Interfacing

Nahin Ul Sadad
Lecturer
CSE, RUET

IO Instructions

IO Operations

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

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**

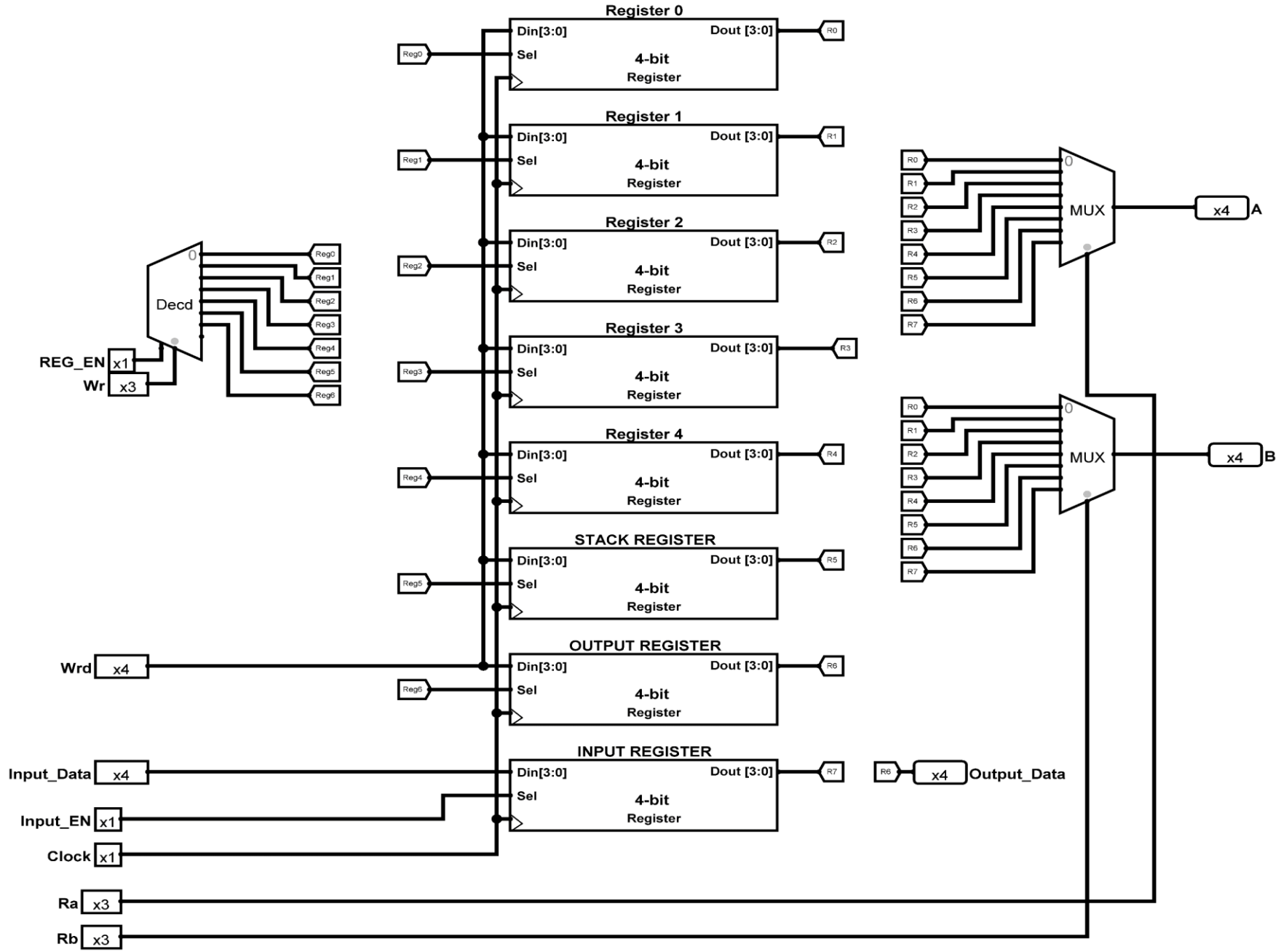
I/O Registers

There are two I/O registers:

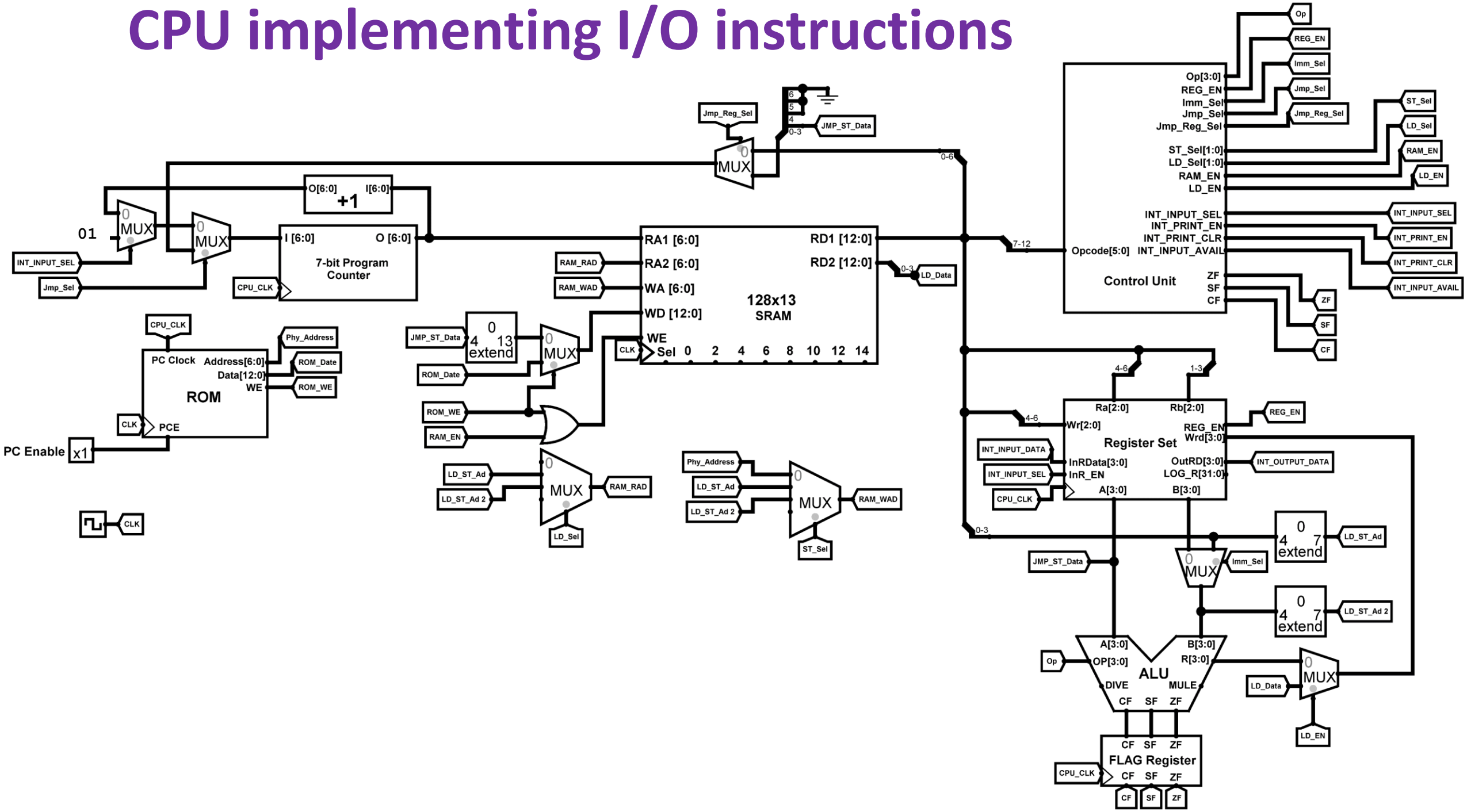
1. **Input Register/Input Port:** This register will be used to store all data from outside CPU.

2. **Output Register/Output Port:** This register will be used to send all data to outside of CPU.

4-bit ALU

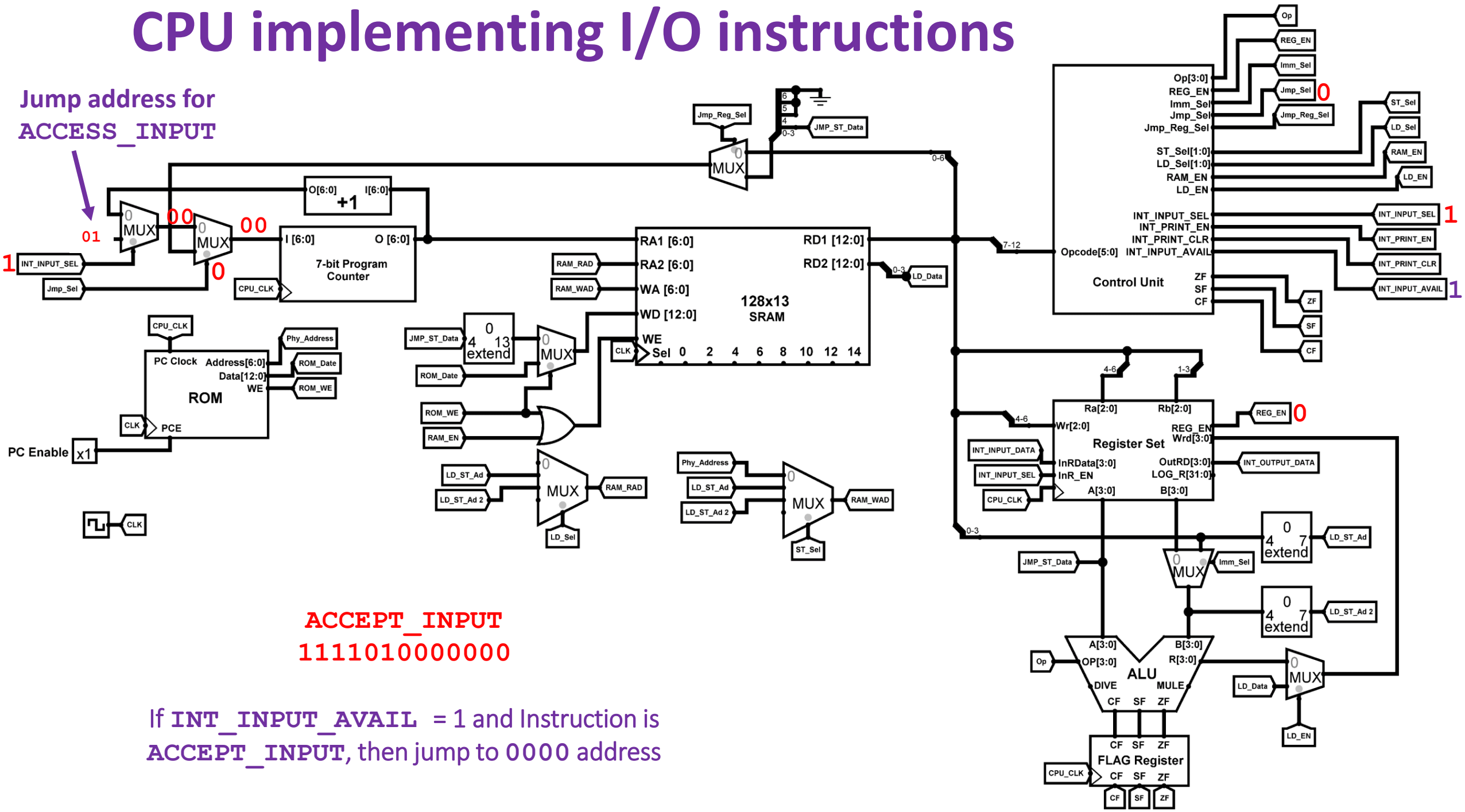

Register 0
Register 1
Register 2
Register 3
Register 4
STACK REGISTER
OUTPUT REGISTER
INPUT REGISTER
Din[3:0]
Dout [3:0]
Sel
4-bit
Register
Decd
REG_EN
Wr
x1
x3
MUX
x4
A
B
Wrd
Input_Data
Input_EN
Clock
Ra
Rb
Output_Data

CPU implementing I/O instructions



CPU implementing I/O instructions

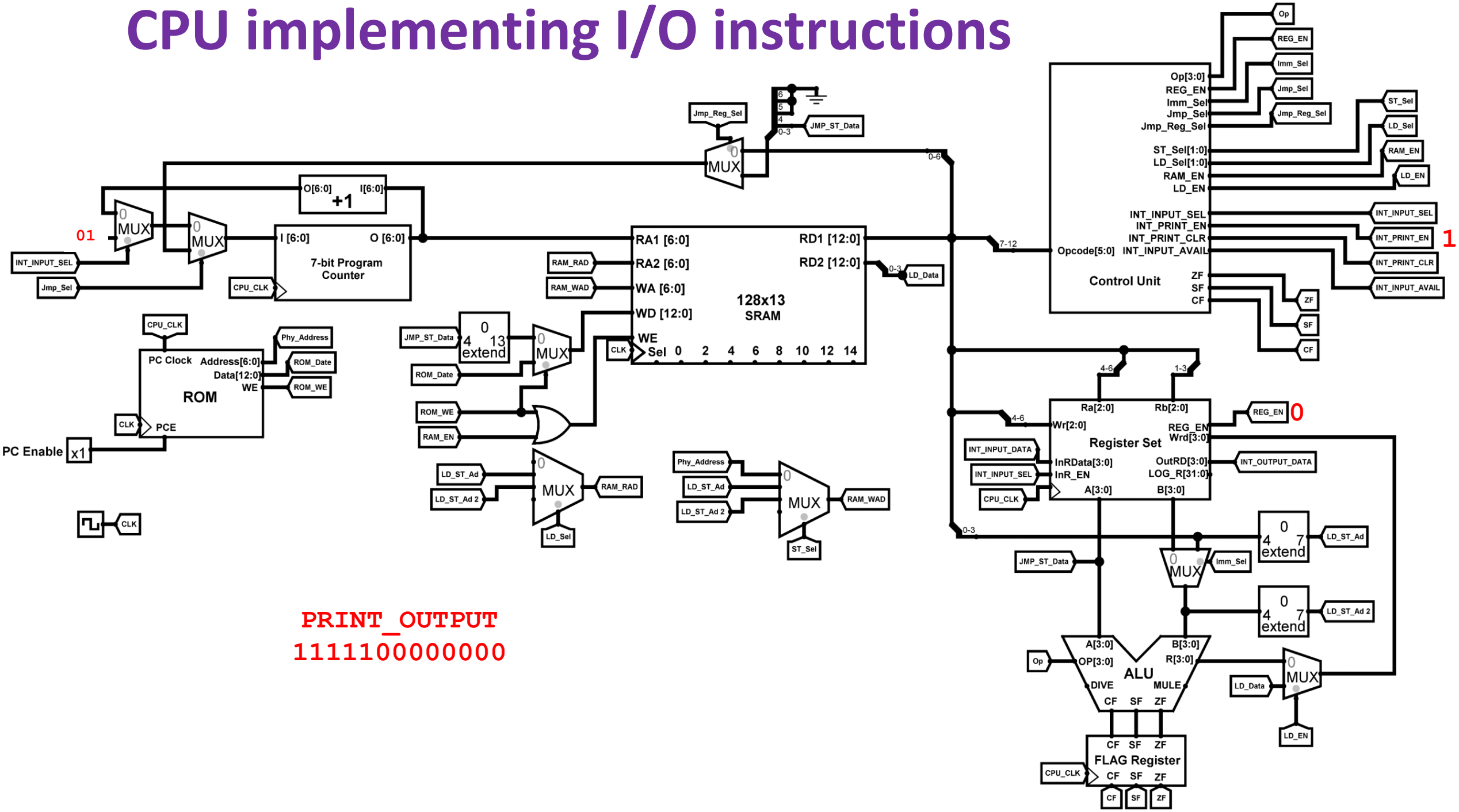
Jump address for
ACCESS_INPUT



ACCEPT_INPUT
1111010000000

If **INT_INPUT_AVAIL** = 1 and Instruction is **ACCEPT_INPUT**, then jump to 0000 address

CPU implementing I/O instructions



PRINT_OUTPUT
1111100000000

Control Logic for Memory Instruction in CPU

Control Logic (Truth Table)

	Input						Output											
	Opcode [5:4]	Opcode [3:0]	ZF	SF	CF	INT_INPUT_AVAIL	Op [3:0]	REG_EN	Imm_sel	Jmp_sel	Jmp_Reg_Sel	LD_Sel [1:0]	LD_EN	ST_Sel [1:0]	RAM_EN	INT_INPUT_SEL	INT_PRINT_EN	INT_PRINT_CLEAR
All Previous Instructions	XX	XXXX	X	X	X	**0**	XXXX	X	X	X	X	X	X	X	X	**0**	**0**	**0**
Memory & IO Instructions	11	0000 (LOAD Direct)	X	X	X	**0**	XXXX	1	0	0	0	01	1	00	1	**0**	**0**	**0**
		0001 (LOAD Register Indirect)										10	1	00	0			
		0100 (STORE Direct)						0				00	0	01	1			
		0101 (STORE Register Indirect)										00	0	10	1			
		1101 (ACCEPT_INPUT)				**1**						**00**	**0**	**00**	**0**	**1**	**0**	**0**
		1110 (PRINT_OUTPUT)				**0**						**00**	**0**	**00**	**0**	**0**	**1**	**0**
		1111 (PRINT_CLEAR)										**00**	**0**	**00**	**0**	**0**	**0**	**1**

Control Logic for I/O Instruction in CPU

Control Logic Circuit

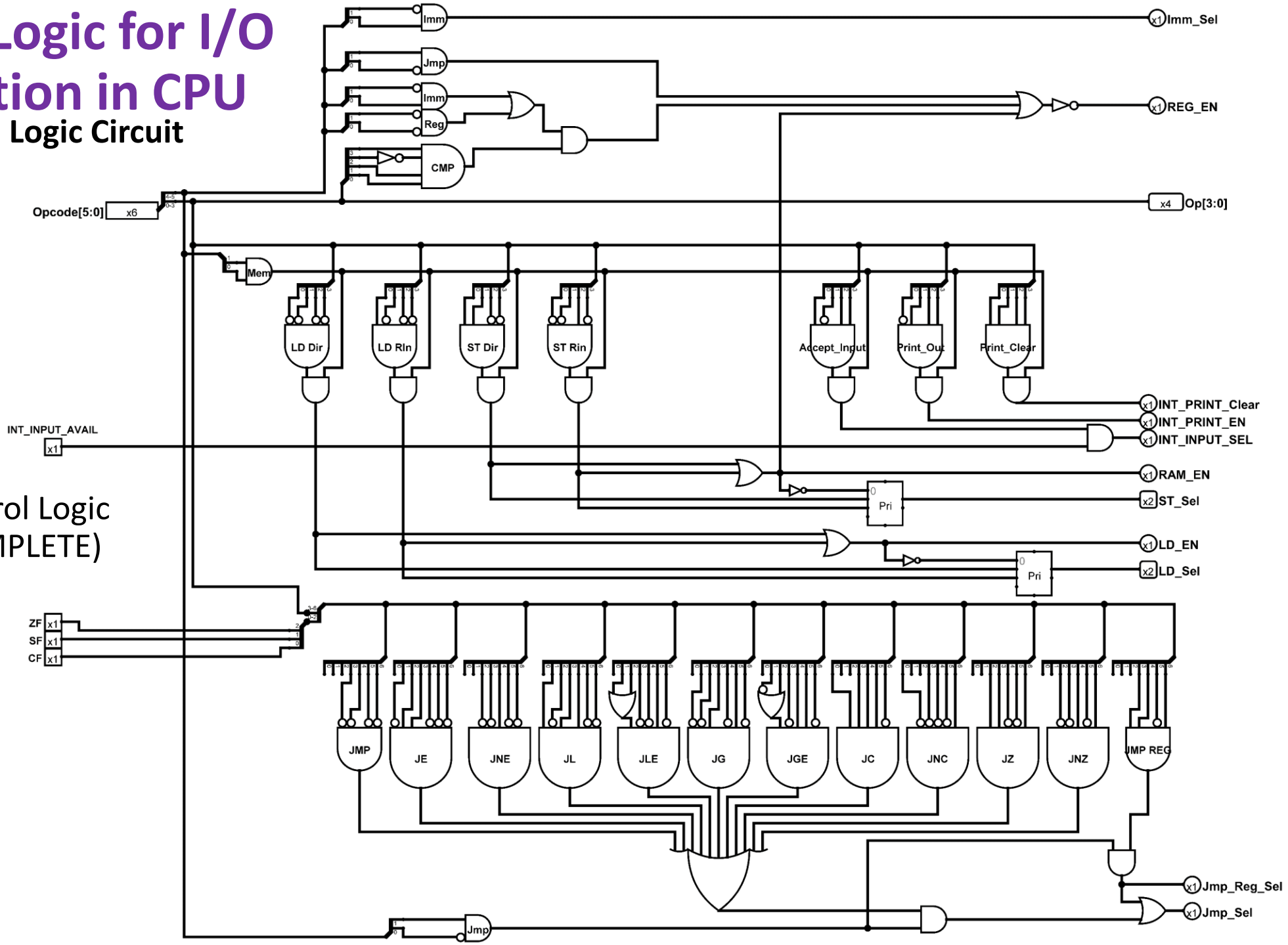
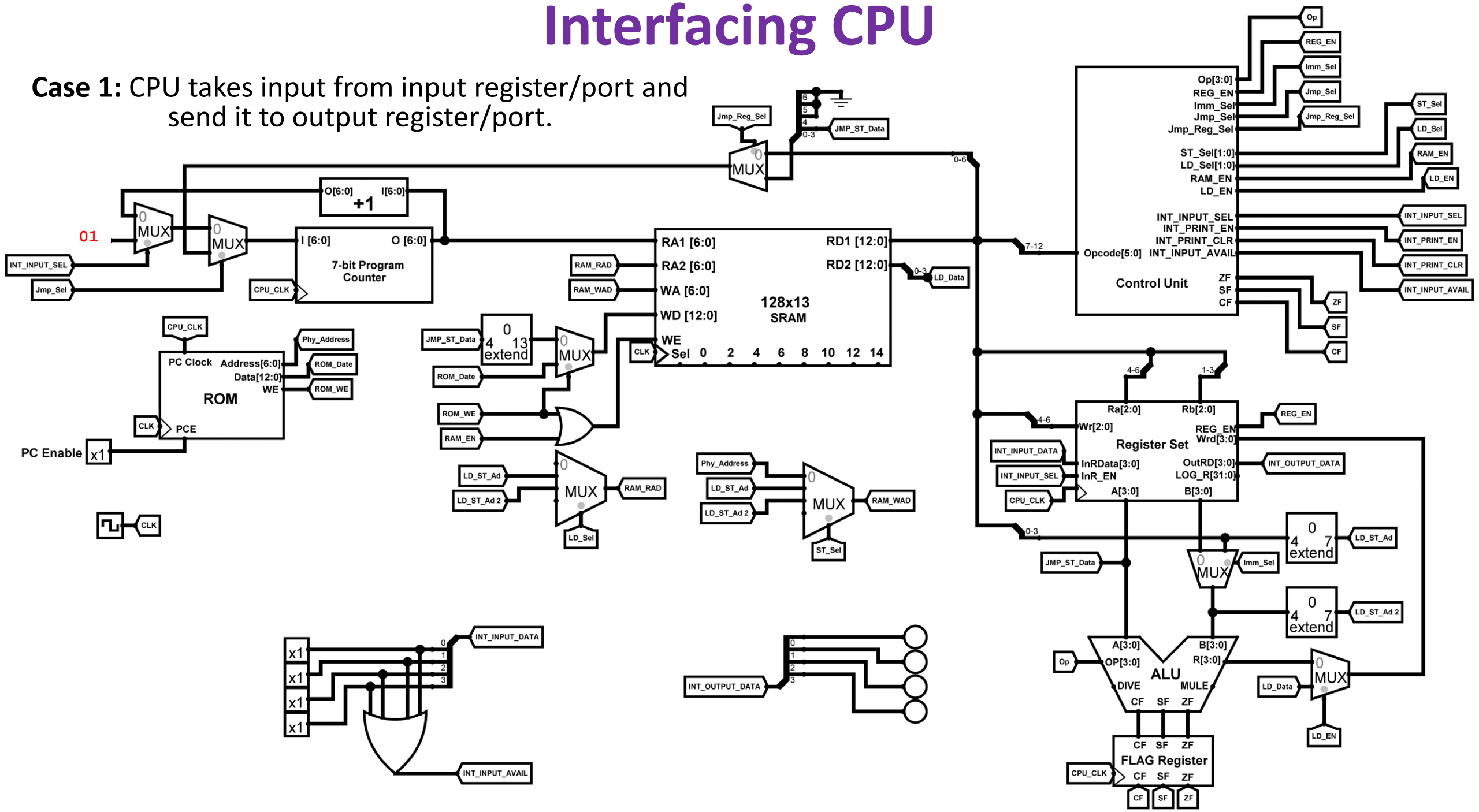


Figure: Control Logic Circuit (COMPLETE)

Interfacing CPU

Case 1: CPU takes input from input register/port and send it to output register/port.



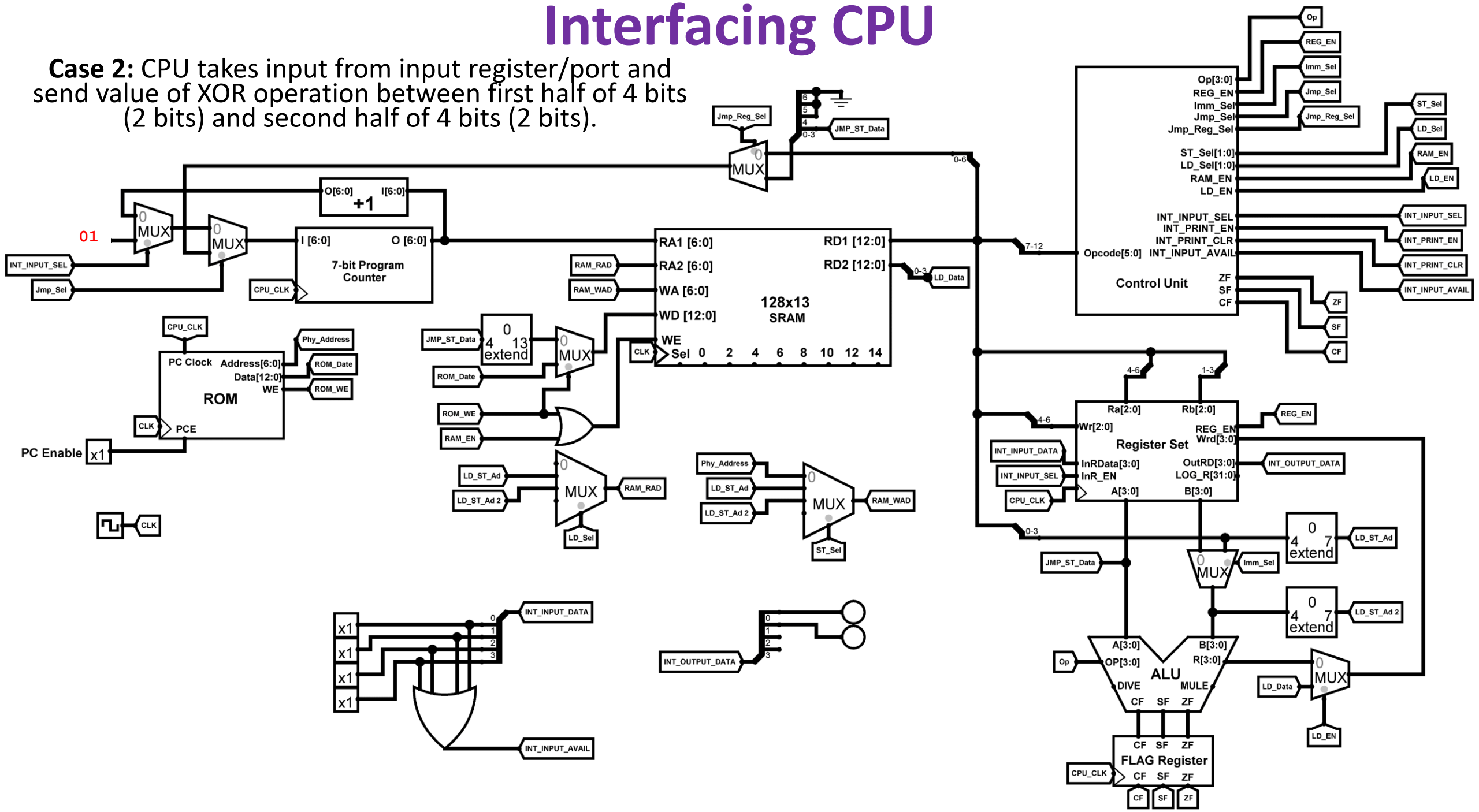
Interfacing CPU

Case 1: CPU takes input from input register/port and send it to output register/port.

```
JMP MAIN              #GOTO MAIN FUNCTION. SKIP NEXT ADDRESS
#=============================================================
#HARDWARE INPUT INTERRUPT
JMP INPUT_FOUND       #JUMP TO INPUT_FOUND WHEN ACCEPT_INPUT IS CALLED
#=============================================================

MAIN: # MAIN FUNCTION WITH NO ARGUMENT

    LOOP:
        XOR R0, R0   #RESET R0
        ACCEPT_INPUT #ALLOW DATA ON INPUT PORT TO BE SAVED ON INPUT REGISTER.
                     #THIS WILL BE ALLOWED WHEN IN_INPUT_AVAIL PIN IS 1.
                     #JUMP TO ADDRESS WHERE HARDWARE INPUT INTERRUPT LOCATION IS SAVED.

    INPUT_FOUND:
        ADD R0, INR      #COPY DATA ON INPUT REGISTER (INR) TO R0 REGISTER

        XOR OUTR, OUTR   #RESET OUTPUT REGISTER (OUTR)
        ADD OUTR, R0     #COPY DATA ON R0 REGISTER TO OUTPUT REGISTER (OUTR)

        JMP LOOP
```

Interfacing CPU

Case 2: CPU takes input from input register/port and send value of XOR operation between first half of 4 bits (2 bits) and second half of 4 bits (2 bits).



Interfacing CPU

Case 2: CPU takes input from input register/port and send value of XOR operation between first half of 4 bits (2 bits) and second half of 4 bits (2 bits) to output port.

```
JMP MAIN              #GOTO MAIN FUNCTION. SKIP NEXT ADDRESS
#=============================================================
#HARDWARE INPUT INTERRUPT
JMP INPUT_FOUND       #JUMP TO INPUT_FOUND WHEN ACCEPT_INPUT IS CALLED
#=============================================================

MAIN: # MAIN FUNCTION WITH NO ARGUMENT

    LOOP:
        XOR R0, R0   #RESET R0
        ACCEPT_INPUT #ALLOW DATA ON INPUT PORT TO BE SAVED ON INPUT REGISTER.
                     #THIS WILL BE ALLOWED WHEN IN_INPUT_AVAIL PIN IS 1.
                     #JUMP TO ADDRESS WHERE HARDWARE INPUT INTERRUPT LOCATION IS SAVED.

    INPUT_FOUND:
        ADD R0, INR      #COPY DATA ON INPUT REGISTER (INR) TO R0 REGISTER

        XOR R1, R1       #RESET R1
        OR R1, R0        #OR OP BETWEEN R0 and R1. IT WILL COPY VALUE OF R0 TO R1.
        AND R1, 12       #AND OP BETWEEN R1 and 1100. IT WILL GET 2ND HALF OF 4 BITS (2 BITS) OF R1.
        SHR R1, 2        #SHIFT RIGHT R1 BY 2 BITS. IT WILL HELP GET REAL VALUE OF 2ND HALF OF 4 BITS (2 BITS).
```

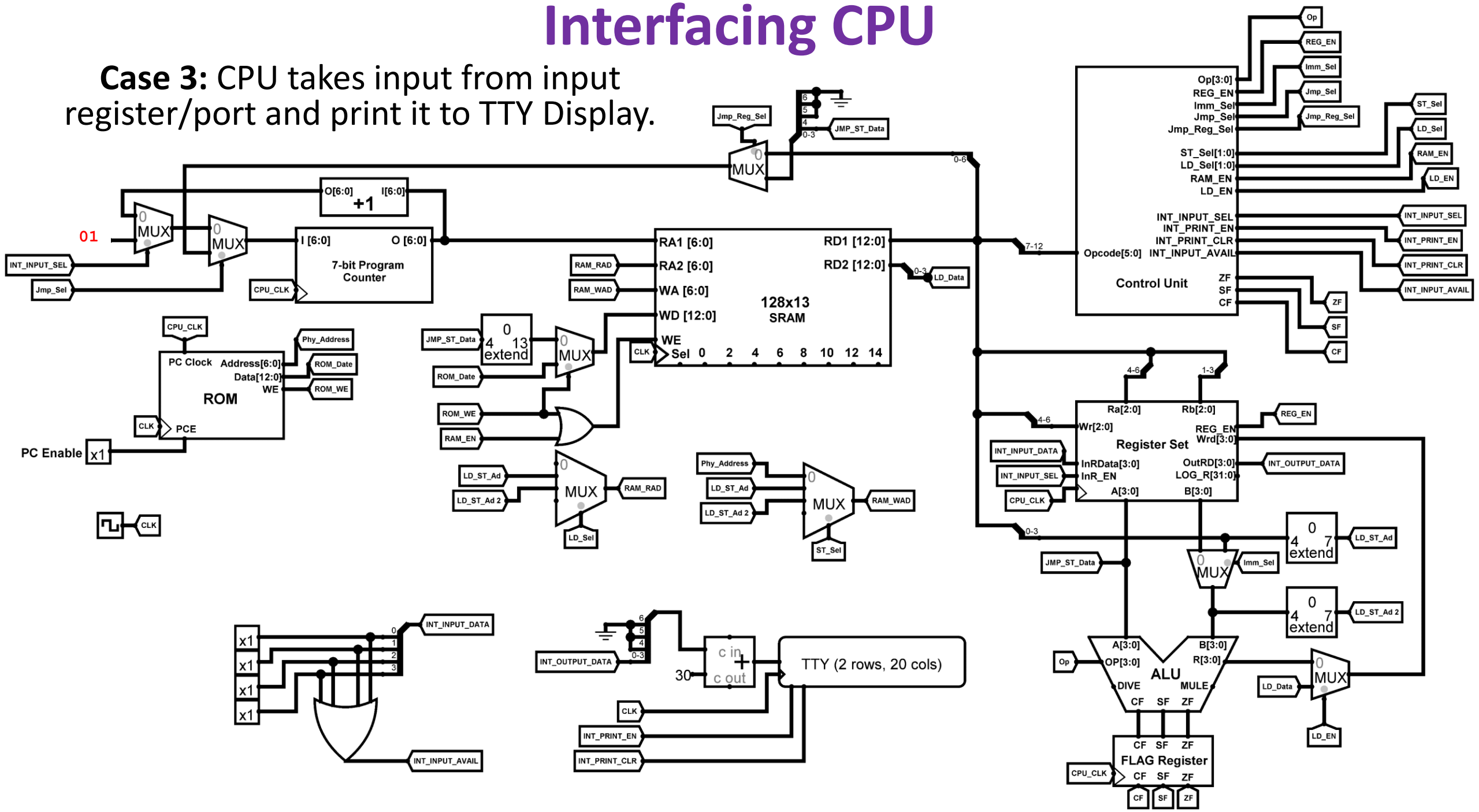
Interfacing CPU

Case 2: CPU takes input from input register/port and send value of XOR operation between first half of 4 bits (2 bits) and second half of 4 bits (2 bits) to output port.

```
XOR R2, R2        #RESET R2
OR R2, R0         #OR OP BETWEEN R2 and R0. IT WILL COPY VALUE OF R0 TO R2.
AND R2, 3         #AND OP BETWEEN R2 and 0011. IT WILL GET 1ST HALF OF 4 BITS (2 BITS) OF R2.

XOR R1, R2        #PERFORM XOR OPERATION BETWEEN 1ST HALF AND 2ND HALF OF 4 BITS.

XOR OUTR, OUTR    #RESET OUTPUT REGISTER (OUTR)
ADD OUTR, R1      #COPY DATA ON R1 REGISTER TO OUTPUT REGISTER (OUTR)
PRINT_CLEAR       #CLEAR DISPLAY SCREEN
PRINT_OUTPUT      #PRINT OUTPUT IN DISPLAY

JMP LOOP
```

Interfacing CPU

Case 3: CPU takes input from input register/port and print it to TTY Display.

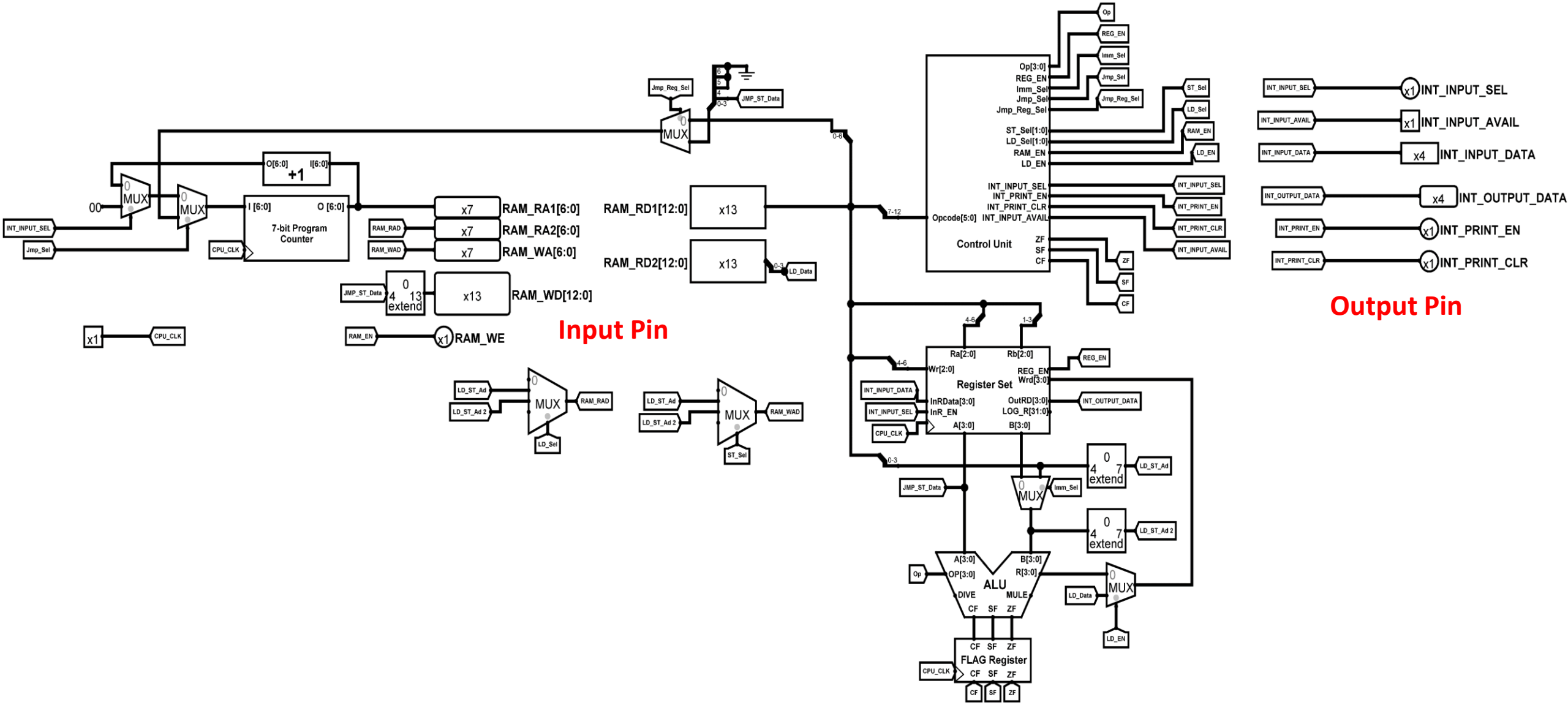


Interfacing CPU

Case 3: CPU takes input from input register/port and print it to TTY display.

```
JMP MAIN              #GOTO MAIN FUNCTION. SKIP NEXT ADDRESS
#=============================================================
#HARDWARE INPUT INTERRUPT
JMP INPUT_FOUND       #JUMP TO INPUT_FOUND WHEN ACCEPT_INPUT IS CALLED
#=============================================================

MAIN: # MAIN FUNCTION WITH NO ARGUMENT

    LOOP:
        XOR R0, R0   #RESET R0
        ACCEPT_INPUT #ALLOW DATA ON INPUT PORT TO BE SAVED ON INPUT REGISTER.
                     #THIS WILL BE ALLOWED WHEN IN_INPUT_AVAIL PIN IS 1.
                     #JUMP TO ADDRESS WHERE HARDWARE INPUT INTERRUPT LOCATION IS SAVED.

    INPUT_FOUND:
        ADD R0, INR      #COPY DATA ON INPUT REGISTER (INR) TO R0 REGISTER

        XOR OUTR, OUTR   #RESET OUTPUT REGISTER (OUTR)
        ADD OUTR, R0     #COPY DATA ON R0 REGISTER TO OUTPUT REGISTER (OUTR)
        PRINT_CLEAR      #CLEAR DISPLAY SCREEN
        PRINT_OUTPUT     #PRINT OUTPUT IN DISPLAY

        JMP LOOP
```

Separating RAM from CPU



Separating RAM from CPU

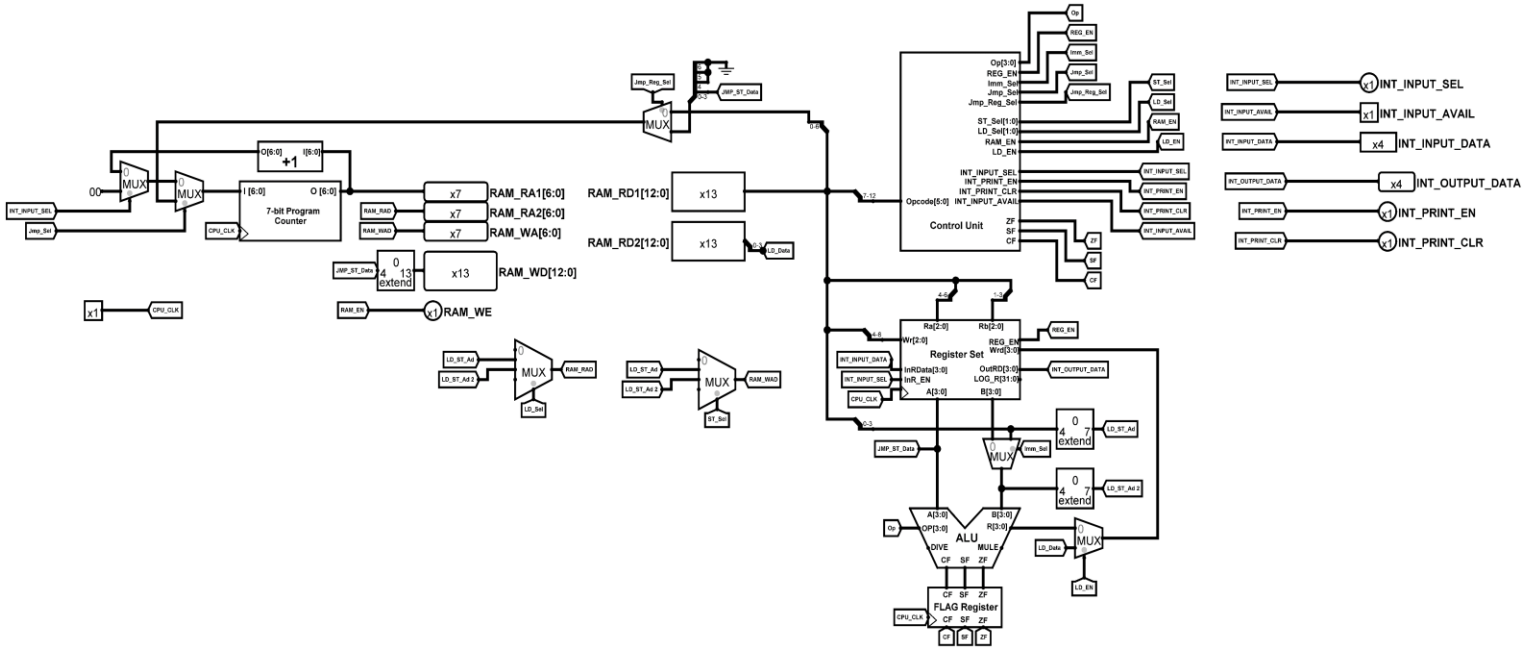
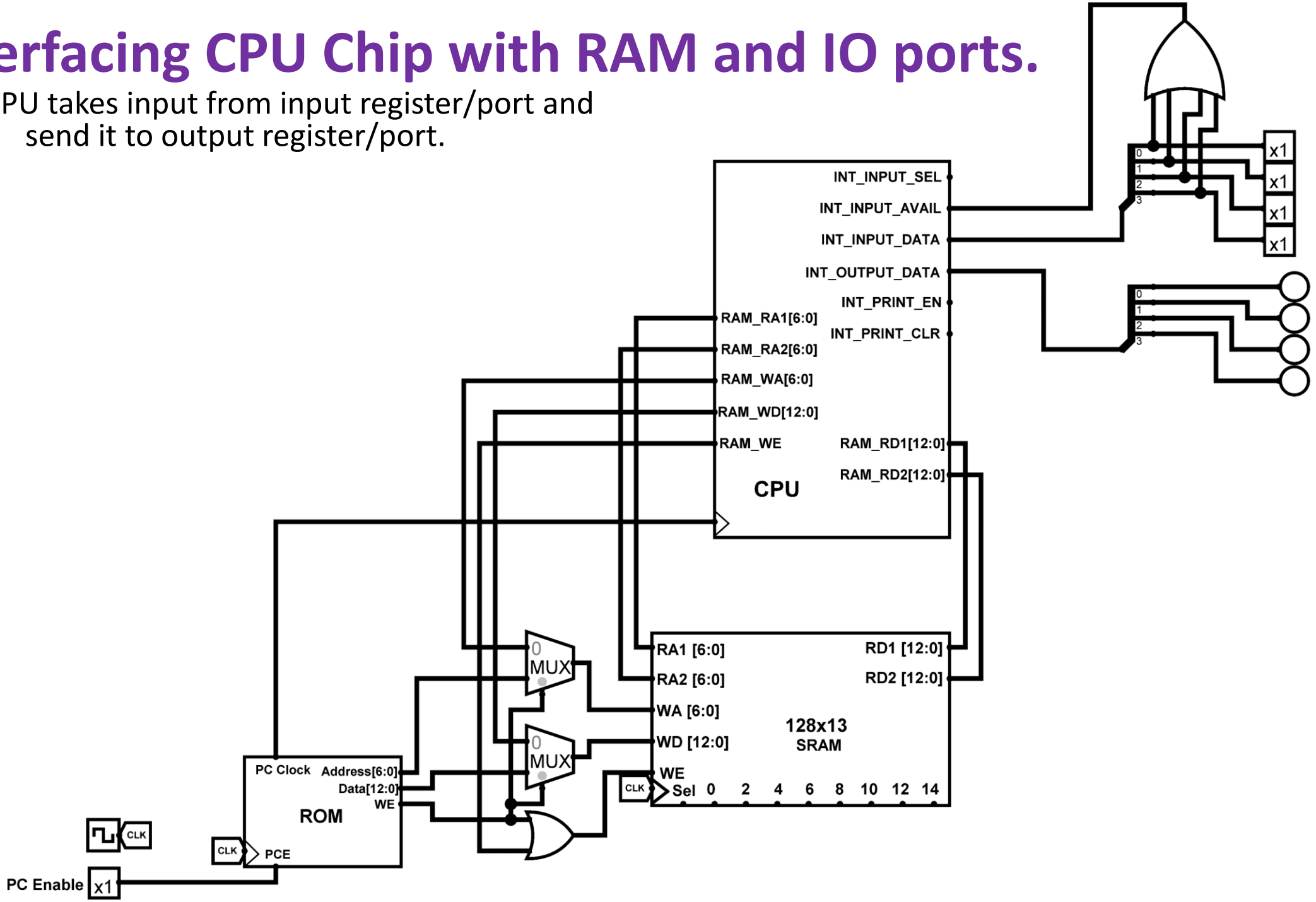


Fig: CPU Chip

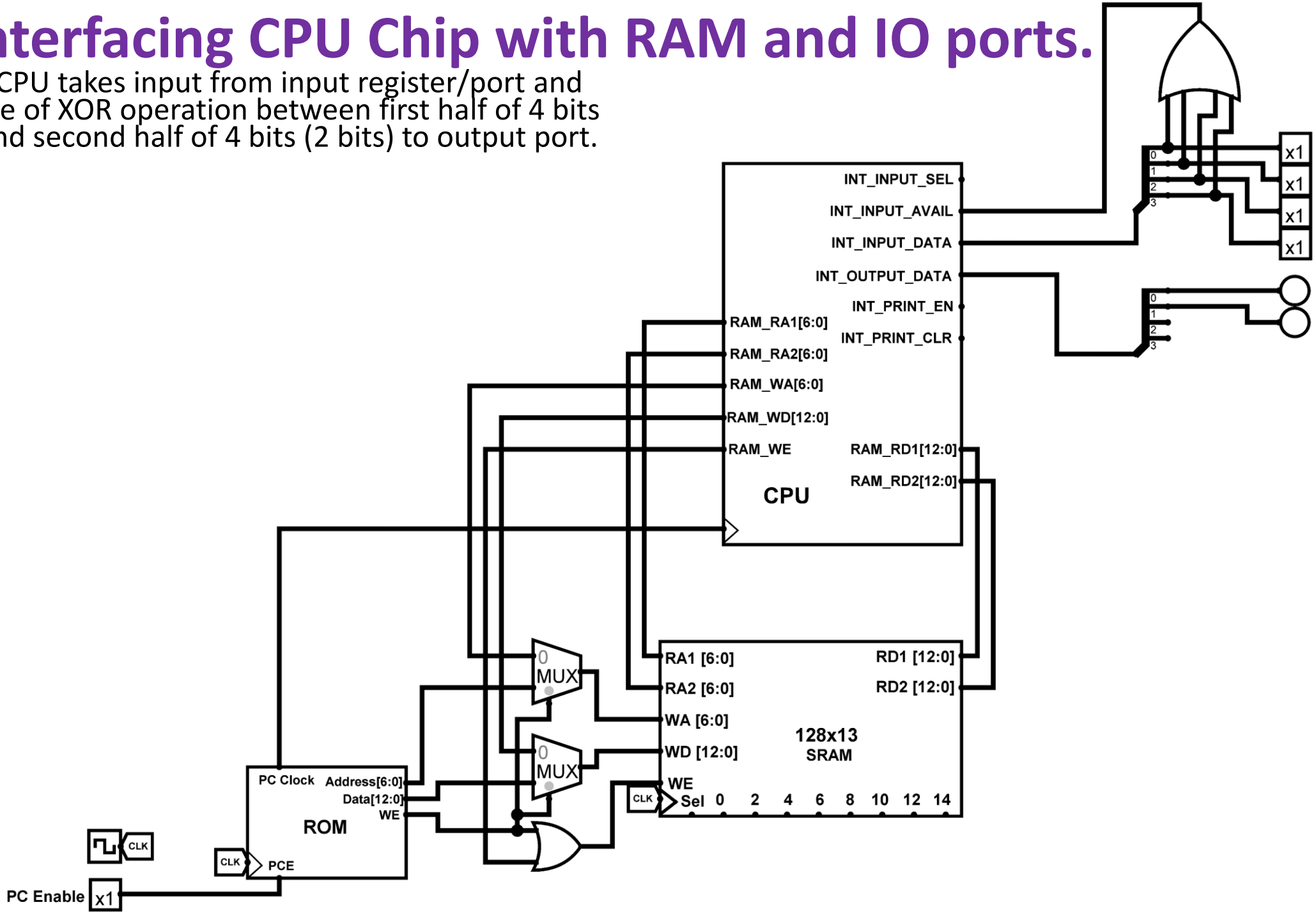
Interfacing CPU Chip with RAM and IO ports.

Case 1: CPU takes input from input register/port and send it to output register/port.



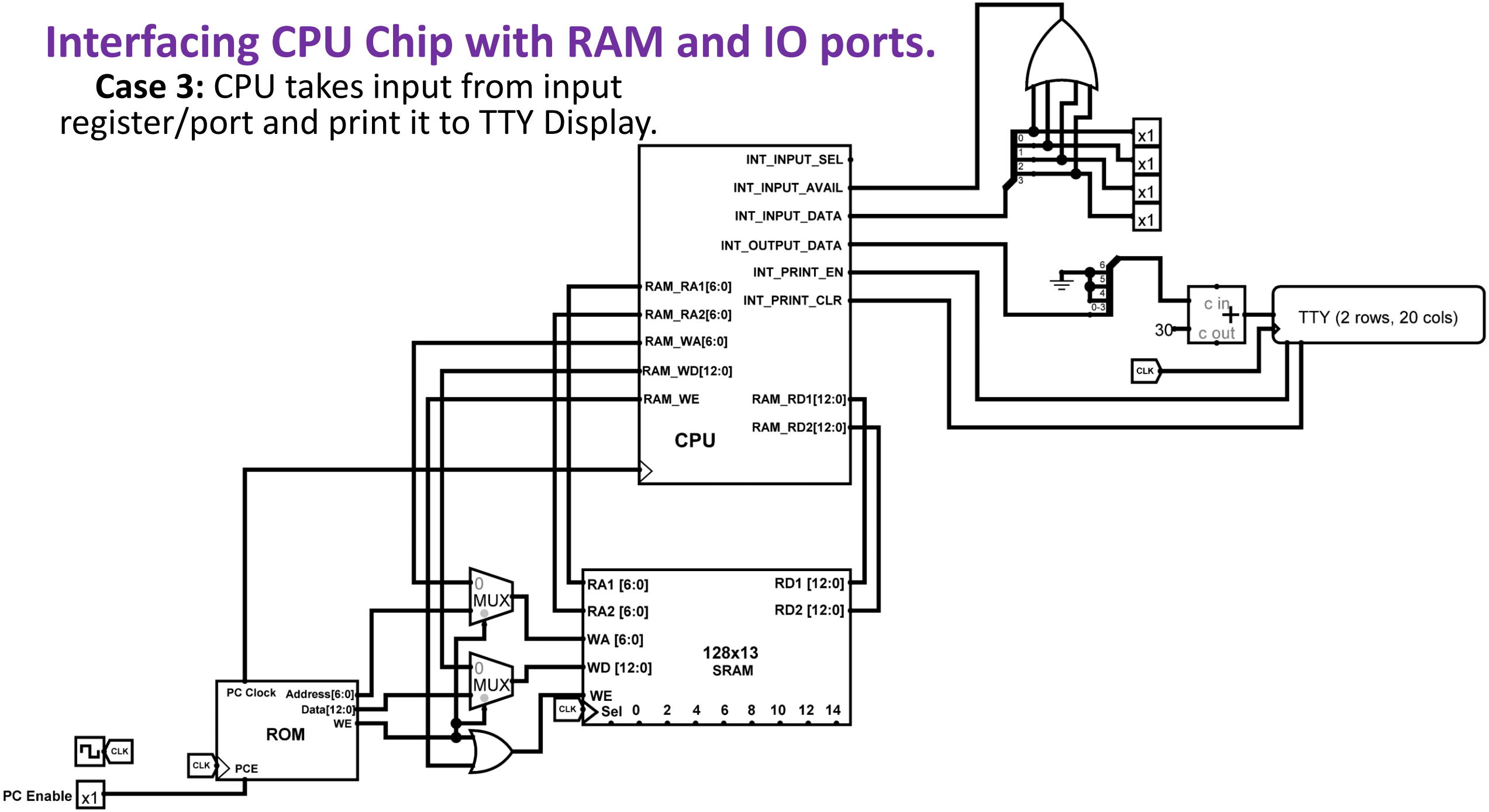
Interfacing CPU Chip with RAM and IO ports.

Case 2: CPU takes input from input register/port and send value of XOR operation between first half of 4 bits (2 bits) and second half of 4 bits (2 bits) to output port.



Interfacing CPU Chip with RAM and IO ports.

Case 3: CPU takes input from input register/port and print it to TTY Display.



Real Life Example
(Temperature Measurement Project)

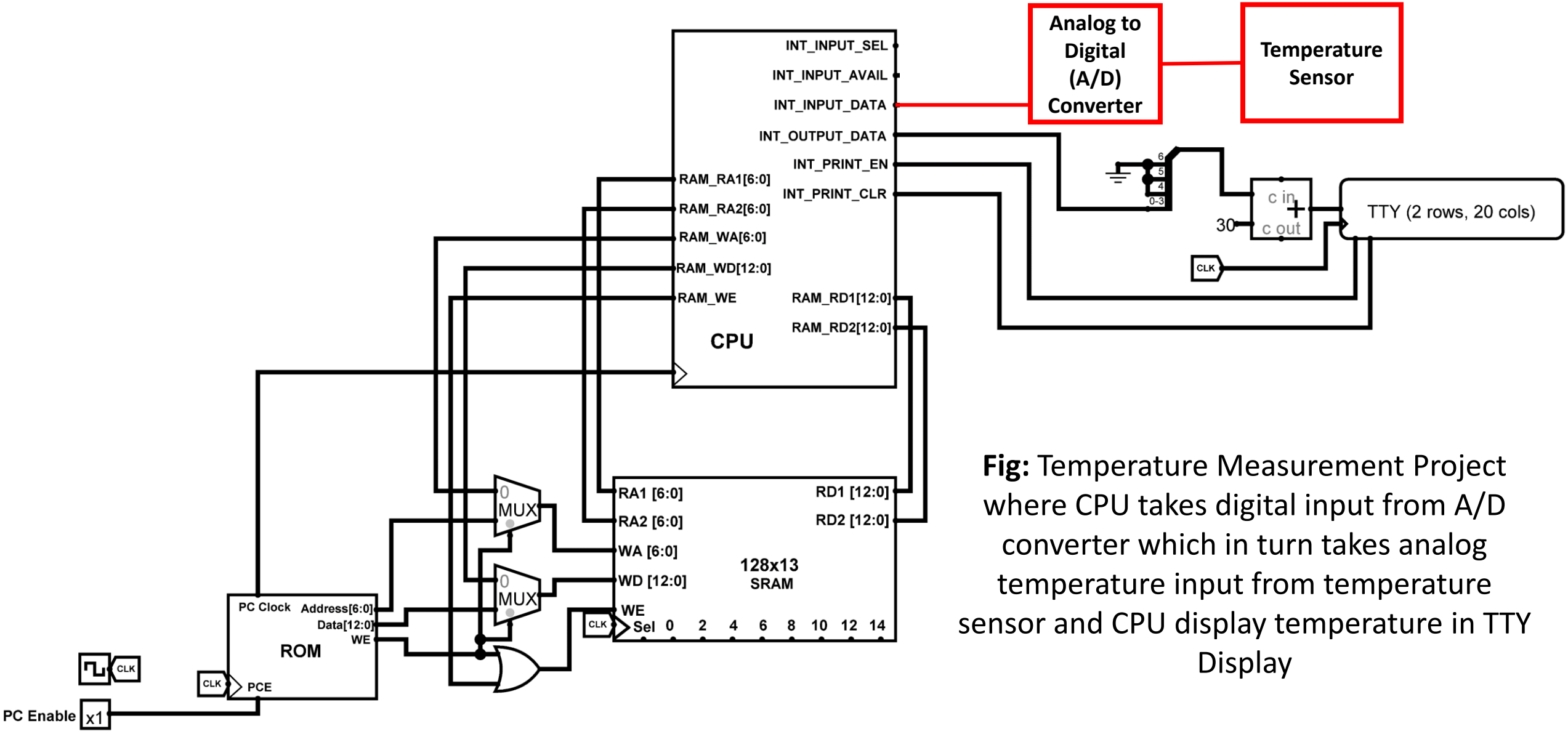
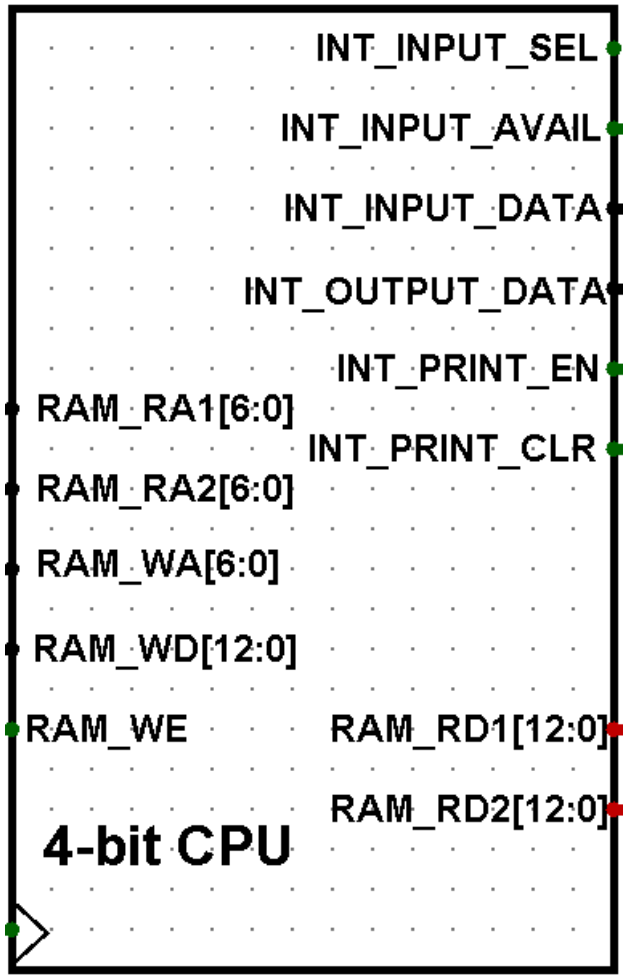


Fig: Temperature Measurement Project where CPU takes digital input from A/D converter which in turn takes analog temperature input from temperature sensor and CPU display temperature in TTY Display

Example: Interfacing CPU with RAM & IO

Question: Consider following CPU chip



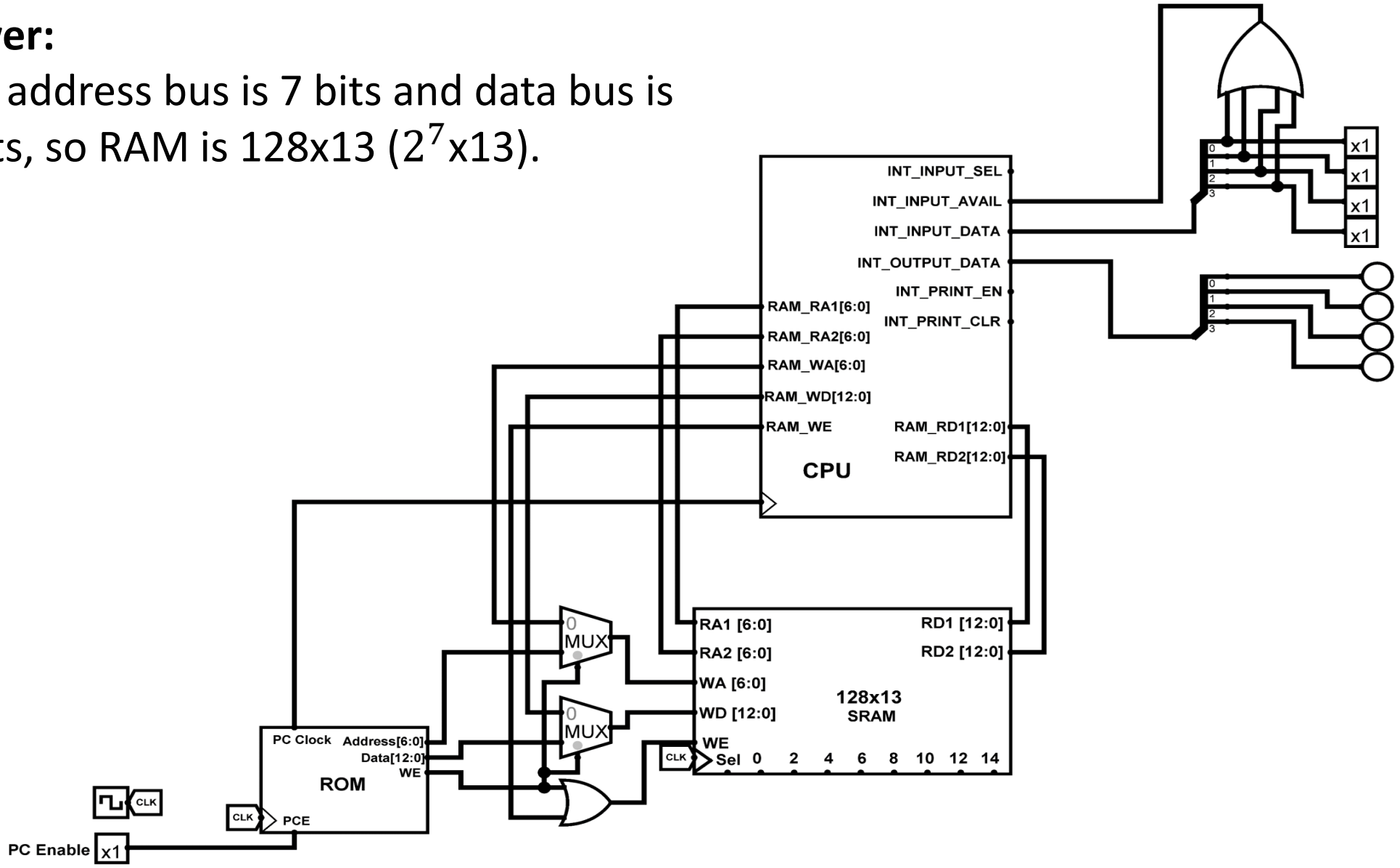
1. Connect CPU chip with RAM chip, connect 4 switches to input port and 4 LEDs to output port of CPU.

2. Write down assembly program so that input data can be read from input port and send it to output port.

Example: Interfacing CPU with RAM & IO

Answer:

Since address bus is 7 bits and data bus is 13 bits, so RAM is 128x13 (2^7x13).



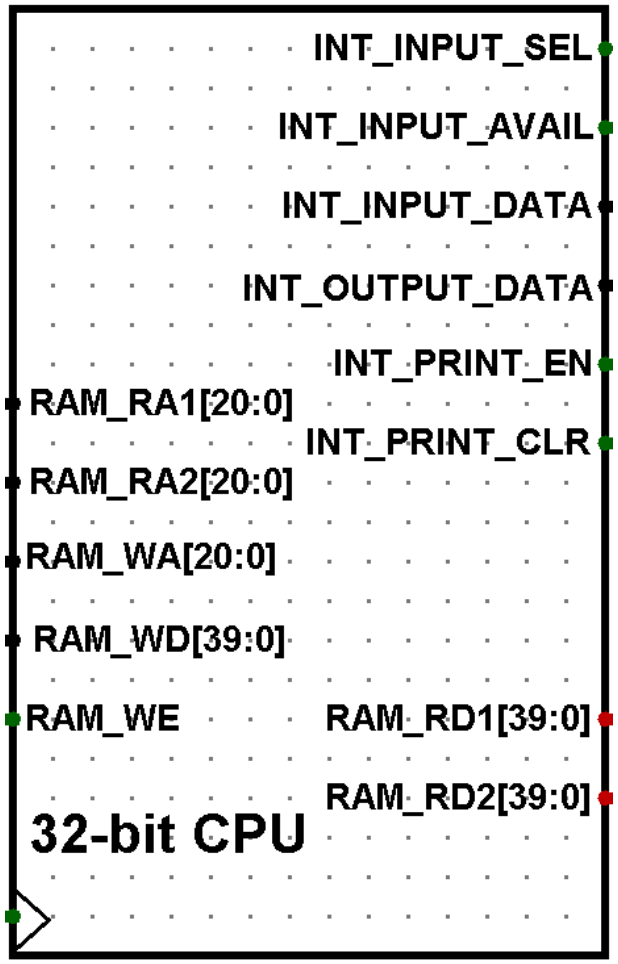
Example: Interfacing CPU with RAM & IO

Assembly program is shown below:

```
JMP MAIN              #GOTO MAIN FUNCTION. SKIP NEXT ADDRESS
#=============================================================
#HARDWARE INPUT INTERRUPT
JMP INPUT_FOUND       #JUMP TO INPUT_FOUND WHEN ACCEPT_INPUT IS CALLED
#=============================================================

MAIN: # MAIN FUNCTION WITH NO ARGUMENT

    LOOP:
        XOR R0, R0   #RESET R0
        ACCEPT_INPUT #ALLOW DATA ON INPUT PORT TO BE SAVED ON INPUT REGISTER.
                     #THIS WILL BE ALLOWED WHEN IN_INPUT_AVAIL PIN IS 1.
                     #JUMP TO ADDRESS WHERE HARDWARE INPUT INTERRUPT LOCATION IS SAVED.

    INPUT_FOUND:
        ADD R0, INR      #COPY DATA ON INPUT REGISTER (INR) TO R0 REGISTER

        XOR OUTR, OUTR   #RESET OUTPUT REGISTER (OUTR)
        ADD OUTR, R0     #COPY DATA ON R0 REGISTER TO OUTPUT REGISTER (OUTR)

        JMP LOOP
```

Exercises

1. Convert following assembly code to machine code:

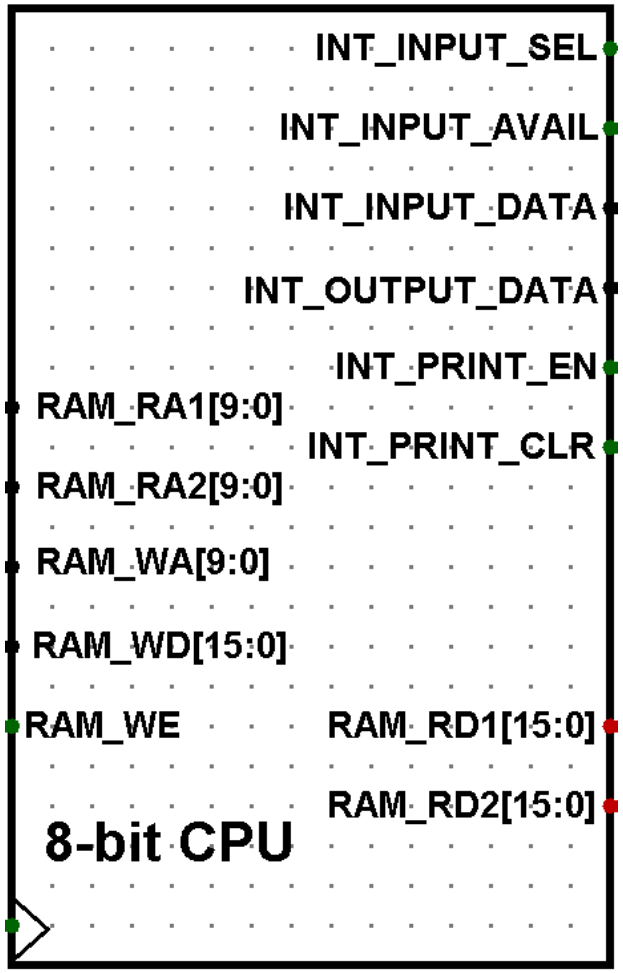


a) Connect CPU chip with RAM chip, connect 32 switches to input port and 32 LEDs to output port of CPU.

b) Write down assembly program so that input data can be read from input port and send value of XOR operation between first half of 32 bits (16 bits) and second half of 32 bits (16 bits) to output port.

Exercises

2. Convert following assembly code to machine code:



a) Connect CPU chip with RAM chip, connect 8 switches to input port and TTY display to output port.

b) Write down assembly program so that add operation can be performed between first half of 32 switches and second half of 32 switches to provide output.

Thank you ☺