t.me/quantum_aktu

aktu quantum

please support
us

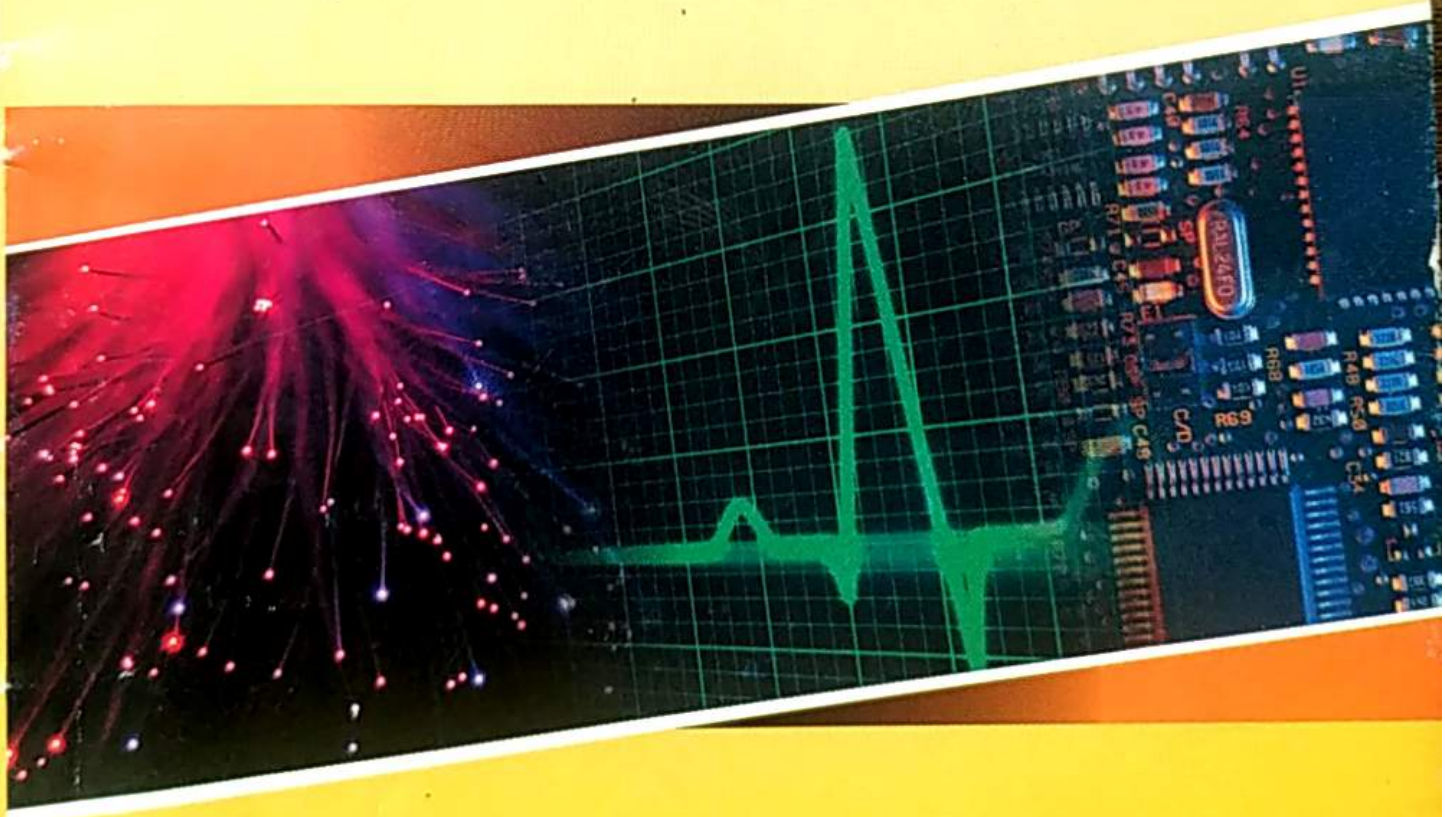
- 5 Sem
- 6 Sem
- 7 Sem
- 8 Sem - future support

Quantum Series®
ENGINEERING

QUANTUM *Series*

Semester - 5 **Electronics & Allied Branches**

Microprocessor & Microcontroller



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

CONTENTS

KEC–502 : MICROPROCESSORS & MICROCONTROLLERS

UNIT-1 : INTRODUCTION TO MICROPROCESSOR (1–1 B to 1–39 B)

Microprocessor architecture and its operations, Memory, Input & output devices, The 8085 MPU-architecture, Pins and signals, Timing Diagrams, Logic devices for interfacing, Memory interfacing, Interfacing output displays, Interfacing input devices, Memory mapped I/O.

UNIT-2 : BASIC PROGRAMMING CONCEPTS (2–1 B to 2–26 B)

Flow chart symbols, Data Transfer operations, Arithmetic operations, Logic Operations, Branch operation, Writing assembly language programs, Programming techniques: looping, counting and indexing. Additional data transfer and 16 bit arithmetic instruction, Logic operation: rotate, compare, counter and time delays, 8085 Interrupts.

UNIT-3 : 16-BIT MICROPROCESSORS (8086) (3–1 B to 3–39 B)

16-bit Microprocessors (8086): Architecture, Pin Description, Physical address, segmentation, memory organization, Addressing modes.
Peripheral Devices: 8237 DMA Controller, 8255 programmable peripheral interface, 8253/8254 programmable timer/counter, 8259 programmable interrupt controller, 8251 USART and RS232C.

UNIT-4 : 8051 MICROCONTROLLER BASICS (4–1 B to 4–21 B)

Inside the Computer, Microcontrollers and Embedded Processors, Block Diagram of 8051, PSW and Flag Bits, 8051 Register Banks and Stack, Internal Memory Organization of 8051, IO Port Usage in 8051, Types of Special Function Registers and their uses in 8051, Pins Of 8051. Memory Address Decoding, 8031/51 Interfacing With External ROM And RAM. 8051 Addressing Modes.

UNIT-5 : ASSEMBLY PROGRAMMING & INSTRUCTION OF 8051 (5–1 B to 5–33 B)

Assembly programming and instruction of 8051: Introduction to 8051 assembly programming, Assembling and running an 8051 program, Data types and Assembler directives, Arithmetic, logic instructions and programs, Jump, loop and call instructions, IO port programming. Programming 8051 Timers. Serial Port Programming, Interrupts Programming, Comparison of Microprocessor, Microcontroller, PIC and ARM processors and their application areas.
Interfacing: LCD & Keyboard Interfacing, ADC, DAC & Sensor Interfacing, External Memory Interface, Stepper Motor and Waveform generation.

SHORT QUESTIONS (SQ-1 B to SQ-15 B)

1 UNIT

Introduction to Microprocessor

CONTENTS

Part-1 :	Introduction to Microprocessor : Microprocessor Architecture and its Operations	1-2B to 1-7B
Part-2 :	Memory, Input and Output Devices	1-7B to 1-13B
Part-3 :	The 8085 MPU-Architecture, Pins and Signals, Timing Diagrams, Logic Devices for Interfacing, Memory Interfacing	1-14B to 1-33B
Part-4 :	Interfacing Output Displays, Interfacing Input Devices, Memory Mapped I/O	1-33B to 1-38B

PART-1

Introduction to Microprocessor : Microprocessor Architecture and Its Operations.

CONCEPT OUTLINE

- The operations performed by microprocessor can be classified into three groups :
  - i. Internal operations
  - ii. Operation initiated by microprocessor
  - iii. Operation initiated by external devices.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.1. **What is a microprocessor ? Differentiate between microprocessor and microcontroller.**

Answer

A. Microprocessor :

Microprocessor is a multipurpose, programmable, clock driven, register based electronic device that reads binary instructions from a storage device called memory; accepts binary data as input, process data according to instructions, and provides output as a result.

B. Difference between microprocessor and microcontroller :

S. No.	Microprocessor	Microcontroller
1.	Microprocessor is complete functional CPU, *i.e.*, it contains ALU, registers, control unit and interrupt processing circuits.	Microcontroller is complete functional microcomputer, *i.e.*, it contains the circuitry of microprocessor with memory (ROM, RAM), I/O circuit and peripherals.

2.	Microprocessor instructions are mainly nibble or byte addressable.	Microcontroller instructions are both bit as well as byte addressable.
3.	Access time for external memory and I/O devices are more, resulting in a slower system.	Access time for on-chip memory and I/O devices are less, resulting in a faster system.
4.	It is used for general purpose application.	It is used for special purpose application.
5.	Software protection is not possible because of the requirement of external code memory.	Software protection is possible because of on-chip code memory.
6.	More expensive.	Less expensive.

Que 1.2. **Explain various types of microprocessor (μP).**

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.
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 1.3. **How does the microprocessor work ? Explain in detail.**

Answer

1. The process of program execution in a microprocessor can be described in the following sequence: read, interpret and perform.
2. The instructions are stored sequentially in the memory. The microprocessor fetches the first instruction from its memory, decodes it and executes that instruction.

3. The sequence of fetch, decode, and execute is continued until the microprocessor comes across an instruction to stop.
4. During the entire process, the microprocessor uses the system bus to fetch the binary instructions and data from the memory.
5. It uses registers from the register section to store data temporarily, and it performs the computing function in the ALU section.
6. Finally, it sends out the result in binary, using the same bus lines, to the output device.

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

Answer

A. 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 most general purpose architecture of microprocessor is shown in Fig. 1.4.1.

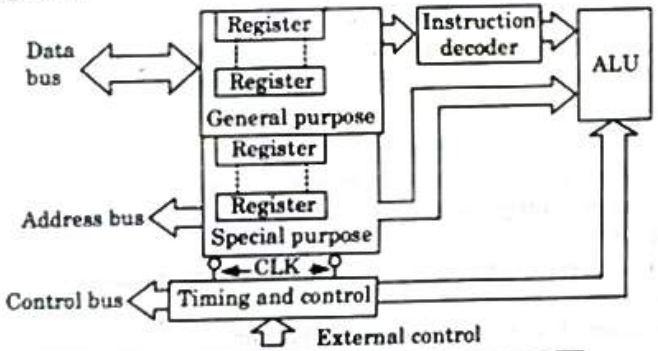


Fig. 1.4.1. General architecture of a microprocessor.

B. Components :

1. **ALU (Arithmetic logical unit) :** It consists of an adder, an accumulator, a temporary register, and a shift register and a status register. This ALU unit performs various arithmetic and logical operation.
2. **General purpose register :** In the microprocessor, there are 8-bit general purpose register or as a 16-bit register pairs, when used in register pair mode. These are used for both storing data as well as the address.

3. **Special purpose register :** This consist of accumulator, program counter (PC), stack pointer (SP) and status flag register. 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 that regulates the transfers and transformations in the system associated with each instruction 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 :** This is used for transmitting address information. 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 single lines used for carrying synchronization signals. The microprocessor uses such lines for providing timing signals.

Que 1.5. What are the operations performed by microprocessor ?

Answer

The operations performed by microprocessor can be classified as follows :

i. Internal operations :

1. Store 8-bit data.
2. Perform arithmetic and logical operations.
3. Test for conditions.
4. Sequence the execution of instructions.
5. Store data temporarily into the stack.

ii. Operations initiated by microprocessor : To communicate with external devices or with memory, microprocessor performs primarily four operations :

1. Read data or instruction from the memory.
2. Write data into the memory.
3. Read data from input devices.
4. Write data into the output devices.

iii. Operations initiated by external devices : External devices can initiate the operation by activating Reset, Interrupt, READY and HOLD pins of the microprocessor. These operations are as follows :

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 1.6. **List the four operations commonly performed by the MPU.**

AKTU 2014-15, Marks 05

Answer

1. The four operations commonly performed by the microprocessor (MPU) are as follows :

i. **Memory read :** Reads data (or instructions) from memory.

ii. **Memory write :** Writes data (or instructions) into memory.

iii. **I/O read :** Accepts data from input devices.

iv. **I/O write :** Sends data to output devices.

2. All these operations are part of the communication process between the MPU and peripheral devices.
3. To communicate with a peripheral (or a memory location), the MPU needs to perform the following steps :

i. Identify the peripheral or the memory location (with its address).

ii. Transfer binary information (data and instructions).

iii. Provide timing or synchronization signals.

PART-2

Memory, Input and Output Devices.

CONCEPT OUTLINE

- Memory is a medium that stores binary information. They are classified into two groups :

1. Primary or main memory.
2. Secondary or storage memory

- Input and output devices are the means by which microprocessor communicates with the outside world.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.7. **What do you understand by memory and give its classification ?**

OR

Explain primary memory.

Answer

A. **Memory :** A memory unit is a device to which binary information is transferred for storage and from which information is retrieved when needed for processing.

B. **Classification :**

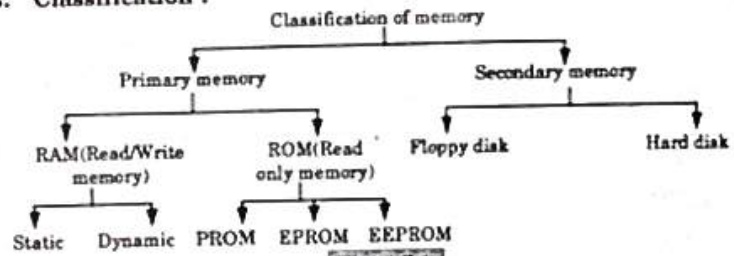


Fig. 1.7.1.

C. **Primary Memory**

i. **RAM :**

1. It is the memory device which stores the binary information permanently.
2. It allows only retrieval of permanently stored data and does not permit modifications of the stored information during normal operation.

ii. **ROM :**

1. Read-only memory (ROM) is a class of storage media used in computers and other electronic devices.
2. ROMs are non-volatile memories, *i.e.*, the stored data are not lost even when the power supply is OFF and refresh operation is not required.
3. Because it cannot (easily) be written to, its main use lies in the distribution of firmware (software that is very closely related to hardware and not likely to need frequent upgrading).
4. Modern semiconductor ROMs typically take the shape of IC packages.

5. ROM in its strictest sense can only be read from, but all ROMs allow data to be written into them at least once, either during initial manufacturing or during a step called "programming".

6. There are three types of ROM :

a. **PROM:**

1. A programmable read-only memory (PROM) is a form of digital memory where the setting of each bit is locked by a fuse or antifuse. Such PROMs are used to store programs permanently.

2. A typical PROM comes with all bits reading as 1, burning fuse during programming causes its bit to read as 0.

3. The memory can be programmed just once after manufacturing by "blowing" the fuses, which is an irreversible process. Blowing a fuse opens a connection while blowing an antifuse closes a connection (hence the name).

b. **EPROM:**

1. An EPROM, or erasable programmable read-only memory, is a type of computer memory chip that retains its data when its power supply is switched OFF. In other words, it is non-volatile.

2. Once programmed, an EPROM can be erased only by exposing it to strong ultraviolet light.

c. **EEPROM:**

1. An EEPROM (also called an E2PROM) or electrically erasable programmable read-only memory is a non-volatile storage chip used in computers and other devices to store small amounts of volatile (configuration) data.

2. The main advantage of EEPROMs over EPROMs is that they are erased electrically instead of by ultraviolet light; this is faster and can be done in circuit.

Que 1.8. **What is transparent latch, and why it is necessary to use a latch with output device ?** **AKTU 2014-15, Marks 05**

Answer

1. A latch is a *D* flip-flop. Two types of *D* flip-flops are available, a transparent latch and a positive-edge-triggered flip-flop.

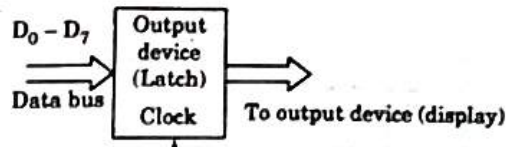


Fig. 1.8.1.

2. In a transparent latch, when the clock signal is high, the output *Q* changes according to the input *D*.

3. When the clock signal goes low, the output *Q* will latch (hold) the last value of the input *D*.

4. Latch is the simplest form of output port.

5. The output device is connected to microprocessor through latch.

6. When microprocessor wants to send data to the output device it puts the data on data bus and activates the clock signal of latch, latching the data from the data bus to output of latch.

7. It is then available at output of latch for the output device.

8. A latch is used for an output port because latch is necessary to hold the output data for display; however the input data byte is obtained by enabling a tri-state buffer and placed in the accumulator.

Que 1.9. **Explain the memory address range of 1k memory and explain the changes in the address if the hardware of the $\overline{CS}$ line is modified. The total available address lines for the addressing are 16.**

Answer

1. The memory chip has 1024 registers therefore 10 address line ($A_9 - A_0$) are required to identify the registers. The remaining six address lines ($A_{15} - A_{10}$) of the microprocessor are used for the chip select ($\overline{CS}$) signal.

2. In Fig. 1.9.1, the memory chip is enabled when the address lines $A_{15} - A_{10}$ are at logic 0. The address lines $A_9 - A_0$ can assume any address of the 1024 registers, starting from all 0's to all 1's.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= 0000 H
Chip select logic						1	1	1	1	1	1	1	1	1	1	= 03FF H

3. The memory addresses range from 0000 H to 03FF H.

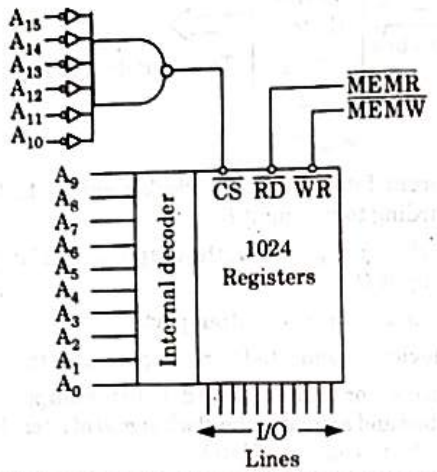


Fig. 1.9.1. Memory address range 1024 bytes of memory.

4. The memory addresses of the 1 K chip in Fig. 1.9.1 can be changed to any other location by changing the hardware of the $(\overline{CS})$ line. For example, if A_{15} is connected to the NAND gate without an inverter, the memory addresses will range from 8000 H to 83FF H.

Que 1.10. Draw and explain the memory map of 4 KB memory interface between 8085 microprocessor (μp) and memory.

Answer

1. Fig. 1.10.1 shows a memory chip of 4 KB. The memory size of chip is expressed as,

   4 KB = 4 × 1024 bits = 2^{12}

2. As we know, 8085 microprocessor has 16 address lines. So, for addressing the memory size we shall need 12 address lines only. Now remaining lines (16 – 12 = 4), will be used for chip select.

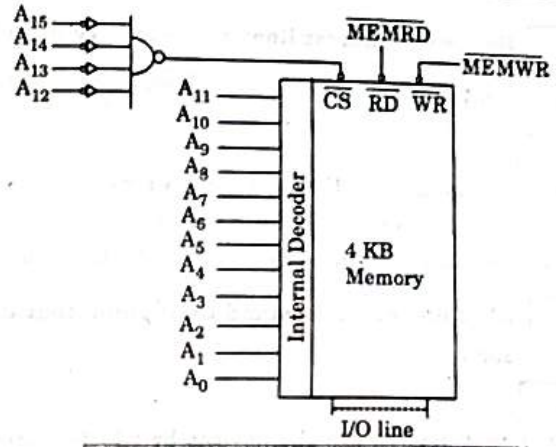


Fig. 1.10.1. Memory map : 4 KB of memory.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
Chip select				Memory address												
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0 = 0000 H
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1 = 0FFF H

3. The address lines $A_{15} - A_{12}$, which are used to select the chip, must have fixed logic levels, and the address lines $A_{11} - A_0$ can be assigned logic levels from all 0's to all 1's and any one in between combination.

4. For example, when $A_{11} - A_0$ are all 0's then register number zero is selected and if all are 1's then register number FFF is selected.

5. The memory chip can be same for selecting other location of address.

   However, by changing the hardware of the chip select logic, the location of the memory in the map can be changed.

   So, if we will take A_{15} as 1 then,

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

6. The location of memory address have been shifted from 0000 H – 0FFF H to 8000 H – 8FFF H and the hardware of the chip select will also change.

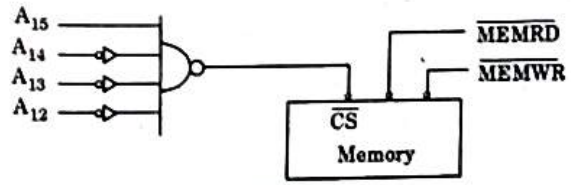


Fig. 1.10.2.

Que 1.11. **How many address lines are necessary to address 8 K byte of memory ?** **AKTU 2014-15, Marks 05**

Answer

Address lines to address 8 K byte of memory are

$$8\ K = 2^3 \times 1024 = 2^3 \times 2^{10} = 2^{13}$$

So, 13 address lines are necessary to address 8 K byte of memory.

Que 1.12. **What do you understand by input/output devices of microprocessor ?**

Answer

1. Input/Output devices are the devices by which microprocessor communicates with the outside world.
2. The microprocessor accepts data in form of binary digits as a input from the devices such as keyboard, and A/D converters and it sends data to the output devices such as LEDs or printers.
3. There are two types of input/output devices :

A. I/O with 8-bit addresses (peripheral-mapped I/O) :

1. In this type of input/output devices, 8 address lines are used to identify an input or output device, this is known as peripheral mapped I/O (also known as I/O mapped I/O).
2. This is an 8-bit numbering system for I/O used in conjunction with input and output instruction. This is also known as I/O space which is 16-bit numbering system.
3. The 8 address lines can have 256 addresses. Thus, microprocessor can identify 256 input and 256 output devices with an address ranging from 00H to FFH.
4. The microprocessor uses I/O read control signals for the input devices and I/O write control signal for output devices.

B. I/O with 16-bit addressing (memory-mapped I/O) :

1. In this type of I/O, 16 addressing lines are used by the microprocessor to identify an I/O device.
2. An I/O device is connected as if it is a memory register. This is known as memory-mapped I/O. The microprocessor uses same control signal and instructions as those of memory.
3. In some microprocessor, all I/Os have 16-bit addresses : I/O and memory share same memory map.
4. The entire range of memory-mapped I/O address is from 0000 H to FFFF H.

PART-3

The 8085 MPU-Architecture, Pins and Signals, Timing Diagrams, Logic Devices for Interfacing, Memory Interfacing.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.13. **Explain the pin configuration of 8085 MPU with neat diagram.**

Answer

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

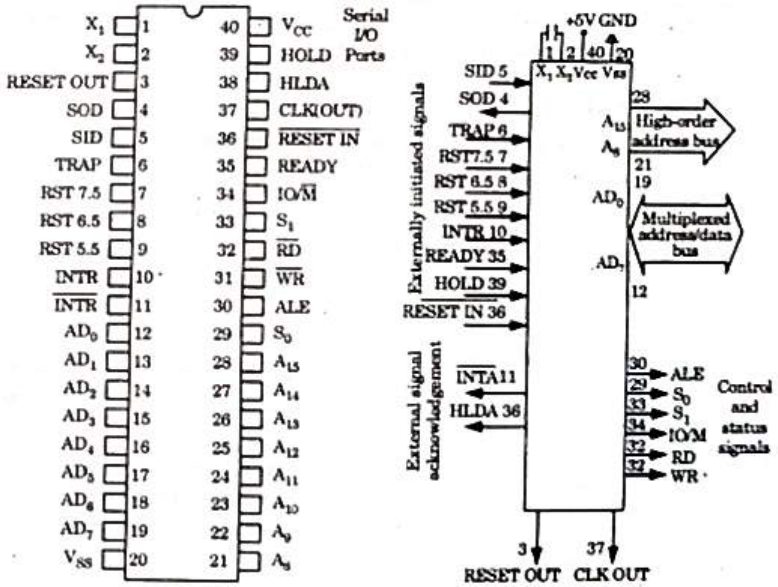


Fig. 1.13.1. Pin configuration and functional pin diagram.

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.

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) :** 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. 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.

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 master 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 (0000 H).

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 1.14. **With a neat diagram describe the internal architecture of 8085. State the function of each block shown.**

AKTU 2014-15, Marks 05

OR

What is the need of de-multiplexing of 8085 ? Discuss the microprocessor architecture and its operation.

AKTU 2015-16, Marks 10

OR

Draw a functional diagram of 8085 microprocessor and also discuss its various pins. **AKTU 2015-16, Marks 10**

Answer

A. **Pins of 8085 :** Refer Q. 1.13, Page 1–14B, Unit-1.

B. **Internal architecture or functional block diagram :** The 8085 is an 8-bit, general-purpose microprocessor that is ideally suited to many applications. Fig. 1.14.1 shows the architecture of 8085.

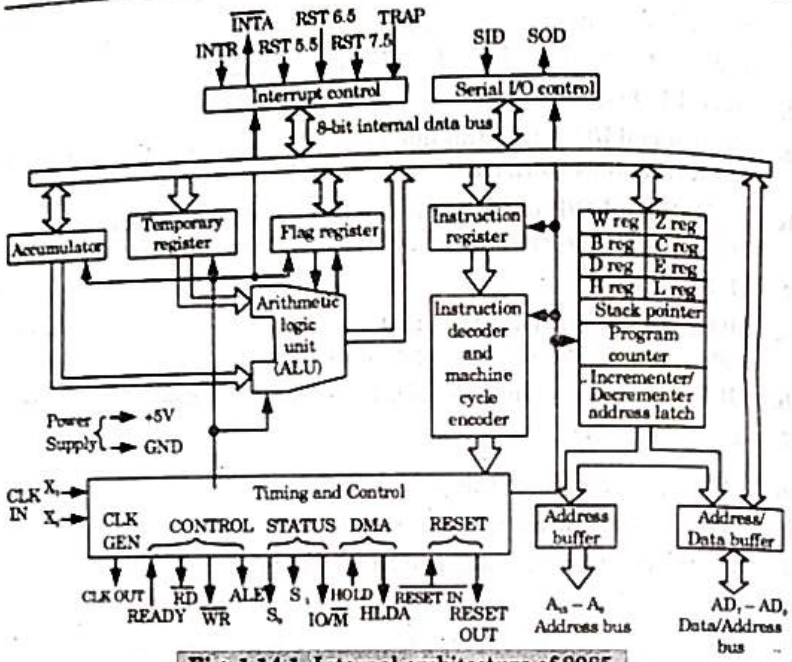


Fig. 1.14.1. Internal architecture of 8085.

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.

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. 1.14.2.

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

Fig. 1.14.2.

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 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.

C. **Need of de-multiplexing :**

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.

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. To achieve this, the data bus $AD_0 - AD_7$ is needed to be demultiplexed.

Que 1.15. **Explain the flag register of 8085 microprocessor with the help of example.** **AKTU 2016-17, Marks 05**

Answer

Flag register : It is 8-bit register in which five bits carry significant information in form of flags as shown in Fig. 1.15.1.

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

Fig. 1.15.1. Flag register.

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.

Example 1 :

85 H = 10000101
1E H = 00011110
A3 H = 10100011

Since, 8th bit of the result is 1, so the number is negative.

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.

Example 2 : XRA A ;

It set the zero flag because this operation reset the content of A.

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.

In Example 1, the carry is generated at 4th bit of the result, hence AC = 1.

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.

Example 3 :

38 H = 00111000

Since, the number of 1's is odd, hence the parity flag is zero, *i.e.*, P = 0.

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.

iii. In both the examples shown below carry flag is set.

ADDITION	SUBTRACTION
9B H	89 H
+ 75 H	− AB H
Carry $\boxed{1}$ 10 H	Borrow $\boxed{1}$ DE H

Que 1.16. **Discuss the following :**

i. **Instruction cycle and machine cycle.**

ii. **T-states with typical timing diagram.**

AKTU 2014-15, Marks 05

Answer

i. Instruction cycle and machine cycle :

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. Some instruction, which require register to register transfer inside the microprocessor, may be executed in one machine cycle, where as some which require data transfer between microprocessor and memory or input/output device may used more than one machine cycle.
6. Fig. 1.16.1 exhibits instruction cycle and machine cycle within it. It shows two machine cycles M_1 and M_2.
7. M_1 in which opcode is fetched and decoded consist of four state.
8. M_2 consists of three states, the data contained in the next byte of instruction is fetched and instruction executed.

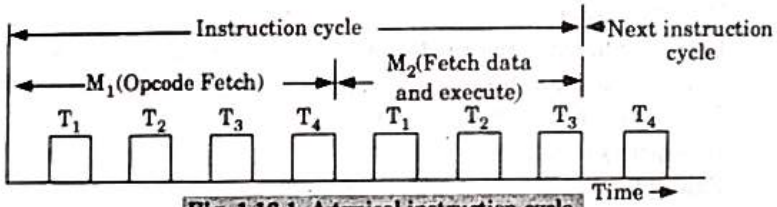


Fig. 1.16.1. A typical instruction cycle.

9. 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. 1.16.2.

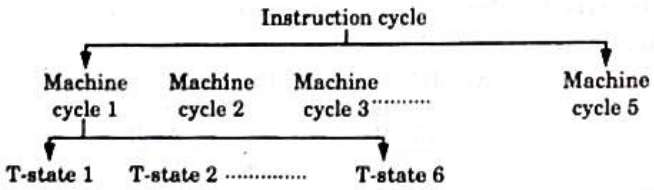


Fig. 1.16.2. Relationship between instruction cycle, machine cycle and T-state.

ii. T-state :

1. One machine cycle consist of several clock cycles each of these clock cycles are known as state.
2. The state within machine cycles is generally referred as T-states (T_1, T_2 ...).
3. One machine cycle consists of 3 to 6 T-states, *i.e.*, 3 to 6 clock periods.
4. The opcode fetch cycle has 4 T-states.
5. The 8085 uses the first 3 states to fetch the code and T_4 to decode and execute the opcode.
6. In 8085 instruction set, some instructions have opcodes with 6 T-states.
7. Ideally the clock signal should be square wave with zero rise time and fall time but in practice, we do not get zero rise time and fall time as shown in Fig. 1.16.3.

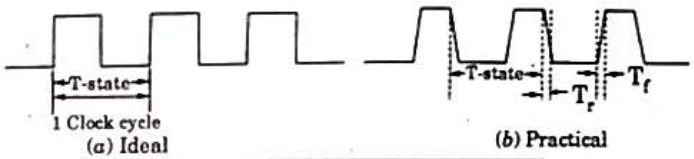


Fig. 1.16.3. Clock signal representation.

8. A typical timing diagram showing ALE activation and its period (T-state) is shown in Fig. 1.16.4.

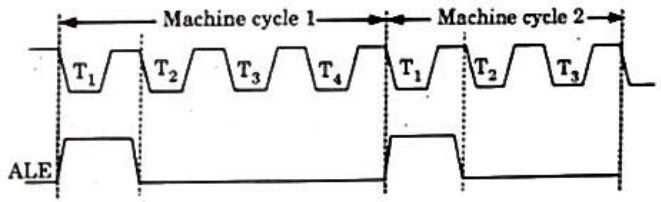


Fig. 1.16.4. ALE activation and its period.

9. ALE signal is activated in the beginning of T_1 state of each machine cycle except bus idle machine cycle.

Que 1.17. What is the importance of ALE and ALU ?

OR

What is function of accumulator ? **AKTU 2014-15, Marks 05**

Answer

A. Importance of ALE :

1. ALE is used to indicate the beginning of any operation.

2. It is a positive going pulse used primarily to latch lower order address from multiplexed bus ($AD_0 - AD_7$) and generate separate set of eight address line ($A_0 - A_7$).

B. Importance of ALU :

1. The ALU handles arithmetic and logical operations, as the name states it.
2. The CU (Control Unit) passes the operands required into the ALU for being processed, which in turn passes it out to registers in order to be saved and shown on the screen for the user to see.

C. Function of accumulator (ACC) :

1. The accumulator is an 8-bit register associated with the ALU.
2. The register A in the 8085 is an accumulator.
3. It is used to hold one of the operands of an arithmetic or logical operation.
4. It serves as one input to the ALU.
5. The other operand for an arithmetic or logic operation may be stored either in the memory or in one of the general-purpose registers.
6. The final result of an arithmetic or logical operation is placed in the accumulator.

Que 1.18. Explain the generation of control signal in 8085 microprocessor.

Answer

1. The 8085 provides $\overline{RD}$ and $\overline{WR}$ signals as control signals. $\overline{RD}$ signal is used both for reading memory and for reading an input device.
2. It is necessary to generate two different read signals, one for the memory and another for input. Similarly, two separate write signals are generated.

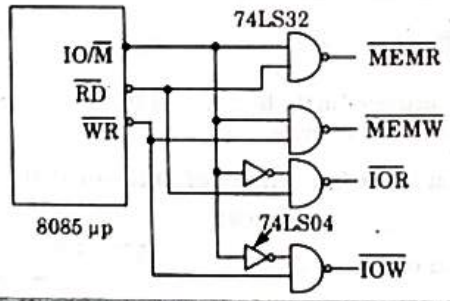


Fig. 1.18.1. Generation of Read/Write control signals for memory and I/O.

3. Fig. 1.18.1 shows that four different control signals are generated by combining the signals $\overline{RD}$, $\overline{WR}$ and $IO/\overline{M}$.

4. The signal $IO/\overline{M}$ goes low for the memory operation. This signal is ANDed with $\overline{RD}$ and $\overline{WR}$ signals by using the 74LS32 quadruple two-input OR gates, as shown in Fig. 1.18.1.
5. The OR gates are functionally connected as negative NAND gates. When both input signals go low, the outputs of the gates go low and generate $\overline{MEMR}$ (memory read) and $\overline{MEMW}$ (memory write) control signals.
6. When the $IO/\overline{M}$ signal goes high, it indicates the peripheral input/output operation. Fig. 1.18.1 shows that this signal is complemented using the hex inverter 74LS04 and ANDed with the $\overline{RD}$ and $\overline{WR}$ signals to generate $\overline{IOR}$ (I/O read) and $\overline{IOW}$ (I/O write) control signals.

Que 1.19. Discuss the various logic devices used in interfacing circuits.

AKTU 2014-15, Marks 05

OR

Write short note on logic devices for interfacing.

AKTU 2015-16, Marks 05

Answer

A. Tri-state devices :

1. These are the devices which have three logic states, *i.e.*, logic 0, logic 1 and high input impedance.
2. These devices have enable line, when activated they work like ordinary logic devices, but when the enable line is disabled then the logic devices switch into high impedance state and behave as they are disconnected from the system.

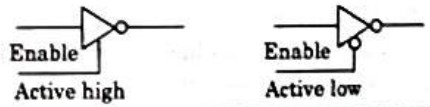


Fig. 1.19.1. Tri-state inverter with active high and active low.

B. Buffer :

1. It is a logic circuit that amplifies the current or power. It has one input and one output line.
2. It provides the same logic level at the output that is applied at the input.
3. Generally, it is used to increase the driving capability of the logic circuit hence known as driver.

4. Fig. 1.19.2 shows a diagram of tri-state buffer. In this buffer, when enable line is low then it behaves as buffer; otherwise it remains in high impedance state.

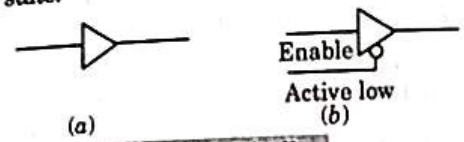


Fig. 1.19.2. A buffer.

C. Decoder :

1. Decoder is a logic circuit that identifies combination of signals that are applied at the input.
2. In decoder if there are n input lines then at the output there are 2^n output lines.
3. Suppose there is 3 × 8 decoder, *i.e.*, there are 3 input lines then there are 2^3, *i.e.*, 8 possible combination which are identified by 0 to 7 output lines.
4. If 111 is applied at the input then the output at 7 will be 1 and all others will remain at logic zero.

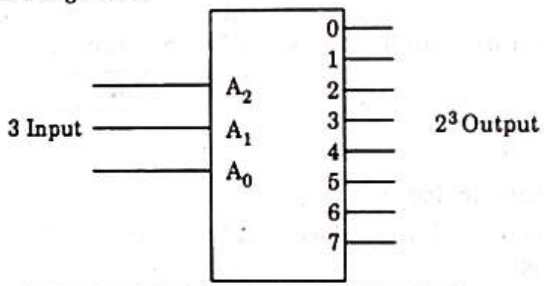


Fig. 1.19.3. 3 to 8 decoder logic symbol.

D. Encoder :

1. Encoder is a logic circuit which is used to provide the appropriate code at the output for each of the signals that are applied at the input. It is the reverse process of the decoding.
2. In the encoder, there are 2^n input lines and n output lines.
3. Suppose signal is applied at input 3 then at the output both LSB and MSB goes high.
4. Similarly, if signal is applied at input 2 then MSB goes high and LSB goes low.
5. But this encoder is unable to provide appropriate code at the output when two or more than two lines are activated simultaneously at a time.

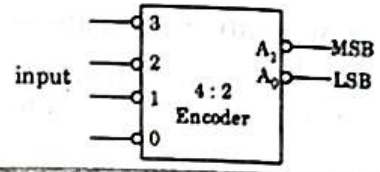


Fig. 1.19.4. Logic symbol for 4 : 2 encoder.

Que 1.20. What are the timing diagrams for memory read and memory write cycle ?

Answer

1. The main function of interfacing is that microprocessor should be able to read from and write into the given register or a memory chip.
2. To perform these operations microprocessor should be able to do following functions :

i. Select the chip.

ii. Identify the register.

iii. Enable the appropriate buffer.

3. To understand the interfacing concept, consider the timing diagram of memory read operation of microprocessor 8085.

i. Microprocessor places a 16-bits address on the address bus and with this address only one register is to be selected. From Fig. 1.20.1, it is clear that only 11 address lines are required to identify 2048 registers.

ii. Therefore, lower order address lines $A_{10} - A_0$ of microprocessor address bus is connected with memory chip. The internal decoder of the memory chip will identify and select the register for the EPROM.

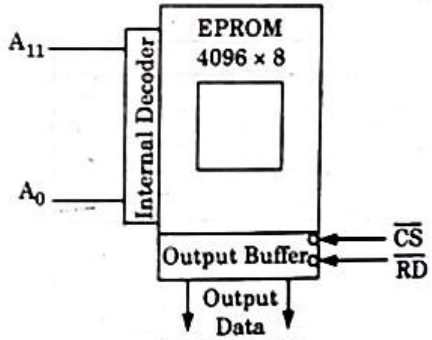


Fig. 1.20.1.

iii. The remaining address lines ($A_{15} - A_{11}$) of 8085 should be decoded to generate a (chip select) signal which is unique to that combination of address logic.

iv. The two signals $IO/\overline{M}$ and $\overline{RD}$ are provided by 8085 to indicate memory read operation. These two signals $IO/\overline{M}$ and $\overline{RD}$ can be combined to generate the (memory read) signal that can be used to enable the output buffer by connecting to the memory signal $\overline{RD}$.

4. Fig. 1.20.2 shows the memory places the data byte from the addressed register during T_2 and that is read by 8085 before end of T_3.

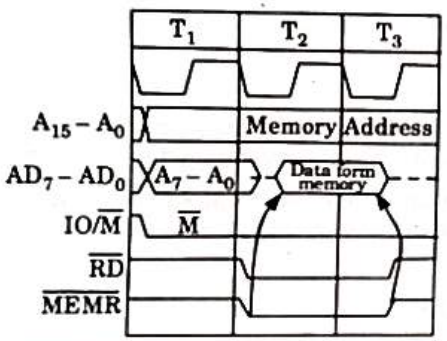


Fig. 1.20.2. Timing diagram of memory read cycle.

5. During write operation into the register microprocessor performs similar steps as in read operation. Fig. 1.20.3 shows memory write cycle, during write operation microprocessor places the address and data and asserts the $IO/\overline{M}$ signal.

6. After allowing sufficient time for data to become stable, it asserts the $\overline{RD}$ (write) signal. The $IO/\overline{M}$ and $\overline{WR}$ signals can be combined to generate $\overline{MEMW}$ signal.

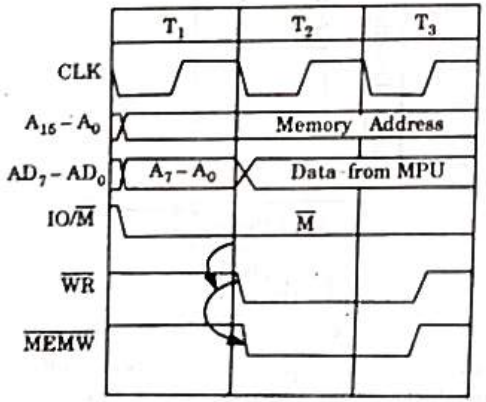


Fig. 1.20.3. Timing diagram of the memory write cycle.

Que 1.21. **Draw a logic diagram of complete interfacing of 8085 microprocessor with memory of size 4K bytes.**

AKTU 2015-16, Marks 10

Answer

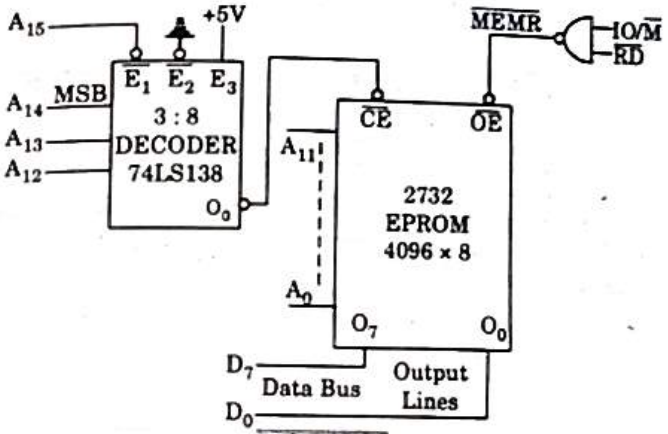


Fig. 1.21.1.

Que 1.22. **Draw a neat diagram for interfacing 8K SRAM and 8K EPROM with the system lines of 8085 microprocessor with memory map.**

AKTU 2016-17, Marks 05

Answer

1. Fig. 1.22.1 shows the desired memory system using IC 2764 (8K) EPROM and 6264 (8K) SRAM memory requires 13 address lines ($A_0 - A_{12}$) since $2^{13} = 8K$.

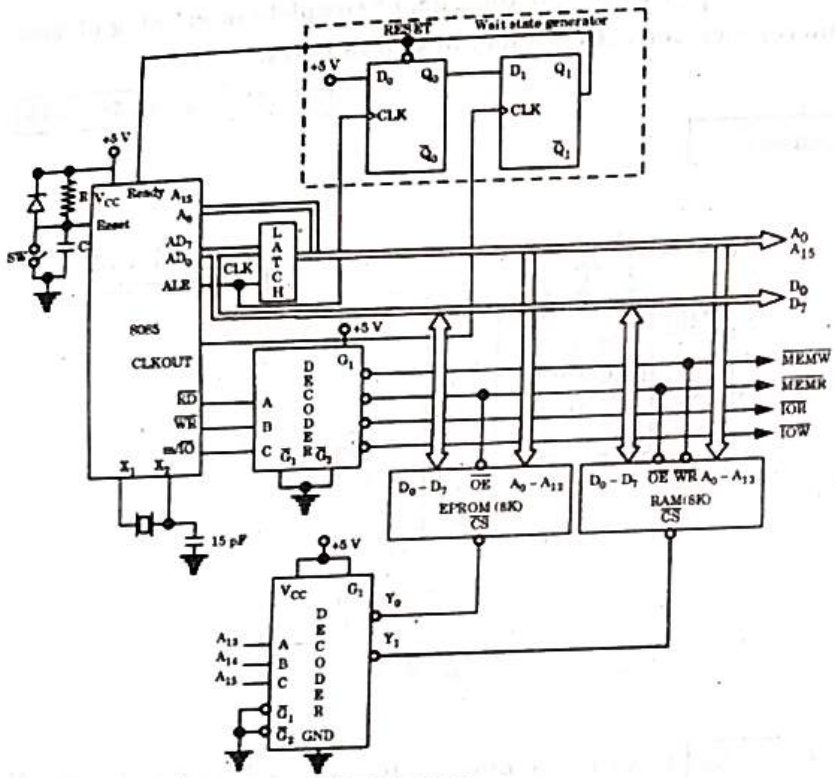


Fig. 1.22.1.

2. The remaining address lines ($A_{13} - A_{15}$) are decoded to generate chip select ($\overline{CS}$) signals. IC 74LS138 is used as decoder.
3. When ($A_{15} - A_{13}$) address lines are zero, the Y_0 output of decoder goes low and selects the EPROM.
4. This means that $A_{15} - A_{13}$ address lines must be zero to read data from EPROM. The address lines $A_0 - A_{12}$ select the particular memory location in the EPROM when $A_{15} - A_{13}$ lines are zero.
5. Similarly when address lines $A_{15} - A_{13}$ are 001, the Y_1 output of decoder goes low and selects the RAM.

Table 1.22.1. Memory map.

Memory ICs	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Address
Starting address of EPROM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
End address of EPROM	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFFH
Starting address of SRAM	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000H
End address of SRAM	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFFH

Que 1.23. **Draw and explain input/output read and input/output write cycle of 8085.**

Answer

Input/output read and input/output write cycle :

1. The I/O read and I/O write machine cycles are similar to the memory read and memory write machine cycles, respectively, except that the IO/$\overline{M}$ signal is high for I/O read and I/O write machine cycles.

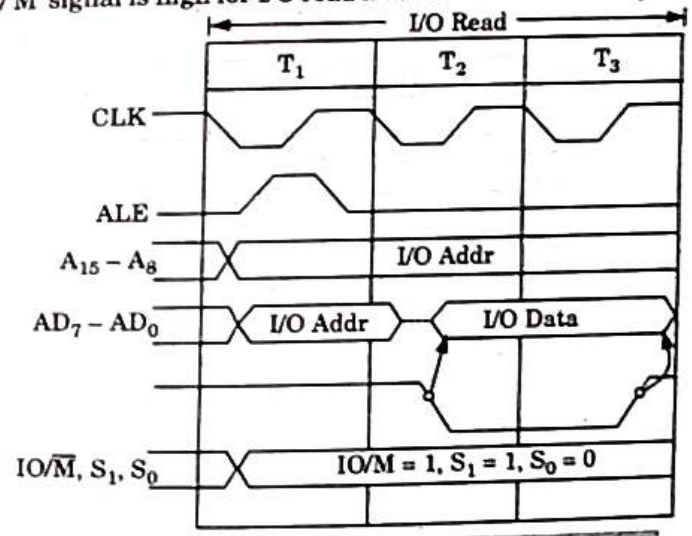


Fig. 1.23.1. I/O Read memory cycle.

2. Here high IO/$\overline{M}$ signal indicate that it is an I/O operation. Fig. 1.23.1 and Fig. 1.23.2 show the timing diagrams for I/O read and I/O write cycles respectively.

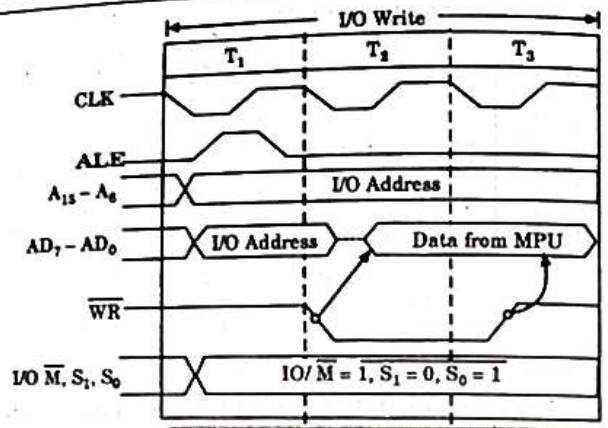


Fig. 1.23.2. I/O Write machine cycle.

Que 1.24. **Draw the timing diagram of the following instruction :**

i. **ADD B**

ii. **CALL 2050 H**

AKTU 2015-16, Marks 10

Answer

i. **ADD B :**

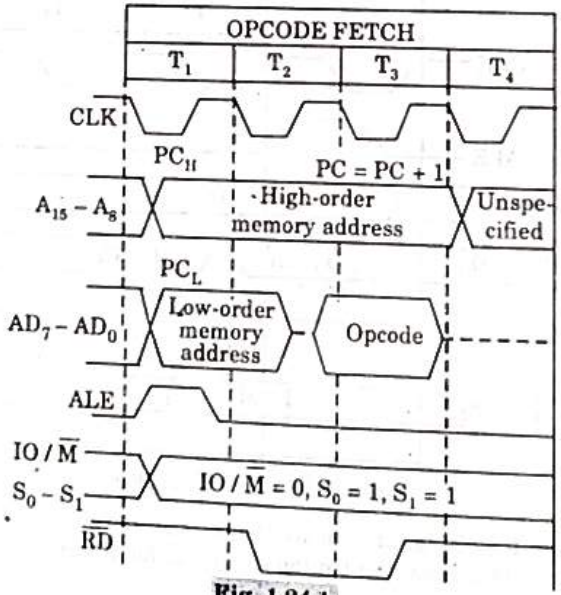


Fig. 1.24.1.

ii. **CALL 2050H :**

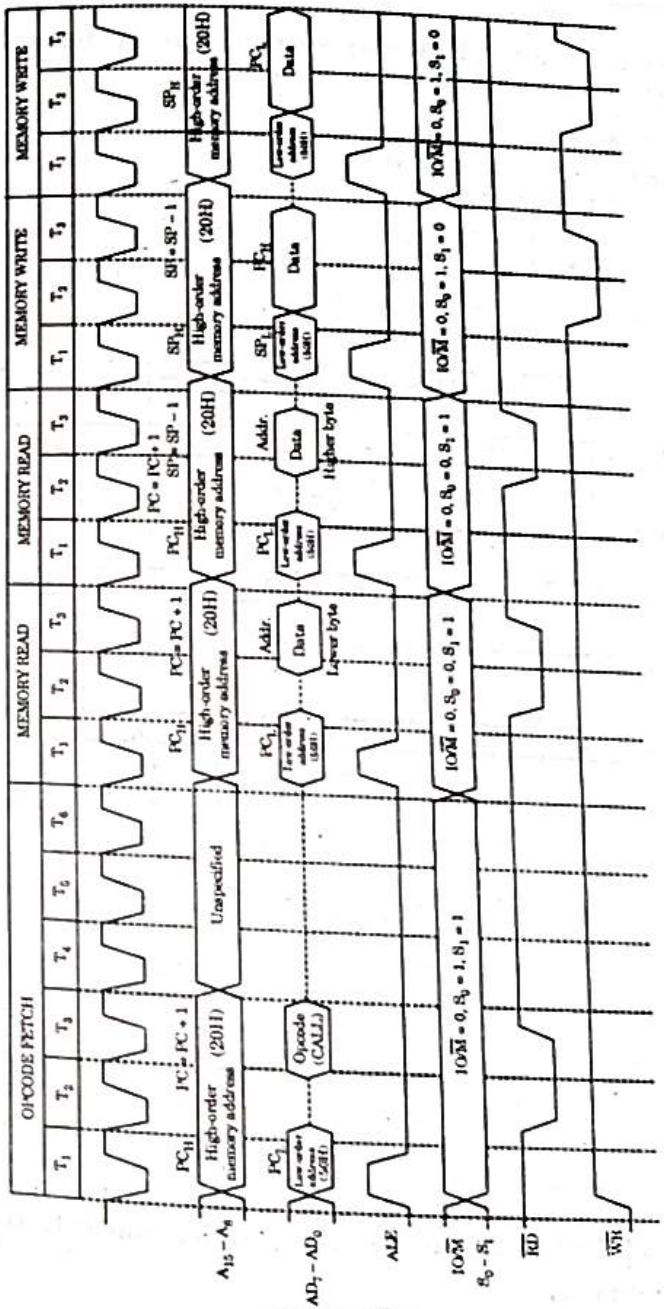


Fig. 1.24.2.

Que 1.25. **Write an instruction to display the content of accumulator to 3500 H memory location. Draw the timing diagram as the instruction is executed.**

Answer

Instruction : STA 3500 H

Timing diagram :

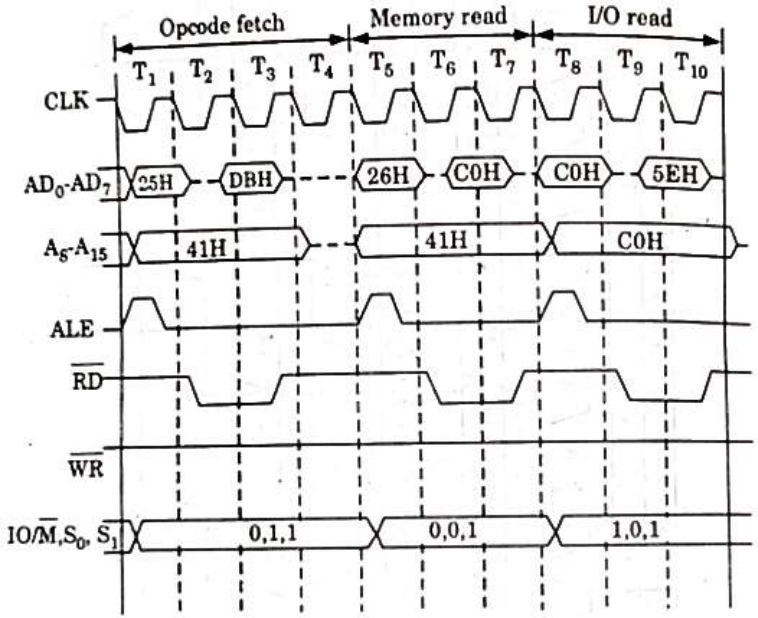


Fig. 1.25.1.

PART-4

Interfacing Output Displays, Interfacing Input Devices, Memory Mapped I/O.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.26. **Explain the interfacing of seven segment LED display as an output device.**

Answer

1. A seven segment display is an output device, consists of one segment of decimal point and seven light emitting diode segment.
2. The physical arrangement of these LEDs is shown in Fig. 1.26.1.
3. The necessary segments are glowed by sending an appropriate signal for current flow through diodes to display a number.
4. Seven segment LED are categorized in two groups : common cathode and common anode.
5. The Seven segment A to G are connected to data lines D_0 to D_6 respectively. DP is connected to data line D_7 if decimal point is used.
6. The binary code required to display a digit is determined by the type of seven segment LED, connection of data lines and logic required to light the segment.

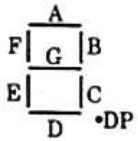


Fig. 1.26.1. Seven-segment LED : LED segments.

7. Consider a seven segment LED that is connected using 3:8 decoder. The circuit diagram is shown in Fig. 1.26.2.
8. Now if we want to display digit 7 on the LED then in common anode seven segment LED, logic 0 is required to be turn ON the segment. The binary code should be equal to 78H.

Data lines	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
Bits	×	1	1	1	1	0	0	0	= 78 H
Segments	NC	G	F	E	D	C	B	A	

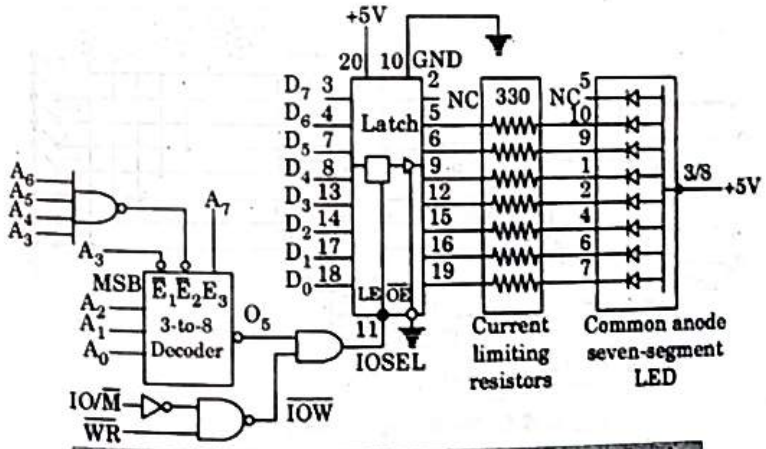


Fig. 1.26.2. Interfacing seven-segment LED using 3:8 decoder.

8. Now if we have to design an output port with port address F5 H, then address line should have following logic.

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
1	1	1	1	0	1	0	1	= F5 H

9. This can be achieved using A_2, A_1 and A_0 as input lines to the decoder. A_1 can be connected to active low enable $\overline{E}_1$ and the remaining address lines can be connected to $\overline{E}_2$ through 4-input NAND gate. Fig. 1.26.2 shows the output port with address F5 H.

Que 1.27. Write a note on interfacing of DIP switches as an input device.

Answer

1. Interfacing of input device is similar to that of output device except that there is difference in bus signals and circuit components.
2. Consider an eight DIP switches interfaced using 3:8 decoder. The circuit used for interfacing eight DIP switches is shown in Fig. 1.27.1.
3. The 3 : 8 decoder is used to decode low order bus and tri-state buffer is used to interface the switches with data bus. The port can be accessed with address 84 H however it has multiple address.
4. The device has two groups of four buffers each and they are controlled by active low signals $\overline{OE}$.
5. When $\overline{OE}$ is low the input data shown up on the output lines and when $\overline{OE}$ signal is high, the output lines assumes high impedance state.

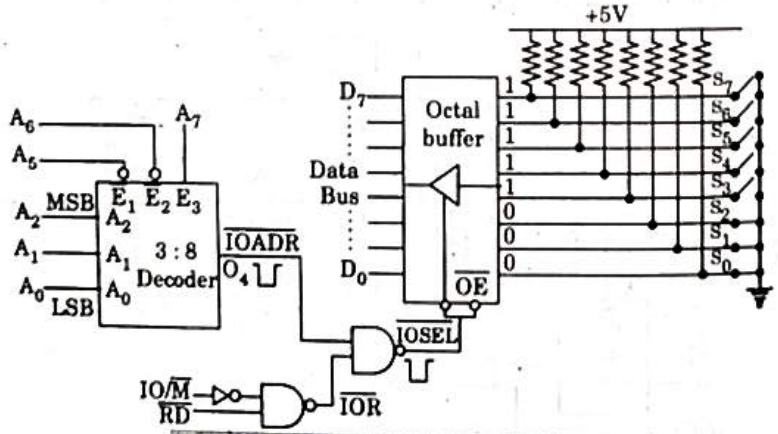


Fig. 1.27.1. Interfacing DIP switches.

6. The low order address bus, except the lines A_4 and A_3, is connected to the decoder. The address lines A_3 and A_4 are left in don't care state.
7. The output line O_4 of the decoder goes low when the address bus has following address

A_7	A_6	A_5	A_4	A_3	A_2	A_0	A_1	
1	0	0	0	0	1	0	0	= 84 H
Enable lines			Don't care		input			

8. The control signal $\overline{I/OR}$ is generated by ANDing the IO/$\overline{M}$ and $\overline{RD}$ in a negative NAND gate, and the I/O select pulse is generated by ANDing the output of decoder and control signal $\overline{IOR}$.
9. When the address is 84 H and control signal $\overline{IOR}$ is asserted, the I/O select the pulse enable tri-state buffer and logic levels of the switches are placed on data bus. The 8085 then begins to read switch position and places the reading in accumulator.

Que 1.28. Explain the memory-mapped I/O with an example.

Answer

1. In memory-mapped I/O, the input and output devices are assigned and identified by 16-bit addresses. To transfer data between the MPU and I/O devices, memory related instructions (such as LDA, STA, etc.) and memory control signals ($\overline{MEMR}$ and $\overline{MEMW}$) are used.

Memory address	Machine code	Mnemonics	Comments
2050	32	STA 8000H	; Store the content of accumulator in memory location 8000H
2051	00		
2052	80		

2. The microprocessor communicates with an I/O device as if it were one of the memory location. For example, the following instruction will transfer the contents of the accumulator to the memory location 8000 *H*.
3. The STA is a 3-byte instruction; the first byte is the opcode, second and third bytes specify the memory address.
4. However, the 16-bit address 8000 H is entered in the reverse order; the low-order byte 00 is stored in location 2051 H. Followed by the high order address 80 H.

5. The instruction LDA (Load accumulator direct) transfers the data from a memory location to the accumulator. The instruction LDA is a 3-byte instruction.

6. To use memory-related instructions for data transfer the control signals Memory read ($\overline{MEMR}$) and Memory write ($\overline{MEMW}$) should be connected to I/O devices instead of IOR and IOW signals and the 16-bit address bus ($A_{15} - A_0$) should be decoded.

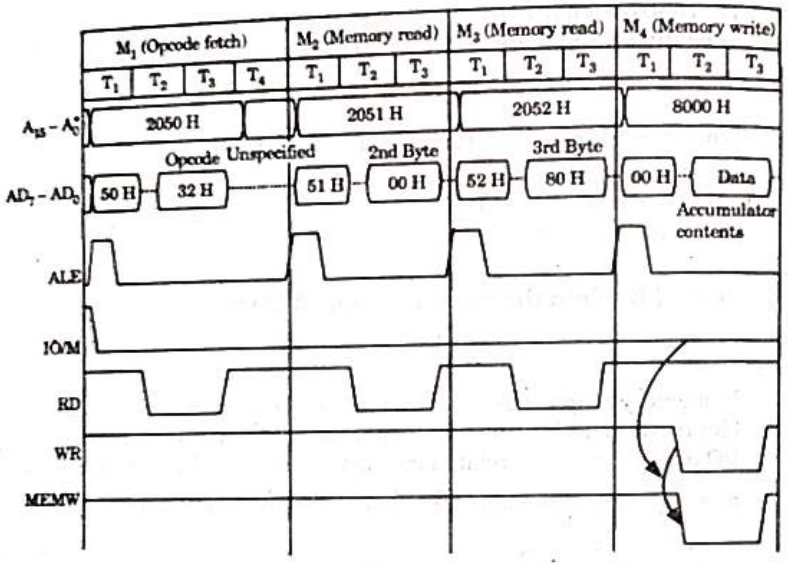


Fig. 1.28.1. Timing of execution of the instruction : STA 8000H.

Device selection and data transfer in memory-mapped I/O require three steps that are similar to those required in peripheral I/O :

i. Decode the address bus to generate the device address pulse.

ii. AND the control signal with the device address pulse to generate the device select (I/O select) pulse.

iii. Use the device select pulse to enable the I/O port.

Que 1.29. Give the comparison between memory-mapped I/O and peripheral I/O.

Answer

S. No.	Memory-mapped I/O	Peripheral I/O
1.	In this device, address is of 16-bit.	In this device, address is of 8-bit.
2.	In this, memory related instructions such as STA, LDA, STAX are used.	In this, OUT and IN instructions are used.
3.	Data transfer takes place between register and I/O device.	Data transfer takes place between I/O and accumulator.
4.	The memory map (64K) is shared between I/Os and system memory.	The I/O map is independent of memory map, 256 input devices and 256 output devices can be connected.
5.	More hardware is needed to decode 16-bit address.	Less hardware is needed to decode 8-bit address.
6.	$\overline{MEMR}$ and $\overline{MEMW}$ control signals are used.	$\overline{IOR}$ and $\overline{IOW}$ control signals are used.
7.	Arithmetic and logical operation can be directly performed with I/O data.	This feature is not available.

VERY IMPORTANT QUESTIONS

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

Q. 1. List the four operations commonly performed by the MPU.

Ans. Refer Q. 1.6.

Q. 2. How many address lines are necessary to address 8 K byte of memory ?

Ans. Refer Q. 1.11.

Q. 3. With a neat diagram describe the internal architecture of 8085. State the function of each block shown.

Ans. Refer Q. 1.14.

Q. 4. Explain the flag register of 8085 microprocessor with the help of example.

Ans. Refer Q. 1.15.

Q. 5. Discuss the various logic devices used in interfacing circuits.

Ans. Refer Q. 1.19.

Q. 6. Draw a neat diagram for interfacing 8K SRAM and 8K EPROM with the system lines of 8085 microprocessor with memory map.

Ans. Refer Q. 1.22.

Q. 7. Draw the timing diagram of the following instruction :

i. ADD B

ii. CALL 2050H

Ans. Refer Q. 1.24.

Q. 8. Give the comparison between memory-mapped I/O and peripheral I/O.

Ans. Refer Q. 1.29.

☺☺☺

t.me/quantum_aktu

aktu quantum

please support
us

- 5 Sem
- 6 Sem
- 7 Sem
- 8 Sem - future support

t.me/quantum_aktu

aktu quantum

please support us

- 5 Sem
- 6 Sem
- 7 Sem
- 8 Sem - future support

2

UNIT

Basic Programming Concepts

CONTENTS

Part-1 : Flow Chart Symbols, Data Transfer Operations, Arithmetic Operations, Logic Operations, Branch Operation ..... **2–2B to 2–9B**

Part-2 : Writing Assembly Language Programs, Programming Techniques : Looping, Counting and Indexing ..... **2–9B to 2–12B**

Part-3 : Additional Data Transfer and 16-bit Arithmetic Instruction, Logic Operation : Rotate, Compare ..... **2–12B to 2–17B**

Part-4 : Counters and Time Delays, 8085 Interrupts ..... **2–17B to 2–25B**

PART-1

Flow Chart Symbols, Data Transfer Operations, Arithmetic Operations, Logic Operations, Branch Operation.

CONCEPT OUTLINE

- A flow chart is a diagrammatic representation of the procedure for solving the problem.
- The data transfer instructions copy data from a source into a destination without modifying the contents of the source.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.1. **What do you mean by flowchart ?**

Answer

A. Flowchart :

1. A flowchart 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 problems.
3. Once developed and thoroughly checked, the flowchart provides an excellence guide for writing a program.
4. It indicates the direction of flow of a process, relevant operation and computation, point of decision and other information which is part of solution.

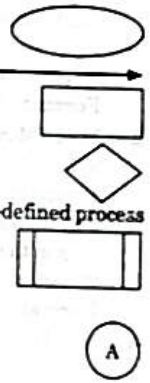
B. Steps to develop a flow chart :

1. Gather information of how the process flows by the use of :
   a. Conservation
   b. Experience
   c. Product development codes
2. Trail process flow.
3. Allow other more familiar personnel to check for accuracy.
4. Make changes if necessary.
5. Compare final actual flow with best possible flow.

C. The graphic symbols used in the flow chart :

1. **Oval :** It indicates start or stop operation.
2. **Arrow :** It indicates flow with direction.
3. **Rectangle :** It indicates process or operation.
4. **Diamond :** It indicates decision making operation.
5. **Double sided rectangle :** It indicates execution of pre-defined process (subroutine).
6. **Circle with alphabet :** It indicates continuation.

   A : Any alphabet



Que 2.2. **Draw a flow chart to determine maximum of three numbers.**

Answer

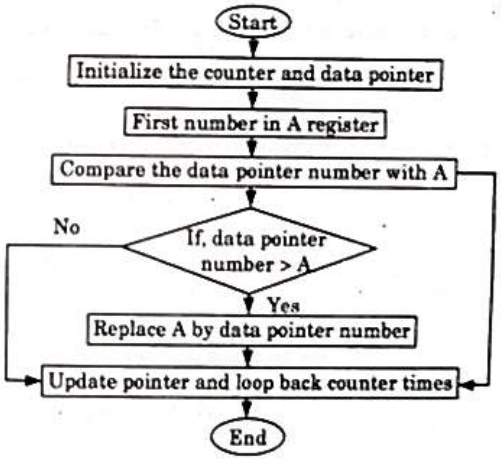


Fig. 2.2.1.

Que 2.3. **What do you mean by data transfer instructions ?**

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. Intructions :

1. **MOV (Move) :** This instruction copies the content of source register to the destination register. If one of the operand is a memory location, it is specified by contents of HL 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. The register pair is used as a memory pointer.

   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. **SPHL (Copy HL registers to stack pointer) :** This instruction loads the contents of register *H* and *L* into stack pointer. The content of *H* register provides the higher order address and content of *L* register provides lower order address.

   Format : SPHL

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

    Format : STAX R_p

Que 2.4. **What are the various instructions used for arithmetic operation in the 8085 microprocessors.**

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.

   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 2.5. **Discuss logical operation in detail.**

Answer

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

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

Que 2.6. **What do you understand by branching instruction ? Explain the various jump instruction.**

Answer

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

i. **JMP (Jump unconditionally) :**

1. The jump instructions specify the memory location explicitly.
2. The program sequence is transferred to memory location specified by the 16-bit address.
3. This is 3-byte instruction : First byte for operation code and the second and third byte specific the address. The unconditionally jump instruction enables the programmer to set up continuous loop.

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

ii. **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

iii. **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

iv. **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.

v. **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 = 0000 H
RST 1	1 × 8 = 0008 H
RST 2	2 × 8 = 0010 H
RST 3	3 × 8 = 0018 H
RST 4	4 × 8 = 0020 H
RST 5	5 × 8 = 0028 H
RST 6	6 × 8 = 0030 H
RST 7	7 × 8 = 0038 H

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.

PART-2

Writing Assembly Language Programs, Programming Techniques : Looping, Counting and Indexing.

CONCEPT OUTLINE

- Looping is a programming technique used to instruct microprocessor to repeat task.
- Indexing means pointing or referencing objects with sequential number.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.7. **Explain the various steps involved in writing the assembly language programs.**

Answer

1. A program is a set of instructions arranged in the specific sequence to do the specific task.
2. It tells the microprocessor what it has to do.
3. The process of writing the set of instructions which tells the microprocessor what to do is called programming.

Steps involved in programming :

i. **Specifying the problem :** The first step in the programming is to find out which task is to be performed. This is called specifying the problem. If the programmer does not understand what is to be done, the programming process cannot begin.

ii. **Designing the problem-solution :** During this process, the exact step by step process that is to be followed, is developed and written down.

iii. **Coding :**

1. Once the program is specified and designed, it can be implemented. Implementation begins with the process of coding the program. Coding the program means to tell the processor the exact step by step process in its language.
2. Each processor has a set of instructions. Programmer has to choose appropriate instructions from the instruction set to build the program.

iv. **Debugging :** Once the program or a part of program is coded, the next step is debugging the code. Debugging is the process of testing the code to see if it does the given task. If program is not working properly, debugging process helps in finding and correcting errors.

Que 2.8. **What are the various types of instructions used in assembly language programming ? Explain one of them in detail.**

AKTU 2015-16, Marks 10

Answer

1. **Data transfer instruction related to microprocessor register and I/O :** Refer Q. 2.3, Page 2–3B, Unit-2.
2. **Logical instructions related to rotating the accumulator bits :** Refer Q. 2.5, Page 2–6B, Unit-2.
3. **16-bit arithmetic instructions :**

a. **ADD *M* (Add memory data to the accumulator) :** This instruction add the contents of memory location specified by the HL register pair with accumulator and the result are stored in accumulator.
Format : ADD *M*

b. **SUB *M* (Subtract memory content from accumulator) :** This instruction subtracts the contents of memory location specified by register pair HL and stores the result into accumulator.
Format : SUB *M*

c. **INR *M* (Increment contents of memory by 1) :** This instruction increment the contents of memory location by 1. The memory location is specified by register pair HL. After execution of this instructions all flags are affected except carry flag.
Format : INR *M*

d. **DCR *M* (Decrement contents of memory by 1) :** This instruction decrease the contents of memory location specified by register pair HL by 1.
Format : DCR *M*

Que 2.9. **Explain the various looping techniques used in programming.**

Answer

1. Looping is a programming technique used to instruct microprocessor to repeat tasks.
2. A loop is set up by instructing the microprocessor to change the sequence of execution and perform the task again.
3. This process is done using jump instruction. Loops are classified into two groups :

i. **Continuous loop :** Unconditional jump instructions are used to set up these loop. A program with this type of loop does not stop until the system is reset.

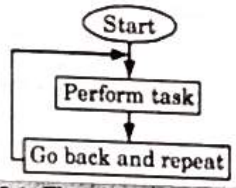


Fig. 2.9.1. Flowchart of continuous loop.

ii. **Conditional loop :** These loops are set up by using conditional jump instructions. These instructions check flags (carry, zero etc) and repeat the specified task if the conditions are satisfied. These loops usually include indexing and counting.

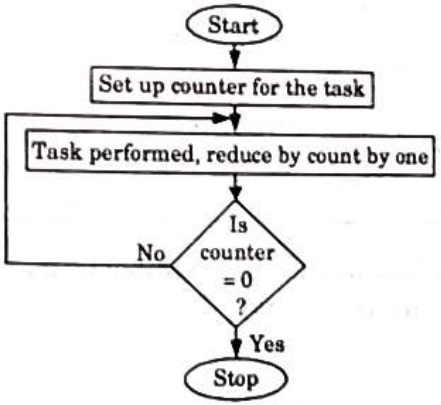


Fig. 2.9.2.

4. **Counter :** A counter is a application of conditions loop. If the microprocessor has to perform a certain repeated task for fix number of times then task is performed using counter and conditional loop. Using conditional loop the task is performed repeatedly and by using the counter number of times, the task is to be performed is counted.

5. **Conditional loop using counting and indexing :**

i. Indexing means pointing or referencing objects with sequential number.

ii. For example, in library, books are arranged according to numbers and they are sorted according to numbers. This is known as indexing, similarly data bytes are stored in memory location and these data bytes are referred to by their memory location.

Que 2.10. **Write an 8085 assembly language program to determine the 2's complement of an 8-bit number without using any logical instruction. Store the result in memory.**

Answer

Let an 8-bit number is stored in memory at 2200 H.

```
LDA 2200 H  ; Get the number
MOV B, A    ; Move content of register A to register B
MVI A, FF H ; Load register A with the data FF H
SUB B       ; Subtract content of register B from content of register A
            (we obtained 1's complement of the number stored in
            register A)
ADI, 01 H   ; Add 1 in the number
STA 2300 H  ; Store the result
HLT         ; Terminate program execution.
```

PART-3

Additional Data Transfer and 16-bit Arithmetic Instruction, Logic Operation : Rotate, Compare.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.11. **Explain various data transfer instruction related to microprocessor register and I/O.**

Answer

A. **16-bit data transfer to register pair (LXI) :**

1. **LXI R_p, 16-bit :** This instruction loads 16-bit data in the register pair. Register pair can be BC, DE and HL. It is a 3-byte instruction.

   Example : LXI D, 2070 H

2. **LXI SP, 16-bit :** This instruction loads 16-bit data in the stack pointer register.

   Example : LXI SP, 0317 H

B. **Data transfer from memory to the microprocessor :**

1. **MOV *R, M* :** This instruction copies the data byte from memory location into a register. It is a 1-byte instruction. *R* can be *A, B, C, D, E, H* and *L* and *M* is specified by the content of HL register.

2. **LDAX B/D :** It is a 1-byte instruction which copies the data byte from memory location into the accumulator. Memory location can be specified by the content of registers BC or DE.

3. **LDA 16-bit :** It is a 3-byte instruction which copies the data of given memory location into accumulator.

   Example : LDA 3757 H

C. **Data transfer from microprocessor to memory or directly into memory :**

1. **MOV *M, R* :** This is a 1-byte instruction that copies data from a register *R*, into the memory location specified by the content of HL registers.

2. **STAX B/D :** The contents of accumulator are copied into memory location specified by the register pairs. It is a 1-byte instruction.

3. **STA 16-bit :** It is a 3-byte instruction that copies data from accumulator into the memory location specified by the 16-byte operand.

4. **MVI *M*, 8-bit :** It is a 2-byte instruction that copies 8-bit data into memory location specified by HL register pair from the accumulator.

Que 2.12. **Explain various 16-bit arithmetic instructions.**

Answer

These are the instructions which perform arithmetic task related to memory such as ADD *M*, INR *M* etc.

i. **ADD *M* (Add memory data to the accumulator) :** This instruction add the contents of memory location specified by the HL register pair with accumulator and the result are stored in accumulator.

   Format : ADD *M*

ii. **SUB *M* (Subtract memory content from accumulator) :** This instruction subtracts the contents of memory location specified by register pair HL and stores the result into accumulator.

   Format : SUB *M*

iii. **INR *M* (Increment contents of memory by 1) :** This instruction increment the contents of memory location by 1. The memory location is specified by register pair HL. After execution of this instructions all flags are affected except carry flag.

Format : INR *M*

iv. **DCR *M* (Decrement contents of memory by 1) :** This instruction decrease the contents of memory location specified by register pair HL by 1.

Format : DCR *M*

v. **INX R_P :** It is a 1-byte instruction. It treats the contents of two register as one 16-bit number and increases the contents by 1.

R_P can be BC, DE, HL and stack pointer.

Format : INX R_P

vi. **DCX R_P :** It is a 1-byte instruction. It decreases the 16-bit contents of a register pair by 1. R_P can be *B*, *D*, *H* and SP (stack pointer).

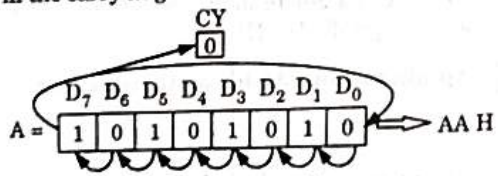
Format : DCX R_P

Que 2.13. Explain the various logical instructions related to rotating the accumulator bits.

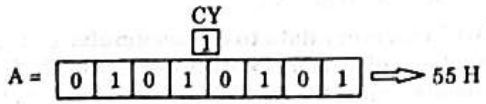
Answer

Rotation instructions are primarily used in arithmetic, multiply and divide operation. These operations include instructions such as RAR, RRC, RAL and RLC.

i. **RLC (Rotate accumulator left) :** This instruction rotates the contents of accumulator left by one position. Bit D_7 is placed in the position of D_0 as well as in the carry flag.



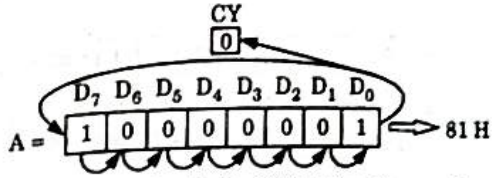
(*a*) Before the instruction.



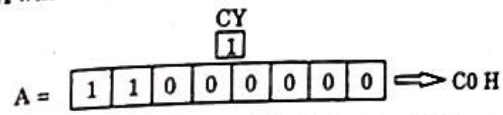
(*b*) After the RLC, (A) will be 55 H with CY set.

Fig. 2.13.1. Accumulator contents after RLC.

ii. **RRC (Rotates accumulator right) :** This instruction rotates the contents of accumulator right by one position. Bit D_0 is placed in position of D_7 as well as in carry.



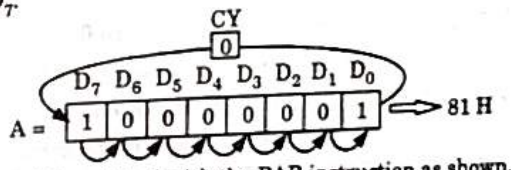
(*a*) A will be rotated with the RRC instruction as shown.



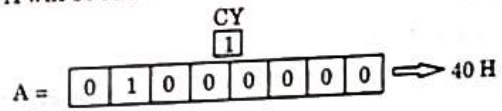
(*b*) After the execution of the instruction RRC, A will be C0 H with the CY flag set.

Fig. 2.13.2. Rotate right instructions.

iii. **RAR (Rotate accumulator right through carry) :** This instruction rotates the contents of the accumulator right by one position. Bit D_0 is placed in carry flag and the bit in carry flag is placed in most significant position D_7.



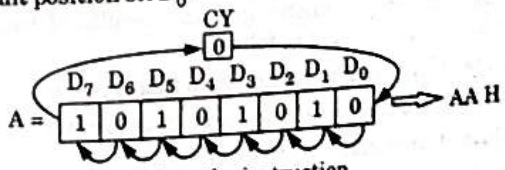
(*a*) A will be rotated with the RAR instruction as shown.



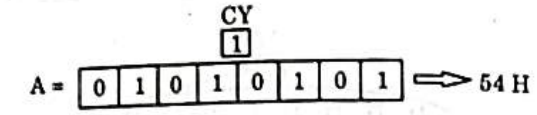
(*b*) After the execution of the RAR instruction, A will be 40 H with the CY flag set.

Fig. 2.13.3. Rotate right instructions.

iv. **RAL (Rotate accumulator left through carry) :** This instruction rotates the contents of the accumulator left by one position through carry flag. Bit D_7 is placed in carry flag and carry flag is placed in least significant position bit D_0.



(*a*) Before the instruction.



(b) After the instruction RAL, (A) will be 54 H with CY set.

Fig. 2.13.4. Accumulator contents after RLC.

Que 2.14. Explain the compare operations of microprocessor 8085.

Answer

The microprocessor 8085 includes two instructions CMP and CPI. These two instructions are used to compare data byte with contents of accumulator by subtracting the data byte or content of memory from accumulator and indicating the result by modifying the flags. However contents are not modified.

i. **CMP R/M (Compare with accumulator) :** This instruction compares the contents of specified register or memory location with the contents of accumulator and both the contents are preserved and comparison is shown by setting the flags.

If (A) < (Reg/Mem) : Carry flag is set and zero flag is reset.

If (A) = (Reg/Mem) : Zero flag is set and carry flag is reset.

If (A) > (Reg/Mem) : Carry and zero flags are reset.

Example : Register B contains data byte 62H and accumulator contains data byte 57H. Compare the contents of register B with those of accumulator.

Instruction: CMP B

Before instruction				After instruction			
A	57	××	Z	A	57	0	Z
B	62	××	C	B	62	1	C

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

Fig. 3.14.1.

ii. **CPI 8-bit (Compare immediate with accumulator) :** The specified byte is compared with the contents of the accumulator. The values being compared remain unchanged and results of comparison are indicated by setting the flag as follows :

If (A) < Data : Carry flag is set and zero flag is reset.

If (A) = Data : Zero flag is set and carry flags is reset.

If (A) > Data : Carry and zero flags are reset.

Example : Assume the accumulator contains BAH. Compare 30H with accumulator contents.

Instruction : CPI 30H

Result after executing instruction :

The accumulator contents remain same.

Z and CY are reset because (A) > Data

Other flags : S = 0, AC = 0, P = 0

Que 2.15. Sixteen bytes of data are stored in memory location at 205F H. Write a programme transfer the entire block of data to new memory locations starting at 2070 H. **AKTU 2016-17, Marks 05**

Answer

```
        LXI H, 2050 H  ; Set up HL as a pointer for the source memory.
        LXI D, 2070 H  ; Set up DE as a pointer for the destination memory.
        MVI B, 10 H    ; Set up B as byte counter
NEXT :  MOV A, M       ; Get data byte from the source memory
        STAX D         ; Store the data byte in destination memory
        INX H
        INX D          ; Get ready to transfer next byte
        DCR B
        JNZ NEXT       ; Go back to get next byte if byte counter ≠ 0
        HLT            ; End of program
```

PART-4

Counters and Time Delays, 8085 Interrupts.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.16. Write a short note on counter and time delay operation in microprocessor.

Answer

A. **Counters :**

1. Counters are mainly used to keep track of events.
2. It can be designed by loading an appropriate number into any one register of microprocessor and then using INR or DCR instructions.

3. Whenever a loop is established using a register, end count is updated then every time count is checked to determine whether it has reached the required number and if not then loop is repeated again.
4. Fig. 2.16.1 shows the flow chart of counter operation.

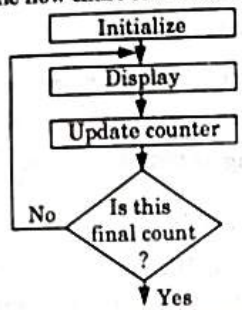


Fig. 2.16.1. Flow chart of a counter.

B. Time delay :

1. This is used to perform 'NOP' means no operation as real time applications such as traffic light, digital clock etc., requires some delay during their application.
2. Time delay can be generated using execution of instruction several times.
3. The flowchart in Fig. 2.16.2 shows the generation of time delay.

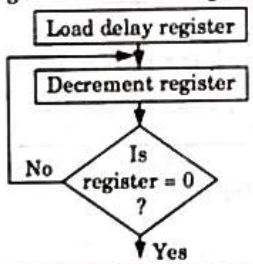


Fig. 2.16.2. Time delay loop flow dingram.

Que 2.17. Write different procedures used to generate specific delay.

Answer

A. Timer delay using one register :

1. A count is loaded in a register and loop is executed until the count reaches zero.
2. The instruction is as follow :

			Number of T-states
	MVI *C*, count	; Load count	7 T-states
BACK :	DCR *C*	; Decrement count	4 T-states
	JNZ BACK	; If count ≠ 0, repeat	10/7 T-states

3. The instructions DCR *C* and JNZ BACK execute number of times equal to count stored in the *C* register.
4. The time taken by this program for execution can be calculated with the help of T-states.
5. There are count – 1 passes through the loop where the condition is met and control is transferred back to the first instruction in the loop (DCR *C*).

∴ Total T-states required to execute the given program

$$= \underset{\text{MVI C}}{7} + \underset{\text{Loops}}{(\text{count} - 1) \times (4 + 10)} + \underset{\text{Last Loop}}{(4 + 7)}$$

For count = 5

Number of T-state = 7 + (5 – 1) × (14) + (11) = 7 + 56 + 11 = 74

Assuming operating frequency of 8085A is 2 MHz,

$$\text{Time required for 1 T-state} = \frac{1}{2\text{ MHz}} = 0.5\ \mu\text{sec.}$$

6. Total time required to execute the program = 74 × 0.5 μsec, = 37 μsec

The maximum count that can be loaded in the 8-bit register is FF H (255) so the maximum delay possible with 8-bit count, assuming operating frequency 2 MHz.

$$= (7 + (255 - 1) \times (14) + (11)) \times 0.5\ \mu\text{sec.}$$

$$= 1787\ \mu\text{sec.}$$

B. Time delay using a register pair :

			Number of T-states
	LXI *B*, count	; Load 16 bit count	10 T-states
BACK :	DCX *B*	; Decrement count	6 T-states
	MOV *A*, *C*	;	4 T-states
	ORA *B*	; Logically OR B and C	4 T-states
	JNZ BACK	; If result is not 0, repeat	10 T-states

1. The instructions DCX *B*, MOV *A*, *C*; ORA *B* and JNZ BACK execute number of times equal to count stored in BC register pair.
2. The instruction LXI *B*, count; is executed only once. It requires 10 T-states. The number of T-states requited for one loop

$$= 6 + 4 + 4 + 10 + = 24 \text{ T-states.}$$

3. The number of T-states required for last loop = 6 + 4 + 4 + 7 = 21 T-states. So total T-states required for execution of given program are

$$= \underset{\text{LXI B}}{10} + \underset{\text{Loops}}{(\text{count} - 1)} \times \underset{\text{Last Loop}}{24 + 21}$$

For count = 03FF H (1023_{10})

Number of T-state = 10 + (1022) × 24 + 21

= 24559

4. Assuming operating frequency of 8085A as 2 MHz, the time required for one T-state = 0.5 μsec.

∴ Total time required to execute the given program

= 24559 × 0.5 μsec = 12279.5 μsec

= 12.2795 msec

C. Timer delay using nested loops :

1. In this, there is more than one loop. The outer loop sets the multiplying count to the delays provided by the innermost loop.

			Number of T-states
	MVI *B*, Multiplier count	; Initialize multiplier	7 T-states
START :	MVI *C*, Delay count	; Initialize delay count	7 T-states
BACK :	DCR *C*	; Decrement delay count	4 T-states
	JNZ BACK	; If not 0, repeat	10/7 T-states
	DCR *B*	; Decrement multiplier count	4 T-states
	JNZ START	; If not 0, repeat	10/7 T-states

2. T-states required for execution of inner loop

$$T_{inner} = 7 + (\text{Delay count} - 1) \times 14 + 11$$

3. T-states required for execution of the given program

$$= (\text{Multiplier count} - 1) \times (T_{inner} + 14) + 11$$

For delay count = 65 H (101) and multiplier count = 51 H (81)

$$T_{inner} = 7 + (101 - 1) \times 14 + 11$$

$$= 1418$$

4. Total time required to execute the given program is (Operating frequency is 2 MHz) = [(81 − 1) × (1418 + 14) + 11] × 0.5 μsec.

= 57.2855 msec.

Que 2.18. Write a delay routine to produce a time delay of 0.5 msec in 8085 processor-based system whose clock source is 6 MHz quartz crystal. **AKTU 2015-16, Marks 10**

Answer

1. In 8085, the operating frequency is half of the crystal frequency.
2. Operating frequency = 6/2 = 3 MHz
3. Time for one T-state = $\frac{1}{3\text{ MHz}}$ = 0.33 μsec
4. Number of T-states required

$$= \frac{\text{Required time}}{\text{Time for one T-state}} = \frac{0.5\text{ msec}}{0.33\ \mu\text{sec}}$$

= 1500

5. **Delay routine :**

```
        LXI B, Count   ; 16-bit count
BACK :  DCX B          ; Decrement count
        MOV A, C
        ORA B          ; Logically OR B and C
        JNZ BACK       ; If result is not zero repeat.
```

6. **Count calculation :**

$$1500 = 10 + (\text{count} - 1) \times 24 + 21$$

$$\text{Count} = \left(\frac{1500 - 31}{24} + 1\right) \approx (62)_{10}$$

Count = 3E H

Que 2.19. If the clock frequency is 5 MHz, how much time is required to execute an instruction of 18-T states ? **AKTU 2014-15, Marks 05**

Answer

1. Clock frequency = 5 MHz = 5×10^6 Hz
2. Time, $T = \frac{1}{\text{frequency}} = \frac{1}{5 \times 10^6} = 0.2 \times 10^{-6} = 0.2\ \mu s$
3. Time required to execute an instruction of 18 T-states = 0.2 × 18 = 3.6 μs

Que 2.20. Write a note on debugging techniques.

Answer

The debugging techniques are used to check the errors in a counter program. Various errors which may occur in the programs are :

1. Errors in counting T-states in a delay loop. Most of the times, first instruction to load a delay register is included in loop by mistake.
2. Errors in calculating how many times a loop is repeated.
3. Failure to convert a delay count from a decimal number into its equivalent hexadecimal number.

4. Error in converting a delay count from a decimal number to its hexadecimal equivalent or vice-versa.
5. Specifying a wrong jump location.
6. Failure to set a flag, especially with 16-bit decrement / increment instruction.
7. Failure to display either the first or last count.
8. Failure to provide a delay between the last and the last-but-one count.

Que 2.21. Write a note on 8085 interrupt system.

Answer

1. In 8085, interrupt enabled flip-flop controls the interrupt process.
2. This flip-flop can be set or reset using software instruction. If the flip-flop is enabled and the input to the interrupt signal (INTR) goes high, the microprocessor is interrupted.
3. This is maskable interrupt and can be disabled using software instruction.
4. There are eight necessary steps required to describe interrupt process of microprocessor 8085.

i. Using interrupt Enable Instruction (EI), interrupt process can be used.

ii. During execution of program, microprocessor checks INTR line during execution of each instruction.

iii. If the INTR line is high and interrupt is enabled, the microprocessor completes the execution of current instruction and disable the interrupt enable flip-flop and send interrupt acknowledge signal ($\overline{\text{INTA}}$). Until the interrupt flip-flop is enabled again, the processor cannot accept any interrupt request.

iv. This $\overline{\text{INTA}}$ signal is used to insert a restart (RST) or a CALL instruction through external hardware. This RST instruction transfers the program control to a specific memory location on page 00 H and start the execution at that memory location after executing step (v).

v. Whenever RST or CALL instruction is received by the microprocessor, it saves the memory address of next instruction on the stack. Then, the program is transferred to the CALL instruction.

vi. The task to be performed is written as a subroutine at the specified location. The microprocessor performs the task. This subroutine is called as service routine.

vii. The service subroutine should include EI to enable the interrupt again.

viii. At the end of subroutine, the RET instruction retrieve the memory address, where the program was interrupted and continues the execution.

Que 2.22. Define interrupt. Explain the hardware interrupts of microprocessor 8085.

Answer

A. **Interrupt :** The process by which the normal working of 8085 processor can be interrupted so as to provide service to I/O or perform another job, is called as an interrupt.

B. **Hardware Interrupts :**

1. The 8085 has five hardware interrupts, among them four are maskable (RST 7.5, RST 6.5, RST 5.5 and INTR) and one is non-maskable (TRAP).
2. When any of these pins, except INTR, is active, the internal control circuit of the 8085 produces a CALL to a predetermined memory location.

i. **TRAP :** It is non-maskable interrupt. It can't be disabled by instruction. In order to service this interrupt the signal on TRAP pin must have high level. The 8085 completes execution of current instruction, pushes the program counter on the stack and branches to location 0024 H (Interrupt address vector for TRAP).

ii. **RST 7.5 :** It is maskable interrupt. It can be enabled and disabled using SIM (Set Interrupt Mask) instruction. 8085 respond to RST 7.5 interrupt when signal on pin has leading edge. In order to service RST 7.5, 8085 complete current execution of instruction, pushes the program counter on the stack and branches to 003C H.

iii. **RST 6.5 :** This can be enabled and disabled using SIM instruction. RST 6.5 is high level sensitive, microprocessor 8085 execute current instruction save the program counter on to the stack and branches to location 0034 H.

iv. **RST 5.5 :** It is maskable interrupt and can be enabled and disabled by SIM instruction. It is high level sensitive. In order to service this interrupt, 8085 completes the execution of current instruction, save the program counter into the stack and branches to location 002C H.

v. **INTR :**

1. It is a maskable interrupt. Whenever INTR signal is high then 8085 completes its current instruction and send active low interrupt acknowledge signal $\overline{\text{INTA}}$ if interrupt is enabled.
2. In response to $\overline{\text{INTA}}$ signal, external logic places an instruction opcode on data bus. On receiving the instruction 8085 save the address of next instruction on stack and execute received instruction.

Interrupts	Call locations
1. TRAP →	0024 H
2. RST 7.5 →	003C H
3. RST 6.5 →	0034 H
4. RST 5.5 →	002C H

The TRAP has the highest priority, followed by RST 7.5, 6.5, 5.5, and INTR. However, the TRAP has a lower priority than the Hold signal used for DMA.

Que 2.23. **What is meant by the software interrupts and why they are used ? List out all the software interrupts of 8085 and give their vector addresses.**

OR

Explain the role of interrupts in programming. Explain the interrupt used in 8085. List out all the vectored interrupts of 8085 and give their vector address. **AKTU 2016-17, Marks 7.5**

OR

Explain the interrupts used in 8085 briefly. **AKTU 2015-16, Marks 05**

Answer

A. **Interrupt :** Refer Q. 2.21, Page 2–22B, Unit-2.

B. **Role of interrupts :** Refer Q. 2.21, Page 2–22B, Unit-2.

C. **Interrupt in 8085 :**

a. **Hardware interrupts :** Refer Q. 2.22, Page 2–23B, Unit-2.

b. **Software interrupts :**

1. Software interrupt includes RST 0 to RST 7. These are restart interrupts.
2. In software interrupts the cause of the interrupt is an execution of the instruction. These are special instructions supported by the microprocessor.
3. 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.
4. Upon completion of the execution of the subroutine program, program control returns to the main program.
5. The 8085 has eight software interrupts from RST 0 to RST 7. The vector address can be calculated as :

   Interrupt number × 8 = Vector address

Instruction	Vector address
RST 0	0000 H
RST 1	0008 H
RST 2	0010 H
RST 3	0018 H
RST 4	0020 H
RST 5	0028 H
RST 6	0030 H
RST 7	0038 H

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 mean by data transfer instructions ?

Ans. Refer Q. 2.3.

Q. 2. What do you understand by branching instruction ? Explain the various jump instruction.

Ans. Refer Q. 2.6.

Q. 3. What are the various types of instructions used in assembly language programming ? Explain one of them in detail.

Ans. Refer Q. 2.8.

Q. 4. Explain various 16-bit arithmetic instructions.

Ans. Refer Q. 2.12.

Q. 5. Sixteen bytes of data are stored in memory location at 205F H. Write a programme to transfer the entire block of data to new memory locations starting at 2070 H.

Ans. Refer Q. 2.15.

Q. 6. Write a delay routine to produce a time delay of 0.5 msec in 8085 processor-based system whose clock source is 6 MHz quartz crystal.

Ans. Refer Q. 2.18.

Q. 7. If the clock frequency is 5 MHz, how much time is required to execute an instruction of 18-T states ?

Ans. Refer Q. 2.19.

Q. 8. What is meant by the software interrupts and why they are used ? List out all the software interrupts of 8085 and give their vector addresses.

Ans. Refer Q. 2.23.

☺☺☺

t.me/quantum_aktu

aktu quantum

please support us

- 5 Sem
- 6 Sem
- 7 Sem
- 8 Sem - future support

3 UNIT

16-bit Microprocessor

CONTENTS

Part-1 : 16-bit Microprocessors (8086) : ..... **3–2B to 3–13B**
Architecture, Pin Description,
Physical Address, Segmentation,
Memory Organization,
Addressing Modes

Part-2 : Peripheral Devices : ..... **3–13B to 3–24B**
8237 DMA Controller,
8255 Programmable
Peripheral Interface

Part-3 : 8253/8254 Programmable ..... **3–25B to 3–32B**
Timer/Counter, 8259
Programmable Interrupt Controller

Part-4 : 8251 USART and RS232C ..... **3–32B to 3–38B**

PART-1

16-bit Microprocessors (8086) : Architecture, Pin Description, Physical Address, Segmentation, Memory Organization, Addressing Modes.

CONCEPT OUTLINE

- 8086 is a 16-bit microprocessor and most of its instructions are designed to work with 16-bit binary words.
- It provides fourteen 16-bit register.
- It supports the pipelining.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

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

Answer

A. Features of 8086 :

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 memory 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} 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.
6. It is possible to perform bit, byte, word and block operations in 8086.
7. The 8086 is designed to operate in two modes namely the minimum mode and the maximum mode.
8. 8086 supports multiprogramming and pipelining.

B. Differences between 8085 and 8086 :

S. No.	8085	8086
1.	8085 is a 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 PIN diagram of 8086 microprocessor and discuss about each pin.** **AKTU 2016-17, Marks 10**

Answer

The functional signal description of pin diagram of microprocessor 8086 is shown in Fig. 3.2.1.

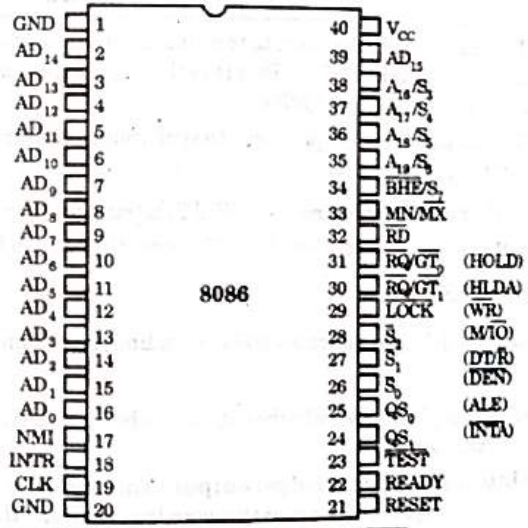


Fig. 3.2.1. Pin configuration of 8086.

1. **$AD_{15} - AD_0$:** These are time multiplexed memory I/O address and data lines which act as address but during first part of machine cycle and data bus for remaining part of the machine cycle.

2. $A_{19}/S_6 - A_{16}/S_3$: These are used to output upper 4-bits of address during first part of machine cycle. During remaining part of the machine cycle these are used to output status, which indicates the type of operation to be performed.

3. $\overline{BHE}/S_7$: **(Bus High Enable)/Status :** Low signal on this pin during first part of machine cycle indicates that at least one of the current transfer is to be made on higher order byte $AD_{15} - AD_8$, otherwise transfer is made on low order byte $AD_7 - AD_0$.

   Status S_7 is output during later part of machine cycle.

4. **NMI :** It is positive edge triggered non-maskable interrupt.

5. **INTR :** It is maskable level triggered input. It is sampled during the last clock cycle of each instruction to determine if processor should enter into interrupt service routine.

6. **CLK :** It provides basic timing for processor operation and bus control activity clock frequency depends on version of 8086.

Processor 8086	Required clock signal
8086	5 MHz
8086-2	8 MHz
8086-1	10 MHz

7. **RESET :** It causes the processor to terminate the current activity and start execution from FFFF0 H. This is active high signal and must be active for at least four clock cycles.

8. **READY :** This signal is used primarily to synchronize slower peripherals with the microprocessor.

9. $\overline{TEST}$: This input is examined by a WAIT instruction. The 8086 enters into a wait state after execution of WAIT instruction until a low signal is applied on $\overline{TEST}$ pin.

10. $\overline{RD}$: Low on $\overline{RD}$ pin indicates 8086 is reading data from memory or an I/O device.

11. **MN/$\overline{MX}$:** This pin decides whether 8086 is to be operated in minimum mode or maximum mode.

12. **INTA (Interrupt Acknowledge) output :** This indicates recognition of an interrupt request. Whenever this goes low, it means that 8086 has accepted the interrupt.

13. **ALE (Address Latch Enable) output :** This signal is provided by 8086 to demultiplex $AD_0 - AD_{15}$ into $A_0 - A_{15}$ and $D_0 - D_{15}$ using external latch.

14. $\overline{DEN}$ **(Data Enable) output :** This signal informs the transceiver that the CPU is ready to send or receive data.

15. **DT/$\overline{R}$ (Data Transmit/Receive) output :** This is used to decide the direction of data flow through the transceivers. When the processor sends out the data this signal is high and when the processor is receiving data this signal is low.

16. **M/$\overline{IO}$:** It is used to distinguish memory data transfer, (M/$\overline{IO}$ = High) and I/O data transfer (M/$\overline{IO}$ = Low).

17. $\overline{WR}$: This signal is low when 8086 is writing into the memory or I/O device.

18. **HOLD Input, HLDA Output :** A high on HOLD pin indicates that another master is requesting to take over the system bus. On receiving HOLD signal, processor output HLDA signal high as acknowledgement. At the same time processor tri-states the system bus. A low on HOLD gives the system bus control back to the processor. Then output is low signal on HLDA.

19. **QS_1, QS_0 (Output) :** These two output signals reflect the status of the instruction queue.

QS_1	QS_0	Status
0	0	No operation (queue is idle)
0	1	First byte of an opcode
1	0	Queue is empty
1	1	Subsequent byte of an opcode

20. **$\overline{S}_2, \overline{S}_1, \overline{S}_0$ (Output) :** These three status signals indicates the type of transfer to be take place during the current bus cycle.

$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	Machine cycle
0	0	0	Interrupt acknowledge
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	Halt
1	0	0	Instruction fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Inactive-passive

21. $\overline{LOCK}$: This signal indicates that an instruction with $\overline{LOCK}$ prefix being executed and the bus is not to be used by another processor.

22. $\overline{RQ}/\overline{GT_1}$ and $\overline{RQ}/\overline{GT_0}$: In maximum mode, HOLD and HLDA pins are replaced by $\overline{RQ}$ (bus request) / $\overline{GT_0}$ (bus grant) and $\overline{RQ}/\overline{GT_1}$ signal. These lines are bi-directional and are used to both request and grant a DMA operation.

Que 3.3. **Draw and explain in detail the architecture of 8086 (pin and functional block diagram).** **AKTU 2015-16, Marks 15**

OR

Explain the role of BIU and EU of 8086 μp.

Answer

A. **Pin diagram :** Refer Q. 3.2, Page 3–3B, Unit-3.

B. **Architecture :**

The internal block diagram of 8086 is shown in Fig. 3.3.1. It is internally divided into two separate functional units as follows :

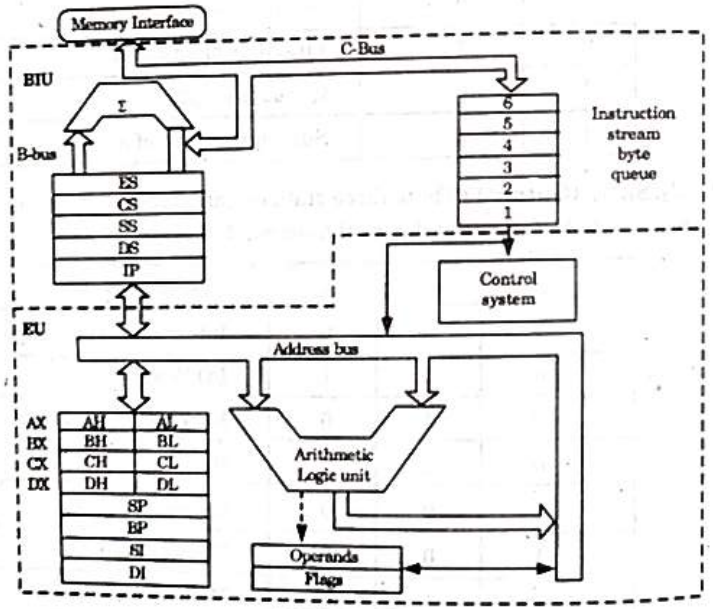


Fig. 3.3.1.

1. **Bus Interface Unit (BIU) :** 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. 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.

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.

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.

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

Que 3.4. **Explain the register organization of 8086.**

OR

Discuss the register organization of 8086 μp and explain the function of each register.

Answer

8086 has a powerful set of registers containing general purpose and special purpose registers. All the registers of 8086 are 16-bit registers.

A. **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.

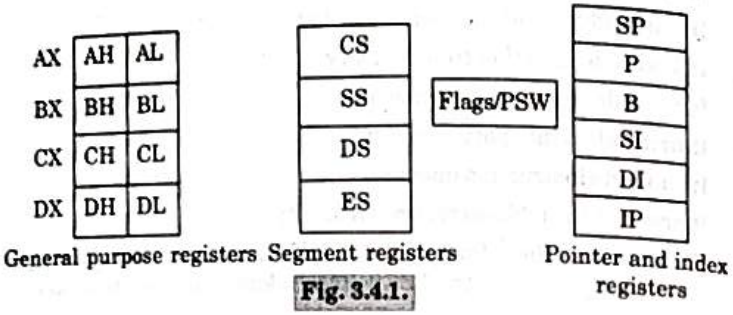


Fig. 3.4.1.

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

1. **CS :** The Code Segment (CS) 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 (DS) register points to the data segment of the memory, where the data is resided.
3. **ES :** The Extra Segment (ES) 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 (SS) register is used for addressing stack segment of memory. The stack segment is that segment of memory which is used to store stack data.

C. Pointer and Index registers :

1. The pointers contain offset within the particular segments. The pointers IP, BP and SP usually contain offsets within the code, data and stack segments respectively.
2. The index registers are used as general purpose registers as well as for offset storage. In case of indexed, based indexed and relative based indexed addressing modes.
3. 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.

D. Flag registers : 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

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 a 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.
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 is the memory segmentation in 8086 ? Explain the advantages of segmentation.

Answer

A. 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. The 16-bit contents of the segment register gives the starting/base address of a particular segment, as shown in Fig. 3.5.1.

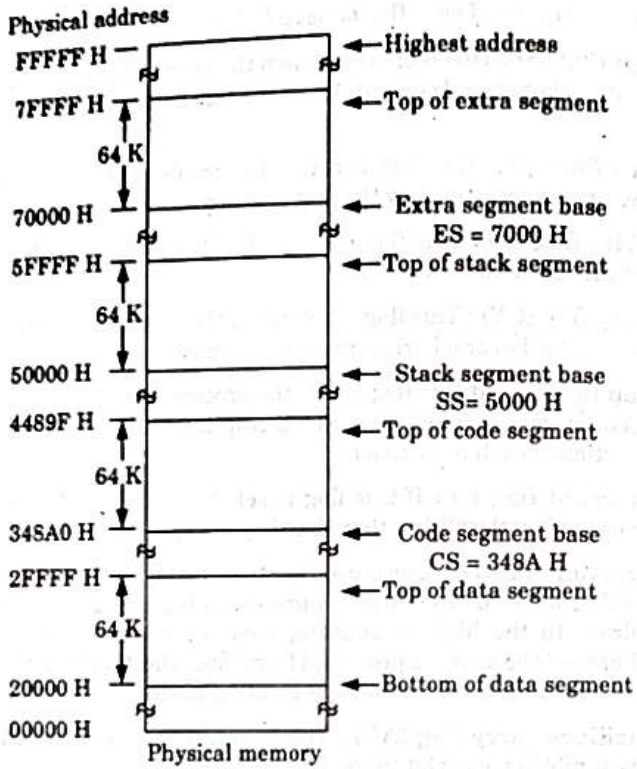


Fig. 3.5.1. Memory segmentation.

5. To address a specific memory location within a segment we need an offset address.
6. The offset address is also 16-bit wide and it is provided by one of the associated pointer or index register.

B. Rules for memory segmentation :

1. The four segments can overlap for small programs. In a minimum system all four segment can start at the address 00000 H.
2. The segment can begin/start at any memory address which is divisible by 16.

C. Advantages of memory segmentation :

1. It allows the memory addressing capacity to be 1 MB even though the address associated with individual instruction is only 16-bit.
2. It allows instruction code, data, stack and portion of program to be more than 64 KB long by using more than one code data, stack segment and extra segment.
3. It facilitates use of separate memory areas for program, data and stack.
4. It is useful in multi-programming.

Que 3.6. Explain the advantages of dividing memory into segments. How is the 20-bit physical address for memory is generated ? Explain with example.

Answer

A. Advantages of memory segmentation : Refer Q. 3.5, Page 3-9B, Unit-3.

B. Physical address formulation :

1. For generating a physical address, the content of a segment register also called as segment address is shifted left bit-wise four times and to this result, content of an offset register also called as offset address is added, to produce a 20-bit physical address.
2. For example, if the segment address is 1005 H and the offset is 5555 H, then the physical address is calculated as follows :

```
        Segment address = 1005 H
          Offset address = 5555 H
        Segment address = 1005 = 0001 0000 0000 0101
Shifted by 4-bit positions =   0001 0000 0000 0101 0000
          Offset address = +         0101 0101 0101 0101
                                 ----------------------
       Physical address =    0001 0101 0101 1010 0101
       Physical address = 155A5 H
```

So, 155A5 H is the physical address.

Generation of 20-bit address :

1. To access a specific memory location from any segment we need 20-bit physical address.
2. The 8086 generates this address using the contents of segment register and the offset register associated with it.
3. The CS register holds the base address of the code segment.
4. The 8086 provides an instruction pointer (IP) which holds the 16-bit address of the next code byte within the code segment.
5. The value contained in the IP is referred to as an offset.
6. This value must be offset from the segment base address in CS to produce the required 20-bit physical address.
7. The contents of the CS register are multiplied by 16 (10 H), *i.e.*, shifted by 4 position to the left by inserting 4 zero bits and then the offset, *i.e.*, the contents of IP register are added to the shifted contents of CS to generate physical address.
8. The content of CS register are 348A H, therefore the shifted contents of CS register are 348A0 H.
9. When the BIU adds the offset of 4214 H in the IP to this starting address, we get 38AB4 H as a 20-bit physical address of memory.

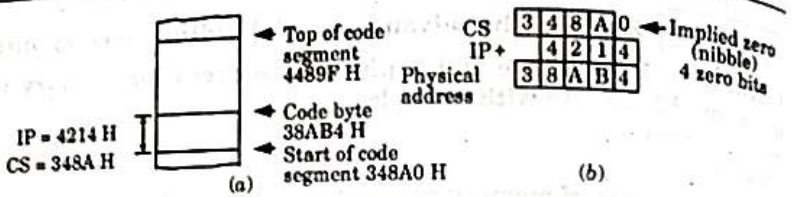


Fig. 3.6.1. Physical address.

Que 3.7. **What is the difference between maximum and minimum mode of operation in 8086 ?**

Answer

S. No.	Maximum mode	Minimum mode
1.	In this mode, there may be more than one microprocessor in system configuration.	There is a single microprocessor in minimum mode system.
2.	Control signals are issued by Intel 8288 bus controller.	CPU issues the control signals required by memory and I/O devices.
3.	In this mode, the 8086 is operated by strapping the MN / $\overline{MX}$ pin to ground.	In this mode, the 8086 is operated by strapping its MN / $\overline{MX}$ pin to logic 1.
4.	In this mode, $\overline{RQ}$ / GT_0 and $\overline{RQ}$ / GT_1 are used to deal with other devices.	In this mode, HOLD and HLDA signals are used to deal with other devices.
5.	This mode can be used for multiplexed operation.	This mode is not used when multiplexed operations are to be performed.

Que 3.8. **Define the addressing modes of 8086 microprocessor. Explain each addressing mode with example.**

Answer

Addressing modes : Addressing mode indicates a way of locating data or operands. The addressing modes describe the types of operands and the way they are accessed for executing an instruction.

The addressing modes for sequential control transfer instructions are explained as follows :

1. **Immediate addressing mode :** In this type of addressing mode immediate data is a part of instruction and appears in the form of successive byte or bytes.
   Example : MOV AX, 0005 H
2. **Direct addressing mode :** In this mode, a 16-bit memory address (offset) is directly specified in the instruction as a part of it.
   Example : MOV AX, [5000 H]
3. **Register addressing mode :** In this mode, the data is stored in a register and it is referred using the particular register. All the registers, except IP, may be used in this mode.
   Example : MOV BX, AX.
4. **Register indirect addressing mode :**
   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 registers. The default segment is either DS or ES.
   Example : MOV AX, [BX]
5. **Indexed addressing mode :** In this addressing mode, offset of the operand is stored in one of the index register. DS and ES are the default segments for index registers SI and DI respectively. This mode is a special case of the register indirect addressing mode.
   Example : MOV AX, [SI]
6. **Register relative addressing mode :** 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).
   Example : MOV AX, 50 H[BX]
7. **Based indexed addressing mode :** 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). The default segment register may be ES or DS.
   Example : MOV AX, [BX] [SI];
8. **Relative based indexed addressing mode :** 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.
   Example : MOV AX, 50 H[BX][SI]

PART-2

Peripheral Devices : 8237 DMA Controller, 8255 Programmable Peripheral Interface.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.9. **Write a short note on Direct Memory Access (DMA) controller.** **AKTU 2015-16, Marks 05**

Answer

1. Direct Memory Access is an I/O technique commonly used for high-speed data transfer.
2. In status check I/O and interrupt I/O data transfer is relatively slow because each instruction needs to be fetched and executed.
3. In DMA, the MPU releases the control of the buses to a device called a DMA controller.
4. The controller manages data transfer between memory and a peripheral under its control thus bypassing the MPU.
5. The two important signals in DMA controller are :

a. **HOLD :**

i. This is an active high input signal to the 8085 from another master, requesting the use of the address and data buses. After receiving the Hold request, the MPU relinquishes the buses in the following machine cycle.

ii. All buses are tri-stated and the Hold acknowledge (HLDA) signal is sent out. The MPU regains the control of the buses after HOLD goes low.

b. **HLDA (Hold acknowledge) :** This is an active high output signal indicating that the MPU is relinquishing control of the buses.

6. A DMA controller uses these signals as if it were a peripheral, requesting the MPU for the control of the buses. The MPU communicates with the controller by using the chip select line buses and control signals. However once the controller has gained control, it plays the role of a processor for data transfer.
7. To perform this function the DMA controller should have :

i. A data bus.

ii. An address bus.

iii. Read/Write control signals.

iv. Control signals to disable its role as a peripheral and to enable its role as a processor.

8. This process is called switching from the slave mode to the master mode.

Que 3.10. **Describe the pin configuration of 8237 DMA controller.**

Answer

Pin diagram of 8237 DMA controller is shown in Fig. 3.10.1. 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.

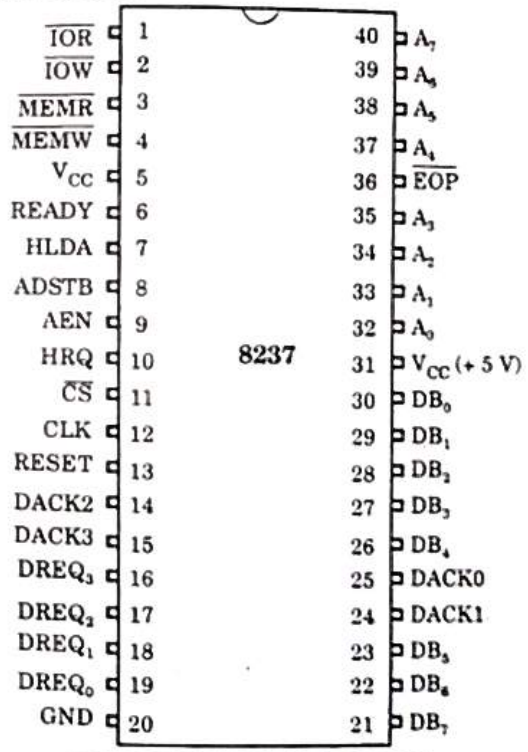


Fig. 3.10.1. Pin diagram of 8237.

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.

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 3.11. **With the help of the block diagram, describe 8237 DMA controller in detail.** **AKTU 2016-17, Marks 10**

Answer

1. The internal block diagram is shown in Fig. 3.11.1.
2. The 8237 contains three basic block of its operational logic.

3. The timing and control block generates the internal timings and external control signals.
4. The programmed command control block decodes the various commands given to the 8237 by the CPU before servicing a DMA request. It also decodes the mode control word used to select the type of DMA during services.
5. The priority encoder block resolves the priority between DMA channels requesting the service simultaneously.
6. The timing and control block derives necessary timing from clock input.
7. **Internal registers :** The 8237A contains 344 bits internal memory in the form of registers, such as base address registers, current address registers, temporary address registers etc.

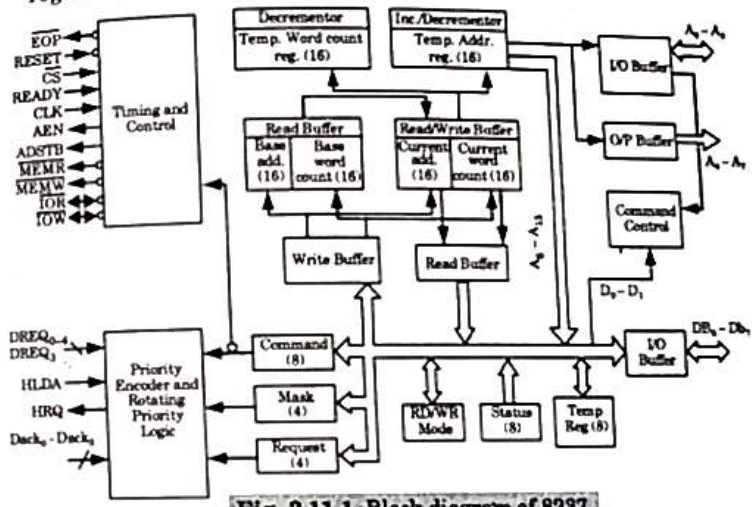


Fig. 3.11.1. Block diagram of 8237.

Que 3.12. **Describe the organization and modes of operations of 8237 DMA controller.**

OR

Discuss internal block diagram of 8237 and explain the operating mode of 8237A. **AKTU 2014-15, Marks 10**

Answer

A. **Internal block diagram and organization :** Refer Q. 3.11, Page 3–16B, Unit-3.

B. **Modes :**

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.
5. For a new DREQ to 8237, it will again activate the HRQ signal to the CPU and the HLDA signal from the CPU will push the 8237 again into single transfer mode. This mode is also called as 'cycle stealing'.

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.
4. Auto initialization may be programmed in this mode.

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.
4. Only the EOP generated by TC or external EOP can cause the auto initialization.

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.
4. The first device is used for prioritizing.

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.

Que 3.13. What are the features of 8255 PPI ?

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. Each port has a unique address, and can be read from or written to a part.
6. The address is assigned to the control register into which control words are written for programming the 8255 to operate in various modes.
7. The 8255 can operate in three I/O modes :

i. In Mode 0, Port *A* and Port *B* can be configured as simple 8-bit input or output ports without handshaking. The two halves of Port *C* can be programmed separately as 4-bit input or output ports.

ii. In Mode 1, two groups each of 12 pins are formed. Group *A* consists of Port *A* and the upper half of Port *C* while Group *B* consists of Port *B* and the lower half of Port *C*. Ports *A* and *B* can be programmed as 8-bit Input or Output ports with three lines of Port *C* in each group used for handshaking.

iii. In Mode 2, only Port *A* can be used as a bi-directional port. The handshaking signals are provided on five lines of Port *C* ($PC_3 - PC_7$). Port *B* can be used in Mode 0 or in Mode 1.

Que 3.14. Explain the architecture of 8255 PPI with neat diagram.

AKTU 2016-17, Marks 7.5

OR

With the help of block diagram explain the operation of 8255 (PPI) in detail.

Answer

1. The 8255 is a Programmable Peripheral Interface (PPI) which is used for parallel data transfer.

2. It is used as a general purpose device for interfacing parallel input/output devices to the system data bus.
3. It has three 8-bit ports : Port *A*, Port *B* and Port *C*. The 8255 can be programmed to operate in three modes : Mode 0, Mode 1 and Mode 2.

Data bus control buffer :

1. This tri-stated bi-directional buffer is used to interface the 8255 to the system data bus. IN or OUT instructions executed by the CPU either read data from, or write data into the buffer.
2. 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.

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.

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.

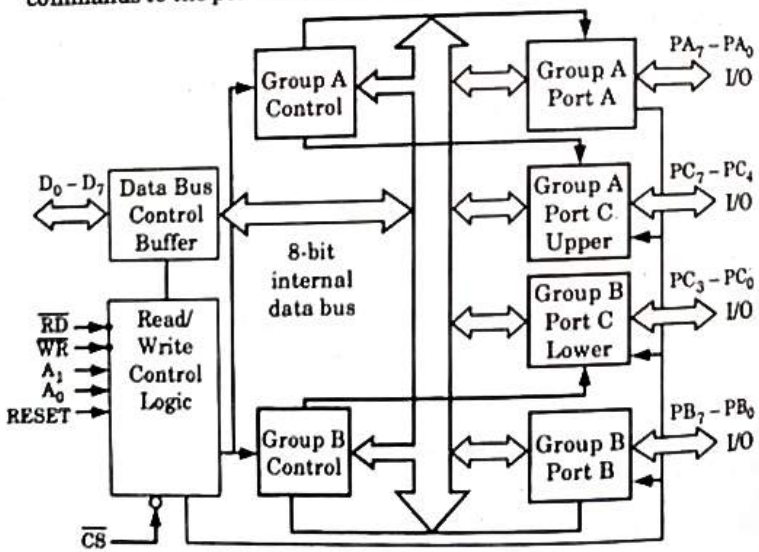


Fig. 3.14.1. 8255 block diagram.

2. The group *A* control block controls Port *A* and $PC_7 - PC_4$, while the group *B* control block controls Port *B* and $PC_3 - PC_0$.

i. **Port *A* :** This has an 8-bit latched and buffered output and 8-bit input latch.

ii. **Port *B* :** This has an 8-bit input/output latch/buffer and an 8-bit data input buffer.

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

Que 3.15. **Describe 8255 (PPI) architecture. Explain its different modes.**

Answer

A. **Architecture :** Refer Q. 3.14, Page 3–19B, Unit-3.

B. **Modes :**

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. Therefore, four ports, each of which can be configured either as an input or an output port are available.
2. Here the ports are simple input or output ports; data is written to or read from the specified port without handshaking. 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). 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.
2. 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. 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 an 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 3.16. **Write a BSR control word subroutine to set bits PC_7 and PC_3 and reset them after 10 ms. The address of control word register is 83 H. Also, write subroutine of 10 ms.** **AKTU 2016-17, Marks 7.5**

Answer

BSR control word :

This control word, when written in the control register sets or resets one bit at a time as specified in Fig. 3.16.1.

Example : Write a BSR control word subroutine to set bits PC_7 and PC_3 and reset them after 10 ms (Fig. 3.16.1).

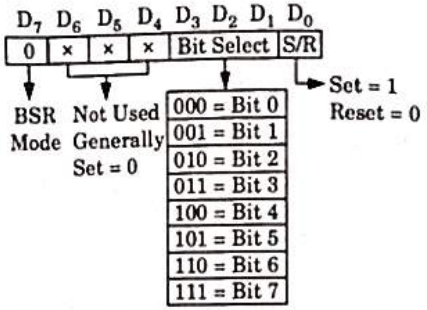


Fig. 3.16.1. 8255 control word format in the BSR mode.

BSR control words :

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
To set bit PC_7 =	0	0	0	0	1	1	1	1	= 0F H
To reset bit PC_7 =	0	0	0	0	1	1	1	0	= 0E H
To set bit PC_3 =	0	0	0	0	0	1	1	1	= 07 H
To reset bit PC_3 =	0	0	0	0	0	1	1	0	= 06 H

Port address : Control register address = 83 H

Subroutine :

```
BSR :  MVI A, 0F H   ; Load byte in accumulator to set PC7
       OUT 83 H      ; Set PC7 = 1
       MVI A, 07 H   ; Load byte in accumulator to set PC3
       OUT 83 H      ; Set PC3 = 1
       CALL DELAY    ; This is a 10 ms delay
       MVI A, 06 H   ; Load accumulator with the byte to reset PC3
       OUT 83 H      ; Reset PC7
       MVI A, 0E H   ; Load accumulator with the byte to reset PC7
```



```
       OUT 83 H      ; Rest PC7
       RET
```

Subroutine of 10 ms :

Required T-states = 10 ms/0.333 μsec
= 30030 T-states

```
       LXI B, COUNT
L1 :   DCX B
       MOV A, B
       ORA C
       JNZ L1
       RET
```

$$T_d = 10 + \text{Count} \times (6 + 4 + 4 + 10) + 10 - 3$$
$$= 30030$$
$$24\ \text{COUNT} = 30030 - 17$$
$$\text{COUNT} = (1250)_{10}$$
$$= 04E2\ \text{H}$$

Que 3.17. **Explain the interface between 8085 μP with 8255 PPI.**

Answer

Step 1 :

i. Lower order of 8-bit address A_0-A_7 is separated from AD_0-AD_7 using address latch/buffer and ALE signal.

ii. The separated address lines A_0-A_7 are connected to A_0-A_7 input pins of 8255 and the separated data bus D_0-D_7 are connected to D_0-D_7 pins of 8255.

iii. Reset out of 8085 is connected to reset pin of 8255.

Step 2 :

i. 8255 does not have internal (separate) control logic generator, hence the IO/$\overline{M}$, $\overline{RD}$ and $\overline{WR}$ control signals are not connected directly to 8255. These pins are first given to decoder and decoded using 3:8 decoder.

ii. The generated control signals $\overline{IOR}$ and $\overline{IOW}$ are connected to $\overline{RD}$ and $\overline{WR}$ input of 8255.

Step 3 :

i. An active low signal of chip select logic is obtained by decoding remaining address lines of lower order addresses A_2-A_7.

ii. Chip select logic and I/O port address for this interfacing circuit are as follows :

Chip select address lines	Address lines to select port	HEX address	Selected I/O						
A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0		
1	0	0	0	0	0	0	0	80 H	PORT A
1	0	0	0	0	0	0	1	81 H	PORT B
1	0	0	0	0	0	1	0	82 H	PORT C
1	0	0	0	0	0	1	1	83 H	Chip select register

Interfacing diagram :

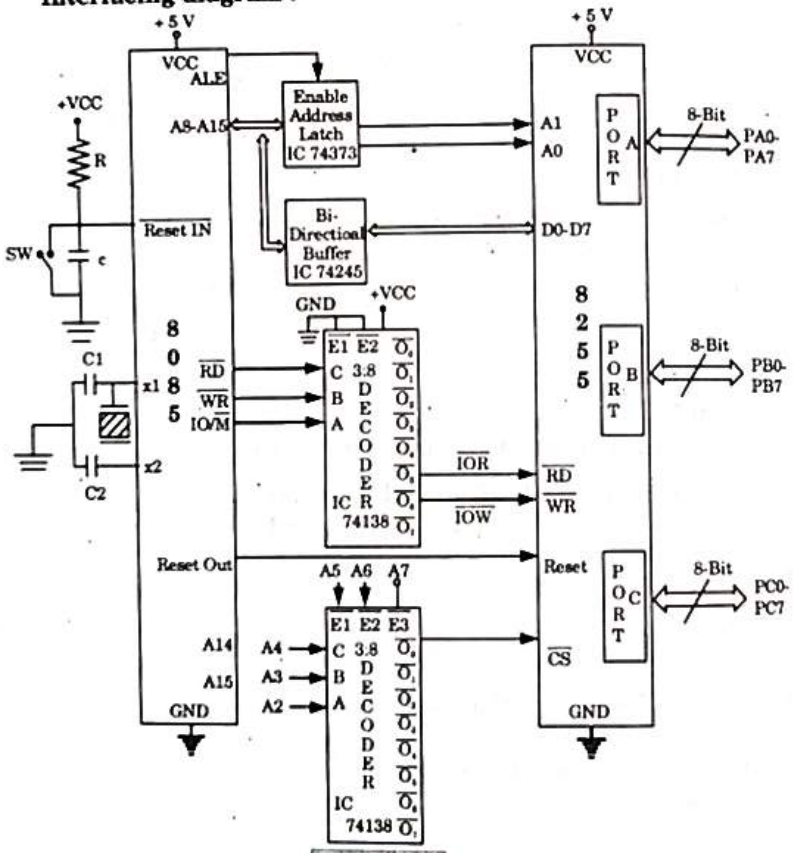


Fig. 3.17.1.

PART-3

8253/8254 Programmable Timer/Counter, 8259 Programmable Interrupt Controller.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.18. **Explain programmable interval timer 8253 or 8254 with block diagram.**

Answer

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.

Architecture and signal descriptions :

1. The internal block diagram of 8253 is shown in Fig. 3.18.1. 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.
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.

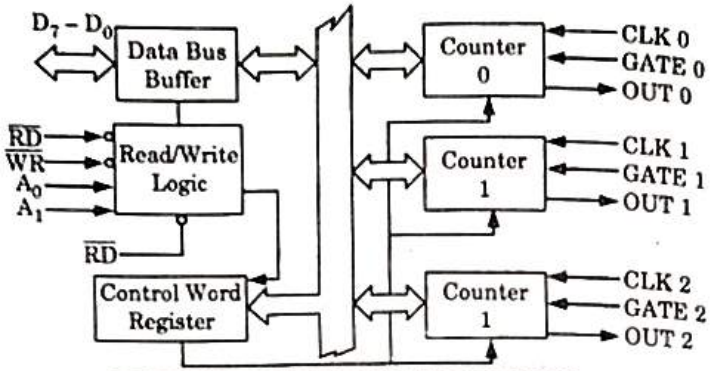


Fig. 3.18.1. Internal block diagram of 8253.

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.

Various operations for control inputs of 8253 :

Table 3.18.1.

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	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 3.19. 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. 3.19.1, 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. 3.19.1. 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 3.20. Define the modes of 8254 PIT in detail with the help of diagrams.

AKTU 2016-17, Marks 7.5

Answer

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

   In this mode, initially the OUT is low. Once a count is loaded in the register, the counter is decremented every cycle and when the count reaches zero, the OUT goes high. This can be used as an interrupt. The OUT remains high until a new count or a command word is loaded.

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

   In this mode, the OUT is initially high. 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.

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

   This mode is used to generate a pulse equal to the clock period at a given interval. When a count is loaded, the OUT stays high until the count reaches 1, and then the OUT goes low for one clock period. The count is reloaded automatically and the pulse is generated continuously.

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

   In this mode, when a count is loaded, the OUT is high. 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. This is repeated continuously; thus a continuous square wave with period equal to the period of the count is generated.

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

   In this mode, the OUT is initially high; it goes low for one clock period at the end of the count. The count must be reloaded for subsequent outputs.

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

   This mode is similar to Mode 4, except that it is triggered by the rising pulse at the gate. Initially, the OUT is low, and when the Gate pulse is triggered from low to high the count begins. At the end of the count, the OUT goes low for one clock period.

Que 3.21. **Explain the architecture of 8259A.**

Answer

The block diagram of 8259A is shown in Fig. 3.21.1. The functional explanation of each block is given as follows :

1. **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.

2. **In-Service Register (ISR) :** This stores all the interrupt requests those are being served, *i.e.*, ISR keeps a track of the request being served.

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

4. **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.

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

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

7. **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.

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

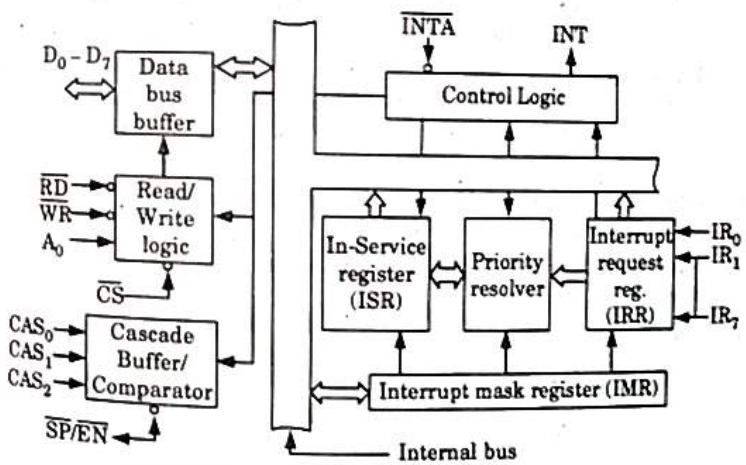


Fig. 3.21.1. Functional block diagram of 8259A.

Que 3.22. **Explain 8259 (programmable interrupt controller) in detail. Explain the different priority modes on 8259 (PIC).**

Answer

A. **8259 (PIC) :** Refer Q. 3.21, Page 3–28B, Unit-3.

B. **Priority modes on 8259 :**

1. **Fully nested mode :** This is a general-purpose mode in which all IR_s (Interrupt Requests) are arranged from highest to lowest, with IR_0 as the highest and IR_7 as the lowest.

   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			

2. **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 seven priority as shown below :

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

3. **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.

   End of Interrupt : After the completion of an interrupt service, the corresponding ISR bit needs to be reset to update the information in the ISR. This is called the End-of-Interrupt (EOI) command. It can be issued in three formats :

i. **Non-specific EOI command :** When this command is sent to the 8259A, it resets the highest priority ISR bit.

ii. **Specific EOI command :** This command specifies which ISR bit to reset.

iii. **Automatic EOI :** In this mode no command is necessary. During the third $\overline{INTA}$, the ISR bit is reset. The major drawback with this mode is that the ISR does not have information on which IR is being serviced. Thus any IR can interrupt the service routine irrespective of its priority if the Interrupt Enable flip-flop (IE) is set.

Que 3.23. **How many 8259 can be interconnected in cascaded mode ? Show their structure.**

Answer

1. Upto eight 8259s may be cascaded together to act as slave unit to a master 8259.
2. Thus, a total of 64 I/O devices may be connected to the μP via the 8259 to transfer data in interrupt driven mode.
3. Fig. 3.23.1 exhibits a system having two 8259s – one master and one slave used for interfacing 15 I/O devices to 8085.

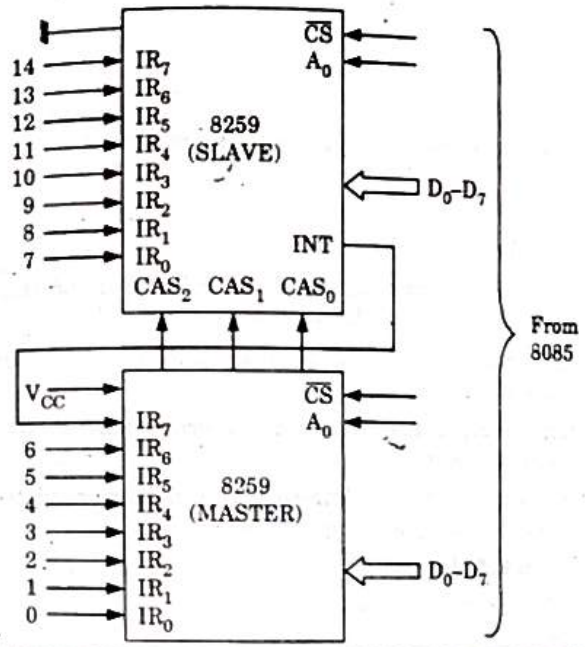


Fig. 3.21.1. Using the 8259 PICs to interface 15 I/O device to 8085 each having a unique ISS address.

4. The INT output of the slave is connected to the IR_7 input of the master.
5. If any of the seven devices 0-6 interrupts, the sequence of actions taken by the master.
6. When any of devices 7-14 interrupts, the slave will issue an interrupt to the μP if no other higher priority interrupt is pending.
7. On receipt of the INTA pulse, the master will send out the first byte of the CALL instruction. It will enable the slave by issuing a slave address code on the CAS lines.

8. The slave corresponding to this code will send the next two bytes of the CALL instruction directly to the μP in the next tow INTA pulses. This address would obviously correspond to the interrupting device.

PART-4

8251 USART and RS232C.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.24. **What is USART ? What are its features ?**

Answer

A. 8251 USART:

1. 8251A Universal Synchronous Asynchronous Receiver and Transmitter (USART) is 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. Single + 5 V supply.
2. Compatible with 8085 and 8086/8088 CPU.
3. Supports both synchronous and asynchronous modes of operations.

i. In synchronous mode it supports 5-8 bit characters, internal or external character synchronization at receiver, automatic sync insertion at transmitter.

ii. In asynchronous mode it supports 5-8 bit characters, clock rate selectable 1, 16 or 64 times baud rate, break character generation, '1, $1^{1/2}$ or 2 stop bits, false start bit detection, odd, even or no parity generation and detection, automatic breaks detect circuitry are also available.

4. Transmitter and receiver contain full duplex, double buffered system.
5. Error detection-Parity, overrun, framing.

6. Separate TXC and RXC clock inputs for transmitter and receiver. So transmitter and receiver can be operated in different baud rates.

Que 3.25. **Explain the internal architecture of 8251.**

Answer

1. The architectural block diagram of 8251A or 8251 is shown in Fig. 3.25.1, 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.

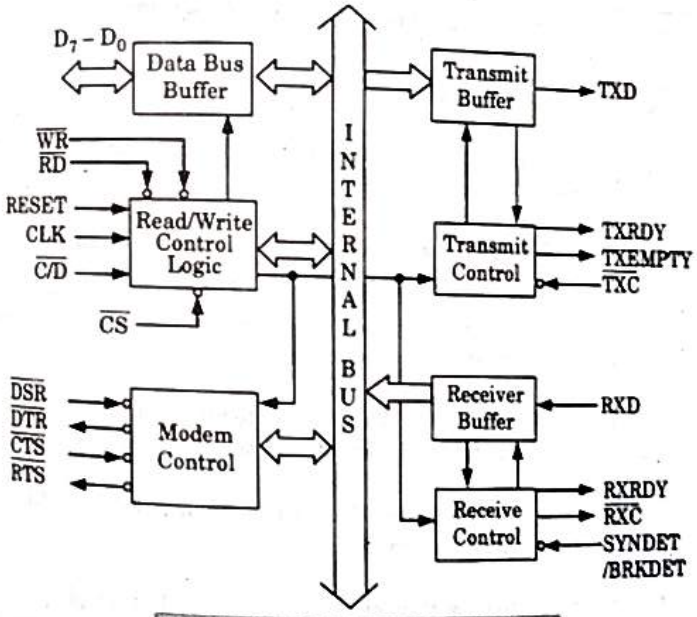


Fig. 3.25.1. 8251A Internal architecture.

4. The modem control unit handles the modem handshake signals to coordinate the communication between the modem and the USART.

5. The transmit control unit transmits the data byte received by the data buffer from the CPU for further serial communication. 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. 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 3.26. **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.

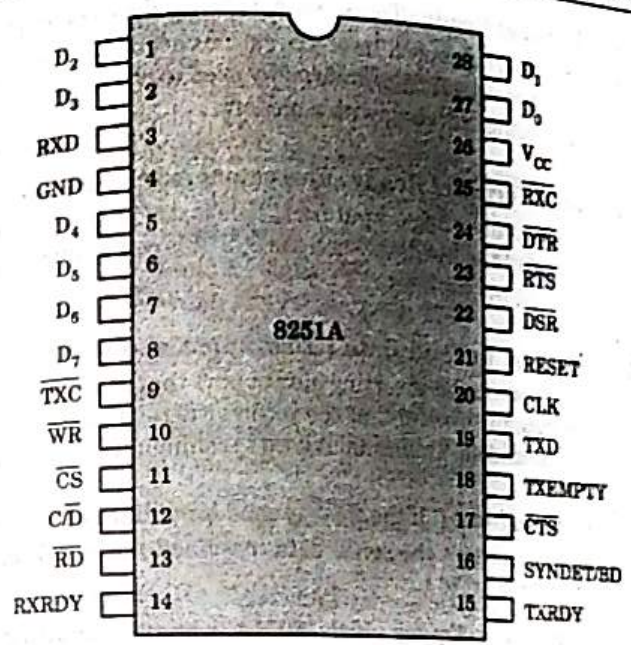


Fig. 3.26.1. 8251A pin configuration.

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.

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 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.

Que 3.27. **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. 3.24.1 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.

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 $\overline{CTS}$.

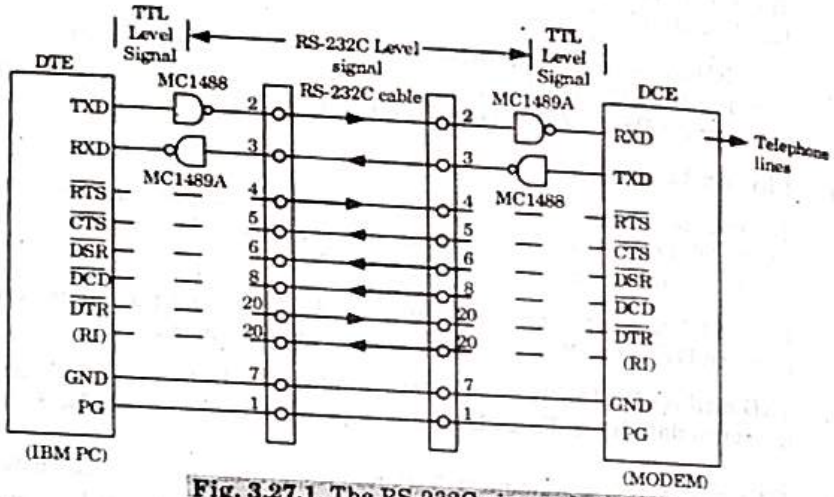


Fig. 3.27.1. The RS-232C pins and signals.

VERY IMPORTANT QUESTIONS

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

Q. 1. Draw the PIN diagram of 8086 microprocessor and discuss about each pin.

Ans. Refer Q. 3.2.

Q. 2. Write a short note on Direct Memory Access (DMA) controller.

Ans. Refer Q. 3.9.

Q. 3. With the help of the block diagram, describe 8237 DMA controller in detail.

Ans. Refer Q. 3.11.

Q. 4. Explain the architecture of 8255 PPI with neat diagram.

Ans. Refer Q. 3.14.

Q. 5. Write a BSR control word subroutine to set bits PC_7 and PC_3 and reset them after 10 ms. The address of control word register is 83H. Also, write subroutine of 10 ms.

Ans. Refer Q. 3.16.

Q. 6. Define the modes of 8254 PIT in detail with the help of diagrams.

Ans. Refer Q. 3.20.

Q. 7. Explain 8259 (programmable interrupt controller) in detail. Explain the different priority modes on 8259 (PIC).

Ans. Refer Q. 3.22.

Q. 8. What is USART ? What are its features ?

Ans. Refer Q. 3.24.

Q. 9. Describe the pin configuration of USART.

Ans. Refer Q. 3.26.

☺☺☺

t.me/quantum_aktu

aktu quantum

please support
us

- 5 Sem
- 6 Sem
- 7 Sem
- 8 Sem - future support

t.me/quantum_aktu

aktu quantum

please support
us

- 5 Sem
- 6 Sem
- 7 Sem
- 8 Sem - future support

4 UNIT

8051 Microcontroller Basics

CONTENTS

Part-1 : Inside the Computer, Microcontrollers and Embedded Processors ..... 4-2B to 4-5B

Part-2 : Block Diagram of 8051, PSW and Flag Bits, 8051 Register Banks and Stack ..... 4-5B to 4-10B

Part-3 : Internal Memory Organization of 8051, Pins of 8051, I/O Port Usage in 8051, Types of Special Function Registers and their uses in 8051 ..... 4-10B to 4-14B

Part-4 : Memory Address Decoding, 8031/8051 Interfacing with External ROM and RAM, 8051 Addressing Modes ..... 4-15B to 4-20B

PART-1

Inside the Computer, Microcontrollers and Embedded Procesors.

CONCEPT OUTLINE

- An embedded system is a combination of special purpose hardware and software for executing a specific set of applications.
- The 8051 is an 8-bit processor that can work only on 8-bit of data at a time. It has 128 bytes of RAM, 4 Kbytes of on-chip ROM, two timers, one serial port and four I/O port (each 8-bit wide).

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.1. **Draw and explain the block diagram of computer.**

Answer

1. The internal working of every computer can be broken down into three parts : CPU (Central Processing Unit), memory and I/O (Input/Output) devices as shown in Fig. 4.1.1.

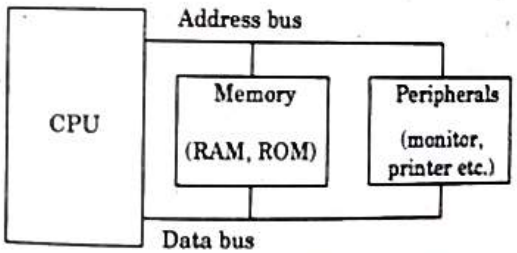


Fig. 4.1.1. Inside the computer.

2. The function of the CPU is to execute (process) information stored in memory.
3. The function of I/O devices such as the keyboard and video monitor is to provide a means of communicating with the CPU.
4. The CPU is connected to memory and I/O through strips of wire called a bus. The bus carries information from place to place inside a computer.

5. In every computer there are three types of buses: address bus, data bus, and control bus.
6. The CPU puts the address (in binary) on the address bus, and the decoding circuitry finds the device. Then the CPU uses the data bus either to get data from that device or to send data to it.
7. The control buses are used to provided read or write signals to the device to indicate if the CPU is asking for information or sending it information.
8. Of the three buses, the address bus and data bus determine the capability of a given CPU.

Que 4.2. **What do you mean by embedded system ? Describe its purpose and main application areas.**

Answer

A. Embedded system :

1. An embedded system is an electronic/electro-mechanical system designed to perform a specific function and is a combination of both hardware and firmware (software).
2. Every embedded system is unique, and the hardware as well as the firmware is highly specialised to the application domain.
3. Embedded systems are becoming an inevitable part of any product or equipment in all fields including household appliances, telecommunications, consumer products etc.

B. Purpose :

Each embedded system is designed to serve the purpose of any one or a combination of the following tasks :

1. Data collection/storage/representation.
2. Data communication.
3. Data (signal) processing.
4. Monitoring.
5. Control.
6. Application specific user interface.

C. Application :

1. Consumer electronics : Camcorders, cameras, etc.
2. Household appliances : T.V, washing machine, fridge, etc.
3. Home automation and security system : Air conditioners, sprinklers, fire alarms, etc.
4. Telecom : Cellular telephones, telephone switches, etc.
5. Computer peripherals : Printer, scanners, etc.
6. Healthcare : EEG, ECG machines etc.

Que 4.3. **List some important features and architecture consideration of an embedded system.**

Answer

A. Important features :

1. Embedded systems execute pre-programmed functions and they have a particular set of requirements.
2. They are programmed hardware devices that run on hardware chips that are programmable.
3. Embedded systems perform a specific function or a set of specific functions unlike a computer, which is used to carry out a wide number of functions.
4. Embedded systems form smaller parts of a much larger device that carries out a specific task.
5. Embedded systems can either have no user interface or possess highly advanced graphical interfaces. It mainly depends on the purpose of the device or the function it is designed to carry out.
6. Simple embedded systems use LEDs, buttons or LCD displays with simple menu options.

B. Architecture considerations :

1. The architecture of an embedded system is an abstraction of the embedded device, meaning that it is a generalization of the system.
2. The hardware and software components in an embedded system are instead represented as composition of interacting elements.
3. Architectural element can be internally integrated within the embedded device or exist externally to the embedded system and interact with internal elements.

Que 4.4. **Explain microcontroller. Why microcontrollers are preferred for controlling operation ?**

Answer

A. Microcontroller :

1. A microcontroller contains a fixed amount of RAM, ROM, parallel I/O ports, serial I/O parts, counter a clock circuit all on a single chip.
2. The block diagram of microcontroller is shown in Fig. 4.4.1.

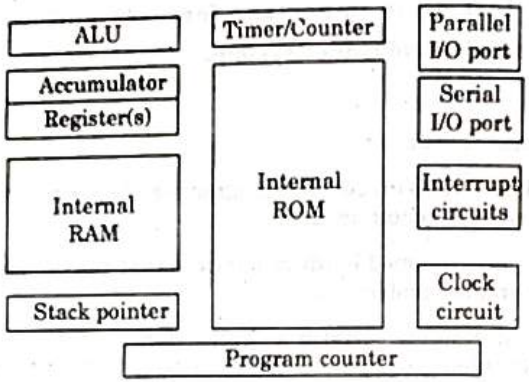


Fig. 4.4.1. Block diagram of a microcontroller.

3. Microcontrollers are intended to be special purpose digital computers.
4. Microcontroller is concerned with rapid movement of bits within the chip. It also consists of many bit-handling instructions.
5. Microcontrollers are normally less expensive than microprocessor.

B. Microcontrollers are preferred over microprocessors for controlling operations because :

1. They meet the computing needs of the task efficiently and cost effectively.
2. Software development tools such as compilers, assemblers and debuggers are available for operations.
3. Wide availability and reliable sources of the microcontroller.
4. The power consumption of this device is also very low.

PART-2

Block Diagram of 8051, PSW and Flag Bits, 8051 Register Banks and Stack.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.5. **Draw the functional block diagram of an 8051 microcontroller.**

Answer

1. The 8051 microcontroller contains 128 bytes of RAM, 4 Kbytes of on-chip ROM, two timers, one serial port, and four ports (each 8-bits wide) all on a single chip. It is also referred as a "system on a chip".
2. The 8051 is an 8-bit processor, meaning that the CPU can work on only 8-bits of data at time. Data larger than 8 bits has to be broken into 8-bit pieces to be processed by the CPU.
3. The 8051 has a total of four I/O ports, each 8 bits wide as shown in Fig. 4.5.1.

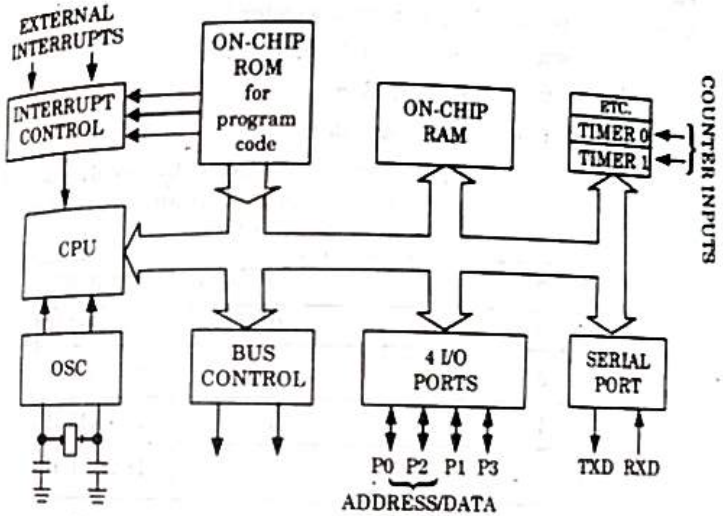


Fig. 4.5.1. Inside the 8051 microcontroller block diagram.

Que 4.6. **What is PSW register ? Explain the format of PSW of 8051.**

Answer

A. PSW : The Program Status Word (PSW) register is an 8-bit register. It is also referred to as the flag register. Although the PSW register is 8 bits wide, only 6 bits of it are used by the 8051. The two unused bits are user-definable flags.

B. **Format of PSW :**

1. In this, four flags are called conditional flags, meaning that they indicate some conditions that result after an instruction executed.
2. These four are CY (carry), AC (auxiliary carry), P (parity), and OV (overflow) as shown in Fig. 4.6.1.

3. The bits PSW.3 and PSW.4 are designated as RS0 and RS1 respectively, and are used to change the bank registers.
4. The PSW.5 and PSW.1 bits are general-purpose status flag bits and can be used by the programmer for any purpose. In other words, they are user-definable.

CY	AC	F0	RS1	RS0	OV	–	P

CY	PSW.7	Carry flag.
AC	PSW.6	Auxiliary carry flag.
F0	PSW.5	Available to the user for general purpose.
RS1	PSW.4	Register bank selector bit 1.
RS0	PSW.3	Register bank selector bit 0.
OV	PSW.2	Overflow flag.
–	PSW.1	User-definable bit.
P	PSW.0	Parity flag. Set/Cleared by hardware each instruction cycle to indicate an odd/even number of 1 bit in the accumulator.

RS1	RS0	Register Bank	Address
0	0	0	00 H-07 H
0	1	1	08 H-0F H
1	0	2	10 H-17 H
1	1	3	18 H-1F H

Fig. 4.6.1.

CY, the carry flag : This flag is set whenever there is a carry out from the D7 bit. This flag bit is affected after an 8-bit addition or subtraction. It can also be set to 1 or 0 directly by an instruction such as "SETB C" and "CLR C" where "SETB C" stands for "set bit carry" and "CLR C" stands for "clear carry".

AC, the auxiliary carry flag :

If there is a carry from D3 to D4 during an ADD or SUB operation, this bit is set, otherwise, it is cleared. This flag is used by instructions that perform BCD (binary coded decimal) arithmetic.

P, the parity flag :

The parity flag reflects the number of 1s in the *A* (accumulator) register only. If the *A* register contains an odd number of 1s, then $P = 1$. Therefore, $P = 0$ if *A* has an even number of 1s.

OV, the overflow flag :

This flag is set whenever the result of a signed number operation is too large, causing the high-order bit to overflow into the sign bit. In general, the carry flag is used to detect errors in unsigned arithmetic operations. The overflow flag is only used to detect errors in signed arithmetic operations.

There is no sign flag in 8051 because of the overflow problem.

Que 4.7. Write a short note on register banks in the 8051 microcontroller.

Answer

1. A total of 32 bytes of RAM is set aside for the register banks and stack as shown in the Fig. 4.7.1. These 32 bytes are divided into 4 banks of registers in which each bank has 8 registers, R0-R7.
2. RAM locations from 0 to 7 are set aside for bank 0 of R0-R7 where R0 is RAM location 0, R1 is RAM location 1, R2 is location 2, and so on, until memory location 7, which belongs to R7 of bank 0.
3. The second bank of register R0-R7 starts at RAM location 08 and goes to location 0F H.
4. The third bank of R0-R7 starts at memory location 10 H and goes to location 17 H.
5. Finally, RAM location 18 H to 1F H are set aside for the fourth bank of R0-R7.

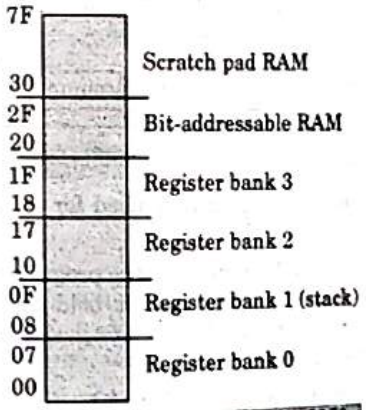


Fig. 4.7.1. RAM allocation in the 8051.

6. Bank 1 uses the same RAM space as the stack. This is a major problem in programming the 8051. One must either not use register bank 1, or allocate another area of RAM for the stack.

	Bank 0		Bank 1		Bank 2		Bank 3
7	R7	F	R7	17	R7	1F	R7
6	R6	E	R6	16	R6	1E	R6
5	R5	D	R5	15	R5	1D	R5
4	R4	C	R4	14	R4	1C	R4
3	R3	B	R3	13	R3	1B	R3
2	R2	A	R2	12	R2	1A	R2
1	R1	9	R1	11	R1	19	R1
0	R0	8	R0	10	R0	18	R0

Fig. 4.7.2. 8051 Register banks and their RAM addresses.

Que 4.8. **Which are the most widely used registers of 8051 microcontroller ?**

Answer

1. Registers are basically used to store the information temporarily. This information can be a byte of a data to be processed, or an address pointing to the data to be fetched.
2. The most widely used registers of 8051 microcontroller are *A* (accumulator), *B*, R0, R1, R2, R3, R4, R5, R6, R7, DPTR (Data Pointer) and PC (Program Counter).
3. All the above registers are 8-bits, except DPTR and PC.

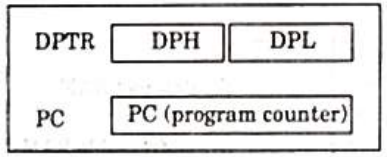


Fig. 4.8.1. Some 8051 16-bit registers.

4. The accumulator or register *A* is used for all arithmetic and logic instructions.
5. The 8051 contains 16-bit registers : the program counter (PC) and the data pointer (DPTR). Each is used to hold the address of a byte in memory. Program instruction bytes are fetched from locations in memory that are addressed by the PC.
6. The DPTR register is made up of two 8-bit registers named DPH and DPL.

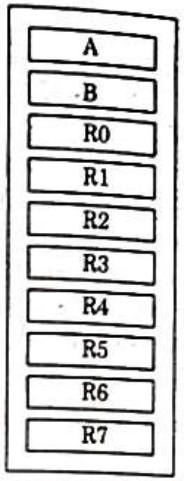


Fig. 4.8.2. Some 8-bit registers of the 8051.

PART-3

Internal Memory Organization of 8051, Pins of 8051, I/O Port Usage in 8051, Types of Special Function Register and their uses in 8051.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.9. **Explain the memory organization in 8051 microcontroller.** **AKTU 2014-15, Marks 05**

Answer

1. Fig. 4.9.1 shows the basic memory organization for 8051. It can access up to 64 K program memory and 64 K data memory. The 8051 has 4 Kbytes of internal program memory and 256 bytes of internal data memory.

   Program Memory : Program Memory (ROM) is used for permanent saving program (CODE) being executed. The memory is read only. Depending on the settings made in compiler, program memory may also used to store constant variables. The 8051 executes programs stored in program memory only.

   Internal Data Memory :

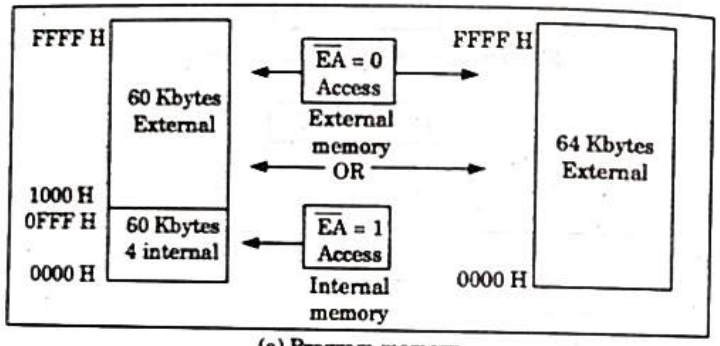
1. Up to 256 bytes of internal data memory are available depending on the 8051 derivative. Locations available to the user occupy addressing

space from 0 to 7F H, *i.e.*, first 128 registers and this part of RAM is divided in several blocks.

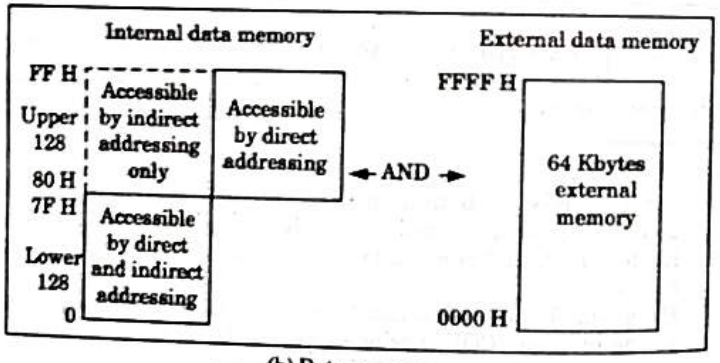
2. The first 128 bytes of internal data memory are both directly and indirectly addressable. The upper 128 bytes of data memory can be addressed only indirectly.

External Data Memory :

1. Access to external memory is slower than access to internal data memory. There may be up to 64 Kbytes of external data memory.
2. Several 8051 devices provide on-chip XRAM (External RAM) space that is accessed with the same instructions as the traditional external data space. This XRAM space is typically enabled via proper setting of SFR register and overlaps the external memory space.



(a) Program memory



(b) Data momory

Fig. 4.9.1. Memory organization of 8051.

Que 4.10. Explain pin diagram of 8051.

Answer

1. The pin diagram of 8051 consists of 40 pins. Out of 40 pins 32 pins are set aside for the four ports P0, P1, P2 and P3, where each ports takes 8 pins.
2. The rest of the pins are designated as V_{CC}, GND, XTAL 1, XTAL 2, RST, $\overline{EA}$, ALE/ $\overline{PROG}$ and $\overline{PSEN}$.

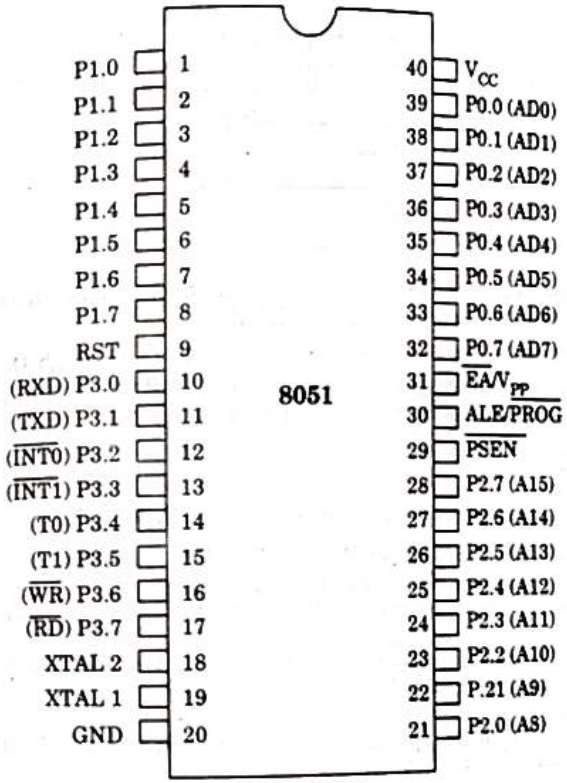


Fig. 4.10.1.

3. There are four ports P0, P1, P2 and P3 each use 8 pins, making them 8-bit ports. All the ports upon RESET are configured as inputs, ready to be used as input ports. When the first 0 is written to a port, it becomes an output. To reconfigure as an input, 1 must be sent to port.
4. V_{CC}: Pin 40 provides supply voltage to the chip. The voltage source is + 5 V.
5. **GND :** Pin 20 is the ground.

6. **XTAL 1 and XTAL 2 :**

i. The 8051 has an on chip oscillator but requires an external clock to run it. A quartz crystal oscillator is connected to inputs XTAL1 (pin 19) and XTAL2 (pin 18).

ii. The quartz crystal connected to XTAL1 and XTAL2 needs two capacitors of 30 pF value. One side of each capacitor is connected to ground as shown in Fig. 4.10.2.

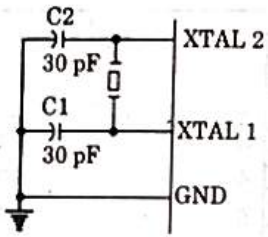


Fig. 4.10.2.

iii. There are various speeds of the 8051 family. Speed refers to the maximum oscillator frequency connected to XTAL.

7. **RST :** Pin 9 is the reset pin. It is an input and is active high. On applying a high pulse to this pin, the microcontroller will reset and terminate all activities. This is referred to as a power-on reset.

8. **$\overline{EA}$:**

i. $\overline{EA}$ stands for external access enable pin. In 8051 chips with on chip ROM such as 8751/52 or DS89C4 × 0, $\overline{EA}$ is connected to V_{cc}. For family members like 8031 and 8032 in which there is no on-chip ROM, code is stored on an external ROM and is fetched by the 8031/32.

ii. Thus for 8031, the $\overline{EA}$ pin must be connected to GND to indicate that code is stored externally.

iii. $\overline{EA}$ pin is an input pin and must be connected to either V_{cc} or GND. It cannot be left unconnected.

9. **$\overline{PSEN}$:** This is an output pin. $\overline{PSEN}$ stands for program store enable. This is an active low output pin. It acts as a strobe to read the external program memory.

10. **ALE :** Address Latch Enable (ALE) is an output pin and is active high. It is used for demultiplexing the address and data bus by connecting to the GND pin of 74LS373 chip. This ALE signal is valid only for external memory access.

Que 4.11. **Explain I/O ports in 8051 microcontroller.**

Answer

There are four I/O ports in 8051 microcontroller

i. **Port 0 :** Port 0 occupies a total of 8 pins (pins 32-39). It can be used for input or output. To use the pins of port 0 as both input and output ports, each pin must be connected externally to a 10 kΩ pull-up resistor.

ii. **Port 1 :** Port 1 occupies a total of 8 pins (pins 1 through 8). It can be used as input or output. It contrast to port 0, this port does not need any pull-up resistors since it already has pull-up resistors internally. Upon reset, port 1 is configured as an input port.

iii. **Port 2 :** Port 2 occupies a total of 8 pins (pins 21 through 28). It can be used as input or output. Port 2 does not need any pull-up resistors since it already has pull-up resistors internally. Upon reset, port 2 is configured as an input port.

iv. **Port 3 :**

1. Port 3 occupies a total of 8 pins (pins 10 through 17). It can be used as input or output. P3 does not need any pull-up resistors, just as P1 and P2 did not.

2. Although port 3 is configured as an input port upon reset, this is not the way it is most commonly used. Port 3 has the additional function of providing some externally important signals such as interrupts.

Que 4.12. **Write a short note on SFR registers and their addresses.**

Answer

A. **SFR registers :**

1. The 8051 operations that do not use the internal 128-bytes RAM addresses from 00 H to 7F H are done by a group of specific internal register, each called a special function register.

2. In 8051, register *A*, *B*, PSW and DPTR are a part of the group of registers referred as SFR.

3. SFR can be accessed by names or by their addresses.

B. **SFR register addresses :**

1. SFR has addresses between 80 H and FF H. These addresses are above 80H, since address 00 to 7F H are addresses of RAM memory inside the 8051.

2. Not all address space of 80 to FF H are used by SFR.

Some SFR addresses are

Accumulator	– 0E0 H
B Register	– 0F0 H
PSW	– 0D0 H etc.

PART-4

Memory Address Decoding, 8031/8051 Interfacing with External ROM and RAM, 8051 Addressing Modes.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.13. Enlist the methods of memory address decoding and explain any of them.

Answer

There are three methods of memory address decoding.

i. Using the 74LS138

ii. Using programmable logic

iii. **Using simple logic gates :**

1. The simplest method of constructing decoding circuitry is the use of a NAND gate. The output of a NAND gate is active low, and the CS pin is also active low, which makes them a perfect match.
2. In cases where the CS input is active high, an AND gate must be used. Using a combination of NAND gates and inverters, one can decode any address range.
3. An example of this is shown in Fig. 4.13.1, which shows that A15 - A12 must be 0011 in order to select the chip.

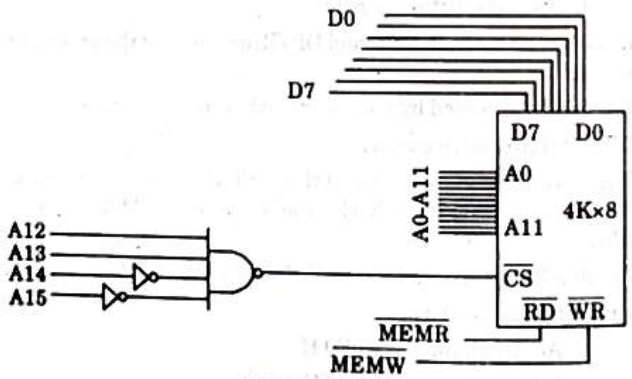


Fig. 4.13.1.

4. This results in the assignment of addresses 3000 H to 3FFF H to this memory chip.

Que 4.14. Explain 74LS138 address decoder.

Answer

1. This is one of the most widely used address decoders.
2. The 3 inputs *A*, *B*, and *C* generate 8 active-low outputs Y0 - Y7 as shown in Fig. 4.14.1.
3. Each *Y* output is connected to CS of a memory chip, allowing control of 8 memory blocks by a single 74LS138.
4. In the 74LS138, where *A*, *B*, and *C* select which output is activated, there are three additional inputs, G2A, G2B, and G1. G2A and G2B are both active low, and G1 is active high.
5. If any one of the inputs G1, G2A, or G2B is not connected to an address signal (sometimes they are connected to a control signal), they must be activated permanently either by V_{CC} or ground, depending on the activation level.

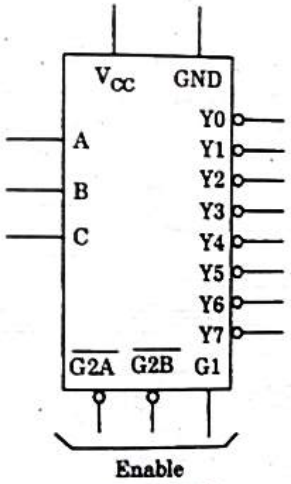


Fig. 4.14.1.

6. Example 4.14.2 shows the design and the address range calculation for the 74LS138 decoder.

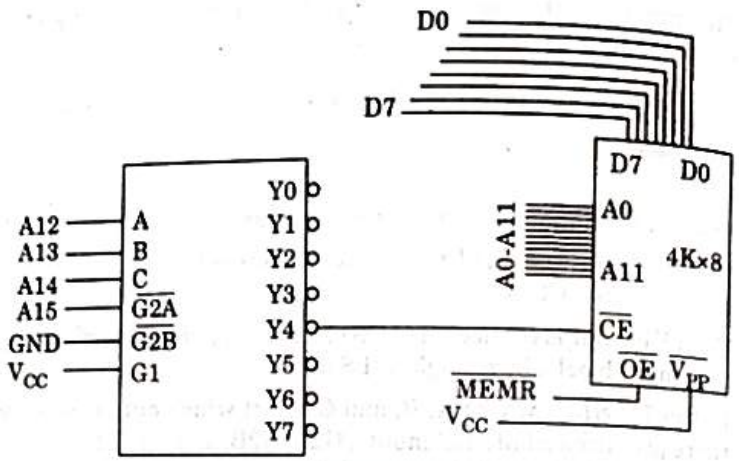


Fig. 4.14.2.

Que 4.15. **With a schematic diagram, explain how an 8051 is interfaced with external RAM ?**

Answer

1. To connect the 8051 to an external RAM both RD and WR are used.
2. In writing data to external data RAM, the instruction "MOVX@DPTR, A" is used, here the contents of register A are written to external RAM whose address is pointed by the DPTR register.

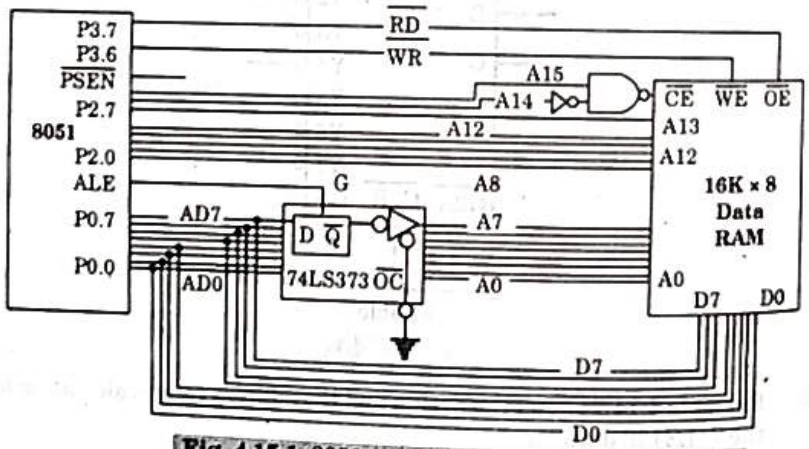


Fig. 4.15.1. 8051 connection of external data RAM.

Que 4.16. **Show the connection of an 8051 to a single 256K × 8 NV-RAM chip.**

Answer

1. The 8051 support only 64 Kbytes of external data memory since DPTR is 16 bit.

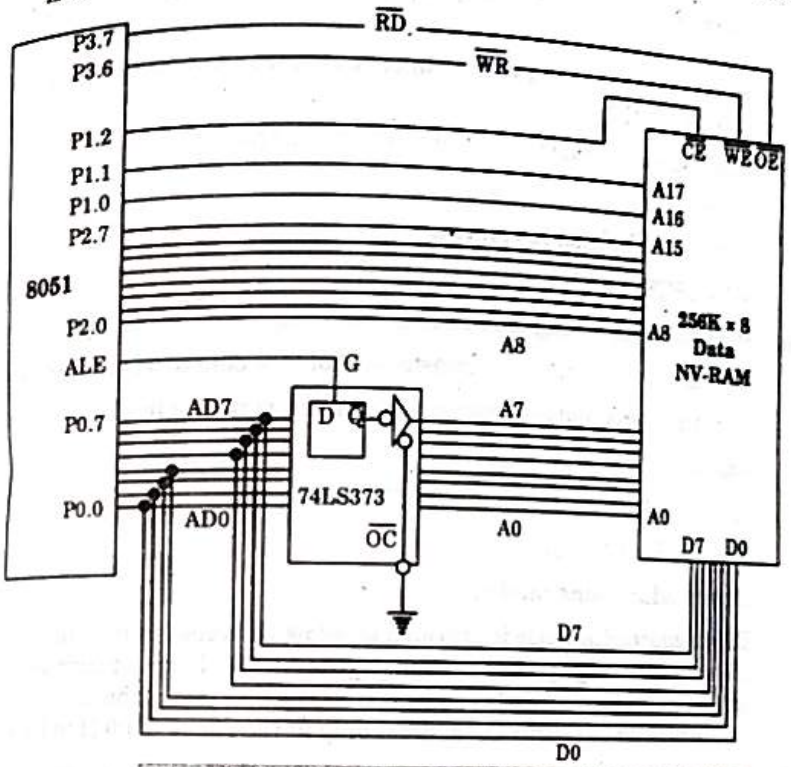


Fig. 4.16.1. 8051 accessing 256K × 8 external NV-RAM.

2. To solve the problem where we need a large amount of memory to store data (for example, 256 Kbytes), we can connect A0-A15 of 8051 directly to the external memory's A0-A15 pins, and use some of the P1 pins to access the 64 Kbyte blocks inside the single 256 K × 8 memory chip, as shown in Fig. 4.16.1.

Que 4.17. **What do you mean by addressing mode ? How many types of addressing modes in 8051 ? Give one example of each type.**

Answer

A. **Addressing mode :**

1. The CPU can access data in many ways. The data could be in a register or in memory or be provided as an immediate value. Thus, the various ways of accessing the data are called addressing modes.

B. Types of addressing mode :

i. Immediate addressing mode :

1. In this mode, the source operand is constant. As the name implies, when the instruction is assembled the operand comes immediately after the opcode.
2. This mode can be used to load information into any of the registers including DPTR.
3. Here immediate data must be preceded by the pound sign "#".

Examples :

MOV *A*, #25 H; Load 25 H into *A*

MOV DPTR, #2550 H

ii. Register addressing mode :

1. This mode involves the use of registers to hold the data to be manipulated.
2. Here the source and destination registers must match in size.

Example :

MOV *A*, R0 ; Copy the contents of R0 into *A*

MOV DPTR, #25F5 H

iii. Direct addressing mode :

1. Direct addressing mode is provided to allow us access to internal data memory, including Special Function Register (SFR). In direct addressing, an 8 bit internal data memory address is specified as part of the instruction and hence, it can specify the address only in the range of 00 H to FF H.
2. In this addressing mode, data is obtained directly from the memory.

Example :

MOV R0, 40 H ; Save content of RAM location 40 H in R0

MOV *A*, R4 ; which means copy R4 into A.

iv. Indirect addressing mode :

1. The indirect addressing mode uses a register to hold the actual address that will be used in data movement. Registers R0, R1, and DPTR are the only registers that can be used as data pointers.
2. Indirect addressing cannot be used to refer to SFR registers. Both R0 and R1 can hold 8 bit address and DPTR can hold 16 bit address.

Example :

MOV *A*, @ R0 ; Move the contents of RAM location whose address is held by R0 into *A*

MOV @ R1, *B* ; Move the contents of *B* into RAM

location whose address is held by R1.

v. Indexed addressing mode :

1. In indexed addressing, a separate register (either the program counter (PC), or the data pointer (DTPR)) is used to hold the base address, and the *A* is used to hold the offset address.
2. Adding the value of the base address to the value of the offset address forms the effective address.
3. Indexed addressing is used with JMP or MOVC instructions. Look up tables are easily implemented with the help of index addressing.

Example :

MOVC A, @ A + DPTR

Que 4.18. **Name the addressing modes of the following instructions :**

i. **MOVC *A*, @*A* + DPTR.**

ii. **MVL *A*, *B*.**

iii. **MOV *B*, #04 H.**

iv. **SUBB *A*, 45 H.**

AKTU 2014-15, Marks 05

Answer

i. Indexed addressing mode.

ii. Register addressing mode.

iii. Immediate addressing mode.

iv. Direct addressing mode.

VERY IMPORTANT QUESTIONS

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

Q. 1. List some important features and architecture consideration of an embedded system.

Ans. Refer Q. 4.3.

Q. 2. Explain microcontroller. Why microcontrollers are preferred for controlling operation ?

Ans. Refer Q. 4.4.

Q. 3. Which are the most widely used registers of 8051 microcontroller ?

Ans. Refer Q. 4.8.

Q. 4. Explain pin diagram of 8051.

Ans. Refer Q. 4.10.

Q. 5. Show the connection of an 8051 to a single 256K × 8 NV-RAM chip.

Ans. Refer Q. 4.16.

Q. 6. What do you mean by addressing mode ? How many types of addressing modes in 8051 ? Give one example of each type.

Ans. Refer Q. 4.17.

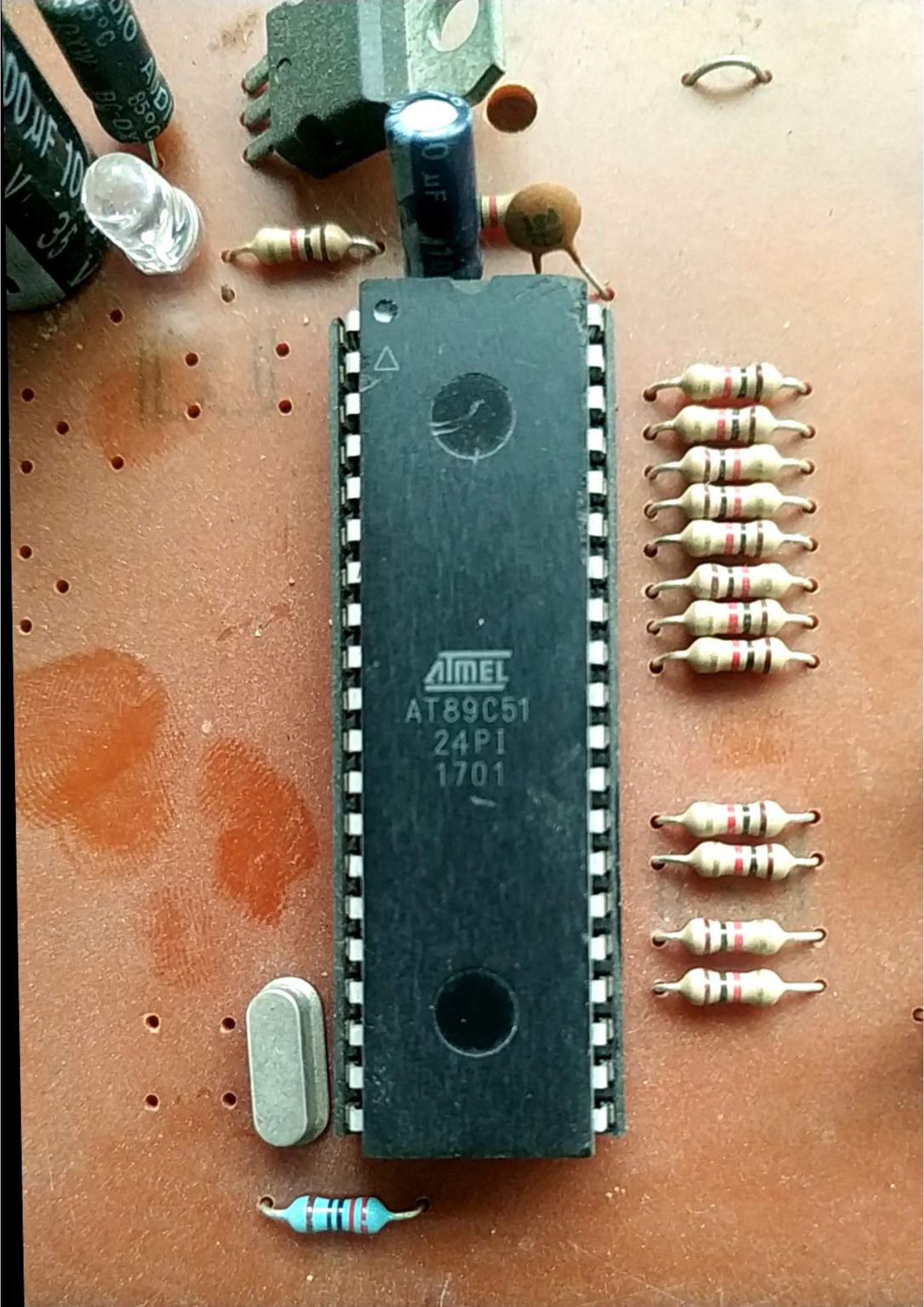
☺☺☺

t.me/quantum_aktu

aktu quantum

please support
us

- 5 Sem
- 6 Sem
- 7 Sem
- 8 Sem - future support


ATMEL
AT89C51
24PI
1701

5 UNIT

Assembly Programming and Instruction of 8051

CONTENTS

Part-1 : Introduction to 8051 Assembly Programming, Assembling and Running an 8051 Program, Data Types and Assembler Directives ..... 5–2B to 5–7B

Part-2 : Arithmetic, Logic Instruction and Programs, Jump, Loop and Call Instructions ..... 5–7B to 5–11B

Part-3 : I/O Port Programming, Programming 8051 Timers, Serial Port Programming, Interrupts Programming ..... 5–11B to 5–22B

Part-4 : Interfacing : LCD & Keyboard Interfacing, ADC, DAC and Sensor Interfacing ..... 5–22B to 5–29B

Part-5 : External Memory Interface, Stepper Motor and Waveform Generation ..... 5–29B to 5–32B

PART-1

Introduction to 8051 Assembly Programming, Assembling and Running An 8051 Program, Data Types and Assembler Directives.

CONCEPT OUTLINE

- The widely used directives of 8051 are :

1. ORG (Origin)
2. EQU (Equate)
3. END directive.

- Assembly language is referred to as a low level language because it deals directly with the internal structure of CPU.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.1. **Write a short note on assembly language programming. Explain it with an example.**

OR

What do you mean by assembler ? Explain machine language, low level language and high level language.

Answer

A. Machine language :

1. In the early days of the computer, programmers coded programs in machine language. A program that consists of 0s and 1s is called machine language.
2. Although hexadecimal system was used as an efficient way to represent binary numbers, the process of working in machine code was still cumbersome for humans.
3. Eventually, assembly languages were developed that provided mnemonics for the machine code instructions, plus other features that made programming faster and less prone to error.
4. The term mnemonic is frequently used in computer science to refer codes and abbreviations that are relatively easy to remember.

B. Assembler : Assembly language programs must be translated into machine code by a program called an assembler.

C. Low-level language :

1. Assembly language is referred to as low-level language because it deals directly with the internal structure of the CPU.
2. To program in assembly language, the programmer must know all the registers of the CPU and the size of each.

D. High-level language :

1. Today, one can use many different programming languages, such as BASIC, C, C++, Java, and numerous others.
2. These languages are called high-level languages because the programmer does not have to be concerned with the internal details of the CPU whereas an assembler is used to translate an assembly language program into machine code (sometimes also called object code or opcode for operation code).
3. High-level languages are translated into machine code by a program called a compiler.

E. Assembly language programming :

1. An assembly language program consists of a series of lines of assembly language instructions.
2. These instructions consist of mnemonic, optionally followed by one or two operands. The operands are the data items being manipulated, and the mnemonics are the commands to the CPU.
3. An assembly language instruction consists :

   [Label :] mnemonic [Operands] [; Comment]

   i. The label field allows the program to refer to a line of code by name. The label field cannot exceed a certain number of characters.

   ii. The assembly language mnemonic (instruction) and operand(s) fields together perform the real work of the program and accomplish the tasks for which the program was written.

   For example :

   ADD *A*, *B*

   MOV A, #67

   iii. The comment field begins with semicolon comment indicator ";". Comments may be at the end of a line or on a line by themselves.

4. An assembly language program given below is a series of statements, or lines, which are either assembly language instructions such as ADD and MOV, or statements called directives.
5. Here instructions tell the CPU what to do and directives (also called pseudo-instructions) give directions to the assembler.
6. For example, in the given program the MOV and ADD instructions are commands to the CPU and ORG, END are directives to the assembler.

7. ORG tells the assembler to place the opcode at memory location 0 while END indicates to the assembler the end of the source code. In other words, one is for the start of the program and the other one for the end of the program.

For example :

```
        ORG   0 H          ; start (origin) at location 0
        MOV   R5, #25 H    ; load 25 H into R5
        MOV   R7, #34 H    ; load 34 H into R7
        MOV   A, #0        ; load 0 into A
        ADD   A, R5        ; add contents of R5 to A
                           ; now A = A + R5
        ADD   A, R7        ; add contents of R7 to A
                           ; now A = A + R7
        ADD   A, #12 H     ; add A to value 12 H
                           ; now A = A + 12 H
HERE :  SJMP HERE          ; stay in this loop
        END                ; end of asm source file
```

Que 5.2. **How an assembly language program is created, assembled and made ready to run ?**

OR

Describe in brief about "asm", "obj" and "lst" file.

Answer

1. The steps that illustrate how an assembly language program is created, assembled and made ready to run is given in the Fig. 5.2.1.

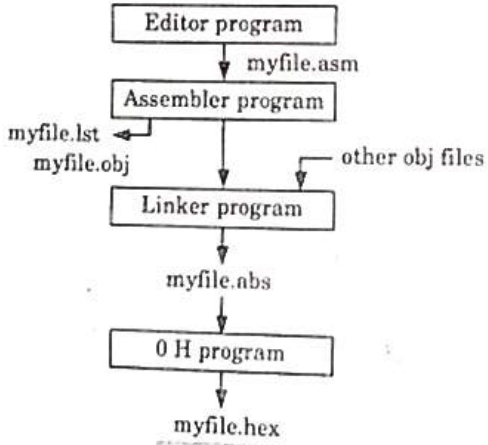


Fig. 5.2.1.

Steps 1 :

i. Firstly an editor is used to create and/ or edit the program. A widely used editor is the MS-DOS edit program (or notepad in windows), which comes with all Microsoft operating systems.

ii. The editor must be able to produce an ASCII file. For many assemblers, the file names follow the usual DOS convention, but the source file has the extension "asm" or "src", depending on which assembler you are using. Check your assembler for the convention.

iii. The "asm" extension for the source file is used by an assembler in the next step.

Steps 2 : The "asm" source file containing the program code created in step 1 is fed to an 8051 assembler. The assembler converts the instructions into machine code. The assembler will produce an object file and a list file. The extension for the object file is "obj" while the extension for the list file is "lst".

Steps 3 : Assemblers require a third step called linking. The link program takes one or more object files and produces an absolute object file with the extension "abs". This abs file is used by 8051 trainers that have a monitor program.

Steps 4 : Next the "abs" file is fed into a program called "0 H" (object to hex converter), which creates a file with extension "hex" that is ready to burn into ROM. This program comes with all 8051 assemblers.

asm and obj files :

1. The "asm" file is also called the source file and for this reason some assemblers require that this file have the "src" extension.
2. The 8051 assembler converts the asm file's assembly language instructions into machine language and provides the obj (object) file.

lst (list) file :

1. The lst (list) file, which is optional, is very useful to the programmer because it lists all the opcodes and addresses as well as errors that the assembler detected.
2. Many assemblers assume that the list file is not wanted unless you indicate that you want to produce it.
3. This file can be accessed by an editor such as DOS EDIT and displayed on the monitor or sent to the printer to produce a hard copy.
4. The programmer uses the list file to find syntax errors.
5. It is only after fixing all the errors indicated in the lst file that the obj file is ready to be input to the linker program.

Que 5.3. **How does an instruction differ from directive ? Discuss the different types of assembler directives of 8051. Why are the ORG and END directives also called pseudocode ?**

Answer

A. Difference :

S. No.	Instruction	Directive
1.	An instruction consists of a mnemonic, optionally followed by one or two operands. It basically tells the CPU what to do.	Directives give the directions to the assembler.
2.	The instructions are translated into machine code (opcode) for the CPU to execute.	Directives do not generate any machine code and are used only by the assembler.

B. Types of assembler directive :

i. ORG (Origin) :

1. The ORG directive is used to indicate the beginning of the address. The number that comes after ORG can be either in hex or in decimal.
2. If the number is not followed by H, it is decimal and the assembler will convert it to hex.

ii. EQU (Equate) :

1. This is used to define a constant without occupying a memory location.
2. The EQU directive does not set aside storage for a data item but associates a constant value with a data label so that when the label appears in the program; its constant value will be substituted for the label.
3. The following uses EQU for the counter constant and then the constant is used to load the R3 register.

```
COUNT    EQU 25
...      ....
MOV      R3, # COUNT
```

4. When executing the instruction "MOV R3, #COUNT", the register R3 will be loaded with the value 25.

iii. END directive :

1. Another important pseudocode is the END directive. This indicates to the assembler the end of the source (asm) file.
2. The END directive is the last line of an 8051 program, meaning that in the source code anything after the END directive is ignored by the assembler.

C. Reason :

1. Pseudocodes are the codes that instruct the assembler in doing its job. ORG and END directives are called pseudocodes because they are directives to the assembler.

2. ORG tells the assembler to place the opcode at memory location 0 while END indicates to the assembler, the end of the source code.

PART-2

Arithmetic, Logic Instruction and Programs, Jump, Loop and Call Instructions.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.4. **Explain arithmetic instruction with example.**

Answer

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

1. **ADD :** This instruction is used to add two operands. The destination operand is always in register *A* while the source operand can be a register, immediate data or in memory.

   Format : ADD *A*, Source

   For example :

   ADD *A*, *R*2 ; Adds contents of *A* and *R*2 and store result in *A*.

   ADD *A*, # 20 H ; Add contents of *A* and 20 H and store result in *A*.

2. **ADDC (Add with carry) :** This instruction is used to add two 16-bit operands. In this, add register, immediate data, or in memory to accumulator with carry and store result in *A*.

   Format : ADDC *A*, source

   For example :

   ADDC *A*, *R*2 ; Adds the contents of *A*, *R*2 and carry flag, and stores result in *A*.

   ADDC *A*, # 50 H ; Adds the contents of *A* and carry flag and 20 H and stores result.

3. **DA :**

i. The DA (decimal adjust for addition) instruction in the 8051 is provided to correct the aforementioned problem associated with BCD addition. The mnemonic "DA" has as its only operand the accumulator "A".

ii. The DA instruction will add 6 to the lower nibble or higher nibble if needed; otherwise, it will leave the result alone.

Format : DA A

For example :

```
MOV  A, #47 H   ; A = 47 H first BCD operand
MOV  B, #25 H   ; B = 25 second BCD operand
ADD  A, B       ; hex (binary) addition (A = 6C H)
DA   A          ; adjust for BCD addition (A = 72 H)
```

4. **SBB (Subtract with borrow) :** This instruction is used for multibyte numbers and will take care of the borrow of the lower operand. If CY = 1 prior to executing the SUBB instruction, it also subtracts 1 from the result.

Format : SUBB *A*, source

For example :

SUBB *A*, *R*3 : Subtracts contents of *R*3 and carry together from *A* and stores result in *A*.

SUBB A, # 20 H : Subtracts 20 H from *A* and storage result in *A*.

5. **MUL :** In this multiplication, one of the operands must be in register *A*, and the second operands must be in register *B*. After multiplication, the result is in the *A* and *B* register; the lower byte is in *A*, and the upper byte is in *B*.

Format : MUL AB

For example :

```
MOV  A, # 25 H   ; load 25 H to reg. A
MOV  B, # 65 H   ; load 65 H to reg. B
MUL  AB          ; 25 H * 65 H = E99 where
                 ; B = 0E H and A = 99 H
```

6. **DIV :** In this, the numerator must be in register A and the denominator must be in B. After the DIV instruction is performed, the quotient is in A and the remainder is in B.

Format : DIV AB

For example :

```
DIV  AB          ; now A = quotient and B = remainder
```

Que 5.5. **Write a program to add two 16-bit numbers the numbers are FC45 H and 02EC H.**

Answer

```
CLR  C          ; make CY = 0
MOV  A, #45 H   ; load the low byte into A
ADD  A, #0EC H  ; add the low byte, now A = 31, CY = 1
```



```
MOV  R0, A      ; save the low byte of sum in R0
MOV  A, #02 H   ; load the high byte into A
ADDC A, #0FC H  ; add the high bytes with carry
                ; 02 + FC H + 1 = FF H
MOV  R1, A      ; save the high byte of result in R1
```

Finally, we get the result as R0 = 31 H and R1 = FF H

Que 5.6. **What do you mean by logical instructions ?**

Answer

The logical instructions are :

1. **AND :** This instruction will perform a logical AND on the two operands and place the result in the destination. The destination is normally the accumulator. The source operand can be a register, in memory, or immediate.

Format : ANL destination, source.

For example :

ANL A, R2 ;

2. **OR :** The destination and source operands are ORed, and the result is placed in the destination. The ORL instruction can be used to set certain bits of operands to 1. The destination is normally the accumulator. The source operand can be a register, in memory, or immediate.

Format : ORL destination, source

For example : ORL C, ACC 7 ;

3. **XOR :** This instruction will perform the XOR operation on the two operands, and place the result in the destination. The destination is normally the accumulator. The source operand can be a register, in memory, or immediate.

Format : XRL destination, source

For example : XRL *A*, # 78 H

4. **CPLA (complement accumulator) :** This instruction complements the contents of register A. The complement action changes the 0s to 1s and the 1s to 0s. This is also called 1's complement.

Format : CPL *A*

For example : MOV A, # 55H

CPL A ; now A = AA H

Que 5.7. **Explain the following instructions :**

i. **Jump instructions.**
ii. **Loop instructions.**
iii. **Call instructions.**

Answer

i. **Jump instruction :**

a. **JZ (jump if $A = 0$) :** In this instruction the content of register A is checked. If it is zero, it jumps to the target address.

b. **JNC (Jump if no carry) :**

1. In this instruction, the carry flag bit in the flag (PSW) register is used to make the decision whether to jump.
2. In executing "JNC label", the processor looks at the carry flag to see if it is raised (CY = 1).
3. If it is not, the CPU starts to fetch and execute instructions from the address of the label. If CY = 1, it will not jump but will execute the next instruction below JNC.

c. **LJMP (Long jump) :**

1. LJMP is an unconditional long jump. It is a 3-byte instruction in which the first byte is the opcode, and the second and third bytes represent the 16-bit address of the target location.
2. The 2 byte target address allows a jump to any memory location from 0000 to FFFF H.

d. **SJMP (Short jump) :**

1. In this 2-byte instruction, the first byte is the opcode and the second byte is the relative address of the target location.
2. The relative address range of 00 = FF H is divided into forward and backward jumps; that is, within – 128 to + 127 bytes of memory relative to the address of the current PC.
3. If the jump is forward, the target address can be within a space of 127 bytes from the current PC.
4. If the target address is backward, the target address can be within – 128 bytes from the current PC.

ii. **Loop instructions :**

1. Repeating a sequence of instructions a certain number of times is called a loop. The loop is one of most widely used actions that any microprocessor performs.
2. In the 8051, the loop action is performed by the instruction "DJNZ reg, label".
3. In this instruction, the register is decremented; if it is not zero, it jumps to the target address referred to by the label.
4. Prior to the start of the loop the register is loaded with the counter for the number of repetitions.

iii. **CALL instruction :**

1. Call instruction is used to call a subroutine. Subroutines are often used to perform tasks that need to be performed frequently.
2. In the 8051 there are two instructions for call : LCALL (long call) and ACALL (absolute call).

a. **LCALL (Long call) :**

1. In this 3-byte instruction, the first byte is the opcode and the second and third bytes are used for the address of the target subroutine.
2. Therefore, LCALL can be used to call subroutines located anywhere within the 64 Kbyte address space of the 8051.

b. **ACALL (Absolute call) :**

1. ACALL is a 2-byte instruction in contrast to LCALL, which is 3 bytes.
2. Since ACALL is a 2-byte instruction, the target address of the subroutine must be within 2 Kbytes because only 11 bits of the 2 bytes are used for the address.
3. There is no difference between ACALL and LCALL in terms of saving the program counter on the stack or the function of the RET instruction.
4. The only difference is that the target address for LCALL can be anywhere within the 64 Kbytes address space of the 8051 while the target address of ACALL must be within a 2 Kbyte range.

PART-3

I/O Port Programming, Programming 8051 Timers, Serial Port Programming, Interrupts Programming.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.8. **Write a test program for the DS89C420/30 chip to toggle all the bits of P0, P1 and P2 every 1/4 of a second. Assume a crystal frequency of 11.0592 MHz.**

Answer

```
; Tested for the DS89C420/30 with XTAL = 11.0592 MHz.
        ORC     0
BACK:   MOV     A, #55 H
        MOV     P0, A
        MOV     P1, A
```

```
        MOV     P2, A
        ACALL   QSDELAY     ; Quarter of a second delay
        MOV     A, # 0AAH
        MOV     P0, A
        MOV     P1, A
        MOV     P2, A
        ACALL   QSDELAY
        SKMP    BACK
; ----------1/4 SECOND DELAY
QSDELAY :
        MOV     R5, #11
H3 :    MOV     R4, #248
H2 :    MOV     R3, #255
H1 :    DJNZ    R3, H1      ; 4 MC for DS89C4x0
        DJNZ    R4, H2
        DJNZ    R5, H3
        RET
        END
```

Delay = 11 × 248 × 255 × 4 MC × 90 ns = 250,430 μs

Use an oscilloscope to verify the delay size.

Que 5.9. **A switch is connected to pin P1.7. Write a program to check the status of SW and perform the following :**

a. If SW = 0, send letter *N* to P2.

b. If SW = 1, send letter *Y* to P2.

Answer

```
        SETB    P1 . 7      ; make P1.7 an input
AGAIN   JB      P1.2 OVER   ; jump if P1.7 = 1
        MOV     P2, # N     ; SW = 0, issue N to P2
        SJMP    AGAIN       ; Keep monitoring
OVER :  MOV     P2, # Y     ; SW = 1, issue Y to P2
        SJMP    AGAIN       ; keep monitoring
```

Que 5.10. **How many timers do 8051 microcontrollers have ?**

Answer

The 8051 has two timers :

A. Timer 0 register :

1. This is a 16-bit register accessed as low byte and high byte.

2. The low byte register is called TL0 (Timer 0 low byte) and high byte register is referred as TH0 (Timer 0 high byte).

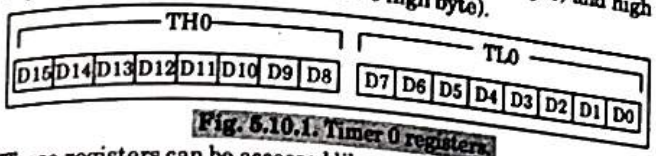


Fig. 5.10.1. Timer 0 registers.

3. These registers can be accessed like any other register such as *A*, *B*, R0, R1, R2 etc.

For example :

MOV TL0, # 4F H ; Moves the value 4F H into TL0, the low byte of timer 0.

MOV R5, TH0 ; Saves the TH0 (high byte of timer 0) in R5.

B. Timer 1 register :

1. It is also 16-bit register which is also accessed as low byte and high byte.
2. The low byte register is called TL1 (Timer 1 low byte) and high byte register is called TH1 (Timer 1 high byte).
3. These registers are accessible in same way as the registers of Timer 0.

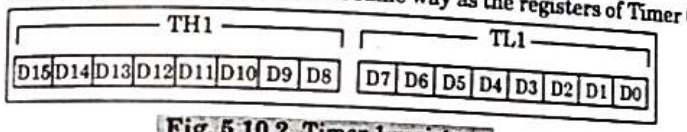


Fig. 5.10.2. Timer 1 registers.

Que 5.11. **Explain the function of each bit of TMOD register of 8051.**

Answer

1. TMOD register is a timer mode register and both timers (0 and 1) use this register to set the various timer operation modes.
2. It is an 8-bit register in which lower 4-bits are for Timer 0 and upper 4-bits for Timer 1. In each case, the lower two bits are used to set the timer mode and upper two bits are used to specify the operation.

MSB LSB

GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer 1				Timer 0			

Fig. 5.11.1. TMOD register.

Table 5.11.1.

M1	M0	Mode	Operating Mode
0	0	0	13-bit timer mode 8-bit timer/counter THx with TLx as 5-bit prescaler
0	1	1	16-bit timer mode 16-bit timer/counters THx and TLx are cascaded; there is no prescaler
1	0	2	8-bit auto reload 8-bit auto reload timer/counter; THx holds a value that is to be reloaded into TLx each time it overflows.
1	1	3	Split timer mode

M1 and M0 :

1. M0 and M1 select the timer mode. There are basically three modes : 0, 1 and 2.
2. Mode 0 is a 13-bit timer, mode 1 is a 16-bit timer and mode 2 is an 8-bit timer. The most widely used modes are 1 and 2.

C/T (Clock/Timer) :

1. This bit in the TMOD register is used to decide whether the timer is used as a delay generator or an event counter.
2. If C/T = 0, it is used as a timer for time delay generation and the crystal frequency attached to 8051 act as source of the clock for the timer.
3. This means that the size of the crystal frequency attached to 8051 also decides the speed at which 8051 timer ticks.
4. The frequency for the timer is always $1/12^{th}$ the frequency of the crystal attached to the 8051.

GATE :

1. In the TMOD register both timers 0 and 1 have the GATE bit. It is because every timer has a means of starting and stopping.
2. Some timers do this by software, some by hardware, and some have both software and hardware controls. The timers in the 8051 have both.
3. The start and stop of the timer are controlled by way of software by the TR (timer start) bit TR0 and TR1.
4. This is achieved by the instructions "SETB TR1" and "CLR TR1" for Timer 1, and "SETB TR0" and "CLR TR0" for Timer 0.
5. The SETB instruction starts it, and it is stopped by the CLR instruction. These instructions start and stop the timers as long as GATE = 0 in the TMOD register.

6. The hardware way of starting and stopping the timer by an external source is achieved by making GATE = 1 in the TMOD register.

Que 5.12. **Explain the structure of TCON register.**

Answer

Structure of TCON register :

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Fig. 5.12.1.

Table 5.12.1.

Bit	Symbol	Function
7	TF1	Timer 1 overflow flag. Set when timer rolls from all 1s to 0. Cleared when processor vectors to execute interrupt service routine located at program address 001B H.
6	TR1	Timer 1 run control bit. Set to 1 by program to enable timer to count; cleared to 0 by program to halt timer. Does not reset timer.
5	TF0	Timer 0 overflow flag. Set when timer rolls from all 1s to 0. Cleared when processor vectors to execute interrupt service routine located at program address 000B H.
4	TR0	Timer 0 run control bit. Set to 1 by program to enable timer to count; cleared to 0 by program to halt timer. Does not reset timer.
3	IE1	External interrupt 1 edge flag. Set to 1 when a high-to-low edge signal is received on port 3 pin 3.3 ($\overline{INT1}$). Cleared when processor vectors to interrupt service routine located at program address 0013 H. Not related to timer operations.
2	IT1	External interrupt 1 signal type control bit. Set to 1 by program to enable external interrupt 1 to be triggered by a falling edge signal. Set to 0 by program to enable a low-level signal on external interrupt 1 to generate an interrupt.

1	IE0	External interrupt 0 edge flag. Set to 1 when a high-to-low edge signal is received on port 3 pin 3.2 ($\overline{INT0}$). Cleared when processor vectors to interrupt service routine located at program address 0003H. Not related to timer operations.
0	IT0	External interrupt 0 signal type control bit. Set to 1 by program to enable external interrupt 0 to be triggered by a falling edge signal. Set to 0 by program to enable a low-level signal on external interrupt 0 to generate an interrupt.

Table 5.12.2. Equivalent instructions for the Timer control register (TCON)

For Timer 0				
SETB	TR0	=	SETB	TCON.4
CLR	TR0	=	CLR	TCON.4
SETB	TF0	=	SETB	TCON.5
CLR	TF0	=	CLR	TCON.5
For Timer 1				
SETB	TR1	=	SETB	TCON.6
CLR	TR1	=	CLR	TCON.6
SETB	TF1	=	SETB	TCON.7
CLR	TF1	=	CLR	TCON.7

Que 5.13. **a. Find the values of TMOD to operate as timers in the following modes :**

i. Mode 1 Timer 1

ii. Mode 0 Timer 0

b. Indicate the selection made in the instruction "MOV TMOD, #20 H".

Answer

a.

i. Mode 1 Timer 1 :

Gate	C/T	M1	M0	Gate	C/T	M1	M0
0	0	0	1	0	0	0	0
Timer 1				Timer 0			

Fig. 5.13.1.

Gate control bit and C/T bits are made 0, and unused timer is also made 0.

For mode 1, M1 = 0 and M0 =1

∴ TMOD = 00010000 = 10 H

ii. Mode 0 Timer 0 :

Gate	C/T	M1	M0	Gate	C/T	M1	M0
0	0	0	0	0	0	0	0
Timer 1				Timer 0			

Fig. 5.13.2.

For mode 0, M0 = 0 and M1 = 0

∴ TMOD = 00000000 = 00 H

b. 20 H = 00100000

This indicates that Timer 0 bit is 0 and M1 = 1, M0 = 0

∴ It is timer 1, mode 2.

Que 5.14. **Generate a square wave with an ON time of 5 ms and an OFF time of 5 ms on all pins of port 0. Assume an XTAL of 12 MHz.**

Answer

1. For crystal frequency of 12 MHz :

The value of instruction cycle = $\frac{1}{12} \times 12$ MHz = 1 MHz

$$\therefore \quad T = \frac{1}{1 \times 10^6} = 1\ \mu s$$

2. **For OFF time calculation :**

$5\ ms/10^{-6} = 5000$ cycle

Now, perform 65536 – n, where n is the decimal value given above.

∴ 65536 – 5000 = 60536 = EC78 H

3. **For ON time calculation :**

$5\ ms/10^{-6}\ \mu s = 5000$ cycle

∴ 65536 – 5000 = 60536 = EC78 H.

4. Let us use Timer 0 in Mode 1.

```
        MOV TMOD, # 01 H     ; Timer 0 in mode 1
BACK :  MOV TL0, # 078 H     ; to generate the OFF time, load TL0
        MOV TH0, # 0EC H     ; load OFF time value in TH0
```

```
        MOV P0, # 00 H          ; make port bits low
        ACALL DELAY             ; call delay routine
        MOV TL0, # 078 H        ; to generate ON time, load TL0
        MOV TH0, # 0EC H        ; load ON time value in TH0
        MOV P0, # 0FF H         ; make port bits high
        ACALL DELAY             ; call delay
        SJMP BACK               ; repeat for reloading counters to
                                  get a continuous square wave
        ORG 300 H
DELAY : SETB TR0                ; start the counter
AGAIN : JNB TF0, AGAIN          ; check the timer overflow
        CLR TR0                 ; when TF0 is set, stop the timer
        CLR TF0                 ; clear timer flag
        RET
        END                     ; end of file.
```

Que 5.15. **Timer-0 of 8051 microcontroller is to be programmed in mode-1 for creating a square wave of duty cycle 50 % on the port P1. 5. Write an algorithm for programming the counter.**

AKTU 2017-18, Marks 10

Answer

A. Program :

```
        MOV     TMOD, #01       ; Timer 0, mode 1(16-bit mode)
HERE :  MOV     TL0, #0F2 H     ; TL0 = F2 H, the low byte
        MOV     TH0, #0FF H     ; TH0 = FF H, the high byte
        CPL     P1.5            ; toggle P1.5
        ACALL   DELAY
        SJMP    HERE            ; load TH, TL Again
Delay   using Timer 0
        DELAY
        SET B   TR0             ; start Timer 0
AGAIN   JNB     TF0, AGAIN      ; Monitor Timer 0 flag until
                                ; it roll over
        CLR     TR0             ; stop timer 0
        CLR     TF0             ; clear timer 0 flag
        RET
```

B. Algorithm :

1. TMOD is loaded.
2. FFF2 H is loaded into TH0 - TL0.
3. P1.5 is toggled for the high and low portions of the pulse.
4. The DELAY subroutine using the timer is called.

5. In the DELAY subroutine, Timer 0 is started by the SETB TR0 instruction.

i. Timer 0 counts up with the passing of each clock, which is provided by the crystal oscillator. As the timer counts up, it goes through the states of FFF3, FFF4, FFF5, FFF6, FFF7, FFF8, FFF9, FFFA, FFFB, and so on until it reaches FFFF H.

ii. One more clock rolls it to 0, raising the timer flag (TF0 = 1). At that point, the JNB instruction falls through.

iii. Timer 0 is stopped by the instruction CLR TR0. The DELAY subroutine ends and the process is repeated.

Que 5.16. **Explain how serial communication takes place in 8051. State special function registers used for serial communication in 8051 microcontroller.**

Answer

1. The serial port of 8051 is full duplex, means it can transmit and receive simultaneously. It uses register SBUF to hold data. Register SCON controls data communication, register-PCON controls data rates and pin RxD (P3.0) and TxD (P3.1) do the data transfer.
2. SBUF is an 8-bit register dedicated for serial communication in 8051. Its address is 99 H. It can be addressed like any other register in 8051.
3. Writing to SBUF loads data to be transmitted and reading SBUF accesses received data.
4. There are two separate and distinct register, the transmit write-only register, and the receive read-only register. This is shown in Fig. 5.16.1.
5. The way in which SBUF is used for the transmission and reception of the data during serial communication is explained below :

a. **Transmission :** When a byte of data is to be transmitted via the TxD pin, the SBUF is loaded with this data byte. As soon as a data byte is written into SBUF, it is framed with the start and stop bits and transmitted serially via the TxD pin.

b. **Reception :** When 8051 receives data serially via RxD pin of it, the 8051 deframes it. The start and stop bits are separated out from a byte of data. This type is placed SBUF register.

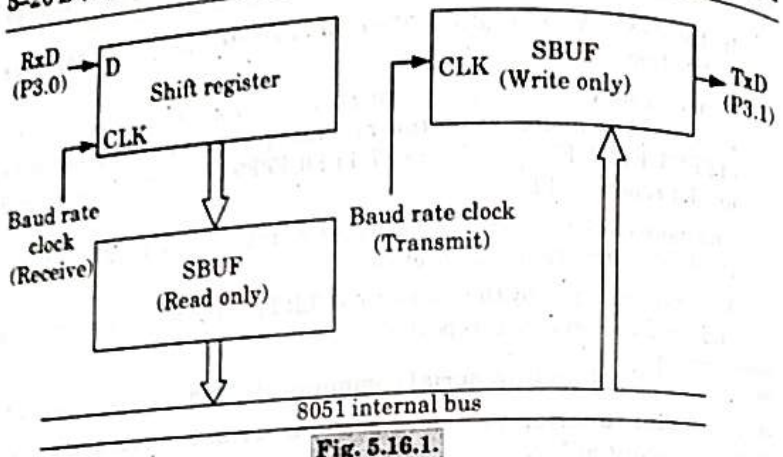


Fig. 5.16.1.

Que 5.17. **Explain the bit pattern of SCON register of 8051.**

Answer

1. The 8051 provides four programmable modes for serial data communication. A particular mode can be selected by setting the SM0 and SM1 bits in SCON. The mode selection also decides the baud rate.
2. The Fig. 5.17.1 shows the bit patterns for SCON.

7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	T1	RI

Fig. 5.17.1.

Symbol	Position	Name and Significance
SM0	SCON.7	Serial port mode control bit 0. Set/cleared by software.
SM1	SCON.6	Serial port mode control bit 1. Set/cleared by software.
SM2	SCON.5	Serial port mode control bit 2. Set by software to disable reception of frame for which bit 8 is zero.
REN	SCON.4	Receiver enable control bit. Set/cleared by software to enable/disable serial data reception.
TB8	SCON.3	Transmit Bit 8. Set cleared by hardware to determine state of ninth data bit transmitted in 9-bit UART mode.

RB8	SCON.2	Receive Bit 8. Set/cleared by hardware to indicate state of ninth data bit received
TI	SON.1	Transmit Interrupt flag. Set by hardware when byte transmitted. Cleared by software after servicing.
RI	SCON.0	Receive Interupt flag. Set by hardware when byte received. Cleared by software after servicing.

Que 5.18. **Draw and explain the bit pattern of PCON register of 8051.**

Answer

Bit pattern of PCON register :

7	6	5	4	3	2	1	0
SMOD	-	-	-	GF1	GF0	PD	IDL

Fig. 5.18.1.

Symbol	Position	Name and Significance
SMOD	PCON.7	Serial baud rate modify bit. It is 0 at reset. It is set to 1 by program to double the baud rate.
–	PCON.6	Not defined.
–	PCON.5	Not defined.
–	PCON.4	Not defined.
GF1	PCON.3	General purpose user flag bit 1. Set/cleared by program.
GF0	PCON.2	General purpose user flag bit 0. Set/cleared by program.
PD	PCON.1	Power down bit. It is set to 1 by program to enter power down configuration for CHMOS microcontrollers.
IDL	PCON.0	Idle mode bit. It is set to 1 by program to enter idle mode configuration for CHMOS microcontrollers.

Que 5.19. **How many interrupts in 8051 ? Also draw interrupt vector table for the 8051.**

Answer

The six interrupts in the 8051 shown in Table 5.19.1 :

Table 5.19.1 : Interrupt vector table

S. No.	Interrupt	ROM Location (Hex)	Pin
1.	Reset	0000	9
2.	External hardware interrupt 0 (INT0)	0003	P3.2(12)
3.	Timer 0 interrupt (TF0)	000B	
4.	External hardware interrupt 1 (INT1)	0013	P3.3(13)
5.	Timer 1 interrupt (TF1)	001B	
6.	Serial COM interrupt (RI and TI)	0023	

1. Reset : When the reset pin is activated, the 8051 jumps to address location 0000.
2. Two interrupts are set aside for the timers : one for timer 0 and one for timer 1. Memory locations 000B H and 001B H in the interrupt vector table belong to timer 0 and timer 1, respectively.
3. Two interrupts are set aside for hardware external hardware interrupts. Pin numbers 12 (P3.2) and 13(P3.3) in port 3 are for the external hardware interrupts INT0 and INT1 respectively. These external interrupts are also referred to as EX1 and EX2.
4. Serial communication has a single interrupt that belongs to both receive and transmit. The interrupt vector table location 0023H belongs to this interrupt.

PART-4

Interfacing : LCD & Keyboard Interfacing, ADC, DAC and Sensor Interfacing.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.20. **What do you mean by LCD module ? Explain the interfacing of LCD module with 8051.**

Answer

A. LCD module :

1. Many LCD modules are available which have built-in drivers for LCD and interfacing circuitry to interface them to microprocessor/microcontroller systems.
2. These LCD modules allow display of characters as well as number. They are available in 16 × 2, 20 × 1, 20 × 2, 20 × 4 and 40 × 2 sizes.
3. The main advantage of using LCD modules is that they require very less power.

B. Interfacing of LCD module :

1. The Fig. 5.20.1 shows the interfacing of a 20 character × 2 line LCD module with the 8051.
2. As shown in the Fig. 5.20.1, the data lines are connected to the port 1 of 8051 and control lines of RS, R/W and *E* are driven by 3.2, 3.3 and 3.4 lines of port 3, respectively.
3. The voltage at V_{EE} pin is adjusted by a potentiometer to adjust the contrast of the LCD.

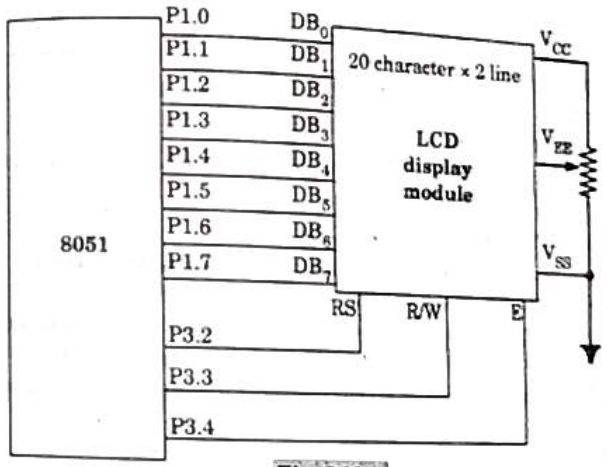


Fig. 5.20.1.

Que 5.21. **Give the pin description of an LCD.**

Answer

Pin description : The LCD module consists of 14 pins. Each pin's description is shown in the table 5.21.1.

Table 5.21.1 : Pin descriptions for LCD

Pin	Symbol	I/O	Description
1	V_{SS}	–	Ground
2	V_{CC}	–	+ 5 V power supply
3	V_{EE}	–	Power supply to control contrast
4	RS	I	RS = 0 to select command register, RS = 1 to select data register
5	R/W	I	R/W = 0 for write, R/W = 1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8-bit data bus
8	DB1	I/O	The 8-bit data bus
9	DB2	I/O	The 8-bit data bus
10	DB3	I/O	The 8-bit data bus
11	DB4	I/O	The 8-bit data bus
12	DB5	I/O	The 8-bit data bus
13	DB6	I/O	The 8-bit data bus
14	DB7	I/O	The 8-bit data bus

i. V_{CC}, V_{SS} and V_{EE}:

V_{CC} and V_{SS} provide + 5 V and ground respectively whereas V_{EE} is used for controlling LCD contrast.

ii. RS (Register select) :

1. There are two very important registers inside the LCD. The RS pin is used for their selection as follows.
2. If RS = 0, the instruction command code register is selected, allowing the user to send a command such as clear display, cursor at home, etc.
3. If RS = 1 the data register is selected, allowing the user to send data to be displayed on the LCD.

iii. R/W (Read/write) :

R/W input allows the user to write information to the LCD or read information from it. R/W = 1 when reading; R/W = 0 when writing.

iv. E (Enable) :

1. The enable pin is used by the LCD to latch information presented to its data pins.
2. When data is supplied to data pins, a high-to-low pulse must be applied to this pin in order for the LCD to latch in the data present at the data pins. This pulse must be a minimum of 450 ns wide.

v. D0-D7 :

1. The 8-bit data pins, D0-D7, are used to send information to the LCD or read the contents of the LCD's internal registers.
2. To display letters and numbers, we send ASCII codes for the letters A-Z, a-z, and numbers 0-9 to these pins while making RS = 1.
3. There are also instruction command codes that can be sent to the LCD to clear the display or force the cursor to the home position or blink the cursor.
4. RS = 0 is used to check the busy flag bit to see if the LCD is ready to receive information.
5. The busy flag is D7 and can be read when R/W = 1 and RS = 0, as follows : if R/W = 1, RS = 0. When D7 = 1 (busy flag = 1), the LCD is busy taking care of internal operations and will not accept any new information.
6. When D7 = 0, the LCD is ready to receive new information.

Que 5.22. Discuss the key press and key detection mechanism of keyboard.

Answer

1. Keyboards are organized in a matrix of rows or columns.
2. The CPU accesses both rows and columns through ports; thus, with two 8-bit ports, an 8 × 8 matrix of keys can be connected to a microprocessor. When a key is pressed, a row and a column make a contact.
3. To understand the mechanism let us consider a 4 × 4 matrix connected to two ports as shown in Fig. 5.22.1.

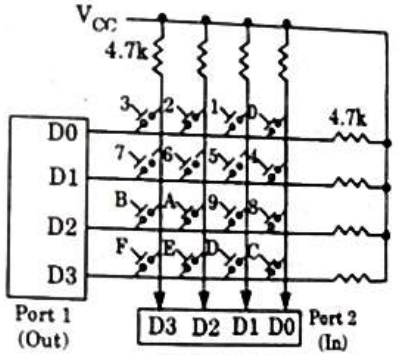


Fig. 5.22.1.

4. The rows are connected to an output port and the columns are connected to an input port.
5. If no key has been pressed, the input port will yield 1s for all columns since they are all connected to high (V_{CC}).

6. If all the rows are grounded and a key is pressed, one of the columns will have 0 since the key pressed provides the path to ground.
7. It is the function of the microcontroller to scan the keyboard continuously to detect and identify the key pressed.
8. To detect a pressed key, the microcontroller ground all rows by providing 0 to the output latch, then it reads the columns. If the data read from the columns is D3-D0 = 1111, no key has been pressed and process continues until a key press is detected.
9. However, if one of the column bits has a zero, this means that a key press has occurred.
10. For example, if D3-D0 = 1101, this means that a key in the D1 column has been pressed. After a key press is detected, the microcontroller will go through the process of identifying the key.
11. Starting with the top row, the microcontroller grounds it by providing a low to row D0 only; then it reads the columns.
12. If the data read is all 1s, no key in that row is activated and the process is moved to the next row. It grounds the next row, reads the columns, and checks for any zero. This process continues until the row is identified.
13. After identification of the row in which the key has been pressed the next task is to find out which column the pressed key belongs to.
14. This should be easy since the microcontroller knows at any time which row and column are being accessed.

Que 5.23. **Write a keyboard program.**

Answer

```
        MOV     P2, #0FFH          ; make P2 an input port
K1 :    MOV     P1, #0             ; ground all rows at once
        MOV     A, P2              ; read all col. ensure all keys open
        ANL     A, #00001111B      ; masked unused bits
        CJNE    A, #00001111B, K1  ; check till all keys released
K2 :    ACALL   DELAY              ; call 20 ms delay
        MOV     A, P2              ; see if any key is pressed
        ANL     A, #00001111B      ; mask unused bits
        CJNE    A, #00001111B,OVER ; key pressed, await closure
        SJMP    K2
OVER:   ACALL   DELAY              ; check if key pressed
        MOV     A, P2              ; wait 20 ms debounce time
        ANL     A, #00001111B      ; check key closure
        CJNE    A, #00001111B,OVER1 ; mask unused bits
        SJMP    K2                 ; key pressed, find row
OVER1 : MOV     P1, #11111110B     ; if none, keep polling
        MOV     A, P2              ; ground row 0
        ANL     A, #00001111B      ; read all columns
        CJNE    A, #00001111B, ROW 0 ; mask unused bits
        MOV     P1, #11111101B     ; key row 0, find the col.
        MOV     A, P2              ; ground row 1
        ANL     A, #00001111B      ; read all columns
                                   ; mask unused bits
```



```
        CJNE    A, #00001111B, ROW 1 ; key row 1, find the col.
        MOV     P1, #11111011B     ; ground row 2
        MOV     A, P2              ; read all columns
        ANL     A, #00001111B      ; mask unused bits
        CJNE    A, #00001111B, ROW 2 ; key row, find the col.
        MOV     P1, #11110111B     ; ground row 3
        MOV     A, P2              ; read all columns
        ANL     A, #00001111B      ; mask unused bits
        CJNE    A, #00001111B, ROW 3 ; key row 3, find the col.
        LJMP    K2                 ; if none, false input, repeat
ROW 0 : MOV     DPTR, #KCODE0      ; set DPTR=start of row 0
        SJMP    FIND               ; find col. key belongs to
ROW 1 : MOV     DPTR, #KCODE1      ; set DPTR=start of row 1
        SJMP    FIND               ; find col. key belongs to
ROW 2 : MOV     DPTR, #KCODE2      ; set DPTR=start of row 2
        SJMP    FIND               ; find col. key belongs to
ROW 3 : MOV     DPTR, #KCODE3      ; set DPTR=start of row 3
        SJMP    FIND               ; find col. key belongs to
FIND :  RRC     A                  ; see if any CY bit is low
        JNC     MATCH              ; if zero, get the ASCII code
        INC     DPTR               ; point to next col. address
        SJMP    FIND               ; keep searching
MATCH : CLR     A                  ; set A=0 (match is found)
        MOVC    A, @A+DPTR         ; get ASCII code from table
        MOV     P0, A              ; display pressed key
        LJMP    K1
; ASCII LOOK-UP TABLE FOR EACH ROW
        ORG     300H
KCODE0 : DB     '0','1', '2', '3'
KCODE1 : DB     '4','5', '6', '7'  ; ROW 0
KCODE2 : DB     '8','9', 'A', 'B'  ; ROW 1
KCODE3 : DB     'C','D', 'E', 'F'  ; ROW 2
        END                        ; ROW 3
```

Que 5.24. **Draw and explain diagram of interfacing ADC 0804 TO 8051. Also write A/D conversion program.**

Answer

A. Interfacing of ADC 0804 with 8051 :

1. Fig. 5.24.1 shows the interfacing of ADC 0803/0804/0805 with 8051 using port 1 and port 2.
2. Here, port 1 is used to read digital data from ADC and port 2 is used to provide control signals to ADC 0803/0804/0805.
3. Potentiometer is used to adjust $+V_{in}$. The clock signal is provided using internal clock generator and two external components, resistor and capacitor as shown in the Fig. 5.24.1.
4. The frequency of such clock can be determined by,

$$f = 1/1.1RC$$

5. The typical values are $R = 10\ k\Omega$ and $C = 150$ pF. With these values we get approximately 606 KHz clock frequency. In such case, the conversion time is around 110 μs.

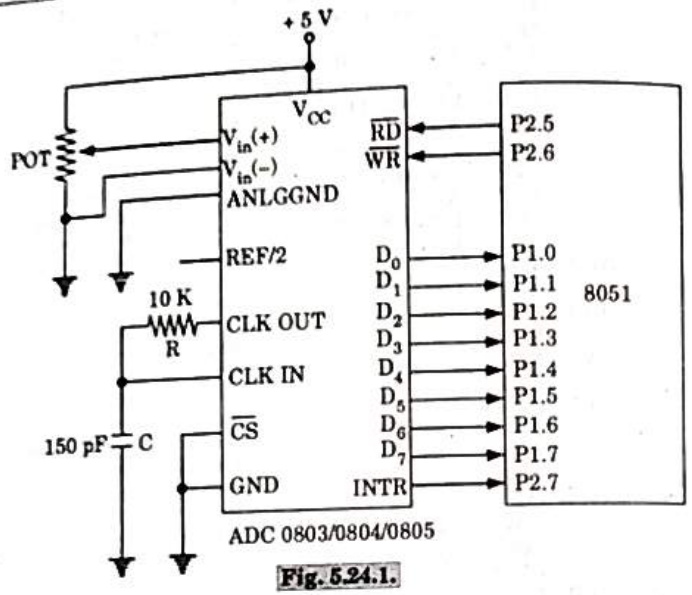


Fig. 5.24.1.

B. A/D conversion program :

```
        MOV P1, # 0FF H      ; Configure port 1 as input
Back :  CLR P2.6             ; [ Make WR = 0 and
        SETB P2.6            ; Make WR = 1 to generate
                             ; start of
                             ; conversion pulse]
Again   JB P2.7, Again       ; Wait for end of conversion
        CLR P2.5             ; Enable read
        MOV A, P1            ; Read data through port 1
        SETB P2.5            ; Disable read after reading data
        SJMP Back            ; go for next conversion cycle
```

Que 5.25. Explain the interfacing of DAC 0808 with 8051.

Answer

1. The Fig. 5.25.1 shows the block schematic of DAC 0808 interfaced to 8051 at port P1.
2. The output current I_o can be given as,

$$I_o = \frac{V_{ref}}{R_{14}}\left(\frac{D_0}{2} + \frac{D_1}{4} + \frac{D_2}{8} + \frac{D_3}{16} + \frac{D_4}{32} + \frac{D_5}{64} + \frac{D_6}{128} + \frac{D_7}{256}\right)$$

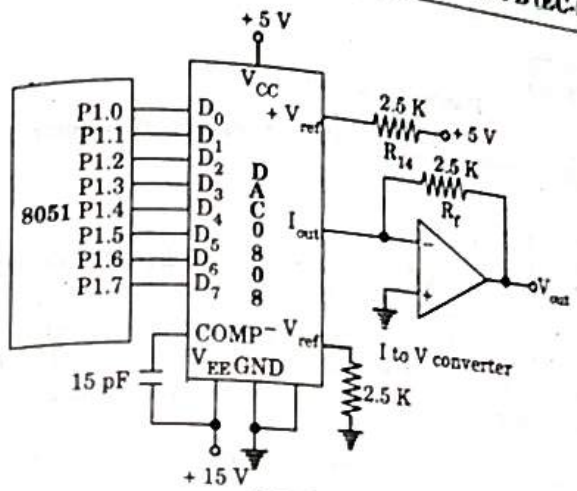


Fig. 5.25.1.

3. Input D_0 through D_7 can be either 0 or 1. Therefore, for typical circuit full scale current can be given as,

$$I_o = \frac{5}{2.5\,K}\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128} + \frac{1}{256}\right)$$

$$= \frac{2\text{ mA} \times 255}{256} = 1.992\text{ mA}$$

4. It shows that the full scale output current is always less than the reference current source of 2 mA.
4. This output current is converted into voltage by *I* to *V* converter. The output voltage for full scale input can be given as,

$$V_o = 1.992 \times 2.5\text{ K} = 4.98\text{ V}$$

PART-5

External Memory Interface, Stepper Motor and Waveform Generation.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.26. Draw the diagram external RAM interfacing with 8051.

Answer

Refer Q. 4.15, Page 4–17B, Unit-4.

Que 5.27. **Draw and explain the interfacing of stepper motor to 8051.**

Answer

1. A stepper motor is a digital motor. It can be driven by digital signal. Fig. 5.27.1 shows the typical 2 phase motor interfaced using 8051.
2. Motor shown in the circuit has two phases, with center-tap winding. The center taps of these windings are connected to the 12 V supply.
3. Due to this, motor can be excited by grounding four terminals of the two windings.
4. Motor can be rotated in steps by giving proper excitation sequence to these windings.
5. The lower nibble of port 1 of the 8051 is used to generate excitation signals in the proper sequence.
6. The Table 5.27.1 shows typical excitation sequence. The given excitation sequence rotates the motor in anticlockwise direction.
7. To rotate motor in anticlockwise direction we have to excite motor in a reverse sequence.

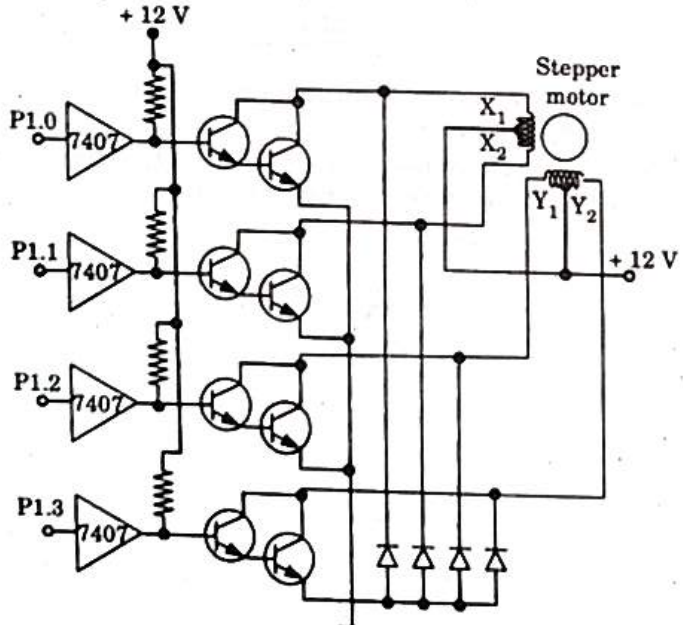


Fig. 5.27.1.

8. The excitation sequence for stepper motor may change due to change in winding connections. However, it is not desirable to excite both the ends of the same winding simultaneously. This cancels the flux and motor winding may damage.

Table 5.27.1 Full step excitation sequence.

Step	X_1	X_2	Y_1	Y_2
1	0	1	0	1
2	1	0	0	1
3	1	0	1	0
4	0	1	1	0
1	0	1	0	1

Que 5.28. **Write an ALP to generate triangular wave using DAC.**

Answer

To generate triangular wave we have to output data from 00 initially, and it should be incremented upto FF. When it reaches FF it should be decremented upto 00.

Program :

```
        MOV SP, #08 H                ; Initialize stack pointer
        MOV RO, # 00 H
REPEAT: MOV P1, R0                   ; Send digital data to the input
                                     ; of DAC 0808
        INC R0                       ; Increment digital data
        CJNE R0 #0FF H, REPEAT       ; Check digital data for peak
                                     ; output if not repeat
REPEAT: MOV P1, R0                   Send digital data to the input
                                     ; of DAC 0808
        DJNZ R0, REPEAT 1            ; Decrement digital data and
                                     check digital data for least
                                     output if not repeat
        LJMP REPEAT
```

Que 5.29. **Write an ALP to generate square wave using DAC.**

Answer

To generate square wave we have to output FF and then 00 on port 1 of 8051. The port 1 is connected as an input to the DAC 0808. The port 1 is connected as an input to the DAC 0808. According to frequency requirement delay is provided between the two outputs.

Program :

```
        MOV SP, #08 H        ; Initialize stack pointer
REPEAT: MOV P1, OFFH         ; Load all 1's in port 1
        LCALL DELAY          ; Call delay routine
        MOV P, #00 H         ; Load all 0's in Port 1
        LCALL DELAY          ; Call delay routine
        LJMP REPEAT          ; Repeat
DELAY:  MOV R0, #0FF H       ; Load delay count
BACK:   DJNZ R0, BACK        ; Decrement and check
                             whether delay count is zero if
                             not repeat the operation
        RET                  ; Return to main program
```

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 assembly language programming. Explain it with an example.

Ans. Refer Q. 5.1.

Q. 2. Write a program to add two 16-bit numbers the numbers are FC45 H and 02EC H.

Ans. Refer Q. 5.5.

Q. 3. Explain the following instructions :

i. Jump instructions.

ii. Loop instructions.

iii. Call instructions.

Ans. Refer Q. 5.7.

Q. 4. How many timers do 8051 microcontrollers have ?

Ans. Refer Q. 5.10.

Q. 5. a. Find the values of TMOD to operate as timers in the following modes :

i. Mode 1 Timer 1

ii. Mode 0 Timer 1

b. Indicate the selection made in the instruction "MOV TMOD, #20 H".

Ans. Refer Q. 5.13.

Q. 6. Explain the bit pattern of SCON register of 8051.

Ans. Refer Q. 5.17.

Q. 7. Discuss the key press and key detection mechanism of keyboard.

Ans. Refer Q. 5.22.

Q. 8. Write an ALP to generate triangular wave using DAC.

Ans. Refer Q. 5.28.

☺☺☺

t.me/quantum_aktu

aktu quantum

please support
us

- 5 Sem
- 6 Sem
- 7 Sem
- 8 Sem - future support

t.me/quantum_aktu

aktu quantum

please support us

- 5 Sem
- 6 Sem
- 7 Sem
- 8 Sem - future support

1 UNIT Introduction to Microprocessor (2 Marks Questions)

1.1. Define microprocessor, computer and microcontroller.

AKTU 2015-16, Marks 02

Ans.

A. **Microprocessor :** Microprocessor is a multipurpose, programmable, clock driven, register based electronic device that reads binary instructions from a storage device called memory; accepts binary data as input, process data according to instructions, and provides output as a result.

B. **Computer :** A computer is a programmable machine. It includes microprocessor, memory, and I/O (input/output) devices.

C. **Microcontroller :** A device that includes microprocessor, memory, and I/O signal lines on a single chip, fabricated using VLSI technology.

1.2. What are the various operations performed by microprocessor ?

AKTU 2015-16, Marks 02

Ans. Operations performed by microprocessor :

1. Microprocessor initiated operation
2. Internal operation
3. Peripheral operation

1.3. Explain the significance of HOLD and READY pin of 8085 microprocessor.

AKTU 2016-17, Marks 02

Ans.

A. **HOLD (Pin 39) :** This pin is required for DMA operation. A high condition (1) on this pin by any input/output device indicates that data is ready for DMA transfer.

B. **READY (Pin 35) :** It indicates to microprocessor to wait for time the data become available, with READY = 0 microprocessor waits till READY = 1. At READY = 1, the microprocessor knows that the data is available from memory or I/O devices.

1.4. Give the set of registers involved in 8085 microprocessor.

Ans.

i. General purpose register
ii. Temporary register
iii. Special purpose register
iv. Sixteen bit register

1.5. Write the use of memory of microprocessor.

Ans. Memory is used to store binary instruction and data for the microprocessor.

1.6. Calculate the address lines required for an 8 Kbyte memory chip.

Ans.

1. 8 Kbyte = 8192 byte
2. Number of address lines,

$$n = \frac{\log 8192}{\log 2}$$

$$= 13 \text{ address lines}$$

1.7. Calculate number of memory chip needed to design 8 Kbyte memory, if the memory chip size is 1024 × 1.

AKTU 2016-17, Marks 02

Ans.

1. The chip 1024 × 1 has 1024 (1K) registers and each register can store 1 bit.
2. For 8 Kbyte,
   $(8 \times 1024) \div (1024 \times 1) = 8$ chips

1.8. Give the description of Address Latch Enable (ALE) signal.

Ans.

1. ALE is a positive giving pulse generated every time when the 8085 begins an operation.
2. It indicates that the bits on $AD_7 - AD_0$ are address bits. It is used to latch the low order address from the multiplexed bus.

1.9. List down the various control and status signals of 8085.

Ans. i. Address latch enable (ALE) ii. $\overline{RD}$ (Read)

iii. $\overline{WR}$ (Write) iv. $IO/\overline{M}$

v. S_1 and S_0

1.10. What is the purpose of tri-state logic device ?

Ans. Tri-state logic devices are essential for proper functioning of bus oriented system, in which same bus lines are shared by several components. It has three states : logic 1, logic 0 and high impedance.

1.11. What is the objective of interfacing an output device ?

Ans. The main objective of interfacing an output device is to get the information or a result out of the processor and store it or display.

1.12. Give the types of interfacing input/output devices.

Ans. There are two types of interfacing input/output devices.

i. Peripheral I/O
ii. Memory mapped I/O

1.13. Give any two difference between memory-mapped I/O and peripheral I/O.

Ans.

S. No.	Memory-mapped I/O	Peripheral I/O
1.	Address is of 16-bit.	Address is of 8-bit.
2.	Data transfer takes place between register and I/O device.	Data transfer take place between I/O and accumulator.

1.14. What is the maximum number of input devices that can be connected to 8085 in memory mapped I/O ?

Ans. The maximum number of input devices that can be connected to 8085 in memory mapped I/O are 65536 (2^{16}).

1.15. Discuss the interfacing of DIP switches as an input device.

Ans. Interfacing of input device is similar to that of output device except that there is difference in bus signals and circuit components. An eight DIP switch can be interfaced using 3 : 8 decoder.

☺☺☺

2 UNIT

Basic Programming Concepts (2 Marks Questions)

2.1. Describe briefly the concept of flow chart.

Ans. It is a pictorial representation that a programmer uses for planning the procedure for solution of problems.

2.2. Discuss the data transfer operation.

Ans. The data transfer operation is used to copy data from one register or memory location called source to another register or memory location called destination.

2.3. List down the various data transfer operation.

Ans.
i. MOV (Move)
ii. MVI (Move immediate data)
iii. LXI (Load register pair immediate)
iv. LDA (Load accumulator direct)
v. SHLD (Store H and L register direct)
vi. STAX (Store accumulator indirect)

2.4. Write instructions to read the data at input port 07 H and at the port 08 H. Display the input data from 07 H at output port 00 H and store the input data from port 08 H in register B. **AKTU 2016-17, Marks 02**

Ans.
IN 07 H
OUT 00 H
IN 08 H
MOV B, A

2.5. Write down the various arithmetic operations.

Ans.
i. ADD (Add register to accumulator)
ii. ADI (Add immediate to accumulator)
iii. SUB (Subtract the number)
iv. SUI (Subtract immediate from accumulator)
v. INR (Increment content by 1)
vi. DCR (Decrement content by 1)

2.6. What operation can be performed by using the instruction SUB A ? Specify the states of Z and CY.

AKTU 2016-17, Marks 02

Ans. The instruction SUB A will clear the accumulator.
The flag status will be : CY = 0, Z = 1.

2.7. Write down the various logical instructions.

Ans.
i. ANA (Logical AND with accumulator)
ii. ANI (AND immediate with accumulator)
iii. XRA (Exclusive OR with accumulator)
iv. XRI (Exclusive OR immediate with accumulator)
v. ORA (Logically OR with accumulator)
vi. ORI (Logically OR immediate with accumulator)

2.8. Describe the following 8085 instructions :
i. DAA
ii. JPE 3040 H

AKTU 2015-16, Marks 02

Ans.
i. **DAA (Decimal adjust accumulator) :** This instruction is used to convert the result of the addition of two packed BCD numbers to a valid BCD number.
ii. **JPE 3040 H :** Jump to the address 3040 H if parity is even.

2.9. Differentiate between the types of loops.

Ans. Loops are classified as continuous loop and conditional loops.
Difference :

S. No.	Continuous loop	Conditional loop
1.	Continuous loop contain unconditional jump instructions. A program with this type of loop does not stop until the system is reset.	Conditional loops are set up by using conditional jump instructions.

2.10. Write short note on counter.

Ans. Counters are mainly used to keep track of events. It can be designed by loading an appropriate number into any one register of microprocessor and then using INR or DCR instructions for counting.

2.11. Describe the delay using one register.

Ans. The instructions are as follows :

```
        MVI C, FF H   ; Load register C
Loop :  DCR C         ; Decrement C
        JNZ Loop      ; Jump back to decrement C
```

2.12. What is the difference between hardware interrupts and software interrupts of 8085 microprocessor ?

AKTU 2016-17, Marks 02

Ans.

S. No.	Hardware Interrupt	Software Interrupt
1.	In hardware interrupts the cause of interrupt is some peripheral device which interrupt the main program for I/O operation.	In software interrput the cause of interrupt is an execution of the instruction. These are special instruction supported by microprocessor.
2.	Hardware interrupts are : RST 7.5, RST 6.5, RST 5.5, INTR, TRAP.	Software interrupts are : RST 0 to RST 7.

2.13. Define the RIM and SIM instruction of 8085 microprocessor.

AKTU 2016-17, Marks 02

Ans.
A. **RIM (Reset Interrupt Mask) :** This instruction is used to read interrupt marks, to identify pending interrupts and to receive serial data.
B. **SIM (Set Interrupt Mask) :** It reads accumulator and enable and disable interrupts according to accumulator content.

2.14. How the microprocessor behaves when the interrupt is generated through interrupt pin of 8085 (TRAP, RST 7.5, RST 6.5, RST 5.5) and DMA interrupt ?

AKTU 2015-16, Marks 02

Ans. **Interrupt pin of 8085 :**
a. **TRAP :** When this interrupt is received the processor saves the contents of the PC register into stack and branches 0024 H.
b. **RST 7.5 :** When this interrupt is received the processor saves the contents of the PC register into stack and branches to 003C H.
c. **RST 6.5 :** When this interrupt is received the processor saves the contents of the PC register into stack and branches to 0034 H.
d. **RST 5.5 :** When this interrupt is received the processor saves the contents of the PC register and branches to 002C H.
e. **DMA interrupt :**
The 8085 has two lines dedicated to DMA operation-the HOLD and HLDA lines. The HOLD line may be used by the I/O devices to receive an acknowledgement to this request.

☺☺☺

3 UNIT

16-bit Microprocessor (2 Marks Questions)

3.1. Differentiate between 8085 and 8086 microprocessor.

Ans.

S. No.	8085	8086
1.	8-bit microprocessor.	16-bit microprocessor.
2.	16 address lines.	20 address lines.
3.	It does not support pipelining.	It supports pipelining.

3.2. List the functional units of 8086.

Ans.

i. Bus interface unit (BIU)
ii. Execution unit (EU)

3.3. Why there are two ground pins in 8086 ? Explain.

AKTU 2016-17, Marks 02

Ans. There are two ground pins in 8086 to distribute power for noise reduction.

3.4. What is the significance of pipelining in 8086 microprocessor ?

Ans. Pipelining is the process of accumulating and executing computer instructions and tasks from the processor. It allows storing, prioritizing, managing & executing tasks and instructions in an orderly pass.

3.5. Explain need of memory segmentation in 8086.

AKTU 2015-16, Marks 02

Ans.

i. It allows the memory addressing capacity to be 1 MB even though the address associated with individual instruction is only 16-bit.
ii. It facilitates use of separate memory areas for program data and stack.

3.6. What is DMA ?

Ans. DMA (Direct Memory Access) is an I/O technique commonly used for high speed data transfer.

3.7. List the priority schemes of DMA requests.

Ans.

i. Fixed priority scheme.
ii. Rotating priority scheme.

3.8. Explain the significance of status register in 8237.

Ans. Status register keeps the track of all the DMA channel pending request and the status of their terminal counts.

3.9. List any two salient features of mode 2 in 8255.

Ans.

i. The 8-bit port is bidirectional and additionally a 5-bit control port is available.
ii. Inputs and outputs are both latched.

3.10. What is 8255 ?

Ans. The 8255 is a programmable peripheral interface which is used for parallel data transfer.

3.11. Write the control word format for BSR mode of 8255.

Ans. Control word format for BSR mode of 8255 is given as :

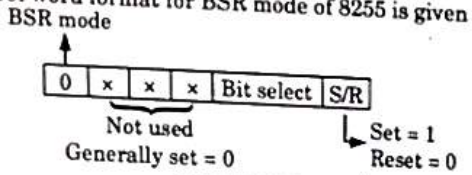


Fig. 3.11.1.

3.12. What are the basic differences between 8253 and 8254 programmable timer/counter ?

Ans.

S. No.	8253	8254
1.	Operating frequency 0-2.6 MHz.	Operating frequency 0-10 MHz.
2.	Read-back command not available.	Read-back command available.

3.13. What is 8259A ?

Ans. 8259A is a programmable interrupt controller and is required to handle a number of interrupts at a time. This controller also decides the priority of interrupts.

3.14. List any two features available in 8259.

Ans.
i. 8259 can be programmed to get 4 priority interrupts.
ii. By cascading 8259, it is possible to get 64 priority interrupts.

☺☺☺

4 UNIT

8051 Microcontroller Basics (2 Marks Questions)

4.1. What do you mean by embedded system ?

Ans.
1. An embedded system is an electronic/electromechanical system designed to perform a specific function and is a combination of both hardware and firmware (software).
2. Each embedded system is unique, and the hardware as well as the software is highly specialized to the application domain.

4.2. What are the main application areas of embedded system ?

Ans. The main application areas are :
1. Consumer electronics : camcorders, cameras etc.
2. Household-appliances : T.V., fridge etc.
3. Telecom
4. Healthcare
5. Computer peripherals.

4.3. Draw a basic block diagram of embedded system.

Ans.

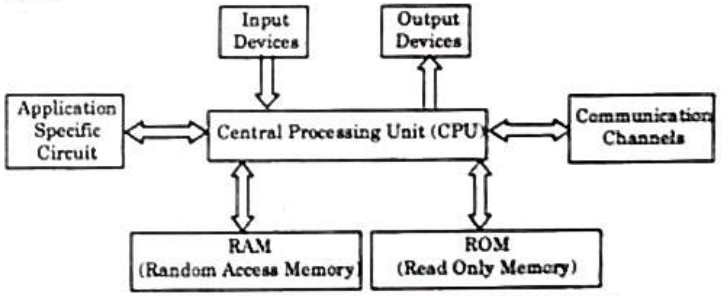


Fig. 4.3.1. Block diagram of an embedded system.

4.4. What are the advantages of microcontroller over microprocessor ?

Ans.
1. Microcontrollers are normally less expensive than microprocessors.
2. It meets the computing needs of the task most effectively.

3. Power consumption is low.
4. Software piracy can be protected in microcontroller.

4.5. Give the comparison of 8051 family members.

Ans.

Feature	8051	8052	8031
ROM (on-chip program space in bytes)	4K	8K	0K
RAM (bytes)	128	256	128
Timers	2	3	2
I/O pins	32	32	32
Serial port	1	1	1
Interrupt sources	6	8	6

4.6. Compare 8051 and MSP430x5xxx main features.

AKTU 2016-17, Marks 02

Ans.

S. No.	8051	MSP430x5xxx
1.	It is 8 bit microcontroller.	It is 16 bit microcontroller.
2.	It is based on CISC architecture.	It is based on RISC architecture.
3.	It has clock frequency of 12 MHz.	It has clock frequency of 3.3 MHz.
4.	It supports 5 addressing modes for source and destination operand.	It supports 7 addressing modes for source operand and 4 addressing modes for destination operand.

4.7. What are flags available in 8051 ?

Ans. The flags available in 8051 are : CY (Carry flag), AC (Auxiliary carry flag), OV (over flow flag) and P (Parity flag).

4.8. Explain the function of the $\overline{PSEN}$ pin of 8051.

Ans. This is an output pin. $\overline{PSEN}$ stands for program store enable. This is an active low output pin. It acts as a strobe to read the external program memory.

4.9. What do you mean by SFR ?

Ans.

1. The 8051 operations that do not use the internal 128-bytes RAM addresses from 00 H to 7F H are done by a group of specific internal register, each called a Special Functions Register (SFR).

2. In 8051, register *A*, *B*, PSW and DPTR are a part of the group of registers referred as SFR.

4.10. What is the size of the program counter register ? What does the program counter do ? **AKTU 2017-18, Marks 02**

Ans. The size of program counter is 20 bit. A program counter is a register that contains the address of the instruction being executed at the current time.

4.11. What do you mean by addressing modes ?

Ans. The CPU can access data in many ways. The data could be in a register or in memory or can be provided as an immediate value. Thus, the various ways of accessing the data are called addressing modes.

4.12. Name the various types of addressing modes used in 8051 microcontroller.

Ans. The 8051 consists of 5 addressing modes :

1. Immediate.
2. Register addressing mode.
3. Register indirect addressing mode.
4. Indexed addressing mode.
5. Direct addressing mode.

4.13. What is the advantage of register indirect addressing mode over direct addressing mode ?

Ans. Advantage of register indirect addressing mode is that it makes accessing data dynamic rather than static as in the case of direct addressing mode.

☺☺☺

5 UNIT

Assembly Programming and Instruction of 8051 (2 Marks Questions)

5.1. What do you mean by assembly language ?

Ans. An assembly language program consist a series of lines of assembly language instructions. These instructions consist of mnemonic, optionally followed by one or two operands. The operands are the data items being manipulated, and mnemonics are the commands to CPU.

5.2. How does an instruction differ from directive ?

Ans.

S. No.	Instruction	Directive
1.	An instruction consists of a mnemonic, optionally followed by one or two operands. It basically tells the CPU what to do.	Directives give the directions to the assembler.
2.	The instructions are translated into machine code (opcode) for the CPU to execute.	Directives do not generate any machine code and are used only by the assembler.

5.3. List the different assembler directives.

Ans. The following are the widely used 8051 assembler directives :

i. ORG (Origin).
ii. EQU (Equate).
iii. END.
iv. DB (Define byte).

5.4. What do you mean by END directive ?

Ans. END directive indicates to the assembler about the end of the source (asm) file. The END directive is the last line of an 8051 program, meaning that in the source code anything after the END directive is ignored by the assembler.

5.5. Why are the ORG and END directives also called pseudocode ?

Ans.

1. ORG and END directives are called pseudocodes because they are directives to the assembler.
2. ORG tells the assembler to place the opcode at memory location 0 while END indicates to the assembler the end of source code.

5.6. Draw the structure of TMOD register.

Ans.

MSB							LSB
GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
Timer 1				Timer 0			

Fig. 5.6.1.

5.7. Define the use of MOVX and MOVC instruction in 8051 microcontroller. **AKTU 2016-17, Marks 02**

Ans.

A. **MOVX :** This instruction transfers data byte between external memory and register.
MOVX destination-byte, source-byte
Example : MOVX A, @ DPTR
MOVX @ DPTR, A

B. **MOVC :** This instruction moves 1 byte of data located in the program area to A (accumulator). The address of the desired byte of data is formed by adding the program counter (PC) register to the original value of A.
Example : MOVC A, @ A + PC
MOVC A, @ A + DPTR

5.8. Enlist the interrupt in 8051 microcontroller.

Ans.

i. Reset,
ii. Timer 0(TF0),
iii. Timer 1(TF1),
iv. INT0,
v. INT1,
vi. Serial COM interrupt (RI and TI).

5.9. Find the timer's clock frequency and its period for various 8051 based systems, with the following crystal frequency :

a. 12 MHz
b. 16 MHz
c. 11.592 MHz

Ans.

a. Timer's clock frequency = 1/12 × 12 MHz = 1 MHz
and T = 1/1 MHz = 1 μs

b. Timer's clock frequency = 1/12 × 16 MHz = 1.333 MHz and T = 1/1.333 MHz = .75 μs

c. Timer's clock frequency = 1/12 × 11.592 MHz = 921.6 kHz;

$$T = 1/921.6 \text{ kHz} = 1.085 \ \mu s$$

5.10. How registers are selected in LCD ?

Ans.

1. The RS (Register Select) is used for the selection of registers in LCD.
2. If RS = 0, the instruction command code register is selected.
3. If RS = 1, the data register is selected allowing the user to send data to be displayed on the LCD.

5.11. Define bit addressable RAM in 8051 microcontroller.

AKTU 2016-17, Marks 02

Ans. Of the 128 byte internal RAM of the 8051, only 16 byte are bit-addressable. The bit addressable RAM locations are 20 H to 2F H. These 16 bytes provide 128-bits of RAM bit-addressability, since 16 × 8 = 128. They are addressed as 0 to 127 (in decimal) or 00 H to 7F H.

5.12. What do you understand by stepper motor ?

Ans. The stepper or stepping motor has a rotor movement in discrete steps. The angular rotation is determined by the number of pulses fed into the control circuit. Each input pulse initiates the drive circuit which produces one step of angular movement.

☺☺☺

t.me/quantum_aktu

aktu quantum

please support
us

- 5 Sem
- 6 Sem
- 7 Sem
- 8 Sem - future support