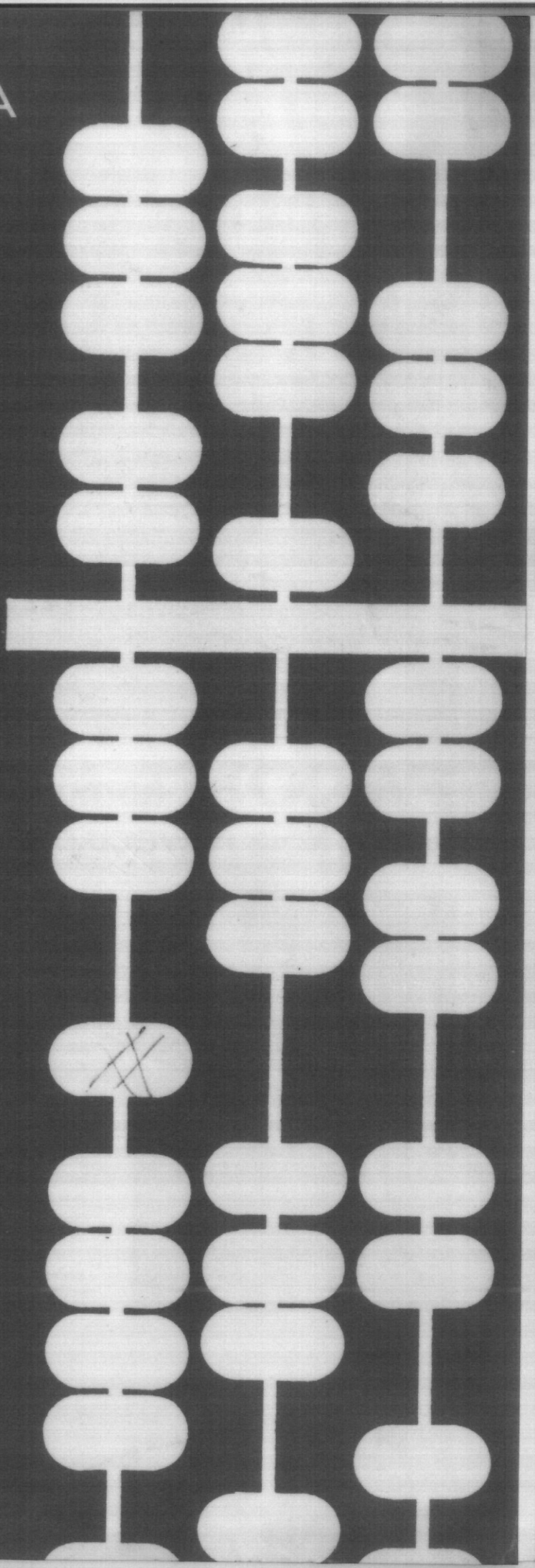
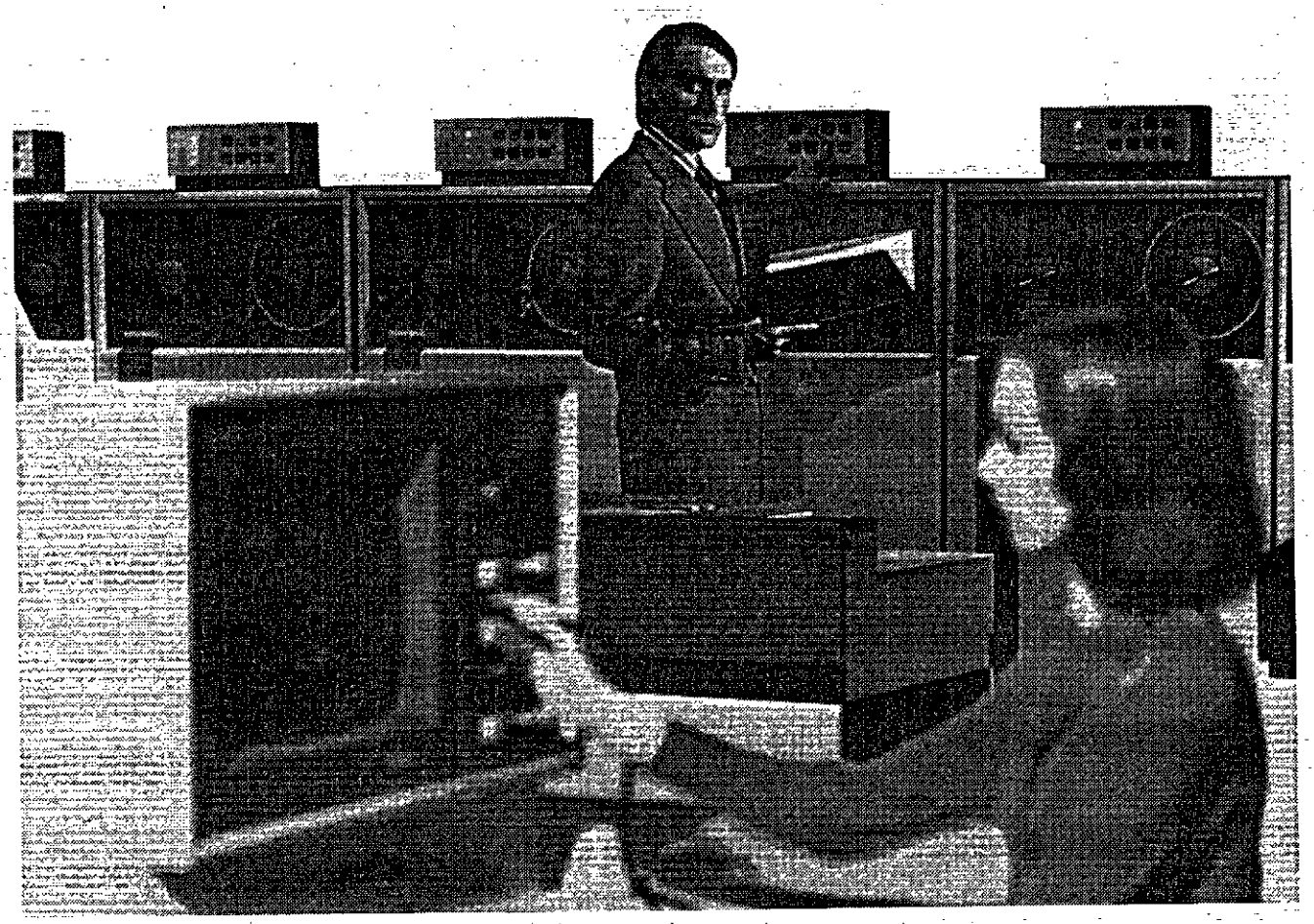
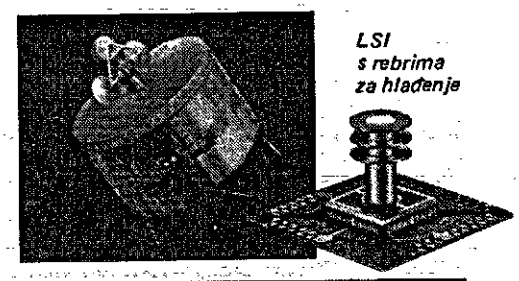
INFORMATICA



1978 2 YU ISSN 0350-5596



FACOM kompjutere proizvodi Fujitsu, tvrtka koja najveću pažnju posvećuje sistemima.



LSI s rebrima za hlađenje

Prije svega kompjuter je sistem, tj. sredstvo za obradu podataka koji u sebi sadrži hardware, software i aplikacionu tehnologiju. Naravno razne tvrtke bave se prodajom kompjutera. Ipak, malo je tvrtki koje mogu ponuditi potpuni izbor sredstava za automatsku obradu podataka — konstruirani tako, da osim optimalnih performanci, imaju mogućnost ugradnje u veće sisteme.

FUJITSU je jedna od tvrtki koja to može ponuditi. Kao vodeći proizvođač kompjuterskih sistema u Japanu, FUJITSU proizvodi široki asortiman proizvoda od minikompjutera s jednim LSI čipom do u svijetu najmoćnijih LSI sistema, kao i široki izbor periferne i terminalne opreme.

FACOM kompjuteri obavljaju važne aktivnosti u poslovnim i državno-administrativnim organizacijama u mnogim zemljama širom svijeta. U Japanu, drugom po redu najvećem tržištu kompjutera u svijetu, instalirano je najviše FACOM sistema u usporedbi s drugim modelima ostalih proizvođača. Ovi moćni, pouzdani FACOM kompjuteri sposobni su za obavljanje svih mogućih poslova. Oni upravljaju satelitima u svemiru, daju prikaz atmosferskih prilika real-time grafikonima u boji, obavljaju bankovno poslovanje pomoću on-line sistema za više od 7.000 filijala i ekspozitura i još mnogo, mnogo toga.

FACOM kompjuteri su potpuno integrirani sistemi gdje se kombinacijom visoko-kvalitetne tehnologije, moćnog softwarea i već provjerenih aplikacionih programa postiže efikasnost i pouzdanost kojima nema premca.

Za dalje informacije obratite se na:

zpr **Zavod za primjenu elektroničkih računala i ekonomski inženjering**

41000 ZAGREB Savska c. 56 Telefon: 518-706, 510-760 Telex: 21689 YU ZPR FJ

FUJITSU

Fujitsu Limited·Tokyo, Japan

INFORMATICA

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

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

Uredništvo in uprava: 61000 Ljubljana, Institut "Jožef Stefan", Jamova 39, telefon (061) 263 261, telegram: JOSTIN, telex: 31 269 YU JOSTIN.

Letna naročnina za delovne organizacije je 300,00 din, za posameznika 100,00 din, prodaja posamezne številke 50,00 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. 4210-7/78 z dne 19.1.1978, je časopis INFORMATICA strokovni časopis, ki je oproščen temeljnega davka od prometa proizvodov.

Tisk: Tiskarna KRESIJA, Ljubljana

Grafična oprema: Rasto Kirn

Letnik 2 , 1978 — številka 2

VSEBINA

INFORMATICA

Journal of Computing and Informatics

Volume 2 , 1978 — no. 2

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

Headquarters : 61000 Ljubljana, Institut "Jožef Stefan", Jamova 39, Phone: (061) 263261, Cable: JOSTIN Ljubljana, Telex: 31 269 YU JOSTIN

Annual subscription rate for abroad is US $ 18 for companies, and US $6 for individuals.

Printed by: Tiskarna Kresija, Ljubljana

DESIGN: Rasto Kirn

CONTENTS

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 dodatnih 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 300,00 din, za posameznike 100,00 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 approximate 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.

Časopis INFORMATICA
Uredništvo, Institut Jožef Stefan, Jamova 39, Ljubljana

Please enter my subscription to INFORMATICA and send me the bill.

Annual subscription price: companies 300,00 din (for abroad US $ 18), individuals 100,00 din (for abroad US $ 6)

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

.....

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 typewriter 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 blanc, 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.

računalniki v iskri

d.reš

Iskra-Elektromehanika
Kranj, Jugoslavija

UDK 681.3:061.5 ISKRA

V članku so na kratko pojasnjeni nekateri razlogi in način, kako in zakaj je Iskra vstopila na področje opreme za poslovno obdelavo podatkov. Poleg okvirnega opisa računalniško dejavnosti Iskre v preteklosti, so obdelani tudi odnosi z inozemskimi partnerji ter principi dejavnosti Iskre-TOZD Računalniki.

COMPUTERS AT ISKRA: The article is a short summary of Iskra's activities on computers. There is a short explanation why, what and how is Iskra entering the area of business computers. Some principles about the operations in Iskra-Computers Plant are explained.

UVOD

Članek ni kompleksni uvodnik oziroma prikaz celotne računalniške dejavnosti v Iskri, ampak le skuša obdelati nekatere probleme pri osvajanju te moderne tehnologije pri enem od domačih proizvajalcev. V njem ni skoraj nobenih številčnih tehničnih podatkov, ki bodo prikazana, upam, v kakem od naslednjih Iskrinih prispevkov.

Kljub temu, da je problematika obravnavana na drugačen način kot je bilo prvotno mišljeno, pa je, upam, dovolj zanimiv tudi za tako strokovno revijo kot je INFORMATICA.

Moje pogledе in izkušnje (elektronik, 27 let delovne dobe, od tega preko 20 let pri Iskri; na računalnikih 7 let, intenzivno nekaj manj kot 2 leti) nikakor ne gre jemati kot postulate, niti ne trdim, da se z njimi strinja vseh 27000 Iskrinih delavcev.

ISKRA IN RAČUNALNIŠTVO

Zelo koristno je vedno navesti svoje reference (svoje sem navedel že v uvodu). Zadnje čase je to postala (na področju računalništva) posebne vrste nuja v naši domovini. Podatki, kaj je neka grupa ali grupacija proizvedla oz. dobavila za enotni jugoslovanski trg, so zaprepaščujoči. Pri tem ni nič važno, da se primerjajo podatki kot, n.pr., podatki o številu vozil v uporabi, pa bi navedel proizvajalec otroških vozičkov isto število kot proizvajalec avtomobilov.

Če se strinjamo s tako metodologijo, potem lahko za Iskro trdimo, da je danes prisotna na preko 400 instalacijah opreme za avtomatsko obdelavo podatkov doma in v tujini. Pošteno je, če ta podatek razčlenim. Na prvem mestu (po vrednosti lastnega vloženega dela) so procesorji za krmiljenje avtomatskih telefonskih central. Teh je okrog 50, med njimi tudi, verjetno, eden od najhitrejših računalnikov, dobavljenih iz tujine, v SSSR.

Drugo mesto zaseda (ne po vrednosti vloženega dela, ampak zato, ker je eden izmed redkih, če ne edini jugoslovanski CPU, ki je razvit brez tuje pomoči in ki ga Iskra serijsko proizvaja iz osnovnih elektronskih elementov) mikroračunalnik ISKRADATA 1680. Poizkusna serija (10 kosov) je bila proizvedena leta 1977, letos je stekla redna proizvodnja. Potem sledijo (vrstni red določa lastno vloženo delo in ne finalna vrednost proizvoda): nekaj 10 sistemov za krmiljenje procesov v elektrogospodarstvu, 250 instalacij Philipsove opreme za AOP (skupina, ki je izdelala aplikacijske software pakete, installirala in vzdržuje to opremo, se je leta 1977 integrirala z Iskro), v okviru Iskrinih zastopstev pa je bilo izvedenih nekaj 10 instalacij CDC opreme in nekaj 10 instalacij Hewlett Packard opreme.

Pri teh projektih, predvsem pa pri osvajanju proizvodnje procesorjev za avtomatske telefonske centrale, je Iskra ob zelo dobrem sodelovanju znanstvenih ustanov (predvsem EF Univerze v Ljubljani in Instituta Jožef Stefan) pridobila toliko znanja, da je začetni (licenčni) odnos do inozemskega partnerja postal nebistven. Že na samem začetku osvajanja hardwarske opreme je šla Iskra svojo - neticenčno - rentabilnejšo pot. Zato je Iskra danes strojno opremljena v taki meri, da lahko proizvaja elektronske podsklope, sklope in aparaturo za računalniške sisteme zelo širokega spektra. Del te proizvodnje lahko pokriva tudi z lastno proizvodnjo integriranih vezij. Seveda pa je vprašanje rentabilnosti kritično, saj so računalniški sistemi proizvajajo v razmeroma majhnih serijah. Velikokrat temelji odločitev bolj na strateški kot na rentabilnostni osnovi. Mogoče je tipičen primer ravno procesor za ATC, kjer je računalnik le del velikega tehničnega projekta.

ODNOS DO INOZEMSKIH PARTNERJEV

Zavedam se "napake", ki sem jo storil vedoma, s tem da sem - zelo posplošeno - prikazal Iskrino dejavnost na računalniškem področju kot hardwarsko proizvodnjo.

Problemi softwarske proizvodnje, šolanja uporabnikov, montaže in uvajanja sistemov ter njih redno vzdrževanje in popravila so seveda tudi za Iskro pomenila in deloma še pomenijo najtrši oreh na tem področju.

Predvsem pri osvajanju znanja o sistemskem softwaru je bila edina pot (stroški in izkušnje) nasloniti se na inozemskega partnerja. Pri tem bi bil seveda idealen kooperacijski odnos (uvoz = izvoz), ki pa ga je bilo na tem tehnološko izredno dinamičnem področju nemogoče doseči. Žal, so želje oziroma predstave nekaterih naših strokovnjakov (ki so dejansko vrhunski znanstveniki na teoretičnem področju računalništva) popolnoma nerealne. Nobena od priznanih proizvajalk tako važne opreme kot so računalniki ne more iti v tveganje, da bi bila za nekatere bistvene dele naprav navezana na dobave od kooperanta, ki je v principu popolnoma neodvisen (finančno in politično) partner. V kolikor pride do kooperantskega odnosa, je to na nebistvenih podsestavih, ki vsebujejo malo ali nič računalniškega znanja, ali pa nudijo "kooperacijo" firme, ki pomenijo malo ali nič v svetu računalnikov.

Edini za nas še sprejemljivi odnos je bil izbor primernega licenčnega partnerja. S tako obliko sodelovanja si lahko zagotovimo dovolj trdno osnovo za sedanjost in prihodnost. Seveda pa je poleg predmeta in določil licenčne pogodbe veliko odvisno od interesa ter znanja obeh partnerjev, kako ta določila izvajata. Tu lahko trdimo, da si je Iskra tekom nekajletnega delovanja na področju specializiranih računalnikov nabrala dovolj izkušenj, da je vstopila še na področje poslovne obdelave podatkov. S tem je izpolnila le nekak imperativ, ki ga je postavil nagel razvoj (in s tem pocenitev) ter prodor računalniške opreme na izredno širokem področju. Brez pretiravanja lahko trdimo, da bi elektronska industrija, posebno profesionalno usmerjena, v nekaj letih propadla, če bi ne obvladala računalniške dejavnosti v dovolj širokem spektru. Za nas, samoupravno družbo, je čim večja osamosvojitev na tem področju še posebej pomembna, kar potrjuje tudi srednjeročni plan SRS, v katerem je Iskra posebej zadolžena za računalniško področje.

Poseben članek bi bil potreben, če bi hotel opisati značilnosti licenčne pogodbe Iskra-CDC za družino miniračunalniških sistemov ISKRADATA C-18. Pogodba vsebuje poleg določil o prenosu znanja tudi čvrsta kupoprodajna določila za dobave, ter nekatere elemente kooperacije. Z njo si je Iskra zagotovila osnovo za izpolnjevanje planskih obveznosti. V tem uvodniku bi raje nanizal še nekaj misli o precej hvaljenem OEM odnosu. OEM pogodba je v bistvu navadna kupoprodajna pogodba nižjega razreda. Nižjega razreda zato, ker se dobavitelj ne obvezuje drugega, kot da dobavi hardware, ki ima določene lastnosti. Navadno taka, OEM, pogodba niti ne vsebuje klavzule o garanciji. Proizvajalec ne jamči za pravilno delovanje podsklopa ali aparature v sistemu, ki ga mora konceptirati kupec (ali prodajalec), niti proizvajalec ne daje časovne garancije. Velikokrat ni obvezan proizvajalec posredovati niti vsega znanja za pravilno uporabo hardwara. Kljub temu se je OEM odnos precej uveljavil in to predvsem v deželah s kapitalistično ureditvijo. Vendar je še tam omejen v glavnem na periferne naprave, ki jih kupujejo tudi veliki proizvajalci računalnikov pri velikih proizvajalcih perifernih naprav, ki so včasih tudi sami priznani proizvajalci računalnikov. Firme - sistemske hiše, ki se navezujejo z OEM odnosom na proizvajalce CPU, so v glavnem muhe enodnevnice, ki jih ustanavljajo večinoma strokovnjaki - odpadniki velikih firm z namenom pobrati začetni profit na nekem novem področju ne glede na politično ali gospodarsko škodo za družbo. Za njimi ostane često tudi "pogorišče", t.j., z izginotjem v OEM nabavo orientirane firme izgine tudi vsa podpora (vzdrževanje sistemov) uporabnikom.

ZAKLJUČEK

Upam, da sem, precej posplošeno sicer, vendar dovolj jasno nanizal osnovne razloge, ki so vodili Iskro, da je vstopila, na opisani način, v področje poslovne obdelave podatkov. Za dopolnitev grobe slike naj navedem še nekaj.

V okviru DO Iskra-Elektromehanika je bila 1. 1977 ustanovljena TOZD Računalniki. TOZD Računalniki je enota, ki deluje bolj v smislu kompletnega engineeringa kot pa na način klasične proizvodnje. Razlog je bil implicitno naveden že v tekstu. DO Elektromehanika je namreč opremljena oziroma proizvaja v svojih tozdih že vse sestavne dele - podsklope, ki so potrebni za CPU. Ekonomsko neopravičljivo bi bilo podvajati to proizvodno opremo. Zato nastopa TOZD Računalniki kot finalist pri proizvodnji hardwara (integracija sistemov) in kot enota, ki oskrbuje uporabnike s sistemi "na ključ". Pri tem seveda ne računamo, da bomo vse probleme lahko obvladali sami. Zanimiv podatek je, da pri tem, več ali manj uspešno, sodelujemo (ali skušamo koordinirati oz. sodelovati) s kar 18 različnimi znanstvenimi institucijami, družbenopolitičnimi forumi, ostalimi jugoslovanskimi proizvajalci in važnejšimi velikimi uporabniki.

Pri tem je seveda vsak prepričan, da so problemi, ki jih gleda iz svojega zornega kota, "naj, naj prioritetnejši". Tako, recimo, bi bilo za nekatere že vse rešeno, če bi Iskra proizvajala le periferijo (ki je problem zase; o njem, če bo urodništvo za to, kdaj drugič), ostali pa bi reševali in rešili "kompleksno dejavnost". Pa ima(m) človek velikokrat občutek, da se niti ne zavedajo, da spada pod "kompleks" tudi, n.pr., vprašanje, kako obvladati čez 2-3 leta vozni park 70 servisnih vozil, če hočemo v redu vzdrževati planirane sisteme pri uporabnikih !

ida-programski paket za izvajanje funkcij digitalnega avtomata z mikro računalnikom

d.čuk
*m.tomšič
r.tavzes

Institut "Jožef Stefan", Ljubljana
*Fakulteta za strojništvo, Ljubljana

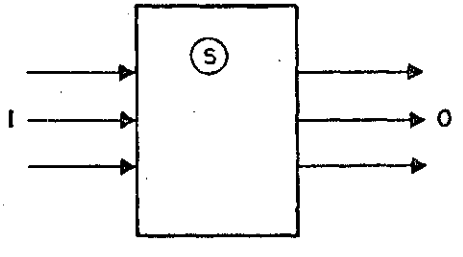
UDK 681.3-181.4.06 IDA

Opisan je programski paket IDA, ki omogoča izvajanje funkcij digitalnega avtomata. Prikazane so prednosti mikroračunalniške izvedbe, s poudarkom na enostavnem vnašanju podatkov v programskem jeziku IDA. Paket IDA omogoča poljubno razširitev osnovnih funkcij digitalnega avtomata z novimi krmilnimi programi za okna. S pomočjo programa IDA je izdelan univerzalni digitalni programator UDP, ki ima že vgrajene krmilne programe za vhodno-izhodne enote in za izvajanje funkcij časovnih relejev in števcev. Opisana je tudi praktična uporaba UDP v programatorju za brizganje plastične mase.

IDA - INTERPRETATIV DIGITAL AUTOMAT MADE WITH MICROCOMPUTER. This paper deals with program packet IDA for execution functions of digital automat. Comparative advantage of microcomputer version is presented, with emphasyse on program language IDA. Program IDA makes possible to expanse normal functions of digital automat by new programs, called drivers. Universal digital programator UDP is made of program package IDA and contains drivers for serial and paralel input-output units, drivers for program timers and counters already. It is also discused about practical use of UDP in programator for automatic injection moulding machines for plastics.

1. UVOD

Avtomat je sistem, katerega delovanje ni neposredno odvisno od človeka; če gre pri tem za digitalne spremenljivke, govorimo o digitalnem avtomatu. Avtomatizacija industrijskih procesov se da pogosto realizirati s pomočjo krmilne enote, ki deluje kot digitalni avtomat. Za avtomat so značilna notranja stanja S, vhodi I in izhodi O (slika 1).



Slika 1. Preprosta shema digitalnega avtomata.

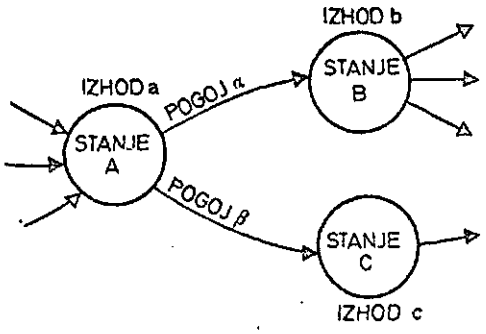
Avtomat prehaja iz stanja v stanje v odvisnosti od vhodov, izhodi pa so lahko:

- odvisni od stanja in prehoda v to stanje (Mally-jev avtomat)
- odvisni le od stanja (Moorov avtomat).

Značilno je, da se da vsak Mally-jev avtomat pretvoriti v Moorovega in obratno /1/. Ker je struktura Moorovega avtomata enostavnejša, je na osnovi tega izdelan digitalni avtomat IDA. Digitalni avtomat se da izdelati s stalno logiko ali programirano - s pomočjo računalnika (ki je tudi digitalni avtomat!). Če želimo spremeniti delovanje avtomata s stalno logiko, moramo izvesti prevezave, ali pa spremeniti, dodati ali odvzeti dele opreme. Avtomati za različne namene bodo imeli tudi različno opremo. Količina opreme (vezij) s povečanjem kompleksnosti močno naraste. Pri računalniški realizaciji je oprema stalna, vsaka sprememba funkcije zahteva le spremembo programa. Z ustrezno organizacijo programov in programskim jezikom (točka 4) pa je to zelo preprosto. Slaba stran računalniške verzije je v hitrosti odziva, saj gre pri tem za zaporedno obdelavo informacij, medtem ko je pri stalni logiki paralelna obdelava. V številnih praktičnih primerih je zahtevani reakcijski čas mnogo večji od reakcijskega časa računalnika, tako da je računalniška izvedba sprejemljiva. Zaradi vse nižje cene mikroračunalnikov, je tudi iz ekonomskega vidika vse bolj upravičena uporaba mikroračunalnika za izvajanje funkcij digitalnega avtomata.

2. OPIS DIGITALNEGA AVTOMATA

Digitalni avtomat se najlaže grafično prikaže z diagramom prehajanja stanj (slika 2). Predpostavimo, da se avtomat nahaja v stanju A. Izhod a je odvisen le od stanja (Moorov avtomat). Sprememba izhodov se izvrši le v primeru, ko avtomat preide v novo stanje, to pa pomeni, da morajo biti vhodi takšni, da je izpolnjen



Slika 2. Izsek iz diagrama prehajanja stanj.

ali pogoj α ali β, v prvem primeru preide avtomat v stanje B, v drugem pa v C. Tabelarično je lahko avtomat opiše s tabelo prehodov (slika 3).

stanje	izhod	pogoj - prehod v	pogoj - prehod v	pogoj - prehod v
⋮				
A	a	α - B	β - C	...
⋮				

Slika 3. Tabela prehodov

V tabeli prehodov določa vrstica trenutno stanje avtomata, stolpci pa pogoje prehodov, prvi stolpec pa določa vrednost izhodov stanja (Moorov avtomat).

3. ZAMISEL IN IZVEDBA

Želeli smo izdelati posplošen digitalni avtomat; ki bi izkoriščal čim več prednosti mikroračunalniške izvedbe, predvsem glede enostavne spremembe funkcije avtomata. Poleg tega smo želeli doseči čim bolj kompakten zapis tabele prehodov. Realizacija zaradi koristnih možnosti, ki jih nudi računalniška izvedba, odstopa od najpreprostejšega, Moorovega avtomata.

3.1. Posplošena okna in njihovi krmilni programi

Pri digitalnem avtomatu so vhodi in izhodi digitalne spremenljivke, to pa pomeni, da so to lahko posamezni biti digitalnih vhodov oz. izhodov računalnika. Ker gre za realizacijo z 8-bitnim mikroračunalnikom, je po 8 vhodov in izhodov združenih, in to predstavlja besedo mikroračunalnika - imenujemo jo okno. Okna torej predstavljajo preslikavo vhodov/izhodov v aktivni (RAM) spomin računalnika. Osnovno so vhodi/izhodi fizični vhodi/izhodi računalnika, pri tem je vsak bit v programskem oknu enak nekemu fizičnemu vhodu ali izhodu.

Funkcijo okna lahko posplošimo tako, da beseda v spominu lahko predstavlja povezavo (okno) med različnimi programi v računalniku, ne pa z zunanjim svetom.

Funkcijo posameznega okna definirajo njemu podrejeni krmilni programi. Ti programi izvedejo preslikavo informacije z/na fizični vhod/izhod, ali kak drug podatek v računalniku. Krmilni programi niso del IDA in so specifični za vsak digitalni avtomat, s čimer je ohranjena splošnost avtomata. Naslovi krmilnih programov so v posebni tabeli v okviru paketa IDA.

3.2. Interne spremenljivke

Moorov avtomat ima eno interno spremenljivko (številka stanja), IDA pa jih lahko vsebuje več. To so spremenljivke, ki se ne uporabljajo za direktno definiranje pogojev prehodov. Uporabljajo jih lahko krmilni programi oken ali pa začetni in ponavljalni programi (glej točko 3.3.).

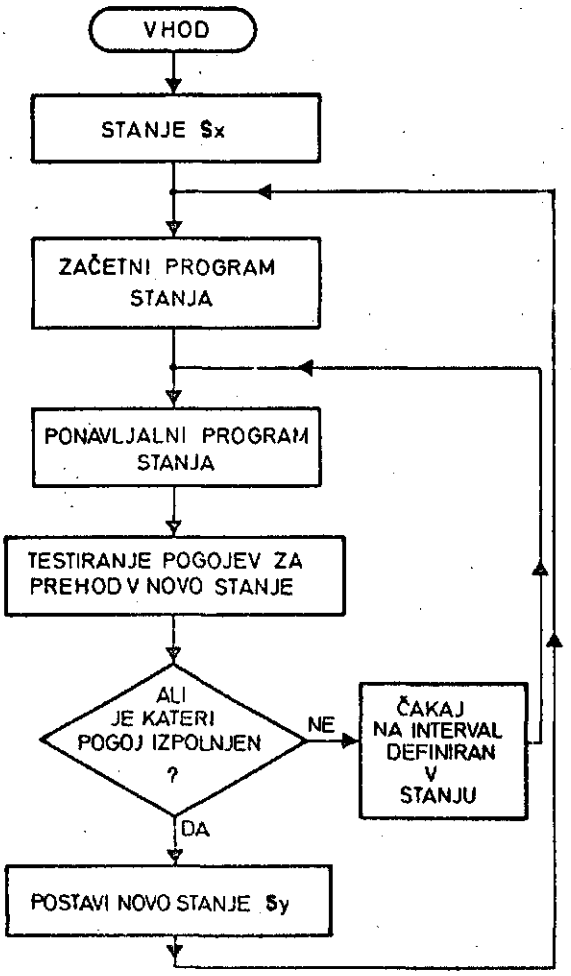
3.3. Primer uporabe posplošenih oken in internih spremenljivk za izvedbo časovnega releja

IDA nudi možnost izdelave programskih časovnih relejev, kar bi pri čistem Moorovem avtomatu moralo biti izvedeno z dodatno opremo. Željeni časovni interval bomo nastavili z zunanjimi stikali, ali vnaprej programsko določili. Eno izmed oken mora imeti krmilni program, ki čita željeni interval in ga spravi v interne spremenljivke. Ob vsaki časovni prekinitvi prekinitveni program odšteva vrednosti notranje spremenljivke. Ko se čas izteče, nastavi program odgovarjajoči bit v izbrano programsko okno. Če je ta bit izbran kot pogoj za prehod, bo avtomat prešel v naslednje stanje.

3.4. Struktura programa IDA

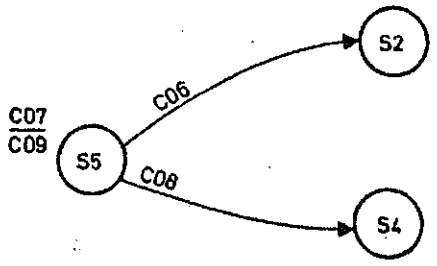
Ideja programov IDA je, da imamo enoten izvršilni program, ki s pomočjo podatkov o avtomatu, zbranih v tabeli, vrši željeno funkcijo. V tabelo se mora dati podatke enostavno vnašati ali jih spreminjati.

Zaradi zaporednega delovanja računalnika se preverjanje pogojev v vsakem stanju izvršuje ponavljajoče (slika 4). To opravi "ponavljalni program (stanja)". Ponavljalni program preveri vse vhode in sicer tako, da najprej kliče krmilne programe za vsa okna, ki jih vsebujejo pogoji za prehod, nato pa preveri vrednosti oken - ali ustrezajo prehodnemu pogoju. Podobno nastavi vsa izhodna okna "začetni program stanja".



Slika 4. Diagram poteka ter glavni izvršilni program IDA

Preslikava in preverjanje oken zahteva nekaj časa. Da tega čim bolj zmanjšamo, posebej definiramo, katera okna je treba upoštevati v začetnem programu in katera okna v nadaljevalnem programu. Na sliki 5 je prikazan del diagrama prehodov z oznakami, ki jih uporabljamo v jeziku IDA.



Slika 5. Diagram prehajanja stanj za opis v jeziku IDA

Ta del diagrama prehajanja stanj opišemo z naslednjimi kodami:

=IS05-07-09
C06-02
C08-04
CFF

Vse oznake, ki se na sl. 5 pričenjajo s C, se nanašajo na okna in jih imenujemo "označbe". Informacije, ki jih vsebuje tabela označb, so enotne po obliki, njihov pomen pa je različen. Oznaka CFF pomeni konec opisa stanja.

Primer opisa označbe v jeziku IDA:

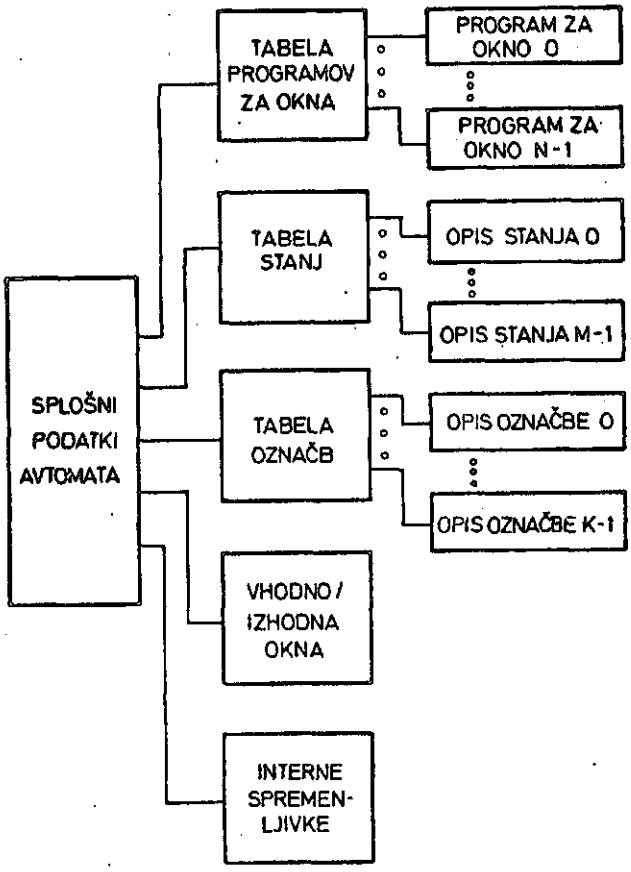
=IC11
ϕ05 = 50-02
ϕ08 = 30-00
ϕFF

Opis označbe zaključuje ϕFF. Ostale vrstice se nanašajo na okna (okno 5 in okno 8). Za enačajem sta dve osembitni besedi (števili), ki ju uporablja krmilni program okna. Označbe so uporabljene na dva načina (glej sl. 5): kot podatek za začetni ali ponavljalni program stanja in kot podatek o pogojih za prehode.

Začetni in ponavljalni programi stanja kličejo krmilne programe za okna, ki so navedena v opisu označbe. S tem se preslikajo vrednosti fizičnih vhodov v spomin (okna) oziroma obratno. Ali je okno vhodno ali izhodno, je definirano s krmilnim programom. Pri vhodnih oknih so podatki, ki sledijo enačaju (n.pr. ϕ05=50-02) običajno brez pomena. Pri izhodnih oknih pa običajno prva beseda vsebuje podatke, kateri biti izhodov naj se ne spremenijo, druga beseda pa, kako naj se ti biti postavijo (0 ali 1). Krmilni programi oken niso del ožjega paketa IDA ter jih lahko poljubno dodajamo, s tem pa tudi lahko spremenimo pomen informacij, vsebovanih v označbah, ki jih uporabljajo začetni in ponavljalni programi.

Označbe, uporabljene v pogojih za prehod iz stanja (n.pr. C06 in C08 na sl. 5) imajo drugačen pomen. Prvi podatek za okno, ki sledi enačaju (n.pr. ϕ5=50-02) označuje, kateri biti okna morajo biti nastavljeni ("1"), drugi podatek pa, kateri morajo biti brisani ("0"), da se prehod izvrši.

Vsi podatki o avtomatu, ki ga realiziramo s programom IDA, so opisani v tabelah. Med vpisovanjem podatkov se lega in obseg tabel sproti prilagaja, zato je naslavljanje podatkov posredno, preko kazalnih tabel (slika 6).



Slika 6. Organizacija podatkov interpretativnega digitalnega avtomata (IDA)

4. JEZIK IDA

Jezik IDA je interaktivni programski jezik, ki omogoča vnašanje, preverjanje in spreminjanje podatkov ter testiranje delovanja željenega avtomata.

Vsebuje naslednje ukaze:

=B - (begin) ničenje tabel v aktivnem spominu (RAM-u). Program namreč formira tabele na RAM-u, kjer tudi testiramo delovanje. Ko smo z delovanjem zadovoljni, prepišemo tabele na PROM-e preko PROM-programatorja.

=I - (insert) vnesemo novo stanje ali označbo (glej 3.3)

=D - (display) prikaže se stanje ali označba v isti obliki kot pri vprašanju

=K - (kill) uniči se določeno stanje ali označba

=G - (go) proženje programa od določenega stanja naprej

=J - (jump) izvajanje programa od stanja do stanja

=P - (program) programiranje mrtvega spomina (PROM-a). Program javi zahtevano število strani (256x8 bitov), s številko pa povemo, katera stran naj se sprogramira

=A - (active) prenese vsebino tabel z mrtvega spomina (PROM-a) v aktivni spomin (RAM).

=L - preadresiranje tabel v predvideno lego mrtvega spomina (za programiranje mrtvega spomina)

=M - preadresiranje tabel za živi spomin (če pomotoma uporabimo ukaz =L).

Podatke o stanjih in označbah, t.j. definicijo strukture avtomata, vpisujemo po ukazu =I na način, ki je opisan v točki 3.3. Pri tem IDA izpisuje iztočnice. Primer vpisa stanja in označbe; znaki, ki jih izpisuje IDA, so podčrtani:

```
=IS05-07-09
C06-02
C08-04
CFF
=IC11
$05=50-02
$08=30-00
$FF
=  IDA čaka nov ukaz
```

Prikaz dela strukture je v enaki obliki, kot jo uporabljamo za vpis. Primer prikaza stanja S05:

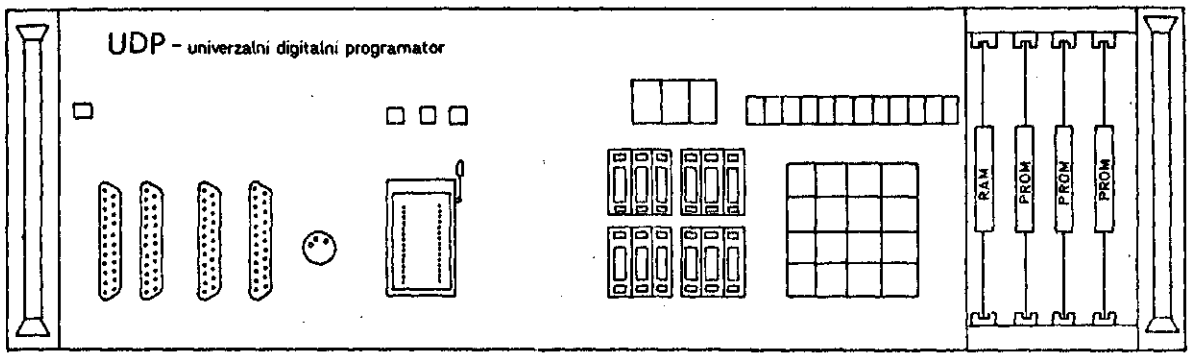
```
=DS05
S05-07-09N
C06-02N
C08-04N
CFFN
=
```

IDA nam lahko javi, da tega stanja ni, ali pa izpiše podatka za začetni in ponavljalni program. Sedaj pričakuje vpis črke N ali katerekoli druge črke. V prvem primeru nam bo izpisal prvi prehod, drugače javi = in čaka na nov ukaz. S čakanjem črke omogočimo uporabo enovrstičnega prikazovalnika za vhodno enoto računalnika. Po izpisu celotnega stanja, čaka IDA nov ukaz.

Podrobnejši opis jezika IDA je v /4/.

5. REALIZACIJA IDA Z MIKROPROCESORJEM TIPA 8080

Prevajalne in izvajalne programe IDA smo realizirali na mikroračunalniku sistema "mikro-m" /5/ s procesorjem tipa 8080. Prevajalni programi obsegajo okoli 2 K spomina (8-bitnega), izvajalni programi pa okoli 1/2 K. Prevajalni programi za izvajanje niso potrebni in jih v dokončno realizacijo ni potrebno vgraditi. Obseg krmilnih programov, ki so za delovanje avtomata potrebni, je odvisen od števila različnih vhodov in izhodov in njihove kompleksnosti. Posamezni krmilni program lahko obsega od nekaj deset do nekaj sto besed.



Slika 7. Zunanji izgled univerzalnega digitalnega programatorja

Smatramo, da je poraba spominskega prostora za opis avtomata t.j. tabele, blizu teoretičnega minimuma. Za opis posameznega stanja je potrebno 8 + 2 x n besed, kjer je n število prehodnih pogojev, za vsako označbo pa $2 \times i(\frac{m}{8} + 7)$, kjer je i število oken v označbi, m pa število vseh uporabljenih oken; ( ) pomeni celoštevilčni del izraza.

Število stanj, označb in oken je v tej realizaciji omejeno na 255, kar pa za večino primerov zadostuje.

Reakcijski čas je odvisen od števila oken, pogojev za prehod iz stanja in časovne zahtevnosti krmilnih programov oken (od manj kot 1 m sek, do nekaj m sek).

Modularnost izvedbe paketa IDA omogoča vključitev številnih dodatkov, ki presegajo definicijo "digitalni avtomat", oziroma razširitev obsega strukture avtomata čez prej opisane omejitve. Vendar taki posegi zahtevajo detajlno poznavanje zgradbe programov in znanje programiranja v strojnem jeziku, medtem ko uporaba osnovnega sistema IDA zahteva le poznavanje nekaterih preprostih navodil.

6. PRIMERI UPORABE IDA

6.1. UDP - univerzalni digitalni programator

S pomočjo programskega paketa IDA in mikroračunalniškega sistema mikro-m je zgrajen univerzalni digitalni programator UDP (slika 7), ki služi za razvoj in izvajanje funkcij digitalnega avtomata. UDP ima vgrajeno:

- tastaturo za delo z jezikom IDA
- 12-znakovni prikazovalnik, ki služi za izhodno enoto pri pripravi in testiranju programa
- 4 grupe po 3-kodirna stikala, ki poljubno služijo ali kot ročni vhodi (npr. za izbiro željenega programa) ali za nastavitev začetnih vrednosti časovnikov oz. števcev. Izbira funkcije je programska.
- trošteviličnі prikazovalniki, kjer se prikazuje v odvisnosti od programa vrednost enega izmed 4 časovnikov ali 4 števcev.
- PROM programator mrtvega spomina (PROM) za zapis končne verzije programa.
- izhodni konektor za optično izoliran serijski vhod/izhod.
- izhodne konektorje (4) za optično izolirane vhode/izhode (4 x 12 vhodov in 4 x 12 izhodov).

Program UDP je razširjen IDA, saj ima že vgrajene krmilne programe za paralelne vhode in izhode, serijski vhod in izhod, za čitanje vhodov preko kodirnih stikal, za programsko izvajanje funkcij časovnega releja (istočasno lahko 4) in števca (do 4). Te lahko nastavljamo programsko ali preko kodirnih stikal. Števci služijo za to, da se lahko določen del v diagramu prehajanja stanj kontrolirano število-krat ponovi.

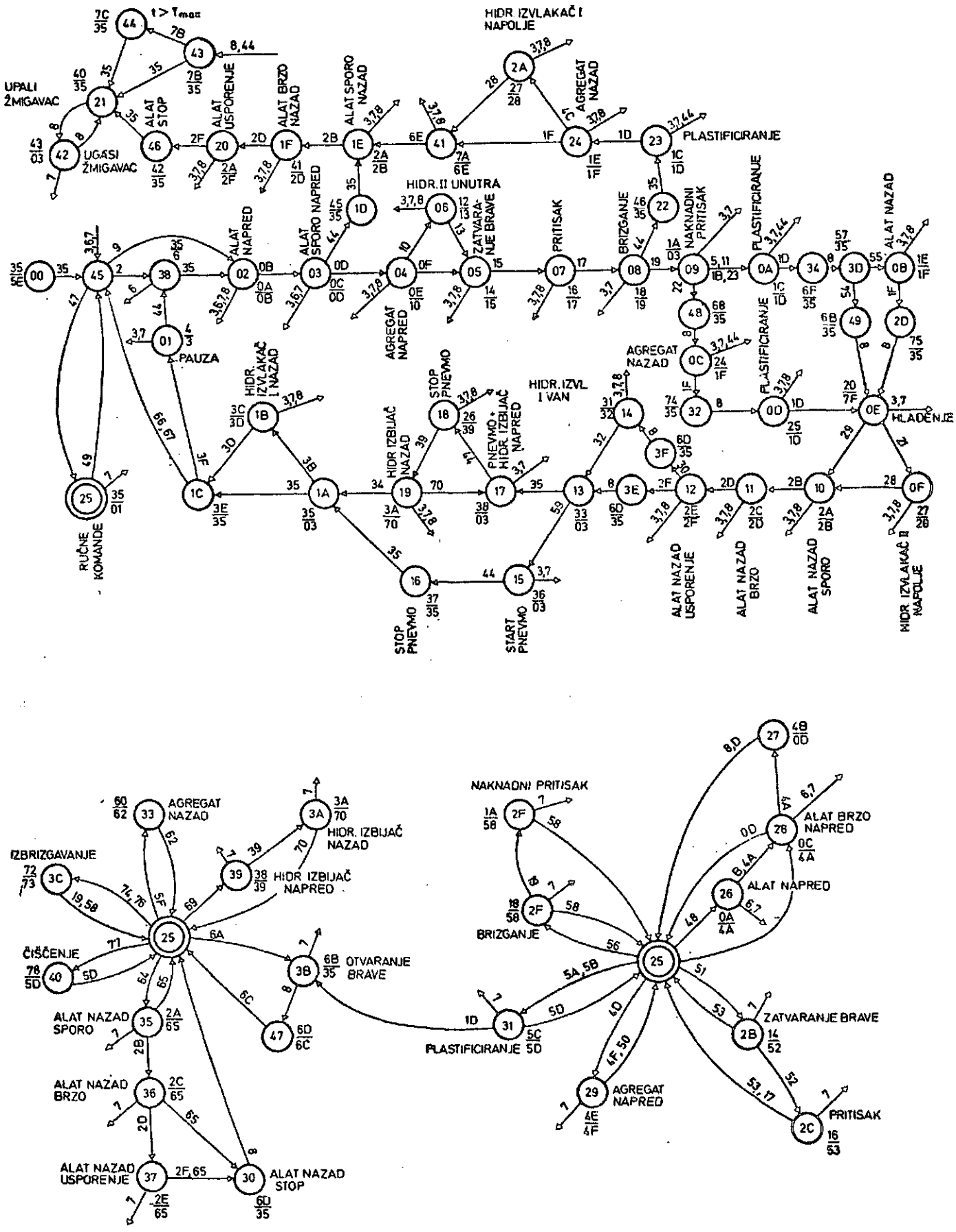
Serijski prenos je izdelan za primer, da gre za razmeroma počasen avtomat, in da je željeno oddaljeno krmiljenje. (hitrost prenosa je 4800 bitov/sek). Razvita je "neinteligenčna" postaja, ki jo vgradimo v oddaljen stroj in komunicira s serijskim prenosom. Z vgradnjo optičnih spojk v to povezavo učinkovito rešimo problem motenj.

Paralelnih vhodno/izhodnih vodov je do 96, možna pa je še nadaljna razširitev (novo ohišje). Vsi vodi so optično izolirani.

UDP služi tako za razvoj avtomata, kot za njegovo delovanje. Izvajalni program UDP zavzema 2 K 8-bitnih besed.

6.2. Programator za stiskalnice

UDP brez razvojnega dela (tastature, prikazovalnika, programatorja in jezika IDA) in z uporabo serijskega prenosa podatkov je vgrajen v programator za brizganje plastične mase.



Slika 8. Primer diagrama prehajanja stanj.

Sestavljen je iz 6 kartic in postaje s 3 karticami evropskega formata. Diagram prehajanja stanj za ta stroj je prikazan na sliki 8. (Dvojni krog v diagramu pomeni, da je stanje večkrat opisano.) Avtomat sestoji iz 74 stanj in 128 označb. Število vhodov (stikal) je 36, izhodov (elektromagnetov) pa 35. Opis avtomata (tabele) zavzema 2 $\frac{1}{4}$ K 8-bitnih besed.

V fazi testiranja se je pokazala velika zanesljivost delovanja mikroračunalniškega programatorja UDP in enostavno spreminjanje funkcije, s čimer smo se približali optimalnemu delovanju stroja. Z uporabo programskih časovnih relejev smo dosegli mirno delovanje stroja brez uporabe proporcionalnih elektromagnetnih ventilov, s čemer dosežemo velik prihranek na hidravlični opremi. Povečala pa se je tudi hitrost delovanja stroja.

6. ZAKLJUČEK

Programski paket in jezik IDA za opis in realizacijo digitalnega avtomata je že v dosedanji razmeroma kratkotrajni uporabi pokazal znatne prednosti pred realizacijo funkcij neposredno s programi v strojnem oziroma zbirnem jeziku ali v kakšnem višjenivojskem jeziku, ki ni problemsko usmerjen (n. pr. FORTRAN). Prednost jezika IDA je še večja za tiste številne potencialne uporabnike, ki sicer znanja računalniškega jezika ne potrebujejo, saj lahko vpišejo podatke o delovanju avtomata neposredno z grafičnega opisa z diagramom prehodov. Univerzalni digitalni programator (UDP) pa s tipizirano izvedbo vhodov in izhodov, ki jih lahko neposredno uporabimo v praksi, pomeni še nadalnji korak k bolj množični uporabi mikroračunalnikov za realizacijo krmilij tipa "digitalni avtomat".

LITERATURA

/1/ A. Hussu: Teorija digitalnih avtomatov, Skripta Fakultete za strojništvo v Ljubljani, 1977

/2/ A. Janeš: Diplomsko delo 2499, Fakulteta za strojništvo v Ljubljani, 1977

/3/ D. Čuk: Avtomatizacija procesov z računalnikom, Prvi podiplomski seminar na Fakulteti za strojništvo v Ljubljani, marec 1978

/4/ Priročnik za uporabo jezika IDA, IJS, Ljubljana, 1978

/5/ Tehnična dokumentacija modularnega mikroračunalnika sistema mikro-m, IJS, Ljubljana, 1978.

-oOo-

mikroračunalniški pid regulator

j.tasič
v.sever

Institut "J. Stefan", Ljubljana

UDK 681.3-181.4:621.316.72-551.454

Članek obravnava vodenje modela procesa s proporcionalnim in proporcionalno-integralno-diferencialnim regulatorjem. Regulator je realiziran z mikroračunalnikom INTEL 8080. Podana je primerjava miniračunalniškega in mikroračunalniškega regulatorja za enako konfiguracijo modela. To govori v prid mikroračunalnika s stališča velikosti, hitrosti in porabljenega pomnilnika.

CONTROL OF A PROCESS MODEL USING A MICROCOMPUTER. Control of a process model using proportional as well as proportional-integral-differential controllers is discussed in this paper. The controller was realized by means of an INTEL 8080 microcomputer. A minicomputer based controller is compared to the microcomputer based controller on the same model.

UVOD

V zadnjem času se v svetu vse več prehaja na področje vodenja industrijskih procesov z mikroračunalniki. Procesni regulator sprejema več podatkov in glede nanje upravlja proces. Večina regulatorjev je danes analognih. Ker pa so industrijski sistemi lahko zelo obširni in zahtevni, so potrebni *številni analogni regulatorji, ki pa so sposobni regulirati le posamezne procese*. V želji, da bi naredili obširen in učinkovit sistem procesnega vodenja, so prišli do realizacije *računalniškega regulacijskega sistema*. Le-ta je dovolj hitar za večino industrijskih procesov in lahko opravlja še razne druge funkcije.

Mikroračunalniške regulatorje se uvaja v procese predvsem zaradi tega, ker je lahko za različne regulacijske sisteme osnovna instrumentalna oprema enaka, razlikuje pa se programska oprema, kar vpliva tudi na nizko ceno mikroračunalniškega regulatorja.

Regulatorji v industrijskih procesih delujejo po vnaprej določenih programih. Najenostavnejši digitalni algoritmi so diskretni ekvivalentni analognih regulatorjev (proporcionalni, *integralni, diferencialni in kombinacije teh*). *Pri enozančnih* regulacijah je PID (proporcionalno-integralno-diferencialni regulator) najobičajnejši in najprimernejši, zato smo takšen regulacijski algoritem tudi realizirali. Računalniški sistem na osnovi ustreznih algoritmov ugotavlja kritične in pomembne vrednosti posameznih funkcij industrijskega procesa in glede nanje in na zahteve vodi proces.

Ker je bilo programiranje izvedeno v zbirnem jeziku, je celoten program obsegal le 2 k pomnilnika. Osnovni regulacijski cikel pri PID regulatorju, ki je minitačunalniku (1) znašal 74 ms, se je znižal pri uporabi mikroračunalnika pod 30 ms. Natančnost regulacijskega sistema znaša približno 0,5 %. Z izboljšanjem natančnosti analogno-digitalnega pretvornika (iz 8 bitne pretvorbe na vsaj 10 bitno) bi natančnost tega sistema povečali na približno 0,2 %.

MATEMATIČNE OSNOVE ZA RAZLIČNE DDC REGULATORJE

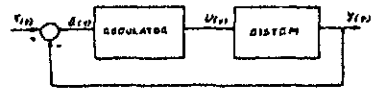
Osnovne matematične analize za regulacijo industrijskih procesov potekajo v časovnem in frekvenčnem prostoru. Pri direktni digitalni regulaciji (DDC) z računalniki pa reguliran sistem ne opazujemo vedno, ampak le v enakomernih intervalih T. Čas T je določen s Shannovo enačbo:

$$T = \frac{1}{2f} \quad , \qquad (1)$$

kjer je f zgornja frekvenca opazovanega sistema.

V takih primerih preide frekvenčni prostor v z-prostor. Ker je v praksi dobrih digitalnih regulacijskih algoritmov malo in je analiza analognih regulacijskih sistemov dobro razvita, se pogosto uporabljajo diskretni ekvivalentni analognih regulatorjev.

Najenostavnejši zvezni regulator je tak, kjer je vhod $u_{(t)}$ v sistem proporcionalen pogrešku $e_{(t)}$. (Slika 0).



Slika 0.

$$e_{(t)} = r_{(t)} - y_{(t)} \quad \text{in} \quad u_{(t)} = P\, e_{(t)} \; . \qquad (2)$$

Ker pa pri regulaciji z računalnikom opazujemo regulirani sistem le v trenutkih vzorčenja, preideta enačbi (2) v obliko enačbe (3).

$$U_n = P\, E_n = P(R_n - Y_n) \qquad (3)$$

Enačba (3) predstavlja takoimenovani pozicijski proporcionalni regulator.

Slabost proporcionalnega regulatorja je v tem, da ima stacionarno napako (zato vpeljemo integralni del) in da je

neobčutljiv na hitre spremembe pogreška (zato vpeljemo diferencialni del). Tako dobimo proporcionalno-diferencialno-integralni regulator ali kratko PID regulator. Ta se v praksi največ uporablja, ker je razmeroma enostaven in učinkovit za splošno-namensko enozančno regulacijo. Čeprav se delovanje digitalnega in analognega regulatorja bistveno razlikuje, pa se lahko porabi tudi pri obravnavi digitalnega regulatorja klasično teorijo.

Analogni PID regulator je opisan z diferencialno enačbo(4):

$$u_{(t)} = k_p \epsilon_{(t)} + k_i \int \epsilon_{(t)} dt + k_d \frac{d\epsilon_{(t)}}{dt} \qquad (4)$$

kjer so k_p, k_i in k_d proporcionalne konstante P, I in D dela PID regulatorja. Enačba (4) ni ugodna za računalniško obdelavo. Ker sistema ne moremo vedno opazovati, je potrebno preiti iz časovnega prostora v z-prostor. Prav tako pa moramo tudi odvod in integral aproksimirati s pomočjo numeričnih metod v razlike in vsote.

Relativna napaka zaradi te poenostavitve pa z naraščajočim razmerjem τ/T hitro upada in že pri razmerju od 5 do 10 pade v velikostni razred uporabljene aritmetike s plavajočo vejico. Ob upoštevanju te poenostavitve lahko enačbo (4) zapišemo v naslednji obliki:

$$U_n = k_p E_n + k_i h \sum_{i=o}^{n} E_i + \frac{k_d}{h} (E_n - E_{n-1}) . \qquad (5)$$

Če pišemo interval h kot časovno konstanto T oziroma vzorčevalni čas, k_i kot $1/T_i$ in je T_i integracijska časovna konstanta, k_p postavimo pred oklepaj kot K_c in določa ojačenje regulatorja, se enačba (5) spremeni v enačbo (6); kar je dobro znana enačba pozicijskega PID algoritma.

$$U_n = K_c [E_n + \frac{T}{T_i} \sum_{i=o}^{n} E_i + \frac{T_d}{T} (E_n - E_{n-1})] \qquad (6)$$

Ta enačba je v bistvu zelo podobna izrazom z z-transformacijo, še bolj pa je z nja smiselno povezana. Natančnejša analiza takega vzorčevalnega algoritma poteka v z-prostoru, kjer tudi rešujemo stabilnostne probleme. Poleg pozicijskega PID algoritma je v literaturi pogost tudi inkrementalni algoritem.

$$U_n = K_c [E_n - E_{n-1} + \frac{T}{T_i} E_n + \frac{T_d}{T} (E_n - 2E_{n-1} + E_{n-2})] \qquad (7)$$

Iz literature je znan tudi počasni vzorčevalni algoritem razvit po Moslerju, ki ima naslednjo obliko:

$$D(z) = \frac{P_o - P_1 z^{-1}}{(1 - z^{-1})(1 + P_2 z^{-1})} \qquad (8)$$

kjer pomeni: $D_{(z)}$ ... diskretna regulacijska prenosna funkcija in

P_0, P_1, P_2 ... nastavljivi parametri.

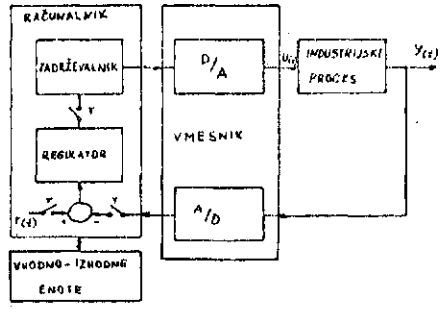
Enačba (8) se v časovnem prostoru lahko zapiše kot:

$$U_n = P_o E_n - P_1 E_{n-1} + (1 - P_2) U_{n-1} + P_2 U_{n-2} . \qquad (9)$$

Iz enačbe (9) je razvidno, da je izhodna spremenljivka odvisna od treh predhodnih stanj podobno kot PID algoritem in to odvisnost določajo trije parametri; odtod ime 3P algoritem. Če je proces aproksimiran s sistemom 1. reda in zakasnitvijo Θ, vhodna funkcija pa je stopničasta, daje ta algoritem najboljše rezultate pri $T : \tau \approx \Theta : \tau$ in je τ časovna konstanta procesa. Važno je torej le, da je čas vzorčenja približno enak mrtvemu času sistema.

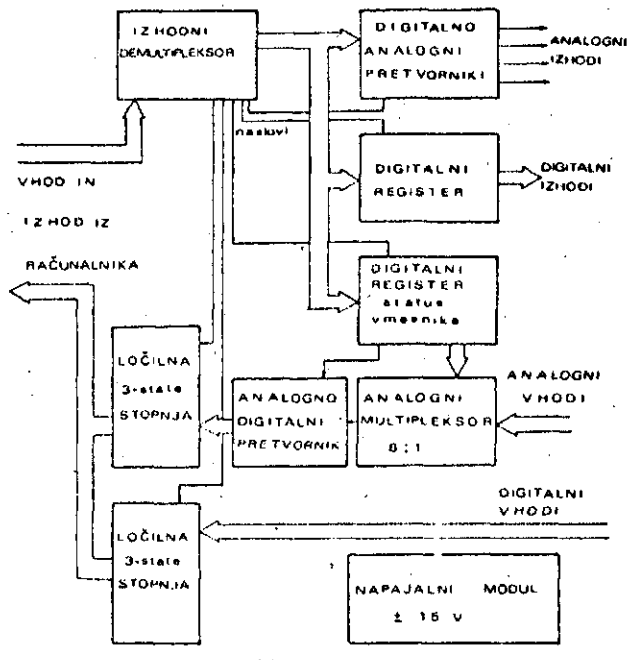
INSTRUMENTALNA OPREMA

Voden industrijski proces je opazovan z mikroračunalnikom preko analogno-digitalnega pretvornika in senzorja. Krmilni podatek pa dobi proces preko digitalno-analognega pretvornika in dajalnika. V realiziranem sistemu je lahko izhodna napetost iz modela v nasprotnem območju od 0 do 10 V, vhodna napetost v model pa v napetostnem območju od -5 do +5V. Osnovno bločno shemo realiziranega mikroračunalniškega regulatorja s stališča vzorčenja podaja sl. 1.



Slika 1.

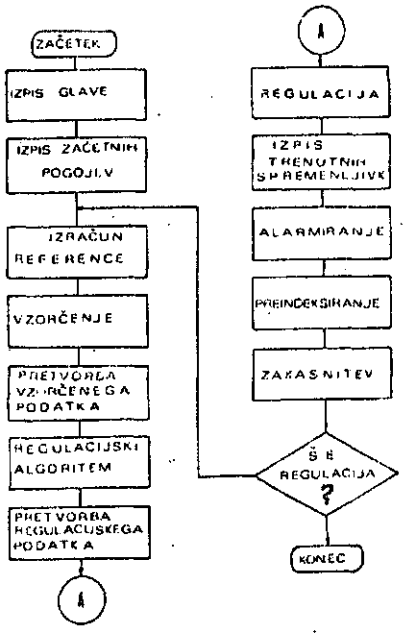
Za računalnik na sliki 1 je uporabljen mikroračunalnik INTEL 8080. Podatki zanj se nahajajo v literaturi (2). Bločno shemo vmesnika prikazuje slika 2.



Slika 2.

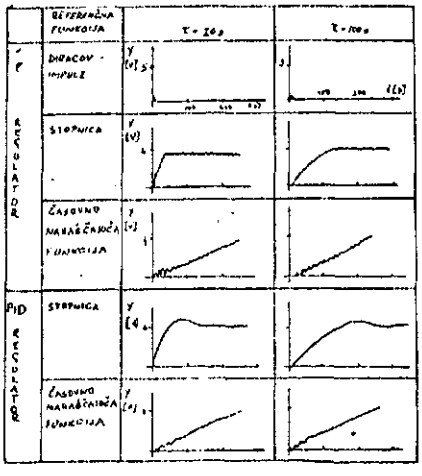
REZULTATI UPORABE MIKRORAČUNALNIŠKEGA REGULATORJA NA SISTEMIH NIŽJEGA REDA (0..,1.)

Okviren program, ki smo ga razvili za regulacijo industrijskega procesa zelo splošno podaja slika 3. Tu je možno poleg izpisov, alarmiranja in zakasnitve tudi vstavljanje različnih regulacijskih algoritmov.



Slika 3.

Odzivi laboratorijskega procesa so prikazani na sliki 4. Referenčna funkcija, ki jo generira mikroračunalnik je Diracov impulz, stopnica in časovno naraščajoča funkcija. Industrijski proces je predstavljen kar s pasivnim sistemom prvega reda s prenosno funkcijo $0,8/(1+s\tau)$, kjer je $\tau = 20$ in 100s. Vzorčevalna perioda je v primeru izpisa posameznih regulacijskih funkcij 5s, brez izpisovanja pa je lahko le 30 ms. Konstante za P in PID regulator so izbrane poljubno, ne pa optimalno glede na reguliran proces, kajti trenutno znane in uporabne metode dajejo postopke za izračun konstant analognih regulatorjev, za diskretne, pa je potrebno poiskati metodo s katero bi določili.



Slika 4.

Primer mikroračunalniškega izpisa pomembnih funkcij pri regulaciji prikazuje tabela 1. Tabela podaja korak, referenčno funkcijo, ki jo generira mikroračunalnik, vzorčeno izhodno funkcijo Y, izračunani pogrešek e in po izbranem regulacijskem algoritmu izračunana regulacijska funkcija U, ki je vhod v regulirani sistem. Izpis je narejen za časovno naraščajočo referenčno funkcijo, za model sistema prvega reda s časovno konstanto $\tau = 20$s in proporcionalni regulator z ojačenjem 10.

TABELA 1.

KORAK	REFERENCA	IZHOD Y	EPSILON	VHOD U
00	+0.000 EXP-00	+0.000 EXP-00	+0.000 EXP-00	+0.000 EXP-00
01	+1.250 EXP-01	+7.812 EXP-02	+4.687 EXP-02	+4.687 EXP-01
02	+2.500 EXP-01	+7.812 EXP-02	+1.718 EXP-01	+1.718 EXP+00
03	+3.750 EXP-01	+6.250 EXP-01	-2.500 EXP-01	-2.500 EXP+00
04	+5.000 EXP-01	+7.812 EXP-02	+4.218 EXP-01	+4.218 EXP+00
05	+6.250 EXP-01	+1.015 EXP+00	-3.906 EXP-01	-3.906 EXP+00
06	+7.500 EXP-01	+7.812 EXP-02	+6.718 EXP-01	+6.718 EXP+00
07	+8.750 EXP-01	+1.171 EXP+00	-2.968 EXP-01	-2.968 EXP+00
08	+1.000 EXP+00	+7.812 EXP-02	+9.218 EXP-01	+9.218 EXP+00
09	+1.125 EXP+00	+1.406 EXP+00	-2.812 EXP-01	-2.812 EXP+00
10	+1.250 EXP+00	+3.125 EXP-01	+9.375 EXP-01	+9.375 EXP+00
11	+1.375 EXP+00	+1.562 EXP+00	-1.875 EXP-01	-1.875 EXP+00
12	+1.500 EXP+00	+7.031 EXP-01	+7.968 EXP-01	+7.968 EXP+00
13	+1.625 EXP+00	+1.796 EXP+00	-1.718 EXP-01	-1.718 EXP+00
14	+1.750 EXP+00	+9.375 EXP-01	+8.125 EXP-01	+8.125 EXP+00
15	+1.875 EXP+00	+1.992 EXP+00	-1.171 EXP-01	-1.171 EXP+00
16	+2.000 EXP+00	+1.171 EXP+00	+8.281 EXP-01	+8.281 EXP+00
17	+2.125 EXP+00	+2.148 EXP+00	-2.343 EXP-02	-2.343 EXP-01
18	+2.250 EXP+00	+1.601 EXP+00	+6.484 EXP-01	+6.484 EXP+00
19	+2.375 EXP+00	+2.382 EXP+00	-7.812 EXP-03	-7.812 EXP-02
20	+2.500 EXP+00	+1.835 EXP+00	+6.640 EXP-01	+6.640 EXP+00
21	+2.625 EXP+00	+2.617 EXP+00	+7.812 EXP-03	+7.812 EXP-02
22	+2.750 EXP+00	+1.953 EXP+00	+7.968 EXP-01	+7.968 EXP+00
23	+2.875 EXP+00	+2.695 EXP+00	+1.796 EXP-01	+1.796 EXP+00
24	+3.000 EXP+00	+2.421 EXP+00	+5.781 EXP-01	+5.781 EXP+00
25	+3.125 EXP+00	+3.007 EXP+00	+1.171 EXP-01	+1.171 EXP+00
26	+3.250 EXP+00	+2.539 EXP+00	+7.109 EXP-01	+7.109 EXP+00
27	+3.375 EXP+00	+3.085 EXP+00	+2.890 EXP-01	+2.890 EXP+00
28	+3.500 EXP+00	+3.085 EXP+00	+4.140 EXP-01	+4.140 EXP+00
29	+3.625 EXP+00	+3.085 EXP+00	+5.390 EXP-01	+5.390 EXP+00
30	+3.750 EXP+00	+3.554 EXP+00	+1.953 EXP-01	+1.953 EXP+00
31	+3.875 EXP+00	+3.085 EXP+00	+7.890 EXP-01	+7.890 EXP+00
32	+4.000 EXP+00	+3.554 EXP+00	+4.453 EXP-01	+4.453 EXP+00
33	+4.125 EXP+00	+3.671 EXP+00	+4.531 EXP-01	+4.531 EXP+00
34	+4.250 EXP+00	+3.945 EXP+00	+3.046 EXP-01	+3.046 EXP+00
35	+4.375 EXP+00	+3.710 EXP+00	+6.640 EXP-01	+6.640 EXP+00
36	+4.500 EXP+00	+3.945 EXP+00	+5.546 EXP-01	+5.546 EXP+00
37	+4.625 EXP+00	+3.867 EXP+00	+7.578 EXP-01	+7.578 EXP+00
38	+4.750 EXP+00	+4.257 EXP+00	+4.921 EXP-01	+4.921 EXP+00
39	+4.875 EXP+00	+4.414 EXP+00	+4.609 EXP-01	+4.609 EXP+00
40	+5.000 EXP+00	+4.492 EXP+00	+5.078 EXP-01	+5.078 EXP+00
41	+5.125 EXP+00	+4.570 EXP+00	+5.546 EXP-01	+5.546 EXP+00
42	+5.250 EXP+00	+4.648 EXP+00	+6.015 EXP-01	+6.015 EXP+00
43	+5.375 EXP+00	+4.687 EXP+00	+6.875 EXP-01	+6.875 EXP+00
44	+5.500 EXP+00	+4.726 EXP+00	+7.734 EXP-01	+7.734 EXP+00
45	+5.625 EXP+00	+4.804 EXP+00	+8.203 EXP-01	+8.203 EXP+00
46	+5.750 EXP+00	+4.648 EXP+00	+1.101 EXP+00	+1.101 EXP+01
47	+5.875 EXP+00	+4.804 EXP+00	+1.070 EXP+00	+1.070 EXP+01
48	+6.000 EXP+00	+4.804 EXP+00	+1.195 EXP+00	+1.195 EXP+01
49	+6.125 EXP+00	+4.804 EXP+00	+1.320 EXP+00	+1.320 EXP+01
50	+6.250 EXP+00	+4.648 EXP+00	+1.601 EXP+00	+1.601 EXP+01
51	+6.375 EXP+00	+4.882 EXP+00	+1.492 EXP+00	+1.492 EXP+01
52	+6.500 EXP+00	+4.648 EXP+00	+1.851 EXP+00	+1.851 EXP+01
53	+6.625 EXP+00	+4.882 EXP+00	+1.742 EXP+00	+1.742 EXP+01

PRIMERJAVA Z ZNANIMI TOVRSTNIMI REGULATORJI

V literaturi (1) obravnavani miniračunalniški regulator smo primerjali z mikroračunalniškim regulatorjem v laboratorijski izvedbi. Miniračunalniški regulator je bil programiran v jeziku SAATO-FOCAL na procesnem miniračunalniku PDP-8/E. Porabljen je bil 16 k pomnilnik pri 12 bitnih besedah. Mikroračunalniški regulator je bil programiran v zbirnem jeziku, porabljen pa je bil 2 k pomnilnik pri 8 bitnih besedah. Iz tega je razvidno, da so bile raziskave z mikroračunalniškim regulatorjem opravičene. To dejstvo potrjuje tudi tabela, ki prikazuje lastnosti posameznih regulatorjev. Iz te je razvidno, da lahko mikroračunalniški regulator nadomesti miniračunalniškega, le da je programiranje prvega dosti težje. V prid mikroračunalniškemu regulatorju, govore tudi vsi ekonomski faktorji.

V tabeli 2 je podan porabljen računalniški (CPU) čas za opisane algoritme realizirane na miniračunalniku.

TABELA 2.

DDC algoritem	Roč. (CPU) čas. (ms)
PID	74
3 P	62
Mrtva cona	111

Pomembne podatke (čas vzpona, faktor dušenja in umiritveni čas) za odziv sistema, ki je reguliran z opisanimi DDC

algoritmi na miniračunalniku podaja tabela 3. Uporabljena je stopničasta referenčna funkcija.

TABELA 3.

DDC algoritem	Čas vzpona (s)	Faktor dušenja	Umiritveni čas ±1% (s)
PI	54	0,03	189
PID	45	0,26	150
Mrtva cona	39	0,33	270

Računalniški (CPU) čas porabljen za različne operacije je podan v tabeli 4. V primerjavi s tabelo 2 je čas porabljen za PID algoritem na mikroračunalniku krajši, kot čas porabljen na miniračunalniku. Če bi realizirali Dahlinov algoritem na mikroračunalniškemu sistemu, bi bil potreben čas maksimalno 120 ms, kar daje prednost temu sistemu glede časovne izkoriščenosti. Ta velika razlika v računalniškem času in porabljenem pomnilniku nastane v glavnem zaradi uporabljenega programiranega jezika FOCAL na miniračunalniku in zbirnega jezika na mikroračunalniku.

TABELA 4.

Opravljena funkcija	Čas
Osnovne matematične ali pretvorniške operacije v aritmetiki s plavajočo vejico	max 2 ms
Vzorčenje	max 1,8 ms
P algoritem	max 5 ms
PID algoritem	max 25 ms
Izpis ene vrstice na TTY	max 5 s
Korak regulacije brez izpisa in brez algoritma	25 ms
Osnovni računalniški strojni cikel	1,9 us
Najkrajše mikroračunalniške inštrukcije	8 us

SKLEP

Začetno delo je bila realizacija aritmetike s plavajočo vejico, katera dosega natančnost 0,01 %. Izkazalo se je, da je ta aritmetika predobra, kadar v regulacijskem algoritmu nastopa malo aritmetičnih operacij kot n.pr. pri P regulatorju. Ker pa se relativna napaka pri računskih operacijah sešteva ali celo množi, bi pri daljšem regulacijskem algoritmu (n.pr. pri Dahlinovem) dobili natančnost le nekaj pod 1 %, kar pa je že v območju natančnosti digitalno-analognega pretvornika. Ključni problem glede natančnosti predstavlja analogno-digitalni pretvornik. Ta razpolaga le z 256-timi kvantizacijskimi nivoji, in ga kaže v bodoče izboljšati.

Bistven podatek o reguliranem sistemu je zgornja frekvenca sistema, ki je še pomembna. Po enačbi (1) je določen minimalni čas vzorčenja. Pri tem sistemu je čas enega regulacijskega koraka velikostnega reda 50 ms. Kadar je torej v reguliranem sistemu 10 Hz še pomembna frekvenca, tedaj bo računalnik zaseden le z opravilom regulacije. Seveda so v industrijskih procesih znatno nižje, v takem primeru pa lahko računalnik opravlja še razne druge funkcije. Ker imamo realiziran program vodenja industrijskega procesa s PID regulacijskim algoritmom, bi računalnik izredno ekonomično vodil hkrati več približno enakih industrijskih procesov.

Narejeni program za P regulator seveda dopušča stacionarno napako, kar je slabost vseh P regulatorjev. Izboljšava tega regulatorja, ki je možna le na računalniku, bi bila, da bi računalnik spoznal stacionarno napako. To vrednost bi odštel od dejanske reference in tako izračunal krmilno funkcijo za regulirani sistem. Posledica tega bi bil P regulator brez stacionarne napake. Sistem bi pri spremenjeni referenci moral ponovno preiti na osnovni koncept regulacije.

Glede natančnosti je treba omeniti, da realizirani regulator ni natančnejši od klasičnega analognega regulatorja. Temu je v prvi vrsti vzrok v pretvorniku. Če bi bil pretvornik bistveno boljši, pa bi se pojavilo vprašanje upravičenosti poenostavitve integriranja in odvajanja kakor tudi celotne aritmetike. Seveda je potrebno vedno poiskati kompromis med zahtevano natančnostjo, računalniškim časom in obširnostjo programa.

Ob zaključku ugotavljamo, da so rezultati dobri. Poleg bistvene razlike v ceni med miniračunalnikom in mikroračunalnikom so tudi časovne razmere in porabljen pomnilnik v mikroračunalniku proti miniračunalniku zelo ugodne, če ne celo odlične. Na istem pripravljenem konceptu bi bilo potrebno testirati še ostale pogostokrat uporabljene pregulacijske algoritme in podati celotno analizo rezultatov. Seveda je nujno, da se izvajalcu omogoči tako delo z mikroračunalnikom kot tudi simulacija procesa na analognem računalniku in realizacijo celotnega provezanega sistema. Ker kaže sistem pri odprti zanki zelo dobre izglede, analize na zaprtozančnem sistemu pa niso bile izvedene v celoti, bi bilo nujno potrebno izvesti zaprtozančno regulacijo za sistem prvega, drugega in višjih redov ter določiti optimalne konstante za posamezen tip regulatorjev.

LITERATURA

1. Paso Uronen, Leena Ylinieni: Department of Process Engineering, University of Oulu, Laboratory of Process Control, Finland, 1977.
2. Control Logic Octal Debugging Technique, Instruction Book M Series Copyright Control Logic 1974.

structuration d'un systeme téléphonique informatisé

*m.exel
m.mekinda
b.popovič

*Institut J.Stefan, Jamova 39, 61000 Ljubljana
Iskra, ATC Labore, Savska Loka 4, 64000 Kranj

UDK 621.395.345:681.3.06

Dans le présent article nous appliquons les principes de la programmation structurée et (pseudo)parallèle pour analyser et restructurer - au moyen d'un langage de haut niveau utilisant le concept de moniteur - un système téléphonique informatisé concret.

Članek opisuje aplikacijo principov strukturiranega in (navidezno) paralelnega programiranja. Predmet aplikacije je analiza in strukturiranje konkretnega programskega sistema telefonske centrale s pomočjo visoko nivojskega programskega jezika, ki uporablja monitorski koncept.

I. INTRODUCTION

Nous nous proposons ici d'analyser et de restructurer un système téléphonique informatisé concret en nous appuyant sur les concepts de programmation structurée et de programmation (pseudo-)parallèle. Ceci n'est pas une étude théorique; nous essayons simplement d'examiner l'applicabilité des concepts mentionnés à un système en temps réel: le systeme téléphonique considéré. La parution récente de quelques articles (5,14) traitant de la programmation en temps réel en liaison avec les méthodes de structuration des systèmes d'exploitation et avec les langages pour la programmation parallèle nous a encouragé à procéder de la sorte.

Les motivations de cette étude sont pratiques: il s'agit de la maintenance, de l'adaptation et des modifications du système informatisé des centraux téléphoniques du type Metaconta 10C Local, réalisé avec les ordinateurs ITT-1600 et 3200. Le logiciel du système est programmé en assembleur et toute modification est difficile. Ceci nous a amené à étudier une restructuration possible du systeme qui permette:
- des modifications aisées et facilement vérifiables;
- une compréhension meilleure du système et, éventuellement,
- une amélioration du degré de parallélisme (potentiel) des différents éléments du système.

Nous présentons dans les chapitres suivants une analyse et une structuration de ce système téléphonique en termes de *processus coopérants, une description de cette structuration* dans un langage qui est pratiquement le Pascal parallèle (2,16) et, finalement, nous discutons des problèmes que pose l'organisation de cette structuration.

II. PRÉSENTATION ET ANALYSE DU SYSTEME

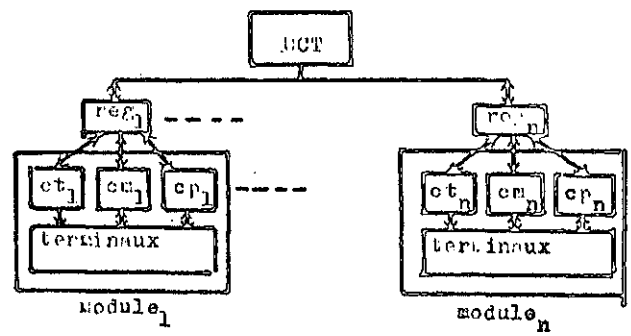
1. Présentation schématique du système

Le système du central téléphonique considéré comprend le matériel suivant: un ou deux ordinateurs ITT-1600 et une périphérie téléphonique comprenant des terminaux d'entrées-sorties, des terminaux de liaison, des éléments de liaison entre ces terminaux et des circuits de tests, de marquage et de pilotage dirigeant ces terminaux. Les terminaux d'E/S sont reliés par des canaux à l'environnement que composent des appareils téléphoniques, d'autres centraux etc...

L'organisation physique de la périphérie téléphonique est modulaire: cette périphérie se compose d'un module de signalisation et au plus de dix modules périphériques. Un module périphérique est composé d'un certain nombre de terminaux (d'E/S et de liaison) et d'éléments de liaison que dirigent un circuit de marquage, trois circuits de tests de types différents et jusqu'à trois circuits de pilotage de types lent, normal et rapide. Nous ne tiendrons pas compte dans la suite de la périphérie classique des ordinateurs.

Fonctionnellement nous distinguons dans ce système du central deux sous-systèmes: le système de contrôle réalisé sur les ordinateurs et le système de coopération avec l'environnement *réalisé par la périphérie téléphonique*. Le système de contrôle contrôle le système de coopération et traite les messages que ce dernier lui communique alors que le système de coopération détecte les signaux provenant de l'environnement, envoie des signaux à ce dernier et établit des liaisons entre les canaux connectés à l'environnement.

Dans la suite nous considérerons un système simplifié comprenant une seule unité centrale de traitement et n modules périphériques dont chacun, relié à l'UCT par un registre périphérique, contient un circuit de tests, un circuit de marquage et un circuit de pilotage de type normal. Un schéma partiel de ce système est présenté dans la figure 1.



ct: circuit de tests
cm: circuit de marquage
cp: circuit de pilotage normal

Figure 1.

2. Processus du systeme: identification et coopération

Nous analysons ici les deux systèmes de contrôle et de coopération en termes de processeurs et de processus séquentiels permanents et coopérants.

Le système du central comprend un processeur principal (l'UCT), un processeur d'horloge à période de 10 msec et les trois fois n processeurs périphériques de tests, de marquage et de pilotage qui sont constitués des circuits correspondants des n modules périphériques et des terminaux de la périphérie téléphonique multiplexés sur ces circuits.

Les processus du système de contrôle se déroulent sur le processeur principal et sur le processeur d'horloge et sont appelés processus internes alors que les processus du système de coopération se deroulent sur les processeurs périphériques de tests, de marquage et de pilotage et sont appelés processus externes (voir la terminologie de (1)) de même nom, respectivement.

Les processus externes de tests réalisent la détection - par l'examen des points de tests des terminaux - des signaux parvenant de l'environnement; les processus externes de pilotage réalisent l'envoi des signaux à l'environnement par l'établissement des liaisons entre les terminaux et les canaux de l'environnement; enfin, les processus externes de marquage réalisent des liaisons entre les terminaux. Il y a quatre groupes de processus internes: les processus internes de tests, de traitement, de marquage et de pilotage, dénotés de facon générique par pt, ptr, pm et pp, respectivement.

Les fonctions de ces processus sont les suivantes: un processus pt détermine un ensemble de points de tests et active un processus externe de tests sur un processeur périphérique donné en lui envoyant une commande par le registre périphérique associé au processeur. Le processus externe prépare sa "réponse" dans le meme registre périphérique qui doit rester occupé par ce processus pendant son activité. Le processus pt analyse ensuite la réponse, envoie un message par l'intermédiaire d'une mémoire-tampon à un processus ptr couplé a pt et reprend la procédure au début jusqu'à ce que l'ensemble de points de tests soit traité. Les processus internes de tests sont activés périodiquement par l'interruption de l'horloge et les différents processus pt se déroulent alors séquentiellement.

Un processus ptr analyse les messages reçus de pt et envoie, en fonction du message reçu, un message à un processus pm ou pp par l'intermédiaire d'une mémoire-tampon ce qui entraîne une interruption activant pp ou pm.

Un processus pp ou pm reçoit des messages des processus ptr et active un processus externe de pilotage ou de marquage, en lui envoyant une commande par le registre périphérique. Le processus externe activé occupe le registre tant que la commande n'est par lue et indique la fin de son activité par une interruption ce qui permet de continuer le processus interne qui l'a activé. Le processus externe n'envoie pas d'autre "réponse".

Nous voyons donc qu'il existe une communication entre les processus internes et externes et entre les différents processus internes. Cette communication se fait par les registres périphériques et les mémoires tampons; ces éléments représentent des ressources partagées que les processus devrons allouer et déallouer dans notre description du système. D'autres ressources partagées sont les processeurs périphériques qui devront être alloués avant l'activation des processus externes se déroulant sur ces processeurs.

L'occupation exclusive des ressources partagées est assurée dans le système actuel par l'exécution sérielle des processus internes. La communication entre les processus du système actuel n'est ni sûre (des messages peuvent se perdre) ni suffisamment souple pour permettre d'exploiter le parallélisme potentiel des processus du système. Nous proposons donc une restructuration du système exposée ci-dessous.

3. Structuration du système

Le système est envisagé comme un ensemble de processus internes séquentiels et permanents (cyclant indéfiniment) dont la coopération est assurée par un ensemble de moniteurs (pour le concept de moniteur voir 1,2,3,4,7,11,12, 13,16), disposés en hierarchie et gérant les ressources du système.

Le noyau du système n'est pas explicité; nous supposons qu'il contient essentiellement un programme d'ordonnancement ("scheduler - dispatcher") des processus internes, les routines pour le traitement des interruptions et deux primitives de communication entre les processus: "bloquer" et "débloquer". Ces deux primitives sont appelées dans les moniteurs et appellent à leur tour le programme d'ordonnancement. Nous supposons un ordonnancement tres simple des processus internes: absence de priorités et stratégie "premier arrivé - premier servi". Notons que la procédure "débloquer" ne provoque pas nécessairement l'activation immédiate du processus débloqué éventuel (nous ne suivons donc pas la stratégie de Hoare et de Brinch Hansen - cf. 2,7 - mais celle, plus générale, exposée dans 18).

Dans le système observé n'existent que deux types d'interruptions: l'interruption de l'horloge et les interruptions provoquées par les processeurs de marquage et de pilotage. Ces interruptions sont "interceptées" par le noyau (qui active par la suite le programme d'ordonnancement) et finalement "acheminées" aux procédures adéquates des moniteurs gérant les processeurs de marquage, de pilotage et l'horloge.

L'occupation exclusive des ressources partagées est assurée par l'inhibition des interruptions dans les procédures des moniteurs ce qui a pour effet l'exclusion "globale" (cf. 11) de ces procédures.

Le système proposé se compose d'éléments suivants:

- processus internes:
- les processus de tests, notés pt_i, $i = 1..t$
- les processus de traitement, notés ptr_i, $i = 1..t$
- les processus de marquage, notés pm_i, $i = 1..m$
- les processus de pilotage, notés pp_i, $i = 1..p$;
- moniteurs:
- les moniteurs des processeurs de tests, notés mt_i, $i = 1..n$
- les moniteurs des processeurs de marquage, notés mm_i, $i = 1..n$
- les moniteurs des processeurs de pilotage, notés mp_i, $i=1..n$
- les moniteurs des registres périphériques, notés mr_i, $i=1..n$
- les moniteurs notés $mb1_i$, $i=1..t$, gérant la communication entre les paires des processus pt_i - ptr_i
- les moniteurs notés $mb2_i$, $i=1..m$, gérant la communication entre les processus ptr et les processus pm_i
- les moniteurs notés $mb3_i$, $i=1..p$, gérant la communication entre les processus ptr et les processus pp_i et
- le moniteur noté mtemp gérant l'interruption de l'horloge.

L'organisation du système est représentée par le graphe de la figure 2. Sur ce graphe apparaissent aussi le processus d'horloge noté ph et les processus externes de marquage et de pilotage, notés cp_i et cm_i, $i=1..n$. Les arcs du graphe représentent les accès (appels) admis dans le système (une interruption étant ici interprétée comme un appel à la procédure d'interruption - de moniteur - correspondante). L'arc partant d'une accolade (resp. aboutissant à une accolade) représente autant d'arcs partants (resp. aboutissants) qu'il y a de processus ou de moniteurs réunis par l'accolade.

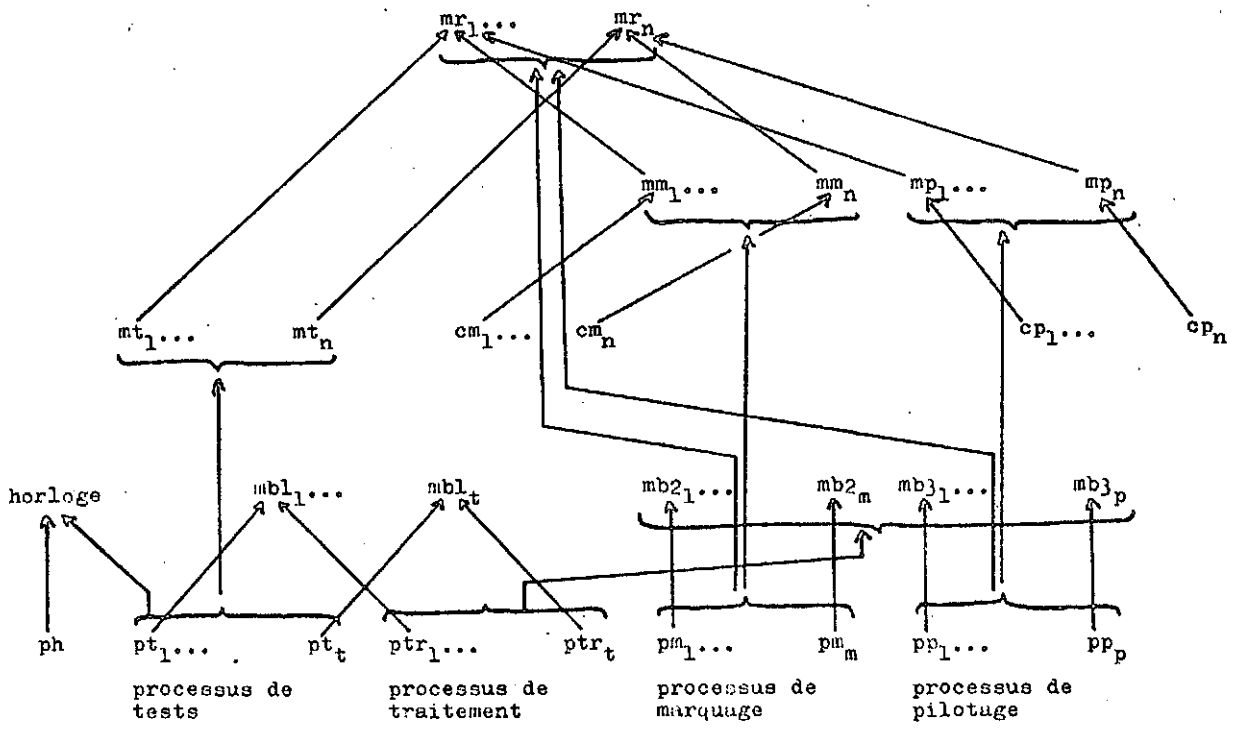


Figure 2.

III. DESCRIPTION DE LA STRUCTURATION PROPOSÉE

Différentes structurations du système sont possibles; elles sont essentiellement fonction du choix des moniteurs. La structure choisie a les effets suivants:

- en allouant d'abord les processeurs de tests, de marquage et de pilotage et ensuite les registres correspondants nous obtenons des files d'attende (sur ces ressources) plus courtes que dans le cas des séquences d'allocation inverses;
- il est possible que les processus externes de marquage et de pilotage d'un même module périphérique se déroulent en parallèle;
- en définissant un moniteur pour chaque ressource nous diminuons le temps passé dans chacune des procédures des moniteurs.

1. Données globales

Ces données globales peuvent être référencées dans tous les moniteurs et dans tous les processus.
Les constantes: t,m,p,n,blon1,blon2,blon3.
Les types:

```
type liste-t = array[1..t] of queue;
     liste-m = array[1..m] of queue;
     liste-p = array[1..p] of queue;
     liste-r = array[1..2] of queue;
     mess1 = { ... }; mess2 = {...}; mess3 = { ... };
     tamp1 = array[1..blon1] of mess1;
     tamp2 = array[1..blon2] of mess2;
     tamp3 = array[1..blon3] of mess3;
type paps = class (long: integer); {cf.(15), p.169: une file
                        "premier arrivé-premier servi"}
     var tête,q,l: integer;
     function entry arrivé: integer;
     begin arrivé:=q; q:=q mod long + 1; l:=l + 1; end;
     function entry départ: integer;
     begin départ:=tête, tête:=tête mod long + 1; l:=l - 1;end;
     function entry vide: boolean;
     begin vide:=(l=0) end;
     function entry plein: boolean;
     begin plein:=(l=long) end;
     begin tête:=1; q:=1; l:=0; end;
```

Les primitives du noyau:

```
procedure bloquer (var q:queue);
  {bloquer le processus courant dans q; quitter le moniteur
   et appeler le programme d'ordonnancement};
procedure débloquer (var q: queue);
  {débloquer le processus dans q; quitter le moniteur et
   appeler le programme d'ordonnancement};
function vide (var q:queue):boolean;
  {vrai si pas de processus en attente dans q}.
```

2. Moniteurs

a) Allocateur de processeurs de tests

```
type mt=
monitor(monreg: mr)
var libre: boolean; liste: liste-t; prochain: paps;
procedure entry allouer;
begin while not libre do bloquer (liste[prochain.arrivé]);
      libre:=false;
      monreg.allouer;
end;
procedure entry déallouer;
begin monreg.déallouer;
      libre:=true;
   if not prochain.vide then débloquer(liste[prochain.départ]);
end;
begin libre:=true; init prochain(t) end;
```

On peut declarer alors n moniteurs du type mt dont chacun est associé à un moniteur du type mr (voir plus bas):

```
var mt1:mt; avec init mt1(mr1);
```

b) Allocateurs de processeurs de marquage et de pilotage

```
type mm=
monitor(monreg: mr)
var libre:boolean; liste:liste-m; prochain:paps;
    inter:boolean; attinter:queue;
procedure entry allouer;
```

```
begin while not libre do bloquer(liste[prochain.arrivé]);
      libre:=false;
      monreg.allouer;
      inter:=false; { le signal d'interruption }
end;
procedure entry déallouer;
begin if not inter then bloquer(attinter);
     {attendre l'interruption }
      libre:=true;
      if not prochain.vide then débloquer(liste[prochain.départ]);
end;
procedure entry minter;{"intercepte" l'interruption du circuit
                        de marquage }
begin inter:=true;
      if not vide(attinter) then débloquer(attinter);
end;
begin libre:=true; inter:=false; init prochain(m) end;
type mp= {analogue au type mm }.
 On pourra déclarer n moniteurs du type mm et n moniteurs
 du type
mp par:
 var mm1:mm;  avec  init mm1(mr1);   etc...
```

c) Allocateur de registre périphérique

```
type mr=
monitor
var libre:boolean;  liste:liste-r;  prochain:paps;
procedure entry allouer; .
begin while not libre do bloquer(liste[prochain.arrivé]);
      libre:=false;
end;
procedure entry déallouer;
begin libre:=true;
      if not prochain.vide then débloquer
      (liste[prochain.départ]);
end;
 On déclarera n moniteurs du type mr par:
var mr1:mr;  etc...
```

d) Allocateur de mémoire-tampon du type mb1

```
type mb1=
monitor
var mb:tamp1; prochain:paps; source, destination:queue;
procedure entry envoyer(m:mess1);
begin if prochain.plein then bloquer(source);
      mb[prochain.arrivé]:=m;
       if not vide(destination) then débloquer(destination);
end;
procedure entry recevoir(var m:mess1);
begin if prochain.vide then bloquer(destination);
      m:= mb[prochain.départ];
      if not vide(source) then débloquer(source);
end;
begin init prochain(blon1) end;
 On déclarera t moniteurs de type mb1: var mb11:mb1; etc...
```

e) Allocateurs des mémoires-tampons du type mb2 et mb3

```
type mb2=
monitor
var mb:tamp2; prochainproc,prochaintamp:paps;
    source:liste-t; destination:queue;
procedure entry envoyer(m:mess2);
begin while prochaintamp.plein do bloquer
      (source[prochainproc.arrivé]);
      mb[prochaintamp.arrivé]:=m;
      if not vide(destination) then débloquer(destination);
end;
procedure entry recevoir(var m:mess2);
begin if prochaintamp.vide then bloquer(destination);
      m:=mb[prochaintamp.départ];
      if not prochainproc.vide then
        debloquer(source[prochainproc.départ]);
end;
begin init prochainproc(t), prochaintamp(blon2)  end;
type mb3 = {analogue au type mb2 }.
```

On déclarera m moniteurs du type mb2 et p moniteurs du type mb3:

```
var mb21:mb2,...   mb31:mb3,...
```

f) Le moniteur de l'horloge

```
type mtemp=
monitor
var attliste:liste-t; {contient les processus pt; qui ont fini leurs
    cycles et attendent l'interruption de l'horloge pour
    recommencer un nouveau cycle}
    prochain:paps;
procedure entry attendre;
begin bloquer(attliste[prochain.arrivé])  end;
procedure entry tempinter;{l'interruption de l'horloge}
begin while not prochain.vide
      do débloquer(attliste[prochain.départ])
end;
begin init prochain(t)  end;
 Déclaration:  var horloge:mtemp;
```

3. Processus

a) Processus de tests

```
type pt=
process(heure:mtemp; buf:mb1; accèst1,{...}, accèstn:mt);
var circuitest:1...n; mess:mess1;
begin
   cycle
   {initialisation:déterminer les tests}
     repeat
     ...
     case circuitest of 1:accèst1.allouer;
    {envoi de la commande au processeur de tests
     correspondant en utilisant le registre correspondant;
     delai "actif"Δ ; examen de la reponse reçue dans le
     registre}
     case circuitest of 1:accèst1.déallouer;
    {preparer le message pour le processus ptr correspondant}
     buf.envoyer(mess);
     until { fin de tests du cycle } ;
   heure.attendre;
   end;
end;
 Les processus de type pt sont déclarés et initialisés par:
var pt1:pt; init pt1(horloge,mb11,mt1,mt2,{...},mtn); etc..
```

b) Processus de traitement

```
type ptr=
process(buf:mb1; tamp1,{...},tampm:mb2 ; tam1,{...},
        tamp:mb3);
var m1:mess1; m2:mess2; m3:mess3;
begin
   cycle   buf.recevoir(m1);
          {traitement du message}
       ...{tamp1.envoyer(m2) ... tam1.envoyer(m3)}...
   end;
end;
 Ces processus sont déclarés et initialisés par:
var ptr1:ptr; init ptr1(mb11,mb21,{...},mb31,{...},mb3p);
```

c) Processus de pilotage et de marquage

```
type pm=
process(buf:mb2; accèsm1,{...}, accèsmn:mn;
                 accèsr1,{...}, accèsrn:mr);
var circuitmarq:1..n; mess:mess2;
begin
 {initialisation}
 cycle buf.recevoir(mess);
  {préparer l'action adéquate}
  case circuitmarq of 1:accèsm1.allouer;
                      ...
   {envoi de la commande au processeur de marquage
    correspondant en utilisant le registre correspondant}
  case circuitmarq of 1:begin accesr1.deallouer;
                              accesm1.déallouer;
                        end;
                        ...
end;  end;
```

Ces processus sont déclarés et initialisés par:
var pm1:pm;
init pm1(mb21,mm1,{...}, mmn,mr1,mr2,{...},mrn); etc...
type pp={ analogue au type pm}.

IV. CONCLUSION

Le fonctionnement correct du système décrit en III (ainsi que celui du système original) est basé sur l'hypothèse suivante: *l'intervalle de temps entre deux interruptions de* l'horloge est sufisamment long pour que, en fin d'intervalle:
- tous les processus pt soient bloqués dans la liste "attliste",
- aucun des messages ne soit perdu et
- *tous les processus externes soient terminés.*

Partant de cette hypothèse de base nous pouvons supposer que le système structuré proposé ici assure un déroulement harmonieux des processus du système c.à.d. sans bloquage des processus (cf. 1).
En effet, d'une part, nous avons dans le système une allocation hiérarchique des ressources permanentes (processeurs périphériques et registres) et, d'autre part, une communication de messages hiérarchique entre les processus internes du système (les processus pt envoient des messages aux processus ptr et ces derniers envoient des messages aux processus pm et pp).

L'implantation du système proposé est basée sur l'hypothèse d'un seul processeur central et est la plus simple possible (les moniteurs sont réalisés par l'inhibition des interruptions). On peut bien évidemment envisager une implantation plus sophistiquée et plus complexe (cf. 17):
- en supposant plusieurs processeurs centraux et/ou
- en introduisant les sémaphores pour assurer une exclusion "locale" des procédures des moniteurs.

Une telle implantation rendrait possible une amélioration théorique du degré de parallélisme des différents processus du système mais - étant donné les temps très courts des cycles des processus - serait probablement prohibitive relativement aux temps d'"accès" aux moniteurs.

Nous pensons que la restructuration proposée du système du central peut, d'une part, servir à modéliser le système et à le simuler et, d'autre part, peut servir comme point de départ d'une configuration nouvelle du système.

BIBLIOGRAPHIE

(1) Brinch Hansen P.: Operating system principles, Prentice-Hall, 1973.
(2) Brinch Hansen P.: The programming language Concurrent Pascal, IEEE trans. on software eng. vol. SE-1, no.2, 1975.
(3) Brinch Hansen P.: A programming methodology for operating system design, IFIP 1974.
(4) Dijkstra E.W.: Hierarchical ordering of sequential processes, Operating systems techniques, Academic Press, 1972.
(5) Gordon R.L.: Systems of cooperating schedulers, IFAC-IFIP workshop on real-time programming, 1975.
(6) Haberman A.N.: Introduction to operating system design, SRA 1976.
(7) Hoare C.A.R.: *Monitors: an operating system structuring* concept, CACM, oct. 1974, p. 549.
(8) Horning J.J., Randell B.: Process structuring, Computing surveys, vol. 5, no.1, 1973.
(9) ITT: *Standard computer modules* ITT 1600, 160 ITT 11000E, 1968.
(10) Jensen K., Wirth N.: Pascal-user manual and report, Springer-Verlag, 1975.
(11) Lister A.M., Maynard K.J.: An implementation of monitors, Software-practice & experience, vol. 6, 377-385, 1976.
(12) Lister A.M., Sayer P.J.: Hierarchical monitors, Proc. 1976 internat. conf. on parallel processing.
(13) Schmid H.A.: On the efficient implementation of conditional critical regions & the construction of monitors, Acta Informatica 6, 1976.
(14) Smedema C.H.: Real-time concepts and Concurrent Pascal, IFAC-IFIP workshop on real-time programming, 1975.
(15) Brinch Hansen P.: The solo operating system: processes, monitors & classes, Software-practice & experience, vol.6, 165-200, 1976.
(16) Brinch Hansen P.: Concurrent Pascal report, Cal.tech., 1975.
(17) Wettstein H.: The implementation of synchronizing operations in various environments, Software-practice & experience, vol.7, 115-126, 1977.
(18) Wettstein H.: The problem of nested monitor calls revisited, Oper. Systems Review, vol.12, no.1, 1978.
(19) Popović B., Exel M., Mekinda M.: Primjena monitorskog koncepta u izgradnji operacionog sistema za periodično aktiviranje programa. J. Informatica, 1977, no. 2.
(20) Mekinda M., Exel M., Popović B.: Strukturiranje programsko vodjenog sistema telefonske centrale, XII. jug. medn. simpozij o obravnavanju podatkov, Bled, oktober 1977, 6-116.

električni stimulator z mikroračunalnikom M 6800

p.šuhel
b.ličar

Fakulteta za elektrotehniko
Univerze v Ljubljani

UDK 61::621.3:681.3-181.4

Delo obravnava primer uporabe mikroračunalnika v sistemu trikanalnega električnega stimulatorja za zdravljenje urinske inkontinence pri človeku. Uporabljen je mikroračunalnik M6800. Obravnavan je problem generiranja funkcijskih oblik pravokotnih stimulacijskih impulzov. Namen študije je ugotoviti prednosti in morebitne pomanjkljivosti stimulatorja z mikroračunalnikom pred stimulatorjem izvedenim z decizijsko logiko. Izdelana je programska oprema za enokanalni in trikanalni sistem, ter predlagana aparaturna oprema.

ELECTRICAL STIMULATOR WITH MICROCOMPUTER M6800. The paper gives an account of microcomputer application in the system of a threechannel electrical stimulator for the treatment of the urinary incontinence in man. The M6800 microcomputer is applied. The problem dealt with is generation of functional forms of rectangular stimulating impulses. The study's intention was to disclose the advantages and possible disadvantages of the stimulator with the microcomputer compared to the stimulator with decision logic. The software for one-channel and three-channel system was designed and the hardware was proposed.

1. UVOD

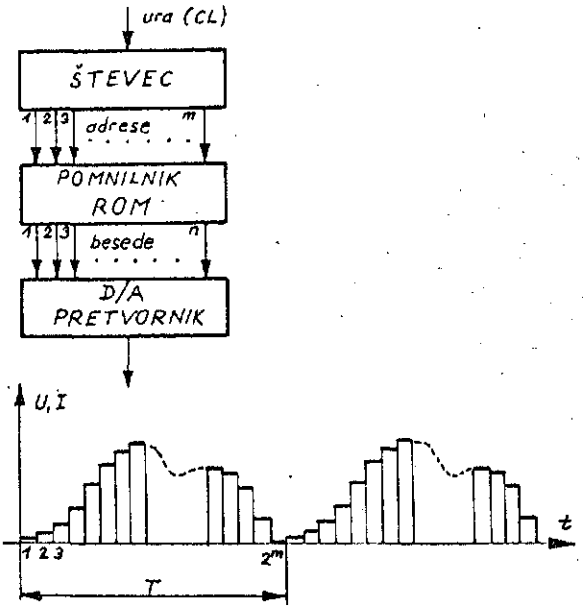
Vedno večje uveljavljanje mikroračunalnika na raznih področjih je privedlo do zamisli izgradnje električnega stimulatorja s pomočjo mikroračunalnika.

S problemi električne stimulacije za zdravljenje urinske inkontinence pri človeku se več ali manj uspešno ukvarjajo tako zdravniki, kot inženirji že desetletja. Nova iskanja narekuje potreba po izboljšavi poznanih sistemov električnih stimulatorjev. Nov sistem naj služi predvsem raziskovalnim namenom, od katerega zahtevamo digitalno nastavljanje parametrov električne stimulacije, ki so: širina in amplituda impulza, ter dolžina pavze. Trikanalni izhod naj bo napetostni ali tokovni vir z napetostno in tokovno omejitvijo. Trikanalni izhod je potreben za hkratno vaginalno stimulacijo, analno stimulacijo in stimulacijo z igelnimi elektrodami. Vsi parametri električne stimulacije morajo biti med seboj funkcijsko neodvisni. Parametri stimulacije naj bodo določeni s programsko opremo.

Celoten sistem električnega stimulatorja lahko razdelimo na dve osnovni enoti:
1) generator stimulacijskih impulzov,
2) izhodna stopnja.
Izhodna stopnja mora ustrezati omenjenim zahtevam, ki jih narekuje fiziološko breme. Generator stimulacijskih impulzov izveden s pomočjo mikroračunalnika je prvi del naloge.

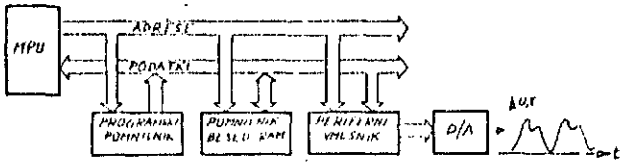
2. MIKROPROCESOR V SISTEMIH SIGNALNIH VIROV

Signalne vire lahko realiziramo s pomočjo analogne ali digitalne tehnike. Izvedba v digitalni tehniki s pomočjo decizijske logike ima prednost pred analogno, ker lahko generiramo poljubno funkcijsko obliko (slika 1). Slaba stran je frekvenčna omejitev izhodnega signala, ki je pogojena s hitrostjo delovanja sestavnih delov, kot so števci, registri, pomnilniki itd.



Slika 1: Digitalni način generiranja funkcijskih oblik

Frekvenco izhodnega signala na sliki 1 spreminjamo s frekvenco ure (CL), ki krmili števec. Funkcijska oblika je določena programsko z vsebino pomnilnika ROM in je ni moč spreminjati. To pomanjkljivost lahko odpravimo z vključitvijo mikroprocesorja v sistem, kot kaže slika 2.
MPU najprej po programu izračunava amplitude in jih po vrsti vpisuje v pomnilnik besed (byte) RAM, kamor mora zapisati eno periodo funkcije. Nato sledi ciklično



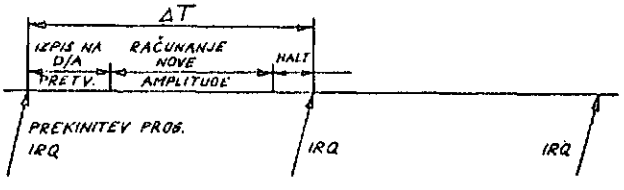
Slika 2: Generiranje funkcijskih oblik z mikroračunalnikom

izpisovanje besed na D/A pretvornik, oziroma generiranje funkcijske oblike. Uporaba mikroprocesorja nudi še dodatno možnost: Amplitude izračunava in jih lahko sproti posreduje na D/A pretvornik. Med obema načinoma je razlika v hitrosti. Čas, potreben za izpis amplitude iz pomnilnika besed na D/A pretvornik je mnogo krajši kot čas, potreben za izračun posamezne amplitude. Tako lahko v prvem primeru dosežemo večjo frekvenco izhodnega signala, v drugem pa nismo omejeni s številom besed za generiranje ene periode, ker amplitude sproti izpisujemo in ne rabimo dodatne pomnilne kapacitete.

Funkcionalna električna stimulacija ima električne impulze pravokotne oblike. s širino impulza T_i, dolžino pavze T_p ter amplitudo U_i. Zahteve so:
- širina impulza T_i = 100 μs - 10 ms
- dolžina pavze T_p = 1 ms - 10 s
- generiranje monofaznih in bifaznih signalov.

Monofazni signal ima impulze iste polaritete, medtem ko si pri bifaznem signalu sledijo impulzi nasprotne polaritete. Napetostna in tokovna omejitev stimulacijskih impulzov v tem delu snovanja programa nista pomembni.

Naloga je, z mikroračunalnikom generirati zahtevane stimulacijske impulze, ki v najneugodnejših razmerah dosežejo T_{imin}= 100 μs, T_{pmax} = 10 s. Razmerje T_{pmax} : T_{imin}= 10^5, kar pomeni, da je potrebno za zapis signala ene periode 10^5 besed. M6800 ima 16-bitno adresno vodilo in lahko adresira 2^{16} = 65 kbytov spominskih lokacij. Le tak način generiranja pride v našem primeru v poštev, ko mikroprocesor izračunane amplitude sproti izpisuje na D/A pretvornik. Tu pa se pojavi problem dinamike mikroprocesorja. Princip generiranja kaže slika 3.



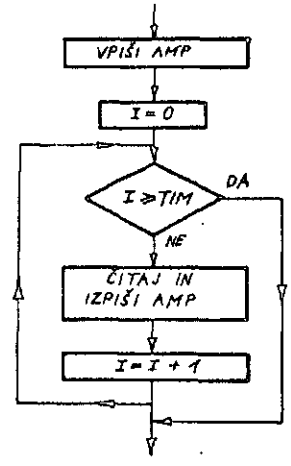
Slika 3: Princip generiranja s prekinitvenim delovanjem

T je časovni interval med dvema izpisoma. Odvisen je od dolžine programa in od uporabljenih instrukcij. Ugotoviti je, ali lahko dosežemo T = 100 μs, kot je zahtevana minimalna širina impulza, oziroma sestaviti program tako, da je T čim krajši.

3. OPIS PROGRAMSKE OPREME

Programska oprema določa tudi generiranje amplitud; tako generiranje impulza kot pavze. Pavza ima amplitudo 0 V, v programu je to opisano z binarno kodo. Če uporabimo D/A pretvornik, ki da na izhodu napetost od 0 do $+U_{max}$ ustreza potencialu 0 V koda 00000000, če pa D/A spreminja svoj izhod od -U do +U ustreza potencialu 0 V koda 01111111. Podatki, ki jih obdeluje M6800 so 8-bitne besede (byte). Z 8-bitno besedo lahko izrazimo cela števila od 0 do 255 v decimalnem zapisu. Le-to pomeni, da lahko kvantiziramo amplitudo za monofazni signal na 0,4% maksimalne vrednosti. Za bifazni signal pa je kvantizacija 1,4%.

Ker generiramo pravokotne impulze, zavzame amplituda preko cele periode le dve oziroma tri vrednosti. Mikroprocesor opravlja le vlogo števca. V ta namen je uvedena zanka in indeks, ki karakterizira širino impulza (TIM) ter dolžino pavze (TP). Med programom se za vsak izpis amplitude poveča pomožni indeks I v indeksnem registru za 1. Ko doseže vrednost pomožnega indeksa vrednost TIM, sledi izstop iz zanke. Program se nadaljuje v zanki za generiranje amplitude pavze, kar se ciklično ponavlja, dokler mikroprocesorja ne resetiramo. Slika 4 prikazuje diagram poteka generiranja impulzov.



Slika 4: Diagram poteka generiranja impulzov

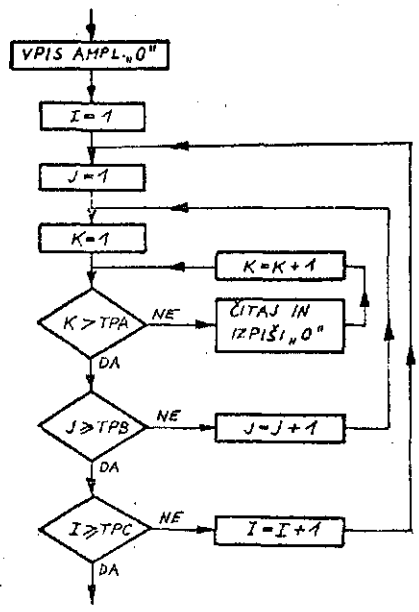
Za generiranje pavze ne zadostuje ena sama zanka. Za dolžino zanke v ΔT = 100 μs je potrebno število izpisov TP = $T_p / \Delta T$.

T_p = 10 s $\qquad$ $TP = \frac{10}{100 . 10^{-6}} = 10^5$

ΔT = 100 μs

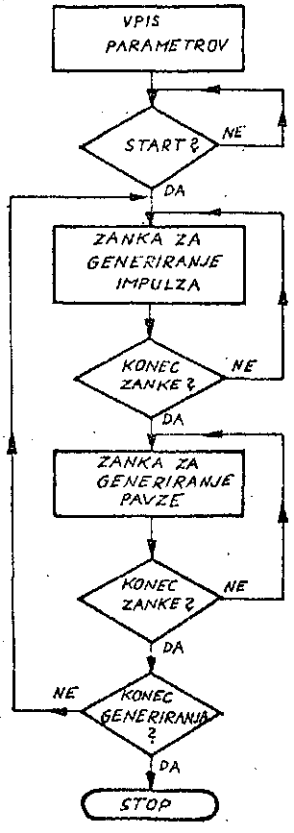
V programu je problem rešen s tremi zankami (slika 5). Posamezne oznake pomenijo:
"O" - amplituda O
TPA - parameter pavze prve zanke
TPB - parameter pavze druge zanke
TPC - parameter pavze tretje zanke
I,J,K - so pomožni indeksi.

Ker se mora notranja zanka izteči, da se pomožni indeks zunanje zanke inkrementira za 1, je parameter za čas pavze TP produkt parametrov TPA, TPB in TPC (TP = TPAXTPBXTPC). Program za enokanalni sistem lahko ponazorimo z diagramom poteka na sliki 6. Program je sestavljen za generiranje monofaznih in bifaznih signalov. Pred začetkom programa po vrsti vpišemo parametre električne stimulacije v spominske lokacije, ki so za to rezervirane. Za bifazni signal vpišemo: AMP1, TIM1, TPA, TPB, TPC (pozitivni impulz



Slika 5: Diagram poteka za generiranje pavze

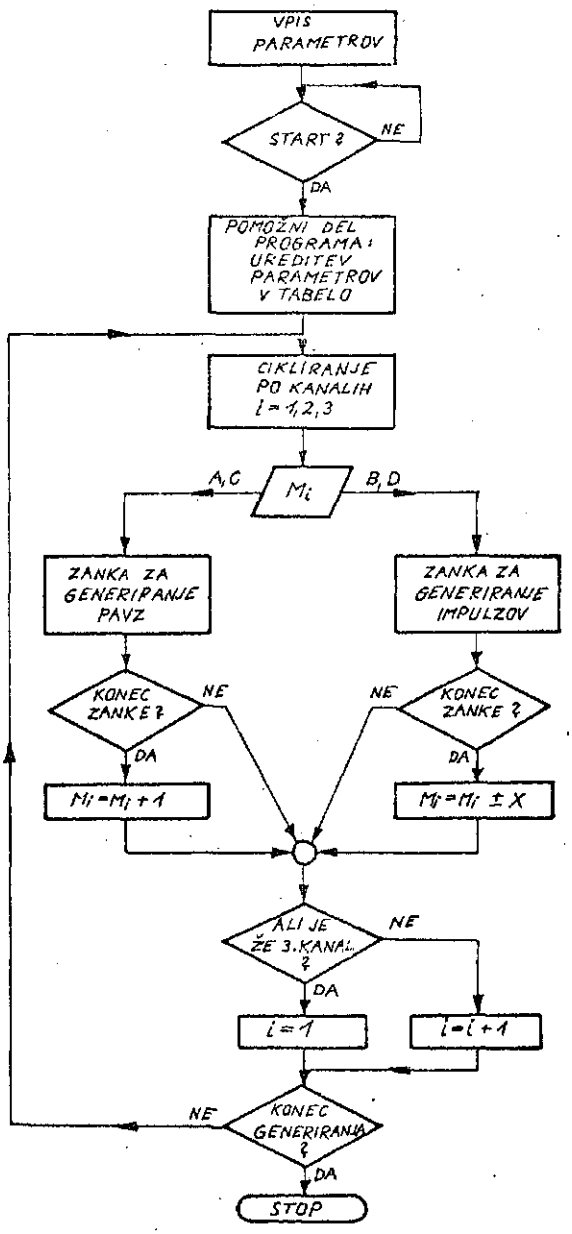
in pavza 1); AMP2, TIM2, TPD, TPE, TPF (negativni impulz in pavza 2). Za monofazni signal vpišemo enako kot za bifaznega, le da velja: AMP1 = AMP2, TIM1 = TIM2, TPA x TPB x TPC = TPD x TPE x TPF. Vsi parametri so 8-bitne besede, oziroma števila od 0 - 255. Širino impulza in dolžino pavze nastavljamo digitalno. Dolžina impulza je: $T_i[s] = TIM \cdot \Delta T[s]$, $T_p[s] = TP \cdot \Delta T[s]$. Za enokanalni sistem je dosežen $\Delta T_{min} = 82\ \mu s$ in je znotraj zahteve $\Delta T_{min} = 100\ \mu s$.



Slika 6: Diagram poteka za enokanalni sistem

Širina impulza T_i je nastavljiva po 100 μs od 100 μs do 25,5 ms in prav tako dolžina pavze, če je to potrebno. V tem primeru je treba računati posamezne parametre TPA, TPB, TPC. Za TP > 255 ne moremo nastavljati T_p po 100 μs za števila, ki so matematično prafaktorji. Seveda se lahko zadovoljimo tudi z bolj grobo kvantizacijo, na prmer tisto, ki je procentualno enaka kvantizaciji širine impulza. V tem primeru sta dva parametra (TPB, TPC) konstantna, T_p pa spreminjamo samo s parametrom TPA. Program za enokanalni sistem zasede 114 bytov spominskih lokacij v RAM-u.

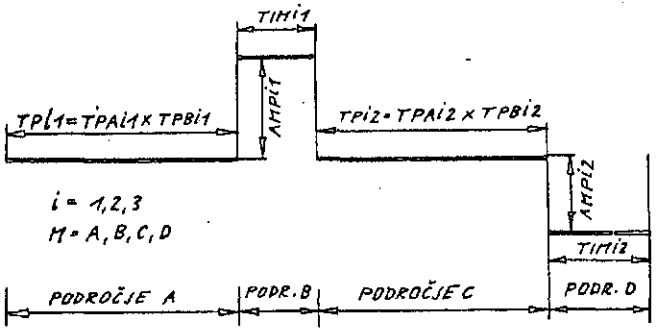
Slika 7 prikazuje diagram poteka za trikanalni sistem. Zahtevamo generiranje treh med seboj neodvisnih časovnih potekov impulzov. Program mora biti sestavljen tako, da MPU zaporedoma izpisuje amplitude za 1., 2. in 3. kanal - multipleksiranje !



Slika 7: Diagram poteka za trikanalni sistem

Pomen posameznih označb in parametrov prikazuje slika 8. M pomeni področje signala. Pred začetkom generiranja vpišemo vse parametre stimulacije po vrsti

v pomnilnik, enako kot za enokanalni sistem. Ker je prisotnih mnogo parametrov, je smotrno, če program razdelimo na dva dela. Pomožni del programa uredi vse vpisane parametre v tabelo in parametre shrani pod druge adrese.



Slika 8: Pomen oznak za trikanalni sistem

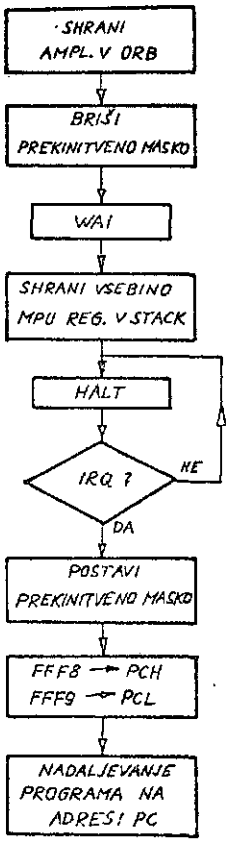
Tabela je sestavljena tako, da je mogoče v teku generiranja čimbolj preprosto indeksno čitati potrebne parametre. Drugi del programa je "glavni del", ki zajema generiranje in izpisovanje amplitud. Če želimo, da ene izmed pavz ni (TP_{ij} = 0), se že v tabeli ustrezno postavi indeks x (slika 7). Za generiranje vseh impulzov je v programu samo ena zanka, ker delamo z indeksnim registrom. Enako velja za generiranje pavz. Program skupno s tabelo in vpisanimi parametri zasede 465 bytov spominskih lokacij. Čas med dvema izpisoma za trikanalni sistem je 176 μs in je zaradi uporabe multipleksiranja ΔT = 3 . 176 μs = 528 μs. Oba programa sta bila sestavljena in preizkušena na mikroračunalniškem sistemu EVK 300.

4. DELOVANJE V REALNEM ČASU

Za delovanje v realnem času je potreben zunanji prekinjevalni signal, ki prekinja glavni program in periferni vmesnik, preko katerega se izpisujejo besede na D/A pretvornik. Uporabljen je periferni vmesnik za paralelen izpis podatkov PIA (Peripheral Interface Adapter). Prekinjevalni signal krmili PIA na enem izmed štirih krmilnih vhodov, ta pa nadzira prekinitveno delovanje mikroprocesorja. Prekinitveno delovanje prikazuje slika 3, diagram poteka prekinitvenega delovanja pa slika 9.

Ko je amplituda shranjena v izhodnem registru PIA-e ORB, se lahko začne prekinitvena sekvenca. Najprej brišemo prekinitveni maskirni bit v pogojno-kodnem registru MPU-ja, da omogočimo zahtevo prekinitve IRQ. Instrukcija WAI poveča vrednost programskega števila za 1, nato shrani vsebine MPU registrov po vrsti v sklad (Stack). MPU sedaj čaka v stanju HALT na prekinitveno zahtevo, ki jo pošlje PIA kot odgovor na prekinjevalni signal CB1. Signal IRQ zopet postavi maskirni bit in s tem onemogoči morebitno novo zahtevo po prekinitvi. Program se nadaljuje na naslovu, ki je shranjen na lokaciji prekinitvenega vektorja IRQ. Na tem naslovu se nahaja instrukcija RTI (Return from Interrupt), ki povzroči, da se vsebine shranjene v skladu vrnejo v registre MPU-ja in program se nadaljuje. Zunanji prekinjevalni signal, ki preko vmesnika PIA prekinja glavni program, ima lahko maksimalno frekvenco $f_{max} = 1/\Delta T$.

Možne širine impulzov so ΔT, $2\Delta T$, $3\Delta T$ ... $255\Delta T$. S spreminjanjem frekvence zunanjega prekinjevalnega signala, je mogoče generirati impulze poljubnih širin



Slika 9: Diagram poteka prekinitvenega delovanja

s pogojem, da te niso manjše od ΔT, oziroma T_{imin}. Frekvenco prekinjevalnega signala lahko spreminjamo s pomočjo sistema fazno-zaključene zanke, ki jo krmili mikroračunalnik. Potrebna je še dodatna programska oprema in program deljenja. Postopek deljenja poteka naslednje ΔT_{min} je določen s programom. Širina impulza je T_i = TIM x ΔT_{min}. V pomnilnik vpišemo kot parameter širino impulza T_i in dolžino pavze T_p.

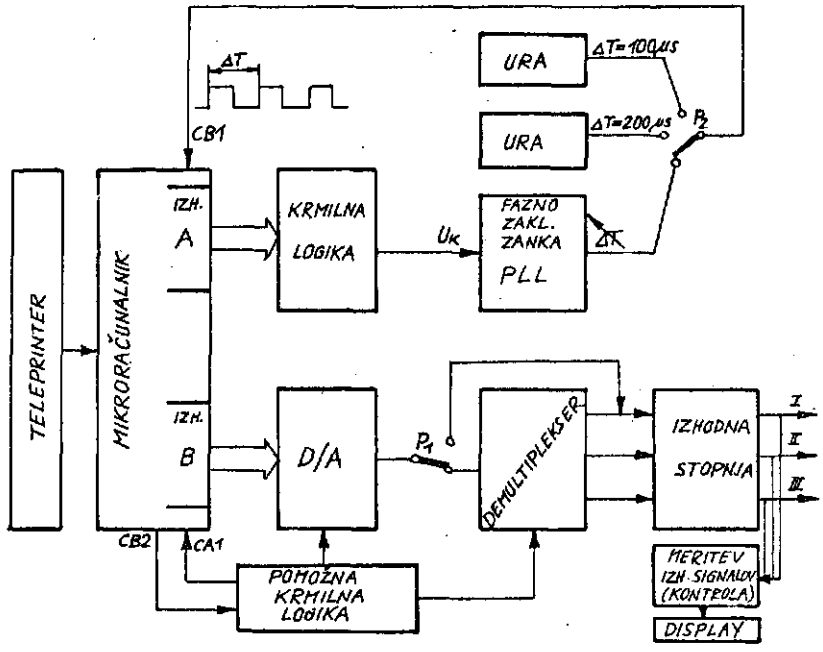
1. korak T_i : ΔT_{min} = TIM′ (pogoj $T_i \geqslant T_{imin}$)
   TIM′ je celoštevilčni rezultat deljenja
2. korak T_i : TIM′ = $\Delta T'$
3. korak T_p : $\Delta T'$ = TP′

TIM′, TP′ sta sedaj nova parametra, ki ju upošteva program. $\Delta T'$ izpiše mikroprocesor v obliki 8-bitne kode na ustrezno periferno krmilno logiko, ki krmili fazno-zaključeno zanko (PLL). PLL generira zunanji prekinjevalni signal s frekvenco $f' = 1/\Delta T'$.

5. APARATURNA OPREMA

Za delovanje stimulatorja je potrebno poleg mikroračunalnika in teleprinterja še druga aparaturna oprema. Za enokanalni sistem je dovolj D/A pretvornik. Za trikanalni sistem potrebujemo še demultiplekser. Potreben je trikanalni analogni demultiplekser. Posledica demultipleksiranja je, da izhodni signali pri trikanalnem sistemu niso v fazi, ampak so med sabo časovno premaknjeni za $\Delta T/3$. Lahko pa dodamo še sistem fazno-zaključene zanke. Blok sheme kompletnega sistema prikazuje slika 10. Na izhodu A (PIA) se izpiše koda za nastavitev frekvence samo enkrat, to je pred začetkom glavnega programa. Krmilna logika mora vsebovati osem pomnilnih celic in D/A pretvornik, ki z napetostjo U_k diktira izhodno frekvenco PLL-ja. S preklopnikom P2 izbiramo različne možnosti:

1) ΔT = 100 μs (enokanalni sistem)



Slika 10: Blok shema stimulatorja

2) $\Delta T = 3 \times 200\,\mu s$ (trikanalni sistem)
3) Spremenljiv časovni interval ΔT.

Če želimo imeti enokanalno stimulacijo, ne potrebujemo demultiplekserja (prekl. P1). Pomožna krmilna logika krmili D/A, demultiplekser in krmilni vhod PIA, CA1.

6. ZAKLJUČEK

Električni stimulator z mikroračunalnikom je le ena od možnih uporab mikroračunalnika. Nakazane so bile le nekatere možnosti za generiranje periodičnih signalov. Omeniti je, da se za stimulacijo uporabljajo tudi signali, kjer so amplitude posameznih impulzov "stohastične", ali pa se med stimulacijo spreminjajo po določeni funkciji. Za stohastične impulze ni treba drugega, kot da s šumnim generatorjem preko A/D pretvornika v določenih intervalih vpisujemo naključno amplitudo v spominsko lokacijo, od koder jo program čita kot parameter za amplitudo stimulacije.

Stimulator upravljamo preko teleprinterja. Za stimulator kot samostojen sistem, bi v ta namen zadostovala heksadecimalna tastatura. Vse programe bi bilo treba zapisati v EPROM-e, preko tastature pa bi vnašali v mikroračunalnik samo glavne podatke za stimulacijo in ukaza start/stop. Smotrna bi bila uporaba mikroračunalniškega sistema na širšem področju urodinamike. Mikroračunalnik bi opravljal različne naloge, kot so npr. merjenje uretralnega profila pritiskov, analizo merilnih rezultatov itd. Generiranje impulzov za električno stimulacijo bi bila samo ena izmed nalog.

7. LITERATURA

1. Lucke G., Mrize J.,P., Carr W.N.: Semiconductor Memory Design and Application, McGraw-Hill, N.Y. 1973

2. McGlynn D.,R.: Microprocessors, Technology,Architecture, and Applications, J.Wiley and Sons, N.Y. 1976

3. Moore A., W. et al: Microprocessors Applications Manual, McGraw-Hill, N.Y. 1975

4. Šuhel P., Virant J.: Mikroračunalnik, Dop. del Univ. Univerzum, Ljubljana 1978

vrsta s sporočili za komuniciranje z uporabo mikroračunalnika

a.p.železnikar

Odsek za računalništvo in informatiko

INSTITUT "JOŽEF STEFAN"
LJUBLJANA

UDK 681.3-181.4:621.39

Članek opisuje program za realizacijo mehanizma "vrsta" v realnem času, ki lahko z vstavitvijo (uporabo) posebnih simbolov v vrsto povzroči "vmesno" izdajanje standardnih, uradnih ali delovnih sporočil in podsporočil. Pri vstavljanju podsporočil je uporabljen rekurzivni koncept vgnezdenja podsporočil (rekurzivna subrutina QUMSG), ki omogoča ponavljanje tekstovnih segmentov (brez potencialne omejitve) z možnostjo prekinitev preko konzole, ko nastopi vračanje sporočanja na višje ležeča podsporočila in sporočila. Z rekurzivno konstrukcijo sporočil je dosežena znatna prožnost in sestavljivost sporočanja. Program FIFO (vrsta z možnostjo vključevanja sporočil) je namenjen žičnemu in brezžičnemu komuniciranju med dvema udeležencema (terminaloma, teleprinterjema) oziroma ustreznima prenosnima napravama (110 do 9600 Baud). Na koncu članka je dodan še program za deklaracijo sporočil in podsporočil preko konzole.

Message Queue for Communications Using Microcomputer. This article deals with a program which realizes a mechanism called queue under real time circustances; the possibility of insertion (usage) of special symbols into queue can cause an "intermediate" output of standarized, official, and working messages and submessages. For submessage insertion (output) a recursive approach of submessage nesting is applied (recursive subroutine QUMSG) by which iteration of several text segments is enabled; iterations can be interrupted via the computer console (in real time) when returning of message transmission to higher submessages and messages is achieved. By recursive message structure essential flexibility and composibility of messages is obtained. The program FIFO (queueusing the possibility of message insertion) is intended for wire and wireless communications between two partners (terminals, teleprinters) or tranceivers (110 - 9600 Baud). At the end of the article a program for message declarations is listed.

1. Uvod

Članek opisuje program za realizacijo mehanizma vrsta v realnem času, ki lahko z vstavitvijo posebnih simbolov v vrsto povzroči "vmesno" izdajanje standardnih, uradnih ali delovnih sporočil in podsporočil. Vrsta je kot komunikacijski pripomoček namenjena predvsem sestavljanju in oddajanju t.i. individualnih sporočil, ko se sporočajo nepredvidene, osebne, slučajne, pomembne zadeve, ki pa se lahko dopolnjujejo s standardnimi, vljudnostnimi in drugimi, že vnaprej formuliranimi sporočili in podsporočili.

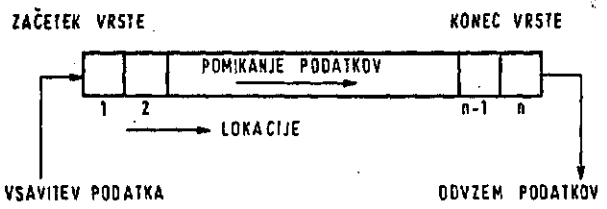
Iz naslednjih poglavij je razvidno, da imamo pri vstavljanju podsporočil in njihovem izdajanju rekurzivni koncept, ki omogoča ponavljanje tekstovnih segmentov (po potrebi in potencialno neomejeno) z možnostjo prekinjanja izdaje teksta preko konzole, v odvisnosti od presoje človeškega ali naključnega operatorja. Z rekurzivno konstrukcijo sporočil je dosežena tudi znatna prožnost sporočanja, seveda če jo operator zna izkoristiti.

Program je napisan za komuniciranje z ASCII znaki oziroma s teksti in nizi, ki so iz teh znakov sestavljeni; ta pristop omogoča izrabo nekega obstoječega operacijskega sistema (monitorja) še zlasti tedaj, ko monitor ni dosledno grajen na osnovi t.i. komunikacijskih kanalov (logičnih enot). Računalnik na eni plošči SDB-80 (s procesorjem Z-80), za katerega je bil obravnavani program vrste napisan, dopušča osposobitev kanalov s pomočjo posebnih programskih vmesnikov (driver, handler); v našem primeru smo predpostavili, da subrutini za čitanje in izpis delujeta za ASCII znake. Seveda pa lahko tem subrutinam dodamo še konverzijske elemente (ASCII-Baudot). Zgradba takih posebnih vmesnikov bo prikazana kdaj kasneje.

Program FIFO je bil napisan za komuniciranje s teksti, ki uporabljajo znake ASCII ter je bil preizkušen za hitrosti med 110 in 9600 Baud. Namenjen je žičnemu in brezžičnemu prenosu sporočil med dvema udeležencema (terminaloma, teleprinterjema), ki uporabljata vrsto s sporočili in podsporočili za tekoče, natančno in individualno komuniciranje.

2. Zamisel krožne vrste

Vrsto (angleško queue ali FIFO, kar je okrajšava za "first-in-first-out"), ki je abstraktna podatkovna struktura, upodobimo v računalniški pomnilnik s t.i. krožno vrsto. Vrsta, ali natančneje linearna vrsta, je zaporedje lokacij, kjer na začetku vrste vstavljamo podatke, na koncu vrste pa jih odvzemamo. Grafična ponazoritev (linearne) vrste izgleda tedaj takole: (slika na naslednji strani prikazuje konfiguracijo vrste, ki ustreza tako formalnemu modelu vrste kot tudi modelu, ki je večkrat uresničen v obliki integriranega vezja; v tem drugem primeru je integrirano vezje FIFO neke vrste pomikalni register, ki dovolj hitro pomakne podatek iz začetka proti koncu vrste)



Podatek, ki smo ga vstavili, se mora iz začetne lokacije pomakniti preko ostalih lokacij do končne lokacije, kjer ga odvzamemo. Linearna vrsta je tedaj nelinearni pomikalni register (ker je pomikanje podatkov od začetka do konca odvisno od zasedenosti vrste), ki pomakne vsak vstavljeni podatek takoj na prvo prosto lokacijo pred koncem vrste. To pomikanje podatkov znotraj vrste pri naključnih frekvencah vstavljanja v vrsto in odvzemanja podatkov iz vrste je lahko dokaj neekonomično s stališča programiranja in uporabe vrste (več ukazov in daljše izvajanje programa). Pomik podatkov se mora opraviti praktično pri vsakem odvzemu ter pri vsaki vstavitvi podatka. Pri odvzemu podatka moramo pomakniti vse podatke na repu (koncu) vrste za eno mesto proti koncu vrste, pri vstavitvi podatka pa moramo opraviti pomik tega podatka do prve proste lokacije.

Krožna vrsta je le racionalnejša preslikava linearne vrste. Pomnilniški prostor, kjer so lokacije za podatke vrste, povežemo s programom v krog (zaključen interval), kot kaže slika 1. Beseda "krog" pomeni, da za krožno vrsto

(1) ne dopuščamo vstavljanja podatkov izven območja kroga, tj. izven danega intervala naslovov in

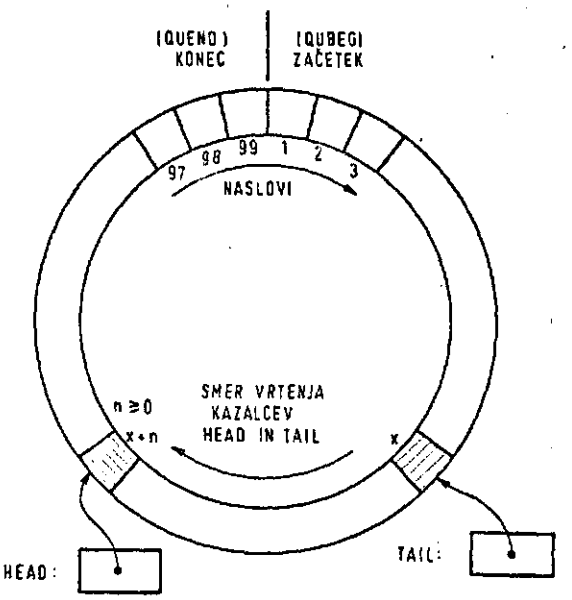
(2) da t.i. začetek in konec vrste krožita v smeri kazalca ure (glede na sliko 1) v danem naslovnem intervalu, in sicer takole: kazalec začetka vrste (angleško HEAD) je ali pred kazalcem konca vrste (angleško TAIL) ali pa sta oba kazalca poravnana (imata enaka naslova kot svoji kazalčni vsebini).

Takšen dogovor o začetku in koncu krožne vrste ima za posledico enolično indikacijo dveh posebnih stanj vrste:

(a) prazna vrsta in
(b) polna vrsta

Ti dve stanji sta povezani z uporabo dveh rutin, tj. z rutino za vrstno vstavljanje (nalaganje) podatkov z imenom QUPUSH ter z rutino za vrstno odvzemanje (izlaganje) podatkov z imenom QUPULL. Kadar sta pri uporabi rutine QUPUSH vrednosti kazalcev HEAD in TAIL enaki, je vrsta polna: v tem primeru novega podatka ne moremo naložiti (ta podatek se lahko tedaj izgubi ali zavrne za kasnejšo naložitev). Kadar se pri uporabi rutine QUPULL vrednosti kazalcev HEAD in TAIL izenačita, je vrsta prazna: v tem primeru iz vrste ni mogoče odvzeti nobenega podatka ter imamo neke vrste indikacijo (posebno stanje določene spremenljivke), da je vrsta prazna.

Pred uporabo moramo mehanizem, ki ga imenujemo vrsta, inicializirati; to opravimo z rutino QUINIT. Inicializacija pomeni definicijo naslovnega intervala ((QUBEG),(QUEND)-1) za vrsto ter začetno nastavitev kazalcev HEAD in TAIL, ki se bosta odslej lahko gibala le po "krogu". K trem rutinam QUINIT, QUPUSH in QUPULL pa je potrebno dodati še povezovalni (uporabniški) programski tekst, s katerim dosežemo delovanje mehanizma, tj. vrste, v realnem času in za konkreten namen. Delovanje v realnem času pomeni, da moramo podatek iz periferije, ki ga je generiral zunanji vir, vstaviti v vrsto takoj, ko se je pojavil (če seveda vrsta ni polna); iz vrste moramo jemati istočasno podatke s frekvenco, ki jo zmore izhodni kanal



Slika 1. Krožna vrsta v naslovnem intervalu ((QUBEG), (QUEND)-1) s kazalcema trenutnega začetka in trenutnega konca vrste (HEAD, TAIL)

(če vrsta ni prazna ali izdajanje znakov ni blokirano). Navadno sta frekvenci (hitrosti) serijskega vhodnega in izhodnega kanala enaki; pri hitrosti 9600 Baud se osembitni podatek lahko pojavi praktično vsako milisekundo in s to hitrostjo se more tudi oddajati.

Zahteva za dupleksni sprejem in oddajo znakov nalaga tedaj mehanizmu "vrsta", da vsebuje čakalno zanko, v kateri se z največjo možno hitrostjo odtipava vhodni in izhodni kanal (perioda zanke za odtipavanje je velikostnega razreda 10 μs); pri uporabi integriranega vezja tipa UART (AY-5-1013) se tedaj odtipavata statusa sprejemnega in oddajnega dela, tj. bita ODA (Output Data Available, pin 19) in TBE (Transmitter Buffer Empty, pin 22).

Glede na povedano, lahko za subrutino QUPUSH napišemo tale psevdo kod:

```
subroutine QUPUSH
  increment HEAD pointer
  if ((HEAD) ≤ (QUEND)-1)
    then
      if ((HEAD) = (TAIL))
        then
          fifo is full;
          decrement (HEAD)
        else
          load (ACCU) to ((HEAD))
      endif
    else
      set (HEAD) to (QUBEG)
      if ((HEAD) = (TAIL)).
        then
          fifo is full;
          decrement (HEAD)
        else
          load (ACCU) to ((HEAD))
      endif
  endif
endsubroutine
```

Dodatna pojasnila k tej subrutini niso potrebna.

```
        NAME Y13
;       NAME Y13
        TITLE F I F O  SUBROUTINES
;       TITLE F I F O  SUBROUTINES
;  QUEUE(FIFO) SUBROUTINES
;*****************************************
;  PROGRAMMED BY ANTON P. ZELEZNIKAR
;    FOR MOSTEK SDB-80 (Z-80) ...
;  DATE:  FEBRUARY 25, 1978
;*****************************************
        GLOBAL QUBEG
        GLOBAL QUEND
        GLOBAL QUINIT
        GLOBAL QUPUSH
        GLOBAL QUPULL
;*****************************************
        ORG 0000H       ;VARIABLE LOCATIONS
QUBEG:  DEFS 2          ;QUBEG LOCATION
QUEND:  DEFS 2          ;QUEND LOCATION
HEAD:   DEFS 2          ;HEAD POINTER
TAIL:   DEFS 2          ;TAIL POINTER
;*****************************************
;  QUEUE(FIFO) INITIALIZATION SUBROUTINE
;    NAMED "QUINIT"
;******
;  PREVIOUS TO QUINIT SUBROUTINE CALL THE
;    CONTENTS OF LOCATIONS QUBEG AND
;    QUEND MUST BE DETERMINED BY HAND OR
;    PROGRAM SETTING ...
;******
QUINIT:PUSH HL          ;SAVE (HL)
       LD HL,(QUBEG)    ;SET HEAD AND TAIL
       LD (HEAD),HL     ; POINTER TO THE
       LD (TAIL),HL     ; FIFO BEGINNING...
       POP HL           ;RELOAD (HL)
       RET              ;RETURN TO CALLER
;*****************************************
;  QUEUE(FIFO) PUSHING SUBROUTINE NAMED
;    "QUPUSH"
;******
;  QUPUSH SUBROUTINE PUSHES THE CONTENT
;    OF ACCU A INTO QUEUE(FIFO) ...
;  IF FIFO IS FULL THE CONTENT OF ACCU
;    WILL NOT BE PUSHED ...
;  THERE IS A FULL STATE INDICATION WHEN
;    ACCU IS SET TO 00H ...
;******
QUPUSH:PUSH HL          ;SAVE (HL) AND LOAD
       LD HL,(HEAD)     ; HEAD POINTER INTO
;                       ; HL...
;      EJECT
       EJECT
       INC HL           ;UPDATE HEAD POINT.
       PUSH BC          ;SAVE (BC)
       PUSH HL          ;SAVE HEAD POINTER
       LD BC,(QUEND)    ;COMPARE IF HEAD
       OR A             ; POINTER IS AT THE
       SBC HL,BC        ; END OF FIFO...
       POP HL           ;IF NEGATIVE GO TO
       JR C,EXIT1-$     ; EXIT1...
       LD HL,(QUBEG)    ;IF EQUAL SET HEAD
;                       ; TO FIFO BEGINNING.
EXIT1: LD BC,(TAIL)     ;LOAD TAIL POINTER
;                       ; INTO BC...
       PUSH HL          ;SAVE HEAD POINTER
       OR A             ;RESET CARRY FLAG
       SBC HL,BC        ;COMPARE POINTERS
       JR Z,FULL-$      ;IF EQUAL FIFO IS
;                       ; FULL...
       POP HL           ;RELOAD HEAD POINT..
       LD (HL),A        ;LOAD ACCU TO FIFO
       LD (HEAD),HL     ;UPDATE HEAD POINTER
EXIT2: POP BC           ;RELOAD (BC)
       POP HL           ;RELOAD (HL)
       RET              ;RETURN TO CALLER
FULL:  LD A,00H         ;SET ACCU TO 00H
       POP HL           ;ADJUST STACK P.
       JR EXIT2-$       ;GO TO EXIT2
;*****************************************
;  QUEUE(FIFO) PULLING SUBROUTINE NAMED
;    "QUPULL"
;******
;  QUPULL SUBROUTINE PULLS THE CONTENT
;    FROM FIFO INTO ACCU A ...
;  IF FIFO IS EMPTY THE CONTENT OF ACCU A
;    WILL BE SET TO 00H ...
;******
QUPULL:PUSH HL          ;SAVE (HL)
       LD HL,(TAIL)     ;TAIL POINT. IN HL
       PUSH BC          ;SAVE (BC)
       PUSH HL          ;SAVE TAIL POINTER
       LD BC,(HEAD)     ;HEAD POINT. TO BC
       OR A             ;RESET CARRY FLAG
       SBC HL,BC        ;COMPARE POINTERS
       JR Z,EMPTY-$     ;IF EQUAL FIFO IS
;                       ; EMPTY...
       POP HL           ;OTHERWISE POP TAIL
       INC HL           ;UPDATE POINTER
       PUSH HL          ; AND SAVE TAIL POIN
       LD BC,(QUEND)    ;LOOK IF TAIL POINT.
       OR A             ; IS AT THE END OF
;      EJECT
       EJECT
       SBC HL,BC        ; FIFO...
       POP HL           ;
       JR C,EXIT3-$     ;IF NOT GO TO EXIT3
       LD HL,(QUBEG)    ;IF YES SET TAIL PO.
;                       ; TO FIFO BEGINNING
EXIT3: LD A,(HL)        ;PULL DATA FROM FIFO
       LD (TAIL),HL     ;UPDATE TAIL POINTER
EXIT4: POP BC           ;RELOAD (BC)
       POP HL           ;RELOAD HL
       RET              ;RETURN TO CALLER
EMPTY: LD A,00H         ;SET ACCU TO 00H
       POP HL           ;ADJUST STACK P.
       JR EXIT4-$       ;GO TO EXIT4
;*****************************************
       END
```

Slika 2. Subrutine QUINIT, QUPUSH in QUPULL, napisane v zbirnem jeziku mikro procesorja Z-80. Te subrutine oblikujejo osnovo za zgraditev mehanizma "vrsta" (angleško "queu") v realnem času in v konkretnem primeru; opisane subrutine udejanijo "krožno vrsto" s kazalcema HEAD (glava: vstavitev podatka v vrsto) in TAIL (rep: odvzem podatka iz vrste). Podatek v ACCU (akumulator A) se vstavlja ali pa je odvzet iz vrste; pri vrnitvi vrednosti (ACCU) = ØØH je vrsta polna (vstavitev) ali prazna (odvzem) (glej tekst!)

Pri subrutini QUPULL se najprej preizkusi, ali je vrsta (FIFO) prazna, šele potem se rep (TAIL) inkrementira. Po inkrementiranju pa primerjava (HEAD) = (TAIL) ni več potrebna oziroma je nedopustna, če želimo izpisati tudi zadnji (poslednji trenutni) znak v vrsti. Prikazana zgradba subrutine QUPULL je posledica zgradbe subrutine QUPUSH in le ustrezna uporaba obeh zagotavlja delovanje mehanizma "vrsta". Psevdo kod za subrutino QUPULL je tedaj:

```
subroutine QUPULL
  if ((HEAD) = (TAIL))
    then
      fifo is empty
    else
      increment TAIL pointer
      if ((TAIL) ≤ (QUEND)-1)
        then
          load ((TAIL)) to ACCU
        else
          set (TAIL) to (QUBEG);
          load((TAIL)) to ACCU
      endif
  endif
endsubroutine
```

Subrutina QUINIT je enostavna. Na sliki 2 imamo subrutine QUINIT, QUPUSH in QUPULL napisane v zbirnem jeziku procesorja Z-80 z ustreznimi globalnimi spremenljivkami.

3. Krožna vrsta, ki poziva dodatna sporočila iz hitrega pomnilnika

Med sprejemom teksta se v odvisnosti od sprejetega teksta oblikujejo sporočila, ali tudi sporočila sporočil (v rekurzivnem pomenu), ki se shranjujejo v pomnilniku. Zunanji vir lahko zahteva vključitev določenega, standardnega sporočila s posebno direktivo (tekstom), ki ima npr. sintakso

Ø ⟨ime⟩

kjer naj bo v naših primerih

⟨ime⟩ ::= ⟨heksadecimalna številka⟩

To pomeni, da zunanji vir pošilja v vrsto med drugimi podatki tudi nize oblike

ØØ, Ø1, Ø2, ... , ØE, ØF

ki se interpretirajo kot sporočila, opredeljena na posebnih lokacijah. Tako imamo možnost izbire 16 standardnih sporočil, tj. glavnih sporočil in 16 podsporočil (pojasnjeno kasneje). Sporočila so na splošno sestavljena iz konstantnih in spremenljivih delov; spremenljivi del je praviloma podsporočilo, ki ga oblikujemo na osnovi sprejetih podatkov iz zunanjega vira. Tip vrste, ki sprejema tekst iz zunanjega vira ter hkrati oddaja (v določenem časovnem razdobju) sporočilo, ki je bilo v vrsti navedeno samo z izrazom

Ø ⟨heksadecimalna številka⟩

bomo imenovali vrsta z generatorjem sporočil; pri tem vemo, da gre v tem primeru za posebno uporabo mehanizma "vrsta".

Vrsta z generatorjem sporočil (z generatorjem sporočil) mora imeti podobno čakalno zanko (natančneje tipalno zanko) kot navadna vrsta, ko se nadzorujeta sprejem znakov iz zunanjega vira in maksimalna hitrost oddaje znakov sporočila v vobče neki drugi zunanji vir. Zunanja vira imata npr. tole vlogo: prvi zunanji vir je človeški operator, ki preko vrste z generatorjem sporočil komunicira z drugim zunanjim virom tj. z drugim človeškim operatorjem. Takšen način komunikacije omogoča:

(1) sestavljanje odgovorov iz individualnih, standardnih in spremenljivih delov;

(2) maksimalno hitrost oddaje tekstov, ker se odgovor lahko oblikuje že med sprejemom teksta korespondenta;

(3) pripravo teksta preko vrste tudi v času oddaje teksta, kjer se z vmesnimi sporočili pridobi čas za sestavljanje individualnih delov teksta.

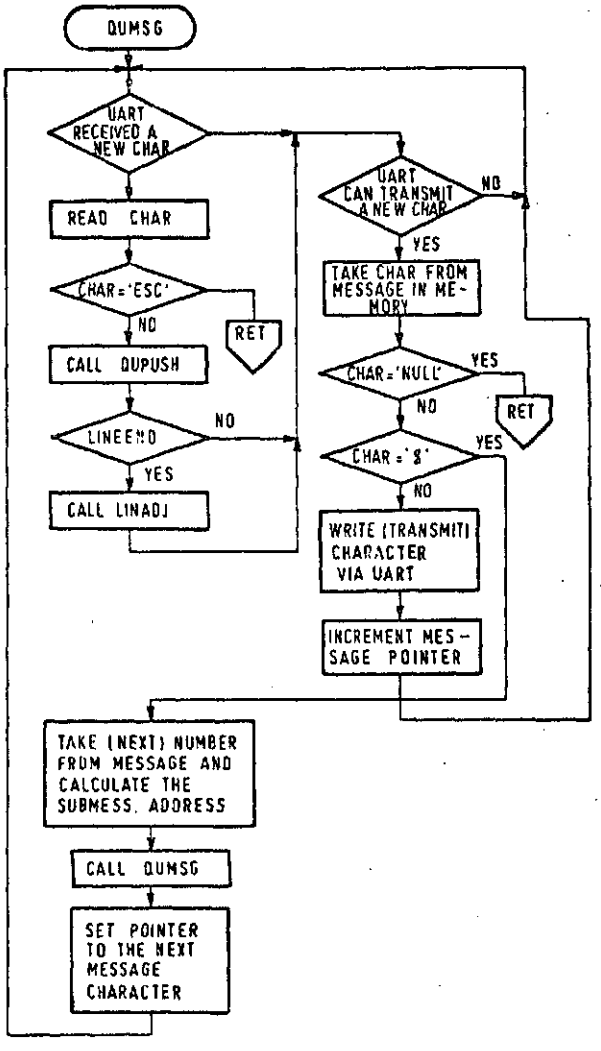
Mehanizem vpisovanja in izpisovanja iz vrste v "realnem času" bomo podrobneje opisali v naslednjem poglavju; tu si podrobneje oglejmo zgradbo subrutine z imenom MESAGE in njene rekurzivne subrutine QUMSG.

Imejmo za subrutino MESAGE tale enostaven psevdo kod:

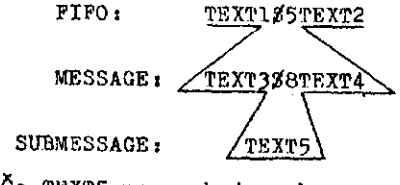
```
subroutine MESAGE
  calculate the message address;
  print message and simultaneously receive
    characters from console into FIFO (e.g.
    call QUMSG)
endsubroutine
```

S prvim delom te subrutine se izračunava naslov, na katerem je shranjen naslov začetka sporočila v odvisnosti od niza Øi (i=0,1,..., F). Drugi del subrutine zagotavlja hkraten sprejem in oddajo znakov z najvišjo dopustno hitrostjo serijskega kanala. Diagram programa za subrutino QUMSG je prikazan na sliki 3.

Subrutina QUMSG je rekurzivna in poziva samo sebe na nivoju podsporočil poljubno globoko (vgnezdenje globine j). Vzemimo za primer interpretacijo tega sporočila:



Slika 3. Programski diagram subrutine QUMSG, ki je rekurzivna (kliče sama sebe) in omogoča poljubno globoko vgnezdenje podsporočil (SUBMESSAGE)



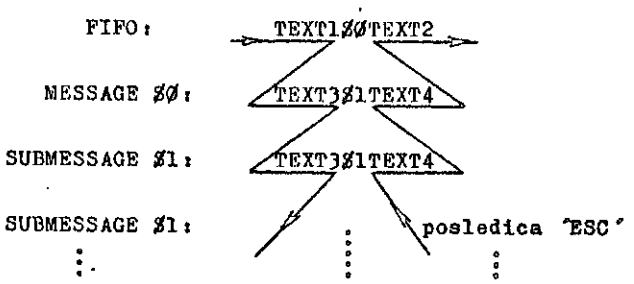
Če TEXT5 ne vsebuje nobenega podsporočila, je rezultat celotno sporočilo (na izhodu)

TEXT1TEXT3TEXT5TEXT4TEXT2

Kadar vsebuje del z označitvijo SUBMESSAGE v zgornji shemi še kakšen niz 'Øi', se v podsporočilo vključi poljubno (definirano) podsporočilo. Rekurzija (potencialno neskončna) nastopi npr. v primeru:

```
        NAME Y16
        TITLE MESSAGE SUBROUTINES
;       TITLE MESSAGE SUBROUTINES
        NAME Y16
;   THREE MESSAGE SUBROUTINES
;*****************************************
;   PROGRAMMED BY ANTON P. ZELEZNIKAR
;     FOR MOSTEK SDB-80 (Z-80) ...
;   DATE:  MARCH 10, 1978
;*****************************************
        GLOBAL MESAGE
        GLOBAL QUMSG
        GLOBAL MSGLOC
        GLOBAL SUBMSG
        GLOBAL LINADJ
        GLOBAL QUPUSH
;*****************************************
;   MESSAGE WRITING SUBROUTINE NAMED
;     "MESAGE"
;******
;   THIS SUBROUTINE CAN PRINT 16 DIFFE-
;     RENT MESSAGES ENDING IN ASCII NULL.
;     THE ADDRESSES OF 16 MESSAGES HAVE
;     TO BE LOAD. TO MSGLOC, MSGLOC+2,...,
;     MSGLOC+30 (SEE PROGRAM TEXT). SO,
;     ONE CAN USE MESSAGES NUMBERED BY
;     0,1,2,...,F (IN HEXADECIMAL). HEX
;     NUMBERS (30,31,...,39,41,42,...,46)
;     HAVE TO BE LOAD. IN ACCU A BEFORE
;     SUBROUTINE CALL ...
;     A MESSAGE CAN USE SUBMESSAGES IN
;     THE SAME WAY:  SUBMSG LOCATIONS
;     STORE THE BEGINNINGS OF SUBMES-
;     SAGES (SEE QUMSG SUBROUTINE)...
;******
        ORG 0000H       ;ADDRESS LOCATIONS
MSGLOC DEFS 032         ;16 ADDRESSES
;******
MESAGE:CALL ASBIN       ;CONV ASCII TO BIN
       CP 16            ;RETURN IF GREATER
       RET P            ; OR EQUAL 16...
       PUSH HL          ;SAVE (HL)
       LD HL,MSGLOC     ;SET HL
       ADD A,A          ;MULTIPLY BY 2
       OR A             ;CLEAR CARRY FLAG
       ADD A,L          ;CALC LOWER BYTE,
       LD L,A           ; LOAD A TO L...
       JR NC,NADJ-$     ;IF NC GO TO NADJ
       INC H            ;ADJUST (H)
NADJ:  LD A,(HL)        ;LOWER ADDR PART
       INC HL           ;INCREMENT (HL)
       LD H,(HL)        ;UPPER ADDR PART
       LD L,A           ;LOWER PART TO L
       CALL QUMSG       ;PRINT MESSAGE
       POP  HL          ;RELOAD (HL)
       RET              ;RETURN TO CALLER
;*****************************************
;   SUBROUTINE NAMED "QUMSG"
;******
;   WHEN CALLED SUBROUTINE QUMSG CAN RE-
;     CEIVE CHARACTERS FROM THE CONSOLE
;     INTO FIFO AND WILL TYPE CHARACTERS
;     OF A MESSAGE IN RAM OR ROM SIMULTA-
;     NEOUSLY. RETURN TO THE CALLER IS
;     PERFORMED IF CHARACTER 'ESC' IS RE-
;     CEIVED FROM CONSOLE; IN THIS CASE
;     THE TYPING OF MESSAGE IS INTERRUP-
;     TED. A RETURN TAKES PLACE ALSO
;     WHEN THE MESSAGE FROM RAM OR ROM IS
;     ENDING IN CHARACTER 'NULL'.
;     SUBMESSAGES $0,...,$F WILL BE
;     PROCESSED BY QUMSG TOO IF SUBMSG
;     LOCATIONS ARE POINTING TO SUB-
;     MESSAGE BEGINNINGS (SUBMSG,
;     SUBMSG+2,...,SUBMSG+30)...
;******
SUBMSG DEFS 032         ;16 ADDRESSES
QUMSG: LD E,00H         ;WAIT CONTROL
IMD5:  IN A,(0DDH)      ;INPUT UART STATUS
       BIT 6,A          ;TEST 'ODA' BIT
       JR Z,IMD4-$      ;IF Z GO TO IMD4
       CALL RDCHR       ;READ CHARACTER
       CP 1BH           ;COMPARE IF 'ESC'
       RET Z            ; THEN RETURN
       CALL QUPUSH      ;PUSH CHARACTER
       DJNZ IMD4-$      ;IF NZ GO TO IMD4
       CALL LINADJ      ;LINE ADJUST
IMD4:  IN A,(0DDH)      ;INPUT UART STATUS.
       BIT 7,A          ;TEST 'TBE' BIT
       JR Z,IMD5-$      ;IF Z GO TO IMD5
       LD A,(HL)        ;LOAD ((HL)) TO A
       CP 00H           ;COMPARE IF 'NULL'
       RET Z            ; THEN RETURN
       CP '$'           ;COMPARE IF '$'
       JR Z,IMD6-$      ;IF Z GO TO IMD6
       LD D,A           ;LOAD A TO D
       CALL WRCHR       ;WRITE CHARACTER
       INC HL           ;INCREMENT (HL)
       JR IMD5-$        ;GO TO IMD5
;******
IMD6:  INC HL           ;TAKE SUBMESSAGE
       LD A,(HL)        ; NUMBER FROM MEM.
       CALL ASBIN       ;CONV ASCII TO BIN
       CP 16            ;IGNORE IF GREATER
       JR NC,IMD5-$     ; OR EQUAL 16...
       PUSH HL          ;SAVE (HL)
       LD HL,SUBMSG     ;SET HL
       ADD A,A          ;MULTIPLY BY 2
       OR A             ;CLEAR CARRY FLAG
       ADD A,L          ;CALC LOWER BYTE
       LD L,A           ; AND LOAD IT TO L
       JR NC,SADJ-$     ;IF NC GO TO SADJ
       INC H            ;ADJUST H
SADJ:  LD A,(HL)        ;LOWER ADDR PART
       INC HL           ;INCREMENT (HL)
       LD H,(HL)        ;UPPER ADDR PART
       LD L,A           ;LOWER ADDR PART
       CALL QUMSG       ;PRINT SUBMESSAGE
       POP HL           ;RELOAD (HL)
       INC HL           ;INCREMENT (HL)
       JR IMD5-$        ;GO TO IMD5
;*****************************************
;   SUBROUTINE NAMED "LINADJ"
;******
;   THIS SUBROUTINE PUSHES TWO CARRIAGE
;     RETURN AND ONE LINE FEED CHARACTER
;     INTO FIFO AND SETS THE CHARACTER
;     COUNTER B TO 64 ...
;******
LINADJ:LD A,0DH         ;PUSH DOUBLE CAR-
       CALL QUPUSH      ; RIAGE RETURN
       CALL QUPUSH      ; INTO FIFO...
       LD A,0AH         ;PUSH LINE FEED
       CALL QUPUSH      ; INTO FIFO...
       LD B,64          ;SET CHR COUNTER
       RET              ;RETURN TO CALLER
;*****************************************
ASBIN  EQU 0E583H       ;ASCII-BIN CONVER.
RDCHR  EQU 0E522H       ;READ ASCII CHAR
WRCHR  EQU 0E527H       ;WRITE ASCII CHAR
;*****************************************
       END
```

Slika 4. Subrutine MESAGE, QUMSG in LINADJ, napisane v zbirnem jeziku mikro procesorja Z-80



Iz takega vgnezdenja (veriženja navzdol) skočimo z vpisom (preko konzole) znaka 'ESC', ki povzroči "vračanje" preko tekstovnih ostankov. Čim globlje je bilo vgnezdenje, tem dalj časa traja "vračanje", seveda, če obstajajo t.i. repi (neprazni tekstovni nizi) podsporočil. Vračanje lahko pospešimo (preskočimo) z večkratnim, dovolj hitrim zaporednim vpisom znaka 'ESC'.

V diagramih sporočil (gornja primera) imamo smeri počasnega (sporočilnega) in hitrega (programskega) gibanja. Vodoravne desne smeri gibanja so tedaj počasne (hitrost serijskega kanala), poševne (navzdolnje in navzgornje) smeri pa so hitre (hitrost izvajanja pripadajočega ukaznega segmenta).

Kadar imamo podsporočilo oblike

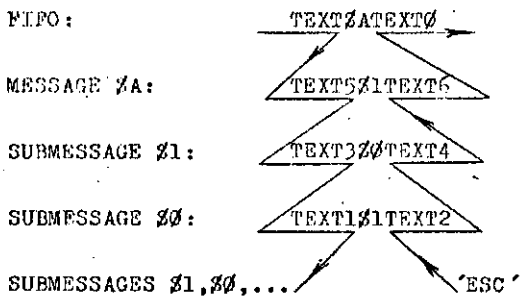
TEXTØ2

ko je poziv nekega podsporočila na koncu niza (tu je poziv Ø2), tedaj je vrnitev v sporočilo, v katerem je to sporočilo vgnezdeno, vselej "hitra". Pri rekurziji takih sporočil je že z enim samim znakom 'ESC' vrnitev preko več sporočil zelo hitra (seštevek hitrih vrnitev).

Pozivanje sporočil je lahko seveda križno oziroma kompleksno (splošno) rekurzivno. Primer križne rekurzije bi dobili tedaj, ko imamo definiciji podsporočil

SUBMESSAGE ØØ: TEXT1Ø1TEXT2
SUBMESSAGE Ø1: TEXT3ØØTEXT4

Tako dobimo:

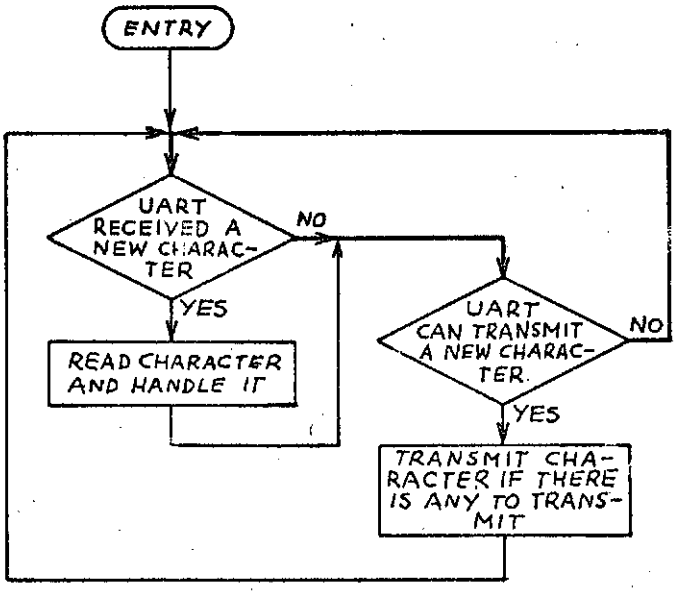


Podobno bi lahko konstruirali tudi drugačno, kompleksnejšo rekurzijo pozivanja podsporočil.

Kot že rečeno, je lastnost rekurzivnega pozivanja podsporočil, določena s subrutino QUMSG, katere programski diagram vidimo na sliki 3. Paket sporočilnih subrutin, ki se povezujejo tudi s subrutinama QUPUSH in QUPULL, je prikazan na sliki 4. Subrutina LINADJ omejuje število znakov na vrstico, in sicer na 64, kar pomeni, da imamo v eni vrstici teksta v FIFO kvečjemu 64 ASCII znakov, nakar se avtomatično generirata dva znaka 'CR' ter en znak 'LF'. Vendar se subrutina LINADJ ne uporablja za omejevanje teksta sporočil in podsporočil; tu moramo tekst formatirati v samih sporočilih, tako da na ustreznih mestih vstavimo zopet dva znaka 'CR' (da pridobimo dvojni čas za vrnitev valja) in 'LF' (pomik vrstice).

4. Delovanje vrste v "realnem času"

V poglavju 3 se je pokazalo, da moramo tudi v programirani, tj. s programom realizirani vrsti (FIFO), zagotoviti pogoje njenega obratovanja v "realnem času". Narekovaje za ta čas smo postavili, ker razumemo pod realnim časom minimalno, z izvajanjem zadevnega programskega segmenta povzročeno zamudo (zakasnitev), ki predstavlja razliko, potrebno, da se akcija vpisa in izpisa začne glede na realne vhodne/izhodne pogoje. Ta zamuda je veliostnega razreda 10 μs in je tako stokrat (9600 Baud) ali celo tisočkrat (110 Baudov) in več kot to manjša od trajanja izpisa ali vpisa enega karakterja.



Slika 5. Osnovni programski diagram za dupleksni prenos podatkov (čitanje in izpis) preko integriranega vezja UART. Debelo izvlečena proga predstavlja čakalno zanko, ko nimamo niti čitanja niti izpisa, ker ustreznih karakterjev ni (karakter se ni pojavil s konzole ali ni na zalogi v vrsti oziroma UART že oddaja en karakter in ima drugega v svojem oddajnem registru). Bloka if v tem diagramu pomenita, da testiramo pri čitanju (receive) bit 'ODA' ter pri izpisu (transmit) bit 'TBE' integriranega vezja UART; ta situacija je razvidna iz programske liste na sliki 4 in na sliki 6.

Z rekurzivno subrutino QUMSG se nam ta lastnost prenaša tudi poljubno globoko v vgnezdenje podsporočil.

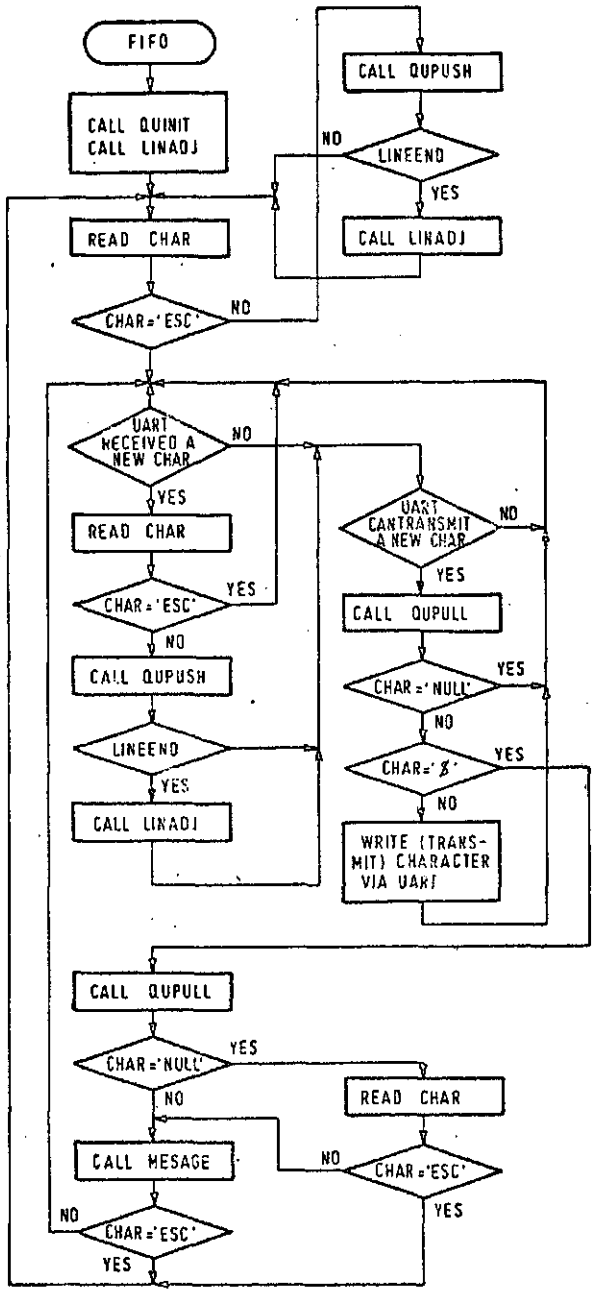
Iz programskega diagrama na sliki 3 dobimo osnovni diagram na sliki 5, ki pojasnjuje pogoje dupleksnega obratovanja vrste kot samostojnega mehanizma ali vpisa v vrsto in izpisa sporočila iz pomnilnika (to je izven vrste) v "realnem času". Podobna struktura se tedaj pojavlja tudi v programskem diagramu na sliki 6.

Diagram na sliki 5 se nanaša na integrirano vezje tipa UART; namesto njega bi seveda lahko v konkretnem primeru vzeli vezje USART ali SIO (npr. Z-80-SIO); v teh primerih bi na podoben način testirali ustrezne bite oziroma signale na sponkah oziroma nožicah vezja.

5. Zgradba in delovanje programa, ki uporablja mehanizem vrste z vstavljanjem sporočil

Doslej smo opisali šest subrutin, in sicer tri osnovne za mehanizem "vrsta" (QUINIT, QUPUSH, QUPULL) ter tri subrutine za sporočanje (MESAGE, QUMSG, LINADJ); subrutina LINADJ je namenjena omejevanju števila znakov v vrstici teksta, ki se oblikuje skozi vrsto; ta rutina se tedaj smiselno navezuje na uporabo subrutine QUPUSH in na inicializacijo programa. Program, ki poveže te subrutine v smiselno celoto, bomo kratko imenovali FIFO.

Program FIFO ima nekatere značilne lastnosti glede na nize karakterjev, ki prihajajo preko vhodnega kanala (interna oziroma sistemske subrutina RDCHR) ali pa so bili preko tega kanala vpisani v vrsto. To so krmilni nizi, ki



Slika 6. Programski diagram programa FIFO, ki kaže vpliv kontrolnih znakov in kontrolnih nizov na programsko izvajanje ter pozive ključnih subrutin in testiranje stanja vezja UART, ki zagotavlja obratovanje mehanizma vrsta v realnem času

povzročajo spremembe funkcionalnih stanj programa FIFO (glej sliko 8). V poglavju 3 smo spoznali vpliv nizov %i (i = Ø,1,2,...,E,F) na nivoju sporočil in podsporočil. K tem nizom moramo dodati še niz

$ 'NULL'

katerega pomen je razvidn iz slike 8 (stanje IMD2) ter karakter (niz z enim samim znakom)

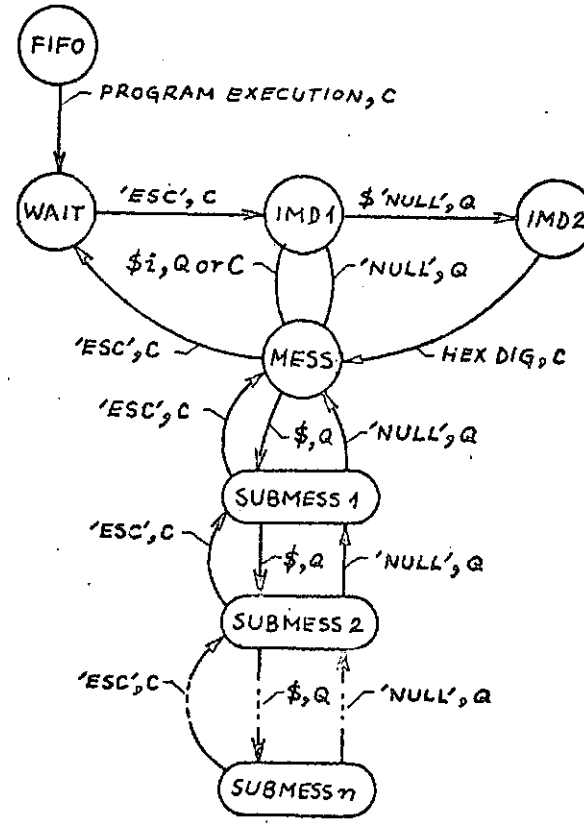
'ESC'

katerega vpliv (krmilna sposobnost) je večstranski.

Program FIFO, ki je shematično prikazan s programskim diagramom na sliki 6 ter z listo na sliki 7, ima štiri značilna funkcionalna stanja: WAIT, IMD1, IMD2 in MESS (MESS1); ta stanja se ujemajo z označitvami v programu na sliki 7.

Oglejmo si delovanje programa v povezavi s sliko 8, kjer imamo diagram stanj programa. Stanje WAIT (wait for FIFO output) je stanje polnenja vrste, ko se podatki v vrsto vstavljajo z naključno frekvenco (do najvišje hitrosti serijskega kanala) in se iz vrste ne odvzemajo. Ko aktiviramo program FIFO, tj. ko ga začnemo izvajati, smo vedno v stanju WAIT.

Iz stanja WAIT preidemo v stanje IMD1 (immediate FIFO input and output), ko se je iz konzole pojavil znak 'ESC'. V tem stanju se znaki iz konzole (počasi ali hitro, do maksimalne hitrosti) vpisujejo v vrsto ter se iz nje izpisujejo v serijski kanal (konzolo) z maksimalno hitrostjo (eden za drugim), ki jo zmore serijski kanal. Pojav znaka 'ESC' s konzole povzroči vrnitev iz stanja IMD1 v stanje WAIT.



STATE:	FUNCTION
FIFO :	START OF FIFO PROGRAM
WAIT :	FIFO INPUT ENABLED
IMD1 :	FIFO INPUT AND OUTPUT ENABLED
IMD2 :	FIFO OUTPUT DISABLED; WAITING FOR HEX DIGIT (MESS NUMBER) FROM CONSOLE
MESS :	FIFO OUTPUT DISABLED; MESSAGE TRANSMISSION ENABLED; FIFO INPUT ENABLED
SUBMES:	FIFO OUTPUT DISABLED; MESSAGE AND/OR SEVERAL SUBMESSAGES TRANSMISSIONS INTERRUPTED; SUBMESSAGE TRANSMISSION ENABLED; FIFO INPUT ENABLED

Slika 8. Programski diagram stanj z legendo stanj v diagramu

```
        TITLE  F I F O  RTTY PROGRAM
        NAME Y15
;       RTTY COMMUNICATION PROGRAM
;*****************************************
;   PROGRAMMED BY ANTON P. ZELEZNIKAR
;     FOR MOSTEK SDB-80 (Z-80) ...
;   DATE: MARCH 16, 1978
;*****************************************
;   RADIO TELETYPE ANSWERING PROGRAM USED
;     FOR COMMUNICATION BY FREE TEXT (FI-
;     FO) AND MESSAGES (FIXED AND VARIA-
;     BLE) ...
;******
;   FIRST, AFTER PROGRAM ACTIVATION ONE
;     PUSHES (INPUTS) TEXT INTO FIFO
;     WITHOUT FIFO OUTPUT (EMPTYING) UN-
;     TIL AN ASCII 'ESC' APPEARS. NOW,
;     FIFO BEGINS TO OUTPUT THE INPUTED
;     TEXT BUT CAN STILL ACCEPT NEW TEXT
;     (BY TESTING OF 'ODA' AND 'TBE' BIT
;     OF UART). WHEN INPUTING (PUSHING)
;     A TEXT 16 PREDETERMINED MESSAGES CAN
;     BE INCLUDED USING DIRECTIVES $0, $1,
;     ... , $F. THE OUTPUT TEXT HAS A FOR-
;     MAT OF 64 CHARACTERS PER LINE BUT
;     MESSAGES MUST HAVE THE FORMAT SET
;     IN THE MEMORY. LOCATIONS MSGLOC,
;     MSGLOC+2,...,MSGLOC+30 HAVE TO BE
;     FILLED PREVIOUSLY TO PROGRAM ACTI-
;     VATION BY ADDRESSES OF MESSAGE BE-
;     GINNINGS. FIFO AREA MUST BE DETER-
;     MINED BY (QUBEG) AND (QUEND).
;     A SECOND ASCII 'ESC' SETS FIFO
;     BACK INTO WAIT STATE (INTERRUPTS
;     FIFO OR MESSAGE OUTPUT); SO,ONLY
;     FIFO INPUT (PUSHING) IS ENABLED.
;******
;   THIS PROGRAM HAS NOT A FIFO FULL INDI-
;     CATION. THEREFORE ONE HAS TO CHOOSE
;     PROPERLY (QUBEG) AND (QUEND) TO EN-
;     SURE A SAFE TRANSMISSION OF TEXT
;     (WITHOUT LOSS)!
;   FOR PROPER (QUBEG) AND (QUEND) INSER-
;     TION LOOK AT QUINIT, QUPUSH, AND
;     QUPULL SUBROUTINES (TAPE NAME Y13)
;******
        GLOBAL QUINIT
        GLOBAL QUPUSH
        GLOBAL QUPULL
        GLOBAL LINADJ
        GLOBAL MESAGE
;*****************************************
        ORG 0000H       ;PROGRAM ORIGIN
        CALL QUINIT     ;CALL FIFO INIT
        CALL LINADJ     ;LINE ADJUST
;
        LD E,00H        ;WAIT CONTROL
WAIT:   CALL RDCHR      ;READ CHARACTER
        CP 1BH          ;COMPARE IF 'ESC'
        JR Z,IMD1-$     ; THEN GO TO IMD1
        CALL QUPUSH     ;PUSH CHR INTO FIFO
        DJNZ WAIT-$     ;RETURN UNTIL LINE
;                       ; END...
        CALL LINADJ     ;LINE ADJUST
        JR WAIT-$       ;REPEAT PUSHING
;*****************************************
IMD1:   IN A,(0DDH)     ;INPUT UART STATUS
        BIT 6,A         ;TEST 'ODA' BIT
        JR Z,IMD3-$     ;IF Z GO TO IMD3
        CALL RDCHR      ;READ CHARACTER
        CP 1BH          ;COMPARE IF 'ESC'
        JR Z,WAIT-$     ; THEN GO TO WAIT
;****** THIS IS SWITCHBACK TO FIFO WRITE
;          ONLY STATE ...
;******
        CALL QUPUSH     ;PUSH CHR INTO FIFO
        DJNZ IMD3-$     ;TEST FOR LINE END
        CALL LINADJ     ;LINE ADJUST
IMD3:   IN A,(0DDH)     ;INPUT UART STATUS
        BIT 7,A         ;TEST 'TBE' BIT
        JR Z,IMD1-$     ;IF Z GO TO IMD1
        CALL QUPULL     ;PULL CHR FROM FIFO
        CP 00H          ;COMPARE IF 'NULL'
        JR Z,IMD1-$     ; THEN GO TO IMD1
;****** IN THIS POINT FIFO IS EMPTY
;******
        CP '$'          ;COMPARE IF '$'
        JR Z,MESS-$     ; THEN GO TO MESS
        LD D,A          ;LOAD A TO D
        CALL WRCHR      ;WRITE CHR
        JR IMD1-$       ;GO TO IMD1
MESS:   CALL QUPULL     ;PULL CHR FROM FIFO
        CP 00H          ;COMPARE IF 'NULL'
        JR Z,IMD2-$     ; THEN GO TO IMD2
MESS1:  CALL MESAGE     ;FIFO PLUS MESSAGE
        CP 1BH          ;COMPARE IF 'ESC'
        JR Z,WAIT-$     ; THEN GO TO WAIT
        JR IMD1-$       ;GO TO IMD1
IMD2:   CALL RDCHR      ;READ CHR
        CP 1BH          ;COMPARE IF 'ESC'
        JR Z,WAIT-$     ;IF Z GO TO WAIT
        JR MESS1-$      ;GO TO MESS1
;*****************************************
RDCHR   EQU 0E522H      ;SUBROUTINE
WRCHR   EQU 0E527H      ;SUBROUTINE
 END
```

Slika 7. Lista programa FIFO z označitvami WAIT, IMD1, MESS (MESS1) in IMD2, ki ponazarjajo ustrezna stanja na sliki 8

Tu je 'ESC' očitno signal za bistabilno igro med stanjema WAIT in IMD1 v odvisnosti od potreb človeškega ali naključnega faktorja (konzole).

Med vstavljanjem podatkov v vrsto, ko je bila ta v stanju WAIT ali IMD1, se je v vrsto lahko vpisal niz oblike %i (i je heksadecimalna številka) ali oblike %'NULL'. Ko pride %i do izpisa iz vrste, preide program iz stanja IMD1 v stanje MESS in v izhodni kanal se začne izpisovati sporočilo, shranjeno v pomnilniku (seveda izven prostora vrste). Ko se je pojavil zaključni znak sporočila 'NULL', preide program zopet iz stanja MESS v stanje IMD1. Oddaja sporočila se lahko prekine, če se pojavi znak 'ESC' iz konzole, ko preide program iz stanja MESS v stanje WAIT. To pomeni, da se začeto sporočilo ne bo več nadaljevalo (lahko pa se bo začelo oddajati znova kdaj kasneje).

Če se je v stanju IMD1 pojavil niz %'NULL' iz vrste, preide program v stanje IMD2. V tem stanju čaka program na številko sporočila iz konzole. Stanje IMD2 je predvideno za primer, ko v sporočanju ni vnaprej jasno, katero sporočilo bo glede na obstoječi kontekst komuniciranja najprimernejše. V tem primeru čaka program na odločitev konzole (ali naključnega generatorja), ki mu določi potek nadaljevanja oddaje sporočila. Ker lahko zbirka sporočil vsebuje tudi prazno sporočilo, je s tem možno anulirati oddajo sporočila, ko skočimo sicer iz stanja IMD2 v stanje MESS, toda takoj nato iz stanja MESS v stanje IMD1. Prazno sporočilo je sestavljeno iz enega samega znaka 'NULL'. Prehod iz stanja IMD2 v stanje WAIT je možen z znakom 'ESC', prehod v stanje IMD1 pa tudi z dvema zaporednima znakoma 'ESC', ki ju oddamo s konzole.

Aktiviranje programskih stanj SUBMES1, SUBMES2, ... , SUBMESn, ko preidemo v stanje SUBMES1 iz stanja MESS in nadalje vobče iz stanja SUBMESj v stanje SUBMESj+1, pri čemer je j potencialno neomejeno, je odvisno od zgradbe sporočil in podsporočil, oziroma v njih vsebovanih podnizov %i (i = 0,1,...,E,F). Povedali smo že, da je tu globina vgnezdenja potencialno neomejena, vendar je zasedenega vedno več prostora v t.i. računalniškem skladu, kjer se shranjujejo naslovi subrutinskih vrnitev ter še parametri obdajajočih subrutin. Iz podsporočila na globini j+1 se vrnemo vobče v podsporočilo na globini j, ko se je pojavil znak 'NULL' na koncu podsporočila globine j+1 ali še pred tem, če se je pojavil znak 'ESC' iz konzole (tasta-

ture). Zaradi opisane lastnosti je možna "hitra" vrnitev preko več podsporočil z więč zaporednimi znaki 'ESC' s konzole.

Diagram stanj programa FIFO na sliki 8 dodobra pojasnjuje uporabo programa FIFO, ko imamo kontrolne znake 'ESC' iz tastature (mimo vrste) ter kontrolne znake 'NULL', Ø in 'heksadecimalna številka' preko vrste (z uporabo rutin QUPUSH in QUPULL).

6. Primeri uporabe programa FIFO

V poglavju 3 smo si ogledali osnovne primere sestavljanja sporočil iz teksta v vrsti ter z vključevanjem (substitucijo) sporočil in podsporočil. Ob tem nastane vprašanje, kdaj je določena uporaba rekurzije, stikanja oziroma kompozicije sporočil smiselna.

Pri zvezah med dvema oddaljenima teleprinterjema poteka komunikacija, ki je sestavljena iz individualnih, standardnih ter mešanih sporočil. Med standardna sporočila sodijo podatki o korespondentu ter časovna, vremenska, dogovorna, delovna sporočila itn. Individualna sporočila nastanejo kot posledica konverzacije med partnerjema, ko se uporablja odprt tekst z različnimi, nepredvidenimi informacijami. Preko mehanizma "vrsta" je moč učinkovito kombinirati individualna, standardna in mešana sporočila s ciljem, da čimbolj natančno zadostimo vprašanjem korespondenta ter da oddajamo tekst z maksimalno možno frekvenco znakov.

Vrsta z vključevanjem sporočil je ugodna tudi tedaj, ko gre za mešana šifrirana standardna in odprta sporočila oziroma za tipična standardna sporočila s spremenljivimi elementi. Ponavljanje važnejših elementov sporočil je pomembno zlasti ob prisotnosti šuma (belega in komunikacijskega) v prenosnem kanalu, ko se kvantitativni elementi (konkretni temeljni podatki) lahko ponavljajo dovolj dolgo, kar je odvisno od individualne ocene operatorja na oddajni strani. Taki ponavljajoči nizi znakov (rekurzija sporočil) se ustrezno prekinjajo z vtipkavanjem znaka 'ESC's konzole.

Vzemimo tale primer stanja v vrsti in definicij sporočila in podsporočil:

FIFO:	ØØ
MESSAGE ØØ:	YOUR_REPORT_IS_ØØ
	LOCATION_IS_Ø1
	MY_NAME_IS_Ø2
SUBMESSAGE ØØ:	589_ØØ
SUBMESSAGE Ø1:	LJUBLJANA_Ø1
SUBMESSAGE Ø2:	TOM_Ø2

Ko postavimo v vrsto niz 'ØØ' (gornji primer), se nam generira z vtipkavanjem znaka 'ESC' tole sporočilo (znak ∇ smo postavili ob vtipkanju znaka 'ESC' s konzole):

YOUR_REPORT_IS_589_589_589_589_5∇

LOCATION_IS_LJUBLJANA_LJUBLJANA_LJUB∇

MY_NAME_IS_TOM_TOM_TOM_TOM_TOM_∇

Na lokacije zadevnih podsporočil lahko seveda postavimo podatke, ki smo jih sprejeli od partnerja, pa čas, zaporedne številke zveze itn. Vse to nam omogoča določeno avtomatizacijo v komuniciranju ter ponavljanje segmentov v različnih delih sporočila, še zlasti tedaj, ko je zveza "uradna" in se moramo držati določenega protokola.

Če postavimo v gornjem primeru

SUBMESSAGE ØØ: 579_ØØ+:

bomo imeli za prvi del sporočila

YOUR_REPORT_IS_579_579_579_5∇+:+:+:

```
        TITLE MESSAGE IN/OUT
        NAME Y17
;*****************************************
;       MESSAGE SETTING PROGRAM
;*****************************************
;   PROGRAMMED BY ANTON P. ZELEZNIKAR
;     FOR MOSTEK SDB-80 (Z-80) ...
;   DATE: JULY 3, 1978
;*****************************************
;   PREVIOUS TO PROGRAM EXECUTION MESSAGE
;     BEGINNING ADDRESS HAS TO BE LOAD TO
;     HL_ VARIABLE. AT THE END OF THE
;     PROGRAM (HL) IS SAVED TO HL_ FOR
;     THE NEXT MESSAGE LOADING. THE BE-
;     GINNING ADDRESS IS PRINTED OUT WITH
;     THE COMPLETE MESSAGE IF IT IS TER-
;     MINATED BY ASCII 'NULL' CHARACTER.
;*****************************************
        ORG 0000H       ;PROGRAM ORIGIN
        LD E,0          ;SET CONTROL F. I/O
        CALL PADD0      ;TYPE BEG ADDRESS
        CALL CRLF       ;TYPE CR & LF
LOOP1:  CALL RDCHR      ;READ CHARACTER
        LD (HL),A       ;PUT CHAR TO MEMORY
        INC HL          ;INCREMENT LOCATION
        CP 0            ;COMPARE FOR 'NULL'
        JR NZ,LOOP1-$   ;IF NZ GO TO LOOP1
        CALL CRLF       ;TYPE CR & LF
        LD HL,(HL_)     ;LOAD BEGIN ADDRESS
LOOP2:  LD D,(HL)       ;TAKE CHAR FROM MEM
        CALL WRCHR      ;TYPE CHARACTER
        INC HL          ;INCREMENT LOCATION
        CP 0            ;COMPARE FOR 'NULL'
        JR NZ,LOOP2-$   ;IF NZ GO TO LOOP2
        LD (HL_),HL     ;SAVE NEXT MESS BEG
        JP RENTRY       ;JUMP TO MONITOR
;*****************************************
RDCHR EQU 0E522H
CRLF  EQU 0E59CH
WRCHR EQU 0E527H
PADD0 EQU 0E604H
HL_   EQU 0FFF4H
RENTRY EQU 0E11DH
;******
        END
```

Slika 9. Lista programa za nalaganje sporočil in podsporočil na dane začetne naslove

ko smo dobili na koncu trikrat niz '+:' zaradi treh celotnih substitucij podsporočila ØØ.

Sporočila in podsporočila vnašamo na ustrezne lokacije s posebnim programom, ki ga imamo na sliki 9. Začetni naslov sporočila shranimo na lokacijo HL_, ki jo inicializiramo pred izvršitvijo programa z direktivo

```
.M :L Ω
:L   FF 00 Ω
:H   FF 10 Ω
```

nato pa program pokličemo v izvršitev. Najprej se natisne naslov, tj. v našem primeru 1000, nato pa vtipkamo želeno sporočilo ali podsporočilo, ki ga končamo z znakom 'NULL'; pri tem se nam vnešeno sporočilo še izpiše (za kontrolo).

7. Absolutni kod programa FIFO in nalagalnega programa

Absolutni kod za oba programa je namenjen neposredni uporabi oziroma reprodukciji programov. Na sliki 10 imamo postopek naložitve štirih paketov oziroma trakov (Y15: program FIFO; Y16: message subroutines; Y13: FIFO subroutines; Y17: message setting program) s pripadajočo simbolno tabelo, iz katere razberemo položaj posameznih spremenljivk. Pred uporabo programa moramo določiti tole:

ZAPOREDJE NALAGANJA PREMESTLJIVIH MODULOV

(1) Program FIFO (Y15) začnemo nalagati od naslova ØØØØH naprej;
(2) za njim naložimo modul MESAGE (Y16);
(3) takoj za naloženim pride modul QUEUE (Y13);
(4) nazadnje naložimo še modul Y17

```
.L 0
BEG ADDR 0000 ┐
EXECUTE  0000 │
END ADDR 005D ├── paket Y15 (FIFO)
UNDEF SYM 05  ┘
*L
BEG ADDR 007E ┐
END ADDR 0111 ├── paket Y16 (MESSAGE)
UNDEF SYM 03  ┘
*L
BEG ADDR 011A ┐
END ADDR 017B ├── paket Y13 (QUEUE)
UNDEF SYM 00  ┘
*T
SYMBOL TABLE (UNDEF=****)

LINADJ 0102   MESAGE 007E   MSGLOC 005E   QUBEG  0112
QUEND  0114   QUINIT 011A   QUMSG  0088   QUPULL 0151
QUPUSH 0126   SUBMSG 0098
*L
BEG ADDR 017C ┐
END ADDR 01A1 ├── paket Y17 (MESS LOADING)
UNDEF SYM 00  ┘
*.
```

Slika 10. Primer naložitve (premestljive, povezovalne) štirih programskih paketov (ustrezno listam na slikah 7, 4, 2 in 9, tj. paketov Y15, Y16, Y13 in Y17) s pripadajočo simbolno tabelo, ki je zlasti koristna za identifikacijo spremenljivk, ki jih moramo inicializirati

(1) QUBEG = 112H: tu naložimo začetek območja vrste, npr. (112) = 2A in (113) = 01, kar pomeni, da začenja vrsta na naslovu 012AH.

(2) QUEND = 114H: tu naložimo konec (natanko konec + 1) območja vrste, npr. (114) = FF in (115) = 0F, kar pomeni, da končuje vrsta na naslovu 0FFFH.

(3) S programom Y17 vstavljamo sporočila in podsporočila na izbrane lokacije. Najprej vstavimo začetek sporočila na lokacijo HL_ (to dosežemo z .M :L↵ itn.) in aktiviramo program Y17 z .E 17C↵. Sporočilo končamo z znakom 'NULL', kar povzroči izpis sporočila in končanje programa Y17. Če je npr. to sporočilo Z4, vnesemo začetni naslov (HL_) v lokacijo MSGLOC +2(i-1) = 5E+6 = 64. Podobno postopamo s podsporočili, le da pri Zi upoštevamo formulo

SUBMSG + 2(i-1)

za naslov, kamor vložimo naslov podsporočila.

Slika 11 prikazuje koda programa FIFO in nalagalnega programa. Na sliki je označena lega naslovov sporočil (MSGLOC), naslovov podsporočil ter začetka (QUBEG) in konca vrste (QUEND). Opisani program FIFO je bil praktično preizkušen z uspehom, in sicer pri hitrostih 110 do 9600 Baud. Pri uporabi tega programa je spočetka priporočljivo, da se ravnamo po diagramu na sliki 8.

8. Sklep

Program FIFO je namenjen tekočemu in informacijsko polnemu komuniciranju med dvema udeležencema po žičnih ali brezžičnih zvezah. Udeleženca razpolagata z majhno mikroračunalniško konfiguracijo, ki omogoča oddajo teksta preko vrste ter vstavljanje standardnih, uradnih in delovnih sporočil. Ker večina navadnega teleprinterskega prometa poteka v Baudotovem kodu s hitrostmi 45,45; 50, 56, 75 in 100 Baud bi lahko k opisanim programskim funkcijam dodali še dva konverzijska programa (ASCII-Baudot in Baudot-ASCII), lahko pa bi podatke sprejemali in oddajali v Baudotovem kodu brez konverzije; v tem primeru bi morali napisati monitor za direktive z Baudotovimi znaki.

Prikazani program FIFO seveda ni edina in najboljša rešitev dane naloge. Vstavljanje znakov v FIFO bi lahko uresničili tudi z zunanjo prekinitvijo (katerega koli drugega programa), ko bi prekinitvena subrutina poskrbela za vpis znaka v vrsto. Izpis znakov iz vrste pa bi udejanili s programirano prekinitvijo (prekini-

```
.M 0,1A1
0000 CD 1A 01 CD 02 01 1E 00 CD 22 E5 FE 1B 28 0A CD   FIFO PROGRAM
0010 26 01 10 F4 CD 02 01 18 EF DB DD CB 77 28 0F CD
0020 22 E5 FE 1B 28 E2 CD 26 01 10 03 CD 02 01 DB DD
0030 CB 7F 28 E5 CD 51 01 FE 00 28 DE FE 24 28 06 57
0040 CD 27 E5 18 D4 CD 51 01 FE 00 28 09 CD 7E 00 FE
0050 1B 28 B5 18 C4 CD 22 E5 FE 1B 28 AC 18 EE FF 00
0060 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00   MSGLOC ADDRESSES
0070 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 CD 33
0080 E5 FE 10 F0 E5 21 5E 00 87 B7 85 6F 30 01 24 7E
0090 23 66 6F CD B8 00 E1 C9 FF 00 FF 00 FF 00 FF 00
00A0 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00   SUBMSG ADDRESSES
00B0 FF 00 FF 00 FF 00 FF 00 1E 00 DB DD CB 77 28 0E
00C0 CD 22 E5 FE 1B C8 CD 26 01 10 03 CD 02 01 DB DD
00D0 CB 7F 28 E6 7E FE 00 C8 FE 24 28 07 57 CD 27 E5
00E0 23 18 D7 23 7E CD 83 E5 FE 10 30 CE E5 21 98 00
00F0 87 B7 85 6F 30 01 24 7E 23 66 6F CD B8 00 E1 23
0100 18 B8 3E 0D CD 26 01 CD 26 01 3E 0A CD 26 01 06
0110 40 C9 FF 00 FF 00 FF 00 FF 00 E5 2A 12 01 22 16   QUBEG | QUEND
0120 01 22 18 01 E1 C9 E5 2A 16 01 23 C5 E5 ED 4B 14
0130 01 B7 ED 42 E1 38 03 2A 12 01 ED 4B 18 01 E5 B7
0140 ED 42 28 08 E1 77 22 16 01 C1 E1 C9 3E 00 E1 18
0150 F8 E5 2A 18 01 C5 E5 ED 4B 16 01 B7 ED 42 28 17
0160 E1 23 E5 ED 4B 14 01 B7 ED 42 E1 38 03 2A 12 01
0170 7E 22 18 01 C1 E1 C9 3E 00 E1 18 F8 1E 00 CD 04   MESS LOADING
0180 E6 CD 9C E5 CD 22 E5 77 23 FE 00 20 F7 CD 9C E5
0190 2A F4 FF 56 CD 27 E5 23 FE 00 20 F7 22 F4 FF C3
01A0 1D E1
```

Slika 11. Kod programa FIFO z lego naslovov sporočil in podsporočil ter kod programa za nalaganje sporočil in podsporočil

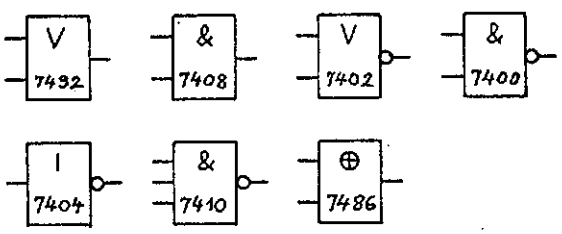
tev s časovnikom) in oddali znak iz vrste, če so pogoji izpolnjeni. V takem sistemu uporabe vrste in izpisovanja sporočil bi bilo mogoče prosti čas mikro računalnika uporabiti za reševanje vzporednih (istočasnih) nalog. Realizacija enega in drugega koncepta vrste pa bo seveda odvisna od koncepta celotnega komunikacijskega sistema oziroma pravil komunikacije, ki naj bi v postopku zveze veljala.

Literatura

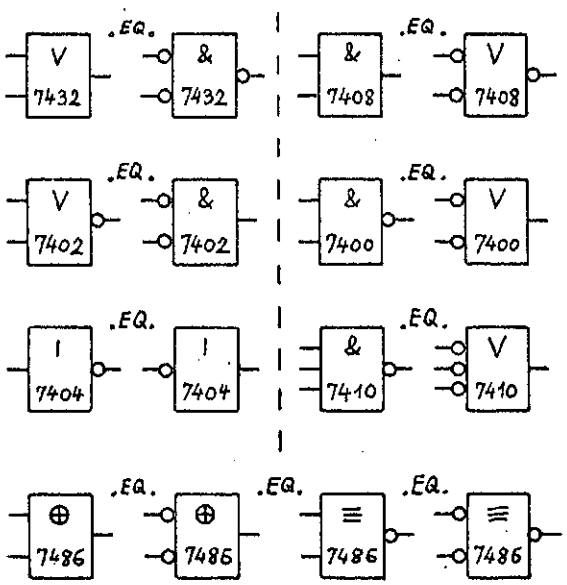
(1) A.P.Železnikar, I.Ozimek, M.Kovačević, D. Novak, Programiranje mikro računalnikov s procesorjem Z 80, Informatica 1 (1977), štev.2, str.5-12.

o sestavljanju logičnih (kombinatornih) vezij

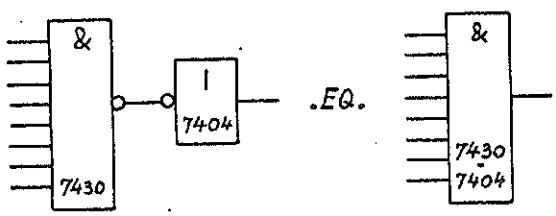
Logična vezja načrtujemo s t.i. pozitivno in negativno logiko (npr. signala 5V in ØV kot logično nazivna signala). Imamo zalogo osnovnih (tehnoloških) elementov, ki jih predstavimo s simboli:

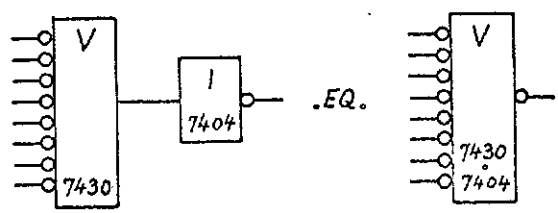


Uvedemo ekvivalence med osnovnimi simboli (vezji), in sicer:



Znaki V, &, I, ⊕ in ≡ v vezjih predstavljajo disjunkcijo, konjunkcijo, negacijo, vsoto po modulu 2 in ekvivalenco. Krogec na vhodu ali izhodu vezja pomeni, da je signal z vrednostjo L (low, npr. ØV, negativna logika) aktiven (namenski) nazivni signal. Vhod ali izhod brez krogca pomeni, da je signal z vrednostjo H (high, npr. 5V, pozitivna logika) aktiven (namenski) nazivni signal. Iz osnovnih vezij dobimo s povezavami sestavljena vezja. Imamo npr.





kjer pomeni 7430•7404 stik oziroma enolično povezavo vezij 7430 in 7404, kot kaže gornja slika. Seveda pa določenih tehnoloških elementov (integriranih vezij) ni mogoče opisati s preprostimi simboli. V takih primerih mora načrtovalec sam ugotoviti ustreznost elementa in mu določiti funcijo v zapletenem vezju. Tipična vezja s sestavljeno logiko so npr. dekoderji, demultipleksorji izbiralniki, primerjalniki, vezja za ugotavljanje parnosti, aritmetična kombinatorna vezja itn.

Dosledno risanje logičnih diagramov s pozitivnimi in negativnimi logičnimi nazivnimi signali (sponkami) omogoča, da načrtovalec v vsaki točki vezja dokaj preprosto preveri logično funkcijo točke in tako razvija in popravlja vezje. Prav zaradi tega je priporočljiva uporaba ekvivalenčnih simbolov iz druge skice ter dosledna uporaba krogcev (označevanje negativnih nazivnih sponk) pri sestavljenih kombinatornih integriranih vezjih.

A.P.Železnikar

univerzalni programator za eprom memorije

m.kovačević
d.novak

Institut "Jožef Stefan" Ljubljana

UDK 621.377.622.25:681.3.065.2

Baveći se mikroprocesorskom tehnikom susrećemo se sa vrlo različitim elementima visokog stepena integracije koji su uključeni u različite module mikroprocesorskih sistema. Priprema mikroprocesorskog sistema za konkretne aplikacije je najčešće povezana sa unošenjem programsko-aplikacijske opreme u EPROM memorijski prostor sistema. Ponenuta šarolikost elemenata visoke integracije zastupljena je i u slučaju EPROM memorija. Time je priprema različitih mikroprocesorskih sistema za aplikacijsku sredinu povezana sa upotrebom različite opreme za programiranje EPROM memorija.

U članku je opisan programator EPROM memorija, terminalskog tipa, koji omogućava programiranje nekoliko vrsta često upotrebljavanih EPROM memorija.

UNIVERSAL EPROM PROGRAMMER - This article describes a terminal type EPROM programmer which enable the programming of usual EPROM chips.

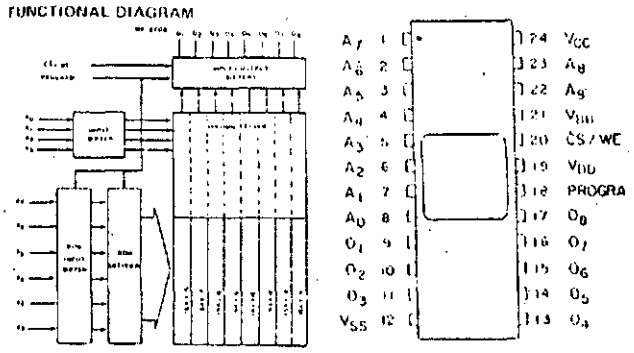
1. UVOD

Značajan produkt savremene MOS tehnologije na području memorijskih elemenata su EPROM memorije (Erasable and Electrically Programmable Read Only Memory). Te memorije tipa "samo čitanje" (Read only) imaju dodatnu ugodnu osobinu koja dopušta brisanje sadržaja. Brisanje ostvarujemo tako da zastakljeni otvor na prednjoj strani takvog integriranog kola (IK) izložimo dejstvu ultra ljubičastog svjetla određenog intenziteta i talasne dužine. Potrebno je višeminutno dejstvo ultra ljubičastog svjetla pa da EPROM postane "izbrisan", što znači da se svi bitovi postave na "0" ili na "1" što se kod različitih tipova razlikuje.

Opisana osobina daje velike mogućnosti primjene EPROM memorija. Svugdje tamo,gdje je povremeno potrebno promijeniti neke parametre ili programske segmente u nekom sistemu,mogu EPROM memorije poslužiti kao vrlo dobro rješenje. Takve memorije se mogu koristiti u sklopu aplikacijskih konfiguracija mikro i mini računarskih sistema kao PSU jedinice (Program Storage Unit) ili kao jedinice za pamćenje tabela, parametara sistema i slično. Pri razvoju aplikacijske programske opreme za sisteme sa višim zahtjevanim sigurnosti rada, mogu ove memorije poslužiti u fazi testiranja takve programske opreme. Onda kada smo sigurni u pravilnost djelovanja razvijene opreme možemo pristupiti formiranju PROM ili ROM memorija koje će konačno biti uključene u sistem.

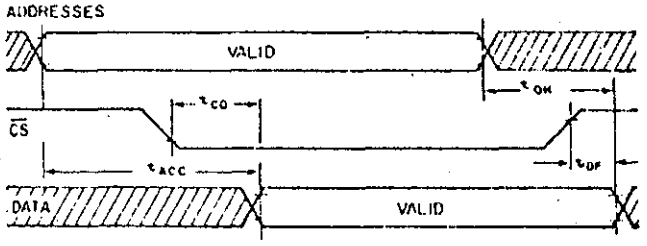
2. OSNOVNA STRUKTURA EPROM INTEGRIRANOG KOLA

Blok šema na slici 1 prikazuje strukturu EPROM integriranog kola MK 2708 firme MOSTEK. To kolo zahtijeva napone napajanja od +5 V, -5 V te +12 V. Sve ulazno izlazne linije su TTL kompatibilne. Memorijske ćelije su organizovane u obliku matrike 1024 x 8. U toku READ načina operiranja mora $\overline{CS}$ signal imati logičnu vrijednost "0", PROGRAM signal mora biti na naponu 0 V.



Slika 1. Funkcionalni diagram za 2708 EPROM

Tipično vrijeme dostupa (Access time) za EPROM integrirana kola je 450 ns, a kreće se u granicama od 150ns do 1 us.

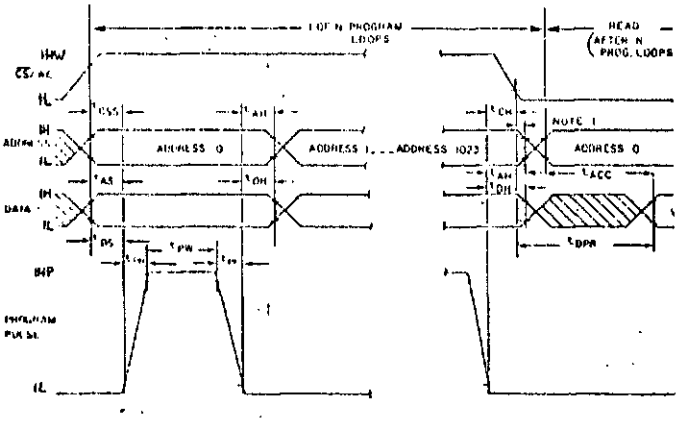


Slika 2. Vremenski diagram za ciklus čitanja za EPROM 2708

U djevičanskom stanju, ili poslije svakog brisanja, EPROM 2708 ima sve bite postavljene na logičnu vrijednost "1". Unošenje informacija postižemo programiranjem "0" u odgovarajuću lokaciju. Kada je jednajsut u neki bit upisana "0" promjena je moguća jedino brisanjem ultra ljubičastim svjetlom. Ovo integrirano kolo dovedemo u modus programiranja tako da na pin $\overline{CS}$/WE dovedemo napon + 12 V. U ovom režimu izlazni pini djeluju kao ulaz za informacije koje se unose u EPROM. Logični nivoi na drugim pinima su isti kao kod ciklusa čitanja. Procedura programiranja teče ovako: pošto se adresne i podatkovne linije stabilizaju mora biti generiran jedan programski impuls (+ 26 V). Svih 1024 lokacija se programira na isti način što čini jednu programsku zamku. Broj zamki (N) je funkcija širine programskog impulsa u skladu sa sljedećom nejednačinom:

$$N \times t_{pw} \geqslant 100 \text{ ms}$$

To znači da je broj zamki minimalno 100 (t_{pw} = 1 ms) do maksimalno 1000 (t_{pw} = 0.1 ms). Brisanje EPROM kola 2708 ostvarujemo izlaganjem zastakljenog otvora djelovanju ultra ljubičastog svjetla talasne dužine 2537 A° te energije 20W/sec cm². Brisanje može da traje 10 do 20 minuta.



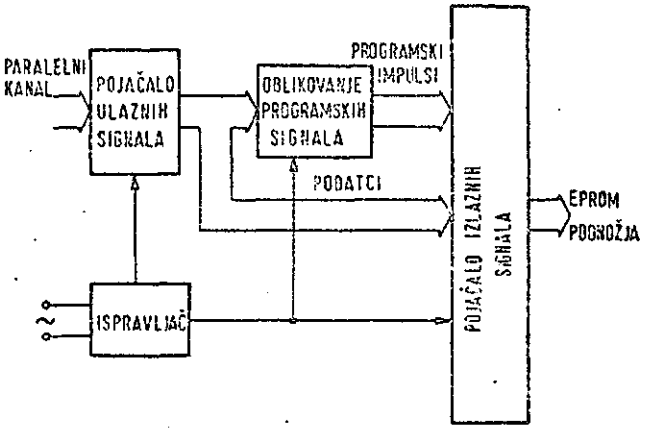
Slika 3. Vremenski diagram za ciklus programiranja za EPROM 2708

Slika 3 prikazuje vremenski diagram za ciklus programiranja. Naponski nivoi sa indeksima IH i IL su TTL nivoi. Indeks HW označava napon od + 12 V, a HP označava gornji nivo programskog impulsa od + 26 V.

3. KONCEPT TERMINALSKOG PROGRAMATORA

Svaka mikro računarska konfiguracija koja se danas može naći na tržištu, pa čak i najminimalniji amaterski kit, sadrži visoko integrirano kolo za priključenje periferije sa paralelnim prenosom podataka. Ta činjenica navodi na zaključak o korisnosti razvoja programatora EPROM memorija terminalskog tipa koji bi se priključivao na paralelni ulazno/izlazni kanal. Takav programator bi dakle, djelovao kao neodvisna ulazno/izlazna jedinica koja bi se mogla priključiti na bilo koji računar pod uslovom da je obezbijeđena potrebna programska oprema za prenos podataka te za generiranje potrebnih programskih signala. Terminal bi morao sadržati sljedeće funcionalne elemente:

pojačala ulaznih signala, kola za oblikovanje programskih impulsa, izvor energije sa potrebnim naponskim nivoima te izlazna pojačala. Blok shema takvog terminalskog programatora je prikazana na slici 4. Univerzalnost, (mogućnost programiranja nekoliko različitih tipova EPROM memorija) postižemo tako da svaki od navedenih funkcionalnih elemenata ima mogućnost izmjene režima rada u skladu sa zahtjevima određenog EPROM-a. Promjene režima rada ostvarujemo sistemom preklopnika te drugim konstrukcijskim rješenjima.



Slika 4. Blok shema terminalskog EPROM programatora

Sa gledišta cijene, gustine integriranih elemenata i kompatibilnosti možemo odabrati sljedeće tipove EPROM memorija kao vrlo zanimljive:

1. 1702, Intel, 256x8 bitova, niska cijena
2. 2704, National, 512x8 bitova, niska cijena
3. 6834, Motorola, 512x8 bitova, niska cijena
4. 2708, National, 1024x8 bitova, povoljna cijena i gustina
5. 2716, Intel, 2048x8 bitova, visoka gustina

Sve navedene tipove EPROM-a proizvode i drugi proizvođači sa više ili manje istim osobinama. Navedeni EPROM-ovi se vrlo često upotrebljavaju što navodi na odluku o razvoju programatora sa mogućnošću programiranja svih navedenih EPROM-ova.

4. ELEKTRIČNI DIO PROGRAMATORA

Najprije je potrebno sistematično proanalizirati električne osobine odabranih EPROM-ova kako bismo mogli pristupiti projektiranju električnog dijela programatora, pri tome se možemo koristiti tabelom 1 koja sadrži potrebne informacije za konstrukciju svakog od već navedenih funkcionalnih modula.

Vrijednosti navedene u tabeli 1 odnose se na PROGRAM modus operiranja odabranih EPROM-ova.

4.1. Ispravljač programatora

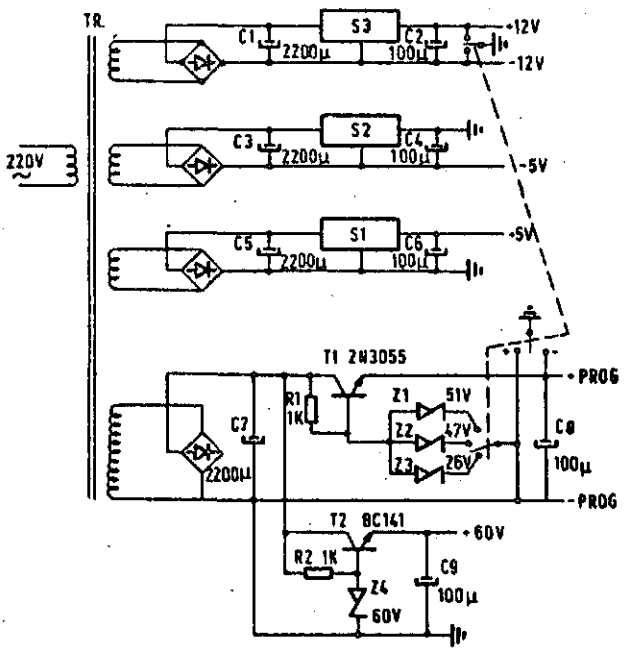
Ispravljač univerzalnog programatora mora zadovoljavati sve strujne i naponske zahtjeve navedene u tabeli 1. Maksimalnu snagu ispravljača ocijenimo na osnovu maksimalnih strujnih i naponskih vrijednosti u tabeli 1, potrebnji

OSOBINA / EPROM	Broj adresnih linija	Naponi			Naponi	Vreme stabiliz.	Maksimalna	Napajanje
		data	address	cs/we	program. impulsa	program. impulsa	struja impulsa	
1702	8	-48 V	-48 V	0	-46 V do -48 V	≥ 100 us	300 mA	+12 V, 10 mA
2704	9	+5 V	+5 V	+12 V	+25 V do +27 V	0,5 - 2 us	20 mA	+5V, 10mA; +12V, 65mA; -5V, 45mA
6834	9	+5 V	+5 V	+5 V	-51 V do -53 V	/	35 mA	+5V, 50mA; -12V, 45mA
2708	10	+5 V	+5 V	+12 V	+25 V do +27 V	0,5 - 2 us	20 mA	+5V, 10mA; +12V 65mA; -5V, 45mA
2716	11	+5 V	+5 V		+25 V	≥5 ns	30 mA	+5 V, 100 mA

Tabela 1. Električne osobine odabrani EPROM-ova

energije u ulaznim i izlaznim pojačalima te ostalim integriranim kolima programator uzimajući u obzir određeni faktor sigurnosti (npr. 2). Konačna snaga transformatora će biti određena na osnovu izračunate snage (po predhodnom postupku) tako da odaberemo najbliže standardne dimenzije jezgra. Možemo ocijeniti sljedeće maksimalne potrošače energije po naponima: Izvor zajedničkog napona programskih impulsa (70 V) mora davati maksimalnu snagu od 4 W; izvor napona od 12 V mora davati maksimalnu snagu od 2 W; izvor napona od 5 V mora davati maksimalnu snagu od 4 W. Ukupna snaga transformatora je tako 10 W.

Namotaji transformatora su sljedeći: primarni namotaj; 70 V sekundarni namotaj snage 4 W; 12 V sekundarni namotaj snage 2 W; +5V sekundarni namotaj snage 3 W i -5V sekundarni namotaj snage 1 W. Električna shema ispravljača predstavljena je na slici 5. Sve napone programskih impulsa formiramo na osnovu jednosmjernog napona od 70 V iz punovalnog ispravljača. Visinu napona biramo promjenom stabiliziranog baznog napona na bazi



Slika 5: Ispravljač univerzalnog programatora

stabilizatorskog tranzistora T1, što ostvarujemo preklopnikom P1. Polaritet napona je određen preklopnikom P2 (na istoj osi sa P1). Po stabilizaciji napona od 12 V moguća je promjena njegovog polariteta preklopnikom P3 (na istoj osi sa P1 i P2). Naponi +5V i -5V su također stabilizovani upotrebom stabilizatora S1 i S2.

4.2. Oblikovanje programskih impulsa

U PROGRAM modusa djelovanja odabranih EPROM-ova moramo dovesti na odgovarajuće nogice impulse, vrlo različitih naponskih nivoa i često dosta visokih apsolutnih vrijednosti. Dakle potrebna je konverzija TTL nivoa pomoću kojih kontrolišemo tok programiranja u naponske nivoe koji neposredno utiču na upisivanje informacija u EPROM. Pri tome se razlikuju dva slučaja; u prvom slučaju se TTL signal pretvara u pozitivni signal sa visokim naponskim nivoima, a u drugom slučaju se TTL signal pretvara u negativni signal sa visokim naponskim nivoima. Slika 6 prikazuje kako se TTL signal pretvara u signal sa pozitivnim naponskim impulsima. Tranzistor T1 na slici 6 prevodi tranzistor T2 iz zatvorenog u otvoreno stanje i obratno u zavisnosti od TTL kontrolnog signala CNT.

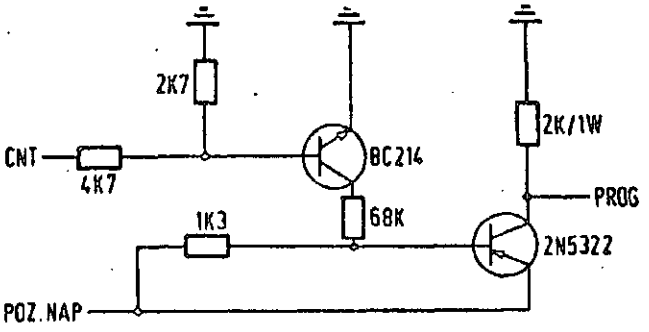
Slika 7 prikazuje kako se TTL signal pretvara u signal sa negativnim naponskim impulsima. Naponski nivo tačke PRG određuje stanje tranzistora T2 na slici 7. Stanje tranzistora T2 određuje tranzistor T1 u zavisnosti od kontrolnog TTL signala. Sklop na slici 7 se upotrebljava pri programiranju EPROM memorija M6834.

EPROM memorije tipa 1702 i slične, zahtijevaju pretvaranje TTL nivoa na više naponske nivoe za sve adresne i podatkovne linije, što ostvarujemo upotrebom jednog tranizstora (NPN) u spoju sa zajedničkim emitorom.

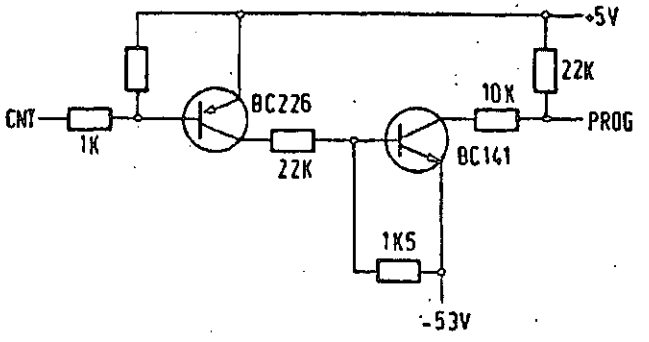
Svi upotrijebljeni tranzistori moraju zadovoljavati zahtjeve iz tabele 1 u pogledu vremena preklapljanja te naponskih i s..ujnih zahtjeva.

4.3. Pojačala signala

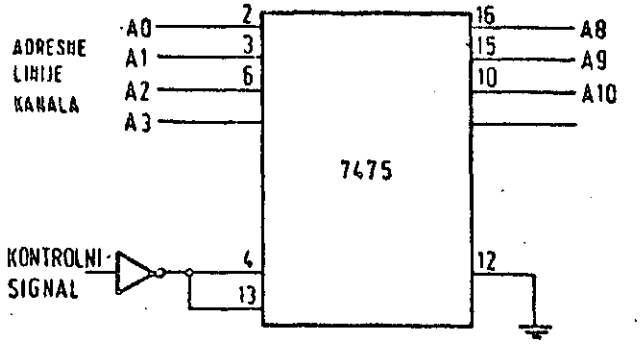
Terminalski univerzalni programator je zamišljen tako da istovremeno rade svi spojevi za oblikovanje signala za sve vrste PROM memorija. Odgovarajućim položajem preklopnika postižemo to da na željenom programskom podnožju dobijemo regularne kontrolne signale za upis informacije u EPROM. Na svim ostalim programskim podnožjima se u pojednostavljenoj verziji programatora mogu pojaviti neregularni signali što znači da ta podnožja moraju ostati slobodna u toku programiranja EPROM-a jedne vrste. TTL linije paralelnog kanala preko kojega



Slika 6: Pretvaranje TTL nivoa u signal sa pozitivnim gornjim nivoom POZ.NAP



Slika 7: Pretvaranje TTL nivoa u signal sa gornjim nivoom +5 V i donjim nivoom od -53 V.



Slika 8: Proširenje broja linija kanala

mikroračunar kontroliše programiranje EPROM-a moraju, dakle, kontrolisati više različitih spojeva za oblikovanje signala, što znači da moramo najprije pojačati linije iz paralelnog kanala. To možemo realizirati upotrebom integriranog kola 7407 koje sadrži šest linijskih drajvera (pojačala). Pojačavanje izlaznih signala (to su signali koji neposredno utiču na upisivanje informacije u EPROM) vršimo u slučaju da želimo istovremeno programirati nekoliko istih EPROM memorija.

Pri programiranju EPROM memorija kapaciteta nad 512 bajtova se pojavi problem pomanjkanja paralelnih linija kanala. Naime predpostavlja se da je paralelni kanal izgrađen nad integriranim kolom M6820 (Peripheral Interface Adapter) ili sličnim. U tom slučaju imamo na raspolaganju 18 ulazno/izlaznih linija što nije dovoljno (10 adresnih linija, 8 podatkovnih linija, R/W, PROGRAM).

Problem riješimo upotrebom memorijskog elementa (integriranog kola) 7475 (4-bit latch). Na taj način proširimo paralelni kanal za 4 linije pri čemu jednu liniju kanala upotrijebimo za kontrolu tog memorijskog elementa. Na ulazne linije tog integrisanog kola vežemo 4 linije kanala predviđene za adresne linije (na primjer A0 do A3). Kontrolni ulaz vežemo na jednu od linija paralelnog kanala. Ispis 10-bitne adrese teče na sljedeći način: na linije kanala, predviđene za adresu, ispišemo bite A8 i A9 adrese desno poravnate (na poziciju bitova A0 i A1). Zatim generiramo kontrolni signal čime se bitovi A8 i A9 zapamte u memorijski element i pojave na izlaznim linijama tog elementa. U sljedećem koraku ispišemo bitove A0 do A7 10-bitne adrese. Dakle na linijama kanala imamo bitove A0 do A7, a na izlaznim linijama memorijskog elementa preostale bitove A8 i A9. Na slici 8 je shema kola sa opisanom funkcijom. Osnovni razlog za širenje broja linija kanala je taj što većina jeftinijih mikroračunarskih sistema ima samo jedno integrirano kolo 6820 (ili slično) u konfiguraciji.

EPROM memorije 2704 i 2708 imaju tu osobitost da upotrebljavaju signal CS/WE (Chip Select/Write enable) sa nivoima +12V i 0V. Dakle potrebno je pretvoriti TTL nivoe na takve nivoe. Taj problem jednostavno riješimo upotrebom NPN tranzistora u spoju sa zajedničkim emitorom. Stanje tranzistora kontrolišemo TTL signalom dovedenim na bazu tranzistora preko otpora od 10 K.

5. PROGRAMSKI DIO PROGRAMATORA

Opisani električni dio terminalskog programatora EPROM memorija ima zadatak da obezbijedi energetske zahtjeve signala za upisivanje informacija u EPROM. Oblikovanje signala se vrši u mikro računaru putem programa. Dakle zadatak programskog dijela programatora je obezbjeđenje pravilnog vremenskog oblika signala. To postižemo ispisivanjem odgovarajućih binarnih vrijednosti na paralelni kanal mikro računara uzimajući u obzir realno vrijeme. Kao osnova za realno vrijeme može poslužiti taktni generator mikro računara ili posebni hardwarski elementi za mjerenje vremena. Najjednostavnije je upotrijebiti posebna integrirana kola za vremenske funkcije (timer, na primjer M6840, Motorola). Međutim i pored visokog stepena integriranosti takvih IK potrebno je određeno iskustvo za uključenje tih elemenata u sistem te se zbog toga preporučuje upotreba hardwarskih vremenskih elemenata u slučaju da ih mikro računarski sistem već sadrži. Mjerenje vremena na osnovu sistemskog taktnog generatora realizujemo jednostavno prebrojavanjem izvršenih naredbi od neke tačke u programu. Osnovna vremenska funkcija koju moramo realizirati je "kašnjenje". To postižemo tako što u program prema potrebi uključimo takav programski segment koji će za svoje izvođenje potrošiti onoliko vremena koliko je potrebno u tom slučaju. Najčešće se pri tome služimo "zamkom za kašnjenje" u kojoj se neki broj naredbi ponavlja toliko puta da se dobije potrebno kašnjenje. Zamka za kašnjenje ima sljedeću strukturu:

definicija broja ponavljanja
tijelo zamke
dekrement broja ponavljanja
test
skok na tijelo zamke u slučaju negativnog skoka

Vrijednost kašnjenja izračunamo po obrascu:

$$T = (nxt + t_d + t_t + t_s)\ N$$

gdje je t vrijeme izvođenja naredbe tijela (tijelo zamke je obično sastavljeno iz nekog broja NOP (No Operation)

naredbi, n broj naredbi u tijelu, t_d vrijeme izvođenja naredbe za dekrement broja ponavljanja, t_t vrijeme izvođenja naredbe za testiranje broja ponavljanja na vrijednost Ø, t_s vrijeme izvođenja naredbe za skok u programu i N definirani broj ponavljanja. Realizacija kašnjenja na osnovu sistemskog taktnog generatora je moguća na svakom mikro računaru bez obzira na tip i konfiguraciju; uz to je i vrlo jednostavna.

5.1. Programski segment za programiranje EPROM memorije tipa 2708

Programske segmente za programiranje određenog EPROM-a realiziramo na osnovu signalnog diagrama za PROGRAM modus djelovanja EPROM-a. Na slici 3 je signalni diagram za PROGRAM modus djelovanja EPROM-a 2708. Na diagramu je prikazana jedna "programska zamka" u okviru koje mora redom biti adresirana svaka lokacija EPROM-a uz primjenu PROGRAM impulsa i podatka. Broj programskih zamki se određuje na osnovu širine programskog signala (PROGRAM). Tok programiranja EPROM-a 2708 bi bio sljedeći: Odredimo područje u RAM memoriji koje treba "prekopirati" u EPROM koji se programira, jednu programsku zamku ponovimo više puta (u zavisnosti od širine programskog signala). Takav programski segment ima sljedeću logičku strukturu uzimajući u obzir već opisani električni dio programatora:

```
I = 1
dountil      (I = N)
  ADRESA = Ø
  dountil    (ADRESA = 1024)
    KANAL <--- WRITE ENABLE SIGNAL
    KANAL <--- ADRESA (BITOVI A9,A8 i A7)
    KANAL <--- KONTROLNI SIGNAL (slika 8)
    KANAL <--- ADRESA (AØ - A6)
    KANAL <--- PODATAK
    KANAL <--- PROGRAM-SIGNAL
    KAŠNJENJE (0.1 ÷ 1 ms)
    KANAL <--- GAŠENJE PROGRAM SIGNALA
    ADRESA <--- ADRESA + 1
  enddo
  I = I + 1
enddo
```

Na sličan način možemo realizirati programske segmente za sve ostale vrste EPROM memorija.

5.2. Programski segment za verifikaciju EPROM memorija tipa 2708

Verifikacija sadržaja EPROM-a 2708 na terminalskom programatoru vršimo na osnovu vremenskog diagrama za ciklus čitanja. Na linijama paralelnog kanala moramo održavati takve signale koji po električnim i vremenskim osobinama odgovaraju onima na slici 2. Verifikaciju vršimo uporedjivanjem sadržaja EPROM memorije u podnožju programatora sa odredjenim blokom informacija u memoriji računarskog sistema koji kontroliše odvijanje ove operacije. Sve potrebne signalne osobine nam obezbjedjuju električna kola programatora (CS/WE možemo postaviti na 0 V, PROGRAM signal, takodjer, možemo držati na naponu 0 V kontrolnim signalima kanala).

Verifikacija teče ovako : najprije se odrede adrese bloka informacija iz memorije te početna adresa bloka u EPROM-u; zatim se po redu na kanal ispisuju adrese lokacija bloka u EPROM-u te R/W signal; neposredno zatim je moguće sa podatkovnih linija kanala pročitati sadržaj odgovarajuće lokacije EPROM memorije. Opažamo da je pri operaciji verifikacije potrebno deklarirati podatkovne linije kao ulazne, što zahtijeva posebne akcije nad samim kanalom (incializacija kanala). U slučaju nepravilnog sadržaja neke od lokacija u EPROM-u ispisuje se poruka na terminal. U pseudo kodu možemo logičnu strukturu segmenta za verifikaciju predstaviti ovako:

```
ČITANJE- adrese bloka u memoriji i početna adresa u
         EPROM-u,
KANAL <--- PROGRAM=Ø - ugasimo programski signal
ADRESA <--- početna adresa u EPROM-u
dountil ( UPOREDILI SMO SVE LOKACIJE )
  KANAL <--- ADRESA
  KANAL <--- READ signal
  PODATAK <---KANAL-podatkovne linije,
  if ( PODATAK je jednak podatku u memoriji )
     then ADRESA = ADRESA + 1
     else  TERMINAL <--- "GREŠKA U EPROM-u"
           ADRESA = ADRESA + 1
  endif
enddo
```

Na osnovu tehnične dokumentacije za EPROM 1702 možemo zaključiti da električni dio opisanog programatora ne obezbjedjuje uslove za operaciju čitanja iz EPROM-a, što znači da nije moguće vršiti verifikaciju sadržine EPROM-a.

5.3. Programski segment za kopiranje EPROM-a u memoriji kontrolnog sistema

Kopiranje sadržaja EPROM-a u memoriji kontrolnog. računarskog sistema, odnosno na odgovarajući vizualni medij (ekran, papir), je vrlo korisna operacija. Time se najčešće koristimo onda kada je potrebno dokumentirati izvršenu operaciju programiranja EPROM -a , ili pak vizualno prekontrolirati sadržaj EPROM-a.

Slično kao pri operaciji verifikacije moramo i pri kopiranju "čitati" iz EPROM-a , što znači da je potrebno linije kanala deklarirati kao ulazne, te na ostalim linijama kanala simulirati operaciju čitanja.

Logični tok operacije kopiranja pokazuje sljedeći pseudo program:

```
INICIALIZACIJA-podatkovne linije na ulazno stanje,
KANAL ---- PROGRAM = ØV,
ČITANJE- adrese bloka u memoriji,
ADRESA-BLOKA ---- početna adresa
ADRESA ---- Ø
dountil ( ADRESA = 1024 )
  KANAL ---- ADRESA
  KANAL <--- READ signal
  ( POČETNA ADRESA ) <--- KANAL-podatkovne linije
  ADRESA <--- ADRESA + 1
  ADRESA-BLOKA <--- ADRESA-BLOKA + 1
enddo
```

Operaciju kopiranja možemo realizirati za sve vrste EPROM-ova izuzev 1702 iz već pomenutog razloga (ne postoji mogućnost čitanja).

5.4. Kontrola izbrisanosti EPROM-a

Upisivanje logične vrednosti "1" u bit EPROM-a 2708 je moguće samo u slučaju da je predhodno izvršena operacija brisanja. Dakle prije no što se odlučimo na programiranje, korisno je ustanoviti u kakvom su stanju ćelije EPROM-a. Programiranju pristupimo u slučaju da sve ćelije imaju logičnu vrijednost "1". Segment za kontrolu izbrisanosti EPROM-a mora čitati redom sadržaje lokacija memorije te

```
                 ******************************************
                 *      BURN 2708 OR EQUIVALENT
                 *
0331 C6 B0       B1      LDAB    £$B0        LOAD NUMBER OF PROGRAM LOOPS
0333 DE 3E               LDX     START
0335 DF 34               STX     XH
0337 DE 36               LDX     BEGA
0339 DF 3A               STX     XREG        STORE BEGIN OF BLOCK ADDRESS
033B DE 34       B11     LDX     XH          PROGRAM LOOP REPEATE
033D DF 3E               STX     START       RESTORE ADDRESS
033F DE 3A               LDX     XREG
0341 DF 36               STX     BEGA
0343 BD 03 60    B12     JSR     OUTADR      PROGRAM LOOP: OUTPUT ADDRESS
0346 DE 36               LDX     BEGA
0348 A6 00               LDAA    X
034A B7 F8 F6            STAA    DATAB       OUTPUT DATA
034D BD 03 9D            JSR     PULSE4      PROGRAM PULSE
0350 86 F7               LDAA    £$F7
0352 B4 F8 F5            ANDA    CONTRA      CLEAR OLD ADDRESS
0355 B7 F8 F5            STAA    CONTRA
0358 9C 38               CPX     ENDA
035A 27 05               BEQ     B13
035C 08                  INX
035D DF 36               STX     BEGA
035F 20 E2               BRA     B12         PROGRAM LOOP
0361 5A          B13     DECB
0362 26 D7               BNE     B11
0364 7E 02 DE            JMP     MAIN
0367 7E 02 DE    B2      JMP     MAIN
036A 7E 02 DE    B3      JMP     MAIN
                 ******************************************
                 *      OUTPUT ADDRESS
                 *
036D 96 3F       OUTADR  LDAA    START+1
036F 48                  ASLA
0370 97 3D               STAA    TEMP+1
0372 96 3E               LDAA    START
0374 48                  ASLA
0375 8A 80               ORAA    £$80        MAINTAIN RW SIGNAL
0377 B7 F8 F4            STAA    DATAA
037A 86 08               LDAA    £8
037C BA F8 F5            ORAA    CONTRA      LATCH HIGH BITS
037F B7 F8 F5            STAA    CONTRA      OF ADDRES
0382 96 3D               LDAA    TEMP+1
0384 44                  LSRA
0385 8A 80               ORAA    £$80        MAINTAIN RW SIGNAL
0387 B7 F8 F4            STAA    DATAA
038A 86 80               LDAA    £$80
038C BA F8 F4            ORAA    DATAA       OUTPUT RW SIGNAL
038F B7 F8 F4            STAA    DATAA
0392 96 3F               LDA A   START+1
0394 4C                  INC A
0395 26 03               BNE     NIH
0397 7C 00 3E            INC     START
039A 97 3F       NIH     STA A   START+1
039C 39                  RTS
```

Tabela 2. Programski segmenti za programiranje, verifikaciju i kopiranje za EPROM 2708

```
                 ******************************************
                 *      PROGRAM PULSE
                 *
039D 86 08       PULSE4  LDAA    £8
039F BA F8 F7            ORAA    CONTRB
03A2 B7 F8 F7            STAA    CONTRB      SET PROGRAM PULSE
03A5 86 60               LDAA    £$60
03A7 4A          P1      DECA                WAIT 0.6 MS
03A8 26 FD               BNE     P1
03AA 86 F7               LDAA    £$F7
03AC B4 F8 F7            ANDA    CONTRB
03AF B7 F8 F7            STAA    CONTRB      RESET PROGRAM PULSE
03B2 39                  RTS
                 ******************************************************
                 *      VERIFY 2708 OR EQUIVALENT
                 *
049D DE 36       V1      LDX     BEGA
049F BD 05 1E            JSR     READPR      READ AT EPROM
04A2 A1 00               CMP A   X
04A4 26 10               BNE     ERR         ERROR
04A6 9C 38               CPX     ENDA
04A8 27 4A               BEQ     EQV         END OF VERIFICATION
04AA 96 37       INBA    LDA A   BEGA+1
04AC 4C                  INC A
04AD 26 03               BNE     VNIH
04AF 7C 00 36            INC     BEGA
04B2 97 37       VNIH    STA A   BEGA+1
04B4 20 E7               BRA     V1
04B6 D6 32       ERR     LDA B   A1
04B8 36                  PSH A
04B9 26 C9               BNE     JHD
04BB CE 02 BE            LDX     £MES7
04BE BD FD A2            JSR     STROUT      ERROR MESAGE
04C1 7C 00 32            INC     A1
04C4 DF 3E       JHD     LDX     START
04C6 09                  DEX
04C7 DF 3A               STX     XREG
04C9 CE 00 3A            LDX     £XREG       OUTPUT ADDRESS
04CC BD FD 1C            JSR     OUT2H
04CF BD FD 1C            JSR     OUT2H
04D2 BD FD 28            JSR     OUTS
04D5 32                  PUL A
04D6 97 34               STA A   XH
04D8 CE 00 34            LDX     £XH
04DB BD FD 1C            JSR     OUT2H       OUTPUT ERRONEOUS
04DE BD FD 28            JSR     OUTS        DATA
04E1 DE 36               LDX     BEGA
04E3 BD FD 1C            JSR     OUT2H       OUTPUT VALID
04E6 CE 02 19            LDX     £MES2       DATA
04E9 BD FD A2            JSR     STROUT
04EC DE 36               LDX     BEGA
04EE 9C 38               CPX     ENDA        END OF VERIFICATION?
04F0 27 0B               BEQ     EQEV
04F2 20 B6               BRA     INBA
04F4 CE 02 9D    EQV     LDX     £MES6
04F7 BD FD A2            JSR     STROUT
04FA 7E 02 DE            JMP     MAIN
04FD CE 02 B1    EQEV    LDX     £MES8
0500 BD FD A2            JSR     STROUT
0503 7E 02 DE            JMP     MAIN
```

dobijene vrijednosti uporediti sa konstantom FF (heksadecimalno). Test je pozitivan ukoliko sve lokacije EPROM-a pokažu sadržaj FF.

I u slučaju operacije kontrola izbrisanosti je isključen tip 1702.

U tabeli 2 su programski segmenti za programiramiranje, verifikaciju i kopiranje EPROM-a 2708. Programski segmenti su pisani u asemblerskom jeziku mikro procesora M6800. Upotrijebljene variable imaju sljedeće značenje:
BEGA- početna adresa bloka u memoriji računara;
ENDA- konačna adresa bloka;
START- početna adresa u EPROM-u;
CONTRA, CONTRB-kontrolni registri paralelnog kanala (M6820);
DATAA, DATAB-podatkovni registri kanala;
XH, XREG, TEMP- pomoćne variable.

5.2. Kontrolni program programatora

Programator mora imati mogućnost vršenja i nekih drugih, pomoćnih, funkcija kao na primjer verifikacija sadržaja EPROM-a, kontrola na Ø, kopiranje u RAM memoriju i slične pomoćne funkcije. Sve to je potrebno primijeniti na sve vrste EPROM memorija za koje je programator projektiran. Dakle potreban je program koji povezuje sve programske segmente, segmente za verifikaciju, segmente za kontrolu na nulti sadržaj te segmente za kopiranje sadržaja EPROM-a u RAM memorije. Glavni zadatak tog kontrolnog programa je komunikacija čovjek-mašina te prenos kontrole na odgovarajuće segmente prema primljenim naredbama.

Logična struktura takvog kontrolnog programa može izgledati ovako:

```
INICIALIZACIJA-početne vrijednosti kanala-adrese,podatci,
program signal na "0", R/W signal na READ ,
TERMINAL <--- "OPERACIJA ?"
ČITANJE-vrsta EPROM-a,
TERMINAL <--- "OPERACIJA ?"
ČITANJE-operacija
case (VRSTA EPROM-a)
  VRSTA EPROM-a = 1702
     POSEBNA INICIALIZACIJA ZA 1702,
     case ( OPERACIJA )
        OPERACIJA = PROGRAMIRANJE
           uključi segment za programiranje 1702
        OPERACIJA = VERIFIKACIJA,TEST ili KOPIRANJE
           TERMINAL <--- "NEDOZVOLJENA OPERACIJA
                          ZA EPROM 1702"
     endcase,
  VRSTA EPROM-a = 2704
     POSEBNA INCIALIZACIJA ZA 2704
     case (OPERACIJA )
        OPERACIJA = PROGRAMIRANJE
           uključi segment za programiranje 2704
        OPERACIJA = VERIFIKACIJA
           uključi segment za verifikaciju 2704
        OPERACIJA = TESTIRANJE
           uključi segment za testiranje izbrisanosti 2704
        OPERACIJA = KOPIRANJE
           uključi segment za kopiranje 2704
     endcase
- - - - - - - - - - - - - -
endcase
```

Takav program je dosta obiman i može da dostigne 1K riječi memorije.

```
                 ********************************
                 *      READ DTA FROM EPROM.
                 *
0515 BD 03 AD READPR JSR    OUTADR     OUTPUT ADDRES
0521 86 7F           LDA A  #$7F
0523 84 F8 F4        AND A  DATAA
0526 B7 F8 F4        STA A  DATAA      OUTPUT READ S
0529 B6 F8 F6        LDA A  DATAB      READ DATA
052C 86 F7           LDA B  #$F7
052E F4 F8 F5        AND B  CONTRA
0531 F7 F8 F5        STA B  CONTRA
0534 C6 80           LDA B  #$80
0536 FA F8 F4        ORA B  DATAA
0539 F7 F8 F4        STA B  DATAA
053C 39              RTS
                 ********************************
                 *      INPUT SEGMENT
                 *

0530 BD 03 B3 INPUT  JSR    SELPR      SELECT EPROM
0540 BD 05 06        JSR    VPIA
0543 DE 36           LDX    BEGA       LOAD ADDRESS
0545 DF 34           STX    XH
0547 DE 3E           LDX    START
0549 DF 36           STX    BEGA
054B DE 34           LDX    XH
054D DF 3E           STX    START
054F DE 36           LDX    BEGA
0551 BD C8      IN1  BSR    READPR     READ DATA
0553 A7 00           STA A  X
0555 A1 00           CMP A  X
0557 26 0E           BNE    NRAM
0559 DE 3E           LDX    START
055B 09              DEX
055C 9C 38           CPX    ENDA       END OF INPUT
055E 27 00           BEQ    E01
0560 DE 36           LDX    BEGA       INCREMENT AD
0562 08              INX
0563 DF 36           STX    BEGA
0565 20 EA           BRA    IN1
0567 CE 02 CC NRAM   LDX    #MES10
056A BD FD A2        JSR    STROUT
056D 7E 02 DE E01    JMP    MAIN
```

Tabela 2 (nadaljevanje).

Programski segmenti za programiranje (B1), verifikaciju (V1) te kopiranje (INPUT) za EPROM 2708. U tabeli su još i podprogram za ispis adrese lokacije koja se programira na kanalu (OUTADR), podprogram za generiranje programskog impulsa širine 0.6 ms (PULSE4) te podprogram za čitanje vrijednosti iz lokacije EPROM-a (READPR).

6. ZAKLJUČAK

Gradnja opisanog programatora ne predstavlja velike teškoće čak ni amaterima; doprinos koji ovakav uređaj predstavlja je vrlo veliki za svakog kome je područje djelatnosti blizu mikroprocesorske tehnike i uopšte digitalne tehnike. To posebno vrijedi ako uzmemo u obzir teškoće oko uvoza ovakve opreme te pomanjkanje i visoka cijena domaćih proizvoda.

LITERATURA

1. John L. Hilburn, Paul M. Lulich: Microcomputers/ Microprocessors: Hardware, Software and Applications, Prentice-Hall, Inc. Englewood cliffs, N.J. 1976
2. Tehnična dokumentacija: National Semiconductors, AMI, Mostek, Motorola, Intel
3. D. Vincent: Low-cost EPROM programmer, Popular Electronics; february 1978.

mesto in vloga uporabnikov informacijskih sistemov v procesu razvoja sistema

v.rajković
i.sirnik

Institut "J.Stefan", Jamova 39
*Izobraževalno skupnost Slovenije, Aškerčeva 9, Ljubljana

UDK 007:681.3

V članku obravnavamo sodelovanje uporabnikov informacijskih sistemov v vseh fazah razvoja sistema, kar pripomore k učinkovitejši sistemski analizi, načrtovanju in delovanju informacijskega sistema. Podanih je nekaj predlogov za oblikovanje metodologije razvoja informacijskih sistemov, ki sloni na sodelovanju širšega kroga uporabnikov.

USERS INVOLVEMENT IN INFORMATION SYSTEM DEVELOPMENT. User participation in the process of information system development is discussed in the paper. The participation leads to better results in information analysis, in system design, and in operation. Some suggestions for a methodology for participative information system development are given.

1. UVOD

S tehnološkim razvojem se povečuje vpliv računalniške tehnologije na učinkovito delovanje informacijskega sistema (IS) in s tem pomembnost ustrezne povezave človeškega in tehnološkega dela sistema. Načrtovanje in razvoj IS je večinoma prepuščen specializiranim skupinam, ki jih običajno sestavljajo predvsem tehniško usmerjeni strokovnjaki (3,9,12, 18,20). Pomanjkljivosti takega načina dela se kažejo v nezadostnem poznavanju resničnih potreb okolja (objektnega sistema), pomanjkanju motivacije s strani sistemskih analitikov in načrtovalcev ter pomanjkanju zaupanja v delo računalnikov tako s strani vodilnega osebja v OZD, kot tudi najširšega kroga uporabnikov (10). Če na grobo povzamemo gre za probleme (neprilagojenost) med človeškim in tehnološkim delom sistema. Referat sloni na predpostavki, da moremo in moramo obstoječe probleme omiliti z ustreznim sodelovanjem neposrednih uporabnikov IS že v fazi načrtovanja sistema. Ne nazadnje je to v skladu z našo pravico in dolžnostjo da tvorno sodelujemo pri stvareh, ki bodo tako ali drugače vplivale na naše življenje in delo.

Analize v ZDA in Veliki Britaniji kažejo, da je več kot polovica prebivalstva na nek način vključena v obdelavo informacij tj. aktivna v okviru IS. Seveda tu ne gre le za računalniško osnovani del, kljub temu pa je vpliv avtomatske obdelave zelo pomemben (8). Pri nas (z ozirom na samoupravno prakso) je ta odstotek kvečjemu večji. Po drugi strani pa iz omenjenih analiz sledi, da se le 3 % prebivalstva lahko pohvali z ustreznim razumevanjem informacijske tehnologije (in le 20 % vodilnega osebja). Rezultat vsega tega je, da od informacijske tehnologije ne dobimo tega kar bi lahko. Slišimo pritožbe kot npr. IS stane več kot je vreden; z uvedbo sodobne informacijske tehnologije smo dehumanizirali življenje in delo; IS ne bomo mogli nikoli razviti tako, da bi izkoristili njegov polni potencial, komu je IS namenjen in čemu služi ipd.

Vsaj del rešitev za te obsežne in zamotane probleme leži v homogenem stiku med tehnološkim in človeškim delom IS, ki ga poleg sistematičnega izobraževanja omogoča tudi vključevanje ljudi v IS že v procesu nastajanja.

2. KAJ SI LAHKO OBETAMO OD SODELOVANJA UPORABNIKOV?

Prispevek k analizi sistema

Definiranje potreb objektnega sistema, ki mu naj bi IS služil, je običajno delo sistemskega analitika, ki v potu svojega obraza zbira informacije, ki misli da so potrebne in koristne za načrtovanje IS (13). Dejanski uporabniki bodočega IS so z večine le pasiven vir informacij. Rezultat tega so običajno težave z neustrezno informacijsko sliko potreb.

Enakopravnejše sodelovanje uporabnikov in sistemskih analitikov (kot informacijskih analitikov in načrtovalcev sistema) pomeni aktivnejši odnos uporabnikov do potrebne informacijske slike dela sistemskega analitika in celotnega IS. Uporabnik čuti svoje probleme drugače kot z večine najeti sistemski analitik. S tem ne le laže zberemo potrebne informacije, ampak dosežemo ravnotežje med uporabniki in tehnologijo, tj. širšo obravnavo problematike tudi v smislu vrednotenja ciljev in razvojnih strategij IS (6,11,14).

Razumevanje sistema

Računalniško osnovani IS naj bi zrasel v OZD kot njen homogeni del in ne kot tujek. Ob kreativnem sodelovanju uporabnikov v procesu nastajanja sistema že od vsega začetka se sistem po naravni poti udomači in ko IS zaživi, je to njihov sistem in ne nekaj kar je vsiljenega od zunaj.

Spremembe kot jih ponavadi prinaša računalniško osnovan IS zahtevajo spremembe v načinu dela in mišljenja uporabnikov (17,19). Zato je potreben čas za razumevanje sprememb in zaupanje med računalnikarji in uporabniki. Vključitev uporabnikov v razvoj sistema pomeni po eni strani motivacijo in možnost za spoznavanje in adaptacija, po drugi strani pa ruši pregrado med tehnologijo in njeno družbeno funkcijo. Uporabniki so deležni zaupanja s strani profesionalnih načrtovalcev, kar se kasneje vrača v obliki zaupanja do novega sistema in razumevanja pomanjkljivosti. To pa pomeni resnično združevanje dela ter s tem skupno lastništvo nad novim sistemom:

Sprotna verifikacija sistema

Udeležba uporabnikov pri razvoju IS oboroži uporabnike s potrebnim znanjem za preverjanje delovanja sistema v skladu z zastavljenimi smotri. Vzpostavi se naravna kontrolna zanka za testiranje nastajajočega sistema, ki deluje nekako samodejno v vseh razvojnih stopnjah (4,7). S tem je omogočena uporabnikom naravna kontrola (upravljanje sistema med nastajanjem) sistema, ki se po drugi strani zrcali v pravočasnem odpravljanju napak in opozarjanju na morebitne šibke točke, kar je v veliko pomoč sistemskim možem na strani informacijske tehnologije (1).

Kot vemo se pri načrtovanju sistema postavlja tudi vprašanje odgovornosti za ustreznost informacijske slike (15). Grobo vzeto se sistemski analitik bori za uporabnikov podpis na identificiranih vhodnih in izhodnih informacijah sistema, s čimer uporabnik prevzema odgovornost za morebitne nepravilnosti. Če v postopku analize sistema uporabnik aktivno sodeluje upada delež doprinosa sistemskega analitika in raste delež uporabnika. Če ne prej bi morali ob zaključku analize uporabnike spoznati, sistemskega analitika le kot svetovalca pri njihovem delu. Na ta način postane uporabnik po naravni poti aktiven lastnik rezultatov analize in lažje sprejme zanje tudi odgovornost.

3. KAKO LAHKO UPORABNIK SODELUJE?

Dosedaj običajni kontakt uporabnik - izvajalec je enostranski v smeri izvajalec uporabniku bodisi v smislu ogledov računalniške opreme demonstracij, tečajev, odgovorov na vprašanja ipd. (2). Za dosego omenjenih ciljev potrebujemo resnični dialog ali z drugimi besedami aktivno sodelovanje uporabnikov. Zastavlja se vprašanje kako v sistemski praksi realizirati združevanje dela uporabnikov in izvajalcev in s tem izboljšati sam proces načrtovanja kot tudi končni proizvod tj. IS.

Zato je možnih več organizacijsko-tehničnih prijemov (3,14), ki se razlikujejo od primera do primera. Pristop, ki bo nakazan v naslednjih vrsticah je le eden možnih in je njegov *namen ta, da ob njem razmišljamo in si skušamo oblikovati* lasten pristop, ki bi bil za konkretno problemsko situacijo najprimernejši.

Ko smo postavljeni pred problem npr. načrtovanje avtomatske obdelave prodaje proizvodov, najprej skušamo identificirati različne skupine ljudi, ki se jih ta problem tiče. To so npr. kupci, prodajni referenti, skladiščniki, odpremljevalci in seveda OZD kot celota. Vsakdo ima v zvezi z rešitvijo problema svoje cilje, ki so neredko tudi nasprotujoči.

Naslednji korak je ustanovitev delovne sistemske skupine, ki jo predstavljajo poleg sistemskih strokovnjakov predstavniki prizadetih skupin oziroma v primeru kupcev človek, ki pozna kupčeve probleme in jih bo pri delu skupine tudi zastopal. Predstavniki različnih skupin imajo poleg dela v *sistemski skupini tudi dolžnost formalno ali neformalno obveščati* svoje tovariše, ki jih zastopajo npr. ostale prodajne referente. Po potrebi lahko sodeluje tudi več predstavnikov ali celo vsa interesna skupina.

Če le čas dopušča je koristno pričeti delo sistemske skupine s preprostim in jasnim zgledom sistemskega dela v obliki konkretne delovne naloge, ki motivira in zbliža ljudi ter uskladi terminologijo.

Za tem najprej identificiramo cilje posameznih skupin. Analiza objektnega sistema poteka v smislu odgovora na vprašanje, kaj določena skupina od sistema pričakuje. Vodja sistemske skupine, ki naj bi bil poklicni sistemski analitik, vodi delo v razgovoru tako, da predlaga drobitev ciljev na podcilje in zahteva argumentacijo. S tem ljudi nekako prisili k strukturiranemu razmišljanju. Vso stvar se pregledno dokumentira tako, da je vedno na dlani razvoj idej.

Naslednji korak je informacijska analiza. Od ciljev preidemo k vhodno-izhodnim informacijskim množicam, ki jih mora sistem dajati, da bi ustregli zastavljenim ciljem. Družno se določijo osnovne opisne karakteristike bodočega sistema. Če ni konfliktov med različnimi skupinami so skupne zahteve *kar unija zahtev skupin. Praviloma pa situacija* ni tako preprosta. Če že ni grobih nesoglasij med cilji skupin pa nastopijo ponavadi nesoglasja glede pomembnosti posameznih informacij in komponent sistema. Pri tem se lahko zatekamo k skupinskim odločitvenim metodam, ki so predmet nadaljnje obravnave.

Analiza objektnega sistema in informacijska analiza predstavljata fazo definiranja sistema, kjer je težišče sodelovanja uporabnikov. V nadaljnjih fazah projektiranja in implementacije sistema, sistemska skupina (v smislu uporabnikov) predstavlja predvsem nadzorno - svetovalno telo. Še posebej to velja v primerih, ko se pri nadaljnjem delu pokaže potreba po redefiniciji sistema, saj vemo, da je ves proces razvoja sistema cikličen. Zato ima omenjena heterogena sistemska skupina ves čas pomembno in odgovorno delo.

4. POSTOPEK ODLOČANJA V HETEROGENI SKUPINI

Omenjena sistemska skupina je prav gotovo heterogena, saj jo sestavljajo ljudje, ki predstavljajo z ozirom na nastajoči sistem različne interesne skupine. V takem kontekstu pa je treba družno odločati v smislu najustreznejših rešitev. Postavlja se vprašanje kako ovrednotiti rešitve in kako izbrati "najboljšo".

Osnovni princip pri reševanju tega problema je jasen, organiziran in argumentiran proces odločanja. Vsakdo si mora biti na jasnem kako in zakaj se sam odloča tako kot se in zakaj drugi drugače.

Za ta namen se kažejo uporabne več parametrske odločitvene *metode (5,16). Problemski prostor alternativnih rešitev* je določen z opisnimi parametri. Tako problemski prostor komunikacije z računalnikom podajajo parametri kot so: enostavnost komunikacije, hitrost, zanesljivost, cena, vzdrževanje ipd. Če nad tem prostorom dodamo še dimenzijo vrednosti, kot odraza parametrov v ciljih tistega, ki odloča dobimo odločitveni prostor. Tako ima npr. za prodajnega referenta veliko vrednost oziroma kvaliteto rešitev, ki zagotavlja enostavno in zanesljivo komunikacijo ob zmirni hitrosti, pri tem pa cena in zdrževanje zanj nista posebej pomembna. Verjetno pa ima drugačno sliko o vrednostih komunikacijske rešitve človek, ki zastopa računski center v OZD. V praksi to pomeni za vsako interesno skupino svojo tabelo, ki po svoje razvrsti alternativne možnosti. Proces usklajevanja poteka skupinsko tako, da se poiščejo kritične vrednosti parametrov v procesu odločanja in se v pogovoru, dogovorimo za kompromisna rešitev. Usklajevanje torej poteka na nivoju parametrov (vzrokov) in ne na nivoju razvrščenih alternativ (posledic) (5).

Pri delu s temi metodami je treba izbrati po možnosti tako, ki je razumljiva in priklodna za skupinsko delo in dopušča medsebojno odvisnost parametrov.

ZAKLJUČEK

Obstojata vsaj dve ključni vprašanji: (1) Kakšna je dejanska korist sodelovanja uporabnikov? (2) Kako naj deluje sistemski analitik v takem postopku oblikovanja sistema?

Če pogledamo na vložen čas in delo potem je jasno, da je

načrtovanje sistema ob neposredni udeležbi uporabnikov dražje. Vendar ravno v zvezi z vrednostjo IS ponavadi denar kot skupni imenovalec odpove. Trud in denar vložen v načrtovanje sistema se obrestuje kasneje saj učinkovitost IS v dobršni meri zavisi od njegove vraščenosti v objektni sistem.

V splošnem naj bi sodelovanje uporabnikov olajšalo delo strokovnjakov čeprav se na začetku zdi, da je ravno narobe. Dragoceni čas in energija se "zapravljajo" za delo z neukimi uporabniki. V ta namen moramo ustrezno preoblikovati tehniko informacijske analize in načrtovanja sistema. Namen tega članka je bil predvsem pokazati na potrebo po metodologiji oblikovanja IS ob sodelovanju uporabnikov in informacijsko - računalniških strokovnjakov, kar je nenazadnje pogojeno tudi z našimi širšimi družbenimi cilji.

LITERATURA

(1) Benjamin, I.R., "Control of the Information System Development Cycle", J.Wiley, 1975.

(2) Boot, R. (ed.),"Approaches to System Design", NCC-Publ., London, 1973.

(3) Couger, J.D., Knapp, W.R. (eds.),"System Analysis Techniques", Wiley, 1974.

(4) Ćećez-Kecmanović, D.,"Okvir izučavanja informacionih sistema", Informatica, Vol. 1, No.1, pp. 25-32, 1977.

(5) Efstation, J., Rajković, V.,"Multi-attribute Decision Making Using Fuzzy Heuristic at Approach", poslano v objavo v IEEE Transaction on Systems, Man,and Cybernetics, 1978.

(6) Grindley, K., Systematics:"A New Approach to Systems Analysis", Mc Graw-Hill, 1975.

(7) Langefors, B., Sundgren, B.,"Information System Architecture", Petrocelli, 1975.

(8) Land, F., "Criteria for the Evaluation and Design of Effective Systems", Proc. of Int. Symp. on Economics of Informatics, Mainz, 1974.

(9) Li, H.D. (ed.),"Design and Management of Information Systems, Science Research", Chicago, Ass., 1972.

(10) London, R.K.,"The People Side at Systems: The Human Aspects of Computer Systems", McGraw-Hill, 1976.

(11) Lucas, C.H.,"The Analysis, Design and Implementation of Information Systems", Mc Graw-Hill, 1976.

(12) Mice, F.G., Turner, S.W., Cashwell, F.L.,"System Development Methodology", North-Holland, 1974.

(13) Mumford, E., "Job Satisfaction: A Study of Computer Science", Mc Graw-Hill, 1977.

(14) Mumford, E., Land, F., Hawgood, J.,"A Participative Approach to Planning and Designing Computer Systems with Procedures to Assist This", Unesco Journal, Vol. 28, No. 3, 1978.

(15) Rajković, V., Majski, J., "Nekateri problemi vrednotenja informacijskih sistemov", Zbornik radova V. Jug. savetovanja o informacionim sistemima, Beograd, 1976.

(16) Rajković, V.,"Vrednotenje računalniške opreme za potrebe reformirane srednje šole", Zbornik del 13. Jug.med. simpozija INFORMATICA 78, Bled, 1978.

(17) Sanders, H.D.,"Computers and Management in a Changing Society", Mc Graw-Hill, 1974.

(18) Sempervivo, C.P., "Systems Analysis: Definition, Process and Design", Science Research Ass., Chicago, 1976.

(19) Sirnik, I., Rajković, V., "Infološki pristop k načrtovanju informacijskih sistemov", Zbornik radova VII. jug. sav. o informacionim sistemima, Beograd, 1978.

(20) Taggart, W.M., Tharp, M.O., "A Survey of Information Requirements Analysis Techniques", Computing Surveys Vol. 9, No. 4, pp. 273-290, 1977.

terminologija u našem računarstvu

d.vitas

UDK 001.4:681.3

Matematički institut
Beograd

Cilj razgovora, koji je reprodukovan u ovom tekstu, je da se formulišu osnovni terminološki problemi u našem računarstvu, kao i da se utvrde osnovni principi za njihovo rešavanje. Ovi problemi su, pre svega, posledica česte upotrebe termina engleskog porekla u našem računarstvu. Diskusija se vodila oko pogodnosti takve prakse, kao i teškoća koje otuda proizilaze.

TERMINOLOGY IN YUGOSLAV COMPUTER SCIENCE. The purpose of the discussion, reproduced in this article, is to formulate the fundamental problems of terminology in Yugoslav Computer Science, and to establich the basic principles for their resolution. These problems arise primarly from the frequent use of English terms in Yugoslav Computer Science. The subject of the discussion was the conformity of this praxis, and the problems to which it gives rise.

1. UVOD

U okviru redovnih sastanaka Seminara iz nauke o računarima, koji radi u organizaciji Matematičkog instituta i Prirodno-matematičkog fakulteta u Beogradu, a pod rukovodstvom dr Nedeljka Parezanovića, održana je 25.aprila 1978. diskusija o terminologiji u našem računarstvu. U razgovoru su učestvovali: akademik dr Mirko Stojaković, sada direktor Matematičkog instituta, dr Dušan Jović, profesor Filološkog fakulteta, dr Nedeljko Parezanović, profesor Prirodno-matematičkog fakulteta, dr Mitar Pešikan, naučni savetnik Instituta za srpsko-hrvatski jezik, dr Slaviša Prešić, profesor Prirodno-matematičkog fakulteta, dr Vidojko Ćirić, profesor na Fakultetu organizacionih nauka, mr Dragan Blagojević i mr Duško Vitas, asistenti Matematičkog instituta.

Razgovor je snimljen. Na osnovu snimka, učesnici su izvršili autorizaciju svoje diskusije.

NAPOMENA

*D.Vitas je izvršio redaktorsku i tehničku obradu ovog teksta.

2. UVODNA REČ

Mr D.VITAS: Informatičari su tokom poslednjih decenija razvili u mnogome specifičan jezik, kako pisani, tako i govorni. Osnovno obeležje ovog jezika su brojni anglo-američki termini, koji ga čine izuzetno nerazumljivim svakome ko nema iskustva sa engleskim jezikom i ne bavi se računarstvom. Glavni uzrok ovakvog stanja je što je većina uvezenih računara (preko 90%) proizvedena u SAD, te se sa uvozom "tela" računara (koje obično nazivamo hardver), uvozi i "duša" računara (softver): programi i različita dokumentacija, pisana na engleskom jeziku. Na ovaj način se spontano u naš jezik unosi jedno novo leksičko polje, zasnovano na engleskoj leksici, koje se formira korišćenjem sledeća tri osnovna postupka:

1. Postupak adaptacije termina, koji se sastoji u morfološkom i fonološkom prilagodjavanju termina našem jeziku (npr. kompilator, bafer, itd.)

2. Postupak integracije termina, koji se sastoji u neposrednom prenošenju termina i što je čest slučaj naročito u govornom jeziku (npr. on line, computer science, itd.).

3. Postupak prevodjenja termina, koji se sastoji u pokušaju iznalaženja dobrog postojećeg ili stvaranju novog domaćeg termina (npr. računar, čitač, bušač).

Pre nego što postavimo probleme koji se javljaju u našoj praksi, pogledajmo letimično kakvo je stanje u drugim jezicima.

Raznovrstnost i snažno širenje računarske tehnike su doveli da se već 1960. godine (IFIP) preduzmu koraci ka standardizaciji engleske terminologije i stvaranju rečnika iz oblasti računarstva. Ipak, postignuti rezultati govore uglavnom o težini problema.

U Francuskoj, navedimo primer odlične Meinadier-ove knjige (Structure et fonctionnement des ordinateurs, Larousse, 1972) u kojoj, uprkos brojnim uspelim terminološkim rešenjima, nailazimo na reči:
pipe-line, software, hardware, firmware, bus, flip-flop, interface, tag, itd. Kao zamenu za ključne reči software i hardware, na drugim mestima ćemo naići na adaptaciju softouare i hardouare ili na programmerie i matériel de traitement, ili pak na termine logiciel i matériel, koje je preporučila francuska Akademija.

U sovjetskoj literaturi iz oblasti računarstva se može zapaziti sličan fenomen.

Pregledali smo dve edicije izdavačke kuće МИР (Библиотека Кибернетического Сборника и Математическое обеспечение ЭВМ) gde u različitim delima nailazimo na termine:
супервизоры , дампинг, интерфейс, таймер, файл on-line файл, дисплей , транспация, компилятор, байт, стек, листинг, флаг, itd. Ovi primeri nam možda mogu nametnuti zaključak da ni u SSSR nije pronadjena zadovoljavajuća zamena za navedene anglicizme, utoliko pre što navedeni primeri potiču iz knjiga objavljenih u poslednjih nekoliko godina.

Pogledajmo sada kakvi se anglicizmi javljaju u našoj literaturi.

U englesko-srpskohrvatskom rečniku termina iz oblasti informatike, koji je 1974. izdao Intertrade, IBM-ov zastupnik za SFRJ, uprkos očiglednom naporu da se iznadje podesan domaći termin, nailazimo i na kompajler (ali i na kompilator i na prevodilac (p.78); no, translator (p.385) je , takodje, prevodilac) zatim asembler, datoteka (latinski koren + grčki sufiks), softver (ili opremljenost programima (?)), hardver (ili opremljenost mašinama (?)), bit, bajt, ploter (ili crtač krivih), debag, pufer(za buffer) dok je push-down stack (p.346) = vraćena gomila (?). U časopisu Informatica 2 (1977), u listi srpsko-hrvatskih ekvivalenata engleskim terminima srešćemo asembler, bafer, kompajler, drajver, editor, flip-flop, hardware, software, monitor, čip, interfejs, formater,itd.

Prisustvo sličnih engleskih termina se još snažnije oseća u svakodnevnom radu po računskim centrima, gde već i najjednostavnija komunikacija sa računarom podrazumeva poznavanje engleske terminologije.

Navedeni primeri otkrivaju bar delimično, terminološke teškoće sa kojima se susrećemo, te se prirodno postavljaju sledeća pitanja: Kako se odnositi prema anglicizmima? Kako ih koristiti ako nemamo pogodan domaći termin, i kada, uopšte, treba tragati za zamenom? Posebno, kako se odnositi prema engleskim rečima sa grčkim i latinskim korenom?

3. DISKUSIJA

Dr D. JOVIĆ: Ja znam: ne bi trebalo da prvi počinjem diskusiju, već bi to trebalo da učine matematičari. Medjutim hteo bih da kažem nekoliko stvari sa gledišta opšte lingvistike.

Pitanje terminologije nije samo naše, tiče se celog sveta.

Čak i u onim jezicima, u kojima se neguje strogo puritanstvo, gde se teži očuvanju svoga, da se autentično prevede svaki termin na vlastiti jezik, kakva je, npr. praksa u islandskom, japanskom, delimično madjarskom; ali se i tu potpuno ne uspeva. Jednostavno, tako nešto nije moguće. Pitanje je sada kako to u našoj praksi razrešiti. Lingvistički je ovo sasvim jasno, a sa matematičke tačke gledišta neću govoriti, jer ste vi, matematičari tu daleko sigurniji. To se može razrešavati na nekoliko načina. Mogu se graditi novi termini, ali je to, inače, vrlo težak posao, i mislim da ne bi time doprinelo većoj jasnosti celog pitanja. U svakom slučaju, sagraditi svoj vlastiti termin, ili preuzeti strani, zahteva da se opisano objasni. Urukčije nije moguće. Uostalom, strani termin je po pravilu nužno na nekakav način adaptirati vlastitoj jezičkoj normi. Pa ako je to već tako, izgleda mi da je bolje, ako je reč o nekoj vrlo stručnoj i uzanoj oblasti, uzeti onaj termin koji ima najveću frekvenciju u svetskoj literaturi, ako takav termin već postoji. Naravno, i u tom slučaju se mora dati objašnjenje.

Ja mogu da kažem da smo kolega Pešikan i ja slušali opštu lingvistiku kod pokojnog prof. A.Belića, čoveka istančanog jezičkog ukusa,i koji je mnogo vodio računa da se u jezik ne unosi sve i svašta, pa je ipak uvek stajao na gledištu: bolje je u datim slučajevima uzeti stranu reč nego praviti novu, na našim osnovama, jer su takve reči po pravilu motivisane, vrlo često stvaraju zabunu, a i takva reč se inače mora objasniti. Uostalom, ono novo što smo stvarali, po pravilu, nije uspevalo. Sasvim je drugo, naravno ako se kod nas stvara neka nova naučna oblast, za koju nema uzora u drugih. U tom slučaju je normalno da terminologiju stvaramo prema našem vlastitom jeziku. Pitanje je, naravno koje i kakve termine uzeti kad pozajmljujemo. Mi svakako ne možemo uzeti termin u "čistom vidu" onako kako se, npr. govori u Engleskoj, jer bi smo tako morali menjati svoju vlastitu jezičku normu. Moramo ga na nekakav način adaptirati. Mislim, neophodno ga je bar fonološki adaptirati, ako drugo nije neophodno.

Mislim da je dobro što u nekim zemljama postoje mnogi naučni rečnici, dvostrani. Npr. u sovjetskoj literaturi: rusko-engleski, ili englesko-ruski za pojedine oblasti itd. Postoje za mašinsko prevodjenje, u pojedinim oblastima tehnike itd. Iako je svaki takav termin na nekakav način opisan, vidi se jasno njegovo značenje.

Dr M.STOJAKOVIĆ: Ja na ove probleme ovako gledam: mislim da računarska tehnika doista utiče na formiranje jezika ali se razvija tako da će, po mome mišljenju, ovo što mi sada radimo biti napravljeno na kratko vreme. Ide se ka tome da se formule, koje se normalnim matematičkim jezikom pišu, u računar prenose onako kako je uobičajeno. Osim toga, već postoje mašine koje primaju komande govornog jezika i možda ćemo uskoro sa računarom razgovarati kao sa robotom. Neće biti potrebno toliko specijalnih termina. Termine koje sada pravimo računska tehnika će da prevazidje. To, s jedne strane. A sa druge, gledajući ovaj naš posao, makar i na kratko vreme ga obavljali, ne možemo sve rešiti za jedan čas. Predlažem da sa rešavanjem danas započnemo tako što ćemo formulisati principe kojih se treba držati a vremenom terminologiju doradjivati i izgradjivati. Ovaj skup, mada kompetentan, ne držim da je i kompletan. Prema tome, ovaj skup može da inicira rešavanje problema i moj je predlog da tako i postupi.

Za početak ću pomenuti termine informatika i informacija. Čak ni tako značajni termini, kao što su naslovi pojedinih disciplina, nisu svima, pa čak ni matematičarima, sasvim jasni. U Novom Sadu je jedan doktor matematike pripremio program za informatiku. Kada smo ga pogledali, videlo se da je to program za teoriju informacija (propusna moć kanala veze, kodiranje, itd.). Informatika proučava obradu informacija, njihovo prikupljanje, čuvanje, sistematizovanje, eksploataciju a nikako matematičku logiku, teoriju formalnih jezika, mehaničko dokazivanje teorema ili pak teoriju automata. A kada tako stoji stvar sa osnovnim pojmovima, kako tek stoji sa pojmovima kao što su buffer, debugging, kompajler i njima slični? Trebalo bi stoga poći od osnovnih termina pa postupno precizirati šta u koju oblast ulazi.

Moj je zaključak s toga vrlo kratak: Ovo je početak a kraj ćemo videti gde će biti. Ja ovoga puta ne bih govorio o pojedinim terminima. Zadržavam pravo, ako bude vremena, da još jednom uzmem reč kao što će to učiniti i drugi kad dodje do rasprave oko pojedinih termina: da li ga prevoditi ili ne prevoditi na srpski, uzeti ga neposredno iz stranog jezika i tome sl.

Dr M.PEŠIKAN: Vrlo teško mi je govoriti o terminima u ovoj oblasti zato što mi je potpuno nepoznata, što sa njom nisam imao dodira ni u srednjoj školi, za razliku recimo od astronomije, fizike, hemije itd. Prema tome, mogu govoriti samo o nekim opštim i načelnim stvarima, o problemima stručne terminologije koji su zajednički za razne struke, izmedju ostalih i za ovu računarsku.

Pomenuti problem engleskog jezičkog materijala u terminologiji pritiska-izgleda-čitav svet. Naime, to nije prosto pitanje domaće i strane reči. U stranu leksiku spadaju i grčke i latinske reči i leksički elementi. Medjutim, veoma se dugo evropska kultura na njih privikavala i svi su jezici osposobljavali svoje sisteme i mehanizme da mogu grčko-latinske elemente primati na pogodan način, uklapati ih u svoj izraz, tako da su oni dobili mesto drukčije nego pozajmice iz bilo kog današnjeg jezika (engleskog, francuskog, nemačkog i dr.). Stvorila se i predstava o značenju tih elemenata, jer nam je malo ostalo grčkih i latinskih leksičkih elemenata koje možemo upotrebiti za nove termine - a da oni već nisu upotrebljeni u postojećoj terminologiji bilo koje grane, zahvaljujući čemu imamo odmah neku predstavu i o značenju novog termina.

Ovde, za anglicizme koji su pomenuti, sasvim smo u drugoj situaciji. Za njih naši postojeći jezički fondovi ne daju nikakvo predznanje, ništa se o značenju tih termina ne može zaključiti na osnovu materijala naše jezičke kulture, nego samo na osnovu znanja engleskog jezika, što se onda odnosi samo na one koji ga direktno znaju.

Veliki problem engleske terminologije, za naš jezik i za mnoge druge, zapravo je u tome što nemamo već gotove načine adaptiranja engleskih termina, njihovog prilagodjavanja da se dobro uklope u naše sisteme. Naprotiv, za grčke i latinske elemente mi imamo dobre i gipke načine uobličavanja, njihovog primanja u obliku bilo imenica bilo glagola bilo prideva. Imamo čitav niz sufiksa, prefiksa i drugih formanata, tako da veliki deo terminologije zasnovane na grčkim i latinskim osnovama ne moramo preuzimati gotov negde iz inostranstva, nego je možemo i sami praviti od elemenata koje imamo u svom tvorbenom potencijalu. Sa engleskim to još nije slučaj, a teško će se i razviti takvi mehanizmi, jer je znatno veća strukturalna razlika našeg jezika od engleskog nego od grčkog i latinskog. Ta veća srodnost ogleda se u tome što i ovi klasični jezici i naši slovenski predstavljaju jezike veoma razvijene fleksije. Čak imamo i odredjenu sličnost ili dobru korespondentnost nastavaka (npr. oblici na -a za ženski rod i množinu srednjeg roda, znatna podudaranja u sistemu gramatičkog roda i dr.).

Možemo, dakle, reći da nemamo više težih problema za adaptiranje reči grčke ili latinske osnove, i dobro bi bilo da to iskoristimo kao jedno praktično pravilo koje će nam pomoći u prilagodjavanju jednog dela anglicizama koje preuzimamo (kad ne nadjemo način da ih prevedemo terminima od domaćih osnova): ako su engleski termini grčke ili latinske osnove, treba da primenimo na njih one uhodane mehanizme prilagodjavanja koje smo već mnogo puta primenili u raznim oblastima praveći termine od grčke i latinske osnove. Prema tome, ne bih bio za uobličavanje tipa kompajler, nego za naknadno vraćanje na lik kakav bi bio da nam termin dolazi neposredno iz latinskog odn. grčkog jezika (oni su u izvesnom smislu spojili svoje fondove kao gradju današnje terminologije). Tu bi moglo doći bilo kompilar, bilo kompilator, ili uopšte bilo koji oblik koji će zvučati kao reč iz klasičnih jezika i imati direktne analogije u našem već postojećem fondu. Uz ovakav postupak se problem adaptacije anglicizama prepolovljuje, jer je za nas srećna odlika engleskog jezika (gotovo bih rekao: jedina srećna) što je u njemu veliki broj reči latinskog odn. romanskog porekla, tako da veliki broj termina koje preuzimamo iz engleskog možemo na pokazani način preraditi. Tu naknadnu preradu na latinski lik primenjivali smo dosta u istoriji naše kulture u odnosu na francuske reči. Mnoge reči koje zvuče kao latinizmi uzeli smo zapravo od Francuza, ali ih nismo uobličavali prema francuskom nego kao da smo ih preuzeli iz latinskog, na primer imenice na - cija a ne na - sjon.

Neka je to samo pseudolatinizam, to će se možda nekome učiniti da je to neka vulgarizacija pridavati lažni latinski oblik reči koja nije bila latinska. To ne smeta, jer je cilj naše jezičke kulture osposobiti terminologiju da kod nas dobro funkcioniše, a ne da izdrži, da tako kažem - nekakve provere pedigrea.

Drugo pitanje: u kojoj meri treba da se trudimo da stvaramo našu, "našku" terminologiju, tj. od domaćih ili potpuno odomaćenih reči (kakva je i račun)?

Za ovo nema nekog čvrstog pravila, ali jedno orijentaciono postoji. Naime, za one ter-

mine za koje se ostali važniji evropski jezici zadovoljavaju internacionalnom rečju, treba njome dase zadovoljimo i mi, kao deo svetske kulturne zajednice. To ne znači zabrana domaće reči, naprotiv - ako nam se ona ponudi, ako nadjemo ili prosto imamo dobar domaći termin, možemo i njega upotrebljavati, ali nikako ne progoneći naporedni internacionalni, pravdajući to borbom protiv varvarizama. Jedna je stvar internacionalizam a druga varvarizam. Primanje internacionalizama ima, istina, odredjenu cenu i znači izvesno opterećenje našeg izraza, ali ima i odredjenu prednost, jer olakšava izlaženje na svetske prostore i sudelovanje u medjunarodnoj kulturi.

Medjutim, drukčije se odnosimo prema onim terminima za koje i ostali narodi pribegavaju domaćem terminu, zasnovanom na nacionalnom jeziku. Tu onda nikako nije dobro - ako nadjemo iole dobru mogućnost u našem leksičkom fondu - -preuzeti anglicizam, germanizam ili bilo koji drugi varvarizam, jer, kao i svaki strani jezički elemenat, opterećuje naš jezik, a ne kompenzira to onim prednostima koje ima internacionalni termin.

Trenutak kada se može dobro intervenisati u terminologiji jeste upravo dok je ona nepoznata u širim krugovima, i u ovoj grani vi ste u stvari u dobroj poziciji, koja vam omogućuje da nešto uradite. Čoveku je naprosto lakše da citira, nego da traži izraz i da upotrebi strani, preuzeti termin nego da pravi novi, jer postoji odredjeni otpor sredine da ga prihvati, naročito isprva , i potrebna je odredjena smelost da se stvori novi termin. Ne treba se plašiti neuspelih pokušaja, naročito ako navodimo naporedo preuzeti i napravljeni termin. Možda novi termin uopšte neće biti prihvaćen, ali će možda vremenom i prestati otpor prema njemu. Mi možda svi govorimo kuplung, ali smo se već manje-više privikli i na reč kvačilo, govorimo volan ali i upravljač, a isprva su te reči morale biti neobične i ličiti na nekakav preosetljivi purizam, ali su postepeno, istrajnom upotrebom stekle svoje mesto.

Ono što je rekao kolega Vitas na početku jako je važno, jer kad bi ova terminologija računarstva ostajala u nekakvom uskom stručnom krugu, onda se praktično ne bi vredelo tu šire angažovati, u uskim krugovima ljudi će se nekako snaći i naći način da se sporazumeju. Ali mi ne možemo danas.ni za najspecijalnije oblasti znanja ili nauke garantovati da će pojmovi i termini ostati zatvoreni u stručne krugove. Jedan deo te terminologije neizbežno izadje na šire prostore, te ako je srećno uobličen, on dobro dopunjava i bogati jezik, a ako nije, on ga opterećuje. Pomenuo je kolega Jović hrvatsku praksu. U vezi s njom, ja pomalo nagovaram našu beogradsku sredinu da iskoristimo onaj purizam koji se tamo ispoljava jer se oni mnogo više nego mi ovde u Beogradu trude da pronalaze domaće zamene za strane termine. Oni su nama time u dosta slučajeva pomogli savladjujući onaj prvi otpor novoj reči, a kad se stvorila potrebna navika, mi smo lakše mogli da preuzmemo domaći termin. Naravno, ima slučajeva preterivanja u traženju domaćih zamena, i nikad se neće moći to nastojanje sprovesti do kraja, sa čim se treba potpuno pomiriti. Ne verujem da će ijedna stručna grana moći potpuno prečistiti terminološke probleme i imati idealnu terminologiju - -bilo kako da je zamisli.

Treba, medjutim, da znamo šta nam je cilj kojemu ćemo što bliže primicati stanje naše terminologije. U pogledu odnosa stranih i domaćih elemenata cilj treba da nam bude da onde gde nema internacionalnog termina imamo dobar domaći; da svaki internacionalni termin ima mesta u našoj jezičkoj kulturi, ali je dobro da za što veći broj pojmova koji imaju značaja za opštu obrazovanost (pre svega za osnovno obrazovanje) pored internacionalnog imamo i domaći termin, koji lakše savladavamo. Jer da bismo se dobro služili nekim terminom, obično moramo razumeti njegov sklop. Na primer, lingvističkim terminom disimilacija stvarno vlada samo onaj, ko - bilo iz neposrednog učenja latinskog, bilo posredno, uporedjenjem sa sličnim rečima - zna da ovde prefiks znači "raz-", osnova "sličan", a sufiks - acija da pokazuje neki proces; naprotiv, sinonimni termin razjednačavanje razumeće i dete. To znači da na nižim stupnjevima obrazovanja uvodjenje u internacionalnu terminologiju treba da bude oprezno i postupno, a da je domaći termin kad god je dobro napravljen i dovoljno običan pogodniji od stranog. Medjutim, takav domaći termin nemamo uvek, i nije realno ni mogućno da ga uvek imamo; tu se možemo samo kretati jednim pravcem, a nikada dostići idealni cilj.

Što se tiče pomenutog problema značenja termina informatika- to će se rešiti na medjunarodnim relacijama, ono što bude u svetu značio - značiće i kod nas. Naša terminologija mora biti usaglašena i korespondentna sa medjunarodnom. S tim u vezi, ne treba se pri pokušajima prevodjenja termina sa engleskog čuvati ni pojedinih dosta bukvalnih prevoda, jer je onda jasan odnos jednog termina prema drugom. Za latinsko ime kola životinja Arthropoda imamo naše sinonime termine zglavkari i člankonošci; prvi termin može i izazivati nedoumice šta znači, ali termin člankonošci samim svojim sklopom jamči da je isto što i Arthropoda. Prema tome, ima i izvesne prednosti termin koji je svojim sklopom veran internacionalnom modelu.

Mr D.BLAGOJEVIĆ: Činjenica da se engleske reči koje nemaju latinski koren teško prevode na srpskohrvatski jezik, ne bi trebalo da nas odvrati od pokušaja da za njih ipak nadjemo odgovarajuće izraze u našem jeziku. Praksa je pokazala da je ponekad potrebno i duže vreme da se dodje do zadovoljavajućeg prevoda za neku reč. Naravno, ne treba po svaku cenu zamenjivati inostranu reč našom pravljenjem rogobatnih kovanica. S druge strane, u velikoj meri je stvar navike da li će nam neki izraz zvučati rogobatno ili neće, tj. ono što nam u prvi mah zazvuči rogobatno može nam posle izvesnog vremena izgledati sasvim prihvatljivo.

Dr S.PRESIĆ: Mislim da su ova pitanja neobično zanimljiva i složena. Pošao bih od onoga što je kolega Blagojević na kraju rekao. U skladu sa tim rekao bih, da po mom mišljenju, napor za gradjenje novih reči mora uvek biti prisutan. Pokušaću da navedem argumente zašto tako mislim. Svima nam je dobro poznato da je najvažnija uloga jezika da nam pomaže u vodjenju razmišljanja, vodjenju misaonog procesa. Iluzurno je pomisliti da je većina od nas u stanju da toliko dobro nauči strani jezik, recimo engleski, da bi sa lakoćom mogla razmišljati isključivom upotrebom tog jezika. A to kao argument u daljem razmatranju je itekako važno. Pomoći ću se navodjenjem jednog jednostavnog primera iz svoje prakse. U matematičkoj logici pored raznih drugih reči koriste se i reči disjunkcija i konjunkcija. Kao odgovarajuće reči našeg jezika obično se upotrebljavaju reči rastav i sastav. Moglo bi se pomisliti da je razlika izmedju tih naših reči i prethodnih latinskih veoma velika. Medjutim, da tako nije ukazuje naredno obrazloženje. U rešavanju mnogih matematičkih problema prirodno se pojavljuje da se

formula kojom je izražen, iskazan problem prevede, transformiše u novu formulu koja je disjunkcija izvesnih jednostavnijih formula. (Strože rečeno, datu formulu prevodimo na neku njenu disjunktivnu formu, recimo na tzv. kanonsku disjunktivnu normalnu formu). U takvom slučaju rešavanje problema je rastavljeno, raščlanjeno na rešavanje tih jednostavnijih formula. Da li bismo u vezi sa rečenim mogli umesto reči rastavljeno upotrebiti reč kao disjunktvoli i sl. budući da se radi o disjunkciji? Očigledno, ne. Naša reč rastav, odnosno rastavljen pojavljuje se prorodno. Čak smo ukazanu opštu činjenicu o prevodu date formule na disjunkciju jednostavnijih mogli i ovako izreći: u matematici se rešavanju mnogih problema pristupa tako što se dati problem, može se tako reći, rastavlja na izvestan broj jednostavnijih.

U vezi sa gradjenjem novih reči podvukao bih sledeće. U matematici reči često po pravilu, po dogovoru, reći ću čak i po naredbi, dobijaju željeno značenje. A to je u skladu sa činjenicom što smo u nemogućnosti da za ono što želimo naravimo potpuno odgovarajuću reč, reč koja će sve podrobno opisati. Uostalom, i u dosadašnjoj matematičkoj praksi mnogo je primera u kojima nove reči ne pružaju baš potpun opis, odnosno opis kakav želimo. Uzmimo primer reči trougao. Svi smo dobro navikli da je to celina koja ima više sastavaka kao: tri temena, tri stranice, tri ugla, tri visine, itd. Pa da li je rečju trougao sve to "pokriveno". Svakako, ne (U stvari, rečju trougao ističu se jedino uglovi, tj. samo jedan deo sastavaka). Umesto reči trougao mogli smo uzeti recimo svaku od reči
trostranik, trotemenik, trovisinik,..
kao i razne druge. Medjutim, i svaka druga izabrana reč bi imala po koji nedostatak.

U slučaju novih reči najveće teškoće u prihvatanju pruža nenaviknutost na njih. Naime, nova reč često zvuči neobično, usiljeno i sl., ali posle odredjenog trenutka takav se utisak u mnogim slučajevima gubi i, slobodnije rečeno, nova reč preživi. Skoro jedini zahtev koji valja imati na umu pri traženju reči koja treba da odgovara definiensu (koji nam je poznat) jeste da nova reč bude što lakša za izgovor (nije "rogobatna") i, naravno, bude u duhu jezika.

U razgovoru su pomenute engleske reči softver i hardver i istaknute su teškoće u traženju naših odgovarajućih. Da li te reči idealno odgovaraju svojim namenama? Grublje rečeno, skoro nikakve veze nemaju sa onim na šta se odnose. Ona slovenačka verzija, odnosno upotreba reči oprema je, čini se, mnogo podesnija. Kratko, i u slučaju tih dveju reči treba da potražimo neke reči koje bi mogle podesno i lepo poslužiti, ali opet i tu ne na silu boga; ako ne možemo stanimo, sačekajmo. Konačno, ne možemo za jednu noć napraviti sve reči koje nam nedostaju. Značajno je da napori za gradjenje što boljih reči budu stalno prisutni. To pored ostalog znači da u pravljenju novih reči treba biti i slobodniji, odnosno treba češće praviti pokušaje. Ali to ne znači da treba da se vladamo ovako: ovi su izmislili računare, odnosno razne pojmove u vezi sa njima, oni drugi su izmislili drugi deo nauke sa odgovarajućim stranim (najčešće engleskim) nazivima i mi sada treba sve te nazive da ubacimo u naš jezik, smatrajući da smo tako ispravno postupili.

Hoću i ovo da kažem. Mi znamo kako se razvijao naš jezik. U srednjem veku i ranije naš narod se bavio raznim drugim poslovima, odnosno nije mogao da se bavi matematikom i sličnim stvarima. Mnogi od naših predaka bili su čobani. Iz takvih razloga u vezi sa delatnostima kojima su se bavili naši preci često imamo čak i obilje odnosnih reči. Da li smo sve te reči, ili bar veći deo od njih, uzimali od naših suseda? Svakako nismo. Naši čobani nisu imali slobodnije rečeno, knjižice sa raznim rečima pomoću kojih su naučili reči kao ovca, stado i sl. U skladu sa tim hoću da kažem da je sada trenutak kada imamo priliku i mogućnosti, pogodnosti da se i naukama bavimo, pa to povlači da treba da se trudimo da nadjemo način da u okviru našeg jezika, pomoću naših jezičkih mogućnosti, tu nauku izrazimo. Ali to nikako ne znači da sam ja u potpunosti protiv usvajanja raznih stranih reči. Pored toga smatram da je neophodno poznavanje raznih stranih naziva, posebno onih koji vuku koren iz grčkog i latinskog, što je u svojoj diskusiji kolega Pešikan lepo podvukao. Uzmimo primer strane reči funkcija. Da li tu reč treba izbacivati? Ja tako ne mislim. Kao što znamo pored reči funkcija koristimo i našu reč preslikavanje. Da li tu našu reč treba odbaciti? Ne. Razlog je što se ta reč, kao i razne njoj slične: slikati, slika, preslikan i dr. često i prirodno pojavljuju. Recimo, lepo se kaže da se datom funkcijom svakom članu jednog skupa dodeljuje, pridružuje po jedan član drugog skupa. Taj drugi član je njegova slika, nastao je "slikanjem", odnosno, kaže se, preslikavanjem. Da li bismo tu mogli reći "funkciranjem"? To je skoro besmislica. Ustvari, i uopšte pri unošenju strane reči u naš jezik otvara se pitanje gradjenja odnosnih prideva, glagola i sl., a to često ide dosta teško. I iz takvih razloga je podesnije pravljenje kovanica u našem jeziku.

Na kraju pomenuo bih da, u skladu sa naporima da budemo što kulturnija i naprednija nacija, mi moramo da se učimo da i okviru svog jezika pronalazimo razne nove mogućnosti. Konačno, što god je jezik razvijeniji na njemu se lakše misli.

Dr V. ĆIRIĆ: Činjenica je da mi koji radimo i pišemo u ovoj oblasti imamo dosta velikih terminoloških teškoća. I zato je krajnje vreme da se dâ inicijativa za potpunije rešavanje ovih problema. Složio bih se sa dr Stojakovićem da se dogovorimo danas o načinu rada neočekujući da ćemo za jedan sat rešiti mnogo. Zanimljivo je na ovom sastanku čuti mišljenja lingvista koji bi nam pomogli da se formulišu odredjeni principi, koji bi trebalo da služe kod izrade terminologije u računarstvu na našem jeziku.

Bio bi tu potreban jedan kombinovani tim informatičara i lingvista na nivou Beogradskog univerziteta koji bi definisao jedan projekat, finansiran od Republičke zajednice za obrazovanje, a rezultat tog projekta bio bi rečnik koji bi bio zvaničan, bar na ovoj našoj teritoriji. Takav jedan rečnik bi uneo puno reda jer ovaj problem nije više problem uskog kruga stručnjaka nego dobija i širi značaj.

Dr N. PAREZANOVIĆ: Imali smo već nekoliko korisnih mišljenja i sugestija. Prvo, zaista, čini mi se da se svi slažemo sa tim da se treba truditi u stvaranju naše terminologije. Mislim da ovo što je Prešić rekao da, ako čovek čini stalno napor za nalaženje dobrih termina, onda posao počinje i da odmiče. Medjutim, ako se to kampanjski radi, puno puta se uzme termin koji prvi dodje pod ruku i onda od svega toga ne bude ništa. Veliki je problem i to što ti termini vrlo dinamično nastaju a može se reći i nestaju. Pitanje je da li neke treba uopšte uzeti jer dok čovek smisli kako da ih uzme, taj će termin da nestane. Recimo, u puno engleskih knjiga o računarstvu govorilo se nekad o B-boksu. Verovatno da malo ko zna šta je B-boks; a to je jedna ćelija u registru kojom se definiše modifikacija i označava slo-

vom B. Kada je B-boks nestao iz engleske literature, ostalo je slovo B za oznaku vrste modifikacije. Taj je termin prosto prošao kroz neke knjige, kada je nastao ovaj pojam modifikacije, i zatim nestao.

Medjutim, postoji još jedan problem kada se radi o pisanju stručnih knjiga. Hteo bih da pomenemo i taj problem. Jedan je stvar pitanje termina, o čemu je već dosta rečeno a druga se javlja kada se radi o crtežima i slovnim oznakama, o promenljivim itd. i tu već postoje izvesne uobičajene stvari. Često se dešava da nemamo literaturu u jednoj oblasti i ako izlažemo tu materiju na školskom nivou i upotrebima oznake koje bi bile celishodne sa gledišta našeg jezika, onda smo doveli čitaoca u situaciju da kada uzme stručnu knjigu, na stranom jeziku, mora sve to preurediti i prihvatiti nove oznake da bi uopšte mogao da čita. Primer toga je ulaz-izlaz. U engleskoj literaturi stoji I/O - input-output ali to je toliko ušlo u englesku terminologiju da prosto štoji I/O- uredjaj. Ako prihvatimo oznaku na našem jeziku to bi bilo U/I. Slično je sa oznakom bistabilnog elementa čiji su ulazi R i S. Mogli bismo da kažemo P i B (postavlja i briše) ali se ipak u literaturi zadržava R i S. Znači i pitanje takvih oznaka i skraćenica nije beznačajno: da li od njih treba odstupiti, a meni se čini da ako se uvede termin, da se mora menjati i ta oznaka što onda zaista otežava korišćenje strane literature. Postoji i mogućnost da slovne oznake ostanu kao i u stranim knjigama ali se onda teže pamte.

Dr M. PEŠIKAN: O problemu simbola i skraćenica: ako je jedan simbol internacionalno prihvaćen, ako je razumljiv na internacionalnim relacijama, onda on ima odredjenu vrednost i za nas, i ne treba zazirati od njega, bilo na kom jeziku da se zasniva; ranije je ta osnova bio latinski, sad-recimo u političkom životu dolazi engleski. Nije nikakva smetnja što imamo OUN i UNO, SAD i USA, pa tako i u vašem stručnom izrazu možemo tolerisati takve simbole, ako su internacionalno prihvaćeni; ako nisu, nego ako sami improvizujemo neku uslovnu skraćenicu, onda je bolje da se zasniva na našem jeziku.

Kolega Prešić je neke stvari veoma dobro rekao, i tačno je - da i ja ponovim njegovu slobodnu formulaciju - da termini nastaju "po naredbi".

Povodom njegovog izlaganja rekao bih da možda naš strah od pravljenja novih termina dolazi u velikoj meri od toga što ne uspevamo da nadjemo idealan termin, koji će tačno preneti suštinu pojma. Nikakav jezik ne pruža takve mogućnosti, ali to ne sme ukočiti stvaranje termina, jer termin nije zamena za enciklopedijske podatke o pojmu, on nije elaborat. Termin treba da bude samo dovoljno jasan simbol - da nas podseti na koji se pojam misli ako taj pojam znamo, a ne da nam predstavi pojam, da iznese sve bitne odlike. Termin trougao govori nam samo o uglovima a ne recimo o stranama, stolar nije čovek koji pravi samo stolove i stolice - ali to nije smetnja. Tako treba da gledamo i na termin računar, dobro je što je prodro taj termin iako je isprva bilo samo kompjuter. Ništa ne smeta i ako samo računanje bude tako reći zaboravljena radnja kod njega, ne treba da zaključimo da nećemo računar više tako zvati zato što više nije namenjen za tu radnju.

Govorim o tome zato što je upravo u matematičarskim krugovima došlo do nekih neopravdanih intervencija i prekrajanja zbog želje da termin bude što pogodjeniji. Imali smo termin ravnostran trougao, na koji smo bili lepo navikli, reč je bila kratka, dobro uobličena, i ništa nije smetalo da ostane i dalje. Neko je medjutim razmišljao: "ravno" to nije "jednako", bolje je dakle jednakostranični. Takvim formalnim sudjenjem i strogim parcelisanjem značenja osporili bismo i termine ravnodnevica, ravnopravnost, ravnoteža i sl., kao da se u složenici ne može tolerisati i neko pobočno značenje formanata (a za najosnovnije pojmove termin uopšte i ne mora biti jasne strukture). U želji da se unapredi izraz napravljeno je nekoliko štetnih stvari: stvara se raskol u jeziku raznih generacija, otežava se služenje starijom literaturom, uvodi se šestosložna reč umesto trosložne, a ni to nije beznačajna stavar, jer naše reči postaju sve glomaznije - a sve u strahu da termin neće biti idealnog sklopa.

Imam utisak da smo se i mi, u lingvističkoj terminologiji, uplašili stvaranja novih termina, i imamo ponegde osetne praznine. Da su se ljudi uvek toga plašili, mi ne bismo imali termine imenice, zamenice, suglasnice, samoglasnici, rečenica i niz drugih - a svaka struka ima takvih - koje ćemo uzaludno tražiti u narodnim govorima, jer u njima nisu ni nastali, nego je neko ko se bavio stručnom materijom smelo napravio novu reč. Ali ko je danas u situaciji da napravi novi termin a da to ne bude uzaludni pokušaj, neka duhovna gimnastika bez značaja i efekta? To su profesor za katedrom i pisac teksta koji će se čitati, jer je dovoljno da nastavnik prenese taj termin u naviku jedne grupe studenata ili da ga primeni pisac udžbenika, priručnika ili uopšte takvog teksta koji će imati dosta čitalaca - i takav termin već ima dobre izglede da udje u život. Naprotiv, ako u nekom kabinetu bude sedela npr. komisija matematičara i jezičara i smišljala termine, čak i ako ih zvanično prosledi ustanovama - ti termini neće živeti.

Bio bi idealan put ono što je rečeno da predstavnici raznih struka rade zajedno - ali uvek uz asistenciju lingvista, da ih prosto slušaju i da im ponegde pripomognu savetom. Jedino takvo telo bi moglo sa potpunom kompetencijom i sa najmanje rizika (ipak - nikad bez rizika) praviti nove termine. Medjutim, moram tu izraziti skepsu - struka je veoma mnogo, a lingvista malo. Treba se čuvati toga da ne trošimo energiju pravećі efemerne termine, kojih ima u nekim granama dosta, ali mislim da to nije odlika ove vaše struke; možda će i kod vas pojam promeniti donekle smisao ali će sam termin ostati, i onda on nije efemeran.

U svakom slučaju, hteo bih na kraju da pozdravim činjenicu što vi, iako vam nije struka jezik, osećate te jezičke probleme kao važne; s pravom, jer oni doista jesu važni i imaju dalekosežan značaj za našu kulturu.

Dr D.JOVIĆ: Ovoga puta biću vrlo kratak. Da se još jednom vratimo na ovaj problem. Kao što vidite ne mogu nikako izbeći reč p r o b l e m, mada reč nikako nije naša, niti mogu izbeći reč p r o g r a m, a ona takodje nije naša.

Hteo bih reći nešto o jezičkom stanju u nas. Očigledno je da u nekim oblastima komunikacije, jezik je veoma malo naš. Termini veoma često, a ponekad i struktura iskaza nije naša. U jeziku književne kritike, publicistike, ima slučajeva kad je oko 60% termina stranog porekla, često posve nepotrebno. Koliko pravljenje vlastite terminologije, ne vodeći računa o stanju date nauke u svetskim okvirima, može otežati shvatanje dostignuća i ideja, pokazaću na primeru lingvistike. Početkom ovoga stoleća pojavile su se dve strukturalne škole: u Evropi jedna, u Americi druga. Obe su strukturalnog tipa i u mnogim stvarima slične. Obe su postavile osnove razvoja moderne lingvistike u svetu.

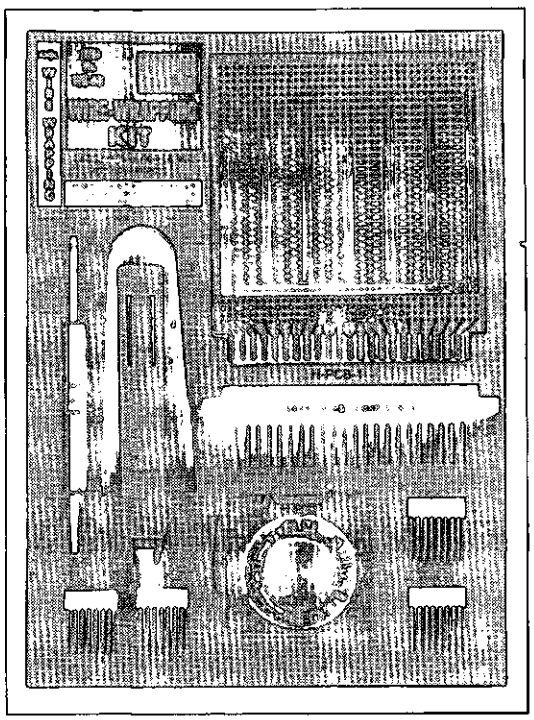
Terminologija je, medjutim, tako različita da je potrebno učiniti poseban napor i dovesti ta dva učenja u vezu, da bi se razumelo u čemu je stvar. To znači, bez obzira hoćemo li mi to ili ne, u pojedinim oblastima neophodno je nešto unificirati, internacionalizovati. Od toga niukom slučaju ne treba bežati. Uzmimo jedan primer: kad bi svaki jezik koji možemo čitati imao vlastite nazive za padeže, imali bismo mnogo problema kako da termine svladamo. Mi u novoj lingvistici usvajamo termin fonema, jer to nije isto što i glas, usvajamo termin morfema jer to nije isto što i nastavak, - to čak uvodimo i u školsku praksu na srednjem stupnju. Mi moramo u tom smislu praviti kompromise da bi olakšali čoveku kad uzme knjigu na nekom stranom jeziku da se lakše snadje, da kraćim putem dodje do suštine. U takvom slučaju postoji nekakva korespondencija izmedju reči koje upotrebljavamo i onih što ih nalazimo u drugim jezicima, bez obzira na to što ih razdvajaju različite adaptacije.

Mislim da nikada nećemo uspeti prevesti sve na vlastiti jezik. To je sigurno nemoguće. Mi nešto dajemo svetu, nešto od sveta uzimamo. Tamo gde je nužno, ponešto može biti internacionalno. Mi nemamo razloga zatvarati se ni u tom pogledu.

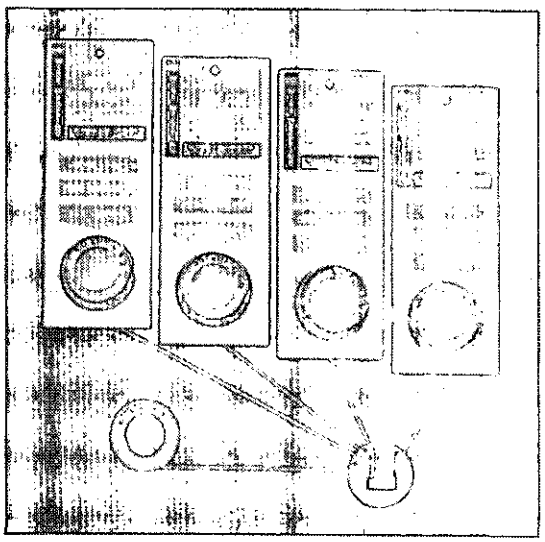
4. ZAHVALNOST

Autor se toplo zahvaljuje Dušanu Pajčiću i Milici Isidorović na pomoći u pripremi ovog teksta za štampu.

WK-4B WIRE-WRAPPING KIT



WIRE-WRAPPING WIRE



WK-4B OŽIČEVALNA SESTAVLJENKA

Sestavljenka vsebuje orodje in dele, ki so potrebni za izdelavo prototipne ali amaterske ploščice. Deli, ki jih ima sestavljenka, so: univerzalna plošča s tiskanim vezjem, standardni 2 x 22-polni konektor z izvodi za ožičevanje, dve 14-in dve 16- kontaktni podnožji za integrirana vezja z izvodi za ožičevanje, orodje za vstavljanje in izvlačenje integriranih vezij, tulec s 15 m žice in posebno orodje WSU-30, ki je kombinacija orodja za ožičevanje in odvijanje žice na trnih s premerom 0,63 mm; v ročaju je vgrajeno rezilo za snemanje izolacije.

ŽICA ZA OVIJANJE

Žica ima najvišjo industrijsko kvaliteto z oznako AWG 30 (0,25 mm), ki je navita v 15 m zvitkih. Žica je primerna za manjše proizvodne serije, razvojna dela, izdelavo prototipov ali za amaterske projekte. Žica je prevlečena s plastjo srebra in je izolirana s posebno plastjo, ki prenese velike mehanske in električne obremenitve.

Na razpolago so štiri barve izolacije: bela, modra, rumena in rdeča. Žica je navita na 40 mm kolutih, ki omogočajo boljše rokovanje in skladiščenje.

Ko pišete proizvajalcu, omenite časopis INFORMATICA.

vzgoja in izobraževanje

ODSJEK ZA INFORMATIKU ELEKTROTEHNIČKOG FAKULTETA U SARAJEVU

Odsjek za informatiku je osnovan 1974 godine kada je počela i nastava na tom odsjeku. Nastava se organizuje kroz poseban nastavni plan u zadnje dvije godine studija, poslije prve dvije, koje su zajedničke sa ostalim studijem na Elektrotehničkom fakultetu.

Diplomirani inženjeri koji se obrazuju iz informatike posjeduju znanja koja im omogućavaju da rade kao:

- stručnjaci za računarski hardver i računarski softver (projektovanje, razvoj, programiranje, održavanje)
- stručnjaci za razvoj i održavanje obrada podataka i informacionih sistema tj. njihovu analizu, projektovanje, implementaciju, održavanje i modifikaciju.

Do sada je diplomiralo oko 30 studenata.

U nastavnom planu studenti slušaju slijedeće predmete:

a) Oblast računarske tehnike i nauke

- Elektronika računarskih sklopova
- Digitalni računari I i II (hardver)
- Osnovno (asemblersko) programiranje
- Programska organizacija računara I i II (računarski softver)
- Programski jezici I i II
- Projektovanje računarskih sistema
- Mini računari
- Teorija algoritama i automata
- Informacione strukture

b) Teorija sistema i teorija upravljanja

- Teorija signala i sistema I i II
- Teorija optimalnih rješenja I i II
- Operaciona istraživanja

c) Opšta ekonomsko obrazovanje

- Ekonomika i organizacija preduzeća
- Upravljanje u preduzeću i javnoj upravi

d) Informacioni sistemi

- Informacione strukture
- Analiza i sinteza informacionih sistema I i II
- Organizacija i obrada podataka
- Programiranje i realizacija obrada

Odsjek za informatiku organizuje i nastavu postdiplomskog magistarskog studija. Na tom studiju pored obaveznih predmeta:

- Elementi moderne matematike
- Metode programiranja
- Računarski i programski sistemi

kandidat bira još tri predmeta prema stručnoj opredjeljenosti kandidata. Do sada se održavala nastava iz slijedećih predmeta:

- Organizacija računara
- Informacione strukture
- Baza podataka
- Uvod u informacione sisteme
- Operativni sistemi
- Projektovanje sistema u realnom vremenu
- Mreža računara
- Pouzdanost računarskih sistema

U okviru Odsjeka za informatiku postoje dvije katedre i to:

- Katedra za računarske mašine
- Katedra za informacione sisteme

Na njima rade 6 nastavnika i 12 asistenata u stalnom radnom odnosu, a angažovano je pored toga više nastavnika i istaknutih stručnjaka iz prakse.

Na Odseku se radi na više istraživačkih projekata kao što su:

- Metodološka istraživanja razvoja informacionih sistema,
- Mreža računara i distributivna obrada,
- Baze podataka,
- Matematski aspekti računarskih nauka,
- Multiprocesorski sistemi

Isto tako saradnici rade na konkretnim projektima obrada podataka za potrebe organizacija udruženog rada i time se povezuju sa praksom.

Šef Odsjeka za informatiku,
dr. Ahmed Mandžić, rednovni prof.

Diplomirani inženjeri informatike
(Odsjek za informatiku Elektrotehničkog fakulteta u Sarajevu)

1. Maksumić Hasan
Diplomski rad: "Principi i sredstva prenosa digitalnih podataka".
Mentor: doc.dr. Zoran Salčić

2. Popić Dušanka
Diplomski rad: "Personalni podsistem u informacionom sistemu konkretne radne organizacije".
Mentor: prof. Mesud Baručija

3. Šuman Hajrudin
Diplomski rad: "Vezivanje rada paketa za analognu simulaciju računara ICL1902A sa graf platerom u cilju dobivanja faznih portreta".
Mentor: prof.dr. Branislava Peruničić

4. Smajlagić Vesna
Diplomski rad: "Informacioni sistem za upravljanje poslovanjem SOUR "Natron" Maglaj".
Mentor: doc. Dragan Kovač

5. Lagumdžija Zlatko
Diplomski rad: "Uvodjenje programskih jezika PL/1 i ALGOL 60".
Mentor: prof. Mesud Baručija

6. Arapčić Vildana
Diplomski rad: "Uvodjenje informacionih sistema".
Mentor: prof. Mesud Baručija

7. Taljanović Kemal
Diplomski rad: "Jedan pristup mjerenju kvaliteta sistema za obradu podataka".
Mentor: prof.dr. Branislava Peruničić

8. Popović Nedjo
Diplomski rad: "Sistem za praćenje troškova proizvodnje".
Mentor: doc. Dragan Kovač

9. Padalo Fadil
Diplomski rad: "Informacioni sistem u oblasti obračuna ličnog dohotka Radne organizacije "Vladimir-Perić-Valter".
Mentor: prof. Mesud Baručija

10. Hamidović Mirsada
Diplomski rad: "Inteligentni terminal za prikupljanje podataka INFOREX 1302/16".
Mentor: doc. Faruk Hadžiomerović

11. Radić Gordana
Diplomski rad: "Investiciono održavanje putničkog vozila kolskog parka ŽTP-a Sarajevo".
Mentor: doc.dr. Zoran Salčić

12. Djuknić Milan
Diplomski rad: "Programski paket za kreiranje i korištenje podataka sa više ključeva".
Mentor: prof. Mesud Baručija

13. Kovačević Goran
Diplomski rad: "Informacioni sistem za upravljanje željezničkim saobraćajem".
Mentor: doc.dr. Zoran Salčić

14. Hajduković Miroslav
Diplomski rad: "Sistem kros-asemblera za mikroračunare".
Mentor: doc.dr. Zoran Salčić

15. Djokić Gordana
Diplomski rad: "Osnovne postavke matričnih igara i rješavanje antagonističkih matričnih igara metodom linearnog programiranja".
Mentor: prof.dr. Branislava Peruničić

16. Čavalić Mirsad
Diplomski rad: "Relacioni model baze podataka".
Mentor: prof. Mesud Baručija

17. Kasumagić Nedžad
Diplomski rad: "Izrada programa za formiranje i ažuriranje datoteke".
Mentor: doc. Dragan Kovač

18. Šapina Josip
Diplomski rad: "Operacioni sistemi malih računara".
Mentor: prof. Mesud Baručija

19. Jakupović Safet
Diplomski rad: "Simulator mikroračunara - Motorola 6800 na mini računaru PDP-11".
Mentor: prof. Mesud Baručija

20. Beus Ljerka
Diplomski rad: "Informacioni sistem zdravstvenog osiguranja".
Mentor: prof.dr. Branislava Peruničić

21. Kadić Aida
Diplomski rad: "Deskriptivno modeliranje organizacije kao metoda analize objektnog sistema".
Mentor: prof.dr. Branislava Peruničić

22. Milanović Vojislav
Diplomski rad: "Neki algoritmi pretraživanja podataka".
Mentor: prof. Mesud Baručija

23. Ridjanović Dženan
Diplomski rad: "Realizacija hijerarhijskog sistema za manipulisanje bazom podataka zasnovanog na B-drveću: Algoritmi za pretraživanje i ažuriranje".
Mentor: doc.dr. Suad Alagić

24. Atijas Ranko
Diplomski rad: "Jedan način dokumentovanja izvještaja informacionog sistema".
Mentor: prof.dr. Branislava Peruničić

25. Maksimović Vojo
Diplomski rad: "Realizacija hijerarhijskog sistema za manipulisanje bazom podataka zasnovanog na B-drveću: Jezički pretrocesor".
Mentor: dr.dr. Suad Alagić

26. Popović Milena
Diplomski rad: "Paket programa linearnog programiranja MARK2, na računaru ICL 1902A".
Mentor: prof.dr. Branislava Peruničić.

27. Isaković Adnan
Diplomski rad: "Relacioni upitni jezici i dokumentacija relacionih upita".
Mentor: doc.dr. Suad Alagić

28. Jović Radovan
Diplomski rad: "Realizacija hijerarhijskog sistema za manipulisanje bazom podataka zasnovanog na B-drveću: Algoritmi za umetanje i izbacivanje".
Mentor: doc.dr. Suad Alagić

PRIMER POČITNIŠKE PRAKSE V RAČUNALNIŠKEM CENTRU

Iz razgovorov s srednješolci in študenti lahko ugotovimo, da le-ti prepogosto svojo obvezno počitniško prakso "preživijo" v delovni organizaciji pasivno opazujoč dogajanja okoli sebe ali v branju literature in prospektov. Ni dvoma, da v takšnih primerih nezaupanja v mlade praktikante (bodoče strokovnjake) nimajo prave koristi niti slednji niti delovna organizacija, v kateri se začasno nahajajo.

Poleti 1978 smo v ERC Rašica vzeli na prakso absolventa prvega letnika Elektrotehniške šole v Ljubljani. Zanimivo je mogoče omeniti, da se je praktikant odločil za prakso v omenjenem računalniškem centru, ker se je že na osnovni šoli udeleževal računalniškega krožka, ki so ga vodili sodelavci omenjenega centra.

Osrednje vodilo pri opredelitvi praktikantovih del je bila želja, da bi bilo to delo zanimivo in obojestransko koristno. Izhajali smo tudi iz dejstva, da je v našem računalniškem centru za dela in naloge operaterja predpisana srednja strokovna izobrazba in da je ravno elektrotehniška smer še posebej zaželena. Praktikant nam je že prvi dan potrdil, da se strinja s tem, da bi mu praksa bila predvsem usmerjena v operaterska dela in naloge.

Znano je, da je počitniška praksa za vse učence prvih, drugih in tretjih letnikov obvezna in da traja tri tedne oz. 15 delovnih dni. Za vsak dan je bil izdelan dokaj natančen program del. Ta je razviden iz priložene vsebine prakse, povzete s prve strani dnevnika, ki jo je praktikant sam sprogramiral, izluknjal in obdelal na računalniku. Značilno je, da je praktikant že šesti dan prakse začel praktično delo na konzoli sistema, delo v katerem se je postopoma izpopolnjeval, in je štiri dni pred koncem prakse nekatere obdelave izvajal samostojno.

Praktikanta smo sprejeli v kolektiv kot enakopravnega člana, mu zaupali in ga spodbujali. Omogočili smo mu celo udeležbo na seji delavskega sveta in smo z njim sodelovali v razpravi o gradivu pred sejo in po njej. Zahtevali smo, da piše dnevnik počitniške prakse redno vsak dan, vsako naslednje jutro pa sta mu dnevnik pregledala mentor in operater in se zmenila za podrobnosti tekočega dne.

Nismo zanemarili niti hardwarsko stran in je praktikant pomagal vzdrževalcu sistema pri meritvah in pri popravljanju okvare na računalniku.

V kolikor pride isti praktikant tudi naslednje poletje v naš center, imamo namen nadaljevati tam, kjer smo se ustavili na koncu pretekle prakse.

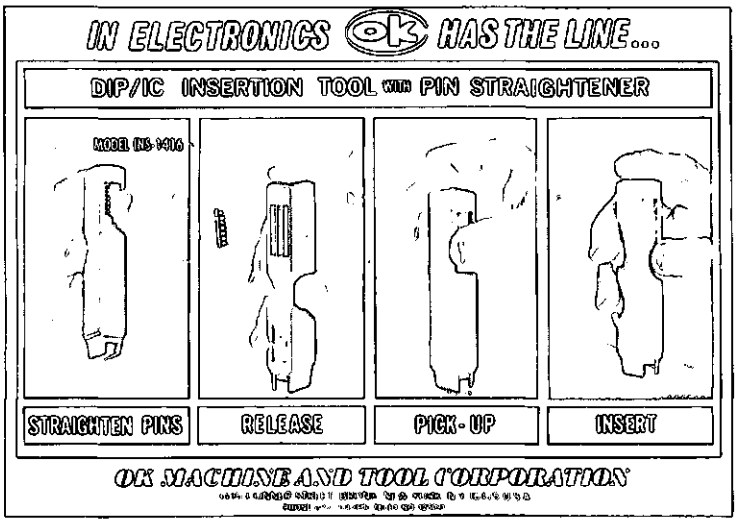
Vsebina počitniške prakse po dnevih:

1. Predstavitev tovarne Rašica in ERC.
2. Seznanitev z luknjačem in delo na njem.
3. Dupliciranje in popravljanje kartic.
4. Video-terminal UNISCOPE 100 in delo na njem.
5. Sistemska soba in seznanitev z računalnikom UNIVAC 9480.
6. Centralni spomin in konzola. Delo na konzoli.
7. Hitri tiskalnik in čitalec kartic.
8. Enote magnetnih trakov. Arhiv ERC.
9. Magnetni diski. Udeležba na seji DS.
10. Vzdrževanje sistema.
11. Popravljanje okvare na računalniku.
12. Dela in naloge operaterja - praktično.
13. Praktično delo na računalniku.
14. Izdelava in izvajanje manjšega programa.
15. Pomoč operaterju. Ogled tovarne.

Na koncu bi povzeli nekatere izkušnje iz večletnega sodelovanja z domačimi in inozemskimi praktikanti:
Imamo vtis, da ob dobro organizirani praksi praktikantu ni težko oceniti mesto in vlogo računalništva v delovni organizaciji. Praktikantu je treba nuditi dovolj priložnosti, da spozna delovanje računalnika in njegove zmogljivosti. Od prvega dne ga je treba navajati na sistematičen pristop k delu, ki ga računalnik sam po sebi zahteva, in je širšega izobraževalnega in življenskega pomena. Imeti mora vedno konkretna, koristna dela in naloge, ki motivirajo in pripomorejo k aktiviranju znanja. Sodeluje naj pri izbiri del in naj spozna neposredno korist teh del za delovno organizacijo. Praktikanta je treba prevzeti v kolektiv kot enakopravnega člana in ga vključiti v reševanje tudi širše problematike ERC in delovne organizacije.

D. Krstič
ERC RAŠICA, Ljubljana

OK MACHINE AND TOOL CORPORATION / 3455 CONNER STREET, BRONX, NEW YORK 10475, U.S.A.
Phone: (212) 994-6600 • Telex: 12-5091 • Telex: 23 2395

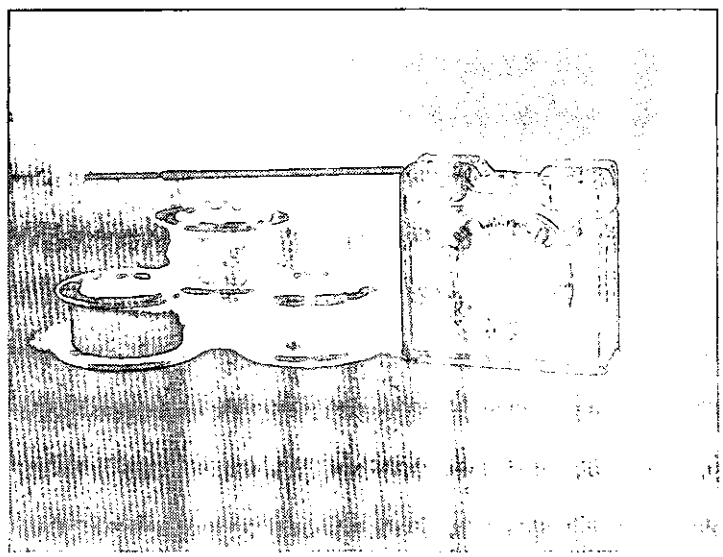


INS 1416 je orodje za vstavljanje 16- ali 14-kontaktnih integriranih vezij v podnožja ali izvrtane luknje tiskanega vezja. Posebnost je zoženi profil, ki omogoča vstavljanje vezij, ki so na plošči tesno skupaj. V držalo sta vrezani vodili za ravnanje deformiranih kontaktov integriranega vezja. Vezje potisnemo v vodilo, pri tem se poravnajo deformirani kontakti, izvlečemo pa ga s pritiskom držala navzdol.

TULEC Z ZAMENLJIVIMI KOLUTI ŽICE ZA OVIJANJE

Prednost tulca je v tem, da ne potrebujemo dodatnega orodja za rezanje žice in snemanje izolacije. Iz tulca se izvleče želena dolžina žice, katero se vstavi v vodilo na vrhu tulca. S pritiskom na vgrajeni gumb, se žico odreže. Potem se žico potegne preko posebnih vgrajenih vilic, ki snemajo izolacijo; isti postopek se ponovi tudi z drugim koncem žice.

Tulec vsebuje kolut s 15 m dolgo žico tipa 30 AWG (0,25 mm), s posebno industrijsko izolacijo in posrebreno bakreno žico. Na razpolago so koluti z belo, modro in rdečo izolacijo.



novice in zanimivosti

ZGRADBA PROCESORJA ZILOG Z 8000

Ta pregled opisuje zgradbo 16-bitnega mikro procesorja Z8000, ko se primerjajo lastnosti Z 8000 s PDP 11/45.

Šestnajstbitni procesorji Z8000 lahko v svoji, t.i. segmentni obliki, nas neposredno 8 milionov zlogov v pomnilniku, ima 16 univerzalnih registrov, obdela sedem različnih podatkovnih tipov ter ima osem načinov za naslavljanje. Procesor doseže visoko stopnjo pretoka podatkov z relativno nizko taktno frekvenco in lahko uporablja cenene dinamične pomnilnike. Njegova primerjava z DECovim PDP 11/45 se izteče v njegovo korist.

Zilogov 16-bitni mikro procesor Z8000 je moč nabaviti v dveh izvedbah: v 48-nožični segmentirani in v 40-nožični nesegmentirani izvedbi. Segmentirana izvedba se uporablja z zunanjo pomnilniško krmilno napravo, ki omogoča spremenljive obsege segmentov ter lahko naslovi do 8 milionov pomnilniških lokacij.

Centralna procesna enota (CPU) vključuje 16 registrov, sprejema sedem podatkovnih tipov, ima veliko ukazno zalogo (več kot 110 osnovnih ukaznih tipov) ter osem načinov naslavljanja. Regularnost registerske organizacije, podatkovni tipi, ukazi in načini naslavljanja pocnostavljajo programiranje ter znižujejo dolžino programov.

Z8000 doseže visok podatkovni pretok z relativno nizko taktno frekvenco in lahko spričo tega uporablja pomnilnike z daljšim časom dostopa. Procesor vsebuje tudi mehanizem za osveževanje dinamičnih pomnilnikov z nastavljivo frekvenco osveževanja.

Kompilatorji ter s kompilatorjem in operacijskim sistemom proizvedeni kodi se izvajajo zelo učinkovito. Procesor Z8000 podpira prevajalnike s svojo polno ukazno zalogo, bogatim naslovnim prostorom, premeščanjem, večkratnimi skladi ter s specifičnimi ukazi (PUSH, POP, INCREMENT, TEST).

Operacijski sistemi so podprti s sistemskimi in normalnimi načini, s sistemskimi in normalnimi skladi, s posebnimi ukazi (SYSTEM CALL, LOAD PROGRAM STATUS, priviligirani ukazi) ter s prekinitveno strukturo in zgradbo pasti. Uporabljajo se trije tipi prekinitev (brezmaskirni, brezvektorski in vektorski) ter pet tipov pasti za sistemske pozive, nelegalne ukaze, priviligirane I/O ukaze, za druge priviligirane ukaze ter za segmentiranje.

Multiprocesorski sistemi (več Z8000) so programsko podprti z izključitvenimi in sinhronizacijskimi ukazi, materialno pa jih podpirata vhod "Micro In" ter izhod "Micro Out".

1. Centralna procesna enota. CPU ima 16 splošnih registrov s po 16 biti ter posebne sistemske registre. Vseh 16 splošnih registrov ima lastnosti akumulatorja ter se lahko uporabljajo tudi kot indeksni registri. Prvih 8 splošnih registrov se lahko uporabi kot 16 registrov s po 8 biti.

Procesor Z8000 podpira sedem glavnih podatkovnih tipov: bit, BCD številka, zlog, beseda (16 bitov), dolga beseda (32 bitov), zlogovni niz in besedni niz. Definirani so tudi podatkovni elementi, kot so: pomnilniški naslov, I/O naslov, vstop segmentne tabele in programska statusna beseda.

Osem načinov naslavljanja ima tele glavne načine: R (register), IR (indirect register), DA (direct address), X (indexed) in IM (immediate). Za nekatere ukaze pa veljajo še drugi načini naslavljanja: BA (base address), BX (base indexed), RA (relative address), samoinkrementirni in samodekrementirni. Procesor ima 110 različnih ukaznih tipov v primerjavi s 60 pri PDP 11/45. Z nekaj izjemami je moč procesirati zloge, besede in dolge besede z vsemi ukazi. Večina ukazov lahko uporablja vseh pet glavnih načinov naslavljanja.

Preko 410 kombinacij ukaznih tipov, podatkovnih elementov in načinov naslavljanja je možnih. Z8000 ima materialno udejanjene tudi ukaze za množenje in deljenje s predznakom za 16 in 32-bitne vrednosti.

2. Naslovni prostori. Z8000 lahko neposredno naslovi 8 mega zlogov pomnilnika na naslovni "prostor". Šest ločenih naslovnih prostorov obstaja za Z8000, in sicer: kodni (ukazni), podatkovni, skladni za sistemski način ter skladni za normalni način.

Upravljanje pomnilnika (segmenti spremenljivega obsega, premeščanje in zaščita) se lahko uresniči s pomočjo posebne naprave za pomnilniško upravljanje.

Edina senčna stran dolgih naslovov zaradi obsežnega naslovnega prostora je večja dolžina ukaza ter potreba po registerskih dvojicah za nekatere načine naslavljanja. Ta problem je v Z8000 zmanjšan s segmentiranimi naslovnimi oblikami, z uporabo kratkih naslovov v vrsti situacij ter z možnostjo razpolaganja velikega števila splošnih registrov.

3. Upravljanje pomnilnika. Programi procesorja Z8000 imajo dostop do celotnega naslovnega prostora. Ker je 8 mega zlogov neposredno naslovljivih, ima vsak ukaz polni naslovni način, kjer je 23 bitov predvidenih za naslov. Toda Z8000 pozna tudi način, ko je moč enak naslov izraziti s 16 biti tedaj, ko je gornjih 8 bitov enakih 0 (kratek način naslavljanja).

Druga oblika, ki je na voljo programerju, je razlikovanje med sistemskim kodom in normalnim kodom ter razlikovanje med ukaznim prostorom, podatkovnim prostorom in prostorom za sklade. Kadar se uporablja ta oblika, lahko Z8000 naslovni 48 mega zlogov pomnilniškega prostora.

S podporo sitemskih programov lahko naprava za pomnilniško upravljanje upravlja ta veliki pomnilni prostor namesto uporabnika. Segmentiranje je mehanizem namenjen naslavljanju velikega obsega pomnilnika, naslovljivega z Z8000. Segmentirani naslov je sestavljen iz dveh delov: segmentne številke in vrednosti potomca. Z8000 lahko označi 128 segmentov, t.j. pomnilniških območji, ki so spremenljiva v intervalu 256 zlogov do 64 000 zlogov v prirastkih po 256 zlogov.

Za uporabe, ki ne zahtevajo velikega naslovnega prostora, je na voljo 40-nožična, nesegmentirana različica procesorja Z8000 (segmentirana ima 48 nožic). Edina razlika med segmentirano in nesegmentirano različico je v številu zlogov na naslov in v številu registrov, ki se uporabljajo za polni naslov. Kod, ki je napisan za 40-nožični Z8000 (nesegmentirani) se lahko izvaja v enem segmentu 48-nožičnega Z8000 (segmentiranega), če se ga uporabi v nesegmentiranem načinu. Tako je dosežena popolna skladnost

med obema različicama.

4. Kodna gostota. Število besed za specifikacijo takih ukazov, ki jih sprejema zbirnik, je bilo minimizirano. Tako imamo samo eno besedo za ukaze, kot so npr. JUMP RELATIVE, CALL RELATIVE, LOAD BYTE REGISTER IMMEDIATE in LOAD WORD REGISTER IMMEDIATE (za majhne takojšnje vrednosti).

Uporablja se tudi mehanizem, ki omogoča redukcijo naslova na eno samo besedo; te mehanizme se avtomatično vključuje z zbirniki in prevajalniki.

5. Učinkovitost prevajalnika. Z8000 je splošno uporaben procesor in zaradi tega razpolaga z jezikovno podporo, ki poenostavi tipično prevajanje in generiranje koda. K temu je treba dodati še regularnost načinov naslavljanja in tipov podatkovnih elementov. Razen tega je moč uporabiti vsak register kot kazalec za sklad z ukazoma PUSH in POP. Segmentiranje in premeščanje sta učinkoviti obliki za implementiranje procedur v visokih programirnih jezikih. Ukaza "INCREMENT BY 1 TO 16" in "DECREMENT BY 1 TO 16" sta učinkovita pri dodeljevanju skladovnih okvirov ter pri anuliranju dodelitve. Za skladovne okvire sta primerna načina naslavljanja BASE ADDRESS in BASE INDEXED.

Testiranje podatkov, logično vrednotenje , inicializacija in primerjanje podatkov je omogočeno z ukazi TEST, TEST CONDITION CODES, LOAD IMMEDIATE INTO MEMORY in COMPARE IMMEDIATE WITH MEMORY. Prevajalniki in zbirniki često manipulirajo z znakovnimi nizi in ukazi TRANSLATE, TRANSLATE AND TEST, BLOCK COMPARE in COMPARE STRING prispevajo k hitrosti izvajanja programov.

6. Operacijski sistemi. Prekinitev in oblike preklapljanja poslov so vključene z namenom, da se izboljša implementacija operacijskega sistema. Pomembno je tudi pomnilniško upravljanje ter oblike prevajalniške podpore.

Prekinitvena struktura ima tri ravni: nemaskirano, nevektorsko in vektorsko. Ko se pojavi prekinitev, se shrani v sklad programsko stanje z označitvijo vzroka tega stanja, ko se preklopi na stanje novega programa tako, da se to stanje naloži iz posebnega pomnilniškega območja. Programsko stanje je sestavljeno iz zastavičnega registra, krmilnih bitov in programskega števnika. Ta informacija se pojavi v obliki vektorja na sistemskem vodilu. Pri vektorski prekinitvi določa vektor tudi skok v naslvono tabelo, ki kaže k prekinitveni procesni rutini.

Organizacija operacijskega sistema je tako izboljšana s sistemskimi in normalnimi načini. V sistemskem načinu se dovoljene vse operacije; v normalnem načinu je prepovedanih nekaj sistemskih ukazov. Ukaz SYSTEM CALL dovoljuje krmiljeno preklopitev načina, implementacija pasti pa zahteva take omejitve.

Pasti se obravnavajo pri shranjevanju (reševanju) programskega stanja podobno kot prekinitve: v obeh primerih se rešena informacija vstavi v sistemski sklad, dočim ostane normalni sklad nedotaknjen. Ukaz LOAD MULTIPLE omogoča učinkovito shranitev registerskih vsebin v pomnilnik ali v sklad. Med izvajanjem programa lahko nastajajo spremembe programskega stanja pod vplivom neposredne programske kontrole z uporabo ukaza LOAD PROGRAM STATUS.

Končno je moč doseči izključitev in zaporednost procesov z ukazoma TEST in SET, ko se doseže sinhronizacija asinhronskih sodelujočih procesov.

7. Multi procesiranje. Mehanizem izključitev/sekvencioniranje (določitev zaporedja) je predviden za uporabo v multiprocesorskih sistemih. Vsak CPU v multiprocesorskem sistemu lahko izključi vse druge asinhrone CPU-je iz kritičnih, deljenih virov z uporabo vhoda MICRO IN in izhoda MICRO OUT v z ukazi REQUEST, RELEASE, TEST MICRO IN, SET MICRO OUT in RESET MICRO OUT.

8. Programska oprema. Procesor Z 8000 je podprt s polnim območjem uporabniške programske opreme ter s pripomočki za razvoj materialne opreme. Prevajalniki za PLZ, BASIC, COBOL in FORTRAN producirajo kod za Z8000.

Z uporabo avtomatičnega prevajalnik je moč pretvoriti kod procesorja Z 80 (8-bitni procesor) v kod procesorja Z8000, saj je ukazna zaloga procesorja Z8000 nadmnožica ukazne zaloge procesorja Z80. Oba mikro procesorja sta registrsko usmerjena; načini naslavljanja za Z80 so podmnožice onih za Z8000; vsi smiselni ukazi za Z80 so bili implementirani tudi za Z8000.

9. Primerjava med Z8000 in PDP. Vsako vrednotenje računalniških zmogljivosti mora upoštevati čase izvajanja tipičnih programov v tipičnih uporabah. Te uporabe so za Z8000 prevajalniki, operacijski sistemi in upravljanje podatkovnih baz. Na teh področjih je Z8000 pet do desetkrat hitrejši kot katerikoli 8-bitni mikro procesor (vključno Z80A) in dva do petkrat hitrejši kot drugi 16-bitni mikro procesorji ter popularni mini računalniki, kot je npr. PDP 11/34. Z8000 doseže te zmogljivosti z NMOS tehnologijo ter s taktno frekvenco 4 MHz, s čemer je omogočena tudi uporaba cenenih dinamičnih pomnilnikov tipa RAM. Razen tega prekrije Z8000 izvajanje trenutnega ukaza s prenosom naslednjega ukaza iz pomnilnika in se tako izogne problemu t.i. globinskega brezpogojnega vnaprejšnjega prenosa ukaza iz pomnilnika.

Procesor Z8000 je hitrejši kot DEC PDP 11/45 in samo neznatno počasnejši od tipa PDP 11/70.

Izvirni način naslavljanja	Z 8000 pri 4 MHz	PDP 11/45 z 8k
registerski	0,75	0,90
indirektno registerski	1,75	1,88
direktno naslavljanje	2,25	2,78
indeksno naslavljanje	2,50	2,78
takojšnje	1,00	1,88

Tabela 1. Časi izvajanja za ukaz LDB R, SOURCE; v mikro sekundah

Tabela 1 kaže primerjavo časovne zmogljivosti procesorja Z8000 s PDP 11/45 za tipičen ukaz naložitve zloga; v tem primeru je prednost za Z8000 očitna. V tabeli 2 je prednost PDP 11/45 le pri množenju. Sicer pa prihaja prednost Z8000 v ospredje v večji moči vrste ukazov, v bolj zapletenih načinih naslavljanja pa tudi takrat, ko je dolžina besede večja.

UKAZ		LD R, DA	ADD R, DA	MULT R, DA
Podatkovni tip		Zilog beseda(16bitov dolga beseda (32 bitov)	Zilog beseda dolga beseda	Zilog beseda dolga beseda
Z8000 pri 4 MHz	Število ukazov	1 1 1	1 1 1	1 1 1
	Število zlogov	4 4 4	4 4 4	8 1 4
	Število ciklov	9 9 12	9 9 15	87 70 350
	čas izvajanja /us/	2,25 2,25 3,00	2,25 2,25 3,75	21,75 17,50 88
PDP 11/45 z 8K	Število ukazov	1 1 2	2 1 3	2 1 17
	Število zlogov	4 4 8	6 4 10	6 4 42
	čas izvajanja /us/	2,78 2,78 5,56	3,68 2,78 6,46	6,61 5,56 33,94

TABELA 2. Število ukazov, Število zlogov in čas izvajanja v odvisnosti od obdelave zloga, besede in dolge besede pri Z8000 in PDP 11/45

Vzorec procesorja Z8000 je že mogoče dobiti pri proizvajalcu (Zilog) in njegovih zastopnikih v Evropi.

APŽ

ZGRADBA PROCESORJA MC6809

Tudi Motorola se je potrudila, da bi držala korak z najnovejšo tehnologijo na področju mikro procesorjev. Pravkar je prišel v prodajo procesor 6809, ki je le vmesna (prehodna) faza med sistemom 6800 s sedmimi integriranimi vezji in prihodnjo, 16-bitno družino MACS (Motorola Advanced Computer System) z integriranimi vezji, ki vsebujejo visoke jezike (VLSI H-MOS). MACS ima 16-bitno ukazno besedo ter 24 bitni naslovni prostor, kar omogoča naslavljanje pomnilnikov s 16 mega zlogov. Podatkovna pretočnost procesorja 6809 se je tako v primerjavi s 6800 znatno povečala.

Procesor 6809 je s svojimi 16-bitnimi registri, z notranjim vodilom, pomnilnimi skladi in 40 možicami primeren tudi za uporabo s prekinitvami. Integrirano vezje izvaja programe s 50% manjšim pomnilnim prostorom (v primerjavi s 6800), ima vključeno taktno vezje s kristalom ter posebno nožico za "ready-state" signal pri počasnih pomnilnikih.

6809 razpolaga z 18 načini naslavljanja in tremi načini prednostnega prekinjanja; 6800 ima le 6 načinov ter eno prekinitveno funkcijo.

Nove lastnosti, ki jih pri 6800 ne najdemo, so vključene v novo shemo naslavljanja: novi indeksni register, skladovni kazalec ter dvojni akumulator; dvojno notranje vodilo; materializirani multipleks; strukturirani visoki jezik ter položajno neodvisno kodiranje.

Funkcije, ki jih 6800 nima, so: operacije nad 16-bitnimi podatki, kot so nalaganje, shranjevanje, premaščanje in izmenjavanje, "push" in "pull", sestevanje in odštevanje pomnilniških podatkov, sestevanje 8 in 16-bitnih registrov, aritmetična primerjava ter naložitveno učinkovito naslavljanje.

Razporeditev signalov na možicah procesorja 6809 je podobna razporeditvi za procesor 6800: majhno število signalov, enaka razporeditev vodil, to so značilne lastnosti.

V čem se 6809 še razlikuje od 6800?
Specifično zanj je, da ne potrebuje absolutnih pomnilniških navedb pri začasnem shranjevanju in za vrednosti v skladu; uporablja bločno strukturirane visoke jezike; operacijski kodi niso združljivi s kodi za 6800, izvirni kodi pa so.

Procesor 6809 vsebuje tele 16-bitne ukaze:

ADD (add memory to D accumulator);
SUBD (subtract memory from D accumulator);
LDD (load accumulator from memory);
STD (store accumulator to memory);
CMPD (compare D accumulator with memory);
LDX,LDY,LDS,LDU (load pointer register from memory);
STX,STY,STS,STU (store pointer register to memory);
CMPX,CMPY,CMPS,CMPU (compare pointer register with memory);
LEAX,LEAY, LEAS,LEAU (load effective address into index register);
SEX (sign extend D accumulator);
TRF register, register (transfer register to register);
EXG register, register (exchange register to register);
PSHS (registers) (push registers onto hardware stack);
PSHU (registers) (push registers onto user stack);
PULS (registers) (pull registers from hardware stack);
PULU (registers) (pull registers from user stack).

Pri 16-bitni notranji arhitekturi uporablja procesor 6809 le 8-bitno podatkovno vodilo tako kot 6800 in druga periferna integrirana vezja.

APŽ

Mikroračunalniški sistem TRS-80 v Evropi.

Eden najbolj razširjenih in znanih osebnih ter majhnih poslovnih računalnikov TRS-80, ki ga masovno proizvaja ameriška firma Radio Shack, se je končno pojavil tudi na evropskem tržišču, seveda ustrezno modificiran (na petost, frekvenca). Sistem uporablja profesionalno tastaturo s 53 tipkami, trenutno najbolj zmogljivi 8-bitni mikro procesor Z-80, pomnilnik pa je moč povečati z dodatnim modulom. K osnovni konfiguraciji sodi še kasetni zapisovalnik, video prikazovalnik (RCA) in dokaj izčrpna dokumentacija za uporabo in programiranje v jeziku BASIC. Osnovna konfiguracija (vsi usmerniki so vključeni) ima evropsko ceno DM 1.795,- (ali FF 3.995, ali FB 29.995), dodatni pomnilnik (16 K zlogov) pa stane DM 889,-. Osnovna konfiguracija vključuje tudi programsko opremo in sicer 4 K monitor s prevajalnikom za jezik BASIC ter 4 K RAM. Naslov evropskega zastopnika je: Tandy Corporation, Parc Industriel, 5140 Narinne, Belguique.

APŽ

Vezja z visoko stopnjo integracije firme FUJITSU.

Iz znanstveno tehnične publikacije firme FUJITSU povzemamo med drugim tudi stanje in tehnološko napoved na področju tehnologije z največjo stopnjo integracije. Pomnilniška vezja (enote, čipi) so v letu 1972 obsegala 1 K bitnih lokacij, v 1974: 4 K, v letu 1976: 16 K, kar je predstavljalo štirikratno povečanje na vsaki dve leti. Letos, v letu 1978, so v prodaji že 64 K bitne enote, medtem ko v letu 1980 napovedujejo prodajo 256 K bitnih in v letu 1982 prodajo 1 M bitnih integriranih vezij. Pri teh napovedih ne gre več za prazni verbalizem, saj je firma FUJITSU skupaj z Musashino Electrical Communication Laboratory in z VLSI Technology Research Association začela z izvajanjem teh projektov že v letu 1976, s ciljem, da osvoji industrijsko proizvodnjo 1 M bitnega čipa v štirih letih.

Teoretične meje so pri današnjem stanju tehnologije, da se v eno integrirano vezje pospravi od 3,6 do 10 M bitov pomnilnika. Z novo tehnologijo, ki bo omogočala 100-krat večjo integracijo (vzorci se realizirajo z elektronskim snopom) pa bodo lahko dosegli gostoto 1000 M bitov na vezje.

APŽ

Kaj tare ameriške računalniške inženirje.

V poljetjih se pojavljajo oblike pomanjkljive koordinacije tehničnega dela: npr. vodja razvoja poroča vodji proizvodnje, ki razume o TTL, CMOS, programski opremi in mikro računalnikih toliko, kot razume inženir o pisanju simfonične glasbe ...Znana institucija IEEE je primerna za cenene kupčije ter za življensko zavarovanje; upravljajo jo združeni (klanovski) izvršniki in akademiki (namesto inženirjev) ... Ali že veste,kako je mogoče dobiti doktorat od pooblaščene univerze za dopisno šolanje brez šolanja,poprejšnje in strokovne izobrazbe? Zakaj IEEE takšnih univerz ne odpihne? Najbrž zategadelj, ker sedi v njenem odboru sedem vplivnih članov z bivšim predsednikom IEEE ...

To so izjave inženirjev, ki jih navaja Digital Design (marec 1978). Splošne pripombe inženirjev pa so: mezdni odnosi, starostna diskriminacija, neplačane nadure, periodična brezposelnost, poslabšano javno mnenje, prenizke razlike v plačah, izkoriščanje inženirjev v industriji, nepraktični in teoretično predimenzionirani kurzi na univerzah itn. Očitno lahko opazimo podobne pojave tudi drugje po svetu.

APŽ

Multiprocesorski sistem na bazi mikro procesora.

Razvojna grupa Scicon's Micro Systems team i National Physical Laboratory su razvili multiprocesorski računar. Prototip sistema će biti izrađen u toku sljedeće godine uz pomoć fondova Government's Advanced Computer Technology Project (ACTP). Komercialni sistem baziran na prototipu će biti vrlo fleksibilan sa mogućnošću skoro neograničenog širenja. Prva aplikacijska područja, za koja će sistem biti priređen, će biti kontrola u realnom vremenu, transakcijsko procesiranje i komunikacije.

Prototipni sistem pod imenom DEMOS će biti sposoban korištenja modernih jezika i tehnike produciranja programske opreme sa akcentom na smanjenju njene cijene. Osnovna DEMOS programska oprema će biti pisana jezikom Concurent Pascal, koji nudi velike mogućnosti za brz i kvaliteta razvoj sistemske programske opreme.

Svaki komponentni računar DEMOS-a ima svoju vlastitu memoriju. Međusobne komunikacije kontrolišu mali, identični programi koji se nalaze u svakom računaru. Ovdje se radi o 8 M-riječi paralelnom prstenu koji može povezati do 250 računara. Jedino međuspojevi (interfaces) između svakog računara i prsten moraju biti prilagođeni potrebama pojedinog upotrijebljenog računara.

DEMOS programska sistemska oprema je pisana kao jedan program. Kada je taj program preveden vrši se dodjeljivanje njegovih komponenata individualnim računarima pomoću posebnog programa za generiranje sistema. Dakle sistemska programska oprema može biti primijenjena na veće i manje sisteme bez potrebe izmjena. To omogućava jednostavno širenje ili povremeno reduciranje sistema u slučaju kvarova na pojedinim elementima. Sistemska programska oprema će biti pisana tako da podržava pojedine aplikacije ili grupe aplikacija. Aplikacijski programi mogu biti pisani u različitim jezicima realnog vremena i mogu biti prevođeni posebno.

MK

Oči za robote.

British Rail upotrebljava VP 102 video lokatore proizvodnje Hampton Video Systems Ltd., pri istraživanju na području industrijskih robota. VP 102 je hardware-ski video procesor koji prihvata informacije iz standardne televizijske kamere te daje digitalnu informaciju o lokaciji cilja za računar. Ovaj procesor omogućava brzo i jeftino riješenje za bezkontaktno određivanje pozicije.

Ovi lokatori se mogu koristiti za industrijske robote i manipulacijske sisteme zajedno sa televizijskom kamerom montiranom na robotu ili na "radnom mjestu".

MK

Novi sistemi ICL.

ICL je objavio dolazak na svijet novog sistema 2972 te poboljšane verzije 2976. Novi sistemi imaju znatno povoljniji odnos cijena/performanse te predstavljaju konkurenčne mašine za IBM 3030 serije. ICL upotrebljava 16K-bitne memorijske čipove za obje mašine, dok IBM ima 2K-bitne čipove na 3030 seriji.

Sa cijenom između 1,3 do 1,8 miliona dolara 2972 je nešto skuplji od 2970, ali zato nudi za 50% bolje performnse. Order code Procesor (glani dio CPU jedinice) je upotrijebljen kako u oba nova sistema, tako i u najvećem sistemu 2780. Inače ICL priprema novi OCP koji će biti uključen u budući sistem 2986 koji treba da zauzme poziciju najvećeg u njihovom asortimanu.

Isti kontroleri za dostup do memorije su upotrijebljeni kako kod 2972 tako u 2976, pri čemu je ovaj drugi brži za 20% pošto sadrži još i međumemorijske elemente.

Sa ciljem da olakša prelazak iz serije 1900 na seriju 2960 ICL je pripremio emulator sistema 1900 na sistemima 2960 pod nazivom Direct Mashine Environment + . Time je omugućen prenos programa iz mašine serije 1900 na mašinu serije 2960.

MK

Report sa National Computing Conference California.

Na ovaj priredbi se moglo vidjeti obilje novih proizvoda. U vezi sa novim procesorima zapaža se ogromna upotreba bit-slice mikroprocesorske tehnologije (LSI tehnologija koja predpostavlja horizontalnu modularnost procesorkih elemenata - modularnost po bitovima), koja predstavlja danas revoluciju u računarskom dizajnu.

Najpopularnija naprava na ovom području je AMD 2901. Ovaj 4-bitni mikro procesor je upotrijebljen u novom modelu Honeywel Level 6 mini računarskoj seriji i u moćnoj, novoj seriji 500, Harris. Na izložbi su bili predstavljeni modeli 550 i 570. Sistem ima 24-bitnu riječ, ali i 48 bitno centralno sistemsko vodilo. Za povećanje brzine djelovanja je uključena cashe memorija sa vremenom dostupa od 80 nsec te 2K riječi. Sistemi su potpuno software-sko kompatibilni sa serijom 100. Sistem podržava višejezični virtualni operacijski sistem sa mogućnošću djelovanja u više načina, ali također ima i svoj vlastiti programski način djelovanja.

BTI Computer Systems of California je predstavila svoj novi tip, 32-bitni računar sa vremenskim dedeljivanjem (time sharing). Namijenjen Evropskom tržištu, BTI 8000 je modularni procesor izgraden nad brzim (67 nsec) vodilom. Projektiran je za uspješno izvodenje kompilatora pisanih u Pascal-u.

Jacquard uvodi novi flopy-disk sistem izgraden na osnovu TI 280 (Texas Instruments) 4-bitnog bit-slice procesora. Sistem je 25 puta brži od svog predhodnika (J50), ima oko 30 ključnih funkcija koje se mogu programirati. Njegova cijena od 9200 $ je vrlo povoljna (J50 - 14000 $). Programska oprema sistema J50 i J 100 je kompletno upotrebljiva na novoj mašini.

Logical Machine Corp. je upotrijebila Intelov 3-bitni slice uP (mikro processor) kao osnovu za svoj novi Adam stolni računar, Adam the Younger. Ključna osobina Adam-a je ta da je programiran u ultra visoko nivojnom jeziku (jezik blizu prirodnog jezika). Sistem se sestoji od terminala sa vizuelnom jedinicom koji sam sadrži procesor, dva dvostrana flopy diska sa dvojnom gustinom i linijskog pisača sa brzinom pisanja od 110 kolona u sekundi. Cijena (15000 $).

Jedna od zanimljivosti na ovoj priredbi je bio džepni RF Radio podatkovni terminal sa Motorolinog štanda, funkcionalno kompatibilan sa IBM 3270. Taj terminal je projektiran za komuniciranje sa IBM 370 sistemima. Terminal je težak svega pola kilograma, 16-znakovni display, 480 znakova memorije te kanal za prenos glasa. Za komunikaciju sa drugim operatorima. Terminal djeluje na frekvenci od 480 MHz, cijena je 60 $.

Novo iz Racal-Milgo je display terminal potpuno kompatibilan sa IBM 3275 (cluster) koji se pojavio na tržištu prije godinu dana. Takav "grozdasti" terminal podržava do 8 terminala ima za osnovu uR kontroler. Grozdasti terminal 4000 se pojavljuje u verzijama sa dva alternativna komunikacijska kontrolera te masovnom memorijom do 1 Mbyt na flopy disku. Na raspolaganju su pored IBM 3277 te 3271 protokola još i protokoli Univac UT5400 te Honeywell VIP 7760.

Na IBM-ovom štandu je bilo moguće vidjeti SNA-kompatibilan 3630 industrijski terminal, kojeg je do sada bilo teško naći na tržištu.

Control Data je razotkrio svoju certainty klasu OEM periferije za IBM Series 1 mini računar.

MK

Majhni roboti.

Doba robotov je pred vrati še zlasti po zaslugi mikro računalnikov. Večina ljudi misli, da so roboti posledica industrijske revolucije; toda nežive robote je človek poznal že pred več tisoč leti. Zamisel umetnega (nenaravnega) bitja je človeka razburjala že daleč pred našo ero. V Iliadi je Homer opisal robotske sle, ki so služili Vulkanu, bogu naravnega ognja, kot zlati trokolesni mehanizmi, s kolesi iz čistega zlata in z nožicami za nenavadne zvijače. Platon pripoveduje o mehaničnih možeh, ki so bili tako sposobni, da so jih morali zvezati, ker bi sicer zbežali.

Med mitom (vizijo, nejasno predstavo) in realnostjo (empirično možnostjo) so se pojavili androidi. Beseda android je izvedena iz grških besed andros in eidos in pomeni "podoben človeku". Menih Albertus Magnus je porabil 30 let za izdelavo mehaničnega moža oziroma androida. Ta android je baje stražil Magnusova vrata v kolonjskem samostanu, pozdravljal obiskovalce ter jih spraševal, zakaj so prišli. V 13. stoletju je zgradil "železnega moža" tudi Robert Bacon.

Rokodelci prejšnjih stoletij so bili sposobni izdelati skoraj neverjetne androide, avtomate in lutke. V 18. stoletju sta Pierre Jacquet-Droz in njegov sin Henri-Luis kreirala "Pisatelja", tj. 75 cm velikega mladeniča, ki je namakal pero v črnilnik z zamahom ter pisal na list papirja. Telesni gibi Pisatelja so bili dokaj realistični in oči so natanko sledile vrsticam pred gibanjem peresa. Proti koncu 19. stoletja je potoval po Evropi Kemplenov šahovski avtomat, ki je premagoval vrhunske evropske tekmovalce, dokler niso ugotovili prevare. V notranjosti so našli prismojenega pritlikavca, ki je preko vzvodov premikal figure na šahovnici. Toda v letu 1914 je španski inovator Torres Y Quevedo zgradil pravega robota za igranje šaha, vendar le za igro z nekaj figurami.

Robote so uporabljali tudi v industriji: v tiskalnih stiskalnicah, avtomatičnih statvah, bombažnih vitlih itn. To so bili primitivni roboti, dokler v letu 1784 ni začela obratovati avtomatična tovarna v Ameriki. To je bil mlin na Red Clay Creeku, ki je dal izdelek, nedotaknjen s človeškimi rokami. Razvoj v letu 1940 je pospešil uporabo robotov: pojavila se je mehanična roka, ki je kasneje dobila "možgane" v obliki računalnika. Mikro računalniki pa so končno omogočili izdelavo zapletenih avtomatov.

Podjetje Gallaher Research Inc. (P.O.Box 10767, Salem Station, Winston-Salem, N.C. 27108, USA) je začelo izdelovati majhne, cenene robote (cena sestavljenk do $ 1000) za amaterje, ki jih je moč preizkušati tudi v industrijskem okolju. Maksimalna konfiguracija obsega dve roki (vsaka s po 6 motorji) in pomikanje na treh kolesih (hitrost 3km/h). Ta robot tehta 20 kg, zagrabi lahko kose do 5 cm široko s pritiskom prstov 4N ter vrti ramena za 180°, zapestja za 180° ter upogne roko, komolec in gornji del roke, vsakega za 45°. Robota lahko krmilimo od zunaj, in sicer vsak motor posebej in z eksperimentiranjem se da v kratkem času doseči, da robot opravlja želene akcije.

APŽ

Sistem za obradu tekstova.

Data Recall nudi na tržištu sistem za obradu tekstova Diamond, koji može zapamtiti tekst sa 450 stranica A4 formata. Sistem uključuje sljedeće materijalne komponente: VDU jedinica sa scroling načinom djelovanja koja može prikazati 24 linije od 80 znakova. Alternativne verzije mogu pokazati 64 linije po 80 znakova ili 32 linije od 132 znaka.

Tekst može biti zapamćen na tri flopy disk jedinice, svaka sa kapacitetom od 250 000 znakova, te može biti ispisan na printer brzinom 45 znakova u sekundi. Printer može primiti normalni A4 format ili 375 širok format. Diamond sistem omogućava simultano operiranje tastature, pamćenja teksta i ispisivanja teksta. Funkcije, kao na primjer sortiranje, editiranje, editiranje na ekranu uz pomoć kursora, automatsko formatiranje, centriranje naslova, automatsko ispisivanje datuma, editiranje zadnjeg znaka, dijeljenje riječi i linija, i druge se ostvaraju pritiskom na dugme. Vrijeme dostupa do svakog podatka ili bilo koje funkcije je maksimalno 1 sekunda.

Diamond može, također biti upotrijebljen za razna obračunavanja i opšte računarske aplikacije. Materijalna oprema za priključenje na tekst je također na raspolaganju.

MK

Ko pišete proizvajalcu, omenite časopis INFORMATICA.

raziskovalne naloge prijavljene na rss v letu 1978

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: Načrtovanje elektromehanskih regulacijskih postrojev
Projekt: Krmilja in regulacije
Nosilec naloge: K. Jezeršnik, VTŠ Maribor, VTO Elektrotehnika

Program raziskave:

A. Okvirni program 3-letne naloge je bil podan ob prijavi I. faze leta 1976.

B. Podrobni program raziskave za 3. leto:
   1. Pregled usmerniških in razsmerniških postaj - primerjava 6-pulznih in 12-pulznih vezij.
   2. Modeliranje podsistemov: 6- in 12-pulzni usmerniki in razsmerniki, filtri za zmanjšanje višjeharmonskih komponent. Modeli sinhronskega stroja, prenosnih vodov energetskih transformatorjev in možnostnih usmernikov bodo prevzeti iz raziskave v I. in II. letu.
   3. Digitalna simulacija sinhronskega stroja, enosmerne sklopke in frekvenčno čvrstega omrežja ter študij vplivov naprav med sabo.
   4. Digitalna simulacija več elektroenergetskih omrežij povezanih z enosmerno sklopko.

- - - -

Nosilec naloge: Raziskava računalniških modelov za razvijanje vozil
Projekt: Računalniško projektiranje
Nosilec naloge: M. Prašnički, VTŠ Maribor

Program raziskave:

Naloga je 3 letna, bi pa se predvidoma razširila na 5 let. Po prvem letu bi zbrali računske postopke in preverili ustreznost in uporabnost postavljene splošne metodologije raziskav tako, da bi ocenili neposredno korist na modelih reševanja vzmeti, voznih sil in zavor. Poznejše raziskave bi bile nadaljevanje in poglabljanje na ostale funkcijske sklope. V prvem letu bo predložen tudi informacijski sistem za službe in dejavnosti, ki razvijajo cestna vozila in opravljajo vzporedne raziskave in sicer tako, da bo omogočil racionaliziran pretok in koriščenje tehničnih informacij med konstrukcijo, preizkusom in tehnologijo.

- - - -

Naslov naloge: Sodobni koncepti upravljanja v industrijskih procesih
Projekt: Računalniška avtomatizacija industrijskih procesov
Nosilec naloge: A. Čižman, IJS, Ljubljana

Program raziskave:

- Računalniška identifikacija multivariabilnih sistemov, analiza in sinteza adaptivnih observerjev ter implementacija na problemu distribucije plina. Poudarek je na razvoju identifikacijskih metod za sprotno identifikacijo parametrov procesa, ki ga želimo uporabljati s procesnim računalnikom.
- Uporaba konjugiranih gradientnih in metričnih metod pri optimalnem upravljanju energetskih sistemov.
- Matematično modeliranje dinamičnih sistemov na osnovi vhodno-izhodnih meritev. Razvite metode bomo implementirali pri določevanju matematičnih modelov v cementni industriji in pri makroekonomskem modeliranju.

- - - -

Naslov naloge: Množilni moduli za mikroračunalnike
Projekt: Računalniška tehnika in proizvodnja
Nosilec naloge: B. Stiglic, VTŠ, VTO Elektrotehnika, Univerza v Mariboru

Program raziskave:

V programskem letu 1978 bodo raziskane metode in možnosti hardwarskega množenja z LSI vezji, ki so sedaj na razpolago na tržišču, glede na potrebno materialno in programsko opremo in ceno ter glede na možnost čim bolj neposredne in preproste povezave z mikroprocesorjem in drugimi enotami mikroračunalnika. Rezultati raziskave bodo preverjeni z izdelanimi množilnimi moduli, ki jih bomo povezali v mikroračunalnik ISKRA DATA 1680.

Delo na tej nalogi je usklajeno z delom drugih nalog projekta Proizvodnja materialne in programske računalniške opreme.

- - - -

Naslov naloge: Večnivojski sistem upravljanja z mini in mikro računalniki II
Projekt: Računalniška avtomatizacija industrijskih procesov
Nosilec naloge: J. Tasič, IJS, Ljubljana

Program raziskave:

- Teoretski aspekti večnivojskega upravljanja
- Struktura optimalne programske opreme za večnivojsko upravljanje
- Zajemanje, koncentriranje in prenos podatkov z mikro računalnikom, analiza metod in realizacija na domačem sistemu PDP in SC/MP.

- Samoučeči se programski sistem PRODIP
- Prilagoditev klasičnih konceptov direktnega digitalnega vodenja procesov pogojem mikroračunalniškega sistema,
- Preizkus razvoja algoritma za spratno identifikacijo procesov za namene optimalne in adaptivne regulacije,
- Poenostavljanje velikih sistemov in njih razstavitev na hierarhično vodenje z mikroračunalniki.

- - - -

Naslov naloge: Intervencijska analiza z uporabo časovnih vrst
Projekt: Informacijski sistemi
Nosilec naloge: D. Čepar, IJS, Ljubljana

Program raziskave:

- Preučevanje metod za analizo posegov v sisteme na osnovi analize časovnih vrst in napovedovanja s pomočjo dosegljive strokovne literature in s pomočjo stikov z drugimi skupinami doma in v svetu, ki se ukvarjajo s podobnimi problemi;
- Teoretično delo na posameznih metodah vključno s preučevanjem učinkov nekaterih družbeno ekonomskih pojavov na druge pojave z matematičnimi modeli;
- Preizkušanje metod in programov na primerih iz prakse in zbiranje informacij o izkušnjah in o področjih uporabe teh programov drugod v svetu;
- Izdelava programov za računalnik CYBER 72 v obliki paketa, ki ga bo enostavno uporabljati in splošno dosegljiv;
- Vzdrževanje in konzultacije pri uporabi programov, ki so bili narejeni v okviru prejšnje raziskovalne naloge;
- Pisanje in razširjanje informacij o uporabnih programih in priročnikov za uporabo programov;
- Organizacija in vodenje seminarjev o uporabi teh metod in programov in pomoč pri njih vpeljavi v organizacijah združenega dela.

- - - -

Naslov naloge: Priprava prevajalnika PASCAL za PDP-11
Projekt: Računalniška tehnika in proizvodnja
Nosilec naloge: B. Barlič, Kemijski institut "B.Kidrič", Ljubljana

Program raziskave:

Priprava prevajalnika za sistem RSX-11M. Delo bo opravljeno na računalniku Cyber v RRC. Glede na to da je prevajalnik sam napisan v Pascalu in da edini dostopni Pascalov prevajalnik teče na tem sistemu, je ta najbolj ekonomičen način za pripravo prevajalnika.

Priprava sistemske knjižnice podprogramov za Pascal pri čemer bomo čim bolj izkoristili že obstoječe podprograme, ki jih uporablja Fortran.

Manjše izboljšave prevajalnika, kot so izpisi dodatnih informacij, itd.

Priprava dokumentacije (priročnik, poročilo) in navezava stika z ostalimi interesenti.

AVTORJI IN SODELAVCI

Rudolf MURN (1930), diplomiral leta 1957 na Fakulteti za elektrotehniko v Ljubljani. Magistriral leta 1966 in doktoriral leta 1974 s področja teorije diagnostike večkratnih napak v digitalnih vezjih. Leta 1976 je prejel skupaj z dvema sodelavcema nagrado Sklada Borisa Kidriča. Je redni sodelavec Instituta "Jožef Stefan, dela na raziskovalnih in aplikativnih nalogah na področju simulacije in testiranja ter načrtovanja digitalnih sistemov. V časopisu Informatica opravlja funkcijo tehničnega urednika.

Franc KOŠIR (1944), diplomiral leta 1967 na Pravni fakulteti v Ljubljani. Zaposlen v Regionalni zdravstveni skupnosti Ljubljana kot vodja centra za avtomatsko obdelavo podatkov od leta 1971 dalje. Poleg organizacijsko operativnih opravil se ukvarja predvsem z načrtovanjem in realizacijo informacijskih sistemov v zdravstvu in aplikativne uporabe računalniške tehnologije v zdravstveni informatiki.

Janez ERŽEN (1948), diplomiral leta 1972 na Fakulteti za elektrotehniko v Ljubljani. Leta 1977 je tam tudi magistriral. Zaposlen je na Železniškem gospodarstvu Ljubljana - Prometni institut. Do sedaj se je ukvarjal z računalniško simulacijo vleke in železniškega prometa, sedaj pa opravlja raziskave s področja zanesljivosti železniških signalno-varnostnih naprav in varnosti železniškega prometa.

Dušan PEČEK (1952), diplomiral leta 1977 na Fakulteti za elektrotehniko v Ljubljani, smer računalništvo in informatika, s tematiko iz področja simulacija digitalnih sistemov. Zaposlen je na Institutu "Jožef Stefan" v Ljubljani, Oddelek za elektroniko. Ukvarja se s problematiko simuliranja poljubnih digitalnih struktur. Iz tega področja je objavil več del v jugoslovanskih časopisih.

CENIK OGLASOV

Ovitek - notranja stran (za letnik 1977)
2 stran ---------- 16.000 din
3 stran ---------- 12.000 din

Vmesne strani (za letnik 1977)
1/1 stran ---------- 8.000 din
1/2 strani ---------- 5.000 din

Vmesne strani (za posamezno številko)
1/1 stran ---------- 3.000 din
1/2 strani ---------- 2.000 din

Oglas o potrebah po kadrih (za posamezno številko)
---------- 1.000 din

Razen oglasov v klasični obliki so zaželjene tudi krajše poslovne, strokovne in propagandne informacije in članki. Cena objave tovrstnega materiala se bo določala sporazumno.

ADVERTIZING RATES

Cover page (for all issues of 1977)
2nd page ---------- 16.000 din
3rd page ---------- 12.000 din

Inside pages (for all issues of 1977)
1/1 page ---------- 8.000 din
1/2 page ---------- 5.000 din

Inside pages (individual issues)
1/1 page ---------- 3.000 din
1/2 page ---------- 2.000 din

Rates for classified advertizing:
each ad ---------- 1.000 din

In addition to advertisments, we wellcome short business or product news, notes and articles. The related charges are negotiable.

THE COMMISSION OF THE EUROPEAN COMMUNITIES

SYMPOSIUM ON COMPUTER AIDED DESIGN OF DIGITAL ELECTRONIC CIRCUITS AND SYSTEMS

The Commission of the European Communities will organise the above-mentioned Symposium for the benefit of designers and users of electronic digital circuits and systems, to be held in Brussels, Belgium on 27 – 29 November, 1978.

The aim of the Symposium is to disseminate the results of the feasibility "Computer Aided Design Electronics Study" undertaken by the European Communities and further to present an assessment of the state of the art of techniques, problem areas and possibilities of further developments in the field of computer aided design (CAD) of digital electronics. Results of the European Economic Communities feasibility study (conducted from July 1977 through September 1978 by a consortium comprising Brunel University, Nixdorf, Plessey, SEMA and SAGET) have been drawn on the basis of a two-iteration survey of 85 institutions in Europe, US and Japan, and a subsequent detailed analysis of the information gathered. Particular attention will be focussed to the impact of the rapid evolution of LSI component and computer technology and to the highest potential opportunity to European Economic Communities Member States, to reach the most beneficial recommendations for possible development work.

Some of the topics of the Symposium are:

Product specification and synthesis, modelling and logic simulation, design for testability, layout, engineering data base, integrated Computed Aided Design Systems, Custom LSI economics, Computed Aided Design for LSI devices versus Computed Aided Design for PCB requirements.

The Symposium will gather senior European Communities officials, Study contractor representatives, and over twenty invited speakers specialising in these fields from Europe, United States and Japan.

Registration Fees BF 6,000.00 before October 31st and BF 7,500.00 thereafter.

For further information, please apply to:

KENESS BELGIUM CONGRESS S.A.
Rue de l'Industrie 17
1040 Brussels, Belgium
Tel. (02) 230 09 53
Telex B 62995

Glavno predsedstvo za notranje tržne in industrijske zadeve evropskih skupnosti organizira simpozij na temo "Načrtovanje digitalnih elektronskih sistemov in vezij, s pomočjo računalnika". Osnovni podatki o simpoziju so:

datum: 27.-29. november, 1978

mesto: Brussels (Belgija), Hotel Hilton

pokrovitelj: Komisija evropskih skupnosti

sekretariat: Keness Belgium Congress S.A.
Rue de l'Industrie, 17, 1040 Brussels
tel. 230 09 53, telex 62 995

cilji:
- predstavitev rezultatov študije evropske skupnosti o možnosti razvoja elektronskih sistemov in vezij s pomočjo *računalnikov*, *ki so bili dobljeni* v Evropi, ZDA in na Japonskem; študija bo predvidoma končana v septembru 1978.
- razpravlja o stanju svetovne tehnologije in o možnih razvojnih akcijah, ki bi povečale učinkovitost načrtovanja s pomočjo računalnika.

predavatelji: povabljeni so priznani strokovnjaki iz Evrope, ZDA in Japonske.

Iskradata

Računalnik Iskradata C 18

- STROJNA OPREMA ZA POSLOVNO IN PROCESNO UPORABO
- FLEKSIBILNA MODULARNA ZGRADBA IN MOŽNOST RAZŠIRITVE
- PROCESOR Z MOŽNOSTJO MIKROPROGRAMIRANJA
- CENTRALNI POMNILNIK DO 512 K BYTE, DISKOVNE ENOTE DO 400 M BYTE, DO 4 TRAČNE ENOTE, DO 64 ZASLONOV S TASTATURO
- SISTEMSKA IN APLIKACIJSKA PROGRAMSKA OPREMA ZA PROCESNO IN POSLOVNO UPORABO
- POVEZAVA NA VEČJE RAČUNALNIKE

Mikroračunalnik Iskradata 1680

-MODULARNO GRAJEN
-OMOGOČA POLJUBNO KONFIGURIRANJE
-NASLAVLJANJE DO 64 K BYTE
-SERIJSKI IN PARALELNI VHODNO/IZHODNI MODULI
-PERIFERNE NAPRAVE: DISKETNE ENOTE, ZASLONI S TASTATURO, TISKALNIK, TELEPRINTER
-ŠOLSKI ENOMODULNI RAČUNALNIK
-LABORATORIJSKI RAČUNALNIK
-PROCESNI RAČUNALNIK
-POSLOVNI RAČUNALNIK

spremljajoče dejavnosti

-LASTEN RAZVOJ STROJNE, SISTEMSKE IN APLIKACIJSKE PROGRAMSKE OPREME
-ŠOLANJE
-TEHNIČNO VZDRŽEVANJE
-STROKOVNA POMOČ

ISKRA, Industrija za telekomunikacije, elektroniko in elektromehaniko, Kranj, TOZD Računalniki,
64000 Kranj, PE Ljubljana, Titova 81,
tel (061) 326-367, 322-241

mikroračunalniki

STE SE ŽE ODLOČILI ZA UPORABO MIKRORAČUNALNIKA V PROIZVODNJI, ALI PA ŽE IMATE MIKRORAČUNALNIK IN ŽELITE IZVESTI REŠITVE PO VAŠI ZAMISLI?

NUDIMO VAM:

- NAJSODOBNEJŠE TEHNIČNE REŠITVE Z UPORABO NOVE TEHNOLOGIJE
- PROJEKTIRANJE VEČJIH SISTEMOV (MULTIPROCESORSKIH)
- DIAGNOSTICIRANJE PROCESOV
- IZDELAVO IN TESTIRANJE UPORABNIŠKIH PROGRAMOV S POMOČJO VELIKIH SISTEMOV
- KONZULTACIJE ZA UPORABO MIKRORAČUNALNIŠKIH SISTEMOV V INDUSTRIJI IN GOSPODARSTVU
- IMAMO IZKUŠNJE Z UPORABO MIKROPROCESORJEV (Z 80, 6800, F–8, 8080, 2650, PFL 16, SC/MP), DINAMIČNIH POMNILNIKOV, PERIFERIJE ITN.

Institut Jožef Stefan, Ljubljana, Jamova 39
ODSEK ZA RAČUNALNIŠTVO IN INFORMATIKO
Telefon (061)63–261 Int. 305

univerza v ljubljani

institut "jožef stefan" ljubljana, jugoslavija

Ali ste strokovnjak s področja računalništva, ki želi razvijati svoje sposobnosti v ustreznem okolju?

Če imate delovne navade, organizacijske sposobnosti, izkušnje pri delu z ljudmi, samostojnost in iniciativnost pri poslovnem ali raziskovalnem delu, se oglasite pri nas — pogovorili se bomo o možnostih sodelovanja.

Omogočimo vam lahko razvijanje vaših lastnih pristopov in uveljavljanje sposobnosti na področjih, ki vam najbolj ustrezajo.
Nudimo vam možnost izobraževanja doma in v tujini, dobre delovne pogoje, nagrajevanje po delu, strokovno okolje, pomoč sodelavcev ter dostop do strokovne literature.

Oglasite se na Odseku za uporabno matematiko — prisluhnili bomo vašim predlogom in željam.

INSTITUT "JOŽEF STEFAN"
Jamova 39, 61000 Ljubljana
tel. 263–261/int. 284