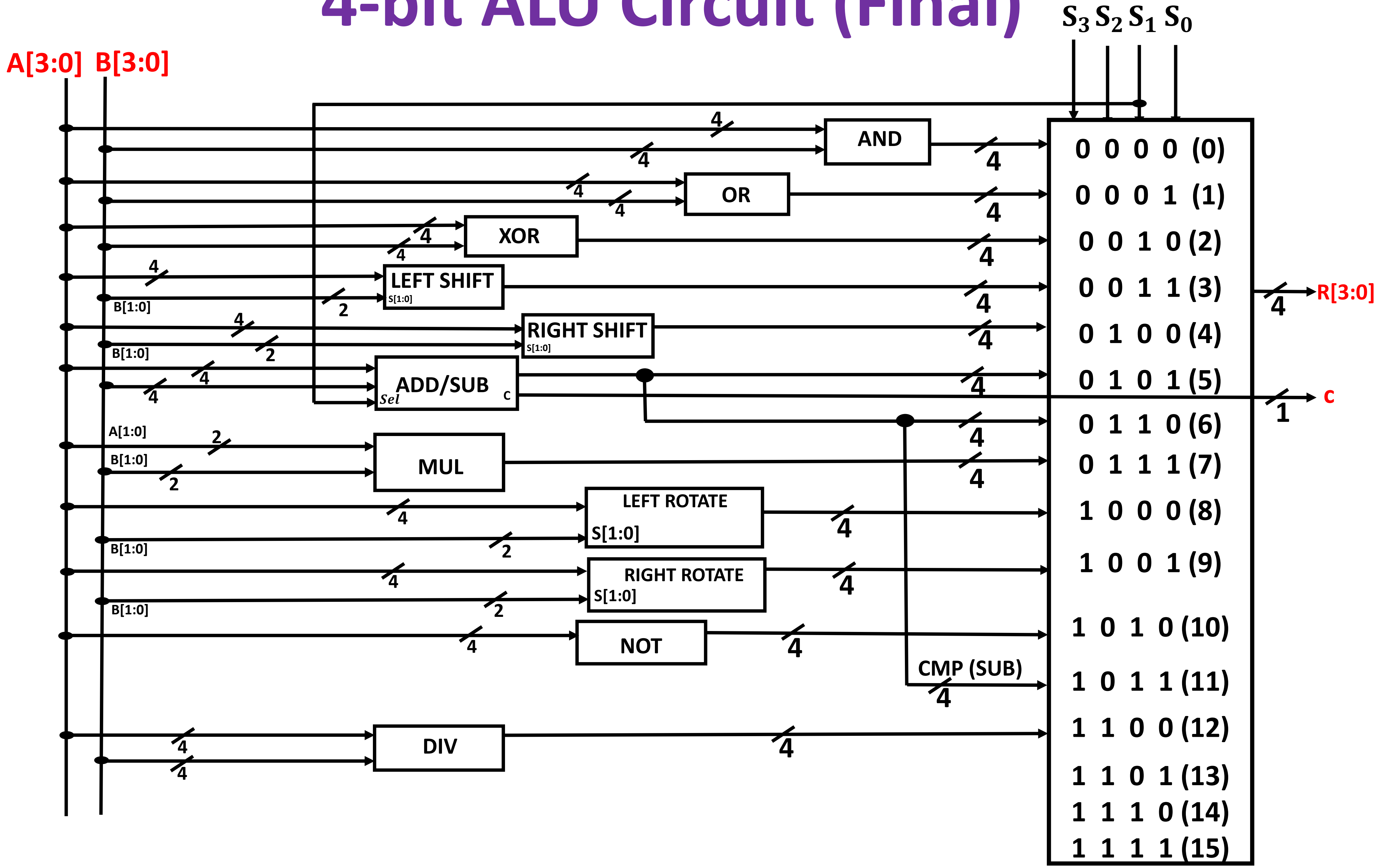
ISA Design

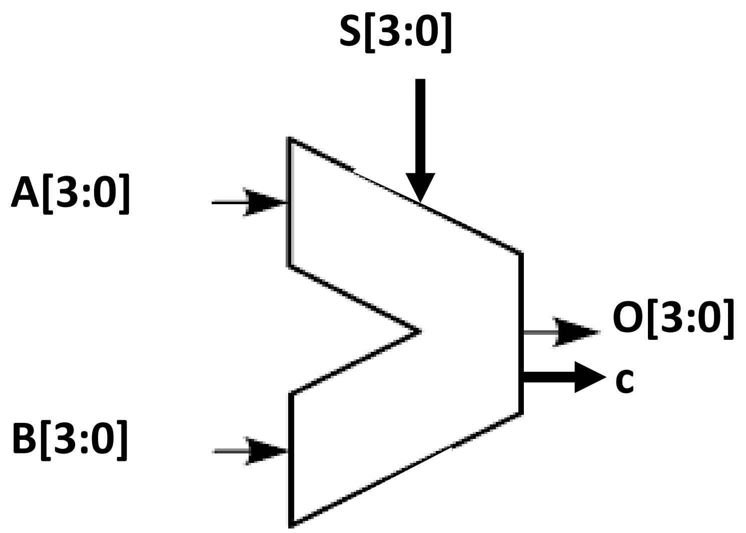
Nahin Ul Sadad
Lecturer
CSE, RUET

ALU and Register Set Review

4-bit ALU Circuit (Final)



4-bit ALU Circuit

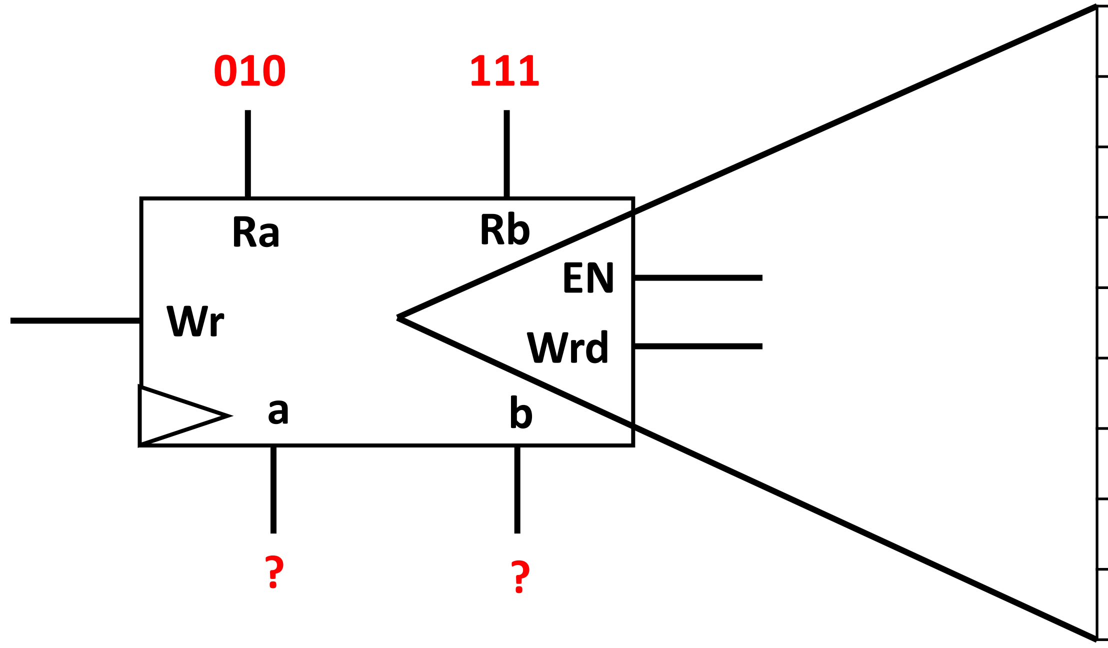


Operation	Selection lines			
	S_3	S_2	S_1	S_0
AND	0	0	0	0
OR	0	0	0	1
XOR	0	0	1	0
NOT	1	0	1	0
ADD	0	1	0	1
SUB	0	1	1	0
MUL	0	1	1	1
DIV	1	1	0	0
SHL	0	0	1	1
SHR	0	1	0	0
ROL	1	0	0	0
ROR	1	0	0	1
CMP	1	0	1	1

Here,
A[3:0] is data to be shifted or rotated.
And B[1:0] is number of shift/rotate (Max 3).

Here, CMP is same as SUB but it doesn't update register value.

4-bit Register Set (Reading)



Address	Register	Value stored
000	R0	0111(7)
001	R1	0100(4)
010	R2	0011(3)
011	R3	0111(7)
100	R4	1000(8)
101	R5	0001(1)
110	R6	1000(8)
111	R7	0010(2)

Suppose, Register set has eight 4-bit registers.

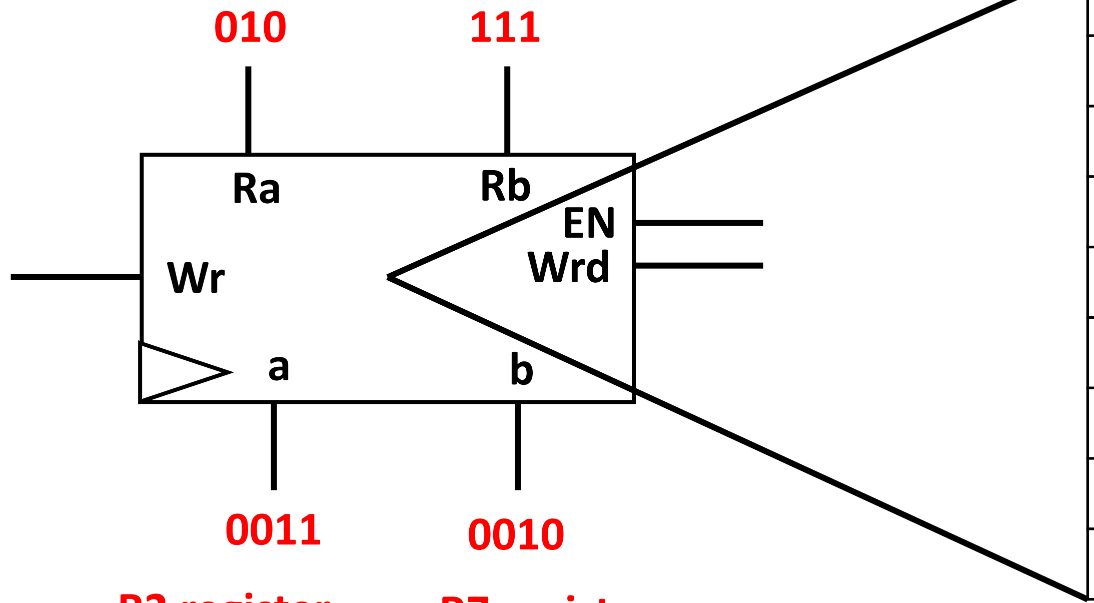
Since Register Set has 8 registers, it will need $log_2 8 = 3$ address lines.

Since registers are 4-bit registers, so it will store 4-bit value.

Reading in Register Set

Ra will take address of register and show value stored in that register in a (4-bit value).
and
Rb will take address of register and show value stored in that register in b (4-bit value).

4-bit Register Set (Reading)



Address	Register	Value stored
000	R0	0111 (7)
001	R1	0100 (4)
010	R2	0011 (3)
011	R3	0111 (7)
100	R4	1000 (8)
101	R5	0001 (1)
110	R6	1000 (8)
111	R7	0010 (2)

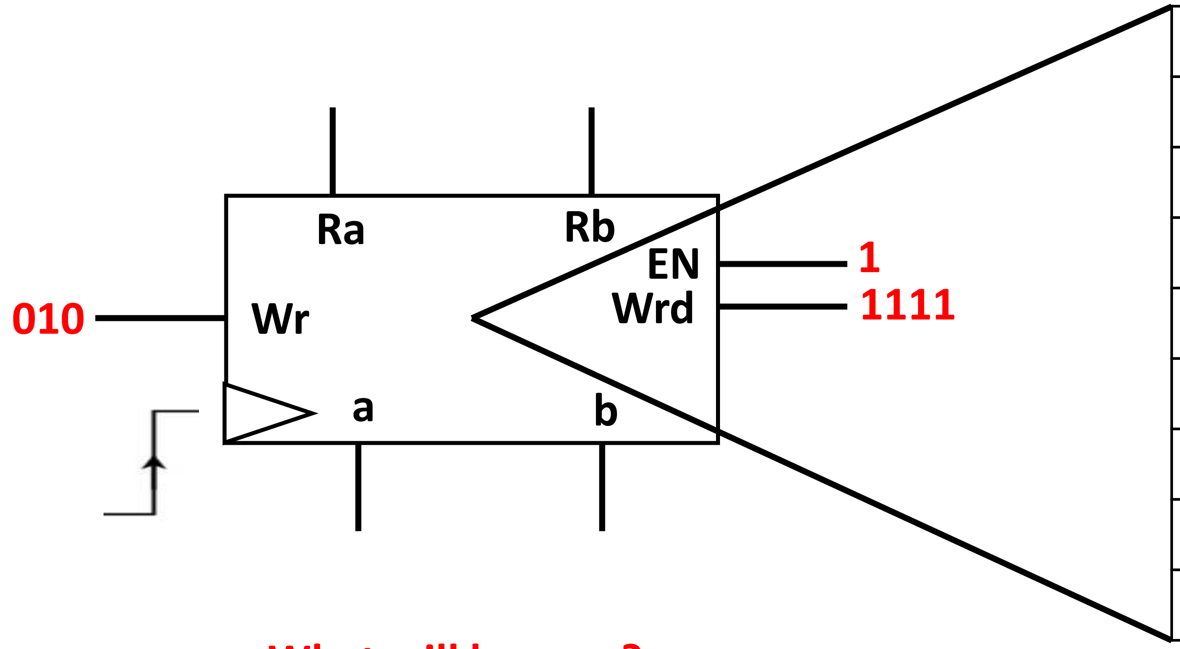
R2 register has 0011 (3) value stored.

R7 register has 0010 (2) value stored.

Reading in Register Set

Ra will take address of register and show value stored in that register in a (4-bit value).
and
Rb will take address of register and show value stored in that register in b (4-bit value).

4-bit Register Set (Writing)



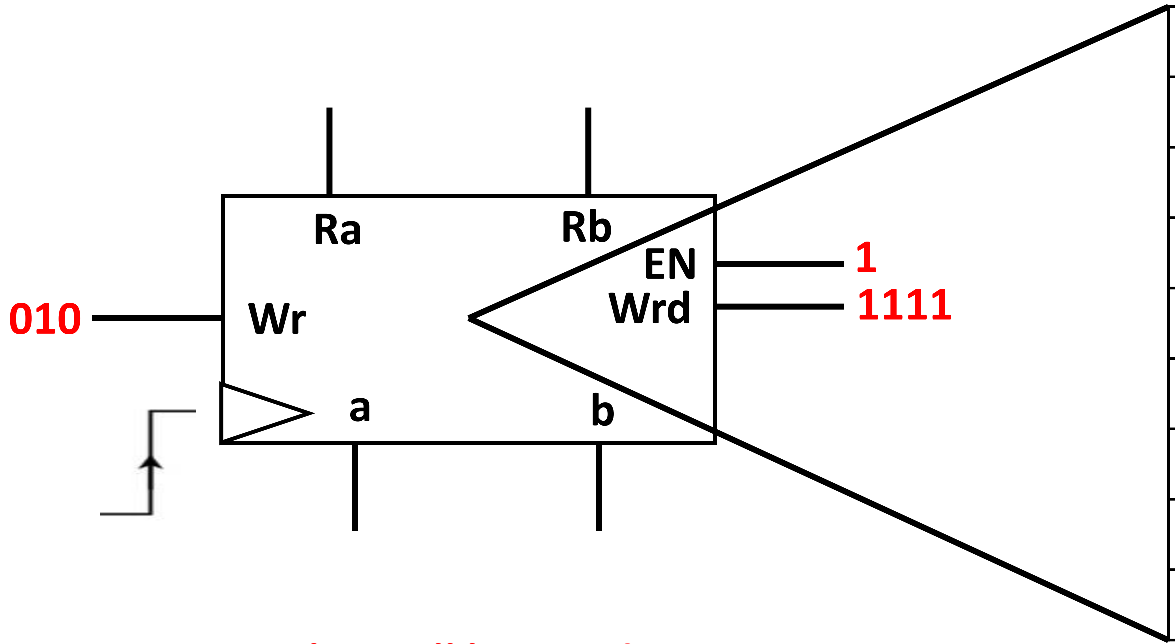
Address	Register	Value stored
000	R0	0111 (7)
001	R1	0100 (4)
010	R2	0011 (3)
011	R3	0111 (7)
100	R4	1000 (8)
101	R5	0001 (1)
110	R6	1000 (8)
111	R7	0010 (2)

What will happen?

EN will enable/disable writing operation in register set.
(1-Enable/0-Disable)
and
Wr will take address of register to be written.
and
Wrd will take (4-bit value) value to be written in Wr register.

Writing in
Register Set

4-bit Register Set (Writing)



Address	Register	Value stored
000	R0	0111 (7)
001	R1	0100 (4)
010	R2	1111 (15)
011	R3	0111 (7)
100	R4	1000 (8)
101	R5	0001 (1)
110	R6	1000 (8)
111	R7	0010 (2)

What will happen?

EN will enable/disable writing operation in register set. (1-Enable/0-Disable)
and
Wr will take address of register to be written.
and
Wrd will take (4-bit value) value to be written in Wr register.

Writing in Register Set

ISA Design

Instruction Set Architecture (ISA)

The Instruction Set Architecture (ISA)/Architecture is the part of the processor that is visible to the programmer or compiler writer. The ISA specifies the behavior of machine code running on implementations of that ISA. The ISA serves as the boundary between software and hardware.

In general, an ISA defines the supported data types, the registers, the hardware support for managing main memory, fundamental features (such as the memory consistency, addressing modes, virtual memory), and the input/output model of a family of implementations of the ISA.22

For example, ISA of Intel/AMD CPU is x86-64 architecture, ISA of ARM CPU is ARM architecture etc.

Instruction Set Architecture (ISA)

Format of ISA is usually like:

Opcode	Operands

ISA design depends on several criteria:

1. Word size of CPU
   (Ex: 2-bit/4-bit/8-bit etc.)
2. No. of registers
   (Ex: 2 registers/4 registers/ 16 registers etc.
3. No. of ALU operations supported
   (Ex: ALU supports 8/10/16 operations etc.)
4. Types of instructions supported
   (Ex: Arithmetic, Logic, Branching, Memory operations etc.)

ISA Design (Opcode)

Opcode of our ISA will depend on two things:
1. Types of instructions supported
2. No. of ALU operations supported

Our CPU supports 4 types of instructions:
1. Arithmetic & Logic Instructions (Register Mode)
2. Arithmetic & Logic Instructions (Immediate Mode)
3. Branching
4. Memory & IO Operations

Our ALU supports 12 operations total: AND, OR, XOR, ADD, SUB, SHL, SHR, MUL, ROL, ROR, NOT and DIV.

ISA Design (Opcode)

So, our CPU has:

1. Types of instructions supported (**4**) $\longrightarrow \log_2 4 = 2$
2. No. of ALU operations supported (**12**) $\longrightarrow \log_2 12 \approx 4$

Format of ISA is usually like:

Opcode	Operands

So, Opcode will be 6 bits in our ISA.

2 bits	4 bits
Types of instruction	Operations (ALU selection lines)

$\Rightarrow$

6 bits
Opcode

ISA Design (Operands)

Our operands of ISA design will depend on two things:
1. No. of registers in register set
2. Word size of CPU

No. of registers in CPU is 8. So, no of bits need to address 8 registers is $\log_2 8 = 3$ and Word size of CPU is 4-bits. So, size of value will be 4-bits too.

For Arithmetic & Logic instructions (Register Mode), consider
```
ADD R2, R3
```
So, we need to select two registers (3 bits each).

For Arithmetic & Logic instructions (Immediate Mode), consider
```
ADD R2, 3
```
So, we need to select one register (3 bits) and one value (4 bits).

ISA Design (Operands)

So, our CPU has:

1. No. of registers in register set (**8**) $\longrightarrow$ $\log_2 8 = 3$
2. Word size of CPU (**4**) $\longrightarrow$ **4 bits (value)**

Format of ISA is usually like:

Opcode	Operands

For Arithmetic & Logic instructions (Register Mode),

3 bits	3 bits
Register 1	Register 2

$\Rightarrow$

6 bits
Operands

For Arithmetic & Logic instructions (Immediate Mode),

3 bits	4 bits
Register 1	Value

$\Rightarrow$

7 bits
Operands

ISA Design (Operands)

Format of ISA is usually like:

6 bits	7 bits
Opcode	Operands

3 bits	3 bits	1 bit
Register 1	Register 2	Unused

3 bits	4 bits
Register 1	Value

2 bits	4 bits
Types of instruction	Operations (ALU selection lines)

Size of our ISA will be 6+7 = 13 bits

Types of Instruction

There will be 4 types of instruction:

Types of instruction	Opcode (First 2 bits)	Example Assembly

Arithmetic & Logic Instruction (Register Mode)	`00`	`ADD R0, R1`
		`XOR R2, R3`
		`SHL R5, 1`

Arithmetic & Logic Instruction (Immediate Mode)	`01`	`ADD R0, 5`
		`XOR R2, 6`

Branching	`10`	`JMP LABEL`
		`JC LABEL`

Memory and Input/Output Operations	`11`	`LOAD R0, [R1]`
		`STORE R1, 10`
		`ACCEPT_INPUT`

ISA of Arithmetic & Logic Instruction (Register Mode)

Opcode (6 bit)		Register 1	Register 2	Unused
2 bits	4 bits	3 bits	3 bits	1 bit
Types of instruction	Operations (ALU selection lines)	Ra (000-111)	Rb (000-111)	X

Convention,
Ra = Ra OP Rb

12 7	6 4	3 1	0
00XXXX (Opcode)	Register 1	Register 2	Unused

ISA of Arithmetic & Logic Instruction (Register Mode)

Opcode		Register 1	Register 2	Assembly Example
Type (2 bits)	Operations (4 bits)	3 bits	3 bits	
00	0000 (AND)	000-100 (R0-R4)	000-100 (R0-R4)	AND R0, R1
	0001 (OR)	000-100 (R0-R4)	000-100 (R0-R4)	OR R1, R2
	0010 (XOR)	000-100 (R0-R4)	000-100 (R0-R4)	XOR R2, R3
	0011 (SHL)	000-100 (R0-R4)	000-100 (R0-R4)	SHL R3, R1
	0100 (SHR)	000-100 (R0-R4)	000-100 (R0-R4)	SHR R4, R2
	0101 (ADD)	000-100 (R0-R4)	000-100 (R0-R4)	ADD R4, R4
	0110 (SUB)	000-100 (R0-R4)	000-100 (R0-R4)	SUB R4, R4
	0111 (MUL)	000-100 (R0-R4)	000-100 (R0-R4)	MUL R3, R4
	1000 (ROL)	000-100 (R0-R4)	000-100 (R0-R4)	ROL R0, R2
	1001 (ROR)	000-100 (R0-R4)	000-100 (R0-R4)	ROR R1, R2
	1010 (NOT)	000-100 (R0-R4)	000-100 (R0-R4)	NOT R4
	1011 (CMP)	000-100 (R0-R4)	000-100 (R0-R4)	CMP R4, R2
	1100 (DIV)	000-100 (R0-R4)	000-100 (R0-R4)	DIV R4, R2

Note: R5-R7 are Special Purpose Registers. That's why they are not included.

This instruction do operation on single register. (They do not need 2nd register [3:1])

Example: ISA of Arithmetic & Logic Instruction (Register Mode)

12	11	10	09	08	07	06	05	04	03	02	01	00
00XXXX (Opcode)						Register 1			Register 2			Unused

Question:

Translate following assembly code to machine code using ISA:

`1. XOR R2, R3`

Answer:

Since type of instruction is Arithmetic & Logic (Register Mode), so opcode of first two bits[12:11] is 00.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	x	x	x	x	x	x	x	x	x	x	x

Since selection lines of XOR operation in ALU is 0010, so rest of opcode[10:7] is 0010

12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	1	0	x	x	x	x	x	x	x

Since first register is R2, it will be register 1[6:4] in ISA and its value in binary is 010.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	1	0	0	1	0	x	x	x	x

Since second register is R3, it will be register 2[3:1] in ISA and its value in binary is 011.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	1	0	0	1	0	0	1	1	x

Unused bit[0] will not be used in this operation. It can be anything 0/1.

Example: ISA of Arithmetic & Logic Instruction (Register Mode)

12	11	10	09	08	07	06	05	04	03	02	01	00
00XXXX (Opcode)						Register 1			Register 2			Unused

Question:

Translate following assembly code to machine code using ISA:

2. CMP R2, R3

Answer:

Since type of instruction is Arithmetic & Logic (Register Mode), so opcode of first two bits[12:11] is 00.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	x	x	x	x	x	x	x	x	x	x	x

Since selection lines of XOR operation in ALU is 1011, so rest of opcode[10:7] is 1011.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	1	0	1	1	x	x	x	x	x	x	x

Since first register is R2, it will be register 1[6:4] in ISA and its value in binary is 010.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	1	0	1	1	0	1	0	x	x	x	x

Since second register is R3, it will be register 2[3:1] in ISA and its value in binary is 011.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	1	0	1	1	0	1	0	0	1	1	x

Unused bit[0] will not be used in this operation. It can be anything 0/1.

Example: ISA of Arithmetic & Logic Instruction (Register Mode)

12	11	10	09	08	07	06	05	04	03	02	01	00
00XXXX (Opcode)						Register 1			Register 2			Unused

Question:
Translate following assembly code to machine code using ISA:

`3. SHL R2, R1`

Answer:
Since type of instruction is Arithmetic & Logic (Register Mode), so opcode of first two bits[12:11] is 00.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	x	x	x	x	x	x	x	x	x	x	x

Since selection lines of XOR operation in ALU is 0011, so rest of opcode[10:7] is 0011.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	1	1	x	x	x	x	x	x	x

Since first register is R2, it will be register 1[6:4] in ISA and its value in binary is 010.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	1	1	0	1	0	x	x	x	x

Since second register is R1, it will be register 2[3:1] in ISA and its value in binary is 001.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	1	1	0	1	0	0	0	1	x

Unused bit[0] will not be used in this operation. It can be anything 0/1.

Example: ISA of Arithmetic & Logic Instruction (Register Mode)

12	11	10	09	08	07	06	05	04	03	02	01	00
00XXXX (Opcode)						Register 1			Register 2			Unused

Question:

4. NOT R4

Answer:

Since type of instruction is Arithmetic & Logic (Register Mode), so opcode of first two bits[12:11] is 00.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	X	X	X	X	X	X	X	X	X	X	X

Since selection lines of NOT operation in ALU is 1100, so rest of opcode[10:7] is 1100.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	1	1	0	0	X	X	X	X	X	X	X

Since first register is R4, it will be register 1[6:4] in ISA and its value in binary is 100.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	1	1	0	0	1	0	0	X	X	X	X

We don't need Register 2[3:1]. Because it is a single register operation.
Unused bit[0] will not be used in this operation.
These bits value can be anything.

ISA of Arithmetic & Logic Instruction (Immediate Mode)

Opcode (6 bit)		Register 1	Constant
2 bits	4 bits	3 bits	4 bits
Types of instruction	Operations (ALU selection lines)	Ra (000-111)	Value (0000-1111)

Convention,
Ra = Ra OP Constant

12 7	6 4	3 0
01XXXX (Opcode)	Register 1	Value

ISA of Arithmetic & Logic Instruction (Immediate Mode)

Opcode		Register 1	Constant	Assembly Example
Type (2 bits)	Operations (4 bits)	3 bits	4 bits	
01	0000 (AND)	000-100 (R0-R4)	0000-1111 (0-15)	AND R0, 1
	0001 (OR)	000-100 (R0-R4)	0000-1111 (0-15)	OR R1, 2
	0010 (XOR)	000-100 (R0-R4)	0000-1111 (0-15)	XOR R2, 3
	0011 (SHL)	000-100 (R0-R4)	0000-1111 (0-15)	SHL R3, 3 (MAX 3)
	0100 (SHR)	000-100 (R0-R4)	0000-1111 (0-15)	SHR R4, 2(MAX 3)
	0101 (ADD)	000-100 (R0-R4)	0000-1111 (0-15)	ADD R4, 4
	0110 (SUB)	000-100 (R0-R4)	0000-1111 (0-15)	SUB R3, 5
	0111 (MUL)	000-100 (R0-R4)	0000-1111 (0-15)	MUL R1, 3 (MAX 3)
	1000 (ROL)	000-100 (R0-R4)	0000-1111 (0-15)	ROL R0, 1 (MAX 3)
	1001 (ROR)	000-100 (R0-R4)	0000-1111 (0-15)	ROR R1, 2 (MAX 3)
	1010 (NOT)	000-100 (R0-R4)	0000-1111 (0-15)	NOT R4
	1011 (CMP)	000-100 (R0-R4)	0000-1111 (0-15)	CMP R1, 7
	1100 (DIV)	000-100 (R0-R4)	0000-1111 (0-15)	DIV R1, 7

Note: R5-R7 are Special Purpose Registers. That's why they are not included.

These instructions do operations on single register and they do the same thing as they do in register mode. (They do not need value [3:0])

Example: ISA of Arithmetic & Logic Instruction (Immediate Mode)

12	11	10	09	08	07	06	05	04	03	02	01	00
01XXXX (Opcode)						Register 1			Value			

Question:

Translate following assembly code to machine code using ISA:

`1. XOR R2, 3`

Answer:

Since type of instruction is Arithmetic & Logic (Immediate Mode), so opcode of first two bits[12:11] is 01.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	1	x	x	x	x	x	x	x	x	x	x	x

Since selection lines of XOR operation in ALU is 0010, so rest of opcode[10:7] is 0010

12	11	10	09	08	07	06	05	04	03	02	01	00
0	1	0	0	1	0	x	x	x	x	x	x	x

Since first register is R2, it will be register 1[6:4] in ISA and its value in binary is 010.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	1	0	0	1	0	0	1	0	x	x	x	x

Since value is 3, it will be value[3:0] in ISA and its value in binary is 0011.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	1	0	0	1	0	0	1	0	0	0	1	1

Example: ISA of Arithmetic & Logic Instruction (Immediate Mode)

12	11	10	09	08	07	06	05	04	03	02	01	00
01XXXX (Opcode)						Register 1			Value			

Question:

2. **NOT R4**

Answer:

Since type of instruction is Arithmetic & Logic (Immediate Mode), so opcode of first two bits[12:11] is 01.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	1	X	X	X	X	X	X	X	X	X	X	X

Since selection lines of NOT operation in ALU is 1100, so rest of opcode[10:7] is 1100.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	1	1	1	0	0	X	X	X	X	X	X	X

Since first register is R4, it will be register 1[6:4] in ISA and its value in binary is 100.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	1	1	1	0	0	1	0	0	X	X	X	X

We don't need value[3:0]. Because it is a single register operation. These bits can be anything.

Example: ISA of Arithmetic & Logic Instruction (Immediate Mode)

12	11	10	09	08	07	06	05	04	03	02	01	00
01XXXX (Opcode)						Register 1			Value			

Question:

3. `ROL R4, 3`

Answer:

Since type of instruction is Arithmetic & Logic (Immediate Mode), so opcode of first two bits[12:11] is 01.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	1	X	X	X	X	X	X	X	X	X	X	X

Since selection lines of ROL operation in ALU is 1000, so rest of opcode[10:7] is 1000.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	1	1	0	0	0	X	X	X	X	X	X	X

Since first register is R4, it will be register 1[6:4] in ISA and its value in binary is 100.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	1	1	0	0	0	1	0	0	X	X	X	X

Since value is 3, it will be value[3:0] in ISA and its value in binary is 0011.

12	11	10	09	08	07	06	05	04	03	02	01	00
0	1	1	0	0	0	1	1	1	0	0	1	1

ISA of Branching

For Every Jump instructions except JMPREG,

Opcode (6 bit)		Address
2 bits	4 bits	7 bits
Types of instruction	Operations	Value (0000000-1111111)

12 7	6 0
10xxxx (Opcode)	Address

For JMPREG instruction,

Opcode (6 bit)		Register 1	Unused
2 bits	4 bits	3 bits	4 bits
Types of instruction	Operati ons	Ra (000-111)	XXXX

ISA of Branching

Opcode		Address	Register 1	Assembly Example
Type (2 bits)	Operations (4 bits)	7 bits	3 bits	
10	0000 (JMP)	0000000-1111111 (0-127)	X	JMP LABEL
	0001 (JE)	0000000-1111111 (0-127)	X	JE LABEL
	0010 (JNE)	0000000-1111111 (0-127)	X	JNE LABEL
	0011 (JL)	0000000-1111111 (0-127)	X	JL LABEL
	0100 (JLE)	0000000-1111111 (0-127)	X	JLE LABEL
	0101 (JG)	0000000-1111111 (0-127)	X	JG LABEL
	0110 (JGE)	0000000-1111111 (0-127)	X	JGE LABEL
	0111 (JC)	0000000-1111111 (0-127)	X	JC LABEL
	1000 (JNC)	0000000-1111111 (0-127)	X	JNC LABEL
	1001 (JZ)	0000000-1111111 (0-127)	X	JZ LABEL
	1010 (JNZ)	0000000-1111111 (0-127)	X	JNZ LABEL
	1011 (JMPREG) (It jumps to address saved in R0-R4 register)	X	000-100 (0-4) (Last 4 bits are unused)	JMPREG R1
	1100 (XXX)	X	X	XXX
	1101 (XXX)	X	X	XXX

Example: ISA of Branching

12	11	10	09	08	07	06	05	04	03	02	01	00
10XXXX (Opcode)						Address						

Question:

Translate following assembly code to machine code using ISA:

1. `JMP 3(LABEL)`

Answer:

Since type of instruction is Branching, so opcode of first two bits[12:11] is 10.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	0	x	x	x	x	x	x	x	x	x	x	x

Since opcode of JMP is 0000, so rest of opcode[10:7] is 0000.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	0	0	0	0	0	x	x	x	x	x	x	x

Since address is 3, it will be in binary [6:0] is 0000011.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	0	0	0	0	0	0	0	0	0	0	1	1

Example: ISA of Branching

12	11	10	09	08	07	06	05	04	03	02	01	00
10XXXX (Opcode)						Address						

Question:

Translate following assembly code to machine code using ISA:

2. **JLE 100(LABEL)**

Answer:

Since type of instruction is Branching, so opcode of first two bits[12:11] is 10.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	0	X	X	X	X	X	X	X	X	X	X	X

Since opcode of JMP is 0100, so rest of opcode[10:7] is 0100.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	0	0	1	0	0	X	X	X	X	X	X	X

Since address is 100, it will be in binary [6:0] is 1100100.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	0	0	1	0	0	1	1	0	0	1	0	0

Example: ISA of Branching

12	11	10	09	08	07	06	05	04	03	02	01	00
10XXXX (Opcode)						Register 1			Unused			

Question:

Translate following assembly code to machine code using ISA:

3. JMPREG R1

Answer:

Since type of instruction is Branching, so opcode of first two bits[12:11] is 10.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	0	X	X	X	X	X	X	X	X	X	X	X

Since opcode of JMP is 1011, so rest of opcode[10:7] is 1011.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	0	1	0	1	1	X	X	X	X	X	X	X

Since Register 1 is 001, it will be in binary [6:0] is 001XXXX.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	0	1	0	1	1	0	0	1	X	X	X	X

Memory Instructions

Addressing Modes

Modes	Example
Register Mode	`ADD R0, R1`
Immediate Mode	`ADD R0, 2`

Direct Mode	`LOAD R0, [2] /`
	`STORE [3], R0`

Register Indirect Mode/Indexed Mode/Based Mode	`LOAD R0, [R2] /`
	`STORE [R3], R0`

Based Indexed Mode	`LOAD R0, [R2+R0] /`
	`STORE [R3+R4], R0`

Based/Indexed with Displacement Mode	`LOAD R0, [R2+2] /`
	`STORE [R3+2], R0`

Based indexed with Displacement Mode	`LOAD R0, [R2+R0+2] /`
	`STORE [R3+R0+3], R0`

Memory Instructions

Addressing Modes Examples

Direct Addressing Mode

`LOAD R0, [2]`

Main Memory (RAM)

Memory Address	Memory Contents
000 (0)	000 000 0000 100 (4)
001 (1)	000 000 0000 010 (2)
010 (2)	000 000 0000 101 (5)
011 (3)	000 000 0000 111 (7)
100 (4)	000 000 0000 110 (6)
101 (5)	000 000 0000 011 (3)
110 (6)	000 000 0001 010 (10)

Step 1

Step 2

Register Set

Register Address	Register Name	Register Contents
000 (0)	**R0**	**0101 (5)**
001 (1)	R1	0001 (1)
010 (2)	R2	0000 (0)
011 (3)	R3	0100 (4)
100 (4)	R4	0101 (5)
101 (5)	R5	0010 (2)
110 (6)	R6	0010 (2)

Step 1: `[2] = 5`
Step 2: `R0 = 5`

Memory Instructions

Addressing Modes Examples

Register Indirect Mode/Indexed Mode/Based Mode

`LOAD R0, [R2]`

Main Memory (RAM)

Memory Address	Memory Contents
000 (0)	000 000 0000 100 (4)
001 (1)	000 000 0000 010 (2)
010 (2)	000 000 0000 101 (5)
011 (3)	000 000 0000 111 (7)
100 (4)	000 000 0000 110 (6)
101 (5)	000 000 0000 011 (3)
110 (6)	000 000 0001 010 (10)

Step 1

Step 2

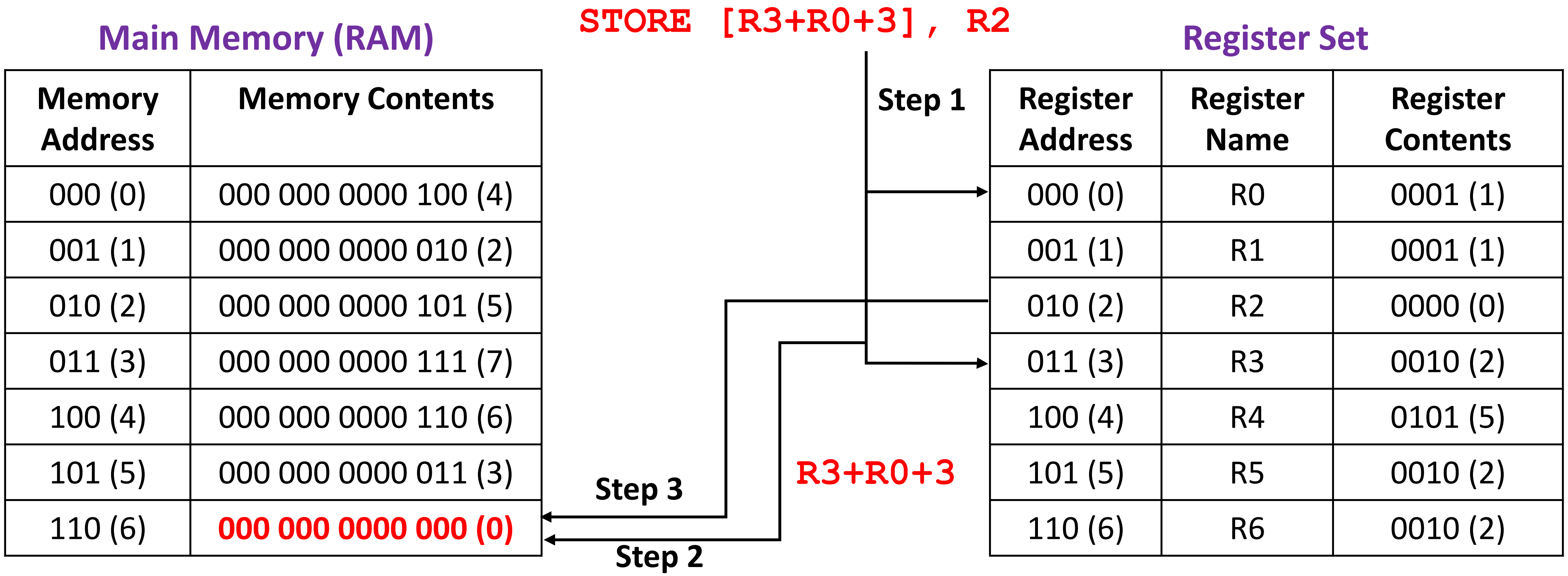
Step 3

Register Set

Register Address	Register Name	Register Contents
000 (0)	R0	0100 (4)
001 (1)	R1	0001 (1)
010 (2)	R2	0000 (0)
011 (3)	R3	0100 (4)
100 (4)	R4	0101 (5)
101 (5)	R5	0010 (2)
110 (6)	R6	0010 (2)

Step 1: `R2=0`
Step 2: `[R2]=[0]=4`
Step3: `R0 = 4`

Memory Instructions
Addressing Modes Examples
Register Indirect Mode/Indexed Mode/Based Mode



Main Memory (RAM)

Memory Address	Memory Contents
000 (0)	000 000 0000 100 (4)
001 (1)	000 000 0000 010 (2)
010 (2)	000 000 0000 101 (5)
011 (3)	000 000 0000 111 (7)
100 (4)	000 000 0000 110 (6)
101 (5)	000 000 0000 011 (3)
110 (6)	000 000 0000 000 (0)

Register Set

Register Address	Register Name	Register Contents
000 (0)	R0	0001 (1)
001 (1)	R1	0001 (1)
010 (2)	R2	0000 (0)
011 (3)	R3	0010 (2)
100 (4)	R4	0101 (5)
101 (5)	R5	0010 (2)
110 (6)	R6	0010 (2)

Step 1: `R3=2,R0=1, R3+R0+3=2+1+3=6`
Step 2: `[R3+R0+3]=[6]`
Step3: `[6] = R2 = 0`

IO Instructions

IO Operations

Operations	Example
Waiting for inputs. Receive input data from Input Register	`ACCEPT_INPUT`
Send data to be printed to Output Register	`PRINT_OUTPUT`
Clear output display	`PRINT_CLEAR`

ISA of Memory Instructions

For Direct Mode (`LOAD R0, [2]/STORE [3], R0`),

Opcode (6 bit)		Register 1	Address/Displacement
2 bits	4 bits	3 bits	4 bits
(11) Types of instruction	Operations	Ra(000-111)	Disp (0000-1111)

For Register Indirect Mode (`LOAD R0, [R2]/STORE [R3], R0`),

Opcode (6 bit)		Register 1	Register 2	Unused
2 bits	4 bits	3 bits	3 bits	1 bits
(11) Types of instruction	Operations	Ra(000-111)	Rb (000-111)	X

For Based with Displacement Mode (`LOAD R0, [R2+2]/STORE [R3+2], R0`),

Opcode (6 bit)		Register 1	Register 2	Displacement
2 bits	4 bits	3 bits	2 bits	2 bits
(11) Types of instruction	Operations	Ra(000-111)	Rb (00-11)	Disp(00-11)

ISA of IO Instructions

For IO Operations (`ACCEPT_INPUT`/ `PRINT_OUTPUT`/ `PRINT_CLEAR`),

Opcode (6 bit)		Unused
2 bits	4 bits	7 bits
(11) Types of instruction	Operations	XXXXXXX

ISA of Memory & IO Instructions

Opcode			Register 1 RA	Register 2 RB	Address Disp	Assembly Example
Type (2 bits)	Operations (4 bits)	Type of Operations				
11	0000 (LOAD)	Direct Mode	000-111 (0-7)	X	0000-1111 (0-15)	LOAD RA,[Disp]
	0001 (LOAD)	Register Indirect Mode/Indexed Mode/Based Mode	000-111 (0-7)	000-111 (0-7)	X	LOAD RA, [RB]
	0010 (LOAD)	Based/Indexed with Displacement Mode	000-111 (0-7)	00-11 (0-3)	00-11 (0-3)	LOAD RA, [RB+Disp]
	0011 (STORE)	Direct Mode	000-111 (0-7)	X	0000-1111 (0-15)	STORE [Disp], RA
	0100 (STORE)	Register Indirect Mode/Indexed Mode/Based Mode	000-111 (0-7)	000-111 (0-7)	X	STORE [RB], RA
	0101 (STORE)	Based/Indexed with Displacement Mode	000-111 (0-7)	00-11 (0-3)	00-11 (0-3)	STORE [RB+Disp], RA
	1101 (ACCEPT_INPUT)	Waiting for inputs. Send input data to Input Register	X	X	X	ACCEPT_INPUT
	1110 (PRINT_OUTPUT)	Send data to be printed to Output Register	X	X	X	PRINT_OUTPUT
	1111 (PRINT_CLEAR	Clear output display	X	X	X	PRINT_CLEAR

Example: ISA of Memory & IO Instructions

12	11	10	09	08	07	06	05	04	03	02	01	00
01XXXX (Opcode)						Register 1(RA)			Address			

Question:

1. `LOAD R3, [2]` (Format: `LOAD RA,[Disp]`)

Answer:

Since type of instruction is Memory & IO operations, so opcode of first two bits[12:11] is 11.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	X	X	X	X	X	X	X	X	X	X	X

Since opcode of this LOAD (Direct mode) operation is 0000, so rest of opcode[10:7] is 0000.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	0	0	0	0	X	X	X	X	X	X	X

Since RA register is R3, it will be register 1[6:4] in ISA and its value in binary is 011.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	0	0	0	0	0	1	1	X	X	X	X

Since address/displacement is 2, it will be Address[3:0] in ISA and its value in binary is 0010.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	0	0	0	0	0	1	1	0	0	1	0

Example: ISA of Memory & IO Instructions

12	11	10	09	08	07	06	05	04	03	02	01	00
01XXXX (Opcode)						Register 1(RA)			Register 2(RB)			Unused

Question:

2. STORE [R2], R1 (Format: STORE [RB], RA)

Answer:

Since type of instruction is Memory & IO operations, so opcode of first two bits[12:11] is 11.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	X	X	X	X	X	X	X	X	X	X	X

Since opcode of this STORE (Register Indirect mode) operation is 0100, so rest of opcode[10:7] is 0100.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	0	1	0	0	X	X	X	X	X	X	X

Since RA register is R1, it will be register 1[6:4] in ISA and its value in binary is 001.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	0	1	0	0	0	0	1	X	X	X	X

Since RB register is R2, it will be register 2[3:1] in ISA and its value in binary is 010.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	0	1	0	0	0	0	1	0	1	0	X

Example: ISA of Memory & IO Instructions

12	11	10	09	08	07	06	05	04	03	02	01	00
01XXXX (Opcode)						Register 1(RA)			Register 2(RB)		Displacement (Disp)	

Question:
3. STORE [R2+2], R1 (Format: STORE [RB+Disp], RA)
Answer:
Since type of instruction is Memory & IO operations, so opcode of first two bits[12:11] is 11.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	X	X	X	X	X	X	X	X	X	X	X

Since opcode of this STORE (Based/Indexed with Displacement Mode) operation is 0100, so rest of opcode[10:7] is 0101.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	0	1	0	1	X	X	X	X	X	X	X

Since RA register is R1, it will be register 1[6:4] in ISA and its value in binary is 001.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	0	1	0	1	0	0	1	X	X	X	X

Since RB register is R2, it will be register 2[3:2] in ISA and its value in binary is 10.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	0	1	0	1	0	0	1	1	0	X	X

Since displacement is 2, it will be Displacement[1:0] in ISA and its value in binary is 10.

12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	0	1	0	1	0	0	1	1	0	1	0

Example: ISA Design

Design an ISA of 16-bit CPU based on following requirements:

i. **Types of Instructions:** ALU Instruction (Immediate) (XOR RA, 2), Jump Instruction (JMP LABEL), Memory Instructions (LOAD Direct) (LOAD RA, [Address]).

ii. **No of Registers:** 32

iii. **Supported RAM:** 128x32

Answer:

Word size of RAM is 32. So, maximum size of instruction is 32.

No of types of Instructions is 3 and Assuming no of ALU Instructions (Immediate) and Jump Instructions is at least 16. So, Opcode is 6 bit where 2 bits is for types of instruction ($2^2 = 4$) and 4 bits for operations ($2^4 = 16$).

ISA format of ALU Instruction (Immediate) is:

Opcode (6 bit)		Register 1	Value	Unused
2 bits	4 bits	5 bits	16 bits	5 bits
(00) Types of instruction	Operations	Ra(00000-11111)	Value (0000000000000000-1111111111111111)	XXXXX

Since no of Registers is 32, 5-bits will be needed to address 32 registers ($2^5 = 32$). Since CPU is 16-bit, size of value will be 16-bit. Size of ISA is 6+5+16 = 27 bits. Assuming opcode of AND RA, 2 is 00000.

Example: ISA Design

ISA format of Jump Instruction is:

Opcode (6 bit)		Address	Unused
2 bits	4 bits	7 bits	19 bits
(01) Types of instruction	Operations	Value (0000000-1111111)	XXXXXXXXXXXXXXXXXXX

Since size of RAM is 128, it will take 7-bits to address all memory locations ($2^7 = 128$). Size of ISA is 6+7 = 13 bits. Assuming opcode of JMP LABEL is 010000.

ISA format of Memory Instruction (LOAD RA, [Address]) is:

Opcode (6 bit)		Register 1	Address	Unused
2 bits	4 bits	5 bits	7 bits	14 bits
(10) Types of instruction	Operations	RA (00000-11111)	Value (0000000-1111111)	XXXXXXXXXXXXXX

Size of ISA is 6+5+7 = 18 bits. Assuming opcode of LOAD RA, [Address] is 100000.

So, total size of ISA must be 27 bits. Extra bits in other instructions will be unused.

Example: ISA Design

ISA format of ALU Instruction (Immediate) is:

Opcode (6 bit)		Register 1	Value
2 bits	4 bits	5 bits	16 bits
(00) Types of instruction	Operations	Ra(00000-11111)	Value (0000000000000000-1111111111111111)

ISA format of Jump Instruction is:

Opcode (6 bit)		Address	Unused
2 bits	4 bits	7 bits	14 bits
(01) Types of instruction	Operations	Value (0000000-1111111)	X

ISA format of Memory Instruction (LOAD RA, [Address]) is:

Opcode (6 bit)		Register 1	Address	Unused
2 bits	4 bits	5 bits	7 bits	9 bits
(10) Types of instruction	Operations	RA (00000-11111)	Value (0000000-1111111)	X

Exercises

1. Translate following assembly code to machine code using ISA:
   i. XOR R2, R3
   ii. ROL R4, 3
   iii. JMP 3 (LABEL)
   iv. JGRE 10 (LABEL)
   v. LOAD R0, [2]
   vi. ACCEPT_INPUT
   vii. All possible instructions of ISA.
2. Design an ISA of 16-bit CPU based on following requirements:
   i. Types of Instructions: ALU (Immediate) and Memory Instructions (LOAD)
   ii. No of Registers: 32
   iii. Supported RAM: 128x16

Thank you ☺