

1. Čemu služe registri $x_{10} - x_{13}$ u RISC-V arhitekturi?

- RISC-V ima 8 registara argumenta ($x_{10} - x_{17}$) koji se koriste za prosledjivanje argumenta u potprogramu.

2. Koje su prednosti korišćenja neposrednog adresiranja u instrukcijama?

- Neposredno adresiranje je najjednostavniji oblik adresiranja gde je vrednost operanda prisutna u instrukciji.

- Prednost je što neva drugih referenci memorije sum doноšења instrukcije koje se zahteva da bi se dobio operand.

- Time se štedi jedan ciklus memorije ili keša u ciklusu instrukcije.

- Nedostatak je što je veličina broja ograničena dimenzijom polja za adresu.

3. I-format instrukcija i značenje pojedinih polja.

- Instrukcije I-tipa konstantu čuvaju unutar sebe, u 16-bitnoj immediate polju.

- Koriste konstantu kao operand.

④ Koji su koraci u pozivu procedura?

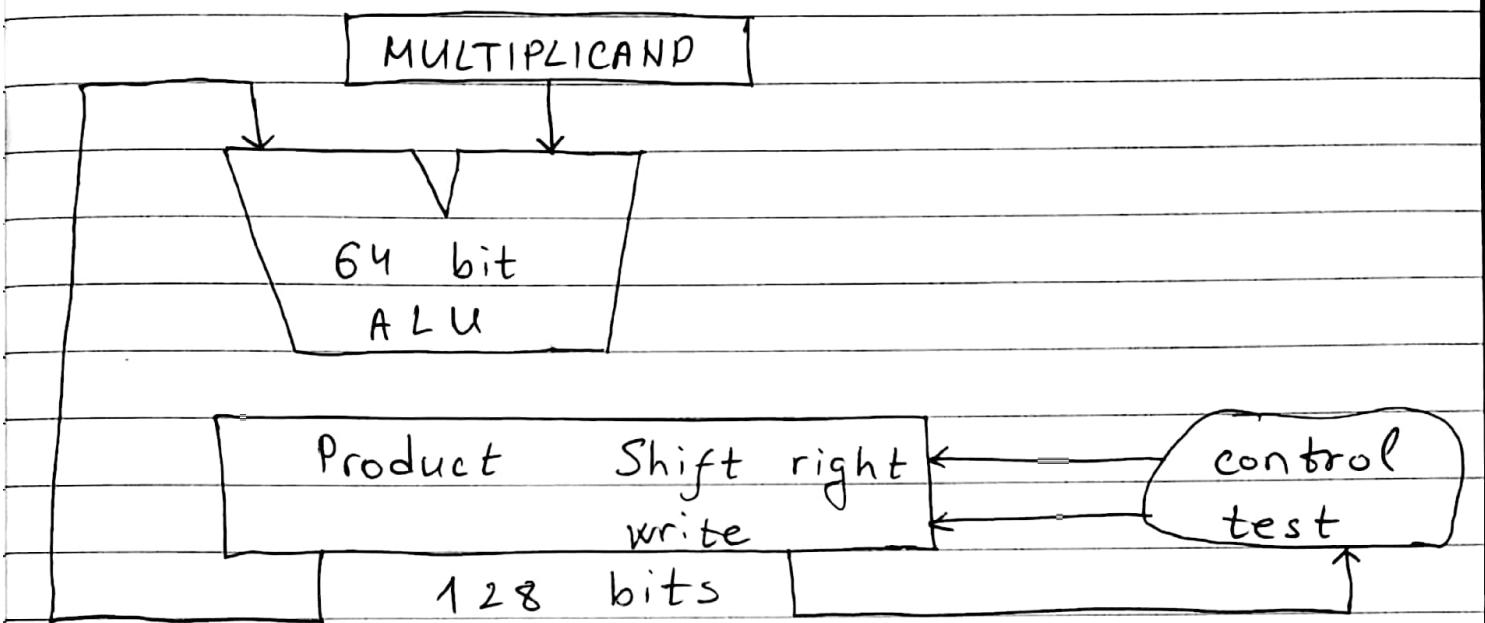
- 1) Postavlja parametre u registre, od x_{10} do x_{17} ;
- 2) Prebavlja kontrolu u registre;
- 3) Nabavlja skladiste za procedure;
- 4) Obavlja operacije procedure;
- 5) Postavlja rezultate u registre za pozivače;
- 6) Vraca se na mesto poziva.

⑤ Čemu služi lui instrukcija? Narediti primer i objasniti.

- Instrukcija lui pripada instrukcijama I-tipa.
- Koristi se za podržavanje rednih konstanti.
- U 20 gornjih bita registar RD svestra vrednost između polja, a u 12 donjih bita svestra 0.

⑥ Nacrtati šemu optimizovanog kola za množenje i objasniti princip rada.

- Množilac se nalazi u desnoj polovini proizvoda koji je za 1 bit narastao na 129 bitova da zadrži iznošenje sabirka.
- Rezultat dodavanja se nalazi u levoj polovini registra proizvoda.



7. Predstaviti broj -25.5 u standardu IEEE Std 754-1985.

$$-25.5 = -11001,1$$

$$-25.5 = -1,10011 \cdot 2^4$$

$$0.5 \cdot 2 = 1$$

ZNAK : - (1)

EXP : 4 (100)

MANTISA : 10011

$$\text{BROJ} : \text{ZNAK} + \text{EXP} + \text{MANTISA} = 110010011$$

8. Čemu služi element Shift left 1 u arhitekturi RISC-V procesora?

- Shift left povećava svaki bit ulovo za jedan.

- Bit nižeg reda se zamenjuje nultim bitom, a bit višeg reda se odbacuje.

(9.) Iz kojih stepena se sastoji RISC-V pipeline?
Šta se radi u stepenu WB?

- Stepeni su : IF, ID, EX, MEM, WB.
- WB (write back) je faza upisa rezultata izvršavanja instrukcije u registrarsku banku.

(10.) Čemu služi mehanizam prosledivanja (forwarding)?

- Koristi se kod rešavanja hazarda.

(11.) Koja je prednost korišćenja registara uveštosti radne memorije?

- Prednost korišćenja registara se ogleda u koupanutnoj operandskoj specifikaciji, što se svodi na favorizaciju dužine instrukcije.
- Ovim se štedi memorijiski prostor i obezbeđuje brže izvršenje instrukcionog ciklusa pribavljanja.
- Vreme pristupa registru je znatno kraće od vremena pristupa memoriji.

12. R-format RISC-V instrukcija i značenje pojedinih polja.

- Aritmetičke i logičke operacije koriste R-format.
- Sve instrukcije imaju isto opcode polje, dva izvorisna i jedan ciljni registar.
- Instrukcije se razaznaju na osnovu funct₃ i funct₇ polja.

13. Navesti faze u pipeline organizaciji procesora i šta se radi u svakoj fazi.

- Faze su :
 - 1) IF - instruction fetch : prihvata instrukcije iz memorije;
 - 2) ID - instruction decode : faza čitanja iz registarske banke i dekodovanja instrukcije;
 - 3) EX - execute : faza u kojoj ALU izvršava određenu operaciju u zavisnosti od prihvaccene instrukcije;
 - 4) MEM - memory access : faza pristupa memoriji za podatke ;
 - 5) WB - write back : faza upisa rezultata izvršavanja instrukcije u registarsku banu.

14. Šta je load-use data hazard? Navesti primer i objasniti.

- Load-use je vrsta hazarda podataka gde podaci koji se učitavaju pomoću load instrukcije nisu dostupni kad su potrebni drugoj instrukciji.
- On zahteva odlaganje izvršenja instrukcije dok rezultat load instrukcije bude dostupan.

15. Napisati i objasniti formule koje se koriste za detekciju potrebe za prosleđivanjem (forwarding).

- Prosleđivanje koristi rezultat nakon izračuna vanja.
- Ne čeka da se sacura u registru.
- Zahteva dodatne veze.

16. ENIAC - opis i način rada.

- ENIAC je prvi računar korišćen za automatsko izračunavanje balističkih podataka.
- Mogao je da ravnaje desetocifrenim brojevima.
- Programiranje ENIAC-a se srođilo na uključivanje i isključivanje kablova i prekidača.
- Njegov program nije bio usuto dišten u centralnu memoriju, ali je mogao da izvodi operacije elektronskow brzinom.

- Koristio je bušene kartice za podatke.
- Bio je ogroman, stalno se kvario.
- Kasnije je korišćen za računanje vremenske prognoze.

17. Osobine 8086 i 8088 procesora.

- Intel 8086 je 16-bitni mikroprocesor.
- Radi sa 16-bitnim binarnim rečima.
- Imala 16-bitnu magistralu podataka.
- Može da prenosi 16 bita ili 8 bita istovremeno.
- Intel 8088 imala brzinu taktova 5-10 MHz.
- Imala 16-bitne registre, 20-bitnu adresnu magistralu i 16-bitnu eksternu magistralu podataka.
- Podržava 1MB memorije.

18. Koliko registara imala RISC-V, a koliki je kapacitet memorije, tj. adresibilni memorijski prostor?

- RISC-V poseduje 32 registra ($x_0 - x_{31}$).
- Adresibilni memorijski prostor se sastoji od 2^{61} memorijskih reči.

(19.) Kako se u RISC-V implementira sinhronizacija?

- 1) Dva procesora koji dele deo memorije;
- 2) Rezultat zavisi od reda pristupa;
- 3) P₁ piše, zatim P₂ čita;
- 4) Trka podataka ako P₁ i P₂ nisu sinhronizovani.

(20.) Zašto su razdvojene memorije za podatke i instrukcije?

- Putanja podataka mora imati odvojene memorije instrukcija i podataka, jer su formati podataka i instrukcija različiti u MIPS-u i stoga se koriste različite memorije.

(21.) Šta je upravljački, tj. kontrolni hazard?
Navesti primer i objasniti.

- Upravljački hazardi nastaju zbog skokova i drugih instrukcija koje menjaju vrednost programskog brojača PC.

- Grana određuje ton kontrole.

- Preuzimanje sledeće instrukcije zavisi od ishoda grane.

- Pipeline ne može uvek da obuhvati ispravnu instrukciju.

- Još uvek radi na fazi ID proiznike.

(22) Šta je staticka, a šta dinamička prediktija suokova?

- Staticka prediktija suoka je fiksna za celokupno izvršavanje.

- Dinamička prediktija se menja u zavisnosti od izvršavanja programa.

(23) Navesti nekoliko najznačajnijih dogadaja u istoriji računarstva do 20. veka.

- U razvoju računarstva značajna su četiri dogadaja:

- 1) pamćenje rezultata;
- 2) mehanizacija procesa računanja;
- 3) odvajanje unošenja podataka i automatizacija procesa računanja;
- 4) opštije korišćenje mašine priuvenom programa.

(24) Osobine 8085 procesora.

- 1) 8-bitna magistrala podataka;
- 2) 16-bitna adresna magistrala koja može adresirati do 64KB;

- 3) 16-bitni programski brojač;
- 4) 16-bitni pokazivač steka;
- 5) šest 8-bitnih registara raspoređenih u parove.

(25) Navesti barem jednu instrukciju uslovnog skoka i jednu instrukciju bezuslovnog skoka u RISC-V arhitekturi.

- Uslonna : branch if equal
- Bezuslonna : jump and link

(26) Zašto se u instrukcijama koristi neposredna veličina (immediate operand)?

- Koristi se za specifikaciju konstante za jednu operaciju ili za davanje adrese skoka za instrukcije grananja.

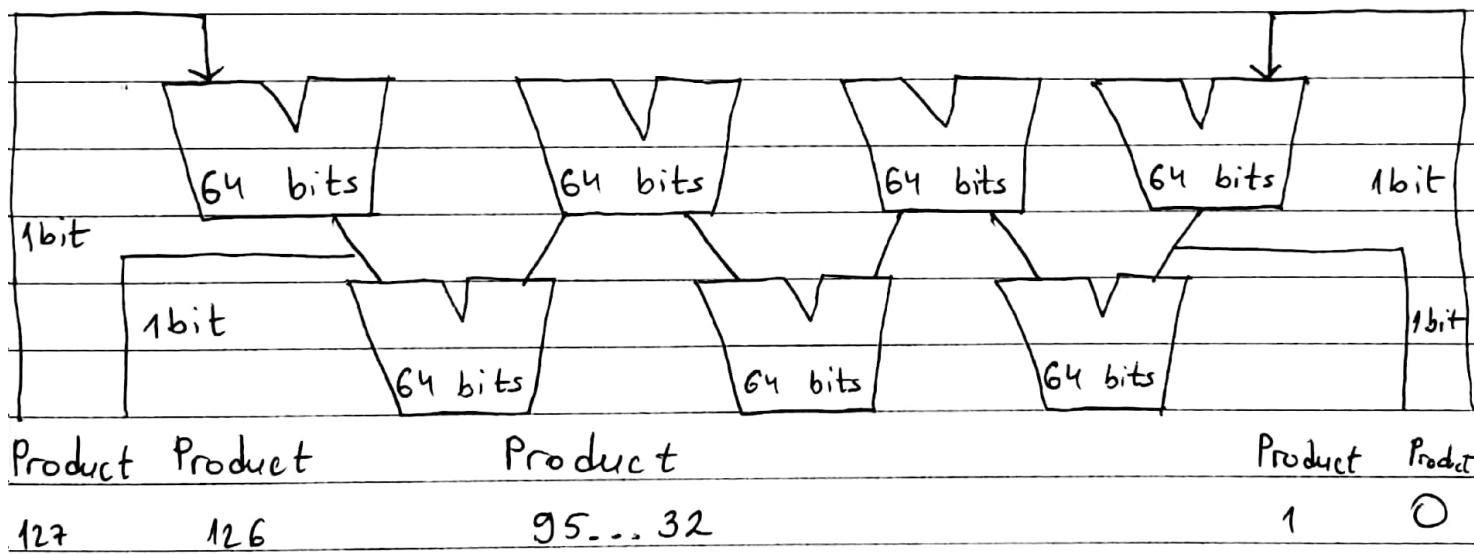
(27) Navesti barem jednu instrukciju koja koristi S-format.

- rs1 : broj registara matične adrese;
- rs2 : registarski broj izvornog operanda;

(28) Koja je jedna od uobičajenih primena OR operacije? Navesti primer.

- Korisno za dodavanje bita u reci.
- Postavlja newe bite na 1, a ostale ostavlja nepročuvnjene.

(29) Načrtati principijelu řemu brzog kola za množenje i objasniti način rada.



- Svaki sabirač proizvodi 64 bita i carry izlaz.
- Njegovanje značajan bit svakog srednjeg zbirala je deo proizvoda.
- Ostalih 63 bita i carry izlaz su preneseni drugom sabiraču.

(30) Kako se u RISC-V predstavlja NaN vrednost?

- Exponent = 111...1;
- Fraction \neq 000...0;
- Not-a-Number;
- Uказује на неизаконит или неodređen rezultat.

(31) Šta je to protočna obrada (pipeline) i zašto se koristi?

- Protočnost je standardna hardverska tehnika koja se koristi za postizanje boljih performansi.
- Protočnost je veoma učinkna tehnika za ubrzanje izvršenja sličnih izračunavanja u nizu.

(32) Šta su to hazardi podataka?

- Hazardi podataka nastaju kada je pristup nekom podatku od strane neke instrukcije u nekom stepenu pipeline-a uslovljen prethodnim pristupom tisu istom podatku od strane neke prethodne instrukcije iz nekog drugog stepena pipeline-a.

(33.) Čemu služe registri x_2 i x_8 u RISC-V arhitekturi?

- x_2 se koristi kao pokazivač steka (stack Pointer) i drži osnovnu adresu steka.
- x_8 (Frame Pointer) sadrži osnovnu adresu okvira funkcije.

(34.) Zasto se zahteva da kompjuter što češće koristi registre prilikom generisanja koda?

- zato što je registarsko adresiranje najbrži i najučinkovitiji način adresiranja, pa se najčešće korisćene vrednosti čuvaju u registru.

(35.) Čemu služe polja funct₃ i funct₇ u R-formatu instrukcija?

- Funct₃ su dodatna 3 bita polju opcode, za detaljnije definisanje instrukcije.
- Funct₇ su dodatnih 7 bita polju opcode, za detaljnije definisanje instrukcije.

(36) Koja je jedna od uobičajenih primena AND operacije? Navesti primer.

- Maskiranje bitova u reči.

(37) Navesti barem tri floating-point instrukcije u RISC-V.

- fadd.s, fsud.s, fmul.s

(38) Ako procesor ima K stepena i izvršavanje u svakom stepenu traje 1 takt, koliko je ukupno vreme izvršavanja programa od n instrukcija u slučaju kad se koristi i kad se ne koristi prototična obrada (pipeline)?

$$n + K - 1 \rightarrow \text{sa pipelineom}$$

(39) Koji je generalni format RISC-V instrukcija? Koja sre polja postoje?

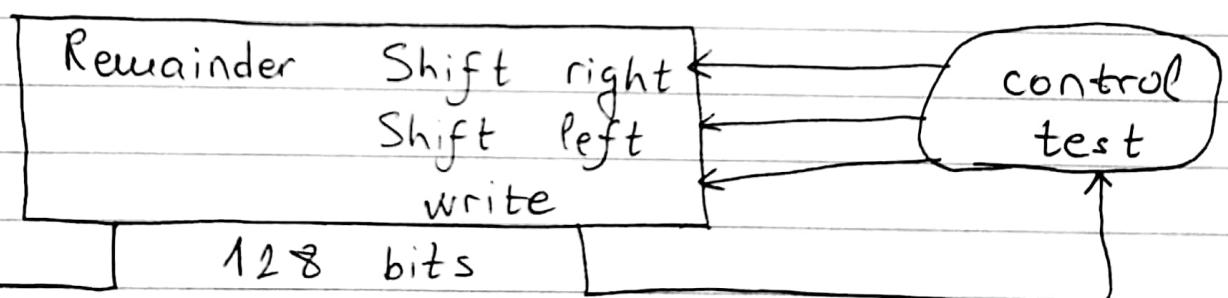
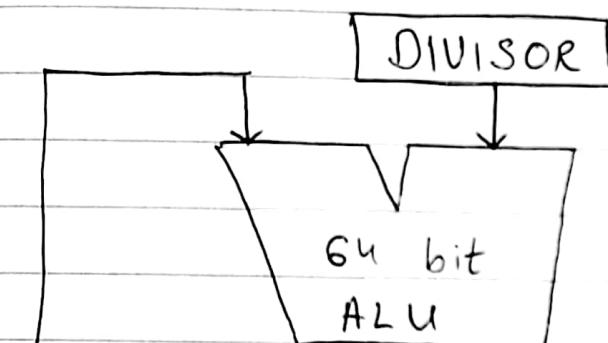
- Generalni format je R-format.

- Polja koja se javljaju su:
- 1) opcode;
 - 2) funct3;
 - 3) funct7.

(40.) Šemu služi registar x_0 u RISC-V arhitekturi?

- x_0 se koristi za inicijalizaciju drugih registara na nulu.
- Uvек је jednak nuli.

(41.) Načrtati principijelnu šemu optimizovanog kola za deljenje i objasniti način rada.



- Kombinuje Quotient registar sa desnom polovinom remaindera registra.
- U početku je to dividenda.
- Remainder registar je porastao do 129 kada se osigura da carry izlaz sabiraca nije izgubljen.

42. Čemu služi i zašto se koristi element sign extend u arhitekturi RISC-V procesora? Da li ga koriste branch instrukcije i zašto?

- Sign extend je operacija povećanja broja bitova binarnog broja uz očuvanje predznaka broja i mernosti.
- Koristi se u RISC-V instrukcijama.
- Branch instrukcije ga koriste za izračunavanje target adrese.

43. Čemu služe elementi ALUOp i ALUControl (u arhitekturi RISC-V procesora)?

- ALUOp je 2-bitno kontrolno polje.
- ALUOp je izveden iz opcode-a.
- ALUControl se koristi za
 - 1) load / store;
 - 2) branch;
 - 3) R-type.

44. Koji formati instrukcija postoje u RISC-V arhitekturi? Koji format koriste load instrukcije?

- Formati instrukcija u RISC-V-u su:
 - 1) R-format;
 - 2) I-format;

- 3) S - format;
- 4) SB - format;
- 5) U - format;
- 6) UI - format.

- Load instrukcije koriste I - format.

45. Šta je jal, a šta jeal instrukcija?

- Jal i jalr su instrukcije za rad sa procedurama.
- Jal se koristi za poziv procedure.
- Jalr se koristi za povratak procedure.

46. Koji su koraci u izvršavanju procedura?

→ Pisala već, budala.

46. Koje dve instrukcije služe kao podrška za sinhronizaciju u RISC-V?

- Load reserved i store conditional.

47. Kako izgleda format za zapis celih brojeva u jednostrukoј tačnosti (single) u IEEE std 754-1985 formatu?

- $(-1)^S \times F + 2^E$.

48. Koja je veličina registarskog fajla u RISC-V arhitekturi? Koja je dužina reči i kakvo je adresiranje (veličina adresibilne jedinice) u RISC-V arhitekturi?

- Veličina registarskog fajla je 32×64 bita.
- 32 bita - word
- 64 bita - double word

49. Objasniti mehanizam dinamičkog povezivanja (DLL).

- Dinamičko povezivanje se vrši u toku izvršavanja i koristi posebnu varijaciju formata biblioteka koja se zove biblioteka dinamičke vete (DLL).
- Pristup je povoljan uada više aplikacija radi u istom sistemu i zahteva ju istu biblioteku.

50. Šta su to strukturni hazardi?

- Strukturni hazard nastaje kada dve instrukcije koje se nalaze u različitim stepenima pipeline treba da pristupe istom resursu.

51. Prednosti i uune korišćenja posebnih memorija za instrukcije i podatke u RISC-V arhitekturi.

- Korišćenje jedinstvene memorije za podatke i instrukcije dovodi do efikasnijeg korišćenja memorije i tipično je kao rešenje kod ranijih računarskih sistema.
- Savremeni RISC procesori koriste veći broj puteva za prenos podataka između procesora, memorije i UI/I podistema.

PROJEKTNI ZADACI 2022:

PRVI projektni zadatak: Napisati asemblerски program za RISC-V koji na poziciji (westu) gde se nalazi najveći element niza upisuje svaku elemenata niza. Niz je potrebno uneti direktno u asemblerски kod. Dozvoljeno je korišćenje pseudoinstrukcija. Ako nešto nije definisano usvojiti razumnu pretpostavku. Preporučuje se korišćenje Venus RISC-V simulatora, ali je i dozvoljeno koristiti bilo koji RISC-V simulator.

Rešenje:

.data

ARRAY:

.word 9
.word 10
.word 15
.word 11

niz trenutno ima 4 elementa zbog lakše provore step by step

.text

la x18, ARRAY # smestimo početnu adresu niza u reg x18 ($y18 - x27 \rightarrow$ saved registers)
// instrukcija za smestanje adrese, prvi član niza je baza adresa niza

addi x28, x0, 0 # inicijalizujemo brojač: $i = 0$

addi x9, x0, 4 # inicijalizujemo dužinu niza zbog broja petlji: $n = 4$

lw x29, 0(x18) # prvi element je maksimum,
uporedjamo sa svanim sledecim
addi x30, x0, 0 # u registru x30 cemo
sustiti maksimum i on je inicijalno = 0

LOOP :

beg x28, x9, END # ukoliko je brojac jednak
broju petlji idemo u END fazu, ne ulazimo
u funkciju MAX (i == n => END)
slli x7, x28, 2 # shiftamo x28 za 2 mesta i
sustavimo u x7, dobijamo poweraj do
sledeceg elementa niza

add x31, x18, x7 # dobijanje adrese trenutnog
elementa niza iz sabiranja poweraja i pocetne
adrese u x18

lw x19, 0(x31) # u x19 sustavimo vrednost
i-tog elementa procitanu sa x31

add x30, x30, x19 # sabiramo trenutnu sumu
(u pocetku = 0) sa vrednoscu trenutnog eleme-
nta niza iz x19 i sustavimo u x30
bge x19, x29, MAX # ukoliko je vrednost trenutnog
elementa iz x19 veca od maksimuma idemo u
func MAX

jal x0, INCREMENT # skok na funkciju za
incrementiranje

MAX:

```
add x29, x19, x0 # setujemo novi maksimalni  
element niza  
addi x5, x31, 0 # u temp registar x5 cuvamo  
adresu  
jal x0, INCREMENT # skok na funkciju za  
incrementiranje
```

INCREMENT:

```
addi x28, x28, 1 # incrementiramo brojac  
jal x0, LOOP # skacemo ponovo u petlju
```

END:

```
sw x30, 0(xs) # skladisti u memorijeskoj  
lokaciji koja je u x6 sadrzaj registra x30
```

DRUGI projektni zadatak: Napisati asemblerски program za x86 koji na poziciji (vestu) где се налази највећи element niza upisuje sumu elemenata niza. Niz има 10 elemenata, а елементе низа је дозвољено унети директно у assemblerски код. Ако нешто није дефинисано усвојити razumnu pretpostavku. Preporучује се коришћење MASM assemblera, ali је дозвољено користити и било који drugi.

- REŠENJE :

name "Projekat 2"
org 100h ;

.data

elements db 2, 1, 6, 5, 20 ; elements = [2, 1, 6, 5, 20]

; H = bits [16-8]
; L = bits [7-0]

; GENERAL REGISTERS

; AX, AH, AL : Called the Accumulator register.
It is used for I/O port access, arithmetic,
interrupt calls, etc... // AX je određeni regi-
star, tu svezatamo nulu

; BX, BH, BL : Called the Base register / AH
(high) gornjih 8 bita, AL (low) donjih 8
bita, AX svih 16 bita registra. It is used
as a base pointer for memory access.

; CX, CH, CL : Called the Counter register. It
is used as a loop counter and for
shifts.

; DX, DH, DL : Called the Data register. It is used for I/O port access, arithmetic, some interrupt calls.

GENERAL REGISTERS

; DI : Destination Index

; SI : Source Index

.code

mov cl, 5 ; cx[7-0] = elements.length
// tu smestamo duzinu niza

mov al, 0 ; ax[7-0] = sum(elements) = 0
// odredeni registar, tu stavljamo da je suma = 0

mov bl, 0 ; bx[7-0] ce da storuje trenutni element niza // base registar za pristup memo-rijii, adresa za pristup je trenutni element niza

mov si, 0 ; si → index

mov dl, elements[si]; dx[7-0] = max(elements) = elements[0] // data registar, smestamo multi-element niza kao max

check Max; ; cx-- after each loop

mov bl, elements[si]; bl = elements[si] // u bl se smesta trenutni element niza

add al, bl ; al += bl // dodaje trenutni element na sumu

• ije predstavlja instruciju uslovnog skoka kojoj obično prethodi cmp, ili ~~an~~ and

• uslov je takav da ako je sadržaj dl veci ili jednaku bl da se pređe na granu continue

• u suprotnom se nastavlja sa izvršavanjem instrukcija

cmp dl, bl ; dl - bl > 0 ? // trenutni element niza upoređuje sa trenutnim maksimumom

ije continue ; if (SF = OF) skip max
change and jump to next function
// ukoliko je flag 1 biće continue,
uvod di, si ; di = index of max // ako je trenutni veci uvi čuvamo njegov index ;
onda njegovu vrednost
uvod dl, bl ; dl = new max

continue :

inc si ; sitt
loop checkMax ; if (count > 0) → loop
// vraćamo se na petlju max

uvor elements [di], al; elements [index of max]
= sum(elements) na poziciji najvećeg elementa
niza stavljaju sumu

ret

TREĆI projektni zadatak: Potrebno je implementirati simulator RISC-V kola za umnoženje i deljenje celih brojeva. Za obe operacije je potrebno napraviti osnovnu i optimizovanu verziju. Simulator treba da ima odgovarajući vizuelni grafički korisnički interfejs. Svaki korak (iteracija) umnoženja i deljenja treba da bude jasno vidljiv u simulatoru, tj. grafičkom interfejsu.

=> neuma rešeno.

(2.) APRIL 2021: IEEE 754 - 2008 standard koristi 16 bita za snestanje celih brojeva. U formatu postoji bit za znak (prvi - lev), eksponent (5 bita, pri čemu je bias 15) i mantisa (10 bita). Podrazumeva se surivna jedinica (hidden 1). Dati su brojevi: 2.645×10^2 i 48.75×10^1 .

- a) Napisati binarnu reprezentaciju navedenih brojeva u datom standardu.
- b) Izvršiti sabiranje navedenih brojeva. Za potrebe zaokruživanja se koriste 1 guard, 1 round i 1 sticuy bit. Napisati binarnu reprezentaciju rezultata u datom standardu.

$$a) 2.645 \cdot 10^2 = 264.5 \cdot 10^0 = 264.5 \cdot 1 = \\ \Rightarrow 264.5 = 100001000,1$$

	1	0	1	1	1
1	0	0	0	0	1
2^9	2^8	2^7	2^6	2^5	2^4
512	256	128	64	32	16
0.5 \cdot 2 = 1					

$$\Rightarrow 264.5 = 1,000010001 \cdot 2^8 \\ \text{ZNAK: } + (0)$$

$$\text{EXP: } 8+15=23 \quad (10111)$$

$$\text{MANTISA: } 0000100010$$

1 BROJ: ZNAK + EXP + MANTISA = 0 10111 0000100010

$$48.75 \cdot 10^1 = 4.875 \cdot 10^0 = 4.875 \cdot 1 = 4.875$$
$$\Rightarrow 4.875 = 100,111 = 1,00111 \cdot 2^2$$

$$0.875 \cdot 2 = 1.75$$
$$0.75 \cdot 2 = 1.5$$
$$0.5 \cdot 2 = 1$$

ZNAK : + (o)

EXP : $2+15=17$ (10001)

$$\begin{array}{r} 1 \\ \hline 2 \\ 16 \end{array} \quad \begin{array}{r} 0 \\ \hline 2 \\ 8 \end{array} \quad \begin{array}{r} 0 \\ \hline 2 \\ 4 \end{array} \quad \begin{array}{r} 0 \\ \hline e' \\ 2 \end{array} \quad \begin{array}{r} 1 \\ \hline 2^0 \\ 1 \end{array}$$

MANTISA : 0011100000

BROJ : ZNAK + EXP + MANTISA = 0100010011100000

b) $2.645 \cdot 10^2 + 48.75 \cdot 10^1 = ?$

$$2.645 \cdot 10^2 = 1,000010001 \cdot 2^8$$

$$48.75 \cdot 10^1 = 1,00111 \cdot 2^2 = 0,00000100111 \cdot 2^8$$

ZNAK : + (o)

$$\begin{array}{r} 1 \\ \hline 1,00001000100 \\ + 0,00000100111 \\ \hline 1,00001101011 \end{array}$$

G = 0 R = 0 S = 0

$$1,00001101011 \cdot 2^8 = 100001101,011 \cdot 2^0 = \\ \Rightarrow 269, 375$$

<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1

$$264 \cdot 5 + 4 \cdot 875 = 269,375$$

② JUN 2021: IEEE 754 - 2008 standard koristi 16 bita za snimanje celih brojeva. U formatu postoji bit za znak (prvi s leva), eksponent (5 bita, pri čemu je bias 15) i mantisa (10 bita). Podrazumeava se surivna jedinica (hidden 1). Dati su brojevi: -2.6125×10^1 i 1.5625×10^{-1} .

a) Napisati binarnu reprezentaciju navedenih brojeva u datom standardu.

b) Izvršiti množenje navedenih brojeva. Za potrebe zaukreživanja se koriste 1 guard, 1 round i 1 sticky bit. Napisati binarnu reprezentaciju rezultata u datom standardu.

$$a) -2.6125 \cdot 10^1 = -26.125 \cdot 10^0 = -26.125 \cdot 1 = \\ \Rightarrow -26.125 = -11010,001 = -1,1010001 \cdot 2^4$$

<u>1</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>
2^7	2^6	2^5	2^4	2^3
128	64	32	16	8
<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>

$$0.125 \cdot 2 = 0.25$$

$$0.25 \cdot 2 = 0.5$$

$$0.5 \cdot 2 = 1$$

ZNAK : - (-)

$$EXP : 4 + 15 = 19 (10011)$$

MANTISA : 1010001000

$$BROJ : ZNAK + EXP + MANTISA = 1100111010001000$$

$$1.5625 \cdot 10^{-1} = 0.15625 \cdot 10^0 = 0.15625 \cdot 1 =$$

$$\Rightarrow 0.15625 = 0,00101 = 1,01 \cdot 2^{-3}$$

$$0.15625 \cdot 2 = 0.3125$$

$$0.3125 \cdot 2 = 0.625$$

$$0.625 \cdot 2 = 1.25$$

$$0.25 \cdot 2 = 0.5$$

$$0.5 \cdot 2 = 1$$

ZNAK : + (+)

$$EXP : 15 + (-3) = 15 - 3 = 12 (01100)$$

$$\begin{array}{r} 0 & 1 & 1 & 0 & 0 \\ \hline 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ 16 & 8 & 4 & 2 & 1 \end{array}$$

MANTISA : 0100000000

~~$$BROJ : ZNAK + EXP + MANTISA = 0011000100000000$$~~

b) $-2.6125 \cdot 10^1 \cdot 1.5625 \cdot 10^{-1} = ?$

$$-2.6125 \cdot 10^1 = -1,1010001 \cdot 2^4$$

$$1.5625 \cdot 10^1 = 1,01 \cdot 2^{-3}$$

ZNAK : -

$$\begin{array}{r} 1,1010001 \\ * 1,0100000 \\ \hline 00000000 \\ 00000000 | \\ 00000000 | \\ 100000000 | \\ 111010001 | \\ 100000000 | \\ 11010001 \downarrow \downarrow \downarrow \downarrow \\ 10,00001010100000 \end{array}$$

$$G=0 \quad R=0 \quad S=0$$

$$\Rightarrow -1,00000101010 \cdot 2^1 =$$

$$\Rightarrow -2 \cdot 041015625 \cdot 2 = -4,08203125$$

$$-2,6125 \cdot 10^1 \cdot 1,5625 \cdot 10^1 = -4,08203125$$

② JUL 2021 : IEEE 754-2008 standard koristi 32 bita za snimanje celih brojeva. U formatu postoji bit za znak (prvi s leva), eksponent (8 bita, pri čemu je bias 127) i mantisa.

1 (10 bita). Podrazumeva se skrivena jedinica (hidden 1). Dati su brojevi: -6.325×10^{-15} i 48.75×10^{-1} .

a) Napisati binarnu reprezentaciju navedenih brojeva u datom standardu.

b) Izvršiti sabiranje navedenih brojeva. Za potrebe zaokruživanja se koriste 1 guard, 1 round i 1 sticky bit. Napisati binarnu reprezentaciju rezultata u datom standardu.

$$a) \text{ } \cancel{-6.325} \cdot 10^1 = -63.25 \cdot 10^0 = -63.25 \cdot 1 =$$

$$\Rightarrow -63.25 = -111111,01 = -1,1111101 \cdot 2^5$$

1	0	1	0	0	0			
1	1	1	1	1	1			
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1

$$0.25 \cdot 2 = 0.5$$

$$0.5 \cdot 2 = 1$$

zNAK: - (1)

EXP: $5+15 = 20$ (10100)

MANTISA: 1111101000

BROJ: zNAK + EXP + MANTISA = 1101001111101000

$$48.75 \cdot 10^1 = 4.875 \cdot 10^0 = 4.875 \cdot 1 = 4.875 =$$

$$\Rightarrow 100,111 = 1,00111 \cdot 2^2$$

$$0.875 \cdot 2 = 1.75$$

$$0.75 \cdot 2 = 1.5$$

$$0.5 \cdot 2 = 1$$

ZNAK : + (0)

EXP : $2 + 15 = 17$ (10001)

$$\begin{array}{r} 1 \quad 0 \quad 0 \quad 0 \quad 1 \\ \underline{2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0} \\ 16 \quad 8 \quad 4 \quad 2 \quad 1 \end{array}$$

MANTISA : 0011100000

BROJ : ZNAK + EXP + MANTISA ; 0100010011100000

b) $-6.325 \cdot 10^1 + 48.75 \cdot 10^1 = ?$

$$-6.325 \cdot 10^1 = -1,1111101 \cdot 2^5$$

$$48.75 \cdot 10^1 = 1,00111 \cdot 2^2 = 0,00100111 \cdot 2^5$$

ZNAK : -

$$1,11111010$$

$$\underline{-0,00100111}$$

$$1,11010011$$

$$G = 0 \quad R = 0 \quad S = 0$$

$$-1,11010011 \cdot 2^5 = -111010,011 \cdot 2^0 = -58,375$$

$$-63.25 + 4.875 = -58.375$$

② MART 2022 isto kao JUL 2021

1 ② SEPTEMBAR 2021: Dat je sledeći programski segment u jeziku C. Napisati RISC-V kod koji njemu odgovara. Pretpostavimo da se i, j, u, n i k nalaze u registruima $x_5, x_6, x_7, x_{28}, x_{29}$, respectivno, a početne adrese nizova A, B i C su u registruima x_{10}, x_{11} i x_{12} , respectivno.

```
for (i=0; i<n; i++)
for (k=0; k<u; k++)
C[i] = A[uj] + B[2k]
```

x_5 i

x_6 j

x_7 u

x_{28} n

x_{29} k

x_{10} A

x_{11} B

x_{12} C

Loop I :

addi $x_5, x_5, 0$ // $i=0$

bge $x_5, x_{28}, \text{exitI}$ // $i>n$ skoci na exitI

addi $x_{29}, x_{29}, 0$ // $k=0$

slli $x_{30}, x_5, 3$ // $i << 3$

add x_{12}, x_{12}, x_{30} // $\&C[i]$

Loop K :

bge $x_{29}, x_7, \text{exitK}$ // $k>u$ skoci na exitK

slli	$x_6, x_6, 3$	// $j \ll 3$
muli	$x_{31}, x_6, 4$	// $x_{31} = 4j$
add	x_{10}, x_{10}, x_{31}	// $\& A[4j]$
ld	$x_{13}, o(x_{10})$	// $x_{13} = A[4j]$
slli	$x_{14}, x_{29}, 3$	// $k \ll 3$
muli	$x_{15}, x_{14}, 2$	// $x_{15} = \cancel{x_{14}} \cdot 2$
add	x_{11}, x_{11}, x_{15}	// $\& B[2k]$
ld	$x_{16}, o(x_{11})$	// $B[2k] = x_{16}$
add	x_{17}, x_{13}, x_{16}	// $x_{17} = A[4j] + B[2k]$
sd	$x_{17}, o(x_{12})$	// $C[i] = A[4j] + B[2k]$
addi	$x_{29}, x_{29}, 1$	// $k++$
jal	x_0 , Loop K	

exit K :

addi	$x_5, x_5, 1$	// $i++$
jal	x_0 , Loop I	

② OKTOBAR 2021: Dat je sledeći programski segment u jesiku C. Napisati RISC-V kod koji njemu odgovara. Pretpostavimo da se i, j, u, n i k nalaze u registruima $x_5, x_6, x_7, x_{28}, x_{29}$, respektivno, a početne adrese nitova A, B i C su u registruima x_{10}, x_{11} i x_{12} , respektivno.

```

11
for (i=0; i<n; i++)
for (k=0; k<w; k++)
C[i] = A[j] + B[2]

```

x5	i
x6	j
x7	w
x28	n
x29	k
x10	A
x11	B
x12	C

Loop I :

```

addi x5, x5, 0    // i=0
bge x5, x28, exitI // i>n  skip next exitI
addi x29, x29, 0    // k=0
slli x30, x5, 3    // i<<3
muli x30, x30, 4    // x30 = 4i
add x12, x12, x30   // &C[4i]

```

Loop K :

```

bge x29, x7, exitK // k>w  skip next exitK
slli x6, x6, 3    // j << 3
add x10, x10, x6    // &A[j]
ld x31, 0(x10)    // x31 = A[j]
add x14, x31, x13   // &B[2]
sd x14, 0(x12)    // C[i] = A[j] + B[2]

```

addi $x_{29}, x_{29}, 1$ // $k++$

jal x_0 , LOOP K

exit K :

addi $x_5, x_5, 1$ // $i++$

jal x_0 , LOOP I

② APRIL 2022: Dat je sledeći programski segment u jeziku C. Napisati RISC-V kod koji njemu odgovara. Pretpostavimo da se i , j , w , n i k nalaze u registrua x_5, x_6, x_7, x_8, x_9 , respektivno, a početne adrese nizova A, B i C su u registrua x_{10}, x_{11} i x_{12} , respektivno.

for ($i=0; i < n; i++$)

for ($j = 0; j < w; j++$)

$C[i] = A[4j] + B[w]$

$x_5 \quad i$

$x_6 \quad j$

$x_7 \quad w$

$x_8 \quad n$

$x_9 \quad k$

$x_{10} \quad A$

$x_{11} \quad B$

$x_{12} \quad C$

! Loop I :

addi $x_5, x_5, 0$ // $i = 0$
bge $x_5, x_{28}, \text{exitI}$ // $i > n$ skoci na exitI
addi $x_6, x_6, 0$ // $j = 0$
slli $x_{29}, x_5, 3$ // $i \ll 3$
add x_{12}, x_{12}, x_{13} // $8C[i]$

! Loop J :

bge x_6, x_7, exitJ // $j > m$ skoci na exitJ
slli $x_{14}, x_6, 3$ // $j \ll 3$
muli $x_{15}, x_{14}, 4$ // $x_{15} = 4j$
add x_{10}, x_{10}, x_{15} // $8A[4j]$
ld $x_{16}, 0(x_{10})$ // $A[4j]$
slli $x_{29}, x_{29}, 3$ // $k \ll 3$
add x_{11}, x_{11}, x_{29} // $8B[k]$
ld $x_{30}, 0(x_{11})$ // $B[k]$
add x_{31}, x_{16}, x_{30} // $8C[i] = A[4j] + B[k]$
sd $x_{31}, 0(x_{12})$ // $C[i] = A[4j] + B[k]$
addi $x_6, x_6, 1$ // $j++$
jal x_0, loopJ

exitJ :

addi $x_5, x_5, 1$ // $i++$
jal x_0, loopI

(4.) APRIL 2021: Tabela 1 sadrži udeo instrukcija u programu. Tabela 2 sadrži učenjenja elemenata arhitekture RISC-V procesora, a tabela 3 cene elemenata.

TABELA 1: udeo

R-type / I-type	ld	sd	beq
45%	25 %	20 %	10 %

TABELA 2: učenjenja

I-MEM / D-MEM	RegisterFile	MUX	ALU	Adder	SingleGate
250	150	20	200	100	5
RegisterRead	RegisterSetup	SignExtend	Control		
30	25	45	40		

TABELA 3: cene

I-MEM / D-MEM	RegisterFile	MUX	ALU	Adder	SingleRegister
2000	200	10	100	40	5
SingleGate	SignExtend	control			
2	50	400			

- Odrediti:
- a) trajanje periode signala tanta.
 - b) Ako je moguće da svaka instrukcija ima svoj tant, odrediti ubrzanje ovakvog sistema u odnosu na slučaj gde je tant jedinstven za sve instrukcije.
 - c) Ako ALU jedinici dodamo unožić, to će joj

povećati kašnjenje za 150ps, ali će suanjiti broj instrukcija za 8%. Odrediti ubrzanje ... (ne vidim do veraja rečnicu).

d) Koliko uansualno kašnjenje može da ima ALU jedinica iz tache c) tako da ne dođe do degradiranja performansi?

a)

R-type:

$$\begin{aligned} & PC + I-MEM + RegFile + MUX + ALU + MUX + RegSetup = \\ & = 30 + 250 + 150 + 20 + 200 + 20 + 25 = 695 \text{ ps} \end{aligned}$$

* PC = RegisterRead *

I-type:

$$\Rightarrow R\text{-type} = 695 \text{ ps}$$

ld:

$$\begin{aligned} & PC + I-MEM + RegFile + MUX + ALU + D-MEM + MUX + RegSetup = \\ & = 30 + 250 + 150 + 20 + 200 + 250 + 20 + 25 = 945 \text{ ps} \end{aligned}$$

sd:

$$\begin{aligned} & PC + I-MEM + RegFile + MUX + ALU + D-MEM = \\ & \Rightarrow 30 + 250 + 150 + 20 + 200 + 250 = 900 \text{ ps} \end{aligned}$$

beq:

$$\begin{aligned} & PC + I-MEM + RegFile + MUX + ALU + SingleGate + MUX + RegSetup = \\ & \Rightarrow 30 + 250 + 150 + 20 + 200 + 5 + 20 + 25 = 700 \text{ ps} \end{aligned}$$

$$\Rightarrow 945 \text{ ps}$$

b) $0.45 \cdot 695 + 0.25 \cdot 945 + 0.2 \cdot 900 + 0.1 \cdot 700 =$
 $\Rightarrow 312.75 + 236.25 + 180 + 70 = 799 \text{ ps}$

$$\frac{945}{799} = 1,182 \rightarrow \text{ubrzanje}$$

* ubrzanje = minimalno trajanje periode signala
Σ udeo · trajanje periode signala

c) $945 \text{ n} \rightarrow \text{pre poboljšanja}$
 $100\% - 8\% = 92\% = 0.92$

$$945 + 150 = 1095$$

↪ dodavanje unožaca

$$1095 \text{ n} \cdot 0.92 = 1007,4 \text{ n} \rightarrow \text{posle}$$

$$\frac{1007,4 \text{ n}}{945 \text{ n}} = 1,066 \rightarrow \text{ubrzanje}$$

odakle ovo?

d) $945 \text{ n} = x \cdot 0,92 \text{ n}$

$$x = \frac{945}{0.92}$$

$$x = 1027$$

$$1027 - 945 = 82$$

$$\text{ALU} = 200 + 82 = 282$$

4. JUN 2021: Tabela 1 sadrži udeo instrukcija u programu. Tabela 2 sadrži učinkovanje elemenata arhitekture RISC-V procesora, a tabela 3 cene elemenata.

TABELA 1: Udeo

R-type / I-type	ld	sd	beq	
40%	30%	20%	10%	

TABELA 2: Učinkovanje

I-MEM/D-MEM	RegisterFile	MUX	ALU	Adder	SingleGate
200	150	25	200	100	5
RegisterRead	RegisterSetup	SignExtend	Control		
35	25	45	40		

TABELA 3: Cene

I-MEM/D-MEM	RegisterFile	MUX	ALU	Adder	SingleRegister
2000	200	10	100	40	5
SingleGate	SignExtend	Control			
2	50	400			

Određiti: a) Koji deo instrukcija koristi Data memoriju?

b) Minimalno trajanje tanta procesora.

c) Ako je moguće da svaka instrukcija ima svoj tant, odrediti ubrzanje okrug sistema u odnosu na slučaj gde je tant jedinstven za

sve instrucije -

a) ~~Load~~ Load i store.

$$\Rightarrow 30\% + 20\% = 50\%$$

b) R-type :

PC + I-MEM + RegFile + MUX + ALU + MUX + RegSetup =

$$\Rightarrow 35 + 200 + 150 + 25 + 200 + 25 + 25 = 660 \text{ ps}$$

I-type : 660ps

ld:

PC + I-MEM + RegFile + MUX + ALU + D-MEM + MUX + RegSetup =

$$\Rightarrow 35 + 200 + 150 + 25 + 200 + 200 + 25 + 25 = 860 \text{ ps}$$

sd:

PC + I-MEM + RegFile + MUX + ALU + D-MEM =

$$\Rightarrow 35 + 200 + 150 + 25 + 200 + 200 = 810 \text{ ps}$$

beg:

PC + I-MEM + RegFile + MUX + ALU + SingleGate + MUX + RegSetup =

$$\Rightarrow 35 + 200 + 150 + 25 + 200 + 5 + 25 + 25 = 665 \text{ ps}$$

$$\Rightarrow 860 \text{ ps}$$

c) $\rightarrow 860 \text{ ps}$

$$0.4 \cdot 660 + 0.3 \cdot 860 + 0.2 \cdot 810 + 0.1 \cdot 665 =$$

$$\Rightarrow 264 + 258 + 162 + 66.5 = 750.5$$

ubrzanje : $\frac{860}{750.5} = 1,145$

(4.) JUL 2021: Tabela 1 sadrži udeo instrukcija u programu. Tabela 2 sadrži kašnjenja elemenata arhitekture RISC-V procesora, a tabela 3 cene elemenata.

TABELA 1: udeo

R1-type / I-type	ld	sd	beq
45 %	25 %	20 %	10 %

TABELA 2: kašnjenja

I-MEM/D-MEM	RegisterFile	MUX	ALU	Adder	Single Gate
200	170	30	150	100	5
Register Read	Register Setup	Sign Extend	Control		
35	25	45	40		

TABELA 3: cene

I-MEM/D-MEM	RegisterFile	MUX	ALU	Adder	Single Register	Single Gate
2000	200	10	100	40	5	2
Sign Extend	Control					
50	400					

Odrediti: a) Koji deo instrukcija koristi Sign extend jedinicu?

b) Sump vrednosti signala koje generise kontrolna jedinica u slucaju instrukcije add.

c) Kaznjene I-type instrukcija.

a) load/store, I-type, branch

b) branch - false

mem Read - false

mem To Reg - 0

ALUOp - "add"

mem Write - false

ALUSrc - 1

reg Write - true

c) I-type:

PC + I-MEM + RegFile + MUX + ALU + MUX + Reg Setup =

$$\Rightarrow 35 + 200 + 170 + 30 + 150 + 30 + 25 = 640 \text{ ps}$$

4. SEPTEMBAR 2021: Tabela 1 sadrzi udeo instrukcija u programu. Tabela 2 sadrzi kaznjena elemenata arhitekture RISC-V procesora, a tabela 3 cene elemenata.

TABELA 1: udeo

R-type	I-type	ld	sd	beq	
25%	20%	25%	20%	10%	

- Odrediti:
- Sup vrednosti signala koje generiše kontrolna jedinica u slučaju instrucije (d.)
 - Minimalnu vrednost signala tanta.
 - Ako je moguće da svaka instrucija ima poseban tant, odrediti koliko je ubrzanje u odnosu na sistem sa jedinstvenim tantom.

TABELA 2: kašnjenja

I-MEM/D-MEM	Register File	MUX	ALU	Adder	SingleGate
200	160	25	150	100	5
Register Read	Register Setup	SignExtend	Control		
35	30	45	30		

TABELA 3: cene

I-MEM/D-MEM	Register File	MUX	ALU	Adder	Single Register
2000	250	15	100	40	10
SingleGate	SignExtend	Control			
5	50	400			

a) branch - false

memRead - true

memToReg - 1

ALU Op - "ld"

memWrite - false

ALU Src - 1

Reg Write - true

b) R-type:

PC + I-MEM + RegFile + MUX + ALU + MUX + Reg Setup =

$$\Rightarrow 35 + 200 + 160 + 25 + 150 + 25 + 30 = 625 \text{ ps}$$

I-type: 625 ps

Pd:

PC + I-MEM + RegFile + MUX + ALU + D-MEM + MUX + Reg Setup =

$$\Rightarrow 35 + 200 + 160 + 25 + 150 + 200 + 25 + 30 = 825 \text{ ps}$$

Sd:

PC + I-MEM + RegFile + MUX + ALU + D-MEM =

$$\Rightarrow 35 + 200 + 160 + 25 + 150 + 200 = 770 \text{ ps}$$

beq:

PC + I-MEM + RegFile + MUX + ALU + Single Gate + MUX + Reg Setup =

$$\Rightarrow 35 + 200 + 160 + 25 + 150 + 5 + 25 + 30 = 630 \text{ ps}$$

$$\Rightarrow 825 \text{ ps}$$

c) $0.45 \cdot 625 + 0.25 \cdot 825 + 0.2 \cdot 770 + 0.1 \cdot 630 =$

$$\Rightarrow 281.25 + 206.25 + 154 + 63 = 704.5$$

Ubrzanje: $\frac{825}{704.5} = 1,171$

4. OKTOBAR 2021: Tabela 1 sadrži udeo instrukcija u programu. Tabela 2 sadrži kašnjenja elemenata arhitekture RISC-V procesora, a tabela 3 cene elemenata. Pretpostavimo da želimo da poboljšamo osnovnu RISC-V arhitekturu tako što joj dodajemo kolo za deljenje. To će smanjiti broj instrukcija za 20%, ali će povećati kašnjenje ALU za 50 ps.

TABELA 1: Udeo

R-type	I-type	ld	sd	branch / jump
25%	20%	25%	20%	10%

TABELA 2: Kašnjenja

I-MEM / D-MEM	RegisterFile	MUX	ALU	Adder	Single Gate
200	160	25	150	100	5
RegisterRead	Register Setup	Sign Extend	Control		
35	30	45	30		

TABELA 3: Cene

I-MEM / D-MEM	RegisterFile	MUX	ALU	Adder	Single Register
2000	250	15	100	40	10
Single Gate	Sign Extend	Control			
5	50	400			

Odrediti : a) Trajanje signala tanta pre i nakon poboljšanja.

b) Ubrzanje nakon poboljšanja.
c) Ako je cena dodatog kola za deljenje 200,
odrediti odnos performanse/cena.

a) R-type:

$$PC + I-MEM + Reg\ File + MUX + ALU + MUX + Reg\ Setup = \\ \Rightarrow 35 + 200 + 160 + 25 + 150 + 25 + 30 = 625\ ps$$

I-type: 625 ps

ρ_d :

$$PC + I-MEM + Reg\ File + MUX + ALU + D-MEM + MUX + Reg\ Setup = \\ \Rightarrow 35 + 200 + 160 + 25 + 150 + 200 + 25 + 30 = 825\ ps$$

ρ_d :

$$PC + I-MEM + Reg\ File + MUX + ALU + D-MEM = \\ \Rightarrow 35 + 200 + 160 + 25 + 150 + 200 = 770\ ps$$

beg:

$$PC + I-MEM + Reg\ File + MUX + ALU + Single\ Gate + MUX + Reg\ Setup = \\ \Rightarrow 35 + 200 + 160 + 25 + 150 + 5 + 25 + 30 = 630\ ps$$

pre poboljšanja: 825 ps

posle poboljšanja: $825 + 50 = 875\ ps$

↪ dodatno uobičajenje
za ALU

I(b) $825n \rightarrow$ pre poboljšanja

$$1 - 0.2 = 0.8$$

$875n \cdot 0.8 = 700n \rightarrow$ posle poboljšanja
odakle?

$$\frac{825n}{700n} = 1,17$$

→ ništa uobičajeno ovde nije jasno.

c) I-MEM = 2000

D-MEM = 2000

Register File = 250

MUX = $15 \cdot 3 = 45$ ↗?

ALU = 100

Adder : $40 \cdot 2 = 80$

Single Register = 10

Single Gate = 5

SignExtend = 50

Control = $400 \cdot 2 = 800$

cena CPU = $2000 + 2000 + 250 + 45 + 100 + 80 + 10 + 5 + 50 + 800 \Rightarrow 5340$

nova cena = cena CPU + 200 = $5340 + 200 = 5540$

$$\frac{5540}{5340} = 1,037$$

cena : 3.7%

performanse : 17% } ?

(4.) OKTOBAR II isti kao (4.) OKTOBAR

(4.) JANUAR 2022: Tabela 1 sadrži udeo instrucija u programu. Tabela 2 sadrži klasnjenja elemenata arhitekture RISC-U procesora, a tabela 3 cene elemenata.

TABELA 1: Udeo

R/I-type / I-type	ld	sd	beq	
45%	25%	10%	10%	

TABELA 2: Klasnjenja

I-MEM / D-MEM	RegisterFile	MUX	ALU	Adder	Single Gate
200	170	30	150	100	5
RegisterRead	Register Setup	Sign Extend	Control		
35	25	45	40		

TABELA 3: cene

I-MEM / D-MEM	RegisterFile	MUX	ALU	Adder	Single Register
2000	200	10	100	40	5
Single Gate	Sign Extend	Control			
2	50	400			

Odrediti: a) Koji deo instrucija koristi Data memoriju?

b) Skup vrednosti signala koje generiše kontrolna

jedinica u slučaju instrukcije or.

c) Minimalnu vrednost periode signala tanta.

a) load ; store

$$25\% + 20\% = 45\%$$

b) branch - false

memRead - false

memToReg - false (0)

ALU Op - „or“

memWrite - false

ALU Src - 1

regWrite - true

c) R-type:

PC + I-MEM + RegFile + MUX + ALU + MUX + RegSetup =

$$\Rightarrow 35 + 200 + 170 + 30 + 150 + 30 + 25 = 640 \text{ ps}$$

I-type: 640 ps

ld:

PC + I-MEM + RegFile + MUX + ALU + D-MEM + MUX + RegSetup =

$$\Rightarrow 35 + 200 + 170 + 30 + 150 + 200 + 30 + 25 = 840 \text{ ps}$$

sd:

PC + I-MEM + RegFile + MUX + ALU + D-MEM =

$$\Rightarrow 35 + 200 + 170 + 30 + \underline{150} + 200 = 785 \text{ ps}$$

bez:

$$\text{PC} + \text{I-MEM} + \text{RegFile} + \text{MUX} + \text{ALU} + \text{Single Gate} + \text{MUX} + \text{Reg Setup} = \\ \Rightarrow 35 + 200 + 170 + 30 + 150 + 5 + 30 + 25 = 645 \text{ ps}$$

$$\Rightarrow 840 \text{ ps}$$

(4.) MART 2022: Tabela 1 sadrži udeo instrukcija u programu. Tabela 2 sadrži kasnjenja elemenata arhitekture RISC-V procesora, a tabela 3 cene elemenata.

TABELA 1: Udeo

R-type / I-type	Pd	sd	bed	
45%	25%	20%	10%	

TABELA 2: kasnjenja

I-MEM / D-MEM	Register File	MUX	ALU	Adder	Single Gate
200	170	30	150	100	5
Register Read	Register Setup	Sign Extend	Control		
35	25	45	40		

TABELA 3: cene

I-MEM / D-MEM	Register File	MUX	ALU	Adder	Single Register
2000	200	10	100	40	5
Single Gate	Sign Extend	Control			
2	50	400			

I Odrediti: a) Koji deo instrukcija koristi Instrukcionu memoriju?

b) Skup vrednosti signala koje generise kontrolna jedinica u slucaju instrukcije ld.

c) Ako je moguce da svaka instrukcija ima svoj tacut, odrediti ubrzanje ovakvog sistema u odnosu na slucaj gde je tacut jedinstven za sve instrukcije.

a) R-type, I-type, ld, sd, beg

b) branch - false

memRead - true

memToReg - 1

ALUOp - "ld"

memWrite - false

ALUSrc - 1

regWrite - true

c) R-type:

PC + I-MEM + RegFile + MUX + ALU + MUX + Reg Setup =

=> 35 + 200 + 170 + 30 + 150 + 30 + 25 = 640 ps

I-type : 640 ps

Pd:

PC + I-MEM + RegFile + MUX + ALU + D-MEM + MUX + Reg Setup =

$$\Rightarrow 35 + 200 + 170 + 30 + 150 + 200 + 30 + 25 = 840 \text{ ps}$$

sd:

$$PC + I-MEM + Reg\ File + MUX + ALU + D-MEM =$$

$$\Rightarrow 35 + 200 + 170 + 30 + 150 + 200 = 785 \text{ ps}$$

beq:

$$PC + I-MEM + Reg\ File + MUX + ALU + Single\ Gate + MUX + Reg\ Setup =$$

$$\Rightarrow 35 + 200 + 170 + 30 + 150 + 5 + 30 + 25 = 645 \text{ ps}$$

$$\Rightarrow 840 \text{ ps}$$

~~scribble~~

$$0.45 \cdot 640 + 0.25 \cdot 840 + 0.2 \cdot 785 + 0.1 \cdot 645 =$$

$$\Rightarrow 288 + 210 + 157 + 64.5 = 719.5$$

$$\text{ubrzanje: } \frac{840}{719.5} = 1,167$$

④ APRIL 2022. isbi kao ④ APRIL 2021.

⑤ APRIL 2021: Za dati programski segment
napraviti 2 multicycle dijagrama, za slučaj
kada postoji i kada ne postoji mehanizam

I prosleđivanja (forwarding).

sd $x_7, 20(x_8)$

ld $x_7, 0(x_5)$

and x_6, x_1, x_7

or x_8, x_7, x_9

sd $x_8, 100(x_2)$

II BEZ PROSLEDIVANJA:

IF ID EX MEM WB \rightarrow UVEK

IF ID EX MEM WB

IF ID STALL STALL EX MEM WB

IF ID STALL STALL EX MEM WB

IF ID STALL STALL STALL EX MEM WB

* Kod ld instrukcije, execute se ~~moraju~~ izvršiti
tek nakon WB faze. *

* Kod svih ostalih instrukcija, execute se ~~moraju~~ izvršiti
tek nakon MEM faze. *

* Ovo važi u situaciji bez prosleđivanja. *

SA PROSLEDIVANJEM:

IF ID EX MEM WB → Uvek

IF ID EX MEM WB

IF ID STALL → EX MEM WB

IF ID STALL EX MEM WB

IF ID STALL → EX MEM WB

* Kod ld instrukcije, execute se može izvršiti tek nakon MEM faze.
 $MEM \rightarrow EX$

*

* Kod ostalih instrukcija, execute se može izvršiti tek nakon EX faze.
 $EX \rightarrow EX$

*

5. JUN 2021: Za dati programski segment napraviti 2 multicycle dijagrama, za slučaj kada postoji i kada ne postoji mehanizam prosledivanja (forwarding).

Pd $x_7, 10 (x_8)$

Sd $x_7, 0 (x_5)$

and x_6, x_1, x_7

or x_8, x_7, x_6

Sd $x_8, 100 (x_2)$

bez prepletivanja :

IF ID EX MEM WB → UVEK

IF ID STALL STALL EX MEM WB

IF ID STALL STALL EX MEM WB

IF ID STALL STALL STALL EX MEM WB

IF ID STALL STALL STALL EX MEM WB

sa prepletanjem :

IF ID EX MEM WB → UVEK

IF ID STALL EX MEM WB

⑤ JUL 2021: Za dati programski segment
napraviti 2 multicycle dijagrama, za slučaj kad
postoji i kad ne postoji mehanizam prepleti-
vanja (forwarding).

sd X7, 0(x5)

ld X7, 20(r8)

and X7, X1, X6

or X8, X7, X6

sd X8, 100(x2)

bez presečivanja:

IF ID EX MEM WB → uvek

IF ID EX MEM WB

IF ID EX MEM WB

IF ID STALL EX MEM WB

IF ID STALL STALL EX MEM WB

sa presečivanjem:

IF ID EX MEM WB → uvek

IF ID EX MEM WB

IF ID EX MEM WB

IF ID STALL → EX MEM WB

IF ID STALL → EX MEM WB

⑤ SEPTEMBAR 2021: Za dati programski segment napraviti 2 multicycle dijagrama, za slučaj kad postoji i kad ne postoji mehanizam presečivanja (forwarding).

sd x6, 0(x5)

and x7, x6, x5

ld v7, 10 (x8)

and x6, x7, x5

or v8, x7, x6

sd x8, 100 (x2)

I bez prosleđivanja:

IF ID EX MEM WB → UVEK

IF ID EX MEM WB

IF ID EX MEM WB

IF ID STALL STALL EX MEM WB

IF ID STALL STALL STALL EX MEM WB

IF ID STALL STALL STALL EX MEM WB

sa prosleđivanjem:

IF ID EX MEM WB → UVEK

IF ID EX MEM WB

IF ID EX MEM WB

IF ID STALL EX MEM WB

IF ID STALL EX MEM WB

IF ID STALL EX MEM WB

⑤ OKTOBAR 2021: Za dati programski segment napraviti 2 multicycle dijagrafu, za slučaj kad postoji i kad ne postoji mehanizam prosleđivanja (forwarding).

ld x1, (x2)0

sub x4, y1, x5

and x6, x1, x7

or x8, x6, x9

sd x15, 100 (x2)

bez prosleđivanja:

IF ID EX MEM WB → UVEK

IF ID STALL STALL EX MEM WB

IF ID STALL STALL EX MEM WB

IF ID STALL STALL STALL EX MEM WB

IF ID STALL STALL STALL EX MEM WB

sa prosleđivanjem:

IF ID EX MEM WB → UVEK

IF ID STALL EX MEM WB

⑤ MART 2022. isti kao ⑤ JUN 2021.

⑤ OKTOBAR II : Tabela 4 sadrži kašnjenja pojedinih faz u pipeline-a.

IF	ID	EX	MEM	WB	
250	350	200	300	250	

Odrediti: a) Trajanje signala taktu u osnovnoj i pipeline arhitekturi.

b) Pretpostavimo da pipeline procesor dodaje NOP instrucije u cilju eliminacije hazarda podataka

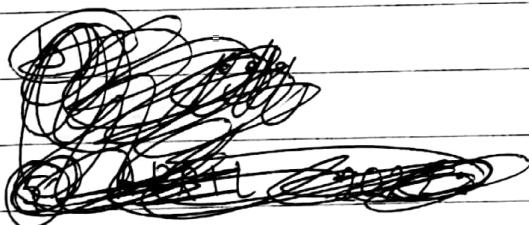
I (0.3n dodatnih NOP instrukcija za program
dužine n). Dodavanjem hardvera za prosledi-
vanje, broj dodatnih NOP instrukcija se smanji
na 0.04n, ali se trajanje signala tanta
poveća za 50ps. Koliko je ubrzanje koje se
dobija dodavanjem ponenutog hardvera?

a) osnovna arhitektura:

$$250 + 350 + 200 + 300 + 250 = 1350$$

pipeline arhitektura :

$$(\text{najduža faza}) \Rightarrow 350$$



b)

$$0.3n + n = 1.3n$$

$$1.3n \cdot 350 = 455n$$

$$350 + 50 = 400$$

$$0.04n + n = 1.04n$$

$$1.04n \cdot 400 = 416n$$

Ubrzanje: $\frac{416n}{400n} = 1.04$

(5.) APRIL 2022: Tabela 4 sadrži kašnjenja pojedinih faza pipeline-a.

IF	ID	EX	MEM	WB
200	350	200	360	250

Odrediti: a) trajanje signala tanka u osnovnoj i pipeline arhitekturi.
 b) Pretpostavimo da pipeline procesor dodaje NOP instrukcije u cilju eliminacije hazarda podataka ($0.25n$ dodatnih NOP instrukcija za program dužine n). Dodavanjem hardvera za preleđivanje, broj dodatnih NOP instrukcija se smanji 60% , ali se trajanje signala tanka poveća za 50ps . Da li se i koliko ubrzanje dobija dodavanjem ponenutog hardvera?

a) osnovna arhitektura:

$$\Rightarrow 200 + 350 + 200 + 360 + 250 = 1360$$

pipeline arhitektura:
 (najduža faza) $\Rightarrow 360$

b)

$$0.25n + n = 1.25n$$

$$1.25n \cdot 360 = 450n$$

$$360 + 50 = 410$$

$$100\% - 60\% = 40\% \Rightarrow 0.4$$

$$0.4n + n = 1.4n$$

$$1.4n \cdot 410 = 574n$$

ubrztanje : $\frac{574n}{450n} = 1.275$