North South University

Department of Electrical & Computer Engineering

ISA Design Proposal

Course Name: **CSE332- Computer Organization and Architecture**

Instructor Name: Dr. Tanzilur Rahman

Section: 02

Group Number: 01

Student Name	ID
1. Abid Ahsan Shanto	1921738042
2. Md. Baker	1911672642
3. Mariya Sharmin Tisha	1911637642

Introduction:

We are to design an ISA for a 10-bit RISC (single cycle) type CPU. The details and issues the CPU will address are described below:

➔ **Operands:**

We have decided to use two operands in our ISA. Most of them are register based as well as constants (MIXED). A total of 8 registers are used, each with a 3-bit address. A register table is given below outlining the details:

Register No.	Register Name	Assigned Value	Comments
$0	$zero	000	Grounded to zero
$01 - $04	$S0 - $S3	001 - 100	Saved Registers (General Purpose)
$05 - $07	$T0 - $T2	101 - 111	Temporary Registers (General Purpose)

➔ **Operations:**

A total of 16 operations were chosen based on the benchmark programs as well as feasibility of performing instructions. Based on the number of operations we have chosen, we have assigned an opcode of 4 bits to each of them.

Category	Operation	Type	Example	Meaning	Comments	Opcode
Arithmetic	add	R	Add $s1 $s2	$s1 = $s1 + $s2	Two register operands for adding values in registers	0000
	subtract	R	Sub $s1 $s2	$s1 = $s1 - $s2	Two register operands for subtracting values in registers	0001
	add immediate	I	Addi $s1 2	$s1 = $s2 + 2	Used for adding constants	0010
Data Transfer	load word	I	Lw $s0 4	$t0 ←mem[4 + $s0]	Word from memory to register	0011

	store word	I	Sw $s0 4	mem[4+ $s0] ←$t0	Word from register to memory	0100
Conditional branch	Branch Equal to Zero	I	Beqz $s0 L	if ($s0 == 0) go to L	Checks if the register value is equal to zero or not	0101
	Branch greater than zero	I	Bgtz $s0 L	If ($s0 > 0) Go to L	Checks if the register value is greater than zero	0110
	Branch less than zero	I	Bltz $s0 L	If ($s0 < 0) Go to L	Checks if the register value is smaller than zero	0111
	Branch not equal to zero	I	Bnez $s0 L	If ($s0 != 0) Go to L	Checks if the register value is not equal	1000

					to than zero or not	
	Branch greater or equal to zero	I	Bgez $s0 L	If ($s0 >= 0) Go to L	Checks if the register value is equal to or greater than zero or not	1001
Unconditional Jump	Jump	J	J target	J L	Jump to target address	1010
Logical	And	R	And $s0 $s1	$s0 = $s0 AND $s1	Logical and	1011
	Or	R	Or $s0 $s1	$s0 = $s0 OR $s1	Logical OR	1100
	Nor	R	Nor $s0 $s1	$s0 = !($s0 + S1)	Logical Nor	1101
I/O	Input	J	In $t0	$t0 ← address port	Stores user input to a register	1110

	Output	J	Out $t0	$t0→ address port	Outputs the result through a port	1111

➔ **Formats:**

The format for three different types of instruction types are given below along with their bit distribution:

➢ **R TYPE**

Opcode: 4 bits	**rs/rd: 3 bits**	**rt: 3 bits**

➢ **I TYPE**

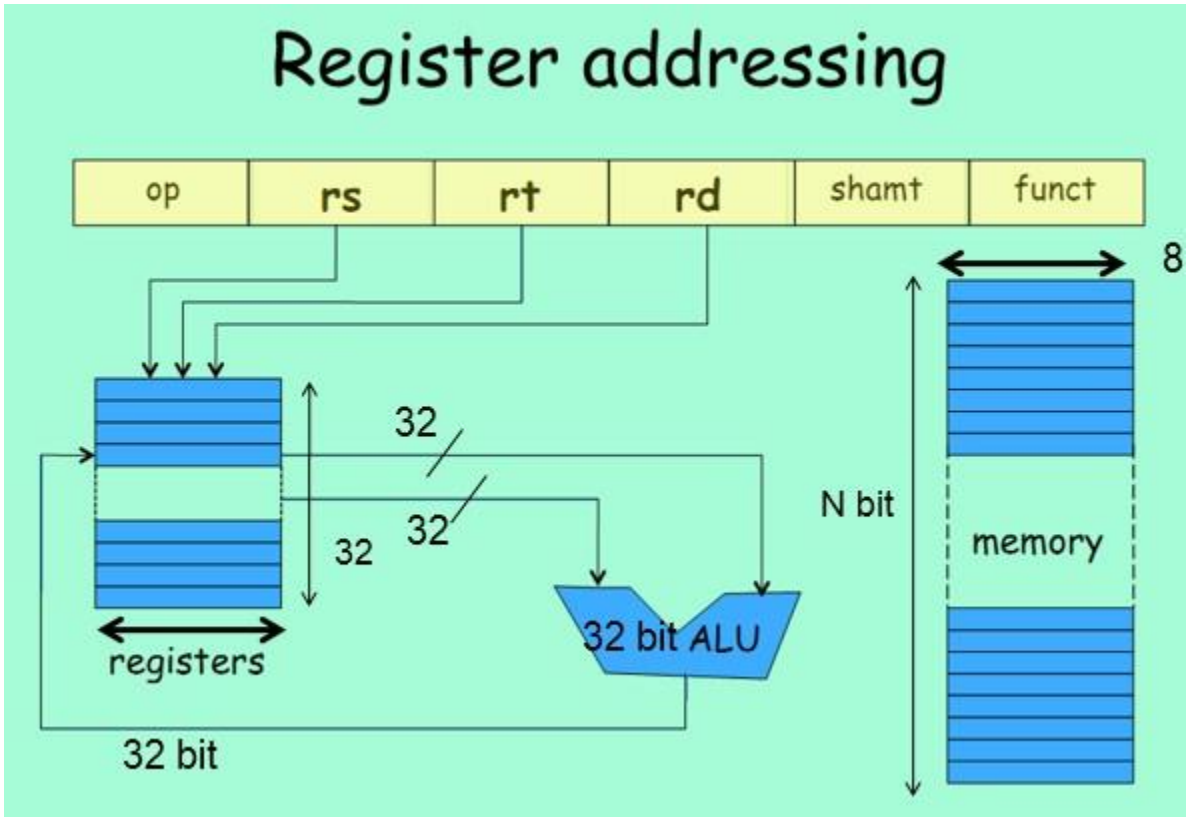
Opcode : 4 bits	**rs/rd: 3 bits**	**immediate: 3 bits**

➢ **J TYPE**

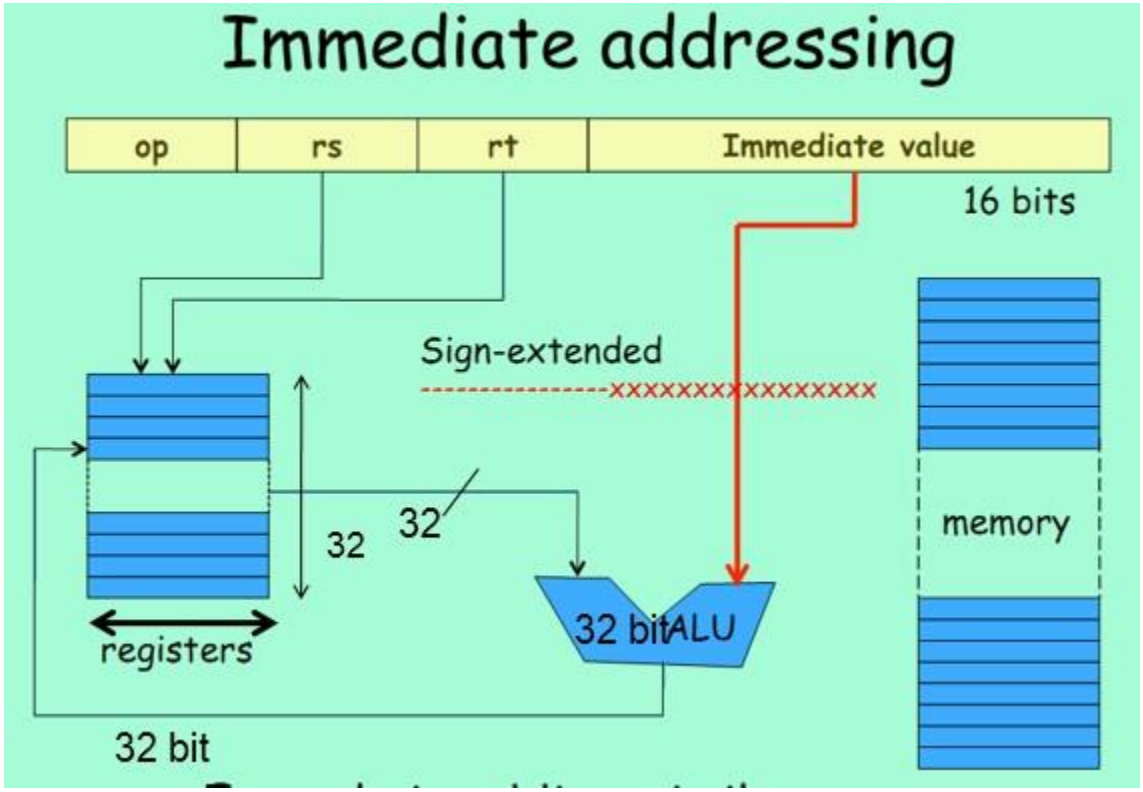
Opcode: 4 bits	**Target: 6 bits**

➔ **<u>Addressing Modes:</u>**

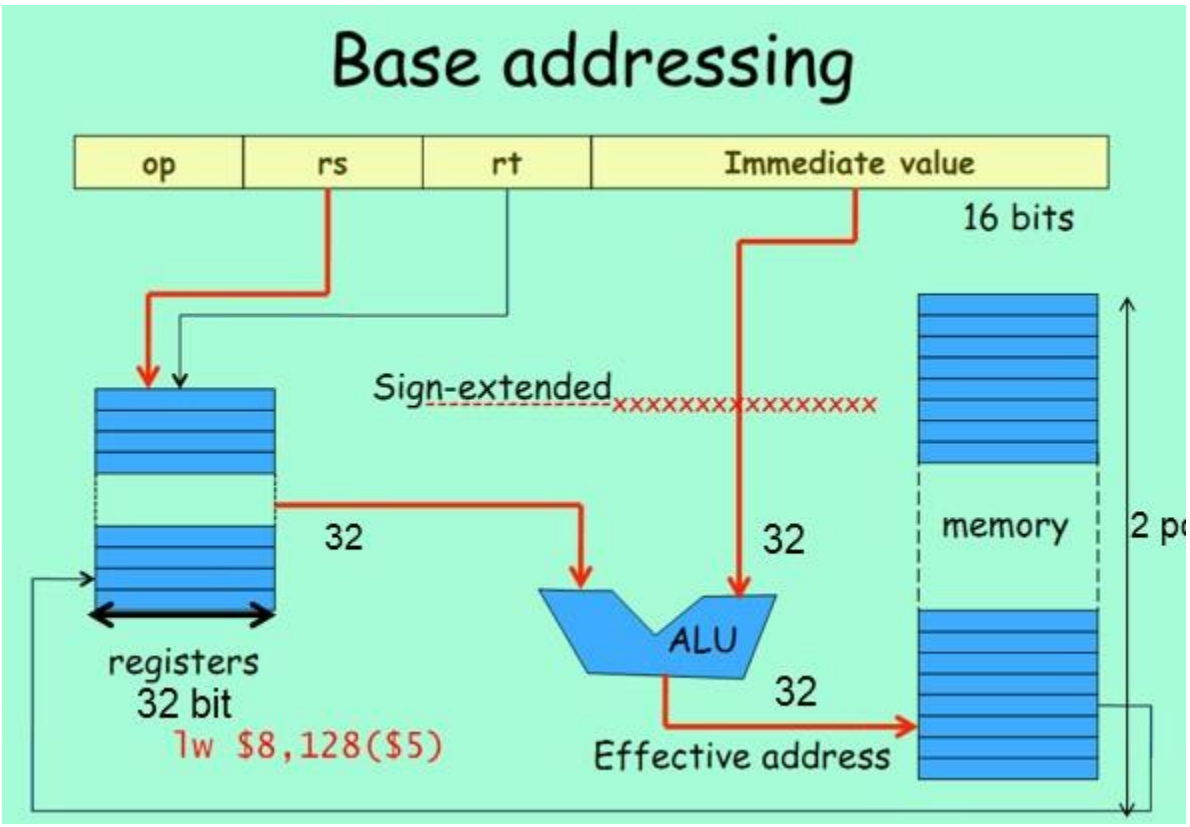
1. Register Addressing Mode:



2. Immediate Addressing:



3. Base- Addressing:



➔ **Example Codes for Benchmark Programs:**

- ## Simple Arithmetic & Logical:

A = A + B [A, B = $s0, $s1]
A = A && 0
A = A || 1
A = A x 3

Assembly CODE:

```
ADD $s0, $s1
AND $s0, $zero
AND $t0,$zero
ADDi $t0,1
OR $s0, $t0
AND $t0,$zero
ADD $t0,$s0
ADD $s0, $t0
ADD $s0,$t0
```

- ## Branch and loop:

```
for (I=0, I<5, i++)
{
        C= C+I
}
```

[$S1 = index i, $S2 = C]

Assembly CODE:

```
AND $S1, $zero
L1: AND $t0,$zero
ADDi $t0,5
```

```
SUB $t0, $S1

beqz $t0, E

Add $S2,$S1

Addi $S1,1

J L1

E
```