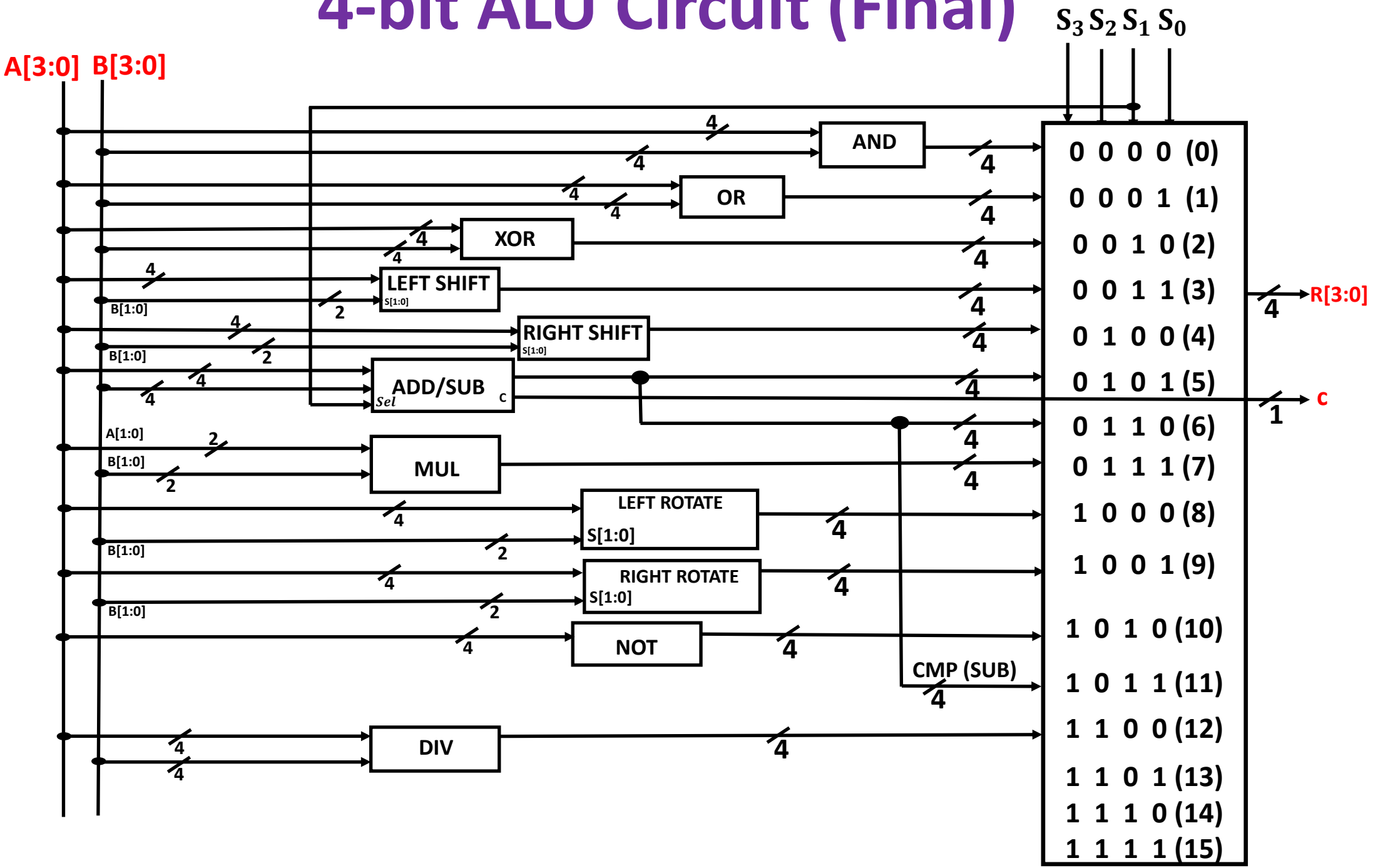
ISA Design
&
CPU Design I

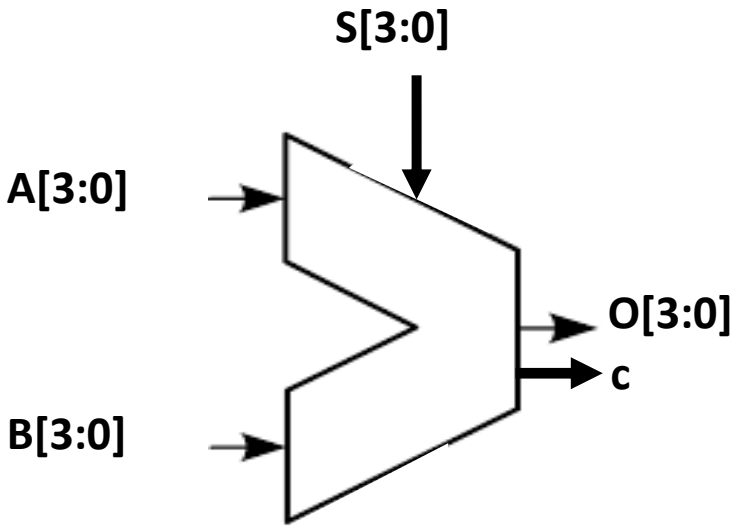
Nahin Ul Sadad
Lecturer
CSE, RUET

ALU and Register Set Review

4-bit ALU Circuit (Final)



4-bit ALU Circuit

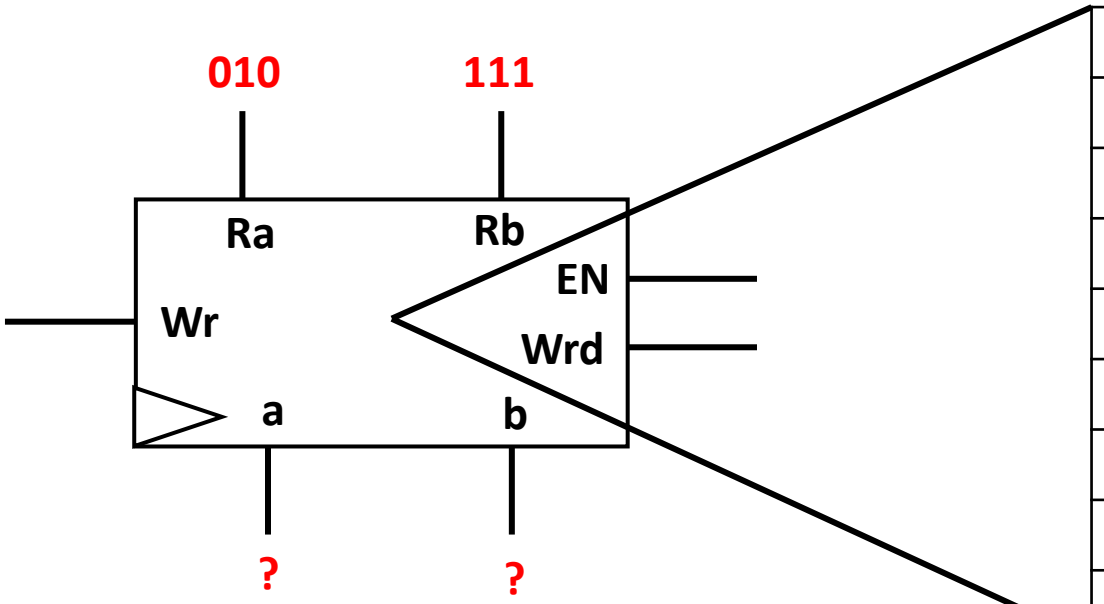


Operation	Selection lines			
	S_3	S_2	S_1	S_0
AND	0	0	0	0
OR	0	0	0	1
XOR	0	0	1	0
NOT	1	0	1	0
ADD	0	1	0	1
SUB	0	1	1	0
MUL	0	1	1	1
DIV	1	1	0	0
SHL	0	0	1	1
SHR	0	1	0	0
ROL	1	0	0	0
ROR	1	0	0	1
CMP	1	0	1	1

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.

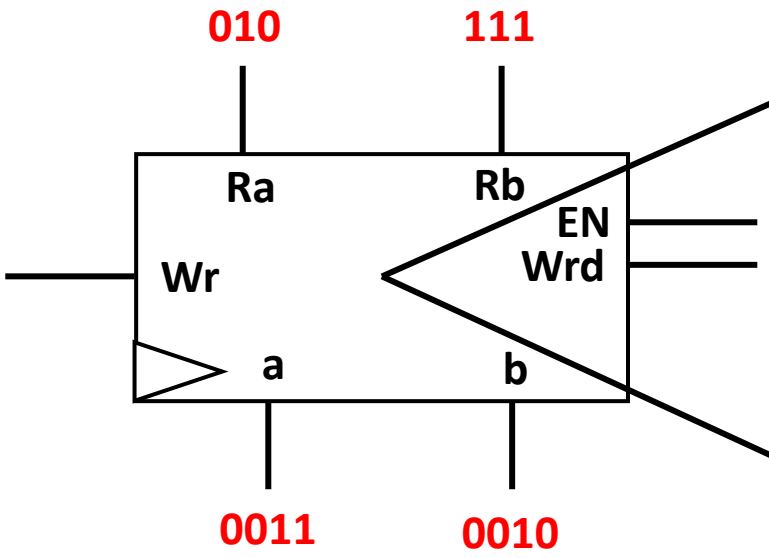
Since Register Set has 8 registers, it will need $log_2 8 = 3$ address lines.

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)

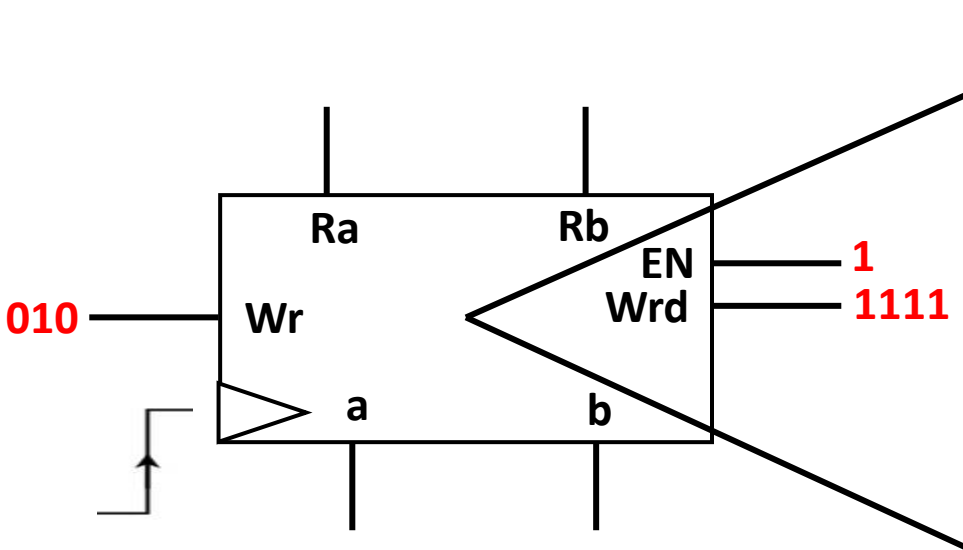
R2 register has 0011 (3) value stored.

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)



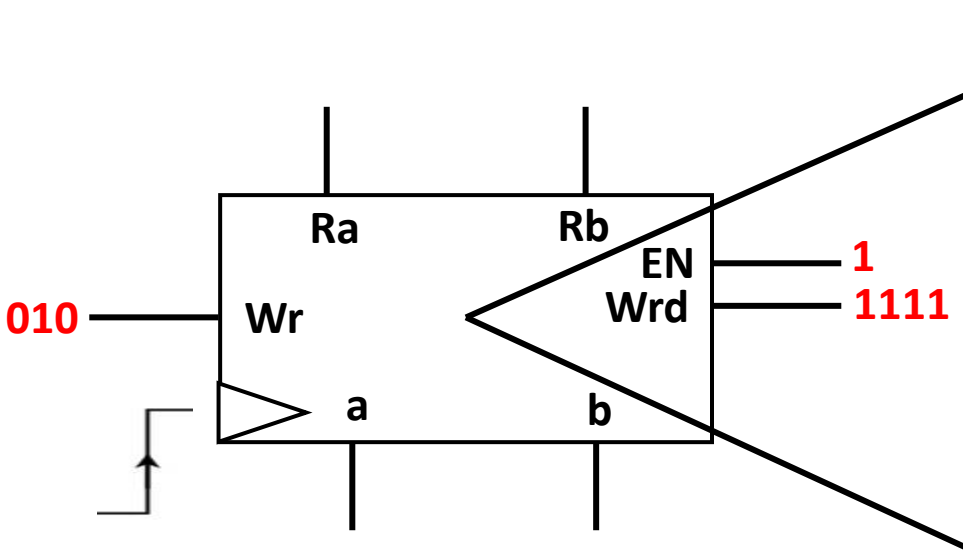
Address	Register	Value stored
000	R0	0111 (7)
001	R1	0100 (4)
010	R2	0011 (3)
011	R3	0111 (7)
100	R4	1000 (8)
101	R5	0001 (1)
110	R6	1000 (8)
111	R7	0010 (2)

What will happen?

Writing in Register Set

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.

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?

Writing in Register Set

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.

ISA Design

Instruction Set Architecture (ISA)

ISA is the contract between software and hardware.

ISA defines machine language of the CPU.

Format of ISA is usually like:

Instruction Set Architecture (ISA)

ISA is the contract between software and hardware.

ISA defines machine language of the CPU.

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 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**) → $\log_2 4 = 2$
2. No. of ALU operations supported (**12**) → $\log_2 12 \approx 4$

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)

→

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**) → $\log_2 8 = 3$
2. Word size of CPU (**4**) → 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

→

6 bits
Operands

For Arithmetic & Logic instructions (Immediate Mode),

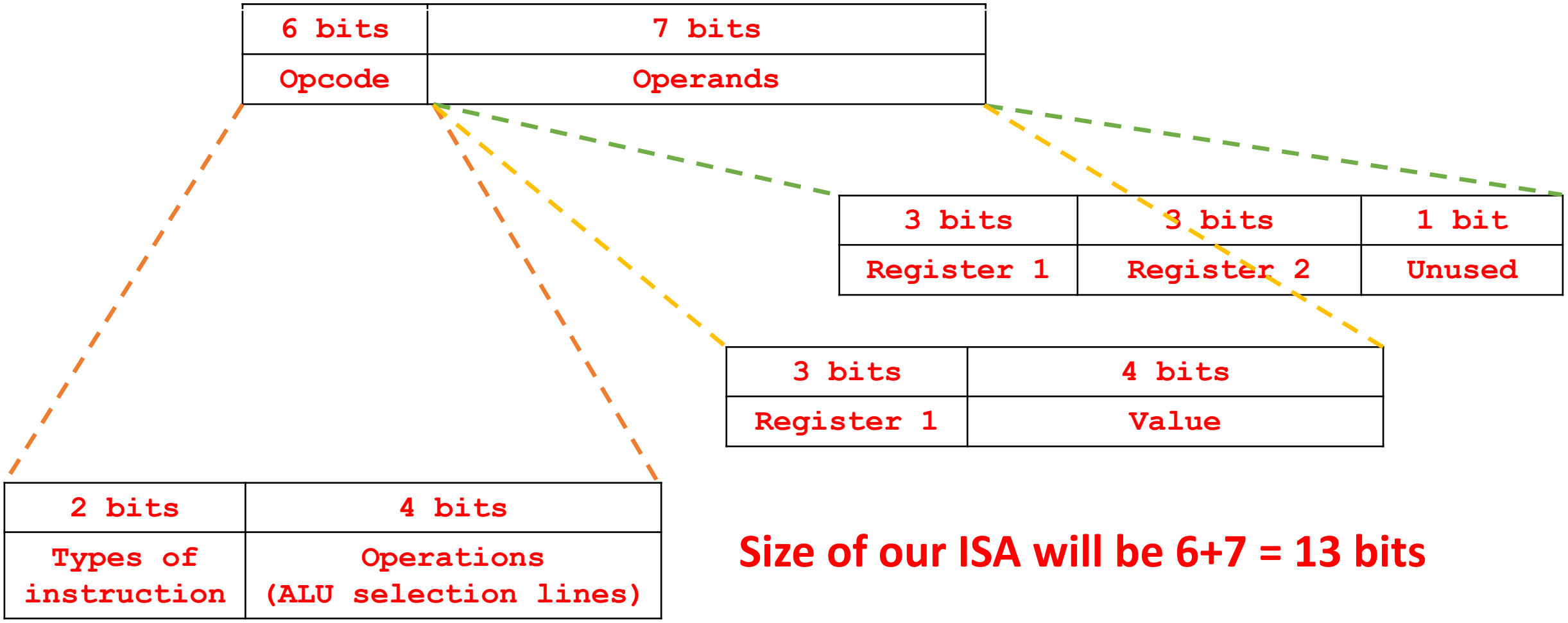
3 bits	4 bits
Register 1	Value

→

7 bits
Operands

ISA Design (Operands)

Format of ISA is usually like:



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 Operations	11	LOAD R0, [R1] STORE R1, 10

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)	000-111 (R0-R7)	000-111 (R0-R7)	AND R0, R1
	0001 (OR)	000-111 (R0-R7)	000-111 (R0-R7)	OR R1, R2
	0010 (XOR)	000-111 (R0-R7)	000-111 (R0-R7)	XOR R2, R3
	0011 (SHL)	000-111 (R0-R7)	000-111 (R0-R7)	SHL R3, R1
	0100 (SHR)	000-111 (R0-R7)	000-111 (R0-R7)	SHR R4, R2
	0101 (ADD)	000-111 (R0-R7)	000-111 (R0-R7)	ADD R5, R4
	0110 (SUB)	000-111 (R0-R7)	000-111 (R0-R7)	SUB R6, R5
	0111 (MUL)	000-111 (R0-R7)	000-111 (R0-R7)	MUL R7, R6
	1000 (ROL)	000-111 (R0-R7)	000-111 (R0-R7)	ROL R0, R5
	1001 (ROR)	000-111 (R0-R7)	000-111 (R0-R7)	ROR R1, R2
	1010 (NOT)	000-111 (R0-R7)	000-111 (R0-R7)	NOT R4
	1011 (CMP)	000-111 (R0-R7)	000-111 (R0-R7)	CMP R5, R7
	1100 (DIV)	000-111 (R0-R7)	000-111 (R0-R7)	DIV R5, R7

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

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.

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, R5

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 R5, it will be register 2[3:1] in ISA and its value in binary is 101.

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

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

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.

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)

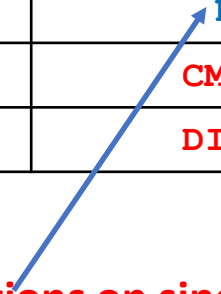
Opcode (6 bit)		Register 1	Constant
2 bits	4 bits	3 bits	4 bits
Types of instruction	Operations (ALU selection lines)	Ra (000-111)	Value (0000-1111)

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)	000-111 (R0-R7)	0000-1111 (0-15)	ANDI R0, 1
	0001 (OR)	000-111 (R0-R7)	0000-1111 (0-15)	ORI R1, 2
	0010 (XOR)	000-111 (R0-R7)	0000-1111 (0-15)	XORI R2, 3
	0011 (SHL)	000-111 (R0-R7)	0000-1111 (0-15)	SHLI R3, 3 (MAX 3)
	0100 (SHR)	000-111 (R0-R7)	0000-1111 (0-15)	SHRI R4, 2(MAX 3)
	0101 (ADD)	000-111 (R0-R7)	0000-1111 (0-15)	ADDI R5, 4
	0110 (SUB)	000-111 (R0-R7)	0000-1111 (0-15)	SUBI R6, 5
	0111 (MUL)	000-111 (R0-R7)	0000-1111 (0-15)	MULI R7, 3 (MAX 3)
	1000 (ROL)	000-111 (R0-R7)	0000-1111 (0-15)	ROLI R0, 1 (MAX 3)
	1001 (ROR)	000-111 (R0-R7)	0000-1111 (0-15)	RORI R1, 2 (MAX 3)
	1010 (NOT)	000-111 (R0-R7)	0000-1111 (0-15)	NOTI R4
	1011 (CMP)	000-111 (R0-R7)	0000-1111 (0-15)	CMPI R5, 7
	1100 (DIV)	000-111 (R0-R7)	0000-1111 (0-15)	DIVI R5, 7



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])

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

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.

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 R7, 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 R7, it will be register 1[6:4] in ISA and its value in binary is 111.

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

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

12 7	6 0
10xxxx (Opcode)	Address

ISA of Branching

Opcode		Address	Assembly Example
Type (2 bits)	Operations (4 bits)	7 bits	
10	0000 (JMP)	0000000-1111111 (0-127)	JMP LABEL
	0001 (JE)	0000000-1111111 (0-127)	JE LABEL
	0010 (JNE)	0000000-1111111 (0-127)	JNE LABEL
	0011 (JL)	0000000-1111111 (0-127)	JL LABEL
	0100 (JLE)	0000000-1111111 (0-127)	JLE LABEL
	0101 (JG)	0000000-1111111 (0-127)	JG LABEL
	0110 (JGE)	0000000-1111111 (0-127)	JGE LABEL
	0111 (JC)	0000000-1111111 (0-127)	JC LABEL
	1000 (JNC)	0000000-1111111 (0-127)	JNC LABEL
	1001 (JZ)	0000000-1111111 (0-127)	JZ LABEL
	1010 (JNZ)	0000000-1111111 (0-127)	JNZ LABEL
	1011 (XXX)	0000000-1111111 (0-127)	XXX
	1100 (XXX)	0000000-1111111 (0-127)	XXX
	1101 (XXX)	0000000-1111111 (0-127)	XXX

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

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

Main Memory (SRAM)

Question:

Suppose, you want to store 100 instructions in your main memory (RAM) and size of your ISA is 13 bits, then

1. **How many address lines will be needed in RAM?**
2. **What should be the minimum word size of RAM so that instruction can be stored in one address?**

Main Memory (SRAM)

Question:

Suppose, you want to store 100 instructions in your main memory (RAM) and size of your ISA is 13 bits, then

1. **How many address lines will be needed in RAM?**

   Answer: No. of address lines = $log_2(100) = 6.64 \approx 7$ **bits**

2. **What should be the minimum word size of RAM so that instruction can be stored in one address?**

   Answer: Since size of ISA is 13 bits, minimum word size of RAM must be 13 bits.

Main Memory (SRAM)

RAM chip is shown below:

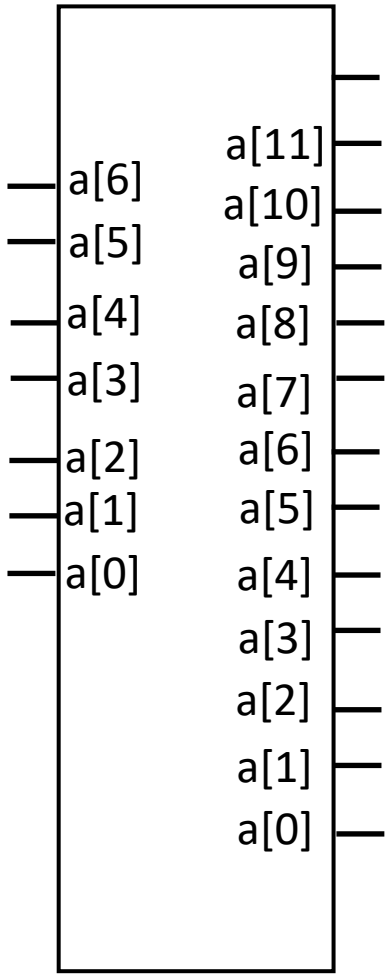


Figure: RAM chip (128x13)

Sample Code

Consider following pseudo code which compares two numbers:

Pseudo Code

`LABEL:`
`R4 = 5`
`JMP LABEL`

Now translate this code to assembly code!

Sample Code

Consider following pseudo code which compares two numbers:

Pseudo Code

```
LABEL:
R4 = 5
JMP LABEL
```

Assembly

Code

```
LABEL:
        XOR R4, R4    ;clearing R4
        ADDI R4, 5    ;R4 = 5
        JMP LABEL     ;JMP to Beginning
```

Sample Code

Consider assembly code with address:

Address	Code
00	`LABEL: XOR R4, R4`
01	`ADDI R4, 5`
02	`JMP LABEL`

Now translate this code to machine code!

Sample Code

Consider assembly code with address:

Address	Code
00	LABEL: XOR R4, R4
01	ADDI R4, 5
02	JMP LABEL

Address	Code	Opcode (2 bits)	Opcode (4 bits)	Register 1	Register 2	Constant	Address	Machine Code											
								12	11	10	9	8	7	6	5	4	3	2	1	0
0000000 (0)	LABEL: XOR R4, R4	00 (A&L Reg)	0010 (XOR)	100 (R4)	100 (R4)	X	X	0	0	0	0	1	0	1	0	0	1	0	0	1
0000001 (1)	ADDI R4, 5	01 (A&L Imm)	0101 (ADD)	100 (R4)	X	0101 (5)	X	0	1	0	1	0	1	1	0	0	0	1	0	1
0000010 (2)	JMP LABEL	10 (Branching)	0000 (JMP)	X	X	X	0000000 (0)	1	0	0	0	0	0	0	0	0	0	0	0	0

Main Memory (SRAM)

- **Suppose, following instructions of a program are stored in RAM after moving it from HDD.**

Address	Instructions												
	12	11	10	9	8	7	6	5	4	3	2	1	0
0000000	0	0	0	0	1	0	1	0	0	1	0	0	1
0000001	0	1	0	1	0	1	1	0	0	0	1	0	1
0000010	1	0	0	0	0	0	0	0	0	0	0	0	0

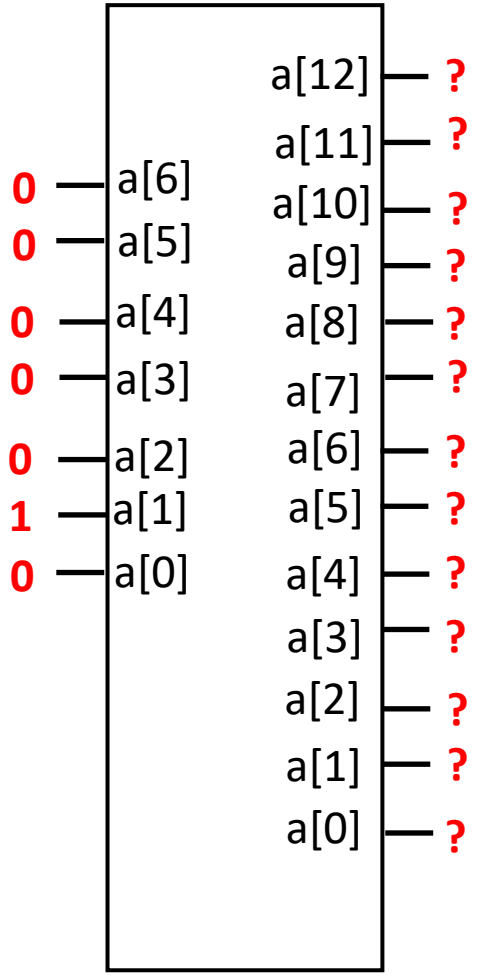


Figure: RAM chip (128x13)

Main Memory (SRAM)

- **Suppose, following instructions of a program are stored in RAM after moving it from HDD.**

Address	Instructions												
	12	11	10	9	8	7	6	5	4	3	2	1	0
0000000	0	0	0	0	1	0	1	0	0	1	0	0	1
0000001	0	1	0	1	0	1	1	0	0	0	1	0	1
0000010	1	0	0	0	0	0	0	0	0	0	0	0	0

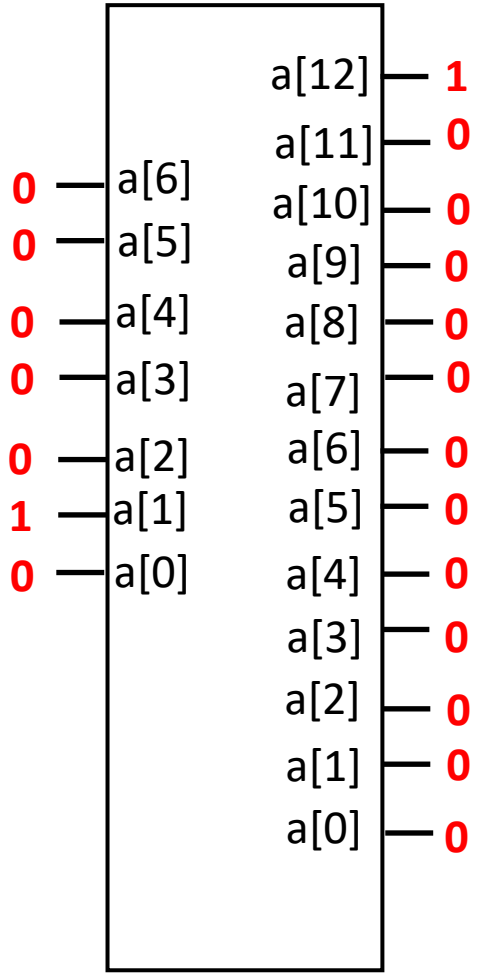


Figure: RAM chip (128x13)

Main Memory (SRAM)

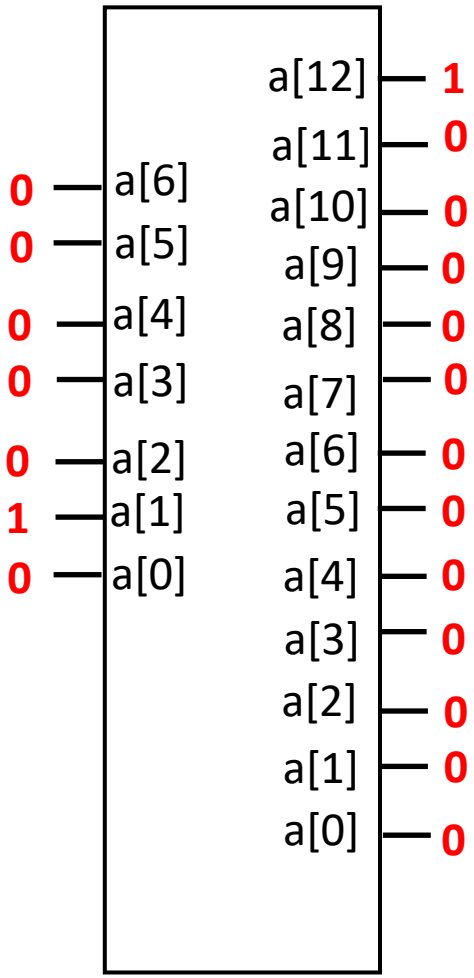


Figure: RAM chip (128x13)

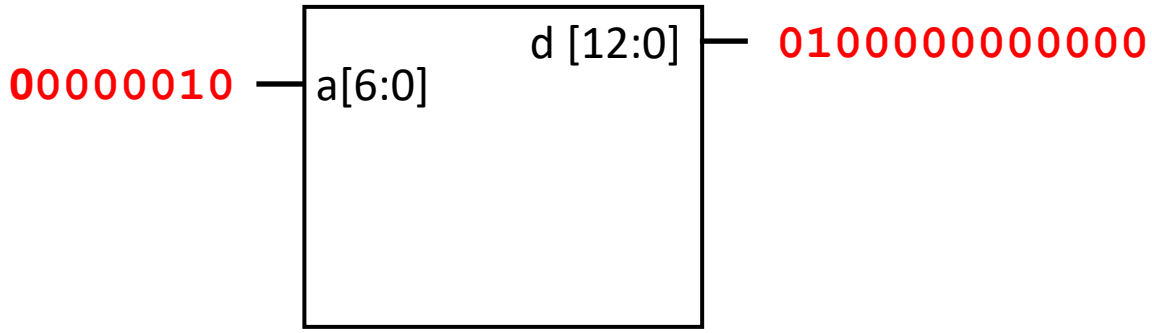
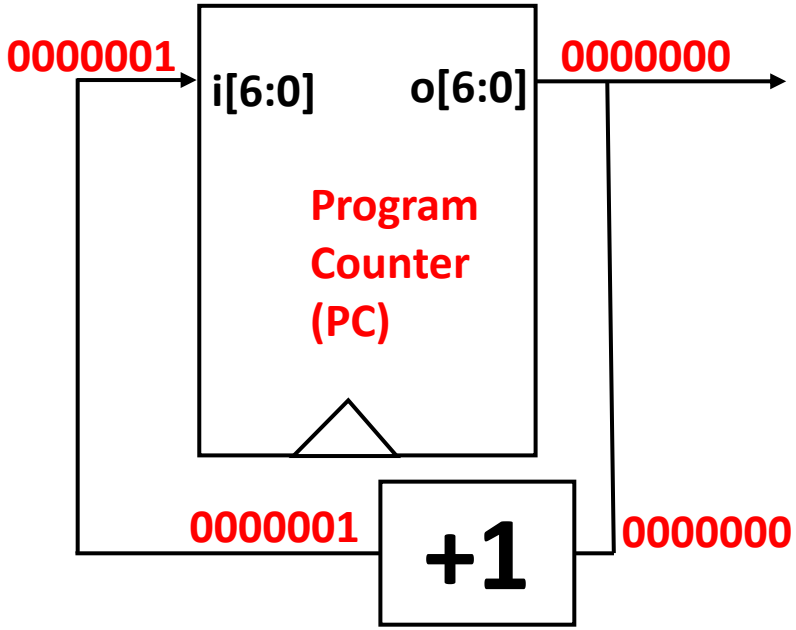


Figure: RAM chip (128x13)

CPU Design I

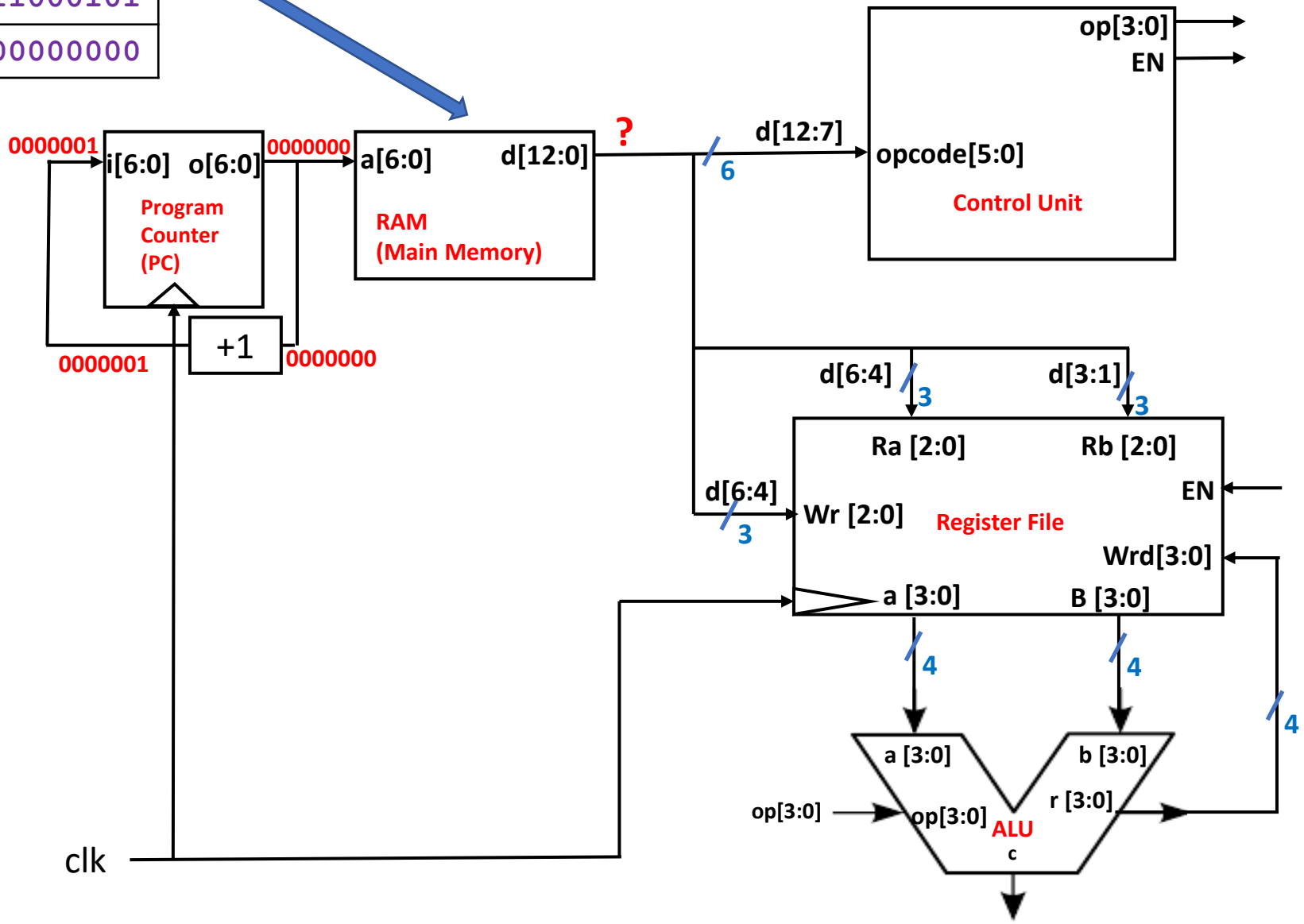
Program Counter

- Program counter will have current address.
- Its next address will also be calculated at the same time
- But it will not be updated until the next clock cycle.



4-bit CPU

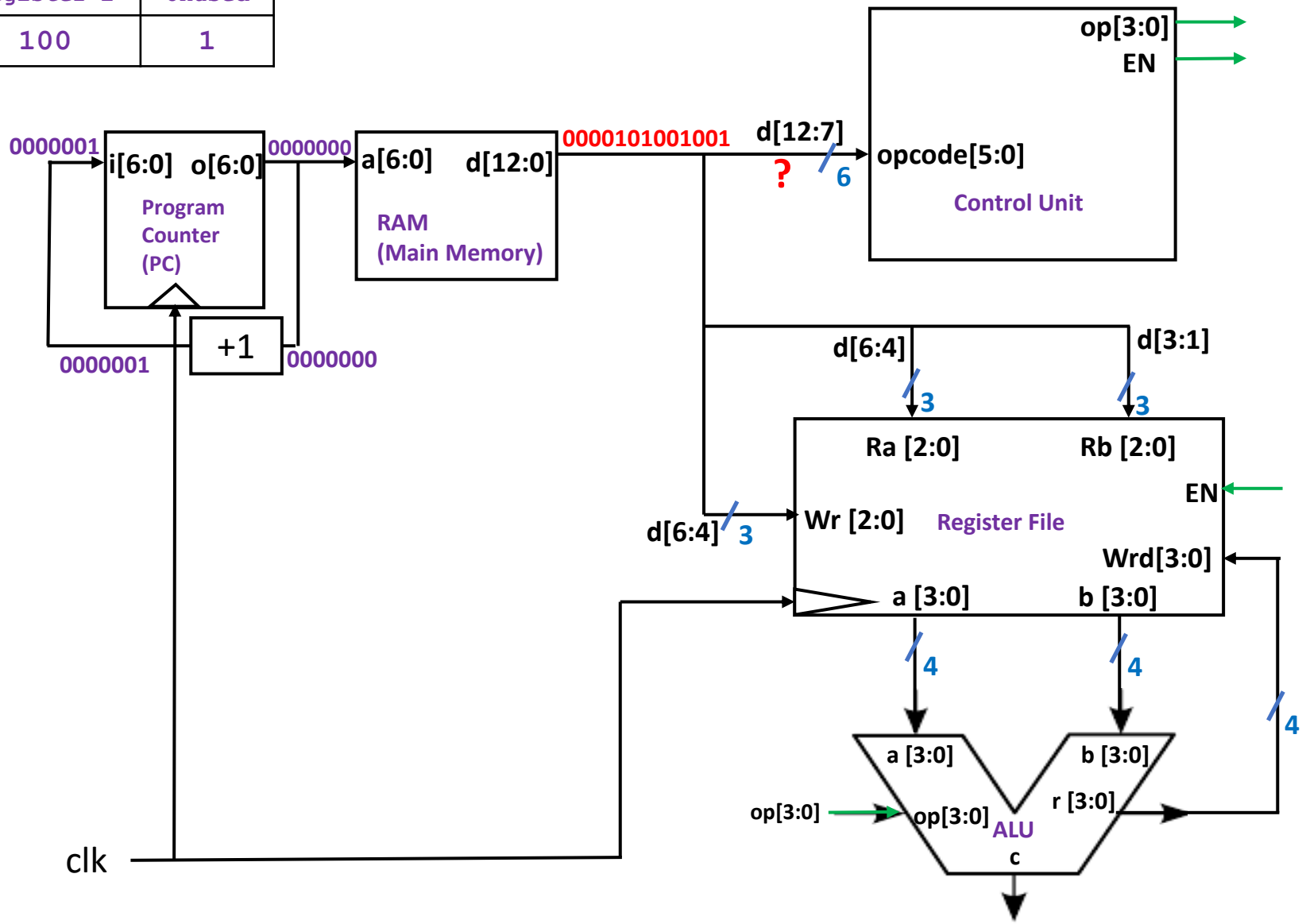
Address	Instructions
0000000	0000101001001
0000001	0101011000101
0000010	1000000000000



4-bit CPU

12 11	10 7	6 4	3 1	0
00	Operation	Register 1	Register 2	Unused
00	0010	100	100	1

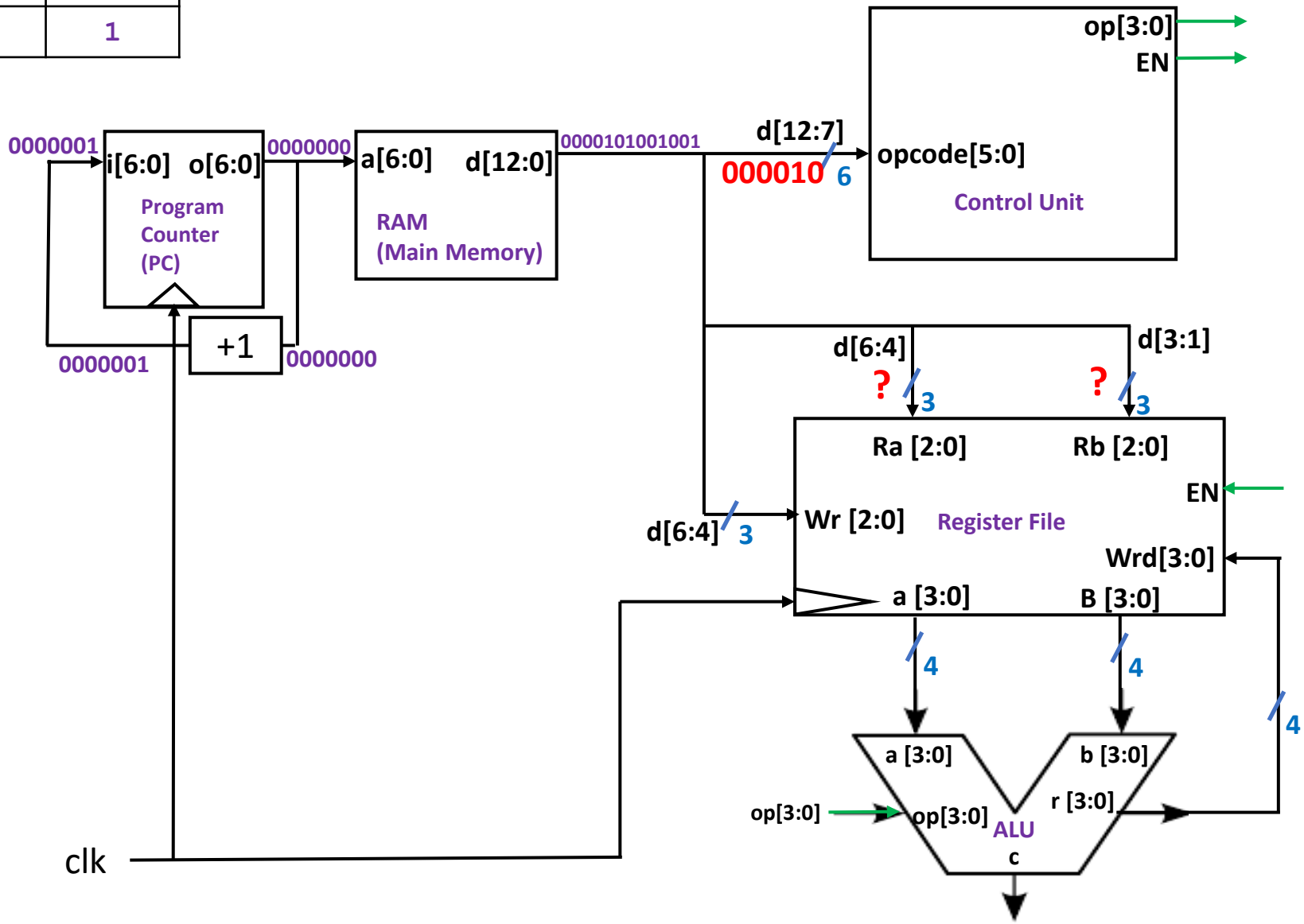
Instruction: 000010100100 1
Opcode (6bits): 000010
Opcode (2bits): 00 (A&L Reg)
Opcode (4bits): 0010 (XOR)
Register 1: 100 (R4)
Register 2: 100 (R4)
Assembly: XOR R4, R4



4-bit CPU

12 11	10 7	6 4	3 1	0
00	Operation	Register 1	Register 2	Unused
00	0010	100	100	1

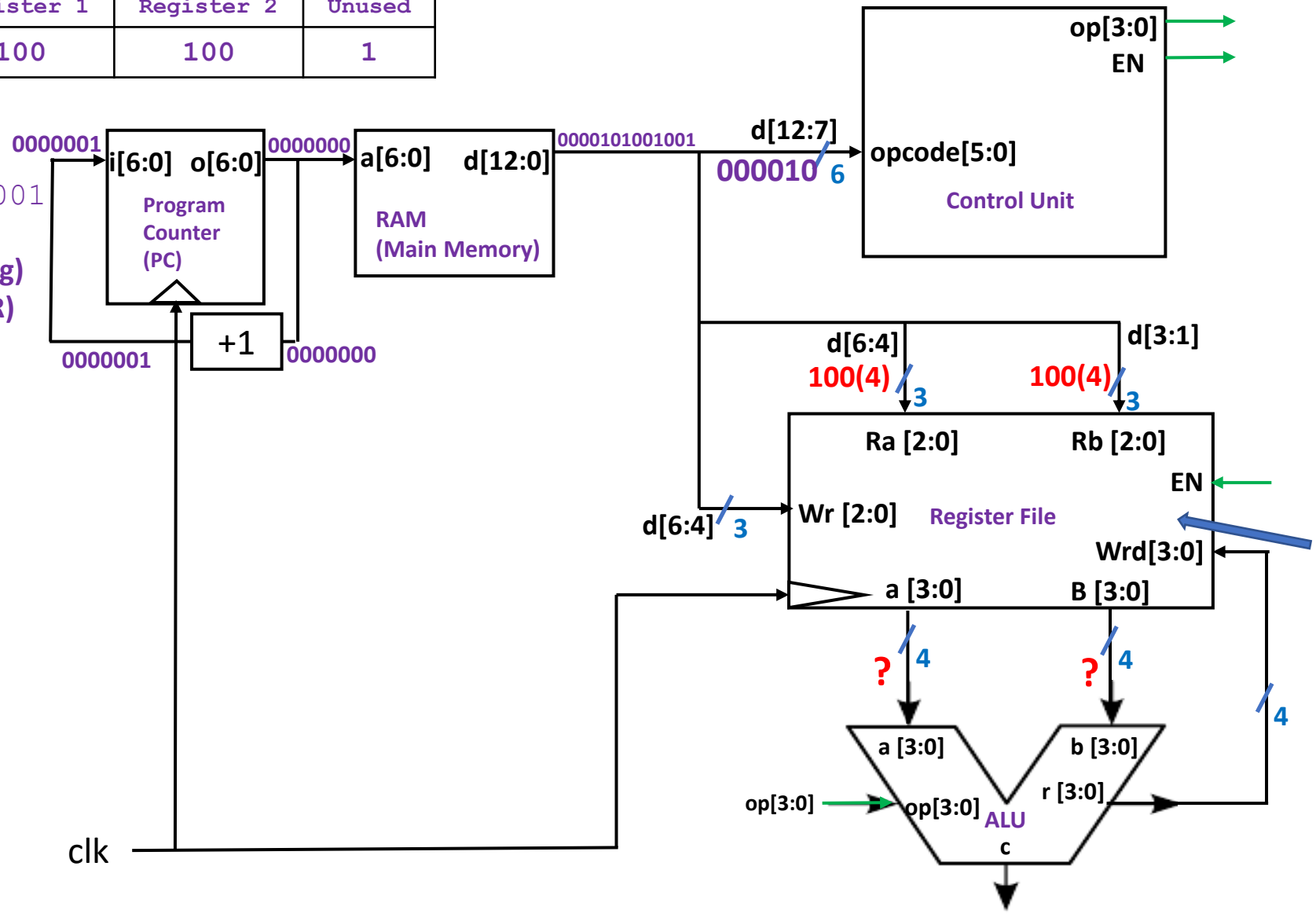
Instruction: 0000101001001
Opcode (6bits): 000010
Opcode (2bits): 00 **(A&L Reg)**
Opcode (4bits): 0010 **(XOR)**
Register 1: 100 (R4)
Register 2: 100 (R4)
Assembly: XOR R4, R4



4-bit CPU

12 11	10 7	6 4	3 1	0
00	Operation	Register 1	Register 2	Unused
00	0010	100	100	1

Instruction: 0000101001001
Opcode (6bits): 000010
Opcode (2bits): 00 **(A&L Reg)**
Opcode (4bits): 0010 **(XOR)**
Register 1: 100 **(R4)**
Register 2: 100 **(R4)**
Assembly: XOR R4, R4

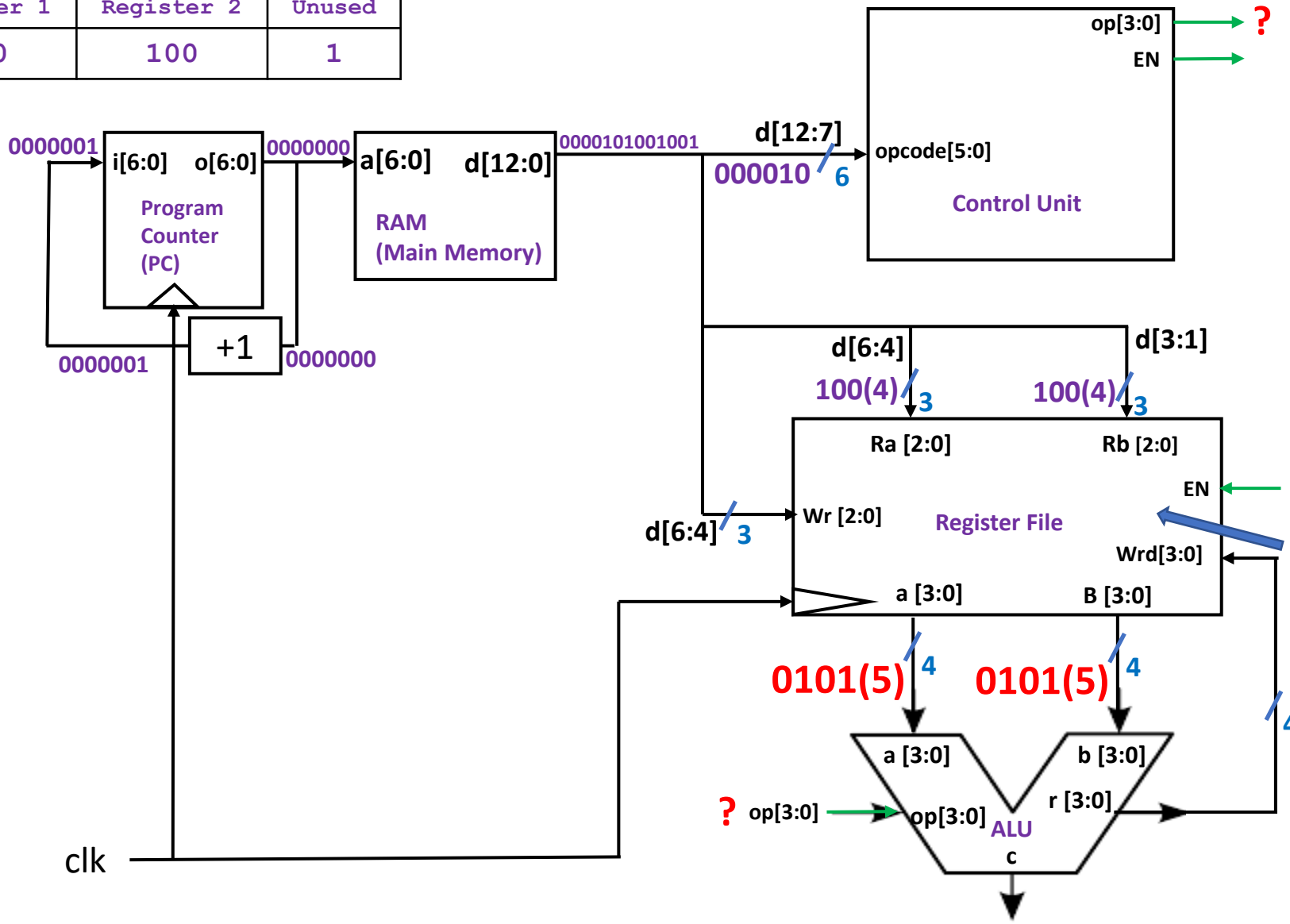


Address	Value
000 (R0)	0000
001 (R1)	0101
010 (R2)	1000
011 (R3)	0111
100 (R4)	0101
101 (R5)	0001
110 (R6)	0010
111 (R7)	0000

4-bit CPU

12 11	10 7	6 4	3 1	0
00	Operation	Register 1	Register 2	Unused
00	0010	100	100	1

Instruction: 0000101001001
Opcode (6bits): 000010
Opcode (2bits): 00 **(A&L Reg)**
Opcode (4bits): 0010 **(XOR)**
Register 1: 100 **(R4)**
Register 2: 100 **(R4)**
Assembly: XOR R4, R4

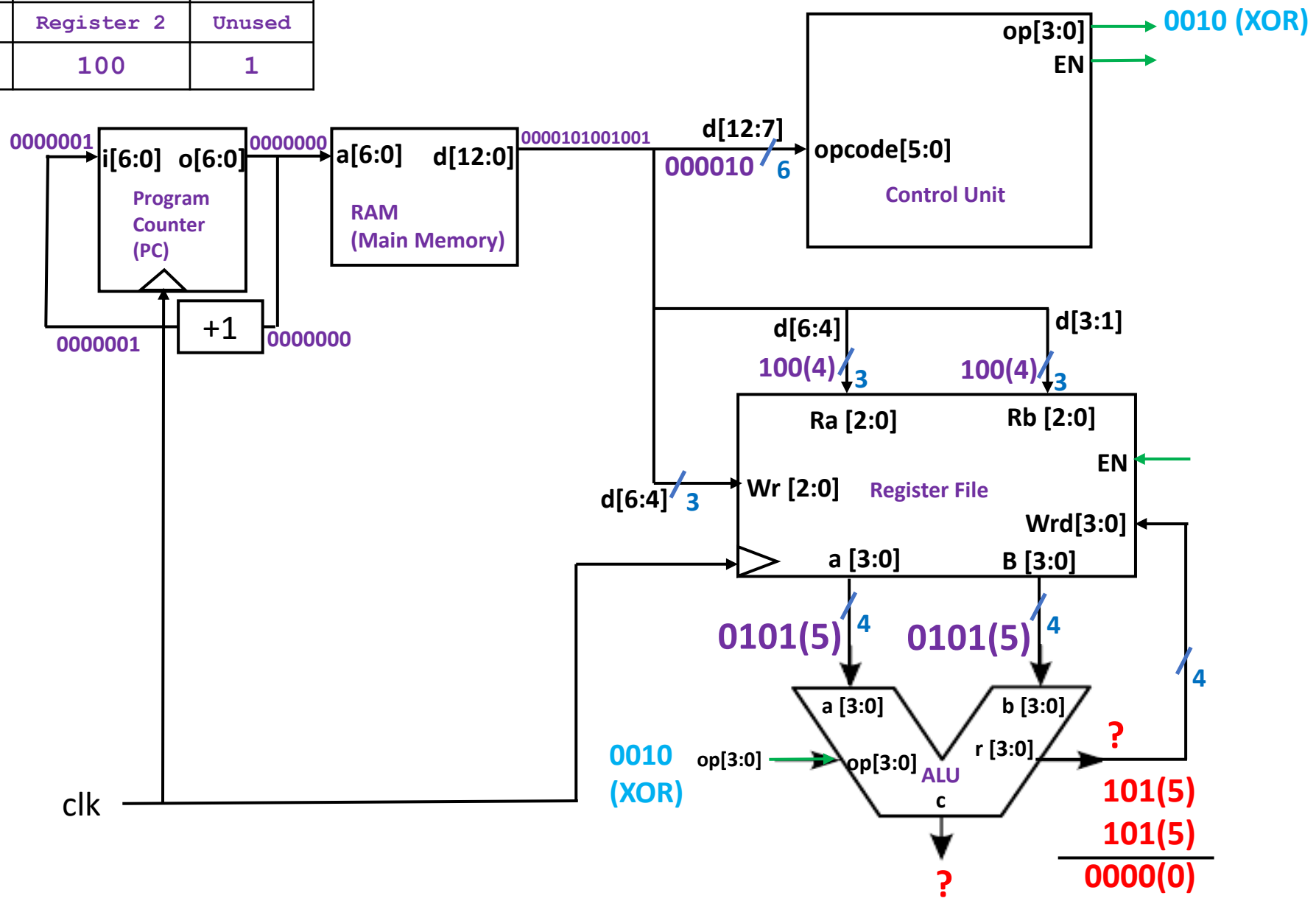


Address	Value
000 (R0)	0000
001 (R1)	0101
010 (R2)	1000
011 (R3)	0111
100 (R4)	0101
101 (R5)	0001
110 (R6)	0010
111 (R7)	0000

4-bit CPU

12 11	10 7	6 4	3 1	0
00	Operation	Register 1	Register 2	Unused
00	0010	100	100	1

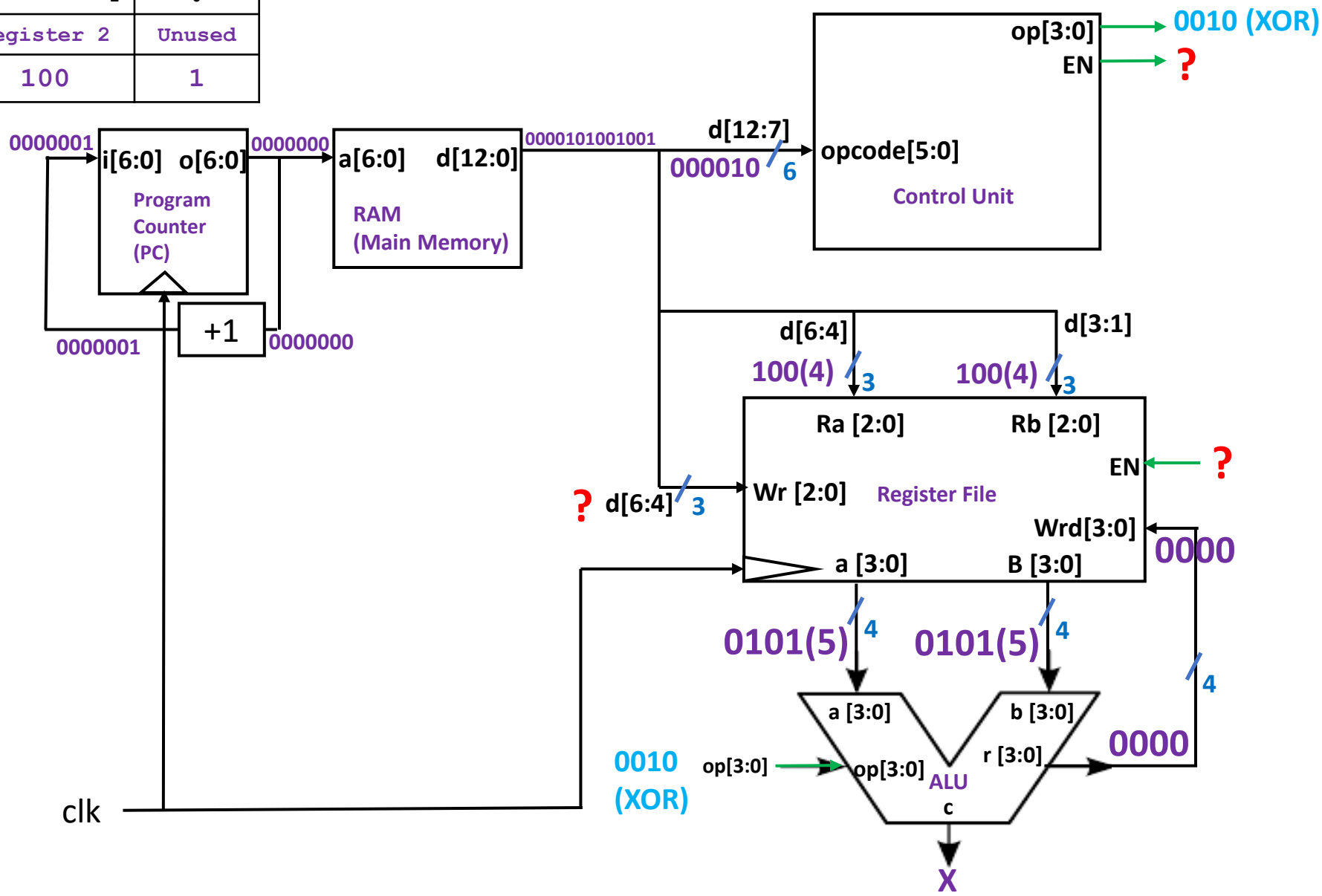
Instruction: 000010100100 1
Opcode (6bits): 000010
Opcode (2bits): 00 **(A&L Reg)**
Opcode (4bits): 0010 **(XOR)**
Register 1: 100 **(R4)**
Register 2: 100 **(R4)**
Assembly: XOR R4, R4



4-bit CPU

12 11	10 7	6 4	3 1	0
00	Operation	Register 1	Register 2	Unused
00	0010	100	100	1

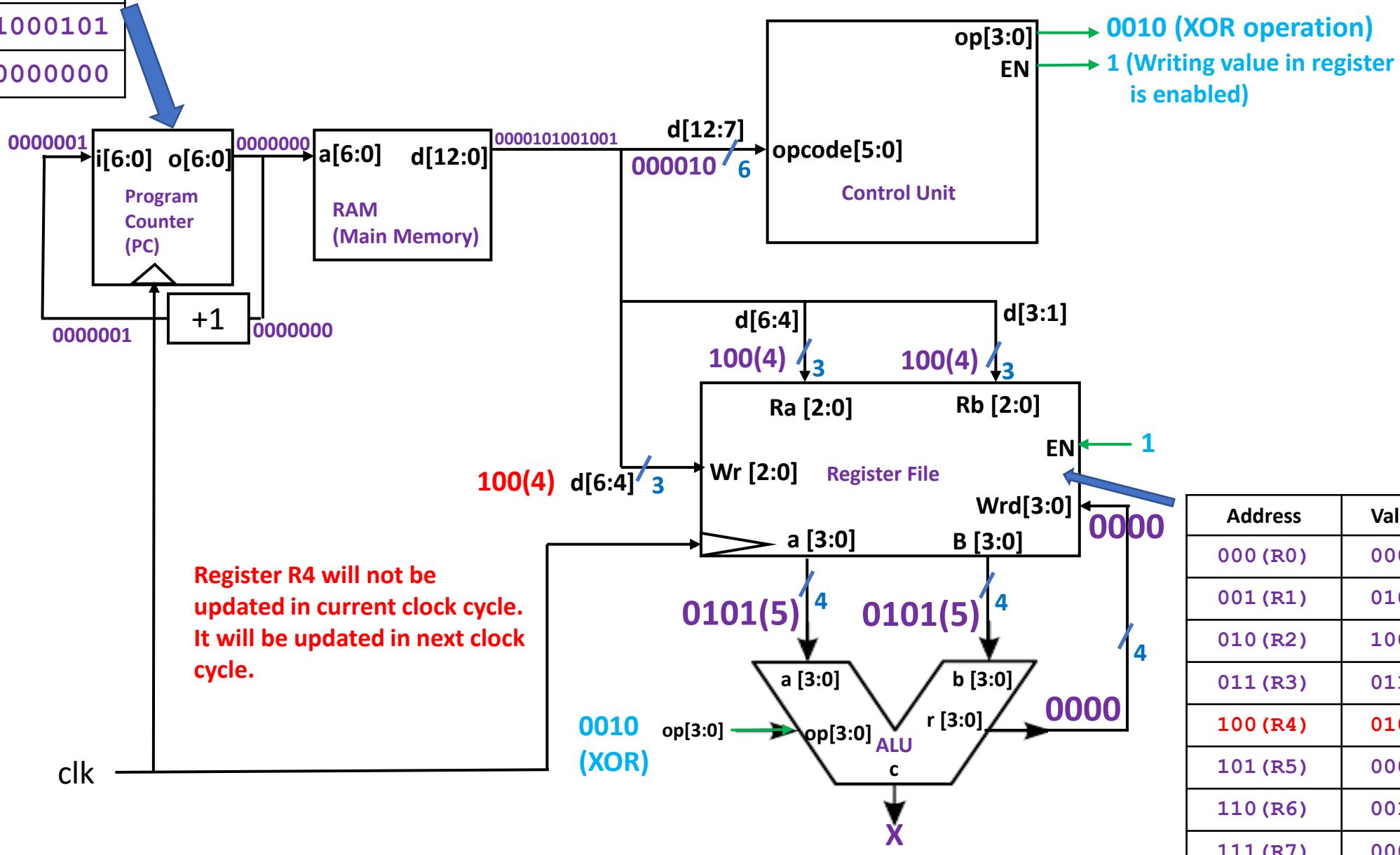
Instruction: 0000101001001
Opcode (6bits): 000010
Opcode (2bits): 00 **(A&L Reg)**
Opcode (4bits): 0010 **(XOR)**
Register 1: 100 (R4)
Register 2: 100 **(R4)**
Assembly: XOR R4, R4



(Carry will be important only for add/sub operation)

4-bit CPU

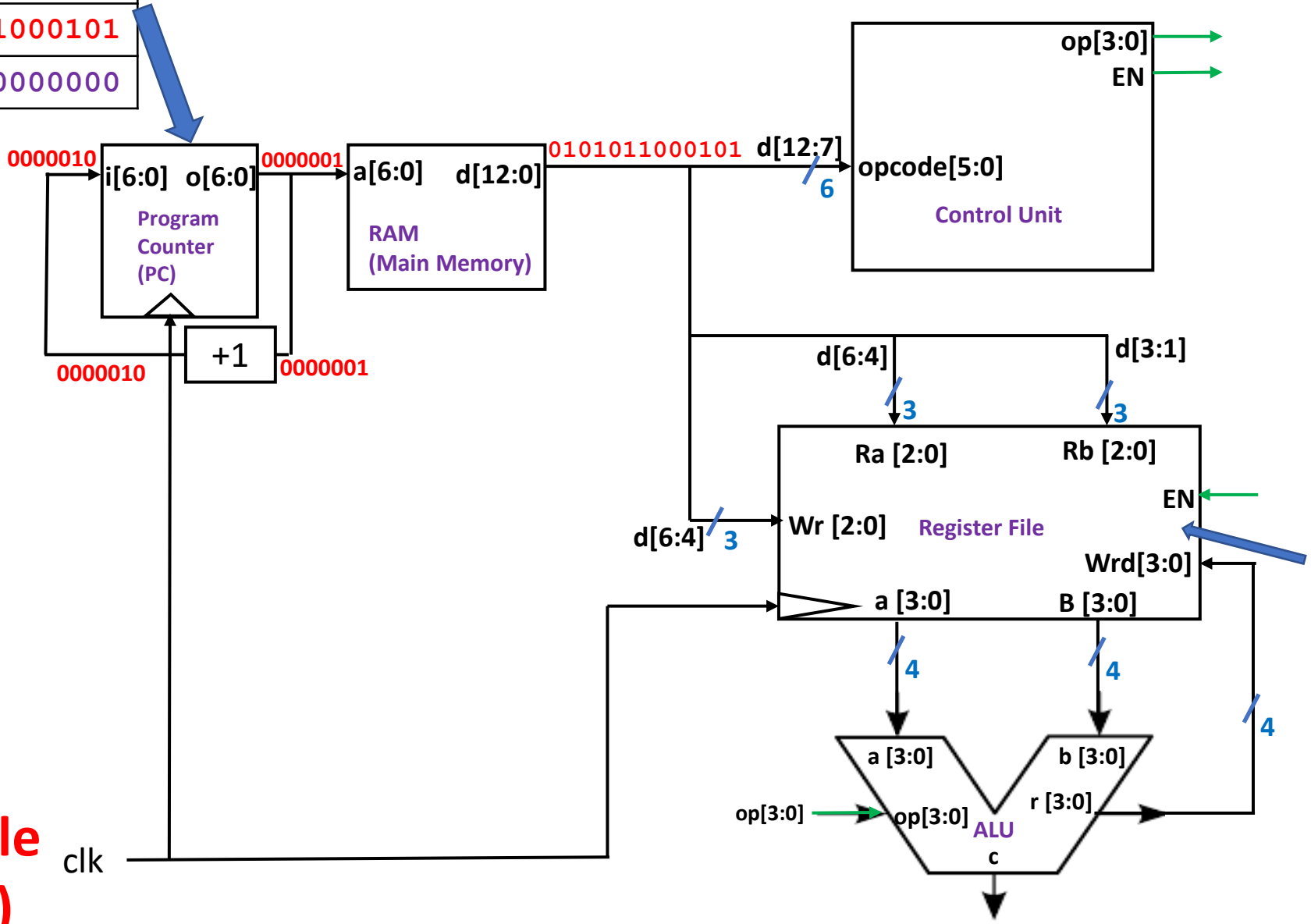
Address	Instructions
0000000	0000101001001
0000001	0101011000101
0000010	1000000000000



Address	Value
000 (R0)	0000
001 (R1)	0101
010 (R2)	1000
011 (R3)	0111
100 (R4)	0101
101 (R5)	0001
110 (R6)	0010
111 (R7)	0000

4-bit CPU

Address	Instructions
0000000	0000101001001
0000001	0101011000101
0000010	1000000000000



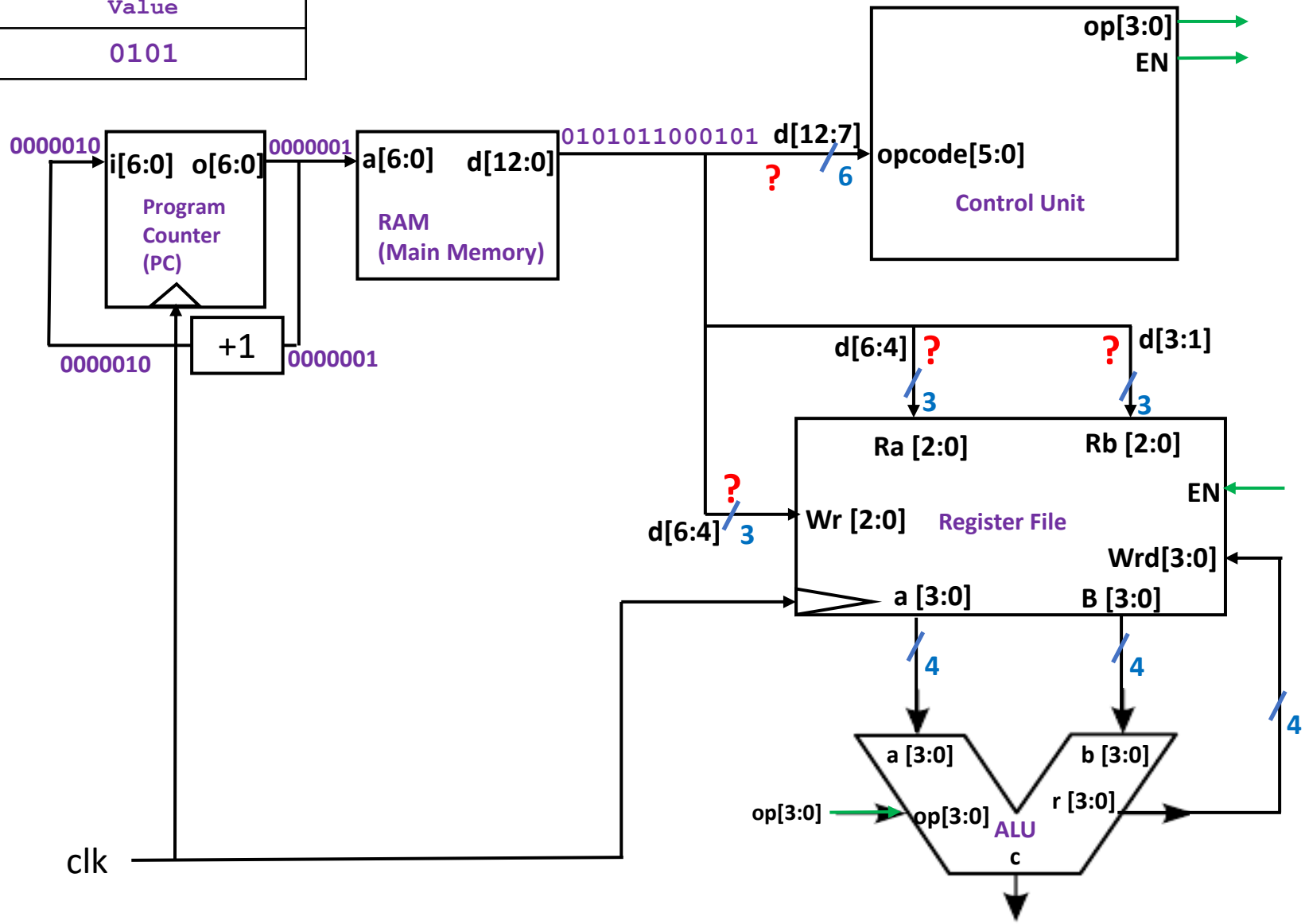
Next clock cycle (Positive Edge)

Address	Value
000 (R0)	0000
001 (R1)	0101
010 (R2)	1000
011 (R3)	0111
100 (R4)	0000
101 (R5)	0001
110 (R6)	0010
111 (R7)	0000

4-bit CPU

12 11	10 7	6 4	3 0
01	Operation	Register 1	Value
01	0101	100	0101

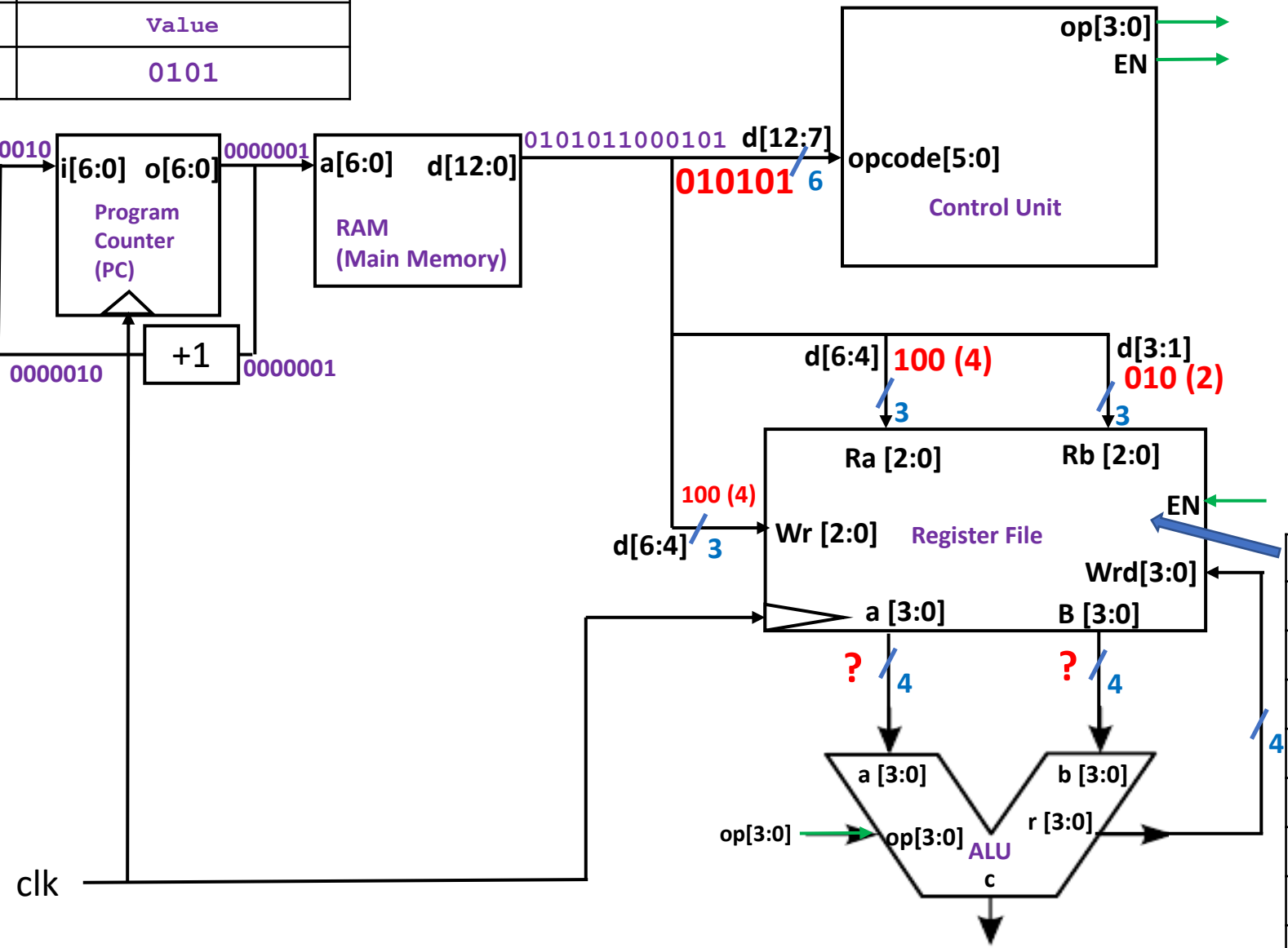
Instruction: 0101011000101
Opcode (6bits): 010101
Opcode (2bits): 01 **(A&L Imm)**
Opcode (4bits): 0101 **(ADD)**
Register 1: 100 (R4)
Value: 0101 **(5)**
Assembly: XOR R4, R4



4-bit CPU

12 11	10 7	6 4	3 0
01	Operation	Register 1	Value
01	0101	100	0101

Instruction: 0101011000101
Opcode (6bits): 010101
Opcode (2bits): 01 **(A&L Imm)**
Opcode (4bits): 0101 **(ADD)**
Register 1: 100 **(R4)**
Value: 0101 **(5)**
Assembly: ADDI R4, 5

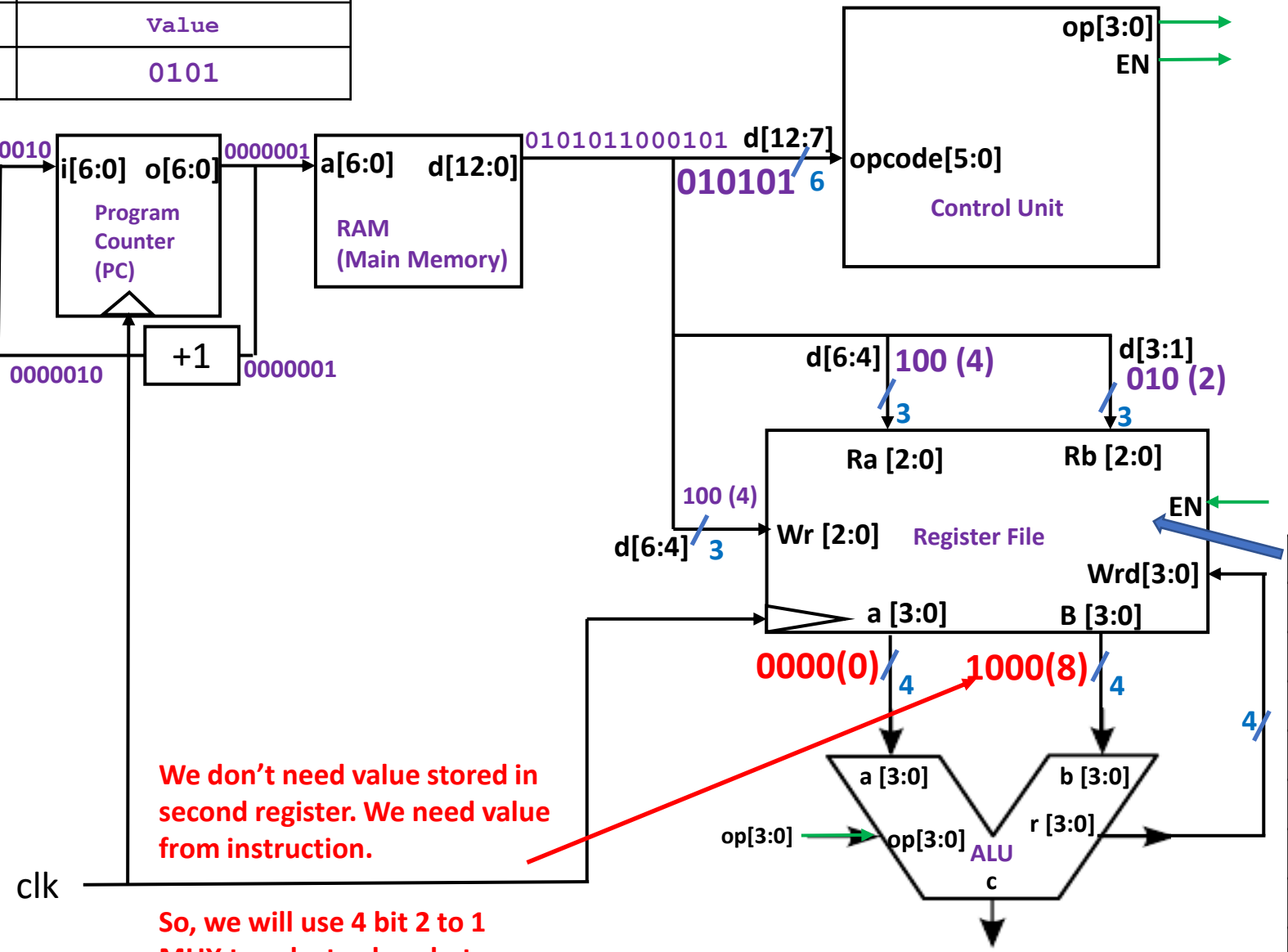


Address	Value
000 (R0)	0000
001 (R1)	0101
010 (R2)	1000
011 (R3)	0111
100 (R4)	0000
101 (R5)	0001
110 (R6)	0010
111 (R7)	0000

4-bit CPU

12 11	10 7	6 4	3 0
01	Operation	Register 1	Value
01	0101	100	0101

Instruction: 0101011000101
Opcode (6bits): 010101
Opcode (2bits): 01 **(A&L Imm)**
Opcode (4bits): 0101 **(ADD)**
Register 1: 100 **(R4)**
Value: 0101 **(5)**
Assembly: ADDI R4, 5

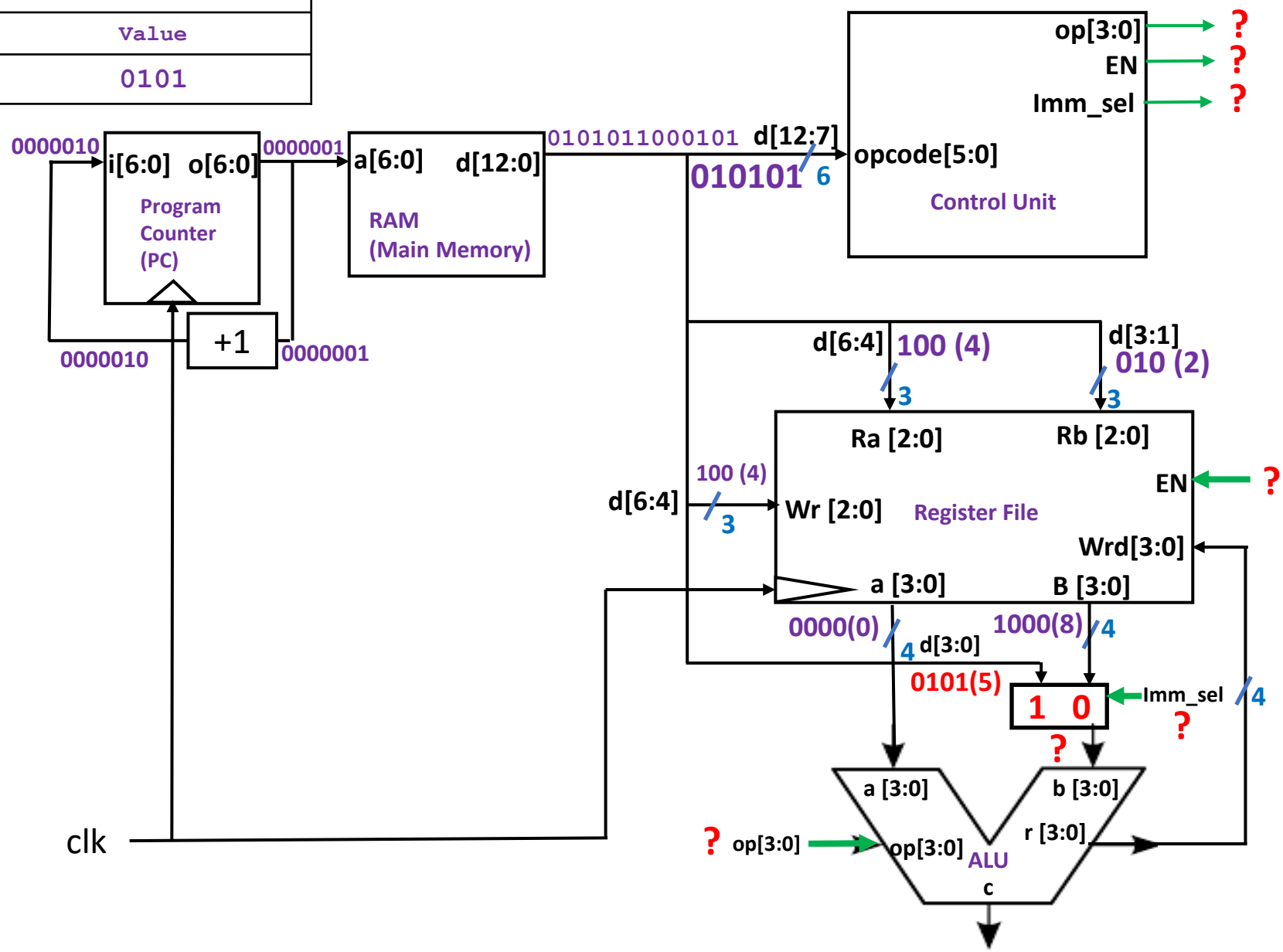


Address	Value
000 (R0)	0000
001 (R1)	0101
010 (R2)	1000
011 (R3)	0111
100 (R4)	0000
101 (R5)	0001
110 (R6)	0010
111 (R7)	0000

4-bit CPU

12 11	10 7	6 4	3 0
01	Operation	Register 1	Value
01	0101	100	0101

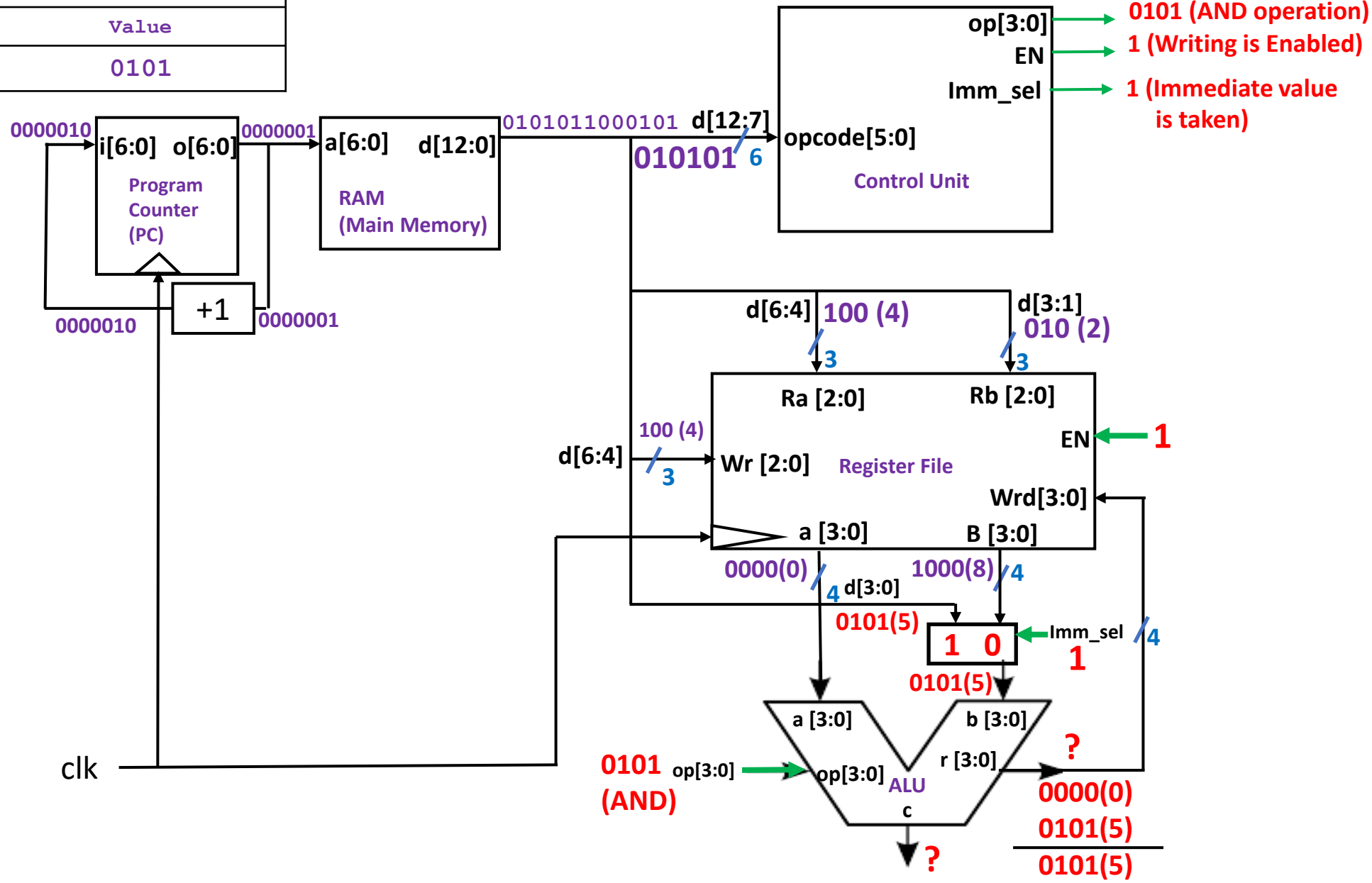
Instruction: 0101011000101
Opcode (6bits): 010101
Opcode (2bits): 01 (A&L Imm)
Opcode (4bits): 0101 (ADD)
Register 1: 100 **(R4)**
Value: 0101 **(5)**
Assembly: ADDI R4, 5



4-bit CPU

12 11	10 7	6 4	3 0
01	Operation	Register 1	Value
01	0101	100	0101

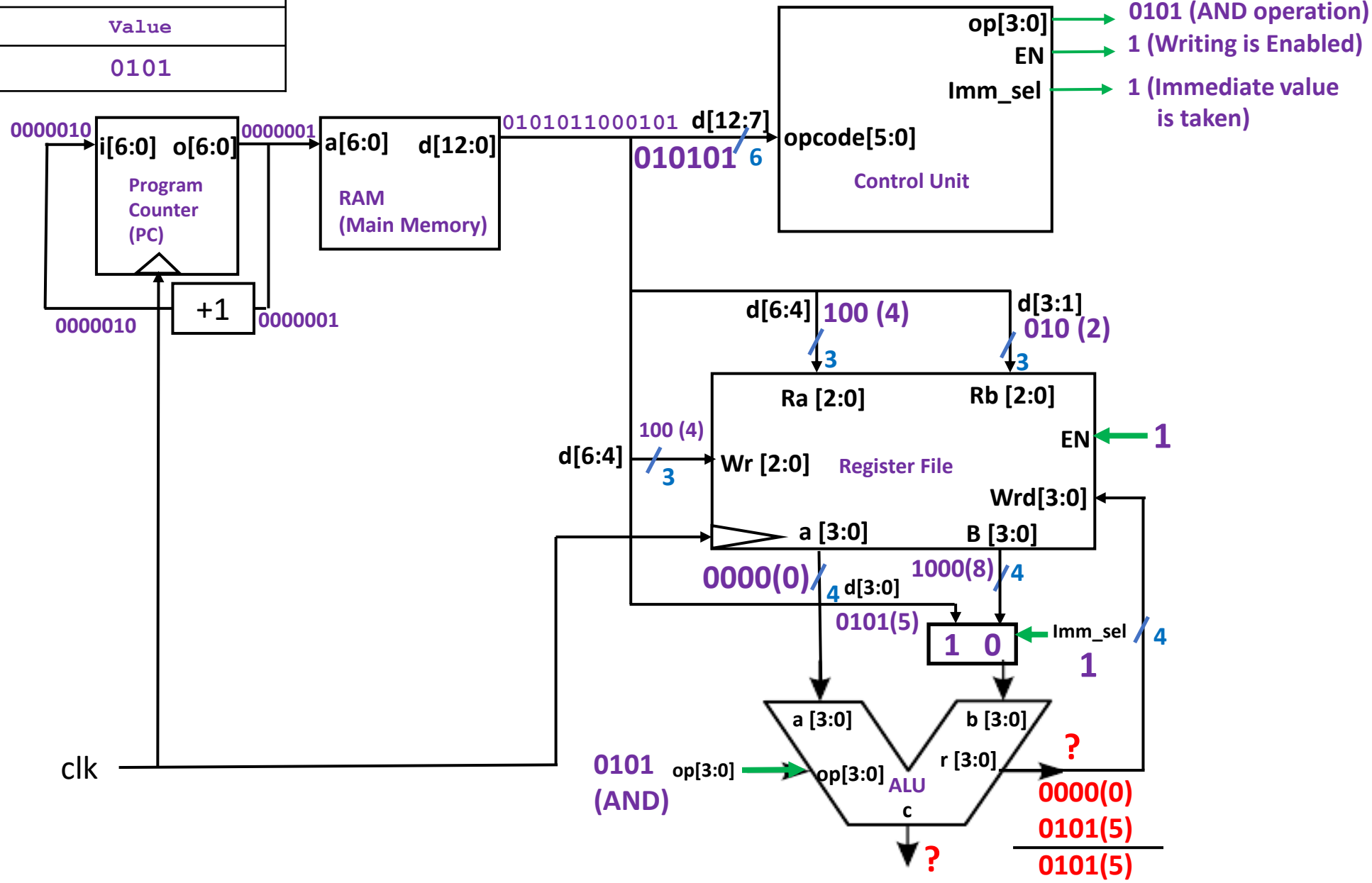
Instruction: 0101011000101
Opcode (6bits): 010101
Opcode (2bits): 01 (A&L Imm)
Opcode (4bits): 0101 (ADD)
Register 1: 100 **(R4)**
Value: 0101 **(5)**
Assembly: ADDI R4, 5



4-bit CPU

12 11	10 7	6 4	3 0
01	Operation	Register 1	Value
01	0101	100	0101

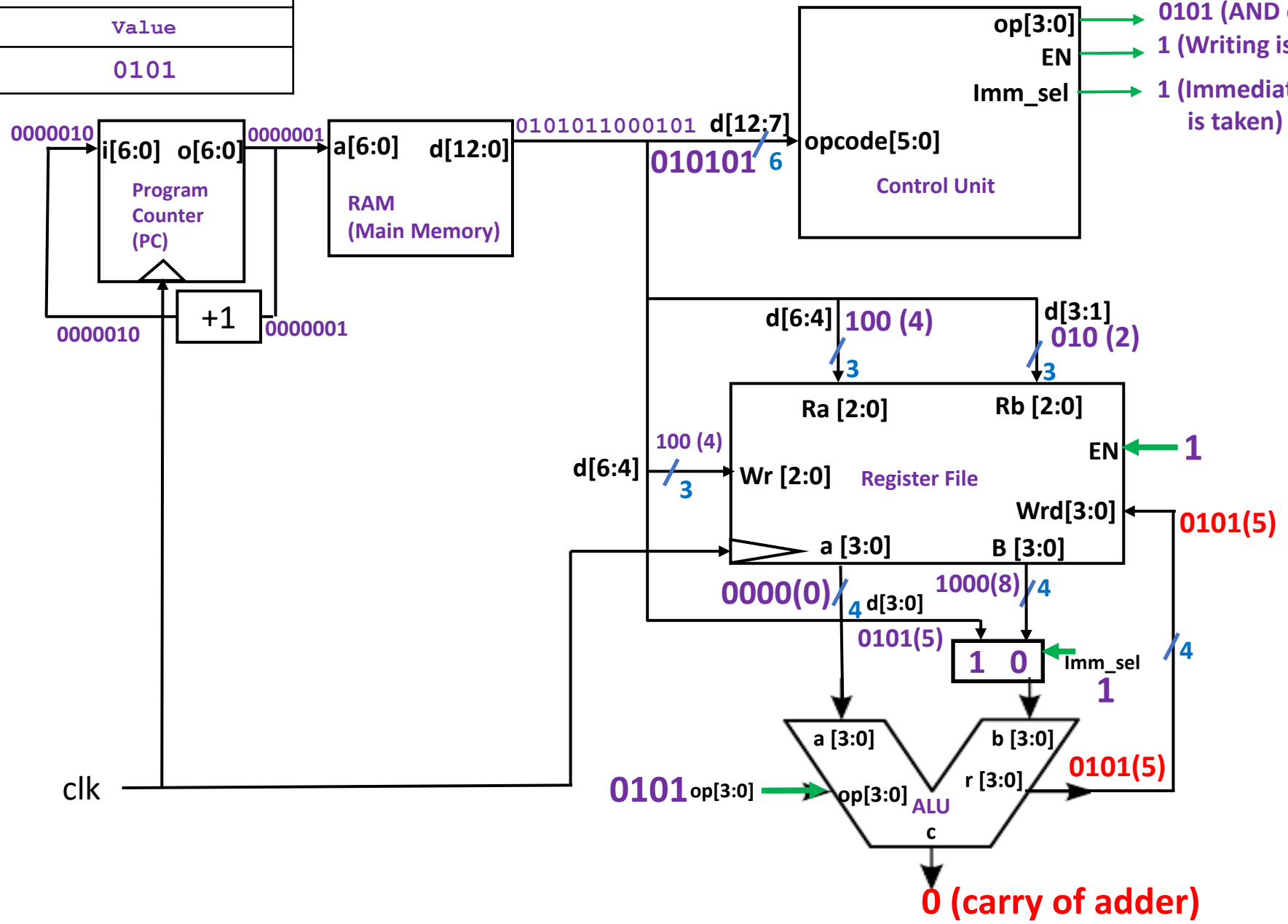
Instruction: 0101011000101
Opcode (6bits): 010101
Opcode (2bits): 01 (A&L Imm)
Opcode (4bits): 0101 (ADD)
Register 1: 100 **(R4)**
Value: 0101 **(5)**
Assembly: ADDI R4, 5



4-bit CPU

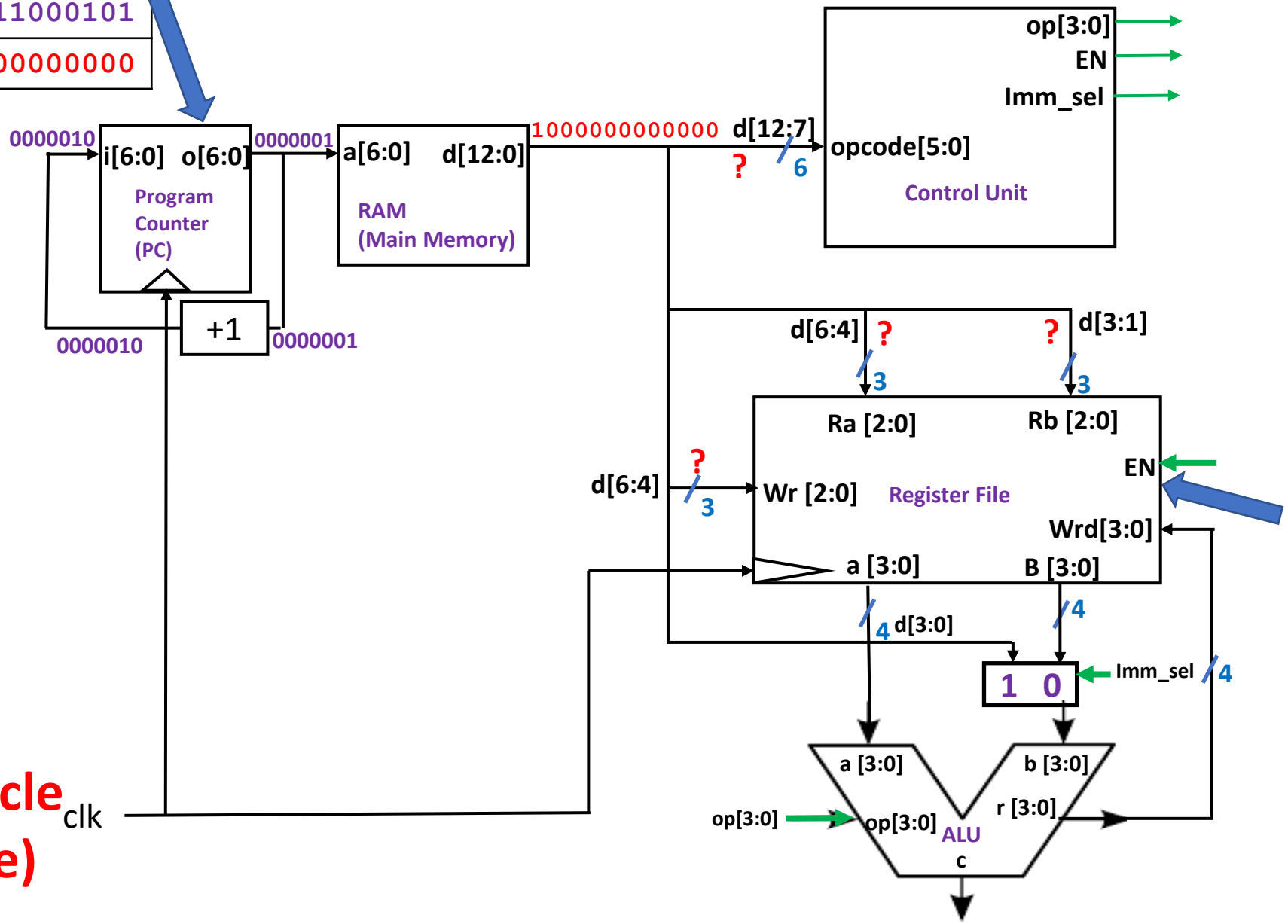
12 11	10 7	6 4	3 0
01	Operation	Register 1	Value
01	0101	100	0101

Instruction: 0101011000101
Opcode (6bits): 010101
Opcode (2bits): 01 (A&L Imm)
Opcode (4bits): 0101 (ADD)
Register 1: 100 **(R4)**
Value: 0101 **(5)**
Assembly: ADDI R4, 5



4-bit CPU

Address	Instructions
0000000	0000101001001
0000001	0101011000101
0000010	1000000000000



Next clock cycle (Positive Edge)

Address	Value
000 (R0)	0000
001 (R1)	0101
010 (R2)	1000
011 (R3)	0111
100 (R4)	0101
101 (R5)	0001
110 (R6)	0010
111 (R7)	0000

4-bit CPU

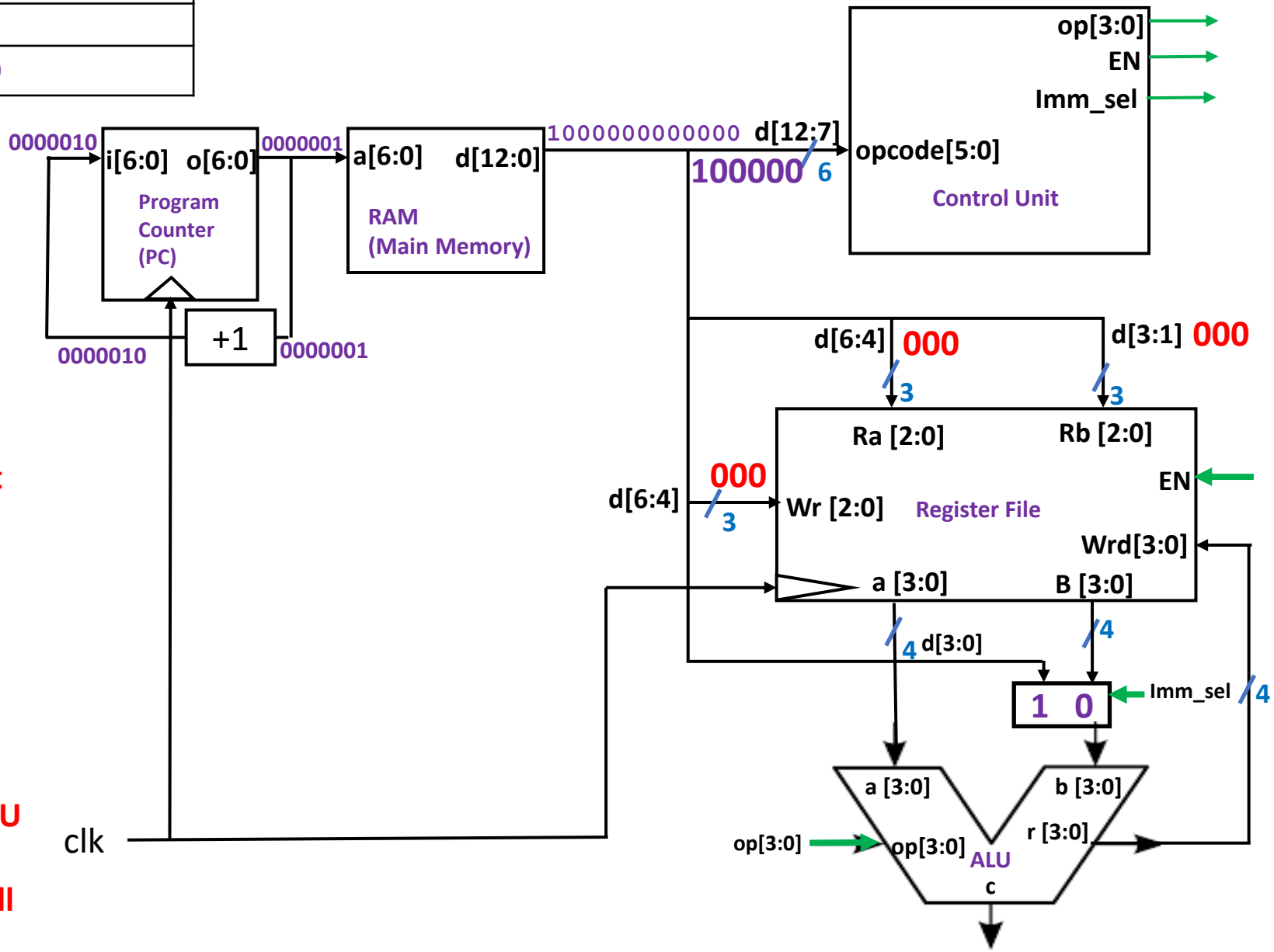
12 11	10 7	6 0
10	Operation	Address
10	0000	0000000

Instruction: 1000000000000
Opcode (6bits): 100000
Opcode (2bits): 00 **(Branching)**
Opcode (4bits): 0000 **(JMP)**
Address: 0000000 **(0)**
Assembly: JMP 0

For branching instructions, register set and ALU operations are not important (Consider them don't care).

Because we just have to update our program counter to jump to address specified in instruction.

In order to make sure that result of ALU operations will not overwrite value stored in register, EN of register set will be 0 (Writing operation disabled).

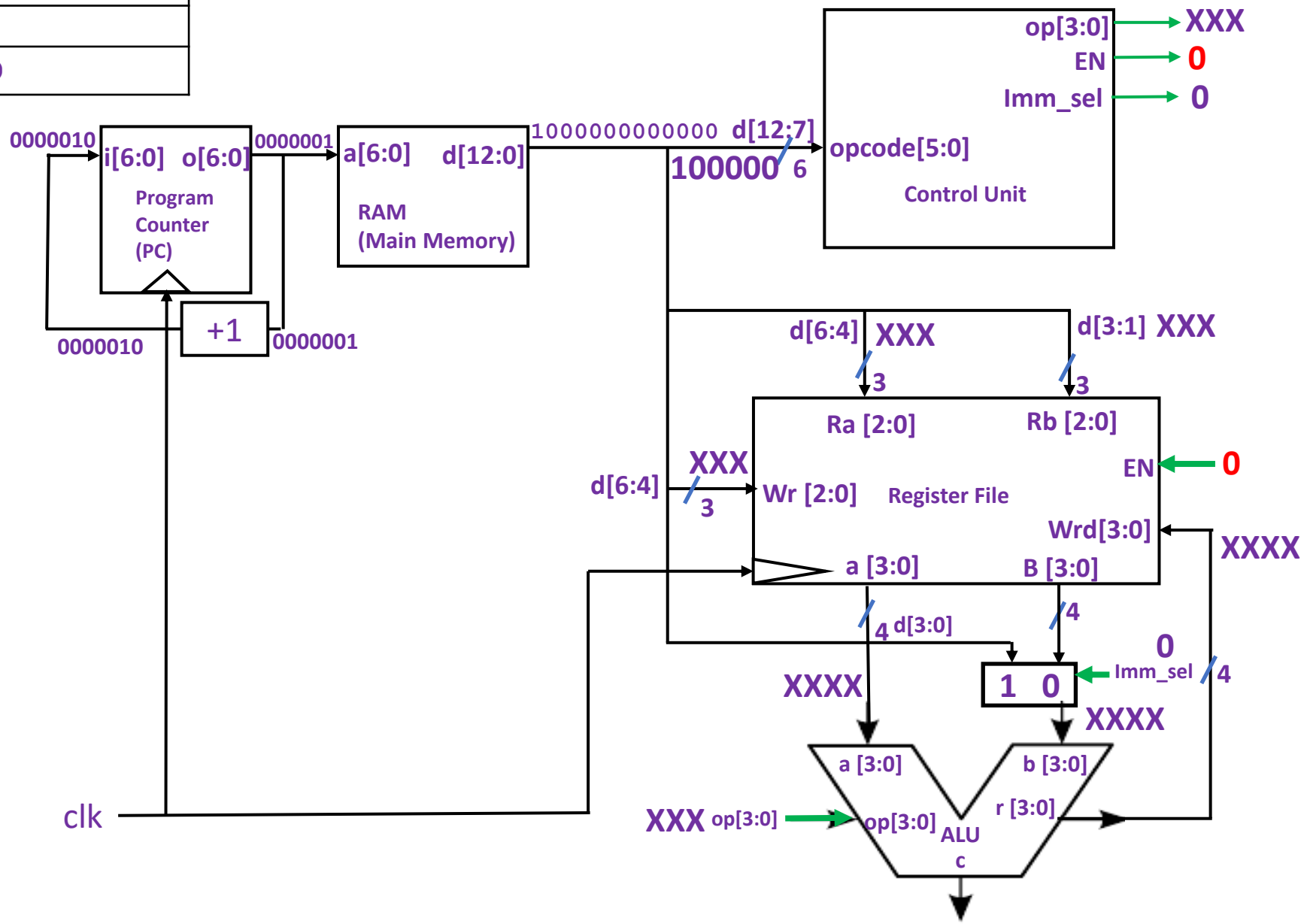


4-bit CPU

12 11	10 7	6 0
10	Operation	Address
10	0000	0000000

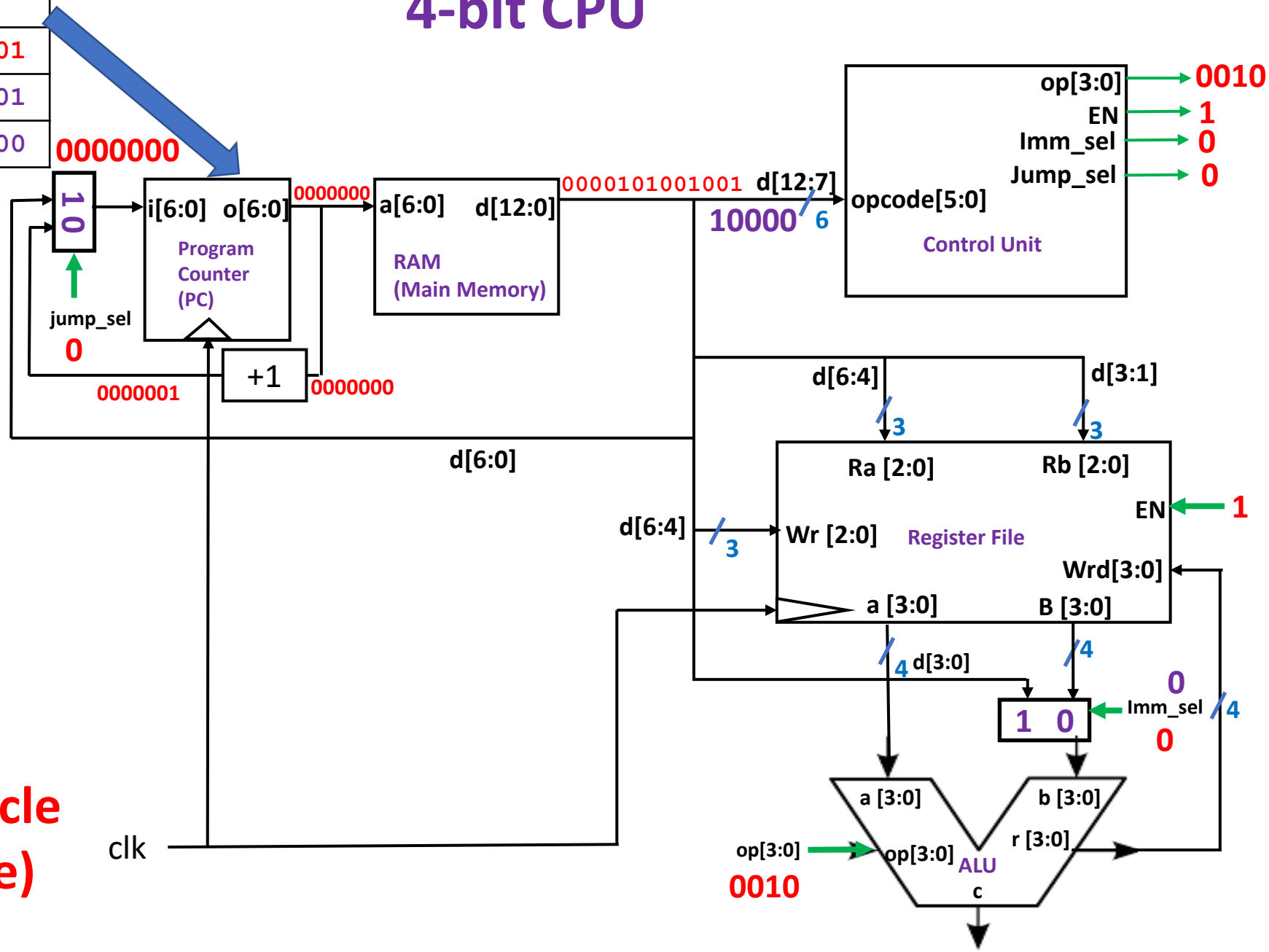
Instruction: 1000000000000
Opcode (6bits): 100000
Opcode (2bits): 00 **(Branching)**
Opcode (4bits): 0000 **(JMP)**
Address: 0000000 **(0)**
Assembly: JMP 0

To update our program counter, We have to use a 7 bit 2 to 1 MUX select between next address and jump address (D[6:0]).



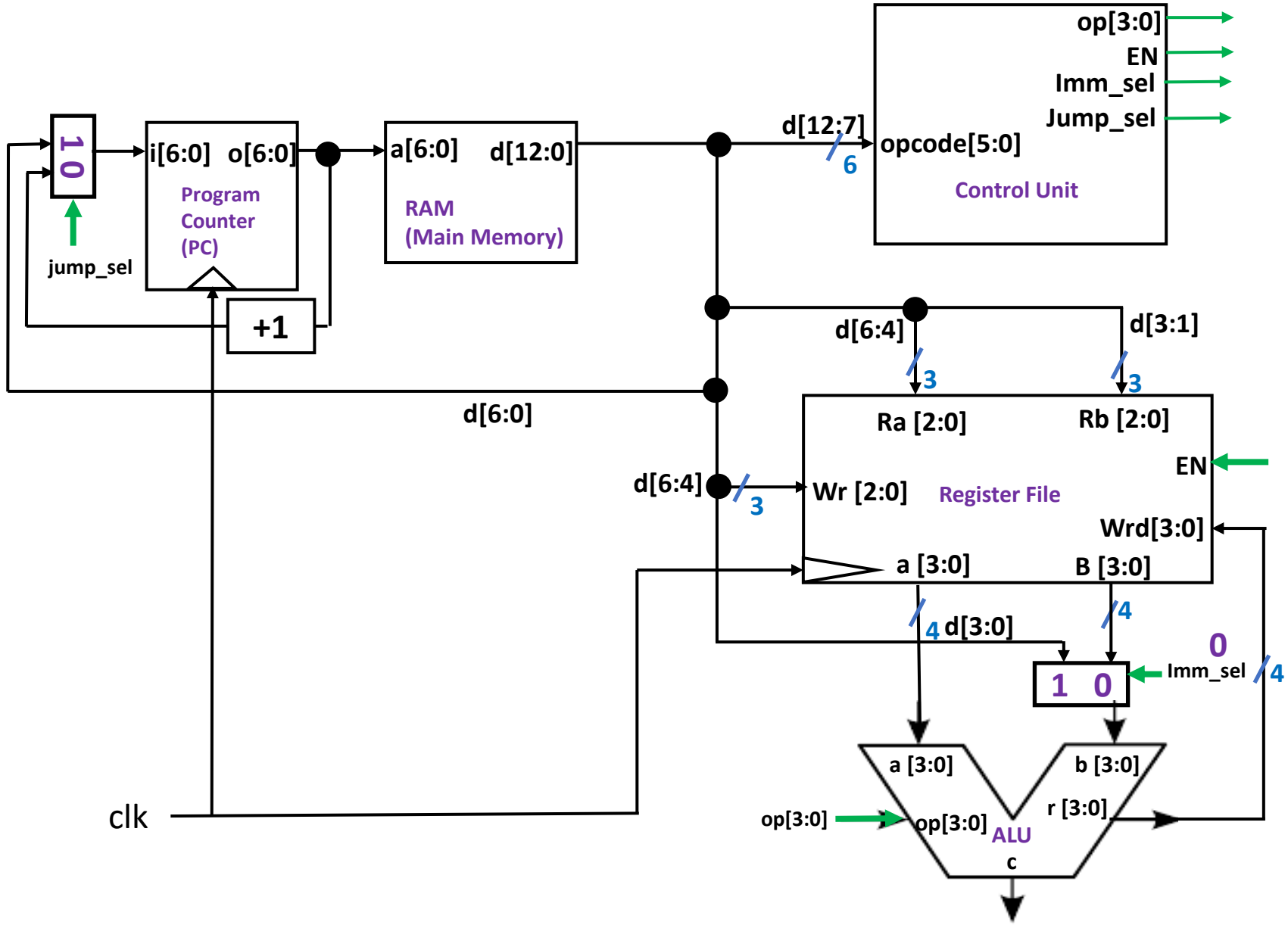
4-bit CPU

Address	Instructions
0000000	0000101001001
0000001	0101011000101
0000010	1000000000000



Next clock cycle (Positive Edge)

4-bit CPU (Final)



It only supports JMP instruction in Branching ISA.

It don't support other branching instructions like JGE, JNE etc.

Control Logic (Truth Table)

	Input	Output			
	d[12:7]	Op[3:0]	EN	Imm_sel	Jump_sel
Arithmetric & Logic Instructions (Register Mode)	00AAAA	AAAA	1 **Except 1011 (CMP)** **EN = 0**	0	0
Arithmetric & Logic Instructions (Immediate Mode)	01BBBB	BBBB	1	1	0
Branching Instructions	10CCCC	CCCC	0	0	1

$$Op[3] = d[10]$$
$$Op[2] = d[9]$$
$$Op[1] = d[8]$$
$$Op[0] = d[7]$$

$$EN = (\overline{d[12]}.\overline{d[11]} + \overline{d[12]}.d[11])$$

$$Imm\_sel = \overline{d[12]}.d[11]$$

$$Jump\_sel = d[12].\overline{d[11]}$$

Thank you ☺