

## Rešenja zadataka - Januar 2007

### Zadatak 1

Pretpostaviti da se program P izvršava na dve različite implementacije mašine, M1 i M2, čija je taktna frekvencija 500 MHz. Skup instrukcija koje se izvršavaju na mašini M1, kao i procenat učešća i broj taktnih intervala za svaku od instrukcija dat je u sledećoj tabeli:

klasa instrukcije	učestanost (%)	broj taktnih intervala
aritmetičke	40	2
logičke	35	3
FP sabiranje	15	8
FP množenje	5	12
Branch	5	10

Mašina M2 ne poseduje hardver za izvršenje operacija FP sabiranje i FP množenje pa se one moraju emulirati korišćenjem aritmetičkih i logičkih operacija, na sledeći način:

FP sabiranje : 10 aritmetičkih i 5 logičkih operacija,

FP množenje : 13 aritmetičkih i 7 logičkih operacija.

a) Ako broj instrukcija programa P koji se izvršava na mašini M1 iznosi  $100 \cdot 10^6$ , odrediti koliko je instrukcija potrebno da bi se isti program izvršio na mašini M2.

b) Za broj instrukcija određen u stavki a) odrediti vreme izvršenja programa na mašinama M1 i M2.

### Odgovor:

a)

klasa instrukcije	br. instr. za M1 ( $\cdot 10^6$ )	br. instr. za M2 ( $\cdot 10^6$ )	učestanost (%) za M2
aritmetičke	40	40 (+150+65)	63
logičke	35	35 (+75+35)	35.8
FP sabiranje	15	( 15*(10a+5l)=150a+75l )	-
FP množenje	5	( 5*(13a+7l)=65a+35l )	-
Branch	5	5	1.2
ukupno:	100	405	100

b)

$$CPIM1 = 0.4 \cdot 2 + 0.35 \cdot 3 + 0.15 \cdot 8 + 0.05 \cdot 12 + 0.05 \cdot 10 = 4.15$$

$$CPIM2 = 0.63 \cdot 2 + 0.358 \cdot 3 + 0.012 \cdot 10 = 2.454$$

$$TM1 = NI \cdot CPI \cdot t = 100 \cdot 10^6 \cdot 4.15 \cdot 0.002 \cdot 10^{-6} = 0.83 \text{ s}$$

$$TM2 = NI \cdot CPI \cdot t = 405 \cdot 10^6 \cdot 2.454 \cdot 0.002 \cdot 10^{-6} = 1.99 \text{ s}$$

### Zadatak 2

Napisati program na asemblerskom jeziku mikroprocesora iz familje x86 koji generiše jediničnu matricu. Dimenzija matrice je smeštena na lokaciji DIM (izabрати vrednost u opsegu 1-10), a elemente matrice, veličine bajt, smeštati u memoriju po vrstama počev od adrese MAT. Nakon generisanja sve elemente matrice treba prikazati na ekranu onim redosledom kojim su smeštani u memoriju i pri tome ih odvajati zarezom.

*Napomena:* 1) Za ispisivanje jednog karaktera, smeštenog u registru DL, koristiti poziv 02h interapta 21h; 2)

Za ispisivanje niza znakova, čija je početna adresa smeštena u registru DX, koristiti poziv 09h interapta 21h;

3) Za povratak u DOS koristiti funkcijski poziv 4Ch interapta 21h; 4) ASCII kôd znaka "," je 2Ch.

### Odgovor:

```
;------  
;Program za kreiranje jedinične matrice dimenzije MXM  
;------
```

\*\*\* Glavni program \*\*\*

```
TITLE          GLAVNI PROGRAM
PAGE           60,80
SMAG           SEGMENT
               DB      64  DUP (?)
SMAG           ENDS
SPOD           SEGMENT
DIM  DB       10
MAT  DB       ?
SPOD           ENDS
GLPROG        SEGMENT
MAIN          PROC  FAR
               ASSUME      CS:GLPROG,DS:SPOD,SS:SMAG
               MOV  AX, SPOD
               MOV  DS, AX
               MOV  BX, OFFSET MAT
               MOV  CH, DIM      ;brojac za vrste
N_VRSTA:      MOV  CL, DIM      ;brojac za kolone
PONOVI:      CMP  CH, CL
               JZ   JEDNAKO    ;granaj se ako je jednako (tu su 1)
               MOV  AL, 00H    ;pripremi nulu
               JMP  DALJE
JEDNAKO:     MOV  AL, 01H      ;pripremi jedinicu
DALJE:      MOV  [BX], AL
               INC  BX          ;BX=BX+1
               DEC  CL          ;smanji brojca za kolone
               JNZ  PONOVI     ;ako nije nula, vrati se na ponovi
               DEC  CH          ;smanji brojca za kolone
               JNZ  N_VRSTA    ;ako nije nula, vrati se na ponovi
               MOV  AL, DIM
               MOV  CL, AL
               MUL  CL
               SUB  BX, AX
               MOV  CX, AX
PRIKAZ:     MOV  DL, [BX]
               ADD  DL, 30H    ;pretvori u ASCII da moze da se prikaze
               MOV  AH, 02H
               INT  21H
               MOV  DL, " , " ;stampa zarez
               INT  21H
               INC  BX
               LOOP PRIKAZ
               MOV  AH, 4Ch
               INT  21h
MAIN          ENDP
GLPROG        ENDS
               END  MAIN
```

### Zadatak 3

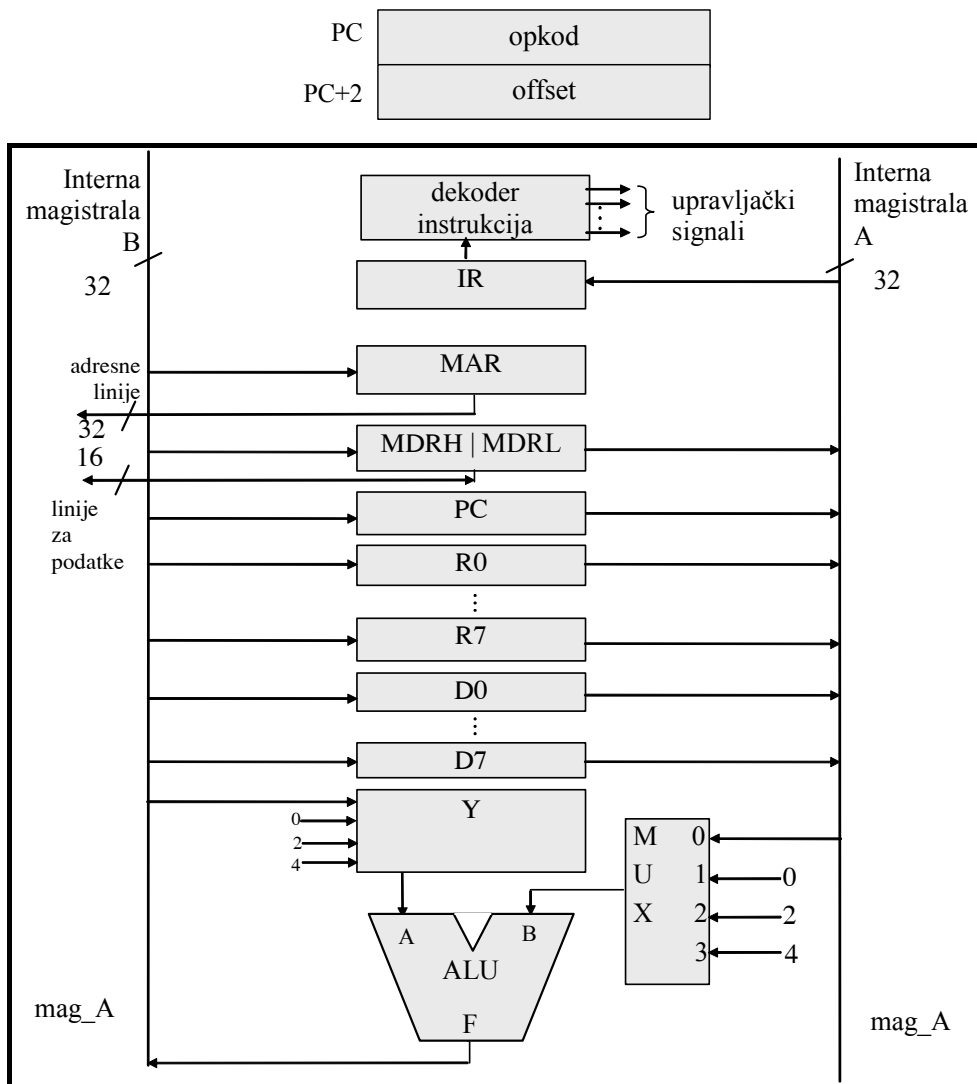
Interna organizacija 32-bitnog CPU-a prikazana je na slici 1. Svi interni registri i putevi podataka su obima 32-bita, dok je magistrala podataka kojom se CPU povezuje sa memorijom 16-bitna. Postoje dve operacije čitanja RDH i RDL, koje čitaju podatke iz memorije i pune ih u MDRH i MDRL, respektivno. Bilo koji od registara u istom upravljačkom koraku (mikrooperaciji) može se pojaviti kao izvorište i kao odredište. Za strukturu staze podataka prikazanu na slici 1 napisati upravljačku sekvencu koja je potrebna za implementaciju sledeće CPU-ove instrukcije

LDINC Rs,offset(Rt)

*Napomena:*

Set  $Y_i \equiv i \rightarrow Y$  ( $i=0,2,4$ );  $Y_{in} \equiv \text{Mag}_B \rightarrow Y$ ;  $\text{MUX}_i = \text{signal sa ulaza } i \text{ ide na izlaz MUX-a } (i=0,1,2,3)$ .

Implementacija 32-bitne instrukcije LDINC  $R_s, \text{offset}(R_t)$  ekvivalentna je sledećoj aktivnosti: U registar  $R_s$  (bitovi 20-16) smešta se podatak sa adrese dobijene sabiranjem znakovno proširenog offset-a (bitovi 15-0) i vrednosti u registru  $R_t$  (bitovi 25-21), a nakon toga vrednost registra  $R_t$  se inkrementira za 4. Drugim rečima:  $R_s = M(\text{offset} + R_t)$ ,  $R_t = R_t + 4$ . Format instrukcije je sledeći:



Slika 1. Interna organizacija 32-bitnog CPU-a oraganizovana oko dve magistrale

### Odgovor 3:

korak	akcije	komentar
1.	PCout, MUX <sub>0</sub> , Set Y <sub>0</sub> , Yout, F=A+B, Yin, MARin, Read <sub>L</sub>	; pribavljanje opkôda instrukcije
2.	Yout, MUX <sub>2</sub> , F=A+B, Yin, PCin, Wait for Ready	; (PC)=(PC)+2
3.	MDRLout, IRin, MUX <sub>1</sub> , Yout, F=A+B, MARin, Read <sub>H</sub>	
4.	MUX <sub>2</sub> , Yout, F=A+B, PCin, Wait for Ready	; (PC)=(PC)+2
5.	MDRHout, MUX <sub>0</sub> , Set Y <sub>0</sub> , Yout, F=A+B, Yin	; neposredni operand smešta se u Y
6.	Rtout, MUX <sub>0</sub> , Yout, F=A+B, MARin, Read	
7.	Rtout, MUX <sub>0</sub> , Set Y <sub>4</sub> , Yout, F=A+B, Rtin, Wait for Ready	
8.	MDRout, MUX <sub>0</sub> , Set Y <sub>0</sub> , Yout, F=A+B, Rs, End	; M(Rt+offset)→Rs

## Zadatak 4

Razmotriti sledeću kodnu sekvencu:

```

0: ld [r1] → r5
1: add r5, 1 → r5
2: ld [r1+4] → r6
3: cmplt r5, r6 → r2
4: bne r2, 0, instruction#7
5: sub r5, 1 → r5
6: ld [r1+8] → r6
7: st r6 → [r1]
    
```

*Napomena:* Instrukcija `cmplt` je ekvivalentna instrukciji `slt`.

a) Identifikovati sve zavisnosti po podacima (RAW, WAR i WAW). Za svaki tip zavisnosti navesti instrukcije između kojih postoji zavisnost kao i registar koji uzrokuje tu zavisnost.

b) Pretpostaviti da se data sekvenca izvršava na petostepenom protočnom procesoru sa sledećim stepenima: F - pribavljanje instrukcije, D - dekodiranje instrukcije, X - pribavljanje operanada, izvršenje operacije i određivanje uslova grananja, M - pristup memoriji, W - upis rezultata u određeni registar. Takođe pretpostaviti da je izvedeno potpuno premošćavanje. To znači da izvršenje svake instrukcije traje jedan ciklus, dok je instrukciji `ld` potreban još jedan ciklus zbog pristupa memoriji. Pretpostaviti da do grananja nije došlo. Nacrtati protočni dijagram izvršenja sekvence na datom procesoru.

c) Pretpostaviti da je za pristup kešu podataka potrebno dva ciklusa i da procesor ima šest protočnih stepena zato što je, u odnosu na slučaj b), za pristup kešu podataka potrebno dva ciklusa (M1 i M2). Pod uslovom da važe sve pretpostavke date pod b), nacrtati protočni dijagram izvršenja sekvence na ovom procesoru.

## Odgovor

a)

RAW: #0→#1 (r5), #0→#3 (r5), #0→#5 (r5), #1→#3 (r5), #1→#5 (r5), #2→#3 (r6), #2→#7 (r6), #3→#4 (r2), #6→#7 (r6)

WAW: #0→#1 (r5), #0→#5 (r5), #1→#5 (r5), #2→#6 (r6)

WAR: #1→#5 (r5), #3→#5 (r5), #3→#6 (r6)

b)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ld [r1] → r5	F	D	X	M	W												
add r5, 1 → r5		F	D	*	X	M	W										
ld [r1+4] → r6			F	*	D	X	M	W									
cmplt r5, r6 → r2					F	D	*	X	M	W							
bne r2, 0, instr#7						F	*	D	X	M	W						
sub r5, 1 → r5								F	*	F	D	X	M	W			
ld [r1+8] → r6											F	D	X	M	W		
st r6 → [r1]												F	D	*	X	M	W

c)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
ld [r1] → r5	F	D	X	M1	M2	W															
add r5, 1 → r5		F	D	*	*	X	M1	M2	W												
ld [r1+4] → r6			F	*	*	D	X	M1	M2	W											
cmplt r5, r6 → r2						F	D	*	*	X	M1	M2	W								
bne r2, 0, instr#7							F	*	*	D	X	M1	M2	W							
sub r5, 1 → r5										F	*	F	D	X	M1	M2	W				
ld [r1+8] → r6													F	D	X	M1	M2	W			
st r6 → [r1]														F	D	*	*	X	M1	M2	W

