

**North South University**

Department of Electrical & Computer Engineering

**CSE332**

**Computer Organization and Architecture**

**ISA & Assembler Design**

|  |  |
| --- | --- |
| **Name** | **ID** |
| **Mohammed Rakib** | **1731176042** |
| **Sanaulla Haq** | **1721380042** |
| **Afra Nawar** | **1812048042** |
| **A.H.M. Fahim Faisal** | **1711758642** |
| **Md. Imran Hossain** | **17120896042** |

**Submitted to**

**Dr. Tanzilur Rahman(Tnr)**

**Date: 01/12/2019**

**Group No. 11**

**ISA - INSTRUCTION SET ARCHITECHTURE**

We are designing a 12 bit ISA which can solve arithmetic, logical, branching and loop operation.

* It has 2 operands.
* The operands are mixed i.e. Memory based and Register based.
* We have allocated 4 bits for the op-code. We have 14 operations in total.
* The 5 different types of operation which can be processed by our ISA are:-

1. Arithmetic
2. Logical
3. Data Transfer
4. Branching
5. Jump

**Instruction Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Category** | **Name** | **Type** | **Opcode** | **Example** | **Interpretation** |
| Arithmetic | add | R | 0000 | add  $s1 $s2 | $s1 = $s1 + $s2 |
| Logical | and | R | 0001 | and  $s1 $s2 | $s1 = $s1 && $s2 |
| Logical | or | R | 0010 | or  $s1 $s2 | $s1 = $s1 || $s2 |
| Logical | nor | R | 0011 | nor  $s1 $s2 | $s1 = ~($s1 || $s2) |
| Logical | slt | R | 0100 | slt  $s1 $s2 | If($s1<$s2) then  $s1=1  else $s1=0 |
| Arithmetic | addi | I | 0101 | addi  $s1 immediate | $s1 = $s1 + immediate |
| Logical | sll | I | 0110 | sll  $s1 2 | $s1 = $s1<<2 |
| Data Transfer | lw | R | 0111 | lw  $s3 $s2 | $s3 = Mem[$s2] |
| Data Transfer | sw | R | 1000 | sw  $s3 $s2 | Mem[$s2]=$s3 |
| Data Transfer | in | R | 1001 | in $s2 | stores user input to $s2 |
| Data Transfer | out | R | 1010 | out $s2 | displays content from $s2 to a display |
| Conditional | bne | I | 1011 | bne  $s3 label | If($s3!=$spr) then go to label |
| Unconditional | j | I | 1100 | j label | go to the line associated with the label |
| Logical | slti | I | 1101 | slti  $s1 2 | If($s1<2) then  $s1=1  else $s1=0 |
| Arithmetic | sub | R | 1110 | sub $s1 $s2 | $s1 = $s1 - $s2 |

**Register Type Instruction Format**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **rd** | **rs** |
| 4 bits | 4 bits | 4 bits |

**Immediate Type Instruction Format**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **rd** | **Immediate** |
| 4 bits | 4 bits | 4 bits |

**Register Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Register Number** | **Conventional Name** | **Usage** | **Binary Value** |
| 0 | $zero | Hardwired to Zero | 0000 |
| 1 | $spr | Special Accumulator Register | 0001 |
| 2 | $s1 | G.P.R. | 0010 |
| 3 | $s2 | G.P.R. | 0011 |
| 4 | $s3 | G.P.R. | 0100 |
| 5 | $s4 | G.P.R. | 0101 |
| 6 | $s5 | G.P.R. | 0110 |
| 7 | $s6 | G.P.R. | 0111 |
| 8 | $s7 | G.P.R. | 1000 |
| 9 | $t0 | Temporary Register | 1001 |
| 10 | $t1 | T.R. | 1010 |
| 11 | $t2 | T.R. | 1011 |
| 12 | $t3 | T.R. | 1100 |
| 13 | $t4 | T.R. | 1101 |
| 14 | $t5 | T.R. | 1110 |
| 15 | $t6 | T.R. | 1111 |

**Limitations: -**

* Our format only allows spaces between instruction words and nothing else. Anything like “,” or “-” will result in an invalid instruction.

**Control Unit Table**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Op**  **Code** | **format** | **Type** | **ALU op** | **in** | **out** | **Result/**  **Load** | **Store** | **Result/**  **Sll** | **Cin** | **Binv** | **Bne** | **Jump** |
| Add | 0000 | 0 | R | 011 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| And | 0001 | 0 | R | 000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Or | 0010 | 0 | R | 001 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Nor | 0011 | 0 | R | 010 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Slt | 0100 | 0 | R | 100 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| Addi | 0101 | 1 | I | 011 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Sll | 0110 | 1 | I | xxx | 0 | 0 | 0 | 0 | 1 | x | x | 0 | 0 |
| Lw | 0111 | 0 | R | xxx | 0 | 0 | 1 | 0 | 0 | x | x | 0 | 0 |
| Sw | 1000 | 0 | R | xxx | 0 | 0 | 0 | 1 | 0 | x | x | 0 | 0 |
| In | 1001 | 0 | R | xxx | 1 | 0 | 0 | 0 | 0 | x | x | 0 | 0 |
| Out | 1010 | 0 | R | xxx | 0 | 1 | 0 | 0 | 0 | x | x | 0 | 0 |
| Bne | 1011 | 1 | I | 101 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| Jump | 1100 | 1 | I | xxx | 0 | 0 | 0 | 0 | 0 | x | x | 0 | 1 |
| Slti | 1101 | 1 | I | 100 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| Sub | 1110 | 0 | R | 101 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |