ज्ञानेन प्रकाशते जगत्
INDUS UNIVERSITY

MICROPROCESSORS

(SUBJECT CODE-CE0506/CS0506)

UNIT-1

B.TECH (CE/CSE)

SEMESTER-V

Shikha Singh

Academic Year 2019-2020

Syllabus

Unit	Chapters
1	**Introduction to 8085 Microprocessor** Basic functions of the microprocessor, System bus, Architecture, Pin Configuration and Programmer's model of Intel 8085 Microprocessor. Overview of the instruction groups of 8085 and the addressing modes..
2	**Intel 8086 Architecture:** features of 8086 processor, 8086/88 CPU Architecture and the pipelined operation, Programmer's Model and Segmented Memory. **Designing the 8086 CPU module:** 8086 pin description in details, Generating the 8086 System Clock and Reset Signals, 8086 Minimum and Maximum Mode CPU Modules, Minimum and Maximum Mode Timing Diagrams, Interrupt Structure, Interrupt Processing and the Predefined interrupts in 8086 Processor

Unit	Chapters
3	**Instruction Set of 8086 and Programming:** Instruction Set of 8086 microprocessor in details, Addressing modes of 8086/88, Programming the 8086 in assembly language, Mixed mode programming with C-language and assembly. **Peripheral Controllers for 8086 family and System Design:** Functional Block Diagram and description, Control Word Formats, Operating Modes and Applications of the Peripheral Controller namely 8255-PPI, 8253-PIT, 8259- PIC and 8237-DMAC. Interfacing of the above Peripheral Controllers. Keyword and Display Interface using 8255. Memory Interfacing: SRAM, ROM and DRAM (using a typical DRAM Controller such as Intel 8203). System Design based on the Memory and Peripherals
4	**Multiprocessor Systems:** Study of Multiprocessor Configurations namely Tightly Coupled System (TCS) and Loosely Coupled System (LCS), TCS with the case study of the Coprocessor, Various System Bus Arbitration Schemes in LCS, and Role of the Bus Arbiter (Intel 8289) in the LCS.

Books

Text Book:

1) Microprocessor architecture and applications with 8085: By Ramesh Gaonkar (Penram
International Publication).
2) 8086/8088 family: Design Programming and Interfacing: By John Uffenbeck (Pearson
Education).
3) 8086 Microprocessor Programming and Interfacing the PC: By Kenneth Ayala
4) Microcomputer Systems: 8086/8088 family Architecture, Programming and Design: By
Liu & Gibson (PHI Publication).
5) Microprocessor and Interfacing: By Douglas Hall (TMH Publication).

Ref. Books:

1) Advanced Microprocessor: By Roy & Bhurchandi (Tata McGraw Hill).
2) Advanced Microprocessors: By Daniel Tabak (McGraw Hill)
3) The SPARC Architecture Manual (Version 8).
4) Intel Manuals.

Introduction to 8085 Microprocessor

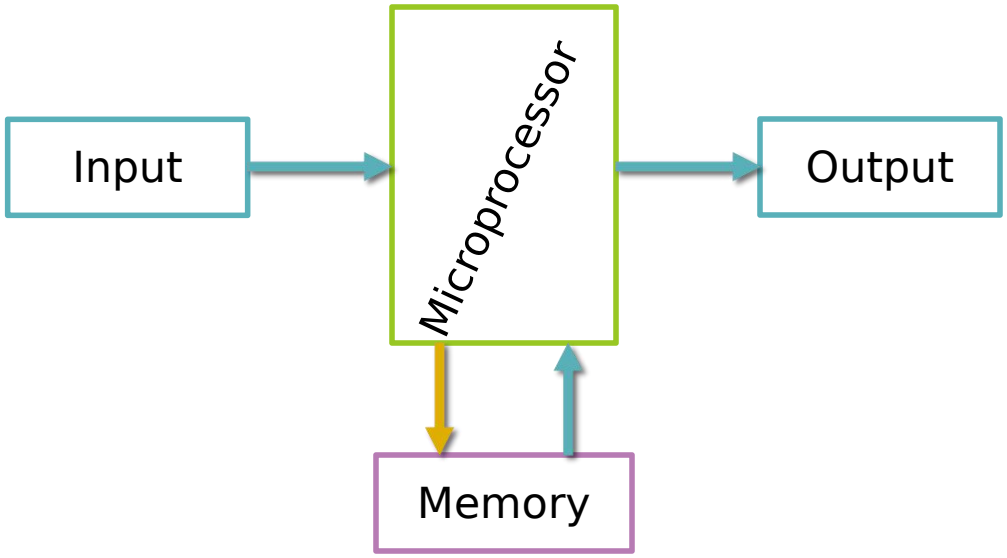
Basic functions of the microprocessor, System bus, Architecture, Pin Configuration and Programmer's model of Intel 8085 Microprocessor. Overview of the instruction groups of 8085 and the addressing modes. (No programming based on 8085).

Definition of the Microprocessor

- The microprocessor is a
  - Multipurpose,
  - Programmable,
  - Clock-driven,
  - Register-based Electronic device that reads binary instructions from the storage device called memory and takes in binary data and process it according to those instructions and provides result as output.

Programmable Machine

Programmable Machine

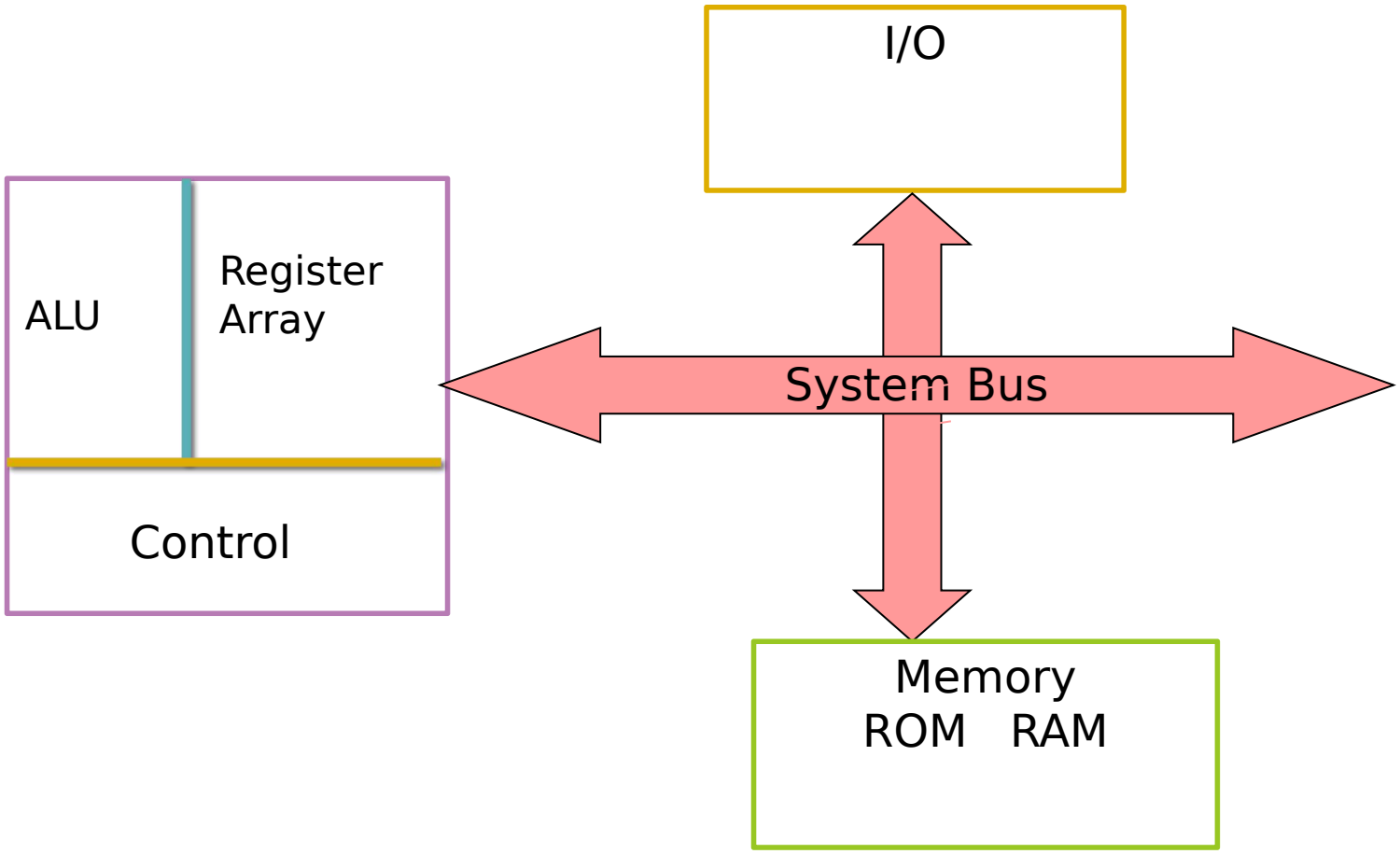


- Internally, the microprocessor is made up of 3 main units.
  - The Arithmetic/Logic Unit (ALU)
  - The Control Unit.
  - An array of registers for holding data while it is being manipulated.

- Arithmetic/Logic Unit
  - Performs all computing and logic operations such as addition and subtraction as well as AND, OR and XOR.
- Register Array
  - A collection of registers within the microprocessor itself. These are used primarily for data storage during program execution. The number and the size of these registers differ from one microprocessor to the other

- Control Unit
  - As the name implies, the control Unit controls what is happening in the microprocessor. It provides the necessary control and timing signals to all operations in the microprocessor as well as it contact to the outside world.

Organization of a microprocessor-based system



Memory

- Memory stores information such as instructions and data in binary format $(0$ and $1)$. It provides this information to the microprocessor whenever it is needed.

Memory(1)

- **Read Only Memory** (**ROM**) (**Non – Volatile**)
  - used to store information that does not change.
- **Random Access Memory** (**RAM**) (also known as Read/Write Memory).
  - used to store information supplied by the user. Such as data.

I/O (Input/Output)- *peripherals*.

- **Input devices** : transfer binary information from the outside world to the microprocessor.
  - **Examples of input devices are: keyboard, mouse, bar code reader, scanner etc.**
- **Output devices**: transfer binary information from the microprocessor to the outside world.
  - **Theses include things like an LED, a monitor, a printer etc.**

System Bus

- It is simply a group of wires carrying the voltages and curents representing the different bit values.

- The microprocessor communicates with only one peripheral at a time and the Controlling the bus is done by the Control Unit.

Program

- **Program**: A program is a sequence of instructions that bring data into the microprocessor, processes it and sends it out.
  - There are many programming languages (C, C++, FORTRAN etc.) However, these programming languages can be grouped into three main levels

Continue…

- Programming Languages
  - **Machine language**
    - Machine language is the lowest level programming language. It is a language intended to be understood by the microprocessor (the machine) only. In this language, every instruction is described by binary patterns.

      e.g. 11001101 **may mean** 1 + 3

    - This is the form in which instructions are stored in memory. This is the only form that the microprocessor understands.

Continue….

- Programming Languages
  - **Assembly language**
    - This language is more understandable by humans. In this language, the binary patterns are assigned mnemonics (short abbreviated names).
    - There is usually one assembly language instruction for each machine language instruction.

      e.g. “Add 1,3” is assigned to the machine language pattern 11001101 mentioned above to refer to the operation 1+3.

Continue….

- Programming Languages
  - **High level languages**
    - These are languages like C, PASCAL and Java. These are more natural for humans to use than assembly or machine languages. They are also more compact .
    - One high level instruction translates into many assembly or machine language instructions.
      e.g. z = y + x may translate into:

```
MOV  R1,#20H
MOV R2,#30H
ADD R1,R2
```

The three cycle instruction execution model

- To execute a program, the microprocessor “reads” each instruction from memory, “interprets” it, then “executes” it.
- To use the right names for the cycles:

  The microprocessor **fetch** each instruction,

  decodes it,

  Then **execute** it.

- This sequence is continued until all instructions are performed.

8085 Microprocessor Architecture

The 8085 Hardware Model & Programming Model

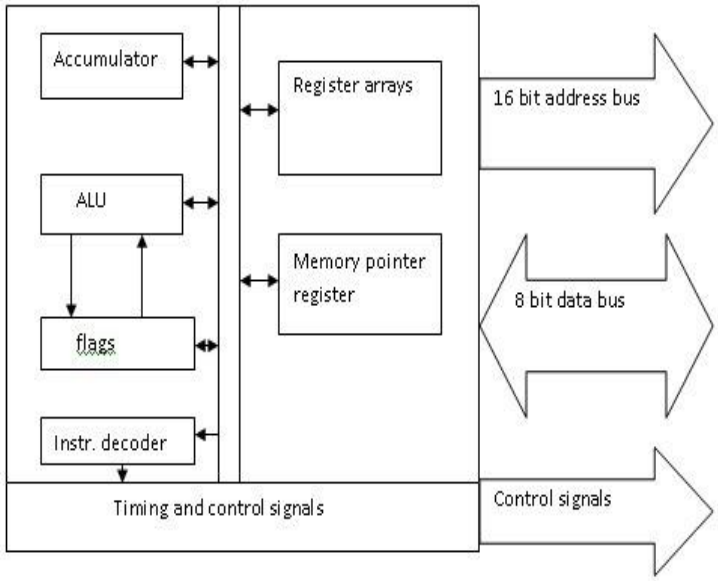


Fig. 1 8085 Hardware Model

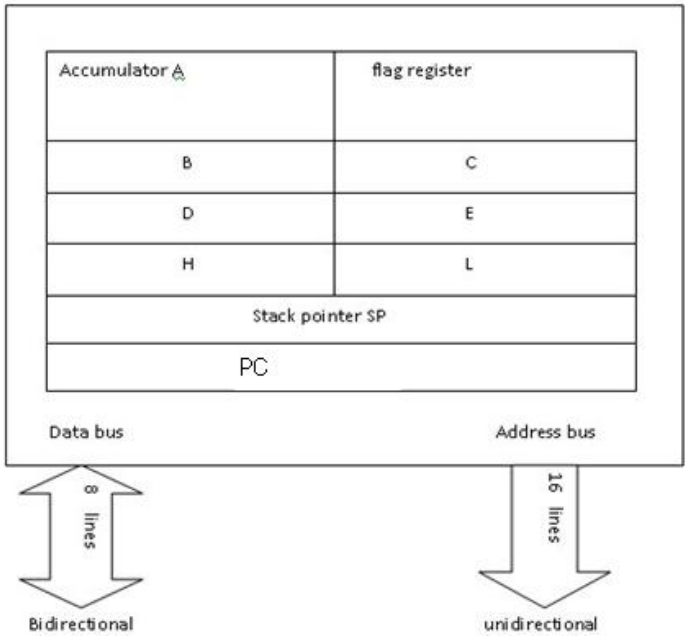
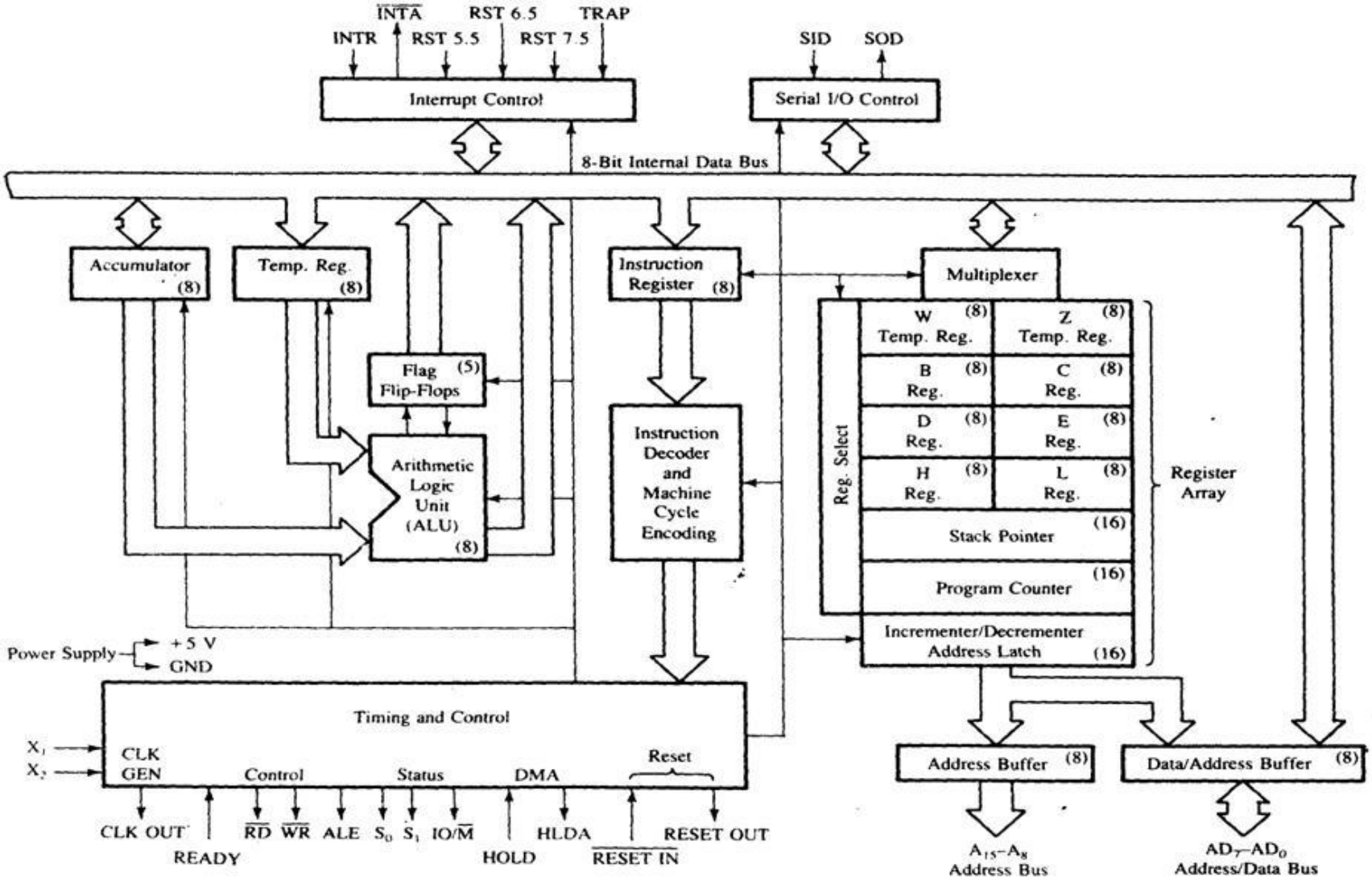


Fig.2 8085 Programming Model

8085 Microprocessor Architecture



8085 Hardware Model

- Fig 1 Shows the hardware model of 8085. it contains two major segment. One segment includes arithmetic / logic unit (ALU) , accumulator , flag register and instruction decoder.
- The second segment shows various 8-bit and 16-bit registers.
- The 8085 uses **8-bit bidirectional data bus** , **16-bit unidirectional address bus** and **control bus**.

8085 Programming Model

- The programming model represents the various registers of the microprocessor. It is very useful to write assembly language programs.

- The 8085 Programming model includes six registers, one accumulator and one flag register and it has two 16 bit registers **Stack Pointer**(**SP**) and **Program Counter** (**PC**) shown in figure 2.

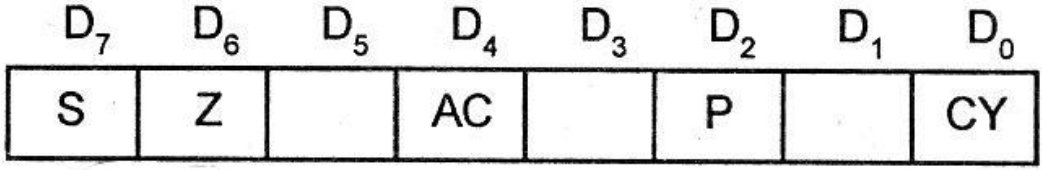
8085 Programming Model (2)

- **Registers**
  - The 8085 has 6 general purpose registers to store 8-bit data. These are identifies as a B,C,D,E,H,L. They can be combines as register pairs BC,DE,HL to perform 16-bit operations

- **Accumulator**
  - The accumulator is a 8-bit register that is part of ALU. This register is used to store the 8 bit data and to perform the arithmetic and logical operations. The result of operation is stored in to accumulator.

8085 Programming Model (3)

- **Flags**
  - The ALU includes five flip–flops , which are set or reset after an operation according to data conditions of the result in accumulator and other registers. The y are called Zero (Z) , Carry (Cy) , Sign (S) , Parity (P) , Auxiliary Carry (AC) flags.



8085 **Flag Register**

Flag Register

- **Zero Flag** (**Z**): The zero flag is set when the result is zero, otherwise it is reset.
- **Carry Flag** (**CY**): If an arithmetic operation results in carry, the carry flag is set, otherwise it is reset .
- **Sign Flag**(**S**) : Sign flag is set if the D_7 of the result is 1, otherwise it is reset .
- **Parity Flag**(**P**) : If the result has an even number of 1's the flag is set, for an odd number of 1's it is reset.
- **Auxiliary Carry** (**AC**): In an arithmetic operation , when a carry is generated by digit D_3 and passed to D_4, the AC flag is set. Used for BCD operations.

Program Counter (PC) & Stack Pointer(SP)

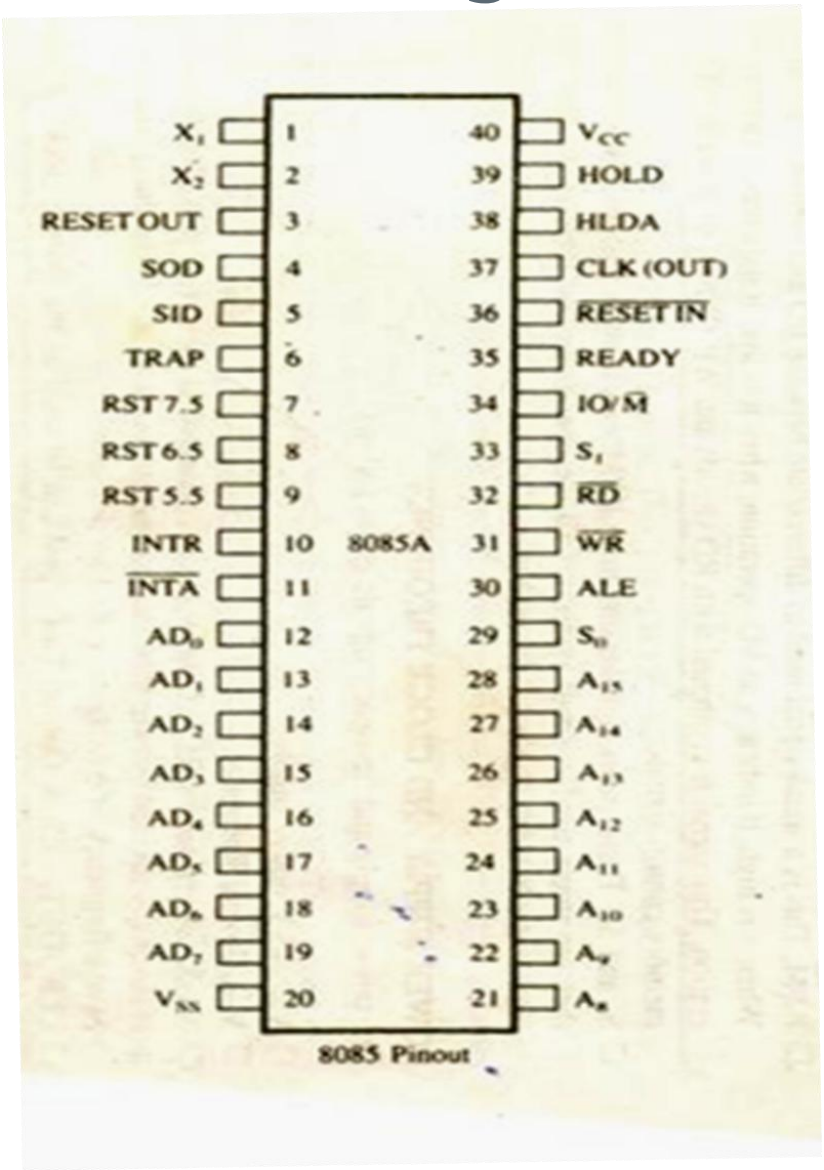
- These are the two 16-bit registers to hold the memory address.
- **PC**:
  - Microprocessor use the PC to Sequence the Execution of the Program. Function of the program counter is to point the memory address from which next byte is to be fetched.
  - **PC contains the Address of the next instruction to be Executed**
- **SP**:
  - The SP points to the R/W memory location and **it is used for the temporary storage of the Data.**

8085 Instruction Classification

- The 8085 instructions can be classified as the following five categories:

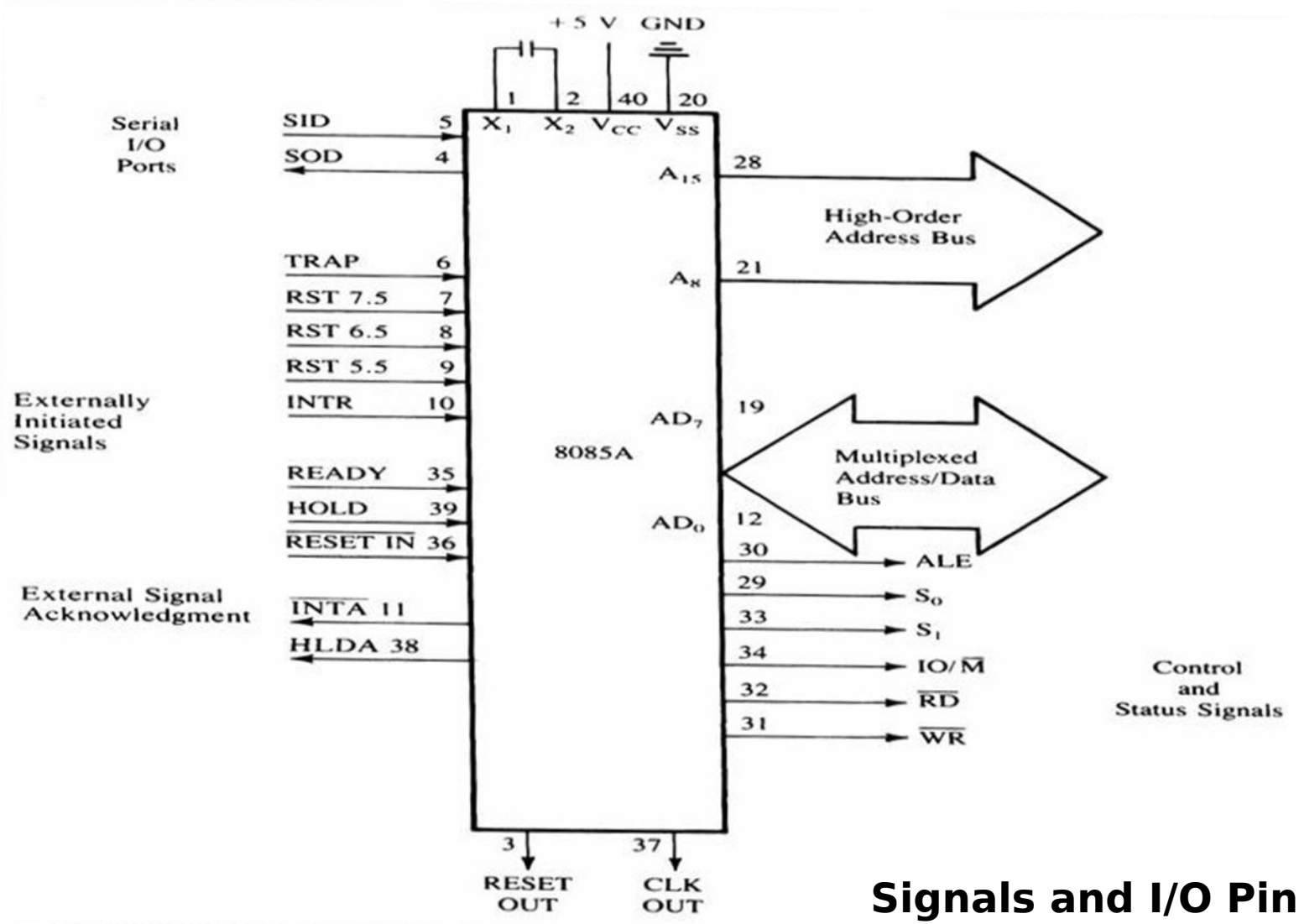
Data transfer Operations - Between registers - Specific data byte to a Register from memory location - Between a Memory location and Register - Between I/O device and the Accumulator	**Arithmetic Operations** - Addition - Subtraction - Increment / Decrement
Logical Operations - AND , OR , X-OR , - Rotate , Compare , Complement	**Branching Operations** - Jump : Conditional & Unconditional - Call , Return and Restart
Machine Control Operations - Halt , Interrupt or Do nothing	

Intel 8085 Pin Configuration



8085 Pinout

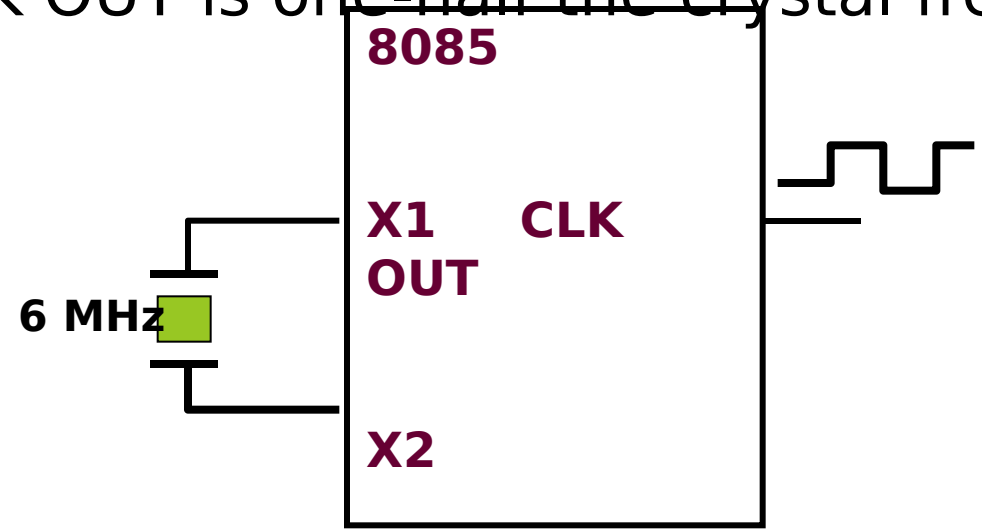
Signals



Signals and I/O Pins

Clock Pins

- 8085 MPU has 3 pins that control or present the clock signal.
  - X1 and X2 pins determine the clock frequency.
  - CLK OUT is a square-wave output clock.
- The CLOCK OUT is one-half the crystal frequency.



8085 Pinout

- 8085 µp consists of 16 signal pins use as address bus.
- Divide into 2 part: A15 – A8 (upper) and AD7 – AD0 (lower).
  - A15 – A8 : Unidirectional, known as ‘high order address bus’.
  - AD7 – AD0 : bidirectional and dual purpose (address and data placed once at a time).
  - AD7 – AD0 also known as ‘low order address’.

Control and Status Signals

- Signals:
  - $\overline{RD}$ – Read (active low). To indicate that the I/O or memory selected is to be read and data are available on the bus.
  - $\overline{WR}$ – Write: Active low. This is to indicate that the data available on the bus are to be written to memory or I/O ports.
  - IO/M – To differentiate I/O operation or memory operations.
    - ‘0’ - indicates a memory operation.
    - ‘1’-indicates an I/O operation.
  - IO/M combined with $\overline{RD}$ and $\overline{WR}$ to generate I/O and memory control signals.
  - S1 and S0: Status signals, similar to IO/M, can identify various operations as shown on the following table :

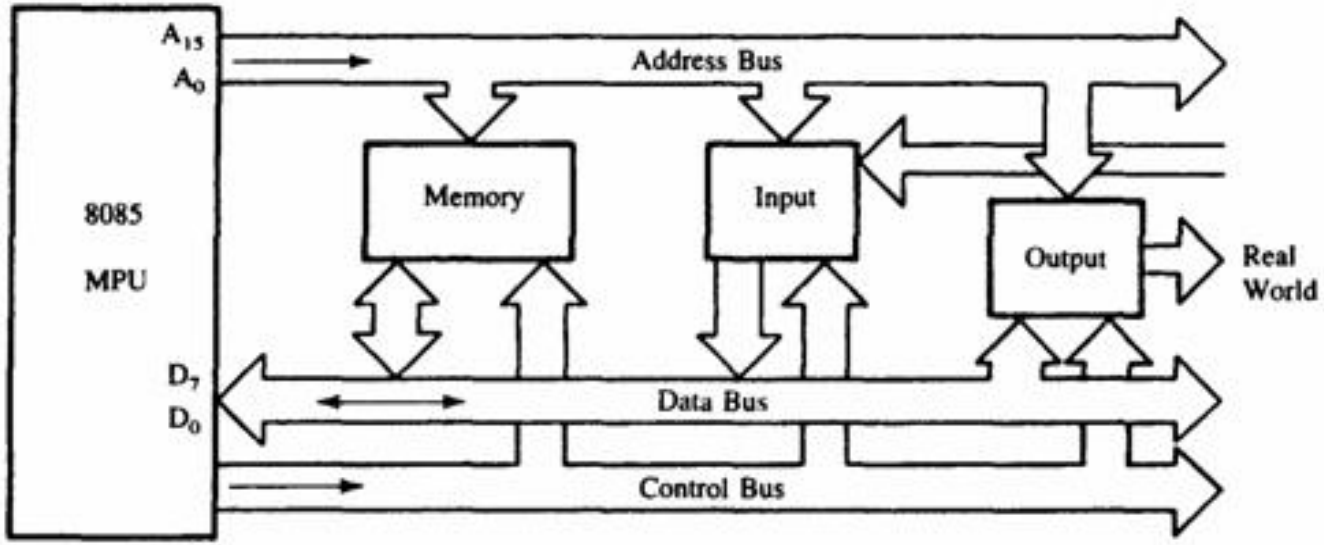
TABLE 4.1
8085 Machine Cycle Status and Control Signals

Machine Cycle	Status IO/$\overline{M}$	Status S_1	Status S_0	Control Signals
Opcode Fetch	0	1	1	$\overline{RD} = 0$
Memory Read	0	1	0	$\overline{RD} = 0$
Memory Write	0	0	1	$\overline{WR} = 0$
I/O Read	1	1	0	$\overline{RD} = 0$
I/O Write	1	0	1	$\overline{WR} = 0$
Interrupt Acknowledge	1	1	1	$\overline{INTA} = 0$
Halt	Z	0	0	$\overline{RD}$, $\overline{WR}$ = Z and $\overline{INTA} = 1$
Hold	Z	X	X	
Reset	Z	X	X	

NOTE: Z = Tri-state (high impedance)
X = Unspecified

S1	S0	Status
0	0	Halt
0	1	Write
1	0	Read
1	1	Fetch

8085 Bus Structure



Interrupt Signals

- 8085 µp has several interrupt and external intiated signals as shown in the following table.

TABLE 4.2
8085 Interrupts and Externally Initiated Signals

☐ INTR (Input)	Interrupt Request: This is used as a general-purpose interrupt; it is similar to the INT signal of the 8080A.
☐ $\overline{\text{INTA}}$ (Output)	Interrupt Acknowledge: This is used to acknowledge an interrupt.
☐ RST 7.5 (Inputs) RST 6.5 RST 5.5	Restart Interrupts: These are vectored interrupts that transfer the program control to specific memory locations. They have higher priorities than the INTR interrupt. Among these three, the priority order is 7.5, 6.5, and 5.5.
☐ TRAP (Input)	This is a nonmaskable interrupt and has the highest priority.
☐ HOLD (Input)	This signal indicates that a peripheral such as a DMA (Direct Memory Access) controller is requesting the use of the address and data buses.
☐ HLDA (Output)	Hold Acknowledge: This signal acknowledges the HOLD request.
☐ READY (Input)	This signal is used to delay the microprocessor Read or Write cycles until a slow-responding peripheral is ready to send or accept data. When this signal goes low, the microprocessor waits for an integral number of clock cycles until it goes high.

Interrupt signals

- An interrupt is a hardware-initiated subroutine CALL.
- When interrupt pin is activated, an ISR will be called, interrupting the program that is currently

Pin	Subroutine Location
TRAP	0024H
RST 5.5	002CH
RST 6.5	0034H
RST 7.5	003CH

RESET signal

- Following are the two kind of RESET signals:
  - RESET IN: an active low input signal, Program Counter (PC) will be set to 0 and thus MPU will reset.
  - RESET OUT: an output reset signal to indicate that the µp was reset (i.e. RESET IN=0). It also used to reset external devices.

ALE signal

- ALE means Address latch Enable and it is used to Demultiplex the lower order address and data lines.

ALE used to Demultiplex address/data bus

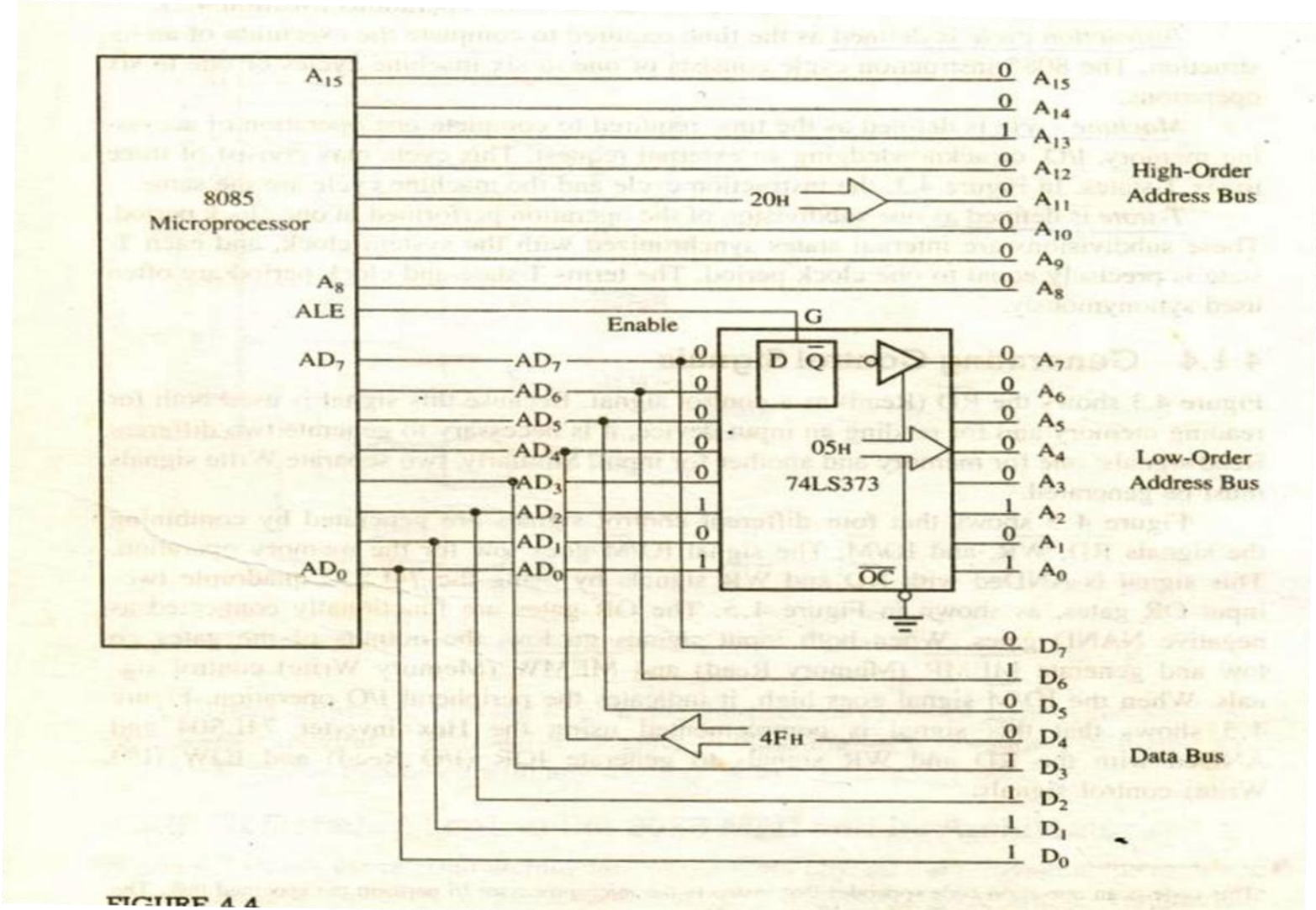
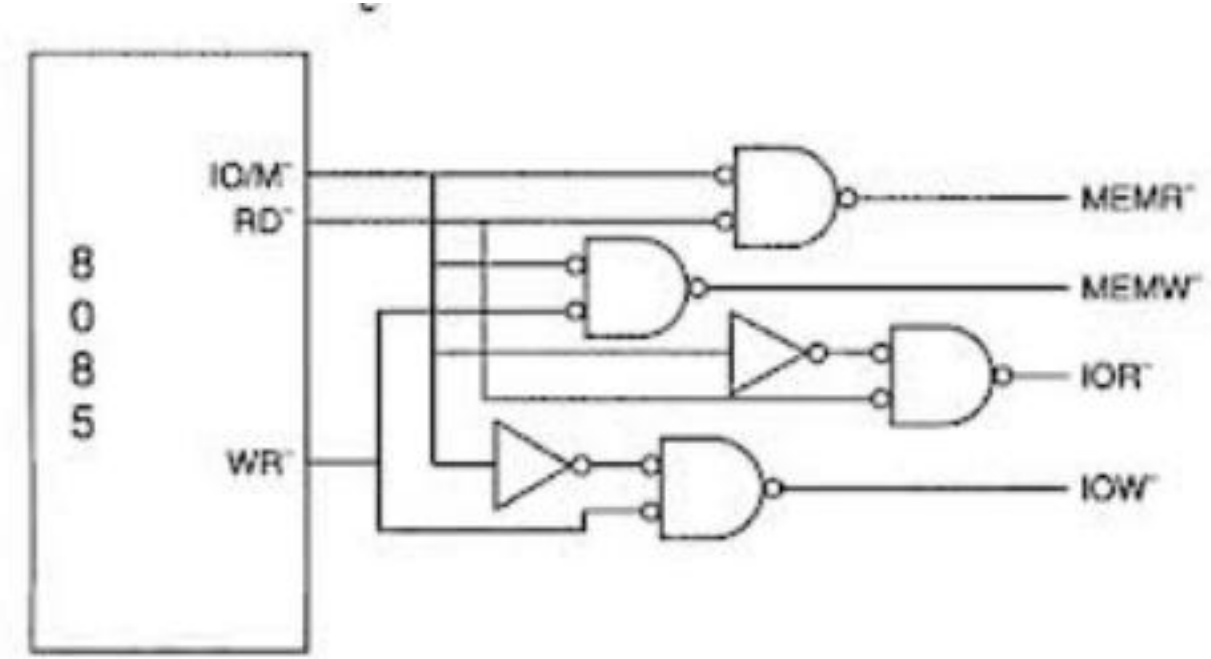


FIGURE 4.4

Generation of Control signal



SID and SOD signal

- SID:-
  - SID is serial input data pin. The bit data on this line is loaded in the seventh bit of the accumulator whenever a RIM instruction is executed.
- SOD:-
  - SOD is serial output data. The output SOD is set or reset as specified by the SIM instruction

Instruction

- **An instruction is a command to the Microprocessor to perform a given task on Specific data.**
- Each instruction has 2- parts
  - Op-Code
  - Operands
- **Instruction Word Size**:
  - **Instruction are commonly referred to in terms of Byte**
    - 1 **Byte Instruction** (**Ex. MOV E,A , ADD E , CMA**)
    - 2 **Byte Instruction** (**Ex. MVI A,** 35**H , MVI B,**29**H**)
    - 3 **Byte Instruction** (**Ex. LDA** 2055**H , JMP** 2080**H**)

Instruction Set of 8085

- An instruction is a binary pattern designed inside a microprocessor to perform a specific function.
- The entire group of instructions that a microprocessor supports is called ***Instruction Set***.
- 8085 has **246** instructions.
- Each instruction is represented by an 8-bit binary value.
- These 8-bits of binary value is called ***Op-Code*** or ***Instruction Byte***.

Classification of Instruction Set

- Data Transfer Instruction
- Arithmetic Instructions
- Logical Instructions
- Branching Instructions
- Machine Control Instructions

Data Transfer Instruction

- These instructions move data between registers, or between memory and registers.
- These instructions copy data from source to destination.
- While copying, the content of source is not modified.

Data Transfer Instruction

Opcode	Operand	Description
MOV	Rd,Rs M, Rs Rd, M	Copy from source to destination.

- This instruction copies the contents of the source register into the destination register.
- The contents of the source register are not altered.
- If one of the operands is a memory location, its location is specified by the contents of the HL registers.
  - **Example**: MOV D,E or MOV B, A

Data Transfer Instruction

Opcode	Operand	Description
MVI	Rd, Data M, Data	Move immediate 8-bit

- The 8-bit data is stored in the destination register or memory.
- If the operand is a memory location, its location is specified by the contents of the H-L registers.
  - **Example**: MVI B, 75H or MVI M, 55H

Data Transfer Instruction

Opcode	Operand	Description
LDA	16-bit address	Load Accumulator Direct

- The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator.
- The contents of the source are not altered.
  - Example: LDA 2037H

Data Transfer Instruction

Opcode	Operand	Description
LDAX	B/D Register Pair	Load accumulator indirect

- The contents of the designated register pair point to a memory location.
- This instruction copies the contents of that memory location into the accumulator.
- The contents of either the register pair or the memory location are not altered.
  - Example: LDAX D

Data Transfer Instruction

Opcode	Operand	Description
LXI	Reg. pair, 16-bit data	Load register pair immediate

oThis instruction loads 16-bit data in the register pair.

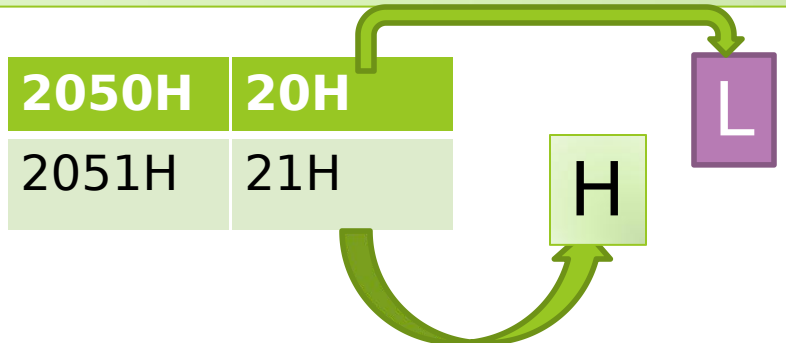
o **Example**: LXI H, 2074 H

o This instruction store 74H in L and 20H in H.

Data Transfer Instruction

Opcode	Operand	Description
LHLD	16-bit address	Load H-L registers direct

- This instruction copies the contents of memory location pointed out by 16-bit address into register L.
- It copies the contents of next memory location into register H.
- Example: LHLD 2050H
- Location 2050H contains 20H and 2051H contains 21H
- 21H will be loaded in to H and 20H will be loaded in to L



Data Transfer Instruction

Opcode	Operand	Description
STA	16-bit address	Store accumulator direct

- The contents of accumulator is copied into the memory location specified by the operand.

- **Example**: STA 2050 H

Data Transfer Instruction

Opcode	Operand	Description
STAX	B/D Reg. pair	Store accumulator indirect

oThe contents of accumulator are copied into the memory location specified by the contents of the register pair.

o **Example**: STAX B

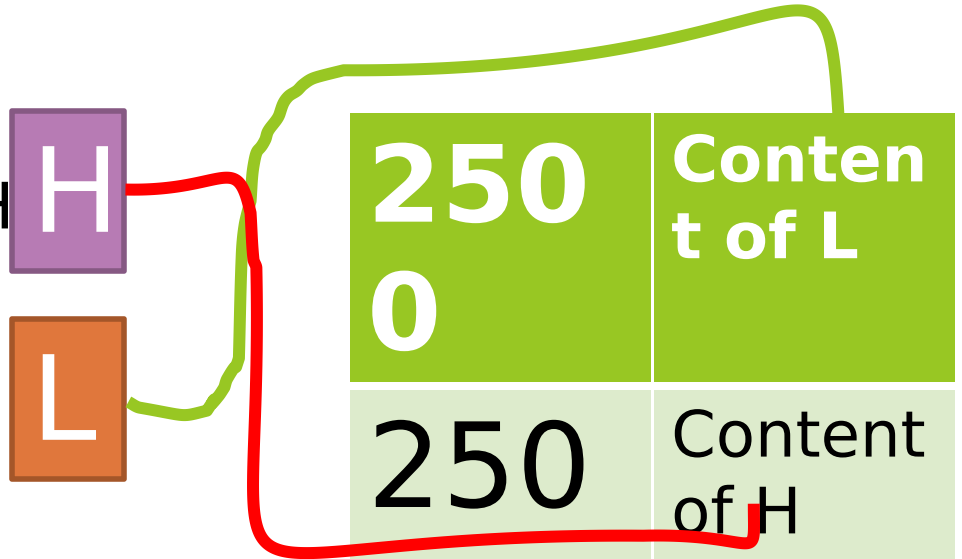
Data Transfer Instruction

Opcode	Operand	Description
SHLD	16-bit address	Store H-L registers direct

oThe contents of register L are stored into memory location specified by the 16-bit address.

oThe contents of register H are stored into the next memory location.

o**Example**: SHLD 2500 H



Data Transfer Instruction

Opcode	Operand	Description
XCHG	None	Exchange H-L with D-E

oThe contents of register H are exchanged with the contents of register D.

oThe contents of register L are exchanged with the contents of register E.

o**Example**: XCHG

Data Transfer Instruction

Opcode	Operand	Description
SPHL	None	Copy H-L pair to the Stack Pointer (SP)

oThis instruction loads the contents of H–L pair into SP.

o**Example**: SPHL

oH provides the higher order byte and L provides the lower order byte.

Data Transfer Instruction

Opcode	Operand	Description
XTHL	None	Exchange H–L with top of stack

oThe contents of L register are exchanged with the location pointed out by the contents of the SP.

oThe contents of H register are exchanged with the next location $(SP + 1)$.

o **Example**: XTHL

Data Transfer Instruction

Opcode	Operand	Description
PCHL	None	Load program counter with H-L contents

oThe contents of registers H and L are copied into the program counter (PC).

oThe contents of H are placed as the high-order byte and the contents of L as the low-order byte.

o**Example**: PCHL

Data Transfer Instruction

Opcode	Operand	Description
PUSH	Reg. pair	Push register pair onto stack

oThe contents of register pair are copied onto stack.

oSP is decremented and the contents of high-order registers (B, D, H, A) are copied into stack.

oSP is again decremented and the contents of low-order registers (C, E, L, Flags) are copied into stack.

o**Example**: PUSH B

Data Transfer Instruction

Opcode	Operand	Description
POP	Reg. pair	Pop of the stack to register pair

oThe contents of top of stack are copied into register pair.

oThe contents of location pointed out by SP are copied to the low-order register (C, E, L, Flags).

oSP is incremented and the contents of location are copied to the high-order register (B, D, H, A).

o**Example**: POP H

```
 LXI  H,2525h
MVI M,25h
MOV B,M
MVI L,26h
MVI M,26H
MOV C,M
SHLD 4000h
LDAX B
SPHL
```

Arithmetic Instruction

Opcode	Operand	Description
ADD	R or M	Add register or memory to accumulator

oThe contents of register or memory are added to the contents of accumulator.

oThe result is stored in accumulator.

oIf the operand is memory location, its address is specified by H–L pair.

oAll f lags are modified to reflect the result of the addition.

o**Example**: ADD C or ADD M

Arithmetic Instruction

Opcode	Operand	Description
ADC	R or M	Add register or memory to accumulator with carry

oThe contents of register or memory and Carry Flag (CY) are added to the contents of accumulator.

oThe result is stored in accumulator.

o If the operand is memory location, its address is specified by H–L pair.

oAll f lags are modified to reflect the result of the addition.

o**Example**: ADC D or ADC M

Arithmetic Instruction

Opcode	Operand	Description
ADI	8-bit data	Add immediate to accumulator

oThe 8-bit data is added to the contents of accumulator.

o The result is stored in accumulator.

oAll f lags are modified to reflect the result of the addition.

o **Example**: ADI 49 H

Arithmetic Instruction

Opcode	Operand	Description
ACI	8-bit data	Add immediate to accumulator with carry

oThe 8-bit data and the Carry Flag (CY) are added to the contents of accumulator.

oThe result is stored in accumulator.

oAll f lags are modified to reflect the result of the addition.

o **Example**: ACI 48 H

Arithmetic Instruction

Opcode	Operand	Description
DAD	Reg. pair	Add register pair to H-L pair

- The 16-bit contents of the register pair are added to the contents of H–L pair.
- The result is stored in H–L pair.
- If the result is larger than 16 bits, then CY is set.
- No other f lags are changed.
- **Example**: DAD D

Arithmetic Instruction

Opcode	Operand	Description
SUB	R or M	Subtract register or memory from accumulator

oThe contents of the register or memory location are subtracted from the contents of the accumulator.

o The result is stored in accumulator.

oIf the operand is memory location, its address is specified by H-L pair.

oAll f lags are modified to reflect the result of subtraction.

o **Example:** SUB H or SUB M

Arithmetic Instruction

Opcode	Operand	Description
SBB	R or M	Subtract register or memory from accumulator with borrow

oThe contents of the register or memory location and Borrow Flag (i.e. CY) are subtracted from the contents of the accumulator.

oIf the operand is memory location, its address is specified by H-L pair.

oAll flags are modified to reflect the result of subtraction.

o **Example**: SBB D or SBB M

Arithmetic Instruction

Opcode	Operand	Description
SUI	8-bit data	Subtract immediate from accumulator

oThe 8-bit data is subtracted from the contents of the accumulator.

o The result is stored in accumulator.

o All f lags are modified to reflect the result of subtraction.

o **Example**: SUI 55 H

Arithmetic Instruction

Opcode	Operand	Description
SBI	8-bit data	Subtract immediate from accumulator with borrow

oThe 8-bit data and the Borrow Flag (i.e. CY) is subtracted from the contents of the accumulator.

oThe result is stored in accumulator.

oAll f lags are modified to reflect the result of subtraction.

o **Example**: SBI 65 H

Arithmetic Instruction

Opcode	Operand	Description
INR	R or M	Increment register or memory by 1

oThe contents of register or memory location are incremented by 1.

oThe result is stored in the same place.

oIf the operand is a memory location, its address is specified by the contents of H–L pair.

o **Example**: INR D or INR M

Arithmetic Instruction

Opcode	Operand	Description
INX	R	Increment register pair by 1

oThe contents of register pair (B, D, H, SP) are incremented by 1.

oThe result is stored in the same place.

oNo flags are affected.

o**Example**: INX B

Arithmetic Instruction

Opcode	Operand	Description
DCR	R or M	Decrement register or memory by 1

oThe contents of register or memory location are decremented by 1.

oThe result is stored in the same place.

oIf the operand is a memory location, its address is specified by the contents of H–L pair.

o **Example**: DCR C or DCR M

Arithmetic Instruction

Opcode	Operand	Description
DCX	R	Decrement register pair by 1

- The contents of register pair(B,D,H,SP) are decremented by 1.
- The result is stored in the same place.
- No flags are affected.
- **Example**: DCX D

Logical Instructions

Opcode	Operand	Description
CMP	R or M	Compare register or memory with accumulator

oThe contents of the operand (register or memory) are compared with the contents of the accumulator.

oBoth contents are preserved .

oThe result of the comparison is shown by setting the flags of the PSW as follows:

Logical Instructions

- if (A) < (reg/mem): carry flag is set
- if (A) = (reg/mem): zero flag is set
- if (A) ! (reg/mem): carry and zero flags are reset.
- **Example**: CMP C or CMP M

Logical Instructions

Opcode	Operand	Description
CPI	8-bit data	Compare immediate with accumulator

oThe 8-bit data is compared with the contents of accumulator.

oThe values being compared remain unchanged.

oThe result of the comparison is shown by setting the flags of the PSW as follows:

Logical Instruction

- if (A) < data: carry f lag is set
- if (A) = data: zero f lag is set
- if (A) ı data: carry and zero f lags are reset
- **Example**: CPI 87H

Logical Instructions

Opcode	Operand	Description
ANA	R or M	Logical AND register or memory with accumulator

oThe contents of the accumulator are logically ANDed with the contents of register or memory.

oThe result is placed in the accumulator.

oS, Z, P are modified to reflect the result of the operation.

o**Example**: ANA C or ANA M.

Logical Instruction

Opcode	Operand	Description
ANI	8-bit data	Logical AND immediate with accumulator

oThe contents of the accumulator are logically ANDed with the 8-bit data.

oThe result is placed in the accumulator.

oS, Z, P are modified to reflect the result.

o**Example**: ANI 96H.

Logical Instruction

Opcode	Operand	Description
ORA	R or M	Logical OR register or memory with accumulator

- The contents of the accumulator are logically ORed with the contents of the register or memory.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example**: ORA B or ORA M.

Logical Instruction

Opcode	Operand	Description
ORI	8-bit data	Logical OR immediate with accumulator

oThe contents of the accumulator are logically ORed with the 8-bit data.

oThe result is placed in the accumulator.

oS, Z, P are modified to reflect the result.

oCY and AC are reset.

o**Example**: ORI 46H.

Logical Instruction

Opcode	Operand	Description
XRA	R or M	Logical XOR register or memory with accumulator

oThe contents of the accumulator are XORed with the contents of the register or memory.

oThe result is placed in the accumulator.

oS, Z, P are modified to reflect the result of the operation.

oCY and AC are reset.

o**Example**: XRA C or XRA M.

Logical Instruction

Opcode	Operand	Description
XRI	8-bit data	XOR immediate with accumulator

oThe contents of the accumulator are XORed with the 8-bit data.

oThe result is placed in the accumulator.

oS, Z, P are modified to reflect the result.

oCY and AC are reset.

o**Example**: XRI 16H.

Logical Instruction

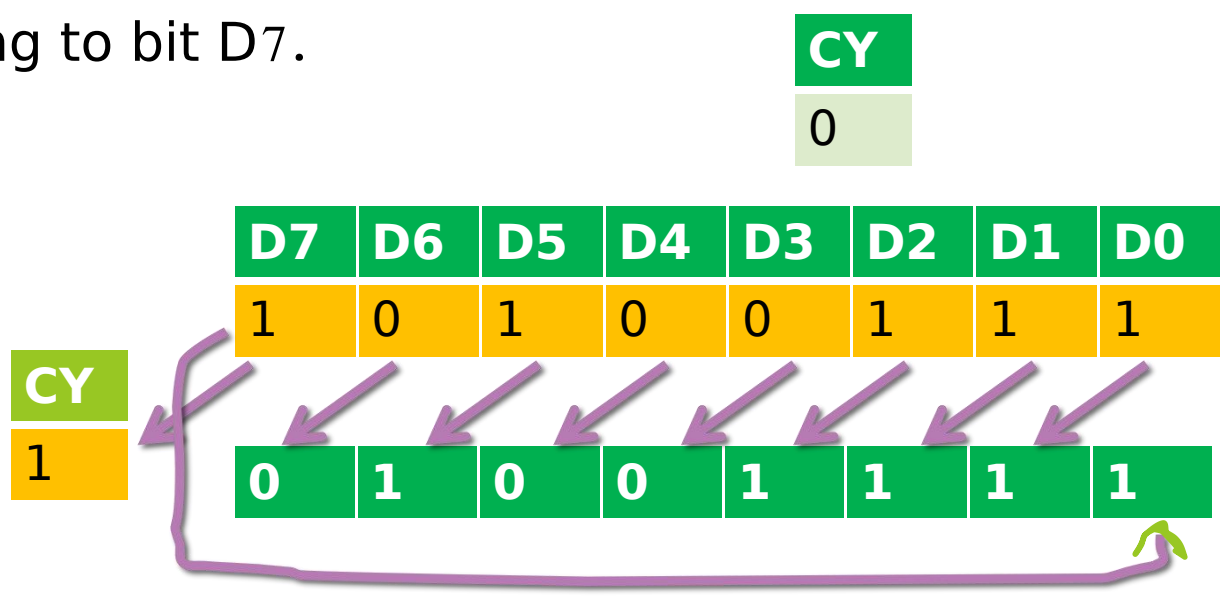
Opcode	Operand	Description
RLC	None	Rotate accumulator left

oEach binary bit of the accumulator is rotated left by one position.

oBit D_7 is placed in the position of D_0 as well as in the Carry f lag.

oCY is modified according to bit D_7.

oExample: RLC.



Logical Instruction

Opcode	Operand	Description
RRC	None	Rotate accumulator right

oEach binary bit of the accumulator is rotated right by one position.

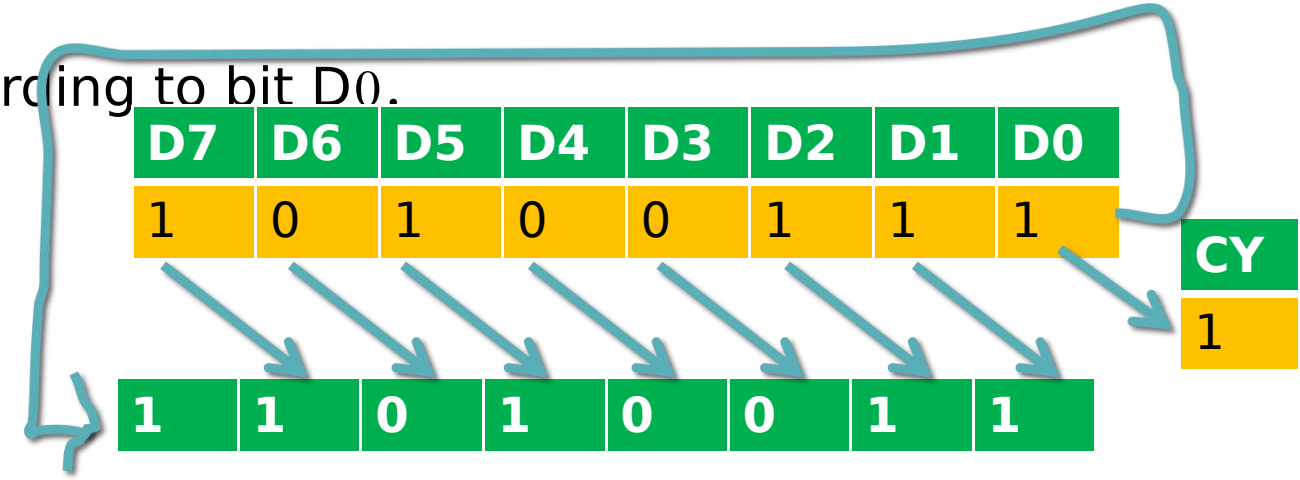
oBit D_0 is placed in the position of D_7 as well as in the Carry f lag.

oCY is modified according to bit D_0.

o**Example**: RRC.

Previous value

CY
0



Logical Instruction

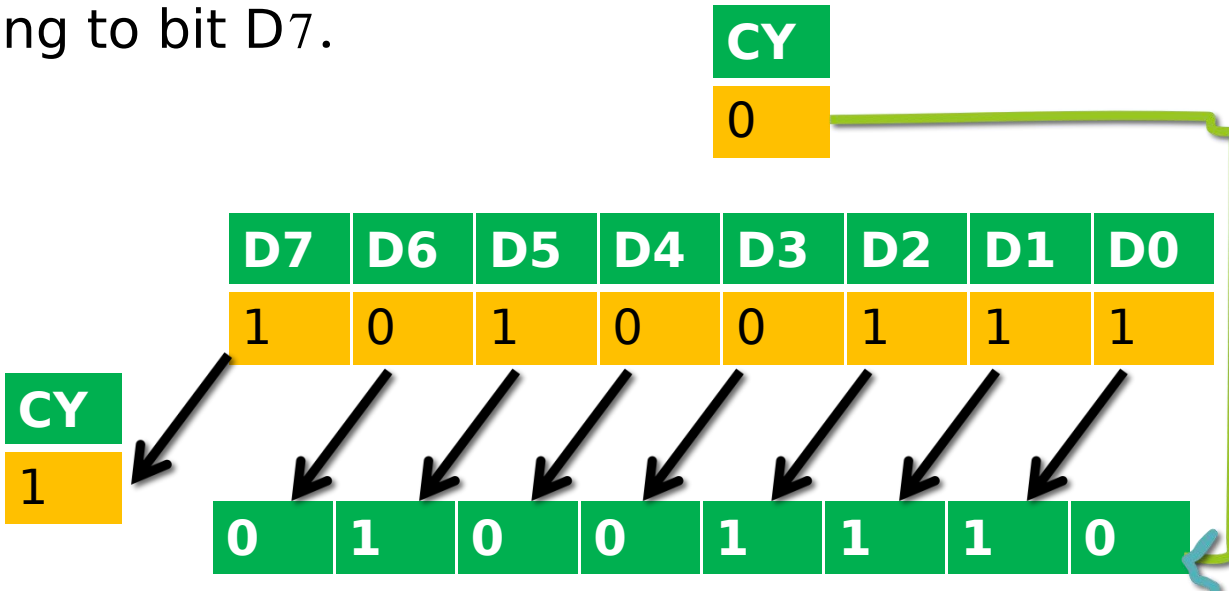
Opcode	Operand	Description
RAL	None	Rotate accumulator left through carry

oEach binary bit of the accumulator is rotated left by one position through the Carry flag.

oBit D7 is placed in the Carry flag, and the Carry f lag is placed in the least significant position D0.

oCY is modified according to bit D7.

o**Example**: RAL.



Logical Instruction

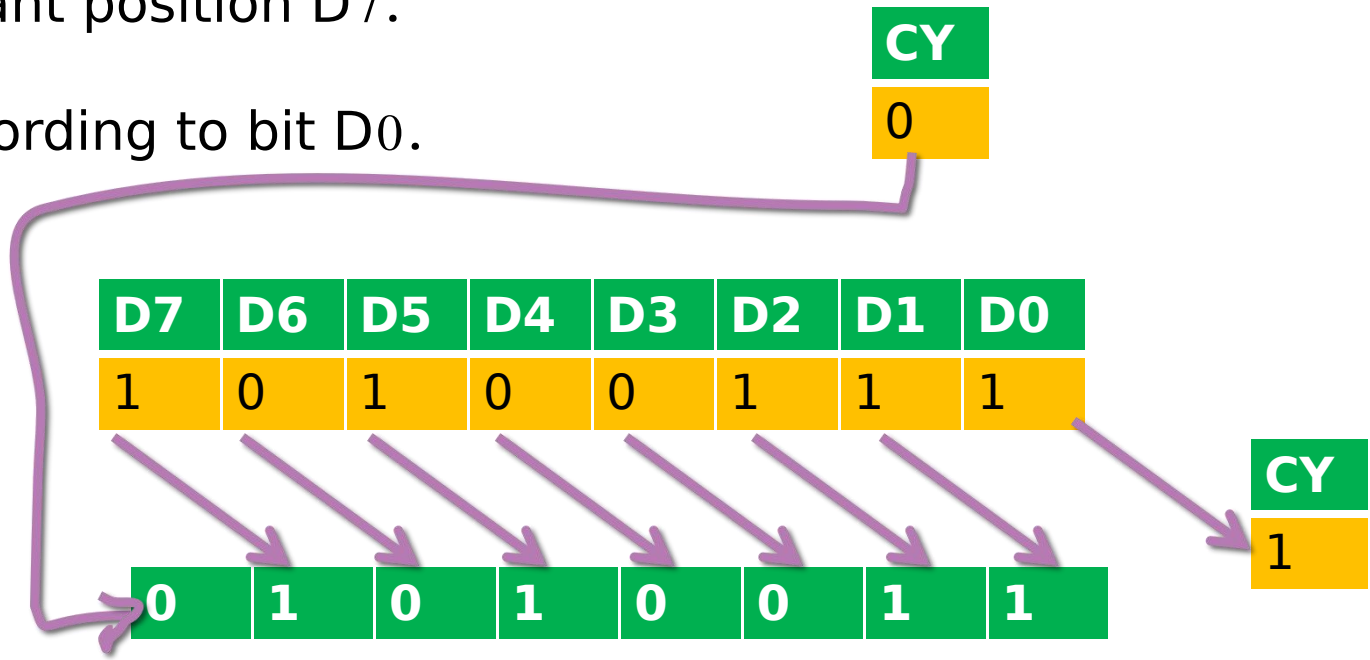
Opcode	Operand	Description
RAR	None	Rotate accumulator right through carry

oEach binary bit of the accumulator is rotated right by one position through the Carry f lag.

oBit D0 is placed in the Carry f lag, and the Carry f lag is placed in the most significant position D7.

oCY is modified according to bit D0.

o**Example**: RAR.



Logical Instruction

Opcode	Operand	Description
CMA	None	Complement accumulator

- The contents of the accumulator are complemented.

- No flags are affected.

- **Example**: CMA.

Logical Instruction

Opcode	Operand	Description
CMC	None	Complement carry

oThe Carry f lag is complemented.

oNo other f lags are affected.

o**Example**: CMC.

Logical Instruction

Opcode	Operand	Description
STC	None	Set carry

- The Carry f lag is set to 1.
- No other flags are affected.
- **Example**: STC.

Branching Instructions

Opcode	Operand	Description
JMP	16-bit address	Jump unconditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.

- **Example**: JMP 2094 H.

Branching Instruction

Opcode	Operand	Description
Jx	16-bit address	Jump conditionally

oThe program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified f lag of the PSW.

o**Example**: JZ 2094 H.

Jump Conditionally

Opcode	Description	Status Flags
JC	Jump if Carry	CY = 1
JNC	Jump if No Carry	CY = 0
JP	Jump if Positive	S = 0
JM	Jump if Minus	S = 1
JZ	Jump if Zero	Z = 1
JNZ	Jump if No Zero	Z = 0
JPE	Jump if Parity Even	P =

Branching Instruction

Opcode	Operand	Description
CALL	16-bit address	Call unconditionally

oThe program sequence is transferred to the memory location specified by the 16-bit address given in the operand.

oBefore the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack.

o**Example**: CALL 2094 H.

Branching Instruction

Opcode	Operand	Description
Cx	16-bit address	Call conditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified f lag of the PSW.

- Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto the stack.

- **Example**: CZ 2094 H.

Call Conditionally

Opcode	Description	Status Flags
CC	Call if Carry	CY = 1
CNC	Call if No Carry	CY = 0
CP	Call if Positive	S = 0
CM	Call if Minus	S = 1
CZ	Call if Zero	Z = 1
CNZ	Call if No Zero	Z = 0
CPE	Call if Parity Even	P = 1
CPO	Call if Parity Odd	P =

Branching Instruction

Opcode	Operand	Description
RET	None	Return unconditionally

oThe program sequence is transferred from the subroutine to the calling program.

oThe two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.

o**Example**: RET.

Branching Instruction

Opcode	Operand	Description
Rx	None	Return conditionally

oThe program sequence is transferred from the subroutine to the calling program based on the specified f lag of the PSW.

oThe two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.

o**Example**: RZ.

Return Conditionally

Opcode	Description	Status Flags
RC	Return if Carry	CY = 1
RNC	Return if No Carry	CY = 0
RP	Return if Positive	S = 0
RM	Return if Minus	S = 1
RZ	Return if Zero	Z = 1
RNZ	Return if No Zero	Z = 0
RPE	Return if Parity Even	P = 1
RPO	Return if Parity Odd	P = 0

Branching Instruction

Opcode	Operand	Description
RST	0 – 7	Restart (Software Interrupts)

- The RST instruction jumps the control to one of eight memory locations depending upon the number.

- These are used as software instructions in a program to transfer program execution to one of the eight locations.

- **Example**: RST 3.

Restart Address Table

Instructions	Restart Address
RST 0	0000 H
RST 1	0008 H
RST 2	0010 H
RST 3	0018 H
RST 4	0020 H
RST 5	0028 H
RST 6	0030 H
RST 7	0038 H

Control Instruction

Opcode	Operand	Description
NOP	None	No operation

oNo operation is performed.

oThe instruction is fetched and decoded but no operation is executed.

o**Example**: NOP

Control Instruction

Opcode	Operand	Description
HLT	None	Halt

- The CPU finishes executing the current instruction and halts any further execution.

- An interrupt or reset is necessary to exit from the halt state.

- **Example**: HLT

Control Instruction

Opcode	Operand	Description
DI	None	Disable interrupt

oThe interrupt enable f lip–f lop is reset and all the interrupts except the TRAP are disabled.

oNo f lags are affected.

o**Example**: DI

Control Instruction

Opcode	Operand	Description
EI	None	Enable interrupt

oThe interrupt enable f lip–f lop is set and all interrupts are enabled.

oNo flags are affected.

oThis instruction is necessary to re–enable the interrupts (except TRAP).

o**Example**: EI

Control Instruction

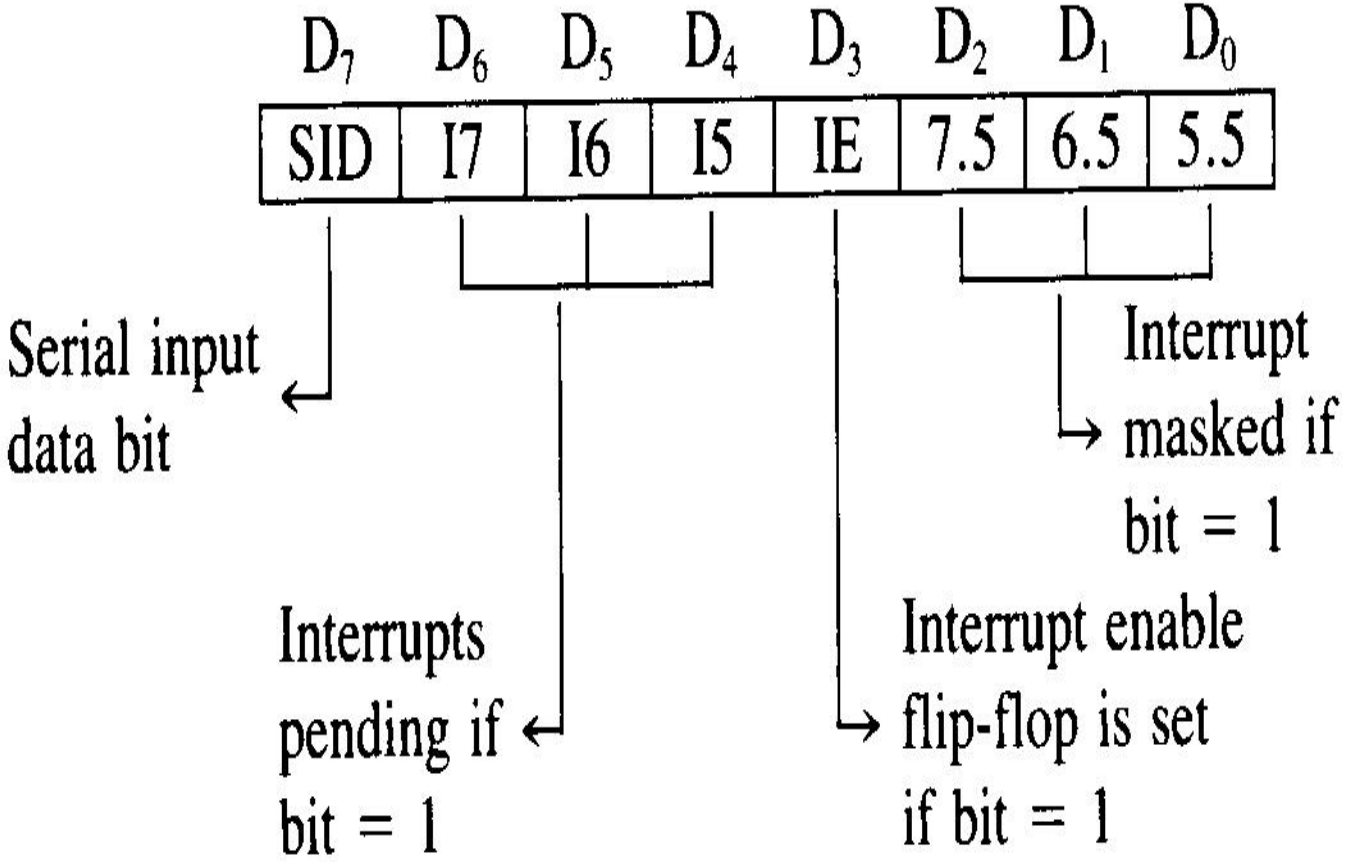
Opcode	Operand	Description
RIM	None	Read Interrupt Mask

oThis is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit.

oThe instruction loads eight bits in the accumulator with the following interpretations.

o**Example**: RIM

RIM Instruction

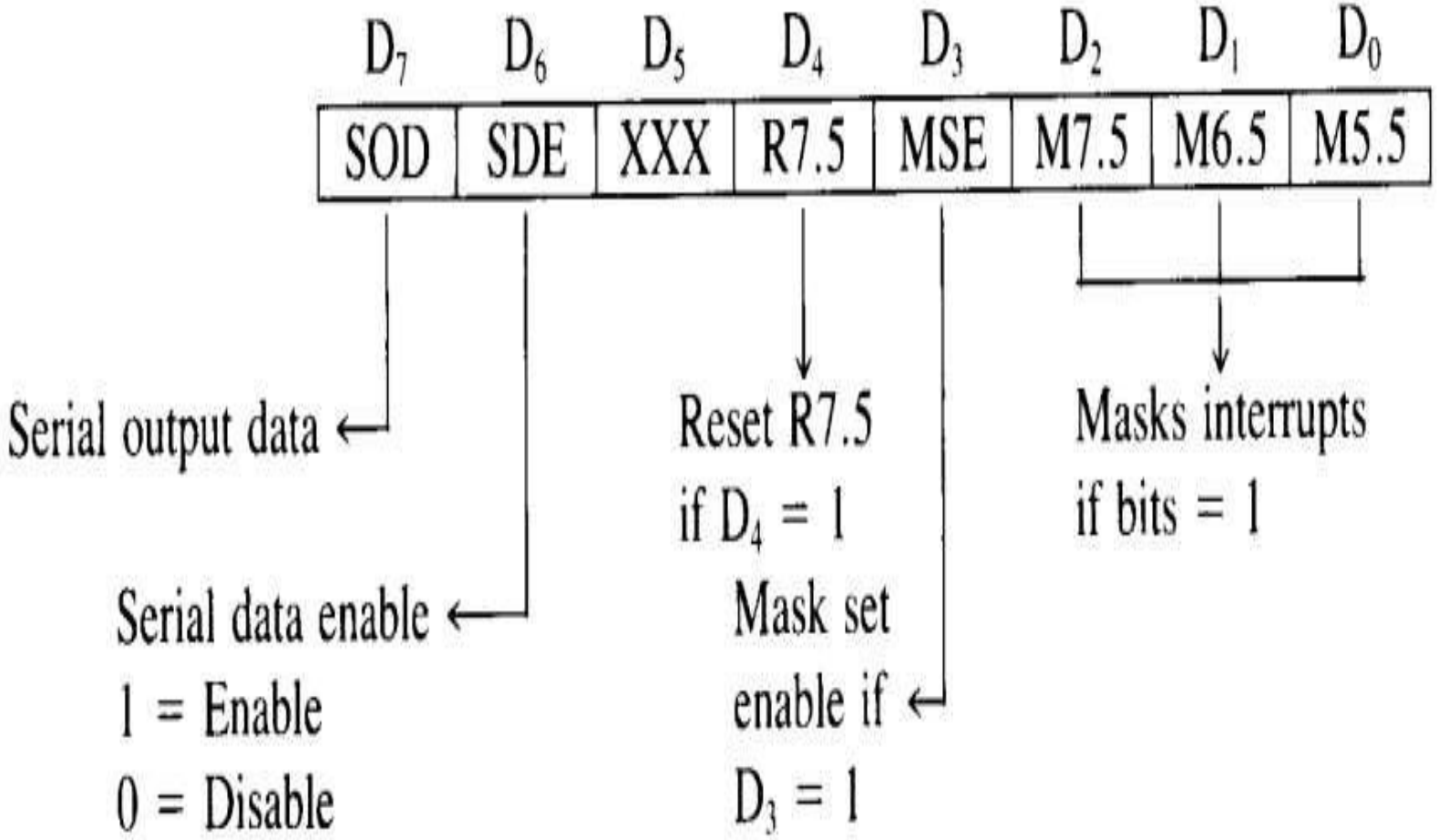


Control Instruction

Opcode	Operand	Description
SIM	None	Set Interrupt Mask

- This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output.
- The instruction interprets the accumulator contents as follows.
- **Example**: SIM

SIM Instruction



Addressing mode

- The 8085 has the following 5 different types of addressing Modes.

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

Immediate Addressing Mode

- 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, 37H – Move the data 37H given in the instruction to B register;
  - LXI D, 3100H.
  - ADI 60H

Direct Addressing Mode

- In direct addressing mode, the address of the data is specified in the instruction. The data will be in memory.

- Example
  - LDA 2050H – Load the data available in memory location 1050H in to accumulator
  - STA 2055H

Register Addressing Mode

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

- Example

  - MOV B, C – Move the content of B register to A register
  - ADD B

Register Indirect Addressing Mode

- In register indirect addressing mode, 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 – The memory data addressed by H L pair is moved to A register
  - LDAX D
  - STAX D

Implied Addressing Mode

- In implied addressing mode, the instruction itself specifies the data to be operated.

- Example

  - CMA – Complement the content of accumulator,
  - RAL – Rotate accumulator left through carry.

Identify the Addressing mode of following instructions

- ORA B
- MVI C,21H
- LHLD 2000H
- LDAX B
- LDA 2000H
- CMP B
- STC
- XCHG
- DAD
- DAA