

Gradnja enostavnega procesorja MIPS

Digitalna vezja

Miha Moškon

miha.moskon@fri.uni-lj.si

<https://fri.uni-lj.si/en/about-faculty/employees/miha-moskon>

Procesor MIPS16

Lotili se bomo izvedbe enostavnega procesorja MIPS s 16-bitnimi ukazi in 16-bitnimi pomnilniškimi besedami

MIPS (Microprocessor without Interlocked Pipelined Stages)

Družina RISC (Reduced Instruction Set Computer)

Bolj znane 32- in 64-bitne izvedbe

Procesor MIPS16

Prilagoditev (poenostavitev) 32-bitne arhitekture na 16-bitno

Izvedba ukazov v eni urini periodi (single-cycle processor)

Ukazi ne gredo čez stopnje, urina perioda mora biti prilagojena najpočasnejšemu ukazu

Harvardov model: ukazni pomnilnik je ločen od podatkovnega; pisanje v ukazni pomnilnik bo onemogočeno

Ukazi

Procesor bo podpiral sledeče ukaze

add

sub

and

or

beq

bne

lw

sw

addi

subi

Ukazi: 2 formata ukazov

I format: 2 registra in konstanta

op. code					RS			RT			value/offset				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

R format: 3 registri

op. code					RS			RT			RD				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

I format ukazov

2 registra in konstanta (operand)

op. code					RS			RT			(unsigned) value				
1	0	1	0	x											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

addi \$RT, \$RS, value

$\$RT \leftarrow \$RS + \text{value}$

subi \$RT, \$RS, value

$\$RT \leftarrow \$RS - \text{value}$

I format ukazov

2 registra in konstanta (odmik)

op. code					RS			RT			offset				
1	0	0	0	x											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

lw \$RT, offset(\$RS)

sw \$RT, offset(\$RS)

$\$RT \leftarrow \text{MEM}[\$RS + \text{offset}]$

$\text{MEM}[\$RS + \text{offset}] \leftarrow \RT

I format ukazov

2 registra in konstanta (odmik)

op. code					RS			RT			offset				
0	1	0	0	0											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

beq \$RT, \$RS, offset

if \$RS == \$RT:

$\$PC \leftarrow \$PC + \text{offset}$

else:

$\$PC \leftarrow \$PC + 1$

I format ukazov

2 registra in konstanta (odmik)

op. code					RS			RT			offset				
0	1	0	0	1											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

bne \$RT, \$RS, offset

if \$RS != \$RT:

$\$PC \leftarrow \$PC + \text{offset}$

else:

$\$PC \leftarrow \$PC + 1$

R format ukazov

3 registri

op. code					RS			RT			RD				
0	0	0	x	x											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

add \$RD, \$RS, \$RT

$\$RD \leftarrow \$RS + \$RT$

sub \$RD, \$RS, \$RT

$\$RD \leftarrow \$RS - \$RT$

and \$RD, \$RS, \$RT

$\$RD \leftarrow \$RS \& \$RT$

or \$RD, \$RS, \$RT

$\$RD \leftarrow \$RS | \$RT$

Ukazi: Operacijske kode

	15	14	13	12	11
ukaz	op. code				
add	0	0	0	0	0
sub	0	0	0	0	1
and	0	0	0	1	0
or	0	0	0	1	1
beq	0	1	0	0	0
bne	0	1	0	0	1
lw	1	0	0	0	0
sw	1	0	0	0	1
addi	1	0	1	0	0
subi	1	0	1	0	1

Komponente procesorja

Ukazni pomnilnik (ROM)

Programski števec (Program Counter)

Kontrolna enota (Control Unit)

Registri

Aritmetično-logična enota (Arithmetic logic unit)

Podatkovni pomnilnik (RAM)

Ostalo: multiplekserji, logična vrata, dodaten seštevalnik (za skoke)...

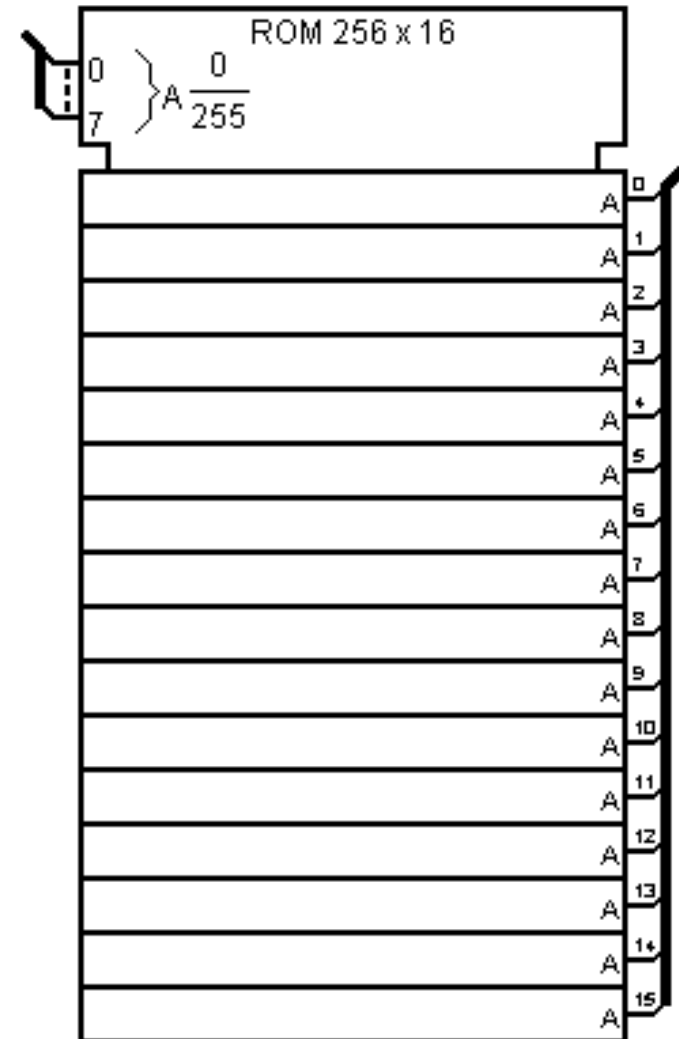
Ukazni pomnilnik (ROM 256 x 16)

Vhod:

- Address (8 bit)

Izhod:

- Data (16 bit)



Podatkovni pomnilnik (RAM 64k x 16)

Vhodi:

- Address (16 bit)
- Store (Write Enable)
- Load (Output Enable)
- Clock (pisanje na pozitivno fronto ure)
- Input (DATA_in, 16 bit)

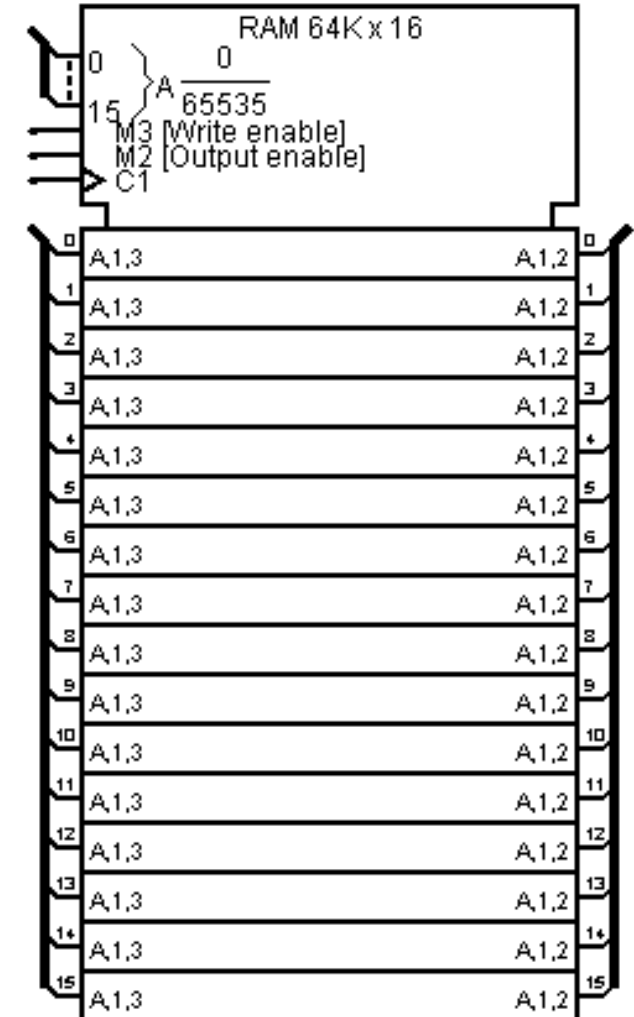
Izhodi

- Output (DATA_out, 16 bit)

Dodatno

Asynchronous read:

Yes



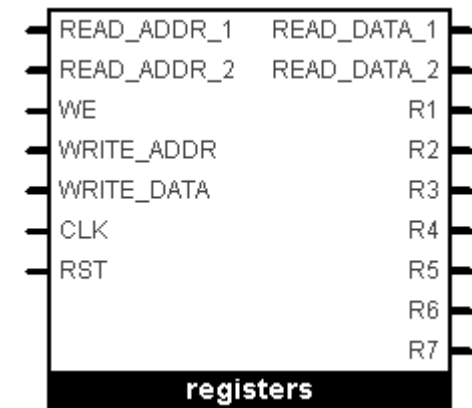
Polje registrov (8 registrov)

Vhodi:

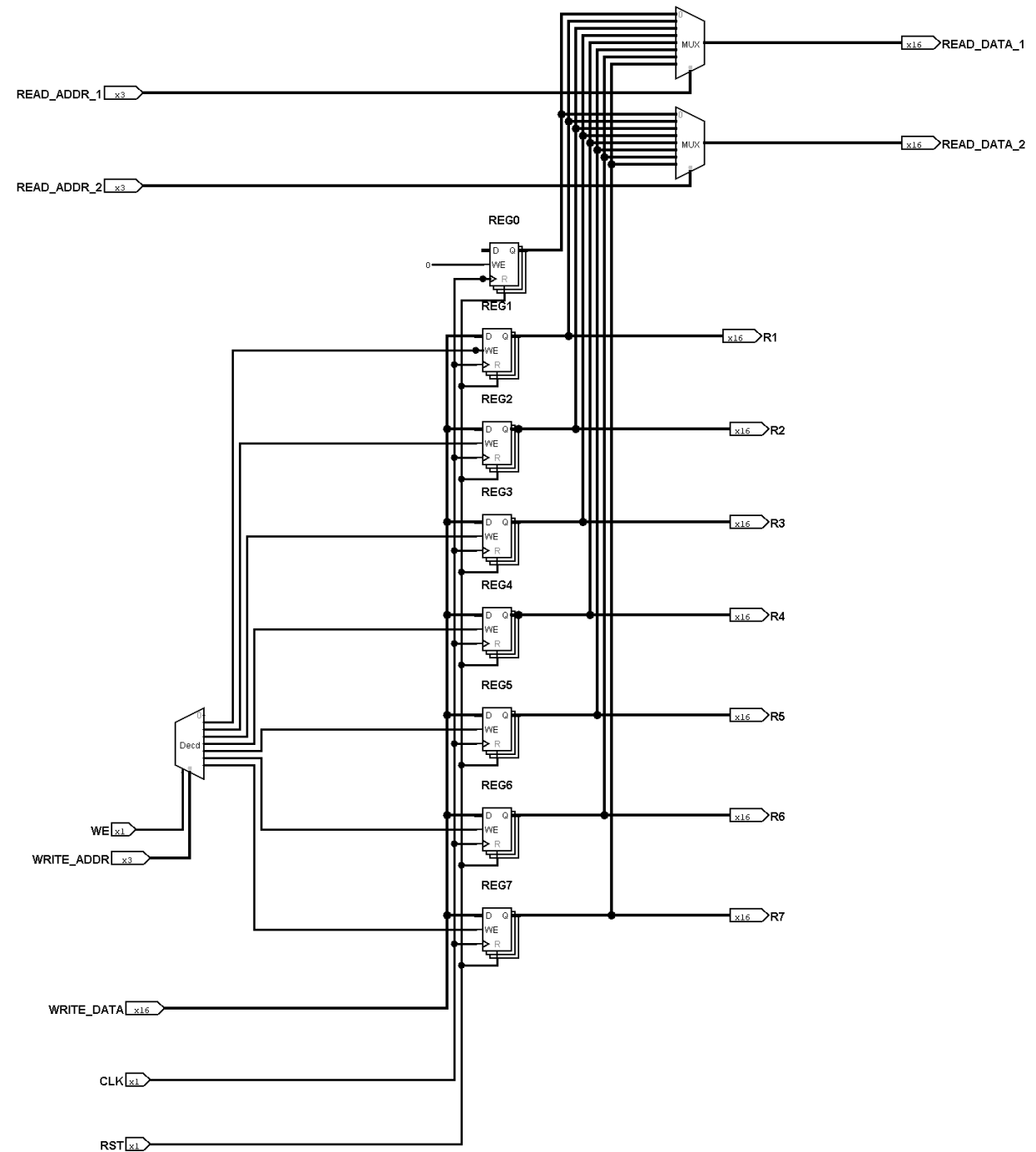
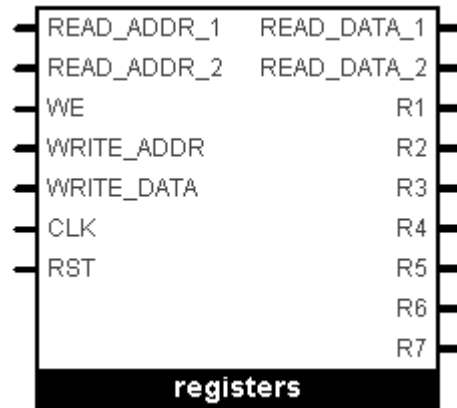
- READ_ADDR_1, READ_ADDR_2 (naslov za branje, 3 bit)
- WE (1 bit)
- WRITE_ADDR (naslov za pisanje, 3 bit)
- WRITE_DATA (podatki za pisanje)
- CLK
- RST

Izhodi:

- READ_DATA_1, READ_DATA_2 (prebrani podatki, 16 bit)
- R1...R7 (debug izhodi, 16 bit)



Polje registrov



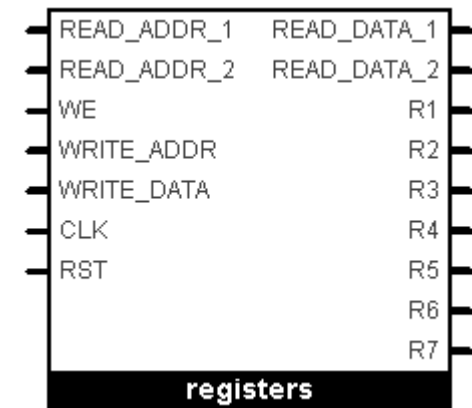
Polje registrov (8 registrov)

Vhodi:

- READ_ADDR_1, READ_ADDR_2 (naslov za branje, 3 bit)
- WE (1 bit)
- WRITE_ADDR (naslov za pisanje, 3 bit)
- WRITE_DATA (podatki za pisanje)
- CLK
- RST

Izhodi:

- READ_DATA_1, READ_DATA_2 (prebrani podatki, 16 bit)
- R1...R7 (debug izhodi, 16 bit)



Aritmetično logična enota

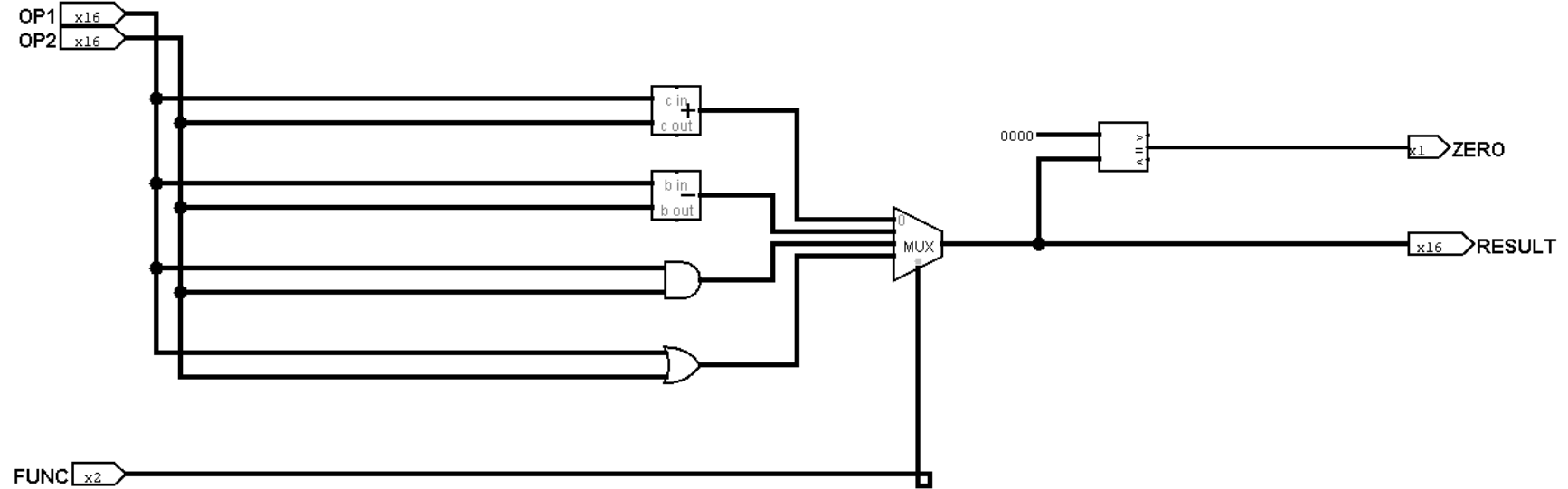


Vhodi:

- operand1 (16 bit)
- operand2 (16 bit)
- func (2 bit):
 - 00 – ADD
 - 01 – SUB
 - 10 – AND
 - 11 – OR

Izhodi:

- result (16 bit)
- zero (1 bit)



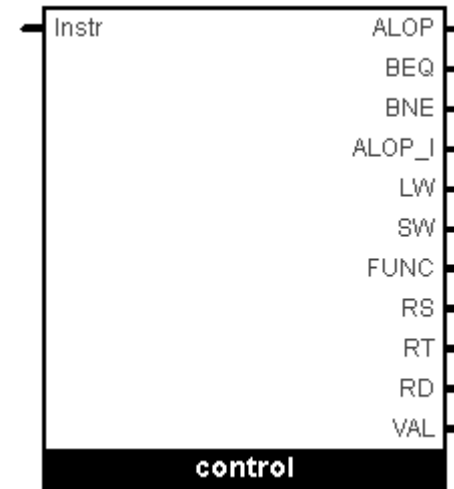
Kontrolna enota

Vhod:

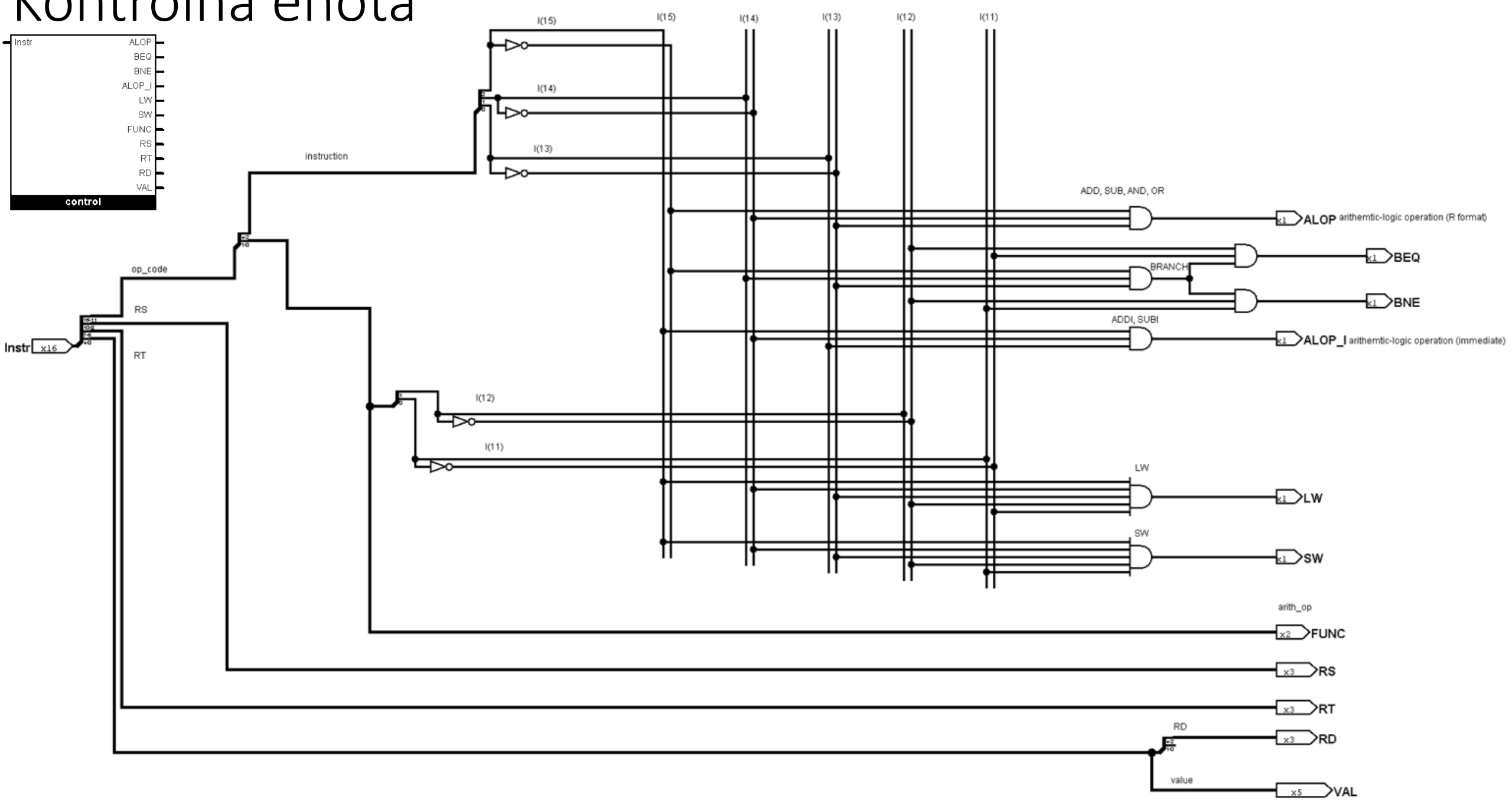
- Instr (16 bit)

Izhodi:

- ALOP, BEQ, BNE, ALOP_I, LW, SW (ukaz / tip ukaza, 1 bit)
- FUNC (AL funkcija, 2 bit)
- RS, RT, RD (naslovi registrov, 3 biti)
- VAL (value/offset, 5 bitov)



Kontrolna enota

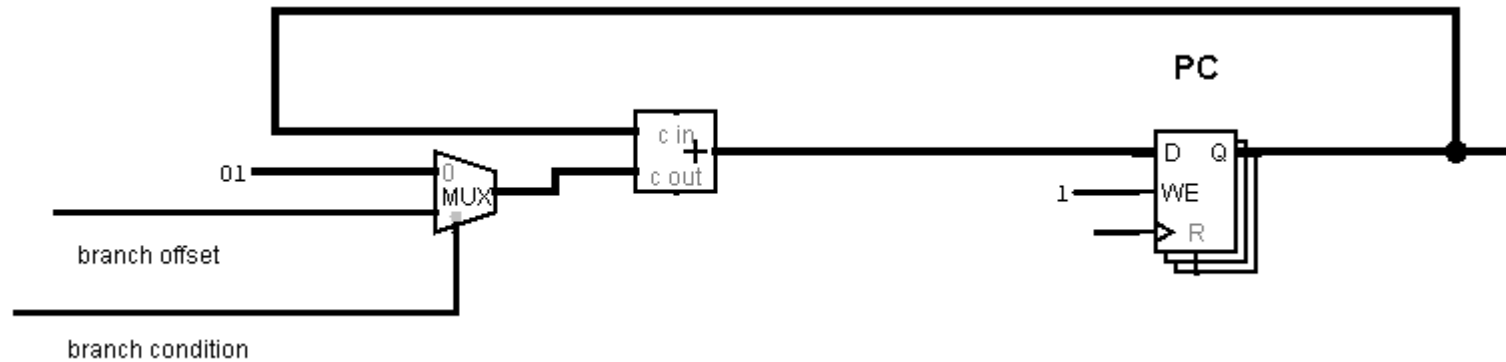


Programski števec

8 bitni števec

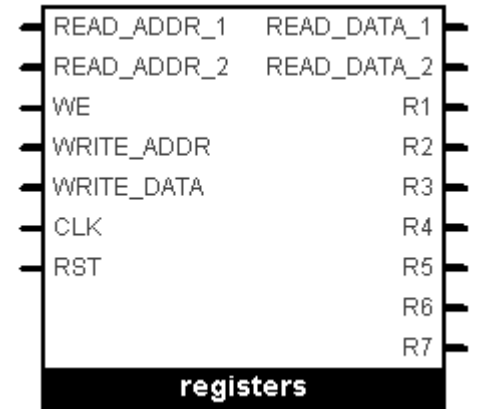
Delovanje:

- $PC \leftarrow PC + \text{offset}$ if branch else $PC + 1$



Povezovanje – registri

- $\text{READ_ADDR_1} \leftarrow \text{control_RS}$
- $\text{READ_ADDR_2} \leftarrow \text{control_RT}$
- $\text{WE} \leftarrow \text{control_ALOP or control_ALOP_I or control_LW}$
- $\text{WRITE_ADDR} \leftarrow \text{control_RD if (control_ALOP = 1) else control_RT}$
- $\text{WRITE_DATA} \leftarrow \text{RAM_output if (control_LW = 1) else ALU_results}$
- $\text{CLK} \leftarrow \text{CLK}$
- $\text{RST} \leftarrow \text{RST}$



Povezovanje – ALU



operand1 $\leftarrow$ registers_READ_DATA_1 (register RS)

operand2 $\leftarrow$ registers_READ_DATA_2 if (control_ALOP or control_BEQ or control_BNE) else extended_val

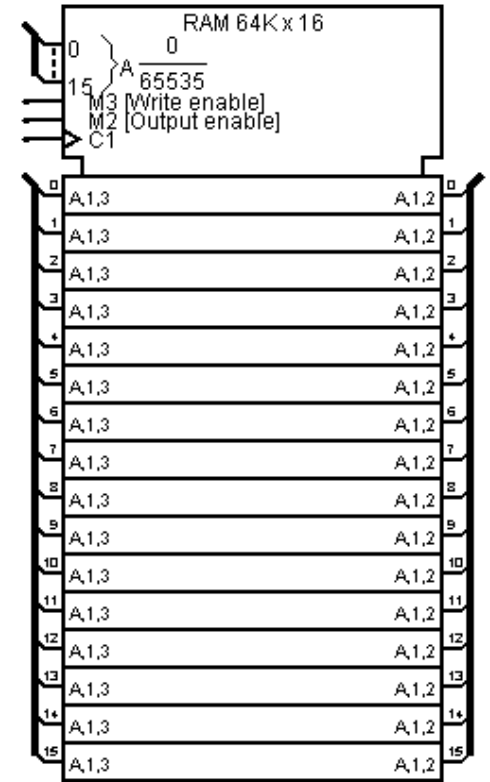
- extended_val $\leftarrow$ 00...00 & control_VAL

FUNC $\leftarrow$ 00 if (control_LW or control_SW) else 01 if (control_BEQ or control_BNE) else control_FUNC

- 00 = ADD
- 01 = SUB

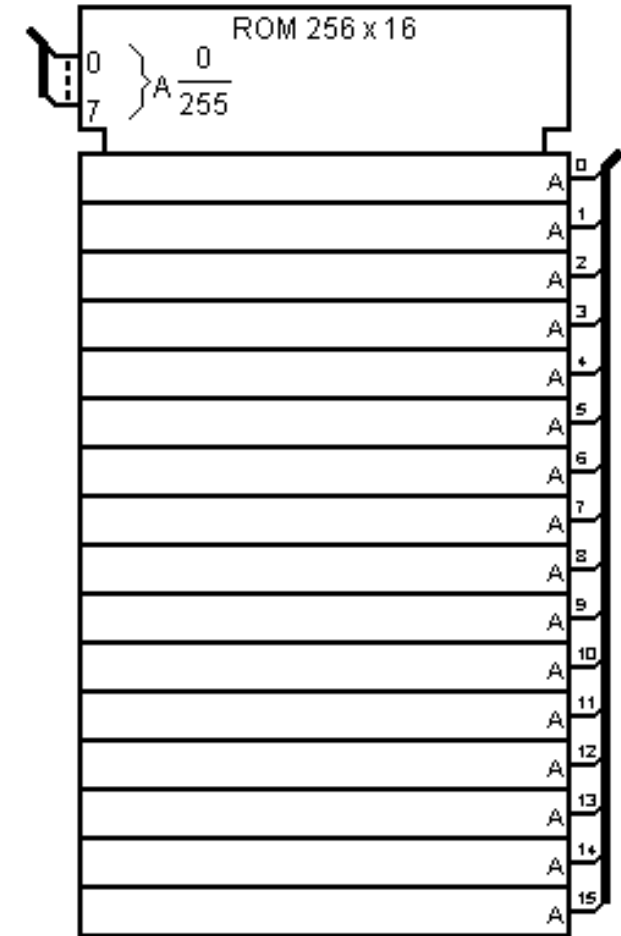
Povezovanje – RAM

- Address $\leftarrow$ ALU_result
- Store $\leftarrow$ control_SW
- Load $\leftarrow$ control_LW
- Clock $\leftarrow$ Clock
- Input $\leftarrow$ registers_READ_DATA_2 (prvi register je za naslov)



Povezovanje – ROM

- Address $\leftarrow$ PC



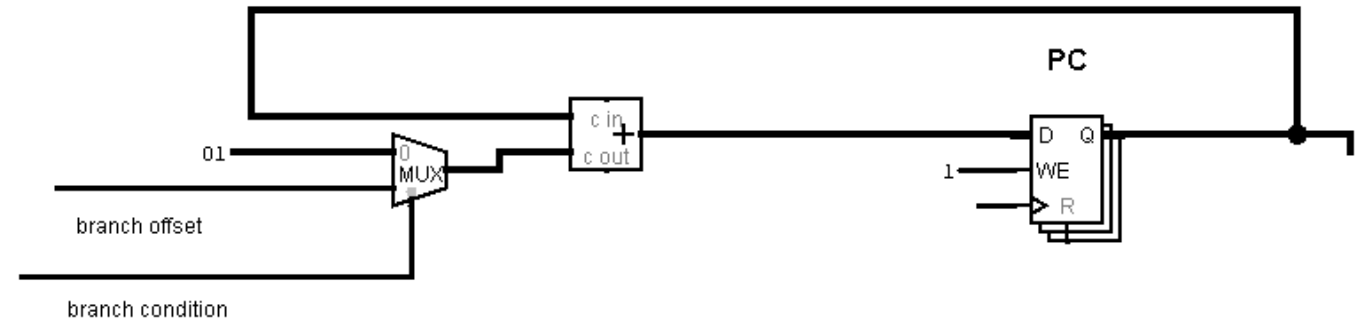
Povezovanje – PC

$WE \leftarrow 1$

$D \leftarrow$ izhod PC seštevalnika

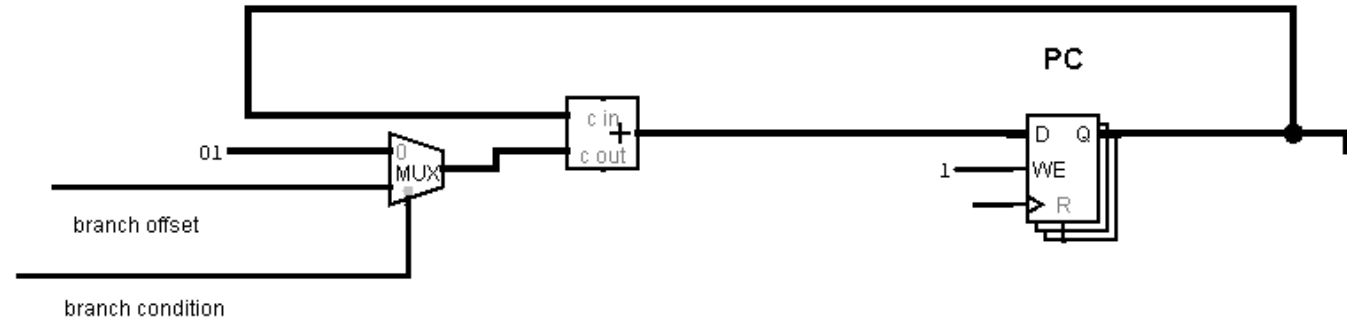
$Clock \leftarrow Clock$

$Reset \leftarrow Reset$



Povezovanje – PC seštevalnik

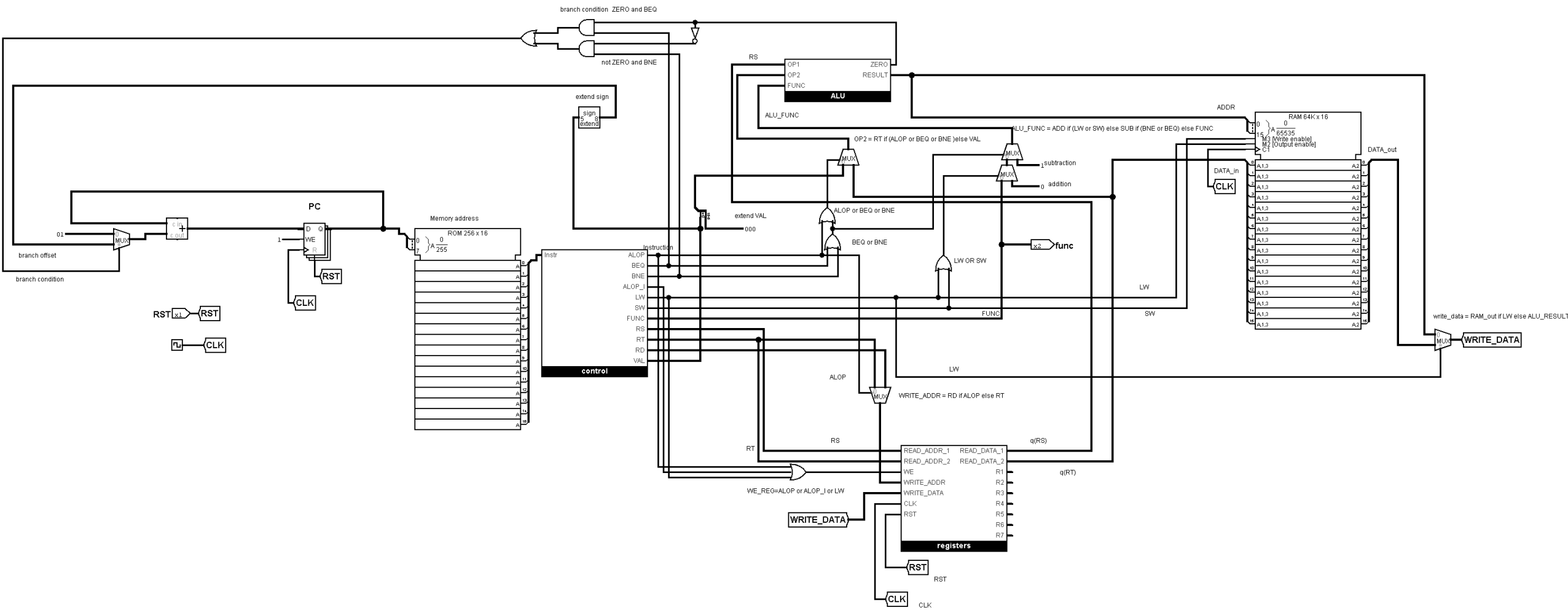
Input1 $\leftarrow$ PC



Input2 $\leftarrow$ 0x01 if (branch_condition = 0) else branch_offset

- branch_condition $\leftarrow$ (control_BEQ and ALU_zero) or (control_BNE and not ALU_zero)
- branch_offset $\leftarrow$ extended_sign(control_val)

Celotna izvedba procesorja



Celotna izvedba procesorja

Glej <https://github.com/mmoskon/MIPS16>