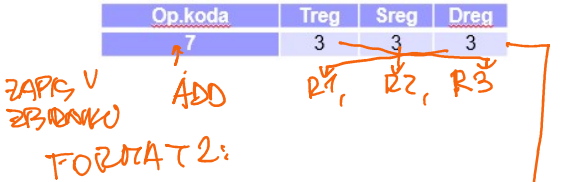
3.2 MiMo - model CPE

ponedeljek, 02. november 2020 17:37

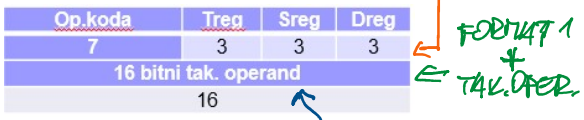
Značilnosti :

- pomnilniška beseda 16 BITOV
- pomnilniški naslov 16 BITNI
- dolžina ukazov 16 ali 32 bitov (2 formata)
  - Format 1 : (primer **ADD R1,R2,R3** # R1<-R2+R3, R1=Dreg, R2=Sreg, R3=Treg)
  - Format 2 : (primer **LI R1, 100** # R1<-100)

FORMAT 1:



FORMAT 2:



LI R1, 100 @ R1 ← 100

- registri:
  - 8x 16bitnih splošno namenskih registrov R0-R7
- operandi (pomnilniški dostopi) SAMO 16-BITNI
- pomnilniško preslikan vhod/izhod
- ~~prekinitve~~

MiMo temelji na tem viru: http://minnie.tuhs.org/Programs/UcodeCPU/index.html

3.2.1 Izvrševanje ukazov

ponedeljek, 02. november 2020 17:42

"FETCH": UKAZNO PREVZEMNI CIKEL

"EXECUTE": IZVRŠILNI CIKEL

"ELEMENTARNI KORAKI" ← mikro

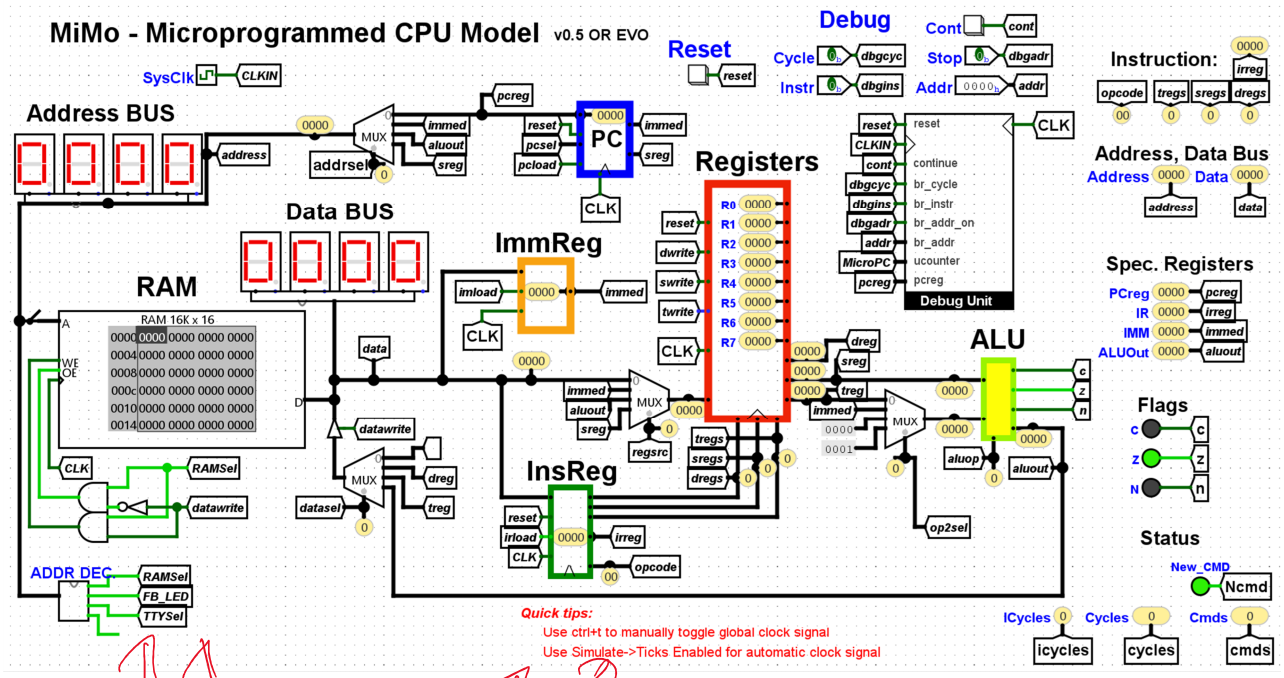
→ "VEČPERIODNA REALIZACIJA"

- BRANJE UKAZA
- DEKODIRANJE (ANALIZA) UKAZA
- PRENOS OPERANDOV
- IZVEDBA OPERACIJE (ALE)
- SHRANITEV REZULTAT

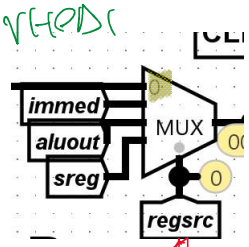
• OBNOVITEV PC je PC ← PC+1

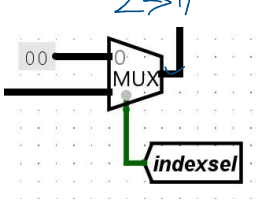
3.2.2 Podatkovna enota

ponedeljek, 02. november 2020 17:42

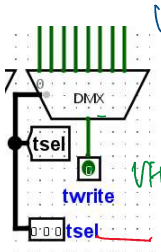


MUX-MULTIPLEXER

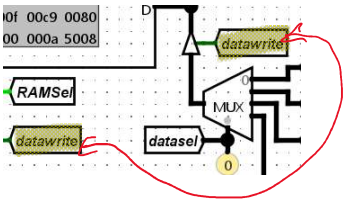




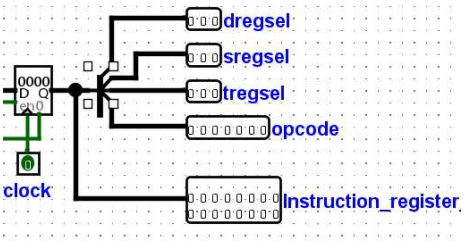
DEMUX-DEMULTIPLEXER



TUNELI (POVEZAVE)

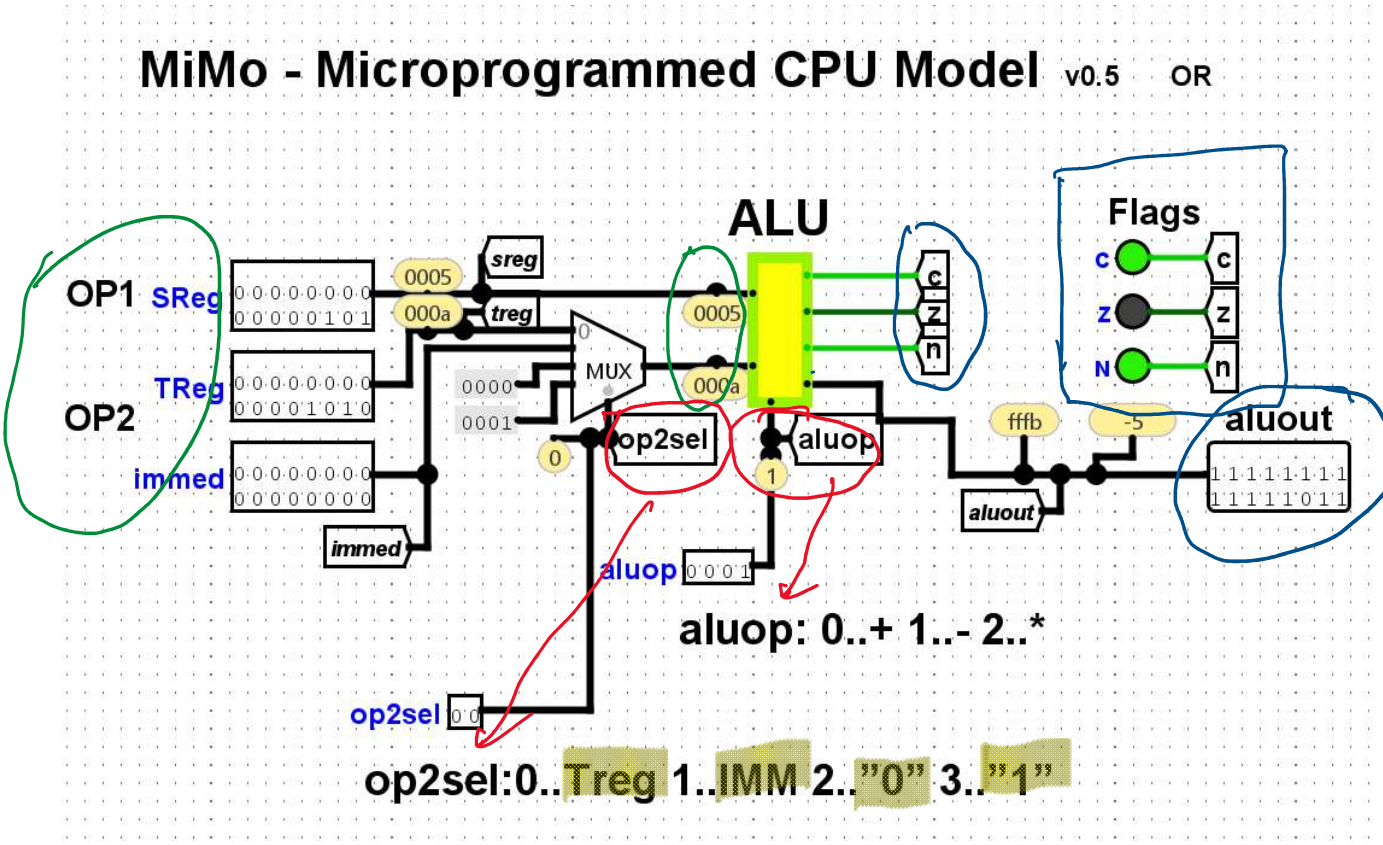


SPLITTER (RAZDELILNIK)



ALE

ponedeljek, 02. november 2020 17:45



ALUOP: VRSTA OPERACIJE

OP2SEL: DOLOČA 2. OPERAND

add Rd,Rs,Rt (0)

Rd <- Rs + Rt

ALUOP = 0

OP2SEL = 0 (TREG)

PC <- PC + 1

jnez Rs,immed (40)

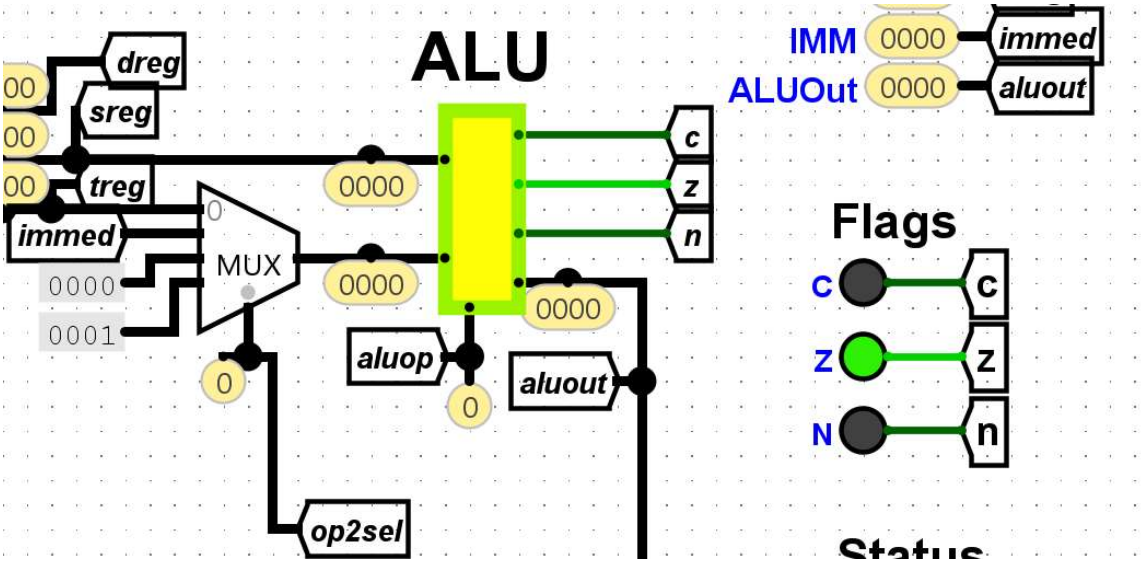
if Rs != 0, PC <- immed else PC <- PC + 2

ALU: R_s != 0

$R_s - 0 \neq 0$ 2 zast.

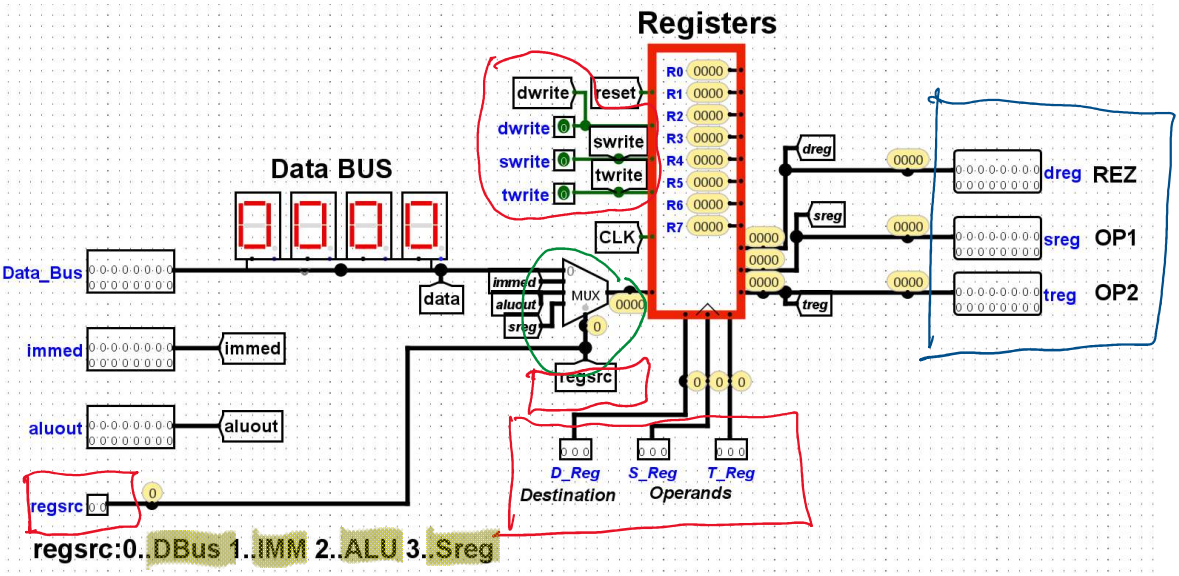
ALUOP: 1

OP2SEL = 2 ("0")

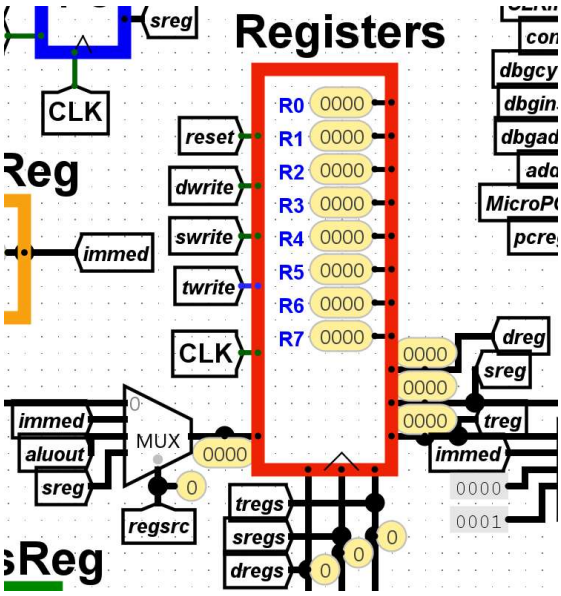

ALU
dreg
sreg
treg
immed
0000
0001
MUX
0
0000
0000
0000
aluop
0
aluout
c
z
n
op2sel
IMM
0000
immed
ALUOut
0000
aluout
Flags
C
Z
N
c
z
n

Registri

ponedeljek, 02. november 2020 17:46

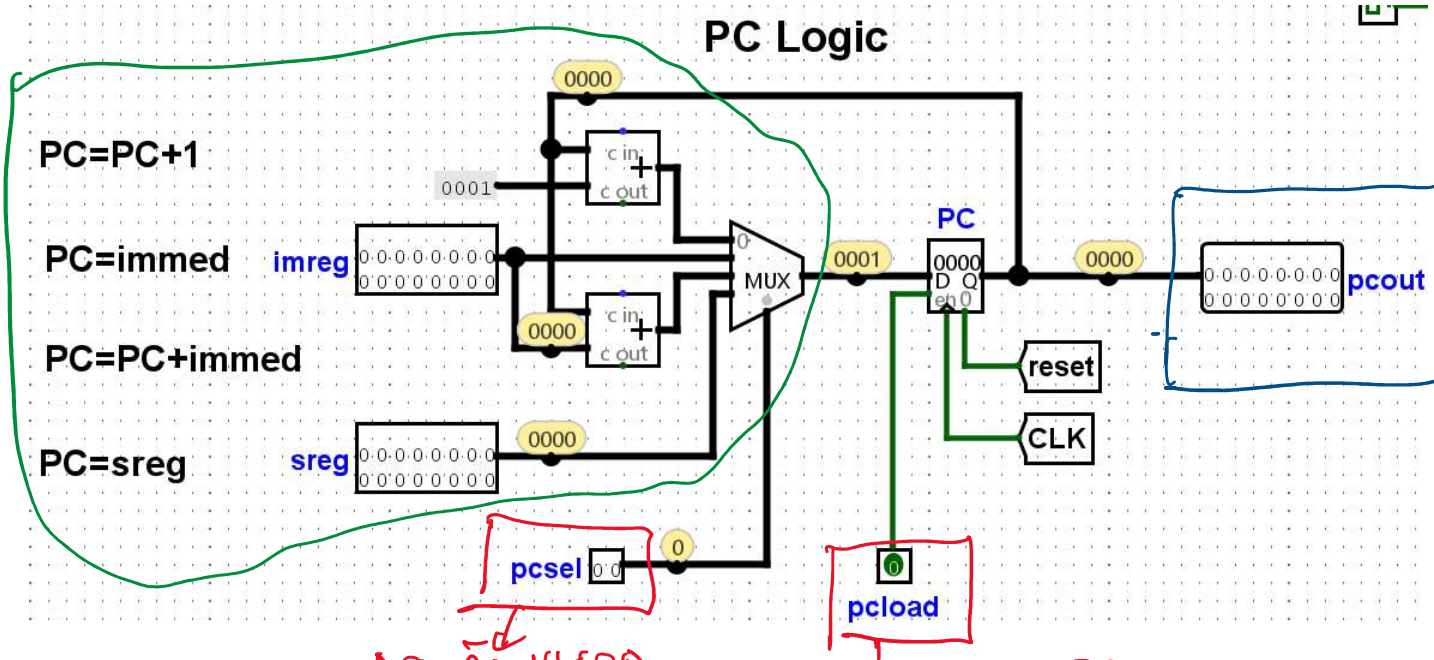


add Rd,Rs,Rt (0)		move Rd,Rs (70)		li Rd,immed (63)	
Rd <- Rs + Rt	PC <- PC + 1	Rd <- Rs	PC <- PC + 1	Rd <- immed	PC <- PC + 2
REGSRC = 2 DWRITE = 1		REGSRC = 3 (SREG) DWRITE = 1		REGSRC = 0 (DBUS) DWRITE = 1	



PC - programski števec

ponedeljek, 02. november 2020 17:49



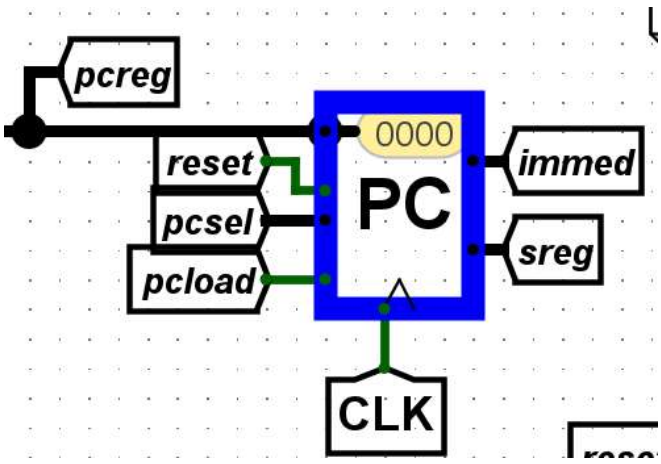
DOLOČI VHOD

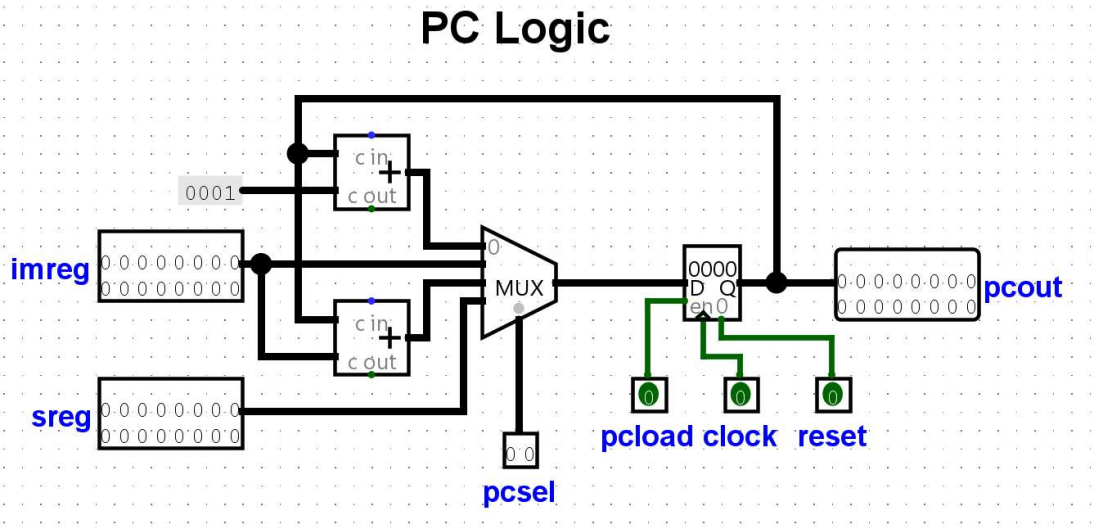
VHODI:
- PC+1
- IMMED (ABS. SKOK)
- PC+IMMED (REL. SKOK)
- SREG ("MOV PC,LR")

VPIS V PC

PCLOAD = 1 ⇒ VPIS

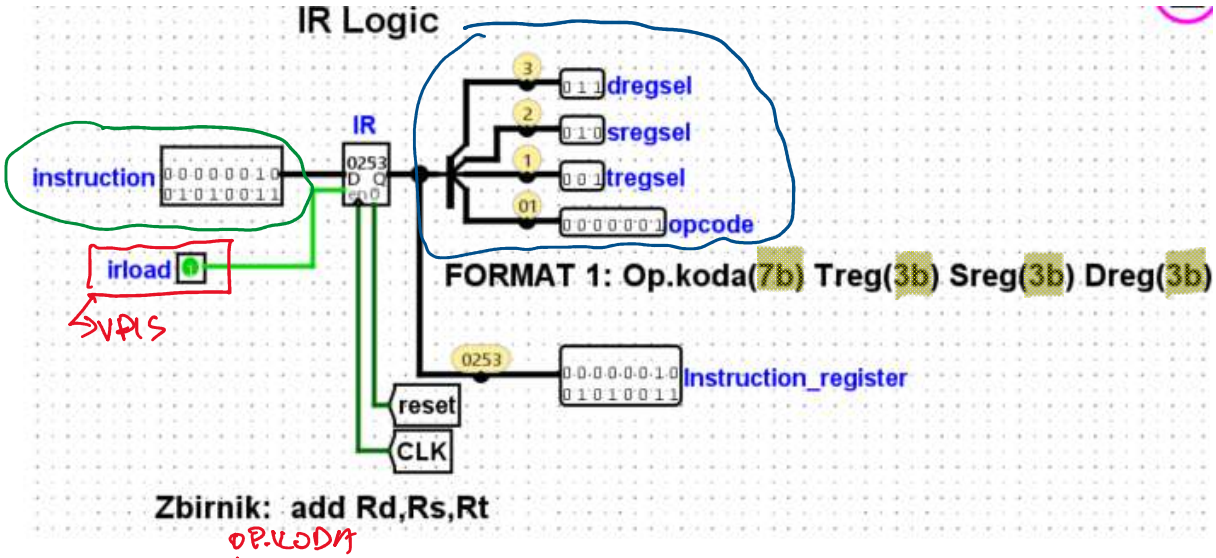
PCSEL ⇒ DOLOČI NOVO VSEBINO




PC Logic
0001
c in
c out
c in
c out
imreg
sreg
0
MUX
0000
D Q
en
pcout
pcload
clock
reset
pcsel

IR - ukazni register

ponedeljek, 02. november 2020 17:50



OP.KODA

sub Rd,Rs,Rt (1)

Rd <- Rs - Rt PC <- PC + 1

D S T

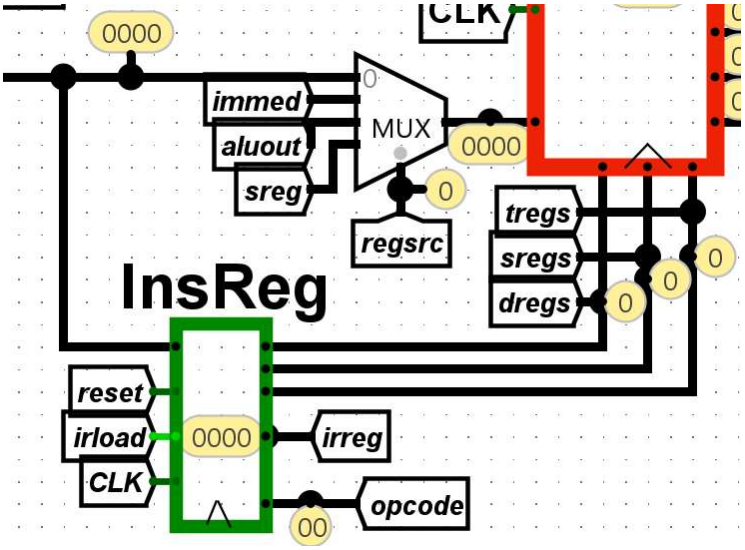
SUB R3,R2,R1

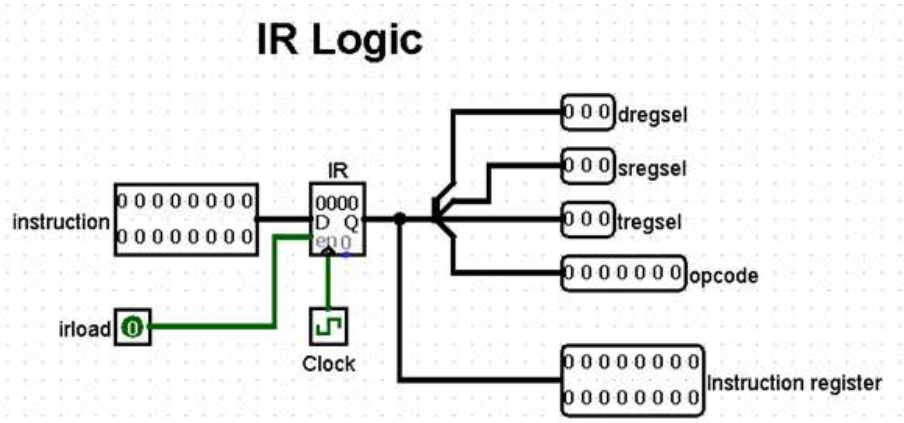
add Rd,Rs,Rt (0)

Rd <- Rs + Rt PC <- PC + 1

Op.koda	Treg	Sreg	Dreg
7	3	3	3
1	1	2	3

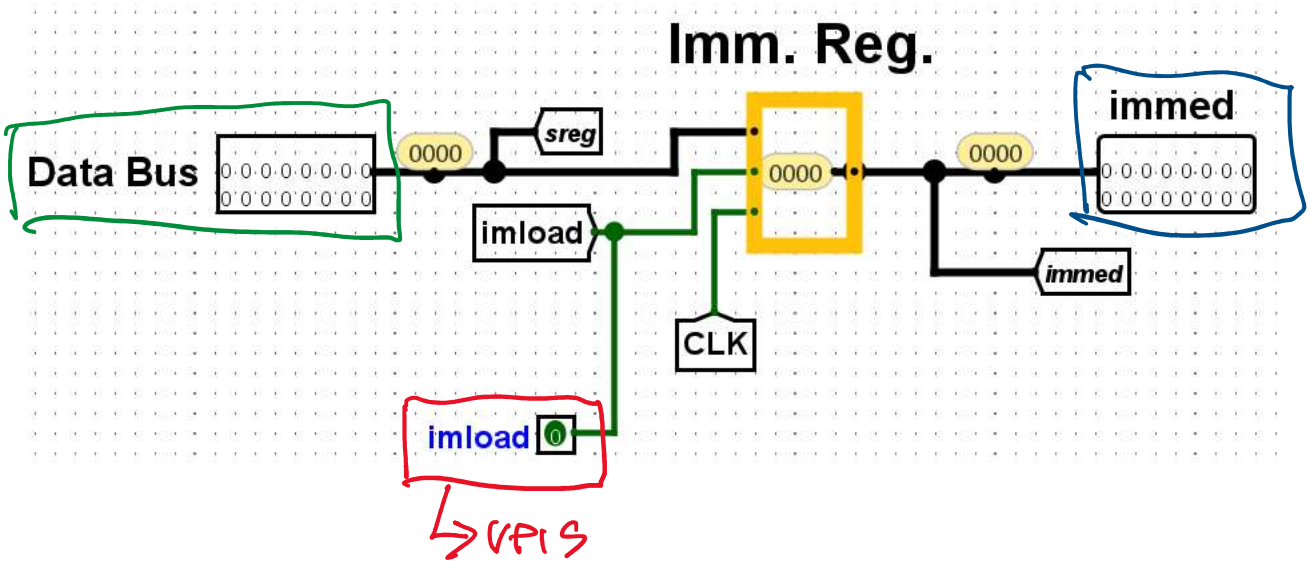
FETCH: • BRANJE UKAZA: IRLOAD = 1 (ADDRSEL = PC)


CLK
0000
immed
aluout
sreg
MUX
0000
0
regsrc
tregs
sregs
dregs
0
0
0
InsReg
reset
irload
CLK
0000
irreg
opcode
00


IR Logic
IR
instruction
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0000
D Q
irload
Clock
0 0 0 dregsel
0 0 0 sregsel
0 0 0 tregsel
0 0 0 0 0 0 0 opcode
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Instruction register

IM - Takojšnji register

ponedeljek, 02. november 2020 17:51



jnez Rs,immed (40)

if Rs != 0, PC <- immed else PC <- PC + 2

BRANJE TAK.O PERIFERIJA:

IMLOAD = 1

(POTUJEMO TAK.OP.)

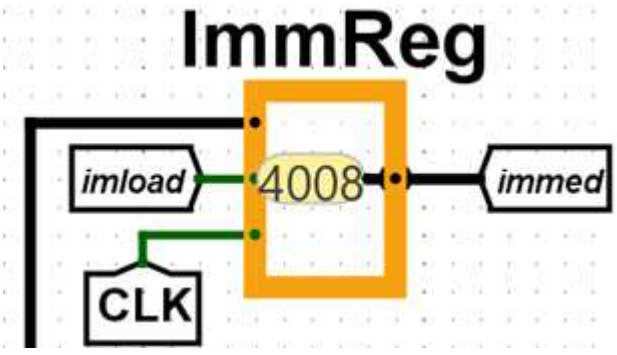
li Rd,immed (63)

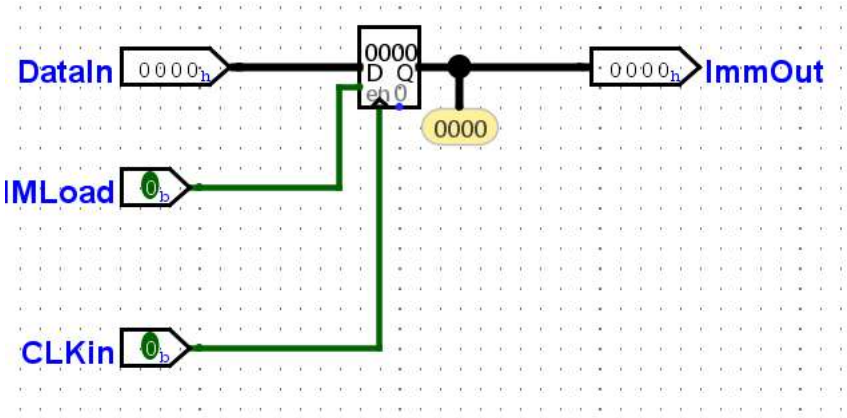
Rd <- immed PC <- PC + 2

TAK.OP → REG.

IMLOAD = 0

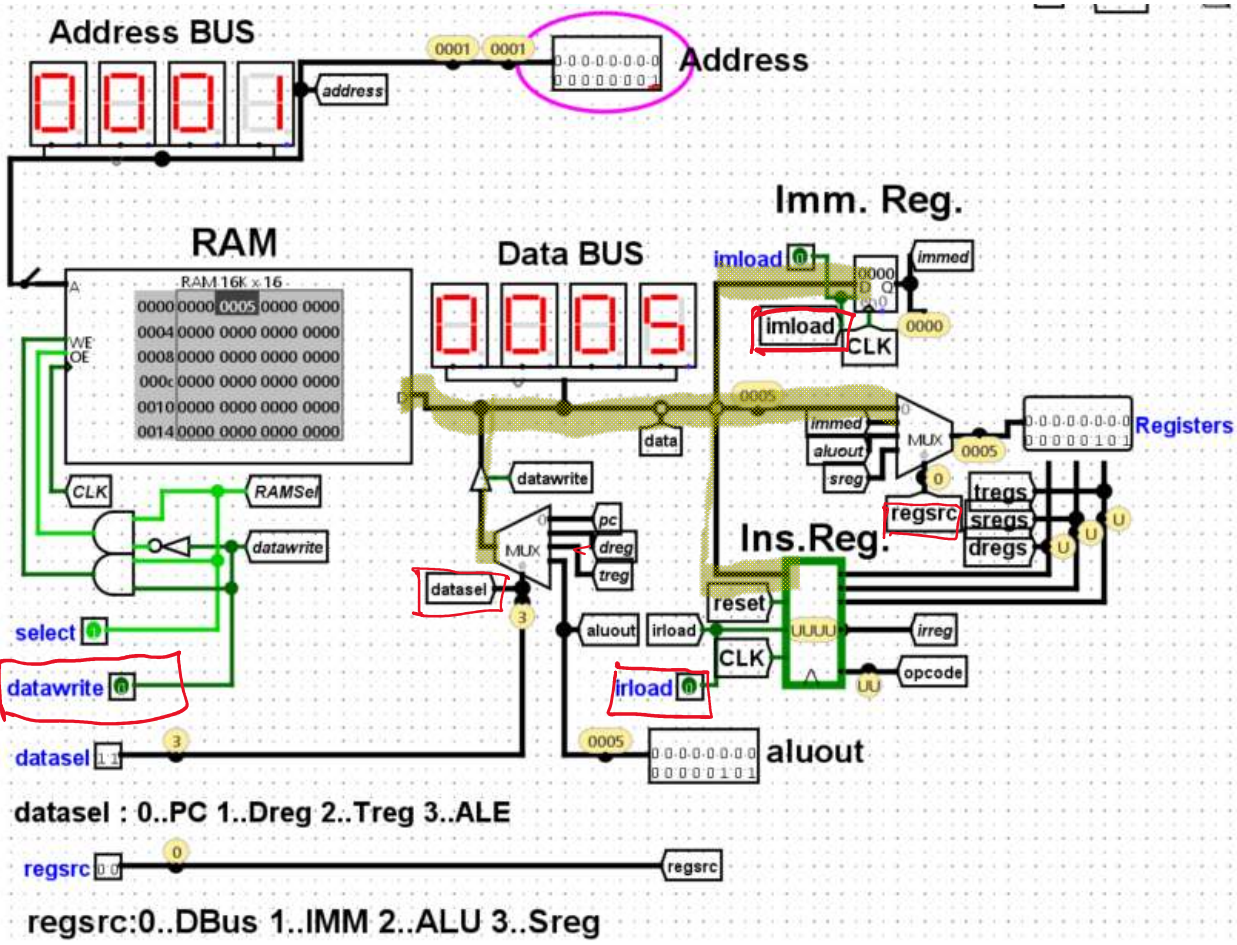
(VPIS V REGISTER DIREKTNO)

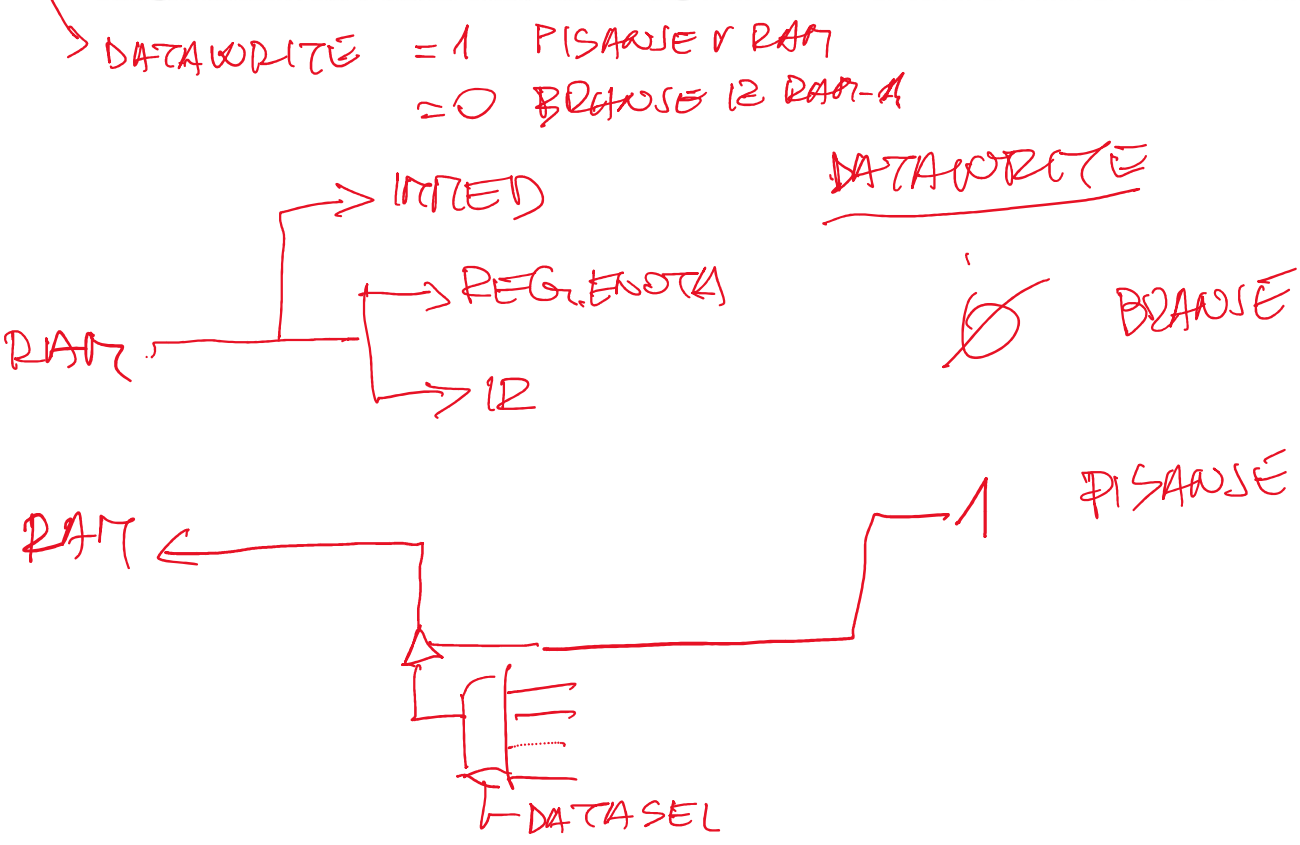



DataIn
0000h
0000
D Q
eh 0
0000
0000h
ImmOut
IMLoad
0b
CLKin
0b

Podatkovno vodilo

ponedeljek, 02. november 2020 17:52





sw Rd,immed (65)

M[immed] <- Rd PC <- PC + 2

DATAWRITE=1

DATASEL=DREG

ADDRSEL=IMMED

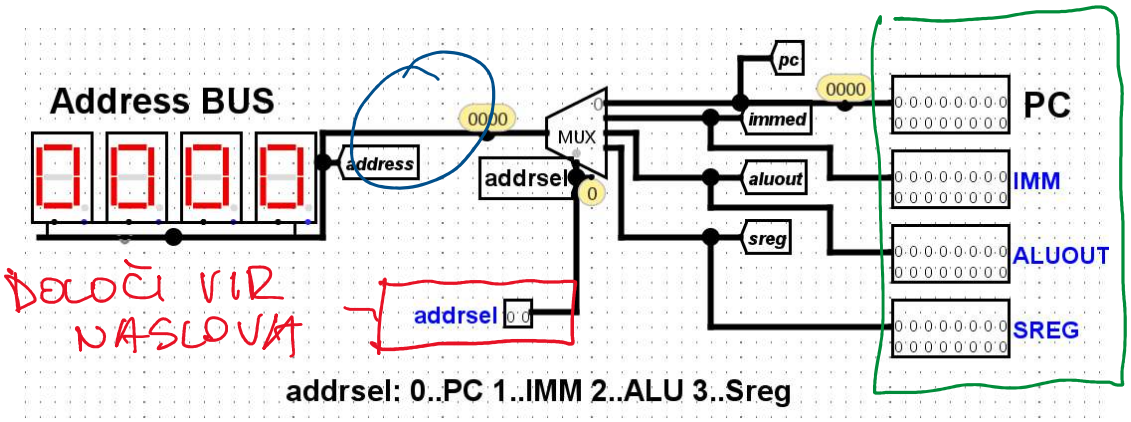
BRANJE IZ RAM-A

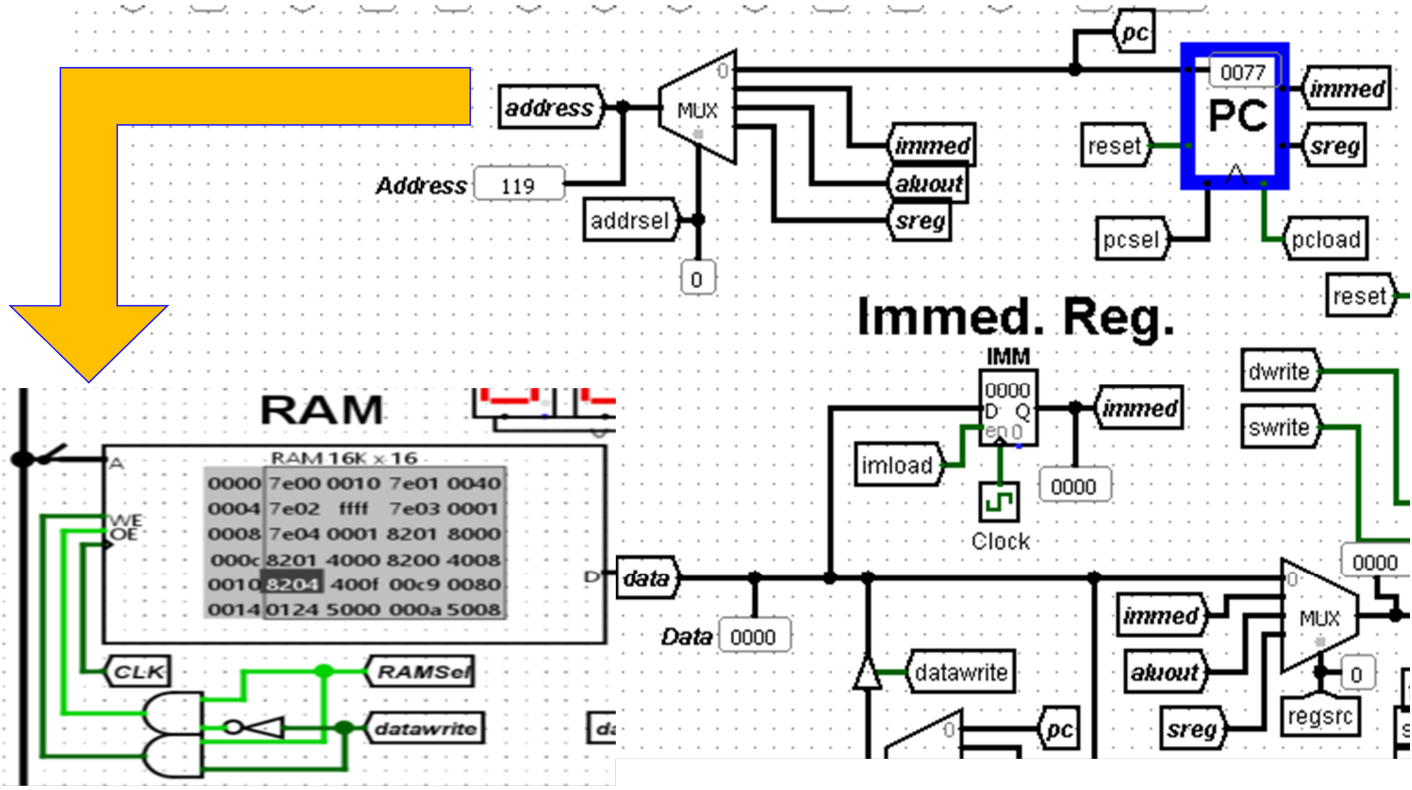
DATAWRITE=0

ADDR -- NASLOV

Naslovno vodilo

ponedeljek, 02. november 2020 17:54





ADDRSEL

IMMED (1)

sw Rd,immed (65)

M[immed] <- Rd PC <- PC + 2

ALUOUT (2)

lwi Rd,Rs,immed (66)

Rd <- M[Rs+immed] PC <- PC + 2

ALUOUT (2)

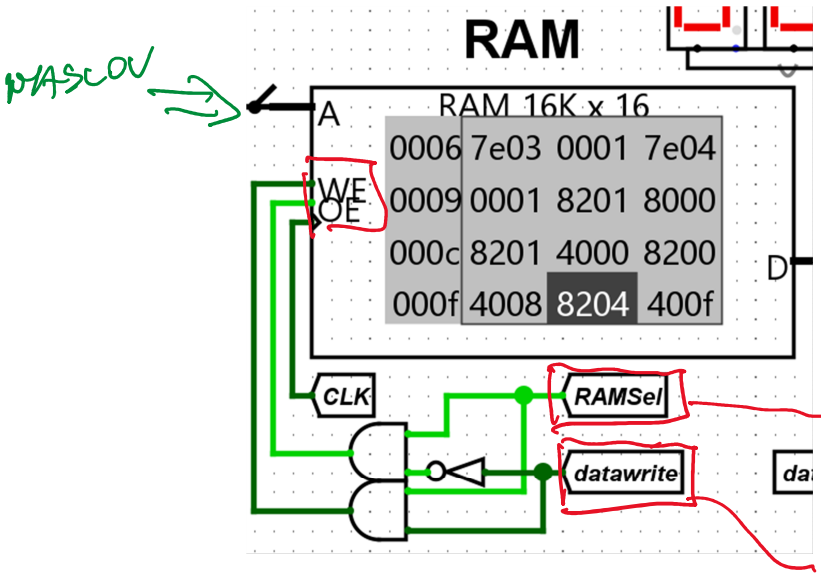
lwri Rd,Rs,Rt (73)

Rd <- M[Rs+Rt] PC <- PC + 1

RAM pomnilnik

ponedeljek, 02. november 2020 17:54

RAM: WE _ WRITE ENABLE (PISANJE)
OE _ OUTPUT ENABLE (BRANJE)



NASLOV

PODATKOVNO VODILO

"CHIP SELECT"

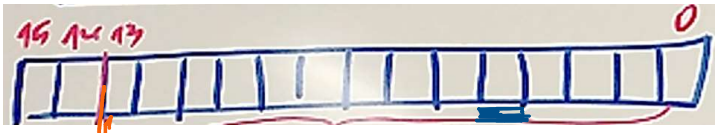
RAM
1 .. AKTIVEN
0 .. NEAKTIVEN

"DATAWRITE"

1 ... PIS
0 .. BRANJE

NEPOSREDNO

NASLOVNO DEKODIRANJE



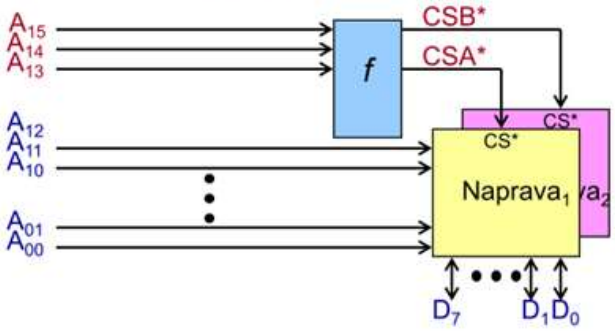
00 RAM
01 FB-LED
10 TTY
11 PROST

NASLOVI

0 - 16383
16384 - 32767
32768 -

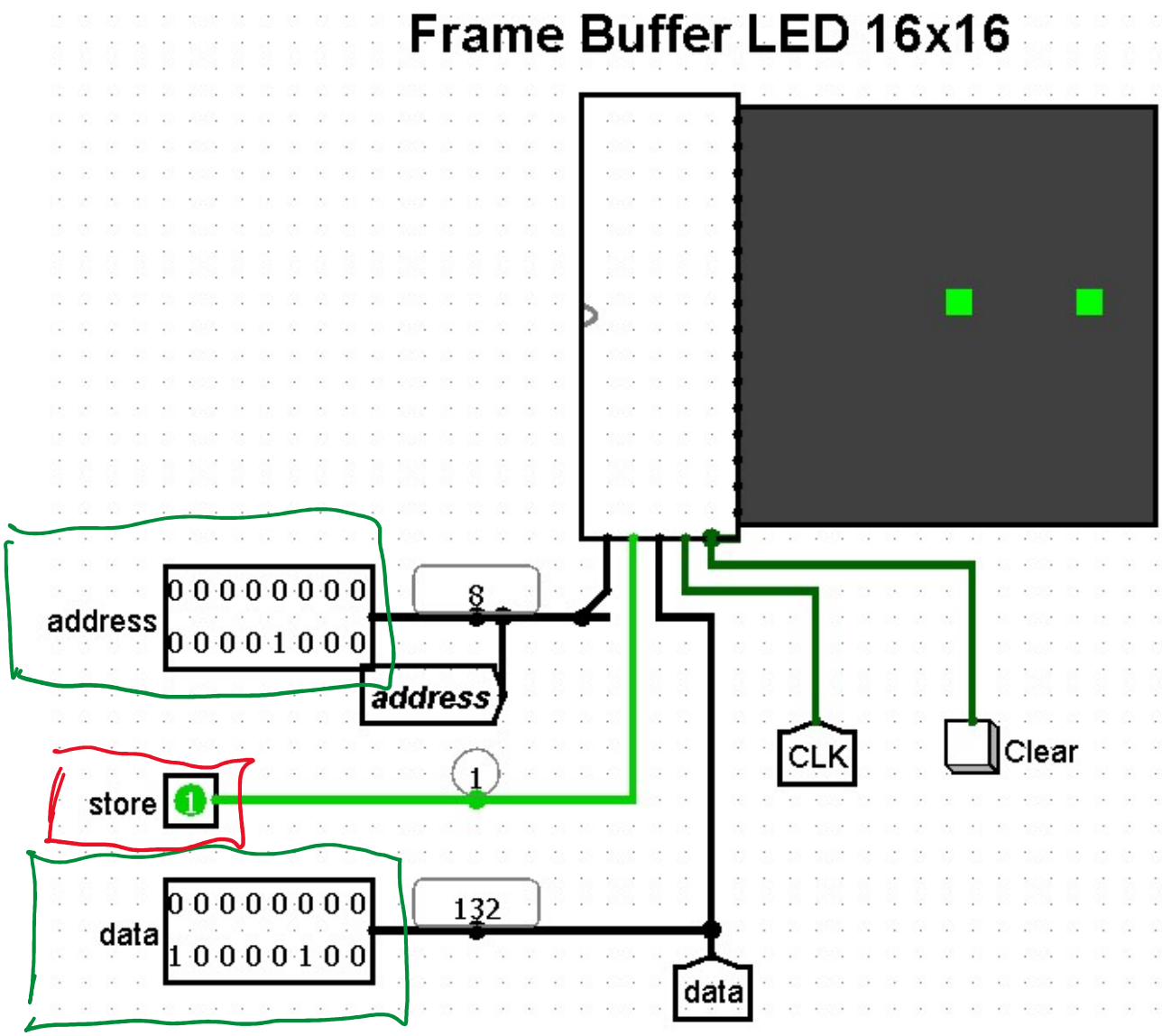
Izbira čipa (CS)

- **Kako priključimo dve (ali več) naprav na vodilo?**
  - Naenkrat mora biti izbran samo en čip (ali nobeden)
  - Za izbiro uporabimo naslednje signale:
    - R/W*, Naslov(A_0-A_{15})
- **Uporabni so biti, ki niso povezani na naslovne signale naprav (A_{15}-A_{13})**
- CSA* in CSB* sta torej funkciji A_{15}-A_{13}



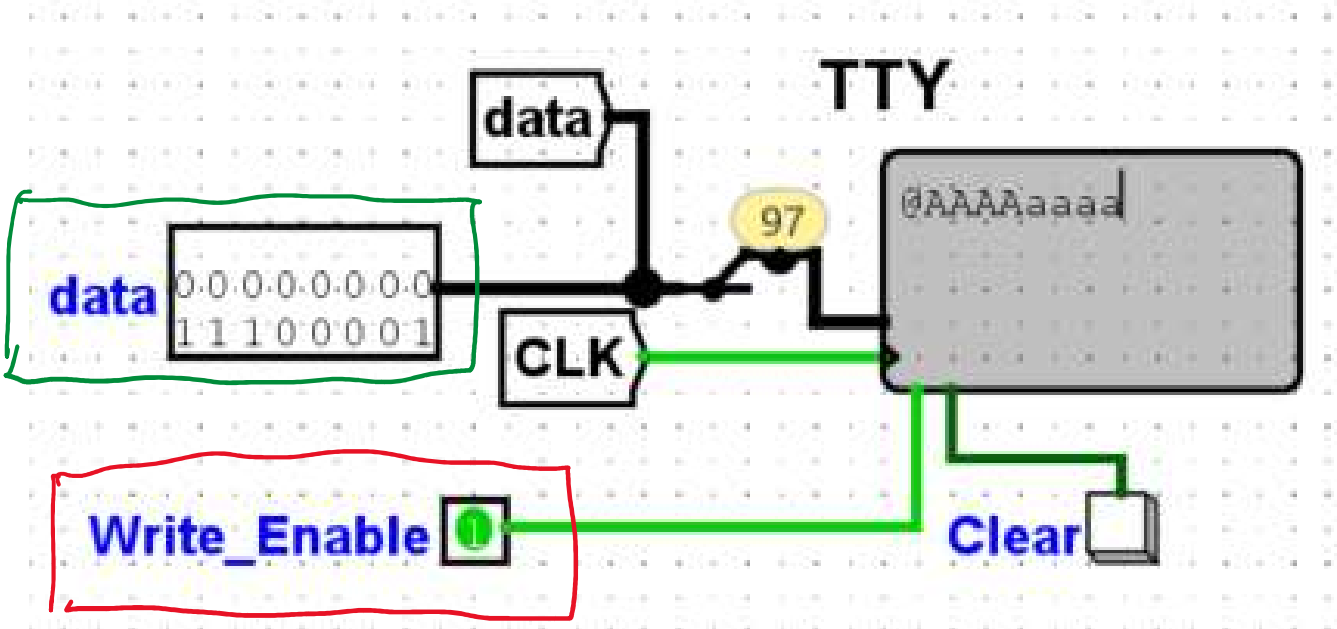
FB - FrameBuffer

ponedeljek, 02. november 2020 17:55



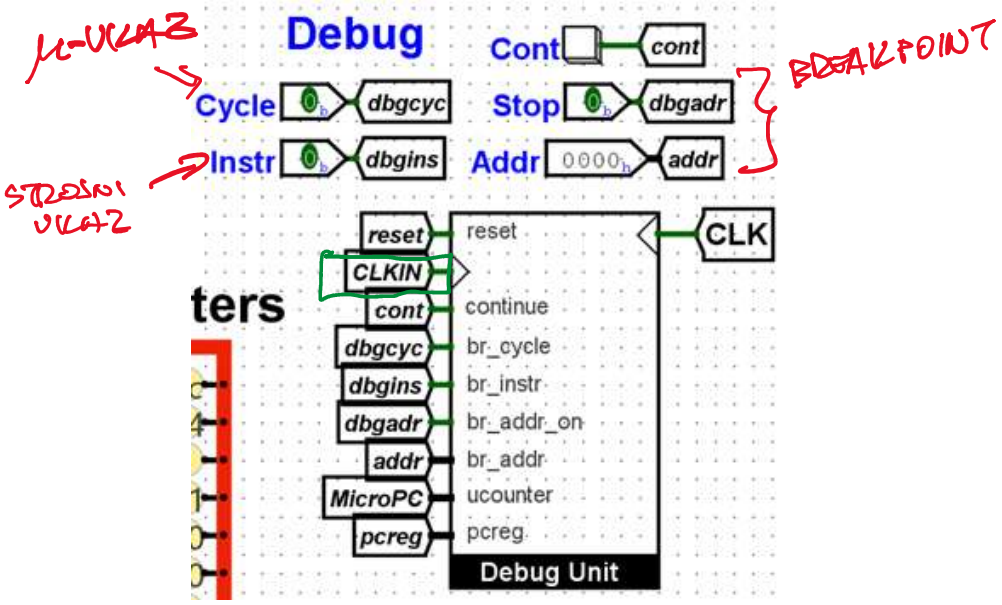
TTY - Serijski terminal

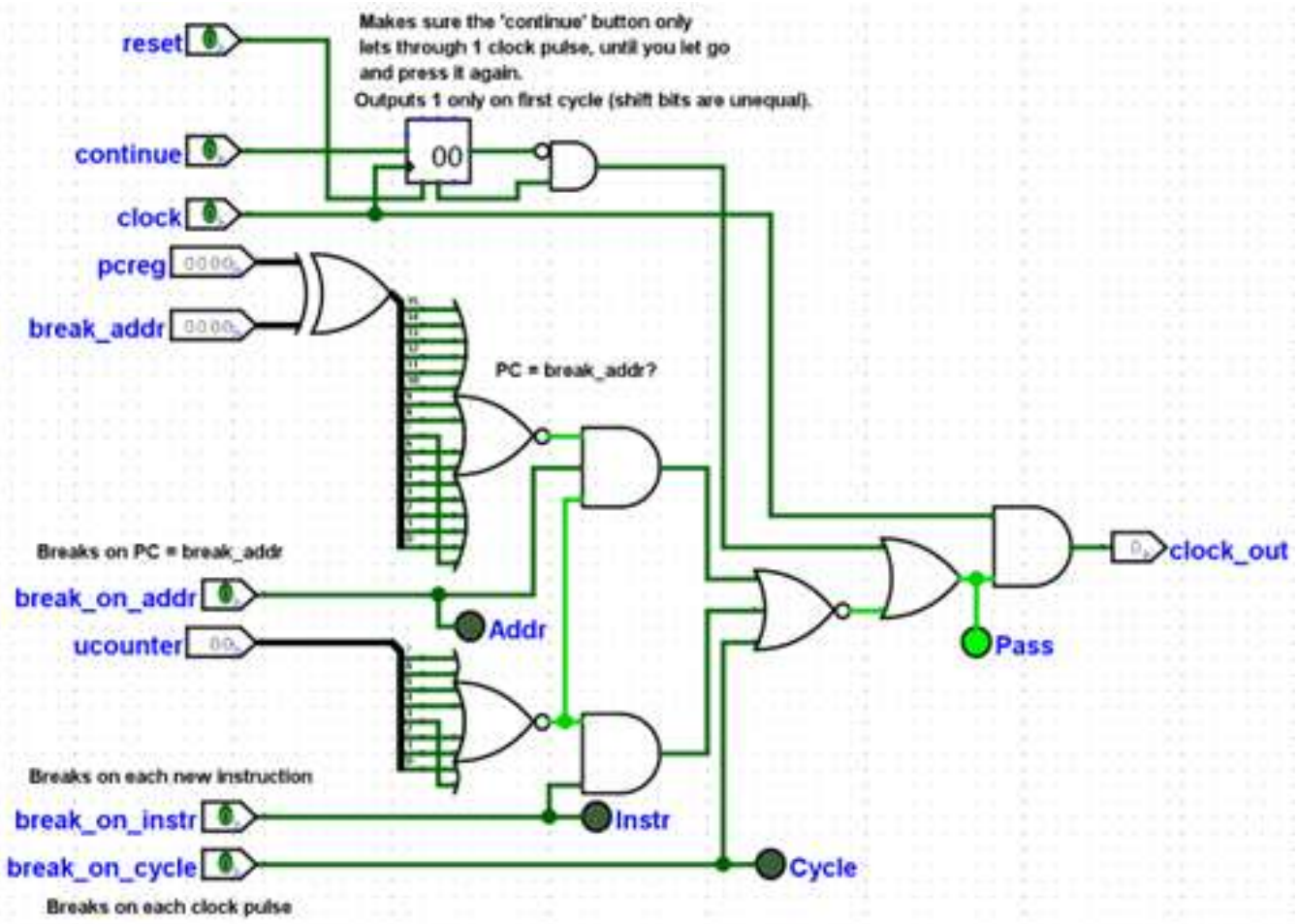
ponedeljek, 02. november 2020 17:56



Debug enota

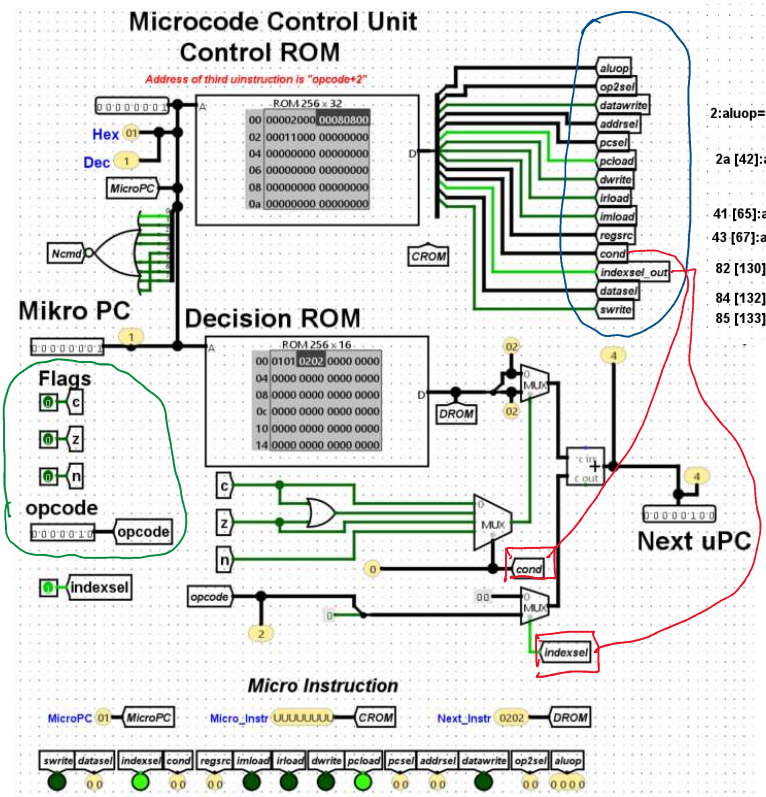
ponedeljek, 24. oktober 2022 17:49




reset
Makes sure the 'continue' button only
lets through 1 clock pulse, until you let go
and press it again.
Outputs 1 only on first cycle (shift bits are unequal).
continue
00
clock
pcreg
break_addr
PC = break_addr?
Breaks on PC = break_addr
break_on_addr
Addr
ucounter
clock_out
Pass
Breaks on each new instruction
break_on_instr
Instr
break_on_cycle
Cycle
Breaks on each clock pulse

3.2.3 Kontrola enota

ponedeljek, 02. november 2020 17:57



Quick tips:

Use ctrl+t to manually toggle global clock signal

Use Simulate->Ticks Enabled for automatic clock signal

```
2:aluop=add  op2sel=treg  dwrite=1  regsrc=aluout, goto fetch  # ALU operations on Rd,Rs,Rt

2a [42]:addrsel=pc  imload=1   # JNEZ Rs,immed

41 [65]:addrsel=pc  dwrite=1  regsrc=databus  # Load immediate Rd, immed
43 [67]:addrsel=pc  imload=1   # Store Rd into address from immed

82 [130]: aluop=sub  op2sel=const0, if z then pcincr else jump

84 [132]:   pcincr: pcload=1  pcsel=pc, goto fetch
85 [133]:   jump:  pcload=1  pcsel=immed, goto fetch
```

KE : SKRBI ZA IZVEDBO UKAZOV

└ VSAK UKAZ IZVEDE PO VSEH ELEMENTARNIH KORAKIH

1 ELEMENTARNI KORAK = 1 μ-UKAZ = 1 TAKT

```
00: 00002000 0101    # fetch: addrsel=pc irload=1
01: 00080800 0202    #        pcload=1  pcsel=pc, opcode_jump
02: 00011000 0000    # 0:     aluop=add  op2sel=treg  dwrite=1  regsrc=aluout, goto fetch
2a: 00004000 8282    # 40:    addrsel=pc  imload=1
41: 00001000 8484    # 63:    addrsel=pc  dwrite=1  regsrc=databus, goto pcincr
43: 00004000 8383    # 65:    addrsel=pc  imload=1
82: 00040021 8485    #        aluop=sub  op2sel=const0, if z then pcincr else jump
83: 001000c0 8484    #        addrsel=immed  datawrite=1  datasel=dreg, goto pcincr
84: 00000800 0000    # pcincr:          pcload=1  pcsel=pc, goto fetch
85: 00000a00 0000    # jump: pcload=1  pcsel=immed, goto fetch
```

SKUPNA ZA VSE UKAZE

3 μ-UKAZE = 3 TAKTE

add Rd,Rs,Rt (0)

Rd <- Rs + Rt PC <- PC + 1

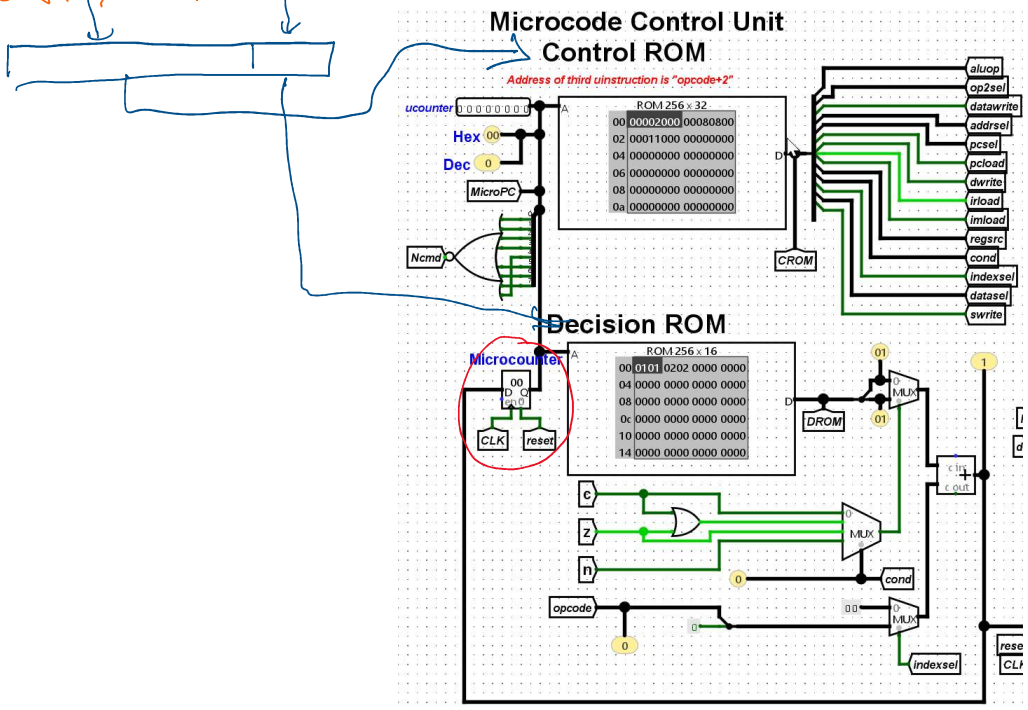
jnez Rs,immed (40)

if Rs != 0, PC <- immed else PC <- PC + 2

5 taktov

MORA DOLOČITI :

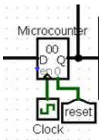
- STANJE KONTR. SIGNALOV
- NASLEDNJI μ-UKAZ



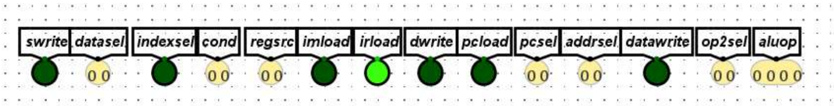
GESTAVA KE :

SESTAVA KE:

- μ-PC → DOLOČA NASLOV μ-UKAZA
  (mikro)



- 2 ROM POMNILNIKA
  - 32-BITNI "CONTROL ROM"
    - → VSEBUJE STANJE KONTR. SIGNALOV



  - 16-BITNI "DECISION ROM"
    - → DOLOČA NASLOV NASL. μ-UKAZA

PRIMERI:

T F

(5,5) ... BREZP. SKOK

(5,4) ...

TRUE: μPC ← 5 FALSE: μPC ← 4

PRIMERA DELOVANJA KE:

① "OPCODE JUMP"

- INDEXSEL = 1 (μPC = 1)
- NASL. NASL. μ-UKAZA = 2 + OP. KODA

② "IF Z THEN PCINCR ELSE JUMP"

NASL. NASL. μ-UKAZA

- COND = Z
- DECISION ROM

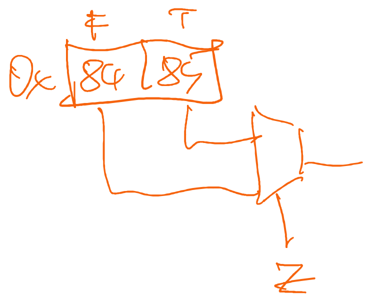


Diagram prehajanja stanj

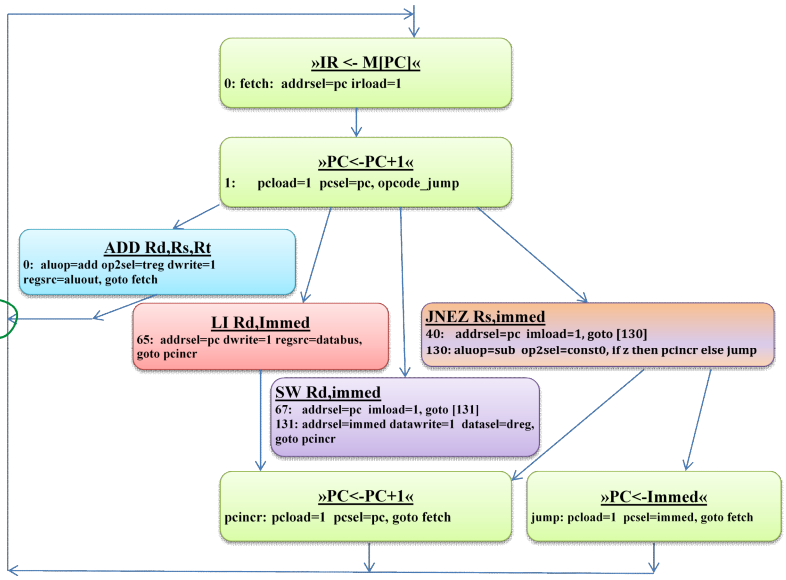
ponedeljek, 09. november 2020 20:59

OBIČAJEN KONCEPT OPISOVANJA KÉ

RAZVIDNA IZVEDBA VVODOV

OSNOVA ZA OPTIMIZACIJO

NOV STROJNI UKAZ



. Mikroukaz : 63: addrsel=pc dwrite=1 regsrc=databus, goto pcincr)

Op.koda	kontrolni signali	naslednji mikroukaz

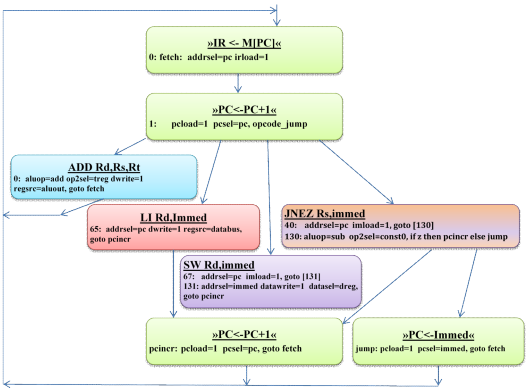
kontr. signal	opisna vrednost	enota
aluop	add, sub, mul, div, rem, and, or, xor, nand, nor, not, lsl, lsr, asr, rol, ror	ALE
op2sel	treg, immed, const0, const1	ALE
addrsel	pc, immed, aluout, sreg	nasl. vodilo
pcsel	pc, immed, pcimmed, sreg	PC
regsrc	databus, immed, aluout, sreg	registri
cond	c, corz, z, n	kontr. enota

datoteka **basic_microcode.def**

```
fetch:  addrsel=pc irload=1                  # Address=PC, Load IR register
        pcload=1  pcsel=pc, opcode_jump      # PC=PC+1, jump to 2+OPC

# ALU operation '+' on Rd,Rs,Rt
0:      aluop=add  op2sel=treg  dwrite=1  regsrc=aluout, goto fetch

# JNEZ Rs,immed
40:     addrsel=pc  imload=1
        aluop=sub  op2sel=const0, if z then pcincr else jump

# li Rd,Immed
63:     addrsel=pc  dwrite=1  regsrc=databus, goto pcincr

# Rd->M[immed]
65:     addrsel=pc  imload=1
        addrsel=immed  datawrite=1  datasel=dreg, goto pcincr

pcincr:  pcload=1  pcsel=pc, goto fetch

jump:    pcload=1  pcsel=immed, goto fetch
```

. RAZPORED µUKAZOV V ROM POMNILNIKIH

Naslov	Vsebina
0	BRANJE UKAZA
1	PC=PC+1, OPCODE_JUMP
2-129	OP.KODE 0 : 127 PRVI µUKAZI ZA VSAK STROJNI UKAZ (128)
130+ 0x82+	VSI OSTALI µ-UKAZI

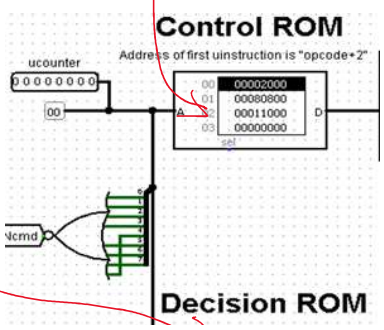


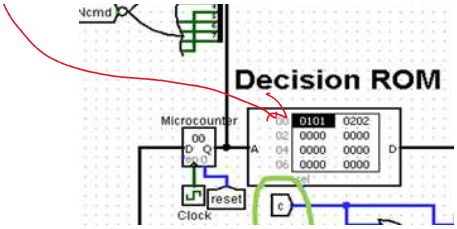
µ-PROGRAM & INI.DEF

MICRO_ASSEMBLER.EXE .PL

→ UCONTROL.ROM

→ UDECISION.ROM





IZVORNA KODA

datoteka **basic_microcode.def**

```
fetch:   addrsel=pc irload=1                    # Address=PC, Load IR register
         pcload=1  pcsel=pc, opcode_jump        # PC=PC+1, jump to 2+OPC

# ALU operation '+' on Rd,Rs,Rt
0:       aluop=add  op2sel=treg  dwrite=1  regsrc=aluout, goto fetch

# JNEZ Rs,immed
40:      addrsel=pc  imload=1
         aluop=sub  op2sel=const0, if z then pcincr else jump

# li Rd,Immed
63:      addrsel=pc  dwrite=1  regsrc=databus, goto pcincr

# Rd->M[immed]
65:      addrsel=pc  imload=1
         addrsel=immed  datawrite=1  datasel=dreg, goto pcincr

pcincr:  pcload=1  pcsel=pc, goto fetch

jump:    pcload=1  pcsel=immed, goto fetch
```

POSEBNO

se prevede v

indexsel, pcload

```
00: 00002000 0101    # fetch: addrsel=pc irload=1
01: 00080800 0202    #          pcload=1  pcsel=pc, opcode_jump
02: 00011000 0000    # 0:       aluop=add  op2sel=treg  dwrite=1  regsrc=aluout,    goto fetch
2a: 00004000 8282    # 40:      addrsel=pc  imload=1                                 (goto 82)
41: 00001000 8484    # 63:      addrsel=pc  dwrite=1  regsrc=databus,                goto pcincr
43: 00004000 8383    # 65:      addrsel=pc  imload=1                                 (goto 83)
82: 00040021 8485    #            aluop=sub  op2sel=const0,                          if z then pcincr else jump
83: 001000c0 8484    #          addrsel=immed  datawrite=1  datasel=dreg,            goto pcincr
84: 00000800 0000    # pcincr:        pcload=1  pcsel=pc,                            goto fetch
85: 00000a00 0000    # jump:    pcload=1  pcsel=immed,                               goto fetch
```

POSK MIKROUK

OSTALI

μ-NASLOV

CONTROL ROM

DECISION ROM

- **Primer (testni):**

```
main:   li    r0, 0            # r0 is the running sum
        li    r1, 100          # r1 is the counter
        li    r2, -1           # Used to decrement r1
loop:   add   r0, r0, r1       # r0= r0 + r1
        add   r1, r1, r2       # r1--
        jnez  r1, loop         # loop if r1 != 0
        sw    r0, 256          # Save the result
inf:    jnez  r2, inf          # loop if r1 != 0 -> loop forever
```

PROGRAM

POROČILO:

```
0000: 00007e00  0111111000000000        main:   li      r0, 0
0001: 00000000  0000000000000000
0002: 00007e01  0111111000000001                li      r1, 100
0003: 00000064  0000000001100100
0004: 00007e02  0111111000000010                li      r2, -1
0005: 0000ffff  1111111111111111
0006: 00000040  0000000001000000        loop:   add     r0, r0, r1
0007: 00000089  0000000010001001                add     r1, r1, r2
0008: 00005008  0101000000001000                jnez    r1, loop
0009: 00000006  0000000000000110
000a: 00008200  1000001000000000                sw      r0, 256
000b: 00000100  0000000100000000
000c: 00005010  0101000000010000        inf:    jnez    r2, inf
000d: 0000000c  0000000000001100
```

2 UKAZA TAKOJ.

NASLOV RAM

Op.koda	Treg	Sreg	Dreg
7	3	3	3

ASSEMBLER.EXE .PL

IME.S → IME.RAM

Zbirnik – primeri ukazov

list_of_instructions.txt (distribucija) :

rdeče: trenutno že implementirani ukazi v modelu MiMo.

- add Rd,Rs,Rt (0) — Rd <- Rs + Rt, PC <- PC + 1
- sub Rd,Rs,Rt (1) — Rd <- Rs – Rt, PC <- PC + 1
- …
- jeqz Rs,immed (39) — if Rs == 0, PC <- immed else PC <- PC + 2
- jnez Rs,immed (40) — if Rs != 0, PC <- immed else PC <- PC + 2
- …
- beq Rs,Rt,immed (46) — if Rs == Rt, PC <- PC + immed else PC <- PC + 2
- bne Rs,Rt,immed (47) — if Rs != Rt, PC <- PC + immed else PC <- PC + 2
- …
- li Rd,immed (63) — Rd <- immed, PC <- PC + 2
- sw Rd,immed (65) — M[immed] <- Rd, PC <- PC + 2
- …
- lw Rd,immed (64) — Rd <- M[immed], PC <- PC + 2
- lwi Rd,Rs,immed (66) — Rd <- M[Rs+immed], PC <- PC + 2
- swi Rd,Rs,immed (67) — M[Rs+immed] <- Rd, PC <- PC + 2

- swi Rd,Rs,immed (67) M[Rs+immed] <- Rd, PC <- PC + 2

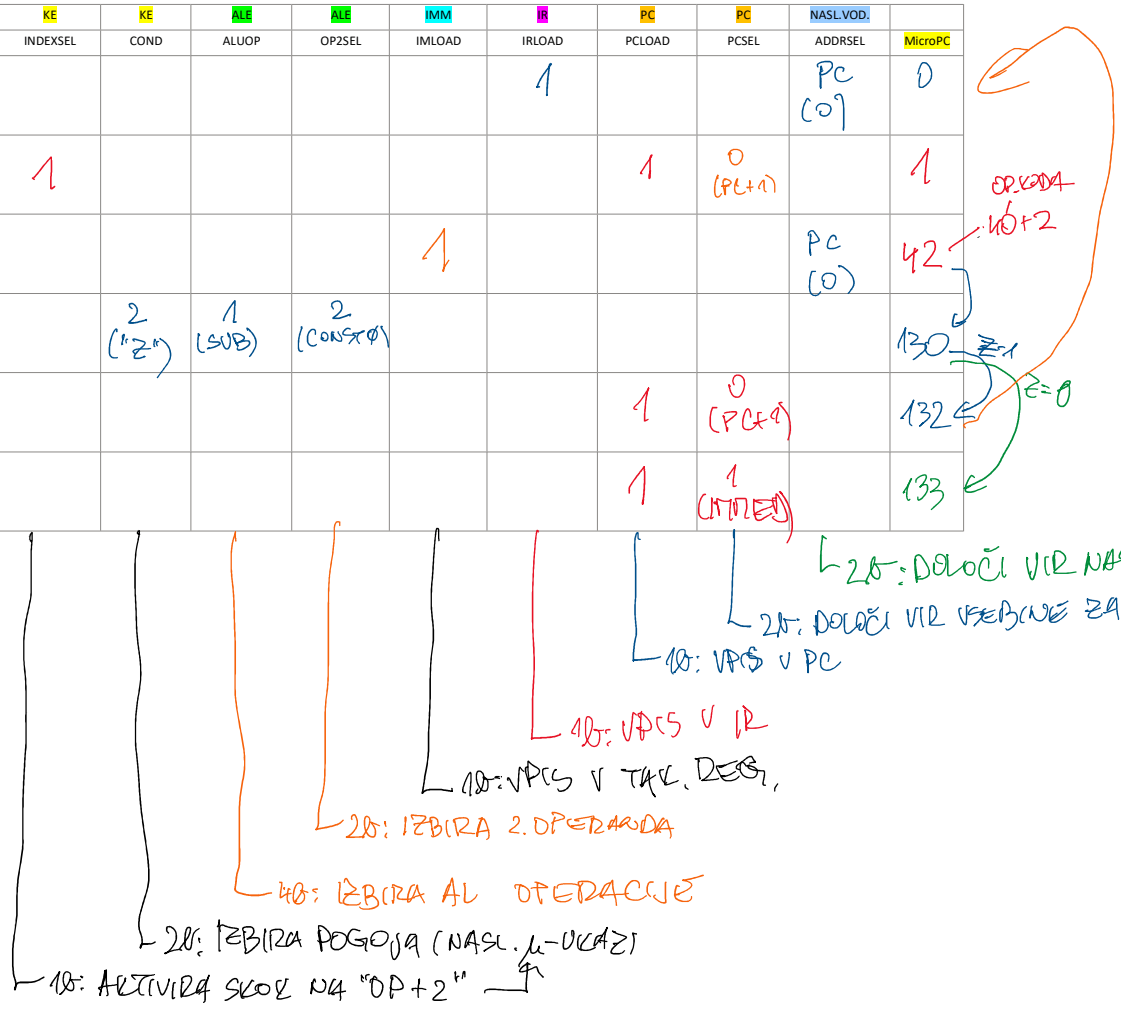
Izvedba JNEZ - drugič

ponedeljek, 15. november 2021 22:55

Collaboration space :

Predavanja-skupno delo

Enota ?	KE	KE	ALE	ALE	IMM	IR	PC	PC	NASL.VOD.	
JNEZ Rs, IMMED	INDEXSEL	COND	ALUOP	OP2SEL	IMLOAD	IRLOAD	PCLOAD	PCSEL	ADDRSEL	MicroPC
# Address=PC, Load IR register						1			PC (0)	0
# PC=PC+1, jump to 2+OP	1						1	0 (PC+1)		1
# Read Immediate operand -> IMRegister					1				PC (0)	42
# ALU: Rs-0, If z then pcincr else jump		2 ("Z")	1 (SUB)	2 (CONST0)						130
# Increment PC and goto new command;							1	0 (PC+1)		132
# Set address to immed and goto new command							1	1 (IMMED)		133



3.2.6 Mikroprogramirana vs trdoožičena CPE

ponedeljek, 09. november 2020 22:01

Izhodišča:

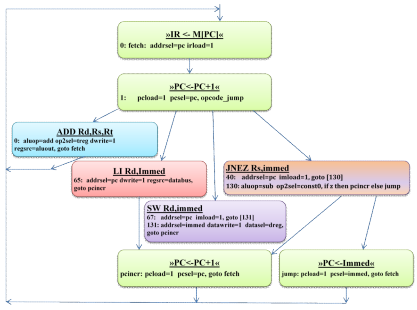
- **Diagram poteka**
- **Elementarni koraki**

ZAPOMNITI!

μ-PROGR.:

+FLEKSIBILNOST

+ENOSTAVNA



TRDOŽIČENA ("HARD WIRED")

+HITROST

-KOMPLEKSNOST

Elementarni koraki:

0 .. Branje ukaza -> IR

1 .. Format 2: branje operanda

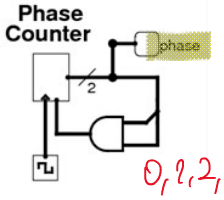
Format 1: nop

2 .. Vse operacije :

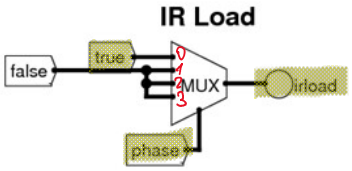
- ALE, skok, reg.write, R/W from Mem

} FETCH

} EXECUTE

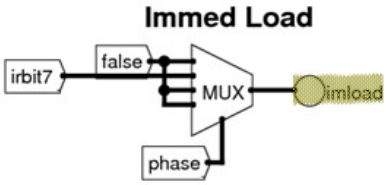


0,1,2,0,1,2,0,1,2...



1... EL.K.=0

0.. SICER



	EL.K.
0	0
b_7	1
0	2

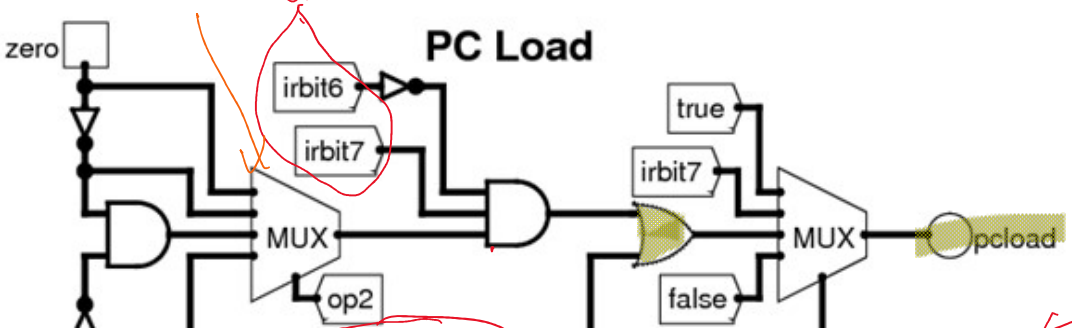
irbit7:

0 .. 8-bitni ukaz (1bajt)

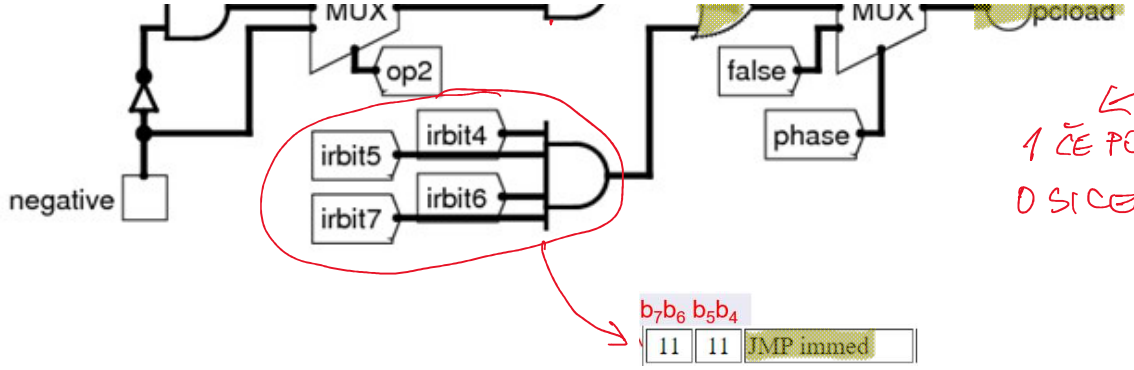
1 .. 16-bitni ukaz (2 bajta)

op1 op2 b_7b_6	b_5b_4	
10	00	JEQ Rd, immed
10	01	JNE Rd, immed
10	10	JGT Rd, immed
10	11	JLT Rd, immed

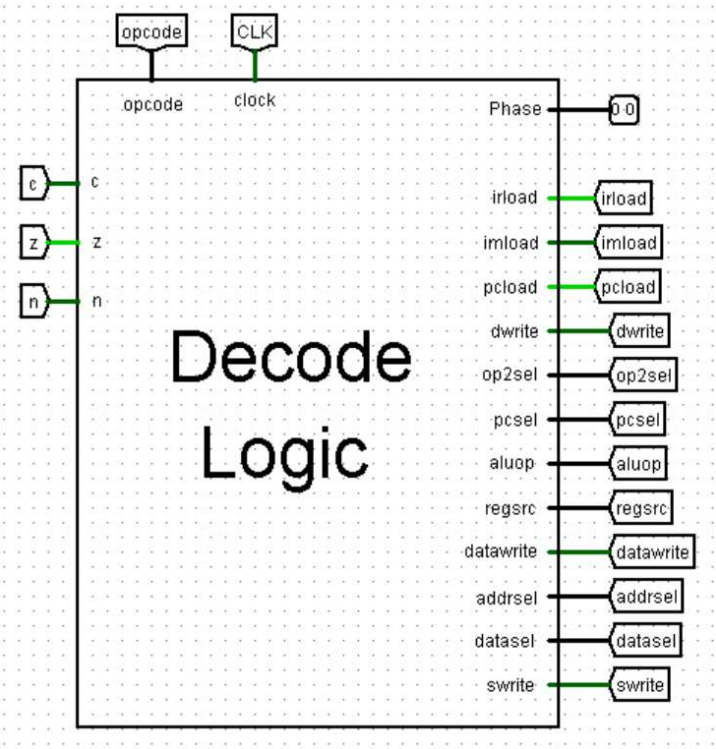
POGOJI

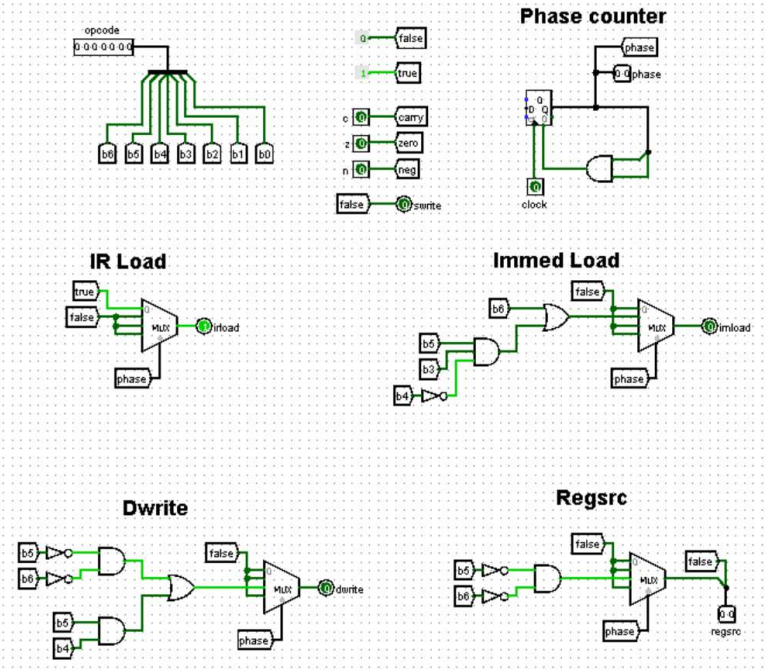


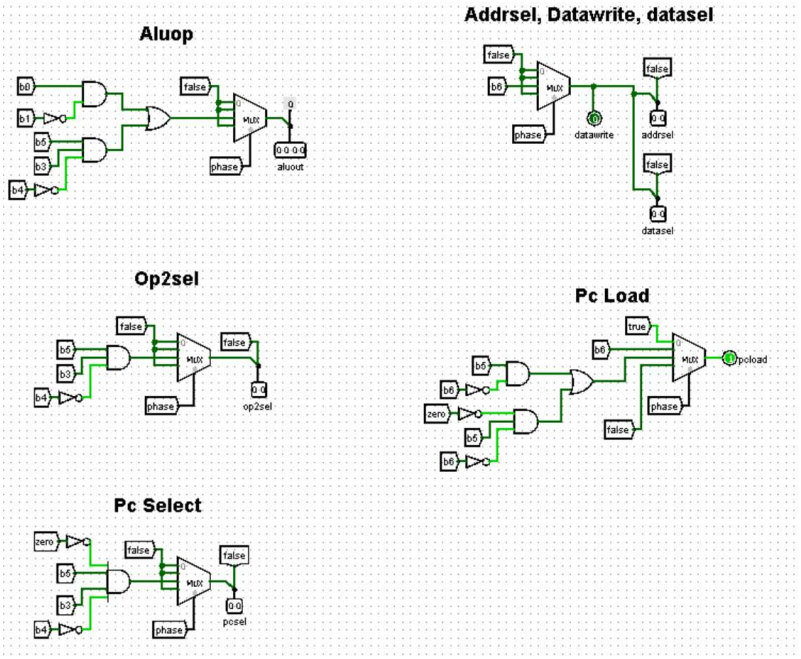
EL.K.	
1	0
b_7	1
	2



Podrobnejši opis - dodatno gradivo:
http://minnie.tuhs.org/CompArch/Tutes/week03.html






Aluop
b0
b1
b5
b3
b4
false
Mux
phase
0 0 0 0
aluout
Addrsel, Datawrite, datasel
false
b6
phase
Mux
datawrite
false
0 0
addrsel
false
0 0
datasel
Op2sel
false
b6
b3
b4
phase
Mux
false
0 0
op2sel
Pc Load
true
b6
b5
b6
zero
b6
b6
Mux
phase
false
pcload
Pc Select
zero
b6
b3
b4
false
Mux
phase
false
0 0
posel