MIPS Ciclu Unic

-- Versiunea pe 32 de biți --

Nume și prenume: Ilieșiu Robert-Mircea

Grupa: 30224

1. Componente folosite și funcționalitatea acestora

Pentru realizarea proiectul MIPS cu ciclu unic, am implementat o versiune pe 32 care conține următoarele componente:

- MPG.vhd reprezintă un Generator de Monopuls Sincron
- SSD.vhd reprezintă Afișoare pe 7 segmente (Seven Segment Display)
- IFetch.vhd reprezintă Unitatea de extragere a instrucțiunilor
- ID.vhd reprezintă Unitatea de Decodificare a instrucțiunilor
- UC.vhd reprezintă Unitatea de Control a procesorului
- EX.vhd reprezintă Unitatea de Execuție a instrucțiunilor
- MEM.vhd reprezintă Unitatea de Memorie
- test env.vhd reprezintă Arhitectura completă a procesorului

Generator de Monopuls Sincron(MPG)

Este compus dintr-un numărător și 3 bistabile D Filp-Flop. Rolul circuitului MPG (Generator de Monopuls Sincron) este de a genera un singur impuls la o apăsare de buton.

Afișor pe 7 segmente(SSD)

Acesta este compus din anozi și catozi. Pentru afișarea uniei cifre se folosesc 7 led-uri active (catozi) iar fiecare cifră este activată de un semnal (anod). Din motive legate de economie a semnalelor alocate, catozii sunt comuni tuturor afișoarelor active, iar pentru a afișa cifre diferite pe afișoare, acestea vor fi activate alternativ, în mod ciclic, pentru perioade scurte de timp (0.16ms).

Unitatea de extragere a instrucțiunilor (IFetch)

Unitatea de extragere a instrucțiunilor (IFetch) primește pe intrările de date adresele de salt și pune la dispoziție, pe ieșiri, adresa imediat următoare (PC+4), respectiv conținutul instrucțiunii curente.

Unitatea de Decodificare a instrucțiunilor (ID)

Unitatea de decodificare a instrucțiunilor (ID) primește pe intrările de date intrucțiunea curentă și valoarea WD, care se scrie în RF, ambele pe 32 de biți. ID pune pe ieșiri, operanzii RD1, RD2 și imediatul extins Ext_Imm(valorile fiind pe 32 de biți). Suplimentar, pe ieșire mai apar câmpurile *function* (6 biți) și *sa* (5 biți) din instrucțiune. Semnalul de control RegDst selectează registrul (adresa) în care se scrie valoarea WD atunci când semnalul de control RegWrite este activ.

Unitatea de Control(UC)

Unitatea de control (UC) , generează semnalele care determină funcționalitatea unităților din calea de date.

Unitatea de Execuție (EX)

Unitatea de execuție (EX) realizează operațiile aritmetice și logice necesare instrucțiunii. Aceasta primește pe intrările de date registrele RD1 și RD2 de la blocul de registre, imediatul extins Ext_imm și adresa de instrucțiune imediat următoare PC+4, codificate pe 32 de biți. Suplimentar, apar câmpurile *func* și *sa* din instrucțiunea curentă, pe 6 biți, respectiv 5 biți. EX pune la dispoziție rezultatul ALU cu semnalul de validare Zero și adresa de salt condiționat Branch Address, calculată astfel:

Branch Address
$$\leq$$
 (PC+4) + (Ext imm \leq 2)

Unitatea de Memorie(MEM)

Unitatea de memorie (MEM) are rol de stocare a datelor, pe 32 de biți. Scrierea în memorie este sincronă pe frontul de ceas ascendent și citirea este asincronă, ca la blocul de registre RF.

Arhitectura completă a procesorului(test env)

Acestă entitate reprezintă arhitectura întregului procesorului și are rolul de a asambla întregul procesor folosind componentele mai sus enumerate.

Toate componentele mai sus enumerate sunt funcționale, niciuna nu prezintă probleme de funcționare. Acestea au fost testate integral pe plăcuță pe parcursul laboratoarelor și au fost remediate problemele apărute pe parcurs. De asemenea unele componente au fost testate și în simulatorul din VIVADO (IFetch.vhd, EX.vhd, test_env.vhd) pentru a fii sigur de funcționalitatea proiectului.

Pe parcursul realizării proiectului s-au întâmpinat diferite probleme de proiectare, valori puse greșit sau greșeli la proiectarea programului MIPS32. Acestea au fost remediate în intregime cu ajutorul cadrului didactic de la laborator sau prin testarea fiecărei componente în parte si remediarea problemelor pas cu pas.

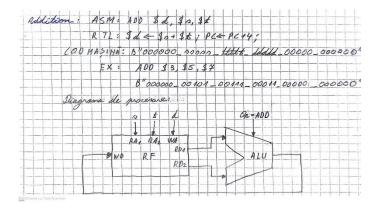
2. Instrucțiuni pentru MIPS 32

Setul de instrucțiuni ales pentru implementarea procesorului MIPS 32 cu ciclu unic este după cum urmează:

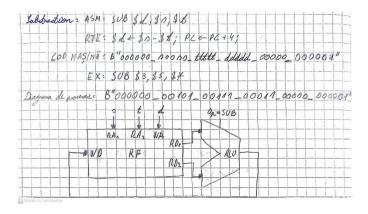
	Addition	add			
	Subtraction	sub			
	Shift Left Logical	sll			
TIP R	Shift Right Logical	srl			
	Logical AND	and			
	Logical OR	or			
	Logical Xclusive-OR	xor			
	Set on Less Than	slt			
	Add Immediate	addi			
	Load Word	1w			
TIP I	Store Word	SW			
	Branch on Equal	beq			
	OR Immediate	ori			
	Set on Less Than Immediate	slti			
TIP J	IP J Jump				

Cele 4 instrucțiuni alse suplimentar sunt: Logical Xclusive-OR (xor), Set on Less Than (slt), OR Immediate (ori) și Set on Less Than Immediate (slti).

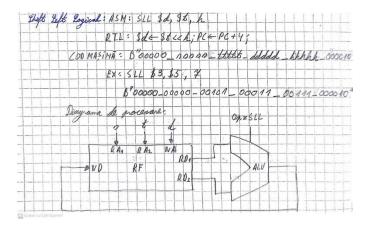
Addition



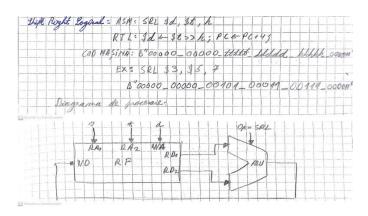
Subtaction



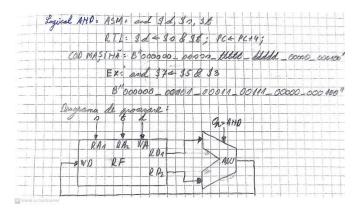
Shift Left Logical



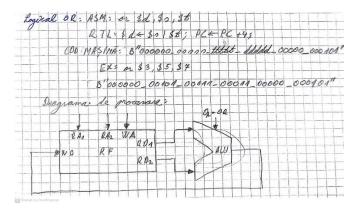
Shift Right Logical



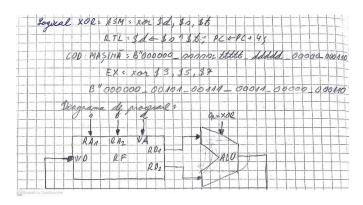
Logical And



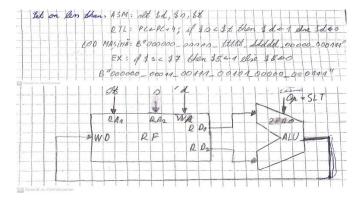
Logical OR



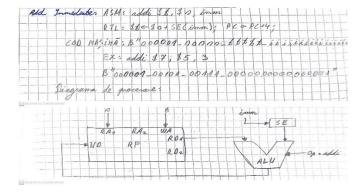
Logical Xclusive-OR



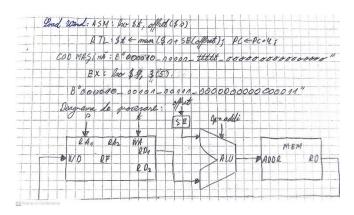
Set on Less Than



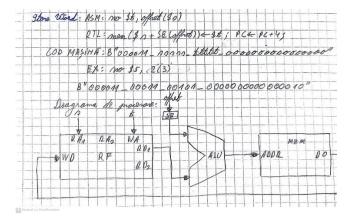
Add Immediate



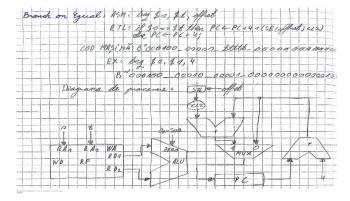
Load Word



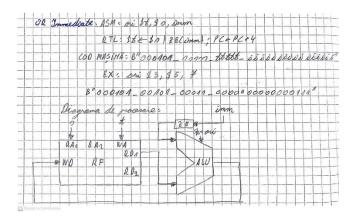
Store Word



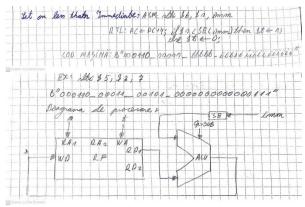
Branch on Equal



OR Immediate



Set on Less Than Immediate



3. Semnale de control MIPS32

Mai jos avem semnalele de control pentru fiecare dintre intrucțiunile alese:

Semnale de control MIPS32

Instruc țiune	Opcode Instr[31-26]	Reg Dst	ExtOp	ALUSrc	Branch	Br_? (optional)	Jump	JmpR (opțional)	Mem Write	Memto Reg	Reg Write	ALUOp[1:0]	function Instr[5-0]	ALUCtrl[2:0]
ADD	000000	1	Х	0	0		0		0	0	1	00 (Tip R)	000000	000 (+)
SUB	000000	1	Х	0	0		0		0	0	1	00 (Tip R)	000001	001 (-)
SLL	000000	1	Х	0	0		0		0	0	1	00 (Tip R)	000010	010 (<<)
SRL	000000	1	Х	0	0		0		0	0	1	00 (Tip R)	000011	011 (>>)
AND	000000	1	Χ	0	0		0		0	0	1	00 (Tip R)	000100	100 (AND)
OR	000000	1	Х	0	0		0		0	0	1	00 (Tip R)	000101	101 (OR)
XOR	000000	1	Х	0	0		0		0	0	1	00 (Tip R)	000110	110 (XOR)
SLT	000000	1	Χ	0	0		0		0	0	1	00 (Tip R)	000111	111 (CMP)
ADDI	000001	0	1	1	0		0		0	0	1	01 (+)	XXXXXX	000 (+)
LW	000010	0	1	1	0		0		0	1	1	01 (+)	XXXXXX	000 (+)
SW	000011	Χ	1	1	0		0		1	Х	0	01 (+)	XXXXXX	000 (+)
BEQ	000100	Х	1	0	1		0		0	Х	0	10 (-)	XXXXXX	001 (-)
ORI	000101	0	0	0	0		0		0	0	1	11()	XXXXXX	101 (OR)
SLTI	000110	0	1	0	0		0		0	0	1	10 (-)	XXXXXX	001 (-)
J	000111	Х	Χ	Х	0		1		0	Х	0	XX	XXXXXX	XXX
			_									•		

4. Trasarea execuției programului de test pentru MIPS32

Am ales din Anexa 7 – problema 12 și enunțul este după cum urmează:

12. Să se determine valoarea pară maximă dintr-un șir de N numere stocate în memorie începând cu adresa A ($A \ge 12$). A și N se citesc de la adresele 0, respectiv 4. Rezultatul se va scrie în memorie la adresa 8.

În tabelul de mai jos va reprezenta ordinea de execuție în timp la apăsarea butonului ENABLE.

Pas	SW(7:5)	"000"	"001"	"010"	"011"	"100" Ext_lmm	"101"	"110" MemData	"111"	De completat numai pentru instructiuni de salt	
	Instr (în asamblare)	Instr (hexa)	PC+4	RD1					WD	BranchAddr	JumpAddr
0	addi \$1, \$0, 0	X"04010001"	X"00000004"	X"000000000"	X"00000000"	X"000000000"	X"00000000"	X"0000000C"	X"00000000"	X"00000000"	X"000000000"
1	addi \$2, \$0, 0	X"04020000"	X"00000008"	X"00000000"	X"00000000"	X"00000000"	X"00000000"	X"0000000C"	X"00000000"	X"00000000"	X"00000000"
2	lw \$3, 4(\$0)	X"08030004"	X"0000000C"	X"00000000"	X"00000000"	X"00000004"	X"00000004"	X"0000000C"	x"0000000c"	X"00000000"	x"00000000"
3	add \$4, \$0, \$3	X"00032000"	X"00000010"	X"00000000"	x"0000000c"	x"00002000"	X"0000000C"	X"00000002"	x"0000000C"	X"00000000"	x"00000000"
4	add \$5, \$0, \$0	X"00002800"	X"0000014"	X"00000000"	X"00000000"	x"00002800"	X"00000000"	X"0000000C"	X"00000000"	X"00000000"	X"00000000"
5	addi \$8, \$0, 0	X"04080000"	X"0000018"	X"00000000"	X"00000000"	x"00000000"	X"00000000"	X"0000000C"	X"00000000"	X"00000000"	X"00000000"
6	beg \$1, \$4, 10	X"1024000A"	X"000001C"	x"00000000"	x"0000000c"	X"0000000A"		X"00000000"	X"FFFFFFF4"	X"0000000A"	X"0C050008"
7	lw \$6, 12(\$2)	X"0846000C"	X"00000020"	X"00000000"	X"00000000"	X"0000000C"		X"00000002"	X"00000002"	X"00000000"	X"00000000"
8	addi \$10, \$0, 0	X"040A0001"	X"00000024"	X"00000000"	X"00000000"	X"00000001"	X"00000001"	X"0000000C"	X"00000001"	X"00000000"	X"00000000"
9	and \$7, \$6, \$10	X"00CA3804"			X"00000001"	X"00003804"		X"0000000C"	X"00000000"		x"00000000"
10	beg \$7, \$10, 3	X"10EA0003"	X"0000002C"	X"00000000"	X"00000001"	x"00000000"		X"00000000"	X"FFFFFFF"	X"00000003"	X"04420004"
11	slt \$8, \$5, \$6	X"00A64007"	X"00000030"	X"00000000"	X"00000002"	X"00004007"		X"0000000C"	X"00000001"	X"00000000"	x"00000000"
12	beg \$8, \$0, 1	X"11000001"	X"00000034"	X"00000001"	x"00000000"	X"00000001"	X"00000001"	X"0000000C"	X"00000001"	X"00000001"	X"04420004"
13	add \$5, \$6, \$0	X"00C02800"	X"00000038"	X"00000002"	x"00000000"	X"00002800"	X"00000002"	X"0000000C"	X"00000002"	X"00000000"	x"00000000"
14	addi \$2, \$2, 4	X"04420004"	X"000003C"	X"00000000"	X"00000000"	X"00000004"	X"00000004"	X"0000000C"	X"00000004"	X"00000000"	X"00000000"
15	addi \$1, \$1, 1	X"04210001"	X"00000040"		X"00000000"	X"00000001"	X"00000001"	X"0000000C"	X"00000001"	X"00000000"	X"00000000"
16	16	X"1C000006"	X"00000044"	x"00000000"	x"00000000"	X"0000006"	X"00000000"	X"0000000C"	x"00000000"	X"00000000"	X"1024000A"
17	sw \$5, 8(\$0)	X"0C050008"	X"00000048"	x"00000000"	X"00000006"	x"00000008"		X"00000000"	X"00000008"	X"00000000"	x"00000000"
18	. , ,, ,										
19											
20											
21											
22											
24											

Trasarea execuției programului de test pentru MIPS32

5. Schema RTL

