****

**Project Milestone-01**

**ISA-Design**

**Course: CSE332 – Computer Organization&Architecture**

**Submitted By: Section: 01**, **Group- 04**

| **Name** | **ID** |
| --- | --- |
| Farhan Ishrak Tahmid | 2031458642 |
| Md. Shahadat Hossain | 2031480642 |
| Nowshin Jahan | 2031633642 |

**Submitted To**

Dr. Tanzilur Rahman (TnR)

Associate Professor, North South University

**Introduction**:

As in the assignment question, the company told us to design a new 12-bit single-cycle CPU that has separate Data and Instruction Memory. The ISA should be general-purpose enough to run provided general programs. It should also be able to connect with the display unit to display results or any data and a keyboard or something similar to get input from the user.

**Design Requirements**

1. **Operands:**

Our Instruction Set Architecture (ISA) will have two operands. This will be applicable for both R-type and I-type instructions for our ISA.

1. **Types of Operands:**

There will be 2 Register-based operands in our ISA.

**Types of Operation:**

There will be 15 dedicated operations in our ISA. There will be 7 R-type operations, 7 I-type operations, and 1 J-Type operation. From these, there will be 3 Arithmetic operations, 4 Logical operations, 4 Data Transfer operations, 4 Conditional operations, and 1 Unconditional operation.

Here is a Table of the instruction sets are given:

**Instructions Table:**

| **Serial** | **Instruction** | **Types** | **Format** | **Example** | **Meaning** | **OPCODE** |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | **Add** | **Arithmetic** | **R** | **Add $s1, $s1,$s2** | **$s1=$s1+$s2** | **0000** |
| **2** | **Addi** | **Arithmetic** | **I** | **Addi $s1, $s1, I** | **$s1=$s1+I** | **0001** |
| **3** | **Sub** | **Arithmetic** | **R** | **Sub $s1, $s1, $s2** | **$s0=$s0-$s1** | **0010** |
| **4** | **AND** | **Logical** | **R** | **AND $s1,$s1,$s2** | **$s1=$s1AND $s2** | **0011** |
| **5** | **OR** | **Logical** | **R** | **OR $s1,$s1,$s2** | **$s1=$s1 OR $s2** | **0100** |
| **6** | **Sll** | **Logical** | **R** | **Sll $s1,$s1,$s2** | **$s1=$s1<<$s2** | **0101** |
| **7** | **Srl** | **Logical** | **R** | **srl $s1,$s1,$s2** | **$s1=$s1>>$s2** | **0110** |
| **8** | **Lw** | **Data Transfer** | **I** | **Lw$s1,offset($s2)** | **$s1=Mem[$s2+offset]** | **0111** |
| **9** | **Sw** | **Data Transfer** | **I** | **Sw $s1,offset($s2)** | **Mem[$s2+offset]=$s1** | **1000** |
| **10** | **beq** | **Conditional** | **I** | **beq $s1 $s2 L** | **If ($s1 == $s2) then branch to L** | **1001** |
| **11** | **bne** | **Conditional** | **I** | **bne $s1 $s2 L** | **If ($s1 != $s2) then branch to L** | **1010** |
| **12** | **J** | **Unconditional** | **J** | **J offset** | **Jump to offset** | **1011** |
| **13** | **Slt** | **Conditional** | **R** | **slt $s1,$s2** | **If$s1 < $s2)**  **$s1=1,**  **else**  **$s1=0** | **1100** |
| **14** | **Slti** | **Conditional** | **I** | **slt $s1,I** | **If$s1 < I)**  **$s1=1,**  **else**  **$s1=0** | **1101** |
| **15** | **In** | **Data Transfer** | **I** | **In $s1** | **Takes value from $in and put it to $s1** | **1110** |
| **16** | **Out** | **Data Transfer** | **I** | **Out $s1** | **Takes value from $s1 and put it to $out** | **1111** |

**Formats:**

In our ISA, we are using three types of formats and those will be R, I, J type.

**R-type ISA format:**

| **OPCODE** | **rs/rd** | **rt** |
| --- | --- | --- |
| 4 bits | 4 bits | 4 bits |

**I-type ISA format:**

| **OPCODE** | **rs/rd** | **immediate** |
| --- | --- | --- |
| 4 bits | 4 bits | 4 bits |

**J-type ISA format:**

| **OPCODE** | **Target** |
| --- | --- |
| 4 bits | 8 bits |

**Registers:**

There will be 16 registers. Because 2^4 = 16

**Register Table:**

|  | **Register Number** | **Conventional Name** | **Usage** | **Assigned** |
| --- | --- | --- | --- | --- |
| **1** | **$0** | **$zero** | Hard wired to 0 | **0000** |
| **2** | **$16** | **$s0** | Saved Register | **0001** |
| **3** | **$17** | **$s1** | Saved Register | **0010** |
| **4** | **$18** | **$s2** | Saved Register | **0011** |
| **5** | **$19** | **$s3** | Saved Register | **0100** |
| **6** | **$20** | **$s4** | Saved Register | **0101** |
| **7** | **$21** | **$s5** | Saved Register | **0110** |
| **8** | **$22** | **$s6** | Saved Register | **0111** |
| **9** | **$23** | **$s7** | Saved Register | **1000** |
| **10** | **$8** | **$t0** | Temporary Data | **1001** |
| **11** | **$9** | **$t1** | Temporary Data | **1010** |
| **12** | **$10** | **$t2** | Temporary Data | **1011** |
| **13** | **$11** | **$t3** | Temporary Data | **1100** |
| **14** | **$12** | **$t4** | Temporary Data | **1101** |
| **15** | **$13** | **$t5** | Temporary Data | **1110** |
| **16** | **$14** | **$t6** | Temporary Data | **1111** |

**Addressing Modes:**

* **Register: *add*** r1,r2 = r1=r1+r2
* **Immediate: *add*** r1, #5 = r1=r1+5
* **Direct: *add*** r1, (0X200) = r1=r1+M[0X200]
* **Auto Increment: *add*** r1, (r2)+ = r1=r1+M[r2], r2++
* **Auto Decrement: *add*** r1, -(r2) = r2- -, r1=r1+M[r2]