Q.2 a. Discuss and differentiate between a Microprocessor and a Microcontroller.

Answer:

Microprocessor is an IC which has only the CPU inside them i.e. only the processing powers such as Intel's Pentium 1,2,3,4, core 2 duo, i3, i5 etc. These microprocessors don't have RAM, ROM, and other peripheral on the chip. A system designer has to add them externally to make them functional. Application of microprocessor includes Desktop PC's, Laptops, notepads etc.

But this is not the case with Microcontrollers. Microcontroller has a CPU, in addition with a fixed amount of RAM, ROM and other peripherals all embedded on a single chip. At times it is also termed as a mini computer or a computer on a single chip. Today different manufacturers produce microcontrollers with a wide range of features available in different versions. Some manufacturers are ATMEL, Microchip, TI, Freescale, Philips, Motorola etc.

Microcontrollers are designed to perform specific tasks. Specific means applications where the relationship of input and output is defined. Depending on the input, some processing needs to be done and output is delivered. For example, keyboards, mouse, washing machine, digicam, pendrive, remote, microwave, cars, bikes, telephone, mobiles, watches, etc. Since the applications are very specific, they need small resources like RAM, ROM, I/O ports etc and hence can be embedded on a single chip. This in turn reduces the size and the cost.

Microprocessor find applications where tasks are unspecific like developing software, games, websites, photo editing, creating documents etc. In such cases the relationship between input and output is not defined. They need high amount of resources like RAM, ROM, I/O ports etc.

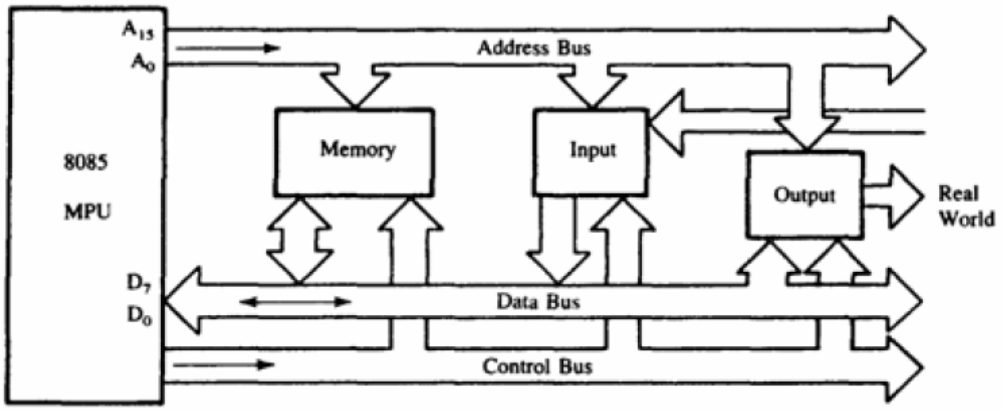
The clock speed of the Microprocessor is quite high as compared to the microcontroller. Whereas the microcontrollers operate from a few MHz to 30 to 50 MHz, today's microprocessor operate above 1GHz as they perform complex tasks.

Comparing microcontroller and microprocessor in terms of cost is not justified. Undoubtedly a microcontroller is far cheaper than a microprocessor. However microcontroller cannot be used in place of microprocessor and using a microprocessor is not advised in place of a microcontroller as it makes the application quite costly. Microprocessor cannot be used stand alone. They need other peripherals like RAM, ROM, buffer, I/O ports etc and hence a system designed around a microprocessor is quite costly.

b. List different system buses of 8085 microprocessor and give function of each bus. (8)

Answer:

A typical 8085 microprocessor communicates with memory and other devices (input and output) using three busses: Address Bus, Data Bus and Control Bus.



Address Bus

One wire for each bit, therefore 16 bits = 16 wires. Binary number carried alerts memory to 'open' the designated box. Data (binary) can then be put in or taken out.The Address Bus consists of 16 wires, therefore 16 bits. Its "width" is 16 bits. A 16 bit binary number allows 216 different numbers, or 32000 different numbers, ie 0000000000000000 up to 1111111111111111. Because memory consists of boxes, each with a unique address, the size of the address bus determines the size of memory, which can be used. To communicate with memory the microprocessor sends an address on the address bus, eg 0000000000000011 (3 in decimal), to the memory. The memory the selects box number 3 for reading or writing data. Address bus is unidirectional, ie numbers only sent from microprocessor to memory, not other way.

Data Bus

Data Bus: carries 'data', in binary form, between µP and other external units, such as memory. Typical size is 8 or 16 bits. Size determined by size of boxes in memory and µP size helps determine performance of µP. The Data Bus typically consists of 8 wires. Therefore, 28 combinations of binary digits. Data bus used to transmit "data", ie information, results of arithmetic, etc, between memory and the microprocessor. Bus is bi-directional. Size of the data bus determines what arithmetic can be done. If only 8 bits wide then largest number is 11111111 (255 in decimal). Therefore, larger number have to be broken down into chunks of 255. This slows microprocessor. Data Bus also carries instructions from memory to the microprocessor. Size of the bus therefore limits the number of possible instructions to 256, each specified by a separate number.

Control Bus

Control Bus are various lines which have specific functions for coordinating and controlling uP operations. Eg: Read/NotWrite line, single binary digit. Control whether memory is being 'written to' (data stored in mem) or 'read from' (data taken out of mem) 1 = Read, 0 = Write. May also include clock line(s) for timing/synchronising, 'interrupts', 'reset' etc. Typically µP has 10 control lines. Cannot function correctly without these vital control signals.
The Control Bus carries control signals partly unidirectional, partly bi-directional. Control signals are things like "read or write". This tells memory that we are either **reading from** a location, specified on the address bus, or **writing to** a location specified. Various other signals to control and coordinate the operation of the system. Modern day microprocessors, like 80386, 80486 have much larger busses. Typically 16 or 32 bit busses, which allow larger number of instructions, more memory

location, and faster arithmetic. Microcontrollers organized along same lines, except: because microcontrollers have memory etc inside the chip, the busses may all be internal. In the microprocessor the three busses are external to the chip (except for the internal data bus). In case of external busses, the chip connects to the busses via buffers, which are simply an electronic connection between external bus and the internal data bus.

Q.3 a. Discuss in detail the isolated I/O mapping and memory mapped I/O devices.

Answer:

There are two types for interfacing I/O devices:

1. **Memory mapped I/O device.**
2. **Standard I/O mapped I/O device or isolated I/O mapping.**

Memory Mapping of I/O device	I/O Mapping of I/O device
1. 16-bit addresses are provided for I/O devices.	1. 8-bit addresses are provided for I/O devices.
2. The devices are accessed by memory read or memory write cycles.	2. The devices are accessed by I/O read or I/O write cycle. During these cycles the 8-bit address is available on both low order address lines and high order address lines.
3. The I/O ports or peripherals can be treated like memory locations and so all instructions related to memory can be used for data transfer between I/O device and the processor.	3. Only IN and OUT instructions can be used for data transfer between I/O device and the processor.
4. In memory mapped ports the data can be moved from any register to ports and vice-versa.	4. In I/O mapped ports the data transfer can take place only between the accumulator and ports.
5. When memory mapping is used for I/O devices, the full memory address space cannot be used for addressing memory. Hence memory mapping is useful only for small systems, where the memory requirement is less.	5. When I/O mapping is used for I/O devices then the full memory address space can be used for addressing memory. Hence it is suitable for systems which requires large memory capacity.
6. In memory mapped I/O devices, a large number of I/O ports can be interfaced.	6. In I/O mapping only 256 ports ($2^8 = 256$) can be interfaced.
7. For accessing the memory mapped devices, the processor executes memory read or write cycle. During this cycle IO/$\overline{M}$ is asserted **low** (IO/$\overline{M} = 0$).	7. For accessing the I/O mapped devices, the processor executes I/O read or write cycle. During this cycle IO/$\overline{M}$ is asserted **high** (IO/$\overline{M} = 1$).

b. Let at the program memory location 4080, the instruction MOV B, A (opcode 47H) is stored while the accumulator content is FFH. Illustrate the execution of this instruction by timing diagram.

Answer:

Since the program counter sequences the execution of instructions, it is assumed that the PC holds the address 4080_H. While the system executes the instruction, the following takes place one after another.

1. The CPU places the address 4080_H (residing in PC) on the address bus—40_H on the high order bus $A_{15} - A_8$ and 80_H on the low order bus $AD_7 - AD_0$.
2. The CPU raises the ALE signal to go high—the H to L transition of ALE at the end of the first T state demultiplexes the low order bus.
3. The CPU identifies the nature of the machine cycle by means of the three status signals IO/$\overline{M}$, S_0 and S_1.

   IO/ = 0, $S_1 = 1$, $S_0 = 1$
4. In T_2, memory is enabled by the signal. The content of PC i.e., 47_H is placed on the data bus. PC is incremented to 4081_H.
5. In T_3, CPU reads 47_H and places it in the instruction register.
6. In T_4, CPU decodes the instruction, places FF_H (accumulator content) in the temporary register and then transfers it to register B. Figure shows the execution of the above. It consists of 4T states.

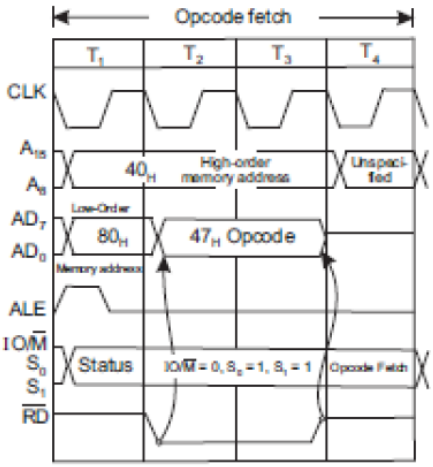


Fig. 8085 timing for execution of the instruction (MOV B, A)

Q.4 a. Write a program in 8085 assembly language to convert BCD to Binary.

Answer:

```
LDA 2000H ;
        ANI F0  ; A = 30
        RRC  ; MAKE MSB LSB
        RRC
        RRC
        RRC
        MVI B, 0AH ; B = 0A
       CONTI ADD A  ;
        DCR B  ;
        JNZ CONTI ;
        MOV B, A ; B = 1E = A
        LDA 2000H ;
        ANI 0F  ; A = 4
        ADD B  ; A = 22
        STA 2001H ; STORE 22
        HLT
```

b. Write a program in 8085 assembly language to multiply two numbers.

Answer:

MEMORY ADDRESS	HEXA-CODE	LABEL	MNEMONICS	COMMENTS
8000	21		LXI H,8100	Initialize first input.
8001	00			
8002	81		MOV B,M	Move the contents of memory to B.
8003	46			
8004	23		INX H	Give the second input.
8005	4E		MOV C,M	Move the contents to memory to C.
8006	3E		MVI A,00	Move immediately constant as 00.
8007	00			
8008	16		MVI D,00	Move immediately D as 00.
8009	00			
800A	80	L2	ADD B	Add B with A.
800B	D2		JNC L1	Jump no carry.
800C	0F			
800D	80		INR D	
800E	14			
800F	OD	L1	DCR C	Decrement the content of C.
8010	C2		JNZ L2	Jump to non zero.
8011	0A			
8012	80			
8013	32		STA 8200	Store the value in 8200.
8014	00			
8015	82			
8016	7A		MOV A,D	Move D to A.
8017	32		STA 8201	Store the value in 8201.
8018	01			
8019	82			
801A	03		HLT	Stop the execution.

Q.5 a. Explain RIM ,SIM instructions with suitable illustration. (8)

Answer:

SIM and RIM for interrupts:
The 8085 provide additional masking facility for RST 7.5, RST 6.5 and RST 5.5 using SIM instruction.

The status of these interrupts can be read by executing RIM instruction.

The masking or unmasking of RST 7.5, RST 6.5 and RST 5.5 interrupts can be performed by moving an 8-bit data to accumulator and then executing SIM instruction.

The format of the 8-bit data is shown below.

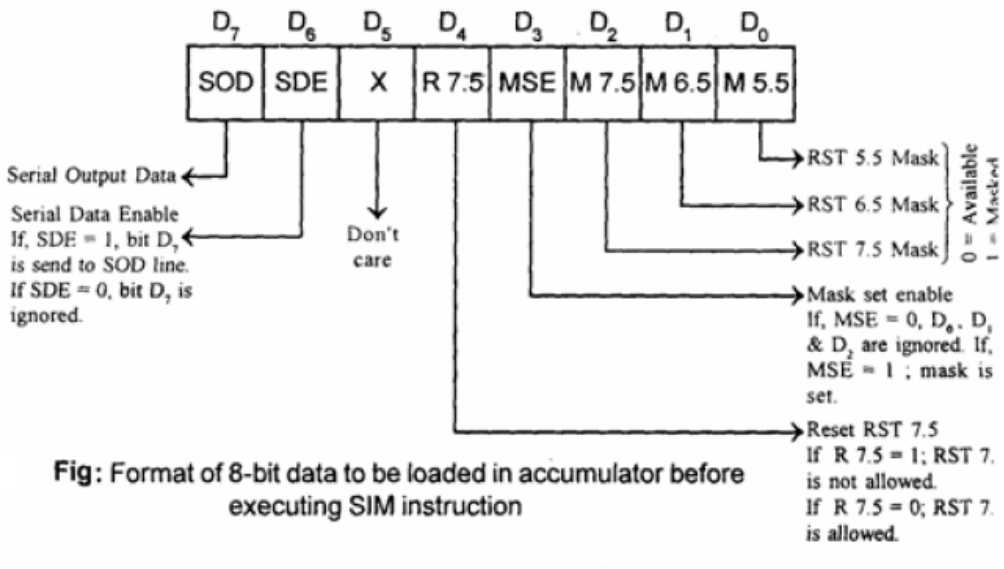
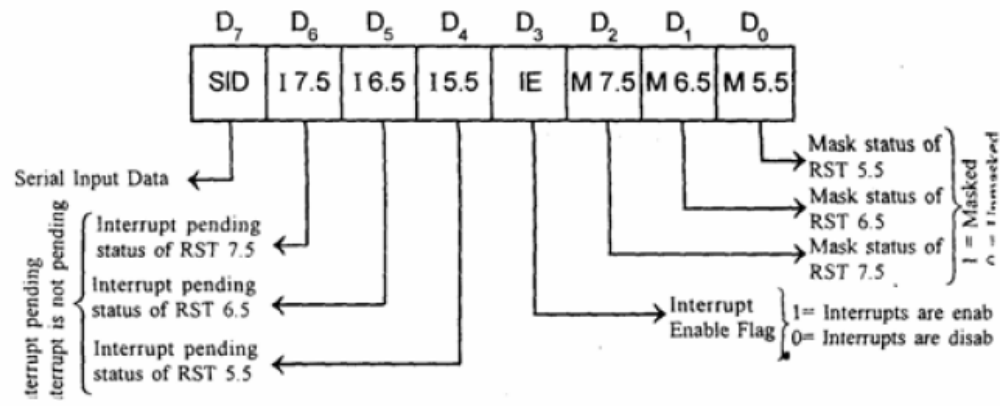


Fig: Format of 8-bit data to be loaded in accumulator before executing SIM instruction

The status of pending interrupts can be read from accumulator after executing RIM instruction.

When RIM instruction is executed an 8-bit data is loaded in accumulator, which can be interpreted as shown in fig.



Interrupt type	Trigger	Priority	Maskable	Vector address
TRAP	Edge and Level	1st	No	0024H
RST 7.5	Edge	2nd	Yes	003CH
RST 6.5	Level	3rd	Yes	0034H
RST 5.5	Level	4th	Yes	002CH
INTR	Level	5th	Yes	-

b. Describe the control port of 8255.

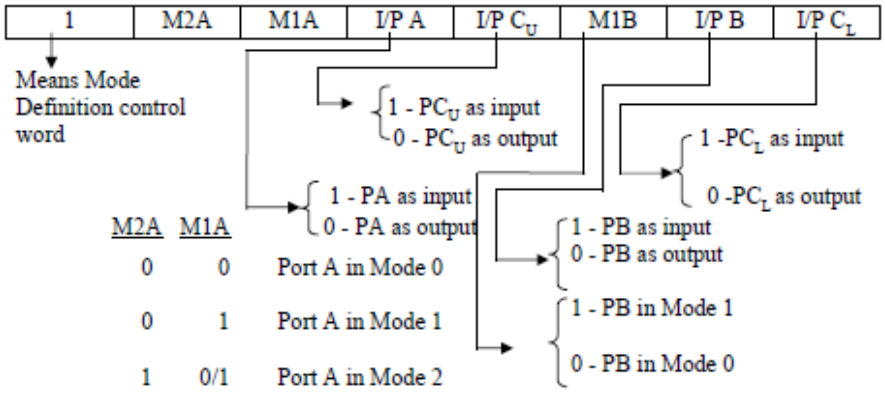
Answer:

A1 A0	Selected port
0 0	Port A
0 1	Port B
1 0	Port C
1 1	Control port

The Control port contents decides the working of 8255. When CS (Chip select) is 0, 8255 is selected for communication by the processor. The chip select circuit connected to the CS pin assigns addresses to the ports of 8255.For the chip select circuit shown, the chip is selected when A7=0, A6=1, A5=1, A4=1, A3=1, A2=1, and M/IO*= 0. Port A, Port B, Port C and Control port will have the addresses as 7CH, 7DH, 7EH, and 7FH respectively.

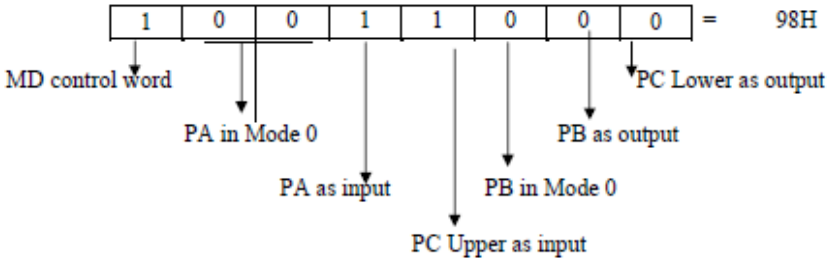
Mode definition control word is used to configure the ports of 8255 as input or output in Mode 0, Mode 1, or Mode 2.

Control port having Mode Definition (MD) control word

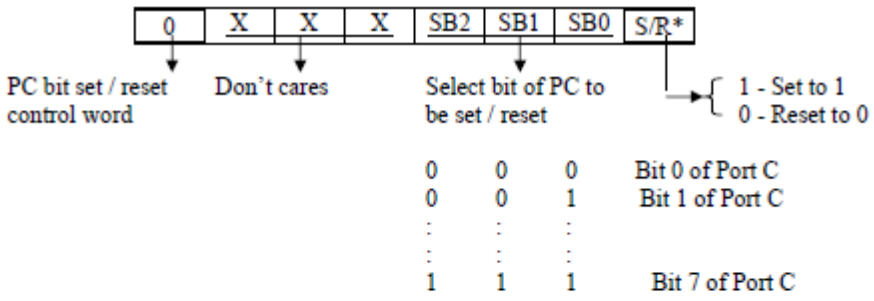


Ex. 1: Configure Port A as input in Mode 0, Port B as output in mode 0, Port C (Lower) as output and Port C (Upper) as input ports.

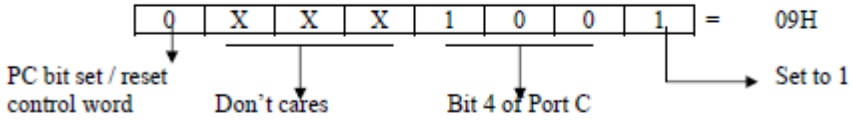
Required MD control word:



Port C Bit Set / Reset (PCBSR) control word is used for setting to 1 or resetting to 0 any one selected bit of Port C. It is useful for enabling or disabling Port A or Port B interrupts when they are in mode 1 or mode 2. Control port having Port C Bit Set / Reset control word



Ex. 1: Set to 1 bit 4 of Port C



Required program segment for setting bit 4 of Port C:

```
MOV AL, 09H
OUT 7FH, AL
```

Q.6 a. Discuss the simulation of a 4-bit ALU.

Answer:

In ECL, TTL and CMOS, there are available integrated packages which are referred to as arithmetic logic units (ALU). The logic circuitry in this units is entirely combinational (i.e. consists of gates with no feedback and no flip-flops).The ALU is an extremely versatile and useful device since, it makes available, in single package, facility for performing many different logical and arithmetic operations.

Arithmetic Logic Unit (ALU) is a critical component of a microprocessor and is the core component of central processing unit.

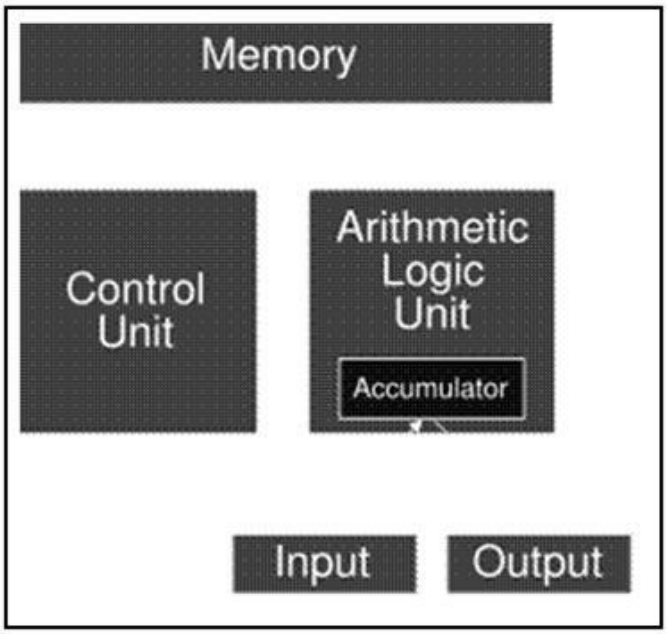


Fig.Central Processing Unit (CPU)

ALU's comprise the combinational logic that implements logic operations such as AND, OR and arithmetic operations, such as ADD, SUBTRACT.

Functionally, the operation of typical ALU is represented as shown in diagram below,

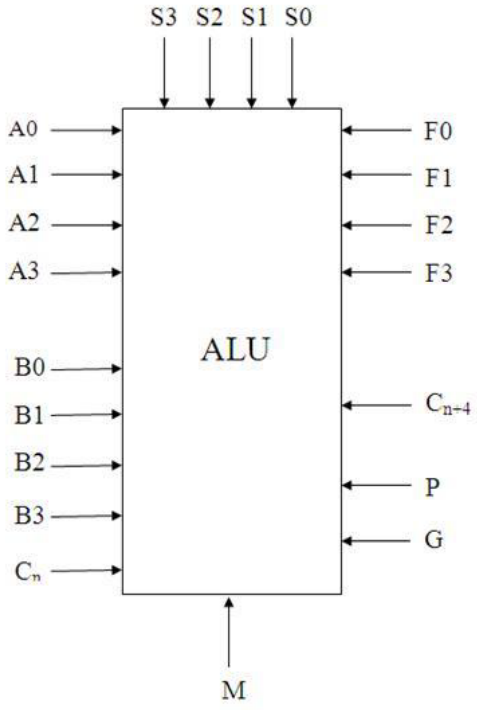


Fig. Functional representation of Arithmetic Logic Unit

Functional Description of 4-bit Arithmetic Logic Unit

Controlled by the four function select inputs (S0 to S3) and the mode control input (M), ALU can perform all the 16 possible logic operations or 16 different arithmetic operations on active HIGH or active LOW operands.

When the mode control input (M) is HIGH, all internal carries are inhibited and the device performs logic operations on the individual bits. When M is LOW, the carries are enabled and the ALU performs arithmetic operations on the two 4-bit words. The ALU incorporates full internal carry look-ahead and provides for either ripple carry between devices using the Cn+4 output, or for carry look-ahead between packages using the carry propagation (P) and carry generate (G) signals. P and G are not affected by carry in.

For high-speed operation the device is used in conjunction with the ALU carry look-ahead circuit. One carry look-ahead package is required for each group of four ALU devices. Carry look-ahead can be provided at various levels and offers high-speed capability over extremely long word lengths. The comparator output (A=B) of the device goes HIGH when all four function outputs (F0 to F3) are HIGH and can be used to indicate logic equivalence over 4 bits when the unit is in the subtract mode. A=B is an open collector output and can be wired-AND with other A=B outputs to give a comparison for more than 4 bits. The open drain output A=B should be used with an external pull-up resistor in order to establish a logic HIGH level. The A=B signal can also be used with the Cn+4 signal to indicate A > B and A < B.

The function table lists the arithmetic operations that are performed without a carry in. An incoming carry adds a one to each operation. Thus, select code LHHL generates A minus B minus 1 (2s complement notation) without a carry in and generates A minus B when a carry is applied.

Because subtraction is actually performed by complementary addition (1s complement), a carry out means borrow; thus, a carry is generated when there is no under-flow and no carry is generated when there is underflow.

As indicated, the ALU can be used with either active LOW inputs producing active LOW outputs (Table 1) or with active HIGH inputs producing active HIGH outputs (Table 2).

MODE SELECT INPUTS				ACTIVE HIGH INPUTS AND OUTPUTS	
S_3	S_2	S_1	S_0	LOGIC (M=H)	ARITHMETIC(2) (M=L; C_n=H)
L	L	L	L	$\overline{A}$	A
L	L	L	H	$\overline{A + B}$	A + B
L	L	H	L	$\overline{A}B$	A + $\overline{B}$
L	L	H	H	logical 0	minus 1
L	H	L	L	$\overline{AB}$	A plus A$\overline{B}$
L	H	L	H	$\overline{B}$	(A + B) plus A$\overline{B}$
L	H	H	L	A ⊕ B	A minus B minus 1
L	H	H	H	A$\overline{B}$	A$\overline{B}$ minus 1
H	L	L	L	$\overline{A}$ + B	A plus AB
H	L	L	H	$\overline{A \oplus B}$	A plus B
H	L	H	L	B	(A + $\overline{B}$) plus AB
H	L	H	H	AB	AB minus 1
H	H	L	L	logical 1	A plus A(1)
H	H	L	H	A + $\overline{B}$	(A + B) plus A
H	H	H	L	A + B	(A + $\overline{B}$) plus A
H	H	H	H	A	A minus 1

Table1: Function Table for active low inputs and outputs

Notes to the function tables:

1. Each bit is shifted to the next more significant position.
2. Arithmetic operations expressed in 2s complement notation.

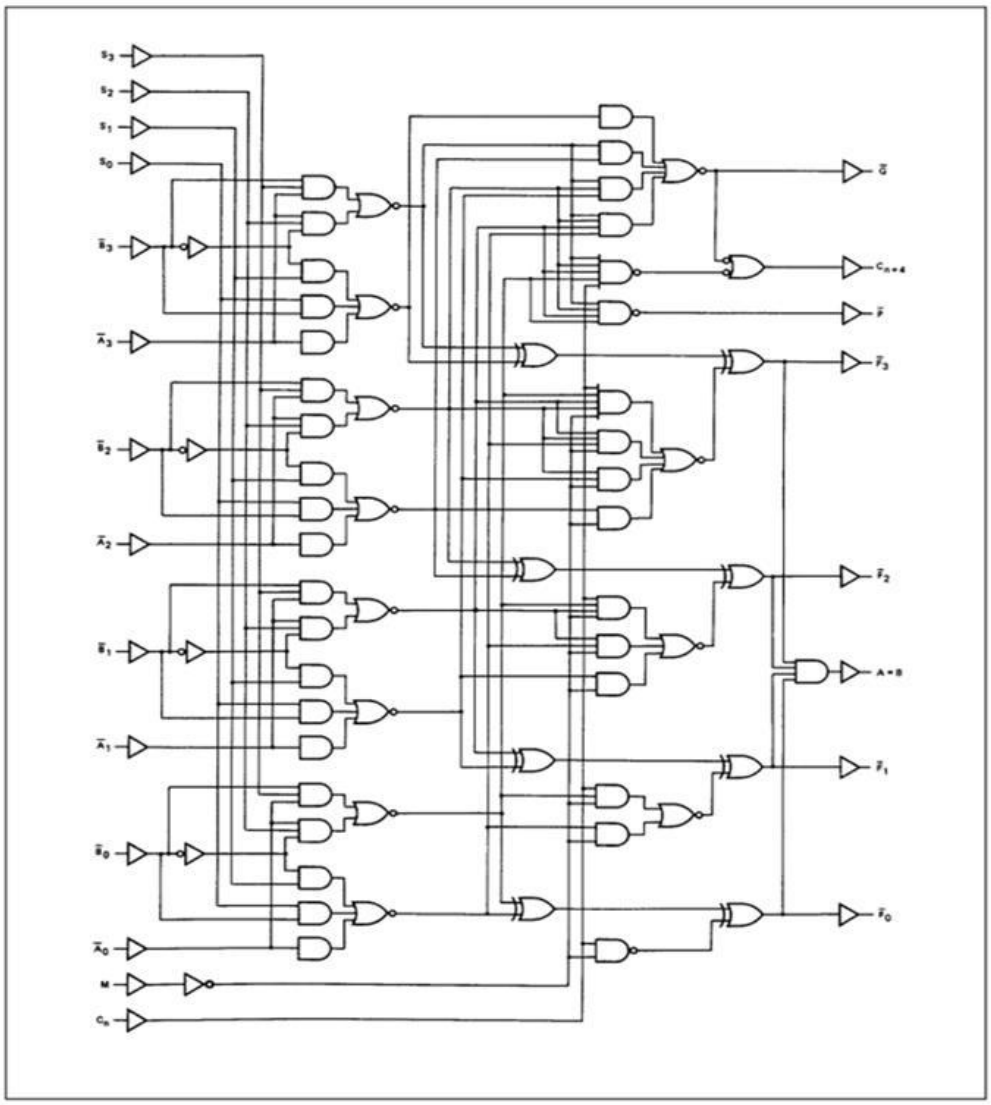
H = HIGH voltage level

L = LOW voltage level

MODE SELECT INPUTS				ACTIVE HIGH INPUTS AND OUTPUTS	
S_3	S_2	S_1	S_0	LOGIC (M=H)	ARITHMETIC(2) (M=L; C_n=H)
L	L	L	L	$\overline{A}$	A
L	L	L	H	$\overline{A + B}$	A + B
L	L	H	L	$\overline{A}B$	$A + \overline{B}$
L	L	H	H	logical 0	minus 1
L	H	L	L	$\overline{AB}$	A plus $A\overline{B}$
L	H	L	H	$\overline{B}$	(A + B) plus $A\overline{B}$
L	H	H	L	$A \oplus B$	A minus B minus 1
L	H	H	H	$A\overline{B}$	$A\overline{B}$ minus 1
H	L	L	L	$\overline{A} + B$	A plus AB
H	L	L	H	$\overline{A \oplus B}$	A plus B
H	L	H	L	B	$(A + \overline{B})$ plus AB
H	L	H	H	AB	AB minus 1
H	H	L	L	logical 1	A plus A(1)
H	H	L	H	$A + \overline{B}$	$(A + \overline{B})$ plus A
H	H	H	L	A + B	$(A + \overline{B})$ plus A
H	H	H	H	A	A minus 1

Table2: Function Table for active high inputs and outputs

Logic Diagram



b. Write a program in 8085 assembly language to interface matrix keyboard.

Answer:

MVI A, 00H : Initialize keyboard/display in encoded
OUT 81H : scan keyboard 2 key lockout mode
MVI A, 34H
OUT 81H : Initialize prescaler count
MVI A, 0BH : Load mask pattern to enable RST 7.5
SIM : mask other interrupts
EI : Enable Interrupt
HERE: JMP HERE : Wait for the interrupt
Interrupt Subroutine:
MVI A, 40H : Initialize 8279 in read FIFO
OUT 81H : RAM mode
IN 80H : Read FIFO RAM (keycode)
EI : Enable Interrupt
RET : Return to main program

**8 x 4 Matrix Keyboard Interface

Statement: Interface an 8 x 4 matrix keyboard to 8085 through 8279.

SOFTWARE FOR INTERFACING 8x4 MATRIX KEYBOARD

Source program:

MVI A, 00H : Initialize keyboard/display in encoded
OUT 81H : scan keyboard 2 key lockout mode
MVI A, 34H
OUT 81H : Initialize prescaler count
MVI A, 0BH : Load mask pattern to enable RST 7.5
SIM : mask other interrupts
EI : Enable Interrupt
HERE: JMP HERE : Wait for the interrupt
Interrupt Subroutine:
MVI A, 40H : Initialize 8279 in read FIFO
OUT 81H : RAM mode
IN 80H : Read FIFO RAM (keycode)
EI : Enable Interrupt
RET : Return to main program

Q.7 a. How do the 8259A accomplish the interrupt activity?

Answer:

The device requiring service signals the PIC via one of the seven PIC interrupt request (IR) input lines. The corresponding bit in the PIC Interrupt Request register (IIR) is set.

- The PIC activates the INT line which is connected to the CPU INTR line.
- If the interrupts are not masked at the CPU, it finishes the currently executing instruction and sends one interrupt acknowledge (INTA) pulse to the PIC.
- In an X85 environment, the PIC responds by setting the highest priority In Service Register (ISR) bit and the corresponding IIR bit is reset. There is no PIC activity on the data bus in this cycle.

- The CPU will initiate a second INTA pulse. During this pulse, the PIC releases an 8-bit pointer on to the data bus where it is read by the CPU.
- The CPU reads the interrupt-type number, determines the associated address of the interrupt service routine (ISR), then fetches and executes the ISR.
- In the Automatic End Of Interrupt (AEOI) Mode the ISR bit is reset at the end of the second INTA pulse. Otherwise the ISR bit remains set until an appropriate EOI command is issued at the end of the ISR.

b. Discuss with a suitable pin diagram, the working of 8257. (8)

Answer:

The Intel® 8257 is a 4-channel direct memory access (DMA) controller. It is specifically designed to simplify the transfer of data at high speeds for the Intel® microcomputer systems. Its primary function is to generate, upon a peripheral request, a sequential memory address which will allow the peripheral to read or write data directly to or from memory. Acquisition of the system bus in accomplished via the CPU's hold function. The 8257 has priority logic that resolves the peripherals requests and issues a composite hold request to the CPU. It maintains the DMA cycle count for each channel and outputs a control signal to notify the peripheral that the programmed number of DMA cycles is complete. Other output control signals simplify sectored data transfers. The 8257 represents a significant savings in component count for DMA-based microcomputer systems and greatly simplifies the transfer of data at high speed between peripherals and memories.

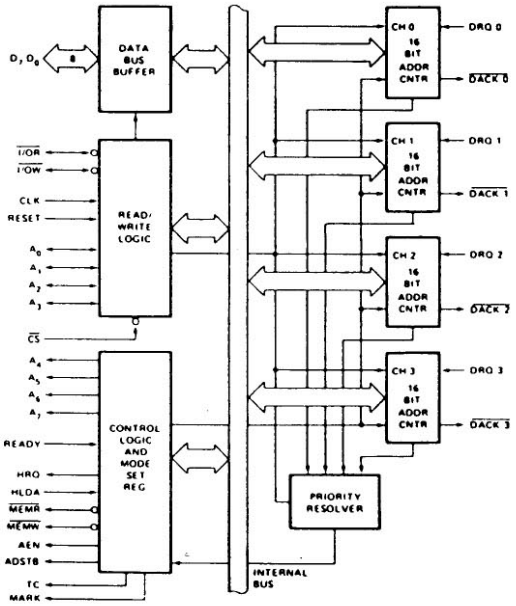


Figure Block Diagram

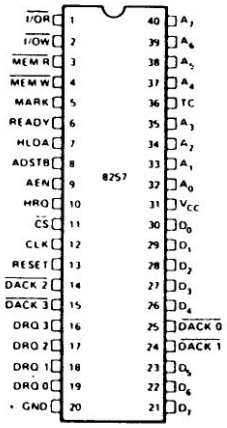


Figure Pin Configuration

The 8257 is a programmable, Direct Memory Access (DMA) device which, when coupled with a single Intel® 8212 I/O port device, provides a complete four-channel DMA controller for use in Intel® microcomputer systems. After being initialized by software, the 8257 can transfer a block of data, containing up to 16,384 bytes, between memory and a peripheral device directly, without further intervention required of the CPU. Upon receiving a DMA transfer request from an enabled peripheral, the 8257:

1. Acquires control of the system bus.
2. Acknowledges that requesting peripheral which is connected to the highest priority channel.
3. Outputs the least significant eight bits of the memory address onto system address lines A_0-A_7, outputs the most significant eight bits of the memory address to the 8212 I/O port via the data bus (the 8212 places these address bits on lines A_8-A_{15}), and
4. Generates the appropriate memory and I/O read/write control signals that cause the peripheral to receive or deposit a data byte directly from or to the addressed location in memory.

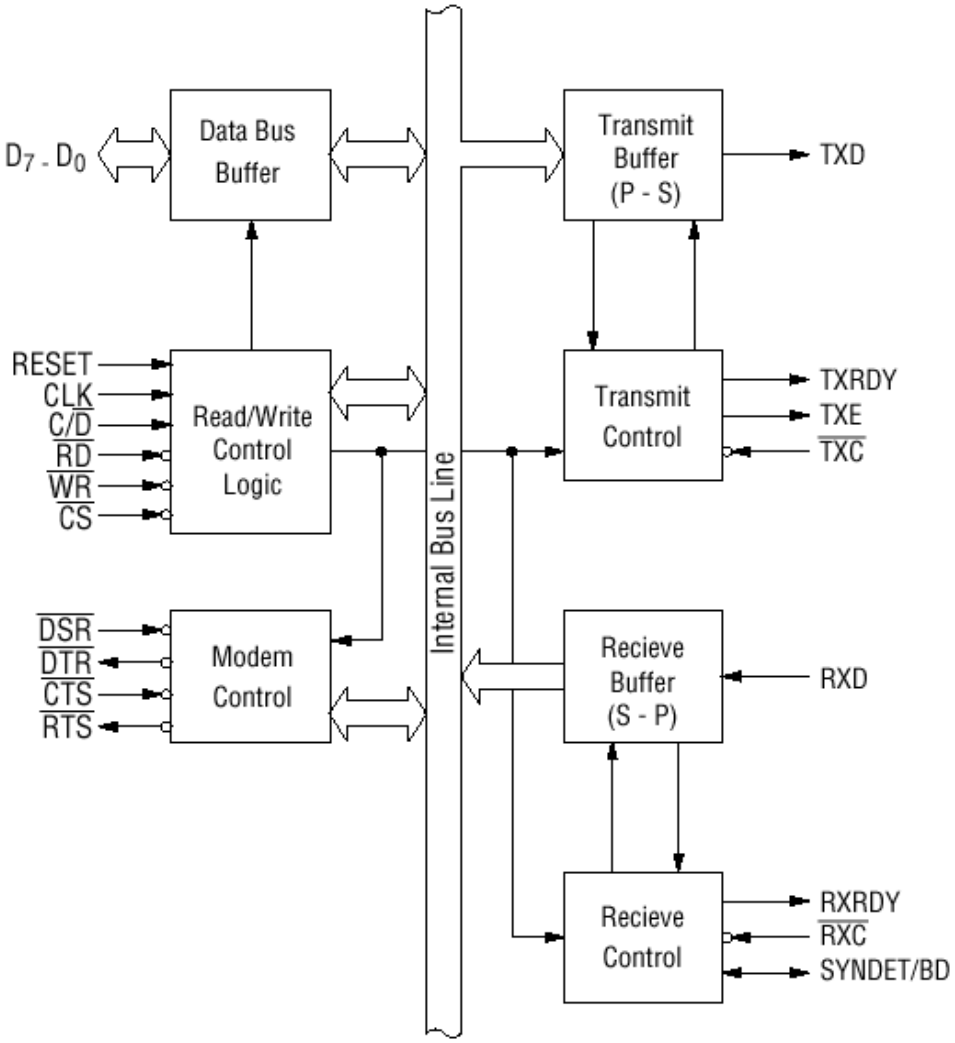
The 8257 will retain control of the system bus and repeat the transfer sequence, as long as a peripheral maintains its DMA request. Thus, the 8257 can transfer a block of data to/from a high speed peripheral (e.g., a sector of data on a floppy disk) in a single "burst". When the specified number of data bytes have been transferred, the 8257 activates its Terminal Count (TC) output, informing the CPU that the operation is complete.

The 8257 offers three different modes of operation: (1) DMA read, which causes data to be transferred from memory to a peripheral; (2) DMA write, which causes data to be transferred from a peripheral to memory; and (3) DMA verify, which does not actually involve the transfer of data. When an 8257 channel is in the DMA verify mode, it will respond the same as described for transfer operations, except that no memory or I/O read/write control signals will be generated, thus preventing the transfer of data. The 8257, however, will gain control of the system bus and will acknowledge the peripheral's DMA request for each DMA cycle. The peripheral can use these acknowledge signals to enable an internal access of each byte of a data block in order to execute some verification procedure, such as the accumulation of a CRC (Cyclic Redundancy Code) checkword. For example, a block of DMA verify cycles might follow a block of DMA read cycles (memory to peripheral) to allow the peripheral to verify its newly acquired data.

Q.8 a. Explain asynchronous transmission using 8251.

Answer:

The 8251 is a USART (Universal Synchronous Asynchronous Receiver Transmitter) for serial data communication. As a peripheral device of a microcomputer system, the 8251 receives parallel data from the CPU and transmits serial data after conversion. This device also receives serial data from the outside and transmits parallel data to the CPU after conversion.



Block diagram of the 8251 USART (Universal Synchronous Asynchronous Receiver Transmitter)

The 8251 functional configuration is programed by software. Operation between the 8251 and a CPU is executed by program control. Table 1 shows the operation between a CPU and the device.

$\overline{CS}$	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	
1	×	×	×	Data Bus 3-State
0	×	1	1	Data Bus 3-State
0	1	0	1	Status → CPU
0	1	1	0	Control Word ← CPU
0	0	0	1	Data → CPU
0	0	1	0	Data ← CPU

Table Operation between a CPU and 8251

Control Words

There are two types of control word.

1. Mode instruction (setting of function)
2. Command (setting of operation)

1) Mode Instruction

Mode instruction is used for setting the function of the 8251. Mode instruction will be in "wait for write" at either internal reset or external reset. That is, the writing of a control word after resetting will be recognized as a "mode instruction."

Items set by mode instruction are as follows:

- Synchronous/asynchronous mode
- Stop bit length (asynchronous mode)
- Character length
- Parity bit
- Baud rate factor (asynchronous mode)
- Internal/external synchronization (synchronous mode)
- Number of synchronous characters (Synchronous mode)

Asynchronous Mode (Transmission)

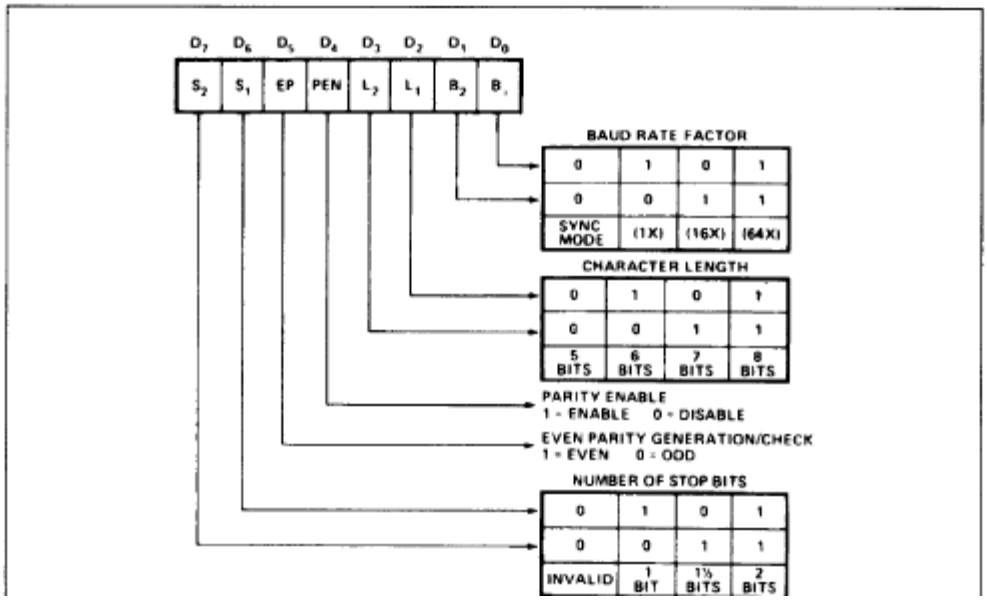
Whenever a data character is sent by the CPU the 8251A automatically adds a Start bit (low level) followed by the data bits (least significant bit first), and the programmed number of Stop bits to each character. Also, an even or odd Parity bit is inserted prior to the Stop bit(s), as defined by the Mode Instruction. The character is then transmitted as a serial data stream on the TxD output. The serial data is shifted out on the falling edge of $\overline{TxC}$ at a rate equal to 1, 1/16, or 1/64 that of the $\overline{TxC}$, as defined by the Mode Instruction. BREAK characters can be continuously sent to the TxD if commanded to do so.

When no data characters have been loaded into the 8251A the TxD output remains "high" (marking) unless a Break (continuously low) has been programmed.

Asynchronous Mode (Receive)

The RxD line is normally high. A falling edge on this line triggers the beginning of a START bit. The validity of this START bit is checked by again strobing this bit at its nominal center (16X or 64X mode only). If a low is detected again, it is a valid START bit, and the bit counter will start counting. The bit counter thus locates the center of the data bits, the parity bit (if it exists) and the stop bits. If parity error occurs, the parity error flag is set. Data and parity bits are sampled on the RxD pin with the rising edge of the $\overline{RxC}$. If a low level is detected as the STOP bit the Framing Error flag will be set. The STOP bit signals the end of a character. Note that the *receiver* requires only *one* stop bit, regardless of the number of stop bits programmed. This character is then loaded into the parallel I/O buffer of the 8251A. The RxRDY pin is raised to signal the CPU that a character is ready to be fetched. If a previous character has not been fetched by the CPU, the present character replaces it in the I/O buffer, and the OVERRUN Error flag

Asynchronous mode



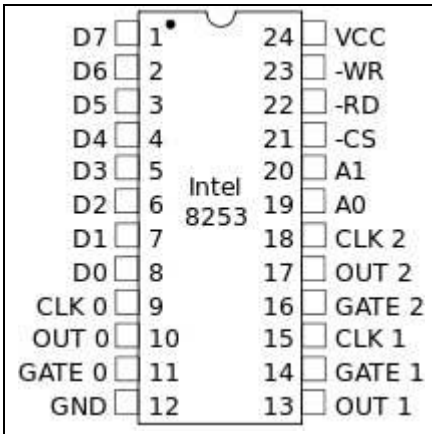
b. Explain the interfacing and programming of 8253 with 8085. (8)

Answer:

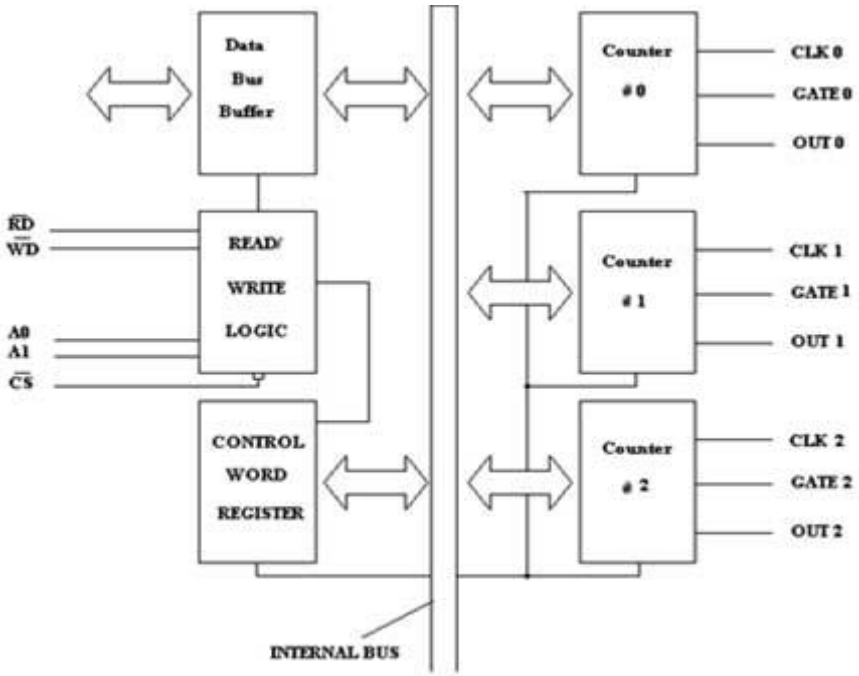
The Interfacing timer Intel **8253** with 8085

- Three independent 16 bit counters
- Input clock frequency 3 MHz
- Programmable counters mode
- Count binary or BCD

a) PIN DIAGRAM AND BLOCK DIAGRAM OF 8253



PIN DIAGRAM OF 8253



BLOCK DIAGRAM OF 8253

FUNCTIONAL DESCRIPTION

DATA BUS BUFFER

This 3-state, be directional, 8-bit buffer is used to interface the **8253** to the systems data bus. Data is transmitted or received by the buffer upon execution of Input and Output CPU instructions. The Data Bus Buffer has three basic functions.

- Programming the MODES of the **8253**
- Loading the count registers
- Reading the count values

READ/WRITE LOGIC:
The Read/Write Logic accepts inputs from the system bus and in turn generate control signals for overall device operation. It is enabled by **CS** so that no operation can occur to change the function unless the device has been selected by the system logic.

RD (READ)
A "low" on this input informs the **8253** that the CPU is inputting data in the form of a counters value.

WR (WRITE)
A "low" on this input informs the **8253** that the CPU is outputting data in the form of mode information or loading counters.

AO, A1
These inputs are normally connected to the address bus. Their function is to select one of the three counters to be operated on and to address the control word register for mode selection.

CS (CHIP SELECT)
A 'low' on this input enables the **8253**. No reading or writing will occur unless the device is selected. The CS Input has no elect upon the actual operation of the counters.

The control signals with which the **8253** interfaces with the CPU are **CS, RD, WR, A1, A2.** The basic operations performed by **8253** are determined by these control signals and are illustrated in the table given below.

CS	RD	WR	A1	A0	FUNCTION
0			0		Load counter 0
0	1	0	0	0	Load counter 1
0	1	0	1	1	Load counter 2
0	1	0	1	0	Write control word
0	1	0	0	1	Read counter 0
0	0	1	0	0	Read counter 1
0	0	1	1	1	Read counter 2
0	0	1	1	0	No – operation 3 state
0	0	1	X	1	Disabled 3 - state
0	X	X	X	X	No – operation 3 - state
	1	0		X	

WRITE OPERATION:

- Write a control word into control register.
- Load the low-order byte of a count in the counter register.
- Load the high-order byte of count in the counter register.

c) CONTROL WORD FOR 8253

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC- SELECT COUNTER:

0	0	Select counter 0
0	1	Select counter 1
1	0	Select counter 2
1	1	Illegal for 8253 read – back command for 8254

RW1 & RW0

RW1	RW0	SIGNIFICANT BYTE
0	0	Counter latch command (see read operations)
0	1	Read/write least significant byte only
1	0	Read/write most significant byte only
1	1	Read/write least significant byte first. Then most significant byte.

M-MODE:

0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

BCD VALUE:

0	Binary counter 16-bits
1	Binary coded decimal (BCD) counter (4 decades)

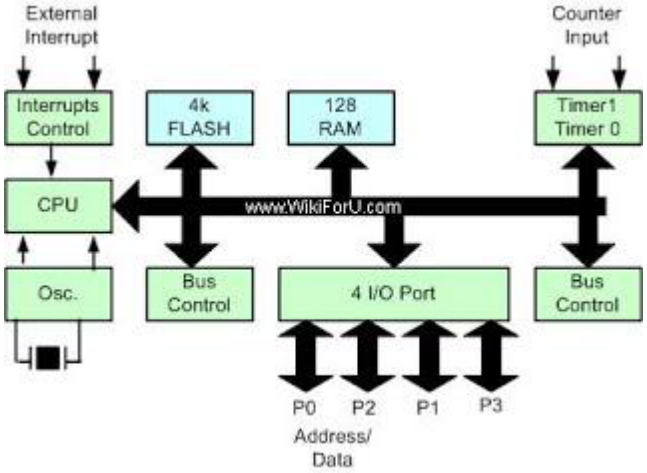
Q.9 a. Describe the functional block of 8051 with a neat diagram. (8)

Answer:

The 8051 Microcontroller is a Microcontroller designed by Intel in 1980's. It was based on Harvard Architecture and developed primarily for use in Embedded Systems. Originally it was developed using NMOS technology but as those requires more power to operate therefore Intel redesigned Microcontroller 8051 using CMOS technology and later versions came with a letter 'C' in their name, for example: 80C51. These latest Microcontrollers requires less power to operate as compared to their predecessors.

Microcontroller 8051 has two buses for program and data. Thus it has two memory spaces of 64K X 8 size for both program and data. It has an 8 bit processing unit and 8 bit accumulator. It also includes 8 bit B register as main processing blocks. It also have some other 8 bit and 16 bit registers.

Microcontroller 8051 have an built in RAM for internal processing. This memory is primary memory and is used for storage of temporary data. It is Volatile memory i.e. its contents get vanished when the power is turned OFF. Following is the block diagram of Microcontroller 8051. Let us have a look at each part or block of this Architecture:



Microcontroller 8051 - Block Diagram

Explanation:
Central Processor Unit(CPU): As you may know that CPU is the brain of any processing device.

It monitors and controls all operations that are performed in the Microcontroller. User have no control over the work of CPU. It reads program written in ROM memory and executes them and do the expected task.

Interrupts: As its name suggests, Interrupt is a subroutine call that interrupts Microcontroller's main operation or work and causes it to execute some another program which is more important at that time. The feature of Interrupt is very useful as it helps in cases of emergency. Interrupts gives us a mechanism to put on hold the ongoing operation , execute a subroutine and then again resumes normal program execution.

The Microcontroller 8051 can be configured in such a way that it temporarily terminates or pause the main program at the occurrence of interrupt. When subroutine is completed then the execution of main program starts as usual. There are five interrupt sources in 8051 Microcontroller. 2 of them are external interrupts, 2 timer interrupts and one serial port interrupt.

Memory: Microcontroller requires a program which is a collection of instructions. This program tells Microcontroller to do specific tasks. These programs requires a memory on which these can be saved and read by Microcontroller to perform specific operation. The memory which is used to store the program of Microcontroller, is known as code memory or Program memory . It is known as '**ROM**'(**Read Only Memory**). Microcontroller also requires a memory to store data or operands temporarily. The memory which is used to temporarily store data for operation is known as Data Memory and we uses '**RAM**'(**Random Access Memory**) for this purpose. Microcontroller 8051 has 4K of Code Memory or Program memory that is it has 4KB Rom and it also have 128 bytes of data memory i.e. RAM. **Bus**: Basically Bus is a collection of wires which work as a communication channel or medium for transfer of Data. These buses consists of 8, 16 or more wires. Thus these can carry 8 bits, 16 bits simultaneously. Buses are of two types:

- Address Bus
- Data Bus

Address Bus: Microcontroller 8051 has a 16 bit address bus. It used to address memory locations. It is used to transfer the address from CPU to Memory.

Data Bus: Microcontroller 8051 has 8 bits data bus. It is used to carry data.

Oscillator: As we know Microcontroller is a digital circuit device, therefore it requires clock for its operation. For this purpose, Microcontroller 8051 has an on-chip oscillator which works as a clock source for Central Processing Unit. As the output pulses of oscillator are stable therefore it enables synchronized work of all parts of 8051 Microcontroller.

Input/Output Port: As we know that Microcontroller is used in Embedded systems to control the operation of machines. Therefore to connect it to other machines, devices or peripherals we requires I/O interfacing ports in Microcontroller. For this purpose Microcontroller 8051 has 4 input output ports to connect it to other peripherals.

Timers/Counters: Microcontroller 8051 has 2 16 bit timers and counters. The counters are divided into 8 bit registers. The timers are used for measurement of intervals , to determine pulse width etc.

b. Explain the following instructions of 8051 with examples: (8)

(i) CJNE destination, source, label
(ii) MUL AB
(iii) RR A
(iv) SWAP A

Answer:

i) The **CJNE** instruction compares the first two operands and branches to the specified destination if their values are not equal. If the values are the same, execution continues with the next instruction.

CJNE destination, source, label

C	AC	F0	RS1	RS0	OV		P

Bytes	3		
Cycles	2		
Encoding	10110101	direct	offset

Operation

```
CJNE
PC = PC + 3
IF A <> (direct)
  PC = PC + offset
IF A < (direct)
  C = 1
ELSE
  C = 0
```

Example CJNE A, 60h, LABEL

(ii). The **MUL** instruction multiplies the unsigned 8-bit integer in the accumulator and the unsigned 8-bit integer in the B register producing a 16-bit product. The low-order byte of the product is returned in the accumulator. The high-order byte of the product is returned in the B register. The OV flag is set if the product is greater than 255 (0FFh), otherwise it is cleared. The carry flag is always cleared.

MUL AB

C	AC	F0	RS1	RS0	OV		P

Bytes	1
Cycles	4
Encoding	10100100
Operation	MUL AB BA = A * B
Example	MUL AB

(iii). The **RR** instruction rotates the eight bits in the accumulator right one bit position. Bit 0 of the accumulator is rotated into bit 7, bit 7 into bit 6, and so on. No flags are affected by this instruction.

RR A

C	AC	F0	RS1	RS0	OV		P

Bytes	1
Cycles	1
Encoding	00000011
Operation	RR $A_n = A_{n+1}$ where n = 0 to 6 $A_7 = A_0$
Example	RR A

(iv). The **SWAP** instruction exchanges the low-order and high-order nibbles within the accumulator. No flags are affected by this instruction.

SWAP A

C	AC	F0	RS1	RS0	OV		P

Bytes	1
Cycles	1
Encoding	11000100
Operation	SWAP A_{3-0} swap A_{7-4}
Example	SWAP A

TEXT BOOK

I. The 8085 Microprocessor; Architecture, Programming and Interfacing, K. Udaya Kumar and B. S. Umashankar, Pearson Education, 2008