

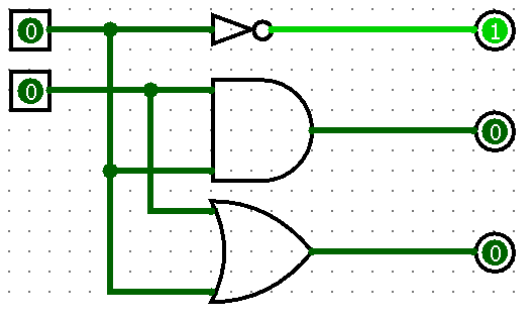


ORGANIZACIJA RAČUNALNIKOV

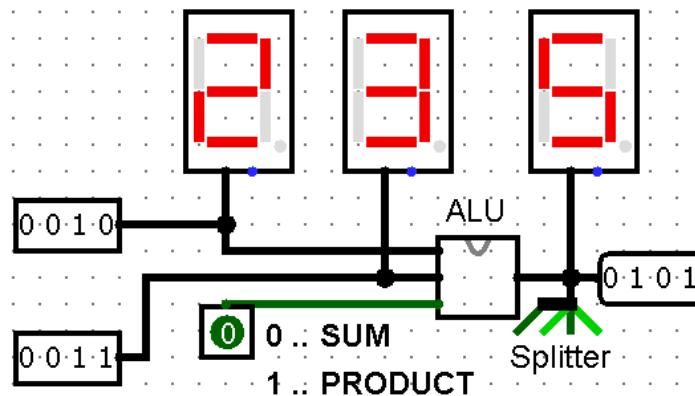
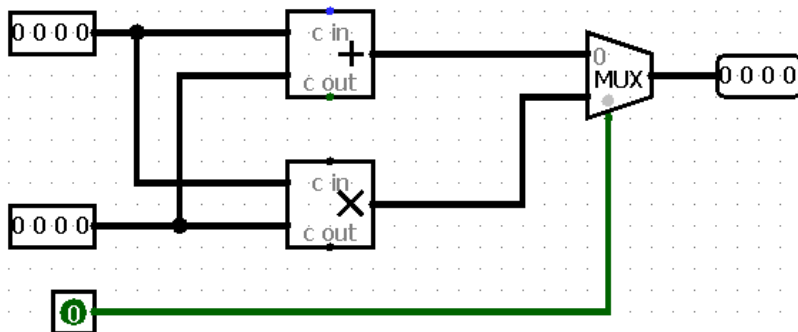
Laboratorijske vaje

Vaja 5: Implementacija strojnih ukazov z mikropodprogrami v MiMo

■ Osnovno vezje



■ Preprosta enota: ALE



Distribucija MiMo Student Release

DISTRIBUCIJA

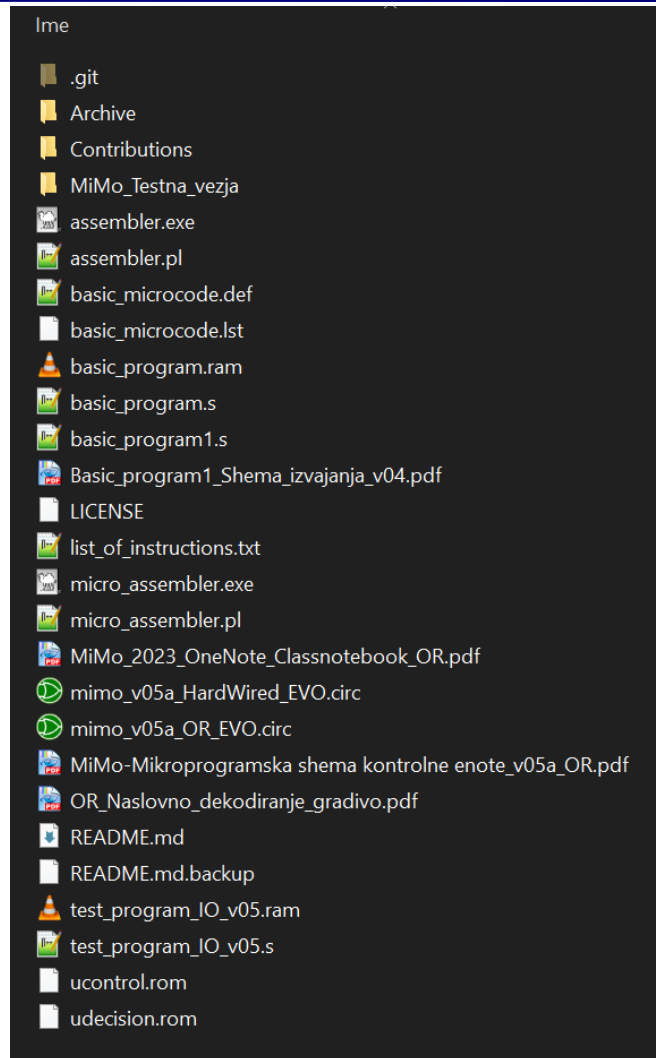
.EXE
.PL (PERL)
.CIRC LOGISIM
.S ZBIRNIK → .RAM
.DEF μ-ZBIRNIK → .ROM

Zbirnik {
Mikroprogram

Spisek ukazov
Mikro-zbirnik {

MiMo model {

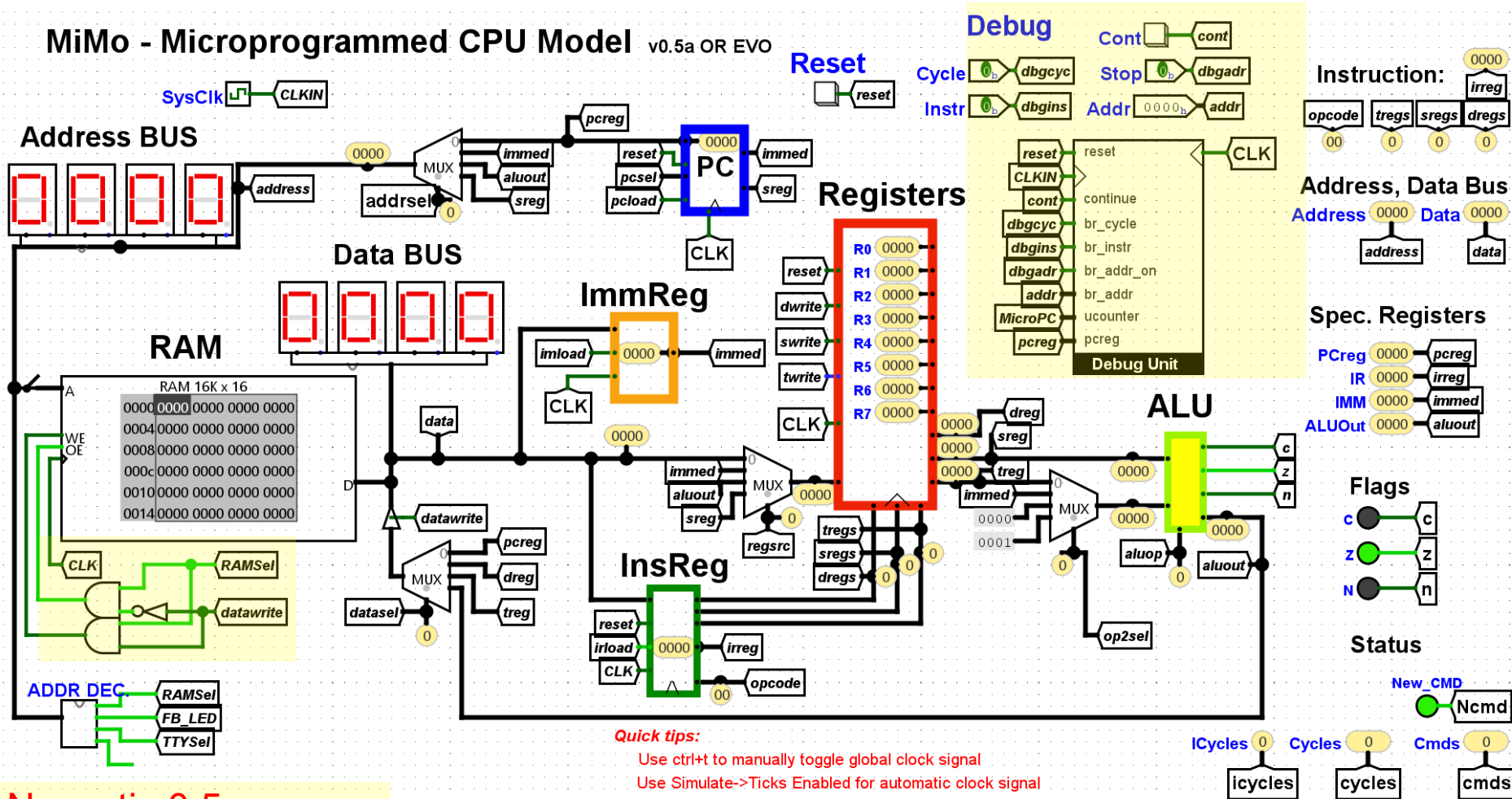
Vsebina .ROM {



https://github.com/LAPSYLAB/MiMo_Student_Release

MiMo – Podatkovna enota v0.5a

MiMo - Microprogrammed CPU Model v0.5a OR EVO



Novosti v0.5a:
pogoji v KE

https://github.com/LAPSyLAB/MiMo_Student_Release

Testni program (basic_program1.s)

BASIC_PROGRAM1.S

```
# This program uses the instructions defined in the
# basic_microcode.def file. It counts down to 0 from 2
# and stores -1 in memory location 16.
# (c) GPL3 Warren Toomey, 2012
#
main:    li      r1, 2           # r1 is the counter
        li      r2, -1        # Used to decrement r1
loop:   add     r1, r1, r2     # r1--
        jnez    r1, loop      # loop if r1 != 0
        sw     r2, 16         # Save the r2
```

$R1 \leftarrow 2$
 $R2 \leftarrow -1$

ZAOKA: $R1 \leftarrow R1 + R2$

B ZAOKA, ČE $R1 \neq 0$

$R2 \rightarrow M[16]$

Shema izvajanja programa v zbirniku (razlaga)

Shema izvajanja programa v zbirniku v MiMo modelu

v 0.4

RAM		Format strojnega ukaza				Program v zbirniku		Kontrolni naslov		Mikroprogram		Decision ROM	
Nasl.	Vsebina strojni uk.	Op.koda	Treg	Sreg	Dreg	oznaka: ukaz operandi	Dec	Hex	Kontrolni signali, naslednji mikroukaz		T	F	
							00	00	fetch: addrsel=pc irload=1		01	01	
							01	01	pcload=1 pcsel=pc, opcode_jump		02	02	
0000:	7e01	63			1	main: li r1, 2	65	41	addrsel=pc dwrite=1 regsrc=databus, goto pcincr		84	84	
0001:	0002	Tak. operand						84	pcincr: pload=1 pcsel=pc, goto fetch		00	00	
0002:	7e02	63			2	li r2, -1	65	41	addrsel=pc dwrite=1 regsrc=databus, goto pcincr		84	84	
0003:	ffff	Tak. operand						84	pcincr: pload=1 pcsel=pc, goto fetch		00	00	
0004:	0089	0	2	1	1	loop: add r1,r1,r2	2	2	aluop=add op2sel=treg dwrite=1 regsrc=aluout, goto fetch		00	00	
0005:	5008	40			1	jnez r1, loop	40	2a	addrsel=pc imload=1		82	82	
0006:	0004	Tak. operand						82	aluop=sub op2sel=const0, if z then pcincr else jump		84	85	
								84	pcincr: pload=1 pcsel=pc, goto fetch		00	00	
								85	jump: pload=1 pcsel=immed, goto fetch		00	00	
0007:	8202	65			2	sw r2, 16	67	43	addrsel=pc imload=1		83	83	
0008:	0010	Tak. operand						83	addrsel=immed datawrite=1 datasel=dreg, goto pcincr		84	84	
								84	pcincr: pload=1 pcsel=pc, goto fetch		00	00	

LI ADD JNE Z
 • • •
 •
 •
 •
 • > ALU

Program: basic_program1.s :

```

main: li r1, 2      # r1 is the counter
      li r2, -1    # Used to decrement r1
loop: add r1, r1, r2 # r1<-r1+r2 (r2=-1 -> r1 decrements)
      jnez r1, loop # if r1 != 0 then jump to loop:
      sw r2, 16    # Save r2 to MEM[16]
    
```

```

0000: 00007e01 0111111000000001  main: li r1, 2
0001: 00000002 0000000000000010
0002: 00007e02 0111111000000010  li r2, -1
0003: 0000ffff 1111111111111111
0004: 00000089 0000000010001001  loop: add r1, r1, r2
0005: 00005008 0101000000001000  jnez r1, loop
0006: 00000004 0000000000000100
0007: 00008202 1000001000000010  sw r2, 16
0008: 00000010 0000000000010000
    
```

```

00: 00002000 0101 # fetch:  addrsel=pc irload=1
01: 00008000 0202 #          pload=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: 00000021 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: pload=1 pcsel=pc, goto fetch
85: 00000a00 0000 # jump:   pload=1 pcsel=immed, goto fetch
    
```

ROM

ZBIRNIK
 -LSTV
 (RAM)

CONTROL → DECISION
 M-NASLOV
 USEBINA ROM

NASLOV
 HEX
 BIN
 VSEBINA

Implementacija ukazov v mikroprogramski CPE MiMo

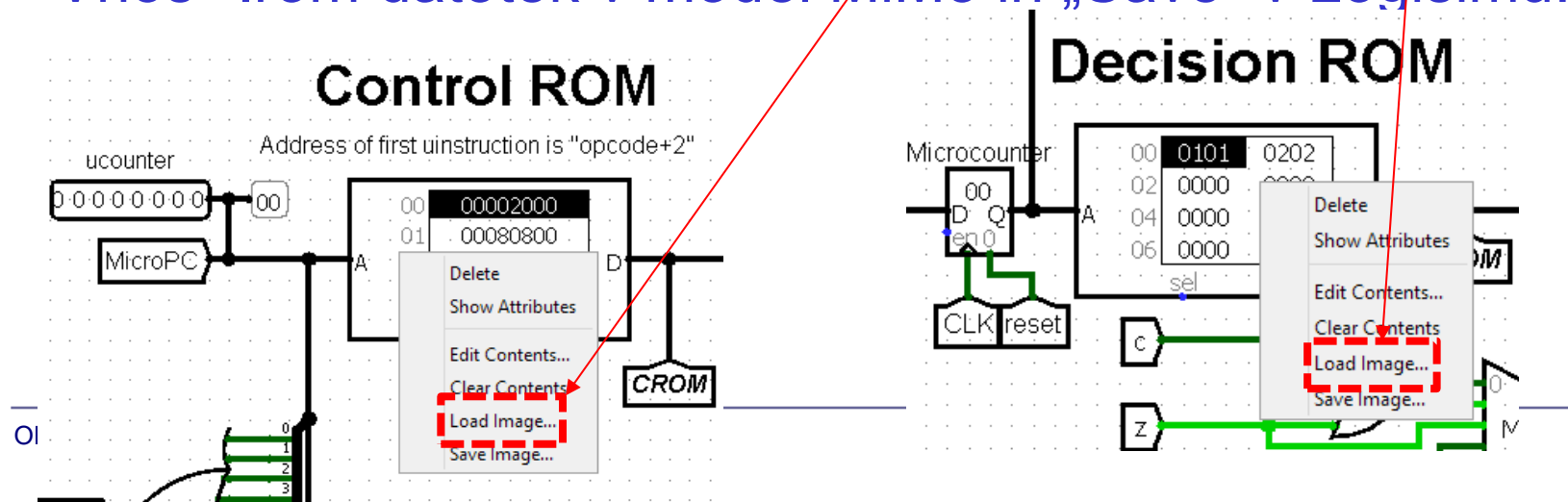
I. Mikroprogramski nivo:

1. Mikroprogramska realizacija v *basic_microcode.def*
2. Prevajanje: *basic_microcode.def* -> *ucontrol,udecision.rom*

```
# sub Rd,Rs,Rt (1)
# Rd <- Rs - Rt      PC <- PC + 1
1: aluop=sub op2sel=treg dwrite=1 regsrc=aluout, goto fetch
```

```
C:\winIDEA\MiMo\Distribucija_2017_18>micro_assembler.exe basic_microcode_sub.def
00: 00002000 0101 # fetch:      addrsel=pc imload=1
01: 00080800 0202 #          pload=1 pcsel=pc, opcode_jump
02: 00011000 0000 # 0:        aluop=add op2sel=treg dwrite=1 regsrc=aluout, goto fetch
03: 00011001 0000 # 1:        aluop=sub 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:   pload=1 pcsel=pc, goto fetch
85: 00000a00 0000 # jump:    pload=1 pcsel=immed, goto fetch
```

3. Vnos *.rom datotek v model MiMo in „Save“ v Logisimu:



II. Nivo zbirnega jezika:

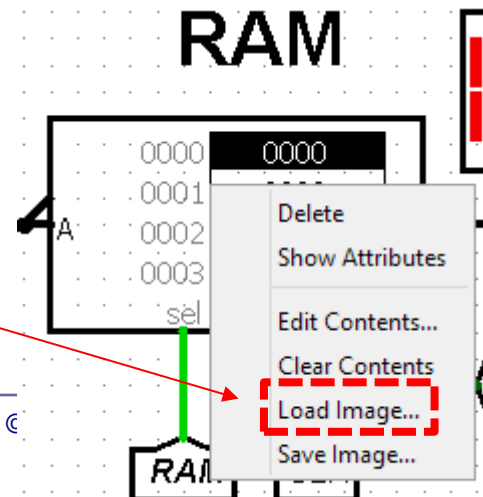
1. Uporaba ukaza v testnem programu:

```
main:  li  r1, 2          # r1 is the counter
      li  r2, 1          # Used to decrement r1
loop:  sub r1, r1, r2    # r1--
      jnez r1, loop     # loop if r1 != 0
      sw  r2, 16        # Save the r2
```

2. Prevajanje: *ime.s* -> *ime.ram*

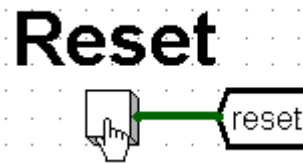
```
C:\winIDEA\MiMo\Distribucija_2017_18>assembler.exe basic_program1_sub.s
0000: 00007e01 0111111000000001  main:  li  r1, 2
0001: 00000002 0000000000000010          li  r2, 1
0002: 00007e02 0111111000000010  loop:  sub r1, r1, r2
0003: 00000001 0000000000000001          jnez r1, loop
0004: 00000289 0000001010001001          sw  r2, 16
0005: 00005008 0101000000001000
0006: 00000004 0000000000000010
0007: 00008202 1000001000000010
0008: 00000010 0000000000001000
```

3. Vnos ime.ram datoteke v model MiMo :



III. Preizkus delovanja:

1. Reset (po potrebi) :



2. Izvajanje po mikroukazih:

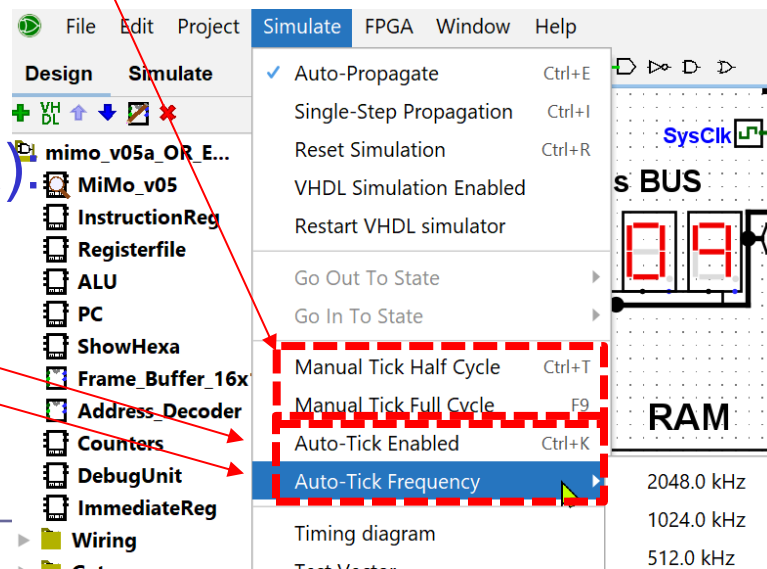
- 2 x pritisk na „Ctrl+T“ ali enkrat „F9“ (ena urina perioda)

3. Uporaba debug enote:

- izvajanje po mikroukazih, strojnih ukazih (manj natančno)
- nastavitvev „breakpoint“-a

4. Tekoče izvajanje (brez ustavljanja).

- Vklop (Ticks Enabled)
- Frekvenca urinega signala





ORGANIZACIJA RAČUNALNIKOV

Laboratorijske vaje

Vaja 5: Dodatna gradiva

3.2.2.11 Debug enota

Vsebina zaslona je prikazana v 4 vrsticah in 16 znakih.

Vhodi:

- clock: urin signal Logisim
- Addr: naslov ukaza ustavitve („breakpoint“)
- uPC (mikroprogramski števec)

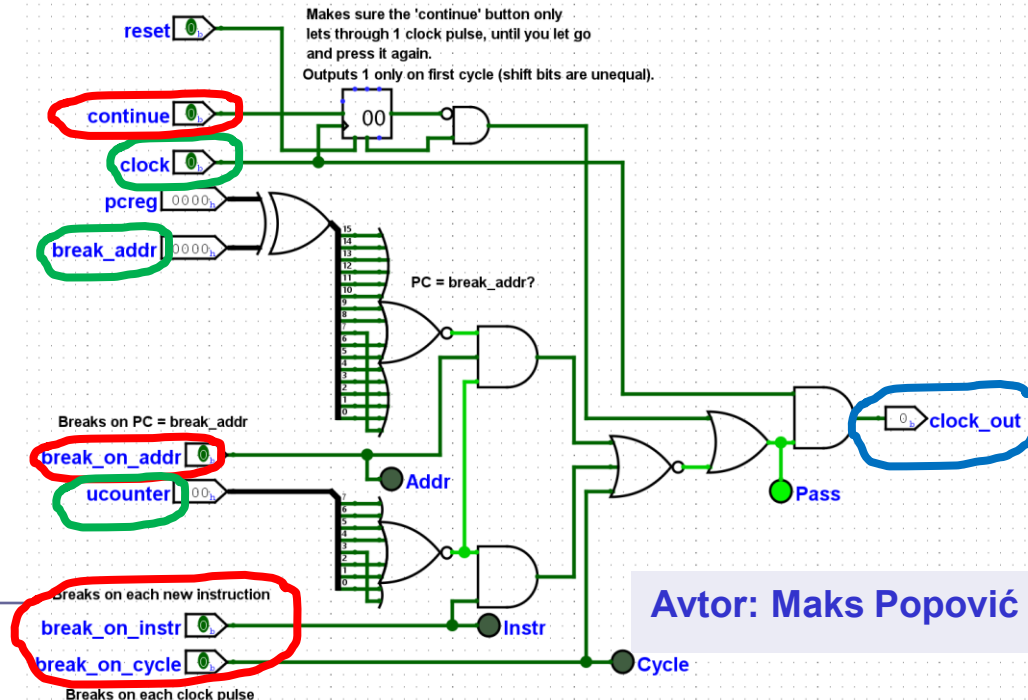
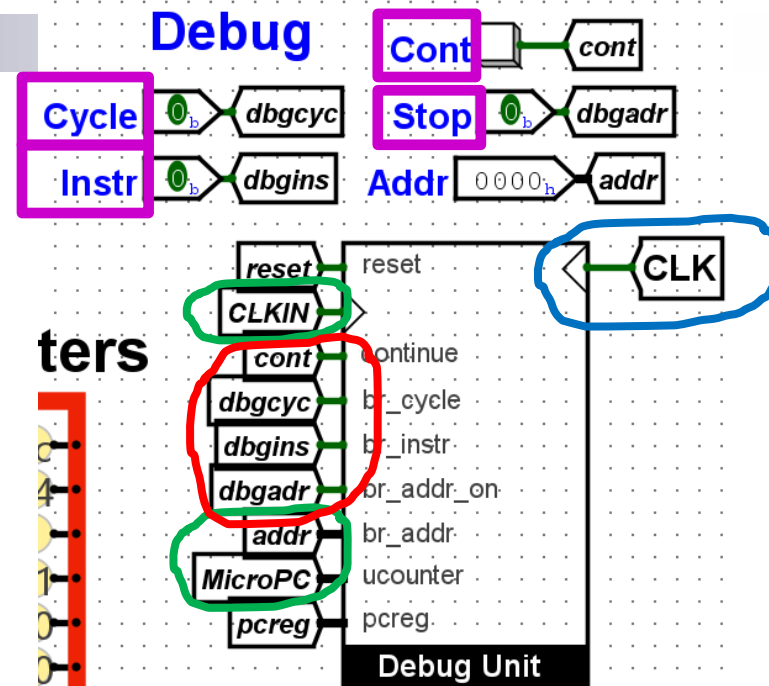
Izhod

- CLK: glavni urin signal sistema

Uporabniške kontrole:

- Cycle: ustavitev vsako periodo
- Instr: ustavitev vsak nov strojni ukaz
- Stop: ustavitev na naslovu Addr
- Cont: nadaljnj izvedbo (po ustavitvi)

Potrebuje stalni urin signal višje frekvence



3.2.2.8 RAM pomnilnik

Naslov RAM 14 bitni

Naslov MiMo 16bitni ???

Naslovno dekodiranje

Izbira čipa (CS)

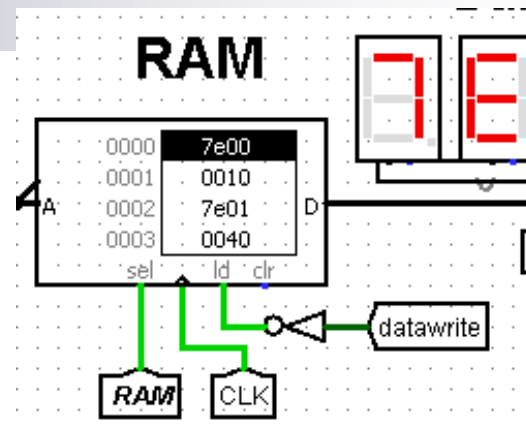
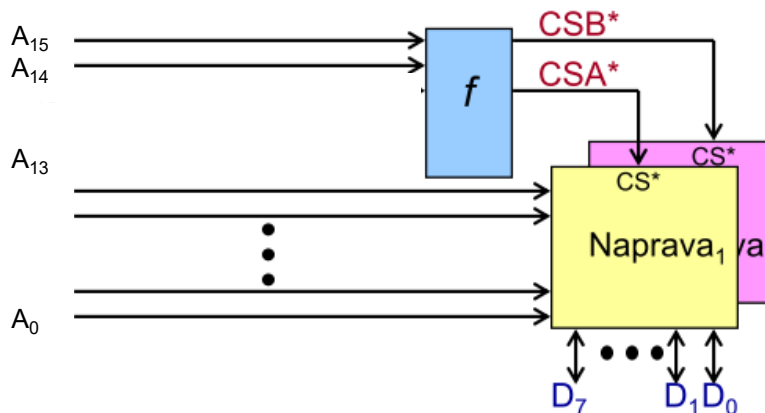
• Kako priključimo dve (ali več) naprav na vodilo?

- Naenkrat mora biti izbran samo en čip (ali nobeden)
- Za izbiro uporabimo naslednje signale:

- R/W*, Naslov(**A₀-A₁₅**)

• Uporabni so biti, ki niso povezani na naslovne signale naprav A₁₅-A₁₄

• **CSA*** in **CSB*** sta torej funkciji A₁₅-A₁₄



Distribucija MiMo Student Release

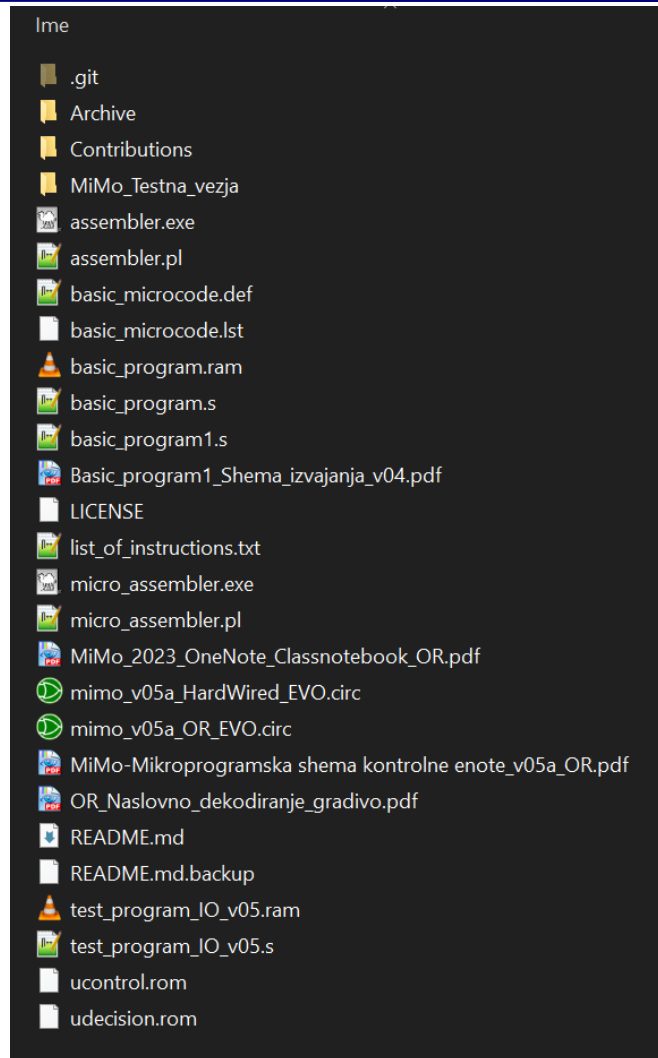
DISTRIBUCIJA

.EXE
.PL (PERL)
.CIRC LOGISIM
.S ZBIRNIK → .RAM
.DEF μ-ZBIRNIK → .ROM

Zbirnik {
Mikroprogram

Spisek ukazov
Mikro-zbirnik {
MiMo model {

Vsebina .ROM {



https://github.com/LAPSYLAB/MiMo_Student_Release

Testni program (basic_program1.s)

BASIC_PROGRAM1.S

```
# This program uses the instructions defined in the
# basic_microcode.def file. It counts down to 0 from 2
# and stores -1 in memory location 16.
# (c) GPL3 Warren Toomey, 2012
#
main:    li      r1, 2           # r1 is the counter
        li      r2, -1        # Used to decrement r1
loop:   add     r1, r1, r2     # r1--
        jnez    r1, loop      # loop if r1 != 0
        sw     r2, 16         # Save the r2
```

$R1 \leftarrow 2$
 $R2 \leftarrow -1$

ZAOKA: $R1 \leftarrow R1 + R2$

B ZAOKA, ČE $R1 \neq 0$

$R2 \rightarrow M[16]$

Testni program (basic_program1.s)

VSEBNA DATI POMNILNIKA PO IZVEDBI SW R2, 16

Logisim: Hex Editor

File Edit Project Simulate Window Help

0000	7e01	0002	7e02	ffff	0089	5008	0004	8202	0010	0000	0000	0000	0000	0000	0000	0000
<u>0010</u>	<u>ffff</u>	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0020	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0030	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Shema izvajanja programa v zbirniku (razlaga)

Shema izvajanja programa v zbirniku v MiMo modelu

v 0.4

RAM		Format strojnega ukaza				Program v zbirniku		Kontrolni naslov		Mikroprogram		Decision ROM	
Nasl.	Vsebina strojni uk.	Op.koda	Treg	Sreg	Dreg	oznaka: ukaz operandi	Dec	Hex	Kontrolni signali, naslednji mikroukaz		T	F	
							00	00	fetch: addrsel=pc irload=1		01	01	
							01	01	pcload=1 pcsel=pc, opcode_jump		02	02	
0000:	7e01	63			1	main: li r1, 2	65	41	addrsel=pc dwrite=1 regsrc=databus, goto pcincr		84	84	
0001:	0002	Tak. operand						84	pcincr: pload=1 pcsel=pc, goto fetch		00	00	
0002:	7e02	63			2	li r2, -1	65	41	addrsel=pc dwrite=1 regsrc=databus, goto pcincr		84	84	
0003:	ffff	Tak. operand						84	pcincr: pload=1 pcsel=pc, goto fetch		00	00	
0004:	0089	0	2	1	1	loop: add r1,r1,r2	2	2	aluop=add op2sel=treg dwrite=1 regsrc=aluout, goto fetch		00	00	
0005:	5008	40			1	jnez r1, loop	40	2a	addrsel=pc imload=1		82	82	
0006:	0004	Tak. operand						82	aluop=sub op2sel=const0, if z then pcincr else jump		84	85	
								84	pcincr: pload=1 pcsel=pc, goto fetch		00	00	
								85	jump: pload=1 pcsel=immed, goto fetch		00	00	
0007:	8202	65			2	sw r2, 16	67	43	addrsel=pc imload=1		83	83	
0008:	0010	Tak. operand						83	addrsel=immed datawrite=1 datasel=dreg, goto pcincr		84	84	
								84	pcincr: pload=1 pcsel=pc, goto fetch		00	00	

LI ADD JNE Z
 • • •
 •
 •
 •
 •> ALU

Program: basic_program1.s :

ZBIRNIK
 LST
 RAM

```
main: li r1, 2          # r1 is the counter
      li r2, -1        # Used to decrement r1
loop: add r1, r1, r2   # r1<-r1+r2 (r2=-1 -> r1 decrements)
      jnez r1, loop   # if r1 != 0 then jump to loop:
      sw r2, 16       # Save r2 to MEM[16]
```

0000:	00007e01	0111111000000001	main: li r1, 2
0001:	00000002	0000000000000010	
0002:	00007e02	0111111000000010	li r2, -1
0003:	0000ffff	1111111111111111	
0004:	00000089	0000000010001001	loop: add r1, r1, r2
0005:	00005008	0101000000000100	jnez r1, loop
0006:	00000004	0000000000000100	
0007:	00008202	1000001000000010	sw r2, 16
0008:	00000010	0000000000010000	

NASLOV
 HEX
 BIN
 VSEBINA

```
00: 00002000 0101 # fetch: addrsel=pc irload=1
01: 00008000 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: 00000021 8485 # 82: aluop=sub op2sel=const0, if z then pcincr else jump
83: 001000c0 8484 # 83: addrsel=immed datawrite=1 datasel=dreg, goto pcincr
84: 00000800 0000 # pcincr: pload=1 pcsel=pc, goto fetch
85: 00000a00 0000 # jump: pload=1 pcsel=immed, goto fetch
```

M-NASLOV
 CONTROL
 DECISION
 USEBINA ROM

ROM

3.2.4 Mikro-zbirnik

kontrolni signali, nasl. mikroukaz

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



- Večbitni kontrolni signali:

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

micro_assembler.pl

```
#!/usr/bin/perl
use strict;
use warnings;

# Microassembler for Warren's 16-bit microcontrolled CPU.
# (c) GPL3 Warren Toomey, 2012

die("Usage: $0 inputfile\n") if (@ARGV!=1);

# Table of control ROM values for the
# known control=value pairs
my %Cvalue= (
    'aluop=add' => 0,
    'aluop=sub' => 1,
    'aluop=mul' => 2,
    'aluop=div' => 3,
    'aluop=rem' => 4,
    'aluop=and' => 5
```

Mikro-zbirnik

datoteka **basic_microcode.def**

```
fetch:  addrsel=pc irload=1          # Address=PC, Load IR register
        pload=1 ptsel=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: pload=1 ptsel=pc, goto fetch

jump:   pload=1 ptsel=immed, goto fetch
```

se prevede v

indexsel, pload

```
00: 00002000 0101 # fetch: addrsel=pc irload=1
01: 00008080 0202 #          pload=1 ptsel=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: pload=1 ptsel=pc, goto fetch
85: 00000a00 0000 # jump:   pload=1 ptsel=immed, goto fetch
```

3.2.5 Zbirnik

- Prevajanje programa v zbirniku:

- .\assembler.exe basic_program.s

- pripravi se datoteka, ki se vnese v RAM pomnilnik v MiMo model (Logisim):

- basic_program.ram**

- primer prevajanja v zbirniku ->

assembler.pl

```
#!/usr/bin/perl
use strict;
use warnings;

# Assembler for Warren's 16-bit microcontrolled CPU.
# (c) GPL3 Warren Toomey, 2012
# v0 : Original file
# v1 : Bug fixed: X processing option (11/2017)
#       few instructions' definitions changed
# v2 : corrected bug for swi,lwi ( from 'dX' to 'dsi' )

die("Usage: $0 inputfile\n") if (@ARGV!=1);

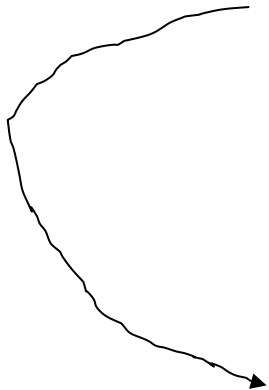
# Table of opcode names, the values
# and their arguments
# Meaning of abbreviations in %Opcode:
# D-reg           if ($atype eq 'd') {
# D-reg, S-reg is D-reg if ($atype eq 'D') {
# S-reg           if ($atype eq 's') {
# T-reg           if ($atype eq 't') {
# Absolute immediate if ($atype eq 'i') {
# Relative immediate if ($atype eq 'I') {
```

3.2.5 Zbirnik

■ 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
  
```



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		<u>jnez</u>	r1, loop
0009:	00000006	0000000000000010			
000a:	00008200	1000001000000000		<u>sw</u>	r0, 256
000b:	00000100	0000000100000000			
000c:	00005010	0101000000010000	inf:	<u>jnez</u>	r2, <u>inf</u>
000d:	0000000c	0000000000001000			

Op.koda	Treg	Sreg	Dreg
7	3	3	3