# PROGRAMSKI SIMULATOR – KORAK K RAZVOJU MIKRORAČUNALNIŠKIH APLIKACIJSKIH PROGRAMOV

D. MILJAN,
J. ŠILC,
P. KOLBEZEN

UDK: 681.3.06:519.682

INSTITUT „JOŽEF STEFAN", LJUBLJANA

Analiza cene razvoja računalniškega sistema kaže, da večji del cene odpade na razvoj programske podpore, zato je izredno pomembno, da ima razvijalec na voljo učinkovite pripomočke za razvoj programske podpore. V članku so nakazani nekateri primeri tovrstnih pripomočkov. Natančneje je opisan pri nas razvit programski simulator za razvoj aplikacij z domačim integriranim mikroračunalnikom Iskra EMZ 1001.

SOFTWARE SIMULATOR - STEP IN DEVELOPMENT OF MICROCOMPUTER APPLICATION PROGRAM. Trends in computer system development costs show quite clearly that the major proportion of these costs must be attributed to software design and implementation. Some microcomputer software development aids available to programmer are introduced. Software is required both to execute a particular application on a system and to assist in the development of new software. Example of software simulator for Iskra EMZ 1001 single chip microcomputer is given.

## UVOD

Računalniške programe smo običajno razvijali bodisi na računalniku za katerega smo program pisali, ali na njemu podobnem, če je bila njegova konfiguracija takšna, da je omogočala uporabo potrebne opreme (programske in materialne) za razvoj aplikacijskih programov. Pod razvojno programsko opremo mislimo: urejevalec teksta (editor), ustrezen prevajalnik in povezovalnik (linker). Pri razvojni materialni opremi je osnovna zahteva dovolj velik pomnilnik in možnost komunikacije človek-stroj s pomočjo CRT-ja ali teleprinterja.

Pri 'večjih' računalnikih običajno ni tovrstnih problemov, pri mikroračunalnikih pa smo pogosto soočeni s pomankanjem pomnilniškega prostora in revnejšo periferno opremo za razvoj aplikacijskih programov. Ti problemi postanejo akutni pri integriranih mikroračunalnikih (single chip computer), pri katerih rešujemo problem razvoja aplikacijskih programov s pomočjo t.i. razvojnih sistemov. V tem prispevku se bomo omejili na področje mikroračunalnikov. Razlike v načinu razvijanja aplikativnih programov pri mikroračunalnikih glede na 'večje' računalnike izhajajo iz naslednjih ugotovitev :

- mikroprocesorje velikokrat uporabljamo za posebne namene v sklopu računalnikov na eni tiskani plošči (single board computer) ali v sklopu posebno načrtovanih vezij za posamezne aplikacije. Takšne minimalne računalniške konfiguracije običajno niso primerne za razvoj aplikativnih programov,
- načrtovalec mikroračunalniških sistemov lahko v odvisnosti od tipa aplikacije izbira med različnimi tipi mikroprocesorjev. Možnost, da bi za vsak od izbranih mikroprocesorjev imel svoj razvojni sistem z vso programsko in materialno podporo za razvoj aplikativnih programov, izključujejo ekonomski razlogi.

Kot je omenjeno, poznamo več tipov razvojne opreme in postopkov za razvoj aplikativnih programov za mikroračunalnike. Nekatere rešitve nudijo proizvajalci računalniške opreme, nekatere pa razvijajo uporabniki mikroprocesorjev sami.

Razvojno opremo in postopke lahko razdelimo v nekaj kategorij :

### i. Mikroračunalnik z osnovnim monitorjem v ROM pomnilniku.

Takšen sistem daje uporabniku minimalne možnosti za vpisovanje in testiranje aplikacijskega programa preko terminala. Takšen monitor je lahko koristen pri razvoju in testiranju manjših programov (do največ 100 računalniških besed) in je praktično neuporaben pri večjih aplikacijah. Kljub temu, takšno opremo ne smemo kar tako izključiti kot eno od možnih razvojnih pripomočkov že zaradi zelo široke uporabe mikroprocesorjev, pri kateri včasih rešujemo programske probleme tudi s takšnim monitorjem.

### ii. Mikroračunalnik opremljen z zbirnikom in urejevalcem teksta.

Programa lahko stalno zasedata del ROM pomnilnika mikroračunalnika ali se po potrebi prečitata iz papirnega traku v RAM pomnilnik. Tak sistem omogoča pisanje in popravljanje aplikativnih programov v zbirnem jeziku in shranjevanje le-teh na papirni trak. Čeprav je to korak naprej od samega monitorja, še zdaleč ni omogočeno učinkovito aplikativno razvojno delo večjih dimenzij. Dodatno nerodnost predstavlja prepisovanje zbirnika in urejevalca texta iz papirnega traku v RAM pomnilnik

mikroračunalnika. Manjšo, a ne tako bistveno izboljšavo tega razvojnega sistema predstavlja nadomestitev sistema s papirnim trakom z sistemom z digitalno kaseto.

iii. Razvojni sistem na bazi uporabljenega mikroprocesorja.

Tak sistem, opremljen z gibkimi diski, omogoča pisanje, popravljanje in urejevanje aplikacijskega programa. Nadalje omogoča še zbiranje in prevajanje ter shranjevanje izvirnih in prevedenih programov. Takšen sistem predstavlja že relativno močno orodje pri razvoju aplikacij, ima pa eno večjo pomanjkljivost (gledano s stališča uporabnika) in to je, vezanost uporabnika na en tip ali eno družino mikroprocesorjev, za katero je razvojni sistem zgrajen. Pri uporabnikih, ki želijo (ali so prisiljeni) izbirati med različnimi mikroprocesorji za različne aplikacije, takšen razvojni sistem ne more biti trajna rešitev.

iv. Razvojni sistem na bazi poljubnega mikroprocesorja.

Takšen sistem (z gibkimi diski) vsebuje poleg urejevalca teksta običajno še nekaj različnih zbirnikov ali prevajalnikov za posamezne tipe mikroprocesorjev katere podpira. Omogočene so vse faze razvoja aplikacijskih programov do faze testiranja. Za testiranje je potrebno objektno kodo aplikacijskega programa prenesti na realno mikroračunalniško aplikacijo. Ta problem lahko rešujemo z dodatno opremo (programsko in materialno), ki je različna in posebna za posamezne mikroprocesorje. To so t.i. emulatorji. Emulator se priključi na razvojni sistem ter omogoča vpis, testiranje in izvajanje aplikativnega programa na aplikativnem sistemu. Čeprav nas takšen razvojni sistem ne omejuje (znotraj končne množice mikroprocesorjev, ki jih podpira) pri izbiri mikroprocesorja in je v veliko primerih zelo dobra rešitev, je za testiranje programov potrebna dodatna nabava že omenjenih emulatorjev. Ti stroški (ki so lahko pomembni) se bodo ponovili pri vsaki aplikaciji z novim tipom mikroprocesorja.

v. Razvojni sistem na 'večjem' računalniku.

Večji računalnik, ki je poleg standardne opreme ustrezno opremljen z križnimi zbirniki, prevajalniki in simulatorji za različne mikroprocesorje je zelo dobra rešitev za tiste uporabnike, ki v svoji opremi že imajo večji računalnik ali lahko razpolagajo z delom kapacitet njim dosegljivega večjega računalnika (time sharing terminal,..). Pri tem odpadejo vsa večja začetna vlaganja v razvojne sisteme. Tudi pri teh sistemih lahko omenimo nekaj problemov, ki se pojavljajo . Prvi problem je pomanjkanje t.i. križnih programskih pripomočkov. Proizvajalci mikroprocesorjev namreč močneje podpirajo razvoj lastnih razvojnih sistemov za lastne (in sorodne) družine mikroprocesorjev v oblikah, ki so opisane v prejšnjih odstavkih in manj podpirajo razvoj križnih programov za večje računalnike. Za takšne programe se bolj zavzemajo proizvajalci mikroprocesorjev, ki ne ponujajo lastne razvojne sisteme ker na ta način širijo uporabnost svojih proizvodov. Iz tega izvira drugi problem in ta je, velika raznolikost v načinu in tehniki uporabe križnih programov, ki so izdelani pri različnih proizvajalcih.

Posebni problemi nastanejo, ko se uporabnik odloči za uporabo (razlogi so lahko tehnične lastnosti, dosegljivost, cena,..) mikroprocesorja ali mikroračunalnika za katerega proizvajalec ponuja zelo skromno aplikativno razvojno opremo, ali pa le-te sploh ne nudi. V tem primeru preostane samo še razvoj lastnih pripomočkov. Odločimo se lahko za eno od prej opisanih kategorij razvojnega sistema. Odločitev bo odvisna od opreme s katero razpolagamo, časa, ki je na voljo za razvoj nove opreme ter od števila razvijalcev.

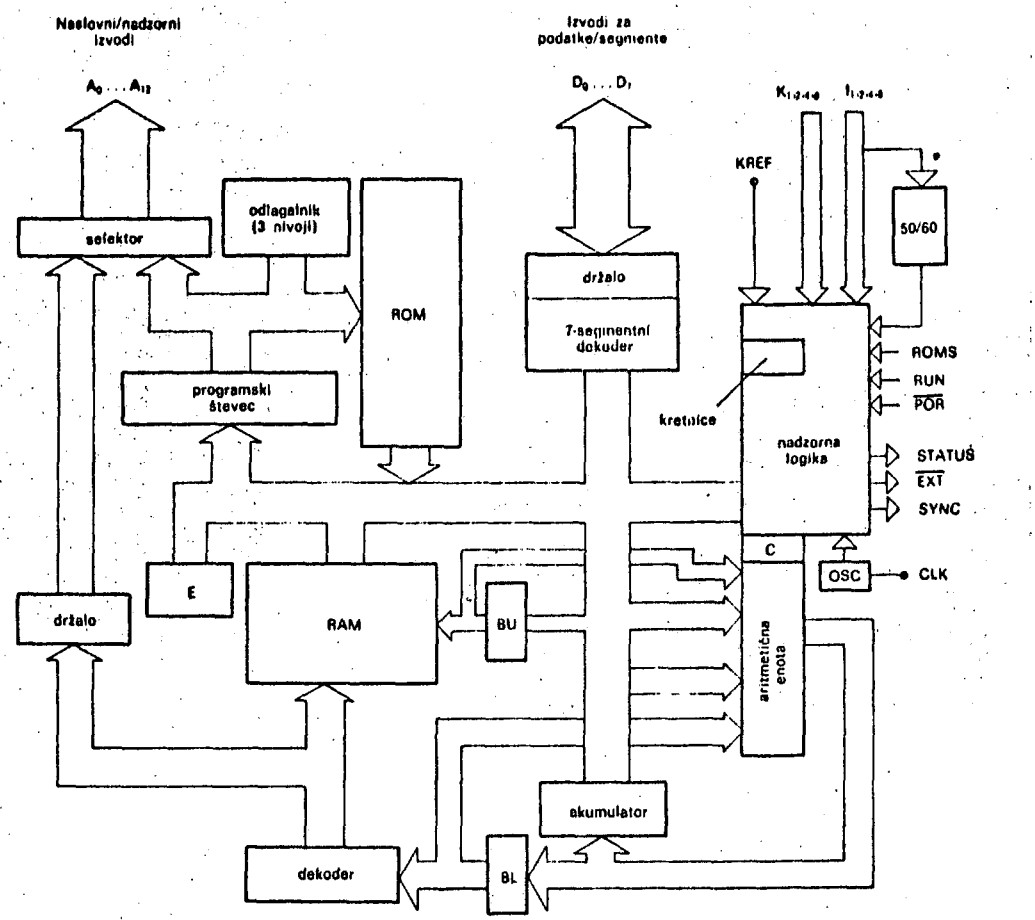
V tem prispevku bomo prikazali primer, ko se uporabnik (razvijalec aplikativnih mikroračunalniških sistemov) odloči za razvoj mikroračunalniškega sistema pri katerem je osnovni element sistema domači integrirani mikroračunalnik Iskra EMZ 1001 za katerega proizvajalec ne ponuja zadosti bogato aplikativno razvojno opremo.

## INTEGRIRANI MIKRORAČUNALNIK Iskra EMZ 1001

Mikroračunalnik Iskra EMZ 1001 [4],[5] z veliko gostoto integriranih elementov prinaša vse prednosti krmiljenja z mikroračunalnikom pri minimalnih stroških opreme. Obdelava podatkov v mikroračunalniku EMZ 1001 poteka preko 4 bitnih besed. Nadzorni del in vhodno-izhodna konstrukcija sta 8 bitno zasnovana. Povezava z zunanjim svetom poteka preko več specijaliziranih vodil. Vodilo A ima 13 bitov, ki služijo samo kot izhodi. Vodilo D je 8 bitno in prenaša podatke v dveh smereh, lahko pa je tudi v t.i. nevtralnem stanju. Vodili I in K sta 4 bitni in služita samo kot vhoda. Pomnilnik RAM je razdeljen na 4 strani, od katerih ima vsaka 16 4-bitnih besed. Pomnilnik RAM se naslavlja z registri BL in BU. Naslovni prostor programskega ROM pomnilnika je razdeljen na 8 pomnilniških bank. Vsaka banka vsebuje 1K 8-bitnih besed. Banka '0' je tisti del pomnilnika, ki je vključen na integrirani ploščici, ostale banke pa po želji dodajamo od zunaj. Omogočena je izbira aktivnega ROM pomnilnika (samo notranji, samo zunanji ali notranji in zunanji pomnilnik obenem) ter izbira načina delovanja (statični, multiplexni in testni). Na osnovi prikazanih lastnosti mikroračunalnika EMZ 1001 (slika 1) se lahko prepričamo, da je ta uporabljiv v številnih manjših aplikacijah. Najbolj smiselne so rešitve v katerih zadošča število razpoložljivih vodil za neposredno krmiljenje sistemske okolice.

## STANDARDNA APLIKACIJSKA RAZVOJNA OPREMA

Iskra Mikroelektronika, proizvajalec mikroračunalnika EMZ 1001, ponuja kot pomoč pri razvoju aplikacij s tem integriranim mikroračunalnikom t.i. statični emulator. Statični emulator izkorišča možnost multipleksnega delovanja mikroračunalnika EMZ 1001 in služi za preverjanje in izvajanje aplikativnih programov tako, kot da bi bil program že zapisan v notranjem ROM pomnilniku. Emulator vsebuje mikroračunalnik EMZ 1001, PROM pomnilnik z uporabnikovim programom, pomnilni vmesnik in 40-žilni kabel s priključkom v obliki mikroračunalnika. Za preverjanje programa spojimo priključek emulatorja s podnožjem, v katerem bo po končanem razvoju mikroračunalnik EMZ 1001 z notranjim uporabniškim programom.




Slika 1.

Poleg statičnega emulatorja, kot samostojne enote in z možnostjo priključka na mikroračunalnik Iskra Data, ponuja Iskra Mikroelektronika, na osnovi primerno podanih specifikacij pogodbeni prevzem izdelave programov in prototipov za posamezne uporabnike.

## LASTNI RAZVOJNI SISTEM

Za večje število obsežnejših aplikacij s tem mikroračunalnikom potrebuje uporabnik več programske in materialne razvojne opreme. Narekuje se razvoj lastnega razvojnega sistema. Ta naj bi čim več pripomogel k učinkovitemu in hitremu razvoju aplikacij pri čim boljši uporabi že obstoječe opreme in z minimalnimi zaščetnimi vlaganji.

Slika 2. prikazuje primer takšnega razvojnega sistema pri katerem smo uporabili že obstoječi računalnik LSI 11/23 in mikroračunalnik na bazi mikroprocesorja Intel 8080. Postopek razvoja aplikativnih programov je pri tem sistemu razdeljen na dve etapi :

- Delo na računalniku LSI 11/23, ki je poleg standardne opreme (učinkovit urejevalec teksta, prevajalniki, gibki diski, winchester disk, tiskalnik, CRT-prikazovalnik,..) opremljen še s križnim zbirnikom za zbirni jezik [2] in s programskim simulatorjem mikroračunalnika EMZ 1001. V tej etapi se aplikativni program piše, testira in popravlja, dokumentira in shranjuje. Po končani prvi fazi, se lahko pri predpostavki, da je bil problem dovolj dobro definiran, razvoj aplikativnega programa tudi konča.

- Objektni kod aplikacijskega programa, ki je dobljen na računalniku, lahko dodatno testiramo na realnem okolju s pomočjo emulatorja [1] ter ga dodatno prilagodimo morebitnim materijalnim (hardwarskim) posebnostim aplikacije.

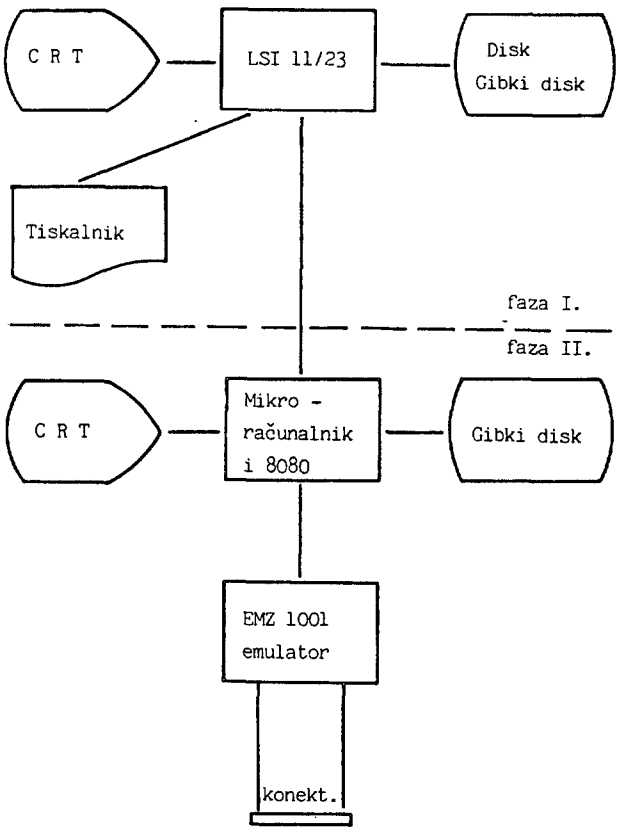
Omenjeni križni zbirnik za mikroračunalnik EMZ 1001 in emulator sta že predstavljena v IJS Delovnih poročilih [1] in [2]. V tem prispevku bomo predstavili še tretji razvojni pripomoček - programski simulator mikroračunalnika EMZ 1001.

## PROGRAMSKI SIMULATOR

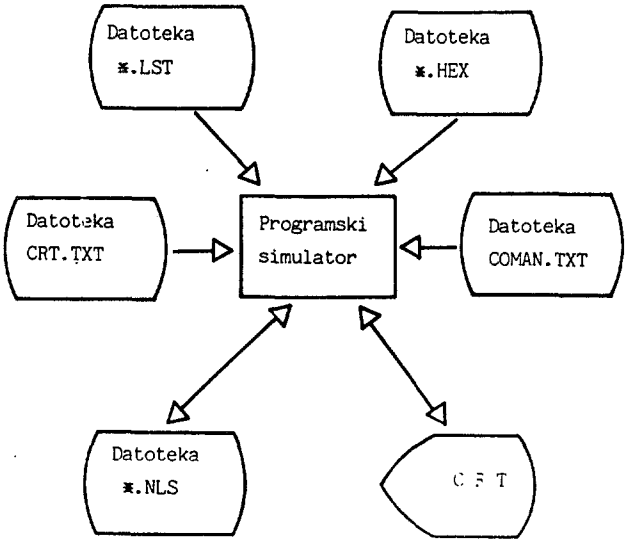
Programski simulator je programski paket, ki, v našem primeru, simulira delovanje mikroračunalnika in pri tem omogoča *uporabniku :*

- vpogled v notranjo strukturo registrov in pomnilnika mikroračunalnika pri vsakem koraku izvajanja aplikativnega programa,
- nadzor nad pretokom vhodno/izhodnih podatkov,
- prekinitev izvajanja pri poljubnih stanjih podatkovnega in adresnega vodila,
- koračno izvajanje programa
- izvajanje posameznih delov (modulov) aplikativnega programa,
- merjanje računalniškega časa,
- diagnosticiranje nekaterih napak.

Programski simulatorji so lahko napisani v kakšnem od 'splošnih' programskih jezikov, ali v posebnih simulacijskih jezikih in po za simulatorje posebno zgrajenih metodah (ISPS language [10], MicroSim [8]). Takšni posebni jeziki in sistemi omogočajo simulacijo mikroračunalniških sistemov in v nekaterih primerih tudi simulacijo multi mikroračunalniških sistemov. Simulator mikroračunalnika EMZ 1001 je napisan v programskem jeziku Pascal, teče na računalniku LSI 11/23 in je namenjen samo simulaciji izvajanja instrukcij tega mikroračunalnika.




Slika 2.

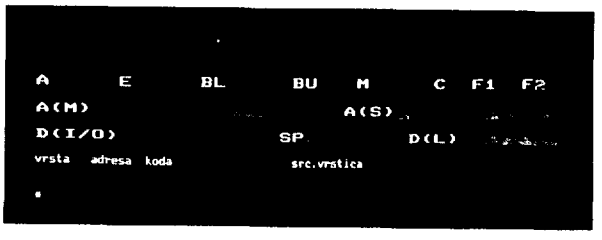



Slika 3.

Vhodni datoteki za 'programski simulator (slika 3.) sta datoteki (*.LST in *.HEX), ki jih je zgradil križni zbirnik. Pri inicijalizaciji bo simulator detoteko *.LST preuredil (odstranjene so prazne in komentarske vrstice) v datoteko z imenom *.NLS.

Datoteka CRT.TXT vsebuje podatke za izpis slike (monitorja) na zgornji polovici CRT-prikazovalnika, kot je razvidno iz slike 4. Monitor bo v času emuliranja stalno na prikazovalniku. Pri vsakem koraku simuliranja programa se vrednosti v okvirjih korigirajo, kar omogoča uporabniku stalen pregled nad vsebinami registrov, vhodno/izhodnimi vodili in računalniškim časom.

Za učinkovito uporabo simulatorja je predpisan enostaven vhodni jezik s katerim uporabnik nadzoruje delovanje simulatorja, vpisuje prekinitvene točke, kontrolira izpisovanje na CRT in podobno. V Tabeli 1. je prikazan nabor ukazov vhodnega jezika s krajšimi pojasnili pomenov le-teh. Podoben tekst kot v Tabeli 1. vsebuje datoteka COMAN.TXT, ki jo uporabnik dobi izpisano na CRT ko uporabi ukaz 'HELP




Slika 4.

Ukazi EMZ-simulatorja :

a) Kontrola delovanja mR EMZ :

HA [LT]
ustavitev delovanja mR

RU [N] [aaaaa]
izvajanje programa pri osmiški adresi 'aaaaa', če adresa ni podana, se začne izvajanje pri adresi 0

RE [SE]
reset mR (inicijalizacija)

b) Testne prekinitve izvajanja :

BAA aaaaa
T.p. pri osmiški adresi 'aaaaa'

BAD aaaaa
T.p. pri osmiškem podatku 'aaaaa' na A-linijah

BDI ddd[O][D][H]
T.p. pri instrukciji 'ddd'

BDD ddd[O][D][H]
T.p. pri V/I podatku 'ddd'

BDL ddd[O][D][H]
T.p. pri zadržanem podatku 'ddd'

S [TEP] [ddd[O][D][H]]
T.p. po 'ddd' korakih (instrukcijah), če število korakov ni podano, izvede 1 korak, nadaljnje enokoračno izvajanje sproži tipka 'presledek'

NS
prepoved koračnega izvajanja programa

c) Kontrola izpisa na CRT :

TR
izpisovanje tekočih vrstic programa

NT
brez ispisovanja tekočih vrstic programa

M [EMO] [ddd[O][D][H]]
izpis 'ddd' vrstice ali celotne vsebine notranjega RAM-pomnilnika mR

TI
resetiranje števca časa

d) Pomožni ukazi :

RA [M]
vpis novih vrednosti v notranji RAM-pomnilnik mR

NE [W]
vpis novih vrednosti v notranje registre mR

CO [MA]
izpis vseh trenutno aktivnih ukazov

CL [EA]
briše vse ukaze (začetno stanje simulatorja)

HE [LP]
izpis liste vseh ukazov za EMZ-simulator

O [UT]
izhod iz EMZ-simulatorja

Tabela 1.

## LASTNOSTI PROGRAMSKEGA SIMULATORJA MIKRORAČUNALNIKA EMZ 1001

### 1. Pregled nad notranjimi registri

Kot je razvidno iz slike 4, so na CRT prikazovalniku stalno (v binarnem zapisu) prikazane vrednosti vseh notranjih registrov in sicer :

- akumulatorja (A)
- splošnega registra (E)
- naslovnih registrov (BL) in (BU)
- kretnic (flags) (C),(F1) in (F2)
- trenutno naslovljene RAM besede (M)
- globine vgnezdenosti podprogramov (stack pointer) (SP)

Nadalje lahko na monitorju opazujemo stanje podatkovnih vodil:

- Vodilo A.
  Prikazani sta vrednosti v t.i. 'slave' (A(S)) in 'master' (A(M)) 13-bitnih registrih. Pomožni (slave) register služi za pripravo izhodnih podatkov, glavni (master) register pa zadržuje podatke, ki so prisotni na zunanjem A-vodilu.

- Vodilo D.
  Prikazani sta vrednosti podatkovnega vodila pri prehodu vhodnih in izhodnih podatkov med mikroračunalnikom in okolico (D(I/O)) ter vrednost zadržanih (latched) podatkov na D-vodilu (D(L)).

Poleg pregledovanja stanja registrov in vodil, lahko s pomočjo ukaza 'NEW' vsem prikazanim elementom priredimo poljubno novo vrednost. Ta možnost se bo pokazala za zelo uporabno pri pripravi začetnih pogojev za izvajanje posameznih podprogramov.

### 2. Pregled nad RAM pomnilnikom

Omenjeno je že, da se vrednost trenutno naslovljene besede RAM pomnilnika stalno prikazuje na 'monitorju'. Pregled celotnega ali poljubnega dela RAM pomnilnika je omogočen z ukazom 'MEMO'. Primer izpisa vseh besed RAM pomnilnika je prikazan na sliki 5. Tako kot pri registrih, uporabnik lahko tudi v RAM pomnilniku spreminja vrednosti na posameznih lokacijah s pomočjo ukaza 'RAM'.

### 3. Nadzor nad vhodnimi podatki

Pri programskem simuliranju ni omogočen priključek posebnih zunanjih vhodno/izhodnih enot, kot je načrtano za realen sistem, zato bo simulator, ko pričakuje vhoden podatek prepustil uporabniku odločitev o vrednosti le-tega. Takšni primeri se pojavijo, ko se simulira EMZ instrukcija 'INP' in ko simulirani program pregleduje K in I vhode mikroračunalnika. Možnost podajanja poljubnih vhodnih stanj, omogoča uporabniku učinkovitejše testiranje tistega dela aplikacijskega programa, ki obdeluje vhodne podatke.

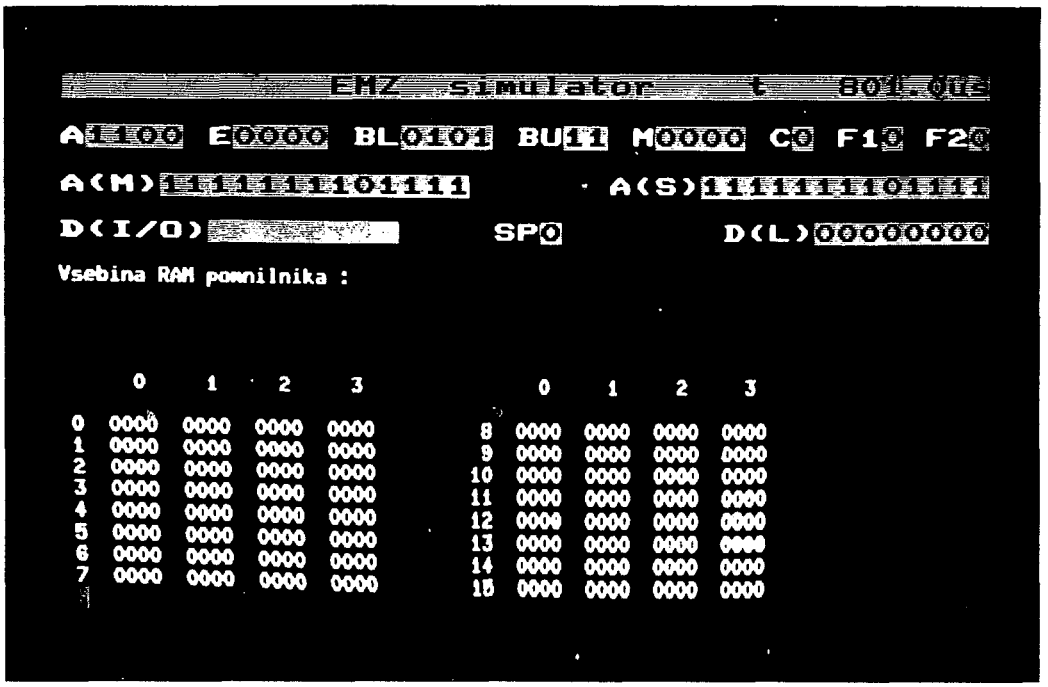
### 4. Testne prekinitve

Iz Tabele 1. je razvidno, da je testna prekinitev omogočena pri poljubni vrednosti na A in D vodilih, in sicer :

- pri instrukciji na poljubni adresi (ukaz 'BAA'),
- pri poljubnem podatku na zunanjem A-vodilu (ukaz 'BAD'),
- pri poljubni instrukciji (ukaz 'BDI'),
- pri poljubnem vhodno/izhodnem podatku (ukaz 'BDD'),
- pri poljubni vrednosti zadržanega podatka na D-vodilu (ukaz 'BDL').

## 6. Računalniški čas

Na CRT prikazovalniku v t.i. monitorju poleg zgoraj opisanih podatkov uporabnik lahko stalno opazuje čas (v μ sek.) izvajanja simuliranega programa (slike 4,5,6). Računalniški čas se resetira z ukazom 'TZ'. Na ta način lahko uporabnik dobi podatke o času izvajanja celotnega ali posameznih delov (podprogramov) simuliranega programa, poišče najkrajši in najdaljši čas izvajanja posameznih ciklov znotraj programa in podobno.




Slika 5.

Pri vsaki testni prekinitvi se na CRT-ju izpiše vrstica izvirnega programa (source line) pri kateri se je zgodila prekinitev. Na ta način se potek simulacije programa lažje spremlja tudi na 'listingu' aplikacijskega programa. Posebno kritične dele programa lahko uporabnik simulira po korakih s pomočjo ukaza 'STEP'. Isti ukaz omogoča testno prekinitev po poljubnem številu izvedenih instrukcij simuliranega programa.

## 5. Simulacija programskih modulov

Ukaz 'RUN' (Tabela 1.) omogoča nastavitev poljubne vrednosti programskega števca (znotraj naslovnega polja ROM pomnilnika) kar omogoča simuliranje posameznih delov simuliranega programa. V primeru simuliranja posameznega programskega modula (podprograma) je običajno potrebno nastaviti nekatere začetne pogoje kot so: začetne vrednosti v določenih registrih in na določenih lokacijah RAM pomnilnika, vrednost števca globine vgnezdenosti modula in računalniški čas. Na ta način lahko opazujemo (testiramo) delovanje posameznih modulov aplikacijskega programa, ki jih v končni fazi razvoja združimo v skupni program.

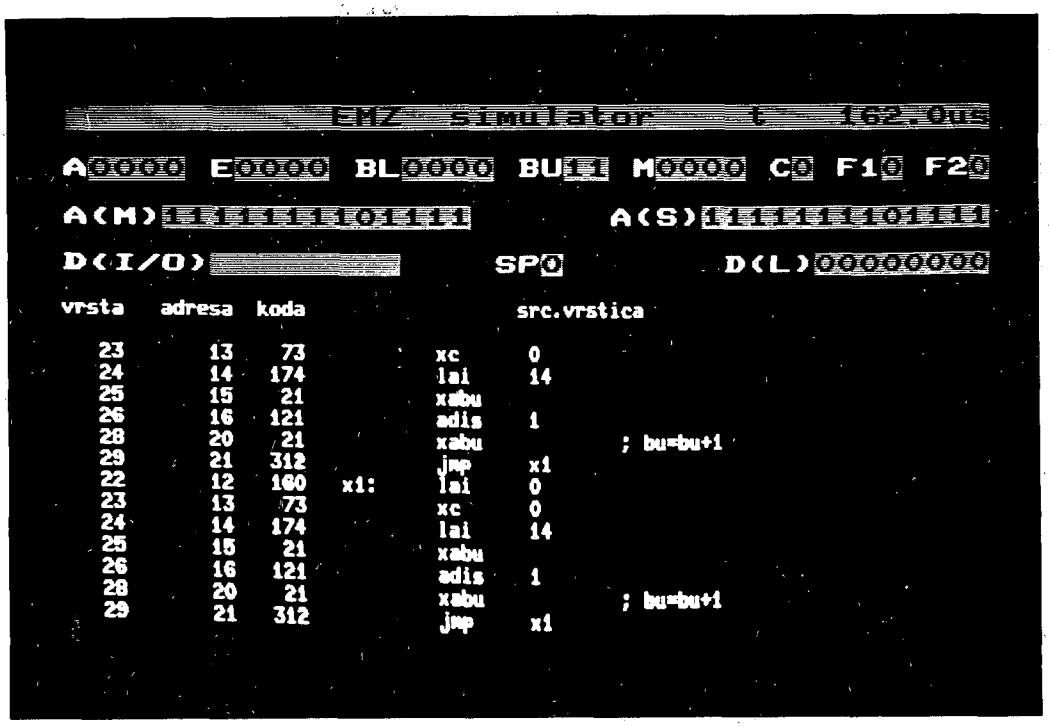
## 7. Diagnostika

Simulator opozarja uporabnika (z izpisom na CRT) na odvečne 'PP'- instrukcije in na preskakovanje več, v zaporedju programiranih 'LAI' in 'LB_' instrukcij. Simulacija se ustavi, če se program začne z 'LAI' ali 'LB' instrukcijo, če pride do prekoračitve globine vgnezdenja podprogramov ( največja globina je 3) in če je prekoračeno naslovno področje ROM pomnilnika.

Simulacija se ustavi tudi, če pri simuliranju 'INP' instrukcije D-vodilo ni bilo predhodno nastavljeno v nevtralno (Tri-state) stanje in če pri odčitavanju I ali K vhodov le-ti niso bili predhodno pravilno naslovljeni.

Pri vseh opozorilih in vstavitvah se na CRT izpiše kratek opis napake in trenutna vrednost programskega števca, pri kateri je ugotovljena napaka.

Poleg opisanih možnosti simulatorja, zaradi lažjega sledenje simulacije, lahko uporabnik z ukazi 'TR' in 'NT' (trace / notrace) kontrolira izpisovanje vrstic simuliranega programa. Primer takšnega izpisa lahko vidimo na sliki 6.




Slika 6.

## ZAKLJUČEK

Razvojni sistem, ki je predstavljen v tem prispevku je prestal začetna testiranja, popravke in izboljšave ter pokazal pozitivne rezultate. Trud in čas, ki sta bila vložena v izdelavo opisane programske in materialne opreme se izplačata že po nekaj aplikacijah. Čas razvoja aplikativnih programov se občutno zmanjša na račun iskanja boljših tehničnih in programerskih rešitev. Ugotovili smo že, da je sedanja rešitev posredovanja vhodnih podatkov, ko le-te vpisuje uporabnik preko CRT-ja neprimerna pri aplikacijah z velikim številom vhodnih podatkov. V takšnih primerih je boljše če bi vhodne podatke imeli vnaprej pripravljene in shranjena na posebni vhodni datoteki. Nadaljnja uporaba razvojnega sistema bo z novimi aplikacijami verjetno prinesla zahteve po dodatnih izboljšavah tako materijalne kot programske opreme.

## LITERATURA

[1] Materialna in programska podpora za razvoj prototipov z integriranim mikroračunalnikom Iskra EMZ 1001 ;
D.Miljan, P.Reinhardt, P.Kolbezen ;
IJS Delovno poročilo Dp-2407,
Ljubljana, december 1981.

[2] Križni zbirnik na bazi makro ekspanzije za mikroračunalnik EMZ 1001 ;
P.Reihardt, R.Reinhardt ;
IJS Delovno poročilo Dp-2406,
Ljubljana, december 1981.

[3] Projektiranje z integriranimi računalniki ;
Davor Miljan ;
Časopis Informatica 4/1980, str.29-35.

[4] Mikroračunalnik EMZ 1001,
katalog Iskra - Industrija elementov za elektroniko - Mikroelektronika,
Izdala: Iskra Commerce 09. 78.

[5] Integrirani mikroračunalnik Iskra EMZ 1001 ;
Dušan Raič ;
Časopis Informatica 1/1979, str.12-23.

[6] Projektiranje z integriranimi mikroračunalniki ;
Davor Miljan ;
IJS Delovno poročilo Dp-2146,
Ljubljana, december 1980.

[7] Microprocessors and software design tools;
Keith D. Baker ;
Microprocessors and microsystems,
Vol 3, No.2, March 79, page 87 - 93.

[8] MicroSim - a new approach to program development ;
David Cosserat ;
Microprocessors and microsystems,
Vol 3, No.2, March 79, page 95 - 98.

[9] Designing with single chip microcomputers
Markus Moser ;
Microprocessors and microsystems,
Vol 3, No.3, April 79, page 135 - 139.

[10] New generation of microsystem simulators
F.W van Linden ;
Microprocessors and microsystems,
Vol 4, No.1, jan/feb 80, page 5 - 9.