MINISTERUL EDUCAȚIEI ȘI CERCETĂRII
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Procesorul MIPS, ciclu unic – versiune pe 16 biți –

Raport de activitate

Procesorul MIPS, ciclu unic – versiune pe 16 biți

Student: Lazea Dragoș-Bogdan
CTI-Română, Seria A, Grupa 30223

An academic 2020-2021

a. Instrucțiuni alese suplimentar

i. Instrucțiunea xor (Sau-exclusiv)

- Instrucțiune de tip R – operațiile se efectuează asupra conținutului unor registre;
- Realizează operația de sau-exclusiv între conținutul a două registre;
- Sintaxa: `xor $rd, $rs, $rt`
- Formatul instrucțiunii:

opcode	rs	rt	rd	sa	func
000	sss	ttt	ddd	0	110

- RTL abstract: **RF[rd] ← RF[rs] ^ RF[rt]**;

ii. Instrucțiunea slt (Set on Less Than)

- Instrucțiune de tip R – operațiile se efectuează asupra conținutului unor registre;
- Setează registrul destinație atunci când conținutul primului registru sursă este mai mic decât conținutul celui de-al doilea registru sursă;
- Sintaxa: `slt $rd, $rs, $rt`
- Formatul instrucțiunii:

opcode	rs	rt	rd	sa	func
000	sss	ttt	ddd	0	111

- RTL abstract: **If (RF[rs] < RF[rt]) then RF[rd] ← 1**
  Else RF[rd] ← 0

iii. Instrucțiunea bgez (Branch on Greater Than or Equal to Zero)

- Instrucțiune de tip I – operație între conținutul unui registru și o valoare imediată;
- Efectuează un salt condiționat la o adresă relativă la adresa instrucțiunii următoare dacă registrul sursă are conținutul mai mare sau egal cu zero;
- Sintaxa: **`bgez $rs, offset`**
- Formatul instrucțiunii:

opcode	rs	rt	imm
101	sss	000	iiiiii

- RTL abstract: **If (RF[rs] >= 0) then PC ← PC + 2 + (offset << 1)**
  Else PC ← PC + 2

iv. Instrucțiunea bltz (Branch on Less Than Zero)

- Instrucțiune de tip I – operație între conținutul unui registru și o valoare imediată;
- Efectuează un salt condiționat la o adresă relativă la adresa instrucțiunii următoare dacă registrul sursă are conținutul mai mic decât zero;
- Sintaxa: **`bltz $rs, offset`**
- Formatul instrucțiunii:

opcode	rs	rt	imm
110	sss	000	iiiiii

- RTL abstract: **If (RF[rs] < 0) then PC ← PC + 2 + (offset << 1)**
  Else PC ← PC + 2

b. Tabel cu valorile semnalelor de control pentru setul de instrucțiuni selectat

Instrucțiune	Reg Dst	Reg Write	ALU Src	ALU Ctrl	Ext Op	Mem Write	Memto Reg	Slt	Branch	Bgez	Bltz	Jump
add	1	1	0	000 (+)	X	0	0	0	0	0	0	0
sub	1	1	0	001 (-)	X	0	0	0	0	0	0	0
sll	1	1	0	010 (<<)	X	0	0	0	0	0	0	0
srl	1	1	0	011 (>>)	X	0	0	0	0	0	0	0
and	1	1	0	100 (and)	X	0	0	0	0	0	0	0
or	1	1	0	101 (or)	X	0	0	0	0	0	0	0
xor	1	1	0	110 (xor)	X	0	0	0	0	0	0	0
slt	1	1	0	111 (cmp)	X	0	0	1	0	0	0	0
addi	0	1	1	000 (+)	1	0	0	0	0	0	0	0
lw	0	1	1	000 (+)	1	0	1	0	0	0	0	0
sw	X	0	1	000 (+)	1	1	X	0	0	0	0	0
beq	X	0	0	001 (-)	1	0	X	0	1	0	0	0
bgez	X	0	0	001 (-)	1	0	X	0	0	1	0	0
bltz	X	0	0	001 (-)	1	0	X	0	0	0	1	0
j	X	0	X	XXX	X	0	X	0	0	0	0	1

Observații:

(+) – în ALU are loc o operație de adunare
(–) – în ALU are loc o operație de scădere
(<<) – în ALU are loc o deplasare logică la stânga cu o poziție
(>>) – în ALU are loc o deplasare logică la dreapta cu o poziție
(and) – în ALU are loc o operație de și-logic
(or) – în ALU are loc o operație de sau-logic
(xor) – în ALU are loc o operație de sau-exclusiv
(cmp) – în ALU are loc o operație de comparare (folosită doar în cazul instrucțiunii *Set on Less Than*)

c. Descrierea în cuvinte, cod C și cod mașină a programului încărcat în memoria ROM

Programul ales pentru exemplificarea funcționării procesorului având setul de instrucțiuni mai sus menționat este reprezentat de algoritmul de determinare a maximului dintr-un șir de 10 numere întregi, aflat la adresa **A_addr = 0** în memoria de date, și scrierea valorii determinate la adresa **max_addr = 20**. De asemenea, se presupune că programul începe de la adresa 0 în memoria de instrucțiuni.

Algoritmul se bazează pe o simplă parcurgere a șirului de numere întregi, cu actualizarea la fiecare pas a valorii maxime determinate până la iterația curentă. Secvența de cod C corespunzătoare programului ales este prezentată în figura de mai jos, partea stângă (figura de sus), iar codul mașină corespunzător programului se regăsește în figura din stânga-jos.

Pentru o mai bună înțelegere a modului de funcționare al programului reținut în memoria de instrucțiuni a procesorului, codul C a fost rescris, utilizând etichete și instrucțiuni de salt, astfel încât acesta să reflecte cât mai clar instrucțiunile programului scris în limbaj de asamblare (figura din dreapta):

```c
maxim = A[0];
for (int i = 0; i < 10; i++) {
    if (A[i] > maxim)
        maxim = A[i];
}
return maxim;
```

Codul mașină corespunzător programului este:

```
B"000_000_000_001_0_000",       -- add $1, $0, $0       #0010
B"001_000_100_0001010",         -- addi $4, $0, 10      #220A
B"000_000_000_010_0_000",       -- add $2, $0, $0       #0020
B"010_010_101_0000000",         -- lw $5, 0($2)         #4A80
B"100_001_100_0000111",         -- beq $1, $4, 7        #8307
B"010_010_011_0000000",         -- lw $3, 0($2)         #4980
B"000_101_011_110_0_001",       -- sub $6, $5, $3       #15E1
B"101_110_000_0000001",         -- bgez $6, 1           #B801
B"000_000_011_101_0_000",       -- add $5, $0, $3       #01D0
B"001_010_010_0000010",         -- addi $2, $2, 2       #2902
B"001_001_001_0000001",         -- addi $1, $1, 1       #2401
B"111_0000000000100",           -- j 4                  #E004
B"011_000_101_0010100",         -- sw $5, 20($0)        #6294
```

```c
int RF[8]; // register-file

void array_max(int A[], int* maxim) {
    RF[0] = 0;
    RF[1] = RF[0] + RF[0];              // add $1, $0, $0
    RF[4] = RF[0] + 10;                 // addi $4, $0, 10
    RF[2] = RF[0] + RF[0];              // add $2, $0, $0
    RF[5] = A[RF[2]];                   // lw $5, A_addr($2)
    begin_loop:
        if (RF[1] == RF[4])             // begin_loop: beq $1, $4, end_loop
            goto end_loop;
        RF[3] = A[RF[2]];               // lw $3, A_addr($2)
        RF[6] = RF[5] - RF[3];          // sub $6, $5, $3
        if (RF[6] >= 0)
            goto label_next;            // bgez $6, label_next
        RF[5] = RF[0] + RF[3];          // add $5, $0, $3
    label_next: RF[2] = RF[2] + 1;      // label_next: addi $2, $2, 2
        RF[1] = RF[1] + 1;              // addi $1, $1, 1
        goto begin_loop;                // j begin_loop
    end_loop: *maxim = RF[5];           // end_loop: sw $5, max_addr($0)
}
```

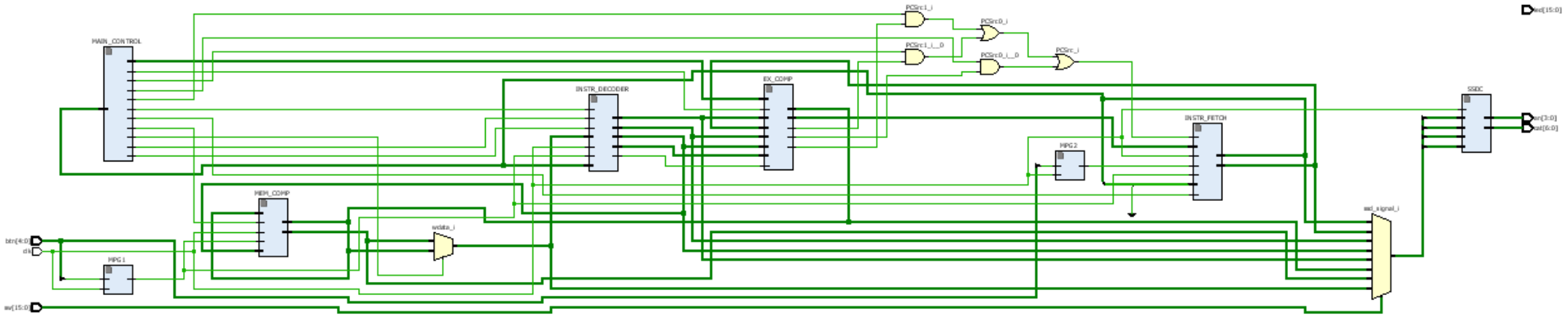
d. Trasarea execuției programului

Presupunem că șirul încărcat în memorie este:

```
A = (X"000A", X"0001", X"0110", X"B001", X"0C60", X"1743", X"3F10", X"2008", X"0403", X"6005")

0       add $1, $0, $0                  -- RD1 = 0, RD2 = 0, ALURes = 0, Zero = 1, Sign = 0
1       addi $4, $0, 10                 -- RD1 = 0, Ext_Imm = 10, ALURes = 10, Zero = 0, Sign = 0
2       add $2, $0, $0                  -- RD1 = 0, RD2 = 0, ALURes = 0, Zero = 1, Sign = 0
3       lw $5, 0($2)                    -- r_data = X"000A"
4       beq $1, $4, 7                   -- Zero = 0 deci nu se efectuează saltul
5       lw $3, 0($2)                    -- r_data = X"000A"
6       sub $6, $5, $3                  -- RD1 = X"000A", RD2 = X"000A", ALURes = 0, Zero = 1, Sign = 0
7       bgez $6, 1                      -- Sign = 0 deci se sare peste o instrucțiune
8       add $5, $0, $3                  -- Nu se execută în prima iterație a buclei
9       addi $2, $2, 2                  -- RD1 = 0, Ext_Imm = 2, ALURes = 2, Zero = 0, Sign = 0
10      addi $1, $1, 1                  -- Rd1 = 0, Ext_Imm = 1, ALURes = 1, Zero = 0, Sign = 0
11      j 4                             -- Se efectuează salt la adresa 4 din memorie
12      sw $5, 20($0)                   -- Nu se efectuează după prima iterație a buclei
```

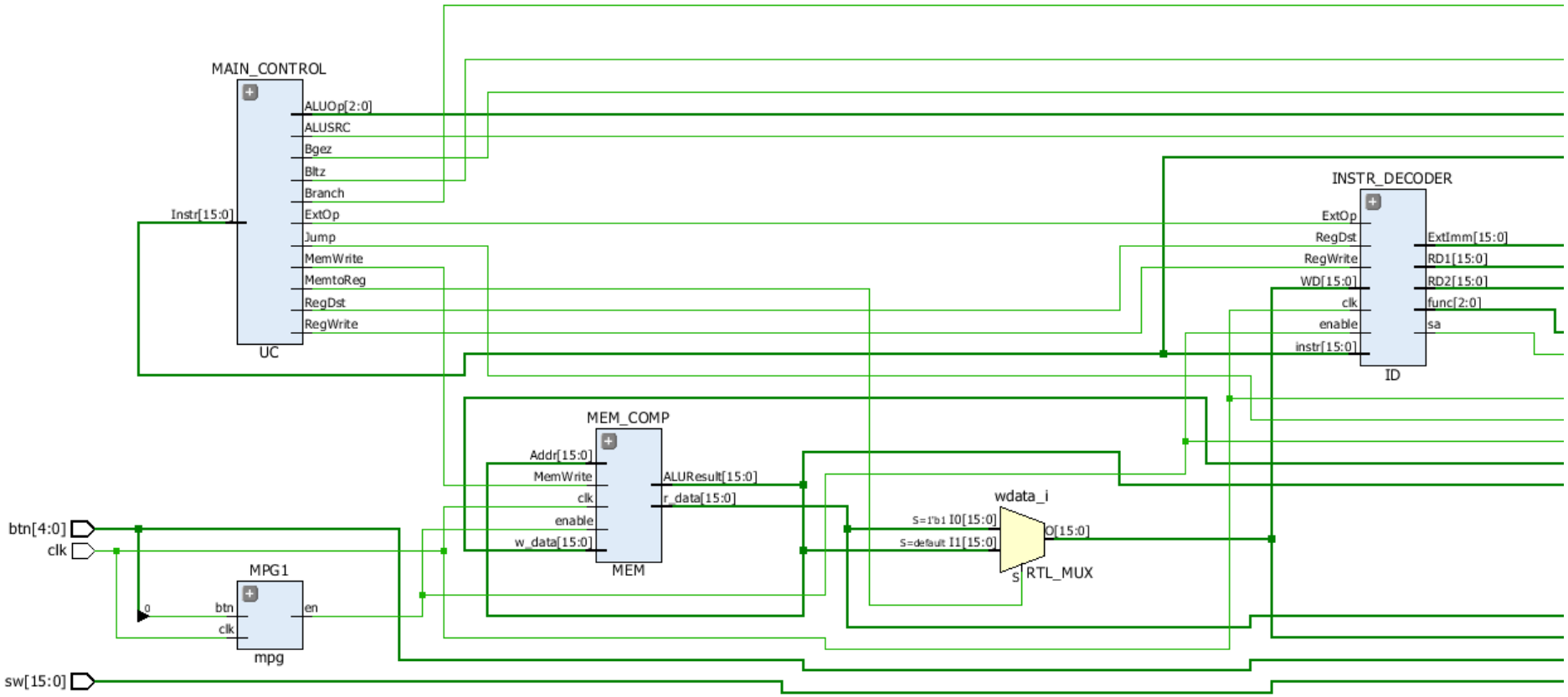
e. RTL schematic pentru entitatea top-level



Pentru o mai bună vizualizare a schemei proiectului, se vor atașa mai jos capturi de ecran mai detaliate ale unor secțiuni alte schemei redate mai sus.

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

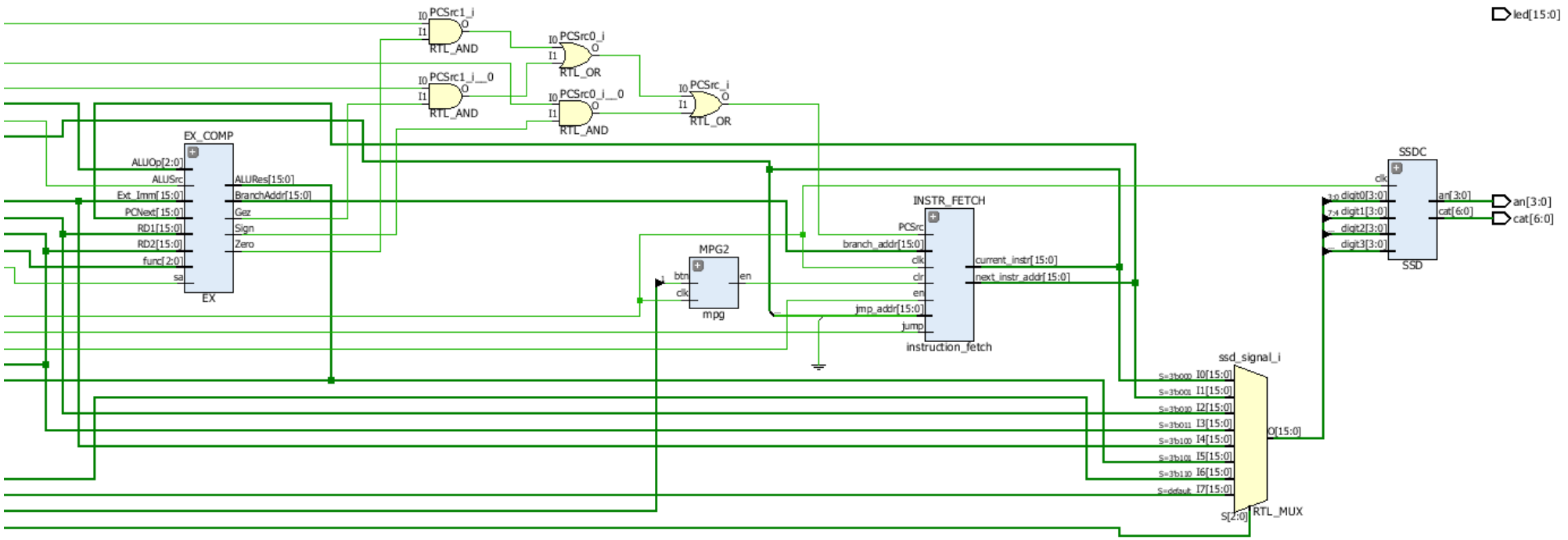


Procesorul MIPS, ciclu unic – versiune pe 16 biți

Student: Lazea Dragoș-Bogdan
CTI-Română, Seria A, Grupa 30223
An academic 2020-2021

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA



Procesorul MIPS, ciclu unic – versiune pe 16 biți

Student: Lazea Dragoș-Bogdan
CTI-Română, Seria A, Grupa 30223
An academic 2020-2021