Jakob Jelovčan 63210123

1. Naloga

Opis mikroukazov pri izvedbi strojnega ukaza `li rd, immed`:

- `fetch: addrsel=pc irload=1`
  - V ukazni register se prenese vrednost iz lokacije shranjene v programskem števcu
- `pcload=1 pcsel=pc, opcode_jump`
  - Programski števec se poveča za 1
  - Izvede se skok na mikroukaz na lokaciji opcode + 2
- `addrsel=pc dwrite=1 regsrc=databus, goto pcincr`
  - V register d se shrani vrednost iz lokacije shranjene v programskem števcu
  - Izvede se skok na mikroukaz pcincr
- `pcincr: pcload=1 pcsel=pc, goto fetch`
  - Programski števec se zaradi branja takojšnjega operanda poveča za 1

2. Naloga

V mikrozbirniku za model MiMo sem implementiral vse ukaze. Nekateri izmed njih so spodaj podrobneje opisani. Fetch in pcincr sem opisal že v prvi nalogi zato sem ju pri spodnjih ukazih izpustil.

38: jlt

- `addrsel=pc imload=1`
  - V takojšnji register se prenese vrednost iz lokacije v programskem števcu
- `aluop=sub op2sel=treg, if z then pcincr`
  - Od vrednosti registra s se odšteje vrednost registra t (rezultat se ne shrani)
  - Če je zastavica z postavljena, se izvede skok na mikroukaz pcincr, sicer se nadaljuje izvajanje naslednjega mikroukaza
- `aluop=sub op2sel=treg, if n then jump else pcincr`
  - Od vrednosti registra s se odšteje vrednost registra t (rezultat se ne shrani)
  - Če je postavljena zastavica n se izvede skok na mikroukaz jump, sicer pa se izvede skok na mikroukaz pcincr
- `jump: pcload=1 pcsel=immed, goto fetch`
  - V programski števec se prenese vrednost takojšnjega operanda
  - Izvede se skok na naslednji ukaz

59: jsr

- `addrsel=pc imload=1`
  - V register za takojšnji operand se shrani vrednost iz naslova v programskem števcu
- `aluop=sub op2sel=const1 swrite=1 regsrc=aluout`
  - Od vrednosti registra s se odšteje 1. Rezultat se rezultat shrani v register s
- `pcload=1 pcsel=pc`
  - Programski števec se poveča za 1
- `datawrite=1 datasel=pc addrsel=sreg`
  - Na naslov v registru s se shrani trenutna vrednost programskega števca
- `pcload=1 pcsel=immed, goto fetch`
  - V programski števec se prenese vrednost iz registra za takojšnji operand
  - Izvede se skok na naslednji ukaz

60: rts

- `addrsel=sreg imload=1`
  - V register za takojšnji operand se prenese vrednost iz naslova v registru s
- `aluop=add op2sel=const1 swrite=1 regsrc=aluout`
  - Vrednosti registra s se prišteje konstanta 1. Rezultat se shrani v register s
- `pcload=1 pcsel=immed, goto fetch`
  - V programski števec se prenese vrednost iz registra za takojšnji operand
  - Izvede se skok na naslednji ukaz

69: pop

- `addrsel=sreg regsrc=databus dwrite=1`
  - Iz naslova v registru s se prenese vrednost v register d
- `aluop=add op2sel=const1 swrite=1 regsrc=aluout, goto fetch`
  - Vrednosti registra s se prišteje konstanta 1. Rezultat se shrani v register s
  - Izvede se skok na naslednji ukaz

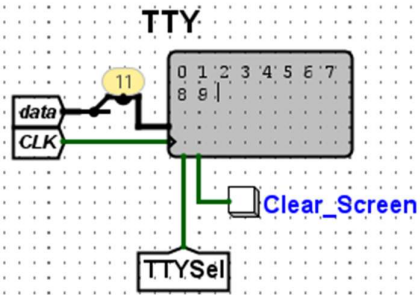
72: neg

- `aluop=nor op2sel=const0 swrite=1 regsrc=aluout`
  - Vrednost registra s se invertira (nor z 0) v eniški komplement. Rezultat se shrani v register s
- `aluop=add op2sel=const1 swrite=1 regsrc=aluout, goto fetch`
  - Vrednosti registra s se prišteje konstanta 1 (dvojiški komplement). Rezultat se shrani v register s
  - Izvede se skok na naslednji ukaz

3. Naloga

Vsebina in dogajanje ob izvedbi posameznih strojnih ukazov sta opisana v nalogi 2.
Za testiranje strojnih ukazov sem napisal program, ki na TTY izhod izpiše števila med 0 in 9.
V programu sem uporabil ukaze, ki so zapisani v spodnji tabeli, skupaj z njihovim časom izvajanja:

Ukaz	Trajanje ukaza v urinih periodah
`li`	4
`add`	3
`remi`	5
`lsl`	3
`jsr`	7
`inc`	3
`dec`	3
`jlt`	5 če je postavljen z, 6 sicer
`jgt`	5 če je postavljen z, 6 sicer
`jmp`	4
`push`	4
`swi`	5
`pop`	4
`rts`	5
`divi`	5
`jnez`	5

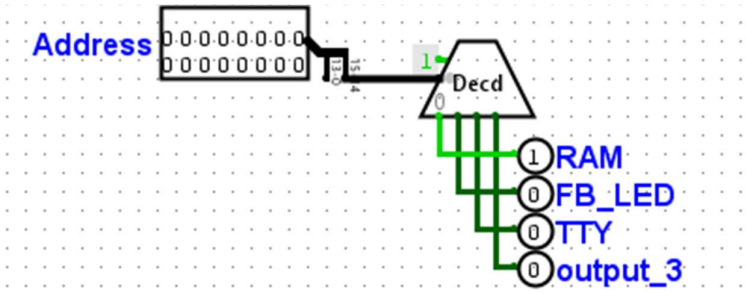


Slika 1: Izpis programa naloge 3

4. Naloga

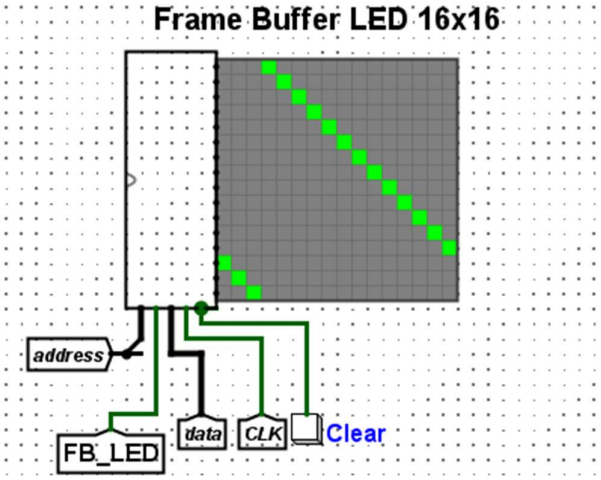
Z razdelivcem se iz naslova prebere zgornja dva bita (bit 15 in 14) ter se njuno vrednost posreduje na dekodirnik. Ta glede na kontrolni signal enega izmed izhodov postavi na 1.

b15, b14	Omogočen izhod
00	RAM
01	FB_LED
10	TTY
11	output_3



Slika 2: Popravljeno naslovno dekodiranje

Za pomnilniškega dekodiranja sem v zbirniku za MiMo sem napisal program, ki na FB zaslon izriše diagonalno črto, ki se v neskončni zanki pomika od desne proti levi.

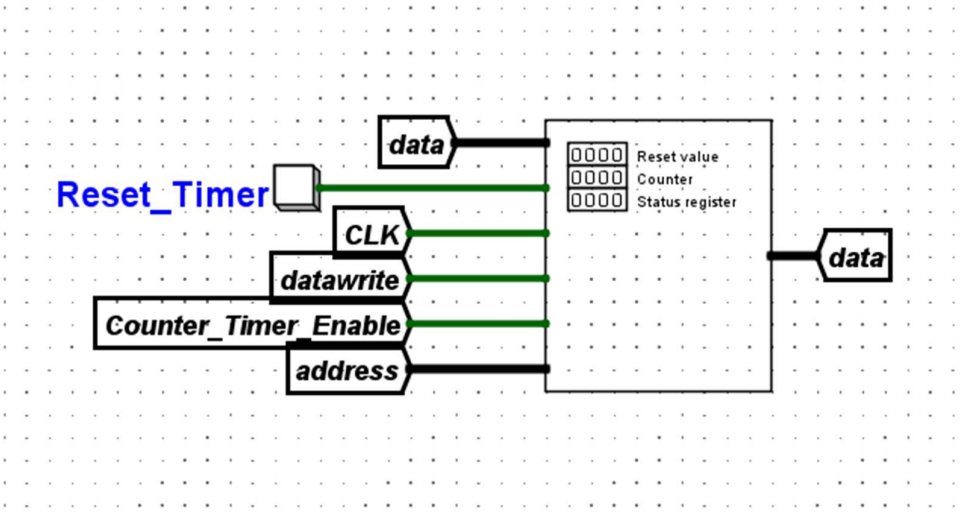


Slika 3: Izris programa naloge 4

5. Naloga

Za dodajanje nove vhodno/izhodne naprave je potrebno:

- Prazen (četrti) izhod naslovnega dekodirnika je potrebno povezati z »enable« signalom naprave
- Naslovni vhod naprave je potrebno povezati z naslovnim vodilom
- Podatkovni vhod in izhod naprave je potrebno povezati s podatkovnim vodilom
- Če naprava podpira branje in pisanje, je potrebno nanjo povezati še signal »datawrite« zato, da naprava ve kdaj prihaja do branja in kdaj do pisanja
- Na napravo je potrebno povezati urin signal



Slika 4: Priključitev vhodno/izhodne naprave

Za demonstracijo priključitve vhodno izhodne naprave na model MiMo sem izdelal preprost števec/časovnik, ki omogoča odštevanje od poljubne vrednosti do 0 ter štetje od 0 do $2^{16} - 1$. Ob vrednosti 0 se na statusnem registru bit 3 postavi na 1. Če je števec v načinu odštevanja, se ob dosegu vrednosti 0 avtomatsko ponastavi na vrednost v »reset value« registru.

Enota vsebuje 5 registrov:

- 0x0 statusni register (branje)
- 0x1 »set« register (pisanje)
- 0x2 »clear« register (pisanje)
- 0x3 »reset value« register (branje in pisanje)
- 0x4 »current value« register (branje)

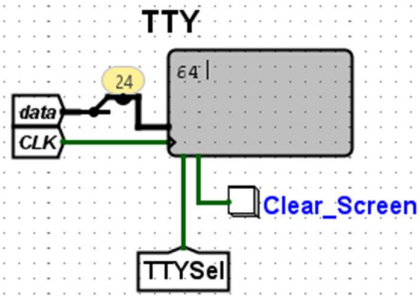
V statusnem registru so uporabljeni prvi štirje biti:

- Bit 0: vključitev/izključitev naprave
- Bit 1: 0 odštevanje, 1 prištevanje
- Bit 2: ponastavitev števca na vrednost 0 (vrednost se ne zapiše v statusni register)
- Bit 3: zastavica

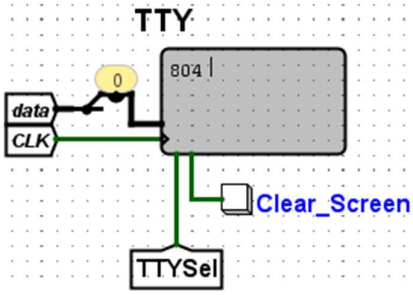
Zastavica se ob pisanju ali branju v statusni register izbriše.

Za testiranje naprave sem napisal dva programa:

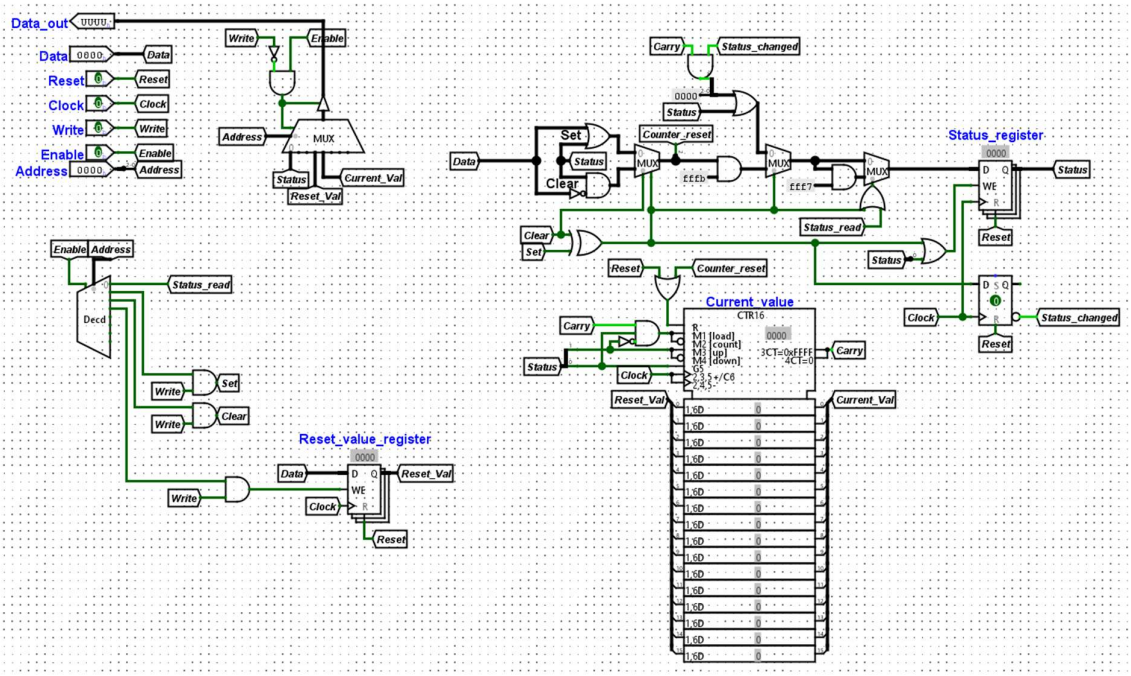
- Program a ima zanko, ki se izvaja dokler ne preteče 1000 urinih period nato na TTY izpiše število iteracij
- Program b ima zanko, ki se ponovi 100x, nato pa se število pretečenih urinih period izpiše na TTY



Slika 5: Izpis programa a naloge 5



Slika 6: Izpis programa b naloge 5



Slika 7: Vezje števca