IITB CPU

EE224 COURSE PROJECT

ANUJA SATHE – 210070010
AYAN DAS- 210070016
BHAKTI MATSYAPAL - 210070021
DEBASISH PANDA – 21d070021

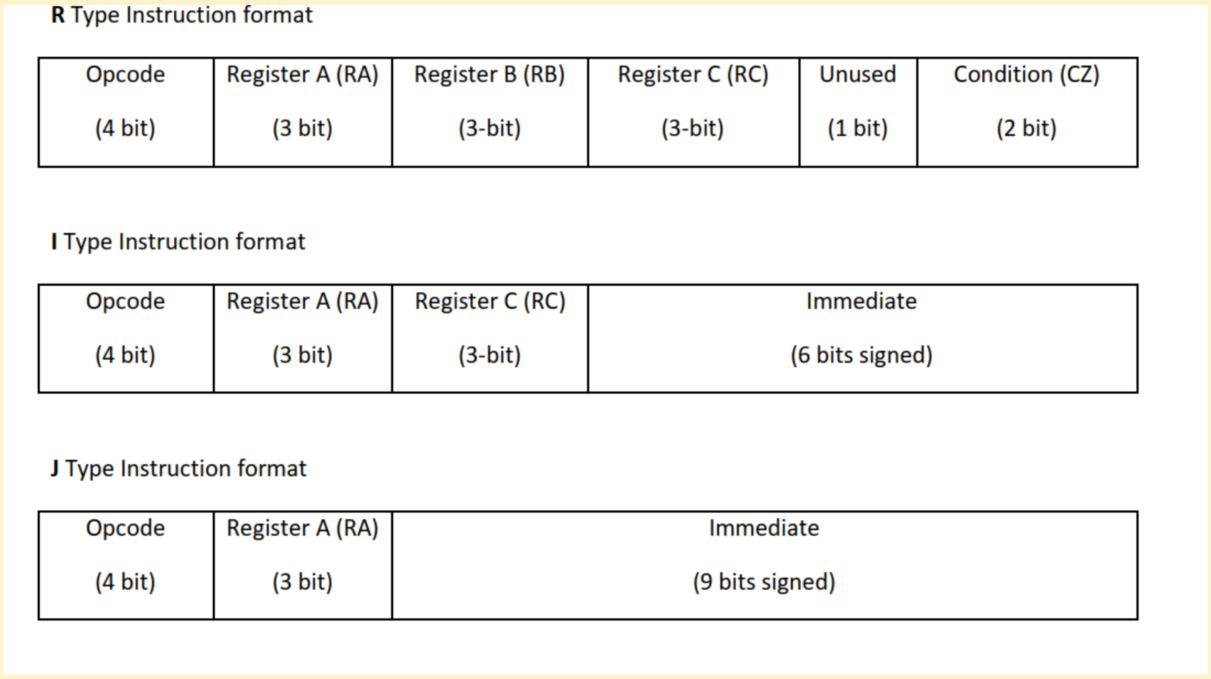
Contents

Overview of the project:

Designing a computing system, IITB-CPU, whose instruction set architecture is provided, using 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 uses point-to-point communication infrastructure.

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. The architecture allows predicated instruction execution and multiple load and store execution. There are three machine-code instruction formats (R, I, and J type) and a total of 14 instructions.



R Type Instruction format

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

I Type Instruction format

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

J Type Instruction format

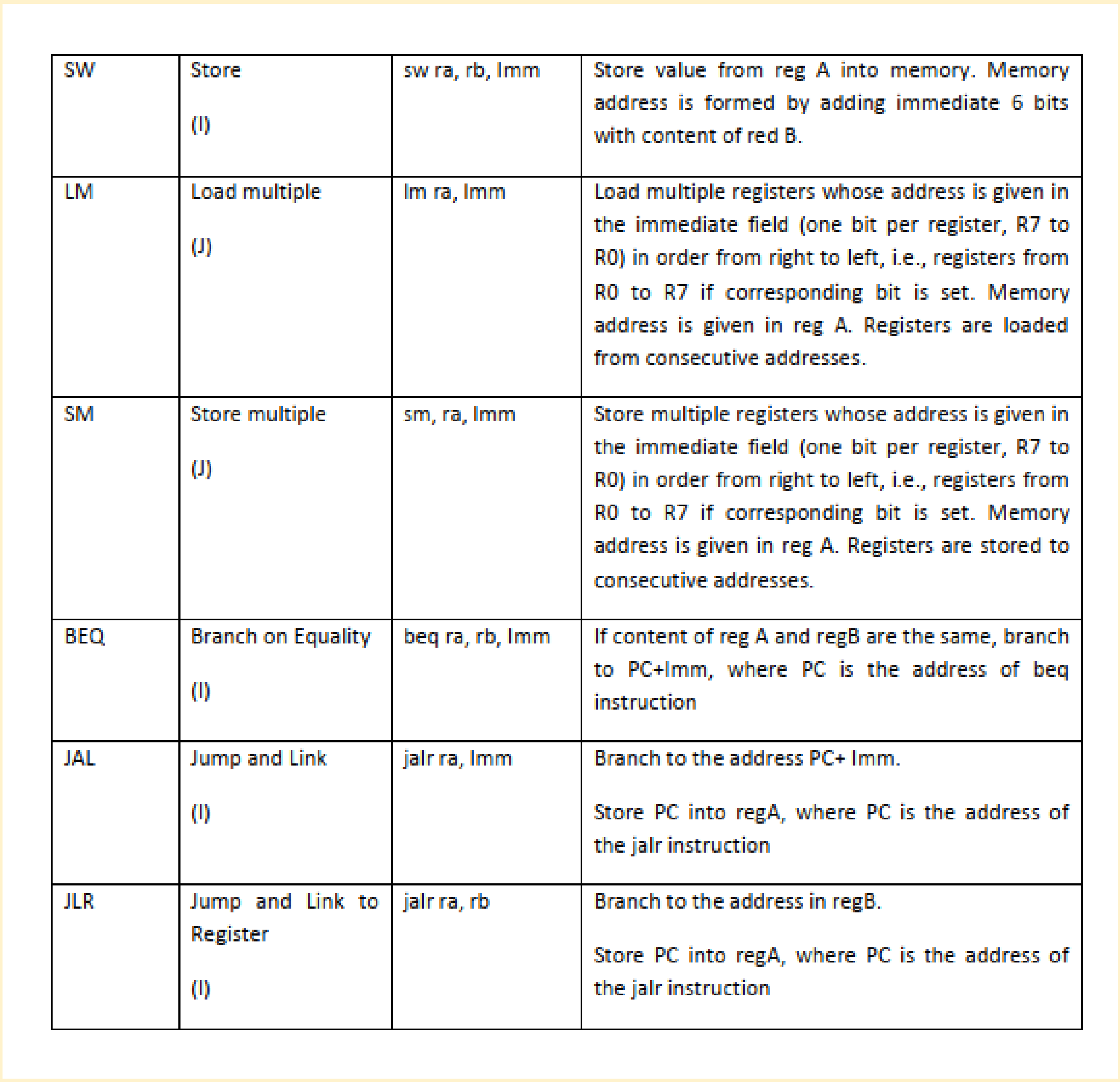
Opcode	Register A (RA)	Immediate
(4 bit)	(3 bit)	(9 bits signed)

Instructions Encoding:

ADD:	00_00	RA	RB	RC	0	00
ADC:	00_00	RA	RB	RC	0	10
ADZ:	00_00	RA	RB	RC	0	01
ADI:	00_01	RA	RB	6 bit Immediate		
NDU:	00_10	RA	RB	RC	0	00
NDC:	00_10	RA	RB	RC	0	10
NDZ:	00_10	RA	RB	RC	0	01
LHI:	00_11	RA	9 bit Immediate			
LW:	01_00	RA	RB	6 bit Immediate		
SW:	01_01	RA	RB	6 bit Immediate		
LM:	01_10	RA	0 + 8 bits corresponding to Reg R7 to R0			
SM:	01_11	RA	0 + 8 bits corresponding to Reg R7 to R0			
BEQ:	11_00	RA	RB	6 bit Immediate		
JAL:	10_00	RA	9 bit Immediate offset			
JLR:	10_01	RA	RB	000_000		

Instruction Description

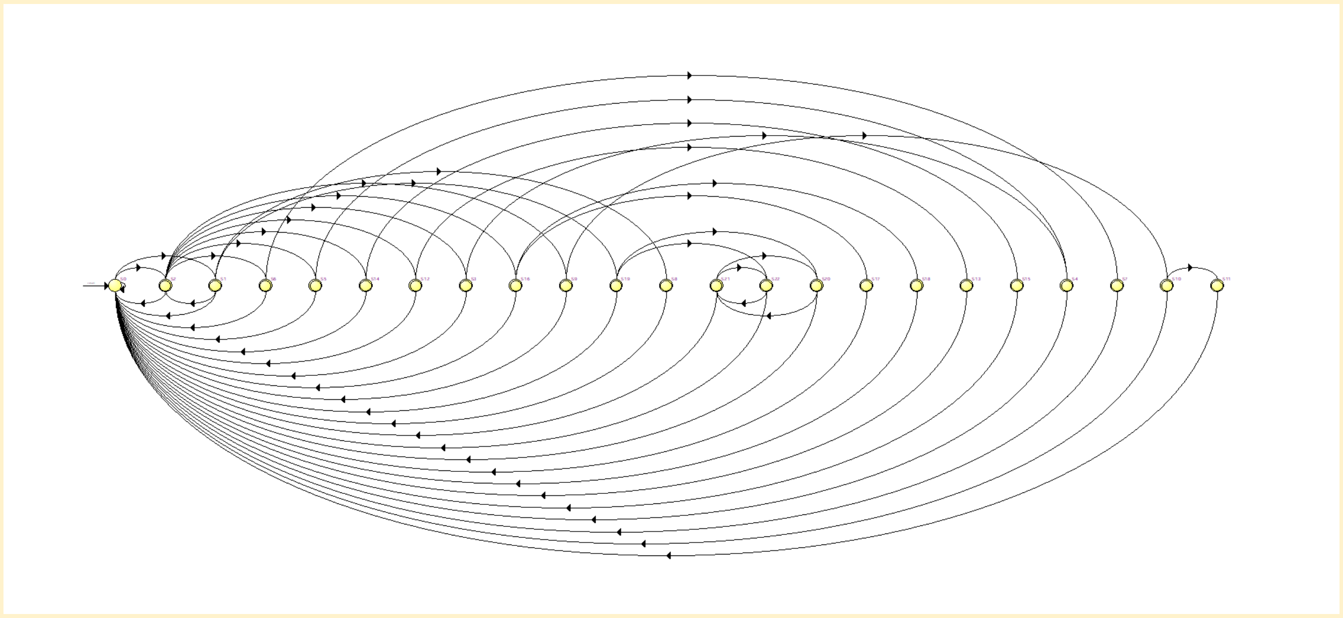
Mnemonic	Name & Format	Assembly	Action
ADD	ADD (R)	add rc, ra, rb	Add content of regB to regA and store result in regC. *It modifies C and Z flags*
ADC	Add if carry set (R)	adc rc, ra, rb	Add content of regB to regA and store result in regC, if carry flag is set. *It modifies C & Z flags*
ADZ	Add if zero set (R)	adz rc, ra, rb	Add content of regB to regA and store result in regC, if zero flag is set. *It modifies C & Z flags*
ADI	Add immediate (I)	adi rb, ra, imm6	Add content of regA with Imm (sign extended) and store result in regB. *It modifies C and Z flags*
NDU	Nand (R)	ndu rc, ra, rb	NAND the content of regB to regA and store result in regC. *It modifies Z flag*
NDC	Nand if carry set (R)	ndc rc, ra, rb	NAND the content of regB to regA and store result in regC if carry flag is set. *It modifies Z flag*
NDZ	Nand if zero set (R)	ndc rc, ra, rb	NAND the content of regB to regA and store result in regC if zero flag is set. *It modifies Z flag*
LHI	Load higher immediate (J)	lhi ra, Imm	Place 9 bits immediate into most significant 9 bits of register A (RA) and lower 7 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. *It modifies flag Z.*



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.
LM	Load multiple (J)	lm ra, Imm	Load multiple registers whose address is given in the immediate field (one bit per register, R7 to R0) in order from right to left, i.e., registers from R0 to R7 if corresponding bit is set. Memory address is given in reg A. Registers are loaded from consecutive addresses.
SM	Store multiple (J)	sm, ra, Imm	Store multiple registers whose address is given in the immediate field (one bit per register, R7 to R0) in order from right to left, i.e., registers from R0 to R7 if corresponding bit is set. Memory address is given in reg A. Registers are stored to consecutive addresses.
BEQ	Branch on Equality (I)	beq ra, rb, Imm	If content of reg A and regB are the same, branch to PC+Imm, where PC is the address of beq instruction
JAL	Jump and Link (I)	jalr ra, Imm	Branch to the address PC+ Imm. 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

Design:

State Diagram:



States:

1. S0:

PC $\Rightarrow$ M_Address	MDR PC_EN
M_Data $\Rightarrow$ T1	

2. S1:

PC $\Rightarrow$ ALU_A	ADD T1_EN
+2 $\Rightarrow$ ALU_B	
ALU_C $\Rightarrow$ PC	

3. S2:

$T1_{11\text{-}9}$ $\Rightarrow$ RF_A1	T2_EN T3_EN
$T1_{8\text{-}6}$ $\Rightarrow$ RF_A2	
RF_D1 $\Rightarrow$ T2	
RF_D2 $\Rightarrow$ T3	

4. S3:

T2 $\rightarrow$ ALU_A	ADD
T3 $\rightarrow$ ALU_B	T2_EN
ALU_C $\rightarrow$ T2	

5. S4:

T2 $\rightarrow$ RF_D3	RF_WR
$T1_{5-3}$ $\rightarrow$ RF_A3	

6. S5:

T2 $\rightarrow$ ALU_A	NAND
T3 $\rightarrow$ ALU_B	T2_EN
ALU_C $\rightarrow$ T2	

7. S6:

T2 $\rightarrow$ ALU_A	T2_EN ADD
IMM($T1_{5-0}$) $\rightarrow$ SE6 $\rightarrow$ ALU_B	
ALU_C $\rightarrow$ T2	

8. S7:

T2 $\rightarrow$ RF_D3	RF_WR
$T1_{8-6}$ $\rightarrow$ RF_A3	

9. S8:

IMM($T1_{8-0}$) $\rightarrow$ SE9 $\rightarrow$ RF_D3	RF_WR
$T1_{11-9}$ $\rightarrow$ RF_A3	

10. S9:

T3 $\rightarrow$ ALU_A	T2_EN ADD
$T1_{5-0}$ $\rightarrow$ SE6 $\rightarrow$ ALU_B	
ALU_C $\rightarrow$ T3	

11. S10:

T3 $\rightarrow$ M_ADDRESS	T2_EN MDR
M_DATA $\rightarrow$ T2	

12. S11:

$T1_{11-9}$ $\rightarrow$ RF_A3	RF_WR
T2 $\rightarrow$ RF_D3	

13. S12:

T3 $\rightarrow$ ALU_A	T2_EN
$T1_{7-0}$ $\rightarrow$ ALU_B	
ALU_C $\rightarrow$ T2	

14. S13:

T3 $\rightarrow$ M_ADDRESS	MDR T2_EN
T2 $\rightarrow$ M_DATA	

15. S14:

T2 $\rightarrow$ ALU_A	PC_EN ADD
T3 $\rightarrow$ ALU_B	
ALU_C $\rightarrow$ Z	

16. S15:

PC $\rightarrow$ ALU_A	Z ADD

if (Z==1) then:
T1$_{5-0}$ $\rightarrow$ SE6 $\rightarrow$ ALU_B
else:
+2 $\rightarrow$ ALU_B	

ALU_C $\rightarrow$ PC	

17. S16:

PC $\rightarrow$ ALU_A	PC_EN ADD
T1$_{8-0}$ $\rightarrow$ SE9 $\rightarrow$ ALU_B	
PC $\rightarrow$ T2	
ALU_C $\rightarrow$ PC	

18. S17:

T2 $\rightarrow$ ALU_A	ADD
+2 $\rightarrow$ ALU_B	
ALU_C $\rightarrow$ T2	

19. S18:

PC $\rightarrow$ ALU_A	ADD T2_EN
+2 $\rightarrow$ ALU_B	
ALU_C $\rightarrow$ T2	

20. S19:

0000000000000000 $\rightarrow$ T2	T2_EN T3_EN
T1$_{11-9}$ $\rightarrow$ RF_A3	
RF_D3 $\rightarrow$ T3	

21. S20:

PC $\rightarrow$ int($T2_{2-0}$)	MDR T3_EN
$T1_{COUNTER}$ $\rightarrow$ RF_WR	
T3 $\rightarrow$ M_ADD	
M_DATA $\rightarrow$ RF_D3	
$T2_{2-0}$ $\rightarrow$ RF_A3	
T3 $\rightarrow$ ALU_A	
+2 $\rightarrow$ ALU_B	
if($T1_{COUNTER}$==1) then ALU_C $\rightarrow$ T3	

22. S21:

T2 $\rightarrow$ ALU_A	T2_EN
+2 $\rightarrow$ ALU_B	
ALU_C $\rightarrow$ T2	

23. S22:

PC $\rightarrow$ int($T2_{2-0}$)	MWR T3_EN
T3 $\rightarrow$ ALU_A	
+2 $\rightarrow$ ALU_B	
If ($T1_{COUNTER}$==1) then: {T3 $\rightarrow$ M_ADDRESS $T2_{2-0}$ $\rightarrow$ RF_A1 RF_D1 $\rightarrow$ M_DATA ALU_C $\rightarrow$ T3}	

State Flow:

1. ADD/ADC/ADZ:
   S0 $\rightarrow$ S1 $\rightarrow$ S2 $\rightarrow$ S3 $\rightarrow$ S4

2. ADI:
   S0 $\rightarrow$ S1 $\rightarrow$ S2 $\rightarrow$ S6 $\rightarrow$ S7

3. NDU/NDC/NDZ:
   S0 $\rightarrow$ S1 $\rightarrow$ S2 $\rightarrow$ S5 $\rightarrow$ S4

4. LHI:
   S0 $\rightarrow$ S1 $\rightarrow$ S8

5. LW:

$$\begin{tikzcd} S0 \arrow[r] & S1 \arrow[r] & S9 \arrow[r] & S10 \arrow[r] & S11 \end{tikzcd}$$

6. SW:

$$\begin{tikzcd} S0 \arrow[r] & S1 \arrow[r] & S2 \arrow[r] & S12 \arrow[r] & S13 \end{tikzcd}$$

7. BEQ:

$$\begin{tikzcd} S0 \arrow[r] & S2 \arrow[r] & S14 \arrow[r] & S15 \end{tikzcd}$$

8. JAL:

$$\begin{tikzcd} S0 \arrow[r] & S2 \arrow[r] & S16 \arrow[r] & S17 \end{tikzcd}$$

9. JLR:

$$\begin{tikzcd} S0 \arrow[r] & S2 \arrow[r] & S16 \arrow[r] & S18 \end{tikzcd}$$

10. LM:

$$\begin{tikzcd} S0 \arrow[r] & S1 \arrow[r] & S19 \arrow[r] & S20 \arrow[r] & S21 \arrow[l, bend left] \end{tikzcd}$$

11. SM:

$$\begin{tikzcd} S0 \arrow[r] & S1 \arrow[r] & S19 \arrow[r] & S22 \arrow[r] & S21 \arrow[l, bend left] \end{tikzcd}$$

Control Variables:

1. CV(CONTROL VARIABLES):

ADD: 00 (ALU performs addition of corresponding inputs)

NAND: 01(ALU performs bitwise NAND of corresponding inputs)

XOR: 10(ALU performs bitwise XOR of corresponding inputs)

2. CZ (CARRY AND ZERO FLAGS):

00: C and Z are unchanged

01: only C is changed

10: only Z is changed

11: both are changed

Based on the above flags, ADD or ADZ or ADC and NDU or NDC or NDZ.

3. WR_EN (WRITE ENABLES):

RF_RD: When set data can be read from register file

RF_WR: When set data can be written in register file

MDR: Enables reading from memory

MWR: Enable writing in to memory

T1_EN, T2_EN, T3_EN: Enables functioning of temporary register

GITHUB Link