

**North South University**

**CSE332.10**

**ISA Design**

**Submitted to**

Dr. Mainul Hossain

Department of Electrical and Computer Engineering

**Submitted by**

|  |  |
| --- | --- |
| Name |  |
| Abrar Jahin | 2031599642 |
| Shuvodip Biswas | 2011680042 |
| Md. Tanvir Ann Noor Meem | 2021516642 |

**1. How many types of  instruction (R-Type, I-Type, J-Type, etc.)?**

Ans- Two types. I-type and R-type.

**2. Describe each of the formats (fields and field length)**

Ans-

**R-Type Format: -**

·  Each field is 4 bits in length

·  OP is an operation code or opcode that selects a specific operation

·  RS and RT are the first and second source registers

·  RD is the destination register

For example: add $4, $3, $2

       0000                  0100           0011                0010

|  |  |  |  |
| --- | --- | --- | --- |
| OP Code  4 bits | RD  2 bits | RS  2 bits | RT  2 bits |

**I-Type Format: -**

· Load work, store work, branch type, & immediate type are I- type

· RS is a source register, an address for loads and stores, or an operand for branch and immediate arithmetic instructions

·  RD is a source register for branches, but a destination register for the other I-type instructions

For Example: lw $5, 8($6)

1010                 0101                    0110                      0100

|  |  |  |  |
| --- | --- | --- | --- |
| OP Code  4 bits | RD  2 bits | RS  2 bits | Address/Immediate  2 bits |

**3. How many operands? (3 operands, 2 operands)**

Ans: 3 operands [e.g.: z = x + y]

**4. How many operations?**

Ans : 6 operations.

**5.Types  of  operations?(Arithmetic,  logical,  branch  type??How many from each category?  List  the opcodes and respective binary values)**

Ans : 3 types

|  |  |  |
| --- | --- | --- |
| **Operation Types** | **Operations** | **Register Type** |
| 01. Arithmetic | ADD, SUB | (R Type) |
| 02. Logical | AND, OR | (R Type) |
| 03. Data Transfer | LW, SW | (I-Type) |

**Format (fields and field length)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Operations** | **Opcode (OP)**  **4 bit** | | | | **Destination (RD)**  **2 bit** | | **Source Reg (RS)**  **2 bit** | | **Target Reg (RT)**  **2 bit** | |
| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **ADD** | 0 | 0 | 0 | 0 | RD | | RS | | RT | |
| **SUB** | 0 | 0 | 0 | 1 | RD | | RS | | RT | |
| **OR** | 0 | 0 | 1 | 0 | RD | | RS | | RT | |
| **AND** | 0 | 0 | 1 | 1 | RD | | RS | | RT | |
| **lw** | 0 | 1 | 0 | 0 | RD | | RS | | Immediate | |
| **Sw** | 0 | 1 | 0 | 1 | RD | | RS | | Immediate | |

**Instruction Description:**

**ADD:** It adds two registers and stores the result in destination register.

Operation: $d = $s + $t

Syntax: add $d, $s, $t

**SUB:** It subtracts two registers and stores the result in destination register.

Operation: $d = $s - $t

Syntax: sub $d, $s, $t

**AND:** It AND’s two register values and stores the result in destination register. Basically, it sets some bits to 0.

Operation: $d = $s && $t

Syntax: and $d, $s, $t

**OR:** It OR’s two register values and stores the result in destination register. Basically, it sets some bits to 1.

Operation: $d=$s || $t

Syntax: or $d, $s, $t

**lw**: It loads required value from the memory and write it back into the register.

Operation: $d = MEM[$s + offset]

Syntax: lw $d, offset($s)

**sw:** It stores specific value from register to memory.

Operation: MEM[$d + offset] = $s

Syntax: sw $s, offset($d)