**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 3 bits opcode, so the executable instructions number will be 23 or 8.

**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 | | 000 | add,  $s, $t | $s = $s + $t |
| Arithmetic | Add number with an immediate | addi | | I | | 001 | addi  $s,2 | $s = $s + 2 |
| Arithmetic | subtraction | sub | | R | | 010 | sub  $s, $t | $s = $s-$t |
| Data transfer | Load word | lw | | I | | 011 | lw  $s, 2 | $s= Mem[$s+2] |
| Data Transfer | Store word | sw | | I | | 100 | sw  $S, 2 | Mem[$s+2]=$s |
| Logical | Shift left | sll | | I | | 101 | sll  $s, 2 | $s=$s<<2 |
| Conditional | Check equality | beq | | I | | 110 | beq  $s, 2, jump | if($s==2) then jump |
| Unconditional | Jump | jump | I | | 111 | | Jump 2 | Go to 2 |
|  | | | | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

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 | $s0 | 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 [g=g + c] #storing g=g + c in $s2

1. g= g – c

Let

g= $s2, c= $s3

sub $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

Lw $s2,0 [$s2 = M($s2+0)] # fetching the value of A[i]

Sub $s3,$s3

add $s3,$s2 [$s3 = $s3 + $s2] #saving the value of A[i] in $s3

4. if ( i == 4 )

i = i+1;

else

i = i-1;

Let i = $s1

beq $s1,4, L1 #comparing $S4=i with 4 ,if not equal then go to L1

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

jump exit # go to exit

L1: add $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 #making $S1=0 by $s1-$s1

Sub $s0, $s0 #making $s0=0 by $s0-$s0

Add $s0, $s2 # storing a=$s2 in $s0

Loop: Beq $s0, exit #if i=a then go to exit

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

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

Jump Loop #go to Loop

Exit:

**Limitation:**

1. As our opcode is 3bits , we can Can not Afford to have more than 8 operations.

**2**. we do not have slt operaion. So instead we have to use

beq operation. So we have to use more instructions.

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

than 2^3= 8.