



**School of Engineering & Physical Sciences**  
**Department of Electrical and Computer Engineering**  
**(ECE)**

**Instruction Set Architecture (ISA Design)**  
**CSE 332: Computer Architecture & Organization**  
**Section – 04**  
**Spring2021**

**Submission deadline:- 22 May 2021**

| Submitted By                                  | Submitted to            |
|---|-------------------------|
| Name: Moasrrat Shazia Kabir<br>ID: 1831228642 | Tanjila Farah (TnF) Mam |

**Objectives:** Our objectives was to design a 16 Bit ISA which can solve a particular problems. For example - Simple arithmetic & logic operations, branching and loops.

**Operands:** For the design principal we are going to take three operands.

**Types of Operands:** To implement arithmetic instruction we need register operands and for data transfer instruction from memory to register we need memory operands. Lastly for I- type instruction we need immediate operands.






So we need three types of operands.

- Register based.
- Memory based.
- Immediate.

**Operations:** We will allocate 4 bits op-code, so the executable instructions number will be  $2^4$  or 16.

**Types of operations:** In our design there will be five different types of operation.

The operations are:

-  Arithmetic
-  Logical
-  Data Transfer
-  Conditional Branch
-  Unconditional Jump

**Format:**

**R – Type:**

|                                |                           |                           |                           |
|--------------------------------|---------------------------|---------------------------|---------------------------|
| <b>Op-code</b><br><b>4 bit</b> | <b>RS</b><br><b>4 bit</b> | <b>RT</b><br><b>4 bit</b> | <b>RD</b><br><b>4 bit</b> |
|--------------------------------|---------------------------|---------------------------|---------------------------|

**I – Type:**

|                                |                           |                           |                                  |
|--------------------------------|---------------------------|---------------------------|----------------------------------|
| <b>Op-code</b><br><b>4 bit</b> | <b>RS</b><br><b>4 bit</b> | <b>RT</b><br><b>4 bit</b> | <b>Immediate</b><br><b>4 bit</b> |
|--------------------------------|---------------------------|---------------------------|----------------------------------|

**J-Type:**

|                                |  |
|--------------------------------|--|
| <b>Op-code</b><br><b>4 bit</b> | <b>Target Address</b><br><b>12 bit</b> |
|--------------------------------|--|

**Operations:**

| Category      | Operation                    | Name | Type | Opcode | Syntax        | Comments                        |
|---------------|------------------------------|------|------|--------|---------------|---------------------------------|
| Data Transfer | Store word                   | sw   | I    | 0000   | R3,10         | Mem[10]=R3                      |
| Data Transfer | Load word                    | lw   | I    | 0001   | R3,15         | R3=Mem[15]                      |
| Conditional   | Check equality               | beq  | I    | 0010   | beq R4,R5,3   | If(R4==R5) then 3               |
| Arithmetic    | Add number with an immediate | addi | I    | 0011   | addi R3,R2, 5 | R2=R3+5                         |
| Arithmetic    | subtraction                  | sub  | R    | 0100   | sub R4,R5,R6  | R6=R4-R5                        |
| unconditional | Jump                         | jmp  | J    | 0101   | jmp 5         | Go to line 5                    |
| Logical       | Bit –by-bit and              | and  | R    | 0110   | and R3,R1,R2  | R2=R3 & R1                      |
| Conditional   | Compare less than            | slt  | R    | 0111   | slt R2,R3,R4  | If(R2<R3)then R4=1<br>else R4=0 |
| Logical       | Shift left                   | sll  | R    | 1000   | sll R5,R2,R3  | R3=R5<<R2                       |
| Arithmetic    | Add two numbers              | add  | R    | 1001   | add R7,R2,R5  | R5=R7+R2                        |

**Registers:**

| <b>Register Number</b> | <b>Conventional Name</b> | <b>Usages</b>   | <b>Binary Value</b> |
|------------------------|--------------------------|-----------------|---------------------|
| 0                      | R0                       | General purpose | 0000                |
| 1                      | R1                       | General purpose | 0001                |
| 2                      | R2                       | General purpose | 0010                |
| 3                      | R3                       | General purpose | 0011                |
| 4                      | R4                       | General purpose | 0100                |
| 5                      | R5                       | General purpose | 0101                |
| 6                      | R6                       | General purpose | 0110                |
| 7                      | R7                       | General purpose | 0111                |
| 8                      | R8                       | General purpose | 1000                |
| 9                      | R9                       | General purpose | 1001                |
| 10                     | R10                      | General purpose | 1010                |
| 11                     | R11                      | General purpose | 1011                |
| 12                     | R12                      | General purpose | 1100                |
| 13                     | R13                      | General purpose | 1101                |
| 14                     | R14                      | General purpose | 1110                |
| 15                     | R15                      | General purpose | 1111                |

### Translating Some HLL codes using our Designed 16 Bit ISA

- |  |  |
|--|--|
| 1. $a = a + b$<br>add R3, R2, R4                               | #R3 = a, R2 = b<br># R4 gets R3 + R2                       |
| 2. $a = a - b$<br>sub R3, R2, R4                               | #R3 = a, R2 = b<br># R4 gets R3 - R2                       |
| 3. $a = a \text{ AND } b$<br>and R3, R2, R4                    | #R3 = a, R2 = b<br># R4 gets R3 && R2                      |
| 4. $b = 0$<br>for( $i=0, i<10, i++$ )<br>{<br>$b = b + i$<br>} |  |
| lw R0 R5 10  | # get the location value of b and store it in R5           |
| lw R0 R6 15  | # get the location value of b and store it in R6           |
| addi R0 R7 10  | #store 10 in R7  |
| beq R6 R7 L  | #comparing $R7 = 10$ with $R6 = i$ . If equal then go to L |
| add R5 R6 R5   | # $b = b + i$  |
| addi R6 R6 1   | # $i++$  |
| jmp c  | # Jump to c  |