MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DOCUMENTATIE MIPS 32
BITI PipeLine

Ardelean Robert Emanuel

Grupa:30225

Instructiuni impelementate suplimentar:

~ ***XOR (bitwise eXclusive-OR)***

SAU-Exclusiv logic între două registre, memorează rezultatul în alt registru
$d <=$s ^ $t; PC <= PC + 4;
Sintaxă xor $d, $s, $t
Format: 000000 sssss ttttt ddddd 00000 100110

~ ***SLT (Set on Less Than)***

Dacă $s < $t, $d este inițializat cu 1, altfel cu 0
PC<= PC + 4; if $s < $t then $d <= 1 else $d <= 0;
Sintaxă slti $t, $s, imm
Format: 001010 sssss ttttt iiiiiiiiiiiiiiii

~ ***ORI (bitwise OR Immediate)***

SAU logic între un registru și o valoare imediată, memorează rezultatul în altregistruRTL $t <- $s | ZE(imm); PC <= PC + 4;
Sintaxă ori $t, $s, imm
Format: 001101 sssss ttttt iiiiiiiiiiiiiiii

~ ***ANDI (AND Immediate)***

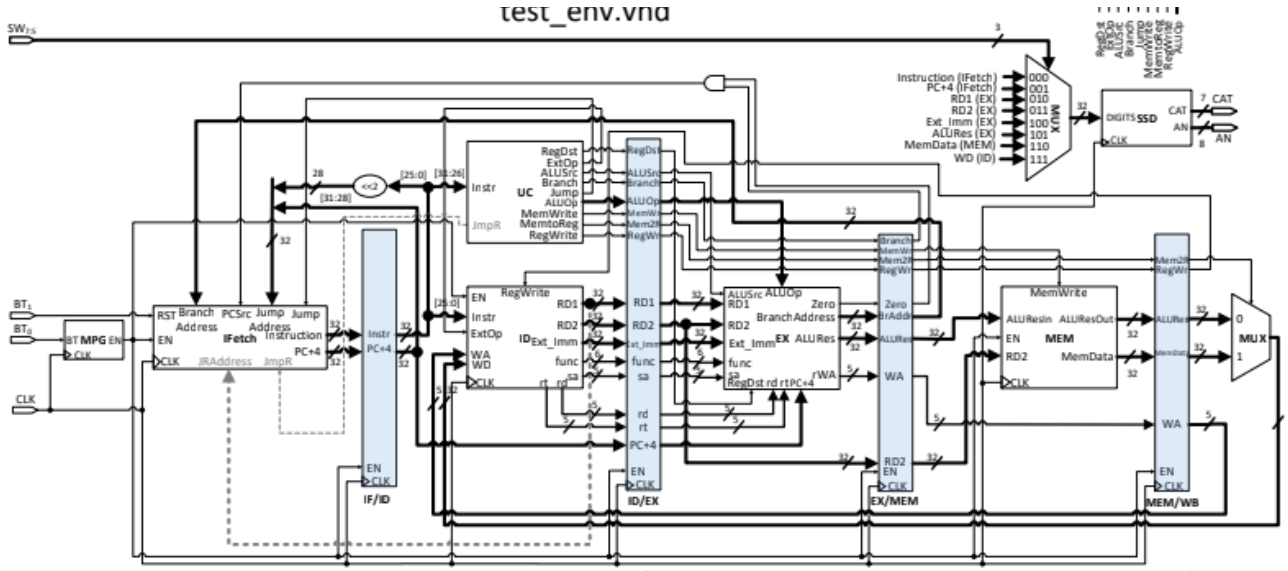
ȘI logic între un registru și o valoare imediată, cu rezultatul în alt registruRTL $t <- $s & ZE(imm); PC <= PC + 4;
Sintaxă andi $t, $s, imm
Format: 001100 sssss ttttt iiiiiiiiiiiiiiii

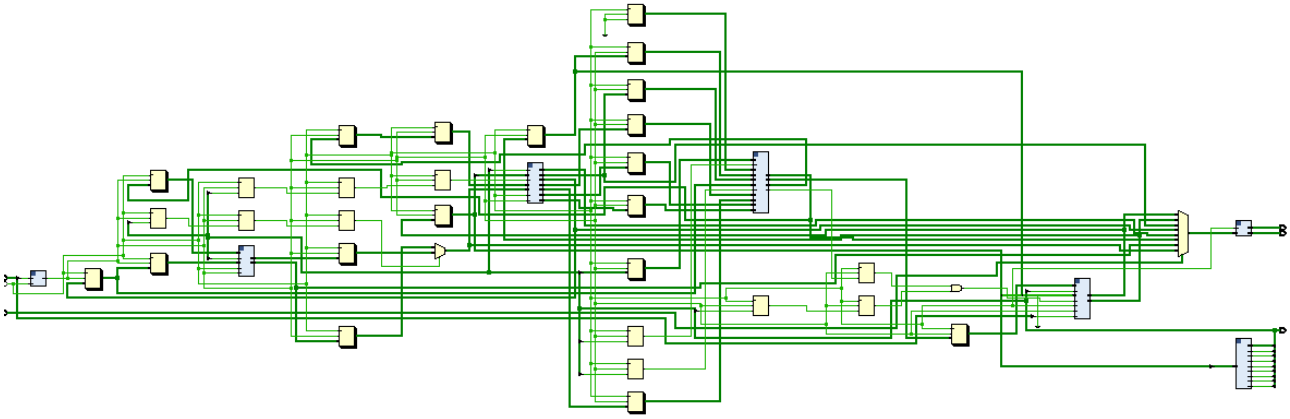
PROBLEMA REZOLVATA SI EXPLICATII:

14. Să se determine suma elementelor cu valori în intervalul [X, Y], dintr-un șir de N numere stocate în memorie începând cu adresa 16. Valorile X, Y și N se citesc din memorie de la adresele 0, 4, respectiv 8. Rezultatul se va scrie în memorie la adresa 12.

```
    # Initializarea registrelor
    lw $4, 0($0)      # X = Mem[0]
    lw $5, 4($0)      # Y = Mem[4]
    lw $6, 8($0)      # N = Mem[8]

    add $2, $0, $0    # Indexul i = 0
    add $11, $0, $0   # Suma s = 0
    add $3, $0, $16   # Adresa de start a vectorului v este 16

    # Loop pentru a itera peste elementele vectorului
loop:
    beq $2, $6, end_loop  # Dacă i == N, ieșim din loop
    lw $7, 0($3)        # v[i] = Mem[$3]
    add $3, $3, 4       # Incrementăm adresa pentru următorul element

    # Verificăm dacă v[i] este între X și Y (inclusiv)
    slt $8, $4, $7      # $8 = 1 dacă X < v[i]; altfel $8 = 0
    slt $9, $7, $5      # $9 = 1 dacă v[i] < Y; altfel $9 = 0
    addi $9, $9, 1      # $9 = 2 dacă v[i] <= Y; altfel $9 = 1
    and $10, $8, $9     # $10 = 1 dacă X <= v[i] și v[i] <= Y

    # Adăugăm v[i] la sumă dacă se încadrează în interval
    beq $10, $0, inafara  # Dacă v[i] nu este în interval, sărim la inafara
    add $11, $11, $7      # s = s + v[i]

inafara:
    addi $2, $2, 1      # Incrementăm indexul i
    j loop              # Sărim înapoi la începutul loop-ului

end_loop:
    sw $11, 12($0)      # Scriem suma în memoria la adresa 12
```

TEST ENV:





```
B"100011_00000_00100_0000000000000000", --lw $4, 0($0)
B"100011_00000_00101_0000000000000100", --lw $5, 4($0)
B"100011_00000_00110_0000000000001000", --lw $6, 8($0)

B"000000_00000_00000_00010_00000_100000", --add $2, $0, $0
B"000000_00000_00000_01011_00000_100000", --add $11, $0, $0
B"000000_00000_10000_00011_00000_100000", --add $3, $0, $16

B"000100_00010_00110_0000000000010101", --beq $2, $6, 21
b"000000_00000_00000_00000_00000_000000", --nop
b"000000_00000_00000_00000_00000_000000", --nop
b"000000_00000_00000_00000_00000_000000", --nop
B"100011_00011_00111_0000000000000000", --lw $7, 0($3)
B"001000_00011_00011_0000000000000100", --addi $3, $3, 4
b"000000_00000_00000_00000_00000_000000", --nop


B"000000_00100_00111_01000_00000_101010", --slt $8, $4, $7
B"000000_00111_00101_01001_00000_101010", --slt $9, $7, $5
b"000000_00000_00000_00000_00000_000000", --nop
b"000000_00000_00000_00000_00000_000000", --nop


B"001000_01001_01001_0000000000000001", --addi $9, $9, 1
b"000000_00000_00000_00000_00000_000000", --nop
b"000000_00000_00000_00000_00000_000000", --nop
B"000000_01000_01001_01010_00000_100100", --and $10, $8, $9

B"000100_01010_00000_0000000000000100", --beq $10, $0, 4
b"000000_00000_00000_00000_00000_000000", --nop
b"000000_00000_00000_00000_00000_000000", --nop
B"000000_01011_00111_01011_00000_100000", --add $11, $11, $7

B"001000_00010_00010_0000000000000001", -- addi $2, $2, 1
B"000010_00000000000000000000001010", --j 7
b"000000_00000_00000_00000_00000_000000", --nop

B"101011_00000_01011_0000000000001100", --sw $11, 12($0)
```

Descrierea Elementelor Funcționale:

PipeLine-ul a fost testat.

1. **Module:**

   a. test_env: Entitatea de nivel superior care integrează diverse componente precum butoane (btn), comutatoare (sw) și LED-uri (led). Simulează un mediu de testare unde intrările pot fi controlate manual, iar ieșirile observate direct.
   b. IFetch: Modulul de aducere a instrucțiunilor responsabil pentru extragerea instrucțiunilor din memorie. Poate răspunde la scenarii de salt, de ramificare și de execuție secvențială normală.
   c. DecID: Modulul de decodare a instrucțiunilor și de decodare imediată care interpretează codul operației, biții funcției și câmpurile imediate ale instrucțiunilor. De asemenea, setează semnalele de control pentru execuția ulterioară.
   d. EX: Unitatea de execuție care efectuează operații aritmetice și logice pe baza instrucțiunii decodate. Gestionează, de asemenea, deplasările și ramificațiile.
   e. Mem: Modulul de acces la memorie pentru operațiuni de încărcare și stocare. Interfațează cu un array de memorie simulat.
   f. MainControl: Unitatea centrală de control care emite semnale de control pe baza codului operației instrucțiunii. Aceste semnale dictează comportamentul altor module din sistem.
   g. MPG: Generatorul de monopuls care oferă semnale de temporizare și control pe baza intrărilor utilizatorului și condițiilor interne.
   h. SSD: Driver pentru afișajul cu șapte segmente care controlează un afișaj cu șapte segmente pentru ieșire pe baza datelor prelucrate.

2. **Semnale și Căi de Date:**

   a. ***Semnale de Date:*** Transportă date specifice instrucțiunii, cum ar fi Instr (instrucțiunea curentă), aluRes (rezultatul de la ALU), PC4 (contorul de program plus patru), etc.
   b. ***Semnale de Control:*** Determină operarea sistemului precum regWrite (controlează operațiunile de scriere în registre), aluOp (definește operația ALU), etc.
   c. ***Ceas și Resetare:*** Elemente tipice de proiectare sincronă unde clk reprezintă ceasul sistemului și rst este utilizat pentru inițializarea sau resetarea stării sistemului.

Elemente Nefuncționale:

1. **Constrângeri și Atribute de Design:**

a. Scalabilitate: Designul modular permite extinderea sau modificarea ușoară a componentelor, cum ar fi adăugarea de noi instrucțiuni sau îmbunătățirea ALU.
b. Fiabilitate: Include mecanisme de verificare și gestionare a erorilor în decodarea și execuția instrucțiunilor pentru a asigura o operare robustă.
c. Utilizare: Configurația mediului de test cu comutatoare și LED-uri permite interacțiunea și testarea ușoară a capacităților procesorului.

2. Provocări în Implementare:

a. Complexitatea Mapării Porturilor: S-au făcut greșeli în timpul mapării porturilor datorită complexității designului.
b. Reactivitatea Aplicației: Aplicația nu a răspuns frecvent comenzilor.
c. Interpretarea Setului de Instrucțiuni MIPS: Probleme în interpretarea setului de instrucțiuni MIPS și modul în care acestea sunt implementate în hardware.
d. Aplicarea Conceptelor Teoretice la Decodarea Practică: Provocări în aplicarea conceptelor teoretice despre decodarea instrucțiunilor la nivel practic.
e. Detectarea și Remedierea Erorilor: Probleme precum bucle infinite sau comportamente neașteptate ale circuitului au fost întâmpinate și rezolvate.