CPU napravljen u Logisim-u

Sadržaj

[CPU napravljen u Logisim-u 1](#_Toc498860928)

[Sadržaj 1](#_Toc498860929)

[1. Arhitektura CPU 7](#_Toc498860930)

[2. Upravljački signali 7](#_Toc498860931)

[Prvih 8 bitova upravljačkih signala iz prvog ROM-a 7](#_Toc498860932)

[Drugih 8 bitova upravljačkih signala iz drugog ROM-a 8](#_Toc498860933)

[Trećih 8 bitova upravljačkih signala iz trećeg ROM-a 8](#_Toc498860934)

[Četvrtih 8 bitova upravljačkih signala iz četvrtog ROM-a 8](#_Toc498860935)

[Petih 8 bitova upravljačkih signala iz petog ROM-a 9](#_Toc498860936)

[Šestih 8 bitova upravljačkih signala iz petog ROM-a 9](#_Toc498860937)

[Sedmih 8 bitova upravljačkih signala iz petog ROM-a 9](#_Toc498860938)

[Način funkcionisanja upravljačkih signala 10](#_Toc498860939)

[Mašinske instrukcije 11](#_Toc498860940)

[3. Instrukcije 11](#_Toc498860941)

[4. Drugi blok instrukcija, čija adresa u firmware-u počinje od 0x100, a imaju neparan OPCODE 13](#_Toc498860942)

[Mikrokod 15](#_Toc498860943)

[Fetch 15](#_Toc498860944)

[nop (0x00) 16](#_Toc498860945)

[5. LOAD operacije 16](#_Toc498860946)

[ld a, b (0x04) 16](#_Toc498860947)

[ld b, a (0x06) 16](#_Toc498860948)

[ld a, xx (0x08, xx) 16](#_Toc498860949)

[ld b, xx (0x0a, xx) 17](#_Toc498860950)

[ld a, [b] (0x0c) 17](#_Toc498860951)

[ld a, [b ± xx] (0x0e, xx) 18](#_Toc498860952)

[ld sp, xx (0x12, xx) 19](#_Toc498860953)

[ld b, sp (0x14) 19](#_Toc498860954)

[ld sp, b (0x16) 19](#_Toc498860955)

[6. ADD i SUB operacije 19](#_Toc498860956)

[add sp, xx (0x1c, xx) 20](#_Toc498860957)

[SP ← SP + xx 20](#_Toc498860958)

[sub sp, xx (0x20, xx) 20](#_Toc498860959)

[add a, b (0x18) 21](#_Toc498860960)

[A ← A + B 21](#_Toc498860961)

[sub a, b (0x1a) 21](#_Toc498860962)

[A ← A – B 21](#_Toc498860963)

[add a, [b] (0x24) 22](#_Toc498860964)

[A ← A + [B] 22](#_Toc498860965)

[sub a, [b] (0x28) 22](#_Toc498860966)

[add a, [b ± xx] (0x2c, xx) 23](#_Toc498860967)

[sub a, [b ± xx] (0x32, xx) 24](#_Toc498860968)

[add a, xx (0x38, xx) 25](#_Toc498860969)

[sub a, xx (0x3c, xx) 25](#_Toc498860970)

[7. CMP operacije 25](#_Toc498860971)

[cmp a, b (0x40) 26](#_Toc498860972)

[cmp a, [b] (0x42) 26](#_Toc498860973)

[cmp a, [b ± xx] (0x71, xx) 27](#_Toc498860974)

[cmp a, xx (0x46, xx) 28](#_Toc498860975)

[cmp b, xx (0x29, xx) 28](#_Toc498860976)

[8. INC i DEC 28](#_Toc498860977)

[dec a (0x4a) 29](#_Toc498860978)

[inc a (0x4c) 29](#_Toc498860979)

[dec b (0x4e) 30](#_Toc498860980)

[inc b (0x50) 30](#_Toc498860981)

[dec [b] (0x52) 31](#_Toc498860982)

[inc [b] (0x56) 32](#_Toc498860983)

[9. AND i OR 32](#_Toc498860984)

[and a, b (0x5a) 33](#_Toc498860985)

[or a, b (0x5c) 33](#_Toc498860986)

[and a, [b] (0x5e) 33](#_Toc498860987)

[or a, [b] (0x62) 34](#_Toc498860988)

[and a, xx (0x66, xx) 35](#_Toc498860989)

[or a, xx (0x6a, xx) 35](#_Toc498860990)

[and a, [b ± xx] (0xaa, xx) 36](#_Toc498860991)

[or a, [b ± xx] (0xb0, xx) 37](#_Toc498860992)

[neg a (0x6e) 37](#_Toc498860993)

[10. Pomeranje u levo i desno 37](#_Toc498860994)

[shl a, b (0xb6) 38](#_Toc498860995)

[A ← A << B 38](#_Toc498860996)

[shr a, b (0xb8) 38](#_Toc498860997)

[A ← A >> B 38](#_Toc498860998)

[shl a, [b] (0xba) 38](#_Toc498860999)

[A ← A << [B] 38](#_Toc498861000)

[shr a, [b] (0xbe) 39](#_Toc498861001)

[shl a, [b ± xx] (0xc2, xx) 40](#_Toc498861002)

[shr a, [b ± xx] (0xc8, xx) 41](#_Toc498861003)

[shl a, xx (0xce, xx) 42](#_Toc498861004)

[shr a, xx (0xd2, xx) 42](#_Toc498861005)

[11. PUSH i POP sa registrima A i B 42](#_Toc498861006)

[push xx (0x70, xx) 43](#_Toc498861007)

[push a (0x74) 43](#_Toc498861008)

[pop a (0x76) 43](#_Toc498861009)

[push b (0x78) 44](#_Toc498861010)

[pop b (0x7a) 44](#_Toc498861011)

[12. Skokovi i pozivi potprograma 44](#_Toc498861012)

[jmp xx (0x7c, xx) 44](#_Toc498861013)

[jz xx (0x7e, xx) 45](#_Toc498861014)

[jnz xx (0x82, xx) 46](#_Toc498861015)

[jp xx (0x9a, xx) 47](#_Toc498861016)

[jnp xx (0x9e, xx) 48](#_Toc498861017)

[jo xx (0xa2, xx) 49](#_Toc498861018)

[jno xx (0xa6, xx) 50](#_Toc498861019)

[call xx (0x86, xx) 51](#_Toc498861020)

[ret (0x8a) 51](#_Toc498861021)

[callz xx (0xd6, xx) 52](#_Toc498861022)

[callnz xx (0xdc, xx) 53](#_Toc498861023)

[callp xx (0xe2, xx) 54](#_Toc498861024)

[callnp xx (0xe8, xx) 55](#_Toc498861025)

[callo xx (0xee, xx) 56](#_Toc498861026)

[callno xx (0xf4, xx) 57](#_Toc498861027)

[13. STORE operacije 57](#_Toc498861028)

[st a, [xx] (0x8c, xx) 58](#_Toc498861029)

[st b, [xx] (0x90, xx) 58](#_Toc498861030)

[st a, [b] (0x94) 59](#_Toc498861031)

[st a, [b ± xx] (0x96, xx) 59](#_Toc498861032)

[14. Množenje i deljenje 59](#_Toc498861033)

[mul a, b (0x01) 60](#_Toc498861034)

[div a, b (0x15) 60](#_Toc498861035)

[mul a, [b ± xx] (0x41) 62](#_Toc498861036)

[div a, [b ± xx] (0x47) 63](#_Toc498861037)

[mul a, xx (0x4d) 64](#_Toc498861038)

[div a, xx (0x51) 65](#_Toc498861039)

[ld a, h (0xfa) 66](#_Toc498861040)

[15. Registar C 66](#_Toc498861041)

[ld c, xx (0x2d, xx) 66](#_Toc498861042)

[ld a, [c] (0x59) 67](#_Toc498861043)

[ld a, [c ± xx] (0x5b, xx) 67](#_Toc498861044)

[st a, [c] (0x2f) 68](#_Toc498861045)

[st a, [c ± xx] (0x31, xx) 69](#_Toc498861046)

[dec c (0x35) 70](#_Toc498861047)

[inc c (0x37) 70](#_Toc498861048)

[ld c, a (0x39) 71](#_Toc498861049)

[ld c, b (0x3b) 71](#_Toc498861050)

[ld a, c (0x55) 71](#_Toc498861051)

[ld b, c (0x57) 72](#_Toc498861052)

[push c (0x3d) 72](#_Toc498861053)

[pop c (0x3f) 72](#_Toc498861054)

[mul a, c (0x5f) 73](#_Toc498861055)

[div a, c (0x61) 73](#_Toc498861056)

[16. XOR operacija 73](#_Toc498861057)

[xor a, b (0x63) 74](#_Toc498861058)

[A ← A xor B 74](#_Toc498861059)

[xor a, c (0x65) 74](#_Toc498861060)

[xor a, [b ± xx] (0x67, xx) 75](#_Toc498861061)

[xor a, [b] (0x6d) 76](#_Toc498861062)

[add a, c (0x77) 76](#_Toc498861063)

[A ← A + C 76](#_Toc498861064)

[sub a, c (0x79) 77](#_Toc498861065)

[A ← A – C 77](#_Toc498861066)

[st a, [b + c ± xx] (0x7b, xx) 78](#_Toc498861067)

[halt (0xff) 79](#_Toc498861068)

[17. 79](#_Toc498861069)

[Programi 80](#_Toc498861070)

[Primer1- za operacije od LD, ADD, SUB, PUSH i POP 80](#_Toc498861071)

[Primer2- za JMP 80](#_Toc498861072)

[Primer3- za JNZ i DEC A 80](#_Toc498861073)

[Primer4- za CALL x i RET 81](#_Toc498861074)

[Primer5- koji računa sumu od 1 do 4 i upisuje u adresu 0x80 (128) 81](#_Toc498861075)

[Primer6- koji proverava OVERFLOW flag 81](#_Toc498861076)

[Primer7- koji upisuje u video memoriju (od 0xf0 do 0xff) ASCII karakter2 (od 0x41 za ‘A’ do 0x51 za ’P’), što izaziva ispis u sva četiri LED matrix elementa (svaki od po 4 karaktera) 82](#_Toc498861077)

[Primer8- razmenjuje sadržaj registra A i B i upisuje u video memoriju (0xf0 i 0xf1) dva ASCII karaktera (0x41 za ‘A’ i 0x42 za ‘B’) iz registara A i B, što izaziva ispis u dva LED matrix elementa 82](#_Toc498861078)

[Primer9- indirektno adresiranje (LD A, [b]) 83](#_Toc498861079)

[Primer10- AND, OR I NOT 83](#_Toc498861080)

[Primer11- JZ, xx komanda 83](#_Toc498861081)

[Primer12- ST A, [b] – indirektno adresiranje - snima registar A u memoriju na adresu na koju ukazuje B 84](#_Toc498861082)

[Primer13- CMP A, B – poredi A i B 84](#_Toc498861083)

[Primer14- CMP A, [b] – poredi A i sadržaj adrese na koju ukazuje B 84](#_Toc498861084)

[Primer15- ADD A, [b] i SUB A, [b] – sabira/oduzima A i sadržaj adrese na koju ukazuje B 85](#_Toc498861085)

[Primer16- DEC [b] i INC [b] – smanjuje/povećava sadržaj adrese na koju ukazuje B 85](#_Toc498861086)

[Primer17- ADD A, xx i SUB A, xx – sabira/oduzima A i broj 85](#_Toc498861087)

[Primer18- CMP A, xx – poredi A i broj 86](#_Toc498861088)

[Primer19- AND-OR A, [B] – poredi A i broj 86](#_Toc498861089)

[Primer20- AND-OR A, xx 86](#_Toc498861090)

[Primer21- poziv potprograma i stack frame 87](#_Toc498861091)

[Primer22- MUL A, B 88](#_Toc498861092)

[Primer23- DIV A, B 88](#_Toc498861093)

[Primer24- Primer dobavljanja HIGH-BYTE iz ALU 89](#_Toc498861094)

[Primer25- HELLO WORLD i registar C 89](#_Toc498861095)

[Primer26- Rekurzivna funkcija faktorijel 90](#_Toc498861096)

[Primer27- MUL A, C 92](#_Toc498861097)

[Primer28- DIV A, C 92](#_Toc498861098)

[Primer29- XOR A, B i XOR A, C 92](#_Toc498861099)

[Primer30- XOR A, [B + 1] 93](#_Toc498861100)

# Arhitektura CPU

* 16-bitna arhitektura
* tri registra opšte namene: A, B i C (A je najjači, odn. ima najviše mogućnosti – klasičan akumulatorski registar, B je malo slabiji i uglavnom se koristi kao stek pointer, a C uglavnom služi za čitanje i pisanje iz memorije i u memoriju),
* MBR, IR, PC i SP,
* ZERO, OVERFLOW i POSITIVE flag,
* ALU, koji ima dva ulaza (A i B – ne mešati sa dva istoimena registra), osam operacija (+, -, AND, OR, negacija, MUL, DIV i XOR) i jedan registar (HIGH-BYTE) u kojem se čuva gornji bajt prilikom množenja (rezultat je 32-bitni, a donja 16-bitna reč je u izlazu od ALU) ili ostatak pri deljenju (rezultat je u A registru).
* video memorija koja počinje od adrese 0xfff0 i ide do 0xffff (16 bajtova). U pitanju je tekstualna video memorija, odn. u navedene lokacije se upisuju ASCII vrednosti karaktera koji se onda tu i prikazuju. Na primer, ako se na lokaciju 0xfff0 upiše broj 65, onda će se na displeju, na prvoj lokaciji (od 16) videti slovo ‘A’.

MAR (Memory Address Register) ne postoji, ali imamo signale (na primer, SP-1-TO-MAR i LOAD-MAR) u kojima se spominje MAR, što zapravo znači da se izlaz iz registara PC, SP ili MBR direktno šalje na adresnu magistralu, kada su ovi signali jedinice.

Firmware sadrži mikrokod, odn. upravljačke signale, podeljene u dva paralelna ROM-a, svaki po 32 bita u ćeliji.

# Upravljački signali

U ovom poglavlju biće navedeni i objašnjeni svi upravljački signali koji se koriste.

## Prvih 8 bitova upravljačkih signala iz prvog ROM-a

|  |  |
| --- | --- |
| **BIT** | **OBJAŠNJENJE** |
| SP-1-TO-MAR | kada je 1, vrednost SP umanjena za 1 (SP-1) dolazi na adresnu magistralu (koristi se kod POP i RET operacija) |
| LOAD-A | kada je 1, puni registar A ulaznim brojem (sinhronizovano taktom iz CLK ulaza) |
| LOAD-B | kada je 1, puni registar B ulaznim brojem (sinhronizovano taktom iz CLK ulaza) |
| MICRO-CODE-COUNTER | kada je 1, adresa u firmware ROM-u ne dolazi od brojača, već od IR; time je omogućeno da se skoči u ROM-u na adresu na koju ukazuje OPCODE |
| OE-B | kada je 1, sadržaj iz B registra odlazi napolje (inače je odspojen) |
| OE-A | kada je 1, sadržaj iz A registra odlazi napolje (inače je odspojen) |
| MEM-WR | kada je 1, piše se u RAM |
| MEM-RD | kada je 1, čita se iz RAM-a |

## Drugih 8 bitova upravljačkih signala iz prvog ROM-a

|  |  |
| --- | --- |
| **BIT** | **OBJAŠNJENJE** |
| ALU-NEG | operacija negacije bitova A ulaza |
| ALU-OR | REZ ← A || B |
| ALU-AND | REZ ← A && B |
| ALU-SUB | REZ ← A - B |
| ALU-ADD | REZ ← A + B |
| ALU-GET-RES | kada je 1, rezultat odlazi iz ALU |
| ALU-A-LOAD-B | kada je 1, u ulaz A dolazi registar B (inače ide registar A) |
| ONE-TO-ALU | kada je 1, ulaz B se puni jedinicom |

## Trećih 8 bitova upravljačkih signala iz prvog ROM-a

|  |  |
| --- | --- |
| **BIT** | **OBJAŠNJENJE** |
| LOAD-MAR | kada je 1, izlaz iz registra ide na adresnu magistralu; stavljeno kod PC, MBR i SP da njihovi sadržaji mogu da odu na adresnu magistralu |
| SP-DEC | smanjuje SP za 1 |
| IR-OE | kada je 1, sadržaj IR odlazi napolje (inače je odspojen) |
| LOAD-IR | kada je 1, puni registar IR ulaznim brojem (sinhronizovano taktom iz CLK ulaza) |
| PC-OE | kada je 1, sadržaj PC odlazi napolje (inače je odspojen) |
| MBR-OE | kada je 1, sadržaj MBR odlazi napolje (inače je odspojen) |
| LOAD-PC | kada je 1, puni registar PC ulaznim brojem (sinhronizovano taktom iz CLK ulaza) |
| LOAD-MBR | kada je 1, puni registar MBR ulaznim brojem (sinhronizovano taktom iz CLK ulaza) |

## Četvrtih 8 bitova upravljačkih signala iz prvog ROM-a

|  |  |
| --- | --- |
| **BIT** | **OBJAŠNJENJE** |
| LOAD-SP | kada je 1, puni registar SP ulaznim brojem iz DATA-BUS-a (sinhronizovano taktom iz CLK ulaza) |
| SP-OE | kada je 1, sadržaj SP odlazi napolje (inače je odspojen) |
| PC-INC | povećava PC za 1 |
| HALT | ako je 1, zaustavlja clock, pa samom time i CPU, odnosno, računar |
| SP-INC | povećava SP za 1 |
| MARK-FLAG | flegovi se pamte samo ako je ovaj pin 1; koristi se kod aritmetičkih operacija da postavi flegove |
| CHECK-ZERO | ovaj pin i ZERO flag idu u AND kolo i ako su oba jedinica, izazivaju ponovni fetch, odn. prelazak na sledeću instrukciju; koristi se kod JNZ instrukcije |
| ZERO-CONST | kada je 1, adresa u firmware ROM-u ide na 0x00 (na početak fetch-a |

## Petih 8 bitova upravljačkih signala iz drugog ROM-a

|  |  |
| --- | --- |
| **BIT** | **OBJAŠNJENJE** |
| ALU-B-LOAD-DATA-BUS | kada je 1, u ALU se na ulaz B stavlja sadržaj sa data magistrale |
| ALU-A-LOAD-SP | kada je 1, u ALU se na ulaz A stavlja sadržaj SP; koristi se kod add sp, xx i sub sp, xx |
| MBR-DATA-BUS | kada je 1, izlaz iz MBR ide na data bus, a ne na adress bus |
| ALU-A-LOAD-MBR | kada je 1, u ALU se na ulaz A stavlja MBR; koristi se kod DEC [b] |
| ALU-B-LOAD-MBR | kada je 1, u ALU se na ulaz B stavlja MBR; koristi se kod CMP A, [b]; ADD A, [b]; SUB A, [b] |
| CHECK-NOT-ZERO | ovaj pin i negirani ZERO flag idu u AND kolo i ako su oba jedinica, izazivaju ponovni fetch, odn. prelazak na sledeću instrukciju; koristi se kod JZ instrukcije |
| B-ADDR-BUS | kada je 1, izlaz iz registra B ide na adresnu magistralu; inače ide na data magistralu |
| PC-TO-DATA | kada je 1, onda ne dozvoljava da sadržaj PC ode na addresnu magistralu, već na data magistralu |

## 

## Šestih 8 bitova upravljačkih signala iz drugog ROM-a

|  |  |
| --- | --- |
| **BIT** | **OBJAŠNJENJE** |
| CHECK-POSITIVE | kada je 1, proverava se da li je POSITIVE flag 1; ako jeste, onda se radi fetch, odn. preskače se ostatak mikrokoda |
| CHECK-NOT-POSITIVE | kada je 1, proverava se da li je POSITIVE flag 0; ako jeste, onda se radi fetch, odn. preskače se ostatak mikrokoda |
| CHECK-OVERFLOW | kada je 1, proverava se da li je OVERFLOW flag 1; ako jeste, onda se radi fetch, odn. preskače se ostatak mikrokoda |
| CHECK-NOT-OVERFLOW | kada je 1, proverava se da li je OVERFLOW flag 0; ako jeste, onda se radi fetch, odn. preskače se ostatak mikrokoda |
| ALU-SHR | kada je 1, ALU radi pomeranje u desno |
| ALU-SHL | kada je 1, ALU radi pomeranje u levo |
| ALU-DIV | kada je 1, ALU radi deljenje |
| ALU-MUL | kada je 1, ALU radi množenje |

## Sedmih 8 bitova upravljačkih signala iz drugog ROM-a

|  |  |
| --- | --- |
| **BIT** | **OBJAŠNJENJE** |
| GET-HIGH | kada je 1, prebacuje sadržaj HIGH-BYTE registra iz ALU (gde se čuva gornjih 8 bita rezultata množenja, ili ostatak pri deljenju) u registar A |
| LOAD-C | kada je 1, učitava sadržaj data magistrale u registar c |
| OE-C | kada je 1, sadržaj registra c odlazi na data magistralu |
| C-ADDR-BUS | kada je 1, sadržaj registra c odlazi na adresnu magistralu |
| DIV-COUNT | kada je 1, mikrokod odlazi na adresu 0x101, gde čeka rezultat deljenja od ALU (16 taktova) |
| MUL-COUNT | kada je 1, mikrokod odlazi na adresu 0x115, gde čeka rezultat množenja od ALU (16 taktova) |
| ALU-B-LOAD-C | kada je 1, u B ulaz ALU dolazi registar C |
| ALU-XOR | kada je 1, ALU radi operaciju XOR između ALU-A i ALU-B ulaza |

## Osmih 8 bitova upravljačkih signala iz drugog ROM-a

|  |  |
| --- | --- |
| **BIT** | **OBJAŠNJENJE** |
| WAIT | kada je 1, takt je blokiran u svemu, sem u ALU; koristi se dok se čeka rezultat množenja ili deljenja iz ALU |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Način funkcionisanja upravljačkih signala

U svakom trenutku u vremenu neki skup od 64 upravljačka signala upravlja procesorom i računarom. Ti upravljački signali:

* definišu koji od registara izbacuje svoj sadržaj (OE-A, OE-B, PC-OE, SP-OE, MBR-OE),
* definišu koji od registara će prihvatiti sadržaj koji se pojavio na njegovom ulazu (LOAD-A, LOAD-B, LOAD-PC, LOAD-SP, LOAD-MBR)
* definišu da li da u registar dođe sadžaj sa data magistrale, ili pak uvećana/smanjena vrednost
* definišu koja će operacija biti izvedena, kako nad podacima (aritmetičke operacije itd.), tako i razne druge operacije nad delovima procesora (zaustavi procesor, itd.).

Mašinske instrukcije

Format mašinskih instrukcija je jednostavan: jedna ili dve reči (svaka reč je dva bajta i učitava se odjednom iz memorije). Ako je jedna reč, onda je ta reč OPCODE (Operation Code), i kao takva koristi se kao adresa u firmware memoriji, da odatle procesor krene da izbacuje upravljačke signale. Ako je instrukcija od dve reči, onda je prva reč OPCODE, a druga je operator (argument, parametar), što je uvek broj.

# Instrukcije

1. [(0x00)](#_nop_(0x00)) – nop; No Operation – ne radi ništa
2. [(0x04)](#_ld_a,_b_1) – ld a, b; a ← b
3. [(0x06)](#_ld_b,_a) – ld b, a; b← a
4. [(0x08, xx)](#_ld_a,_xx) – ld a, xx; prvi bajt je opcode (0x08), drugi bajt je broj (0 – 255, ili -128 – 127)
5. [(0x0a, xx)](#_ld_b,_xx_1) – ld b, xx; prvi bajt je opcode (0x0a), drugi bajt je broj (0 – 255, ili -128 – 127)
6. [(0x0c)](#_ld_a,_[b]) – ld a, [b]; u a stavlja sadržaj memorijske lokacije na koju ukazuje b
7. [(0x0e, xx)](#_ld_a,_[b) – ld a, [b ± xx]; smešta u registar a sadržaj memorijske lokacije na koju ukazuje zbir/razlika (b ± xx)
8. [(0x12, xx)](#_ld_sp,_xx) – ld sp, xx; prvi bajt je opcode (0x12), drugi bajt je broj (0 – 255, ili -128 – 127)
9. [(0x14)](#_ld_b,_sp) – ld b, sp; u b stavlja sadržaj registra sp
10. [(0x16)](#_ld_sp,_b) – ld sp, b; u sp stavlja sadržaj registra b
11. [(0x18)](#_add_a,_b) – add a, b; jedini bajt je opcode (0x18); a ← a + b
12. [(0x1a)](#_sub_a,_b) – sub a, b; jedini bajt je opcode (0x1a); a ← a – b
13. [(0x1c, xx)](#_add_sp,_xx_1) – add sp, xx; sabira sp i broj xx, koji je odmah posle opcode-a (0x1c)
14. [(0x20, xx)](#_sub_sp,_xx) – sub sp, xx; oduzima broj xx od sp, koji je odmah posle opcode-a (0x20)
15. [(0x24)](#_add_a,_[b]) – add a, [b]; sabira sadržaj a i lokacije na koju ukazuje registar b
16. [(0x28)](#_sub_a,_[b]) – sub a, [b]; oduzima sadržaj a i lokacije na koju ukazuje registar b
17. [(0x2c, xx)](#_add_a,_[b) – add a, [b ± xx]; sabira registar a i sadržaj memorijske lokacije na koju ukazuje zbir/razlika (b ± xx)
18. [(0x32, xx)](#_sub_a,_[b) – sub a, [b ± xx]; oduzima od registra a sadržaj memorijske lokacije na koju ukazuje zbir/razlika (b ± xx)
19. [(0x38, xx)](#_add_a,_xx) – add a, xx; sabira a i broj xx; prvi bajt je opcode (0x38), a drugi je broj xx koji se dodaje na a; rezultat se smešta u a
20. [(0x3c, xx)](#_sub_a,_xx) – sub a, xx; oduzima broj xx od a; prvi bajt je opcode (0x3c), a drugi je broj xx koji se oduzima od a; rezultat se smešta u a
21. [(0x40)](#_cmp_a,_b) – cmp a, b; poredi a i b i ako su jednaki, zero je jedinica
22. [(0x42)](#_cmp_a,_[b]) – cmp a, [b]; poredi registar a i memorijsku lokaciju na koju ukazuje registar b, i ako su jednaki, zero je 1
23. [(0x46, xx)](#_cmp_a,_xx) – cmp a, xx; poredi a i broj xx; ako su isti, postavlja zero na 1
24. [(0x4a)](#_dec_a_(0x60)) – dec a; smanjuje vrednost sadržaja registra a za jedan
25. [(0x4c)](#_inc_a_(0x64)) – inc a; povećava vrednost sadržaja registra a za jedan
26. [(0x4e)](#_dec_b_) – dec b; smanjuje vrednost sadržaja registra b za jedan
27. [(0x50)](#_inc_b_(0x6c)) – inc b; povećava vrednost sadržaja registra b za jedan
28. [(0x52)](#_dec_[b]_) – dec [b]; umanjuje sadržaj memorijske lokacije na koju ukazuje b za jedan
29. [(0x56)](#_inc_[b]_) – inc [b]; povećava sadržaj memorijske lokacije na koju ukazuje b za jedan
30. [(0x5a)](#_and_a,_b) – and a, b; konjukcija; a 🡨 a and b
31. [(0x5c)](#_or_a,_b) – or a, b; disjunkcija; a 🡨 a or b
32. [(0x5e)](#_and_a,_[b]) – and a, [b]; računa konjukciju a i memorijske lokacije na koju ukazuje b; postavlja sve flegove
33. [(0x62)](#_or_a,_[b]) – or a, [b]; računa disjunkciju a i memorijske lokacije na koju ukazuje b; postavlja sve flegove
34. [(0x66)](#_and_a,_xx) – and a, xx; računa konjukciju a i broja xx; postavlja sve flegove
35. [(0x6a)](#_or_a,_xx) – or a, xx; računa disjunkciju a i broja xx; postavlja sve flegove
36. [(0x6e)](#_neg_a_(0x6e)) – neg a; potpuni komplement; a 🡨 a’
37. [(0x70, xx)](#_push_xx_(0xcc,) – push xx; stavlja na stek broj xx, koji je odmah posle opcode-a (0x70)
38. [(0x74)](#_push_a_(0x38)) – push a; stavlja na stek sadržaj registra a (i povećava sp za jedan)
39. [(0x76)](#_pop_a_(0x40)) – pop a; skida sa steka broj, stavlja ga u a i smanjuje sp za jedan
40. [(0x78)](#_push_b_(0x48)) – push b; stavlja na stek sadržaj registra b (i povećava sp za jedan)
41. [(0x7a)](#_pop_b_(0x50)) – pop b; skida sa steka broj, stavlja ga u b i smanjuje sp za jedan
42. [(0x7c, xx)](#_jmp_xx_) – jmp xx; prvi bajt je opcode (0x7c), drugi bajt je adresa na koju se skače
43. [(0x7e, xx)](#_jz_xx_) – jz xx; skače na adresu xx, ako zero flag jeste 1; prvi bajt je opcode (0x7e), drugi bajt je adresa na koju se skače; ako je zero flag nula, onda ide na sledeću instrukciju u memoriji
44. [(0x82, xx)](#_jnz_xx_) – jnz xx; skače na adresu xx, ako zero flag nije 1; prvi bajt je opcode (0x82), drugi bajt je adresa na koju se skače; ako je zero flag jedinica, onda ide na sledeću instrukciju u memoriji
45. [(0x86, xx)](#_call_x_) – call xx; poziva potprogram na adresi xx; pre toga na stek stavlja adresu sledeće instrukcije, da bi se uz pomoć ret moglo vratiti u glavni program, na tu adresu.
46. [(0x8a)](#_ret__(0x78)) – ret; povratak iz potprograma (skida povratnu adresu sa steka i stavlja je u pc)
47. [(0x8c, xx)](#_st_a,_[xx]_1) – st a, [xx]; snima u lokaciju xx sadržaj registra a
48. [(0x90, xx)](#_st_b,_[xx]) – st b, [xx]; snima u lokaciju xx sadržaj registra b
49. [(0x94)](#_st_a,_[b]) – st a, [b]; snima sadržaj a u lokaciju na koju ukazuje registar b
50. [(0x96, xx)](#_st_a,_[b) – st a, [b ± xx]; smešta u sadržaj memorijske lokacije na koju ukazuje zbir/razlika (b ± xx) registar a
51. [(0x9a, xx)](#_jp_xx__1) – jp xx; skače na adresu xx, ako je POSITIVE flag 1; ako nije, nastavlja dalje
52. [(0x9e, xx)](#_jnp_xx_) – jnp xx; skače na adresu xx, ako je POSITIVE flag 0; ako nije, nastavlja dalje
53. [(0xa2, xx)](#_jo_xx_) – jo xx; skače na adresu xx, ako je OVERFLOW flag 1; ako nije, nastavlja dalje
54. [(0xa6, xx)](#_jno_xx_) – jno xx; skače na adresu xx, ako je OVERFLOW flag 0; ako nije, nastavlja dalje
55. [(0xaa, xx)](#_neg_a_(0x14)) – and a, [b ± xx]; konjukcija registra a i sadržaja memorijske lokacije na koju ukazuje zbir b ± xx
56. [(0xb0, xx)](#_or_a,_[b) – or a, [b ± xx]; disjunkcija registra a i sadržaja memorijske lokacije na koju ukazuje zbir b ± xx
57. [(0xb6)](#_shl_a,_b_1) – shl a, b; a ← a << b; pomera sadržaj registra a u levo b puta; vrednost pomeranja je maksimalno 7
58. [(0xb8)](#_shr_a,_b) – shr a, b; a ← a >> b; pomera sadržaj registra a u desno b puta; vrednost pomeranja je maksimalno 7
59. [(0xba)](#_shl_a,_[b]) – shl a, [b]; pomera sadržaj registra a u levo onoliko puta koliko iznosi sadržaj memorijske lokacije na koju ukazuje b
60. [(0xbe)](#_shr_a,_[b]) – shr a, [b]; pomera sadržaj registra a u desno onoliko puta koliko iznosi sadržaj memorijske lokacije na koju ukazuje b
61. [(0xc2, xx)](#_shl_a,_[b) – shl a, [b ± xx]; pomera sadržaj registra u levo a onoliko puta koliko iznosi sadržaj memorijske lokacije na koju ukazuje zbir/razlika (b ± xx)
62. [(0xc8, xx)](#_shr_a,_[b) – shr a, [b ± xx]; pomera sadržaj registra a u desno onoliko puta koliko iznosi sadržaj memorijske lokacije na koju ukazuje zbir/razlika (b ± xx)
63. [(0xce, xx)](#_shl_a,_xx) – shl a, xx; sabira a i broj xx; prvi bajt je opcode (0xce), a drugi je broj xx koji se dodaje na a; rezultat se smešta u a
64. [(0xd2, xx)](#_shr_a,_xx) – shr a, xx; oduzima broj xx od a; prvi bajt je opcode (0xd2), a drugi je broj xx koji se oduzima od a; rezultat se smešta u a
65. [(0xd6, xx)](#_callz_xx_) – callz xx; poziva potprogram na adresi xx ako je ZERO flag 1; pre toga na stek stavlja adresu sledeće instrukcije, da bi se uz pomoć ret moglo vratiti u glavni program, na tu adresu.
66. [(0xdc, xx)](#_callnz_xx_) – callnz xx; poziva potprogram na adresi xx ako je ZERO flag 0; pre toga na stek stavlja adresu sledeće instrukcije, da bi se uz pomoć ret moglo vratiti u glavni program, na tu adresu.
67. [(0xe2, xx)](#_callp_xx_) – callp xx; poziva potprogram na adresi xx ako je POSITIVE flag 1; pre toga na stek stavlja adresu sledeće instrukcije, da bi se uz pomoć ret moglo vratiti u glavni program, na tu adresu.
68. [(0xe8, xx)](#_callnp_xx_) – callnp xx; poziva potprogram na adresi xx ako je POSITIVE flag 0; pre toga na stek stavlja adresu sledeće instrukcije, da bi se uz pomoć ret moglo vratiti u glavni program, na tu adresu.
69. [(0xee, xx)](#_callo_xx_) – callo xx; poziva potprogram na adresi xx ako je OVERFLOW flag 1; pre toga na stek stavlja adresu sledeće instrukcije, da bi se uz pomoć ret moglo vratiti u glavni program, na tu adresu.
70. [(0xf4, xx)](#_callno_xx_) – callno xx; poziva potprogram na adresi xx ako je OVERFLOW flag 0; pre toga na stek stavlja adresu sledeće instrukcije, da bi se uz pomoć ret moglo vratiti u glavni program, na tu adresu.
71. [(0xfa)](#_ld_a,_h_2) – ld a, h; dobavlja gornji bajt od rezultata množenja, ili ostatak pri deljenju iz HIGH-BYTE registra u ALU i smešta ga u registar a

# Drugi blok instrukcija, čija adresa u firmware-u počinje od 0x100, a imaju neparan OPCODE

Svi OPCODE-ovi ove grupe instrukcija su neparni, ali se u hardveru pretvore u parne (najniži bit se postavi na nulu) i saberu sa 0x100. Tako, na primer, instrukcija **div a, b** sa OPCODE-om **0x15** se prvo pretvori u parnu (**0x14**) i sabere sa **0x100**, pa je konačna adresa u frimware-u **0x114**.

Ovo je urađeno da bi se dobilo još 256 lokacija u firmware-u za sve naredne instrukcije. Do ovog trenutka, potrošeno je prvih 256 lokacija u formware-u, a bilo je potrebno dodati još instrukcija koje ne bismo imali gde da implementiramo. Ovako je ukupan kapacitet firmware-a 512 lokacija (firmware ROM ima 9-bitnu adresnu magistralu).

1. [(0x01)](#_mul_a,_b) – mul a, b; a ← a \* b; množi a i b i rezultat smešta u a (donjih 8 bitova); gornjih osam bitova je smešteno u HIGH-BYTE izlazu od ALU; za dobavljanje gornjeg bajta koristi se[: ld a, h](#_ld_a,_h_2)
2. [(0x15)](#_div_a,_b) – div a, b; a ← a / b; podeli a sa b i rezultat smešta u a; ostatak je smešten u HIGH-BYTE izlazu od ALU; za dobavljanje gornjeg bajta koristi se[: ld a, h](#_ld_a,_h_2)
3. [(0x29)](#_cmp_b,_xx) – cmp b, xx; poredi b i broj xx; ako su isti, postavlja zero na 1
4. [(0x2d, xx)](#_ld_c,_xx) – ld c, xx; napuni registar c brojem xx koji sledi odmah posle OPCODE-a
5. [(0x2f)](#_st_a,_[c]_1) – st a, [c]; napuni memorijsku lokaciju na koju ukazuje registar c sadržajem registra a
6. [(0x31, xx)](#_st_a,_[c) – st a, [c ± xx]; stavi u memorijsku lokaciju na koju ukazuje zbir registra c i broja xx sadrzaj registra a
7. [(0x35)](#_dec_c_(0x35)) – dec c; smanji c za jedan
8. [(0x37)](#_inc_c_(0x37)) – inc c; povećaj c za jedan
9. [(0x39)](#_ld_c,_a) – ld c, a; u c stavi sadržaj registra a
10. [(0x3b)](#_ld_c,_b) – ld c, b; u c stavi sadržaj registra b
11. [(0x3d)](#_push_c_(0x3d)_1) – push c; stavi c na stek
12. [(0x3f)](#_pop_c_(0x3f)) – pop c; pokupi sa steka broj i stavi u c
13. [(0x41, xx)](#_mul_a,_[b) – mul a, [b ± xx]; množi a i sadržaj memorijske lokacije na koju ukazuje zbir registra b i broja xx; donjih 8 bita je u registru a, a gornjih 8 bita je u HIGH-BYTE registru ALU; za dobavljanje HIGH-BYTE koristi se[: ld a, h](#_ld_a,_h_2)
14. [(0x47, xx)](#_div_a,_[b) – div a, [b ± xx]; deli a i sadržaj memorijske lokacije na koju ukazuje zbir registra b i broja xx; rezultat je u registru a, a ostatak u HIGH-BYTE registru ALU; za dobavljanje HIGH-BYTE koristi se[: ld a, h](#_ld_a,_h_2)
15. [(0x4d, xx)](#_ld_a,_h_1) – mul a, xx; množi a i broj xx; viši bajt je u registru HIGH-BYTE od ALU
16. [(0x51, xx)](#_div_a,_xx) – diva a, xx; deli a brojem xx; ostatak je u HIGH-BYTE registru od ALU
17. [(0x55)](#_ld_a,_c) – ld a, c; u registar a stavlja vrednost registra c
18. [(0x57)](#_ld_b,_c) – ld b, c; u registar b stavlja vrednost registra c
19. [(0x59)](#_ld_a,_[c]) – ld a, [c]; u registar a smešta sadržaj memorijske lokacije na koju ukazuje registar c
20. [(0x5b, xx)](#_ld_a,_[c) – ld a, [c ± xx]; u registar a smešta sadržaj memorijske lokacije na koju ukazuje zbir registra c i broja xx
21. [(0x5f)](#_mul_a,_c) – mul a, c; množi a i c
22. [(0x61)](#_div_a,_c) – div a, c; deli a brojem u c
23. [(0x63)](#_xor_a,_b) – xor a, b
24. [(0x65)](#_xor_a,_c) – xor a, c
25. [(0x67)](#_xor_a,_[b) – xor a, [b± xx]; uradi xor između registra a i lokacije na koju ukazuje b± xx
26. [(0x6d)](#_xor_a,_[b]) – xor a, [b]; uradi xor između registra a i lokacije na koju ukazuje b
27. [(0x71)](#_cmp_a,_[b_1) – cmp a, [b± xx]; poredi sadržaj registra a i lokacije na koju ukazuje b± xx
28. [(0x77)](#_add_a,_c) – add a, c; sabira a i c, a rezultat ide nazad u a
29. [(0x79)](#_sub_a,_c) – sub a, c; oduzima c od a i rezultat ide nazad u a
30. [(0x7b, xx)](#_st_a,_[b_1) – st a, [b + c ± xx]; stavlja sadržaj registra a u memorijsku lokaciju na koju ukazuje zbir registrara b i c, kao i broja xx
31. [(0x81, xx)](#_cmp_[b_±) – cmp [b± xx], a; poredi sadržaj lokacije na koju ukazuje b± xx i sadržaj registra a
32. [(0xff)](#_halt__(0xff)_1) – halt; zaustavlja računar

Mikrokod

Mikrokod predstavlja skup stanja svih upravljačkih signala u nekom trenutku, odn. u nekom taktu. Takt procesora se dovodi na brojač, a iz brojača izlazi binarni broj (koliko je brojač izbrojao). Taj broj se dovodi na dva paralalno spojena ROM-a i njihov izlaz je ukupno 64 bita (2 x 32 bita). Tih 64 bita predstavlja sve upravljačke signale koji se koriste u jednom trenutku.

Zašto 2 ROM-a? LogiSim ima maksimalnu širinu reči u ROM-u od 32 bita, što nije dovoljno za sve signale, pa su stavljena dva paralelna ROM-a, tako da u svakom trenutku ista adresa ulazi u oba ROM-a, gde imamo iz oba po 32 bita, što u totalu daje 64 bita.

## Fetch

Ovo se izvršava za svaku instrukciju. Dobavlja se OPCODE od instrukcije na tekućoj adresi u memoriji, na koju ukazuje PC, pa se taj OPCODE smešta u IR, a zatim se sadržaj IR koristi da bi se skočilo u mikrokod koji se nalazi na adresi na koju ukazuje OPCODE. PC se svakako poveća za jedan, pa ili ukazuje na sledeću instrukciju, ili na operand.

OPCODE je adresa u ROM-u, gde se nalaze mikroinstrukcije. Sve mikroinstrukcije su zapravo stanja upravljačkih signala i u svakom trenutku oni upravljaju ponašanjem procesora i računara.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0 | address\_bus ← PC, data\_bus←memory, IR ← data\_bus | PC-OE,  LOAD-MAR,  MEM-RD,  LOAD-IR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1001 1000  0x98  LOAD-MAR,  LOAD-IR, PC-OE | 0000 0000  0x00 |
| 1 | PC ← PC + 1 | PC-INC  LOAD-PC | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0010  0x02  LOAD-PC | 0010 0000  0x20  PC-INC |
| 2 | firmware\_addr ← IR  (dekodiranje) | IR-OE, MICRO-CODE-COUNTER | 0001 0000  0x10  MICRO-CODE-COUNTER | 0000 0000  0x00 | 0010 0000  0x20  IR-OE | 0000 0000  0x00 |

## nop (0x00)

Ne radi ništa, odn. odmah radi fetch sledeće instrukcije u memoriji.

# LOAD operacije

## ld a, b (0x04)

Stavlja vrednost B u A. PC se poveća za jedan da ukaže na sledeću instrukciju.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x04 | data\_bus← B, A← data\_bus, | OE-B,  LOAD-A | 0100 1000  0x48  LOAD-A, OE-B | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 |
| 0x05 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## ld b, a (0x06)

Stavlja vrednost A u B. PC se poveća za jedan da ukaže na sledeću instrukciju.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x06 | data\_bus← A, B← data\_bus | OE-A,  LOAD-B | 0010 0100  0x24  LOAD-B, OE-A | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 |
| 0x07 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## ld a, xx (0x08, xx)

Stavlja broj u A. Broj se nalazi u memoriji odmah posle OPCODE-a (0x0a). PC se poveća za jedan da ukaže na sledeću instrukciju.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x08 | address\_bus ← PC, data\_bus←memory, A ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-A | 0100 0001  0x41  LOAD-A, MEM-RD | 0000 0000  0x00 | 1000 1010  0x8a  LOAD-MAR, PC-OE, LOAD-PC | 0010 0000  0x20  PC-INC |
| 0x09 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## ld b, xx (0x0a, xx)

Stavlja broj u B. Broj se nalazi u memoriji odmah posle OPCODE-a (0x1a). PC se poveća za jedan da ukaže na sledeću instrukciju.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x0a | address\_bus ← PC, data\_bus←memory, B ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-B | 0010 0001  0x21  LOAD-B, MEM-RD | 0000 0000  0x00 | 1000 1010  0x8a  LOAD-MAR, PC-OE, LOAD-PC | 0010 0000  0x20  PC-INC |
| 0x0b | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## ld a, [b] (0x0c)

Stavlja sadržaj memorijske lokacije na koju ukazuje B u A. PC se poveća za jedan da ukaže na sledeću instrukciju.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x0c | address\_bus← B, A← data\_bus,  PC←PC+1 | OE-B, B-ADDR-BUS, MEM-RD, LOAD-A | 0100 1001  0x49  LOAD-A,  OE-B, MEM-RD | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0010  0x02  B-ADDR-BUS |
| 0x0d | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## ld a, [b ± xx] (0x0e, xx)

Stavlja sadržaj memorijske lokacije na koju ukazuje (B+xx) u A. PC se poveća za jedan da ukaže na sledeću instrukciju.

Ako je operator oduzimanja (b - xx), onda se u memoriji posle OPCODE-a nalazi potpuni komplement broja xx.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x0e | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x0f | ALU-A ← B  ALU-B ← MBR  MBR←ALU-A + ALU-B | ALU-A-LOAD-B,  ALU-B-LOAD-MBR  ALU-ADD, ALU-GET-RES,  LOAD-MBR | 0000 0000  0x00 | 0000 1110  0x0e  ALU-ADD,  ALU-GET-RES,  ALU-A-LOAD-B | 0000 0001  0x01  LOAD-MBR | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x10 | address\_bus← MBR, A← data\_bus, | MBR-OE,  LOAD-MAR,  MEM-RD,  LOAD-A | 0100 0001  0x41  LOAD-A,  MEM-RD | 0000 0000  0x00 | 1000 0100  0x84  LOAD-MAR,  MBR-OE | 0000 0000  0x00 | 0000 0000  0x00 |
| 0x11 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## ld sp, xx (0x12, xx)

SP ← xx; puni SP brojem (postavljanje steka). Broj se nalazi odmah posle OPCODE-a (0x2a).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x12 | address\_bus ← PC, data\_bus←memory, SP ← data\_bus,  PC ← PC+1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-SP,  PC-INC  LOAD-PC | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1010  0x8a  LOAD-MAR, PC-OE,  LOAD-PC | 1010 0000  0xa0  LOAD-SP,  PC-INC |
| 0x13 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## ld b, sp (0x14)

Stavlja vrednost SP u B. PC se poveća za jedan da ukaže na sledeću instrukciju.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x14 | data\_bus← SP, B← data\_bus | SP-OE,  LOAD-B | 0010 0000  0x20  LOAD-B | 0000 0000  0x00 | 0000 0000  0x00 | 0100 0000  0x40  SP-OE |
| 0x15 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## ld sp, b (0x16)

Stavlja vrednost SP u B. PC se poveća za jedan da ukaže na sledeću instrukciju.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x16 | data\_bus← B, SP← data\_bus | OE-B,  LOAD-SP | 0000 1000  0x08  OE-B | 0000 0000  0x00 | 0000 0000  0x00 | 1000 0000  0x80  LOAD-SP |
| 0x17 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

# ADD i SUB operacije

## add sp, xx (0x1c, xx)

SP ← SP + xx

Sabira SP i xx i smešta rezultat nazad u SP. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x1c | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x1d | ALU-A ← SP  ALU-B ← MBR  SP <-  ALU-A+ ALU-B | ALU-ADD  ALU-GET-RES,  ALU-A-LOAD-SP  ALU-B-LOAD-MBR  LOAD-SP,  MARK-FLAG | 0000 0000  0x00 | 0000 1100  0x0c  ALU-ADD, ALU-GET-RES | 0000 0000  0x00 | 1000 0100  0x84  LOAD-SP  MARK-FLAG | 0100 1000  0x48  ALU-A-LOAD-SP  ALU-B-LOAD-MBR |
| 0x1e | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## sub sp, xx (0x20, xx)

SP ← SP - xx

Oduzima xx od SP i smešta rezultat nazad u SP. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x20 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x21 | ALU-A ← SP  ALU-B ← MBR  SP <-  ALU-A- ALU-B | ALU-SUB  ALU-GET-RES,  ALU-A-LOAD-SP  ALU-B-LOAD-MBR  LOAD-SP,  MARK-FLAG | 0000 0000  0x00 | 0001 0100  0x14  ALU-SUB, ALU-GET-RES | 0000 0000  0x00 | 1000 0100  0x84  LOAD-SP  MARK-FLAG | 0100 1000  0x48  ALU-A-LOAD-SP  ALU-B-LOAD-MBR |
| 0x22 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## add a, b (0x18)

A ← A + B

Sabira A i B i rezultat smešta u A. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x18 | ALU-A ← A  ALU-B ← B  A ← ALU-A + ALU-B | ALU-ADD  ALU-GET-RES,  LOAD-A,  MARK-FLAG | 0100 0000  0x40  LOAD-A | 0000 1100  0x0c  ALU-ADD, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG |
| 0x19 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## sub a, b (0x1a)

A ← A – B

Oduzima B od A i smešta rezultat nazad u A. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x1a | ALU-A ← A  ALU-B ← B  A <-  ALU-A - ALU-B | ALU-SUB  ALU-GET-RES,  LOAD-A,  MARK-FLAG | 0100 0000  0x40  LOAD-A | 0001 0100  0x14  ALU-SUB, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG |
| 0x1b | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## add a, [b] (0x24)

A ← A + [B]

Sabira A i sadržaj memorijske lokacije na koju ukazuje B i rezultat smešta u A. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x24 | address\_bus←B  MBR ←memory | OE-B,  B-ADDR-BUS,  LOAD-MAR,  MEM-RD, LOAD-MBR, | 0000 1001  0x09  OE-B, MEM-RD | 0000 0000  0x00 | 1000 0001  0x81  LOAD-MAR,  LOAD-MBR | 0000 0000  0x00 | 0000 0010  0x02  B-ADDR-BUS |
| 0x25 | ALU-A ← A  ALU-B ← MBR  A <- ALU-A + ALU-B | ALU-ADD  ALU-GET-RES,  LOAD-A,  MARK-FLAG,  ALU-B-LOAD-MBR | 0100 0000  0x40  LOAD-A | 0000 1100  0x0c  ALU-ADD, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x26 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## sub a, [b] (0x28)

A ← A - [b]

Sabira A i sadržaj memorijske lokacije na koju ukazuje B i rezultat smešta u A. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x28 | address\_bus←B  MBR ←memory | OE-B,  B-ADDR-BUS,  LOAD-MAR,  MEM-RD, LOAD-MBR, | 0000 1001  0x09  OE-B, MEM-RD | 0000 0000  0x00 | 1000 0001  0x81  LOAD-MAR,  LOAD-MBR | 0000 0000  0x00 | 0000 0010  0x02  B-ADDR-BUS |
| 0x29 | ALU-A ← A  ALU-B ← MBR  A <- ALU-A - ALU-B | ALU-SUB  ALU-GET-RES,  LOAD-A,  MARK-FLAG,  ALU-B-LOAD-MBR | 0100 0000  0x40  LOAD-A | 0001 0100  0x14  ALU-SUB, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x2a | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## add a, [b ± xx] (0x2c, xx)

sabira registar a i sadržaj memorijske lokacije na koju ukazuje (B+xx). PC se poveća za jedan da ukaže na sledeću instrukciju. Flegovi se postavljaju nakon što se dobije rezultat.

Ako je operator oduzimanja (b - xx), onda se u memoriji posle OPCODE-a nalazi potpuni komplement broja xx.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x2c | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x2d | ALU-A ← B  ALU-B ← MBR  MBR←ALU-A + ALU-B | ALU-A-LOAD-B,  ALU-B-LOAD-MBR  ALU-ADD, ALU-GET-RES,  LOAD-MBR | 0000 0000  0x00 | 0000 1110  0x0e  ALU-ADD,  ALU-GET-RES,  ALU-A-LOAD-B | 0000 0001  0x01  LOAD-MBR | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x2e | address\_bus← MBR,  MBR← data\_bus, | MBR-OE,  LOAD-MAR,  MEM-RD,  LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 0101  0x85  LOAD-MAR,  MBR-OE,  LOAD-MBR | 0000 0000  0x00 | 0000 0000  0x00 |
| 0x2f | ALU-A←A  ALU-B← MBR,  A←ALU-A + ALU-B | ALU-B-LOAD-MBR,  ALU-ADD,  LOAD-A | 0100 0000  0x40  LOAD-A | 0000 1100  0x0c  ALU-ADD,  ALU-GET-RES | 0000 0000  0x00 | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x30 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## sub a, [b ± xx] (0x32, xx)

Oduzima od registra a sadržaj memorijske lokacije na koju ukazuje (B+xx). PC se poveća za jedan da ukaže na sledeću instrukciju. Flegovi se postavljaju nakon što se dobije rezultat.

Ako je operator oduzimanja (b - xx), onda se u memoriji posle OPCODE-a nalazi potpuni komplement broja xx.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x32 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x33 | ALU-A ← B  ALU-B ← MBR  MBR←ALU-A + ALU-B | ALU-A-LOAD-B,  ALU-B-LOAD-MBR  ALU-ADD, ALU-GET-RES,  LOAD-MBR | 0000 0000  0x00 | 0000 1110  0x0e  ALU-ADD,  ALU-GET-RES,  ALU-A-LOAD-B | 0000 0001  0x01  LOAD-MBR | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x34 | address\_bus← MBR,  MBR← data\_bus, | MBR-OE,  LOAD-MAR,  MEM-RD,  LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 0101  0x85  LOAD-MAR,  MBR-OE,  LOAD-MBR | 0000 0000  0x00 | 0000 0000  0x00 |
| 0x35 | ALU-A←A  ALU-B← MBR,  A←ALU-A - ALU-B | ALU-B-LOAD-MBR,  ALU-SUB,  LOAD-A | 0100 0000  0x40  LOAD-A | 0001 0100  0x14  ALU-SUB  ALU-GET-RES | 0000 0000  0x00 | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x36 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## add a, xx (0x38, xx)

Sabira A i broj. Broj se nalazi u memoriji odmah posle OPCODE-a (0x0a). PC se poveća za jedan da ukaže na sledeću instrukciju. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x38 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x39 | ALU-A ← A  ALU-B ← MBR  A <- ALU-A + ALU-B | ALU-ADD  ALU-GET-RES,  LOAD-A,  MARK-FLAG,  ALU-B-LOAD-MBR | 0100 0000  0x40  LOAD-A | 0000 1100  0x0c  ALU-ADD, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x3a | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## sub a, xx (0x3c, xx)

Sabira A i broj. Broj se nalazi u memoriji odmah posle OPCODE-a (0x0a). PC se poveća za jedan da ukaže na sledeću instrukciju. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x3c | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x3d | ALU-A ← A  ALU-B ← MBR  A <- ALU-A - ALU-B | ALU-SUB  ALU-GET-RES,  LOAD-A,  MARK-FLAG,  ALU-B-LOAD-MBR | 0100 0000  0x40  LOAD-A | 0001 0100  0x14  ALU-SUB, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x3e | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

# CMP operacije

## cmp a, b (0x40)

Poredi A i B. Ako su jednaki, postavlja ZERO flag na jedinicu.

To radi tako što oduzme B od A i ne smesti rezultat. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x40 | ALU-A ← A  ALU-B ← B  ALU-A - ALU-B | ALU-SUB  ALU-GET-RES,  MARK-FLAG | 0000 0000  0x00 | 0001 0100  0x14  ALU-SUB, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG |
| 0x41 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## cmp a, [b] (0x42)

Poredi A i sadržaj memorijske lokacije na koju ukazuje B. Ako su jednaki, postavlja ZERO flag na jedinicu.

To radi tako što oduzme sadržaj memorijske lokacije na koju ukazuje B od A i ne smesti rezultat. Flegovi se postavljaju nakon što se dobije rezultat.

U toku računanja, vrednost iz memorijske lokacije na koju ukazuje B smešta u MBR.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x42 | address\_bus←B  MBR ←memory | OE-B,  B-ADDR-BUS,  LOAD-MAR,  MEM-RD, LOAD-MBR, | 0000 1001  0x09  OE-B, MEM-RD | 0000 0000  0x00 | 1000 0001  0x81  LOAD-MAR,  LOAD-MBR | 0000 0000  0x00 | 0000 0010  0x02  B-ADDR-BUS |
| 0x43 | ALU-A ← A  ALU-B ← MBR  RES <-  ALU-A - ALU-B | ALU-SUB  ALU-GET-RES,  MARK-FLAG,  ALU-B-LOAD-MBR | 0000 0000  0x00 | 0001 0100  0x14  ALU-SUB, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x44 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## cmp a, [b ± xx] (0x71, xx)

Poredi registar a i sadržaj memorijske lokacije na koju ukazuje (B+xx). PC se poveća za jedan da ukaže na sledeću instrukciju. Flegovi se postavljaju nakon što se dobije rezultat.

Ako je operator oduzimanja (b - xx), onda se u memoriji posle OPCODE-a nalazi potpuni komplement broja xx.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x170 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x171 | ALU-A ← B  ALU-B ← MBR  MBR←ALU-A + ALU-B | ALU-A-LOAD-B,  ALU-B-LOAD-MBR  ALU-ADD, ALU-GET-RES,  LOAD-MBR | 0000 0000  0x00 | 0000 1110  0x0e  ALU-ADD,  ALU-GET-RES,  ALU-A-LOAD-B | 0000 0001  0x01  LOAD-MBR | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x172 | address\_bus← MBR,  MBR← data\_bus, | MBR-OE,  LOAD-MAR,  MEM-RD,  LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 0101  0x85  LOAD-MAR,  MBR-OE,  LOAD-MBR | 0000 0000  0x00 | 0000 0000  0x00 |
| 0x173 | ALU-A←A  ALU-B← MBR,  REZ←ALU-A - ALU-B | ALU-B-LOAD-MBR,  ALU-SUB | 0000 0000  0x00 | 0001 0100  0x14  ALU-SUB  ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x174 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## cmp a, xx (0x46, xx)

poredi A i broj. Broj se nalazi u memoriji odmah posle OPCODE-a (0xbc). PC se poveća za jedan da ukaže na sledeću instrukciju. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x46 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x47 | ALU-A ← A  ALU-B ← MBR | ALU-SUB  ALU-GET-RES,  MARK-FLAG,  ALU-B-LOAD-MBR | 0000 0000  0x00 | 0001 0100  0x14  ALU-SUB, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x48 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## cmp b, xx (0x29, xx)

poredi B i broj. Broj se nalazi u memoriji odmah posle OPCODE-a (0x29). PC se poveća za jedan da ukaže na sledeću instrukciju. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x128 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x129 | ALU-A ← B  ALU-B ← MBR | ALU-SUB,  ALU-GET-RES,  MARK-FLAG,  ALU-A-LOAD-B,  ALU-B-LOAD-MBR | 0000 0000  0x00 | 0001 0110  0x16  ALU-SUB, ALU-GET-RES,  ALU-A-LOAD-B | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x12a | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

# INC i DEC

## dec a (0x4a)

Smanji A za 1. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x4a | A ← A - 1 | ONE-TO-ALU,  ALU-SUB, ALU-GET-RES, LOAD-A, MARK-FLAG | 0100 0000  0x40  LOAD-A | 0001 0101  0x15  ALU-SUB, ALU-GET-RES,  ONE-TO-ALU | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG |
| 0x4b | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## inc a (0x4c)

Povećaj A za 1. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x4c | A ← A + 1 | ONE-TO-ALU,  ALU-ADD, ALU-GET-RES, LOAD-A, MARK-FLAG | 0100 0000  0x40  LOAD-A | 0000 1101  0x0d  ALU-ADD, ALU-GET-RES,  ONE-TO-ALU | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG |
| 0x4d | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## dec b (0x4e)

Smanji B za 1. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x4e | B ← B - 1 (prvi takt) | ONE-TO-ALU,  ALU-SUB, ALU-GET-RES, LOAD-B, MARK-FLAG,  ALU-A-LOAD-B | 0010 0000  0x20  LOAD-A | 0001 0111  0x17  ALU-SUB, ALU-GET-RES,  ALU-A-LOAD-B,  ONE-TO-ALU | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG |
| 0x4f | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## inc b (0x50)

Povećaj A za 1. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x50 | B ← B + 1 | ONE-TO-ALU,  ALU-ADD, ALU-GET-RES,  ALU-A-LOAD-B,  LOAD-B, MARK-FLAG | 0010 0000  0x20  LOAD-B | 0000 1111  0x0f  ALU-ADD, ALU-GET-RES,  ALU-A-LOAD-B,  ONE-TO-ALU | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG |
| 0x51 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## dec [b] (0x52)

[b] ← [b] - 1

Smanjuje sadržaj memorijske lokacije na koju ukazuje B za jedan. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x52 | address\_bus←B  MBR ←memory | OE-B,  B-ADDR-BUS,  LOAD-MAR,  MEM-RD, LOAD-MBR, | 0000 1001  0x09  OE-B, MEM-RD | 0000 0000  0x00 | 1000 0001  0x81  LOAD-MAR,  LOAD-MBR | 0000 0000  0x00 | 0000 0010  0x02  B-ADDR-BUS |
| 0x53 | ALU-A ← MBR  ALU-B ← 1  MBR <- ALU-A - ALU-B | ALU-SUB  ALU-GET-RES,  LOAD-MBR,  MARK-FLAG,  ALU-A-LOAD-MBR  ONE-TO-ALU | 0000 0000  0x00 | 0001 0101  0x15  ALU-SUB, ALU-GET-RES,  ONE-TO-ALU | 0000 0001  0x01  LOAD-MBR | 0000 0100  0x04  MARK-FLAG | 0001 0000  0x10  ALU-A-LOAD-MBR |
| 0x54 | address\_bus ← B, memory←MBR | OE-B,  B-ADDR-BUS  MBR-OE,  MEM-WR | 0000 1010  0x0a  OE-B,  MEM-WR | 0000 0000  0x00 | 0000 0100  0x04  MBR-OE | 0000 0000  0x00 | 0000 0010  0x02  B-ADDR-BUS |
| 0x55 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## inc [b] (0x56)

[b] ← [b] + 1

Povećava sadržaj memorijske lokacije na koju ukazuje B za jedan. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x56 | address\_bus←B  MBR ←memory | OE-B,  B-ADDR-BUS,  LOAD-MAR,  MEM-RD, LOAD-MBR, | 0000 1001  0x09  OE-B, MEM-RD | 0000 0000  0x00 | 1000 0001  0x81  LOAD-MAR,  LOAD-MBR | 0000 0000  0x00 | 0000 0010  0x02  B-ADDR-BUS |
| 0x57 | ALU-A ← MBR  ALU-B ← 1  MBR <- ALU-A + ALU-B | ALU-ADD  ALU-GET-RES,  LOAD-MBR,  MARK-FLAG,  ALU-A-LOAD-MBR  ONE-TO-ALU | 0000 0000  0x00 | 0000 1101  0x0d  ALU-ADD, ALU-GET-RES,  ONE-TO-ALU | 0000 0001  0x01  LOAD-MBR | 0000 0100  0x04  MARK-FLAG | 0001 0000  0x10  ALU-A-LOAD-MBR |
| 0x58 | address\_bus ← B, memory←MBR | OE-B,  B-ADDR-BUS  MBR-OE,  MEM-WR | 0000 1010  0x0a  OE-B,  MEM-WR | 0000 0000  0x00 | 0000 0100  0x04  MBR-OE | 0000 0000  0x00 | 0000 0010  0x02  B-ADDR-BUS |
| 0x59 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

# AND i OR

## and a, b (0x5a)

A ← A && B (AND, konjukcija). Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x5a | ALU-A ← A  ALU-B ← B  RES <-  ALU-A && ALU-B | ALU-AND,  ALU-GET-RES,  LOAD-A,  MARK-FLAG | 0100 0000  0x40  LOAD-A | 0010 0100  0x24  ALU-AND, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG |
| 0x5b | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## or a, b (0x5c)

A ← A || B (OR, disjunkcija). Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x5c | ALU-A ← A  ALU-B ← B  RES <-  ALU-A || ALU-B | ALU-OR,  ALU-GET-RES,  LOAD-A,  MARK-FLAG | 0100 0000  0x40  LOAD-A | 0100 0100  0x44  ALU-OR, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG |
| 0x5d | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## and a, [b] (0x5e)

A ← A && [b]

Računa konjukciju A i sadržaja memorijske lokacije na koju ukazuje B i rezultat smešta u A. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x5e | address\_bus←B  MBR ←memory | OE-B,  B-ADDR-BUS,  LOAD-MAR,  MEM-RD, LOAD-MBR, | 0000 1001  0x09  OE-B, MEM-RD | 0000 0000  0x00 | 1000 0001  0x81  LOAD-MAR,  LOAD-MBR | 0000 0000  0x00 | 0000 0010  0x02  B-ADDR-BUS |
| 0x5f | ALU-A ← A  ALU-B ← MBR  RES <- ALU-A && ALU-B | ALU-AND  ALU-GET-RES,  LOAD-A,  MARK-FLAG,  ALU-B-LOAD-MBR | 0100 0000  0x40  LOAD-A | 0010 0100  0x24  ALU-AND, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x60 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## or a, [b] (0x62)

A ← A || [b]

Računa disjunkciju A i sadržaja memorijske lokacije na koju ukazuje B i rezultat smešta u A. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x62 | address\_bus←B  MBR ←memory | OE-B,  B-ADDR-BUS,  LOAD-MAR,  MEM-RD, LOAD-MBR, | 0000 1001  0x09  OE-B, MEM-RD | 0000 0000  0x00 | 1000 0001  0x81  LOAD-MAR,  LOAD-MBR | 0000 0000  0x00 | 0000 0010  0x02  B-ADDR-BUS |
| 0x63 | ALU-A ← A  ALU-B ← MBR  RES <- ALU-A || ALU-B | ALU-OR  ALU-GET-RES,  LOAD-A,  MARK-FLAG,  ALU-B-LOAD-MBR | 0100 0000  0x40  LOAD-A | 0100 0100  0x44  ALU-AND, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x64 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## and a, xx (0x66, xx)

Sabira A i broj. Broj se nalazi u memoriji odmah posle OPCODE-a (0x0a). PC se poveća za jedan da ukaže na sledeću instrukciju. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x66 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x67 | ALU-A ← A  ALU-B ← MBR  A <- ALU-A && ALU-B | ALU-AND  ALU-GET-RES,  LOAD-A,  MARK-FLAG,  ALU-B-LOAD-MBR | 0100 0000  0x40  LOAD-A | 0010 0100  0x24  ALU-AND, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x68 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## or a, xx (0x6a, xx)

Sabira A i broj. Broj se nalazi u memoriji odmah posle OPCODE-a (0x0a). PC se poveća za jedan da ukaže na sledeću instrukciju. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x6a | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x6b | ALU-A ← A  ALU-B ← MBR  A <- ALU-A || ALU-B | ALU-OR  ALU-GET-RES,  LOAD-A,  MARK-FLAG,  ALU-B-LOAD-MBR | 0100 0000  0x40  LOAD-A | 0100 0100  0x44  ALU-OR, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x6c | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## and a, [b ± xx] (0xaa, xx)

Računa konjukciju registra a i sadržaja memorijske lokacije na koju ukazuje (B+xx). PC se poveća za jedan da ukaže na sledeću instrukciju. Flegovi se postavljaju nakon što se dobije rezultat.

Ako je operator oduzimanja (b - xx), onda se u memoriji posle OPCODE-a nalazi potpuni komplement broja xx.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0xaa | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0xab | ALU-A ← B  ALU-B ← MBR  MBR←ALU-A + ALU-B | ALU-A-LOAD-B,  ALU-B-LOAD-MBR  ALU-ADD, ALU-GET-RES,  LOAD-MBR | 0000 0000  0x00 | 0000 1110  0x0e  ALU-ADD,  ALU-GET-RES,  ALU-A-LOAD-B | 0000 0001  0x01  LOAD-MBR | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0xac | address\_bus← MBR,  MBR← data\_bus, | MBR-OE,  LOAD-MAR,  MEM-RD,  LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 0101  0x85  LOAD-MAR,  MBR-OE,  LOAD-MBR | 0000 0000  0x00 | 0000 0000  0x00 |
| 0xad | ALU-A←A  ALU-B← MBR,  A←ALU-A && ALU-B | ALU-B-LOAD-MBR,  ALU-AND,  LOAD-A | 0100 0000  0x40  LOAD-A | 0010 0100  0x24  ALU-AND,  ALU-GET-RES | 0000 0000  0x00 | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0xae | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## or a, [b ± xx] (0xb0, xx)

Računa disjunkciju registra a i sadržaja memorijske lokacije na koju ukazuje (B+xx). PC se poveća za jedan da ukaže na sledeću instrukciju. Flegovi se postavljaju nakon što se dobije rezultat.

Ako je operator oduzimanja (b - xx), onda se u memoriji posle OPCODE-a nalazi potpuni komplement broja xx.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0xb0 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0xb1 | ALU-A ← B  ALU-B ← MBR  MBR←ALU-A + ALU-B | ALU-A-LOAD-B,  ALU-B-LOAD-MBR  ALU-ADD, ALU-GET-RES,  LOAD-MBR | 0000 0000  0x00 | 0000 1110  0x0e  ALU-ADD,  ALU-GET-RES,  ALU-A-LOAD-B | 0000 0001  0x01  LOAD-MBR | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0xb2 | address\_bus← MBR,  MBR← data\_bus, | MBR-OE,  LOAD-MAR,  MEM-RD,  LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 0101  0x85  LOAD-MAR,  MBR-OE,  LOAD-MBR | 0000 0000  0x00 | 0000 0000  0x00 |
| 0xb3 | ALU-A←A  ALU-B← MBR,  A←ALU-A || ALU-B | ALU-B-LOAD-MBR,  ALU-OR,  LOAD-A | 0100 0000  0x40  LOAD-A | 0100 0100  0x44  ALU-OR,  ALU-GET-RES | 0000 0000  0x00 | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0xb4 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## neg a (0x6e)

A ← A’ (potpuni komplement). Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x6e | ALU-A ← A  RES <-  NEG ALU-A | ALU-NEG,  ALU-GET-RES,  LOAD-A,  MARK-FLAG | 0100 0000  0x40  LOAD-A | 1000 0100  0x84  ALU-NEG, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG |
| 0x6f | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

# Pomeranje u levo i desno

## shl a, b (0xb6)

A ← A << B

Pomera A u levo B puta (b ne veći od 7) i rezultat smešta u A. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Šestih 8 bitova |
| 0xb6 | ALU-A ← A  ALU-B ← B  A ← ALU-A << ALU-B | ALU-SHL  ALU-GET-RES,  LOAD-A,  MARK-FLAG | 0100 0000  0x40  LOAD-A | 0000 0100  0x04  ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG |  |
| 0xb7 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## shr a, b (0xb8)

A ← A >> B

Pomera A u desno (aritmetički) B puta (B ne veći od 7) i smešta rezultat nazad u A. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0xb8 | ALU-A ← A  ALU-B ← B  A <-  ALU-A - ALU-B | ALU-SUB  ALU-GET-RES,  LOAD-A,  MARK-FLAG | 0100 0000  0x40  LOAD-A | 0001 0100  0x14  ALU-SUB, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG |
| 0xb9 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## shl a, [b] (0xba)

A ← A << [B]

Pomera A u levo onoliko puta koliki je sadržaj memorijske lokacije na koju ukazuje B (ne više od 7) i rezultat smešta u A. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0xba | address\_bus←B  MBR ←memory | OE-B,  B-ADDR-BUS,  LOAD-MAR,  MEM-RD, LOAD-MBR, | 0000 1001  0x09  OE-B, MEM-RD | 0000 0000  0x00 | 1000 0001  0x81  LOAD-MAR,  LOAD-MBR | 0000 0000  0x00 | 0000 0010  0x02  B-ADDR-BUS |
| 0xbb | ALU-A ← A  ALU-B ← MBR  A <- ALU-A + ALU-B | ALU-ADD  ALU-GET-RES,  LOAD-A,  MARK-FLAG,  ALU-B-LOAD-MBR | 0100 0000  0x40  LOAD-A | 0000 1100  0x0c  ALU-ADD, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0xbc | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## shr a, [b] (0xbe)

A ← A >> [b]

Pomera desno A onoliko puta koliki je sadržaj memorijske lokacije na koju ukazuje B (ne više od 7) i rezultat smešta u A. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0xbe | address\_bus←B  MBR ←memory | OE-B,  B-ADDR-BUS,  LOAD-MAR,  MEM-RD, LOAD-MBR, | 0000 1001  0x09  OE-B, MEM-RD | 0000 0000  0x00 | 1000 0001  0x81  LOAD-MAR,  LOAD-MBR | 0000 0000  0x00 | 0000 0010  0x02  B-ADDR-BUS |
| 0xbf | ALU-A ← A  ALU-B ← MBR  A <- ALU-A - ALU-B | ALU-SUB  ALU-GET-RES,  LOAD-A,  MARK-FLAG,  ALU-B-LOAD-MBR | 0100 0000  0x40  LOAD-A | 0001 0100  0x14  ALU-SUB, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0xc0 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## shl a, [b ± xx] (0xc2, xx)

Pomera A u levo onoliko puta koliki je sadržaj memorijske lokacije na koju ukazuje (B+xx) (ne više od 7) i rezultat smešta u A.. PC se poveća za jedan da ukaže na sledeću instrukciju.

Ako je operator oduzimanja (b - xx), onda se u memoriji posle OPCODE-a nalazi potpuni komplement broja xx.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0xc2 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0xc3 | ALU-A ← B  ALU-B ← MBR  MBR←ALU-A + ALU-B | ALU-A-LOAD-B,  ALU-B-LOAD-MBR  ALU-ADD, ALU-GET-RES,  LOAD-MBR | 0000 0000  0x00 | 0000 1110  0x0e  ALU-ADD,  ALU-GET-RES,  ALU-A-LOAD-B | 0000 0001  0x01  LOAD-MBR | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0xc4 | address\_bus← MBR,  MBR← data\_bus, | MBR-OE,  LOAD-MAR,  MEM-RD,  LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 0101  0x85  LOAD-MAR,  MBR-OE,  LOAD-MBR | 0000 0000  0x00 | 0000 0000  0x00 |
| 0xc5 | ALU-A←A  ALU-B← MBR,  A←ALU-A + ALU-B | ALU-B-LOAD-MBR,  ALU-ADD,  LOAD-A | 0100 0000  0x40  LOAD-A | 0000 1100  0x0c  ALU-ADD,  ALU-GET-RES | 0000 0000  0x00 | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0xc6 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## shr a, [b ± xx] (0xc8, xx)

Pomera desno A onoliko puta koliki je sadržaj memorijske lokacije na koju ukazuje (B+xx) (ne više od 7) i rezultat smešta u A. PC se poveća za jedan da ukaže na sledeću instrukciju.

Ako je operator oduzimanja (b - xx), onda se u memoriji posle OPCODE-a nalazi potpuni komplement broja xx.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0xc8 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0xc9 | ALU-A ← B  ALU-B ← MBR  MBR←ALU-A + ALU-B | ALU-A-LOAD-B,  ALU-B-LOAD-MBR  ALU-ADD, ALU-GET-RES,  LOAD-MBR | 0000 0000  0x00 | 0000 1110  0x0e  ALU-ADD,  ALU-GET-RES,  ALU-A-LOAD-B | 0000 0001  0x01  LOAD-MBR | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0xca | address\_bus← MBR,  MBR← data\_bus, | MBR-OE,  LOAD-MAR,  MEM-RD,  LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 0101  0x85  LOAD-MAR,  MBR-OE,  LOAD-MBR | 0000 0000  0x00 | 0000 0000  0x00 |
| 0xcb | ALU-A←A  ALU-B← MBR,  A←ALU-A - ALU-B | ALU-B-LOAD-MBR,  ALU-SUB,  LOAD-A | 0100 0000  0x40  LOAD-A | 0001 0100  0x14  ALU-SUB  ALU-GET-RES | 0000 0000  0x00 | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0xcc | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## shl a, xx (0xce, xx)

Pomera u levo A zadati broj puta (xx – ne veće od 7). Broj se nalazi u memoriji odmah posle OPCODE-a (0x0a). PC se poveća za jedan da ukaže na sledeću instrukciju.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0xce | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0xcf | ALU-A ← A  ALU-B ← MBR  A <- ALU-A + ALU-B | ALU-ADD  ALU-GET-RES,  LOAD-A,  MARK-FLAG,  ALU-B-LOAD-MBR | 0100 0000  0x40  LOAD-A | 0000 1100  0x0c  ALU-ADD, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0xd0 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## shr a, xx (0xd2, xx)

Pomera u desno A zadati broj puta (xx – ne veće od 7). Broj se nalazi u memoriji odmah posle OPCODE-a (0x0a). PC se poveća za jedan da ukaže na sledeću instrukciju.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0xd2 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0xd3 | ALU-A ← A  ALU-B ← MBR  A <- ALU-A - ALU-B | ALU-SUB  ALU-GET-RES,  LOAD-A,  MARK-FLAG,  ALU-B-LOAD-MBR | 0100 0000  0x40  LOAD-A | 0001 0100  0x14  ALU-SUB, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0xd4 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

# PUSH i POP sa registrima A i B

## push xx (0x70, xx)

Stavlja na stek broj xx. SP se povećava za jedan. Broj xx se nalazi u memoriji odmah iza OPCODE-a (0xca).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x70 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC, LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x71 | address\_bus ← SP,  memory ← MBR,  SP ← SP + 1 | SP-OE,  LOAD-MAR,  MBR-OE,  MBR-DATA-BUS  MEM-WR,  SP-INC,  LOAD-SP | 0000 0010  0x02  MEM-WR | 0000 0000  0x00 | 1000 0100  0x84  LOAD-MAR,  MBR-OE | 1100 1000  0xc8  LOAD-SP,  SP-OE,  SP-INC | 0010 0000  0x20  MBR-DATA-BUS |
| 0x72 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## push a (0x74)

Stavlja na stek A. SP se povećava za jedan.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x74 | address\_bus ← SP,  memory ← A,  SP ← SP + 1 | SP-OE, LOAD-MAR,  OE-A,  MEM-WR,  SP-INC,  LOAD-SP | 0000 0110  0x06  OE-A, MEM-WR | 0000 0000  0x00 | 1000 0000  0x80  LOAD-MAR | 1100 1000  0xc8  LOAD-SP,  SP-OE,  SP-INC |
| 0x75 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## pop a (0x76)

Kupi sa steka broj i stavlja u A. SP se smanjuje za jedan.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x76 | SP ← SP-1, address\_bus ← SP,  A←memory | SP-DEC,  LOAD-SP,  LOAD-MAR,  MEM-RD,  LOAD-A,  SP-1-TO-MAR | 1100 0001  0xc1  SP-1-TO-MAR  LOAD-A, MEM-RD | 0000 0000  0x00 | 1100 0000  0xc0  LOAD-MAR  SP-DEC | 8000 0000  0x80  LOAD-SP |
| 0x77 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## push b (0x78)

Stavlja na stek B. SP se povećava za jedan.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x78 | address\_bus ← SP,  memory ← B,  SP ← SP + 1 | SP-OE, LOAD-MAR,  OE-B,  MEM-WR,  SP-INC | 0000 1010  0x0a  OE-B, MEM-WR | 0000 0000  0x00 | 1000 1000  0x80  LOAD-MAR | 1100 1000  0xc8  LOAD-SP,  SP-OE,  SP-INC |
| 0x79 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## pop b (0x7a)

Kupi sa steka broj i stavlja u B. SP smanjuje za jedan.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x7a | SP ← SP-1, address\_bus ← SP,  B←memory | SP-DEC,  LOAD-SP,  LOAD-MAR,  MEM-RD,  LOAD-B,  SP-1-TO-MAR | 1010 0001  0xa1  SP-1-TO-MAR  LOAD-B, MEM-RD | 0000 0000  0x00 | 1100 0000  0xc0  LOAD-MAR  SP-DEC | 1000 0000  0x80  LOAD-SP |
| 0x7b | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

# Skokovi i pozivi potprograma

## jmp xx (0x7c, xx)

Bezuslovno skoči na adresu xx.

Posle OPCODE-a, PC ukazuje na operand, odn. na adresu skoka. Kada uzme adresu skoka, nju stavlja u PC.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x7c | address\_bus ← PC,  PC ← memory (iz DATA-BUS-a) | PC-OE, LOAD-MAR,  MAR-OE, MEM-RD, LOAD-PC | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1100 1010  0xca  LOAD-MAR,  MAR-OE,  LOAD-PC, PC-OE | 0000 0000  0x00 |
| 0x7d | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## jz xx (0x7e, xx)

Skoči na adresu xx, ako je ZERO flag na jedinici; inače pređi na sledeću operaciju.

PC posle prvog bajta (OPCODE) ukazuje na operand, odn. na adresu skoka. Ako ZERO nije jedan, PC prelazi na sledeću operaciju, posle adrese. Ideja je da se proveri stanje ZERO flaga i ako nije jedinica, onda se izazove momentalni fetch sledeće instrukcije, koja sledi iza ove (nije zadovoljen uslov, pa se prelazi na sledeću instrukciju).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x7e | address\_bus←PC  MBR ←memory,  PC ← PC + 1 | PC-OE, LOAD-MAR, MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x7f | proveri ZERO flag; ako nije 1, idi odmah na fetch, a PC je namešten na instrukciju posle | CHECK-NOT-ZERO | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0100  0x04  CHECK-NOT-ZERO |
| 0x80 | PC ← MBR  ovo je skok na učitanu adresu, ako je ZERO bio 1 | MBR-OE, LOAD-PC | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0110  0x06  MBR-OE, LOAD-PC | 0000 0000  0x00 |  |
| 0x81 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## jnz xx (0x82, xx)

Skoči na adresu xx, ako je ZERO flag na nuli; inače pređi na sledeću operaciju.

PC posle prvog bajta (OPCODE) ukazuje na operand, odn. na adresu skoka. Ako ZERO nije jedan, PC prelazi na sledeću operaciju, posle adrese. Ideja je da se proveri stanje ZERO flaga i ako jeste jedinica, onda se izazove momentalni fetch sledeće instrukcije, koja sledi iza ove (nije zadovoljen uslov, pa se prelazi na sledeću instrukciju).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x82 | address\_bus ←PC  MBR ←memory,  PC ← PC + 1 | PC-OE, LOAD-MAR, MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |
| 0x83 | proveri ZERO flag; ako je 1, idi odmah na fetch, a PC je namešten na instrukciju posle | CHECK-ZERO | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0010  0x02  CHECK-ZERO |
| 0x84 | PC ← MBR  ovo je skok na učitanu adresu, ako je ZERO bio 0 | MBR-OE, LOAD-PC | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0110  0x06  MBR-OE, LOAD-PC | 0000 0000  0x00 |
| 0x85 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## jp xx (0x9a, xx)

Skoči na adresu xx, ako je POSITIVE flag na jedinici; inače pređi na sledeću operaciju.

PC posle prvog bajta (OPCODE) ukazuje na operand, odn. na adresu skoka. Ako POSITIVE nije jedan, PC prelazi na sledeću operaciju, posle adrese. Ideja je da se proveri stanje POSITIVE flaga i ako nije jedinica, onda se izazove momentalni fetch sledeće instrukcije, koja sledi iza ove (nije zadovoljen uslov, pa se prelazi na sledeću instrukciju).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Šestih 8 bitova |
| 0x9a | address\_bus←PC  MBR ←memory,  PC ← PC + 1 | PC-OE, LOAD-MAR, MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x9b | proveri POSITIVE flag; ako nije 1, idi odmah na fetch, a PC je namešten na instrukciju posle | CHECK-NOT-POSITIVE | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0100 0000  0x40  CHECK-NOT-POSITIVE |
| 0x9c | PC ← MBR  ovo je skok na učitanu adresu, ako je ZERO bio 1 | MBR-OE, LOAD-PC | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0110  0x06  MBR-OE, LOAD-PC | 0000 0000  0x00 |  |
| 0x9d | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## jnp xx (0x9e, xx)

Skoči na adresu xx, ako je POSITIVE flag na nuli; inače pređi na sledeću operaciju.

PC posle prvog bajta (OPCODE) ukazuje na operand, odn. na adresu skoka. Ako POSITIVE nije jedan, PC prelazi na sledeću operaciju, posle adrese. Ideja je da se proveri stanje POSITIVE flaga i ako jeste jedinica, onda se izazove momentalni fetch sledeće instrukcije, koja sledi iza ove (nije zadovoljen uslov, pa se prelazi na sledeću instrukciju).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Šestih 8 bitova |
| 0x9e | address\_bus ←PC  MBR ←memory,  PC ← PC + 1 | PC-OE, LOAD-MAR, MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x9f | proveri POSITIVE flag; ako je 1, idi odmah na fetch, a PC je namešten na instrukciju posle | CHECK-POSITIVE | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 1000 0000  0x80  CHECK-POSITIVE |
| 0xa0 | PC ← MBR  ovo je skok na učitanu adresu, ako je ZERO bio 0 | MBR-OE, LOAD-PC | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0110  0x06  MBR-OE, LOAD-PC | 0000 0000  0x00 |  |
| 0xa1 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## jo xx (0xa2, xx)

Skoči na adresu xx, ako je OVERFLOW flag na jedinici; inače pređi na sledeću operaciju.

PC posle prvog bajta (OPCODE) ukazuje na operand, odn. na adresu skoka. Ako OVERFLOW nije jedan, PC prelazi na sledeću operaciju, posle adrese. Ideja je da se proveri stanje OVERFLOW flaga i ako nije jedinica, onda se izazove momentalni fetch sledeće instrukcije, koja sledi iza ove (nije zadovoljen uslov, pa se prelazi na sledeću instrukciju).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Šestih 8 bitova |
| 0xa2 | address\_bus←PC  MBR ←memory,  PC ← PC + 1 | PC-OE, LOAD-MAR, MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0xa3 | proveri OVERFLOW flag; ako nije 1, idi odmah na fetch, a PC je namešten na instrukciju posle | CHECK-NOT-OVERFLOW | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0001 0000  0x10  CHECK-NOT-OVERFLOW |
| 0xa4 | PC ← MBR  ovo je skok na učitanu adresu, ako je ZERO bio 1 | MBR-OE, LOAD-PC | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0110  0x06  MBR-OE, LOAD-PC | 0000 0000  0x00 |  |
| 0xa5 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## jno xx (0xa6, xx)

Skoči na adresu xx, ako je POSITIVE flag na nuli; inače pređi na sledeću operaciju.

PC posle prvog bajta (OPCODE) ukazuje na operand, odn. na adresu skoka. Ako POSITIVE nije jedan, PC prelazi na sledeću operaciju, posle adrese. Ideja je da se proveri stanje POSITIVE flaga i ako jeste jedinica, onda se izazove momentalni fetch sledeće instrukcije, koja sledi iza ove (nije zadovoljen uslov, pa se prelazi na sledeću instrukciju).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Šestih 8 bitova |
| 0xa6 | address\_bus ←PC  MBR ←memory,  PC ← PC + 1 | PC-OE, LOAD-MAR, MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0xa7 | proveri OVERFLOW flag; ako je 1, idi odmah na fetch, a PC je namešten na instrukciju posle | CHECK-OVERFLOW | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0010 0000  0x20  CHECK-OVERFLOW |
| 0xa8 | PC ← MBR  ovo je skok na učitanu adresu, ako je ZERO bio 0 | MBR-OE, LOAD-PC | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0110  0x06  MBR-OE, LOAD-PC | 0000 0000  0x00 |  |
| 0xa9 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## call xx (0x86, xx)

Poziva potprogram na adresi xx. Povratnu adresu pre toga stavlja na stek. Povratna adresa je posle adrese skoka. Adresa skoka xx se nalazi odmah posle OPCODE-a.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x86 | MAR←PC  MBR ←memory,  PC←PC+1 | PC-OE, LOAD-MAR, MEM-RD, LOAD-MBR, PC-INC | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x87 | address\_bus ← SP,  memory ← PC,  SP ← SP + 1 | SP-OE, LOAD-MAR,  PC-OE, MEM-WR,  SP-INC | 0000 0010  0x02  MEM-WR | 0000 0000  0x00 | 1000 1000  0x88  LOAD-MAR,  PC-OE, | 1100 1000  0xc8  LOAD-SP,  SP-OE,  SP-INC | 0000 0001  0x01  PC-TO-DATA  da izlaz iz PC ne ode na adresni, već data bus |
| 0x88 | PC ← MBR  skoči na potprogram | MBR-OE, LOAD-PC | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0110  0x06  MBR-OE, LOAD-PC | 0000 0000  0x00 |  |
| 0x89 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## ret (0x8a)

Kupi sa steka povratnu adresu i skače na nju.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x8a | SP ← SP-1, address\_bus ← SP,  PC←memory | SP-DEC,  LOAD-SP,  LOAD-MAR,  MEM-RD,  LOAD-PC | 1000 0001  0x81  SP-1-TO-MAR  MEM-RD | 0000 0000  0x00 | 1100 0010  0xc2  LOAD-MAR  SP-DEC,  LOAD-PC | 1000 0000  0x80  LOAD-SP |
| 0x8b | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## callz xx (0xd6, xx)

Poziva potprogram na adresi xx ako je ZERO flag jedinica. Povratnu adresu pre toga stavlja na stek. Povratna adresa je posle adrese skoka. Adresa skoka xx se nalazi odmah posle OPCODE-a.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0xd6 | MAR←PC  MBR ←memory,  PC←PC+1 | PC-OE, LOAD-MAR, MEM-RD, LOAD-MBR, PC-INC | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0xd7 | proveri ZERO flag; ako nije 1, idi odmah na fetch, a PC je namešten na instrukciju posle | CHECK-NOT-ZERO | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0100  0x04  CHECK-NOT-ZERO |
| 0xd8 | address\_bus ← SP,  memory ← PC,  SP ← SP + 1 | SP-OE, LOAD-MAR,  PC-OE, MEM-WR,  SP-INC | 0000 0010  0x02  MEM-WR | 0000 0000  0x00 | 1000 1000  0x88  LOAD-MAR,  PC-OE, | 1100 1000  0xc8  LOAD-SP,  SP-OE,  SP-INC | 0000 0001  0x01  PC-TO-DATA  da izlaz iz PC ne ode na adresni, već data bus |
| 0xd9 | PC ← MBR  skoči na potprogram | MBR-OE, LOAD-PC | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0110  0x06  MBR-OE, LOAD-PC | 0000 0000  0x00 |  |
| 0xda | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## callnz xx (0xdc, xx)

Poziva potprogram na adresi xx ako ZERO flag nije jedinica. Povratnu adresu pre toga stavlja na stek. Povratna adresa je posle adrese skoka. Adresa skoka xx se nalazi odmah posle OPCODE-a.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0xdc | MAR←PC  MBR ←memory,  PC←PC+1 | PC-OE, LOAD-MAR, MEM-RD, LOAD-MBR, PC-INC | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0xdd | proveri ZERO flag; ako je 1, idi odmah na fetch, a PC je namešten na instrukciju posle | CHECK-ZERO | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0010  0x02  CHECK-ZERO | proveri ZERO flag; ako je 1, idi odmah na fetch, a PC je namešten na instrukciju posle |
| 0xde | address\_bus ← SP,  memory ← PC,  SP ← SP + 1 | SP-OE, LOAD-MAR,  PC-OE, MEM-WR,  SP-INC | 0000 0010  0x02  MEM-WR | 0000 0000  0x00 | 1000 1000  0x88  LOAD-MAR,  PC-OE, | 1100 1000  0xc8  LOAD-SP,  SP-OE,  SP-INC | 0000 0001  0x01  PC-TO-DATA  da izlaz iz PC ne ode na adresni, već data bus |
| 0xdf | PC ← MBR  skoči na potprogram | MBR-OE, LOAD-PC | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0110  0x06  MBR-OE, LOAD-PC | 0000 0000  0x00 |  |
| 0xe0 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## callp xx (0xe2, xx)

Poziva potprogram na adresi xx ako je POSITIVE flag jedinica. Povratnu adresu pre toga stavlja na stek. Povratna adresa je posle adrese skoka. Adresa skoka xx se nalazi odmah posle OPCODE-a.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Šestih 8 bitova |
| 0xe2 | MAR←PC  MBR ←memory,  PC←PC+1 | PC-OE, LOAD-MAR, MEM-RD, LOAD-MBR, PC-INC | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0xe3 | proveri POSITIVE flag; ako nije 1, idi odmah na fetch, a PC je namešten na instrukciju posle | CHECK-NOT-POSITIVE | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0100 0000  0x40  CHECK-NOT-POSITIVE |
| 0xe4 | address\_bus ← SP,  memory ← PC,  SP ← SP + 1 | SP-OE, LOAD-MAR,  PC-OE, MEM-WR,  SP-INC | 0000 0010  0x02  MEM-WR | 0000 0000  0x00 | 1000 1000  0x88  LOAD-MAR,  PC-OE, | 1100 1000  0xc8  LOAD-SP,  SP-OE,  SP-INC | 0000 0001  0x01  PC-TO-DATA  da izlaz iz PC ne ode na adresni, već data bus |
| 0xe5 | PC ← MBR  skoči na potprogram | MBR-OE, LOAD-PC | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0110  0x06  MBR-OE, LOAD-PC | 0000 0000  0x00 |  |
| 0xe6 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## callnp xx (0xe8, xx)

Poziva potprogram na adresi xx ako POSITIVE flag nije jedinica. Povratnu adresu pre toga stavlja na stek. Povratna adresa je posle adrese skoka. Adresa skoka xx se nalazi odmah posle OPCODE-a.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Šestih 8 bitova |
| 0xe8 | MAR←PC  MBR ←memory,  PC←PC+1 | PC-OE, LOAD-MAR, MEM-RD, LOAD-MBR, PC-INC | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0xe9 | proveri POSITIVE flag; ako je 1, idi odmah na fetch, a PC je namešten na instrukciju posle | CHECK-POSITIVE | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 1000 0000  0x80  CHECK-POSITIVE |
| 0xea | address\_bus ← SP,  memory ← PC,  SP ← SP + 1 | SP-OE, LOAD-MAR,  PC-OE, MEM-WR,  SP-INC | 0000 0010  0x02  MEM-WR | 0000 0000  0x00 | 1000 1000  0x88  LOAD-MAR,  PC-OE, | 1100 1000  0xc8  LOAD-SP,  SP-OE,  SP-INC | 0000 0001  0x01  PC-TO-DATA  da izlaz iz PC ne ode na adresni, već data bus |
| 0xeb | PC ← MBR  skoči na potprogram | MBR-OE, LOAD-PC | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0110  0x06  MBR-OE, LOAD-PC | 0000 0000  0x00 |  |
| 0xec | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## callo xx (0xee, xx)

Poziva potprogram na adresi xx ako je OVERFLOW flag jedinica. Povratnu adresu pre toga stavlja na stek. Povratna adresa je posle adrese skoka. Adresa skoka xx se nalazi odmah posle OPCODE-a.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Šestih 8 bitova |
| 0xee | MAR←PC  MBR ←memory,  PC←PC+1 | PC-OE, LOAD-MAR, MEM-RD, LOAD-MBR, PC-INC | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0xef | proveri OVERFLOW flag; ako nije 1, idi odmah na fetch, a PC je namešten na instrukciju posle | CHECK-NOT-OVERFLOW | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0001 0000  0x10  CHECK-NOT-OVERFLOW |
| 0xf0 | address\_bus ← SP,  memory ← PC,  SP ← SP + 1 | SP-OE, LOAD-MAR,  PC-OE, MEM-WR,  SP-INC | 0000 0010  0x02  MEM-WR | 0000 0000  0x00 | 1000 1000  0x88  LOAD-MAR,  PC-OE, | 1100 1000  0xc8  LOAD-SP,  SP-OE,  SP-INC | 0000 0001  0x01  PC-TO-DATA  da izlaz iz PC ne ode na adresni, već data bus |
| 0xf1 | PC ← MBR  skoči na potprogram | MBR-OE, LOAD-PC | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0110  0x06  MBR-OE, LOAD-PC | 0000 0000  0x00 |  |
| 0xf2 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## callno xx (0xf4, xx)

Poziva potprogram na adresi xx ako OVERFLOW flag nije jedinica. Povratnu adresu pre toga stavlja na stek. Povratna adresa je posle adrese skoka. Adresa skoka xx se nalazi odmah posle OPCODE-a.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Šestih 8 bitova |
| 0xf4 | MAR←PC  MBR ←memory,  PC←PC+1 | PC-OE, LOAD-MAR, MEM-RD, LOAD-MBR, PC-INC | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0xf5 | proveri OVERFLOW flag; ako je 1, idi odmah na fetch, a PC je namešten na instrukciju posle | CHECK-OVERFLOW | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0010 0000  0x20  CHECK-OVERFLOW |
| 0xf6 | address\_bus ← SP,  memory ← PC,  SP ← SP + 1 | SP-OE, LOAD-MAR,  PC-OE, MEM-WR,  SP-INC | 0000 0010  0x02  MEM-WR | 0000 0000  0x00 | 1000 1000  0x88  LOAD-MAR,  PC-OE, | 1100 1000  0xc8  LOAD-SP,  SP-OE,  SP-INC | 0000 0001  0x01  PC-TO-DATA  da izlaz iz PC ne ode na adresni, već data bus |
| 0xf7 | PC ← MBR  skoči na potprogram | MBR-OE, LOAD-PC | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0110  0x06  MBR-OE, LOAD-PC | 0000 0000  0x00 |  |
| 0xf8 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

# STORE operacije

## st a, [xx] (0x8c, xx)

Stavlja vrednost iz A u lokaciju xx. Adresa lokacije se nalazi u memoriji odmah posle OPCODE-a (0x90).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x8c | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus  PC←PC+1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR,  LOAD-PC,  PC-INC | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |
| 0x8d | address\_bus ← MBR  memory←A | MBR-OE, LOAD-MAR,  OE-A, MEM-WR | 0000 0110  0x06  OE-A, MEM-WR | 0000 0000  0x00 | 1000 0100  0x84  LOAD-MAR,  MBR-OE | 0000 0000  0x00 |
| 0x8e | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## st b, [xx] (0x90, xx)

Stavlja vrednost iz B u lokaciju xx. Adresa lokacije se nalazi u memoriji odmah posle OPCODE-a (0x90).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x90 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus  PC←PC+1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR,  LOAD-PC,  PC-INC | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |
| 0x91 | address\_bus ← MBR  memory←B | MBR-OE, LOAD-MAR,  OE-B, MEM-WR | 0000 1010  0x0a  OE-B, MEM-WR | 0000 0000  0x00 | 1000 0100  0x84  LOAD-MAR,  MBR-OE | 0000 0000  0x00 |
| 0x92 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |

## st a, [b] (0x94)

Stavlja vrednost iz A u lokaciju u memoriji na koju ukazuje B.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x94 | address\_bus ← B, memory←A | OE-B,  B-ADDR-BUS  OE-A,  MEM-WR | 0000 1110  0x0e  OE-B,  OE-A,  MEM-WR | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0010  0x02  B-ADDR-BUS |
| 0x95 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## st a, [b ± xx] (0x96, xx)

Stavlja registar a u sadržaj memorijske lokacije na koju ukazuje (B+xx). PC se poveća za jedan da ukaže na sledeću instrukciju.

Ako je operator oduzimanja (b - xx), onda se u memoriji posle OPCODE-a nalazi potpuni komplement broja xx.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x96 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR,  LOAD-PC,  PC-INC | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x97 | ALU-A ← B  ALU-B ← MBR  MBR←ALU-A + ALU-B | ALU-A-LOAD-B,  ALU-B-LOAD-MBR  ALU-ADD, ALU-GET-RES,  LOAD-MBR | 0000 0000  0x00 | 0000 1110  0x0e  ALU-ADD,  ALU-GET-RES,  ALU-A-LOAD-B | 0000 0001  0x01  LOAD-MBR | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |
| 0x98 | address\_bus← MBR, data\_bus← A,  PC←PC+1 | MBR-OE, MBR-DATA-BUS,  LOAD-MAR,  MEM-WR,  OE-A | 0000 0110  0x06  OE-A,  MEM-WR | 0000 0000  0x00 | 1000 0100  0x84  LOAD-MAR,  MBR-OE | 0000 0000  0x00 | 0000 0000  0x00 |
| 0x99 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

# Množenje i deljenje

## mul a, b (0x01)

Množi a i b i stavlja donjih 8 bita u a. Gornjih 8 bita je u HIGH-BYTE registru u ALU. PC se poveća za jedan da ukaže na sledeću instrukciju.

OPCODE je neparan, pa se zapravo od njega računa adresa u ROM-u tako što se uzme parna verzija (0x00) i doda se na 0x100, pa je onda adresa u ROM-u za ovu instrukciju zapravo 0x100.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Šestih 8 bitova | Osmih 8 bitova |
| 0x100 | ALU-A ← A  ALU-B ← B  RES <-  ALU-A \* ALU-B | ALU-MUL | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ALU-MUL |  |
| 0x101 | čekamo da se završi množenje | WAIT | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 |  | 1000 0000  0x80  WAIT |
| 0x102 | A<-RES | LOAD-A,  ALU-GET-RES,  MARK-FLAG | 0100 0000  0x40  LOAD-A | 0000 0100  0x04  ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG |  |  |
| 0x103 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |  |

## div a, b (0x15)

Deli a sa b i stavlja rezultat u a. Ostatak je u HIGH-BYTE registru u ALU. PC se poveća za jedan da ukaže na sledeću instrukciju.

OPCODE je neparan, pa se zapravo od njega računa adresa u ROM-u tako što se uzme parna verzija (0x14) i doda se na 0x100, pa je onda adresa u ROM-u za ovu instrukciju zapravo 0x114.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Šestih 8 bitova | Osmih 8 bitova |
| 0x114 | ALU-A ← A  ALU-B ← B  RES <-  ALU-A / ALU-B | ALU-DIV | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0010  0x02  ALU-DIV |  |
| 0x115 | čekamo da se završi deljenje | WAIT | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 |  | 1000 0000  0x80  WAIT |
| 0x116 | A <- RES | LOAD-A,  ALU-GET-RES,  MARK-FLAG | 0100 0000  0x40  LOAD-A | 0000 0100  0x04  ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG |  |  |
| 0x117 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |  |

## mul a, [b ± xx] (0x41)

Množi a i b i stavlja donjih 8 bita u a. Gornjih 8 bita je u HIGH-BYTE registru u ALU. PC se poveća za jedan da ukaže na sledeću instrukciju.

OPCODE je neparan, pa se zapravo od njega računa adresa u ROM-u tako što se uzme parna verzija (0x00) i doda se na 0x100, pa je onda adresa u ROM-u za ovu instrukciju zapravo 0x100.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova | Šestih 8 bitova | Sedmih 8 bitova |
| 0x140 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |  |  |
| 0x141 | ALU-A ← B  ALU-B ← MBR  MBR←ALU-A + ALU-B | ALU-A-LOAD-B,  ALU-B-LOAD-MBR  ALU-ADD, ALU-GET-RES,  LOAD-MBR | 0000 0000  0x00 | 0000 1110  0x0e  ALU-ADD,  ALU-GET-RES,  ALU-A-LOAD-B | 0000 0001  0x01  LOAD-MBR | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |  |  |
| 0x142 | address\_bus← MBR,  MBR← data\_bus, | MBR-OE,  LOAD-MAR,  MEM-RD,  LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 0101  0x85  LOAD-MAR,  MBR-OE,  LOAD-MBR | 0000 0000  0x00 |  |  |  |
| 0x143 | ALU-A ← A  ALU-B ← MBR  RES <-  ALU-A \* ALU-B | ALU-B-LOAD-MBR  ALU-MUL | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR | 0000 0001  0x01  ALU-MUL |  |
| 0x144 | ponovimo signal ALU-B-LOAD-MBR | ALU-B-LOAD-MBR | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |  |  |
| 0x145 | idi na 0x101 (završetak množenja) | MICRO-CODE-COUNTER,  MUL-CONT | 0001 0000  0x10  MICRO-CODE-COUNTER | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 |  |  | 0000 0100  0x04  MUL-CONT |

## div a, [b ± xx] (0x47)

Deli a i b i stavlja rezultat u a. Ostatak je u HIGH-BYTE registru u ALU. PC se poveća za jedan da ukaže na sledeću instrukciju.

OPCODE je neparan, pa se zapravo od njega računa adresa u ROM-u tako što se uzme parna verzija (0x14) i doda se na 0x100, pa je onda adresa u ROM-u za ovu instrukciju zapravo 0x114.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova | Šestih 8 bitova | Sedmih 8 bitova |
| 0x146 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |  |  |
| 0x147 | ALU-A ← B  ALU-B ← MBR  MBR←ALU-A + ALU-B | ALU-A-LOAD-B,  ALU-B-LOAD-MBR  ALU-ADD, ALU-GET-RES,  LOAD-MBR | 0000 0000  0x00 | 0000 1110  0x0e  ALU-ADD,  ALU-GET-RES,  ALU-A-LOAD-B | 0000 0001  0x01  LOAD-MBR | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |  |  |
| 0x148 | address\_bus← MBR,  MBR← data\_bus, | MBR-OE,  LOAD-MAR,  MEM-RD,  LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 0101  0x85  LOAD-MAR,  MBR-OE,  LOAD-MBR | 0000 0000  0x00 |  |  |  |
| 0x149 | ALU-A ← A  ALU-B ← MBR  RES <-  ALU-A / ALU-B | ALU-B-LOAD-MBR  ALU-DIV | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR | 0000 0010  0x02  ALU-DIV |  |
| 0x14a | čekamo da se završi deljenje | ALU-B-LOAD-MBR | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |  |  |
| 0x14b | idi na 0x115 (završetak deljenja) | MICRO-CODE-COUNTER,  DIV-CONT | 0001 0000  0x10  MICRO-CODE-COUNTER | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 |  |  | 0000 1000  0x08  DIV-CONT |

## mul a, xx (0x4d)

Množi a i broj xx i stavlja donjih 8 bita u a. Gornjih 8 bita je u HIGH-BYTE registru u ALU. PC se poveća za jedan da ukaže na sledeću instrukciju.

OPCODE je neparan, pa se zapravo od njega računa adresa u ROM-u tako što se uzme parna verzija (0x0c) i doda se na 0x100, pa je onda adresa u ROM-u za ovu instrukciju zapravo 0x16c.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova | Šestih 8 bitova | Sedmih 8 bitova |
| 0x14c | address\_bus←PC  MBR ←memory,  PC ← PC + 1 | PC-OE, LOAD-MAR, MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |  |  |
| 0x14d | ALU-A ← A  ALU-B ← MBR  RES <-  ALU-A \* ALU-B | ALU-MUL,  ALU-B-LOAD-MBR | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR | 0000 0001  0x01  ALU-MUL |  |
| 0x14e | čekamo da se završi množenje | ALU-B-LOAD-MBR | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |  |  |
| 0x14f | idi na 0x101 (završetak množenja) | MICRO-CODE-COUNTER,  MUL-CONT | 0001 0000  0x10  MICRO-CODE-COUNTER | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 |  |  | 0000 0100  0x04  MUL-CONT |

## div a, xx (0x51)

Deli a brojem xx i stavlja rezultat u a. Ostatak je u HIGH-BYTE registru u ALU. PC se poveća za jedan da ukaže na sledeću instrukciju.

OPCODE je neparan, pa se zapravo od njega računa adresa u ROM-u tako što se uzme parna verzija (0x14) i doda se na 0x100, pa je onda adresa u ROM-u za ovu instrukciju zapravo 0x114.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova | Šestih 8 bitova | Sedmih 8 bitova |
| 0x150 | address\_bus←PC  MBR ←memory,  PC ← PC + 1 | PC-OE, LOAD-MAR, MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |  |  |
| 0x151 | ALU-A ← A  ALU-B ← MBR  RES <-  ALU-A / ALU-B | ALU-DIV,  ALU-B-LOAD-MBR | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR | 0000 0010  0x02  ALU-DIV |  |
| 0x152 | čekamo da se završi deljenje | ALU-B-LOAD-MBR | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |  |  |
| 0x153 | idi na 0x115 (završetak deljenja) | MICRO-CODE-COUNTER,  DIV-CONT | 0001 0000  0x10  MICRO-CODE-COUNTER | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 |  |  | 0000 1000  0x08  DIV-CONT |

## ld a, h (0xfa)

Stavlja vrednost HIGH bajta iz ALU u A. PC se poveća za jedan da ukaže na sledeću instrukciju.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Sedmih 8 bitova |
| 0xfa | A← HIGH-BYTE, | LOAD-A,  GET-HIGH | 0100 0000  0x40  LOAD-A | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 1000 0000  0x80  GET-HIGH |
| 0xfb | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

# Registar C

## ld c, xx (0x2d, xx)

Stavlja broj u C. Broj se nalazi u memoriji odmah posle OPCODE-a (0x29). PC se poveća za jedan da ukaže na sledeću instrukciju.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Sedmih 8 bitova |
| 0x12c | address\_bus ← PC, data\_bus←memory, C ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-C | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1010  0x8a  LOAD-MAR, PC-OE, LOAD-PC | 0010 0000  0x20  PC-INC | 0100 0000  0x40  LOAD-C |
| 0x12d | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## ld a, [c] (0x59)

Stavlja sadržaj memorijske lokacije na koju ukazuje C u A. PC se poveća za jedan da ukaže na sledeću instrukciju.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Sedmih 8 bitova |
| 0x158 | address\_bus← C, A← data\_bus,  PC←PC+1 | OE-C, C-ADDR-BUS, MEM-RD, LOAD-A | 0100 0001  0x41  LOAD-A,  MEM-RD | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0011 0000  0x30  OE-C  C-ADDR-BUS |
| 0x159 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## ld a, [c ± xx] (0x5b, xx)

Stavlja sadržaj memorijske lokacije na koju ukazuje (c+xx) u A. PC se poveća za jedan da ukaže na sledeću instrukciju.

Ako je operator oduzimanja (c - xx), onda se u memoriji posle OPCODE-a nalazi potpuni komplement broja xx.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x15a | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x15b | ALU-A ← C  ALU-B ← MBR  MBR←ALU-A + ALU-B | ALU-A-LOAD-B,  ALU-A-LOAD-MBR  ALU-B-LOAD-MBR  ALU-ADD, ALU-GET-RES,  LOAD-MBR | 0000 0000  0x00 | 0000 1110  0x0e  ALU-ADD,  ALU-GET-RES,  ALU-A-LOAD-B | 0000 0001  0x01  LOAD-MBR | 0000 0000  0x00 | 0001 1000  0x18  ALU-A-LOAD-MBR  ALU-B-LOAD-MBR |
| 0x15c | address\_bus← MBR, A← data\_bus, | MBR-OE,  LOAD-MAR,  MEM-RD,  LOAD-A | 0100 0001  0x41  LOAD-A,  MEM-RD | 0000 0000  0x00 | 1000 0100  0x84  LOAD-MAR,  MBR-OE | 0000 0000  0x00 | 0000 0000  0x00 |
| 0x15d | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## st a, [c] (0x2f)

Stavlja vrednost iz A u lokaciju u memoriji na koju ukazuje C.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova | Sedmih 8 bitova |
| 0x12e | address\_bus ← C, memory←A | OE-C,  C-ADDR-BUS  OE-A,  MEM-WR | 0000 0110  0x06  OE-A,  MEM-WR | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0011 0000  0x30  OE-C,  C-ADDR-BUS |
| 0x12f | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |  |

## st a, [c ± xx] (0x31, xx)

Stavlja registar a u sadržaj memorijske lokacije na koju ukazuje (C+xx). PC se poveća za jedan da ukaže na sledeću instrukciju.

Ako je operator oduzimanja (c - xx), onda se u memoriji posle OPCODE-a nalazi potpuni komplement broja xx.

Napomena: da bi se C doveo na A ulaz od ALU, stavljaju se sledeća dva kontrolna signala na 1: ALU-A-LOAD-B i ALU-A-LOAD-MBR.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova |
| 0x130 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR,  LOAD-PC,  PC-INC | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |
| 0x131 | ALU-A ← C  ALU-B ← MBR  MBR←ALU-A + ALU-B | ALU-A-LOAD-B,  ALU-A-LOAD-MBR,  ALU-B-LOAD-MBR  ALU-ADD, ALU-GET-RES,  LOAD-MBR | 0000 0000  0x00 | 0000 1110  0x0e  ALU-ADD,  ALU-GET-RES,  ALU-A-LOAD-B | 0000 0001  0x01  LOAD-MBR | 0000 0000  0x00 | 0001 1000  0x18  ALU-A-LOAD-MBR,  ALU-B-LOAD-MBR |
| 0x132 | address\_bus← MBR, data\_bus← A,  PC←PC+1 | MBR-OE, MBR-DATA-BUS,  LOAD-MAR,  MEM-WR,  OE-A | 0000 0110  0x06  OE-A,  MEM-WR | 0000 0000  0x00 | 1000 0100  0x84  LOAD-MAR,  MBR-OE | 0000 0000  0x00 | 0000 0000  0x00 |
| 0x133 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |

## dec c (0x35)

Smanji C za 1. Flegovi se postavljaju nakon što se dobije rezultat.

Napomena: da bi se C doveo na A ulaz od ALU, stavljaju se sledeća dva kontrolna signala na 1: ALU-A-LOAD-B i ALU-A-LOAD-MBR.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova | Sedmih 8 bitova |
| 0x134 | C ← C - 1 | ONE-TO-ALU,  ALU-SUB, ALU-GET-RES, LOAD-C, MARK-FLAG  ALU-A-LOAD-B,  ALU-A-LOAD-MBR | 0000 0000  0x00 | 0001 0111  0x15  ALU-SUB, ALU-GET-RES,  ALU-A-LOAD-B  ONE-TO-ALU | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0001 0000  0x10  ALU-A-LOAD-MBR | 0100 0000  0x40  LOAD-C |
| 0x135 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |  |

## inc c (0x37)

Povećaj C za 1. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova | Sedmih 8 bitova |
| 0x136 | C ← C + 1 | ONE-TO-ALU,  ALU-ADD, ALU-GET-RES, LOAD-C, MARK-FLAG,  ALU-A-LOAD-B,  ALU-A-LOAD-MBR | 0000 0000  0x00 | 0000 1111  0x0f  ALU-ADD, ALU-GET-RES,  ALU-A-LOAD-B,  ONE-TO-ALU | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0001 0000  0x10  ALU-A-LOAD-MBR | 0100 0000  0x40  LOAD-C |
| 0x137 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |  |

## ld c, a (0x39)

Stavlja vrednost A u C. PC se poveća za jedan da ukaže na sledeću instrukciju.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Sedmih 8 bitova |
| 0x138 | data\_bus← A, C← data\_bus, | OE-A,  LOAD-C | 0000 0100  0x04  OE-A | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0100 0000  0x40  LOAD-C |
| 0x139 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## ld c, b (0x3b)

Stavlja vrednost B u C. PC se poveća za jedan da ukaže na sledeću instrukciju.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Sedmih 8 bitova |
| 0x13a | data\_bus← A, C← data\_bus, | OE-B,  LOAD-C | 0000 1000  0x08  OE-B | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0100 0000  0x40  LOAD-C |
| 0x13b | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## ld a, c (0x55)

Stavlja vrednost C u A. PC se poveća za jedan da ukaže na sledeću instrukciju.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Sedmih 8 bitova |
| 0x154 | data\_bus← C, A← data\_bus, | OE-C,  LOAD-A | 0100 0000  0x40  LOAD-A | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0010 0000  0x20  OE-C |
| 0x155 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## ld b, c (0x57)

Stavlja vrednost C u B. PC se poveća za jedan da ukaže na sledeću instrukciju.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Sedmih 8 bitova |
| 0x156 | data\_bus← C,  B←data\_bus | OE-C,  LOAD-B | 0010 0000  0x20  LOAD-B | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0010 0000  0x20  OE-C |
| 0x157 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## push c (0x3d)

Stavlja na stek C. SP se povećava za jedan.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Sedmih 8 bitova |
| 0x13c | address\_bus ← SP,  memory ← C,  SP ← SP + 1 | SP-OE, LOAD-MAR,  OE-C, MEM-WR,  SP-INC,  LOAD-SP | 0000 0010  0x02  MEM-WR | 0000 0000  0x00 | 1000 0000  0x80  LOAD-MAR | 1100 1000  0xc8  LOAD-SP,  SP-OE,  SP-INC | 0010 0000  0x20  OE-C |
| 0x13d | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## pop c (0x3f)

Kupi sa steka broj i stavlja u C. SP se smanjuje za jedan.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Sedmih 8 bitova |
| 0x13e | SP ← SP-1, address\_bus ← SP,  C←memory | SP-DEC,  LOAD-SP,  LOAD-MAR,  MEM-RD,  LOAD-C,  SP-1-TO-MAR | 1000 0001  0x81  SP-1-TO-MAR  MEM-RD | 0000 0000  0x00 | 1100 0000  0xc0  LOAD-MAR  SP-DEC | 8000 0000  0x80  LOAD-SP | 0100 0000  0x40  LOAD-C |
| 0x13f | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## mul a, c (0x5f)

Množi a i b i stavlja donjih 8 bita u a. Gornjih 8 bita je u HIGH-BYTE registru u ALU. PC se poveća za jedan da ukaže na sledeću instrukciju.

OPCODE je neparan, pa se zapravo od njega računa adresa u ROM-u tako što se uzme parna verzija (0x00) i doda se na 0x100, pa je onda adresa u ROM-u za ovu instrukciju zapravo 0x100.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova | Šestih 8 bitova | Sedmih 8 bitova |
| 0x15e | ALU-A ← A  ALU-B ← C  RES <-  ALU-A \* ALU-B | ALU-MUL | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 |  | 0000 0001  0x01  ALU-MUL | 0000 0010  0x02  ALU-B-LOAD-C |
| 0x15f | idi na 0x101 (završetak množenja) | MICRO-CODE-COUNTER,  MUL-CONT, ALU-B-LOAD-C | 0001 0000  0x10  MICRO-CODE-COUNTER | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 |  |  | 0000 0110  0x06  MUL-CONT,  ALU-B-LOAD-C |

## div a, c (0x61)

Deli a sa b i stavlja rezultat u a. Ostatak je u HIGH-BYTE registru u ALU. PC se poveća za jedan da ukaže na sledeću instrukciju.

OPCODE je neparan, pa se zapravo od njega računa adresa u ROM-u tako što se uzme parna verzija (0x14) i doda se na 0x100, pa je onda adresa u ROM-u za ovu instrukciju zapravo 0x114.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |  | Šestih 8 bitova | Sedmih 8 bitova |
| 0x160 | ALU-A ← A  ALU-B ← C  RES <-  ALU-A / ALU-B | ALU-DIV | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 |  | 0000 0010  0x02  ALU-DIV | 0000 0010  0x02  ALU-B-LOAD-C |
| 0x161 | idi na 0x115 (završetak deljenja) | MICRO-CODE-COUNTER,  DIV-CONT | 0001 0000  0x10  MICRO-CODE-COUNTER, ALU-B-LOAD-C | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 |  |  | 0000 1010  0x0a  DIV-CONT,  ALU-B-LOAD-C |

# XOR operacija

## xor a, b (0x63)

A ← A xor B

Sabira A i B i rezultat smešta u A. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Sedmih 8 bitova |
| 0x162 | ALU-A ← A  ALU-B ← B  A ← ALU-A xor ALU-B | ALU-XOR  ALU-GET-RES,  LOAD-A,  MARK-FLAG | 0100 0000  0x40  LOAD-A | 0000 0100  0x04  ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 0001  0x01  ALU-XOR |
| 0x163 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## xor a, c (0x65)

A ← A xor C

Sabira A i B i rezultat smešta u A. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Sedmih 8 bitova |
| 0x164 | ALU-A ← A  ALU-B ← C  A ← ALU-A xor ALU-B | ALU-XOR  ALU-GET-RES,  LOAD-A,  MARK-FLAG | 0100 0000  0x40  LOAD-A | 0000 0100  0x04  ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 0011  0x03  ALU-B-LOAD-C,  ALU-XOR |
| 0x165 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## xor a, [b ± xx] (0x67, xx)

Računa XOR registra a i sadržaja memorijske lokacije na koju ukazuje (B+xx). PC se poveća za jedan da ukaže na sledeću instrukciju. Flegovi se postavljaju nakon što se dobije rezultat.

Ako je operator oduzimanja (b - xx), onda se u memoriji posle OPCODE-a nalazi potpuni komplement broja xx.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova | Sedmih 8 bitova |
| 0x166 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |  |
| 0x167 | ALU-A ← B  ALU-B ← MBR  MBR←ALU-A + ALU-B | ALU-A-LOAD-B,  ALU-B-LOAD-MBR  ALU-ADD, ALU-GET-RES,  LOAD-MBR | 0000 0000  0x00 | 0000 1110  0x0e  ALU-ADD,  ALU-GET-RES,  ALU-A-LOAD-B | 0000 0001  0x01  LOAD-MBR | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |  |
| 0x168 | address\_bus← MBR,  MBR← data\_bus, | MBR-OE,  LOAD-MAR,  MEM-RD,  LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 0101  0x85  LOAD-MAR,  MBR-OE,  LOAD-MBR | 0000 0000  0x00 | 0000 0000  0x00 |  |
| 0x169 | ALU-A←A  ALU-B← MBR,  A←ALU-A xor ALU-B | ALU-B-LOAD-MBR,  ALU-XOR,  LOAD-A | 0100 0000  0x40  LOAD-A | 0000 0100  0x04  ALU-GET-RES | 0000 0000  0x00 | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR | 0000 0001  0x01  ALU-XOR |
| 0x16a | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |  |

## xor a, [b] (0x6d)

A ← A xor [B]

Radi XOR registra A i sadržaja memorijske lokacije na koju ukazuje B i rezultat smešta u A. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova | Sedmih 8 bitova |
| 0x16c | address\_bus←B  MBR ←memory | OE-B,  B-ADDR-BUS,  LOAD-MAR,  MEM-RD, LOAD-MBR, | 0000 1001  0x09  OE-B, MEM-RD | 0000 0000  0x00 | 1000 0001  0x81  LOAD-MAR,  LOAD-MBR | 0000 0000  0x00 | 0000 0010  0x02  B-ADDR-BUS |  |
| 0x16d | ALU-A ← A  ALU-B ← MBR  A <- ALU-A xor ALU-B | ALU-XOR,  ALU-GET-RES,  LOAD-A,  MARK-FLAG,  ALU-B-LOAD-MBR | 0100 0000  0x40  LOAD-A | 0000 0100  0x04  ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 1000  0x08  ALU-B-LOAD-MBR | 0000 0001  0x01  ALU-XOR |
| 0x16e | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |  |

## add a, c (0x77)

A ← A + C

Sabira A i C i rezultat smešta u A. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Sedmih 8 bitova |
| 0x176 | ALU-A ← A  ALU-B ← C  A ← ALU-A + ALU-B | ALU-ADD  ALU-GET-RES,  LOAD-A,  MARK-FLAG,  ALU-B-LOAD-C | 0100 0000  0x40  LOAD-A | 0000 1100  0x0c  ALU-ADD, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 0010  0x02  ALU-B-LOAD-C |
| 0x177 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## sub a, c (0x79)

A ← A – C

Oduzima C od A i smešta rezultat nazad u A. Flegovi se postavljaju nakon što se dobije rezultat.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Sedmih 8 bitova |
| 0x178 | ALU-A ← A  ALU-B ← C  A <-  ALU-A - ALU-B | ALU-SUB  ALU-GET-RES,  LOAD-A,  MARK-FLAG,  ALU-LOAD-C | 0100 0000  0x40  LOAD-A | 0001 0100  0x14  ALU-SUB, ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0000 0010  0x02  ALU-B-LOAD-C |
| 0x179 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST |  |

## st a, [b + c ± xx] (0x7b, xx)

Stavlja registar a u sadržaj memorijske lokacije na koju ukazuje (B + C +xx). PC se poveća za jedan da ukaže na sledeću instrukciju.

Ako je operator oduzimanja ((b + c )- xx), onda se u memoriji posle OPCODE-a nalazi potpuni komplement broja xx.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova | Sedmih 8 bitova |
| 0x17a | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR,  LOAD-PC,  PC-INC | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE,  LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |  |
| 0x17b | ALU-A ← B  ALU-B ← MBR  MBR←ALU-A + ALU-B | ALU-A-LOAD-B,  ALU-B-LOAD-MBR  ALU-ADD, ALU-GET-RES,  LOAD-MBR | 0000 0000  0x00 | 0000 1110  0x0e  ALU-ADD,  ALU-GET-RES,  ALU-A-LOAD-B | 0000 0001  0x01  LOAD-MBR | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |  |
| 0x17c | ALU-A ← MBR  ALU-B ← C  MBR←ALU-A + ALU-B | ALU-A-LOAD-MBR,  ALU-B-LOAD-C,  ALU-ADD, ALU-GET-RES,  LOAD-MBR | 0000 0000  0x00 | 0000 1100  0x0c  ALU-ADD,  ALU-GET-RES | 0000 0001  0x01  LOAD-MBR | 0000 0000  0x00 | 0001 0000  0x10  ALU-A-LOAD-MBR | 0000 0010  0x02  ALU-B-LOAD-C |
| 0x17d | address\_bus← MBR, data\_bus← A,  PC←PC+1 | MBR-OE, MBR-DATA-BUS,  LOAD-MAR,  MEM-WR,  OE-A | 0000 0110  0x06  OE-A,  MEM-WR | 0000 0000  0x00 | 1000 0100  0x84  LOAD-MAR,  MBR-OE | 0000 0000  0x00 | 0000 0000  0x00 |  |
| 0x17e | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |  |

## cmp [b ± xx], a (0x71, xx)

Poredi sadržaj memorijske lokacije na koju ukazuje (B+xx) i registar a. PC se poveća za jedan da ukaže na sledeću instrukciju. Flegovi se postavljaju nakon što se dobije rezultat.

Ako je operator oduzimanja (b - xx), onda se u memoriji posle OPCODE-a nalazi potpuni komplement broja xx.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova | Petih 8 bitova | Osmih 8 bitova |
| 0x180 | address\_bus ← PC, data\_bus←memory, MBR ← data\_bus,  PC ← PC + 1 | PC-OE,  LOAD-MAR,  MEM-RD, LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 1011  0x8b  LOAD-MAR, PC-OE, LOAD-PC,  LOAD-MBR | 0010 0000  0x20  PC-INC |  |  |
| 0x181 | ALU-A ← B  ALU-B ← MBR  MBR←ALU-A + ALU-B | ALU-A-LOAD-B,  ALU-B-LOAD-MBR  ALU-ADD, ALU-GET-RES,  LOAD-MBR | 0000 0000  0x00 | 0000 1110  0x0e  ALU-ADD,  ALU-GET-RES,  ALU-A-LOAD-B | 0000 0001  0x01  LOAD-MBR | 0000 0000  0x00 | 0000 1000  0x08  ALU-B-LOAD-MBR |  |
| 0x182 | address\_bus← MBR,  MBR← data\_bus, | MBR-OE,  LOAD-MAR,  MEM-RD,  LOAD-MBR | 0000 0001  0x01  MEM-RD | 0000 0000  0x00 | 1000 0101  0x85  LOAD-MAR,  MBR-OE,  LOAD-MBR | 0000 0000  0x00 | 0000 0000  0x00 |  |
| 0x183 | ALU-A←MBR  ALU-B← A,  REZ←ALU-A - ALU-B | ALU-A-LOAD-MBR,  ALU-B-LOAD-A  ALU-SUB | 0000 0000  0x00 | 0001 0100  0x14  ALU-SUB  ALU-GET-RES | 0000 0000  0x00 | 0000 0100  0x04  MARK-FLAG | 0001 0000  0x10  ALU-A-LOAD-MBR | 0100 0000  0x40  ALU-B-LOAD-A |
| 0x184 | idi na 0x00 (fetch) | ZERO\_CONST | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0001  0x01  ZERO\_CONST | 0000 0000  0x00 |  |

## halt (0xff)

Zaustavlja računar. Translacijom adresa dobijamo da se najniži bit pretvara u najviši, 9. bit u ROM-u, pa se umesto 0xff dobija 0x1fe adresa, a tamo se nalazi signal za HALT.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adresa u memoriji | Mikrokod | Aktivni signali | Prvih 8 bitova | Drugih 8 Bitova | Trećih 8 bitova | Četvrtih 8 bitova |
| 0x1fe | HALT | HALT | 0000 0000  0x00 | 0000 0000  0x00 | 0000 0000  0x00 | 0001 0000  0x10  HALT |

# 

Programi

## Primer1- za operacije od LD, ADD, SUB, PUSH i POP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 0x05 | 0x08 | 0x05 | stavi u a broj 5 |
| 0x02 | ld b, 0x06 | 0x0a | 0x06 | stavi u b broj 6 |
| 0x04 | add a, b | 0x18 |  | a ← a + b (a je sada 11, odn. 0x0b) |
| 0x05 | sub a, b | 0x1a |  | a ← a – b (a je sada 5) |
| 0x06 | ld sp, 0x30 | 0x12 | 0x30 | postavi sp na broj 0x30 |
| 0x08 | push a | 74 |  | stavi a (trenutno je 5) na stek i sp pomeri za jedan na više |
| 0x09 | ld a, 0x01 | 0x08 | 0x01 | stavi u a broj 1 (pregazi peticu) |
| 0x0b | pop a | 0x76 |  | vrati sa steka broj (5) u a i spusti sp za jedan |
| 0x0c | push b | 0x78 |  | stavi b na stek (6) i povećaj sp za jedan |
| 0x0d | ld b, 0x02 | 0x0a | 0x02 | stavi u b broj 2 |
| 0x0f | pop b | 0x7a |  | vrati sa steka broj (6) u b i spusti sp za jedan |
| 0x10 | halt | 0xff |  | zaustavi računar |

## Primer2- za JMP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | jmp 0x04 | 0x7c | 0x04 | skoči na adresu 0x04 |
| 0x02 | nop | 0x00 |  |  |
| 0x03 | nop | 0x00 |  |  |
| 0x04 | ld a, 0x05 | 0x08 | 0x05 | napuni a brojem 5 |
| 0x06 | halt | 0xff |  | zaustavi računar |

## Primer3- za JNZ i DEC A

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 0x02 | 0x08 | 0x02 | napuni a brojem 2 |
| 0x02 | nop | 0x00 |  |  |
| 0x03 | dec a | 0x4a |  | smanji a za jedan (zero će postati 1, ako smanjenjem a dođe do nule) |
| 0x04 | jnz 0x02 | 0x82 | 0x02 | skoči na 0x02 ako zero nije jedinica |
| 0x06 | halt | 0xff |  | zaustavlja računar |

## Primer4- za CALL x i RET

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld sp, 0x10 | 0x12 | 0x10 | postavi sp na broj 0x10 |
| 0x02 | call 0x06 | 0x86 | 0x06 | pozovi potprogram na adresi 6 |
| 0x04 | ld b, a | 0x06 |  | b← a  sada i u b stoji 5 |
| 0x05 | halt | 0xff |  | zaustavlja računar |
| 0x06 | ld a, 0x05 | 0x08 | 0x05 | a ← 5 |
| 0x08 | ret | 0x8a |  | povratak iz potprograma |

## Primer5- koji računa sumu od 1 do 4 i upisuje u adresu 0x80 (128)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld b, 0x04 | 0x0a | 0x04 | napuni b brojem 4 |
| 0x02 | ld a, 0x00 | 0x08 | 0x00 | napuni a brojem 0 |
| 0x04 | add a, b | 0x18 |  | a ← a + b |
| 0x05 | dec b | 0x4e |  | smanji b za jedan (zero će postati 1, ako smanjenjem b dođe do nule) |
| 0x06 | jnz 0x04 | 0x82 | 0x04 | skoči na 0x04 ako zero nije jedinica |
| 0x08 | st a, [0x80] | 0x8c | 0x80 | smesti sadržaj registra a (sumu brojeva) na lokaciju 0x80 (128) |
| 0x0a | halt | 0xff |  | zaustavlja računar |

## Primer6- koji proverava OVERFLOW flag

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld b, 0x6000 | 0x0a | 0x6000 | napuni b brojem 0x6000 |
| 0x02 | ld a, 0x6000 | 0x08 | 0x6000 | napuni a brojem 0x6000 |
| 0x04 | add a, b | 0x18 |  | a ← a + b (ovde bi trebalo da overflow bude 1),  pošto je zbor broj 0xc000, gde je bit znaka jedan, a to znači da je rezultat negativan (iako su oba operanda pozitivni) |
| 0x05 | halt | 0xff |  | zaustavlja računar |

## Primer7- koji upisuje u video memoriju (od 0xfff0 do 0xffff) ASCII karakter (od 0x41 za ‘A’ do 0x51 za ’P’), što izaziva ispis u sva četiri LED matrix elementa (svaki od po 4 karaktera)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 0x41 | 0x08 | 0x41 | napuni a brojem 0x41 (ascii kod slova ‘a’ – 65) |
| 0x02 | ld b, 0 | 0x0a | 0x00 | napuni b brojem 0 |
| 0x04 | st a, [b + 0xf0] | 0x96 | 0xf0 | smesti vrednost a na adresu na koju ukazuje zbir registar b i 0xf0 (prva lokacija video memorije); ispisaće slovo ‘a’ na prvoj ćeliji |
| 0x06 | inc a | 0x4c |  | a ← a + 1 |
| 0x07 | inc b | 0x50 |  | b ← b + 1 |
| 0x08 | cmp b, 16 | 0x29 | 0x10 | Poredi registar b i broj 16 (0x10); ako su jednaki, ZERO će biti 1 |
| 0x0a | jnz 0x04 | 0x82 | 0x04 | Skoči na lokaciju 0x04 ako ZERO nije 1 |
| 0x0c | halt | 0xff |  | zaustavlja računar |

## Primer8- razmenjuje sadržaj registra A i B i upisuje u video memoriju (0xfff0 i 0xfff1) dva ASCII karaktera (0x41 za ‘A’ i 0x42 za ‘B’) iz registara A i B, što izaziva ispis u dva LED matrix elementa

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld b, 0x41 | 0x0a | 0x41 | napuni b brojem 0x41 (ascii kod slova ‘A’ – 65) |
| 0x02 | ld a, b | 0x04 |  | a←b |
| 0x03 | st a, [0xfff0] | 0x8c | 0xf0 | smesti vrednost a na adresu 0xfff0 (prva lokacija video memorije); ispisaće slovo ‘A’ |
| 0x05 | inc a | 0x4c |  | a ← a + 1 (sada je u a broj 0x42, što je ascii kod slova ‘B’ |
| 0x06 | ld b, a | 0x06 |  | b ← a |
| 0x07 | st b, [0xfff1] | 0x90 | 0xf1 | smesti vrednost b na adresu 0xfff1 (druga lokacija video memorije); ispisaće slovo ‘B’ |
| 0x09 | halt | 0xff |  | zaustavlja računar |

## Primer9- indirektno adresiranje (LD A, [b])

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld b, 0x43 | 0x0a | 0x43 | napuni b brojem 0x43 (ascii kod slova ‘C’ – 67) |
| 0x02 | st b, [0x80] | 0x90 | 0x80 | smesti vrednost b na adresu 0x80 |
| 0x04 | ld b, 0x80 | 0x0a | 0x80 | napuni b brojem 0x80 |
| 0x06 | ld a, [b] | 0x0c |  | a ← sadržaj memorijske lokacije na koju ukazuje b;  čita sa lokacije 0x80 bajt (tamo je 0x43) i smešta to u a |
| 0x07 | st a, [0xfff0] | 0x8c | 0xf0 | smesti vrednost a na adresu 0xfff0 (prva lokacija video memorije); ispisaće slovo ‘C’ |
| 0x09 | halt | 0xff |  | zaustavlja računar |

## Primer10- AND, OR I NOT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 0x01 | 0x08 | 0x01 | napuni a brojem 0x01 |
| 0x02 | ld b, 0x02 | 0x0a | 0x02 | napuni b brojem 0x02 |
| 0x04 | or a, b | 0x5c |  | a 🡨 a or b (sada je u a 0x03) |
| 0x05 | ld b, a | 0x06 |  | b 🡨 a (sada je u b 0x03) |
| 0x06 | neg a | 0x6e |  | a 🡨 neg a (sada je u a 0xfffd) |
| 0x07 | and a, b | 0x5a |  | a 🡨 a and b (sada je u a 0x0001) |
| 0x08 | halt | 0xff |  | zaustavlja računar |

## Primer11- JZ, xx komanda

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 0x02 | 0x08 | 0x02 | napuni a brojem 0x02 |
| 0x02 | dec a | 0x4a |  | smanji a za jedan |
| 0x03 | jz 0x08 | 0x7e | 0x08 | ako je zero na jedinici, skoči na lokaciju 0x08, ako nije, nastavi dalje |
| 0x05 | jmp 0x02 | 0x7c | 0x02 | skoči na lokaciju 0x02 |
| 0x07 | nop | 0x00 |  |  |
| 0x08 | halt | 0xff |  | zaustavlja računar |

## Primer12- ST A, [b] – indirektno adresiranje - snima registar A u memoriju na adresu na koju ukazuje B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 0x44 | 0x08 | 0x44 | napuni a brojem 0x44 (ascii kod slova ‘d’) |
| 0x02 | ld b, 0xfff0 | 0x0a | 0xf0 | napuni b brojem 0xfff0 (prvi karakter u video memoriji) |
| 0x04 | st a, [b] | 0x94 |  | snimi u memoriju, na adresi na koju ukazuje b, sadržaj registra a;  sada će se videti u prvom displeju slovo ‘d’ |
| 0x05 | halt | 0xff |  | zaustavlja računar |

## Primer13- CMP A, B – poredi A i B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 0x01 | 0x08 | 0x01 | napuni a brojem 1 |
| 0x02 | ld b, 0x01 | 0x0a | 0x01 | napuni b brojem 1 |
| 0x04 | cmp a, b | 0x40 |  | poredi a i b; pošto su jednaki, zero će biti jedan |
| 0x05 | ld a, 0x02 | 0x08 | 0x02 | napuni a brojem 2 |
| 0x07 | ld b, 0x01 | 0x0a | 0x01 | napuni b brojem 1 |
| 0x09 | cmp a, b | 0x40 |  | poredi a i b; pošto su različiti, zero će biti nula |
| 0x0a | halt | 0xff |  | zaustavlja računar |

## Primer14- CMP A, [b] – poredi A i sadržaj adrese na koju ukazuje B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 0x01 | 0x08 | 0x01 | napuni a brojem 1 |
| 0x02 | ld b, 0x01 | 0x0a | 0x01 | napuni b brojem 1 |
| 0x04 | st b, [0x10] | 0x90 | 0x10 | smesti sadržaj b (0x01) na adresu 0x10 |
| 0x06 | ld b, 0x10 | 0x0a | 0x10 | napuni b brojem 0x10 |
| 0x08 | cmp a, [b] | 0x42 |  | poredi a i sadržaj lokacije na koju ukazuje b; pošto su jednaki, zero će biti jedan |
| 0x09 | halt | 0xff |  | zaustavlja računar |

## Primer15- ADD A, [b] i SUB A, [b] – sabira/oduzima A i sadržaj adrese na koju ukazuje B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 0x01 | 0x08 | 0x01 | napuni a brojem 1 |
| 0x02 | ld b, 0x01 | 0x0a | 0x01 | napuni b brojem 1 |
| 0x04 | st b, [0x10] | 0x90 | 0x10 | smesti sadržaj b (0x01) na adresu 0x10 |
| 0x06 | ld b, 0x10 | 0x0a | 0x10 | napuni b brojem 0x10 |
| 0x08 | add a, [b] | 0x24 |  | saberi a i sadržaj lokacije na koju ukazuje b; rezultat smesti u a (u a će biti 0x02) |
| 0x09 | sub a, [b] | 0x28 |  | oduzmi od a sadržaj lokacije na koju ukazuje b; rezultat smesti u a (u a će biti 0x01) |
| 0x0a | halt | 0xff |  | zaustavlja računar |

## Primer16- DEC [b] i INC [b] – smanjuje/povećava sadržaj adrese na koju ukazuje B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 0x01 | 0x08 | 0x01 | napuni a brojem 1 |
| 0x02 | ld b, 0x10 | 0x0a | 0x10 | napuni b brojem 0x10 |
| 0x04 | st a, [b] | 0x94 |  | smesti a (0x01) u memorijsku lokaciju na koju ukazuje b (0x10) |
| 0x05 | dec [b] | 0x52 |  | smanji sadržaj memorijske lokacije na koju ukazuje b za jedan; sada je b == 0, pa je zero == 1 |
| 0x06 | inc [b] | 0x56 |  | povećaj sadržaj memorijske lokacije na koju ukazuje b za jedan; sada je b == 1, pa je zero == 0 |
| 0x07 | halt | 0xff |  | zaustavlja računar |

## Primer17- ADD A, xx i SUB A, xx – sabira/oduzima A i broj

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 0x01 | 0x08 | 0x01 | napuni a brojem 1 |
| 0x02 | add a, 0x03 | 0x38 | 0x03 | sabere a i 0x03 |
| 0x04 | sub a, 0x04 | 0x3c | 0x04 | oduzme 0x04 od a; u a je sada 0, pa je zero == 1 |
| 0x06 | halt | 0xff |  | zaustavlja računar |

## Primer18- CMP A, xx – poredi A i broj

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 0x01 | 0x08 | 0x01 | napuni a brojem 1 |
| 0x02 | cmp a, 0x01 | 0x46 | 0x01 | poredi a i 0x01; pošto su jednaki, u ZERO će biti 1 |
| 0x03 | halt | 0xff |  | zaustavlja računar |

## Primer19- AND-OR A, [B] – poredi A i broj

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 0x01 | 0x08 | 0x01 | napuni a brojem 1 |
| 0x02 | ld b, 0x02 | 0x0a | 0x02 | napuni a brojem 2 |
| 0x04 | st b, [0x10] | 0x90 | 0x10 | stavi b u lokaciju 0x10 |
| 0x06 | ld b, 0x10 | 0x0a | 0x10 | stavi u b broj 0x10 |
| 0x08 | and a, [b] | 0x5e |  | uradi konjukciju registra a i sadržaja lokacije na koju ukazuje b (u a je sada 0x00) |
| 0x09 | or a, [b] | 0x62 |  | uradi disjunkciju registra a i sadržaja lokacije na koju ukazuje b (u a je sada 0x02) |
| 0x0a | halt | 0xff |  | zaustavlja računar |

## Primer20- AND-OR A, xx

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 0x01 | 0x08 | 0x01 | napuni a brojem 1 |
| 0x02 | and a, 0x02 | 0x66 | 0x02 | konjukcija regista a i broja 0x02; rezultat je 0x00 i ZERO flag je 1 |
| 0x03 | or a, 0x02 | 0x6a | 0x02 | disjunkcija registra a (trenutno 0x00) i broja 0x02; rezultat je 0x02 i ZERO je 0 |
| 0x04 | halt | 0xff |  | zaustavlja računar |

## Primer21- poziv potprograma i stack frame

imamo funkciju koja vraća zbir dva prosleđena broja:

**int saberi(int a, int b) {**

**int c = a + b;**

**return c;**

**}**

pozivamo: **saberi(1, 2)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld sp, 0x80 | 0x12 | 0x80 | napuni sp brojem 0x80 (inicijalizuj stek) |
| 0x02 | ld b, 8 | 0x0a | 0x08 | stavi u registar b proizvoljan broj (8); neka to bude tekući stack frame pointer |
| 0x04 | push 0x01 | 0x70 | 0x01 | stavi na stek broj 1 (prvi parametar funkcije) |
| 0x06 | push 0x02 | 0x70 | 0x02 | stavi na stek broj 2 (drugi parametar funkcije) |
| 0x08 | call 0x11 | 0x86 | 0x11 | poziv potprograma na adresi 0x11 |
| 0x0a | sub sp, 2 | 0x20 | 0x02 | smanji sp za 2, čime sklanjamo parametre funkcije sa steka |
| 0x0c | add a, 48 | 0x38 | 0x30 | saberi rezultat funkcije (u registru a) sa konstantom 48, da bi jednocifreno rešenje rada funkcije pretvorio u cifru za ekran (u a je 0x03, pa će na ekranu biti ‘3’) |
| 0x0e | st a, [0xfff0] | 0x8c | 0xf0 | smesti karakter u video memoriju, na prvu poziciju |
| 0x10 | halt | 0xff |  |  |
| 0x11 | push b | 0x78 |  | stavi registar b na stek |
| 0x12 | ld b, sp | 0x14 |  | smesti u registar b sadržaj registra sp |
| 0x13 | add sp, 2 | 0x1c | 0x02 | dodamo 0x02 na sp da iskocimo izvan stack frame-a i da sp bude spreman za nov poziv potprograma  sabrali smo ga sa dva po formuli: add sp, <broj\_lokalnih\_promenljivih> + 1 |
| 0x15 | ld a, [b – 3] | 0x0e | 0xfd | smesti u registar a sadržaj memorijske lokacije na koju ukazuje razlika b – 3 (drugi parametar funkcije – broj 2) |
| 0x17 | add a, [b - 4] | 0x2c | 0xfc | saberi a i sadržaj lokacije na koju ukazuje razlika b – 4 (prvi parametar funkcije – broj 1); u a je sada 3 |
| 0x19 | st a, [b + 1] | 0x96 | 0x01 | smesti u memorijsku lokaciju na koju ukazuje zbir b + 1 sadržaj registra a; to znači da u lokalnu promenljivu smeštamo rezultat sabiranja (broj 0x03) |
| 0x1b | ld sp, b | 0x16 |  | vrati u sp sadržaj registra b |
| 0x1c | pop b | 0x7a |  | vrati u b sačuvanu vrednost sa steka |
| 0x1d | ret | 0x8a |  | vrati se iz potprograma |

## Primer22- MUL A, B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 3 | 0x08 | 0x03 | napuni a brojem 3 |
| 0x02 | ld b, -4 | 0x0a | 0xfffc | napuni b brojem -4 (0xfffc) |
| 0x04 | mul a, b | 0x01 |  | pomnoži a i b; donji bajt rezultata je u a, a gornji u HIGH registru u ALU; u a je -12 (0xfff4) |
| 0x05 | ld b, 14 | 0x0a | 0x0e | napuni b brojem 14 (0x0e) |
| 0x07 | add a, b | 0x18 |  | saberi a i b i smesti rezultat u a; sada je u a 2 (0x0002) |
| 0x08 | halt | 0xff |  | zaustavlja računar |

## Primer23- DIV A, B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, -8 | 0x08 | 0xfff8 | napuni a brojem -8 |
| 0x02 | ld b, 4 | 0x0a | 0x04 | napuni b brojem 4 |
| 0x04 | div a, b | 0x15 |  | podeli a sa b; rezultat je u a, a ostatak u HIGH registru u ALU; u a je -2 (0xfffe) |
| 0x05 | ld b, 12 | 0x0a | 0x0c | napuni b brojem 12 (0x0c) |
| 0x07 | add a, b | 0x18 |  | saberi a i b i smesti rezultat u a; sada je u a 10 (0x0a) |
| 0x08 | halt | 0xff |  | zaustavlja računar |

## Primer24- Primer dobavljanja HIGH-BYTE iz ALU

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 30000 | 0x08 | 0x7530 | napuni a brojem 30000 |
| 0x02 | ld b, 10 | 0x0a | 0x0a | napuni b brojem 10 |
| 0x04 | mul a, b | 0x15 |  | pomnoži a i b; zbir je sada 300000 (0x0493e0); donja reč je u a, a gornja u h, pa to znaci da je u a sada -37856 (0x93e0), a u HIGH je 4 (0x04) |
| 0x05 | ld a, h | 0xfa |  | napuni a vrednošću iz HIGH-BYTE iz ALU (registar h) – sada je u a broj 4 |
| 0x06 | neg a | 0x6e |  | potpuni komplement od a – sada je u a -4, odn. 0xfffc |
| 0x07 | div a, b | 0x15 |  | podeli a i b; delimo -4 i 10; rezultat je 0, a ostatak -4 (0xffcd) |
| 0x08 | ld a, h | 0xfa |  | prebaci h u a; sada je u a -4 (0xfffc) |
| 0x09 | halt | 0xff |  | zaustavlja računar |

## 

## Primer25- HELLO WORLD i registar C

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld b, 0x10 | 0x0a | 0x10 | napuni b brojem 0x10 |
| 0x02 | ld c, 0 | 0x2d | 0x00 | napuni c brojem 0 |
| 0x04 | ld a, [b] | 0x0c |  | napuni a sadržajem memorijske lokacije na koju ukazuje b |
| 0x05 | cmp a, 0 | 0x46 | 0x00 | poredi a o u, ako su jednaki, ZERO je 1 |
| 0x07 | jz 0x0f | 0x7e | 0x0f | ako je ZERO jedinica, idi na lokaciju 0x0f |
| 0x09 | st a, [c + 0xfff0] | 0x31 | 0xf0 | smesti sadržaj a u lokaciju na koju ukazuje zbir registra c i broja 0xfff0 (video memorija) |
| 0x0b | inc b | 0x50 |  | povećaj b za jedan |
| 0x0c | inc c | 0x37 |  | povećaj c za jedan |
| 0x0d | jmp 0x04 | 0x7c | 0x04 | skoči na lokaciju 0x04 |
| 0x0f | halt |  |  | zaustavi računar |
| 0x10-0x1b | “HELLO WORLD\0” | 0x48 0x45 0x4c 0x4c 0x4f 0x20 0x57 0x4f 0x52 0x4c 0x44 0x00 |  | String “HELLO WORLD” sa nulom na kraju |

## 

## Primer26- Rekurzivna funkcija faktorijel

**int faktorijel(int num) {**

**if (num < 2)**

**return 1;**

**return num \* faktorijel(num – 1)**

**}**

pozivamo: **faktorijel(4)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld sp, 0x80 | 0x12 | 0x80 | postavljamo sp na 0x80 |
| 0x02 | ld b, 8 | 0x0a | 0x08 | napuni b brojem 0x08 |
| 0x04 | push 6 | 0x70 | 0x04 | stavi parametar funkcije faktorijel na stek (broj 6) |
| 0x06 | call 0x10 | 0x86 | 0x10 | pozovi funkciju faktorijel |
| 0x08 | sub sp, 1 | 0x20 | 0x01 | počisti sa steka parametar funkcije (ovo uvek radimo kada se vratimo iz potprograma) |
| 0x0a | push a | 0x74 |  | stavimo rezultat rada funkcije faktorijel na stek, kao parametar funkciji za ispis |
| 0x0b | call 0x2a | 0x86 | 0x2a | pozovi potprogram za ispis broja |
| 0x0d | sub sp, 1 | 0x20 | 0x01 | počisti parametar funkcije sa steka |
| 0x0f | halt |  |  | zaustavi računar |
| 0x10 | push b | 0x78 |  | Početak funkcije faktorijel  stavi b (stack frame pointer na stek) |
| 0x11 | ld b, sp | 0x14 |  | napuni b vredošću od sp |
| 0x12 | add sp, 1 | 0x1c | 0x01 | preskočili smo tekući stack frame, tako da sp bude spreman za novi poziv potprograma  sabrali smo ga sa dva po formuli: add sp, <broj\_lokalnih\_promenljivih> + 1 |
| 0x14 | ld a, [b - 3] | 0x0e | 0xfd | stavimo u a parametar funkcije |
| 0x16 | cmp a, 2 | 0x46 | 0x02 | poredimo a i 2; poređenje se radi oduzimanjem, pa ako je u a broj veći ili jednak 2, POSITIVE flag će biti 1 |
| 0x18 | jp 0x1e | 0x9a | 0x1e | ako je POSITIVE flag 1, skoči na 0x1e  u prevodu, ako je a >=2, skoči na 0x1e; inače pređi na sledeći instrukciju |
| 0x1a | ld a, 1 | 0x08 | 0x01 | za vrednosti parametra manje od 2 (za 1 ili 0), vrednost faktorijela je 1, pa to i stavljamo u a |
| 0x1c | jmp 0x27 | 0x7c | 0x27 | skoči na kraj funkcije, gde čistimo stack frame i vraćamo se nazad |
| 0x1e | sub a, 1 | 0x3c | 0x01 | smanjimo parametar za jedan |
| 0x20 | push a | 0x74 |  | pozvaćemo faktorijel od ovog umanjenog broja, pa pre poziva stavljamo umanjeni parametar na stek, kao parametar funkcije faktorijel |
| 0x21 | call 0x10 | 0x86 | 0x10 | pozovemo rekurzivno samog sebe |
| 0x23 | sub sp, 1 | 0x20 | 0x01 | počistimo stek po povratku iz potprograma |
| 0x25 | mul a, [b - 3] | 0x41 | 0xfd | pomnožimo parametar funkcije sa povratnom vrednošću rekurzivnog poziva |
| 0x27 | ld sp, b | 0x16 |  | čistimo stack frame pre povratka iz funkcije; vraćamo stari sp |
| 0x28 | pop b | 0x7a |  | vraćamo stari stack frame pointer |
| 0x29 | ret | 0x8a |  | povratak |

## Primer27- MUL A, C

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 3 | 0x08 | 0x03 | napuni a brojem 3 |
| 0x02 | ld c, -30000 | 0x2d | 0x8ad0 | napuni b brojem -30000 (0x8ad0) |
| 0x04 | mul a, c | 0x5f |  | pomnoži a i c; rezultat je -300000 (0xfffea070); donja reč rezultata je u a, a gornja u HIGH registru u ALU; u a je (0xa070), u HIGH-BYTE-u je (0xfffe) |
| 0x05 | halt | 0xff |  | zaustavlja računar |

## Primer28- DIV A, C

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, -8 | 0x08 | 0xfff8 | napuni a brojem -8 |
| 0x02 | ld c, 4 | 0x2d | 0x04 | napuni b brojem 4 |
| 0x04 | div a, c | 0x15 |  | podeli a sa c; rezultat je u a, a ostatak u HIGH registru u ALU; u a je -2 (0xfffe), a ostatak je 0 |
| 0x05 | halt | 0xff |  | zaustavlja računar |

## Primer29- XOR A, B i XOR A, C

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 48 | 0x08 | 0x30 | napuni a brojem 48 (0x30) |
| 0x02 | ld b, 85 | 0x0a | 0x55 | napuni b brojem 85 |
| 0x04 | ld c, 85 | 0x2d | 0x55 | napuni c brojem 85 |
| 0x06 | xor a, b | 0x63 |  | a<-- a xor b  sada je u a 0x65 |
| 0x07 | xor a, c | 0x65 |  | a<-- a xor c  sada je u a 0x30 |
| 0x08 | halt | 0xff |  | zaustavlja računar |

## Primer30- XOR A, [B + 1]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adresa | Asemblerski kod | Prva reč mašinskog koda | Druga reč mašinskog koda | Komentar |
| 0x00 | ld a, 56 | 0x08 | 0x38 | napuni a brojem 56 |
| 0x02 | ld b, 0x80 | 0x0a | 0x80 | napuni b brojem 0x80 |
| 0x04 | st a, [b + 1] | 0x96 | 0x01 | stavi na lokaciju [b + 1] vrednost registra a |
| 0x06 | ld a, 1 | 0x08 | 0x01 | napuni a brojem 1 |
| 0x08 | xor a, [b + 1] | 0x67 | 0x01 | a <-- a xor [b + 1]  sada je u a broj 0x39 |
| 0x0a | xor a, [b + 1] | 0x67 | 0x01 | a <-- a xor [b + 1]  sada je u a broj 1 |
| 0x0c | halt | 0xff |  | zaustavlja računar |