**Objectives:** Our objectives was to design a 16 Bit ISA which can solve a particular problems i. e. Simple arithmetic & logic operations, branching and jump.

**Types of Operands:** To implement arithmetic instruction we need register operands and for data transfer instruction from memory to register we need memory operands. So we need two types of operands.

* **Register based.**
* **Memory based.**

**Operations:** We will allocate 4 bits opcode, so the executable instructions number will be 24 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
* Jump

**Syntax: Opcode:**

nop R1 R2 R3 0000

and R1 R2 R3 0001

add R1 R2 R3 0010

sw R1 R2 3 0011

jmp 12 0100

slt R1 R2 R3 0101

beq R1 R2 7 0110

sub R1 R2 R3 0111

addi R1 R2 17 1000

lw R1 R2 1 1001

sll R1 R2 R3 1010

**Formats:**

We would like to use two types of formats for our ISA. They are:

* **Register Type – R type**
* **Immediate Type – I type**
* **Jump Type – J type**

|  |  |  |  |
| --- | --- | --- | --- |
| **Opcode** | **RS** | **RT** | **RD** |
| 4 bits | 4 bits | 4 bits | 4bits |

* **R Type ISA Format:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Opcode** | **RS** | **RT** | **Immediate** |
| 4 bits | 4 bits | 4 bits | 4 bits |

* **I Type ISA Format:**

|  |  |
| --- | --- |
| **Opcode** | **Immediate** |
| 4 bits | 12 bits |

* **J Type ISA Format:**

**List of Register:**

As we have allocated four bits register so the number of register will be 24  = 16.

|  |  |  |  |
| --- | --- | --- | --- |
| **Register Number** | **Conventional Name** | **Usage** | **Binary Value** |
| 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. **c = a + b # R1 = a, R2 = b, R3 = c**

**add R1, R2, R3 # R3 gets R1 + R2**

* + - 1. **c = a – b #R1 = a, R2 = b, R3= c**

**sub R1, R2, R3 #R3 gets R1 – R2**

* + - 1. **c = a AND b #R1 = a, R2 = b,**

**And R1, R2, R3 # R3 gets R1 && R2**

* + - 1. **C = A[i] #R1 = I, R2=A , R3=C**

**sll R1, R1, R3 # R3 = R2<<R1**

**add R2, R0, R2 #R2 = R2 +R0**

**lw 0 # getting the location value of A[i] from R0 and store**

**5. if(i ==4)**

**i = i+2**

**else**

**i = i -2 # R1 = i**

**sub R0, R0, R0 # R0 = 0**

**addi R0, R0, 4 # R0 = 4**

**beq R1, R0, L #comparing R1 = i with R0 = 4. If equal then it will go to the last line**

**addi R1, R1, -2 # i = i – 2**

**jmp exit # Jump to exit**

**addi R1, R1, 2 # i = i + 2**