



COURSE TITLE: Computer Organization and Architecture

Project : ISA

SUBMITTED TO: TnR

COURSE CODE: CSE 332

SECTION 1

SEMESTER: Fall 2020

SUBMISSION DATE: 14/12/2020

Group Number: 3

PREPARED BY:

| Name | ID |
|----------------------|------------|
| Tanzim Alam Fahim | 1831917642 |
| Nishat Naoal | 1831146642 |
| Shanjida Nowshin | 1731056642 |
| Farzana Rahman | 1712173642 |
| Md. Ridwanul Islam | 1812608642 |
| Sadman Hossain Ridoy | 1610456042 |

Introduction:

Our task is to design a new 14 bit single-cycle CPU that has separate Data and Instruction Memory.

We planned our ISA centering on the taking after three categories of programs:

- a) Basic arithmetic & logic operations,
- b) Programs that require checking conditions,
- c) Loop type of programs.

Design:

We have restricted our ISA to work with 16 dedicated instructions.

- **Operand Type:** We use register type operands and memory-based operands.
- **Operands:** We are going to use 2 operands where operands are 5 bits.
- **Operations:** In our design we have decided to use 16 operations so we have dedicated 4 bits to Opcode.
- **Operation Type:** In our Design we have 5 types of operations.
 - 1.Arithmetic: Add, Sub, Addi
 - 2.Logical: AND, OR, NOR, Sll
 - 3.Conditional: beq, bne, Slt, Slti
 - 4.Data Transfer: lw ,sw ,Din ,Dout
 - 5.Unconditional: J
- **Formats:** We use 3 types of Formats.
 - 1. Register Type.
 - 2. Immediate Type.
 - 3. Jump Type.

The ISA Format of a 14 bit Vending Machine are tabulated below:

ISA FORMAT:

R-Type:

| Opcode | Destination Register | Source Register |
|--------|----------------------|-----------------|
| OP | RD | RS |
| 4 Bits | 5 Bits | 5Bits |

I-Type:

| Opcode | Source Register | IMMEDIATE |
|--------|-----------------|-----------|
| OP | RS | IMM |
| 4 Bits | 5 Bits | 5 Bits |

J-Type:

| Opcode | Target |
|--------|---------|
| 4 Bits | 10 Bits |

- Register Table:

| Name of the Registers | Register Number | Value Assigned (5 Bits) | Register purpose |
|-----------------------|-----------------|-------------------------|-------------------------------------------|
| \$zero | \$0 | 00000 | Hardwired to 0 |
| \$sp | \$1 | 00001 | Stack Pointer |
| \$v0 | \$2 | 00010 | Return Values From functions |
| \$v1 | \$3 | 00011 | Same as before |
| \$a0 | \$4 | 00100 | Arguments to functions |
| \$a1 | \$5 | 00101 | Same as before |
| \$a2 | \$6 | 00110 | Same as before |
| \$a3 | \$7 | 00111 | Same as before |
| \$s0 | \$8 | 01000 | Saved registers, preserved by subprograms |
| \$s1 | \$9 | 01001 | Same as before |
| \$s2 | \$10 | 01010 | Same as before |
| \$s3 | \$11 | 01011 | Same as before |
| \$s4 | \$12 | 01100 | Same as before |

| | | | |
|------|------|-------|-------------------------------------------------|
| \$s5 | \$13 | 01101 | Same as before |
| \$s6 | \$14 | 01110 | Same as before |
| \$s7 | \$15 | 01111 | Same as before |
| \$s8 | \$16 | 10000 | Same as before |
| \$s9 | \$17 | 10001 | Same as before |
| \$t0 | \$18 | 10010 | Temporary data, not preserved by subprograms |
| \$t1 | \$19 | 10011 | Same as before |
| \$t2 | \$20 | 10100 | Same as before |
| \$t3 | \$21 | 10101 | Same as before |
| \$t4 | \$22 | 10110 | Same as before |
| \$t5 | \$23 | 10111 | Same as before |
| \$t6 | \$24 | 11000 | Same as before |
| \$t7 | \$25 | 11001 | Same as before |
| \$t8 | \$26 | 11010 | Same as before |
| \$t9 | \$27 | 11011 | Same as before |
| \$at | \$28 | 11100 | Reserved for pseudo instructions |
| \$k0 | \$29 | 11101 | Reserved for kernel |
| \$k1 | \$30 | 11110 | Same as before |
| \$fp | \$31 | 11111 | Frame Pointer |

Table for Operation Type, Instructions, Syntax, Format and Opcode:

| Operation Type | Instruction | Format | Syntax | Meaning | Opcode | Notes |
|----------------|-------------|--------|------------------|--------------------------------------------------------|--------|--------------------------------------------------|
| Logical | AND | R | AND SS_1, SS_2 | $SS_1 = SS_1 \& SS_2$ | 0000 | AND operation between two registers. |
| | OR | R | OR SS_1, SS_2 | $SS_1 = SS_1 SS_2$ | 0001 | OR operation between two registers. |
| | NOR | R | NOR SS_1, SS_2 | $SS_1 = (SS_1 SS_2)'$ | 0010 | Complement of OR operation between two registers |
| | Sll | I | Sll $SS_1, 5$ | $SS_1 = SS_1 \ll 5$ | 0011 | Shift left |
| Arithmetic | ADD | R | ADD SS_1, SS_2 | $SS_1 = SS_1 + SS_2$ | 0100 | Adding two registers |
| | SUB | R | SUB SS_1, SS_2 | $SS_1 = SS_1 - SS_2$ | 0101 | Subtracting two registers |
| | ADDi | I | ADD $SS_1, 9$ | $SS_1 = SS_1 + 9$ | 0110 | Adding constant to register |
| Data Transfer | LW | R | LW SS_1, SS_2 | $SS_1 = SS_2$ | 0111 | Loads from memory |
| | SW | R | SW SS_1, SS_2 | $SS_2 = SS_1$ | 1000 | Stores into memory |
| | Din | R | Din SS_1 | $SS_1 = \text{User Input}$ | 1001 | Input taken from User |
| | Dout | R | Dout SS_1 | $SS_1 = \text{User Output}$ | 1010 | Output shown to User |
| Conditional | Slt | R | Slt SS_1, SS_2 | If($SS_1 < SS_2$) $SS_1 = 1$, Else $SS_1 = 0$ | 1011 | Comparison between two registers |
| | Slti | I | Slti $SS_1, 5$ | If($SS_1 < 5$) $SS_1 = 1$, Else $SS_1 = 0$ | 1100 | Comparison between register and constant |
| | Beq | I | Beq $SS_1, 6$ | If($SS_1 == SS_2$) Go to line 6 | 1101 | Equality check between registers |
| | Bne | I | Bne $SS_1, 6$ | If($SS_1 != SS_2$) Go to line 6 | 1110 | Inequality check between registers. |
| Unconditional | Jump | J | J 7 | Jump to line 7 | 1111 | Jump to another instruction. |

The addressing Modes we will use are given below:

1. Register addressing.
2. Immediate addressing.
3. Base addressing.
4. Pc-relative addressing.
5. Indirect addressing.
6. Direct addressing.