**COA Project Part I: ISA Design**

**Submitted By:**

Nishank Jain (20ucs134)

Nitesh Singh (20ucs135)

Pranav Chatur (20ucs138)

Pranav Mehta (20ucs140)

Word Length = 8 bits

Instruction Length = 16 bits

Registers = 4

***Instruction Format***:-

| Odpcode | Register | Addressing mode | Immediate / Address / Register |
| --- | --- | --- | --- |
| 4 bits | 2 bits | 2 bits | 8 bits |

| Register | Binary Representation |
| --- | --- |
| R0 | 00 |
| R1 | 01 |
| R2 | 10 |
| R3 | 11 |

| Addressing mode | Binary Representation |
| --- | --- |
| Immediate | 00 |
| Register | 01 |
| Direct | 10 |

Description of Operations :

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Opcode** | **Format** | **Description** |
| **Data Movement** | | | |
| LOAD | 0000 | LOAD R0, #10 | Load an immediate value(#10) to register(R0).  Instruction :0000 00 00 00001010 |
|  | | LOAD R1, 100 | Load a value from memory location(100) to register(R1).  Instruction :0000 01 10 01100100 |
| STORE | 0001 | STORE R0,100 | Store a value from register(R0) to memory location(100).  Instruction: 0001 00 01 01100100 |
| MOVE | 0010 | MOVE R0,R1 | Move data to register; move data from register (R1) to register(R0).  Instruction: 0010 00 01 00000001 |
| **Logical Operation** | | | |
| AND | 0011 | AND R0,R1 | Perform AND operation on register (R0) and register (R1) and store result in register(R0).  Instruction: 0011 00 01 00000001 |
|  | | AND R0,#10 | Perform AND operation on register(R0) and immediate value(10) and store result in register(R0).  Instruction: 0011 00 00 00001010 |
| AND R0,100 | Perform AND operation on register (R0) and value stored on a memory location(100) and store result in register(R0).  Instruction: 0011 00 10 01100100 |
| OR | 0100 | OR R0,R1 | Perform OR operation on R0 and R1 and sore result in R0.  Instruction: 0100 00 10 00000001 |
| NOT | 0101 | NOT R0 | Perform NOT operation on R0 and R1 and sore result in R0.  Instruction: 0101 00 00 00000000 |
| **Arithmetic Operations** | | | |
| ADD | 0110 | ADD R0,R1 | Add register(R0) to register(R1) and store in register(R0).  Instruction: 0110 00 01 00000001 |
|  | | ADD R0,#10 | Add register(R0) to immediate value(10) and store in register(R0).  Instruction: 0110 00 00 00001010 |
| ADD R0,100 | Add register(R0) and value stored on a memory location(100) and store in register(R0).  Instruction: 0110 00 10 01100100 |
| SUB | 0111 | SUB R0,R1 | Subtract register(R1) from register(R0) and store it in register (R0).  Instruction: 0111 00 01 00000001 |
| MUL | 100 | MUL R0,R1 | Multiply register(R0) and register(R1) and store in register(R0).  Instruction: 1000 00 01 00000001 |
| DIV | 1001 | DIV R0,R1 | Divide register(R0) by register(R1) and store in register(R0).  Instruction: 1001 00 01 00000001 |
| **Branching** | | | |
| CMP | 1011 | CMP R0,R1 | Compare number with register.  Instruction: 01 1011 00 00000001 |
|  | | CMP R0,#10 | Instruction: 00 1011 00 00001010 |
| Jump(JMP) | 1100 | JMP #10 | Jump to instruction address unconditionally. JMP offset(10).  Instruction: 00 1100 00 00001010 |
|  | | JMP 100 | Jump to memory address 100.  Instruction: 10 1100 00 00001010 |
| Jump if True(JEQ) | 1101 | JEQ 100 | Jump to instruction address(100) if a given condition satisfies.  Instruction: 10 1101 00 01100100 |
| Jump if False(JNQ) | 1110 | JNQ 100 | Jump to instruction address(100) if a given condition doesn’t satisfy.  Instruction: 10 1110 00 01100100 |

**Flow of Control:**

For implicit control:

PC <- PC + …