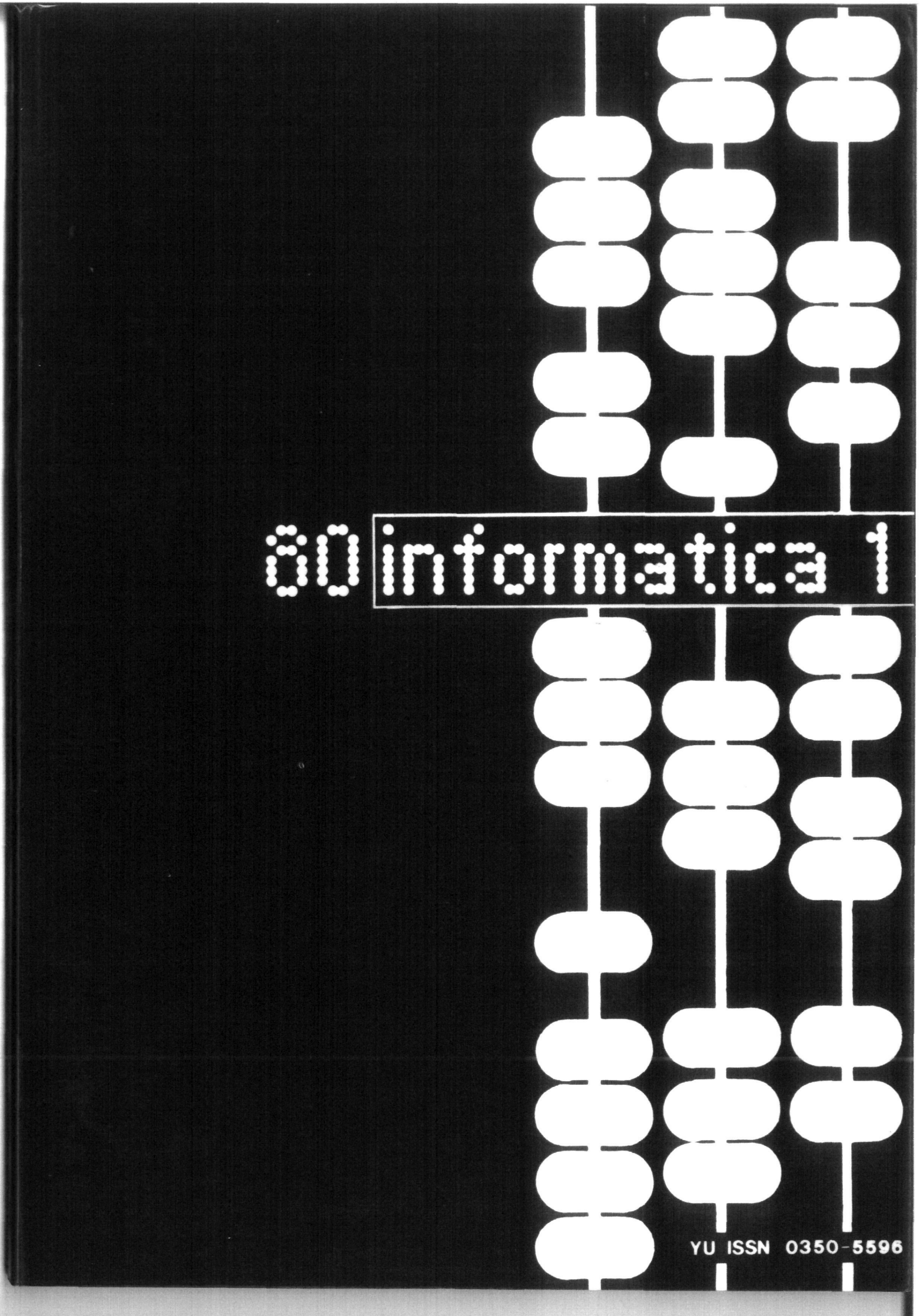


YU ISSN 0350-5596

1892 - 1980

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. 1

CONTENTS

informatica

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

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

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

Pri financiranju revije sodeluje tudi Raziskovalna skupnost Slovenije.

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

Tisk: Tiskarna KRESIJA, Ljubljana

Grafična oprema: Rasto Kirn

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

YU ISSN 0350 – 5596

LETNIK 4, 1980 – št. 1

VSEBINA

V NOVI LETNIK ČASOPISA INFORMATICA

ANTON P. ŽELEZNIKAR

SLOVENSKO DRUŠTVO INFORMATIKA

Časopis INFORMATICA je zaključil svoje poslovno leto 1979 s standardno zakasnitvijo ter z uspehi in tudi s pomanjkljivostmi, vendar je smiselno na začetku letnika postaviti vprašanja njegove drugačne in dopolnilne usmeritve pa tudi našega osebnega dela in prizadevanj v delovnih organizacijah, ki sta jim računalništvo in informatika temeljni proizvodni, razvojni in tržni dejavnosti. Nanizajmo v tem uvodniku le nekaj tistih izhodišč, ki se pojavljajo v praksi računalništva in informatike v prelomnem razdobju, ki je pred nami.

Avtorska baza časopisa ostaja še naprej preozka: predvsem nam manjkajo prispevki inženirjev in upravljavcev iz proizvodnih organizacij za računalništvo, npr. v SR Sloveniji iz SOZD Iskra, Gorenje in Elektrotehna (Delta). Nimamo prispevkov s področja organizacije proizvodnje računalnikov, programske opreme, planiranja, trženja, izvoznih dosežkov in možnosti, razvoja in vzgoje kadrov v industrijskem okolju.

Nadalje pogrešamo prispevke strokovnih služb zadevnih republiških organov, kjer se koncipirajo razvojni in restriktivni ukrepi v okviru izvajanja določene politike, planskih smernic in predvidevanj. Strokovne službe in upravni organi bi prek časopisa lahko razgrinjali in pojasnevali svoje zamisli in ukrepe tudi strokovni javnosti, seveda v primerni, kratki in zanimivi obliki.

Časopis INFORMATICA naj bi vspodbujal k kritičnim stališčem do pojavov neustreznega, neidentificiranega in stihijskega razvoja računalniških in informatičnih dejavnosti. Strokovnjaki in zainteresirani družbeni in gospodarski sektorji bi lahko bolj prizadevno podpirali kritično konfrontacijo in alternativnost mnenj v okviru polemičnih prispevkov. Na ta način bi vsi skupaj dvigali strokovno kulturo izhodišč, ukrepov in restrikcij v nadaljnem razvoju računalništva in informatike.

Nujna bi bila večja občutljivost, opredeljenost in zavzetost nasproti nastajajoči računalniški ter z njo povezani industriji. Tu nam zaenkrat manjkajo ustrezni strokovni, tehnološki, ekonomski in razvojni kriteriji. Nemogoče je planirati proizvodnjo in njen izvoz na neopredeljenosti in pomanjkljivi identifikaciji računalniške tehnologije. Ugotavljamo kar povprek, da nimamo prave proizvodnje računalnikov, da smo le sestavljavci in prodajalci tujega. Doslej še nismo potegnili ločnice, naše, dogovorjene, s katero bi lahko opredelili, kje začenja od nas in za nas priznana proizvodnja, od kje in do kod smo sestavljavci in s kakšnimi kriteriji in zakaj in do kod dopuščamo tudi omejeno, toda čisto prodajo tujega.

Prišel je čas, ko je v SFRJ moč integrirati sposobne proizvodne faktorje, gospodarske organizacije, ustanove in posameznike, s široko in odprto samoupravno platformo, v kateri so združljivi interesi sposobnosti in pripravljenosti, tehnološkega napredka, tržne uspešnosti in povezovanja partnerjev. Ta pobuda naj bi bila prisotna na vseh upravljavskih ravninah, pri nosilcih in sodelavcih združevalnega projekta. Seveda pa je na zahtevnem tehnološkem in organizacijskem področju potreben stalen razvoj združevalnega aparata, njegovo prilagajanje in skrb za njegovo intenzivno delovanje.

Časopis INFORMATICA pogreša tudi sodelovanje študentov, opise njihovega dela in problematike. Določena skrb za kvalitetnejše delovne pogoje študentov je nekako odsotna, ni prave pobude s strani študentske organizacije (na ustreznih fakultetah) niti s strani izobraževalnih ustanov. Vendar so tudi izjeme. Mestna raziskovalna skupnost Ljubljane je predvidela posebno vzpodbujanje raziskovalne dejavnosti študentov, seveda pa bi naj to veljalo tudi za področje računalništva in informatike.

Časopis INFORMATICA stopa tako v letnik 1980 s pobudo, s pozivom avtorjem, da bolj pogosto, strokovno in prizadevno sodelujejo s svojimi prispevki. Ker v letu 1980 zaradi ustalitvenih naporov ne bo organiziran simpozij INFORMATICA 80 na Bledu, poziva Slovensko društvo INFORMATIKA vse potencialne avtorje za ta simpozij, da pošljejo svoje prispevke v časopis, ki bo po potrebi lahko tudi znatno razširjen.

Naj končam! Časopis INFORMATICA je tudi vaš časopis, zato mu dodajte drobec, novico, razpravo, članek tudi vi, inženirji, tehniki, upravljavci, študentje. Skozi časopis se tudi vi vključujete v napore tehnološkega, ekonomskega in raziskovalnega napredka sodobne družbe, zlasti pa računalništva in informatike.

RAZVOJ RAČUNALNIŠKIH SISTEMOV

ANTON P. ŽELEZNIKAR

UDK: 681.32

SOZD ELEKTROTEHNA, DO DELTA, LJUBLJANA

Članek opisuje sodobne tokove razvoja računalniških sistemov, tako da zajame bistvene sistemske koncepte, kot so virtualnost, porazdeljena obdelava posatkov, multiprocesorski sistemi in podatkovne mreže. Prikazana je primerjava, povezava in odvisnost teh sistemskih zamisli s tehnološkega, operacijskega in ekonomskega/funkcionalnega vidika. V kontekstu podatkovnih in računalniških mrež je poudarjena pomembnost mikroračunalniške tehnologije in njene ekonomičnosti v kompleksni sistemski povezavi. Na koncu so opisane perspektive in lastnosti supraprevodnih superračunalnikov, ki bodo lahko s svojimi zmogljivostmi kvantitativno in kvalitativno presegli zmogljivost človeških možganov.

DEVELOPMENT OF COMPUTER SYSTEMS.
This article describes trends of modern computer system development including essential system concepts of virtuality, distributed data processing, multiprocessing and data networks. Comparision and dependence of these system concepts are made from technological, operating and price-performance aspect. In the context of data and computer networks the microprocessor technology is becoming more and more important where microcomputer systems are considered at the lower end of computer networks. Performances of superconductive supercomputers being in development are described and compared with the human brain performance.

1. UVOD

V razvoju računalniških sistemov poznamo nekatere značilne usmeritve, ki jim lahko prisodimo določeno prednost. Poznamo mikro, mini in tudi megaračunalnike ali superračunalnike, ki jih napoveduje najsodobnejša materialna tehnologija. V okvir zapletenih sistemov se uvrščajo predvsem računalniške mreže, kot univerzalen in smiseln koncept integriranega, planetarnega ali lokalnega sistema. Ta sistem zahteva razvite komunikacije in teleprocesiranje, ob tem pa ostaja sistemska zmogljivost še naprej eno glavnih izhodišč in ciljev.

Zmogljivost računalniških sistemov in njihovih izpeljank, predvsem informacijskih sistemov, postaja zahteva okolja in vse bolj razvite računalniške uporabe. Večkrat je zaželjena in smiselna celo t.i. navidezna ali vitrualna zmogljivost, ki omogoča, da uporabniško izkoristimo že obstoječo arhitekturno zmogljivost računalniškega sistema. Virtualnost sistema je seveda lahko večnamenska in vsestranska, npr. pomnilniška, uporabniška, sistemska, periferna itn. Druga vrsta ekonomske zmogljivosti je distribuiranost ali porazdeljenost zmogljivosti ali inteligence v mrežnem sistemu. Skladno s položajem v računalniški mreži želimo imeti podatke in zmogljivost ustrezno ekonomično porazdeljene v zapletenem, povezanem omrežnem sistemu.

V novejšem času se pojavljajo razvojni projekti pa tudi že realizacije, ki napovedujejo nov val sistemov, t.i. supersisteme, ki naj bi prvič v sodobni človeški zgodovini presegli zmogljivost človeških možganov. Tem koncepcijam in realizacijam botruje nov supraprevodni element, ki je pogojen z Josephsonovim efektom oziroma spojem.

Cena računalniške tehnologije in s tem računalniških sistemov hitro pada, narašča pa operacijska hitrost. Mikroračunalniki in njim podobne naprave postajajo vedno bolj pomembni in priročni za vsakodnevno uporabo, postajajo masovni sistemi, ki se bodo vključevali v največje sisteme iz naših domov. Procesno krmiljenje v tovarnah in robotizacija opravil bo omogočena prav z mikroračunalniki in z njihovimi upravljavci v mrežah in v bodočih sodobnih informacijskih sistemih.

Seveda pa je moč povečati sistemsko zmogljivost tudi z nevirtualnimi pristopi, ko se uporabi multiprocesorski sistem in uvede multiprogramiranje. Zgradba večprocesorskega sistema je seveda drugačna od virtualne zgradbe in predstavlja določeno njeno nasprotje. Multiprocesiranje pomeni dejansko povečanje zmogljivosti glede na en sam procesor, virtualnost pa uporablja en sam procesor za več zaporednih procesov. Združitev večprocesorskega koncepta in virtualnosti pripelje do zelo učinkovitih in ekonomsko utemeljenih sistemskih rešitev.

Opišimo po vsem tem nekatere bistvene sistemske koncepte in težnje njihovega razvoja.

2. Z VIRTUALNOSTJO POVEČANA ZMOGLJIVOST

V okviru sistemskega razvoja imamo nove in stare zamisli v arhitekturi (funkcionalni zgradbi) in v operacijskih zmogljivostih sistemov. Eden teh konceptov je virtualnost, ki zahteva prilagoditev zgradbe procesorja in ustrezen operacijski sistem, čeprav je poudarek, kot bomo opisali kasneje, na operacijskem sistemu.

Storilnost računalniške obdelave podatkov lahko na določen način znatno povečamo z uporabo

** interaktivnih sistemov in
** sistemov s porazdeljenim časom

Najbolj značilna sistemska rešitev za povečano interaktivnost in razdeljevanje sistemskega časa več uporabnikom je zamisel

virtualnega stroja,

ki je sposoben proizvesti več programskih kopij različnih realnih računalniških sistemov na enem samem realnem procesorju. Vsak virtualni ali navidezni stroj ima neko popolno zbirko vhodnih/izhodnih naprav in zmore funkcije, ki so enake realnemu stroju.

Primeri tipičnih virtualnih strojev so VAX-11/780 (glej (1), proizvajalec je tudi Delta/Elektrotehna), IBM System/38 (3) in operacijski sistem IBM VM/370 (2). Zadnji primer je virtualni stroj, ki ga dobimo z operacijskim, tj. s programskim sistemom. Novi virtualni računalniki imajo vgrajeno tudi dodatno materiano opremo, za bolj učinkovito doseganje svoje funkcije virtualnosti.

Namen virtualnega stroja je materialno navidezno in odtod programsko in funkcionalno dejansko povečanje zmogljivosti oziroma izkoriščenosti računalnika in navidezno povečanje obsega oziroma naslovnega prostora njegovega pomnilnika. Navadno razumamo pod virtualnostjo na določen, toda vselej navidezen način povečan naslovni prostor, ki se kaže uporabniku kot praktično neomejen. Ta neomejenost je v visokem številu razpoložljivih navideznih pomnilnih lokacij, npr. 280 trilijonov lokacij v IBM S/38, 2 trilijona lokacij v Cyber 205, in 4G lokacij v VAX-11/780, čeprav je število dejanskih lokacij neprimerno nižje (npr. samo 256k).

Gledanje računalniškega sistema skozi njegove navidezne pomnilniške zmogljivosti je seveda omejeno, ker zajema zamisel virtualnega stroja tudi razdeljevanje procesorskega časa več uporabnikom, to pa pomeni interaktivnost in višjo stopnjo organizacije zgradbe procesorja in pripadajočega operacijskega sistema. Ta operacijski sistem omogoča dostop do računalniških zmogljivosti več uporabnikom prek tastaturnega in prikazovalnega terminala in ostalim kanalom v času, ki je posameznemu uporabniku dodeljen. Uporabnik razpolaga tako s funkcijami testiranja, konverzije in izvajanja svojih programov.

Operacijski sistem virtualnega stroja mora upravljati z računalniškimi in perifernimi viri, tako da vsak uporabnik sistema, ki je blizu ali oddaljen, razpolaga s celotno kopijo sistema, vključno z vsemi vhodnimi/izhodnimi napravami. Pri tem je razumljivo, da vsak uporabnik izbere svoj operacijski sistem na različnih virtualnih konfiguracijah stroja. Virtualni sistem se kaže zunanjemu opazovalcu kot naprava, ki izvaja hkrati več uporabniških operacijskih sistemov.

Za boljše razumevanje zamisli virtualnega sistema si oglejmo mehanizem navideznega pomnilnika. T.i. stranenje (paging) ali delitev programa pri njegovem izvajanju na pomnilniške strani (stran je tu mišljena v pomenu strani v knjigi) je povezano s pojmom virtualnega pomnilnika. Programer lahko v primeru pomanjkanja pomnilniškega prostora sam razdeli svoj program na segmente, ki jih imenujemo plasti (overlay). Plast je na določen način pomensko (funkcionalno) zaokrožena celota, ki je le tako obsežna, da jo je moč shraniti na določeno količino lokacij glavnega pomnilnika, tj. na eno stran. Pri izvajanju programa se najprej naloži prva plast, ki se določen čas izvaja. Ko je izvajanje te plasti končano, se včita naslednja plast, ki se zopet izvaja in tako naprej. V tem primeru programer sam odgovarja za ustrezno delitev programa na plasti, ko določi mesto plasti v sekundarnem pomnilniku, prenos plasti med glavnim in sekundarnim pomnilnikom in tako sam ureja celoten proces delitve programa brez pomoči računalnika.

Avtomatizacija delitve programa na plasti in izvajanje plasti imenujemo mehanizem virtualnega pomnilnika in ta razreši programerja vsakega knjigovodstva in premišljevanja o plastni strukturi zadevnega programa. Zamisel navideznega pomnilnika je tedaj v ločitvi naslovnega prostora programa od dejanskih pomnilniških lokacij fizičnega pomnilnika.

Imejmo tale primer. Dan naj bo računalnik, ki sicer ima 64k-zložni naslovni prostor, toda samo 4k zlogov dejanskega pomnilnika za uporabniške programe. Naslovni prostor računalnika vsebuje naslove 0, 1, 2, ... , 65535. Zamisel ločitve naslovnega prostora programa in dejanskih pomnilniških naslovov je npr. tale: v vsakem trenutku lahko imamo dostop do poljubne dejanske lokacije v intervalu 0, 1, ... , 4095 danega fizičnega pomnilnika. Za računalnik pa predpišemo, da pomeni programski naslov 4096 fizični naslov 0, naslov 4097 fizični naslov 1 in končno naslov 8191 fizični naslov 4095 itn. Stranenje, ki je avtomatično, pomeni seveda, da je potrebno ustrezno stran vselej naložiti v glavni pomnilnik, ko se navede naslov iz območja te strani. Postopek stranenja postane tako avtomatičen, spremlja pa ga relativno pogosten postopek nalaganja strani iz sekundarnega pomnilnika v glavni.

Pri zgradbi (arhitekturi) virtualnega stroja ločimo dve usmeritvi: uporabniško in sistemsko. Virtualnost je tedaj tudi uporabniška in sistemska in je kot taka namenjena dvem kategorijam strokovnjakov-uporabnikov. Za navadnega uporabnika so pomembni podatki o virtualnem naslovnem prostoru (npr. 4 giga zlogov pri VAX-11), o razpoložljivih podatkovnih tipih, o ukaznih formatih itn. Sistemski programer želi informacijo o prednostnih ukazih, zgradbi procesa, upravljanju pomnilnika, zgradbi prekinitev itd. Ukazi na višji ravnini vsebujejo npr. še upravljanje poslov in čakalnih vrst, programske povezovalne funkcije itn. Veliki, virtualni, enolično naslovljivi pomnilnik sistema je lahko krmiljen z enim samim naslavljalnim mehanizmom za pomnilnik, periferne naprave in podatkovne ter sistemske vire. Opisani koncepti računalniške arhitekture omogočajo tako uresničitev sistemov z visoko in kompleksno stopnjo virtualnosti, ki je pomnilniška, uporabniška, periferna, mrežna, sistemska in operacijska.

Visoka stopnja virtualnosti kot sistemske lastnosti zagotavlja dobro in večstransko izkoriščenost sistema, njegovo uporabo na več

mestih v mreži in pri različnih uporabnikih. Prihodnji razvoj malih, srednjih in velikih računalniških sistemov bo uporabo virtualnosti še povečal, in sicer s kompleksnostjo v arhitekturi procesorjev in v operacijskih sistemih, to pa bo omogočilo razvoj nove, razširljive in povezljive materialne in programske opreme v sistemih, ki bodo povezani in medsebojno procesno sodelujoči v podatkovnih mrežah.

Virtualnost prav gotovo ni le enostaven koncept, ki bi omogočal uporabnikom izvajanje neskončno dolgih programov, kot napačno mislijo nekateri nepoučeni in z virtualnostjo komercialno seznanjeni prodajalci. Velik pomnilniški prostor bo npr. olajšal komuniciranje v mrežah, delitev procesorskega časa pa bo pomembna za dodeljevanje določene procesne moči specifičnemu uporabniku. Baze podatkov bodo postale absolutno naslovljive neglede na njihovo velikost in število itn. Te možnosti so bile upoštevane zlasti pri razvoju sistema /38, ki je predviden za vključevanje v svetovno satelitsko mrežo.

3. Dejanska zmogljivost multiprocesorjev

Multiprocesorski in multiprogramirani sistemi predstavljajo prehod iz virtualnih (navideznih) lastnosti na dejanske. Seveda je mogoče sistemski lastnosti virtualnosti in multiprocesiranja tudi združiti. Vendar velja, da so multiprocesorski sistemi lahko časovno in funkcionalno bolj zmogljivi od čistih virtualnih sistemov, saj imamo namesto razdeljevanja časa enega procesorja več uporabnikom paralelno izvajanje več uporabniških in/ali sistemskih programov. Zamisel multiprocesiranja ni nova in delitev poslov na neodvisno delujoče procesorje za izvajanje vhodnih in izhodnih funkcij, komuniciranja med periferijo in centralnimi procesorji ter izvajanje več uporabniških programov hkrati je bila realizirana na velikih sistemih (IBM, CDC, Borroughs, Univac in drugje). Takšni sistemi pa zahtevajo višjo stopnjo sistemskega programiranja, kjer se uporabljajo pripomočki za koordinacijo med časovno paralelno tekočimi procesi. Ti materialno omogočeni programski koncepti so semaforji, monitorji in procesni moduli (5, 6, 7), ki se izdatno uporabljajo v operacijskih sistemih multiprocesorjev.

Tem novejšim sistemskim konceptom se je dobro prilagodila mikroprocesorska tehnologija, ki premore integrirane centralne procesorje z neposredno podporo za koordinacijo paralelnih procesov ter posebne, dodatne procesorje za upravljanje s pomnilniki (nadomestilo virtualnosti, avtomatično premeščanje in povezovanje modulov pri nalaganju v hitri pomnilnik), izredno funkcionalne in zmogljive periferne, aritmetične, prekinitvene, šifrirne in druge integrirane procesorje, ki skupaj delujejo kot kompleksen in raznovrsten multiprocesorski sistem.

Z mikroprocesorji so bili zgrajeni poskusni večprocesorski sistemi, ki so presegli zmogljivosti najmočnejših komercialnih računalniških sistemov. Pri tem so povezali več sto in tisoč mikroprocesorjev v računalniški sistem. Novejši 16-bitni mikroprocesorji (I8086, MC68000 in Z8000) pa zdržijo primerjavo tudi z nekaterimi srednevelikimi sistemi družine PDP-11 in jih celo funkcionalno in po operacijski hitrosti presegajo. Vsi ti procesorji imajo že vgrajeno materialno in programsko podporo za multiprocesorske sisteme.

4. Sistemi s porazdeljeno (distribuirano) obdelavo podatkov

Zamisel porazdeljene ali distribuirane obdelave podatkov (kratko POP) je v sistemski gradnji že nekaj časa prisotna in naj bi reševala problematiko obdelave podatkov med več računalniškimi lokacijami. Tehnološki razvoj je omogočil tudi ekonomsko uresničitev te zamisli, pa tudi vsakodnevne zahteve uporabnikov so narekovale take rešitve. V okviru porazdeljene obdelave podatkov poznamo več opredelitev in praktičnih rešitev, vendar opišimo le nekatere.

POP je določena realizacija povezave množic programov v dveh ali več centrih za obdelavo podatkov. Te centre imenujemo kratko vozlišča. Programi so povezani tako, da si delijo podatke ter jih medsebojno izmenjujejo. Vsako vozlišče lahko vobče izvaja obdelavo podatkov neodvisno, torej ima svoj podatkovni pomnilnik in naprave za izvajanje programov.

Komunikacija, ki je potrebna pri delitvi in izmenjavi podatkov med programi, je lahko

** sinhrona ali
** asinhrona

Sinhrona komunikacija predpostavlja, da sta oba programa hkrati v izvajanju in da tečejo sporočila v obeh smereh, ko prvi program čaka drugega, da mu ta odgovori na njegovo zadnje sporočilo in obratno. Asinhrona komunikacija predpostavlja, da oddajni program ne čaka na odgovor in da ni potrebno istočasno izvajanje obeh programov. Pri asinhronem načinu delovanja se komunikacijska sporočila navadno postavijo v vrsto in se kasneje paketno obdelajo.

Mehanizmi za komuniciranje so naprave za neposredni pomnilniški dostop (DMA) ali pa telekomunikacijske povezave. Pri asinhronem načinu komunikacije se lahko uporabljajo prenosljivi pomnilniški mediji, kot sta tračni kolut in disketa. V teh primerih imamo teleprocesorske povezave pa tudi druge oblike komunikacijskih mehanizmov.

Bistveni element definicije POP je komunikacija med programi. Kadar dve vozlišči medsebojno ne komunicirata, ko izvajata uporabniške programe, govorimo o

** decentralizirani obdelavi podatkov

Decentralizirana obdelava podatkov (DOP) je tako le poseben (poenostavljen) primer porazdeljene obdelave podatkov.

Bistvene lastnosti porazdeljene obdelave podatkov so tedaj tele:

** Funkcije in podatki so razpršeni v mreži računalnikov (tu ni nujno, da so geografsko razpršeni)

** Funkcije in podatki so razpršeni na koordiniran, usklajen način, kjer skrbi za koordinacijo eden ali več logičnih upravljavnikov podatkovne mreže

** Vozlišča so avtonomna le do določene stopnje in so odvisna od drugih vozlišč glede koordinacije in krmiljenja

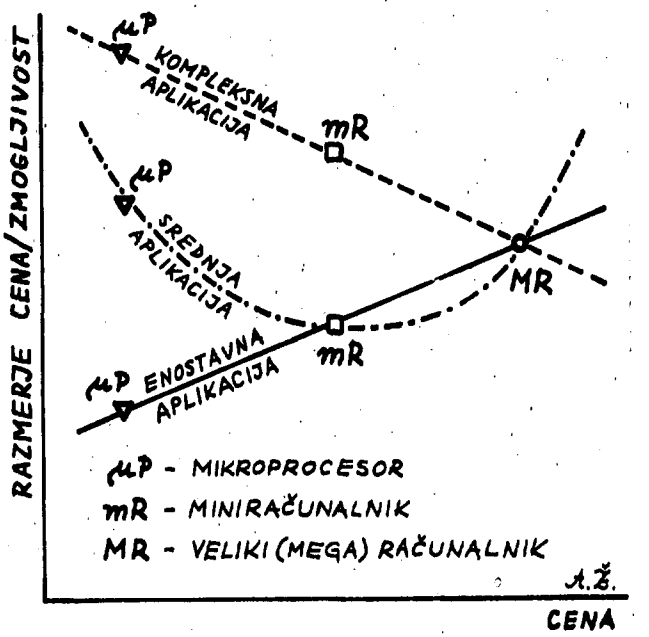
Večkrat govorimo o porazdeljenih ali distribuiranih informacijskih sistemih (PIS). Porazdeljeno pomeni, da imamo logično združeni informacijski sistem, ki pa ima fizično porazdeljene funkcije in podatke vsaj prek dveh računalniških enot. Uporabnik ima v PIS dostop do podatkov in funkcij iz vsake udeležene,

povezane računaniške enote, podobno kot v centraliziranem informacijskem sistemu. Z organizacijskega vidika pa pomeni PIS tudi, da se nahaja obdelovalna moč tam, kjer je potrebna.

Ker je v PIS informacija (programska in podatkovna) porazdeljena, mora ta sistem razpolagati s komunikacijskim sistemom za prenos in izmenjavo sporočil in podatkov ter z upravljavnim sistemom za sinhronizacijo, integriteto, skladnost, zasebnost, varnost in namestitev podatkov. Porazdelitev upravljanja pomeni dodeljevanje, iskanje in dostop do mrežnih virov ter sinhronizacijo uslug, kot sta npr. obnavljanje podatkov (ažuriranje) in njihovo vzdrževanje. Nepomnilniške komunikacije prek kanalov in linij pa zahtevajo zapletene in zanesljive podatkovne protokole.

Vzpodbuden dejavnik za uporabo POP je prav gotovo sodelovanje različnih računalniških sistemov in različnih tipov procesorjev, od velikih sistemov prek specializiranih miniračunalnikov do mikroprocesorjev. Takšni parcialni sistemi in računalniki v mrežnem sistemu lahko imajo znatne prednosti v razmerju cena/zmogljivost nasproti velikim in univerzalnim računalniškim sistemom. Enostaven primer nam to pokaže. Vzemimo, da določen aplikativni program zahteva osembitno aritmetiko in da ocenjevalni program (benchmark) meri zmogljivost različnih kandidatov-procesorjev za zaporedje ukazov, ki izvaja osembitno aritmetiko. To oceno je moč opraviti praktično na vsakem računalniku, od enostavnega mikroprocesorja do najmočnejšega megaračunalnika. V tem primeru je razmerje cena/zmogljivost tudi tisočkrat manjše za mikroprocesor kot za veliki centralni procesor.

Slika 1 kaže odvisnost med procesorskim razmerjem cena/zmogljivost in ceno procesorja za vrsto arhitektur in stopenj aplikacijske kompleksnosti. Razmerje cena/zmogljivost je optimizirano tedaj, ko je aplikacijska kompleksnost prilagojena arhitekturi. Spremembe razmerja cena/zmogljivost so lahko različne za tri velikostne razrede med mikroprocesorjem in velikim računalnikom ter za dva velikostna razreda med mini- in mikroračunalnikom. To kaže, da bodo mikroračunalniki odigrali odločilno vlogo v porazdeljeni obdelavi podatkov, saj je tu vrsta nalog z enostavnimi aplikacijami.



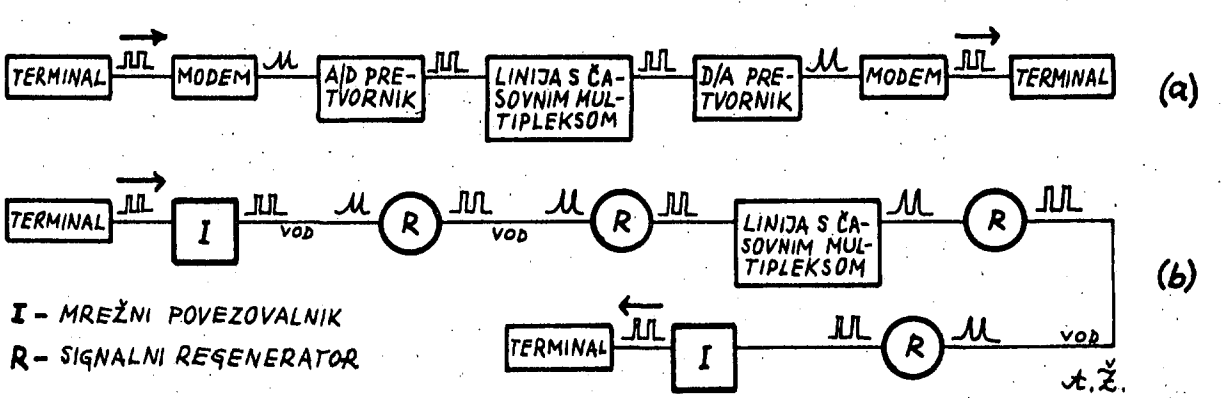
SLIKA 1. Razmerje cena/zmogljivost kot funkcija cene za tri računalniške arhitekture in aplikacijske kompleksnosti

5. RAČUNALNIŠKE (PODATKOVNE) MREŽE

Razvoj v smeri porazdeljene obdelave podatkov z mrežami računalnikov je povzročil novo, tesnejšo povezavo med računalniško in komunikacijsko industrijo. S tem je naraslo zanimaje in pojavile so se potrebe za razvoj komunikacijskih protokolov, za delovanje in upravljanje računalniških mrež in za ustrezno novo tehnologijo. Možnosti povezave računalniških in komunikacijskih dejavnosti ter skupnih uslug za širok uporabniški prostor so obojestranske in v tej povezavi in združevanju se pojavlja izrazito inovativno področje, z novimi vprašanji in nalogami, ki jih bo potrebno reševati v prihodnosti. Javni in družbeni interesi so v razmerju računalništva in komunikacij posebej poudarjeni, saj gre tu za masovne in javne usluge ter za uvajanje in proizvodnjo nove in investicijsko zahtevne tehnologije.

Potrebe po novih telekomunikacijskih uslugah bodo narekovale večjo določenost v zgradbi računalniških mrež, napredek v krmiljenju mrež in razvoj novih aplikativnih sistemov, ki ostajajo do nadaljnega odprti za tiste inovacije, ki bodo zagotovile dovolj hitre možne rešitve bistvenih problemov. Tu so

SLIKA 2. Analogna (a) in digitalna oblika (b) prenosa podatkov v računalniški mreži



možnosti za razvoj in raziskave zares raznovrstne. Združevanje obdelave podatkov in telekomunikacij je tako očitno zlasti na inovativnih področjih, kot so komunikacijski sateliti, optični prenos podatkov, mikroprocesorji, magnetni mehurčki in tehnologija velikega zgoščevanja elementov v integriranih vezjih.

Nove aplikacije v okviru združevanja računalništva in telekomunikacij se kažejo v prihodnosti na področjih, kot so potovalne rezervacije, elektronsko bančništvo, prodajne in nabavne transakcije, računalniške konference, shranjevanje in razdeljevanje zvočnih sporočil, razdeljevanje faksimilov in obdelava slik, interaktivna grafika, procesiranje tekstov in razdeljevanje dokumentov ter v t.i. televizijskih informacijskih uslugah. Tu so velike nove možnosti za združeno domačo računalniško in komunikacijsko industrijo.

S sistemskega stališča je računalniška ali podatkovna mreža eden osrednjih prihodnjih problemov računalništva, telekomunikacij in informatike, kot združenih in medsebojno odvisnih dejavnosti. Pojem podatkovne mreže je več kot dve desetletji prisoten v razmišljanju in pri sistemski gradnji, ni pa vselej razjasnjeno vprašanje njegove konceptualne, funkcijske in materialne zapletenosti. Na planetu bo letos obratovalo že okoli 30 javnih podatkovnih mrež (11). V petdesetih letih so bile telefonske analogne linije prvič uporabljene za prenos računalniških podatkov.

Shema prenosa podatkov po linijah v analogni obliki je prikazana na sliki 2(a), v sodobnejši digitalni obliki pa na sliki 2(b). Prednost digitalnega v primerjavi z analognim prenosom je v relativno enostavni regeneraciji izvirnih digitalnih signalov vzdolž in povprek v podatkovni mreži. Zaradi te značilnosti, tj. možnosti regeneracije, imamo v digitalnih mrežah le en napačen bit v desetih milijonih bitov, v analognih mrežah pa v stotisoč bitih. Tudi povezovalne enote terminala na mrežo so v digitalni mreži glede na modem v analogni mreži enostavnejše in cenejše. Prve javne digitalne mreže so bile oblikovane že leta 1973 (Dataroute in Infodat v Kanadi, Dataphone v ZDA, Transdata v Braziliji, Digital Data Circuit na Japonskem). Splošna problematika podatkovnih mrež in protokolov je bila objavljena tudi v domači literaturi (12, 13, 14).

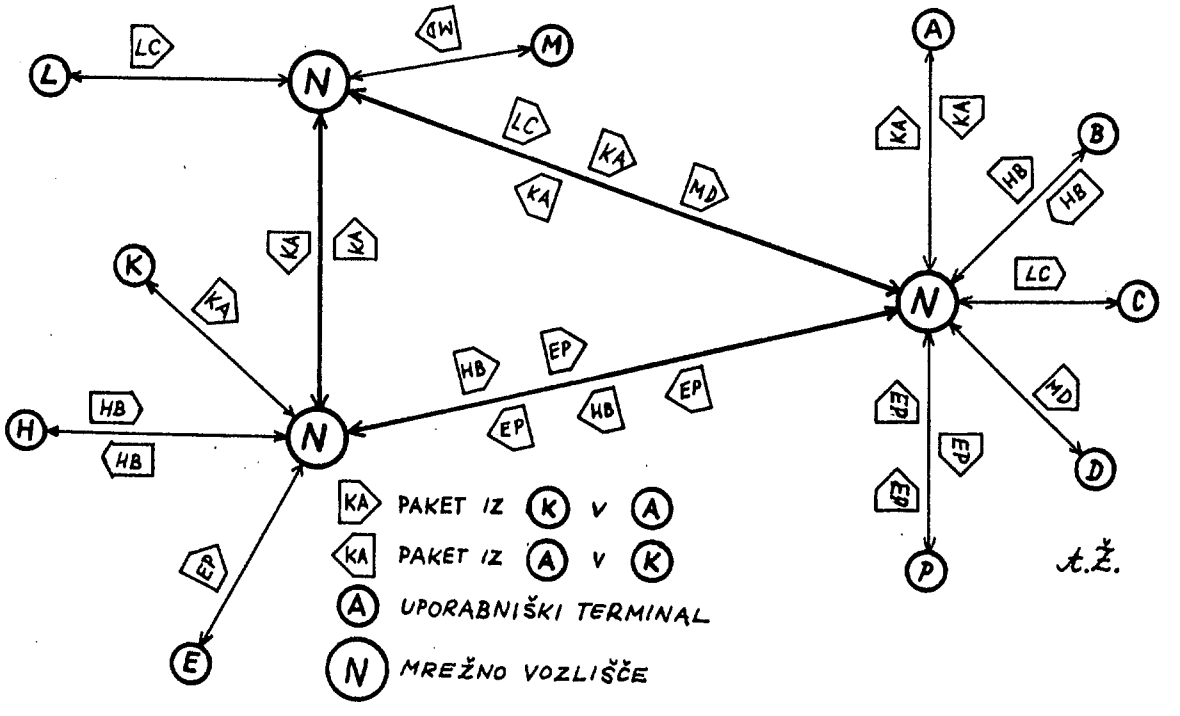
Na sliki 3 imamo še t.i. paketno podatkovno mrežo, iz katere je razvidno paketno pošiljanje podatkov s časovno delitvijo prenosnih poti med paketi v eni in v drugi smeri.

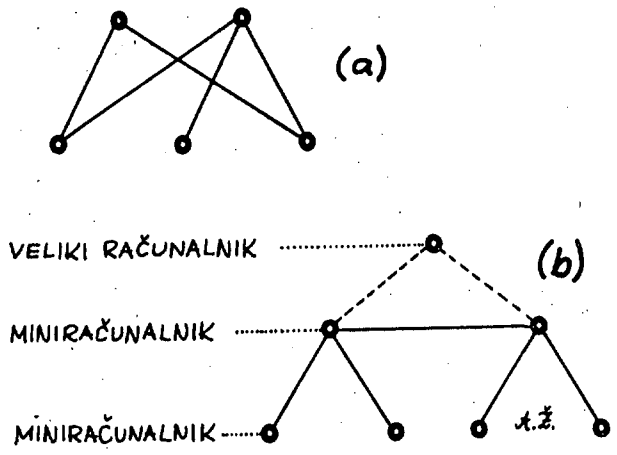
Dobro znana računalniška mreža je tudi DECNET, ki uporablja t.i. parno interakcijo (peer) (15). Mreža DECNET ni bila oblikovana le za miniračunalnike iz proizvodnje podjetja DEC (kot sta PDP-8 in PDP-11), marveč tudi za večje sisteme (kot je DECsystem-10). Namen te mreže je povezava računalnikov v pentljasto obliko na sliki 4(a) ali v hierarhično strukturo na sliki 4(b) ali v druge mrežne konfiguracije. Razvoj mreže DECNET je bil v zasnovi tipično navzgornji, tj. od uporabnika miniračunalnika do drugih uporabnikov z računalniki približno enakih moči, ki so se naprej povezovali medsebojno in kasneje v enoten veliki gostiteljski sistem. Nasproten navzgornjemu je navzdolnji razvoj mreže, ki ga je oblikovalo podjetje IBM pri zasnovi svojih hierarhično organiziranih računalniških mrež.

6. MIKRORAČUNALNIŠKI SISTEMI

Doslej smo izčrpno opisali visoki in srednji nivo računalniških sistemov, nismo pa se posebej ozrli na nižji računalniški nivo, ki ga predstavljajo mikroračunalniški sistemi. V podatkovnih mrežah, v porazdeljeni obdelavi podatkov in sploh pri smiselno grajenih velikih

SLIKA 3. Gibanje podatkovnih paketov in delitev prenosnih poti med paketi v paketni podatkovni mreži





SLIKA 4. Pentljasta povezava miniračunalnikov (a) in navzgornji razvoj povezave v računalniški mreži DECNET (b)

sistemih se na ravnini enostavne kompleksnosti pojavljajo mikroprocesorji, periferni procesorji in drugi posebni krmilniki. Tudi podatkovno oziroma računalniško mrežo moramo gledati kot trinivojsko zgradbo, z mega, mini in mikroravnino. Razvojni tokovi računalniških in predvsem informacijskih sistemov kažejo, da naj bi bili v prihodnje vsi računalniki, neglede na njihovo velikost, na določen način povezani v mrežo. To velja seveda tudi za t.i. inteligentne terminale ali terminale nasploh, ki se bodo lahko enakopravno vključevali v javne in zasebne podatkovne mreže.

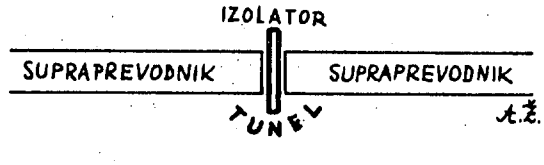
Decentralizacija obdelave in še posebej porazdeljena obdelava podatkov bo kot že rečeno, povzročila večjo uveljavitev mikroprocesorjev ter naprav in sistemov s temi procesorji. Tudi najmočnejši svetovni proizvajalci sistemov, kot je npr. podjetje IBM, se ne bodo odrekli samostojni mikroračunalniški tehnologiji in priročnim sistemom z mikroprocesorji. V računalniških mrežah bodo dobili mikrosistemi polno veljavo in to v obliki namiznih, domačih, uradniških, inženirskih in osebnih računalnikov.

Mikroračunalniški sistemi postajajo tudi čedalje bolj opremljeni z visoko sistemsko programsko opremo in tako znatno posegajo na področje klasičnih miniračunalniških aplikacij. Prav tu se kažejo boljša razmerja cena/zmogljivost za vrsto specifičnih uporab, kot so npr. razvoj programske opreme za mikroračunalnike s pomočje visokih programirnih jezikov, večuporabniški sistemi z mikroprocesorji za zbiranje podatkov in tudi za razvoj programske opreme in za procesiranje tekstov, sistemi za krmiljenje proizvodnih postopkov v realnem času itn. Na mikroprocesorje so bili upodobljeni tudi nekateri najuspešnejši operacijski sistemi, kot so npr. UNIX, ISIS, OASIS, CP/M, MDOS itn. Za različne mikroprocesorje obstajajo tudi interpreti in prevajalniki visokih programirnih jezikov, kot so PL/I (subset G), Pascal (polna različica), Fortran, Cobol, Basic, Lisp itn. Te težnje po razvoju učinkovitih opracijskih sistemov in prevajalnikov bodo znatno narasle pri novih 16-bitnih mikroprocesorjih, ki bodo v svojih sistemih uporabljali tudi največje winchesterske diske za več sto megazlogov.

Novi mikroprocesorji imajo že danes večmilijonske pomnilne prostore (do 48M zlogov), tako da virtualni pomnilniki zaenkrat sploh niso potrebni. Seveda pa so pri tako velikih pomnilniških obsegih potrebna dovolj močna pomnilniška integrirana vezja. Na področju pomnilniške tehnologije so dinamična pomnilna vezja (tipa RAM) za 16k bitov v enem vezju že standardna, pravkar pa se začenja redna dobava vezij za 64k bitov. Japonska podjetja prehitevajo ZDA z napovedjo skorajšnje prodaje vezij z 256k biti. Vsa ta pomnilniška vezja so zaradi nižjih delovnih hitrosti veliko boljše prilagojena uporabi v mikroračunalniških sistemih kot v sistemih izredno hitrih megaračunalnikov.

Mikroračunalniška tehnologija obsega danes 4-, 8-, 16- in 32-bitne centralne procesorske enote (CPE) in seveda tudi rezinaste enote, iz katerih je moč sestaviti zelo hitre in po številu bitov neomejene procesne enote. Hkrati se že pojavljajo ločene aritmetične procesne enote s 16-, 32- in 64-bitno strukturo (integrirana vezja 9511, 9512, 8089) za operacije fiksne in plavajoče vejice in za operacije +, -, *, /, sin, cos, tg, ctg, exp, log, arc, kvadratni koren itn. Posebej perspektivni so tudi t.i. pomnilniški krmilniki, ki upravljajo z velikimi pomnilniki, samodejno premeščajo programe in jih povezujejo brez dodatne programske opreme. Krmilniki za gibke diske generirajo vse potrebne signale ter imajo sprejemnike za zapise z enojno, dvojno in četverno gostoto. Prav tako so zanimivi tudi krmilniki za prikazovalne elektronke (CRT), ki jih lahko enostavno priključimo na sisteme z mikroprocesorji 6800, 68000, 6809, 8080A, 8085, 8086, 8088, Z80, Z8000 itn. Obstajajo tudi univerzalni CRT krmilniki v enem ali dveh integriranih vezjih in tudi v povezavi z DMA krmilniki.

Zaščita podatkov bo zahtevala naprave, ki imajo možnost šifriranja poslovnih podatkov pa tudi samih programov, ki se prodajajo na licenčni oziroma zaščiteni osnovi. V prihodnje naj ne bi oddajali le šifriranih sporočil, temveč naj bi bili tudi vsi podatki na diskih šifrirani. Šifriranje se danes opravlja s posebnim algoritmom (DES), ki ga lahko tudi sprogramiramo, največkrat pa uporabimo za to opravilo posebne integrirane šifratorje. Takšni šifrirni krmilniki obstajajo danes že za procesorske družine 8080A, 6800 in v samostojni obliki do konverzijske hitrosti 13M bitov na sekundo (Fairchild). Pri tem nismo posebej opisali še vseh drugih krmilnikov, kot so krmilniki za prekinitve, za neposreden pomnilniški dostop, za hitre diske in za komunikacijske protokole (SDLC, HDLC, X-25). Vsi ti krmilniki so enostavni povezljivi s CPE mikroračunalniških sistemov.



SLIKA 5. Josephsonov spoj je s tankim izolatorjem (elektronskim tunelom) prekinjeni supraprevodnik

7. SUPERRAČUNALNIŠKI SISTEMI

Superračunalniški sistemi naj bi imeli predvsem visoke operacijske hitrosti, velike centralne pomnilnike, nizko porabo energije in veliko kompleksnost materialne in programske opreme. Superračunalnik naj bi bil veliki računalniški sistem ali megaračunalnik, katerega uporaba bi bila smiselna predvsem na višjih ravninah podatkovnih mrež. in pri posebnih aplikacijah, kjer se zahteva visoka stopnja računalne kompleksnosti.

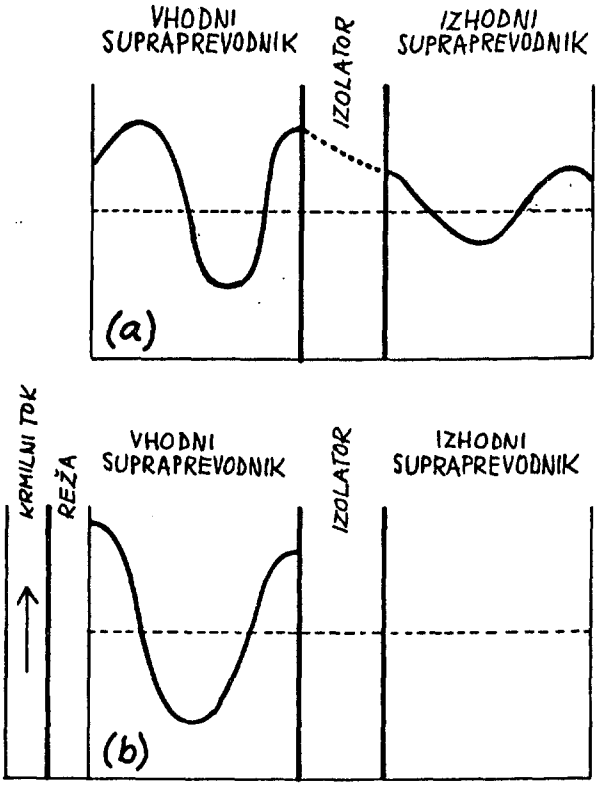
Tehnološka osnova za superračunalnik je t.i. Josephsonov spoj (16), ki je shematično prikazan na sliki 5. Supraprevodnost je lastnost, ko nekatere snovi zgubijo električno uporanost (R = 0) pri izredno nizkih temperaturah (v bližini absolutne temperaturne ničle -273 stopinj C) in je bila odkrita že leta 1911. Nobelov nagrajenec B.D.Josephson je v letu 1962 odkril spoj (efekt), ki omogoča gradnjo zelo hitrih preklopnih logičnih vezij. Karakteristika Josephsonovega spoja je prikazana s kvantno valovno funkcijo na sliki 6(a). Krmilne učinke pri tem spoju dosežemo s posebnim krmilnim prevodnikom, ko se prekine tunelski efekt z magnetnim poljem, ki ga povzroči tok skozi krmilni prevodnik na sliki 6(b).

Josephsonov spoj doseže preklopne hitrosti med 30 in 100 ps (japonski spoji so baje še hitrejši) ter ima zelo nizko porabo energije in je zaradi tega primeren za gradnjo zelo velikih, zgoščenih in hitrih računalnikov. Poskusni pomnilniki z Josephsonovimi spoji so dosegli čase dostopa pod 7 ns in podjetje IBM je pravkar preizkusilo prototipni računalnik s temi spoji pri taktni frekvenci 2,5 ns, kar je osemkrat hitreje od sistema 370/168.

Prva razvojna konfiguracija superračunalnika pri podjetju IBM je bila izdelana v kocki s stranico 10 cm, ter vsebuje CPE, pomnilnik tipa cache, 16M-zložni pomnilnik (128M bitov) in zmore (le) 70 milijonov ukazov na sekundo. Manj konzervativna projekcija pa predvideva v kocki s stranico 2,5 cm 64M- do 128M-zložni pomnilnik in milijardo ukazov na sekundo. Največji, danes proizvajani IBM računalnik ima 6M-zložni pomnilnik in taktni čas 50 do 80 ns s 3,5 milijoni ukazov na sekundo.

Pred 25 leti je John von Neumann naredil primerjavo med računalnikom in človeškimi možgani, ko je primerjal zmogljivost obstoječe računalniške tehnologije z biološko zmogljivostjo možganov. Ta primerjava je obsegala računanje na volumen ali produkt hitrosti elementarne enote (vezja ali nevrona) in njegove volumske gostote. Druga primerjava se je nanašala na energijo za računanje. V obeh primerih je von Neumann ugotovil, da so možgani v primerjavi z računalnikom zmogljivejši za faktor 10000.

Od tistih časov se je računalniška tehnologija tako izboljšala, da so mogoče enakopravne primerjave med računalnikom in človeškimi možgani, kot kaže tabela 1. Primerljivost je ustrezna upoštevaje celotno računalniško moč v odvisnosti od celotnega preklaplanja v časovni enoti. Toda Josephsonovi računalniki bodo v novejši človekovi zgodovini prvič kvalitativno presegli možgane, in sicer takole: za faktor 10000 v številu izračunov na enoto volumna, za faktor 1000 v številu izračunov na enoto energije in za faktor 100 v totalni računalni moči. Le na področju velikosti pomnilnikov bodo računalniki še vedno znatno zaostajali za močjo človeškega spomina (glej tabelo 1).



SLIKA 6. Potek kvantne valovne funkcije, ki kaže, kako prehaja tok skozi tunel (izolator), ko nimamo krmilnega toka (a) in ko s krmilnim tokom (magnetnim poljem) prekinemo prehod skozi tunel (b)

V generacijo superračunalnikov sodi tudi sistem CDC Cyber 205, ki zmore 800 milijonov operacij na sekundo in ima 16M-zložni glavni pomnilnik. Njegova aplikacija je zaenkrat eksotična, in sicer v iskanju novih, alternativnih virov energije, v izdelavi dolgoročnih vremenskih napovedi, konstruiranju letal, v obdelavi potresnih podatkov za naftno industrijo itn.

8. INFORMACIJSKI SISTEMI

Informacijski sistemi (IS) so uporabniško oblikovani računalniški in podatkovni sistemi in na sliki 7 je prikazana njihova groba delitev. Simboli M, m in μ označujejo računalniške arhitekture, ko imamo velike (mega-) mini- in mikroračunalniške sisteme. Pri tem je značilno, da moramo danes razlikovati velike, srednje in male informacijske sisteme, ki se na določen način povezujejo v širši (prostorski) informacijski sistem.

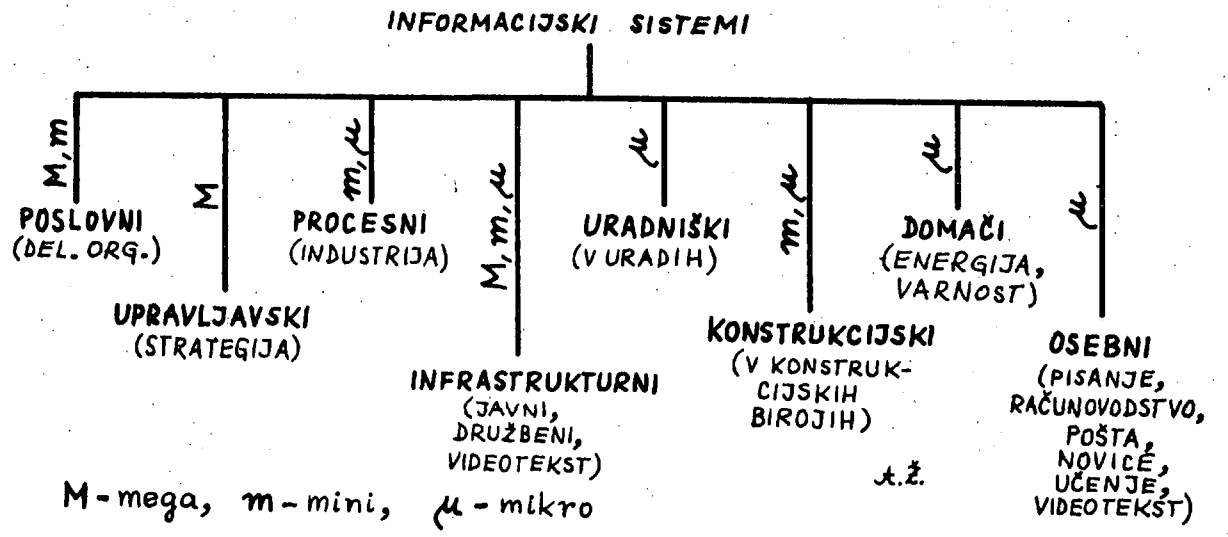
Poslovni IS so tipični za uporabo v delovnih organizacijah (tudi bankah), kjer se obdelujejo podatki, povezani s poslovanjem podjetja. Upravljavski IS so posebej usposobljeni poslovni sistemi in njihov namen je določanje strategijskih informacij, kot so poslovne, tehnološke in družbene odločitve, analiza stanja, strategija razvoja, planiranja, investicijske alternative itn.

Procesni IS so porazdeljeni sistemi v proizvodnji (ali drugih procesih), povezani s poslovnimi in upravljavskimi IS. Ti sistemi so

LASTNOST	1956	1980	1990 JOSEPHSON	ČLOVEŠKI MOŽGANI
VOLUMSKA GOSTOTA (VEZJA/ KUB. CM)	10^{-1}	10	10^{5}	10^{7}
MOČNOSTNE IZGUBE (W/VRATA)	10^{-1}	3.10^{-2}	10^{-6}	10^{-9}
HITROST (TAKTI/S)	10^{6}	10^{8}	10^{9}	10^{2}
PREKLAPLANJE/ KUB.CM/S	10^{5}	10^{9}	10^{14}	10^{9}
PREKLAPLANJE/ JOULE	10^{7}	3.10^{9}	10^{15}	10^{11}
POMNILNIŠKA GOSTOTA (BITI/KUB.CM)	5.10^{2}	5.10^{4}	5.10^{7}	10^{16}
TOTALNO ŠTEVILO ELEMENTOV V ENOTI (VEZIJ,NEVRONOV)	10^{5}	10^{5}	10^{5}	10^{10}
RAČUNALNIŠKA MOČ (PREKLOPI/S)	10^{11}	10^{13}	10^{14}	10^{12}

TABELA 1. PRIMERJAVA ZMOGLJIVOSTI RAČUNALNIKOV V LETIH 1956, 1980 IN 1990 IN ČLOVEŠKIH MOŽGANOV

SLIKA 7. Uporabniška delitev informacijskih sistemov s področji njihove uporabe in z uporabo računalniških arhitektur (M, m, μ)



posebej osposobljeni za delovanje v realnem času in njihove informacije so tudi signalne in ne samo podatkovne. Posebno novo področje uporabe računalniških sistemov so infrastrukturni IS, ki so povezani s terminali na javnih in zasebnih mestih ter so pod konceptualnim, vsebinskim in funkcionalnim nadzorom in usmerjanjem javnih in družbenih dejavnikov. Tu gre za široko podatkovno mrežo, ki nudi svoje usluge prek različnih komunikacijskig sredstev (telefon, televizija, brezžični prenos), masovno in različnim uporabnikom (podjetja, pošta, informativne službe, posamezniki).

Uradniški IS predstavljajo sisteme za avtomatizacijo vrste uradniških opravil, od procesiranja tekstov, razpošiljanja pošte, shranjevanja aktov do razdeljevanja dokumentov in slik. Vsi veliki svetovni proizvajalci računalniških sistemov računajo z znatnim plasmanom svoje proizvodnje na področje naprav in postopkov t.i. integriranega urada. Konstrukcijski IS so namenjeni inženirski in projektantski dejavnosti, kjer se izdeluje projektna in tehnična dokumentacija (tudi razvojna), so narejeni za generiranje tekstov, risb, izračunov itn.

Posebno novo področje so domači in osebni IS, kjer je predvidena uporaba mikroračunalniških sistemov. Domači IS naj bi predvsem skrbeli za vzdrževanje in delovanje doma, s posebnim poudarkom na smotrni porabi energije in zagotovitvi varnosti bivanja. Osebni IS se izredno hitro razvijajo za različne osebne usluge, kot je pisanje tekstov, osebno računovodstvo, poštne in bančne usluge ter novice prek posebnih javnih podatkovnih in uslužnostnih mrež. Osebni IS postajajo potreba in zahteva za različne osebne rabe in nagnenja, s ciljem, da zvišujejo osebno evidenco nad podatki in stanji ter produktivnost osebnega dela doma in na delovnem mestu.

9. Sklep

Pregled razvoja računalniških in informacijskih sistemov v tem članku obravnava le globalna izhodišča in lastnosti sistemov in njihove uporabe. Kot bistveni prihodnji sistem poudarja in razlaga podatkovno mrežo, ki naj bi postala cilj in možnost povezovanja izoliranih računalniških sistemov, terminalov in osebnih računalnikov. Virtualnost, multiprocesiranje in porazdeljena obdelava podatkov so tudi posebni pristopi prihodnjih, sodobno organiziranih računalniških in podatkovnih mrež.

Pregleda razvoja računalniških sistemov ne bi bilo moč sestaviti brez pomoči Strokovne knjižnice IBM, DO Intertrade v Ljubljani, ki se ji avtor iskreno zahvaljuje za pomoč.

Slovstvo

(1) W.D.Strecker: VAX-11/780: A Virtual Address Extension to the DEC PDP-11 Family, Computer Engineering, Digital Press, 1978.

(2) L.H.Seawright, R.A.MacKinnon: VM370 - A Study of Multiplicity and Usefulness, IBM Systems Jour. 18 (1979), Nr.1, 4-17.

(3) G.G.Henry: Introduction to IBM System/38 Architecture, IBM S/38 Tech Dev, 1978.

(4) J.G.Maisonrouge: The Information Industry, La Hulpe, Jan 30, 1979.

(5) E.W.Dijkstra: Co-operating Sequential Processes, Programming Languages, Ed. F.Genuys, Academic Press, 1968.

(6) C.A.R.Hoare: Monitors: An Operating System Structuring Concept, Comm. ACM 17 (1974), 549-557.

(7) N.Wirth: MODULA: A Language for Modular Multiprogramming, Software Practice and Experience 7 (1977), 3-35.

(8) A.L.Scherr: Distributed Data Processing, IBM Systems Jour. 17 (1978), Nr.4, 324-343.

(9) S.C.Kiely: An Operating System for Distributed Processing - DPPX, IBM Systems Jour. 18 (1979), Nr.4, 507-525.

(10) L.M.Branscomb: Computing and Communications - A Perspective of the Evolving Environment, IBM Systems Jour. 18 (1979), Nr.2, 189-201.

(11) J.R.Halsey, L.E.Hardy, L.F.Powning: Public Data Networks: Their Evolution, Interfaces, and Status, IBM Systems Jour. 18 (1979), Nr.2, 223-243.

(12) D.L.A.Barber: Vloga računalniških mrež se spreminja, Informatica 3 (1979), št.3, 28-35.

(13) D.L.A.Barber, T.Kalin: A Scenario for Computer Network Research, Informatica 3 (1979), št.2, 29-33.

(14) M.Kapus, A.P.Železnikar: Komunikacijski protokoli, Informatica 3 (1979), št.2, 50-57.

(15) P.E.Green: An Introduction to Network Architecture and Protocols, IBM Systems Jour. 18 (1979), Nr.2, 202-222.

(16) J.G.Maisonrouge: Die Herausforderung, IBM Nachrichten 29 (1979), Nr.246, 13-19.

(17) A.P.Železnikar, D.Novak: Mikroračunalnik IKE-1 s procesorjem I8086, Informatica 3 (1979), št.2, 3-13.

(18) E.Lerner: Superconducting Super Computers are on the Way, High Technology 2 (1980), Nr.3, 67-73.

(19) A.P.Železnikar, Sodobni tokovi razvoja računalništva, Delta informator 2 (1980), št.3, 6-11.

EVROPA IN NOVA INFORMACIJSKA TEHNOLOGIJA

UDK: 007:681.3(4)

EUROPE AND THE NEW INFORMATION TEHNOLOGY
March 1980, Commision of the European Communities, Directorate-General for Information, Rue de la Loi 200-B-1049 Brussels.

Telematika in informacijska tehnologija so besede, ki jih danes redko uporabljamo, jutri pa bodo tako pogoste, kot so telefon, televizija, obdelava podatkov ali satelit. Kot vse kaže, bo povezovanje teh različnih tehnologij v novo dinamično tehnologijo - telematiko - povzročilo globoke spremembe v človeški družbi.

Predvsem bo vsakodnevno življenje lažje. Elektronski denar se že danes uporablja in ob uporabi računalniških terminalov na ulicah, bi se gotovina lahko umaknila iz bančnih računov. Z uporabo telefona ali televizije s posebno testaturo, lahko že sedaj pregledujemo trgovske kataloge in celo naročamo ali pa dobivamo informacije o družbenih dejavnostih, lokalnih aktivnostih itd. Tudi pošto že lahko hitro prenašamo preko telekopirnih aparatov. Z medsebojnim povezovanjem sistemov za obdelavo teksta lahko tipkamo tekst na enem mestu (na primer doma), ga popravljamo brez ponovnega pretipkavanja ter istočasno prenašamo na drugo lokacijo. Končno nam bo uporaba mikroprocesorjev - resničnih miniračunalnikov - ki združujejo tisoče elektronskih vezij v enem paketu manjšem od glave vžigalice in se že masovno proizvajajo, omogočila širok spekter specializirane opreme, potrebne za telesno prizadete osebe, uporabo računalnikov doma, v avtomobilu itd. V jutrišnjem svetu bomo lahko avtomatično kontrolirali porabo goriva, varnostno razdaljo med avtomobili in podobno.

Širše vzeto bo dostop do informacij v vseh oblikah (pisani, slušnovidni itd.) mnogo enostavnejši, kar bo koristilo posameznikom, gospodarskim organizacijam in javnim službam, ki so pogosto obremenjene z goro podatkov.

Učinkovito odločanje zahteva hiter in enostaven dostop do vseh informacij, ki se nanašajo na določeno področje ter njihovo sintezo. Odločanje zahteva tudi hitro komuniciranje s partnerji, strankami, itd. Upravljanje z informacijami in njihova uporaba je ključni problem družbe. V določenem trenutku bodo najbolj razvite dežele prešle iz industrijske v informacijsko ekonomijo.

Telematika je skupen izraz za vse informacijske tehnologije, potrebne za tako spremembo : banke podatkov, računalniška oprema in še posebno prej omenjeni mikroprocesorji ter končno nova tehnologija prenosa informacij preko kablov iz optičnih vlaken in satelitov, ki bodo izrinili tradicionalna komunikacijska omrežja.

Taka tehnološka revolucija seveda ne more ostati brez resnih posledic:

Na ekonomskem področju:

Informacijska tehnologija še vedno predstavlja omejeno področje, ki pa naglo raste : okoli 15 % letno. Potrebe svetovnega trga so zelo velike in pogosto zelo elementarne. V ZDA je od leta 1940 do danes razmerje med zaposlenimi, ki so vključeni v razne oblike obdelave podatkov (za razliko od direktne proizvodnje) naraslo od 25 % na 45 %. Poleg samega razraščanja industrije, ki proizvaja komponente za novo informacijsko tehnologijo, prinaša ta industrija že sama po sebi globoke spremembe tudi na druga industrijska področja. Uporaba mikroprocesorjev bo postala običajna v avtomobilih, doma in v pisarnah, obdelovalnih strojih itd. Poleg tega bo naglo zniževanje cen že pripravljenih in programiranih sistemov pospešilo avtomatizacijske procese,

kjerkoli bo to možno. Kot gonilna sila industrijskega prestrukturiranja bo telematika omogočala tudi geografsko decentralizacijo določenih industrijskih aktivnosti (tekstilna industrija, elektronika, strojna industrija itd.) in celo redno delo doma (na primer določena pisarniška opravila).

Na družbenem področju:

Končna posledica naraščanja avtomatizacije bo zmanjšanje potreb za določenimi delovnimi mesti. Toda pod pogojem, da bo primerno usmerjena, bo ta revolucija prinesla tudi veliko število povsem novih opravil: načrtovanje, proizvodnja in vzdrževanje novih načinov avtomatiziranja, široka uporaba mikroprocesorjev v raznih napravah, v izobraževanju itd. Pokazala se bo potreba po ustvarjanju novih uslužnostnih dejavnosti za industrijo in za uporabnike te opreme. S spodbujanjem rasti manjših proizvodnih enot bo telematika odprla povsem nove možnosti industrijski iniciativi.

Na političnem in kulturnem področju:

Lažje in popolnejše zbiranje podatkov predstavlja problem za zavarovanje osebnih pravic v odnosu do javnih in privatnih organizacij. Vendar lahko avtomatizacija osebno svobodo tudi poveča. Da bi dosegli vse postavljene cilje, bo potrebno kvalitetno dvigniti tudi tehnično izobrazbo. Telematika bo ravno tako omogočila širši in direktnejši pristop do najrazličnejših virov informacij, vezanih na kulturne dejavnosti.

TRENUTEN POLOŽAJ: AKCIJA EVROPSKIH DRŽAV

Svetovno tržišče telematike zelo hitro raste, evropsko tržišče pa predstavlja njegov precejšen delež. Svetovno tržišče v letu 1978 je bilo tole:

Telekomunikacijska oprema:

Okoli 27 miljard EUA (Europen units of account) (1 EUA = okoli 0,63 £-menjalni tečaj z dne 4.o2.198o), od katerih odpade na Evropo 29 %; letna rast tega tržišča bo do leta 1987 nekje med 5 % do 8 %.

Sistemi za obdelavo podatkov (računalniki, periferija in servis):

Okóli 53 miljard EUA, od katerih odpade 26 % na Evropo; letna rast na tem področju pa bo okoli 17 %.

Integrirana vezja:

Okoli 5 miljard EUA, na Evropo odpade 19 % ; v letu 1985 bo doseženo 11 miljard EUA.

Ocenjuje se, da je tržna vrednost računalniških sistemov na svetu, ki se uporabljajo v administraciji in gospodarstvu, okoli 40 do 50 miljard EUA ; Evropski delež je okoli 25 %, vendar samo 17 % glede na dobavljeno opremo.

Čeprav je Evropa veliko tržišče, je njen delež v proizvodnji nezadovoljiv in je v nevarnem položaju, da bi se stanje poslabšalo. ZDA so glavni dobavitelj te opreme na svetovnem tržišču. To je v glavnem poseldica množičnih naročil zvezne vlade in NASA. Napredna ameriška polprevodniška in mikroprocesorska tehnologija se bo še nadalje ojačala z 200 miljoni $ v 6 letnem programu, ki ga je razpisalo obrambno ministrstvo za razvoj in produkcijo zelo hitrih integriranih vezij (VHSI).

Japonska je presenetila s spektakularno potjo od leta 1967, ko je objavila "Načrt informacijske družbe", ki je združeval vso podporo in spremljajoče ukrepe, ki so se v začetku kakazali predvsem v protekcionizmu. Ta politika je bila pred kratkim okrepljena z načrtom o produkciji izredno zgoščenih integriranih vezij. Z združevanjem tehnične kvalitete in konkurenčnih cen se Japonska industrija pripravlja na zavzetje svetovnega trga.

Poglavitne slabosti Evrope so:

- V proizvodnji računalnikov je največja evropska firma na 8.mestu, vendar je kljub temu le 1/12 velikosti IBM. Evropa je že izgubila bitko za trg zelo velikih računalnikov.

- V produkciji periferije (terminalov itd.) so obeti Evrope boljši, vendar je tudi tu opaženo nazadovanje zaradi napredka ameriških miniračunalnikov.

- Pri ključnih komponentah, posebno mikroprocesorjih, je stanje zelo vznemirjajoče, saj evropska industrija oskrbuje le 10 % svojega

trga, ki pa se zelo naglo širi.

- Pri bankah podatkov je situacija komaj kaj boljša. Poleg IRS banke podatkov, ki jo podpira evropska vesoljska agencija (ESA) in DIANE-EURONET omrežja, ki je usmerjeno na znanstveno in tehnično dokumentacijo (evropska komisija in nacionalni PTT tesno sodelujejo, da bi omrežje čimprej zaživelo), je komaj kaj bank podatkov na nivoju EGS.

Nasprotno pa je Evropa v mnogo boljšem položaju pri obdelavi podatkov in izdelavi programske opreme (software), kjer je njena industrija v polni ekspanziji ter pri telekomunikacijah, kjer oskrbujejo 30 % svetovnega trga (z dvema tretjinama firm evropskega porekla). Toda rast zelo obstoječega "telematičnega omrežja" bo ustavljena, če bo Evropa še nadalje zaostajala v razvoju celotne verige proizvodov. Stanje na poglavitnih področjih obdelave podatkov se v bližnji prihodnosti ne more bistveno spremeniti. Ni več niti zanesljivo, da bo samo napor vlad omogočil deveterici položaj, ki ji bo zagotovil trdne osnove na hitro se razvijajočih področjih, ko so periferija, miniračunalniki, ključne komponente, banke podatkov itd. Dominacija ZDA in Japonske na področju telematike, je za Evropo nevarna, ker lahko pomeni:

Dokončno zapravljeno možnost aktivnega sodelovanja na trgu prihodnosti;

Zmanjšanje konkurenčne sposobnosti doma in v svetu ;

Izgubo velikega števila delovnih mest in nevarne poseldice za svobodo in demokracijo. Če Evropa ne bo razvila svoje proizvodne kapacitete za telematične materiale, ki jih bo morala v vsakem primeru prej ali slej uporabljati, ne bo sposobna ponovno pridobiti delovnih mest, ki jih bo izgubila z napredkom avtomatizacije;

Naraščanje njene politične, tehnološke in kulturne odvisnosti.

Evropa se je še vedno sposobna boriti. Še vedno lahko najde prostor v revoluciji, ki bo ločevala družbe 21. stoletja. Potreben pa je enoten cilj: doseči eno tretjino svetovnega trga leta 1990. Da bi to dosegla, ima Evropa tri adute, ki pa jih lahko izigra samo v okviru skupne strategije deveterice.

Prvi faktor je obstoj domačega trga, velikega kot v ZDA, ki bo omogočal razvoj industrije, dokler se ne ukinejo nacionalne in tehnične ovire ter nacionalna urejanja in monopoli na področju telekomunikacij (kjer niso prav posebno pripravljeni na sodelovanje in povezvoanje).

Drugi faktor so vladna naročila deveterici, ki bi omogočila temu trgu, da bi imel koristi od evropskih uporabnikov in dala prednost evropskim proizvajalcem pri velikih projektih, novi proizvodnji in servisu. To se bo sicer razvilo v določeno diskriminacijo proti tujim proizvajalcev, vendar bo v korist in v okviru prave evropske strategije;

Tretji faktor je povečanje učinkovitosti nacionalnih razvojnih programov s sistematično koordinacijo, ki bo preprečevala nasprotja in podvajanje naporov ter nepotrebno konkurenco, rezultati raziskav pa bi morali biti dostopni vsem, da bi se kar največ povečal industrijski razvoj na evropskem nivoju:

- Nemčija ima pripravljena sredstva v višini 196 miljonov EUA za 4-letni program telematike, združenega s programi na prodročju računalniških komponent in tehnologije ter njenega vpliva na družbo ter končno projekt za direktno satelitsko televizijo;

- Francija je pripravila načrt računališke družbe (392 miljonov EUA v petih letih), povezanega z odločitvijo o izstrelitvi telekomunikacijskih in televizijskih satelitov;

- Italija šele razvija program razvoja elektronike v okviru svojega zakona o industrijski oživitvi in prestrukturiranju;

- Anglija pripravlja akcijo za podporo industriji mikroelektronskih komponent.

NAČRTOVANJE EVROPSKE STRATEGIJE

Evropska strategija na področju telematike, ki jo je predlagala evropska komisija, ni podprta s posebno centralizirano politiko, ki bi zahtevala velike finančne vire ter konventracijo kadrov. Njen namen je predvsem v spodbujanju dinamičnega dela in v povezo-

vanju različnih aktivnosti prek nacionalnih vlad, posameznih družb ter institucij deveterice.

Ključne točke evropske strategije so:

Telekomunikacije:

- narodno usklajevanje, osnovano na skupni poziciji deveterice, osnovnih karakteristik sistema bo omogočalo uvajanje integriranih digitalnih omrežij na evropskem nivoju, ki bodo omogočale prenos informacij (klasične oblike, kot so na primer telefoni, hkrati a tudi popolnoma nov nivo uslug) v numerični obliki, ki se bodo lahko obdelovali hitreje in v večjih količinah;

- prvi evropski informacijski sistem, ki bo povezoval institucije deveterice z vladami posameznih članic, bo olajšal prenos podatkov med javnimi institucijami in stimuliral razvoj nove opreme in servisa z industrijskimi in nacionalnimi telekomunikacijskimi omrežji ;

Sateliti:

TV programe bodo usmerjali direktno na hišne antene preko cele Evrope, hkrati pa bodo služili kot releji za prenos podatkov med organizacijami, njihovimi dislociranimi enotami ali partnerji. Ravno tako se bodo uporabljali za nadzor okolja ter sledili naravnim virom na kopnem in morju. Tudi akcijo evropske vesoljske agencije, ki ji dolgujemo uspešno izstrelitev evropske rakete Ariane ob koncu leta 1979, bo zaradi narave teh načrtov in obsega njihovega vpliva, zahtevala koordinacijo vseh evropskih naporov ter skupen pristop k njenemu vodenju. Zagotoviti si moramo predvsem standardizacijo in kompantibilnost osnovnih terminalov, razviti učinkovito uporabo daljinskih nadzornih programov itd.

Obdelava podatkov:

Program deveterice s pripravljenimi sredstvi v višini 25 miljonov EUA za obdobje 1979 do 1983 zahteva splošno akcijo (standardizacija, skupen trg, sodelovanje med razvojnimi centri, študij tehnologije in njenega vpliva na zaposlovanje, zaščita osebnih pravic in varovanje podatkov, zakonska zaščita računalniških programov). Deveterica bi morala vskladiti splošne standarde za vso opremo, da bi zagotovila enostavno izmenjavo informacij po letu 1983. Evropski program ravno tako zahteva podporo pri razvoju aplikativnih programov, podpiranje naporov uporabnikov pri koordinaciji na nivoju deveterice in odpiranju trga, ki bi ga lahko obvladovala evropska industrija.

Periferija (terminali itd.):

Deveterica bi morala omogočiti specializacijo manjših firm, če je potrebno s finančno podporo in koordinirano družbeno pomočjo ali s širšim evropskim sodelovanjem.

Odpiranje svetovnega trga evropskim proizvodom:

Našemu izvozu se odpirata dve možnosti: trg industrializiranih in mediteranskih dežel Afrike, Karibov in Pacifika, združenih seveda z evropskim trgom. Na področju uporabe satelitov in bank podatkov je potrebno posebno upoštevati potencial afriških uporabnikov.

Vendar pa ni dovolj začeti z raziskavami, industrijskim razvojem ali širjenjem tržišča. Tudi družbeno okolje moramo pripravljati na informacijsko tehnologijo, da bi zagotovili rezultate, ki jih tehnika omogoča ter zmanjšali tveganje v prehodnem obdobju.

Zaposlovanje:

Da bi preprečili težave in hkrati ustvarili ozračje zaupanja v inovacije, bo potrebno zelo previdno in kontrolirano uvajati novo tehnologijo:

- z dogovarjanjem;
- s periodičnim ugotavljanjem vpliva novih tehnik ;
- z evropskim povezovanjem ter združevanjem študija in razvoja;

Izobraževanje:

Priučevanje delavcev, uporabnikov in širše javnosti na novo tehnologijo, bi morali olajšati s :

- sistematičnim študijem bodočih potreb glede na področje in kvalifikacije (prvo poročilo bo izdelano leta 1981) ;
- uporabo evropskih socialnih fondov za izobraževanje in preusmeritev projektov na

elektronske tehnike;
- podpiranjem izmenjave izkušenj pri izobraževanju in uporabi nove tehnike v šolah;
- podpiranjem izmenjave izkušenj in organiziranje specializiranih seminarjev za vodstvene kadre v gospodarstvu in sindikatih

Zavarovanje svobode:

Zahodnoevropske države poskušajo usklajevati svoje zakone z njihovim prilagajanjem konvencij o zaščiti posameznikov pred računalniško obdelavo osebnih podatkov.

Če se telesa, ki trenutno presegajo meje deveterice, ne bodo zedinila o tekstu, okoli katerega se trenutno pogajajo, bo potrebna ločena akcija deveterice.

Informacijska tehnološka revolucija je v teku in je ni mogoče več ustaviti. Kot vsak tehnološki skok, je tudi ta poln možnosti in tudi tveganja. Evropa bo prenesla ta tveganja in izkoristila priložnost, ki se ji ponuja le, če bo na tem področju tesno sodelovala. Še vedno ni prepozno, vendar pa časa že primanjkuje.

JEDNA METODA EMULACIJE MEMORIJE MIKRORAČUNALA S MINIRAČUNALOM

N. BOGUNOVIĆ
I. MARIĆ

UDK: 681.3:519.685.8

INSTITUT „RUDJER BOŠKOVIĆ", ZAGREB

Opisana je metoda koja povećava efikasnost razvoja programa za mikroračunala na miniračunalu. Povezivanjem "host" i "target" sistema u logičku i funkcionalnu cjelinu, metoda omogućuje sklopovsku emulaciju PROM dijela memorije mikroračunala. Osnovna prednost ove metode pred drugim uobičajenim postupcima testiranja programa, očituje se mogućnostima verifikacije konačne i readresirane verzije mikroračunskog programa pomoću sistemskih resursa miniračunala.

A METHOD FOR THE MICROCOMPUTER MEMORY EMULATION BY A MINICOMPUTER: A method which offers the increased efficiency of the microcomputer program developing process by a minicomputer is described. Interconnecting the host and the target system into a logical and functional integral unit, the method enables in-circuit emulation of the PROM part of the microcomputer memory. The main advantage of this method over the other more conventional program debugging techniques, lies in the possibility to verify the final and readdressed microcomputer program version with minicomputer system resources.

UVOD

Efikasan razvoj programa za mikroračunala zahtijeva poseban sistem za razvoj jer je mikroprocesor u osnovi poluvodička komponenta bez sklopovsko-programske podrške koju u pravilu uvijek nalazimo uz mini i veća računala. Sistem za razvoj je kompleksan instrument čije se funkcije mogu implementirati u specijalnom mikroprocesorskom sistemu uz rezidentne sistematske programe ili se može koristiti računalo opće namjene i programski medjuprodukti. Oba ova opća pristupa imaju svoje pozitivne strane kao i nedostatke.

Većina specijalnih sistema za razvoj mikroračunala namjenjeni su isključivo jednoj mikroprocesorskoj familiji, budući da ih na tržište stavljaju proizvodjači poluvodičkih komponenata, integriranih krugova i mikroprocesora za potrebe razvoja njihovih familija mikroračunala /1/. Implementacija viših jezika je u tim sistemima nedovoljno razvijena. Primjena specijalnih sistema za razvoj, u uvjetima dinamičnog razvoja ove tehnologije, opravdana je samo u organizacijama koje mikroračunala uključuju u visokoserijske proizvode.

S druge strane, razvoj programa za mikroračunala na računarskim sistemima opće namjene, u osnovi pruža korisniku veću fleksibilnost. Na takvim sistemima moguće je u načelu, koristeći programske medjuprodukte, razvijati programe za više različitih mikroprocesorskih familija. Nedostaci se očituju u potrebi za detaljnim poznavanjem dva ili više sistema ("host" i "target") umjesto jednog, programski medjuprodukti su kompleksni i često se moraju razvijati na licu mjesta, a kao najvažnije, ne postoji nužna "on-line" veza do mikroračunarskog prototipa preko koje bi se mogla verificirati integracija programsko-sklopovskih rješenja.

U ovom radu opisat će se metoda koja povezivanjem miniračunarskog sistema za razvoj i mikroračunarskog prototipa, emulacijom dijela memorije mikroračunala, omogućuje testiranje programa za mikroračunala koristeći pri tom sistemske resurse miniračunala. Time se otklanjaju neki nedostaci razvoja programa za mikroračunala preko medjuprodukata.

Emulacija je proces medjusobne interakcije sistema za razvoj i mikroračunarskog prototipa u kojem se pojedini resursi oba sistema medjusobno zamjenjuju. CPU, memorijski sklopovi i ulazno-izlazni sklopovi prototipa te programske rutine, u početku su razvoja fizički pridjeljeni sistemu za razvoj. Kako napreduje implementiranje pojedinih programskih rutina ti se resursi postepeno vraćaju u mikroračunarski prototip. U završnoj fazi svi se sklopovski i programski resursi nalaze i fizički u prototipu, a sistem za razvoj samo prati odvijanje programa uz neke osnovne kontrolne funkcije. Komercijalni sistemi, koji imaju mogućnost emulacije, dozvoljavaju samo takvo pridjeljivanje u kojem se CPU jedinica zadnja vraća prototipu tj. nema emulacije memorijskih sklopova ili ulazno-izlaznih sklopova ako se istovremeno ne emulira i mikroračunarska CPU jedinica.

Smatramo da je to ozbiljan nedostatak komercijalnih razvojnih sistema jer je CPU jedinica jedan od najkompleksnijih sklopova koji se moraju provjeriti u realnom okolišu prototipa u najranijoj fazi razvoja. Pored toga, bitno je da se komunikacija CPU jedinice s ulazno-izlaznim sklopovima odvija u realnom vremenu zbog kritičnih vremenskih odnosa pri zahtjevima za prekid, direktnim pristupom u memoriju (DMA) ili zaustavljanju (HLT). Ako se CPU prototipa fizički ne nalazi u sistemu za razvoj, spojni putevi navedenih

signala nisu realni pa su i vremenski odnosi neadekvatni stvarnom radu, a javljaju se i poteškoće pri korištenju osnovnog takt oscilatora iz prototipa. Ti se nedostaci djelomično otklanjaju smještanjem CPU prototipa za vrijeme emulacije vrlo blizu njegovom stvarnom mjestu u prototipu /2/.

Metoda emulacije memorije, koja je dana u ovom radu, dozvoljava pridjeljivanje ROM memorijskih sklopova prototipa s pripadnim sadržajem sistemu za razvoj uz fizičku prisutnost CPU jedinice u prototipu. Opisani sistem izveden je s miniračunarskim razvojnim sistemom baziranim oko miniračunala PDP-8 tvrtke "DEC" i mikroračunarskim prototipom na bazi mikroprocesora 8080 A tvrtke "Intel" /3/. Iako je opisan specifičan mini-mikro par iz razmatranja će se uočiti da metoda ima opću primjenu.

KONCEPCIJA EMULACIJE PROMA

U velikoj većini sistema s mikroprocesorom konačne verzije programskih rutina su trajno smještene u ROM dijelu memorije. Sukladno ranijim razmatranjima vrlo korisna odlika sistema za razvoj sastoji se u mogućnosti emulacije pojedinih PROM sklopova s pripadnim sadržajem uz zadržavanje svih ostalih resursa u prototipu. Realizacija ove ideje olakšana je činjenicom da postoji jedna opće prihvaćena familija EPROM sklopova koja se gotovo bez iznimke koristi u realizaciji prototipa ili sistema manjih serija. To je familija EPROM sklopova tipa 1702, 2708, (2758), 2716, 2732, tj. kapaciteta od 256 do 4 K byte-a. Zamišljeno je da sistem za razvoj generira, kao rezultat operacije krosasembliranja mašinski kod mikroračunala u jedinstvenom modulu čija je veličina 256 do 4 K byte-a, odnosno odgovara EPROM-u u koji će se konačno i upisati taj dio programa. Tako dobiveni modularni dio programa ostaje u radnoj memoriji miniračunarskog sistema za razvoj. Pristup do tog dijela programa, mikroračunarski prototip može ostvariti preko posebne interface jedinice koja s jedne strane ima priključak do ptototipa preko podnožja predvidjenog za pripadni PROM, a s druge strane do miniračunala. Bilo bi poželjno kad bi se komunikacija izmedju prototipa i emuliranog dijela memorije mogla ostvariti u realnom vremenu.

Najkraće vrijeme do pristupa sadržaju memorije miniračunala ostvarilo bi se direktnim pristupom (DMA) u kojem vanjska logika kontrolira prijenos podataka. Miniračunalu se mora specificirati lokacija memorije iz koje želimo pročitati podatak, zatim osigurati vanjski registar za prihvat podatka te logičke signale koje odredjuju smjer prijenosa, tip prijenosa (jednociklusni ili multiciklusni) i signal zahtjeva za prijenos. Po prijemu signala zahtjeva za DMA u času t_r, miniračunalo završi tekuću instrukciju u času t_c, istovremeno daje signal o prihvaćanju adrese te prelazi u DMA ciklus trajanja 1,5 μs tokom kojega tj. najranije 450 μs nakon početka, vanjska jedinica može pročitati podatke.

Ova analiza pokazuje da miniračunalo nije u stanju generirati željene podatke u realnom vremenu rada mikroračunala jer se podaci pojavljuju nakon t_a vremena gdje je:

$$t_a(\mu s) = (t_c - t_r) + 0{,}45$$

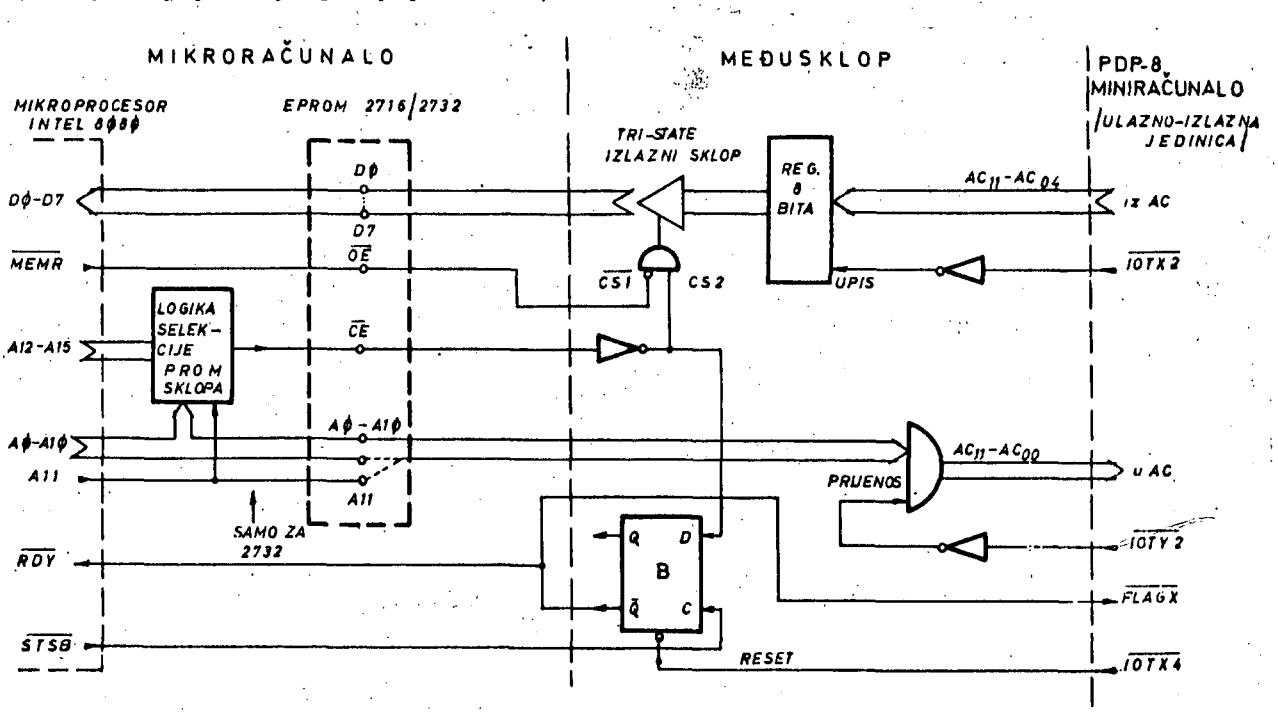
U najgorem slučaju to je trenutak početka najduže instrukcije te prvi član iznosi prema podacima iz /4/:

$$t_c - t_r = 4{,}5\ \mu s$$

odnosno:

$$t_{a_{max}} \doteq 5\ \mu s.$$

Na temelju iznesenoga odlučeno je da se s miniračunarske strane koristi uobičajeni uvjetno programirani prijenos podataka koristeći signal za preskok (skip). Programirani prijenos podataka ujedno omogućuje korisniku intervencije u tok dijela mikroračunarskog programa koji koristi emulirani PROM.



Sl. 1. Sklopovska blok shema sistema za emulaciju

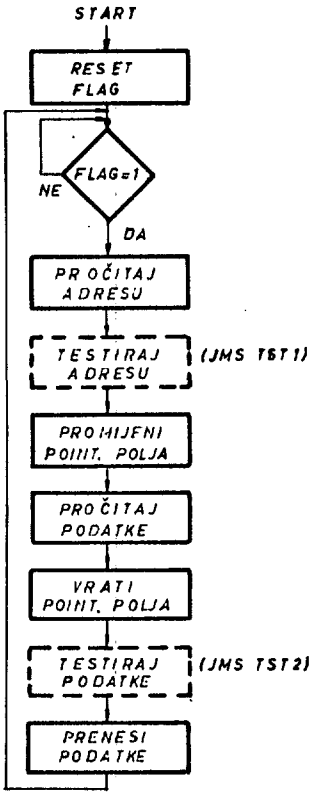
SKLOPOVSKO-PROGRAMSKA STRUKTURA IZVEDENE EMULACIJE

U daljem razmatranju ograničit ćemo se na implementaciju sistema za emulaciju EPROM memorije tipa 2716 i 2732 jer se moguće izvedbe emulacije za ostale EPROM-ove iz spomenute familije razlikuju gotovo neznatno. Na slici 1 prikazana je sklopovska blok shema cjelokupnog sistema. Pretpostavimo da je mikroračunalo implementirano s mikroprocesorom tipa 8080 iako se navedeni EPROM sklopovi mogu koristiti i s većinom ostalih mikroprocesora.

Neka na početku nekog instrukcijskog ciklusa mikroprocesor zahtijeva sadržaj emuliranog EPROM-a. Za vrijeme prvog elementarnog stanja T_1 mašinskog ciklusa, preko logike za selekciju PROM sklopa odabere se signalom CE odabrani PROM. Tokom drugog stanja T_2 mašinskog ciklusa, generira se signal STSB koji preko bistabila B generira signal RDY. Signal RDY zahtijeva čekanje i mikroprocesor u trećem stanju mašinskog ciklusa ulazi u stanje čekanja T_w. Istovremeno bistabil B generira i signal FLAGx koji miniračunalu daje znak da se traži prijenos. Adresa traženog podatka upiše se u miniračunalo, a odgovarajući 8 bitni podatak iz miniračunala upiše se u registar medjusklopa. Bistabil se vraća u početno stanje signalom IOTX4 te nestaje zahtjev RDY pa mikroračunalo izlazi iz stanja čekanja. U slijedećem stanju T_3, generira se signal čitanja MEMR koji preko EPROM podnožja (OE i CS 1) omogućuje izlaz podataka iz registra medjusklopa na sabirnice mikroračunala. Time je jedan ciklus emulacije završen.

Prije prijelaza na rutinu za emulaciju u memoriji miniračunala postavlja se polje 8 bitnih podataka koji odgovaraju sadržaju PROM-a. Emulacija EPROM-a 2716 i 2732 traži 2 K odnosno 4 K byte-a te je za tu namjenu rezervirano cijelo polje miniračunala PDP-8. Podaci koji se u to polje upisuju iz sistemske memorije, predstavljaju rezultat krosasembliranja dijela izvornog programa. Posebni dijelovi sistemskih programa miniračunala, koji se ovdje neće posebno navoditi, omogućuju automatsko odredjivanje veličine i formiranje polja podataka u skladu s emuliranim PROM-om /5/. Programska rutina za emulaciju izradjena je vrlo jednostavno s jednom petljom, koristeći minimalan broj instrukcija kako bi se traženi podaci prezentirali mikroračunalu u najkraćem mogućem vremenu. Dijagram toka te rutine je dan na slici 2.

Posebnu pažnju treba obratiti na opcije skokova na subrutine TST1 i TST2 za vrijeme ciklusa emulacije. Naime, na taj način je moguće dinamički ispitivati tok mikroračunarskog programa i eventualno preko drugih dijelova sistema za razvoj /3/ utjecati na njegovo odvijanje. Razumljivo je da se može organizirati i samo pasivno praćenje toka mikroračunarskog programa uz prikaz relevantnih podataka na terminalu miniračunala. Trenutak prikaza može biti odabran sasvim proizvoljno, npr. nakon n-tog pristupa m-toj adresi. Bez skokova na subrutinu TST1 i TST2, rutina emulacije traje 20 do 25 μs te je toliko produžen i mašinski ciklus prijenosa podatka iz emulirane memorije. Jasno je da su ostali vremenski odnosi, a posebno interakcija CPU jedinice s ulazno-izlaznim sklopovima, realni.



Sl. 2 Dijagram toka emulacije

ZAKLJUČAK

Prikazana je metoda koja u okviru integralnog sistema za razvoj programa za mikroračunala, emulacijom PROM dijela memorije omogućuje jednostavnu verifikaciju programa. Metoda je implementirana u sistem za razvoj baziran na miniračunalu, te se odlikuje svim prednostima kako razvojnog sistema univerzalne namjene tako i "on-line" razvoja. Nasuprot komercijalnim sistemima, koji mogu emulirati memoriju prototipa samo ako se emulira i CPU, ova metoda ne pretpostavlja takve i slične uvjete.

Odabrana familija EPROM sklopova, omogućuje vrlo široku primjenu prikazane metode emulacije jer se navedeni sklopovi mogu koristiti uz većinu 8 i 16 bitnih mikroprocesora. Sklopovska rješenja i struktura pripadnih sistemskih programskih rutina emulacije, pokazuju jednostavnost izvedbe i primjene. Priključak na mikroračunarski prototip ostvaruje se preko podnožja za emulirani EPROM i s dvije dodatne linije kontrolnih signala (u slučaju rada s mikroprocesorom 8080 to su RDY i STSB). Time se na prototipu izvrše minimalne izmjene.

Navedene prednosti ove metode emulacije povoljno utječu na efikasnost razvoja mikroračunarskih programa zadržavajući istovremeno fleksibilnost tj. mogućnost rada s raznim mikroprocesorskim familijama.

Nedostatak metode je izražen u usporavanju mašinskog ciklusa mikroprocesora za vrijeme čitanja podataka iz emulirane memorije. To je usporavanje vezano uz rad sistema za razvoj, u ovom slučaju miniračunala PDP-8 starije generacije. Smatramo da bi se s računalom modernijih izvedbi sistem mogao koncipirati tako da se rad mikroračunala za vrijeme emulacije približi realnom vremenu.

REFERENCE:

1. Microprocessor Development Systems, Product Focus, Electronic Engineering, Mart, 1980, str. 110-149.
2. M.Y. Yen: Fast Emulator Debugs 8085 - based Computers in Real Time, Electronics, July 21, 1977, str. 108-112.
3. N. Bogunović, M. Jelavić: Miniračunarski sistem za "on-line" razvoj mikroračunarskih programa, Zbornik del, Informatica 79, Bled, oktobar 1979, str. 1-210.
4. Digital Equipment Corp., Small Computer Handbook, PDP-8/I edition, 1971.
5. N. Bogunović, L. Cucančić: Sistem za čitanje, upis i prikaz procesa upisa u PROM, Zbornik radova, Informatica 78, Bled, oktobar 1978, str. 2-206.

RIGHT-TO-LEFT CODE GENERATION FOR ARITHMETIC EXPRESSIONS

DUŠAN M. VELAŠEVIĆ

ELEKTROTEHNIČKI FAKULTET, BEOGRAD

UDK: 681.3:512

ABSTRACT

The paper presents a method of right-to-left code generation for arithmetic expressions given in postfix notation. The number of generated instruction lines is equal to that obtained from the binary tree structure. However, the right-to-left code generator is faster and requires less memory with respect to the generator from the binary tree. Such results were obtained by introducing a vector, called vector-generatrice, assigned to every postfix string. A translation grammar which generates the postfix string and its associated vector-generatrice is defined. An experiment has been performed over a set of arithmetic expressions to compare certain characteristics of the right-to-left code generator to those of the generator from the binary tree.

GENERISANJE KODA S LEVA U DESNO ZA ARITMETIČKE IZRAZE. - U radu je izložena jedna metoda za generisanje simboličko-mašinskog koda s leva u desno za aritmetičke izraze predstavljene u postfiksnoj notaciji. Broj generisanih linija instrukcija isti je kao u slučaju binarnog stabla. Medjutim, prikazani generator je brži od generatora koda iz binarnog stabla i zahteva manje memorijskog prostora. To je postignuto uvodjenjem posebnog vektora, nazvanog vektor-generatrisa, i njegovim pridruživanjem svakom postfiksnom nizu. Definisana je translaciona gramatika koja generiše postfiksni niz i vektor-generatrisu. Obavljen je eksperiment nad skupom aritmetičkih izraza radi uporedjenja nekih karakteristika predloženog generatora i generatora koda iz binarnog stabla.

INTRODUCTION

There are various methods of code generation for arithmetic expressions[1,2,3,4,5,6,7,8,9] but two of them are mostly discussed: 1) generation from the binary tree, and 2) generation from the postfix string. The binary tree structure makes possible generation of shortest code (excluding any optimization) and minimizes the number of used temporary variables. The price paid for it is the lower speed of the code generator. On the other hand, the transformation of arithmetic expressions to postfix strings and the code generation are very simple. However, the code generated from postfix strings is inferior to the code obtained from the binary tree with respect to the number of instruction lines and the used temporary variables..

An experiment has been performed over a set of postfix strings and corresponding binary trees to examine the difference in length of generated code according to Eq. (1):

$$D_L = \frac{L_{ps} - L_{bt}}{L_{bt}} \qquad (1)$$

where

L_{ps} - length of code generated from postfix string,
L_{bt} - length of code generated from binary tree,
D_L - relative difference in length of generated code.

It was found that D_L has brought about 0 - 0.4. This fact becomes significant in highly repetitive loops with a great amount of calculations. The code obtained from the binary tree structure is shorter because the whole arithmetic expression is in such a form which provides the alternatives in particular points during the code generation[6]. There are no such alternatives in postfix string which is scanned strictly sequentially thus eliminating any possibility of economizing in code generation.

The second comparison is made to investigate the difference in time necessary to perform the syntax

analysis simultaneously with transformation to postfix string, according to Eq. (2):

$$D_T^S = \frac{T_{bt}^S - T_{ps}^S}{T_{ps}^S} \qquad (2)$$

where

T_{ps}^S - time necessary to perform syntax analysis and transformation to postfix string,

T_{bt}^S - time necessary to perform syntax analysis and transformation to binary tree,

D_T^S - relative difference in time spent to syntax analysis and transformation.

D_T^S is found by dividing the set of arithmetic expressions in subsets each comprising arithmetic expressions of the same length and then evaluating the average D_T^S for each subset. The results obtained have shown that such D_T^S is independent of length and brings about 0.15.

The next comparison is made to investigate the difference in time necessary to realize the code generation, according to Eq. (3):

$$D_T^C = \frac{T_{bt}^C - T_{ps}^C}{T_{ps}^C} \qquad (3)$$

where

T_{bt}^C - time necessary to perform code generation from binary tree,

T_{ps}^C - time necessary to perform code generation from postfix string.

Applying the same procedure as to D_T^S, it was found that D_T^C is also independent of length of arithmetic expressions and brings about 0.33. A total relative difference in time domain is given by Eq. (4):

$$D_T = \frac{(T_{bt}^S + T_{bt}^C) - (T_{ps}^S + T_{ps}^C)}{T_{ps}^S + T_{ps}^C} \qquad (4)$$

Because D_T^S and D_T^C are independent of length of arithmetic expressions, the results obtained for D_T possess the same property. The value of D_T brings about 0.24.

The memory requirement for the transformation to binary tree was 50 % higher and approximately the same for the phase of code generation when compared to the implementation with the postfix string.

The basic idea introduced here is to reduce the code generation time for arithmetic expressions achieved for the binary tree but retaining the same length of the code. This idea was motivated by the results of the investigation of distribution of executable statemets[10]. It was found that the assignment and IF statements have together over 60 % of static distribution[10].

VECTOR-GENERATRICE

As the first step toward the goal, we associate with every postfix string a vector, called vector-generatrice (VG), obtained from the following simplified translation grammar for the arithmetic expressions:

< start expression > → { INIT1} {INIT2} < expression> $ {$} {INC3}
< expression > → < expression > + < term > {+} {INC1}
< expression > → < expression > - < term > {-} {INC1}
< expression > → - < term > {⊥} {INC1}
< expression > → + < term >
< term > → < term > * < primary > {*} {INC1}
< term > → < term > / < primary > {/} {INC1}
< term > → < primary >
< primary > → < variable > {I} {COMP} {INC2}
< primary > → < unsigned number > {N} {COMP} {INC2}
< primary > → (< expression >)

The function designator and the operation ↑ are not included in the grammar only for the sake of clear and concise presentation of the algorithm. They do not influence in any case the final conclusion on the characteristics of the algorithm.

The activity sequence generated by this grammar consists generally of the following elements:

- input symbols: I, N, +, -, *, /, (,), $
- output symbols: I, N, +, -, ⊥ (unary minus), *, /, $
- action routines: INIT1, INIT2, INC1, INC2, INC3, COMP

The action routines do the following:

INIT1

The routine initializes to zero the space allocated to VG.

INIT2

The routine initializes a pointer K pointing to the first component of VG:

$$K = 1 \qquad (5)$$

and sets

$$VG(K + 2) = 1.$$

INC1

The routine performs the operation

$$VG(K + 1) = VG(K + 1) + 1 \qquad (6)$$

after outputting one of symbols: +, -, ⊥, *, /.

INC2

The routine performs the operation

$$VG(K) = VG(K) + 1 \qquad (7)$$

after outputting the symbols I or N.

INC3

The routine terminates the activity sequence by performing the following operations:

$$K = K + 3$$
$$VG(K) = 1$$
$$VG(K + 1) = 0 \qquad (8)$$
$$VG(K + 2) = VG(K - 1) + VG(K - 2) + VG(K - 3)$$

after outputting the symbol $.

COMP

The routine compares the class of the previously outputted symbol to the current one, I or N, for K ≠ 1. If they differ in class, the routine does the following:

$$VG(K + 5) = VG(K) + VG(K + 1) + VG(K + 2)$$
$$K = K + 3 \qquad (9)$$

Otherwise, no action is performed.

As a result of the applied translation grammar, one obtains a standard postfix string with an associated vector VG. There is a very simple practical explanation of VG. Let be given an arithmetic expression E_i in infix notation which is transformed to postfix string E_p:

$$E_p = a_1a_2a_3 \cdots a_mb_1b_2 \cdots b_ja_{m+1}a_{m+2} \cdots$$
$$\cdots a_{m+i}b_{j+1}b_{j+2} \cdots b_{j+k} \cdots$$
$$\cdots a_{m+n}a_{m+n+1} \cdots a_{n+M}b_{j+1}b_{j+1+1} \cdots$$
$$\cdots b_{j+J} \qquad (10)$$

where $a_1a_2a_3 \cdots a_m \cdots a_{m+M}$ represent the operands (or their addresses) and $b_1b_2b_3 \cdots b_j \cdots b_{j+J}$ the operators (or their indices in the delimiter table). The dollar sign $ denotes the end of the string, i.e.

$$b_{j+J} = \$ \qquad (11)$$

Using Eq. (10), we can generate VG in the following way. Firstly, we define the substrings of E_p. A substring is the sequence of all successive operands or operators. If the number of operand substrings is N_1, the number of operator substrings N_2 must be

$$N_2 = N_1 \qquad (12)$$

The total number of substrings N_3 (including the endmarker as an operand substring) is

$$N_3 = 2N_1 + 1 \qquad (13)$$

By assigning to each substring of operands a successor string of operators, one obtains a pair of connected substrings CS_l, where l is its ordinal number. The number of pairs N_{CS} is

$$N_{CS} = N_3/2 = N_1 \qquad (14)$$

For each pair of connected substrings we create a 3-tuple (N_l^O, N_l^B, P_l), where

N_l^O - number of operands in CS_l, i.e., $VG(K) = N_l^O$; (Eq. (7))

N_l^B - number of operators in CS_l, i.e., $VG(K + 1) = N_l^B$; (Eq. (6))

P_l - pointer to start of CS_l in E_p, i.e., $VG(K + 2) = P_l$; (Eq. (9))

VG is formed now by associating all 3 - tuples of connected substrings by their ordinal numbers into a vector of $(N_{CS} + 1) \times 3$ dimension. The endmarker is considered as an operand substring connected to a dummy operator substring.

RIGHT-TO-LEFT GENERATION

The term *right-to-left code generation* designates the code generation from the right end of E_p. One of the key points in introducing the vector-generatrice is just to enable the right-to-left code generation. The algorithm is based on the following principles:

- The elementary unit for code generation is the pair of connected substrings CS_l.
- The code generation from an elementary unit is the same as for any standard postfix string.
- The code is generated until $N_l^O = 1 \vee N_l^B = 0$.
- When the current CS_l becomes temporary inactive, i.e., $N_l^O = 1 \vee N_l^B = 0$, the next $CS_{l'}$ is taken, where $l' = l + 1 \vee l' = l - 1$ respectively.
- The separate stack for operands is not needed. Instead, the space for E_p itself is used for N_{CS} stacks, each preset already with the operand substring in the phase of transformation to E_p.
- The transition from one CS_l to another $CS_{l'}$ is performed by means of P_l, given by 3-tuple for CS_l.

Before we proceed to detailed presentation of algorithm, we shall explain the procedure for code generation from an elementary unit, CS_l. Let CS_l be given by

$$CS_l = a_1^l a_2^l a_3^l \cdots a_m^l b_1^l b_2^l b_3^l \cdots b_n^l \qquad (15)$$

The space occupied by the operand substring $a_1^l a_2^l a_3^l \cdots a_m^l$ has to be considered as a stack preset with $a_1^l a_2^l a_3^l \cdots a_m^l$. Starting from the top element of the stack, $T.OPD(a_m)$, and the first operator in the connected operator substring (b_1^l), we generate the following assembly code:

```
        LDA  T,OPD
        NEG
```

if b_1 is the unary operator (++), or

```
        LDA  BT,OPD
        OPT  T,OPD
```

if b_1 is a binary operator (+, -, *, /), where

BT.OPD - designates generally the below-the-top element of the stack.

OPT - symbolic representation of operators (ADD, SUB, MUL, DIV, NEG).

If any of succeeding operators is unary, an NEG is generated. Otherwise, the operator is checked for commutativity. If it is a commutative one, an instruction line

```
        OPT     T,OPD
```

is generated. The noncommutative operator requires the generation of the code line

```
        STA     T
```

with

```
T.OPD ← 'T
```

and then

```
LDA  BT.OPD
OPT  T.OPD
```

where T is a temporary variable. The code generation is continued in this way until $N_l^O = 1 \vee N_l^B = 0$. Then, the next $CS_{l+1 \vee l-1}$ is taken.

```
procedure code_generator (EP, VG, IL)
comment: The procedure code-generator generates the
        assembly code for arithmetic expressions
        starting from the right end of the postfix
        string with an associated vector-generatrice.
        EP - postfix string,
        VG - vector-generatrice
        IL - generated instruction line,
        NCS - number of elementary units,
        NZO - number of operands in the current
              elementary unit,
        NZB - number of operators in the current
              elementary unit,
        CSl - current elementary unit,
        CSlPLUSONE - elementary unit right to the cur-
                     rent one,
        CSlMINUSONE - elementary unit left to the
                      current one,
        l - pointer to the beginning of a 3-tuple in
            VG,
        lc - pointer to the leftmost 3-tuple currently
             handled,
        OPT - symbolic representation of operators
              (ADD, SUB, MUL, DIV, NEG),
        OPT1 - standard representation of operators
               (+, -, *, /, ++) in the operator
               substring,
        T.OPD - top element of the stack for CSl,
        BT.OPD - below-the-top element of the stack
                 for CSl,
        T - symbolic representation of temporary
            variable;
begin
        lc:=(NCS-1)*3+1;
        NZO:=VG[lc];
        NZB:=VG[lc+1];
        if NZB=0 then begin
                        generate_code_line_LDA_T.OPD ;
                        goto exit;
                      end
        else L1: l:=lc; if OPT1 = '++ 'then begin
                              generate_code_lines_ LDA_
                              T.OPD_and_NEG; goto L5;
                                          end
        else if NZO = 1 then
                L2: begin
                    comment: terminate code gene-
                             ration from an elemen-
                             tary unit CSl and
                             access the next one
                             CSlMINUSONE;
                    lc:=lc-3; NZO:=
                    =VG[lc] NZB:=
                    =VG[lc+1]:
                    goto L1;
                end
        else
           L3:begin
                generate_code_line_LDA_BT.OPD;
              L4:generate_code_line_OPT_T.OPD;
                NZO:=NZO-1;
              L5:NZB:=NZB-1;
              end;
        if NZB = 0 then goto L6;
        else if OPT1 = '++'then begin
                        generate_code_line_NEG;
                        goto L5;
                            end
        else if NZO = 1 then begin
                        generate_code_line_STA_T;
                        T.OPD='T';
                        goto L2;
                          end
        else if OPT1 = '+'v'*'then goto L4;
        else begin
               generate_code_line_STA_T; T.OPD='T';
               goto L3;
             end;
        L6: begin
              comment: terminate code generation from
                       an elementary unit CSl and
                       access the next one
                       CSlPLUSONE; l:=l+3; NZO:=
                       =VG[l]; NZB:=VG[l+1];
            if NZB ≠ 0 then begin
                             generate_code_line_
                             _OPT_T.OPD; goto L5;
                           end
            else if T.OPD = '$' then goto exit;
            else goto L6;
            end;
        exit: end code_generator;
```

Theorem: *A binary tree structure and a postfix string with an associated vector-generatrice give the codes of the same length.*

Proof: Let CS_l be given by Eq.(15). It is evident that CS_l can be represented generally as a *partial* subtree. The partial subtrees are those subtrees which have not all their nodes resolved. This uncompleteness is the consequence of the fact that the relation

$$N_l^O = N_l^B + 1 \qquad (16)$$

is not always satisfied. Bearing in mind the procedure for the binary tree derivation, the graph representation

of CS_l must form a partial subtree in the binary tree structure. Thus, the code generated from CS_l is of the same length as that obtained from the corresponding partial subtree belonging to the binary tree.

If the following relation for CS_l

$$N_l^O \leq N_l^B \qquad (17)$$

is valid, then, according to the given algorithm, CS_{l-1} will be taken, causing an instruction line

STA T

to be generated. When we generate the code from the binary tree structure, we examine each node to verify whether it contains a commutative operator. If it is so, the left or right subtree of the node can be chosen. If not, then only the right subtree must be chosen. Upon the generation of the code from the chosen subtree, the alternative subtree is taken with an instruction line

STA T

being generated. We can conclude that the choice of the left or right subtree for commutative operators does not influence the length of code obtained for that node. Thus, the fact that the given algorithm chooses always CS_{l-1}, i.e. the nearest (smallest number of nodes to be passed) right partial subtree if Eq. (17) is satisfied, will not influence the number of STA and LDA instructions to be generated. Now, we can state the following:

- *number of instruction lines generated for each CS_l is equal to that obtained from the corresponding partial subtree, and*
- *the number of STA and LDA instruction lines generated due to transition from CS_l to CS_{l-1}, CS_{l-1} to CS_{l-2}, ... etc., or due to transition from one to another corresponding partial subtree, ... etc., are equal as long as Eq. (17) is satisfied.*

To close the proof, we shall prove that for

$$N_l^O > N_l^B \qquad (18)$$

we obtained the same length of code as with the binary tree moving to the right in VG space until $N^O_{lc} = 1 \wedge \wedge N^O_{lc+1} = 0 \wedge N^O_{lc+2} = 0 \wedge \ldots N^O_{lc+i} = 0 \wedge N^B_{lc+i} \geq 1$. It is clear that when a CS_l satisfying Eq. (18) is encountered, one or several partial subtrees satisfying Eq. (17) are already handled. Moving to the right in VG space will not cause any paired instructions STA and LDA to be generated because the first operand (as a result of same previous operation) sill always reside in the accumulator. CS_l which satisfies Eq.(18) corresponds to a left subtree in the binary tree structure. Bearing in mind the fact that the code generation from a partial subtree satisfying Eq.(17) has reduced it to a leef, the code generation for a node having as its left side a subtree, and as its right side a leef, will not cause any paired instructions STA and LDA to be generated. Thus, the code generated from a CS_l satisfying Eq.(18) will be of the same length as the code generated from the corresponding partial left subtree and the leef. This closes the proof.

In fact, the code generator based on the vector generatrice simulates the code generation from the corresponding binary tree with only one exception that at each node always the right subtree is firstly handled independently from the commutativity of operator. This could have for consequence only the different generated codes.

EXPERIMENTAL RESULTS

An experiment has been performed over a set of postfix strings with associated vector-generatrices to examine the characteristics of the algorithm and compare them to those of the binary tree. The results are presented in the following table:

D_L	D_T^S	D_T^C	D_T	D_M^S	D_M^C	D_I^S	D_I^C	D_I
0	0.03	0.21	0.13	-0.12	0.12	≈ 0	≈ 0.06	0.024

D_L, D_T^S, D_T^C and D_T are defined analogously to Eqs.(1), (2), (3), and (4):

$$D_L = \frac{L_{bt} - L_{vg}}{L_{bt}}, \quad D_T^S = \frac{T_{bt}^S - T_{vg}^S}{T_{bt}^S}, \quad D_T^C = \frac{T_{bt}^C - T_{vg}^C}{T_{bt}^C}$$

$$D_T = \frac{(T_{bt}^S + T_{bt}^C) - (T_{vg}^S + T_{vg}^C)}{T_{bt}^S + T_{bt}^C}$$

D_M^S, D_M^C, D_M, D_I^S, D_I^C and D_I are defined as follows:

$$D_M^S = \frac{M_{bt}^S - M_{vg}^S}{M_{bt}^S}, \quad D_M^C = \frac{M_{bt}^C - M_{vg}^C}{M_{bt}^C},$$

$$D_M = \frac{(M_{bt}^S + M_{bt}^C) - (M_{vg}^S + M_{vg}^C)}{M_{bt}^S + M_{bt}^C}$$

$$D_I^S = \frac{I_{bt}^S - I_{vg}^S}{I_{bt}^S}, \quad D_I^C = \frac{I_{bt}^C - I_{vg}^C}{I_{bt}^C},$$

$$D_I = \frac{(I_{bt}^S + I_{bt}^C) - (I_{vg}^S + I_{vg}^C)}{I_{bt}^S + I_{bt}^C}$$

where:

- M_{bt}^S - memory space necessary for syntax analysis and transformation to binary tree,
- M_{vg}^S - memory space necessary for syntax analysis and transformation to postfix string with associated vector-generatrice,
- M_{bt}^C - memory space necessary for code generation from binary tree,
- M_{vg}^C - memory space necessary for code generation from vector-generatrice,

I^s_{bt} - number of instruction lines of syntax analyser (binary tree),

I^s_{vg} - number of instruction lines of syntax analyser (vector-generatrice),

I^c_{bt} - number of instruction lines of code generator (binary tree),

I^c_{vg} - number of instruction lines of code generator (vector-generatrice).

The experimental results obtained have shown a significant time decrease for the code generation (21 %) by applying the vector-generatrice. The overall improvement in time domain is decreased to 13 % when the syntax analysis is taken into account. Other overall improvements in performances expressed through parameters D_M and D_I are practically negligible. The code obtained from a vector-generatrice has used one up to two temporaries more than the code obtained from a binary tree even for very long expressions (in the case when the left subtrees are chosen for commutative operators).

The programming of the right-to-left code generator has been performed in such a way that after its execution, VG space is cleared (with exception of VG(K + 2). Thus, the action routine INIT1 can be ommitted from the given translation grammar, Eq.(5) reduced only to K = 1, and VG cleared only once.

CONCLUSION

The right-to-left generator has demonstrated better characteristics than that obtained from a binary tree. The length of code generated from a vector-generatrice is equal to that obtained from a binary tree. The compilation time for arithmetic expressions (syntax analysis + code generation) is appreciably shorter. This advantage is more significant for the stand-alone code generator. The total memory space (syntax analyser + code generator) was approximately the same. The memory space required by the syntax analyser based on the vector-generatrice was greater but it has been balanced by the decrease of the memory requirement of the corresponding code generator. With respect to programming, the code generator based on the vector-generatrice was programmed with smaller number of instruction lines but not significantly.

It would be of interest to extend the investigations in this domain to the application of the vector-generatrice in all operator precedence grammars and code optimization.

REFERENCES

1. Sheridan, P.B.: The arithmetic translator-compiler of the IBM FORTRAN automatic coding system, CACM, February 1959.
2. Hawkins, E.N., Huxtable, D.R.: A multi-pass translation scheme for ALGOL 60, Annual Review in Automatic Programming, 3, 1963.
3. Kanner, H., Kosinski, P., Robinson, C.L.: The structure of yet another ALGOL compiler, CACM, July 1965.
4. Randell, B., Russel, L.J.: ALGOL 60 implementation, Academic Press, 1964.
5. Graham, R.M.: Bounded context translation, SJCC, 17, 1964.
6. Anderson, J.P.: A note on some compling algorithms, CACM, March 1964.
7. Nakata, I.: On compiling algorithms for arithmetic expressions, CACM, August 1967.
8. Redziejowski, R.R.: On arithmetic expression and trees, CACM, February 1969.
9. Gries, D.: Complier construction for digital computers, John Wiley & Sons, New York, 1971.
10. Tanenbaum, A.S.: Implications of structured programming for machine architecture, CACM, March 1978.

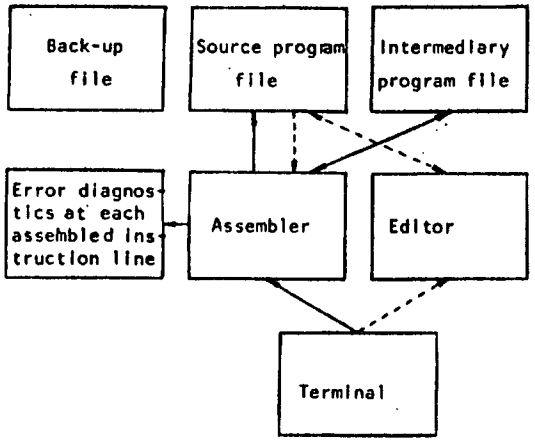


Fig. 1.

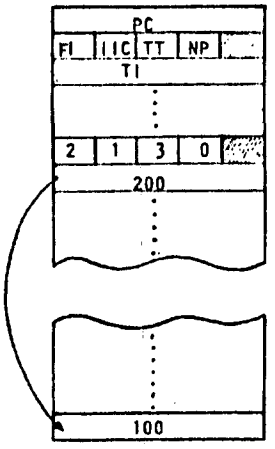


Fig. 2.

Syntax state	IIC											
	1	2	3	4	5	6	7	8	9	10	11	12
1	d_{11}	d_{12}	d_{13}	d_{14}	d_{15}							d_{112}
2	d_{21}	d_{22}	d_{23}	d_{24}	d_{25}							d_{212}
...					⋮							
i	d_{i1}	d_{i2}	d_{i3}	d_{i4}	d_{i5}							d_{i12}
...					⋮							
n	d_{n1}	d_{n2}	d_{n3}	d_{n4}	d_{n5}							d_{n12}

Fig. 3.

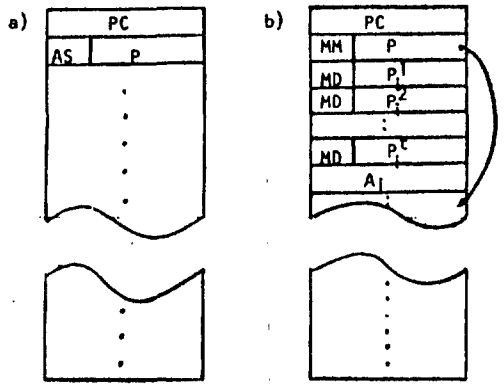


Fig. 4.

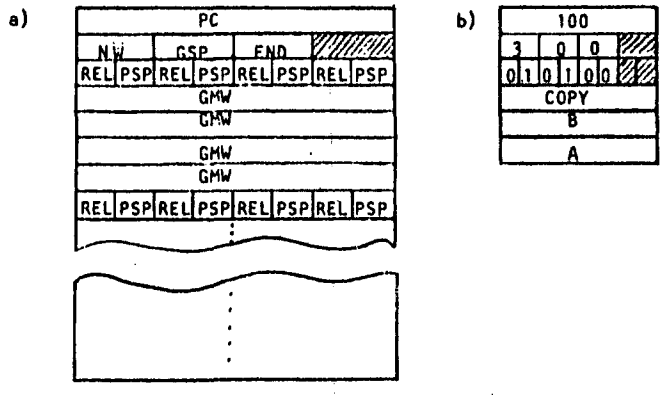


Fig. 5.

Dodatak k članku D.M.Velašević-a: AN ASSEMBLER CONSTRUCTION FOR THE INTERACTIVE PROGRAM DEVELOPMENT -Informatica broj 4/1979, ujedno se izvinjavamo čitaocima i autoru zbog ovog propusta i molimo za razumevanje.

MULTIPROGRAMIRANJE IN MULTIPROCESIRANJE NA VELIKIH RAČUNALNIŠKIH SISTEMIH BURROUGHS

KONRAD STEBLOVNIK

TGO GORENJE, VELENJE

UDK: 681.3:519.682.6

Članek na kratko opisuje izvedbo posla in osnovno podatkovno zgradbo na velikem Burroughsovem računalniku B6700. Opisani so tudi multiprogramski pripomočki, ki jih lahko uporabljamo v okviru sistemske programske opreme na tem računalniku.

MULTIPROGRAMMING AND MULTIPROCESSING ON LARGE SCALE BURROUGHS'S COMPUTERS. The paper describes the concept of the job and basic data structure for Burroughs's large scale computer B6700. Multiprogramming facilities, wich can be used in the environment of sistem software of this computer, are described also.

UVOD

V letu 1961 je združenje Burroughs pričelo proizvajati vrsto obsežnih računalniških sistemov, katerih organizacija se je radikalno razlikovala od konvencionalnih von Neumanovih strojev. Razvoj se je pričel s sistemom imenovanim B5000. Njegova modifikacija je bil B5500. V letu 1969 je ta sistem doživel veliko novosti pod imenom B6500. Še kasneje sta ta dva sistema preimenovana v B6700. Današnja izvedba velikih Burroughsovih računalniških sistemov je B6800. Našteta generacija računalnikov Burroughs se bistveno razlikuje od ostalih tako po organizaciji kot programski strojni opremi. Isti sistemi so bili med prvimi, ki so nudili učinkovito multiprogramiranje in multiprocesiranje - cilj današnjih velikih računalniških sistemov in tudi večine manjših. Načrtovalci teh sistemov so uporabili za algoritme operativnega sistema blokovno zgradbo. Te algoritme so zapisali v razširjenem programskem jeziku Algol 60, ki lahko opisuje takšno blokovno zgradbo. Čeprav ni namen tega zapisa opis strojne arhitekture sistema B6700, naj le omenimo najzanimivejšo posebnost - vhodno izhodni podsistem ter njegova povezava, po kateri se pretakajo informacije (V/I crossbar matrika) med statičnimi moduli (pomnilnik) in aktivnimi enotami (procesorji).

Članek opisuje in navaja multiprogramske pripomočke, ki jih lahko uporabljamo v okviru sistemske programske opreme na računalniku B6700, kakor tudi na vseh ostalih računalnikih Burroughs, ki spadajo k tej družini. Razširjeni programski jezik Algol 60, [1] ga navaja tudi pod imenom Burroughsov Algol, vsebuje naslednje multiprogramske pripomočke: dogodki (event), procesi (process, task), pridevki procesov (task attribute), programske prekinitve (software interrupt) in kritičen del (procure .... liberate). Z uporabo teh elementov multiprogramiranja (multitasking) lahko sočasno izvajamo na enem ali več dejanskih procesorjih, ki so vgrajeni v aparaturno opremo, več sočasnih procesov (taskov). Sistem, ki ga bomo opisali, ima zelo svojsko izvedbo posla (job) in pripadajočo podatkovno zgradbo. Na kratko bomo prikazali poleg omenjenih multiprogramskih pripomočkov, tudi posel tega računalnika, ki se lahko izvaja na enem ali več procesorjih, ter pripadajočo podatkovno zgradbo.

1. POSEL V RAČUNALNIKU B6700

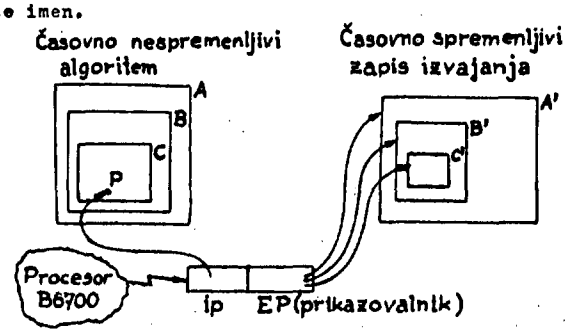
Posel tega računalnika sestavljata časovno nespremenljivi algoritem in časovno spremenljiva podatkovna zgradba, ki je zapis izvajanja tega algoritma. Algoritem sestavlja skupek nespremnljivih zakodiranih segmentov, ki so neposredno naslovljivi.

Zapis izvajanja je mnogonamenska podatkovna zgradba, ki v kateremkoli trenutku podaja:

a) stanje izvajanja posla vključno z vrednostmi za vse spremenljivke;
b) naslovljivo okolje, ki ga stvarni procesor lahko dosega, ko izvaja posel. Možno je doseganje več okolij;
c) nadzor nad preteklim sodelovanjem med bloki, postopki in procesi (taska).

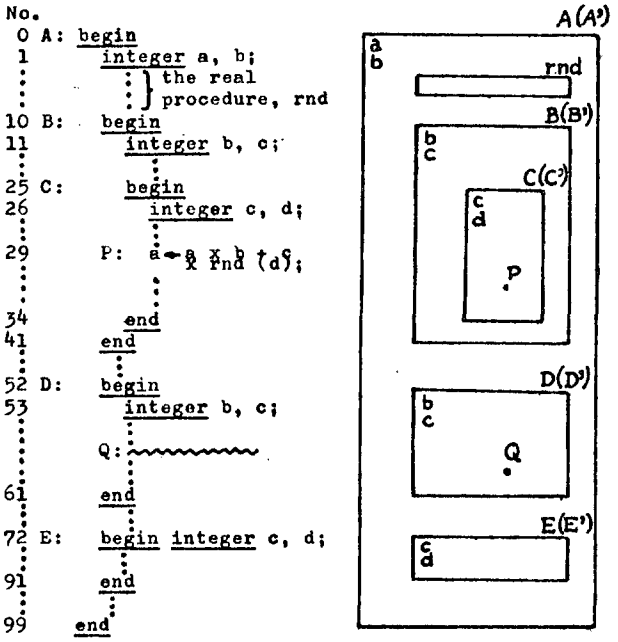
Dejanski procesor izvaja posel s pomočjo ukaznega kazalca (ip) in kazalca na okolje (ep), ki definirata stanje izvajanja.

Slika 1 prikazuje posel v trenutku, ko procesor izvaja ukaz P na katerega kaže ukazni kazalec. Kazalec na okolje EP (prikazovalnik) deluje kot vektor kazalcev, ki kaže na področja zapisa C', B' in A'. A', B', C' so podatkovna območja, ki se dodeljujejo blokom A, B, C ob vsakokratnem pozivu (izvajanju) bloka. Takšna zgradba odgovarja visokim programskim jezikom (ALGOL 60, PL/1) z blokovno zgradbo. Dostopno okolje za procesor deluje kot unija dostopnih področij C', B' in A'. Gnezdenje teh področij odgovarja gnezdenju programskih blokov A, B in C, kateri vsak definira območje dosega programske liste imen.



Slika 1. Skica posla v računalniku B6700.

Lep primer programa, napisanega v programskem jeziku Algol in odgovarjajoče skice zgradbe, je na sliki 2. Takšna skica zgradbe odgovarja tako algoritmu, kot zapisu izvajanja. Posamezna področja E', D', C', B' in A' vsebujejo celice z ustrezno listo imen (npr. področje C vsebuje listo c, d). Te celice so dostopne po imenu.



Algolski tekst Struktura vgnezdenja

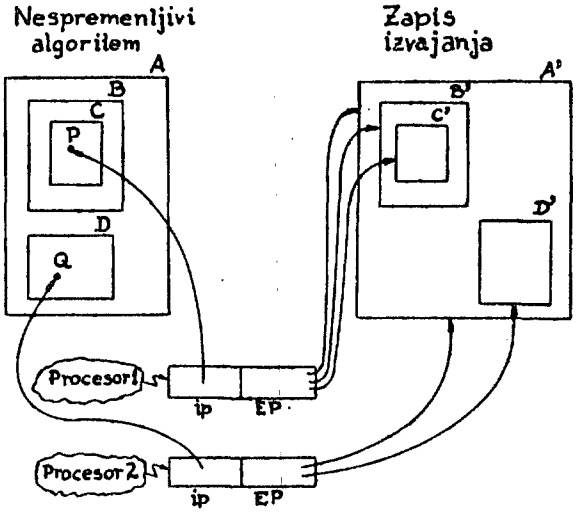
Slika 2. Program zapisan v programskem jeziku Algol in odgovarjajoča skica zgradbe.

Za primer, kako deluje takšna zgradba, si oglejmo trenutek, ko se izvaja ukaz P in zahteva dostop do celice z imenom a. Izvede se pregledovanje zapisnih področij C', B' in A' in sicer v smeri od C' proti A' (to pregledovanje seveda ne vključuje področij D' in E'). Takoj, ko se najde celica z imenom a, se pregledovanje zaključi. Podobno pregledovanje se izvede za celico b v področju B's tem, da so v trenutku izvajanja ukaza P celici b v področju A' in področju D' "nevidni" za procesor. To pregledovanje se izvršuje prek statičnih povezav med posameznimi področji.

V primeru, da postopke (procedure) izvajamo kot procese, ne pa kot podprograme, doseže program t.i. drugi nivo aktivnosti (site of activity).

V računalnik B6700 je lahko vgrajenih tudi dva ali več procesorjev. Slika 3 prikazuje koncept izvajanja procesa; prikazan je shematično. Slika kaže, kako si dva procesorja razpodelita isto kodo in isti zapis izvajanja (pravzaprav del zapisa). Ker je področje A skupno okolju obeh procesorjev, je potrebno izvesti nekatere konstrukte za dosego vzajemne izključenosti (mutual exclusion). Program, ki ga lahko izvajata dva ali več procesorjev, se imenuje algoritem z večkratno aktivnostjo (multiple activity algorithm).

Programu, ki ga lahko izvaja več procesorjev, ni potrebno za učinkovito izvajanje dodeliti več procesorjev. Enojen procesor je lahko razdeljen na več nivojev aktivnosti. V tem primeru lahko en procesor upoštevamo kot virtualen ali psevdoprocesor in je potrebna za izvajanje posla le učinkovita politika razvrščanja (scheduling). V B6700 lahko virtualni procesor zamenjamo z dejanskim procesorjem (in seveda obratno) z enostavnim ukazom.



Slika 3. Skica posla v računalniku B6700, ki ga izvajata dva procesorja.

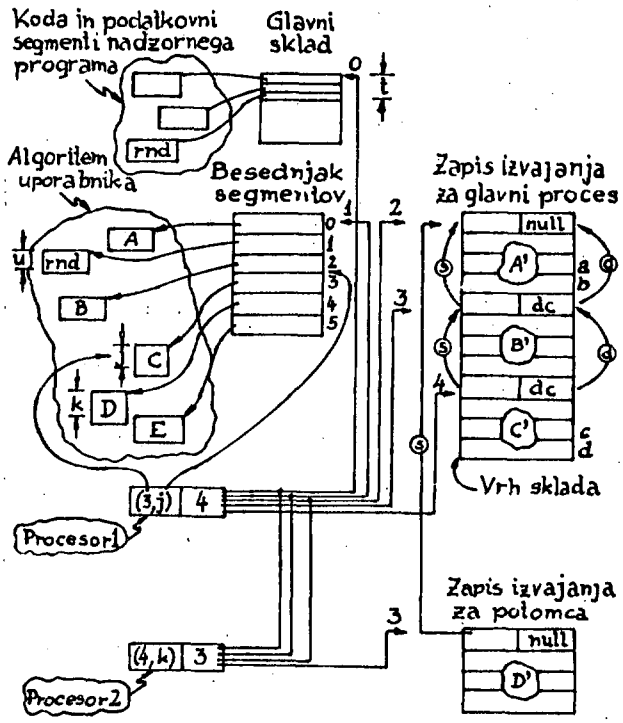
Skica na sliki 3 prikazuje posel v trenutku, ko se izvajata dva procesa na B6700. Glavni proces izvaja ukaz P z dostopnim okoljem C', B' in A', medtem, ko njegov potomec, ki je bil klican v neki prejšnji točki izvajanja glavnega procesa, izvaja ukaz Q z dostopnim okoljem D' in A'.

2. OSNOVNA PODATKOVNA ZGRADBA ZA ALGORITEM B6700

V tem poglavju bomo na kratko opisali podatkovno zgradbo za B6700, ki temelji na principu sklada. Nekatere osnovne pojme si bomo razjasnili ob sl. 4, ki prikazuje algoritem in odgovarjajoči zapis izvajanja iz slike 2. Sestoji se iz dveh skladov, besednjaka segmentov in glavnega sklada. Prikazovalna vektorja za oba procesorja kažeta na naslovno okolje (tudi skladovna zgradba), katerih vrh sta zapisa za C' in D' na nivojih 4 oz. 3.

Koda za B6700 algoritem je razdeljena na bloke in koda za vsak blok je shranjena kot fizično ločen segment. Vsak vstop v besednjak segmentov se izvede s pomočjo segmentnega kazalca. V fizično naslovljivem pomnilniku morajo biti prisotni samo segmenti, ki so dejansko potrebni aktivnemu procesorju. Prisotnost segmenta definira bit prisotnosti. Besednjaku segmentov pripada še odgovarjajoči osnovni naslov. Ukazni kazalec je dvojen: prva komponenta (v tem primeru 3) je odmik znotraj besednjaka segmentov in druga komponenta (j) je odmik znotraj segmenta. Ukazni kazalec ima lahko tudi tri komponente, kar bomo spoznali pozneje. Osnovni naslov besednjaka segmentov se določi prek kazalca, ki je del prikazovalne povezave. Kadarkoli izvajanje programa zahteva nov blok, se dodeli segmentnemu skladu nov aktivirani zapis, ki se doda vrhu sklada. Na predhodni zapis se naveže s pomočjo povezovalne besede na dva načina: statično in dinamično. Statična povezava definira gnezdenje okolja, dinamična pa je v tem trenutku navezovanja kar enaka statični. Dinamične povezave zagotavljajo procesorju vse potrebne informacije za pravilno dodeljevanje in opuščanje aktiviranih zapisov.

Področje z imenom glavni sklad opisuje delovanje sistemskega nadzornega programa in med drugim vsebuje zapis za vse kode nadzornih segmentov in sistemske tabele. Segmenti tega sklada so tudi dostopni prek prikazovalnega kazalca na nivoju 0.



Slika 4. Podatkovna zgradba za računalnik B6700.

Procesor potrebuje seveda še nekaj dodatnega začasnega pomnilnika za začasno pomnenje vmesnih rezultatov npr. pri razvijanju ukazov. V ta namen uporabi poseben operativni sklad za vsak virtualni procesor.

Program na sliki 2 vsebuje tudi postopek z imenom "rnd". Ko procesor prične izvajati ta postopek, se prek dinamične povezave naveže na zapis izvajanja nov aktivirani zapis, ki odgovarja postopku "rnd". Ta zapis vsebuje poleg imen začasnih spremenljivk še povratno informacijo. Ukazni kazalec se v tem primeru sestoji iz treh komponent (1, 1, u). Prvi element vodi do tabele, ki vsebuje opis segmenta ukazov (1 pomeni besednjak segmentov); naslednja dva elementa pa pomenita odmik znotraj besednjaka segmentov (1) in znotraj samega segmenta (u). Aktivirani zapis za "rnd" je dinamično povezan na blok C in statično povezan na blok A, ker je postopek "rnd" definiran v bloku A.

Predpostavimo, da je sedaj "rnd" ali pa katerikoli drugi del programa definiran kot sistemska operacija; opis segmenta, ki vsebuje njegovo kodo, pa je del glavnega dela sklada. V zapisu izvajanja se v trenutku vstopa v ta segment tvori odgovarjajoči aktivirani zapis, ki se dinamično naveže na blok C in je statično povezan z glavnim skladom. Ukazni kazalec se sestoji tudi iz treh komponent (0,t,u):
0 - pomeni glavni sklad, t odmik znotraj sklada in u odmik znotraj segmenta.

V primeru klica procedure (rnd) prvi element v ukaznem kazalcu pove, ali je ta procedura definirana na nivoju našega algoritma ali na nivoju sistemskega nadzornega programa.

B6700 vključuje tudi prekinitve, ki se obravnavajo kot nepričakovani podprogramski klici (postopki).

3. KOMUNIKACIJA MED SEKVENČNIMI PROCESI

Na računalniškem sistemu se izvajajo razni procesi, ki so lahko stalni ali pa začasni. Med njimi obstajajo določene zveze v času in prostoru. Glede na to jih razdelimo v tri vrste:

- neodvisne procese
- paralelne procese
- komunicirajoče ali odvisne procese.

Nas zanimajo predvsem odvisni procesi, ki lahko spreminjajo informacije, ki so jim skupne in so dalje razdeljeni v medlo odvisne procese in sodelujoče procese. Odvisni procesi si morajo deliti različne računalniške vire in jih je potrebno zaradi tega sinhronizirati ter onemogočiti sočasno uporabo teh virov (vzajemna izključenost). S tem v zvezi se pojavi problem kritičnega dela procesa. Ta del procesa je kritičen zaradi tega, ker se znotraj njega, ob uporabi določenega vira spreminja ali dosega informacija, ki je skupna več procesom. V literaturi zasledimo več načinov reševanja problemov, ki nastopijo ob takšni medprocesni komunikaciji. Najbolj znani pobudniki za reševanje teh problemov so Dijkstra, Hoare in Hansen, ki so uvedli metode strukturiranega programiranja na tem področju in s tem v zvezi naslednje pojme:

- binarni in splošni semafor
- pogojno kritičen del procesa
- monitor ali tajnik in
- hierarhija sekvenčnih procesov in monitorjev.

Omenimo naj še multiprogramski jezik Modula [2], ki ga je definiral Wirth in je namenjen za programiranje sprotnih sistemov ter krmilnih programov za vhodne/izhodne naprave.

Na te splošne ugotovitve lahko navežemo teorijo o sestavljanju družine procesov na velikih Burroughsovih sistemih. Ta teorija sloni na razširjenem Burroughsovem programskem jeziku Algol.

3. 1. Klicanje in vsklajevanje procesov

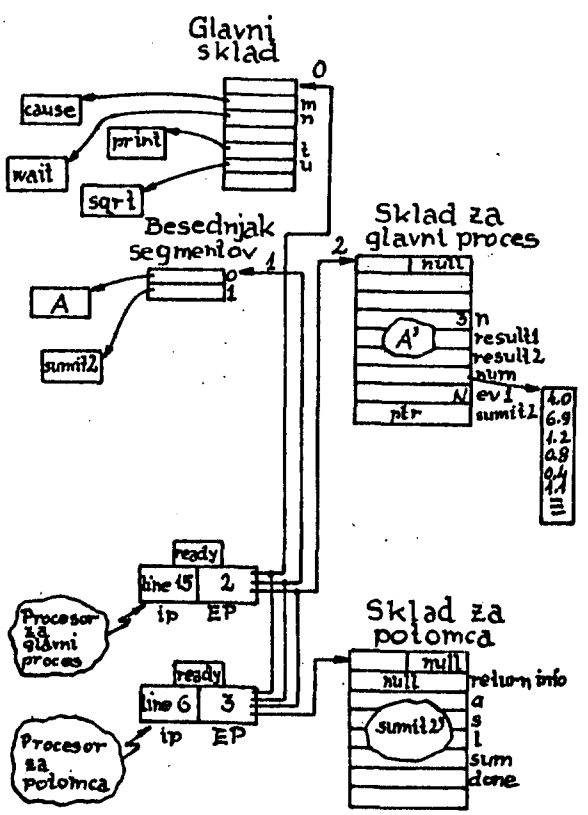
V tem poglavju bomo opisali, kako program doseže takoimenovani drugi nivo aktivnosti; to pomeni, da program v določeni točki kliče proces, ki lahko prevzame del njegovega opravila ter ga izvaja do določenega trenutka asinhrono in neodvisno, vendar sočasno. Algoritem za program, kjer glavni proces sproži svojega potomca (offspring task), je napisan na sliki 5.

```
Line
No.
 0  begin
 1    integer j, n; real result1, result2;
 2    array num 0:99 ;
 3    event ev1, null: comment deklaracija
                       dogodkovnih spremenljivk;
 4    procedure sumit2 (a, s, l, sum, done);
 5      value s, l; integer s, l; real sum;
                    array a[*]; event done;
 6      begin
 7        integer i;
 8        sum←0;
 9        for i←s step 1 until l do
          sum←sum+sqrt (a[i]);
10        cause (done); comment sistemska operacija;
11      end sumit2
12 ;
13   [input value for n  50 and {num}, for j=0step
      1 until 2x n - 1]
14   process sumit2 (num, n, 2 x n - 1, result2,
     ev1); comment glavni proces sproži svojega
           potomca;
15   sumit2 (num, 0, n - 1, result1, null);
     comment glavni proces kliče proceduro sumit2;
16   wait (ev1); comment sistemska operacija;
17   print (result1 + result2); comment sistemska
                                operacija;
18   end
```

Slika 5. Algoritem za program v Burroughsovem Algolu, kjer določeno opravilo izvajata dva istočasna procesa.

Na sliki 5 je v vrsticah 3, 5, 10, 14, in 16 večina novih sintaktičnih enot, ki omogočajo enostavno vsklajevanje med procesi. V vrstici 3 je deklarira-

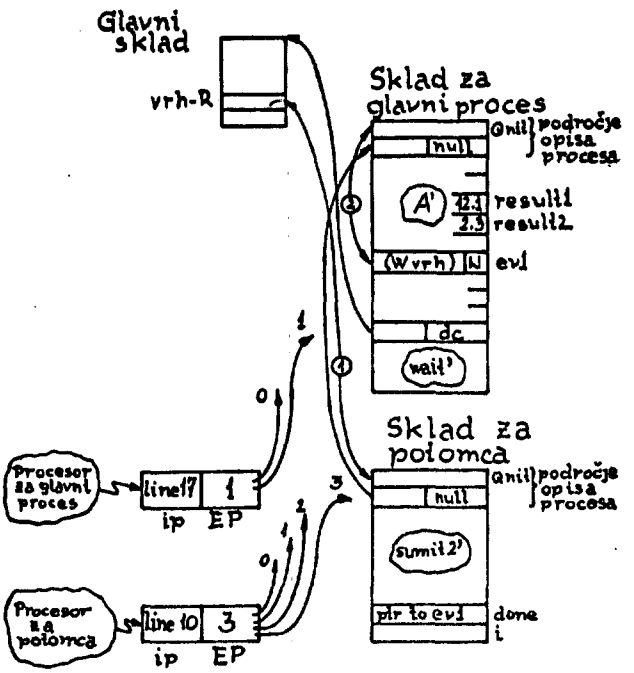
na spremenljivka "ev 1" tipa event (dogodek) kot osnova za vsklajevanje oziroma sinhroniziranje. V vrstici 5 je deklariran prilagoditveni formalni parameter z imenom "done" za postopek sumit2. Burroughsov Algol uvaja tudi ključno besedo process, da razlikuje klic procesa (vrstica 14) od običajnega klica postopka (vrstica 15). Klic procesa posreduje postopku sumit2 dejanski parameter, da lahko potomec preko njega signalizira glavnemu procesu, da je zaključil svoje opravilo. To opravi prek sistemske operacije cause (vrstica 10). Potem , ko je glavni program sprožil proces v vrstici 14, kot svojega potomca, izvede običajni klic postopka sumit2, ki posreduje referenco null kot sintaktičnega statista za formalni parameter done. Po izvedbi postopka pokliče sistemsko operacijo wait, ki se izvaja toliko časa, dokler dejanski parameter ev1 ne doseže vrednosti, ki se lahko interpretira kot: "dogodek se je zgodil" (N-not happend, H-happend). Po zaključku sistemske operacije wait algoritem pokliče operacijo print za izpis vsote dveh vrednosti result1 in result2. Slika 6 prikazuje podatkovno zgradbo za algoritem iz slike 5 v trenutku, ko glavni program izvaja klic postopka sumit2 v vrstici 15, njegov potomec pa vrstico 6.



Slika 6. Podatkovna zgradba za algoritem iz slike 5

Spremenljivke tipa event so strukturirane in eno polje v tej zgradbi je binarno stikalo, ki se imenuje dogodkovni bit. Začetna vrednost tega bita je: "dogodek se ni zgodil" (N) in operacija cause ga postavi na vrednost: "dogodek se je zgodil" (H).

Slika 7 prikazuje dejansko vključitev naših procesov v sistem B6700 in v njegov nadzorni program.



Slika 7. Dejanska vključitev algoritma iz slike 5 v sistem računalnika B6700.

Sklad za vsak nov proces se prične s področjem opisa procesa (task description area), ki ima stalno širino. Temu sledi prvi zapis aktiviranja. Kombinacija aparaturne opreme in sistemskih programov zagotovi, da je to področje dostopno samo nadzornemu programu. Eden od ključnih delov informacije tega posebnega področja je nit (thread - Q), s pomočjo katere se proces lahko naveže na glavni seznam (list head), ki definira stanje razvrstitve (queue state) procesa. Če je proces pripravljen (ready), je nit Q navezana (črta označeno z 1 na sliki 7) na vrh (head - R) v glavnem skladu. Z uporabo takšne povezave, lahko sistemski nadzorni program izbira procese, ki se naj izvajajo.

Povratni naslov za vsak proces je na sliki 6 označen z null. To bi pomenilo, da preide vsak proces ob svojem zaključku na izvajanje stavka null. Dejansko je na sistemu B6700 podana namesto vrednosti null vstopna točka v sistemsko nadzorno proceduro.

Zapis aktiviranja za sistemsko proceduro, ki zaključi proces, se doda glavnemu skladu.

3. 2. Pridevki procesov

Da dosežemo večji nadzor in/ ali možnost sodelovanja znotraj družine procesov, so vsakemu procesu dodeljene strukturirane spremenljivke, katere elementi definirajo različne lastnosti procesov.

V času sprožitve procesa mora biti deklarirana spremenljivka tipa task. Zaradi tega je algoritem na sliki 5 sintaktično nepravilno zapisan. Pravilen zapis vrstice 14 tega algoritma bi bil v Burroughsovem Algolu tale:

process sumit 2 (num, n, 2 x n-1, result2, ev1) [charlie];

če je Charlie ime tega procesa.

Predhodno pa je potrebno deklarirati še spremenljivko Charlie tipa task:

task charlie

Spremenljivka tipa task ima stalno zgradbo in je dodeljena v informacijsko področje procesa kot del

sklada procesa v času, ko se proces kliče.

V času, ko je proces klican, dobijo njegovi pridevki neko začetno vrednost. Ta začetna vrednost lahko kasneje v času, ko je proces aktiven, ostane nespremenjena, lahko jo spremeni proces sam ali pa določeni drugi procesi iz družine, navadno procesi, ki ga kličejo.

Ko je sprožen proces, se lahko nanj predhodno navežejo različni pridevki. V primeru procesa Charlie, bi lahko zapisali takšne pridevke na tale način:

12.1 Charlie. priority ← 5;
12.2 Charlie. maxproctime ← 20; comment in seconds;
12.3 Charlie. stacksize ← 2^8;

Skupek pridevkov procesa, ki so zapisani v vrsticah 12.1, 12.2 in 12.3, razpozna sistem, začetne vrednosti pa postavi proces, ki ga je sprožil. Te vrednosti upošteva sistemski algoritem za razvrščevanje ali drugi moduli za upravljanje z viri sistema.

Posebne medsebojne odnose med različnimi procesi dosežemo z nekaterimi ključnimi pridevki procesov. Opisali bomo štiri različne pridevke tega tipa: status, exeptiontask, exeptionevent in partner.

Pridevek status opisuje trenutno stanje izvedbe določenega procesa. Proces je lahko razvrščen, aktiven, začasno ustavljen, zaključen. Ta pridevek je dostopen kateremukoli procesu v družini, za katerega je ime procesa (Charlie) vidno, to je procesu, ki ga je sprožil ali kateremukoli nasledniku, za katerega je Charlie globalen. Takšni procesi lahko prek spremenljivke status vsilijo določeno stanje procesu Charlie: lahko ga začasno ustavijo, zaključijo itd. Npr.

myself. status ← suspended

Proces, ki je sprožil proces (npr. Charlie ali katerikoli drug proces, ki lahko "vidi" spremenljivke procesa Charlie), lahko deluje kot Charliejev nadzornik ali "starejši brat", ker takšen proces (razen Charlieja samega) lahko spreminja ali čita njegove pridevke. Če je npr. proces Pete sprožil Charlieja, ga lahko aktivira, začasno ustavi ali pa zaključi in tako lahko nadzoruje njegovo stanje prek pridevka Charlie. status. S pomočjo pridevka exceptiontask pa lahko kasneje prenesemo funkcijo nadzora nad procesom Charlie, ki ga je od začetka imel Pete in sam Charlie, na drug proces z naslednjim stavkom:

myself. exceptiontask ← brothertom;

Na ta način se lahko dva procesa povežeta v zaključeno verigo.

S pomočjo pridevka partner lahko omogočimo, da dva procesa delujeta sinhrono kot sosledni operaciji. Procesa A in B lahko delujeta kot sosledni operaciji potem, ko priredimo: A. partner = B in B.partner = A. Če sedaj proces A izvede stavek

continue;

se proces A ustavi in proces B se prične izvajati tam, kjer se je ustavil ob nastopu continue stavka. Trije ali več procesov lahko delujejo kot sosledne operacije tako, da npr. tvorijo obroč:
A.partner ← B, B.partner ← C; C.partner ← A.

Kadar se zahteva kakršnakoli operacija nad temi pridevki, sistemski razvrščevalnik odloči, kdaj se bo to izvršilo. Kasneje lahko samo procesi, ki so obveščeni o tej operaciji, izvršijo nasprotno akcijo.

3. 3. Primer

V prejšnjem poglavju smo podali semantiko spremenljivk procesov in pridevkov procesov in sedaj z njihovo pomočjo rešimo naslednji problem.

Predpostavimo, da imamo tri identične krmilnike (controller) pretoka, ki lahko sodelujejo med sabo in vplivajo drug na drugega. Vsak krmilnik lahko nadzoruje pretok, ki si ga zamislimo kot pretok olja, vode ali zrnatega materiala ali pa kot pretok števil. Vsak krmilnik dovoljuje nadzorovani pretok v zbiralnik ter pozna nakopičeno vrednost v zbiralniku, ker lahko stalno meri vrednost pretoka. Naloga vsakega krmilnika je, da naj v sodelovanju z ostalimi dovoljuje približno enako velikost pretoka, kot se trenutno prevaja preko ostalih. Za zapis računalniškega programa, ki ustreza takšni situaciji, predpostavimo da se pretoka zaporedje celih spremenljivk, ki se iz vhodne datoteke čitajo kot konstantne vrednosti. Za poenostavitev predpostavimo, da krmilni program A meri pretok enostavno tako, da generira vsoto celih vhodnih spremenljivk. To vsoto imenujemo SA. Krmilni program A pozna nakopičeno vrednost v ostalih dveh zbiralnikih, ker ima dostop do generiranih vsot SB in SC ostalih dveh krmilnikov.

Prav tako lahko A periodično nadzoruje pogoj:

SA > SB and SA > SB = true

in če je pogoj izpolnjen, lahko prekine pretok v svoj zbiralnik. Predpostavimo, da lahko krmilni program A učinkovito sporoči najmanj enemu od ostalih dveh to, da je začasno ustavljen. Če pa pogoj ni izpolnjen, pa A nadaljuje s pretokom v svoj zbiralnik. Ko krmilni program A ugotovi, da je pogoj SA > INMAX izpolnjen, lahko zaključi z izvajanjem in preneha obstajati. Ta opis velja seveda za vsak krmilnik A, B in C.

Možnih je več rešitev. Vsakemu krmilniku priredimo lasten proces in ti procesi sodelujejo med sabo. Lahko bi jih tudi izvedli kot sosledne operacije. Na sliki 8 je algoritem, ki uporablja procesna pridevka status in exeptiontask. Ko dani krmilnik ugotovi, da prehiteva ostala dva krmilnika, ju obvesti o tem vedno po round robin algoritmu: A ← B, B ← C in C ← A.

Proces, ki je začasno ustavil samega sebe (vrstica 12) lahko ponovno sproži samo drug proces, ki je njegov izjemni proces (exeptiontask), to je na sliki 8 v vrstici 11.

Opisane spremenljivke procesov in njihovi pridevki za nadzor in komuniciranje med procesi niso vse. Literatura [1] navaja, da obstaja še več drugih pridevkov procesov npr. pridevek locked tipa Boolean ali pridevek taskvalue tipa real. Ta kratek pregled nas lahko prepriča, da ima programer močno orodje za tvorbo kompleksnih podsistemov v obliki družine procesov.

Sintaktične konstrukte, ki smo jih spoznali v tem poglavju, lahko na kratko primerjamo z rešitvami, ki so jih navedli drugi avtorji na področju multiprogramiranja. Pripomoček imenovan proces, je podan tudi v programskem jeziku Modula [2] in je podoben proceduram, s tem da se izvaja sočasno s programom (oziroma procesom), ki ga je klical. Procesi so v Moduli lahko inicializirani samo v glavnem procesu in ne morejo biti vgnezdeni.

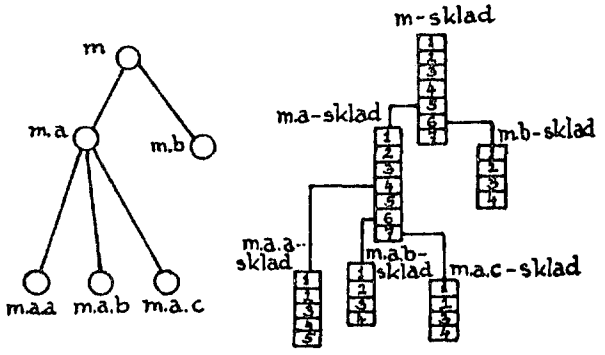
Sinhronizacijo med procesi lahko v našem primeru dosežemo s spremenljivkami tipa event kot argumente operacij "wait" in "cause" (čakaj na dogodek, povzroči dogodek). V programskem jeziku Modula lahko primerjamo s tema dvema operacijama operaciji "wait" in "send" (čakaj na signal, pošlji signal), medtem ko bi dogodku odgovarjala spremenljivka tipa signal. Signalu pa pri Hoarju [4] odgovarja pojem pogoja, pri Hansenu pa pojem vrste.

Pri Moduli je vpliv na potek procesov dosežen z uporabo signalov in skupnih oziroma deljenih spremenljivk. Na B6700 lahko v Burroughsovem Algolu dosežemo takšen vpliv z dogodkovnimi spremenljivkami in pridevki procesov. Kritični deli procesov, ki delujejo nad skupnimi spremenljivkami, so v Moduli de-

```
Line
No.
 1  begin
 2    integer SA,SB,SC, INMAX; task A, B, C;
                               event doneABC;
 3    procedure controller1 (s1, s2, s3, n,done);
 4      value n; integer s1, s2, s3, n; event done;
 5      begin
 6        integer VAL; label L;
 7        L: If s1 > s2 and s1 > s3
 8           then begin
 9             [print values of s1, s2, and s3 in
                appropriate columns
10               of a table, based on the value
                 of n];
11             (myself.exeptiontask).
               status ← "wakeup";
12             myself.status ← "suspended";
13             go to L; end
14           else begin
15             [input a value of VAL from input
                file n];
16             s1 ← s2 + VAL;
17             if s1<INMAX then go to L
               else cause(done);
18               end
19     end controller1
20   ;
21   A.exeptiontask ← B;
22   B.exeptiontask ← C;
23   C.exeptiontask ← A;
24   [input a value for INMAX];
25   SA ← SB ← SC ← 0;
26   process controller1(SA,SB,SC,1,doneABC)[A];
27   process controller1(SB,SA,SC,2,doneABC)[B];
28   process controller1(SC,SA,SB,3,doneABC)[C];
29   wait(doneABC);
30 end
```

Slika 8. Program za krmiljenje pretoka števil v tri zbiralnike.

klarirani kot vmesniške procedure in so zbrani v vmesniških modulih (monitor pri Hoaru in Hansenu). V našem primeru se take operacije izvajajo pod kontrolo sistemskega nadzornega programa, ki poskrbi, da ne pride do konfliktnih situacij. S pridevki procesov lahko torej v nasprotju z Modulo kontroliramo potek procesov. Kasneje bomo spoznali še eno možnost vpliva na potek procesov, to je s programskimi prekinitvami.



Slika 9. Drevesna zgradba družine procesov.

V nasprotju z Modulo lahko v Burroughsovem Algolu tvorimo družino procesov, ki imajo drevesno zgradbo. Takšen, zelo poenostavljen primer je prikazan na sliki 9. Na tej sliki je na levi strani drevesna struktura, na desni pa poenostavljena povezava posameznih zapisov izvajanja, ki imajo skladovno zgradbo in jih sestavljajo posamezni zapisi aktiviranja, ki so statično povezani. Prav tako so statično povezani posamezni procesi in sicer tako, da je npr. proces m.a.a. statično navezan na m.a. sklad na zapise aktiviranja 4,3,2 in 1 ne pa na 5,6 in 7. Zapis aktiviranja ali blok s številko 4 v m.a. skladu se imenuje kritičen blok za m.a.a. sklad. Ko se proces m.a. konča ali se kako drugače neha izvajati, njegov naslednik ne more več eksistirati, ker nima več dostopa do okolja svojega predhodnika. Zaradi tega mora proces, ki konča, končati tudi vse svoje potomce.

3.4. Programske prekinitve

Programska prekinitev ima nekatere podobnosti z ožičeno prekinitvijo. Izvor programske prekinitve se nahaja v nekem drugem procesu iz družine procesov in proces, kateremu je prekinitev namenjena, preide, ko prejme prekinitev, na izvajanje neke v- naprej definirane procedure (prekinitvene procedure). Prejemnik prekinitve jo lahko ignorira tako, da začasno onemogoči programsko prekinitev. Če se pa proces, kateremu je namenjena prekinitev, trenutno ne izvaja, sistemski nadzorni program poskrbi, da se prekinitveni signal postavi v vrsto in se obravnava v trenutku, ko se odgovarjajoči proces prične ponovno izvajati.

Prekinitev deklariramo v B6700 Algolu z naslednjim stavkom:

interrupt (name of interrupt procedure);(statement);

Če zapišemo:

interrupt i1; begin . . . end;

pomeni, da je to prekinitvena procedura z imenom i1

Prekinitvena procedura se lahko naveže na dogodkovno spremenljivko s stavkom:

attach (name of interrupt procedure) to (event variable);

Primer:

```
1  event ev;
2  interrupt i1; begin ... end;
3  attach i1; to ev;
4  enable i1;
```

V vrstici 1 je deklarirana dogodkovna spremenljivka ev; vrstica 2 deklarira prekinitveno proceduro z imenom i1; vrstica 3 naveže prekinitveno proceduro i1 na dogodkovno spremenljivko ev; vrstica 4 pa omogoči prekinitev, ki se povzroči preko spremenljivke ev.

Prekinitev lahko onemogočimo s stavkom:

disable i1;

Na začetku se privzame, če ne vključimo stavka enable, da je prekinitev omogočena.

Dogodkovna spremenljivka se lahko onemogoči s stavkom:

detach i1;

nakar lahko navežemo na prekinitev i1 katerikoli drug dogodek.

3.5. Doseganje virov

Kot smo že na kratko omenili, si morajo procesi, ki se odvijajo na nekem računalniškem sistemu, učinkovito razpodeliti uporabo računalniških virov. V času, ko proces zahteva določen vir, se mora izvesti kritičen del programa, ki ga uvaja Dijkstra [3]. Predhodno smo dogodke opisovali kot dvostanjske s stanji: "dogodek se je zgodil" in "dogodek se ni zgodil". Na sistemu B6700 lahko dogodke interpretiramo še na naslednji način: " dogodek je na razpolago" ali "dogodek ni na razpolago".

Proces, ki izvaja stavek:

```
procure(ev);
```

je začasno ustavljen, če dogodek ev ni na razpolago, v nasprotnem pa izvede nasljednji stavek zaporedja, ki sledi. Kritičen del se v Burroughsovem Algolu zapiše na nasljednji način:

```
procure(ev);
   —
   —
liberate(ev);
```

Operaciji "procure" in "liberate" torej lahko primerjamo z operacijama "P" in "V", ki se izvajajo nad semaforjem in jih je uvedel Dijkstra.

SKLEP

Iz tega zapisa in nekaterih opisanih primerov vidimo, da so načrtovalci programske opreme Burroughsovih računalnikov že zelo zgodaj razvili učinkovite pripomočke za multiprogramiranje na računalnikih, ki so tudi multiprocesorski. Čeprav so verjetno svojo opremo razvijali ločeno od ostalih teoretikov multiprogramiranja, lahko napravimo primerjavo med posameznimi rešitvami.

LITERATURA

[1] Elliot J. Organick: Computer System Organisation, Academic Press 1973, New York and London.

[2] N. Wirth: Modula: a Language for Modular Multiprogramming, Software-Practice and Experience, vol. 7, 335 (1977).

[3] Dijkstra E. W.: Cooperating Sequential processes, in Programming languages, ed. Genuys, Academic Press 1968.

[4] C. A. R. Hoare: Monitors-an Operating System Structuring Concept, Com. of the ACM, vol. 17, no. 10 (Oct. 1974).

PODATKOVNA BAZA PROGRAMSKEGA PAKETA ZA AVTOMATSKO PROJEKTIRANJE

M. MARUŠIČ
B. VILFAN
M. TONI*

FAKULTETA ZA ELEKTROTEHNIKO, UNIVERZA EDVARDA KARDELJA V LJUBLJANI
*INŠTITUT JOŽEF STEFAN, ODSEK ZA UPORABNO MATEMATIKO

UDK: 681.3:519.685

V sestavku je opisana podatkovna baza za računalniški programski paket za avtomatsko projektiranje in avtomatsko izdelavo projektne dokumentacije. Trenutno se ta paket uporablja za projektiranje zavarovanja cestno - železniških prehodov. Zaradi popolnejšega pregleda je najprej podan kratek opis celotnega paketa, nato pa se opis enotne podatkovne baze.

DATA BASE OF A SOFTWARE PACKAGE FOR COMPUTER AIDED DESIGN

In these section a description is given of unified data base for the software package for the automation of design activities and the automatic production of design documentation. The package is currently being used by a consulting organisation in the design of railway - highway intersection safety equipment.

1. UVOD

Projektiranje je dejavnost, ki zahteva visokokvalificirane in izkušene strokovnjake ter veliko predelovanja informacij. Pri svojem delu se projektanti velikokrat srečujejo z rutinskimi dobro utečenimi deli pri katerih je njihova ustvarjalnost na precej niski ravni sledé na sposobnosti in znanje, ki ga ti ljudje imajo. Prav zato je vsak pripomoček ali postopek, ki zmanjšuje obseg takega dela, zelo zaželjen. Projektant dobi tako možnost, da večino svojih sil usmeri v kreativno delo in tako tudi dvigne svojo produktivnost.

Skupina računalniških strokovnjakov iz Fakultete za elektrotehniko Univerze Edvarda Kardelja v Ljubljani (B. Vilfan, V. Mahnič, M. Marušič) in iz Instituta 'Jožef Stefan' Odseka za uporabno matematiko (M. Toni, R. Kolar, J. Barle) je izdelala za potrebe organizacije ISKRA Avtomatika TOZD Inženiringi računalniški programski paket z namenom, da osvobodi projektanta rutinskih opravil in obenem tudi skrajša čas projektiranja. Konkretna aplikacija se nanaša na izdelavo projekta za zavarovanje cestno - železniških prehodov.

2. SPLOSNO O PAKETU

Preden smo pristopili k izdelavi programskega paketa smo se morali spoznati z vsemi deli, ki jih mora opraviti projektant pri projektiranju zavarovanja cestno - železniških prehodov.

Pod zavarovanjem cestno - železniških prehodov razumemo nabor naprav in povezav med temi napravami, ki so nameščene na železniški progi in ob njej. Njihova naloga je zagotoviti varnost prometa pri križanju železniške proge s cesto v istem nivoju. Osnovne naprave, ki spadajo v ta sistem zavarovanja, so: cestni svetlobni signali, zapornice, magnetni tirni kontakti, relejne naprave za krmiljenje signalov in zapornic ter kabli in druge pomožne naprave. Konfiguracija zavarovalnega sistema je odvisna od karakteristik proge in prometa na njej ter od ceste in cestnega prometa. Naloga projektanta je analizirati vse značilnosti, ki vplivajo na konfiguracijo zavarovalnega sistema. Na podlagi te analize projektant izbere ustrezne naprave, naprave geografsko locira in predvidi njihive povezave. Končni projektantov izdelek je 'projekt', ki vsebuje:

- besedni opis celotnega zavarovalnega sistema in njegovih elementarnih delov
- različni načrti
- specifikacija vseh naprav in del, ki so potrebna za realizacijo zavarovanja
- podrobna navodila in predpisi za postavljanje in povezavo naprav.

Kaj se spremeni v delu projektanta z uporabo našega paketa?
Programi izpišejo besedilo, ki je sprejemljivo za končnega naročnika projekta, izdelajo načrte ustrezne kvalitete, izdelajo tabele s specifikacijo opreme in del, tabele kabelskih povezav ter še tabele za vrsto drugih podatkov. Projektant sedaj nadzoruje računalniške rezultate in v primerih, ko z rezultati ni zadovoljen ali pa je prehod nestandarden, lahko sam posega v vmesne faze avtomatskega projektiranja. Možnost ima spreminjati in popravljati vhodne podatke in tako prilagajati rezultate posameznih obdelav. Za vnašanje večine podatkov sedaj skrbijo koderji, projektant pa vnaša le zahtevnejše podatke.

3. OPIS PROGRAMSKEGA PAKETA

Programski paket za avtomatizacijo projektiranja je sestavljen iz treh delov:

- programi za zajemanje podatkov in vzdrževanje podatkovne baze
- programi za oblikovanje besedil in načrtov
- aplikacijski programi namenjeni konkretno projektiranju zavarovanja cestno - železniških prehodov.

Iz slike 1 je razvidno, da imajo vsi programski deli dostop do skupne podatkovne baze iz katere lahko črpajo podatke. Samo programski podpaket UPDAT, ki ima za nalogo vzdrževati podatkovno bazo, lahko podatke v bazo vstavlja, jih popravlja ali briše. Poglejmo vloge posameznih podpaketov:

3.1. Obdelava besedila

Usotovili smo, da so besedila za opis celotnega sistema zavarovanja in njegovih posameznih delov sestavljena iz standardnih delov, ki se ponavljajo iz projekta v projekt. Naredili smo nekakšen katalog vseh standardnih tekstov in ga shranili v podatkovno bazo. V podatkovno bazo se preko vprašalnikov in programa UPDAT vnesejo tudi vse zahteve projektanta iz katerih standardnih besedil naj bo končni tekst sestavljen. Obdelava poteka v dveh korakih. Najprej se zlepijo vsi standardni deli besedil, se vstavijo v tekst spremenljivke, ki jih poda projektant preko istih vprašalnikov kot je podal oznake standardnih besedil, in tiste spremenljivke, katere program sam avtomatično poišče v podatkovni bazi. V naslednjem koraku ta tekst obdela dokaj močan text procesor, ki oblikuje strani, določa dolžino in širino strani, deli besede, podčrtuje, razmika, izdela kazalo vsebine, stvarno kazalo ... Končni rezultat tega preoblikovanja sta dve besedili. Prvo je primerno za izpis na vrstičnem tiskalniku, drugo pa na električnem pisalnem stroju.

3.2. Izdelava načrtov

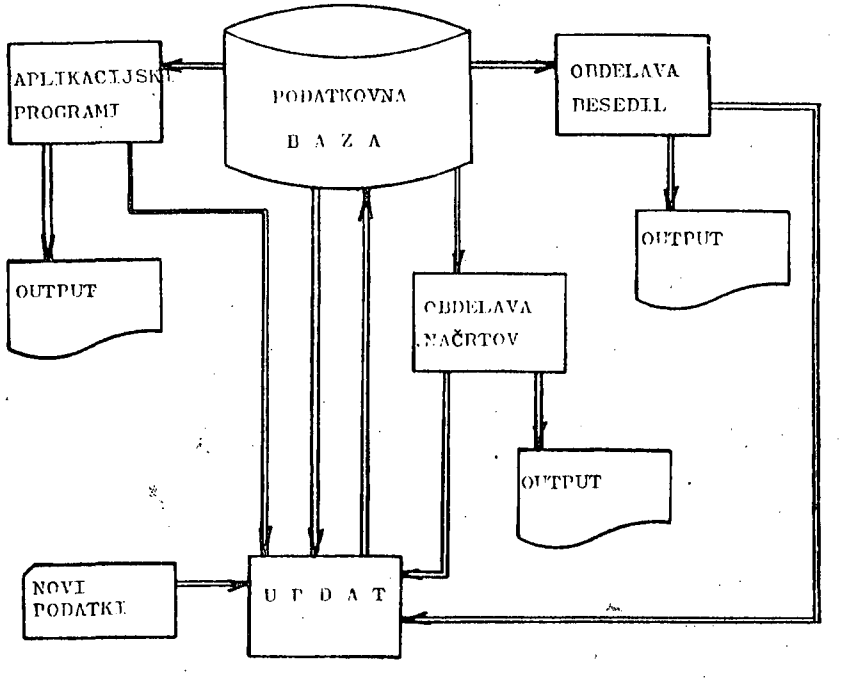
Prvi korak pri izdelavi načrtov je bila standardizacija osnovnih likov in simbolov, ki jih pri načrtih projekta uporabljamo. S pomočjo obstoječega grafičnega paketa na CDC Cyber, ki je sestavljen iz nabora rutin za posamezne elementarne operacije pri risanju, smo sestavili standardne objekte risb, od katerih ima vsak svojo šifro. Nekateri načrti se izdelujejo avtomatično, na podlagi podatkov o progi, drugi pa se morajo ročno zakodirati preko ustreznega formularja v katerem se našteje vse šifre elementov, ki nastopajo na načrtu, in se poda njihova lega, velikost ... Vsi ti podatki so v podatkovni bazi. Program za izdelavo načrtov kreira datoteko z direktivami za ploter, ki nato izriše načrt.

3.3. Aplikacijski programi

Ti programi se nanašajo konkretno na zavarovanje cestno - železniških prehodov. To so izračuni različnih tabel o vklopnih točkah, časih zadrževanja, napajanje zavarovalnih naprav, povezave kablov, specifikacije opreme in del s cenami in končno ceno ... Ti programi jemljejo podatke iz podatkovne baze, jih predelujejo in izpisujejo na OUTPUT ali pa preko programa UPDAT spet predelane shranjujejo v podatkovno bazo, kjer so navoljo ostalim programom paketa.

3.4. Programi za zajemanje podatkov in vzdrževanje podatkovne baze

Ta programski podpaket je razdelan v naslednjih poglavjih.



Slika 1. Podatkovni pretok med deli paketa

4. ZAJEMANJE PODATKOV

Pri načrtovanju zajemanja podatkov smo se zavedali, da gre danes razvoj v smeri uporabe terminalov in izločevanja kartic kot osovnega nosilca vhodnih informacij, vendar zaradi omejitve strojne opreme (delamo na računalniku CDC Cyber 172 in naročnik projekta nima svojega terminala) smo morali sprejeti kartice kot vhodni medij.

Osnova za zajemanje podatkov so vprašalniki in formularji (primer vprašalnika za zajemanje splošnih podatkov o projektu in vprašalnika o vzdolžnem profilu proge vidimo v prilogi). Nekatera polja so že izpolnjena z oznakami značilnimi za vsak vprašalnik:

- prvih 6 mest je šifra vprašalnika (za osnovne podatke o projektu je to CPPRO2),
- eno znakovno mesto za tip zapisa, ki nam sledе na spremenljivke , ki sestavljajo ta zapis, pove kateri zapisi imajo enako sestavo. To omogoča enako kontrolo teh podatkov. V našem primeru vprašalnika o osnovnih podatkih o projektu so to številke 0, 1, 1, 1, 1, 1, 2, 3 kar pomeni, da imajo zapisi z oznako 1 vsi enako kontrolo.
  zapisi z oznako 1 vsi enako kontrolo).
- dvomestna zaporedna številka karice na vprašalniku. V našem primeru so to številke od 01, 02 ... do zadnje , ki je vedno 99.

Vse te oznake so že predtiskane na vprašalnikih , tako da za projektanta nimajo pomena. Pomembne so le za program UPDAT, ki vprašalnike obdela in shrani podatke v bazo. Naloga projektanta oziroma koderja je, da izpolni samo prazna polja. Ko so ti vprašalniki izpolnjeni jih zluknjamo na kartice (pod tekstom so številke, ki povedo v katero kolono naj se luknja). Te podatke s kartic obdela poseben program za vzdrževanje podatkovne baze, program UPDAT. Vprašalniki so tako sestavljeni, da se podatki zajemajo le enkrat in so v bazi dostopni vsem programom paketa.

5. OPIS PODATKOVNE BAZE

Naš paket sestavljajo vglavnem programi pisani v PASCALu in le nekaj v FORTRANu. Skupaj obsegajo 30.000 kartic. Vsi ti programi operirajo nad skupno podatkovno bazo, ki obsega od 600.000 do 700.000 znakov. Ugotovili smo, da bi ti programi potrebovali 70 do 80 datotek, ki bi morale biti vse dostopne paketu. S stališča operacijskega sistema jih ni ugodno hraniti kot *ločene datoteke* . Odločili smo se seg - mentirano datoteko v smislu PASCALa . Tako nam sedaj en segment te datoteke predstavlja eno izmed potrebnih datotek.

Vsak segment ima ime dolžine 6 znakov, ki je del vsakega zapisa v segmentu. Zaradi različnega pomena podatkov v posameznih segmentih smo se odločili, da podatke hranimo kar v obliki kartičnih slik.

Ključni segment v naši podatkovni bazi se imenuje AAAAAA in predstavlja seznam vseh trenutno prisotnih segmentov v bazi. Vsakič, ko kakšen program potrebuje nek segment iz baze, se v tem segmentu, našem kazalu, poišče naslov iskanega segmenta.

Zapisi v vsakem segmentu so urejeni po sortirnih ključih, ki pripadajo temu segmentu. Vsak segment ima lahko enega ali več sortirnih ključev, ki so lahko pri različnih segmentih na različnih delih zapisa. Podatki o sortirnem ključu segmentov so shranjeni v posebnem segmentu z imenom SORTKL.

Segmenti se razlikujejo tudi po tem, da zahtevajo različne kontrolne postopke za verifikacijo podatkov. Vse informacije, kako naj se kontrolirajo spremenljivke v zapisih v vsakem segmentu, so shranjene v segmentu KONTRL.

Pri vnašanju podatkov v podatkovno bazo se nam včasih zgodi, da se nekatera polja na zaporednih karticah ponavljajo. Da bi se izognili odvečnemu pisanju in luknjanju, nam zapisi v segmentu PREPIS definirajo katera polja na kartici se bodo izpolnila z vsebino predhodne kartice, če jih pustimo prazna.

Včasih želimo izpisati na listing kakšen vprašalnik za katerega imamo podatke že v bazi, v ta namen nam služi segment IZPISI, kjer je zakodirana oblika izpisa za vse vprašalnike.

V podatkovni bazi morajo biti tudi segmenti, ki predstavljajo šifrante. Naprimer CPSIF1 - šifrant elementov vzdolžnega profila, CPSFPR - šifrant prog.

Sledijo segmenti v katere vnaša podatke projektant oziroma koder s pomočjo vprašalnikov in formularjev. To so naprimer segmenti: CPPRO2 - osnovni podatki o projektu, CPVZD1 - podatki o elementih vzdolžnega profila proge.

Obstajajo še segmenti, katere generirajo aplikacijski programi in služijo za nadaljne obdelave v drugih programih. Primer takega segmenta je CPTABP v katerem so shranjeni vsi podatki o vklopnih točkah, času zadrževanja vlaka ...

Za vzdrževanje take podatkovne strukture smo naredili program s katerim je mogoče podatkovno bazo ažurirati, izvajati kontrolo nad novimi podatki, vsebino baze izpisati... Ta program se imenuje UPDAT in le on lahko spreminja vsebino podatkovne baze. Vsi ostali programi iz paketa pa imajo samo dostop do podatkov, ne morejo jih pa spreminjati. Na ta način smo se obvarovali pred vnašanjem napak v našo podatkovno bazo.

V prilogi je podana slika dela podatkovne baze in sicer segment za kontrolo vhodnih podatkov.

6. PROGRAM UPDAT

Program UPDAT je pisan v PASCALU (3.000 kartic). Namenjen je vzdrževanju podatkovne baze.

Vhodne datoteke so:
- podatkovna baza
- RWD datoteka novih podatkov , ki jih želimo vstaviti in direktiv za brisanje
- SPL datoteka posebnih ukazov za izpis

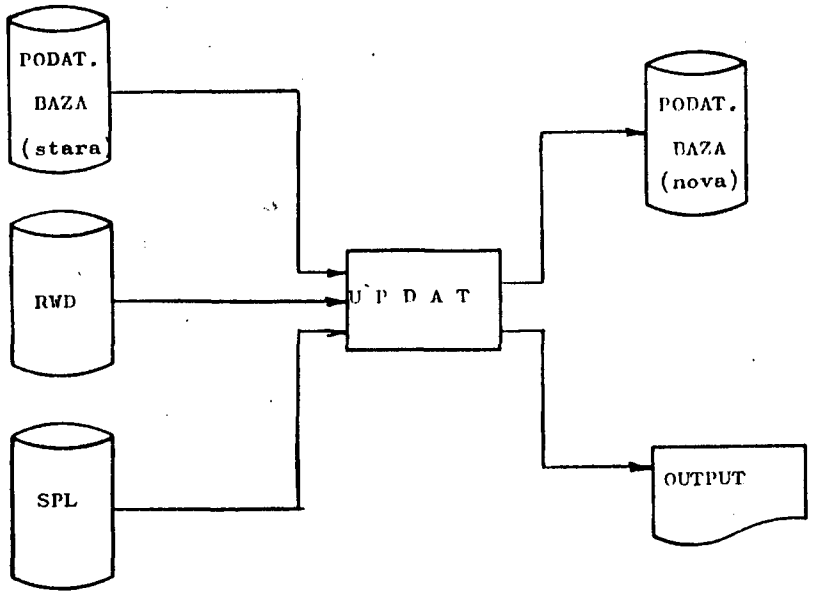
Izhodne datoteke:
- spremenjena baza podatkov
- OUTPUT izpis opozoril o morebitnih napakah v vhodnih podatkih.

A. OSNOVNI PODATKI O PROJEKTU C P P R Ø 2 0 0 1

1. Št. projekta:
2. Št. vprašalnika: 3. Datum:
4. Naziv projekta:
5. Izdajatelj projektne naloge (1) C P P R Ø 2 1 0 2
6. Izdajatelj projektne naloge (2) C P P R Ø 2 1 0 3
7. Izdajatelj projektne naloge (3) C P P R Ø 2 1 0 4
8. Izdajatelj projektne naloge (4) C P P R Ø 2 1 0 5
9. Izdajatelj projektne naloge (5) C P P R Ø 2 1 0 6

B. PODATKI O PROGI C P P R Ø 2 2 0 7

1. Proga:
2. Šifra proge:
3. Odsek:
4. Šifra odseka: C P P R Ø 2 3 9 9
5. Hitrosti vlakov (km/h) maksimalna: (Vmax)
minimalna: (Vmin)
6. Največje možno število osi:
7. Največja možna dolžina vlaka (m):
8. Zavorna razdalja (m):
9. Število voženj v 8 urah največjega prometa:
10. Pospešek
11. Številka pogodbe za projektiranje
12. Število prehodov v projektu
13. Začetni kilometer projekta
14. Končni kilometer projekta
15. Oznaka prvega razdelilca v projektu



Slika 2. Vhodi in izhodi programa UPDAT

Program UPDAT omogoča kontrolo in vstavljanje novih podatkov ter popravljanje in brisanje starih podatkov. Preko datoteke SPL lahko zahtevamo tudi izpise želenih vprašalnikov.

Eden najpomembnejših delov programa UPDAT je kontrola vhodnih podatkov. Pri tem se program naslanja na segment KONTRL, kjer dobi informacijo kako naj kontrolira posamezne podatke. Zahteve za kontrolo vsebine zapisov podajamo glede na segment v katerem je zapis in glede na tip zapisa v tem segmentu. Vsaki spremenljivki iz vprašalnika, ki jo kontroliramo, pripada vsaj en kontrolni zapis. Sestava kontrolnega zapisa je naslednja:

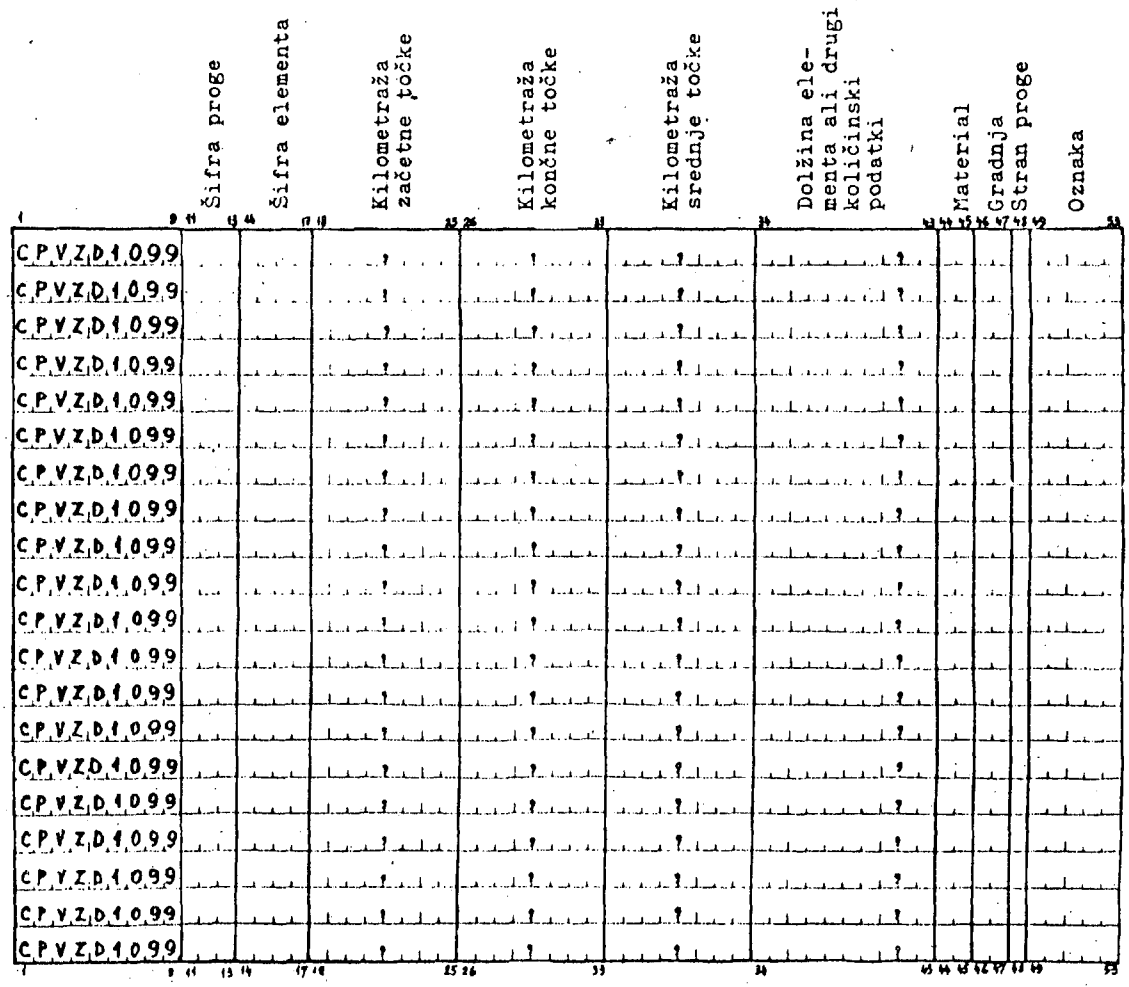
1 .. 10 - standardna oznaka segmenta (KONTR099)
11 .. 16 - šifra segmenta v katerega spada spremenljivka , ki jo želimo kontrolirati
17 - tip zapisa v segmentu
18 .. 19 - začetek polja spremenljivke na kartici nad katero želimo izvajati kontrolo
20 .. 21 - dolžina spremenlji v ke v znakih
22 - tip kontrole, ki jo želimo izvršiti nad spremenljivko. Možne imamo štiri kontrole:
1 - intervalna kontrola : na mestih od 23 do 32 podamo spodnjo mejo intervala , od 33 do 42 pa zgornjo mejo intervala. S tem želimo ugotoviti, če je podano število v dopustnih mejah . Naprimer maksimalna hitrost vlaka je od 0 do 90 km/uro.
2 - znakovna kontrola : trenutno imamo 12 različnih podmnožic znakov . Kontroliramo lahko, če je znak v določeni pod - množici (zakodirano od 23 do 32 mesta). Primer te kontrole je preverjanje , če je znak številka, številka, znak iz posebne podmnožice..
3 - kontrola pripadnosti množici Na mestih od 23 do 32 imamo šifro datoteke, segmenta iz katerega program pobere sortirne ključe in ugotavlja ali je dana spremenljivka enaka kakšnemu izmed izbranih ključev . S to kontrolo naprimer pri vstavljanju osnovnih podatkov o posameznem prehodu lahko ugotovimo , če je ta prehod sploh vključen v kakem projektu ali če je šifra za nek element vzdol - žnega profila prava - je v šifrantu vzdolžnega profila.
4 - posebne kontrole : trenutno imamo 4 take kontrole, ki so programsko realizirane (so del UPDATa). Na mestih od 23 do 32 navedemo stevilko posebne kontrolne procedure.

Tako oblikovan program za kontrolo vhodnih podatkov nam dopušča zelo enostavno dodajanje ali spreminjanje kontrole nad posameznimi spremenljivkami. Pri dodajanju novih kontrol moramo posegati samo v podatkovno bazo, v kontrolni segment. Program sam ostane nespremenjen. Program UPDAT iz vsebine podatkovne baze ugotovi kako mora kontrolirati vhodne podatke. Če program ugotovi na vhodni kartici napako, to kartico izloči, izpiše na OUTPUT in označi mesto napake ter poda njen tip. Na koncu poda tudi statistiko o številu in tipih napak. Pravilni podatki se vstavijo v podatkovno bazo. Tak pristop k zasnovi kontrole smo izbrali zaradi ogromne količine podatkov, ki zahtevajo različne kontrole, in zaradi še nejasnih zahtev po konkretnih kontrolah pri snovanju programa.

Druga naloga programa UPDAT je vstavljanje, brisanje in popravljanje podatkov v segmentih. V ta namen uporabljamo za direktive 79 in 80 stolpec na kartici. Če sta stolpca prazna pomeni, da želimo zapis vstavljati. Če je na 79 mestu zvezdica (*) pomeni, da bo 80 mesto povedalo ali gre za brisanje enega podatka (B) ali celega intervala (D) . Popravljamo tako, da ponovno zluknjamo vprašalnik, ki ga spet obdela UPDAT. Ko program ugotovi, da ima ta vprašalnik že v podatkovni bazi (glede na sortirne ključe), ga zamenja z novim. UPDAT zagotavlja, da pri brisanju ne pride do tega, da bi v podatkovni bazi ostali nepopolni podatki.

7. ZAKLJUČEK

Celotni programski paket je paketno orientiran, kajti zaradi pomanjkanja strojne opreme pri naročniku projekta nismo mogli uresničiti ideje o interaktivnem delu, ki je veliko prikladnejše za tako vrsto obdelav. Nadaljni razvoj projekta pa gre prav v to smer, paket prilagoditi interaktivnemu delu, kajti naročnik je dobil svojo strojno opremo, ki tako delo omogoča. Uvedba interaktivnega dela bo nekoliko vplivala na podatkovno bazo in zahtevala manjše spremembe programa UPDAT, ki to bazo vzdržuje.



ŠIFRANT ELEMENTOV VZDOLŽNEGA PROFILA

PRIMJENA EPROM MEMORIJA U PRIKUPLJANJU I OBRADI PODATAKA DOBIVENIH IZ PROCESA

MARIO ŽAGAR

ELEKTROTEHNIČKI FAKULTET, ZAGREB
ZAVOD ZA REG. I SIGN. TEHNIKU, ZAGREB

UDK: 681.327.28

Bitan dio uredjaja za prikupljanje podataka iz različitih procesa je memorija za pohranjivanje podataka. Korištenje EPROM memorije kao zamjene za masovne memorije u nekim specifičnim uvjetima ima niz prednosti pred klasičnim masovnim memorijama. Da bi se to moglo koristiti, potrebno je realizirati "ON- LINE" programiranje EPROM memorija. Jedan mogući pristup i način rješavanja dan je u ovom članku.

USING EPROM MEMORIES IN DATA LOGGING SYSTEMS: The important part of data logging system is mass memory. Using EPROM memories as a replacement for classical file storage (magnetic tapes,etc.) in some specific cases has many advantages. For this purpose an "ON - LINE" programming of EPROMs must be realized. One of the possibilities for realizing such a system is described in this article.

1. UVOD

O korištenju EPROM (Erasable Programmable Read Only Memory) memorija, načinima zapisivanja, uredjajima za njihovo programiranje dosta je rečeno i napisano. U ovom članku bit će razmotren jedan specifičan vid korištenja EPROM memorija kao zamjene za masovne memorije tamo gdje je to moguće.

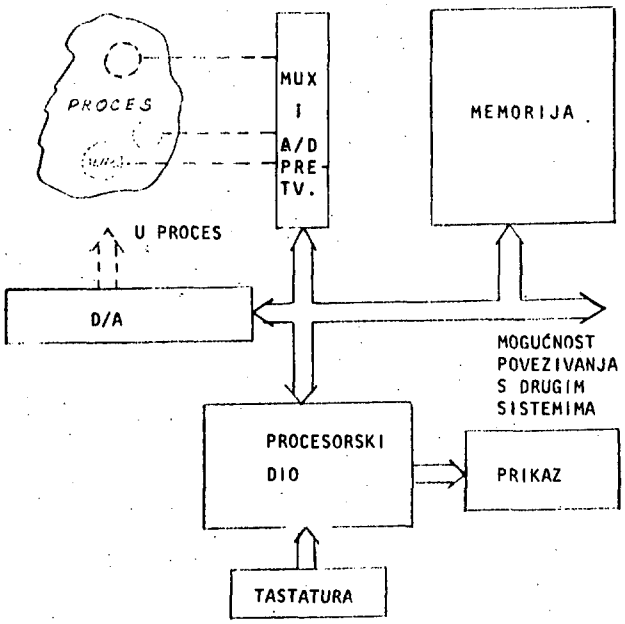
Kod sistema za prikupljanje podataka iz procesa, njihovo pohranjivanje u memoriju te kasniju analizu, susrećemo se sa nizom karakterističnih dijelova sistema kao što su: elementi za pretvaranje različitih vrsta signala u električne (pritisak, temperatura, protok, vlažnost, itd.) , zatim elementi za pretvorbu analognih veličina u digitalne i njihovo unošenje u memoriju, elementi za obradu podataka te elementi za ručno unošenje podataka, upravljanje radom i prikaz podataka (sl.1).

Teži se da takovi sistemi budu što intelligentniji kako bi informacije koje od njih dobivamo bile u jednoj sažetoj formi tj. da iz velikog broja podataka koji udju u sistem dobijemo samo najbitnije i najkorisnije informacije u odredjenom trenutku.

Postupak prikupljanja podataka i njihovog pohranjivanja započinje slanjem električnih impulsa iz različitih senzora, njihovom A/D pretvorbom, pamćenjem na odredjenim medijima kao što su to magnetske trake, kazete i prikazom na ekranu (štampanjem) i sl.

Vrlo često se ti podaci koriste u kasnijoj obradi pa moraju biti memorirani tako da budu prikladni za daljnju obradu. U velikim i složenim procesima kojima upravljaju veći računski strojevi sa odgovarajućim popratnim elementima kao što su masovne memorije, periferijske jedinice, izvori napajanja, linije za prijenos podataka i sl. moguće je pohranjivanje podataka na masovnu memoriju ali i njihova trenutna obrada i interpretacija te intervenriranje u samom procesu.
U ovom članku to neće biti razmatrano.

Razvojem "mikroračunarske tehnologije" dolazi u obzir upravljanje procesima koji nisu toliko složeni, a koji zbog svojih karakteristika nisu mogli biti upravljani klasičnim računalima. Postoji niz takvih primjena kao npr. saobraćajna vozila, poljoprivredni strojevi, kućanski aparati, razne meteorološke i druge stanice na



Slika 1

usamljenim mjestima, prijenosni uredjaji za prikupljanje odredjenih podataka i uzimanje uzoraka na terenu (kao što su npr. aparati za mjerenje osvjetljenja, temperature, vlage, zagadjenja okoline, opterećenja, frekvencije saobraćaja i sl.).

Najbitnije karakteristike koje moraju zadovoljavati uredjaji za upravljanje, prikupljanje i obradu podataka u tim procesima su:
- da budu malih dimenzija
- mala potrošnja električne energije
- otpornost na nepovoljne uvjete rada (termičke, mehaničke i dr.)
- da budu jeftini s obzirom na cijenu cijelog sistema u koji se ugradjuju
- da budu jednostavni, ali da imaju velike mogućnosti u prikupljanju podataka iz procesa, pohranjivanju, obradi.

Većina tih problema riješena je korištenjem mikroprocesora i dodatnih komponenata u izgradnji takovih uredjaja. Za ispis podataka (gdje je to potrebno) koriste se mali video displeji ili štampači, a za masovno pohranjivanje podataka jedinice magnetskih kazeta. Medjutim, one predstavljaju velik problem u radu cijelog uredjaja i zbog svoje veličine, potrošnje elektr. energije, male otpornosti na termalna, mehanička i druga naprezanja, veće mogućnosti kvarova i oštećenja zbog pokretnih dijelova u njima, a i same cijene uredjaja za magnetski zapis.

Postoje procesi gdje je nužno da budu uredjaji za zapisivanje podataka na magnetske kazete i njihovo čitanje (zbog vrlo velike količine podataka koja se mora prikupiti). Medjutim isto tako postoji niz procesa gdje ti uvjeti ne moraju biti zadovoljeni. U tom slučaju teži se za iznalaženjem drugih, pogodnijih oblika za memoriranje podataka.

2. JEDAN VID KORIŠTENJA EPROM MEMORIJA KAO ZAMJENE ZA MASOVNU MEMORIJU

EPROM memorije su memorije u koje se može upisivati podatke dovodjenjem odgovarajućih naponskih nivoa u odredjenim vremenskim intervalima. Ti podaci ostaju tako dugo zapisani u memoriji dok se ne izbrišu ultravioletnim zračenjem dakle kad se u tu memoriju upišu odredjeni podaci, oni ostanu sačuvani ako nestane napajanja. Cijela memorija može se izvaditi iz odgovarajućeg podnožja i premjestiti u neki drugi uredjaj i tamo koristiti zapisane podatke. Kada podaci postanu beskorisni, obriše se cijela memorija (ultravioletnim zračenjem u odredjenom vremenskom periodu) i opet može poslužiti za zapisivanje novih podataka.

Danas postoji čitava serija takvih memorija koje se po svojim karakteristikama sve više usavršavaju. Trenutno najpopularnije EPROM memorije (1Kx8, 2Kx8, 4Kx8 a najavljuju se i 8Kx8) vrlo su pogodne za upisivanje podataka u njih bez "posebnog hardvera". Baziraju se na karakteristikama Intelove EPROM memorije I2716 (samo jedan izvor napajanja od +5 V za normalan rad te jedan dodatni izvor za programiranje (+25 V) koje se vrši dovodjenjem TTL impulsa odredjenog vremenskog trajanja na odredjeni pin memorije.

Postoje dva osnovna vida korištenja tih memorija. Kao memorije za razvoj korisničkih programa u fazi gradnje i testiranja odredjenog mikroprocesorskog sistema. Mogli bismo to nazvati i "OFF- LINE" primjena u kojoj se koriste posebni uredjaji (programatori) za upisivanje programa koji se zatim testiraju na konkretnim sistemima, uočavaju se pogreške, izbrišu se memorije i u njih nanovo upisuju modificirani programi. Kad je razvoj završen, zamijene se EPROM memorije sa PROM ili ROM memorijama (ovisno o broju primjeraka koji se koriste).

Drugi vid je korištenje EPROM memorije za direktno prikupljanje podataka ("ON - LINE") i njihovo memoriranje za trenutnu i kasniju obradu (mogu se prikupljati podaci, a isto tako mogu se vršiti izmjene nad parametrima programa i programskim odsječcima pod čijim nadzorom se odvija neki proces i na taj način može se izvršiti optimiziranje i prilagodjavanje upravljanja procesom za vrijeme dok on traje.

Kao što je već prethodno navedeno, danas postoje memorije s kapacitetom 32Kbita, a najavljuje se i 64Kbita u jednom čipu pa korištenjem odgovarajućeg broja čipova možemo sasvim zadovoljiti potrebe masovne memorije u nekim manjim sistemima.

3. MOGUĆNOSTI REALIZACIJE SISTEMA ZA PRIKUPLJANJE PODATAKA I NJIHOVO TRAJNO PAMĆENJE

Realizacija unošenja podataka u računalo (nakon što se oni pretvore u digitalni oblik) vrši se ili preko linija za podatke ili preko ulaznih linija mikroračunarskog sistema. Načini pohranjivanja podataka u cilju trajnog pamćenja mogu biti različiti.

U slučaju korištenja programabilnih memorija potrebno je zadovoljiti odredjene tehničke karakteristike koje propisuju proizvodjači tih memorija.
Izvedba programiranja EPROM memorija može biti različito riješena, ali uvijek postoji neki kompromis. Ako oslobadjamo mikroprocesor tog posla i izgradimo sklopovski programator, povećali smo broj komponenata cijelog uredjaja, a time i fizičku veličinu, cijenu, potrpšnju električne energije i dr. medjutim dobra strana tog rješenja je u tome da mikroprocesor može za vrijeme programiranja memorije izvršavati neki drugi posao.

Rješenja kod kojih se koristi djelomično upravljanje programiranjem memorija preko mikroprocesorskog sistema a djelomično dodatnim sklopovima (lit. 1 i 2) smanjuju broj komponenata medjutim nisu izvediva kod svih mikroprocesorskih sistema (koriste se adresne linije i linije za podatke za vrijeme upisa podataka u memoriju, a za to vrijeme mikroprocesor se dovede u stanje čekanja (WAIT). Vremenski impuls od 50 milisekundi potreban za programiranje generira se sklopovski preko monostabila. Mikroprocesor je potpuno zauzet za vrijeme programiranja memorije.

Za rješenje koje se predlaže u ovom članku potrebno je da mikroprocesorski sistem ima tri 8-bitna ulazno/izlazna porta i sklop koji će ovisno o TTL ulaznom nivou prespajati na izlaz +5 ili +25 V (realizacija može biti tranzistorska ili pomoću "REED" releja ili na neki drugi način).
Svi ostali problemi oko programiranja memorije rješavaju se programski (generiranje vremenskog impulsa odredjenog trajanja - 50 msek., slanje svih potrebnih podataka za programiranje na U/I linije koje se koriste u ovom slučaju, kontrola ispravnog upisivanja podataka, način rada - čitanje iz memorije ili upisivanje u memoriju i dr.). Prednosti ovog rješenja su:

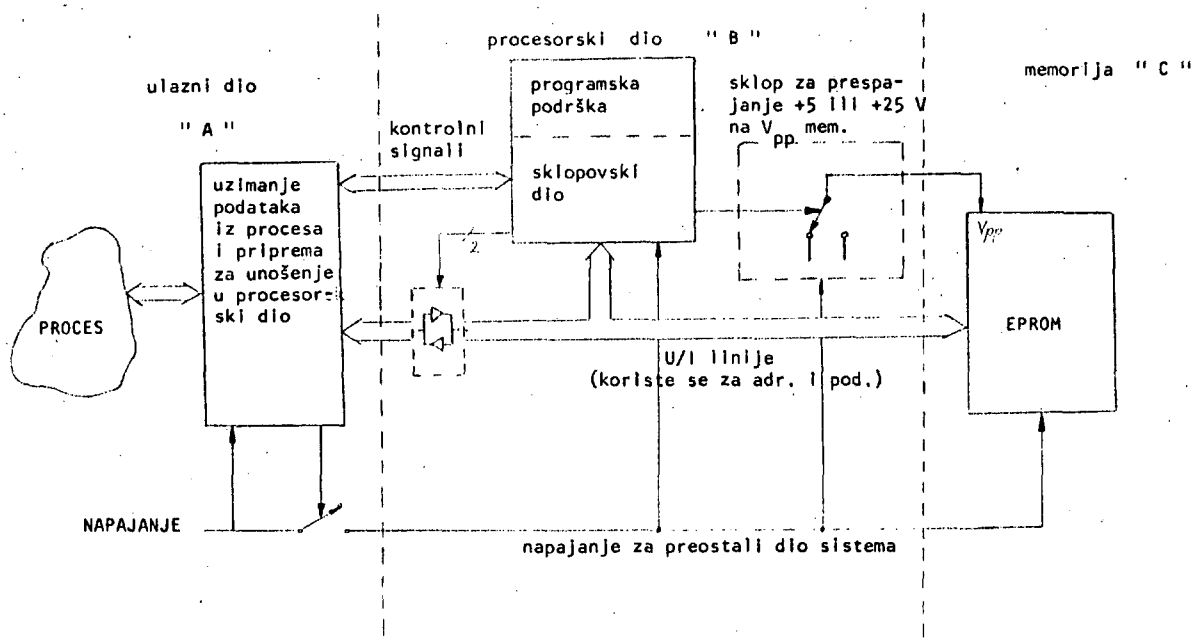
- minimalan dodatni hardver
- mikroprocesor može obavljati neke druge poslove za vrijeme upisa podataka u memoriju (uz uvjet da se ne naruši programsko generiranje vremenskih impulsa)
- univerzalnost primjene (budući da se komuniciranje s EPROM memorijom vrši preko tri U/I porta, može se koristiti sa bilo kojim tipom mikroprocesora uz programsku modifikaciju, a bez hardverskih zahvata
- ti isti portovi mogu se koristiti kao ulazno/izlazne linije prema procesu (za vrijeme dok se ne koristi EPROM memorija), ako im se dodaju "bidirectional TREE-STATE buffers".

Nedostatak tog načina rješenja je:
- dio sistemske memorije potrebno je rezervirati za program koji upravlja programatorom
- budući da se vremenski impulsi generiraju programski, potrebno je da mikroprocesor ima stabilan oscilator (kristal)
- promjena brzine rada mikroprocesora zahtijeva i promjene u programskoj petlji koja generira vremenske impulse (inače može doći do većih oštećenja EPROM memorije koja se programira).

Jedan mogući način realizacije sistema za prikupljanje podataka i njihovo trajno pamćenje prikazan je na sl.2. Izmjene u konkretnim slučajevima ovise o vrsti procesa, većem ili manjem broju ulazno/izlaznih linija, složenosti programa koji upravlja radom cijelog sistema, veličinom EPROM memorije i dr.

Podaci koji se unose u računalo ovise o procesu iz kojeg se uzimaju. To mogu biti jednostavni digitalni podaci ("ON - OFF"), analogni podaci (koji se moraju pretvoriti u digitalni oblik prije unošenja u računalo), broj odredjenih dogadjaja (npr. brojenje saobraćaja, nuklearni procesi), podaci u odredjenim vremenskim intervalima i sl.

Sklopovi iz grupe A (sl.2) brinu se da te podatke dovedu u prikladan oblik za unošenje u računalo. Za vrijeme dok se o podacima brine dio A, dijelovi sistema B i C isključeni su iz napajanja. Na taj način utrošak električne energije je minimalan.
U trenutku kad je A dio gotov sa pripremom podataka, uključuje se preostali dio sistema na napajanje. Mikroračunalo se inicijalizira i počinje sa radom prema zadanom programu. Podaci iz ulaznog dijela A unose se u procesorski dio sistema. Nad njima se (kao i nad prethodno upisanim podacima u memoriju do kojih je vrijeme pristupa jednako za bilo koji podatak) izvršavaju odredjene matematičke i logičke operacije (ako je to potrebno). Mikroprocesor na temelju programskih uputa odlučuje koje podatke treba spremiti u EPROM i što treba poduzeti prema procesu. Nakon što je faza upisivanja podataka u EPROM i dijaloga sa procesom završena, daje se nalog ulaznom dijelu A da isključi preostali dio sistema iz napajanja i nastavi sa prikupljanjem podataka.
Ovakvom organizacijom rada omogućuje se minimalna potrošnja električne energije, a da pri tome svi podaci budu trajno zapamćeni.
Uredjaj je samostalan i potrebna je intervencija čovjeka samo u trenucima kada se želi zamijeniti popunjenu EPROM memoriju sa praznom.



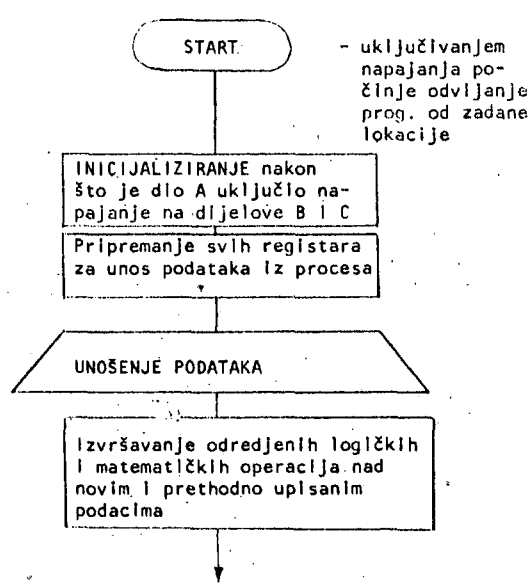
Slika 2

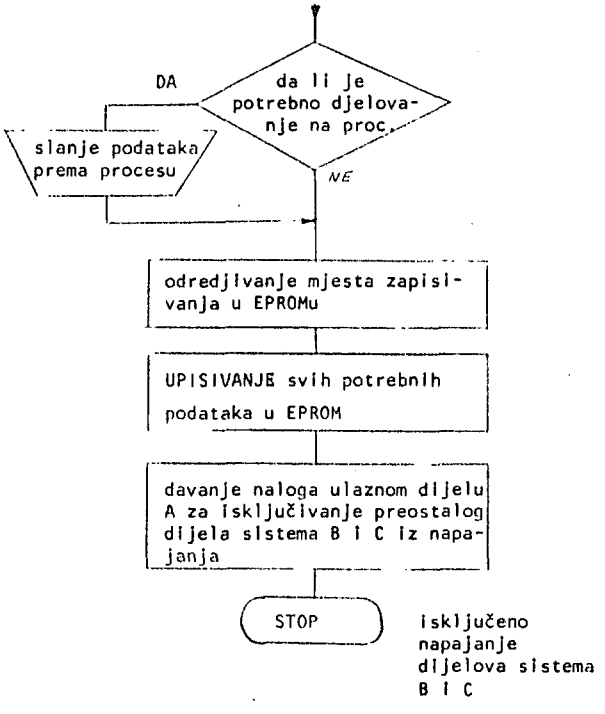
4. PROGRAMSKA PODRŠKA UREDJAJA ZA PRIKUPLJANJE I PAMĆENJE PODATAKA

Osim standardne programske podrške koja omogućava komunikaciju izmedju ulaznog dijela A i procesorskog dijela B (sl. 2), eventualnih matematičkih i logičkih operacija nad podacima te programske podrške za upravljanje radom programatora, ovdje se javlja potreba za jednim specifičnim programom koji će brinuti o popunjenosti memorije, slobodnim lokacijama (u koje još nije ništa upisano), pretraživanju memorije.
Ovo je potrebno zbog toga jer nakon što se sistem B i C uključi na napajanje ne postoji nikakva trenutna evidencija o tome koji je podatak zadnji upisan, koji su podaci poništeni, koje su lokacije memorije slobodne za upis novih podataka i dr.

Taj se problem može riješiti posebnom organizacijom zapisa u EPROM memoriju.
U "čistom" EPROM-u u koji još nisu upisivani podaci, sadržaji svih lokacija memorije su FF_{16}. Princip programiranja je u tome da se bitovi čija je vrijednost jednaka "1" ne mijenjaju. Programiraju se samo bitovi memorije čija vrijednost je "0".

Dakle jedan od mogućih načina na koji jednostavno možemo otkriti slobodnu lokaciju u koju se može izvršiti upis novog podatka je taj da zabranimo upis podatka FF_{16}.
Prije nego upišemo novi podatak, pretražujemo memoriju tako dugo dok ne pronadjemo kombinaciju FF_{16} i ona nam označava da je od tog mjesta pa na dalje memorija spremna za upisivanje novih podataka.
Na sličan način možemo odrediti da npr. kombinacija 00_{16} znači izbrisane (poništene) podatke. Neka druga kombinacija može nam označavati početak (kraj) bloka podataka, datum upisa, broj podataka i sl. Drugim riječima treba kreirati organizaciju i baratanje sa podacima koji se upisuju u EPROM kako bismo iz upisanih podataka na jednostavan način dobili sve potrebne informacije.
Gruba programska koncepcija cijelog uredjaja može se vidjeti iz blok dijagrama na slici 3.
Konkretan program za odredjeni mikroprocesor ovisit će o specifičnostima odredjenog problema koji treba riješiti.





Slika 3

5. ZAKLJUČAK

Korištenje EPROM memorije kao zamjene za masovne memorije u nekim specifičnim uvjetima (za područja gdje je broj podataka i frekvencija prikupljanja mala, a potrebna je jeftina i jednostavna memorija koja trajno pamti podatke - brojenje saobraćaja, mjerenja na terenu, meteorološka ispitivanja na zemlji i u zraku, praćenje industrijskih procesa, prikupljanje podataka o radu uredjaja na brodu i dr.) ima niz prednosti:

- postoji mogućnost komuniciranja sa više mikroprocesorskih sistema (zajednički podaci i njihov prikaz -aerodromi, pošte, banke, kolodvori i dr.)
- brzina pristupa do bilo kojeg podatka je jednaka (što je važno zbog pretraživanja)
- sve veći kapacitet uz manju cijenu
- male dimenzije (na jednu "EURO" karticu stane više od 10 Kbajta podataka)
- nema pokretnih dijelova pa je manja osjetljivost na mehanička i termička naprezanja
- dok se ne koristi, može biti potpuno isključena iz napajanja, a u normalnom radu ima malu potrošnju električne energije uz samo +5 V napajanja.

Naravno, postoji i niz primjena gdje se zahtjeva memoriranje velikog broja podataka u kratkim vremenskim intervalima i tu se moraju koristiti klasične masovne memorije (magnetske trake, kazete i dr.).

Realizacijom samog uredjaja za "ON - LINE" zapisivanje podataka u EPROM memoriju nisu riješeni i svi problemi oko upotrebe takvog tipa uredjaja za prikupljanje podataka.
Ovdje je razmatran samo manji dio svih mogućih problema koji se javljaju kod široke upotrebe odredjenog uredjaja (nabavka komponenata, razvoj programa, obuka za ispravno korištenje uredjaja, obrada dobivenih podataka, brisanje iskorištenih memorija i priprema za ponovnu upotrebu i dr.).

Razvoj tehnologije i radovi na tom području pokazat će opravdanost takvog načina rješavanja problema oko prikupljanja, pohranjivanja i obrade podataka dobivenih iz procesa.

LITERATURA:

1. G. Godliman: Using EPROMs for file storage, Microprocessors and Microsystems, V 2 No 6 , Dec. 1978.
2. D. Passey: Low-cost processor package programs EPROMs, Electronics , June 7, 1979.
3. Intel Data Catalog 1979.
4. MOSTEK Z80, Technical Manual, CPU, 1978.
5. MOSTEK Z80, Technical Manual, Parallel I/O Controller, 1978.
6. F8 Users Guide, 1976.

O JEDNOM ALGORITMU ZA NALAŽENJE NULA FUNKCIJA

D. B. POPOVSKI

TEHNIČKI FAKULTET, BITOLA

UDK: 681.3:51

U radu je prezentiran jedan algoritam za nalaženje nula funkcija u kome se koristi kombinacija metoda sečice i bisekcije. Algoritam je realizovan u obliku opšteg potprograma na programskom jeziku FORTRAN IV. Pokazano je da je predloženi algoritam efektivniji od algoritma Paunovića i Slavića koji koristi kombinaciju metoda n-sekcije, bisekcije i regula falsi.

ON AN ALGORITHM FOR FINDING FUNCTION ZEROS. *In this paper an algorithm for finding function zeros is presented in which a combination of the secant and bisection methods is used. The algorithm is realized in the form of a FORTRAN IV subroutine. It is shown that this algorithm is more effective than Paunović-Slavić algorithm in which a combination of n-section, bisection and regula falsi methods is used.*

Dj.S.Paunović i D.V.Slavić[1] predložili su jedan algoritam za nalaženje nula funkcija u kome koriste pogodnu kombinaciju metoda n-sekcije, bisekcije i regula falsi. Predloženi algoritam, realizovan u obliku jednog FORTRAN IV potprograma nazvanog ZERO, daje bolje rezultate od sve tri pomenute metode. Medjutim, autori u svom algoritmu nisu do kraja iskoristili mogućnosti metoda podele intervala i linearne aproksimacije. Oni koriste sporiju varijantu metode linearne aproksimacije - metodu linearne interpolacije odnosno regula falsi kod koje se koriste zadnje dve iteracije čije vrednosti funkcije imaju suprotan znak. Umesto nje, mogli su upotrebiti bržu, interpolaciono-ekstrapolacionu varijantu - metodu sečice kod koje se koriste zadnje dve iteracije bez obzira na znak vrednosti funkcije. Autori koriste metodu n-sekcije intervala iako je ona sporija od bisekcije. Oni naizmenično koriste metode bisekcije i regula falsi. Korišćenje bisekcije u svakoj drugoj iteraciji je suvišno. Bisekciju treba koristiti samo u slučajevima kada je to neophodno - kada metoda sečice daje loše rezultate (usled ekstrapolacije). Imajući sve ovo na umu, u ovom radu, predlaže se sledeći algoritam za nalaženje nula funkcija:

Neka funkcija $f(x)$ ima realnu nulu u intervalu definisanom početnim aproksimacijama x_1 i x_2 t.j. neka je

$$signf(x_1) \neq signf(x_2) \qquad (1)$$

Označiti $x_b = x_1$

(I) *Naći* x_3 *metodom sečice*

$$x_3 = x_2 - f(x_2)\,\frac{x_2 - x_1}{f(x_2) - f(x_1)} \qquad (2)$$

i proveriti dali dobivena vrednost leži u intervalu definisanom aproksimacijama x_2 *i* x_b. *Ako zadnji uslov nije ispunjen, naći* x_3 *metodom polovljenja*

$$x_3 = 0.5\,(x_2 + x_b) \qquad (3)$$

Izračunati $f(x_3)$. *Ako je*

$$signf(x_3) \neq signf(x_2)\ , \qquad (4)$$

staviti $x_b = x_2$. *Zameniti* $(x_1, f(x_1))$ *sa* $(x_2, f(x_2))$ *i* $(x_2, f(x_2))$ *sa* $(x_3, f(x_3))$. *Vratiti se na poziciji* (I).

U opisanom algoritmu metoda sečice se koristi kad god je to moguće. Nađuži interval u kome se nalazi tražena nula funkcije definisan je aproksimacijama x_2 i x_b[2-4]. Kad god metoda sečice daje rezultat koji je van ovog intervala primenjuje se bisekcija. Predloženi algoritam realizovan je u obliku jednog opšteg potprograma nazvanog SB na programskom jeziku

FORTRAN IV - osnovna verzija koja se može koristiti na svakom računaru . Slično potprogramu ZERO, kao kriterijumi zaustavljanja iterativnog procesa usvojeni su postizanje maksimalnog broja tačnih cifara rezultata (testira se mašinska nula rešenja) i eventualno odredjivanje prave nule funkcije. Potprogram SB zauzima 228 16-bitnih reči memorijskog prostora (što je manje od 260 koliko zauzima ZERO).

```
      SUBROUTINE SB(F,A,B,R,I)
      I=2
      C=A
      D=F(C)
      E=B
      G=F(E)
      H=E-C
      IF(D)1,12,2
    1 IF(G)13,14,5
    2 IF(G)5,14,13
    3 C=E
    4 E=R
    5 D=D-G
      R=G*H/D+E
      H=R-E
      IF(H)6,15,7
    6 IF(R-C)8,8,9
    7 IF(R-C)9,8,8
    8 R=(C+E)*0.5
      H=R-E
    9 D=G
      G=F(R)
      I=I+1
      IF(G)10,15,11
   10 IF(D)4,15,3
   11 IF(D)3,15,4
   12 R=C
      RETURN
   13 I=1
   14 R=E
   15 RETURN
      END
```

Opis parametara :

F - *Ime funkcijskog potprograma u kome je definisana funkcija* $f(x)$.

A - *Početna aproksimacija* x_1.

B - *Početna aproksimacija* x_2.

R - *Nula funkcije.*

I - *Broj iteracija odnosno broj poziva potprograma* F(X).

I=1 - *Rešenje nije nadjeno : postupak je prekinut jer uslov* (1) *nije ispunjen.*

Kao potvrdu da je potprogram SB efektivniji od potprograma ZERO u tabeli 1 dati su rezultati nekoliko numeričkih primera . Proračuni su vršeni u proširenoj tačnosti (37 bita u mantisi) na računaru IBM 1130 . Računsko vreme mereno je specijalnim potprogramom za automatsko merenje vremene na računaru IBM 1130[5].

LITERATURA

1. Dj.S.Paunović , D.V.Slavić : *Algoritam za nalaženje tačaka promene znaka vrednosti funkcije* , Informatica 76 - Bled , October 1976, 1 127.
2. R.F.King : *Methods without Secant Steps for Finding a Bracketed Root* , Computing 17(1976), 49-57.
3. D.B.Popovski: *A Hybrid Algorithm for Finding Roots*, Informatica 3/1979, 16-17.
4. D.B.Popovski : *A FORTRAN IV Subroutine for Finding a Bracketed Root* , Informatica 4/1979, 23-24.
5. Računski centar Elektrotehničkog fakulteta u Beogradu: Lične informacije.

Tabela 1. Primeri (ZERO : N=1).

F(X)	A	B	X (ZERO) R (SB)	K+1 (ZERO)	I (SB)	Računsko vreme (s) ZERO	SB
((X-14.)*X+57.)*X-65.	0.	3.	1.926437	15	10	0.138	0.112
	3.	5.	4.393098	14	11	0.127	0.128
	5.	8.	7.680463	18	13	0.168	0.154
EXP(1.-X)*(X-1.)*10.-1.	0.	2.	1.111832	13	11	0.186	0.182
	2.	5.	4.577152	18	8	0.264	0.133
(((X-20.)*X+133.)*X-330.)*X+236.	0.	2.	1.189383	15	11	0.160	0.141
	2.	4.	3.425451	14	10	0.156	0.127
	4.	8.	6.574548	14	12	0.157	0.164
	8.	9.	8.810616	12	12	0.136	0.165
ALOG((0.25*X-2.)*X+4.5)-1.	0.	3.	1.021220	19	8	0.331	0.150
	3.	7.	6.978779	10	6	0.178	0.117
			Suma	162	112	2.001	1.573
			Indeks	145	100	127	100

KONTROLA MIKRORAČUNALNIŠKEGA NAPAJALNIKA

B. MIHOVILOVIĆ
P. KOLBEZEN
P. REINHARDT

UDK: 681.3:621.721.1

INSTITUT JOŽEF STEFAN, LJUBLJANA

Članek obravnava napajalni sistem mikro računalnika, ki ima dva energetska vira: omrežje in baterijo. Mikro računalnik s posebno materialno in programsko opremo nadzoruje napajalnik tako, da je zaščiten pred morebitnim izpadom omrežni napetosti ali pred posledicami nepravilnega delovanja samega napajalnika.

CONTROL OF MICROCOMPUTER POWER-SUPPLY. This article deals with the power supply which it has two power sources: Ac power supply and battery. Microcomputer with special hardware and software controls the power supply, which is protected from eventual loss of Ac power or consequences of false working of power supply.

1. UVOD

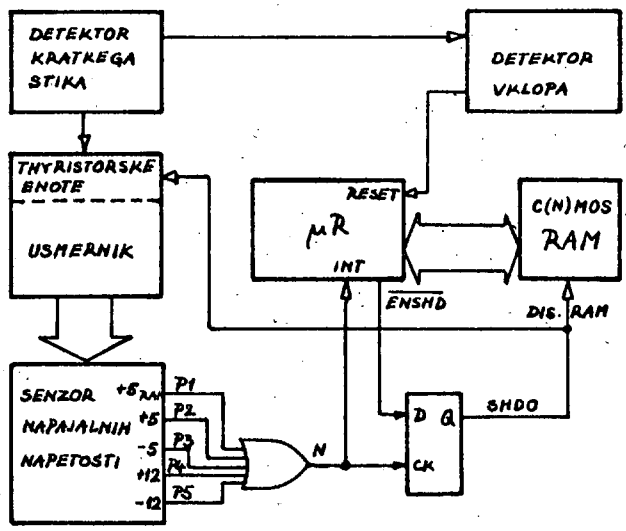
Izguba podatkov, ki so v času izpada napajalne omrežne napetosti shranjeni v RAM pomnilniku, lahko povzroči precejšnje nevšečnosti. Čeprav se sistemski program nahaja v pomnilniškem mediju (ROM, EPROM, masovni pomnilniki), v katerem se vsebina ohranja kljub izgubi napajalne napetosti, je pravilno izvajanje programa in s tem pravilno delovanje sistema odvisno tudi od podatkov, ki so trenutno shranjeni v RAM pomnilniku.

Od dobro grajenega sistema zahtevamo, da ob izgubi napajalne napetosti ohrani vse pomembne podatke v RAM pomnilniku; ob ponovni vspostavitvi napajanja pa μR nadaljuje izvajanje programa pri tistem programskem koraku, pri katerem je bil zaradi napake prekinjen. Omenjeni pomnilniški medij (RAM) pa mora imeti posebno karakteristiko. Ohraniti mora lastnost pomnenja še pri znatno manjši porabi energije, kot jo potrebuje za normalno obratovanje.

V članku je opisana materialna oprema, ki omogoča dodatno baterijsko napajanje na primer statičnih NMOS RAM pomnilnikov 2102 AL ter CMOS RAM pomnilnikov 5101, kakor tudi kontrolno logiko s potrebno programsko opremo za omenjeno zaščito.

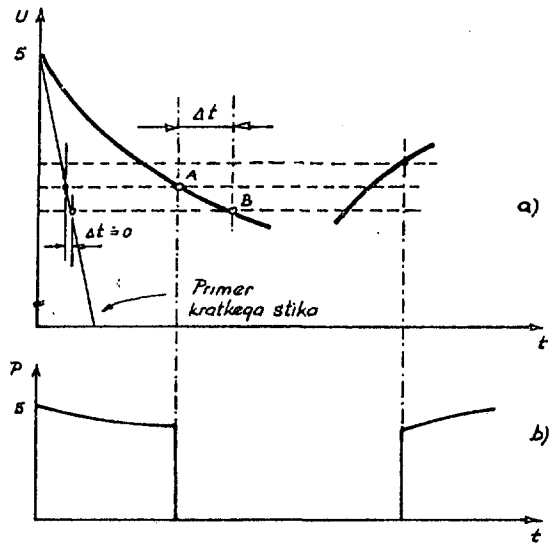
2. Kontrolna logika

Komponente sistema, ki omogoča mikroračunalniško kontrolo napajalnika z dodatnim baterijskim virom, so razvidne iz blokovne sheme sistema na sliki 1.



Slika 1.

Na sliki vidimo, da usmerniška enota s kontrolnim vezjem (detektor vklapa usmernika, senzor velikosti napajalnih napetosti) ne predstavlja več ločene enote, ki bi računalniškemu sistemu nudila zgolj enosmerno napajanje, temveč je tesno povezana z mikroračunalnikom. Usmerniška enota s kontrolnimi signali sporoča mikro računalniku svoje stanje, le-ta pa tem signalom ustrezno krmili njeno delovanje. Napajalna enota in mikro računalnik medsebojno komunicirata v času Δt pod pogojem, da se ne pojavi izpad katerekoli napajalne napetosti (kratek stik), ki bi onemogočil omenjeno komunikacijo. Na sliki 2 je prikazana karakteristika izpada napajalne napetosti +5V.



Slika 2.

Elektrolitski kondenzatorji v usmerniškem filtru zagotavljajo potrebno napajanje µR v času Δt (po izpadu napajalne napetosti). Čas Δt je približno 1ms. V tem času mora mikroprocesor opraviti naslednje:

- zavarovati informacije v RAM pomnilniku (save RAM)
- ugotoviti in shraniti stanje sistema, ki je skupaj s stanjem pomnilnika RAM določeno z vrednostjo programskega števnika (save state).

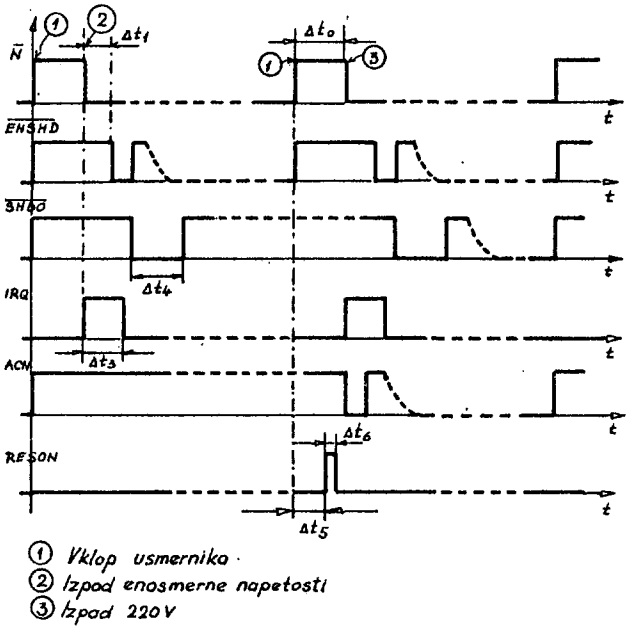
Senzor napajalne napetosti, ki je zgrajen iz napetostnih komparatorjev, "opazuje" napetosti v tolerančnem območju Unap ±10%. Vsako odstopanje iz tolerančnega območja komparator zaznamuje z "logično 0" na svojem izhodu. Disjunkcija signalov na izhodih napetostnih komparatorjev vseh napajalnih napetosti oblikuje logični signal $\bar{N}$, kot prikazuje slika 2b. Signal $\bar{N}$ postavi mikro računalniku zahtevo za prekinitev z absolutno prioriteto. V času Δt_1 steče prekinitveni program, ki poskrbi, da se shranijo v pomnilniški sklad vrednosti akumulatorja, delovnih in statusnih registrov in vrednost programskega števnika, ki jo je imel mikroprocesor v trenutku, ko je bilo prekinjeno izvajanje programa. Prekinitveni program zaključuje ukaz za postavitev stanja ENSHD, ki mora v "konjunkciji" s signalom $\bar{N}$:

- onemogočiti čitanje iz pomnilnika
- izkrmiliti tiristorske enote, ki kratko vežejo izhode usmernika (hitro praznenje elektrolitov)
- vključiti "stand by" za RAM pomnilnike

3. Prekinitveni program

Z naborom instrukcij, ki jih ima µR INTEL 8080 lahko zapišemo prekinitveni program takole:

```
INTR,  DI          /V času izvajanja tega podprograma
                   ne dopuščamo servisiranje ostalih
                   programskih prekinitev.
```



Slika 3.

```
PSHX  HL       /
PSHX  DE       /V sklad shranimo vrednosti aku-
PSHX  BC         mulatorja, delovnih registrov
PSHX  PSW        in statusnega registra.
SP    HL
STHL  STACK    /V pomnilnik shranimo zadnjo
                 vrednost kazalca sklada.
LAI   377      /Zaznamujemo uspešno izvajanje
                 programa.
STA   NSCT
OUT   ENUST    /Ukaz, s katerim se kratko skle-
HLT              nejo izhodi napajalnih napeto-
                 sti.
```

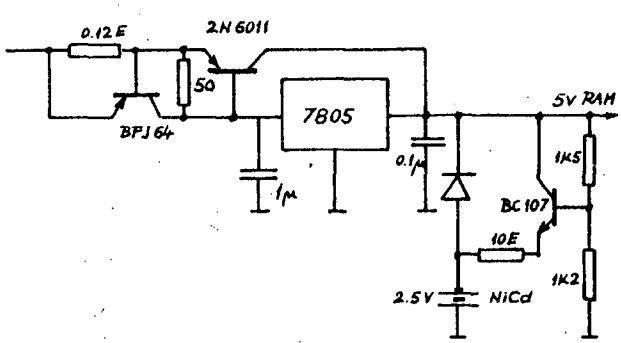
Organizacija pomnilniškega sklada po izvršenem prekinitvenem programu je prikazana na sliki 4b. Pri tem moramo poudariti, da se obravnavani podprogram ne more izvršiti v primeru, če je v usmerniškem delu µR prišlo do kratkega stika, to je do izpada katerekoli napajalne napetosti. V takšnem primeru kontrolno vezje kar najhitreje sklene izhode usmernika in tako prepreči preobremenitve čipov. S tem pa se trenutna vsebina akumulatorja, delovnih registrov in statusnih registrov izgubi, ohrani se le vsebina pomnilnika RAM.

Kontrolno vezje detektira tudi ponovno prisotnost napajalnih napetosti. V takšnem primeru da ukaz za reset mikro procesorja. Različni µR imajo različno organiziran RESET, odnosno RESTART mehanizem. Pri nekaterih µR se RESET ukaz izvaja tako dolgo, dokler je na vhodu RESET prisoten ustrezen logični nivo; pri drugih pa oblikuje RESET stanje prehod iz enega v drug logični nivo. Kakorkoli že, aktiven RESET vhod povzroči, da se v programskem števniku oblikuje adresa posebne lokacije v (E)ROM pomnilniku, kjer se nahaja prva ukazna beseda

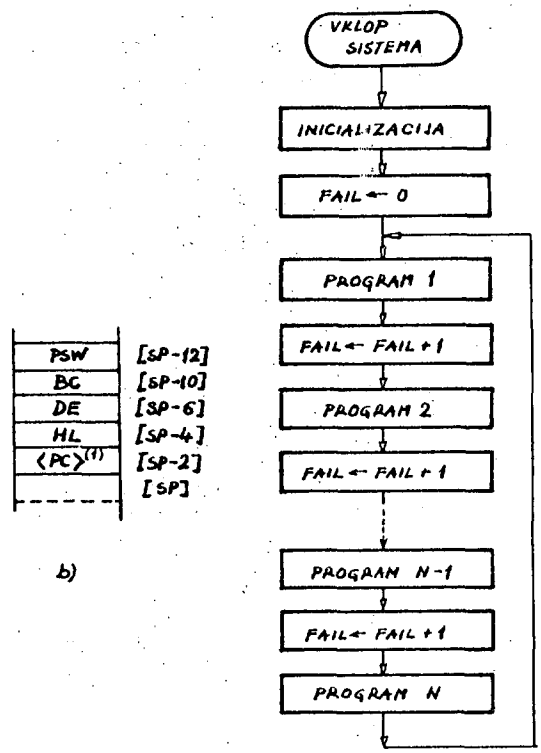
sledečega programa:

```
RESET,  DI          /V času izvajanja tega programa ne
                    dopuščamo servisiranje programskih
                    prekinitev.
        LDA   NSCT
        CPI   377       (2)
        JFZ INICI
        LDHL  STACK /Nastavimo vrednost kazalca vrha
                    sklada
        SP  HL
                    /Od programa, ki vsebuje potrebne
                    programske ukaze za inicializacijo
                    kontrolerja programske prekinitve
                    (INTEL 8259) in za določitev stanj
                    uporabnikov kontrolerja.
        POPX PSW
        POPX BC
        POPX DE
        POPX HL
        EI          /Omogočimo servisiranje prekinitev
                    s strani µR.
        RET         /Vrnemo se v sistemski program.
```

(2) V primeru, da je napaka v napajalnem delu µR kratek stik, je v pomnilniku shranjena napačna vrednost besede NSCT. Ko se pojavi takšen primer, steče program za inicializacijo. Ta pregleduje vrednost besede FAIL, to je besede v pomnilniku, katere vrednost se s številom zase zaključenih programov, ki jih µR izvaja, inkrementira (slika 4). Ob prisotnosti omenjene napake bo vrednost besede FAIL kazala na tisto opravilo, ki je bilo prekinjeno zaradi izpada usmernika. Ob ponovni vspostavitvi vseh napajalnih napetosti inicializacijski program ugotovi, katero opravilo je bilo prekinjeno, in to opravilo se prične nato ponovno izvajati.



Slika 5



Slika 4.

4. "Stand by" za RAM pomnilnik

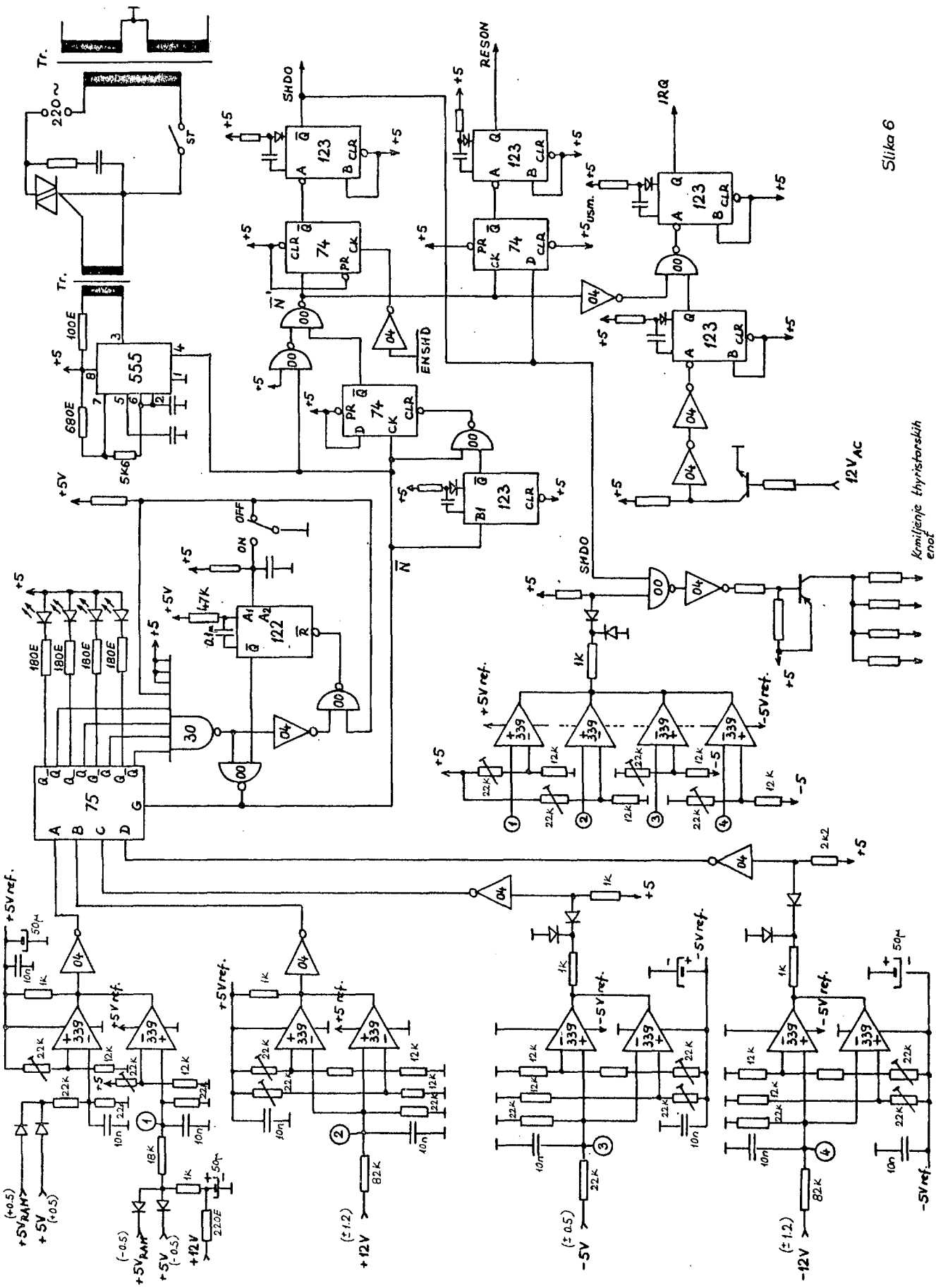
Vezje na sliki 5 omogoča ločeno napajanje RAM pomnilnikov s stabilizirano napetostjo 5V in z možnim avtomatskim preklopom na "stand by" napetost 1,8V.

5. Zaključek

Primer mikroračunalniško nadzorovanega usmernika kaže, kako je možno učinkovito kontrolirati delovanje le-tega. Na sliki 6 prikazana shema kontrolnega vezja je le primer realizacije enega od osnovnih konceptov, ki prispeva k večji zanesljivosti delovanja mikroračunalniško vodenih objektov. Kontrola napajalnika in obravnavana zaščita računalnika namreč odločno prispevata k zanesljivosti delovanja celotnega sistema.

6. Literatura

1. John B. Peatman: Microcomputer- based design Mc Graw-Hill book Company, 1977
2. Tehnična dokumentacija: National Semiconductors, Intel.


Tr.
220~
ST
Tr.
100E
680E
555
5K6
+5
+5V
SHDO
RESON
IRQ
123
74
04
00
ENSHD
+5usm.
12V AC
Komiljenje tiristorskih enot
OFF
ON
4,7K
0,1µ
122
180E
30
75
339
+5V ref.
-5V ref.
22K
12K
1K
2K2
82K
18K
10n
50µ
220E
(+0.5)
(-0.5)
(±1.2)
(±0.5)
-12V
-5V
Slika 6

16 BITNI MIKROPROCESOR MOTOROLA MC 68000

JANEZ URATNIK

ISKRA RAČUNALNIKI, LJUBLJANA

UDK: 681.3-181.4

Članek podaja osnovne lastnosti zgradbe, delovanja in uporabe 16-bitnega procesorja nove generacije Motorola MC 68ooo. Procesor ima vrsto novosti v arhitekturi in v ukazni zalogi.

16 BIT MICROPROCESSOR MOTOROLA MC 68000
This article describes the basic concepts of architecture, operation and use of the new generation 16 bit processor MOTOROLA MC 68ooo. The described processor has some novelties in architecture and in instruction set.

1. U V O D

Generacija 8 bitnih mikroprocesorjev različnih proizvajalcev že nekaj let uspešno izpopolnjuje pričakovanja načrtovalcev in služi kot osnova raznim sistemom na razvojnem, procesnem, hobističnem in poslovnem področju. Vendar pa nam hitri razvoj mikroprocesorske tehnologije ponuja že naslednjo, 16 bitno generacijo, katere tipični predstavniki so Intel 8o86, Zilog 8ooo in Motorola MC 68ooo. Uporabniki mikroračunalnikov težko sledijo novostim, saj je ustrezne, dostopne literature malo in je nepopolna, zato naj pričujoči članek pripomore k boljšemu poznavanju novinca, mikroprocesorja Motorola MC 68ooo. Opis lastnosti procesorja MC 68ooo naj uporabniku - poznavalcu mikroračunalnikov služi kot pomoč pri primerjavi z lastnostmi procesorjev Intel 8o86 in Z 8ooo, opisanih v prejšnjih številkah časopisa INFORMATICA.

2. O S N O V N E L A S T N O S T I

MC 68ooo je prvi Motorolin procesor zgrajen v sodobni VLSI HMOS (high density, short channel MOS) tehnologiji in v novi arhitekturi, ki omogoča vsaj za en razred boljše karakteristike od predhodnikov. Zgradba in osnovni ukazi so prilagojeni prevajalnikom za visoke programske jezike, omogočajo gradnjo multimikroprocesorskih sistemov z nadzorom in dodeljevanjem vodila ter pomnilnika in visoko prepustnost do dva milijona ukazov in prenosov podatkovnih besed na sekundo.

Mikroprocesor MC 68ooo karakterizirajo:

- 32 bitni podatkovni in naslovni registri
- 16 M bytno neposredno naslovno področje
- 56 osnovnih ukazov
- 14 naslovnih načinov
- pet glavnih podatkovnih tipov, ki vsebujejo:
  - bit
  - BCD znak - 4 bite
  - zlog - 8 bitov
  - besedo -16 bitov
  - dolgo besedo - 32 bitov
- sistemski takt 8 MHz
- napajalna napetost V_{DD} = 5V
- poraba moči (pri 8 MHz) P_D = 1,2 W
- programska kompatibilnost s procesorjem MC 68oo preko prevajanja programov
- enostavna priključitev perifernih enot procesorja MC 68oo (kot so MC 6854 ADLC, MC 68488 GPIA, MC 6821 PIA, MC 6843 FDC, idr), ki jo omogočajo signali E, $\overline{VMA}$ in $\overline{VPA}$ procesorja MC 68ooo.

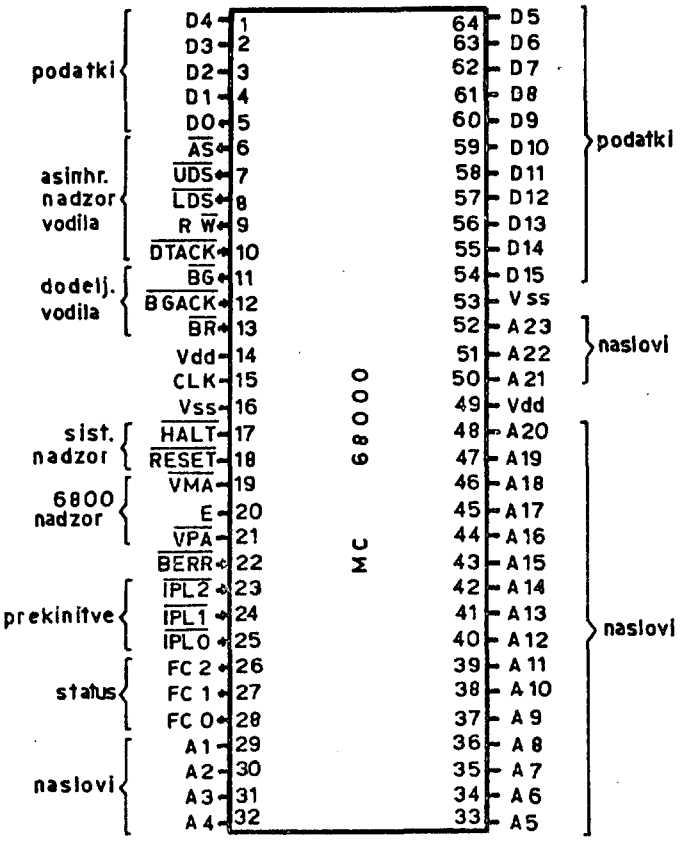
Konsistentnost (ortogonalnost) strukture procesorja omogoča majhno število ukazov. Mnemoniku za vsak tip operacije določi programer velikost podatka, izvirni in končni naslovni način in s tem definira operacijo. Lastnosti zgradbe procesorja omogočajo povratno (Reentrant) modilarno programiranje. Osnova temu je vektorska prednostna predkinitvena struktura, ki ima sedem maskiranih prednostnih nivojev z 192 vektorji in 7 avtovektorji (vsega skupaj je na voljo 255 vektorjev za prekinitve, strojne in programske pasti). Več vgrajenih strojnih pasti odkriva naslednja nenormalna stanja: dostop k besedi z lihim naslovom, napačen ukaz, neznan ukaz, napaka vodila, deljenje z ničlo, ipd. Programske pasti uporablja programer poljubno glede na aplikacijo. Dodatna možnost testiranja je sledilni način (Trace Mode) z izvajanjem ukaza za ukazom.

3. Z G R A D B A P R O C E S O R J A MC 68ooo

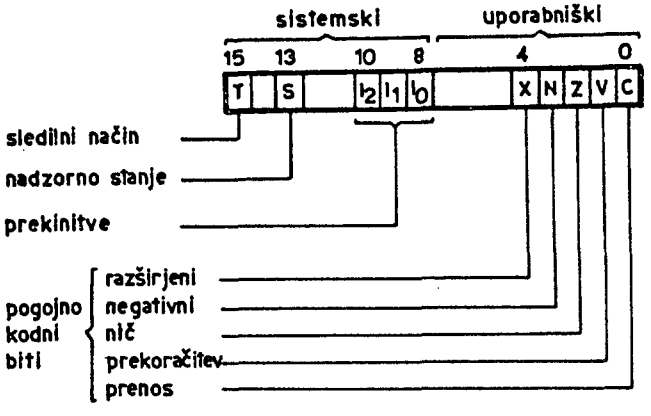
3.1 R e g i s t r i

Procesorko vezje je vsebovano v 64 množičnem ohišju, ki ga prikazuje slika 1.
Uporabniku je dostopnih 17 32- bitnih registrov, 32-bitni programski števec in 16 -bitni statusni register (slika 2).
Prvih osem registrov so podatkovni registri (Do - D7) za zlog (8 bitov), besedo (16 bitov) in dolgo besedo (32 bitov). Naslednjih sedem registrov (Ao - A6) in sistemski skladovni kazalec (A7) so uparabljani kot osnovni naslovni registri, programski skladovni kazalci in za naslovne operacije pri besedah in dolgih besedah. Vseh sedemnajst registrov služi tudi kot indeksni registri.

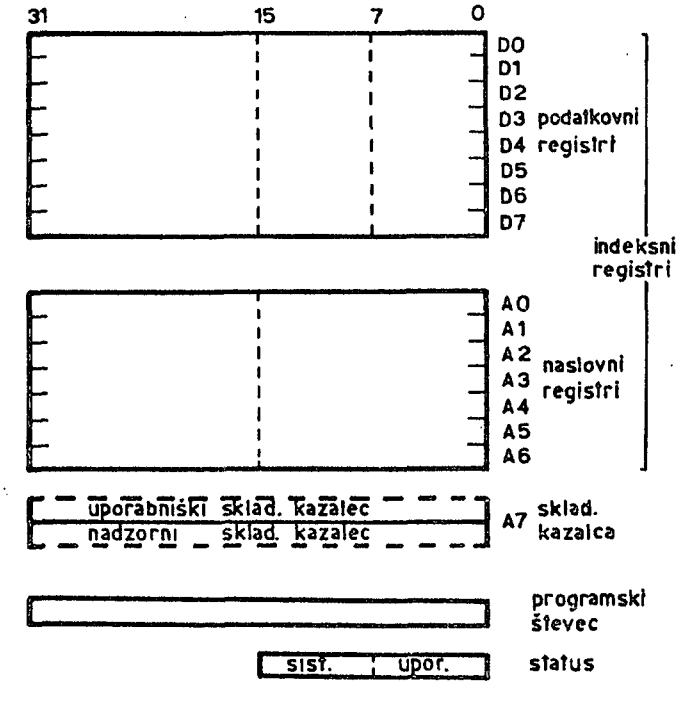
16- bitni statusni register (slika 3) prikazuje prekinitveno masko (3 biti - 8 nivojev), status izvajanja (5 bitov) in status procesorja (2 bita - sledilni in nadzorni način).



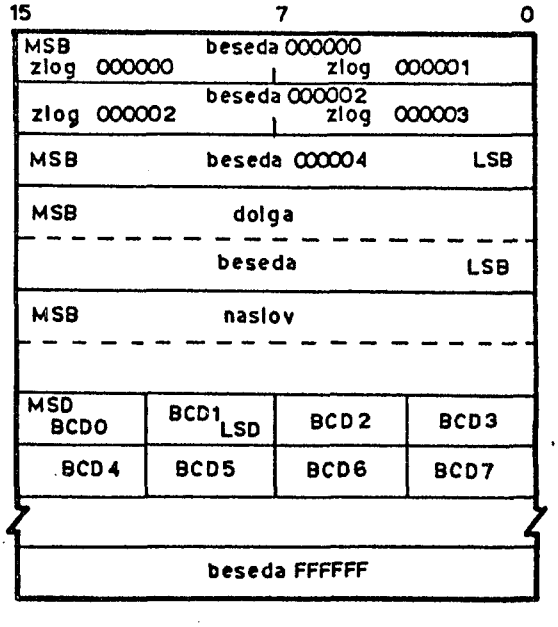
SLIKA 1



SLIKA 3



SLIKA 2



SLIKA 4

Podatkovni registri so 32- bitni in vsebujejo operande dolžine 1, 8, 16 ali 32 bitov v spodnjem delu registra (odbita b o navzgor). Pri operacijah nad podatkovnimi registri se ustrezno spremeni le spodnji del registra, zgornji ni uporabljen.

Naslovni registri podpirajo le 32- bitne naslovne operande in v operacijah nastopajo v celoti.

Organizacija podatkov v pomnilniku upošteva zloge na višjem mestu v besedi s sodim naslovom, ki je enak ustrezni besedi v kateri se zlog nahaja. Ukazi in več zložni podatki so dostopni le na sodih naslovih. Uporabljeni podatkovni tipi so 1 bit, BCD, zlog, beseda in dolga beseda (slika 4).

3. 2 S p o n k e

Procesor MC 68ooo povezuje z zunanjostjo več skupin linij, ki jih prikazuje slika 1.

Naslovno vodilo (Al - A 23) ima 23 enosmernih, tro - stanskih linij, ki lahko naslovijo 16 M zlogov (natančneje 16.777,216 zlogov), kar daje skupaj s krmilnikom pomnilnika dobro osnovo za neposredno izvajanje večjih modularnih programov. V prekinitvenem ciklu prikazujejo linije Al, A2 in A3 nivo prekinitve.

Podatkovno vodilo (Do - D 15) prenaša besede ali zloge. V prekinitvenem potredilnem ciklu vsebujejo linije D o - D 7 številko prekinitvenega vektorja.

Asinhroni nadzor vodila omogoča 5 linij. Naslovni signal $\overline{AS}$ (Address Strobe) določa veljavnost naslova na naslovnem vodilu. R/$\overline{W}$ določa smer podatkov - čitanje/**pisanje** in vpliva na izbiro podatkov skupaj z linijama $\overline{UDS}$ in $\overline{LDS}$. Zgornji in spodnji podatkovni signal, $\overline{UDS}$ in $\overline{LDS}$ (Upper and Lower Data Strobe) nadzira podatke na podatkovnem vodilu in izbira spodnji zlog, zgornji zlog ali oba zloga v besedi. Potrditev prenosa - $\overline{DTACK}$ (Data transfer Acknowledge) označuje konec prenosa podatkov.

Dodeljevanje vodila (Bus Arbitration) različnim enotam na vodilu opravljajo tri linije. Zahteva vodila - $\overline{BR}$ (Bus Request) sporoča procesorju, da želi neka druga enota na vodilu uporabljati vodilo za sebe. Potrditev vodila $\overline{BG}$ (Bus Grant) naznačuje sprostitev vodila ob koncu tekočega procesorjevega cikla. Potrditev prevzema vodila - $\overline{BGACK}$ (Bus Grant Acknowledge) označuje, da je neka druga enota na vodilu prevzela nadzor nad vodilom.

Nadzor prekinitev ($\overline{IPL\ o}$, $\overline{IPL\ 1}$, $\overline{IPL\ 2}$) so linije, ki označujejo zakodirani prednostni nivo enote, ki zahteva prekinitev. Prekinitvenih prednostnih nivojev je 8. Ničti ne predstavlja zahteve, sedmi je najvišji.

Nadzor sistema so tri linije, ki procesor resetirajo, ustavijo ali označijo napako na vodilu. Napuka na vodilu - $\overline{BERR}$ (Bus Error) je vhodni signal, ki proce - sorju označuje napako v izvajanu trenutnega cikla, katere vzrok je lahko: neaktivna enota na vodilu, neprečitan prekinitveni vektor, nepravilna zahteva za dostop k pomnilniku ali druge napake odvisne od programa. Po sprejemu $\overline{BERR}$ signala procesor izbira med prekinitveno rutino in ponovitvijo ciklusa na vodilu. Pri zaznavi dveh hkratnih napak na vodilu se procesor postavi v Halt stanje. Reset - $\overline{RESET}$ linija je dvo - smerna in resetira procesor, če je to vhod oziroma resetira zunanje enote na vodilu, če je to izhod kot posledica ukaza RESET. Popolno resetiranje sistema se doseže s sočasnim zunanjim reset in halt signalom. Ustavitveni - $\overline{HALT}$ signal sproži ustavitev procesorja ob zaključku tekočega ciklusa in postavi tristanjske linije v stanje visoke impedence.

Nadzor nad sinhronimi perifernimi enotami družine 68oo opravljajo tri linije. To so signal za aktiviranje E (Enable), veljavni periferni naslov - $\overline{VPA}$ (Valid Peripheral Address), ki označuje enote družine 68oo in veljaven pomnilniški naslov - $\overline{VMA}$ (Valid Memory Address), ki označuje sinhroniziranost procesorja.

Status procesorja označujejo linije FC o, FC 1 in FC 2. Možna stanja so: uporabniški podatki, uporabniški program, nadzorni podatki, nadzorni program in potrditev prekinitve.

4. STRUKTURA UKAZOV, NAČINI NASLAVLJANJA IN UKAZI

4. 1 S t r u k t u r a u k a z o v

Pred samo razlago načinov naslavljanja je potrebno podati tudi vsebino in format ukazov.

Ukazi vsebujejo podatke o izvajanih funkcijah in o lokacijah operandov. Le te so podane na tri načine:

s podajanjem registra (Register Specification) kjer je številka registra podana v registrskem polju ukaza, z efektivnim naslavljanjem (Effective Address) in implicitno (Implicid Reference) kjer že definicija ukaza določa uporabo registra. Pomnilniške operacije so deljene na programske kjer so izvajani programi in na podatkovne kjer so podatki - operandi.

Ukazi so dolgi od ene do pet besed. Prva beseda - operacijska - definira operacijo in dolžino ukaza. Naslednje besede vsebujejo operande, ki so ali takojšnji (immediate) operandi ali pa odmiki od efektivnega naslova definiranega v operacijski besedi. Operacijska beneda vsebuje efektivni naslov sestavljen iz treh bitov, ki definirajo način in iz treh bitov, ki definirajo register.

4. 2 N a č i n i n a s l a v l j a n j a

Štirinajst fleksibilnih načinov naslavljanja vsebuje šest osnovnih tipov, kot prikazuje slika 5.

Mode	Generation
Register Direct Addressing	
Data Register Direct	EA = Dn
Address Register Direct	EA = An
Absolute Data Addressing	
Absolute Short	EA = (Next Word)
Absolute Long	EA = (Next Two Words)
Program Counter Relative Addressing	
Relative with Offset	EA = (PC) + d_{16}
Relative with Index and Offset	*EA = (PC) + (Xn) + d_8*
Register Indirect Addressing	
Register Indirect	EA = (An)
Postincrement Register Indirect	EA = (An), An ← An + N
Predecrement Register Indirect	An ← An - N, EA = (An)
Register Indirect With Offset	EA = (An) + d_{16}
Indexed Register Indirect With Offset	EA = (An) + (Xn) + d_8
Immediate Data Addressing	
Immediate	DATA = Next Word(s)
Quick Immediate	Inherent Data
Implied Addressing	
Implied Register	EA = SR, USP, SP, PC

NOTES:
EA = Effective Address
An = Address Register
Dn = Data Register
Xn = Address or Data Register used as Index Register
SR = Status Register
PC = Program Counter
d_8 = Eight-bit Offset (displacement)
d_{16} = Sixteen-bit Offset (displacement)
N = 1 for Byte, 2 for Words and 4 for Long Words
( ) = Contents of
← = Replaces

SLIKA 5

Načini naslavljanja so razdeljeni na tri skupnine:

4. 2. 1 Registrsko neposredno naslavljanje, ki vključuje:

- podatkovne registre neposredno (Data Register Direct), kjer je operand v podatkovnem registru podanim z efektivnim naslovom.
- naslovne registre *neposredno* (Address Register Direct), kjer je operand v naslovnem registru podanem z efektivnim naslovom;

4. 2. 2 Pomnilniško naslavljanje, ki vključuje:

- naslovno posredno (Address Register Indirect) kjer je naslov operanda v naslovnem registru podanem v registrskem polju operacijske besede
- naslovno posredno s postinkrementom (Add. Register Indirect With Postincrement), ki je enako prejšnjemu, le da se naslov operanda po uporabi poveča za ena, dva ali štiri glede na dolžino operanda
- naslovno posredno s predekrementom (Add. Register Indirect With Predecrement), ki je enako prejšnjemu, le da se naslov operanda pred uporabo zmanjša za ena, dva ali štiri
- naslovno posredno s premikom (Address Register Indirect With displacement), ki potrebuje eno besedo razširitve - premik, katera se prišteje k vrednosti naslova v naslovnem registru
- naslovno posredno indeksno (Address Register Indirect With Indeks), ki potrebuje eno besedo razširitve, katera se prišteje k vsoti vrednosti naslovnega registra in indeksnega registra. Vsota vseh treh je naslov operanda.

4. 2. 3 Posebno naslavljanje, ki vključuje:

- kratek naslov (Absolute Short Address), kjer je naslov operanda v razširitveni besedi
- dolg naslov (Absolute Long Address), kjer je naslov operanda podan v dveh razširitvenih besedah
- programski števec s premikom (Program Counter With Displacement), kjer je naslov operanda vsota vrednosti v programskem števcu (to je naslov razširitvene besede) in vrednosti razširitvene besede
- programski števec z indeksom (Program Counter With Indeks), kjer je naslov operanda vsota vrednosti v programskem števcu, razširitvene besede in indeksnega registra
- Takojšnje (Immediate Data), kjer je operand v eni ali dveh razširitvenih besedah za operacijsko besedo
- pogojno kodno ali statusno (Conditin codes or Status Register), kjer vrsta ukazov vpliva na pogojno kodni in statusni register

4. 3 U k a z i

Nabor ukazov procesorja MC 68ooo (slika 6) je sestavljen iz množice osnovnih ukazov in podmnožice izpeljank iz teh.

Večina ukazov, ki so bili načrtovani za delo z visokimi strukturiranimi programskimi jeziki, operira nad zlogi, besedami in dolgimi besedami in uporablja omenjene načine naslavljanja. Njihova simetrična, mikrokodirana struktura nudi pleksibilno osnovo za bodoči razvoj.

Ukaze lahko glede na njihov pomen delimo na več skupin:

- ukazi za premik podatkov (Data Movement) so osnova za premikanje (MOVE) podatkov med pomnilnimi celicami, pomnilnikom in registri in za izpeljane, posebne ukaze (MOVEM, MOVEP, EXG, LEA, PEA, LING, UNLK, MOVEQ)
- ukazi z celoštevilčno aritmetiko (Integer Arithmetic Operations) vključuje osnovne operacije (ADD, SUB, MUL, DIV, CMP, CLR in NEG) in izpeljane ukaze za multiprecizijsko aritmetiko (ADDX, SUBX, EXT, NEGX)
- logični ukazi (Logical Operations) vključujejo AND, OR, EOR, NOT za vse velikosti operandov, tudi v takojšnjem (Immediate) načinu
- ukazi za pomike in rotiranja (Shift and Rotate Operations) so aritmetični (ASR, ASL) in logični (LSR, LSL) oziroma z razširitvijo (ROXR, ROXL) ali brez (ROR, ROL)
- ukazi nad biti (Bit Manipulation Operations) so BTST, BSET, BCLR in BCHG
- BCD operacije Omogočajo multiprecizjsko aritmetiko z ukazi ABCD, SBCD in NBCD
- programski nadzor (Program Control Operations) je ustvarjen z vrsto pogojnih in nepogojnih skočnih ukazov (BCC, DBCC, SCC, BRA, BSR, JMP, JSR, RTS, RTR)
- sistemski nadzor (System Control Operations) je ustvarjen s perdnostnimi (Privileged) ukazi, zančnimi (Trap) ukazi in statusnimi ukazi. To so RESET, RTE, STOP, ORI to SR, MOVE USP, ANDI to SR, EORI to SR, MOVE EA to SR, TRAP, TRAPV, CHK.

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
B_{cc}	Branch Conditionally
BCHG	Bit Test and Change
BCLR	Bit Test and Clear
BRA	Branch Always
BSET	Bit Test and Set
BSR	Branch to Subroutine
BTST	Bit Test
CHK	Check Register Against Bounds
CLR	Clear Operand
CMP	Compare
DB_{cc}	Test Cond., Decrement and Branch
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Exclusive Or
EXG	Exchange Registers
EXT	Sign Extend
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link Stack
LSL	Logical Shift Left
LSR	Logical Shift Right

Mnemonic	Description
MOVE	Move
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral Data
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NOP	No Operation
NOT	One's Complement
OR	Logical Or
PEA	Push Effective Address
RESET	Reset External Devices
ROL	Rotate Left without Extend
ROR	Rotate Right without Extend
ROXL	Rotate Left with Extend
ROXR	Rotate Right with Extend
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
S_{cc}	Set Conditional
STOP	Stop
SUB	Subtract
SWAP	Swap Data Register Halves
TAS	Test and Set Operand
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test
UNLK	Unlink

SLIKA 6

5. S I S T E M S K E L A S T N O S T I

5. 1 S k l a d i n v r s t a

Uporabnik lahko podatke organizira v obliki sklada (LIFO register) ali vrste (FIFO register).
Sklad nastopa kot sistemski ali kot uporabniški. Sistemski je določen z naslovnim registrom A 7 in nastopa kot nadzorni (SSP) ali kot uporabniški (USP) skladovni kazalec glede na stanje S bita v statusnem registru. Uporabniški sklad je določen z naslovnimi registri Ao - A6 in se polni od zgoraj navzdol ali obratno. Kreiramo ga za naslovno posrednim naslovnim načinom s post ali pre dekrementom.

Vrste so določene z uporabo dveh od sedmih naslovnih registrov (Ao - A6) hkrati in se polnijo od zgoraj ali od spodaj. Dva registra sta potrebna za definiranje začetnega in končnega naslova vrste. Vrsto lahko uporabimo kot krožni pomnilni vmesnik - *Buffer*.

5. 2 P r e n o s p o d a t k o v p o v o d i l u

Prenos podatkov po vodilu je asinhron. Poleg že poznanega in čitalnega cikla nastopa tu še novost: čitalno - modificirani - pisalni cikel (Read - Modify-Write cycle). V tem ciklu se podatek prečita iz naslova, modificira v aritmetično logični enoti in vpiše nazaj na isti naslov. Tak način prenosa skupaj z ukazom TST (Test and Set) je pomemben za komunikacijo med procesorji v multiprocesorskem okolju.

5. 3 S t a n j a p r o c e s o r j a

Procesor MC 68ooo je vedno v enem od treh stanj: normalnem, izjemnem ali ustavljenem.

V normalnem stanju procesor čita ukaze in operande in jih izvaja.

Izjamno stanje je pogojeno s prekinitvami, zankami in izjemnimi pogoji. Nastopi lahko interno zaradi ukaza ali izjemnega pogoja in eksterno zaradi prekinitve, napake vodila ali RESETA.

V ustavljenem stanju je procesor le po katastrofalnih napakah. Odpravi ga le zunanji reset.

V normalnem stanju lahko procesor deljuje na dva načina: nadzorno (supervizor state) ali uporabniško (User state). Programi v enem načinu nimajo vpliva na programe v drugem načinu, razen operacijski, ki je v nadzornem načinu in ima dostop do vseh področij. To povečuje varnost in zanesljivost sistema. Nadzorni način je določen z S bitom statusnega registra in ima višjo prioriteto kot uporabniški način. V nadzornem načinu se odvija izjemno procesiranje in le to lahko spremeni način iz uporabniškega v nadzornega. Način v katerem se procesor nahaja sporoča navzven linije FCo, FC1 in FC2. Možna stanja so: uporabniški podatki *uporabniški program*, *nadzorni podatki*, *nadzorni program* in sprejem prekinitve. Če so pomnilniška področja za prve štiri načine zunaj ločena, lahko procesor naslavlja do 64 M zlogov pomnilnika.

5. 4 I z j e m n o p r o c e s i r a n j e

Do izjemnega procesiranja pride pri izjemnih stanjih v uporabniškem načinu. Procesiranje prične s kopiranjem statusnega registra in postavljanjem v stanje izjemnega procesiranja. V drugem koraku je določen izjemni vektor, v tretjem je shranjeno trenutno procesorjevo stanje. Četrti korak prinese novo stanje in procesor prične z izvajanjem izjemnega procesiranja. Izjeme so določene z vektorji podanimi od zunaj ali *znotraj procesorja glede na vzrok izjeme. To so* lahko prekinitve, napaka vodila, reset, posamezni ukazi ali napake naslova. Izjeme so deljene v tri skupine. Najvišjo prednost ima reset. Odvijanje vsake od izjem ima poseben potek.

6. S K L E P

Članek o procesorju MC 68ooo je informativne narave, saj bo potrebno za gradnjo in programiranje sistema s tem procesorjem še veliko študija in dela. Vseeno pa podaja osnove lastnosti in delovanja in se ne spušča v tehnične podrobnosti.

7. L I T E R A T U R A

1. 16 bit Microprocessor User's Manual, Motorola sept. 1979

2. MC 68ooo, 16 bit Microprocessor, Motorola 1979

TRETJA SMER RAČUNALNIŠKEGA IZOBRAŽEVANJA

SAŠA DEKLEVA

VISOKA ŠOLA ZA ORGANIZACIJO DELA, KRANJ

UDK: 681.3:378.1

Informatika je področje, kjer je praksa prehitevala izobraževalni proces na univerzah po svetu. S še posebno zamudo spoznavamo, da obstaja poleg na univerzah najprej uveljavljenih izobraževalnih programov za področji razvoja računalnikov in tzv. računalniške znanosti še tretje, ločeno in pomembno področje, ki ga pri nas največkrat imenujemo poslovna informatika. V tem prispevku pojasnjujemo v čem se poslovna informatika razlikuje od ostalih dveh področij in kakšna znanja potrebujemo za uspešno delo na tem "neznanstvenem" in za to na univerzah zapostavljenem področju dela računalnikarjev.

THIRD COMPUTER-ORIENTED EDUCATION FIELD.

It is typical that computer-oriented education on the universities all over the world goes behind the developments in practice. Universities are especially late with recognition of the third education field, called Information Systems or Business Data Processing, which is important and separate from Computer Engineering and Computer Science programs. The author explains the differences between Information Systems and other two programs as well as knowledge, needed for success in this "nonscientific", and on universities therefore neglected working area of computer specialists.

1. Uvod

Pred kakimi 20 leti se je pojavila na področju glasbe nova, tretja smer. Tako so imenovali glasbeno ustvarjalnost, pri kateri so združevali zakonitosti klasične glasbe z značilnostmi jazza. Izvajalci so bili največkrat sinfonični orkester, ki se mu je pridružil jazzovski ansambel. Toda ti poskusi se niso trajno uveljavili, saj taka glasba ni bila res nova.

Na drugačnih temeljih stoji uveljavljanje tretje smeri izobraževanja računalnikarjev, saj gre za podajanje novih znanj. Frustrirani s prepogostimi neuspehi pri poskusih uvajanja na videz kar se da koristnih računalniških aplikacij v najrazličnejših sredinah, so se strokovnjaki in znanstveniki začeli za razloge zanimati in jih raziskovati. Spoznali so, da razvijalci teh računalniških aplikacij niso primerno izobraženi in da veščina programiranja sploh ni dovolj za uspešno razvijanje in uvajanje novih sistemov. Ena od ugotovljenih očitnih zablod teh razvijalcev je bila ta, da so se osredotočali skoraj docela na problematiko, vezano na računalniško-tehnična vprašanja, jedro problema pa je bil pravzaprav človek.

Razvijalci in izvajalci izobraževanja za področje poslovnih informatikov so se koncem sedemdesetih let na različnih koncih sveta odločili, da ne smejo ostati več "tiha večina". Pisali so o tem in organizirali posvetovanja, vendar do danes še nimamo vzorca priporočljivega učnega programa. V roku enega leta pa bo tak vzorec vsaj eno od vplivnih mednarodnih združenj, kot so IFIP, ACM in IEEE, poslalo v svet. Ta združenja so opravila veliko dobrega dela pri opredelitvah potreb po univerzitetnih učnih programih za področje računalništva, vendar večji del njihovih priporočil ni primeren za področje poslovne informatike in učiteljem s tega področja ni v pomoč.

2. Razlike med visokošolskimi učnimi programi

Razlike med programi za izobraževanje računalnikarjev se kažejo na več načinov:
- različnost nazivov programov,
- različna področja zaposlovanja diplomantov,
- različno usmerjeni učni programi,
- različne izkušnje učiteljev,
- različni interesi študentov.

Analizirajmo vsako od navedenih razlik nekoliko podrobneje!

2.1. Zmeda pri nazivih usmeritev oz. programov.

Tudi pri nas imenujemo usmeritve študija s področja računalništva nadvse pestro, ne le v svetu. v ZDA najdemo, npr., take različice:
- computer science (računalniška znanost),
- information science (informacijska znanost),
- business data processing (poslovna obravnava podatkov),
- information systems (informacijski sistemi),

- computer engineering (računalniški inženiring),
- data processing technology (tehnologija obravnave podatkov),
- management information systems (upravljalni informacijski sistemi),
- business systems analysis (analiza poslovnih sistemov),
- accounting information (računovodska informatika) ipd.

Pri nas govorimo o:

- poslovni informatiki,
- organizacijsko-računalniški usmeritvi,
- študiju statistike in avtomatske obravnave podatkov,
- poslovnem računalništvu itd.

Seveda je od naziva študijske usmeritve pomembnejša njena vsebina, vendar je v obdobju velikih sprememb v izobraževalnem sistemu lepa prilika, da se dogovorimo tudi o tej podrobnosti. Bojimo se namreč, da tudi zmeda v nazivih vpliva na nesporazume, ki jih študenti odkrijejo šele med študijem.

2.2. Različna področja zaposlovanja diplomantov

Če prezremo zadnjih nekaj let, smo v Jugoslaviji potrebovali strokovnjake, ki so do podrobnosti poznali izgradnjo in delovanje računalnikov,predvsem za njihovo vzdrževanje. Te so tuji proizvajalci usposobili odkrivati napake v njihovem delovanju in jih odpravljati. Izobraževali so se v ustreznih tečajih, največkrat samo za posamezne modele in enote, pri čemer so jim bili v pomoč diagnostični programi in številni priročniki in jim skoraj ni bilo treba poznati arhitekture računalnikov. To velja tako za vzdrževanje aparaturne kot sistemske programske opreme.

S prvimi poskusi lastnega razvoja računalnikov - temu preburnemu dogajanju smo priče zadnjih nekaj let - se je potreba po strokovnjakih, sposobnih razvijanja računalnikov, šele pravzaprav resneje pokazala. Vemo, da je to predvsem tehnično, inženirsko delo, ki seveda zahteva tovrstno izobrazbo. Tako smo odkrili prvo področje zaposlovanja računalniških strokovnjakov. Njihovo znanje prav gotovo temelji na elektroniki in fini mehaniki ter recimo programskem inženiringu.

Pri proizvajalcih računalnikov, v velikih računalniških centrih in v znanstvenih inštitucijah najdemo drugo tipično področje zaposlovanja računalniških strokovnjakov, ki obvladajo tzv. računalniško znanost. Za razliko od prejšnjih inženirjev so to znanstveniki, ki iščejo načine optimizacije rabe računalnikov. Največkrat so razvijalci programske opreme, kot so prevajalniki, operacijski sistemi, pomožni programi itd. Ti strokovnjaki poznajo naslednja področja znanj:

- teorija numerične analize,
- uporabna matematika in numerična analiza,
- matematično programiranje,
- Monte Carlo metode,
- verjetnost in statistika,
- logične osnove računalniške teorije,
- naravni jeziki in računalništvo,
- deskriptivna lingvistika,
- umetna inteligenca in teorija psihologije,
- dedukcija in reševanje problemov z računalniki,
- razpoznavanje vzorcev, adaptivni sistemi in učenje,
- teorija grafov,
- analiza algoritmov,
- teorija avtomatov,
- operacijski sistemi,
- programski jeziki,
- baze podatkov itd. itd.

Njihovo najmočnejše orožje je matematika in obvladovanje znanstvenega načina dela.

Za razliko od omenjenih dveh področij najde strokovnjak s področja poslovne informatike zaposlitev pri uporabnikih in ne proizvajalcih računalnikov. Odgovoren je za snovanje in razvoj uporabniku namenjenih programov. Za to delo primerne strokovnjake imenujemo poslovni informatiki, zaposlujejo pa se za naloge organizatorjev računalniško podprtih informacijskih sistemov in programerjev teh sistemov. Zanimajo se za podpiranje upravljanja s pomočjo računalnikov. Poleg splošnega znanja o uporabnosti računalnikov pri upravljanju se opirajo na znanja o poslovnih procesih, analizi in snovanju informacijskih sistemov ter njihovem razvoju, uvajanju itd.

2.3. Različno usmerjeni učni programi.

Iz prejšnjega razmišljanja o področjih zaposlovanja računalniških strokovnjakov se nam že bistri silhueta treh različnih in potrebnih študijskih usmeritev. Te tri se morajo

razlikovati v:
- različnih povdarkih v učnih programih,
- različnih izkušnjah učiteljev in
- različnih pričakovanjih študentov.

Zavajanje študentov na zanje neprivlačno področje dela ni potrebno komentirati. Važno je povdariti, da je za učitelje na usmeritvi poslovne informatike izkušenost pri delu z računalnikom mnogo premalo. Praktične izkušnje in poznavanje poslovnih sistemov so zanje nenadomestljivo znanje (karikirano rečeno: poznavanje VM ali DELTAM operacijskega sistema pri pogovoru s poslovodnim organom ali skladiščnikom nič ne koristi).

Učni programi se delno prekrivajo, vendar bi povdarke v treh različnih poenostavljeno lahko takole opredelili:

1/3 predmetov s področja splošnega izobraževanja, trajanje študija pri vseh treh pa štiri leta.

- aparaturna oprema } računalniški inženiring
- operacijski sistemi
- programiranje } računalniška znanost
- analiza in snovanje sistemov } poslovna informatika
- teorija organizacije

Študenti pa bi se morali odločati med računalniško znanostjo in poslovno informatiko glede njihove nagnjenosti k proučevanju takole:

računalniška znanost		poslovna informatika
gotovost	←——→	negotovost
kvantitativno	←——→	kvalitativno
objektivno	←——→	subjektivno
usmerjeno k predm.	←——→	usmerjeno k ljudem
razvoj dokazov	←——→	posredovanje idej
znanstvene metode	←——→	sistemski pristop
znanstveni pristop in inženirstvo	←——→	poznavanje organizacij

Področji in načina dela sta torej močno različna, verjetno pa so različne tudi želje študentov. Naloga družbenih in visokošolskih delavcev pa je študentom jasno predstaviti različna področja dela in zanje primerne učne programe še pred njihovim prvim vpisom na univerzo.

3. Zaključek.

Kar sami po sebi so se nam razkrili trije različni učni programi (za katere bomo morda našli lepša imena):
- računalniški inženiring
- računalniška znanost in
- poslovna informatika.

Za prva dva imamo že relativno dolgo priporočena vzorca njunih vsebin, na vzorec za tretjega, ki je definitivno nekaj čisto drugega od prvih dveh, pa nam ne bo treba več dolgo čakati. Zahteva zanj je namreč z vseh strokovnih sredin po svetu vedno glasnejša. Tak vzorec nam ne bi koristil toliko pri opredeljevanju vsebine naših učnih programov kot pri verifikaciji upravičenosti borbe za spoznanje tretje smeri izobraževanja računalnikarjev pri vseh, ki vplivajo na organiziranje izobraževalnega procesa. Potrebo po tretji smeri, ki je nekaj novega, že dolgo poznajo v krogih uporabnikov poslovnih računalnikov.

Kot veliko spodbudo za uveljavljanje posebnega učnega programa za poslovno informatiko razumemo tudi ploden razvoj metodologij za prav vse faze razvoja računalniško podprtih informacijskih sistemov, nastalih v sedemdesetih letih. S tem mislimo predvsem na metodologije s področij sistemske analize, snovanja sistemov, opredeljevanja podatkovnih struktur, programiranja itd. Delo na področju poslovne informatike tako ni več niti artizem niti diletantstvo. Za razliko od tretje smeri v glasbi, tu ni prostora za improvizacije.

POSVET O RAČUNALNIŠTVU V USMERJENEM IZOBRAŽEVANJU

Ob 4. republiškem tekmovanju srednješolcev iz računalništva, ki je bilo 19. aprila 1980 na Fakulteti za elektrotehniko v Ljubljani je bil organiziran tudi posvet o računalništvu v reformirani srednji šoli. Sodelovali so prisotni učitelji računalništva (spremljevalci tekmovalcev), predstavniki Zavoda SRS za šolstvo, računalniški strokovnjaki (fakultetni učitelji in drugi), predstavniki delovnih organizacij (proizvajalci in uporabniki računalnikov), predstavniki organizacij za računalniške raziskave in razvoj in predstavniki občil.

Osnovna ugotovitev je bila, da preobrazba vzgoje in izobraževanja daje pomemben impulz srednješolskemu računalništvu. Pri tem moramo ločiti računalništvo kot splošno izobraževalni predmet v različnih usmeritvah in računalniško usmeritev, ki bo dajala dva profila kadrov: (1) računalniški tehnik in (2) programerski tehnik.

Splošno računalništvo se naslanja na projekt "Pouk računalništva v usmerjenem izobraževanju" pri Računalniškem centru za programirano učenje. Druga veja, specializirana računalniška usmeritev, se naslanja na rezulatate delovne skupine za pripravo programov računalniških profilov s srednješolsko izobrazbo pri izobraževalni skupnosti za elektro stroko. V okviru naravoslovno-matematične usmeritve pa se bo izobraževal tudi matematik tehnik, ki bo imel izdatnejši pouk računalništva.

Živahna debata se je razvila okoli posebne izobraževalne skupnosti za računalništvo. Začasno spada računalništvo v izobraževalno skupnost za elektro stroko. Nekateri proizvajalci računalniške opreme (Iskra in Gorenje) menijo, da naj ostane tako tudi v prihodnje. Spet drugi (Elektrotehna) podpirajo posebno izobraževalno skupnost za računalništvo, prav takega mnenja so bili navzoči predstavniki delovnih organizacij in uporabnikov iz računskih centrov.

Slovensko društvo Informatika podpira posebno izobraževalno skupnost za računalništvo, ker se je računalništvo v več kot tridesetih letih razvoja osamosvojilo kot posebna panoga industrije, znanosti in izobraževanja. Odnos med računalništvom in elektrotehniko (ali katerimkoli drugim področjem, ki je mnogo prispevalo k razvoju računalništva - npr. matematiko) bi lahko primerjali npr. z odnosom med medicino in kemijo. Menimo, da zavzemanje za računanlništvo v elektro stroki vleče kolo razvoja nazaj. Položaj računalništva kot stroke nujno vodi do posebne obravnave te stroke tudi v izobraževanju. V novi posebni izobraževalni skupnosti za računalništvo bi morali združiti računalniško izobraževanje na vseh ravneh in ga trdno povezati z združenim delom in raziskovalno dejavnostjo na tem področju.

Ker je pomanjkanje računalniških kadrov kritično, so se nekateri proizvajalci zatekli k neposrednemu dogovarjanju (in investicijam v opremo) s posameznimi srednjimi šolami (npr. Elektrotehna s I. Gimnazijo Ljubljana Bežigrad in Tehniško elektro šolo v Ljubljani). Pozitivna prizadevanja pri povezovanju združenega dela z izobraževalnimi organizacijami sodijo v okvir samoupravnega dogovarjanja, s katerim se ureja naše srednje šolstvo. Ta pojav kaže tudi na dejstvo, da je treba mrežo šol načrtovati bolj življenjsko, upoštevati je treba tudi iniciativo, tradicijo in prizadevanja posameznih šol.

Poudarjena je bila nujnost sistematičnega izobraževanja učiteljev računalništva. Dosedanje začasno izobraževanje poteka v okviru tečajev in seminarjev pod okriljem Zavoda SRS za šolstvo. VTO matematika in mehanika Fakultete za naravoslovje in tehnologijo izobražuje učitelje matematike, ki so po sedanjem študijskem programu dokaj usposobljeni za pouk računalništva kot splošno izobraževalnega predmeta, v teku pa so tudi prizadevanja za izobraževanje učiteljev računalništva. Katedra za računalništvo in informatiko Fakultete za elektrotehniko tudi snuje pedagoški modul. Ta pedagoški modul je treba skrbno načrtovati in ob upoštevanju dosedanjih izkušenj čimprej oživiti.

Izpostavljen je bil tudi problem "vertikalne kontinuitete" vzgojno izobraževalnih programov. Številne fakultete imajo v svojih študijskih programih računalniške predmete. Posebej je mogoče študirati računalništvo in informatiko na Fakulteti za elektrotehniko v Ljubljani (tretji in četrti letnik in podiplomski študij) in na Fakulteti za naravoslovje in tehnologijo, VTO matematika in mehanika (smer Uporabna matematika in prvostopenjski študij ob delu). Na Visoki šoli za organizacijo dela v Kranju poteka štiriletni študij poslovnega računalništva in informatike, na Visoki ekonomsko komercialni šoli v Mariboru pa lahko študenti pri vpisu v drugi letnik izberejo smer poslovna informatika. Z uvedbo računalniške usmeritve na srednji stopnji se postavlja zahteva za ustreznejšo povezavo tudi s višjim in visokim izobraževanjem tako v smislu globalnega načrtovanja smeri, kot tudi konkretnih študijskih programov (predvsem v 1. letniku). Abiturienti računalniške usmeritve naj imajo širšo možnost nadaljnega izobraževanja. Prav tako pa naj bo vpis na računalniške smeri na naših univerzah bolj odprt.

Učitelji so poudarili problem "centralizacije" šolanja, ki je povezan z odhodom zdoma. Tega problema ne rešujejo le dijaški domovi. Posebej je bil poudarjen pomen celovitega vzgojno-izobraževalnega procesa v tem obdobju človekovega razvoja.

Izpostavljen je bil tudi problem opreme za izvajanje pouka računalništva na neračunalniških usmeritvah. Potrebno je izdelati minimalne standarde za opremo. Osnova naj bo eden ali več (oženih) terminalnih priključkov (preko telefonske linije) na večje računalnike. Načeloma naj bi bil to univerzitetni računalnik. S tem bi oba Računalniška centra Univerz pokrivala tudi področje srednjih šol.

Poudarjena je bila potreba po širši in boljši informiranosti o dogajanjih v okviru računalništva v usmerjenem izobraževanju, kakor tudi o povsem konkretnih delovnih rezultatih omenjenega projekta in delovne skupine.

Zapisala: V. Rajkovič in A. Cokan

ČETRTO REPUBLIŠKO TEKMOVANJE SREDNJEŠOLCEV S PODROČJA RAČUNALNIŠTVA

R. REINHARDT
M. MARTINEC

UDK: 681.3:371.27 (497.12)

SLOVENSKO DRUŠTVO INFORMATIKA, LJUBLJANA

Povzetek. Prispevek podaja poročilo o četrtem republiškem tekmovanju srednješolcev s področja računalništva, ki ga je organiziralo Slovensko društvo Informatika v aprilu 1980. V prispevku so vse naloge z rešitvami in pregled rezultatov.

FOURTH COMPUTER SCIENCE CONTEST FOR HIGH-SCHOOL STUDENTS. The article gives a report on The Fourth Computer Science Contest. It includes the complete set of problems with their solutions and a short overwiev of contest results.

I. Uvod

Ena od rednih dejavnosti Slovenskega društva Informatika je tudi popularizacija računalništva in informatike med srednješolsko mladino. Komisija za popularizacijo računalništva je zato organizirala že četrto republiško tekmovanje srednješolcev s področja računalništva.

Tekmovanje je bilo 19. aprila 1980 na Fakulteti za elektrotehniko v Ljubljani, udeležilo pa se ga je 59 tekmovalcev po enem letu pouka računalništva in 29 tekmovalcev po dveh letih pouka računalništva.

Pri organizaciji letošnjega tekmovanja so poleg Društva in Fakultete za elektrotehniko sodelovali še Inštitut Jožef Stefan ter VTO matematika in mehanika, pokrovitelj tekmovanja pa je bila Delta - Elektrotehna.

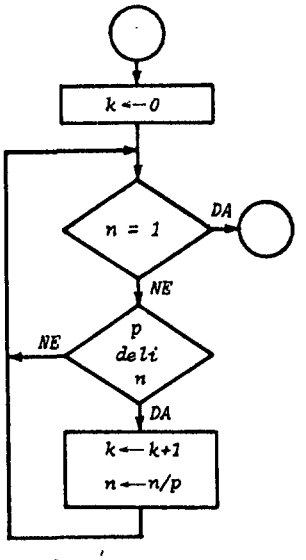
Tekmovanje je otvoril predsednik Slovenskega društva Informatika dr. A. P. Železnikar, tekmovalce pa so pozdravili še dekan Fakultete za elektrotehniko, predstavnik pokrovitelja in predsednik tekmovalne komisije.

Primarni cilj tekmovanja je popularizacija računalništva, obenem pa se učenci srednjih šol seznanijo s možnostmi študija na področju računalništva. Ker večino tekmovalcev spremljajo učitelji računalništva, je tekmovanje tudi priložnost za izmenjavo izkušenj in mnenj. Zato je potekal vzporedno s tekmovanjem tudi pogovor o pouku računalništva v usmerjenem izobraževanju. Na ta pogovor smo povabili tudi predstavnike višjih in visokih šol, Izobraževalne skupnosti SRS, Zavoda SRS za šolstvo, proizvajalce računalnikov in uporabnike računalnikov. Zapis pogovora objavljamo v tej številki našega časopisa.

Po tekmovanju so si udeleženci organizirano ogledali bližnje računalniške centre, predstavniki višjih in visokih šol iz obeh slovenskih Univerz pa so jih podrobneje seznanili s študijem in učnimi načrti svojih organizacij. Sledila je razglasitev rezultatov, na kateri so prvouvrščeni tekmovalci prejeli plakete in knjižne nagrade.

II. Naloge za učence po prvem letu pouka računalništva

1. n in p sta celoštevilčna podatka. Za algoritem:



ugotovi:

a. kakšna je vrednost k ob koncu izvajanja algoritma, če je bilo v začetku n = 81, p = 3;
b. za kakšne vrednosti n in p se algoritem konča;
c. kakšna je vrednost spremenljivke k, kadar se izvajanje algoritma konča?

2. Ob železniški progi imamo postavljeno merilno napravo, ki ugotovi, če se mimo nje pelje vagon. Vagoni so dolgi 10m, med njimi je 1m prostora. Dva vlaka si ne sledita v manj kot eni minuti. Napiši postopek (ali program), ki vsakokrat, ko mimo odbrzi vlak, izpiše njegovo hitrost. Podani sta funkciji:

VAGON *ima vrednost DA (1, true), če je ob merilni napravi vagon, sicer pa NE (0, false);*
ČAS *njena vrednost pove čas v sekundah, ki je minil od nekega fiksnega trenutka dalje.*

3. *Napiši program, ki med števili od 1 do N poišče število, ki ima največ deliteljev. N je podatek.*

4. *V nekem programskem jeziku imamo definirane logične operatorje: negacija (NE, NOT), konjunkcija (IN, AND) in disjunkcija (ALI, OR) ter relacijski operator ⩾ (večji ali enak). Zapiši izraze: $a < b$, $a > b$, $a = b$, $a \neq b$ in $a \leqslant b$ samo s definiranimi operatorji.*

5. *Po izteku spodnjega dela programa, kjer sta n in s celoštevilčni spremenljivki, je postal s = 10. Ugotovi, najmanjšo možno začetno vrednost n, ki da tak rezultat.*

```
. . .
s := 0;
while n>0 do
  begin
    if odd(n) then s := s + 1;
    n := n div 2
  end;
. . .
```

```
      . . .
      S = 0
1     IF(N.LE.0) GO TO 2
        IF(IFIX(N/2.0) .NE. N/2.0) S = S + 1
        N = N/2
      GO TO 1
2     CONTINUE
      . . .
```

III. Naloge za učence po drugem letu pouka računalništva

1. *Vzemimo naslednji program:*

```
program VERIGA(input);
  var k,i: integer;
      T: array [1..100] of 0..100;
  begin
    for i := 1 to 100 do read(T[i]);
    read(k);
    while T[k]<>0 do k := T[k]
  end.
```

```
C     PROGRAM VERIGA
        INTEGER K
        INTEGER T(100)
C
        READ(2,1) T
1       FORMAT(20I3)
        READ(2,2) K
2       FORMAT(I3)
3       IF(T(K) .EQ. 0) GO TO 4
          K = T(K)
        GO TO 3
4       CONTINUE
      STOP
      END
```

Napiši program, ki prebere podatke zgornjega programa VERIGA (vrednosti elementov tabele T, ki so med 0 in 100 in začetni indeks). Program naj izpiše, ali se program VERIGA za prebrane podatke konča ali ne.

2. *Ob železniški progi imamo postavljeno merilno napravo, ki ugotovi, če se mimo nje pelje vagon. Vagoni so dolgi 10m, med njimi je 1m prostora. Dva vlaka si ne sledita v manj kot eni minuti. Napiši postopek (ali program), ki za vsak vlak izpiše število vagonov in čas, ko je odpeljal mimo zadnji vagon. Podani sta funkciji:*

VAGON *ima vrednost DA (1, true), če je ob merilni napravi vagon, sicer pa NE (0, false):*
ČAS *njena vrednost pove čas v sekundah, ki je minil od polnoči.*

3. *Napiši program, ki izpiše oznake postaj po vrsti, kot jih obišče avtobus na neki progi, če imamo podane pare (postaja, naslednja postaja). Na celi progi je n postaj (manj kot 1980), oznake postaj pa so števila od 1 do n. Podatki opisujejo natanko eno krožno progo in jih ni treba testirati.*

4. *Na mizi imamo n kozarcev z različnimi volumni. V vsakem kozarcu je lahko neka količina vode. Vodo smemo pretakati med kozarci na dva načina:*

- *prelijemo vso vodo iz enega kozarca v drugega, če je v drugem dovolj prostora,*
- *prelijemo v drug kozarec toliko vode, da je le-ta poln.*

Napiši, kako bi problem (podatke) predstavil v programskem jeziku, in napiši podprogram, ki opravi prelivanje med dvema kozarcema. Kozarci so oštevilčeni od 1 do n.

5. *Ista naloga kot 5. naloga za tekmovalce po prvem letu pouka.*

IV. Rezultati prvouvrščenih tekmovalcev v vsaki skupini

PO PRVEM LETU POUKA RAČUNALNIŠTVA

mesto	*št točk*	*tekmovalec, šola*
1	*93*	*Marjan SELJAK, I. gimnazija Ljubljana Bežigrad*
2	*91*	*Ivan PEPELNJAK, I. gimnazija Ljubljana Bežigrad*
3	*88*	*Dušan PONIKVAR, Elektrotehniška srednja šola Ljubljana*
4	*83*	*Nevenka PUSTAVRH, Gimnazija B. Ziherl Škofja Loka*
5	*82*	*Igor ŠTEMBERGER, Elektrotehniška srednja šola Ljubljana*
6	*80*	*Branko DERNAČ, Šolski center - gimnazija Brežice*
7	*78*	*Marjan MILIČ, Gimnazija Koper*
7	*78*	*Damjan ŠTRUCL, Gimnazija Ljubljana Šentvid*
9	*74*	*Niko MOLE, I. gimnazija Ljubljana Bežigrad*
9	*74*	*Matjaž KALUŽA, Gimnazija M. Zidanška Maribor*

PO DRUGEM LETU POUKA RAČUNALNIŠTVA

mesto	*št točk*	*tekmovalec, šola*
1	*97*	*Bojan CESTNIK, I. gimnazija Ljubljana Bežigrad*
2	*93*	*Uroš KUNAVER, I. gimnazija Ljubljana Bežigrad*
3	*87*	*Andrej BRODNIK, I. gimnazija Ljubljana Bežigrad*

4	85	*Anton VERBOVŠEK, I. gimnazija Ljubljana Bežigrad*
4	85	*Leon MATOH, Gimnazija Novo mesto*
4	85	*Marjan HORVATIČ, Gimnazija Novo mesto*
7	75	*Gorazd PLANINŠIČ, I. gimnazija Ljubljana Bežigrad*
8	72	*Jana PADEŽNIK, Gimnazija M. Zidanška Maribor*
9	70	*Tomi DOLENC, I. gimnazija Ljubljana Bežigrad*

V. Rešitve nalog za učence po prvem letu pouka računalništva

1. a) Z enostavnim sledenjem algoritma ugotovimo, da dobi k vrednost 4.
b) Algoritem se konča, če je n oblike p^k (p na k), kajti le tedaj dobimo po nekaj zaporednih deljenjih n s p kvocient 1.
c) Očitno je $k = \log_p n = \ln(n)/\ln(p)$.

2. Postopek zapišimo v pascalu

```
program vlak1(output);
 var
  t,dt: real;
  v: real;
 function cas: real; external;
 function vagon: boolean; external;
 begin {vlak1}
  repeat
   repeat until vagon;
   t:= cas;
   repeat until not vagon;
   v:= 10/(cas-t);
   repeat
    if vagon then
     begin
      repeat until not vagon;
      t:= cas
     end
   until cas-t > 60;
   writeln('Vlak je peljal s hitrostjo',
     v:3:2, ' m/s')
  until false
 end {vlak1}.
```

3. Zapišimo najenostavnejšo rešitev (ki seveda ne dela prav učinkovito). Z bolj učinkovitimi rešitvami je precej več dela.

```
C     program delitelji
      INTEGER I,J,N,D,MAX,MAXI,I2
      READ (2,1) N
1     FORMAT(I3)
      MAX = 0
      MAXI = 0
      DO 4 I = 1,N
        D = 2
        I2 = I/2
        DO 2 J = 2,I2
          IF (I/J*J .EQ. I) D = D+1
2       CONTINUE
        IF (D .LE. MAX) GO TO 3
          MAX = D
          MAXI = I
3       CONTINUE
4     CONTINUE
      WRITE (3,5) MAX,MAXI
5     FORMAT(' ŠTEVILO Z NAJVEČ (',I3,
     -       ' ) DELITELJI JE',I4)
      STOP
      END
```

```
program delitelji;
 var
  i,j,n,d,max,maxi: integer;
 begin {delitelji}
  readln(n);
  max:= 0; maxi:= 0;
  for i:= 1 to n do
   begin
    d:= 2;
    for j:= 2 to i div 2 do
     if (i mod j) = 0 then d:= d+1;
    if d > max then
     begin max:= d; maxi:= i end
   end;
   writeln('Število z največ (',max:3,
     ' ) deliteljі je',maxi:3)
 end {delitelji}.
```

4. Rešitve so na dlani

$a = b \iff (a \geqslant b)$ and $(b \geqslant a)$
$a \neq b \iff$ not $(b \geqslant a)$ or not $(a \geqslant b)$
$a > b \iff$ not $(b \geqslant a)$
$a < b \iff$ not $(a \geqslant b)$
$a \leqslant b \iff b \geqslant a$

5. Program šteje enice v dvojiškem zapisu števila n. Najmanjši n z 10 enicami je

oblike: $n = 2^{10} - 1 = 1023$

VI. Rešitve nalog za učence po drugem letu pouka računalništva

```
1. program testveriga(input,output);
    const max = 100;
    var
     i,k: integer;
     t: array [1..max] of 0..max;
     videl: array [1.,max] of boolean;
    begin {testveriga}
     for i:= 1 to max do
      begin read(t[i]); videl[i]:= false end;
     read(k);
     while (t[k] <> 0) and not videl[k] do
      begin videl[k]:= true; k:= t[k] end;
     if t[k] = 0 then
      writeln('program VERIGA se konča')
     else
      writeln('program VERIGA se ne konča')
    end {testveriga}.
```

```
C     program testverige
      INTEGER T(100),I,K
      LOGICAL VIDEL(100)
      READ (2,1) T
1     FORMAT (2013)
      READ (2,2) K
2     FORMAT (I3)
      DO 3 I = 1,100
        VIDEL(I) = .FALSE.
3     CONTINUE
4     IF ((T(K) .EQ. 0) .OR. VIDEL(K)) GO TO 5
        VIDEL(K) = .TRUE.
        K = T(K)
      GO TO 4
5     CONTINUE
      IF (T(K) .EQ. 0) GO TO 7
        WRITE (3,6)
6       FORMAT(' PROGRAM VERIGA,SE NE KONČA')
      GO TO 9
7     CONTINUE
        WRITE (3,8)
8       FORMAT(' PROGRAM VERIGA SE KONČA')
9     CONTINUE
      STOP
      END
```

2.
```
program vlak2(output);
 var
  t,dt: real;
  n: integer;
 function cas: real; external;
 function vagon: boolean; external;
 begin {vlak2}
  repeat
   repeat until vagon;
   n:= 0; t:= cas;
   repeat
    if vagon then
     begin
      n:= n+1;
      repeat until not vagon;
      t:= cas
     end;
    dt:= cas-t;
    if dt < 0 then dt:= dt + 24.0*3600.0
   until dt > 60;
   write('Vlak ob ', trunc(t/3600): 2,
     ':', trunc(t/60) mod 60 :2);
   writeln(' je imel', n:3, ' vagonov')
  until false
 end {vlak2}.
```

3.
```
program viator;
 const
  max = 1980;
 var
  t: array [1..max] of integer;
  n,k,i: integer;
 begin
  readln(n);
  for i := 1 to n do
   readln(k,t[k]);
  k := 1;
  while t[k]<>0 do
   begin
    writeln(k);
    i := k;
    k := t[k];
    t[i] := 0
   end
 end.
```

```
C     program viator
        INTEGER T(1980),N,K,I
        READ (2,1) N
1       FORMAT(I4)
        READ (2,2) (K,T(K), I = 1,N)
2       FORMAT(2I4)
        K = 1
3       IF (T(K) .EQ. 0) GO TO 5
          WRITE (3,4) K
4         FORMAT(I6)
          I = K
          K = T(K)
          T(I) = 0
        GO TO 3
5       CONTINUE
      STOP
      END
```

4. *Volumen i-tega kozarca hranimo v VOL(I), količino vode v njem pa v POLN(I).*

```
      REAL VOL(n),POLN(n)

      SUBROUTINE PRELIJ(VOL,POLN,I,J)
C       prelijemo vsebino kozarca I v kozarec J
        REAL RAZ
        RAZ = VOL(J) - POLN(J)
        IF (RAZ .LT. POLN(I)) GO TO 1
          POLN(J) = POLN(J) + POLN(I)
          POLN(I) = 0
        GO TO 2
1       CONTINUE
          POLN(I) = POLN(I) - RAZ
          POLN(J) = VOL(J)
2       CONTINUE
      RETURN
      END
```

```
const stkozarcev = n;
type
 kozarec = 1..stkozarcev;
 problem =
  array [kozarec] of
   record
    volumen, polnost: real
   end;

procedure prelij (var kozarci: problem;
                                  i,j: kozarec);
 var razlika: real;
 begin
  razlika:= kozarci[j].volumen -
                              kozarci[j].polnost;
  if razlika >= kozarci[i].polnost then
   begin
    kozarci[j].polnost:= kozarci[j].polnost +
                              kozarci[i].polnost;
    kozarci[i].polnost:= 0
   end
  else
   begin
    kozarci[i].polnost:= kozarci[i].polnost -
                                        razlika;
    kozarci[j].polnost:= kozarci[j].volumen
   end
 end;
```

5. *Glej rešitev 5. naloge po prvem letu.*

TABELA 1: Število udeležencev na vseh štirih tekmovanjih

	1977	1978	1979	1980
po 1. letu	?	52	56	59
po 2. letu	?	27	36	29
Skupaj	47	79	92	88

Opomba: V letu 1977 tekmovanje ni bilo ločeno v dve skupini.

TABELA 2: Število tekmovalcev in povprečen uspeh po šolah (največje število točk je 100)

	1. leto	2. leti
Elektrotehniška šola Ljubljana	9 50.44	
Gimnazija B. Ziherl Škofja Loka	3 64.00	
Gimnazija Brežice	1 80.00	2 37.50
Gimnazija J. Vege Idrija	1 49.00	
Gimnazija Jesenice	3 27.33	5 13.80
Gimnazija Koper	4 47.00	
Gimnazija Kočevje	2 19.50	
Gimnazija Ljubljana Bežigrad	9 59.11	8 80.13
Gimnazija Ljubljana Šentvid	3 55.00	5 15.80
Gimnazija Ljubljana Vič	1 64.00	
Gimnazija M. Zidanška Maribor	3 50.67	2 56.50
Gimnazija Nova Gorica	3 40.67	3 28.33
Gimnazija Novo mesto	2 19.50	4 61.25
Gimnazija Trbovlje	5 27.80	
ŠC KMP Ljubljana	2 29.00	
ŠC Vojvodina Tolmin	2 21.00	
Tehniška šola Maribor	2 53.50	
Tehniška šola Celje	1 57.00	
Tehniška šola Trbovlje	3 28.33	
S k u p a j	59 44.85	29 45.07
Število šol	19	7

POLEMIKA O RAČUNALNIŠTVU IN INFORMATIKI

ČEMU TOLIKŠNA BIROKRATIZACIJA?

F. Žerdin, Ljubljana

Upravičeno, pa tudi neupravičeno se jezimo nad vedno večjo birokratizacijo naše družbe. Razvoj medsebojnih odnosov postaja vedno bolj kompleksen, zato je povečanje birokracije do neke mere povsem smotrno, vendar ne kakršnekoli. Potrebujemo bolj šolano in bolje opremljeno birokracijo, ki bo zmogla današnje in jutrišnje zahteve naše samoupravne družbe.

V zadnjem času se vse več ljudi spotika nad naraščanjem birokracije v naši družbi. To postaja tema dneva. Neki karikaturist si predstavlja našega Guliverja kot lepo rejenega in mogočnega birokrata, ki leži na soncu, od daleč ga pa hodijo gledat pridni delavci. Meni se zdi resnica malo drugačna. Odkar smo izredno pomembne odločitve v gospodarstvu prenesli spet na birokracijo, se po raznih SIS in še kje kakšen birokrat (ali birokratinja) trudi, da bi po svojih najboljših močeh rešil izredno zapletene probleme. Dostikrat mu (ji) namreč neprofesionalni predsednik ustrezne komisije po dolgem sestanku le površno naroči, kaj je treba storiti glede na sprejete sklepe. Čez kakšen dan, ko je delo v polnem teku, pride za to področje odgovorni profesionalni delavec in zahteva povsem drugo smer obdelave problema, ker da so se na ponovnem sestanku dokončno drugače sporazumeli.

Dokler je šlo pri takem "dogovarjanju" le za kakšna gradbena dovoljenja individualnih graditeljev, družbe kot celote ni dosti prizadelo. Toda trenutno gre za veliko več. Takorekoč čez noč zahtevamo od istih ljudi povsem drugačen in veliko bolj učinkovit način dela, čeprav jih ni nihče na to pripravil. Pri tem nam sploh ne bi zadoščalo, četudi bi postali ti delavci 10 ali 20% produktivnejši,ker problemi, ki naj bi jih reševali, rastejo tako hitro, da njihovo rast lahko merimo le s faktorji.

Problem je torej veliko hujši. Naša samoupravna demokracija omogoča, da smo netolerantni eden do drugega, da vidimo le svoje probleme, namesto da bi se poglabljali tudi v probleme drugih in tako skušali priti do najustreznejše rešitve. Mi pa skušamo drug drugega prepričati o svojem prav, ker smo bolj pametni kot nasprotna stran ali bolj ugledni ali zastopamo to in to nerazvito področje itn. Neracionalnim utemeljitvam ni ne konca ne kraja.

Ali se potem še lahko čudimo neučinkovitosti naše administracije. Prav poučen primer se je zgodil pred kratkim, ko je zvezna skupščina pokarala zvezni izvršni svet, ker ni pripravil ocene koliko deviz smo prihranili, odkar se dva dni v tednu ne vozimo. Toda predstavnik zveznega izvršnega sveta ne more nič drugega storiti kot prositi za podatke predstavnike republiških in pokrajinskih izvršnih svetov, ti pa morajo spet prositi ali zahtevati podatke od nekoga drugega. Možnosti je več kot preveč, da se tokokrog pri iskanju podatkov, ki praviloma niso eksaktni, nekje pretrga, četudi zahtevamo take podatke predčasno, ne pa v konkretnem primeru, ko je že sama zahteva bila poslana prepozno.

Če bi se nam kaj takega zgodilo v šolstvu, bi gotovo zahtevali še eno reformo, četudi bi prejšnjo komaj končali. Pri administraciji se zadovoljimo z javnimi opomini. Toda to je premalo. Potrebno je dejansko spremeniti način dela in

način miselnosti. Take spremembe niso možne čez noč, še zlasti, ker moramo vpeljati nova orodja - računalnike. Dosedanje izkušnje kažejo, da ob uvajanju računalništva iz raznoraznih razlogov zanemarjamo predvsem področje izobraževanja širokega kroga tako neposrednih kot tudi posrednih uporabnikov. Če bomo hoteli torej izboljšati učinkovitost administracije na vseh nivojih, bo potrebno šolati poleg tistih delavcev, ki bodo neposredno pripravljali podatke za računalnike, tudi vse voljene in druge funkcionarje in to od občine preko republike do federacije. Sicer ne bo izobraženih uporabnikov rezultatov, pa tudi nihče ne bo vedel naročiti kaj dejansko rabi.

Bliža se čas, ko bomo morali dojeti, da je ažurna, adekvatna in popolna informacija toliko vredna kot kakršenkoli drugi vir, n.pr. denar, strojne kapacitete ali delavci. Življenje pričakuje od nas takšno spoznanje že danes. Zaradi pomanjkljivih informacij poteka samoupravno dogovarjanje pri nas, vključno z dogovarjanjem na nivoju federacije, prepočasi. To pa nas vrača nazaj v kremplje birokracije. Začeti bo treba z vsem tistim delom tudi v upravi, ki ga že bolj ali manj uspešno izvaja združeno delo na področju uvajanja računalništva. Nič več le globokih in lepih misli o računalniško podprtih informacijskih sistemih v upravi, ampak trdo in načrtno delo, ki bo začelo čimprej dajati prve sadove, v začetku zelo skromne, nato pa vedno večje. Večina republik in pokrajin se je prikopala do kakšne strokovne institucije na področju informatike, saj je celo nam v Sloveniji uspelo ustanoviti Center za informacijski sistem v upravi, ki smo ga dolgo in željno pričakovali. Še nekaj krepkih dogovorov na nivoju federacije, kako si izmenjavati informacije med seboj, kakšnim standardnim proceduram bo morala zadoščati informacija, da bo izmenljiva v vertikalni in horizontalni smeri in delo lahko steče.

Zanimivo bi bilo izvedeti koliko smo mi investirali v obdelovalce podatkov, ki jih ponavadi zmerjamo z birokrati. V razvitih deželah ugotavljajo, da je višina naložb v gospodarstvo okrog 25-krat večja na posameznega delavca kot v negospodarstvo. Pri nas to razmerje ne more biti tako drastično veliko zaradi znatnega vlaganja v negospodarstvo, vendar še vedno dovolj veliko. Potrebno bo načrtno vlaganje v delovne pripomočke obdelovalcev podatkov, da bomo lahko upravičeno pričakovali znatni dvig produktivnosti. Sem ne sodijo samo majhni, srednji, veliki in zelo veliki računalniki kot jih pogosto poimenujejo. Sem sodijo tudi tekstovni procesorji, sem sodi avtomatizacija pisarn, predvsem pa sodi sem možnost medsebojnega povezovanja vseh teh naprav. Ta povezava je tehnološko že izvedljiva, saj imamo že tudi pri nas več načrtovanih računalniških mrež.

Hujši problem predstavlja ustrezno vsebinsko povezovanje. Če imamo na voljo računalnik, bi si želeli, da bi pri pisanju tekstov poklicali le določen program, ki bi izdelal na razpoložljivem računalniku določene kvantifikatorje in jih vgradil v tekst. Tako bi si zagotovili pravo, enkratno in popolno informacijo. Podobno naj bi imel pisarniški avtomat pri poslovodnem odboru doseg do zgoščenih in ažurnih informacij, ki bi jih lahko sproti posredoval članom odbora. Tak pisarniški avtomat bo imel še kopico drugih nalog kot je zbiranje telefonskih pogovorov, zbiranje poročil iz drugih sektorjev ali delavcev, pripravljanje terminskih planov, vključevanje in sodelovanje pri telekonferencah, če naštejemo nekatere.

Če si kritiki žele takšne "birokracije", ki bo zmogla zadostiti tem skiciranim današnjim in jutrišnjim potrebam, je kritika upravičena. Nesposobne in premalo izobražene birokracije imamo res preveč, čeprav bi morali kje in kdaj tudi premisliti, zakaj je do tega prišlo. Zavedati se moramo, da se bomo z razvojem birokratizirali, ker bomo potrebovali vedno več ustreznih informacij. V razvitih družbah se približuje število obdelovalcev podatkov in informacij že polovici vseh zaposlenih. Pri nas se bližamo nekako 40% vseh zaposlenih. Spoznati pa moramo, da lahko na tem področju dvignemo produktivnost dela le, če začnemo intenzivno vključevati računalnike in znanje, sicer bomo kmalu imeli procentualno več birokracije kot kjerkoli drugje, vendar bo vseeno povsem neučinkovita. Torej postavimo racionalne in sodobne zahteve po modernizaciji vsega našega negospodarstva tudi v srednjeročni plan. Začnimo z modernizacijo danes, ne jutri. Ne pozabimo, da nam nesposobna birokracija povzroča ogromno škodo, ki jo le stežka nadoknadimo.

SLOVENSKO DRUŠTVO INFORMATIKA
LJUBLJANA

LJUBLJANA, 16.5.1980

ZAPISNIK SEJE IO SDI, Z DNE 9.5.1980

Prisotni: S.Divjak, R.Faleskini, J.Grad, B.Horvat, A.Jerman-Blažič, F.Mandelc, M.Mekinda, M.Špegel, A.P.Železnikar in F.Žerdin.

Dnevni red:

1) Potrditev in pregled zapisnika zadnje seje (28.3.1980)
2) Predlog organizacije simpozija Informatica 80
3) Razno

Ad 1)

Sklepi iz zapisnika zadnje seje (28.3.1980) so bili izvršeni z izjemo sklica Založniškega sveta časopisa Informatica. IO SDI je potrdil zapisnik.

Ad 2)

Institut Jožef Stefan je predložil pogodbo za organizacijo simpozija Informatica 80. Pripombe na pogodbo je pripravil predsednik IO SDI, tov. M.Špegel pa je obrazložil pogodbo s strani predlagatelja (IJS). V izčrpni razpravi so sodelovali vsi prisotni. IO SDI je izglasoval tale sklep: Simpozij Informatica 80 ne bo organiziran. SDI objavi ta sklep v republiških in pokrajinskih dnevnih časopisih s pojasnilom, da se SDI pridružuje ustalitvenim ukrepom, vendar bo simpozij Informatica 81 organiziran na Bledu. Potencialni avtorji lahko pošljejo svoje prispevke za objavo v časopisu Informatica, ki bo v te namene posebej razširjen.

Ad 3)

Za pridobitev prostorov SDI in nastavitev tajnice je imenovana komisija v sestavi R.Faleskini, A.Jerman-Blažič, A.P.Železnikar in F.Žerdin. Naloga komisije je, da predloži do naslednje seje IO SDI konkretno obliko pridobitve prostorov in pripravi razpis za tajnico SDI, takoj ko bodo zagotovljeni osnovni pogoji za njeno delo. Tov. A.Jerman-Blažič je zagotovil, da bo pridobil mlajšega sodelavca za opravljanje naloge sekretarja SDI.

Zapisnikar:

A.P.Železnikar

OBVESTILO NAROČNIKOM

Cenjene naročnike obveščamo, da se je z letnikom 1980 povišala naročnina (delovne organizacije 350,00 din, posamezniki 120,00 din)

UREDNIŠTVO

Iskradata

ZA VEČJO PRODUKTIVNOST

ISKRADATA računalnik C 18

ISKRADATA mikroračunalnik 1680

Področje uporabe:

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

NOVICE IN ZANIMIVOSTI

****
STROKOVNI SESTANEK:
MIKROPROCESORSKI RAČUNALNIKI
****

Zavod za elektroniko Elektrotehnične fakultete v Zagrebu naj bi organiziral ta strokovni sestanek 24. in 25. septembra 1980, če bo zanj dovolj veliko zanimanje. Teme zasedanj tega sestanka pa naj bi bile: arhitektura in organizacija mikroprocesorjev, zbirni sistem in periferija, sistemska programska oprema ter aplikativna programska oprema. Pojasnila daje mr. B. Kette, Elektrotehnički fakultet, Unska ulica 17/XI, 41000 Zagreb. Kotizacija 2500.- din.

A.P.Železnikar

****
PREČNA PROGRAMSKA OPREMA ZA Z8000
****

Programsko opremo za 16-bitni mikroprocesor Z8000 lahko razvijamo z domačim miniračunalnikom Delta, če uporabimo prečno programsko opremo, ki vsebuje prevajalnik za programirni jezik C, kodni optimizator za ta prevajalnik, zbirnik za Z8000, povezovalnik in nalagalnik. Naslov proizvajalca te programske opreme je: Zilog (UK) Ltd, Babbage House, King St, Maidenhead, Berks SL6 1DU, England.

A.P.Železnikar

ISKRA NA HANNOVERSKEM SEJMU

MED DRUGIM JE ISKRA NA HANNOVERSKEM SEJMU PRIKAZALA TUDI DVA SVOJA NOVA MIKRORAČUNALNIŠKA SISTEMA, IN SICER ISKRADATA 1680-20/21 DATA ENTRY SYSTEM TER ISKRADATA 1680-10 MICROCOMPUTER DEVELOPMENT SYSTEM. OBA NAVEDENA SISTEMA STA PLOD DOMAČEGA RAZVOJA, OBLIKOVANJA IN PROIZVODNJE.

****
RAZLIČNE ZANIMIVOSTI
****

Shugart napoveduje proizvodnjo novih minigibkih diskov za 500K zlogov, z oznako SA450. Pri tem uporablja glavi podjetja Tandom Magnetics Corp. ... Dataland iz Danske uvaja računalniški sistem, ki pretvori skladateljevo glazbo v tiskano partituro. Uporablja posebno tastaturo, ki omogoča, da se vnešeni zvoki pretvarjajo v note. Računalnik obdela vhod in pošlje izhod na digitalni risalnik, ki izpiše partituro. ... Intel proizvaja 16-bitne mikroprocesorje 8086, ki imajo taktno hitrost 8 MHz. Prejšnja največja hitrost teh procesorjev je bila le 5 MHz. ... Ameriško ministrstvo za obrambo napoveduje povečanje stroškov izdelave programske opreme iz sedanjih 40 dolarjev na vrstico na 65 dolarjev na vrstico v letu 1984. Tako bo izdelava programske opreme zajela že 8% celotnega proračuna v vojne nameone ZDA in se bo povzpela iz 6,6 milijard dolarjev v letu 1979 na 10,5 milijard dolarjev v letu 1984. ... Motorola je uvedla optoizolatorje za 7,5 kV.

A.P.Železnikar

****
RADIO SHACK V LETU 1979
****

Radio Shack je v preteklem letu prodal v svojih trgovinah za 100 milijonov dolarjev mikroračunalnikov. Od tega je bilo 150.000 računalnikov TRS-80. Izvedenci ocenjujejo, da ima Radio Shack 30% tržišča osebnih računalnikov, vendar se je pri modelu TRS-80 že pokazalo določeno zasičenje tržišča. V preteklem letu je začel Radio Shack dobavljati TRS-80 Model II in je prodal 1000 teh sistemov. V letu 1980 se bo to število povzpelo na 15.000. Radio Shack bo uvedel tudi barvni prikazovalnik za sisteme TRS-80 in tako povečal tržno zanimivost masovnega izdelka TRS-80.

A.P.Železnikar

****
EPROM ZA 64K BITOV
****

Motorola je začela deliti vzorce za EPROMe za 64K bitov, ki imajo organizacijo 8K zlogov. Prodaja naj bi stekla v začetku drugega polletja. Vezje ima 24 nožic. Intel in TI bosta izdelovala podobno vezje z 28 nožicami. Medtem je padla cena za klasične EPROMe 2708 že tudi pod 6 dolarjev. Povpraševanje po EPROMih 2716 je še vedno izredno in njihova cena še vedno dokaj visoka ($ 20 do 24). TI je trenutno največji proizvajalec EPROMov ter pokriva 38% tržišča, Intel 29%, Fujitsu in Hitachi pa vsak po 8%.

A.P.Železnikar

****
VELIKA POMNILNA VEZJA
****

Japonski proizvajalci napovedujejo nova velika polprevodniška pomnilna vezja, ki so vzbudila pozornost tudi pri ameriških inženirjih. Američani očitajo Japoncem, da so 64k-bitna vezja pravzaprav njihova; to naj bi veljalo za dinamične pomnilnike tipa RAM. Matsushita pa je napovedala 64k statični RAM in NTT Musashino 256k dinamični RAM, NEC-Toshiba 256k dinamični RAM, ki se prilega v podnožji za 16k in 64k dinamični RAM s 16 nožicami, Hitachi napoveduje 5V 64k dinamični in 16k statični RAM z uporabo HCMOS tehnologije. Če se bodo te napovedi uresničile, bodo Japonci zares postali pomnilniška velesila ter bodo vsaj za nekaj časa prehiteli Američane.

A.P.Železnikar

****
PAKETNO PREKLAPLJANJE
****

Western Digital dobavlja nov krmilnik za paketno preklapljanje, z oznako 2501, ki je namenjen za HDLC (High-level Data Link Control) paketno preklapljanje v računalniških in podatkovnih mrežah. Vezje ima 48 nožic in zmore celoten CCITT X.25 protokol in povezavo z navadnim 8-bitnim mikroprocesorjem. Vezje realizira celotno HDLC mehaniko. Po ukazu iz gostiteljskega mikroprocesorja generira podatkovne zastavice, naslovna polja in krmilna polja. Ima tudi direkten dostop do podatkov v pomnilniku ter pošilja podatke z DMA načinom. Končno generira še korekcijski, 16-bitni preizkusni karakter ter ga pripne k oddanemu podatkovnemu toku in konča paket. Prenosne hitrosti so 1 Mbit/s ali več; cena za OEM

količine (vsaj 100 kosov) je $ 200 ali manj. Vezje 2511 pa bo podpiralo HDLC protokol v izravnalnem načinu.

A.P.Železnikar

****
MIKRORAČUNALNIKI V PRIHODNOSTI
****

Tržišče mikroračunalnikov in mikroprocesorjev bo iz 1 milijarde dolarjev letos naraslo na 6 milijard dolarjev v letu 1985. Tehnologija in uporaba napredujeta na tem področju še vedno prehitro, da bi bilo moč razviti smiselne ocene. Mikroračunalniki postajajo vsebolj računalniki v enem integriranem vezju. V ZDA imamo več kot 25 proizvajalcev mikroprocesorjev, 15 izmed njih proizvaja mikroračunalnike v enem vezju. Najenostavnejši mikroračunalniki imajo ceno pod 3 dolarji in se uporabljajo kot krmilniki v mikrovalovnih pečeh, v igrah, v avtomobilih in drygje. Večji in sposobnejši procesorji se uporabljajo v blagajnah, procesni instrumentaciji, pri zbiranju podatkov itn. 8-bitni procesorji so v terminalih, v napravah za procesiranje tekstov, obračunavanje, v osebnih računalnikih in v majhnih poslovnih sistemih. 16-bitni mikroprocesorji predstavljajo višjo stopnjo in najdemo jih v inteligenčnih terminalih in v miniračunalnikih. Kmalu se bodo pojavili še mikroračunalniki za krmiljenje vodil, ki bodo dovolj hitri. Tem se bodo pridružili t.i. vodilnousmerjeni mikroračunalniki, ki bodo lahko specializirali vsebovano arhitekturo tudi v centralnem procesorju. To bodo dodatki k obstoječim 16-bitnim mikroprocesorjem. Pojavili se bodo tudi analogni mikroračunalniki za procese v realnem času. Posebni mikroračunalniki bodo t.i. sistemi v enem vezju za potrebe krmiljenja okolice, v telekomunikacijah in v osebnih računalnikih. Od 11 milijonov naprav letos bo proizvodnja narasla na 118 kosov v letu 1985. Povečala se bo znatno tudi hitrost mikroprocesorjev itn. itn.

A.P.Železnikar

****
RAČUNALNIŠKO STAVLJENJE
****

Donald Knuth, avtor znane zbirke The Art of Computer Programming, je napisal delo TEX and METAFONT, New Directions in Typesetting, ki opisuje novo metodo stavljenja tekstov. Knuth pojasnjuje, kako je moč TEX, ki je bil prvotno razvit za stavljenje matematičnih in tehniških tekstov, uporabiti za računalniško stavljenje. METAFONT pa je sistem za oblikovanje abeced, ki je primeren za realizacijo na napravah s sistemom rasterja, ki tekst tiskajo ali prikazujejo. S tem sistemom lahko računalniki oblikujejo nove oblike karakterjev praktično takoj. TEX in METAFONT predstavljata izboljšavo stavljenja, ki bo lahko bistvena za stavljenja znanstvenih in tehniških tekstov. Knjiga je sestavljena iz treh delov. Prvi del opisuje matematično tipografijo. Ostala dva dela opisujeta TEX in METAFONT. Cena knjige je 12 dolarjev in jo je moč naročiti pri Dept TM:X, Digital Press, Educational Services, Digital Equipment Corp., 12-A Esquire Rd., N. Billerica, MA 01862.

A.P.Železnikar

****
LISP ZA Z-80 SISTEME
****

LISP Company je napovedala različico jezika LISP za Z-80. Ta različica z oznako TLC-LISP omogoča manipulacijo funkcij kot podatkovnih objektov, objektno usmerjeni programirni stil, definira funkcije s spremenljivim številom parametrov, vsebuje mehanizme za strukturirano iteracijo in nestrukturirane izstope, vsebuje procesiranje nizov in karakterjev ter fiksno in plavajočo aritmetiko. Jezikovni sistem vsebuje še tabelnoupravljani razpoznavnik, krmiljenje pri napakah in samodejno nalaganje, ki "virtualizira" redko uporabljane funkcije in konstante k diskovnim zbirkam in tako sprošča programirni pomnilnik. TLC-LISP je prirejen za Z-80 CP/M šistem, cena je 150 dolarjev. Naslov proizvajalca: LISP Co.,POB 487, Radwood Estates, CA 95044.

A.P.Železnikar

****
GIBKI DISK ZA ČIŠČENJE GLAVE
****

Na tržišču se je pojavil bistveni element za sisteme z gibkimi diski, in sicer disk, ki očisti glavo. Uporaba takega diska ne povzroči obrabe glave in ji zagotavlja čistost v okviru industrijskih standardov za magnetne medije. Disk vstavimo v pogon ter pritisnemo nanj v obratovanju glavo za 30 sekund. Praviloma opravimo ta postopek vsak dan kot zaščito proti oksidaciji glave. Čistilni diski so 5-1/4 in 8 colski in cena posamičnega diska je 20 dolarjev ali 45 dolarjev za tri diske. Pri vsakodnevnem čiščenju je disk uporaben tri mesece. Naslov proizvajalca: Lifeboat Associates, 2248 Broadway, New York, NY 10024.

A.P.Železnikar

****
MICROSYSTEMS - NOVI ČASOPIS
****

S-100 Microsystems je nov časopis za uporabnike mikroračunalniških sistemov z vodilom S-100. Vsebina tega časopisa naj bi obravnavala problematiko povezave na vodilo S-100, operacijski sistem CP/M, jezik Pascal, programsko opremo v zvezi z zbirniki ter v jezikih FORTRAN, BASIC itn. Časopis bo pokrival tudi problematiko 16-bitnih mikroprocesorjev, multiprocesorjev, paralelnega procesiranja, časovnega razdeljevanja, procesiranja tekstov, razvoja sistemov, upravljanja podatkovnih baz ter uporabe v znanosti in drugje. Problematika časopisa bo vezana na S-100 sisteme, kot so npr. Cromemco, North Star, Intersystems, IMSAI, Poly Morphics, Processor Technology (Sol), Xitan in drugi.

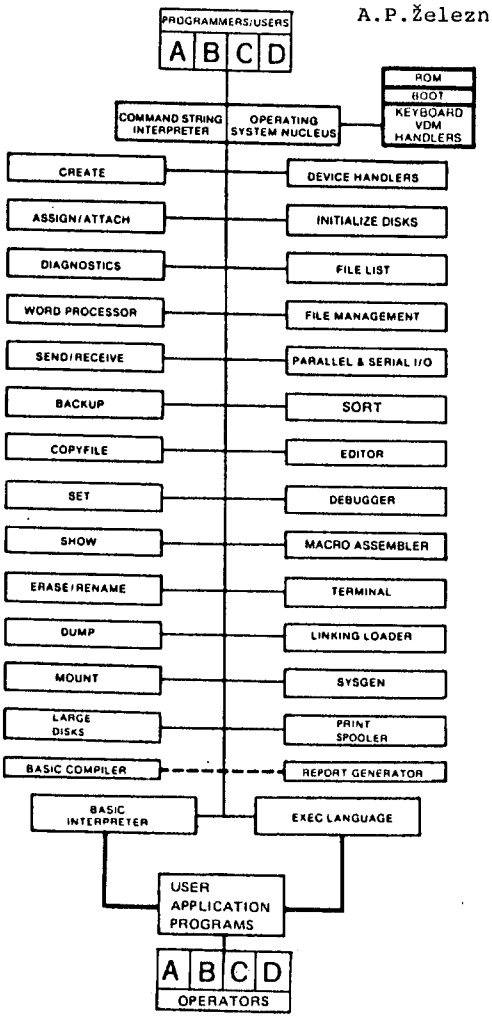
Urednik časopisa S-100 Microsystems je Sol Libes, ki je napisal 13 knjig in vrsto člankov v različnih časopisih. Je predsednik največje organizacije za osebne računalnike, to je Amateur Computer Group v New Jerseyu. Prva številka S-100 Microsystems prinaša celoten S-100 standard (IEEE), uvod v sistem CP/M, modifikacijo videoplošče za pascalske urejevalne funkcije, inverzni zbirnik za 8080A itn. Časopis bo izhajal šestkrat letno, cena posamezne številke pa je 2 dolarja. Naslov časopisa: S-100 Microsystems, POB 1192, Mountainside, NJ 07092.

A.P.Železnikar

OASIS, OPERACIJSKI SISTEM ZA Z-80

OASIS (Online Application System Interactive Software) je operacijski sistem, ki je za razliko od sistema CP/M napisan za procesor Z-80 (sistem CP/M je napisan za procesor 8080A). Ta operacijski sistem omogača uporabo zbirk tipa ISAM, neizmenljivih in gibkih diskov, prevajalnika za jezik BASIC itn. ter naredi iz mikroračunalnika napravo, ki se lahko meri z miniračunalnikom. OASIS je integriran z drugimi programskimi proizvodi do večuporabniškega sistema. Njegove glavne lastnosti so: sistem za enega ali več uporabnikov, obračunavanje uporabnikov, multiprocesiranje, varnost zbirk in zapisov, gesla in dodatna zaščita, tipkane (ISAM), direktne in zaporedna zbirke, izčrpna dokumentacija za prilagoditev itn. Cena sistema OASIS za enega uporabnika je 150 dolarjev in za prevajalnik jezika BASIC 100 dolarjev. Za sistem z več uporabniki sta ti dve ceni 250 in 145 dolarjev. Sestavni deli sistema OASIS pa so tile: makrojski premeščevalni zbirnik, iskalnik napak, povezovalnik, urejevalnik, izhodni oblikovalnik teksta, paralelno izvajanja uporabniškega programa in tiskanje izhoda (spooler), komunikacijski paket, sortiranje, diagnostika in vzdrževanje, interaktivni jezik EXEC tar prevajalnik BASIC s ponovnim vstopom. K temu je moč dodati še prevajalnika za FORTRAN in COBOL (74 ANSI). Naslov proizvajalca: Phase One Systems, Inc., 7700 Edgewater Dr., Suite 710, Oakland, CA 94621.

A.P.Železnikar



INFORMATICA V REFERATIVNIH ŽURNALIH

Inštitut za znastvene informacije Akademije znanosti ZSSR (VINITI) izdaja 25 serij Referativnih žurnalov z osnovnih področij znanosti in tehnike. Za te prikaze sprejema 25000 znanstvenih in strokovnih časopisov iz 130 držav in njegov namen je polna in ažurna informacija o novejših dosežkih znanosti in tehnike. V zvezi s tem svojim poslanstvom se je sovjetski inštitut obrnil na uredništvo našega časopisa Informatica, da redno pošilja po en izvod časopisa v Moskvo. Uredništvo se je z veseljem odzvalo tej prošnji, saj bodo avtorji člankov našega časopisa deležni odslej tudi te mednarodne pozornosti, seveda le, če bodo njihovi prispevki dovolj kvalitetni in zanimivi. Na ta način bo Informatica dosegla tudi določeno popularnost v mednarodnih strokovnih krogih.

A.P.Železnikar

MEDNARODNA KONFERENCA ZA UMETNO INTELIGENCO IN ROBOTIKO

Mednarodna konferenca Artificial Intelligence and Information - Control Sysytems of Robots se je odvijala na gradu Smolenice blizu Bratislave od 30.6. do 4.7.1980. V mednarodnem programskem odboru je sodeloval tudi A.P.Železnikar, vrsta jugoslovanskih avtorjev pa je prikazala svoje referate. V okviru sekcije za teoretične probleme umetne inteligence je naš prispevek prikazal V.P.Masnikosa: Proof of Realizability of Artificial Intelligence (Beograd), M. Kantardžić: The Protection of Data Bases from the User"s Generating of Forbidden Information (Sarajevo) v sekciji za komuniciranje z računalniki v naravnih jezikih, ostali referati pa so bili še J.Lenarčič, P.Oblak, V.Stanič: Optimization of Nominal Robot Trajectories (Ljubljana), V.Potkonjak, M.Vukobratović: Contribution to the Computer-Aided Design of Industrial Manipulators (Beograd), M.Špegel, F.Dacar: Model-Based Programming of Arc Welding Robots (Ljubljana) in P.Tancig: A LISP Data Base Management System for AI Projects (Ljubljana).

A.P.Železnikar

KRITIČNA PRENOSNIŠKA VEZJA

Oktalno prenosniško integrirano vezje 74LS245 postaja vsled svojega pomanjkanja na tržišču vse dražje. To vezje je uporabno tam, kjer je potrebno priključiti več digitalnih naprav na eno vodilo. Vsaka taka naprava ima svoj naslov in zahteva svoj prenosnik. Vezje 74LS245 zmore seveda tri stanja - nič, ena in plavajoče stanje. Poraba teh vezij je velika zlasti v procesnem krmiljenju in v podatkovnih sistemih z več terminali. Proizvajalci tega vezja ga prodajajo za 1,50 dolarja, toda zaradi pomanjkanja vezja, dosega njegova cena tudi vrednost 9 dolarjev. Največji proizvajalec teh vezij je Texas Instruments, za njim pa so National Semiconductors, Motorola, Fairchild in drugi. Namesto tega vezja se vgrajuje kombinacija oddajnega in sprejemnega vezja, da se tako nadomesti prenosnik.

A.P.Železnikar

IZPOPOLNJENI TERMINALI

Predelava starih terminalov v izpopolnjene lahko pomeni znatno novo tržišče, saj je samo v ZDA milijon starih terminalov in 350 tisoč teleprinterjev. Nekatera podjetja v ZDA so poslala na tržišče izboljševalne sestavljenke, ki omogočajo urejevanje tekstov in njihovo shranjevanje. Te sestavljenke imajo pomnilniške vmesnike za 16k zlogov in lahko pošiljajo snope podatkov v komunikacijski kanal s hitrostjo 1200 zlogov na sekundo ali desetkrat hitreje kot normalni terminali. S tem se zmanjšajo pristojbine za uporabo linije. Majhne mikroprocesorske sestavljenke se vgrajujejo v obstoječe terminale in pod okrove teleprinterjev, sprejemajo enozložne krmilne ukaze, njihova cena pa je do 300 dolarjev.

A.P.Železnikar

PREVAJALNIK PL/I-80

Podjetje Digital Research je začelo dobavljati prevajalnik PL/I-80 in podporno programsko opremo, ki ju je razvijalo dve leti. Jezik PL/I-80 je podjezik jezika PL/I in ustreza priporočilu ANSI za podjezik G, ki je definiran za mikroračunalniške sisteme. Ta prevajalnik se izvaja pod operacijskima sistemoma CP/M ali MP/M, ki sta standardna operacijska sistema za mikroračunalnike s procesorji 8080A, 8085 in Z80.

PL/I-80 uporablja standardne podatkovne tipe: osem- in šestnajstbitna cela števila s predznakom, decimalna števila s 15 številkami za poslovne izračune, nize karakterjev za procesiranje tekstov, binarna števila s plavajočo vejico za znastveno uporabo, nadalje nize bitov, kazalčne spremenljivke ter vstopne in zbirčne podatkovne tipe. Nad temi tipi se uporabljajo matematični, nizni in krmilni operatorji. Zbirčni sistem omogoča aparaturno neodvisen dostop do zbirk v sekvenčni ali naključni obliki (v povezavi s CP/M).

Programski paket PL/I-80 je zaokrožen sistem, ki vsebuje prevajalnik, premeščevalni makrozbirnik RMAC, povezovalni urejevalnik LINK-80, knjižnjičarja LIB-80 in subrutinsko knjižnjico. Vključeni so tudi nekateri PL/I programi v izvirni obliki, ki rabijo kot pomoč pri zagonu sistema. Ti programi segajo od enostavnega kopirnega programa za kopiranje zbirk do zapletenega šahovskega programa. Dokumentacijo sestavljajo: PL/I-80 Reference Manual, PL/I-80 Applications Guide, LINK-80 Operators Manual.

Prevajalnik za PL/I-80 je triprehoden, z optimizacijo koda in potrebuje 38k zlogov pomnilnika za izvajanje, deluje pa lagodno pri 48k CP/M ali 64k MP/M. Programi se prevedejo neposredno v premestljivi strojni kod, se povežejo s subrutinami ali drugimi moduli z uporabo paketa link-80. Ta paket piše tudi simbolno tabelo, ki je uporabna v povezavi s SID (že opisano v prejšnji številki Informatice) ali s simboličnim popravljalnikom napak. Dobljeni programi so v razponu od 800 zlogov do polnega obsega konkretnega računalniškega sistema.

Cena prevajalnika je 500 dolarjev in dobava je na 8" disketi.

A.P.Železnikar

JOSEPHSONOV SPOJ

Supraprevodnost je lastnost nekaterih snovi, da zgubijo upornost za električni tok pri ekstremno nizkih temperaturah in je bila odkrita že leta 1911. Josephsonov efekt je pojav supraprevodnosti, ki ga je odkril Brian Josephson leta 1962 na univerzi v Cambridgu. Josephsonovi spoji so naprave, ki uporabljajo ta efekt. Josephson je na osnovi kvantne teorije napovedal, da bodo elektroni prehajali iz enega supraprevodnika v drugega kot skozi tunel, če bosta ta dva supraprevodnika ločena z izjemno tanko plastjo izolacijske snovi.

Tok skozi spoj lahko krmilimo z magnetnim poljem, tako da namestimo krmilni vodnik v bližino vhodnega supraprevodnika. Na ta način lahko oblikujemo tudi digitalna vrata in z ustreznimi kombinacijami tudi Josephsonove bistabilne elemente. Z Josephsonovimi spoji dobimo tako poljubno zapletena kombinatorna in bistabilna vezja. Izboljšani Josephsonov spoj se imenuje Josephsonov interferometer. Njegova lastnost je zmanjšanje kapacitivnosti in krmilnega toka, tako da se lahko dosežejo ekstremno visoke preklopne hitrosti, in sicer od 30 ps navzgor. Josephsonovi spoji so tako do 100-krat hitrejši od navadnih preklopnih vezij.

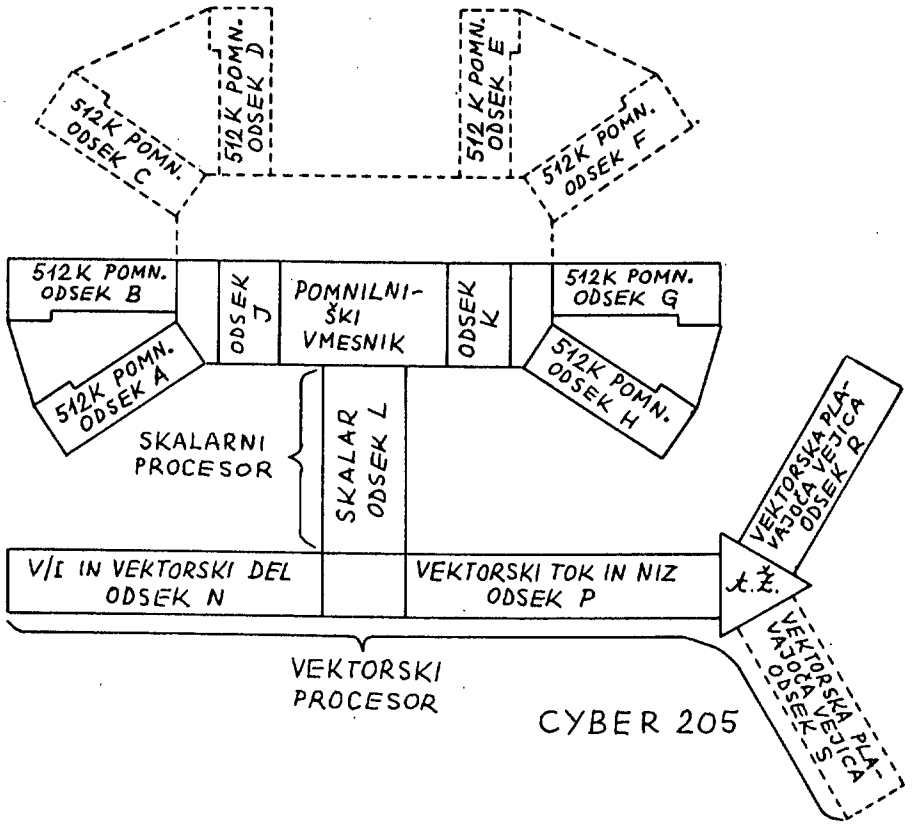
Podjetje IBM je zaradi nizke porabe energije in visoke hitrosti že uporabilo Josephsonov spoj pri razvoju pomnilnika, kjer je bil dosežen čas dostopa pod 7 ns ter preizkusilo prototipni računalnik s temi spoji, ki ima taktno periodo 2,5 ns, kar je osemkrat hitreje od sistema 370/168. Ta računalnik se nahaja v kocki s stranico 4" in obsega CPU, prikriti in glavni pomnilnik za 16M zlogov (128 milijonov bitov) ter zmore 70 milijonov operacij (ukazov) v sekundi. Manj konzervativna različica tega računalnika pa bo imela 128M zložni pomnilnik in bo zmogla milijardo ukazov na sekundo. Največji IBMov računalnik danes ima 6M-zložni pomnilnik, taktno periodo 50 do 80 ns in zmore le 3,5 milijonov ukazov na sekundo.

Seveda nastane ob vsem tem vprašanje smisla takega superračunalnika. Oglejmo si primerjavo, ki jo je pred 25 leti naredil John von Neumann med računalniki in človeškimi možgani. V te namene je uvedel razmerje kvalitete, da je lahko primerjal tedanjo računalniško tehnologijo z biološko zmogljivostjo. Eno teh razmerij je bilo računanje na volumen ali produkt hitrosti vsake osnovne enote (vezja ali nevrona) na njihovo prostorsko gostoto. Drugo razmerje je bila energija na računanje. Za oba primera je ugotovil, da ostajajo možgani superiorni nad računalniki za faktor 10000. Z izboljšanimi lastnostmi računalnikov pa postajajo ti primerljivi z zmogljivostjo možganov. Josephsonovi računalniki bodo prve mehanične naprave, ki bodo presegle zmogljivost možganov, in sicer za faktor 10000 pri računanju na enoto volumna, za 1000 pri računanju na enoto energije in za 100 v celotni računalni moči. Le v obsegu pomnilnikov bodo superračunalniki še vedno pod pomnilno močjo človeških možganov.

A.P.Železnikar

CDC 205 Z 800 MILIJONI OPERACIJ V SEKUNDI

V svetu superračunalnikov, kjer bo na voljo 16M zlogov glavnega pomnilnika in ko se govori o več sto milijonih operacij na sekundo (Mop/s), je podjetje Control Data Corp.



najavilo Cyber 205, ki naj bi v svoji največji konfiguraciji zmogel 800 Mop/s. Cyber 205 je drugi član družine Cyber 200 in je podobno kot njegov predhodnik 203 zgrajen z uporabo visokointegriranih bipolarnih vezij.

Cyber 205 je sestavljen iz skalarne in vektorske enote, kot kaže slika. Skalarna (serijska) enota opravi eno samo operacijo ali ukaz do največ 50 Mop/s in deluje s 64-besednim ukaznim skladom in z 256-besedno registrsko zbirko. Vektorska (paralelna) enota opravlja aritmetične operacije na poljih ali tokovih podatkov in dopušča 32- ali 64-bitne operande k in iz centralnega pomnilnika vsakih 20 ns. Centralni pomnilnik sprejme 4 milijone 64-bitnih besed in podatki med pomnilnikom in centralnim procesorjem se prenašajo s hitrostjo 25,6 milijard bitov na sekundo pri milijonbesednih inkrementih centralnega pomnilnika.

Osnovni sistem Cyber 205 ima 8 do 16 V/I vrat s prenosno hitrostjo 200 milijonov bitov na sekundo. Dva trilijona besed virtualnega pomnilnika sta na razpolago uporabniku. Sistem deluje z operacijskim sistemom 200-OS, ki omogoča dostop k sistemu preko paketnih in interaktivnih terminalov ali preko lokalnih naprav v okviru končnih procesorjev.

Cena sistema Cyber 205 je od 7,9 do 16,5 milijonov dolarjev za t.i. tipično konfiguracijo, dobavljati pa ga bodo začeli v januarju 1981. Pred leti so se največji računalniki uporabljali v nuklearni in visokoenergijski fiziki, danes pa bodo našli uporabo pri iskanju alternativnih virov energije, pri dolgoročnih vremenskih napovedih ter pri oblikovanju in proizvodnji letal. Naftna industrija potrebuje velike računalnike za obdelavo potresnih podatkov in za simulacijo. Žal ni nikjer nakazana uporaba tega superračunalnika v administrativne namene.

A.P.Železnikar

****
NOV IBMov MIKRORAČUNALNIK
****

Podjetje IBM ne stoji ob strani v mikroračunalniški revoluciji, čeprav večina uporabnikov velikih sistemov to misli. Zapleteni veliki sistemi in predvsem računalniške mreže predvidevajo na svojem t.i. spodnjem koncu sistema izdatno uporabo mikroračunalnikov. V prihodnje naj bi bil vsak računalnik vključen v mrežo, tudi osebni in mali poslovni računalniki.

IBM uvaja nov namizni računalnik, tj. model 5120, ki ga prodaja za 13500 dolarjev ter ima 16k zlogov programirljivega pomnilnika in BASIC ali APL v pomnilniku tipa ROM. Mikroračunalnik IBM 5105 pa se bo pradajal za 4500 dolarjev in bo proizvajan na Japonskem. Ta mikroračunalnik bo uporabljal vodilo S-100 in bo v tem smislu tudi razširljiv. Imel bo 16k zlogov pomnilnika, hitri magnetni trak in toplotni tiskalnik. Večterminalna različica tega mikroračunalnika bo imela oznako 5130.

A.P.Železnikar

****
DISKI TIPA WINCHESTER
****

Večkrat govorimo o diskih tipa Winchester ali winchesterskih diskih, brez da bi točno vedeli, za kakšno vrsto diskovnih naprav pri tem gre. Bistvo diskovne winchesterske tehnologije je v majhnih, lahkih glavah, ki mehko pristanejo na plošči in v zapečateni in proti onesnaženju varni komori, v kateri so nameščeni pogon in glave. Razvoj takšne diskovne tehnologije je bil povzročen zaradi zahtev po večji gostoti podatkov na disku, ko

naj bi se znižala cena na bit in dosegla hkrati še večja zanesljivost. Za povečanje gostote brez povečanja zapisne površine sta pomembna dva parametra. Najprej povečamo število stez (sledi) na površinsko enoto, potem pa povečamo še število bitov na vsaki stezi.

Izboljšanje v določenosti signala in povečanje zapisne gostote se doseže z zmanjšanjem razdalje med glavo in ploščo. Dodatno se zahteva večja natančnost položaja stez, da se tako doseže večja čitalna zanesljivost. Položajna odstopanja stez se pri navadnih diskih pojavljajo zaradi toleranc pri zamenljivih diskih, v sami vrtavki in v nameščevalnem mehanizmu. Te razlike se pojavljajo že med posameznimi diskovnimi pogoni, pa tudi med nameščanjem ploče v pogon. Tako sta dve nadaljni metodi za povečanje gostote zapisa na ploščo, da je plošča nezamenljiva in da tečejo glave bližje površine plošče. Bližje gibanje glave pa zahteva odstranitev vsakega prahu, torej komoro, v kateri kroži zrak, ki je bil predhodno filtriran.

Važno je tudi gibanje glave pri zagonu in ustavitvi diska. Pri navadnih diskih se glave približajo in odmaknejo, ko je disk še ali že v gibanju. Ta postopek je nestabilen in večkrat pride do dotika glave in diska, kar povzroči oksidacijo in obrabo glave. V winchesterskem disku to ni dopustno in glava pristane na disku šele tedaj, ko se je ta že ustavil. Ta način pa zahteva minimalen pritisk glave na površino diska in posebno prevleko diska, ki zmanjša možnost oksidnega prenosa. Tako je pritisk winchesterske glave le 8 do 10 gm, pri navadnih glavah pa 40 do 60 gm. Majhna glava zmanjšuje tudi zračno upornost, tako da je pri majhni razdalji položaj glave stabilen. Glave pristajajo ob ustavitvi v posebni coni in ne na površini zapisne steze, pri zagonu vrtavke pa se dvignejo že po eni tretjini ali polovici obrata plošče.

Druge, dodatne lastnosti winchesterskih diskov so še stezam sledeči servomehanizmi, dvojne glave na površino, enosmerni motorji vrtavk in odstranitev vsakega mehanizma od zapečatene komore. Te naprave dosežejo m.t.b.f. nad 20000 ur, dočim je m.t.b.f. usmernika, dodatne elektronike in motorja med 5000 in 10000 urami. Srednja cena teh diskovnih naprav je 3000 dolarjev, maksimalno število zlogov pa znaša 160M (Ampex). Nekoliko višje so tudi cene evropskih proizvajalcev, ki dosežejo vrednosti 4000 funtov.

A.P.Železnikar

****
PREIZKUŠANJE MIKRORAČUNALNIŠKIH SISTEMOV
****

Mikroračunalniške sisteme na terenu lahko preizkušamo s posebno programsko opremo ali ročno, kot bomo opisali. Taki preizkusni postopki so priporočljivi tudi tedaj, ko nimamo na razpolago videoterminala. Vzemimo, da imamo mikroračunalniško konfiguracijo, ki je sestavljena iz CPE, nekaj RAMa in EPROMa, iz piskalnika (vmesnik, zvočni generator, zvočnik), serijskih in paralelnih vrat, programirljivega časovnika, analognih senzorjev itn. Za takšen sistem lahko izdelamo posebno avtodiagnostično programsko opremo.

Ta programska oprema (v obliki firmwara) ocenjuje do določene stopnje vse sistemske sestavne dele, in sicer z začasnim samopreizkusom, avtomatičnim samopreizkusom in z ročno avtodiagnostiko. Ta preizkus je omogočen z uporabo mikroprocesorja v sistemu in izboljšuje pogoje terenskega servisiranja. Začasni preizkus se opravi na minimalnem številu delujočih sestavnih delov. Npr. piskalnik se priključi direktno na vodilo mikroprocesorja in preizkušajo se signali k posameznim komponentam. Tako se preiskusi delovanje vseh naslovnih vodov in podatkovnih linij pa tudi generiranje vseh naslovov in podatkovnih kombinacij, ko slišimo pravilne signalne vzorce. Nepravilni piskalni vzorci kažejo na konkretne napake v sistemu, ki jih lahko identificiramo.

RAM preizkusimo tako, da uvedemo dva programska števnika, ki štejeta naprej od 0 do 65535 in nazaj in opazujemo, ali sta istočasno dosegla ničlo z uporabo posebnega podprogramskega poziva (ukaz CALL). To je t.i. stabilnostni preizkus. Z ostalimi testi preizkušamo delovanje različnih sestavnih delov, in sicer najprej tiste z manjšim številom delovnih delov.

Avtomatični samopreizkus opravi bolj izčrpno vrednotenje posebno RAMa in EPROMa. S t.i. drsečim vzorcem (posebnim algoritmom) preizkusimo RAM, EPROM pa s posebno preizkusno vsoto za vsako integrirano vezje, tako da ugotovimo napačno delujoče vezje. Nekateri EPROMi zgubljajo bitne informacije po nekaj urah njihovega oblikovanja (zapisa). Tako lahko ugotovimo te napake že v fazi izdelovanja, kar nam prihrani obilo kasnejšega časa pri iskanju napake. Seveda lahko preizkušamo tudi analogne aktuatorje, ki so priključeni na mikroračunalniški sistem.

Ročna diagnostika lahko temelji na uporabi telefonske linije s servisnim centrom ali na testiranju, ki ga opravi servisni tehnik na terenu. Samopreizkusne funkcije obsegajo tele operacije: vključitev vseh LEDov, korakanje skozi lokacije RAMa, prikaz koda za pritisk na določeno tipko, pošiljanje abecede skozi RS-232 kanal itd.

A.P.Železnikar

****
PASCAL ZA MIKRORAČUNALNIKE
****

PASCAL/M za mikroračunalnike na operacijskem sistemu CP/M 2.2 je pravkar prišel v prodajo za ceno 175 dolarjev. Ta PASCAL obsega celotno Wirthovo realizacijo in potrebuje 56K pomnilnik tipa RAM. Značilnost te različice jezika PASCAL je v tem, da ga je mogoče posebej modificirati glede na uporabljane mikroprocesorje. Čeprav je CP/M pisan za procesor 8080A in s tem tudi za Z80, je mogoče kompilator posebej modificirati za procesor 8080A ali Z80 ali Z80 in aritmetični procesor 9511. To modifikacijo opravi globalno že proizvajalec, uporabnik pa vstavi še svoje konkretne pogoje. Naslov proizvajalca je: Sorcim, 2273 Calle De Luna, Santa Clara, CA 95050.

PASCAL/M ima tele osnovne lastnosti: zaznava napake, teče na operacijskih sistemih CP/M 1.4 in 2.2, njegove razširitve so usklajene z drugimi znanimi različicami pascalskih kompilatorjev, uporablja naključne zbirke in daljša cela števila (32 bitov, 9 številk), V/I je v celoti prilagojen na zgradbo zbirk v CP/M, uporablja lahko prednosti procesorjev Z80 in 9511 itn. Proizvajalec dobavlja tudi samo priročnik na 90 straneh in Jensenovo in Wirthovo knjigo (10 + 7.90 dolarjev).

A.P.Železnikar

MIKRORAČUNALNIŠKI RAZVOJNI SISTEM

Podjetje Tektronix je dalo v prodajo univerzalni mikroprocesorski razvojni sistem MDA 8002A.Osnovni sistem vsebuje naslednje enote:Procesor sistema(2650),modul za čiščenje,pomnilnik sistema(32-K zlogov), modul sistem-komunikacija,disketno enoto z dvema pogonoma in procesor za zbirnik (Z-80).

Za opremo razvojnega mesta so na voljo naslednje možnosti:

- pogonski sistem za izbran mikroprocesor z urejevalnikom,zbirnik(Macro-Assembler),časovnik,komunikacijska programska oprema,programi za PROM programator,čistilnik(debug)in prototipni analizator v realnem času.
- emulatorski modul za izbrani mikro procesor za izvajanje uporabniških programov.
- prototipni krmilnik(Control Probe) za izbrani mikroprocesor.

Na voljo so še naslednje dopolnitve:

- programska razširitev pomnilnika na 64 K zlogov(kot 16K ali 32K modul).
- PROM programator(za 1702A,2704/08).
- prototipni analizator v realnem času(adrese podatki,krmilna vodila in 8 perifernih signalov).
- sistemski terminal CT 8100(Video) in CT 8101(Pisalni terminal) ter 4024/4025 računalniški terminal.
- vrstični pisalnik LP 8200.

Podjetje Tektronix omogoča interesentom eno tedenske tečaje za spoznavanje sistema s praktičnimi vajami.

R.M.

NOVO INTEGRIRANO VEZJE ZA DETEKCIJO NAPAK

V zadnjem času so proizvajalci polprevodniških elementov obogatili tržišče z nekaterimi novimi enotami za posebne namene.Tako je podjetje American Micro Devices Inc. izdelalo integrirano vezje za detekcijo in korekcijo napak,zlasti primerno za diagnostiko diskovnih pomnilnikov. Uporabno pa je tudi za različne druge namene.

Nov N MOS AmZ 8065 procesor lahko detektira in popravlja 12-bitov dolge skupinske napake v serijskih nizih podatkov s pogostostjo prenosa do 20 miljonov bitov na sek. Detekcijski postopek je osnovan na deljenju z izbranim generatorskim polinomom,kot pri znani CRC metodi.Preizkuševalni kod dolžine 32 bitov lahko detektira 11 bitno skupino (burst)napak v 42987 bitnem sektorju diska. 56 bitni kod lahko detektira in popravi 11 bitno skupino napak v 585442 bitnem sektorju.

Čipi bodo dosegljivi na tržišču tekom letošnje jeseni.Čip AmZ 8065 je prilagojen za delo z mikro računalnikom Z 8000(16 bit.), dočim je splošno uporabljen čip z oznako Am 9520.

R.M.

NOV NEC-OV MIKRORAČUNALNIK

Znano japonsko podjetje NEC je izdelalo nov 8-bitni mikro računalnik v enem čipu z oznako uCOM-87.Računalnik združuje 4K zlogovni ROM pomnilnik,128 zlogovni RAM pomnilnik,časovnik,serijski vmesnik,vhodno-izhodna vrata itd.v enem čipu.Omogoča pa tudi neposredno naslovlanje zunanjih pomnilnikov do 64K zlogov.Izredno hiter serijski vmesnik omogoča enostavno multiprocesorsko uporabo ob podpori 140 instrukcij.Računalnik vsebuje dvojni akumulator in podvojeni sestav splošnih registrov,kar omogoča desetkrat hitrejšo prekinitveno opravilo.

Povečana pomnilniška zmogljivost in sistemska razširljivost omogočata uporabo računalniškega sistema za najrazličnejša poslovna opravila ter v procesni tehniki.V ta namen planira podjetje NEC širok spekter programske opreme.(Npr. PDA-80/800 in INTELLEC SERIES-II).Omenjeni mikro računalnik po zmoglivosti daleč presega že znane 8-bitne mikro računalnike istega podjetja,kot so: uCOM-80,uCOM-85,uCOM-82 in uCOM-84.

Predvidevajo,da bo nov proizvod uCOM-87 (uPD 7801G)na tržišču že v letošnjem maju.

R.M.

SREČANJA

2- 4 julij, Aberdeen, Velika Britanija

International Conference on Databases

Organizator: University of Aberdeen, British Computer Society
Informacije: S.M. Deen, Dept. of Computer Science, University of Aberdeen, Aberdeen A B9 2UB, Scotland

7-11 julij, Stanford, ZDA

Tutorial Conference on Practical Optimization

Organizator: Stanford UniversitySystems Optimization Laboratory
Informacije: Philip E. Gill, Systems Optimization Laboratory Dept. of Operations Research, Stanford Univrsity, Stanford, CA 94305, USA

8-11 julij, Les Arc, Savoie, Francija

5th Conference on Automated Deduction

Organizator: Institut de Recherche D'Informatique et D'Automatique
Informacije: IRIA, SErvice des Relations Exterieures Domaine de Voluceau, 78150 le Chesnay , France

14-16 julij, Madison, Wis., ZDA

Nonlinear Programming Symposium

Organizator: Mathematical Programming Society
Informacije: Nonlinear Programming Symposium 4, Computer Science Dept. University of Wisconin, 1210 W. Dayton st. Madison, WI 53706 (608 262-1204, USA

14-16 julij, Budimešta, Madžarska

Logic Programming Workshop

Organizator: von Neuman Computer Science Society
Informacija: Sten-Ake Tarnlund, Dept. of Computer Science University of Stockholm, 106 91, Stockholm, Sweden

14-18 julij, Seattle, ZDA

SIGGRAPH 80, Seventh Annual Conference on Computer Graphics and Interactive Techniques,

Organizator: ACM, SIGGRAPH
Informacije: SIGGRAPH 80, Box 88203, Seattle, WA 98 188, ZDA

14-18 julij, Noordwijkerhout, Nizozemska

Seventh International Colloquium on Automata, Languages, and Programming

Organizator : European Association for Theoretical Computer Science
Informacije: Jan van Leeuwen, Dept. of CS, University of Utrecht, Box 80012, 3508 TA Utrecht, The Netherlands

21-23 julij, Durban, Južnoafriška Republika

Sixth South African Symposium on Numerical Mathematics

Organizator: University of Natal
Informacije: Chairman, Computer Science Dept., University of Natal, King George Avenue, Durban 4001, Republic of South Africa

17-21 avgust, Toronto, Kanada

1980 Urban and Regional Information Systems Association Conference

Organizator: URISA
Informacije: URISA, 180 North Michigan Ave., Suite 800, Chicago, Il 60601, USA

18-22 avgust, Edinburgh, Velika Britanija

4th Symposium on Computational Statistics

Organizator: International Association for Statistical Computing
Informacije: COMPSTAT 1980, Director, Program Library Unit, Edinburgh University, 18 Buccleuch Place, Edinburgh EH8 9LN, Scotland

18-20 avgust, Montreal, Kanada

3rd IFAC Symposium on Automation in Mining Mineral and Metal Processing

Organizator: IFAC
Informacije: IFIP Sec., 3 rue du Marche, CH-1204 Geneva, Switzerland

19-21 avgust, Palo Alto, ZDA

National Artificial Intelligence Conference

Organizator: American Association for Artificial Intelligence
Informacije: J. Marty Tenenbaum, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025, USA

20-22 avgust, Northridge, California, ZDA

8th SIMULA Users Conference

Organizator: Association of SIMULA Users,
Informacije: Eileen Schreiner, Norwegian Computing Center, P.O.B 335, Blindern Oslo 3, Norway

24-27 avgust, Stanford, ZDA

1980 LISP Conference

Organizator: Stanford University
Informacije: John Allen, Stanford Artificial Intelligence Laboratory, Stanford University, CA 94305

24-29 avgust, San Francisco, ZDA

Symposium on Supercomputers in Chemistry

Organizator: ACS, Division on Computers in Chemistry
Informacije: Peter Lykos, Illinois Institute of Technology, Chicago, IL 60616

24-31 avgust, Poznan, Poljska

Data Processing in Chemistry 80

Organizator: Committee of Chemical Sciences, Polish Academy of Sciences
Informacije: I. Lukasiewicz Technical University, Instute of Chemical Technology, 35-959, Poznan, P.O. Box 85, Poland

25-27 avgust, Seattle, ZDA

1980 Summer Computer Simulation Conference

Organizator: SCS, AMS, ISA, AGU, AIAA, BMES, IMACS, AIChE, IEEE
Informacije: David R.S. McCall, Boeing Aerospace Co., Mail Stop 84-16, P.O. Box 3999, Seattle, WA 98124, USA

26-29 avgust, Boyne Highlands Convetion Center, ZDA

1980 International Conference on Parallel Processing

Organizator: Wright State University, IEEE-CS
Informacije: Kay Garringer, Dept. of Computer Science, Wright State University, Dayton, OH 45435 USA

27-28 avgust, Asilomar, California, ZDA

Second Workshop on Picture Data Description and Management

Organizator: IEEE-CS
Informacije: Harry Hayman, Box 639, Silver Spring, MD 20901, USA

1-5 september, Rydzyna, Poljska

9th International Symposium on Mathematical Foundation of Computer Science

Organizator: Polish Academy of Science, Institute of Computer Science
Informacije: Jan Maluszynski, Institute of Computer Science Polish Academy of Sciences, Box 22, 00-901 Warsaw PKiN, Poland

2-5 september, Lisbon, Portugal

Modelling and Simulation of Enviromental Systems with Inadequate Data

Organizator: IFIP WG 7.1
Informacije: IFIP Sec., 3 rue du Marche, CH-1204 Geneva, Switzerland

3-5 september, Geneva, Švica

Eurographics 80

OrganizatorEuropean Association for Computer Graphics
InformacijeEurographics, Christian Pellegrini, Universite de Geneve, Centre Universitaire d' Informatique, 24 rue General Dufour, 1211 Geneva 4, Switzerland

3-5 september, Los Angeles, ZDA

19th Annual Lake Arrowhead Workshop on Office Information Systems

Organizator: IEEE-CS
Informacije: Clarence Ellis and Gary Nutt, Xerox PARC, 3333 Coyote Hill Road, Palo Alto, CA 94304, USA

4-8 september, Dubrovnik, Jugoslavija

1st WC on Global Modelling

Organizator: IFIP, TC WG.7.1
Informacije: IFIP SEc., rue du Marche, CH-1204 Geneva, Switzerland

8-12 september, Washington, ZDA

Integrated Systems 80

Organizator: National Micrographics Association
Informacije: National Micrographics Association, 8719 Colesville Rd., Silver Spring, MD 20910, USA

9-10 september, Montreal, Kanada

First IASTED International Symposium and Exhibition on Office Automation

Organizator:Canadian Section of IASTED
Informacije:The Secretary, Office Automation, Box 25, Station G, Calgary Alberta, Canada T2N 1N4

9-10 september,Montreal, Kanada

13th International Symposium and Exhibition on Mini and Micro Computers Applications

Organizator:ISMM
Informacije:M.H. Hamza, Dept. of EE,University of Calgary Calgary, Alberta, Canada T2N 1N4

9-12 september, Bologna , Italija

Europhysics Conference on Computing in High Energy and Nuclear Physics

Organizator:European Physical Society Italian National Institute of Nuclear Physics, University of Bologna
Informacije:Frederic James, EPS/CPG Bologna Conference, C.E.R.N., Division DD, CH 1211 Geneva ,Switzerland

15-20 september, Budimpešta, Madžarska

WC on Stochastic Differential Systems

Organizator: IFIP WG 7.1, TC 7
Informacije:IFIP SEc., rue du Marche 3, CH 12-40, Geneva Switzerland

16-18 september ,London, Velika Britanija

EUROMICRO 80, 6th Symposium on Microprocessing and Microprogramming

Organizator:European Association for Microprocessing and Microprogramming
Informacije:Lionel Thompson, H.S.D.E.,Hatfield AL 109 LP, England

17-19 september, Palo Alto, ZDA

Joint ACM SIGSMALL/SIGPC Symposium on Small Systems

Organizator: ACM SIGSMALL,SIGSP
Informacije:Philippe Lehot, Ford Aerospace ,3939 Fabian Way, Palo Alto, CA 94303, USA

17-19 september, Denver , ZDA

Integrated Programs for Aerospace Vehicle Design National Symposium

Organizator:NASA
Informacije:IPAD Project Office, Mail Stop246, NASA Langley Res.Ctr.,Hampton,VA 23665, USA

22-25 september, Philadelphia, ZDA

12th Annual Conference of the Society for Managemrnt Information Systems

Organizator:SMIS
Informacije:SMIS Headquarters, 111 East Wacker Drive, Chicago, IL 60601, USA

22-26 september, Washington, ZDA

COMPCON FALL 80

Organizator:IEEE-CS
Informacije:Compcon Fall 80, Box 639, Silver Spring, MD 20901, USA

24-27 september, San Diego, ZDA

10th Annual Conference of the Society for Computer Medicine

Organizator:SCM
Informacije:Society for Computer Medicine,19101 N.Ft.Myer Drive, Suite 602, Arlington, VA 22209

24-27 september, Paris, Francija

Impact of CAD in Small and Medium Sized Industries

Organizator:IFIP TC 5
Informacije: IFIP SEc., 3 rue du Maeche,CH 1204, Geneva Switzerland

29 september -4 oktober

International Conference on Computational Linguistics

Organizator:International Committee for Computational Linguistics
Informacije:DAvid Hayes,5048 Lake Shore Road, Hamburg, NY 1475, USA

29 september - 4 oktober, Tokyo, Japonska

MEDINFO 80
Organizator:IFIP,Medical Information Systems Development Center, Japan Society of Medical Electronics and

Biological Engineering, Kansai Institute of Information Systems
Informacije: IFIP Foundation 40, Paulus Potterstraat, 1071 Amsterdam, The Netherlands

30 september - 2 oktober, Saarbrücken, ZR Nemčija

Annual Conference 80

Organizator: Gesellschaft für Informatik
Informacije: R. Wilhelm, FB 10 - Informatik, Universität des Saarlandes, D-6600, Saarbrücken 11, West Germany

1-3 oktober, Lake Placide, ZDA

Twenty First Annual Symposium on Foundations of Computer Science

Organizator: IEEE-CS
Informacije: Andrew Yao, Computer Science Dept., Stanford University, Stanford CA 94305, USA

1-3 oktober, Montreal, Kanada

Sixth International Conference on Very Large Data Bases

Organizator: ACM, SIGBDP, SIGIR, IEE-CS
Informacije: William Armstrong, Dept. d'Informatique et de R.O. Université de Montreal, C.P. 6128, Succursale A, Montreal, Quebec, Canada H3C 3J7

1-3 oktober, Kyoto, Japonska

Tenth International Symposium on Fault Tolerant Computing

Organizator: IEEE-CS, Tech. Comm. on Fault-Tolerant Computing of Japan Tech. Groups on Electronic Computers and on Reliability
Informacije: Shuzo Yajima, Dept. on Information Science, Kyoto University, Kyoto 606, Japan

2 -4 oktober, Tokyo, Japonska

Working Conference on Man-Machine Comunication in CAD/CAM

Organizator: IFIP WG 5.2, WG 5.3
Informacije: T. Sata, Dept. of Precision Machine Eng. Faculty of Eng., University of Tokyo, 4 7-3 Hongo, Bunkyo-ku, Tokyo 113, Japan

4-6 oktober, Tokyo, Japonska

International Conference on Automated Multiphasic Health Testing and Services

Organizator: IHEA, Japan Society of AMHTS
Informacije: Secretariat of ICAMHTS, c/o P.L. Medical Data Center, P.O. Box-1, Tondabayashi, Osaka 58400, Japan

5-10 oktober, Anaheim, ZDA

Annual Meeting of the American Society for Information Science

Organizator: ASIS
Informacije: Alan Benenfeld, University of California, 8251 Boelter Hall, Los Angeles, CA 90024, USA

6-9 oktober, Tokyo, Japonska, 14-17 oktober, Melbourne, Avstralija

IFIP Congress 80

Organizator: IFIP, Information Processing Society of Japan, Australian Computer Society
Informacije: 8th World Computer Congress, c/o AFIPS, 1815 N. Lynn St., Suite 805, Arlington, VA 22209, USA

7-10 oktober, Berlin, ZR Nemčija

International Congress for Data Processing

Organizator: Gesellschaft für Informatik, ACME, Conference on European Computer Users Association
Informacije: Robert Parslow, 6 Ormond Ave., Hampton, Middlesex, England

8-10 oktober, Cambridge, ZDA

1980 International Conference on Cybernetics and Society

Organizator: IEEE Systems, Man and Cybernetics Society
Informacije: Richard Vidale, Dept. of Systems and Computer Engineering, Boston University, Boston, MA 02215, USA

8-10 oktober, Allerton House, ZDA

18th Annual Allerton Conference on Communication, Control and Computing

Organizator: University of Illinois at Urbana Champaign
Informacije: P.V. Sarmate, Coordinate Science Laboratory University of Illinois, Urbana, IL 61801

8-11 oktober, Kyoto, Japonska

Seventh International CODATA Conference

Organizator: International Council of Science Unions Comm. on Data for Science and Technology
Informacije: CODATA Secretariat, 51 Boulevard de Montmorency, 75016 Paris, France

15-17,oktober, Toyohashi, Japonska

Fifth International Conference on Computers in Chemical Research and Education

Organizator:International Council of Scientific Unions Comm. on Data for Science and Technology
Informacije:S.Sasaki, School of Material Science, Toyohashi University of Technology, Tempaku,Toyohahi, Japan

20 oktober,Washington, ZDA

National Information Systems

Organizator:Computer Law Association
Informacije:Michel Yourshaw,1776 K St., N.W., Washington,DC 20006,USA

21-24,oktober,Jakarta, Indonezija

SEARCC 80

Organizator:South East Asia Regional Computer Society
Informacije:SEARCC 80 Conference Implementation Committee, P.O.Box 4487, Jakarta, Indonesia

Razstave in Sejmi

22-28,avgust, Düsseldorf, ZRNemčija

International HiFi-Ausstellung Festival

27 avgust - 1 september, Zürich, Švica

Schweinerische Fernseh, Radio und HiFi Ausstellung

29 avgust -7 september, Amsterdam, Holandija

FIRATO -Internationale Funkausstellung

31 avgust -3 september, Frankfurt, ZR Nemčija

Mednarodni Frankfurtski Sejem

31 avgust -7 september, Leipzig, DR Nemčija

Mednarodni sejem elektronike

4-9 september ,Mailand, Italija

Mednarodni sejem HiFi opreme

6 -14 september,Dunaj, Avstrija

Mednarodni Dunajski Sejem za Elektroniko in HiFi

7-9 september, Basel, Švica

INTERFEREX

10-14 september, Stuttgard,ZR Nemčija

Razstava opreme za HobbyElektroniko im Mikroračunalnike

15-19 september,Essen, ZR Nemčija

Četrti mednarodni Sejem, Security 80

16-19, september, Anaheim, California, ZDA

Western Electronic Show and Convention

16-19 september, Helsinki, Finska

Mednarodni Tehnični Sejem

17-26 september,Paris, Francija

SICOB,Mednarodni sejem za opremo za obdelavo podatkov in pisarniške opreme

20-24 september, London, Velika Britanija

Internationale Broadcasasting Convetion

26-september -5 oktober, Budimpešta, Madžarska

Mednarodni tehnični sejem

27 september-5 oktober, Graz, Avstrija

Graški velesejem

27 september -5 oktober, Barcelona, Španija

Mednarodni salon Elektronike

Seminarji

23-25 julij,München, ZR Nemčija

Bit-Slice Seminar firme AMD

23-27 julij, Cannes,Francija

Programiranje Alf Z 8000 s Pascalom in Assemblerjem

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: Zasnova interaktivne računalniške grafike

Projekt: Računalniško projektiranje

Nosilec naloge: Milan Kac, Visoka tehniška šola, Maribor

Program raziskave:
- Študij realiziranih modelov interaktivne računalniške grafike pri nas in na tujem.
- Študij obstoječih standardov s področja računalniške grafike.
- Primerjava grafične periferije raznih proizvajalcev in primerjava računalniških sistemov glede na interaktivno grafiko.
- Organizacija in porazdelitev obdelav med centralnim in grafičnim procesorjem.
- Zasnova prikazovalnih datotek.
- Vprašanja implementacije pogonskih programov za grafično periferijo.

Naslov naloge: Računalniško omrežje.

Projekt: Informacijski sistemi.

Nosilec naloge: Tomaž Kalin, Institut J.Stefan, Ljubljana

Program raziskave:

- Zasledovanje razvoja na področju omrežij za preklapljanje paketov in s tem povezavo dela v strokovnih telesih projekta COST II bis.
- Nadaljevanje dela na vmesnem mrežnem jeziku z vključitvijo interaktivnega načina dela.
- Nadaljevanje dela na projektni študiji za načrtovanje hierarhije protokolov v meteorološko-ekološki mreži s poudarkom na radijsko paketno mrežo.
- Študij zasnove lokalnih mrež in njihove povezave preko javnih omrežij.

Naslov naloge: Prenos vernih in gospodarnih informacij IV.

Projekt: Informacijski sistemi.

Nosilec naloge: Bogomir Horvat, Visoka tehniška šola, Maribor.

Program raziskave:

- Naloga predstavlja četrto fazo raziskave. V tej fazi bi fizikalno realizirali predprocesorski komunikacijski modulski sestav. Ta modulski sestav bi bil na eni strani povezan s telekomunikacijskim kanalom, na drugih izhodih bi pa imeli možnost dostopa do vseh perifernih enot mikroračunalnika.
- Zmogljivost, zanesljivost in funkcionalnost preveriti ob realizaciji.

Naslov naloge: Večnivojski sistemi upravljanja z mini in mikro računalniki.

Projekt: Računalniška avtomatizacija industrijskih procesov.

Nosilec naloge: Jurij Tasič, Institut J.Stefan, Ljubljana.

Program raziskave:

- Teoretski aspekti večnivojskega upravljanja s stališča optimizacije velikih sistemov (metoda koordinacije ciljev z upoštevanjem omejitev, hierarhična povratnozančna regulacija).
- Programska oprema za omenjeni metodi.
- Metoda fleksibilnih poliedrov in metoda diagonalnih gradientov ter njena uporaba pri upravljanju.
- Programska oprema za omenjeni metodi.
- Računalniške komunikacije pri sistemu DARTA 80 za predpisane standarde (materialna in programska oprema).
- Mikroračunalniška hitra aritmetična enota (materialna in programska oprema) namenjena omenjenemu procesnemu mikroračunalniku.
- Mikroračunalniški regulator z upoštevanjem optimalnosti (Bellman).

Naslov naloge: Razpoznavanje funkcij z računalniško analizo.

Projekt: Informacijski sistemi-individualne naloge.

Nosilec naloge: Marjan Ribarič, Inst.J.Stefan, Ljubljana.

Program raziskave:

- Teoretično izpopolnjevanje analitičnega orodja za raziskavo eksperimentalno dobljenih funkcijskih odvisnosti s pomočjo diferencialnih enačb.
- Izdelava novih metod na podlagi teh teoretičnih spoznanj.
- Vključitev novih metod v obstoječi programski paket (sistemska analiza, oblikovanje dodatne programske strukture. izdelava programov, testiranje, dokumentiranje, arhiviranje).

Vsporedno s tem bomo uresničevali še naslednje naloge:
- Nadaljne proučevanje metod za kvalitativno analizo funkcijskih odvisnosti eksperimentalno dobljenih podatkov s pomočjo dosegljive strokovne literature in s stiki z drugimi skupinami doma in v tujini, ki se ukvarjajo s podobnimi problemi.
- Razvoj novih metod in njihovo preizkušanje v primerih iz prakse.
- Vzdrževanje programov in konsultacije pri njihovi uporabi in razširjanje programskega paketa na osnovi stikov z uporabniki.
- Razširjanje dokumentacije in priročnikov.
- Poročanje na domačih in tujih kongresih in seminarjih.

AVTORJI IN SODELAVCI

Hodžić Migdat(1952). Elektrotehnički fakultet završio u Banja Luci,na odsjeku za automatiku i računarsku tehniku 1975 godine.Od tada radi u preduzeću "Rudi Čajavec" OOUR Radarsko-računarska tehnika.Problematika kojom se bavi je iz domena optimalnog upravljanja deterministi-čkim i stohastičkim sistemima,te iz domena real-time implementacije odgovarajućih algoritama.Na Elektro-tehničkom fakultetu u Beogradu,odsjek Upravljanje siste-mima,dovršava magistarsku tezu na temu implementacije u realnom vremenu filtara za praćenje ciljeva koji mane-vrišu.

Dušan M.Velašević (1939).Diplomirao je na Elekrotehničkom fakultetu u Beogradu 1963 godine.Od 1963 do 1974 godine radio je u Laboratoriji za elektroniku Instituta za nuklearne nauke "Boris Kidrič"u Vinči.Od 1974 godine radi na Elektrotehničkom fakultetu u Beogradu kao docent u oblasti računarske tehnike i informatike.Trenutno,njegov istraživački interes uključuje programske prevodioce, operativne sisteme,zaštitu podataka i sisteme direktne obrade.

Velimir Otović .Diplomirao 1975 godine na George Washington University-Dept.of Electrical Engineering and Computer Science,Washington D.C.,SAD.Trenutno radi u "Rudi Čajavec-u" u OOUR Računarsko radarske tehnike,na poslovima sistem analize i simulacije.Oblasti koje ga interesuju su operativni sistemi za multiprocesorske ra-čunare i digitalne simulacije.Radi magistarski rad iz ob-lasti simulacije računara.

D.B.Popovski (1951). Diplomirao g.1975 i magistrirao g.1977 na termotehničkoj grupi Mašinskog fakulteta u Beogradu.Asistent je Tehničkog fakulteta u Bitolju.Bavi se numeričkom analizom i primenom numeričkih metoda u toplotnoj tehnici.Ima objavljeno više radova u domačim i stranim časopisima iz pomenutih oblasti.

Davor Miljan (1955).Diplomiral je leta 1978 na Fakulteti za Elektrotehniko v Ljubljani,smer računalništvo in informatika.Po diplomi se je zaposlil na odseku za računal-ništvo in informatiko na Institutu Jožef Stefan v Ljubljani. Ukvarja se z načrtovanjem digitalnih vezij in mikroraču-nalniškimi aplikacijami.

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 total

DELTA/TOTAL je vsestransko uporaben upravljalni sistem podatkovnih baz, katerega lahko nudimo uporabnikom računalniških sistemov DELTA. Sistem DELTA/TOTAL je učinkovit, preprost za uporabo in preizkušen programski proizvod.

Pojem baza podatkov ni nov. Baze podatkov so v naših delovnih organizacijah obstajale že od prvih začetkov dalje, in to v kartotekah in imenskih seznamih.
Z uvajanjem računalniške opreme so te baze podatkov prenesli v datoteke podatkov.
Ti elektronsko shranjeni podatki so bili ponavadi urejeni in vzdrževani v okviru funkcionalnih in oddelčnih možnosti, dosegljivi pa so bili na osnovi med seboj neodvisnih aplikacij. Kot rezultat takega pristopa so se pojavljali trije različni problemi:
— nepovezanost podatkov
— preveč odvečnih podatkov
— nepopolnost podatkov
Tako zasnovani sistemi niso bili primerni za spremembe, bodisi neznatne ali večje.
V večini primerov je cena spremembe preprečevala kakršnokoli spremembo sploh.
Vse to pa je danes preteklost.

DELTA/TOTAL je resnični upravljalni sistem podatkovnih baz. Lahko upravlja neomejen izbor podatkov na integrirani in enoviti osnovi, ki omogoča dostop in zvezo z vsemi ostalimi podatki v bazi podatkov. To poveča vrednost osnovnih podatkov, ki pri nadaljnjem delu zagotavljajo kvalitetne informacije. TOTAL je upravljalni sistem podatkovnih baz, katerega uporaba je v svetu najbolj razširjena. Dejstvo je, da je več uporabnikov sistema TOTAL, kot pa uporabnikov katerihkoli drugih baz podatkov skupaj. Sedaj je dosegljiv tudi uporabnikom računalniških sistemov DELTA. Naš cilj je, da zagotovimo programsko pomoč in ustrezno programsko opremo najvišje možne kvalitete. Zavedamo se namreč, kaj izražamo z mislijo: USTVARJAMO UČINKOVITOST!

UČINKOVITOST SISTEMA DELTA/TOTAL

OPTIMALNO DELOVANJE IN UČINKOVITOST sta za uporabnike računalniških sistemov DELTA največji prednosti pri uporabi sistema DELTA/TOTAL. Tradicionalno sta si ta dva, za uspeh vsakega sistema življenjsko važna faktorja, nasprotovala drug drugemu. Pri sistemu DELTA/TOTAL temu ni tako.

MODULARNOST IN PRILAGODLJIVOST V NAČRTOVANJU IN UPORABI sta prednosti sistema DELTA/TOTAL v zvezi z znižanjem stroškov sprememb, bodisi večjih ali neznatnih. Tako je rešen problem zastojev z analizo, ki je očitna v načrtovanju sistemov.

NEODVISNOST PODATKOV — aplikacijski programi operirajo le s podatkovnimi elementi (polji), ne pa z zapisi. To preprečuje ponovno prevajanje starih programov zaradi fizičnih sprememb zapisov, datotek, povezav, itd.

ENKRATNOST PODATKOV — onemogoča tako imenovane številne inačice resnice. Za vzdrževanje baze podatkov potrebujemo zato manj diskovnega prostora. To občutno zmanjšuje stroške vzdrževanja in zmanjšuje možnosti netočnih podatkov.

POVEZANOST PODATKOV — omogoča bazi podatkov, da ima prav tako strukturo, kot jo ima način poslovanja delovne organizacije. To daje ustreznim informacijam realno osnovo.

CELOVITOST IN VARNOST PODATKOV — sta zelo važni za bazo podatkov. Baza podatkov postane tako ena najbolj dragocenih pridobitev OZD. Preprečuje uničenje datotek in zapisov z nedovoljenimi posegi aplikacijskih programov v bazo in iz baze podatkov.

OPREMA, PROGRAMSKI JEZIK IN OKOLJE SO NEODVISNI — to daje visoko stopnjo neodvisnosti na tako važnih področjih, kot so operacijski sistemi, računalniški sistemi, diskovne enote, uporabljeni programski jeziki in pristop k obdelavi, npr. paketni ali interaktivni. Zato je možnost različnih konverzij v bodočem razvoju avtomatske obdelave podatkov bistveno manjša.

VEČNAMENSKA ZMOGLJIVOST — eno kopijo sistema DELTA/TOTAL, ki je stalno v računalniku, lahko uporablja neomejeno število uporabnikovih programov, ki se izvajajo pod nadzorom operacijskega sistema.

TOTAL DATOTEKE

Za učinkovito vzdrževanje in uporabo veliko različnih kategorij podatkov, daje DELTA/TOTAL dve vrsti datotek: glavne datoteke (z enojnim vhodom) in variabilne datoteke (z različnim številom vhodov).

GLAVNE DATOTEKE so organizirane in vzdrževane z ozirom na uporabnikova prvotno izbrana kontrolna polja. Ta polja so dolžine 1 do 256 bytov, s kakršnokoli vsebino. V vsakem datotečnem zapisu je število podatkovnih elementov in velikost njihovih polj omejeno samo z velikostjo samega logičnega zapisa. Če je potrebno, je lahko vsak zapis v datoteki neposredno povezan z največ 2500 drugimi datotekami. Pri normalni uporabi se glavne datoteke sistema DELTA/TOTAL optimizirajo same in jih je treba le včasih preurediti. Če je zapis izbrisan, je prostor takoj dosegljiv za ponovno uporabo. Če želi uporabnik spremeniti razsežnost osnovnega zapisa, ali če so prekoračene fizične dimenzije, moramo samo prizadete podatke prepisati iz baze in nazaj v bazo podatkov. Vse ostale datoteke ne zahtevajo sprememb, preureditev ali dodatnih obdelav.

VARIABILNE DATOTEKE so lahko dostopne preko različnega števila kontrolnih polj. Vsebujejo različno (in morda tudi spremenljivo) število podatkovnih zapisov. Te datoteke in podatkovni zapisi imajo lahko edinstvene povezave, odvisno od uporabnikovih zahtev.

Variabilne datoteke sistema DELTA/TOTAL, ki odražajo poslovne funkcije tako, kot se te izvajajo, so vedno povezane z glavnimi datotekami.
V eni variabilni datoteki je lahko do 2500 različnih vrst zapisov. Zopet ni nobenih omejitev glede na velikost, število, vsebino ali medsebojni odnos podatkovnih elementov, kot le tiste, katere določa računalniški sistem Delta in njegov operacijski sistem.
Tako kot pri glavnih datotekah, je tudi tukaj upravljanje diskovnih površin optimizirano zato, da se zmanjša zahtevani pomnilni prostor in poveča učinkovitost delovanja.
Z uporabo pripomočkov, kot so jezik za definiranje baze podatkov in jezik upravljanja s podatki, zagotavlja DELTA/TOTAL v celoti integrirano bazo podatkov, ki je dosegljiva aplikativnemu programerju.

DEFINIRANJE BAZE PODATKOV

TOTAL jezik za definiranje baze podatkov se uporablja za opisovanje in določevanje sistema baze podatkov. Definicijo opravimo v posebnem modulu jezika za definiranje baze podatkov. Po definiranju baze podatkov je modul opisa baze podatkov preveden in katalogiziran. Programerje in analitike računalniških sistemov Delta lahko v kratkem času naučimo, da definirajo in razvijajo bazo podatkov določene težavnostne stopnje. Spremembe in dopolnitve, bodisi določenega podsistema baze podatkov ali celotne baze podatkov, so vprašanje le nekaj minut računalniškega časa.

ODPRTA NAROČILA IN SPREJETA VPLAČILA

VPLAČILA KUPCEV	GLAVNA DATOTEKA KUPCEV	NAROČILA KUPCEV

Primer uporabe TOTAL jezika za definiranje baze podatkov:

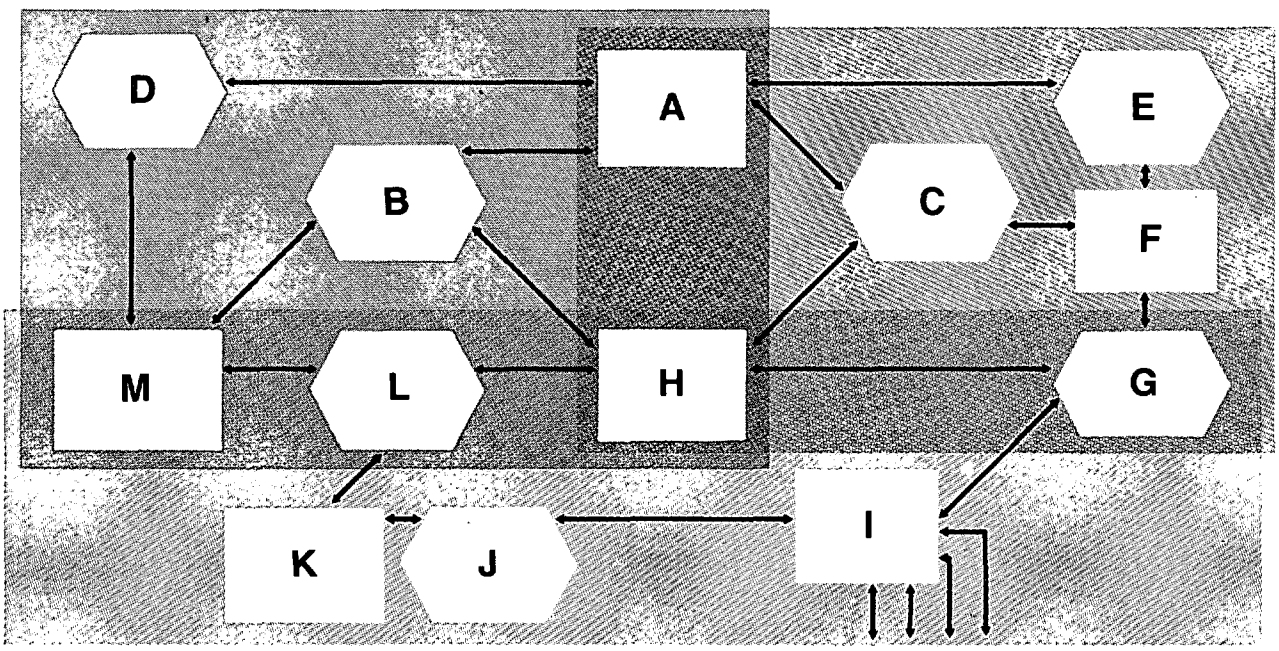
Glavna datoteka kupcev (KUPC) in vse druge datoteke, ki so vključene v to bazo podatkov, bodo opisane v TOTAL jeziku takole:

PODATKOVNI ELEMENT	IME	DOLŽINA
šifra kupca	KUPCCTRL	6
ime kupca	KUPCNAZI	30
naslov kupca	KUPCNASL	30
mesto	KUPCMEST	20
poštna številka	KUPCPOST	5
posebni kreditni pogoji	KUPCKRPO	6
drugi potrebni elementi		
kreditna omejitev kupca	KUPCKMEJ	4
sprejeta vplačila	KUPCVPLA	4
neplačani računi 15 dni	KUPCRA15	4
neplačani računi 60 dni	KUPCRA60	4

Podatkovne elemente, imena podatkov in dolžine nadzoruje uporabnik. Dolžine elementov, njihovo število in mešanice alfanumeričnih ali posebnih znakov, nimajo nobenih omejitev.

STRUKTURA TOTAL BAZE PODATKOV

Bistvenih prednosti upravljalnega sistema podatkovnih baz DELTA/TOTAL ni potrebno več poudarjati. To je sposobnost odstraniti kakršnokoli fizično odvisnost podatkov od aplikativnih programov.
S komuniciranjem na nivoju podatkovnega elementa (polje v logičnem zapisu), doseže DELTA/TOTAL največjo možno stopnjo podatkovne neodvisnosti, ki omogoča uporabnikom računalniških sistemov Delta visoko stopnjo modularnosti in prilagodljivosti ob rasti in spremembah. Ta prilagodljiv kocept sistema DELTA/TOTAL omogoča enostavno dodajanje novih podatkov v zapise, dodajanje novih datotek in povezav podatkov, ne da bi škodili že obstoječim sistemom. Prikazana baza podatkov bi se sčasoma lahko povečala in vsebovala stotine novih datotek in povezav, ob le majhnem posegu v delovne programe.



"N" povezav

Skica prikazuje, da se nekatere datoteke pojavljajo v več kot le v eni bazi podatkov in da se datoteka "H" pojavlja v vseh bazah podatkov. Vsi podatki, ki jih je katerikoli program ažuriral v datoteki "H", bodo avtomatično dosegljivi v vseh nadaljnjih obdelavah vsem programom, ki uporabljajo te skupne podatke. DELTA/TOTAL daje vedno le eno verzijo resnice !

JEZIK UPRAVLJANJA S PODATKI

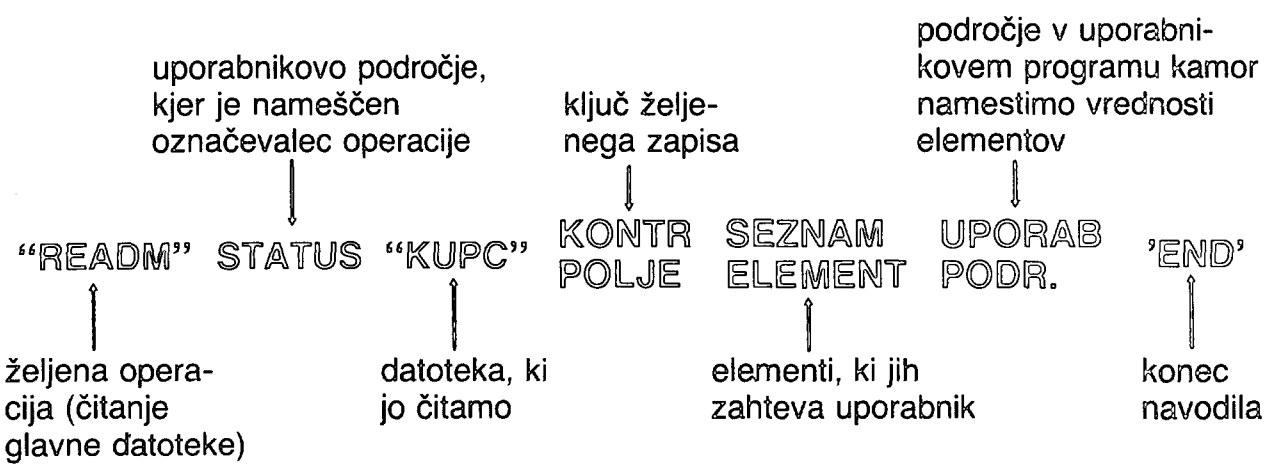
TOTAL jezik upravljanja s podatki uporablja programer za komunikacijo med svojim programom in bazo podatkov. To ni popolen jezik sam po sebi, ampak deluje v okviru programskega jezika gostitelja. Na nivoju klicnega ukaza CALL omogoča prenašanje podatkov v bazo in iz baze podatkov. Uporabljamo ga lahko v kombinaciji s programskimi jeziki kot so MACRO-11, FORTRAN IV, FORTRAN IV PLUS, BASIC PLUS 2 in COBOL.

Izredno učinkovit in razumljiv seznam klicnih ukazov CALL za upravljanje z bazo podatkov omogoča programerju, da z izjemno lahkoto poišče, dodaja ali briše podatke ali povezave med njimi.

STAVEK TOTAL JEZIKA ZA UPRAVLJANJE S PODATKI

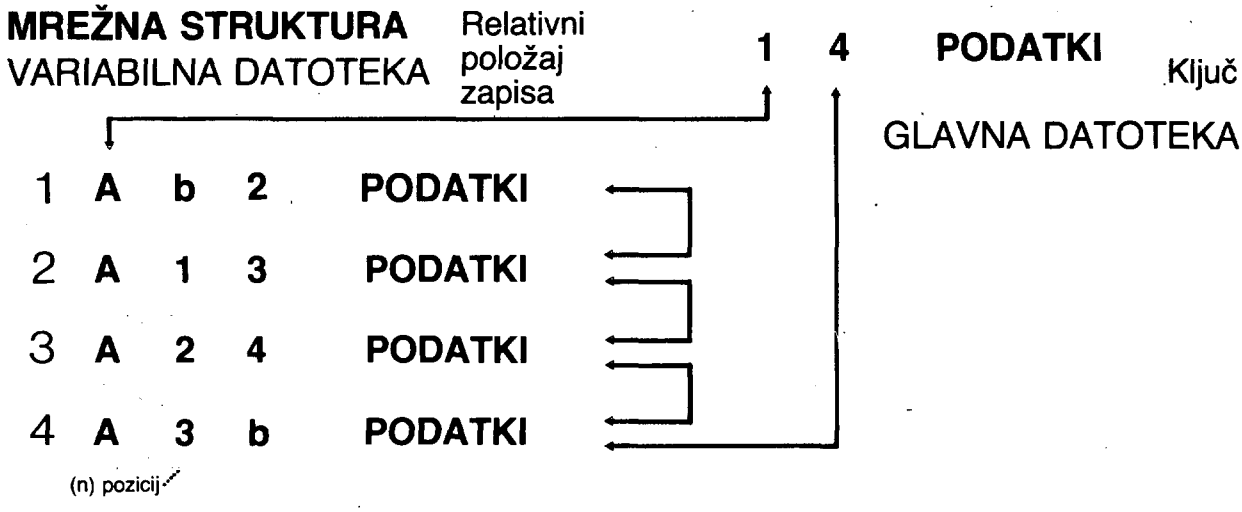
Sledeči seznami parametrov morajo biti dani stavku CALL upoštevajoč pravila programskega jezika. Parametri v narekovaju morajo biti predstavljeni klicnemu ukazu CALL kot literalne vrednosti, ostali parametri pa morajo biti definirani kot pomnilna področja, ki jih uporabljata uporabnikov program in DELTA/TOTAL.

CALL "DATBAS" USING



Po izvršitvi tega stavka so določeni elementi zapisa zahtevani za Kupca št. 123 (predpostavljamo, da je bil Kupec št. 123 želeni ključ v kontrolnem polju), nameščeni v uporabnikova pomnilna področja natančno tako, kot jih želi programer aplikacije. Ni mu potrebno izvajati nobenih izločevanj ali premikanj.

Področje STATUS bo vsebovalo****, kar označuje uspešno opravljen potek operacije. Kontrola se vrne v program in uporabnik lahko zatem izvaja katerokoli naslednjo operacijo želi.



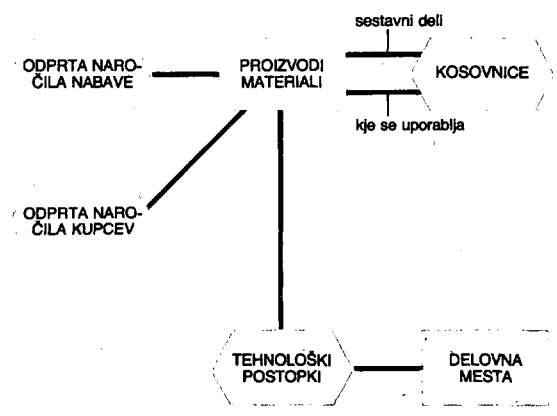
vse povezave so dvosmerne
povezave zapisov: brez indexa, tabel, itd.

FLEKSIBILNOST SISTEMA DELTA/TOTAL

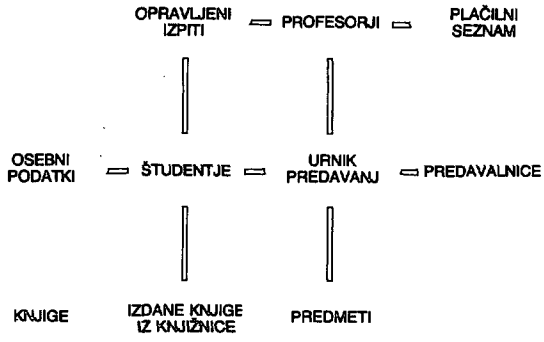
Uporabniki sistema TOTAL so vodilni v proizvodnji, izobraževanju, zavarovalništvu in bančništvu, če naštejemo le nekatere. Njihov skupni imenovalec je, da so uspešni. Predstavljajo široko razvejano področje različnih dejavnosti in so po vsem svetu. Odkrito smemo reči, da so vodilni tudi v tehnologiji upravljalno-informacijskih sistemov. Zanje ta upravljalno-informacijski sistem niso nemogoče sanje: to je resničnost, ki se jim obrestuje !

V svojih zahtevah DELTA/TOTAL ne obremenjuje preveč centralnega pomnilnika (približno 50K Byte, odvisno od števila in velikosti vključenih modulov), je preprost za uporabo, fleksibilen in ima visoko stopnjo učinkovitosti delovanja. Precej lahko poveča produktivnost instalacije, ne da bi se pri tem bistveno povečali režijski stroški.
DELTA/TOTAL upravljalni sistem podatkovnih baz je neodvisen od aplikacij, zato ga lahko vpeljemo na kakršnokoli področje poslovnega življenja. Sledeče sheme baz podatkov opisujejo zato le nekatere možne aplikacije sistema DELTA/TOTAL v poslovanju uporabnikov računalniških sistemov DELTA.

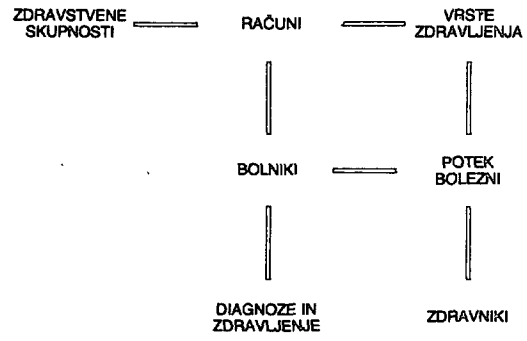
PROIZVODNJA



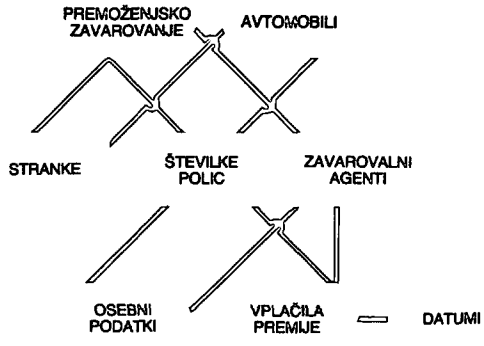
UNIVERZE



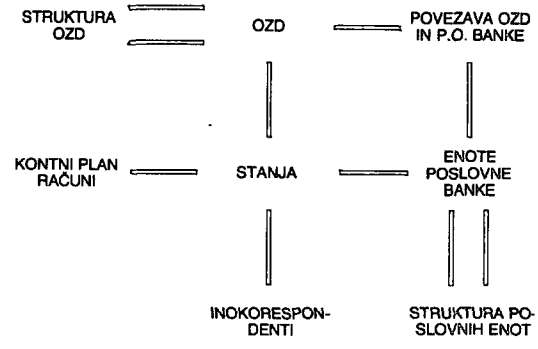
ZDRAVSTVO — BOLNICE



ZAVAROVALNIŠTVO



BANČNIŠTVO



delta računalniški sistemi

SOZD ELEKTROTEHNA, o. o., DO DELTA, proizvodnja računalniških sistemov in inženiring, p.o.
61000 Ljubljana, Linhartova 62a
Telefon: 061/323-585, 326-661
Telex: 31 578 YU ELDEC

— SLUŽBA ZA PROGRAMSKO OPREMO IN
IZOBRAŽEVALNI CENTER DELTA
TITOVA 51, 61000 LJUBLJANA, Telefon: 061/320-241, int. 482
— PRODAJNA SLUŽBA
TITOVA 51, 61000 LJUBLJANA
Telefon: 061/320-241, int. 397, 420
— SLUŽBA ZA RAZVOJ STROJNE OPREME
Telefon: 061/23-251, 21-874
— SLUŽBA ZA RAZVOJ PROGRAMSKE OPREME
Telefon: 061/28-216

POSLOVNA ENOTA ZAGREB

ZAGREBAČKI VELESAJAM, II. UPRAVNA ZGRADA
ALEJA BORISA KIDRIČA 2, 41000 ZAGREB
Telefon: 041/520-003, 516-690

POSLOVNA ENOTA BEOGRAD

— VZDRŽEVALNA SLUŽBA
KARAĐORĐEV TRG 13, 11080 ZEMUN
Telefon: 011/694-537, 695-604
— PRODAJNA SLUŽBA, »SAVA CENTAR«
MILENTIJE POPOVIĆA 9, 11070 NOVI BEOGRAD
Telefon: 011/453-885
— SLUŽBA ZA PROGRAMSKO OPREMO
»SAVA CENTAR«
Telefon: 011/356-591

PRODAJNA SLUŽBA,
SLUŽBA ZA
PROGRAMSKO OPREMO

SERVISNI CENTRI
LJUBLJANA, MARIBOR, CELJE, ZAGREB, RIJEKA,
SPLIT, SARAJEVO, BEOGRAD, SKOPJE