# Rješenje ispita – IP.26.04.2021

1: Odabrati tačne tvrdnje:
a) Ukoliko program, napisan u programskom jeziku C, radi sa nizom cijelih brojeva, onda će MIPS arhitektura koristiti lw instrukciju za upis u memoriju.
**b) Ukoliko program, napisan u programskom jeziku C, radi sa nizom cijelih brojeva, onda će MIPS arhitektura koristiti lw instrukciju za čitanje iz memorije.**
c) Ukoliko program, napisan u programskom jeziku C, radi sa nizom karaktera, onda će MIPS arhitektura koristiti sb instrukciju za čitanje iz memorije.
d) Ukoliko program, napisan u programskom jeziku C, radi sa nizom karaktera, onda će MIPS arhitektura koristiti sw instrukciju za upis u memoriju.
**e) Ukoliko program, napisan u programskom jeziku C, radi sa nizom karaktera, onda će MIPS arhitektura koristiti sb instrukciju za upis u memoriju.**
f) Ukoliko program, napisan u programskom jeziku C, radi sa nizom cijelih brojeva, onda će MIPS arhitektura koristiti lw instrukciju za upis u memoriju.
g) Nijedna od navedenih tvrdnji nije tačna.

2: Za instrukciju **bne t2, t3, 13** odabrati tačne tvrdnje:
a) Instrukcija je J-tipa
b) **Instrukcija je I-tipa**
c) Instrukcija je R-tipa
d) Konstanta **13** se specificira sa gornjih 16 bita.
e) **Konstanta 13 se specificira sa donjih 16 bita.**
f) Konstanta **13** se specificira sa 32 bita.
g) Instrukcija sprema rezultat u registar t2.
h) Instrukcija sprema rezultat u registar t3.
i) Instrukcija nije MIPS instrukcija.

3: Dobavljanje operanada iz registara se vrši u:
a) prvoj polovini EX faze
b) drugoj polovini EX faze
c) prvoj polovini ID faze
d) **drugoj polovini ID faze**
e) prvoj polovini IF faze
f) drugoj polovini IF faze

4: Za sekvencu instrukcija:
*lb r1, 16(r0)*
*beq r2, r1, 12*
*ori r2, r4, 4*
*add r2, r4, r5*
*sw r4, 12(r0)*
Koliki je procenat instrukcija kojima je potreban izlaz iz *Sign-Extend* komponente?
a) 60%
b) 100%
c) **80%**
d) 40%

5: Odabrati tačne tvrdnje:
a) Za instrukciju *beq r1, r2, 4 Zero flag* je uvijek postavljen na 1.
b) **Za instrukciju *addi r1, r2, 12* kontrolni signal *RegDst* je postavljen na 0, a *ALUSrc* je postavljen na 1.**

c) S obzirom da instrukcija *sub r2, r3, r4* upisuje rezultat izvršavanja u registar *r2*, kontrolni signal *MemToReg* je postavljen na 1.

d) **Za instrukciju *and r1, r2, r3* kontrolni signal *RegDst* je postavljen na 1, *ALUSrc* je postavljen na 0.**

e) S obzirom da instrukcija *lw r2, 0(r4)* čita podatak sa memorijske lokacije čija se adresa nalazi u registru *r4*, instrukcija ne koristi ALU komponentu.

f) Nijedna od navedenih tvrdnji nije tačna.

6: Neka je u nekom programu 25% instrukcija čitanja iz memorije podataka (load), 10% instrukcija pisanja u memoriju podataka (store), 10% instrukcija grananja i 55% aritmetičko-logičkih instrukcija. Iza 40% load instrukcija se nalaze instrukcije koje koriste registar u koji load instrukcija dobavlja podatak.
Program se izvršava u oglednoj protočnoj strukturi sa prosljeđivanjem.
CPI programa iznosi:

a) 1.4
b) 1.0
c) 1.25
d) 1.8
e) **1.1**

7: Neka je treća instrukcija u sekvenci instrukcija *lw r1,12(r0)* pri čemu su se prve dvije instrukcije izvršile bez zastoja. Odabrati tačne tvrdnje:

a) Instrukcija *lw* će vrijednost registra *r0* moći proslijediti na kraju šestog ciklusa izvršavanja sekvence instrukcija.

b) **Instrukcija *lw* će vrijednost registra *r1* moći proslijediti na kraju šestog ciklusa izvršavanja sekvence instrukcija.**

c) S obzirom da se radi o instrukciji I tipa, nikakvo prosljeđivanje nije moguće.

d) Instrukcija *lw* će vrijednost registra *r0* moći proslijediti na kraju petog ciklusa izvršavanja sekvence instrukcija.

e) Instrukcija *lw* će vrijednost registra *r1* moći proslijediti na kraju petog ciklusa izvršavanja sekvence instrukcija.

8: Neka se sljedeća sekvenca instrukcija izvršava u oglednoj protočnoj strukturi koji podržava mehanizam prosljeđivanja:

```
add r2, r4, r9
sub r3, r1, r2
lw r1, 50(r1)
and r2, r1, r5
```

Koliko je zastoja potrebno da bi se sekvenca pravilno izvršila?

a) nijedan
b) **jedan**
c) dva
d) nijedno od ponuđenih

9: Neka se izvršava sljedeća sekvenca instrukcija:

```
0x0: add r2,r2,r3
0x4: beq r2,r3,0x9
0x8: sub r8,r4,r5
...
0x28: sub r2,r2,r2
0x2C: add r7,r4,r5
```

Odabrati tačne tvrdnje:

a) Ukoliko kompajler pretpostavi da je vjerovatnoća da će doći do grananja 1%, kao instrukcija zadrške će biti odabrana instrukcija sa adresom 0x28.
b) Ukoliko kompajler pretpostavi da je vjerovatnoća da će doći do grananja 1%, kao instrukcija zadrške će biti odabrana instrukcija sa adresom 0x2C.
c) **Ukoliko kompajler pretpostavi da je vjerovatnoća da će doći do grananja 1%, kao instrukcija zadrške će biti odabrana instrukcija sa adresom 0x8.**
d) Prva instrukcija koja će se izvršavati je instrukcija sa adresom 0x4, ukoliko se koristi tehnika odgođenog grananja.
e) Instrukcija zadrške će biti instrukcija sa adresom 0x0.
f) **Ukoliko kompajler pretpostavi da je vjerovatnoća da će doći do grananja 98%, treća instrukcija od početka sekcence će biti na adresi 0x2C.**
g) Sekvenca instrukcija je takva da nije moguće odabrati instrukciju zadrške.
h) Instrukcija zadrške će se odabrati nakon što se uporede vrijednosti registara *r2* i *r3* korištenjem *Zero flag*-a.
i) Nijedna od ponuđenih tvrdnji nije tačna.

10: Neka se izvršava instrukcija grananja *beq r1,r2,0x10* koja se nalazi na adresi 0x40. Odabrati tačne tvrdnje:

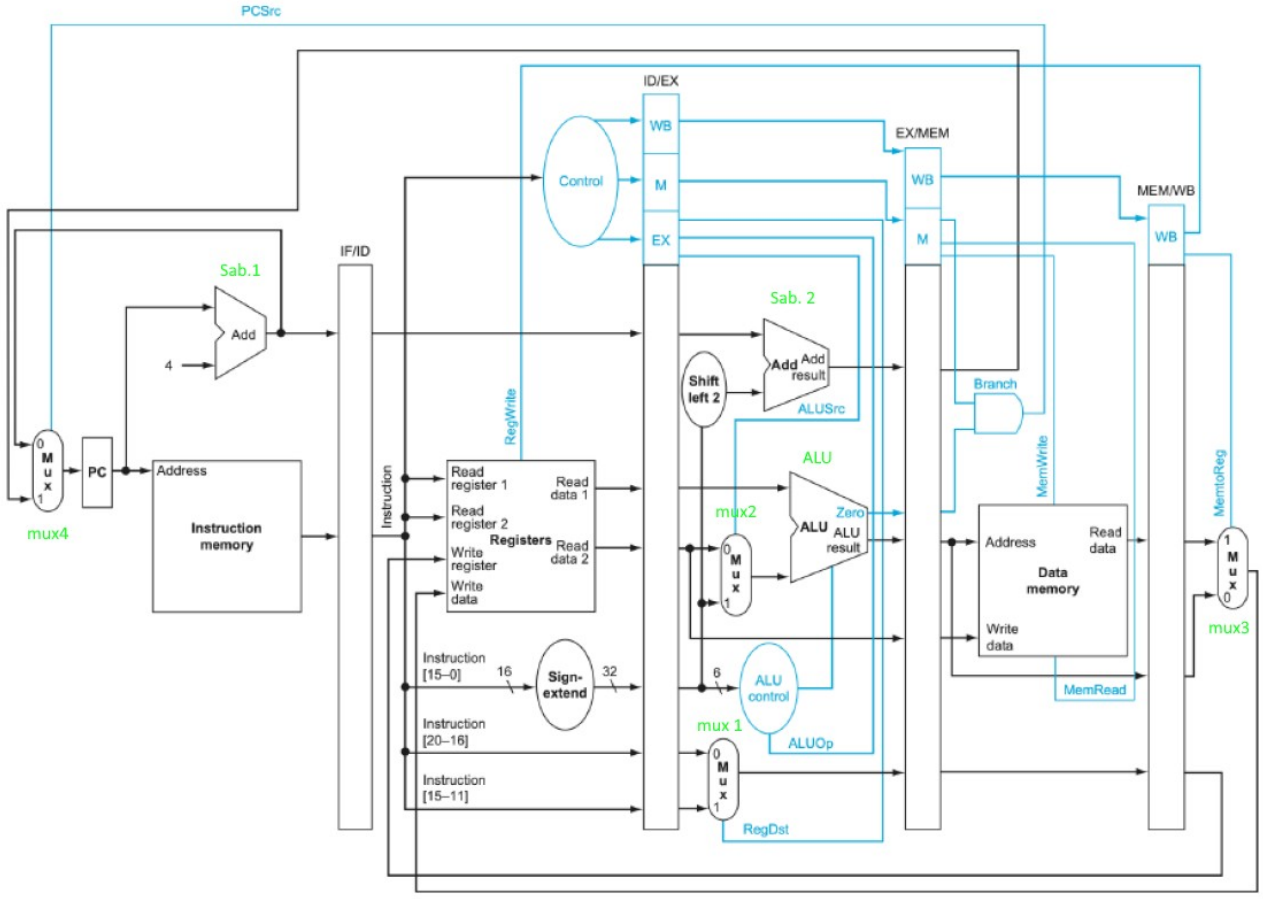
a) Ukoliko uslov grananja nije zadovoljen, adresa sljedeće instrukcije koja se izvršava je 0x41.
b) Ukoliko je uslov grananja zadovoljen, adresa sljedeće instrukcije koja se izvršava je 0x50.
c) Ukoliko je uslov grananja zadovoljen, adresa sljedeće instrukcije koja se izvršava je 0x54.
d) **Nijedan od ponuđenih odgovora nije tačan.**

**Zadatak 1:** Neka se na izlaznom portu memorije instrukcija nalazi riječ:
**00000001011010010101000000100010**
Pretpostaviti da su u memoriji podataka na svim adresama upisane vrijednosti 0, a vrijednost registra $r_i = i$.
Vrijednost polja *opcode* i *funct* je prikazana u tabeli:

**Opcode Table**

| Instruction | Opcode/Function | Syntax |
|---|---|---|
| add | 100000 | ArithLog |
| addu | 100001 | ArithLog |
| addi | 001000 | ArithLogI |
| addiu | 001001 | ArithLogI |
| and | 100100 | ArithLog |
| andi | 001100 | ArithLogI |
| div | 011010 | DivMult |
| divu | 011011 | DivMult |
| mult | 011000 | DivMult |
| multu | 011001 | DivMult |
| nor | 100111 | ArithLog |
| or | 100101 | ArithLog |
| ori | 001101 | ArithLogI |
| sll | 000000 | Shift |
| sllv | 000100 | ShiftV |
| sra | 000011 | Shift |
| srav | 000111 | ShiftV |
| srl | 000010 | Shift |
| srlv | 000110 | ShiftV |
| sub | 100010 | ArithLog |
| subu | 100011 | ArithLog |
| xor | 100110 | ArithLog |
| xori | 001110 | ArithLogI |
| lhi | 011001 | LoadI |
| llo | 011000 | LoadI |

| Instruction | Opcode/Function | Syntax |
|---|---|---|
| slt | 101010 | ArithLog |
| sltu | 101001 | ArithLog |
| slti | 001010 | ArithLogI |
| sltiu | 001001 | ArithLogI |
| beq | 000100 | Branch |
| bgtz | 000111 | BranchZ |
| blez | 000110 | BranchZ |
| bne | 000101 | Branch |
| j | 000010 | Jump |
| jal | 000011 | Jump |
| jalr | 001001 | JumpR |
| jr | 001000 | JumpR |
| lb | 100000 | LoadStore |
| lbu | 100100 | LoadStore |
| lh | 100001 | LoadStore |
| lhu | 100101 | LoadStore |
| lw | 100011 | LoadStore |
| sb | 101000 | LoadStore |
| sh | 101001 | LoadStore |
| sw | 101011 | LoadStore |
| mfhi | 010000 | MoveFrom |
| mflo | 010010 | MoveFrom |
| mthi | 010001 | MoveTo |
| mtlo | 010011 | MoveTo |
| trap | 011010 | Trap |



a) Koje su vrijednosti svih ulaza i izlaza u skupu registara (*register file*) kada je ova instrukcija u ID fazi?
b) Za multipleksere na gornjoj slici (mux1, mux2, mux3), sabirač (sab2) i ALU prikazati ulazne i izlazne vrijednosti, kao i vrijednosti kontrolnih signala za multipleksere kada je ova instrukcija u odgovarajućim fazama.

Rješenje:

**00000001011010010101000000100010**
Opcode instrukcije je 000000, ovo je instrukcija R tipa.

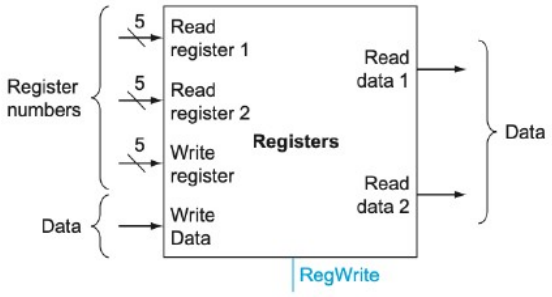
Razdvajamo instrukciju na polja:

| op | rs | rt | rd | shamt | funct |
|---|---|---|---|---|---|
| 6 bita | 5 bita | 5 bita | 5 bita | 5 bita | 6 bita |

**000000 01011 01001 01010 00000 100010**

Vidimo da je **funct** polje = **100010**, iz tabele čitamo da ova instrukcija predstavlja **sub** operaciju.
Vrijednost **rs** polja = **01011**, kodira registar r11 ($t3).
Vrijednost **rt** polja = **01001**, kodira registar r9 ($t1).
Vrijednost **rd** polja = **01010**, kodira registar r10 ($t2).
Instrukcija je: **sub r10, r11, r9** (**sub $t2, $t3, $t1**).

**a) Koje su vrijednosti svih ulaza i izlaza u skupu registara (*register file*) kada je ova instrukcija u ID fazi?**




| Ulaz | Vrijednost |
|---|---|
| Read register 1 | 01011 |
| Read register 2 | 01001 |
| Write register | 01010 |
| Write data | nedefinisano |

| Izlaz | Vrijednost |
|---|---|
| Read data 1 | 11 |
| Read data 2 | 9 |

vrijednost registra $r_i = i$

**b)Za multipleksere na gornjoj slici (mux1, mux2, mux3), sabirač (sab2) i ALU prikazati ulazne i izlazne vrijednosti, kao i vrijednosti kontrolnih signala za multipleksere kada je ova instrukcija u odgovarajućim fazama.**

| Mux 1 | |
|---|---|
| Ulaz 0 | 01001 |
| Ulaz 1 | 01010 |
| Signal | 1 |
| Izlaz | 01010 |

| Mux 2 | |
|---|---|
| Ulaz 0 | 9 (32 bita) |
| Ulaz 1 | 01010 00000 100010 (32 bita) |
| Signal | 0 |
| Izlaz | 9 (32 bita) |

| Mux 3 | |
|---|---|
| Ulaz 0 | 2 (32 bita) |
| Ulaz 1 | nedefinisano |
| Signal | 0 |
| Izlaz | 2 (32 bita) |

| Sabirač 2 | |
|---|---|
| Ulaz 0 | PC + 4 |
| Ulaz 1 | 01010 00000 100010 (32 bita) * 4 |
| Izlaz | (PC + 4) + (01010 00000 100010 (32 bita) * 4) |

| ALU | |
|---|---|
| Ulaz 0 | 11 (32 bita) |
| Ulaz 1 | 9 (32 bita) |
| Signal | Kontrolna vrijednost koja inicira oduzimanje u ALU |
| Izlaz | 2 (32 bita) |
| Zero flag | 0 |

**Zadatak 2:** Data je sekvenca instrukcija:

```
    lw r4, 3(r0)
    add r2, r1, r5
    bne r6, r1, lbl
    sub r7, r1, r0
    and r5, r1, r5
lbl: add r3, r1, r5
    sw r3, 3(r0)
```

Pretpostaviti da su u memoriji podataka na svim adresama upisane vrijednosti 0, a vrijednost registra $r_i = i$.
a) Ukoliko protočna struktura podržava prosljeđivanje, napisati rapored faza izvršavanja po ciklusima. Podrazumijeva se korištenje mehanizma odgođenog grananja.
b) Objasniti uzroke svih zastoja.
c) Navesti gdje i kada je potrebno prosljeđivanje podataka.
d) Napisati sadržaj EX/MEM pregradnog registra u 6. ciklusu.

**a) Ukoliko protočna struktura podržava prosljeđivanje, napisati rapored faza izvršavanja po ciklusima. Podrazumijeva se korištenje mehanizma odgođenog grananja.**

Potrebno je 9 ciklusa.

| | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 |
|---|---|---|---|---|---|---|---|---|---|
| *lw r4, 3(r0)* | IF | ID | EX | M | WB | | | | |
| *bne r6, r1, lbl* | | IF | ID | EX | M | WB | | | |
| *add r2, r1, r5* | | | IF | ID | EX | M | WB | | |
| *lbl: add r3, r1, r5* | | | | IF | ID | EX | M | WB | |
| *sw r3, 3(r0)* | | | | | IF | ID | EX | M | WB |

**b) Objasniti uzroke svih zastoja.**
Iz gornje tabele vidimo da u sekvenci nema zastoja.

**c) Navesti gdje i kada je potrebno prosljeđivanje podataka.**
Prosljeđivanje je potrebno u 7. ciklusu iz EXE/MEM pregradnog registra u EXE fazu sw instrukcije.

**d) Napisati sadržaj EX/MEM pregradnog registra u 6. ciklusu.**

U 6. ciklusu *lbl: add r3, r1, r5* je u EXE fazi. EX/MEM pregradni registar sadrži sljedeće vrijednosti:

1) Adresu koja predstavlja odredište grananja(32 bita): (PC + 4) + (4*posljedjih 16. bita ins: *add r3, r1, r5*)
2) Rezultat izvršavanja aritemtičko-logičke operacije (32 bita): 1+5=6
3) Vrijednost zero flag-a (1 bit): 0
4) Vrijednost pročitana iz registarske datoteke(32 bita): Vrijednost registra r5 = 5
5) Redni broj odredišnog registra: r3 je odredišni reegistar, redni broj je 3.
6) Sve MEM i WB kontrolne signale:
   I. Branch: 0
   II. MemWrite: 0
   III. MemRead: 0
   IV. RegWrite: 1
   V. MemToReg: 0