Indian Institute of Technology Bombay
Department of Electrical Engineering
EE-224: Digital Design

Course Project

Design a computing system, IITB-CPU, whose instruction set architecture is provided. Use VHDL as HDL to implement. *IITB-CPU* is a 16-bit very simple computer developed for the teaching purpose. The *IITB-CPU* is an 8-register, 16-bit computer system, i.e., it can process 16 bits at a time. It should use point-to-point communication infrastructure.

Max Group Size: FOUR

Submission deadlines:

November 13:

Complete Design Document (on paper) – Flow charts, FSM, components.

November 30 :

VHDL code of the controller-FSM. Integration with the datapath along with the test bench .

December 2-3 :

Demonstration of the complete design on FPGA and Viva.

IITB-CPU Instruction Set Architecture

IITB-CPU is a 16-bit very simple computer developed for the teaching that is based on the Little Computer Architecture. The *IITB-CPU* is an 8-register, 16-bit computer system. It has 8 general-purpose registers (R0 to R7). Register R7 is always stores Program Counter. PC points to the next instruction. All addresses are short word addresses (i.e., address 0 corresponds to the first two bytes of main memory, address 1 corresponds to the second two bytes of main memory, etc.). This architecture uses condition code register which has two flags Carry flag (c) and Zero flag (z). The *IITB-CPU* is very simple, but it is general enough to solve complex problems. There are three machine-code instruction formats (R, I, and J type) and a total of 14 instructions. They are illustrated in the figure below.

R Type Instruction format

Opcode (4 bit)	Register A (RA) (3 bit)	Register B (RB) (3-bit)	Register C (RC) (3-bit)	Unused (1 bit)	Condition (CZ) (2 bit)

I Type Instruction format

Opcode (4 bit)	Register A (RA) (3 bit)	Register C (RC) (3-bit)	Immediate (6 bits signed)

J Type Instruction format

Opcode (4 bit)	Register A (RA) (3 bit)	Immediate (9 bits signed)

Instructions Encoding:

	15-12	11-9	8-6	5-3	2	1-0
ADD:	00_00	RA	RB	RC	0	00
SUB:	00_10	RA	RB	RC	0	00
MUL:	00_11	RA	RB	RC	0	00
ADI:	00_01	RA	RB	6 bit Immediate		
AND:	01_00	RA	RB	RC	0	00
ORA:	01_01	RA	RB	RC	0	00
IMP:	01_10	RA	RB	RC	0	00
LHI:	10_00	RA	0 + 8 bit Immediate			
LLI:	10_01	RA	0 + 8 bit Immediate			
LW:	10_10	RA	RB	6 bit Immediate		
SW:	10_11	RA	RB	6 bit Immediate		
BEQ:	11_00	RA	RB	6 bit Immediate		
JAL:	11_01	RA	9 bit Immediate offset			
JLR:	11_11	RA	RB	000_000		

RA: Register A

RB: Register B

RC: Register C

Instruction Description

Mnemonic	Name & Format	Assembly	Action
ADD	Addition (R)	add rc, ra, rb	Add content of regB to regA and store result in regC.
SUB	Subtract (R)	sub rc, ra, rb	Subtract content of regB to regA and store result in regC
MUL	Multiply (R)	mul rc, ra, rb	Multiply content of regB (4 least significant bits) to regA (4 least significant bits) and store result in regC
ADI	Add immediate (I)	adi rb, ra, imm6	Add content of regA with Imm (sign extended) and store result in regB.
AND	Logical And (R)	and rc, ra, rb	Logical AND the content of regB to regA and store result in regC.
ORA	Logical OR (R)	ora rc, ra, rb	Logical OR the content of regB to regA and store result in regC
IMP	Logical Implication (R)	imp rc, ra, rb	Logical Implication of the content of regB to regA and store result in regC
LHI	Load higher immediate (J)	lhi ra, Imm	Place 8 bits immediate into most significant 9 bits of register A (RA) and lower 8 bits are assigned to zero.
LLI	Load lower immediate (J)	lli ra, Imm	Place 8 bits immediate into least significant 9 bits of register A (RA) and higher 0 bits are assigned to zero.
LW	Load (I)	lw ra, rb, Imm	Load value from memory into reg A. Memory address is computed by adding immediate 6 bits with content of reg B.
SW	Store (I)	sw ra, rb, Imm	Store value from reg A into memory. Memory address is formed by adding immediate 6 bits with content of red B.

BEQ	Branch on Equality (I)	beq ra, rb, Imm	If content of reg A (RA) and regB (RB) are the same, branch to PC+Imm*2, where PC is the address of beq instruction
JAL	Jump and Link (I)	jalr ra, Imm	Branch to the address PC+ Imm*2. Store PC into regA, where PC is the address of the jalr instruction
JLR	Jump and Link to Register (I)	jalr ra, rb	Branch to the address in regB. Store PC into regA, where PC is the address of the jalr instruction