

Noida Institute of Engineering and Technology, Greater Noida

Microprocessor (ACSE0405)

Unit: 2

8085 Instructions And
Programming Techniques

B.Tech 4th Semester



Khushboo
Assistant Professor
ECE



Evaluation Scheme

Sl. No.	Subject Codes	Subject Name	Periods			Evaluation Scheme				End Semester		Total	Credit
			L	T	P	CT	TA	TOTAL	PS	TE	PE		
1	AAS0402	Engineering Mathematics-IV	3	1	0	30	20	50		100		150	4
2	AASL0401	Technical Communication	2	1	0	30	20	50		100		150	3
3	ACSE0405	Microprocessor	3	0	0	30	20	50		100		150	3
4	ACSE0403A	Operating Systems	3	0	0	30	20	50		100		150	3
5	ACSE0404	Theory of Automata and Formal Languages	3	0	0	30	20	50		100		150	3
6	ACSE0401	Design and Analysis of Algorithm	3	1	0	30	20	50		100		150	4
7	ACSE0455	Microprocessor Lab	0	0	2				25		25	50	1
8	ACSE0453A	Operating Systems Lab	0	0	2				25		25	50	1
9	ACSE0451	Design and Analysis of Algorithm Lab	0	0	2				25		25	50	1
10	ACSE0459	Mini Project using Open Technology	0	0	2				50			50	1
11	ANC0402 / ANC0401	Environmental Science*/ Cyber Security*(Non Credit)	2	0	0	30	20	50		50		100	0
12		MOOCs** (For B.Tech. Hons. Degree)											
		GRAND TOTAL										1100	24

Subject Syllabus

Course Contents / Syllabus		
UNIT-I	8085 Microprocessor	8 Hours
Introduction to Microprocessor, Microprocessor evolution and types, Microprocessor architecture and its operation, Logic devices for interfacing, Pin diagram and internal architecture of 8085 Microprocessor, Example of an 8085 based computer, Instruction and data flow, timer and timing diagram, interrupt and machine cycle, Addressing modes.		
UNIT-II	8085 Instructions and Programming Techniques	8 Hours
Instruction sets, Instruction Classification: data transfer operations, arithmetic operations, logical operations, branching operations, machine control and assembler directives, writing assembly language programs, Programming techniques: looping, counting and indexing		
UNIT-III	Code Conversion and BCD Arithmetic	8 Hours
Counter and time delays, Illustrative program: Hexadecimal counter, zero-to-nine, (module ten) counter, generating pulse waveforms, Stack, Subroutine, Restart, Conditional call and return instructions, Advance subroutine concepts, Program: BCD-to-Binary conversion, Binary-to-BCD conversion, BCD-to-Seven segment code converter, Binary-to-ASCII and ASCII-to-Binary code conversion, BCD Addition, BCD Subtraction, Introduction to Advance instructions and Application, Multiplication		
UNIT-IV	Interfacing of I/O devices	8 Hours
Basic interfacing concepts, Memory interfacing, Interfacing output displays, Interfacing input devices, Memory mapped I/O, Interfacing keyboard and seven segment displays, The 8085 Interrupts, 8085 vector interrupts, 8259 programmable interrupt controller,		
UNIT-V	Programmable Peripheral IC's and 8086 Microprocessor	8 Hours
Peripheral Devices: 8255 programmable peripheral interface, 8253/8254 programmable timer/counter, 8237 DMA Controller, 8251 USART and RS232C. Introduction to 8086 microprocessors: Architecture of 8086 (Pin diagram, Functional block diagram, register organization), Addressing Modes		

Branch Wise Application

- **Microprocessor**, any of a type of miniature electronic device that contains the arithmetic, logic, and control circuitry necessary to perform the functions of a digital computer's central processing unit.
- This kind of integrated circuit can interpret and execute program instructions as well as handle arithmetic operations.

Course Objective

- The objective of this course is to understand basic concepts of Microprocessor based systems and able to do programming in Assembly Language of 8085. They will be able to learn and program Peripheral IC's.

At the end of this course students will able to:

- Apply a basic concept of digital fundamentals to Microprocessor based personal computer system.
- Analyze a detailed s/w & h/w structure of the Microprocessor.
- Illustrate how the different peripherals (8085/8086) are interfaced with Microprocessor.
- Analyze the properties of Microprocessors (8085/8086).
- Evaluate the data transfer information through serial & parallel ports.

Program Outcomes

- **Program Outcomes** are narrow statements that describe what the students are expected to know and would be able to do upon the graduation.
 - These relate to the skills, knowledge, and behavior that students acquire through the programmed.
-
- | | |
|---|------------------------------------|
| 1. Engineering knowledge | 9. Individual and team work |
| 2. Problem analysis | 10. Communication |
| 3. Design/development of solutions | 11. Project management and finance |
| 4. Conduct investigations of complex problems | 12. Life-long learning |
| 5. Modern tool usage | |
| 6. The engineer and society | |
| 7. Environment and sustainability | |
| 8. Ethics | |

CO-PO Mapping

Course Outcome	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
ACSE0405.1	3	1	2		1	-	-	-	-	-	-	3
ACSE0405.2	3	1	-		1	-	-	-	-	-	-	2
ACSE0405.3	3	1	3		1	-	-	-	-	-	-	2
ACSE0405.4	3	1	-		1	-	-	-	-	-	-	2
ACSE0405.5	3	1	2		1	-	-	-	-	-	-	2
Average	3	1	2.3		1	-	-	-	-	-	-	2.1

PSOs Mapping

Sr. No.	Course Outcome	PSO1	PSO2	PSO3
1	ACSE0405.1	3	2	1
2	ACSE0405.2	3	2	1
3	ACSE0405.3	3	2	1
4	ACSE0405.4	3	2	1
5	ACSE0405.5	3	2	2
	Average	3	2	1.2

End Semester Question Paper Template

B.Tech (Semester IV Theory Examination 2021-22)

Total Marks : 100

Note: Attempt all sections. If require any missing data, then choose suitably. **Time: 3 hours**

Section A

1. Attempt all questions in brief.

2 X 10 = 20

Q. No.	Question	Marks	CO
a. to j		2	

Section B

2. Attempt any three of the following

3 X 10 = 30

Q. No.	Question	Marks	CO
a to e		10	

Section C

Question no. 3,4,5,6,7. Attempt any one of the following

1 X 10 = 10

a		10	
b		10	

Prerequisites

- Basic knowledge of 8085 microprocessor
- Knowledge of its architecture and operations

Introduction to microprocessor

- 8085 is an **8-bit microprocessor** as it operates on 8 bits at a time and is created with **N-MOS technology**.
This microprocessor exhibits some unique characteristics and this is the reason it still holds popularity among the microprocessors.
- Basically, 8085 was the **first commercially successful microprocessor by Intel**.

Unit Content

- Course Objective
- Unit Objective
- Course Outcome
- CO and PO Mapping
- Topic Objective
- Prerequisite
- Instruction sets
- Instructions classification
- Data transfer operations
- Arithmetic operations
- Logical operations
- Branching operations
- Machine control and assembler directives
- Writing assembly language programs
- Programming techniques
- Daily quiz & MCQs
- Old Question Papers
- Recap
- Video Links
- Weekly Assignments
- References

Unit Objective

1. To understand the instructions sets of 8085 microprocessor
2. To learn writing assembly language programs
3. To discuss concept of classification of instructions
4. To acquire the knowledge of programming techniques

Instruction Set of 8085

Name of the Topic	Objective of the topic	Mapping with CO
Instruction Set of 8085	To understand the instruction of 8085 microprocessor.	CO2

Prerequisite

- Basic concept of Digital Logic.
- Understanding of memory unit & data size.

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
- Control Instructions

Video Link



<https://www.youtube.com/watch?v=ekkoleonyzg>

- What is true about microprocessor?
 - A. Microprocessor is a controlling unit of a micro-computer
 - B. It is fabricated on a small chip capable of performing ALU (Arithmetic Logical Unit) operations
 - C. It also communicate with the other devices connected to it.
 - D. All of the above**
- Microprocessor consists of?
 - A. ALU
 - B. register array
 - C. control unit
 - D. All of the above**

Daily Quiz

- Which control instruction is followed by an un-conditional branch instructions so as to branch to a single location from the double ones with respect to specified status-bit condition?
 - A. jump instruction
 - B. branch instruction
 - C. skip instruction**
 - D. return- from-subroutine instructions
- Which category of microprocessor instructions detect the status conditions in registers and accordingly exhibit the variations in program sequence on the basis of detected results?
 - A. Transfer Instructions
 - B. Operation Instructions
 - C. Control Instructions**
 - D. All of the above

Recap

- Detail of various types of instruction set is discussed.

Name of the Topic	Objective of the topic	Mapping with CO
Data Transfer Operation	To learn different types of data transfer instructions.	CO2

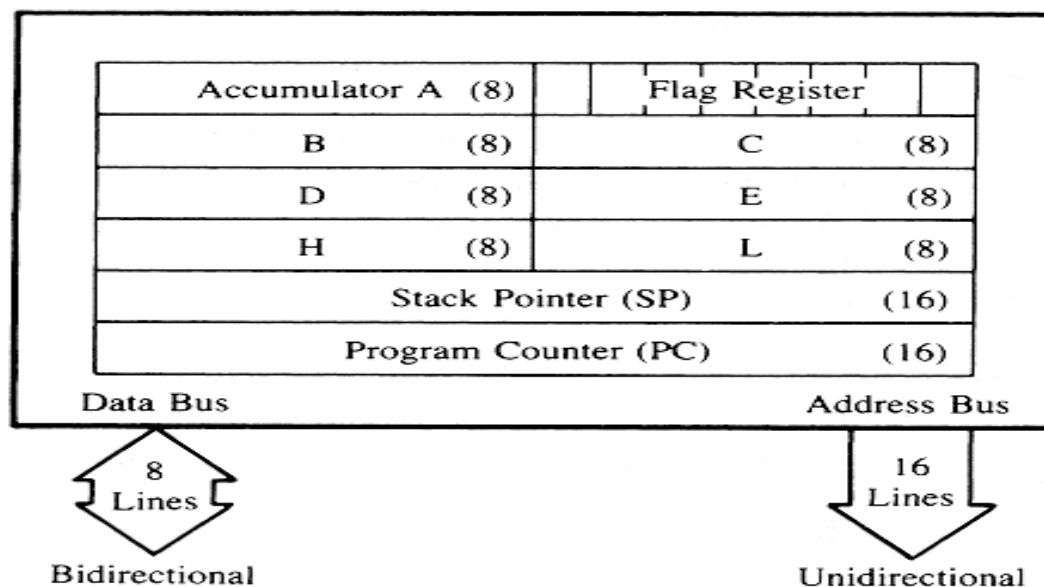
Prerequisite

- Basic concept of Digital Logic.
- Overview of register set of 8085 microprocessor.

Data Transfer Operation

Data Transfer Instructions

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



Data Transfer Instructions

- This instruction perform following six operation.
 - Load an 8 bit number in a register.
 - Load 16 bit number in a register pair.
 - Copy from register to register.
 - Copy between register & memory.
 - Copy between I/O & accumulator.
 - Copy between registers & stack memory.

Data Transfer Operation

Data Transfer Instructions

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 B, C or MOV B, M.

Data Transfer Operation

Data Transfer Instructions

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, 57H or MVI M, 57H

Data Transfer Operation

Data Transfer Instructions

Opcode	Operand	Description
LDA	16-bit address	Load Accumulator

- 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 2034H

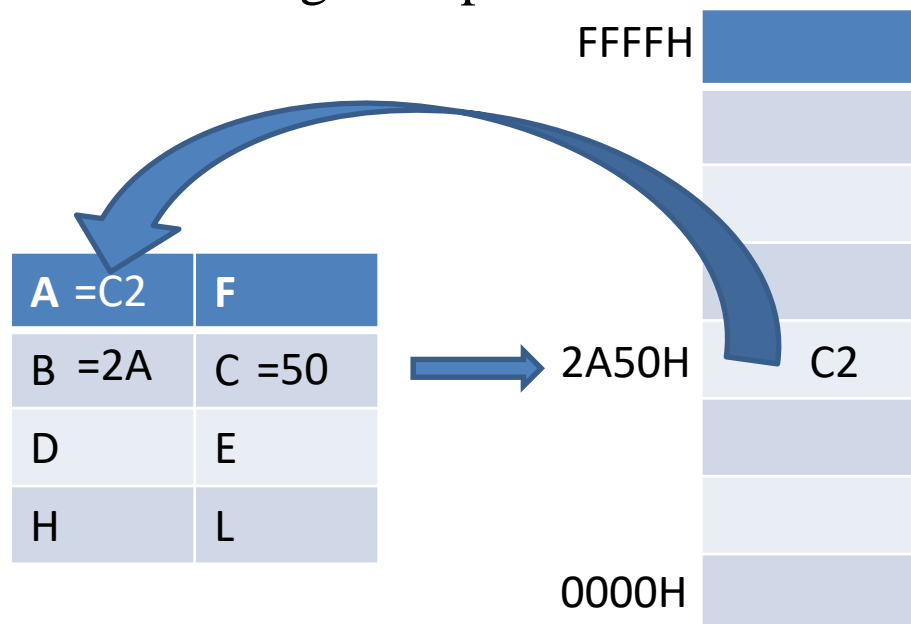
Data Transfer Operation

Data Transfer Instructions

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 B**



Data Transfer Instructions

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

- This instruction loads 16-bit data in the register pair.
- **Example:** LXI H, 2034 H

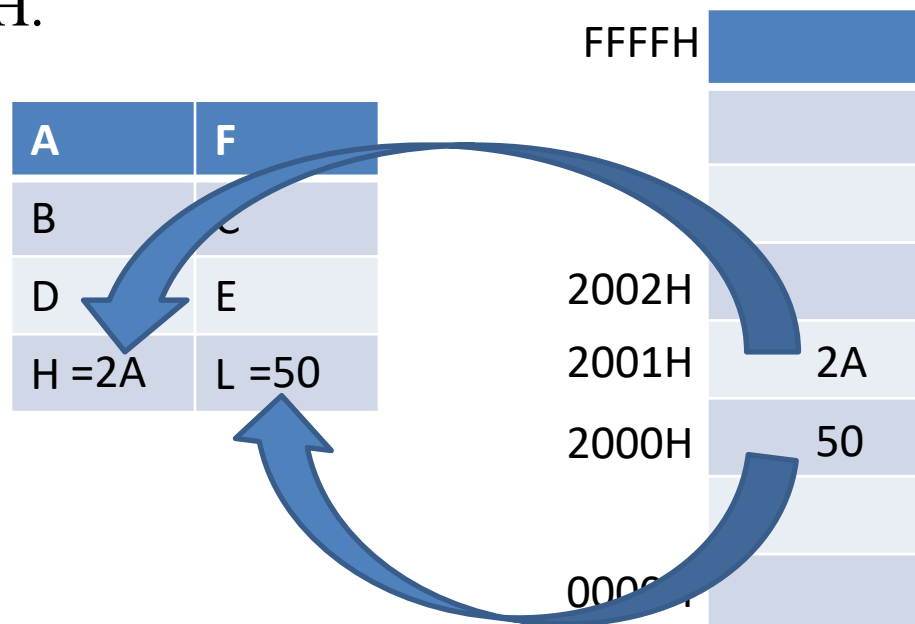
Data Transfer Operation

Data Transfer Instructions

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 2000 H



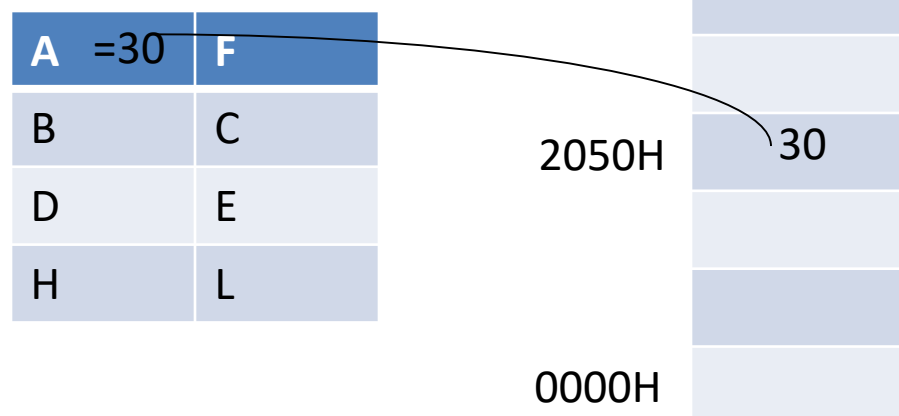
Data Transfer Operation

Data Transfer Instructions

Opcode	Operand	Description
STA	16-bit address	Store accumulator direct

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

- Example:** STA 2050 H



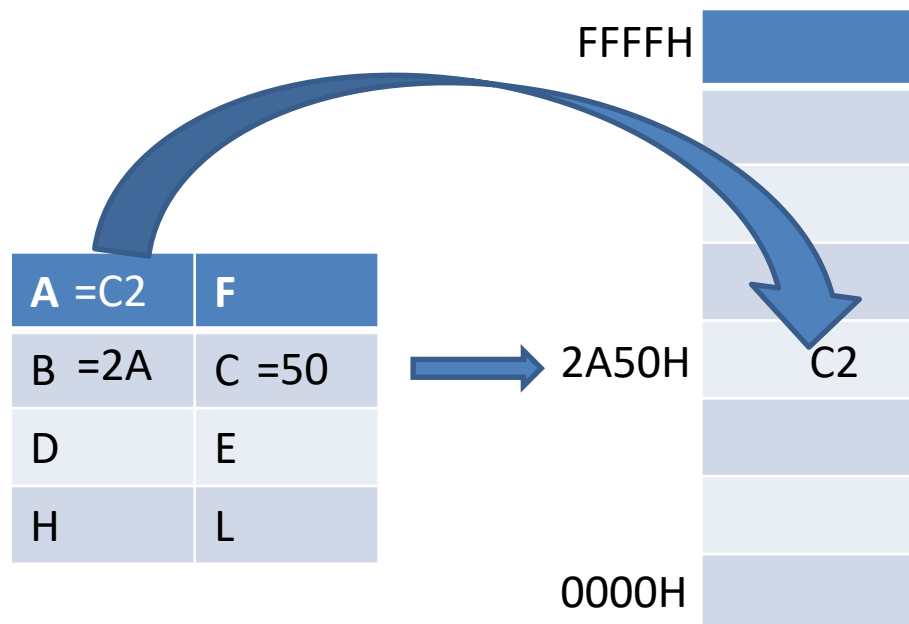
Data Transfer Operation

Data Transfer Instructions

Opcode	Operand	Description
STAX	Reg. pair	Store accumulator indirect

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

- Example: STAX B**

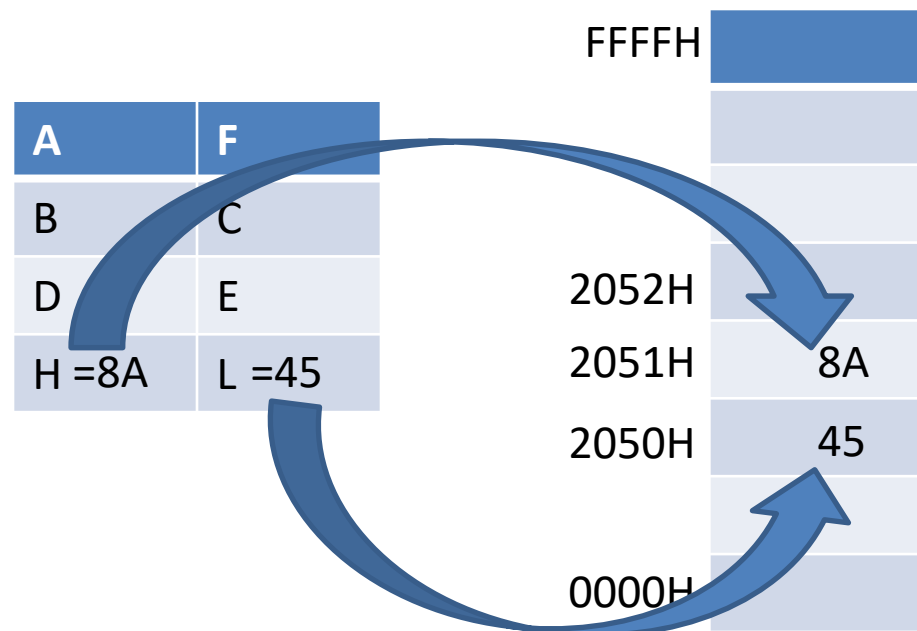


Data Transfer Operation

Data Transfer Instructions

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

- The contents of register L are stored into memory location specified by the 16-bit address.
- The contents of register H are stored into the next memory location.
- Example:** SHLD 2050 H



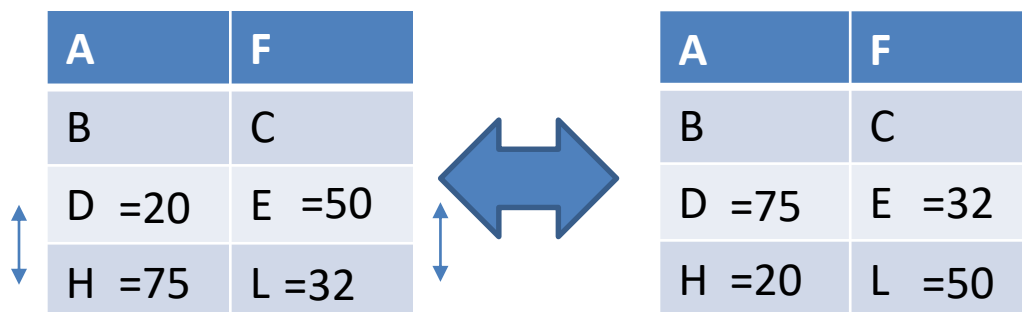
Data Transfer Operation

Data Transfer Instructions

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

- The contents of register H are exchanged with the contents of register D.
- The contents of register L are exchanged with the contents of register E.

- Example: XCHG**



Data Transfer Operation

Data Transfer Instructions

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

- This instruction loads the contents of H-L pair into SP.
- **Example:** SPHL

Data Transfer Operation

Data Transfer Instructions

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

- The contents of L register are exchanged with the location pointed out by the contents of the SP.
- The contents of H register are exchanged with the next location (SP + 1).
- **Example: XTHL**

Data Transfer Operation

Data Transfer Instructions

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

- The contents of registers H and L are copied into the program counter (PC).
- The contents of H are placed as the high-order byte and the contents of L as the low-order byte.
- **Example: PCHL**

Data Transfer Operation

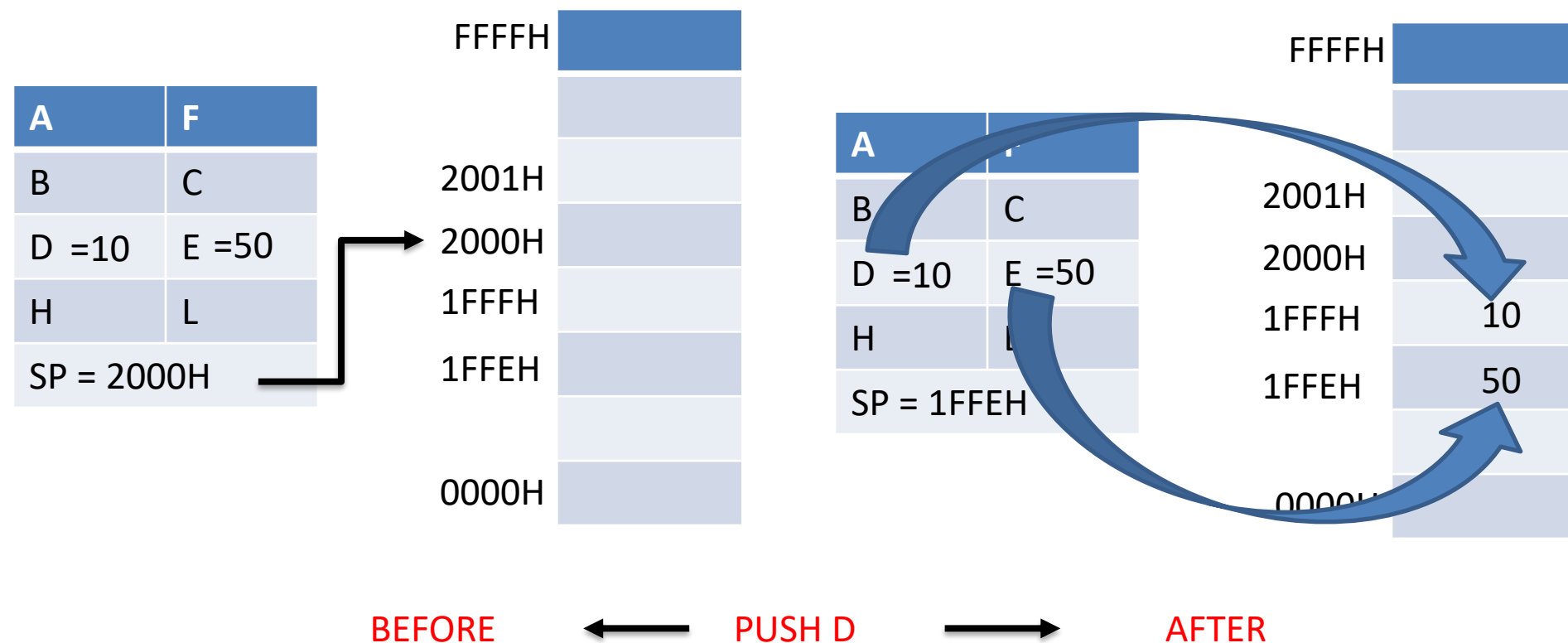
Data Transfer Instructions

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

- The contents of register pair are copied onto stack.
- SP is decremented and the contents of high-order registers (B, D, H, A) are copied into stack.
- SP is again decremented and the contents of low-order registers (C, E, L, Flags) are copied into stack.
- **Example:** PUSH D

Data Transfer Operation

Example : PUSH D



Data Transfer Operation

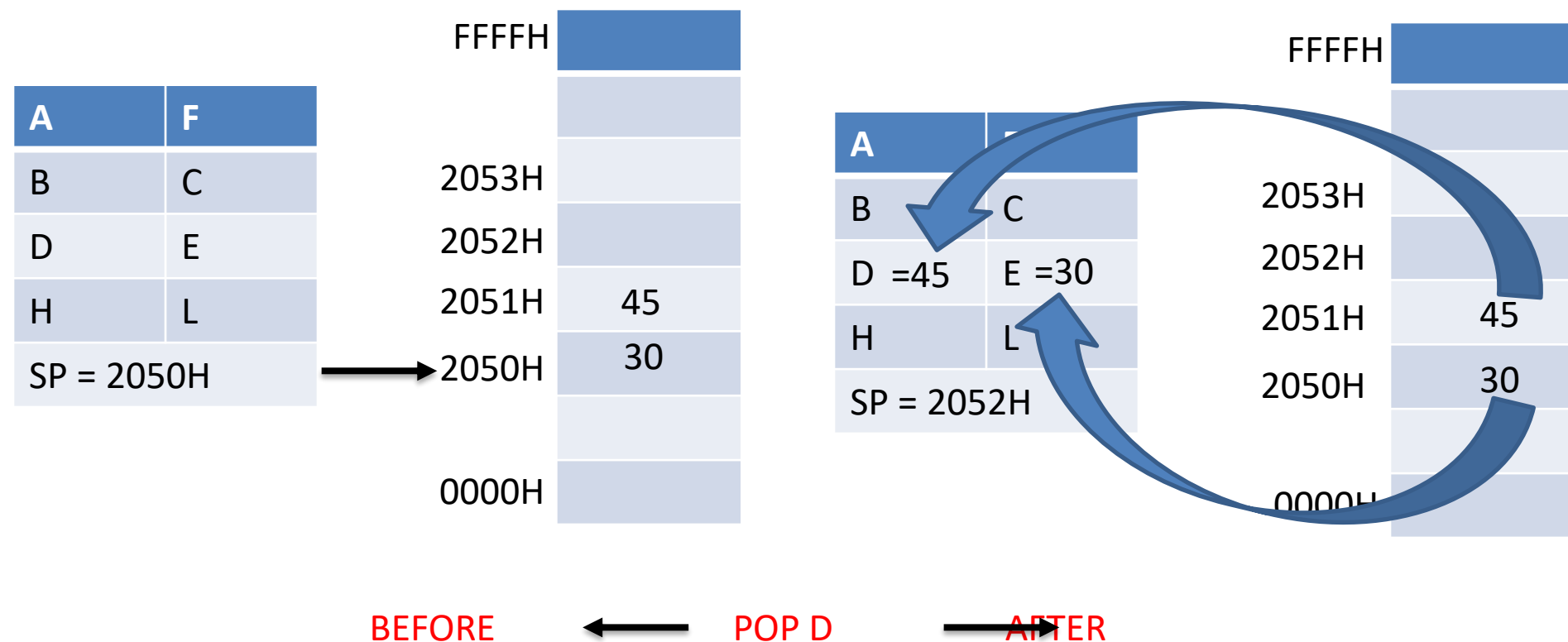
Data Transfer Instructions

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

- The contents of top of stack are copied into register pair.
- The contents of location pointed out by SP are copied to the low-order register (C, E, L, Flags).
- SP is incremented and the contents of location are copied to the high-order register (B, D, H, A).
- **Example:** POP H

Data Transfer Operation

Example : POP D



Data Transfer Operation

Data Transfer Instructions

Opcode	Operand	Description
OUT	8-bit port address	Copy data from accumulator to a port with 8-bit address

- The contents of accumulator are copied into the I/O port.
- **Example:** OUT 78 H

Data Transfer Operation

Data Transfer Instructions

Opcode	Operand	Description
IN	8-bit port address	Copy data to accumulator from a port with 8-bit address

- The contents of I/O port are copied into accumulator.
- **Example:** IN 8C H



<https://www.youtube.com/watch?v=ekkoleonyzg>

Daily Quiz

- DAD is for Double ADD. DAD B instruction in 8085 is used for ?
 - a) **Content of BC is added HL Result is in HL.**
 - b) Content of BC is added to Accumulator
 - c) Content of BC is added to DE
- DAA instruction is used for _____.
 - a) Double Add Accumulator.
 - b) **Decimal Adjust Accumulator.**
 - c) Decrement Accumulator
- To save accumulator value and the flag register on to the stack, which of the following instructions is used?
 - a) **PUSH PSW**
 - b) PUSH A
 - c) PUSH SP
 - d) POP PSW

- What is the content of A at the end of this program?

MVI A, 06H

RLC

MOV B, A

RLC

RLC

ADD B

a) **3Ch**

b) 18h

c) 0Ch

d) 00h

- In 8085, HLT opcode means:

a) Remain idle for 10 seconds.\

b) Remain idle for 0.1 seconds

c) **End of Program**

Recap

- Data transfer instructions of 8085 microprocessor is discussed.

Arithmetic operations

Name of the Topic	Objective of the topic	Mapping with CO
Arithmetic operations	To understand the arithmetic instructions of 8085 microprocessor.	CO2

Prerequisite

- Basic concept of Digital Logic.
- Knowledge of Boolean Algebra.
- Register set of 8085 microprocessor.

Arithmetic Instructions

Addition

- Any 8-bit number, or the contents of register, or the contents of memory location can be added to the contents of accumulator.
- The result (sum) is stored in the accumulator.
- No two other 8-bit registers can be added directly.

Note: The contents of register B cannot be added directly to the contents of register C.

Arithmetic Instructions

Subtraction

- Any 8-bit number, or the contents of register, or the contents of memory location can be subtracted from the contents of accumulator.
- The result is stored in the accumulator.
- Subtraction is performed in 2's complement form.
- If the result is negative, it is stored in 2's complement form.
- No two other 8-bit registers can be subtracted directly.

Arithmetic Instructions

Increment / Decrement

- The 8-bit contents of a register or a memory location can be incremented or decremented by 1.
- The 16-bit contents of a register pair can be incremented or decremented by 1.
- Increment or decrement can be performed on any register or a memory location.

Arithmetic operations

Arithmetic Instructions

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

- The contents of register or memory are added to the contents of accumulator. The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of the addition.
- **Example:** ADD B ($A+B \longrightarrow A$)

ADD M ($A+ M(HL) \longrightarrow A$)

Arithmetic operations

Arithmetic Instructions

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

- The contents of register or memory and Carry Flag (CY) are added to the contents of accumulator. The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of the addition.

Example: $\text{ADC B (A+B + CY } \longrightarrow \text{A)}$

$\text{ADC M (A+M(HL) + CY } \longrightarrow \text{A)}$

Arithmetic operations

Arithmetic Instructions

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

- The 8-bit data is added to the contents of accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of the addition.
- **Example:** ADI 45 H ($A + 45 \rightarrow A$)

Arithmetic Instructions

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

- The 8-bit data and the Carry Flag (CY) are added to the contents of accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of the addition.
- **Example:** ACI 45 H ($A + 45 + CY \longrightarrow A$)

Arithmetic Instructions

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 flags are changed.
- **Example:** DAD B (HL + BC \longrightarrow HL)

Arithmetic Instructions

PROBLEM

DAD D

Let D=30H, E=20H
H= 1AH, L = 42 H

SOLUTION

H= 4AH & L = 62H

Arithmetic operations

Arithmetic Instructions

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

- The contents of the register or memory location are subtracted from the contents of the accumulator. The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of subtraction.

Example: SUB B ($A - B \longrightarrow A$)

 SUB M ($A - M(HL) \longrightarrow A$)

Arithmetic operations

Arithmetic Instructions

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

- The contents of the register or memory location and Borrow Flag (i.e. CY) are subtracted from the contents of the accumulator. The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of subtraction.

Example: $\text{SBB B } (A - B - \text{CY} \longrightarrow A)$

$\text{SBB M } (A - \text{M (HL)} - \text{CY} \longrightarrow A)$

Khushboo Microprocessor UNIT- 2

Arithmetic operations

Arithmetic Instructions

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

- The 8-bit data is subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of subtraction.
- **Example:** SUI 54 H ($A - 54 \longrightarrow A$)

Arithmetic Instructions

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

- The 8-bit data and the Borrow Flag (i.e. CY) is subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of subtraction.
- **Example:** SBI 45 H ($A - 45 - CY \longrightarrow A$)

Arithmetic operations

Arithmetic Instructions

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

- The contents of register or memory location are incremented by 1. The result is stored in the same place.
- If the operand is a memory location, its address is specified by the contents of H-L pair.

Example: INR B ($B + 1 \longrightarrow B$)

INR M ($M + 1 \longrightarrow M$)

Arithmetic Instructions

Opcode	Operand	Description
INX	R	Increment register pair by 1

- The contents of register pair are incremented by 1.
- The result is stored in the same place.
- **Example:** INX H ($HL+1 \longrightarrow HL$)

Arithmetic operations

Arithmetic Instructions

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

- The contents of register or memory location are decremented by 1. The result is stored in the same place.
- If the operand is a memory location, its address is specified by the contents of H-L pair.

Example: DCR B ($B - 1 \longrightarrow B$)

DCR M ($M - 1 \longrightarrow M$)

Arithmetic Instructions

Opcode	Operand	Description
DCX	R	Decrement register pair by 1

- The contents of register pair are decremented by 1.
- The result is stored in the same place.
- **Example:** DCX H ($HL - 1 \longrightarrow HL$)

Arithmetic Instructions

Opcode	Operand	Description
DAA		Decimal Adjust Accumulator (BCD format)

- It converts hexadecimal to BCD no.
- This instruction is used after addition instruction.

Operation:

- If lower nibble is >9 or $AC = 1$ then add 06.
- If higher nibble is >9 or $CY = 1$ then add 06.

Arithmetic Instructions

DAA Instruction Example

1) $A=34H$ & $Data = 48H$

2) $A=37H$ & $Data = 15H$

3) $A=85H$ & $Data = 68H$



<https://www.youtube.com/watch?v=ekcoleonyzg>

- What is the content of PC Register at the end of the following program?

LXI H 8A79h

MOV A L

ADD H

DAA

MOV H A

PCHL

- a) 2304 b) **6979** c) 7100 d) 8255
- What does microprocessor speed depends on?
 - a) Clock
 - b) Data bus width
 - c) Address bus width
 - d) **All of these**

- What is the status of z flag, cy flag, sign flag at the end of this program?

MVI A, 02H

MVI B, 03H

ADD B

XRA A

a) **1,0,0** b) 0,1,0 c) 1,0,0 d) 1,0,1

- What is the content of Register A at the end?

XRA A

MVI B, 4DH

SUI 4FH

ANA B

HLT

a) 00h b) **01h** c) 4Dh

Weekly Assignment

1. Specify the contents of the A,B,C,D,E,H,L ,M registers as each if the following instructions is being executed.

MVI C,FFH

LXI H,2030H

LXI D,7050H

MOV M,C

XCHG

LDAX D

HLT

2. Specify the number of times the following loops are executed

LXI B,1000H

LOOP: DCX B

NOP

JNZ LOOP

Weekly Assignment

3. Specify the number of times the following loops are executed

MVI A,17H

LOOP: RAL

ORA A

JNC LOOP

4. Explain the meaning of the following program

LXI H,0000H

DAD SP

MOV A,H

OUT PORT1

MOV A,L

OUT PORT2

Recap

- Arithmetic instructions of 8085 microprocessor with example is discussed.

Name of the Topic	Objective of the topic	Mapping with CO
Logic Operations	To understand logic instructions of 8085 microprocessor.	CO2

Prerequisite

- Basic concept of Digital Logic
- Knowledge of Boolean Algebra
- Register set of 8085 microprocessor.

Logical Instructions

- These instructions perform logical operations on data stored in registers, memory and status flags.
- The logical operations are:
 - AND
 - OR
 - XOR
 - Rotate
 - Compare
 - Complement

AND, OR, XOR

- Any 8-bit data, or the contents of register, or memory location can logically have
 - AND operation
 - OR operation
 - XOR operationwith the contents of accumulator.
- The result is stored in accumulator.

Rotate

- Each bit in the accumulator can be shifted either left or right to the next position.

Compare

- Any 8-bit data, or the contents of register, or memory location can be compares for:
 - Equality
 - Greater Than
 - Less Thanwith the contents of accumulator.
- The result is reflected in status flags.

Complement

- The contents of accumulator can be complemented.
- Each 0 is replaced by 1 and each 1 is replaced by 0.

Logical Instructions

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

- The contents of the accumulator are logically ANDed with the contents of register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation.
- CY is reset, AC is set.
- **Example:** ANA B or ANA M.

Logical Instructions

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

- The contents of the accumulator are logically ANDed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY is reset, AC is set.
- **Example:** ANI 86H.

Logical Instructions

Opcode	Operand	Description
ORA	R 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.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example:** ORA E or ORA M.

Logical Instructions

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

- The contents of the accumulator are logically ORed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example:** ORI 20H.

Logical Instructions

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

- The contents of the accumulator are XORed with the contents of the register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation.
- CY and AC are reset.
- **Example:** XRA D or XRA M.

Logical Instructions

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

- The contents of the accumulator are XORed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example:** XRI 23H.

Logic Operations

Instruction	Sign (S)	Zero (Z)	Auxillary Carry (AC)	Parity (P)	Carry (CY)
ANA R	√	√	1	√	0
ANI 8 bit data	√	√	1	√	0
ORA R	√	√	0	√	0
ORI 8 bit data	√	√	0	√	0
XRA R	√	√	0	√	0
XRI 8 bit data	√	√	0	√	0

√ - Will change according to result

Logical Instructions

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

- The contents of the operand (register or memory) are compared with the contents of the accumulator.
- Both contents are preserved .
- The result of the comparison is shown by setting the flags of the PSW as follows:

Logical Instructions

- if $(A) < (\text{reg/mem})$: carry flag is set
- if $(A) = (\text{reg/mem})$: zero flag is set
- if $(A) > (\text{reg/mem})$: carry and zero flags are reset.

Example: `CMP C` or `CMP D` or `CMP M`

Logical Instructions

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

- The 8-bit data is compared with the contents of accumulator.
- The values being compared remain unchanged.
- The result of the comparison is shown by setting the flags of the PSW as follows:

Logical Instructions

- if $(A) < \text{data}$: carry flag is set
- if $(A) = \text{data}$: zero flag is set
- if $(A) > \text{data}$: carry and zero flags are reset

Example: CPI A6H

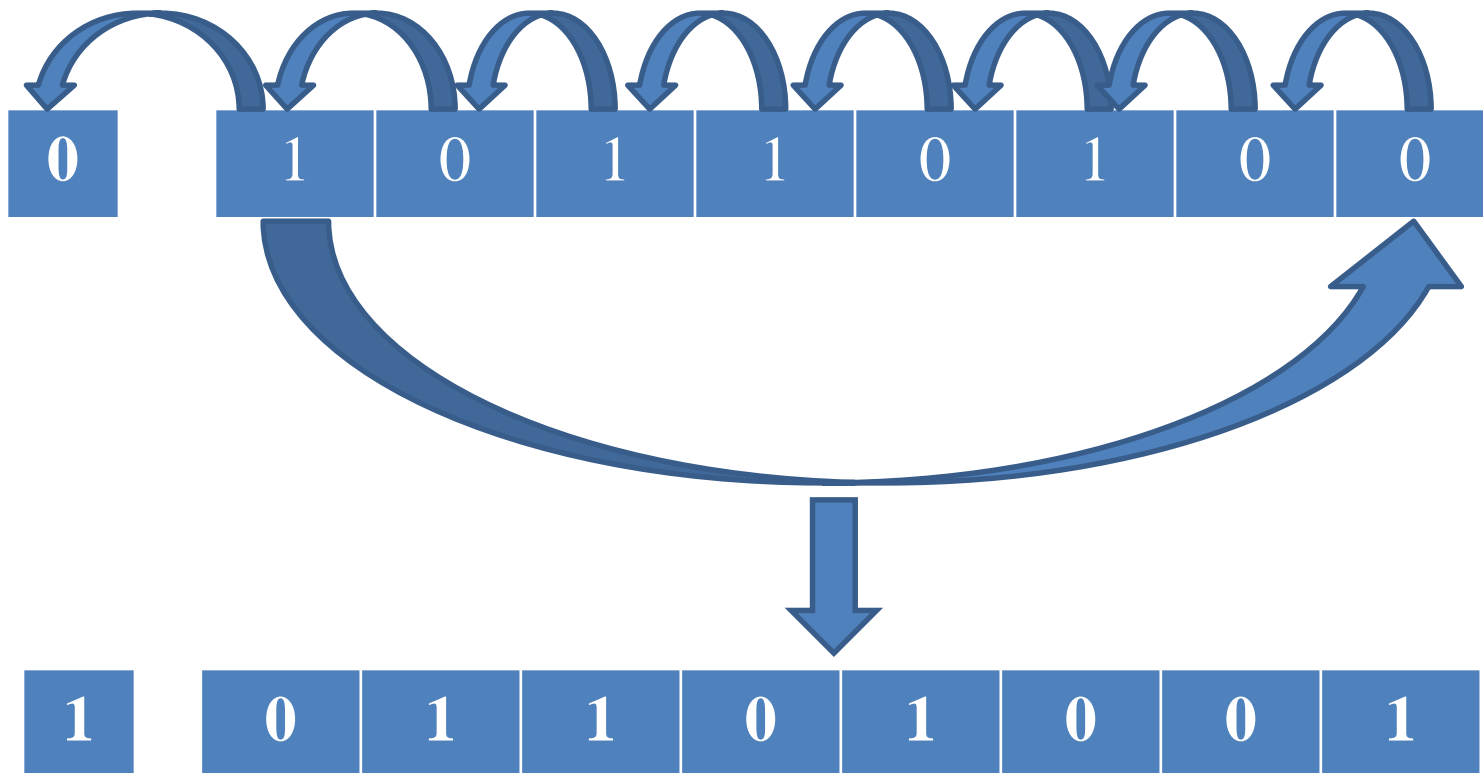
Logical Instructions

Opcode	Operand	Description
RLC	None	Rotate accumulator left

- Each binary bit of the accumulator is rotated left by one position.
- Bit D7 is placed in the position of D0 as well as in the Carry flag.
- CY is modified according to bit D7.
- S, Z, P, AC are not affected.

Example: RLC

Logical Instructions



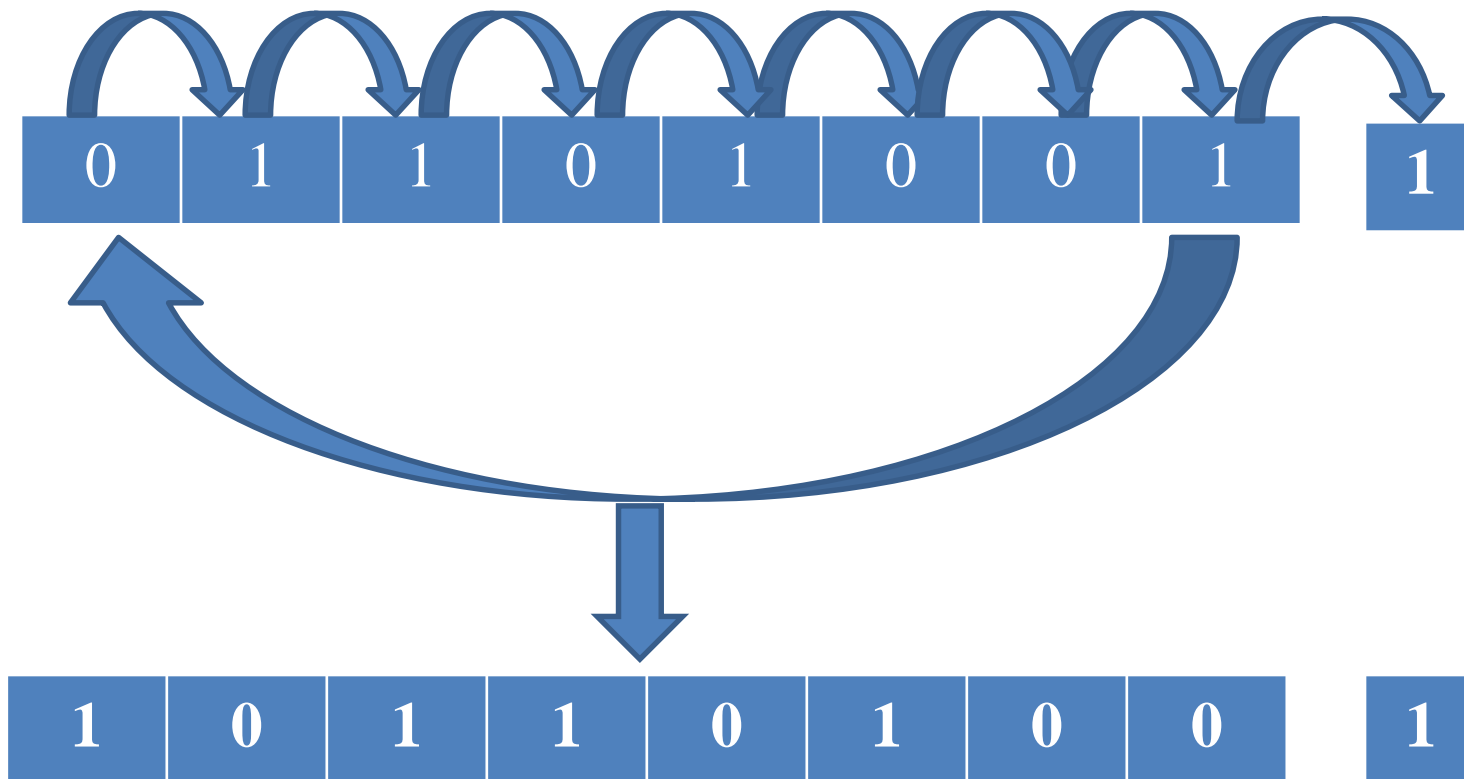
Logical Instructions

Opcode	Operand	Description
RRC	None	Rotate accumulator right

- Each binary bit of the accumulator is rotated right by one position.
- Bit D0 is placed in the position of D7 as well as in the Carry flag.
- CY is modified according to bit D0.
- S, Z, P, AC are not affected.

Example: RRC

Logical Instructions

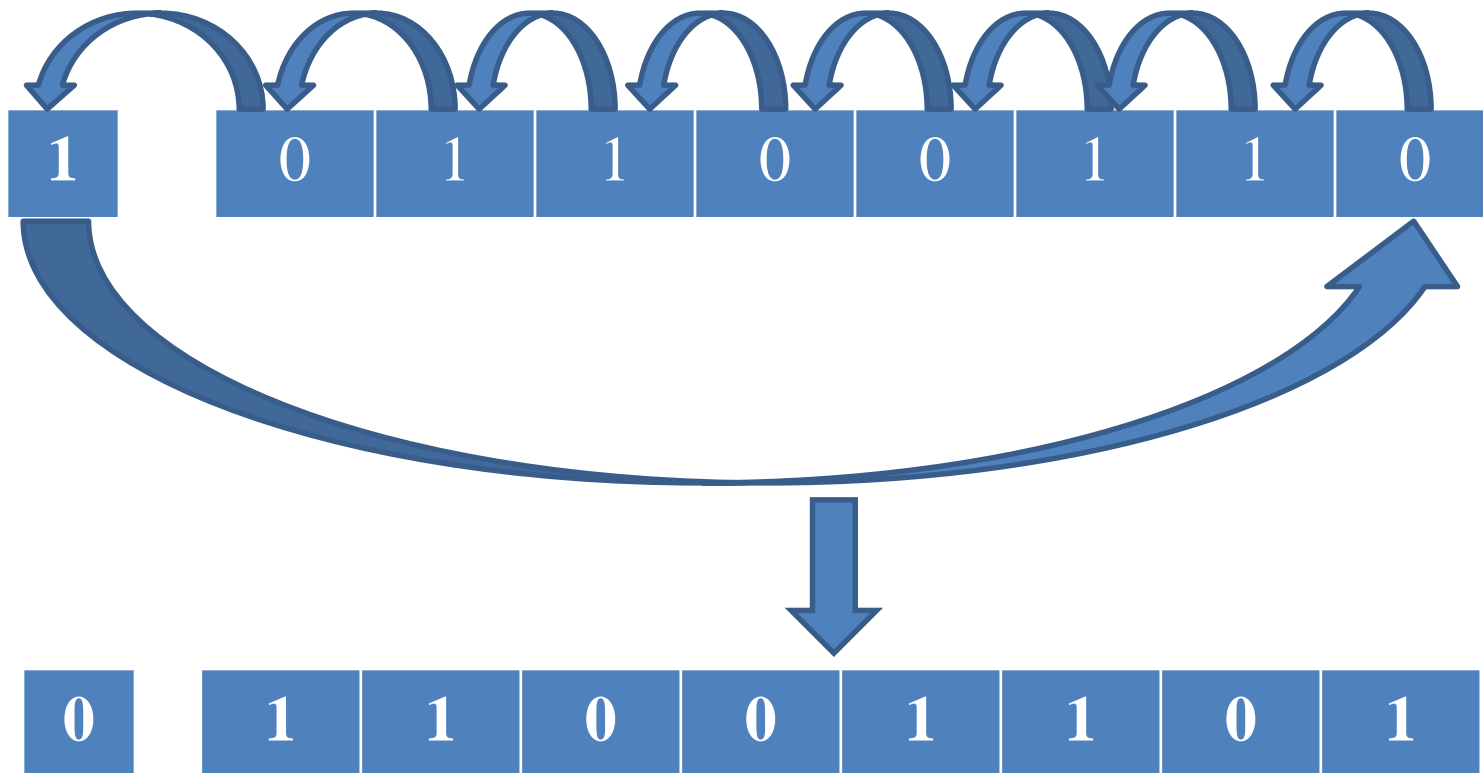


Logical Instructions

Opcode	Operand	Description
RAL	None	Rotate accumulator left through carry

- Each binary bit of the accumulator is rotated left by one position through the Carry flag.
- Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0.
- CY is modified according to bit D7.
- S, Z, P, AC are not affected.
- **Example:** RAL.

Logical Instructions

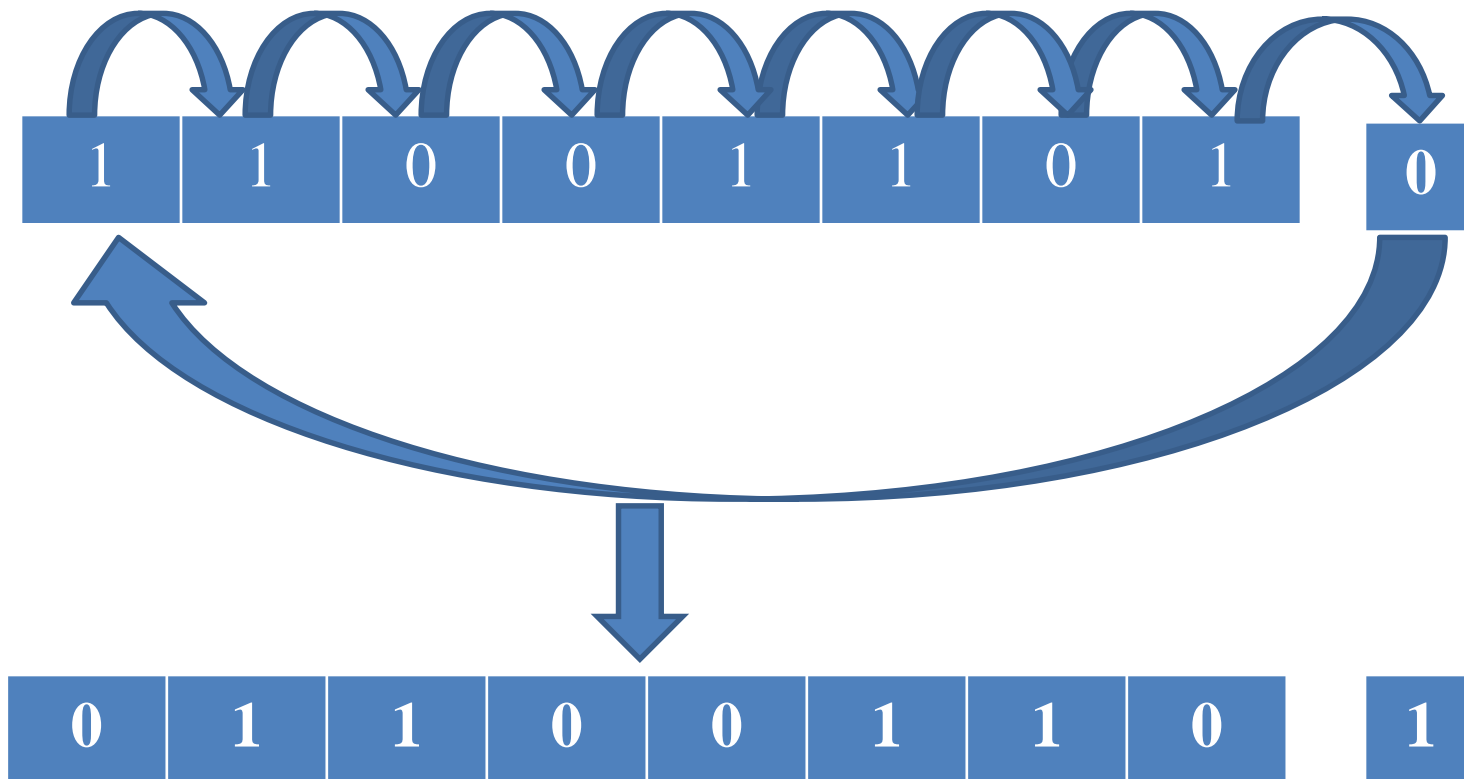


Logical Instructions

Opcode	Operand	Description
RAR	None	Rotate accumulator right through carry

- Each binary bit of the accumulator is rotated right by one position through the Carry flag.
- Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7.
- CY is modified according to bit D0.
- S, Z, P, AC are not affected.
- **Example:** RAR.

Logical Instructions



Logical Instructions

Opcode	Operand	Description
CMA	None	Complement accumulator

- The contents of the accumulator are complemented.
- No flags are affected.
- **Example: CMA**

$A = \overline{A}$

Logical Instructions

Opcode	Operand	Description
CMC	None	Complement carry

- The Carry flag is complemented.
- No other flags are affected.

Example: CMC

Logical Instructions

Opcode	Operand	Description
STC	None	Set carry

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

Video Link



<https://www.youtube.com/watch?v=ekcoleonyzg>

- The instruction that represents the ‘rotate source, count’ is
 - a) RCL
 - b) RCR
 - c) ROR
 - d) **All of the mentioned**
- What is the function of STC instruction?
 - a) Store to C Register, the value of Accumulator
 - b) **Set Carry to 1**
 - c) Clear the Stack pointer.
- After "XRA A" instruction is executed, what will be the status of Zero Flag?
 - a) **1**
 - b) 0
 - c) No change

- What is the content of A-Register at the end of this program?

XRA A

MVI B, F0H

SUB B

a) 01H b) 0FH c) F0H **d) 10H**

- What is the content of A at the end of this program?

STC

MVI A, 35H

ACI 26H

a) 2Dh **b) 5Ch** c) 23h d) 5B h

Recap

- Logical instructions of 8085 microprocessor with example is discussed.

Branch operation

Name of the Topic	Objective of the topic	Mapping with CO
Branch operation	To understand the branch & control instructions of 8085 microprocessor.	CO2

Prerequisite

- Basic concept of Digital Logic
- Knowledge of Boolean Algebra
- Understanding of 8085 microprocessor architecture.

Branching Instructions

- The branching instruction alter the normal sequential flow.
- These instructions alter either unconditionally or conditionally.

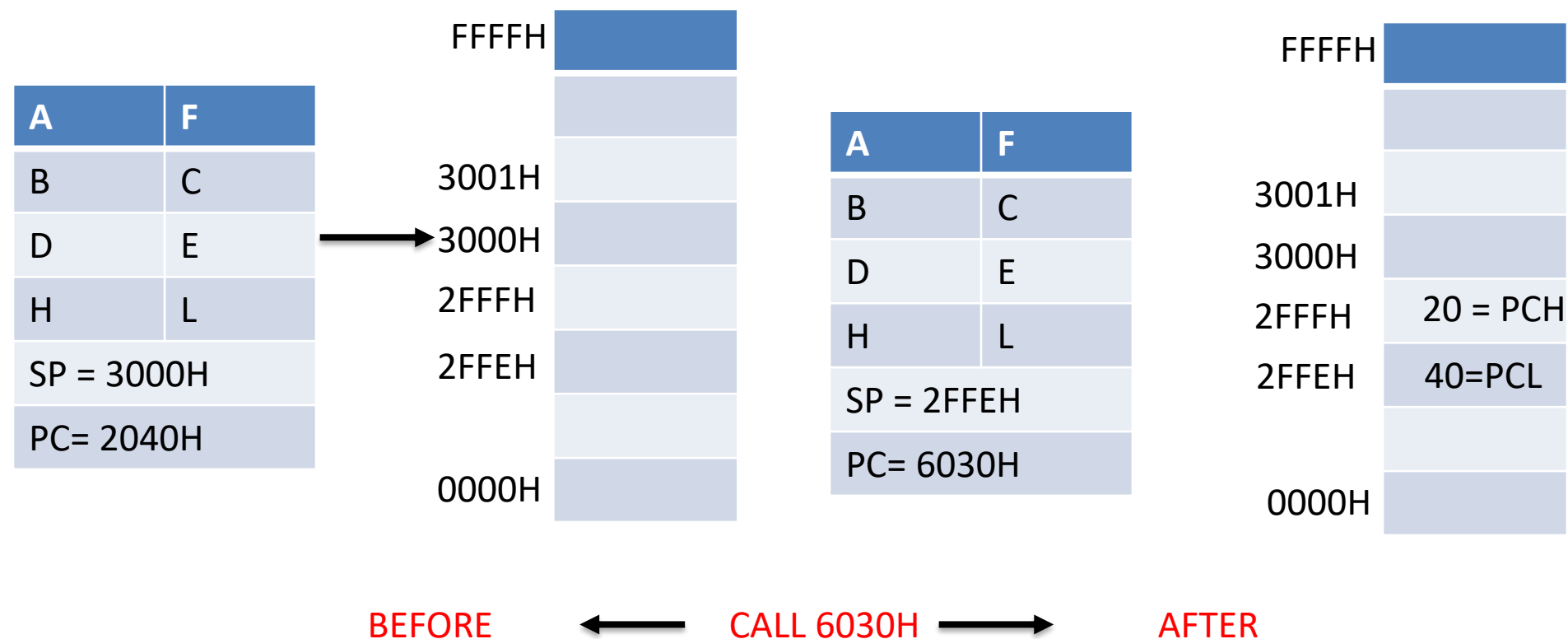
Branch operation

Opcode	Operand	Description
CALL	16-bit address	Call unconditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
- Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack.
- **Example:** CALL 2034 H.

Branch operation

Example : CALL 6030H



Branch operation

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 flag 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 2034 H.

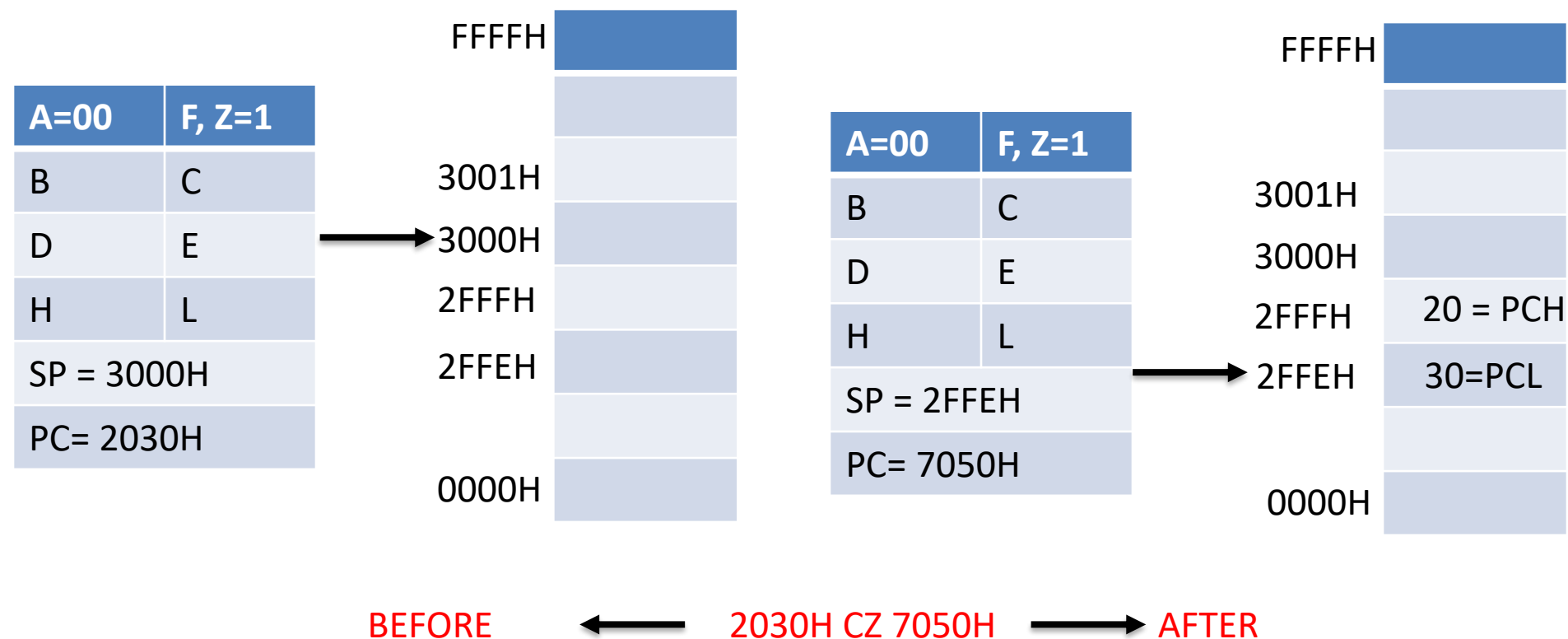
Branch operation

Call Conditionally

Opcode	Description	Status Flags
CC	Call on Carry	$CY = 1$
CNC	Call on Not Carry	$CY = 0$
CP	Call on Positive	$S = 0$
CM	Call on Minus	$S = 1$
CZ	Call on Zero	$Z = 1$
CNZ	Call on Not Zero	$Z = 0$
CPE	Call on Parity Even	$P = 1$
CPO	Call on Parity Odd	$P = 0$

Branch operation

Example : 2030H CZ 7050H



Branch operation

Opcode	Operand	Description
RET	None	Return unconditionally

- The program sequence is transferred from the subroutine to the calling program.
- The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.
- **Example: RET.**

Branch operation

Opcode	Operand	Description
Rx	None	Call conditionally

- The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW.
- The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.
- **Example: RZ.**

Branch operation

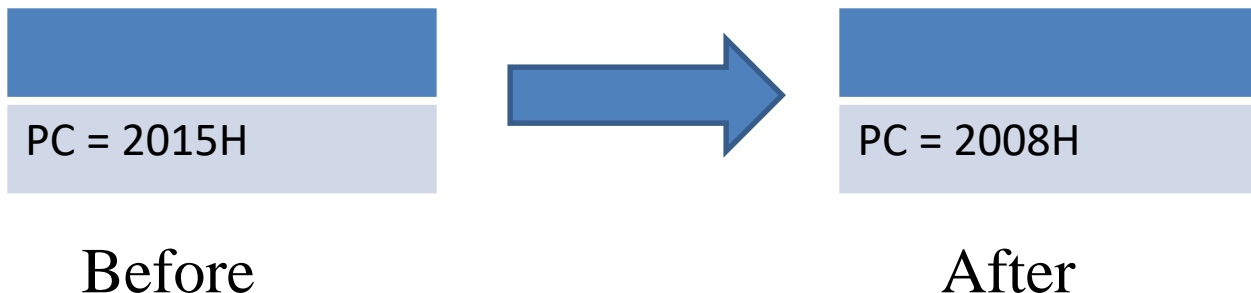
Return Conditionally

Opcode	Description	Status Flags
RC	Return if Carry	CY = 1
RNC	Return if Not Carry	CY = 0
RP	Return if Positive	S = 0
RM	Return if Minus	S = 1
RZ	Return if Zero	Z = 1
RNZ	Return if Not Zero	Z = 0
RPE	Return if Parity Even	P = 1
RPO	Return if Parity Odd	P = 0

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:** 2015H JMP 2008 H.



Branching Instructions

Opcode	Operand	Description
Jx	16-bit address	Jump conditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW.
- **Example:** JZ 2034 H.

Branch operation

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 = 1
JPO	Jump if Parity Odd	P = 0

Branch operation

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.

Branch operation

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 Instructions

- The control instructions control the operation of microprocessor.

Opcode	Operand	Description
NOP	None	No operation

- No operation is performed.
- The instruction is fetched and decoded but no operation is executed.
- **Example:** NOP

Control Operation

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 Operation

Opcode	Operand	Description
DI	None	Disable interrupt

- The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled.
- No flags are affected.
- **Example:** DI

Control Operation

Opcode	Operand	Description
EI	None	Enable interrupt

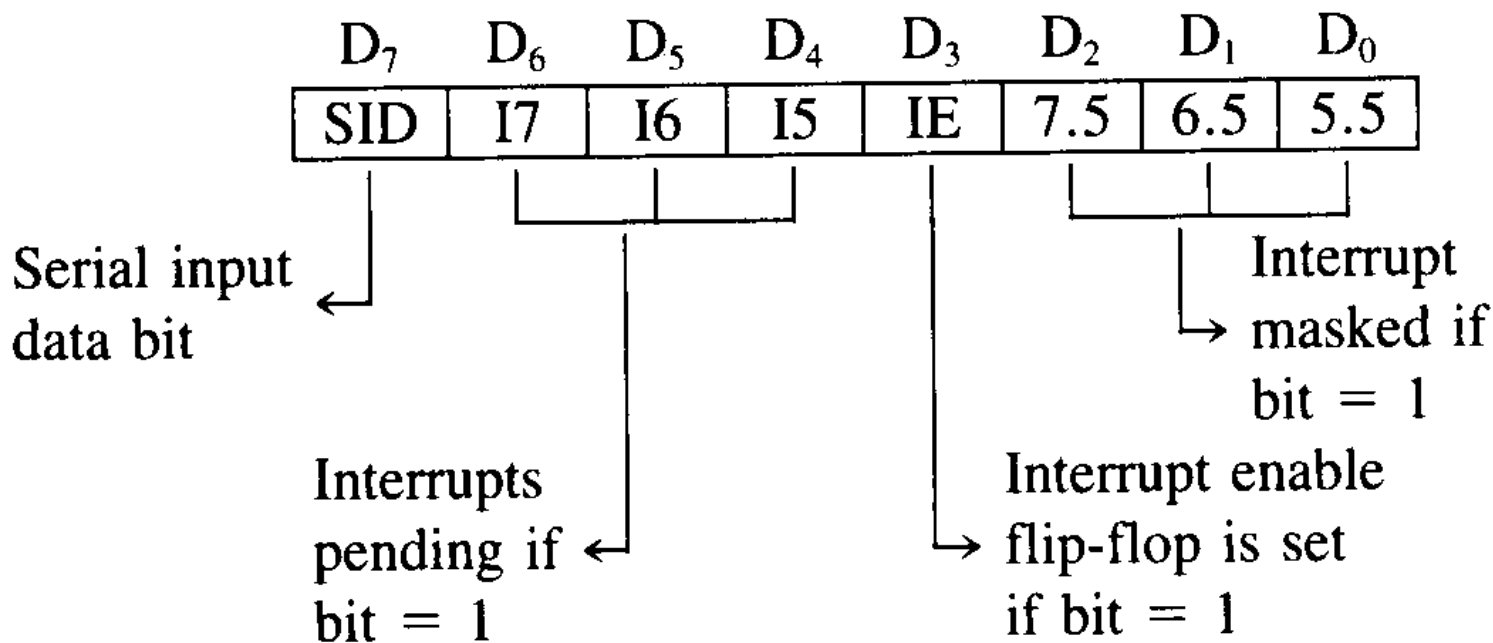
- The interrupt enable flip-flop is set and all interrupts are enabled.
- No flags are affected.
- This instruction is necessary to re-enable the interrupts (except TRAP).
- **Example: EI**

Control Operation

Opcode	Operand	Description
RIM	None	Read Interrupt Mask

- This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit.
- The instruction loads eight bits in the accumulator with the following interpretations.
- **Example: RIM**

RIM Instruction

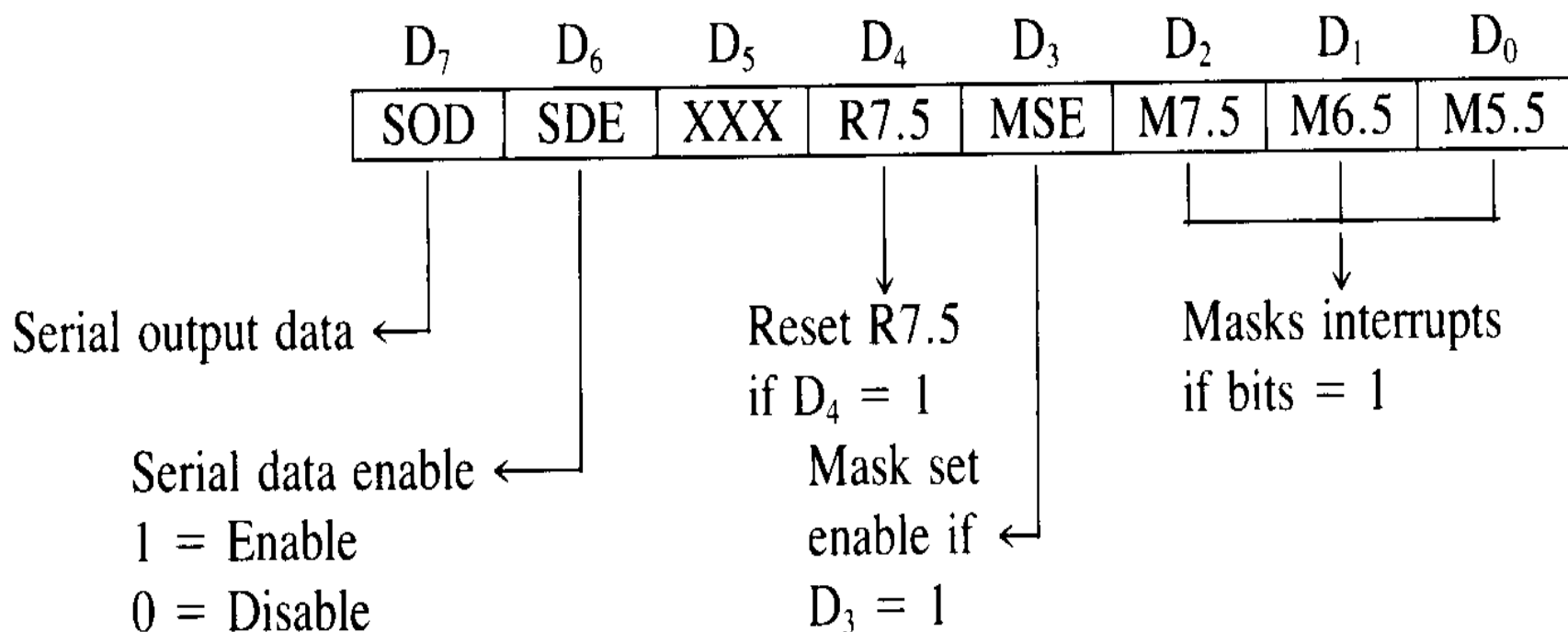


Control Operation

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





<https://www.youtube.com/watch?v=ekkoleonyzg>

- What does the last instruction of each subroutine that transfer the control to the instruction in the calling program with temporary address storage , called as?
 - A. jump to subroutine
 - B. branch to subroutine
 - C. return from subroutine**
 - D. call subroutine
- What is SIM?
 - A. Select interrupt mask
 - B. Sorting interrupt mask
 - C. Set interrupt mask**
 - D. None of these

- Which instruction is used to set the interrupt by maintaining the serial output bit in set mode of operation?
 - A. **SIM**
 - B. STC
 - C. SBI Data
 - D. SUI Data
- Which instruction indicates the transfer of program sequence to the address specified by 16 bit value if Z flag =0 ?
 - A. CZ Address
 - B. CNZ Address
 - C. CPE Address
 - D. CPO Address

Weekly Assignment

1. How many times NOP instruction will be executed in the following program;

MVI A, 10H

MVI B, 10H

BACK: NOP

ADD B

RLC

JNC BACK

HLT

Weekly Assignment

2. What is the output at 1236,1237 for the following program?

```
MVI C,00  
MVI D,80H  
MVI E,80H  
MOV A,D  
MOV B,A  
MOV A,E  
ADD B  
JNC LOOP  
INR C  
LOOP: STA 1236  
MOV A,C  
STA 1237  
HLT
```

Recap

- Logical & control instructions of 8085 microprocessor are discussed.

Assembler Directives

- Assembler Directives
- **Definition:** Assembler directives are the directions to the assembler which indicate how an operand or section of the program is to be processed. These are also called pseudo operations which are not executable by the microprocessor.
- For example, the assembler must be told at what address to start assembling the program. These assembler directives are command placed in the program by the designer that provides information to the processor.

Assembler Directives

The **assembler directives** are the instructions to the assembler concerning the program being assembled; they also are called *pseudo instructions* or *pseudo opcodes*. These instructions are neither translated into machine code nor assigned any memory locations in the object file. Some of the important assembler directives for an assembler or a cross-assembler are listed and described here.

Assembler

Directives	Example	Description
1. ORG (Origin)	ORG 2000H	The next block of instructions should be stored in memory locations starting at 2000H.
2. END	END	End of assembly. The HLT instruction suggests the end of a program, but that does not necessarily mean it is the end of the assembly.

Assembler Directives

- | | | | |
|----|---------------------------|---|---|
| 3. | EQU
(Equate) | PORT1 EQU 01H

INBUF EQU 2099H

STACK EQU INBUF+1 | <p>The value of the term PORT1 is equal to 01H. Generally, this means the PORT1 has the port address 01H.</p> <p>The value of the term INBUF is 2099H. This may be the memory location used as Input Buffer.</p> <p>The equate can be expressed by using the label of another equate. This example defines the stack as next location of INBUF.</p> |
| 4. | DB
(Define
Byte) | DATA: DB A2H,9FH | <p>Initializes an area byte by byte.</p> <p>Assembled bytes of data are stored in successive memory locations until all values are stored. This is a convenient way of writing a data string. The label is optional.</p> |
| 5. | DW
(Define
Word) | DW 2050H | <p>Initializes an area two bytes at a time.</p> |
| 6. | DS
(Define
Storage) | OUTBUF: DS 4 | <p>Reserves a specified number of memory locations. In this example, four memory locations are reserved for OUTBUF.</p> |



<https://www.youtube.com/watch?v=CPgFscob2T8>

- In an Intel 8085A, which is the first machine cycle of an instruction?
 - a. An op-code fetch cycle
 - b. A memory read cycle
 - c. A memory write cycle
 - d. An I/O read cycle
- What are level Triggering interrupts?
 - a) INTR&TRAP
 - b) RST6.5&RST5.5
 - c) RST7.5&RST6.5
- What is the RST for the TRAP?
 - a) RST5.5
 - b) RST4.5
 - c) RST4

- Which interrupt is not level sensitive in 8085?
 - a) RST6.5 is a raising edge-triggering interrupt.
 - b) RST7.5 is a raising edge-triggering interrupt.
 - c) a & b.
- RIM is used to check whether, _____
 - a) The write operation is done or not
 - b) The interrupt is Masked or not
 - c) a & b
- What is meant by Maskable interrupts?
 - a) An interrupt which can never be turned off.
 - b) An interrupt that can be turned off by the programmer.
 - c) none

Recap

- Problem related to T States & examples of assembly level languages are discussed.

6.5.1 Getting Started

Writing a program is equivalent to giving specific *commands* to the microprocessor in a *sequence to perform a task*. The italicized words provide clues to writing a program. Let us examine these terms.

- ☐ *Perform a Task*. What is the task you are asking it to do?
- ☐ *Sequence*. What is the sequence you want it to follow?
- ☐ *Commands*. What are the commands (instruction set) it can understand?

These terms can be translated into steps as follows:

Step 1: Read the problem carefully.

Step 2: Break it down into small steps.

Step 3: Represent these small steps in a possible sequence with a flowchart—a plan of attack.

Step 4: Translate each block of the flowchart into appropriate mnemonic instructions.

Step 5: Translate mnemonics into the machine code.

Step 6: Enter the machine code in memory and execute. Only on rare occasions is a program successfully executed on the first attempt.

Assembly language programming

Write a program to

1. turn on the five conveyer belts according to the ON/OFF positions of switches S_4 – S_0 at port F1H.
2. turn off the conveyer belts and turn on the emergency alarm only when both switches— S_7 from port F1H and S_7' from port F2H—are triggered.
3. monitor the switches continuously.

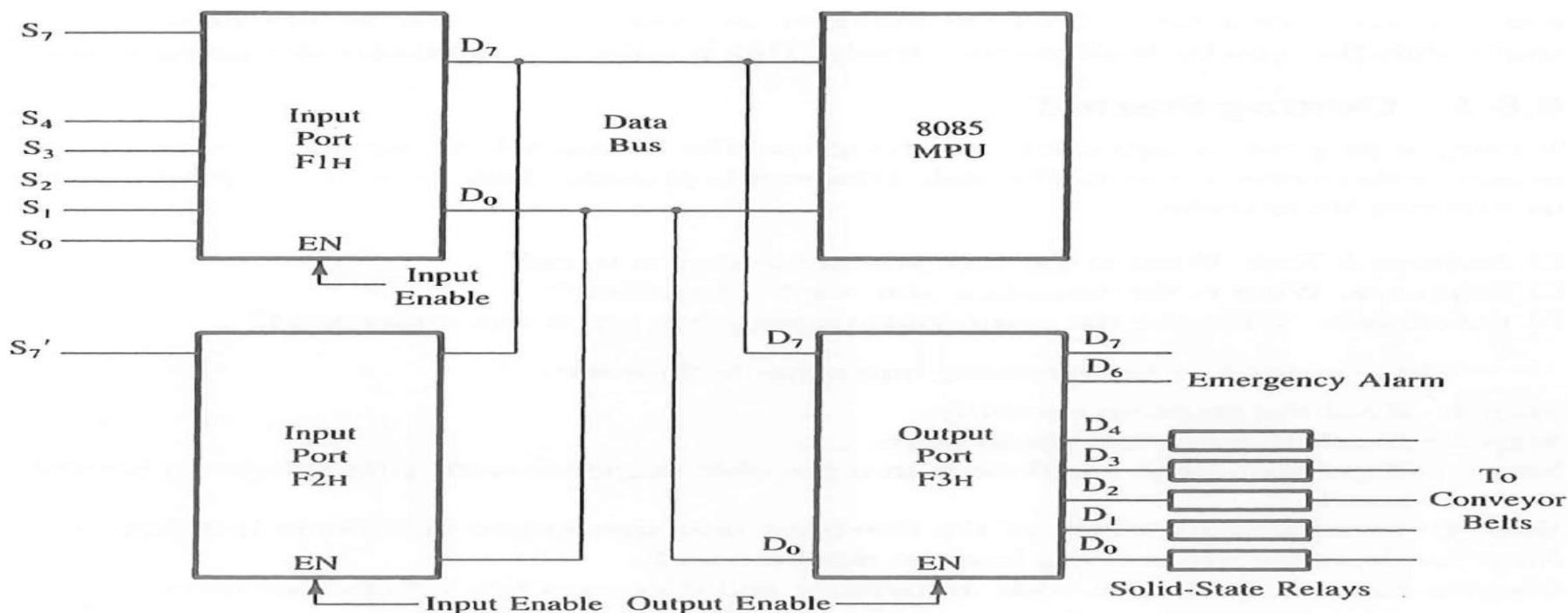


FIGURE 6.12
Input/Output Ports to Control Manufacturing Processes

Assembly language programming

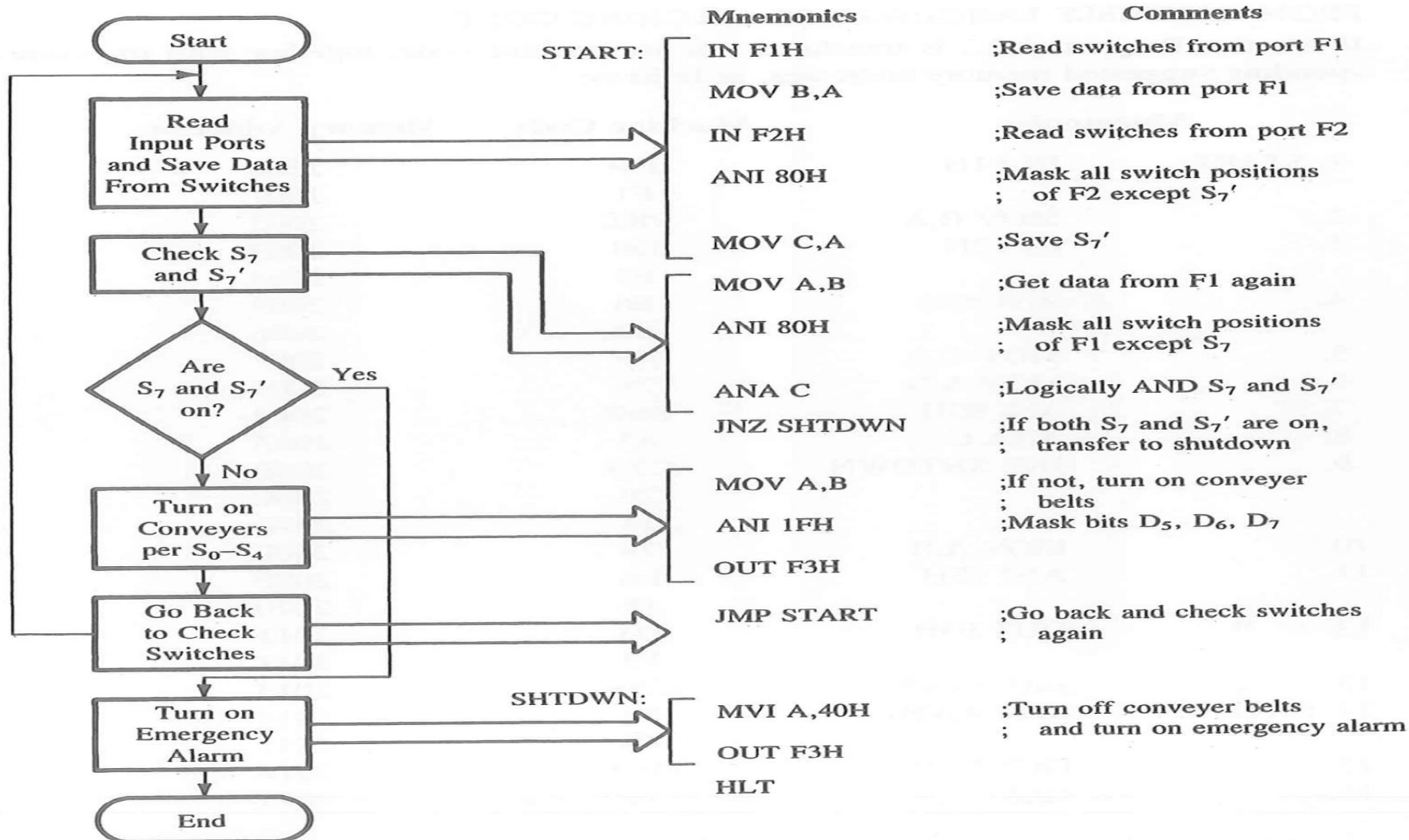
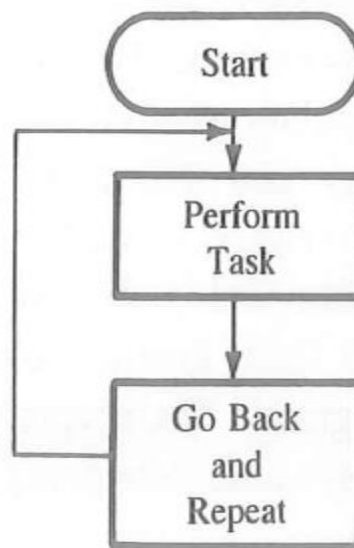


FIGURE 6.13
Flowchart and Program for Controlling Manufacturing Processes

Programming Techniques

- **LOOPING, COUNTING AND INDEXING LOOPING:** The programming technique used to instruct the microprocessor to repeat tasks is called looping. This task is accomplished by using jump instructions. Classification Of Loops: 1. Continuous loop 2. Conditional loop
Continuous Loop: Repeats a task continuously. A continuous loop is set up by using the unconditional jump instruction. A program with a continuous loop does not stop repeating the tasks until the system is reset.

FIGURE 7.1
Flowchart of a Continuous Loop



Programming Techniques

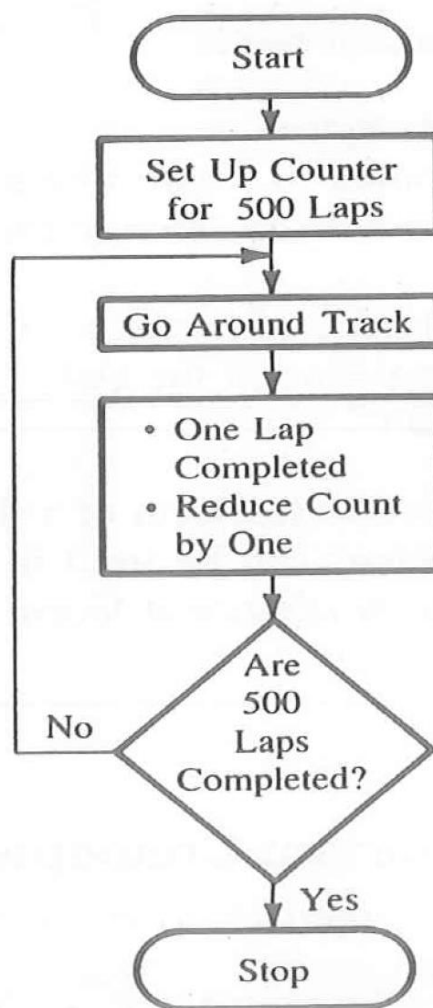
Conditional Loop:

A conditional loop is set up by a conditional jump instructions. These instructions

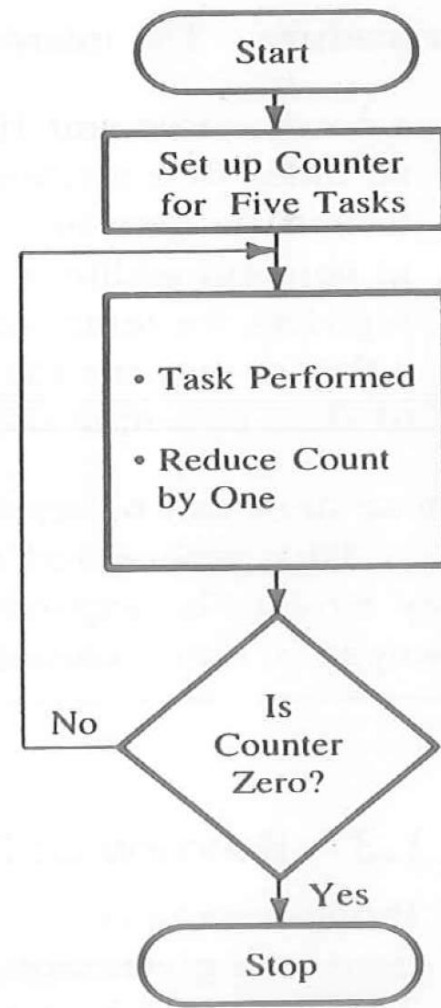
check flags(Z,CY,P,S) and repeat the tasks if the conditions are satisfied. These loops include counting and indexing.

Conditional Loop And Counter:

- A counter is a typical application of the conditional loop.
- A microprocessor needs a counter, flag to accomplish the looping task.
- Counter is set up by loading an appropriate count in a register.
- Counting is performed by either increment or decrement the counter.
- Loop is set up by a conditional jump instruction.
- End of counting is indicated by a flag.



(a) Laps in a Car Race



(b) Task Repetitions

Programming Techniques

Conditional Loop, Counter, and Indexing Another type of loop includes indexing along with a counter. (*Indexing* means pointing or referencing objects with sequential numbers. In a library, books are arranged according to numbers, and they are referred to or sorted by numbers. This is called indexing.) Similarly, data bytes are stored in memory locations, and those data bytes are referred to by their memory locations.

PROBLEM STATEMENT

Six bytes of data are stored in memory locations starting at XX50H. Add all the data bytes. Use register B to save any carries generated, while adding the data bytes. Display the entire sum at two output ports, or store the sum at two consecutive memory locations, XX70H and XX71H.

Data(H) A2, FA, DF, E5, 98, 8B

Programming Techniques

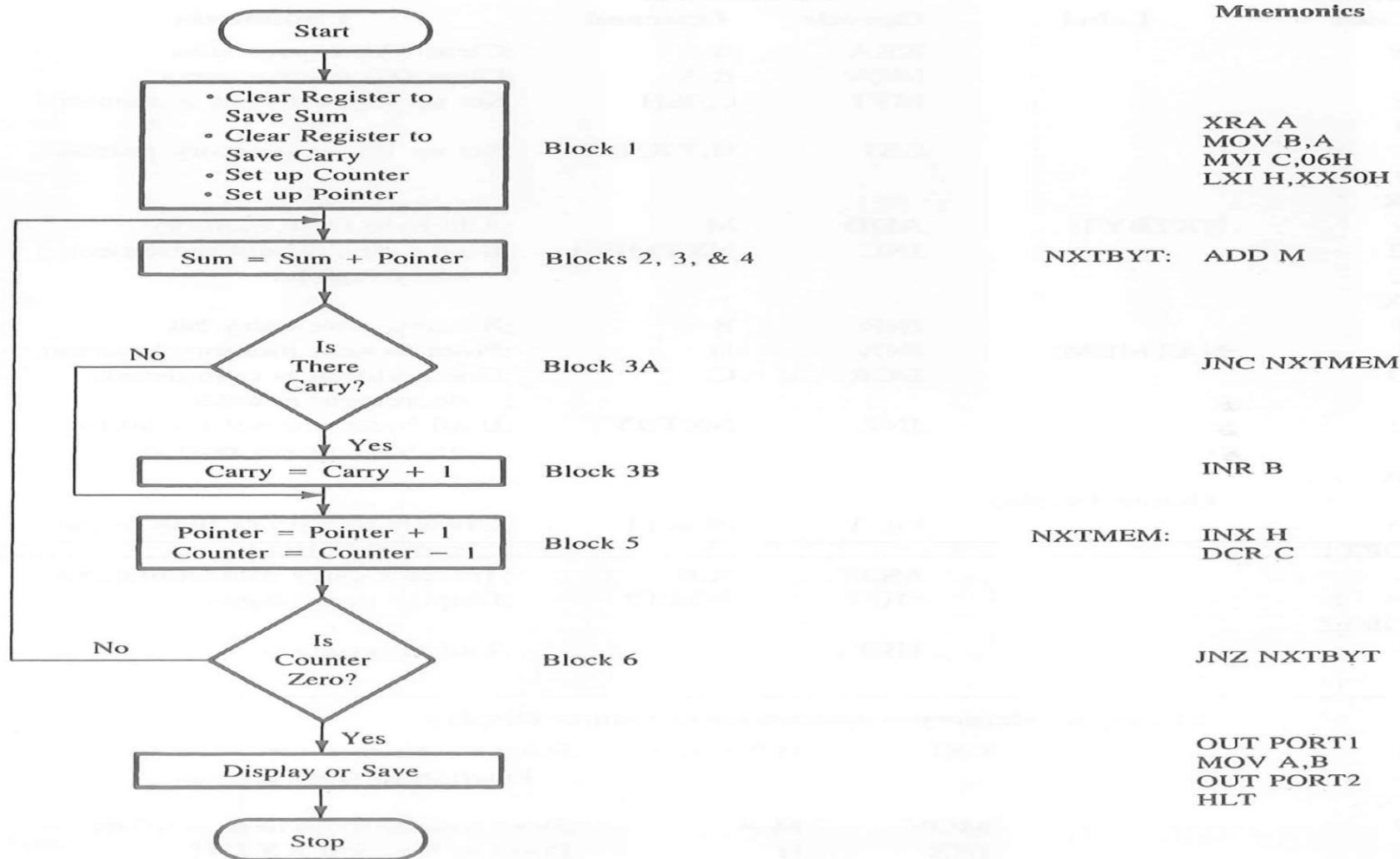


FIGURE 7.12
 Flowchart for Addition with Carry

Programming Techniques

PROBLEM STATEMENT

A set of current readings is stored in memory locations starting at XX50H. The end of the data string is indicated by the data byte 00H. Add the set of readings. The answer may be larger than FFH. Display the entire sum at PORT1 and PORT2 or store the answer in the memory locations XX70 and XX71H.

Data(H) 32, 52, F2, A5, 00



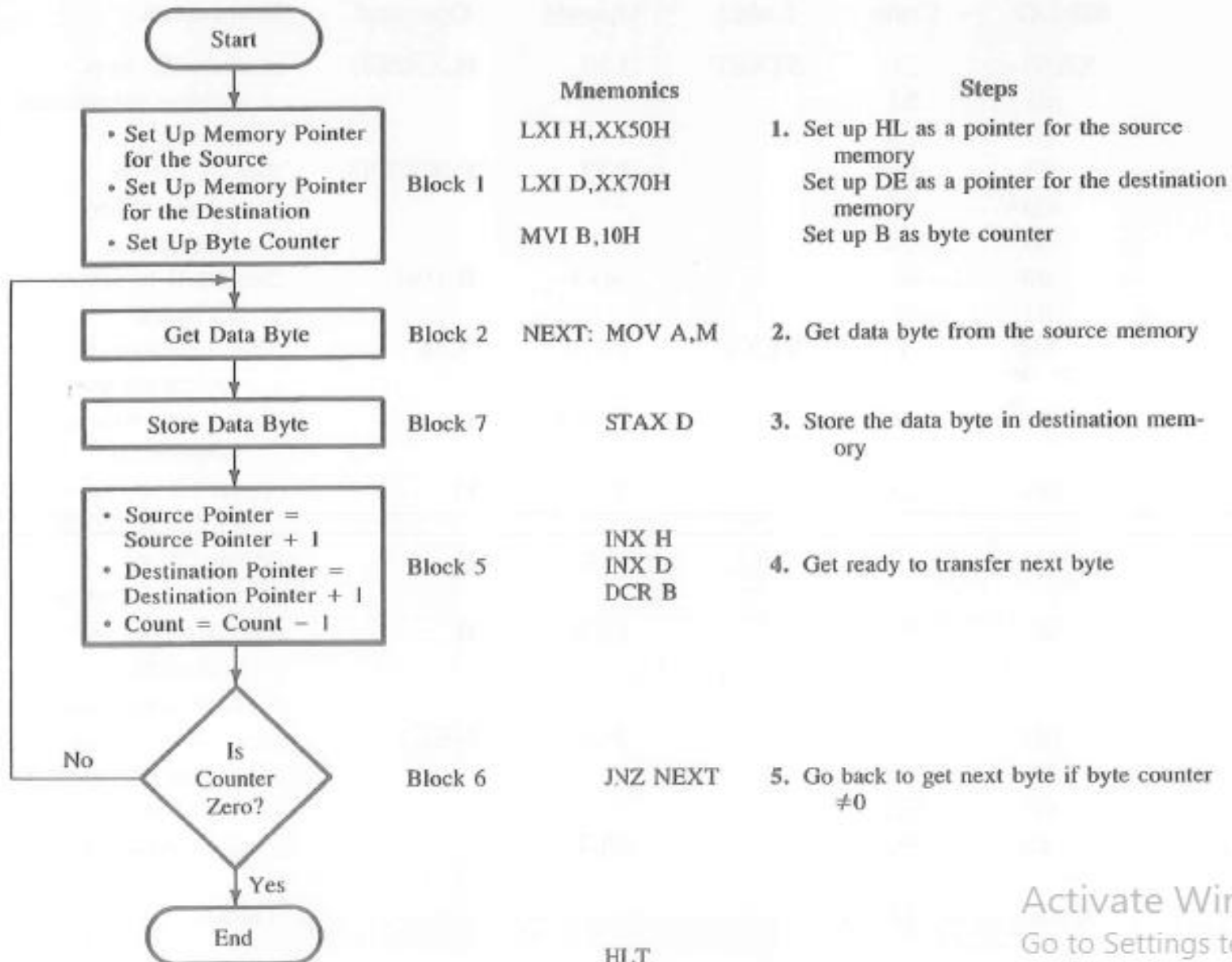
Programming Techniques

PROBLEM STATEMENT

Sixteen bytes of data are stored in memory locations at XX50H to XX5FH. Transfer the entire block of data to new memory locations starting at XX70H.

Data(H) 37, A2, F2, 82, 57, 5A, 7F, DA, E5, 8B, A7, C2, B8, 10, 19, 98

Programming Techniques



Daily quiz

- Which of the following is used as a primary storage device?
 - A. Magnetic drum
 - B. PROM**
 - C. Floppy disk
 - D. All of these
- Which of the following memories needs refresh?
 - A. SRAM
 - B. DRAM**
 - C. ROM
 - D. All of above

Weekly Assignment

1. How many times NOP instruction will be executed in the following program;

MVI A, 10H

MVI B, 10H

BACK: NOP

ADD B

RLC

JNC BACK

HLT

Weekly Assignment

2. What is the output at 1236,1237 for the following program?

```
MVI C,00  
MVI D,80H  
MVI E,80H  
MOV A,D  
MOV B,A  
MOV A,E  
ADD B  
JNC LOOP  
INR C  
LOOP: STA 1236  
MOV A,C  
STA 1237  
HLT
```


Youtube /other Video Links

<https://www.youtube.com/watch?v=iJmcgQRk048>

<https://www.youtube.com/watch?v=ekkleonxyzg>

<https://www.youtube.com/watch?v=79icCUmqyPc>

<https://www.youtube.com/watch?v=Bn0FhaaH6RA>

- Which of the following are the two main components of the CPU?
 - A. Control Unit and Registers
 - B. Registers and Main Memory
 - C. Control unit and ALU**
 - D. ALU and bus
- The language that the computer can understand and execute is called:
 - A. Machine language**
 - B. Application software
 - C. System program
 - D. All of the above

- Which of the following is used as a primary storage device?
 - A. Magnetic drum
 - B. PROM**
 - C. Floppy disk
 - D. All of these
- Which of the following memories needs refresh?
 - A. SRAM
 - B. DRAM**
 - C. ROM
 - D. All of above

- What is another name of memory stack especially given for the fundamental function performed by it?
 - A. Last-in-first-out (LIFO)**
 - B. First-in-last-out (FILO)
 - C. First-in-first-out (FIFO)
 - D. Last-in-last-out (LILO)
- Registers, which are partially visible to users and used to hold conditional, are known as
 - A. PC
 - B. Memory address registers
 - C. General purpose register**
 - D. Flags

Glossary questions

(Machine language, Control unit and ALU, DRAM, PROM)

Choose the correct one from above :

What are the two main components of the CPU?

The language that the computer can understand and execute is called:

Which is used as a primary storage device?

Which memories needs refresh?

Old question papers

1. Write an ALP to multiply two eight bit numbers.
2. Write a program for transferring the contents of Memory location 2050 H to the register B and the contents of 2051 H to Register C. The contents of memory location 2050 H are 10 of 2051 H are 11.
3. Find the total time to execute the above program if the frequency of the processor is 5MHz.
 1. MVI A,17H
 2. LOOP: RAL
 3. ORA A
 4. JNC LOOP

Expected Questions for University Exam

1. Explain the sequence of events during the execution of the CALL instruction by 8085 processor with the help of neat timing diagram.
2. Write an assembly language program with comment lines. An 8-bit number is stored in memory location 2A00H. Count number of ones (i.e. 1) in this byte and store this count in memory location 3200H.
3. Explain the following instructions
CALL, DAD B, XTHL, STAX B, CMP M
4. What is meant by interrupt? And explain different types of interrupt available in 8085.
5. Write a Program to perform the following functions and verify the output steps:
 - a) Load the number 5C22H in DE register pair
 - b) Move the number 9EH in register C

Recap of unit

This chapter discusses instruction set of 8085 Microprocessor.

Thank You