

# Mikroprocesorska elektronika - Projektovanje mikroprocesora

prof. dr Ivan Mezei

22. okt. 2021.



## Glava 3

# Metodologija, principi i primeri projektovanja mikroprocesora

U ovom poglavlju će biti prikazani metodologija i principi projektovanja mikroprocesora, kao i konkretni primeri projektovanja jednostavnih edukacionih mikroprocesora. Osnovni cilj je bolje razumevanja principa rada mikroprocesora kao ključnog dela svakog mikroračunara ili embeded sistema. Ovde se ograničavamo samo na rad sa edukacionim mikroprocesorima. Metodologija i projektovanje složenih, pa i komercijalnih, mikroprocesora daleko prevazilazi okvire ovog predmeta.

U skladu sa podelom mikroprocesora sa stanovišta načina i vremena izvršavanja na jednotaktne, višetaktne i mikroprocesore sa protočnom obradom, u okviru ovog poglavlja ćemo podrazumevati isključivo višetaktne mikroprocesore. Drugim rečima, mikroprocesore koji izvršavaju instrukcije u više taktova. Više reči o funkcionisanju jednotaktnih i mikroprocesora sa protočnom obradom će biti u poglavlju ??.

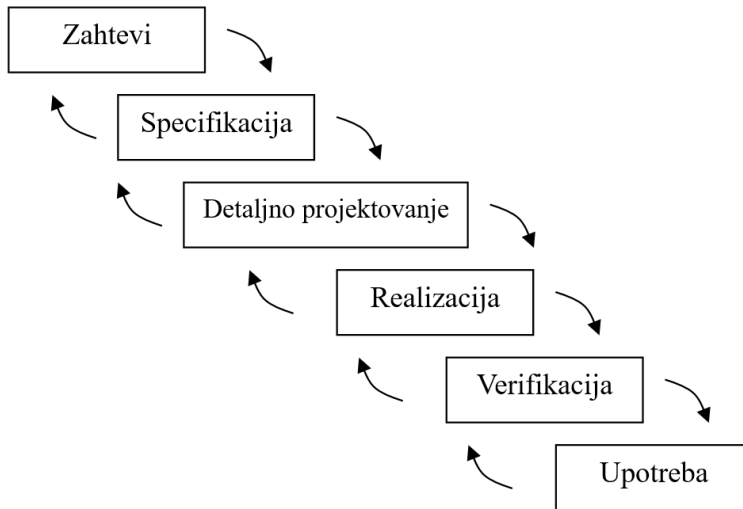
### 3.1 Metodologija projektovanja CPU

Ukoliko se posmatra model vodopada<sup>1</sup> [13] predstavljen na slici 3.1, on pokazuje sled događaja tj. faze u projektovanju mikroprocesora. U prvoj fazi se definišu

---

<sup>1</sup>Ovo je prevod engleskog termina *waterfall* po kome je ovaj model originalno nazvan.

**zahtevi** koji se postavljaju pred mikroprocesor koji se projektuje. U tom smislu je neophodno definisati hardverski i softverski skup komponenata: arhitekturu, skup instrukcija i načine adresiranja. Na ovaj način će biti definisan strukturalni model mikroprocesora koji uključuje tokove podataka (eng. *datapath*) i koji se naziva i mikroarhitektura, a kada se specificira i skup instrukcija i načini adresiranja imamo *Instruction Set Architecture* (ISA) koja predstavlja skup hardverskih i softverskih komponenata CPU. Ovaj model preciznije definiše zahteve koji se postavljaju pred mikroprocesor.



Slika 3.1: Model vodopada

Druga faza, koja sledi posle definisanja zahteva, je faza **specifikacije**. Za potrebe specifikacije mikroprocesora najbolje je usvojiti neki formalizam za opis (npr. RTL, ISP, RTN, LISA i dr.). Za potrebe ovog predmeta odabran je RTN (eng. *Register Transfer Notation*) jezik [14] pomoću kojeg će biti definisani apstraktan i konkretan model o čemu će biti reči kasnije. Konkretan model, između ostalog, ima za cilj da olakša detaljno projektovanje.

U fazi **detaljnog projektovanja** je potrebno projektovati sve blokove unutar mikroprocesora (registre, ALJ, multipleksere i sl.). Ovi blokovi se potom mogu **realizovati** koristeći konkretan opis u nekom od jezika za opis i modelovanje hardvera (eng. *Hardware description languages* - VHDL, Verilog, SystemVerilog i sl.).

U fazi **verifikacije** je potrebno verifikovati sve prethodno realizovane blokove unutar mikroprocesora (registre, ALJ, multipleksere i sl.) kao i ceo mikroprocesor. Ovo se najčešće radi u nekom od jezika za verifikaciju ili jezika koji to podržavaju (e, VHDL, Verilog, SystemVerilog i sl.). Pored toga se koriste i odgovarajući test programi da bi se utvrdila funkcionalna ispravnost.

U okviru ovog predmeta ćemo se baviti principima projektovanja mikroprocesora zaključno sa detaljnim projektovanjem sa ciljem njihovog boljeg razumevanja. Procesi konkretne realizacije, verifikacije i implementacije mikroprocesora na nekom npr. konkretnom programabilnom razvojnom sistemu, što bi spadalo u fazu upotrebe, će biti teme predmeta na višim godinama studija.

## 3.2 Zahtevi

U fazi zahteva kod projektovanja CPU-a neophodno je definisati njegov skup instrukcija kao i način na koji će se instrukcije kodirati i izvršavati, a spregnuto sa elementima tokova podataka (eng. *datapath*, mikroarhitektura) da bismo time zaokružili arhitekturu CPU-a koju ćemo nazvati strukturalni model. Ovo se u engleskoj literaturi naziva i *Instruction Set Architecture* (ISA) [8].

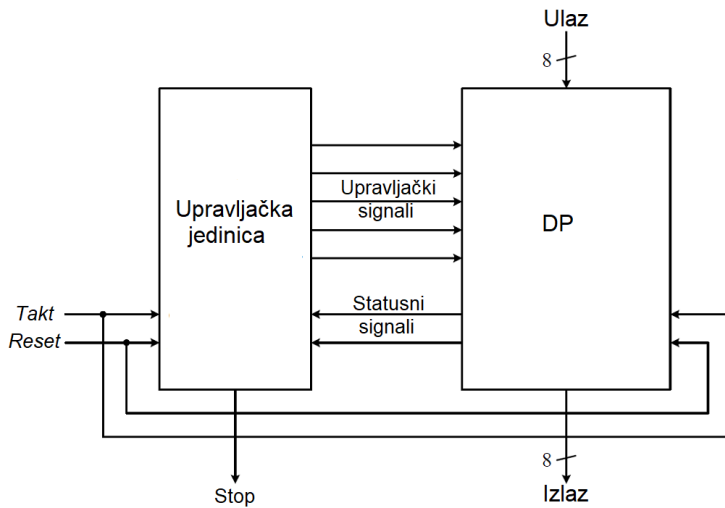
Drugim rečima, neophodno je dati odgovor na sledeća pitanja u vezi skupa instrukcija:

1. Koliko instrukcija želimo da postoji u skupu instrukcija?
2. Koje će to instrukcije biti?
3. Koje grupe instrukcija postoje?
4. Kakav binarni kôd (kodiranje se naziva operacioni kôd ili opkôd) će se dodeliti svakoj instrukciji?
5. Kakav će biti format instrukcija, tj. koliko bitova će se koristiti za kodiranje instrukcije, a koliko bitova za ostala polja i koja je njihova funkcija?
6. Koje načine adresiranja primeniti?

Nakon što smo odlučili i specificirali kakav će biti skup instrukcija, potrebno je nakon toga da se nastavi sa projektovanjem blokova za tokove i obradu podataka gde će se izvršavati sve instrukcije iz skupa instrukcija. U ovom koraku je neophodno dati i odgovore na sledeća pitanja:

7. Koje su funkcionalne jedinice potrebne?

8. Koliko registara je potrebno i kog tipa?
9. Da li treba koristiti jedinstveno registarsko polje (koncept koji se naziva na engleskom *register file*) ili posebne registre?
10. Kako su različite jedinice međusobno povezane?
11. Koliko magistrala će povezivati tokove podataka?
12. Kakav će biti pristup memoriji i memorijski model?
13. Na koji način realizovati upravljačku jedinicu?



Slika 3.2: Generalna struktura mikroprocesora opšte namene

Na slici 3.2 prikazana je generalna struktura mikroprocesora opšte namene koji se sastoji iz ulaza upravljačke jedinice, DP bloka, internih signala i izlaza. Projektovanje bloka za tokove i obradu podataka (DP blok) za mikroprocesor treba da obezbedi da pored toga što mikroprocesor treba da bude u stanju da obavi sve instrukcije iz skupa instrukcija, omogućí izvršavanje i drugih operacija koje manipulišu podacima i registrima. Operacije nad podacima i registrima tiču se toga kako će mikroprocesor zahvatati instrukcije iz memorije i izvršavati ih. U prethodnom poglavlju u sekciji o registrima su izdvojeni pojedini registri posebne ili

specijalne namene (koji su unutar DP bloka) koji tome služe. Programski brojač (PC) čuva adresu memorijske lokacije gde se nalazi naredna instrukcija u programu. Instrukcioni registar (IR) se koristi za memorisanje zahvaćene instrukcije iz memorije. Svaki put kada se instrukcija zahvati sa memorijske lokacije na koju pokazuje PC, PC se uvećava kako bi ukazao na narednu memorijsku lokaciju na kojoj se nalazi sledeća instrukcija ili operand. Alternativno, ako je instrukcija tipa grananja, u PC mora da se upiše vrednost memorijske adrese od koje program dalje nastavlja izvršavanje. Ovi specijalni registri nisu dostupni programeru, ali su od vitalnog značaja za funkcionisanje mikroprocesora.

Upravljačka jedinica mikroprocesora je svakako jedan od važnijih i složenijih unutrašnjih blokova CPU. Ona u suštini izvršava sledeća tri uobičajena koraka, poznata kao ciklus izvršavanja instrukcije:

Korak 1 - zahvat instrukcije,

Korak 2 - dekodiranje instrukcije, i

Korak 3 - izvršenje instrukcije.

Ciklus izvršavanja instrukcija je objašnjen u prethodnom poglavlju. Ovde je neophodno naglasiti samo da je bitno opredeliti se za način realizacije upravljačke jedinice.

Da bismo ilustrovali primer zahteva za neki konkretan mikroprocesor poći ćemo od najjednostavnijeg mogućeg, ovde iznačen kao mCPU0.

### 3.2.1 Zahtevi mCPU0

Mikroprocesor<sup>2</sup> mCPU0 je najjednostavniji moguć i izuzetno ograničenih mogućnosti, ali je pogodan za prvu ilustraciju ručnim projektovanjem. Sa ciljem da se očuva mogućnost ručnog projektovanja mikroprocesora, neophodno je da:

- broj instrukcija bude najmanji moguć,
- instrukcije budu što jednostavnije, i
- funkcionalne jedinice i arhitektura CPU bude što jednostavnija.

U skladu sa zahtevom da mCPU0 ima minimalno moguć skup instrukcija, on ima ukupno četiri instrukcije. To su po jedna instrukcija za: ulaz podataka, izlaz podataka, aritmetičku operaciju oduzimanja jedinice i kraj programa. Mana ovog skupa instrukcija je nemogućnost realizacije grananja što ćemo nadograditi kod mCPU1 i ostalim mikroprocesorima u sekciji Primeri projektovanja CPU. Skup

---

<sup>2</sup>Verovatno bi prikladniji termin bio 'nanoprocessor'.

instrukcija, uključujući njihove nazive kodovanje i objašnjenje, je prikazan u narednoj tabeli. Znak x na nekim pozicijama označava da vrednost nije bitna. Svaka instrukcija se sastoji iz mnemonika i iz operanda, ako ga ima. Dakle mnemonici su IN, SUB, OUT i END, a samo instrukcija SUB ima operande. To su akumulator A i broj 1. IN instrukcija služi da se sa Ulaza učitava 8-bitna vrednost u akumulator, SUB instrukcija vrši fiksno oduzimanje za 1, OUT instrukcija nema neku posebnu ulogu jer je izlaz akumulatora svakako povezan na Izlaz, a END instrukcija završava rad programa i ostaje u tom stanju do pojave reseta.

Tabela 3.1: Skup instrukcija mikroprocesora mCPU0

Instrukcija	Kodovana vrednost	Objašnjenje
IN	000xxxxx	Unos podatka u akumulator.
SUB A, 1	001xxxxx	Oduzimanje sadržaja akumulatora za 1.
OUT	010xxxxx	Izlazna vrednost (iz akumulatora).
END	111xxxxx	Kraj programa.

Memorija je veličine 16x8 (16 reči veličine 8 bita) i moguće je samo čitati sadržaj svake lokacije postavkom odgovarajuće 4-bitne adrese na adresni ulaz (ROM tip memorije). Od registara je minimalno potrebno: akumulator (A), programski brojč (PC) i instrukcioni registar (IR). Svi registri su 8-bitni, a pored ulaza i izlaza potrebni su i signal dozvole upisa, reset i takt. Za izbor da li upis u akumulator ide spolja ili sa izlaza ALJ potreban je jedan multiplexer 2-na-1. Upravljačka jedinica realizuje 4 upravljačka signala i u ovom slučaju je moguća ožičena realizacija ili preko malog automata. Oznake upravljačkih signala su: Aload, INmux, PCload i IRload. Arhitektura mCPU0 je prikazana na slici 3.3. Na ovaj način su dati odgovori na svih 13 pitanja vezanih za zahteve koji se postavljaju pred CPU pa se može preći na njegovu specifikaciju.

### 3.3 Specifikacija

U fazi specifikacije se radi specifikacija apstraktnog modela mikroprocesora u RTN (eng. *Register Transfer Notation*) jeziku (ili nekom sličnom) [14]. Naziv apstraktan model potiče od činjenice da ovako realizovan model nije povezan sa konkretnom hardverskom realizacijom i ne uključuje vreme (takt). Apstraktan model definiše statičke resurse (registri, memorija) i dinamičke operacije među njima (instrukcije). Osnovna pravila RTN jezika su prikazana u tabeli 3.2.





Tabela 3.2: Osnovna pravila RTN jezika

Element RTN jezika	Objašnjenje
ime_registra <veličina podataka - 1 .. 0>	opis registra
M[0 .. kapacitet] <veličina podataka - 1 .. 0>	opis memorije
$\rightarrow$	oznaka za «sledi»
$\leftarrow$	oznaka za «dobija vrednost»
$:=$	dodela vrednosti
$=$	provera jednakosti
:	oznaka istovremenih operacija
$+$	operator sabiranja
$-$	operator oduzimanja
$\neg$	operator logičke negacije
$\wedge$	operator logičke I operacije
$\vee$	operator logičke ILI operacije
$\oplus$	operator logičke EX-ILI operacije

Tabela 3.3: Specifikacija delova mCPU0

Deo CPU	Objašnjenje
PC<3..0>	programski brojač
IR<7..0>	instrukcioni registar
A<7..0>	akumulator
M[0..2 <sup>4</sup> - 1] < 7..0 >	memorija
op<2..0> := IR<7..5>	kôd operacije

SUB A, 1 ( $:= op = 1$ )  $\rightarrow (A \leftarrow A - 1)$

OUT ( $:= op = 2$ )  $\rightarrow (A \rightarrow Izlaz)$

END ( $:= op = 7$ )  $\rightarrow IR \leftarrow 111xxxx$

Analizom potreba za upravljačkim signalima dobija se da je pored sistemskog reseta potrebno još četiri upravljačka signala. U tabeli 3.4 je dat spisak svih upravljačkih signala i kratko objašnjenje signala.

Tabela 3.4: Upravljački signali mCPU0

Signal	Objašnjenje
IRload	Upis u instrukcioni registar
PCload	Upis u programski brojač
INmux	Određivanje operanda za upis u akumulator
Aload	Upis u akumulator

### 3.4 Detaljno projektovanje

U ovoj fazi je neophodno detaljno projektovati sve elemente mikroprocesora koji su postavljeni u fazi zahteva i fazi specifikacije. Da bi to moglo da se uradi neophodno je uraditi specifikaciju konkretnog modela instrukcija koji će sadržati kompletan skup mikroinstrukcija neophodan da bi se izvršila svaka instrukcija. Drugim rečima koji će uključiti vreme u specifikaciju tj. takt. Potom je potrebno uraditi proširenje konkretnog modela upravljačkim signalima. Na osnovu ovoga je moguće projektovati upravljačku jedinicu.

Konkretni RTN opis predstavlja detaljniju specifikaciju modela mikroračunara i pre svega je korišćen za detaljan opis faza izvršavanja svih instrukcija. Svaka instrukcija se sastoji od niza mikroinstrukcija, a vreme njenog izvršavanja izraženo je u periodama takta mikroprocesora. Na osnovu ovakvog modela, koji uključuje i vreme, moguće je jednoznačno definisati upravljačke signale jer se ceo sistem ponaša sinhrono u odnosu na takt. Ovako definisana specifikacija predstavlja bazu za realizaciju hardverskih komponenti mikroračunara kao i pojedinih delova programske podrške. U nastavku će biti dat primer detaljnog projektovanja mCPU0.

#### 3.4.1 Detaljno projektovanje mCPU0

Svaka instrukcija počinje fazom zahvata instrukcije. Konkretna RTN opis ove faze se može uraditi na osnovu apstraktnog RTN opisa iz prethodne sekcije:

Takt	RTN
T0	$IR \leftarrow M[PC]$

Nakon faze zahvata instrukcije, sledi faza dekodovanja. Kod mCPU0 se u ovoj fazi sadržaj programskog brojača uvećava za jedan:

Takt	RTN
T1	$PC \leftarrow PC + 1$

Nakon faze zahvata, i faze dekodovanja, koje ukupno traju dve periode takta, sledi faza izvršavanja instrukcije. Faza izvršavanja instrukcije kod mCPU0 traje jednu periodu takta. U narednoj tabeli ilustrovani su konkretni RTN modeli svih instrukcija.

Takt	RTN: IN
T0	Zahvat instrukcije
T1	Dekodovanje
T2	$A \leftarrow \text{Ulaz}$
Takt	RTN: SUB A, 1
T0	Zahvat instrukcije
T1	Dekodovanje
T2	$A \leftarrow A - 1$
Takt	RTN: OUT
T0	Zahvat instrukcije
T1	Dekodovanje
T2	$\text{Izlaz} \leftarrow A$
Takt	RTN: END
T0	Zahvat instrukcije
T1	Dekodovanje
T2	$IR \leftarrow 111xxxxx$

Do sada je kompletiran konkretan RTN model mikroračunara i on predstavlja bazu za realizaciju sistema. Ako se podsetimo modela vodopada, nakon definisanja konkretnog modela može se reći da je faza specifikacije kompletno urađena, a faza detaljnog projektovanja delimično. U nastavku slede elementi detaljnog projektovanja koji su neophodni da bi se moglo preći na fazu realizacije. U tom smislu konkretan RTN opis svih instrukcija je neophodan uz proširenje upravljačkim signalima za svaku periodu takta. Na osnovu ovog modela će biti znatno olakšano projektovanje upravljačke jedinice.

Faza zahvata:

Takt	RTN	Upravljački signali
T0	$IR \leftarrow M[PC]$	IRload

Faza dekodovanja:

Takt	RTN	Upravljački signali
T1	$PC \leftarrow PC + 1$	PCload

Izvršavanje ostalih instrukcija:

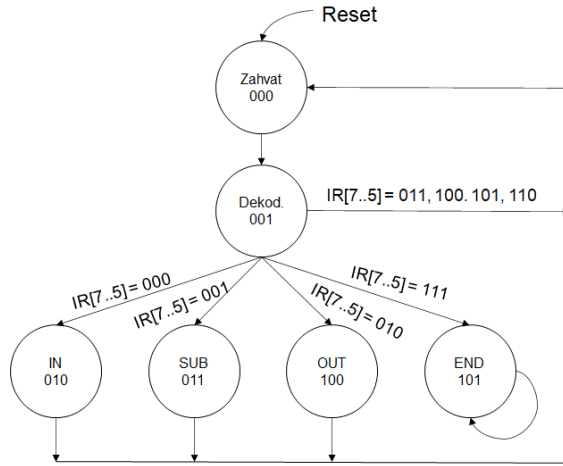
Takt	RTN: IN	Upravljački signali
T0	Zahvat instrukcije	IRload
T1	Dekodovanje	PCload
T2	$A \leftarrow \text{Ulaz}$	INmux, Aload
Takt	RTN: SUB A, 1	Upravljački signali
T0	Zahvat instrukcije	IRload
T1	Dekodovanje	PCload
T2	$A \leftarrow A - 1$	Aload
Takt	RTN: OUT	Upravljački signali
T0	Zahvat instrukcije	IRload
T1	Dekodovanje	PCload
T2	$\text{Izlaz} \leftarrow A$	
Takt	RTN: END	Upravljački signali
T0	Zahvat instrukcije	IRload
T1	Dekodovanje	PCload
T2	$IR \leftarrow 111xxxxx$	

Nakon završetka proširenog konkretnog modela mCPU0 potrebno je još projektovati i upravljačku jedinicu. Ona se može realizovati na bilo koji od načina jer je jako jednostavna<sup>3</sup>. Ovde je prikazana ožičena realizacija zadata preko dijagrama stanja prikazanog na 3.4. Na dijagramu nisu prikazani upravljački signali u svakom stanju da ne opterećuju sliku, već će to biti dato tablicom izlaza.

Automat ima 6 stanja što znači da će biti potrebno 3 bita za kodovanje svakog stanja. Stanjima koja se odnose na izvršenje instrukcije su data ista imena kao i mnemonicima instrukcija. U prvo stanje Zahvat (000), se dolazi nakon reseta.

---

<sup>3</sup>Podsetiti se o načinima realizacije upravljačkih jedinica datim u prethodnom poglavlju.



Slika 3.4: Dijagram stanja upravljačke jedinice mCPU0

Potom direktno sledi stanje Dekodovanje (001) i potom ostala stanja koja su ekvivalentni svakoj od instrukcija: IN (010), SUB (011), OUT (100) i END (101). Prelazi iz stanja dekodovanja u ostala stanja zavise od tri bita najveće značajnosti u IR registru. ( $IR[7..5]$ ). Nakon završetka svake instrukcije prelazi se u stanje Zahvat, kada počinje zahvat nove instrukcije.

Na osnovu datog dijagrama stanja moguće je formirati tablicu prelaza prikazanu u tabeli 3.5 i potom izračunati jednačine pobude. Jednačine pobude za D2, D1 i D0 flip-flobove se mogu izračunati minimizacijom.

Jednačine izlaza su zadate tabelom 3.6. Iz nje se za svaki upravljački signal može izračunati odgovarajuća jednačina. Na osnovu dobijenih jednačina (pobude i izlaza) se može nacrtati šema upravljačke jedinice koja se sastoji od 3 flip-flopa, odgovarajućih logičkih kola zavisno od jednačina i međusobnih veza. Ulazi u UJ su  $IR[7..5]$ , a izlazi su upravljački signali IRload, PCload, INmux i Aload (vidi sliku 3.3). Računanje ovih jednačina i crtanje šeme već delimično spadaju u fazu realizacije, a svakako izlaze iz okvira ovog predmeta pa su ostavljeni čitaocu za samostalan rad.

Nakon faza zahteva, specifikacije i detaljnog projektovanja, bi se moglo pristupiti ostalim fazama (realizacija, verifikacija, upotreba) što izlazi iz okvira ovog predmeta. Ilustracije radi, realizacija UJ preko konačnog automata opisanog u je-

Tabela 3.5: Tablica prelaza

Tekuće stanje	Naredno stanje ( $Q_{2n}Q_{1n}Q_{0n}$ ) za IR[7..5]								
	$Q_2Q_1Q_0$	000	001	010	011	100	101	110	111
000 Zahvat	001	001	001	001	001	001	001	001	001
001 Dekod.	010	011	100	000	000	000	000	000	101
010 IN	000	000	000	000	000	000	000	000	000
011 SUB	000	000	000	000	000	000	000	000	000
100 OUT	000	000	000	000	000	000	000	000	000
101 END	101	101	101	101	101	101	101	101	101

Tabela 3.6: Tablica izlaza

Upravljl. reč	Stanje $Q_2Q_1Q_0$	IRload	PCload	INmux	Aload
0	000 Zahvat	1	0	0	0
1	001 Dekod.	0	1	0	0
2	010 IN	0	0	1	1
3	011 SUB	0	0	0	1
4	100 OUT	0	0	0	0
5	101 END	0	0	0	0

ziku za opis hardvera VHDL prikazana na slici 3.5) <sup>4</sup> <sup>5</sup> može da se uradi u okviru 50-ak linija. Primer test programa za mCPU0 bi bio jednostavan linearni program:

IN

SUB A, 1

OUT

END

<sup>4</sup>VHDL-u je posvećen predmet 'Jezici za modelovanje hardvera' u V semestru.

<sup>5</sup>Primetite da su pojedina stanja automata drugačije imenovana zbog rezervisanih reči u VHDL-u.

```

library ieee; use ieee.std_logic_1164.all;

entity UJ is
port (
  clk, reset : in std_logic;
  ir : in std_logic_vector(2 downto 0);
  up_sig : out std_logic_vector(3 downto 0) -- IRload, PCload, INmux, Aload
);
end entity;

architecture behl of UJ is
  type state_type is (Zahvat, Dekod, INput, SUB, OUTput, ENDkraj);
  signal next_state, state: state_type;
begin
  process (state, ir)
  begin
    case state is
      when Zahvat =>
        next_state <= Dekod;
        up_sig <= "1000";
      when Dekod =>
        if (ir = "010") then next_state <= INput;
        elsif (ir = "011") then next_state <= SUB;
        elsif (ir = "100") then next_state <= OUTput;
        elsif (ir = "101") then next_state <= ENDkraj;
        else next_state <= Zahvat;
        end if;
        up_sig <= "0100";
      when INput =>
        next_state <= Zahvat;
        up_sig <= "0011";
      when SUB =>
        next_state <= Zahvat;
        up_sig <= "0001";
      when OUTput =>
        next_state <= Zahvat;
        up_sig <= (others => '0');
      when ENDkraj =>
        next_state <= ENDkraj;
        up_sig <= (others => '0');
    end case;
  end process;

  process (clk, reset)
  begin
    if (clk'event and clk = '1') then
      if (reset = '1') then
        state <= Zahvat;
      else
        state <= next_state;
      end if;
    end if;
  end process;
end behl;

```

Slika 3.5: VHDL opis upravljačke jedinice mCPU0



### 3.4.2 Zadaci

**Zadatak 1** – U osnovnoj realizaciji mCPU0 izlaz akumulatora je direktno vezan na izlaz tako da instrukcija OUT A nema puno smisla. Modifikovati mCPU0 tako da se na izlaz prosleđuje sadržaj akumulatora samo kada se izvrši instrukcija OUT A instrukcija.

Rešenje:

Potrebno je ubaciti jedan *tri-state* bafer neposredno pre izlaza iz mCPU0 i dodati odgovarajući upravljački signal koji aktivira izlaz bafera.

**Zadatak 2** – U osnovnoj realizaciji mCPU0 skup instrukcija se sastoji od 4 instrukcije. Napraviti odgovarajuće izmene da bi se u skup instrukcija dodala instrukcija ADD A, 1 koja uvećava sadržaj akumulatora za jedan. Neka je opkod ove instrukcije 011.

Rešenje:

Operaciju uvećanja akumulatora za 1 možemo izvesti na bar dva načina. Svakako je neophodna promena ALJ da omogući i sabiranje. Izbor operacije (SUB ili ADD) podrazumeva uvođenje odgovarajućeg upravljačko-selekcionog signal OP-sel. Ovaj signal može da se dobija od upravljačke jedinice. Druga mogućnost je da se iskoristi bit IR4 koji u osnovnoj realizaciji mCPU0 nema nikakvu namenu, pa da on određuje koja se operacija vrši. Dakle da u samom instrukcionom formatu određujemo koja se operacija vrši.

## 3.5 Primeri projektovanja CPU

Nakon što smo u prethodnim sekcijama videli kako se postavljaju zahtevi, specifikacija i detaljno projektovanje mikroprocesora, u ovoj sekciji ćemo videti još tri primera projektovanja CPU. To su mCPU1 koji je realizovan slično kao mikrorračunar EC1 iz [15]. Potom slede još mCPU2 i mEdulent gde svaki predstavlja nadgradnju prethodnog u određenoj meri.

### 3.5.1 mCPU1

Mikroprocesor mCPU0 je imao je za cilj da pokaže najosnovnije koncepte u vezi rada CPU. Skup instrukcija je potpuno pojednostavljen i omogućava jako malo funkcionalnosti, praktično se svodi na pisanje linearnih programa bez grananja. Mikroprocesor mCPU1 predstavlja prvu nadgradnju gde se dodaju i instrukcije grananja kojima se mogu realizovati i petlje. Uvedene su dve instrukcije za to JZ i JMP. JZ je instrukcija koja omogućava skok ako je rezultat prethodne operacije

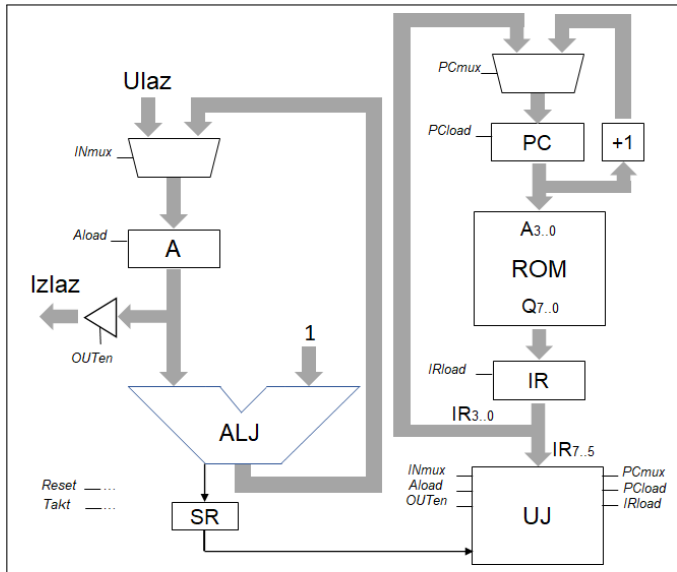
Tabela 3.7: Skup instrukcija mikroprocesora mCPU1

Instrukcija	Kodovana vrednost	Objašnjenje
IN	000xxxxx	Unos podatka u akumulator.
SUB A, 1	001xxxxx	Oduzimanje sadržaja akumulatora za 1.
OUT	010xxxxx	Izlazna vrednost (iz akumulatora).
JZ adresa	011xaaaa	Ako je Z=1 onda skoči na adresu aaaa
JMP adresa	100xaaaa	Skoči na adresu aaaa
END	111xxxxx	Kraj programa.

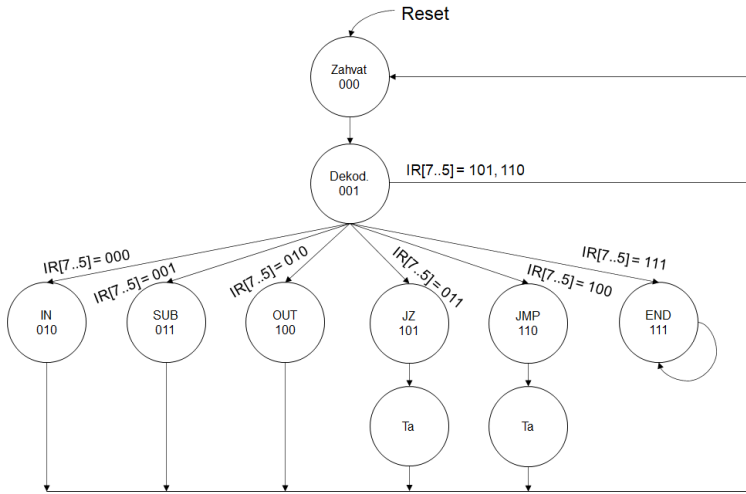
nula, a JMP je instrukcija bezuslovnog skoka. Ovim dodatkom se menjaju sve faze metodologije projektovanja, a ovde ćemo istaći samo promene u svakoj.

Zahtevi mCPU1:

Da bi se mogao detektovati nulti rezultat operacije ALJ, potrebno je imati indikator nule (Z) i uvesti statusni registar (SR) koji će u ovom slučaju biti jednobitni. Novi skup instrukcija nastao dodatkom dve instrukcije grananja je prikazan u tabeli 3.7. Ovako zadate instrukcije grananja su primer kada se adresa skoka nalazi



Slika 3.6: Arhitektura mikroprocesora mCPU1



Slika 3.7: Dijagram stanja upravljačke jedinice mCPU1

unutar tela instrukcije. Videćemo kasnije i druge primere. Na osnovu prethodno opisanih zahteva moguće je uraditi i izmene arhitekture mCPU1 što je prikazano na slici 3.6.

Izmenjen dijagram stanja upravljačke jedinice je prikazan na slici 3.7. Ovde treba primetiti da kod stanja grananja imamo jedno dodatno stanje Ta. Ono je neophodno da se može učitati adresa na koju se vrši skok. Samim tim je trajanje ove dve instrukcije duže za jedan takt. Broj stanja je za ovu upravljačku jedinicu jednak osam.

Specifikacija mCPU1:

U okviru ALJ imamo detekciju nultog rezultata operacije (pomoću npr. 8-ulaznog NILI kola). Ukoliko je rezultat ALJ operacije nula, tada se u statusni registar (SR), koji je potrebno dodati u DP blok, upisuje jedinica na mestu Z bita koji predstavlja indikator nule. Kod mCPU1 ovo je jedini indikatorski bit.

Dodatno u odnosu na mCPU0 je RTN opis dodatih instrukcija JZ i JMP:

adresa  $\langle 3..0 \rangle := \text{IR}\langle 3..0 \rangle := \text{aaaa}$

JZ adresa  $(:= \text{op} = 3) \rightarrow (Z = 1 \rightarrow PC \leftarrow \text{aaaa})$

JMP adresa  $(:= \text{op} = 4) \rightarrow (PC \leftarrow \text{aaaa})$

Upravljački signali su prošireni sa dva dodatna signala OUTen i PCmux. OUTen i odgovarajući dodati *tri-state* bafer služe da OUT instrukcija ima smisla (videti Zadatak 1.). Signal PCmux i odgovarajući multiplekser 2-na-1 je dodat da omogućiti upis adrese skoka kod instrukcija skokova.

Konkretnan RTN model proširen upravljačkim signalima kod mCPU1 je proširen sledećim tablicama za instrukcije skokova u odnosu na mCPU0:

Takt	RTN: OUT	Upravljački signali
T0	Zahvat instrukcije	IRload
T1	Dekodovanje	PCload
T2	Izlaz $\leftarrow A$	OUTen
Takt	RTN: JZ	Upravljački signali
T0	Zahvat instrukcije	IRload
T1	Dekodovanje	PCload
T2	$(Z = 1) \rightarrow PC \leftarrow aaaa$	PCload, PCmux
T3	dodatni takt za realizaciju upisa adrese	
Takt	RTN: JMP	Upravljački signali
T0	Zahvat instrukcije	IRload
T1	Dekodovanje	PCload
T2	$PC \leftarrow aaaa$	PCload, PCmux
T3	dodatni takt za realizaciju upisa adrese	

Ostale faze (realizacija, verifikacija, upotreba) izlaze iz okvira ovog predmeta. Ovde navodimo samo primer test programa koji sadrži petlju:

Adresa	Sadržaj	Kôd
0000 (0)	000xxxxx	IN
0001 (1)	010xxxxx	OUT
0010 (2)	001xxxxx	SUB A, 1
0011 (3)	011x0101	JZ 5
0100 (4)	100x0001	JMP 1
0101 (5)	111xxxxx	END

### 3.5.2 Zadaci

**Zadatak 3** – Uraditi odgovarajuće izmene da bi dodali u skup instrukcija i MOV A, adresa instrukciju koja učitava u akumulator vrednost iz memorije sa adrese aaaa (4 lsb bita u IR). Neka je opkod ove instrukcije 101.

Rešenje:

Da bismo pročitano vrednost sa lokacije aaaa mogli upisati u akumulator potrebno je povećati multiplexer 2-na-1 na 4-na-1 jer sada imamo 3 mogućnosti pri upisu u akumulator. To znači da INmux upravljački signal treba da bude dvobitan. Potrebno je povezati i izlaz ROM-a sa jednim od ulaza multipleksera. Pored toga potrebno je adresu sa koje se čita podatak, koja se nalazi u 4 lsb bita IR registra, proslediti na adresni ulaz ROM-a. Kako sada imamo 2 mogućnosti za ovaj ulaz (sa PC registra ili iz IR3..0 registra) potrebno je dodati i jedan multiplexer 2-na-1 i odgovarajući upravljački signal.

**Zadatak 4** – Šta je u formatu instrukcija potrebno promeniti ako želimo da skup instrukcija umesto sadašnjih 6 sadrži npr. 9 instrukcija?

**Zadatak 5** – Ukoliko bi želeli da dodamo instrukciju JNZ adresa u skup instrukcija, šta bi sve trebalo promeniti?

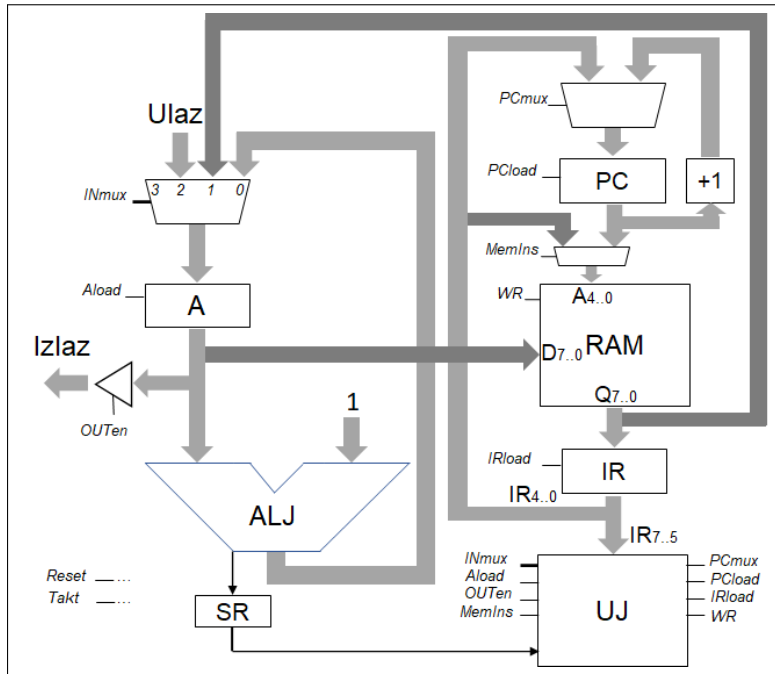
### 3.5.3 mCPU2

Kod mCPU0 i mCPU1 je jedna od velikih mana to što je memorija ROM tipa, tj. iz nje je moguće samo čitati podatak, a upis nije moguć. Kao dalja nadgradnja prethodnih procesora dodajemo još dve instrukcije koje omogućavaju upis i čitanje memorije. Tako da sada imamo memoriju RAM tipa. Skup instrukcija mikroprocesora mCPU2 prikazan je u tabeli 3.8.

Tabela 3.8: Skup instrukcija mikroprocesora mCPU2

Instrukcija	Kodovana vrednost	Objašnjenje
IN	000xxxxx	Unos podatka u akumulator.
SUB A, 1	001xxxxx	Oduzimanje sadržaja akumulatora za 1.
OUT	010xxxxx	Izlazna vrednost (iz akumulatora).
JZ adresa	011aaaaa	Ako je Z=1 onda skoči na adresu aaaaa.
JMP adresa	100aaaaa	Skoči na adresu aaaaa.
MOV A, adresa	101aaaaa	Upiši sadržaj M[aaaaa] u A.
MOV adresa, A	101aaaaa	Upiši sadržaj A u M[aaaaa].
END	111xxxxx	Kraj programa.

RAM memorija je veličine 32x8 (32 reči veličine 8 bita) i moguće je čitati ili upisati sadržaj svake lokacije postavkom odgovarajuće 5-bitne adrese na adresni ulaz memorije ( $A < 4..0$ ). Upravljačka jedinica realizuje upravljačka signala: Aload, INmux, PCload i IRload. Arhitektura mCPU2 je prikazana na slici 3.8. Na ovaj način su dati odgovori na svih 13 pitanja vezanih za zahteve koji se postavljaju



Slika 3.8: Arhitektura mikroprocesora mCPU2

pred CPU pa se može preći na njegovu specifikaciju.

Kod mCPU2 u okviru specifikacije imam nekoliko promena i dodataka. Memorija je duplo veća:  $M[0..2^5 - 1] < 7..0 >$ . Adrese su sada 5-bitne: adresa  $<4..0> := IR<4..0> := aaaaa$ . Imamo i dve nove instrukcije čija je apstraktna RTN specifikacija sledeća:

MOV A, adresa ( $:= op = 5$ )  $\rightarrow (A \leftarrow M[aaaaa])$

MOV adresa, A ( $:= op = 6$ )  $\rightarrow (M[aaaaa] \leftarrow A)$

Potrebno je povećati ulazni multiplexer sa 2-na-1 na 4-na-1 jer je sada potrebno obezbediti i da imamo mogućnost fa upisujemo sadržaj akumulatora iz memorije (videti zadatak 3). Upravljački signali su prošireni sa dva dodatna signala MemIns i WR. MemIns označava da je u pitanju memorijska instrukcija upisa ili čitanja memorije pa sa „žabobilazi” programski brojač i direktno adresira memo-

rija. Signal WR služi da aktivira upis u memoriju. Upravljačka jedinice u skladu sa tim ima dva dodatna stanja za ove dve instrukcije.

Konkretna RTN model proširen upravljačkim signalima kod mCPU2 je proširen sledećim tablicama za instrukcije skokova u odnosu na mCPU0 i mCPU1.

Takt	RTN: MOV A, adresa	Upravljački signali
T0	Zahvat instrukcije	IRload
T1	Dekodovanje	PCload
T2	$A \leftarrow M[\text{aaaaa}]$	Aload, INmux=01, MemIns
Takt	RTN: MOV adresa, A	Upravljački signali
T0	Zahvat instrukcije	IRload
T1	Dekodovanje	PCload
T2	$M[\text{aaaaa}] \leftarrow A$	MemIns, WR

Ostale faze (realizacija, verifikacija, upotreba) projektovanja mikroprocesora izlaze iz okvira ovog predmeta.

### 3.5.4 Zadaci

**Zadatak 6** – Nacrtati dijagram stanja za mCPU2.

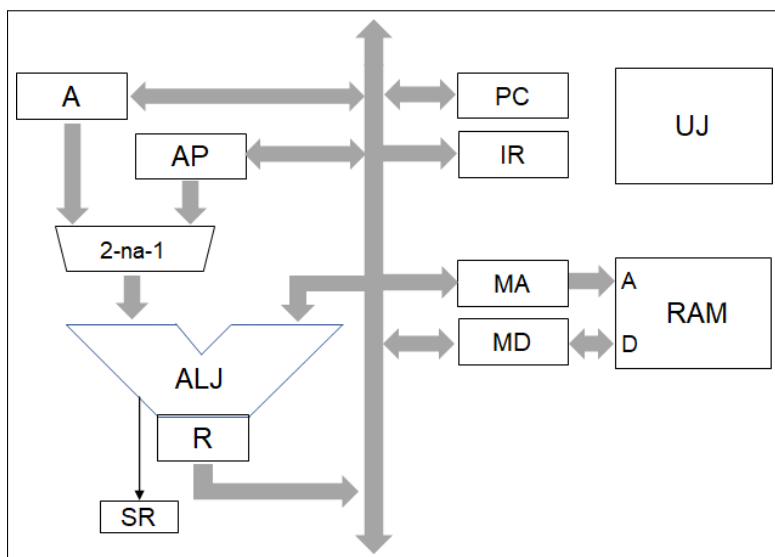
**Zadatak 7** – Projektovati prošireni mikroprocesor mCPU2 tako da ima i instrukciju NOT koja radi invertovanje sadržaja akumulatora.

Rešenje: Ovaj zadatak nije moguće realizovati bez većih modifikacija mCPU2 jer je maksimalan broj mogućih instrukcija koje se mogu realizovati 3-bitnim kodom operacije već realizovan. Jedno od (loših) rešenja je da se izbacii neka instrukciju i na njeno mesto staviti NOT uz dodatak odgovarajućeg 8-ulaznog invertora u ALJ i realizaciju upravljačkog signala za izbor ALJ operacije. Druga ideja bi bila da se adrese vrte na 4-bitne i da onda selekciju ALJ operacije vrši bit  $IR<4>$ . Mana je smanjenje memorije. Alternativno je moguće taj slobodan bit iskoristiti za proširenje koda operacije da bude 4-bitni (vidi zadatak 4).

**Zadatak 8** – Realizovati mCPU2 tako da ima Harvard arhitekturu memorije.

### 3.5.5 mEduleut

Poslednji primer projektovanja mikroprocesora u ovom udžbeniku biće mikroprocesor mEduleut koji predstavlja pojednostavljenu verziju mikroprocesora Eduleut. Na vežbama će studenti biti upoznati sa punom verzijom ovog mikroprocesora i na njegovom simulatoru će imati mogućnost programiranja u simboličkom



Slika 3.9: Strukturalni model mikroprocesora mEdulent

mašinskom jeziku. Detalje o razvoju mikroprocesora Edulent i njegovom mestu u okviru učenja o mikroprocesorima je moguće pogledati u [19].

Radi jasnijeg definisanja zahteva u vezi mikroprocesora mEdulent, ovde polazimo od strukturalnog modela (kompletan prikaz arhitekture bi previše opterećivao sliku) prikazanog na slici 3.9. Na osnovu ovog modela moguće je precizirati zahteve.

Mikroprocesor mEdulent ima **8 registara** i svi su **8-bitni**. Dva su registra direktno programski dostupna korisniku:

- Aakumulator (A) – registar opšte namene koji se koristi u najvećem broju instrukcija. Njegov izlaz je povezan na ulaz aritmetičko-logičke jedinice (ALJ), a isto tako i na 8-bitnu magistralu.
- Adresni pokazivač (AP) – registar koji se koristi za smeštanje adrese pri registarskom indirektnom adresiranju, kao i kod direktnog i neposrednog adresiranja za ADD, SUB i MOV instrukcije. O načinima adresiranja će biti reči nešto kasnije. Ovaj registar je povezan i sa ALJ i sa magistralom.

Registri koji imaju uticaja na tok programa, ali im korisnik ne može direktno pristupiti su:



- Programski brojač (PC) – pokazuje na memorijsku lokaciju sa koje je učitana instrukcija, adresa ili konstanta.
- Statusni registar (SR) – sadrži dva indikatora: indikator prenosa (eng. *Carry*) i indikator nule (eng. *Zero*) koji se postavljaju ukoliko ima prenosa odnosno ukoliko je rezultat aritmetičko-logičke operacije jednak nuli.

Ostali registri specijalne namene korisniku sa programerske tačke gledišta nisu vidljivi:

- Instrukcioni registar (IR) – koristi se za prihvatanje instrukcija iz memorije i koristi se kod svih instrukcija.
- Registar rezultata (R) – služi za privremeno smeštanje rezultata iz ALU jedinice.
- Memorijski adresni registar (MA) – je direktno povezan na adresne linije memorije i služi za adresiranje memorijskih lokacija.
- Memorijski registar podataka (MD) – služi za prihvatanje podataka iz memorije ili za prihvatanje podataka koji treba da se upišu u memoriju.

### **Načini adresiranja instrukcija mEdulenta**

Mikroprocesor Edulent podržava četiri načina adresiranja:

- direktno,
- neposredno,
- registarsko indirektno, i
- predekrement/postinkrement adresiranje.

Mikroprocesor mEdulent podržava dva načina adresiranja:

- direktno, i
- neposredno.

Direktno adresiranje podrazumeva da je operand instrukcije adresa sa koje je potrebno preuzeti podatak iz memorije ili ga postaviti u memoriju. Neposredno (eng. *immediate*) adresiranje podrazumeva da je operand instrukcije neposredni deo instrukcije. Za izvršenje instrukcija koje koriste neposredno i direktno adresiranje potrebne su dve 8-bitne memorijske lokacije (2 bajta), jer se u drugom bajtu

nalazi adresa ili konstanta.

### Skup instrukcija mEdulenta

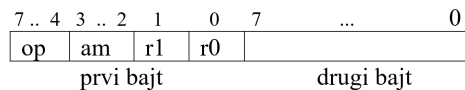
Edulent u svojoj izvornoj verziji ima ukupno 39 instrukcija (uključujući različite načine adresiranja) i mogu da se podele na sledeće grupe:

- instrukcije prenosa podataka,
- aritmetičke,
- logičke,
- instrukcije grananja,
- instrukcije za rad sa potprogramima, i
- instrukcije programske kontrole.

Uzimajući u obzir sve oblike adresiranja, ukupan broj instrukcija mEdulenta je 16. U tabeli 3.9 je dat spisak svih instrukcija mEdulenta.

Tabela 3.9: Spisak svih instrukcija i njihovih mašinskih kodova mEdulenta

Dvobajtnе instrukcije		Jednobajtnе instrukcije	
MOV A, address	0x11	NOP	0x00
MOV AP, address	0x13	END	0x02
MOV A, const	0x19		
MOV AP, const	0x1B		
MOV address, A	0x21		
MOV address, AP	0x23		
ADD A, const	0x39		
ADD AP,const	0x3B		
SUB A, const	0x49		
SUB AP, const	0x4B		
XOR const	0x89		
JMP label	0xA1		
JZ label	0xA5		
JC label	0xA9		



Slika 3.10: Format instrukcije mikroprocesora mEdulent

### Format instrukcija mEdulenta

Kod prethodnih procesora smo videli da je format instrukcije takat da ima 3-bitni kod operacije, pto znači maksimalno osam instrukcija, i 4 ili 5 bita za adresu. Kod mEdulenta su instrukcije su 2-bajtna ili 1-bajtna. Pri tome je tačno određen značaj i funkcija pojedinih bita (ili grupa bita). Radi lakše upotrebe koristimo posebna imena za određene grupe bita:

- *op* – 4-bitni kod operacije
- *am* – tip adresiranja (00 za direktno, 10 za neposredno)
- *r1* – određuje da li se koristi A ili AP registar
- *r0* – određuje dužinu instrukcije (1 ili 2 bajta)

Ukoliko je instrukcija dvobajtna, u drugom bajtu se nalazi konstanta ili adresa u zavisnosti od tipa adresiranja.

Tabela 3.10: Specifikacija delova mEdulent-a

Deo CPU	Objašnjenje
PC<7..0>	programski brojač
IR<7..0>	instrukcioni registar
A<7..0>	akumulator
AP<7..0>	adresni pokazivač
MA<7..0>	memorijski adresni registar
MD<7..0>	memorijski registar podataka
SR<7..0>	statusni registar (koriste se samo 2 lsb bita)
R<7..0>	registar rezultata
M[0..2 <sup>7</sup> - 1] < 7..0 >	memorija
op<3..0> := IR<7..5>	kôd operacije
am<1..0> := IR<3..2>	tip adresiranja ili tip skoka
r1 := IR<1>	označava da li se koristi A ili AP
r0 := IR<0>	označava dužinu instrukcije (jedan ili dva bajta)

Format 2-bajtnje instrukcije prikazan je na slici 3.10.

Specifikacija strukturalnog modela mEdulenta je prikazana u tabeli 3.10 Data je specifikacija registara, memorije i instrukcione reči mEdulenta.

### Apstraktna specifikacija skupa instrukcija

U nastavku će biti dat apstraktni RTN opis skupa instrukcija, počevši sa apstraktnim opisom faze zahvata. Instrukcije mEdulenta se izvršavaju u dve ili tri faze u zavisnosti od dužine. Sve instrukcije imaju fazu zahvata instrukcije. RTN opis faze zahvata dat je sledećom relacijom:

$$zahvat\_instrukcije \rightarrow (IR \leftarrow M[PC] : PC \leftarrow PC + 1)$$

Nakon zahvata instrukcije u instrukcionom registru (IR) se nalazi instrukcija, a istovremeno se programski brojač uveća za jedan. Ovde se vidi rudimentarni primer paralelizacije. Kod dvobajtnih instrukcija druga faza predstavlja prihvatanje operanda nakon koje se u memorijskom registru podataka nalazi operand. RTN opis ove faze dat je sledećom relacijom:

$$zahvat\_operanda \rightarrow (MD \leftarrow M[PC] : PC \leftarrow PC + 1)$$

Nakon ovih faza sledi faza izvršenja instrukcija i u nastavku je prikazana za glavne instrukcije po grupama.

Instrukcije prenosa podataka:

Ove instrukcije služe za prenos podataka iz memorije u A ili AP registar (op = 1) ili za prenos podataka iz A ili AP u memoriju (op = 2).

$$\begin{aligned} \text{MOV } (:= op = 1) \rightarrow & ((am < 1..0 >= 0 \wedge r1 = 0) \rightarrow A \leftarrow M[address]) : \\ & (am < 1..0 >= 0 \wedge r1 = 1) \rightarrow AP \leftarrow M[address] : \\ & (am < 1..0 >= 2 \wedge r1 = 0) \rightarrow A \leftarrow const : \\ & (am < 1..0 >= 2 \wedge r1 = 1) \rightarrow AP \leftarrow const) \end{aligned}$$

$$\begin{aligned} \text{MOV } (:= op = 2) \rightarrow & ((am < 1..0 >= 0 \wedge r1 = 0) \rightarrow M[address] \leftarrow A) : \\ & (am < 1..0 >= 0 \wedge r1 = 1) \rightarrow M[address] \leftarrow AP) \end{aligned}$$

Aritmetiko-logičke operacije:

Instrukcije sabiranja i oduzimanja predstavljaju aritmetičke instrukcije. Logičko ekskluzivno ili predstavlja jedinu logičku instrukciju.

$$\text{ADD} (:= op = 3) \rightarrow \begin{aligned} & ((am < 1..0 \geq 2 \wedge r1 = 0) \rightarrow A \leftarrow A + const) : \\ & (am < 1..0 \geq 2 \wedge r1 = 1) \rightarrow AP \leftarrow AP + const) \end{aligned}$$

$$\text{SUB} (:= op = 4) \rightarrow \begin{aligned} & ((am < 1..0 \geq 2 \wedge r1 = 0) \rightarrow A \leftarrow A - const) : \\ & (am < 1..0 \geq 2 \wedge r1 = 1) \rightarrow AP \leftarrow AP - const) \end{aligned}$$

$$\text{XOR} (:= op = 8) \rightarrow ((am < 1..0 \geq 2 \wedge r1 = 0) \rightarrow A \leftarrow A \oplus const)$$

Instrukcije grananja:

Radi realizacije grananja u programima, definisana je jedna instrukcija bezuslovnog programskog skoka i dve instrukcije uslovnog programskog skoka.

$$\text{JMP} (:= op = A) \rightarrow ((am < 1..0 \geq 0) \rightarrow PC \leftarrow address)$$

$$\text{JZ} (:= op = A) \rightarrow (((am < 1..0 \geq 1) \wedge (SR < 0 \geq 1)) \rightarrow PC \leftarrow address)$$

$$\text{JC} (:= op = A) \rightarrow (((am < 1..0 \geq 2) \wedge (SR < 1 \geq 1)) \rightarrow PC \leftarrow address)$$

Analizom potreba za upravljačkim signalima dobija se da je potrebno 15 signala. Ovih 15 signala su raspoređeni prema rasporedu prikazanom u tabeli 3.11. Na slici 3.11 je prikazana mikrokontrolna reč gde se vidi pozicija svakog od 15 upravljačkih signala.

14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRin	Ain	APin	Rin	Rout	MDin	MDout	MAin	RD	WR	Xsel	Xout	PCin	PCout	PC++

Slika 3.11: Mikrokontrolna reč mikroprocesora mEdulent

### Detaljno projektovanje mikroprocesora mEdulent

U ovoj fazi je neophodno detaljno projektovati sve elemente mikroprocesora koji su postavljeni u fazi zahteva i fazi specifikacije. Da bi to moglo da se uradi neophodno je uraditi specifikaciju konkretnog RTN modela instrukcija koji će sadržati kompletan skup mikroinstrukcija neophodan da bi se izvršila svaka instrukcija. U narednim tablicama će biti odmah prikazan konkretan RTN opis instrukcija proširen upravljačkim signalima. Ako nam je potreban samo konkretan

Tabela 3.11: Upravljački signali mEdulent

Signal	Objašnjenje
MAin	Upis u memorijski adresni registar
MDin	Upis u memorijski registar podataka
MDout	Čitanje memorijskog registra podataka
Ain	Upis u akumulator
APin	Upis u AP registar
XSEL	Koristi A ili AP u tekućoj operaciji
Xout	Čitanje A ili AP (zavisi od Xsel)
Rin	Upis u registar rezultata ALU
Rout	Čitanje registra rezultata
PCin	Upis u programski brojač
PCout	Čitanje programskog brojača
PC++	Uvećanje programskog brojača za 1
IRin	Upis u instrukcioni registar
RD	Čitanje memorije
WR	Upis u memoriju

RTN model, možemo ga dobiti uklanjanjem poslednje kolone u svakoj tablici.

Faza zahvata instrukcije

Takt	RTN	Upravljački signali
T0	$MA \leftarrow PC$	PCout, MAin
T1	$MD \leftarrow M[MA] : PC \leftarrow PC + 1$	RD, MDin, PC++
T2	$IR \leftarrow MD$	MDout, IRin

Faza zahvata operanda

Takt	RTN	Upravljački signali
T3	$MA \leftarrow PC$	PCout, MAin
T4	$MD \leftarrow M[MA] : PC \leftarrow PC + 1$	RD, MDin, PC++

Faza izvršavanja instrukcija

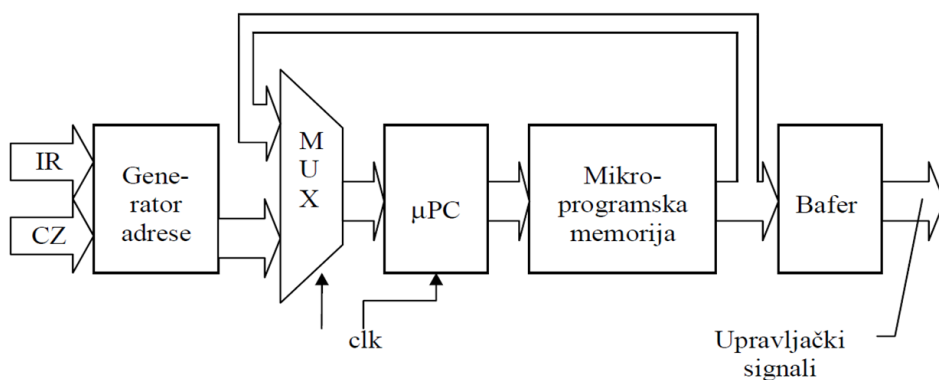
<b>Takt</b>	<b>MOV A, address (0x11, address&lt;7..0&gt;)</b>	<b>Upravljački signali</b>
T0-T2	Zahvat instrukcije	MDout, MAin RD, MDin MDout, Ain
T3-T4	Zahvat operanda	
T5	$MA \leftarrow MD$	
T6	$MD \leftarrow M[MA]$	
T7	$A \leftarrow MD$	
<b>Takt</b>	<b>MOV AP, address (0x13, address&lt;7..0&gt;)</b>	<b>Upravljački signali</b>
T0-T2	Zahvat instrukcije	MDout, MAin RD, MDin MDout, APin
T3-T4	Zahvat operanda	
T5	$MA \leftarrow MD$	
T6	$MD \leftarrow M[MA]$	
T7	$AP \leftarrow MD$	
<b>Takt</b>	<b>MOV A, const (0x19, const&lt;7..0&gt;)</b>	<b>Upravljački signali</b>
T0-T2	Zahvat instrukcije	MDout, Ain
T3-T4	Zahvat operanda	
T5	$A \leftarrow MD$	
<b>Takt</b>	<b>MOV AP, const (0x1B, const&lt;7..0&gt;)</b>	<b>Upravljački signali</b>
T0-T2	Zahvat instrukcije	MDout, APin
T3-T4	Zahvat operanda	
T5	$AP \leftarrow MD$	
<b>Takt</b>	<b>MOV address, A (0x21, address&lt;7..0&gt;)</b>	<b>Upravljački signali</b>
T0-T2	Zahvat instrukcije	MDout, MAin Xout, MDin MDout, WR
T3-T4	Zahvat operanda	
T5	$MA \leftarrow MD$	
T6	$MD \leftarrow A$	
T7	$M[MA] \leftarrow MD$	
<b>Takt</b>	<b>MOV address, AP (0x23, address&lt;7..0&gt;)</b>	<b>Upravljački signali</b>
T0-T2	Zahvat instrukcije	MDout, MAin Xsel, Xout, MDin MDout, WR
T3-T4	Zahvat operanda	
T5	$MA \leftarrow MD$	
T6	$MD \leftarrow AP$	
T7	$M[MA] \leftarrow MD$	

<b>Takt</b>	ADD A, const (0x39, const<7..0>) SUB A, const (0x49, const<7..0>)	<b>Upravljački signali</b>
T0-T2 T3-T4 T5 T6	Zahvat instrukcije Zahvat operanda $R \leftarrow A + MD$ $R \leftarrow A - MD$ $A \leftarrow R$	MDout, ALUFUN<2..0>=3, Rin MDout, ALUFUN<2..0>=4, Rin Rout, Ain
<b>Takt</b>	ADD AP, const (0xB9, const<7..0>) SUB AP, const (0x4B, const<7..0>)	<b>Upravljački signali</b>
T0-T2 T3-T4 T5 T6	Zahvat instrukcije Zahvat operanda $R \leftarrow AP + MD$ $R \leftarrow AP - MD$ $AP \leftarrow R$	MDout, ALUFUN<2..0>=3, Rin MDout, ALUFUN<2..0>=4, Rin Rout, APin
<b>Takt</b>	XOR const (0x89, const<7..0>)	<b>Upravljački signali</b>
T0-T2 T3-T4 T5 T6	Zahvat instrukcije Zahvat operanda $R \leftarrow A \oplus MD$ $A \leftarrow R$	MDout, ALUFUN<2..0>=0, Rin Rout, Ain
<b>Takt</b>	JMP label (0xF1, address<7..0>)	<b>Upravljački signali</b>
T0-T2 T3-T4 T5	Zahvat instrukcije Zahvat adrese $PC \leftarrow MD$	MDout, PCin
<b>Takt</b>	JZ label (0xF5, address<7..0>)	<b>Upravljački signali</b>
T0-T2 T3-T4 T5	Zahvat instrukcije Zahvat adrese $SR < 1..0 > = 1 \rightarrow PC \leftarrow MD$	MDout, PCin
<b>Takt</b>	JC label (0xF9, address<7..0>)	<b>Upravljački signali</b>
T0-T2 T3-T4 T5	Zahvat instrukcije Zahvat adrese $SR < 1..0 > = 2 \rightarrow PC \leftarrow MD$	MDout, PCin



NOP i END instrukcije imaju samo fazu zahvata instrukcije.

Na osnovu prethodne analize i proširenja konkretnog modela upravljačkim signalima, moguće je odrediti sadržaj mikroprogramske upravljačke memorije. Pre toga se podsetimo opšte strukture mikroprogramske upravljačke jedinice, ali primenjene na mEdulent (videti sliku 3.12). Razlika u odnosu na opštu blok šemu datu u prethodnom poglavlju je u tome što se sekvencer realizuje povratnom granom sa izlaza mikroprogramske memorije i koji predstavlja narednu adresu u memoriji. Nakon izvršene poslednje mikroinstrukcije neke instrukcije, se obavezno prelazi na adresu 0 gde počinje izvršavanje mikroinstrukcija za zahvat nove instrukcije. Delimično popunjen sadržaj mikroprogramske memorije je dat u tabeli 3.12.



Slika 3.12: Mikroprogramska upravljačka jedinica mikroprocesora mEdulent

### 3.5.6 Zadaci

Zadatak 9 - Odrediti vrednost sadržaja mikroprogramske memorije na lokacijama 8-25 (označene ?? u tabeli 3.12).

Zadatak 10 - Nacrtati dijagram stanja upravljačke jedinice mEdulenta.

Zadatak 11 - Projektovati neke od edukacionih mikroprocesora (npr. [7], [8], [15], [16], [17] ili [18]).

Zadatak 12 - Razmisliti o mogućnostima realizacije mEdulenta sa Harvard arhitekturom memorije.

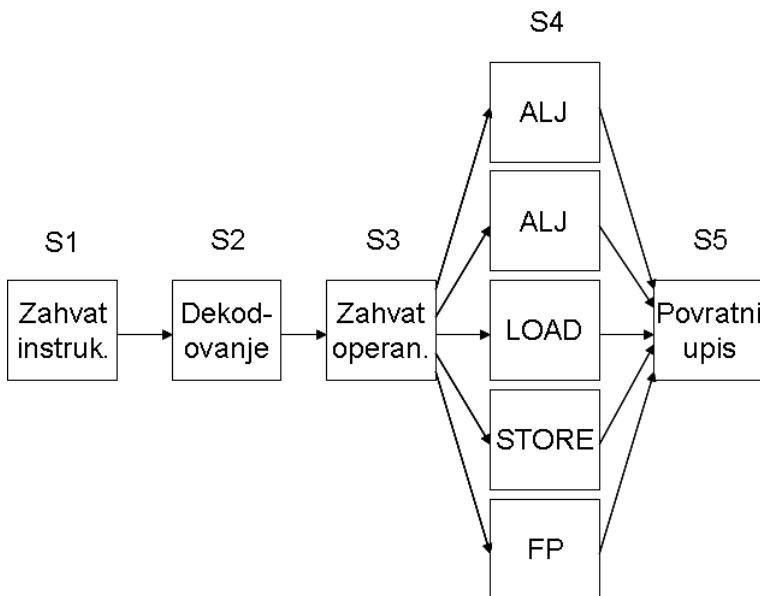
Tabela 3.12: Sadržaj mikroprogramske memorije

n	n+1	Sadržaj	Upravljački signali	RTN	Instrukcija
0	1	0x0082	PCout, MAin	$MA \leftarrow PC$	zahvat instr.
1	2	0x0241	RD, MDin, PC++	$MD \leftarrow M[MA]$ $PC \leftarrow PC + 1$	
2	3	0x4100	MDout, IRin	$IR \leftarrow MD$	
3	4	0x0082	PCout, MAin	$MA \leftarrow PC$	zahvat oper.
4	x	0x0241	RD, MDin, PC++	$MD \leftarrow M[PC]$ $PC \leftarrow PC + 1$	
5	6	0x0180	MDout, MAin	$MA \leftarrow MD$	MOV A, ad.
6	7	0x0240	RD, MDin	$MD \leftarrow M[MA]$	
7	0	0x2100	MDout, Ain	$A \leftarrow MD$	
8	9	??	MDout, MAin	$MA \leftarrow MD$	MOV AP, ad.
9	10	??	RD, MDin	$MD \leftarrow M[MA]$	
10	0	??	MDout, APin	$AP \leftarrow MD$	
11	0	??	MDout, Ain	$A \leftarrow MD$	MOV A, con.
12	0	??	MDout, APin	$AP \leftarrow MD$	MOV AP, con.
13	14	??	MDout, MAin	$MA \leftarrow MD$	MOV ad., A
14	15	??	Xout, MDin	$MD \leftarrow A$	
15	0	??	MDout, WR	$M[MA] \leftarrow MD$	
16	17	??	MDout, MAin	$MA \leftarrow MD$	MOV ad., AP
17	18	??	Xsel, Xout, MDin	$MD \leftarrow AP$	
18	0	??	MDout, WR	$M[MA] \leftarrow MD$	
19	20	??	ALUFUN<2..0>=3, MDout, Rin	$R \leftarrow A + MD$	ADD A, con.
			ALUFUN<2..0>=4, MDout, Rin	$R \leftarrow A - MD$	SUB A, con.
20	0	??	Rout, Ain	$A \leftarrow R$	
21	22	??	ALUFUN<2..0>=3, Xsel, MDout, Rin	$R \leftarrow AP + MD$	ADD AP, con.
			ALUFUN<2..0>=4, Xsel, MDout, Rin	$R \leftarrow AP - MD$	SUB AP, con.
22	0	??	Rout, APin	$AP \leftarrow R$	
23	24	??	ALUFUN<2..0>=0, MDout, Rin	$R \leftarrow A \oplus MD$	XOR con.
24	0	??	Rout, Ain	$A \leftarrow R$	
25	0	??	MDout, PCin	$PC \leftarrow MD$	JMPovi
26	0	0x0000			NOP
27	27	0x0000			END

## 3.6 Principi projektovanja

Za kraj ovog poglavlja ćemo navesti nekoliko ključnih principa projektovanja savremenih arhitektura mikroprocesora. Neki od njih će detaljnije biti objašnjeni u kasnijim poglavljima. Neki od principa su:

1. Sve instrukcije je poželjno da se izvršavaju direktno u hardveru. To drugim rečima znači da nema interpretiranja instrukcija. Na taj način se obezbeđuje veća brzina izvršavanja.
2. Maksimizovati brzinu pokretanja instrukcija tako da se dobije što veći broj instrukcija po sekundi (MIPS)
3. Superskalarne arhitekture koriste više funkcionalnih jedinica u paraleli. Primer sa 5 funkcionalnih jedinica u paraleli je prikazan na slici 3.13.



Slika 3.13: Primer superskalarne arhitekture

4. Poželjno je što više registara.

5. Poželjna je fiksna dužina instrukcija, mali broj polja u instrukcionom formatu, visok nivo regularnosti, što manje različitih formata i sl. Cilj je da instrukcije mogu da se jednostavno dekoduju.
6. Samo Load i Store instrukcije pristupaju memoriji. Što više operacija među registrima to će brzina izvršavanja biti veća jer je manje pristupa memoriji.
7. Upotreba protočne obrade je poželjna.
8. Radi ubrzanja koristiti razne vrste paralelizacije.
9. Težnja da se sve instrukcije izvrše u jednom taktnom periodu ( $CPI = 1$ ).

### 3.7 Pitanja

1. Nacrtati model vodopada i ukratko objasniti pojedinačne faze za slučaj projektovanja CPU.
2. Na koja pitanja je potrebno odgovoriti u fazi zahteva kod projektovanja CPU-a?
3. Nacrtati i objasniti generalnu strukturu mikroprocesora opšte namene.
4. Objasniti razliku između apstraktnog i konkretnog RTN modela CPU.
5. Koja je osnovna namena i velika prednost konkretnog RTN modela proširenog upravljačkim signalima?
6. Objasniti čemu služe tablica prelaza, tablica izlaza, jednačine pobude i jednačine izlaza kod ožičene realizacije upravljačke jedinice.
7. Dat je skup od nekoliko instrukcija u sledećoj tablici (tablica će biti data na kolokvijumu po analogiji sa mCPUx). Nacrtati datapath ovog mikroprocesora po analogiji sa mCPUx.
8. Za dati skup instrukcija (po ugledu na mCPUx) nacrtati dijagram stanja upravljačke jedinice i napisati koji se upravljački signali generišu.
9. Šta je u formatu instrukcija mCPU1 potrebno promeniti ako želimo da skup instrukcija umesto sadašnjih 6 sadrži npr. 9 instrukcija?
10. Ukoliko bi želeli da dodamo instrukciju JNZ adresa (adresa je 4-bitna vrednost aaaa) u skup instrukcija mCPU1, šta bi sve trebalo promeniti?

11. Napisati konkretan RTN model za JZ adresa instrukciju za mCPU1.
12. Međusobno uporediti mCPU0, mCPU1 i mCPU2.
13. Objasniti šta znači: "modifikovana akumulatorska 8-bitna arhitektura von Neumann tipa".
14. Nacrtati strukturalni model mEdulent-a.
15. Napisati izraze za apstraktni i konkretan RTN model faze zahvata instrukcije mEdulenta.
16. Napisati izraze za apstraktni i konkretan RTN model neke instrukcije mEdulenta po želji.
17. Objasniti format instrukcije mEdulenta dat slikom 3.10.
18. Koji su neki od principa projektovanja savremenih arhitektura CPU?
19. Objasniti šta predstavlja superskalarna arhitektura i šta se njome postiže.



# Bibliografija

- [1] Arthur W. Burks, Herman H. Goldstine, and John von Neumann, "Preliminary discussion of the logical design of an electronic computing instrument", The Institute of Advanced Study, Princeton, USA, 1946-47.
- [2] J. Biggs, J. Myers, J. Kufel et al. A natively flexible 32-bit Arm microprocessor, *Nature* 595, pp. 532–536, 2021.
- [3] F. Arute, K. Arya, R. Babbush et al. Quantum supremacy using a programmable superconducting processor. *Nature* 574, pp. 505–510, 2019.
- [4] J. Hochstetter, R. Zhu, A. Loeffler et al. Avalanches and edge-of-chaos learning in neuromorphic nanowire networks. *Nat Commun* 12, 4008, 2021.
- [5] D. A. Patterson, J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, Fourth Edition, Morgan Kaufmann, 2011.
- [6] M. Abd-El-Barr, H. El-Rewini, *Fundamentals Of Computer Organization And Architecture*, John Wiley Sons, Inc., 2005.
- [7] R. S. Sandige, M. L. Sandige, *Fundamentals of Digital and Computer Design with VHDL*, McGraw Hill, 2012.
- [8] M. M. Mano, C. R. Kime, T. Martin, *Logic and Computer Design Fundamentals*, Fifth edition, Pearson, 2015.
- [9] J.L. Gustafson, Moore's Law. In: D. Padua D. (eds) *Encyclopedia of Parallel Computing*. Springer, 2011.
- [10] S.A. McKee, R.W. Wisniewski, Memory Wall. In: D. Padua (eds) *Encyclopedia of Parallel Computing*. Springer, 2011.

- [11] P. Bose, Power Wall. In: D. Padua (eds) Encyclopedia of Parallel Computing. Springer, 2011.
- [12] A. Tsakyridis, T. Alexoudi, A. Miliou, N. Pleros, and C. Vagionas, "10 Gb/s optical random access memory (RAM) cell," Opt. Lett. 44, pp. 1821-1824, 2019.
- [13] W.W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques", IEEE Wescon Proc., Aug. 1970.
- [14] C.G. Bell, A. Newell, Computer structures: Readings and Examples, McGraw-Hill, 1971.
- [15] E.O. Hwang, Digital Logic and Microprocessor Design With VHDL with Interfacing, Cengage Learning, Second Edition, 2017.
- [16] S. Brown, Z. Vranesic, Fundamentals of Digital Logic with VHDL Design, McGraw Hill, 2009.
- [17] D. L. Perry, VHDL: Programming by Example, Fourth edition, McGraw Hill, 2002.
- [18] L. Null, J. Lobur, Essentials of Computer Organization and Architecture, Jones Bartlet Learning, 3rd Ed., 2010.
- [19] I. Mezei "Evolution of an educational microprocessor", Computer Applications in Engineering Education, 28(5), Wiley, pp. 1265-1277, 2020.
- [20] D. Patterson, J.L. Hennessy, Computer Organization and Design RISC-V Edition: The Hardware Software Interface, The Morgan Kaufmann Series in Computer Architecture and Design, 1st Edition, 2017.