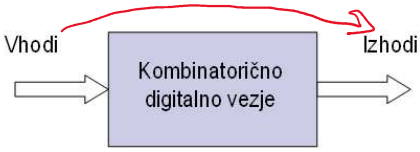
3. Mikroarhitektura računalnika ≡ ORGANIZACIJA

ponedeljek, 02. november 2020 17:29

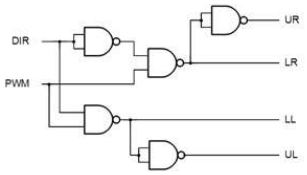
Logična vezja

ponedeljek, 02. november 2020 17:30

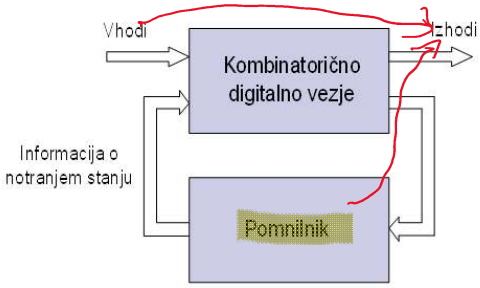
Kombinatorična logična vezja: (LOGIČNA VRATA)



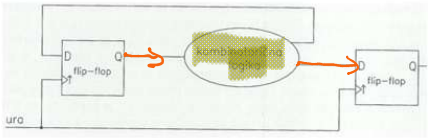
NPR: SEŠTEVALNIK



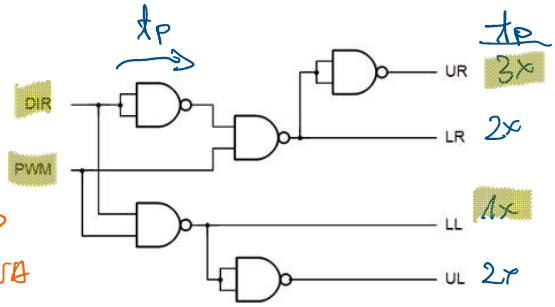
Sekvenčna logična vezja:



NPR: FF, REGISTRI, ŠTEVCI, POMNILNIKI



Asinhronska digitalna vezja :



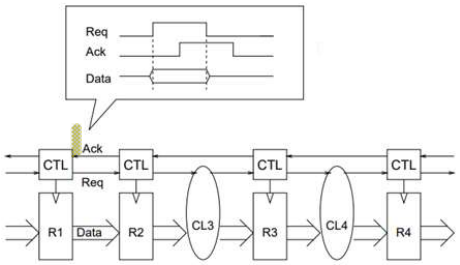
LOČENA OBRAVNAVA POSAMEZNIH VEJ

SINHRONSKI PRISTOP

$t_{CPE} \geq 3t_p$

PRILAGODIMO SE NAJPOČASNEJŠI VEJI

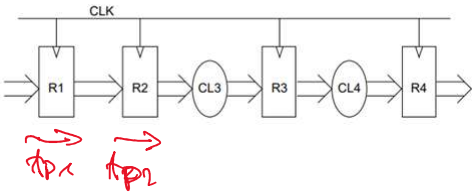
ASINHRONSKI



+ HITROST
+ PORABA
− KOMPLEKSNOST (RAZL. ZAKASNITVE)

SINHRONSKI

$t_{CPE} \geq MAX(t_{p1}, t_{p2}, \ldots)$



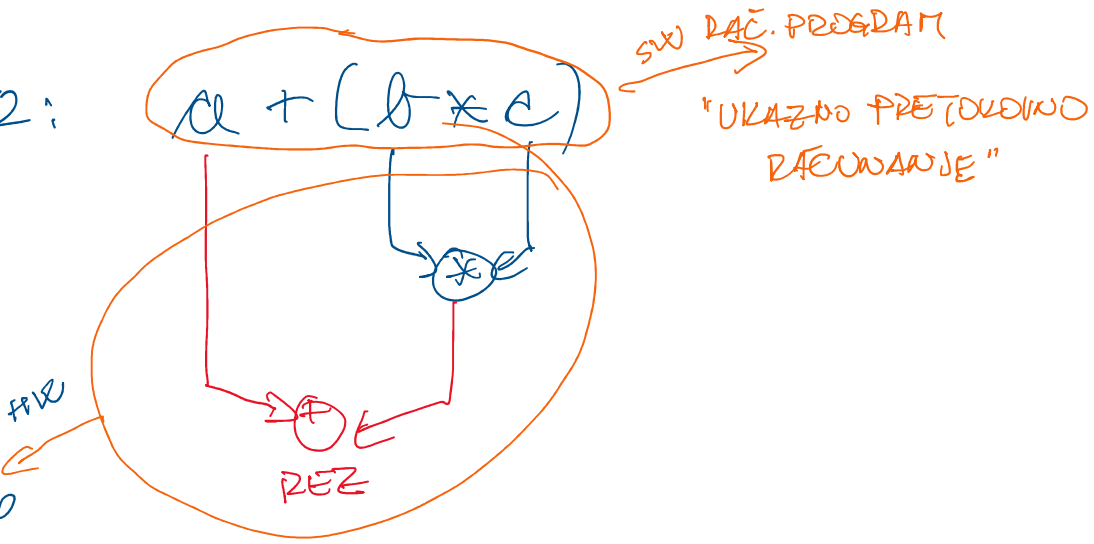
t_{p1} t_{p2}

PRIMER: MIPS 3000

− ASINHRONSKA IZVEDBA
+ 4x HITREJŠA

ARM → AMULET (ASINHRONSKI "ARM")

PRIMER:



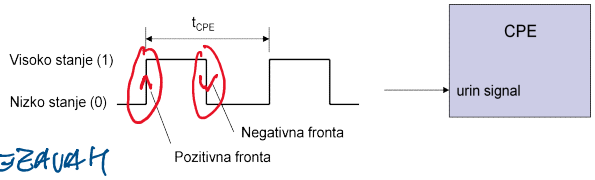
MAXELER

3.1 Sinhronska digitalna vezja

ponedeljek, 02. november 2020 17:34

3.1.1. URIN SIGNAL

t_{CPE} ODVISNA OD:
- HITROST VEZIJ
- ŠT. VEZIJ
- ZAKASNITVE V POVEZAVAH



3.1.2 SPREMEMBA STANJA

① t_S ... "SETUP" (VZPOSTAVITVENI ČAS)

② t_H ... "HOLD" (DRŽALNI ČAS)

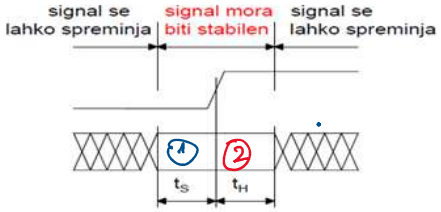
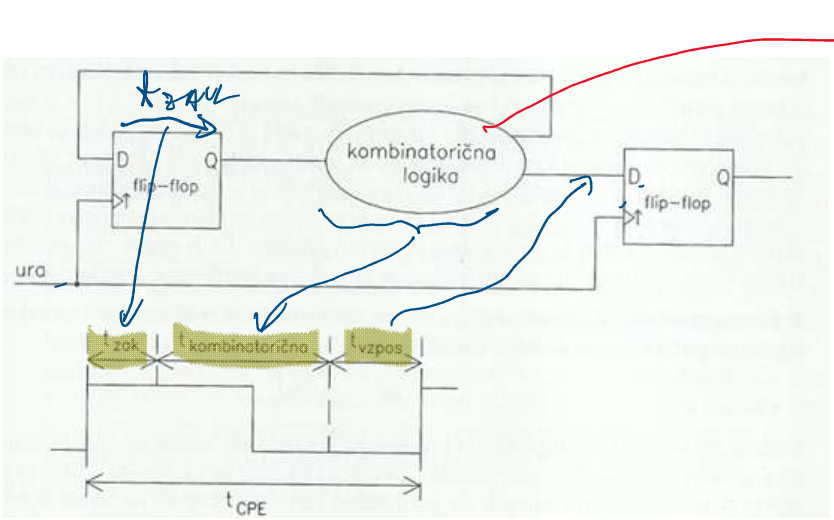
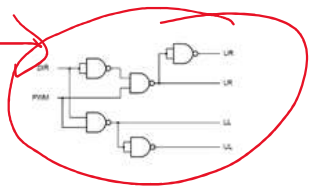


Figure 1.27: Vzpostavitveni (t_S) in držalni čas (t_H).





$t_{CPE} \geq t_{ZAK} + t_{KOMB} + t_{VZP}$

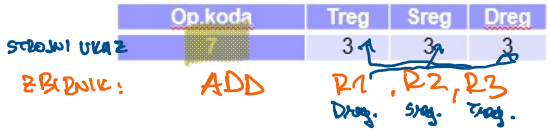
3.2 MiMo - model CPE

ponedeljek, 02. november 2020 17:37

Značilnosti :

- pomnilniška beseda 16-BITNA
- 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: (FORMAT 1 + 16B TAK. OPERAND)

- 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 PREVZETNI CIKEL
- "EXECUTE": IZVRŠILNI CIKEL

"ELEMENTARNI KORAKI":
- BRANJE UKAZA
- DEKODIRANJE (ANALIZA) UKAZA
- BRANJE OPERANDOV
- IZVEDBA OPERACIJE
- SHRANITEV REZULTATA

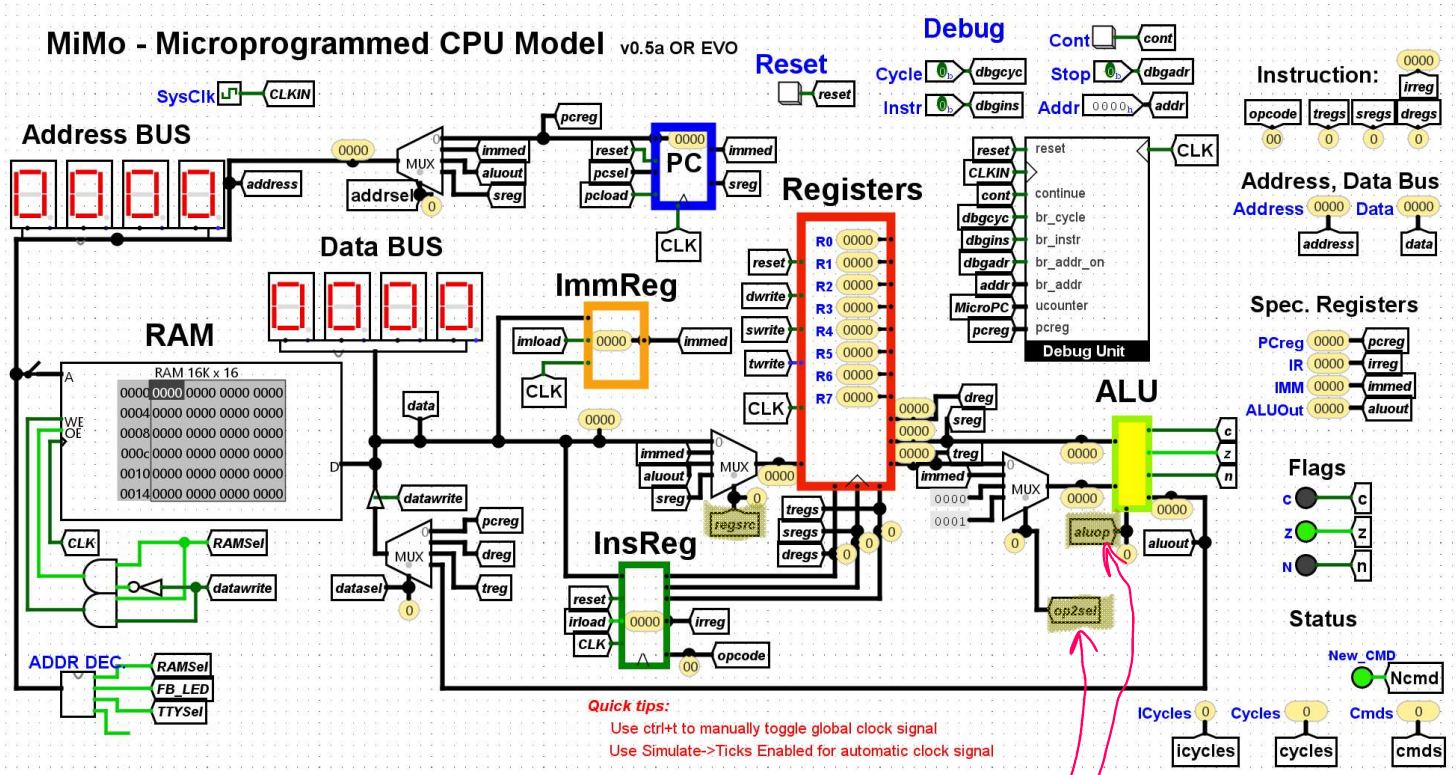
PC=PC+1

1 UKAZ ⇉ ELEMENTARNI KORAKI

VSAK KORAK ≡ μ-UKAZ ⇔ STANJE KONTROLNIH SIGNALOV V CPE

3.2.2 Podatkovna enota

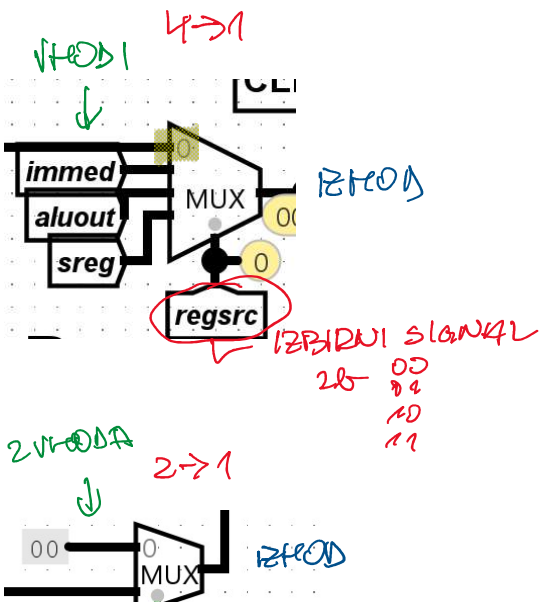
ponedeljek, 02. november 2020 17:42

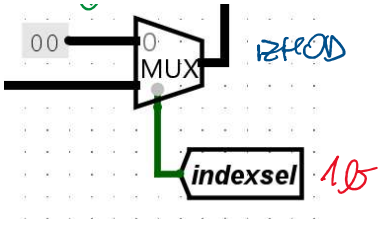


KONTROLNI SIGNALI

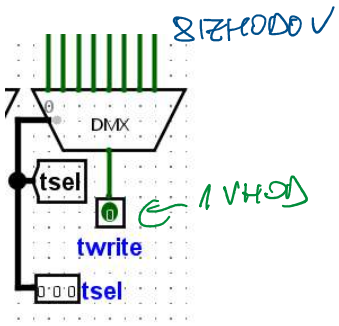
KE

MUX - MULTIPLEKSER

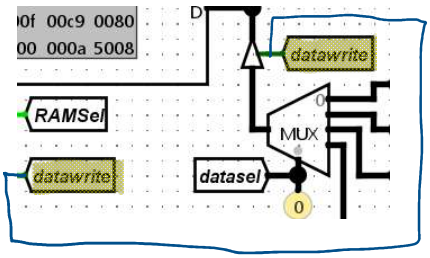




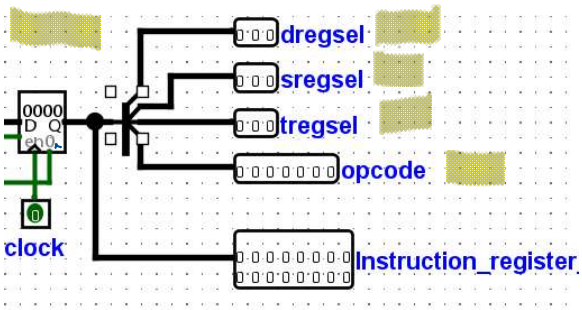
DEMUX – DEMULTIPLEKSER



TUNELI (POVEZAVE)

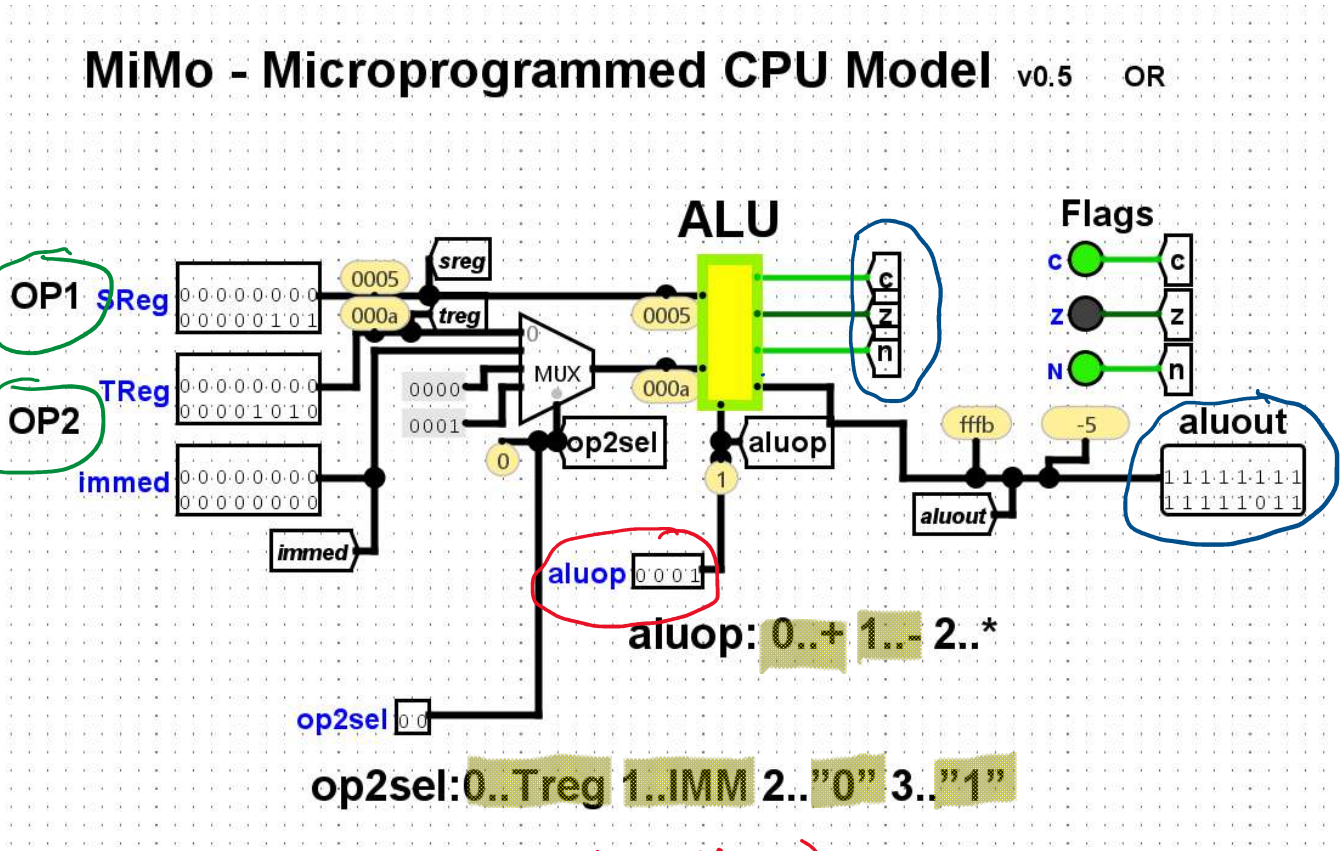


SPLITTER / DELILNIK



ALE

ponedeljek, 02. november 2020 17:45



ALUOP: VRSTA OPERACIJE (+, −, *, —)

OP2SEL: DOLOČA 2. OPERAND

add Rd,Rs,Rt (0)

Rd <- Rs + Rt

PC <- PC + 1

ALUOP = 0 ("+")

OP2SEL = 0 (2.OP = R_T)

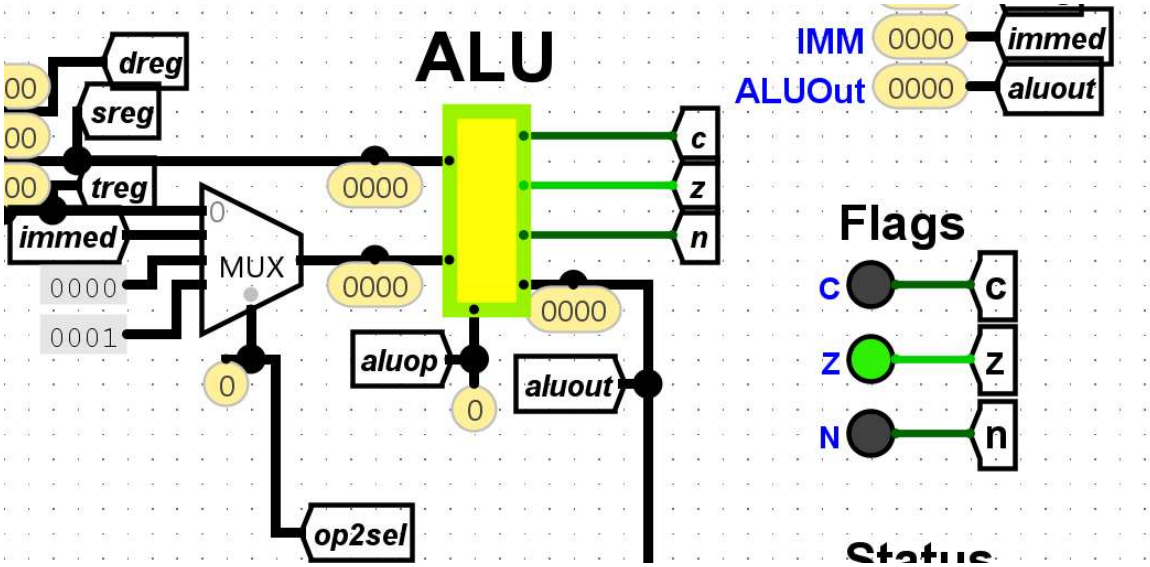
jnez Rs,immed (40)

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

ALE: $R_S - 0$

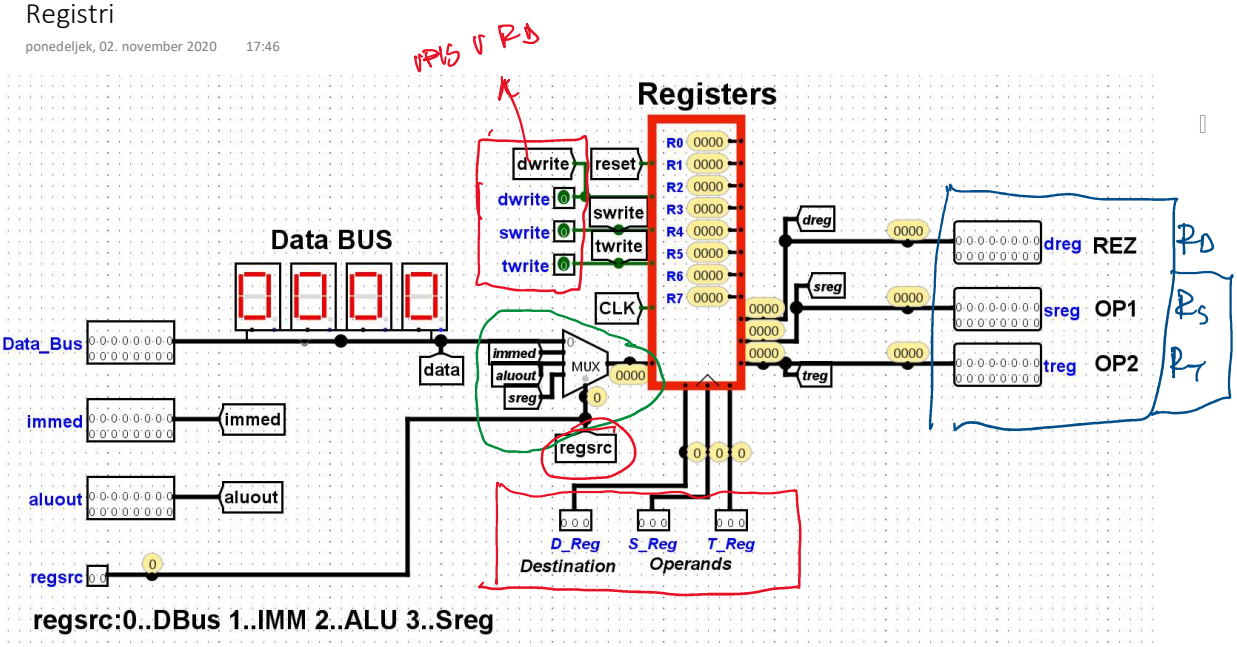
ALUOP = 1 ("-")

OP2SEL = 2 ("0")


ALU
dreg
sreg
treg
immed
MUX
0000
0001
0
0000
0000
0000
aluop
aluout
op2sel
c
z
n
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)
Rd <- Rs + Rt ALE
PC <- PC + 1

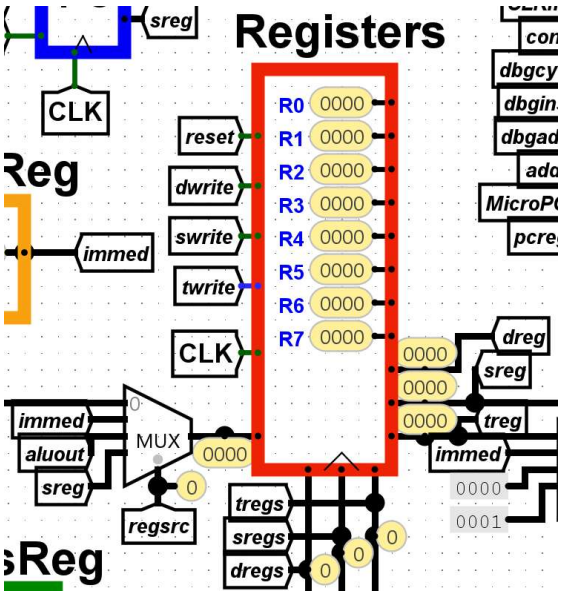
REGSRC = 2 ("ALUOUT")
DWRITE = 1
VPIS: ALUOUT → Rd

move Rd,Rs (70)
Rd <- Rs
PC <- PC + 1

REGSRC = 3 (Rs)
DWRITE = 1
VPIS: Rs → Rd

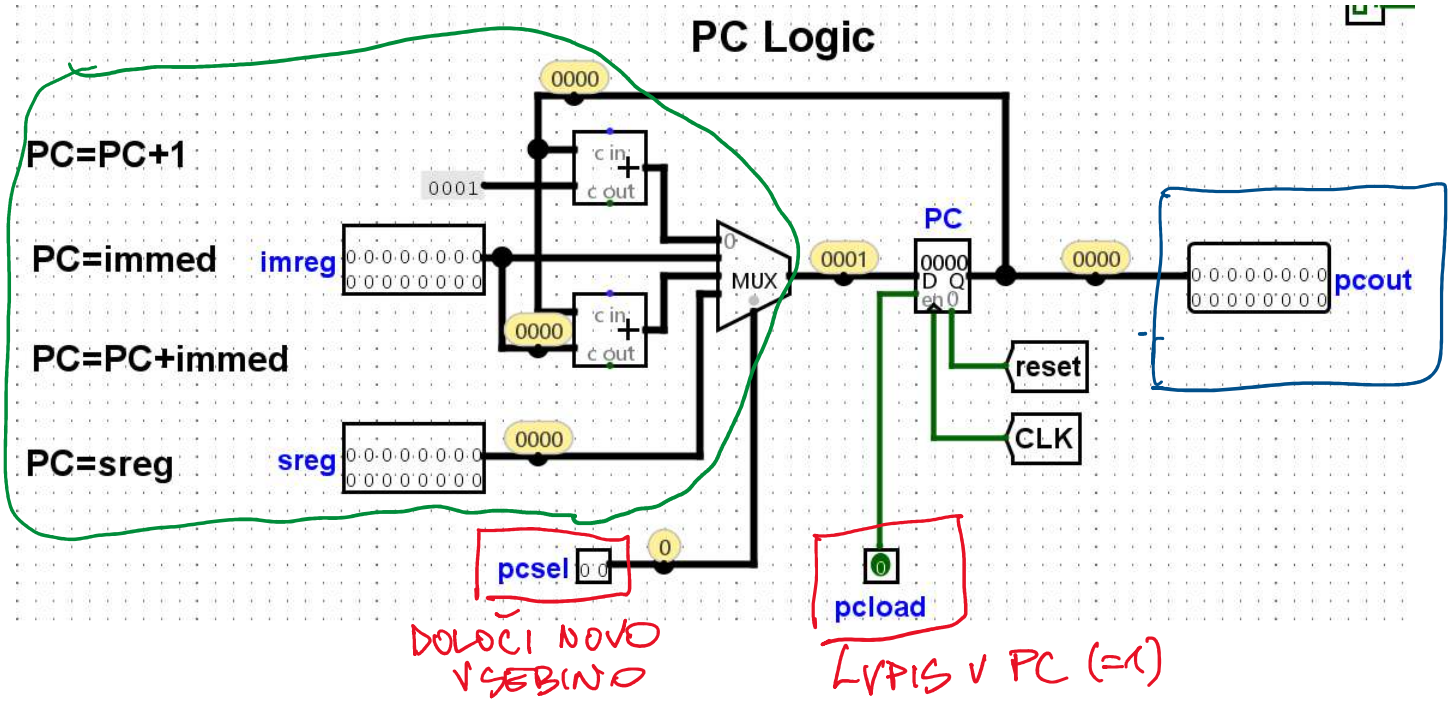
li Rd,immed (63)
Rd <- immed
PC <- PC + 2

REGSRC = Ø (DATABUS)
DWRITE = 1
VPIS: DATABUS → RD (IMM-OP)



PC - programski števec

ponedeljek, 02. november 2020 17:49



DOLOČI NOVO VSEBINO

VPIS V PC (=1)

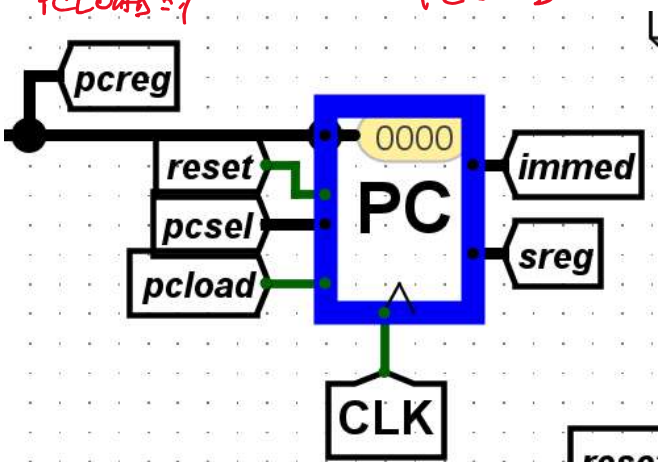
VHODI:

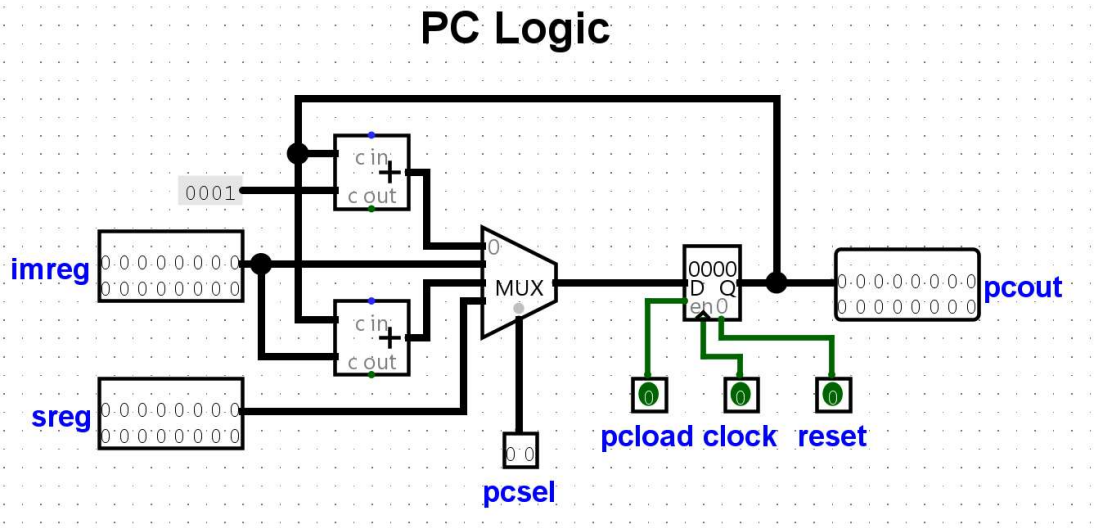
- PC+1
- IMMED (ABS. SKOK)
- PC+IMMED (REL. SKOK)
- SREG ("MOV PC,R")

PC ← PC+1
PCSEL = 0 (PC+1)
PCLOAD = 1

ABS. SKOK
PCSEL = 1 (IMM)
PCLOAD = 1

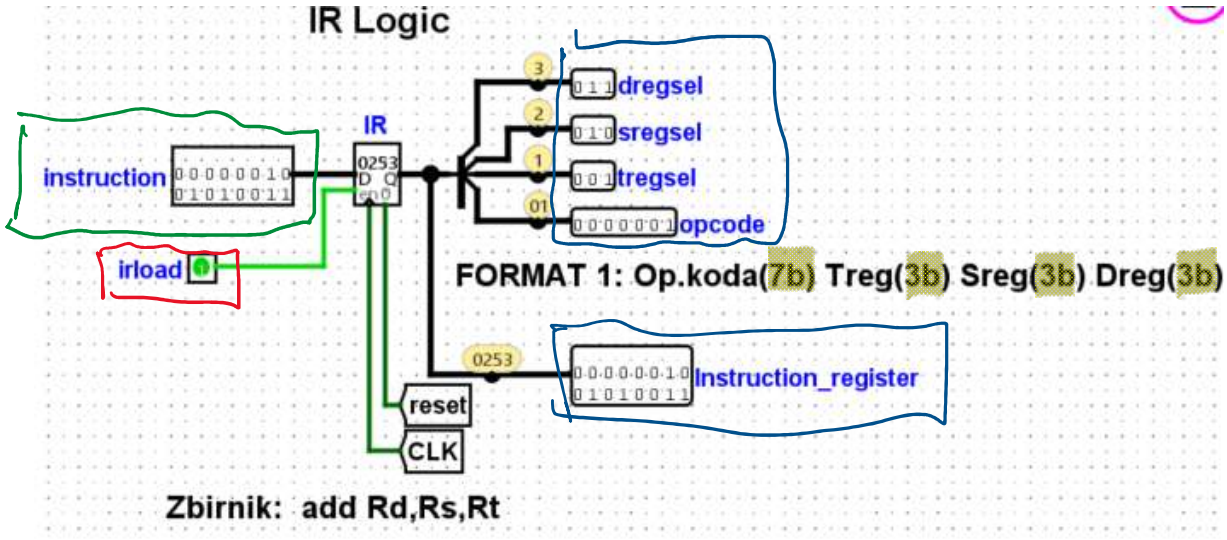
REL. SKOK
PCSEL = 2 (PC+IMM)
PCLOAD = 1




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



sub Rd,Rs,Rt (1)

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

add Rd,Rs,Rt (0)

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

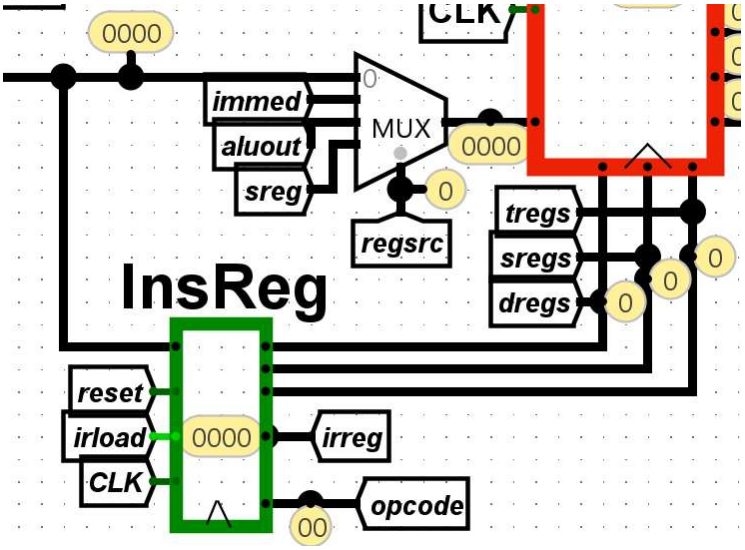
SUB R3,R2,R1

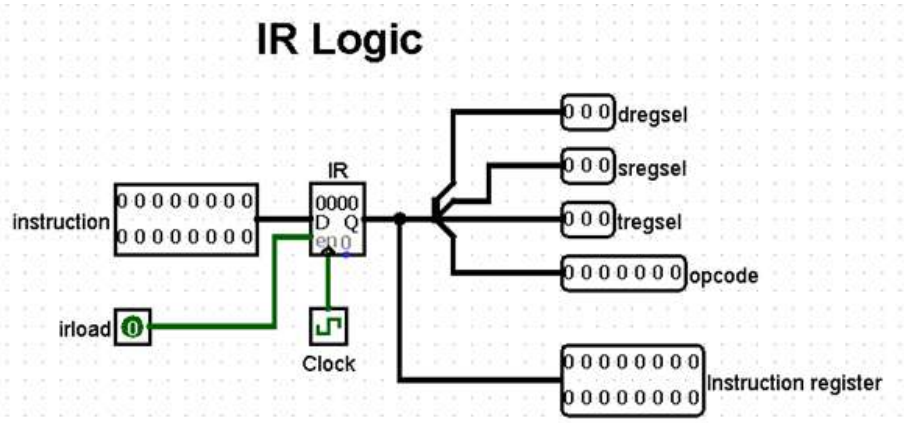
Op.koda	Treg	Sreg	Dreg
7	3	3	3

FETCH (BRANJE UKAZA):

IRLOAD=1

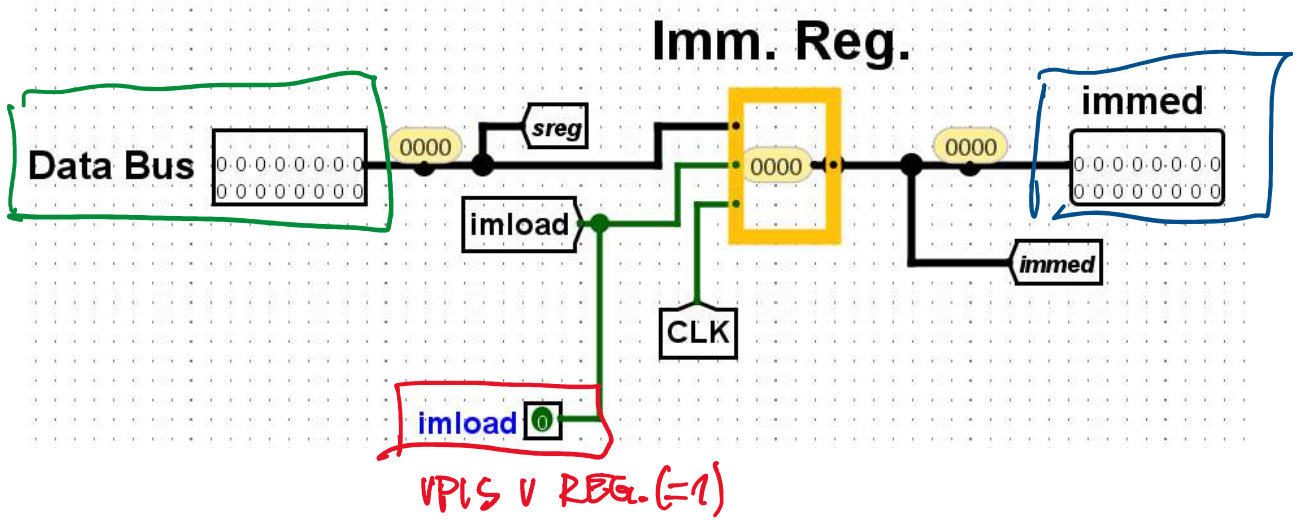
(ADDRSEL=PC)


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 R1, LOOP

jnez Rs,immed (40)

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

BRANJE TAK. OP.: IMLOAD = 1

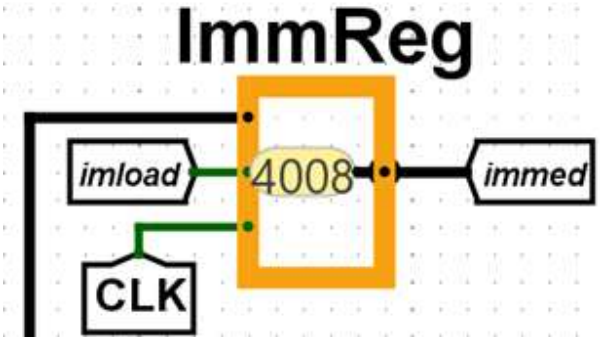
li Rd,immed (63)

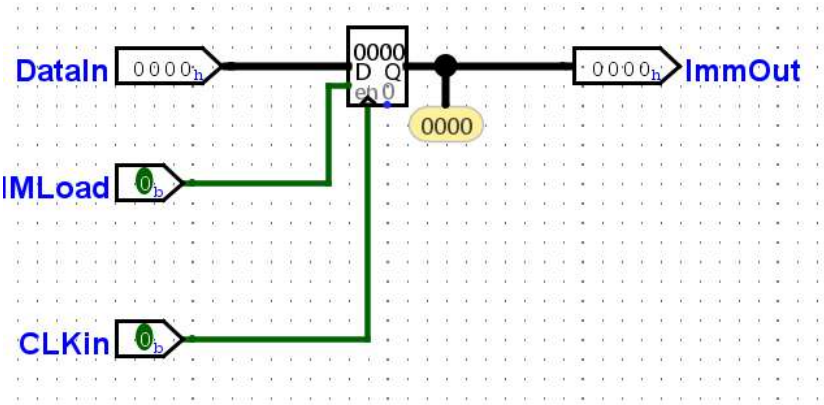
Rd <- immed PC <- PC + 2

IMLOAD = 0 (BUDITE = 1) } VPISEMO DIREKTNO V REGISTER Rd

Format 2 : (primer LI R1, 100 # R1<-100)

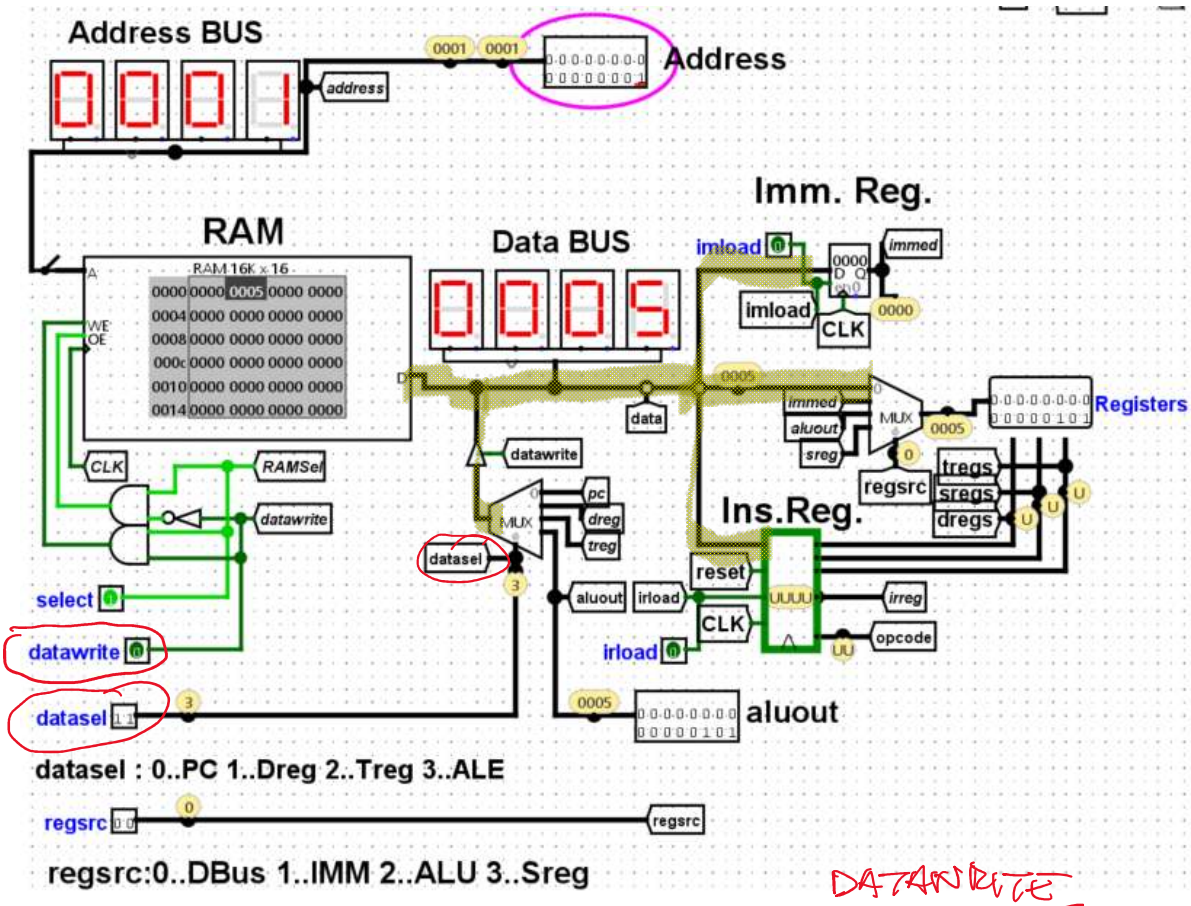
Op.koda	Treg	Sreg	Dreg
7	3	3	3
16 bitni tak. operand			
16			


ImmReg
imload
4008
immed
CLK

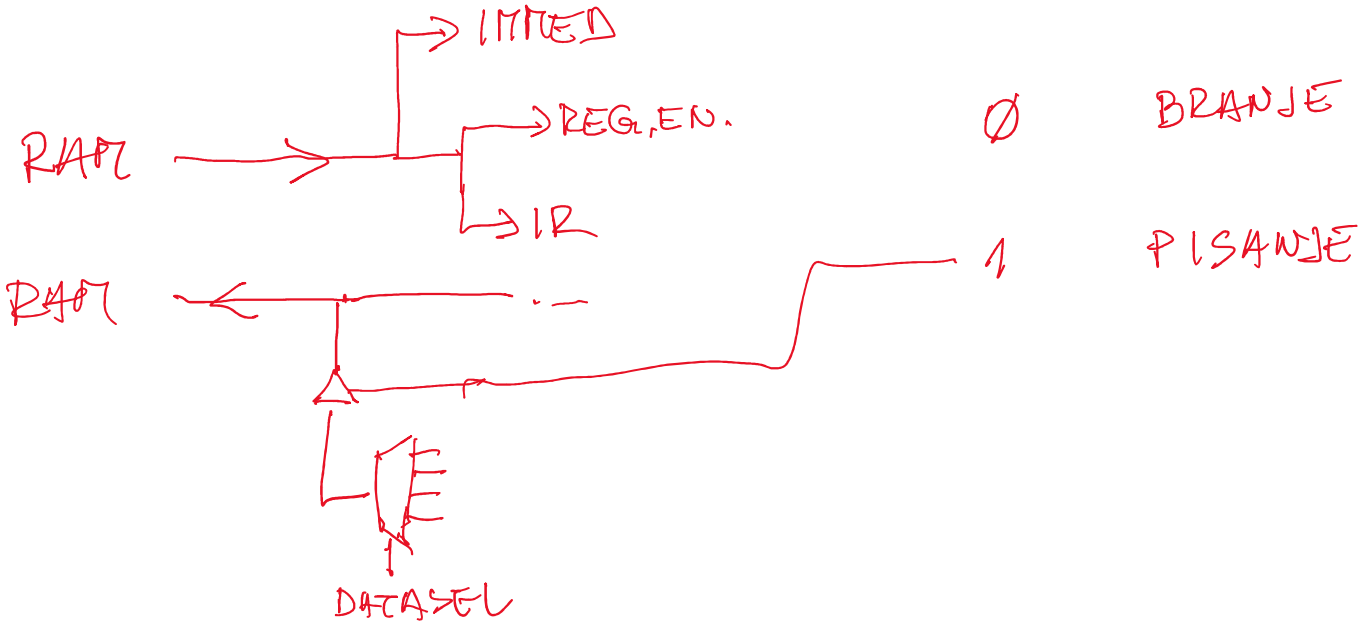

DataIn
0000h
0000
D Q
en 0
0000h
ImmOut
0000
IMLoad
0b
CLKin
0b

Podatkovno vodilo

ponedeljek, 02. november 2020 17:52



DATAWRITE



sw Rd,immed (65)

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

DATAWRITE = 1

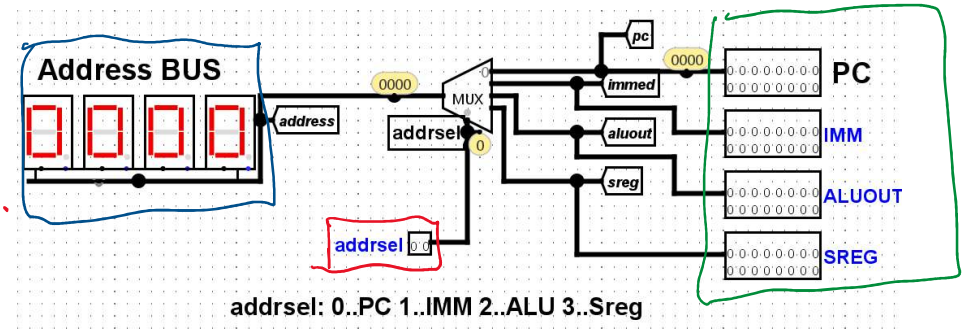
DATASEL = 1 (R_D)

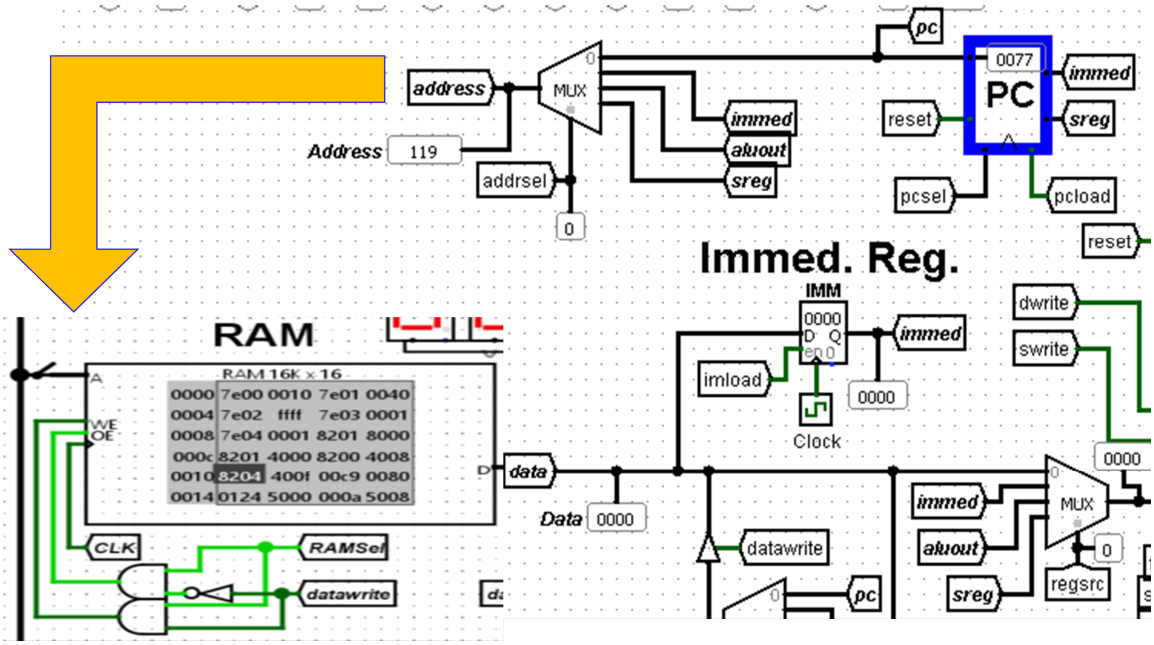
(ADDRSEL = "IMMED")

BRANCH ili SKOK

DATAWRITE = 0

(NASLOV)





ADDRSEL

IMMED (1)

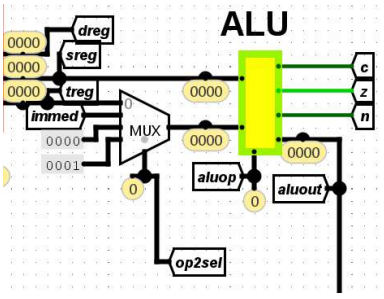
sw Rd,immed (65)
M[immed] <- Rd PC <- PC + 2

ALUOUT (2)

lwi Rd,Rs,immed (66)
Rd <- M[Rs+immed] PC <- PC + 2
ALE

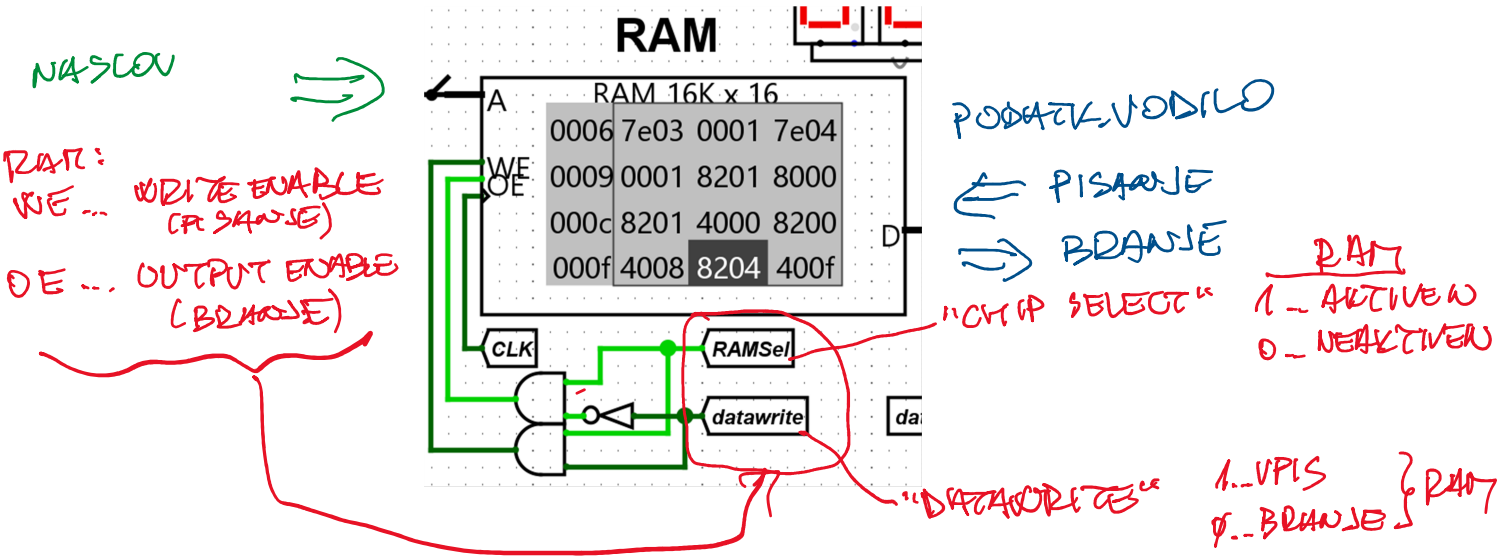
''

lwri Rd,Rs,Rt (73)
Rd <- M[Rs+Rt] PC <- PC + 1
ALE



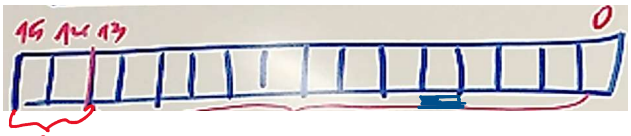
RAM pomnilnik

ponedeljek, 02. november 2020 17:54



NEPOPOLNO

NASLOVNO DEKODIRANJE



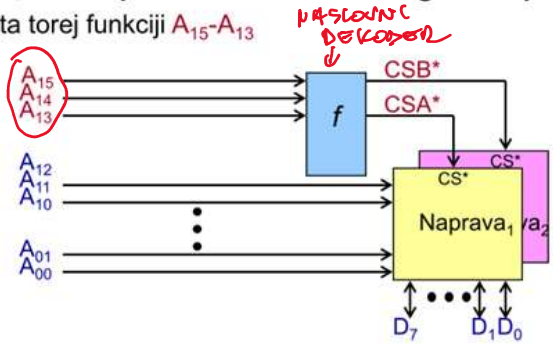
00 RAM (16K)
01 FB_LED (4B)
10 TTY
11 PROSTO

NASLOVI:
0 - 16383
16384 - 32767
32768 - 49151
49152 - 65535

RAM	FB_LED	TTY
1	0	0
0	1	0
0	0	1

Izbira čipa (CS)

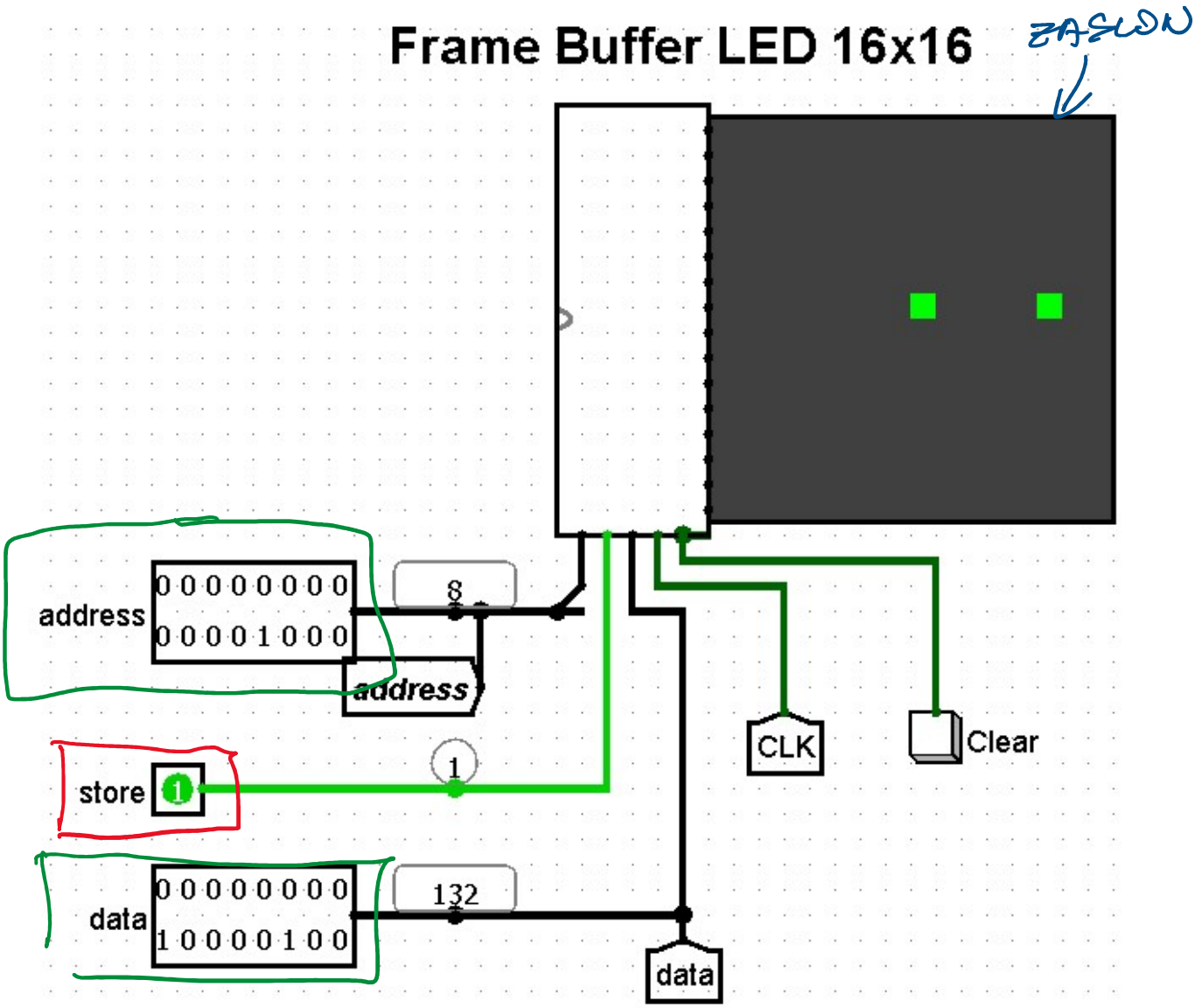
- **Kako priključimo dve (ali več) naprav na vodilo?**
  - Naenkrat mora biti izbran samo en čip (ali nobeden)
  - Za izbiro uporabimo naslednje signale:
    - R/W*, Naslov(A_0-A_{15})
- **Uporabni so biti, ki niso povezani na naslovne signale naprav (A_{15}-A_{13})**
- CSA* in CSB* sta torej funkciji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



FB - FrameBuffer

16 BESED PO 16 BITOV

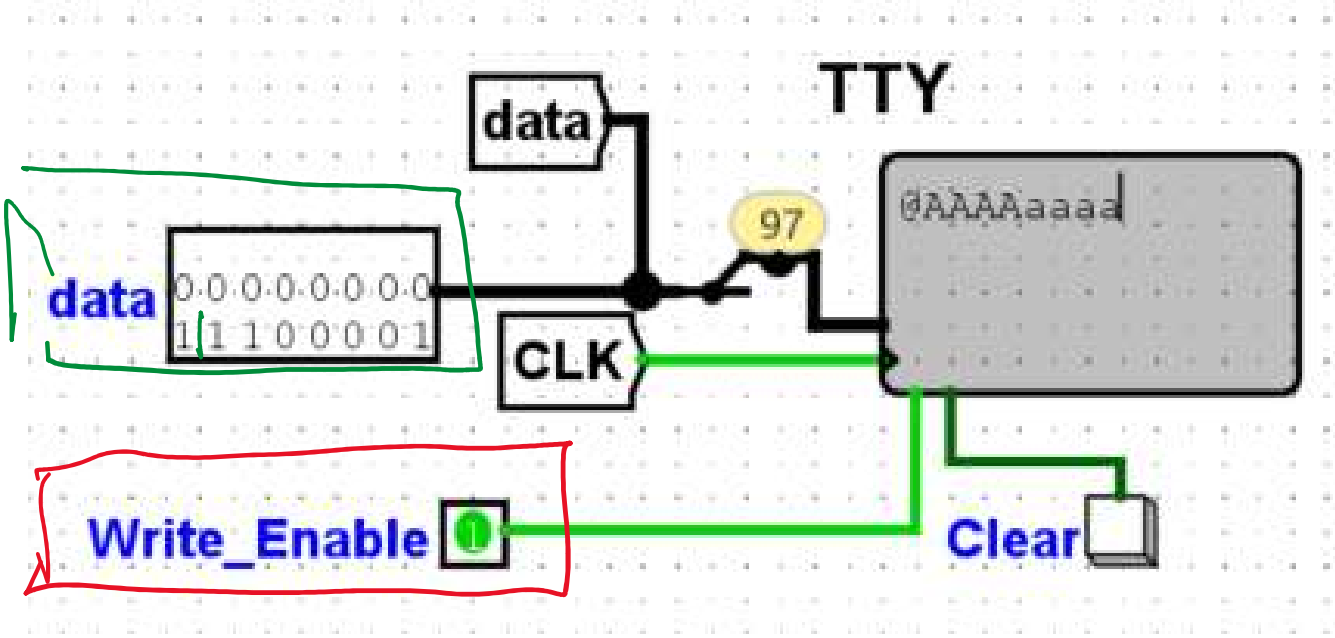
ponedeljek, 02. november 2020 17:55



TTY - Serijski terminal

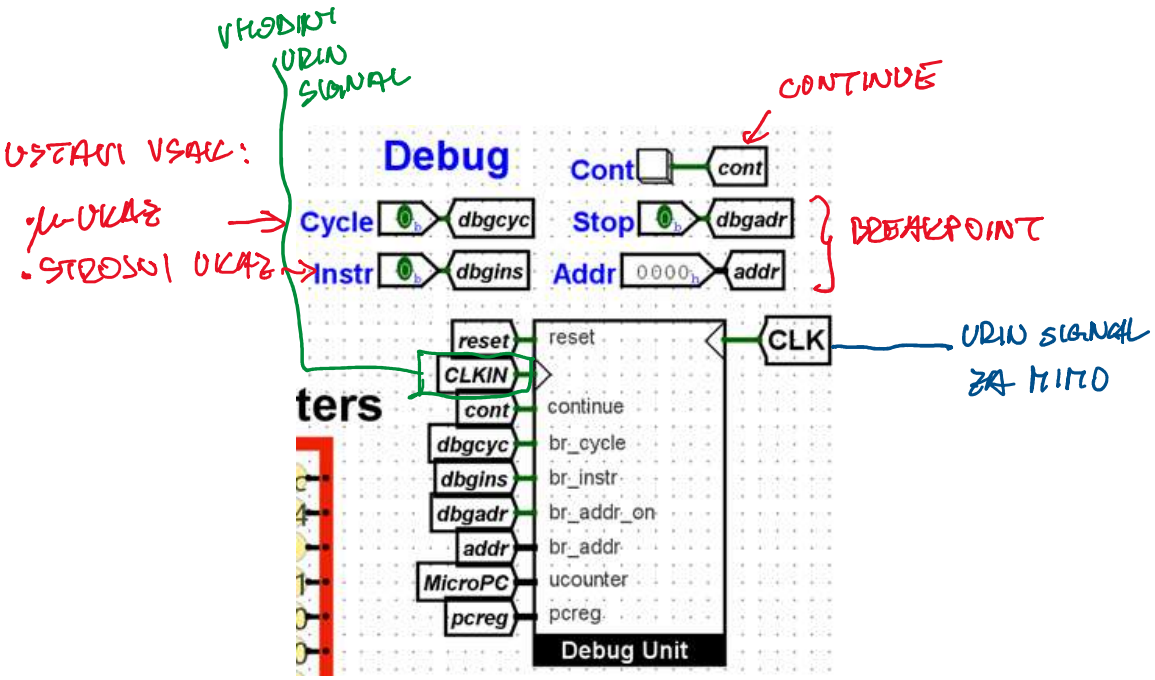
7 BITNA ASCII KODA

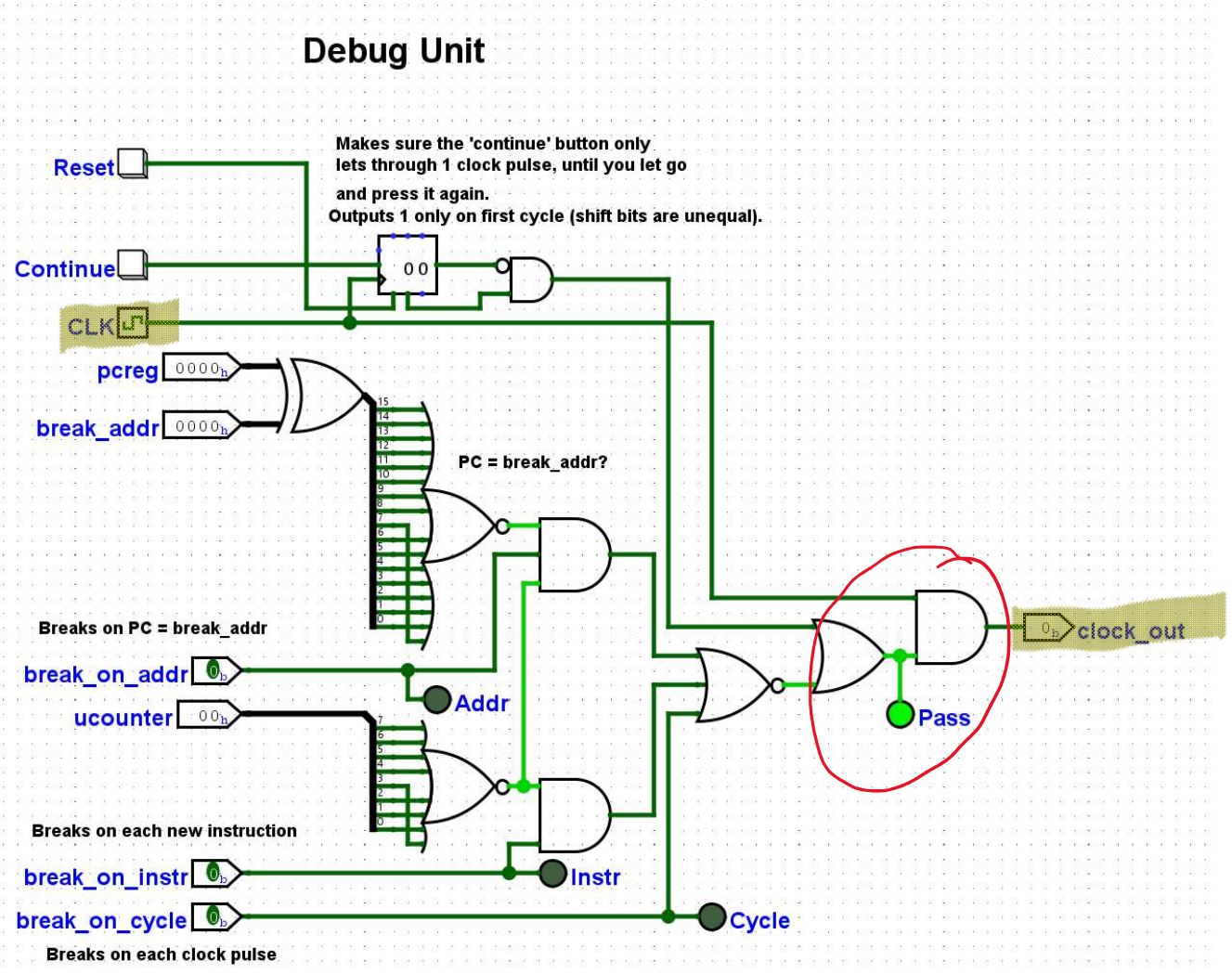
ponedeljek, 02. november 2020 17:56



Debug enota

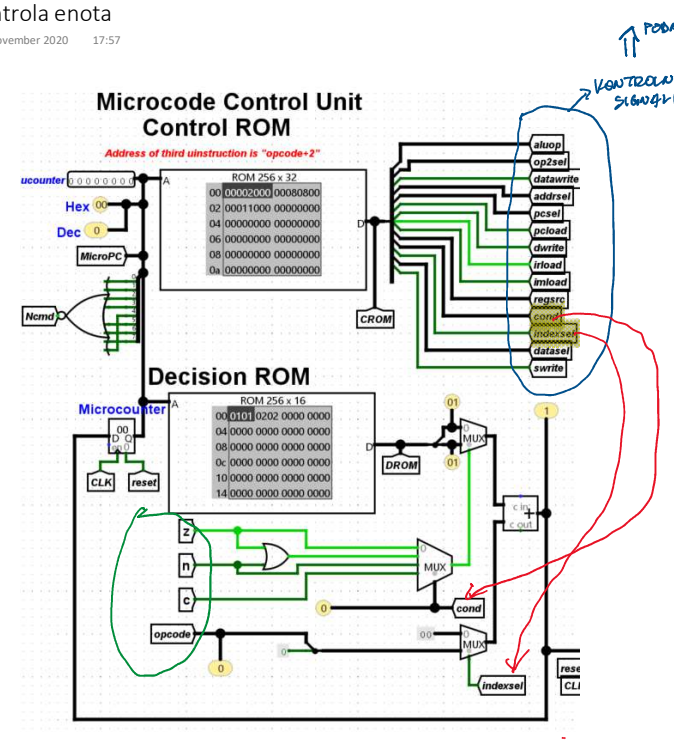
ponedeljek, 24. oktober 2022 17:49




Debug Unit
Reset
Continue
CLK
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).
00
pcreg
0000h
break_addr
0000h
PC = break_addr?
Breaks on PC = break_addr
break_on_addr
0b
Addr
ucounter
00h
Breaks on each new instruction
break_on_instr
0b
Instr
break_on_cycle
0b
Cycle
Breaks on each clock pulse
Pass
0b
clock_out

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

```
1: pcload=1  pcsel=pc, opcode_jump          # PC=PC+1, jump to 2+OPC
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 SE IZVEDE PO USEH ELEMENTARNIH KORAKIH

↓

1 ELEMENTARNI KORAK = 1 μ-UKAZ

DOLOČIMO!

- STANJE KONTR. SIGNALOV
- NASLEDNJI μ-UKAZ

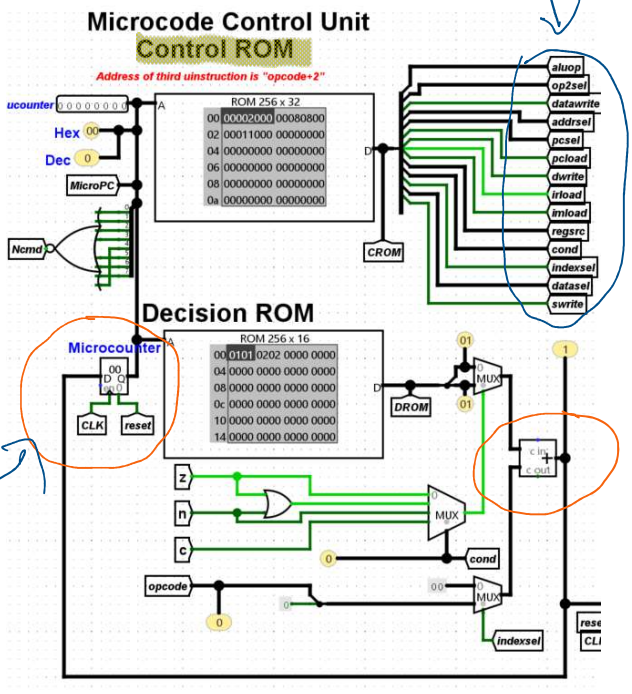
```
00: 00002000 0101    # fetch: addrsel=pc irload=1
01: 00080800 0202    #        pcload=1  pcsel=pc, opcode_jump
02: 00011000 0000    # 0:     aluop=add  op2sel=treg  dwrite=1  regsrc=aluout, goto fetch
2a: 00004000 8282    # 40:    addrsel=pc  imload=1
41: 00001000 8484    # 63:    addrsel=pc  dwrite=1  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
```

2 SKUPNA μUKAZA ZA VSE STROJNE UKAZE

STROJNA μ-UKAZA

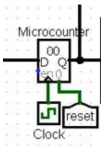
add Rd,Rs,Rt (0) 3μ-UKAZE
Rd <- Rs + Rt PC <- PC + 1

jnez Rs,immed (40) 5μ-UKAZOV
if Rs != 0, PC <- immed else PC <- PC + 2

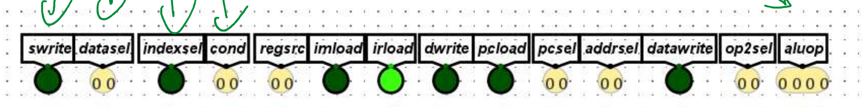


SESTAVA KE:

- μPC
  DOLOČA NASLOV μ-UKAZA ZA IZVEDBO



- 2 ROM POMNILNIKA
  - 32BITNI "CONTROL" ROM
    - VSEBUJE STANJE KONTROLNIH SIGNALOV

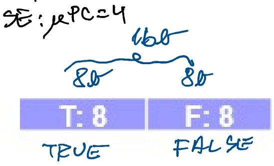


  - 16BITNI "DECISION" ROM
    - DOLOČA NASLOV NASLEDNJEGA μ-UKAZA

PRIMERI:

(5, 5) ... BREZPOG. SKOK (T, F)

(5, 4) ... TRUE: μPC=5, FALSE: μPC=4



PRIMERA DEKODIRANJA KE:

① "OPCODE JUMP"
- SKOČI NA NASLOV "2+OP.KODA"

μ-PC=1:
- INDEX SEL=1
- DECISION ROM (2,2)

② POGOJNO DOLOČANJE NASLEDNJ. μ-UKAZA

PRIMER: μPC=0x82

IF [2] THEN PCINCR 0x84
ELSE JUMP 0x85

- COND ⇒ "2" (0)
- DECISION ROM 0x[84|85] (T, F)

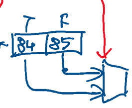


Diagram prehajanja stanj

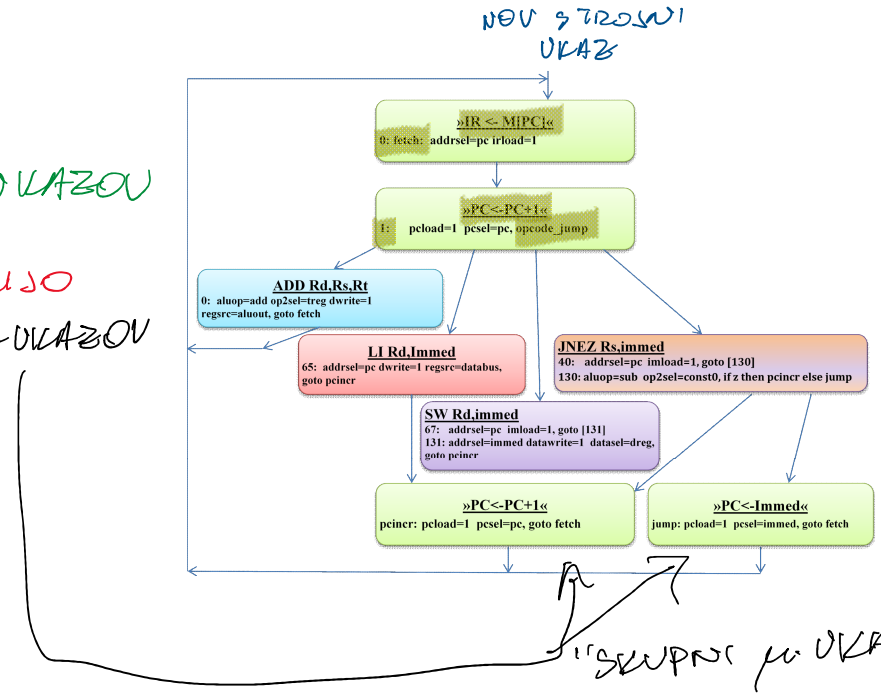
ponedeljek, 09. november 2020 20:59

OBIČAJEN KONCEPT ZA OPIS KE

RAZVIDNA IZVEDBA UKAZOV

OSNOVA ZA OPTIMIZACIJO

- ŠT. STANJ ⇔ ŠT. μ-UKAZOV



CEVOVODNA IZVEDBA:

- POENOSTAVLJENJE ZA VSE UKAZE
- ELEMENTARNI KORAKI → STOPNJE

SIGNAL1=VREDNO1 SIGNAL2=... NASL. μ-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	z, norz, n, c	kontr. enota

NASLEDNJI μ-UKAZ

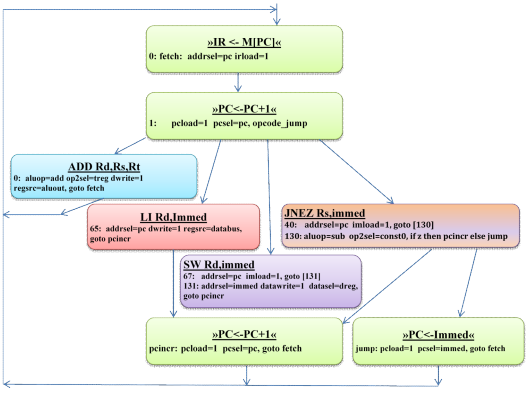
- PRAZNO → μPC+1
- GOTO OZNAKA → V VRSTICI "OZNAKA:"
- IF POGOJ THEN OZNAKA1 ELSE OZNAKA2
- IF POGOJ THEN OZNAKA (ELSE μPC+1)
- OPCODE_JUMP → NA NASLOVU "OP.KODA+2"

JNEZ 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
```

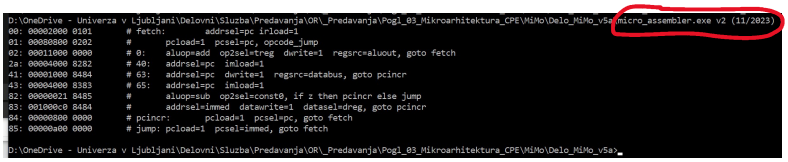
RAZPORED μUKAZOV V ROM POMNILNIKIH

Naslov	Vsebina
0	FETCH
1	PC←PC+1, OPCODE_JUMP
2-129	OP.K. 0 ... 127 — PRVI POSEBNI μ-UKAZ ZA VSAK STROJNI UKAZ (128)
130+ 0x82+	VSI OSTALI μ-UKAZI



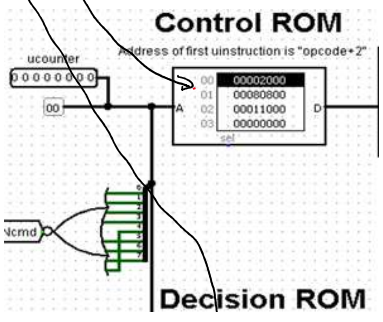
MICRO_ASSEMBLER.EXE .PL

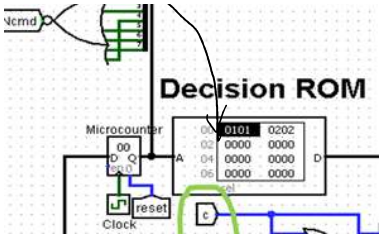
IME.DEF



UCONTROL.ROM

UDECISION.ROM





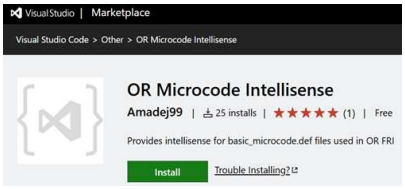
datoteka **basic_microcode.def**

```
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
```

se prevede v

indexsel, pcload

```
00: 00002000 0101    # fetch: addrsel=pc irload=1
01: 00080000 0202    #            pcload=1  pcsel=pc, opcode_jump
02: 00011000 0000    # 0:       aluop=add  op2sel=treg  dwrite=1  regsrc=aluout,    goto fetch
2a: 00004001 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
```



VSCode u-ass plugin
Orodje za lažje mikroprogramiranje
(Amadej Milićev)

3.2.5 Zbirnik

ponedeljek, 09. november 2020 22:01

- **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

```
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
```

0xC

R1 R2

Op.koda	Treg	Sreg	Dreg
7	3	3	3

FORMAT 1

Op.koda	Treg	Sreg	Dreg
7	3	3	3
16 bitni tak. operand			
16			

FORMAT 2

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

SINTAKSA OP.KODA OPIS

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			(0) PC	0
# PC=PC+1, jump to 2+OP	1 (OPCODE_JUMP)						1	0 (PC+1)		1
# Read Immediate operand -> IMRegister					1				(0) PC	42 =OPK+2
# ALU: Rs-0, If z then pcincr else jump		0 ("Z")	1 (SUB)	2 (CONSTP)						130
# Increment PC and goto new command;							1	0 (PC+1)		132
# Set address to immed and goto new command							1	1 (IMMED)		133

z=0

z=1

GOTO FETCH

2b: VIR NASLOVA

2b: VIR VSEBINE PC

1b: VPIS V PC

1b: VPIS V IR

1b: VPIS V TAK.REG. (IMM)

2b: 2.OPERAND ZA

4b: VRSTA AL OPERACIJE

2b: POGOJ ZA NASL. μ-UKAZ

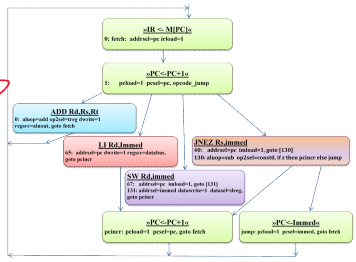
1b: SKOK NA "OPK+2"

3.2.6 Mikroprogramirana vs trdoožičena CPE

ponedeljek, 09. november 2020 22:01

Izhodišča:

- **Diagram poteka**
- **Elementarni koraki**
- **Poenostavitev CPE,ISA**



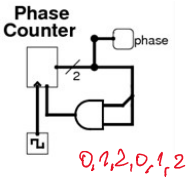
μPROGR. VE
+FLEKSIBILNOST

TRDOOŽIČENA VE
+HITREJŠA
-KOMPLEKSNEJŠA

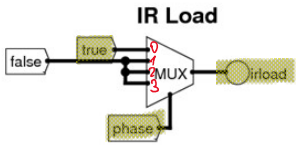
Elementarni koraki:

0 .. Branje ukaza -> IR
1 .. Format 2: branje operanda
Format 1: nop
2 .. Vse operacije :
ALE, skok, reg.write, R/W from Mem

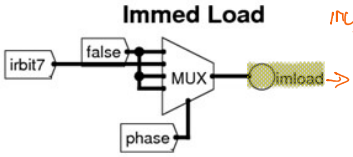
FETCH
EXECUTE



0,1,2,0,1,2 ---



PLOAD ELK.
1 0
0 1,2



IMLOAD ELK.
0 0
b7 1
0 2

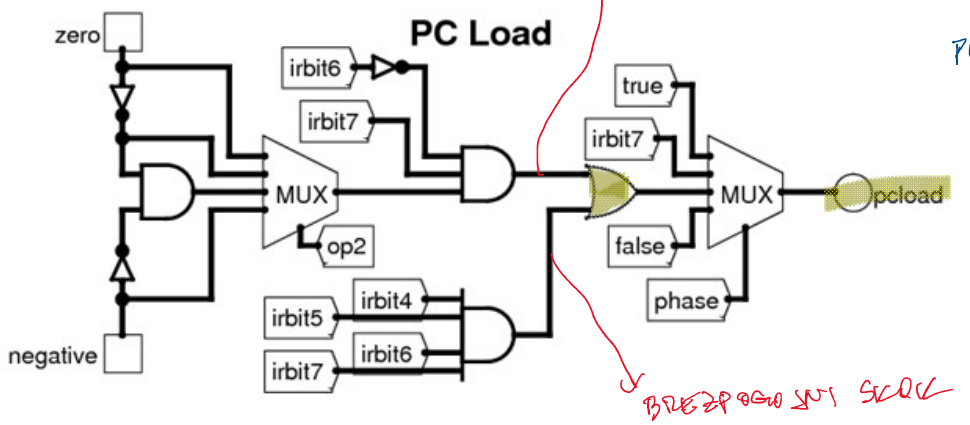
irbit7:

0 .. 8-bitni ukaz (1bajt)
1 .. 16-bitni ukaz (2 bajta)

POGOJNI SKOKI

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



PCLOAD ELK.
1 0 (PC+1)
b7 1 (IMM)
2

1 .. SKOK → POGOJNI / POGOJNI
0 .. SICER

BREZPOGOJNI SKOK

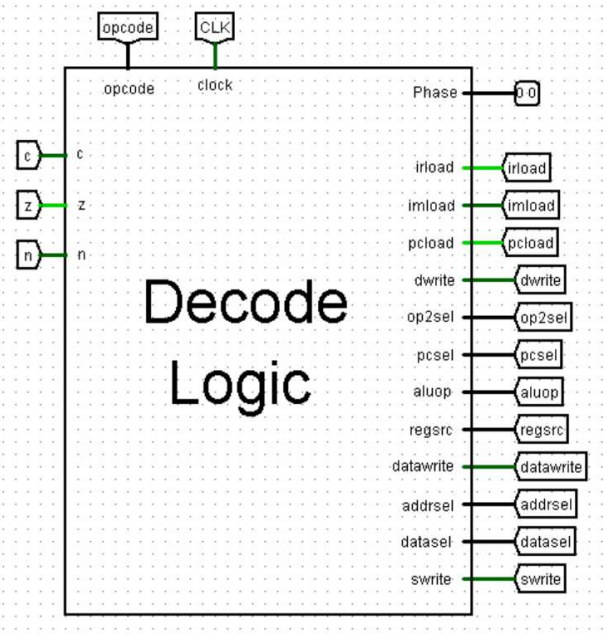
b_7b_6 b_5b_4

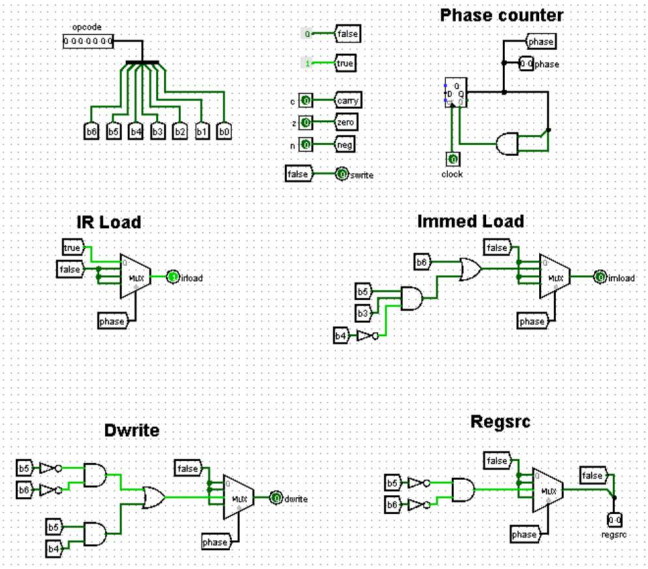
11	11	JMP immed

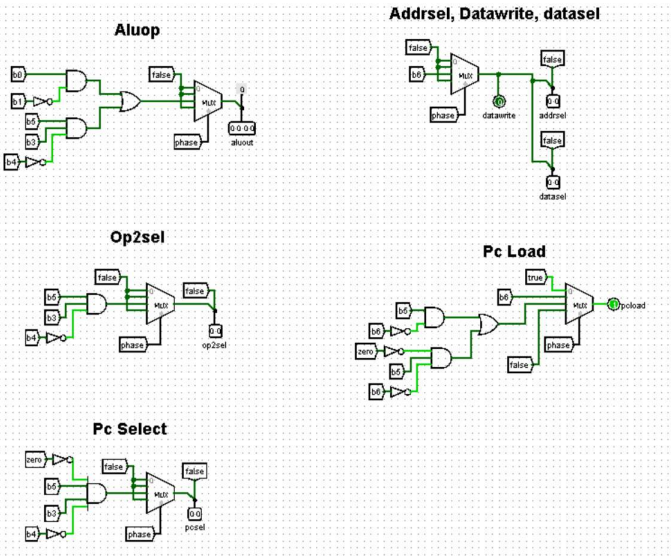
Podrobnejši opis - dodatno gradivo:
http://minnie.tuhs.org/CompArch/Tutes/week03.html

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

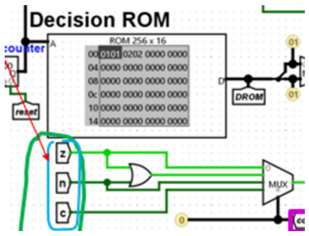
TRDOŽIČEVA LE:
- V/I ENAKI
- SIGNALE GENERIRAMO Z LOG. VEZJI





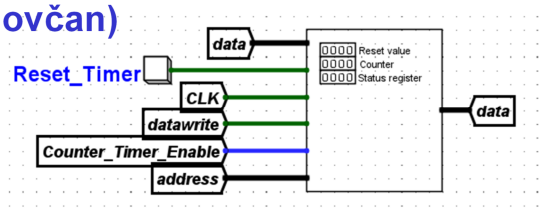


2023/24 – novosti: MiMo v1 v05a

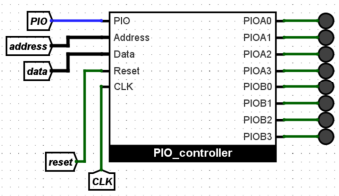


- **Sprememba MiMo v05a:**
  - vezave zastavic v KE
  - mikrozbirnika: micro_assembler.exe v2 (11/2023)

- **Dodatni študentski prispevki:**
  - Timer/Counter : (avtor: Jakob Jelovčan)

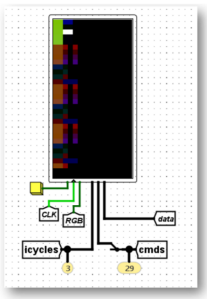


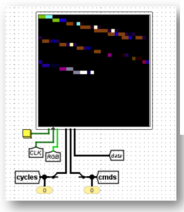
  - PIO Krmilnik (po FRI-SMS):



2024/25 – novosti: MiMo v1 v0.5a

- Dodatni študentski prispevki:





Druga različica te naprave je vizualizacija dolžine ukaza v programu. V tem primeru je spremenjena vrednost vhoda X koordinate. Vhod je št. ciklov v posameznem ukazu. Za vsak ukaz v programu začne barvati od leve prosti desni, vsak ukaz v svoji vrstici.

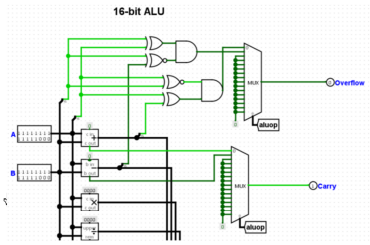
MiMo: primer trdoožičene realizacij...

Odločil sem se dodati RGB zaslon. Ima 6 vhodov.

1) Reset – z kliku na rumen gumb resetira zaslon v na povsem črno
2) Clock – tunel CLK (časovnik)
3) Write enable – tunel RGB (prižge ko je pravi naslov)
4) X coordinate – št. vseh ciklov
5) Y coordinate – št. izvedenih ukazov
6) Data in 565 RGB (16 bit) – tunel data (navadno vrednost registra)

Ta naprava deluje na istih naslovih kot RAM, saj grafično predstavlja izvajanje ukazov iz RAM-a. Prvi ukaz začne levo zgoraj in proti desni, drugi ukaz eno vrstico nižje itd.

RGB zaslona (Gašper Čulk)



I also renamed the 'norz' flag to 'v' and addressed a minor issue in the 'micro_assembler.pl' file. After making the necessary fixes, I packaged it into an executable file named micro_assembler_v2. The issue involved the assembler incorrectly translating the 'if x then y' partial jumps. The solution was straightforward: I simply needed to rename the variable in line 169 from $jump to $tjump.

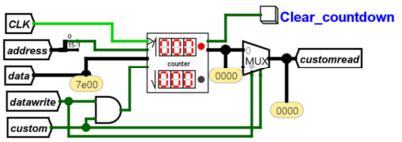
MiMo Overflow flag (Nik Uljarević)

Realiziral sem števec, ki lahko hkrati šteje (štoparica) in odšteva (časovnik). Nastavi se mu lahko začetna vrednost do 999, če je več avtomatsko nastavi na 999. Odštevanje se začne na naslovih, ki imajo lsb 1, seštevanje pa na naslovih ki imajo lsb 0. Vrednost lahko tudi preberemo.

Testni program je v točki 4.

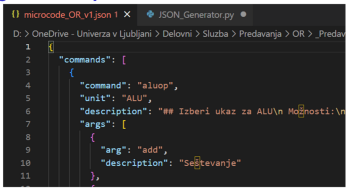
```
...
    li r5, 999     # counter value
    sw r5, 49153   # start countdown with start value of 999
...
    lw r5, 49153   # load value of countdown
...
```

Končen izgled in povezava naprave

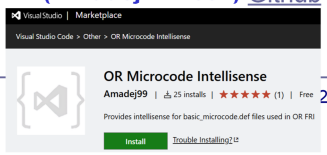


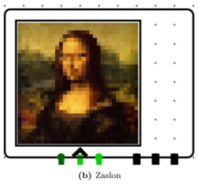
Števec/časovnik (Klemen Šuštar)

RGB zaslon 256x256 (Jan Ljubič)



VSCode u-ass plugin (Amadej Milićev) Github





RGB zaslon 32x32 (Marcel Homšak)

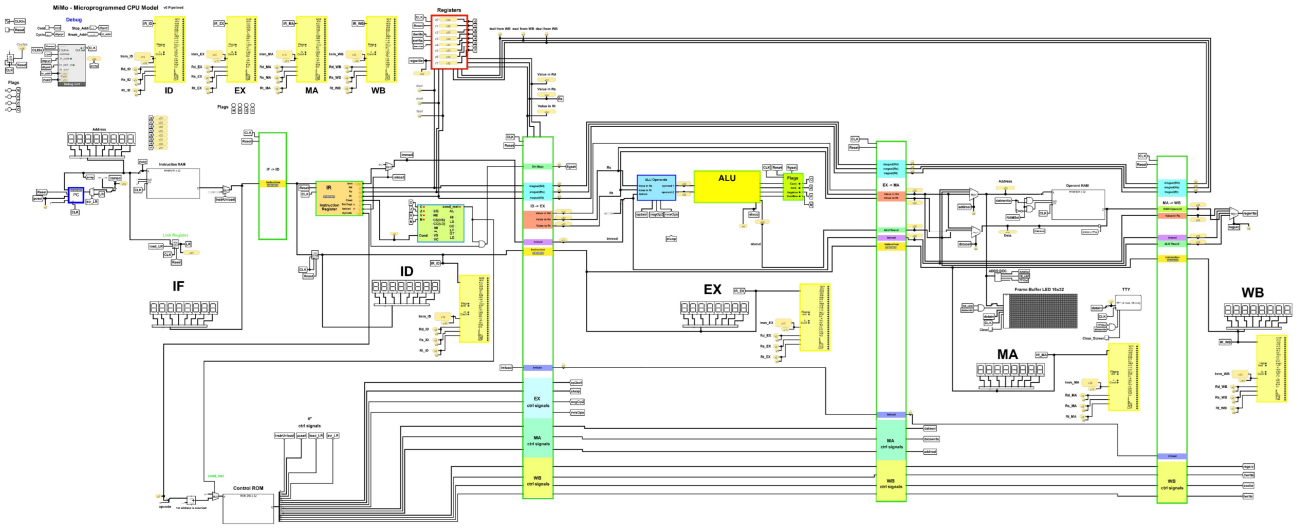
2024/25 – novosti: MiMo v2 (cevovodna različica)

Cevovodne različice MiMo v1 modela:

- mimo_32bit_v3 (osnovna cevovodna izvedba)
- mimo_32bit_v3.1 (v3+zaklenitev)
- mimo_32bit_v3.2 (v3.1+premoščanje)
- mimo_32bit_v3.3 (v3.2+predikcije)

Osnovna distribucija :

- OR:
  - https://github.com/LAPSyLAB/MiMo_Student_Release/tree/main/MiMo_v2_Pipelined_versions
- Diploma:
  - https://github.com/kiriltofiloski/pipelined-MiMo-CPU

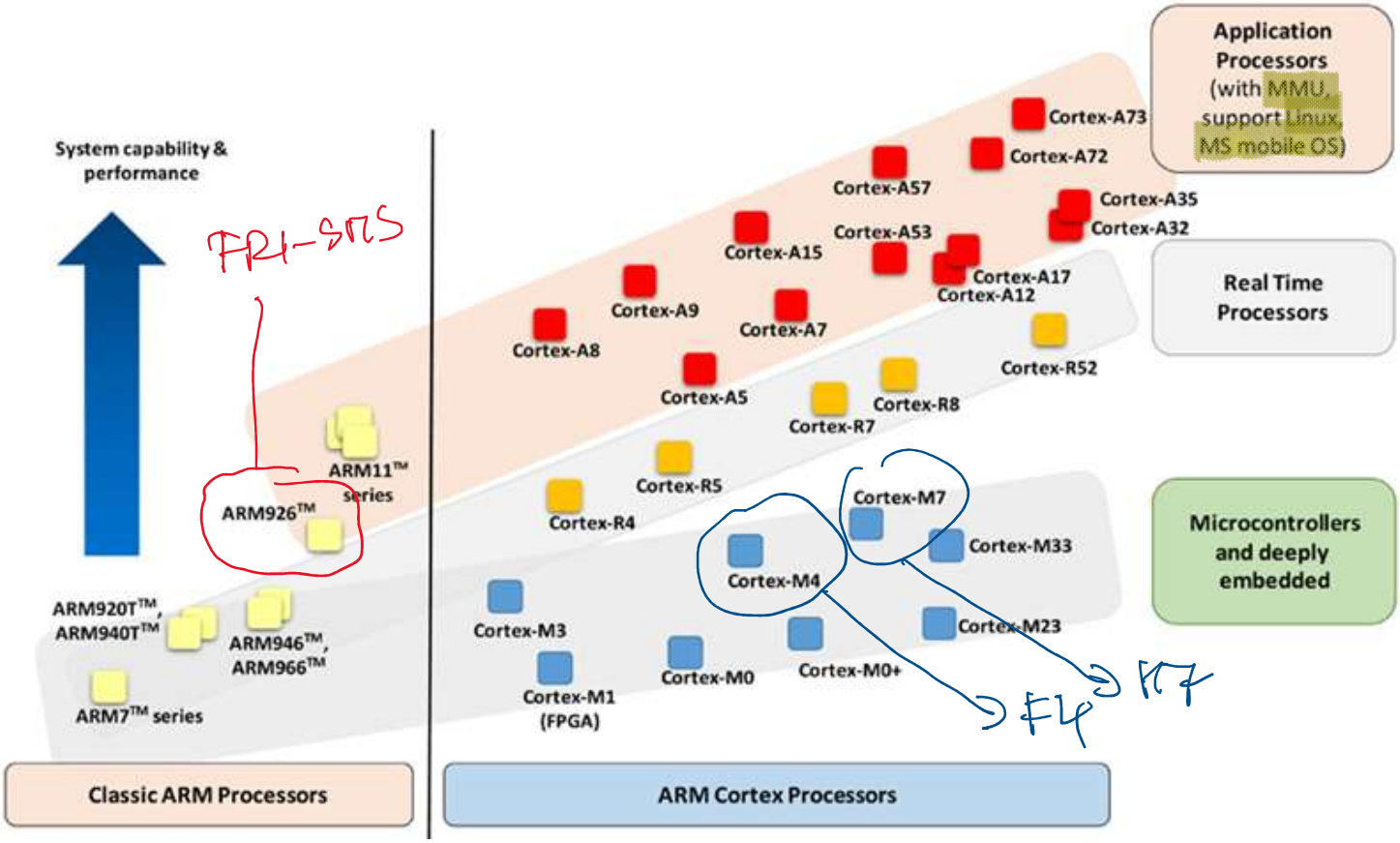


Izhodiščna dokumentacija (diploma še ni objavljena):
https://mimo-pipelined.gitbook.io/documentation

Instruction Code	Arguments	Function
mov	Rd, Rs/Immediate	Rd <- Rs/Immediate
mvn	Rd, Rs/Immediate	Rd <- NOT Rs/Immediate
add	Rd, Rs, Rt/Immediate	Rd <- Rs + Rt/Immediate
sub	Rd, Rs, Rt/Immediate	Rd <- Rs - Rt/Immediate
rsb	Rd, Rs, Rt/Immediate	Rd <- Rt/Immediate - Rs
mul	Rd, Rs, Rt/Immediate	Rd <- Rs * Rt/Immediate
div	Rd, Rs, Rt/Immediate	Rd <- Rs / Rt/Immediate
rem	Rd, Rs, Rt/Immediate	Rd <- Rs % Rt/Immediate
and	Rd, Rs, Rt/Immediate	Rd <- Rs AND Rt/Immediate
orr	Rd, Rs, Rt/Immediate	Rd <- Rs OR Rt/Immediate
eor	Rd, Rs, Rt/Immediate	Rd <- Rs XOR Rt/Immediate
nand	Rd, Rs, Rt/Immediate	Rd <- Rs NAND Rt/Immediate
nor	Rd, Rs, Rt/Immediate	Rd <- Rs NOR Rt/Immediate
bic	Rd, Rs, Rt/Immediate	Rd <- Rs AND NOT(Rt/Immediate)
cmp	Rs, Rt/Immediate	Rs - Rt Set flags
cmn	Rs, Rt/Immediate	Rs + Rt Set flags
tst	Rs, Rt/Immediate	Rs XOR Rt Set flags
teq	Rs, Rt/Immediate	Rs AND Rt Set flags
lsl	Rd, Rs, Rt/Immediate	Rd <- Rs << Rt/Immediate
lsr	Rd, Rs, Rt/Immediate	Rd <- Rs >> Rt/Immediate
asr	Rd, Rs, Rt/Immediate	Rd <- Rs >> Rt/Immediate, Filled bits are sign bit
ror	Rd, Rs, Rt/Immediate	Rd <- Rs ROLL RIGHT Rt/Immediate
rol	Rd, Rs, Rt/Immediate	Rd <- Rs ROLL LEFT Rt/Immediate
j	Immediate/Label	PC <- Immediate/Label
b	Immediate/Label	PC <- PC + Immediate/Label
bl	Label	Jump to subroutine load Link Register with return address
rts	None	Return from subroutine
ldr	Rd, Immediate	Rd <- M[Immediate]
str	Rd, Immediate	M[Immediate] <- Rd
nop	None	No operation (Used to skip cycle and avoid pipeline hazards)

3.3 Družina ARM procesorjev

ponedeljek, 09. november 2020 22:03



CORTEX - A (APPL.) → SPLOŠNI NAMEN

CORTEX - M (MIKROKRMILNIKI) → VGRAJENI SISTEMI

↳ RTOS → FREERTOS
→ ZEPHYR