ARCHITECTURE OF MICROPROCESSORS

1

GPSPDC Stands for General purpose stored program digital computer

Block Structure

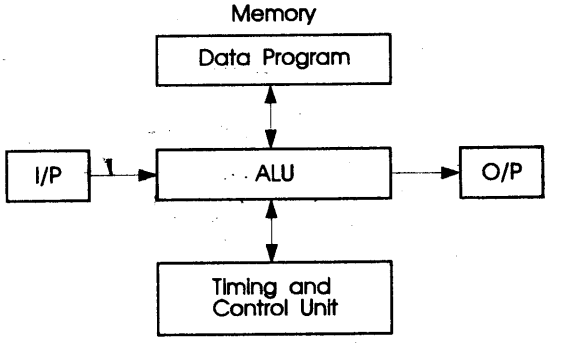


Fig. 1. Block diagram of a digital computer.

In a microcomputer we will have

(*i*) CPU (*ii*) RAM (*iii*) ROM (*iv*) I/O

(a) CPU on a single chip is called a microprocessor.

(b) CPU + Memory + I/O devices is a microcomputer.

(c) Microcomputer + software is a microcomputer system.

Now it is clear that the job of a CPU is to be done with a microprocessor. What should be the integral arrangement of microprocessor to perform the task ?

1.1 COMPONENTS OF A MICROPROCESSOR

Suppose if we wish to execute a program written in any language. This program will be stored in memory just before execution.

(PC) → READ i

(PC + 1) → Write i, a

(PC + 2) → STOP

Now microprocessor must know the program starts from which place of memory. This has to be told to microprocessor. Once microprocessor knows the starting location (normally called as address) the microprocessor will send address to memory and then get in the contents of memory. After that it will do some processing (at this time it is unimportant what it does with fetched data) and then again it shall read the content of next location. This is done by incrementing the current number which is stored in this as address. The process of fetching is repeated. Again the number is incremented and so on.

The above task is done by a Register called PROGRAM COUNTER (PC). Microprocessor fetches the contents of memory from location specified in PC, execute the instruction and then again increment the PC and this process is repeated till the program is over.

Now let us worry about PC in the following ways.

(*i*) What should be the initial setting of PC ?

(*ii*) At what time it should be incremented ?

(*iii*) What should be the length of PC ?

(*iv*) For point (*i*) initial setting of PC should always be the number of starting address of program in memory, which may change with program writing. This will complicate the situation and hence a standard procedure is adopted that on power ON/Reset, PC always contains all zeros *i.e.*, it starts from all zeros. Then the first location's address becomes all zeros. At this location programer can write jumps instruction in a small program which loads the new value in PC.

For point (*ii*) Once an instruction is fetched and executed, the next instruction is to be fetched by incrementing PC. This increment can be done at the end of current instruction under execution. Incrementing PC will certainly need atleast one clock period. In this scheme microprocessor will increment PC and remaining components of microprocessor will maintain their status. It is suggested that if we know in advance that PC is going to be incremented, then why not increment it in time before the current instruction execution process is finished. This will take 4 clock periods in normal cycle timing. (Normal cycle time may be from 4 clock – 18 clock periods). Hence we say that PC always points to the next instruction which is going to be executed, because it increments PC as soon as current instruction is fetched.

For point (*iii*) what should be the number of address lines or length of this program counter say it is of 8 bit length. Then maximum number of memory location it can address is $2^8 = 256$ location. Hence the maximum size program can be 256 bytes only. Similarly for 16 bits it is 64 K which looks to be quite adequate for addressing large memory space. Though microprocessors with 24 and 32 bits program counter are also available.

1.1.1 INSTRUCTION REGISTER

Once the instruction is fetched from memory, internally it travels on a 8 bit parallel bus and is stored in a register called instruction register.

1.1.2 INSTRUCTION DECODER

The code so fetched from memory is stored in a decoder. It is fed to Instruction decoder. This decoder is simply a bit input and 2^n bit output signals are generated out of it. Now these signals act as control signals. The output of instruction decoder is fed to control and timing unit. The control and timing unit generates control and timing signals.

1.1.3 ACCUMULATOR

Inside the microprocessor, arithmetic and logical operations will be carried out on data. Normally the

operand is in accumulator and length of accumulator is equal to its word length. If the operation is between two operands, one of the operand will always be in the accumulator. Result of the operation will be stored in ACC (earlier microprocessors) and old contents will be washed out. It is not necessary that microprocessor should have only one accumulator, it may have more than one also.

1.1.4 ARITHMETIC LOGIC UNIT (ALU)

The operation between two operands or on a single operand is carried out in this unit. This has the capability to carry out simple arithmetic operation like addition subtraction etc. Logical operations are OR, AND, NOT. Exclusive OR etc. Again the result of the operation is normally stored in accumulator. There are few temporary registers connected to this unit, which are not accessible to the user and is used by ALU for storage of intermediate results generated due to the instruction execution process.

1.1.5 GENERAL PURPOSE SCRATCH PAD REGISTER

Once the memory is addressed by microprocessor on address lines, data is fetched and is placed in accumulator or other registers inside the microprocessor. If this data is to be operated upon, it is placed in a register called accumulator. If their is already some data in accumulator, the current data is stored in some general purpose registers which are there inside the microprocessor. Length of these registers is normally equal to the word length of a microprocessor. It can be higher than the word length also. It is also possible to use these registers either as simple registers and two registers can be used as a pair also.

1.1.6 FLAGS

Due to many logical and arithmetic operations on data certain cases may occur like overflow of ALU adder or underflow, accumulator contents may become zero as a result of execution of some instuction. Some times extra flip-flops and circuitary is provided to indicate sign of data or parity of data. Such as indication is provided by few flip-flops which are set/reset by ALU depending upon the operation's result. These flip-flops normally are :

(*i*) Zero (*ii*) Carry (*iii*) Parity (*iv*) Sign (*v*) AC

It is possible for program to jump to another portion of program depending upon status of these flags.

1.1.7 STACK REGISTER (SP) AND STACK AREA

We have already seen that how program counter works and is incremented by 1 every time an instruction is fetched. This rule is violated when a jump instruction is executed. During an excution of a jump instruction the processor replaces the contents of program counter with the address given in the jump instruction.

A special kind of jump occurs when program calls another program (subroutine). In this kind of jump, the processor is required to remember the contents of program counter at the time of jump. This will enable processor to resume the task of execution of main program when it has finished the last instruction of the subroutine.

How processors handles these routines ? On receiving CALL instruction, it increments program counter and stores counter contents in a specified reserved memory area known as STACK. Therefore STACK contains the address of the instruction to be executed after the subroutine is completed. After transferring contents of PC to the stack, new contents are reloaded into program counter.

The last instruction of any subroutine is a RETURN instruction. On recieving this instruction processor places contents from stack to program counter and the old program is continued.

Now question arises that where should be this stack area. Many microprocessors maintains this area inside the processor itself. That type is called as internal stacks. Other way is to have this stack some where in memory and have a pointer pointed to the stack top (or next availabale empty location).

The external stack will provide any number of subroutine nesting and also many times, the contents of ACC or general purpose registers is pushed to stacks to be retrieved later. These facilities will not be available in case the stack area is inside the microprocessor due to limited area of stack.

1.1.8 INTERNAL DATA BUS

We know that there are various components and each has different function to do. Transfer of data may take place either between them or an outer data lines and inner registers. For this communication purpose an internal n bit (n is word length) data bus is provided. General purpose Registers, Instruction Register, Arithmetic logic unit, Accumulator, Flags, Temporary Registers have been connected to this data bus in a bidirectional way, as shown in fig 1.2.

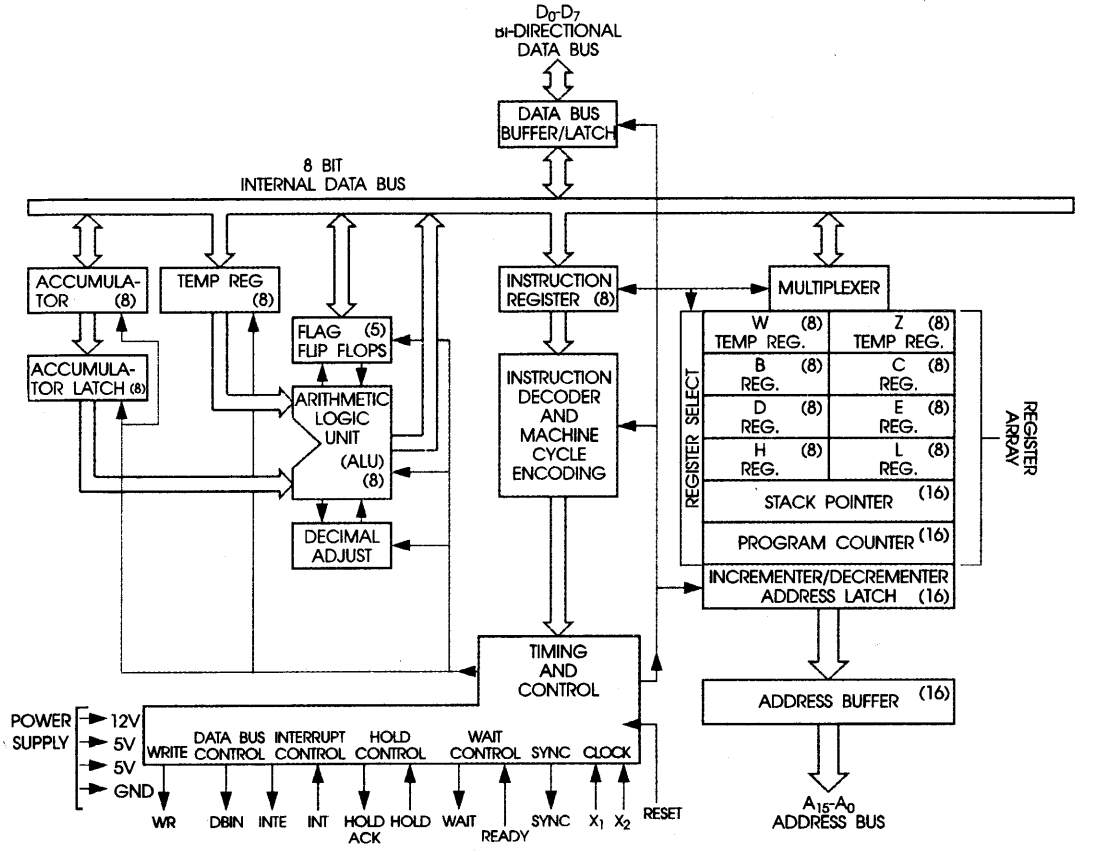


Fig. 1.2. Block diagram

Communication of program counter to the outside world takes place through Address buffer and similarly data transactions between internal data bus and outside world takes place through Data bus latch.

Above is architecture which is available in microprocessor 8080. Now let us see how this architecture could be improved upon and construct the architecture of 8085.

1.1.9 INTERRUPTS

Interrupts are useful to improve efficiency of processing. Consider the case when a microprocessor is processing a large amount of data and certain amount of data is also to be outputted to printer. Now CPU can output a single byte in one machine cycle where as printer may take certain amount of time for printing some data. Infact it may take few milliseconds of time. The CPU has to wait for printer being free to print the next data.

Thus CPU remains idle for several cycles of times. Hence if an interrupt capability is introduced in which the printer will interrupt CPU when ever it is free. It asks for data and on outputting required data, CPU will return to its main program. In this way CPU does not waste time. Hence the system efficiency could be improved by using interrupts.

Interrupts are normally difficult to handle, they have random nature and need careful handling. Their occurence is completely random and can be compared with the process of getting a telephone call by a subscriber. The phenomena of occurence (ringing of telephone) is perfectly random in nature and it is never known when the phone is going to ring, and similarly it is not known to CPU when interrupt will occur in its main program.

Similar to vested calls, their can be vested interrupts also. By now the reader can visualize the similarity in execution of interrupt and a call subroutine. Earlier microprocessors (like 8080, 6800) used to have only one interrupt line and this one interrupt line is not certainly sufficient as number of devices are connected to CPU. These devices normally will carry out data transfer in interrupt mode. If there is only one interrupt line and interrupting devices are more than one, extra hardware is used in between so as to enable number of devices to communicate with single interrupt line.

1.1.10 SERIAL INPUT AND OUTPUT

Many times the output of equipments like teletypewriter or teleprinter is available in serial form and if such a data is to be processed by microprocessor after reading this data, then first this data has to be converted into a parallel form (by using a serial to parallel convertor) and then microprocessor can read this data and process it.

Also their is normally requirement of data to be available in serial form. Such a requirement normally exists when CPU and the devices (peripheral) are located far away. Also there are definite advantages of transmitting serial data over parallel transmission of data

(*a*) Only a pair of wire is needed for serial transmission as compared to set of 8 wires needed in parallel transmission.

(*b*) Cross talk is negligible in serial transmission as compared to parallel transmission.

(*c*) Channel Band width required is less in serial transmission as compared to parallel transmission of data.

It will be certainly beneficial if the microprocessor can have some intellegence to provide this serial data reception and serial data transmission capability. In microprocessor 8085 such a function is provided on two of its lines namely SID (Serial Input Data) and SOD (Serial Output Data).

1.1.11 MULTIPLEXED ADDRESS AND DATA LINES

Because of additional facilities available in 8085, which we have discussed above, the situation was created that the number of pin went beyond 40. Normally standard IC package is of 40 pins only. So to conserve the number of pins, lower address lines A_0–A_7 and Data lines D_0–D_7 is used in multiplexed mode, and are designated as AD_0–AD_7.

If points (a), (b) and (c) discussed above are added to the architecture of 8080, the architecture of 8085 is arrived. The pin diagram and block diagram of 8085 is shown in figure 1.3.

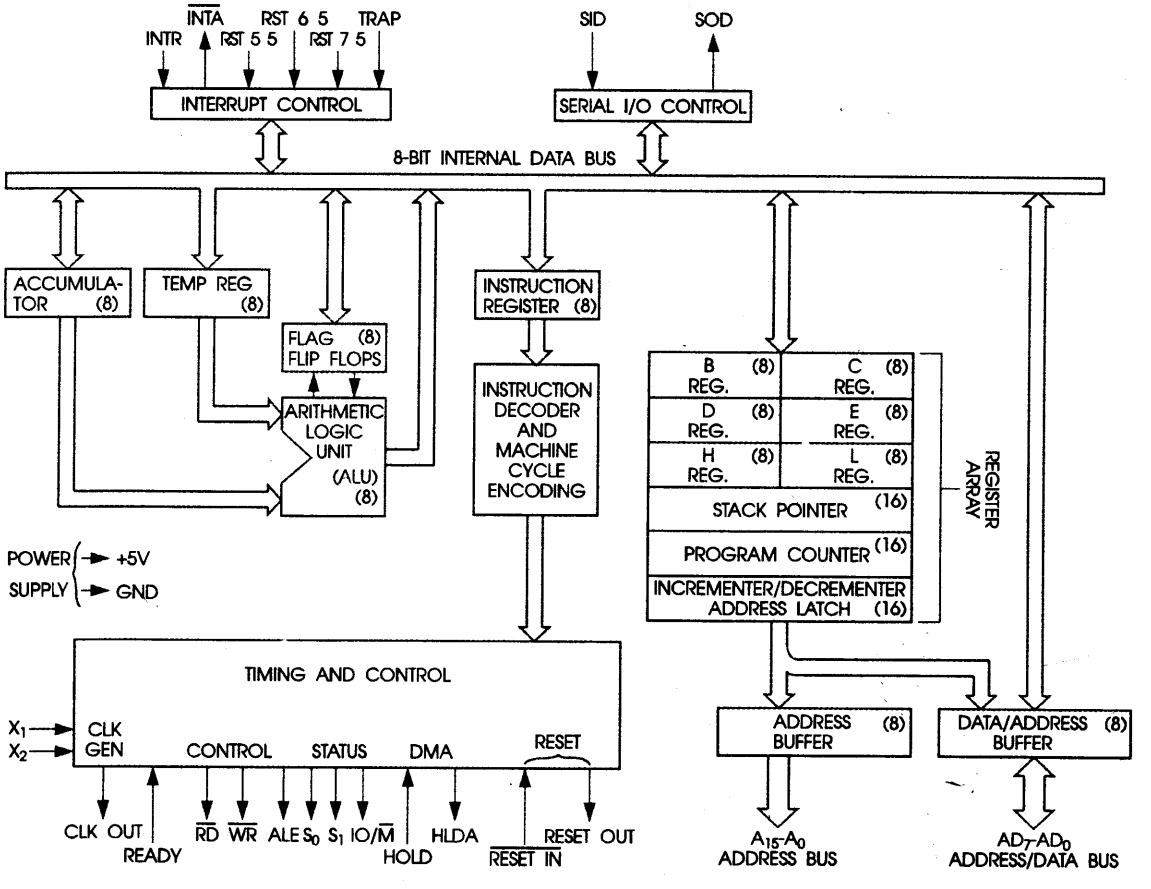


Fig. 1.3. 8085A CPU functional block diagram.

1.1.12 CONTROL SIGNALS

Now let us discuss the various control signals of microprocessor 8085 and other associated circuitry in timing and control unit.

(*i*) **X_1 and X_2** : These two pins are connected outside to a crystal. The clock generation circuitary is built inside the unit. The microprocessor operates at half of the crystal frequency. The clock generation circuitry which generates clock is fed to a simple flip-flop (Divide by two) to obtain 50% duty cycle. Thus the frequency at which microprocessor operates is half of the crystal frequency. The operation inside the microprocessor takes place at both rising and falling edge of clock, hence the duty cycle required is 50%. Crystal for 8085 used is of 6.144 MHz maximum thus the clock period is 320 nano seconds for operational frequency of 3.072 MHz (crystal frequency divided by two).

(*ii*) **Clock Out :** The operational clock frequency is also provided at its one of the pin so as to synchronize other devices (Memory or I/O) with microprocessor. This will be 3.072 MHz for 8085.

(*iii*) **ALE (Address Latch Enable) :** The multiplexed address and data lines (AD_0–AD_7) has to be demultiplexed for the purpose of separately connecting address and data lines to devices

(memory or I/O). For this purpose AD_0 to AD_7 (8 lines) are to be connected to an 8 bit latch (IC 8212 used in output mode) and address is latched on to the latch by the enable signal ALE from microprocessor. This signal is issued by 8085 in the first clock period of every machine cycle when address appears on AD_0–AD_7 lines.

(*iv*) $\overline{\text{RD}}$ **(Read Bar) :** This is an active low signal to be connected to memories read input (output Enable input of memories) or to I/O read input to enable the input-output buffer. This signal should be active for $1\frac{1}{2}$ clock states (480 ns). Access time is the time taken by memory to give out its data after addressing it.

(*v*) $\overline{\text{WR}}$ **(Write Bar) :** It has the function similar to $\overline{\text{RD}}$ but for writing data to devices. $\overline{\text{WR}}$ signal should also be active for $1\frac{1}{2}$ clock states (480 ns).

(*vi*) **IO/$\overline{\text{M}}$:** This signal distinguishes that the addresses or data is meant for either I/O devices or Memory. Whenever IO/$\overline{\text{M}}$ is 1, microprocessor is communicating to I/O devices and when IO/$\overline{\text{M}}$ is 0, it is communicating with memory units.

(*vii*) **S0, S1 (Status signals) :** These two signals along with IO/$\overline{\text{M}}$ indicates on these lines which machine cycle is currently in progress. Different machine cycles are memory Read, Memory write, I/O Read, I/O Write etc. The status of these lines for these machine cycles is shown in a table below :

MACHINE CYCLE	IO/$\overline{\text{M}}$	S1	S0	
OP CODE FETCH	0	1	1	
MEMORY READ	0	1	0	
MEMORY WRITE	0	0	1	
I/O READ	1	1	0	
I/O WRITE	1	0	1	
INTR ACK	1	1	1	
BUS IDLE	T_S	0	0	Halt
	T_S	X	X	hold, Reset

X = unspecified

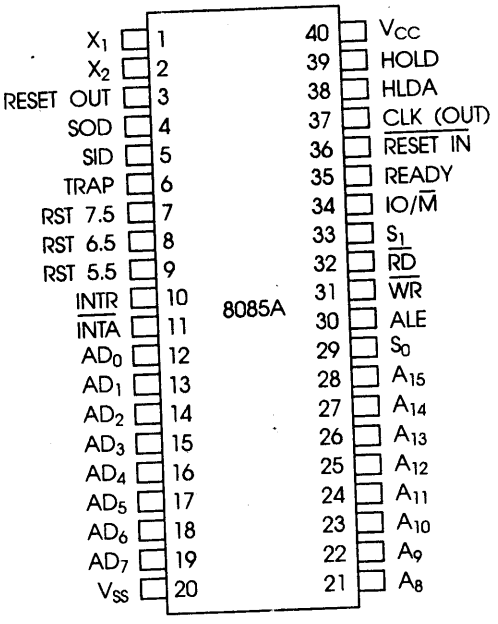
(*viii*) **Hold and HLDA :** These signals are used for DMA (Direct Memory Access) operation.

What is DMA ?

In a microcomputer system, the data transfer between I/O devices and memory will take place through microprocessor. If we are able to transfer data directly from an I/O device to memory without microprocessor getting involved is called as Direct Memory Access. DMA will save the time and hence the through put of the system.

It is possible by using above lines to hold the microprocessor and carry out DMA operation whenever hold line is made active, CPU temporarily suspends its operation and allows DMA operation. It reserves its job on negating the Hold line signal.

(*ix*) **RESET IN and RESET OUT :** When the power to the microprocessor is switched-on, the contents of program counter are not guaranteed to make microprocessor run from 0000h location. It is necessary to reset which is achieved by applying Reset in signal to microprocessor (active low signal). In response to reset in, microprocessor provides reset out signal which could be used by all external hardware. Reset in also disables the interrupt system.



(*x*) **READY :** Ready line is used to slow down microprocessor if it is interfaced with slower memories or memories with higher access time. If this line is active microprocessor will go into wait state and will be in wait state till this line is negated.

1.2 PIN DESCRIPTION

Symbol	Type	Name and Function
A_8-A_{15}	O	**Address Bus :** The most significant 8 bits of the memory address or the 8 bits of the I/O address. 3-stated during Hold and Halt modes and during RESET.
AD_{0-7}	I/O	**Multiplexed Address/Data Bus :** Lower 8 bits of the memory address (or I/O address) appear on the bus during the first clock cycle (T state) of a machine cycle. It then becomes the data bus during the second and third clock cycles.

Symbol	Type	Name and Function
ALE	O	**Address Latch Enable :** It occurs during the first clock state of a machine cycle and enables the address to get latched into the on-chip latch of peripherals. The falling edge of ALE is set to gurantee setup and hold times for the address information. The falling edge of ALE can also be used to strobe the status information. ALE is never 3-stated.
S_0, S_1 and IO/$\overline{M}$	O	**Machine Cycle Status :** **IO/$\overline{M}$ S_1 S_0 Status** 0 0 1 Memory write 0 1 0 Memory read 1 0 1 I/O write 1 1 0 I/O read 0 1 1 Opcode fetch 1 1 1 Interrupt Acknowledge . 0 0 Halt . X X Hold . X X Reset . = 3-state (high impedance) X = unspecified S_1 can be used as an advanced R/$\overline{W}$ status. IO/$\overline{M}$. S_0 and S_1 become valid at the beginning of a machine cycle and remain stable throughout the cycle. The falling edge of ALE may be used to latch the state of these lines.
$\overline{RD}$	O	**Read Control** : A low level on $\overline{RD}$ indicates the selected memory or I/O device is to be read and that the Data Bus is available for the data transfer, 3-stated during Hold and halt modes and during RESET.
$\overline{WR}$	O	**Write Control** : A low level on $\overline{WR}$ indicates the data on the Data Bus is to be written into the selected memory or I/O location. Data is set up at the trailing edge of $\overline{WR}$. 3-stated during Hold and Halt modes and during RESET.
READY	I	**Ready :** If READY is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If READY is low, the CPU will wait an integral number of clock cycles for READY to go high before completing the read or write cycle. READY must conform to specified setup and hold times.

Symbol	Type	Name and Function
HOLD	I	**Hold :** Indicates that another master is requesting the use of the address and data buses. The CPU, upon receiving the hold request, will relinquish the use of the bus as soon as the completion of the current bus transfer. Internal processing can continue. The processor can regain the bus only after the HOLD is removed. When the HOLD is acknowledged, the Address, Data $\overline{RD}$, $\overline{WR}$, and IO/$\overline{M}$ lines are 3-stated.
HLDA	O	**Hold Acknowledge :** Indicates that the CPU has received the HOLD request and that it will relinquish the bus in the next clock cycle. HLDA goes low after the Hold request is removed. The CPU takes the bus one half clock cycle after HLDA goes low.
INTR	I	**Interrupt Request :** Is used as a general purpose interrupt. It is sampled only during the next to the last clock cycle of an instruction and during Hold and Halt states. If it is active, the Program Counter (PC) will be inhibited from incrementing and an $\overline{INTA}$ will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.
$\overline{INTA}$	O	**Interrupt Acknowledge :** Is used instead of (and has the same timing as) $\overline{RD}$ during the Instruction cycle after an INTR is accepted. It can be used to activate an 8259A interrupt chip or some other interrupt port.
RST 5.5 RST 6.5 RST 7.5	I	**Restart Interrupts :** These three inputs have the same timing as INTR except they cause an internal RESTART to be automatically inserted. The priority of these interrupts is ordered as shown in Table 1.1. These interrupts have a higher priority than INTR. In addition, they may be individually masked out using the SIM instruction.
TRAP	I	**Trap** : Trap interrupt is a non-maskable RESTART interrupt. It is recognized at the same time as INTR or RST 5.5–7.5. It is unaffected by any mask or interrupt Enable. It has the highest priority of any interrupt. (See Table 1.1).

Symbol	Type	Name and Function
$\overline{\text{RESET IN}}$	I	**Reset In :** Sets the Program Counter to zero and resets the interrupt Enable and HLDA flip-flops. The data and address buses and the control lines are 3-stated during RESET and because of the asynchronous nature of RESET, the processor's internal registers and flags may be altered by RESET, with unpredictable results. $\overline{\text{RESET IN}}$ is a Schmitt-triggered input, allowing connection to an R–C network for power-on RESET delay (see Figure 1.4). Upon power-up, $\overline{\text{RESET IN}}$ must remain low for at least 10 ms after minimum V_{CC} has been reached. For proper reset operation after the power-up duration, $\overline{\text{RESET IN}}$ should be kept low a minimum of three clock periods. The CPU is held in the reset condition as long as $\overline{\text{RESET IN}}$ is applied.
RESET OUT	O	**Reset Out :** Reset Out indicates CPU is being reset. Can be used as a system reset. The signal is synchronized to the processor clock and lasts an integral number of clock periods.
X_1, X_2	I	**X_1 and X_2 :** Are connected to a crystal, LC, or RC network to drive the internal clock generator. X_1 can also be an external clock input from a logic gate. The input frequency is divided by 2 to give the processor's internal operating frequency.
CLK	O	**Clock :** Clock output for use as a system clock. The period of CLK is twice the X_1, X_2 input period.
SID	I	**Serial Input Data Line :** The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.
SOD	O	**Serial Output Data Line :** The output SOD is set or reset as specified by the SIM instruction.
V_{CC}		**Power :** +5 volt supply.
V_{SS}		**Ground :** Reference.

Table 1.1 : Interrupt Priority, Restart Address, and Sensitivity

Name	Priority	Address Branched To (1) When Interrupt Occurs	Type Trigger
TRAP	1	24H	Rising edge AND high level until sampled.
RST 7.5	2	3CH	Rising edge (latched).
RST 6.5	3	34H	High level until sampled.
RST 5.5	4	2CH	High level until sampled.
INTR	5	See Note (2)	High level until sampled.

Notes : 1. The processor pushes the PC on the stack before branching to the indicated address.

2. The address branched to depends on the instruction provided to the CPU when the interrupt is acknowledged.

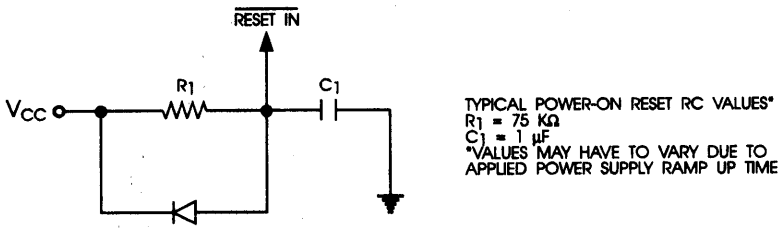


Fig. 1.4. Power-on reset circuit.

1.3 FUNCTIONAL DESCRIPTION

The 8085AH is a complete 8-bit parallel central processor. It is designed with N-channel, depletion load, silicon gate technology (HMOS), and requires a single +5 volt supply. Its basic clock speed is 3 MHz (8085AH), 5 MHz (8085AH-2), or 6 MHz (8085AH-1), thus improving on the persent 8080A's performance with higher system speed. Also it is designed to fit into a minimum system of three IC's : The CPU (8085AH), a RAM/IO (8156H), and a ROM or EPROM/IO chip (8355 or 8755A).

The 8085AH has twelve addressable 8-bit registers. Four of them can fuction only as two 16-bit register pairs. Six other can be used interchangeably as 8-bit registers or as 16-bit register pairs. The 8085AH register set is as follows :

Mnemonic	Register	Contents
ACC or A	Accumulator	8 bit 8
PC	Program Counter	16-bit address

BC, DE, HL	General-Purpose Registers; data pointer (HL)	8 bits × 6 or 16 bits × 3
SP	Stack Pointer	16-bit address
Flag or F	Flag Register	5 flags (8-bit space)

The 8085AH uses a multiplexed Data Bus. The address is split between the higher 8-bit Address Bus and the lower 8-bit Address/Data Bus. During the first T state (clock cycle) of a machine cycle the low order address is sent out on the Address/Data bus. These lower 8 bits may be latched externally by the Address Latch Enable signal (ALE). During the rest of the machine cycle the data bus is used for memory or I/O data.

The 8085AH provides $\overline{RD}$, $\overline{WR}$, S_0, S_1 and IO/$\overline{M}$ signals for bus control. An Interrupt Acknowledge signal ($\overline{INTA}$) is also provided. HOLD and all Interrupts are synchronized with the processor's internal clock. The 8085AH also provides Serial Input Data (SID) and Serial Output Data (SOD) lines for simple serial interface.

In addition to these features, the 8085AH has three maskable, vector interrupt pins, one non-maskable TRAP interrupt, and a bus vectored interrupt, INTR.

1.3.1 INTERRUPT AND SERIAL I/O

The 8085AH has 5 interrupt inputs : INTR, RST 5.5, RST 6.5, RST 7.5, and TRAP. INTR is identical in function to the 8080A INT. Each of the three RESTART inputs 5.5, 6.5 and 7.5, has a programmable mask. TRAP is also a RESTART interrupt but it is non-maskable.

The three maskable interrupts cause the internal execution of RESTART (saving the program counter in the stack and branching to the RESTART address) if the interrupts are enabled and if the interrupt mask is not set. The non-maskable TRAP causes the internal execution of a RESTART vector independent of the state of the interrupt enable or masks. (See Table 1.1)

There are two different types of inputs in the restart interrupts. RST 5.5 and RST 6.5 are *high level-sensitive* INTR (and INT on the 8080) and are recognized with the same timing as INTR. RST 7.5 is *rising edge-sensitive.*

For RST 7.5, only a pulse is required to set an internal flip-flop which generates the internal interrupt request (a normally high level signal with a low going pulse is recommended for highest system noise immunity). The RST 7.5 request flip-flop remains set until the request is serviced. Then it is reset automatically. This flip-flop may also be reset by using the SIM instruction or by issuing a $\overline{RESET\ IN}$ to the 8085AH. The RST 7.5 internal flip-flop will be set by a pulse on the RST 7.5 pin even when the RST 7.5 interrupt is masked out.

The status of the three RST interrupt masks can only be affected by the SIM instruction and $\overline{RESET\ IN}$.

The interrupts are arranged in a fixed priority that determines which interrupt is to be recognized if more than one is pending as follows : TRAP—highest priority, RST 7.5, RST 6.5, RST 5.5, INTR—lowest priority. This priority scheme does not take into account the priority of a routine that was started by a higher priority interrupt. RST 5.5 can interrupt an RST 7.5 routine if the interrupts are re-enabled before the end of the RST 7.5 routine.

The TRAP interrupt is useful for catastrophic events such as power failure or bus error. The TRAP input is recognized just as any other interrupt but has the highest priority. It is not affected by any flag or mask. The TRAP input is both *edge and level sensitive.* The TRAP input must go high and remain high until it is acknowledged. It will not be recognized again until it goes, low then high again. This avoids any false triggering due to noise or logic glitches. Figure 1.5 illustrates the TRAP interrupt

request circuitry within the 8085AH. Note that the servicing of any interrupt (TRAP, RST 7.5, RST 6.5, RST 5.5, INTR) disables all future interrupts (except TRAPs) until an EI instruction is executed.

The TRAP interrupt is special in that it disables interrupts, but preserves the previous interrupt enable status. Perfoming the first RIM instruction following a TRAP interrupt allows you to determine whether interrupts were enabled or disabled prior to the trap. All subsequent RIM instructions provide current interrupt enable status. Performing a RIM instruction following INTR, or RST 5.5–7.5 will provide current Interrupt Enable status, revealing that Interrupts are disabled.

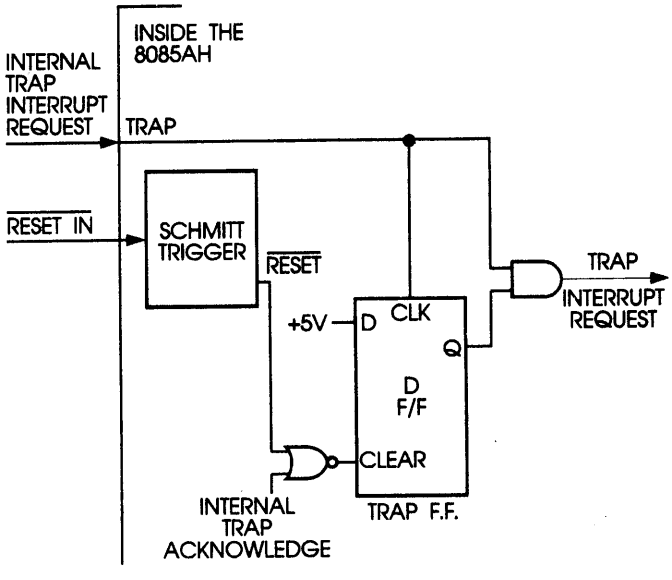


Fig. 1.5. TRAP and RESET IN circuit.

The serial I/O system is also controlled by the RIM and SIM instructions. SID is read by RIM, and SIM sets the SOD data.

1.3.2 DRIVING THE X_1 AND X_2 INPUTS

You may drive the clock inputs of the 8085AH, 8085AH-2, or 8085AH-1 with a crystal, an LC tuned circuit, an RC network, or an external clock source. The crystal frequency must be at least 1 MHz, and must be twice the desired internal clock frequency; hence, the 8085AH is operated with a 6 MHz crystal (for 3 MHz clock), the 8085AH-2 operated with a 10 MHz crystal (for 5 MHz clock), and the 8085AH-1 can be operated with a 12 MHz crystal (for 6 MHz clock). If a crystal is used, it must have the following characteristics :

Parallel resonance at twice the clock frequency desired

C_L (load capacitance) $\leq$ 30 pF
C_S (shunt capacitance) $\leq$ 7 pF
R_S (equivalent shunt resistance) $\leq$ 75 Ohms
Drive level : 10 mW
Frequency tolerance : $\pm$ .005% (suggested)

Note the use of the 20 pF capacitor between X_2 and ground. This capacitor is required with crystal frequencies below 4 MHz to assure oscillator start up at the correct frequency. A parallel-resonant LC

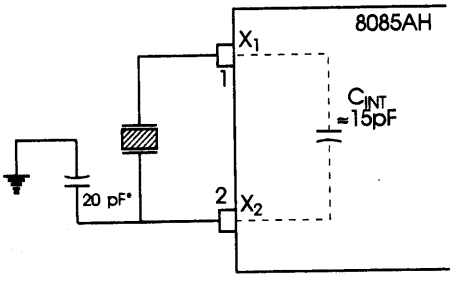
circuit may be used as the frequency-determining network for the 8085AH, providing that its frequency tolerance of approximately ± 10% is acceptable. The components are chosen from the formula :

$$f = \frac{1}{2\pi \sqrt{L(C_{ext} + C_{int})}}$$

The minimize variations in frequency, it is recomended that you choose a value for C_{ext} that is at least twice that of C_{int}, or 30 pF. The use of an LC circuit is not recommended for frequencies higher than approximately 5 MHz.

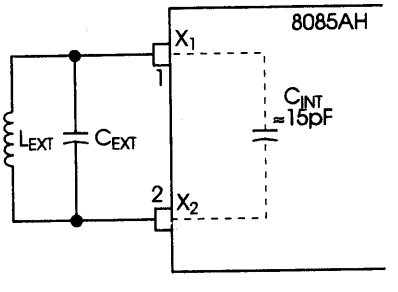
An RC circuit may be used as the frequency-determining network for the 8085AH if maintaining a precise clock frequency is of no importance. Variations in the on-chip timing generating can cause a wide variation in frequency when using the RC mode. Its advantage is its low component cost. The driving frequency generated by the circuit shown is approximately 3 MHz. It is not recommended that frequencies greatly higher or lower than this be attempted.

Figure 1.6 shows the recommended clock driver circuits. Note in D and E that pull up resistors

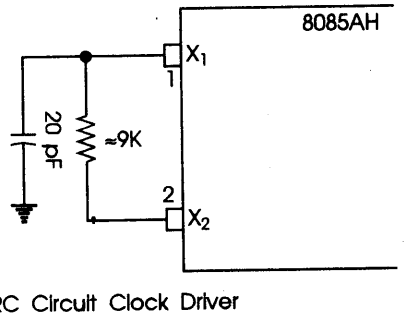


*20 pF CAPACITORS REQUIRED FOR CRYSTAL FREQUENCY < 4 MHz ONLY.

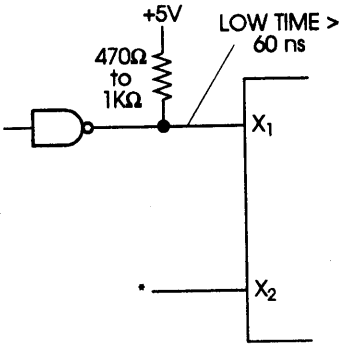
a. Quartz Crystal Clock Driver



b. LC Tuned Circuit Clock Driver

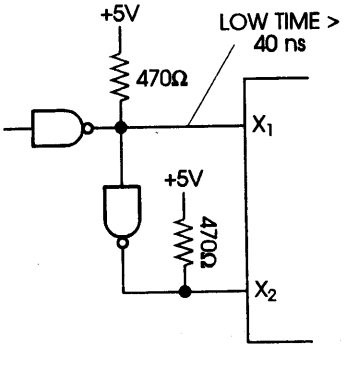


c. RC Circuit Clock Driver



*X_2 LEFT FLOATING

d. 1-6 MHZ Input Frequency External Clock Driver Circuit



e. 1-12 MHZ Input Frequency External Clock Driver Circuit

Fig. 1.6. Clock driver circuits

are required to assure that the high level voltage of the input is at least 4V and maximum low level voltage of 0.8V.

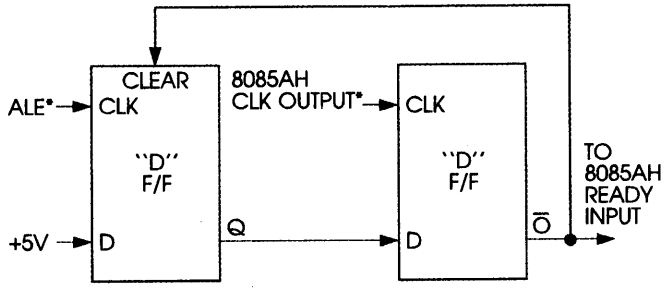
For driving frequencies up to and including 6 MHz you may supply the driving signal to X_1 and leave X_2 open-circuited (Figure 1.6D). If the driving frequency is from 6 MHz to 12 MHz, stability of the clock generator will be improved by driving both X_1 and X_2 with a push-pull source (Figure 1.6E). To prevent self-oscillation of the 8085AH, be sure that X_2 is not coupled back to X_1 through the driving circuit.

1.3.3 GENERATING AN 8085AH WAIT STATE

If your system requirements are such that slow memories or peripheral devices are being used, the circuit shown in Figure 1.7 may be used to insert one WAIT in each 8085AH machine cycle.

The D flip-flops should be chosen so that

- CLK is rising edge-triggered
- CLEAR is low-level active.



*ALE AND CLK (OUT) SHOULD BE BUFFERED IF CLK INPUT OF LATCH EXCEEDS 8085AH IOL OR IOH.

Fig. 1.7. Generation of a wait state for 8085AH CPU.

1.4 HOW THE MICRO COMPUTER WORKS ?

The execution of any program consists of a sequence of READ and WRITE operations, of which each operation transfers a byte of data between a microprocessor and a particular memory location or I/O addresses. These read and write operations are the only communication between the processor and other components and are all that are necessary to execute any instruction or program.

Each read or write operation of 8085 is called as Machine cycle and execution of any instruction consists of execution of number of these machine cycles varying from 1 to 5 for 8085. Each machine cycle normally consists of 3 to 6 clock states (also called as T states).

Consider an instruction STA 2000 (Store ACC at 2000) (Figure 1.8.) This instruction will cause the storage of accumulator contents at the direct address specified (at 2000) in second and third byte.

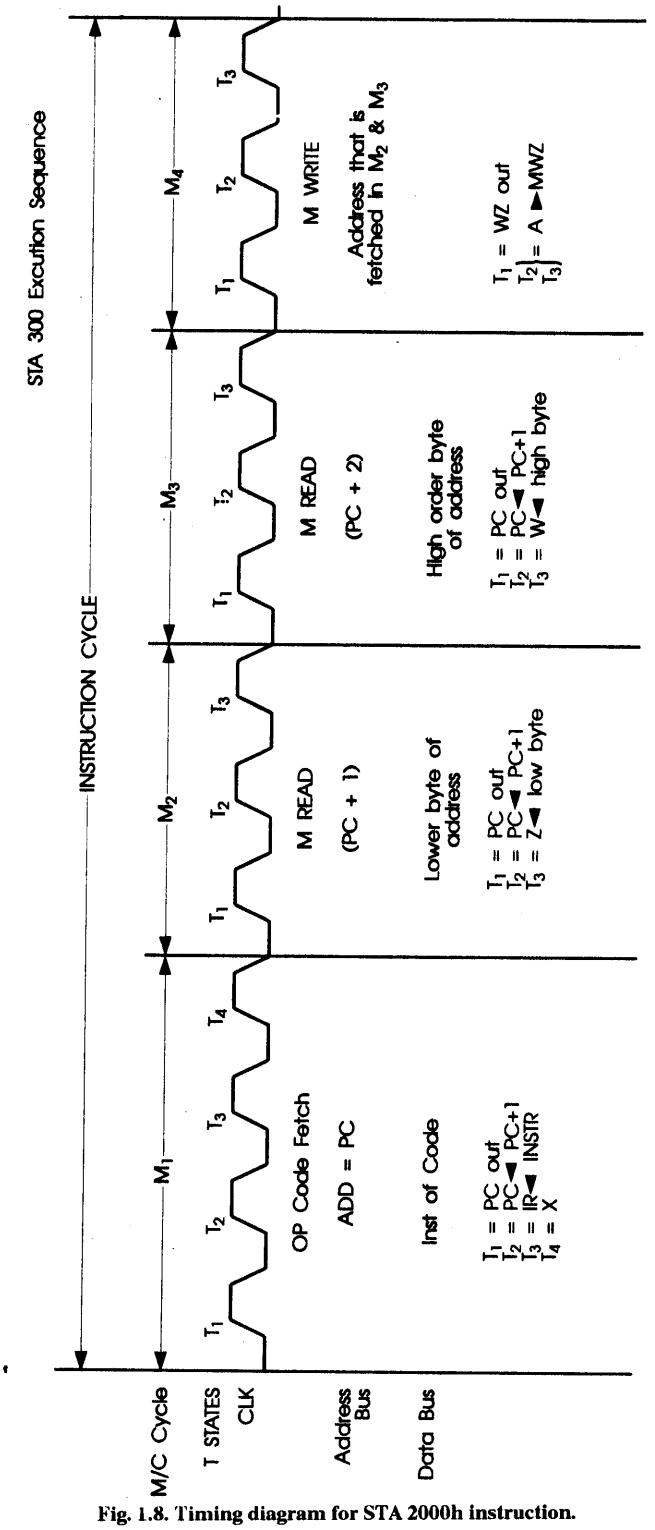


Fig. 1.8. Timing diagram for STA 2000h instruction.

During the first machine cycle (which is always opcode fetch cycle), the CPU puts contents of program counter (PC) on address bus and performs memory read operation to read opcode of STA from the memory. The opcode fetch takes 3 clock periods and in the fourth clock period of M_1, CPU interprets the opcode and interprets its meaning, which in this case is STA instruction. At this point CPU knows that it should perform three more machine cycles. Out of these three cycles two will be memory read to read the address at which data is to be stored and the third cycle will be memory write cycle.

The 8085 then increments the program counter so that it points to the next byte of instruction and performs a memory Read machine cycle (M_2) at address (PC + 1). The accessed memory places data (2nd byte = 00H) on data lines and then CPU stores this number which is low order byte of direct address in temporary registers internally to CPU. Program counter is incremented. It is now pointing to (PC + 2) place and reads the memory (M_3). This data is high order byte of direct address.

At this moment 8085 has accessed all three bytes of the STA instruction, which it must now execute. The execution consists of placing the data accessed in M_2 and M_3 on address bus and then performing a memory write (M_4). On finishing of M_4, the CPU fetches M_1, the first byte of next instruction and so on. The schematic illustrates the above operation.

1.4.1 MACHINE CYCLE TIMING

The memory read or memory write operations need only three clock periods (T_1 T_2 and T_3 only). The opcode fetch machine cycle takes 4 to 6 clock periods. Most of the instructions will consists of opcode fetch of 4 clock periods (T_1 T_2 T_3 T_4) and in few cases it will also have (T_5 and T_6). Thus the first three states of memory read and opcode fetch will remain same. Opcode fetch cycle will have extra states (T_4 or T_4 T_5 T_6). The timings which are valid for memory read cycle will remain same for T_1 T_2 and T_3 states of Opcode fetch cycle.

1.4.2 OPCODE FETCH CYCLE

The timing diagram for DCX instructions is shown in Fig. 1.9. This instruction has 6 clock states T_1, T_2, T_3, T_4, T_5 and T_6.

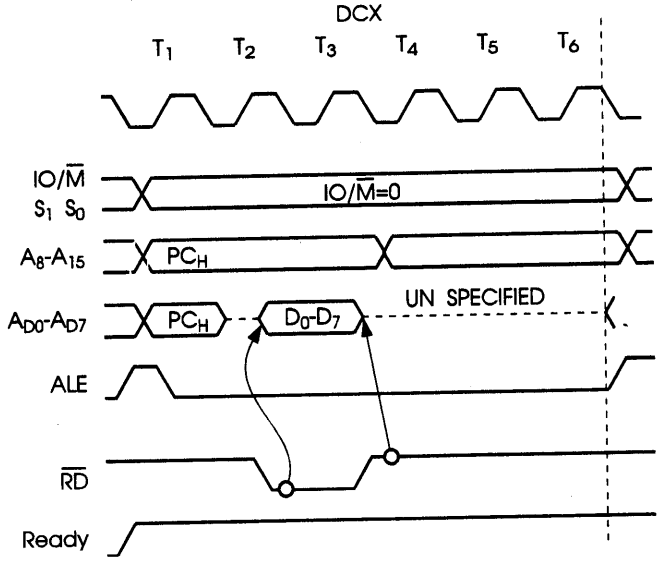


Fig. 1.9. Timing diagram for DCX H instruction.

(*i*) At the beginning of every machine cycle in T_1, CPU sends out three status signals ($IO/\overline{M}$, S_1, S0) to indicate which machine cycle is going to take place. For Opcode fetch these signals will be $IO/\overline{M} = 0$ $S_1 = S_0 = 1$.

(*ii*) At the begining of every machine cycle, CPU also sends out a 16 bit address to specify the particular memory location or port number which is going to be read.

(*iii*) In this case contents of PC are placed on address bus. PC low order byte is placed on AD_0 – AD_7 lines. The information on AD_0–AD_7 lines is present only for T_1 duration and disappear in T_2. Hence this information must be latched on a latch (IC 8212) using Address Latch Enable (ALE) signal provided by microprocessor so that Address A_0–A_7 is available for rest of the period of machine cycles. Higher order byte A_8–A_{15} appears on beginning of T_1 and is present for full duration of machine cycle *i.e.*, for T_1 T_2 T_3 and T_4.

(*iv*) After this the drivers on lines AD_0–AD_7 are disabled and CPU provides a low level on $\overline{RD}$ line in T_2 to enable the addressed memory device. And on low to high transitions of clock during T_2, memory starts driving and places data on data bus.

(*v*) After certain period of time (Access time of memory) memory will put data on data bus. 8085 during T_3 will load this data into its instruction register and then negate $\overline{RD}$ signal. $\overline{RD}$ signal remains active for about 1.5 T which is approx 480 nsecs. Thus access time of memory should be around 450 n sec. (30 n sec. margin) and data bus enter into tristate.

(*vi*) Since it is the M_1 cycle (it needs T_1 T_2 T_3 T_4) microprocessor will automatically enter into T_4 and during decoding of T_4, CPU will decide whether to enter into T_5 or start a new machine cycle. For DCX instruction it will enter into T_5–T_6 before going to T_1 of next machine cycle.

A_8–A_{15} lines may change during T_4–T_6 and it is most important that all I/O devices and memory should qualify with $\overline{RD}$ signal. If they don't qualify $\overline{RD}$ they may be spuriously selected. The generation of spurious addresses may also occur during transitional periods in T_1. Therefore, selection of all devices must qualify with $\overline{RD}$ or $\overline{WR}$.

1.4.3 OPCODE FETCH WITH READY LINE

Now during T_2 CPU monitors Ready line, if found low, CPU will enter into Twait state and remain in Twait indefinitely until Ready line goes high. Then CPU will exit out and enter in T_3 state. This stretching effect allows us to interface memory of higher access time (>450) to be interfaced with with 8085. Another use of Ready line is in single stepping.

1.4.4 MEMORY READ CYCLE

The timing duration T_1 . T_3 of opcode fetch machine cycle and Memory Read is same. The important differences between two are :—

(*i*) The status will be as for MR cycle *i.e.*, $IO/\overline{M} = 0$, $S_1 = 1$, $S_0 = 0$.

(*ii*) After completion of T_3 it will enter into T_1 in case of Memory Read, where as it will enter into T_4 in case of opcode fetch.

(*iii*) The memory used in opcode Fetch cycle is always the contents of program counter and fetched data is placed in instruction Register where as the address may have different data in case of memory read and data will go into one of the general purpose register and not into instruction register.

1.4.5 MEMORY WRITE CYCLE

It also takes three clock cycles T_1, T_2 and T_3.

Now in this case CPU first transmits address on AD_0–AD_7 lines and then on the same line it transmits data, here there is no need for switching off the CPU transmitters.

(*a*) In T_1, CPU sends out status signals, address on address bus and ALE signal is issued to latch address on external latch.

(*b*) In T_2, in memory read cycle transmitters are disabled. Here it is not needed to switch off transmitters and CPU floats data on AD_0 to AD_7 lines and this data remains on these lines till end of T_3. $\overline{WR}$ signal is issued during this period to enable the writing of addressed memory location.

(*c*) During T_3 $\overline{WR}$ is deactivated indicating the termination of write cycle.

Ready line will have same effect as explained earlier in Memory write.

During T_2 Ready line will be sampled if found low, processor will enter into Twait state. The moment Ready line goes high, processor will enter into T_3. Memory write cycle will consists of T_1, T_2, T_3 if ready line is not pulled low during T_2 and will consists of T_1, T_2, Twait and T_3 if ready line is pulled low as shown in diagram (Fig. 1.10).

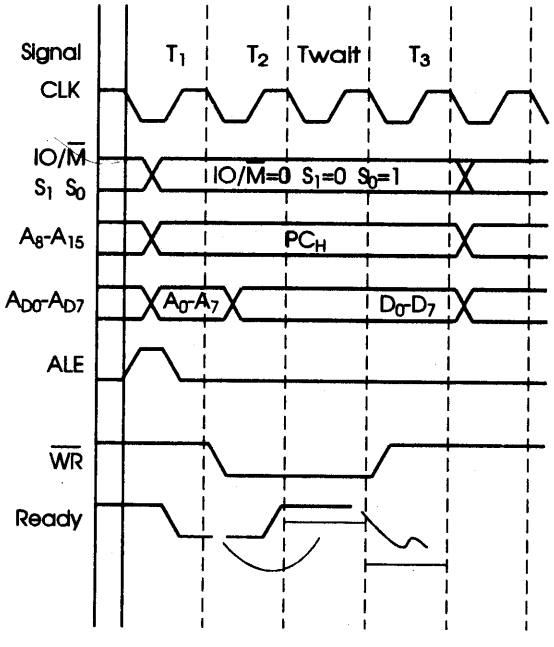


Fig. 1.10.