Microprocessor & Assembly Language Programming

Dr. Jamuna Kanta Sing

Department of Computer Science and Engineering
Jadavpur University
188, Raja S. C. Mallik Road
Kolkata 700032, India
Email: jksing@ieee.org

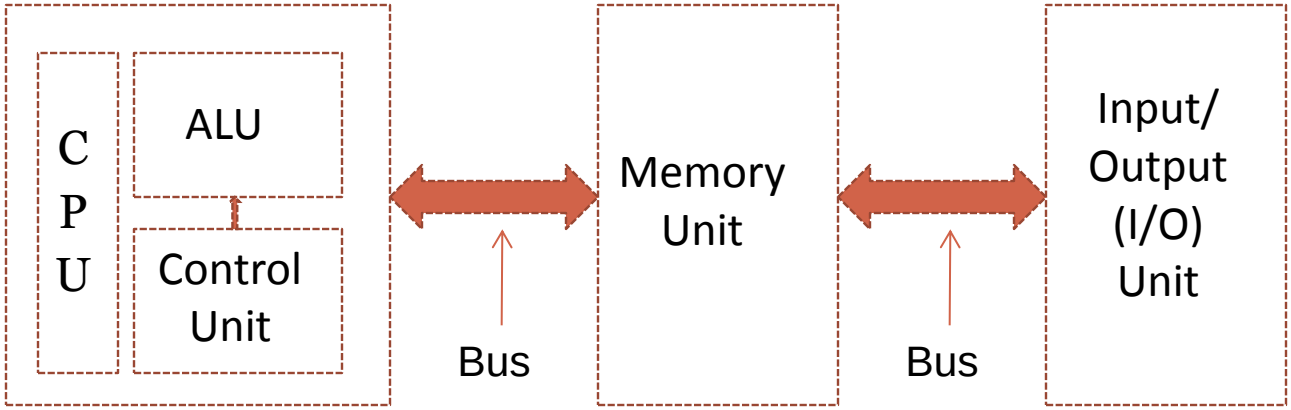
Course Contents

1) Microprocessor Structure and Operations
2) Classification of Microprocessors
3) Intel 8085 Microprocessor
4) Instruction Set of 8085 and Programming
5) Memory Organization
6) Memory Interfacing Techniques
7) Interrupts
8) Input-Output Devices
9) Interfacing Input-Output Devices
10) Serial Transmission
11) Operating Systems for Microprocessor

Text Books: (i) Lecture Notes

(ii) Ramesh Goankar: Microprocessor Architecture,

Programming and Applications with the 8085, Printice Hall.



Basic Functional Units of a Computer Systems

1. I/O Unit
   Allows the computer system to interact with the outside world by moving data into and out of the system.
   - Input Unit: Accepts coded information as data.
   - Output Unit: Sends processed result to the outside world.

2. Memory Unit: Stores programmed data.

Memory	
Main Memory	Secondary Memory
(i) Fast (ii) Expensive (iii) Low capacity (iv) Connects directly to the CPU (v) Program must reside during execution Examples: RAM	(i) Slower (ii) Cheaper (iii) Large capacity (iv) Not connected directly to the CPU Example: HDD, Floppy Disk, CD-ROMs, etc.

3. Central Processing Unit (CPU)

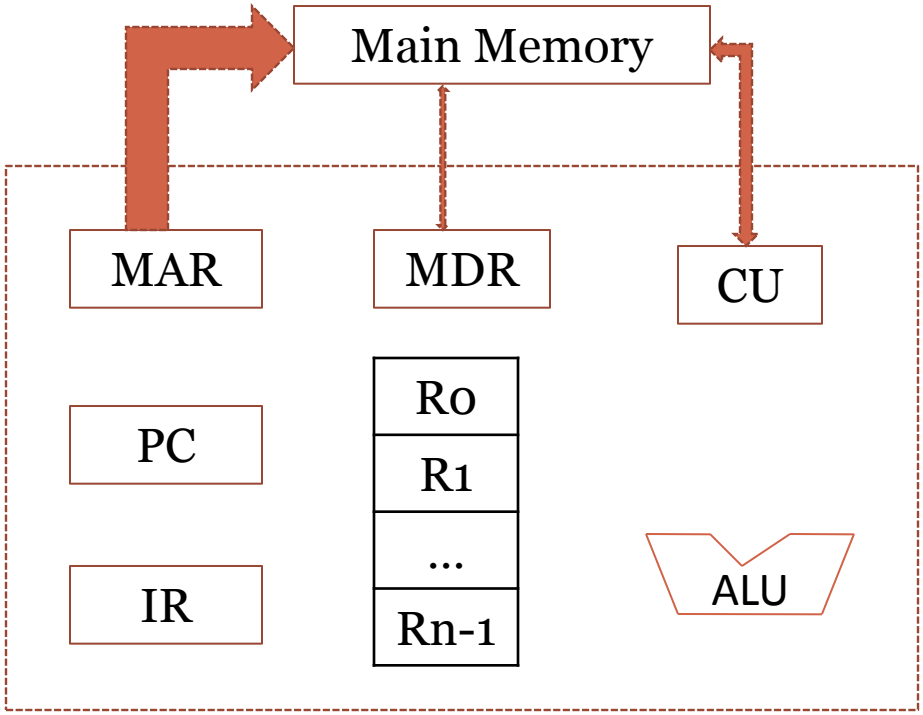
(a) Arithmetic and Logical Unit (ALU)

(i) Dedicated unit to perform arithmetic and logical operations.

(ii) Processor holds a number of high speed registers to hold temporary data.

(b) Control Unit (CU)

(i) Controls the operations all units by sending appropriate signals to them.



Interconnection between memory and CPU

Memory Address Register (MAR): Holds the address of the memory location to or from which data to be transferred.

Program counter (PC): Keeps the address of the instruction currently being executed.

Memory Data Register (MDR): Contains the data to be written into or read out of the addressed memory location.

Instruction Register (IR): Holds the instruction that is currently being executed.

How does the Microprocessor work?

1. The Instructions are stored sequentially in the memory.
2. The Microprocessor *fetches* one instruction at a time, *decodes* and *executes*.
3. The sequence of fetch, decode and execute is continued until the Microprocessor comes across an instruction to *stop*.
4. During the entire process, microprocessor uses the system bus to fetch instruction and data from the memory.
5. It uses registers within CPU to hold data temporarily and performs arithmetic and logical functions using ALU.

Classification of Microprocessors

MPU classification based on word length:

Word: It is defined as number of bits the microprocessor recognizes and process at a time. It ranges from 4-bit to 64-bit.

4-bit MPU:	Intel 4004 (1971)
8-bit MPU:	Intel 8008, 8080, 8085, Motorola 6800Zilog Z80 (1972-1976)
32-bit MPU:	Intel 8086, 8088, 80486 (1979-1989)
64-bit MPU:	Intel Pentium (1993), Pentium II (1997) – Pentium 4 (2000) Core 2 Duo

Microprocessor

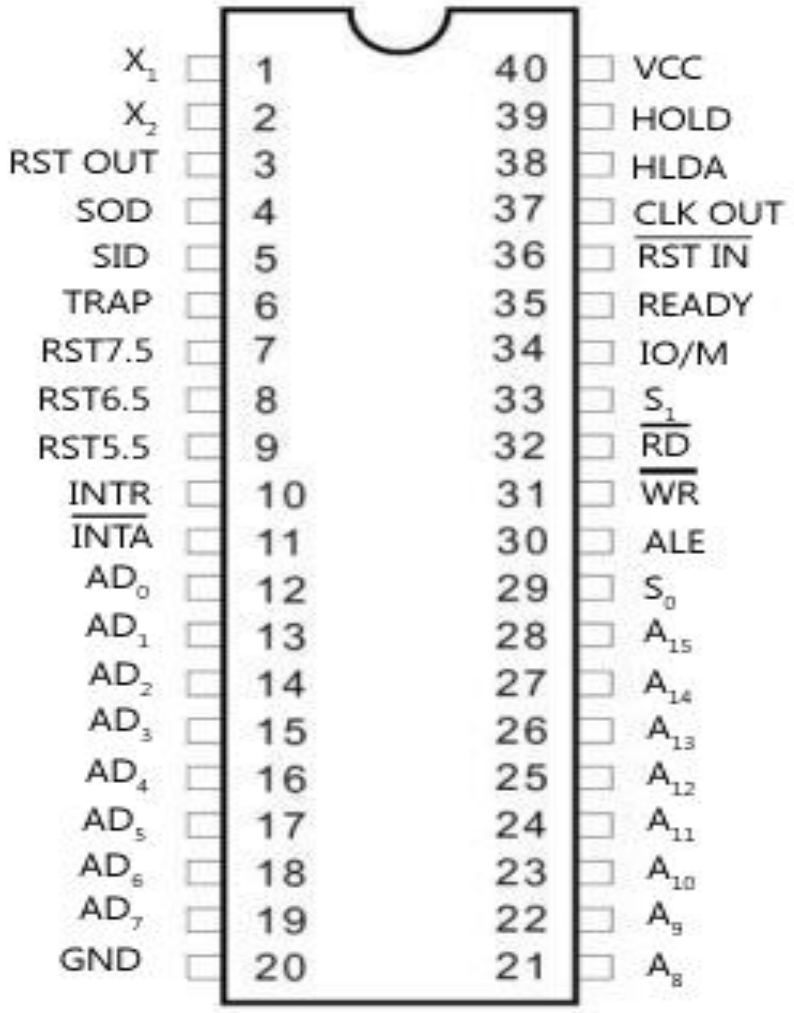
Definition:

- A programmable device to control processes or devices.
- A processing unit of a computer.
- A CPU of a microprocessor-based system.

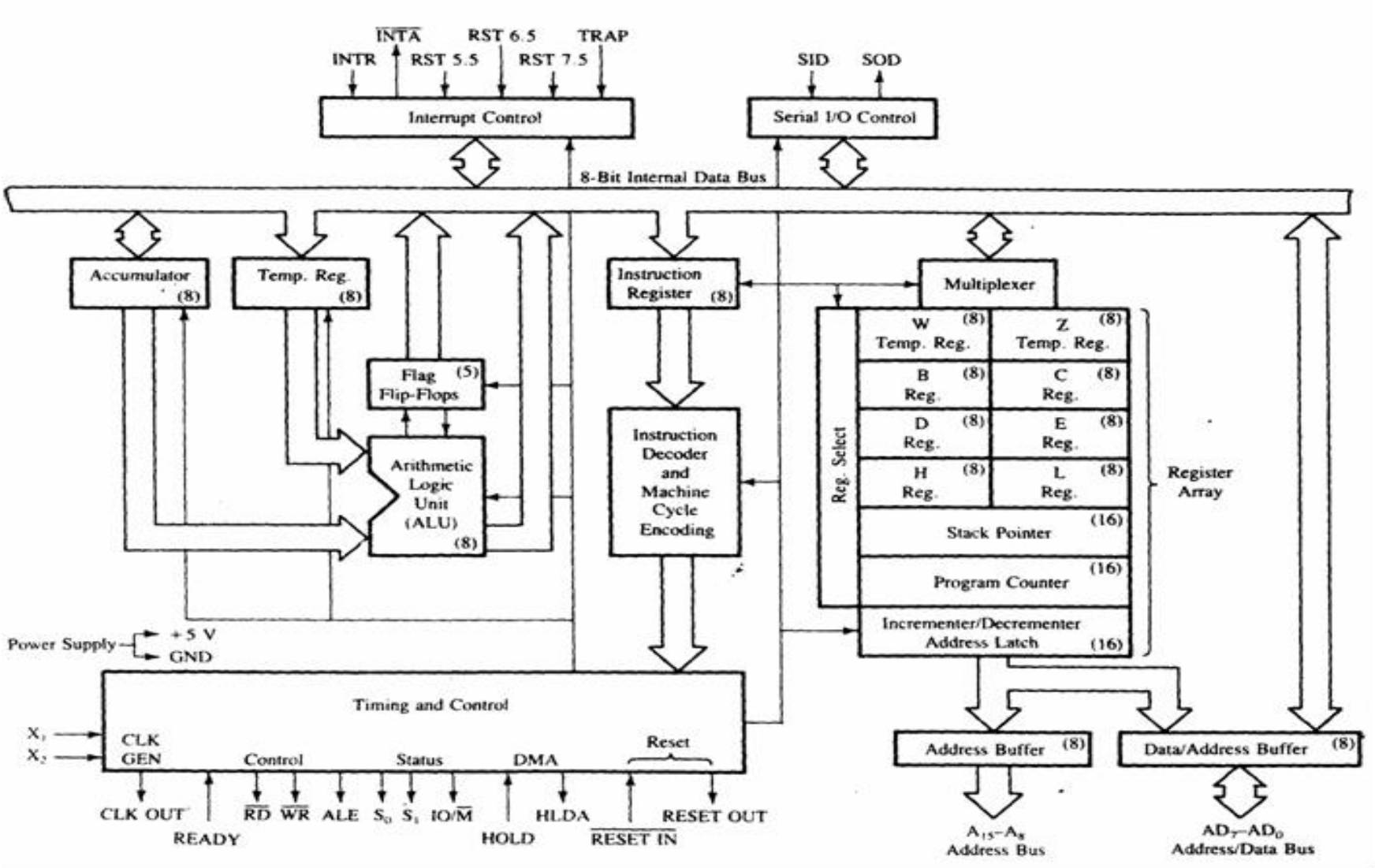
Microprocessor communicates and operates on bits – 0 and 1

4-bit string:	Nibble
8-bit string:	Byte
One or more byte:	Word

Pin Diagram of 8085 Microprocessor



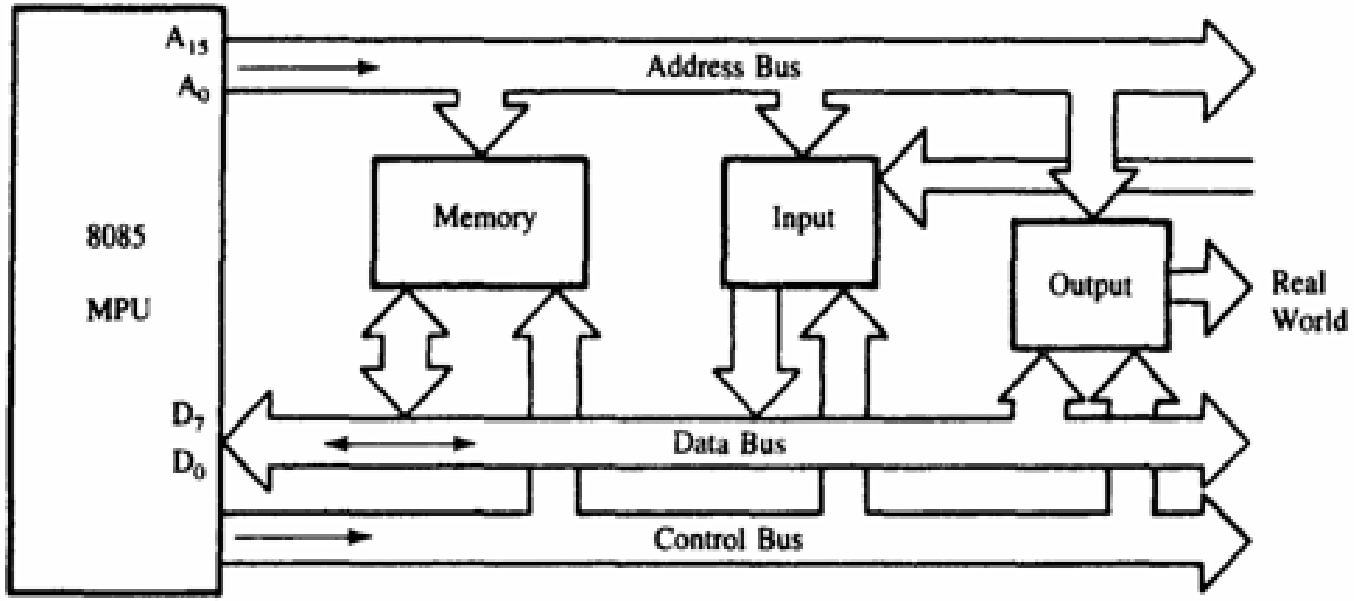
The functional block diagram or architecture of 8085 Microprocessor



8085 Bus Structure

Address Bus:

- The address bus is a group of 16 lines generally identified as A0 to A15.
- The address bus is unidirectional: bits flow in one direction-from the MPU to peripheral devices.
- The MPU uses the address bus to perform the first function: identifying a peripheral or a memory location.



Data Bus:

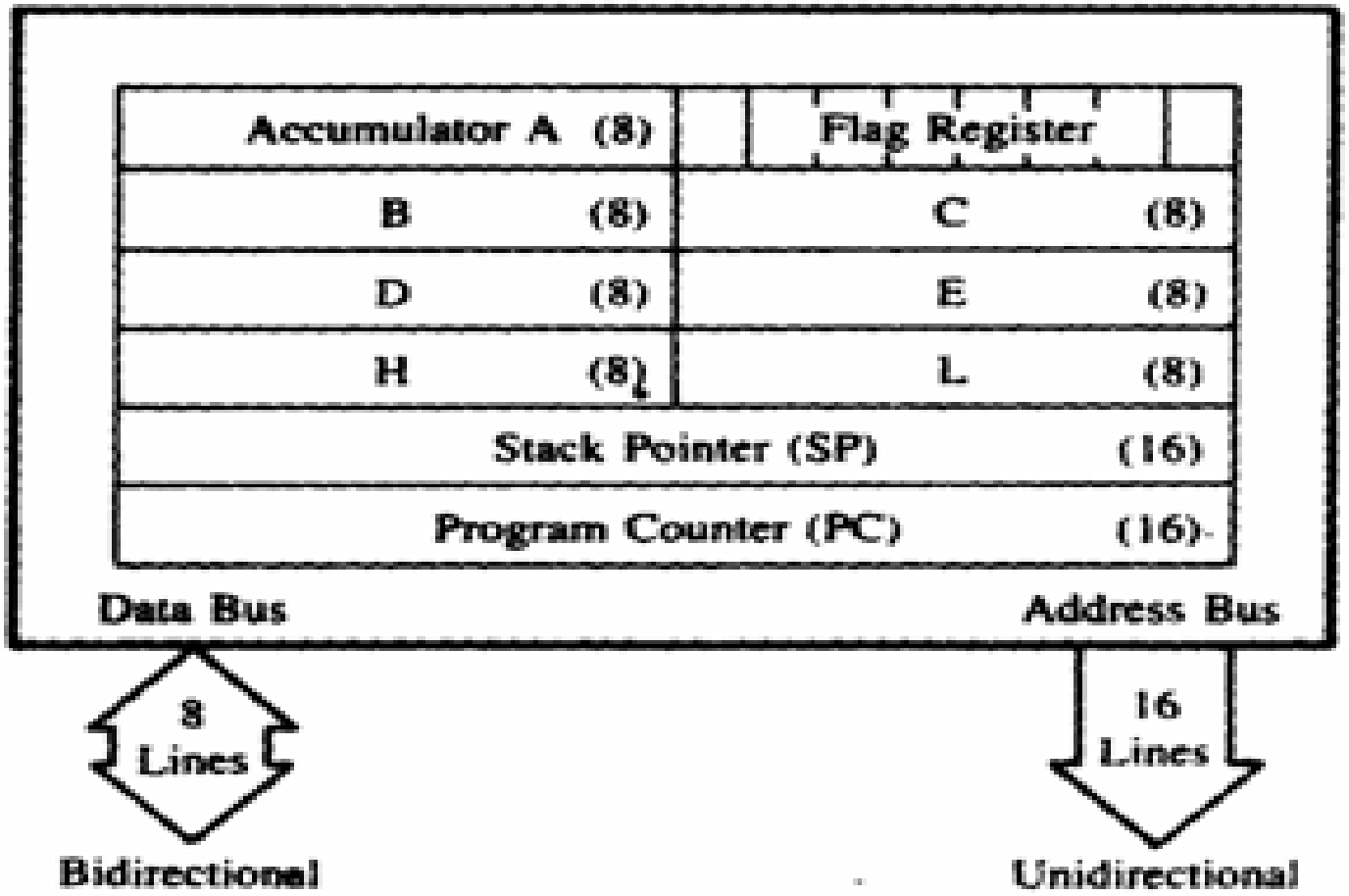
- The data bus is a group of eight lines used for data flow.
- These lines are bi-directional - data flow in both directions between the MPU and memory and peripheral devices.
- The MPU uses the data bus to perform the second function: transferring binary information.
- The eight data lines enable the MPU to manipulate 8-bit data ranging from 00 to FF (28 = 256 numbers).
- The largest number that can appear on the data bus is 11111111.

Control Bus:

- The control bus carries synchronization signals and providing timing signals.
- The MPU generates specific control signals for every operation it performs. These signals are used to identify a device type with which the MPU wants to communicate.

Registers of 8085:

- The 8085 have six general-purpose registers to store 8-bit data during program execution.
- These registers are identified as B, C, D, E, H, and L.
- They can be combined as register pairs-BC, DE, and HL-to perform some 16-bit operations.


Accumulator A (8)
Flag Register
B (8)
C (8)
D (8)
E (8)
H (8)
L (8)
Stack Pointer (SP) (16)
Program Counter (PC) (16)
Data Bus
Address Bus
8 Lines
16 Lines
Bidirectional
Unidirectional

Accumulator (A):

- The accumulator is an 8-bit register that is part of the arithmetic/logic unit (ALU).
- This register is used to store 8-bit data and to perform arithmetic and logical operations.
- The result of an operation is stored in the accumulator.

Flags:

- The ALU includes five flip-flops that are set or reset according to the result of an operation.
- The microprocessor uses the flags for testing the data conditions.
- They are Zero (Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags. The most commonly used flags are Sign, Zero, and Carry.

The bit position for the flags in flag register is as follows:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

1. **Sign Flag (S):**
   - After execution of any arithmetic and logical operation, if D7 of the result is 1, the sign flag is set. Otherwise it is reset.
   - D7 is reserved for indicating the sign; the remaining is the magnitude of number.
   - If D7 is 1, the number will be viewed as negative number. If D7 is 0, the number will be viewed as positive number.

2. **Zero Flag (z):**
   - If the result of arithmetic and logical operation is zero, then zero flag is set otherwise it is reset.

3. Auxiliary Carry Flag (AC):

- If D3 generates any carry when doing any arithmetic and logical operation, this flag is set. Otherwise it is reset.

4. Parity Flag (P):

- If the result of arithmetic and logical operation contains even number of 1's then this flag will be set and if it is odd number of 1's it will be reset.

5. Carry Flag (CY):

- If any arithmetic and logical operation result any carry then carry flag is set otherwise it is reset.

Arithmetic and Logic Unit (ALU):

- It is used to perform the arithmetic operations like addition, subtraction, multiplication, division, increment and decrement and logical operations like AND, OR and EX-OR.
- It receives the data from accumulator and registers.
- According to the result it set or reset the flags.

Program Counter (PC):

- This 16-bit register sequencing the execution of instructions.
- It is a memory pointer. Memory locations have 16-bit addresses, and that is why this is a 16-bit register.
- The function of the program counter is to point to the memory address of the next instruction to be executed.
- When an opcode is being fetched, the program counter is incremented by one to point to the next memory location.

Stack Pointer (SP):

- The stack pointer is also a 16-bit register used as a memory pointer.
- It points to a memory location in R/W memory, called the stack.
- The beginning of the stack is defined by loading a 16-bit address in the stack pointer (register).

Temporary Register: It is used to hold the data during the arithmetic and logical operations.

Instruction Register: When an instruction is fetched from the memory, it is loaded in the instruction register.

Instruction Decoder: It gets the instruction from the instruction register and decodes the instruction. It identifies the instruction to be performed.

Serial I/O Control: It has two control signals named SID and SOD for serial data transmission.

Timing and Control unit:

- It has three control signals ALE, RD (Active low) and WR (Active low) and three status signals IO/M(Active low), S0 and S1.

- ALE is used for provide control signal to synchronize the components of microprocessor and timing for instruction to perform the operation.

- RD (Active low) and WR (Active low) are used to indicate whether the operation is reading the data from memory or writing the data into memory respectively.

IO/M (Active low) is used to indicate whether the operation is belongs to the memory or peripherals.

If,

IO/M (Active Low)	S1	S2	Data Bus Status(Output)
0	0	0	Halt
0	0	1	Memory WRITE
0	1	0	Memory READ
1	0	1	IO WRITE
1	1	0	IO READ
0	1	1	Opcode fetch
1	1	1	Interrupt acknowledge

Interrupt Control Unit:

It receives hardware interrupt signals and sends an acknowledgement for receiving the interrupt signal.

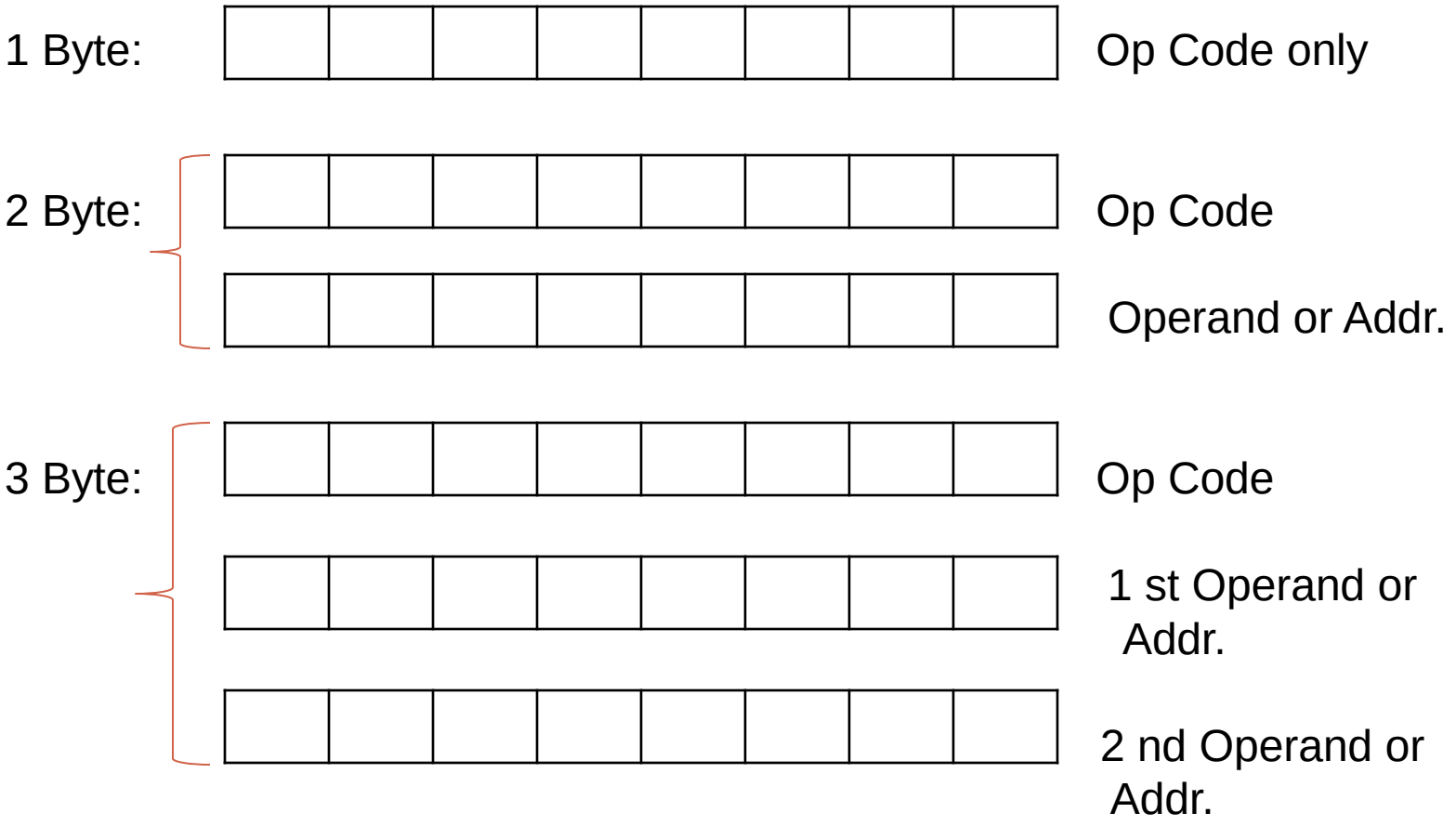
Instruction Set

Format:

Mnemonics (Op Code)	Operands

- Types of Instructions:
  - Based on:
    I. Operation: Add, Subtract, etc.
    II. Size: 1 Byte, 2 byte and 3 byte

Instructions based on sizes



Storing of Instruction in Main Memory

Address	Content	Instn. Type
0000_H	...	
0001_H	...	
$000i_H$	Op Code	1 byte instruction
	...	
$000j_H$	Op Code	2 byte instruction
$000j+1_H$	Operand	
	...	
$000k_H$	Op Code	3 byte instruction
$000k+1_H$	Operand $_L$	
$000k+2_H$	Operand $_H$	
	...	

- Addressing Modes: It refers to the way in which operands are specified in an instruction.

1. **Immediate Addressing**
2. **Direct Addressing**
3. **Register Addressing**
4. **Register Indirect Addressing**
5. **Implied Addressing**

1. Immediate Addressing:

In immediate addressing mode, the data is specified in the instruction itself. The data will be a part of the program instruction.

Example:

MVI B, $3E_H$; B = $3E_H$;

LXI SP, 2700_H; SP = 2700_H, SPL = 00_H, SPH = 27_H

2. Direct Addressing:

In direct addressing mode, the address of the data is specified in the instruction. The data will be in memory. In this addressing mode, the program instructions and data can be stored in different memory.

Example:

LDA 1050_H ; A = M[1050_H];

LHLD 3000_H ; L = M[3000_H], H = M[3001_H]

3. Register Addressing:

In register addressing mode, the instruction specifies the name of the register in which the data is available.

Example:

```
MOV A, B ;      A = B
ADD C;          A = A +C
```

4. Register Indirect Addressing:

The instruction specifies the name of the register in which the address of the data is available. Here the data will be in memory and the address will be in the register pair.

Example:

MOV A, M ; A = M[HL]

LDAX B; A = M[BC]

5. Implied Addressing:

The instruction itself specifies the data to be operated.

Example:

CMA - Complement the content of accumulator;

RAL – Rotate Acc. left by 1 bit

The 8085 instruction set can be classified into the following five functional headings.

1. DATA TRANSFER INSTRUCTIONS:

It includes the instructions that move (copies) data between registers or between memory locations and registers. In all data transfer operations the content of source register is not altered. Hence the data transfer is copying operation.

Ex: (1) MOV A, B ; A = B
(2) MVI C, 45H; C = 45H
(3) LXI H, 2005H; H = 20H, L = 05H
(4) MOV A, M; A = M[HL] (5) LDA 2050H; A = M[2050H]
(6) STA 2010H; M[2010H] = A

2. ARITHMETIC INSTRUCTIONS:

Includes the instructions, which performs the addition, subtraction, increment or decrement operations. The flag conditions are altered after execution of an instruction in this group.

Ex:

(1) ADD B ; A = A+B

(2) ADC B; A = A + B + CY

(3) ADD M; A = A + M[HL]

(4) ADC M; A = A + M[HL] +CY

(5) ADI 05H; A = A + 05H

(6) SUB B; A = A – B

(7) SUB M; A = A – M[HL]

(8) SUI 05H; A = A – 05H

(9) INR B; B = B + 1 (10) INX B; BC = BC + 1

(11) DCR B; B = B – 1; (12) DCX B; BC = BC - 1

3. LOGICAL INSTRUCTIONS:

The instructions which performs the logical operations like AND, OR, EXCLUSIVE- OR, complement, compare and rotate instructions are grouped under this heading. The flag conditions are altered after execution of an instruction in this group.

Ex: (1) ORA A
(2) ANI B, 01H

4. BRANCHING INSTRUCTIONS:

The instructions that are used to transfer the program control from one memory location to another memory location are grouped under this heading.

Ex: (1) CALL 2050H (Subroutine call)
(2) JMP 4100H (Unconditional jump)
(3) JNZ 2040H (Conditional jump)

5. MACHINE CONTROL INSTRUCTIONS:

It includes the instructions related to interrupts and the instruction used to stop the program execution.

Ex: (1) NOP
(2) HLT
(3) RST 1

6. STACK INSTRUCTIONS:

Used for stack operations.

Ex. (1) PUSH B; M[SP--] = B, M[SP--] = C
(2) POP B; C = M[++SP], B = M[++SP]

7. I/O INSTRUCTIONS:

Used for i/o operations.

Ex. (1) IN 45H
(2) OUT 55H

Instruction Execution

Execution of an instruction consists of two separate cycles: (i) Fetch cycle and (ii) Execution cycle.

(i) Fetch cycle:

a) Instruction addressed by the content of program counter (PC) is loaded in instruction register (IR) from memory. MAR = [PC], MDR = M[MAR], IR = [MDR];

b) PC is incremented. PC = PC + 1;

c) Fetching of operand data initiated.

(ii) Execution cycle:

a) Instruction in IR is executed and specified operation is performed by CPU.

Programming in 8085

Example 1: Add two numbers, 05 and 03

```
MVI   A,  05H      ; A = 05
MVI   B,  03H      ; B = 03
ADD   B            ; A = A + B
STA       2050H    ; M[2050] = A
RST 5
```

m/m loc.	content
2500	3E
2501	05
2502	06
2503	03
2504	80
2505	32
2506	50
2507	20
2508	EF

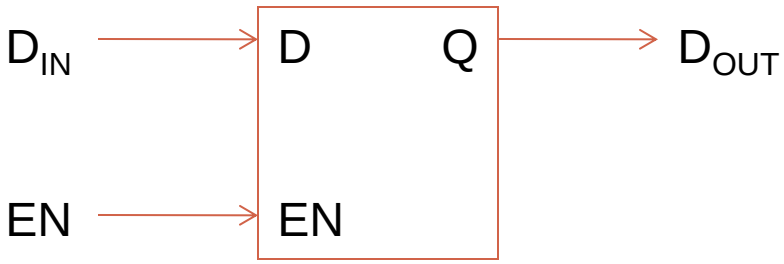
Memory locations and Addresses

Address	Data/Instruction					Word
0						0
1						1
…			…			…
i	A_{n-1}	A_{n-2}	…	A_1	A_0	i
…			…			…
2^k-1						2^k-1

Memory

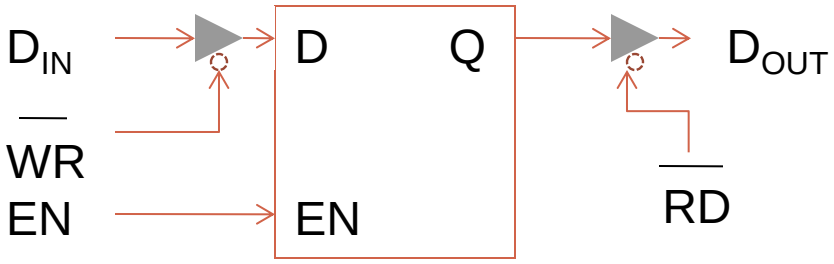
1. Read/Write (R/W) Memory
   - Made of Registers
     - Registers made of Flip-Flops (F/Fs)

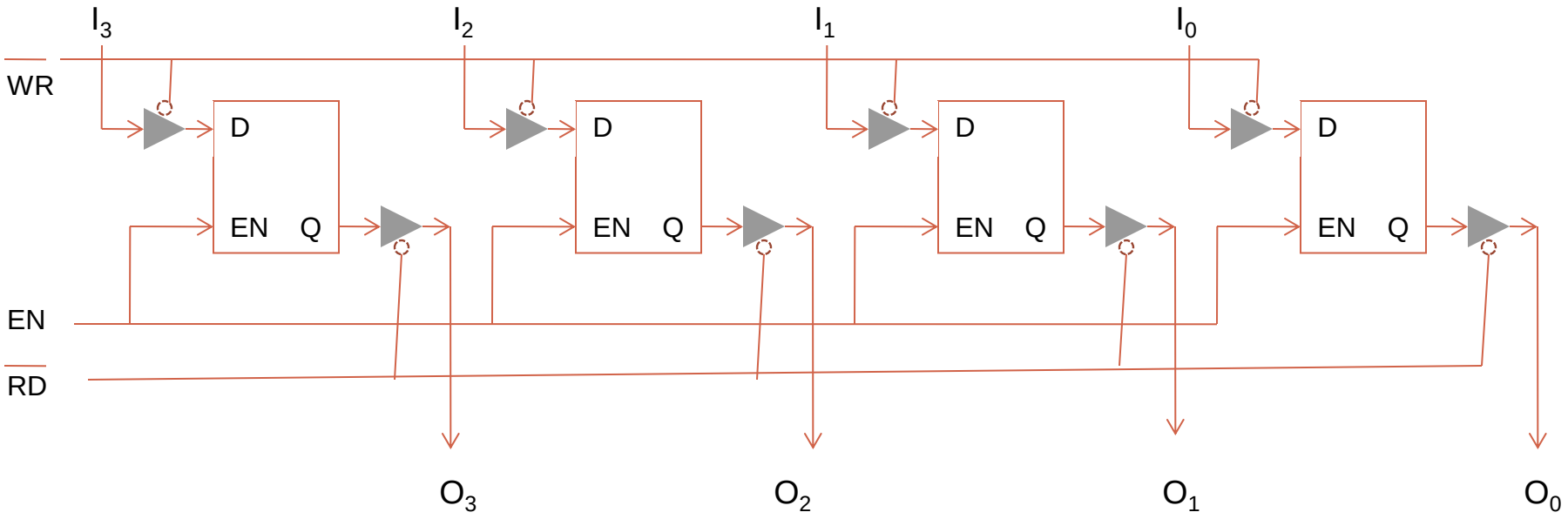
1. Read Only Memory (ROM)
   - Made of Registers
     - Registers made of diodes

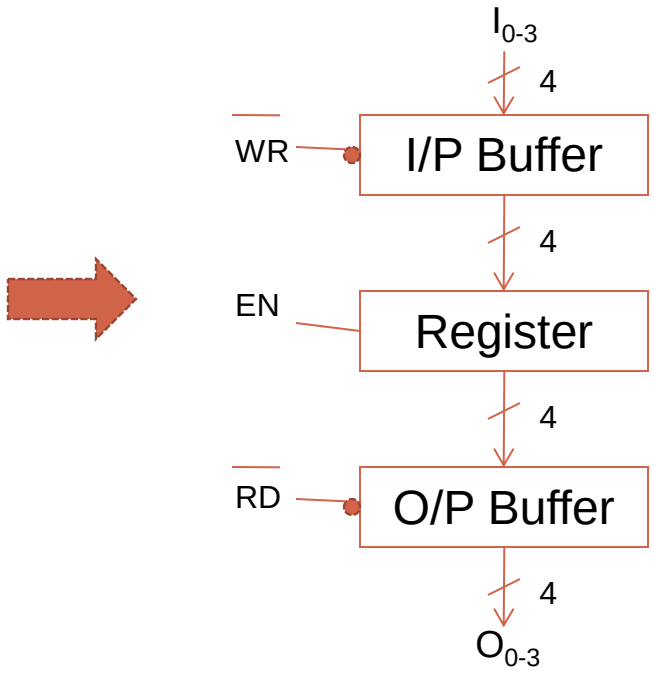


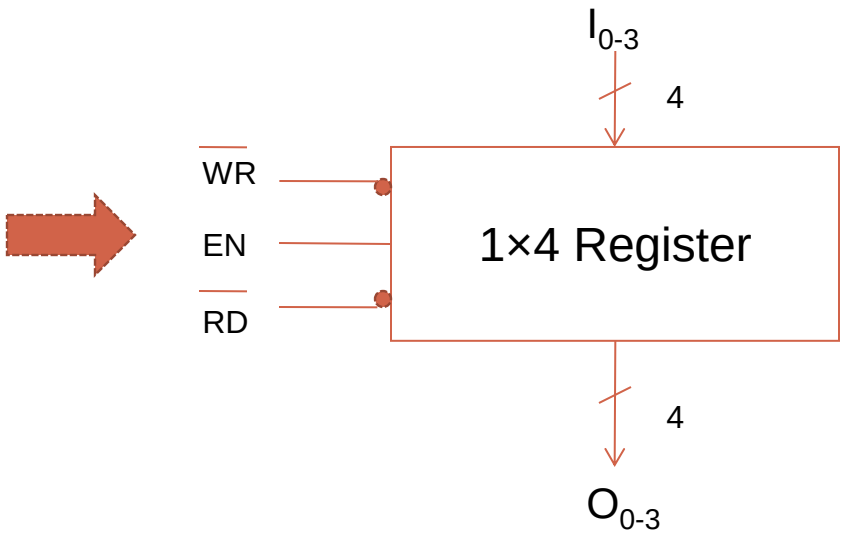
F/F or Latch

- To prevent unintentional change of input (i/p) and control of read operation, the F/F may be modified using a tri-state buffer.




I_3
I_2
I_1
I_0
WR
D
EN Q
D
EN Q
D
EN Q
D
EN Q
EN
RD
O_3
O_2
O_1
O_0


I_{0-3}
4
WR
I/P Buffer
4
EN
Register
4
RD
O/P Buffer
4
O_{0-3}


I_{0-3}
4
WR
EN
RD
1×4 Register
4
O_{0-3}

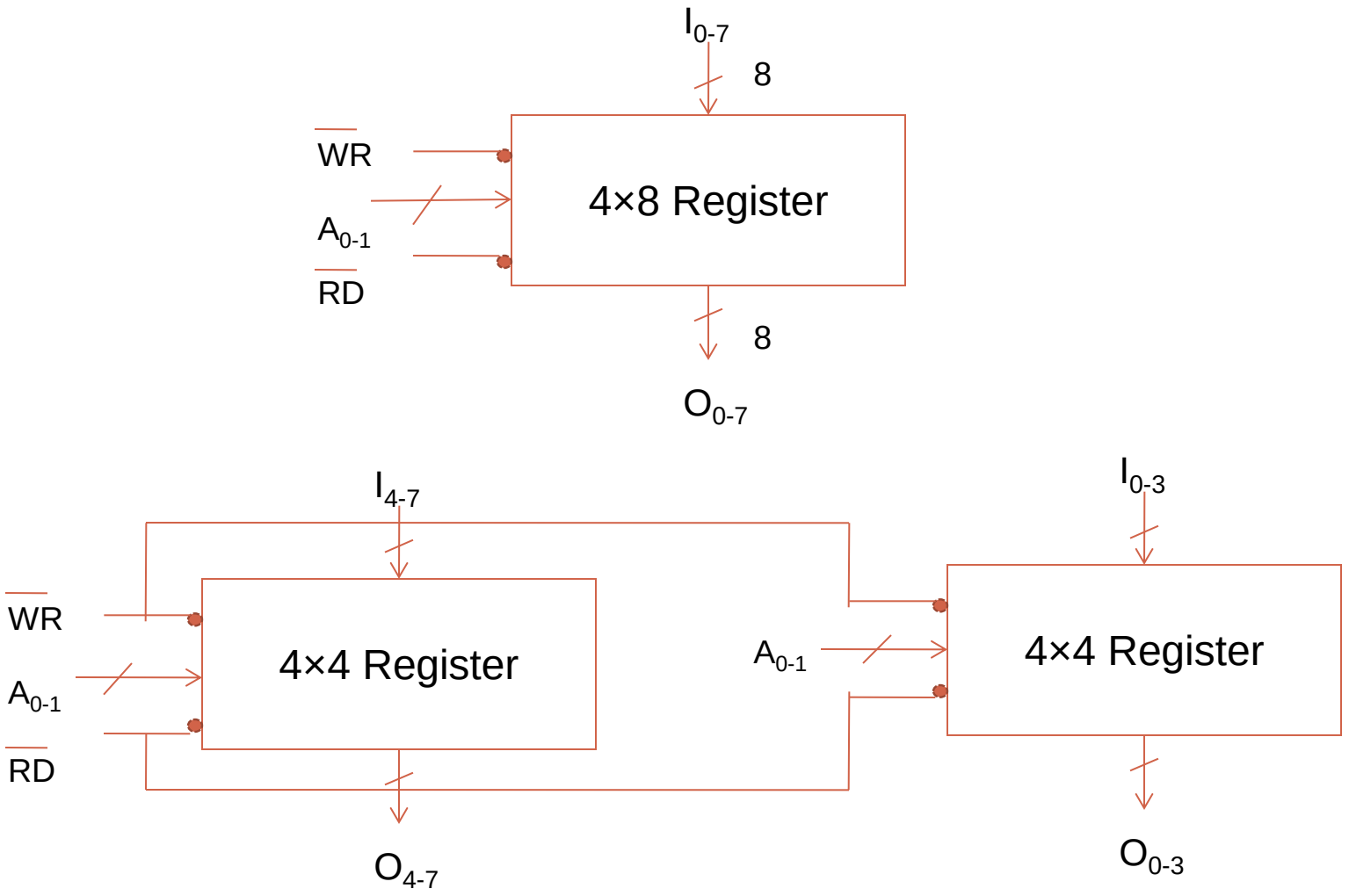
Memory Organization

In a memory (m/m) chip all the registers are arranged in a sequence and are identified by a binary numbers called the m/m addresses.

Communication with memory

1) Select the memory chip.
2) Identify the required register.
3) Perform Read or Write operation.

Memory Organization

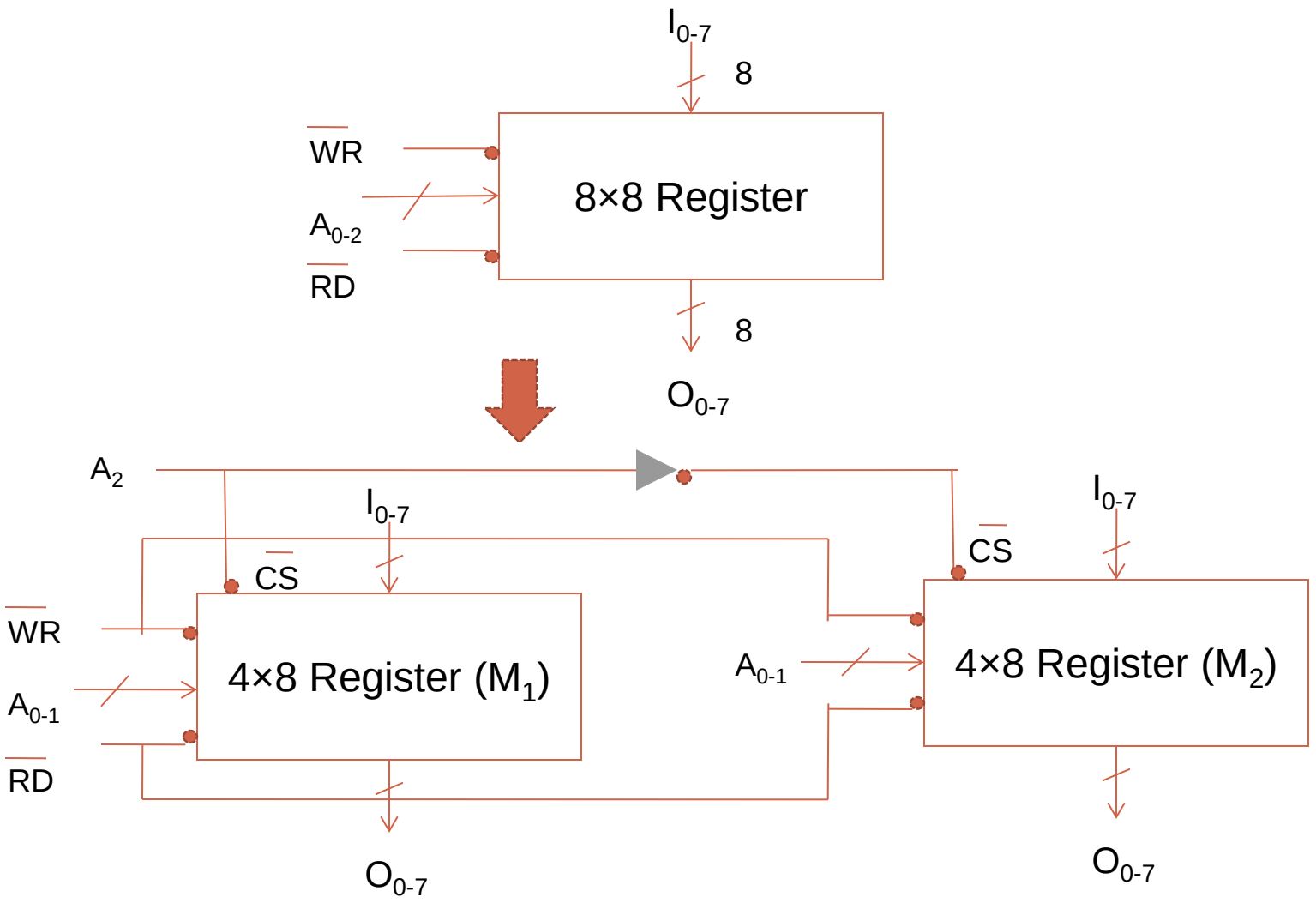


Made up with 2 chips with two 4 x 4 registers

Task:

Express an 8 x 16 Register using 8 x 4 Registers.

Memory Organization



Made up with 2 m/m chips of 4 x 8 registers. A_2 acts as chip select.

Task:

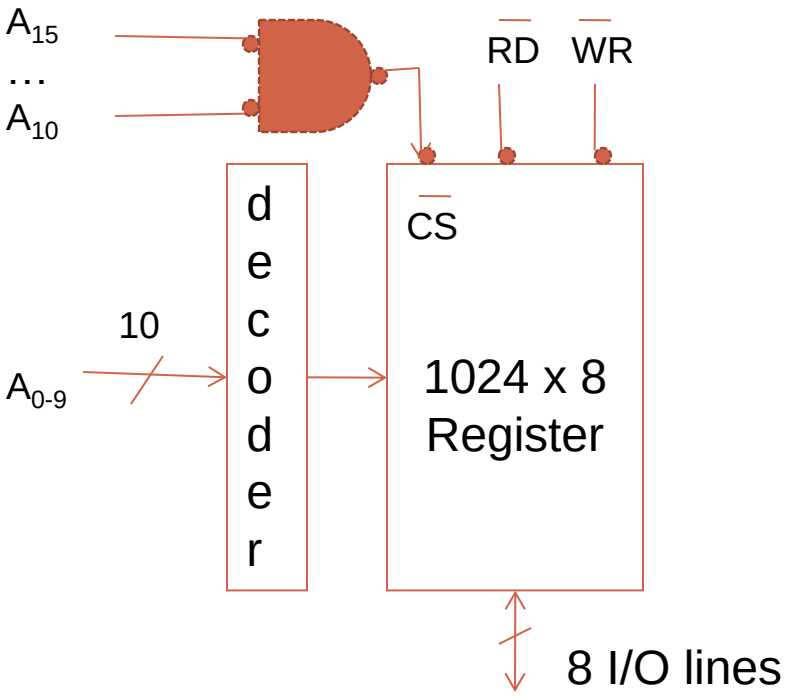
Express a 16 x 16 Register using 4 x 16 Registers.

Memory Map and Addresses

How is memory mapped and are addresses generated?

1) Select the number of address lines required to identify one register of the memory module. Assign the address lines starting from $A_{0\text{-}i}$.
2) Use the remaining address lines $A_{(i+1)\text{-}15}$ to generate a unique chip select (CS) signal.
3) Starting address is: $A_{15}\ A_{14}\ \ldots\ A_{i+1}\ 0\ 0\ \ldots\ 0$
4) Ending address is: $A_{15}\ A_{14}\ \ldots\ A_{i+1}\ 1\ 1\ \ldots\ 1$

Example: Memory address range of 1K (1024 x 8) memory



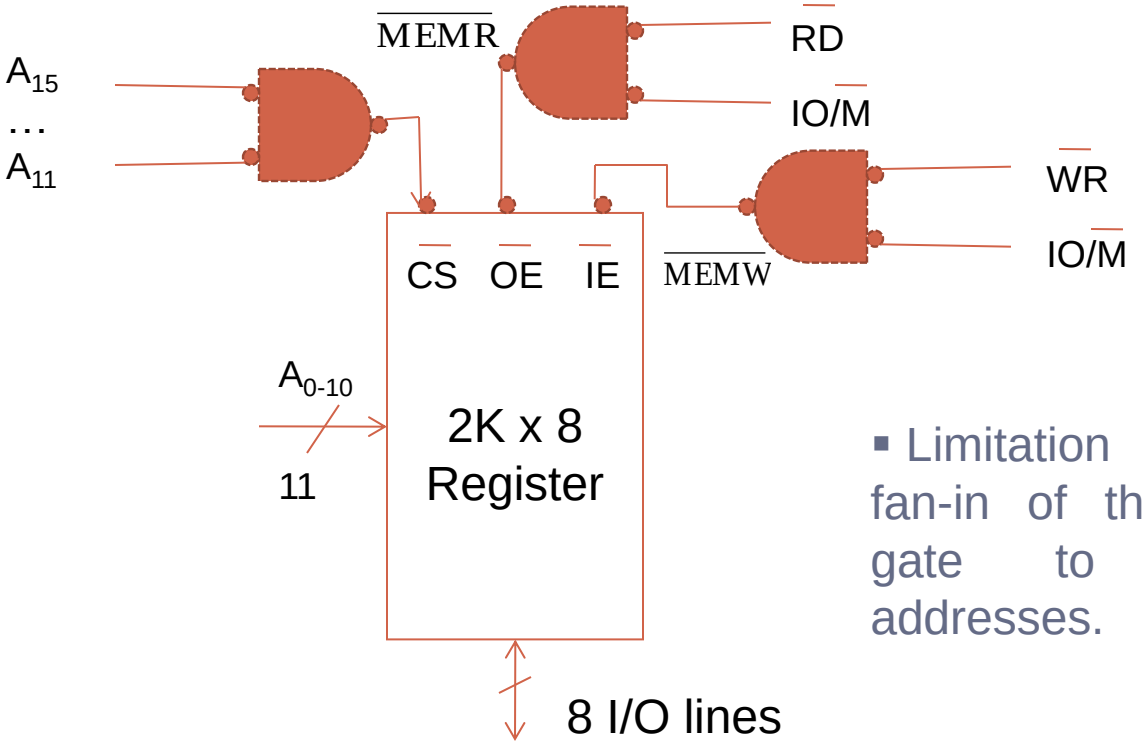
- By changing the combination of $A_{10\text{-}15}$, different memory address range can be formed.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Address range
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= $0\,0\,0\,0_H$
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	= $0\,3\,F\,F_H$

Memory Interfacing

1) Connect the required address lines of the address bus to the address lines of the memory chip.
2) Decode the remaining address lines of the address bus to generate a unique chip select signal.
3) Generate control signals $\overline{\text{MEMR}}$ and/or $\overline{\text{MEMW}}$ by combining $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signals with $\text{IO}/\overline{\text{M}}$ signal, respectively. Use them to enable appropriate input and/or output buffer.

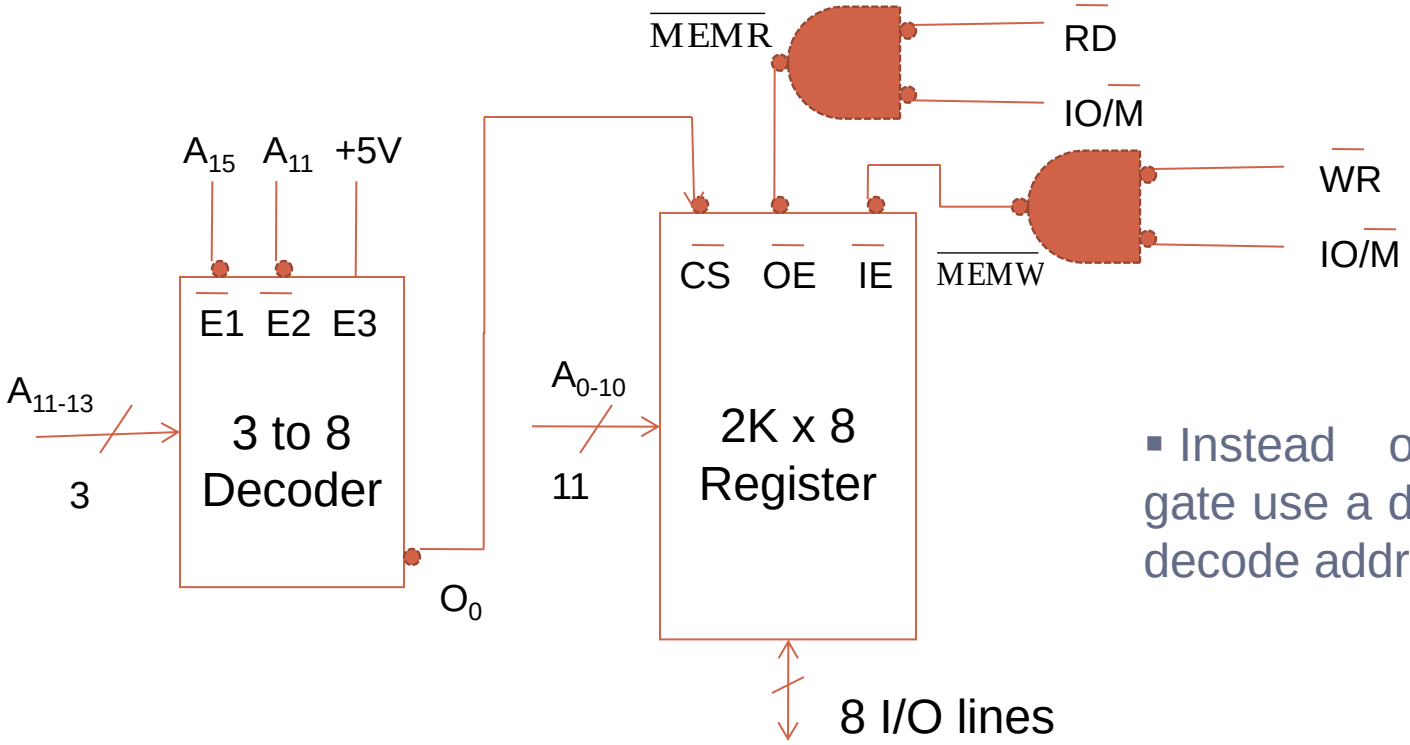
Example: Interfacing 2K Read/Write memory



- Limitation due to fan-in of the NAND gate to decode addresses.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Address range
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= $0\,0\,0\,0_H$
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	= $0\,7\,F\,F_H$

Example: Interfacing 2K Read/Write memory (Contd.)



- Instead of NAND gate use a decoder to decode addresses.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Address range
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= $0\,0\,0\,0_H$
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	= $0\,7\,F\,F_H$

Address Decoding

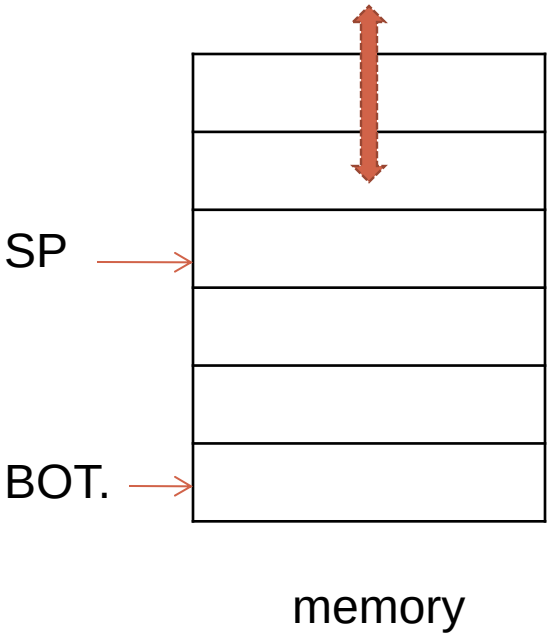
1) Complete or Absolute Decoding: All the available address lines are used for address decoding.
2) Partial Decoding: Not all the available address lines are used for address decoding. Others are kept as *don't* care condition.

Foldback or Mirror Memory: For each combination of don't care address lines, we have separate address range. Keeping one combination as primary address range, remaining address ranges are called the foldback or mirror memory of the primary address range.

Organization of data

1) Stack (LIFO - Last In First Out)

a) Addition and deletion from top of the stack by push and pop, respectively.
b) Only one pointer (SP) is required. SP points to the empty location.
c) Bottom end is fixed.



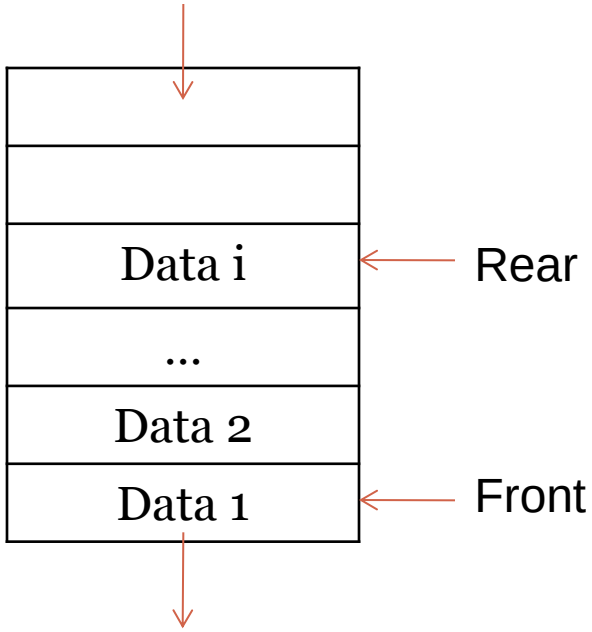
Example:

PUSH B ; M[SP--] = B, M[SP--] = C

POP B; C = M[++SP], B = M[++SP]

2) Queue (FIFO - First In First Out)

a) New data are added at the rear end and retrieved from the front end.
b) Two pointers are required to track front and rear ends.
c) Two ends are not fixed.



Subroutines: Used to perform a particular task many times on different data.

- Saves memory spaces.

Subroutine Call:

1) Store the content of PC in the link register.
2) Branch to the target address specified in the instruction.

Return from Subroutine:

1) PC is loaded with the content of link register.
2) Branch to the specified address.

```
Example:  CALL  SUB_NAME
          Next Instn.

SUB_NAME:  Instns
           RET
```