Microprocessor - 8085 Architecture

8085 is pronounced as "eighty-eighty-five" microprocessor. It is an 8-bit microprocessor designed by Intel in 1977 using NMOS technology.

It has the following configuration −

- 8-bit data bus
- 16-bit address bus, which can address upto 64KB
- A 16-bit program counter
- A 16-bit stack pointer
- Six 8-bit registers arranged in pairs: BC, DE, HL
- Requires +5V supply to operate at 3.2 MHZ single phase clock

It is used in washing machines, microwave ovens, mobile phones, etc.

8085 Microprocessor – Functional Units

8085 consists of the following functional units −

Accumulator

It is an 8-bit register used to perform arithmetic, logical, I/O & LOAD/STORE operations. It is connected to internal data bus & ALU.

Arithmetic and logic unit

As the name suggests, it performs arithmetic and logical operations like Addition, Subtraction, AND, OR, etc. on 8-bit data.

General purpose register

There are 6 general purpose registers in 8085 processor, i.e. B, C, D, E, H & L. Each register can hold 8-bit data.

These registers can work in pair to hold 16-bit data and their pairing combination is like B-C, D-E & H-L.

Program counter

It is a 16-bit register used to store the memory address location of the next instruction to be executed. Microprocessor increments the program whenever an instruction is being executed, so that the program counter points to the memory address of the next instruction that is going to be executed.

Stack pointer

It is also a 16-bit register works like stack, which is always incremented/decremented by 2 during push & pop operations.

Temporary register

It is an 8-bit register, which holds the temporary data of arithmetic and logical operations.

Flag register

It is an 8-bit register having five 1-bit flip-flops, which holds either 0 or 1 depending upon the result stored in the accumulator.

These are the set of 5 flip-flops −

- Sign (S)
- Zero (Z)
- Auxiliary Carry (AC)
- Parity (P)
- Carry (C)

Its bit position is shown in the following table −

D7	D6	D5	D4	D3	D2	D1	D0
S	Z		AC		P		CY

Instruction register and decoder

It is an 8-bit register. When an instruction is fetched from memory then it is stored in the Instruction register. Instruction decoder decodes the information present in the Instruction register.

Timing and control unit

It provides timing and control signal to the microprocessor to perform operations. Following are the timing and control signals, which control external and internal circuits −

- Control Signals: READY, RD', WR', ALE
- Status Signals: S0, S1, IO/M'
- DMA Signals: HOLD, HLDA
- RESET Signals: RESET IN, RESET OUT

Interrupt control

As the name suggests it controls the interrupts during a process. When a microprocessor is executing a main program and whenever an interrupt occurs, the microprocessor shifts the control from the main program to process the incoming request. After the request is completed, the control goes back to the main program.

There are 5 interrupt signals in 8085 microprocessor: INTR, RST 7.5, RST 6.5, RST 5.5, TRAP.

Serial Input/output control

It controls the serial data communication by using these two instructions: SID (Serial input data) and SOD (Serial output data).

Address buffer and address-data buffer

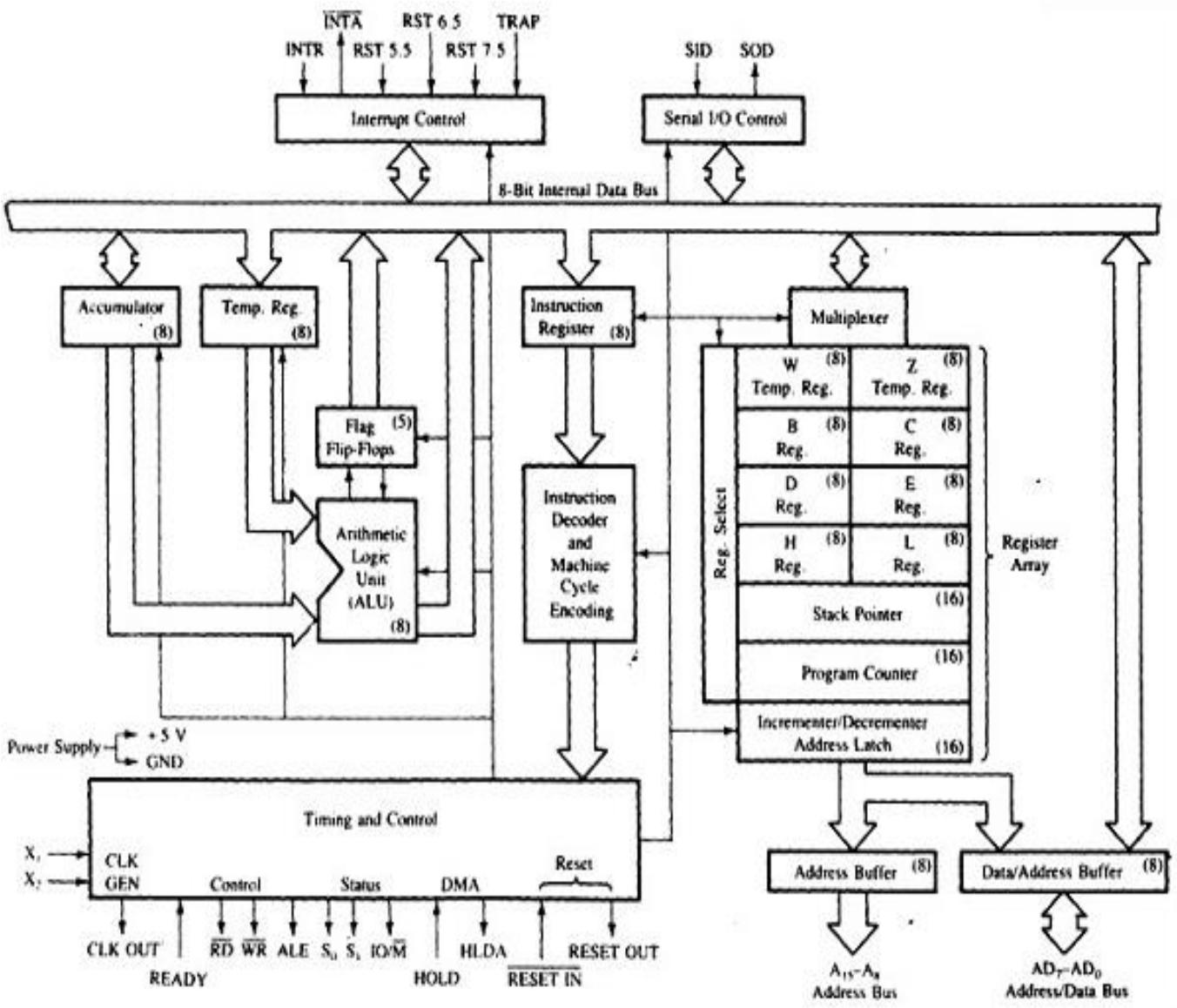
The content stored in the stack pointer and program counter is loaded into the address buffer and address-data buffer to communicate with the CPU. The memory and I/O chips are connected to these buses; the CPU can exchange the desired data with the memory and I/O chips.

Address bus and data bus

Data bus carries the data to be stored. It is bidirectional, whereas address bus carries the location to where it should be stored and it is unidirectional. It is used to transfer the data & Address I/O devices.

8085 Architecture

We have tried to depict the architecture of 8085 with this following image −



Instruction cycle in 8085 Microprocessor

The Program and data which are stored in the memory, are used externally to the microprocessor for executing the complete instruction cycle. Thus to execute a complete instruction of the program, the following steps should be performed by the 8085 microprocessor.

- Fetching the opcode from the memory;
- Decoding the opcode to identify the specific set of instructions;
- Fetching the remaining Bytes left for the instruction, if the instruction length is of 2 Bytes or 3 Bytes;
- Executing the complete instruction procedure.

The given steps altogether constitute the complete instruction cycle. These above mentioned steps are described in detail later. The above instructions are assumed by us for being in the memory, at the specified locations allocated for the memory.

The points to be noted as without fetching of the opcode from the memory the complete instruction would remain incomplete. Secondly decoding should be done, thirdly the fetching process should be done depending on the instruction length. Thirdly the complete execution process should be carried out to complete the entire process of execution.

To have a better idea on Instruction Cycle, let us consider the instruction DCX SP and its instruction cycle into details –

In 8085 Instruction set, **DCX SP** instruction is used to decrement the SP contents by 1. DCX SP instruction is a special case of DCX rp instruction which decreases the content of the register pair. This instruction occupies only 1-Byte in memory.

Mnemonics, Operand	Opcode (in HEX)	Bytes
DCX SP	3B	1

Let us consider that the initial content of SP is 4050H. So after decrement of the content of SP by using **DCX SP** instruction, SP would have the value 404FH. Here is the required tracing table as below –

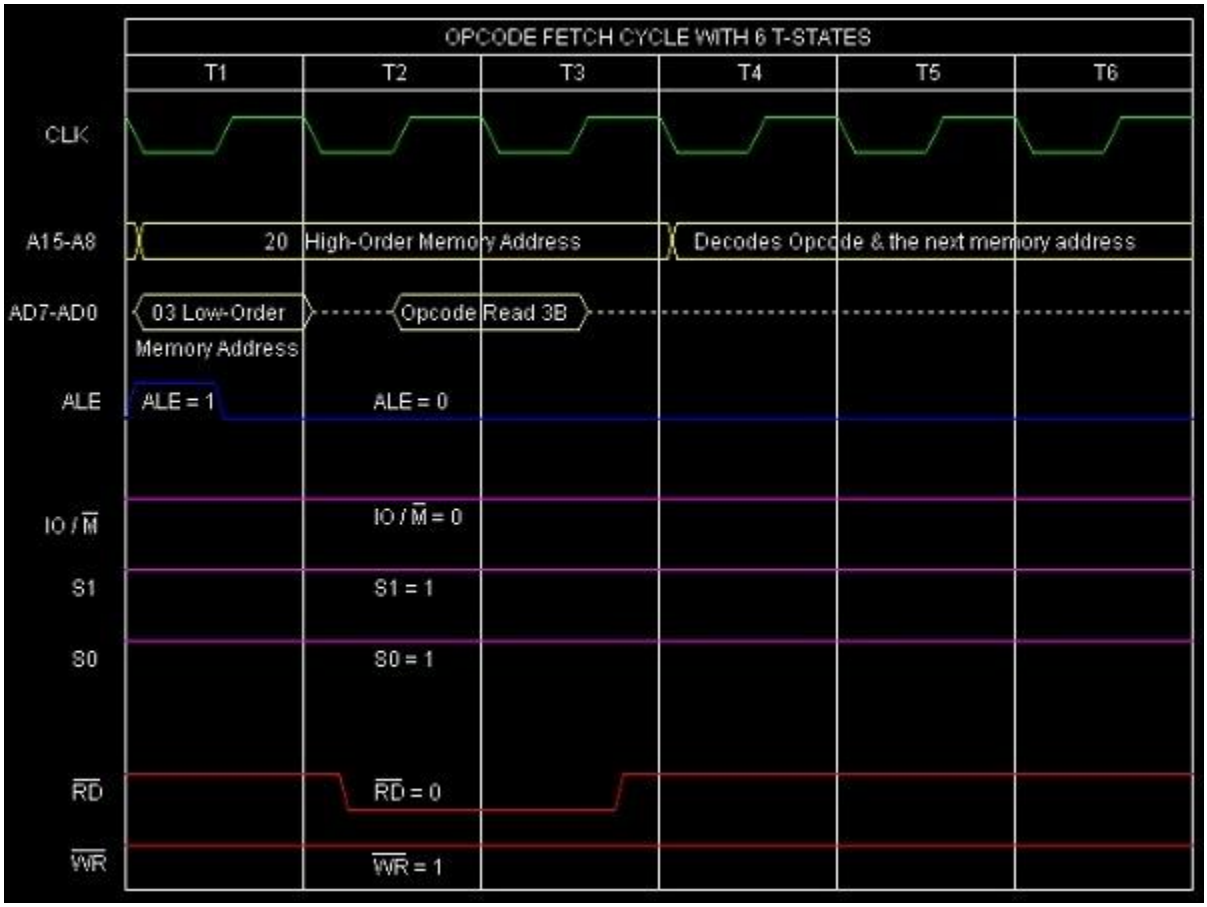
	Before	After

	Before	After
(SP)	4050H	404FH

Here is the required tracing table as below –

Address	Hex Codes	Mnemonic	Comment
2003	3B	DCX SP	SP <-SP – 1

The timing diagram against this instruction **DCX SP** execution is as follows –



Summary: So this instruction **DCX SP** requires 1-Byte, 1-Machine Cycle (Opcode Fetch) and 6 T-States for execution as shown in the timing diagram.

Timing Diagram and machine cycles of 8085 Microprocessor

Timing Diagram

Timing Diagram is a graphical representation. It represents the execution time taken by each instruction in a graphical format. The execution time is represented in T-states.

Instruction Cycle:

The time required to execute an instruction is called instruction cycle.

Machine Cycle:

The time required to access the memory or input/output devices is called machine cycle.

T-State:

The machine cycle and instruction cycle takes multiple clock periods.

A portion of an operation carried out in one system clock period is called as T-state.

1 Machine cycles of 8085

The 8085 microprocessor has 5 (seven) basic machine cycles. They are

Opcode fetch cycle (4T)

Memory read cycle (3 T)

Memory write cycle (3 T)

I/O read cycle (3 T)

I/O write cycle (3 T)

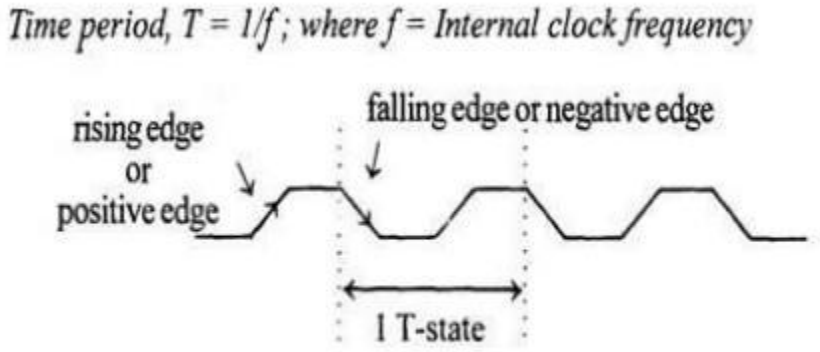


Fig 1.7 Clock Signal

Signal <u>1.Opcode fetch machine cycle of 8085 :</u>

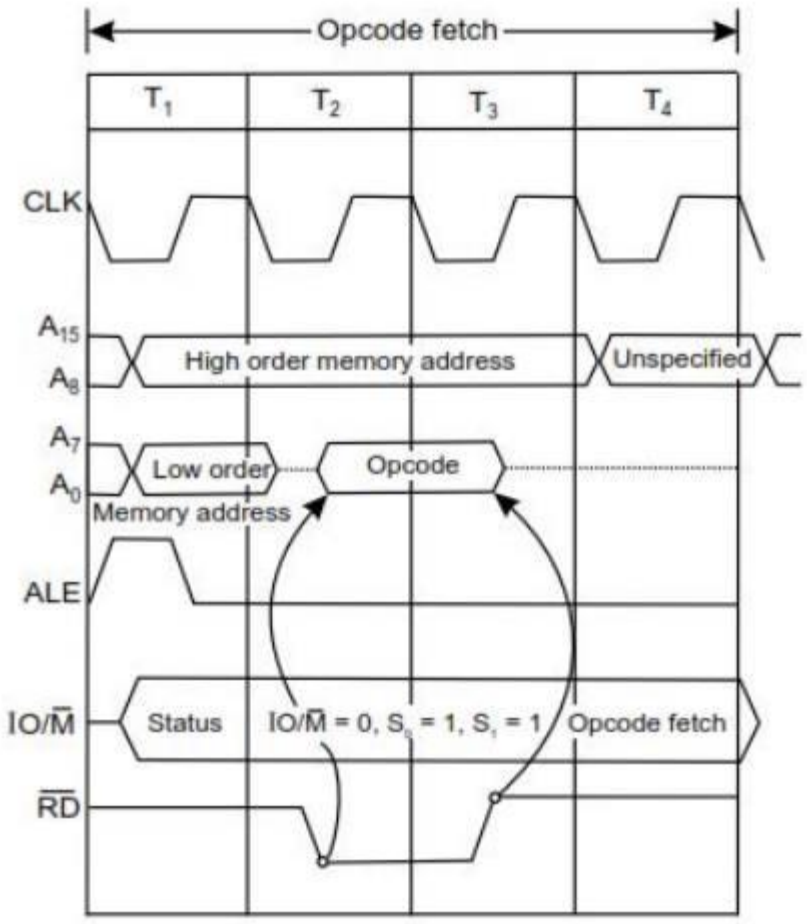


Fig 1.8 Opcode fetch machine cycle

Each instruction of the processor has one byte opcode.

The opcodes are stored in memory. So, the processor executes the opcode fetch machine cycle to fetch the opcode from memory.

Hence, every instruction starts with opcode fetch machine cycle.

The time taken by the processor to execute the opcode fetch cycle is 4T.

In this time, the first, 3 T-states are used for fetching the opcode from memory and the remaining T-states are used for internal operations by the processor.

2. Memory Read Machine Cycle of 8085:

The memory read machine cycle is executed by the processor to read a data byte from memory.

The processor takes 3T states to execute this cycle.

The instructions which have more than one byte word size will use the machine cycle after the opcode fetch machine cycle.

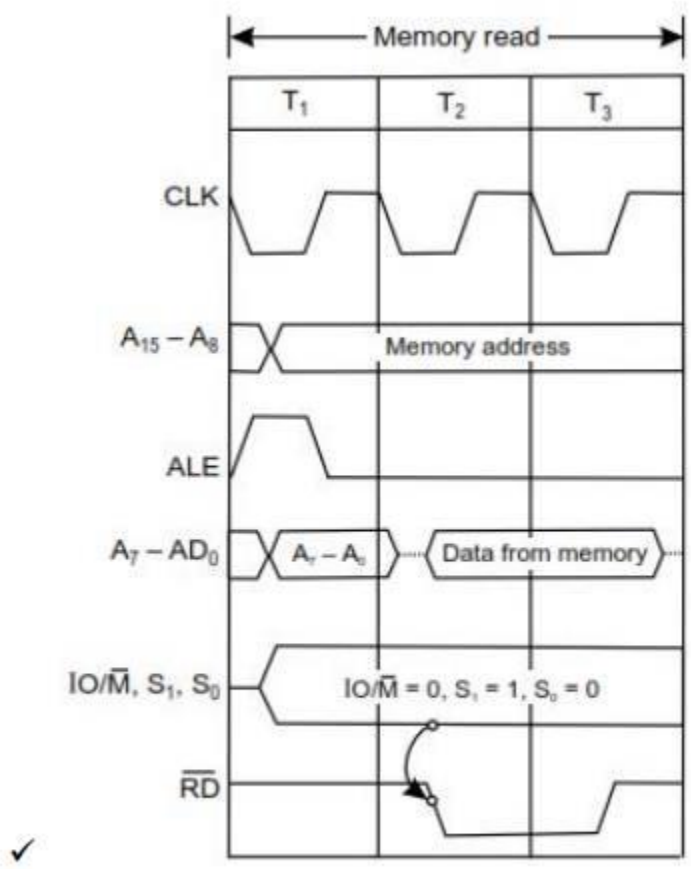


Fig 1.9 Memory Read Machine Cycle

Cycle 3. Memory Write Machine Cycle of 8085

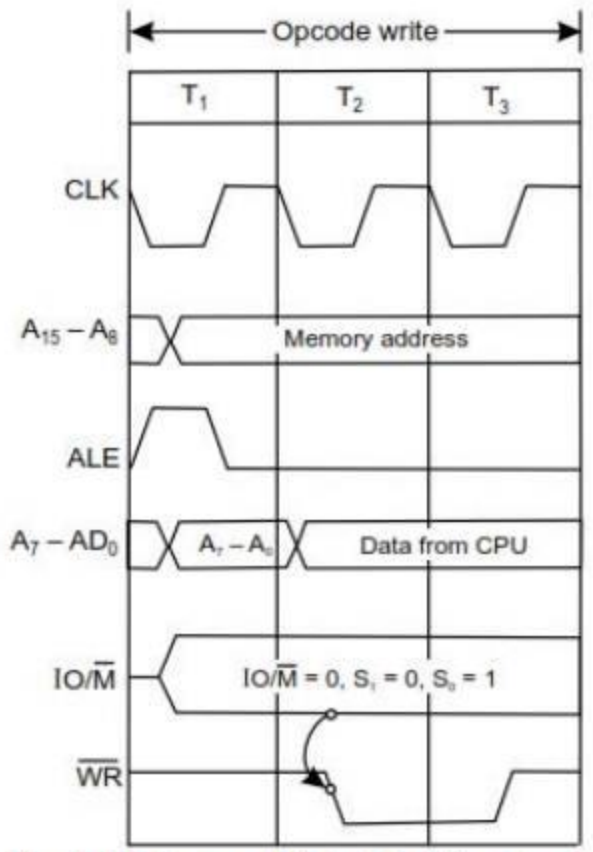


Fig 1.10 Memory Write Machine Cycle

The memory write machine cycle is executed by the processor to write a data byte in a memory location.

The processor takes, 3T states to execute this machine cycle.

4. I/O Read Cycle of 8085

The I/O Read cycle is executed by the processor to read a data byte from I/O port or from the peripheral, which is I/O, mapped in the system.

The processor takes 3T states to execute this machine cycle.

The IN instruction uses this machine cycle during the execution.

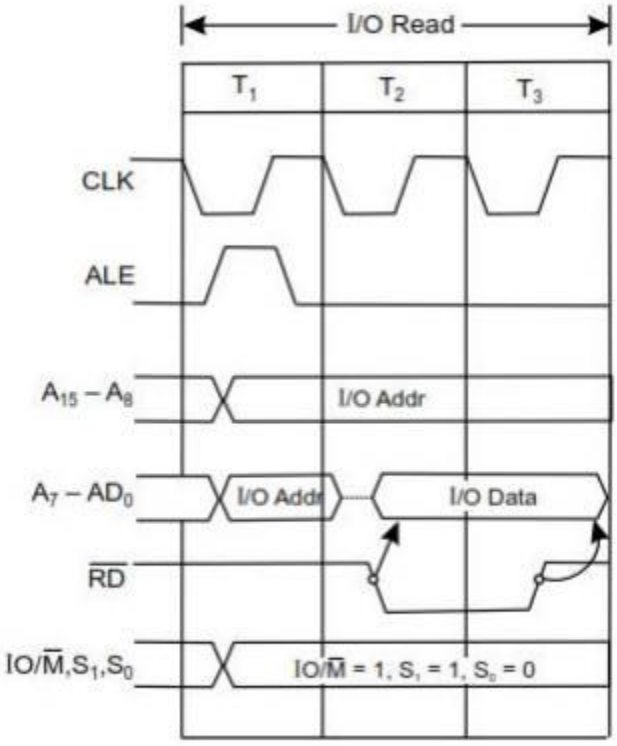


Fig 1.11 I/O Read Cycle

Cycle 1.4.2 Timing diagram for STA 526AH

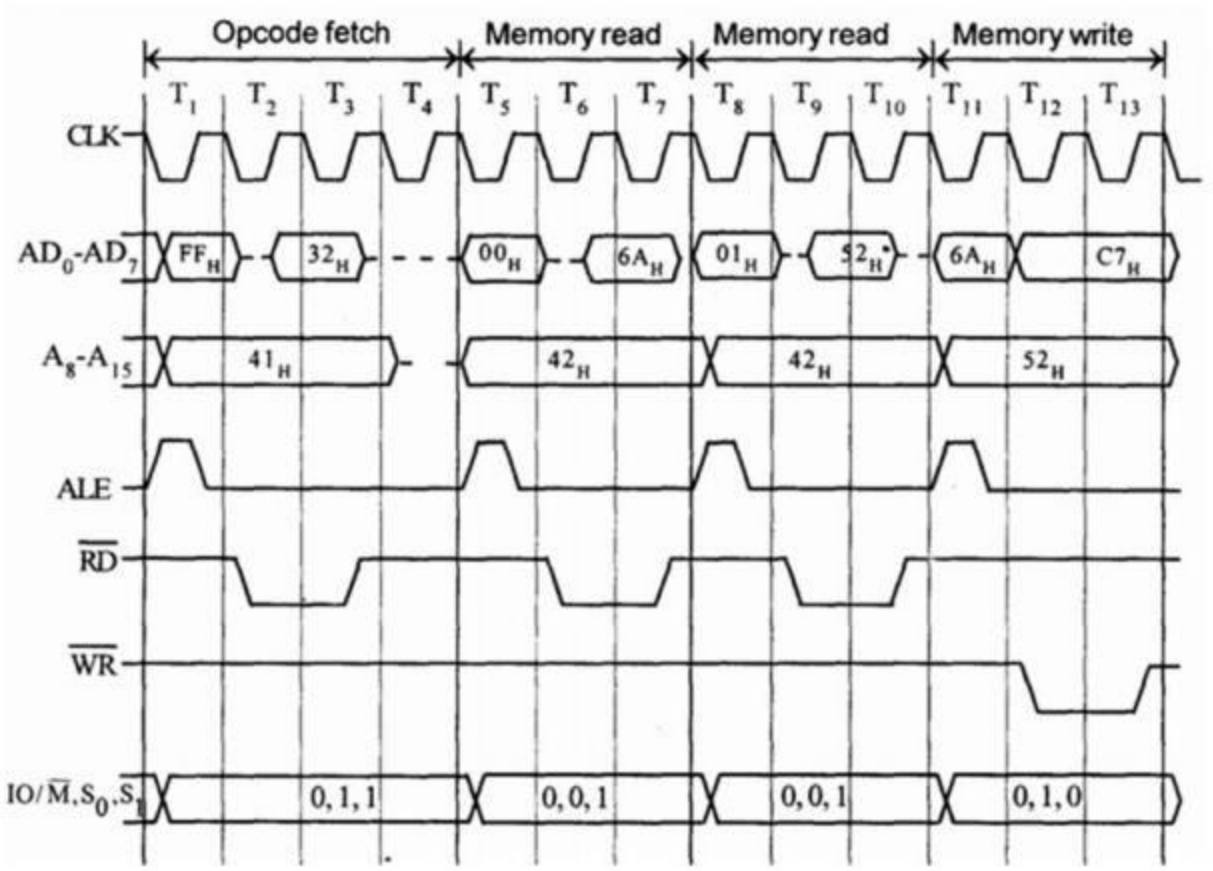


Fig 1.12 Timing Diagram for STA 526A H

Address	Mnemonics	Opcode
41FF	STA 526A$_H$	32$_H$
4200		6A$_H$
4201		52$_H$

STA means Store Accumulator -The contents of the accumulator is stored in the specified address (526A).

The opcode of the STA instruction is said to be 32H. It is fetched from the memory 41FFH (see fig). - OF machine cycle

Then the lower order memory address is read (6A). - Memory Read Machine Cycle

Read the higher order memory address (52).- Memory Read Machine Cycle

The combination of both the addresses are considered and the content from accumulator is written in 526A. - Memory Write Machine Cycle

Assume the memory address for the instruction and let the content of accumulator is C7H. So, C7H from accumulator is now stored in 526A.

Instruction Format

An instruction is a command to the microprocessor to perform a given task on a specified data. Each instruction has two parts: one is task to be performed, called the operation code (opcode), and the second is the data to be operated on, called the operand. The operand (or data) can be specified in various ways. It may include 8-bit (or 16-bit) data, an internal register, a memory location, or 8-bit (or 16-bit) address. In some instructions, the operand is implicit.

Instruction word size

The 8085 instruction set is classified into the following three groups according to word size:

- ✓ One-word or 1-byte instructions
- ✓ Two-word or 2-byte instructions
- ✓ Three-word or 3-byte instructions

In the 8085, "byte" and "word" are synonymous because it is an 8-bit microprocessor. However, instructions are commonly referred to in terms of bytes rather than words.

1 One-Byte Instructions

A 1-byte instruction includes the opcode and operand in the same byte. Operand(s) are internal register and are coded into the instruction

Table 2.1 Example for 1 byte Instruction

Task	Op code	Operand	Binary Code	Hex Code
Copy the contents of the accumulator in the register C.	MOV	C,A	0100 1111	4FH
Add the contents of register B to the contents of the accumulator.	ADD	B	1000 0000	80H
Invert (compliment) each bit in the accumulator.	CMA		0010 1111	2FH

These instructions are 1-byte instructions performing three different tasks. In the first instruction, both operand registers are specified. In the second instruction, the operand B is specified and the accumulator is assumed. Similarly, in the third instruction, the accumulator is assumed to be the implicit operand. These instructions are stored in 8- bit binary format in memory; each requires one memory location.

MOV rd, rs

rd ← rs copies contents of rs into rd.

Coded as 01 ddd sss

where ddd is a code for one of the 7 general registers which is the destination of the data, sss is the code of the source register.

Example: MOV A,B

Coded as 01111000 = 78H = 170 octal (octal was used extensively in instruction design of such processors).

ADD r

A ← A + r

2 Two-Byte Instructions

In a two-byte instruction, the first byte specifies the operation code and the second byte specifies the operand. Source operand is a data byte immediately following the opcode. For example:

Table 2.2 Example for 2 byte Instruction

Table 2.2 Example for 2 byte Instruction

Task	Opcode	Operand	Binary Code	Hex Code	
Load an 8-bit data byte in the accumulator.	MVI	A, Data	0011 1110 DATA	3E Data	First Byte Second Byte

The instruction would require two memory locations to store in memory.

MVI r,data

r ← data

Example: MVI A,30H coded as 3EH 30H as two contiguous bytes.

This is an example of immediate addressing.

ADI data

A ← A + data

OUT port

0011 1110

DATA

Where port is an 8-bit device address. (Port) ← A.

Since the byte is not the data but points directly to where it is located this is called direct addressing.

3 Three-Byte Instructions

In a three-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit address. Note that the second byte is the low-order address and the third byte is the high-order address.

opcode + data byte + data byte

Table 3.3 Example for 3 byte Instruction

Table 3.3 Example for 3 byte Instruction

Task	Opcode	Operand	Binary code	Hex Code	
Transfer the program sequence to the memory location 2085H.	JMP	2085H	1100 0011	C3	First byte
			1000 0101	85	Second Byte
			0010 0000	20	Third Byte

This instruction would require three memory locations to store in memory.

Three byte instructions - opcode + data byte + data byte

LXI rp, data16

rp is one of the pairs of registers BC, DE, HL used as 16-bit registers. The two data bytes are 16-bit data in L H order of significance.

rp ← data16

LXI H,0520H coded as 21H 20H 50H in three bytes. This is also immediate addressing.

LDA addr

A ← (addr) Addr is a 16-bit address in L H order.

Example: LDA 2134H coded as 3AH 34H 21H. This is also an example of direct addressing.

Addressing Modes in 8085

These are the instructions used to transfer the data from one register to another register, from the memory to the register, and from the register to the memory without any alteration in the content. Addressing modes in 8085 is classified into 5 groups −

Immediate addressing mode

In this mode, the 8/16-bit data is specified in the instruction itself as one of its operand. **For example:** MVI K, 20F: means 20F is copied into register K.

Register addressing mode

In this mode, the data is copied from one register to another. **For example:** MOV K, B: means data in register B is copied to register K.

Direct addressing mode

In this mode, the data is directly copied from the given address to the register. **For example:** LDB 5000K: means the data at address 5000K is copied to register B.

Indirect addressing mode

In this mode, the data is transferred from one register to another by using the address pointed by the register. **For example:** MOV K, B: means data is transferred from the memory address pointed by the register to the register K.

Implied addressing mode

This mode doesn’t require any operand; the data is specified by the opcode itself. **For example:** CMP.

Interrupts in 8085

Interrupts are the signals generated by the external devices to request the microprocessor to perform a task. There are 5 interrupt signals, i.e. TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR.

Interrupt are classified into following groups based on their parameter −

- **Vector interrupt** − In this type of interrupt, the interrupt address is known to the processor. **For example:** RST7.5, RST6.5, RST5.5, TRAP.
- **Non-Vector interrupt** − In this type of interrupt, the interrupt address is not known to the processor so, the interrupt address needs to be sent externally by the device to perform interrupts. **For example:** INTR.
- **Maskable interrupt** − In this type of interrupt, we can disable the interrupt by writing some instructions into the program. **For example:** RST7.5, RST6.5, RST5.5.
- **Non-Maskable interrupt** − In this type of interrupt, we cannot disable the interrupt by writing some instructions into the program. **For example:**TRAP.
- **Software interrupt** − In this type of interrupt, the programmer has to add the instructions into the program to execute the interrupt. There are 8 software interrupts in 8085, i.e. RST0, RST1, RST2, RST3, RST4, RST5, RST6, and RST7.
- **Hardware interrupt** − There are 5 interrupt pins in 8085 used as hardware interrupts, i.e. TRAP, RST7.5, RST6.5, RST5.5, INTA.

Note − NTA is not an interrupt, it is used by the microprocessor for sending acknowledgement. TRAP has the highest priority, then RST7.5 and so on.

Interrupt Service Routine (ISR)

A small program or a routine that when executed, services the corresponding interrupting source is called an ISR.

TRAP

It is a non-maskable interrupt, having the highest priority among all interrupts. Bydefault, it is enabled until it gets acknowledged. In case of failure, it executes as ISR and sends the data to backup memory. This interrupt transfers the control to the location 0024H.

RST7.5

It is a maskable interrupt, having the second highest priority among all interrupts. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 003CH address.

RST 6.5

It is a maskable interrupt, having the third highest priority among all interrupts. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 0034H address.

RST 5.5

It is a maskable interrupt. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 002CH address.

INTR

It is a maskable interrupt, having the lowest priority among all interrupts. It can be disabled by resetting the microprocessor.

When **INTR signal goes high**, the following events can occur −

- The microprocessor checks the status of INTR signal during the execution of each instruction.
- When the INTR signal is high, then the microprocessor completes its current instruction and sends active low interrupt acknowledge signal.
- When instructions are received, then the microprocessor saves the address of the next instruction on stack and executes the received instruction.

Data transfer instructions in 8085 microprocessor

Data tranfer instructions are the instructions which transfers data in the microprocessor. They are also called copy instructions.

Following is the table showing the list of logical instructions:

OPCODE	OPERAND	EXPLANATION	EXAMPLE
MOV	Rd, Rs	Rd = Rs	MOV A, B
MOV	Rd, M	Rd = Mc	MOV A, 2050
MOV	M, Rs	M = Rs	MOV 2050, A

OPCODE	OPERAND	EXPLANATION	EXAMPLE
MVI	Rd, 8-bit data	Rd = 8-bit data	MVI A, 50
MVI	M, 8-bit data	M = 8-bit data	MVI 2050, 50
LDA	16-bit address	A = contents at address	LDA 2050
STA	16-bit address	contents at address = A	STA 2050
LHLD	16-bit address	directly loads at H & L registers	LHLD 2050
SHLD	16-bit address	directly stores from H & L registers	SHLD 2050
LXI	r.p., 16-bit data	loads the specified register pair with data	LXI H, 3050
LDAX	r.p.	indirectly loads at the accumulator A	LDAX H
STAX	16-bit address	indirectly stores from the accumulator A	STAX 2050
XCHG	none	exchanges H with D, and L with E	XCHG
PUSH	r.p.	pushes r.p. to the stack	PUSH H
POP	r.p.	pops the stack to r.p.	POP H

OPCODE	OPERAND	EXPLANATION	EXAMPLE
IN	8-bit port address	inputs contents of the specified port to A	IN 15
OUT	8-bit port address	outputs contents of A to the specified port	OUT 15

In the table,
R stands for register
M stands for memory
r.p. stands for register pair

Arithmetic instructions in 8085 microprocessor

Arithmetic Instructions are the instructions which perform basic arithmetic operations such as addition, subtraction and a few more. In 8085 microprocessor, the destination operand is generally the accumulator. In 8085 microprocessor, the destination operand is generally the accumulator.

Following is the table showing the list of arithmetic instructions:

OPCODE	OPERAND	EXPLANATION	EXAMPLE
ADD	R	A = A + R	ADD B
ADD	M	A = A + Mc	ADD 2050
ADI	8-bit data	A = A + 8-bit data	ADI 50
ADC	R	A = A + R + prev. carry	ADC B
ADC	M	A = A + Mc + prev. carry	ADC 2050

OPCODE	OPERAND	EXPLANATION	EXAMPLE
ACI	8-bit data	A = A + 8-bit data + prev. carry	ACI 50
SUB	R	A = A – R	SUB B
SUB	M	A = A – Mc	SUB 2050
SUI	8-bit data	A = A – 8-bit data	SUI 50
SBB	R	A = A – R – prev. carry	SBB B
SBB	M	A = A – Mc -prev. carry	SBB 2050
SBI	8-bit data	A = A – 8-bit data – prev. carry	SBI 50
INR	R	R = R + 1	INR B
INR	M	M = Mc + 1	INR 2050
INX	r.p.	r.p. = r.p. + 1	INX H
DCR	R	R = R – 1	DCR B
DCR	M	M = Mc – 1	DCR 2050

OPCODE	OPERAND	EXPLANATION	EXAMPLE
DCX	r.p.	r.p. = r.p. – 1	DCX H
DAD	r.p.	HL = HL + r.p.	DAD H

Logical instructions in 8085 microprocessor

Last Updated: 22-05-2018

Logical instructions are the instructions which perform basic logical operations such as AND, OR, etc. In 8085 microprocessor, the destination operand is always the accumulator. Here logical operation works on a bitwise level.

Following is the table showing the list of logical instructions:

OPCODE	OPERAND	DESTINATION	EXAMPLE
ANA	R	A = A AND R	ANA B
ANA	M	A = A AND Mc	ANA 2050
ANI	8-bit data	A = A AND 8-bit data	ANI 50
ORA	R	A = A OR R	ORA B
ORA	M	A = A OR Mc	ORA 2050

OPCODE	OPERAND	DESTINATION	EXAMPLE
ORI	8-bit data	A = A OR 8-bit data	ORI 50
XRA	R	A = A XOR R	XRA B
XRA	M	A = A XOR Mc	XRA 2050
XRI	8-bit data	A = A XOR 8-bit data	XRI 50
CMA	none	A = 1’s compliment of A	CMA
CMP	R	Compares R with A and triggers the flag register	CMP B
CMP	M	Compares Mc with A and triggers the flag register	CMP 2050
CPI	8-bit data	Compares 8-bit data with A and triggers the flag register	CPI 50
RRC	none	Rotate accumulator right without carry	RRC
RLC	none	Rotate accumulator left without carry	RLC
RAR	none	Rotate accumulator right with carry	RAR
RAL	none	Rotate accumulator left with carry	RAR

OPCODE	OPERAND	DESTINATION	EXAMPLE
CMC	none	Compliments the carry flag	CMC
STC	none	Sets the carry flag	STC

In the table,
R stands for register
M stands for memory
Mc stands for memory contents

Branching instructions in 8085 microprocessor

Last Updated: 29-07-2020

Branching instructions refer to the act of switching execution to a different instruction sequence as a result of executing a branch instruction.

The three types of branching instructions are:

1. Jump (unconditional and conditional)
2. Call (unconditional and conditional)
3. Return (unconditional and conditional)

1. Jump Instructions – The jump instruction transfers the program sequence to the memory address given in the operand based on the specified flag. Jump instructions are 2 types: Unconditional Jump Instructions and Conditional Jump Instructions.
(a) Unconditional Jump Instructions: Transfers the program sequence to the described memory address.

OPCODE	OPERAND	EXPLANATION	EXAMPLE

OPCODE	OPERAND	EXPLANATION	EXAMPLE
JMP	address	Jumps to the address	JMP 2050

(b) Conditional Jump Instructions: Transfers the program sequence to the described memory address only if the condition in satisfied.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
JC	address	Jumps to the address if carry flag is 1	JC 2050
JNC	address	Jumps to the address if carry flag is 0	JNC 2050
JZ	address	Jumps to the address if zero flag is 1	JZ 2050
JNZ	address	Jumps to the address if zero flag is 0	JNZ 2050
JPE	address	Jumps to the address if parity flag is 1	JPE 2050
JPO	address	Jumps to the address if parity flag is 0	JPO 2050
JM	address	Jumps to the address if sign flag is 1	JM 2050
JP	address	Jumps to the address if sign flag 0	JP 2050

2. Call Instructions – The call instruction transfers the program sequence to the memory address given in the operand. Before transferring, the address of the next instruction after CALL is pushed onto the stack. Call instructions are 2 types: Unconditional Call Instructions and Conditional Call Instructions.

(a) Unconditional Call Instructions: It transfers the program sequence to the memory address given in the operand.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
CALL	address	Unconditionally calls	CALL 2050

(b) Conditional Call Instructions: Only if the condition is satisfied, the instructions executes.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
CC	address	Call if carry flag is 1	CC 2050
CNC	address	Call if carry flag is 0	CNC 2050
CZ	address	Calls if zero flag is 1	CZ 2050
CNZ	address	Calls if zero flag is 0	CNZ 2050
CPE	address	Calls if parity flag is 1	CPE 2050
CPO	address	Calls if parity flag is 0	CPO 2050
CM	address	Calls if sign flag is 1	CM 2050
CP	address	Calls if sign flag is 0	CP 2050

3. Return Instructions – The return instruction transfers the program sequence from the subroutine to the calling program. Return instructions are 2 types: Unconditional Jump Instructions and Conditional Jump Instructions.

(a) Unconditional Return Instruction: The program sequence is transferred unconditionally from the subroutine to the calling program.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
RET	none	Return from the subroutine unconditionally	RET

(b) Conditional Return Instruction: The program sequence is transferred unconditionally from the subroutine to the calling program only is the condition is satisfied.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
RC	none	Return from the subroutine if carry flag is 1	RC
RNC	none	Return from the subroutine if carry flag is 0	RNC
RZ	none	Return from the subroutine if zero flag is 1	RZ
RNZ	none	Return from the subroutine if zero flag is 0	RNZ
RPE	none	Return from the subroutine if parity flag is 1	RPE
RPO	none	Return from the subroutine if parity flag is 0	RPO
RM	none	Returns from the subroutine if sign flag is 1	RM
RP	none	Returns from the subroutine if sign flag is 0	RP

Stack I/O, and Machine Control Instructions:

The following instructions affect the Stack and/or Stack Pointer:

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

The I/0 instructions are as follows:

IN - Initiate Input Operation

OUT - Initiate Output Operation

The Machine Control instructions are as follows:

EI - Enable Interrupt System

DI - Disable Interrupt System

HLT - Halt

NOP - No Operation