Objective:

After converting a high-level programming language into assembly language by a compiler, the assembler takes the assembly language as input and convert into a binary instruction (machine language) for hardware as 16-bit output**.**

Operands: Our goal is to use accumulator base ISA. For this reason, we have used 3 type of operands in our design. They are mainly register based.

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 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

Category	Operation	Name	Type	Opcode	Syntax	Comments
	nothing	nop		0000		nothing
Arithmetic	Add number with an immediate	addi	I	0001	addi $r1 $r2 5	$r2 = $r1 + 5
Logical	Bit –by-bit and	And	R	0010	and $r1 $r2 $r3	$r3=$r1 & $r2
Conditional	Compare less than	slt	R	0011	slt $r1 $r2 $r3	If($r1<$r2)then $r3=1 else $r3=0
Data transfer	Load word	lw	I	0100	lw $r1 $r2 16	$r2=Mem[$r1+16]
Conditional	Check equality	beq	I	0101	beq $r1 $r2 7	If($r1==$r2) then 7
Arithmetic	Add two numbers	add	R	0110	add $r1 $r2 $r3	$r3 = $r1 + $r2
Logical	Shift left	sll	I	0111	sll $r1 $r2 2	$r2=$r1<<2
Unconditional	Jump	jmp	j	1000	jmp 12	Go to line 12
Arithmetic	subtraction	sub	R	1001	sub $r1 $r2 $r3	$r3 = $r1 - $r2
Data Transfer	Store word	sw	I	1010	sw $r1 $r2 16	Mem[$r1+16]=$r2

Formats:

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

- **R type**
- **I type**
- **J type**

R-Type

Op-Code 4 bit	rs 4 bit	rt 4 bit	rd 4 bit

I-Type

Op-Code 4 bit	rs 4 bit	rt 4 bit	Immediate 4 bit

J-Type

Op-Code 4 bit	Target Address 12 bit

List of Register:

As we have allocated four bits register so the number of registers will be $2^4 = 16$.

Register Number	Conventional Name	Usage	Binary Value
0	$r1	General purpose	0000
1	$r2	General purpose	0001
2	$r3	General purpose	0010
3	$r4	General purpose	0011
4	$r5	General purpose	0100
5	$r6	General purpose	0101
6	$r7	General purpose	0110
7	$r8	General purpose	0111
8	$t1	General purpose	1000
9	$t2	General purpose	1001
10	$t3	General purpose	1010
11	$t4	General purpose	1011
12	$t5	General purpose	1100
13	$t6	General purpose	1101
14	$t7	General purpose	1110
15	$t8	General purpose	1111

Translating Some HLL codes using our Designed 16 Bit ISA

1. a = a + b #$r1 = a, $r2 = b

add $r1, $r2, $r3 # $s3 gets $r1 + $r2

2. a = a – b #$r1 = a, $r2 = b

sub $r1, $r2, $r3 # $r3 gets $r1 - $r2

3. a = a and b #$r1 = a, $r2 = b

And $r1, $r2, $r3 # $r3 gets $r1 && $r2

4. a=b

Jump to 5

Beq $r1 $r2 5 #$r1=a, $r2=b

5.If(a<b)

C=a+b #$r4=C

else C=a-b

```
1   lw $r5 $r1 0          # $r1= MEM[$r5+0] , $r1=a
1   lw $r5 $r2 1          # $r2= MEM[$r5+1], $r2=b
2   addi $r6 $r6 1        # $r6=1
3   slt $r1 $r2 $r3       # $r1<$r2 then $r3=1 else $r3=0
4   beq $r3 $r6 7         # $r3==$r6 then jump to 7
5   sub $r1 $r2 $r4       #$r4=$r1-$r2
6   jmp 8                 # jump to exit
7   add $r1 $r2 $r4       # $r4=$r1+$r2
8   END
```