

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE DEPARTAMENTUL CALCULATOARE

DOCUMENTATIE MIPS 32 BITI



Ardelean Robert Emanuel
Grupa:30225



FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE DEPARTAMENTUL CALCULATOARE

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

 $PC \le PC + 4$; if $s < then <math>d \le 1$ else $d \le 0$;

Sintaxă slti \$t, \$s, imm

~ ORI (bitwise OR Immediate)

SAU logic între un registru și o valoare imediată, memorează rezultatul în altregistruRTL $t < - \ | \ ZE(imm); \ PC <= \ PC + 4;$

Sintaxă ori \$t, \$s, imm

~ 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



DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE DEPARTAMENTUL CALCULATOARE

TABEL DE CONTROL:

Semnale de control MIPS32

Semnale de Branch opționale: ? ∈ {gez, ne, gtz}, se va înlocui ? cu o valoare din paranteză, dacă e cazu de operații care se pun în paranteză la ALUOp si ALUCtr (+), (-), (&), (|), (^), (<</), (<</v), (>>/), (>>a), (< OR, ^ - XOR, I - logic, a - aritmetic, v - cu variabilo Instruc Memto function Opcode Reg Mem Reg Br_? JmpR ALUSrc ExtOp Branch ALUOp[2:0] ALUCtrl[3:0] Jump țiune . Instr[31-26] Dst Write Reg Instr[5-0] ADD OR SUB х XOR SLL SLT SRL AND LW XXXXXX SW Ω XXXXXX ADDI Х XXXXXX ORI XXXXXX BEQ XXXXXX BNE XXXXXX Х х ADDI XXXXXX J Х Х Х XXX XXXXXX XXXX

TRASAREA EXECUTIEI:

										-	
Pas	SW(7:5)	"000"	"001"	"010"	"011"	"100"	"101"	"110"	"111"	De completat numai pentru instrucțiuni de salt	
	Instr (în asamblare)	Instr (hexa)	PC+4	RD1	RD2	Ext_Imm	ALURes	MemData	WD	BranchAddr	JumpAddr
0	<u>lw</u> \$4, 0(\$0)	X" 8C040000"	X"00000004"	x"00000000"	x"00000000"	x"00000000"	x"00000000"	X"0000000A"	X"0000000A"	x"00000000"	x"00000000"
1	<u>lw</u> \$5, 4(\$0)	X" 8C050004"	x"00000008"	x"00000000"	x"00000000"	x"00000000"	x"00000000"	x"0000000B	x"0000000B	x"00000000"	x"00000000"
2	<u>lw</u> \$6, 8(\$0)	X" 8C060008"	x"0000000C	x"00000000"	x"00000000"	x"00000000"	x"00000000"	x"0000000C	x"0000000C	x"00000000"	x"00000000"
3	add \$2, \$0, \$0	X"00001020"	x"00000010	x"00000000"	x"00000000"	x"00000000"	x"000000000	x"00000000"	x"000000000	x"00000000"	x"00000000"
4	add \$11, \$0, \$0	X"00005820"	x"00000014	x"00000000"	x"00000000"	x"00000000"	x"00000000	x"00000000"	x"000000000	x"00000000"	x"00000000"
5	add \$3, \$0, \$16	X"101820"	x"00000018	x"00000000"	x"00000000"	x"00000000"	x"00000010	x"00000000"	x"00000010	x"00000000"	x"00000000"
6	beg \$2, \$6, 10	X" 1046000A"	x"0000001C	x"000000000	x"0000000C	x"00000000"	x"00000000"	x"00000000"	x"00000000"	x"00000000"	x"00000000"
7	lw \$7, 0(\$3)	X" 8C670000"	x"00000020	x"00000000"	x"00000000"						
8	add \$3, \$3, 4	X" 641820"	x"00000024	x"00000000"	x"00000000"	x"00000000"	x"00000014	x"00000000"	x"00000014	x"00000000"	x"00000000"
9	<u>slt</u> \$8, \$4, \$7	X" 87402A"	x"00000028	x"0000000A	x"00000000"	x"00000000"	x"00000001	x"00000000"	x"00000001	x"00000000"	x"00000000"
10	slt \$9, \$7, \$5	X" E5482A"	x"0000002C	x"00000000"	x"0000000B	x"00000000"	x"00000001	x"00000000"	x"00000001	x"00000000"	x"00000000"
11	addi \$9, \$9, 1	X" 21290001"	x"00000030	x"00000001"	x"00000000"	x"00000001	x"00000002	x"00000000"	x"00000002	x"00000000"	x"00000000"
12	and \$10, \$8, \$9	X" 1095024"	x"00000034	x"00000001	x"00000002	x"00000000"	x"00000000"	x"00000000"	x"00000000"	x"00000000"	x"00000000"
13	beg \$10, \$0, 1	X" 11400001"	x"00000038	x"00000000	x"00000000	x"00000000"	x"00000000"	x"00000000"	x"00000000"	x"0000003C	x"00000000"
14	add \$11, \$11, \$7	X" 1675820"	X"0000003C	x"000000000	x"00000000"	x"00000000"	x"00000000"	x"00000000"	x"00000000"	x"00000000"	x"00000000"
15	addi \$2, \$2, 1	X" 20420001"	x"00000040	x"000000000	x"00000000"	x"00000001	x"00000001	x"00000000"	x"00000001	x"00000000"	x"00000000"
16	j7	X" 8000007"	X"xxxxxxxx"	x"00000000"	x"00000000"	x"00000000"	x"00000000"	x"00000000"	x"00000000"	x"00000020	x"00000000"
17	sw \$11, 12(\$0)	X"AC0B000C"	x"00000044	x"00000000"	x"00000000"						
					I					I	



DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE DEPARTAMENTUL CALCULATOARE

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]
                    # N = Mem[8]
  lw $6, 8($0)
  add $2, $0, $0
                     # Indexul i = 0
                    \# Suma s = 0
  add $11, $0, $0
  add $3, $0, $16 # Adresa de start a vectorului v este 16
  # Loop pentru a itera peste elementele vectorului
loop:
  beg $2, $6, end loop # Dacă i == N, ieşim 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 \text{ dacă } X < v[i]; \text{ altfel } \$8 = 0
                      \# \$9 = 1 \text{ dacă v[i]} < Y; \text{ altfel } \$9 = 0
  slt $9, $7, $5
                      \# \$9 = 2 \text{ dacă v[i]} \le Y; \text{ altfel } \$9 = 1
  addi $9, $9, 1
                       \# \$10 = 1 \text{ dacă } X \le v[i] \text{ si } v[i] \le Y
  and $10, $8, $9
  # Adăugăm v[i] la sumă dacă se încadrează în interval
  beg $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
                   # Sărim înapoi la începutul loop-ului
  j loop
end_loop:
  sw $11, 12($0)
                        # Scriem suma în memoria la adresa 12
```



FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE DEPARTAMENTUL CALCULATOARE

TEST ENV:

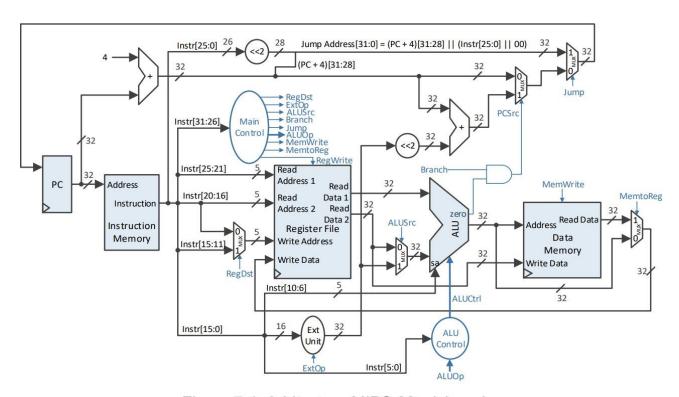


Figura 7-1: Arhitectura MIPS 32, ciclu-unic



FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE DEPARTAMENTUL CALCULATOARE

Descrierea Elementelor Funcționale:

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. Performanță: Optimizat pentru cicluri rapide de decodare și execuție, esențial pentru o simulare a procesorului.



FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE DEPARTAMENTUL CALCULATOARE

- b. Scalabilitate: Designul modular permite extinderea sau modificarea ușoară a componentelor, cum ar fi adăugarea de noi instrucțiuni sau îmbunătățirea ALU.
- c. Fiabilitate: Include mecanisme de verificare și gestionare a erorilor în decodarea și execuția instrucțiunilor pentru a asigura o operare robustă.
- d. 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.