in skupinskih razpravah, da bi se tako razvijali koncepti planiranja in upravljanja ter se poudarjali kritični vidiki obnašanja odločitvenih sistemov.

Vsebina modula

1. Informacijske zahteve organizacije

neformalni in formalni komunikacijski kanali
opredelitev odločitve
odločitveni kriteriji
običajno (navadno) odločevanje
programirano odločevanje
upravljanje z izjemami
strategijska, taktična in operativna ravnina odločanja
zunanji in notranji informacijski viri in zahteve
integriranje informacijskih sistemov
cena in vrednost informacije
upravljavski inteligentni sistemi
--- 30 % ---

2. Sistemi na operativni ravnini

informacijske potrebe operativnih vodij in njihovih podrejenih
graditev na obstoječih sistemih
--- 10 % ---

3. Sistemi na taktični ravnini

informacijske potrebe srednjega upravljavskega nivoja
sistemi planiranja in upravljanja
spreminjanje organizacijskih potreb
uslužnostne in proizvodno usmerjene organizacije
--- 10 % ---

4. Sistemi na strategijski ravnini

informacijske potrebe izvršilnega upravljavskega nivoja
učinki centralizirane in decentralizirane organizacijske strukture
modeli strategijskega planiranja
analitični in simulacijski modeli odločanja
--- 20 % ---

5. Načini medsebojnega vpliva

upravljavski/računalniški interaktivni sistemi (tehnični in vedenjski pogoji)
sistemski izhodi (tiskani, slušni, grafični)
pogoji podatkovnih baz za hitro odzivne sisteme
prostori upravljavskega krmiljenja
--- 15 % ---

6. Planiranje razvijajočega informacijskega sistema

vodenje projekta za sistemski razvoj
upravljanje informacijske funkcije
navzdolnji in navzgornji pristop k splošnemu upravljavskemu informacijskemu sistemu
integriranje sistemov
--- 10 % ---

7. Merjenje učinkovitosti informacijskega sistema

postavitev meril učinkovitosti
vrednotenje učinkovitosti (merjenje sistemske zmogljivosti, izračun stroškov in zmogljivosti, razmerje cena/zmogljivost)
merjenje uporabniške zadovoljivosti
--- 5 % ---

MODUL 4.3

Operacijski sistemi in zgradba računalnikov I (3-3-2)

Cilji

Operacijski sistem (OS) je središče sodobnega računalniškega sistema. Izdelava in uporaba OS je zahtevna in težavna naloga, ki terja podrobno znanje o materialni in programski tehnologiji ter njuni povezavi (odvisnosti). Predlaga se razvoj malega operacijskega sistema na malem računalniku v obliki laboratorijskega projekta. Učitelj naj določi specifikacije za razvoj in izpelje delo manjših skupin študentov v obliki podsistemov OS (modulov).

Ciljna izhodišča pa so:

a) komunikacije med perifernimi napravami in obdelava prekinitev

b) načrt in analiza malega multiprogramiranega OS s študijem krmiljenja pretokov v okviru upravljanja nalog CPE in podatkov

c) pregled ostalih delov multiprogramiranega OS

d) nabiranje praktičnih izkušenj študentov

Poudarek modula

Računalniški operacijski sistem je zbirka sistemskih programov, katerih naloga je učinkovito upravljanje z vsemi sistemskimi viri, tj. z napravami, programi in podatki. Glavna naloga operacijskega sistema je razvrščanje poslov (opravil), s katerim se opredeljuje, kateri od poslov bo dobil zahtevani vir, ko vir je ali bo razpoložljiv. Algoritmi razvrščanja imajo dva tipa informacije: statično, kot so npr. uporabniško določene prednosti in parametri zahtev za vire ter dinamično, ki predstavlja lastnosti posla v izvajanju, ki jih med izvajanjem opazi sistem. Primer dinamičnega tipa je zahteva strani v glavnem pomnilniku pri virtualnem pomnilniškem mehanizmu. Razvrščanje poslov mora biti občutljivo na določeno politiko urejanja poslov, kot so npr. zahteve realnega časa, toda mora biti neobčutljivo na hujše logične napake, kot je sistemsko blokiranje, ki lahko prepreči dodeljevanje virov in obdelavo poslov. Tesno povezano z obdelavo je spremljanje posla, ko ima sistem pregled nad uporabo virov v poslu, ko se hkrati opravlja analiza računovodstva, razvrščanja in zmogljivosti.

Eden glavnih ciljev sodobnega operacijskega sistema je prožno razdeljevanje virov. Primeri naj vključujejo razdeljevanje časa in multiprogramirane sisteme različnih tipov, vključno z virtualnim pomnilnikom. Takšno razdeljevanje zahteva zaščitne mehanizme, ki preprečujejo vpliv izvajajočega posla na informacijo, ki pripada drugemu poslu, če za to nimajo ustreznega dovoljenja. Pojasnjujejo naj se dovolilni mehanizmi in njihovo vključevanje v operacijski sistem.

Uporabniška povezava z operacijskim sistemom je navadno uresničena s krmilnim jezikom poslov (KJP), ki je lahko zapleten ali enostaven programirni jezik. V snov se lahko vključijo primeri in pripomočki, kot so kompilatorji, sortiranje, ločevanje in dostop do zbirk v okviru operacijskega sistema.

Izdelava operacijskega sistema zahteva določene materialne in programirne postopke. Po-

trebne materialne lastnosti vključujejo prekinitvene mehanizme, zaščito pomnilnika, privilegirane načine pa tudi kakšno obliko premeščanja z materialno opremo. Te funkcije, ki morajo biti prisotne v sistemu in se pogostno opravljajo, so vgrajene v materialno opremo, ostale pa se lahko realizirajo s programi. Materialna oprema vključuje tako poljubno mikroprogramirano zaporedje.

Vsebina modula

1. U v o d v d a n i r a č u n a l n i k

pregled programa za sistem s paketnimi procesi
zbiranje in kompiliranje
nalaganje, izvrševanje
omejitve povezovalnega časa
inkrementno povezovanje
zaporedni procesi
organizacija procesorja
multiprogramiranje
multiprocesorski sistemi
metode naslavljanja: indeksno, indirektno, premeščanje, segmentiranje
organizacija pomnilnika
hierarhija pomnilniških tipov: centralni pomnilnik, boben, disk, trak, ostali
cena in čas dostopa
koncepti dostopa
naključni dostop
direktni, zaporedni dostop
indeksnozaporedni dostop
dostop prek strani
iskalne strategije
asociativni pomnilniki
krmiljenje V/I
kanali
komunikacija s CPE
podatkovne strukture
skladi, vrste, obroči, seznami
procesni in podatkovni moduli
segmenti
procesne podatkovne baze
medsegmentno povezovanje
medprocedurne komunikacije
procesni skladi
obročna zaščita

--- 30 % ---

2. P r o c e s i

paralelnost v operacijskih sistemih
vzroki za paralelno programiranje
prekinitveno procesiranje
semaforji
kritična območja
blokade
vzroki za asinhrono procesiranje
abstraktni opis procesne interakcije
medsebojna izključitev:
  dostop do deljenih podatkov
  kritična območja in čakalne oblike
  osnovne funkcije zapiranja in odpiranja
  operaciji P iv V
sinhronizacija:
  sinhronizacija dogodkov
  blokiranje in zbujanje
  stikalo za čakanje/zbuditev
  P in V kot sinhronizacijski funkciji
krmiljenje preklopov in prometa:
  procesorska stanja in statusne besede:
    izvajanje, pripravljenost
    blokiranje, neaktivnost
    materialni mehanizmi prekinitev

  algoritmi razvrščanja:
    procesor in procesno razvrščanje
    krožno razvrščanje
    delitev procesorja
    prednosti itn.
  sistemske tabele in krmilni bloki za
    procesno upravljanje
    realizacija osnovnih funkcij
  sistemske blokade:
    pojem problema
    razreševanje blokad
    zaščita pred blokado

--- 30 % ---

3. M u l t i p r o g r a m i r a n i s i s t e m i

osnovne funkcije
metode:
  zahteve V/I
  časovne rezine
  drugo
komunikacija s periferniki
nadzorovanje V/I
upravljanje vrst
upravljanje pomnilnika

--- 20 % ---

4. D r u g e z a m i s l i

multiprocesorski sistemi
virtualni pomnilnik in virtualni stroj
paketni sistemi in sistemi z delitvijo časa
zaščita
upravljanje pomnilnikov
upravljanje zbirk
sistemsko računovodstvo
daljinsko krmiljenje poslov
hierarhično krmiljenje poslov
mikroprogramiranje
ukazni jeziki
operacijsko obnašanje
paralelno asociativno in pretočno procesiranje
mreže

--- 20 % ---

MODUL 4.5

Strukture podatkov in algoritmi II (2-2-2)

Cilji

a) uvajanje zapletenih podatkovnih struktur
b) seznanjanje študentov s koncepti, ki se uporabljajo pri upravljanju podatkovnih baz

Poudarek modula

Vsebina se navezuje na snov modula 3.5. Poudarja naj se, da ima vsaka podatkovna struktura podatkovno organizacijo in metodo dostopa. Obravnava naj se vsaj upravljanje pomnilnika tipa sklad (LIFO) in uporaba sekljalnih tabel in algoritmov z navzkrižji.

Vsebina modula

1. K o m p l e k s n e s t r u k t u r e

grafi
rekurzivni seznami
obročne strukture

--- 10 % ---

2. U p r a v l j a n j e p o m n i l n i k a

uporaba pomnilnika tipa sklad:
  skladi
  bloki določenega obsega
  bloki spremenljivega obsega
  najboljše prileganje
  prvo prileganje itn.
hierarhično dodeljevanje pomnilnika
  pomnilniške zaloge
zbiranje odpadkov, referenčno štetje
markirni in zasledovalni algoritmi
metode:
  razpoznavanje pomnilnih blokov z urejevanjem obsegov
  razpoznavanje blokov z urejevanjem naslovov
  problem odlomkov
  delitev blokov
  pregibanje
  strnjevanje

premeščanje
splošni pomnilniški premeščevalniki
učinkovitost in uporaba teh metod
--- 40 % ---

3. D o d a t n e t e m e

uvod v sortiranje
sekljalne tabele
metode direktnega dostopa
shranjevanje tekstov
kompresija podatkov
--- 20 % ---

4. P r i m e r i

linearna polja in enostavni seznami
večrazsežnostna polja
raztresene matrike in ortogonalni seznami
enostavni slovarski in drevesno organizirani slovarski seznami
enostavno in dvojno povezani seznami
dodeljevanje pomnilnika z urejenostjo obsegov in z urejenostjo naslovov
--- 30 % ---

MODUL 4.4

Uvod v obdelavo zbirk (3-2-2)

Cilji

a) uvajanje zamisli in metod podatkovnega strukturiranja na obsežnih (prostornih) pomnilnih napravah

b) pridobivanje izkušenj z uporabo obsežnih pomnilnih naprav

c) pridobivanje osnov za uporabo podatkovnih struktur in metod obdelave zbirk

Poudarek modula

Poudarek bo odvisen od računalniške opremljenosti za vaje študentov. Organizirajo naj se programirni projekti s področje obdelave zbirk. Prikažejo naj se značilnosti in uporabnost različnih pomnilniških naprav, tudi če te naprave niso del računalniškega sistema. Uporabijo naj se algoritmična analiza in programirne metode, ki so bile razvite v modulu 2.4.

Vsebina modula

1. U v o d v u p o r a b o j e z i k a

za upravljanje podatkovnih zbirk
za upravljanje poslov
--- 5 % ---

2. P r e g l e d V/I s i s t e m s k e a r h i t e k t u r e

tipi pomnilniškega dostopa:
  nakljućni
  zaporedni
  krožni
glavne značilnosti tipičnih komponent pomnilniške hierarhije:
  kapacitivnost
  prenosna hitrost
  cena
  čas dostopa
pomnilniški dostop:
  kanali
  kanalski programi
  V/I prekinitve
--- 25 % ---

3. L o g i č n a o r g a n i z a c i j a z b i r k

uporabniška izhodišča
zapisi, zbirke
zaporedne zbirke
nakljućne zbirke
hierarhične zbirke
obrnjene zbirke
delno obrnjene zbirke
multiseznamske zbirke
asociativne (vsebinsko naslovljive) zbirke
operacije:
  razpoznavanje
  popravljanje
  dodajanje
--- 25 % ---

4. P r e s l i k a v a l o g i č n e o r g a n i z a c i j e n a f i z i č n i p o m n i l n i k

merila zmogljivosti
vmesno shranjevanje
blokiranje in doblokiranje
strukture imenikov
drevesa, ostalo
--- 10 % ---

5. P o v e z a v a z b i r š n e g a s i s t e m a z o p r a c i j s k i m s i s t e m o m

odpiranje in zapiranje zbirk
pridobivanje dostopa do zbirk
  zaščitni mehanizmi
  politika varnosti
metode dostopa:
  metoda vrstnozaporednega dostopa (QSAM)
  metoda indeksnozaporednega dostopa (ISAM)
  metoda hierarhičnozaporednega dostopa (HSAM) itn.
oblikovanje in brisanje zbirk:
  upravljanje sekundarnih pomnilnikov
sistemski katalogi
procedure za shranjevanje (kopiranje)
popravljanje zbirk
--- 25 % ---

6. U p r a v l j a n j e z b i r k z v i s o k i m i j e z i k i

V/I pripomočki visokih jezikov:
  fortran, cobol
  pl/I, dl/I
--- 10 % ---

MODUL 5.3

Operacijski sistemi in zgradba računalnikov II (3-3-2)

Cilji

a) preučevanje operacijskih sistemov in z njimi povezanih metod

b) razumevanje nujnosti med zanesljivostjo, splošnostjo, učinkovitostjo, zapletenostjo in združljivostjo

Poudarek modula

Nadaljuje se snov iz modula 4.3 s poudarkom na notranjih sistemskih komunikacijah. Operacijski sistem razpoložljivega računalnika se naj uporablja kot primer. Študentje naj pišejo različne dele operacijskega sistema in si pridobivajo praktične izkušnje.

Vsebina modula

1. I z h o d i š č a z a a n a l i z o i n n a č r t o v a n j e O S

zanesljivost
splošnost
učinkovitost
razširljivost
--- 5 % ---

2. U p r a v l j a n j e p o m n i l n i k a

potreba o sodobnih pomnilniških sistemih
  metode enoravninskega pomnilnika
  metode imenskega upravljanja

uvod in razprava o abstrakcijah:
  naslovni in pomnilni prostori
  naslovna preslikava
  dvoravninski pomnilni prostor
  veliki virtualni pomnilniki
  vezalni čas

lastnosti enoravninskega pomnilnika:
  doseganje programirnih ciljev
  doseganje sistemskih ciljev
  premeščanje
  povečevanje hitrosti
  glavni in vmesni pomnilniki
  razmerje prostor/čas
  zaščita

implementacija:
  oblika in uporaba tabel
  fiksni in spremenljivi obsegi blokov

strategija dodeljevanja:
  jemanje iz pomnilnika
  izmenjava
  obseg bloka
  zamisel stranenja
  pomožno upravljanje pomnilnika
  izginjanje strani itn.

razširitev na multiprogramiranje:
  osnovni registri in skupni bloki
  zamisel upravljanja

--- 25 % ---

3. I m e n s k o u p r a v l j a n j e

vzroki:
  dostop do procedur in baz podatkov
  sistemska obravnava imen

osnovni koncepti:
  oblike imen
  identifikacija
  naslovi
  prevod imen s kompilatorji, nalagalniki in s sistemom
  kontekst za predstavitev imen
  primeri

sistemi zbirk:
  zbirke
  hierarhija imenikov
  struktura zbirk
  predstavitev zbirk na napravah
  implementacija
  doseganje sistemskih ciljev
  omejitve pri sistemih zbirk

segmentirani naslovni prostor:
  segmenti
  dvodelni naslovi
  povezovalna procedura za podatke na naslovni prostor
  implementacija z osnovnimi registri in s stranenjem

dinamične strukture
  polja s spremenljivimi obsegi
  povezane strukture
  upravljanje naslovnega prostora za dinamične strukture
  upravljanje prostega prostora
  zbiranje odpadkov
  strnjevanje
  kršitve imenskega principa
  zahteve za velikim, segmentiranim naslovnim prostorom

modularnost:
  osnovne zahteve
  ustrezna predstavitev podatkov
  povezovalni dogovori
  zamisel modularne uporabe procedur
  struktura procedur
  struktura argumentov
  sistemske omejitve modularnega programiranja
  primeri v fortranu, algolu 60, algolu w ali pascalu, algolu 68 ali v pl/I

delitev virov
  uporaba povezav v imenikih za omogočitev kontroliranega dostopa
  vzroki za deljeno uporabo informacije v glavnem pomnilniku
  implementacijske alternative
  skupni naslovni prostor
  razdeljeni naslovni prostor z relativnim naslavljanjem
  problemi povezave

--- 30 % ---

4. Z a š č i t a

vzroki zaščite
  kontrola uporabnikov
  kontrola programov
območja zaščite
  različni konteksti
  idealiziran sistem s sporočili
slabosti predložene zaščite
metode implementacije
zaščita pomnilnika

--- 30 % ---

5. D o d e l j e v a n j e v i r o v

dodeljevalne strategije
  viri in prioritete
  dodeljevanje procesorjev
  dodeljevanje pomnilnika
  pristopi dodeljevanja procesorja/pomnilnika
strategijsko vrednotenje
  verjetnostni modeli
  deterministični in diskretni modeli
  simulacija
  naknadno vrednotenje
uravnovešenje virov in zahtev

--- 10 % ---

MODUL 5.4

Telekomunikacije in porazdeljeni sistemi (2-0-2)

Cilji

a) pregled problemov multiprogramiranja v omejenem okolju (več procesorjev si deli lokalni, skupni pomnilnik)

b) pregled problemov povezovanja geografsko porazdeljenih računalnikov v komunikacijski mreži

Poudarek modula

Poučevanje naj temelji pretežno na primerih. Zamisel podatkovne komunikacijske mreže se primerja s standardno telefonsko mrežo, vendar s poudarkom na razlikah

Vsebina modula

1. M u l t i p r o c e s o r j i i n p o r a z d e l j e n a o b d e l a v a

centralizacija in decentralizacija
viri in njihovo lastništvo
koordinacija procesov

--- 10 % ---

2. I s t o č a s n o s t i n s o d e l o v a n j e r a z p r š e n i h p r o c e s o v

teoretični
topološka izbira za obdelavo, preklapljanje, usmerjanje in krmiljenje
programi za krmiljenje mrež

--- 10 % ---

3. P r o t o k o l i

komunikacije in programski protokoli
upravljanje mreže
  interaktivni terminali
  terminali tipa RJE
zmogljivost protokolov

--- 20 % ---

4. S i s t e m s k i p o g o j i

načini prenosa
  simpleksni, polduplesksni in dupleksni prenos
multipleksiranje in koncentracija
odtipavanje
kodiranje za kontrolo napak
linijske discipline

PROGRAMSKA PODRŠKA ZA MIKROPROCESORSKI UPRAVLJANU JEDINICU MAGNETSKIH KAZETA

MARIO ŽAGAR
NIKICA RENDIĆ*

ELEKTROTEHNIČKI FAKULTET, ZAGREB
ULJANIK RO TESU, ZAGREB

UDK: 681.327.63.06

U članku je opisana programska podrška za kontrolu kompjuterskih perifernih jedinica upravljanih mikroprocesorom.Razmotrene su mogučnosti realizacije takvih kompjuterskih perifernih jedinica.

SOFTWARE SUPPORT FOR MICROPROCESSOR CONTROLLED DATA CARTRIDGE UNIT.In this article control of computer peripheral units using microprocessors is considered.Software support for such control is described.Possibilities of realizing computer peripherals is discussed.

1. UVOD

Velik napredak u razvoju mikroračunala u posljednjih nekoliko godina zahtijeva isto tako i razvoj svih popratnih elemenata koji su nužni da bi se mogućnosti koje nam danas mikroračunala pružaju mogle u potpunosti iskoristiti.

Unatoč velikim naporima koji se ulažu na tom području,osjeća se velik raskorak.Od pojave novog tipa mikroprocesora (boljih karakteristika od njegovih predhodnika) potreban je izvjestan period vremena da bi se razvile i sve popratne komponente koje mogu optimalno zadovoljavati potrebe mikroprocesora (ulazno/izlazne komponente,memorijske komponente,elementi specijalne namjene i sl.).

Što se tiče vanjskih jedinica mikroračunala,taj raskorak još više dolazi do izražaja.Na današnje "moderne" vanjske jedinice računala postavljaju se vrlo veliki zahtjevi:

- velika brzina rada
- velik kapacitet memorije
- mala potrošnja energije
- male fizičke dimenzije
- velika pouzdanost
- mala cijena koštanja
- te kao najvažnija karakteristika- sposobnost izvršavanja složenih zadataka uz minimalnu intervenciju glavnog računala (mikroračunala).

Da bi sve to bilo moguće nužno je koristiti se najnovijim dostignućima tehnologije u proizvodnji magnetskih materijala,mehaničkim konstrukcijama i rješenjima te korištenjem mikroračunala za upravljanje radom vanjskih jedinica računala.

2. MOGUĆNOSTI REALIZACIJE MIKROPROCESORSKI UPRAVLJANIH VANJSKIH JEDINICA RAČUNALA

Za konkretnu realizaciju odredjenih "inteligentnih" vanjskih jedinica računala potrebno je savladati čitav niz prepreka od idejnog rješenja,izrade prototipa do organizacije proizvodnje i održavanja.

Ne ulazeći u sve te probleme potrebno je možda samo naglasiti da se korištenjem mikroprocesora i popratnih elemenata elektronički dio materijala svodi na minimalan broj komponenata,a bitno se povećava programska podrška koja u jednom takvom uredjaju preuzima na sebe veliku većinu zadataka (prije rješavanih klasičnim sklopovima),a osim toga pruža niz no-

Referirano na stručnom sastanku"mikroproc.računala" od24 do 25 IX.1980 na EF u Zagrebu

vih mogućnosti koje se klasičnim sklopovima ne da riješiti ili je to vrlo komplicirano i nespretno.

Sva "pamet" jedne moderne vanjske jedinice računala realizirana je korištenjem dosta složenog programskog paketa koji upravlja radom cijele jedinice.

Važna je činjenica da u cijeni gotovog proizvoda manji dio iznose komponente,a najbitniji dio je programska podrška koja je vrlo skupa.

Činjenica je takodjer da mikroprocesorske komponente za sada ne proizvodimo medjuti m programsku podršku potrebnu za uspešan rad mikroprocesora možemo realizirati u domaćim uvjetima.

3. USPOREDBA REALIZACIJE KLASIČNIM SKLOPOVIMA I MIKROPROCESORSKE REALIZACIJE

Već je prije spomenuto da moderne vanjske jedinice računala moraju zadovoljavati niz zahtjeva koji se postavljaju na njih."Klasičnim" načinom realizacije potreban je velik broj komponenata što zahtjeva velike izvore napajanja, fizički prostor,veću cijenu upotrijebljenog materijala (više štampanih pločoca,veće kutije, više integriranih elemenata) uz smanjenje pouzdanosti rada zbog velikog broja elemenata i spojnuh linija što je kod "modernih" realizacija korištenjem mikroprocesorskih komponenata svedeno na najmanju moguću mjeru.

No svi bi ti nedostaci "klasičnih" rješenja mogli biti zanemareni ako ne bismo uzeli u obzir najbitnije prednosti koje nam mikroprocesorska realizacija pruža,a to su:velika elastičnost u izmjenama načina rada i mogućim promjenama i dopunama koje s vremenom korisnici zahtijevaju te izvanredne mogućnosti samostalnog rješavanja niza problema od strane samog uredjaja koje se klasičnim načinima vrlo teško ili nikako ne mogu realizirati.

Za konkretan uredjaj,jedinicu magnetskih kazeta to su: visok stupanj komunikacije izmedju jedinice i vanjskog svijeta (potreban je minimalan broj parametara za izvršavanje jedne ili niza složenih operacija),o izvršavanju se brine sama jedinica,rješavajući moguće nepredvidive situacije samostalno i zahtijevajući intervenciju iz vana samo u neophodnim slučajevima i dr. Te prednosti moguće su zato jer se upravljanje radom uredjaja vrši korištenjem odredjenih programskih paketa.

4. REALIZACIJA PROGRAMSKOG PAKETA ZA MIKROPROCESORSKI UPRAVLJANU JEDINICU MAGNETSKIH KAZETA

Analiza rada jedinice magnetskih kazeta i mogućnosti realizacije njenog upravljanja pomoću mikroprocesora razmatrane su na mikroprocesorskom sistemu F8 čija je oprema bila dostupna.Na slici 1 prikazan je programsko-sklopovski dio potreban za realizaciju upravljanja radom jedinice magnetskih kazeta.

Mikroprocesorski sistem sa mikroprocesorom F8 i odgovarjaućom podrškom (monitor, asembler,editor) iskorišten je i za direktno upravljanje jedinice magn. kazeta i za simuliranje komunikacije magn. kazete s glavnim računalom.

Ovakav pristup rješavanju problema upravljanja jedinice magn. kazeta pomoću mikroprocesorskog sistema ima općenit karakter te se može upotrijebiti i za razvoj drugih vanjskih jedinica računala.

Budući da je cijela problematika programskog rješenja upravljanja radom jedinice magn. kazeta te komunikacije s vanjskim svijetom složena i dosta opširna,prišlo se rješavanju problema po odredjenim cjelinama:

- razmatranje i rješavanje problema komunikacije jedinice magn. kazeta prema glavnom računalu
- komunikacija s korisnikom u lokalnom načinu rada
- komunikacija prema samoj jedinici magn. kazeta i njezino upravljanje.

Programski paket koji je realiziran , omogućavao je različite režime rada:

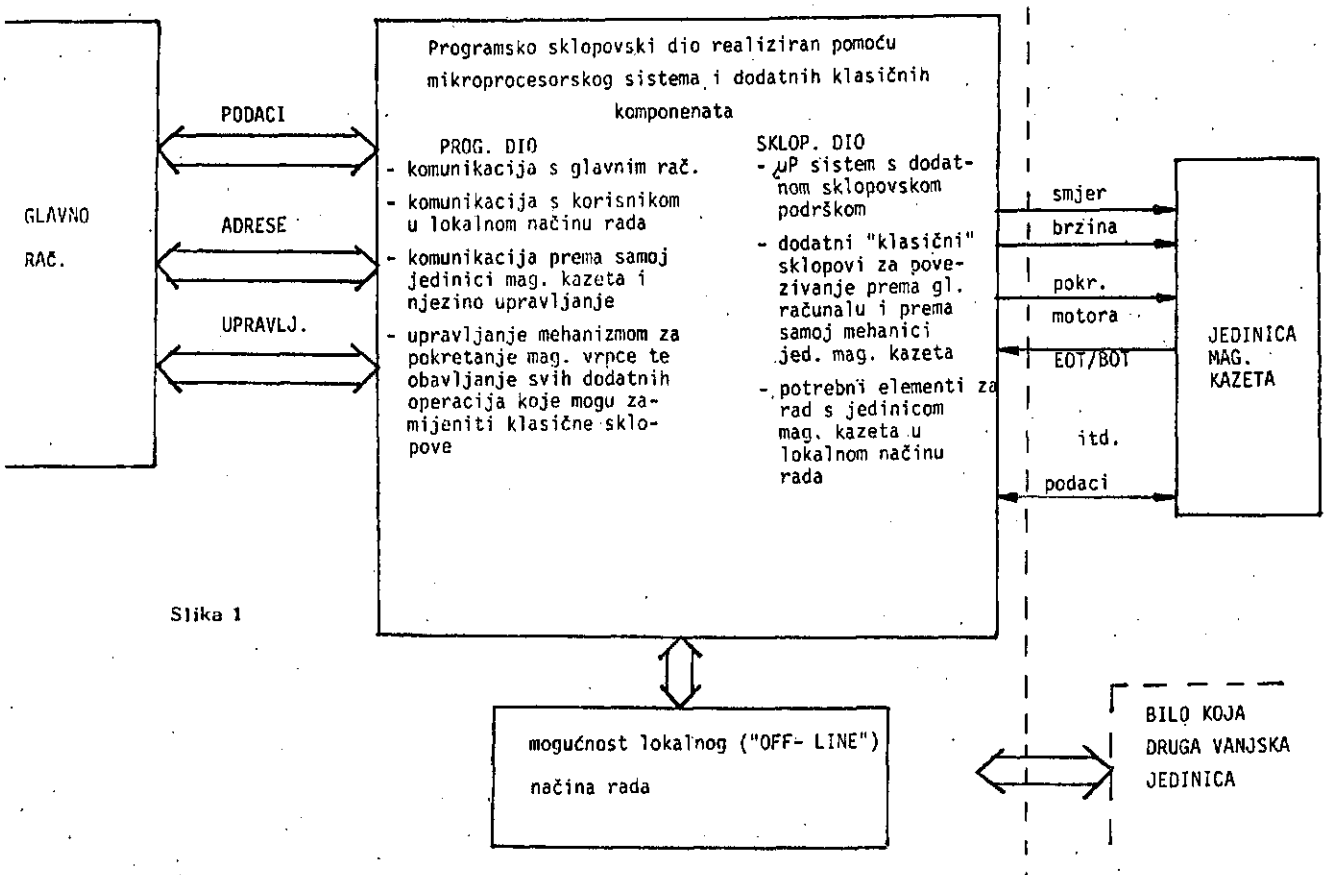
- kao masovna memorija računala
- direktna zamjena za čitač/busač papirne trake
- jedinica za prikupljanje podataka u lokalnom načinu rada.

Iskustva stečena tokom samog razvoja i analize rada mikroprocesorski upravljanje jedinice magn. kazeta korištena su u konkretnoj realizaciji upravljanja pomoću mikroprocesora Z80 i dodatnih komponenata koje su u medjuvremenu postale dostupne.

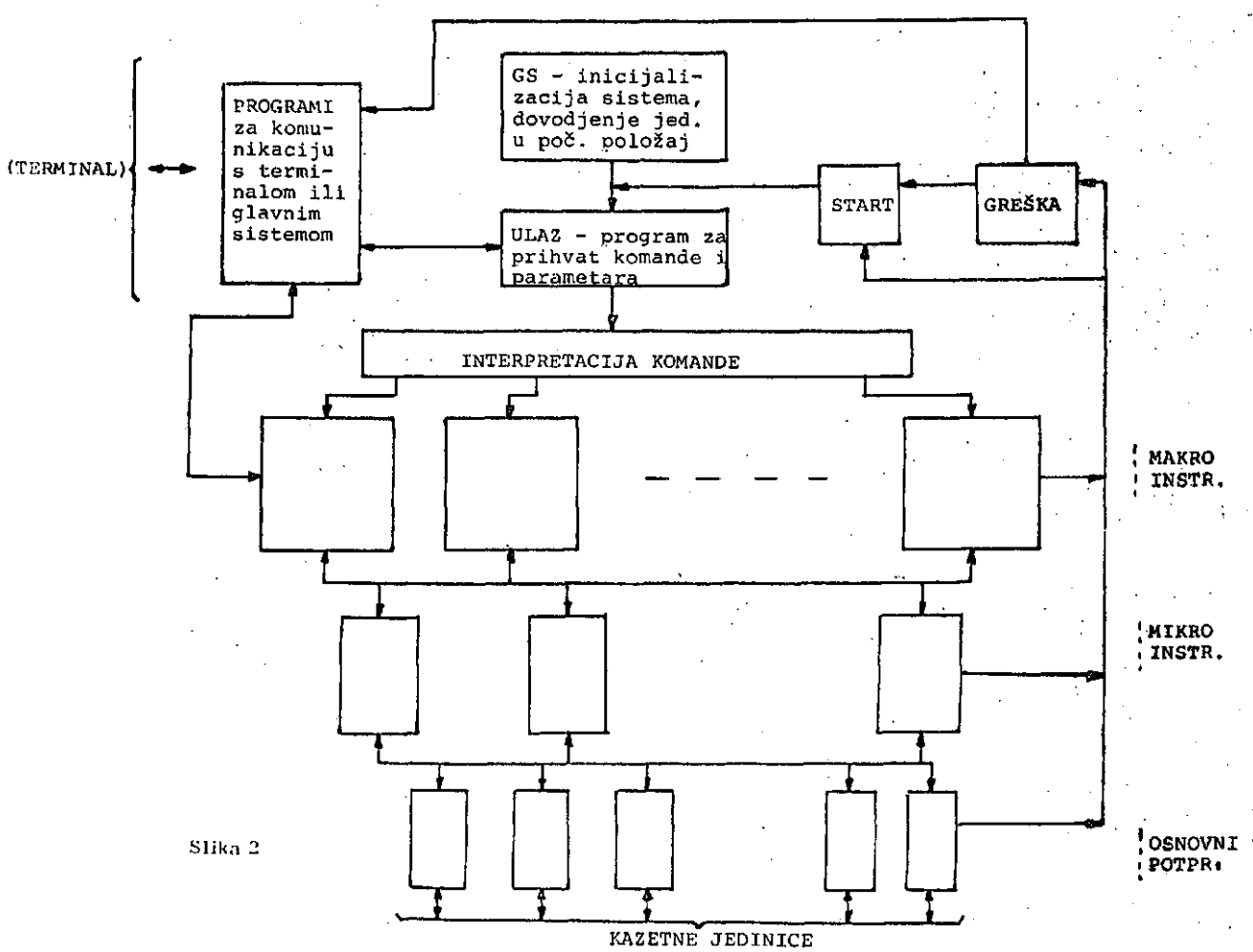
Koncepcija programa koji preko mikroprocesora Z80 upravljaju radom jedinice magn. kazeta dana je na slici 2.

Bitni dijelovi programskog paketa su:

- programi za inicijalizaciju
- interpreter komandi
- programi za paralelnu,i serijsku komunikaciju
- programi za uprav.radom same jed.magn.kazeta
- dijagnostika pogrešaka



Slika 1



Slika 2

Korištenjem Z80 mikroprocesorskih komponenata potrebno je sve komponente isprogramirati za konkretnu namjenu,a takodjer inicijalizirati cijeli uredjaj za odredjeni način rada (odredredjivanje veličine bloka podataka koji se čita ili zapisuje na kazetu,selektiranje jedne od četiri moguće jedinice za zapis,odredjivanje jednog od četiri traga na odabranoj jedinici,pozicioniranje na početak i dr.

Interpreter komandi brine se otome da čovjeku blisku komandu pretvori u odgovarjajući niz parametara potrebnih za izvršenje odredjenih operacija,upozorava na neispravnosti i odredjena stanja jedinice i povezuje vanjski svijet sa upravljačkim programima.

Komunikacija s jedinicom magnetskih kazeta može ići prega serijskog ili paralelnog kanala.Ponašanje uredjaja nije ovisno o vrsti komunikacije osim kod čitanja odnosno zapisa niza blokova podataka.Paralelni prijenos omogućava ,za razliku od serijskog,vrlo brz tok podataka i ako je brzina prijenosa veća ili jednaka brzini čitanja odnosno pisanja na kazetu (48000 bit/s),čitanje i pisanje niza blokova podataka može se izvršavati bez zaustavljanja trake nakon svakog bloka.To povečava brzinu kazetne jedinice i bitno poboljšava režim rada mehanike kazetnog sistema.

Programi za upravljanje samim mehanizmom jedinice magn. kazeta predstavljaju najsloženiji dio.(Tip kazete koja se koristi je DC300A). Od trenutka izdavanja naredbe od strane interpretera sve akcije uredjaja su pod kontrolom tih programa.Ograničenost prost ora dozvoljava samo kratko nabrajanje tih akcija:pokretanje i zaustavljanje motora,njegovo ubrzavanje i usporavanje,odredjivanje jedinice i kanala na njoj, odredjivanje pozicije magnetske kazete,razlikovanje podataka od medjublokovskih praznina i oznaka polja,pomicanje odredjeni broj polja i blokova naprijed i natrag uz istovremeno pamćenje o trenutnoj poziciji,evidencija o zapisanim ili pročitanim podacima,višestruko ponavljanje čitanja ili zapisivanja podataka u slučaju greške i niz drugih stvari.

Dijagnostika pogrešaka brine se o tome da se sve operacije izvršavaju ispravno.U slučaju da se dana komanda zbog nečega ne može izvršiti (jedinica isključena,kazeta izvadjena, kraj trake,itd.) ili je tokom izvršenja komande došlo do neke greške (ne postoji traženo polje ili blok,"hard error" kod čitanja ili zapisivanja) šalje se obavjest u obliku poruke ili koda greške.

Koncepcija programa je takva da se kod inicijalizacije odredjuje status svakog od priključenih kazetnih sistema,odredjuje se vrsta komunikacije i definira veličina bloka podataka koji će se zapisivati.Fiksnim duljinom bloka postiza se mogućnost kasnijih izmjena nad zapisanim podacima ("editiranje") tako da se ne unište ostali podaci.Upravljački program smješten je u 4K EPROM memorije i koristi 5K (8-bitnih) lokacija RAMa.Na slici 3 navedene su komande koje izvršava ovaj uredjaj.Uz komande se dodaju parametri kojima se definira jedinica,trag,polje i blok na koji se komanda odnosi.

R - čitanje pojedinačnih blokova
W - zapis pojedinačnih blokova
CR - čitanje niza blokova
CW - zapis niza blokova
F - zapis oznake polja
P - pozicioniranje
S - status
T - brisanje traga
I - inicijaliz.jedinice,traga i dužine bloka
U - izbacivanje kazete

Slika 3

npr.:R 23 - čitanje trećeg bloka u drugom polju na trenutno aktivnoj jedinici i tragu
CW 20 - zapis dvadeset blokova podataka
I 11 128 - inicijalizacija prvog traga (0-3) na prvoj jedinici (0-3).Veličina bloka je 128 bajt.
P 3214 - pozicioniranje glave na drugom tragu terće kazete za četiri bloka u prvom polj.Itd.

5. ZAKLJUČAK

Realizacijom mikroprocesorskog upravljanja odredjenim procesom(u konkretnom slučaju radom jedinice mgn.kazeta)omogućena je zamjena za klasične elemente koji su se do sada upotrebljavali i koji u usporedbi s mikroproc.komponentama imaju niz nedostataka.Omogućene su i neke dodatne mogućnosti jednostavnog odabiranja svojstava uredjaja minimalnim izmjenama u upravljačkom paketu programa i poboljšanje rada u odnosu na klasične elemente i što je najbitnije dodan je niz svojstava i mogućnosti rada koje se klasičnim načinom uopče ne bi mogle realizirati.Ovakav pristup i dobiveni rezultati pokazuju da u domaćim uvjetima možemo realizirati složene i skupe uredjaje uz minimalan uvoz osnovnih komponent.i domaći razvoj mikroproc.sklopovskih i programskih rješenja.

LITERATURA

1.M.Žagar:Prednosti upotrebe mag.kazete DC300A u odnosu na klasične,Informatika,1978;
2.M.Žagar:Primjena mikroproc.komponenata Z80, Informatica,br.3,1978;
3.M.Žagar:Uprav.komjut.periferijskih jedinica.. ETF ZGB.1978,Mag.rad;
4.M.Žagar,N.Rendić:ETF Zagreb,1980(izvještaj);
5.G.Smiljanić:mikroproc.i mogućnjihove proizv. u Jugoslaviji,Automatika 1977;

KEMIJSKI INFORMACIJSKI SISTEMI II ALGORITMI ZA OBRAVNAVO IN OBDELAVO KEMIJSKO-STRUKTURNIH INFORMACIJ

UDK: 681.3 : 54

B. DŽONOVA-JERMAN-BLAŽIČ
N. TRINAJSTIĆ*

INSTITUT JOŽEF STEFAN, LJUBLJANA
*INSTITUT RUDJER BOŠKOVIĆ, ZAGREB

Vsebina: V prispevku smo podali kratek pregled algoritmov za obravnavo in obdelavo računalniško zapisanih kemijskih struktur. Poskusili smo oceniti njihovo učinkovitost in uporabnost z ozirom na funkcije v okviru kemijsko-informacijskega sistema.

CHEMICAL INFORMATION SYSTEMS II: ALGORITHMS FOR HANDLING AND PROCESSING CHEMICAL INFORMATIONS.
Abstract: The paper discusse the computer-based algorithms that support the handling and processing of information about chemical substances. Some assesments are made about the efficiency and usefulness of the algorithms that performe interconversion, registration, structure and substructure searching, according to the functions they realize in the chemical information systems.

1. UVOD

Vsak informacijski sistem definirajo štiri funkcionalne enote(1):

- zbiranje in shranjevanje informacij tekom določenega časovnega intervala,
- tehnike in metode vnašanja novih informacij ter poizvedovanje o zahtevanih informacijah, ali z drugimi besedami priprava odgovorov na vprašanja uporabnikov,
- skupina ljudi, ki vodi in oblikuje informacijski sistem. *Skupina izloča uporabne informacije, dopolnjuje podatkovno bazo*, ažurira podatke v podatkovni bazi, pripravlja vprašanja, posreduje odgovore uporabnikom ter implementira in dopolnjuje tehnike in metode za izvajanje različnih funkcij sistema,
- skupina uporabnikov sistema, ki ovrednoti sistem glede na svoje potrebe in kriterije.

Kemijski informacijski sistemi (KIS v nadaljevanju) imajo vse značilnosti splošnih informacijskih sistemov, so pa glede na svojstvenosti metod obdelave in posredovanja informacij dobili posebno mesto v okviru splošnih informacijskih sistemov (2). Najbolj pogoste zahteve, oziroma vprašanja uporabnikov KIS najdemo v naslednji skupini vprašanj:

- ali je določena spojina že navedena v literaturi (kdaj in od koga),
- ali obstajajo spojine, ki so podobne po strukturi spojini definirani v vprašanju,
- katere so lastnosti te spojine (fizikalne, biološke, kemijske ipd.),
- kakšne so in katere so skupne lastnosti spojin iz navedene skupine spojin,
- kako pripravimo oziroma kako sintetiziramo navedeno spojino ali serijo spojin,
- kdaj je spojina prvič sintetizirana, kateri so postopki sinteze ipd.,
- katere spojine imajo navedene lastnosti.

Vsa navedena vprašanja zahtevajo, ali identifikacijo spojine v podatkovni datoteki ali poizvedovanje o lastnosti spojine. Vprašanja v katerih je struktura spojine natančno definirana in, ki zahtevajo identifikacijo spojine v sistemu imenujemo strukturno-definirana vprašanja.

Vprašanja v katerih so definirane fizikalno-kemijske lastnosti spojine in, ki kot odgovor zahtevajo ime spojine ali skupine spojin, ki kažejo določene lastnosti imenujemo vprašanja za iskanje po vsebini datoteke (3). Ta vprašanja se definirajo s pomočjo deskriptorjev (4). Deskriptorji so besede ali kode s katerim so opisane lastnosti spojin ali kakšen drug pojem kot je: število in identiteta atomov v spojini, število in identiteta kemijskih zvez v spojini, molekulska teža, število in identiteta obročev, opis okolja določene podstrukture v spojini, molekulska povezanost, molekulska geometrija, molekulski model v prostoru ipd.(4). Na sl. 1 smo pokazali kot ilustracijo elemente, ki definirajo podatkovno bazo "Toxicology Data Bank" (3) realizirane v okviru programa Library's Toxicology Information Program v ZDA.

Večina obstoječih KIS je načrtovana tako da omogoča dostop do informacij na oba omenjena načina: glede na strukturo spojine in glede na vsebino datoteke. Za te namene v okviru KIS najbolj pogosto se kreirajo invertirane datoteke, ki omogočajo hiter dostop do informacij z različnimi ključi oziroma deskriptorji (5), (6). Poleg tega zaradi fleksibilnosti in vse večjih potreb uporabnikov, je v velikih KIS omogočeno zapisovanje strukture spojine na več načinov. Primerno temu, so implementirane tehnike, ki omogočajo transformacije med različnimi zapisi, ali drugače povedano interkonverzijo med zapisi. Algoritmi, ki omogočajo avtomatsko interkonverzijo med različnimi predstavitvami spojin realizirajo nekatere funkcije v KIS, od katerih navajamo najbolj pomembne:

- obdelava strukturnih diagramov ter priprava strukturnih formul v obliko sprejemljivo za publiciranje,
- prikazovanje strukturnih diagramov na video-terminalih, na podlagi zapisov z linearnimi notacijami,
- transformacija strukturnih diagramov iz video-terminalov v zapise z obliko tabele povezanosti,
- izmenjava podatkov med različnimi bazami podatkov.

V nadaljevanju bomo na kratko obdelali ter analizirali skupino najbolj pomembnih algoritmov, na katerih sloni realizacija vseh osnovnih funkcij KIS.

Toxicology Data Bank

Podatkovni elementi

1. Identifikacija substance
   a) kemijsko ime
   b) registrska številka službe CAS
   c) sinonimi
   d) molekulska formula
   e) molekulska teža
   f) zapis v Wiswesserjevi notaciji
2. Razvrstitev substance
   a) kemijski razred
   b) najbolj pogosta uporaba
3. *Kemijsko/fizikalne lastnosti*
   a) temperatura topljenja
   b) temperatura vrenja
   c) gostota/specifična teža
   d) barva/oblika
   e) stabilnost/življenjska doba molekulskih obel
   f) spektroskopski in drugi podatki
   g) raztopljivost
4. Toksikološki učinki: eksperimentalne študije:
   a) na živalih
   b) z ljudmi
5. Toksikološke vrednosti
   a) minimalna strupena količina
   b) *maksimalna dovoljena dnevna količina*
   c) LD-vrednosti
6. Laboratorijske metode in sinteze
7. Interakcije v bioloških sistemih
8. Farmakologija
   a) metabolizem
   b) absorpcija, distribucija, izločanje
9. Farmakoterapija
10. Ukrepi v primerih zastrupitve oziroma eksplozije
11. Informacije proizvajalcev
12. Metode prevoza
13. Podatki v zvezi z okoljem
   a) meje eksplozivnosti
   b) *možnosti vžiga*
   c) meje strupenosti
   d) meje radioaktivnosti
   d) meje onesnaženja
   f) meje izpostavljanja
   g) prag mejnih vrednosti
   h) kopičenje, razgrajevanje in obstojnost v okolju.

Slika 1.

2. REGISTRACIJA SPOJINE

Registracija spojine je algoritemski postopek, ki *omogoča sprejemanje, povezovanje in ureditev vseh informacij* v KIS, ki se nanašajo na določeno spojino. Postopek mora najprej ugotoviti ali se v datoteki nahaja snov, ki je po strukturi ekvivalentna kandidatu za vpis v podatkovni datoteki. V primeru da takšne spojine ni, zapis nove spojine ter ostale informacije se uvrstijo na ustrezno mesto glede na lastnosti in konfiguracijo spojine.

Postopek za registracijo spojin je ozko vezan in omejen z osnovnim sistemom za predstavitev spojin. Najbolj pomembna faza je primerjava strukture kandidata s strukturami spojin v osnovni datoteki sistema. Uporabljene metode so zelo različne in prilagojene uporabljeni notaciji za predstavitev kemijskih struktur. Za vse metode pa velja naslednje:

- kandidat za registracijo je zmeraj zapisan na enoličen in nedvoumen način,
- osnovna datoteka je urejena tako da omogoča urejanje spojin v skupine s skupnim strukturnim značajem,
- poleg osnovnega zapisa, omogočeno je vnašanje dodatnih parametrov, (to je najbolj pogosto molekulska teža ali molekulska formula), zaradi kontroliranja napak.

Če je datoteka organizirana tako da so spojine grupirane v skupine, najprej najdemo odgovarjajočo skupino in zatem začnemo s primerjavo med spojinami. Učinkovitost postopka je odvisna od velikosti skupin oziroma od izbire parametrov, ki ločijo posamezne skupine. Največji vpliv na čas za registracijo spojine ima uporabljena metoda za zapisovanje spojin. Linearne notacije omogočajo dokaj hitro preiskovanje podatkovnih datotek zaradi enoličnosti, nedvoumnosti *ter kompaktnosti zapisov. Registracija spojine zapisane* v Wisweserjevi notaciji se izvaja v okviru sistema CROSSBOW (7).

V sistemih, kjer je osnovni zapis spojin v obliki tabel povezanosti so nujno potrebne tehnike za generiranje enoličnih in nedvoumnih zapisov iz poljubno podanih tabel povezanosti. Ti zapisi so znani pod imenom "kanonične tabele povezanosti" (8), (9). Problem generiranja kanonične tabele povezanosti se sestoji v izbiri invariantnega oštevilčenja atomov. Problem invariantnega oštevilčenja atomov in zapisa v tabeli povezanosti je ekvivalenten problemu izomorfizma grafov. Molekulske strukturne formule lahko opisujemo kot grafe, ki imajo vozlišča s semantično vsebino. Če sta *enolična in nedvoumna zapisa dveh grafov* G1 in G2 enaka, oziroma še sta njihovi kodi enaki, potem sta G1 in G2 izomorfna in sam postopek kodiranja je izomorfizem (10). Najbolj preprost postopek za zapis spojin v kanonični obliki je generiranje vseh n! možnih tabel povezanosti, oziroma vseh možnih oštevilčenj atomov v molekulskem grafu in leksikografski ureditvi n! tabel. Kanonična oblika tabele bi bila tista, ki bi imela najnižjo leksikografsko ureditev. To način izbire kanonične oblike je izredno zamuden in je primeren le če imamo spojine z zelo majhnim številom atomov. V primeru da je število atomov 20 hipotetičen računalnik, ki lahko generira eno matriko in to matriko primerja z drugo v eni mikrosekundi bi porabil več kot 75000 let (11) *za izpeljavo 20! operacij.*

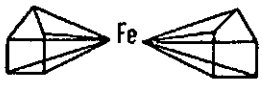
V preteklosti je bilo več poizkusov za izpeljavo (8), (12), (13a, 13b), matematične funkcije, ki bi omogočala hitro identifikacijo izomorfizma grafov. Do danes ni take funkcije, ki bi to opravila v polinomskem času za poljubni graf.

Na splošno problem izomorfizma grafov se nahaja v skupini NP-popolnih problemov. NP-popolni problemi so dobro raziskani problemi imenovani teški problemi. V te probleme štejemo: problem trgovskega potnika s področja operacijskih raziskav, problem tautologije iz propozicijskega računa ter druge podobne kombinatorične probleme. NP-popolni problemi imajo to lastnost da če je en problem iz skupine *NP-popolnih problemov rešljiv z algoritmom, ki ima polinomski* čas, potem so vsi ostali problemi tudi rešljivi v polinomskem času. Seveda za vsak problem je treba dokazati da res pripada skupini NP-popolnih problemov.

Pri reševanju problemov iz skupine NP-popolnih problemov, so zelo pogosto bili uporabljeni hevristični algoritmi, posebej takrat ko je rešitev problema bila povezana z dejansko realizacijo kakšnega sistema za obdelavo podatkov. Tako so Ungar (14) ter drugi avtorji (15), (16), (17) poizkusili zmanjšati časovno kompleksnost algoritmov za ugotavljanje izomorfizma grafov s pomočjo hevrstičnih pravil. Uspeh je bil dosežen le pri načrtovanju algoritmov za ugotavljanje izomorfizma ravninskih grafov (18), (19), (20). Predlagani *algoritmi imajo polinomske čase. Ti algoritmi so neuporabni* za splošne grafe.

Dosedanje izkušnje so pokazale da problem izomorfizma grafov nasplošno, ni mogoče rešiti z dobrim algoritmom, zato se je treba pri konkretnih problemih zadovoljiti s hevrističnimi algoritmi, ki dajejo dobre rešitve v večini primerov. Algoritem te vrste, ki se je v praksi pokazal kot zelo učinkovit, so razvili in implementirali v KIS službe CAS (5). Algoritem za kanonično oštevilčenje atomov, omejuje generiranje vseh možnih tabel povezanosti, tako da predčasno uredi atome in shrani rezultate začetnih poizkusov

oštevilčenja. Algoritem je implementiran na IBM 370/168 v obliki programa za registracijo spojine. Program obdela približno 13000 spojin tedensko, poprečen čas obdelave je 1000 struktur na minuto CPU(5). Zaradi nekatere spojine s simetrijo v strukturi, ki zahtevajo veliko število iteracij, implementacija algoritma predvideva prekinitev obdelave, če je čas porabljen za eno strukturo večji od 3 sec. Med 677000 strukturami obdelanih tekom 1975 leta obdelava 990 je zahtevala več kot 3 sec. Zgled spojine te vrste je ferocene (sl. 2). Za te primere se uporablja posebna tehnika oštevil-



Slika 2.

čenja, ki je znana kot tehnika za sortiranje in registriranje izomerov. Da bi postopek pojasnili, smo v nadaljevanju ponazorili algoritem za hitro generiranje kompaktne kanonične tabele povezanosti. Delovanje algoritma smo ilustrirali z zgledom preproste spojine (slika 3):

Algoritem za generiranje kanonične tabele povezanosti v sistemu službe CAS:

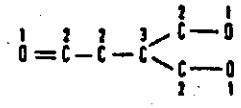
N - množica poljubno oštevilčenih atomov
n ∈ N, n je atom iz N z zaporedno številko n

1. določi vrednost povezanosti l_n vsem elementom iz N: l_n^i je enako številu neogljikovih atomov vezanih na atom n (i za to korak je enako 1),
2. določi spremenljivko k^i, k^i je enako številu različnih vrednosti l_n^i,
3. $i \rightarrow i+1$, določi novo vrednost povezanosti l, $l_n^i = \sum_1^r l_j^{i-1}$ (l_n^i je enako vsoti l vseh r atomov vezanih na atom n)
4. določi spremenljivko k^i,
5. če je $k^i < k^{i+1}$, nadaljuj, drugače pojdi na 7,
6. pojdi na 3,
7. vrednost povezanosti elementov iz N je enaka l_n^i,
8. atom št. 1 (n=1) je atom z največjo vrednostjo l_n^i, atom št. 2,3, ..., itd. so atomi vezani na atom št. 1 z opadajočimi vrednostmi l od 2 naprej. V primeru da dva atoma imata enako vrednost l potem manjšo zaporedno številko dobi atom z nižjo leksikografsko vrednostjo. Če ta razločitev ni možna (enake vrednosti l ter atome istega elementa), potem oštevilči ta par atomov poljubno in označi da so ti atomi poljubno oštevilčeni,
9. neoštevilčene atome vezane na atom št. 2 oštevilči glede naopadajoče vrednosti l. Če srečaš med sosedi atoma št. 2 atome, ki se ne dajo oštevilčiti po danih pravilih, potem te atome oštevilči poljubno, ter označi da so ti atomi poljubno oštevilčeni,
10. oštevilči vse atome iz N z enako proceduro in zgradi tabelo povezanosti,
11. vrni se k atomom, ki je bil poljubno oštevilčen in ki ima največjo zaporedno številko. Če poleg tega atoma ni drugih atomov, ki so bili poljubno oštevilčeni končaj. Generirana matrika je enolična. Končaj postopek. Če to ni res nadaljuj,
12. zamenjaj zaporedne številke atomom, ki so bili poljubno oštevilčeni. Izbriši oznake o poljubnem oštevilčenju,
13. konstruiraj tabelo povezanosti,
14. primerjaj to tabelo s predhodno,
15. če nova tabela povezanosti ima nižjo leksikografsko vrednost od predhodne, potem zamenjaj staro z novo, staro zbriši in pojdi na 11.

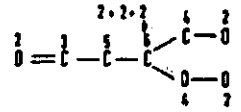
Čas potreben za generiranje kanonične tabele povezanosti je odvisen od število prisotnih atomov v spojini ter od število atomov z enako vrednostjo povezanosti l. Za te atome je treba poizkusiti vse možne kombinacije oštevilčenja, kar vpliva na to da je postopek zamuden. Opisani algoritem ni splošna rešitev problema izomorfizma grafov v okviru KIS, je pa relativno dobra rešitev za grafe, ki imajo najbolj pogosto zasedenost vozlišč od 1 do 4.

3. POIZVEDOVANJE O SPOJINAH, KI VSEBUJEJO DOLOČENI STRUKTURNI FRAGMENT

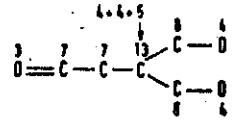
Postopek registracije spojine je oblika iskanja informacij o popolni strukturi spojine, ali z drugimi besedami to je postopek iskanja grafa, ki je izomorfen molekulskemu grafu spojine definirane v vprašanju uporabnika. Druga zelo pomembna zahteva uporabnikov v okviru KIS je iskanje strukturnih fragmentov oziroma iskanje spojin v datoteki sistema, ki vsebujejo strukturni fragment definiran v vprašanju uporabnika. Povedano z besedami iz teorije grafov, to iskanje je ekvivalentno posplošitvi problema izomorfizma grafov v problem izomorfizma podgrafov. Če imamo dva grafa G1=(V1,E1) in G2=(V2,E2), potem G1 je izomorfen podgrafu grafa G2, če in samo če je V1 podmnožica V2 in E1 podmnožica E2. Problem izomorfizma podgrafov se sestoji v določanju ali je graf G1 izomorfen podgrafu grafa G2.



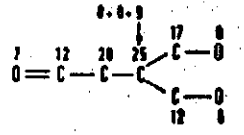
Prvi poizkus
vrednost l = 1,2,3
število različnih k = 3



Drugi poizkus
vrednost l = 2,3,4,5,6
število različnih k = 5



Tretji poizkus
vrednost l = 3,4,7,8,9,13
število različnih k = 6



Četrti poizkus
vrednost l = 7,8,12,17,20,25
število različnih k = 6

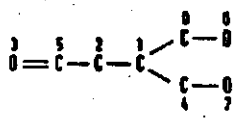


TABELA POVEZANOSTI št. 1

atom št.	1	2	3	4	5	6	7	8
povezave	1	1	1	2	3	4	5	
element	C	C	C	C	C	O	C	O
zveza	S	S	S	S	S	S	S	D

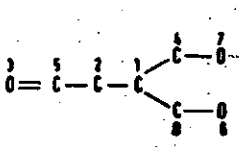


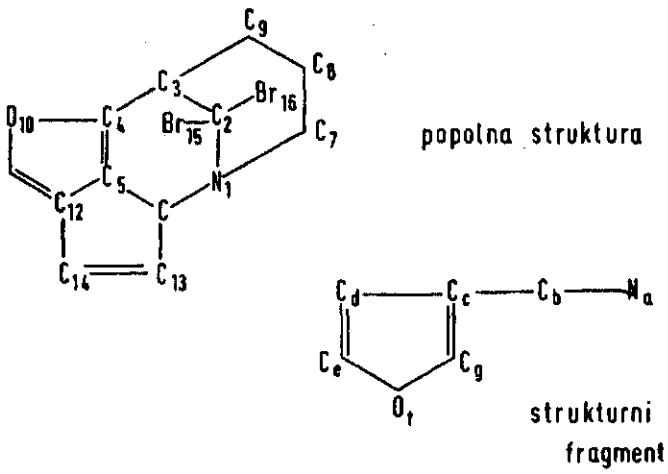
TABELA POVEZANOSTI št. 2

atom št.	1	2	3	4	5	6	7	8
povezave	1	1	1	2	3	4	5	
element	C	C	C	C	C	C	O	O

Slika 3.

Problem ugotavljanja izomorfizma podgrafov je bolj težaven od ugotavljanja izomorfizma grafov. V okviru KIS so ta problem reševali na več različnih načinov. Uspeh posameznih metod je bil odvisen od sistema za zapisovanje in shranjevanje kemijskih spojin ter od zahtevane natančnosti poizvedovanja. Določeni problemi so še naprej nezadostno obdelani in čakajo na boljše rešitve (6). Nekatere prednosti linearnih notacij, kanoničnih tabel povezanosti ter nomenklaturnih zapisov, ki so jih imeli pri registriranju spojin zaradi enoličnosti in nedvoumnosti zapisov, v tem iskanju se popolnoma zgubijo. Tako, pri kanoničnih tabelah se lahko zgodi, da so atomi v podstrukturi oštevilčeni različno od atomov v popolni strukturi ker ti atomi nimajo iste atome za sosede. (V nadaljevanju bomo pod "popolno strukturo" razumeli strukturno formulo spojine zapisane v enem od možnih računalniških zapisov v datotekah KIS in pod podstrukturo, strukturno formulo nepopolne strukture definirane v vprašanju uporabnika). Kot ilustracijo smo na sliki 4 pokazali eno strukturo in odgovarjajočo podstrukturo. Pri nomeklaturnih ali notacijskih zapisih, je podstruktura lahko zapisana s simboli, ki se razlikujejo od simbolov uporabljenih za zapis popolne strukture, spet zaradi različnega okolja v katerem se podstruktura nahaja v različnih spojinah.

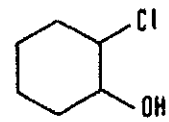
Iskanje po podstrukturah se lahko odvija na več nivojev natančnosti. Najnižji nivo natančnosti je iskanje v datoteki kjer imamo zapis spojin s fragmentacijskimi kodami ali če iščemo s pomočjo mask. Maske (5) se najbolj pogosto uporabljajo pri preiskovanju velikih bank podatkov, in so podobne fragmentacijskim kodam, le da predstavljajo računalniško generirane podatke v katerih je prisotnost oziroma odsotnost kemijskih značilnosti (elemente, povezave, obroče ipd.) označena z enicami ali ničlami, v nekaterih primerih tudi s števili. Za ilustracijo smo na sl. 5 ponazorili en zapis spojine s fragmentacijsko kodo. V primeru sistema s fragmentacijskimi kodami, v vprašanju uporabnika se zahtevani fragment zapiše s kodo, ki je bila uporabljena pri ustvarjanju datoteke. Odgovor vsebuje seznam molekul v katerih se zahtevani fragment nahaja.



Slika 4.

V sistemu GREMAS (21) se datoteka fragmentov generira avtomatsko iz tabele povezanosti ter ostalih topoloških deskriptorjev. Avtomatsko generiranje fragmentacijskih kod omogoča nemoteno spreminjanje uporabljenih kod, oziroma dodavanje novih, brisanje starih ipd. Podobno generiranje datotek s fragmenti je realizirano v KIS National Institute of Health in Environmental Protection Agency v ZDA ali skrajšano NIH-EPA (6). V tem KIS, se za vsako poizvedovanje po podstrukturah generirajo invertirane datoteke iz osnovne datoteke. Pri tem obstajajo dve možnosti: generiranje datoteke kjer so spojine zapisane kot niz fragmentov in generiranje datoteke s podatki o obročih. Datoteka fragmentov, ki poleg fragmentov vsebuje še podatek o registrski številki, se generira z obdelavo vsakega atoma in vsake povezave posamezne spojine. Pri tem se določajo naslednji podatki: dimenzije fragmenta (število atomov v fragmentu), semantiko centralnega atoma (centralni atom je atom, ki povezuje fragment z ostalo strukturo spojine), semantiko prvega soseda (prvi sosed je atom vezan na centralnega, ki se najmanj pogosto sreča v datoteki, ponavadi je različen od C,N,O, če takega atoma ni potem je to C atom z dvojno ali trojno povezavo ipd.), semantiko drugega, tretjega in četrtega soseda.

Koda	Pomen
2/12	obroč
4/1	šestčlanski benzenov obroč
17/1	klor (en atom)
18/1	ena OH skupina



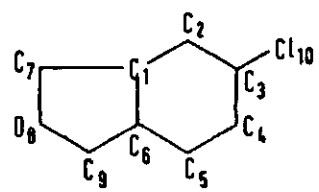
Slika 5.

Generiranje se konča ko je celotna struktura zapisana v obliki fragmentov. Datoteka s podatki o obročih ima enako strukturo, le da tukaj nastopajo podatki o obročih. Za vsakega obročka se generira zgoščena koda (hash koda) iz tabele povezanosti ter podatki o hetero atomih, njihovi poziciji, o substituentih ipd.). Poleg tega se generirajo še podatki o možnih kombinacijah med obroči (2,3,4 skupaj), kjer se ostali obroči obravnavajo kot substituenti. Za ilustracijo smo na sliki 6 pokazali iskanje podstrukture sestavljene iz petčlanskega heteroatomskega obroča in iz šestčlanskega obroča s substituentom pri tretjem atomu. Pogoji iskanja so bili naslednji:

zahteva št. 1: kombinacija obročev v popolni strukturi naj bo enaka (ni dovoljeno več kot dva obroča),
zahteva št. 2: le vozlišče 8 je lahko različno od C.
zahteva št. 3: v šestčlanskem obroču je prisoten najmanj en substituent in naj bo vezan na atomu št. 3.

Pri iskanju je bila uporabljena podatkovna baza NIH-EPA Mass Spectral Search System (22), (23), ki vsebuje okrog 30000 različnih spojin in njihove masne spektre. V prvi invertirani datoteki s podatki o obročih je bilo najdeno 18 spojin, s po enim ali dveh obročkov. V drugi datoteki s podatki o fragmentih je bilo najdeno 180 struktur, ki vsebujejo fragment s substituentom iz slike 6. Presek množic spojin iz

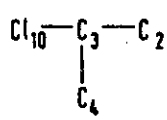
a) definicija podstrukture



večkratne povezave ter H atomi niso določeni

številke označujejo oštevilčenje atomov v tabeli povezanosti

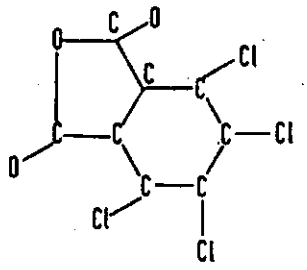
b) definicija zahtevanega fragmenta



Slika 6.

obeh datotek je dalo le eno spojino pokazano na sliki 7. Seveda, večkrat se zgodi, da presek dveh datotek da več kot eno spojino, v tem primeru se uporablja natančnejši način iskanja, znan kot "iskanje atom za atomom" (27).

najdena struktura



molekulska formula : C 8 Cl 4 O 3

registrska številka CAS : 112088

Slika 7.

Pri vseh kompleksnih KIS, se natančno poizvedovanje po podstrukturah izvaja z iskanjem atom za atomom. To iskanje je precej zamudno in zaradi tega se povsod pred tem izvaja predhodno iskanje po fragmentih (tako kot je bilo navedeno v primeru zgoraj), po nomenklaturnih zapisih, ali po linearnih notacijah. V okviru sistema CAS je predhodno iskanje izvedeno s pomočjo nomeklaturnih zapisov. Pri tem so bili uporabljeni najnovejši dosežki s področja preiskovanja tekstov in računalniško čitljivih datotek z obliko teksta (24).

V sistemu z Dyson-IUPAC-ovo notacijo predhodno iskanje se izvaja s pomočjo datotek s permutiranimi indeksi (25). Wiswesserjeva notacija je bila uporabljena za isti namen v sistemu "Institut-a for Scientific Information" (26). Uspeh iskanja je zelo odvisen tudi od uporabnika oziroma od sposobnosti da se pravilno opiše in kodira pričakovano okolje podstrukture. V koliko je to bolj uspešno v toliko je manjše število spojin za natančnejše preiskovanje.

Natančno poizvedovanje o tem ali se določena podstruktura nahaja v nekaterih strukturah je možno le s primerjanjem atomov iz podstrukture z atomi strukture in z primerjanjem povezav iz podstrukture s povezavami iz strukture. Primerjanje je možno le če so strukture zapisane v topološki obliki, oziroma če zapis spojin izhaja iz njihovih grafov. V dosedanji tehnologiji KIS srečamo dva algoritma za primerjavo podstrukture s strukturo implementiranih v najrazličnejših enačicah. To so: iterativna tehnika primerjanja atom z atomom (27) in tehnika postopnega izločanja množic (16). Oba algoritma v svoji prvotni obliki, so bila načrtovana za primerjanje popolnih struktur, oziroma za ugotavljanje izomorfizma dveh grafov. Univerzalnost postapkov je omogočila uporabo tudi pri določanju izomorfizma podgrafov.

3.1 Iterativno iskanje in primerjanje atom z atomom

Iterativno iskanje se sestoji v primerjanju atomov iz podstrukture z atomi iz strukture do popolnega ujemanja ali neujemanja. Da bi skrajšali čas primerjanja in hitreje izločili neprimerne strukture, se primerjanje začne z atomom iz podstrukture, ki se najmanj pogosto sreča v datoteki. Vse strukture, ki ta atom ne vsebujejo odpadejo že na začetku. Po prvi uspešni primerjavi dveh atomov, se iskanje nadaljuje na enak način. Za naslednji atom se vzame atom vezan na predhodnega in ki se najmanj pogosto sreča v datoteki. Če do ujemanja pride, se primerjanje nadaljuje z naslednjim sosedom, ki se izbira po enakih kriterijih. V primeru da pride do neujemanja, postopek se vrne v točki zadnjega ujemanja in se primerjanje nadaljuje z drugim atomom. Prehojena pot primerjanja se sproti zaznamuje zaradi vračanja v primerih neujemanja. Postopek je iterativen in teče do ugotovitve popolnega ujemanja med podstrukturo in kakšnim delom strukture. V primeru da postopek prehodi celotno strukturo in do ujemanja ne pride, postopek konča, kar pomeni da podstruktura ni vsebovana v strukturi.

3.2. Postopno izločanje množic atomov

Postopek postopnega izločanja množic je popolnoma različen od postopka primerjanja atom z atomom in je bolj računalniško pobarvan. Avtor algoritma je Sussenguth (16), posamezne spremembe s ciljem izboljšanja algoritma so predlagali Ming in Tauber (29).

Postopek je izpeljan na podlagi naslednjih trditev:
- če sta grafa G in G* izomorfna, potem podmnožice vozlišč grafa G, ki se razlikujejo med seboj glede na nekatere lastnosti vozlišč, so ekvivalentne podmnožicam grafa G*,
- če podmnožice vozlišč grafov G in G* z enakimi lastnostmi vozlišč nimajo enako število elementov potem G in G* nista izomorfna.

V primeru izomorfizma med grafom G in podgrafom grafa G* so pogoji oslabljeni in glasijo: odgovarjajoča vozlišča grafa G so vsebovana v množice vozlišč grafa G* z enakimi lastnostmi. Zapisana z matematičnim jezikom te dve trditvi dobita naslednjo obliko:

a) graf G je izomorfen grafu G*

vrednost vozlišča
$(x: \text{vrednost}(x)=v) = (x^*: \text{vrednost}(x^*)=v)$
vrednost povezave
$(x: \text{vrednost}((x,y))=b) = (x^*: \text{vrednost}((x^*,y^*))=b)$
valenca
$(x: \text{valenca}(x)=d) = (x^*: \text{valenca}(x^*)=d)$
stopnja zasedenosti
$(x: \text{stopnja}(x)=d) = (x^*: \text{stopnja}(x^*)=d)$
povezanost $\quad A = A^* \rightarrow \nearrow A = \nearrow^* A^*$
$\nearrow x$ - množica vozlišč vezana na vozlišče x

b) graf G je izomorfen podgrafu grafa G*

vrednost vozlišča
$(x: \text{vrednost}(x)=v) \subseteq (x^*: \text{vrednost}(x^*)=v)$
vrednost povezave
$(x: \text{vrednost}((x,y))=b) \subseteq (x^*: \text{vrednost}((x^*,y^{**}))=b)$
valenca
$(x: \text{valenca}(x)=d) \subseteq (x^*: \text{valenca}(x^*)=d)$
stopnja zasedenosti
$(x: \text{stopnja}(x)=d) \subseteq (x^*: \text{stopnja}(x^*)=d)$
povezanost $\quad A \subseteq A^* \rightarrow \nearrow A \subseteq \nearrow^* A^*$

Postopek dela v dveh fazah: generiranje podmnožic in izločevanje podmnožic. Generiranje podmnožic smo ilustrirali na zgledu ponazorjenim na sliki 8. Množice se medseboj razlikujejo po vsebini elementov (C,O,N,Br), vrednost povezave (dvojna,enojna), valence atomov v strukturi brez H atomov (1,2,3), (stopnja zasedenosti), številu povezav v najmanjšem ciklu kateremu vozlišče pripada (povezanost). Izločevanje množic, omogoča zmanjševanje števila vozlišč, ki so možni ekvivalentni vozlišču x. Idealen primer je, če je to število enako 1, potem je $x = x^*$. Če sta Si in Si* par množic z elementi z enakimi lastnostmi in če velja $Si \subseteq Si^*$, potem lahko sklepamo naslednje: če je x v Si, potem je tudi x^* v Si*. Za vse i v katerih je x vsebovan v Si in x^* je vsebovan v Si*, In od tukaj sledi naslednja relacija:

$$x^* \in \bigcap_{i \subset I} Si^* \qquad 1$$

i $\subset$ I, le če je x v Si.
Če sta Sj in Sj* množici z enakim številom vozlišč, Sj = Sj*,

Množice podstrukture		Množice strukture
vrednost vozlišč:	C (b,c,d,g)	(2,3,4,5,6,7,8,9,11,12,13,14
	O (f)	(10)
	N (a)	(1)
	Br ∅	(15,16)
vrednost povezav:	enojna	
	(a,b,c,d,e,f,g)	(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16)
	dvojna	
	(c,d,e,g)	(4,5,7,8,11,12,13,14)
stopnja zasedenosti:	1	
	(a)	(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16)
	2	
	(b,d,c,f,g)	(1,2,3,4,5,6,7,8,9,10,11,12,13,14)
	3	
	(c)	(1,2,3,4,5,6,12)
povezanost:	5	
	(c,d,e,f,g)	(4,5,6,10,11,12,13,14)
	(a,b)	(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16)

Slika 8.

potem velja naslednje $\bar{S}j \subseteq \bar{S}j^*$. Vozlišče, ki ne pripada Si, tj., ki pripada Sj nima ekvivalenta v Sj* (ker je $\bar{S}j = Sj^*$ in je ekvivalenca vozlišč ena proti ena), kar pomeni da so ekvivalenti vozlišč iz $\bar{S}j$ vsebovani v $\bar{S}j^*$. Z drugimi besedami, če je $Sj = Sj^*$ in če $x \notin Sj$, potem $x^* \notin \bar{S}j$. Z uporabo množic, ki imajo to lastnost dobimo naslednjo relacijo:

$$x^* \subseteq \bigcap_{i \in I} Si^* \cap \bigcap_{j \in J} Sj^* \qquad 2$$

$i \in I$, če je $x \in Si$ in $j \in J$, če je $Sj = Sj^*$ in $x \in Sj$. Z operacijo presek iz relacije 2 v veliki meri zmanjšamo število elementov v množicah S in S*. V primeru da za vsako vozlišče iz S najdemo ekvivalent v S*, smo problem rešili in postopek lahko zaključimo. Če se to ne zgodi, potem je treba prvo fazo - generiranje novih parov množic z elementi z enakimi lastnostmi ponoviti in zatem tudi drugo fazo. Če so v prvi iteraciji bili uporabljeni množici, ki so se razlikovali med seboj po vsebini elementov, v drugi iteraciji bodo uporabljeni množici, ki se razlikujejo v vrednosti povezav. Zgled generiranja množic in izvajanja operacije presek je ponazorjen na sliki 9. Postopek iterativno teče do

Množice podstrukture		Množice strukture	
(1,2,4)	S 1	(b,d,e,f)	S* 1
(1,3)	S 2	(a,d)	S* 2
(2,4,5)	S 3	(b,c,d,e,f)	S* 3
(3,4,5)	S 4	(a,b,c,f)	S* 4

Rezultati po operaciji presek

Množice strukture		Množice podstrukture uporabljene množice
(1)	(d)	S ∩ S 1 2
(2)	(b,e,f)	S ∩ S ∩ S 1 2 3
(3)	(a)	S ∩ S 1 4
(4)	(b,f)	S ∩ S ∩ S ∩ S 1 2 3 4
(5)	(b,c,f)	S ∩ S ∩ S 2 3 4

Slika 9.

do izpolnitve enega od naslednjih pogojev:

a) za vse atome iz G so določeni ekvivalenti v G*,
b) podmnožica S vsebuje več elementov od odgovarjajoče množice S* (v tem primeru G ne more biti izomorfen s podgrafom grafa G in izomorfizem ni možen),
c) ni pogojev za generiranje novih podmnožic (vse lastnosti po katerih se atomi v grafu G razlikujejo med seboj so izčrpani in rezultat je nedoločen). To pomeni da za nekatere atome ekvivalent ni določen ali da za nekatere atome ni pokazano da ekvivalenti ne obstajajo.

Nedoločen rezultat dobimo v dveh primerih: med G in G* obstaja večkratni izomorfizem (ni metode, ki bi izbrala enega od izomorfizmov) in uporabljene lastnosti elementov niso zadosti močne, da bi razločile posamezne atome. V tem primeru se ekvivalenti določajo naključno, vsaka naključna izbira je podvržena potrditvi. Če je potrditev negativna potem izberemo naključno nov par atomov in postopek ponovimo dokler ne dobimo pravilno izbiro.

Oba algoritma, primerjanje atom z atomom (11) in tehnika postopnega izločanja množic (16) imata veliko časovno kompleksnost (28), kar pomeni da čas potreben za ugotavljanje izomorfizma narašča izredno hitro s številom atomov v grafu. Poleg tega velik vpliv na časovno kompleksnost ima tudi struktura grafa in način kako so podatki zapisani v računalniku. Časovna kompleksnost algoritma za primerjanje atom z atomom je ocenjena na $O((d-1)^n)$, kjer je d maksimalna valenca atoma v strukturi in n število atomov v strukturi. Bolj uspešni algoritmi za ugotavljanje izomorfizma grafov uporabljajo različne kombinacije obeh algoritmov (10), (14), (30), (31).

4. INTERKONVERZIJA MED ZAPISI

Algoritmi, ki omogočajo avtomatsko interkonverzijo med različnimi zapisi spojin omogočajo realizacijo nekaterih pomembnih funkcij vsakega KIS na najbolj enostaven način. Poleg osrednjega pomena, ki ga algoritmi za interkonverzijo imajo v okviru interne uporabe posameznega KIS, ti algoritmi omogočajo tudi izmenjavo informacij med različnimi podatkovnimi bazami.

Algoritme za interkonverzijo lahko grupiramo v tri skupine:

a) algoritmi za interkonverzijo iz linearnih notacij, nomenklaturnih zapisov, koordinatnih zapisov in zapisov v obliki strukturnih diagramov v tabele povezanosti,
b) algoritmi za interkonverzijo med različnimi tabelami povezanosti,

c) algoritmi za interkonverzijo iz tabel povezanosti v linearne notacije, nomenklaturne zapise, koordinatne zapise in predstavitve v obliki strukturnih diagramov.

V prvi skupini algoritmov so najbolj zanimivi algoritmi, ki omogočajo generiranje zapisov v obliki tabel povezanosti iz strukturnega diagrama predstavljenega na grafičnem zaslonu. Koncepte teh algoritmov zasledimo pri Corey in Wipke-u (32), realizacijo algoritmov za namene KIS pri Feldmanu v okviru NIH-EPA (33), (34), ter v sistemu GREMAS (35), (36). Podobno funkcijo opravljajo algoritmi za generiranje tabel povezanosti iz koordinatnih zapisov kemijskih struktur, ki se v računalniku vnašajo s pomočjo pisalnega stroja za kemijske formule (37). Konverzija linearnih notacij in nomeklaturnih zapisov v tabele povezanosti zasledimo pri vseh večjih KIS. Tako danes obstajajo programi za konverzijo iz nomenklaturnih zapisov (38), Wiswesserjeve notacije (39), Hayward-ove (40), in IUPAC-ove linearne notacije (25) v tabele povezanosti. Fragmentacijske kode ne predstavljajo popoln in enoličen zapis struktur, zato njihova konverzija v druge zapise ni mogoča.

Druga skupina algoritmov se uporablja za generiranje kompaktnih tabel povezanosti iz redundančnih in obratno (41), (42) ter pri konverziji tabel povezanosti iz različnih KIS. Tako danes obstajajo programi za konverzijo tabele povezanosti iz sistema CROSSBOW v tabelo povezanosti sistema CAS (43). Algoritmi za konverzijo iz ene tabele povezanosti v drugo, se uporabljajo tudi pri konverziji zapisov iz ene linearne notacije v drugo. Konverzija zapisa iz linearne notacije v tabelo povezanosti je veliko bolj enostavna od obratne konverzije, zaradi zapletenih sintaksnih in semantičnih pravil notacije (6). Veliko bolj zapletena je konverzija iz ene linearne notacije v drugo. Zato se ta postopek opravlja preko transformacij v tabelah povezanosti in potem iz tabele v linearno notacijo.

Tretja skupina algoritmov realizira konverzijo zapisov iz tabele povezanosti v kakšen drug nedvoumen in enoličen zapis. Tabela povezanosti je najmanj strukturirana predstavitev kemijske spojine in ne vsebuje elemente s kemijsko značilnostjo razen seznama atomov in kemijskih zvez. Obratna konverzija iz bolj strukturiranih predstavitev v obliki tabel povezanosti zahteva le interpretacijo simbolov sintakse. Konverzija tabele povezanosti v kakšno linearno notacijo je možna le s pomočjo kompleksnih algoritmov za analizo strukture in za ustvarjanje linearnega zapisa upoštevajoči zapletena pravila notacije. Eden od bolj dognanih algoritmov za konverzijo tabele povezanosti v Wiswesserjevo notacijo (44), (45), je omogočil razvoj algoritmov za urejanje Wiswesserjevih notacij za kemijske sisteme s kompleksnimi obroči. Podoben temu algoritmu je algoritem Farrella (46) ter algoritmi v okviru sistema DARC (47).

Iz te skupine algoritmov, so zelo zanimivi algoritmi za generiranje strukturnih diagramov iz linearnih notacij. Večina teh algoritmov so bili načrtovani za prikazovanje diagramov preko vrstičnega tiskalnika in tehnične zmogljivosti teh aparatur so omejevale kompleksnost prikazanih diagramov (48), (49). Dognani sistemi za prikazovanje strukturnih diagramov so realizirani v okviru službe CAS (50). Poleg podatkov v obliki tabele povezanosti, sistem za generiranje strukturnega diagramov uporablja datoteko s koordinatnimi podatki o posameznih obročih, ter o strukturah sestavljenih iz več obročev. Podatki za to datoteko so ročno pripravljeni. Datoteka vsebuje več kot 15000 najrazličnejših oblik obročev in sistemov obročev, ki se nahajajo v $3,5 \times 10^{6}$ različnih spojin zapisani v osrednji datoteki sistema službe CAS (CAS Chemical Registry System). Koordinatni zapis obročev iz datoteke obročev omogoča hitro generiranje strukturnega diagrama (izpis je na elektrostatskem risalniku ali na fotostavnim strojem), ker so vsi zapletljaji okrog določanja koordinatov obročev na ta način onemogočeni. V bistvu datoteka s podatki o koordinatih obročev precej poenostavi postopek, čigava osrednja naloga je da ugotovi medsebojno povezavo obročev, verig, linearnih acikličnih nizov s končnim atomom ali brez ter substituentov v obročih. Potem se šele začne sestavljanje koordinatov strukture ali direktno (za aciklični del) ali iz podatkov iz datoteke z obroči. Opisani sistem je bil izpopolnjen s programom, ki direktno iz grafičnega zaslona sprejema strukturne diagrame, generira tabela povezanosti ter strukturni diagram nariše na risalniku ali ga posreduje fotostavnim strojem (51).

5. SKLEPNE BESEDE IN BODOČE USMERITVE

S pregledom algoritmov za obravnavo in obdelavo kemijskih struktur v okviru različnih KIS smo poizkusili oceniti njihovo učinkovitost in uporabnost z ozirom na funkcije, ki jih opravljajo. Lahko rečemo, da je za večino problemov v zvezi z obdelavo računalniško zapisanih kemijskih struktur, najdena učinkovita in praktična rešitev. O tem priča tudi veliko število zelo učinkovitih KIS, ki posredujejo uporabnikom po celem svetu široko paleto najrazličnejših podatkov. Tako kot pri predstavitvi kemijskih struktur (2), tako tudi za algoritme za obravnavo in obdelavo kemijskih informacij lahko ugotovimo da so se prilagajali sistemom s katerim so spojine zapisane, uporabljeni materialni opremi, zahtevam naročnikov ter ostalim posebnosti v zvezi s posedovanjem KIS. Od tukaj tudi izhaja raznovrstnost načrtovanih postopkov ter uporabljenih tehnik. Pri predstavitvi algoritmov smo več prostora uporabili za predstavitev algoritmov bolj splošnega značaja in širše uporabe. To so algoritmi za ugotavljanje izomorfizma grafov in izomorfizma podgrafov v okviru KIS. Za oba algoritma velja da problema ne rešujeta v polinomskem času. Zaradi pomembnosti problemov in potrebi po praktični rešitvi v okviru KIS, so se pri iskanju rešitev zatekli k različnim hevrističnim algoritmom, ali k metodam, ki občutno zmanjšajo število spojin za preiskovanje. Implementirani hevristični algoritmi za ugotavljanje izomorfizma grafov, ki predstavljajo kemijske strukture, so dali zelo dobre rezultate v praksi. Sevedo problemi iz splošne teorije grafov še naprej privlačijo pozornost znanstvenikov, ki si prizadevajo izboljšati časovno kompleksnost omenjenih algoritmov (10).

Algoritmi za obravnavo in obdelavo strukturnih kemijskih informacij, poleg v sistemih za shranjevanje in iskanje informacij se uporabljajo tudi v drugih področjih kemije, kjer se računalnik uporablja kot zelo koristen pripomoček(52). Tako na primer izjemne dosežke zasledimo pri uporabi računalnikov pri iskanju korelacij biološka aktivnost spojine/struktura spojine (53), (54). Podobne analize in uporabo velikih podatkovnih baz, zasledimo pri študiju reaktantov in produktov v sintetskih reakcijah s ciljem izločanja parcialnih struktur značilnih za določene sintetske reakcije (54), (55), ter pri matematski sintezi in analizi struktur molekul in njihovih homologov (56). Algoritme za obdelavo kemijsko-strukturnih informacijah zasledimo tudi pri programih za geometrijsko modeliranje molekul, oziroma za generiranje trodimenzionalnih deskriptorjev kemijskih spojin (57), (58), ter v programih za načrtovanje sintetskih poti (59).

6. REFERENCE:

1. C.M. Bowman, The development of Chemical Information Systems, v knjigi Chemical Information Systems, ed. Ash & Hyde, J.Wiley & Sons, New York, N.Y. (1974)
2. B. Džonova-Jerman-Blažič, Kem.Ind., 28 (1979), 67
3. M.A. Oxman, H. Kissman, J. Burnside, J. Edge, C. Haberman, A. Wyres, J.Chem.Inf.Comp.Sci., 16 (1976) 19
4. A.J. Stuper, W.E. Burgger, P.C. Jurs, Computer-Assisted Studies of Chemical Structures and Biological Functions, J. Wiley & Sons, New York, N.Y. (1979), pogl. 3.
5. L.J. O'Korn, Algorithms in the Computer Handling of Chemical Information, v knjigi Algorithms for Chemical Computation, ed. E. Christoffersen, ACS Symposium Series No. 46, Washington (1977), 122

6. J. Feldman, G.W.A. Milne, S.R. Heller, A. Fein, J.A. Muller, B. Koch, Jour. Chem. Inf. Comp. Sci., 17 (1977) 173
7. D. Schmidt, L. Druffel, Jour. ACM, 23 (1976) 433
8. J. Turner, SIAM & Appl. Math., 16 (1968), 520
9. C. Jochum, J. Gasteiger, Jour. Chem. Inf. Com. Sci., 17 (1977), 2
10. R.E. Tarjan, Graph Algorithms in Chemical Computation, v knjigi Algorithms for Chemical Computation, ed. E. Christoffersen, ACS Symposium Series No. 46, Washington (1977), 1
11. A. Bertziss, J. ACM, 20 (1973), 365
12. M. Karp, Tech. Rep., 3 Comp. Sci. Dep. Univ. of California, (april 1972)
13. a) M. Randić, J.Chem.Inf.Comp.Sci., 15 (1975), 105
    b) M. Randić, J.Chem.Phys., 60 (1974), 3920
14. S. Unger, Comm. ACM, 7 (1964), 26
15. G. Saucier, Rev.Francaise Inform.Rech.Oper., 5 (1971), 39
16. E. Sussenguth Jr., J.Chem.Doc., 5 (1965), 36
17. J. Steen, Jour. Chem.Doc., 3 (1969), 51
18. J. Hopcroft, E. Tarjan, v knjigi Complexity of Computer Computation, ed. R.E. Miller & J.W. Tatcher, Plenum Press, New York, N.Y. (1972), 143
19. J. Hopcroft, J. Wong, Proc. 6th Annual ACM Symp., Seattle, Washington, (april 1974), 172
20. L. Weinberg, IEEE Trans. on Circuit Theory, CT-13, (1966), 142
21. R. Fugman, The IDC System, v knjigi Chemical Information Systems, ed. Ash & Hyde, J.Wiley & Sons, New York, N.Y., (1975)
22. S.R. Heller, H.M. Fales, G.W. Milne, Org. Mass. Spectr., 7 (1973), 107
23. S.R. Heller, G.W. Milne, R.J. Feldman, R.S. Heller, Jour. Chem.Inf.Comp.Sci., 16 (1976), 176
24. W. Fisanick, L.D. Mitchell, J.A. Scott, C.G. Wonder Stouw, Jour.Chem.Inf.Comp.Sci., 15 (1975), 73
25. G.M. Dyson, The Dyson-IUPAC Notation, v knjigi Chemical Information Systems, ed. Ash & Hyde, J. Wiley & Sons, New York, N.Y., 1975
26. C.E. Granito, E. Garfield, Naturwissenschaften, 60 (1973), 189
27. L.C. Ray, R.A. Kirsh, Science, 126 (1957), 814
28. M.O. Robin, Comm. ACM, 20 (1977), 625
29. T.K. Ming, S.J. Tauber, J.Chem.Doc., 11 (1971), 47
30. D.G. Corneil, C.C. Gotlieb, Jour.ACM 17 (1970), 51
31. Y. Shah, G. David, M. McCarthy, Trans.on Systems Man and Cybernetics, SMC-4 (1974), 313
32. E.J. Corey, W.T. Wipke, Science, 166 (1969), 179
33. R.J. Feldman, S.R. Heller, Jour.Chem.Doc., 12 (1973), 48
34. R.J. Feldman, Interactive Graphic Chemical Structure Searching, v knjigi Computer Representation and Manipulation of Chemical Information, ed. Wipke, Heller, Feldman, Hyde, J. Wiley & Sons, New York, N.Y., (1974)
35. E. Meyer, Topological Search of Compounds in Large Files, v knjigi Computer Representation and Manipulation of Chemical Information, ed. Wipke, Heller, Feldman, Hyde, J. Wiley & Sons, New York, N.Y., 1974
36. E. Ziegler, K. Boll, Anal.Chem.Acta, Computer Technique and Optimization, 103 (1978) 237
37. A. Zamora, D.L. Dayton, Jour.Chem.Inf.Comp.Sci., 8 (1976), 74
38. E. Wander Stouw, P.M. Elliot, A.C. Isenberg, Jour. Chem.Doc., 14 (1979), 185
39. E. Hyde, L. Matthews, L.H. Thompson, J.W. Wiswesser, Jour.Chem.Doc., 7 (1967), 200
40. S.J. Tauber, S.J. Fraction, H.W. Hayward, v knjigi Chemical Structures as Information: Representation Transformations and Calculations, Spartan Books, Washington D.C., 1965
41. J.E. Ash, Connection Tables and Their Role in a System, v knjigi Chemical Information Systems, J.Wiley & Sons, New York, N.Y., 1974
42. D. Lefkovitz, Jour.Chem.Doc., 1 (1967), 186
43. L.H. Campey, E. Hyde, H. Jackson, Chem. Br., 6 (1970), 427
44. C. Ebe, T. Tommy, A. Zamora, Jour.Chem.Inf.Comp. Sci., 16 (1976), 36
45. C.M. Bowman, F.A. Lander, N.W. Lee, M.H. Reslock, Jour.Chem.Doc., 8 (1968), 133
46. C.D. Farrell, A.R. Chauvenet, D.A. Conlver, Jour. Chem.Doc., 11 (1971), 52
47. J.E. Dubois, DARC System in Chemistry, v knjigi Computer Representation and Manipulation of Chemical Information, ed. Wipke, Heller, Feldman, Hyde, J. Willey & Sons, New York, N.Y., 1974
48. R. Rogers, M.A.T. "CROSSBOW", Proc. 158th National Meeting of ACS, New York, N.Y., 1969
49. B.L. Zimmerman, Computer Generated Chemical Structural Formulas with Standard Ring Orientation, doktorska disertacija University of Pennsylvania, Philadelphia, 1971
50. P.G. Dittmar, J. Mockus, Jour.Chem.Inf.Comp.Sci., 17 (1977), 186
51. J. Blake, N. Farmer, R. Haines, Jour. Chem. Inf. Comp. Sci., 17 (1977), 223
52. H. Skolnik, Jour.Chem.Inf.Comp.Sci., 17 (1977), 234
53. A. Stuper, W. Brugger, P.C. Jurs, Computer-assisted Studies of Chemical Structures and Biological Function, J. Wiley & Sons, New York, N.Y., 1979, poglavja 5,6,7
54. P.C. Jurs, T.L. Isenhour, Chemical Application of Pattern Recognition, J. Willey & Sons, New York, N.Y., 1975
55. J. Walls, Chemical Reaction Indexing v knjigi Chemical Information Systems, J. Willey & Sons, New York, ed. Ash & Hyde, N.Y., 1975
56. V.V. Serov, M.E. Elyashberg, L.A. Gribov, J. Mol. Str., 31 (1976), 381
57. G.R. Marshall, H.E. Bossard, R.A. Ellis, Computer Handling of Chemical Structures: Application in Crystallography, Conformational Analysis and Drug Design, v knjigi Computer Representation and Manipulation of Chemical Structures, ed Wipke, Heller, Feldman, Hyde, J. Willey & Sons, New York, N.Y., 1975
58. A. Stuper, W. Brugger, P.C. Jurs, Computer-Assisted Studies of Chemical Structures and Biological Function, J. Willey & Sons, New York, N.Y., 1979, poglavje 8
59. M. Bersohn, M. Esack, Chemical Reviews, 76 (1976), 269

USLOVI STABILNOSTI KLIZNOG REŽIMA SISTEMA DRUGOG REDA SA PROMENLJIVOM STRUKTUROM, DISKRETNOM OBRADOM INFORMACIJE I DISKRETNOM POVRATNOM SPREGOM

ČEDOMIR MILOSAVLJEVIĆ

ELEKTRONSKI FAKULTET, NIŠ

UDK: 62 – 50.001.4

U radu su analizirani uslovi stabilnosti kliznog režima sistema drugog reda sa promenljivom strukturom. Proces koji se upravlja je sa astatizmom drugog reda čija je funkcija prenosa K/p^2. Informacija o stanju procesa obradjuje se diskretno. Glavna povratna sprega je takodje diskretizovana. Utvrdjeni su uslovi stabilnosti kliznog režima u funkciji periode odabiranjam pri sinhronom radu odabirača.

CONDITION OF THE SLIDING MODE STABILITY OF THE SECOND ORDER VSS WITH DISCRETE DATA PROCESSING AND DISCRETE FEEDBACK. Condition of the Sliding Mode Stability of the Second Order Variable Structure System (VSS) a analysed. The operated system is with the second order astatism whose transfer function is K/p^2. Both information on the object condition and the main feedback a processed discretely. The conditions of sliding mode stability in terms of the selecting periods (selector operation being synchronous) are established.

1. UVOD

Savremene tendencije razvoja mikroelektronike i primena LSI integrisanih kola u oblasti računske tehnike i elektronike, a posebno primena mikroprocesora, nameće potrebu bržeg i šireg uvodjenja diskretnih regulatora u sistemima automatskog upravljanja tehnološkim procesima.

Uvodjenje diskretne obrade signala dovodi do kvalitativno novih prilaza analizi ponašanja sistema i sintezi regulatora. Od bitnog uticaja na rad sistema je i perioda uzimanja podataka o stanju upravljanog objekta. Poznato je da se pri prenošenju signala diskretnim putem oni mogu u potpunosti reprodukovati ako su ispunjeni uslovi teoreme diskretizacije. Osim toga, kod cifarskih sistema je veoma važno vreme trajanja obrade informacije.

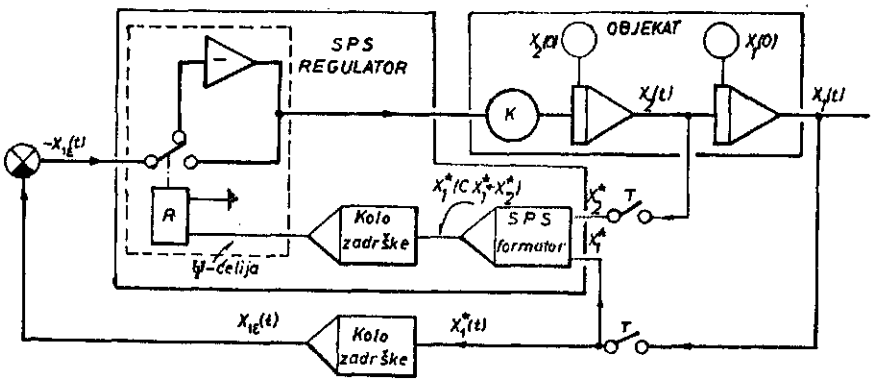
Sistemi sa promenljivom strukturom (SPS), zbog jednostavne konstrukcije i niza novih dobrih osobina, postaju sve atraktivniji za primenu u savremenim sistemima automatskog upravljanja. Za realizaciju SPS u dosadašnjoj praksi korišćeni su elementi analogne tehnike u kontaktnom ili bezkonatktnom izvodjenju. Mada ove realizacije imaju u svom sastavu i logičke elemente ipak su u osnovi rešenja analogna, jer se informacija o stanju upravljanog objekta obradjivala kontinualno. Izvršni element koji ostvaruje komutaciju struktura je relejnog tipa sa dva radna stanja. Ukoliko izvršni element ima kašnjenje ili histerezis u radu SPS dolazi do promena uslova i režima kretanja. Kod SPS postoje tri režima kretanja a u praksi se najčešće koristi klizni režim kao optimalni oblik kretanja. U [1-3] analizirani su uslovi stabilnosti kliznog režima kada izvršni element nije idealan.

Analizom principa rada SPS [1-5] dolazi se do zaključka da je u ovim sistemima upravljanja veoma pogodna primena mikroprocesora za sintezu regulatora. S tim u vezi potrebno je izučiti uticaj diskretne obrade informacije u procesu formiranja zakona komutacije struktura sistema i uslove stabilnosti kliznog režima kao optimalnog oblika kretanja.

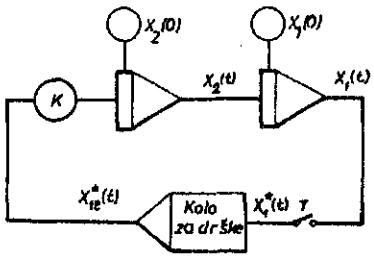
U ovom radu se razmatra sistem drugog reda sa promenljivom strukturom kojim se upravlja astatički objekat funkcije prenosa K/p^2. Osnovna pažnja se posvećuje stabilnosti kliznog režima u funkciji periode diskretizacije. Kašnjenje u izvršnom elementu i u procesu obrade informacije se zanemaruje ali se dobijeni rezultati mogu lako korigovati. Na osnovu rezultata ovoga rada može se vršiti analiza i sinteza sličnih SPS sa diskretnom obradom informacije, a takodje i adekvatna simulacija odgovarajućih analognih SPS na digitalnom računaru. Rezultati dobijeni u radu ilustrovani su na jednom primeru koji je simuliran na računaru.

2. MODEL SISTEMA

Blok šema sistema koji se razmatra prikazana je na sl.1.



Sl.1.



Sl.2

Moguća su tri načina rada odabirača:
- sinhroni i sinfazni rad odabirača,
- asinhroni rad sa istim periodama odabiranja,
- Periode odabiranja su različite.

Prvi i drugi način su interesantniji sa stanovišta praktične realizacije. U ovom radu biće razmotren samo prvi način.

Pretpostavlja se da su koordinate stanja objekta upravljanja dostupne za merenje. Posle odabirača uključena su kola zadrške nultog reda koja pamte izmerenu vrednost promenljive stanja na početku intervala odabiranja u toku cele periode diskretizacije.

Sistem na sl.1. može se opisati sledećim relacijama:

$$\dot{x}_1(t) = x_2(t)$$
$$\dot{x}_2(t) = \Psi K x_{1e}(t) \qquad (1)$$

$$x_{1e}(t) = \begin{cases} x_1(t) & nT < t < (nT+e) \\ x_1(nT+e) & (nT+e) < t < (n+1)T \end{cases} \qquad (2)$$

$e \ll T$, $n = 0,1,2,\ldots$, T- perioda uzimanja uzoraka.

$$\Psi = \begin{cases} 1 & s^* x_1^* \leq 0 \\ -1 & s^* x_1^* \geq 0 \end{cases} \qquad (3)$$

$$s^* = C x_1^* + x_2^* \qquad (4)$$

$C > 0$ - parametar regulatora kojim se može podešavati režim kretanja SPS. Jednačina $s=0$ naziva se klizna prava [1].

Komutacija struktura sistema se ostvaruje na osnovu znaka funkcije

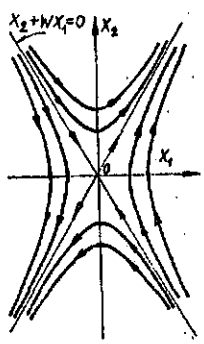
$$P = \operatorname{sgn}\{ x_1^* (C x_1^* + x_2^*) \qquad (5)$$

Radi dalje analize potrebno je najpre definisati strukture sistema, izraze za njihove fazne trajektorije i tip faznih trajektorija.

Prva struktura. Analogni model ove strukture dat je na sl.2. Na osnovu elementarnih transformacija za dati model se mogu izvesti sledeće diferencne jednačine koje opisuju dinamiku prve strukture:

$$x_1((n+1)T) = (1+KT^2/2)\, x_1(nT) + T \cdot x_2(nT)$$
$$x_2((n+1)T) = K \cdot T\, x_1(nT) + x_2(nT) \qquad (6)$$

Leko se može pokazati da sistemu (6) odgovaraju fazne trajektorije tipa sedla (sl.3.).Singularne fazne trajektorije su prave linije



Sl.3.

$$x_2(nT) + W x_1(nT) = 0$$

$n = 0,1,2,\ldots$

Zamenom $n=0$ i $n=1$ dobija se sistem jednačina po W,čije je rešenje koeficijent nagiba singularnih trajektorija

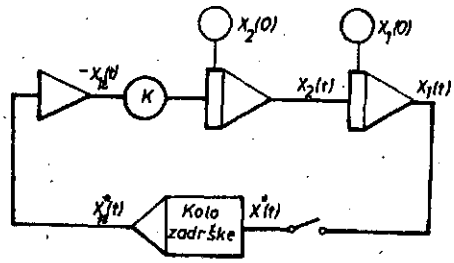
$$W = -\frac{KT}{4}\left(1 \pm \sqrt{1+\frac{16}{K \cdot T^2}}\right) \qquad (7)$$

Neposrednom proverom se može utvrditi da uvodjenje diskretizacije osnovne povratne sprege dovodi do zakretanja faznog portreta razmatrane strukture, u odnosu na odgovarajući portret kontinualnog sistema, za neki ugao u smeru kazaljke na satu.

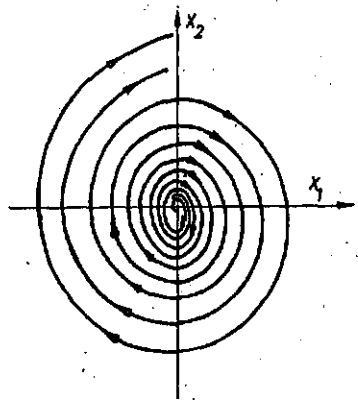
Druga struktura. Analogni model ove strukture prikazan je na sl.4. Fazne trajektorije se opisuju sledećim sistemom diferencnih jednačina:

$$x_1((n+1)T) = (1-KT^2/2)x_1(nT) + T x_2(nT)$$
$$x_2((n+1)T) = -K \cdot T \cdot x_1(nT) + x_2(nT) \qquad (8)$$

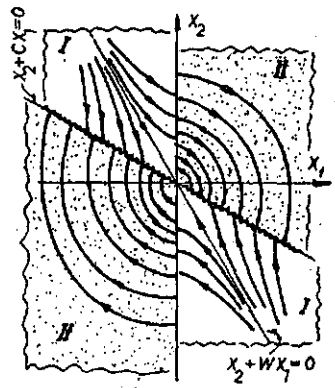
Fazni portret je tipa nestabilnog fokusa (sl.5) za razliku od polaznog kontinualnog sistema koji je u ovoj strukturi imao fazni portr-



Sl.4.



Sl.5.



Sl.6.

et tipa centra.

Očigledno je da se kombinacijom ovih struktura može ostvariti klizni režim u SPS datom na sl.1. Da bi se klizni režim ostvario koeficijent C u (4) mora biti izabran u skladu sa sledećom relacijom

$$C < \frac{KT}{4}\left(1+\sqrt{1+\frac{16}{KT^2}}\right), \qquad (9)$$

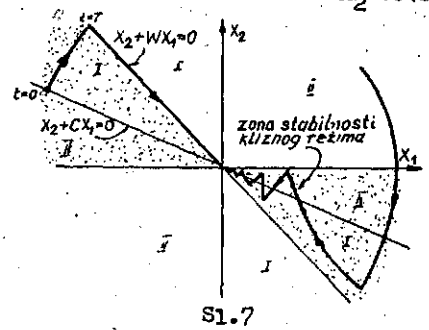
a pri tome se dobija poznati fazni portret SPS kod koga je ostvaren klizni režim (sl.6.).

3. USLOVI STABILNOSTI KLIZNOG REŽIMA

Lako se može pokazati da se u kliznom režimu umesto (6) može koristiti jednostavnija relacija

$$P_1 = \operatorname{sgn}\{Cx_1(nT)+x_2(nT)\} \qquad (10)$$

Zbog diskretnog uzimanja podataka u sistemu se javlja kašnjenje koje je utoliko veće, ukoliko je perioda odabiranja (T) veća. U literaturi [1-3] granice stabilnosti kliznog režima su odredjene stabilnom singularnom trajektorijom prve strukture i osom $x_2=0$.(sl.7.).



Sl.7

Pretpostavimo da se u trenutku t=0, fazna tačka nalazi na kliznoj pravi $s^*=0$ i da je u skladu sa (3) uključena druga struktura. Trajektorija po kojoj će se kretati fazna tačka je opisana diferencnim jednačinama (8). Pošto je sistem sa diskretnom obradom informacije u trenutku $t=0^+$ neće nastupiti promena strukture. Posle vremena $t=T$ vrši se ponovo uzimanje odbiraka stanja objekta, a fazna tačka će se nalaziti u oblasti prve strukture. Ukoliko je perioda odabiramja (T) izabrana tako da se u trenutku t=T fazna tačka nadje na stabilnoj singularnoj trajektoriji prve strukture klizni režim neće nastupiti. Sistem će raditi u režimu kretanja po singularnim trajektorijama. Ako je perioda odabiranja manja od opisane u sistemu će nastupiti klizni režim. Prema tome za granični slučaj možemo napisati:

$$\begin{aligned} x_1(T) &= (1-KT^2/2)\,x_1(0)+T\,x_2(0) \\ x_2(T) &= -KT\,x_1(0)+x_2(0) \\ x_2(0)&+Cx_1(0)=0 \\ x_2(T)&+Wx_1(T)=0 \end{aligned} \qquad (11)$$

Rešavanjem sistema (11) po T i uzimanjem u obzir (7) dobijamo sledeću algebarsku jednačinu

$$T^3+\frac{2}{KC}(C^2-3K)T^2-\frac{14}{K}T-\frac{4(C^2-K)}{K^2C}=0 \qquad (12)$$

čije nam najmanje nenegativno rešenje (realno) daje jednu graničnu vrednost periode odabiranja T_{2gr}.

Ako je pri istim početnim uslovima kao napred umesto druge bila uključena prva struktura, fazna tačka će se kretati po faznim trajektorijama prve strukture. Ako se perioda odabiranja (T) odabere tako da posle vremena t=T fazna trajektorija prve strukture ne preseca osu $x_2=0$ u sistemu će nastupiti klizni režim. Prema tome, uslov nastanka kliznog režima na ovoj etapi kretanja može se dobiti rešenjem sistema jednačina

$$\begin{aligned} x_2(T) &= KT\,x_1(0)+x_2(0) \\ x_2(0) &= -Cx_1(0) \end{aligned} \qquad (13)$$

odakle se dobija

$$T_{1gr}=\frac{C}{K} \qquad (14)$$

Na osnovu izloženog uslov stabilnosti kliznog režima se može napisati u obliku

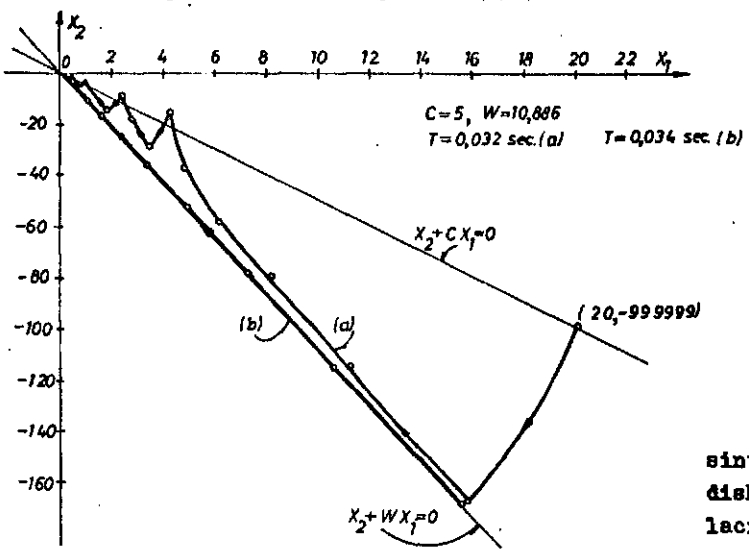
$$T < \min\{T_{1gr}, T_{2gr}\} \qquad (15)$$

4. ILUSTRATIVNI PRIMER

Zadato je K= 100,0 i C=5,0.

Na osnovu (13) nalazimo T_{2gr}= 0,034 sec. a na osnovu (14) T_{1gr}= 0,05 sec. Usvojićemo T≤ 0,034 sec. u skladu sa (15).

Ovaj SPS je modeliran na digitalnom računaru. Rezultati simulacije prikazani su na Sl.8, a algoritam simulacije na sl.9.



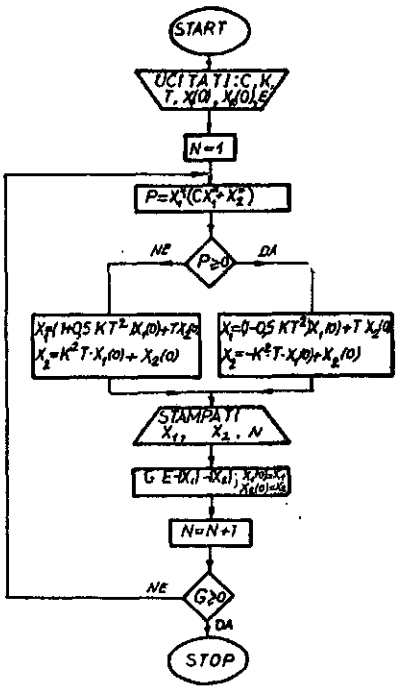
Sl.8.

5. ZAKLJUČAK

Analizirane su mogućnosti primene diskretne obrade informacije kod jednog sistema drugog reda sa promenljivom strukturom koji je namenjen regulaciji objekta čija je funkcija prenosa K/p^2. Pored diskretne obrade informacije u SPS formatoru, osnovna povratna sprega razmatranog sistema je diskretizovana, a odabirači rade sinhrono.

Na osnovu razmatranja faznih trajektorija struktura sistema odredjeni su uslovi izbora periode diskretizacije u funkciji parametara sistema, koja će garantovati nastajanje i stabilnost kliznog režima.

Na jednom primeru koji je simuliran na digitalnom računaru potvrdjeni su izvedeni zaključci.



Sl.9.

Rezultati rada se mogu koristiti kako za sintezu sistema sa promenljivom strukturom sa diskretnom obradom informacije tako i za simulaciju analognih SPS na digitalnom računaru.

LITERATURA

[1] Emel'janov S.V., Sistemy avtomatičeskogo upravlenija s peremennoj strukturoj. "Nauka", Moskva 1968.

[2] Emel'janov S.V. i dr., Teorija sistem s peremennoj strukturoj. "Nauka", Moskva 1970.

[3] Utkin V.I., Skoljzjaščie režimy i ih primenenie v sistemah s peremennoj strukturoj. "Nauka", Moskva 1974.

[4] Taran V.A., Primenenije nelinejnoj korrekcii i peremennoj struktury dlja uluščenija dinamičeskih svojstv sistem avtomatičeskogo regulirovenija (obzor). Avtomatika i telemehanika, tom XXV, No 1 (1964).

[5] Utkin V.I., Variable Structure Systems with Sliding Modes, IEEE Trans. on Autom. Contr., Vol. AC-22, No 2, April (1977).

NOVICE IN ZANIMIVOSTI

VREDNOST SISTEMOV IBM 370

Intel, Motorola in Fujitsu napovedujejo napredovanje v razvoju LSI različice integriranega vezja za izvajanje ukazov sistema IBM 370, kjer naj bi bila zmogljivost tega monolitnega vezja primerljiva s sistemom 370/115. Načrtovanje vezja je zapleteno in počasno, vendar napreduje zanesljivo, kot javljajo iz Silicijske doline (Intel). IBM zatrjuje, da je to nalogo že opravilo!

16-bitni dvoboj med Motorolo in Intelom je povzročil predčasno napoved Intelovega 32-bitnega procesorja, ki naj bi zasenčil M68000 v letu 1981. Ali bo ta sistem v enem vezju že imel lastnosti procesorja 370 in še kaj več?

A.P.Železnikar

POMOČ PRI RAZVOJU PROGRAMSKE OPREME

Hughes Aircraft Company, Culver City, CA 90230 napoveduje pomoč programerjem pri razvoju programske opreme z računalniki. Računalniški pripomoček, ki je bil razvit pri podjetju Hughes, rabi kot risar, knjižničar in pisec poročil v postopku, ki se imenuje načrtovalna sekcija (zasedanje). Ta sistem z imenom AIDES (Automated Interactive Design and Evaluation System) se pogovarja z načrtovalci v angleščini in riše diagrame na slikovnih zaslonih in risalnikih. Sistem analizira dobljene rezultate na preizkusljivost in zvočnost. AIDES znižuje delovno intenzivnost razvijalca pri razvoju programske opreme in hkrati povečuje združljivost, usklajenost in kakovost. Opažanja kažejo, da ta sistem zniža načrtovalni čas za 30% in stroške dokumentiranja za 95%.

A.P.Železnikar

TEHNOLOŠKE PREDNOSTI PODJETJA IBM

Podjetje IBM uporablja v svojih napravah večplastna tiskana vezja z visoko kakovostjo. Pri razvoju vezij in novih sistemov ima na voljo lastno, močno računalniško podporo (CAD), tako da lahko razvija napredne, zapletene sisteme, ki so seveda vselej nad konkurenčnostjo ostalih podjetij. To velja tudi za IBMovo lastno LSI tehnologijo. V vezjih in sistemih so vgrajeni duhoviti in natančni mehanizmi za preizkušanje in diagnosticiranje. Na področju programske opreme proizvaja nove, privlačne in napredne programske pakete. To so le nekateri elementi razvoja in proizvodnje podjetja IBM, ki so zlasti pomembni za konkurenco.

A.P.Železnikar

RAČUNALNIŠKA INDUSTRIJA V ZDA

Porast računalniške industrije v ZDA v letu 1979 je znašal 15,7% glede na leto 1978. Totalni dohodek je bil 45,6 milijarde dolarjev (kratko 45,6Gd) za področje obdelave podatkov (tu so izvzeti drugi dohodki računalniških podjetij). Porast prvih desetih podjetij (največjih) je bil le 13% glede na prejšnje leto. Osem podjetij je priposlovalo dohodek nad 1Gd (glej tabelo 1).

TABELA 1

	dohodek M$,1979	rast %,1979	mesto v 1978
IBM	18 338	7,4	1
Borrougs	2 432	13,5	2
NCR	2 404	24,4	4
Control Data	2 273	21,7	5
Sperry Rand	2 270	15,5	3
DEC	2 032	26,9	6
Honeywell	1 453	12,3	7
Hewlett-Packard	1 030	41,1	8

Zelo veliko rast so dosegla tri mikroračunalniška podjetja, in sicer Apple, Commodore in Tandy, kot kaže tabela 2.

TABELA 2

	preskok mest	mesto v 1979	rast %
Apple	39	61	650
Commodore	19	75	150
Tandy	19	39	131

Tabela 3 kaže procentualno delitev dohodka glede na dejavnosti oziroma posamezne proizvode v celotnem kompleksu računalniške proizvodnje, kot so centralna materialna oprema (CMO), miniračunalniki (mR), periferne naprave in terminali (P&T), usluge in programska oprema (U&SW) ter ostalo (mediji in naprave).

TABELA 3

	1977 Gd	1977 %tot	1979 Gd	1979 %tot	% spr	letni %
CMO	7,387	24	7,185	16	(3)	(1)
mR	2,502	8	4,360	10	74	32
P&T	13,766	44	20,507	45	49	22
U&SW	5,685	18	11,828	25	108	44
ostalo	1,771	6	1,807	4	2	1
tot	31,111	100	45,687	100	47	21

Tabela 3 je poučna tudi za nas, saj kaže razmerja v dohodku glede na posamezne komponente računalniške proizvodnje. Centralna oprema (materialna in operacijska) predstavlja le še 16% od celotnega proizvoda računalniške industrije, usluge in programska oprema pa 25%. Daleč največji dohodek prinaša proizvodnja perifernih naprav in terminalov, in sicer 45%. Letni porast je največji na področju uslug in programske opreme ter na področju miniračunalnikov. To daje misliti tudi domačim proizvajalcem računalniške opreme, saj so možnosti plasmana največje na področju perifernih naprav in terminalov, intenzivno pa se razvijata področji uslug in programske opreme ter miniračunalnikov.

Merilo zdravja določene industrijske veje je njena akumulacija (doseganje določenega čistega dohodka, tj. profita). Na poslovne rezultate v letu 1979 so vplivali tekmovalni in inflacijski pritiski, povečana vrednost kapitala ter večje najemanje namesto nakupa. Razlike v čistem dohodku so bile glede na leto 1978 tudi negativne, npr. -3,2% pri IBM. Ob neupoštevanju podjetja IBM je čisti dohodek narastel za 13%.

TABELA 4: Proizvajalci centralne opreme

mesto v 1979, porast dohodka	podjetje	1979 porast dohodka	1979 porast dohodka v ZDA	1979 dohodek	1979 % totalni dohodek	1979 porast čistega dohodka
3	Cray Research	148,3	-2,1	42,7	100,0	200,0
51	NCR	24,4	24,4	2404,0	80,1	n.p.
56	Control Data	21,7	23,8	2273,0	69,9	n.p.
75	Sperry Univac	15,5	11,4	2270,0	49,5	n.p.
78	Borroughs	13,5	11,2	2432,0	87,3	20,6
83	Honeywell	12,3	17,3	1453,0	34,5	n.p.
89	IBM	7,4	4,9	18338,0	80,2	-3,2
96	Amdahl	-6,6	-14,5	299,6	100,0	-64,5
100	Itel	-54,5	-54,5	221,3	48,1	n.p.

n.p. pomeni "ni podatka"

Tabela 4 kaže razmerja pri proizvodnji centralne opreme. Nekatera podjetja imajo še proizvodnjo, ki ne sodi v področje obdelave podatkov, zato je njihov totalni dohodek sestavljen iz OP in ostalega dohodka (stolpec 6 v tabeli 4).

Pojasnimo kratko še podatke v tabeli 4. Stolpec 1 kaže mesto podjetja glede na porast dohodka v razmerju 1979/1978, kjer je procentualni porast naveden v stolpcu 3. Ta porast se seveda nanaša samo na vrednost proizvodnje centralne opreme (materialne in operacijske). V stolpcu 4 je naveden porast, ki je bil realiziran na območju ZDA. V stolpcu 5 je naveden celotni dohodek za področje proizvodnje sredstev za obdelavo podatkov (druge aktivnosti podjetij so izključene) v milijonih dolarjev. Zadnji stolpec prikazuje porast čistega oziroma nečistega dohodka (negativni porast).

V tabeli 5 so zbrani podatki uspešnosti prvih 13 podjetij v proizvodnji miniračunalnikov. Pomen stolpcev v tabeli 5 je enak pomenu stolpcev tabele 4. Nekatera manj znana in manjša podjetja hitro napredujejo, dočim se večji proizvajalci nahajajo v sredini lestvice. Tabela 5 pojasnjuje sama sebe.

Oglejmo si ločeno še proizvajalce periferne im terminalne opreme. Proizvajalci periferne opreme predstavljajo 4 milijardni segment (v dolarjih) ali 9% celotnega dohodka računalniške industrije. Po produktu je na čelu te skupine Memorex s 664 milijoni dolarjev, kot kaže tudi razpredelnica 6. V tej skupini ni tako izrazite dominacije nekaterih podjetij, kot smo jo imeli v skupinah za proizvodnjo centralne opreme in miniračunalnikov, saj zbere prvih pet podjetij le 54% celote. Periferija je namreč zelo raznolika, kot so tiskalniki, diskovni pogoni, tračne enote in naprave za vstop podatkov. Proizvajalci tiskalnikov, kot so Dataproducts, Centronics in Printronix imajo izdelke za miniračunalniško tržišče, kjer gre za prodajo na osnovi OEM (Original Equipment Manufacturing), medtem ko je npr. Documation prodala svoje tiskalnike končnim uporabnikom. Razmerje rasti tega sektorja je znašalo 23% ter so ga zniževali nekateri proizvajalci pomnilnikov, diskovnih in tračnih enot (Memorex, Telex). Dataproducts je imela težave z zagonom novih proizvodnih linij, ostali proizvajalci tiskalnikov pa so se kar dobro odrezali (Printronix 92%, Documetion 66% in Centronics 35%).

TABELA 5: Proizvajalci miniračunalnikov

mesto v 1979, porast dohodka	podjetje	1979 porast dohodka	1979 porast dohodka v ZDA	1979 dohodek	1979 % totalni dohodek	1979 porast čistega dohodka
6	Tandem Computers	115,6	144,9	66,4	100,0	117,9
9	AM International	98,6	98,7	69,5	8,5	n.p.
15	Prime Computer	63,5	63,9	152,9	100,0	101,0
20	Wang Labs.	53,8	58,4	280,0	68,3	n.p.
28	Hewlett-Packard	41,1	38,5	1030,0	41,5	n.p.
35	Texas Instruments	32,8	34,6	425,0	13,2	n.p.
38	Data General	31,7	29,4	539,6	100,0	14,1
43	Management Assistance	28,0	17,4	283,4	100,0	-4,8
47	Digital Equipment	26,9	26,9	2031,6	100,0	35,3
48	Qantel	26,1	20,8	58,0	100,0	150,0
52	Nixdorf Computer	23,5	23,5	100,0	100,0	n.k.
64	Pertec Computer	19,7	22,6	171,0	100,0	-38,9
70	Harris	16,7	15,4	210,0	20,0	n.k.

TABELA 6: Proizvajalci periferije

mesto v 1979, porast dohodka	podjetje	1979 porast dohodka	1979 porast dohodka v ZDA	1979 dohodek	1979 % totalni dohodek	1979 porast čistega dohodka
10	Printronix	91,8	79,7	32,8	100,0	81,3
14	Documation	66,1	66,4	84,9	100,0	-59,3
17	Storage Technology	59,6	53,0	479,5	100,0	48,1
18	Xerox	57,8	49,6	475,0	6,8	n.p.
31	Centronics	35,4	29,6	129,2	100,0	32,6
49	ITT	25,0	25,0	260,0	1,2	n.p.
57	General Instrument	21,6	17,2	141,0	21,1	n.p.
65	Telex	19,0	21,3	104,4	65,0	n.p.
66	TRW	18,9	27,8	440,0	9,6	n.p.
67	Ampex	17,7	11,4	156,3	35,4	n.p.
68	MSI Data	17,1	13,0	46,5	99,8	-45,9
73	Memorex	16,5	8,9	664,0	90,0	-37,3
80	Recognition Equip.	13,3	12,8	98,2	100,0	-44,7
82	Conrac	12,7	7,9	62,0	45,2	n.p.
86	3M	10,7	11,6	310,0	5,7	n.p.
88	Northern Telecom	9,1	9,1	300,0	100,0	n.p.
90	Dataproducts	6,6	1,6	149,5	86,6	-38,9

Proizvajalci terminalov in terminalskih podsistemov so proizvedli za 1,5 milijarde dolarjev ali 4% celote in so izlistani v tabeli 7. Nekateri teh proizvajalcev izdelujejo naprave za porazdeljeno obdelavo podatkov, drugi pa CRT prikazovalnike, terminalne sisteme za finančno industrijo ter za računalniško podprto oblikovanje in proizvodnjo. Proizvajalci terminalov so imeli odlično letino z velikimi porasti, in sicer Computervision s 112%, Gerber Scientific s 40% (CAD/CAM) in Datapoint s 45% ter Four-Phase z 32% na področju registerskih blagajn v porazdeljenem procesiranju. Najmanjši porast so dosegli CRT proizvajalci in izdelovalci terminalov za finančno industrijo.

Največjo skupino glede na število proizvajalcev predstavlja izdelava programske opreme in uslug, kjer je 30 podjetij doseglo 8% totalnega dohodka (glej tabelo 8). Tu imamo mešanico dejavnosti, kot so daljinske računalniške usluge, razdeljevanje časa, dostop v podatkovne baze, operacijska sistemska programska oprema in aplikativno programiranje. Pet največjih proizvajalcev je s 47% pobralo skoraj polovico od celotne realizacije 3,7 milijard dolarjev te skupine. Kar 86% dohodka je bilo narejenega v ZDA, kjer se uslužnostni posli razvijajo hitreje kot v Evropi. V letu 1979 je bil porast tega področja proizvodnje 26% in zelo je narasel dohodek prvih treh v tabeli 8.

Poslednja dva sektorja, ki si ju oglejmo v tabeli 9, sta proizvodnja osebnih računalnikov in računalniških medijev. V tej skupini najdemo 4 podjetja za diskovne/tračne medije in 3 podjetja za osebne računalnike, ki predstavljajo skupno 1% celotnega dohodka računalniške industrije. Tandy je edini proizvajalec, ki se uvršča med prvih 50 največjih podjetij. Vendar je proizvodnja osebnih računalnikov predstavljala najhitreje razvijajočo proizvodnjo v letu 1979. Podjetja Apple Computer, Comodore International in Tandy so naredila glede na prejšnje leto velike skoke, kot kaže tabela 9. To velja deloma tudi za proizvajalce računalniških medijev, kot je razvidno iz tabele.

TABELA 7: Proizvajalci terminalov

mesto v 1979, porast dohodka	podjetje	1979 porast dohodka	1979 porast dohodka v ZDA	1979 dohodek	1979 % totalni dohodek	1979 porast čistega dohodka
7	Computervision	11,5	130,6	103,0	78,3	n.p.
13	Data Terminal Sy.	80,6	74,6	36,3	37,7	n.p.
23	Lear Siegler	44,4	36,1	65,0	4,8	n.p.
27	Datapoint	41,1	45,9	252,3	100,0	54,5
29	Gerber Scientific	39,0	21,8	36,3	64,5	n.p.
33	Tektronix	33,9	32,4	195,0	22,2	n.p.
37	Four-Phase Systems	31,8	26,3	178,7	100,0	36,9
41	Raytheon	28,0	28,9	180,0	4,3	n.p.
59	Mohawk Data Sci.	21,1	26,2	198,2	100,0	69,4
61	Quotron Systems	20,3	20,3	47,4	100,0	55,6
63	Hazeltine	19,8	19,9	73,9	59,9	n.p.
69	Teletype	16,9	16,9	145,0	44,3	n.p.
71	Olivetti	16,7	16,7	35,0	20,6	n.p.
77	Bunker Ramo	14,5	10,9	135,9	31,9	n.p.
87	Applied Digital Data	10,7	10,7	51,9	100,0	-56,1

TABELA 8: Proizvajalci programske opreme in uslug

mesto v 1979, porast dohodka	podjetje	1979 porast dohodka	1979 porast dohodka v ZDA	1979 dohodek	1979 % totalni dohodek	1979 porast čistega dohodka
8	Comshare	99,7	11,9	66,9	100,0	35,3
11	Anacomp	85,4	89,3	35,8	71,2	n.p.
12	Dun & Bradstreet	83,3	52,5	110,0	11,6	n.p.
21	American Mgmt Sy.	51,3	51,1	48,1	100,0	21,4
22	Interactive Data	50,0	49,8	42,0	100,0	n.p.
24	Martin Marietta	43,1	42,9	67,1	3,3	n.p.
25	Reynolds & Reynolds	43,0	41,5	109,5	57,9	n.p.
26	Manufacturing Data	42,6	35,3	45,5	100,0	32,4
30	Mgmt Science America	36,7	22,2	35,0	100,0	60,0
34	United Telecom	33,8	34,4	138,4	7,7	n.p.
36	Electronic Data Sy.	31,9	31,0	311,5	95,9	18,3
39	Shared Medical Sy.	31,2	31,2	82,8	100,0	25,6
40	Computer Sciences	28,7	22,1	416,1	100,0	19,7
42	The Sun Company	28,0	28,0	64,0	0,6	n.p.
44	Commerce Clearing H.	27,7	27,7	49,3	23,3	n.p.
45	Tymshare	27,5	27,5	176,0	91,1	37,7
46	General Electric	27,3	14,6	350,0	1,5	n.p.
50	First Data Resources	25,0	25,0	50,0	100,0	n.p.
53	National Data	22,8	24,1	54,4	100,0	41,7
54	Automatic Data Proc.	22,5	22,5	400,8	98,0	20,1
55	McDonell Douglas	22,2	17,7	253,0	4,8	n.p.
58	Informatics	21,5	14,9	112,4	100,0	106,7
60	Optimum Systems	20,3	20,3	35,3	100,0	n.p.
62	Boeing	20,0	0,0	96,0	1,2	n.p.
72	Grumman	16,7	16,7	35,0	2,3	n.p.
74	Cincom Systems	15,7	15,7	30,9	100,0	-43,5
81	Wyly	12,8	14,1	89,0	100,0	80,8
93	System Development	2,3	2,3	163,1	100,0	210,0
95	Bradford National	1,2	1,2	120,1	100,0	16,3
97	Planning Research	-8,6	15,8	119,6	48,1	n.p.

TABELA 9: Proizvajalci osebnih računalnikov in računalniških medijev

mesto v 1979, porast dohodka	podjetje	1979 porast dohodka	1979 porast dohodka v ZDA	1979 dohodek	1979 % totalni dohodek	1979 porast čistega dohodka
1	Apple Computer	650,7	650,7	75,0	100,0	n.p.
2	Commodore Internat.	150,0	92,3	55,0	55,4	n.p.
4	Tandy	130,8	125,8	150,0	11,6	n.p.
5	Dysan	118,3	118,3	38,2	100,0	-12,5
16	Verbatim	63,0	55,0	45,0	100,0	-9,1
19	Nashua	56,1	69,1	39,5	6,5	n.p.
32	BASF Systems	35,0	35,0	54,0	n.p.	n.p.

TABELA 10: Porazdelitev dohodka glede na proizvode

podjetje	dohodek (milijoni dolarjev)	procenti (%)				
		CPE	mikro in mini	periferija terminali	progr.oprema usluge	ostalo
IBM	18 338	26	8	44	18	4
Borroughs	2 432	20	8	42	27	3
NCR	2 404	10	20	40	28	2
CDC	2 273	18	0	32	42	8
Sperry Rand	2 270	25	5	44	24	2
DEC	2 032	5	20	49	24	2
Honeywell	1 453	21	5	48	24	2
Hewlett-Packard	1 030	0	24	50	26	0
Memorex	664	0	0	53	10	37
Data General	540	0	28	56	14	2
povprečje	3 344	12,5	11,8	45,8	21,9	6,4

Dohodek računalniške industrije smo upoštevali samo za računalniške proizvode in usluge v letu 1979. Pri tem so bili eksplicitno izključeni iz ocene: komunikacijske naprave, kot so modemi, multiplekserji in posredovalne naprave, navadne komunikacijske usluge, samostojne naprave brez povezave s sistemi za obdelavo podatkov, kot so uradniške naprave, elektronski in magnetni pisalni stroji, elektronske registrske blagajne, nadalje instrumentacija ter druge dodatne naprave za obdelavo podatkov.

V tabeli 10 je za 10 največjih podjetij prikazana porazdelitev posameznih dejavnosti, kjer pomeni CPE velike centralne procesne enote, ostali stolpci pa pojasnjujejo sami sebe. V poslednji vrstici so zbrana tudi povprečja. Kot je razvidno iz tabele, je bila dejavnost prvih desetih največja na področju periferije in terminalov, nato pa pri programski opremi in uslugah, pri CPE, mikro in mini. Povprečje prvih deset predstavlja dohodek 3344 milijonov dolarjev za področje obdelave podatkov.

A.P.Železnikar

Novi operacijski sistem UNIX

Podjetje Microsoft je najavilo novo različico operacijskega sistema UNIX, ki bo napisan v visokem jeziku C, bo prenosljiv, izvedljiv na sistemu s poljubnim procesorjem z vso drugo programsko opremo. Trenutne različice so prirejene za procesorje 8086, 68000 in Z8000. Kot je znano, je bil operacijski sistem razvit v Bellovih laboratorijih ter predstavlja enega najboljših standardnih operacijskih sistemov za 16-bitne mikroprocesorje, ki imajo ustrezne ukaze in dovolj velike naslovne obsege. Standardna Bellova licenca stane 28000 dolarjev, kar je seveda predrago. Cena Microsoftove različice naj bi znašala samo še desetino Bellove licence, v prodajo pa bo dana še letos. Toda Microsoft bo imel močno konkurenco v vrsti drugih sistemov UNIX, ki so v razvoju pri drugih podjetjih.

A.P.Železnikar

Paralelno procesiranje v superračunalnikih

V ZDA se je v poslednjem času pojavilo nekaj proizvajalcev superračunalnikov. HEP (Heterogeneous Element Processor) podjetja Deneclor, je paralelni procesor, ki lahko oblikuje 1 do 16 izvajalnih procesnih modulov, tj. centralnih procesnih enot in 128 bank podatkov s po 1 milijon besedami. Taka enota lahko izvaja do 160 milijonov ukazov na sekundo nad 64-bitnimi besedami, polna konfiguracija pa lahko ima hitri pomnilnik z 1 milijardo zlogov. Visoka pretočnost procesorja se doseže z nenavadnim večukaznim tokom in z večtokovno podatkovno arhitekturo.

Takšno paralelno procesiranje dopušča, da se hkrati izvaja več programov, kjer deluje vsak program nad svojo zalogo podatkov iz skupnega (deljenega) pomnilnika. Večina (drugih) superračunalnikov uporablja vektorski pristop, ko procesira en sam ukazni tok v trenutku, toda uporablja ta ukazni tok hkrati na več podatkovnih tokovih. Deneclor je razvil svojo novo arhitekturo v šestih letih s pomočjo ameriške vlade. Toda konkurenca na področju superračunalnikov je močna. Podjetje Cray Research je razvilo svoj Cray-1, ki opravi 80 do 100 milijonov operacij s plavajočo vejico v sekundi in se prodaja za ceno 4,5 do 16 milijonov dolarjev. Poročali smo že, da je CDC najavilo svoj Cyber 205 z 800 milijoni operacij s plavajočo vejico v sekundi, cena tega sistema pa je 7,9 do 16,5 milijonov dolarjev. Borroughs, izdelovalec znanega superračunalnika Illiac IV, pripravlja različico svojega prvega znanstvenega procesorja s 50 milijoni operacij s pomično vejico na sekundo.

Izvajalni moduli HEP so zgrajeni iz standardnih delov serije 10000, so na tiskanih karticah 35 krat 46 cm, vsak modul pa je sestavljen iz 100 do 150 kartic. Vsak modul ima 2048 splošnih registrov, 4096 registrov za konstante in milijon besed posebnega programskega pomnilnika. Procesni moduli so povezani s pomnilniškimi moduli prek digitalnih preklopnikov, uporablja pa se paketni prenos podatkov. Operacijski sistem se nahaja v vsakem procesorskem modulu in sistemski viri se prirejajo različnim procesom. Programiranje poteka v zbirnem jeziku ali v Fortranu. Prvi sistem za 2,4 milijona dolarjev je bil narejen za vojaški balistični laboratorij. HEP sistemi za široko uporabo bodo imeli en sam procesor z 10 milijoni ukazov na sekundo, z enim pomnilniškim modulom, preklopnikom in z nekaj perifernimi napravami. Cena te konfiguracije bo 1,5 milijona dolarjev.

A.P.Železnikar

S/38 se je končno pojavil

Po enoletni zamudi je IBM končno začel dobavljati sistem/38, ki ga je uvedel že v novembru 1978. Ta mali računalnik ima novo računalniško arhitekturo z zelo zmogljivim novim operacijskim sistemom, ki zagotavlja učinkovito upravljanje podatkovnih baz, ki zaenkrat nima enakovredne konkurence. Do zamude je prišlo predvsem zaradi zapletenosti nove programske opreme. Do zloma sistema je prišlo zaradi previsokega števila podatkovnih izmenjav med virtualnim pomnilnikom z 281 trilijoni zlogov in perifernimi pomnilniki. Ti problemi so bili uspešno rešeni do januarja letos, ko so izboljšali tudi paketne operacije interaktivnega, operacijsko usmerjenega operacijskega sistema. To je bilo zlasti kritično, ko so uporabniki paketno usmerjenega sistema/3 želeli preiti na nove sisteme. Medtem so očistili tudi povezave med posameznimi programskimi moduli in jih preizkusili.V začetku julija je bil sistem že v zadovoljivem stanju, tako da so lahko začeli z dobavami. Trije drugi uporabniki pa so preizkušali sistem/38 že od februarja dalje, ko so lahko ugotovili, da sistem deluje pravilno.

A.P.Železnikar

Pohod Pascala na Japonskem

Jezik Pascal se bo v kratkem pojavil v mikroračunalnikih 13 japonskih proizvajalcev, prevajalniki pa bodo izdelani po licenci Softech Microsystems iz San Diega, Ca. Ta podjetja bodo smela podeljevati podlicence tudi drugim. Pascal naj bi v naslednjih letih v celoti zamenjal Basic. 12 pascalskih licenc imajo: AI Electronics, Fujitsu, Hitachi, Matsushita, Nikko Telecommunication, Nippon Electric, Oki Electric, Sanyo Electric, Sharp, Sony in Sord Computer Systems. Toshiba je glavni zastopnik.

A.P.Železnikar

16 BITNI MIKROPROCESOR TVRDKE NATIONAL SEMICONDUCT.

Tvrdka National Semiconductors bo pričela z dobavo vzorcev 16 bitnih mikroprocesorjev.To so 16008,16 bitni mikroprocesor z 8 bitnim V/I delom;16032,16 bitni mikroprocesor s sposobnostjo naslavljanja pomnilnika s 24 biti (8Mbytov). R.Č.

NOVI IBM MIKRORAČUNALNIK

Tvrdka IBM ne počiva,tudi ona se je vključila v razvoj mikroračunalnikov.Najavila je novi namizni računalnik model 5120,ki ga prodaja za okoli 13 000 $.Računalnik ima 16K bitov programljivega pomnilnika-BASIC in APL v ROMu.Toda Electronics Magazine-Mc Graw Hill publikacija- je pred kratkim objavil prodajo IBM računalnika za $ 4500.Ime modela IBM 5105 je objavilo podjetje Creative Strategies.Model 5105 bodo proizvajali na Japonskem,konstruiran pa bo za priključitev na vodilo S-100.Računalnik bo imel poleg ostalih funkcij,še 16K programljivega pomnilnika,biter čitalnik magnetnega traku na kaseti za masovni pomnilnik in majhen termični printer.Ravno tako napovedujejo model 5130-večterminalno verzijo 5105 računalnika.

R.Č.

INTERAKTIVNI JEZIK ZA Z80

Rosseta Inc.,podjetje iz Hustona,Texas že vrsto let dela na razvoju interaktivnega jezika z imenom "Rosseta Small Talk ".Pobudo za izdelavo jezika je dala tvrdka Xerox,ki pa ni lastnik " Small Talk " jezika.Jezik razširja možnosti uporabe Z80 sistemov.Za razvojne potrebe dobavlja Rosseta Inc. prototipno verzijo jezika nekaterim izbranim uporabnikom Z80 sistemov.

R.Č.

ZENITH PROIZVAJA RAČUNALNIKE ZA DOMAČO UPORABO

Zenith radio korporacija je ena od največjih proizvajalcev TV opreme v ZDA.Sedaj se je orientirala na tržišče računalnikov za domačo uprabo.Dejansko je Zenith vstopil na to tržišče že prejšne leto z pridobitvijo Heatha in formacijo Zenith Data sistemov.Planira proizvodnjo računalnika za domačo uporabo,s ceno izpod $ 1000,na svoji liniji za proizvodnjo barvnih televizorjev.Računalnik se bo dopolnjeval z Radio Sheck TRS-80 in ostalimi sistemi.

R.Č.

ZILOG POVEČAL NABOR INSTRUKCIJ ZA Z 8000

Zilog je najavil dve novi inačici procesorja Z8000,imenovani Z8001 in Z8002.Obe verziji sta namenjeni delovanju z razširljivimi procesnimi enotami(Extended Processing Units). To so vezja,ki razširjajo nabor instrukcij procesorja Z8000.Sistemu se lahko doda eden ali več EPU,ki uporablja predhodno nedefinirane OP kode za pridobivanje aritmetike s plavaločo vejico,iskanje baznih podatkov,testnih operacij, mrežnih vmesnikov in operacij za podporo grafike.Ideja je zasnovana na osnovi Intel-ovega 8087 matematičnega procesorja za I 8086.Standardni Z8000 ne deluje z EPU,pri verzijah Z8001/2 pa je dodanih šest instrukcij za delovanje z EPU.

R.Č.

NEODVISEN STROJNI JEZIK ZA 8 IN 16 BITNE MIKRORAČUNALNIKE

Tvrdka System Consultants Inc.iz San Diega je predstavila prvi univerzalni visoko nivojni prevajalni jezik-PLMX za mikroračunalniške sisteme.Prevajalnik vsebuje knjižnico prevedenih programov,V/I vmesnik in generator koda.Zgradba PLMX je identična intelovemu PL/M. Trenutno so na voljo verzije za TEXDOS(Textronix)in CP/M operacijski sistem.Kod se lahko generira za 8080,8085,8086,Z80,6800,TMS 9900,CDP 1802 procesorje.

R.Č.

TANDY POSTAL DOBAVITELJ GIBKIH DISKOV

Tandy korporacija je podpisala pogodbo z Data Point podjetjem za proizvodnjo 8" in 5" diskovnih pogonov.Radio Shack trenutno uporablja diskovne enote od tvrdke Shugart,CDC in Tandem Magnetics.(Ustanovitelj Tandy korporacije je Radio Shacke).

R.Č.

Kaj smo videli novega na sejmu Elektronika 1980 v Ljubljani

Od 6 do 10 oktobra je bil na Gospodarskem razstavišču v Ljubljani sejem Sodobna elektronika. Domači in tuji proizvajalci elektronske opreme so razstavili svoje izdelke s področja telekomunikacij, elektroakustike merilne opreme, krmilne opreme, opreme za radio in televizijo ipd.

Obiskali smo raztavne prostore domačih in tujih proizvajalcev računalniške opreme ter se pozanimali za najnovejše produkte na tem področju. Verjetno je največje zanimanje med obiskovalci sejma vzbudil robot GORO - 1, ki je bil raztavljen v paviljonu Jurček. Robot GORO - 1 je prvi slovenski inteligentni robot, ki zna nanašati in brizgati emajl in lak za določene predmete in je plod sodelovanja instituta Jožef Štefan iz Ljubljane, instituta Mihailo Pupin iz Beograda in Gorenja iz Velenja, ki je tako prvi jugoslovanski proizvajalec robotov.

Robot GORO 1 je namenjen predvsem emajliranju in lakiranju, zato je na njegovem vrhu nameščena brizgalna pištola. S svojo konfiguracijo lahko doseže vse pozicije in orientacije v delovnem območju.

Delovanje robota GORO 1 temelji na treh režimih. Izbiro omogoča posebno pretikalo.

Učenje :
Robot pomni časovne poteke pozicij brizgalne pištole, ki so potrebne za obdelavo danega predmeta. Zaradi poenostavitve učenja je to izvedeno tako, da operater, potem ko nastavi kodo (ime) predmeta, robotu z ročnim vodenjem časovno in prostorsko demonstrira potrebne gibe.

Delo :
Robot prepoznava posamezne objekte in jih po naučenem postopku primerno obdela. Robot 1 deluje popolnoma samodejno, samostojno, inteligentno. Oblike predmetov, hitrost gibanja linije, na kateri so ti obešeni, vrstni red obdelave, razmaki med predmeti in podobno so zanj nepomembni in enostavno rešljivi problemi.

Čakanje :
Robot omogoča popravljanje, brisanje, kopiranje, dopolnjevanje programov. Izkušen operater lahko z ustrezno instrumentalno opremo sega v same sistemske programe in s tem po želji spreminja robotove lastnosti.

Robot GORO 1 sestavljajo tri med seboj povezane večje enote :

Mehanska enota - skelet :
Izdelana je iz najboljših jeklenih litin in drugih materialov, ki so odporni proti deformacijam.

Pogonska enota - mišičje :
Sestavljena je iz agregata, hidravličnih akumulatorjev, ventilov, servoventilov in drugih hidravličnih elementov, ki so izredno hitri, natančni in vsestransko uporabni v industriji.

Krmilna enota - možgani :
Sestavljena je iz krmilne močnostne elektronike in mikroračunalniškega sistema s programsko opremo, ki robot GORO 1 uvršča med inteligentne robote.

RIZ iz Zagreba je na svojem razstavnem prostoru raztavil Hewlett - Packard - ov računalniški sistem iz serije 3000, model III. Sistem je namenjen za vodenje poslovanja v pogojih distribuirane obdelave podatkov. Operacijski sistema 3000, III in virtualni spomin omogočata priključitev in vključitev 63 uporabnikov in vključevanje v računalniške mreže, ki so sestavljene iz različnih računalnikov (HP 3000, 1 000 in IBM - ovi sistemi). V računalniku HP 3 000 III je implementirana naslednja programska oprema : IMAGE / 3 000 in QUERY za delo s podatkovnim bazam ter prevajalniki za COBOL, FORTRAN, BASIC in AML. Poleg tega je uporabnikom na razpolago še programski paket MPG 3 000 za delo na inženirskih aplikacij, planiranju porabe materjala, obdelave inventarja ipd. Osnovni sistem ima kapaciteto pomnilnika od 256 Kbytov in disk s kapaciteto od 50 mega bytov. Poleg tega proizvajalec ponuja razširitev pomnilnika do 2 mega byta in priključitev diskov do 960 mega byta. V osnovni konzuli sistema je vgrajen mikroprocesor, ki ima za nalogo, da poenostavi in olajša diagnostiko osnovnega sistema. Cena računalnikov iz serije 3 000 se giblje od 80 000 $ do 200 000 $ v odvisnosti od dodatne opreme.

DIGITRON iz Buj je razstavil inteligentni terminal TERA 2. Terminal TERA 2 je izveden v CMOS tehnologiji in z uporabo mikroprocesorjev. TERA 2 lahko opravlja knjigovodstvene posle kot je fakturiranje, obdelava osebnih dohodkov ipd., za zahtevnejše obdelave pa shrani podatke na kaseto. Spomin TERE 2 ima lahko kapaciteto od 2 Kbytov do 64 K bytov. Standardna izvedba ima 4 Kbytov spomina in 512 registrov. Tiskalnik in tastatura sta standardni I / O enoti. Implementirani makroasembler vsebuje 86 ukazov (aritmetične, logične, tranfer ter I / O ukaze). Cena terminala TERA - 2 je 750 000 din.

Madžarski Institut za meritve in računalniško tehniko je razstavil računalnik TPA - L iz serije mikroprocesorsko zasnovanih mini - raču-

nalnikov. Zgradba računalnika TPA - L je modularna in zato je ta računalnik uporaben za najrazličnejše aplikacije. Standartne I / O enote so : matrični tiskalnik, zaslon , gibki disk in CAMAC. Za računalnik TPA - L / 32 je značilna nasljedna materjalna oprema: CMOS mikroprocesor (7 registrov in 500 ukazov), ura realnega časa, pomnilnik s kapaciteto od 32 Kbytov, in CP spomin (Control Panel memory).. Vgrajeni mikroprocesorji v TPA - L imajo dve vrsti I / O sistemov: bus direktno vezan s mikroprocesorjem, ki omogoča programirani prenos podatkov, DMA, prenos podatkov z prekinitvami in notranji programirani in avtomatski DT busi, ki služijo za povezovanje periferije računalnika TPA z mikroprocesorjem ter za širjenje periferije z dodatnimi vmesniki.

Standardni konfiguraciji TPA se lahko dodajajo nasljedni pomnilniški moduli: 4K statični MOS RAM, 4k statični CMOS RAM, 3K EPROM ali PROM in 1K statični CMOS RAM za shranjevanje nespremenljivih (fiksnih) programov, 16K statični MOS RAM, 16 K statični CMOS RAM ter 8,16, 24K feritni spomin v posebnem modulu z dvojnimi vratmi.

Od perifernih enot,računalniku TPA - L lahko dodajamo še magnetno - tračno enoto, luknjalnik in čitalnik papirnatega traku, komunikacijsko opremo, terminale, vmesnike za priključitev mini, midi in velikih računalniških sistemov, ter analizatorjev tipa ICA - 70, analogne računalnike, CAMAC periferne enote, paralelne procesorje za večprocesorske sisteme in pd. Od programske opreme za računalnik TPAL so na razpolago tudi: assemblerji, STANG 1, 2, 3, 4, CAMAC, CAMACRO, programi za urejanje tekstov, za diagnosticiranje napak, prevajalniki za FOKAL, FORTRAN II, IV, BASIC, MINISOL, OPAL, MIDIBOL. Od operacijskih sistemov lahko izbiramo med: SYDAC, COS / I, RTS / i in DISC Monitor System.

Na raztavnem prostoru zastopstva podjetja Hewlett Packard je bil raztavljen prvič v Ljubljani tki. namizni računalnik iz serije 9800, model 45. Ta računalnik je namenjen predvsem inženirjem kot orodje za reševanje kompleksnih problemov s področja numerične analize, načrtovanja, procesnega upravljanja ter avtomatskega pridobivanja podatkov. Sistem 9845 se je izkazal kot odlično orodje pri načrtovanju za mehanska ogrodja, termalni analizi, planiranju proizvodnje ipd. Sistem 9845 ima 449K bytov spomina, CRT barvni zaslon z 4913 različnimi barvami, tastaturo in tiskalnik. Operacijski sistem je realiziran z ROMom, tako je ves Read / Write spomin na razpolagi uporabniku. V sistemu HP 9845 je implementiran razširjeni BASIC, ki ima nekatere lastnosti FORTRANa in APL - a, kot so: uporaba večdimenzionalnih tabel, uporaba večznakovnih spremenljivk (največ do 15 znakov), uporaba if / then stavkov, uporaba podprogramov, ON - KEY prekinitev (interrupt) ter ukazov za uporabo barvastega grafičnega terminala. Dva dodatna ROMa omogočata programiranje v assemblerju, prvi je namenjen za pisanje in razvoj programov drugi za izvajanje assemblerskih programov. Razstavljeni namizni računalnik 9845, model C stane cca 20 000 $. Poleg računalnika 9845 HP, smo si na raztavnem prostoru zastopstva tvrtke Hewlett Packard lahko ogledali razvojni sistem za vse obstoječe mikroračunalnike. Sprotna (real - time) emulacija mikroprocesorjev je omogočena z arhitekturo samega razvojnega sistema v katerem je gostiteljski procesor in njegov spomin ločen od emulacijskega sistema. Poleg klasičnih tipk v tastaturi sistema so vgrajene tipke, ki sprožijo določene funkcije, ki jih vidimo na zaslonu. Za gostiteljskim procesorjem in za njegovim spominom ter I / O modulov je prostor za 10 dodatnih plošč: za emulacijsko ploščico mikroprocesorja katerega razvijamo, za emulacijski spomin s kapaciteto 128K, za ploščico notranjega logičnega analizatorja, za PROM krmilno ploščico in za krmilnik kasetne enote. Vgrajeni PROM omogoča avtomatsko izbiro parametrov najbolj popularnih mikroprocesorjev. K razvojnem sistemu HP 6400 lahko priključimo diske s kapaciteto 20 do 120 megabytov. Ta pomnilniški medij omogoča paralelno delo šestim uporabnikom, dostop do zadnjih inačicah(verzijah)programov ter hkrati razvoj programske in materialne opreme. Z emulacijo mikroprocesorja lahko začnemo že takrat, ko je realiziran le funkcionalni del mikroprocesorja, za spomin pa uporabimo ali emulacijski pomnilnik ali sistemske ROMe in RAMe. Logični analizator sistema 6400 omogoča vpogled v delovanje busa mikroprocesorja in na ta način je možen tudi vpogled v sistem, ko ta dela z normalno hitrostjo. Delovanje sistema lahko po želji zaustavimo, nato pregledamo delovanje programov in registre nanovo inicializiramo. V razvojnemsistemu HP 6400 tvrtke Hewlett Packard so vgrajeni mikroprocesorji INTEL 8080, INTEL 8085, MOTOROLA 6800 in ZILOG Z80.

Na razstavnem prostoru Industrijsko montažnega podjetja iz Ljubljane v hali B smo si lahko ogledali sistem ARCIS, katerega so razvili v lastnem razvoju in v sodelovanju z Institutom Jožef Stefan. Sistem ARCIS je namenjen avtomatizaciji poslovanja v zgradbah. Sistem sestavljajo: komunikacijska mreža, centralna računalniška enota ter za določeno aplikacijo

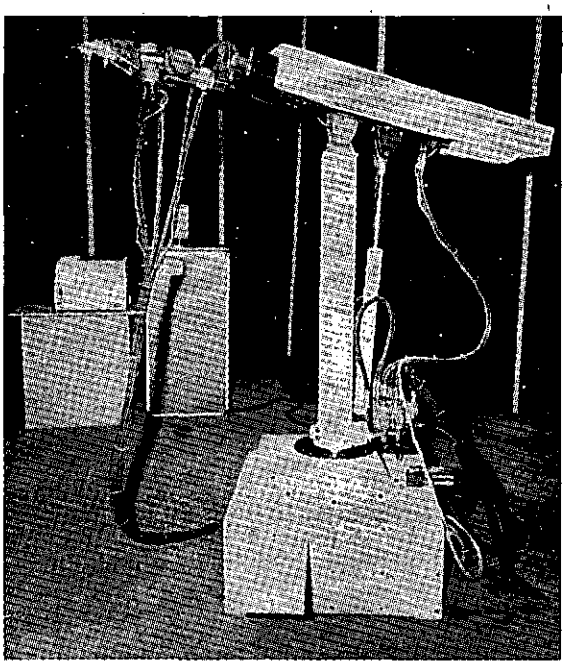
namenjene periferne enote.
Na komunikacijsko mrežo lahko priključimo nekaj sto specifičnih enot ali pa standardno računalniško periferijo kot so: video terminali, tiskalniki ipd. Sistem ARCIS - H je namensko razvit za hotele. Hotelskemu osebju omogoča lažje in hitrejše opravljanje delovnih opravil, gostom pa tudi večjo udobnost. Sistem racionalno povezuje v organizacijsko celoto hotelske sobe, službe v nadstropjih, recepcijo z vratarsko službo, administracijo hotela, kuhinjo, blagajno, telefonsko posredovalnico in vsa naplačilna mesta. Vnašanje podatkov je avtomatizirano. Računalniško obdelani in sortirani podatki so receptorski službi in vsem ostalim službam optično in pismeno posredovani. Sistem je modularno zasnovan, zato ga je mogoče postopno dograjevati in razširjati. Zmogljivost osnovne izvedbe je za 800 oseb.

DO Delta je razstavila nov video - terminal DELTA. Poleg ostalih lastnosti standardnih terminalov domače proizvodnje omogoča terminal Delta še nastavitev zaslona v dveh ravninah, dopolnitev terminala v samostojni sistem za obdelavo podatkov, uporabo posebnih grafičnih znakov, enostavno servisiranje ter branje pozicij zaslonskega kazalca. Sektor za razvoj strojne opreme v DO Delta je pokazal na sejmu Sodobna elektronika še vmesnike za vrstični tiskalnik, večkratne asinhronske serijske vmesnike za povezavo računalnikov Delta, polprevodniške spominske module ter programatorje bralnih pomnilnikov. Programatorja bralnih pomnilnikov lahko direktno priključimo na vodilo vseh računalnikov Delta in PDP 11. Vsa razstavljena oprema je plod lastnega razvoja v DO Delta. Za obdelavo bralnih pomnilnikov EPROM 2704, 2708, 2716, 2732 ter za PROM MMI 6306 so v DO Delta razvili tudi programsko opremo.

Iskra, TOZD Računalniki iz Kranja je na sejmu Sodobna elektronika pokazala svoj nov elektronski pisalnik ID - 80. Računalniški del elektronskega pisalnika je grajen na osnovi mikroprocesorja Z - 80 in ima v osnovni izvedbi 64 Kbytov hitrega spomina. V ISKRI so nam povedali, da je to gradbeni kamen novo zastavljene družine ID - 80, ki je plod lastnega Iskrinega razvoja v aparaturnem in programskem smislu. Družina ID - 80 predstavlja osnovo za razvoj in osvajanje tržišča na novem področju telematike, ki predstavlja združitev vseh vrst procesiranja na daljavo v enovite komunikacijske komplekse. Iskra ima s svojim že doslej poznanim sistemom elektronskih central vse pogoje za postopno uresničevanje teh konceptov.

Elektronski pisalnik ID - 80 omogoča računalniško editiranje (urejanje) teksta, v katerem so zajete vse znane funkcije editiranja kot je: pomnenje teksta, formatiziranje izpisa, brisanje teksta, vrivanje novih znakov in besed, avtomatsko iskanje in zamenjava odstavkov in vrstic ter premik besedila vzdolž celotnega pomnilniškega prostora. Poleg tega so na elektronskem pisalniku ID - 80 realizirane nekatere posebne funkcije kot so: priprava okrožnic ali krožnih pisem,(na določena mesta v ponavljajočim tekstom vrivamo vložke z različno vsebino,) priprava sestavljene korespondence (shranjene tekste, katere pogosto uporabljamo, pokličemo na osnovi ključev in vlagamo individualne programirane vložke odvisno od potrebe), uporaba sistema kartic (že vnaprej zapisane besede ali več besed, katere pogosto uporabljamo v določenih dokumentih pokličemo in vnesemo v tekst le na podlagi dveh znakov, katere odtipkamo «tastaturo).Tiskalnik, ki je del elektronskega pisalnika ID-80 je nov tip elektronskega tiskalnika, ki je v tujini znan kot lepopisni tiskalnik ali marjetični tiskalnik. Ime je dobil zaradi izjemno čistega tiska, oziroma zaradi mehanizma tiskanja znan pod imenom marjetica. Tiskalnik v ID-80 je izdelek firme DATA-PRODUCTS in ima hitrost tiskanja 50 znakov na sekundo. Na zaslonu elektronskega pisalnika ID-80 in v tisku tiskalnika je na razpolago kompletna jugoslovanska abeceda.

Zbrala: B.Džonova Jerman-Blažič



Robot GORO-1

IN MEMORIAM

Dr Maksimilijan Konrad, dipl. inž, naš poznati znanstveni radnik, predavač i znanstveni savjetnik, član JAZU, preminuo je 5. srpnja 1980. u 56. godini života.

Diplomirao je 1949. na Elektrotehničkom odjelu Tehničkog fakulteta u Zagrebu, a doktorirao 1955. na University of Birmingham, u V. Britaniji, obranom teze " The Cyclotron Beam Wave Form " iz područja fizike. Od duljih boravaka u inozemstvu, osim na specijalizaciji na Sveučilištu u Birminghamu od 1951. do 1955. godine, boravio je od 1965. do 1968. godina na Columbia University u New Yorku, radeći na problemima nuklearne instrumentacije.

Dr Konrad je bio jedan od vodećih znanstvenih radnika u zemlji i svijetu iz područja nuklearne elektroničke instrumentacije, te elektroničke instrumentacije i sistema u širem smislu. Bio je dugogodišnji pročelnik Odjela elektronike Instituta"Ruđer Bošković "i nosilac odnosno voditelj većeg broja istraživačkih zadaka s područja nuklearne elektronike i realizacije specijalne elektroničke instrumentacije.

Dao je istaknut doprinos u području teorije ciklotrona, a značajan je njegov rad na projektiranju i konstrukciji zagrebačkog ciklotrona.

U svijetu su poznati njegovi radovi iz područja elektroničke instrumentacije, a posebno radovi na teoriji razlučivanja pri mjerenju u nuklearnoj spektroskopiji, te na optimalnim metodama filtriranja električnih impulsa, što je od značaja za ta mjerenja. Daljnji su važni radovi dr Konrada iz područja automatskog mjerenja i obrade podataka u procesima.

Teorijski rad dr Konrada gotovo je uvijek bio povezan s primjenom tih rezultata, o čemu svjedoči niz realiziranih elektroničkih uređaja i sistema, koji su našli primjenu u znanstvenim istraživanjima i industriji.

Njegova djelatnost na području elektronike poglavito se očitovala u više od 50 objavljenih znanstvenih i stručnih radova i stručnih realizacija.

Sudjelovao je na brojnim znanstvenim skupovima u zemlji i inozemstvu - s referatima, kao pozvani predavač, te kao voditelj sjednica.

Na nivou međudržavne razmjene, boravio je kao ekspert SFRJ iz područja nuklearne elektronike u UAR 1964. godine.

Svoje iskustvo i znanje dr Konrad je prenosio na mlađe suradnike vodeći magistarske i doktorske radove, te predavanjima u nastavi dodiplomskog i poslijediplomskog studija na Elektrotehničkom fakultetu, Prirodoslovno - matematičkom fakultetu i VTŠ KoV u Zagrebu.

Do reorganizacije bio je suradnik Tehničke sekcije Odjela za matematičke, fizičke i tehničke znanosti Jugoslavenske akademije znanosti i umjetnosti u Zagrebu, a od 1975. izvanredni član JAZU.

Za istaknute zasluge dr Konrad je odlikovan Ordenom rada sa zlatnim vijencom 1962, a Ordenom zasluga za narod sa srebrnom zvijezdom 1976. godine. Republičku nagradu " Nikola Tesla " dobio je 1964. godine.

RAZISKOVALNE NALOGE PRIJAVLJENE NA RSS V LETU 1980

V tej rubriki objavljamo kratke povzetke raziskovalnih nalog,ki jih financira Področna raziskovalna skupnost za avtomatiko,računalništvo in informatiko,ki so s področja računalništva in informatike.

Naslov naloge:Računalniška obdelava slučajnih procesov

Projekt: Individualne naloge

Nosilec naloge: Drago Čepar,IJS,Ljubljana

Program raziskave:
- preučevanje metod za analizo in napovedovanje več procesov hkrati in za grupiranje časovnih vrst s pomočjo dosegljive strokovne literature in s pomočjo stikov z drugimi skupinami doma in v svetu,ki se ukvarjajo s podobnimi problemi;
- teoretično delo na posameznih metodah vključno s preučevanjem učinkov nekaterih družbeno ekonomskih pojavov na druge pojave z matematičnimi modeli;
- izdelava programov za analizo in napovedovanje dveh slučajnih procesov z upoštevanjem obojestranskega medsebojnega vpliva za računalnik CYBER 72 v obliki paketa,ki ga bo enostavno uporabljati in bo splošno dosegljiv (za leto 1980 prvi del paketa:določanje modela in delno ocenjevanje parametrov);
- programska in metodološka dopolnitev v letu 1979 narejenega paketa za grozdičenje časovnih vrst za računalnik CYBER 72;
- preizkušanje metod in programov na primerih iz prakse in zbiranje informacij o izkušnjah in o področjih uporabe teh programov drugod v svetu;
- raziskava in realizacija možnosti za napovedovanje procesov z žepnimi in namiznimi računalniki;
- vzdrževanje programov in konzultacije pri uporabi programov,ki so bili narejeni v okviru prejšnih raziskovalnih nalog;
- pisanje in razširjanje informacij o uporabnih programih in priročnikov za uporabo program.
- organizacija in vodenje seminarjev o uporabi teh metod in programov in pomoč pri njih vpeljavi v organizacijah združenega dela.

Naslov naloge: Modeli informacijskih sistemov v industriji,II faza

Projekt: Informacijski sistemi

Nosilec naloge: Saša Dekleva,Visoka šola za organizacijo dela,Kranj

Program raziskave:
- opredelitev najpotrebnejših informacijskih procesov za doseganje poslovne uspešnosti pri posameznih tipih industrijskih poslovnih sistemov;
- opredelitev modelov računalniško podprih poslovnih informacijskihsistemov za posamezne tipe industrijskih poslovnih sistemov;
- anketiranje;
- ocenitev efektov uvedbe računalniško podprtih poslovnuh informacijskih sistemov.

Naslov naloge: Raziskava računalniških modelov za razvijanje vozil,II del

Projekt: Računalniško projektiranje

Nosilec naloge: Martin Prašnički,Visoka teh. šola,Maribor

Program raziskave:

Naloga je tri letna,bi pa se predvidoma razširila na pet let.
- po prvem letu bi zbrali računske postopke in preverili ustreznost in uporabnost postavljene splošne metodologije raziskav tako,da bi ocenili neposredno korist na modelih reševanja vzmeti,voznih sil in zavor.Poznejše raziskave bi bile nadaljevanje in poglabljanje na ostale funkcijske sklope.

Naslov naloge: Aplikativna robotika

Projekt: Avtomatski industijski modularni sistemi

Nosilec naloge: Marjan Špegel,IJS,Ljubljana

Program raziskave:
- poglobitev raziskav na področju računalniškega modeliranja prostorskih objektov ter razvoj metodologije in razvoj programskega sistema za lokalno modeliranje varjencev in zvarov ter za računalniško obravnavanje lokalnih modelov vključno z njihovo interaktivno gradnjo,spreminjanjem,uporabo v varilnih procesih in njihovim prikazovanjem na grafičnem zaslonu ter implementacija tega sistema na računalnikih CYBER in PDP-11/34;
- poglobitev raziskav stanja na področju robotskih senzorskih sistemov in razvoj tipala ter pripadajočega podsistema za ugotavljanje oziroma preverjanje lege varilnih poti ter implementacija tega podsistema na računalniku PDP-11/34;
- poglobitev raziskave stanja na področju robotskega vida ter razvoj in realizacija sistema za računalniško gledanje in razpoznavanje objektov na tekočem traku emajlirnice v Gorenju ter eksperimentalna nesprotna implementacija tega sistema na računalniku PDP;
- razvoj računalniškega modela fizikalnih procesov pri obločnem varenju ter razvoj baze varilskega znanja,kar bo osnova ekspertnega sistema za avtomatično programiranje robotskega varjenja;
- simulacija realnega delovnega okolja robota za obločno varjenje v takšni meri,da bo omogočena računalniška simulacija delovanja inteligentnega robota za obločno varjenje;
- razvoj in simulacija celotnega upravljanja inteligentnega robota za obločno varjenje vključno s simulacijo učnega procesa;
- analiza možnosti in potreb za robotizacijo sestavljalnih postopkov v elektromehanski in elektronski industriji,raziskava problema programiranja mehaničnega sestavljanja s pomočjo visokih programskih jezikov ter konceptualni razvoj inteligentnega rabotskega sistema za mehanično sestavljanje;
- instalacija,dopolnjevanje in vzdrževanje knjižnice publikacij na področju robotike na računalnikih PDP na IJS,na CYBRU v RRC in DEC ljubljanske univerze,ter priprava klasificirane bibliografije za to področje.

Naslov naloge:Metode umetne inteligence v informacijskih sistemih

Projekt: Informacijski sistemi

Nosilec naloge: Ivan Bratko,IJS,Ljubljana

Program raziskave:
- razvoj in implementacija algoritmov za aktivno uporabo znanih dejstev,vsebovanih v bazi znanja o problemski domeni;
- študij metod za računalniško predstavitev znanja v ekspertnih informacijskih sistemih ter mehanizmov sklepanja;
- implementacija programa za avtomatsko učenje kot komponente ekspertnih sistemov;
- instalacija,dopolnjevanje in vzdrževanje programske opreme za implementacijo metod in tehnik umetne inteligence:prevajalniki za jezike POP 2,PROLOG in LISP za novi računalnik DEC-10 na Univerzi E.Kardelja in za PDP-11.

Naslov naloge: Računalniški merilni sistemi v " okolju "

Projekt: Računalniška avtomatizacija industrijskih procesov

Nosilec naloge: Jože Šnajder,IJS,Ljubljana

Program raziskave:
- optimizacija metod računalniških obdelav glede na poterben čas obdelave.Posebej bomo študirali zahteve pri sprotni obdelavi kompleksnih podatkov o vetru pri študiju mikroturbolenc in določanju stabilnostnih razmer v atmosferi,ki so posebej zanimive za transportne pojave v ekologiji in meteorologiji;
- merjenje izredno nizkih koncentracij škodljivih plinov reda 10^{-9} je posebno problematic v izvenlaboratorijskih razmerah,posebno še na dislociranih avtomatskih postajah.Da bi našli rešitve za te probleme bomo študirali možnosti in metode za povečanje natančnosti in zanesljivosti z uporabo mikroračunalniške teh.
- nadaljevanje študije lastnosti meteoroloških odjemnikov,ki jo kot dolgotrajno in zamudno nalogo delamo skupaj s sodelavci Meteorološkega zavoda;
- s kontuiriniranimi avtomatskimi merjenji na terenu-okolica TE Šoštanj,kjer zasledujemo koncentracijo SO_2 in vse meteorološke parametre,bomo nadaljevali študij korelacij med emisijo in imisijo v okolici.

SREČANJA

26-28 november, Frankfurt, ZR Nemčija

STRUCTO 80, Kongress fur innovatives Informations- Management

Organisator: Control Data Institut
Informacije:Control Data , Stresemannalle 30, D-6000 Frankfurt am Main 70, BDR

30 november - 2 december, Colorado Springs, ZDA

Micro 13 - 13th Annual Workshop on Micro - programming

Organizator: ACM, SIGMICRO, IEEE-CS
Informacije: G.R.Johnson, Dept. Eng. of Engineering Science, Colorado State University, Fort Collins, CO 80523, USA SPZ 1-4 december, Miami Beach, ZDA

1-4 december, Miami Beach, ZDA

5th International Conference on Pattern Recognition

Organizator: IAPR, IEEE-CS
Informacije:5-ICPR, Box 639, Silver Spring, MD 20901, USA

1-5 december, Paris, Francija

Second International Symposium on Numerical Methods in Engineering

Organizator:Groupe pour l'Avancement des Methodes Numeriques de l'Ingenieur
Informacije: P.Lascaux, Dept. Mathematique et Informatique, Conservatoire National des Arts et Metiers, 292 rue Saint Martin, 75141 Paris Cedex 03, France

8-10 december, Philadelphia, ZDA

Conference on Information Systems

Organizator: Case Western Reserve University, Minnesota University, University of Pennsylvania, ACM
Informacije:E.R. McLean, Graduate School of Management University of California, Los Angeles, CA 90024, USA

8-10 december, Boston, ZDA

ACM SIGPLAN Symposium on the ADA Programming Language

Organizator:ACM, SIGPLAN
Informacije:Robert M. Graham, COINS Graduate Research Center, University of Massachusetts, Amherst, MA 01003, USA

12-15 december, Acapulko, Mehiko

International Congress on Applied Systems Research and Cybernetics

Organizator:SGSR, CIPS, CSA, SIF
Informacije:G.E.Lasker, School of Computer Science University of Windsor, Windsor, Ont., Canada N9B 3P4 SPZ 15-17 december, Fallbrook, California, ZDA

15-17 december, Fallbrook, ZDA

Workshop on Fundamental Issues in Distributed Computing

Organizator:ACM, SIGOPS, SIGPLAN
Informacije:Barbara Liskov, MIT Laboratory for Computer Science, 545 Main St., Cambridge, MA 02139, USA

1981 leto

26-28 januar,Williamsburg, Washington, ZDA

8th Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages

Organizator:ACM SIGACT- SIGPLAN
Informacije:R.Noonan, Dept. of Mathematics and Computer Science, College of William and Mary, Williamsbourg, VA 23185 USA

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

27-28 februar, Lausanne, Švica

International Conference on Aspects of Document Preparation Systems

Organizator: ACM Swiss Chapter, IEEE, APCET, INRIA, GESO
Informacije: J.D.Nicoud, Calculatrices Digital, E.P.F.L. Bellerive 16, CH-1007 Lausanne, Switzerland

24-26 februar, St. Louis, ZDA

ACM Computer Science Confrence

Organizator: ACM
Informacije: 1981 ComputerScience Conference, Computer Science Departement, University of Missouri-Rolla, Rolla , MO 64401, USA

8-12 marec, San Diego, ZDA

5th International Conference on Software Engineering

Organizator:IEEE CSC
Informacije:L.Stucki, Boeing Computer Services Company, PO Box 24346, Seattle, Washington 98124, USA

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

26-27 marec, St Louis, 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 65401, USA

6-8 april, Ottawa, Kanada

International Symposium on Computer Message Systems

Organizator:IFIP TC-6
Informacije:R.P.Uhlig, Bell Nothern Research Dept., 3D20, Box 3511, Station C Ottawa, Ontario, Canada K1Y 4H7

30 marec- 1 april, London, Velika Britanija

ICS 1981, International Computing Symposium on System Architecture

Organizator:Microprocessors and Microsystems in cooperation with ACM European REgion
Informacije:I.S.Torsun, Dept. Comp. of Computer Science, Brunel University, Uxbridge, Middlesex UB 8 PH England

13-16 april , Mexico Sity, Mehiko

15th International Symposium on Mini and Microcomputers

Organizator:International Society for Mini and Microcomputers
Informacije: ing. Jorge Gil, Academic Secretary, MIMI, IIMASUNAM, Apartado Postal 20-726, Mexico 20 D.F.

20-24 april, Zagreb, Jugoslavija

JUREMA

Organizator:JUREMA
Informacije: Tajništvo JUREMA , ul.Djure Salaja 5/IV pp 398 41 000 Zagreb, Jugoslavija

26-29 april, Annapolis, Md., ZDA

9th Annual Telecomunications Policy Research Conference

Organizator: National Science Fondation, Federal Communications Commission, Markle Foundation
Informacije:TPRC Organizing Commitee, c/o William Taylor, Bell Laboratoires EC-258, 600 Mountain Avenue, Murray Hill, NJ 07974

12-14 maj, Minneapolis, ZDA

8th International Symposium on Computer Architecture

Organizator: ACM SIGARCH, IEEE-CS
Informacije:V.P.Franta, Computer Science Dept., Minneapolis, University of Minnesota, 143 Space Center, MN 55455, 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 Pittsburgh, Pittsburgh, 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

27-30 maj, ROUSSE, Bugarija

Organization and Automation of the Experimental Research

Organizator:State Committee of Science and Technical Progress and Central Council of Scientific and Technical Unions in Bulgaria
Informacije:The Organizing Commettee of the third Conference "Organization and Automation of the Experimental Research", 1000 Sofia, Rakovski str. 108, Bulgaria

19-12 juni, Waterloo, Ontario, Kanada

7th Conference of the Canadian Man Computer Communication Society

Organizator: Canadian Man-Computer Communication Society
Informacije:Marceli Wein, Computer Graphics Section, Division of EE, National Research Council, Ottawa, Ontario, Canada, K1A 0R8

21-25 juni, Los Angeles, ZDA

Computers in Education Program at Annual Conference of the American Society for Engineering Education

Organizator:American Society for Engineering Education
Informacije:Dean K. Frederick, Control Theory and System Program, Corporate Research and Development Center, General Electric Company, 1 River Road, Schenectady, NY 12345, USA

15-17 juni, Portorož, Jugoslavija

Second Yugoslav Symposium on Applied Robotics

Organizator:Yugoslav Committee for ETAN, Institut Mihailo Pupin, Institut Jožef Stefan, Institut za Automatiku i Računarske nauke - Energoinvest
Informacije:Yugoslav Committee for ETAN, Symposium on Applied Robotics, POB 356, 11001 Beograd, Jugoslavija

5-7 avgust, 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: J.F. Cavinese, Dept., of Mathematical Sciences, Rensselaer Polytehnic Institute, Troy NY 12181, USA

24-28 avgust, Vancouver, Kanada

7th International Joint Conference on Artificial Intelligence

Organizator: IJCAI
Informacije: Roger C. Shank, Computer Science Dept. Yale University, Box 2158 YS New Haven, CT 06520

24-28 avgust, Kyoto, Japonska

8th IFAC World Conference

Organizator: IFAC
Informacije: IFAC Secretariat, A-2361 Laxemburg Schlossplatz 12, Austria

25-28 avgust, Bangkok, Tajland

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.Box 2754, Bangkok, Thailand

9-11 september, Darmstadt, ZR Nemčija

Eurographics 81, Annual Conference of the Eurographics Society

Organizator: German Computer Society
J.Encarnaco, Eurographics 81, Technische Hochschule Darmstadt, FG Graphisch - Interactive Systeme, Steubenplatz 12, D-6100 Darmstadt, Federal Republic of Germany

28-30 september, Brno, Čehoslovaška

International Conference on Fault - Tolerant Systems and Diagnostics

Organizator: Czechoslovak Scientific and Technical Society Czech Central Committee for Electrotechnics, Central Professional Group for Diagnostics in Electronics, House of Technology ČSVTS in Ceske Budejovice
Informacije: DUM Techniky ČSVTS, Trida 5, kvetna 42, 37 021, Ceske Budejovice, ČSSR

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

Organizator: ACM
Informacije: A.J. Toni Shetler, Xerox Business Systems, AL-30, 701 South Aviation Blvd., El Segundo, CA 90245, USA

SEMINARJI

V mednarodnem kongresnem centru v Zahodnem Berlinu je George Washington University, School of Engineering and Applied Science organiziral naslednje seminarje:

12-16 januar 1981
ECM and ECCM for Digital Communications

12-16 januar
Bubble Domain Memory Technology

19-23 januar
Fiber and Integrated Optics

19-23 januar
Designing and Operating Computer Distributed Information Systems

26-23 januar
Remote Sensing for Global Resource Applications

STROKOVNE RAZSTAVE

25 november- 3 december 1980

Systemotechnika 80
Mednarodni Sejem za Birotehniko, Leningrad USSR

28 november-7 december

Salon International des Inventions et des Techniques nouvelles Mednarodni Sejem za Inovacije in Nove tehnike, Genf, Švica

28 november-7 december

Techn. Messe fur Industrielle Electrik, Automatisierung und Elektronik
Mednarodni Sejem za Industrijsko Elektrotehniko, Avtomatiko in Elektroniko, Madrid, Španija

28 november-7 december

Breadbord 80 - The Kits and Bits Show for the Home Electronic Enthusiasts
Sejem elektronskih komponent in opreme za amaterje, London, Anglija

Vabilo k sodelovanju

Od 20 do 24 aprila 1981 bo održana tradicionalna prireditev JUREMA. V okviru prireditve je organizator pripravil naslednja posvetovanja:

-Posvetovanje o digitalnem upravljanju proizvodnih procesov
-Osmi simpozij o sistemih in kibernetiki v sodobni znanosti in družbi
-Drugi Simpozij o tehnični diagnostiki

-Posvetovanje o izobraževanju na področju avtomatike in teorije sistemov

-Mednarodna razstava merilne in regulacijske tehnike in avtomatizacije

Rok za izdelavo in pošiljanje prispevkov je 1 februar 1981. Prispevke je treba poslati na naslov : Tajništvo JUREMA , ul.Djuro Salaja 5/IV pp 398 , 41 001 Zagreb. Informacije lahko dobite na telefon 041 - 518-811.

Vabilo k sodelovanju

Jugoslovenska zveza za ETAN v sodelovanju z Institutom Mihailo Pupin iz Beograda, Institutom Jožef Stefan iz Ljubljane in Institutom IRCA iz Sarajeva je organizator Drugega Jugoslovenskega simpozija za uporabno robotiko.Prvi simpozij je bil održan 1979 l. v Dubrovniku.
V okviru Drugega Simpozija, ki bo održan v Portorožu od 15-17 junija 1981l., bo organizirana razstava o dosežkih Industrijske robotike.

Organizator vabi autorje da pošljejo prijavo prispevka in vsebino prispevka (največ 100 besed) najkasneje do 1 decembra na naslov:

Yugoslav Committee for ETAN
Symposium on Applied - Robotics
P O 3 356
11001 Beograd

Prispevki so lahko iz naslednjih področij Simpozija :

-Mehanika robotov in manipulatorjev
-Sinteza funkcionalnega gibanja
-Upravljalni algoritmi in njihova uporaba
-Konstruiranje robotov in manipulatorjev s pomočjo računalnikov
-Komponente robotov in manipulatorjev
-Senzori in elementi umetne inteligence
-Industrijska aplikacija in realizacija
-Ostale aplikacije robotov in amnipulatorjev

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