CPU Design

Nahin Ul Sadad
Lecturer
CSE, RUET

Central Processing Unit (CPU)

A central processing unit (CPU)/central processor/main processor/processor/microprocessor is the electronic circuitry within a computer that executes instructions that make up a computer program.

The CPU performs basic arithmetic, logic, controlling, and input/output (I/O) operations specified by the instructions in the program.

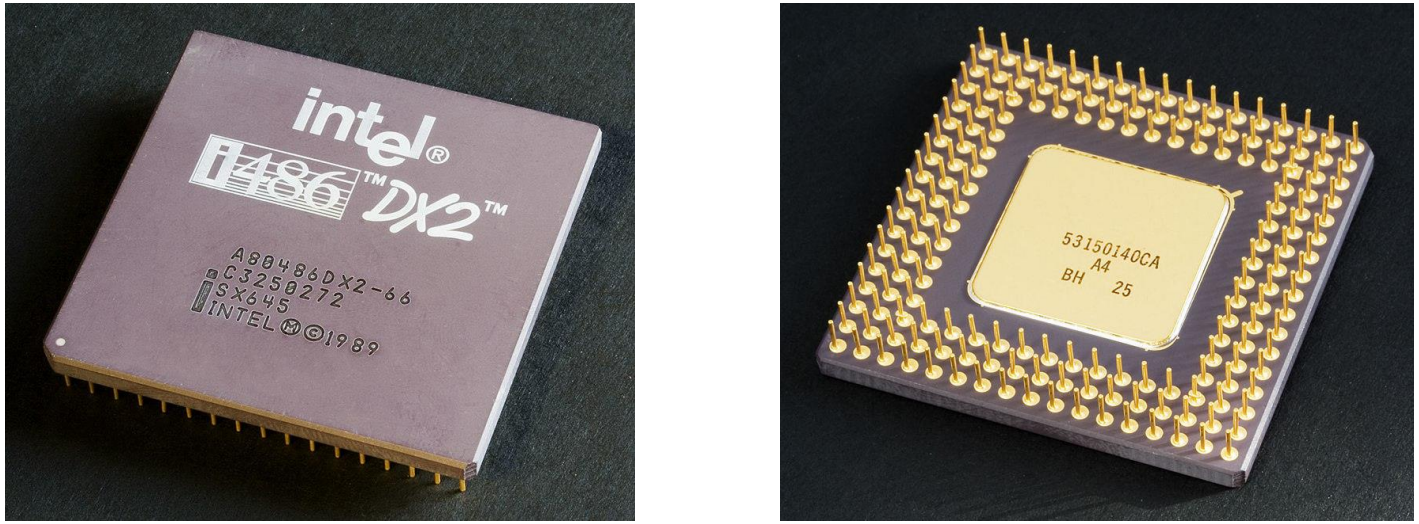


Figure: Front side and Bottom side of Intel CPU.

Microarchitecture/Computer Organization

In computer engineering, Microarchitecture/Computer Organization is the way a given instruction set architecture (ISA) is implemented in a particular processor.

It is sometimes abbreviated as µarch or uarch. A given ISA may be implemented with different microarchitectures. Implementations may vary due to different goals of a given design or due to shifts in technology.

For example, both Intel & AMD creates CPU based on x86-64 architecture (ISA). Microarchitecture of Intel & AMD CPU will be different even though both CPU has same ISA.

Computer Architecture

Computer architecture is the combination of microarchitecture and instruction set architecture (ISA).

Computer architecture involves instruction set architecture design, microarchitecture design, logic design, and implementation.

n-bit CPU

n-bit is a type of CPU architecture that is capable of transferring n bits of data per clock cycle. It is the amount of information your CPU can process each time it performs an operation. In more technical terms, this means processors can work with 32-bit binary numbers.

For example, our 4-bit CPU can perform ALU operations on 4-bit data. So, ALU must be 4-bit. Register size also will be 4-bit. It means it can store 4-bit data at a time.

Performance Equation

As we know basic performance equation,

$$T = \frac{N \times S}{R}$$

where, N is the actual number of instruction executions

S is the average number of basic steps needed to execute one machine instruction

T is the processor time required to execute a program that has been prepared in some high-level language and

R is the clock rate

RISC vs CISC

As we know basic performance equation,

$$T = \frac{N \times S}{R}$$

where, N is the actual number of instruction executions

S is the average number of basic steps needed to execute one machine instruction

T is the processor time required to execute a program that has been prepared in some high-level language and

R is the clock rate

1. **RISC:** In Reduced Instruction Set Computers (RISC), smaller S likely leads to larger N. So, the RISC approach attempts to minimize the cycles per instruction at the cost of the number of instructions per program.

2. **CISC:** In Complex Instruction Set Computers (CISC), larger S likely leads to smaller N. So, the CISC approach attempts to minimize the number of instructions per program at the cost of the number of cycles per instruction.

RISC vs CISC

RISC stands for Reduced Instruction Set Computer. RISC processor usually execute an instruction within one clock cycle. Size of each instruction is fixed in RISC processor. RISC processor usually has smaller instruction set, allowing faster accessing of “common” instructions. Length of program written for RISC processor is larger than the same program written for CISC processor. Example of real-life RISC processor is ARM, MIPS etc.

CISC stands for Complex Instruction Set Computers. CISC processor usually execute an instruction within more than one clock cycle. CISC processor usually has large and complicated instruction set, allowing faster accessing of “specialized” instructions. Example of real-life CISC processor is Intel, AMD etc.

Byte Ordering/Endianness

Byte Ordering/Endianness refers to the order of the bytes comprising a digital word in computer memory. It refers to the order of multi-byte values are stored by the hardware.

There are two types of byte ordering. They are:

1. **Big endian:** The mapping on the left stores the most significant byte in the lowest numerical byte address is called big endian. It is equivalent to left-to-right order.

   For example, some machines such as the Intel 80x86, x86, VAX, and Alpha are little-endian machines.

2. **Little endian:** The mapping on the right stores the least significant byte in the lowest numerical byte address is called little endian. It is equivalent to right-to-left order.

   For example, some machines such as the IBM System 370/390, the Motorola 680x0, Sun SPARC, and most RISC machines are big endian machines.

Byte Ordering/Endianness

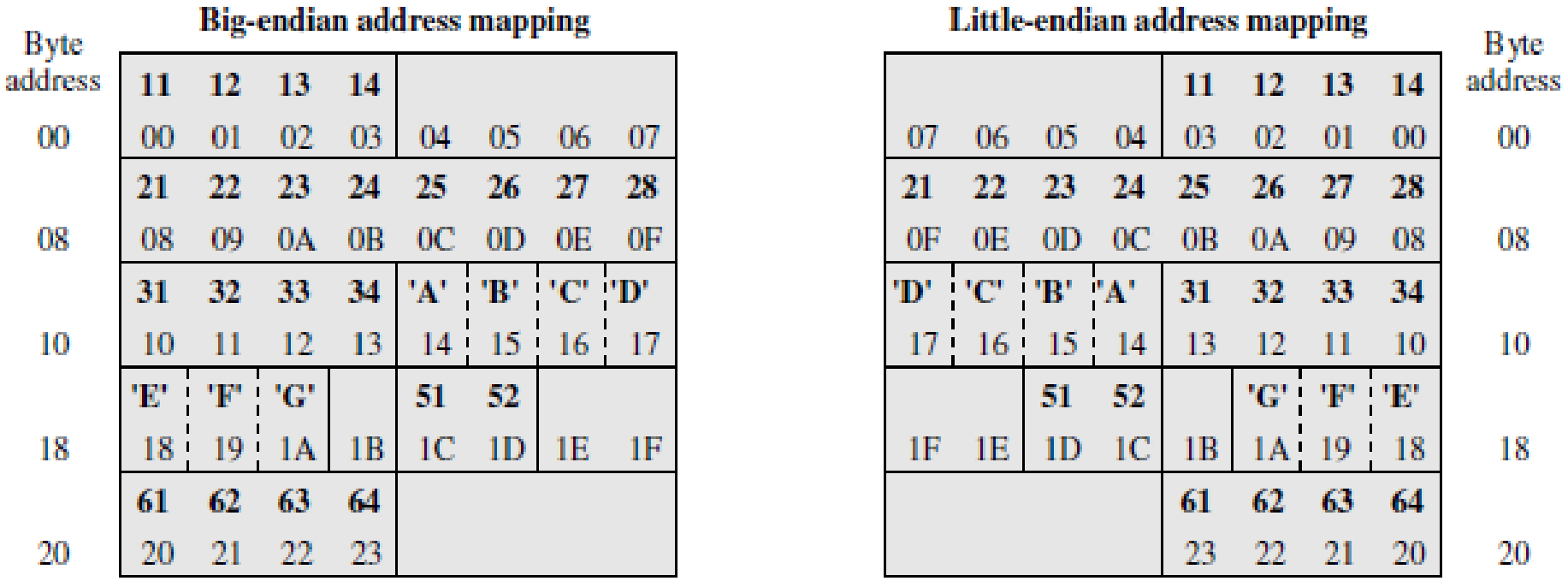


Figure 10.18 Example C Data Structure and its Endian Maps

Figure: Example of Big-Endian and Little-Endian Byte Ordering

Program Counter (PC)

Program counter keeps track of address of current instruction. Its next address will also be calculated at the same time. But it will not be updated until the next clock cycle.

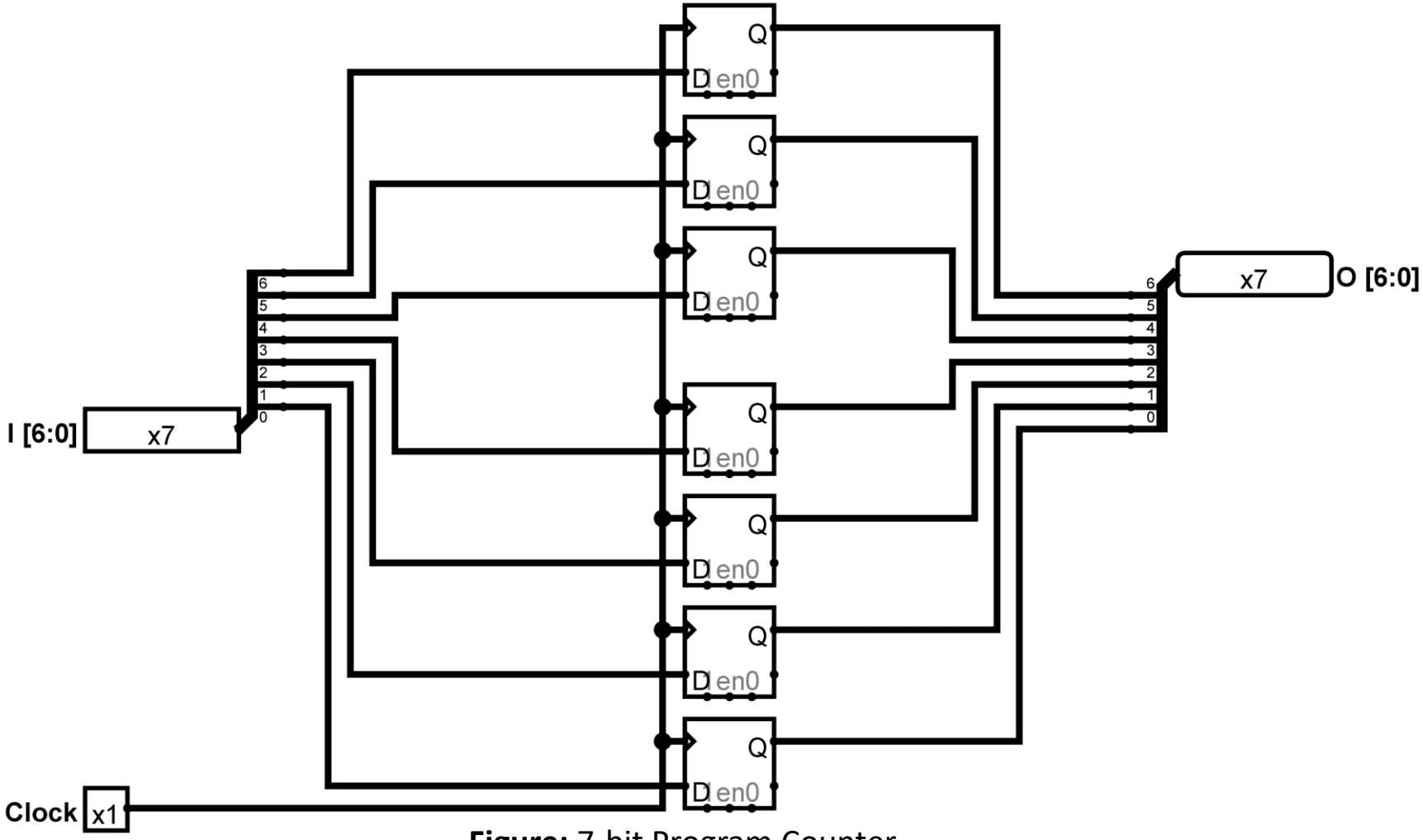


Figure: 7-bit Program Counter

Program Counter Adder

Program Counter Adder adds 1 with Current Address to get the Next Address.

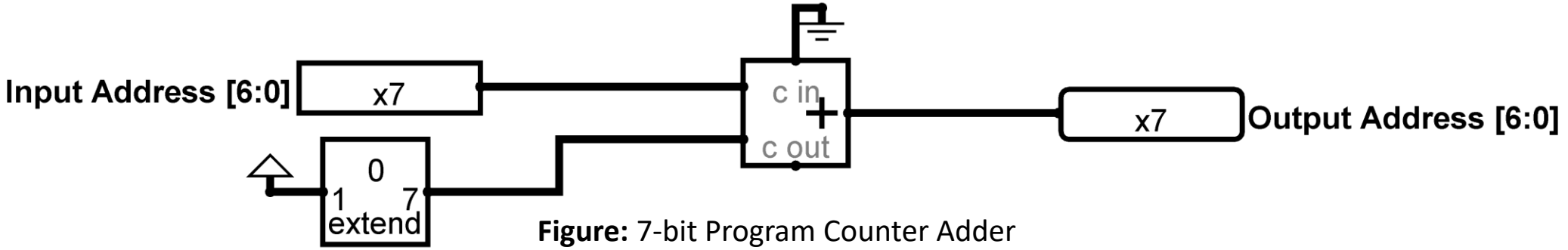


Figure: 7-bit Program Counter Adder

Read Only Memory (ROM)

ROM will store programs which will then be loaded into Main Memory (RAM). Because ROM is non-volatile and RAM is volatile memory.

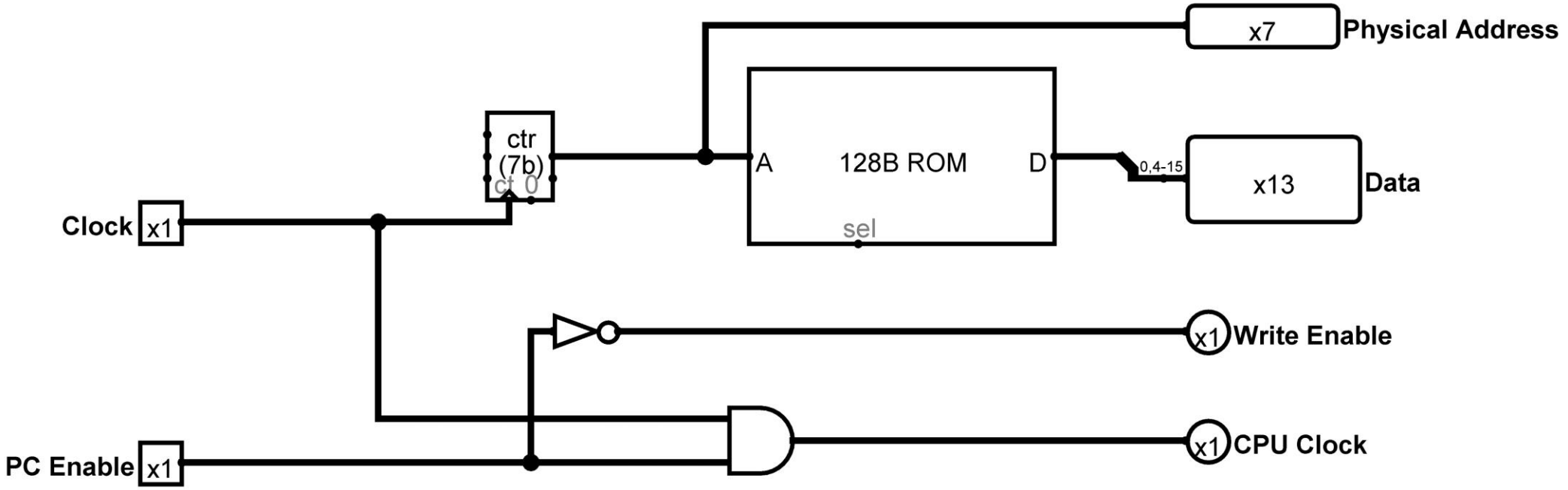


Figure: ROM

Read Only Memory (ROM)

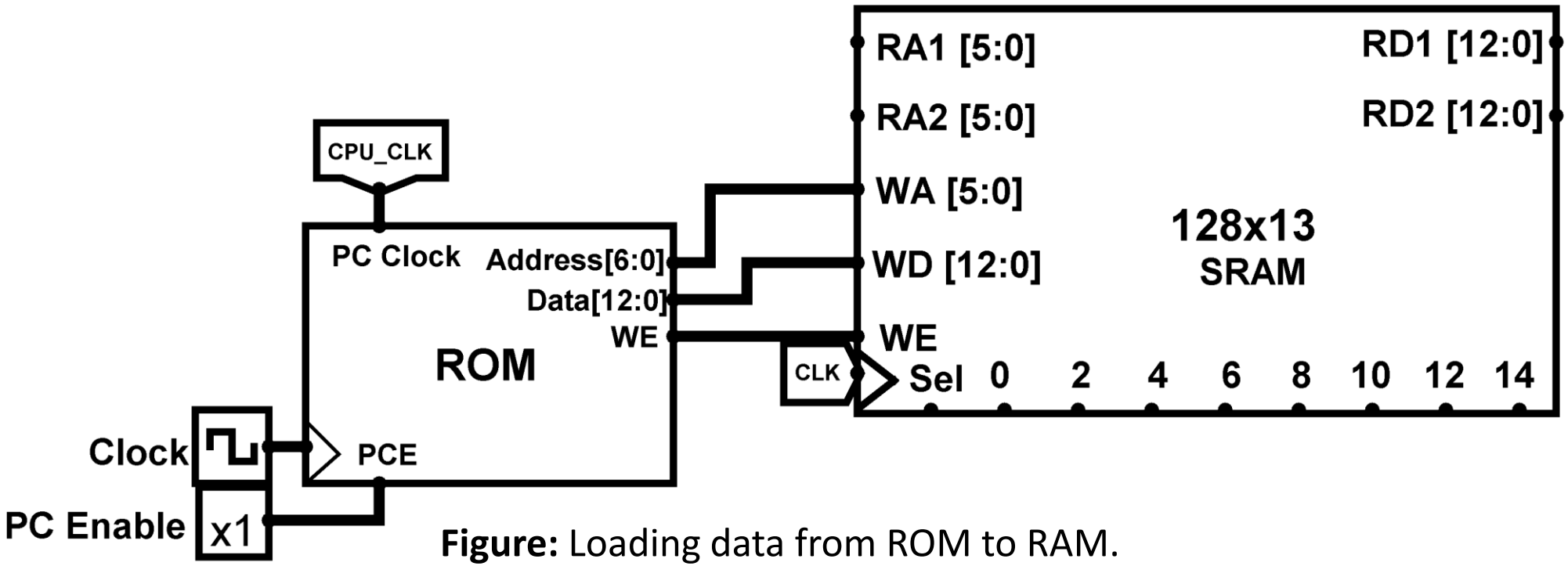


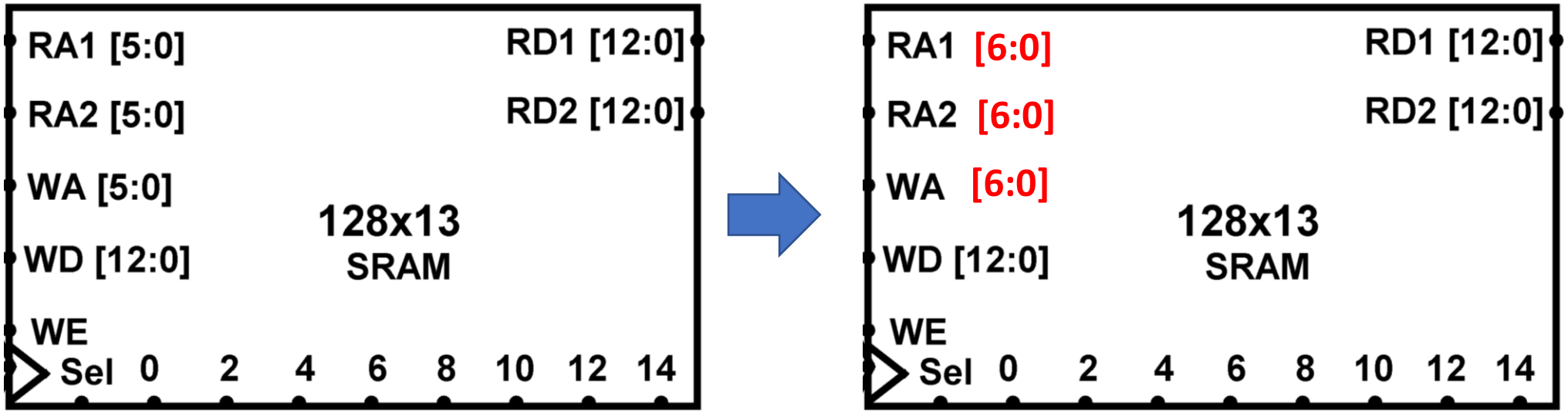
Figure: Loading data from ROM to RAM.

Here, Clock is the real clock that is supplied.

At the beginning, Data of ROM will be loaded into RAM and CPU will be disabled by disabling CPU_CLK as PC Enable is 0.

After loading data from ROM into RAM, PC Enable will be 1 which will enable CPU_CLK and CPU will start running.

Errata



In RAM chip, bus width of RA1, RA2 & WA must be [6:0] instead of [5:0] because $2^7 = 128$. But this chip is used in rest of the lecture.

Sample Code

Consider assembly code with address:

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

Address	Instructions
0000000	0000101001001
0000001	0101011000101
0000010	1000000000000

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

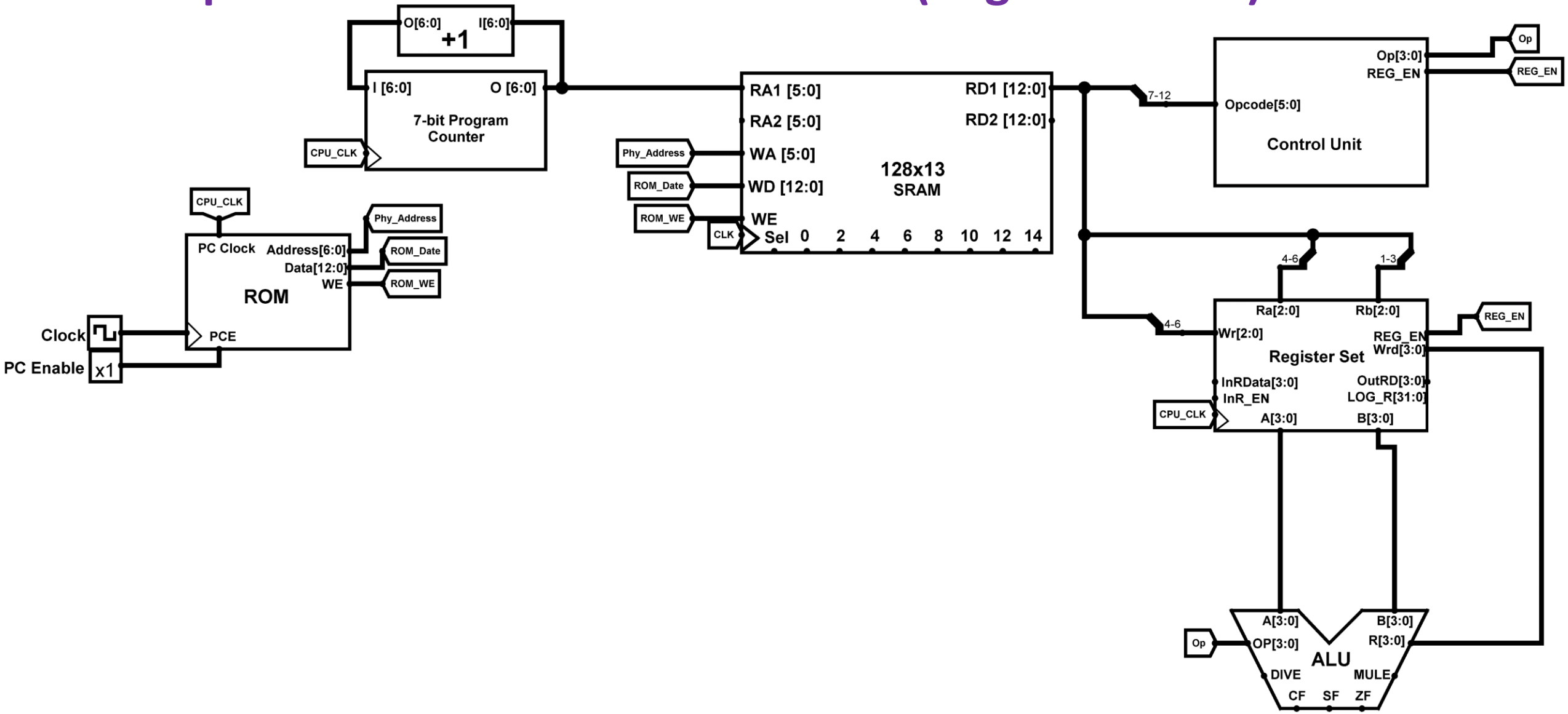
Convention,
Ra = Ra OP Rb

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

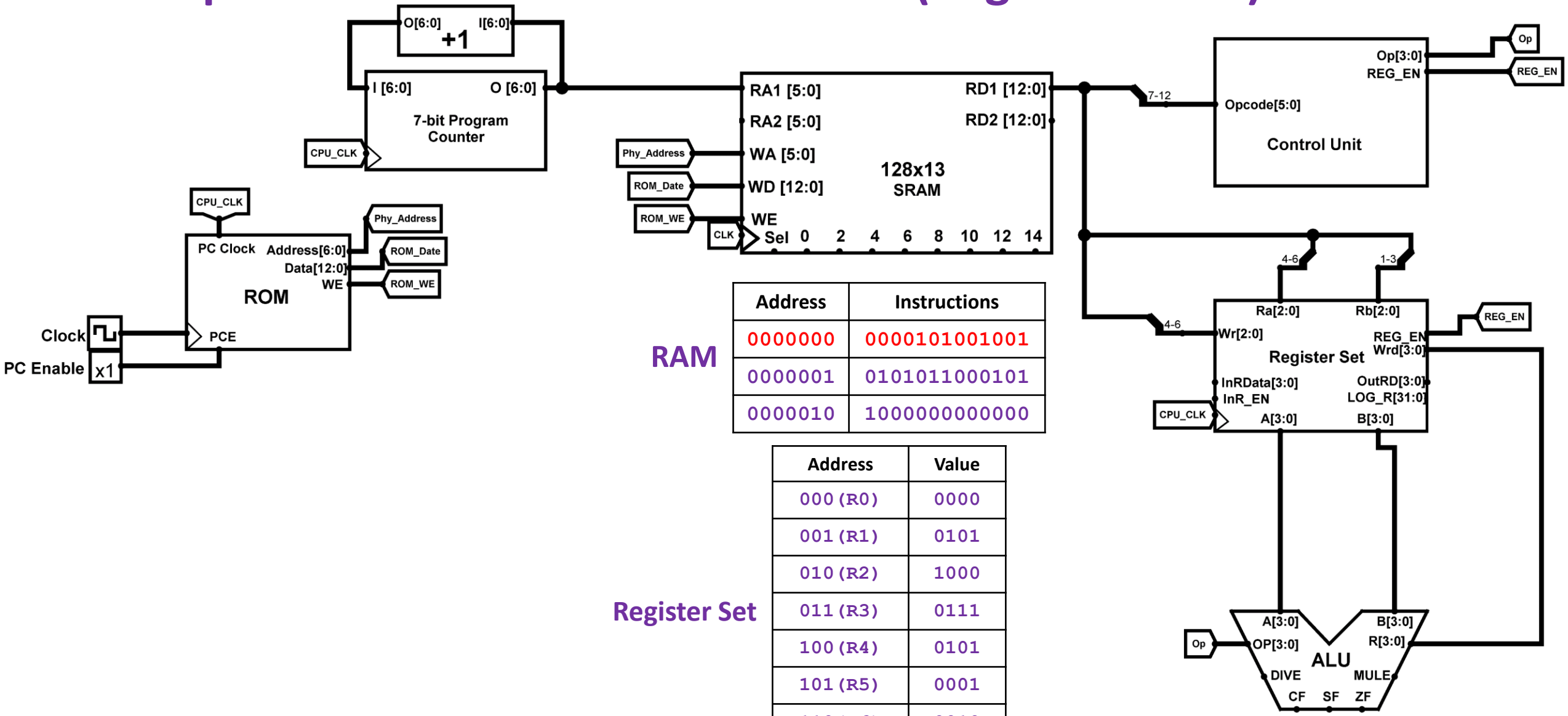
ISA of Arithmetic & Logic Instruction (Register Mode)

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

Implementation of ALU Instruction (Register Mode) in CPU



Implementation of ALU Instruction (Register Mode) in CPU



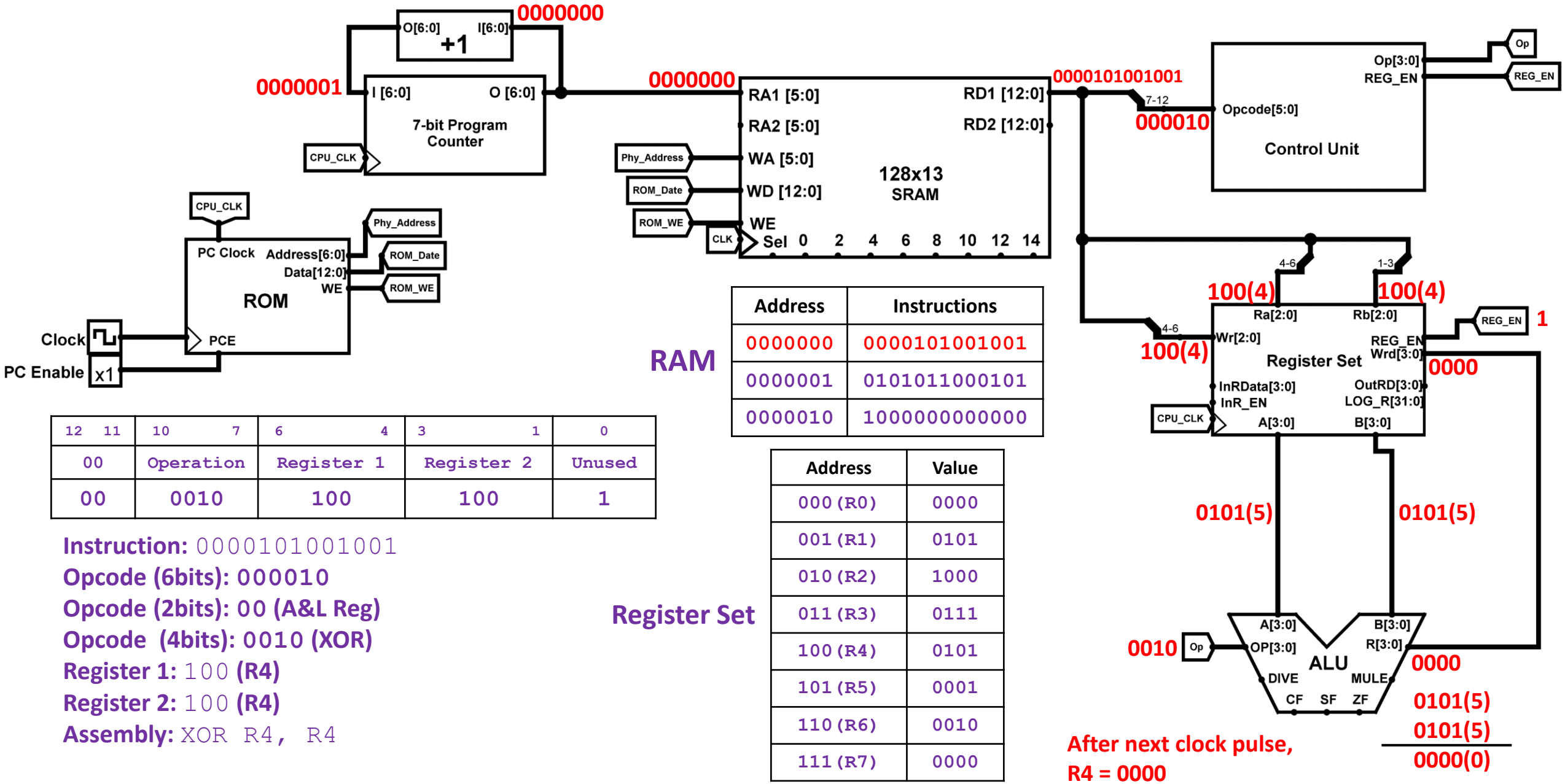
RAM

Address	Instructions
0000000	0000101001001
0000001	0101011000101
0000010	1000000000000

Register Set

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

Implementation of ALU Instruction (Register Mode) in 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

RAM

Address	Instructions
0000000	0000101001001
0000001	0101011000101
0000010	1000000000000

Register Set

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

After next clock pulse,
R4 = 0000

Control Logic for ALU Instruction (Register Mode) in CPU

Control Logic (Truth Table)

	Input	Output	
	Opcode[5:0]	Op[3:0]	REG_EN
Arithmetic & Logic Instructions (Register Mode)	00AAAA	AAAA	1 Except 1011 (CMP) REG_EN = 0

Control Logic Circuit

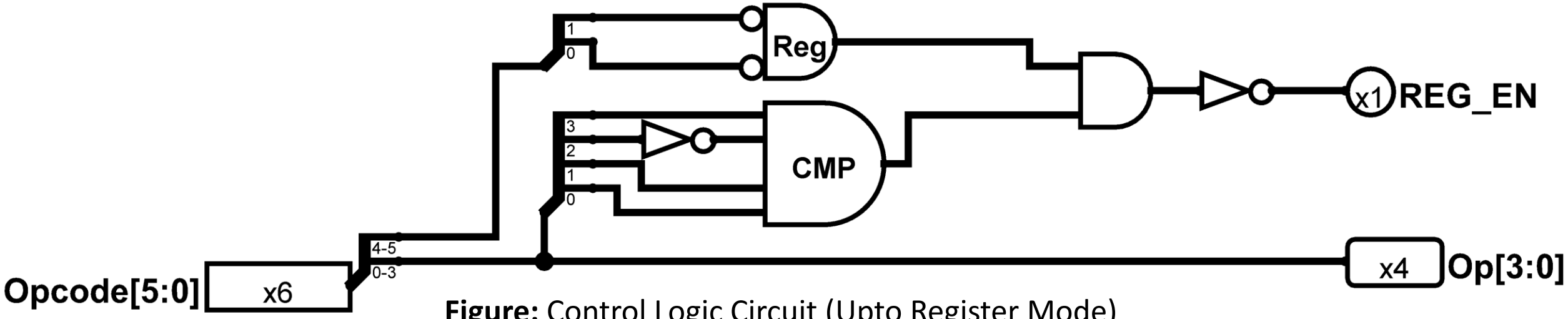


Figure: Control Logic Circuit (Upto Register Mode)

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)

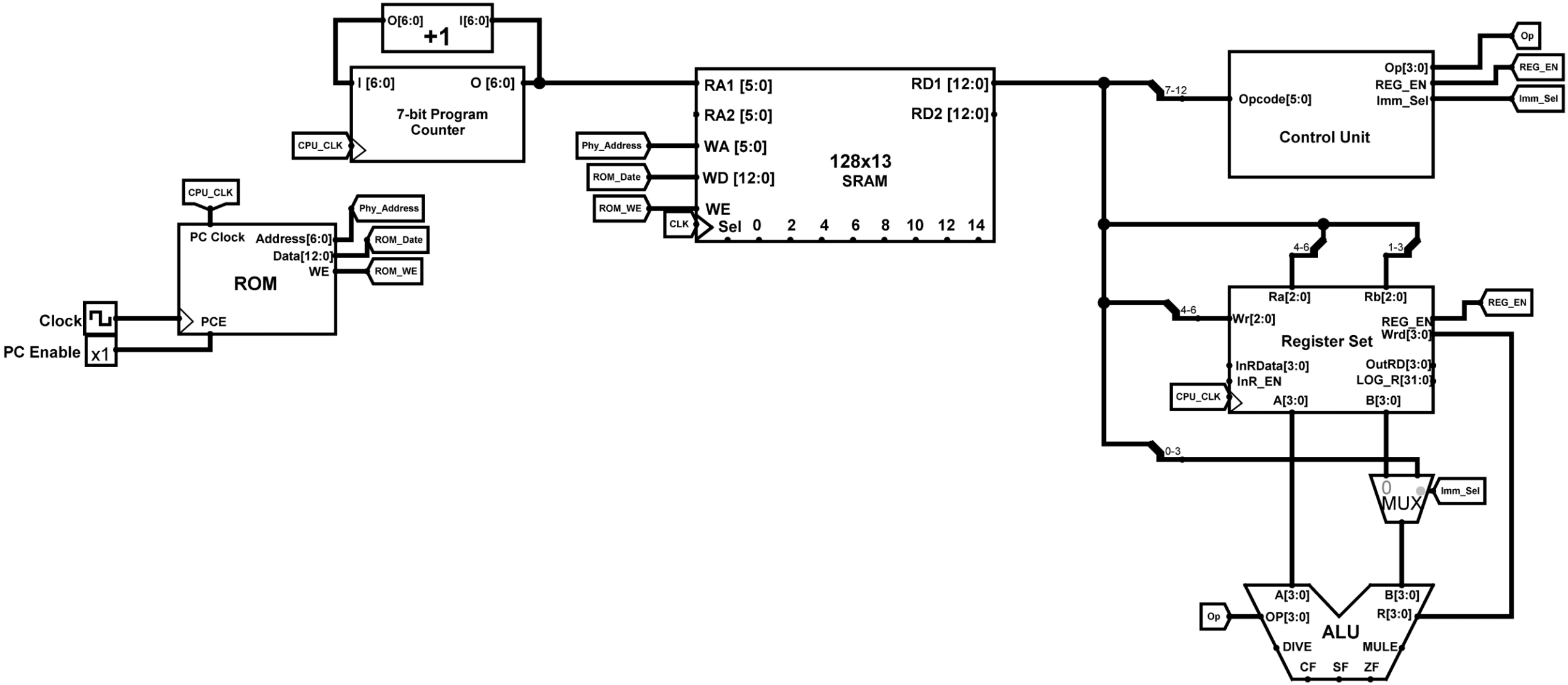
Convention,
Ra = Ra OP Constant

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

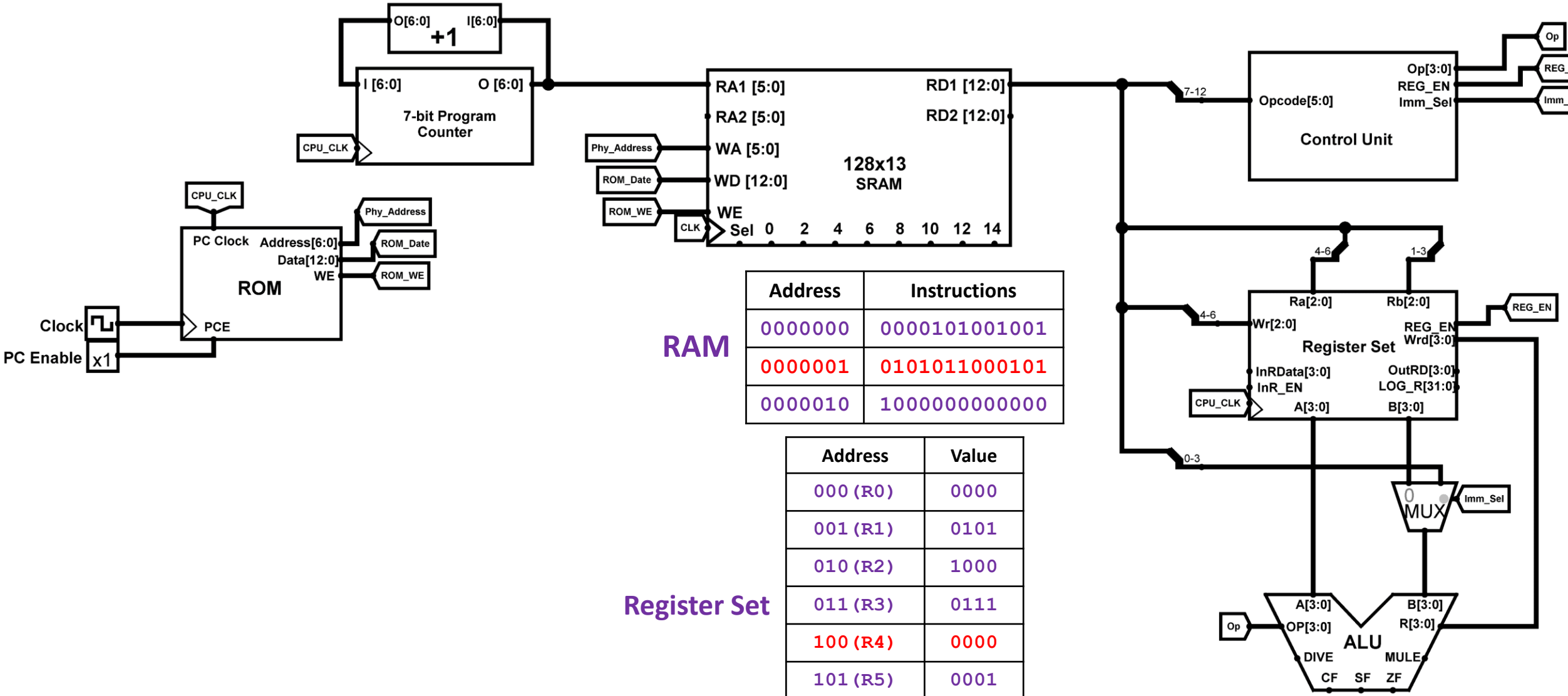
ISA of Arithmetic & Logic Instruction (Immediate Mode)

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

Implementation of ALU Instruction (Immediate Mode) in CPU



Implementation of ALU Instruction (Immediate Mode) in CPU



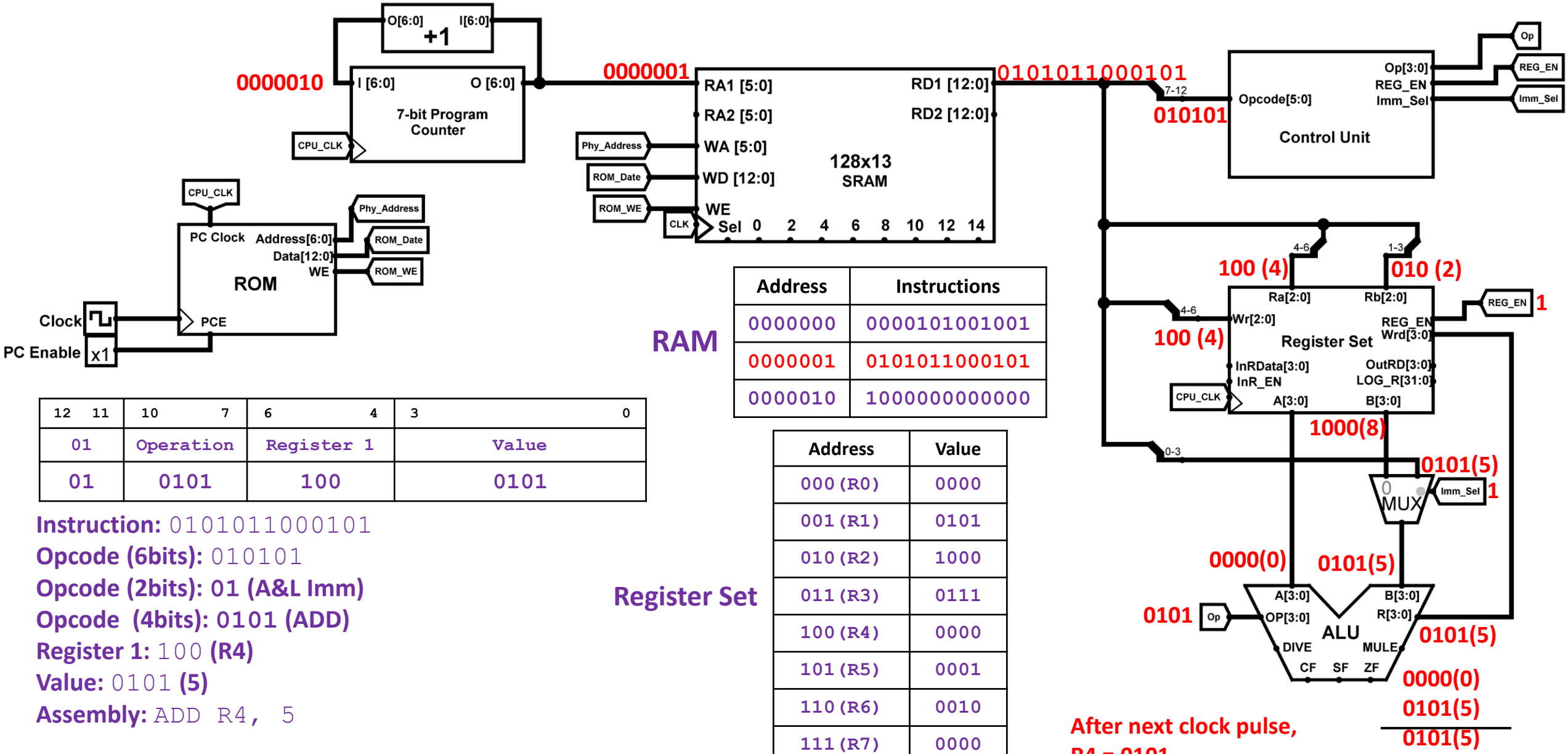
RAM

Address	Instructions
0000000	0000101001001
0000001	0101011000101
0000010	1000000000000

Register Set

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

Implementation of ALU Instruction (Immediate Mode) in 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: ADD R4, 5

RAM

Address	Instructions
0000000	0000101001001
0000001	0101011000101
0000010	1000000000000

Register Set

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

After next clock pulse,
R4 = 0101

Control Logic for ALU Instruction (Immediate Mode) in CPU

Control Logic (Truth Table)

	Input	Output		
	`Opcode[5:0]`	`Op[3:0]`	`REG_EN`	`Imm_sel`
Arithmetic & Logic Instructions (Register Mode)	00AAAA	AAAA	1 Except 1011 (CMP) `REG_EN` = 0	**0**
Arithmetic & Logic Instructions (Immediate Mode)	**01BBBB**	**BBBB**	**1** **Except 1011 (CMP)** **`REG_EN` = 0**	**1**

Control Logic Circuit

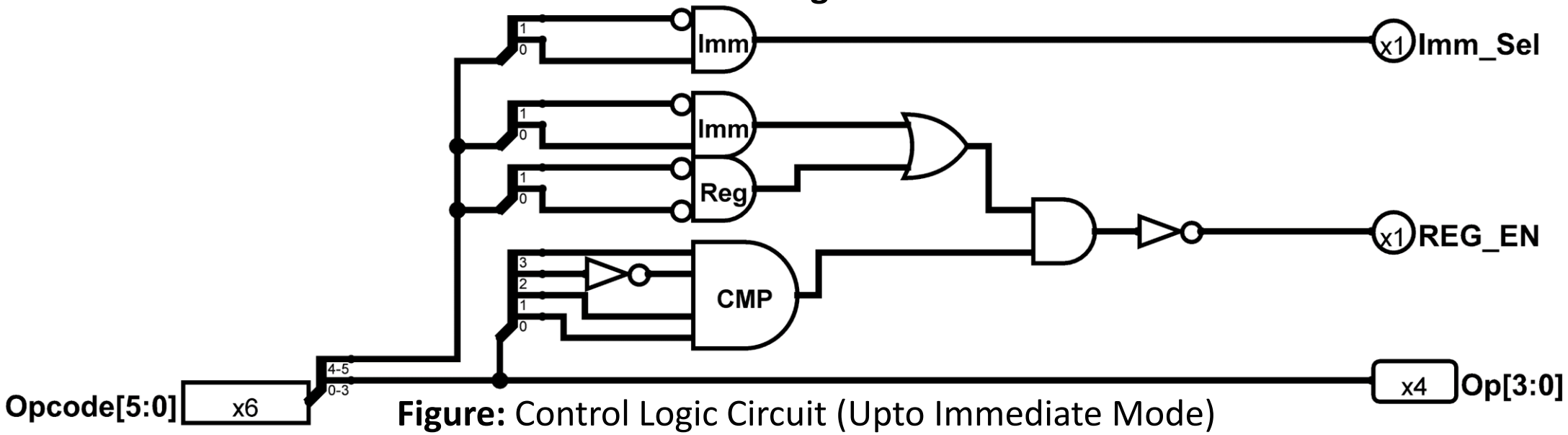


Figure: Control Logic Circuit (Upto Immediate Mode)

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

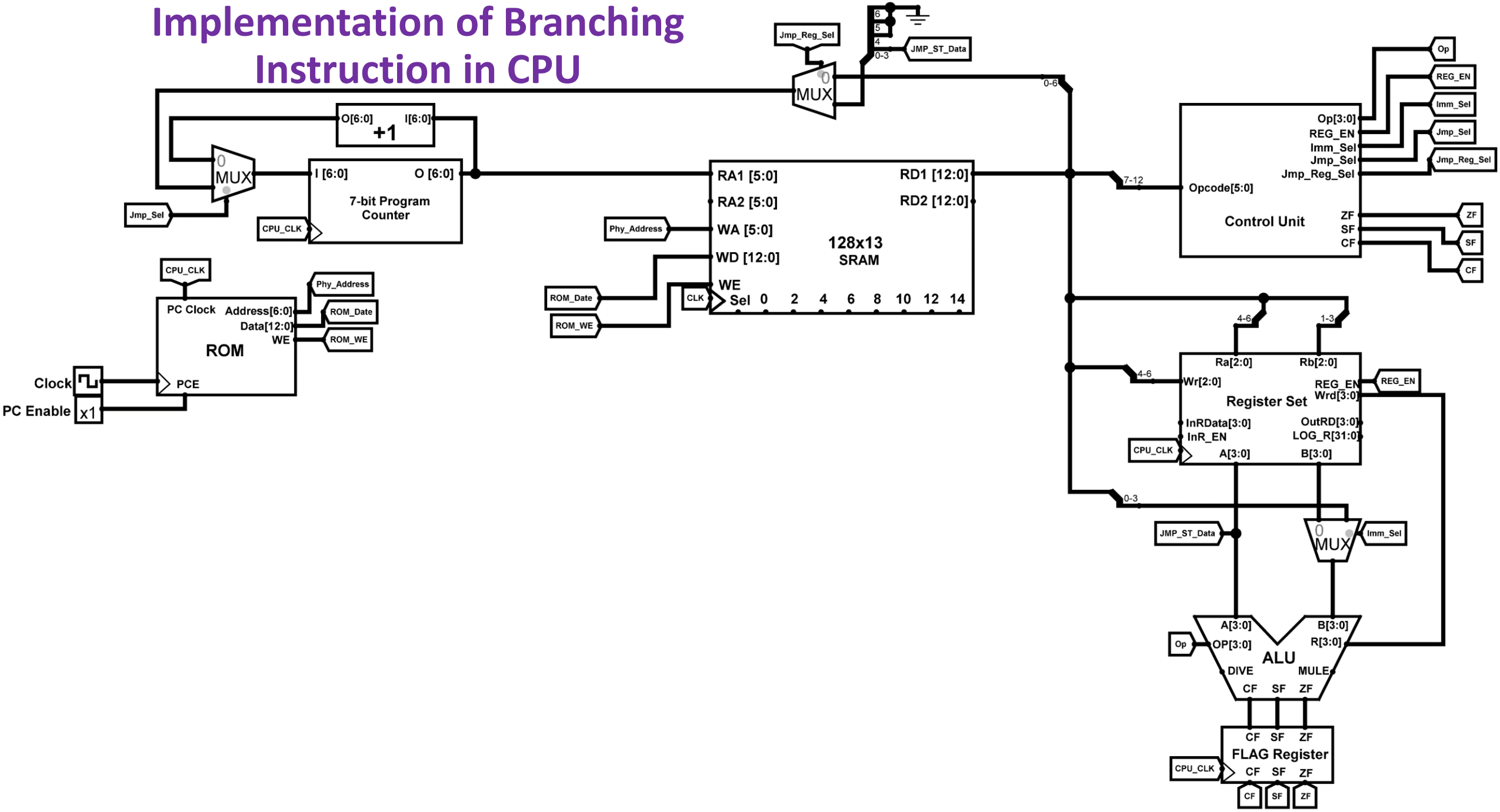
For JMPREG instruction,

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

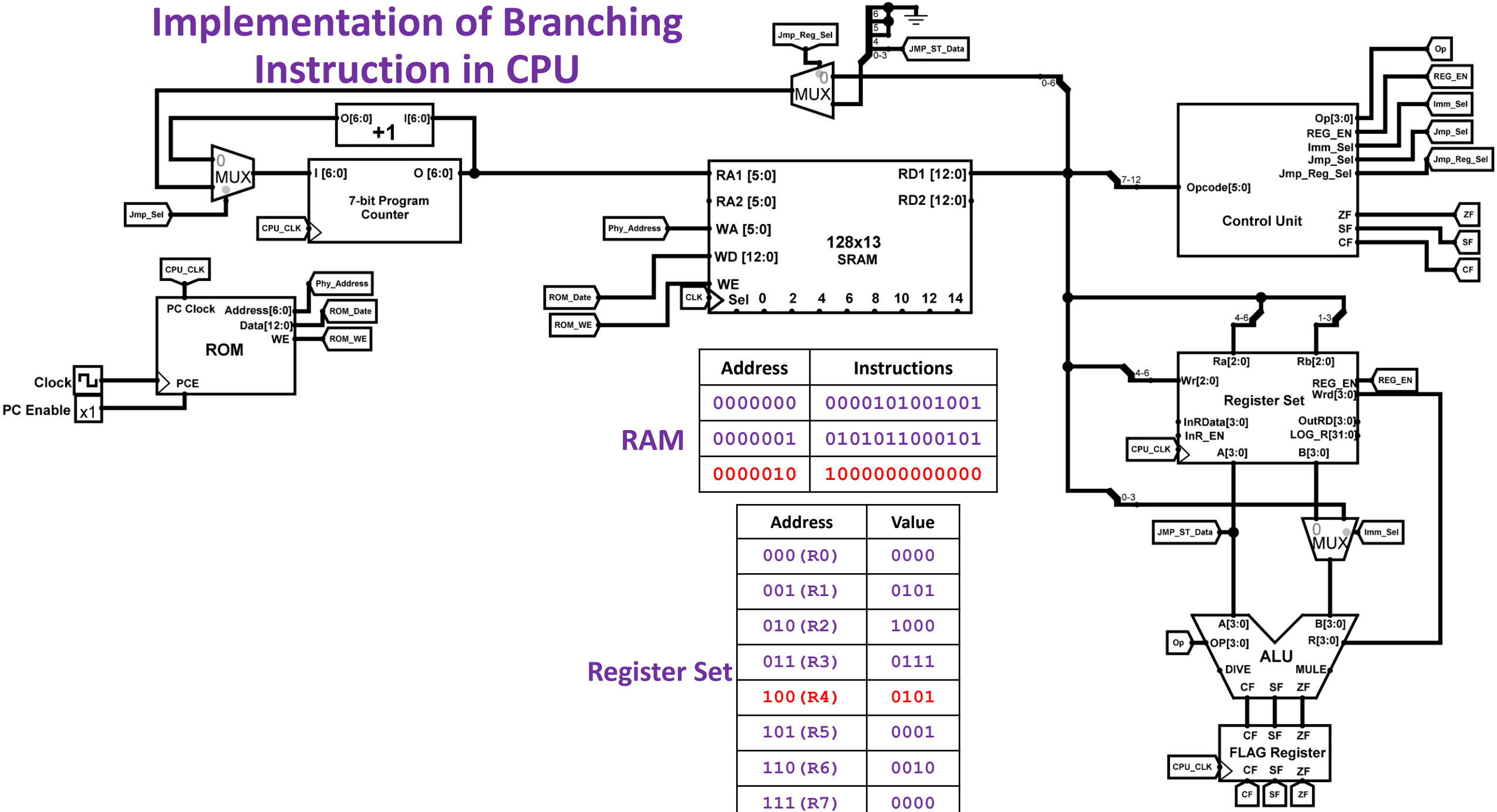
ISA of Branching

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)	X	000-100 (0-4) (Last 4 bits are unused)	JMPREG R1
	1100 (XXX)	X	X	XXX
	1101 (XXX)	X	X	XXX

Implementation of Branching Instruction in CPU



Implementation of Branching Instruction in CPU



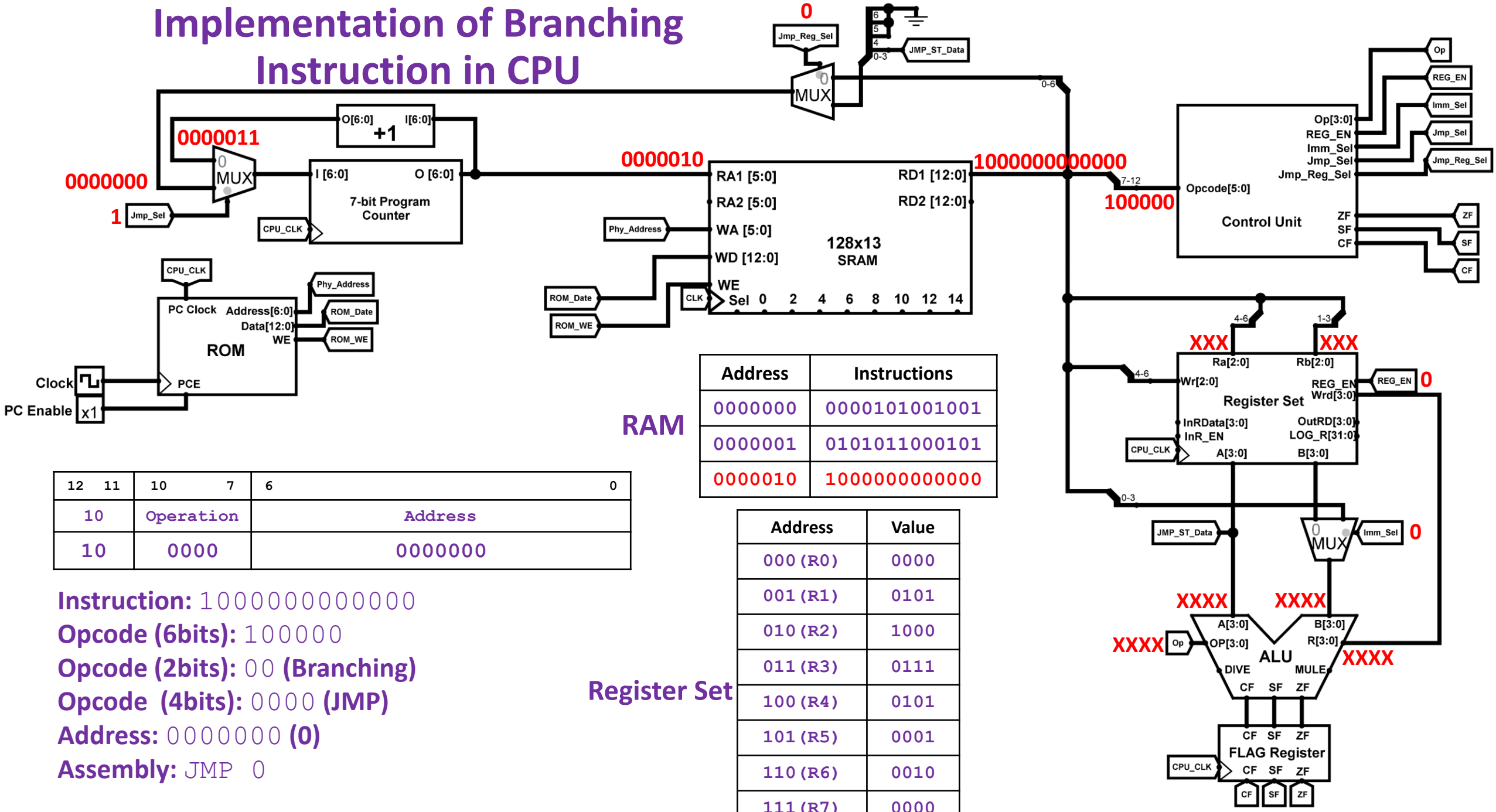
RAM

Address	Instructions
0000000	0000101001001
0000001	0101011000101
0000010	1000000000000

Register Set

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

Implementation of Branching Instruction in 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

RAM

Address	Instructions
0000000	0000101001001
0000001	0101011000101
0000010	1000000000000

Register Set

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

Control Logic for Branching Instruction in CPU

Control Logic (Truth Table)

	Input					Output				
	Opcode [5:4]	Opcode [3:0]	ZF	SF	CF	Op [3:0]	REG_EN	Imm _sel	Jmp _sel	Jmp _Reg_Sel
Arithmetic & Logic Instructions (Register Mode)	00	AAAA				AAAA	1 **Except 1011**	**0**	**0**	**0**
Arithmetic & Logic Instructions (Immediate Mode)	01	BBBB				BBBB	1 **Except 1011**	**1**	**0**	**0**
Branching Instructions	**10**	**0000 (JMP)**	**X**	**X**	**X**	**XXXX**	**0**	**0**	**1**	**0**
		0001 (JE)	**1**	**0**	**X**					
		0010 (JNE)	**0**	**X**	**X**					
		0011 (JL)	**0**	**1**	**X**					
		0100 (JLE)	**sf=1 or zf=1**		**X**					
		0101 (JG)	**0**	**0**	**X**					
		0110 (JGE)	**sf=0 or zf=1**		**X**					
		0111 (JC)	**X**	**X**	**1**					
		1000 (JNC)	**X**	**X**	**0**					
		1001 (JZ)	**1**	**X**	**X**					
		1010 (JNZ)	**0**	**X**	**X**					
		1011 (JMPREG)	**X**	**X**	**X**					**1**

Control Logic for Branching Instruction in CPU

Control Logic Circuit

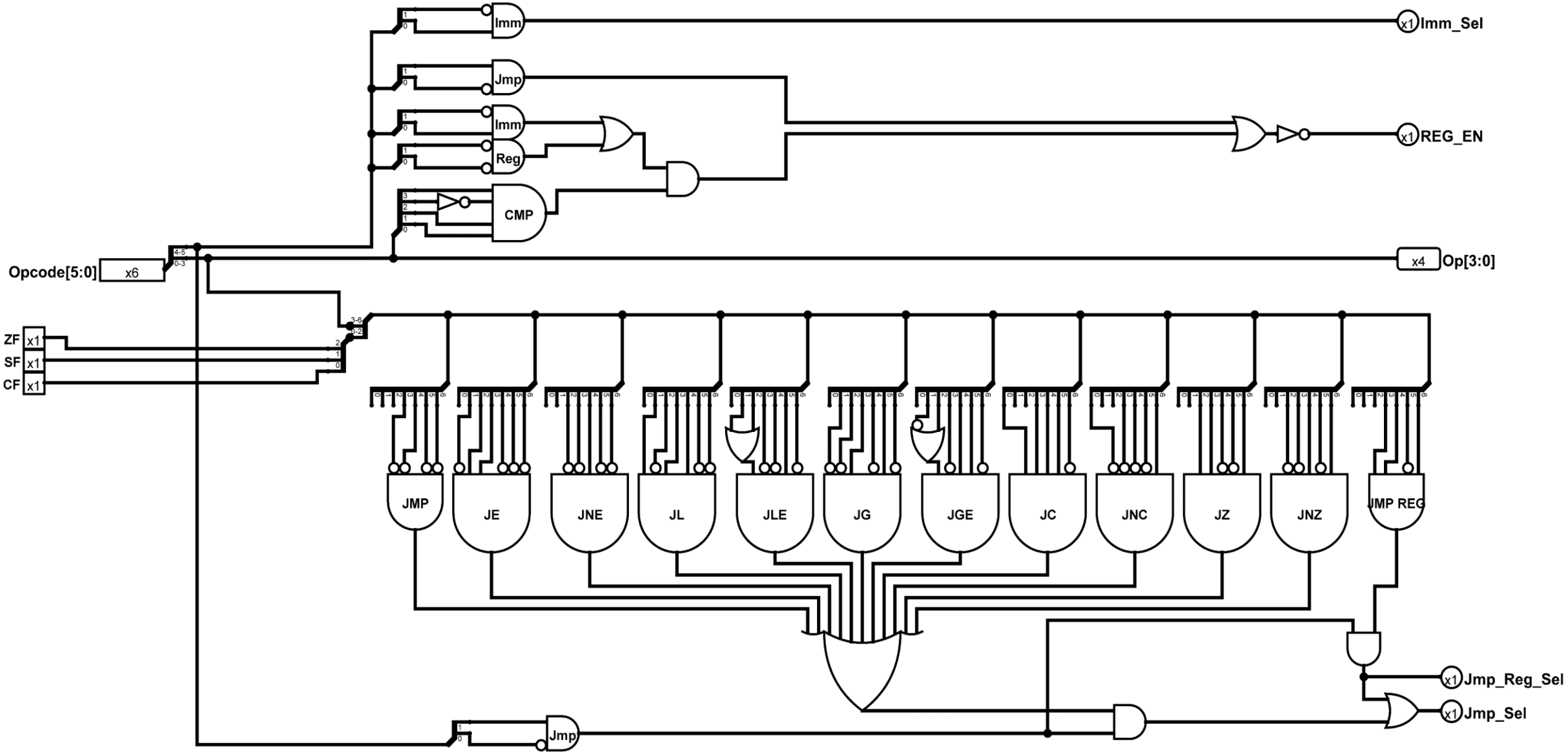


Figure: Control Logic Circuit (Upto Branching)

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

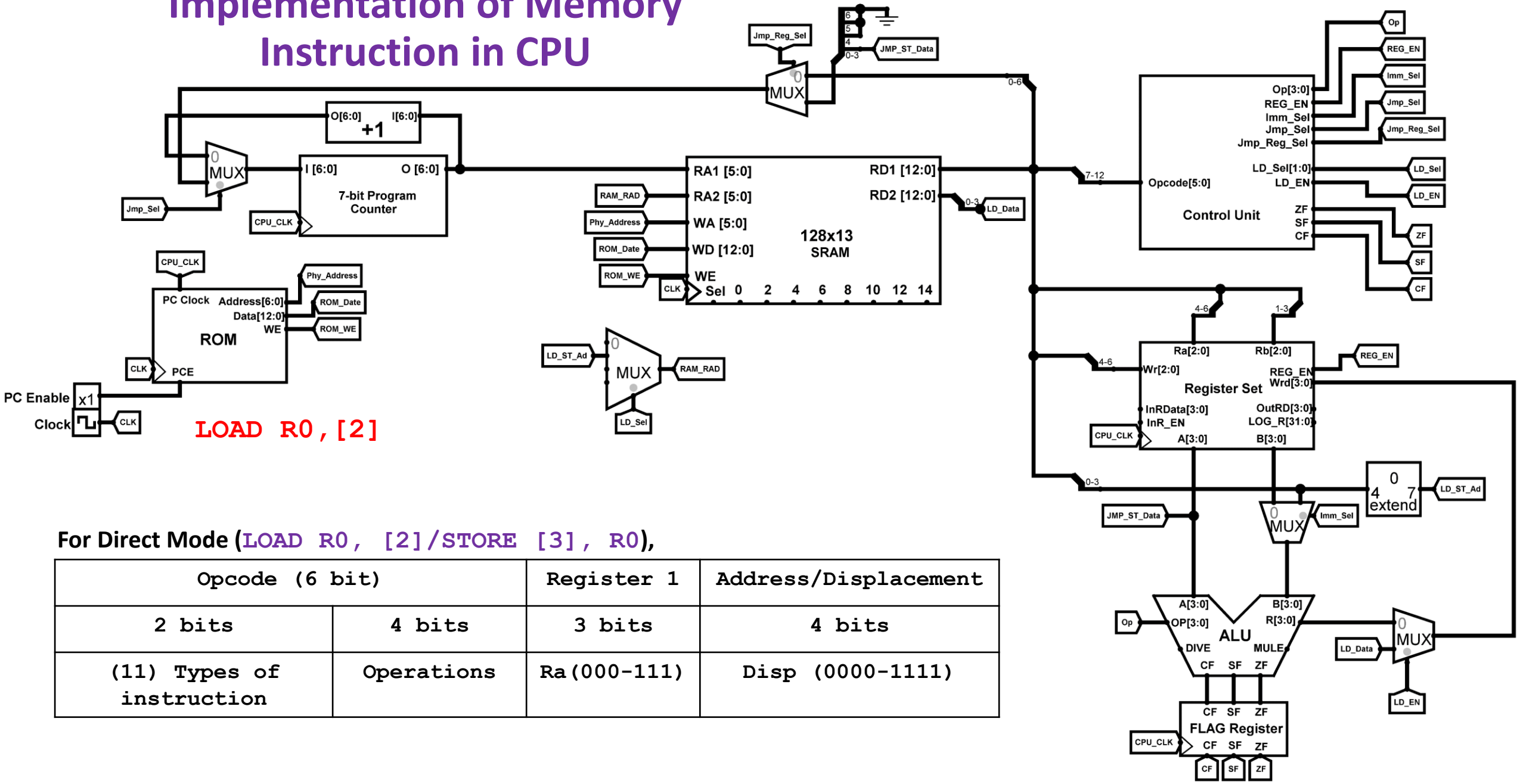
For Based with Displacement Mode (`LOAD R0, [R2+2]/STORE [R3+2], R0`),

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 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

Implementation of Memory Instruction in CPU

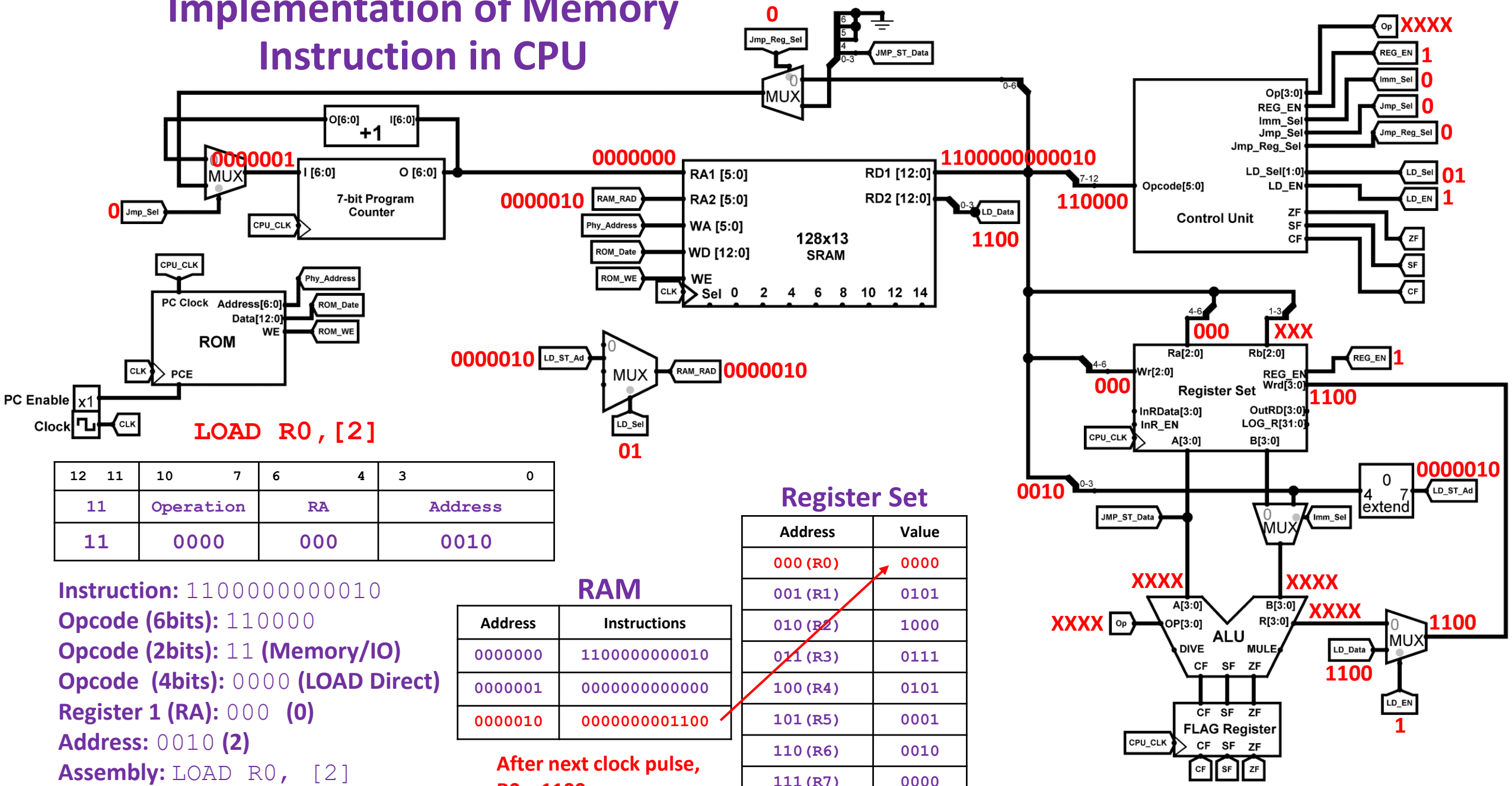


LOAD R0,[2]

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)

Implementation of Memory Instruction in CPU



LOAD R0,[2]

12 11	10 7	6 4	3 0
11	Operation	RA	Address
11	0000	000	0010

Instruction: 1100000000010
Opcode (6bits): 110000
Opcode (2bits): 11 **(Memory/IO)**
Opcode (4bits): 0000 **(LOAD Direct)**
Register 1 (RA): 000 **(0)**
Address: 0010 **(2)**
Assembly: LOAD R0, [2]

RAM

Address	Instructions
0000000	1100000000010
0000001	0000000000000
0000010	0000000001100

After next clock pulse, R0 = 1100

Register Set

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

Control Logic for Memory Instruction in CPU

Control Logic (Truth Table)

	Input					Output						
	Opcode [5:4]	Opcode [3:0]	ZF	SF	CF	Op [3:0]	REG_EN	Imm _sel	Jmp _sel	Jmp _Reg_Sel	LD_Sel [1:0]	LD_EN
Arithmetic & Logic Instructions (Register Mode)	00	AAAA				AAAA	1 **(Except 1011)**	**0**	0	0	**00**	**0**
Arithmetic & Logic Instructions (Immediate Mode)	01	BBBB				BBBB	1 **(Except 1011)**	**1**	0	0	**00**	**0**
Branching Instructions	10	0000 (JMP)	X	X	X	XXXX	0	0	1	0	**00**	**0**
		0001 (JE)	1	0	X							
		0010 (JNE)	0	X	X							
		0011 (JL)	0	1	X							
		0100 (JLE)	sf=1 or zf=1		X							
		0101 (JG)	0	0	X							
		0110 (JGE)	sf=0 or zf=1		X							
		0111 (JC)	X	X	1							
		1000 (JNC)	X	X	0							
		1001 (JZ)	1	X	X							
		1010 (JNZ)	0	X	X							
		1011 (JMPREG)	X	X	X					1	**00**	**0**
Memory & IO Instructions	**11**	**0000 (LOAD Direct)**	**X**	**X**	**X**	**XXXX**	**1**	**0**	**0**	**0**	**01**	**1**

Control Logic for Memory Instruction in CPU

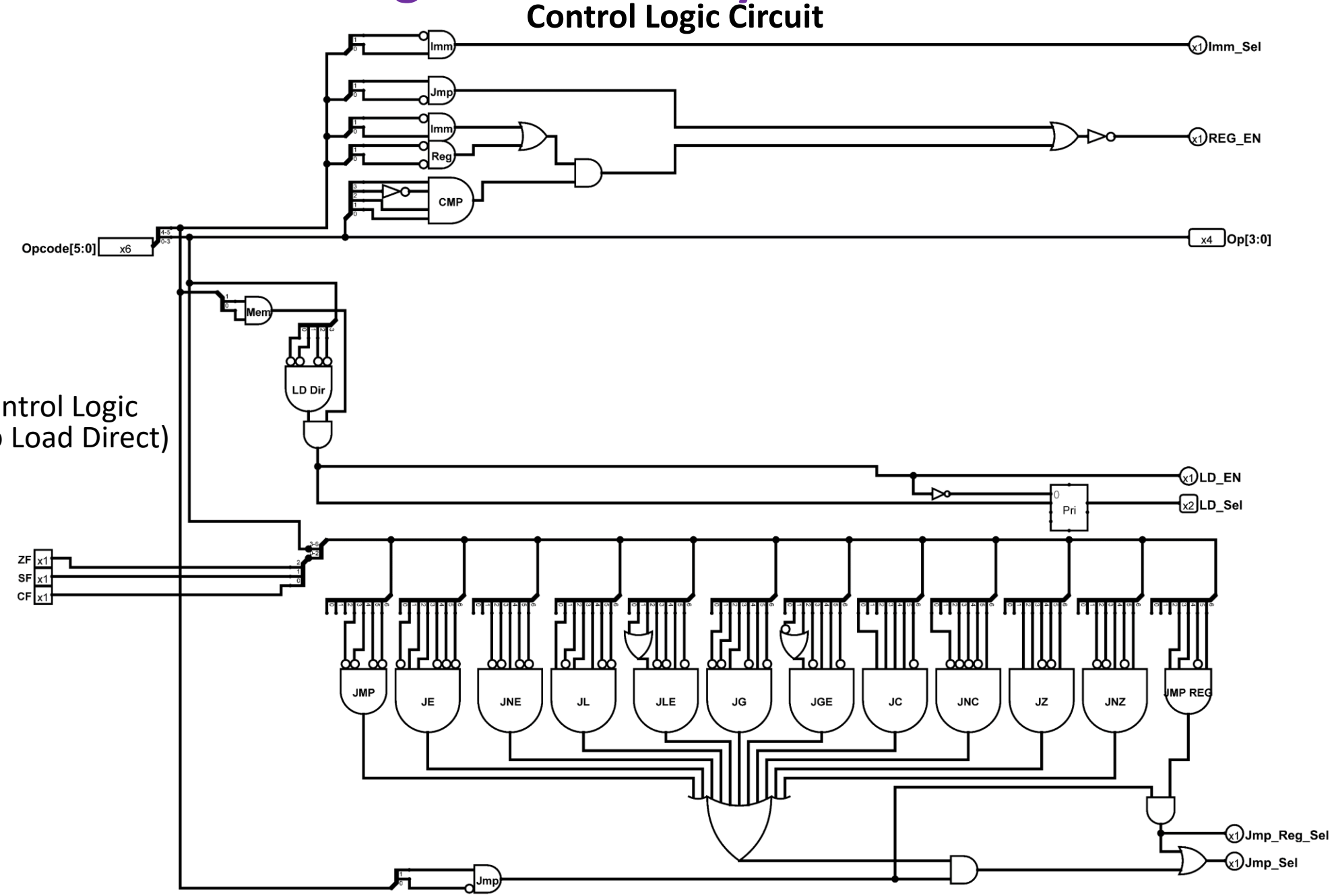
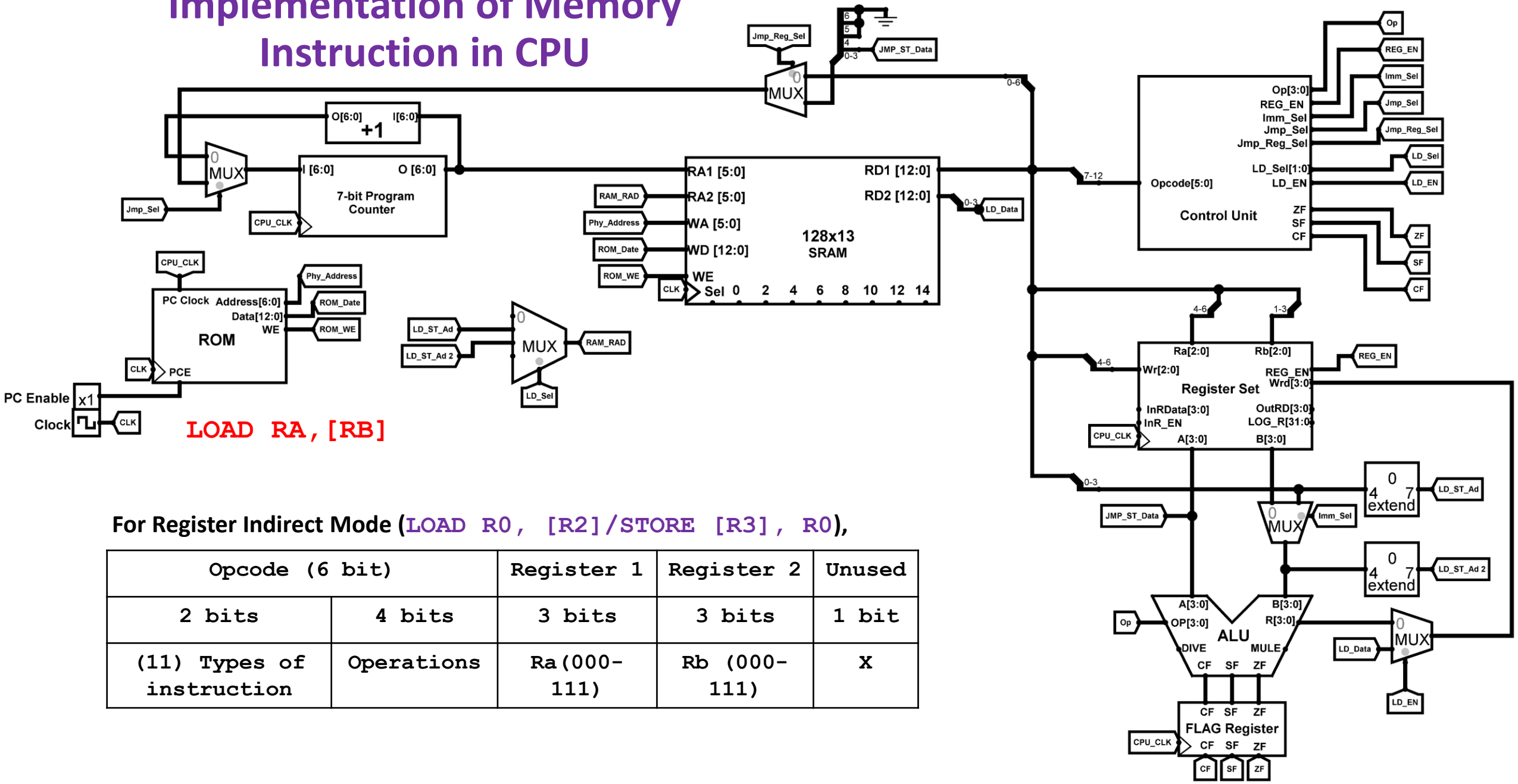


Figure: Control Logic Circuit (Upto Load Direct)

Implementation of Memory Instruction in CPU

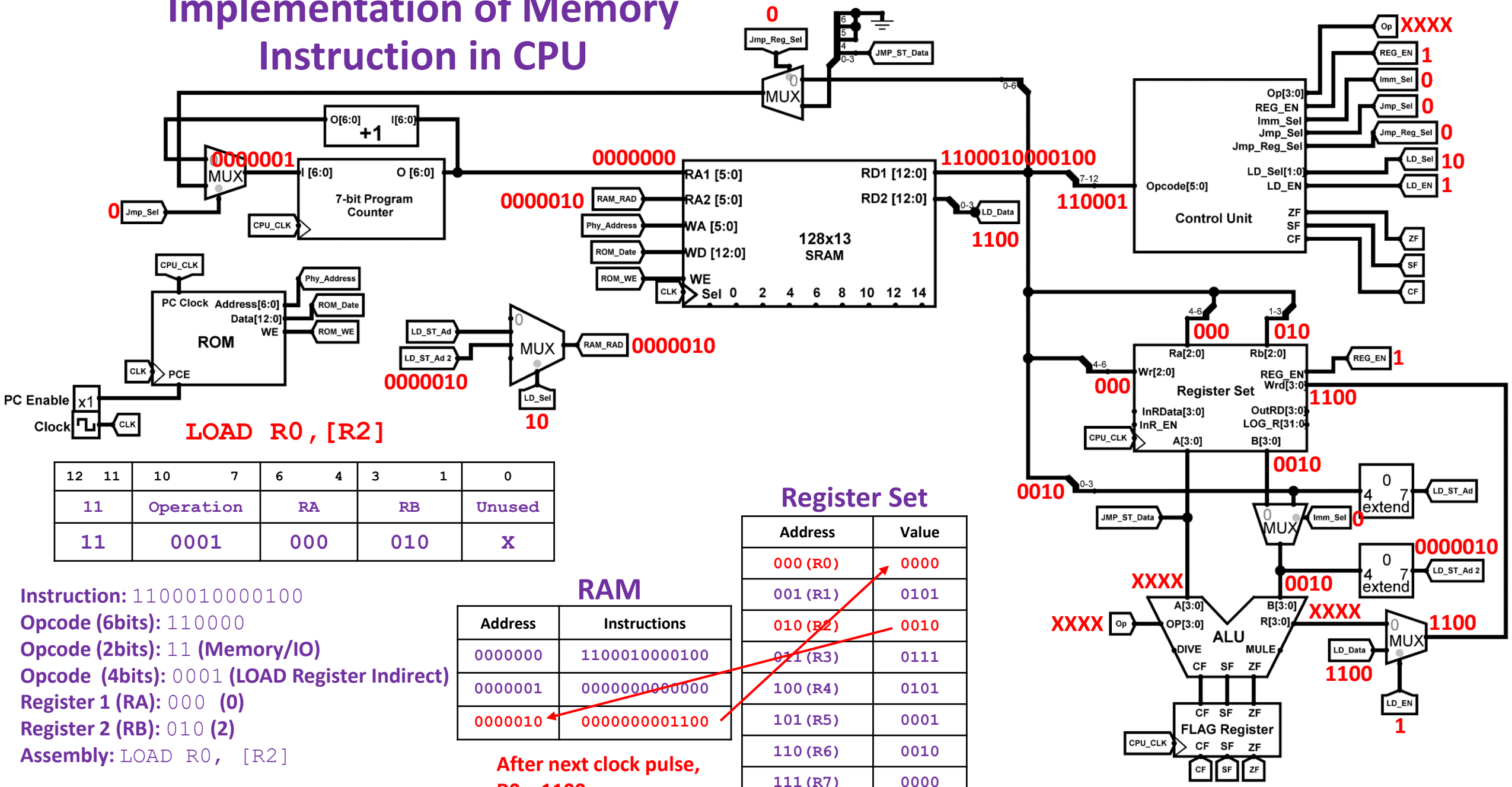


LOAD RA,[RB]

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 bit
(11) Types of instruction	Operations	Ra(000-111)	Rb (000-111)	X

Implementation of Memory Instruction in CPU



LOAD R0,[R2]

12 11	10 7	6 4	3 1	0
11	Operation	RA	RB	Unused
11	0001	000	010	X

Instruction: 1100010000100
Opcode (6bits): 110000
Opcode (2bits): 11 **(Memory/IO)**
Opcode (4bits): 0001 **(LOAD Register Indirect)**
Register 1 (RA): 000 **(0)**
Register 2 (RB): 010 **(2)**
Assembly: LOAD R0, [R2]

RAM

Address	Instructions
0000000	1100010000100
0000001	0000000000000
0000010	0000000001100

After next clock pulse, R0 = 1100

Register Set

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

Control Logic for Memory Instruction in CPU

Control Logic (Truth Table)

	Input					Output						
	Opcode [5:4]	Opcode [3:0]	ZF	SF	CF	Op [3:0]	REG_EN	Imm _sel	Jmp _sel	Jmp _Reg_Sel	LD_Sel [1:0]	LD_EN
Arithmetic & Logic Instructions (Register Mode)	00	AAAA				AAAA	1 **(Except 1011)**	**0**	0	0	**00**	**0**
Arithmetic & Logic Instructions (Immediate Mode)	01	BBBB				BBBB	1 **(Except 1011)**	**1**	0	0	**00**	**0**
Branching Instructions	10	0000 (JMP)	X	X	X	XXXX	0	0	1	0	**00**	**0**
		0001 (JE)	1	0	X							
		0010 (JNE)	0	X	X							
		0011 (JL)	0	1	X							
		0100 (JLE)	sf=1 or zf=1		X							
		0101 (JG)	0	0	X							
		0110 (JGE)	sf=0 or zf=1		X							
		0111 (JC)	X	X	1							
		1000 (JNC)	X	X	0							
		1001 (JZ)	1	X	X							
		1010 (JNZ)	0	X	X							
		1011 (JMPREG)	X	X	X					1	**00**	**0**
Memory & IO Instructions	**11**	0000 (LOAD Direct)	**X**	**X**	**X**	**XXXX**	**1**	**0**	**0**	**0**	**01**	**1**
		0000 (LOAD Register Indirect)									**10**	

Control Logic for Memory Instruction in CPU

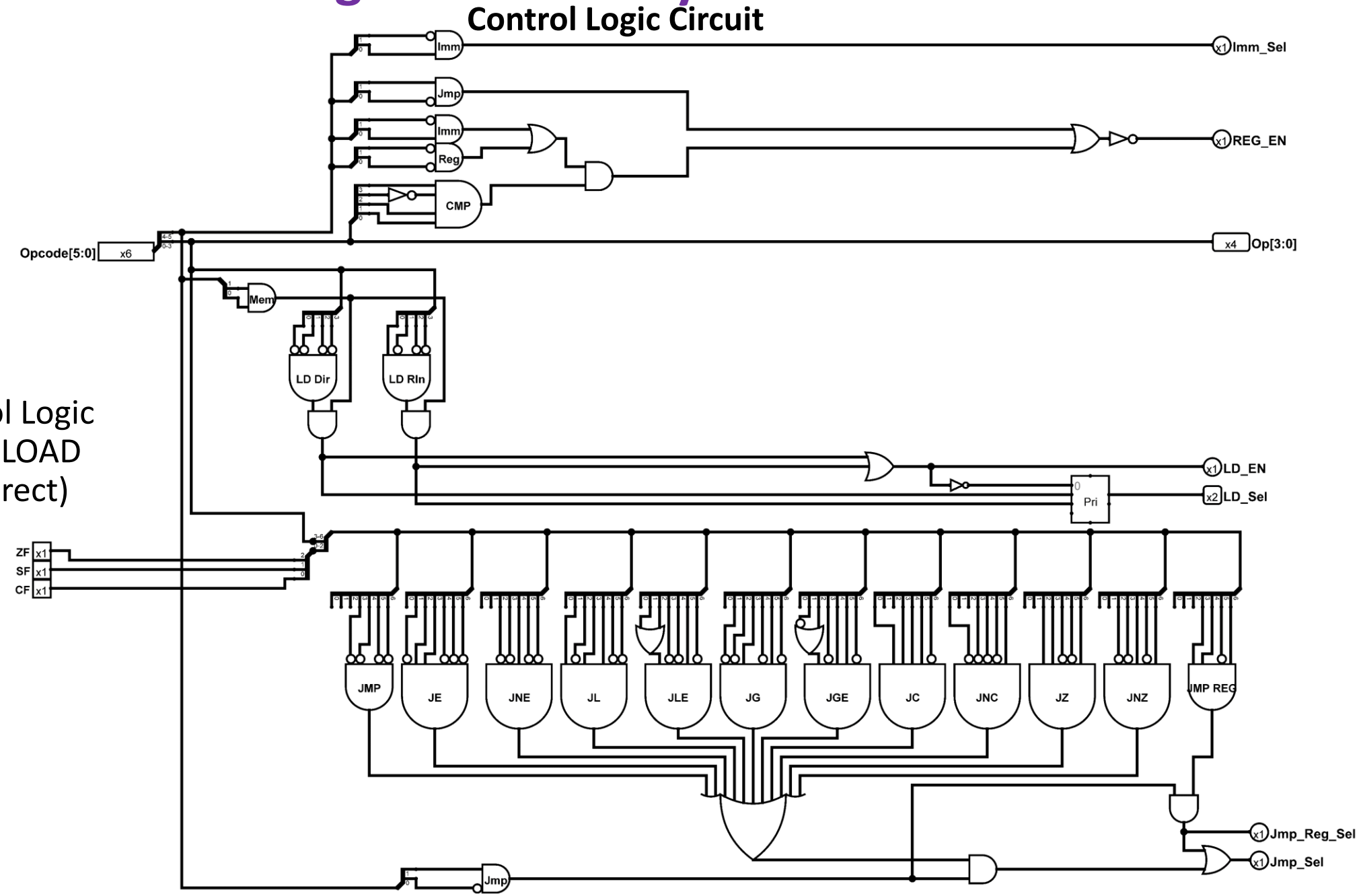
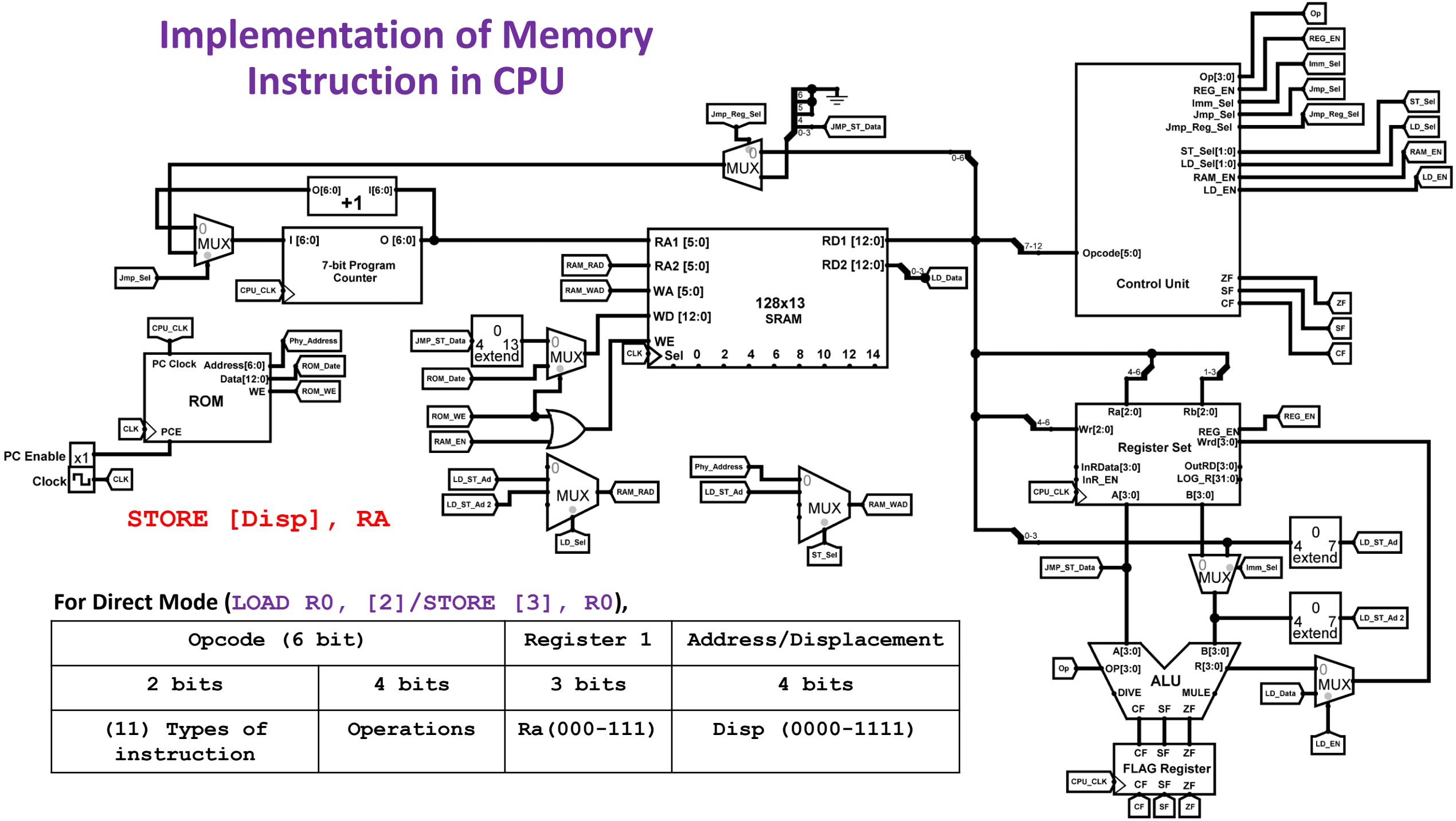


Figure: Control Logic Circuit (Upto LOAD Register Indirect)

Implementation of Memory Instruction in CPU

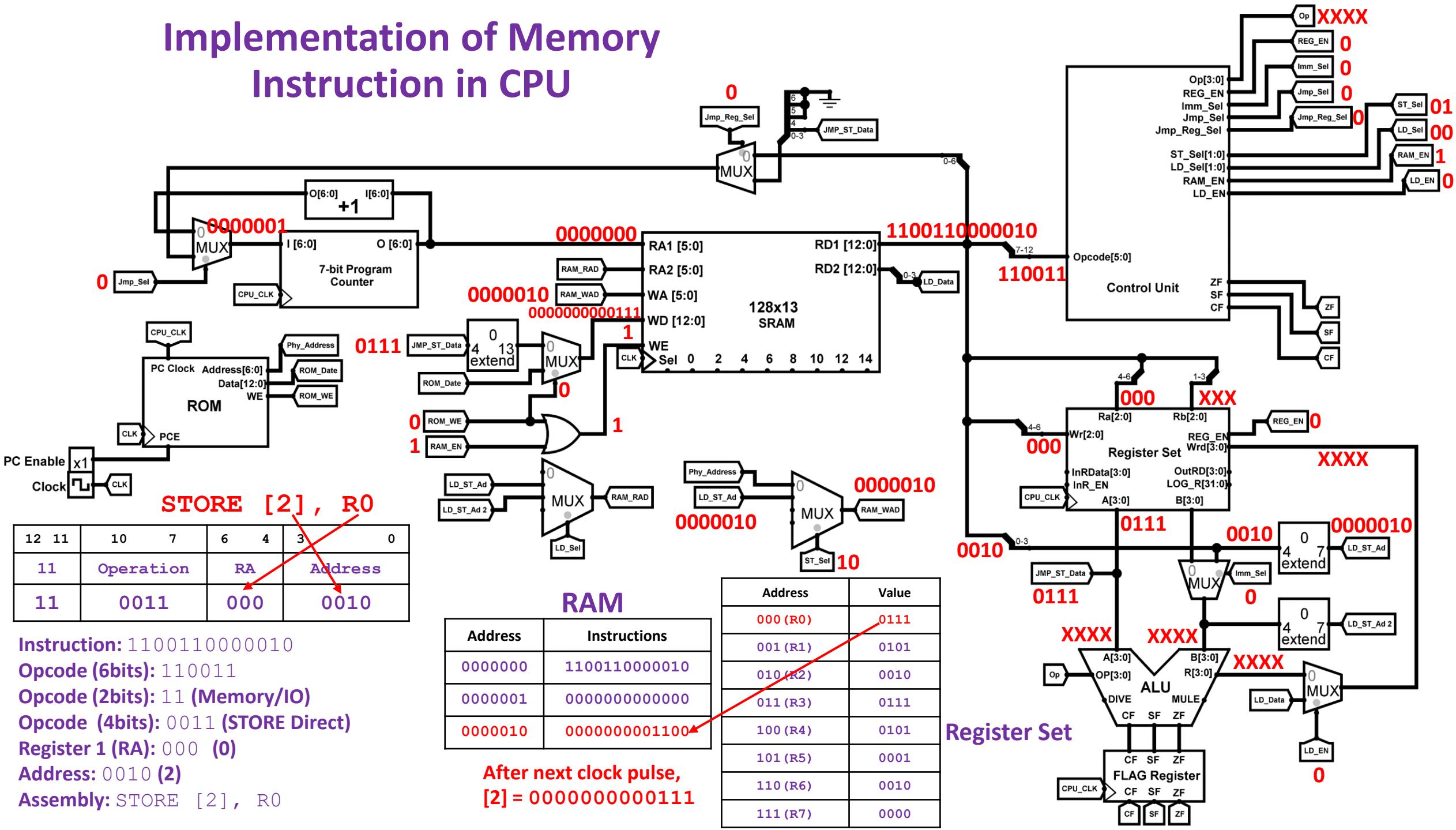


STORE [Disp], RA

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)

Implementation of Memory Instruction in CPU



STORE [2], R0

12 11	10 7	6 4	3 0
11	Operation	RA	Address
11	0011	000	0010

Instruction: 1100110000010
Opcode (6bits): 110011
Opcode (2bits): 11 **(Memory/IO)**
Opcode (4bits): 0011 **(STORE Direct)**
Register 1 (RA): 000 **(0)**
Address: 0010 **(2)**
Assembly: STORE [2], R0

RAM

Address	Instructions
0000000	1100110000010
0000001	0000000000000
0000010	0000000001100

After next clock pulse,
[2] = 0000000000111

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

Control Logic for Memory Instruction in CPU

Control Logic (Truth Table)

	Input					Output								
	Opcode [5:4]	Opcode [3:0]	ZF	SF	CF	Op [3:0]	REG_EN	Imm_sel	Jmp_sel	Jmp_Reg_Sel	LD_Sel [1:0]	LD_EN	ST_Sel [1:0]	RAM_EN
Arithmetic & Logic Instructions (Register Mode)	00	AAAA				AAAA	1 **(Except 1011)**	**0**	0	0	00	0	**00**	**0**
Arithmetic & Logic Instructions (Immediate Mode)	01	BBBB				BBBB	1 **(Except 1011)**	**1**	0	0	00	0	**00**	**0**
Branching Instructions	10	0000 (JMP)	X	X	X	XXXX	0	0	1	0	00	0	**00**	**0**
		0001 (JE)	1	0	X									
		0010 (JNE)	0	X	X									
		0011 (JL)	0	1	X									
		0100 (JLE)	sf=1 or zf=1		X									
		0101 (JG)	0	0	X									
		0110 (JGE)	sf=0 or zf=1		X									
		0111 (JC)	X	X	1									
		1000 (JNC)	X	X	0									
		1001 (JZ)	1	X	X									
		1010 (JNZ)	0	X	X									
		1011 (JMPREG)	X	X	X					1	00	0	**00**	**0**
Memory & IO Instructions	**11**	0000 (LOAD Direct)	**X**	**X**	**X**	**XXXX**	**1**	**0**	**0**	**0**	01	1	**00**	**1**
		0001 (LOAD Register Indirect)									10	1	**00**	**1**
		0100 (STORE Direct)					0				**00**	**0**	**01**	**1**

Control Logic for Memory Instruction in CPU

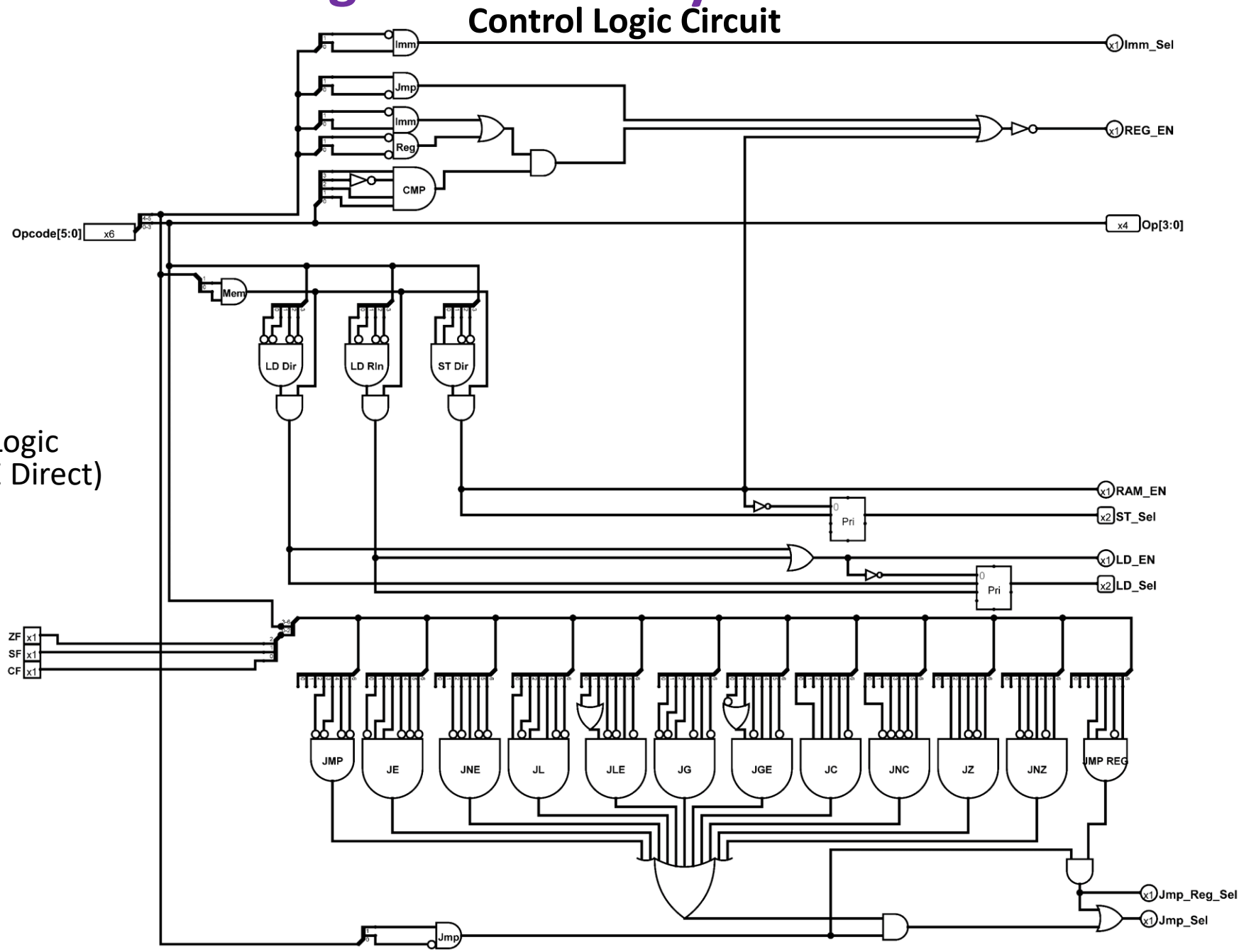
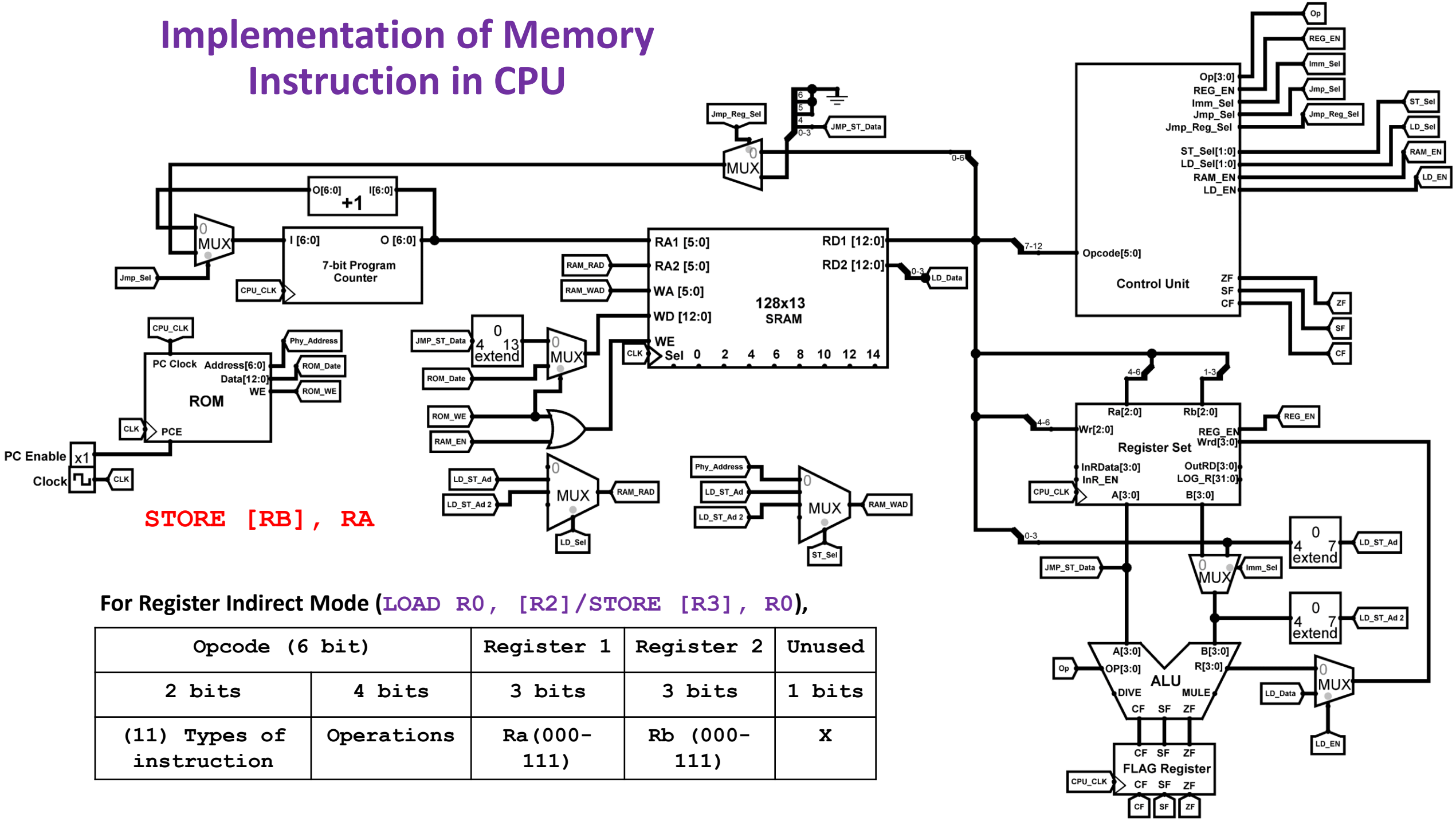


Figure: Control Logic Circuit (Upto STORE Direct)

Implementation of Memory Instruction in CPU

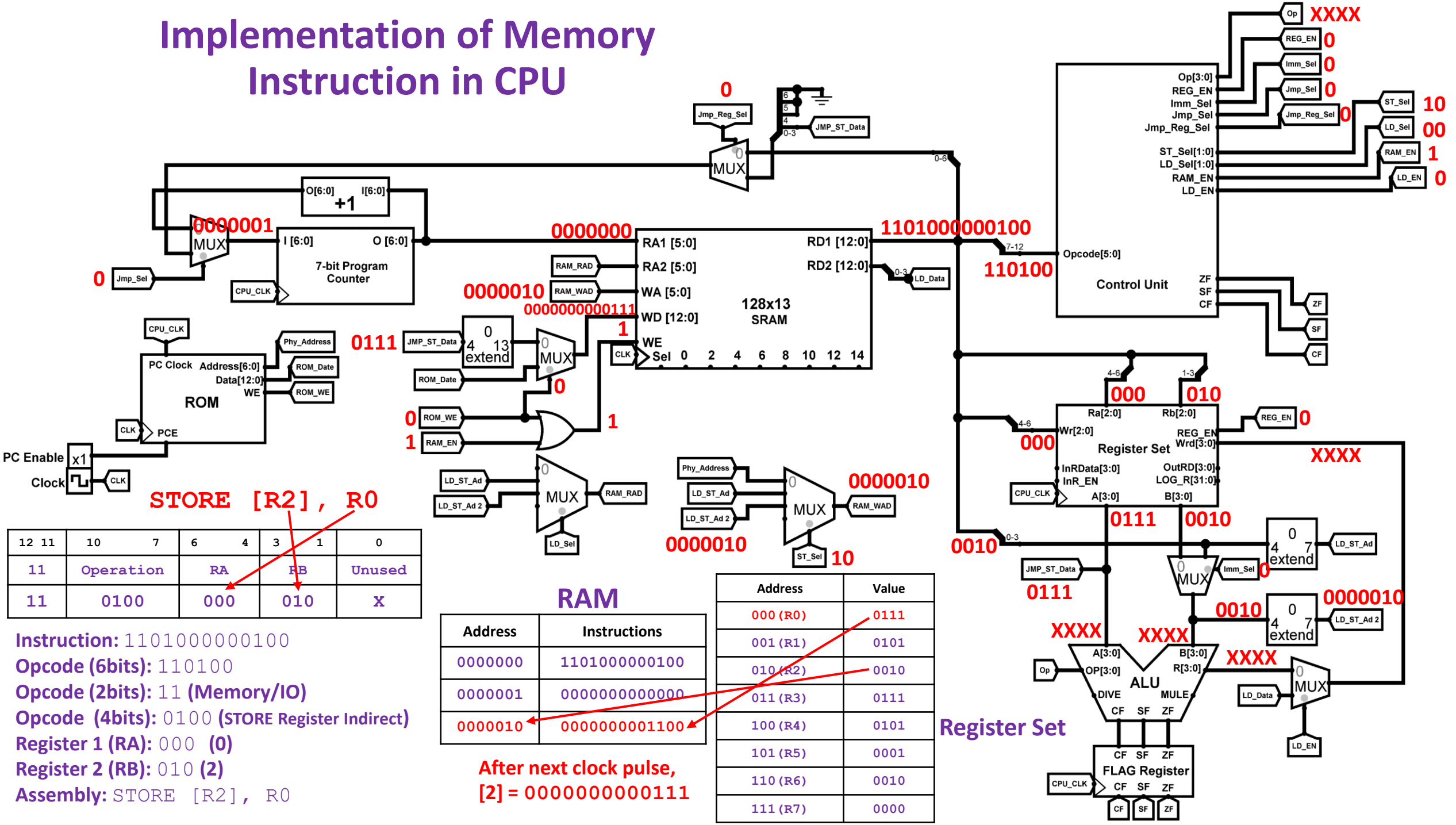


STORE [RB], RA

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

Implementation of Memory Instruction in CPU



STORE [R2], R0

12 11	10 7	6 4	3 1	0
11	Operation	RA	RB	Unused
11	0100	000	010	X

Instruction: 1101000000100
Opcode (6bits): 110100
Opcode (2bits): 11 **(Memory/IO)**
Opcode (4bits): 0100 **(STORE Register Indirect)**
Register 1 (RA): 000 **(0)**
Register 2 (RB): 010 **(2)**
Assembly: STORE [R2], R0

RAM

Address	Instructions
0000000	1101000000100
0000001	0000000000000
0000010	0000000001100

After next clock pulse,
[2] = 0000000000111

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

Register Set

Control Logic for Memory Instruction in CPU

Control Logic (Truth Table)

	Input					Output								
	Opcode [5:4]	Opcode [3:0]	ZF	SF	CF	Op [3:0]	REG_EN	Imm _sel	Jmp _sel	Jmp _Reg_Sel	LD_Sel [1:0]	LD_EN	ST_Sel [1:0]	RAM_EN
Arithmetic & Logic Instructions (Register Mode)	00	AAAA				AAAA	1 **(Except 1011)**	**0**	0	0	00	0	**00**	**0**
Arithmetic & Logic Instructions (Immediate Mode)	01	BBBB				BBBB	1 **(Except 1011)**	**1**	0	0	00	0	**00**	**0**
Branching Instructions	10	0000 (JMP)	X	X	X	XXXX	0	0	1	0	00	0	**00**	**0**
		0001 (JE)	1	0	X									
		0010 (JNE)	0	X	X									
		0011 (JL)	0	1	X									
		0100 (JLE)	sf=1 or zf=1		X									
		0101 (JG)	0	0	X									
		0110 (JGE)	sf=0 or zf=1		X									
		0111 (JC)	X	X	1									
		1000 (JNC)	X	X	0									
		1001 (JZ)	1	X	X									
		1010 (JNZ)	0	X	X									
		1011 (JMPREG)	X	X	X					1	00	0	**00**	**0**
Memory & IO Instructions	**11**	0000 (LOAD Direct)	**X**	**X**	**X**	**XXXX**	**1**	**0**	**0**	**0**	01	1	**00**	**1**
		0001 (LOAD Register Indirect)									10	1	**00**	**0**
		0100 (STORE Direct)									00	0	**01**	**1**
		0101 (STORE Register Indirect)					0				**00**	**0**	**10**	**1**

Control Logic for Memory Instruction in CPU

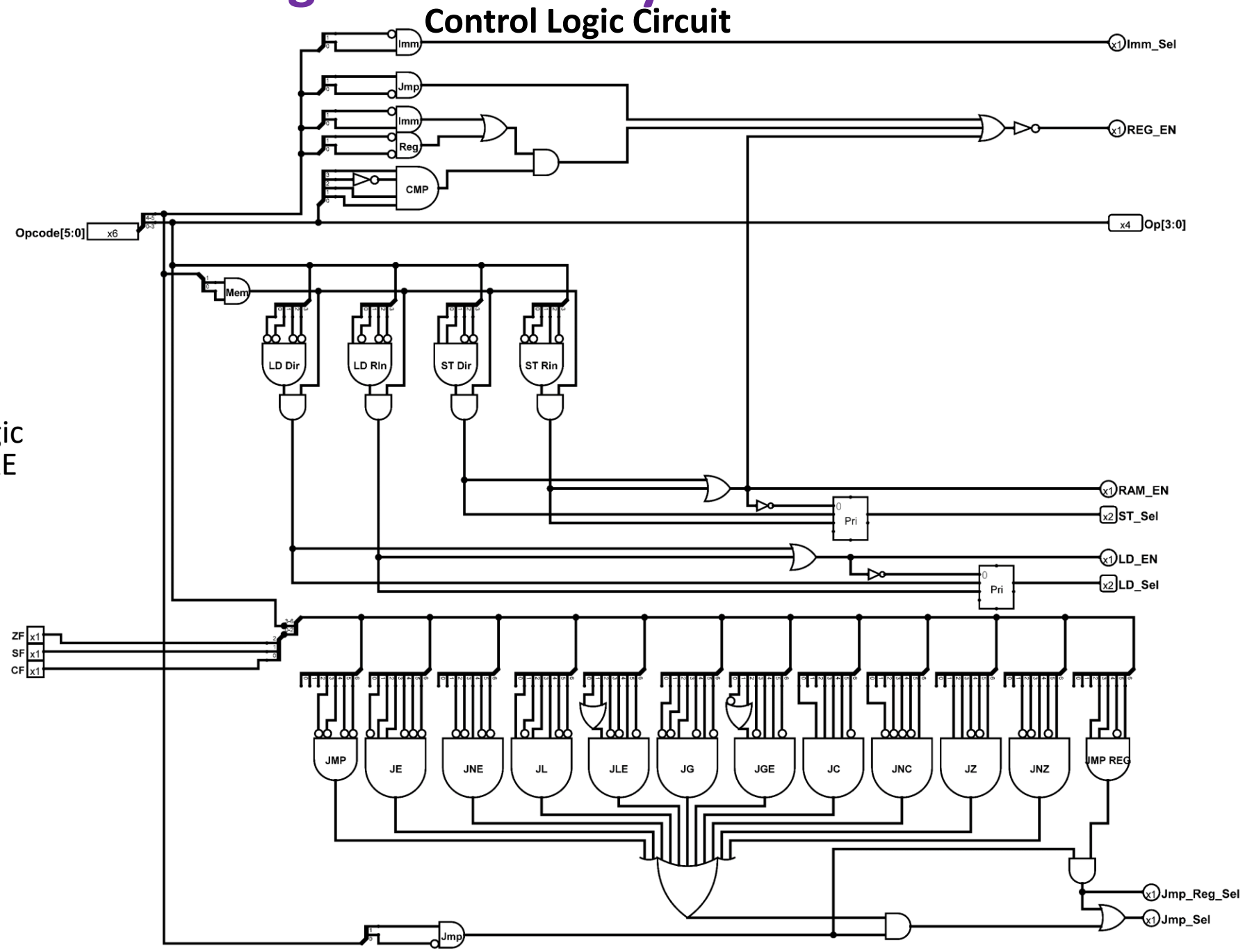
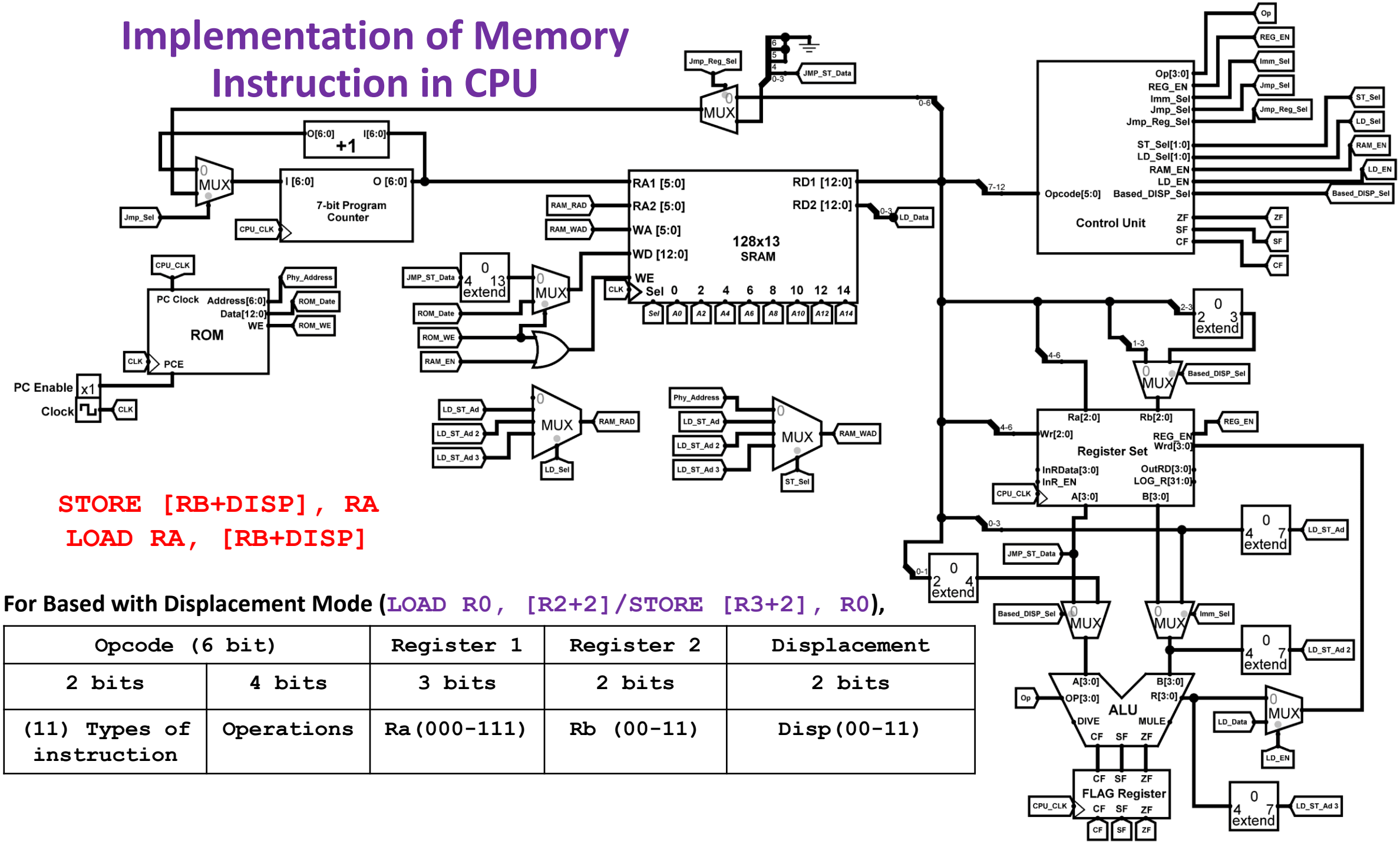


Figure: Control Logic Circuit (Upto STORE Register Indirect)

Implementation of Memory Instruction in CPU



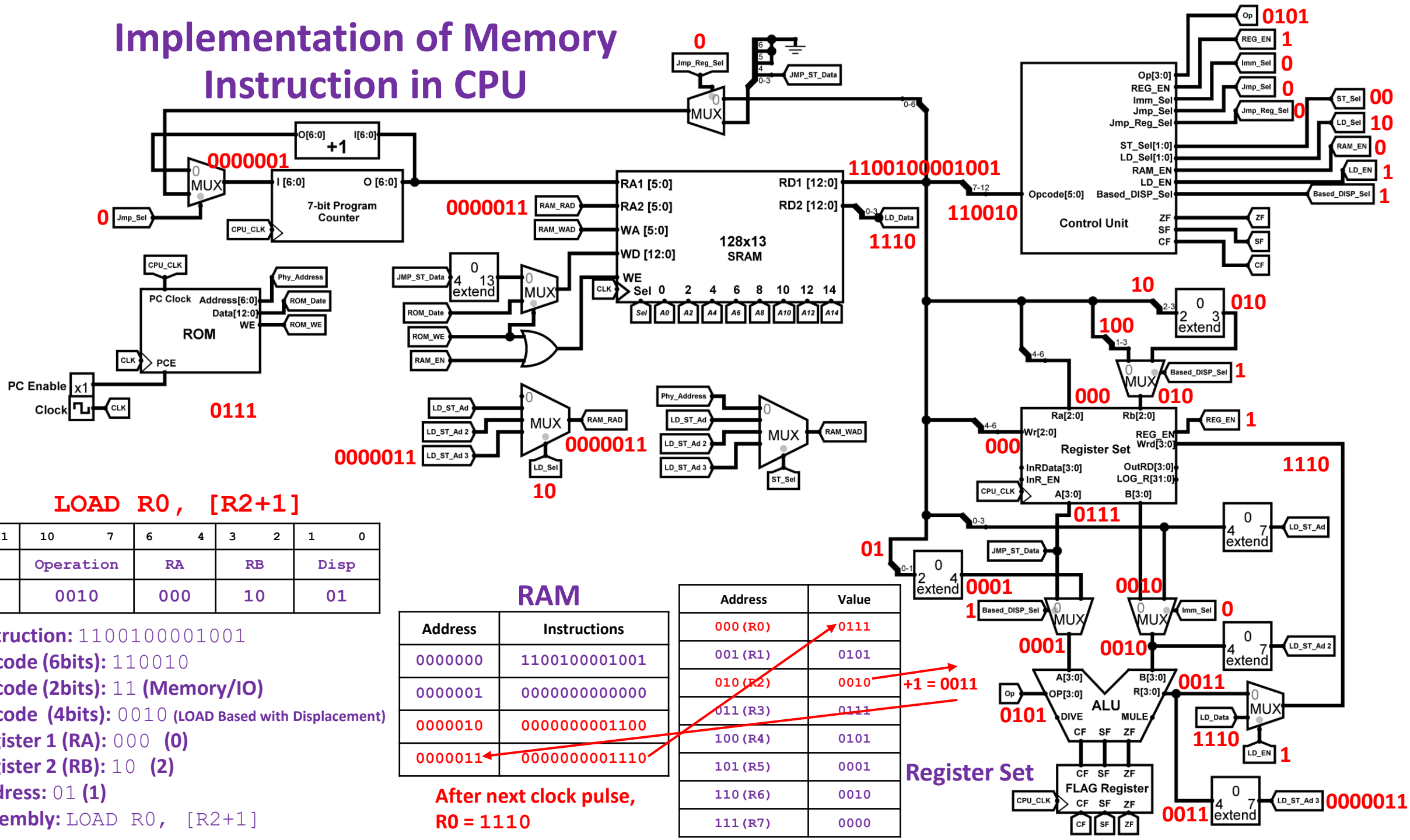
STORE [RB+DISP], RA

LOAD RA, [RB+DISP]

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)

Implementation of Memory Instruction in CPU



LOAD R0, [R2+1]

12 11	10 7	6 4	3 2	1 0
11	Operation	RA	RB	Disp
11	0010	000	10	01

Instruction: 1100100001001
Opcode (6bits): 110010
Opcode (2bits): 11 **(Memory/IO)**
Opcode (4bits): 0010 **(LOAD Based with Displacement)**
Register 1 (RA): 000 **(0)**
Register 2 (RB): 10 **(2)**
Address: 01 **(1)**
Assembly: LOAD R0, [R2+1]

RAM

Address	Instructions
0000000	1100100001001
0000001	0000000000000
0000010	0000000001100
0000011	0000000001110

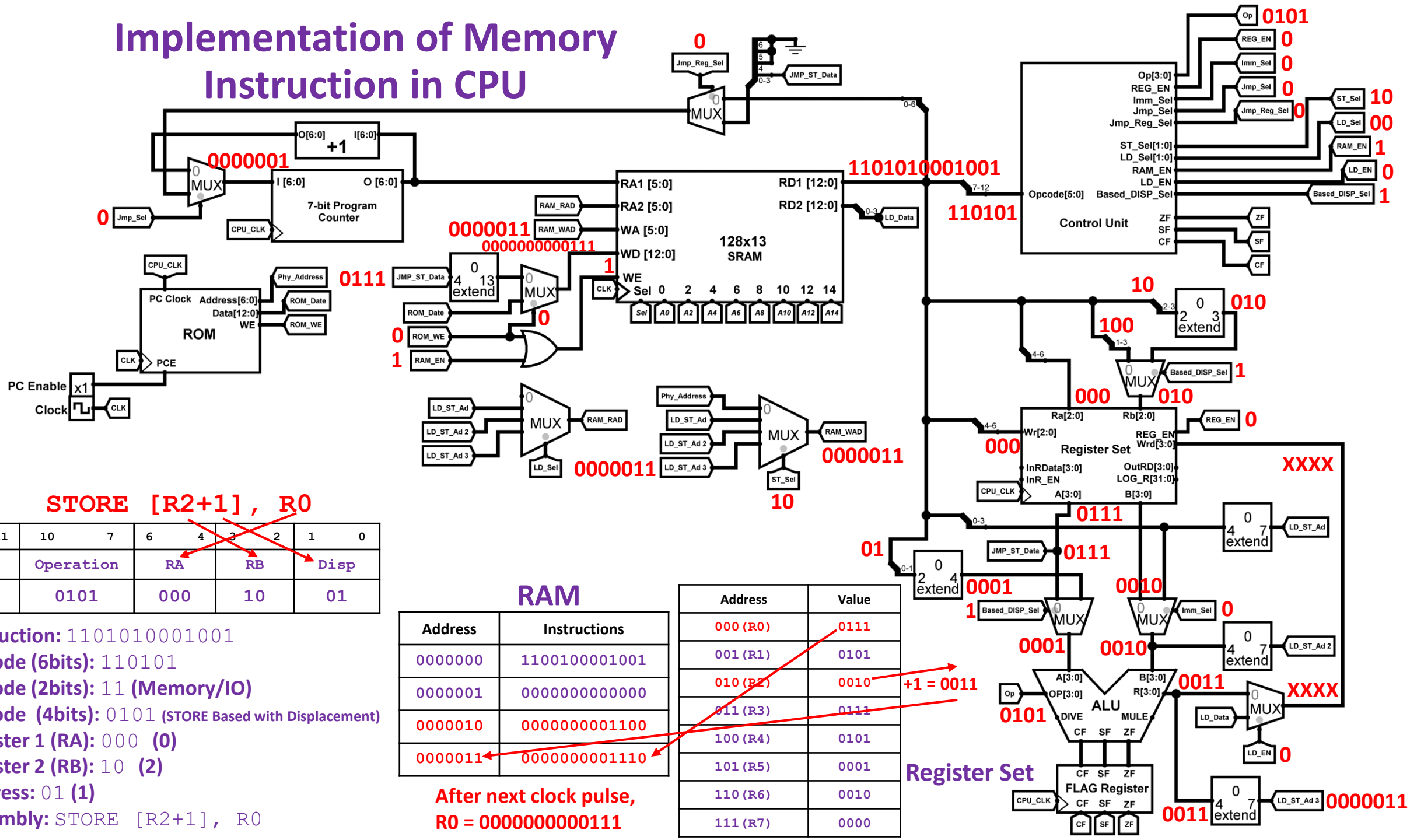
After next clock pulse,
R0 = 1110

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

+1 = 0011

Register Set

Implementation of Memory Instruction in CPU



STORE [R2+1], R0

12 11	10 7	6 4	3 2	1 0
11	Operation	RA	RB	Disp
11	0101	000	10	01

Instruction: 1101010001001
Opcode (6bits): 110101
Opcode (2bits): 11 **(Memory/IO)**
Opcode (4bits): 0101 **(STORE Based with Displacement)**
Register 1 (RA): 000 **(0)**
Register 2 (RB): 10 **(2)**
Address: 01 **(1)**
Assembly: STORE [R2+1], R0

RAM

Address	Instructions
0000000	1100100001001
0000001	0000000000000
0000010	0000000001100
0000011	0000000001110

After next clock pulse, R0 = 0000000000111

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

+1 = 0011

Register Set

Control Logic for Memory Instruction in CPU

Control Logic (Truth Table)

	Input					Output									
	Opcode [5:4]	Opcode [3:0]	ZF	SF	CF	Op [3:0]	REG_EN	Imm _sel	Jmp _sel	Jmp _Reg_Sel	LD_Sel [1:0]	LD_EN	ST_Sel [1:0]	RAM_ EN	Based_ DISP_Sel
Arithmetic & Logic Instructions (Register Mode)	00	AAAA				AAAA	1 **(Except 1011)**	**0**	0	0	00	0	00	0	**0**
Arithmetic & Logic Instructions (Immediate Mode)	01	BBBB				BBBB	1 **(Except 1011)**	**1**	0	0	00	0	00	0	**0**
Branching Instructions	10	0000 (JMP)	X	X	X	XXXX	0	0	1	0	00	0	00	0	**0**
		0001 (JE)	1	0	X										
		0010 (JNE)	0	X	X										
		0011 (JL)	0	1	X										
		0100 (JLE)	sf=1 or zf=1		X										
		0101 (JG)	0	0	X										
		0110 (JGE)	sf=0 or zf=1		X										
		0111 (JC)	X	X	1										
		1000 (JNC)	X	X	0										
		1001 (JZ)	1	X	X										
		1010 (JNZ)	0	X	X										
		1011 (JMPREG)	X	X	X					1	00	0	00	0	**0**
Memory & IO Instructions	**11**	0000 (LOAD Direct)	**X**	**X**	**X**	**XXXX**	**1**	**0**	**0**	**0**	01	1	00	0	**0**
		0001 (LOAD Register Indirect)									10	1	00	0	**0**
		0010 (LOAD Based with Displacement)									**11**	**1**	**00**	**0**	**1**
		0100 (STORE Direct)					**0**				00	0	01	1	**0**
		0101 (STORE Register Indirect)									00	0	10	1	**0**
		0110 (STORE Based with Displacement)									**00**	**0**	**11**	**1**	**1**

Control Logic Circuit

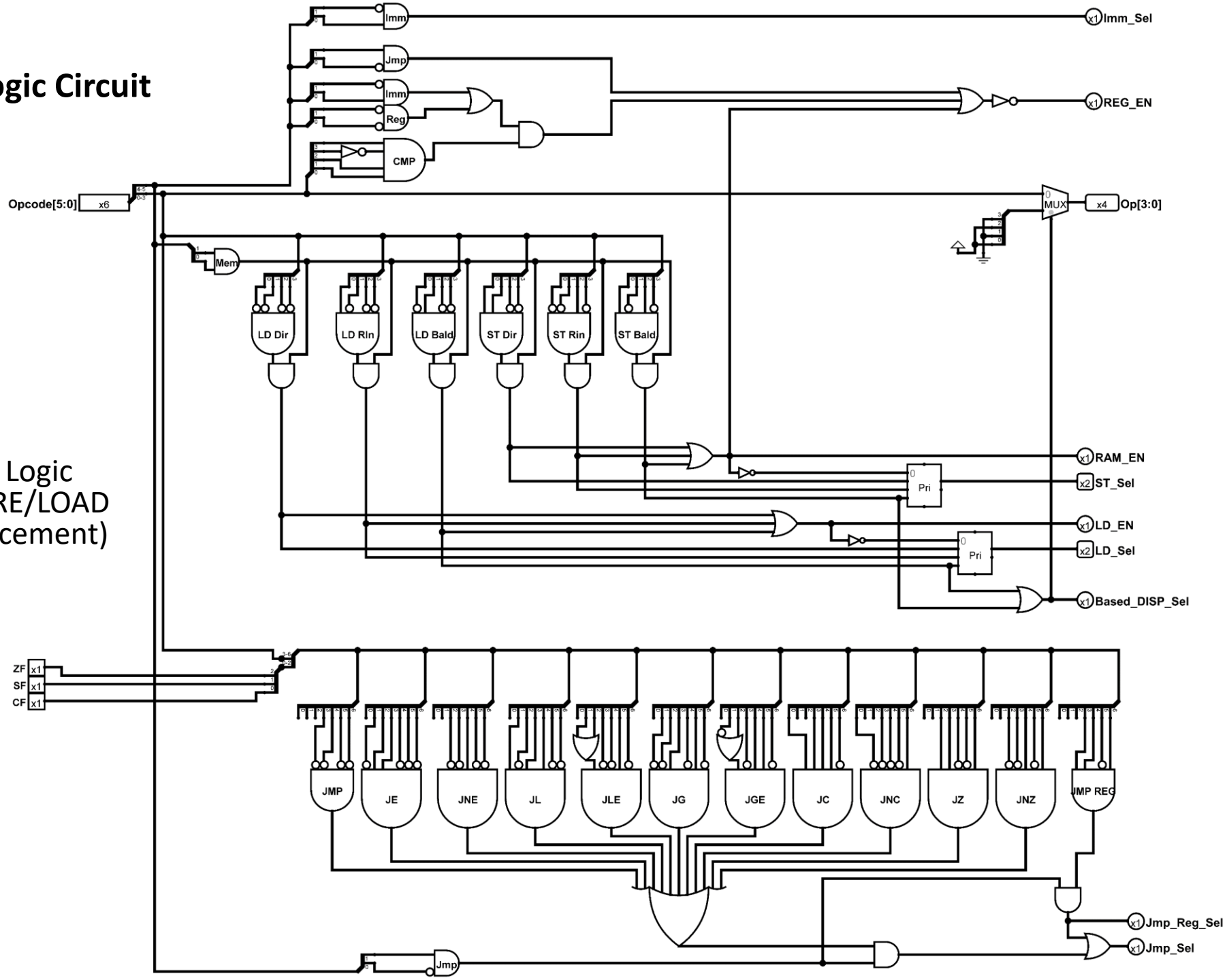


Figure: Control Logic Circuit (Upto STORE/LOAD Based with Displacement)

Example: CPU Design

Question:

Draw a 16-bit CPU with 32 registers which supports following instructions:

i. ALU Instruction (Immediate Mode): XOR RA, Value
ii. Jump Instruction: JL LABEL
iii. Memory Instruction: LOAD RA, [Address]

Show ISA and Control Unit of CPU.

Answer:

Suppose, supported RAM is 256x32.

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 (XOR=1111)	Ra(00000-11111)	Value (0000000000000000-1111111111111111)	XXXXX

Example: CPU Design

ISA format of Jump Instruction is:

Opcode (6 bit)		Address	Unused
2 bits	4 bits	8 bits	18 bits
(01) Types of instruction	Operations (JL=1010)	Value (00000000-11111111)	XXXXXXXXXXXXXXXXXX

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

Opcode (6 bit)		Register 1	Address	Unused
2 bits	4 bits	5 bits	8 bits	13 bits
(10) Types of instruction	Operations (LOAD Dir = 1100)	RA (00000-11111)	Value (00000000-11111111)	XXXXXXXXXXXXX

Example: CPU Design

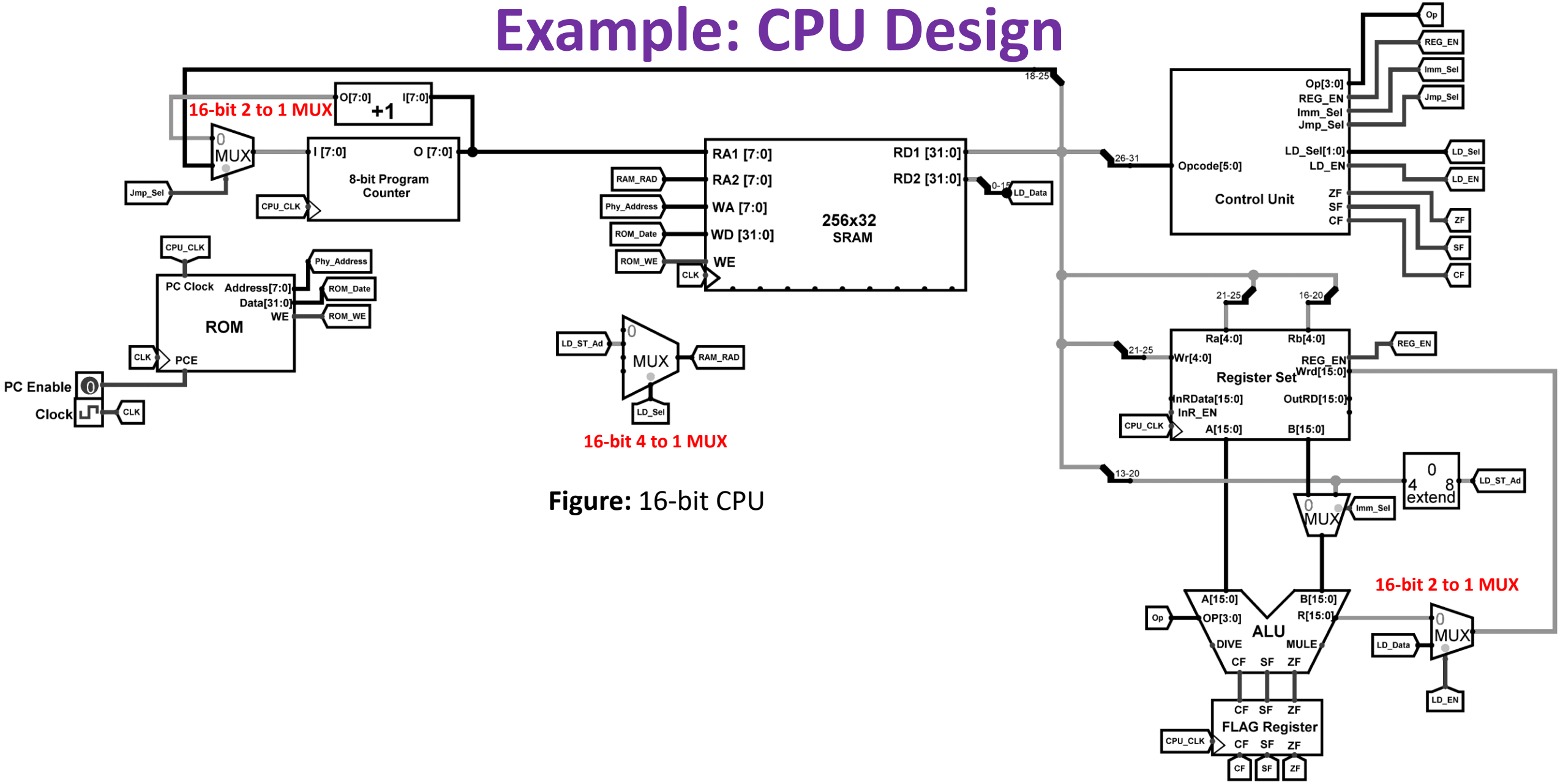


Figure: 16-bit CPU

Example: CPU Design

Control Logic (Truth Table)

	Input					Output					
	Opcode [5:4]	Opcode [3:0]	ZF	SF	CF	Op [3:0]	REG_EN	Imm_sel	Jmp_sel	LD_Sel [1:0]	LD_EN
Arithmetic & Logic Instructions (Immediate Mode)	00	1111 (XOR)				1111	1	1	0	00	0
Branching Instructions	01	1010 (JL)	0	1	X	XXXX	0	0	1	00	0
Memory & IO Instructions	10	1100 (LOAD Direct)	X	X	X	XXXX	1	0	0	01	1

Example: CPU Design

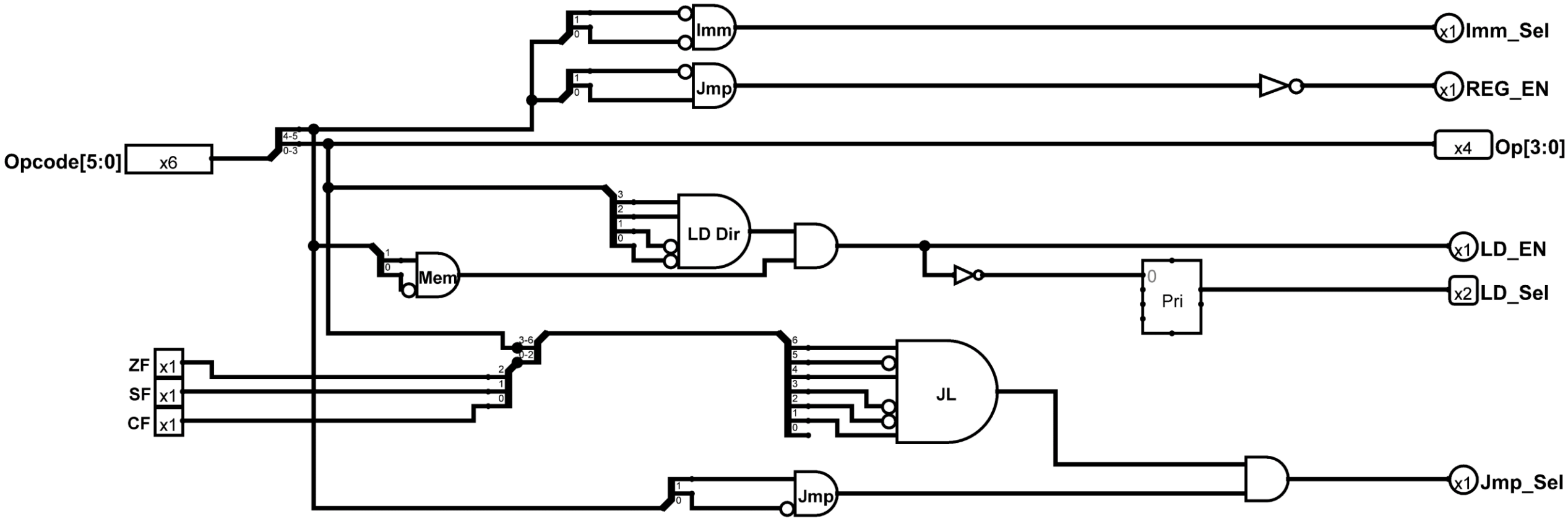


Figure: Control Logic Circuit

Exercises

1. Draw a 8-bit CPU with 16 registers which supports following instructions:
   i. ALU Instruction (Register Mode): ADD RA, RB
   ii. ALU Instruction (Immediate Mode): AND RA, Value
   iii. Jump Instruction: JL LABEL

   Show ISA and Control Unit of CPU.

2. Implement a 32-bit CPU with 32 registers which supports following instructions:
   i. Jump Instruction: JMPREG LABEL
   ii. Memory Instruction: STORE [RA+2], RC

   Show ISA and Control Unit of CPU.

3. Suppose following instructions are stored inside 128x32 RAM:

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

Design a 10-bit CPU with 6 Registers which can execute all the instructions stored in RAM.
Show ISA and Control Unit of CPU. Show how CPU will execute these instructions step by step.

Thank you ☺