Tanul Gabriel-Stefan-30222

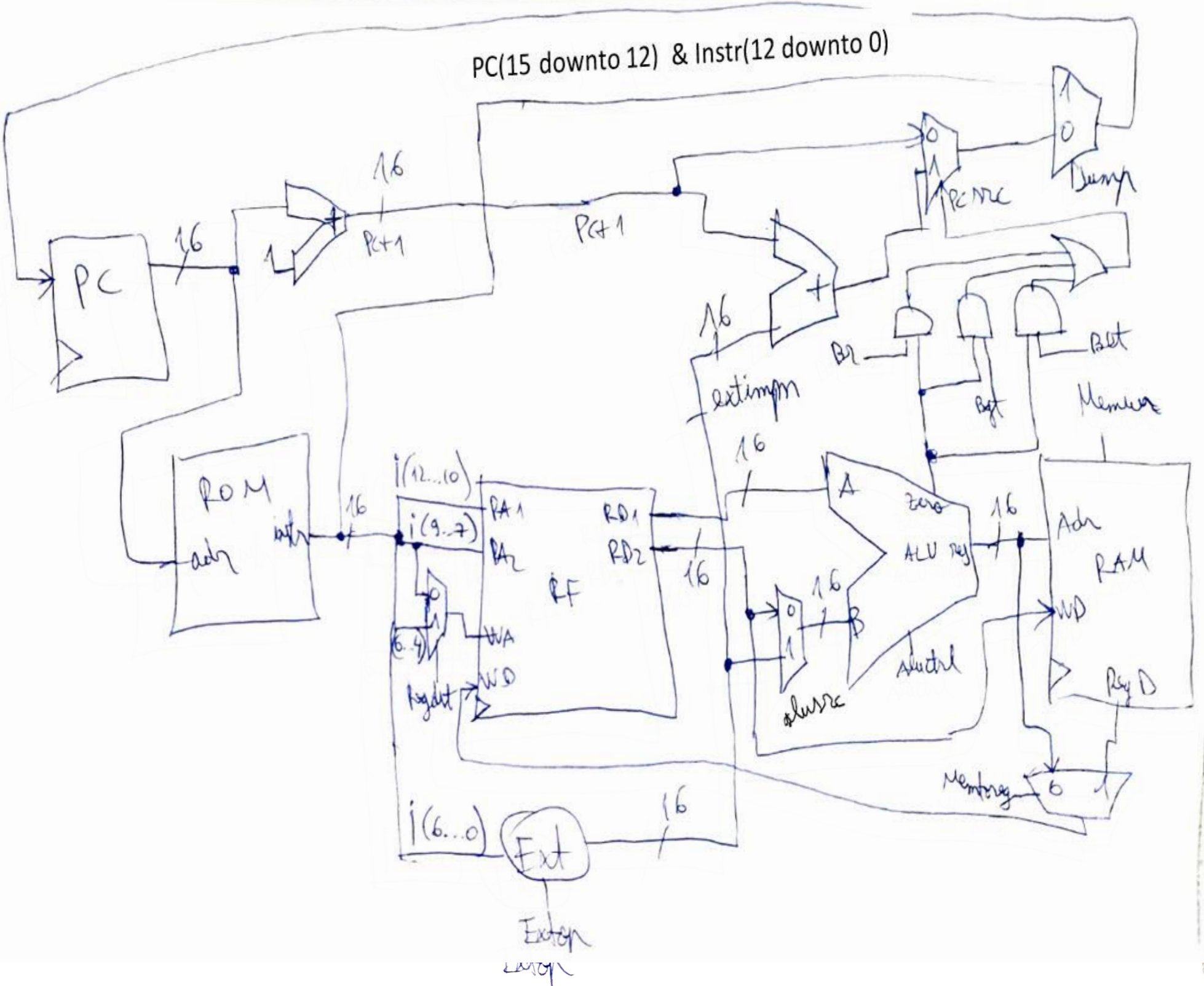
Semnale control MIPS16 pentru Anexa 5

Opcode 000 - tip R
Func 000 – xnor
001 – add
010 - sub
011 – sll
100 - srl
101 – and
110 - or
111 – xor

tip I
Opcode:
001 – lw
010 - sw
011 - beq
100 - bgt
101 – blt
110 - addi

tip J
Opcode:
111 - jump

Instructiune	**Opcode** *Instr(15-13)*	**RegDst**	**ExtOp**	**ALUSrc**	**Branch**	**Blt**	**Bgt**	**Jump**	**Mem Write**	**Memto Reg**	**Reg Write**	**func** *Instr(2-0)*
add	000	1	0	0	0	0	0	0	0	0	1	001
sub	000	1	0	0	0	0	0	0	0	0	1	010
sll	000	0	1	1	0	0	0	0	0	0	1	011
srl	000	0	1	1	0	0	0	0	0	0	1	100
and	000	1	0	0	0	0	0	0	0	0	1	101
or	000	1	0	0	0	0	0	0	0	0	1	110
xor	000	1	0	0	0	0	0	0	0	0	1	111
xnor	000	1	0	0	0	0	0	0	0	0	1	000
lw	001	1	1	1	0	0	0	0	0	1	1	---
sw	010	0	1	0	0	0	0	0	1	0	0	---
beq	011	0	1	0	1	0	0	0	0	0	0	---
bgt	100	0	1	0	0	0	1	0	0	0	0	---
blt	101	0	1	0	0	1	0	0	0	0	0	---
addi	110	0	1	1	0	0	0	0	0	0	1	---
Jump	111	0	0	0	0	0	0	1	0	0	0	---


PC(15 downto 12) & Instr(12 downto 0)
PC
16
PC+1
PC+1
PCSrc
Jump
16
extimm
Br
Bne
Bgt
Memtoreg
ROM
adr
instr
16
i(12...10)
i(9...7)
RA1
RA2
RF
RD1
RD2
16
A
Zero
ALU
16
Adr
RAM
WD
RegD
16
16
ALUSrc
ALUctrl
i(6...0)
Ext
16
ExtOp
Memtoreg

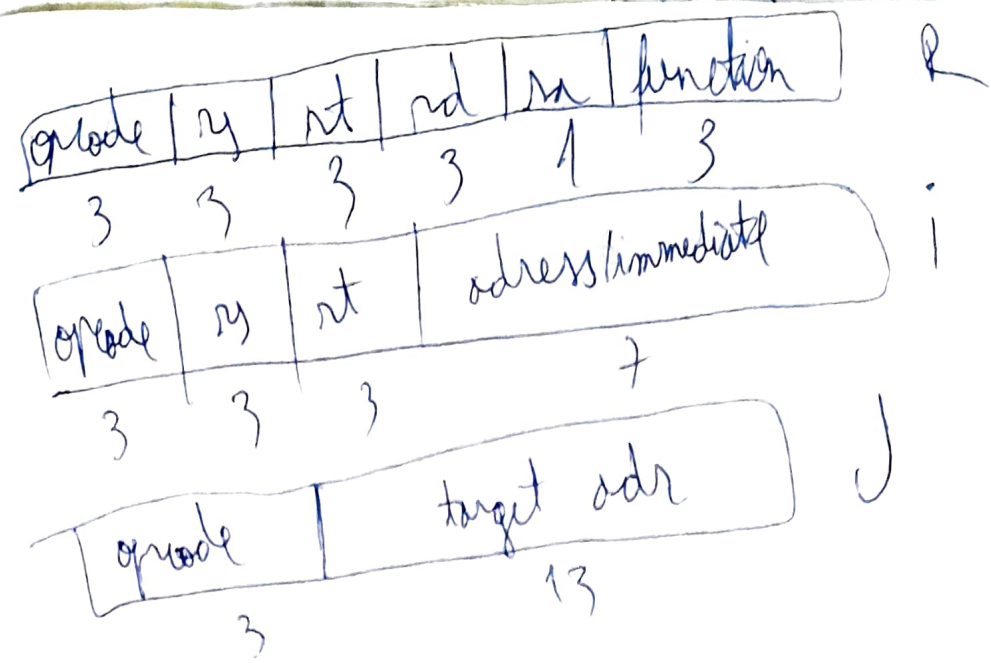
Împărțire

21/5

```
0 lw $5, 1 mem(3)
1 lw $1, 21 mem(1)
2 lw $2, 5 , mem(2)
3 beq $2, $1, jmp (7) (nr. egale, câtul 1)
4   sub $1, $1, $2 -- diferența deîmpărțit-împărțitor
5   addi $5, $0, 1 -- incrementare cât
6   bgt $1, $2, jmp (4)
7 sw mem(0), $1 -- restul
8 sw mem(9), $5, -- câtul
9 jmp out (adresă mare 2^13)
```

21 − 5 = 16	c = 1 r = 16
16 − 5 = 11	c = 2 r = 11
11 − 5 = 6	c = 3 r = 6
6 − 5 = 1	c = 4 > r = 1
1 < 5, final	

instr de tip R	addition	add
	subtraction	sub
	shift left logical (ca sa)	sll
	shift right logical	srl
	logical AND	and
	logical OR	or
	logical XOR	xor
	logical XNOR	xnor
instr de tip I	add immediate	addi
	load word	lw
	store word	sw
	branch on equal	beq
	branch on greater than	bgt
	branch on less than	blt
instr. de tip J		jump j



add

Descriere	Adună 2 reg și pune în al 3-lea
Operație	$d ← $s + $t; PC ← PC + 2
Sintaxă	add $d, $s, $t
Format	000_sss_ttt_ddd_0_001

add $2, $3, $4

B"000_011_100_010_0_001

sule

Descriere	Scade 2 reg și pune în al 3-lea
Operație	$d ← $s - $t; PC ← PC + 2
Sintaxă	sule $d, $s, $t
Format	000_sss_ttt_ddd_0_010

sule $3, $4, $5

B"000_100_101_011_0_010

sll

Descriere	Deplasează la stânga cu sa
Operație	$d ← $t << sa; PC ← PC + 2
Sintaxă	sll $d, $t, sa
Format	000_sss_ttt_ddd_sa_011

sll $5, $3, 1

000_000_011_101_1_011

srl

Descriere	Deplasare la dreapta cu sa
Operație	$d ← $t >> sa
Sintaxă	srl $d, $t, sa
format	000_111_ttt_ddd_sa_100

srl $6, $5, 1
000_000_101_110_1 100

and

Descriere	AND logic pe 2 reg; mem ul 3-lea
Operație	$d ← $s & $t; PC ← PC+2
Sintaxă	and $d, $s, $t
Format	000_sss_ttt_ddd_0_101

and $3, $1, $2
000_001_010_011_0_101

or

Descriere	sau logic pe 2 reg, mem ul 3-lea
Operație	$d ← $s | $t, PC ← PC+2
Sintaxă	or $d, $s, $t
format	000_sss_ttt_ddd_0_110

or $4, $1, $5
000_001_101_100_0_110

xor

Descriere	xor logic pe 2 reg, mem ul 3-lea
Operație	$d ← $s ^ $t, PC ← PC+2
Sintaxă	xor $d, $s, $t
Format	000_sss_ttt_ddd_0_111

xor $5, $1, $2
000_001_010_101_0_111

xnor

Descriere	xnor logic pe 2 reg, mem ul 3-lea
Operație	$d ← $s xnor $t, PC ← PC+2
Sintaxă	xnor $d, $s, $t
Format	000_sss_ttt_ddd_0_000

xnor $6, $5, $4
000_101_100_110_0_000

addi

addi $3, $4, 5
110_100-011-0000101

Descriere	adunare imediată cu o Const
Operație	$t ← $s + imm; PC ← PC+2
Sintaxă	addi $t, $s, imm
Format	110_sss_ttt_iiiiiii

lw

lw $5, offset($6)
001_110_101_xxxxxxx

Descriere	un cuvânt încărcat în reg
Operație	$t ← MEM[$s+offset]; PC ← PC+2
Sintaxă	lw $t, offset($s)
Format	001_sss_ttt_iiiiiii

sw

sw offset($6), $5
010_101_110_xxxxxxx

Descriere	un reg stocat în memorie
Operație	MEM[$t+offset] ← $s; PC ← PC+2
Sintaxă	sw offset($t), $s
Format	010_sss_ttt_iiiiiii

beq

Descriere	Salt condiționat dacă este = Rotire 2 Reg
Operație	if $s == $t then PC ← PC+2+(offset<<1) else PC ← PC+2;
Sintaxă	beq $s, $t, offset
Format	011_sss_ttt_iiiiiii

beq $6, $7, offset
011_110_111_offset

bgt

Descriere	Salt cond. dacă reg a mai mare reg b
Operație	if (\$s > \$t) then PC ← PC+2+(offset<<1) else PC ← PC+2;
Sintaxă	bgt \$s, \$t, offset
Format	100-ssss-tttt-iiiiiiii

bgt \$5, \$7, offset
100-101-111-offset

blt

Descriere	Salt condiționat dacă reg a mai mic decât b
Operație	if (\$s < \$t) then PC ← PC+2+offset(<<1) else PC ← PC+2;
Sintaxă	blt \$5, \$t, offset
Format	101-sss1-~~tttt~~-iiiiiiii

blt \$6, \$7, offset
101-110-111-offset

j

Descriere	Salt la adresă
Operație	PC ← (PC+2) & 0xf0000000 | (target<<1)
Sintaxă	j target
Format	111-iiiiiiiiiiiiii

j offset
111-offset