



Indian Institute of Technology Bombay
Department of Computer Science & Engineering
CS-226: Digital Logic Design

Course Project

Design a multi-cycle processor, IITB-Proc, whose instruction set architecture is provided. Use VHDL as HDL to implement. *IITB-Proc* is a 16-bit very simple computer developed for the teaching purpose. The *IITB-Proc* is an 8-register, 16-bit computer system. It should use point-to-point communication infrastructure.

Max Group Size: FOUR

Submission deadlines:

May 12, 2021 (Wednesday):

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

VHDL code of the controller-FSM and datapath (ALU, Registers, Register file etc.).
Integration with the datapath along with the test bench .

Demonstration of the complete design on FPGA.

IITB-Proc Instruction Set Architecture

IITB-Proc is a 16-bit very simple computer developed for the teaching that is based on the Little Computer Architecture. The *IITB-Proc* is an 8-register, 16-bit computer system. It has 8 general-purpose registers (R0 to R7). 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-Proc* 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. They are illustrated in the figure below.

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) |

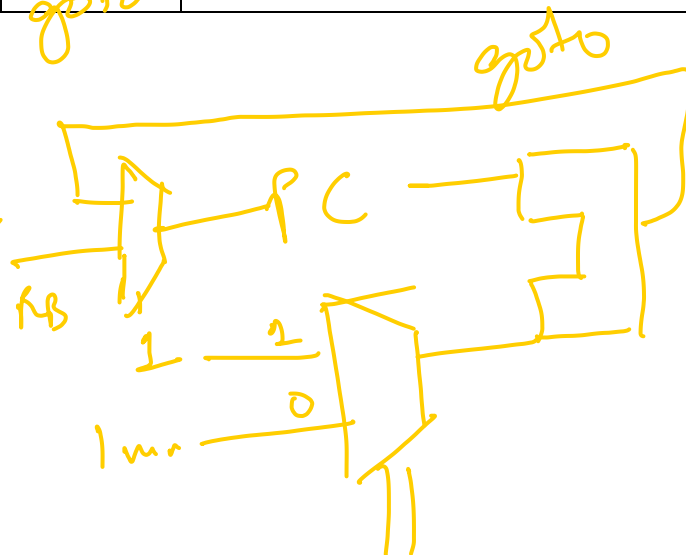
Instructions Encoding:

| | | | | | |
|--------------|----|------------------------|-----------------|---|-----------|
| <u>00_00</u> | RA | RB | RC | 0 | <u>00</u> |
| 00_00 | RA | RB | RC | 0 | 10 |
| 00_00 | RA | RB | RC | 0 | 01 |
| 00_01 | RA | RB | 6 bit Immediate | | |
| 00_10 | RA | RB | RC | 0 | 00 |
| 00_10 | RA | RB | RC | 0 | 10 |
| 00_10 | RA | RB | RC | 0 | 01 |
| 00_11 | RA | 9 bit Immediate | | | |
| 01_00 | RA | RB | 6 bit Immediate | | |
| 01_01 | RA | RB | 6 bit Immediate | | |
| <u>01_10</u> | RA | | | | |
| 01_11 | RA | | | | |
| 11_00 | RA | RB | 6 bit Immediate | | |
| 10_00 | RA | 9 bit Immediate offset | | | |
| 10_01 | RA | RB | 000_000 | | |

RA: Register A

RB: Register B

RC: Register C



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. <i>It modifies C and Z flags</i> |
| 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. <i>It modifies C & Z flags</i> |
| 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. <i>It modifies C & Z flags</i> |
| ADI | Add immediate (I) | adi rb, ra, imm6 | Add content of regA with Imm (sign extended) and store result in regB. <i>It modifies C and Z flags</i> |
| NDU | Nand (R) | ndu rc, ra, rb | NAND the content of regB to regA and store result in regC. <i>It modifies Z flag</i> |
| 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. <i>It modifies Z flag</i> |
| 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. <i>It modifies Z flag</i> |
| 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. <i>It modifies flag Z.</i> |

| | | | |
|-----|----------------------------------|-----------------|---|
| 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. |
| LA | Load All (J) | lm ra | Load all registers (in a sequence of register, R0 to R7) Memory address is given in reg A. Registers are loaded from consecutive addresses. |
| SA | Store All (J) | sm, ra | Store all registers (in a sequence of register, R0 to R7). 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 |