MINISTERUL EDUCAȚIEI ȘI CERCETĂRII
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Procesorul MIPS - ciclu unic - 32 biți –

Raport de activitate

Student: Ureche Simona Elena

Grupa: 30224

Componente:

- **IFetch** ✓

  - Componenta functionala.
  - Memoria ROM este initializata cu, codificarea binara a instructiunilor care urmaresc rezolvarea urmatoarei cerinte:

    15. Să se determine dacă valorile unui șir de N elemente sunt ordonate crescător. Șirul este stocat în memorie începând cu adresa A (A≥12). A și N se citesc de la adresele 0, respectiv 4. Rezultatul (1=true / 0=false) se va scrie la adresa 8.

C:

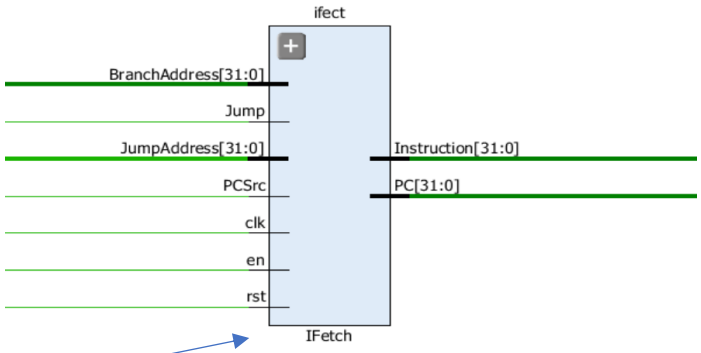
```c
int N = 10;
int A[N] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int sorted = 1;
for (int i = 0; i < N - 1; i++) {
    if (A[i] > A[i + 1]) {
        sorted = 0;
        break;
    }
}

if (sorted == 1) { // Dacă vectorul este sortat
    A[2] = 1;
} else {
    A[2] = 0;
}
```

Cod masina:

```
b"000111_00000_00001_0000000000000000", -- X"1C01 0000". 00 -
b"000111_00000_00010_0000000000000100", -- X"1C02 0004". 01 -

b"000001_00000_00011_0000000000000001", -- X"0403 0001". 02 -

----loop:
b"000111_00001_00100_0000000000000000", -- X"1C24 0000". 03 -
b"000111_00001_00101_0000000000000100", -- X"1C25 0004". 04 -

b"000000_00100_00101_00110_00000_010000", --X"0085 3010". 05 -
b"000100_00110_00000_0000000000000110", -- X"10C0 0006". 06 -

b"000001_00001_00001_0000000000000100", -- X"0421 0004". 07, -
b"000001_00011_00011_0000000000000001", -- X"0463 0001". 08 -

b"000110_00011_00010_1111111111111001", -- X"1862 FFF9". 09 -
----end loop:

--sorted - ajungem aici doar daca sirul este ordonat crescator
b"000101_00000_01000_0000000000000001", -- X"1408 0001". 10 -
b"000011_00000_01000_0000000000001000", --  X"0C08 0008". 11 -
b"100000_00000000000000000000010110", --  X"8000 0016". 12 j 1

--not sorted
b"000101_00000_01000_0000000000000000", -- X"1408 0000",. 13 -
b"000011_00000_01000_0000000000001000", -- X"0C08 0008". 14 -
```

Assembly:

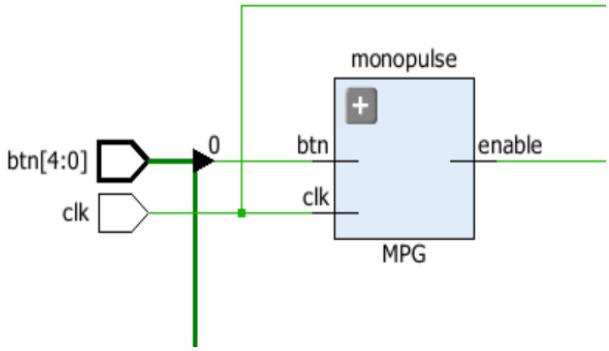
```
lw $1, 0($0)
lw $2, 4($0)
addi $3,$0, 1
lw $4, 0($1)
$5, 4($1)
slt $6, $4, $5
beq $6, $0, 6
addi $1,$1, 4
addi $3,$3, 1
bne $3, $2, -7

ori $8,$0, 1
sw $8, 8($0)
j 16

 ori $8,$0, 0
 sw $8, 8($0)
```



- "IFetch" reprezinta unitatea de preluare a instrucțiunilor pentru procesor. Am intampinat o problema in cadrul instructiunilor de salt, care dupa mai multe incercari a fost rezolvata.

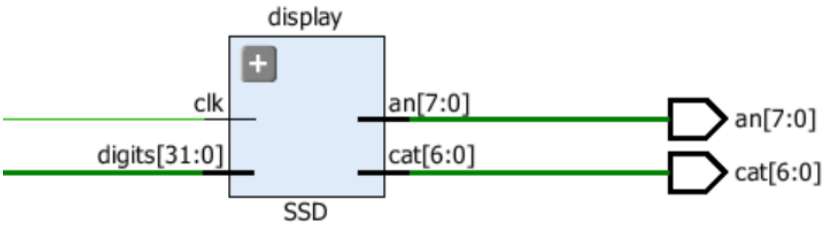
MPG

- Componenta functionala.
- "MPG", servește drept mecanism simplu de generare a unui semnal de activare bazat pe un semnal de ceas și pe starea unui buton:



SSD

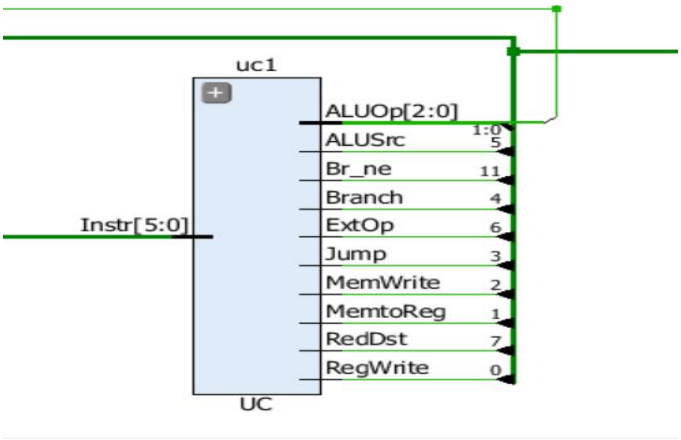
- Componenta functionala.
- "SSD" (Seven Segment Display) este utilizat pentru afișarea cifrelor pe un display cu șapte segmente:



UC

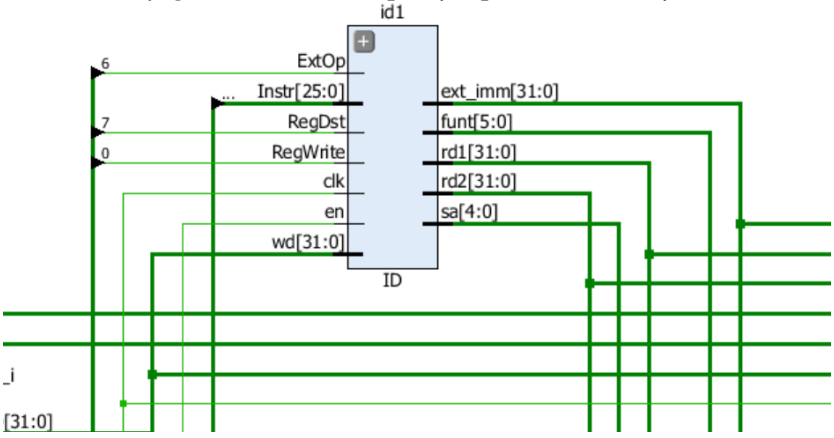
- Componenta functionala.

- Arhitectura UC implementează un decodificator care are intrarea Instr și semnalele de control ca ieșiri. În fiecare ramură "when" a structurii case, se actualizează doar semnalele de control care trebuie să fie diferite de zero pentru instrucțiunea respectivă.



ID

- Componenta functionala.
- „ID" este responsabilă pentru decodificarea instrucțiunilor primite și pregătirea datelor necesare pentru execuția acestora. Aceasta primește instrucțiuni și semnale de control și generează semnale corespunzătoare pentru citirea și scrierea în registre, extinderea imediatelor și gestionarea altor operații specifice instrucțiunilor.

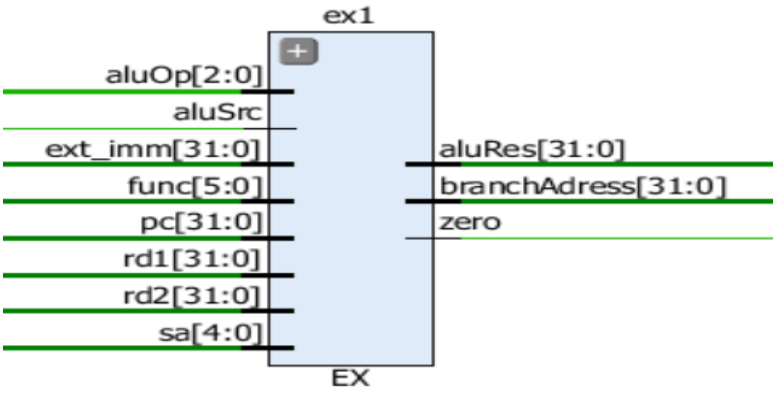


-

EX

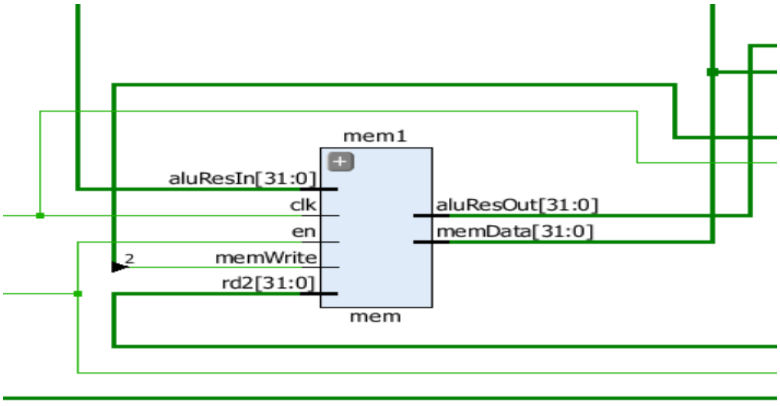
- Componenta functionala.

- "EX" este responsabilă pentru execuția operațiilor aritmetice și logice specifice instrucțiunilor, precum și pentru calcularea adresei de branch în cazul instrucțiunilor de tip branch.



- **MEM** ✓

- Componenta functionala.
  "mem" este responsabilă pentru gestionarea accesului la memoria de date a sistemului. Aceasta primește datele de la componenta de execuție (EX), inclusiv rezultatele operațiilor ALU și datele citite din registre, și efectuează operații de scriere în memorie dacă este necesar.



- Am intampinat probleme din cauza faptului ca nu am conectat AluResOut la AluResIn; Acestea au fost rezolvate ulterior.

⇨ CODUL A FOST TESTAT PE PLACUTA.

- **Instrucțiuni suplimentare**
  - **Tip I:**

1. **Instrucțiunea ori (**bitwise OR Immediate)
   - SAU logic între un registru și o valoare imediată, memorează rezultatul în alt registru;
   - RTL: $t <- $s | ZE(imm); PC <- PC + 4;
   - Sintaxa: ori $t, $s, imm
   - Format:

opcode	rs	rt	imm
000101	sssss	ttttt	iiiiiiiiiiiiiii

   - Semnale : **ExtOp = '1', ALUSrc = '1', RegWrite = '1' si ALUOp = "011"** (restul 0)

2. **Instrucțiunea bne (**Branch on Not Equal)
   - Salt condiționat dacă două registre sunt diferite;
   - RTL: if $s ≠ $t then PC <- (PC + 4) + (SE(offset) << 2) else PC <- PC + 4;
   - Sintaxa: bne $s, $t, offset
   - Format:

opcode	rs	rt	imm
000110	sssss	ttttt	oooooooooooooooo

   - Semnale : **ExtOp = '1', Br_ne = '1', si ALUOp = "010"** (restul 0)

b. Tip R:

3. **Instrucțiunea xor(**bitwise eXclusive-OR)
   - SAU-Exclusiv logic între două registre, memorează rezultatul în alt registru;
   - RTL $d <- $s ^ $t; PC <- PC + 4;;
   - Sintaxa: xor $d, $s, $t
   - Format:

opcode	rs	rt	rd	sa	function
000000	sssss	ttttt	ddddd	00000	100010

   - Semnale : **RegDst = '1', RegWrite = '1', RegWrite = '1'** (restul 0)

4. **Instrucțiunea slt(**Set on Less Than (signed))
   - Dacă $s < $t, $d este inițializat cu 1, altfel cu 0;

- RTL PC <- PC + 4; if $s < $t then $d <- 1 else $d <- 0;
- Sintaxa: slt $d, $s, $t
- Format:

opcode	rs	rt	rd	sa	function
000000	sssss	ttttt	ddddd	00000	010000

- Semnale : **RegDst = '1', RegWrite = '1', RegWrite = '1'** (restul 0)

Observații:

⇨ **AluCtrl:**

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