Univerzitet u Kragujevcu
TEHNIČKI FAKULTET
Čačak

ARHITEKTURA PROCESORA
FTN_EDU_Rev.1

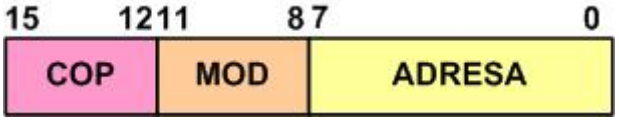
Laboratorija za računarsku tehniku

Čačak
Novembar 2016. godine

OPIS ELEMENATA ARHITEKTURE FTN_EDU_Rev.1

Ovaj tekst je namenjen opisu elemenata arhitekture procesora FTN_EDU_Rev.1. Namena ovoga procesora je demonstracija načina projektovanja računarskog hardvera, njegove simulacije, korišćenjem nekih standardnih računarskih programa, kao što je **LogiSim** i modeliranjem istog hardvera u nekom od HDL jezika, npr. **VHDL**. U ovoj verziji arhitekture biće definisane sve instrukcije koje mogu da se realizuju na nivou arhitekture.

Pri specifikaciji arhitekture poćiće se od pretpostavke da procesor ima jedinstveni format instrukcije, koja je fiksne dužine (**16 bita**) i koja je prikazana na slici 1.



Slika 1. Format instrukcije procesora

Polje **COP** (dužine **4 bita**) definiše operaciju koja se izvršava. Poljem **MOD** (dužine **4 bita**) definiše se način adresiranja. Konačno polje **ADRESA** (dužine **8 bita**), sadrži adresu u memoriji sa koje se preuzima potrebni operand ili predstavlja sam operand u slučaju neposrednog adresiranja. Kao što se iz formata vidi, instrukcije su jednoadresne, tj. procesor od programski dostupnih registara ima samo akumulator, koji je izvorište **prvog operanda** i odredište **rezultata**.

U prvoj verziji arhitekture, shodno formatu instrukcije, karakteristike procesora su sledeće:

1. Dužina registarskog i memorijskog operanda – **16 bita**
2. Dužina neposrednog operanda – **8 bita**
3. Dužina adresne reči – **8 bita**
4. Dužina procesorske reči – **16 bita**
5. Broj instrukcija – **16**
6. Načini adresiranja
    a. **Direktno adresiranje** – adresa u instrukciji ukazuje na lokaciju u memoriji koja sadrži operand
    b. **Neposredno adresiranje** – sadržaj adresnog polja instrukcije predstavlja neposredni operand

7. Procesor ima stek obima **16 lokacija**, koji funkcioniše po principu da **SP** ukazuje na prvu slobodnu lokaciju. Dužina reči koja se pamti u steku je **8 bita**.
8. Skup instrukcija procesora čine:
    a. **Sabiranje**
    b. **Oduzimanje**
    c. **Množenje**
    d. **Deljenje**
    e. **Poređenje**
    f. **Logičko I**
    g. **Logičko ILI**
    h. **Invertovanje**

Aritmetičko – logičke se izvršvaju nad sadržajem akumulatora kao prvim operandom i drugim operandom koji može biti sadržaj određene memorijske lokacije ili neposrednog operanda. Rezultat operacije se pamti u akumulatoru. Shodno tome aritmetičko – logičke operacije mogu da se izvršavaju korišćenjem:

- **Direktnog adresiranja** – prvi operand je u akumulatoru, a drugi operand je definisan adresom u instrukciji. Rezultat se pamti u akumulatoru.
- **Neposrednog adresiranja** – prvi operand je u akumulatoru, a drugi operand je neposredni podatak koji se nalazi u instrukciji. Rezultat se pamti u akumulatoru.

i. **Pomeranje** – ova operacija podrazumeva pomeranje sadržaja akumulatora u određenom smeru za dati broj bitova. Smer pomeranja se definiše bitom MOD0, a broj bitova za koji se sadržaj akumulatora pomera sadržajem adresnog polja instrukcije
    i. **Pomeranje u levo** – MOD1 = 0
    ii. **Pomeranje u desno** – MOD1 =1

j. **Push** – operacija stavljanja podatka na stek. Izvorište podatka koji se stavlja na stek definiše se bitom MOD1
    i. **Pamćenje na steku sadržaja akumulatora (niži bajt)** – MOD1 = 0
    ii. **Pamćenje na steku sadržaja PC (Program Counter)** – MOD1 = 1

k. **Pop** – operacija uzimanja podatka sa steka. Odredište podatka koji se uzima sa steka definiše se bitom MOD1

i. **Čitanje sa steka u akumulator (u niži bajt)** – MOD1 = 0
ii. **Čitanje sa steka u PC** – MOD1 = 1

l. **Jump** – vrsta skoka se definiše bitom MOD0. Ostali bitovi polja MOD definišu kod uslova koji se testira
   i. **Bezuslovni skok** – MOD0 = 0
   ii. **Uslovni skok** – MOD0 = 1

m. **Load** – operacija punjenja akumulatora podatkom. U slučaju neposrednog adresiranja bitom MOD0 se definiše odredište podatka koji se puni u akumulator.
   i. **Direktno adresiranje**
   ii. **Neposredno adresiranje**
      1. **Punjenje neposrednog operanda u niži bajt akumulatora** – MOD0 = 0
      2. **Punjenje neposrednog operanda u viši bajt akumulatora** – MOD0 = 1

n. **Store** – Pamćenje sadržaja akumulatora na određenu lokaciju u memoriji
   i. **Pamćenje sadržaja akumulatora u memoriju**

o. **Halt** – zaustavlja se sinhro jezgro procesora

Formati instrukcija

1. **<u>Sabiranje</u>**

Direktno adresiranje	0000	0000	Adresa
Neposredno adresiranje	0000	0001	Podatak

2. **<u>Oduzimanje</u>**

Direktno adresiranje	0001	0000	Adresa
Neposredno adresiranje	0001	0001	Podatak

3. **<u>Množenje</u>**

Direktno adresiranje	0010	0000	Adresa
Neposredno adresiranje	0010	0001	Podatak

4. **Deljenje**

Direktno adresiranje	0011	0000	Adresa
Neposredno adresiranje	0011	0001	Podatak

5. **Poređenje**

Aritmetičko – direktno	0100	0000	Adresa
Aritmetičko – neposredno	0100	0001	Podatak
Logičko – direktno	0100	0010	Adresa
Logičko – neposredno	0100	0011	Podatak

6. **Pomeranje**

Pomeranje u levo	0101	0000	Podatak
Pomeranje u desno	0101	0010	Podatak

7. **Logičko I**

Direktno adresiranje	0110	0000	Adresa
Neposredno adresiranje	0110	0001	Podatak

8. **Logičko ILI**

Direktno adresiranje	0111	0000	Adresa
Neposredno adresiranje	0111	0001	Podatak

9. **Invertovanje**

Direktno adresiranje	1000	0000	Adresa
Neposredno adresiranje	1000	0001	Podatak
Akumulatorsko	1000	0011	xxxxxxxx

10. PUSH

Pamćenje akumulatora	1001	0000	xxxxxxxx
Pamćenje PC	1001	0010	xxxxxxxx

11. POP

Čitanje u akumulator	1010	0000	xxxxxxxx
Čitanje u PC	1010	0010	xxxxxxxx

12. JUMP

Bezuslovni skok	1100	1110	Adresa
Skok ako je EQU	1100	0010	Adresa
Skok ako je GE	1100	0100	Adresa
Skok ako je LE	1100	0110	Adresa
Skok ako je GT	1100	1000	Adresa
Skok ako je LT	1100	1010	Adresa

13. LOAD

Punjenje iz memorije	1101	0000	Adresa
Punjenje neposredno – niži	1101	0001	Podatak
Punjenje neposredno – viši	1101	0011	Podatak

14. STORE

Pamćenje u memoriji	1110	0000	Adresa

15. HALT

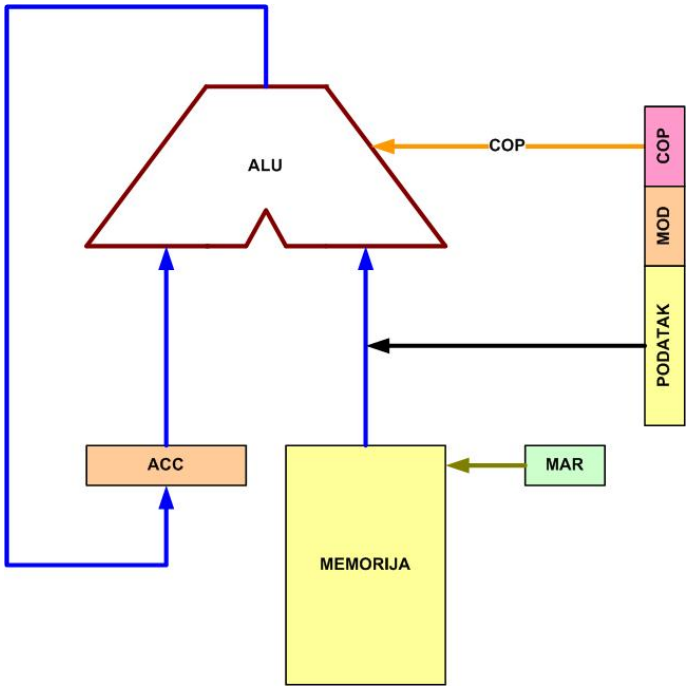
Zaustavljanje	1111	xxxx	xxxxxx

Opis postupka izvršavanja instrukcija

Najveći broj aritmetičko – logičkih instrukcija se izvršavaju na isti način. U te instrukcije spadaju:

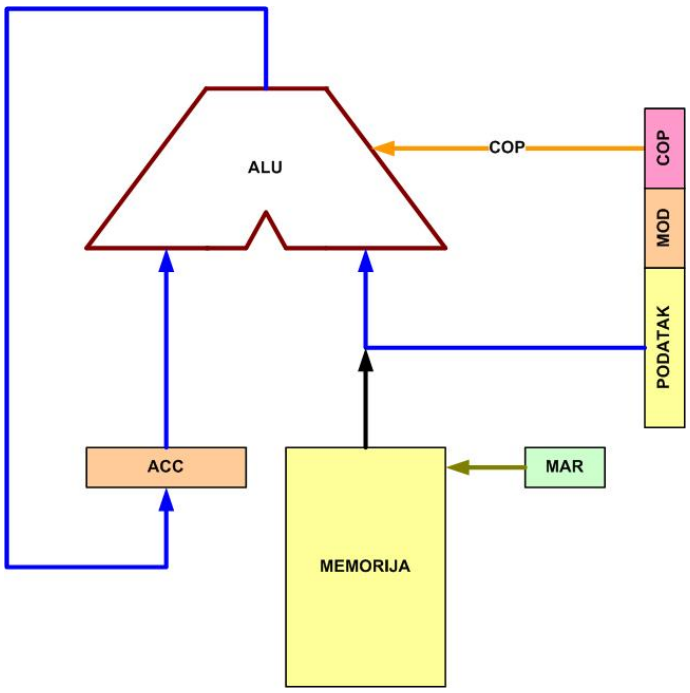
- Sabiranje
- Oduzimanje
- Poređenje
- Množenje
- Deljenje
- Logičko I
- Logičko ILI

Na slici 2 je prikazan način izvršavanja navedenih aritmetičko – logičkih instrukcija u slučaju direktnog adresiranja.



Slika 2. Izvršavanje aritmetičko – logičkih instrukcija u slučaju direktnog adresiranja

Na slici 3 je prikazan način izvršavanja navedenih aritmetičko – logičkih instrukcija u slučaju neposrednog adresiranja.



Slika 3. Izvršavanje aritmetičko – logičkih instrukcija u slučaju neposrednog adresiranja

Plavom bojom su označene aktivne linije podataka, a crnom bojom linije koje trenutno nisu aktivne. Takva situacija postoji na ulazu B aritmetičko – logičke jedinice. U slučaju **direktnog** adresiranja na ulaz B dolaze podaci iz memorije, a u slučaju **neposrednog** adresiranja na ulaz B dolaze podaci iz polja **PODATAK** instrukcijskog registra (IR). Shodno tome na ulazu B mora postojati dvoulazni multiplekser kojim će se multipleksirati podaci iz dva različita izvora. Osnovni pristup multipleksiranju podataka na ulazu B ALU jedinice prikazan je na slici 4.

Multiplekser ima dva ulaza IN0[15:0] i IN1[15:0], izlaz OUT[15:0] i kontrolni ulaz CNT_MUX_ALU_B. Funkcionisanje multipleksera je opisano tabelom 1.

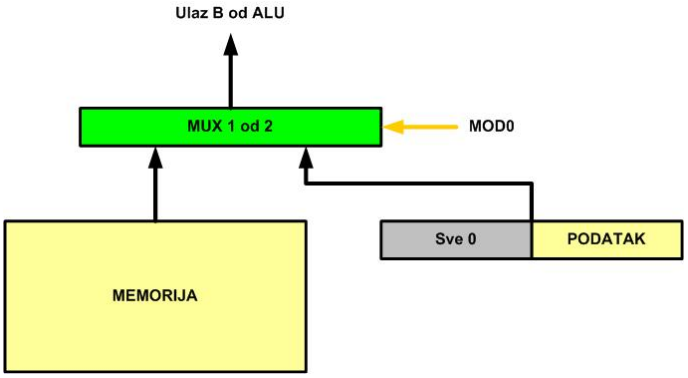
Tabela 1. Upravljanje multiplekserom na ulazu B aritmetičko – logičke jedinice

IN0	IN1	CNT_MUX_ALU_B	OUT
MEMORIJA	**X**	**MOD[0]=0**	**MEMORIJA**
X	**PODATAK**	**MOD[1]=1**	**PODATAK**

Bitom **MOD[0]** se definiše način adresiranja

- kada ima vrednost **0** primenjuje se **direktno** adresiranje;
- kada ima vrednost **1** primenjuje se **neposredno** adresiranje.

Ovo se može iskoristiti za upravljanje multipleksoerom, direktnim dovođenjem bita **MOD[0]** na kontrolni ulaz multiplerksera.

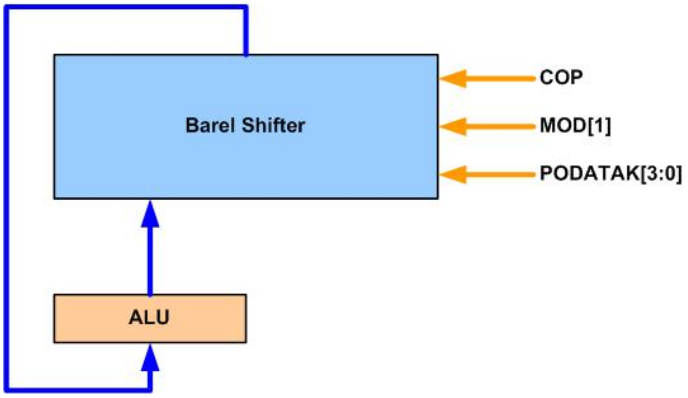


Slika 4. Multipleksiranje podataka na ulazu B ALU jedinice

Pošto je dužina polja **PODACI** u instrukciji 8 bita, a procesorska reč ima dužinu 16 bita da bi podatak i u slučaju neposrednog adresiranja imao definisane vrednosti za svih 16 bita na gornjih 8 ulaza **IN1** ulaznog niza se dovodi vrednost logičke nule („0“).

Pomeranje

Instrukcija pomeranja se realizuje kroz dve operacije – pomeranje u levo i pomeranje u desno sadržaja akumulatora. Smer pomeranja je definisan poljem **MOD[1]**. Ako ovo polje ima vrednost **1** onda se sadržaj akumulatora pomera u levo, a ako ima vrednost **0** onda se sadržaj akumulatora pomera u desno. Broj bitova za koji se vrši pomeranje je definisan sadržajem polja **PODATAK[3:0]** iz instrukcije. Pomeranje je logičko, tj. bitovi koji ostaju „prazni“ posle pomeranja popunjavaju se vrednostima **0**. Izvršavanje operacija pomeranja ilustrovano je skicom na slici 5.

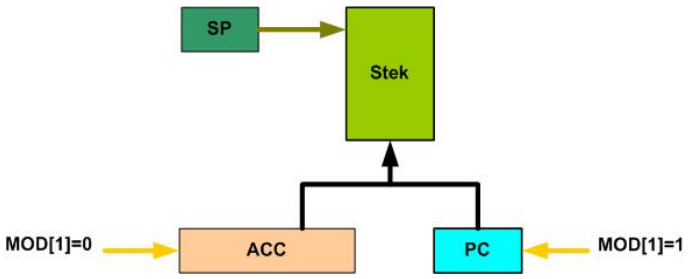


Slika 5. Izvršavanje operacija pomeranja

Push naredba

Ova naredba inicira operaciju stavljanja podatka na stek. Dužina lokacije steka je 8 bita. Podaci se na stek mogu staviti iz dva izvora: **akumulatora (ACC)** i **programskog brojača**

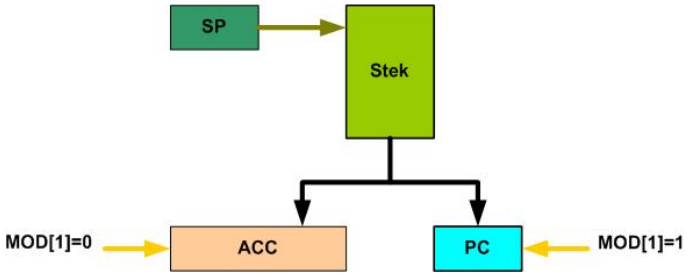
(PC). Kada je izvorište podatka koji se stavlja na stek akumulator onda se pamti niži bajt akumulatora. Izvorište podatka koji se stavlja na stek određuje se sadržajem polja **MOD[1]**. U radu steka je usvojen princip da pokazivač steka **(SP**, **Stack Pointer)** ukazuje na prvu praznu lokaciju u steku. Izvršavanje operacije **PUSH** ilustrovano je na slici 6.



Slika 6. Izvršavanje operacije PUSH

Pop naredba

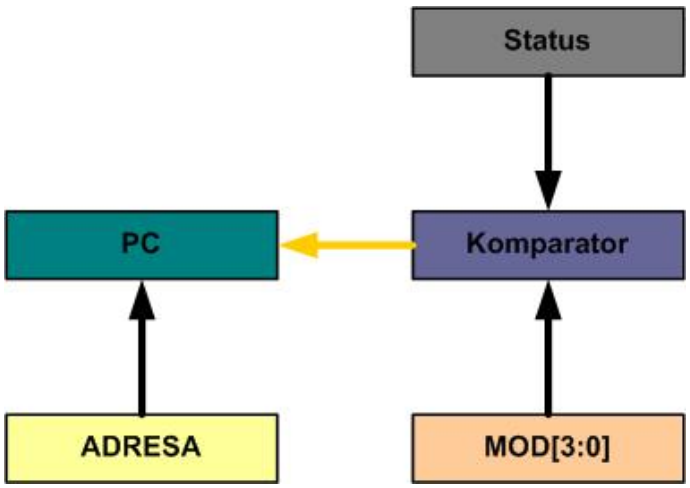
Ova naredba inicira operaciju uzimanja podatka sa steka. Podaci uzeti sa steka mogu upisati u dva odredišta: **akumulator (ACC)** i **programski brojač (PC)**. Kada je odredište pri čitanju podatka sa steka akumulator, onda se pročitani podataka upisuje u niži bajt akumulatora. Odredište podatka koji se stavlja na stek određuje se sadržajem polja **MOD[1]**. Izvršavanje operacije **POP** ilustrovano je na slici 7.



Slika 7. Izvršavanje operacije POP

Operacije skoka

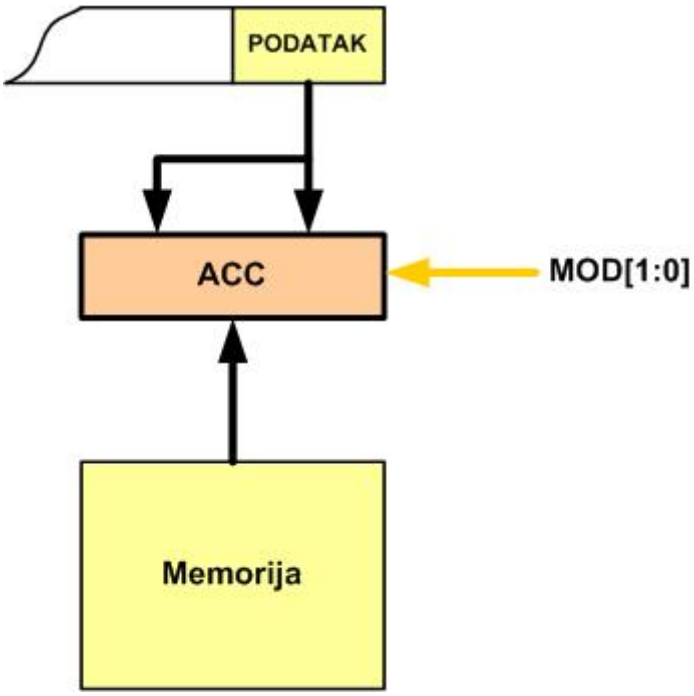
Operacijama skoka se kontroliše tok izvršavanja programa. Adresa na koju se prelazi u programu je definisana poljem **ADRESA** u formatu instrukcije (Bitovi 7:0 instrukcije). Bitovi polja **MOD[3:1]** određuju uslov koji se proverava. Način izvršavanja operacija skoka je ilustrovan na slici 8.



Slika 8. Izvršavanje instrukcija skoka

Punjenje akumulatora

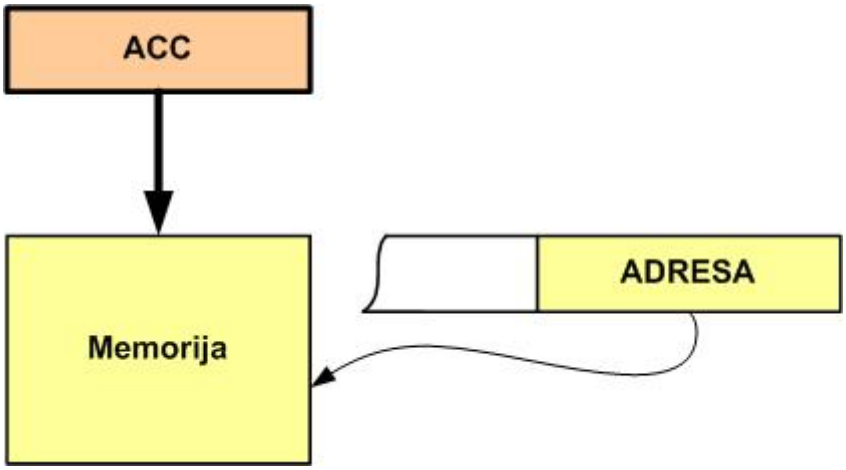
Podatak u akumulator se može napuniti iz dva izvora: **memorije** i **neposredno iz instrukcije**. Pošto je adresno polje u instrukciji, koje se u slučaju neposrednog adresiranja koristi kao izvor podatka, dužine 8 bita, omogućeno je da se on može puniti ili u niži ili u viši bajt akumulatora. Ovo se definiše poljem **MOD[1]**. Izvršavanje operacije punjenja akumulatora ilustrovano je na slici 9.



Slika 9. Izvršavanje operacije punjenja akumulatora

Pamćenje sadržaja akumulatora

Sadržaj akumulatora može da se zapamti u memorijskoj lokaciji čija je adresa specificirana adresnim poljem instrukcije. Izvršavanje operacije pamćenja akumulatora ilustrovano je slikom 10.



Slika 10. Izvršavanje operacije pamćenja akumulatora

SIMBOLIČKI MAŠINSKI JEZIK

Na bazi specificirane arhitekture procesora FTN_EDU_Rev.1 daće se specifikacija odgovarajućg simboličkog mašinskog jezika.

ADD, ADI – Sabiranje

ADD MEM – sabiranje sadržaja akumulatora sa sadržajem memorijske lokacije

- ADD – simboličko ime operacije sabiranja sa direktnim adresiranjem
- MEM – simboličko ime adrese memorijske lokacije gde se nalazi drugi operand[1]

ADI CONST – sabiranje sadržaja akumulatora sa neposrednim operandom

- ADI – simboličko ime operacije sabiranja u slučaju neposrednog adresiranja
- CONST – simboličko ime konstante (neposrednog operanda)[2]

SUB, SUI – Oduzimanje

SUB MEM – oduzimanje sadržaja memorijske lokacije od sadržaja akumulatora

- SUB – simboličko ime operacije oduzimanja u slučaju direktnog adresiranja
- MEM – simboličko ime adrese memorisjke lokacije gde se nalazi drugi operand

SUI CONST – oduzimanje neposrednog operanda od sadržaja akumulatora

- SUI – simboličko ime operacije oduzimanja u slučaju neposrednog adresiranja
- CONST – simboličko ime konstante (neposrednog operanda)

[1] U opštem slučaju adresa kod direktnog adresiranja ne mora se obavezno dati preko simboličkog imena. Adresu je moguće zadati i u obliku brojne vrednosti u decimalnom, binarnom ili heksadecimalnom kodu.

[2] U opštem slučaju konstanta kod neposrednog adresiranja ne mora se obavezno dati preko simboličkog imena. Konstantu je moguće zadati i u obliku brojene vrednosti u decimalnom, binarnom ili heksadecimalnom kosdu.

MUL, MUI – Množenje

MUL MEM – množenje sadržaja akumulatora sa sadržajem memorijske lokacije

- MUL – simboličko ime operacije množenja u slučaju direktnog adresiranja
- MEM – simboličko ime adrese memorisjke lokacije gde se nalazi drugi operand

MUI CONST – množenje sadržaja akumulatora sa neposrednim operandom

- MUI – simboličko ime operacije množenja u slučaju direktnog adresiranja
- CONST – simboličko ime konstante (neposrednog operanda)

DIV, DII – Deljenje

DIV MEM – deljenje sadržaja akumulatora sa sadržajem memorijske lokacije

- DIV – simboličko ime operacije deljenja u slučaju neposrednog adresiranja
- MEM – simboličko ime adrese memorijske lokacije gde se nalazi drugi operand

DII CONST – deljenje sadržaja akumulatora sa neposrednim operandom

- DII – simboličko ime operacije deljenja u slučaju neposrednog adresiranja
- CONST – simboličko ime konstante (neposrednog operanda)

CAD, CAI, CLD, CLI – Poređenje

CAD MEM – aritmemtičko poređenje sadržaja akumulatora sa sadržajem memorije

- CAD – simboličko ime operacije aritmetičkog poređenja u slučaju direktnog adresiranja
- MEM – simboličko ime adrese memorijske lokacije gde se nalazi drugi operand

CAI CONST – aritmemtičko poređenja sadržaja akumulatora sa konstantom

- CAI – simboličko ime operacije aritmemtičkog poređenja u slučaju neposrednog adresiranja
- CONST – simboličko ime konstante (neposrednog operanda)

CLD MEM – logičko poređenje sadržaja akumulatora sa sadržajem memorije

- CLD – simboličko ime operacije logičkog poređenja u slučaju direktnog adresiranja

- MEM – simboličko ime adrese memorijske lokacije gde se nalazi drugi operand

CLI CONST – logičko poređenje sadržaja akumulatora sa konstantom

- CLI – simboličko ime operacije logičkog poređenja u slučaju neposrednog adresiranja
- CONST – simboličko ime konstante (neposrednog operanda)

AND, ANI – Logičko I

AND MEM – logička I funkcija sadržaja akumulatora sa sadržajem memorije

- AND – simboličko ime operacije u slučaju direktnog adresiranja
- MEM – simboličko ime adrese memorisjke lokacije gde se nalazi drugi operanf

ANI CONST – logička I funkcija sadržaja akumulatora sa konstantom

- ANI – simboličko ime operacije u slučaju neposrednog adresiranja
- CONST simboličko ime konstante (neposrednog operanda)

OR, ORI – Logičko ILI

OR MEM – logička ILI funkcija sadržaja akumulatora sa sadržajem memorije

- OR – simboličko ime operacije u slučaju direktnog adresiranja
- MEM – simboličko ime adrese memorisjke lokacije gde se nalazi drugi operanf

ORI CONST – logička I funkcija sadržaja akumulatora sa konstantom

- ORI – simboličko ime operacije u slučaju neposrednog adresiranja
- CONST simboličko ime konstante (neposrednog operanda)

NOT – Invertovanje

NOT – invertovanje sadržaja akumulatora

SHL, SHR – Logičko pomeranje

SHL CONST – logičko pomeranje u levo za broj bitova definisan konstantom

- SHL – simboličko ime operacije logičkog pomeranja u levo
- CONST – simboličko ime konstante koja definiše broj bitova pomeranja

SHR CONST – logičko pomeranje u levo za broj bitova definisan konstantom

- SHR – simboličko ime operacije logičkog pomeranja u desno
- CONST – simboličko ime konstante koja definiše broj bitova pomeranja

PUSHA, PUSHP, POP A, POPP – Operacije nad stekom

PUSH – operacija stavljanja sadržaja akumulatora/programskog brojača na stek

- PUSHA/PUSHP – simboličko ime operacije stavljanja sadržaja akumulatora/programskog brojača na stek

POP – operacija uzimanja sadržaja steka i upis u akumumlator/programski brojač

- POPA/POPP – simboličko ime operacije uzimanja sadržaja steka i upis u akumulator/programski brojač

JUMP – operacije skoka

JUMP ADDR – operacija bezuslovnog skoka na adresu definisanu sa ADDR

- JUMP – simboličko ime operacije bezuslovnog skoka
- ADDR – simboličko ime adrese skoka

JEQ ADDR – operacija skoka ako je su operandi jednaki

- JEQ – simboličko ime operacije skoka kada su operandi jednaki
- ADDR – simboličko ime adrese skoka

JGE ADDR – operacija skoka ako je prvi operand veći ili su operandi jednaki

- JGE – simboličko ime operacije skoka kada je prvi operand veći ili su operandi jednaki
- ADDR – simboličko ime adrese skoka

JLE ADDR – operacija skoka ako je su operandi jednaki

- JLE – simboličko ime operacije skoka kada kada je prvi operand manji ili su operandi jednaki
- ADDR – simboličko ime adrese skoka

JGT ADDR – operacija skoka ako je prvi operand veći

- JLE – simboličko ime operacije skoka kada kada je prvi operand veći
- ADDR – simboličko ime adrese skoka

JLT ADDR – operacija skoka ako je prvi operand manji

- JLE – simboličko ime operacije skoka kada kada je prvi operand manji
- ADDR – simboličko ime adrese skoka

LDA, LA0, LA1 – Punjenje akumulatora

LDA ADDR – operacija punjenja akumulatora sadržajem memorije

- LDA – simboličko ime operacije punjenja akumulatora sadržajem memorije
- ADDR – simboličko ime adrese lokacije u memoriji sa koje se puni sadržaj u akumulator

LA0 CONST – operacija punjenja nižeg bajta akumulatora neposrednim operandom

- LA0 – simboličko ime operacije punjenja nižeg bajta akumulatora neposrednim operandom
- CONST – simboličko ime konstante (neposrednog operanda)

LA1 CONST – operacija punjenja višeg bajta akumulatora neposrednim operandom

- LA1 – simboličko ime operacije punjenja višeg bajta akumulatora neposrednim operandom
- CONST – simboličko ime konstante (neposrednog operanda)

STA – Pamćenje sadržaja akumulatora u memoriji

STA ADDR – operacija pamćenja sadržaja akumulatora u memoriji

- STA – simboličko ime operacije pamćenja sadržaja akumulatora u memoriji
- ADDR – simboličko ime adrese memorijske lokacije u kojoj se pamti sadržaj akumulatora

HALT – Operacija zaustavljanja sinhro jezgra procesora

HALT – operacija zaustavljanja sinhro jezgra procesora

- HALT – simnoličko ime operacije zaustavljanja sinhro jezgra procesora

Simboličko ime	Binarni kod	Način adresiranja	Adresa	Konstanta
ADD	00000000	Direktno	$0 – 2^8 – 1$	
ADI	00000001	Neposredno		$0 – 2^8 – 1$
SUB	00010000	Direktno		
SUI	00010001	Neposredno		
MUL	00100000	Direktno		
MUI	00100001	Neposredno		
DIV	00110000	Direktno		
DII	00110001	Neposredno		
CAD	01000000	Direktno		
CAI	01000001	Neposredno		
CLD	01000010	Direktno		
CLI	01000011	Neposredno		

AND	01010000	Direktno		
ANI	01010001	Neposredno		
OR	01100000	Direktno		
ORI	01100001	Neposredno		
NOT	01110000			
SHL	10000000	Neposredno		1 – 16
SHR	10000001	Neposredno		1 – 16
PUSHA	10010000	Direktno		
PUSHP	10010010	Direktno		
POPA	10100000	Direktno		
POPP	10100010	Direktno		
JUMP	11001110	Direktno		
JEQ	11000001	Direktno		
JGE	11000100	Direktno		
JLE	11000110	Direktno		
JGT	11001000	Direktno		
JLT	11001010	Direktno		
LDA	11010000	Direktno		
LA0	11010001	Neposredno		
LA1	11010011	Neposredno		
STA	11100000	Direktno		
HALT	1111xxxx			