8085 Instruction set & addressing mode

MICROPROCESSORS HISTORY

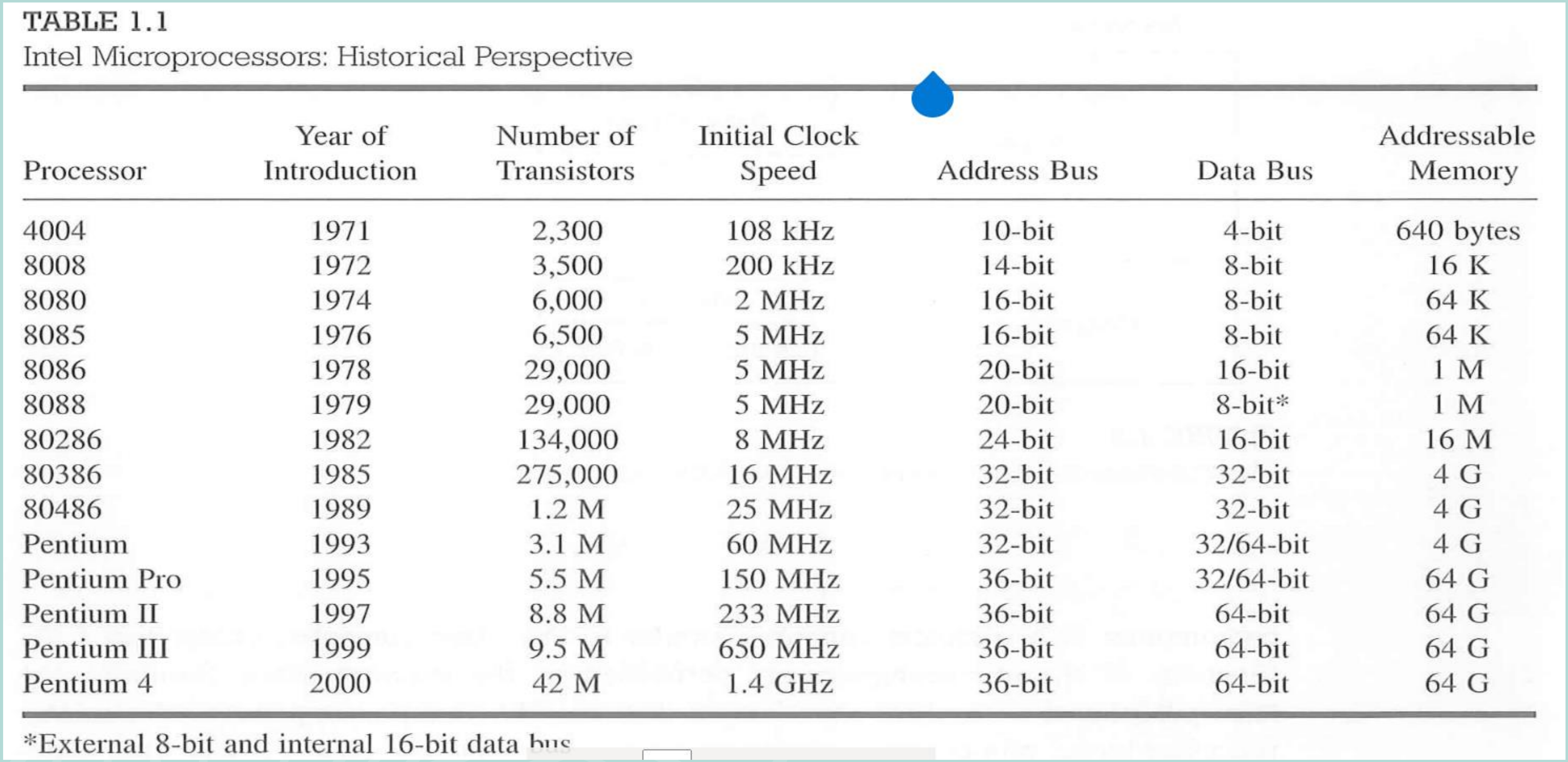


TABLE 1.1
Intel Microprocessors: Historical Perspective

Processor	Year of Introduction	Number of Transistors	Initial Clock Speed	Address Bus	Data Bus	Addressable Memory
4004	1971	2,300	108 kHz	10-bit	4-bit	640 bytes
8008	1972	3,500	200 kHz	14-bit	8-bit	16 K
8080	1974	6,000	2 MHz	16-bit	8-bit	64 K
8085	1976	6,500	5 MHz	16-bit	8-bit	64 K
8086	1978	29,000	5 MHz	20-bit	16-bit	1 M
8088	1979	29,000	5 MHz	20-bit	8-bit*	1 M
80286	1982	134,000	8 MHz	24-bit	16-bit	16 M
80386	1985	275,000	16 MHz	32-bit	32-bit	4 G
80486	1989	1.2 M	25 MHz	32-bit	32-bit	4 G
Pentium	1993	3.1 M	60 MHz	32-bit	32/64-bit	4 G
Pentium Pro	1995	5.5 M	150 MHz	36-bit	32/64-bit	64 G
Pentium II	1997	8.8 M	233 MHz	36-bit	64-bit	64 G
Pentium III	1999	9.5 M	650 MHz	36-bit	64-bit	64 G
Pentium 4	2000	42 M	1.4 GHz	36-bit	64-bit	64 G

*External 8-bit and internal 16-bit data bus

POINTS TO REMEMBER . . .

- Microprocessor uses buses for connecting different components & peripherals.
- There are three types of buses being used. They are 1. Data bus 2. Address bus and 3. Control bus.
- 8085 has 8 bit data lines and so why it is referred to be as a microprocessor.
- 8085 has 16 address lines.
- ALU (Arithmetical and Logical unit) is one of the most important components of a microprocessor which performs all arithmetical and logical operations.
- There are 6 general purpose registers:- **B,C,D,E,H & L**
- Accumulator, Program counter, Stack pointer are all referred to be as special function registers. They are all meant for a specific purpose.

CONTD.,

- Stack pointer is a register which points to the temporary storage area within the available memory.
- Program counter is a 16 bit register that will hold the address of the next instruction to be executed.
- 8085 has flag register which helps in getting the status of the arithmetic operations.
- The operating frequency of 8085 is 3 MHz.
- Interrupt and Polling are the options available for handling multiple processes.
- Interrupt is the best option comparing polling.
- TRAP carries a very high priority and it cannot be stopped by any command or any other mask. It is of the highest priority among all the other interrupts.

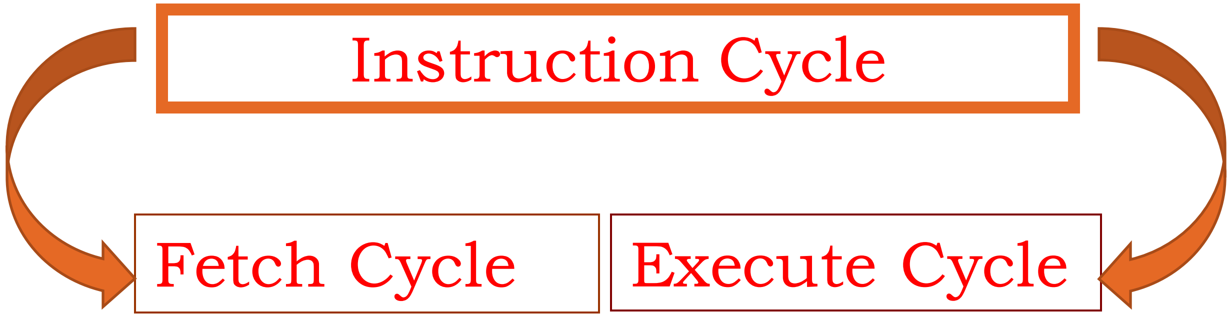
Processor Cycles

Instruction Cycle:

- The sequence of operations that a processor has to carry out while executing the instruction is called instruction cycle.

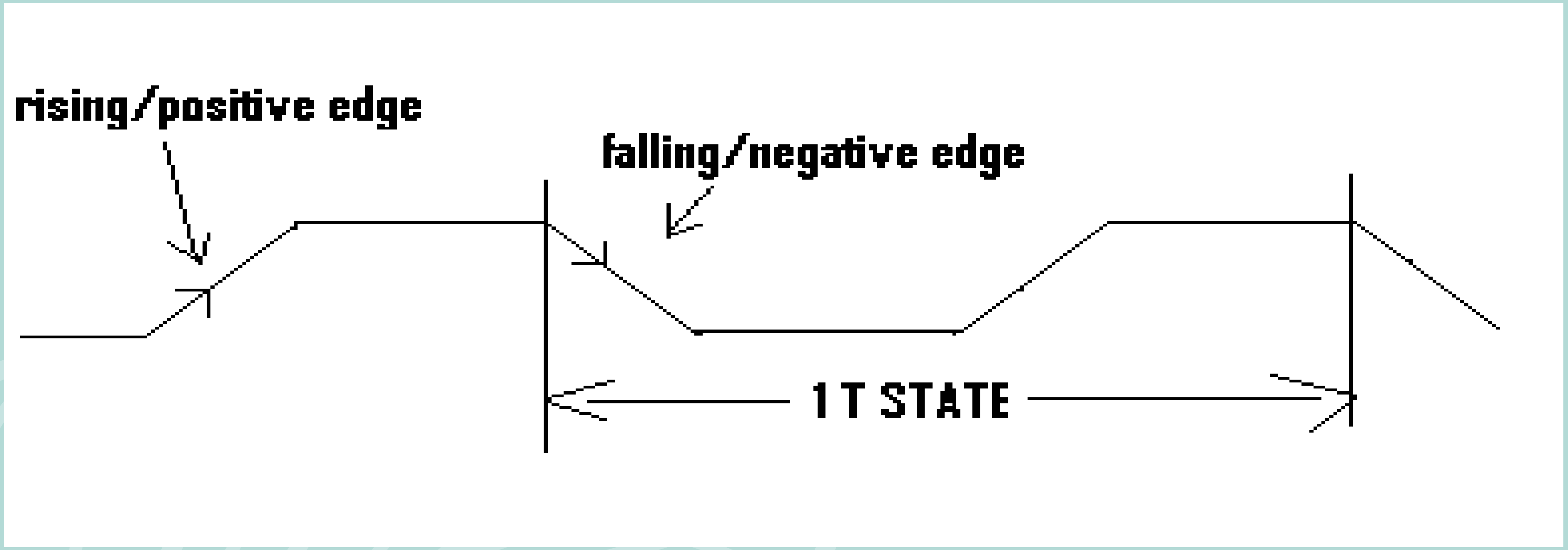
Machine Cycle:

- Each instruction cycle of a processor consist of a number of basic operations called machine cycles or processor cycles.



T States

- The time taken by the processor to execute a machine cycle is expressed in T - States.
- One T – state is equal to the time period of the internal clock signal of the processor.
- The T- state starts at the falling edge of a clock.



8085 – HOW AN INSTRUCTION WILL LOOK LIKE?

- **What is an instruction?**
  - It is a command that can make the microprocessor perform a specific function.
  - All instructions are collectively grouped as instruction set.
  - An Instruction is basically a combination of two parts one is **operation code** and the next is **operands**.

❑Operation code is also called as opcode.

- ❑ The opcode will be used to identify what kind of operation needs to be performed.
- ❑ Operands can be one or two based on the instruction.
- ❑ Operands will be used to spot the source and destination. An operand can be any of the following:

1. Registers (like Accumulator, B register and so on)
2. Immediate value (Say 10H, 010101B)
3. An address, i.e. memory location. (Say 30H)
4. Any of the supported input or output ports.

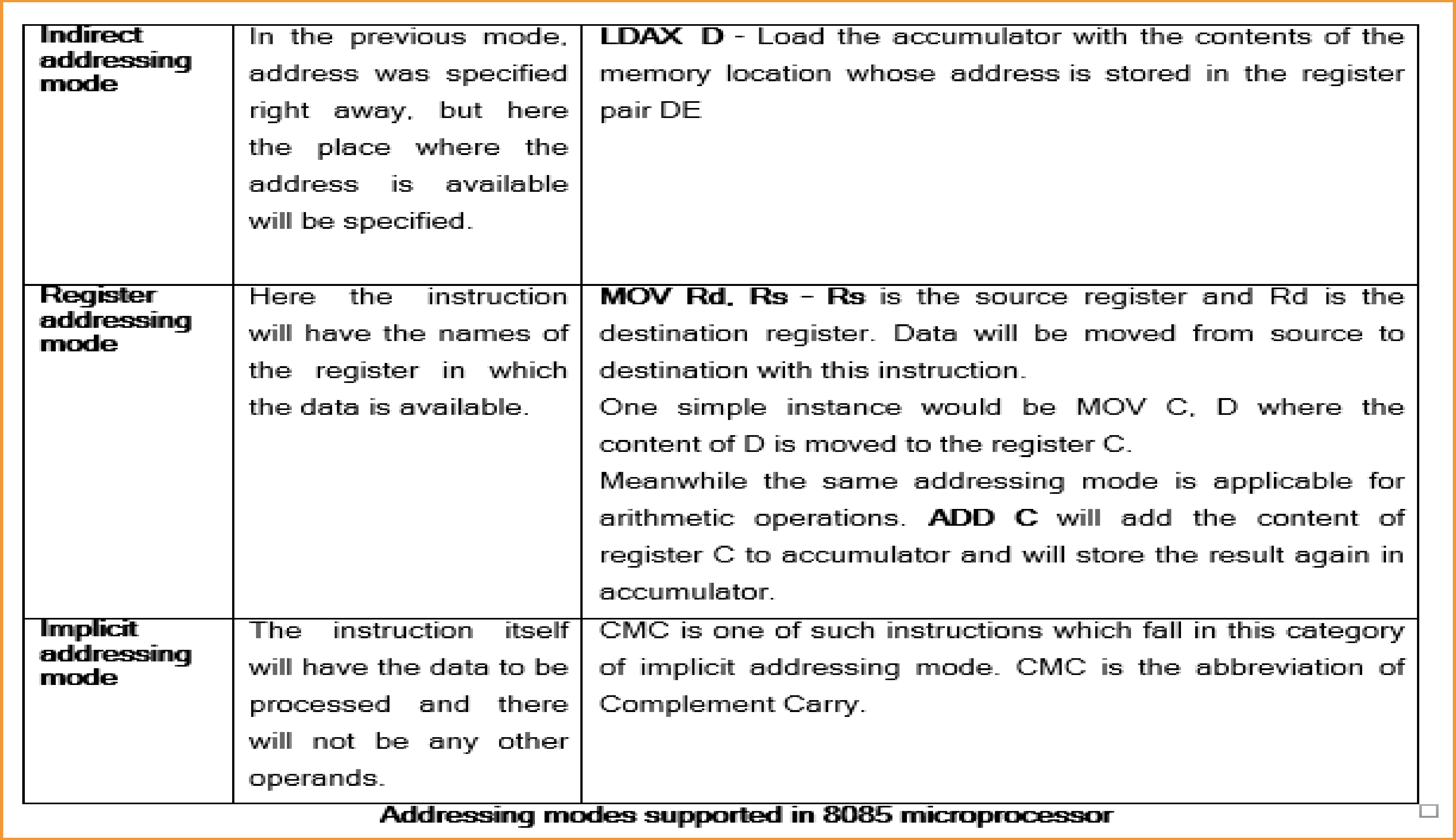
ADDRESSING MODES

8085 supports five types of addressing:

The addressing modes are

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

Mode	Description	Example
Immediate addressing mode	Simplest mode, where the instruction itself will have the information to be moved or to be processed. The data to be processed can be moved to any register or memory location.	<ul><li>**MVI A,05H** - 05H is the data to be moved to Accumulator with MVI instruction which is abbreviation of Move Immediate. Similarly immediate addressing mode can be used in arithmetic operations as well. Simple example would be:</li><li>ADI 01H, where 1 is added to the content already available in the accumulator.</li></ul>
Direct addressing mode	In this mode one of the operands will be the address itself where the information is available	<ul><li>**LDA 2000H** – Where, 2000H is the address of the source and LDA is the instruction which is expanded as load accumulator. The contents available in the memory location 2000H will be stored in accumulator once this instruction is executed.</li><li>**STA 2000H** – This is yet another instruction which falls in the category of direct addressing more. This instruction prompts the microprocessor to store the content of the accumulator into the specified address location 2000H.</li></ul>



Indirect addressing mode	In the previous mode, address was specified right away, but here the place where the address is available will be specified.	**LDAX D** - Load the accumulator with the contents of the memory location whose address is stored in the register pair DE
Register addressing mode	Here the instruction will have the names of the register in which the data is available.	**MOV Rd, Rs – Rs** is the source register and Rd is the destination register. Data will be moved from source to destination with this instruction. One simple instance would be MOV C, D where the content of D is moved to the register C. Meanwhile the same addressing mode is applicable for arithmetic operations. **ADD C** will add the content of register C to accumulator and will store the result again in accumulator.
Implicit addressing mode	The instruction itself will have the data to be processed and there will not be any other operands.	CMC is one of such instructions which fall in this category of implicit addressing mode. CMC is the abbreviation of Complement Carry.

Addressing modes supported in 8085 microprocessor

8085 – INSTRUCTION SET.

246 instructions are available in 8085.

The instruction set can be classified as follows:

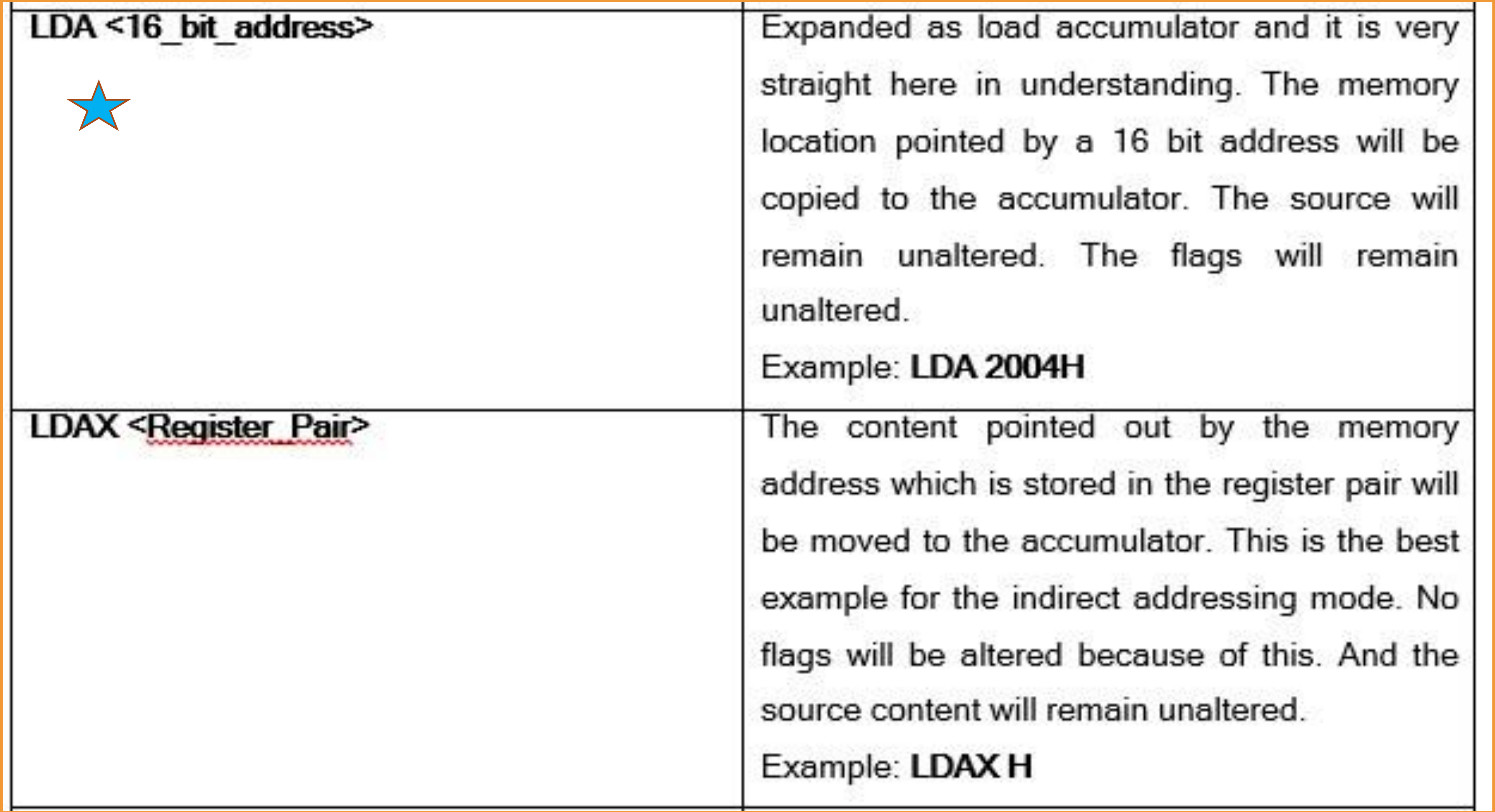
1. Data transfer instructions
2. Arithmetic instructions
3. Logical instructions
4. Branching instructions and
5. Control instructions.

1. Data transfer instructions – it is used for moving data from source to destination. The content of source will not be modified after the data transfer is complete. It will remain the same.

MOV Rd, Rs **MOV Rd, M** **MOV M, Rs**	This instruction copies the contents of the source register into the destination register and as mentioned already, it would not alter the content of the destination register. There are three instructions supported for this category: 1. From a register to register move. 2. From memory location to a register move. 3. From a register to memory. Point to note is the memory address will be hold or pointed by the HL register pair. Example: **MOV B,C** **MOV B,M** **MOV M,C** And another thing to note is no flags will be affected due to these instructions.

MVI Rd, Data **MVI M, Data** ★	This is an immediate data operation where the data is moved to the destination register mentioned. Example: **MVI Rd, 76H** Also the immediate data can be moved to a memory location specified by HL registers. Example: **MVI M, 76H** No flags will be affected due to the usage of these instructions.
LXI <Register_Pair> , <16 _bit_data> ★	Immediate addressing mode can be realized through this instruction. The 16 bit data is moved to the register pair mentioned as one of the operands. Example: **LXI H, 2545H**

POP Register Pair	It is exactly opposite to PUSH. It will extract the top two memory spots from the stack and will put it on to the register pair mentioned. Example : **POP B**
OUT <8 bit address>	If the programmer wishes to export the data in the accumulator to be moved to the external world, it can be done only through ports. This can be accomplished through OUT instruction which will output the content of the accumulator to the mentioned port address. Example: **OUT 54H**
IN <8 bit address>	This is simply opposite to the previous command. Here the data can be input from the port to the accumulator. Example: **IN 40H**



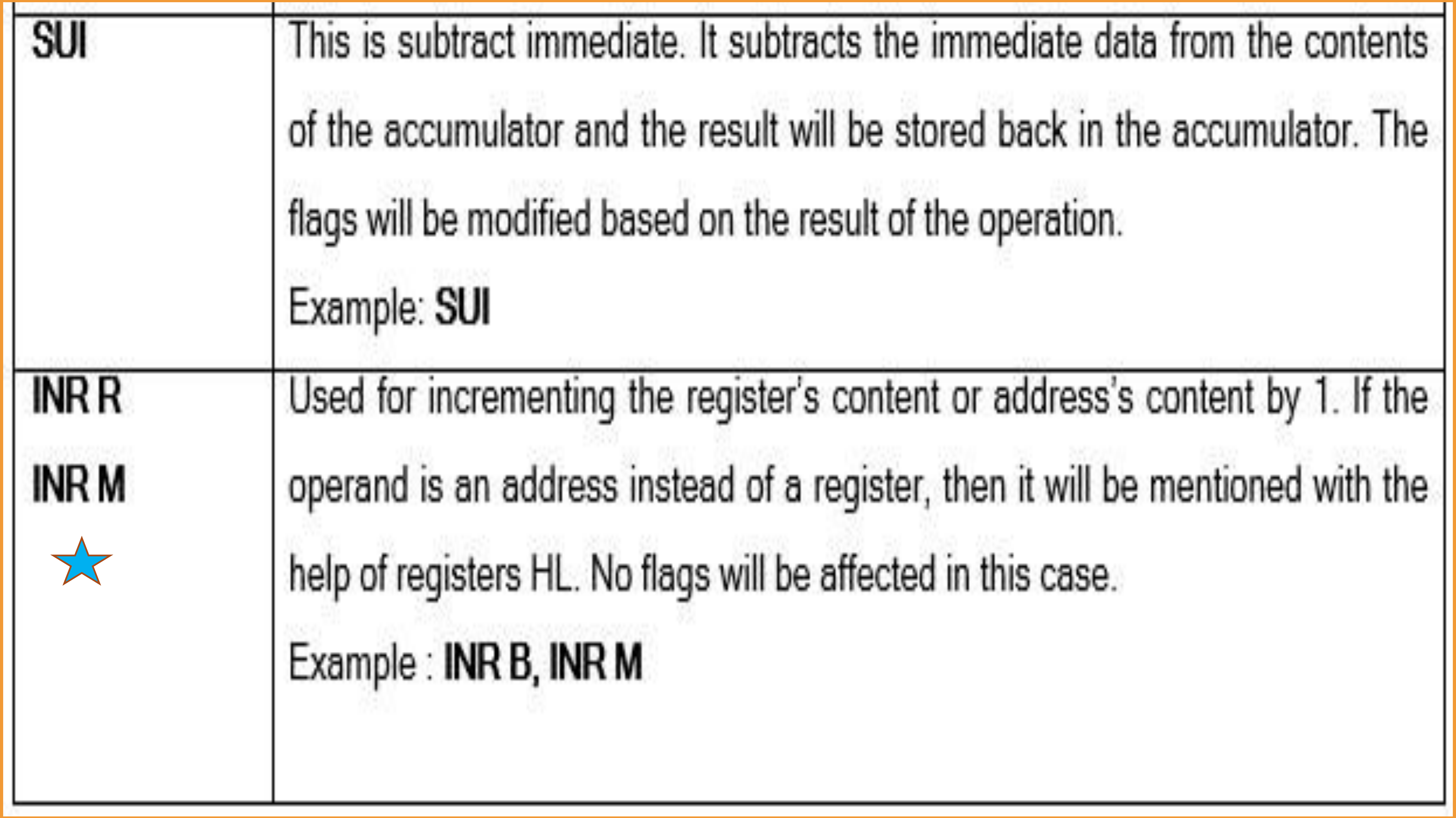
LDA <16_bit_address>	Expanded as load accumulator and it is very straight here in understanding. The memory location pointed by a 16 bit address will be copied to the accumulator. The source will remain unaltered. The flags will remain unaltered. Example: **LDA 2004H**
LDAX <Register_Pair>	The content pointed out by the memory address which is stored in the register pair will be moved to the accumulator. This is the best example for the indirect addressing mode. No flags will be altered because of this. And the source content will remain unaltered. Example: **LDAX H**

STAX <Register_Pair>	This will store the contents of the accumulator to the address pointed by the register pair and this is an awesome example for the reader to understand the indirect addressing mode. There will not be any alteration in the source i.e. the accumulator. Example: **STAX H**
LHLD <16_bit_address>	Instruction is expanded as Load H and L. This is one of the classical examples for direct addressing mode. The contents of the memory location pointed by 16 bit address will copied into the L and the next memory location's content will be copied to H register. The content of the source will not be altered and no flags will be affected as an impact of this instruction. Example: **LHLD 2000H**
SHLD <16_bit_address>	This is exactly in reverse to the previous instruction. Here the contents of the H and L registers will be stored in the 2 consecutive memory locations as pointed by the 16_bit_address. . Source remains unaltered and the flags are so as well. Example: **SHLD 2000H**

SPHL	Used to copy the content of H and L register pair into the stack pointer. Contents of the source remain unaltered. Example: **SPHL**
XCHG	Expanded as exchange. This instruction will help in exchanging the contents of HL with DE register pair. No flags will be affected. Example: **XCHG**
XTHL	This is also an instruction related to exchanging the data. But, this time the content of HL register pair will be exchanged with the contents of top of the stack. Example: **XTHL**

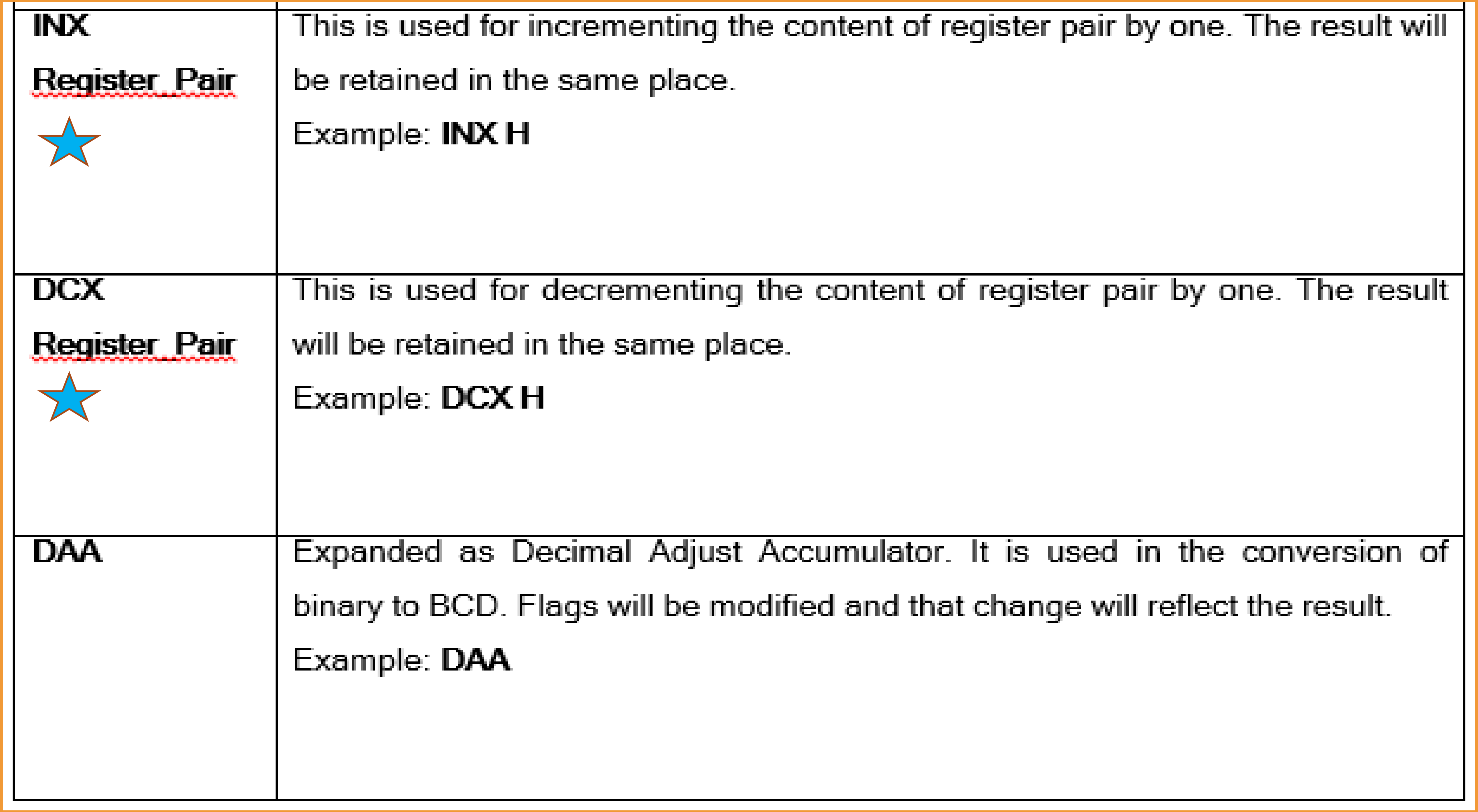
2. ARITHMETIC INSTRUCTIONS

ADD R **ADD M**	The contents from the register or memory locations will be added to the content of the accumulator and the result will be stored back in the accumulator. Also the flags will be affected and status will be reflected in the flags. Add with carry options are also available and can be used for cases where carry operations are required. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. Example : **ADD B** **ADD M**
ADI	Expanded as Add Immediate. It will add the immediate data mentioned as operand with the content of the accumulator. All the flags could get affected and might be modified based on the execution results. Example: **ADI 54H**
SUB R **SUB M**	The contents from the register or memory locations will be subtracted from the contents of the accumulator, and the result will be stored back in the accumulator. Also the flags will be affected and status will be reflected in the flags. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. Example : **SUB B, SUB M**

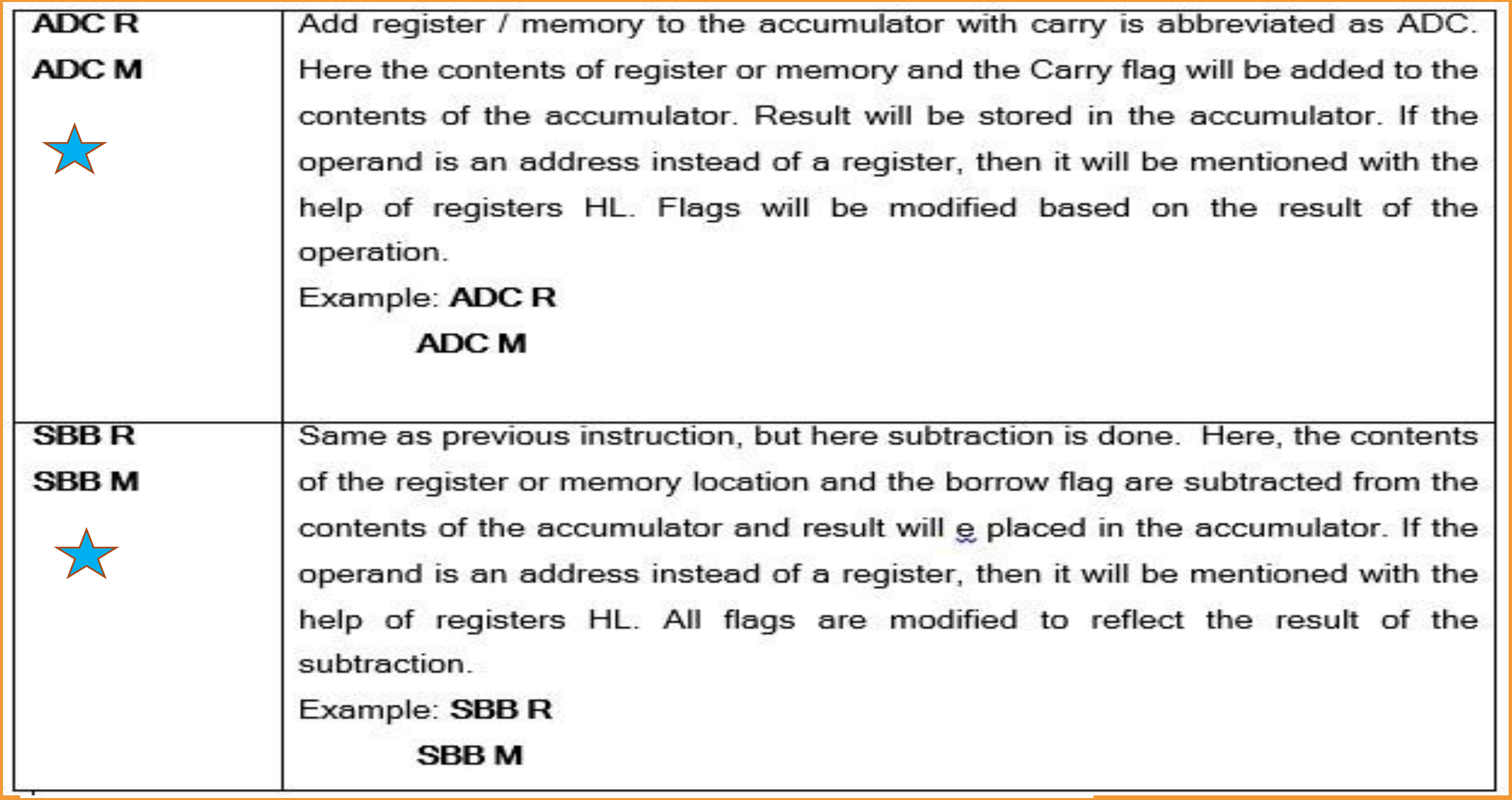


SUI	This is subtract immediate. It subtracts the immediate data from the contents of the accumulator and the result will be stored back in the accumulator. The flags will be modified based on the result of the operation. Example: **SUI**
INR R **INR M**	Used for incrementing the register's content or address's content by 1. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. No flags will be affected in this case. Example : **INR B, INR M**

DCR M **DCR R**	Used for decrementing the content of the address or register by 1. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. No flags will be affected in this case. Example : **DCR B, DCR M**
ACI	Expanded as add immediate to the accumulator with carry. The 8 bit data and the Carry flag will be added to the contents of the accumulator. Then the result will be stored back in the accumulator. The flags will be changed based on the result of the operation. Example: **ACI 51H**
SBI	Expanded as subtract immediate from the accumulator with borrow. The 8 bit data and the borrow flag will be subtracted from the contents of accumulator and the result is stored back in the accumulator. The flags will be changed based on the result of the operation. Example: **SBI 51H**
DAD **Register_Pair**	This is used for 16 bit addition. Contents of the register pair specified will be added to the content of HL register pair and result will be stored in the HL register. Based on the result the CY flag might be altered. Example: **DAD B**



INX Register_Pair ★	This is used for incrementing the content of register pair by one. The result will be retained in the same place. Example: **INX H**
DCX Register_Pair ★	This is used for decrementing the content of register pair by one. The result will be retained in the same place. Example: **DCX H**
DAA	Expanded as Decimal Adjust Accumulator. It is used in the conversion of binary to BCD. Flags will be modified and that change will reflect the result. Example: **DAA**



ADC R **ADC M** ⭐	Add register / memory to the accumulator with carry is abbreviated as ADC. Here the contents of register or memory and the Carry flag will be added to the contents of the accumulator. Result will be stored in the accumulator. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. Flags will be modified based on the result of the operation. Example: **ADC R** **ADC M**
SBB R **SBB M** ⭐	Same as previous instruction, but here subtraction is done. Here, the contents of the register or memory location and the borrow flag are subtracted from the contents of the accumulator and result will e placed in the accumulator. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. All flags are modified to reflect the result of the subtraction. Example: **SBB R** **SBB M**

3. LOGICAL INSTRUCTIONS

- ❑ Points to be noted before learning/reading on logical instructions are summarized as follows:
- ❑ One of the operands must be stored in the accumulator without which it would not be possible to use the logical instructions.
- ❑ Any addressing mode can be used with the logical instructions. There is no restriction in this regard.
- ❑ All the instructions that perform logical operations are categorized under LOGICAL instruction set.

CMP R **CMP M**	Expanded as Compare. The contents of the register or the memory location (R or M) will be compared with the contents of the accumulator. Both the contents of register and accumulator will be retained after the operation. <u>Example:</u> **CMP C** The result of the operation compare can be known through the flags. Following are the cases: ◆ if (A) < (register/memory): carry flag is set ◆ if (A) = (register/memory): zero flag is set ◆ if (A) > (register/memory): carry and zero flags are reset
CPI – 8 bit	This instruction is expanded as Compare Immediate. It compares the immediate operand (8 bit data) with the content of the accumulator. The contents of accumulator and 8 bit data which is being compared will remain unchanged. <u>Example:</u> **CPI 90H** The result of the operation compare can be known through the flags. Following are the cases: ◆ if (A) < data: carry flag is set ◆ if (A) = data: zero flag is set ◆ if (A) > data: carry and zero flags are reset

ANA R **ANA M**	This will logically AND the content of the register or the memory location / register with the content of accumulator and result will be stored in accumulator. Flags will be affected based on the result. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. Example: **ANA B** The flags S, Z, P in the flag registers will be modified based on the result of the operation. CY is reset. AC is set.
ANI	Expanded as AND Immediate. It will logically AND the contents with the contents in the accumulator. Result will be stored in the accumulator. Example: **ANI 01H** Flags S,Z and P will be modified based on the result of the operation. Also CY flag is reset. AC flag is set.
CMA	CMA is expanded as Complement Accumulator. It will complement the contents of the accumulator and result will be stored back in accumulator. No flags will be affected because of this instruction. Example: **CMA**
CMC	CMC is expanded as Complement Carry flag. It will not affect any other flags. Example: **CMC**

STC	Set Carry is abbreviated as STC. It sets the Carry flag and it will not have impact on any other flags. Example: **STC**
XRA A **XRA M**	This will Exclusively OR (EXOR) the content of the register or the memory location / register with the content of accumulator and result will be stored in accumulator. Flags will be affected based on the result. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. Flags S,Z and P will be modified based on the result of the operation. CY and AC flags will be reset. Example: **XRA B** **XRA M**

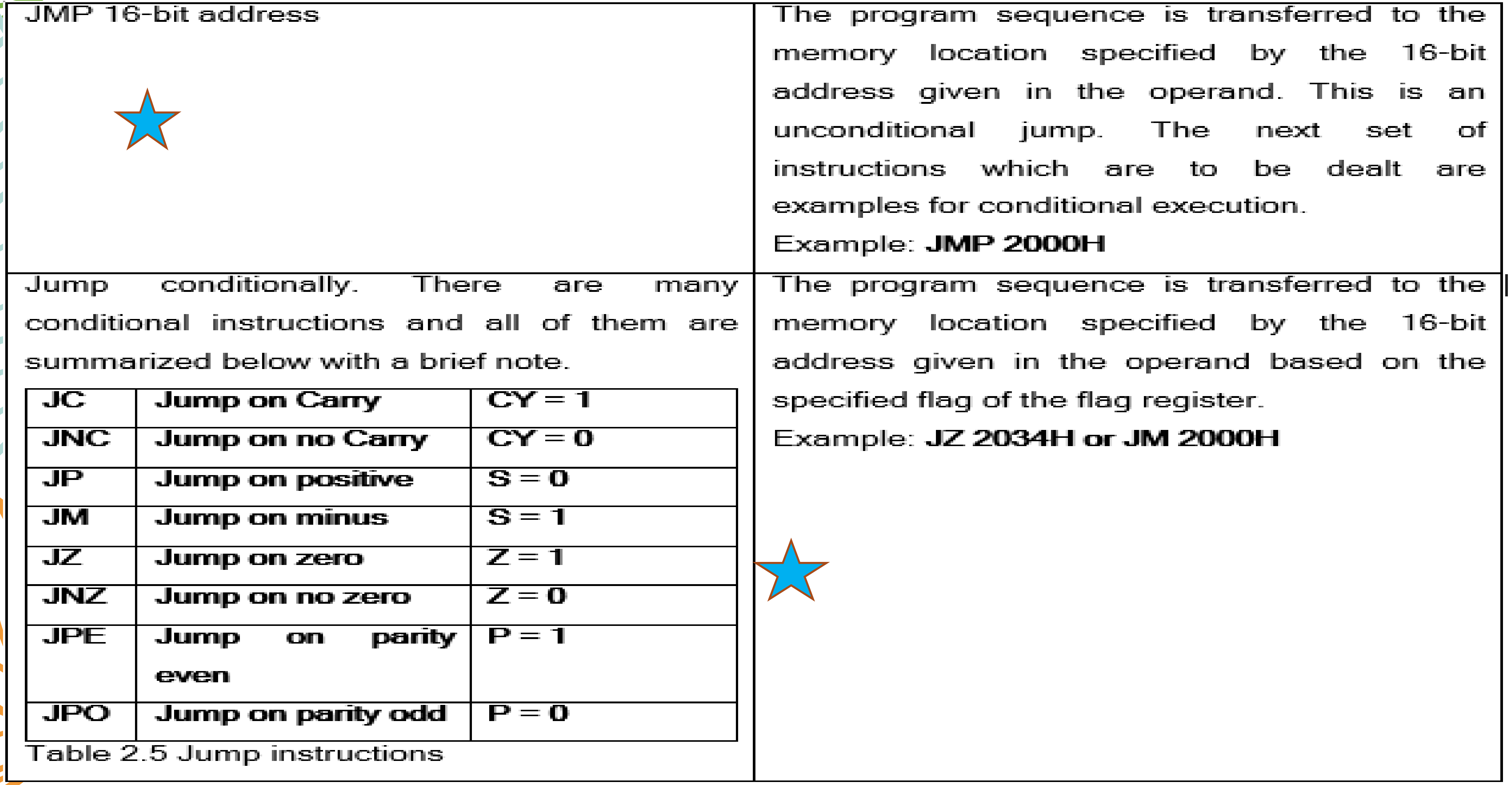
ORA A **ORA M**	This will logically OR the content of the register or the memory location / register with the content of accumulator and result will be stored in accumulator. Flags will be affected based on the result. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. Flags S, Z and P will be modified based on the result of the operation. CY and AC flags will be reset. Example: **ORA B** **ORA M**
XRI	XRI is expanded as XOR immediate. The contents of the accumulator will be XORed with immediate 8 bit data and the result will be placed in the accumulator. Flags S,Z and P will be modified based on the result of the operation. CY and AC flags will be reset. Example: **XRI 10H**
ORI	ORI is expanded as OR immediate. The contents of the accumulator will be ORed with immediate 8 bit data and the result will be placed in the accumulator. Flags S,Z and P will be modified based on the result of the operation. CY and AC flags will be reset. Example: **ORI 10H**

RAL ⭐	Expanded as 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**.
RAR ⭐	Expanded as 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**.

RLC ★	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.**
RRC ★	Expanded as Rotate Accumulator to the right. Each 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

4. Branching instructions

Used to alter the normal flow and shift the control to the different address/area.



JMP 16-bit address	The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. This is an unconditional jump. The next set of instructions which are to be dealt are examples for conditional execution. Example: **JMP 2000H**
Jump conditionally. There are many conditional instructions and all of them are summarized below with a brief note. (see Table 2.5 below)	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 flag register. Example: **JZ 2034H or JM 2000H**

JC	**Jump on Carry**	**CY = 1**
JNC	**Jump on no Carry**	**CY = 0**
JP	**Jump on positive**	**S = 0**
JM	**Jump on minus**	**S = 1**
JZ	**Jump on zero**	**Z = 1**
JNZ	**Jump on no zero**	**Z = 0**
JPE	**Jump on parity even**	**P = 1**
JPO	**Jump on parity odd**	**P = 0**

Table 2.5 Jump instructions

Conditional jump instructions are also available and they are listed in the table 2.6 as follows with a brief note on what each instruction does.

CB	**Call on Carry**	**CY = 1**
CNC	**Call on no Carry**	**CY = 0**
CP	**Call on positive**	**S = 0**
CM	**Call on minus**	**S = 1**
CZ	**Call on zero**	**Z = 1**
CNZ	**Call on no zero**	**Z = 0**
CPE	**Call on parity even**	**P = 1**
CPO	**Call on parity odd**	**P = 0**

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 flag register.

Example: **CP 2000H, CZ 2300H**

Unconditional return from the subroutine **RET**	The program sequence will be transferred from the subroutine to the calling program. This will be done unconditionally. Example: RET
Conditional return instructions are supported as well in 8085. They are listed in the table 2.7 with a brief note.	The program sequence will get transferred from the subroutine to main calling program based on the flag status as shown in the table 2.7. Example: RC

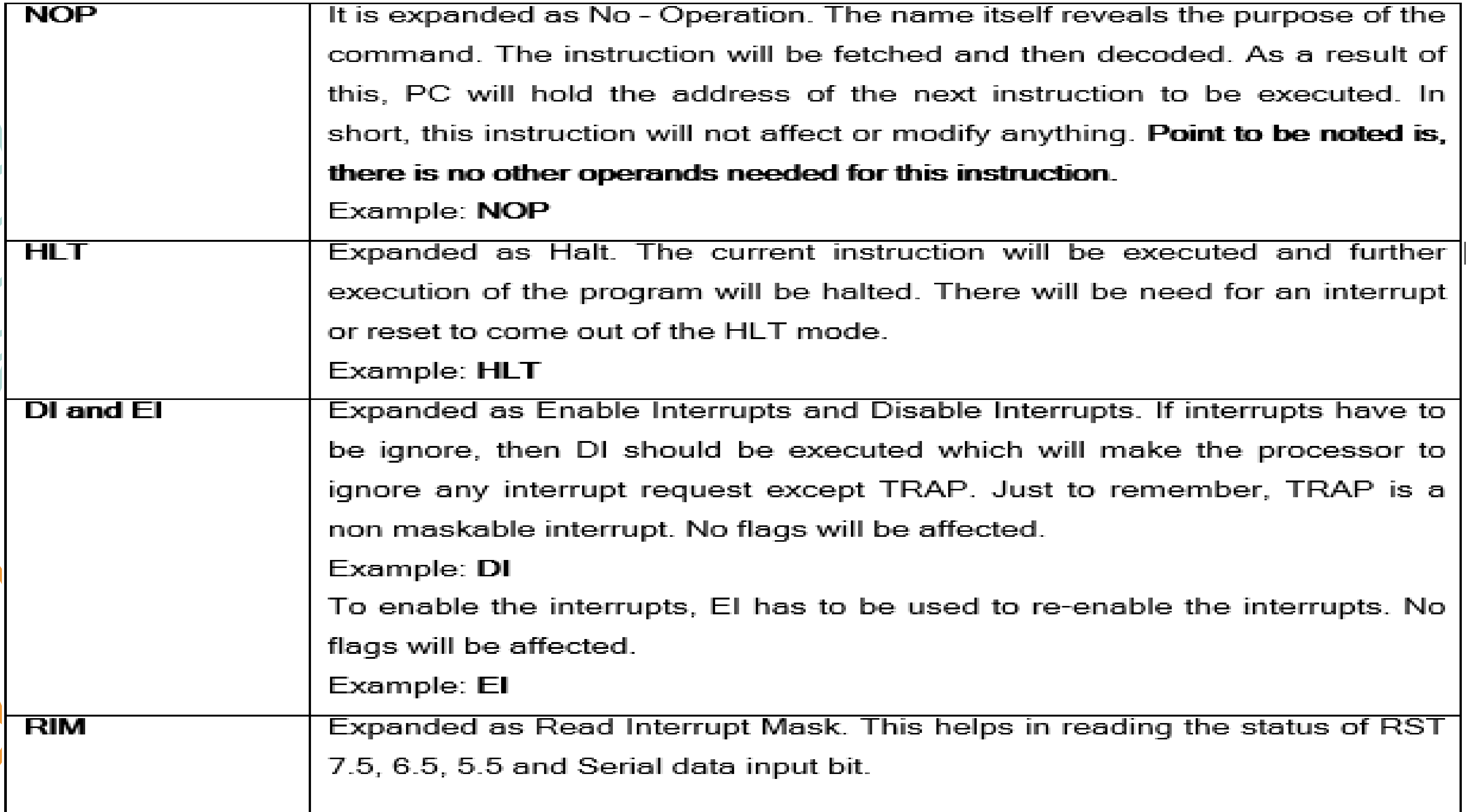
RC	**Return on Carry**	**CY = 1**
RNC	**Return on no Carry**	**CY = 0**
RP	**Return on positive**	**S = 0**
RM	**Return on minus**	**S = 1**
RZ	**Return on zero**	**Z = 1**
RNZ	**Return on no zero**	**Z = 0**
RPE	**Return on parity even**	**P = 1**
RPO	**Return on parity odd**	**P = 0**

RST (0-7)	These instructions can be used to transfer the control of the program to the one of the 8 locations mapped to RST 0 to 7 as follows as shown in table 2.8

Instruction	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

5. Control instructions

Microprocessor is controlled by these control instructions



NOP	It is expanded as No – Operation. The name itself reveals the purpose of the command. The instruction will be fetched and then decoded. As a result of this, PC will hold the address of the next instruction to be executed. In short, this instruction will not affect or modify anything. **Point to be noted is, there is no other operands needed for this instruction.** Example: **NOP**
HLT	Expanded as Halt. The current instruction will be executed and further execution of the program will be halted. There will be need for an interrupt or reset to come out of the HLT mode. Example: **HLT**
DI and EI	Expanded as Enable Interrupts and Disable Interrupts. If interrupts have to be ignore, then DI should be executed which will make the processor to ignore any interrupt request except TRAP. Just to remember, TRAP is a non maskable interrupt. No flags will be affected. Example: **DI** To enable the interrupts, EI has to be used to re-enable the interrupts. No flags will be affected. Example: **EI**
RIM	Expanded as Read Interrupt Mask. This helps in reading the status of RST 7.5, 6.5, 5.5 and Serial data input bit.

Thank You

Reference: Chapter 2, Ramesh Gaonkar, 8085 Microprocessor Architecture Programming and applications with the 8085