**Aim:** Our aim is to design a new 8 bit single-cycle CPU.

**Purposes:** Our purposes is to propose a details design of the ISA in order to solve different types of problems like arithmetic operation, conditional operation etc.

**Operands:** Our goal is to use accumulator base ISA. For this reason we are going to take two operands. We will address these two operands as **s** and **t**.

**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.

1. **Register based.**
2. **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 categories are:

1. Arithmetic
2. Logical
3. Data Transfer
4. Conditional Branch
5. Unconditional Jump

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Category** | **Operation** | **Name** | | **Type** | | **Opcode** | **Syntax** | **Meaning** |
| Arithmetic | Add two numbers | add | | R | | 0000 | add,  $s1,$s2 | $s1 = $s1 + $s2 |
| Arithmetic | Add number with an immediate | addi | | I | | 0001 | addi  $s1,2 | $s1 = $s1 + 2 |
| Arithmetic | subtraction | sub | | R | | 0010 | sub  $s1, $s3 | $s1 = $s1-$s3 |
| Data transfer | Load word | lw | | I | | 0011 | lw  $1s, 2 | $s1=Mem[$ac+2] |
| Data Transfer | Store word | sw | | I | | 0100 | Sw  $s1,2 | Mem[$ac+2]=$s1 |
| Logical | Shift left | sll | | I | | 0101 | sll  $s1, 2 | $s1=$ac<<2 |
| Conditional | Check equality | beq | | I | | 0110 | Beq  $s1,2 | If($s==ac) then 2 |
| Unconditional | Jump | jump | I | | 0111 | | pJmp $ac,2 | Go to +2 |
|  | | | | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Logical | Bit –by-bit and | And | R | 1000 | And  $s1 , $s2 | $s1=$s1 & $s2 |
| Logical | Negative jump | njmp | R | 1001 | Njmp $ac,2 | Go to -2 |
| Logical | Bit –by-bit nand | Nand | R | 1010 | Nand  $s1,$s2 | $s1 = $s1 Nand $s2 |
| Conditional | Compare less than | slt | I | 1011 | slt  $ac,$s1 | If($s<$ac)then  $ac=0  else $ac=1 |
| Data transfer | Display Values | disp | I | 1101 | disp $s1 | Display value of $s1 |
| Data transfer | Load accumulator | la | I | 1110 | la  $s1 | $ac= value[$s1] |
| Data transfer | Load constant to accumulator | lai | I | 1111 | Lai $ac 3 | $ac= value(3) |

**Formats:**

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

1. **Register Type – R type**
2. **Immediate Type – I type**

**R Type ISA Format**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **rs** | **rt** |
| 4 bits | 2 bits | 2 bits |

**I Type ISA Format**

|  |  |  |
| --- | --- | --- |
| **Opcode** | **rs** | **Immediate** |
| 4 bits | 2 bits | 2 bits |

**List of Register:**

As we have allocated two bits register so the number of register will be 22 = 4.

|  |  |  |  |
| --- | --- | --- | --- |
| **Register Number** | **Conventional Name** | **Usage** | **Binary Value** |
| $0 | $ac | General purpose | 00 |
| $1 | $s1 | General purpose | 01 |
| $2 | $s2 | General purpose | 10 |
| $3 | $s3 | General purpose | 11 |
|  |  |  |  |

**Translating some HLL to our design ISA:**

1. g= g + c

Let

g= $s2, c= $s3

Add $s2, $s3 [$s2 = $s2+ $s3] #storing g = g + c in $s2

1. g= g – c

Let

g= $s2, c= $s3

sub $s2, $s3 [$s2 = $s2- $s3] #storing g = g - c in $s2

1. B = A[i]

Let,

A= $s2, B= $s3, i= $s1

Sll $s1,1 [I =2\*i ] # here $s1=i,i=i\*2

Add $s2, $s1 [$s2= $s2 + $s1 ] # here $s2 = base address of

A ,adding$s1=i with the base address

la $ac,$s2 [$ac = value[$s2]] # loading $s2 value to accumulator

Lw $s3,0 [$s3 = M($ac+0)] # fetching the value of A[i] and save the value of A[i] in $s3

4. if ( i == 4 )

i = i+1;

else

i = i-1;

Let i = $s1

La $ac, 4 [$ac =value(4)] # loading 4 to accumulator

beq $s1,L1 #comparing $S4=i with $ac=4 ,if equal then go to L1

addi $s1, -1 # i=i-1

pjmp $ac, exit# go to exit

L1: addi $s1,1 #i =i+1

Exit:

5.for(int i=0,i<a;i++){

C= C\*2;

}

Let i = $s1, a= $s2, c= $s3

Sub $s1, $s1 [$s1= $s1-$s1] #making i =0

La $ac, $s2 [$ac= value[$s2]] # storing $ac= $s2

Loop: Beq $s1 ,exit [if($s1==$ac)] #if i=a then go to exit

Sll $s3,1 #shifting 1 bit in $s3

Addi $s1 ,1 # $s3=$s3+1

Njmp $ac, Loop #go to Loop

Exit:

**Limitation:**

**1.** As our opcode is 4bits , we can Can not Afford to have more than 16 operations.

**2**.we can not use immediate operation if it’s size Is larger

than 2^2= 4.