CS220 Assignment 8 Report

Hitarth Kirankumar Makawana (230479) Pittala Sruthi (230751)

PDS1

The registers used in this assignment are the same as the registers used in QTSPIM/MIPS convention. However `PC` and `hi, lo` are not considered as a part of the register file. Moreover, There are 32 registers in the register file each 32 bits wide.

PDS2

The instruction size is kept 32 bits wide i.e. 32×1 bits. The data and instruction memory both are implemented using the **Distributed Memory** present in the **IP catalog** of **vivado Vitis**. Both the instruction memory and data memory are 32 bits wide and 512 words (1 word = 32 bits) in depth.

Memory and Register Field Size Distribution

Name	**Size (bits)**	**Usage**
data memory	512×32	Stores the data corresponding to the lw and sw instructions
inst. memory	512×32	Stores the MIPS assembly code instructions to be executed
instruction	32	Stores the current instruction fetched by PC
op_code	6	Opcode of instruction fetched
rs_add	5	Address of the source register rs
rt_add	5	Address of the source register rt
rd_add	5	Address of the destination register rd
shamt	5	Shift amount
func	6	Function code of the instruction
j_add	26	absolute jump address of the target
const	16	immediate values for immediate inst. and branch inst.
RF	32×32	Register File that stores the data of MIPS registers
FPR	32×32	Float Reg. File that stores the data of MIPS float pt. registers
PC	32	Program Counter points to the current instruction being fetched

hi, lo	32	hi and lo MIPS registers used to store the mult result
rs, rt, rd	32	Register contents of inst. fetched

PDS3

Size encoding of R-type, I-type and J-type instructions

R-Type:

[31:26] op_code [25:21] rs_add [20:16] rt_add [15:11] rd_add [10:6] shamt [5:0] func

I-Type:

[31:26] op_code [25:21] rs_add [20:16] rt_add [15:0] const

J-Type:

[31:26] op_code [25:0] j_add

Instruction encoding for the specified instructions

R-Type Instructions

Type	Instruction	Encoding
R-type	add r0, r1, r2	op_code = 0, rs_add, rt_add, rd_add, shamt, func = 32
R-type	sub r0, r1, r2	op_code = 0, rs_add, rt_add, rd_add, shamt, func = 34
R-type	addu r0, r1, r2	op_code = 0, rs_add, rt_add, rd_add, shamt, func = 33
R-type	subu r0, r1, r2	op_code = 0, rs_add, rt_add, rd_add, shamt, func = 35
R-type	mul r0, r1	op_code = 0, rs_add, rt_add, rd_add = 0, shamt = 0, func = 24
R-type	madd r0, r1	op_code = 0, rs_add, rt_add, rd_add = 0, shamt = 0, func = 25
R-type	maddu r0, r1	op_code = 0, rs_add, rt_add, rd_add = 0, shamt = 0, func = 26
R-type	and r0, r1, r2	op_code = 0, rs_add, rt_add, rd_add, shamt, func = 36
R-type	or r0, r1, r2	op_code = 0, rs_add, rt_add, rd_add, shamt, func = 37
R-type	xor r0, r1, r2	op_code = 0, rs_add, rt_add, rd_add, shamt, func = 38
R-type	not r0, r1	op_code = 0, rs_add, rt_add, rd_add = 0, shamt = 0, func = 39
R-type	sll r0, r1, 10	op_code = 0, rs_add, rt_add = 0, rd_add, shamt = 10, func = 0

R-type	srl r0, r1, 10	op_code = 0, rs_add, rt_add = 0, rd_add, shamt = 10, func = 2
R-type	sla r0, r1, 10	op_code = 0, rs_add, rt_add = 0, rd_add, shamt = 10, func = 4
R-type	sra r0, r1, 10	op_code = 0, rs_add, rt_add = 0, rd_add, shamt = 10, func = 3
R-type	slt r0, r1, r2	op_code = 0, rs_add, rt_add, rd_add, shamt, func = 41
R-type	mfc1 r0, f0	op_code = 11, rs_add, rt_add = 0, rd_add, shamt = 0, func = 56
R-type	mtc1 f0, r0	op_code = 11, rs_add, rt_add = 0, rd_add, shamt = 0, func = 55
R-type	mov.s cc f0, f1	op_code = 11, rs_add, rt_add = 0, rd_add, shamt = 0, func = 52
R-type	add.s f0, f1, f2	op_code = 11, rs_add, rt_add, rd_add, shamt, func = 53
R-type	sub.s f0, f1, f2	op_code = 11, rs_add, rt_add, rd_add, shamt, func = 54
R-type	c.eq.s cc f0, f1	op_code = 11, rs_add, rt_add, rd_add, shamt, func = 41
R-type	c.le.s cc f0, f1	op_code = 11, rs_add, rt_add, rd_add, shamt, func = 48
R-type	c.lt.s cc f0, f1	op_code = 11, rs_add, rt_add, rd_add, shamt, func = 49
R-type	c.ge.s cc f0, f1	op_code = 11, rs_add, rt_add, rd_add, shamt, func = 50
R-type	c.gt.s cc f0, f1	op_code = 11, rs_add, rt_add, rd_add, shamt, func = 51

J-Type Instructions

Type	**Instruction**	**Encoding**
J-type	j 10	op_code = 1, j_add = 10
J-type	jal 10	op_code = 2, j_add = 10

I-Type Instructions

Type	**Instruction**	**Encoding**
I-type	addi r0, r1, 1000	op_code = 8, rs, rt, const = 1000
I-type	addiu r0, r1, 1000	op_code = 9, rs, rt, const = 1000
I-type	andi r0, r1, 1000	op_code = 15, rs, rt, const = 1000
I-type	ori r0, r1, 1000	op_code = 13, rs, rt, const = 1000
I-type	xori r0, r1, 1000	op_code = 14, rs, rt, const = 1000
I-type	lw r0, 10(r1)	op_code = 35, rs, rt, const = 10
I-type	sw r0, 10(r1)	op_code = 43, rs, rt, const = 10
I-type	lui r0, 1000	op_code = 36, rs, rt, const = 1000
I-type	beq r0, r1, 10	op_code = 41, rs, rt, const = 10

I-type	bne r0, r1, 10	op_code = 48, rs, rt, const = 10
I-type	bgt r0, r1, 10	op_code = 49, rs, rt, const = 10
I-type	bgte r0, r1, 10	op_code = 50, rs, rt, const = 10
I-type	ble r0, r1, 10	op_code = 51, rs, rt, const = 10
I-type	bleq r0, r1, 10	op_code = 52, rs, rt, const = 10
I-type	bleu r0, r1, 10	op_code = 53, rs, rt, const = 10
I-type	bgtu r0, r1, 10	op_code = 54, rs, rt, const = 10
I-type	slti r0, r1, 100	op_code = 10, rs, rt, const = 100
I-type	seq r0, r1, 100	op_code = 10, rs, rt, const = 100
I-type	jr r0	op_code = 10, rs, rt = 0, const = 0