Microprocessor and Interfacing
(MPI)
GTU # 3160712

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Unit-4: Assembly Language Programming Basics

PART-I: 8085 Instruction Set

Positive Vibes:MPI is the interesting, easiest and scoring subject.

Prof. Swati R Sharma
Computer Engineering Department
Darshan Institute of Engineering & Technology, Rajkot

swati.sharma@darshan.ac.in
(O) 9727747317

Subject Overview

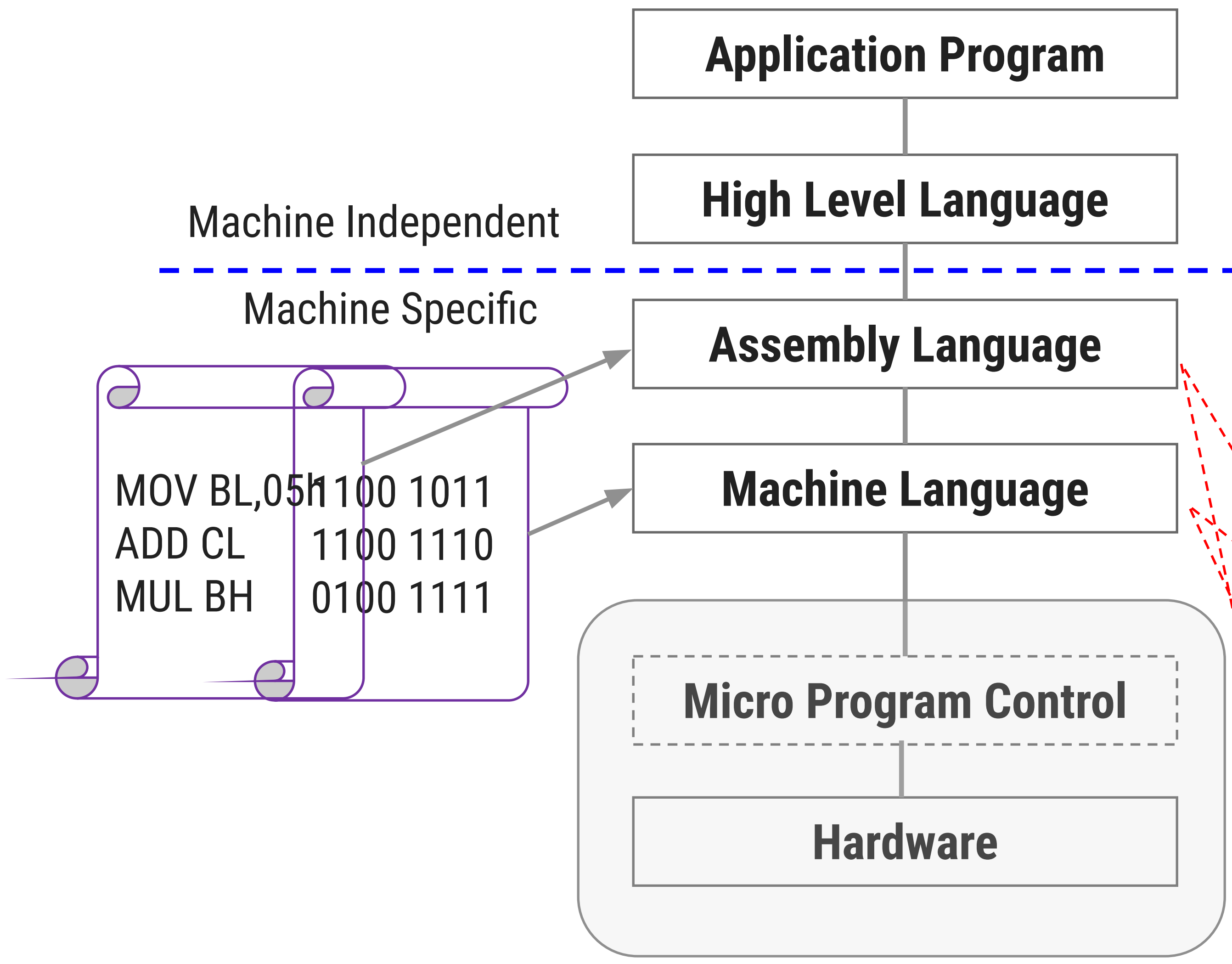
Sr. No.	Unit	% Weightage
1	Introduction to Microprocessor	8%
2	Microprocessor Architecture and Operations	7%
3	8085 Microprocessor	12%
4	**Assembly Language Programming Basics**	**13%**
5	8085 Assembly Language Programs	12%
6	Stack & Subroutines	13%
7	I/O Interfacing	20%
8	Advanced Microprocessors	15%

Topics to be covered

- Assembly Language Programming Basics
  - ✓ Hierarchy of Languages
  - ✓ Compilers and Assemblers
  - ✓ Instructions and Machine Language
  - ✓ Advantages of High-Level Languages
  - ✓ Why to Learn Assembly Language?
- Assembly Language Programming Tools
- Classification of 8085 Instructions
  - ✓ Based on Byte Size
  - ✓ Based on Function

Hierarchy of Languages

Hierarchy of Languages



High Level Language

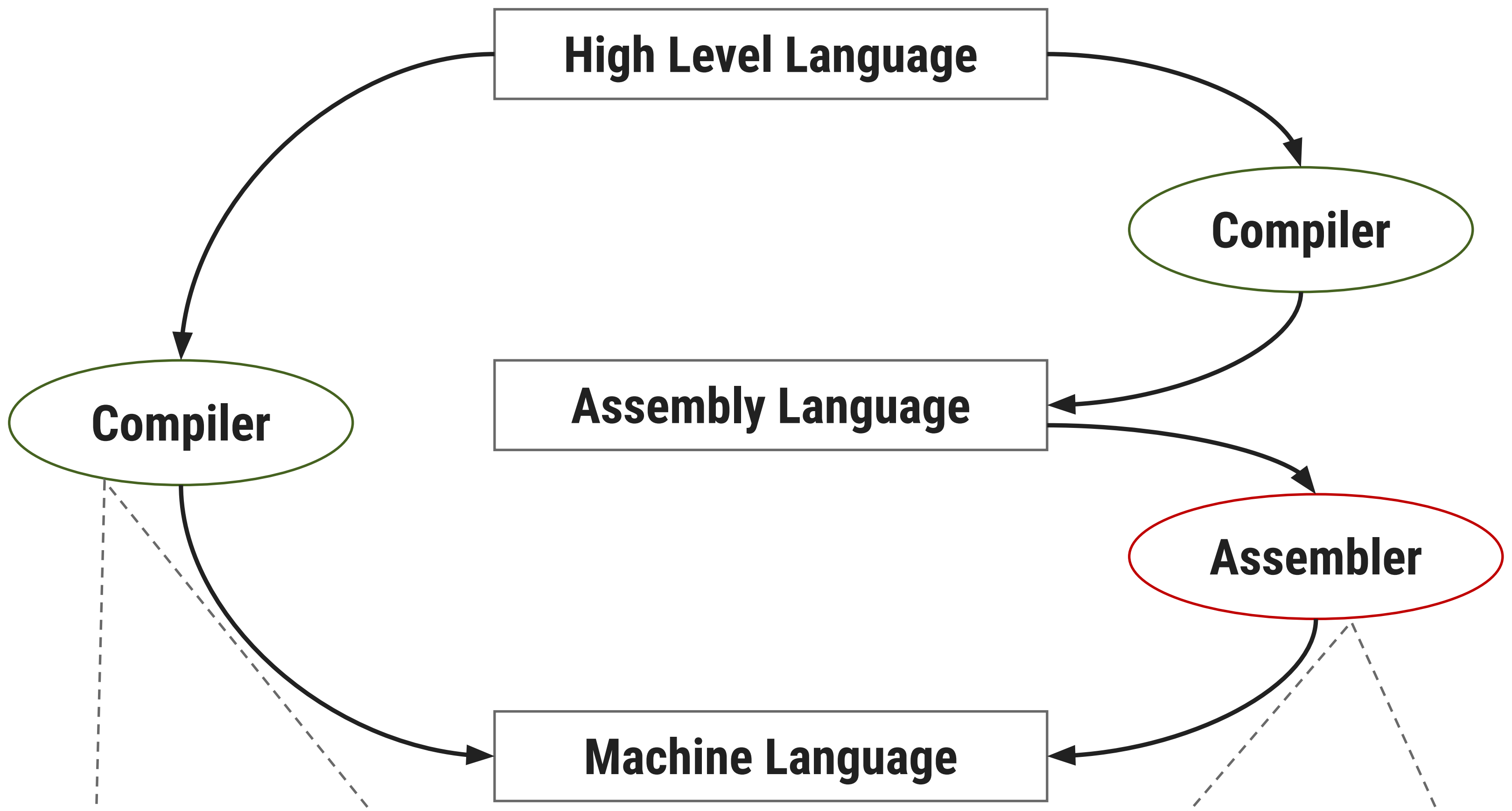
Low Level Language

- Assembly language is a low-level programming language designed for a specific type of processor.
- Uses symbolic names to represent operations, registers and memory locations.

- Native to a processor: executed directly by hardware
- Instructions consist of binary code: 1s and 0s

Compilers and Assemblers

Compilers and Assemblers



Compilers translate high-lev
either directly, or Indirectly via an assembler

Assemblers translates assembly code to machine code

Instructions and Machine Language

Instructions and Machine Language

- Each command of a program is called an instruction (it instructs the computer, *what to do?*).
- Computers only deal with binary data, hence the instructions must be in binary format (0's and 1's).
- Therefore, each Opcode is having unique bit pattern of (0's and 1's).

Instruction Fields

- Assembly language instructions usually are made up of several fields.
- Each field specifies different information.

The major two fields are:

1. **Opcode**: Operation code that specifies operation to be performed.
   Each operation has its unique **opcode**.
2. **Operands**: Fields which specify, where to get the source and destination operands for the operation specified by the **opcode**.

Instruction Fields

Opcode	Operand
MOV	Rd, Rs M, Rs R, M

$R_d \longrightarrow$ Destination Register
$R_s \longrightarrow$ Source Register
$M \longrightarrow$ Memory

Translating Languages

English: Sum of A and B

↓

High-Level Language: A + B

↓

A statement in a high-level language is translated typically into several machine-level instructions

Assembly Language:

```
MVI A,02
MVI B,03
ADD B
```

→

Machine Language:

```
1001 1011 0010
1001 1001 0011
1011 0011 0010
```

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Advantages of High-Level Languages

Advantages of High-Level Languages

- Program development is faster
  - High-level statements: fewer instructions to code.
- Program maintenance is easier
  - As Higher Level Language contains fewer instruction code.
- Programs are portable
  - Contain few machine-dependent details.
  - Can be used with little or no modifications on different machines.
  - Compiler translates to the target machine language.

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Why to Learn Assembly Language?

Why to Learn Assembly Language?

- Accessibility to system hardware
  - Assembly Language is useful for implementing system software.
  - It is also useful for small embedded system applications.
- Space and Time efficiency
  - Understanding sources of program efficiency.
  - Tuning program performance.
  - Writing compact code.
- It is helpful for
  - Compiler writing
  - Programming microcontrollers
  - Device drivers
  - System design
  - Low-level numeric routines

Why to Learn Assembly Language?

- Writing assembly programs gives the computer designer, deep understanding of the instruction set and how to design.
- To be able to write compilers for HLL (Higher Level Language), we need to be expert with the machine language. Assembly programming provides such experience.

Assembly Language Programming Tools

Assembly Language Programming Tools

1. Assembler
2. Linker
3. Debugger
4. Editor

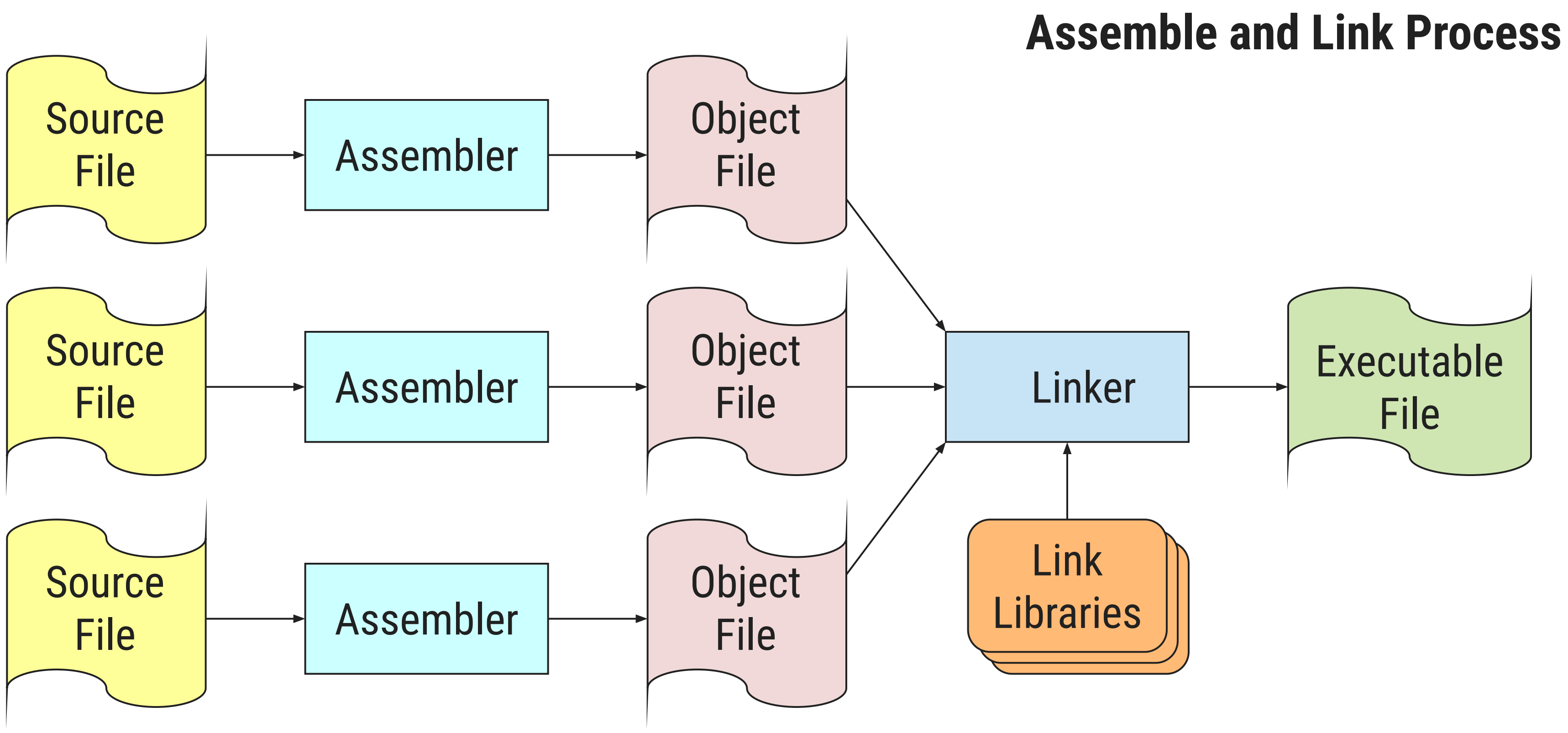
Assembler

- An assembler is a program that converts programs written in assembly language into object files(machine language).
- Popular assemblers have emerged over the years for the Intel family of processors. These include …
    - TASM (Turbo Assembler from Borland).
    - NASM (Net wide Assembler for both Windows and Linux), and
    - GNU assembler distributed by the free software foundation.

Linker

- A linker program is required to produce executable files.
- It combines program's object file created by the assembler with other object files and link libraries, to produces a single executable program.

Assembly Language Programming Tools



A project may consist of multiple source files

Assembler translates each source file separately into an object file

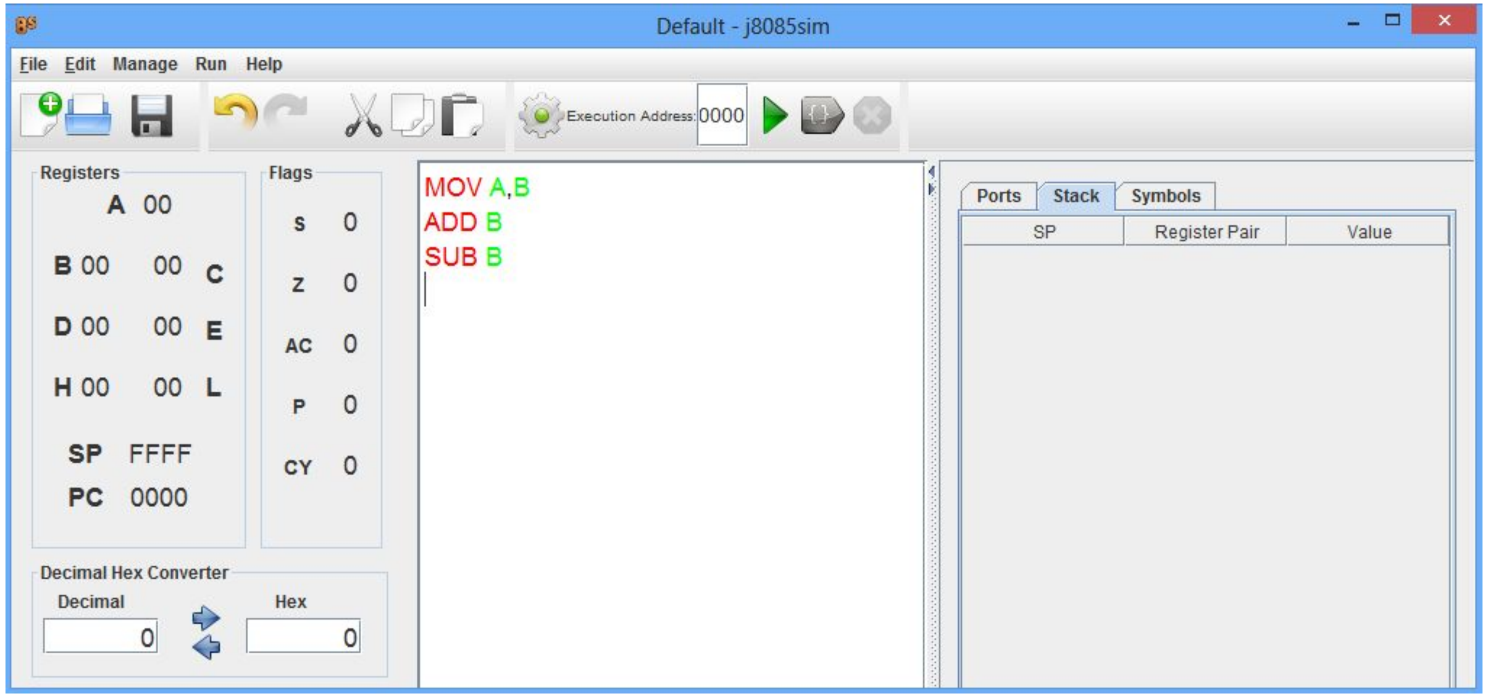
Linker links all object files together with link libraries

Debugger

- Allows you to trace the execution of a program.
- Allows you to view code, memory, registers etc.
- **Example**: WinDbg, GDB - the GNU debugger, etc.

Editor

- Allows to create assembly language source files.
- Some editors provide syntax highlighting features and can be customized as per programming environment.



```
MOV A,B
ADD B
SUB B
```

Registers			
A	00		
B	00	00	C
D	00	00	E
H	00	00	L
SP	FFFF		
PC	0000		

Flags	
S	0
Z	0
AC	0
P	0
CY	0

Classification of 8085 Instructions

Classification of 8085 Instructions

- **Based on Byte Size**
  - **One-byte Instructions**
    Requires one memory location
    to perform an operation
    E.g. CMA, ADD
  - **Two-byte Instructions**
    Requires two memory locations
    to perform an operation
    E.g. MVI A,32H
  - **Three-byte Instructions**
    Requires three memory locations
    to perform an operation
    E.g. JMP, CALL
- **Based on Function**
  - **Data Transfer Instructions**
  - **Arithmetic Instructions**
  - **Logic & Bit Manipulation Instructions**
  - **Branch Instructions**
  - **Control Instructions**

Classification of 8085 instructions

- An instruction is a **binary pattern** designed inside a microprocessor to perform a specific function.
- Each instruction is represented by an 8-bit binary value called **Op-Code**.
- The entire group of instructions that a microprocessor supports is known as an **Instruction Set**.

Classification of 8085 instructions

Instruction Set

- It is the set of instructions that the microprocessor can understand.

Opcode

- Known as Operation Code.
- This required field contains the mnemonic operation code for the 8085 instruction.

Operand

- The operand field identifies the data to be operated on by the specified opcode.
- Some instructions require no operands, while others require one or two operands.

```
MVI   D,  8BH
```

Opcode 🟥 🟩 **Operand**

Classification of 8085 instructions

General Terms

R	8085 8-bit register (A, B, C, D, E, H, L)
M	Memory
R_s	Register Source
R_d	Register Destination
R_p	Register Pair (BC, DE, HL)

One-byte Instruction

One-byte instructions includes **Opcode** and **Operand** in the same byte.

Instruction		Binary Code	Hexa Code
Opcode	Operand		
MOV	C,A	0100 1111	4FH
ADD	B	1000 0000	80H
CMA		0010 1111	2FH

List of one-byte Instructions

Sr.	Instruction	Sr.	Instruction	Sr.	Instruction	Sr.	Instruction
1	MOV dest.,src	13	SBB R/M	25	RNZ	37	CMA
2	LDAX R_P (B/D)	14	INR R/M	26	RPE	38	CMC
3	STAX R_P	15	INX R_P	27	RPO	39	STC
4	XCHG	16	DCR R/M	28	RST 0-7	40	NOP
5	SPHL	17	DCX R_P	29	CMP R/M	41	HLT
6	XTHL	18	DAA	30	ANA R/M	42	DI
7	PUSH R_P	19	RET	31	XRA R/M	43	EI
8	POP R_P	20	RC	32	ORA R/M	44	RIM
9	ADD R/M	21	RNC	33	RLC	45	SIM
10	ADC R/M	22	RP	34	RRC		
11	DAD	23	RM	35	RAL		
12	SUB R/M	24	RZ	36	RAR		

Two-byte Instruction

In two-byte instruction,

1st Byte : Specifies Opcode

2nd Byte: Specifies Operand

Instruction		Binary Code	Hexa Code
Opcode	**Operand**		
MVI	A,32H	0011 1110 0011 0010	3E: 1st Byte 32: 2nd Byte
MVI	B,F2H	0011 1110 1111 0010	3E: 1st Byte F2: 2nd Byte
IN	0AH	1101 1011 0000 1010	DB: 1st Byte 0A: 2nd Byte

List of two-byte Instructions

Sr.	Instruction
1	MVI destination,8-bit data
2	OUT 8-bit port address
3	IN 8-bit port address
4	ADI 8-bit data
5	ACI 8-bit data
6	SUI 8-bit data
7	SBI 8-bit data
8	CPI 8-bit data
9	ANI 8-bit data
10	XRI 8-bit data
11	ORI 8-bit data

Three-byte Instruction

In three-byte instruction,

1st Byte: Specifies Opcode

2nd Byte: Specifies lower order 8-bit address

3rd Byte: Specifies higher order 8-bit address

Instruction		Binary Code	Hexa Code
Opcode	**Operand**		
LDA	2050H	0011 1010 0101 0000 0010 0000	3A: 1st Byte 50: 2nd Byte 20: 3rd Byte
JMP	2085H	1100 0011 1000 0101 0010 0000	C3: 1st Byte 85: 2nd Byte 20: 3rd Byte

List of three-byte Instructions

Sr.	Instruction	Sr.	Instruction
1	LDA 16-bit address	13	JPE 16-bit address
2	LXI R_p, 16-bit data	14	JPO 16-bit address
3	LHLD 16-bit address	15	CALL 16-bit address
4	STA 16-bit address	16	CC 16-bit address
5	SHLD 16-bit address	17	CNC 16-bit address
6	JMP 16-bit address	18	CP 16-bit address
7	JC 16-bit address	19	CM 16-bit address
8	JNC 16-bit address	20	CZ 16-bit address
9	JP 16-bit address	21	CNZ 16-bit address
10	JM 16-bit address	22	CPE 16-bit address
11	JZ 16-bit address	23	CPO 16-bit address
12	JNZ 16-bit address		

GTU Exam Questions

Sr	Questions	Marks	Year
1.	Explain the functions of following instructions of 8085 – state the bytes occupied, number of Machine cycle required and T-States 1. LXI H, 2050H 2. MOV B,A 3. STA 5050H 4. ADD C	4	W'18
2.	Explain the functions of following instructions of 8085 – state its number of bytes occupied, number of Machine cycle required and T-states. 1. MOV A,M 2. LXI H,1000H 3. DAA	4	S'19
3.	Explain One byte, Two byte, Three byte instruction.	4	W'19

Classification of 8085 Instructions

- **Based on Byte Size**
  - **One-byte Instructions**
    Requires one memory location
    to perform an operation
    E.g. CMA, ADD
  - **Two-byte Instructions**
    Requires two memory locations
    to perform an operation
    E.g. MVI A,32H
  - **Three-byte Instructions**
    Requires three memory locations
    to perform an operation
    E.g. JMP, CALL
- **Based on Function**
  - **Data Transfer Instructions**
  - **Arithmetic Instructions**
  - **Logic & Bit Manipulation Instructions**
  - **Branch Instructions**
  - **Control Instructions**

Data Transfer Instructions

Data Transfer Instructions

- Instructions copy data from source to destination.
- While copying, the contents of source is not modified.
- Data Transfer Instructions do not affect the flags.

MOV: Move data from source to destination

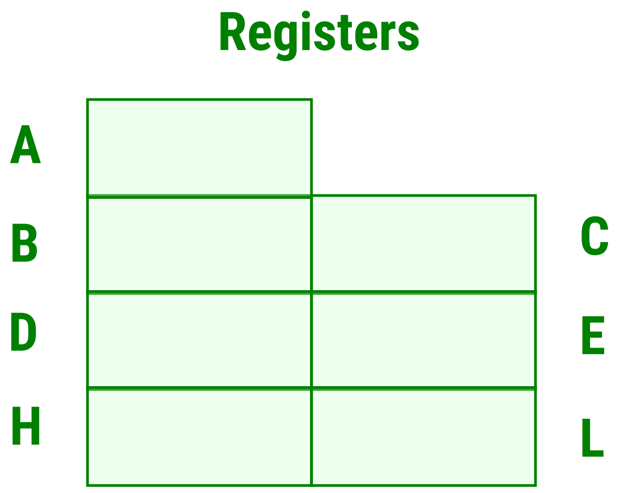
Instruction		Description	Example
Opcode	Operand		
MOV MOV MOV	R_d, R_s M, R R, M	• This instruction copies the contents of the source register into the destination register. • Contents of the source register is not altered. • If one of the operands is a memory, its location is specified by the contents of the HL registers. • one-byte instruction.	MOV B, C ; *B←C* MOV B, M;*B←M[HL]* MOV M, B; *M[HL]←B*

MVI: Load 8-bit to Register/Memory

Instruction		Description	Example
Opcode	Operand		
MVI	R, Data M, Data	• 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 HL registers. • Two-byte instruction.	MVI B, 57H; *B←12* MVI M, 12H ; *M[HL]←12*

Example: LDA Instruction

LDA 2050H



Memory

02	2000
04	…
0A	..
06	.
0F	2049
0D	2050
05	2051
03	2052

LDA: Load Accumulator

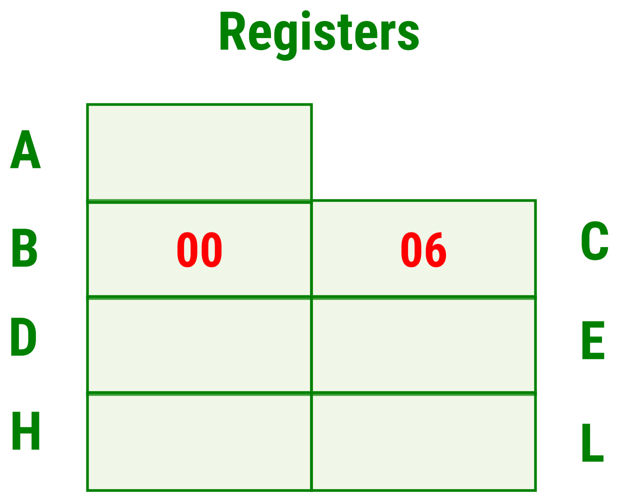
Instruction		Description	Example
Opcode	Operand		
LDA	16-bit address	• The contents of a memory location, specified by a 16-bit address in the operand, is copied to the accumulator. • The contents of the source is not altered. • Three-byte instruction.	LDA 2050H LDA 0006H

LDAX: Load the accumulator indirect

Instruction		Description	Example
Opcode	Operand		
LDAX	R_p (B/D)	• The contents of a memory location, specified by a 16-bit address in the operand, is copied to the accumulator. • The contents of the source is not altered.	**LDAX B** ; *A←M[BC]* **LDAX D** ; *A←M[DE]*

Example: LDAX Instruction

LDAX B ; *A⟵M[BC]*

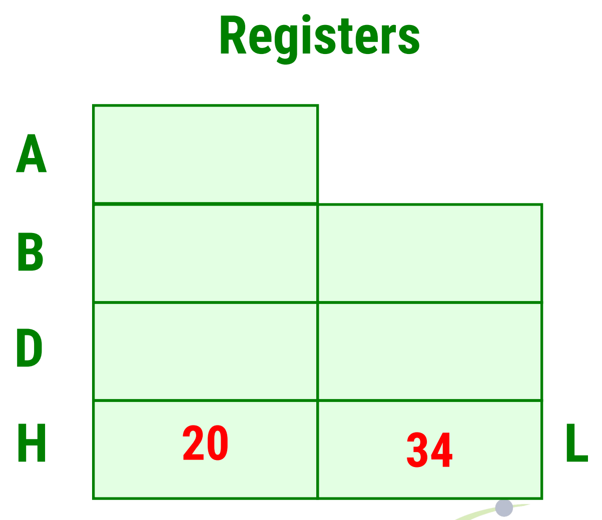


Memory

02	0001
04	0002
0A	0003
06	0004
0F	0005
0D	0006
05	0007
03	0008

LXI: Load the immediate register pair

Instruction		Description	Example
Opcode	Operand		
LXI	$R_{p,}$ 16-bit Data	The instruction loads immediate 16-bit data into register pair.	LXI H, 2034H

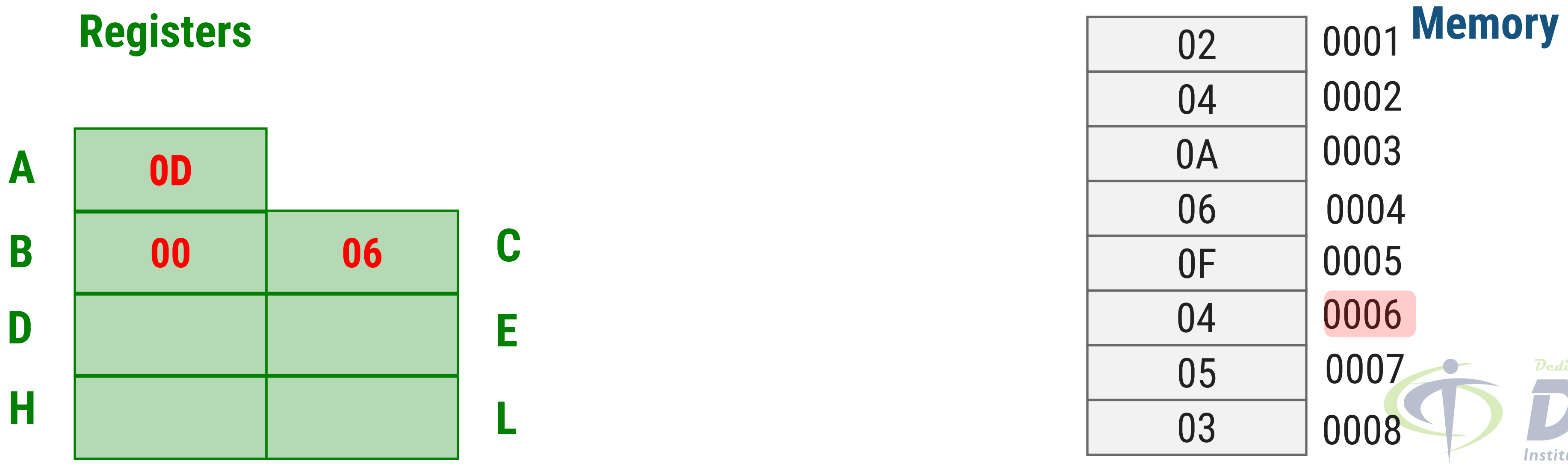


STA: Store Accumulator

Instruction		Description	Example
Opcode	Operand		
STA	16-bit address	The contents of accumulator is copied into the memory location specified by the operand.	STA 0002H 16-bit memory address

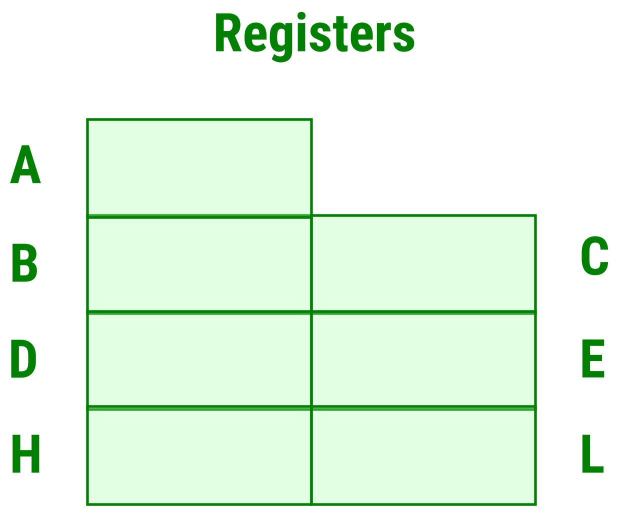
STAX: Store Accumulator Indirect

Instruction		Description	Example
Opcode	Operand		
STAX	R_p	The contents of accumulator is copied into memory location specified by the contents of the operand (register pair). The contents of the accumulator is not altered.	STAX B;*M[BC]⟵A*



Example: LHLD Instruction

LHLD 0006H



Memory

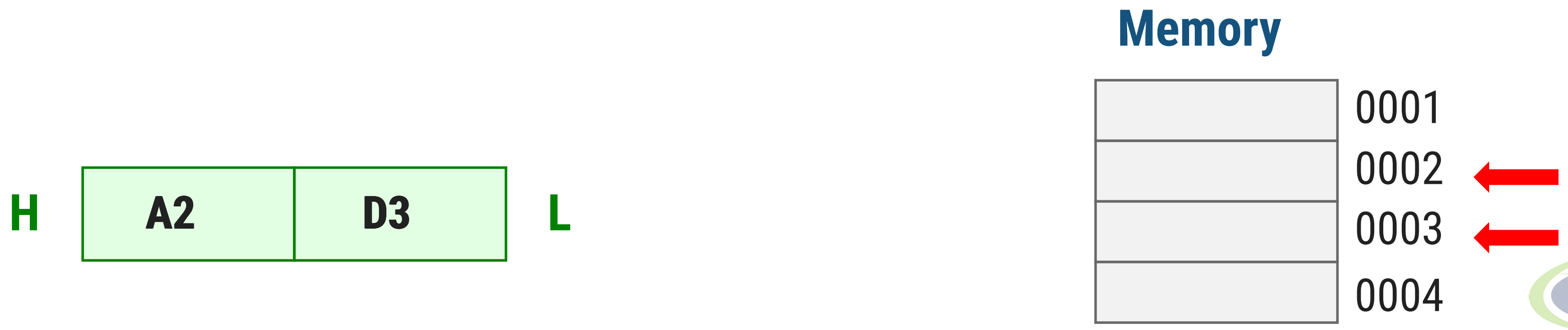
02	0001
04	0002
0A	0003
06	0004
0F	0005
0D	0006
05	0007
03	0008

LHLD: Load H and L registers direct

Instruction		Description	Example
Opcode	Operand		
LHLD	16-bit address	• The instruction copies contents of the memory location pointed out by the address into register L and copies the contents of the next memory location into register H. • The contents of source memory locations is not altered.	LHLD 2050H LHLD 0006H

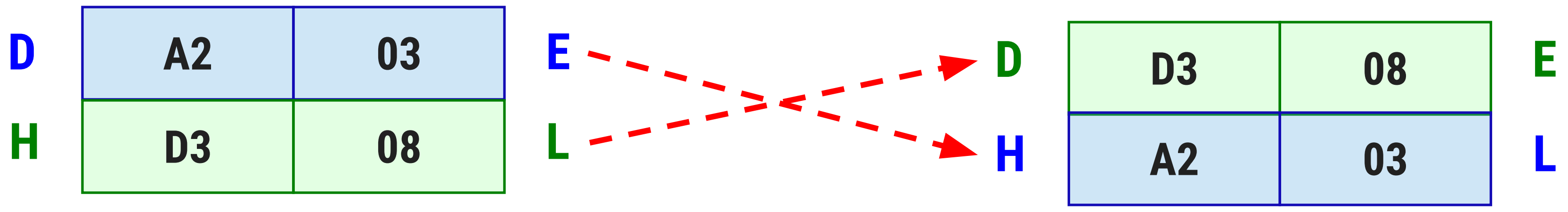
SHLD: Store H and L registers direct

Instruction		Description	Example
Opcode	Operand		
SHLD	16-bit address	The contents of register L is stored in memory location specified by the 16-bit address in the operand and the contents of H register is stored into the next memory location by incrementing the operand.	SHLD 0002H



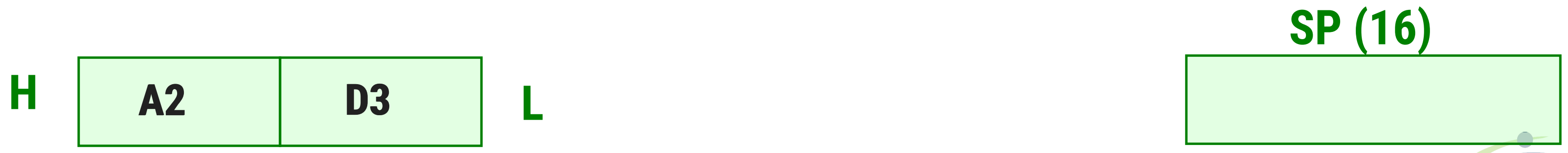
XCHG: Exchange H and L with D and E

Instruction		Description	Example
Opcode	Operand		
XCHG	None	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E.	XCHG



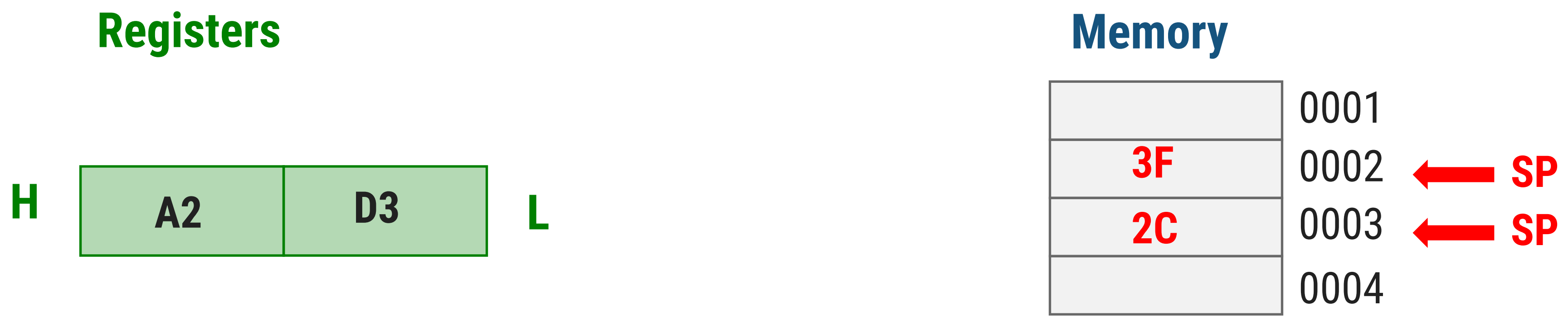
SPHL: Copy H and L registers to stack pointer

Instruction		Description	Example
Opcode	Operand		
SPHL	None	The instruction loads the contents of the H and L registers into the stack pointer register, the contents of H register provide the high-order address and the contents of L register provide the low-order address. The contents of the H and L registers are not altered.	SPHL



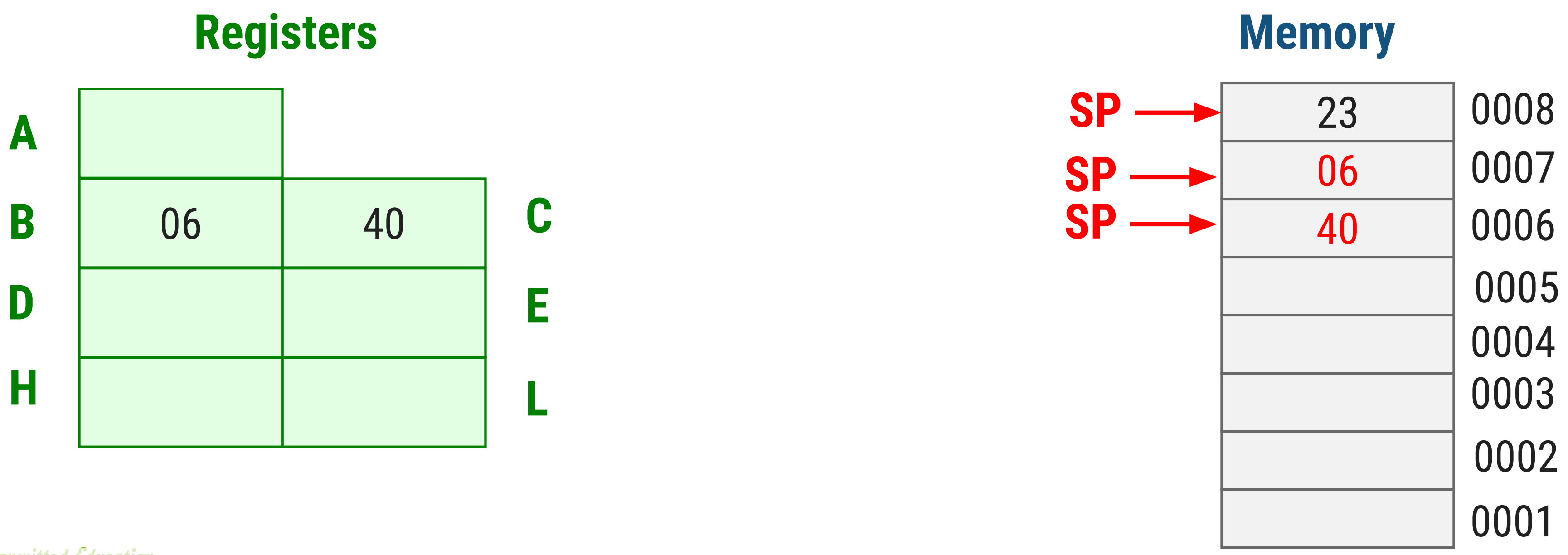
XTHL: Exchange H and L with top of stack

Instruction		Description	Example
Opcode	Operand		
XTHL	None	The contents of L register is exchanged with stack location pointed out by contents SP. The contents of the H register are exchanged with the next stack location (SP+1).	XTHL



Example: PUSH Instruction

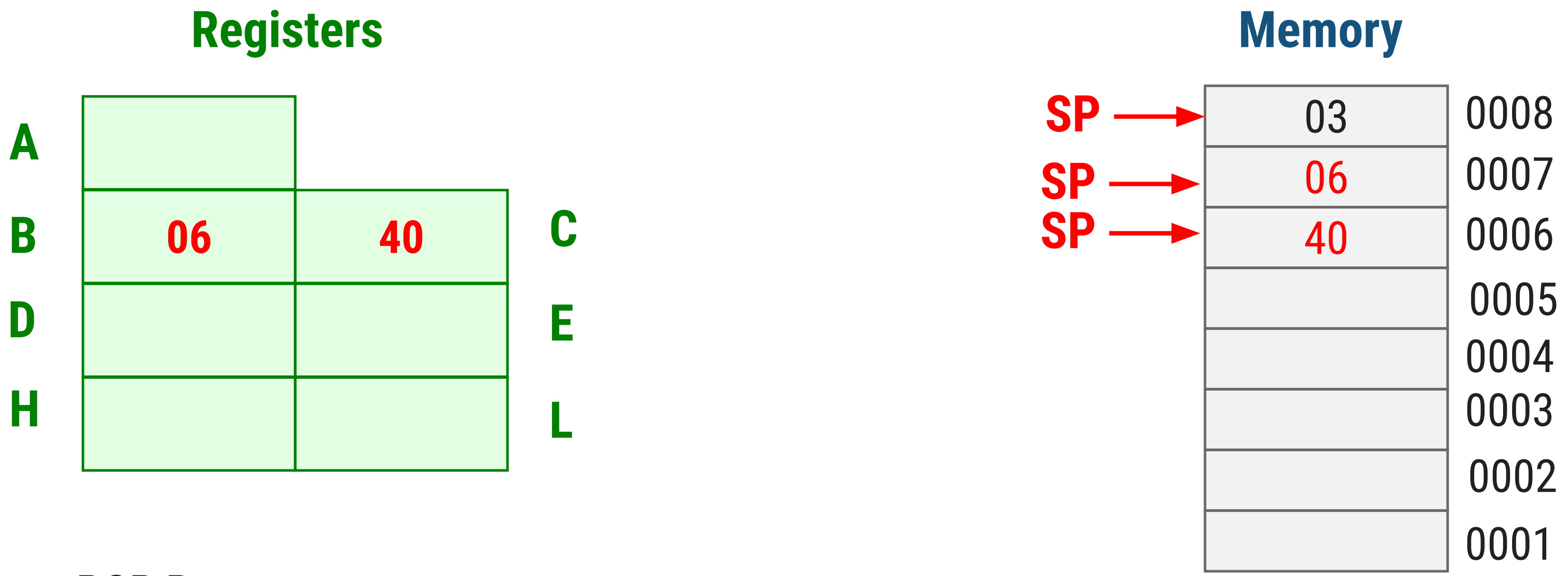
PUSH B
SP <- SP-1
SP <- B *;transfer high order bit to TOS*
SP <- SP-1
SP <- C *;transfer low order bit to TOS*



PUSH: Push the register pair onto the stack

Instruction		Description	Example
Opcode	Operand		
PUSH	R_p	The contents of the register pair designated in the operand are copied onto the stack in the following sequence. 1. The SP register is decremented and the contents of the high order register (B, D, H) are copied into that location. 2. The SP register is decremented again and the contents of the low-order register (C, E, L) are copied to that location.	**PUSH B**

Example: POP Instruction



POP B

C <- SP *; transfer to low order bit from TOS*
SP <- SP+1

B <- SP *; transfer to high order bit from TOS*
SP <- SP+1

POP: Pop off stack to the register pair

Instruction		Description	Example
Opcode	Operand		
POP	R_p	The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L) of the operand. 1. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H) of the operand. 2. The stack pointer register is again incremented by 1.	**POP B**

OUT: Output from Accumulator to 8-bit port

Instruction		Description	Example
Opcode	Operand		
OUT	8-bit port address	The contents of the accumulator are copied into the I/O port specified by the operand.	**OUT 0AH**

IN: Input data to accumulator from with 8-bit port

Instruction		Description	Example
Opcode	Operand		
IN	8-bit port address	The contents of the input port designated in the operand are read and loaded into the accumulator.	**IN 0AH**

Data Transfer Instructions

1	MOV Dst,Src	Copy content	1 Byte
2	MVI (R/M), 8-bit Data	Load 8-bit to Register/Memory	2 Byte
3	LDA 16-bit address	Load Accumulator	3 Byte
4	LDAX Rp(B/D)	Load the accumulator indirect	1 Byte
5	LXI Rp, 16-bit Data	Load the register pair immediate	3 Byte
6	STA 16-bit address	Store Accumulator	3 Byte
7	STAX Rp	Store Accumulator Indirect	1 Byte
8	LHLD 16-bit address	Load H and L registers direct	3 Byte
9	SHLD 16-bit address	Store H and L registers direct	3 Byte
10	XCHG None	Exchange H and L with D and E	1 Byte
11	SPHL None	Copy H and L registers to the stack pointer	1 Byte
12	XTHL None	Exchange H and L with top of stack	1 Byte
13	PUSH Rp	Push the register pair onto the stack	1 Byte
14	POP Rp	Pop off stack to the register pair	1 Byte
15	OUT 8-bit port address	Output from Accumulator to 8-bit port address	2 Byte
16	IN 8-bit port address	Input data to accumulator from a port with 8-bit address	2 Byte

GTU Exam Questions

Sr.	Questions	Marks	Year
1.	Explain the PUSH and POP instructions of the 8085 microprocessor with example.	4	W'17
2.	Explain the following instructions of the 8085 microprocessor with suitable example: STA, LDAX, XTHL	7	W'17
3.	Explain 8085 data transfer instructions with suitable examples.	7	S'18
4.	Define opcode and operand, and specify the opcode and the operand in the instruction MOV H, L	3	W'18
5.	Explain the following instructions of the 8085 microprocessor with suitable example: LHLD, SPHL, LDAX, XTHL	4	S'19
6.	The memory location 2070H holds the data byte F2H.Write instructions to transfer the data byte to the accumulator using three different opcodes: MOV, LDAX, and LDA.	4	W'19
7.	Register D contains 72H.Illustrate the instructions MOV and STAX to copy the contents of register B into memory location 8020H using indirect addressing.	4	W'19

Classification of 8085 Instructions

Based on Byte Size

- **One-byte Instructions**
  Requires one memory location
  to perform an operation
  E.g. CMA, ADD
- **Two-byte Instructions**
  Requires two memory locations
  to perform an operation
  E.g. MVI A,32H
- **Three-byte Instructions**
  Requires three memory locations
  to perform an operation
  E.g. JMP, CALL

Based on Function

- **Data Transfer Instructions**
- **Arithmetic Instructions**
- **Logic & Bit Manipulation Instructions**
- **Branch Instructions**
- **Control Instructions**

Arithmetic Instructions

Arithmetic Instructions

1	ADD R/M	Add register or memory, to the accumulator	1 Byte
2	ADC R/M	Add register to the accumulator with carry	1 Byte
3	ADI 8-bit data	Add the immediate to the accumulator	2 Byte
4	ACI 8-bit data	Add the immediate to the accumulator with carry	2 Byte
5	DAD R_p	Add the register pair to H and L registers	1 Byte
6	SUB R/M	Subtract the register/memory from accumulator	1 Byte
7	SBB R/M	Subtract the source and borrow from accumulator	1 Byte
8	SUI 8-bit data	Subtract the immediate from the accumulator	2 Byte
9	SBI 8-bit data	Subtract immediate from accumulator with borrow	2 Byte
10	INR R/M	Increment the register or the memory by 1	1 Byte
11	INX R_p	Increment register pair by 1	1 Byte
12	DCR R/M	Decrement the register or the memory by 1	1 Byte
13	DCX R_p	Decrement register pair by 1	1 Byte
14	DAA	Decimal adjust accumulator	1 Byte

ADD: Add register/memory to accumulator

Instruction		Description	Example
Opcode	Operand		
ADD	R/M	• The contents of the operand (register or memory) are added to the contents of the accumulator and the result is stored in the accumulator. • If the operand is a memory location, its location is specified by the contents of the HL registers. • All flags are modified to reflect the result of the addition.	**ADD B;** *A = A + B* **ADD M;** *A = A + M[HL]*

ADC: Add register to accumulator with carry

Instruction		Description	Example
Opcode	Operand		
ADC	R/M	• The contents of the operand (register or memory) and the **Carry** flag are added to the contents of the accumulator and the result is stored in the accumulator. • If the operand is a memory location, its location is specified by the contents of the HL registers. • All flags are modified to reflect the result of the addition.	**ADC B;** *A = A + B + CY* **ADC M;** *A = A + M[HL]+CY*

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

ADI: Add immediate 8-bit with accumulator

Instruction		Description	Example
Opcode	Operand		
ADI	8-bit data	• The 8-bit data (operand) is added to the contents of the accumulator and the result is stored in the accumulator. • All flags are modified to reflect the result of the addition.	**ADI 03;** *A = A + 03h*

ACI: Add immediate 8-bit to accumulator with carry

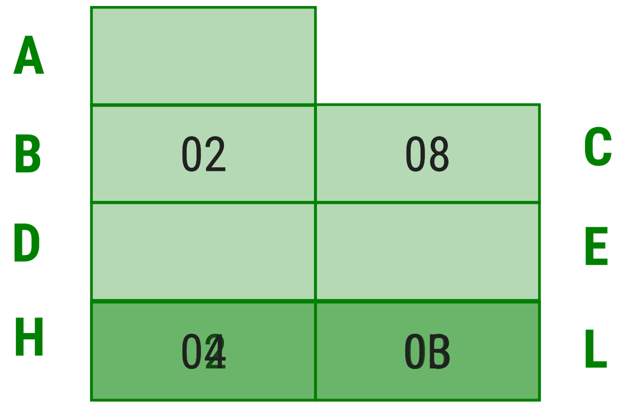
Instruction		Description	Example
Opcode	Operand		
ACI	8-bit data	• The 8-bit data (operand) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. • All flags are modified to reflect the result of the addition.	**ACI 03;** *A = A + 03h + CY*

DAD: Add register pair to H and L registers

Instruction		Description	Example
Opcode	Operand		
DAD	R_p	• The 16-bit contents of the specified register pair are added to the contents of the HL register and the sum is stored in the HL register. • The contents of the source register pair are not altered. • If the result is larger than 16 bits, the CY flag is set. No other flags are affected.	DAD B

DAD Instruction

Registers



```
+
   02    03
  ___________
   04    0B
```

DAD B

SUB: Subtract register/memory from accumulator

Instruction		Description	Example
Opcode	Operand		
SUB	R/M	• The contents of the operand (register or memory) is subtracted from the contents of the accumulator, and the result is stored in the accumulator. • If the operand is a memory location, its location is specified by the contents of the HL registers. • All flags are modified to reflect the result of the subtraction.	SUB B *; A=A-B* SUB M *; A=A-M[HL]*

SBB: Subtract source & borrow from accumulator

Instruction		Description	Example
Opcode	Operand		
SBB	R/M	• The contents of the operand (register or memory) and the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. • If the operand is a memory location, its location is specified by the contents of the HL registers. • All flags are modified to reflect the result of the subtraction.	SBB B; *A=A - (B+CY)* SBB M; *A=A-(M[HL]+CY)*

SUI: Subtract immediate 8-bit from accumulator

Instruction		Description	Example
Opcode	Operand		
SUI	8-bit data	• The 8-bit data (operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator.	SUI 08h; *A = A - 08h*

SBI: Subtract immediate from accumulator with borrow

Instruction		Description	Example
Opcode	Operand		
SBI	8-bit data	• The 8-bit data (operand) and the borrow (CY) are subtracted from the contents of the accumulator and the result is stored in the accumulator.	SBI 08h; *A=A - (08h+CY)*

INR: Increment register/memory by 1

Instruction		Description	Example
Opcode	Operand		
INR	R/M	• The contents of the designated register or memory is incremented by 1 and the result is stored at the same place. • If the operand is a memory location, its location is specified by the contents of the HL registers.	INR B;*B=B+01* INR M;*M[HL]=M[HL]+01*

INX: Increment register pair by 1

Instruction		Description	Example
Opcode	Operand		
INX	R_p	The contents of the designated register pair is incremented by 1 and the result is stored at the same place.	INX D; *DE=DE+0001*

DCR: Decrement register/ memory by 1

Instruction		Description	Example
Opcode	Operand		
DCR	R/M	• The contents of the designated register or memory is decremented by 1 and the result is stored in the same place. • If the operand is a memory location, its location is specified by the contents of the HL registers.	DCR B;*B=B-01* DCR M;*M[HL]=M[HL]-01*

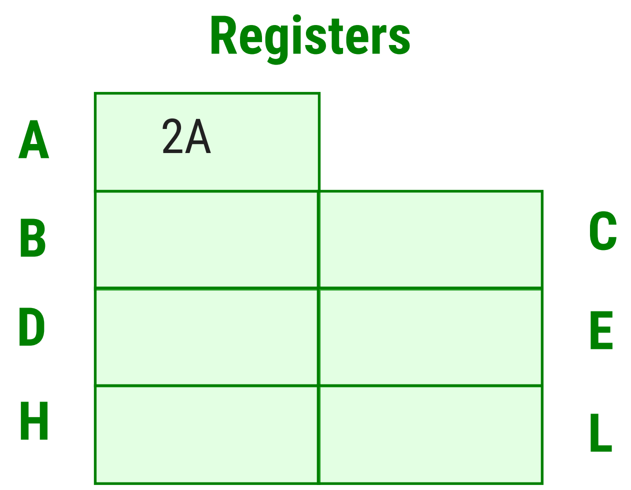
DCX: Decrement register pair by 1

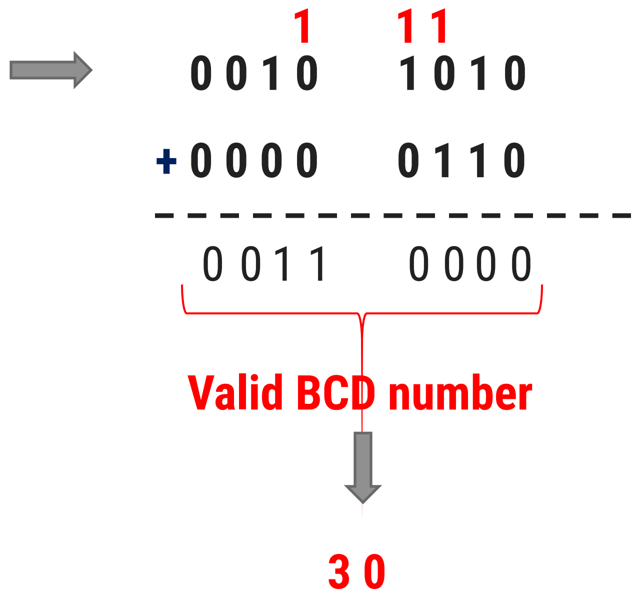
Instruction		Description	Example
Opcode	Operand		
DCX	R_p	The contents of the designated register pair is decremented by 1 and their result is stored at the same place.	DCX B; *BC=BC- 0001* DCX D; *DE=DE- 0001*

DAA: Decimal Adjust Accumulator

Instruction		Description	Example
Opcode	Operand		
DAA	None	• The contents of the accumulator is changed from a binary value to two 4-bit BCD digits. • If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits. • If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits.	DAA

DAA Instruction





Arithmetic Instructions

1	ADD R/M	Add register or memory, to the accumulator	1 Byte
2	ADC R/M	Add register to the accumulator with carry	1 Byte
3	ADI 8-bit data	Add the immediate to the accumulator	2 Byte
4	ACI 8-bit data	Add the immediate to the accumulator with carry	2 Byte
5	DAD R_p	Add the register pair to H and L registers	1 Byte
6	SUB R/M	Subtract the register/memory from accumulator	1 Byte
7	SBB R/M	Subtract the source and borrow from accumulator	1 Byte
8	SUI 8-bit data	Subtract the immediate from the accumulator	2 Byte
9	SBI 8-bit data	Subtract immediate from accumulator with borrow	2 Byte
10	INR R/M	Increment the register or the memory by 1	1 Byte
11	INX R_p	Increment register pair by 1	1 Byte
12	DCR R/M	Decrement the register or the memory by 1	1 Byte
13	DCX R_p	Decrement register pair by 1	1 Byte
14	DAA	Decimal adjust accumulator	1 Byte

Classification of 8085 Instructions

- **Based on Byte Size**
  - **One-byte Instructions**
    Requires one memory location
    to perform an operation
    E.g. CMA, ADD
  - **Two-byte Instructions**
    Requires two memory locations
    to perform an operation
    E.g. MVI A,32H
  - **Three-byte Instructions**
    Requires three memory locations
    to perform an operation
    E.g. JMP, CALL
- **Based on Function**
  - **Data Transfer Instructions**
  - **Arithmetic Instructions**
  - **Logic & Bit Manipulation Instructions**
  - **Branch Instructions**
  - **Control Instructions**

Branch Instructions

Branch Instructions

1	JMP	Jump unconditionally	3 Byte
2	JC	Jump on carry	3 Byte
3	JNC	Jump on no carry	3 Byte
4	JP	Jump on positive	3 Byte
5	JM	Jump on minus	3 Byte
6	JZ	Jump on zero	3 Byte
7	JNZ	Jump on no zero	3 Byte
8	JPE	Jump on parity even	3 Byte
9	JPO	Jump on parity odd	3 Byte
10	CALL	Call unconditionally	3 Byte
11	CC	Call on carry	3 Byte
12	CNC	Call on no carry	3 Byte
13	CP	Call on positive	3 Byte
14	CM	Call on minus	3 Byte
15	CZ	Call on zero	3 Byte
16	CNZ	Call on no zero	3 Byte
17	CPE	Call on parity even	3 Byte
18	CPO	Call on parity odd	3 Byte
19	RET	Return unconditionally	1 Byte
20	RC	Return on carry	1 Byte
21	RNC	Return on no carry	1 Byte
22	RP	Return on positive	1 Byte
23	RM	Return on minus	1 Byte
24	RZ	Return on zero	1 Byte
25	RNZ	Return on no zero	1 Byte
26	RPE	Return on parity even	1 Byte
27	RPO	Return on parity odd	1 Byte
28	PCHL	Load program counter with HL contents	1 Byte
29	RST	Restart	1 Byte

JMP: Jump unconditionally

Instruction		Description	Example
Opcode	Operand		
JMP	16-bit address	The program sequence is transferred to the memory address given in the operand.	JMP 000AH

JMP: Jump unconditionally

Memory **Address**	Instructions
0000H	MVI A,05
0002H	MOV B,A
0003H	MOV C,B
0004H	JMP **0009**
0007H	ADI 02
0009H	SUB B
000AH	HLT

Memory **Label**	Instructions
	MVI A,05
	MOV B,A
	MOV C,B
	JMP **L1**
	ADI 02
L1:	SUB B
	HLT

Branch Instruction

Jump Conditionally

Instruction		Description	Example
Opcode	Operand		
JC	16-bit address	Jump on Carry, Flag Status: CY=1	JC 2030H
JNC	16-bit address	Jump on No Carry, Flag Status: CY=0	JNC 2030H
JZ	16-bit address	Jump on Zero, Flag Status: Z=1	JZ 2030H
JNZ	16-bit address	Jump on No Zero, Flag Status: Z=0	JNZ 2030H
JP	16-bit address	Jump on Positive, Flag Status: S=0	JP 2030H
JM	16-bit address	Jump on Minus, Flag Status: S=1	JM 2030H
JPE	16-bit address	Jump on Parity Even, Flag Status: P=1	JPE 2030H
JPO	16-bit address	Jump on Parity Odd, Flag Status: P=0	JPO 2030H

RET: Return from subroutine unconditionally

Instruction		Description	Example
Opcode	Operand		
RET		The program sequence is transferred from the subroutine to the calling program.	RET

CALL: Call Unconditionally

Instruction		Description	Example
Opcode	Operand		
CALL	16-bit address	Instruction transfers the program sequence to the memory address given in the operand. Before transferring, the address of the next instruction(PC) is pushed onto the stack.	CALL 000AH

CALL: Call Unconditionally

Line	Instruction	Address	PC
1	LXI H,1002	[0000]	[0003]
2	LXI D,3002	[0003]	[0006]
3	**CALL ADD1**	[0006]	**[0009]**
4	LXI B,4002	[0009]	[000C]
5	**ADD1**:MOV A,D	[000C]	[000D]
6	ADD H	[000D]	[000E]
7	**RET**	[000E]	

SP→	05	[2008]
SP→	**00**	[2007]
SP→	**09**	[2006]
		[2005]

Branch Instruction

CALL Conditionally

Instruction		Description	Example
Opcode	Operand		
CC	16-bit address	Call on Carry, Flag Status: CY=1	CC 2030H
CNC	16-bit address	Call on No Carry, Flag Status: CY=0	CNC 2030H
CZ	16-bit address	Call on Zero, Flag Status: Z=1	CZ 2030H
CNZ	16-bit address	Call on No Zero, Flag Status: Z=0	CNZ 2030H
CP	16-bit address	Call on Positive, Flag Status: S=0	CP 2030H
CM	16-bit address	Call on Minus, Flag Status: S=1	CM 2030H
CPE	16-bit address	Call on Parity Even, Flag Status: P=1	CPE 2030H
CPO	16-bit address	Call on Parity Odd, Flag Status: P=0	CPO 2030H

Branch Instruction

Return from Subroutine

Instruction		Description	Example
Opcode	Operand		
RC	16-bit address	Return on Carry, CY=1	RC
RNC	16-bit address	Return on No Carry, CY=0	RNC
RZ	16-bit address	Return on Zero, Z=1	RZ
RNZ	16-bit address	Return on No Zero, Z=0	RNZ
RP	16-bit address	Return on Positive, S=0	RP
RM	16-bit address	Return on Minus, S=1	RP
RPE	16-bit address	Return on Parity Even, Flag Status: P=1	RPE
RPO	16-bit address	Return on Parity Odd, Flag Status: P=0	RPO

PCHL: Load program counter with HL contents

Instruction		Description	Example
Opcode	Operand		
PCHL	None	• The contents of registers H & L are copied into the program counter. • The contents of H are placed as the high-order byte and the contents of L as the low-order byte.	PCHL

RST: Restart

Instruction		Description	Example
Opcode	Operand		
RST	0-7(N)	The RST instruction is used as software instructions in a program to transfer the program execution to one of the following eight locations. **Instruction** **Restart Address** RST 0 0000H RST 1 0008H RST 2 0010H RST 3 0018H RST 4 0020H RST 5 0028H RST 6 0030H RST 7 0038H	RST 5

Branch Instruction

The 8085 has additionally 4 interrupts, which can generate RST instructions internally and doesn't require any external hardware.

Instruction		Description	Example
Opcode	Operand		
TRAP	None	It restart from address 0024H	TRAP
RST 5.5	None	It restart from address 002CH	RST 5.5
RST 6.5	None	It restart from address 0034H	RST 6.5
RST 7.5	None	It restart from address 003CH	RST 7.5

Branch Instructions

1	JMP	Jump unconditionally	3 Byte
2	JC	Jump on carry	3 Byte
3	JNC	Jump on no carry	3 Byte
4	JP	Jump on positive	3 Byte
5	JM	Jump on minus	3 Byte
6	JZ	Jump on zero	3 Byte
7	JNZ	Jump on no zero	3 Byte
8	JPE	Jump on parity even	3 Byte
9	JPO	Jump on parity odd	3 Byte
10	CALL	Call unconditionally	3 Byte
11	CC	Call on carry	3 Byte
12	CNC	Call on no carry	3 Byte
13	CP	Call on positive	3 Byte
14	CM	Call on minus	3 Byte
15	CZ	Call on zero	3 Byte
16	CNZ	Call on no zero	3 Byte
17	CPE	Call on parity even	3 Byte
18	CPO	Call on parity odd	3 Byte
19	RET	Return unconditionally	1 Byte
20	RC	Return on carry	1 Byte
21	RNC	Return on no carry	1 Byte
22	RP	Return on positive	1 Byte
23	RM	Return on minus	1 Byte
24	RZ	Return on zero	1 Byte
25	RNZ	Return on no zero	1 Byte
26	RPE	Return on parity even	1 Byte
27	RPO	Return on parity odd	1 Byte
28	PCHL	Load program counter with HL contents	1 Byte
29	RST	Restart	1 Byte

Classification of 8085 Instructions

- **Based on Byte Size**
  - **One-byte Instructions**
    Requires one memory location to perform an operation
    E.g. CMA, ADD
  - **Two-byte Instructions**
    Requires two memory locations to perform an operation
    E.g. MVI A,32H
  - **Three-byte Instructions**
    Requires three memory locations to perform an operation
    E.g. JMP, CALL
- **Based on Function**
  - **Data Transfer Instructions**
  - **Arithmetic Instructions**
  - **Logic & Bit Manipulation Instructions**
  - **Branch Instructions**
  - **Control Instructions**

Logical & Bit Manipulation Instructions

Logical & Bit Manipulation Instructions

1	CMP	Compare register or memory with accumulator	1 Byte
2	CPI	Compare immediate with accumulator	2 Byte
3	ANA	Logical AND register or memory with accumulator	1 Byte
4	ANI	Logical AND immediate with accumulator	2 Byte
5	XRA	Exclusive OR register or memory with accumulator	1 Byte
6	XRI	Exclusive OR immediate with accumulator	2 Byte
7	ORA	Logical OR register or memory with accumulator	1 Byte
8	ORI	Logical OR immediate with accumulator	2 Byte
9	RLC	Rotate accumulator left	1 Byte
10	RRC	Rotate accumulator right	1 Byte
11	RAL	Rotate accumulator left through carry	1 Byte
12	RAR	Rotate accumulator right through carry	1 Byte
13	CMA	Complement accumulator	1 Byte
14	CMC	Complement carry	1 Byte
15	STC	Set carry	1 Byte

CMP: Compare register/memory with accumulator

Instruction		Description	Example
Opcode	Operand		
CMP	R/M	The contents of the operand (register or memory) is compared with the contents of the accumulator. Both contents are preserved. The result of the comparison is shown by setting the flags: 1. if (A) < (reg/mem): carry flag is set(1). 2. if (A) = (reg/mem): zero flag is set(1). 3. if (A) > (reg/mem): carry and zero flags are reset(0).	CMP B CMP M

CPI: Compare immediate with accumulator

Instruction		Description	Example
Opcode	Operand		
CPI	8-bit data	• The second byte data is compared with the contents of the accumulator. • The values being compared remain unchanged. The result of the comparison is shown by setting the flags: 1. if (A) < data: carry flag is set(1). 2. if (A) = data: zero flag is set(1). 3. if (A) > data: carry and zero flags are reset(0).	CPI 89H

ANA: AND register/memory with accumulator

Instruction		Description	Example
Opcode	Operand		
ANA	R/M	• The contents of the accumulator are logically ANDed with the contents of the operand (register or memory), and the result is placed in the accumulator. • If the operand is a memory location, its address is specified by the contents of HL registers. • S, Z, P are modified to reflect the result of the operation. • CY is reset. AC is set.	ANA B ANA M

ANI: AND immediate with accumulator

Instruction		Description	Example
Opcode	Operand		
ANI	8-bit data	• The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. • S, Z, P are modified to reflect the result of the operation. • CY is reset. AC is set.	ANI 02H

ORA: OR register/memory with accumulator

Instruction		Description	Example
Opcode	Operand		
ORA	R/M	• The contents of the accumulator is logically ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. • If the operand is a memory location, its address is specified by the contents of HL registers. • S, Z, P are modified to reflect the result of the operation. CY and AC are reset.	ORA B ORA M

ORI: OR immediate with accumulator

Instruction		Description	Example
Opcode	Operand		
ORI	8-bit data	• The contents of the accumulator is logically ORed with the 8-bit data (operand) and the result is placed in the accumulator. • S, Z, P are modified to reflect the result of the operation. • CY is reset. AC is set.	ORI 02H

XRA: Exclusive OR register/memory with accumulator

Instruction		Description	Example
Opcode	Operand		
XRA	R/M	• The contents of the accumulator is Exclusive ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. • If the operand is a memory location, its address is specified by the contents of HL registers. • S, Z, P are modified to reflect the result of the operation. CY and AC are reset.	XRA B XRA M

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

XRI: Exclusive OR immediate with accumulator

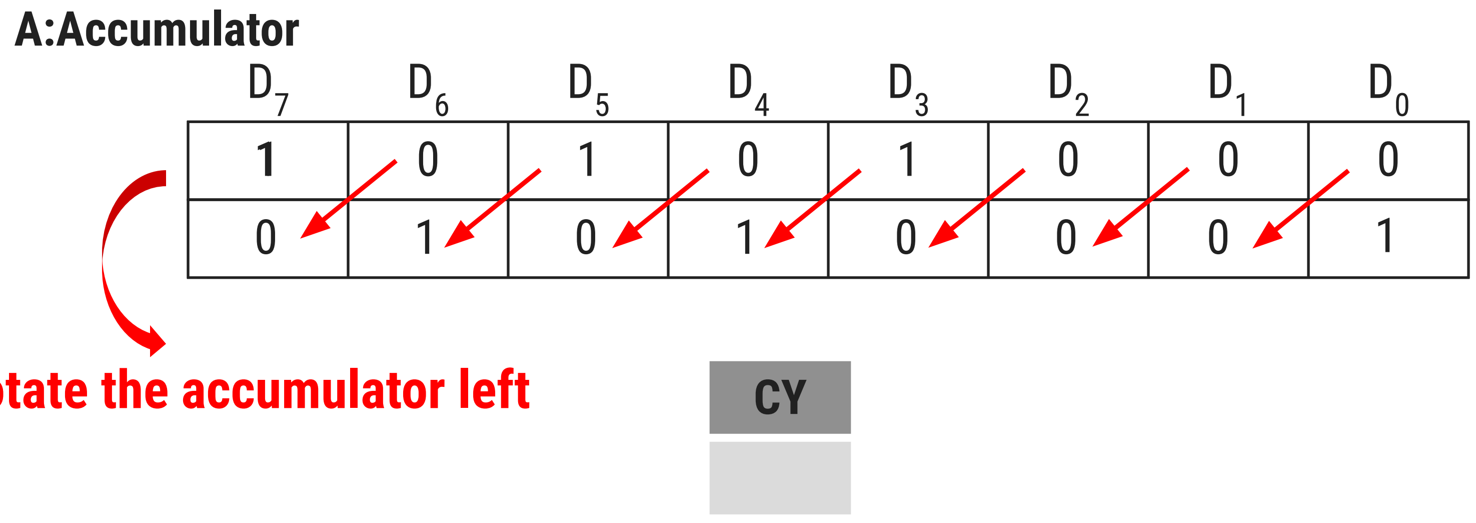
Instruction		Description	Example
Opcode	Operand		
XRI	8-bit data	• The contents of the accumulator are Exclusive Ored with the 8-bit data (operand) and the result is placed in the accumulator. • S, Z, P are modified to reflect the result of the operation. • CY is reset. AC is set.	XRI 02H

RLC: Rotate accumulator left

Instruction		Description	Example
Opcode	Operand		
RLC	None	• Each binary bit of the accumulator is rotated left by one position. • Bit D_7 is placed in the position of D_0 as well as in the Carry flag(CY). • CY is modified according to bit D_7. • S, Z, P, AC are not affected.	RLC

Logical Instruction

RLC

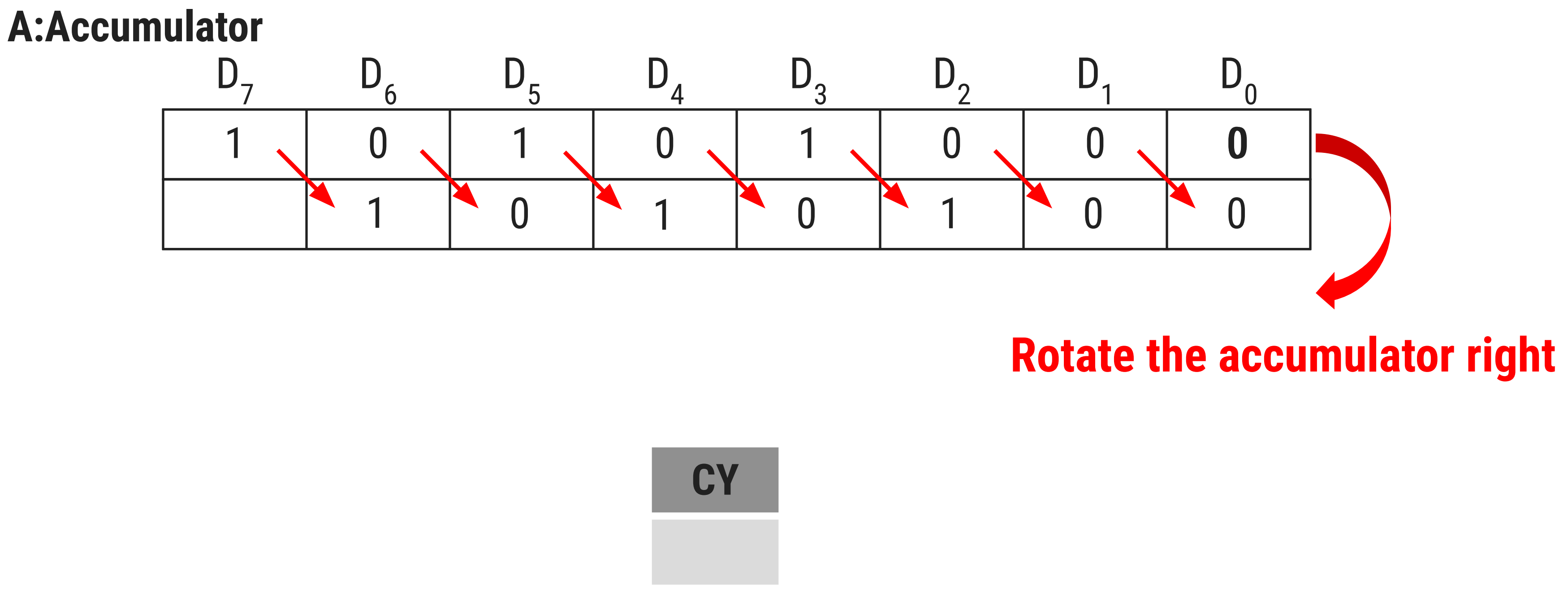


Rotate the accumulator left

RRC: Rotate accumulator right

Instruction		Description	Example
Opcode	Operand		
RRC	None	• Each binary bit of the accumulator is rotated right by one position. • Bit D_0 is placed in the position of D_7 as well as in the Carry flag(CY). • CY is modified according to bit D_0. • S, Z, P, AC are not affected.	RRC

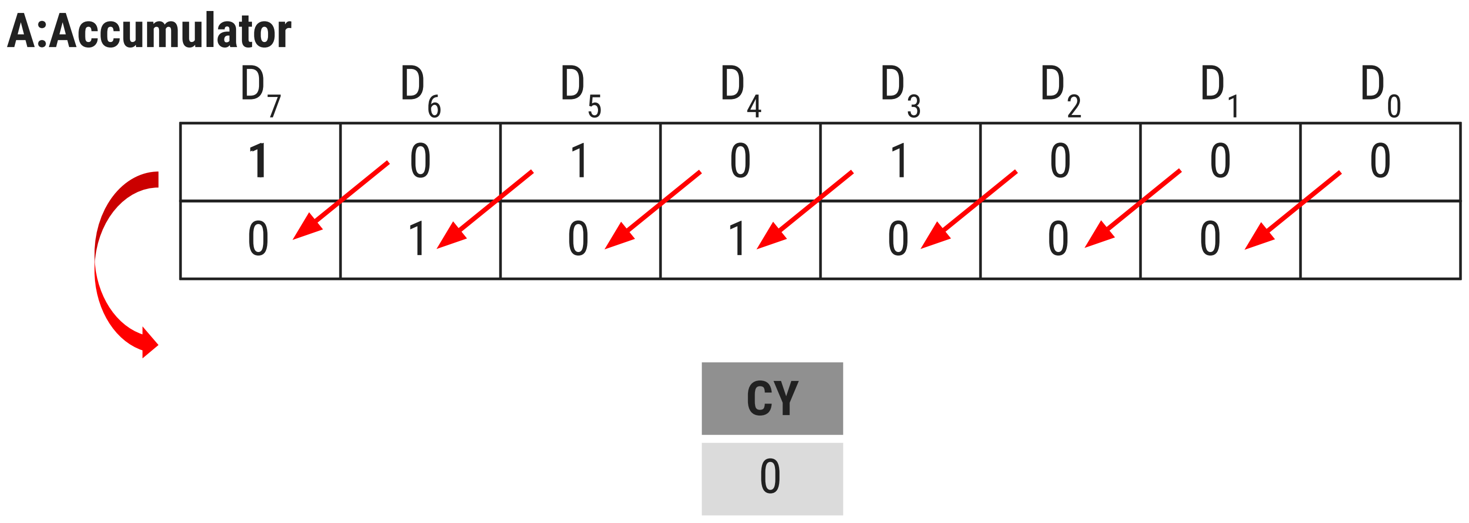
RRC: Example


A:Accumulator
D7 D6 D5 D4 D3 D2 D1 D0
1 0 1 0 1 0 0 0
1 0 1 0 1 0 0
Rotate the accumulator right
CY

RAL: Rotate accumulator left through carry

Instruction		Description	Example
Opcode	Operand		
RAL	None	• Each binary bit of the accumulator is rotated left by one position through the Carry flag. • Bit D_7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D_0. • CY is modified according to bit D_7. • S, Z, P, AC are not affected.	RAL

RAL: Example

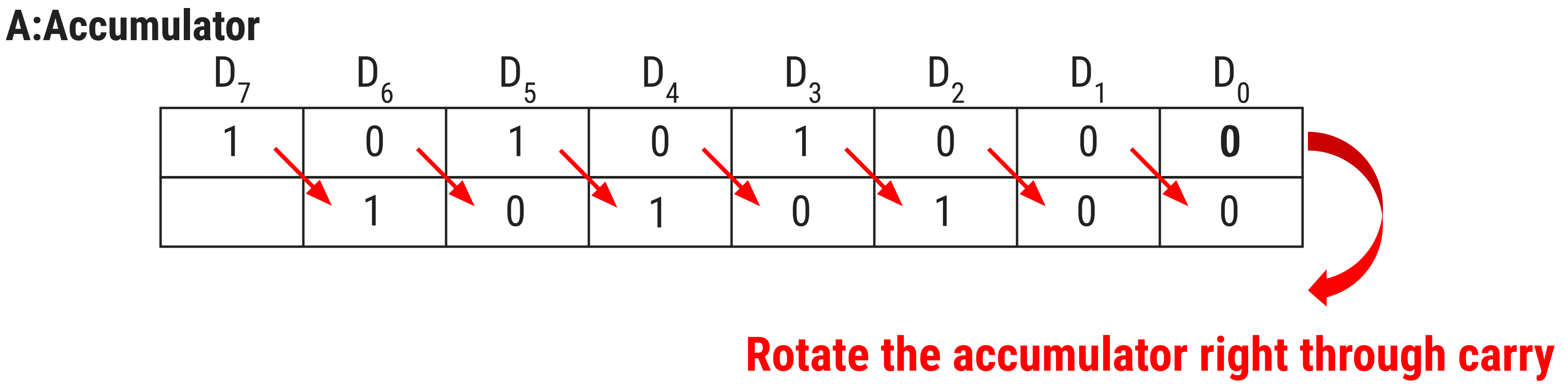


Rotate the accumulator left through carry

RAR: Rotate accumulator right through carry

Instruction		Description	Example
Opcode	Operand		
RAR	None	• Each binary bit of the accumulator is rotated right by one position through the Carry flag. • Bit D_0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D_7. • CY is modified according to bit D_0. • S, Z, P, AC are not affected.	RAR

RAR: Example



CY
1

CMA: Complement accumulator

Instruction		Description	Example
Opcode	Operand		
CMA	None	The contents of the accumulator are complemented. No flags are affected.	CMA

A $\boxed{2A}$ ⇨ CMA

0 0 1 0   1 0 1 0

1 1 0 1   0 1 0 1

D   5

Logical Instruction

CMC: Complement Carry

Instruction		Description	Example
Opcode	Operand		
CMC	None	The Carry flag is complemented. No other flags are affected.	CMC

STC: Set Carry

Instruction		Description	Example
Opcode	Operand		
STC	None	The Carry flag is set(1). No other flags are affected.	STC

Logical & Bit Manipulation Instructions

1	CMP	Compare register or memory with accumulator	1 Byte
2	CPI	Compare immediate with accumulator	2 Byte
3	ANA	Logical AND register or memory with accumulator	1 Byte
4	ANI	Logical AND immediate with accumulator	2 Byte
5	XRA	Exclusive OR register or memory with accumulator	1 Byte
6	XRI	Exclusive OR immediate with accumulator	2 Byte
7	ORA	Logical OR register or memory with accumulator	1 Byte
8	ORI	Logical OR immediate with accumulator	2 Byte
9	RLC	Rotate accumulator left	1 Byte
10	RRC	Rotate accumulator right	1 Byte
11	RAL	Rotate accumulator left through carry	1 Byte
12	RAR	Rotate accumulator right through carry	1 Byte
13	CMA	Complement accumulator	1 Byte
14	CMC	Complement carry	1 Byte
15	STC	Set carry	1 Byte

Classification of 8085 Instructions

Based on Byte Size

- **One-byte Instructions**
  Requires one memory location
  to perform an operation
  E.g. CMA, ADD
- **Two-byte Instructions**
  Requires two memory locations
  to perform an operation
  E.g. MVI A,32H
- **Three-byte Instructions**
  Requires three memory locations
  to perform an operation
  E.g. JMP, CALL

Based on Function

- **Data Transfer Instructions**
- **Arithmetic Instructions**
- **Logic & Bit Manipulation Instructions**
- **Branch Instructions**
- **Control Instructions**

Control Instructions

Control Instructions

1	NOP	No operation	1 Byte
2	HLT	Halt	1 Byte
3	DI	Disable interrupts	1 Byte
4	EI	Enable interrupts	1 Byte
5	RIM	Read interrupt mask	1 Byte
6	SIM	Set interrupt mask	1 Byte

Control Instructions

Instruction		Description	Example
Opcode	Operand		
NOP	None	• No operation is performed. The instruction is fetched and decoded. However no operation is executed. • It is used to increase processing time of execution. • 1 CPU cycle is "wasted" to execute a NOP instruction.	NOP
HLT	None	The CPU finishes executing the current instruction and stops further execution. An interrupt or reset is necessary to exit from the halt state.	HLT

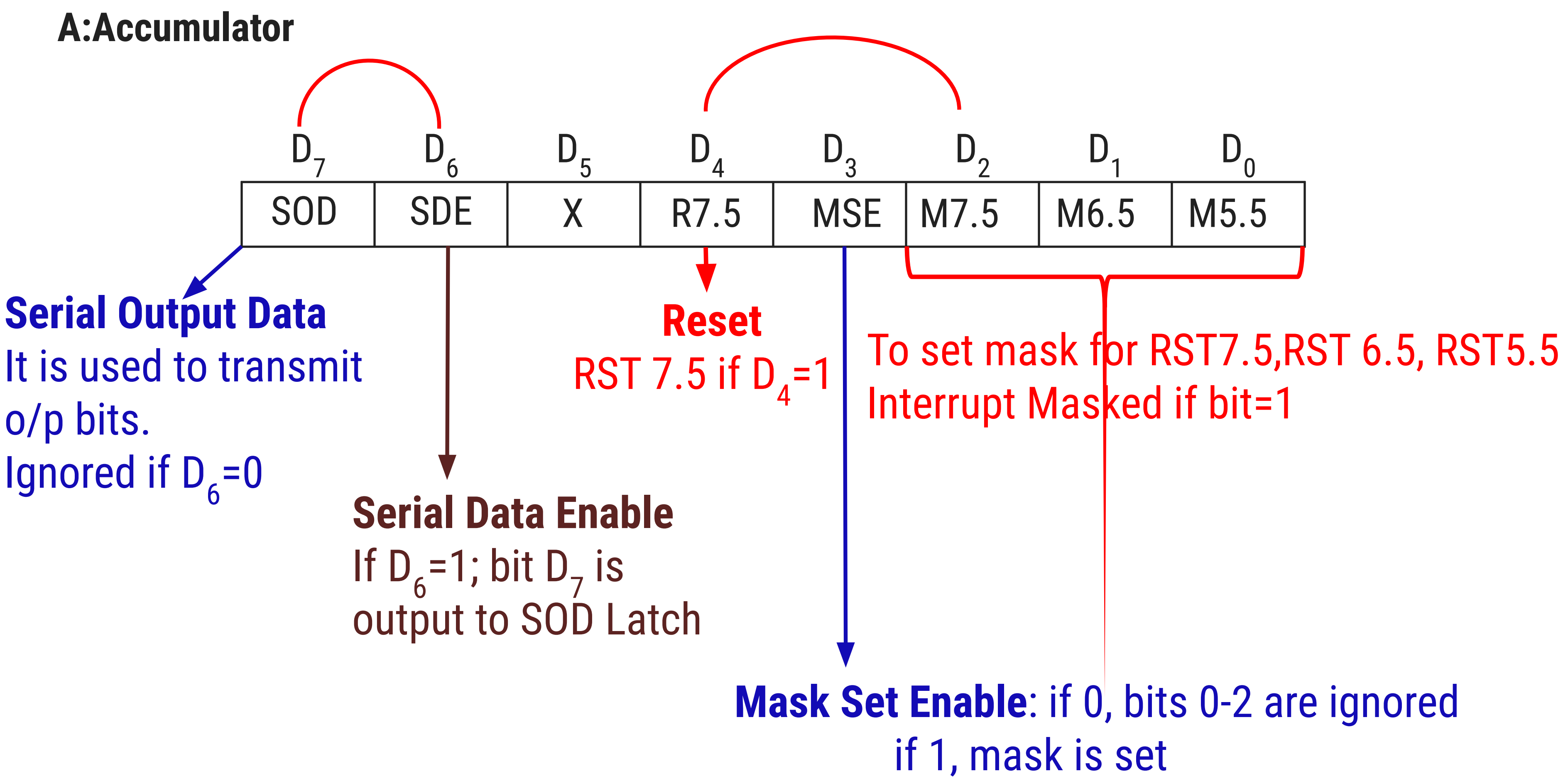
Control Instructions

Instruction		Description	Example
Opcode	Operand		
DI	None	**Disable Interrupt** The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected.	DI
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).	EI

SIM: Set Interrupt Mask

Instruction		Description	Example
Opcode	Operand		
SIM	None	This is a multipurpose instruction used to : 1. Set the status of interrupts 7.5, 6.5, 5.5 2. Set serial data input bit. The instruction loads eight bits in the accumulator with the following interpretations.	SIM

SIM Instruction



Serial Output Data
It is used to transmit o/p bits.
Ignored if D_6=0

Serial Data Enable
If D_6=1; bit D_7 is output to SOD Latch

Reset
RST 7.5 if D_4=1

To set mask for RST7.5,RST 6.5, RST5.5
Interrupt Masked if bit=1

Mask Set Enable: if 0, bits 0-2 are ignored
if 1, mask is set

SIM Instruction

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Accumulator	SOD	SDE	X	R7.5	MSE	M7.5	M6.5	M5.5

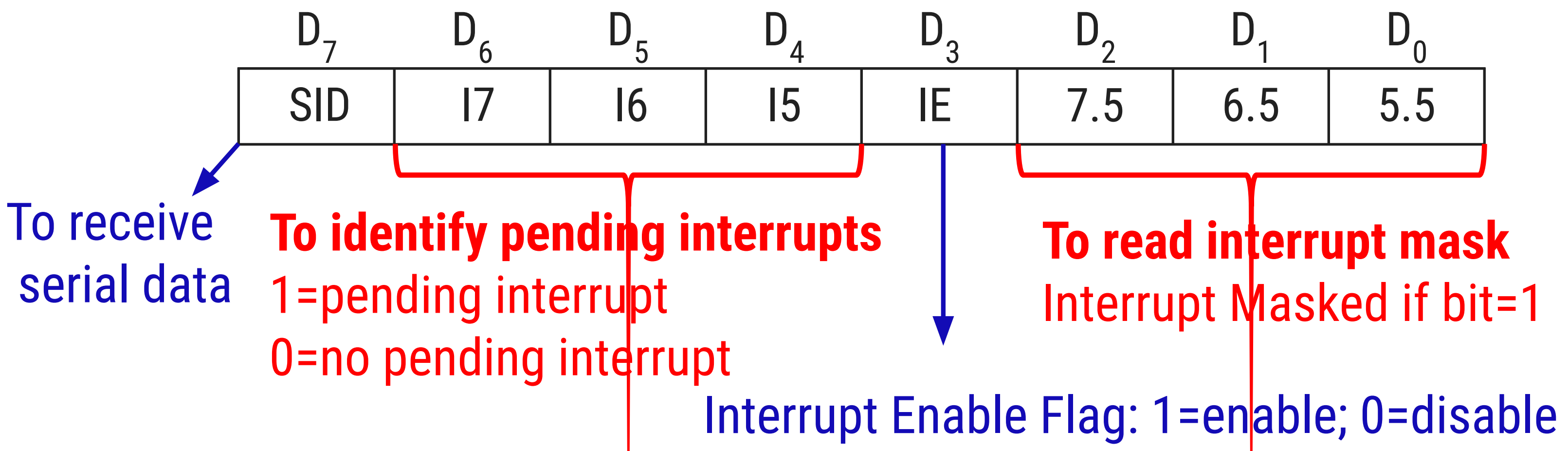
Example 1: MVI A,08H
SIM

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	1	0	0	0

RIM: Read Interrupt Mask

Instruction		Description	Example
Opcode	Operand		
RIM	None	This is a multipurpose instruction used to 1. Read the status of interrupts 7.5, 6.5, 5.5 2. Read serial data input bit. It reads eight bits from accumulator with following interpretations.	RIM

A:Accumulator



D7 D6 D5 D4 D3 D2 D1 D0
SID I7 I6 I5 IE 7.5 6.5 5.5
To receive serial data
To identify pending interrupts
1=pending interrupt
0=no pending interrupt
Interrupt Enable Flag: 1=enable; 0=disable
To read interrupt mask
Interrupt Masked if bit=1

Control Instructions

1	NOP	No operation	1 Byte
2	HLT	Halt	1 Byte
3	DI	Disable interrupts	1 Byte
4	EI	Enable interrupts	1 Byte
5	RIM	Read interrupt mask	1 Byte
6	SIM	Set interrupt mask	1 Byte

GTU Exam Questions

Sr.	Questions	Marks	Year
1.	Explain the SIM and RIM instructions of the 8085 microprocessor.	3	S’19
2.	Explain the Functions of following instructions: i. RAL ii. LDAX iii. ADC	3	W’19
3.	Explain the Functions of following instructions: i. RLC ii. LHLD iii. SBB	3	W’19

References

Book:	Microprocessor Architecture, Programming, and Applications with the 8085, Ramesh S. Gaonkar Pub: Penram International
Mobile Application	*8085 and 8086 Microprocessor Opcodes app from Play Store:* *http://tiny.cc/aopcodes*

Thank You