

BLICEVI I ZADACI
ZA
1. LABORATORIJSKU VJEŽBU

sve je kopirano iz prošlogodišnjih tema sa FER2, god 2012/2013 nadalje
rješenja su također kopirana

PITANJA IZ BLICEVA

- 1) Gdje se spremaju mikroprogrami?
- U posebnoj memoriji u upravljačkoj jedinici (tako nešto slično je bilo ponuđeno)
 - 2) Ako neka makroinstrukcija za izvođenje koristi mikroprogram sa tri mikroinstrukcije koliko onda zauzima mjesta u glavnoj memoriji?
- Tu je mislim bilo ponuđeno 2B, 3B, 4B, i $3 \cdot 32b$, nisam siguran kaj je točno pa nisam zaokružio
 - 3) Ako želimo izbrisati sadržaj nekog registra (postaviti na 0), koristi ćemo mikroinstrukciju:
- Ne sjećam se točno instrukcija, no točna je ona koja sprema dva puta iste podatke u registar (npr. $r1 \leftarrow 6$, $r2 \leftarrow 6$) i zatim se na ALU odabere XOR i spremi u isti registar
 - 4) Statusni registar se u navedenom modelu nalazi:
- U jednom od registara od $r0$ do $r7$ (inače je u $r6$, ali nije bilo ponuđeno konkretno)
 - 5) U našem modelu, IR se sastoji od:
- 16 bita, od kojih prvih 6 bita ide operacijski kod (mislim da je to točno jer se MAR i MDR tretiraju posebno, ispravite me ako sam u krivu)
- Ostale su mi bile aritmetičke naredbe (tipa ako je u ovom registru spremljeno to, a u drugom ono, uz ove naredbe rezultat će bit bla bla) ... tak da se ne sjećam najbolje ..
- 6) U koliko se najmanje mikroinstrukcija može ostvariti JMP(addr)?
- Ja sam stavio 1 jer je se može samo staviti adresa u PC, možda sam u krivu no mislim da se može tak. Bilo je još ponuđeno 2,3,4 ...
 - 7) Ako se u svim registrima nalazi broj 7, i zatim izvede naredba XOR nad $r6$ i $r7$, te se rezultati sprema u $r1$ i $r2$, što će biti nakon toga u registrima? (Nije bilo baš tako zadano već mikroinstrukcijama, no poanta je ista)
- Bit će: $r1=0$, $r2=0$, $r6=7$, $r7=7$

9) Što radi sljedeći kod:

Makroinstrukcija: 000001 00 01 10 0000

Mikroinstrukcija: opcode[1]: rj_sel, rk_sel, a_sel=4, b_sel=3, alu_sel=ADD, r0_write, goto fetch0; ?
 $r1 + r2 \rightarrow r0$

10) Ako se mikro instrukcija sastoji od 5 instrukcija koliko će bita memorije trošiti u računalu?

a) $16 \cdot 5$

b) 32

c) 5

d) 2

e) 16

(dunno?)

11) Koliko bitova ima IR?

(malo sam se preduhitrio ipak je 16)

12) S koliko bitova je definirana instrukcija (nešto u tom stilu)

6

13) Kako je riješen PC u simulatoru?

Simulator nema PC, koristimo R7 kao PC

14) Od ponuđenih funkcija koja koristi samo 1 registar

SUBA

15) Neka mikroinstrukcija, nemogu se sjetiti, kako izgleda stanje registara nakon što se ona izvede, ako je u svim registrima u početku pisalo (neki broj).

16) Sto se tice onog zadatka ovako je isao:

U svim opcim registrima upisan je broj 7. Nakon slijedece mikroinstrukcije sta ce se desiti?

a_sel=3, b_sel=4, alu_sel=ADDA, r1_write, r2_write;

S tim da ovdje gledaju da su opci registri R1-R4.

Uglavnom odgovor je: R1=7 R2=7 R3=7 R4=7

17) makroinstrukcija zauzima 16 bita ili 2 bajta u glavnoj memoriji, nebitno je koliko ima mikroinstrukcija jer se to ne sprema u glavnu memoriju

18) U svim registrima r0-r3 se nalazi vrijednost 7. Propuštaju se r3 i r4 na ALU, a ko se izvede SUBA i mikroinstrukcijske naredbe r0_write i r1_write, što će bit u registrima?

R: r0=6, r1 = 6, r2=7, r3=7

19) S obzirom na veličinu MDR-a i MAR-a, od koliko adresa je građena memorija? (nešto u tom smislu)


R: Nisam siguran ali ako u IR možemo spremit adresu kao konstantu u donjih 8 bitova a MDR i MAR su veličine 8 bitova, onda mislim da je 2^8 8-bitnih adresa = 256 8-bitnih adresa

20) U ovom simulatoru se ne može obaviti koja ALU operacija?

R: NOR

21) Gdje (ili kako) je zadan početak koda (prva instrukcija)?

Odgovori: 1.) Jednoznačno u PCu 2.) Operacijom ADD nad `adress_true` i `adress_false` 3.) U makroprogramu 4.)....

R: Nisam siguran 

22) Kakve konstante mogu biti u ovom simulatoru? (nešto takvo)

R: 8-bitne ili 4-bitne s predznačnim proširenjem

24) Ako želimo izvesti operaciju PUSH, koristit ćemo sljedeće mikroistrukcije:

R: Tu je malo navlakuša koliko sam skužio jer imaju odgovori gdje se na ALU pušta `a_sel = 6` ili `a_sel = SP`. `r6` je SR ali koliko ja znam `a_sel` se ne može zadati kao `a_sel = SP` nego samo broj pa onda ovdje kao SP koristimo registar `r6` a ne `r5` kako bi trebalo bit. I onda ima još SUBA ili SUBA i `c_in`, onda bi trebala bit ova operacija koja nema aktivan `c_in` jer SUBA radi sljedeće: $a_bus - 1 + c_in$ i sad ako aktiviramo `c_in` rezultat će ostati isti ($a_bus - 1 + 1 = a_bus$) pa nećemo smanjiti kazaljku stoga.

25) Zadatak ko i prošle godine, aktivni i `a_sel` i `b_sel` i `rj_sel` i `rk_sel`, sprema se u `r0` s `r0_write` i ima zadan makrokod. `rj_sel` i `rk_sel` imaju veći prioritet pa nema veze što su aktivni i `a_sel` i `b_sel`, gleda se po makrokodu koja 2 registra se zbrajaju i spremaju u `r0`

26) Jedna makroistrukcija zadana je s 5 mikroistrukcija, koliko zauzima mjesta u memoriji?

R: Trebalo bi bit 2B, mikroistrukcije imaju svoju memoriju a svaka makroistrukcija se sastoji od 16 bitova = 2B

27) s obzirom na veličinu MAR-a btw. naš ima 8 bitova, kolika je MAX veličina memorije, ja mislim da je 256 8-bitnih riječi

28) u svim općim registrima su zapisane sedmice (7) ako mikro instrukcija glasi `a_sel = 3, b_sel = 4, alu_sel = SUB, r0_write, r1_write`; stanja registra? mislim da je `r0=0, r1=0, r2=7, r3=7` je bilo jedno od ponuđenih

29) koliko najmanje mikroistrukcija treba strojnoj instrukciji JMP addr mislim da je 1

30) gdje su spremljene mikroistrukcije? u posebnu upravljačku memoriju

31) kako se znamo koliko traje čitanje i pisanje iz memorije mislim da je prema wait signalu, da nije unaprijed određeno, ali nisam siguran

32) nešto tipa ako je zadana makroinstrukcija tipa 100010 01 10 11 0000
i mikro instrukcije: rj_sel, rk_sel, ri_sel, alu_sel=ADD; što se događa:
r1 <- r2 + r3

ili je bilo sa a_sel = 3, b_sel = 4 ne znam više... ugl. nešto s tim određivanjem registara...

33) koliko je velika makroinstrukcija ako postoji 5 mikroinstrukcija tako nešto, ugl. odgovor je 2B (16 bitova) jer je makronaredba uvijek te veličine

34) gdje je operacijski kod u makronaredbi, nešto tipa to. uglavnom u najviših 6 bitova je op kod... u pripremi piše točno da je ostalo...

ZADACI NA LABOSU

1) Za napisati nam je došlo LOAD ri, (rj)
S time da u r0 je zapisan broj 50 i na 50oj lokaciji je neki broj.
I sada nisam zihér ali mislim da je zadano da napišeš LOAD r1 r0 (možda nei drugi registri)
Aha, i nemorate pisati fetch, pretpostavlja se da je napisan...

rješenje:

.ucode

// ===== DIO OPERACIJSKIH KODOVA =====

// 1) LOAD ri, (rj)

opcode[1]: a_sel=4, b_sel=4, alu_sel=XOR, r4_write, goto opcode1.1;

// ===== DIO EKSTENZIJE =====

// LOAD ri, (rj)

opcode1.1: rj_sel, b_sel=4, alu_sel=OR, mar_sel=LOAD, goto opcode1.2;

opcode1.2: ir0_sel=LOAD, read, if wait then goto opcode1.2 else goto opcode1.3 endif;

opcode1.3: result_sel=IR_CONST8, ri_sel, goto fetch0;

.mem

0: 000001 01

1: 00 00 0000

2) zadatak napisati instrukciju STZR ri, postaviti sadržaj ri na 0

rješenje:

Ako moras smjestit 0 u ri jednostavno se napise:

opcode[1]: a_sel=4, b_sel=4, alu_sel=XOR, ri_sel;

i zapisat ce se 0 u ri...

3) napisati mikroprogram za makroinstrukciju: asl rj (aritmeticki posmak sadrzaja registra rj u lijevo za 1 mjesto). neka je operacijski kod instrukcije asl jednak 000100 pretpostavite da je mikroprogram za fazu pribavi vec definiran a da prva mikroinstrukcija pocinje sa fetch0. uloge

registara su- r7 pc, r6 sr, r5 sp, r4 pomocni reg.

Moguće rješenje:

```
opcode[1]: a_sel=4, b_sel=4, alu_sel=XOR, r4_write, goto opcode1.1; //u r4=0
```

```
opcode1.1: a_sel=4, c_in, alu_sel=ADDA, r4_write; //u r4 = 00000001;
```

```
opcode1.2: rj_sel, b_sel=4, alu_sel=AND, r4_write; //spremio najnizi bit od rj u r4
```

```
opcode1.3: rj_sel, b_sel=4, alu_sel=ADD, r4_write; //zbrojio najnizi bit sa rj
```

```
opcode1.4: rj_sel, b_sel=4, alu_sel=ADD, ri_sel; //jos jednom zbrojio, ali ovaj put spremio u ri
```

4) u ri zapisati sumu rj i sadržaj mem lok (rk + konst)

5) nama je bilo ostvariti $ri \leftarrow rj + \text{mem}(\text{konst} + rk)$

6) zadatak je bio aritmetički posmak u lijevo, ASL ri,rj

7) ROTL ri, rj

rotiraj bitove iz registra rj i rezultat spremi u ri.

r7 - PC, r6 - SR, r5 - SP, r4 - pomocni. ne treba postavljati zastavice za rezultat. napiši mikro i makro (s tim da je OP kod od ROTL zadan)

rješenje:

```
opcoide[1]: rj_sel, alu_sel=ADDA, r4_write, goto opcode 1.1;
```

```
opcode 1.1: a_sel = 4, b_sel = 4, alu_sel=ADD, r4_write;
```

```
if c_out then goto opcode 1.2 else goto opcode 1.3;
```

```
opcode 1.2: a_sel = 4, c_in, alu_sel=ADDA, ri_sel, goto fetch0;
```

```
opcode 1.3: a_sel = 4, alu_sel=ADDA, ri_sel, goto fetch0;
```

8) zadatak- napiši naredbu $\text{ADD } ri, rj, \text{konst}(rk) \Rightarrow ri \leftarrow rj + \text{MEM}(\text{konst} + rk)$;

rješenje:

```
opcode[6]: result_sel=IR_CONST4, r4_write;
```

```
rk_sel, a_sel=4, alu_sel=ADD, mar_sel=LOAD, goto opcode6.1;
```

```
opcode6.1: mdr_sel=LOAD_MEM, read, if wait then goto opcode6.1 endif;
```

```
result_sel=MDR, r4_write;
```

```
ri_sel, rj_sel, b_sel=4, alu_sel=ADD, goto fetch0;
```

9) LOAD Ri, konst(Rj) // $MAR \leftarrow ir_const4 + Rj$, $Ri \leftarrow MEM(MAR)$

rješenje:

```
opcode[10]: result_sel = ir_const4, rj_sel, b_sel=4, alu_sel=ADD, mar_sel=LOAD, goto
opcode10.1;
```

```
opcode10.1: read, while wait goto opcode 10.1 endif;
```

```
mdr_sel=LOAD_MEM, result_sel=mdr, ri_sel, goto fetch0;
```

10) zadatak- napiši naredbu ADD ri, rj, konst(rk) $\implies ri \leftarrow rj + MEM(konst + rk)$;

rješenje:

```
opcode[6]: result_sel=IR_CONST4, r4_write;
```

```
rk_sel, a_sel=4, alu_sel=ADD, mar_sel=LOAD, goto opcode6.1;
```

```
opcode6.1: mdr_sel=LOAD_MEM, read, if wait then goto opcode6.1 endif;
```

```
result_sel=MDR, r4_write;
```

```
ri_sel, rj_sel, b_sel=4, alu_sel=ADD, goto fetch0;
```

%

0: 000110 00

1: 01 10 0101