Semester- III Paper- III

DSC -1005 C: Electronics Communication and Microprocessor 8085

Section II: Microprocessor 8085

UNIT 3: Instruction Set of 8085 Microprocessor

Presented By:
Dr. C. B. Patil

Vivekanad College (Autonomous), Kolhapur

Syllabus:

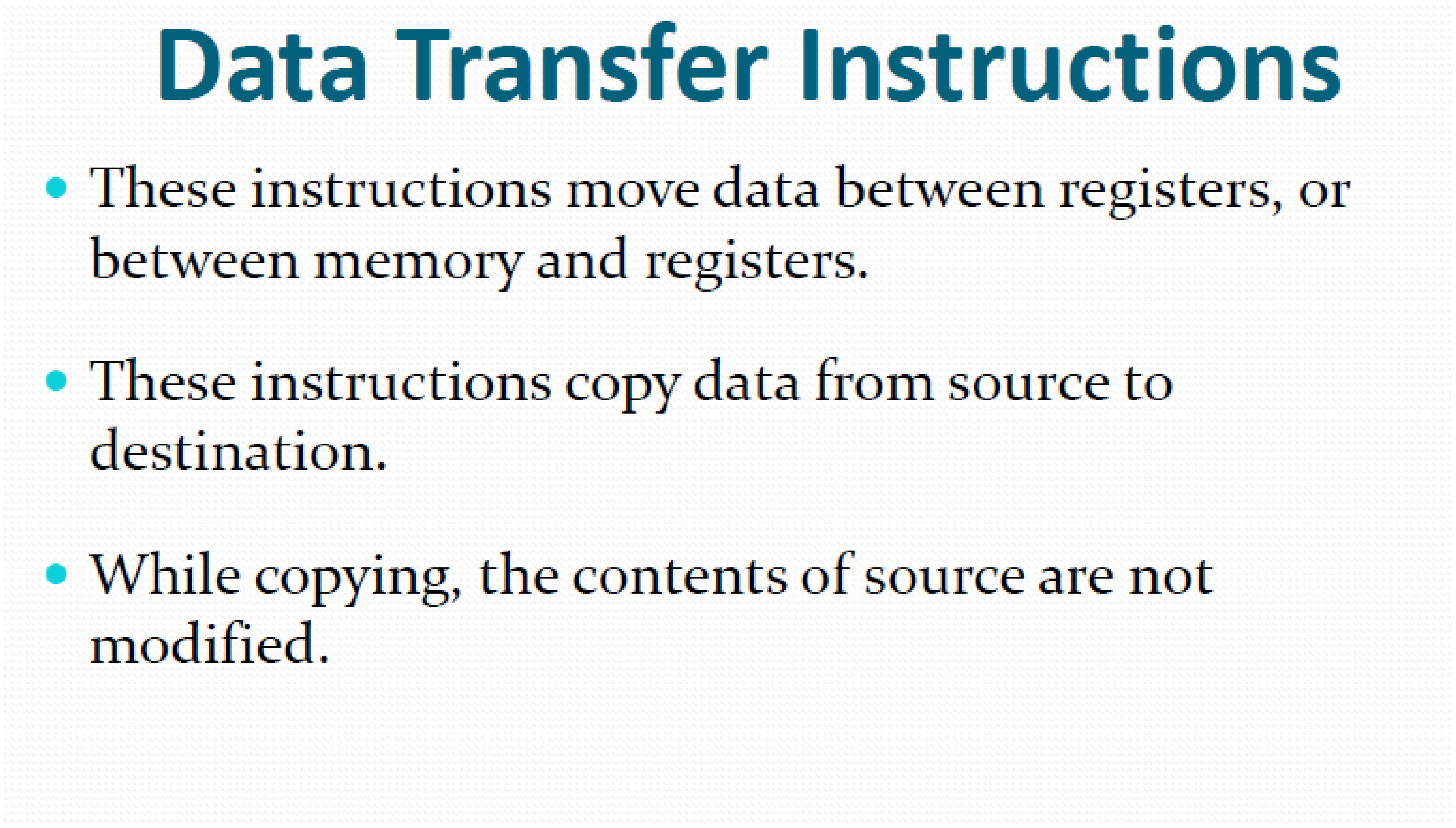
- Instruction set, classification of Instruction Set, Instruction format, Addressing modes of Instructions, Instruction set: Data transfer (including stacks), Arithmetic, logical, branch and control instructions).

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

1. **Data Transfer Instructions**
2. **Arithmetic Instructions**
3. **Logical Instructions**
4. **Branch Control Instructions**
5. **Stack, I/O and Machine Control Instructions**



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

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 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 Instructions

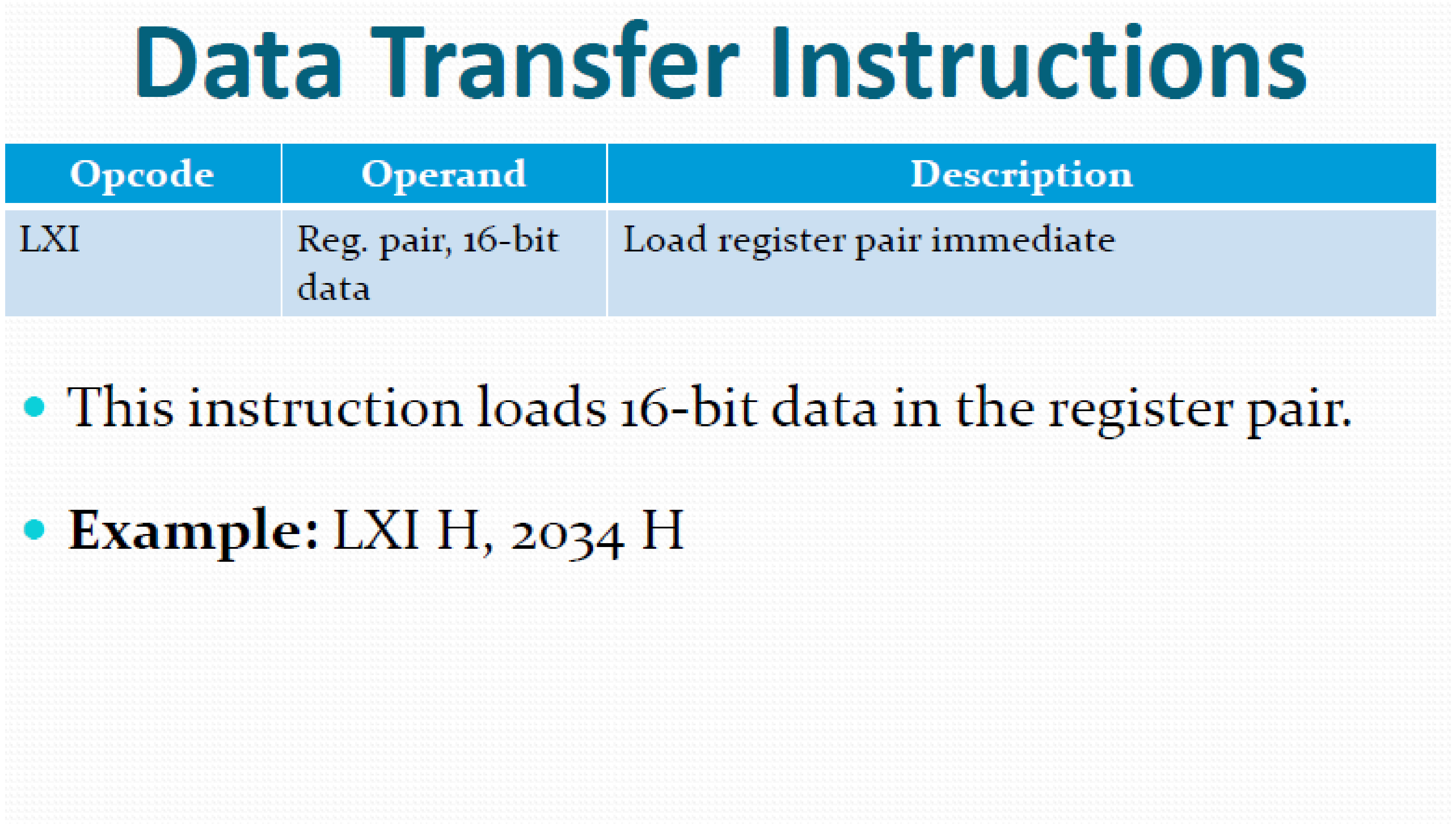
Opcode	Operand	Description
LDA	16-bit address	Load Accumulator

- 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 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 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 2040 H

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 2500 H

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 Instructions

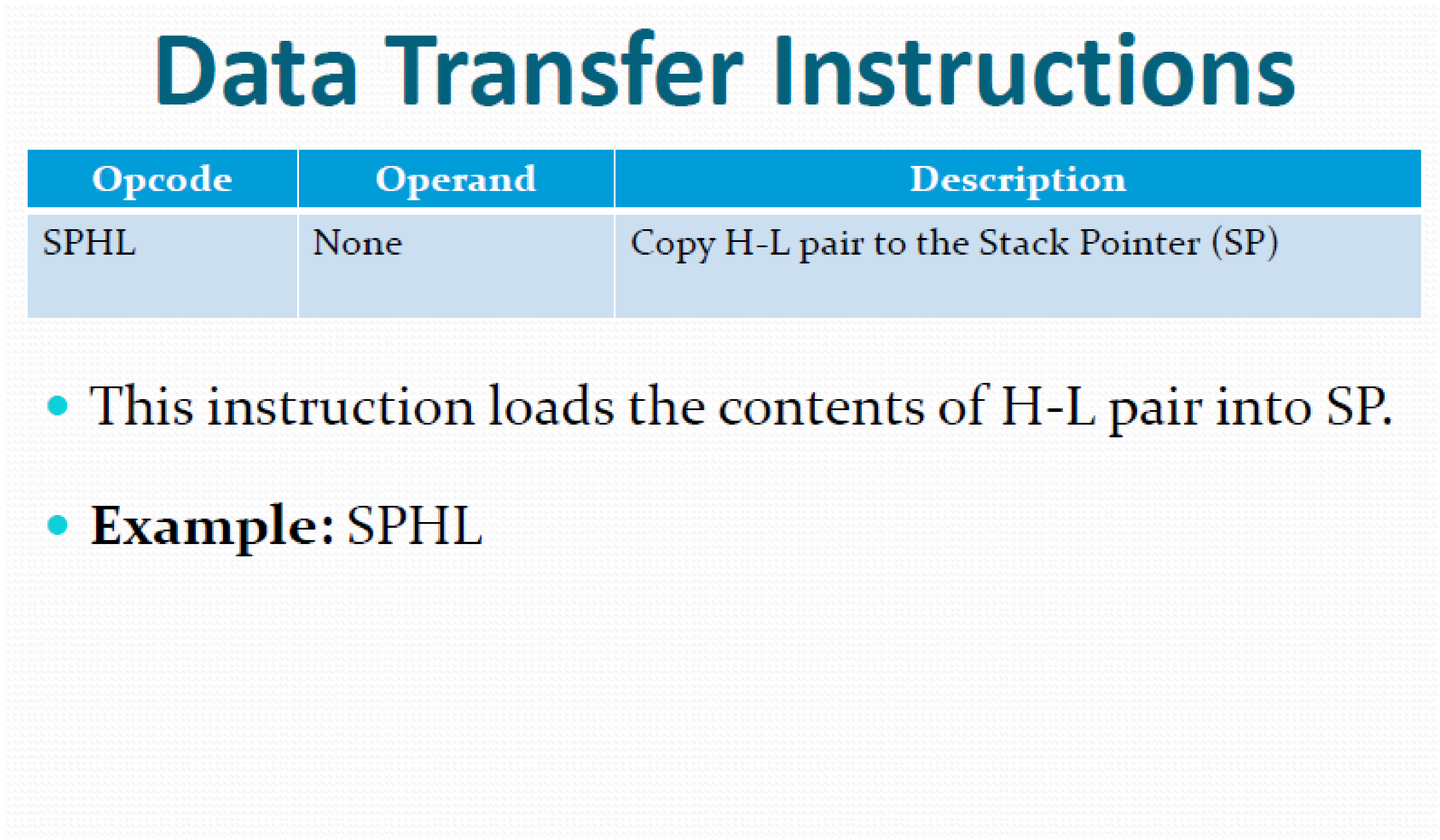
Opcode	Operand	Description
SHLD	16-bit address	Store H-L registers direct

- 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 2550 H

Data Transfer Instructions

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

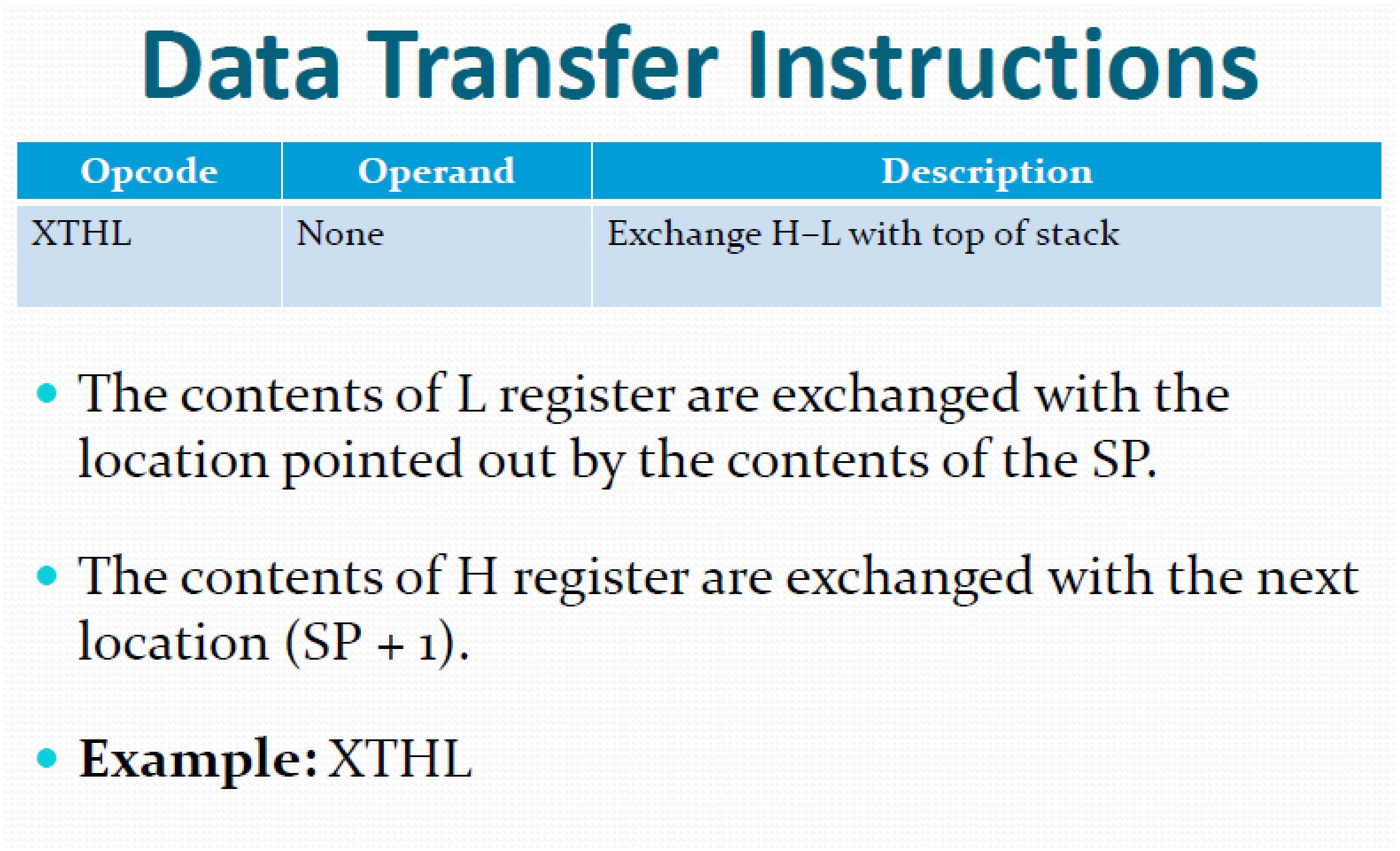
- The contents of register H are exchanged with the contents of register D.
- The contents of register L are exchanged with the contents of register E.
- **Example:** XCHG



Data Transfer 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 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 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 Instructions

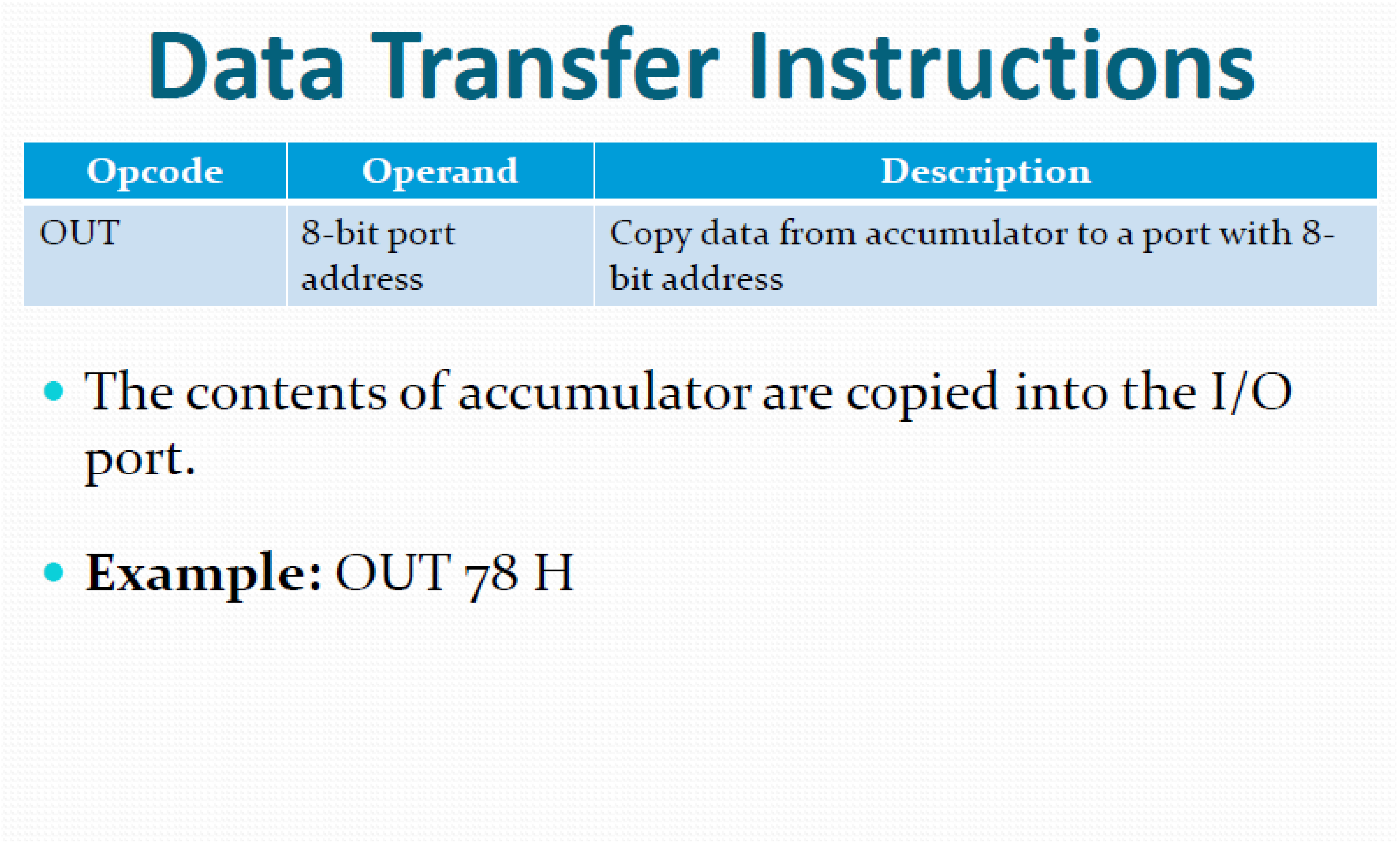
Opcode	Operand	Description
PUSH	Reg. pair	Push register pair onto stack

- 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 B

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 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

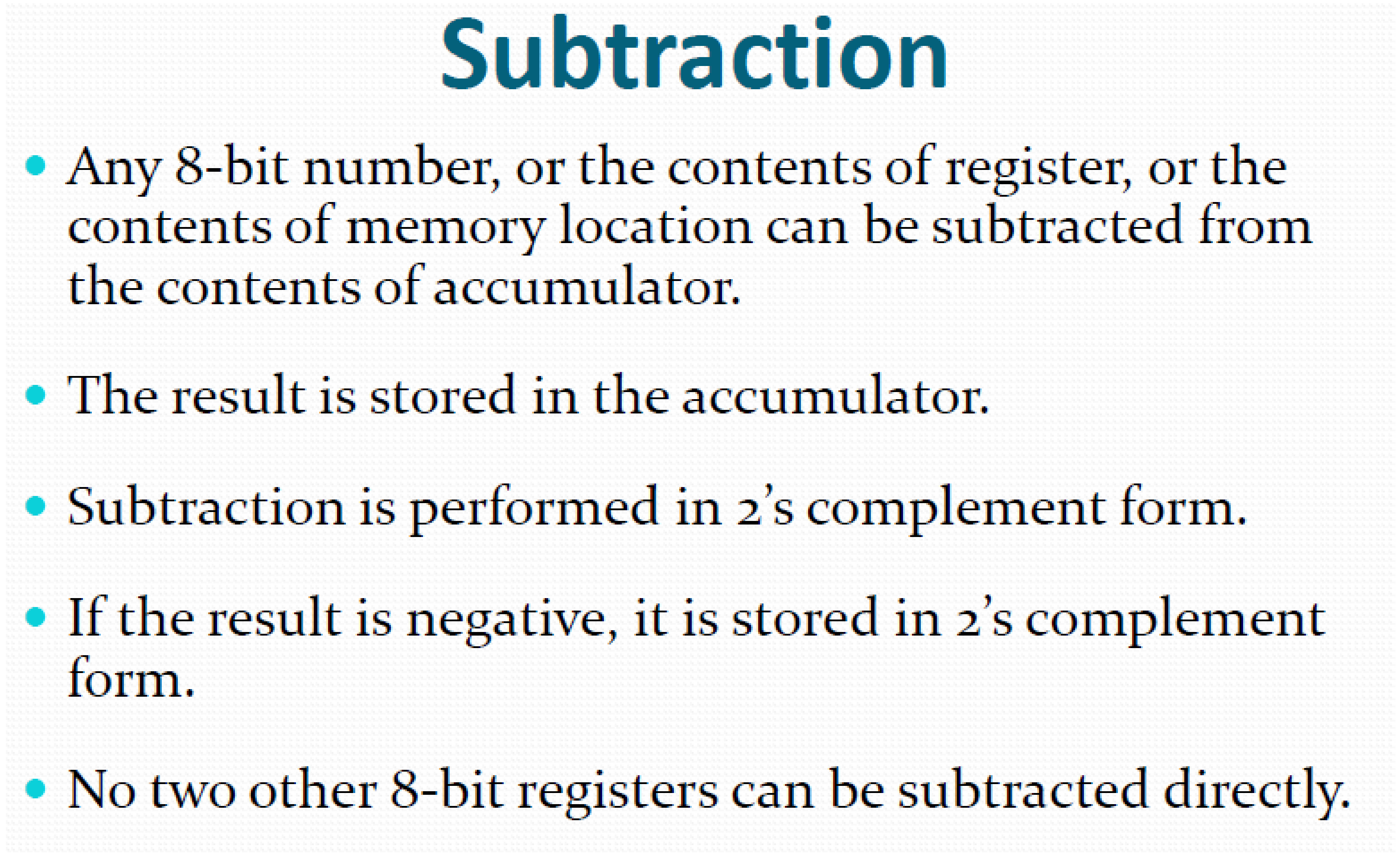
Data Transfer Instructions

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

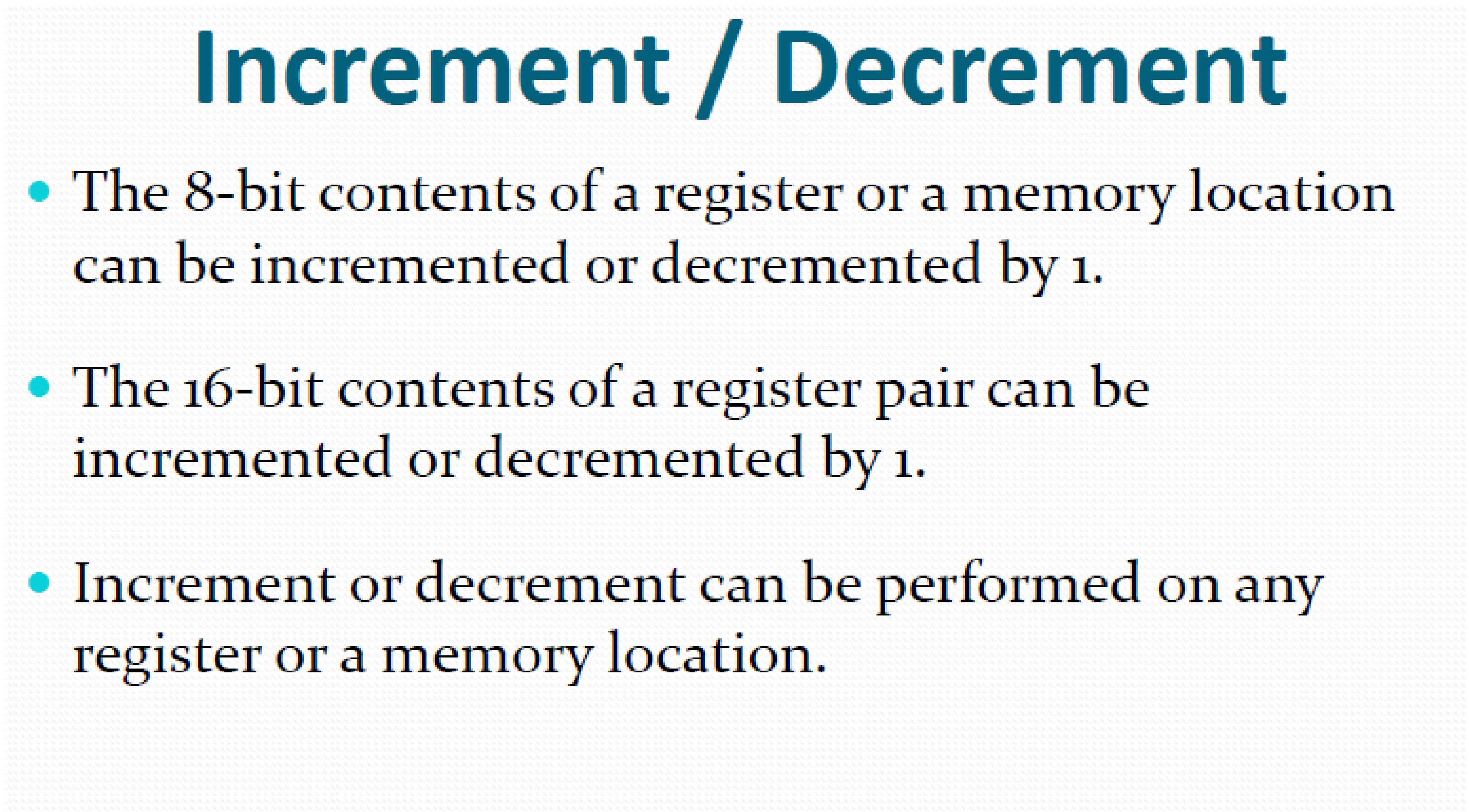
Addition

- 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.
- **Example:** The contents of register B cannot be added directly to the contents of register C.



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.



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 Instructions

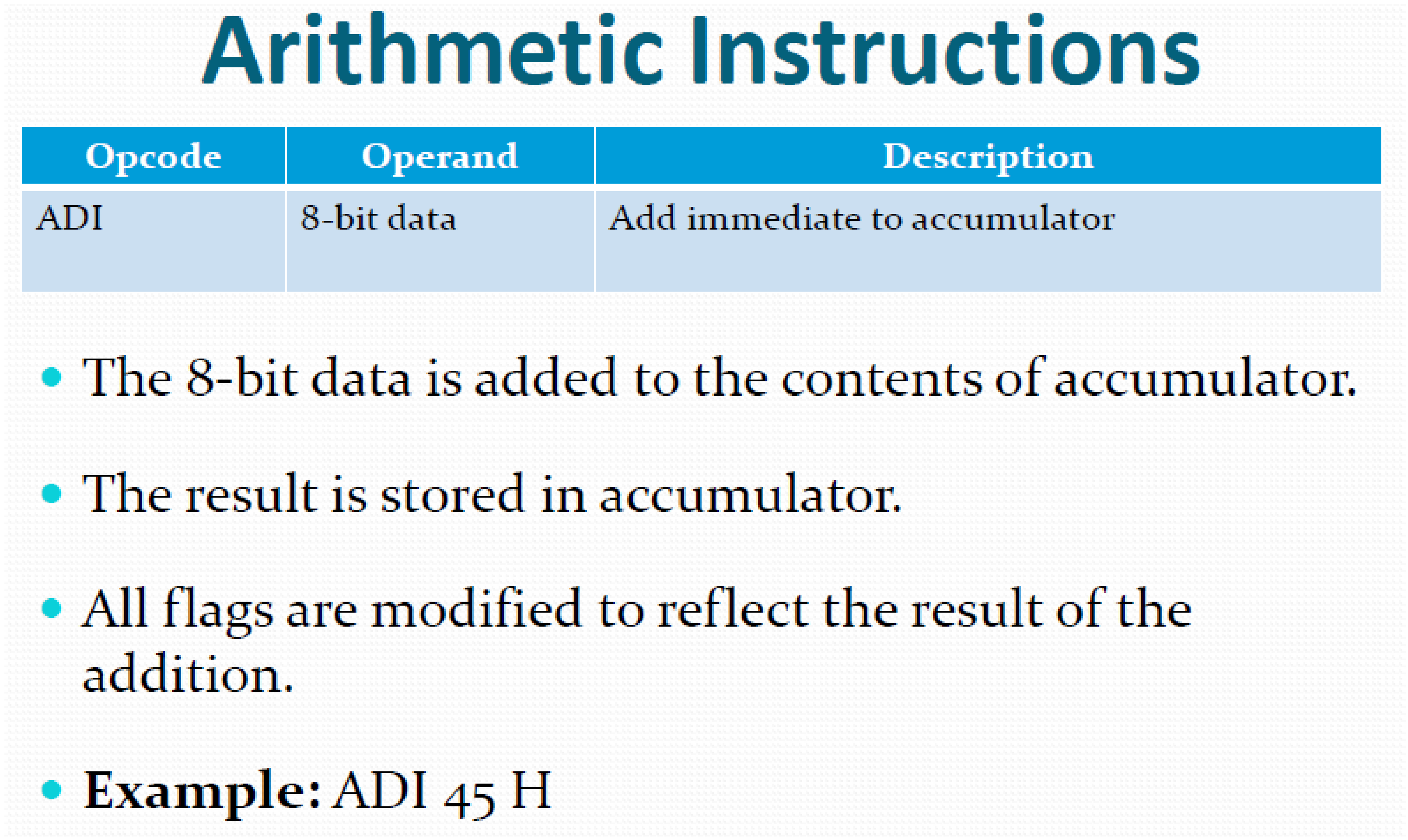
Opcode	Operand	Description
ADD	R M	Add register or memory to accumulator

- 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 or ADD M

Arithmetic Instructions

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

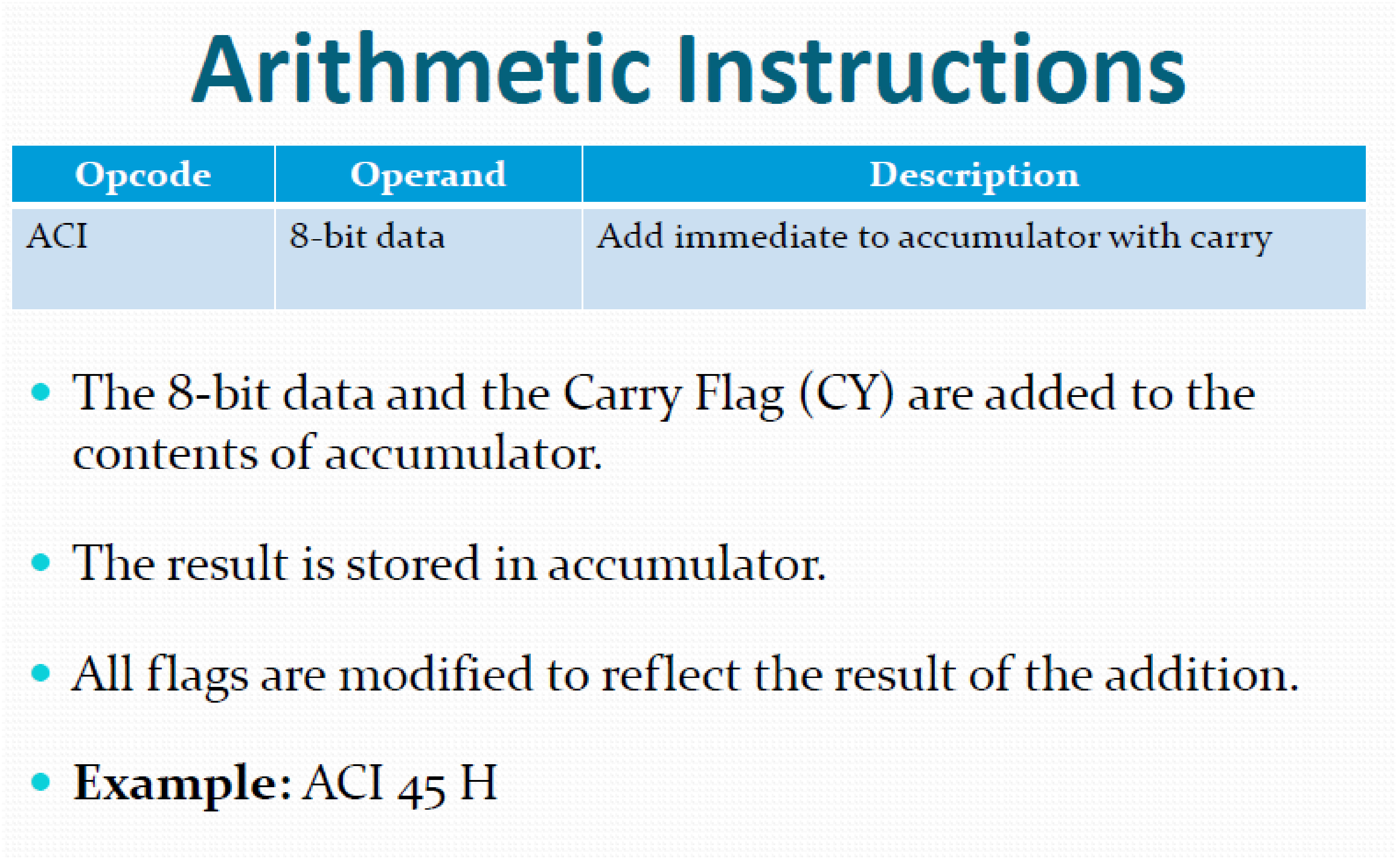
- The contents of register or memory and Carry Flag (CY) are added to the contents of accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of the addition.
- **Example:** ADC B or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



Arithmetic Instructions

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

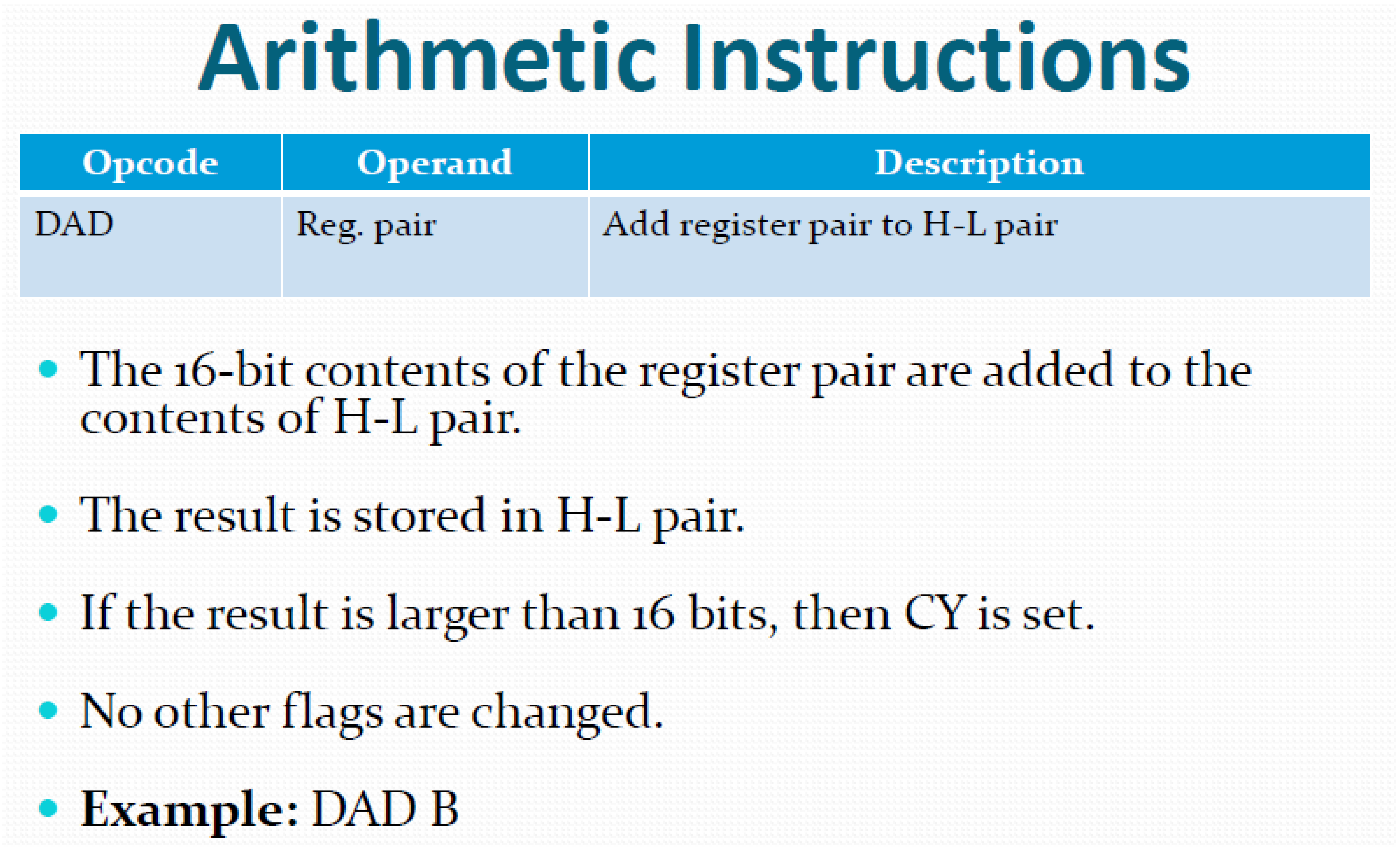
- The 8-bit data is added to the contents of accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of the addition.
- **Example:** ADI 45 H



Arithmetic Instructions

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

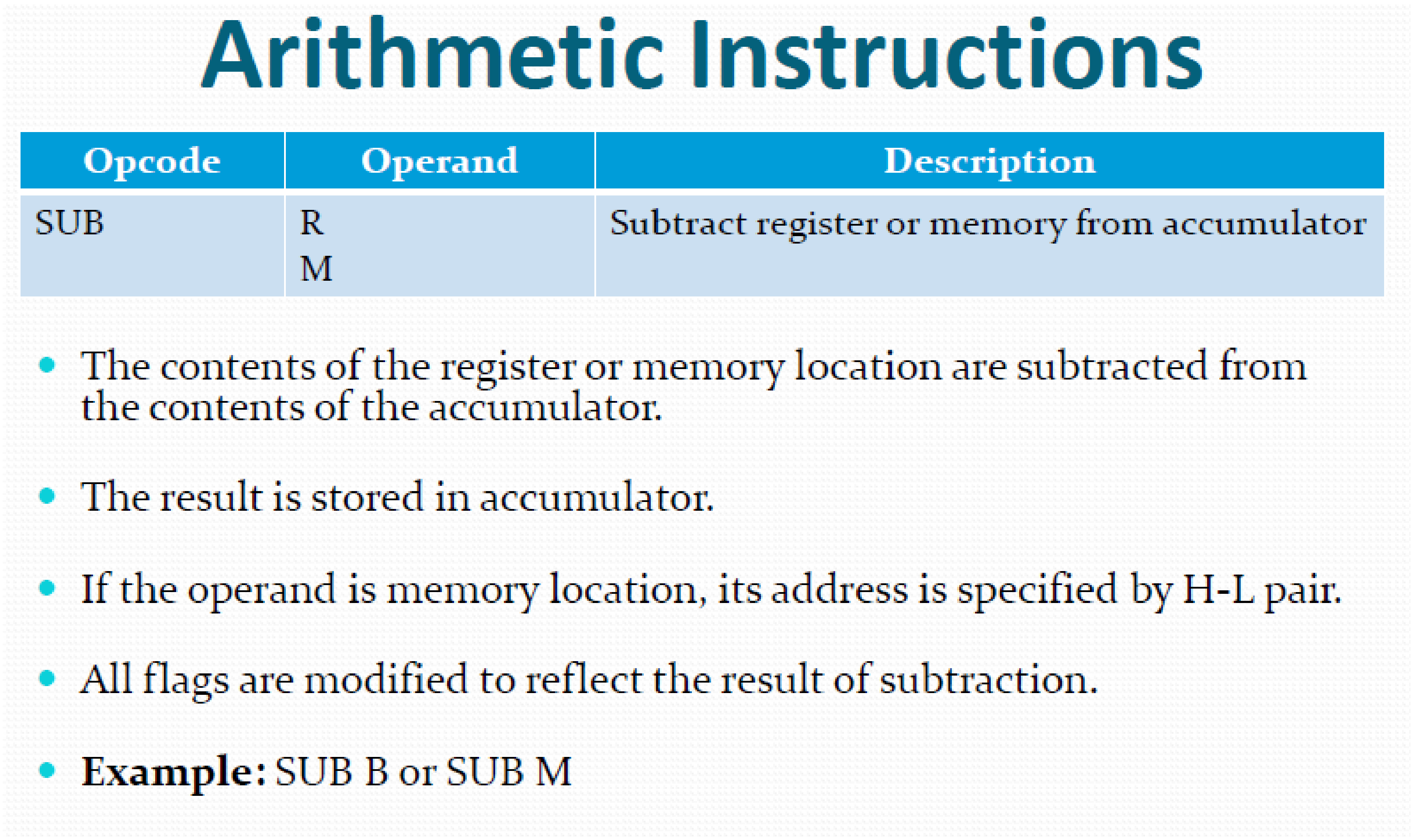
- The 8-bit data and the Carry Flag (CY) are added to the contents of accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of the addition.
- **Example:** ACI 45 H



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



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 or SUB M

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:** SBB B or SBB M

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 45 H

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

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 or INR 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

Arithmetic Instructions

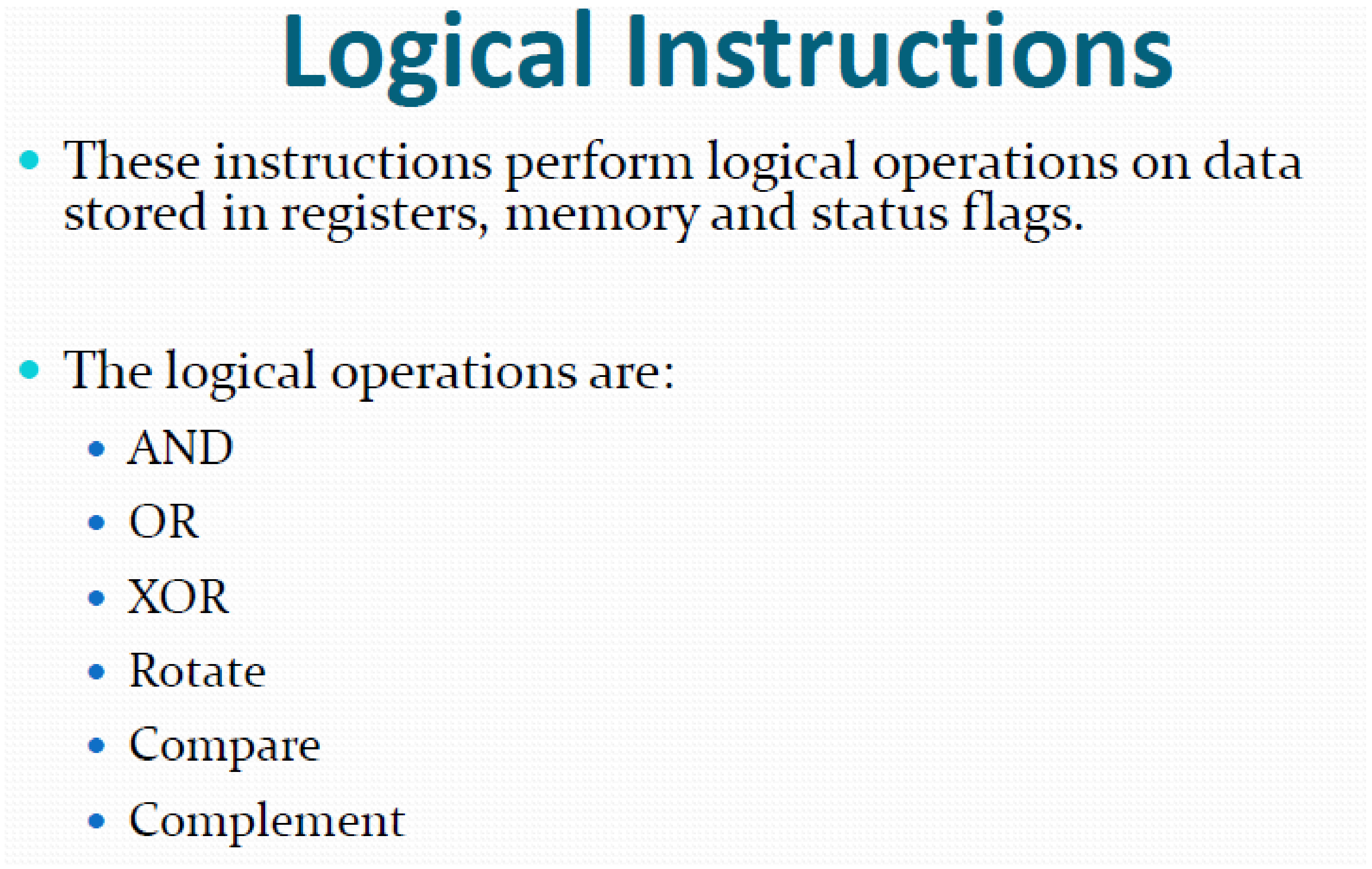
Opcode	Operand	Description
DCR	R M	Decrement register or memory by 1

- 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 or DCR M

Arithmetic Instructions

Opcode	Operand	Description
DCX	R	Decrement register pair by 1

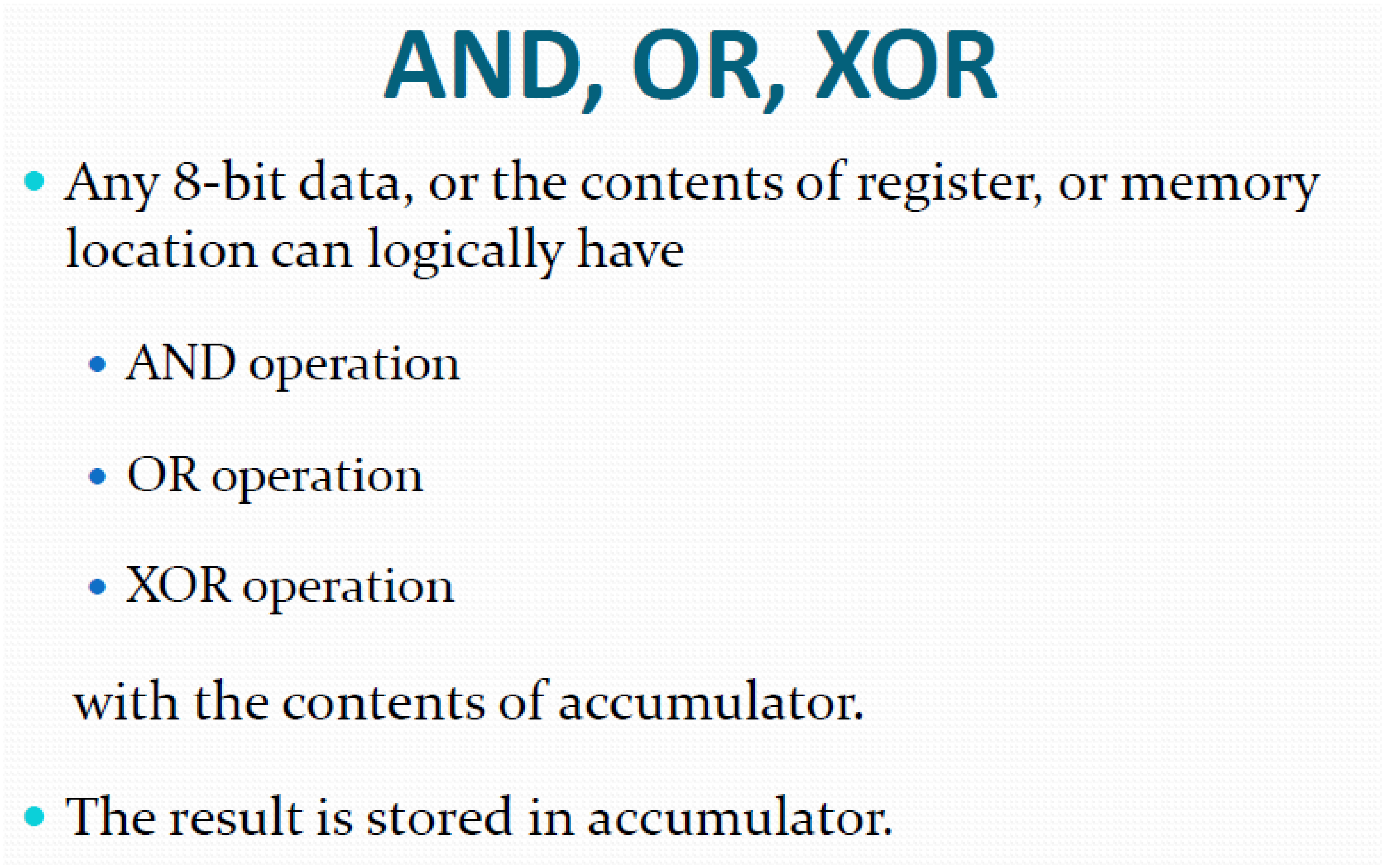
- The contents of register pair are decremented by 1.
- The result is stored in the same place.
- **Example:** DCX H



Logical Instructions

- These instructions perform logical operations on data stored in registers, memory and status flags.

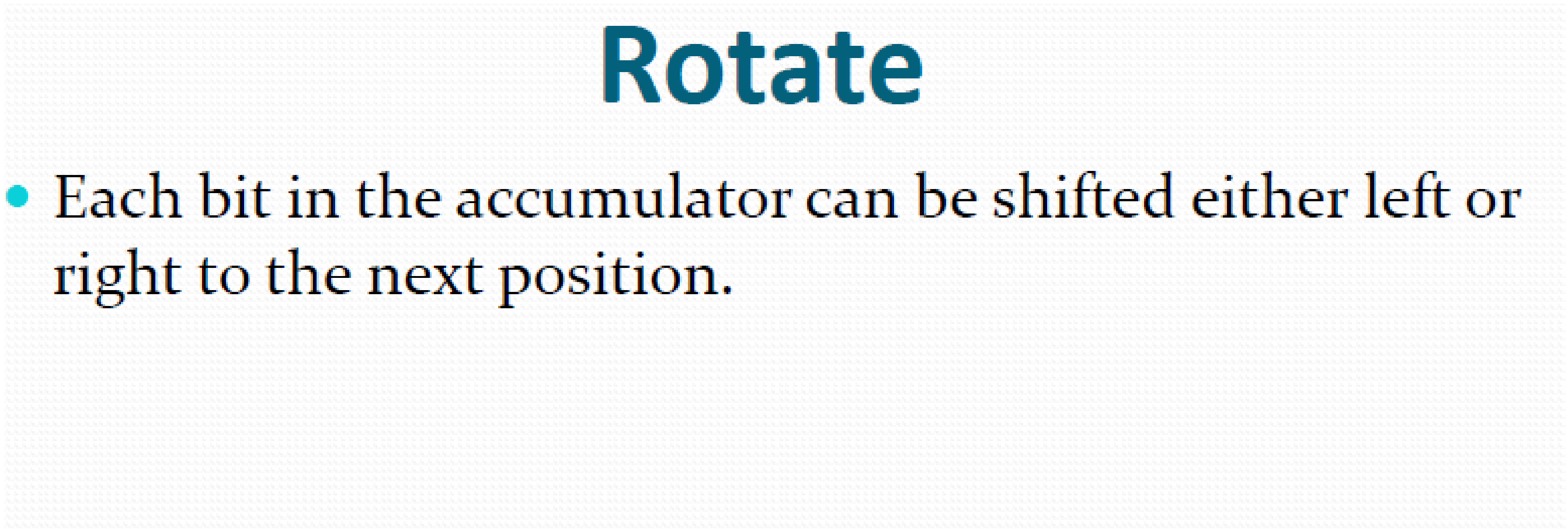
- The logical operations are:
  - AND
  - OR
  - XOR
  - Rotate
  - Compare
  - Complement



AND, OR, XOR

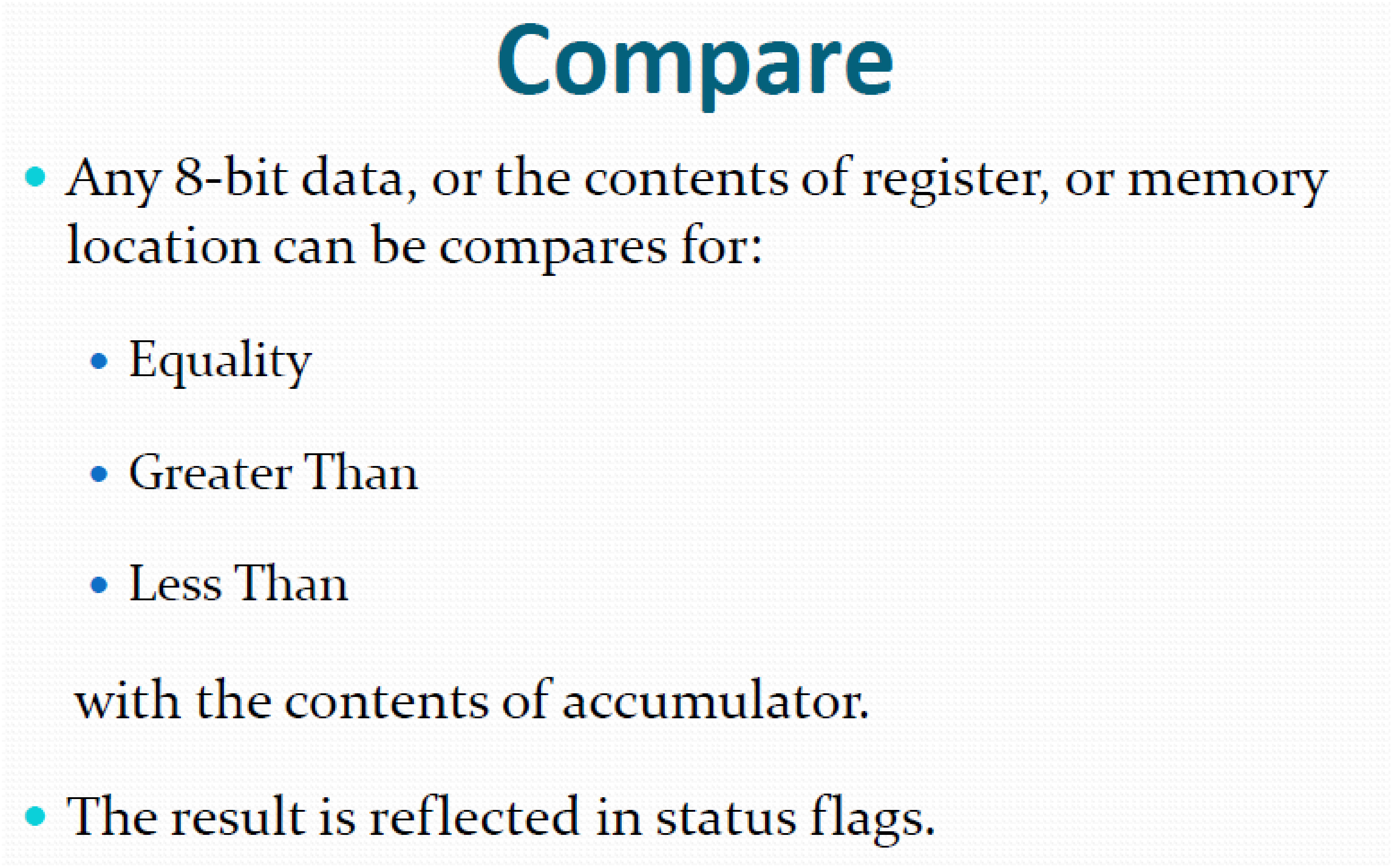
- Any 8-bit data, or the contents of register, or memory location can logically have
  - AND operation
  - OR operation
  - XOR operation

  with the contents of accumulator.
- The result is stored in accumulator.



Rotate

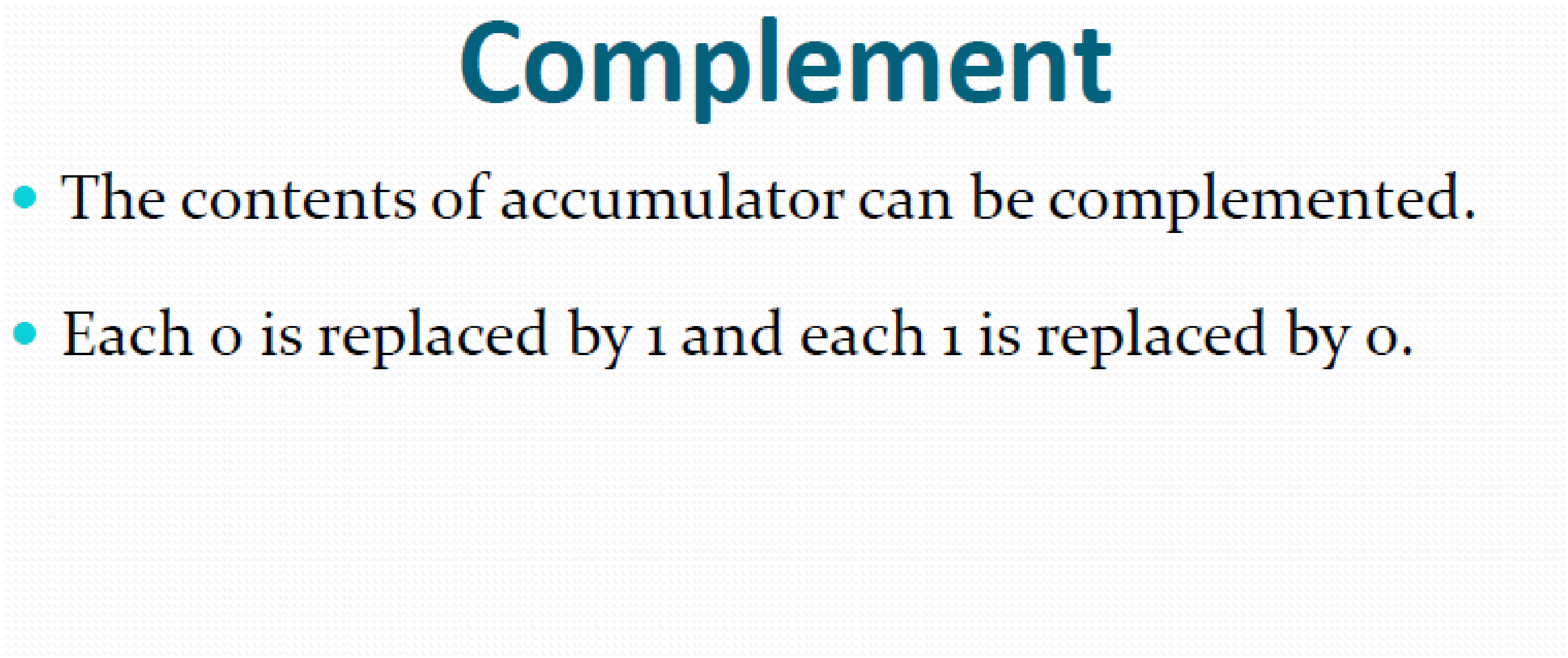
- Each bit in the accumulator can be shifted either left or right to the next position.



Compare

- Any 8-bit data, or the contents of register, or memory location can be compares for:
  - Equality
  - Greater Than
  - Less Than

  with the contents of accumulator.
- The result is reflected in status flags.



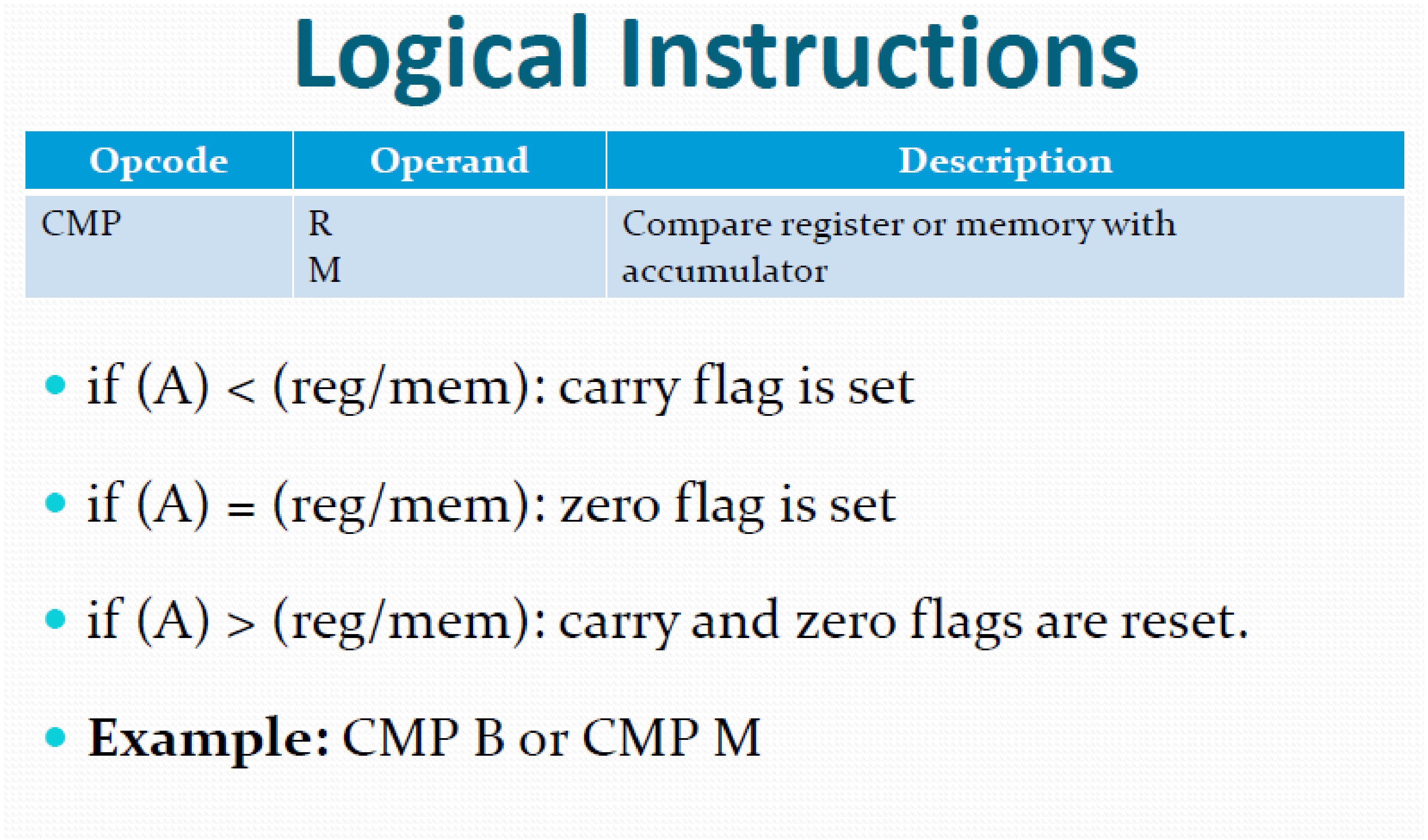
Complement

- 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
CMP	R M	Compare register or memory with accumulator

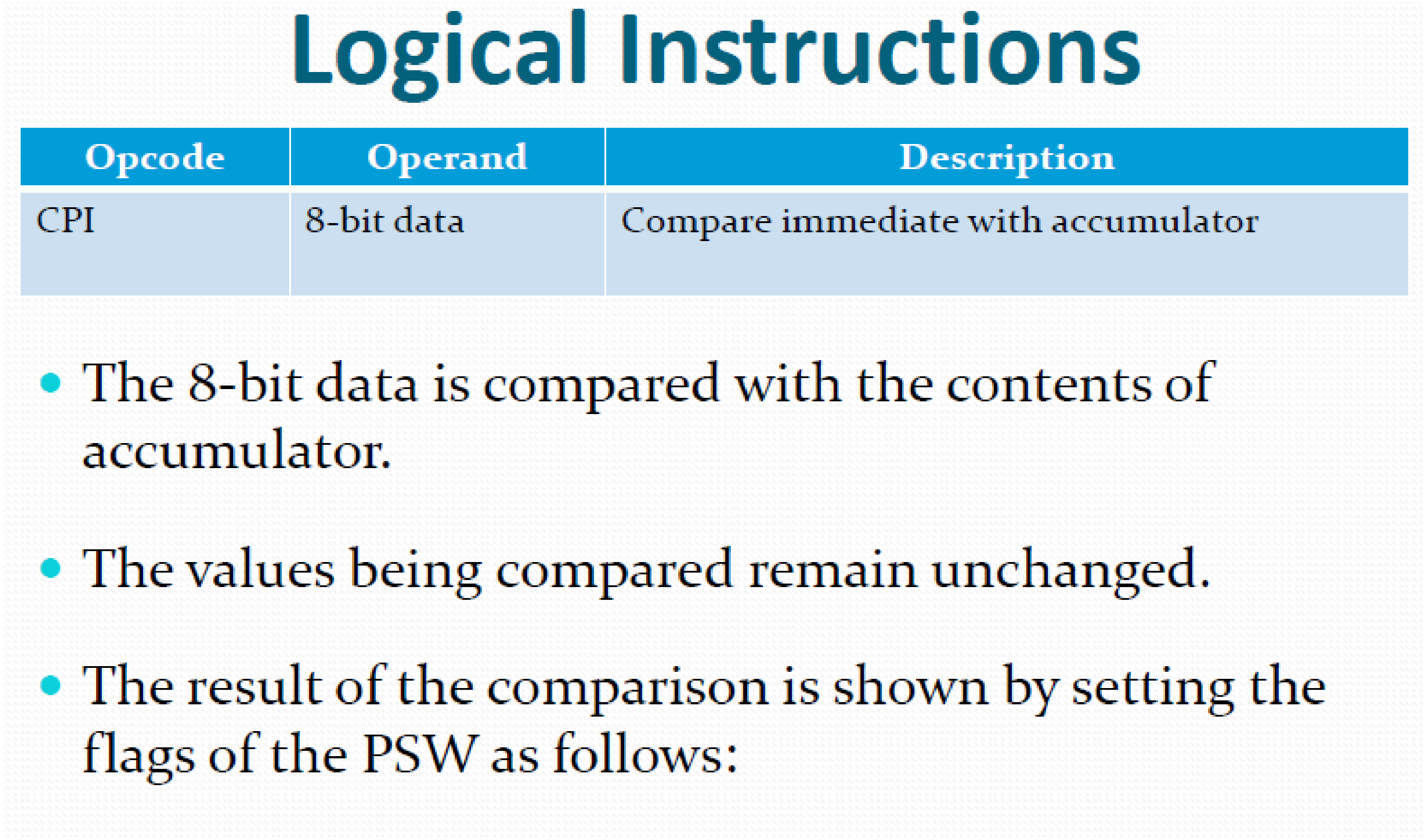
- The contents of the operand (register or memory) are compared with the contents of the accumulator.
- Both contents are preserved .
- The result of the comparison is shown by setting the flags of the PSW as follows:



Logical Instructions

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

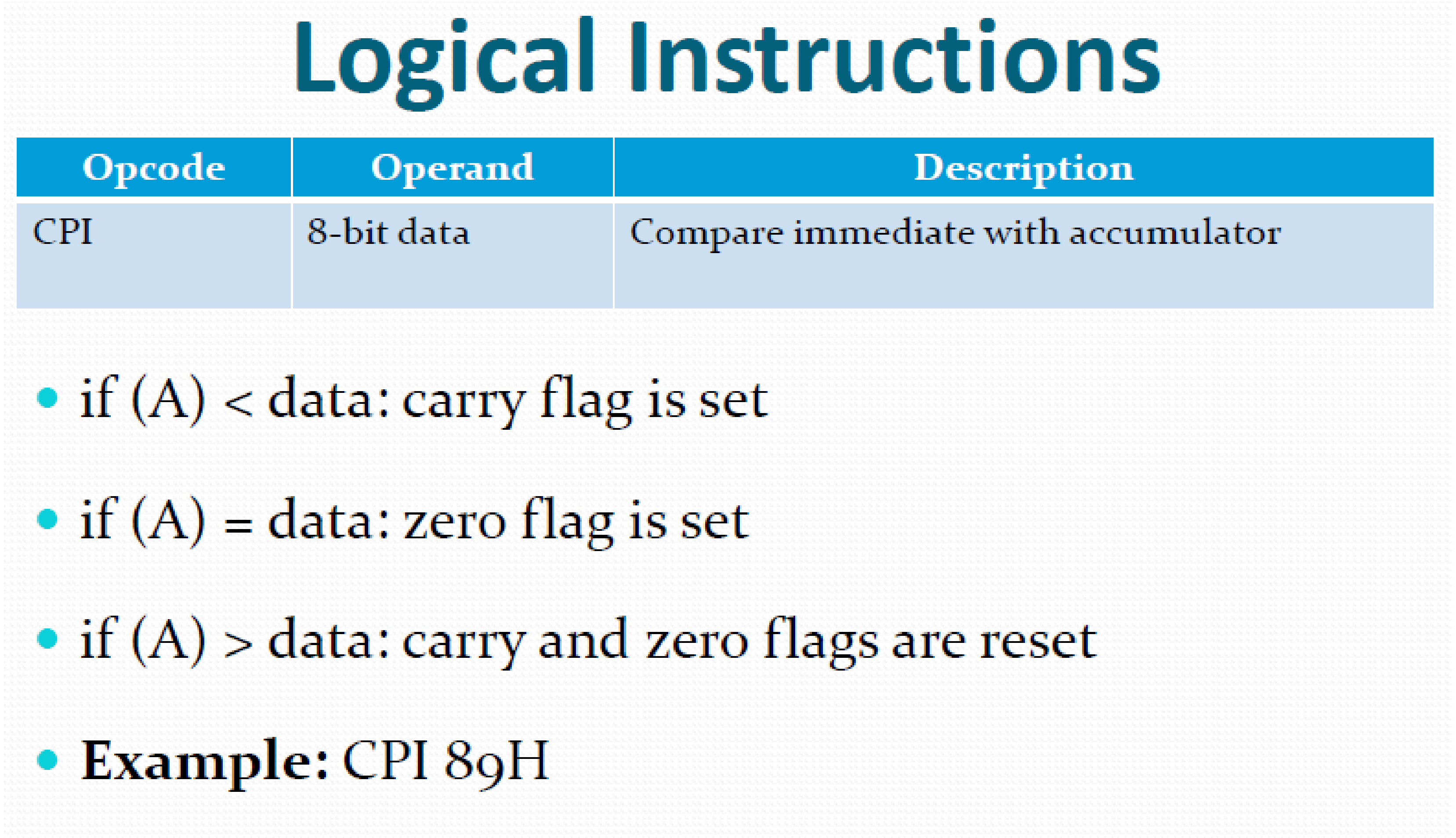
- if (A) < (reg/mem): carry flag is set
- if (A) = (reg/mem): zero flag is set
- if (A) > (reg/mem): carry and zero flags are reset.
- **Example:** CMP B or CMP M



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

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

- if (A) < data: carry flag is set
- if (A) = data: zero flag is set
- if (A) > data: carry and zero flags are reset
- **Example:** CPI 89H

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 and 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 B 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 86H.

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 B 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 86H.

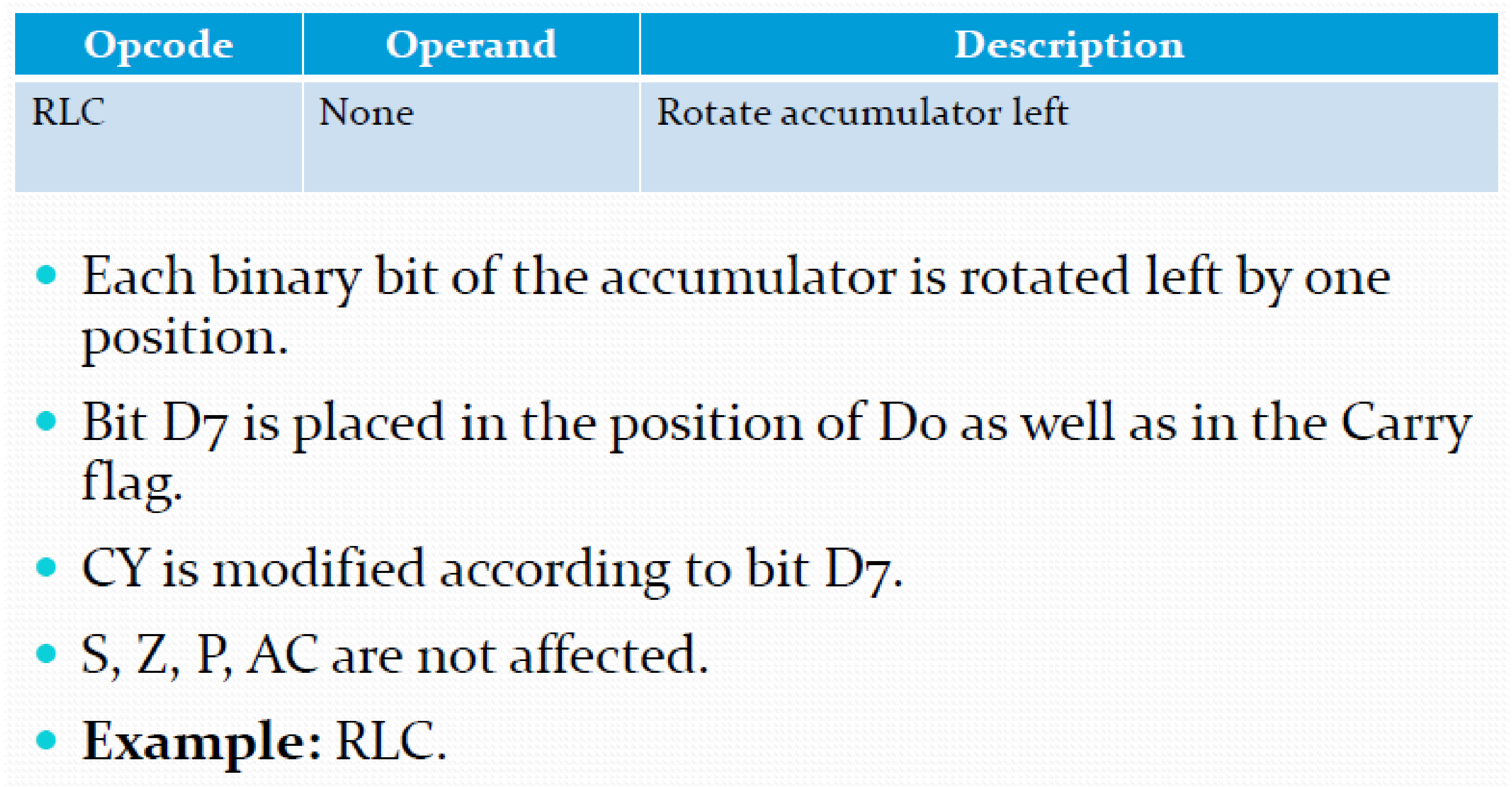
Opcode	Operand	Description
RAL	None	Rotate accumulator left through carry

- 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 Do.
- CY is modified according to bit D7.
- S, Z, P, AC are not affected.
- **Example:** RAL.

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 Do is placed in the Carry flag, and the Carry flag is placed in the most significant position D7.
- CY is modified according to bit Do.
- S, Z, P, AC are not affected.
- **Example:** RAR.

circular Left shift



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.

- circular right shift

Opcode	Operand	Description
RRC	None	Rotate accumulator right

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

Logical Instructions

Opcode	Operand	Description
CMA	None	Complement accumulator

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

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.

Branching Instructions

- The branching instruction alter the normal sequential flow.

- These instructions alter either unconditionally or conditionally.

Branching Instructions

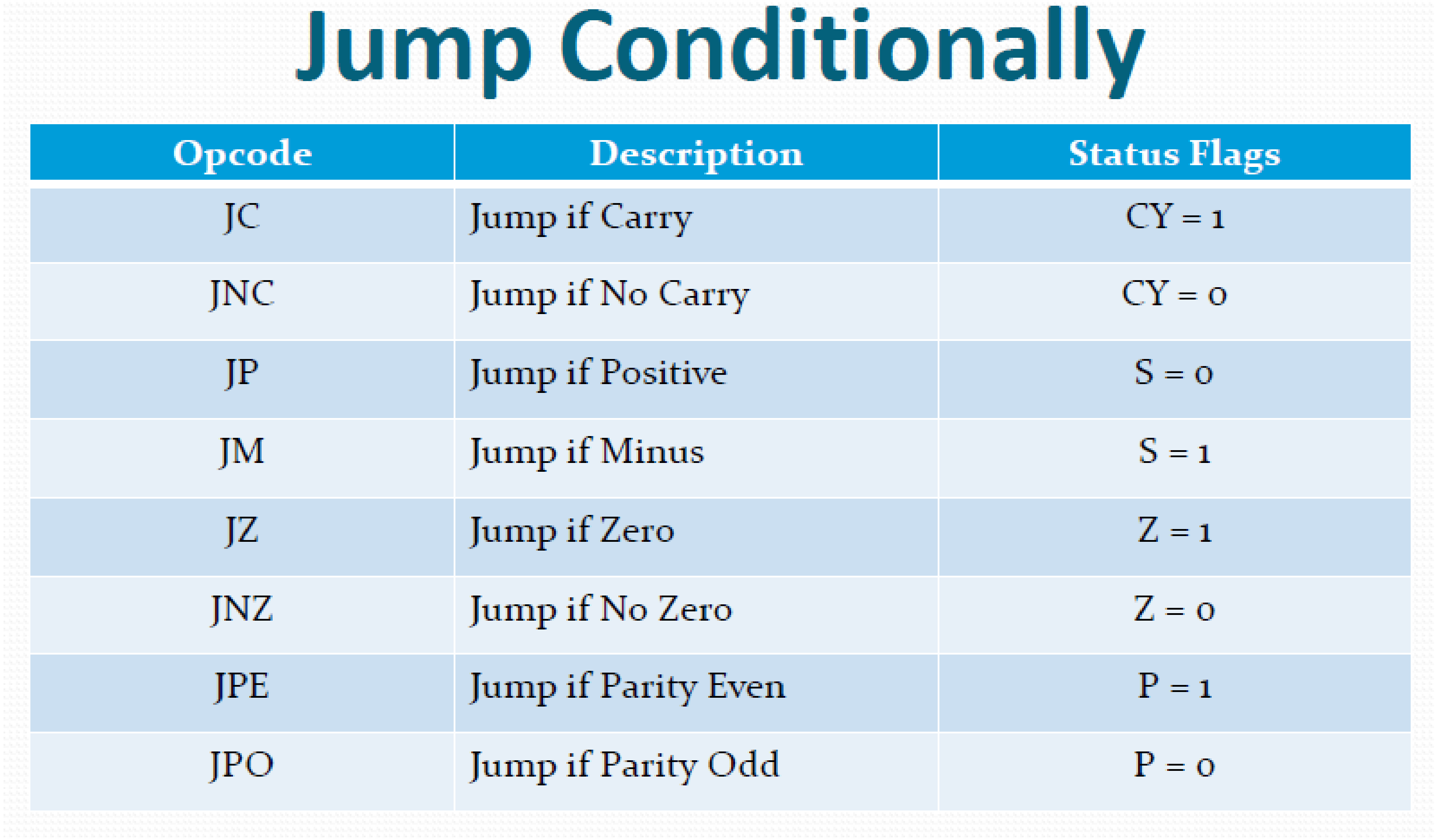
Opcode	Operand	Description
JMP	16-bit address	Jump unconditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
- **Example:** JMP 2034 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.



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

Branching Instructions

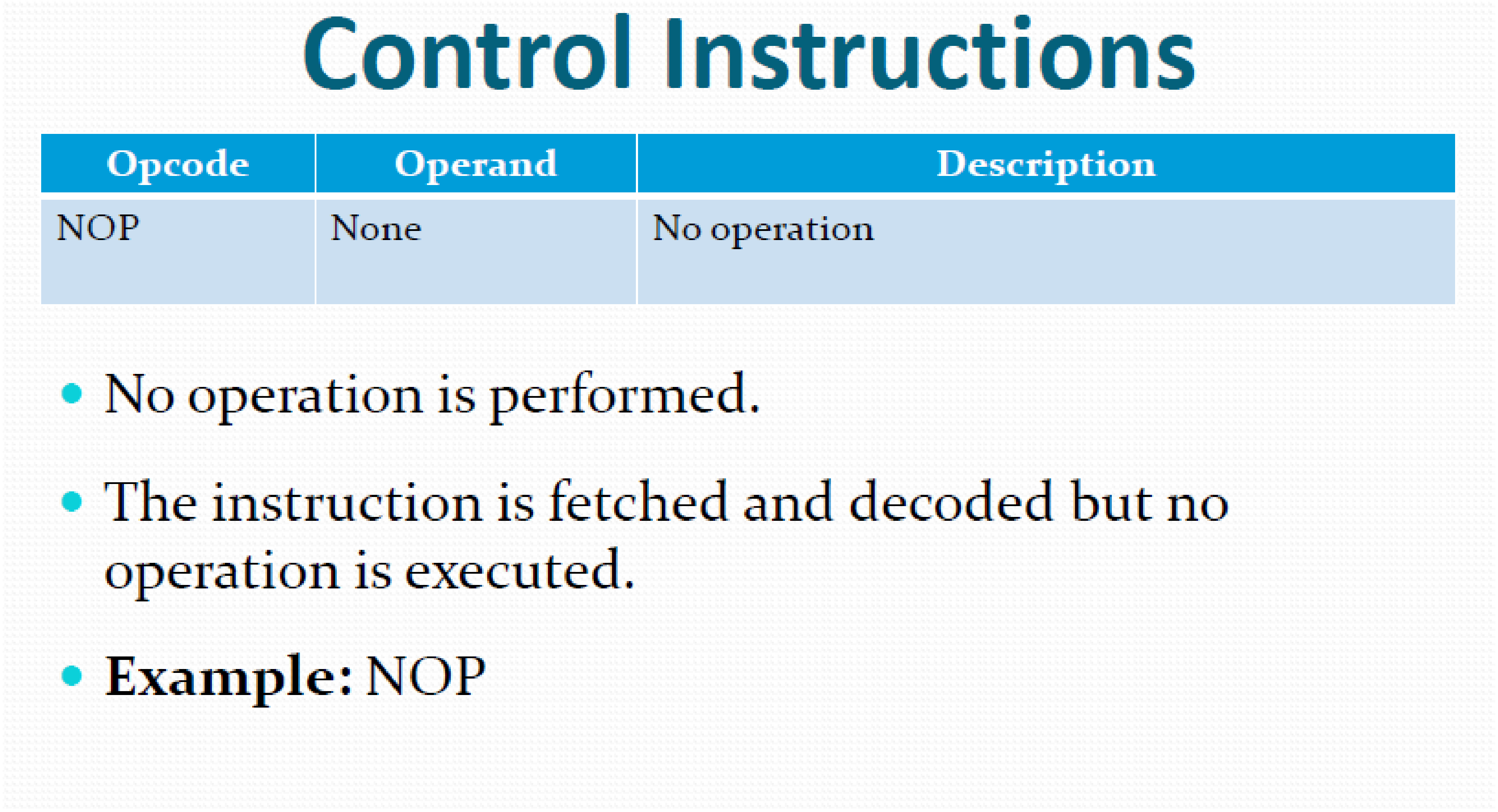
Opcode	Operand	Description
CALL	16-bit address	Call unconditionally

- 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.

Branching Instructions

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.



Control Instructions

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 Instructions

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 Instructions

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 Instructions

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

Summary – Data transfer

MOV	Move
MVI	Move Immediate
LDA	Load Accumulator Directly from Memory
STA	Store Accumulator Directly in Memory
LHLD	Load H & L Registers Directly from Memory
SHLD	Store H & L Registers Directly in Memory

Summary Data transfer

- An 'X' in the name of a data transfer instruction implies that it deals with a register pair (16-bits);

LXI	Load Register Pair with Immediate data
LDAX	Load Accumulator from Address in Register Pair
STAX	Store Accumulator in Address in Register Pair
XCHG	Exchange H & L with D & E
XTHL	Exchange Top of Stack with H & L

Summary - Arithmetic Group

- Add, Subtract, Increment / Decrement data in registers or memory.

ADD	Add to Accumulator
ADI	Add Immediate Data to Accumulator
ADC	Add to Accumulator Using Carry Flag
ACI	Add Immediate data to Accumulator Using Carry
SUB	Subtract from Accumulator
SUI	Subtract Immediate Data from Accumulator
SBB	Subtract from Accumulator Using Borrow (Carry) Flag
SBI	Subtract Immediate from Accumulator Using Borrow (Carry) Flag
INR	Increment Specified Byte by One
DCR	Decrement Specified Byte by One
INX	Increment Register Pair by One
DCX	Decrement Register Pair by One
DAD	Double Register Add; Add Content of Register Pair to H & L Register Pair

Summary Logical Group

- This group performs logical (Boolean) operations on data in registers and memory and on condition flags.
- These instructions enable you to set specific bits in the accumulator ON or OFF.

ANA	Logical AND with Accumulator
ANI	Logical AND with Accumulator Using Immediate Data
ORA	Logical OR with Accumulator
OR	Logical OR with Accumulator Using Immediate Data
XRA	Exclusive Logical OR with Accumulator
XRI	Exclusive OR Using Immediate Data

- The Compare instructions compare the content of an 8-bit value with the contents of the accumulator;

CMP	Compare
CPI	Compare Using Immediate Data

- The rotate instructions shift the contents of the accumulator one bit position to the left or right:

RLC	Rotate Accumulator Left
RRC	Rotate Accumulator Right
RAL	Rotate Left Through Carry
RAR	Rotate Right Through Carry

- Complement and carry flag instructions:

CMA	Complement Accumulator
CMC	Complement Carry Flag
STC	Set Carry Flag

Summary - Branch Group

- Unconditional branching

JMP	Jump
CALL	Call
RET	Return

- Conditions

NZ	Not Zero (Z = 0)
Z	Zero (Z = 1)
NC	No Carry (C = 0)
C	Carry (C = 1)
PO	Parity Odd (P = 0)
PE	Parity Even (P = 1)
P	Plus (S = 0)
M	Minus (S = 1)

- Conditional branching

Summary - Stack

PUSH	Push Two bytes of Data onto the Stack
POP	Pop Two Bytes of Data off the Stack
XTHL	Exchange Top of Stack with H & L
SPHL	Move content of H & L to Stack Pointer

I/0 instructions

IN	Initiate Input Operation
OUT	Initiate Output Operation

Summary -Machine Control instructions

EI	Enable Interrupt System
DI	Disable Interrupt System
HLT	Halt
NOP	No Operation

Addressing Modes

→ The different ways that a microprocessor can access data are referred to as Addressing modes

- Immediate Addressing mode
- Register Addressing mode
- Direct Addressing mode
- Indirect Addressing mode
- Implied Addressing mode

Immediate Addressing Mode

8 or 16 bit data can be specified as a part of Instruction

The instruction having ‘I’ (Immediate) letter fall under this category

Examples :

```
MVI C,25H
MVI M,7CH
LXI D,245EH
ADI 87H
```

Register Addressing Mode

Data transfer between Registers

Specifies the Source ,Destination or Both Operand in an 8085 registers

Faster Execution (it is not necessary to access memory locations)

Examples :

```
MOV A,B
ADD E
SPHL
XCHG
INR L
```

Direct Addressing Mode

Specifies 16 bit address of the operand within instruction itself

Second & third bytes of instruction contain 16 bit

Note : In interface IO port address is only 8 bit

Examples : STA 6000 H
LDA 2000H
LHLD 1111H
IN 75H

Indirect Addressing Mode

The memory address where the operand located is specified by the contents of a register pair

Examples :	STAX D	
	LDAX B	
	MOV M,D	
Immediate Indirect	MVI M,55H	
Register Indirect	ADC M	[A← A +Cy+(M)]
	DCR M	[(HL)←(HL)+1]
	PUSH PSW	

Implied Addressing Mode

Opcode specifies the address of the operands

Examples :

CMA (A $\leftarrow$ A)

STC (Cy $\leftarrow$ 1)

RAL

DAA

Instruction Classification according to word size or byte size

8085 instruction set can be classified into three categories on the basis of how many bytes are required to store the instruction in the memory.

1. One-byte instruction
2. Two-byte instruction
3. Three-byte instruction

One-byte Instructions:

Address	Hex code	Mnemonic	Comment
8000H	78H	MOV A, B	Copy the content of B register into Accumulator

Two-byte Instructions:

Address	Hex code	Mnemonic	Comment
8000H 8001H	0EH 05H	MVI C,05H	Load the C register with immediate data 05H

Three-byte Instructions:

Address	Hex code	Mnemonic	Comment
8000H	21H	LXI H,9000H	Load the HL register with 9000H
8001H	00H		
8002H	90H		

Addition of Two 8 Bit Numbers : (Indirect method)

Statement: Add the contents of memory location 6001H into the memory location 6000H and place the result in memory location 6002H.

Memory Address	Label	Mnemonics	Comments
7000	START:	LXI H, 6000H	; HL point's memory location 6000H
7003		MOV A, M	; Get first operand from location 6000H
7004		INX H	; HL point's 6001H
7005		ADD M	; Add second operand in first
7006		INX H	; HL points 6002H
7007		MOV M, A	; Store result at 6002H
7008		HLT	;Terminate program execution

Result:-

Memory Location	Before Execution	After Execution
6000H	08H	08H
6001H	04H	04H
6002H	XXH	0CH

Addition of Two 8 Bit Numbers : (Direct method)

Statement: Add the contents of memory location 6001H into the memory location 6000H and place the result in memory location 6002H.

Memory Address	Label	Mnemonics	Comments
7000	START:	LDA 6000H	; Load ACC from memory location 6000H
7003		MOV B, A	; move first operand from ACC
7004		LDA 6001H	; Load ACC from memory location 6001H
7007		ADD B	; Add second operand in first
7008		STA 6002H	; store the result at 6002H
700B		HLT	;Terminate program execution

Result:-

Memory Location	Before Execution	After Execution
6000H	08H	08H
6001H	04H	04H
6002H	XXH	0CH

Addition of Two 8 Bit Numbers : (Direct method)

Statement: Add the given two numbers and place the result in memory location 6002H.

Memory Address	Label	Mnemonics	Comments
7000	START:	MVI A ,08H	; get first operand into ACC
7002		MVI B, 04H	; get second operand into reg. B
7004		ADD B	; Add second operand in first
7005		STA 6002H	; store the result at 6002H
7008		HLT	;Terminate program execution

Result:-

Memory Location	Before Execution	After Execution
6002H	XXH	0CH

Subtraction of Two 8 Bit Numbers :

Statement: Subtract the contents of memory location 6001H from the contents of memory location 6000H and place the result in memory location 6002H.

Memory Address	Label	Mnemonics	Comments
7000	START:	LXI H, 6000H	; HL point's memory location 6000H
7003		MOV A, M	; Get first operand from location 6000H
7004		INX H	; HL point's 6001H
7005		SUB M	; Subtract operand from first operand
7006		INX H	; HL points 6002H
7007		MOV M, A	; Store result at 6002H
7008		HLT	;Terminate program execution

Result:-

Memory Location	Before Execution	After Execution
6000H	0FH	0FH
6001H	04H	04H
6002H	XXH	0BH

Multiplication of Two 8 Bit Numbers

Statement: Multiply the contents of memory location 6001H by the contents of memory location 6000H and place the result in memory location 6002H.
Microprocessor 8085 has no direct multiplication instruction and the multiplication is done by using successive addition method. e.g 5x2=5+5
Here the number which is to be multiply is loaded in B and C register.
Clear the accumulator(A) and add the contents of C in A repeatedly until contents of B becomes Zero.(for every addition, B is decremented by 1).

Memory Address	Label	Mnemonics	Comments
7000	START:	LXI H, 6000H	; HL point's memory location 6000H
7003		MOV B, M	; Get first operand from location 6000H in B register
7004		INX H	; HL point's 6001H
7005		MOV C,M	; Get Second operand from location 6001H in C register
7006		MVI A, 00H	;Clear the contents of the accumulator(A)
7008	HERE:	ADD C	;Add the contents of reg .C in accumulator
7009		DCR B	;Decrement contents of B by 1
700A		JNZ HERE	; If not zero, repeat
700D		INX H	; HL points 6002H
700E		MOV M, A	; Store result at 6002H
700F		HLT	; Terminate program execution

Division of Two 8 Bit Numbers

Statement: Divide the contents of memory location 6001H by the contents of memory location 6000H and place the result with quotient in memory location 6002H and remainder in memory location 6003H. Microprocessor 8085 has no direct division instruction. So the division is done by using successive subtraction method. e.g. 5/2 then 5-2-2=1 (Reminder) two times subtraction means quotient is 2.
During each successive subtraction, we need to monitor the remainder should not be less than the quotient. if the remainder is less than the divisor then we need to stop the successive division. This condition is monitor by CMP B instruction. If accumulator data is larger than the data in register B then carry flag is reset and hence the successive division will continue. The number of repetitions will give the value of quotient and Accumulator holds the remainder.

Memory Address	Label	Mnemonics	Comments
7000	START:	LXI H, 6000H	; HL point's memory location 6000H
7003		MOV A, M	; Get first operand from location 6000H in A register
7004		INX H	; HL point's 6001H
7005		MOV B,M	; Get Second operand from location 6001H in B register
7006		MVI C,00H	;Clear the contains of the register C
7008	HERE:	SUB B	;Subtract contents of reg . B from A
7009		INR C	;Increment contents of C by 1
700A		CMP B	; Compare B with A
700B		JNC HERE	; If carry not generated then repeat
700E		INX H	; HL points 6002H
700F		MOV M, C	; Store result (quotient) at 6002H
7010		INX H	; HL points 6003H
7011		MOV M, A	; Store result (Remainder) at 6003H
7012		HLT	; Terminate program execution

Block Transfer

Statement: Write a program to transfer a block of 10 numbers from memory location 6000H into the 8000H.
Block transfer means to copy a block of data and store it a new location. To perform the block transfer operation two data pointers are required one is at source and second is at the destination. Here in the program HL pointer reads the data from the source and DE pointer stores the data at the destination. After coping a byte of data both pointers are incremented by one till all the data is transferred. Register C is configured as a counter for the data transfer which decrements with each data transfer.

Memory Address	Label	Mnemonics	Comments
7000	START:	MVI C, 09H	; Initialize counter
7002		LXI H, 6000H	; HL point's memory location 6000H
7005		LXI D, 8000H	; DE point's memory location 8000H
7008	BACK:	MOV A,M	;Get no. from memory location pointed by HL pair
7009		STAX D	;Store no. into memory location pointed DE pair
700A		INX H	; HL point's next memory location
700B		INX D	; DE point's next memory location
700C		DCR C	; Decrement counter by 1
700D		JNZ BACK	: If not zero, repeat
7010		HLT	; Terminate program execution

- **Long answer questions (8 Marks)**

1. Explain ADD, ADC B, DAD B, ADD M instruction with proper example.
2. Explain the different addressing modes with example.
3. Explain logical instructions of 8085.
4. Write an ALP(Assembly Language Program) to divide two 8-bit numbers. The numbers are stored 6000 and 6001H memory location. Store the result at 6002H memory location after division.
5. Write an ALP(Assembly Language Program) to multiply two8-bit numbers. The numbers are stored 6000 and 6001H memory location. Store the result at 6002H memory location after multiplication.

6. Write an ALP(Assembly Language Program) to transfer a block of data from 6000H and store it at 8000H. copy 10 number of bytes in sequential manner.

- **Short answer questions (4 marks)**

1. What is the difference between JMP and CALL instruction?
2. Explain the following instructions ANA B and ANIB.
3. Explain the CMP and CPI instruction with the flag status.
4. What is instruction? Explain the format of instruction
5. Write an ALP (Assembly Language Program) to add two 8 bit numbers. The numbers are stored 6000 and 6001H memory location. Store the result at 6002H memory location after addition.
6. Write an ALP (Assembly Language Program) to subtract two 8 bit numbers. The numbers are stored 6000 and 6001H memory location. Store the result at 6002H memory location after subtraction.

- **Select the most correct alternative from the following**

1. LDA 9000H is a ----------------byte instruction

- a) one b) two **c) three** d) four

2. To clear the accumulator following instruction can be used

- a) **SUB A** b) MVI A, 01H c) ANI 05H d) MOV A, B

3. If accumulator A= 08, after execution of ORI 05H accumulator will have ---------

- a) 13H **b) 0DH** c) 85H d) 58H

4. ---------instruction is always used at the end of the subroutine

- a) HLT b) NOP c) CALL **d) RET**

5. To call a subroutine unconditionally ----------------instruction is used

- **a) CALL** b) CZ c) JMP D) CNZ

- 6. Maskable interrupt can be disable ---------- using instruction
  - **a) DI** b) EI c)RST 1 d) RIM
- 7. SIM is ------------byte instruction
  - **a) one** b) two c) three d) four

8. 8085 assembly language consists of --------number of instructions

- a) 100b) 256**c) 74** D) 255

9. ---------- instruction can perform 16 bit addition

- **a) DAD**b) ADC c) ADDD) ADI

10. During a CALL instruction stack pointer is ------------

- a) Incremented **b) decremented**c) does not change D) at 0000H