

Design and implementation of 16-bit RISC Processor using Harvard Architecture

Motivating Background

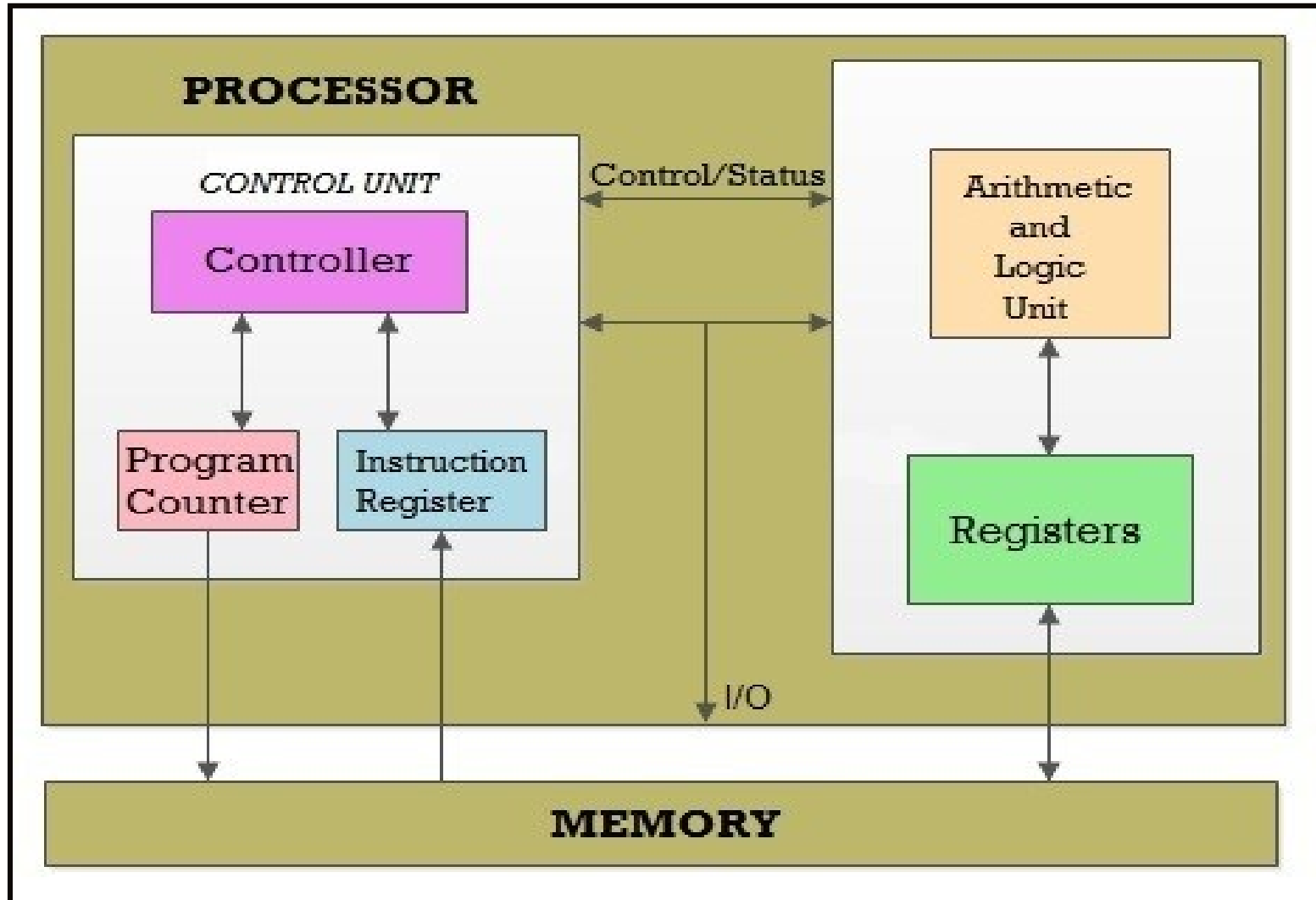
❑ INTEL 8085 microprocessor

- 8085 - based on

von Neumann architecture

- Simple instructions for programs - *RISC* strategy - simplified CPU design.
-

GENERIC PROCESSOR



The components of a generic processor

Control unit (CU)

Arithmetic and Logic unit (ALU)

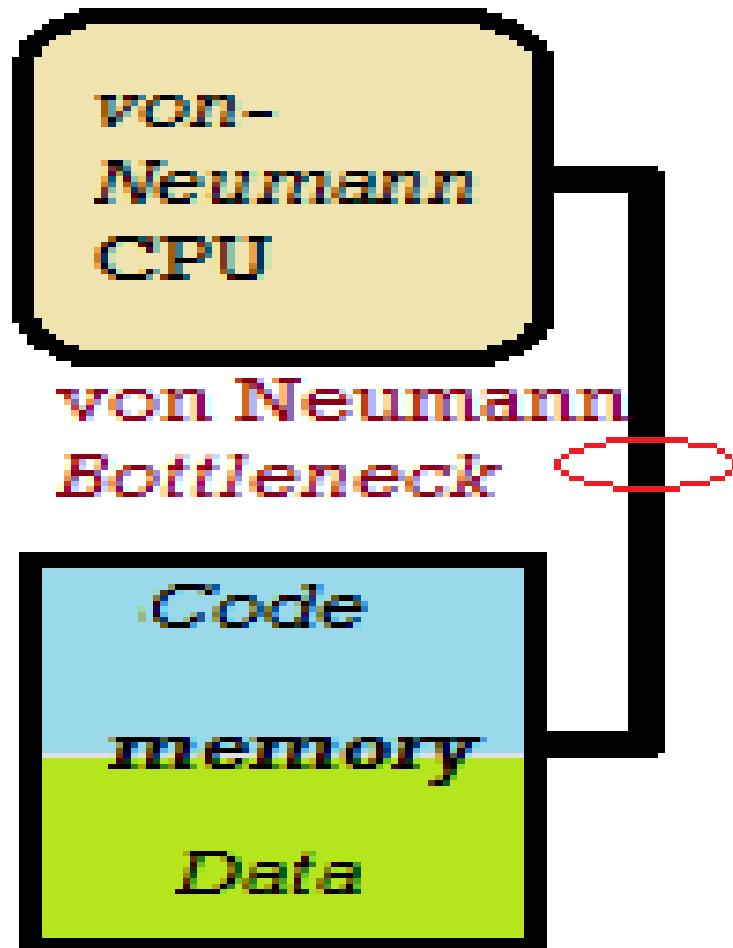
Register

Program Counter

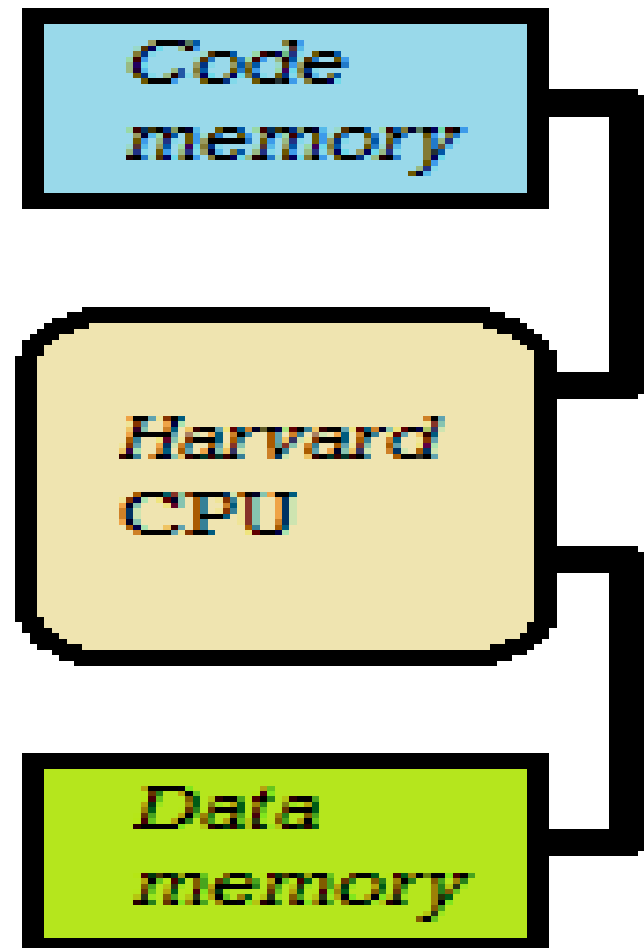
Instruction Register

ARCHITECTURE

von Neumann



Harvard



COMPUTING

RISC = *Reduced Instruction Set Computer*

Simplified instructions => Faster Execution

More program storage memory

Lesser Hardware

CISC = *Complex Instruction Set Computer*

Complex instructions => Slower Execution

Lesser program storage memory

More Hardware

A solid orange horizontal bar at the bottom of the slide.

PROCESSOR DESCRIPTION

RISC CPU – 16 bit Data – 12 Bit Address

Harvard architecture

2K bytes code memory ROM

2K bytes data memory RAM

Processor components:

Arithmetic and logic Unit (ALU)

Accumulator (Acc) - 16 bit register

Program Counter (PC) - 12 bit register

Instruction Register (IR) - 4 bit register

Memory Address Register (MAR) - 12 bit register

Control unit (CU)

Bus:

Data bus (16 bit)

Address bus (12 bit)

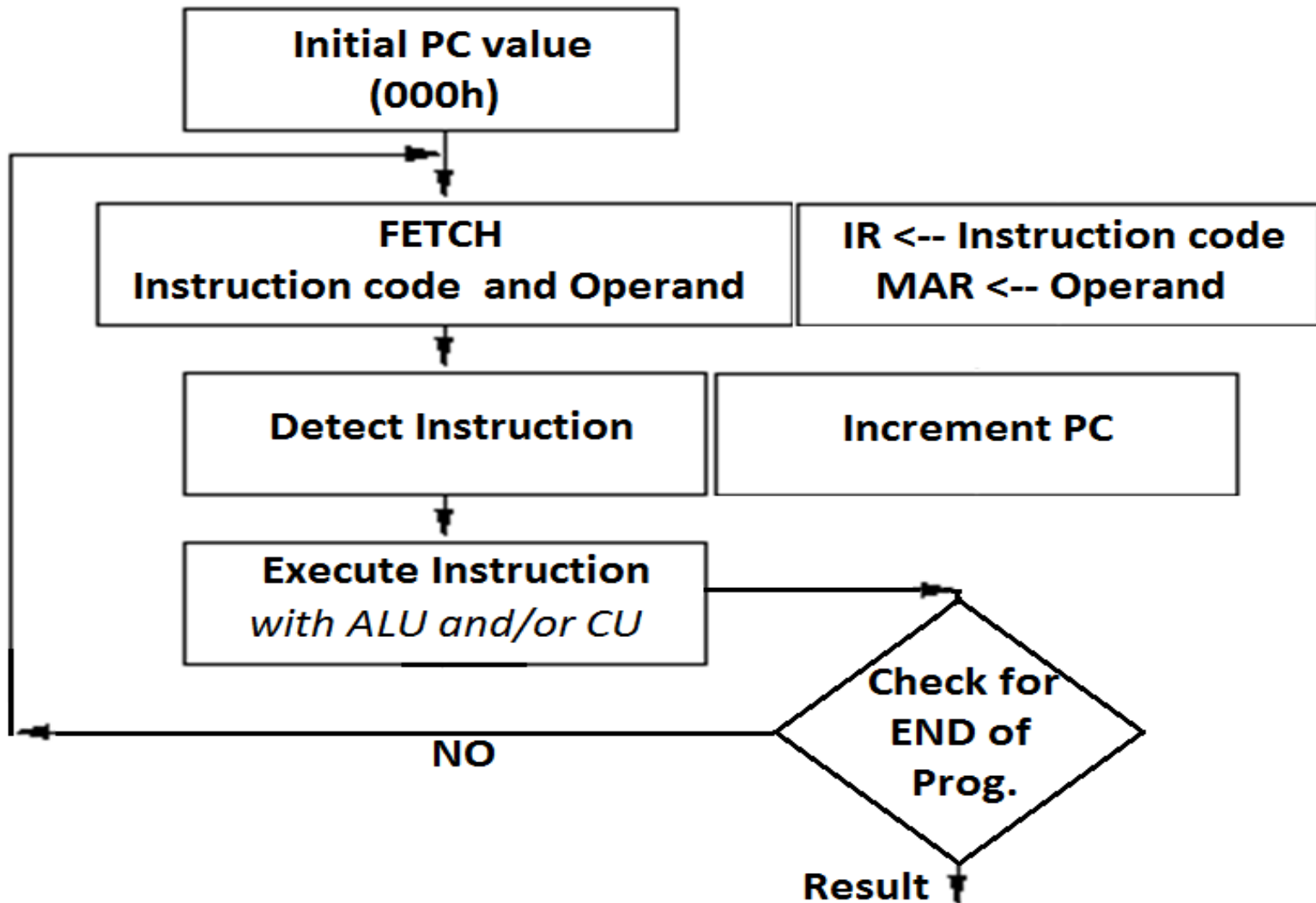
Memory:

Program memory (ROM) - 2K bytes)

Data Memory (RAM) – 2K bytes)

A solid orange horizontal bar at the bottom of the slide.

Operation Flow :



Instruction Set

<i>Op-Code</i>	<i>Instruction</i>	<i>Operation</i>
0h	ADD #Addr	$A \leftarrow [A] + [\text{\#Addr}]$
1h	SUB #Addr	$A \leftarrow [A] - [\text{\#Addr}]$
2h	MUL #Addr	$A \leftarrow [A] * [\text{\#Addr}]$
3h	DIV #Addr	$A \leftarrow [A] / [\text{\#Addr}]$
4h	AND #Addr	$A \leftarrow [A] \text{ AND } [\text{\#Addr}]$
5h	OR #Addr	$A \leftarrow [A] \text{ OR } [\text{\#Addr}]$
6h	XOR #Addr	$A \leftarrow [A] \text{ XOR } [\text{\#Addr}]$
7h	NOTA	$A \leftarrow \text{NOT } [A]$
8h	LDA #Addr	$A \leftarrow [\text{\#Addr}]$
9h	STA #Addr	$\text{\#Addr} \leftarrow [A]$
Ah	JZ #Addr	$\text{PC} \leftarrow \text{\#Addr}$ if $[A] = 0$
Bh	JNZ #Addr	$\text{PC} \leftarrow \text{\#Addr}$ if $[A] \text{ not } = 0$
Ch	JMP #Addr	$\text{PC} \leftarrow \text{\#Addr}$
Dh		
Eh		
Fh	END	$\text{PC} \leftarrow 000$

Hexadecimal '7h'
= 4 bit binary '0111'

#Addr =>
Hexadecimal value of
address (000 to FFF)

[#Addr] => The content
of memory address
#Addr

A => Accumulator
register

[A] => The content of
the
Accumulator

ARITHMETIC GROUP

- **ADD #Addr**
- **SUB #Addr**
- **MUL #Addr**
- **DIV #Addr**

DATA TRANSFER GROUP

- **LDA #Addr**
- **STA #Addr**

LOGIC GROUP

- **AND #Addr**
- **OR #Addr**
- **XOR #Addr**
- **NOT Addr**

BRANCH GROUP

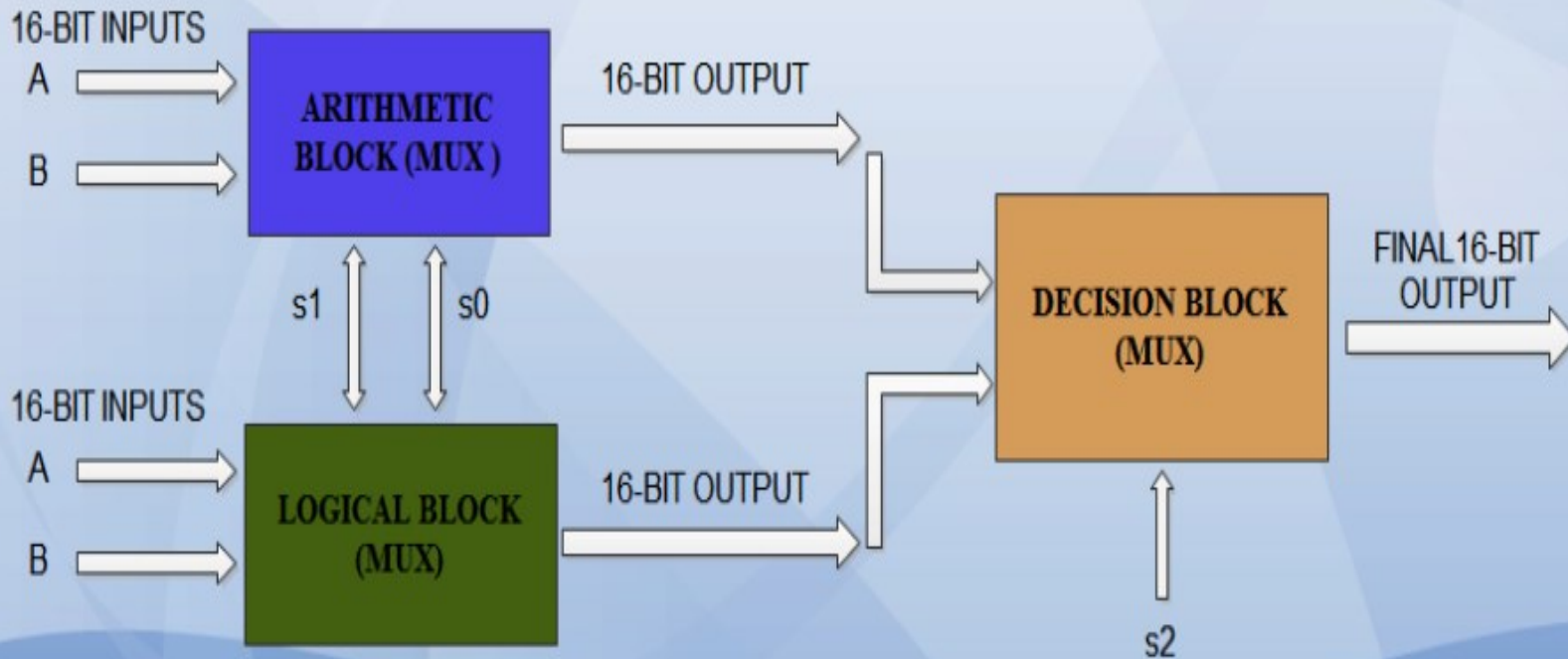
- **JZ #Addr**
- **JNZ #Addr**
- **JMP #Addr**

OTHER

- **END**

ALU Modeling

STRUCTURAL BLOCK DIAGRAM OF ALU.



LEGEND : s2,s1 and s0 :SELECT LINES

SELECT LINES

<i>S2</i>	<i>S1</i>	<i>S0</i>	<i>Operation</i>
0	0	0	ADD
0	0	1	SUBTRACT
0	1	0	MULTIPLY
0	1	1	DIVIDE
1	0	0	AND
1	0	1	OR
1	1	0	XOR
1	1	1	NOT

Memory Modeling

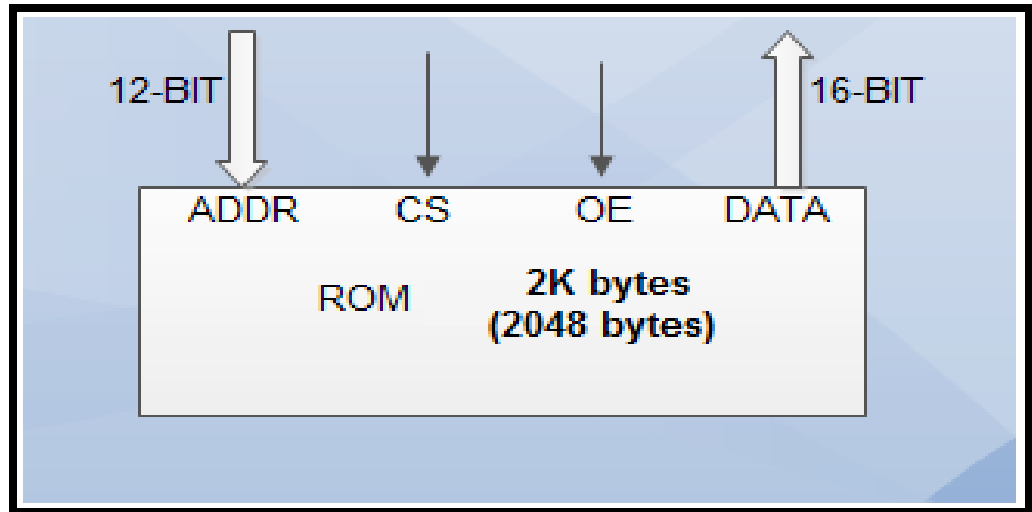
ROM

Inputs:

- Chip Select - CS
- Output Enable - OE
- Address - ADDR 12bit

Output:

- Data - DATA 16 bit



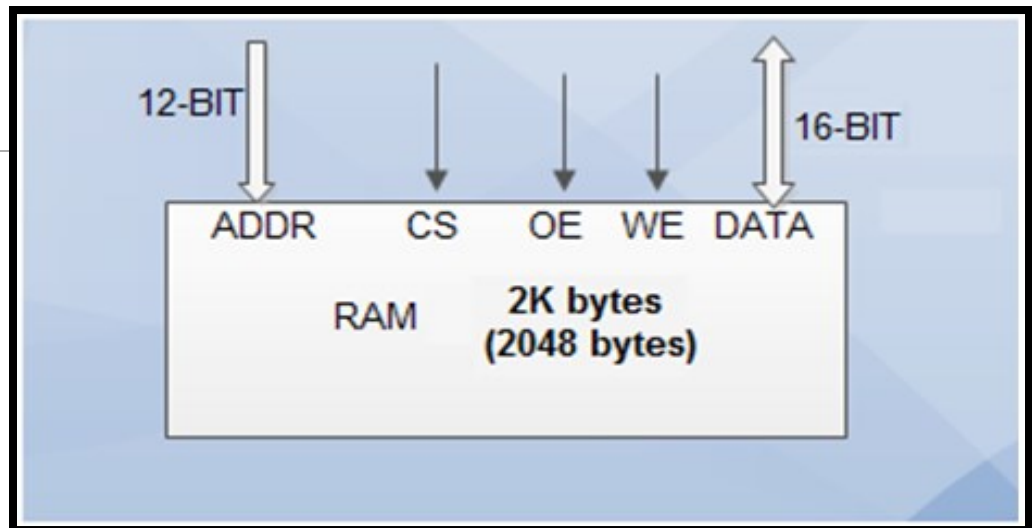
RAM

Inputs:

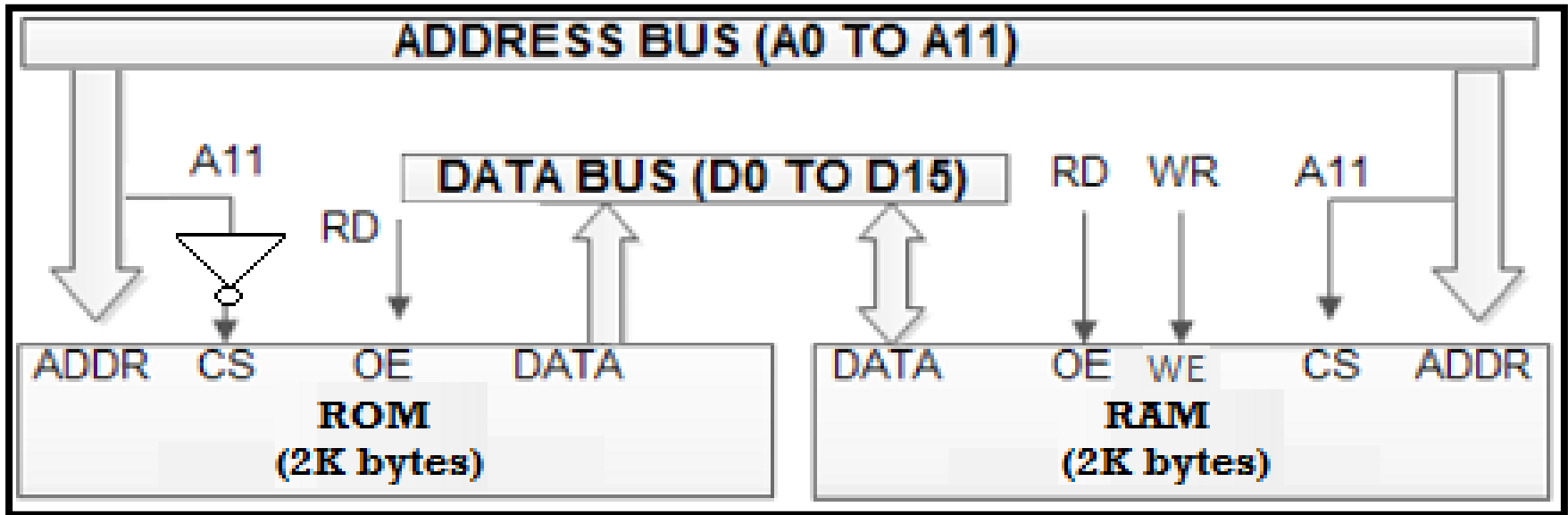
- Chip Select - CS)
- Output Enable - OE
- Write Enable - WE
- Address - ADDR 12 bit

Bidirectional:

- Data - DATA 16 bit



Memory Interfacing



A11 → chip select input for the ROM and the RAM.

- ***When A11 is '0' (low), then ROM gets selected***
- ***When A11 is '1' (high), then RAM gets selected***

ROM address 000h to 7FFh

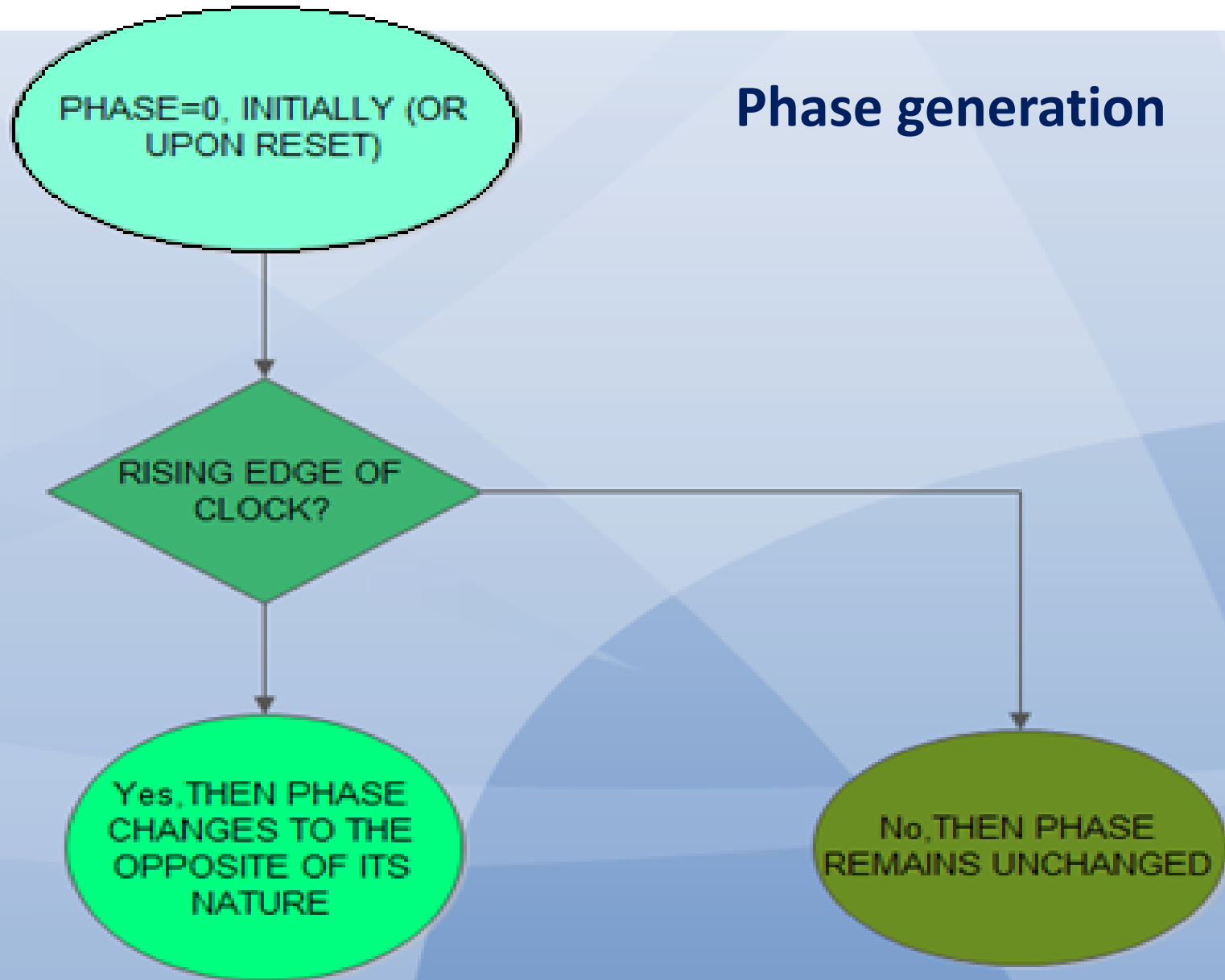
RAM address 800h to FFFh

Control Unit Modeling

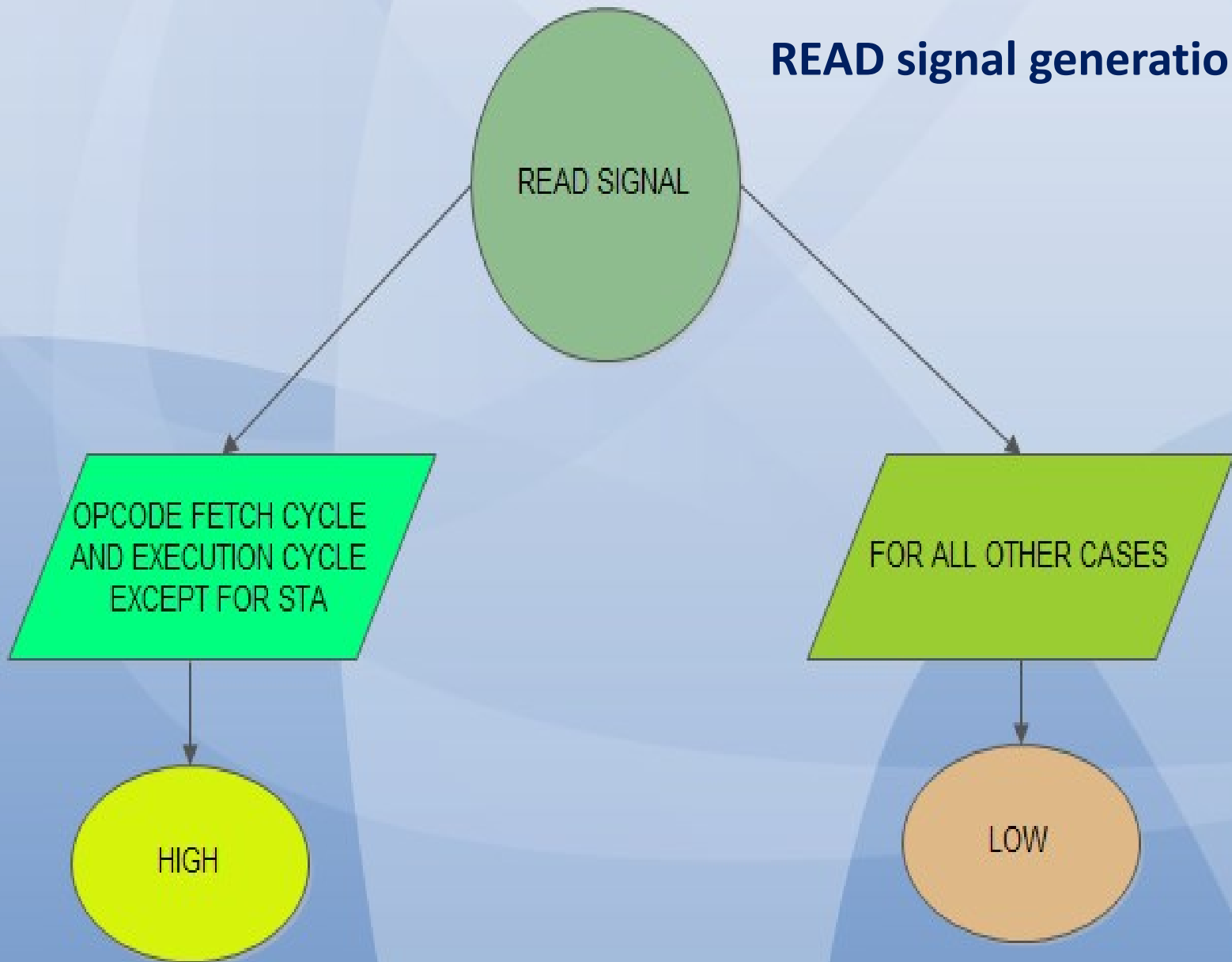
Different Control Unit functions are:

- ✓ Phase generation
 - ✓ Read & Write signal generation
 - ✓ Transfer of specific address to Address Bus
 - ✓ Transfer of data into Data Bus
 - ✓ Program Counter update
 - ✓ Op-Code & Operand fetch
 - ✓ Accumulator Update
-

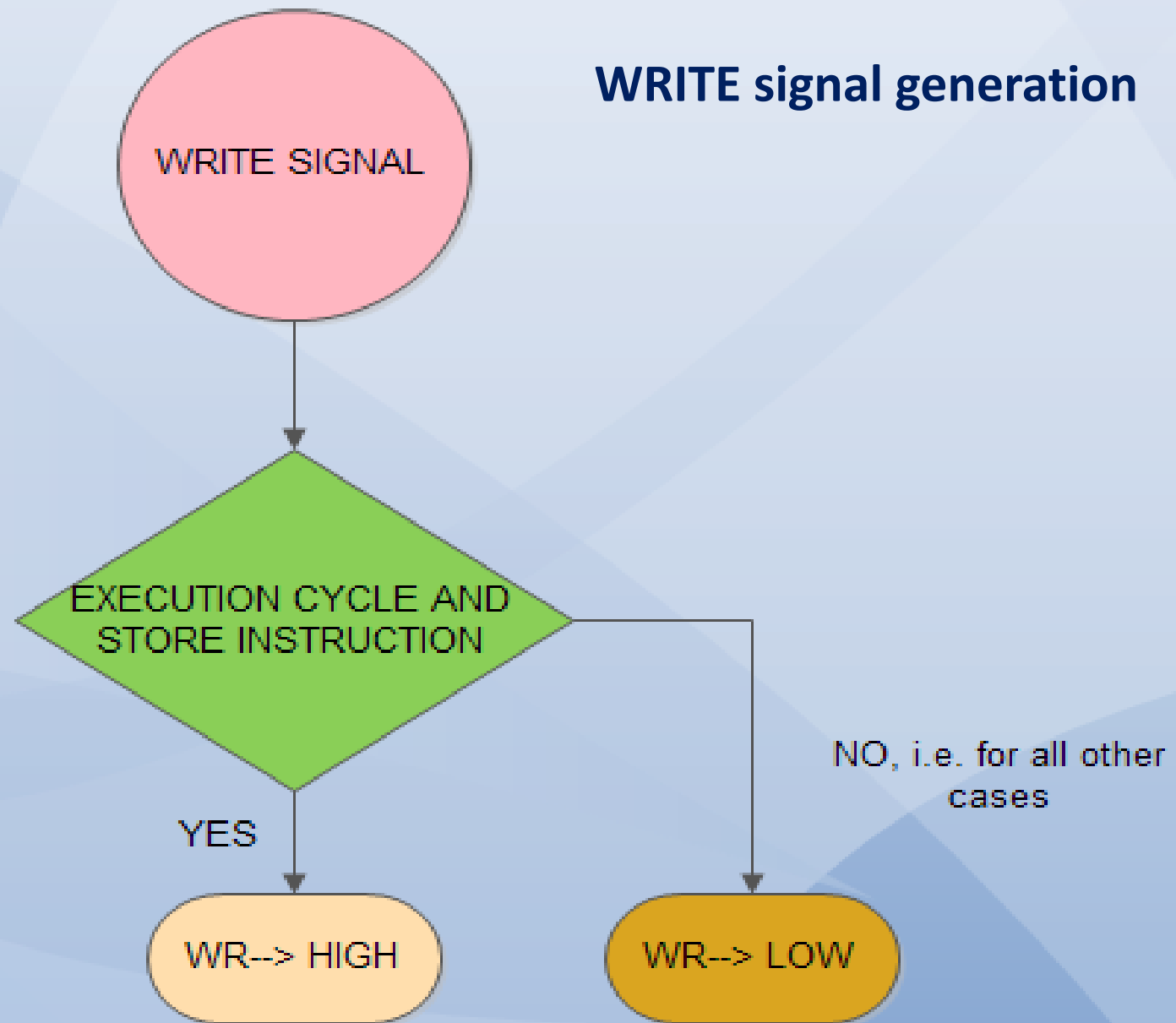
Phase generation



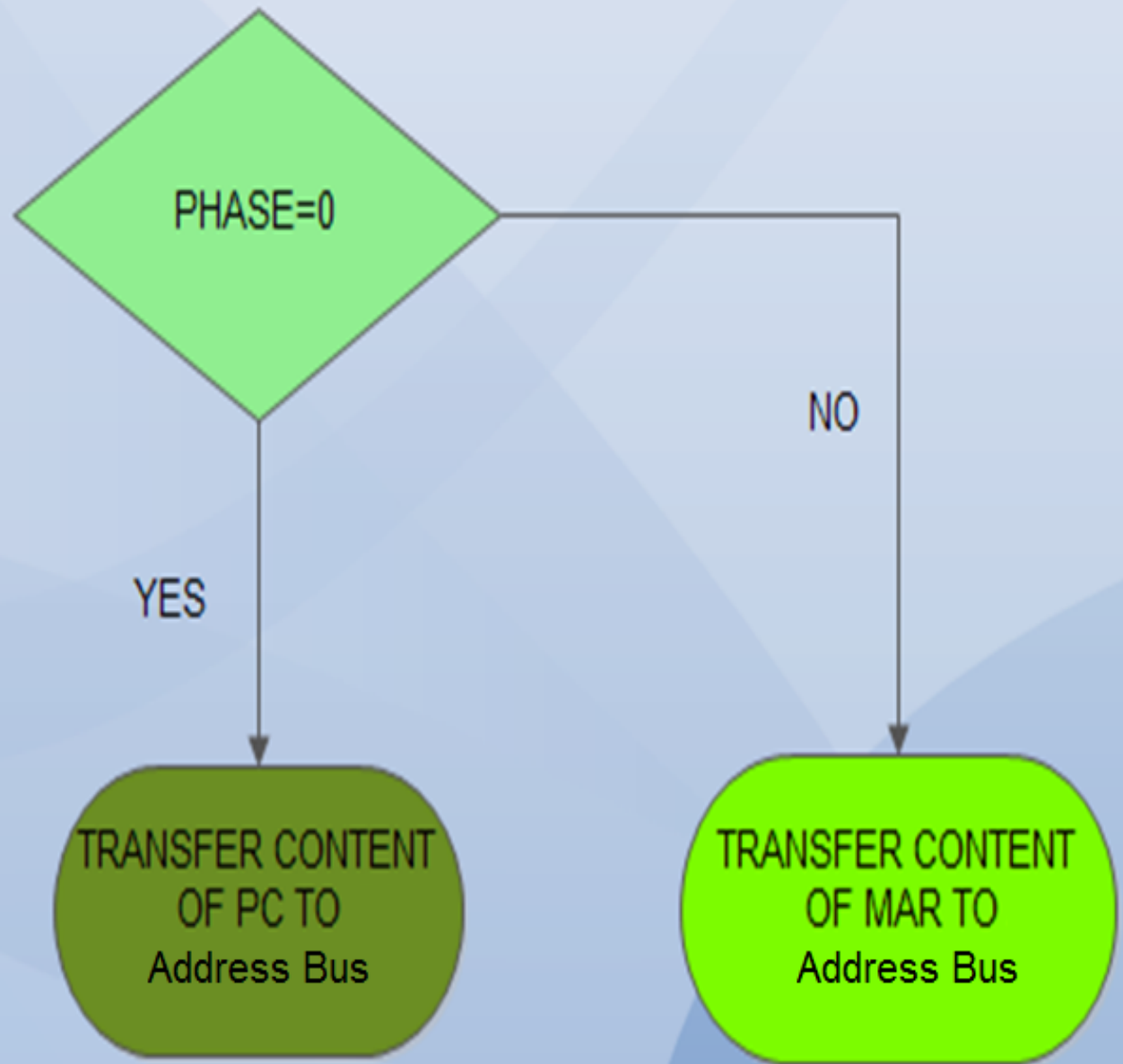
READ signal generation



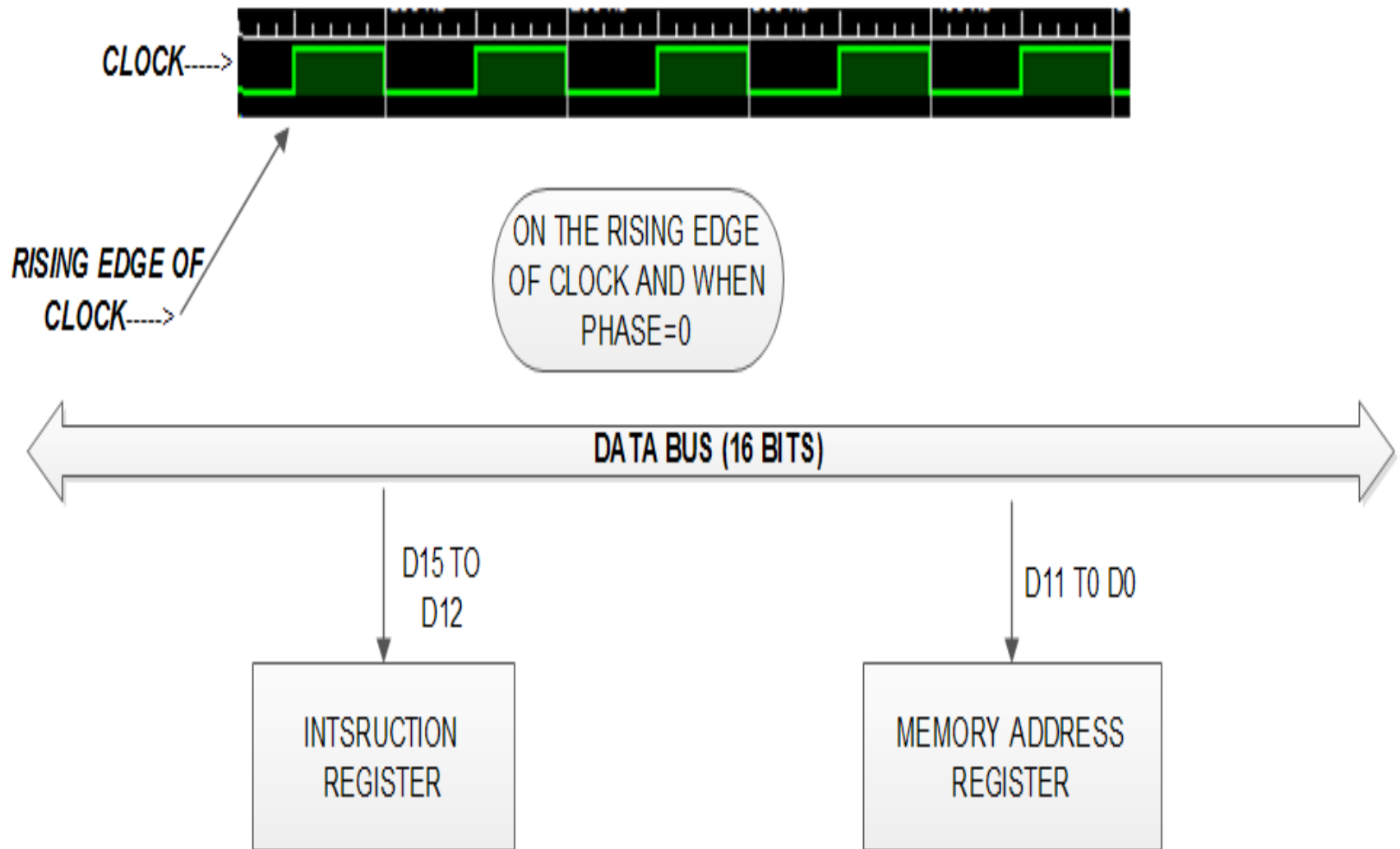
WRITE signal generation



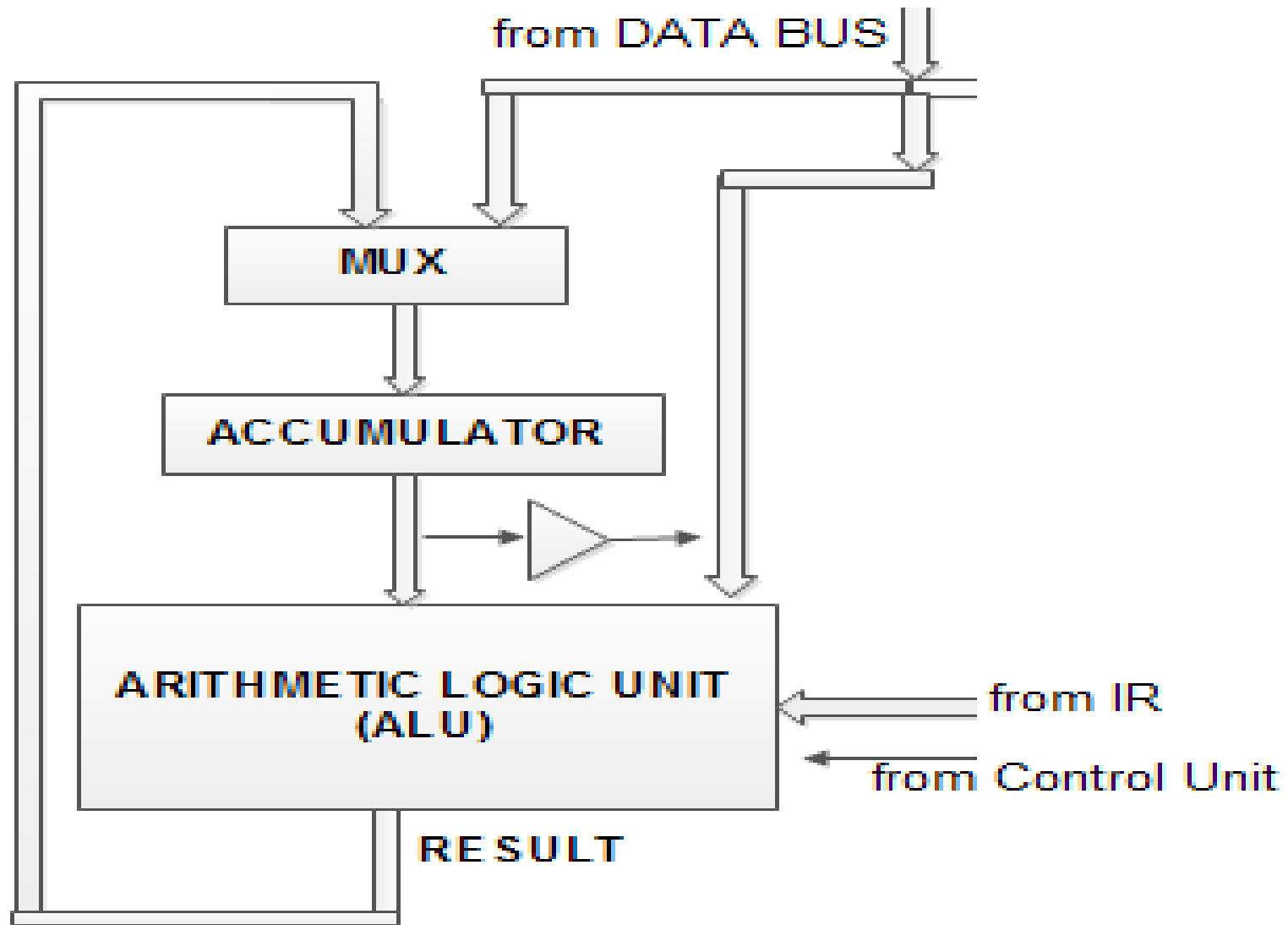
NOTE: PHASE=0
MEANS OPCODE
FETCH CYCLE
PHASE=1 MEANS
EXECUTION CYCLE



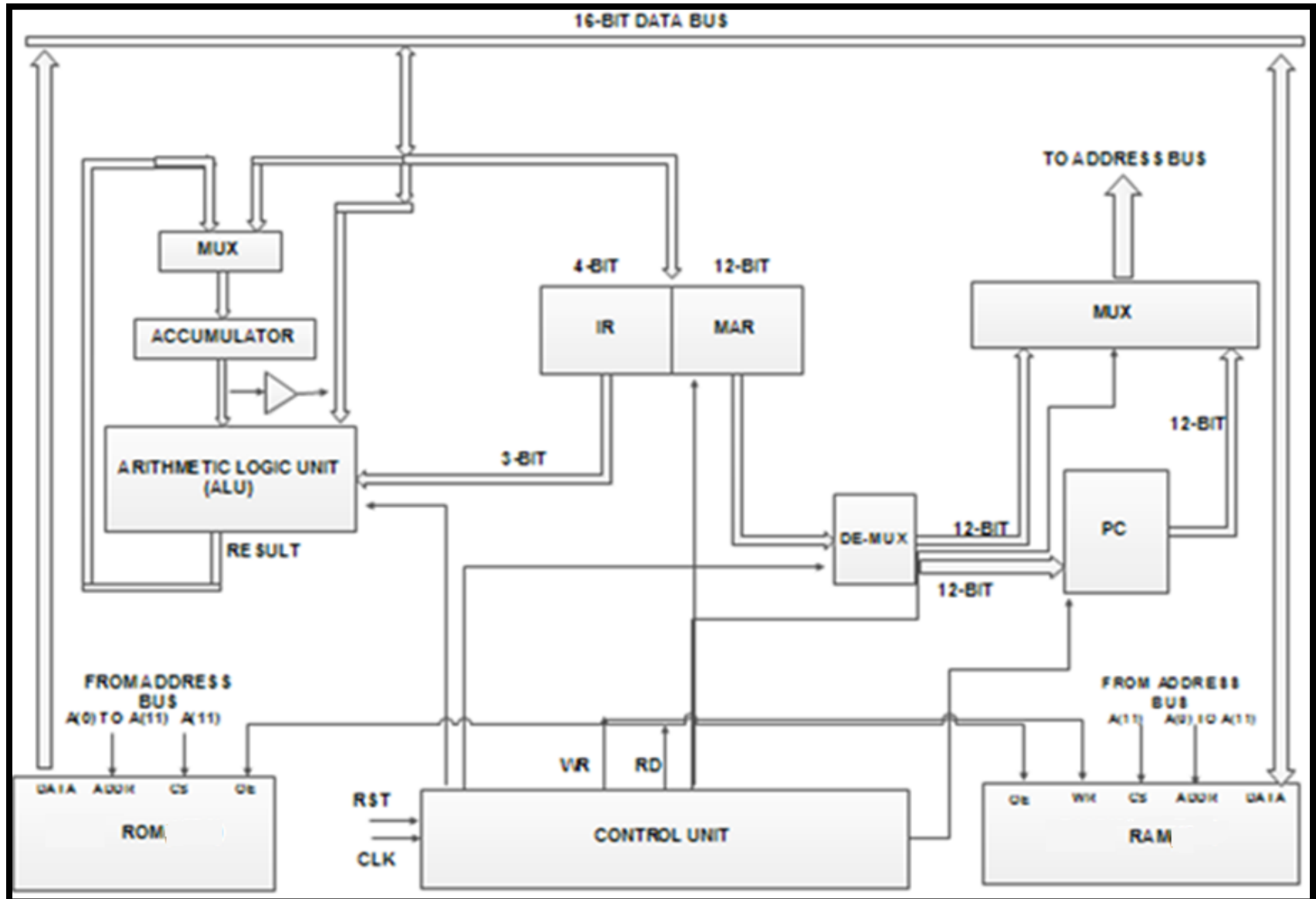
Op-Code and Operand Fetch



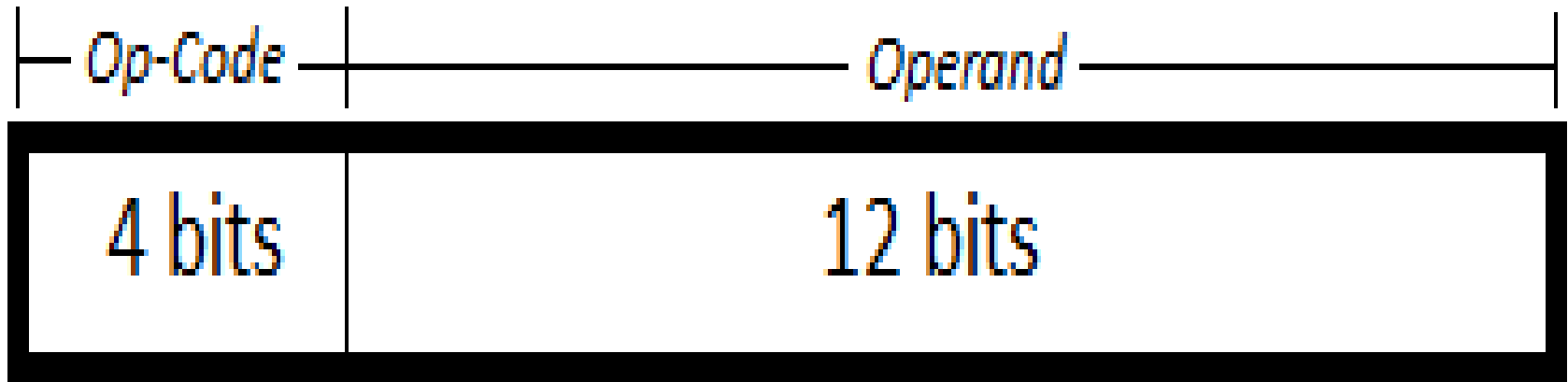
Operation Model of ALU



Schematic diagram of the CPU



Instruction

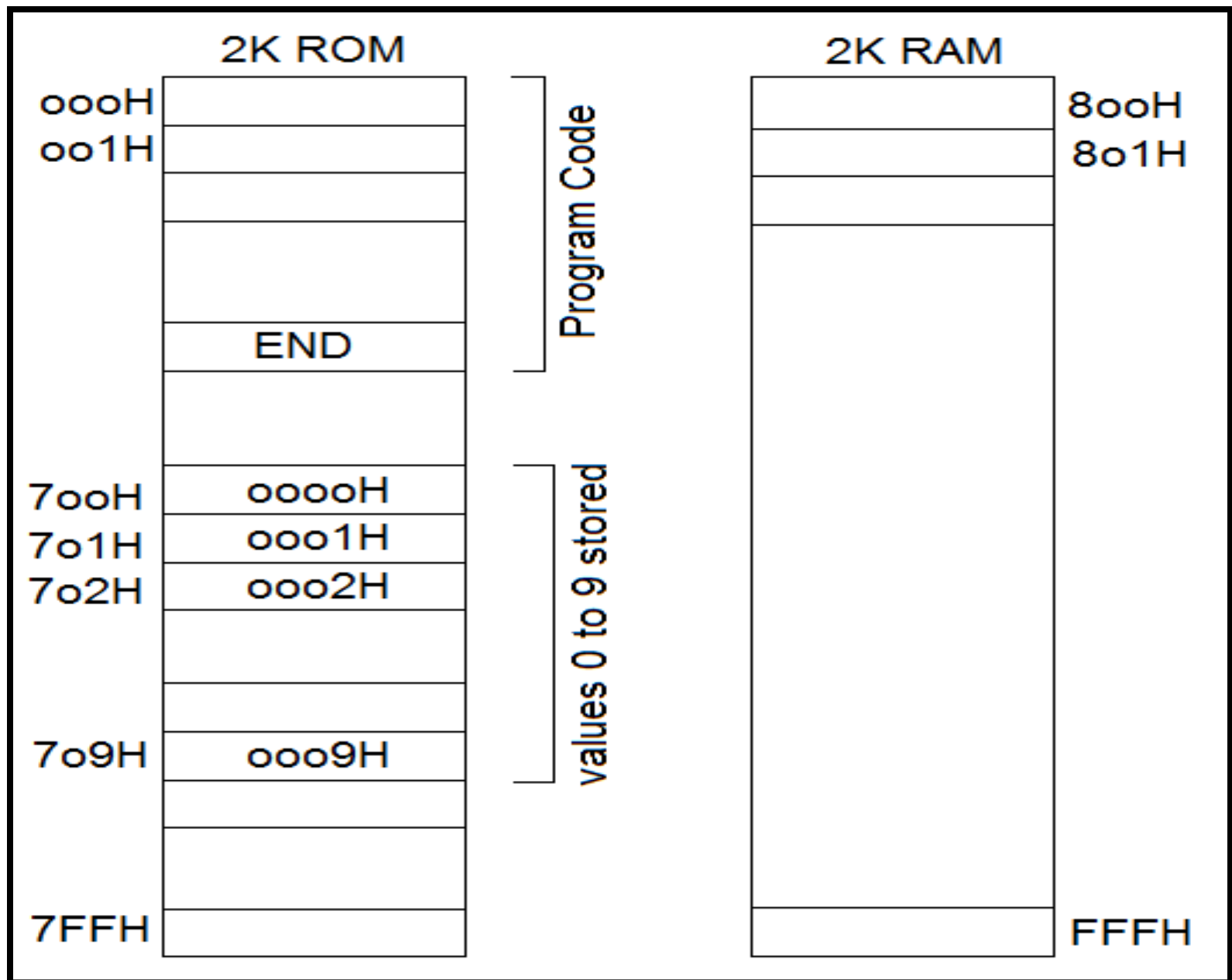


Programming the Processor

The program code consists of instructions of 16 bit, which are placed one by one in ROM locations starting from 000h. The PC of the processor accesses the ROM for the Op-code starting from location 000h till the END instruction is encountered.

In the ROM memory, values 0 to 9 are permanently loaded in locations 700h to 709h respectively, for testing purpose.

Program code starts from location 000h.



TEST OPERATION 1: $((1+2) \times (4/2)) - 2$

Expected Output: 4

TEST CODE:

LDA 701 - load value '1' at ROM loc 701h to Acc

ADD 702 - add value '2' at ROM loc 702h to content of Acc

STA 800 - store result in Acc to RAM loc 800h

LDA 704 - load value '4' at ROM loc 704h to Acc

DIV 702 - divide content of Acc by value '2'

STA 801 - store result in Acc to RAM loc 801h

LDA 800 - load value in RAM loc 800h to Acc

MUL 801 - multiply content of Acc by value at RAM loc 801h

STA 802 - store result in Acc to RAM loc 802h

LDA 802 - load value in RAM loc 802h to Acc

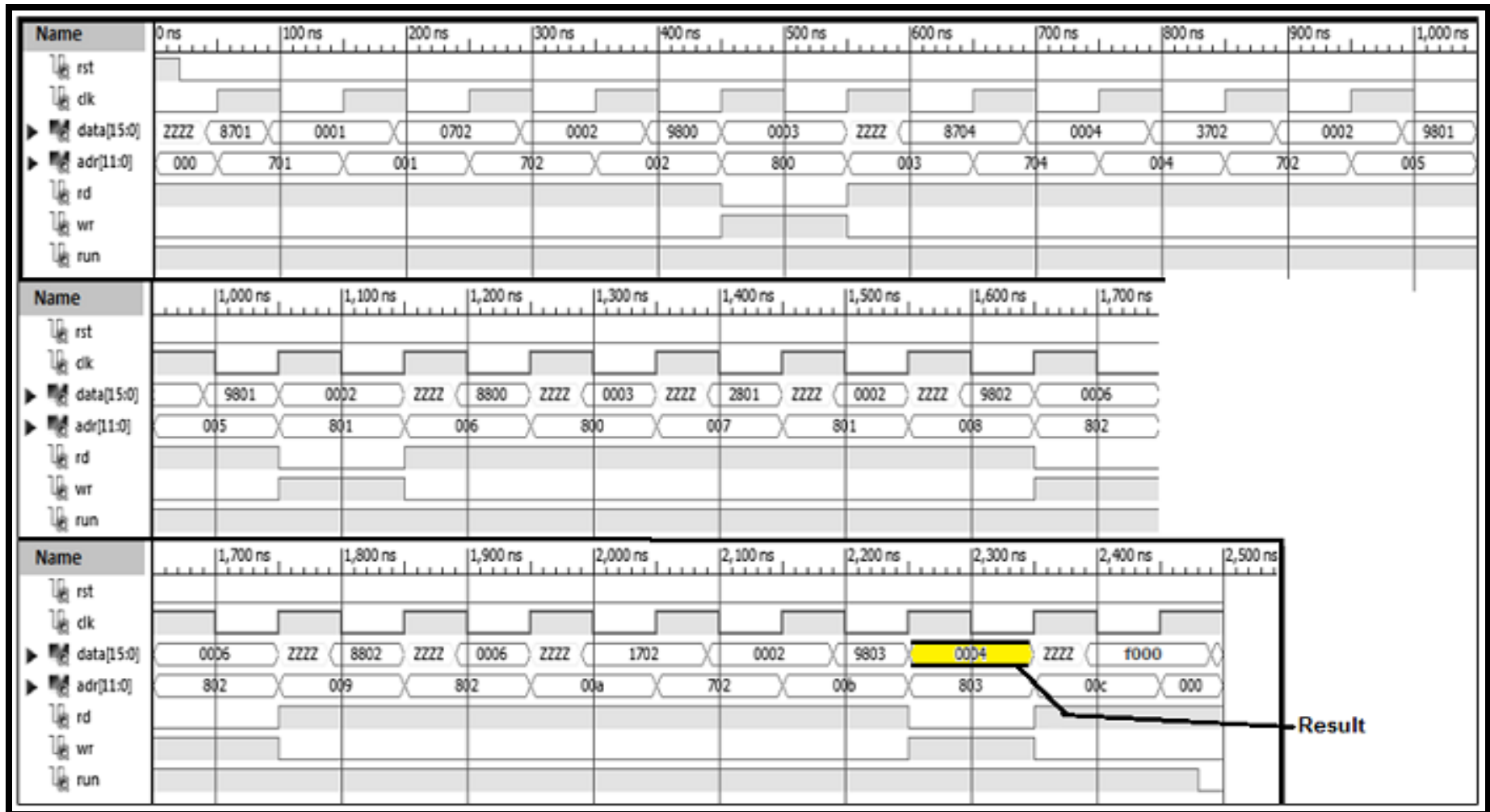
SUB 702 - subtract value '2' at ROM loc 702h from Acc content

STA 803 - store result in Acc to RAM loc 803h

END - jump to ROM loc 000h i.e. start of program

The hex code in ROM from 000h :

*8701, 0702, 9800, 8704, 3702, 9801, 8800, 2801, 9802, 8802,
1702, 9803, F000*



TEST OPERATION 2: (2 OR 7) AND (6 XOR 9)

Expected Output: 7

TEST CODE:

LDA 702 - load value '2' at ROM loc 702h to Acc

OR 707 - OR value '7' at ROM loc 707h to content of Acc

STA 800 - store result in Acc to RAM loc 800h

LDA 706 - load value '6' at ROM loc 706h to Acc

XOR 709 - XOR value '9' at ROM loc 709h to content of Acc

STA 801 - store result in Acc to RAM loc 801h

LDA 800 - load value in RAM loc 800h to Acc

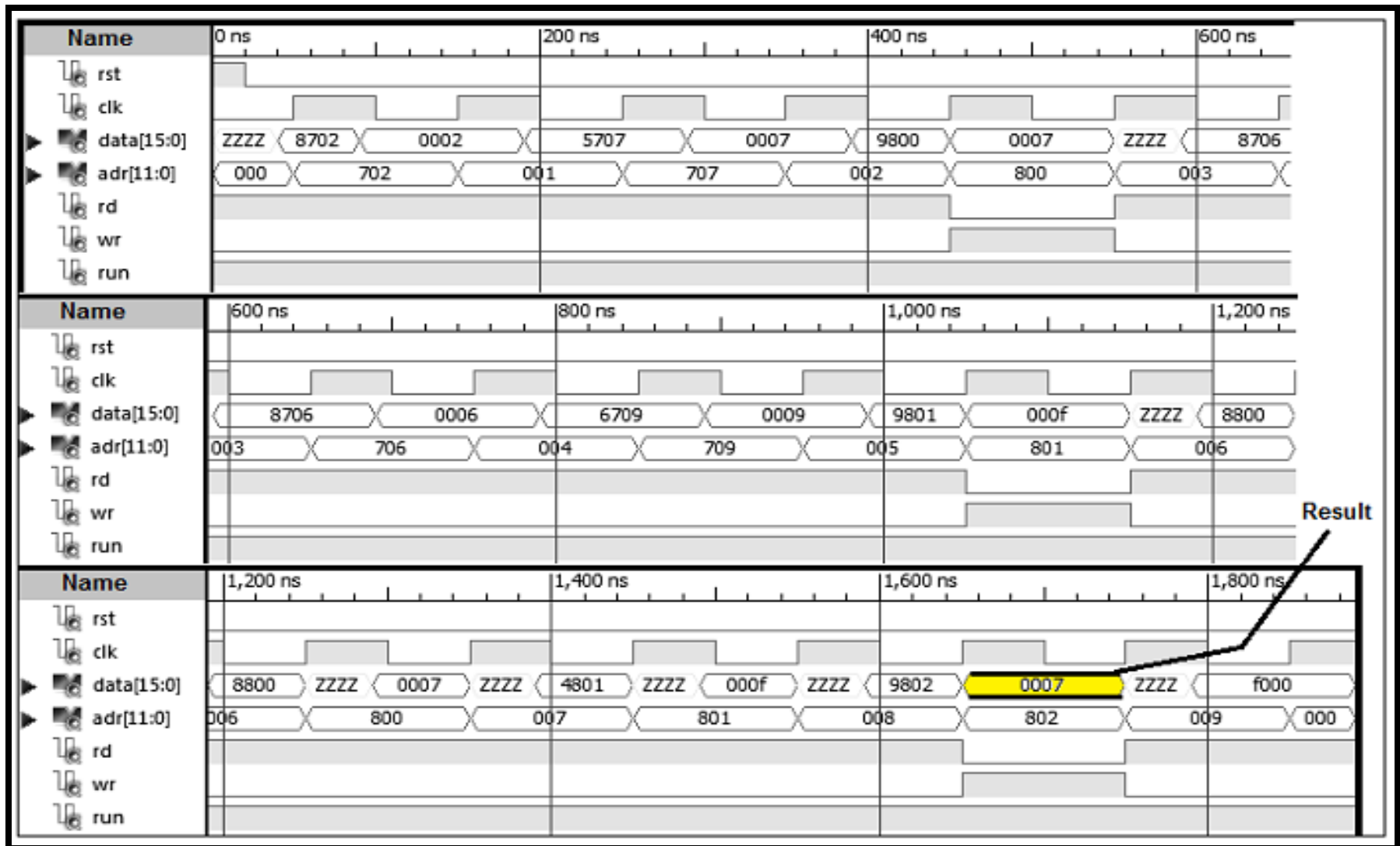
AND 801 - AND content of Acc by value at RAM loc 801h

STA 802 - store result in Acc to RAM loc 802h

END - jump to ROM loc 000h i.e. start of program

The hex code in ROM from 000h :

8702, 5707, 9800, 8706, 6709, 9801, 8800, 4801, 9802, F000



CPU Project Status (04/28/2014 - 01:05:19)

Project File:	CPU16_TB.xise	Parser Errors:	No Errors
Module Name:	CPU	Implementation State:	Synthesized
Target Device:	xc3s100e-4tq144	• Errors:	No Errors
Product Version:	ISE 14.6	• Warnings:	2 Warnings (2 new)
Design Goal:	Balanced	• Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	
Environment:	System Settings	• Final Timing Score:	

Device Utilization Summary (estimated values)

[\[-\]](#)

Logic Utilization	Used	Available	Utilization
Number of Slices	269	960	28%
Number of Slice Flip Flops	45	1920	2%
Number of 4 input LUTs	513	1920	26%
Number of bonded IOBs	32	108	29%
Number of MULT18X18SIOs	1	4	25%
Number of GCLKs	1	24	4%

PROCESSOR	INSTRUCTIONS	GATES
8085 (CISC)	246	6500
80286	20 K	134 K
Our Processor	14	10 K (including memory)

Future Research :

The CPU working on 16 bit data line and 12 bit address line is totally modifiable. The future target is to exploit the Harvard architecture for pipelining extensively using more instructions.

ACKNOWLEDGMENT

We express our deepest gratitude to our project supervisor *Mr. Sohan Ghorai*

Assistant professor, Electronics and Communication Engineering,
Narula Institute of Technology, Kolkata

We thank *Prof. (Dr.) Maitreyi Roy Kanjilal*

Head of the Department, Electronics & Communication Engineering,
Narula Institute of Technology, Kolkata