Quantum Series®
ENGINEERING

QUANTUM *Series*

Semester - 4 Computer Science & IT

Microprocessor

Session 2019-20 Even Semester

- Topic-wise coverage of entire syllabus in Question-Answer form.
- Short Questions (2 Marks)

Includes solution of following AKTU Question Papers:
2014-15 • 2015-16 • 2016-17 • 2017-18 • 2018-19

Scanned by TapScanner

CONTENTS

KCS 403 : Microprocessor

UNIT-1 : BASICS **(1–1 C to 1–17 C)**

Microprocessor evolution and types, microprocessor architecture and operation of its components, addressing modes, interrupts, data transfer schemes, instruction and data flow, timer and timing diagram, Interfacing devices.

UNIT-2 : 8085 MICROPROCESSOR **(2–1 C to 2–29 C)**

Pin diagram and internal architecture of 8085 microprocessor, registers, ALU, Control & status, interrupt and machine cycle. Instruction sets. Addressing modes. Instruction formats Instruction Classification: data transfer, arithmetic operations, logical operations, branching operations, machine control and assembler directives.

UNIT-3 : 8086 MICROPROCESSOR **(3–1 C to 3–29 C)**

Architecture of 8086 microprocessor: register organization, bus interface unit, execution unit, memory addressing, and memory segmentation. Operating modes. Instruction sets, instruction format, Types of instructions. Interrupts: hardware and software interrupts.

UNIT-4 : ASSEMBLY LANGUAGE PROGRAMMING **(4–1 C to 4–19 C)**

Assembly language programming based on intel 8085/8086. Instructions, data transfer, arithmetic, logic, branch operations, looping, counting, indexing, programming techniques, counters and time delays, stacks and subroutines, conditional call and return instructions.

UNIT-5 : PERIPHERAL DEVICES **(5–1 C to 5–35 C)**

Peripheral Devices: 8237 DMA Controller, 8255 programmable peripheral interface, 8253/8254 programmable timer/counter, 8259 programmable interrupt controller, 8251 USART and RS232C.

SHORT QUESTIONS **(SQ-1C to SQ-19C)**

SOLVED PAPERS (2014-15 TO 2018-19) **(SP-1C to SP-25C)**

Scanned by TapScanner

1 UNIT

Basics

CONTENTS

Part-1 : Microprocessor Evolution and Types, Microprocessor Architecture and Operation of its Components ..... **1-2C to 1-7C**

Part-2 : Addressing Modes, Interrupts, Data Transfer Scheme and Instructions ..... **1-8C to 1-12C**

Part-3 : Data Flow, Timer, Timing Diagram and Interfacing Devices ..... **1-12C to 1-16C**

PART-1

Microprocessor Evolution and Types, Microprocessor Architecture and Operation of Its Components.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.1. **What do you understand by microprocessor ?**

Answer

1. The microprocessor is a programmable integrated device that has computing and decision-making capability similar to that of central processing unit (CPU).
2. A microprocessor is designed to perform arithmetic and logic operations that make use of small number-holding areas called registers.
3. Typical microprocessor operations include adding, subtracting, comparing two numbers and fetching numbers from one area to another.
4. A microprocessor typically serves as a central processing unit microprocessor in a computer system.

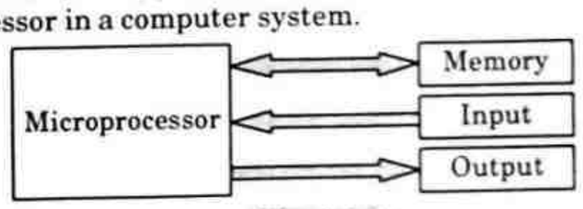


Fig. 1.1.1.

Que 1.2. **Write a short note on evolution of the microprocessor.**

OR

Elaborate the evolution of microprocessor from a 4-bit processor to the contemporary dual core processors.

AKTU 2018-19, Marks 07

Answer

A. 1st generation (1971-1973) :

1. Intel corporation introduced the first microprocessor, the 4004, in 1971. The 4004 evolved from a development effort while designing a calculator chip set.

Scanned by TapScanner

2. Soon after the 4004 appeared in the commercial market, three other microprocessors were introduced. These were the Rockwell International PPS-4, the Intel 8-bit 8008, and the National Semiconductor 16-bit IMP-16.
3. The microprocessors introduced between 1971 and 1973 were the first-generation system. They were designed using the PMOS (p-type MOS) technology.

B. 2nd generation (1973-1978) :

1. After 1973, second-generation microprocessors such as Motorola 6800 and 6809, Intel 8085, and Zilog Z80 evolved.
2. These processors were fabricated using the NMOS (n-type MOS) technology.

C. 3rd generation (1979-1980) :

1. After 1978, the third-generation microprocessors were introduced. These processors are 16 bits wide and include typical processors such as Intel 8086 / 80186 / 80286 and Motorola 68000 / 68010.
2. Recently, Intel utilized the HMOS technology to fabricate the 8085A.

D. 4th generation (1980-1995) :

1. In 1980, fourth-generation microprocessors evolved. Intel introduced the first commercial 32-bit microprocessor, the problematic Intel 432. This processor was eventually discontinued by Intel.
2. Since 1985, more 32-bit microprocessors have been introduced. These include Motorola's MC 68020 and 68030 and Intel 80386.
3. These processors are fabricated using the low-power version of the HMOS technology called the HCMOS.
4. Recently, Motorola has introduced a 32-bit RISC (Reduced instruction set computer) microprocessor with a simplified instruction set called the MC88100.

E. 5th generation (1995 onwards) :

1. From 1995 to until now this generation has been bringing out high-performance and high-speed processors that make use of 64-bit processors.
2. Such processors include Pentium, Celeron, Dual and Quad core processors.
3. The fifth generation microprocessors represent advancement in specifications.

Que 1.3. Explain various types of microprocessor.

Answer

A. 8085 microprocessor :

1. The 8085 CPU is the most popular CPU amongst all the 8-bit CPUs.
2. The 8085 CPU houses an on-chip clock generator and provides good performance utilizing an optimum set of registers and reasonably powerful ALU.
3. The major limitation of this 8-bit microprocessors are limited memory addressing capacity, slow speed of execution, limited number of scratchpad registers and non-availability of compiler instruction set and addressing modes.

B. 8086 microprocessor :

1. The first 16-bit CPU from Intel was a result of the designer's efforts to produce more powerful and efficient computing machine.
2. The 8086 contains a set of 16-bit general purpose registers, supports a 16-bit ALU, a rich instruction set and provides segmented memory addressing scheme.
3. The introduction of a set of segment registers for addressing the segmented memory in 8086 was indeed a major step in the process of evaluation.
4. The major limitation in 8086 was that it did not have the memory management and protection capabilities.

C. 80286 microprocessor :

1. 80286 was the first CPU to possess the ability of memory management, privilege and protection.
2. However, the 80286 CPU had a limitation on the maximum segment size supported by it.
3. Another limitation of 80286 was that, once it was switched to protected mode, it was difficult to get it back to real mode.
4. The only way of reverting it to the real mode was to reset the system.

D. 80386 microprocessor :

1. 80386 was the first 32-bit CPU from Intel.
2. The memory management capability of 80386 was enhanced to support virtual memory, paging and four levels of protection.
3. The maximum segment size in 80386 was enhanced and this could be as large as 4 GB.
4. The 80386 along with its math coprocessor 80387 provided a high speed environment.

E. 80486 microprocessor :

1. The 80486 was designed with an integrated math coprocessor.

Scanned by TapScanner

2. After getting integrated, the speed of execution of mathematical operations enhanced three folds.
3. Also for the first time, an 8 kB four-way set associative code and data cache was introduced in 80486.
4. A five stage instruction pipeline was also introduced.

F. Pentium microprocessor :

1. It has a super scalar, super pipelined architecture.
2. It has two integer pipelines U and V, where each one is a 4-stage pipeline.
3. It has an on-chip floating point unit, which has increased the floating point performance.
4. Pentium-II is the next version of Pentium.
5. It incorporates all features of Pentium-Pro and it has a large cache.
6. Pentium-III has been developed on 0.25 microtechnology and includes over 9.5 million transistors.

Que 1.4. Explain the general microprocessor architecture and operation of its components.

Answer

A. General architecture :

1. Microprocessor architecture defines suitable placement of its various functional blocks in the form of required circuitry for efficient flow of data and result from one block to another.
2. The general purpose architecture of microprocessor is shown in Fig. 1.4.1.

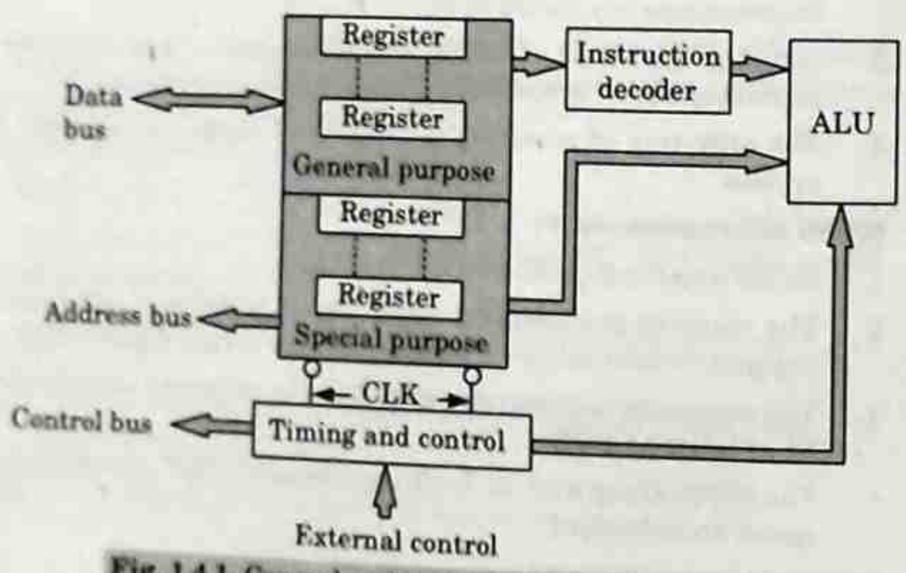


Fig. 1.4.1. General architecture of a microprocessor.

B. Architecture of microprocessor contains :

1. **ALU (Arithmetic Logical Unit) :**
   i. It consists of an adder, an accumulator, a temporary register, and a shift register and a status register.
   ii. This ALU unit performs various arithmetic and logical operation.
2. **General purpose register :**
   i. In the microprocessor, there are 8-bit general purpose register or as a 16-bit register pairs, when used in register pair mode.
   ii. These are used for both storing data as well as the address.
3. **Special purpose register :**
   i. This consists of accumulator, Program Counter (PC), Stack Pointer (SP) and status flag register.
   ii. These registers are used for some specific applications designated by the manufacturers.
4. **Instruction decoder :** This receives the contents of instruction register and develops control signals that enable data paths necessary to execute the instruction.
5. **Timing and control unit :**
   i. This unit controls and synchronizes all the operations inside and outside the microprocessor.
   ii. The timing and control signals are derived from the master clock.
   iii. The control unit also accepts the control signals generated by the other devices associated with microprocessor system.
6. **Address bus :**
   i. This is used for transmitting address information.
   ii. The address bus will normally contain 16-bits to provide for addressing and addressing capability of up to 64 kB of memory.
7. **Control bus :** This comprises of various signal lines used for carrying synchronization signals. The microprocessor uses such lines for providing timing signals.

Que 1.5. Discuss Harvard architecture of microprocessor.

Answer

A. Harvard architecture :

1. Harvard architecture uses separate memories for program and data with their independent address and data buses.
2. Because of two different streams of data and address, there is no need to have any time division multiplexing of address and data buses.

Scanned by TapScanner

3. Not only the architecture supports parallel buses for address and data, but also it allows a different internal organization such that instruction can be prefetched and decoded while multiple data are being fetched and operated on.
4. Further, the data bus may have different size than the address bus. This allows the optimal bus widths of the data and address buses for fast execution of the instruction.

B. Example :

1. MCS-51 family of microcontrollers by Intel has Harvard architecture because there are different memory spaces for program and data and separate (internal) buses for address and data.
2. PIC microcontrollers by microchip use Harvard architecture.

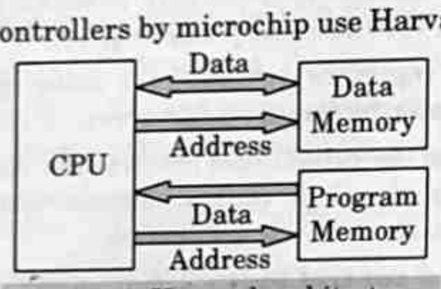


Fig. 1.5.1. Harvard architecture.

Que 1.6. **Explain Princeton architecture of microprocessor.**

Answer

A. Princeton architecture :

1. In Princeton or Von-Neumann architecture, programs and data share the same memory space.
2. Fig. 1.6.1 shows the Princeton architecture that allow storing or modifying the programs easily.
3. However, the code storage may not be optimal and requires multiple fetches to form the instruction.
4. Program and data fetches are done using time division multiplexing which affect its performance.

B. Example : Microcontroller using the Princeton architecture is Motorola 68HC11 microcontroller.

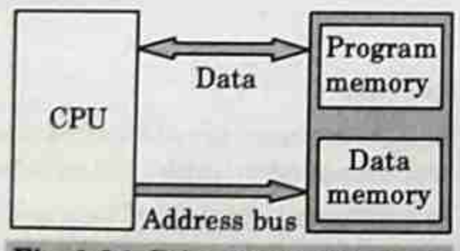


Fig. 1.6.1. Princeton architecture.

PART-2

Addressing Modes, Interrupts and Data Transfer Scheme, Instructions.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.7. **What is addressing modes? Give its types.**

Answer

A. Addressing modes : 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.

B. Addressing modes in 8085 is classified into 5 groups :

1. Immediate addressing mode
2. Register addressing mode
3. Direct addressing mode
4. Indirect addressing mode
6. Implied addressing mode

C. Addressing modes in 8086 is classified into 8 groups :

1. Immediate addressing mode
2. Register addressing mode
3. Direct addressing mode
4. Register indirect addressing mode
5. Based addressing mode
6. Indexed addressing mode
7. Based-index addressing mode
8. Based indexed with displacement mode

Que 1.8. **Define interrupt and give the interrupt pins in 8085 and 8086.**

Answer

A. Interrupt : Interrupts are the signals generated by external devices to request the microprocessor to perform a task.

Scanned by TapScanner

B. Types of interrupt in 8085 :

a. Hardware interrupts :

i. TRAP ii. RST 7.5
iii. RST 6.5 iv. RST 5.5
v. INTR

b. Software interrupts : RST 0 to RST 7.

C. Types of interrupts in 8086 :

1. Hardware Interrupt
2. Divide by Zero Interrupt (Type 0)
3. Single Step Interrupt (Type 1)
4. Software Interrupts (Type 0-255)
5. Non Maskable Interrupt (Type 2)
6. Maskable Interrupt (INTR).

Que 1.9. Explain data transfer schemes. Give its types.

Answer

A. Data transfer schemes :

1. In 8085 microprocessor based systems several input and output devices are connected. The data transfer may take place between microprocessor and memory, microprocessor and I/O devices and memory & I/O devices.
2. The main problems arise due to mismatch of the speed of the I/O devices and the speed of microprocessor or memory. To overcome this problem of speed mismatch between the microprocessor and I/O devices data transfer schemes of 8085 microprocessor is used.

B. Types : The data transfer schemes of 8085 microprocessor were categorised depending upon the capabilities of I/O devices for accepting serial or parallel data.

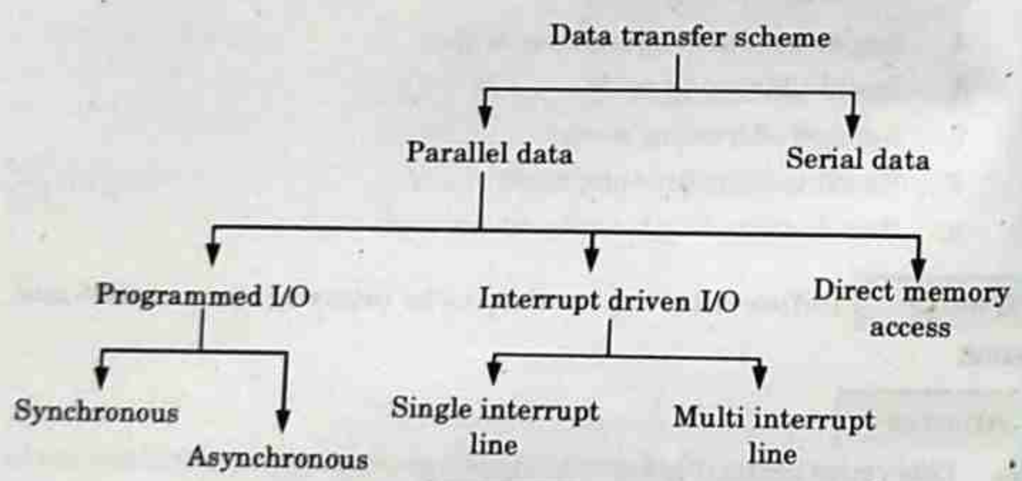


Fig. 1.9.1.

a. Serial I/O mode transfer :

1. In serial I/O mode transfer a single bit of data on a single line at a time. For serial I/O data transmission mode, 8-bit parallel word is converted to a stream of eight serial bit using parallel-to-serial converter.
2. Similarly, in serial reception of data, the microprocessor receives a stream of 8-bit one by one which are then converted to 8-bit parallel word using serial-to-parallel converter.

b. Parallel data transfer scheme :

1. Parallel data transfer scheme is faster than serial I/O transfer, in parallel data transfer 8-bit data is send all together with 8 parallel wires.
2. In 8085 microprocessor mainly three types of parallel data transfer scheme are observed. Those are
   i. Programmed I/O Data Transfer
   ii. Interrupt Driven I/O Data Transfer
   iii. Direct Memory Access (DMA) Data Transfer.
3. The 8085 microprocessor is a parallel device. That means it transfers eight bits of data simultaneously over eight data lines (parallel I/O mode).
4. However in many situations, the parallel I/O mode is either impractical or impossible.
5. For example, parallel data communication over a long distance becomes very expensive. Similarly, parallel data communication is not possible with devices such as CRT terminal or Cassette tape etc.

Que 1.10. Define instruction. Classify instructions.

Answer

A. Instruction : An instruction is a binary pattern designed to perform a specific function. The list of entire instructions is called the instruction set. The instruction set determines what function the microprocessor can perform.

B. The following notations are used in the description of the instructions :

1. R = 8085 8-bit registers (B, C, D, E, H, L)
2. M = Memory register (location) pointed by value HL
3. R_s = Register source
4. R_d = Register destination (B, C, D, E, H, L)
5. R_p = Register pair (BC, DE, HL)
6. ( ) = Contents of

Scanned by TapScanner

C. The 8085 instruction set can be classified into the following five categories :

i. Data transfer (copy) instructions :

1. These instructions perform the following six operations :

   Load 8-bit number in a register.

   Load 16-bit number in a register pair.

   Copy from register to register.

   Copy between register and memory.

   Copy between I/O and accumulator.

   Copy between registers and stack memory.

2. **Instructions :**

MVI R, 8-bit	MOV R_d, R_s
LXI R_p, 16-bit	OUT 8-bit
IN 8-bit	LDA 16-bit
STA 16-bit	LDAX R_p
STAX R_p	MOV R, M
MOV M, R	

ii. Arithmetic instructions : The frequently used arithmetic operations are :

Add, Subtract, Increment (add 1), Decrement (subtract 1)

ADD R	ADI 8-bit
ADD M	SUB R
SUI 8-bit	SUM M
INR R	INR M
DCR R	DCR M
INX R_p	DCX R

3. **Logical and bit manipulation instructions :** These instructions include the following operations :

   AND, OR, XOR, compare, rotate bits

ANA R	ANI 8-bit
ANA M	ORA R
ORI 8-bit	ORA M
XRA R	XRI 8-bit
XRA M	CMP R
CPI 8-bit	

4. **Branching instructions :** The following instructions change the program sequence :

JMP 16-bit	JZ 16-bit
JNZ 16-bit	JC 16-bit
JNC 16-bit	CALL 16-bit
RET	

5. **Miscellaneous instructions :** There are a number of instructions related with data transfer among the register, the stack operation instructions and interrupt operations of 8085 MP which are kept in this group. They are:

   PUSH, POP

   EI, DI

6. **Machine control instructions :** These instructions affect the operation of the processor.

   HLT, NOP

PART-3

Data Flow, Timer and Timing Diagram and Interfacing Devices.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.11. **Explain CPU and buses of the microprocessor.**

OR

Name the different types of buses used in Microprocessor family. Explain the need to demultiplex the bus AD_7-AD_0 with neat diagram.

AKTU 2018-19, Marks 07

Answer

A. CPU (Central processing unit) :

1. Its purpose is to fetch binary coded instruction from memory then decodes the fetched signal and generates the control signal required to execute the instructions.
2. It executes instructions, controls and directs the inputs, informs ALU (Arithmetic and logic Unit) for processing, arranges the storage and directs the data to the output device.
3. It is responsible for directing the flow of instruction and data within control signals.
4. It contains the necessary logic to interpret instructions and to generate the signals necessary for the execution of those instructions.

Scanned by TapScanner

B. System Buses : Buses are data communication path over which information is transferred a byte or word at a time. The system Bus consists of three types of buses :

i. Data Bus :

1. A type of bus that is used to transfer data from memory to CPU and I/O device to memory is called Data Bus.
2. Data bus is the bidirectional bus *i.e.*, it can communicate in two ways, but in one direction at a time.
3. It handles the transfer of data and instructions.
4. It carries data (Operands) to and from the CPU and memory as required.
5. It is also used to transfer data between memory and I/O devices during input output operations.

ii. Address Bus :

1. A type of bus that is used to carry the address (not data) is called Address Bus.
2. It carries address between memory and CPU (central processing unit) and between memory and I/O devices.
3. An address is defined as a label, symbol, or other set of characters used to designate a location or register where information is store.
4. Before data or instructions can be written into or read from memory by the CPU or I/O sections, an address must be transmitted to memory over the address bus.
5. The number of lines on the bus determines the number of addressable memory elements.

iii. Control Bus :

1. Control Bus is a type of bus that carries control instructions to run operates hardware.
2. The Control Bus is used by the CPU (central processing unit) to direct and monitor the actions of the other functional areas of the computer.
3. It is used to transmit a variety of individual signals (read, write, interrupt, acknowledge) and coordinate the operations of the computer.
4. The size of control bus is from 8 to 16 bits.

C. Need to demultiplex :

1. The signal lines $AD_0 - AD_7$ are bidirectional. They are used as the low-order address bus as well as the data bus.

2. In executing an instruction, during the earlier part of the cycle, these lines are used as the low order address bus and as the data bus during the later cycle.

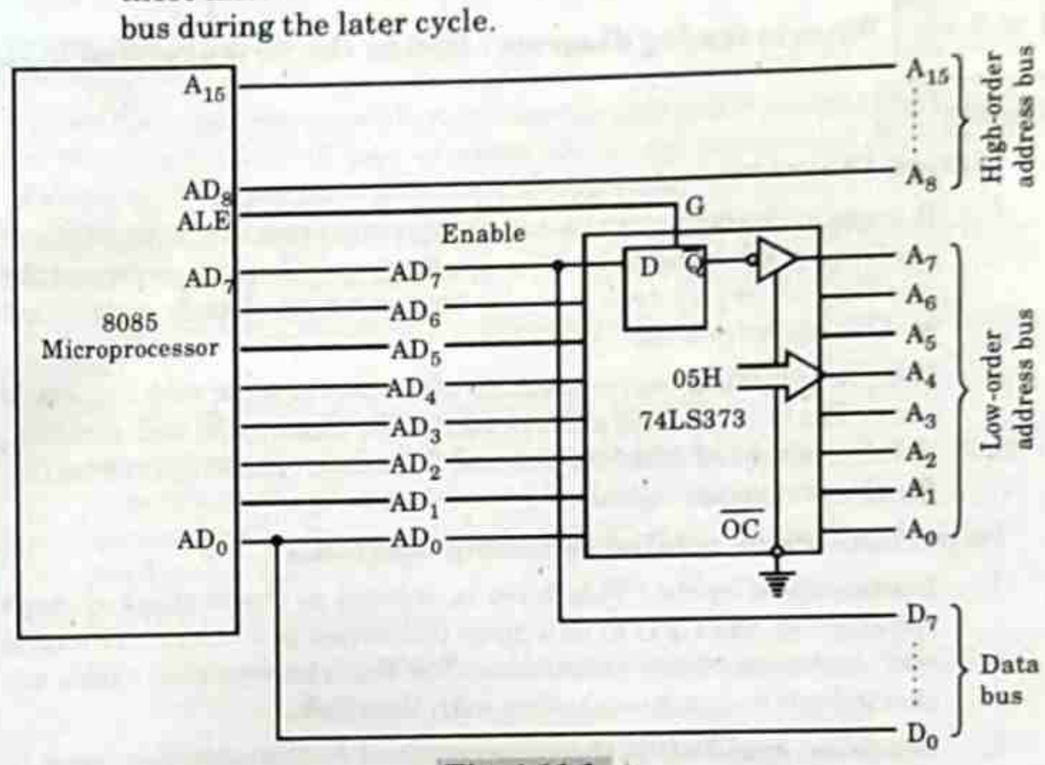


Fig. 1.11.1.

3. However, the lower order address is lost after the first clock period. This address needs to be latched and used for identifying the memory address.
4. To achieve this, the data bus $AD_0 - AD_7$ is needed to be demultiplexed. Fig. 1.11.1 shows the use of latch and ALE signal to demultiplex the bus.

Que 1.12. What is timer ? Give the registers of 8051.

Answer

A. Timer :

1. A timer is a specialized type of clock which is used to measure time intervals. A timer that counts from zero upwards for measuring time elapsed is often called a stopwatch.
2. It is a device that counts down from a specified time interval and used to generate a time delay, for example, an hourglass is a timer.

B. Timers of 8051 and their associated registers :

1. The 8051 has two timers, Timer 0 and Timer 1. They can be used as timers or as event counters. Both Timer 0 and Timer 1 are 16-bit wide.

Scanned by TapScanner

2. Since the 8051 follows an 8-bit architecture, each 16 bit is accessed as two separate registers of low-byte and high-byte.

Que 1.13. What is timing diagram? Define the terms related to it.

Answer

A. Timing Diagrams :

1. It is one of the best ways to understand to process of microprocessor/ controller. With the help of timing diagram we can understand the working of any system, step by step working of each instruction and its execution, etc.
2. It is the graphical representation of process in steps with respect to time. The timing diagram represents the clock cycle and duration, delay, content of address bus and data bus, type of operation *i.e.*, Read/write/status signals.

B. Important terms related to timing diagrams :

1. **Instruction cycle :** This term is defined as the number of steps required by the CPU to complete the entire process *i.e.*, Fetching and execution of one instruction. The fetch and execute cycles are carried out in synchronization with the clock.
2. **Machine cycle :** It is the time required by the microprocessor to complete the operation of accessing the memory devices or I/O devices. In machine cycle various operations like opcode fetch, memory read, memory write, I/O read, I/O write are performed.
3. **T-state :** Each clock cycle is called as T-states.

C. Rules to identify number of machine cycles in an instruction :

1. If an addressing mode is direct, immediate or implicit then No. of machine cycles = No. of bytes.
2. If the addressing mode is indirect then No. of machine cycles = No. of bytes + 1. Add +1 to the No. of machine cycles if it is memory read/write operation.
3. If the operand is 8-bit or 16-bit address then, No. of machine cycles = No. of bytes + 1.
4. These rules are applicable to 80 % of the instructions of 8085.

D. Diagram :

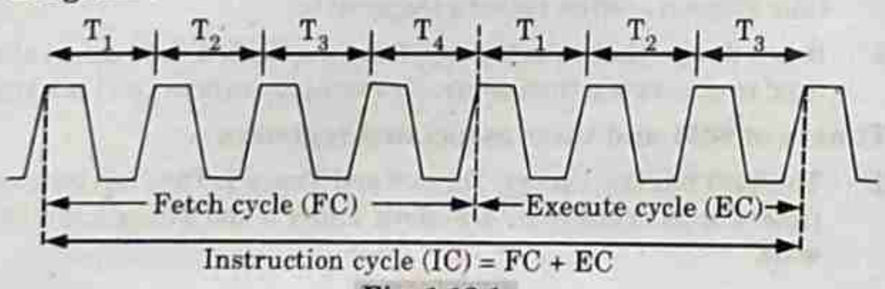


Fig. 1.13.1.

Que 1.14. Name various interfacing devices.

Answer

1. Every microprocessor based system design generally involves interfacing of the processor with one or more than one peripheral devices for communication purpose with the environment.
2. Large numbers of general and special purpose peripheral devices have been developed by the manufacturer over the past several years.
3. Their use reduces the chip count of the system and simplifies the design process.
4. Interfacing devices may be classified into two types :
   i. **General purpose peripherals :** These are the devices that perform a specific task but may be used for interfacing a variety of input/ output devices to the microprocessor. Some examples of general purpose peripheral are :
      a. An input/output port
      b. Programmable peripheral interface (also known as peripheral interface adapter)
      c. Programmable Interrupt Controller (PIC)
      d. Programmable DMA controller
      e. Programmable communication interface
      f. Programmable interval timer
   ii. **Special purpose peripherals :** These are the devices that may be used for interfacing a microprocessor to a specific type of input/ output device. These peripherals are more complex and relatively more costly than general purpose peripherals. Some examples of special purpose peripherals are :
      a. Programmable CRT controller
      b. Programmable hard disk controller
      c. Programmable keyboard and display interface.

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. Write a short note on evolution of the microprocessor.

Ans. Refer Q. 1.2.

Scanned by TapScanner

Q. 2. Explain various types of microprocessor.

Ans. Refer Q. 1.3.

Q. 3. Explain the general microprocessor architecture and operation of its components.

Ans. Refer Q. 1.4.

Q. 4. Explain data transfer schemes. Give its types.

Ans. Refer Q. 1.9.

Q. 5. Define instruction. Classify instructions.

Ans. Refer Q. 1.10.

Q. 6. What is timing diagram? Define the terms related to it.

Ans. Refer Q. 1.13.

Q. 7. Name various interfacing devices.

Ans. Refer Q. 1.14.

☺☺☺

2 UNIT

8085 Microprocessor

CONTENTS

Part-1 :	Pin Diagram and Internal Architecture of 8085 Microprocessor, Registers, ALU, Control and Status	**2-2C to 2-8C**
Part-2 :	Interrupt, Machine Cycle and Instruction Set : Addressing Modes	**2-8C to 2-17C**
Part-3 :	Instruction Format, Instruction Classification : Data Transfer, Arithmetic Operation, Logical Operation, Branching Operations, Machine Control	**2-17C to 2-26C**
Part-4 :	Assembler Directive	**2-26C to 2-28C**

Scanned by TapScanner

PART-1

Pin Diagram and Internal Architecture of 8085 Microprocessor, Registers, ALU, Control and Status.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.1. **Define the features of 8085 μP.**

Answer

1. It is an 8-bit microprocessor.
2. It operates on clock cycle with 50 % duty cycle.
3. It can operate with 3 MHz clock frequency.
4. It provides 8-bit I/O addresses to access (2^8) 256 I/O ports.
5. It provides five hardware interrupts : TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR.
6. It provides control signal (IO/ $\overline{M}$, $\overline{RD}$, $\overline{WR}$) to control the bus cycle and hence external bus controller is not required.
7. It has 8-bit accumulator, flag register, instruction register, six 8-bit general purpose register and two 16-bit registers (SP and PC).

Que 2.2. **Draw the pin diagram of 8085 and specify the function and direction of information flow of address bus, data bus and control bus.** **AKTU 2018-19, Marks 07**

Answer

The signals of 8085 can be classified into seven groups according to their functions :

1. Power supply and frequency signals :

a. V_{CC} : It requires a single + 5 V power supply.

b. V_{SS} : Ground reference.

c. **X_1 and X_2 :** A tuned circuit like LC, RC or crystal is connected at these two pins. The internal clock generator divides oscillator frequency by 2, therefore, to operate a system at 3 MHz, the crystal of tuned circuit must have a frequency of 6 MHz.

d. **CLK OUT :** This signal is used as a system clock for other devices. Its frequency is half the oscillator frequency.

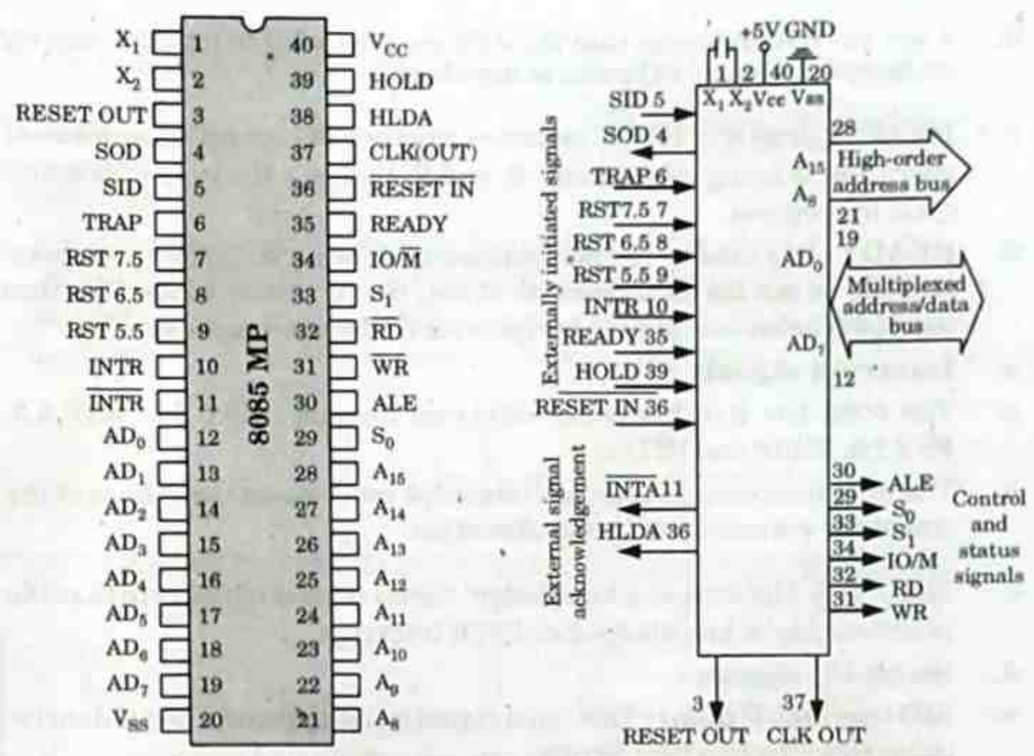


Fig. 2.2.1. Pin configuration and functional pin diagram.

2. Data bus and address bus :

a. **AD_0 to AD_7 :** The 8-bit data bus ($D_0 - D_7$) is multiplexed with the lower half ($A_0 - A_7$) of the 16-bit address bus. During first part of the machine cycle (T_1), lower 8-bits of memory address or I/O address appear on the bus. During remaining part of the machine cycle (T_2 and T_3) these lines are used as a bi-directional data bus.

b. **A_8 to A_{15} :** The upper half of the 16-bit address appears on the address lines A_8 to A_{15}. These lines are exclusively used for the most significant 8 bits of the 16-bit address lines.

3. Control and status signals :

a. **ALE (Address latch enable) :**

i. This is a positive going pulse generated every time when 8085 begins an operation (machine cycle). It indicates that the bits on $AD_7 - AD_0$ are address bits.

ii. This signal is used primarily to latch the low order address from the multiplexed bus and generate a separate set of eight address line, A_7–A_0.

b. **$\overline{RD}$ and $\overline{WR}$:**

i. These signals are basically used to control the direction of the data flow between processor and memory or I/O device/port.

ii. A low on $\overline{RD}$ indicates that the data must be read from the selected memory location or I/O port via data bus.

Scanned by TapScanner

iii. A low on $\overline{WR}$ indicates that the data must be written into the selected memory location or I/O port via data bus.

c. **IO/$\overline{M}$, S_0 and S_1 :** IO/$\overline{M}$ indicates whether I/O operation or memory operation is being carried out. S_1 and S_0 indicate the type of machine cycle in progress.

d. **READY :** It is used by the microprocessor to sense whether a peripheral is ready or not for data transfer. If not, the processor waits. It is thus used to synchronize slower peripherals to the microprocessor.

4. **Interrupt signals :**

a. The 8085 has five hardware interrupt signals : RST 5.5, RST 6.5, RST 7.5, TRAP and INTR.

b. The microprocessor recognizes interrupt requests on these lines at the end of the current instruction execution.

c. The $\overline{INTA}$ (Interrupt acknowledge) signal is used to indicate that the processor has acknowledged an INTR interrupt.

5. **Serial I/O signals :**

a. **SID (Serial I/P data) :** This input signal is used to accept serial data bit by bit from the external device.

b. **SOD (Serial O/P data) :** This is an output signal which enables the transmission of serial data bit by bit to the external device.

6. **DMA Signal :**

a. **HOLD :** This signal indicates that another peripheral device is requesting for the use of address bus, data bus and control bus.

b. **HLDA :** This active high signal is used to acknowledge HOLD request.

7. **Reset signals :**

a. **$\overline{RESET\ IN}$:** A low on this pin :

i. Sets the program counter to zero (0000H).

ii. Resets the interrupt enable and HLDA flip-flops.

iii. Tri-states the data bus, address bus and control bus.

iv. Affects the contents of processor's internal registers randomly.

b. **RESET OUT :** This active high signal indicates that processor is being reset. This signal is synchronized to the processor clock and it can be used to reset other devices connected in the system.

Que 2.3. **With the neat pin and block diagram, describe the internal architecture of 8085. State the function of each block.**

AKTU 2015-16, Marks 10

OR

Explain the architecture of 8085 microprocessor in brief with the help of neat diagram.

AKTU 2016-17, Marks 10

OR

With the help of neat block diagram, explain the architecture of 8085 microprocessor.

AKTU 2018-19, Marks 07

Answer

A. **Pin diagram of 8085 :** Refer Q. 2.2, Page 2–2C, Unit-2.

B. **Internal architecture or functional block diagram :** The 8085 is an 8-bit, general-purpose microprocessor that is ideally suited to many applications. Fig. 2.3.1 shows the architecture of 8085.

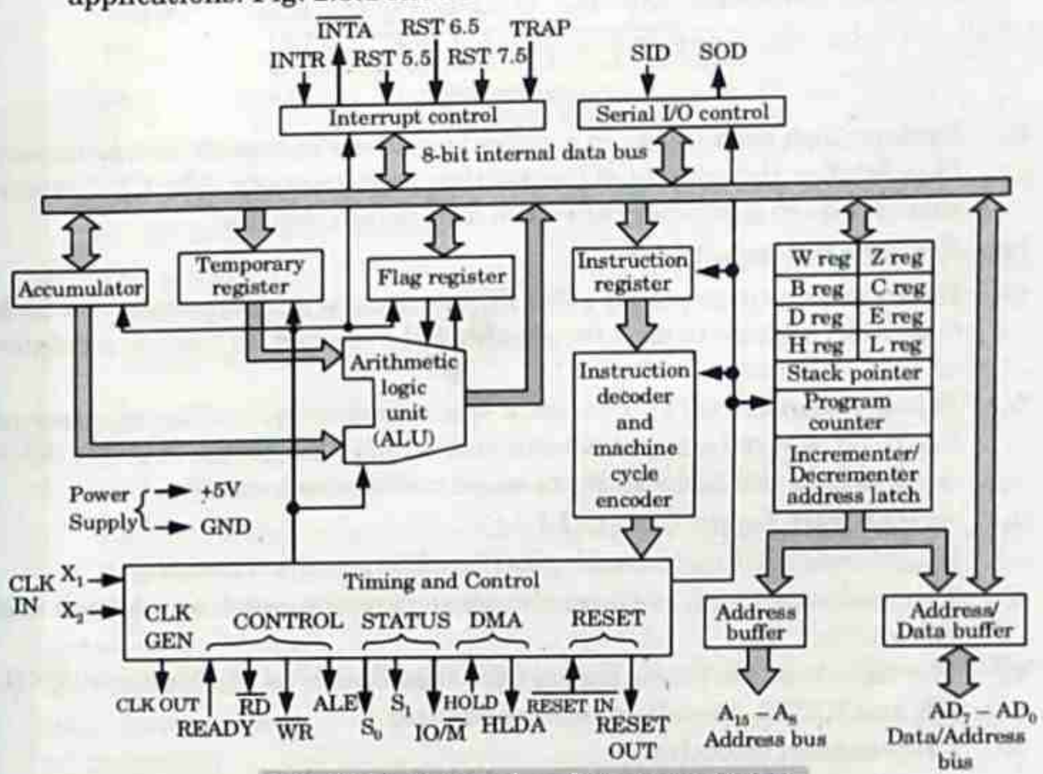


Fig. 2.3.1. Internal architecture of 8085.

C. **8085 consists of following functional blocks :**

1. **Registers :** The 8085 registers are classified as :

i. **General purpose register :** B, C, D, E, H and L are 8-bit general purpose registers, which can be used as a separate 8-bit registers or as 16-bit register pairs, BC, DE and HL.

ii. **Temporary registers :**

a. **Temporary data register :** It provides input to the ALU. The programmer cannot access temporary data register. It is internally used for execution of most of the arithmetic and logical instructions.

b. **W and Z registers :** These registers are used to hold 8-bit data during execution of some instructions. These registers are not available for programmer, since 8085 uses them internally.

Scanned by TapScanner

iii. Special purpose registers :

a. **Register A (Accumulator) :** It is an 8-bit register. It is used extensively in arithmetic, logic, load and store operations. Most of the times the result of arithmetic and logical operations is stored in the register A. Hence, it is also identified as accumulator.

b. **Flag register :** It is an 8-bit register, in which five of the bits carry significant information in the form of flags : S (Sign flag), Z (Zero flag), AC (Auxiliary carry flag), P (Parity flag) and CY (Carry flag) as shown in Fig. 2.3.2.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	×	AC	×	P	×	CY

Fig. 2.3.2.

c. **Instruction registers :** In a typical processor operation, the processor first fetches the opcode of instruction from memory. The CPU stores this opcode in a register called the instruction register.

iv. Sixteen bit registers :

a. **Program Counter (PC) :** The PC is a special purpose register which stores the address of next instruction to be fetched. PC acts as a pointer to next instruction.

b. **Stack Pointer (SP) :** The stack is a reserved area of the memory in the RAM where temporary information may be stored. A 16-bit SP is used to hold the address of the most recent stack entry.

2. Arithmetic Logic Unit (ALU) :

i. It performs arithmetic and logical functions on 8-bit variables.

ii. The arithmetic unit performs bitwise operations such as addition and subtraction.

iii. The logic unit performs logical operations such as complement, AND, OR and EXOR, as well as rotate and clear.

3. Instruction decoder :

i. The opcode of instruction which is fetched by the processor is stored in the instruction register.

ii. Then it is sent to the instruction decoder.

iii. The instruction decoder, decodes it and accordingly gives the timing and control signals which controls the register, the data buffers, ALU and external peripheral signals $\overline{INTA}$.

4. Address buffer :

i. This is an 8-bit unidirectional buffer.

ii. It is used to drive external high order address bus ($A_{15} - A_8$).

iii. It is also used to tri-state the high order address bus under certain conditions such as reset, hold, halt and when address lines are not in use.

5. Address/Data buffer :

i. This is an 8-bit bi-directional buffer. It is used to drive multiplexed address/data bus, *i.e.*, low-order address bus ($A_7 - A_0$) and data bus ($D_7 - D_0$).

ii. The address and data buffers are used to drive external address and data buses respectively.

6. Interrupt control :

i. Sometimes it is necessary to execute the collection of special routine whenever special condition exists within a program.

ii. The occurrence of this special condition is referred as interrupt.

iii. The interrupt control block has five interrupt inputs RST 5.5, RST 6.5, RST 7.5, TRAP and INTR and one acknowledge signal $\overline{INTA}$.

7. Serial I/O control :

i. The 8085 serial input/output control provides two lines, SOD and SID for serial communication.

ii. The serial output data (SOD) line is used to send data serially and serial input data (SID) line is used to receive data serially.

8. Timing and control circuitry :

i. The control circuitry in the 8085 is responsible for control and synchronization of all operations.

ii. The operations in 8085 are synchronized with the help of clock signal.

iii. Along with the control of fetching and decoding operations and generating appropriate signals for instruction execution, control circuitry also generates signal required to interface external devices to the 8085.

Que 2.4. **What is flag register ? Explain each flag of 8085 microprocessor.**

Answer

A. **Flag register :** It is 8-bit register in which five bits carry significant information in form of flags as shown in Fig. 2.4.1.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	×	AC	×	P	×	CY

Fig. 2.4.1. Flag register.

B. **Types :**

1. S (Sign flag) :

i. After the arithmetic or logical operations, if bit D_7 of the result is 1, the sign flag is set.

ii. In a given byte if D_7 is 1, the number will be viewed as negative number.

iii. If D_7 is 0, the number will be considered as positive.

Scanned by TapScanner

2. **Z (Zero flag) :**

i. The zero flag sets, if the result of operation in ALU is zero and flag reset if the result is non-zero.

ii. The zero flag is also set if a certain register content becomes zero following an increment and decrement operation of that register.

3. **AC (Auxiliary carry flag) :**

i. This flag is set if there is overflow out of bit 3, *i.e.*, carry from lower nibble to higher nibble (D_3 to D_4 bit).

ii. This flag register is used for BCD operations and not available for programmer.

4. **P (Parity flag) :**

i. Parity is defined by the number of 1's present in the accumulator.

ii. After arithmetic or logic operation if the result has an even number of 1's, *i.e.*, even parity, the flag is set.

iii. If the parity is odd, flag is reset.

5. **CY (Carry flag) :**

i. The flag is set if there is an overflow out of bit 7.

ii. The carry flag also serves as a borrow flag for subtraction.

PART-2

Interrupt, Machine Cycle and Instruction Set : Addressing Modes.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.5. **Define interrupt.**

OR

Enlist and explain the interrupt pins in 8085.

AKTU 2014-15, Marks 05

Answer

A. **Interrupt :** Interrupts are the signals generated by external devices to request the microprocessor to perform a task.

B. **Types of interrupt :**

a. **Hardware interrupts :**

1. 8085 microprocessor have five interrupt pin namely TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR.

2. If the signals on these pin go high then TRAP will be serviced first followed by RST 7.5, RST 6.5, RST 5.5 and INTR.

3. When all the interrupts are serviced then all the interrupts except TRAP are disabled. They can be enabled by executing enable interrupt (EI) instruction.

4. **Types of hardware interrupts :**

i. **TRAP :**

1. It is non-maskable interrupt. In order to service this interrupt the signal on TRAP pin must have high level.

2. If this condition occur then 8085 completes execution of current instruction, pushes the program counter on the stack and branches to location 0024H (Interrupt address vector for TRAP).

ii. **RST 7.5 :**

1. It is maskable interrupt. It can be enabled and disabled using SIM (Set interrupt mask) instruction.

2. In order to service RST 7.5, 8085 complete current execution of instruction, pushes the program counter onto the stack and branches to 003CH.

iii. **RST 6.5 :**

1. This can be enabled and disabled using SIM instruction.

2. RST 6.5 is high level sensitive, microprocessor 8085 execute current instruction, save the program counter onto the stack and branches to location 0034H.

iv. **RST 5.5 :**

1. It is maskable interrupt and can be enabled and disabled by SIM instruction. It is high level sensitive.

2. In order to service this interrupt 8085 completes the execution of current instruction, save the program counter onto the stack and branches to location 002CH.

v. **INTR :**

1. It is non-maskable interrupt. Whenever INTR signal is high then 8085 completes its current instruction and send active low interrupt acknowledge signal $\overline{INTA}$.

2. If interrupt is enabled in response to $\overline{INTA}$ signal, external logic places an instruction opcode on data bus on receiving the instruction.

3. 8085 save the address of next instruction onto stack and execute received instruction.

b. **Software interrupts :**

1. In software interrupts, the cause of the interrupt is an execution of the instruction. These are special instructions supported by the microprocessor.

Scanned by TapScanner

2. After execution of these instructions, microprocessor completes the execution of the instruction it is currently executing and transfers the program control to the subroutine program.
3. After completion of the execution of the subroutine program, program control returns to the main program.
4. The 8085 has eight software interrupts from RST 0 to RST 7.
5. The vector address can be calculated as shown in Table 2.5.1 :

   Interrupt number × 8 = Vector address

Table 2.5.1.

Instruction	Hex code	Vector address
RST 0	C7	0000H
RST 1	CF	0008H
RST 2	D7	0010H
RST 3	DF	0018H
RST 4	E7	0020H
RST 5	EF	0028H
RST 6	F7	0030H
RST 7	FF	0038H

Que 2.6. **Explain the role of interrupts in programming. Explain the interrupts used in 8085. List out all the vectored interrupts of 8085 and give their vector address.** **AKTU 2015-16, Marks 10**

Answer

A. **Role and types of interrupts :** Refer Q. 2.5, Page 2–8C, Unit-2.

B. **Vectored interrupts :**

1. When the internal control circuit of processor produces a CALL to a predetermined memory location which is the starting address of interrupt service routine, the address is called vector address and such interrupts are called vectored interrupts.

   Example : TRAP, RST 7.5, RST 6.5, RST 5.5

Interrupt type	Trigger	Priority	Maskable	Vector address
TRAP	Edge and level	1st (Highest)	No	0024H
RST 7.5	Edge	2nd	Yes	003CH
RST 6.5	Level	3rd	Yes	0034H
RST 6.5	Level	4th	Yes	002CH

2. The TRAP has the highest priority, followed by RST 7.5, 6.5, 5.5, and INTR, in that order; however, the TRAP has a lower priority than the HOLD signal used for DMA.

C. **Non-vectored interrupts :** When address of subroutine is received from the external device, is called non-vectored interrupt.

   Example : INTR.

Que 2.7. **Discuss the instruction cycle and machine cycle.**

Answer

1. Instruction cycle consists of opcode fetch followed by an execute cycle.
2. The execute cycle itself consists of zero or more fetch cycles which may be required to fetch the operand.
3. All these operations are performed in different time slots known as machine cycles.
4. An instruction cycle may consist of more than one machine cycles.
5. The first of these is always the opcode fetch cycle.
6. Some instruction, which require register to register transfer inside the microprocessor, may be executed in one machine cycle.
7. Whereas some which require data transfer between microprocessor and memory or input/output device may used more than one machine cycle.
8. Fig. 2.7.1 exhibits instruction cycle and machine cycle within it. It shows two machine cycles M_1 and M_2.
9. M_1 in which opcode is fetched and decoded consist of four state.
10. M_2 consists of three states, the data contained in the next byte of instruction is fetched and instruction executed.

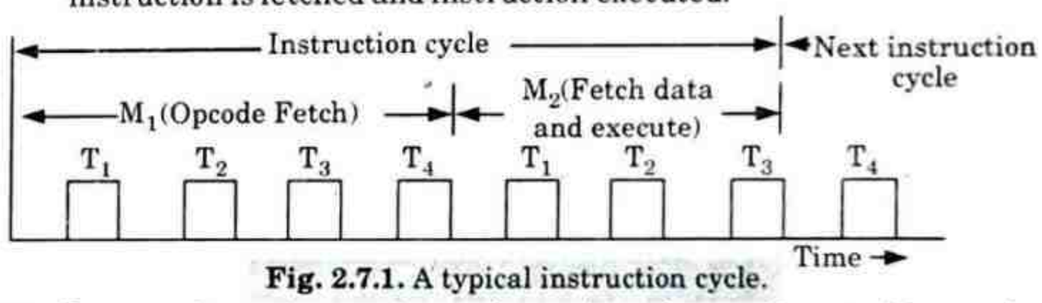


Fig. 2.7.1. A typical instruction cycle.

11. In general, one instruction cycle consists of one to five machine cycles and one machine cycle consists of three to six T-states, *i.e.*, three to six clock periods, as shown in Fig. 2.7.2.

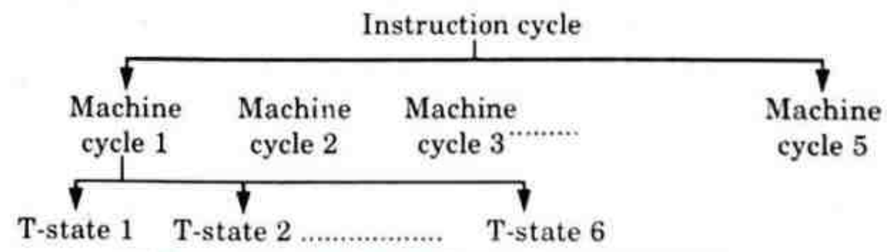


Fig. 2.7.2. Relationship between instruction cycle, machine cycle and T-state.

Scanned by TapScanner

Que 2.8. **Draw and explain the memory and I/O read cycle of 8085.**

AKTU 2016-17, 2017-18; Marks 10

OR

Draw and explain the timing diagram of memory read operation in 8085. Write different steps used in it. **AKTU 2015-16, Marks 10**

Answer

A. Read cycle :

1. The high order address ($A_{15} \Leftrightarrow A_8$) and low order address ($AD_7 \Leftrightarrow AD_0$) are asserted on 1st low going transition of the clock pulse.
2. The timing diagram for $IO/\overline{M}$ read are shown in Fig. 2.8.1 and Fig. 2.8.2. $A_{15} \Leftrightarrow A_8$ remains valid in T_1, T_2 and T_3 *i.e.*, duration of the bus cycle, but $AD_7 \Leftrightarrow AD_0$ remains valid only in T_1.
3. Since it has to remain valid for the whole bus cycle, it must be saved for its use in T_2 and T_3.

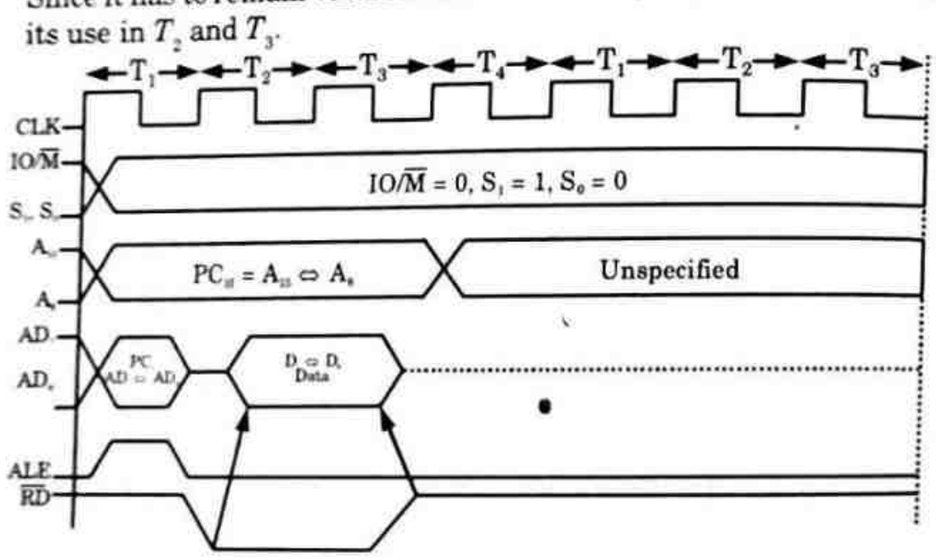


Fig. 2.8.1. Memory read timing diagram.

4. ALE is asserted at the beginning of T_1 of each bus cycle.
5. ALE is active during T_1 only and is used as the clock pulse to latch the address ($AD_7 \Leftrightarrow AD_0$) during T_1. The $\overline{RD}$ is asserted near the beginning of T_2.
6. It ends at the end of T_3.
7. As soon as the $\overline{RD}$ becomes active, it forces the memory or I/O port to assert data.
8. $\overline{RD}$ becomes inactive towards the end of T_3, causing the port or memory to terminate the data.

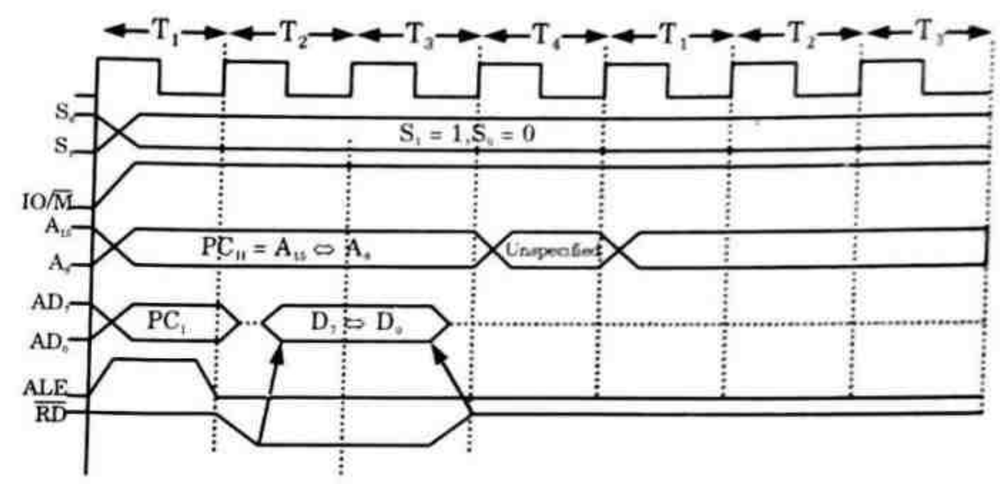


Fig. 2.8.2. I/O read timing diagram.

B. Write Cycle :

1. Immediately after the termination of the low order address, at the beginning of the T_2, data is asserted on the address/data bus by the processor.
2. $\overline{WR}$ control is activated near the start of T_2 and becomes inactive at the end of T_3.
3. The processor maintains valid data until after $\overline{WR}$ is terminated.
4. This ensures that the memory or port has valid data while $\overline{WR}$ is active, it is clear from Fig. 2.8.3.
5. For READ bus cycle, the data appears on the bus as a result of activating $\overline{RD}$ and for the $\overline{WR}$ bus cycle, valid data is appeared on the bus during the time when $\overline{WR}$ is active.

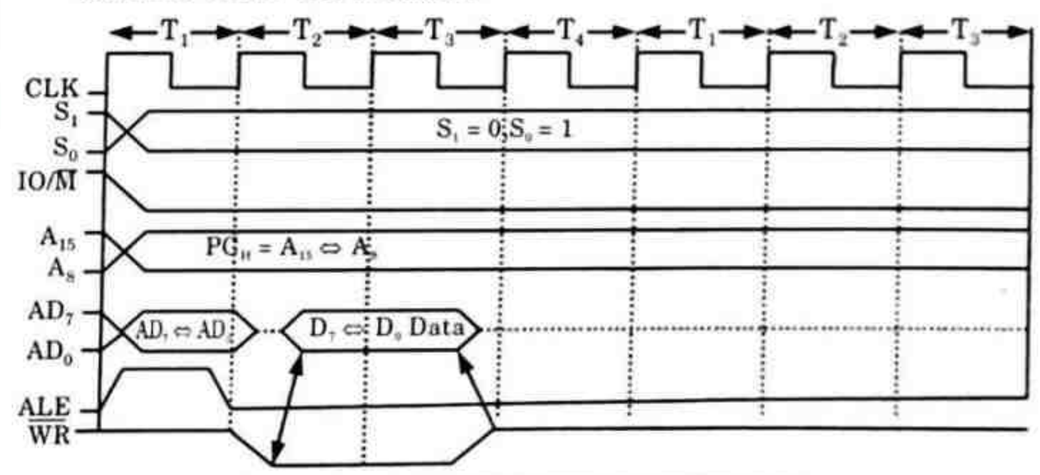


Fig. 2.8.3. Memory write timing diagram.

Scanned by TapScanner

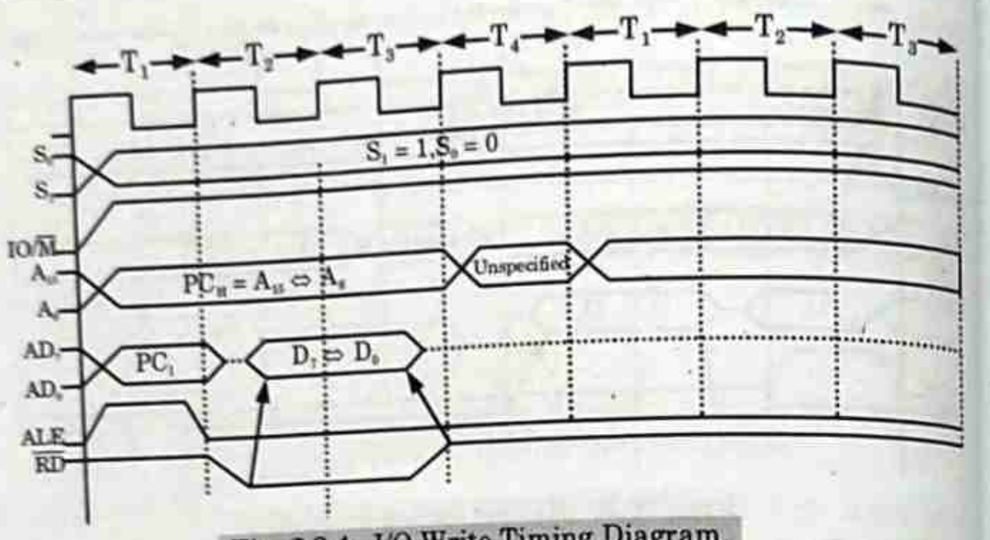


Fig. 2.8.4. I/O Write Timing Diagram.

Que 2.9. **Draw and discuss the instruction fetch, read and write cycle of 8085.** **AKTU 2014-15, Marks 10**

Answer

A. Read cycle : Refer Q. 2.8, Page 2–12C, Unit-2.

B. Write cycle : Refer Q. 2.8, Page 2–12C, Unit-2.

C. Instruction fetch cycle :

Memory Location	Machine Code	Instruction
2000H	0 0 1 1 1 1 1 0 → 3EH	MVI A,46H ; Load byte 46H in the accumulator
2001H	0 1 0 0 0 1 1 0 → 46H	

1. This instruction consists of two bytes : The first is the opcode and the second is the data byte.
2. The 8085 need to read these bytes first from memory and thus require at least two machine cycles.
3. The first machine cycle is Opcode Fetch and the second machine cycle is Memory Read, as shown in Fig. 2.9.1.

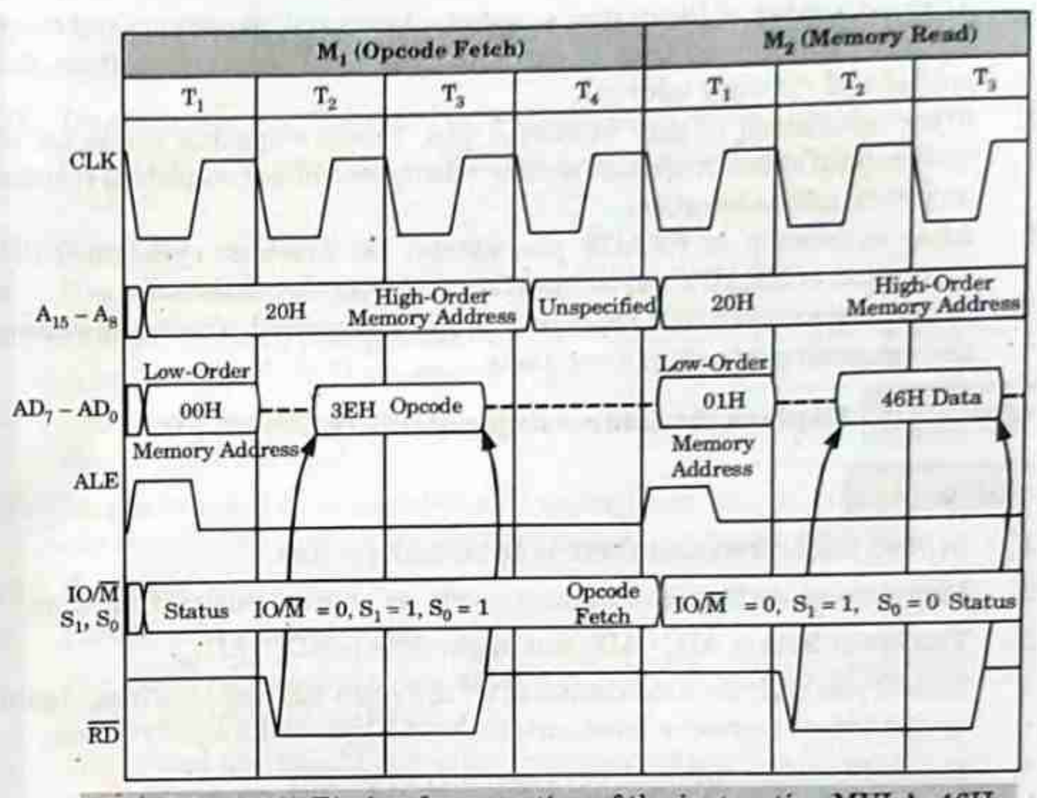


Fig. 2.9.1. 8085 Timing for execution of the instruction MVI A, 46H.

4. This instruction requires seven T-states for these two machine cycles :

i. The first machine cycle M_1 (Opcode Fetch) is identical in bus timings with the machine cycle except for the bus contents.

ii. After completion of the Opcode Fetch cycle, the 8085 places the address 2001H on the address bus and increments the program counter to the next address 2002H.

iii. The second machine cycle M_2 is identified as the Memory Read cycle ($IO/\overline{M}$) = 0, S_1 = 1, and S_0 = 0) and the ALE is asserted.

iv. At T_2, the $\overline{RD}$ signal becomes active and enables the memory chip.

v. At the rising edge of T_2, the 8085 activates the data bus as an input bus, memory places the data byte 46H on the data bus, and the 8085 read and stores the byte in the accumulator during T_3.

Que 2.10. **Explain the function of externally initiated signals of 8085.** **AKTU 2016-17, Marks 05**

Answer

External devices can initiate the operation by activating Reset, Interrupt, READY or HOLD pins of the microprocessor. These operations are as follows :

Scanned by TapScanner

1. After activation of Reset pin, suspend all internal operations and clear program counter so that it can fetch the next instruction from the predefined memory address.
2. After activation of any interrupt pin, execute specific sequence of instructions called interrupt service routine and after completion resume its interrupted operation.
3. After activation of READY pin, extend the machine cycle until the activation of READY pin to interface with the slower devices.
4. After activation of HOLD pin, relinquish the control of buses and allow the external controller to use them.

Que 2.11. Explain the addressing capability of 8085 μP.

Answer

1. In 8085 microprocessor there is 16 bit address line.
2. Microprocessor 8085 communicates via its address bus of 16 bit wide.
3. The lower byte is AD_0 - AD_7 and upper byte is AD_8 - AD_{15}.
4. Thus it can address a maximum of 2^{16} different address locations. Again each address (memory location) can hold 1 byte of data/instruction.
5. Hence the maximum address capability of 8085 is

$$= 2^{16} \times 1 \text{ Byte} = 65{,}536 \times 1 \text{ Byte}$$
$$= 64 \text{ kB (where 1 k = 1024 Bytes)}$$

Que 2.12. Explain the addressing modes of 8085.

AKTU 2014-15, Marks 05

OR

Explain the data addressing modes of 8085 with example.

AKTU 2016-17, Marks 10

OR

Explain evolution of microprocessor with its different generation. What do you mean by addressing mode, explain different addressing mode used in 8085 with suitable example. **AKTU 2017-18, Marks 10**

Answer

A. **Evolution of microprocessor :** Refer Q. 1.2, Page 1–2C, Unit-1.

B. **Addressing modes :** The different ways that a microprocessor can access data are referred to as addressing modes.

C. **Types :**

1. **Immediate addressing mode :**

i. In immediate addressing mode, the operand is specified within the instruction itself.

ii. One or two bytes within the instruction are used for specifying the data itself.

iii. **Importance :** In the immediate addressing mode, the data is stored along with the instruction and the data is directly transferred to the register.

iv. **Example :** MVI A, 05H

2. **Register addressing mode :**

i. The register addressing mode specifies the source operand, destination operand, or both to be contained in an 8085 registers.

ii. This results in faster execution, since it is not necessary to access memory locations for operands.

iii. **Importance :** In the register addressing mode, the data is copied from one register and stored in another register (or register pair).

iv. **Example :** MOV A, B

3. **Direct addressing mode :**

i. The direct addressing mode specifies the 16-bit address of the operand within the instruction itself.

ii. The second and third bytes of instruction contain this 16-bit address.

iii. **Example :** LDA 2000H

4. **Indirect addressing mode :**

i. In indirect addressing mode, the memory address where the operand located is specified by the contents of a register pair.

ii. **Example :** LDAX B

5. **Implied or implicit addressing mode :**

i. In implied addressing mode, opcode specifies the address of the operands.

ii. **Importance :** In the register addressing mode, no operand (register or memory location or data) is specified.

iii. **Example :** CMA

PART-3

Instruction Format, Instruction Classification : Data Transfer, Arithmetic Operation, Logical Operation, Branching Operations, Machine Control.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Scanned by TapScanner

Que 2.13. **Define the term instruction and explain the instruction format of microprocessor 8085.**

Answer

A. **Instruction :**

1. Instruction is a command given by the user to the microprocessor and needs to be get executed by the processor, the entire group of instruction are called instruction set.
2. Thus, instruction format of 8085 has two fundamental fields :

i. Opcode (Information of operation)

ii. Operand (Address of location)

B. **Types of instruction :**

1. **1-byte instructions :** In single byte instruction there is only one field, Example : INRC & ADDC.
2. **2-byte instructions :** The 2-byte instructions have 2-fields, namely, the opcode field and the data or address field. Example : SBI 02H.
3. **3-byte instructions :** The 3-byte instructions also have only 2-fields, namely opcode and data/address field. Example : LXI D, 2060H.

Que 2.14. **What do you mean by data transfer instructions ? Explain.**

Answer

A. **Data transfer instructions :** These are the instructions used to copy the data from one memory location or register to another memory location or register.

B. **Instructions :**

1. **MOV (Move) :** This instruction copies the content of source register to the destination register.

   Format : MOV Source, Dest.

2. **MVI (Move immediate data) :** This instruction is used to move immediate data into the specified register.

   Format : MVI R, 8-bit data

3. **LXI (Load register pair immediate) :** This instruction load 16-bit data in the register pair B-C, D-E and H-L with the data byte available immediately.

   Format : LXI R_p, 16-bit address

4. **LHLD (Load H and L register direct) :** This instruction copies the contents of memory location pointed by 16-bit address in register L and copies the contents of next memory location in register H.

   Format : LHLD 16-bit address

5. **LDA (Load accumulator direct) :** This instruction copies the data of given memory location in accumulator.

   Format : LDA 16-bit address

6. **LDAX (Load accumulator indirect) :** This instruction copies the contents of memory location whose address is specified by register pair into the accumulator.

   Format : LDAX R_p

7. **SHLD (Store H and L register direct) :** This instruction stores the contents of L register in the memory location given within the instruction and content of H register at address next to it.

   Format : SHLD 16-bit address

8. **STA (Store accumulator direct) :** The contents of accumulator is copied into memory location specified by operand.

   Format : STA 16-bit address

9. **STAX (Store accumulator indirect) :** The contents of accumulator are copied into memory location specified by register pair.

   Format : STAX R_p

Que 2.15. **What do you understand by arithmetic instruction ?**

Answer

A. **Arithmetic instruction :** These are the instructions used for arithmetic operation like addition, subtraction etc.

B. **Instructions :**

1. **ADD (Add register data to accumulator) :** This instruction add the contents of register or memory with the content of accumulator and result is stored in accumulator.

   Format : ADD R/M

2. **ACI (Add immediate data to accumulator with carry) :** This instruction add the 8-bit data and carry flag with the accumulator contents and result is stored in accumulator.

   Format : ACI 8-bit data

3. **ADC (Add register data to accumulator with carry) :** The contents of register or memory and carry flag are added to contents of accumulator and result is placed in the accumulator.

   Format : ADC R/M

4. **ADI (Add immediate data to accumulator) :** The 8-bit data are added to contents of accumulator and result is stored in accumulator.

   Format : ADI 8-bit data

5. **DAD (Add register pair to H and L register) :** This instruction is used to add the 16-bit contents of specified register with contents of register pair HL and sum is saved in HL register.

Scanned by TapScanner

Format : DAD R_p

6. **SUB (Subtract register or memory from accumulator) :** The contents of register or memory location specified by operand are subtracted from the contents of accumulator.

   Format : SUB R/M

7. **SUI (Subtract immediate data from accumulator) :** The 8-bit data are subtracted from contents of accumulator and result is stored in accumulator.

   Format : SUI 8-bit data

8. **SBB (Subtract source and borrow from accumulator) :** The contents of register or memory and the borrow flag are subtracted from contents of accumulator and result are placed in accumulator.

   Format : SBB R/M

9. **SBI (Subtract immediate with borrow) :** The 8-bit data and borrow are subtracted from the contents of accumulator.

   Format : SBI 8-bit data

10. **DAA (Decimal adjust accumulator) :** The contents of the accumulator are change from the binary value to two 4-bits binary coded decimal digit. This is the only instruction that uses auxiliary flag to perform binary to BCD conversion.

    Format : DAA

11. **INR (Increment contents of register / memory by 1) :** This instruction increments the contents of register/memory by 1 and result are stored in same place.

    Format : INR R/M

12. **INX (Increment register pair by 1) :** This instruction increments the contents of specified register pair by 1.

    Format : INX R_p

13. **DCR (Decrement source by 1) :** The contents of designated register/memory is decremented by 1 and store the result in same place.

    Format : DCR R/M

14. **DCX (Decrement register pair by 1) :** This instruction decrements the contents of specified register pair by 1.

    Format : DCX R_p

Que 2.16. **What do you understand by logical instructions ?**

Answer

A. **Logical instructions :** These are the instructions used to perform the logical operation like AND, OR, XOR etc.

B. **Instructions :**

1. **ANA (Logical AND with accumulator) :** The contents of accumulator are logically ANDed with the contents of register or memory location and result is stored in the accumulator.

   Format : ANA R/M

2. **ANI (AND immediate with accumulator) :** This instruction ANDed the contents of accumulator with specified 8-bit data and result is placed in accumulator.

   Format : ANI 8-bit data

3. **XRA (Exclusive OR with accumulator) :** The contents of specified register/memory location are exclusive ORed with contents of the accumulator and result is placed in accumulator.

   Format : XRA R/M

4. **XRI (Exclusive OR immediate with accumulator) :** The 8-bit data are exclusive ORed with contents of accumulator and result is placed in the accumulator.

   Format : XRI 8-bit data

5. **ORA (Logically OR with accumulator) :** This instruction logically ORed the contents of accumulator with contents of register on memory location.

   Format : ORA R/M

6. **ORI (Logically OR immediate) :** The contents of accumulator are logically ORed with the 8-bit data in operand and the result is stored in accumulator.

   Format : ORI 8-bit data

7. **CPI data (Logically compare immediate) :** Compare data with the contents of A of less than equal to or greater.

8. **CMP M (Logically compare) :** Compare the contents of memory with the contents of A for less than equal to or greater than.

9. **RAR :** This instruction rotates the content of accumulator right by one position. Bit B_0 is placed in CY and CY is placed in B_7.

Que 2.17. **What do you understand by branch operation ? List all the branching instruction and give their details.**

Answer

A. **Branching instruction :** This group of instruction alters the sequence of program execution either conditionally or unconditionally.

B. **Instructions :**

1. **JMP (Jump unconditionally) :** The jump instructions specify the memory location explicitly. The program sequence is transferred to memory location specified by the 16-bit address. The unconditionally jump instruction enables the programmer to set up continuous loop.

Scanned by TapScanner

Format : JMP 16-bit address

Instruction	Description	Condition for JUMP
JC	Jump if carry	CY = 1
JNC	Jump if no carry	CY = 0
JP	Jump if positive	S = 0
JM	Jump if minus	S = 1
JPE	Jump if parity even	P = 1
JPO	Jump if parity odd	P = 0
JZ	Jump if zero	Z = 1
JNZ	Jump if no zero	Z = 0

2. **CALL (Unconditional subroutine call) :** The program sequence is transferred to address specified by operand. Before the transfer, the address of next instruction to CALL is pushed on the stack.

Format : CALL 16-bit address

Instruction Code	Description	Flag status
CC	Call if carry	CY = 1
CNC	Call if no carry	CY = 0
CP	Call if positive	S = 0
CM	Call if minus	S = 1
CPE	Call if parity even	P = 1
CPO	Call if parity odd	P = 0
CZ	Call if zero	Z = 1
CNZ	Call if no zero	Z = 0

3. **RET (Return from subroutine unconditionally) :** This instruction pops the return address from the stack and loads program counter with this return address. Thus transfers program control to the instruction next to CALL in the main program.

Instruction code	Description	Flag status
RC	Return if carry	CY = 1
RNC	Return if no carry	CY = 0
RP	Return if positive	S = 0
RM	Return if minus	S = 1
RPE	Return if parity even	P = 1
RPO	Return if parity odd	P = 0
RZ	Return if zero	Z = 1
RNZ	Return if no zero	Z = 0

4. **PCHL (Load program counter with HL contents) :** The contents of register H and L are copied into program counter by the help of this instruction.

5. **RST (Restart) :** This instruction transfers the program control to the specific memory location. This instruction is like a fixed address CALL instruction. This fixed address is also called as vectored address.

Instruction code	Vector	Address
RST 0	0 × 8	= 0000H
RST 1	1 × 8	= 0008H
RST 2	2 × 8	= 0010H
RST 3	3 × 8	= 0018H
RST 4	4 × 8	= 0020H
RST 5	5 × 8	= 0028H
RST 6	6 × 8	= 0030H
RST 7	7 × 8	= 0038H

The processor multiplies the RST number by 8 to calculate these vector addresses. The RST instruction saves the current program counter contents on the stack like CALL instruction.

Que 2.18. **Explain machine control instructions ?**

OR

Explain the following instructions of 8085 microprocessor

a. POP PSW **b. XTHL**
c. SPHL **d. PUSH PSW**
e. CMP M

AKTU 2017-18, Marks 10

Answer

A. Machine control instruction : These instructions control machine function such as Halt, Interrupt or do nothing.

B. Various machine control instructions are :

1. **DI (Disable Interrupts) :** This instruction resets the interrupt disable flip flop and all interrupt except TRAP are disabled.
2. **EI (Enable Interrupt) :** This instruction will set the interrupt enable flip flop and all the interrupts are enabled.
3. **HLT (Halt and Enter Wait State) :** This instruction Halt the processor. It can be restarted by a valid interrupt or by applying a RESET signal.
4. **NOP (No Operation) :** The instruction is fetched and decoded, however, no operation is executed.
5. **SIM (Set Interrupt Mask) :** This is a multipurpose instruction and is used to implement the 8085 interrupts. For this instruction command byte must be loaded in accumulator.
6. **RIM (Read Interrupt Mask) :** This instruction loads the status of interrupt mask, the pending interrupt and the contents of serial input data line into the accumulator.

Scanned by TapScanner

7. **PUSH (Push Register Pair onto Stack/PSW) :** This instruction copies the contents of register pair on the stack in the following sequence :

i. The stack pointer is decremented and the contents of higher order register (B, D, H, A) are copied on that location.

ii. The stack pointer is decremented again and the contents of low order register (C, E, L) are copied to that location.

8. **XCHG :** This instruction exchanges the contents of H & L with D & E registers.

9. **POP (Pop off Stack to Register Pair/PSW) :**

i. This instruction copies the contents of memory location pointed by the stack pointer into the lower byte of specified register pair and increment the stack pointer by one.

ii. It then copies the contents of memory location pointed by stack pointer into higher bytes of specified register pair and increment the stack pointer again by one.

iii. Only higher order register is to be specified in the instruction.

10. **SPHL (Copy H and L register to the stack pointer) :**

i. This instruction copies the contents of HL register pair into the stack pointer.

ii. The contents of H register are copied to higher order byte of stack pointer and contents of L register are copied to the lower byte of stack pointer.

11. **XTHL (Exchange H and L with top stack) :**

i. This instruction exchanges the contents of memory location pointed by stack pointer with contents of L-register and contents of next memory location with the contents of H register.

ii. This instruction does not modify the stack pointer.

12. **IN (Input Data to Accumulator from the Port with 8 bit address) :** This instruction copies the data at the port whose address is specified in the instruction into the accumulator.

13. **OUT (Output Data from Accumulator to a Port with 8-bit Address) :** This instruction copies the contents of the accumulator into the output port specified in the instruction.

14. **Other :**

i. **CMP M :** Refer Q. 2.16, Page 2–20C, Unit-2.

ii. **DAA :** Refer Q. 2.15, Page 2–19C, Unit-2.

iii. **SUI data :** Refer Q. 2.15, Page 2–19C, Unit-2.

iv. **CNC addr :** Refer Q. 2.17, Page 2–21C, Unit-2.

v. **RAR :** Refer Q. 2.16, Page 2–20C, Unit-2.

vi. **RST 5 :** Refer Q. 2.17, Page 2–21C, Unit-2.

Que 2.19. **What is result of executing the following instructions of 8085 ?**

i. **MOV AL, AIH**
ii. **CBW**
iii. **CWD**

Answer

A. **MOV AL, AIH :** This instruction moves the content of AI in the AH.

B. **CBW :**

1. CBW also known as convert signed byte to signed word.
2. This instruction fills AH with the sign of the byte in AL. AH is then the sign extension of AL.
3. CBW is usually done before the signed byte in AL can be divided by another signed byte with the IDIV instruction.
4. It does not affect any flag.

C. **CWD :**

1. CWD also known as the convert signed word to signed double word instruction.
2. It is used to extend the sign bit of a word in AX register to all the bits of the DX register.
3. Generally used before a signed word in AX. Then it is divided by another signed word using IDIV instruction.
4. It does not affect any flag.

Que 2.20. **Explain the flag register of 8085 microprocessor. What is flags status when addition of following hexadecimal number is performed ?**

i. **EE + 7D**
ii. **A5 + 59**
iii. **87 + 5B**

AKTU 2014-15, Marks 10

Answer

A. **Flag registers :** Refer Q. 2.4, Page 2–7C, Unit-2.

Scanned by TapScanner

B. Numerical :

i. EE + 7D :

```
EE   11101110
7D + 01111101
Sum  11011011
```

Flag status S = 1, Z = 0, AC = 0, P = 1, CY = 0

ii. A5 + 59 :

```
A5   10100101
59 + 01011001
Sum  11111110
```

Flag status S = 1, Z = 0, AC = 0, P = 0, CY = 1

iii. 87 + 5B :

```
87   10000111
5B + 01011011
Sum  11011110
```

Flag status S = 1, Z = 0, AC = 0, P = 0, CY = 1

Que 2.21. **Explain the flags of 8085 microprocessor. Give the flag status when following additions are performed.**

i. 51H + A9H ii. 2EH + 5AH iii. 76H + A4H

AKTU 2016-17, Marks 10

Answer

A. Flag : Refer Q. 2.4, Page 2–7C, Unit-2.

B. Numerical :

i.
```
  51H = 01010001
+ A9H = 10101001
SUM = 11111010
```
Flag status : S = 1, Z = 0, CY = 0, P = 1, AC = 0

ii.
```
  2 EH = 00101110
+ 5AH = 01011010
SUM = 10001000
```
Flag status : S = 1, Z = 0, CY = 0, P = 1, AC = 1

iii.
```
  76H = 01110110
+ A4H = 10100100
SUM = 000 11010
```
Flat status : S = 0, Z = 0, CY = 1, P = 0, AC = 0

PART-4

Assembler Directive.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.22. **What is assembler directive ? Explain its type with examples.**

Answer

A. Assembler directives :

1. Assembler directives are also called as pseudoinstructions. There are a number of assembler directives.
2. They enable us to control the way a program is assembled and listed. They do not generate any machine code for execution. The most commonly used assembler directives are ORG, DB, DW, EQU, and END.

B. Types :

i. **ORG directive :** This directive provides the origin or starting address for the assembly. It can also be written as ORIGIN. If this directive is not used, the starting address defaults to 0000H.

ii. DB directive :

a. DB stands for Define Byte. It can also be written as .DB or.BYTE or BYTE. This directive reserves 1 byte of memory and initializes it with the value following the DB directive.

b. If no value is indicated after the DS directive, the value defaults to 00H. If more than one value is indicated with comma as the separator, they are stored in consecutive byte locations.

iii. DW directive :

a. DW stands for Define Word. It can also be written as DW or .WORD or WORD. This directive reserves one word of memory (2-byte locations) and initializes it with the value following the DW directive.

b. If no value is indicated after the DW directive, the value defaults to 0000H. If more than one value is indicated with comma as the separator, they are stored in consecutive word locations.

iv. EQU directive :

a. This directive equates the label to the left of EQU directive with the value to the right of the EQU directive. The value on the right of the EQU directive could be a number or an arithmetic expression or another symbol.

Scanned by TapScanner

b. Normally, there will be a number to the right of the EQU directive. The EQU directive can also be written as EQUAL. The EQU directive does not reserve any memory locations, and it does not initialize the contents of any memory location

v. **END directive :**

a. This directive defines the physical end of a program. There can be a symbolic address immediately following the END directive.

b. If the address is specified, it indicates the program starting address. Without this address, the program execution will be from the first executable instruction in the program.

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. Draw the pin diagram of 8085 and specify the function and direction of information flow of address bus, data bus and control bus.

Ans. Refer Q. 2.2.

Q. 2. With the neat pin and block diagram, describe the internal architecture of 8085. State the function of each block.

Ans. Refer Q. 2.3.

Q. 3. Enlist and explain the interrupt pins in 8085.

Ans. Refer Q. 2.5.

Q. 4. Explain the role of interrupts in programming. Explain the interrupts used in 8085. List out all the vectored interrupts of 8085 and give their vector address.

Ans. Refer Q. 2.6.

Q. 5. Draw and explain the memory and I/O read cycle of 8085.

Ans. Refer Q. 2.8.

Q. 6. Draw and discuss the instruction fetch, read and write cycle of 8085.

Ans. Refer Q. 2.9.

Q. 7. Explain evolution of microprocessor with its different generation. What do you mean by addressing mode, explain different addressing mode used in 8085 with suitable example.

Ans. Refer Q. 2.12.

Q. 8. Explain machine control instructions ?

Ans. Refer Q. 2.18.

Q. 9. What is result of executing the following instructions of 8085 ?

i. MOV AL, AIH
ii. CBW
iii. CWD

Ans. Refer Q. 2.19.

Q. 10. Explain the flag register of 8085 microprocessor. What is flags status when addition of following hexadecimal number is performed ?

i. EE + 7D
ii. A5 + 59
iii. 87 + 5B

Ans. Refer Q. 2.20.

Q. 11. What is assembler directive ? Explain types with examples.

Ans. Refer Q. 2.22.

☺☺☺

Scanned by TapScanner

3 UNIT

8086 Microprocessor

CONTENTS

Part-1 :	Architecture of 8086 Microprocessor : Bus Interface Unit and Execution Unit	**3–2C to 3–5C**
Part-2 :	Register Organization, Memory Addressing, Memory Segmentation	**3–5C to 3–8C**
Part-3 :	Operating Modes	**3–9C to 3–15C**
Part-4 :	Instruction Set and Instruction Format	**3–15C to 3–19C**
Part-5 :	Types of Instructions	**3–19C to 3–25C**
Part-6 :	Interrupts : Hardware and Software Interrupts	**3–25C to 3–28C**

PART-1

Architecture of 8086 Microprocessor : Bus Interface Unit and Execution Unit.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.1. **State the features of 8086. Give the difference between 8085 and 8086 microprocessor.**

Answer

A. Features :

1. The 8086 is a 16-bit microprocessor. The arithmetic logic unit, internal registers and most of its instruction are designed to work with 16-bit binary words.
2. The 8086 has a 16-bit data bus, so it can read data from or write data to memory and ports either 16-bits or 8-bits at a time.
3. The 8086 has a 20-bit address bus, so it can directly access 2^{20} or 1 MB memory locations.
4. The 8086 has multiplexed address and data bus which reduces the number of pins needed, but it slows down the transfer of data.
5. The 8086 requires one phase clock with a 33 % duty cycle to provide optimized internal timing.

B. Difference between 8085 & 8086 :

S. No.	8085 μP	8086 μP
1.	8085 is an 8-bit microprocessor.	8086 is a 16-bit microprocessor.
2.	8085 has 16 address lines.	8086 has 20 address lines.
3.	8085 has only five flags.	8086 has nine flags.
4.	It does not support pipelining.	It supports pipelining.
5.	8085 operates on clock cycle with 50 % duty cycle.	8086 operates on clock cycle with 33 % duty cycle.

Que 3.2. **Draw the internal architecture of 8086 and discuss each block diagram.**

Scanned by TapScanner

Answer

A. The internal block diagram (architecture) of 8086 :

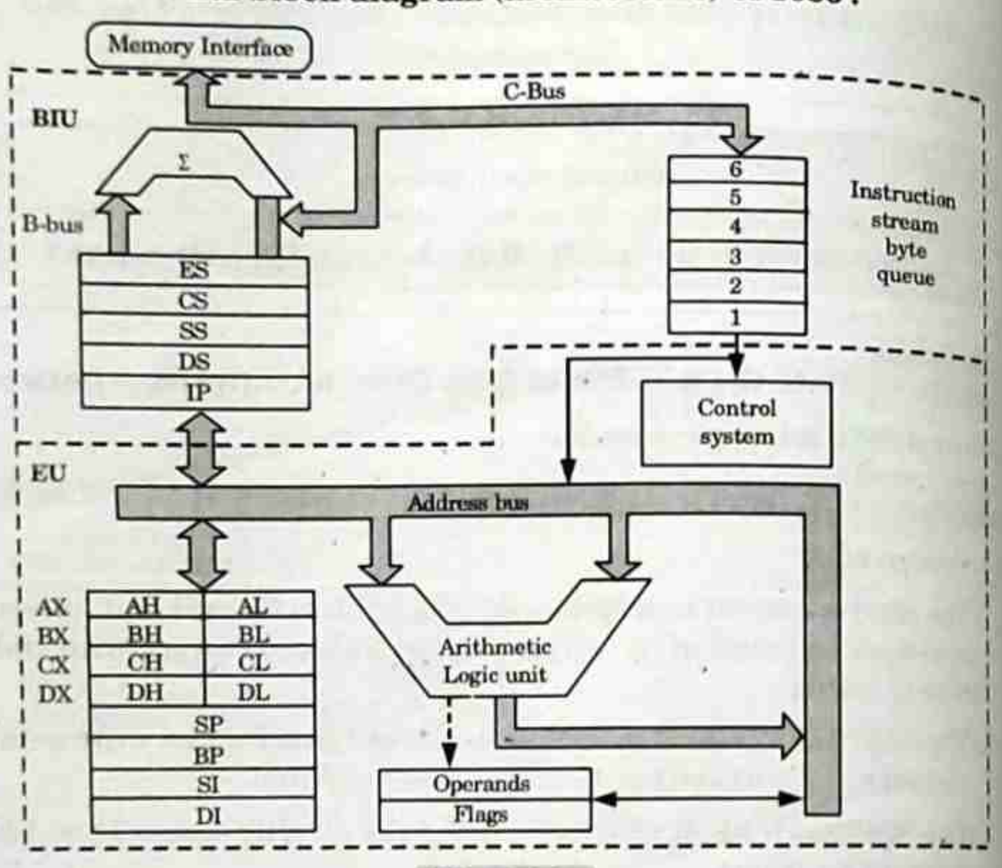


Fig. 3.2.1.

B. It is internally divided into two separate functional units :

1. Bus Interface Unit (BIU) :

a. The bus interface unit is the 8086's interface to the outside world. It provides a full 16-bit bi-directional data bus and 20-bit address bus.

b. The bus interface unit is responsible for performing all following external bus operations :

i. It sends address of the memory or I/O.

ii. It fetches instruction from memory.

iii. It reads data from port/memory.

iv. It writes data into port/memory.

v. It supports instruction queuing.

vi. It provides the address relocation facility.

c. To implement these functions, the BIU contains the instruction queue, segment registers, instruction pointer, address summer and bus control logic.

2. Execution Unit (EU) : The execution unit of 8086 tells the BIU from where to fetch instructions or data, decodes instructions and executes instructions. It contains :

i. Control circuitry

ii. Instruction decoder

iii. Arithmetic logic unit (ALU)

iv. Registers organization :

a. Flag registers

b. General purpose registers

c. Pointers and index register.

3. The control circuitry in the EU directs the internal operations. A decoder in the EU translates the instructions fetched from memory into a series of actions which the EU performs.

4. ALU is 16-bit. It can add, subtract, AND, OR, XOR, increment, decrements, complement and shift binary numbers.

Que 3.3. Differentiate among address and data buses of 8085 and 8086 microprocessor ? **AKTU 2014-15, Marks 05**

OR

How does the asynchronous behavior of EU and BIU increase the throughput of 8086 microprocessor ? **AKTU 2018-19, Marks 3.5**

Answer

A. Difference :

S. No.	8085	8086
1.	Address bus is of 16 bits.	Address bus is of 20 bits.
2.	8085 can address 65,536 different memory locations.	8086 can address 10,48,576 different memory locations.
3.	Address bus is multiplexed with 8 bit data bus.	Address bus is multiplexed with 16 bit data bus.
4.	MSB of address goes through Address bus A_7-A_0 and LSB goes through multiplexed data bus AD_0-AD_7.	MSB of address goes through Address bus AD_{15}-AD_8 and LSB goes through multiplexed data bus AD_0-AD_7. A_{16}-A_{19}/S_3-S_6 are the 4 address/status buses.
5.	Data bus is 8 bits long.	Data bus is 16 bits long.

B. Behaviour :

1. The 8086 CPU logic has been partitioned into two functional units namely Bus Interface Unit (BIU) and Execution Unit (EU).

2. The major reason for this separation is to increase the processing speed of the processor.

Scanned by TapScanner

3. The BIU has to interact with memory and input and output devices in fetching the instructions and data required by the EU.
4. EU is responsible for executing the instructions of the programs and to carry out the required processing.

PART-2

Register Organization, Memory Addressing, Memory Segmentation.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.4. **Draw the register organization of 8086 and explain the significance of each register.** **AKTU 2014-15, Marks 10**

OR

Explain pipelining architecture of 8086 microprocessor. Show bit-wise flag register of 8086 and explain each of them.

AKTU 2018-19, Marks 07

Answer

A. Architecture of 8086 : Refer Q. 3.2, Page 3–2C, Unit-3.

B. General data registers :

1. The registers AX, BX, CX and DX are the general purpose 16-bit registers.
2. Each 16-bit register can be split into two 8-bit registers. Letter L and H specify the lower and higher bytes.
3. These registers are either used for holding data, variables, and intermediate results temporarily, or as counters.

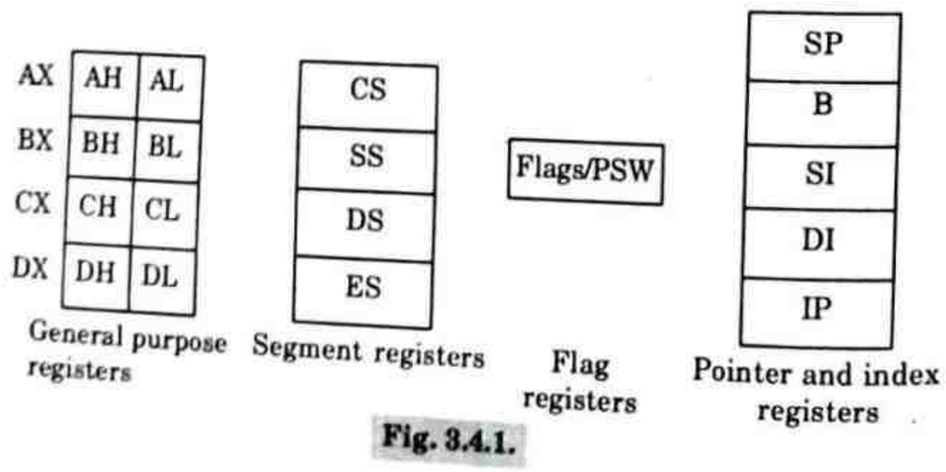


Fig. 3.4.1.

C. Segment registers : There are four types of segment registers :

1. **CS :** The code segment register is used for addressing a memory location in code segment of the memory, where the executable program is stored.
2. **DS :** The data segment register points to the data segment of the memory, where the data is resided.
3. **ES :** The extra segment refers to a segment which essentially is another data segment of the memory. Thus, the extra segment also contains data.
4. **SS :** The stack segment register is used for addressing stack segment of memory. The stack segment is that segment of memory which is used to store stack data.

D. Pointer and Index registers :

1. The pointers contain offset within the particular segments.
2. The pointers IP, BP and SP usually contain offsets within the code, data and stack segments respectively.
3. The index registers are used as general purpose registers as well as for offset storage.
4. The register SI is generally used to store the offset of source data in data segment while the register DI is used to store the offset of destination in data or extra segment.

E. Flag registers :

i. The 8086 flag register contents indicate the result of computations in the ALU. It also contains some flag bits to control the CPU operations.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
×	×	×	×	O	D	I	T	S	Z	×	AC	×	P	×	CY

ii. The description of each flag bit is as follows :

1. **Sign flag (S) :** This flag is set, when the result of any computation is negative. For signed computations, the sign flag equals the MSB of the result.
2. **Zero flag (Z) :** This flag is set, if the result of the computation or comparison performed by the previous instruction is zero.
3. **Parity flag (P) :** This flag is set to 1, if the lower byte of the result contains even number of 1's.
4. **Carry flag (CY) :** This flag is set, when there is a carry out of MSB in case of addition or borrow in case of subtraction.
5. **Trap flag (T) :** If this flag is set, the processor enters the single step execution mode. In other words, a trap interrupt is generated after execution of each instruction.

Scanned by TapScanner

6. **Interrupt flag (I) :** If this flag is set, the maskable interrupts are recognized by the CPU, otherwise they are ignored.
7. **Direction flag (D) :** This is used by string manipulation instructions. If this flag bit is '0', the string is processed beginning from the lowest address to the highest address, means auto-incrementing mode. Otherwise, the string is processed from the highest address towards the lowest address means auto-decrementing mode.
8. **Auxiliary carry flag (AC) :** This is set, if there is a carry from the lowest nibble means bit three.
9. **Overflow flag (O) :** This flag is set, if an overflow occurs.

Que 3.5. What do you mean by memory addressing ? Explain memory segmentation.

Answer

A. Memory addressing :

1. It is the method of specifying the memory location of an operand.
2. It explains how the memory space is available to the processor.

B. Types :

i. **Linear addressing :** In linear addressing the entire memory space is available to the processor in one linear array.

ii. **Segmented addressing or Memory segmentation :**

1. In the segmented addressing, on the other hand, the available memory space is divided into "chunks" called segments. Such a memory is known as segmented memory.
2. In 8086 system the available memory space is 1 Mbytes. This memory is divided into number of logical segments.
3. Each segment is 64 kbytes in size and addressed by one of the segment registers.
4. To address a specific memory location within a segment we need an offset address.
5. The offset address is also 16-bit wide and it is provided by one of the associated pointer or index register.

Que 3.6. Differentiate between physical and offset or effective addresses in 8086.

AKTU 2014-15, Marks 05

Answer

1. Physical address is a memory address that is represented in the form of a binary number on the address bus circuitry in order to enable the data bus to access a particular storage cell of main memory, or a register of memory mapped I/O device.
2. The offset address bus is used to determine the memory location distance from starting address within the memory segment.
3. It is be calculated from segment address and offset address as,

   Physical address = Segment address × 16 + Effective address
4. It can be calculated depending upon addressing modes. It will be different for different addressing modes.

Que 3.7. Explain the addressing capability of 8085 and 8086 microprocessor. How 20 bit address of memory is addressed ?

AKTU 2016-17, Marks 05

Answer

A. Addressing capability of 8085 : Refer Q. 2.11, Page 2–16C, Unit-2.

B. Addressing capability of 8086 :

1. In 8086 microprocessor the address bus is of 20 bit wide. Hence the addressing capability of 8086 can be calculated as,

$$= 2^{20} \times 1 \text{ Byte} = 1{,}048{,}576 \text{ Byte}$$

$$= 1024 \text{ kB (where 1 k = 1024 Bytes)}$$

C. Addressing of 20 bit address :

1. 8086's BIU produces the 20-bit physical memory address by combining a 16-bit segment address with a 16-bit offset address.
2. There are four 16-bit segment registers, viz., the code segment (CS), the stack segment (SS), the extra segment (ES), and the data segment (DS).
3. These segment registers hold the corresponding 16-bit segment addresses.
4. A segment address is the upper 16-bits of the starting address of that segment.
5. The lower 4-bits of the starting address of a segment are always zero.
6. The offset address is held by another 16-bit register.
7. The physical 20-bit address is calculated by shifting the segment address 4-bit left and then adding that to the offset address.

Scanned by TapScanner

PART-3

Operating Modes.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.8. **Explain minimum mode operation of 8086 microprocessor with block diagram.** **AKTU 2017-18, Marks 10**

Answer

1. In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/ $\overline{MX}$ pin to logic 1.
2. In this mode, all the control signals are given out by the microprocessor chip itself.
3. There is a single microprocessor in the minimum mode system.
4. The remaining components in the system are latches, transceivers, clock generator, memory and I/O devices.
5. The latches are generally buffered output D-type flip-flops, like, 74LS373 or 8282.
6. They are used for separating the valid address from the multiplexed address/data signals and are controlled by the ALE signal generated by 8086.
7. Transceivers are the bidirectional buffers and sometimes they are called data amplifiers.
8. They are required to separate the valid data from the time multiplexed address/ data signal.
9. They are controlled by two signals, namely, $\overline{DEN}$ and DT/R.
10. The $\overline{DEN}$ signal indicates that the valid data is available on the data bus, while DT/R indicates the direction of data, *i.e.*, from / to the processor.
11. The system contains memory for the monitor and users program storage.
12. Usually, EPROMS are used for monitor storage, while RAMs for users program storage.
13. A system may contain I/O devices for communication with the processor as well as some special purpose I/O devices.

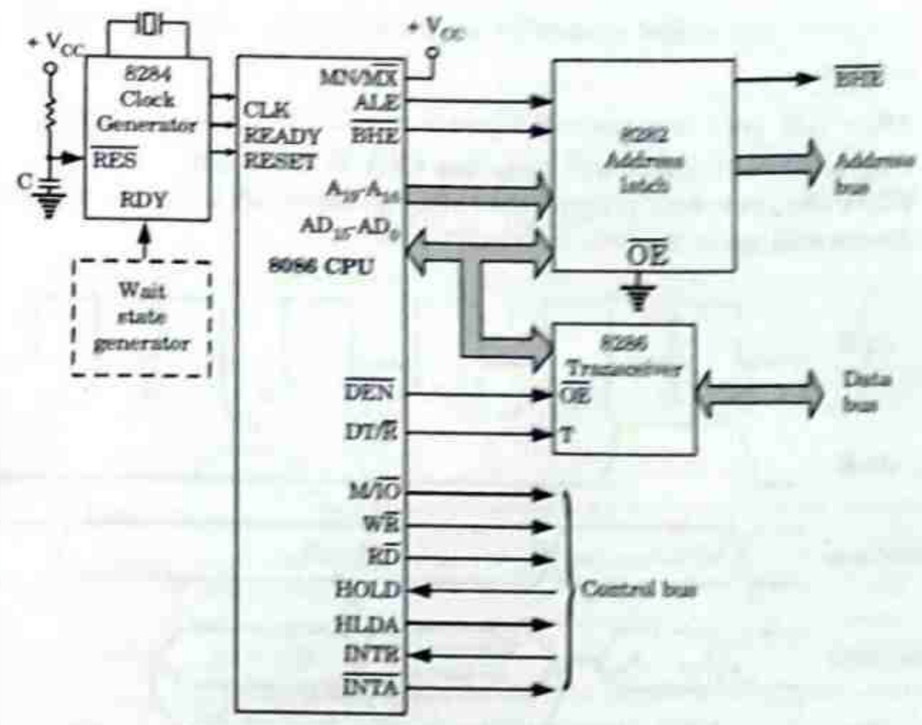


Fig. 3.8.1. Minimum mode 8086 system.

14. The clock generator (IC8284) generates the clock from the crystal oscillator and then shapes it to make it more precise so that it can be used as an accurate timing reference for the system.
15. The clock generator also synchronizes some external signals with the system clock.

Que 3.9. **Explain the minimum mode read and write bus cycle with proper timing diagram.**

Answer

A. Read cycle :

1. The read cycle begins the T_1 with the assertion of the address latch enable (ALE) signal and also M/ $\overline{IO}$ signal.
2. During the negative going edge of this signal, the valid address is latched on the local bus.
3. The $\overline{BHE}$ and A_0 signals addresses low, high or both bytes.
4. From T_1 to T_4, the M/ $\overline{IO}$ signal indicate a memory or I/O operation.
5. At T_2, the address is removed from the local bus and is sent to the output.
6. The bus is then tristated.
7. The read $\overline{RD}$ control signal is also activated in T_2.

Scanned by TapScanner

8. The read $\overline{RD}$ signal causes the addressed device to enable its data bus drivers.
9. After $\overline{RD}$ goes low, the valid data is available on the data bus.
10. The addressed device will drive the READY line high.
11. When the processor returns the read signal to high level, the addressed device will again tristate its bus drivers.

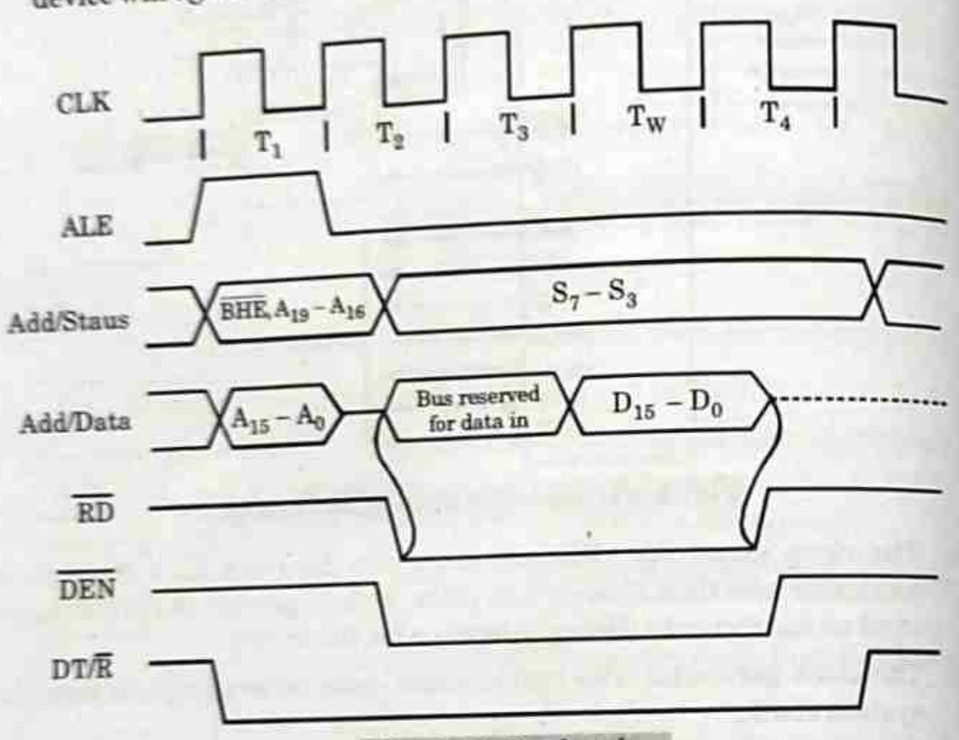


Fig. 3.9.1. Read cycle.

B. Write cycle :

1. A write cycle also begins with the assertion of ALE and the emission of the address.
2. The M/$\overline{IO}$ signal is again asserted to indicate a memory or I/O operation.
3. In T_2, after sending the address in T_1, the processor sends the data to be written to the addressed location.
4. The data remains on the bus until middle of T_4 state.
5. The $\overline{WR}$ becomes active at the beginning of T_2.
6. The $\overline{BHE}$ and A_0 signals are used to select the proper byte or bytes of memory or I/O word to be read or written.
7. The M/$\overline{IO}$, $\overline{RD}$ and $\overline{WR}$ signals indicate the types of data transfer as described in Table. 3.9.1.

Table 3.9.1.

M / $\overline{IO}$	$\overline{RD}$	$\overline{WR}$	Transfer type
0	0	1	I/O read
0	1	0	I/O write
1	0	1	memory read
1	1	0	memory write

8. The timing diagram of write cycle is shown in Fig. 3.9.2.

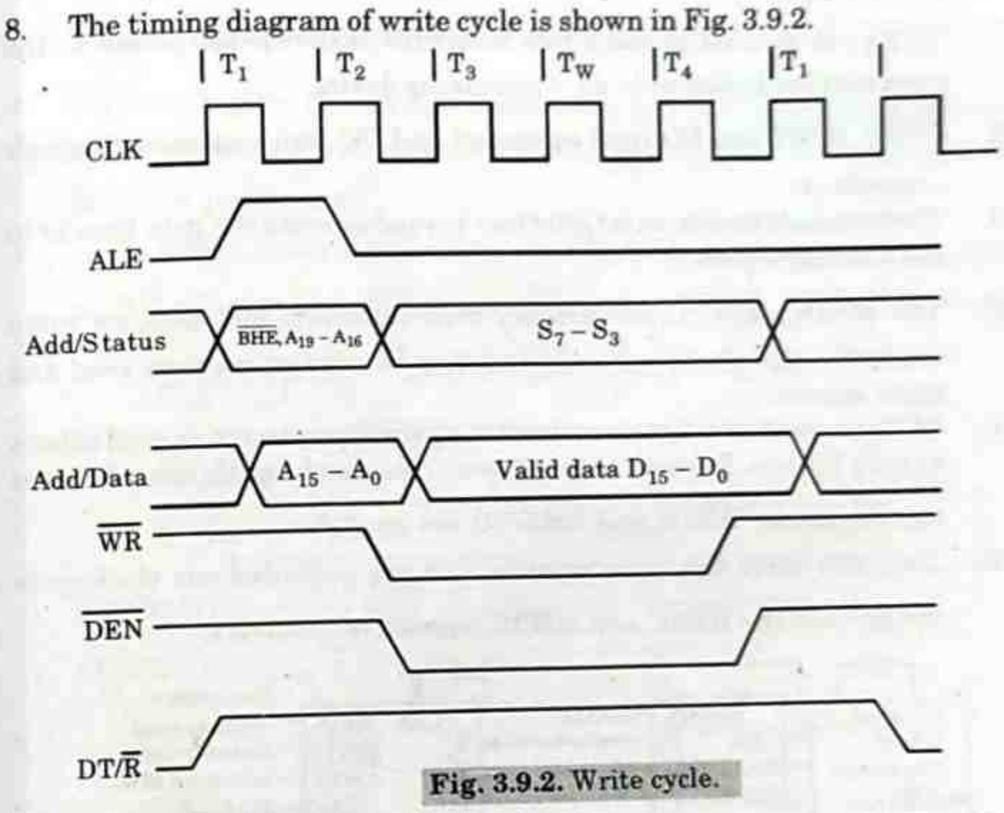


Fig. 3.9.2. Write cycle.

Que 3.10. **Explain maximum mode operation of 8086 microprocessor with block diagram.**

Answer

1. In the maximum mode, the 8086 is operated by strapping the MN / $\overline{MX}$ pin to ground. In this mode, the processor derives the status signals $\overline{S}_2, \overline{S}_1$ and $\overline{S}_0$.
2. Another chip called bus controller derives the control signals using this status information.
3. The basic functions of the bus controller chip IC8288, is to derive control signals like $\overline{RD}$ and $\overline{WR}$ (for memory and I/O devices), $\overline{DEN}$, DT / $\overline{R}$

Scanned by TapScanner

ALE, etc., using the information made available by the processor on the status lines.

4. The bus controller chip has input lines $\overline{S}_2, \overline{S}_1$ and $\overline{S}_0$ and CLK, which are driven by the CPU.
5. $\overline{AEN}$ and IOB are generally grounded. CEN pin is usually tied to + 5V.
6. The significance of the MCE/PDEN output depends upon the status of the IOB pin.
7. $\overline{INTA}$ pin is used to issue two interrupt acknowledge pulses to the interrupt controller or to an interrupting device.
8. $\overline{IORC}$, $\overline{IOWC}$ are I/O read command and I/O write command signals respectively.
9. These signals enable an IO interface to read or write the data from or to the addressed port.
10. The $\overline{MRDC}$, $\overline{MWTC}$ are memory read command and memory write command signals respectively and may be used as memory read and write signals.
11. All these command signals instruct the memory to accept or send data to or from the bus. For both of these write command signals, the advanced signals namely $\overline{AIOW}$ and $\overline{AMWTC}$ are available.
12. They also serve the same purpose, but are activated one clock cycle earlier than the $\overline{IOWC}$ and $\overline{MWTC}$ signals, respectively.

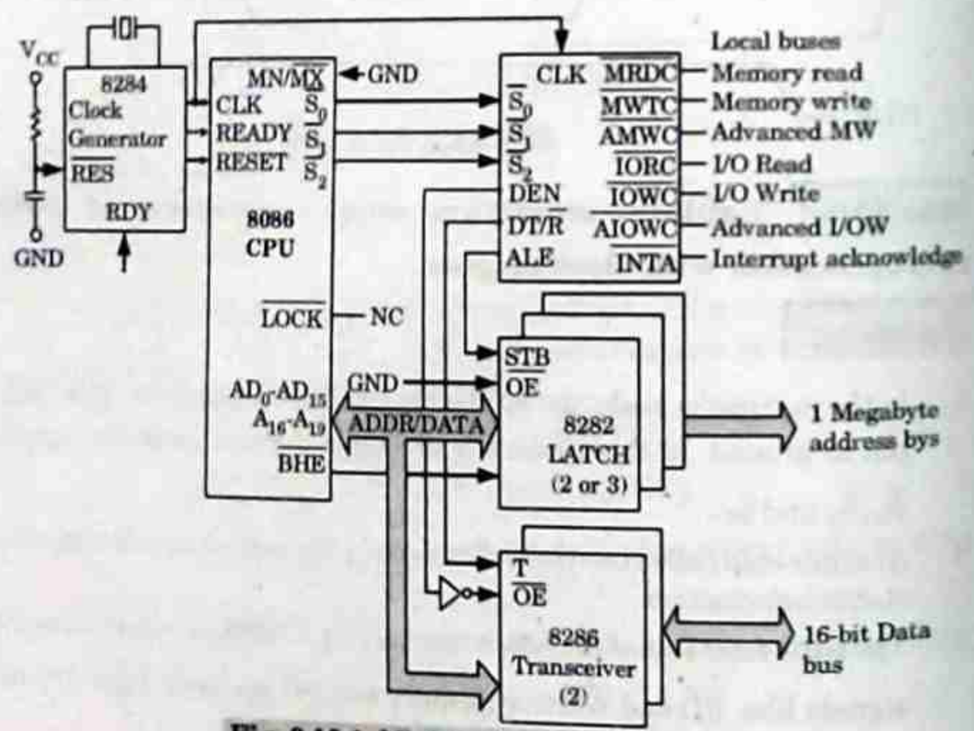


Fig. 3.10.1. Minimum mode 8086 system.

Que 3.11. **Explain minimum and maximum operating modes of 8086 with timing diagram.** **AKTU 2017-18, Marks 10**

Answer

A. Minimum operating modes : Refer Q. 3.9, Page 3–10C, Unit-3.

B. Maximum operating modes :

1. S_0, $\overline{S}_1$, $\overline{S}_2$ are set at the beginning of bus cycle.
2. On detecting the change on passive state $S_0 = \overline{S}_1 = \overline{S}_2 = 1$, the 8288 bus controller will output a pulse on its ALE and apply a required signal to its DT/$\overline{R}$ pin during T_1.
3. In T_2, 8288 will set DEN = 1 thus enabling transceiver. For an input, 8288 it will activates $\overline{MRDC}$ or $\overline{IORC}$.

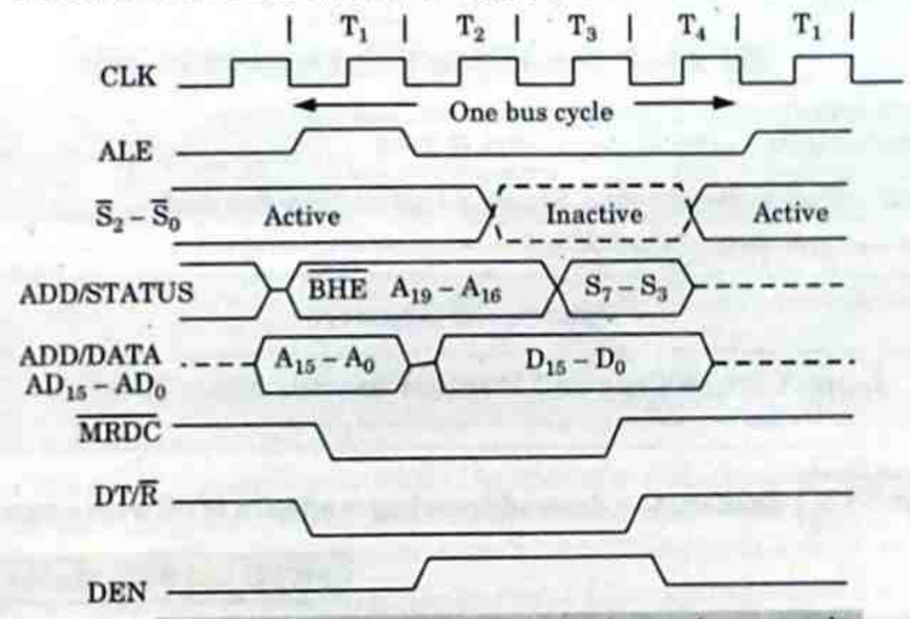


Fig. 3.11.1. Memory read timing in maximum mode.

4. These signals are activated until T_4. For an output, the $\overline{AMWC}$ or $\overline{IOWC}$ is activated from T_3 to T_4.
5. The status bits $\overline{S}_2$ to $\overline{S}_0$ remain active until T_3, and become passive during T_3 and T_4.
6. If ready input is not activated before T_3, wait state will be inserted between T_3 and T_4.

Scanned by TapScanner

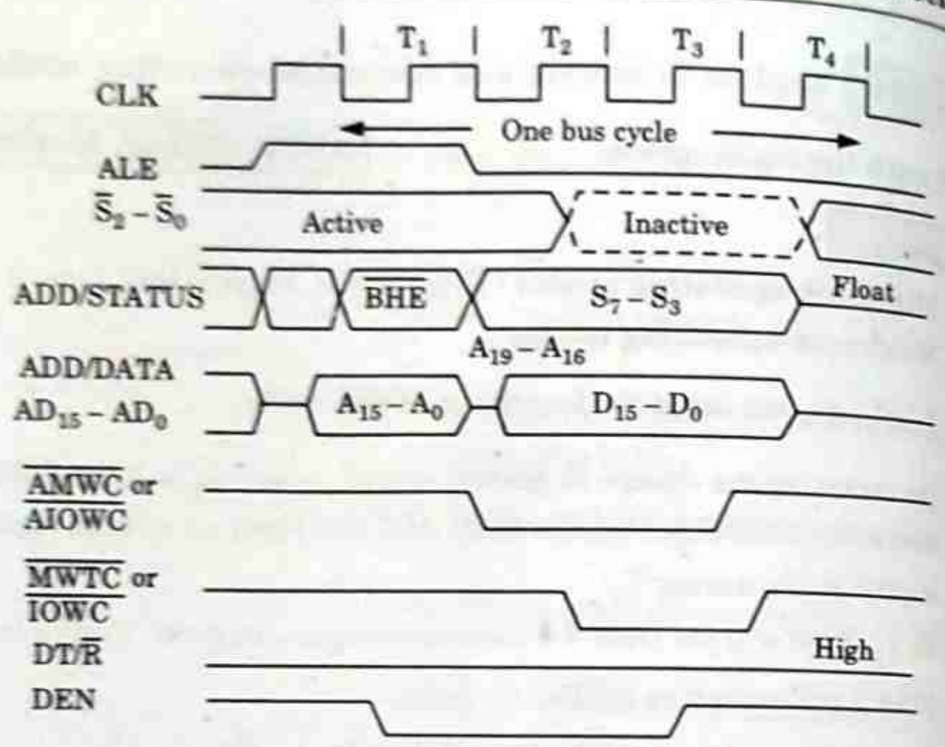


Fig. 3.11.2. Memory write timing in maximum mode.

PART-4

Instruction Set and Instruction Format.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.12. **Explain the data addressing modes in 8086 with example.**

AKTU 2014-15, Marks 10

Answer

1. **Immediate :**

i. In this type of addressing, immediate data is a part of instruction and appears in the form of successive byte or bytes.

ii. **Example :** MOV AX, 0005H

2. **Direct :**

i. In the direct addressing mode, a 16-bit memory address (offset) is directly specified in the instruction as a part of it.

ii. **Example :** MOV AX, [5000H]

3. **Register :**

i. In register addressing mode, the data is stored in a register and it is referred using the particular register.

ii. All the registers, except IP, may be used in this mode.

iii. **Example :** MOV BX, AX.

4. **Register Indirect :**

i. The address of memory location which contains data or operand known determined in an indirect way, using the offset registers. This mode of addressing is known as register indirect mode.

ii. In this mode, the offset address of data is in either BX or SI or DI register.

iii. The default segment is either DS or ES.

iv. **Example :** MOV AX, [BX]

5. **Implied :**

i. In this mode, instruction itself will specify the data to be operated by the instruction.

ii. **Example :** CLC.

Que 3.13. **Differentiate between data addressing and branch addressing in 8086. Explain the branch addressing modes with example.**

AKTU 2016-17, Marks 10

Answer

A. **Branch addressing modes :**

1. **Intrasegment direct mode :**

i. In this mode, the address to which the control is to be transferred lies in the same segment in which the control transfer instruction lies and appears directly in the instruction as an immediate displacement value.

ii. In this addressing mode, the displacement is computed relative to the content of the instruction pointer IP.

2. **Intrasegment indirect mode :**

i. In this mode, the displacement to which the control is to be transferred is in the same segment in which the control transfer instruction lies, but it is passed to the instruction indirectly.

ii. Here, the branch address is found as the content of a register or a memory location.

3. **Intersegment direct mode :**

i. In this mode, the address to which the control is to be transferred is in different segment.

Scanned by TapScanner

ii. This addressing mode provides a means of branching from one code segment to another code segment.

iii. Here, the CS and IP of the destination address are specified directly in the instruction.

4. Intersegment indirect mode :

i. In this mode, the address to which the control is to be transferred lies in a different segment and it is passed to the instruction indirectly.

ii. The starting address of the memory block may be referred using any of the addressing modes, except immediate mode.

B. Difference :

S. No.	Data addressing	Branch addressing
1.	Data addressing modes are for defining a data operand in the instruction.	Branch addressing modes are the ways of specifying a branch address in control transfer instruction.
2.	These indicate data transfer from register to register and register to memory.	These indicate the branch address in the call and jump instruction.

Que 3.14. **Describe the various addressing modes of 8086 with suitable example of each.** **AKTU 2015-16, Marks 10**

Answer

1. **Data addressing modes :** Refer Q. 3.12, Page 3–15C, Unit-3.
2. **Branch addressing modes :** Refer Q. 3.13, Page 3–16C, Unit-3.
3. **Indexed :**

i. In this addressing mode, offset of the operand is stored in one of the index register.

ii. DS and ES are the default segments for index registers SI and DI respectively.

iii. **Example :** MOV AX, [SI]

4. **Register relative :**

i. In this addressing mode, the data is available at an effective address formed by adding an 8-bit or 16-bit displacement with the content of any one of the registers BX, SI and DI in the default segment (ES or DS).

ii. **Example :** MOV AX, 50H[BX]

5. **Based Indexed :**

i. The effective address of data is formed, in this addressing mode, by adding content of a base register (BX or BP) to the content of an index register (SI or DI).

ii. The default segment register may be ES or DS.

iii. **Example :** MOV AX, [BX] [SI]

6. **Relative Based Indexed :**

i. The effective address is formed by adding on 8 or 16-bit displacement with the sum of contents of any one of the base registers (BX or BP) and any one of the index registers in a default segment.

ii. **Example :** MOV AX, 50H [BX] [SI].

Que 3.15. **Explain instruction formats of 8086. Also explain the function of special bits used in instruction format.** **AKTU 2016-17, Marks 10**

Answer

A. Instruction format :

1. A machine language instruction format has one or more number of fields associated with it. The first field is called operation code field or opcode field which indicates the type of operation to be performed by the CPU.
2. The instruction format also contains other field known as operand fields.
3. The CPU executes the instructions using the information which resides in these fields.
4. The length of instruction may vary from one byte to six bytes.

B. Types :

1. **One byte instruction :** This format is only one byte long and may have implied data or register operands. The least significant 3-bits of opcode are used for specifying the register operand.
2. **Resister to Register :** This format is 2-byte long. The first byte of code specified the operation code and width of operand specified in bit. The second byte of code shows the register operands.
3. **Register to/from memory with no displacement :** This format is also 2-bytes long and similar to register format except for MOD field.
4. **Register to/from memory displacement :** This type of instruction format contains one or two additional bytes for displacement along with 2-byte the format of register to/from memory without displacement.
5. **Immediate operand to register :** In this the first byte as well as 3-bits from second byte which are used for REG field in case of Register to Register format are used for opcode. It also contains one or two bytes of immediate data.
6. **Immediate operand to memory with 16-bit displacement :** This format requires 5 to 6-byte for coding. The first two byte contains

Scanned by TapScanner

information regarding OPCODE, MOD, and R/M field. The remaining 4-byte contains 2-byte of displacement and 2-byte of data.

C. Functions of special bits :

1. **W-bit :** The *W*-bit in the opcode of an instruction specifies whether instruction is a byte instruction ($W = 0$) or a word instruction ($W = 1$).
2. **D-bit :** The *D*-bit in the opcode of the instruction indicates that the register specified within the instruction is a source register ($D = 0$) or destination register ($D = 1$).
3. **S-bit :** An 8-bit 2's complement number can be extended to a 16-bit 2's complement number by making all of the bits in the higher-order byte equal the most significant bit in the low order byte. This is known as sign extension.
4. **V-bit :** *V*-bit decides the number of shifts for rotate and shift instructions. If $V = 0$, then count = 1; if $V = 1$, the count is in CL register.
5. **Z-bit :** It is used for string primitives such as REP for comparison with ZF Flag.

PART-5

Types of Instructions.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.16. **What do you mean by Data transfer instruction ?**

Answer

A. Data transfer function : These are the instructions which are used to copy the data from one location to another location.

B. Instructions :

1. **MOV :** This instruction is used to copy the byte or word from the provided source to the provided destination.
2. **PUSH :** This instruction is used to put a word at the top of the stack.
3. **POP :** This instruction is used to get a word from the top of the stack to the provided location.
4. **PUSHA :** This instruction is used to put all the registers into the stack.
5. **POPA :** This instruction is used to get words from the stack to all registers.
6. **XCHG :** This instruction is used to exchange the data from two locations.
7. **XLAT :** This instruction is used to translate a byte in AL using a table in the memory.

8. **IN :** This instruction is used to read a byte or word from the provided port to the accumulator.
9. **OUT :** This instruction is used to send out a byte or word from the accumulator to the provided port.
10. **LEA :** This instruction is used to load the address of operand into the provided register.
11. **LDS :** This instruction is used to load DS register and other provided register from the memory
12. **LES :** This instruction is used to load ES register and other provided register from the memory.
13. **LAHF :** This instruction is used to load AH with the low byte of the flag register.
14. **SAHF :** This instruction is used to store AH register to low byte of the flag register.
15. **PUSHF :** This instruction is used to copy the flag register at the top of the stack.
16. **POPF :** This instruction is used to copy a word at the top of the stack to the flag register.

Que 3.17. **What do you understand by Arithmetic Instructions ?**

OR

Explain the division and multiplication instructions of 8086.

AKTU 2014-15, Marks 05

Answer

A. Arithmetic Instructions : These are the instructions which are used to perform the arithmetic operations like addition, subtraction, multiplication and division along with ASCII and decimal adjust instruction.

B. Instructions :

1. **ADD :** This instruction is used to add the provided byte to byte/word to word.
2. **ADC :** This instruction is used to add with carry.
3. **INC :** This instruction is used to increment the provided byte/word by 1.
4. **AAA :** This instruction is used to adjust ASCII after addition.
5. **DAA :** This instruction is used to adjust the decimal after the addition/subtraction operation.
6. **SUB :** This instruction is used to subtract the byte from byte/word from word.
7. **SBB :** This instruction is used to perform subtraction with borrow.

Scanned by TapScanner

8. **DEC :** This instruction is used to decrement the provided byte/word by 1.
9. **NPG :** This instruction is used to negate each bit of the provided byte/word and add 1/2's complement.
10. **CMP :** This instruction is used to compare 2 provided byte/word.
11. **AAS :** This instruction is used to adjust ASCII codes after subtraction.
12. **DAS :** This instruction is used to adjust decimal after subtraction.
13. **AAM :** This instruction is used to adjust ASCII codes after multiplication.
14. **AAD :** This instruction is used to adjust ASCII codes after division.
15. **CBW :** This instruction is used to fill the upper byte of the word with the copies of sign bit of the lower byte.
16. **CWD :** This instruction is used to fill the upper word of the double word with the sign bit of the lower word.

Division and multiplication instructions :

17. **MUL (Unsigned Multiplication Byte or Word) :**

i. This instruction multiplies an unsigned byte or word by the contents of AL.

ii. The most significant word of the result is stored in DX, while the least significant word of the result is stored in AX.

18. **IMUL (Signed Multiplication) :**

i. This instruction multiplies a signed byte from same source and a signed byte in AL, or a signed word from same source and a signed word in AX.

ii. When a signed byte is multiplied by AL, a signed result will be put in AX.

19. **DIV (Unsigned Division) :**

i. This instruction performs unsigned division.

ii. It divides an unsigned word or double word by a 16-bit or 8-bit operand.

iii. The result (Quotient) will be in AL while AH will contain the remainder.

20. **IDIV (Signed Division) :** This instruction is used to divide a signed word by a signed byte, or to divide a signed double word (32-bits) by a signed word, rest all is similar to DIV instruction.

Que 3.18. **What do you understand by logical instruction ?**

OR

Explain the following Instruction of 8086 with example.

i. LDS	**ii. SBB**	**iii. ADD**
iv. MOV	**v. ADC**	**vi. XCHG**
vii. PUSH	**viii. POP**	**ix. XLAT**
x. IN		

AKTU 2014-15, Marks 10

OR

Explain the following instructions in 8086 with example.

i. LDS	**ii. SBB**	**iii. MUL**
iv. IDIV	**v. ADC**	**vi. CMPS**
vii. TEST	**viii. XOR**	**ix. RCR**

AKTU 2016-17, Marks 10

Answer

A. **Logical instruction :** These are the instructions which are used to perform the logical operations such as logical AND, logical OR, logical EX-OR etc.

B. **Instructions :**

1. **NOT :** This instruction is used to invert each bit of a byte or word.
2. **AND :** This instruction is used for adding each bit in a byte/word with the corresponding bit in another byte/word.
3. **OR :** This instruction is used to multiply each bit in a byte/word with the corresponding bit in another byte/word.
4. **XOR :** This instruction is used to perform Exclusive-OR operation over each bit in a byte/word with the corresponding bit in another byte/word.
5. **TEST :** This instruction is used to add operands to update flags, without affecting operands.
6. **SHL/SAL :** This instruction is used to shift bits of a byte/word towards left and put zero(S) in LSBs.
7. **SHR :** This instruction is used to shift bits of a byte/word towards the right and put zero(S) in MSBs.
8. **SAR :** This instruction is used to shift bits of a byte/word towards the right and copy the old MSB into the new MSB.
9. **ROL :** This instruction is used to rotate bits of byte/word towards the left, *i.e.*, MSB to LSB and to Carry Flag [CF].
10. **ROR :** This instruction is used to rotate bits of byte/word towards the right, *i.e.*, LSB to MSB and to Carry Flag [CF].
11. **RCR :** This instruction is used to rotate bits of byte/word towards the right, *i.e.*, LSB to CF and CF to MSB.
12. **RCL :** This instruction is used to rotate bits of byte/word towards the left, *i.e.*, MSB to CF and CF to LSB.
13. **Other :**

i. **LDS :** Refer Q. 3.16, Page 3–19C, Unit-3.

ii. **SBB :** Refer Q. 3.17, Page 3–20C, Unit-3.

iii. **MUL :** Refer Q. 3.17, Page 3–20C, Unit-3.

iv. **IDIV :** Refer Q. 3.17, Page 3–20C, Unit-3.

v. **ADC :** Refer Q. 3.17, Page 3–20C, Unit-3.

vi. **CMPS :** This instruction compares the two strings.

Scanned by TapScanner

vii. **MOV :** Refer Q. 3.16, Page 3–19C, Unit-3.

viii. **XCHG :** Refer Q. 3.16, Page 3–19C, Unit-3.

ix. **PUSH :** Refer Q. 3.16, Page 3–19C, Unit-3.

x. **POP :** Refer Q. 3.16, Page 3–19C, Unit-3.

xi. **XLAT :** Refer Q. 3.16, Page 3–19C, Unit-3.

xii. **IN :** Refer Q. 3.16, Page 3–19C, Unit-3.

xiii. **ADD :** Refer Q. 3.17, Page 3–20C, Unit-3.

Que 3.19. **What do you understand by string manipulation instructions ?**

Answer

A. **String manipulation instructions :** These are the instructions used for performing various string operations like comparison of two strings, moving of string from one location to another location, scanning of string etc.

B. **Instructions :**

1. **REP :** This instruction is used to repeat the given instruction till CX ≠ 0.
2. **REPE/REPZ :** This instruction is used to repeat the given instruction until CX = 0 or zero flag ZF = 1.
3. **REPNE/REPNZ :** This instruction is used to repeat the given instruction until CX = 0 or zero flag ZF = 1.
4. **MOVS/MOVSB/MOVSW :** This instruction is used to move the byte/word from one string to another.
5. **COMS/COMPSB/COMPSW :** This instruction is used to compare two string bytes/words.
6. **INS/INSB/INSW :** This instruction is used as an input string/byte/word from the I/O port to the provided memory location.
7. **OUTS/OUTSB/OUTSW :** This instruction is used as an output string/byte/word from the provided memory location to the I/O port.
8. **SCAS/SCASB/SCASW :** This instruction is used to scan a string and compare its byte with a byte in AL or string word with a word in AX.
9. **LODS/LODSB/LODSW :** This instruction is used to store the string byte into AL or string word into AX.

Que 3.20. **What do you understand by branch control instruction ?**

Answer

A. **Program Execution Transfer Instructions (Branch and Loop Instructions) :** These instructions are used to transfer the instructions during an execution.

B. **Instructions :**

1. **CALL :** This instruction is used to call a procedure and save their return address to the stack.
2. **RET :** This instruction is used to return from the procedure to the main program.
3. **JMP :** This instruction is used to jump to the provided address to proceed to the next instruction.
4. **JA/JNBE :** This instruction is used to jump if above/not below/equal instruction satisfies.
5. **JAE/JNB :** This instruction is used to jump if above/not below instruction satisfies.
6. **JBE/JNA :** This instruction is used to jump if below/equal/ not above instruction satisfies.
7. **JC :** This instruction is used to jump if carry flag CF = 1.
8. **JE/JZ :** This instruction is used to jump if equal/zero flag ZF = 1.
9. **JG/JNLE :** This instruction is used to jump if greater/not less than/equal instruction satisfies.
10. **JGE/JNL :** This instruction is used to jump if greater than/equal/not less than instruction satisfies.
11. **JL/JNGE :** This instruction is used to jump if less than/not greater than/equal instruction satisfies.
12. **JLE/JNG :** This instruction is used to jump if less than/equal/if not greater than instruction satisfies.
13. **JNC :** This instruction is used to jump if no carry flag (CF = 0).
14. **JNE/JNZ :** This instruction is used to jump if not equal/zero flag ZF = 0.
15. **JNO :** This instruction is used to jump if no overflow flag OF = 0.
16. **JNP/JPO :** This instruction is used to jump if not parity/parity odd PF = 0.
17. **JNS :** This instruction is used to jump if not sign SF = 0.
18. **JO :** This instruction is used to jump if overflow flag OF = 1.
19. **JP/JPE :** This instruction is used to jump if parity/parity even PF = 1.
20. **JS :** This instruction is used to jump if sign flag SF = 1.

Que 3.21. **What do you understand by processor control instruction ?**

Answer

A. **Processor Control Instructions :** These instructions are used to control the processor action by setting/resetting the flag values.

Scanned by TapScanner

B. Instructions :

1. **STC :** This instruction is used to set carry flag CF to 1.
2. **CLC :** This instruction is used to clear/reset carry flag CF to 0.
3. **CMC :** This instruction is used to put complement at the state of carry flag CF.
4. **STD :** This instruction is used to set the direction flag DF to 1.
5. **CLD :** This instruction is used to clear/reset the direction flag DF to 0.
6. **STI :** This instruction is used to set the interrupt enable flag to 1, *i.e.*, enable INTR input.
7. **CLI :** This instruction is used to clear the interrupt enable flag to 0, *i.e.*, disable INTR input.

Que 3.22. Explain iteration control instruction.

Answer

A. Iteration Control Instructions :

These instructions are used to execute the given instructions for number of times.

B. Instructions :

1. **LOOP :** This instruction is used to loop a group of instructions until the condition satisfies, *i.e.*, CX = 0
2. **LOOPE/LOOPZ :** This instruction is used to loop a group of instructions till it satisfies ZF = 1 & CX = 0
3. **LOOPNE/LOOPNZ :** This instruction is used to loop a group of instructions till it satisfies ZF = 0 & CX = 0
4. **JCXZ :** Used to jump to the provided address if CX = 0

PART-6

Interrupts : Hardware and Software Interrupts.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.23. Explain the interrupts and interrupt sequence in 8086. Also explain the following instructions :

i. **INT 30H**

ii. **INT 45H**

AKTU 2014-15, Marks 10

Answer

A. Interrupts :

1. Whenever a microprocessor is interrupted, it stops executing its current program and calls a special routine which services the interrupt. The event that causes the interruption is called interrupt and the special routine executed to service the interrupt is called interrupt service routine/procedure.
2. Normally program can be interrupted in three ways :

i. By external signal.

ii. By special instruction in program.

iii. By occurrence of some condition.

B. Interrupt sequence in 8086 :

The Interrupt sequence in an 8086 system is described as follows :

1. One or more IR lines are raised high that set corresponding IRR bits.
2. 8086 resolves priority and sends an INT signal to CPU.
3. The CPU acknowledge with INTA pulse.
4. Upon receiving an INTA signal from the CPU, the highest priority ISR bit is set and the corresponding IRR bit is reset. The 8086 does not drive data during this period.
5. The 8086 will initiate a second INTA pulse. During this period 8086 releases an 8-bit pointer on to a data bus from where it is read by the CPU.
6. This completes the interrupt cycle.
7. The ISR bit is reset at the end of the second INTA pulse if automatic end of interrupt (AEOI) mode is programmed.
8. Otherwise ISR bit remains set until an appropriate EOI command is issued at the end of interrupt subroutine.

C. Instructions :

1. **INT 30H :** INT 30H is shorthand for BIOS interrupt call in an x86-based computer system. It is used to get the current DOS version of the computer system.
2. **INT 45H :** INT 45H is shorthand for BIOS interrupt call in an x86-based computer system. It is used to Lock/Unlock Drive of the computer system.

Que 3.24. Explain the interrupts sequence and types of interrupts in 8086.

AKTU 2016-17, Marks 10

OR

Explain the interrupts sequence and types of interrupt in 8086.

AKTU 2017-18, Marks 10

Scanned by TapScanner

Answer

A. **Interrupt sequence :** Refer Q. 3.23, Page 3–25C, Unit-3.

B. **Types :**

1. **Hardware Interrupt :** Microprocessor 8086 can get interrupt from external signal applied to non maskable interrupt (NMI) input pin or interrupt (INTR) input pin.

2. **Divide by Zero Interrupt (Type 0) :** When the quotient from either a DIV or IDIV instruction is too large to fit in result register, 8086 will automatically execute type 0 interrupt.

3. **Single Step Interrupt (Type 1) :**

i. Type 1 interrupt is a single step TRAP.

ii. In the single step mode, system will execute one instruction and wait for further direction from user.

iii. Then user can examine the contents of register and memory location if they are correct, user can tell the system to execute next instruction.

iv. This feature is useful for debugging.

v. An 8086 system is used in single step mode by setting trap flag and then 8086 will automatically execute type 1 interrupt.

vi. To reset the TRAP flag we have to reset bits 8, this can be done using AND operation.

4. **Software Interrupts (Type 0–255) :**

i. The 8086 INT instruction can be used to cause the 8086 to do one of the 256 possible interrupt types.

ii. The desired interrupt types are specified as a part of the instruction.

iii. For example the instruction INT 32 will cause the 8086 to do a type 32 interrupt response.

iv. The 8086 will push the flag register on the stack, reset TF and IF and push the CS and IP values for the start of interrupt service procedure from interrupt pointer table in memory.

v. The IP value for any interrupt type is always at an address of 4 times the interrupt type and the CS value is at the location two address higher.

5. **Non Maskable Interrupt (Type 2) :**

i. As the name suggest interrupt can't be disabled by any software instruction.

ii. This interrupt is activated by low to high transition on 8086 NMI input pin.

iii. In response 8086 will do type 2 interrupt.

6. **Maskable Interrupt (INTR) :**

i. The 8086 INTR input can be used to interrupt a program execution.

ii. This interrupt can be implemented using two pins INTR and $\overline{INTA}$.

iv. This interrupt can be enabled and disabled using STI (IF = 1) or CLI (IF = 0).

4. When 8086 is reset, interrupt flag is automatically cleared (IF = 0) so, after reset INTR is disabled, user have to execute STI function to enable INTR interrupt.

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. Differentiate among address and data buses of 8085 and 8086 microprocessor ?

Ans. Refer Q. 3.3.

Q. 2. Draw the register organization of 8086 and explain the significance of each register.

Ans. Refer Q. 3.4.

Q. 3. Differentiate between physical and offset or effective addresses in 8086.

Ans. Refer Q. 3.6.

Q. 4. Explain the addressing capability of 8085 and 8086 microprocessor. How 20 bit address of memory is addressed ?

Ans. Refer Q. 3.7.

Q. 5. Explain minimum and maximum operating modes of 8086 with timing diagram.

Ans. Refer Q. 3.11.

Q. 6. Explain the data addressing modes in 8086 with example.

Ans. Refer Q. 3.12.

Q. 7. Describe the various addressing modes of 8086 with suitable example of each.

Ans. Refer Q. 3.14.

Q. 8. Explain instruction formats of 8086. Also explain the function of special bits used in instruction format.

Ans. Refer Q. 3.15.

Scanned by TapScanner

Q. 9. What do you understand by logical instruction ?

Ans. Refer Q. 3.18.

Q. 10. Explain the interrupts sequence and types of interrupts in 8086.

Ans. Refer Q. 3.24.

☺☺☺

4 UNIT

Assembly Language Programming

CONTENTS

Part-1 : Assembly Language Programming based on 8085/8086 : Instruction, Data Transfer, Arithmetic, Logic, Branch Operation, Stacks and Subroutine : Conditional Call and Return Instruction ..... **4–2C to 4–12C**

Part-2 : Programming Techniques : Looping, Counting, Indexing, Counters and Time Delay ..... **4–13C to 4–18C**

Scanned by TapScanner

PART-1

Assembly Language Programming based on 8085/8086 : Instruction, Data Transfer, Arithmetic, Logic, Branch Operation, Stacks and Subroutine : Conditional Call and Return Instruction.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.1. **Why assembly language is used to program microprocessor ? What are the disadvantages of microprocessor ?**

AKTU 2018-19, Marks 07

OR

Explain assembler level programming and draw the flowchart of assembler level programming. **AKTU 2017-18, Marks 10**

Answer

A. Assembly language programming :

1. The writing of program in machine language is very difficult. Hence, to facilitate programmers easily under *n* standable languages have been developed. Assembly language is one of them.
2. A programmer can easily write a program in alphanumeric symbols instead of zeros and ones. Meaningful and easily rememberable symbols are chosen for the purpose.
3. Examples : ADD for addition, SUB for subtraction, CMP for comparison etc. Such symbols are called mnemonics.
4. A program written in mnemonics is known as assembly language program.
5. The writing of a program in assembly language is much easier and faster as compared to the writing of a program in machine language.
6. Both assembly language and machine language are microprocessor-specific. A microprocessor-specific language is known as a low-level language.
7. A program which translates an assembly language program into a machine language program is called an assembler.

B. Disadvantages :

1. The microprocessor has a limitation on the size of data.

2. Most of the microprocessor does not support floating point operations.
3. The main disadvantage is its overheating physically.
4. The microprocessor is does not have any internal peripheral like ROM, RAM and other I/O devices.

C. Developing an assembly language program is a four step process :

i. To specify the source code as per the assembly language definition.
ii. Assemble the program to create the object code.
iii. Link the program to create an executable code.
iv. Test and debug the program.

Fig. 4.1.1 shows the steps involved in developing and executing an assembly language program.

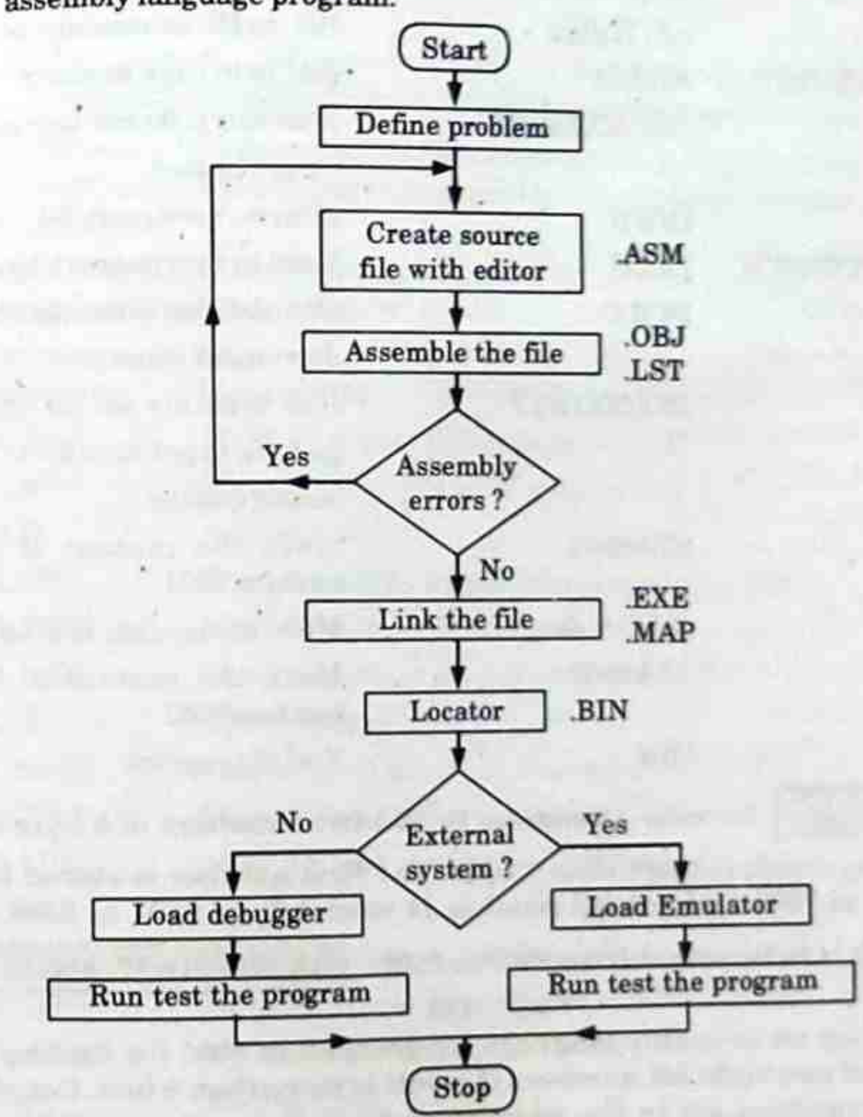


Fig. 4.1.1.

Scanned by TapScanner

Que 4.2. **Develop an assembly language program for 8085 to add 5 numbers of 8 bits, whose sum is of more than 8 bits. The numbers are stored from 8501 to 8505. The result is to be stored in 8601 and 8602.**

AKTU 2016-17, Marks 10

Answer

Label	Mnemonics		Comments
	XRA A	:	Clear (A) to save sum
	MOV B, A	:	Clear (B) to save carry
	MVI C, 05H	:	Set up register C as a counter
	LXI H 8501	:	Set up HL as memory pointer
NXTBYT :	ADD M	:	Add byte from memory
	JNC NXTMEM	:	If no carry, do not increment carry register
	INR B	:	If carry, save carry bit
NXTMEM :	INX H	:	Point to next memory location
	DCR C	:	One addition is completed; decrement counter
	JNZ NXTBYT	:	If all bytes are not yet added, go back to get next byte output display
	STA 8601	:	Move the content of A to location 8601
	MOV A, B	:	Move the content of B to A
	STA 8602	:	Move the content of A to location 8602
	HLT	:	End of program

Que 4.3. **Develop a program to add two numbers of 8 byte long whose result is more than 8 byte. The first number is stored from 7501 to 7508 and second number is stored from 8501 to 8508 and result is to be stored from 9501 to 9509.**

AKTU 2016-17, Marks 10

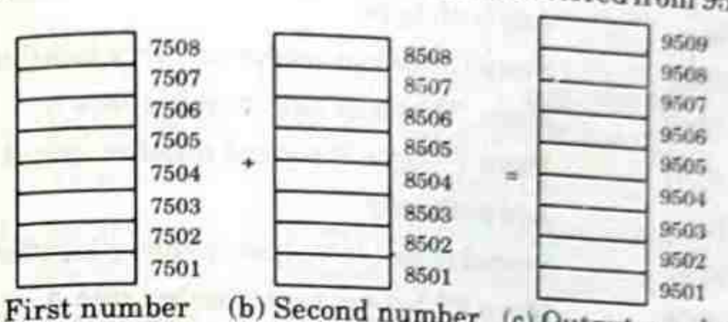
OR

Develop an assembly language programme in 8085 for finding the sum of two eight bit numbers if result is more than 8 bits. Consider that numbers are in the memory and result is to be stored in the memory.

AKTU 2014-15, Marks 10

Answer

1. Given that the first number is stored from 7501 to 7508 and second number is stored from 8501 to 8508.
2. The output of the sum of these numbers is to be stored from 9501 to 9509.



(a) First number (b) Second number (c) Output number

Fig. 4.3.1. Memory location of number and output.

3. For addition of two 8-byte number, we have to perform addition of each byte of each number and the carry generated by the addition is forwarded to the next byte.
4. The sum of each byte is to be stored from 9501 to 9508 and the carry generated at 8th bit is to be stored in 9509.

Program :

Mnemonics		Comments
LDA, 7501	:	Move 1st byte of first number into A
LDB, 8501	:	Move 1st byte of second number into B
ADD B	:	Add both byte
STA, 9501	:	Stored result to output memory location 9501
LDA, 7502	:	Move 2nd byte of first number into A
LDB, 8502	:	Move 2nd byte of second number into B
ADC B	:	Add both byte
STA, 9502	:	Stored result to output memory location 9502
LDA, 7503	:	Move 3rd byte of first number into A
LDB, 8503	:	Move 3rd byte of second number into B
ADC B	:	Add both byte
STA, 9503	:	Stored result to output memory location 9503
LDA, 7504	:	Move 4th byte of first number into A
LDB, 8504	:	Move 4th byte of second number into B
ADC B	:	Add both byte
STA, 9504	:	Stored result to output memory location 9504
LDA, 7505	:	Move 5th byte of first number into A
LDB, 8505	:	Move 5th byte of second number into B

Scanned by TapScanner

ADC B	:	Add both byte
STA, 9505	:	Stored result to output memory location 9505
LDA, 7506	:	Move 6th byte of first number into A
LDB, 8506	:	Move 6th byte of second number into B
ADC B	:	Add both byte
STA, 9506	:	Stored result to output memory location 9506
LDA, 7507	:	Move 7th byte of first number into A
LDB, 8507	:	Move 7th byte of second number into B
ADC B	:	Add both byte
STA, 9507	:	Stored result to output memory location 9507
LDA, 7508	:	Move 8th byte of first number into A
LDB, 8508	:	Move 8th byte of second number into B
ADC B	:	Add both byte
STA, 9508	:	Stored result to output memory location 9508
MVI A, 0	:	Move 0 into A
MVI, B, 0	:	Move 0 into B
ADC B	:	Add both byte
STA, 9509	:	Stored carry to output memory location 9509

Que 4.4. **Write a program to obtain larger of the contents of memory location 2140H and 2141H.**

Answer

Program :

Label	Mnemonics		Comments
	LXI H, 2140H	:	HL pair points at location 2140H.
	MVI M, A6H	:	Store A6H at location 2140H.
	MOV A, M	:	Load accumulator from location 2140H.
	INX H	:	HL pair points at next location *i.e.*, 2141H.
	MVI M, B5H	:	Store B5H at location 2141H.
	CMP M	:	Compare contents of location 2140H with 2141H
	JNC MOON	:	Jump to MOON if no carry *i.e.* (A) is bigger.
	MOV A, M	:	Load A-register from 2141H.
MOON :	INX H	:	HL pair points at location 2142H
	MOV M, A	:	Store contents of accumulator at 2142H.
	HLT	:	Stop

Que 4.5. **Write a program to multiply two 8-bit numbers to result in 8-bit number only *i.e.*, maximum value could be FEH.**

Answer

Program :

Label	Mnemonics		Comments
	MVI B, 11H	:	Load B-register with data 11H.
	MVI C, 0FH	:	Load C-register with data 0FH.
	XRA A	:	Clear accumulator.
RENU :	ADD B	:	Add contents B-register to accumulator.
	DCR C	:	Decrement C-register.
	JNZ RENU	:	Jump to label RENU if not zero
	MOV D, A	:	Move contents of accumulator to D-register.
	HLT	:	Stop

Que 4.6. **Write an assembly language program to divide a 16-bit number by an 8-bit number.**

Answer

1. Assume 16-bit unsigned number is in memory location 3200H and 3201H (MSB in 3201H) and 8-bit unsigned number is in memory location 3300H.
2. The quotient and remainder are stored at location 3400H and 3401H.

Label	Mnemonics		Comments
	LHLD 3200H	:	Get dividend in HL pair
	LDA 3300H	:	Get divisor
	MOV B, A	:	Store divisor in register B
	MVI C, 08H	:	C-register as counter
CALCU :	DAD H	:	HL pair added to HL pair *i.e.*, shift HL 1 bit left
	MOV A, H		
	SUB B	:	Is MSB of dividend greater than divisor
	JC NEXT	:	Jump if carry to NEXT
	MOV H, A	:	Store divisor in H
	INR L	:	Increment L register
NEXT :	DCR C	:	Decrement counter
	JNZ CALCU	:	Jump if not zero to CALCU

Scanned by TapScanner

SHLD 3400H	:	Store at 3400H and 3401H
HLT	:	Stop

3. Register C contains the count = 8 as 8 times successive subtraction is done for an 8-bit divisor. It results into 8-bit quotient.
4. DAD H instruction shifts the contents of HL pair left by 1 bit leaving LSB of the HL pair vacant.
5. In the vacant position, the quotient is stored by incrementing the register L.
6. In this program, the MSB should be zero otherwise it will not work.

Que 4.7. **Draw the flow chart and write assembly language program for the addition of two 16-bit numbers considering carry. The numbers are stored in memory starting from 2000H. Store the result of addition and carry from memory 3000H.**

AKTU 2015-16, Marks 10

Answer

A. Flow chart :

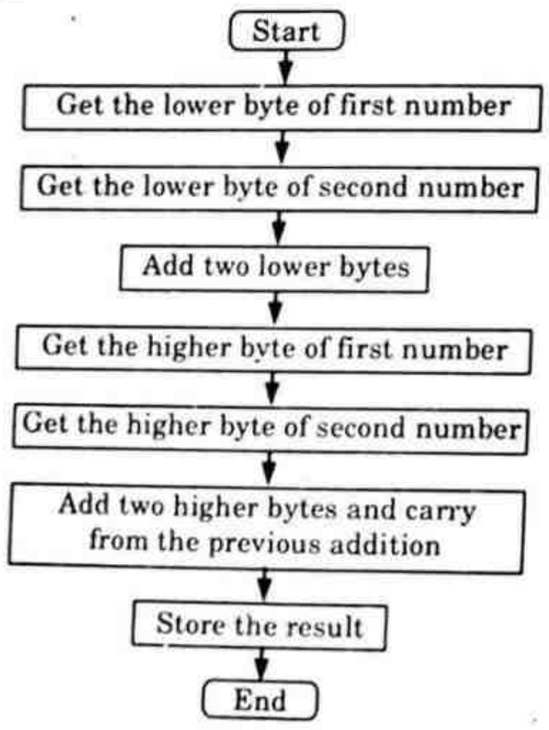


Fig. 4.7.1.

B. Program :

Mnemonics		Comment
LHLD 2000H	:	Get first 16-bit number in HL
XCHG	:	Save first 16-bit number in DE

LHLD 2002H	:	Get second 16-bit number in HL
MOV A, E	:	Get lower byte of the first number
ADD L	:	Add lower byte of the second number
MOV L, A	:	Store result in L register
MOV A, D	:	Get higher byte of the first number
ADC H	:	Add higher byte of the second number with carry
MOV H, A	:	Store result in H register
SHLD 3000H	:	Store 16-bit result in memory locations 3001H and 3002H
HLT	:	Terminate program execution

Que 4.8. **Write a program to arrange a data array in ascending order.**

Answer

Program :

Label	Mnemonics		Comments
	LXI D, 2601	:	Memory locations to store result.
	LXI H, 2500	:	Count address in HL pair
	MOV B, M	:	Count in register B to check whether all numbers have been arranged in ascending order.
START :	CALL 2200	:	Call Subroutine-1 to find smallest number.
	STAX D	:	Store the result.
	CALL 2050	:	Call Subroutine-2 to check which number is smallest.
	INX D		
	DCR B	:	Have all numbers been arranged in ascending order ?
	JNZ START	:	No, repeat process.
	HLT	:	Stop

Subroutine-1 : To find the smallest number

LXI H, 2500	:	Count address in HL pair.
MOV C, M	:	Count in register C.
MVI A, FF	:	Get FF in accumulator.

LOOP : INX H

Scanned by TapScanner

	MVI A, FF	:	Get FF in accumulator.
	CMP M	:	Compare next number with previous smallest. Is next number < Previous smallest ?
	JC AHEAD	:	No, smallest number is in the accumulator. Go to AHEAD.
	MOV A, M	:	Yes, get smaller number in accumulator.
AHEAD :	DCR C		
	JNZ LOOP		
	RET		

Subroutine-2 :

	LXI H, 2500 H		
	MOV C, M	:	Count to check which number was smallest.
BEHIND :	INX H	:	Get next number.
	CMP M	:	Compare the next number with smallest number which is in the accumulator.
	JZ FORWARD	:	Is the present number the smallest one? Yes, go to FORWARD.
	DCR C	:	Decrement count.
	JNZ BEHIND	:	No, jump to take up next number.
FORWARD :	MVI A, FF		
	MOV M, A	:	Replace the smallest number by FF.
	RET		

Que 4.9. **Write an assembly language program in 8085 to transfer the contents of flag register into *D* register.**

AKTU 2018-19, Marks 07

Answer

Program :

Mnemonics		Comments
LXI SP, 9000	:	Initialize stack pointer
MVI D,45H	:	Load B with 45H
MOV E, D	:	Save D into E
PUSH PSW	:	Store AF into stack
PUSH D	:	Store DE into stack

POP PSW	:	Pop and store to AF
POP D	:	Pop and store to DE
MOV D, E	:	Move E to D
HLT	:	Terminate the program

Que 4.10. **Write an assembly level program in 8085 to convert a binary number into its equivalent BCD form.**

AKTU 2018-19, Marks 07

Answer

Labels	Mnemonics		Comments
	LXI H, 8000H	:	Initialize memory pointer
	MVI D, 00H	:	Clear D-reg for most significant Byte
	XRA A	:	Clear accumulator
	MOV C, M	:	Get HEX data
LOOP :	ADI 01H	:	Count the number one by one
	DAA	:	Adjust for BCD count
	JNC SKIP	:	Jump to SKIP
	INR D	:	Increase D
SKIP :	DCR C	:	Decrease C register
	JNZ LOOP	:	Jump to LOOP
	MOV L, A	:	Load the least significant byte
	MOV H, D	:	Load the most significant byte
	SHLD 8050H	:	Store the BCD
	HLT	:	Terminate the program

Que 4.11. **An 8085 is executing the following program :**

```
2000:   LXI H, 4325H;
        LXI SP, 3000H;
        MOV A, H;
        ADD L;
        PUSH PSW;
        POP H;
        END
```

At the end of program execution, what will be the contents of the HL register pair ?

AKTU 2018-19, Marks 07

Scanned by TapScanner

Answer

Address	Mnemonics	Comments
2000	LXI H, 4325H	Store 4325H into the HL register pair H → 43H L → 25H
2003	LXI SP, 3000H	Initialize stack pointer with 16-bit data 3000H 2FFFH [ ] SP → 3000H [ ]
2006	MOV A, H	Move 8-bit data of H in Accumulator (A) in A → 43H
2007	ADD L	Add data of A with data of L and then store the result in A A 43H 0100 0011 L 25H 0010 0101 + 0110 1000 in A store → 68H
2008	PUSH PSW	Store the contents of PSW into two locations of stack. PSW is 16-bit register which is combination of A and flag register Flag register → S Z X AC X P X C Result is positive so SF = 0 Result is not zero so ZF = 0 No auxiliary carry is generated so AC = 0 There is odd parity so PF = 0 No carry is generated so CF = 0 Flag register → 00000000 ⇒ 00H 2FFEH 00 (flag register) 2FFFH 68 (data of A) 3000H X
2009H	POP H	Load register pair, H, L by popping out 2 bytes from top of the stack So, In L → 00H In H → 68H

Contents of register pair HL is 6800H

PART-2

Programming Techniques : Looping, Counting, Indexing, Counters and Time Delay.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.12. **Explain programming techniques.**

Answer

Programming Techniques : The program is an implementation of certain logic by executing group of instructions. To implement program logic we need to take help of some common programming techniques using 8085 such as

A. **Counting :** This technique allows programmer to count how many times the instruction/set of instructions are executed.

B. **Indexing :** This technique allows programmer to point or refer the data stored in sequential memory locations one by one.

C. **Looping :**

i. In this technique, the programming techniques using 8085 is instructed to execute certain set of instructions repeatedly to execute a particular task number of times.

ii. For example, to add ten numbers stored in the consecutive memory locations we have to perform addition ten times.

iii. The programming techniques using 8085 loop is the basic structure which forces the processor to repeat a sequence of instructions. Loops have four sections :

1. Initialization section.
2. Processing section.
3. Loop control section
4. Result section.

Scanned by TapScanner

D. Code Conversion :

i. This programming techniques in 8085 to translate a number represented using one coding system to another. For example, when we accept any number from the keyboard it is in ASCII code.

ii. But for processing, we have to convert this number in its hex equivalent. The code conversion involves some basic conversions such as

1. BCD to Binary conversion
2. Binary to BCD conversion
3. BCD to seven segment code conversion
4. Binary to ASCII conversion and
5. ASCII to binary conversion.

Que 4.13. **Develop an assembly language programme for 8086 to add two binary numbers each of 16 byte long. The numbers are stored from 9300 to 930F and 9400 to 940F. The result is to be stored from 9300 to 930F.** **AKTU 2014-15, Marks 10**

Answer

Label	Mnemonics		Comments
START :	MOV AX, DATA	:	Initialize data segment
	MOV DS, AX		
	MOV AX, OPP1	:	Take 1st operand in AX
	MOV BX, OPP2	:	Take 2nd operand in BX
	CLC	:	Clear previous carry if any
	ADD AX, BX	:	Add BX to AX
	MOV DI, OFFSET	:	Take offset of RESULT in DI
	MOV [DI], AX	:	Store the result at memory address in DI
	MOV AH, 4CH	:	Return to DOS prompt
	INT 21H		
CODE :	ENDS	:	CODE segment ends
	END START	:	Program ends

Que 4.14. **Write an assembly level program to find square root of given number.** **AKTU 2017-18, Marks 10**

Answer

Program :

Label	Mnemonics		Comments
	MVI D, 01	:	Initialize register D with 01
	MVI E, 01	:	Initialize register E with 01
	LDA 2050	:	Loads the content of memory location 2050 in accumulator A
2007 :	SUB D	:	Subtract value of D from A
	JZ 2011	:	Make jump to memory location 2011 if zero flag is set
	INR D	:	Increments value of register D by 1. Since it is used two times, therefore value of D is incremented by 2
	INR E	:	Increments value of register E by 1
	JMP 2007	:	Make jump to memory location 2007
2011 :	MOV A, E	:	Moves the value of register E in accumulator A
	STA 3050	:	Stores value of A in 3050
	HLT	:	Stops executing the program and halts any further execution

Que 4.15. **What is the purpose of a flow-chart ? Write an assembly level program in 8086 to implement the following flow-chart.**

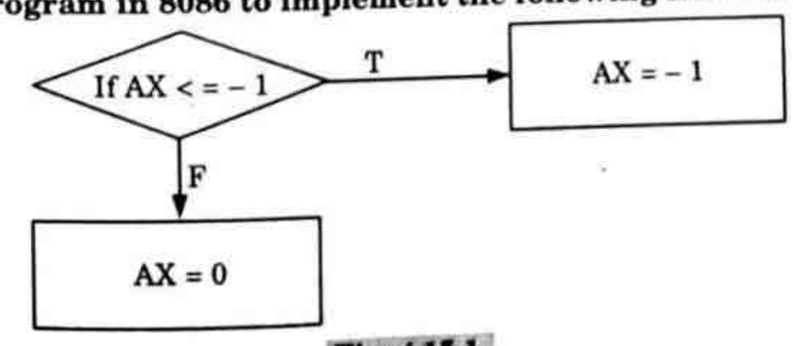


Fig. 4.15.1.

AKTU 2018-19, Marks 07

Scanned by TapScanner

Answer

i. Flow chart :

1. A flow chart is a diagrammatic representation of the procedure for solving the problem.
2. It is a pictorial representation that a programmer uses for planning the procedure for solution of problem.
3. It is used to indicate the direction of flow of a process, relevant operation and computation, point of decision and other information which is part of solution.

ii. Assembly level program :

```
Label   Mnemonics
        CMP AX, FFFFH
        JLE L1
        MOV AX, 0000H
        JMP L2
L1 :    MOV AX, FFFFH
L2 :    HLT
```

Que 4.16. **What are the methods to generate delay in software ?**

Answer

1. **Time delay using NOP instruction :** NOP instruction does nothing but takes 4T states of processor time to execute. So by executing NOP instruction in between two instructions we can get delay of 4 T-state

$$1\text{T state} = \frac{1}{\text{Operating frequncy of 8085}}$$

2. **Time delay using counters :** Counting can create time delays. Since the execution times of the instructions used in a counting routine are known, the initial value of the counter, required to get specific time delay can be determined.
3. **Time delay using nested loops :** In this, there is more than one loop. The innermost loop is same as in counters. The outer loop sets the multiplying count to the delays provided by the innermost loop.

Que 4.17. **Write an assembly language program to generate a delay of 1 msec. Also show the calculation of time delay. Assume that the crystal frequency of 8085 is 6 MHz.** **AKTU 2015-16, Marks 10**

Answer

A. Program :

```
        LXI B, Count
BACK :  DCX B
        MOV A, C
        ORA B
        JNZ BACK
```

B. Calculation of time delay :

1. Operating frequency $= \frac{6 \times 10^6}{2} = 3$ MHz
2. Time for one T-state $= \frac{1}{3 \times 10^6} = 0.33\ \mu s$
3. Required delay = 1 ms
4. Total number of T-state

$$= \frac{\text{Required delay}}{\text{Time for } T\text{-state}} = \frac{1 \times 10^{-3}}{0.33 \times 10^{-6}} = 3 \times 10^3$$

5. Now, calculating count

$$3 \times 10^3 = 10 + (\text{Count} - 1) \times 24 + 21$$

$$\frac{3 \times 10^3 - 31}{24} + 1 = \text{Count}$$

$$\text{Count} = 124.708 \approx (124)_{10} = 7\text{CH}$$

Que 4.18. **What are the methods to generate delay in software ? Write a programme to generate a delay of 200 ms using 8085 system that runs on 50 MHz frequency.** **AKTU 2014-15, Marks 10**

Scanned by TapScanner

Answer

A. **Methods to generate delay in software :** Refer Q. 4.16, Page 4–16C, Unit-4

B. **Program :** The procedure is same as Q. 4.17, Page 4–17C, Unit-4

C. **Calculation :** The procedure is same as Q. 4.17, Page 4–17C, Unit-4

(Ans. Count = 32 DCDR)

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. Why assembly language is used to program microprocessor ? What are the disadvantages of microprocessor ?

Ans. Refer Q. 4.1.

Q. 2. Develop an assembly language program for 8085 to add 5 numbers of 8 bits, whose sum is of more than 8 bits. The numbers are stored from 8501 to 8505. The result is to be stored in 8601 and 8602.

Ans. Refer Q. 4.2.

Q. 3. Develop a program to add two numbers of 8 byte long whose result is more than 8 byte. The first number is stored from 7501 to 7508 and second number is stored from 8501 to 8508 and result is to be stored from 9501 to 9509.

Ans. Refer Q. 4.3.

Q. 4. Write a program to obtain larger of the contents of memory location 2140H and 2141H.

Ans. Refer Q. 4.4.

Q. 5. An 8085 is executing the following program :

```
2000 :  LXI H, 4325H;
        LXI SP, 3000H;
        MOV A, H;
        ADD L;
        PUSH PSW;
        POP H;
        END
```

At the end of program execution, what will be the contents of the HL register pair ?

Ans. Refer Q. 4.11.

Q. 6. Write an assembly language program to generate a delay of 1 msec. Also show the calculation of time delay. Assume that the crystal frequency of 8085 is 6 MHz.

Ans. Refer Q. 4.17.

Q. 7. What are the methods to generate delay in software ? Write a programme to generate a delay of 200 ms using 8085 system that runs on 50 MHz frequency.

Ans. Refer Q. 4.18.

☺☺☺

Scanned by TapScanner

5 UNIT

Peripheral Devices

CONTENTS

Part-1 : 8237 : DMA Controller ..... **5–2C to 5–7C**

Part-2 : 8255 Programmable Peripheral Interface ..... **5–7C to 5–14C**

Part-3 : 8253/8254 : Programmable Timer/Control ..... **5–14C to 5–20C**

Part-4 : 8259 Programmable Interrupt Controller ..... **5–20C to 5–27C**

Part-5 : 8251 : USRAT ..... **5–27C to 5–31C**

Part-6 : RS-232C ..... **5–32C to 5–33C**

PART-1

8237 : DMA Controller.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.1. **Describe the pin configuration of 8237 DMA controller and give features also.**

Answer

A. Features :

1. Intel 8237, direct memory access (DMA) controller, enables data transfer between memory and the I/O with reduced load on the system's main processor by providing the memory with control signals and memory address information during the DMA transfer.
2. The 8237 is a four-channel device that can be expanded to include any number of DMA channel inputs.
3. The 8237 is capable of DMA transfers at rates of up to 1.6 megabyte per second.

B. Pin diagram : Pin diagram of 8237 DMA controller is shown in Fig. 5.1.1. The functional signal description in brief as follows :

1. **V_{cc} :** This is + 5 V supply pin required for operation of circuit.
2. **GND :** This is return line for the supply (ground pin of IC).
3. **CLK :** This is the internally required clock signal for deriving the internal timings required for the circuit operation.
4. **$\overline{CS}$:** This is an active-low chip select the input of IC.
5. **RESET :** A high on this input line clears the command, status, request and temporary register. It also clear internal first/last flip-flop and set the master register.
6. **READY :** This active high input is used to match the read or write speed of 8237 with slow memories or input/output devices.
7. **Address bus ($A_0 - A_3$ and $A_4 - A_7$) :** The four least significant lines $A_0 - A_3$ are bi-directional three state signal. In idle cycle, they are input and used by CPU to address the control register to be read or written. The four most significant address lines ($A_4 - A_7$) are activated only during DMA services to generate the respective address bits.

Scanned by TapScanner

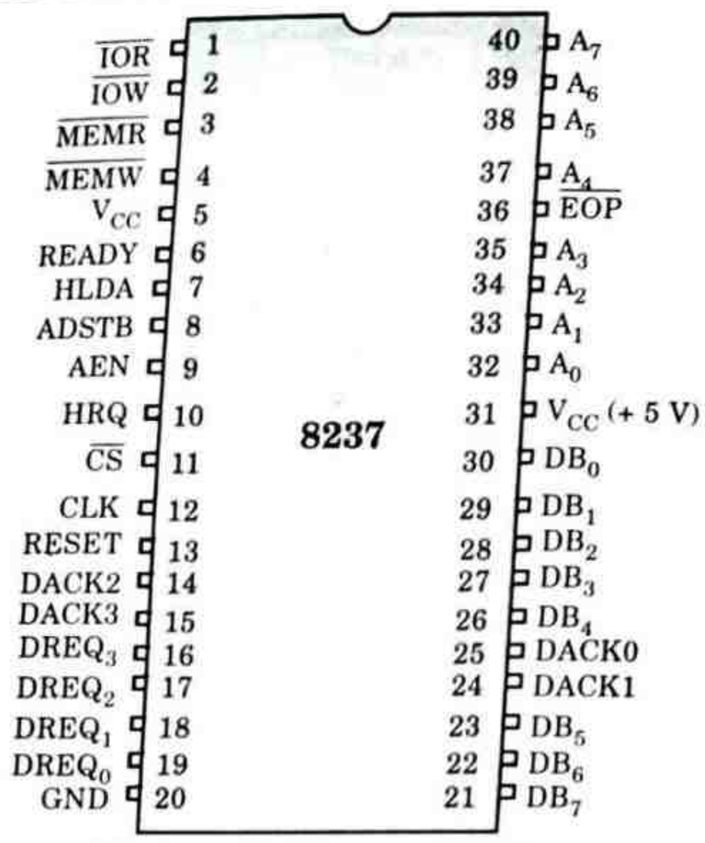


Fig. 5.1.1. Pin diagram of 8237.

8. **HRQ (Hold request) :** The HRQ is an output pin used to request the control of the system bus from the CPU. If the corresponding mask bit is not set, every valid DREQ to 8237 will issue HRQ signal to the CPU.
9. **$DB_0 - DB_7$ (Data bus) :** These are bi-directional lines used to transfer data to/from memory or input/output.
10. **$\overline{IOR}$ and $\overline{IOW}$ (I/O read and I/O write) :** These are active low bi-directional symbol. In idle cycle, these are input control signals used by CPU to read/write the control registers. In the active cycle $\overline{IOR}$ signal is used to access the data from a peripheral and $\overline{IOW}$ signal is used to load data to the peripheral.
11. **$DACK_0 - DACK_3$ (DMA acknowledge) :** These are used to indicate peripheral devices that the DMA request is granted.
12. **AEN (Address enable) :** This active-high output enables the 8-bit latch that drives the upper 8-bit address bus. The AEN pin is used to disable other bus drivers during DMA transfers.
13. **ADSTB (Address strobe) :** This output line is used to strobe the upper address byte generated by 8237, in master mode into an external latch.
14. **$DREQ_0 - DREQ_3$:** These are used to indicate peripheral devices that the DMA request is granted.
15. **$\overline{MEMR}$:** This active-low output is used to access data from the selected memory location, during DMA read or a memory to memory transfer.

16. **$\overline{MEMW}$:** This active low output signal is used to write data to the selected memory location during DMA write or memory to memory transfer.
17. **$\overline{EOP}$ (End of Process) :** This is an active low bi-directional (input or output) pin, used to indicate the completion of DMA operation.

Que 5.2. **What do you understand by DMA ? With the help of block diagram explain the working of 8237/8257.**

AKTU 2015-16, Marks 15

OR

Give the features and functional block diagram of 8237 DMA controller.

AKTU 2017-18, Marks 10

Answer

A. **DMA :**

1. To transfer large block of data at high speed, a special control unit may be provided to allow transfer of a block of data directly between an external device and the main memory, without continuous intervention by the processor. This approach is called direct memory access or DMA.
2. DMA transfers are performed by a control circuit associated with the I/O device and this circuit is referred as DMA controller.

B. **Features :** Refer Q. 5.1, Page 5–2C, Unit-5.

C. **Internal architecture of 8237 :**

i. The internal block diagram is shown in Fig. 5.2.1.

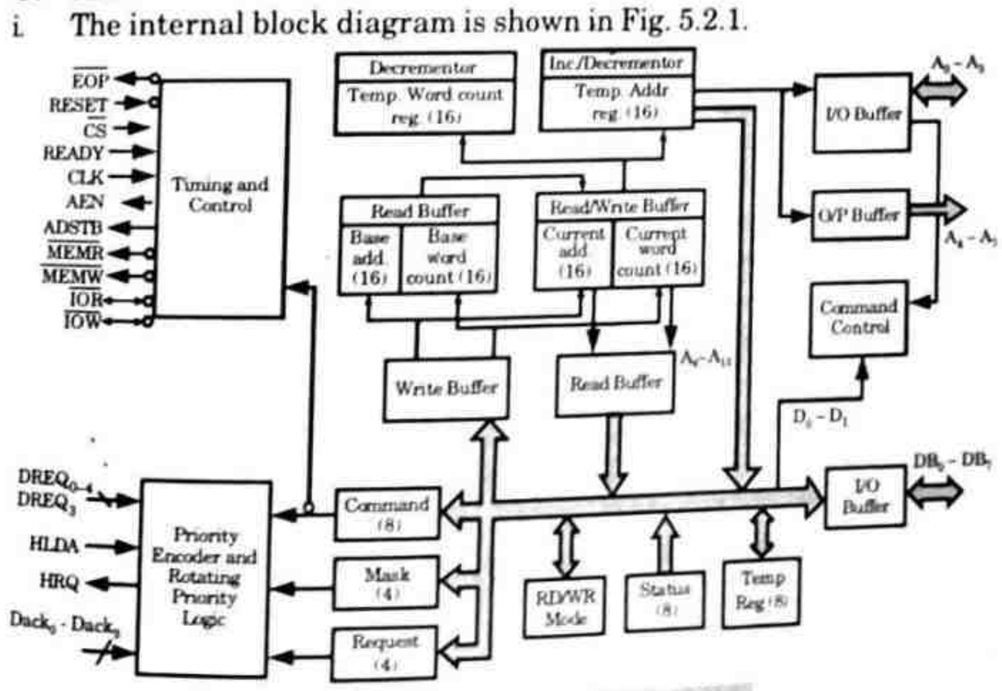


Fig. 5.2.1. Block diagram of 8237.

ii. 8237A contain three basic block of control logic :

Scanned by TapScanner

a. **Timing and control Block :** It generates internal timing and external control signal to the 8237A.

b. **Program command control Block :** It decodes various commands given to the 8237 by the microprocessor before servicing a DMA request. It also decodes the mode control word, which is used to select the type of DMA during the servicing.

c. **Priority encoder Block :** It resolves the priority between DMA channels requesting the service.

d. **Internal Registers :** 8237 contain 344 bits of internal memory in the form of register. Various types of register are :

1. **Current Address Register :** Each of 4 DMA channels of 8237 has a 10-bit current address register that hold the current memory address, being accessed during DMA transfer.
2. **Current Word Register :** Each channel has 16-bit current word register that holds the number of data byte transfers to be carried out.
3. **Base Address and Base Word Count Register :** Each channel has a pair of these registers. They maintain an original copy of respective initial current address register and current word register respectively.
4. **Command Register :** This 8-bit register controls the complete operation of 8237. This can be programmed operation.
5. **Mode Register :**

i. Each DMA channel has 8 bit mode registers. This is written by CPU in program mode. Bits 0 and 1 of mode register determine which of the 4 channel mode register is to be written.

ii. The bits 2 and 3 indicate type of DMA transfer. Bit 4 of mode register indicates whether auto initialization is selected or not.

iii. While bit 5 indicates whether address increment or decrement mode is selected.

6. **Request Register :**

i. Each channel has a request associated with it, in the request register.

ii. These are non-maskable and subject to prioritization by the priority resolving network of 8237.

7. **Mask Register :** Each of four channels of DMA has mask bit which can be set under program control to disable the incoming DREQ requests at the specific channel.
8. **Temporary register :** The temporary register holds data during memory to memory data transfers.
9. **Status Register :** This register keeps the track of all the DMA channel pending request and the status of their terminal counts.

Que 5.3. **Discuss internal block diagram of 8237 and explain the operating mode of 8237A.**

Answer

A. **Internal block diagram and organization :** Refer Q. 5.2, Page 5–4C, Unit-5.

B. **Operating mode of 8237A :**

i. **Single transfer mode :**

1. In this mode, the device transfer one byte per request.
2. The word count is decremented and the address is decremented or incremented after each transfer.
3. For each transfer the DREQ must be active until the DACK is activated, in order to get recognized.
4. After terminal count (TC), the bus will be relinquished for the CPU.

ii. **Block transfer mode :**

1. In this mode, 8237 is activated by DREQ to continue the transfer until a TC is reached, *i.e.*, a block of data is transferred.
2. The transfer cycle may be terminated due to $\overline{EOP}$ which forces TC.
3. The DREQ needs to be activated only till the DACK signal is activated by DMA controller.

iii. **Demand transfer mode :**

1. In this mode, the device continuously transfer until a TC is reached or an external $\overline{EOP}$ is detected or the DREQ signal goes inactive.
2. Thus, a transfer may exhaust the capacity of data transfer of an input/output device.
3. After the input/output device able to catch up, the service may be re-established by activating the DREQ signal again.

iv. **Cascade mode :**

1. In this mode, more than one 8237 can be connected together to provide more than four DMA channels.
2. The HRQ and HLDA signals from additional 8237s are connected with DREQ and DACK pins of channel of the host 8237 respectively.
3. The priorities of the DMA requests may be preserved at each level.

v. **Memory to memory transfer :**

1. In this mode, block of data from one memory address is moved to another memory address.
2. In this mode, current register of channel 0 is used to point the source address and current register of channel 1 is used to point destination address in first transfer cycle, data byte from source address is loaded in the temporary register of 8237 and in the next transfer cycle the data from the temporary register is stored in memory pointed by destination address.
3. After each data transfer current address register are decremented or incremented according to current setting.
4. When the word count of channel 1 goes to FFFFH, a TC is generated which activates $\overline{EOP}$ output terminating the DMA service.

Scanned by TapScanner

Que 5.4. **What is meant by direct memory access ? With the help of neat schematic, explain the interaction between 8086 MPU and DMA controller for providing direct memory access to a peripheral device.**

AKTU 2018-19, Marks 07

Answer

i. **DMA :** Refer Q. 5.2, Page 5–4C, Unit-5.

ii. **For 8086 in maximum mode :** The RQ/GT_1 and RQ/GT_1 pins are utilized to issue DMA signals of request and receive acknowledge.

Sequence of events of a usual DMA process :

1. Peripheral asserts one of the request pins, for example : RQ/GT_1 or RQ/GT_0 (has higher priority).
2. 8086 finishes its current bus cycle and enters in a HOLD state.
3. 8086 grants the right of bus control through asserting a grant signal via similar pin as the request signal.
4. DMA operation starts.
5. Upon end of the DMA operation, there peripheral asserts the request/grant pin again to relinquish bus control.

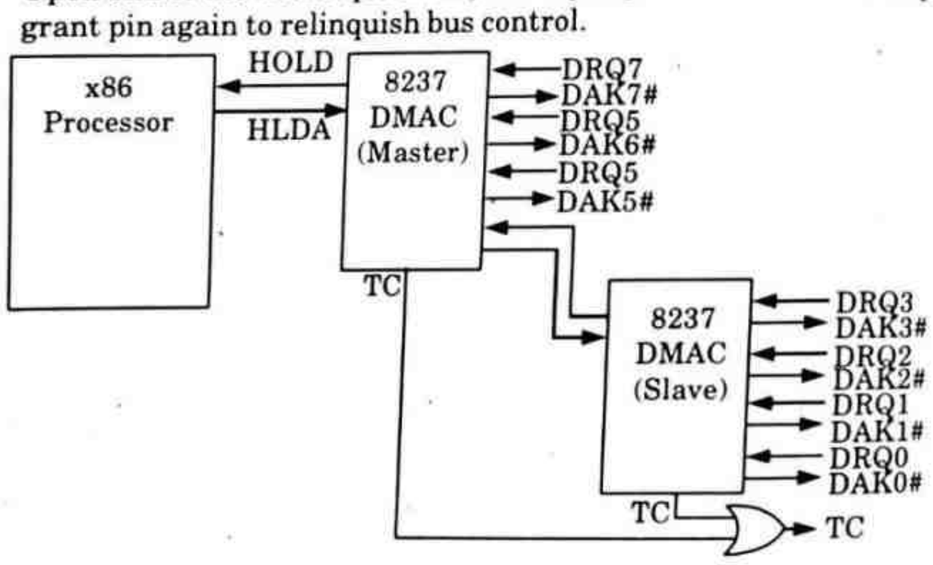


Fig. 5.4.1. Interfacing 8257/37 with 8086.

PART-2

8255 : Programmable Peripheral Interface.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.5. **What are the features of 8255 PPI ?**

OR

With a neat diagram discuss internal architecture of 8255.

AKTU 2015-16, Marks 05

OR

Explain the internal architecture of 8255.

AKTU 2017-18, Marks 10

Answer

1. The 8255 is a widely used, programmable, parallel I/O device.
2. It can be programmed to transfer data under various conditions, from simple I/O to interrupt I/O.
3. It is compatible with all Intel and most other microprocessors.
4. It is completely TTL compatible.
5. It has three 8-bit ports : Port A, Port B, and Port C, which are arranged in two groups of 12 pins.
6. Each port has a unique address, and can be read from or written to a part.
7. The address is assigned to the control register into which control words are written for programming the 8255 to operate in various modes.

A. Internal architecture of 8255 :

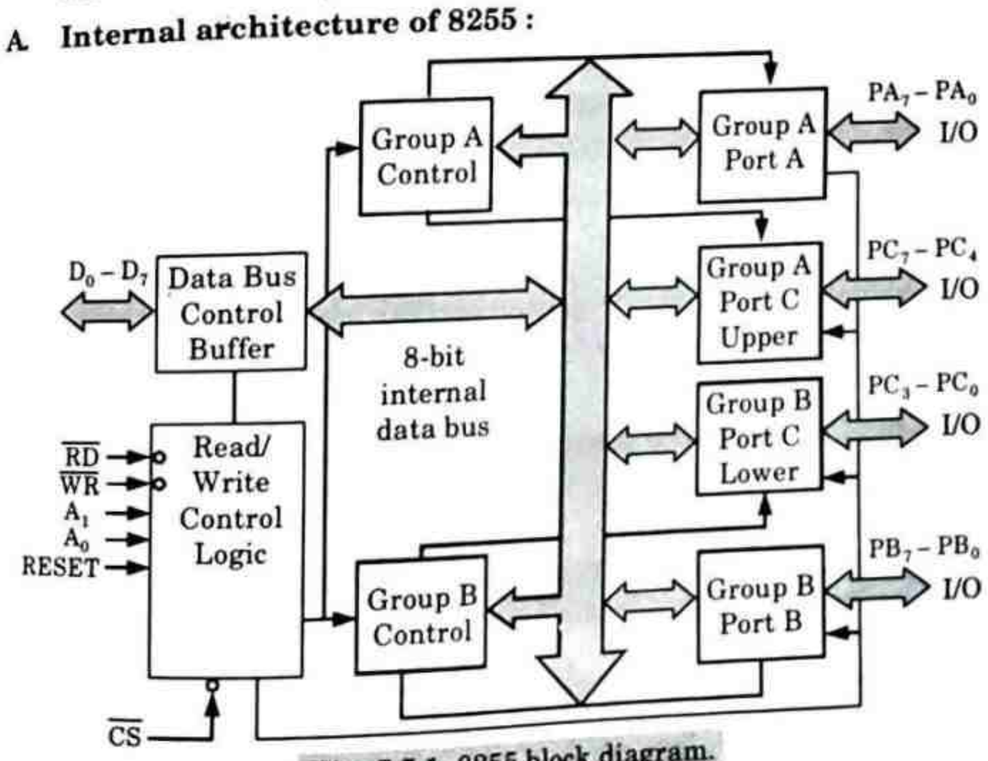


Fig. 5.5.1. 8255 block diagram.

Scanned by TapScanner

B. Functional blocks :

a. Data bus control buffer :

1. This tri-stated bi-directional buffer is used to interface the 8255 to the system data bus.
2. IN or OUT instructions executed by the CPU either read data from, or write data into the buffer.
3. Output data from the CPU to the ports or control register, and input data to the CPU from the ports or status register are all passed through the buffer.

b. Read/Write control logic :

1. The control logic block accepts control bus signals as well as inputs from the address bus, and issues commands to the individual group control blocks.
2. It issues appropriate enabling signals to access the required data/control words or status word. $\overline{RD}$, $\overline{WR}$, RESET, and $\overline{CS}$ are 4-control signals generated by control section.

c. Group A and Group B controls :

1. Each of the group A and group B control blocks receives control words from the CPU through the data bus buffer and the internal data bus accepts commands from the control logic block and issues appropriate commands to the ports associated with it.
2. The group A control block controls Port A and $PC_7 - PC_4$, while the group B control block controls Port B and $PC_3 - PC_0$.

d. Ports :

1. **Port A :** This has an 8-bit latched and buffered output and 8-bit input latch.
2. **Port B :** This has an 8-bit input/output latch/buffer and an 8-bit data input buffer.
3. **Port C :** This has one 8-bit unlatched input buffer and an 8-bit output latch/buffer. Port C can be splitted into two parts and each can be used for control signal outputs/inputs for Port A and Port B in the handshake mode.

Que 5.6. **Describe the pin configuration 8255.**

Answer

1. $PA_7 - PA_0$: These are eight port A lines that act as either latched output or buffered input lines depending upon the control word loaded into the control word register.
2. $PC_7 - PC_4$: Upper nibble of port C lines. They may act as either output latches or input buffer lines. This port can also be used for generation of handshake lines in mode 1 or mode 2.
3. $PC_3 - PC_0$: These are the lower port C lines.

4. $PB_0 - PB_7$: These are the eight port B lines which are used as latched output lines or buffered input lines in the same way as port A.
5. $\overline{RD}$: This is the input line driven by the microprocessor and should be low to indicate read operation to 8255.
6. $\overline{WR}$: This is an input line driven by the microprocessor. A low on this line indicates write operation.
7. $\overline{CS}$: This is a chip select line. If this line goes low, it enables the 8255 to respond $\overline{RD}$ and $\overline{WR}$ signals, otherwise $\overline{RD}$ and $\overline{WR}$ signals are neglected.
8. $A_1 - A_0$: These are the address input lines and are driven by microprocessor.
9. $D_0 - D_7$: These are the data bus lines those carry data or control word to/from the microprocessor.
10. **RESET :** A logic high on this line clears the control word register of 8255. All ports are set as input ports by default after reset.

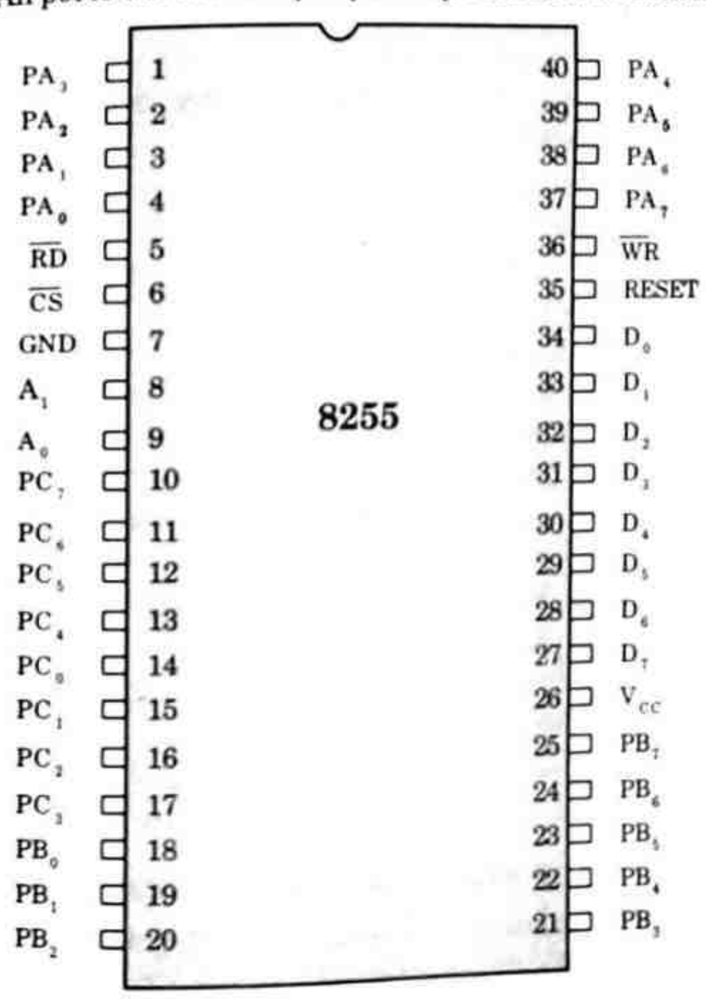


Fig. 5.6.1. 8255 pin configuration.

Scanned by TapScanner

Que 5.7. **Draw and explain the internal architecture of 8255. What are its modes of operation ?** **AKTU 2014-15, Marks 10**

Answer

A. Architecture : Refer Q. 5.5, Page 5–8C, Unit-5.

B. The 8255 has following modes of operation :

i. Mode 0 (Basic input/output) :

1. In this mode, in addition to Port A and Port B, $PC_7 - PC_4$ and $PC_3 - PC_0$ of Port C can be considered as two individual 4-bit ports.
2. Therefore, four ports, each of which can be configured either as an input or an output port are available.
3. Here the ports are simple input or output ports, data is written to or read from the specified port without handshaking.
4. The data send out to the output ports are latched, whereas inputs are not latched.

ii. Mode 1 (Strobe input/output) :

1. In this mode, input or output data transfer is affected by strobe (handshaking signals).
2. The two groups, Group A and Group B, can be configured separately, with each group consisting of an 8-bit port and a 4-bit port.
3. The 8-bit port can be programmed for input or output. The 4-bit port is used for handshaking.

iii. Mode 2 (Strobe bi-directional bus I/O) :

1. This mode allows bi-directional data transfer over a single 8-bit data bus using handshaking signals.
2. This feature is available only in Group A with port A as the 8-bit bi-directional data bus, and $PC_4 - PC_7$ are used for handshaking purposes.
3. In this mode, both input and output are latched.

iv. Bit Set-Reset (BSR) Mode :

1. In this mode, any of the 8-bits of port C can be set or reset by sending out a OUT instruction to the control register.
2. When port C is used for control/status operation, this feature can be used to set or reset individual bits as if they were output ports.

Que 5.8. **Draw and explain the internal architecture of 8255 parallel I/O peripheral device. Also describe the bits of control word.**

AKTU 2016-17, Marks 10

Answer

A. Internal architecture 8255 : Refer Q. 5.5, Page 5–8C, Unit-5.

B. Bits of control word :

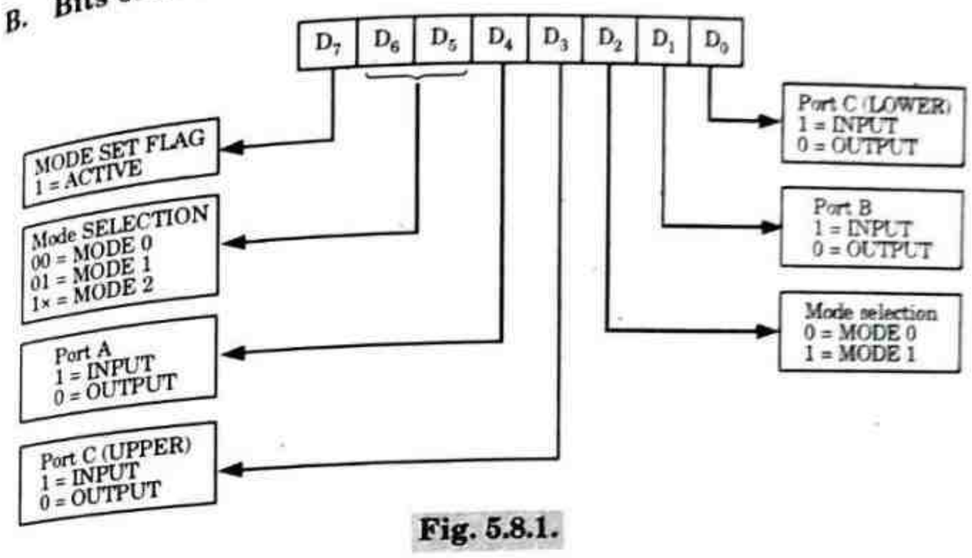


Fig. 5.8.1.

Que 5.9. **Form a control word to set the D_3 bit of port-C of the 8255. Write a program that drives 8 numbers of LEDs sequentially using any one of the ports of PPI.** **AKTU 2018-19, Marks 07**

Answer

i. Control word :

1. According to the requirement a port can be programmed to act either as an input port or an output port. For programming the ports of 8255 a control word is formed. The bits of the control word are as shown in Fig. 5.9.1.
2. Control word is written into the control word register which is within 8255. No read operation of the control word register is allowed.
3. The control word bit corresponding to a particular port is set to either 1 or 0 depending upon the definition of the port, whether it is to be made an input port or output port.
4. If a particular port is to be made an input port, the bit corresponding to that port is set to 1. For making a port an output port, the corresponding bit for the port is set to 0.

Scanned by TapScanner

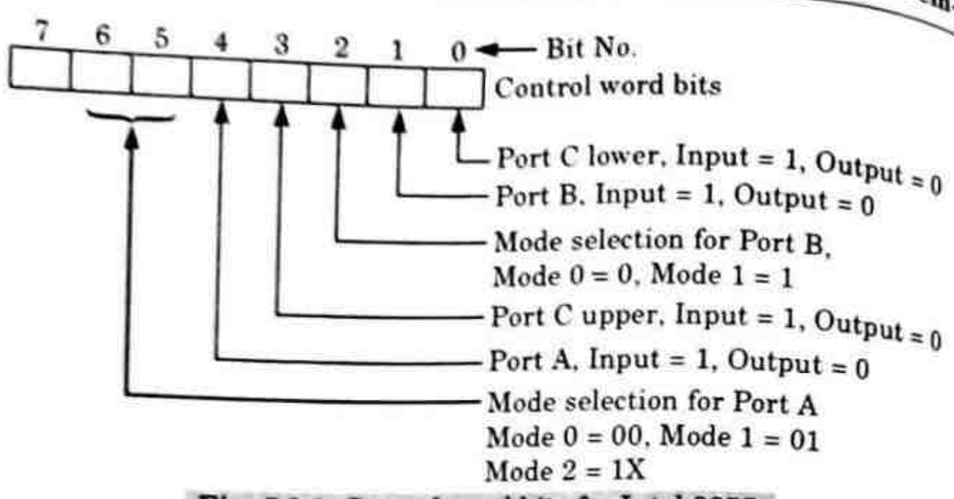


Fig. 5.9.1. Control word bits for Intel 8255.

5. **Bit No. 3 :** It is for the port C_{upper}.
   To make port C_{upper} an input port, the bit is set to 1.
   To make port C_{upper} an output port, the bit is set to 0.

ii. **Program :**

```
        MVI A,80H     ; Load accumulator with control word 80H.
        OUT 03H       ; Shift ((A)) to CWR whose address is 03H.
        MVI A,01H     ; Load accumulator with 01H to make its
                        0th bit high.
MOON :  OUT 02H       ; Shift ((A)) to Port-C.
        CALL DELAY    ; Call delay sub-routine.
        RLC           ; Move 1 from right to left starting from
                        0th bit.
        JMP MOON      ; Jump to label MOON to repeat.
```

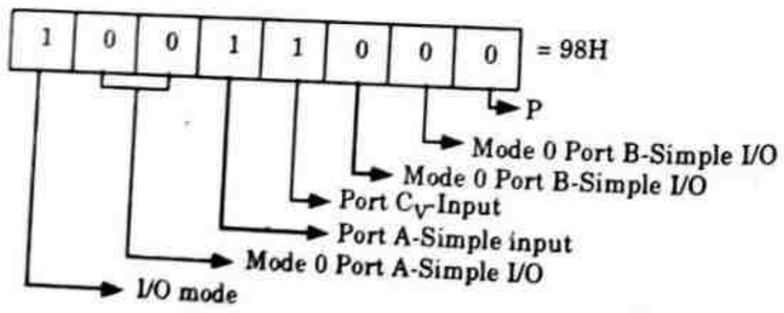
Que 5.10. **Write a program to initialize 8255 as follows :**

Port A : Simple input port
Port B : Simple output port
Port C_L : Output port
Port C_v : Input port

Assume the address of control register is 03H.

AKTU 2015-16, Marks 05

Answer



Program :

```
MVI A, 98H
OUT 03H
```

PART-3

8253/8254 Programmable Timer/Control.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.11. **What is 8253/8254 programmable interval timer, draw and explain its internal architecture.** **AKTU 2015-16, Marks 7.5**

Answer

A. **Features :**

1. Delay routines cannot give time delay precisely. 8253 facilitates the generation of accurate time delays.
2. Also when 8253 is used as a timing and delay generation peripheral, the microprocessor becomes free from the tasks related to counting and can execute the programs in memory. This minimizes the software overhead on the microprocessor.
3. The 8254 is an upgraded version of the 8253, and they are pin compatible. Features of the two devices are almost identical except that

i. The 8254 can operate on higher frequency than the 8253.

ii. The 8254 includes a status read-back command that can latch the count and the status of the counter.

B. **Architecture and signal descriptions :**

1. The internal block diagram of 8253 is shown in Fig. 5.11.1. The programmable timer device 8253 contains three independent 16-bit counters. It is thus possible to generate three independent delays or three independent counters simultaneously.
2. The three counters are controlled by programming the three internal command word registers.
3. The 8-bit, bi-directional data buffer interfaces internal circuit of 8253 to microprocessor systems bus.
4. Data is transmitted or received by the buffer upon the execution of IN or OUT instruction. The IN instruction reads data while OUT instruction writes data to a peripheral.

Scanned by TapScanner

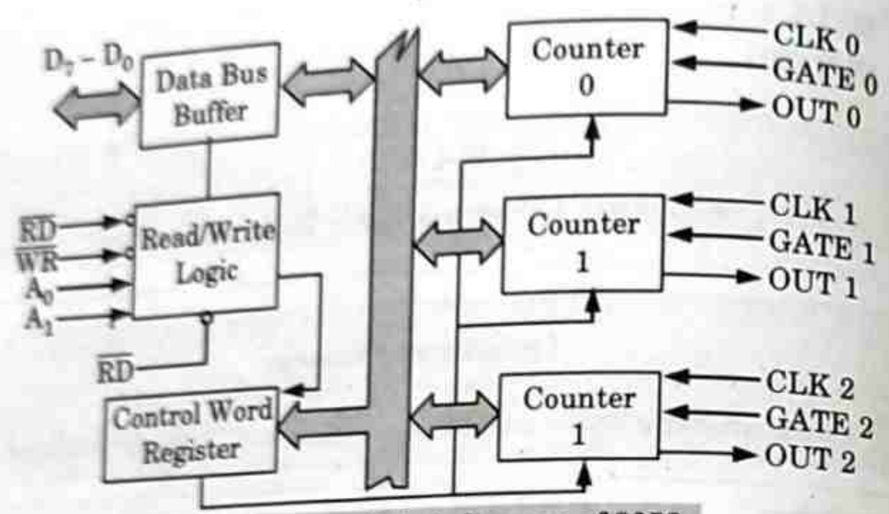


Fig. 5.11.1. Internal block diagram of 8253.

5. The three counters available in 8253 are independent of each other in operation, but they are identical to each other in organization.
6. The specialty of the 8253 counters is that they can be easily read on line without disturbing the clock input to the counter.
7. A_0, A_1 pins are address input pins and are required internally for addressing the control word registers and the three counter registers.
8. A low on $\overline{CS}$ line enables the 8253. No operation will be performed by 8253 till it is enabled.

B. Various operations for control inputs of 8253 :

Table 5.11.1.

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	Selected Operation
0	1	0	0	0	Write Counter 0
0	1	0	0	1	Write Counter 1
0	1	0	1	0	Write Counter 2
0	1	0	1	1	Write Control Word
0	0	1	0	0	Read Counter 0
0	0	1	0	1	Read Counter 1
0	0	1	1	0	Read Counter 2
0	0	1	1	1	No Operation (tri-stated)
0	1	1	×	×	No Operation (tri-stated)
1	×	×	×	×	Disabled (tri-stated)

A control word register accepts the 8-bit control word written by the microprocessor and stores it for controlling the complete operation of the specific counter.

Que 5.12. Draw and explain block diagram and pin configuration of IC-8253.

AKTU 2017-18, Marks 10

Answer

A. Block diagram of 8253/8254 : Refer Q. 5.11, Page 5–14C, Unit-5.

B. Pin configuration of IC 8253 :

1. **V_{cc} and GND :** Power supply and ground pins. 8253 uses + 5 V power supply.
2. **D_{7-0}:** Eight bi-directional data pins for communication with the processor.
3. **$\overline{RD}$:** Active low input pin that is activated by the processor to read counter information from the 8253.
4. **$\overline{WR}$:** Active low input pin that is activated by the processor to write control information to the 8253.
5. **$\overline{CS}$:** Active low input pin used for selecting the chip.
6. **A_1, A_0:** Address input pins. They are used along with $\overline{RD}$, $\overline{WR}$, and $\overline{CS}$ to select one of the counters or the control port. The control port can be written only from the processor. The processor cannot read it. The counters can be read or written.

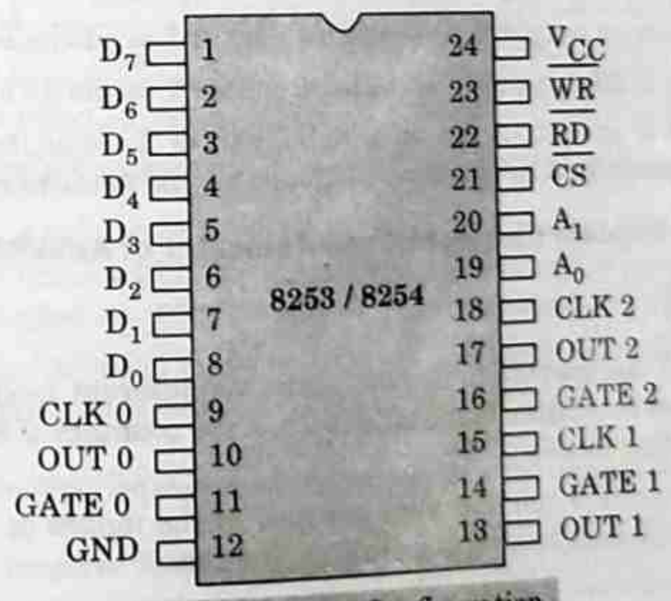


Fig. 5.12.1. Pin Configuration.

Scanned by TapScanner

Table 5.12.1. Selection of a counter port or control port

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Operation
0	0	0	1	0	Read Counter 0
0	1	0	1	0	Read Counter 1
1	0	0	1	0	Read Counter 2
1	1	0	1	0	No operation
0	0	1	0	0	Write to Counter 0
0	1	1	0	0	Write to Counter 1
1	0	1	0	0	Write to Counter 2
1	1	1	0	0	Write to control port
X	X	X	X	1	No operation
X	X	1	1	0	No operation

7. **CLK_0:** Provides clock input for Counter 0.
8. **CLK_1:** Provides clock input for Counter 1.
9. **CLK_2:** Provides clock input for Counter 2. Maximum frequency allowed for any of these clocks is 2 MHz.
10. **$Gate_0$:** It is an input pin that controls the function of Counter 0.
11. **$Gate_1$:** It is an input pin that controls the function of Counter 1.
12. **$Gate_2$:** It is an input pin that controls the function of Counter 2. The gate input has different effects in different modes on the functioning of the corresponding counters.
13. **Out_0:** It is an output pin on which Counter 1 sends its output.
14. **Out_1:** It is an output pin on which Counter 1 sends its output.
15. **Out_2:** It is an output pin on which Counter 2 sends its output. The output generated by the timer depends on the mode of operation.

Que 5.13. **Explain the control word register of 8253/8254.**

Answer

1. The 8253 can operate in anyone of the six different modes. A control word must be written to initialize each of the counters of 8253 to decide its operating mode.
2. All the counters can operate in anyone of the modes or they may be even in different modes of operation, at a time.
3. The control word format is shown Fig. 5.13.1, along with the definition of each bit :

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC_1	SC_0	RW_1	RW_0	M_2	M_1	M_0	BCD

Control word format

SC_1	SC_0	Operation
0	0	Select counter 0
0	1	Select counter 1
1	0	Select counter 2
1	1	Read-back command

SC-Select counter bit definitions

RW_1	RW_0	Operation
0	0	Latch counter for "ON THE FLY" reading
0	1	Read/Write least significant Byte only
1	0	Read/Write MSB only
1	1	Read/Write LSB first then MSB

RW-Read/Write bit definitions

M_2	M_1	M_0	Selected Mode
0	0	0	Mode 0
0	0	1	Mode 1
×	1	0	Mode 2
×	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

$M_2M_1M_0$ – Mode select bit definitions

BCD	Operation
0	Hexadecimal Count
1	BCD Count

HEX/BCD bit definition

Fig. 5.13.1. Control word format and bit definitions.

4. While writing a count in the counter, it should be noted that, the count is written in the counter only after the data is put on the data bus and a falling edge appears at the clock pin of the peripheral thereafter.
5. Any reading operation of the counter, before the falling edge appears may result is garbage data.

Que 5.14. **Discuss the mode of operation of 8253 program, internal times with its control format.** **AKTU 2016-17, Marks 10**

OR

Explain how 8253/8254 can be used as a square wave generator. **AKTU 2015-16, Marks 7.5**

Scanned by TapScanner

Answer

1. **Mode 0 (Interrupt on terminal count) :**

a. In this mode, initially the OUT is low.

b. Once a count is loaded in the register, the counter is decremented every cycle and when the count reaches zero, the OUT goes high.

c. This can be used as an interrupt. The OUT remains high until a new count or a command word is loaded.

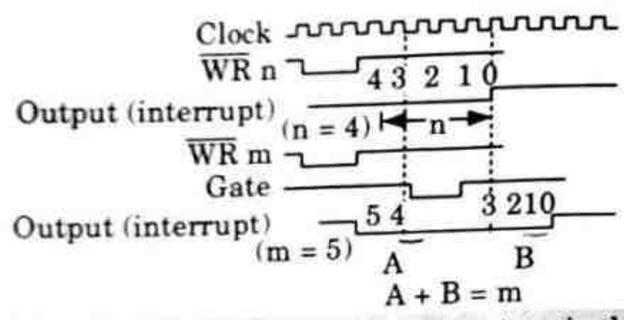


Fig. 5.14.1. Mode 0 : Interrupt on terminal count.

2. **Mode 1 (Hardware-retriggerable one-shot) :**

a. In this mode, the OUT is initially high.

b. When the Gate is triggered, the OUT goes low, and at the end of the count, the OUT goes high again, thus generating a one-shot pulse.

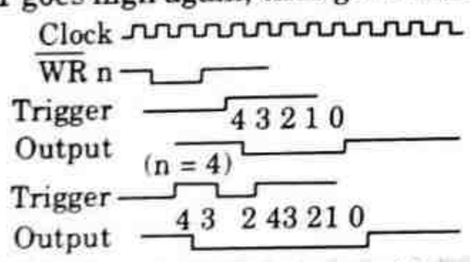


Fig. 5.14.2. Mode 1 : Programmable one-shot.

3. **Mode 2 (Rate generator) :**

a. This mode is used to generate a pulse equal to the clock period at a given interval.

b. When a count is loaded, the OUT stays high until the count reaches 1, and then the OUT goes low for one clock period.

c. The count is reloaded automatically and the pulse is generated continuously.

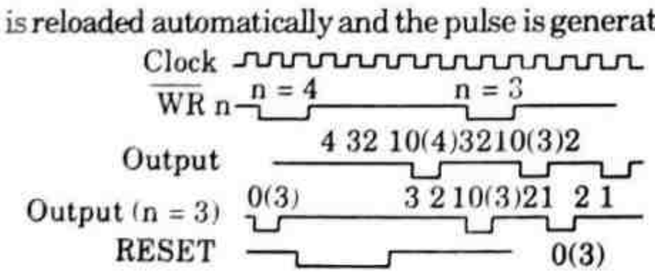


Fig. 5.14.3. Mode 2 : Generator clock.

4. **Mode 3 (Square wave generator) :**

a. In this mode, when a count is loaded, the OUT is high.

b. The count is decremented by two at every clock cycle and when it reaches zero, the OUT goes low and the count is reloaded again.

c. This is repeated continuously; thus a continuous square wave with period equal to the period of the count is generated.

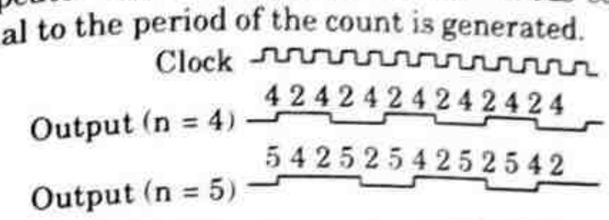


Fig. 5.14.4. Square wave generator.

5. **Mode 4 (Software-triggered strobe) :**

a. In this mode, the OUT is initially high; it goes low for one clock period at the end of the count.

b. The count must be reloaded for subsequent outputs.

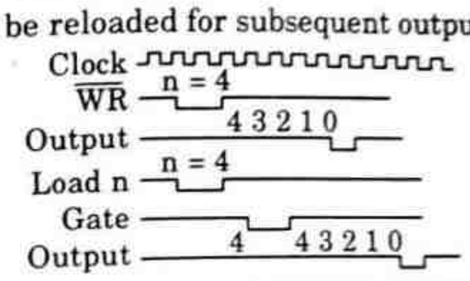


Fig. 5.14.5. Software triggered strobe.

6. **Mode 5 (Hardware-triggered strobe) :**

a. This mode is similar to Mode 4, except that it is triggered by the rising pulse at the gate.

b. Initially, the OUT is low, and when the Gate pulse is triggered from low to high the count begins.

c. At the end of the count, the OUT goes low for one clock period.

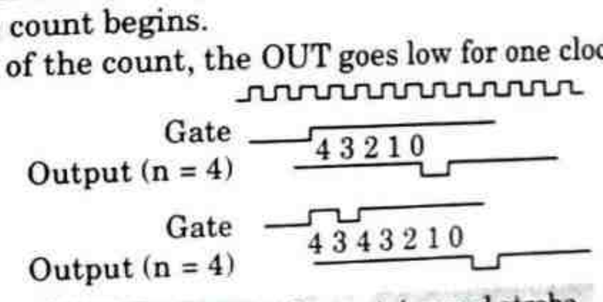


Fig. 5.14.6. Hardware triggered strobe.

PART-4

8259 : Programmable Interrupt Controller.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Scanned by TapScanner

Que 5.15. **Explain 8259 programmable interrupt controller.**

AKTU 2017-18, Marks 05

Answer

1. The 8259A is packed in 28-pin plastic DIP using nMOS technology and requires only one power supply of +5 V.
2. It does not require any clock input.
3. It provides the facility of 8-vectored priority interrupts for the μP on a single chip.
4. The 8259 A has several modes of operation.
5. The Intel's 8259 A is a Programmable Priority Interrupt Controller (PIC).
6. It accepts interrupt requests from 8-I/O devices, determines its priority, and priority, and issues interrupt to the μP.
7. After receiving the acknowledgement from the μP, the 8259 places CALL instruction along with the associated address of the current interrupt on the data bus.
8. Without any additional circuitry nine numbers of the 8259A can be cascaded to handle up to 64-vectored priority interrupts.
9. It accept requests from the peripheral devices, determines the highest priority (importance) of the device.

Que 5.16. **Explain pin diagram of 8259 A (PIC).**

Answer

1. $\overline{WR}$: This pin is an active low write enable input to 8259A. This enables it to accept command words from CPU.
2. $\overline{RD}$: This is an active-low read enable input to 8259A. A low on this line enables 8259A to release status onto the data bus of CPU.
3. D_7-D_0 : These pins form a bidirectional data bus that carries 8-bit data either to control word or from status word registers. It also carries interrupt vector information.
4. **INT :** This pin goes high whenever a valid interrupt request is asserted. This is used to interrupt the CPU and is connected to the interrupt input of CPU.
5. IR_0-IR_7 **(Interrupt requests) :** These pins act as inputs to accept interrupt requests to the CPU. In the edge triggered mode, an interrupt service is requested by raising an IR pin from a low to a high state. It is held high until it is acknowledged, and just by latching it to high level, if used in the level triggered mode.
6. $\overline{INTA}$ **(Interrupt acknowledge) :** This pin is an input used to strobe-in 8259A interrupt vector data on to the data bus. In conjunction with

CS, WR, and RD pins, this selects the different operations like, writing command words, reading status word, etc.

7. CAS_2–CAS_0 **(I/O Cascade Lines) :** The CAS lines form a private 8259A bus to control a multiple 8259A structure. These pins are outputs for a master 82C59A and inputs for a slave 82C59A.
8. $\overline{SP}$ / $\overline{EN}$ **(I/O Slave Program/Enable Buffer) :** This is a dual function pin. In the buffered mode, it can be used as an output to control buffer transceivers ($\overline{EN}$). When not in the buffered mode, it is used as an input to designate a master ($\overline{SP}$ = 1) or slave ($\overline{SP}$ = 0). A HIGH on the ($\overline{SP}$ = $\overline{EN}$ = 1) pin designates the 8259 as the master, a LOW on ($\overline{SP}$ = $\overline{EN}$ = 0) designates it as slave.

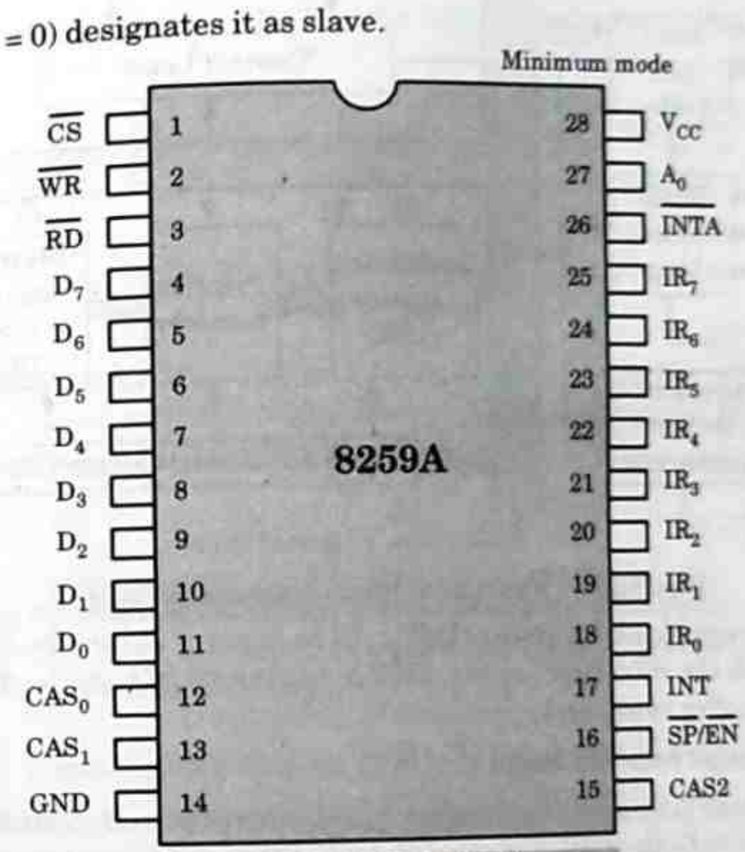


Fig. 5.16.1. 8259A pin diagram.

Que 5.17. **Explain the architecture of 8259A.**

Answer

The architectural block diagram of 8259A is shown in Fig. 5.17.1. The functional explanation of each block is given as follows :

i. **Interrupt request register (IRR) :** The interrupts at IRQ input lines are handled by interrupt request register internally. IRR stores all the interrupt requests in it in order to serve them one by one on the priority basis.

Scanned by TapScanner

ii. **In-service register (ISR) :** It stores all the interrupt requests those are being served, *i.e.*, ISR keeps a track of the request being served.

iii. **Priority resolver :**

1. This unit determines the priorities of the interrupt request appearing simultaneously.
2. The highest priority is selected and stored into the corresponding bit of ISR during $\overline{INTA}$ pulse.
3. The IR_0 has the highest priority while the IR_7 has the lowest one, normally in fixed priority mode.

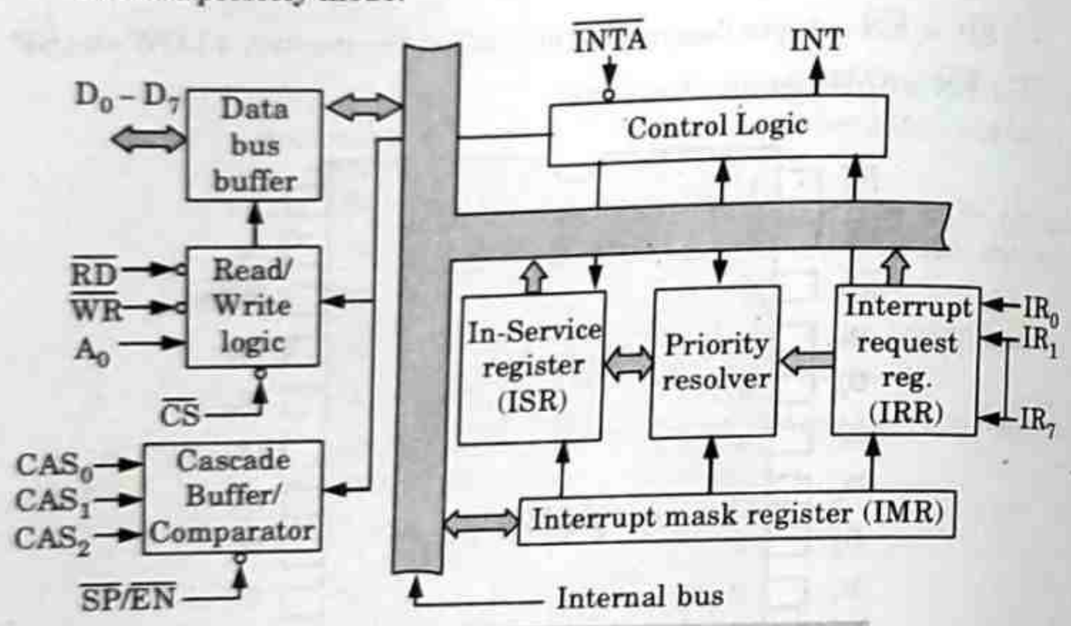


Fig. 5.17.1. Functional block diagram of 8259A.

iv. **Interrupt Mask Register (IMR) :** This register stores the bit required to mask the interrupt inputs. IMR operates on IRR at the direction of the priority resolver.

v. **Interrupt control logic :**

1. This block manages the interrupt and interrupt acknowledge signals to be sent to the CPU for serving one of the eight interrupt requests.
2. This also accepts the interrupt acknowledge (INTA) signal from CPU that causes the 8259A to release vector address on the data bus.

vi. **Data bus buffer :**

1. This tri-state bi-directional buffer interfaces internal 8259A bus to the microprocessor system data bus.
2. Control words, status and vector information pass through data buffer during read or write operations.

vii. **Read/write control logic :** This circuit accepts and decodes commands from the CPU. This block also allows the status of the 8259A to be transferred on the data bus.

viii. **Cascade buffer/comparator :** This block stores and compares the ID's of all the 8259A's used in the system.

Que 5.18. Explain different modes of operation of 8259 A.

AKTU 2017-18, Marks 10

Answer

i. **Fully nested mode :**

1. This is a general-purpose mode in which all IRs (interrupt requests) are arranged from highest to lowest, with IR_0 as the highest and IR_7 as the lowest.
2. In the example below, IR_4 has the highest priority and IR_3 has the lowest priority :

IR_0	IR_1	IR_2	IR_3	IR_4	IR_5	IR_6	IR_7
4	5	6	7	0	1	2	3
			↑ Lowest priority	↑ Highest priority			

ii. **Automatic rotation mode :** In this mode, a device, after being serviced, receives the lowest priority. Assuming that the IR_2 has just been serviced, it will receive the seventh priority, as shown below :

IR_0	IR_1	IR_2	IR_3	IR_4	IR_5	IR_6	IR_7
1	6	7	0	1	2	3	4

iii. **Specific rotation mode :** This mode is similar to the automatic rotation mode, except that the user can select any IR for the lowest priority, thus fixing all other priorities.

Que 5.19. Draw and explain the internal architecture of 8259 interrupt controller. What are its operation command words ?

AKTU 2014-15, Marks 10

OR

Draw and explain the internal architecture of 8259 interrupt controller. Also describe its initialization command words.

AKTU 2016-17, Marks 10

Answer

A. **Internal architecture of 8259 (8259 A) :** Refer Q. 5.17, Page 5–22C, Unit-5.

Scanned by TapScanner

B. The command word of 8259A are classified into following groups :

a. Initialization Command Words (ICWs) :

1. It is used to program the basic operation of 8259A.
2. Before it starts functioning, the 8259A must be initialized by writing two to four command words into the respective command word registers. These are called as initialization command words (ICWs).

3. Types :

i. Initialization Command Word 1 (ICW 1) :

1. A write command issue to the 8259A with $A_0 = 0$ and $D_4 = 1$, is recognized as ICW 1.
2. During ICW 1, the following steps occurs :

a. It reset edge sense circuit. This implies that after initialization, an interrupt must make a low to high transition.

b. The interrupt mask register is cleared.

c. Lowest priority is assigned to IR_7.

d. The slave mode address is set to 7 (1, 1, 1).

e. It clears special mode and sets the status read to IRR.

f. If bit D_0 in ICW 1 is set to '0', then it clears all functions associated with ICW 4.

ii. Initialization Command Word 2 (ICW 2) :

1. A write command issue after ICW 1 with $A_0 = 1$, is recognized as ICW 2.
2. The ICW 2 format is shown in Fig. 5.19.1.

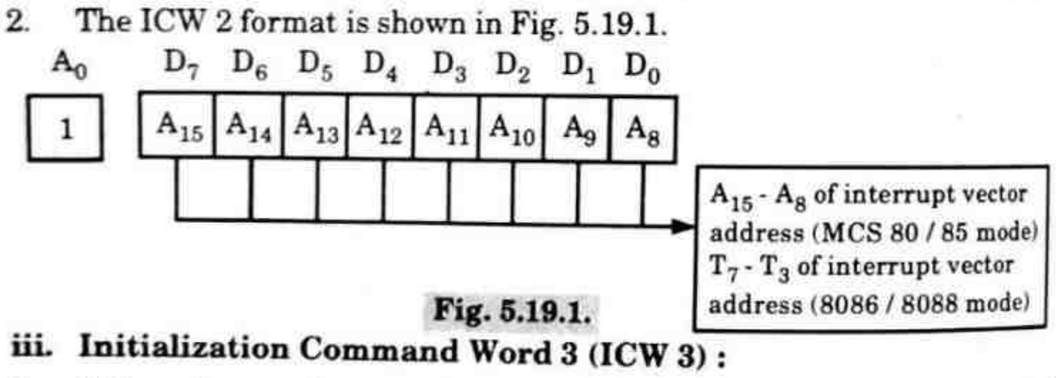


Fig. 5.19.1.

iii. Initialization Command Word 3 (ICW 3) :

1. If there is more than one 8259A in the system and if they are cascaded then ICW 3 is required.
2. There are two types of ICW 3 : Master ICW 3 and Slave ICW 3.
3. The master ICW 3 is used to specify whether it has a slave 8259 connected to its interrupt request input. The slave ICW 3 is used to assign a slave identification number.

iv. Initialization Command Word 4 (ICW 4) :

1. The ICW 4 is loaded only if the D_0 bit of ICW 1 is set.
2. The ICW 4 is used to initialize the 8259 in the following modes :

a. Special Fully Nested Mode (SFNM)

b. Buffered Mode (BUF)

c. Auto EOI Mode (AEOI)

d. 8086/8088 Mode

b. Operation Command Words :

1. Once 8259A is initialized, it is ready for its normal function, *i.e.*, for accepting the interrupts but 8259A.
2. However, during operation, it might be necessary to change the manner of processing the interrupts.
3. Operation Command Word (OCW) is used for this purpose.

4. Types :

i Operation Command Word 1 (OCW 1) :

1. OCW 1 is used to mask the unwanted interrupt requests. If the mask bit is '1', the corresponding interrupt request is masked, and if it is '0', the request is enabled.
2. A write command issued after initialization with $A_0 = 1$ is considered as OCW 1.

ii. Operation Command Word 2 (OCW 2) :

1. In OCW 2 the three bits, viz. R, SL and EOI control the end of interrupt, the rotate mode and their combinations.
2. The three bits L_2, L_1 and L_0 in OCW 2 determine the interrupt level to be selected for operation.

iii. Operation Command Word 3 (OCW 3) :

1. In OCW 3, if the ESMM bit, *i.e.*, enable special mask mode bit is set to '1', the SMM bit is enabled to select or mask the special mask mode.
2. When ESMM bit is '0', the SMM bit is neglected. If the SMM bit, *i.e.*, special mask mode bit is '1' the 8259 will enter special mask mode provided ESMM = 1.
3. If ESMM = 1 and SMM = 0, the 8259A will return to the normal mask mode.

Que 5.20. How many 8259 can be interconnected in cascade mode ? Show their cascading structure. **AKTU 2018-19, Marks 07**

Answer

i. Nine 8259 can be interconnected.

ii. **Cascading structure :**

1. One 8259 as master and other 8 number of 8259 as slave in cascade can generate 64-priority interrupt levels.
2. The master controls the slaves with $CSA_2 \Leftrightarrow CSA_0$ pins acting as the chip select inputs for slaves.

Scanned by TapScanner

3. The slave INT outputs are connected to the master IR inputs. When a slave request line is activated and acknowledged, the master will enable the slave to release the vector address during the 2nd pulse of $\overline{INTA}$ sequence.

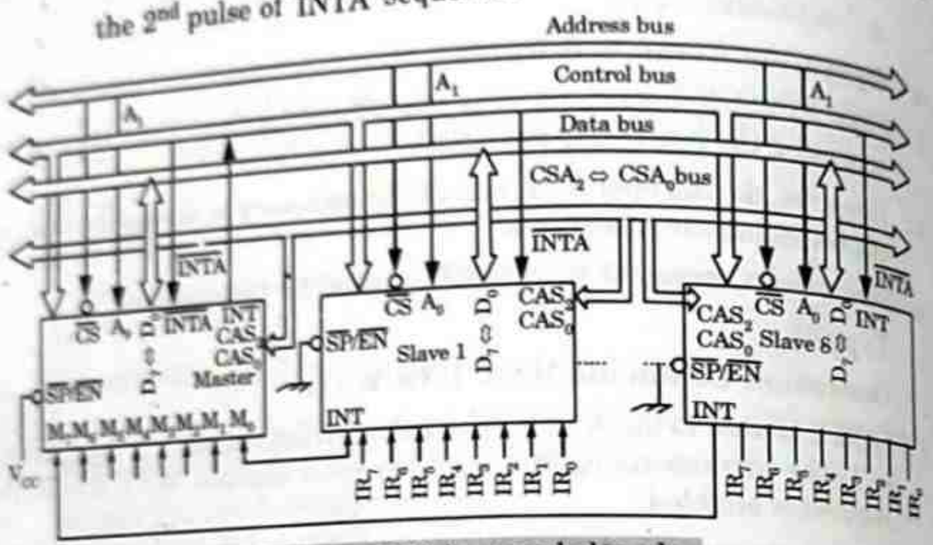


Fig. 5.20.1. 8259 in cascaded mode.

4. The cascaded lines are normally held low and contain slave address codes from the trailing edge of the the 1st $\overline{INTA}$ pulse to the trailing edge of the 2nd $\overline{INTA}$ pulse.
5. The separate EOI commands must be issued twice, once for the master and the other for the slave.
6. A separate decode is required to select each chip of the 8259 by activating its chip select pin. Fig. 5.20.1 shows the details of the cascaded mode of 8259.

PART-5

8251 USART.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.21. **What is USART ? What are its features ?**

Answer

A. Programmable communication interface 8251 USART :

1. Intel's 8251 A is a universal synchronous asynchronous receiver and transmitter compatible with Intel's processors.

2. This may be programmed to operate in any of the serial communication modes built into it.
3. This chip converts the parallel data into a serial stream of bits suitable for serial transmission.
4. It is also able to receive a serial stream of bits and convert it into parallel data bytes to be read by a microprocessor.

B. Features of 8251 USART :

1. 28-pin DIP package, all inputs and outputs are TTL compatible.
2. Single + 5 V supply.
3. Compatible with 8085 and 8086/8088 CPU.
4. Supports both synchronous and asynchronous modes of operations.
5. In synchronous mode it supports 5-8 bit characters, internal or external character synchronization at receiver, automatic sync insertion at transmitter.
6. In asynchronous mode it supports 5-8 bit characters, clock rate selectable 1, 16 or 64 times baud rate, break character generation, '1, 1½ or 2 stop bits, false start bit detection, odd, even or no parity generation and detection, automatic breaks detect circuitry are also available.
7. Synchronous baud rate - DC to 64 k baud.
8. Transmitter and receiver contain full duplex, double buffered system.
9. Error detection-Parity, overrun, framing.
10. Separate TXC and RXC clock inputs for transmitter and receiver. So transmitter and receiver can be operated in different baud rates.

Que 5.22. **Explain the internal architecture of 8251.**

Answer

Architecture and signal descriptions of 8251 :

1. The architectural block diagram of 8251 A is shown in Fig. 5.22.1, followed by the functional description of each block.
2. The data buffer interfaces the internal bus of the circuit with the system bus.
3. The read write control logic controls the operation of the peripheral depending upon the operations initiated by the CPU. This unit also selects one of the two internal addresses those are control address and data address at the behest of the $C/\overline{D}$ signal.

Scanned by TapScanner

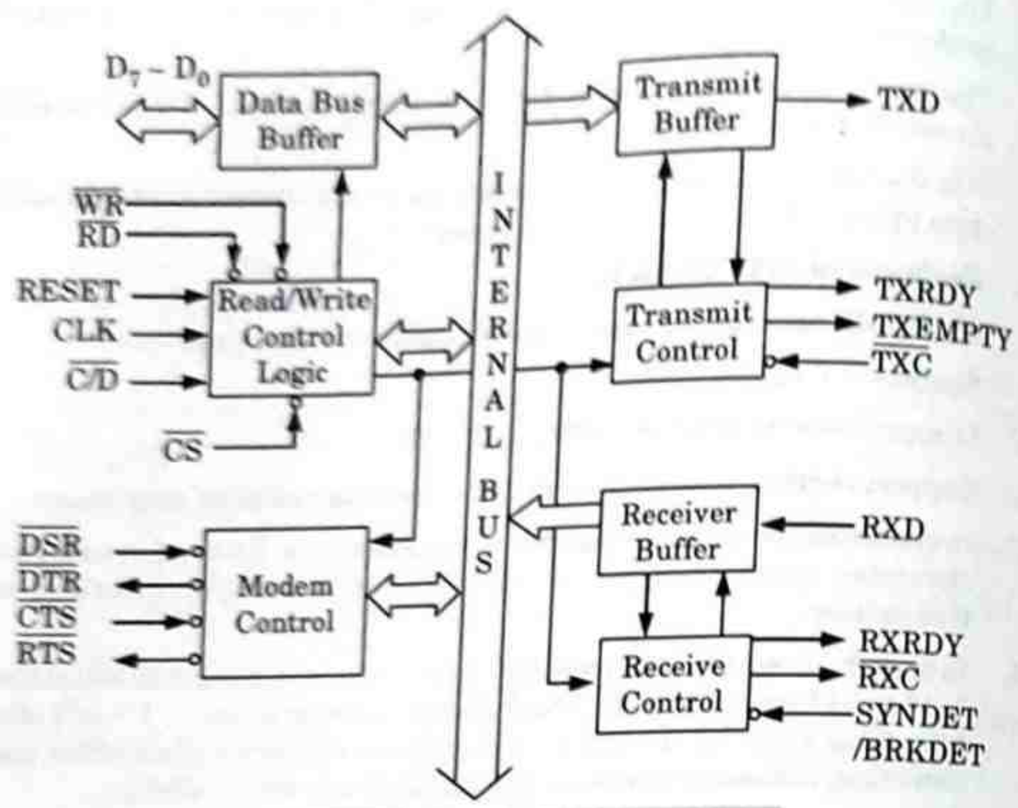


Fig. 5.22.1. 8251A Internal architecture.

4. The modem control unit handles the modem handshake signals to coordinate the communication between the modem and the USART. The transmit control unit transmits the data byte received by the data buffer from the CPU for further serial communication.
5. This decides the transmission rate which is controlled by the $\overline{TXC}$ input frequency. This unit also derives two transmitter status signals namely TXRDY and TXEMPTY. These may be used by the CPU for handshaking.
6. The transmit buffer is a parallel to serial converter that receives a parallel byte for conversion into a serial signal and further transmission onto the communication channel.
7. The receive control unit decides the receiver frequency as controlled by the $\overline{RXC}$ input frequency. This unit generates a receiver ready (RXRDY) signal that may be used by the CPU for handshaking.
8. This unit also detects a break in the data string while the 8251 is in asynchronous mode. In synchronous mode, the 8251 detect SYNC characters using SYNDET/BD pin.

Que 5.23. **Describe the pin configuration of USART.**

Answer

1. **D_0-D_7 :** This is an 8-bit data bus used to read or write status, command word or data from or to the 8251A.
2. ***C / $\overline{D}$* Control Word/Data :** This input pin, together with $\overline{RD}$ and $\overline{WR}$ inputs, informs the 8251A that the word on the data bus is either a data or control word/status information. If this pin is 1, control/status is on the bus, otherwise data is on the bus.
3. ***$\overline{RD}$* :** This active-low input to 8251A is used to inform it that the CPU is reading either data or status information from its internal registers.
4. ***$\overline{WR}$* :** This active-low input to 8251A is used to inform it that the CPU is writing data or control word to 8251A.
5. ***$\overline{CS}$* :** This is an active-low chip select input of 8251A. If it is high, no read or write operation can be carried out on 8251. The data bus is tristated if this pin is high.
6. **CLK :** This input is used to generate internal device timings and is normally connected to clock generator output.

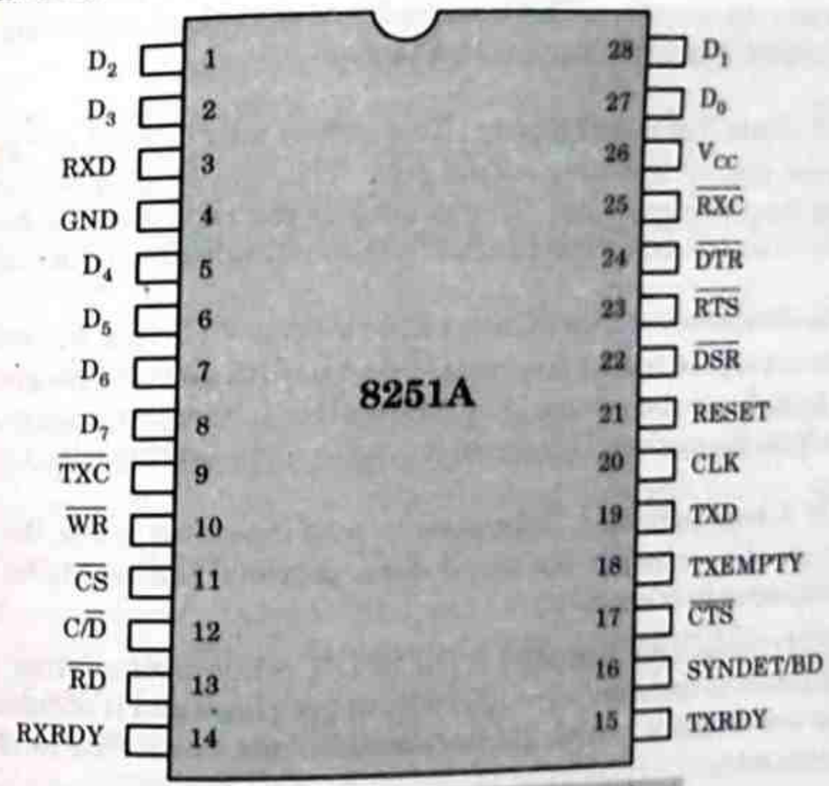


Fig. 5.23.1. 8251A pin configuration.

7. **RESET :** A high on this input forces the 8251A into an idle state. The device will remain idle till this input signal again goes low and a new set of control word is written into it.

Scanned by TapScanner

8. $\overline{TXC}$ **-Transmitter Clock Input :** This transmitter clock input controls the rate at which the character is to be transmitted.

9. **TXD-Transmitted Data Output :** This output pin carries serial stream of the transmitted data bits along with other information like start bit, stop bits and parity bit, etc.

10. $\overline{RXC}$ **-Receiver Clock Input :** This receiver clock input pin controls the rate at which the character to be received.

11. **RXD-Receive Data Input :** This input pin of 8251A receives a composite stream of the data to be received by 8251A.

12. **RXRDY-Receiver Ready Output :** This output indicates that the 8251A contains a character to be read by the CPU. The RXRDY signal may be used either to interrupt the CPU or may be polled by the CPU.

13. **TXRDY-Transmitter Ready :** This output signal indicates to the CPU that the internal circuit of the transmitter is ready to accept a new character for transmission from the CPU.

14. $\overline{DSR}$ **-Data Set Ready :** This input may be used as a general purpose one bit inverting input port. Its status can be checked by the CPU using a status read operation. This is normally used to check if the data set is ready when communicating with a modem.

15. $\overline{DTR}$ **-Data Terminal Ready :** This output may be used as a general purpose one bit inverting output port. This can be programmed low using the command word. This is used to indicate that the device is ready to accept data when the 8251A is communicating with a modem.

16. $\overline{RTS}$ **-Request to Send Data :** This output also may be used as a general purpose one bit inverting output port that can be programmed low to indicate the modem that the receiver is ready to receive a data byte from the modem. This signal is used to communicate with a modem.

17. $\overline{CTS}$ **-Clear to Send :** If the clear to send input line is low, the 8251A is enabled to transmit the serial data, provided the enable bit in the command byte is set to '1'.

18. **TXE-Transmitter Empty :** If the 8251A, while transmitting, has no characters to transmit, the TXE output goes high and it automatically goes low when a character is received from the CPU, for further transmission.

19. **SYNDET/BD-Synch Detect/Break Detect :** This pin is used in the synchronous mode for detecting SYNC characters (SYNDET) and may be used as either input or output.

PART-6

RS-232C.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.24. Explain RS-232C.

Answer

A. **The standards :** There are several standards that specify protocol for data transfer between the devices and the electrical characteristics of line drivers and receivers. RS-232C, RS-422A, RS-423A and RS-485 are the few standards. RS-232C is one of the most commonly used standards.

B. **The RS-232C :**

1. This standard was mainly designed to connect Data Terminal Equipment (DTE) that are sending and receiving serial data (such as a computer) and Data Communication Equipment (DCE) that are used to send data over long distances (such as a modem).

2. The driver for RS-232C converts TTL logic low into a signal of +3 to + 25 V and TTL logic high into a signal of -3 to 25 V. The receiver converts the signals of +3 to -25 V to TTL logic low and -3 to -25 V to TTL logic high.

3. The ICs MC1488 and MC1489A are used as line drivers and receivers for RS-232C standard.

4. The MC1488 contains four drivers and each driver converts a TTL level signal to RS-232C level signal. The MC1489A contains four receivers that convert RS-232C level signals into TTL levels.

C. **Pins and signals :**

1. The standard prescribes the 25-pin connector for connecting the DTE and DCE. The rate of data transmission is restricted to a maximum of 20 k band and to a maximum distance of 50 feet.

2. Fig. 5.24.1 shows the pins and signals of RS-232C and the connection between DTE and DCE using line drivers and receivers.

i. **TXD and RXD :** The Transmit Data and Receive Data on the DTE are the serial data lines. These lines have opposite functions on a DCE.

Scanned by TapScanner

ii. $\overline{RTS}$ **and** $\overline{CTS}$ **:** The Request To Send is activated by the transmitter when it wishes to send data over the line. This line is active till the end of communication. The Clear To Send is activated by the receiver to inform the transmitter whether it is ready or not ready to accept the data. It is also held active throughout the transmission.

iii. $\overline{DTR}$ **and** $\overline{DSR}$ **:** The DTE informs the DCE through the Data Terminal Ready line that it is in on-line mode and communication is possible. The Data Set Ready signal is to indicate that DCE is ready for communication.

iv. $\overline{DCD}$ **:** The Data Carrier Detect is activated by the DCE to indicate that it has established a contact with DTE. It is usually activated when RTS is granted.

v. **RI :** The Ring Indicator is activated by DCE when it detects an incoming call on the telephone line.

3. The following sequence of signal handshaking occurs when a computer sends a data to a modem. The computer activates the $\overline{RTS}$ signal to modem and the modem in turn activates the $\overline{DCD}$ and then $\overline{CTS}$.

4. The computer then sends data on TXD. After the data transmission is complete the computer deactivates the $\overline{RTS}$ which causes the modem to deactivate $\overline{CTS}$.

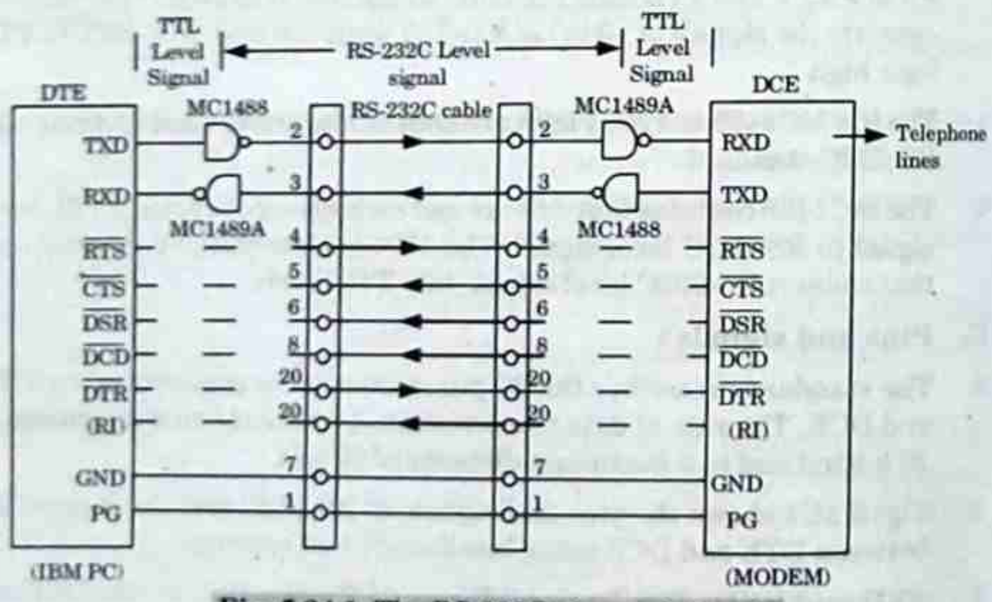


Fig. 5.24.1. The RS-232C pins and signals.

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. What do you understand by DMA ? With the help of block diagram explain the working of 8237/8257.

Ans. Refer Q. 5.2.

Q. 2. What is meant by direct memory access ? With the help of neat schematic, explain the interaction between 8086 MPU and DMA controller for providing direct memory access to a peripheral device.

Ans. Refer Q. 5.4.

Q. 3. What are the features of 8255 PPI ?

Ans. Refer Q. 5.5.

Q. 4. Draw and explain the internal architecture of 8255. What are its modes of operation ?

Ans. Refer Q. 5.7.

Q. 5. Draw and explain the internal architecture of 8255 parallel I/O peripheral device. Also describe the bits of control word.

Ans. Refer Q. 5.8.

Q. 6. What is 8253/8254 programmable interval timer, draw and explain its internal architecture.

Ans. Refer Q. 5.11.

Q. 7. Draw and explain block diagram and pin configuration of IC-8253.

Ans. Refer Q. 5.12.

Q. 8. Discuss the mode of operation of 8253 program, internal times with its control format.

Ans. Refer Q. 5.14.

Q. 9. Explain 8259 programmable interrupt controller.

Ans. Refer Q. 5.15.

Scanned by TapScanner