



Indian Institute of Technology Bombay
Department of Electrical Engineering
EE-309: Microprocessor

Project 1

Design a multi-cycle processor, IITB-RISC-22, whose instruction set architecture is provided. *IITB-RISC* is a 16-bit very simple computer developed for the teaching that is based on the Little Computer Architecture. The *IITB-RISC-22* is a 16-bit computer system with 8 registers.

Group: Group of FOUR

Submission deadline: 14th April 2022 (Thursday) 23:59 PM

IITB-RISC Instruction Set Architecture

IITB-RISC is a 16-bit very simple computer developed for the teaching that is based on the Little Computer Architecture. The *IITB-RISC* is an 8-register, 16-bit computer system. It has 8 general-purpose registers (R0 to R7). Register R7 is always stores Program Counter. 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-RISC* 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 17 instructions. They are illustrated in the figure below.

R Type Instruction format

| | | | | | |
|---------|-----------------|-----------------|-----------------|---------|----------------|
| Opcode | Register A (RA) | Register B (RB) | Register B (RB) | 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_01 | RA | RB | RC | 0 | 00 |
| ADC: | 00_01 | RA | RB | RC | 0 | 10 |
| ADZ: | 00_01 | RA | RB | RC | 0 | 01 |
| ADL: | 00_01 | RA | RB | RC | 0 | 11 |
| ADI: 0000 | 00_00 | 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: 0011 | 00_00 | RA | 9 bit Immediate | | | |
| LW: | 01_11 | RA | RB | 6 bit Immediate | | |
| SW: | 01_01 | RA | RB | 6 bit Immediate | | |
| LM: | 11_00 | RA | 0 + 8 bits corresponding to Reg R0 to R7 (right to left) | | | |
| SM: | 11_01 | RA | 0 + 8 bits corresponding to Reg R0 to R7 (right to left) | | | |
| BEQ: | 10_00 | RA | RB | 6 bit Immediate | | |
| JAL: | 10_01 | RA | 9 bit Immediate offset | | | |
| JLR: | 10_10 | RA | RB | 000_000 | | |
| JRI | 10_11 | RA | 9 bit Immediate offset | | | |

RA: Register A

RB: Register B

RC: Register C

Instruction Description

| Mnemonic | Name & Format | Assembly | Action |
|----------|------------------------------------------|-------------------------|---------------------------------------------------------------------------------------------------------------------------|
| ADD | ADD (R) | <i>add rc, ra, rb</i> | 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) | <i>adc rc, ra, rb</i> | Add content of regB to regA and store result in regC, if carry flaf is set. <i>It modifies C & Z flags</i> |
| ADZ | Add if zero set (R) | <i>adz rc, ra, rb</i> | Add content of regB to regA and store result in regC, if zero flag is set. <i>It modifies C & Z flags</i> |
| ADL | Add with one bit left shift of RB (R) | <i>Adl rc,ra,rb</i> | Add content of regB (after one bit left shift) to regA and store result in regC <i>It modifies C & Z flags</i> |
| ADI | Add immediate (I) | <i>adi rb, ra, imm6</i> | Add content of regA with Imm (sign extended) and store result in regB. <i>It modifies C and Z flags</i> |
| NDU | Nand (R) | <i>ndu rc, ra, rb</i> | NAND the content of regB to regA and store result in regC. <i>It modifies Z flag</i> |
| NDC | Nand if carry set (R) | <i>ndc rc, ra, rb</i> | 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) | <i>ndc rc, ra, rb</i> | 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) | <i>lhi ra, Imm</i> | Place 9 bits immediate into most significant 9 bits of register A (RA) and lower 7 bits are assigned to zero. |
| LW | Load (I) | <i>lw ra, rb, Imm</i> | Load value from memory into reg A. Memory address is formed by adding immediate 6 bits with content of reg B. <i>It modifies zero flag.</i> |
| SW | Store (I) | <i>sw ra, rb, Imm</i> | Store value from reg A into memory. Memory address is formed by adding immediate 6 bits with content of reg B. |
| LM | Load multiple (J) | <i>lw ra, Imm</i> | Load multiple registers whose address is given in the immediate field (one bit per register, R0 to R7) 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) | <i>sm, ra, Imm</i> | Store multiple registers whose address is given in the immediate field (one bit per register, R0 to R7) 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) | <i>beq ra, rb, Imm</i> | 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) | <i>jalr ra, Imm</i> | Branch to the address PC+ Imm. Store PC+1 into regA, where PC is the address of the jalr instruction |
| JLR | Jump and Link to Register (I) | <i>jalr ra, rb</i> | Branch to the address in regB. Store PC+1 into regA, where PC is the address of the jalr instruction |

| | | | |
|-----|-------------------------|-------------|----------------------------------------------------|
| JRI | Jump to register (J) | jri ra, Imm | Branch to memory location given by the RA + Imm |
|-----|-------------------------|-------------|----------------------------------------------------|