UNIT I  8085 MICROPROCESSOR


PART-A

1. What is the need for ALE signal in 8085 microprocessor?

The ALE signal goes high at the beginning of each machine cycle indicating the availability of the address on the address bus, and the signal is used to latch the loworder address bus.

2. How many machine cycles are needed to execute STA 1800?

4 Machine cycles are needed.

3. What is the need for interfacing?

Generally I/O devices are slow devices. Therefore the speed of I/O devices does not match with the speed of microprocessor. And so an interface is provided between system bus and I/O devices.

4. Compare memory mapped I/O and peripheral mapped I/O.

Instead of a memory register, if an output device is connected at the address, the accumulator contents will be transferred to the output device. This is called memory mapped I/O.

5. State the disadvantages of memory mapped I/O scheme.

When I/O devices are memory mapped, some of the addresses are allotted to I/O devices and so the full address space cannot be used for addressing memory (i.e., physical memory address space will be reduced. Hence memory mapping is useful only for small systems, where the memory requirement is less.
6. What are the requirements to be met while interfacing I/O devices to microprocessor/microcontroller?

Use IO/M signal

IO device selection, Handshaking signals

7. Define interfacing.

Generally I/O devices are slow devices. Therefore the speed of I/O devices does not match with the speed of microprocessor. And so an interface is provided between system bus and I/O devices.

8. Show the common anode seven segments LED Configuration. How to switch it on and off?

In common anode, all anodes of LEDs are connected together.

9. Differentiate between software and hardware interrupts.

The Software interrupt is initiated by the main program, but the Hardware interrupt is initiated by an external device. In 8085, the Software interrupt cannot be disabled or masked but the Hardware interrupt except TRAP can be disabled or masked.

10. What is interrupt?

Interrupt is a signal send by an external device to the processor so as to request the processor to perform a particular task or work.

11. Name the vectored and non vectored interrupt of 8085 system.

When an interrupt is accepted, if the processor control branches to a specific address defined by the manufacturer then the interrupt is called vectored interrupt.

In Non-vectored interrupt there is no specific address for storing the interrupt service routine. Hence the interrupted device should give the address of the interrupt service routine.

12. What do you mean by timing diagram?

The timing diagram provides information regarding the status of various signals, when a machine cycle is executed. The knowledge of timing diagram is essential for system designer to select matched peripheral devices like memories, latches, ports, etc., to form a microprocessor system.
13. Define i) Instruction cycle

The sequence of operations that a processor has to carry out while executing the instruction is called Instruction cycle. Each instruction cycle of a processor indium consists of a number of machine cycles.

ii) Machine cycle

The processor cycle or machine cycle is the basic operation performed by the processor. To execute an instruction, the processor will run one or more machine cycles in a particular order.

14. Define T-state and In which T-cycle the ALE signal is activated?

T-State is defined as one subdivision of the operation performed in one clock period. These subdivisions are internal states synchronized with the system clock, and each T-State is precisely equal to one clock period. At the beginning of first T state.

15. What do you mean by masking the interrupt? How it is activated in 8085?

Masking is preventing the interrupt from disturbing the current program execution. When the processor is performing an important job (process) and if the process should not be interrupted then all the interrupts should be masked or disabled. In processor with multiple interrupts, the lower priority interrupt can be masked so as to prevent it from interrupting, the execution of interrupt service routine of higher priority interrupt.

16. List the main applications of 8 bit microprocessors?

LCD

Microcomputer

Keyboard display.

17. What is Address Bus?

The address is an identification number used by the microprocessor to identify or access a memory location or I/O device. It is an output signal from the processor. Hence the address bus is unidirectional.

18. What is System Bus?

Bus is a group of conducting lines that carries data, address and control signals.

19. What are the limitations of 8085 MPU?
(i) The lower order address bus of the 8085 microprocessor is multiplexed (time shared) with the data bus. The buses need to be demultiplexed.

(ii) Appropriate control signals need to be generated to interface memory and I/O with the 8085.

20. Why is the data bus bi-directional?

The microprocessor has to fetch (read) the data from memory or input device for processing and after processing, it has to store (write) the data to memory or output device. Hence the data bus is bi-directional.

21. What is a flag?

The data conditions, after arithmetic or logical operations, are indicated by setting or resetting the flip-flops called flags.

22. Why are the program counter and the stack pointer 16-bit registers?

Memory locations for the program counter and stack pointer have 16-bit addresses. So the PC and SP have 16-bit registers.

23. Explain the function of ALE and IO/M signals in the 8085 architecture?

The ALE signal goes high at the beginning of each machine cycle indicating the availability of the address on the address bus, and the signal is used to latch the loworder address bus. The IO/M signal is a status signal indicating whether the machine cycle is I/O or memory operation. The IO/M signal is combined with the RD and WR control signals to generate IOR, IOW, MEMW, MEMR.

24. Write down the control and status signals?

Two Control signals and three status signals

Control signals: RD and WR

Status signals: IO/M, S1, S2

25. Define T-state?

T-state is defined as one subdivision of the operation of performed in one clock period.
PART-B

THE 8085 AND 8086 MICROPROCESSORS

1. Explain the Internal architecture of the 8085 microprocessor?
   (Apr-2006,12 Marks)

The 8085 microprocessor is an 8-bit microprocessor with a 40 pin dual in line package. 8085 consists of various units and each unit performs its own functions. The various units of a microprocessor are listed below:

- Accumulator
- Arithmetic and logic Unit
- General purpose register
- Program counter
- Stack pointer
- Temporary register
- Flags
- Instruction register and Decoder
- Timing and Control unit
- Interrupt control
- Serial Input/output control
- Address buffer and Address-Data buffer
- Address bus and Data bus

Accumulator

Accumulator is nothing but a register which can hold 8-bit data. Accumulator aids in storing two quantities. The data to be processed by arithmetic and logic unit is stored in accumulator. It also stores the result of the operation carried out by the Arithmetic and Logic unit. The accumulator is also called an 8-bit register. The accumulator is connected to Internal Data bus and ALU (arithmetic and logic unit). The accumulator can be used to send or receive data from the Internal Data bus.

Arithmetic and Logic Unit
There is always a need to perform arithmetic operations like +, -, *, / and to perform logical operations like AND, OR, NOT etc. So there is a necessity for creating a separate unit which can perform such types of operations. These operations are performed by the Arithmetic and Logic Unit (ALU). ALU performs these operations on 8-bit data.

**General Purpose Registers**

These general purpose registers are used to hold data like any other registers. The general purpose registers in 8085 processors are B, C, D, E, H and L. Each register can hold 8-bit data. Apart from the above function these registers can also be used to work in pairs to hold 16-bit data. They can work in pairs such as B-C, D-E and H-L to store 16-bit data. The H-L pair works as a memory pointer. A memory pointer holds the address of a particular memory location. They can store 16-bit address as they work in pair.

**Program Counter and Stack Pointer**

**Program counter** is a special purpose register. A program counter stores the address of the next instruction to be executed. In other words the program counter keeps track of the memory address of the instructions that are being executed by the microprocessor and the memory address of the next instruction that is going to be executed.

**Stack pointer** is also a 16-bit register which is used as a memory pointer. A stack is nothing but the portion of RAM (Random access memory). Stack pointer maintains the address of the last byte that is entered into stack. Each time when the data is loaded into stack, Stack pointer gets decremented. Conversely it is incremented when data is retrieved from stack.

**Temporary Register:**

this register acts as a temporary memory during the arithmetic and logical operations. This temporary register can only be accessed by the microprocessor and it is completely inaccessible to programmers. Temporary register is an 8-bit register.

- FLAGS
- TIMING AND CONTROL UNIT
- INTERRUPT CONTROL

**Instruction Register and Decoder**

When such an instruction is fetched from memory, it is directed to Instruction register. So the instruction registers are specifically to store the instructions that are fetched from memory.

**Timing and Control Unit**

Timing and control unit is a very important unit as it synchronizes the registers and flow of data through various registers and other units. This unit consists of an oscillator and controller sequencer which sends control signals needed for internal and external control of data and other units.
The oscillator generates two-phase clock signals which aids in synchronizing all the registers of 8085 microprocessor. Signals that are associated with Timing and control unit are:

Control Signals: READY, RD’, WR’, ALE

Status Signals: S0, S1, IO/M’

DMA Signals: HOLD, HLDA

RESET Signals: RESET IN, RESET OUT

---

2. Give the format of flag Register. Explain each flag. (Apr-2005, 8 Marks)

**Flags**

Flags are nothing but a group of individual Flip-flops. The flags are mainly associated with arithmetic and logic operations. The flags will show either a logical (0 or 1) (i.e.) a set or reset depending on the data conditions in accumulator or various other registers. A flag is actually a latch which can hold some bits of information. It alerts the processor that some event has taken place. Intel processors have a set of 5 flags.

- Carry flag
- Parity flag
- Auxiliary carry flag
- Zero flag
- Sign flag

1. Carry flag

two binary numbers. For example:

1100 0000

1000 0000

When we add the above two numbers, a carry is generated in the most significant bit. The number in the extreme right is least significant bit, while the number in extreme left is most significant bit. So a ninth bit is generated due to the carry. For this purpose the Carry flag is used. The carry flag is set whenever a carry is generated and reset whenever there is no carry.

2. Auxiliary carry flag

0000 0100, 0000 0101

When we add both the numbers a carry is generated in the third bit from the least significant bit. This sets the auxiliary carry flag. When there is no carry, the auxiliary carry flag is reset.
3. Parity flag:

Parity checks whether it’s even or add parity. This flag returns a 0 if it is odd parity and returns a 1 if it is an even parity. Sometimes they are also called as parity bit which is used to check errors while data transmission is carried out.

4. Zero flag:

Zero flag shows whether the output of the operation is 0 or not. If the value of Zero flag is 0 then the result of operation is not zero. If it is zero the flag returns value 1.

5. Sign flag:

Sign flag shows whether the output of operation has positive sign or negative sign. A value 0 is returned for positive sign and 1 is returned for negative sign.

------------------------------------------------------------------------------------------------------------------

3. List the features of 8085. (Nov-2006,4 Marks)

1. It can accept ,process or provide 8bit data simultaneously.
2. It operates on+5V power supply connected at Vcc.
3. It can operate with a 3mHZ clock frequency.
4. It has 16 address lines hence it can access 64Kbytes of memory.
5. It provides 8bit I/O address to access 256 I/O ports.
6. It supports 74 Instructions with the following addressing modes.
   i) Immediate ii) Register iii) Direct iv) Indirect v) Implied
7. Lower 8bit address bus (A0-A7) and data bus (D0-D7) are multiplexed to reduce number of external pins. External hardware(latch) is required to separate address lines and data lines.
8. It provides 5 hardware interrupts.
   i) TRAP ii) RST 5.5 iii) RST6.5 iv) RST 4.5 v) INTR

====================================================================================================
4. **Explain in detail about Address buffer and Address-Data buffer (Nov-2005, 8 Marks)**

The contents of the stack pointer and program counter are loaded into the address buffer and address-data buffer. These buffers are then used to drive the external address bus and address-data bus. As the memory and I/O chips are connected to these buses, the CPU can exchange desired data to the memory and I/O chips.

The address-data buffer is not only connected to the external data bus but also to the internal data bus which consists of 8-bits. The address data buffer can both send and receive data from internal data bus.

**Address bus and Data bus:**

8085 is an 8-bit microprocessor. So the data bus present in the microprocessor is also 8-bits wide. So 8-bits of data can be transmitted from or to the microprocessor. But 8085 processor requires 16 bit address bus as the memory addresses are 16-bit wide.

The 8 most significant bits of the address are transmitted with the help of address bus and the 8 least significant bits are transmitted with the help of multiplexed address/data bus. The eight bit data bus is multiplexed with the eight least significant bits of address bus. The address/data bus is time multiplexed. This means for few microseconds, the 8 least significant bits of address are generated, while for next few seconds the same pin generates the data. This is called Time multiplexing.

But there are situations where there is a need to transmit both data and address simultaneously. For this purpose a signal called ALE (address latch enable) is used. ALE signal holds the obtained address in its latch for a long time until the data is obtained and so when the microprocessor sends the data next time the address is also available at the output latch. This technique is called Address/Data demultiplexing.

---

**1. Define addressing modes. Explain the addressing modes of 8085? (Apr-2008, 12 Marks)**
The process of identifying the operands for a particular instruction can be carried out in several ways. The various types of addressing modes are as shown below.

- Immediate addressing mode
- Direct addressing mode
- Register addressing mode
- Register indirect addressing mode
- Implicit addressing mode

**Immediate Addressing Mode:**

ADI 34H – This instruction adds the immediate data, 34H to the accumulator.

34H is the data here. H represents Hexadecimal value and the immediate value is added to the accumulator. In this case 34H is added to the accumulator. Suppose if accumulator has a value 8H and when this instruction is executed, 34H is added to the 8H and the result is stored in accumulator. In the above instruction the operand is specified within instruction itself.

**Direct Addressing Mode:**

There is a subtle difference between the direct addressing modes and immediate addressing modes. In immediate addressing mode the data itself is specified within instruction, but in direct addressing mode the address of the data is specified in the instruction.

**Example:**

LDA 4100H

STA 2000H

Consider the instruction STA 2000H

When this instruction is executed, the contents of the accumulator are stored in the memory location specified. In the above example the contents of accumulator are stored in memory location 2000H.
Register Addressing Mode:

In this type of addressing mode the instruction specifies the name of the register in which the data is available and Opcode specifies the name (or) address of the register on which the operation would be performed.

Example:

MOV A, B

Here the Opcode is MOV. If the above instruction is executed, the contents of Register B are moved to the Register A, which is nothing but the accumulator.

Register Indirect Addressing Mode:

This is indirect way of addressing. In this mode the instruction specifies the name of the register in which the address of the data is available.

Example:

MOV A, M

SUB M

MOV A, M. This instruction will move the contents of memory location, whose address is in H-L register pair to the accumulator.

M represents the address present in the H-L register pair. So when MOV A, M is executed, the contents of the address specified in H-L register pair are moved to accumulator.
Implicit Addressing Mode:

There are certain instructions in 8085 which does not require the address of the operand to perform the operation. They operate only upon the contents of accumulator.

Example:

1. CMA  2. RAL  3. RAR

CMA complements the contents of accumulator.

If RAL is executed the contents of accumulator is rotated left one bit through carry.

If RAR is executed the contents of accumulator is rotated right one bit through carry.

2. Explain the instruction sets of 8085? (Apr-2007, 12 marks)

Data Transfer Group:

The data transfer instructions move data between registers or between memory and registers.

MOV  Move

STA  Store Accumulator Directly in Memory

Arithmetic Group:
The arithmetic instructions add, subtract, increment, or decrement data in registers or memory.

ADD    Add to Accumulator
SUB    Subtract from Accumulator

Logical Group:

This group performs logical (Boolean) operations on data in registers and memory and on condition flags. The logical AND, OR, and Exclusive OR instructions enable you to set specific bits in the accumulator ON or OFF.

ANA    Logical AND with Accumulator
OR     Logical OR with Accumulator Using Immediate Data

Branch Group:

The branching instructions alter normal sequential program flow, either unconditionally or conditionally. The unconditional branching instructions are as follows:

JMP    Jump
CALL   Call
RET    Return

Stack I/O, and Machine Control Instructions:

PUSH   Push Two bytes of Data onto the Stack

The I/O instructions are as follows:

IN     Initiate Input Operation
OUT    Initiate Output Operation

The Machine Control instructions are as follows:

EI     Enable Interrupt System

7. Write a Program to find the Maximum number from a given array using 8085 (Nov2006,8 Marks)

LXI H, ADD1
MOV C, M  
INX H  
MOV A, M  
ADD2: INX H  
    CMP M  
    JNC LESS  
    MOV A, M  
LESS: DCR C  
    JNZ ADD2  
STA ADD_OUTPUT  

8. Write a Program to sort n numbers using 8085 (Nov2006, 8 Marks)  

Statement: a program to sort given 10 numbers from memory location 2200H in the ascending order  

Source program:  

MVI B, 09 : Initialize counter  
START : LXI H, 2200H: Initialize memory pointer  
MVI C, 09H : Initialize counter 2  
BACK: MOV A, M : Get the number  
INX H : Increment memory pointer  
CMP M : Compare number with next number  
JC SKIP : If less, don't interchange  
JZ SKIP : If equal, don't interchange  
MOV D, M  
MOV M, A  
DCX H  
MOV M, D  
INX H : Interchange two numbers  
SKIP: DCR C  
JNZ BACK  
DCR B  
JNZ START  
HLT :
9. Write a Program to sort n numbers by selection sort using 8085

(Apr-2008, 8 Marks)

LXI H, 0060H
MVI B, 05H
LOOP1: MOV C, B
MOV D, H
MOV E, L
INR E
LOOP2: LDAX D
CMP M
JNC NO_SWP
SWAP: MOV D, M
MOV M, A
MOV A, D
MOV D, H
STAX D
NO_SWP: INR E
DCR C
JNZ LOOP2
INX H
DCR B
JNZ LOOP1
HLT
1. What are the modes in which 8086 can operate?
The 8086 can operate in two modes and they are minimum (or uniprocessor) mode and maximum (or multiprocessor) mode.

2. What is the data and address size in 8086?
The 8086 can operate on either 8-bit or 16-bit data. The 8086 uses 20 bit address to access memory and 16-bit address to access I/O devices.

3. Explain the function of Memory I/O in 8086.
The signal Memory I/O is used to differentiate memory address and I/O address when the processor is accessing memory locations Memory I/O is asserted high and when it is accessing I/O mapped devices it is asserted low.

4. Write the flags of 8086.
The 8086 has nine flags and they are
1. Carry Flag (CF)
2. Parity Flag (PF)
3. Auxiliary carry Flag (AF)
4. Zero Flag (ZF)
5. Sign Flag (SF)
6. Overflow Flag (OF)
7. Trap Flag (TF)
8. Interrupt Flag (IF)
9. Direction Flag (DF)

5. What are the interrupts of 8086?
The interrupts of 8085 are INTR and NMI. The INTR is general maskable interrupt and NMI is non-maskable interrupt.

6. How clock signal is generated in 8086? What is the maximum internal clock frequency of 8086?
The 8086 does not have on-chip clock generation circuit. Hence the clock generator chip, 8284 is connected to the CLK pin of 8086. The clock signal supplied by 8284 is divided by three for internal use. The maximum internal clock frequency of 8086 is 5MHz.

7. Write the special functions carried by the general purpose registers of 8086.
The special functions carried by the registers of 8086 are the following.

Register Special function
1. AX 16-bit Accumulator
2. AL 8-bit Accumulator
3. BX Base Register
4. CX Count Register
5. DX Data Register

8. What is pipelined architecture?
In pipelined architecture the processor will have number of functional units and the execution time of functional units is overlapped. Each functional unit works independently most of the time.
9. What are the functional units available in 8086 architecture?

The bus interface unit and execution unit are the two functional units available in 8086 architecture.

10. List the segment registers of 8086.

The segment registers of 8086 are Code segment, Data segment, Stack segment and Extra segment registers.

11. Define machine cycle.

Machine cycle is defined as the time required to complete one operation of accessing memory, I/O, or acknowledging an external request. This cycle may consist of three to six T-states.


T-State is defined as one subdivision of the operation performed in one clock period. These subdivisions are internal states synchronized with the system clock, and each T-State is precisely equal to one clock period.

13. List the components of microprocessor (single board microcomputer) based system

The microprocessor based system consist of microprocessor as CPU, semiconductor memories like EPROM and RAM, input device, output device and interfacing devices.

14. Why interfacing is needed for I/O devices?

Generally I/O devices are slow devices. Therefore the speed of I/O devices does not match with the speed of microprocessor. And so an interface is provided between system bus and I/O devices.

15. What is the difference between CPU bus and system bus?

The CPU bus has multiplexed lines but the system bus has separate lines for each signal. (The multiplexed CPU lines are demultiplexed by the CPU interface circuit to form system bus).
1. With the neat sketch explain the architecture of 8086 processor? (Nov-2008,12Marks)

Memory

- Program, data and stack memories occupy the same memory space. The total addressable memory size is 1MB KB.
- As the most of the processor instructions use 16-bit pointers the processor can effectively address only 64 KB of memory. To access memory outside of 64 KB the CPU uses special
segment registers to specify where the code, stack and data 64 KB segments are positioned within 1 MB of memory (see the “Registers” section below).

**Program memory** – program can be located anywhere in memory. Jump and call instructions can be used for short jumps within currently selected 64 KB code segment, as well as for far jumps anywhere within 1 MB of memory. All conditional jump instructions can be used to jump within approximately +127 – -127 bytes from current instruction.

**Data memory** – the processor can access data in any one out of 4 available segments, which limits the size of accessible memory to 256 KB (if all four segments point to different 64 KB blocks). Accessing data from the Data, Code, Stack or Extra segments can be usually done by prefixing instructions with the DS:, CS:, SS: or ES: (some registers and instructions by default may use the ES or SS segments instead of DS segment).

Word data can be located at odd or even byte boundaries. The processor uses two memory accesses to read 16-bit word located at odd byte boundaries. Reading word data from even byte boundaries requires only one memory access.

**Stack memory** can be placed anywhere in memory. The stack can be located at odd memory addresses, but it is not recommended for performance reasons (see “Data Memory” above).

**Reserved locations:**

- 0000h – 03FFh are reserved for interrupt vectors. Each interrupt vector is a 32-bit pointer in format segment: offset.
- FFFF0h – FFFF0h – after RESET the processor always starts program execution at the FFFF0h address.

**Interrupts**

The processor has the following interrupts:

**INTR** is a maskable hardware interrupt. The interrupt can be enabled/disabled using STI/CLI instructions or using more complicated method of updating the FLAGS register with the help of the POPF instruction. When an interrupt occurs, the processor stores FLAGS register into stack, disables further interrupts, fetches from the bus one byte representing interrupt type, and jumps to interrupt processing routine address of which is stored in location 4 * <interrupt type>. Interrupt processing routine should return with the IRET instruction.

**NMI** is a non-maskable interrupt. Interrupt is processed in the same way as the INTR interrupt. Interrupt type of the NMI is 2, i.e. the address of the NMI processing routine is stored in location 0008h. This interrupt has higher priority then the maskable interrupt.

**Software interrupts** can be caused by:

- INT instruction – breakpoint interrupt. This is a type 3 interrupt.
- INT <interrupt number> instruction – any one interrupt from available 256 interrupts.
- INTO instruction – interrupt on overflow
- Single-step interrupt – generated if the TF flag is set. This is a type 1 interrupt. When the CPU processes this interrupt it clears TF flag before calling the interrupt processing routine.
- Processor exceptions: divide error (type 0), unused opcode (type 6) and escape opcode (type 7).

Software interrupt processing is the same as for the hardware interrupts.

**I/O ports**

**65536 8-bit I/O ports**: These ports can be also addressed as 32768 16-bit I/O ports.

---

**3. EXPLAIN IN DETAIL ABOUT 8086 REGISTERS** (Apr-2005, 12 Marks)

Most of the registers contain data/instruction offsets within 64 KB memory segment. There are four different 64 KB segments for instructions, stack, data and extra data. To specify where in 1 MB of processor memory these 4 segments are located the processor uses four segment registers:

**Code segment** (CS) is a 16-bit register containing address of 64 KB segment with processor instructions. The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register. CS register cannot be changed directly. The CS register is automatically updated during far jump, far call and far return instructions.

**Stack segment** (SS) is a 16-bit register containing address of 64KB segment with program stack. By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment. SS register can be changed directly using POP instruction.

**Data segment** (DS) is a 16-bit register containing address of 64KB segment with program data. By default, the processor assumes that all data referenced by general registers (AX, BX, CX, DX) and index register (SI, DI) is located in the data segment. DS register can be changed directly using POP and LDS instructions.

**Extra segment** (ES) is a 16-bit register containing address of 64KB segment, usually with program data. By default, the processor assumes that the DI register references the ES segment in string manipulation instructions. ES register can be changed directly using POP and LES instructions.

It is possible to change default segments used by general and index registers by prefixing instructions with a CS, SS, DS or ES prefix.

All general registers of the 8086 microprocessor can be used for arithmetic and logic operations. The general registers are:
**Accumulator** register consists of 2 8-bit registers AL and AH, which can be combined together and used as a 16-bit register AX. AL in this case contains the low-order byte of the word, and AH contains the high-order byte. Accumulator can be used for I/O operations and string manipulation.

**Base** register consists of 2 8-bit registers BL and BH, which can be combined together and used as a 16-bit register BX. BL in this case contains the low-order byte of the word, and BH contains the high-order byte. BX register usually contains a data pointer used for based, based indexed or register indirect addressing.

**Count** register consists of 2 8-bit registers CL and CH, which can be combined together and used as a 16-bit register CX. When combined, CL register contains the low-order byte of the word, and CH contains the high-order byte. Count register can be used as a counter in string manipulation and shift/rotate instructions.

**Data** register consists of 2 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX. When combined, DL register contains the low-order byte of the word, and DH contains the high-order byte. Data register can be used as a port number in I/O operations. In integer 32-bit multiply and divide instruction the DX register contains high-order word of the initial or resulting number.

The following registers are both general and index registers:

**Stack Pointer** (SP) is a 16-bit register pointing to program stack.

**Base Pointer** (BP) is a 16-bit register pointing to data in stack segment. BP register is usually used for based, based indexed or register indirect addressing.

**Source Index** (SI) is a 16-bit register. SI is used for indexed, based indexed and register indirect addressing, as well as a source data address in string manipulation instructions.

**Destination Index** (DI) is a 16-bit register. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions.

**Other registers:**

**Instruction Pointer** (IP) is a 16-bit register.

A **flag** is a 16-bit register containing 9 1-bit flags:

- Overflow Flag (OF) – set if the result is too large positive number, or is too small negative number to fit into destination operand.
- Direction Flag (DF) – if set then string manipulation instructions will auto-decrement index registers. If cleared then the index registers will be auto-incremented.
- Interrupt-enable Flag (IF) – setting this bit enables maskable interrupts.
- Single-step Flag (TF) – if set then single-step interrupt will occur after the next instruction.
- Sign Flag (SF) – set if the most significant bit of the result is set.
• Zero Flag (ZF) – set if the result is zero.
• Auxiliary carry Flag (AF) – set if there was a carry from or borrow to bits 0-3 in the AL register.
• Parity Flag (PF) – set if parity (the number of “1″ bits) in the low-order byte of the result is even.
• Carry Flag (CF) – set if there was a carry from or borrow to the most significant bit during last result calculation.

3. Explain the addressing modes of 8086 with the help of examples. (Apr-2008,12 Marks)

1.) Register Addressing – references the data in a register or in a register pair.

   MOV BX, CX
   MOV CL, BL

2) Immediate – the data is provided in the instruction.

   MOV BL, 26H
   MOV CX, 4567H

3) Direct Addressing – the instruction operand specifies the memory address where data is located.

   MOV CL, [9823H]

4) Register indirect – instruction specifies a register containing an address, where data is located. This addressing mode works with SI, DI, BX and BP registers.

   MOV [DI], BX

5 )Based Addressing – 8-bit or 16-bit instruction operand is added to the contents of a base register (BX or BP), the resulting value is a pointer to location where data resides.

   MOV AL, LAST [BX]

6 )Indexed Addressing– 8-bit or 16-bit instruction operand is added to the contents of an index register (SI or DI), the resulting value is a pointer to location where data resides.

   MOV BH, LAST [ST]
7 )Based Indexed – 8-bit or 16-bit instruction operand is added to the contents of a base register (BX or BP) and index register (SI or DI), the resulting value is a pointer to location where data resides.

MOV TOT [ST] [BX], CL

4 Explain the operation of max & min mode of 8086? (Nov-2008,10Marks)

SIGNAL DESCRIPTION OF 8086:
The Microprocessor 8086 is a 16-bit CPU available in different clock rates and packaged in a 40 pin CERDIP or plastic package.
• The 8086 operates in single processor or multiprocessor configuration to achieve high performance. The pins serve a particular function in minimum mode (single processor mode) and other function in maximum mode configuration (multiprocessor mode).
• The 8086 signals can be categorised in three groups. The first are the signal having common functions in minimum as well as maximum mode.
• The second are the signals which have special functions for minimum mode and third are the signals having special functions for maximum mode.

The following signal descriptions are common for both modes.

•AD15-AD0:
These are the time multiplexed memory I/O address and data lines.
• Address remains on the lines during T1 state, while the data is available on the data bus during T2, T3, Tw and T4.
• These lines are active high and float to a tristate during interrupt acknowledge and local bus hold acknowledge cycles.

•A19/S6,A18/S5,A17/S4,A16/S3:
These are the time multiplexed address and status lines.
• During T1 these are the most significant address lines for memory operations.
• During I/O operations, these lines are low. During memory or I/O operations, status information is available on those lines for T2,T3,Tw and T4.
• The status of the interrupt enable flag bit is updated at the beginning of each clock cycle.
• The S4 and S3 combinedly indicate which segment register is presently being used for memory accesses as in below fig.
• These lines float to tri-state off during the local bus hold acknowledge. The status line S6 is always low.
• The address bit are separated from the status bit using latches controlled by the ALE signal.

<table>
<thead>
<tr>
<th>S4</th>
<th>S3</th>
<th>Indication</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Alternate Data</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Stack</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Code or none</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Data</td>
</tr>
</tbody>
</table>
• **BHE /S7:**
  The bus high enable is used to indicate the transfer of data over the higher order (D15-D8) data bus as shown in table. It goes low for the data transfer over D15-D8 and is used to derive chip selects of odd address memory bank or peripherals. BHE is low during T1 for read, write and interrupt acknowledge cycles, whenever a byte is to be transferred on higher byte of data bus. The status information is available during T2, T3 and T4. The signal is active low and tristated during hold. It is low during T1 for the first pulse of the interrupt.

<table>
<thead>
<tr>
<th>BHE</th>
<th>(A_0)</th>
<th>Indication</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Whole word</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Upper byte from or to even address</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Lower byte from or to even address</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>None</td>
</tr>
</tbody>
</table>

• **RD Read:**
  This signal on low indicates the peripheral that the processor is performing memory or I/O read operation. RD is active low and shows the state for T2, T3, T4 of any read cycle. The signal remains tristated during the hold acknowledge.

• **READY:**
  This is the acknowledgement from the slow device or memory that they have completed the data transfer. The signal made available by the devices is synchronized by the 8284A clock generator to provide ready input to the 8086. The signal is active high.

• **INTR-Interrupt Request:**
  This is a triggered input. This is sampled during the last clock cycles of each instruction to determine the availability of the request. If any interrupt request is pending, the processor enters the interrupt acknowledge cycle.
  • This can be internally masked by resulting the interrupt enable flag. This signal is active high and internally synchronized.

• **TEST**
  This input is examined by a ‘WAIT’ instruction. If the TEST pin goes low, execution will continue, else the processor remains in an idle state. The input is synchronized internally during each clock cycle on leading edge of clock.

• **CLK- Clock Input:**
  The clock input provides the basic timing for processor operation and bus control activity. Its an asymmetric square wave with 33% duty cycle.

• **MN/ MX—**:
  The logic level at this pin decides whether the processor is to operate in either minimum or maximum mode.

**THE FOLLOWING PIN FUNCTIONS ARE FOR THE MINIMUM MODE OPERATION OF 8086.**

• **M/ IO – Memory /IO:**
This is a status line logically equivalent to S2 in maximum mode.
When it is low, it indicates the CPU is having an I/O operation, and when it is high, it indicates that the CPU is having a memory operation. This line becomes active high in the previous T4 and remains active till final T4 of the current cycle. It is tristated during local bus “hold acknowledge “.

**INTA : Interrupt Acknowledge:**
This signal is used as a read strobe for interrupt acknowledge cycles. i.e. when it goes low, the processor has accepted the interrupt.

**ALE – Address Latch Enable:**
This output signal indicates the availability of the valid address on the address/data lines, and is connected to latch enable input of latches. This signal is active high and is never tristated.

**DT/ R – Data Transmit/Receive:**
This output is used to decide the direction of data flow through the transreceivers (bidirectional buffers). When the processor sends out data, this signal is high and when the processor is receiving data, this signal is low.

**DEN – Data Enable:**
This signal indicates the availability of valid data over the address/data lines. It is used to enable the transreceivers (bidirectional buffers) to separate the data from the multiplexed address/data signal. It is active from the middle of T2 until the middle of T4. This is tristated during ‘hold acknowledge’ cycle.

**HOLD, HLDA- Acknowledge:**

---

**Write Cycle Timing Diagram for Minimum Mode**

- **ALE**
- **ADD/STATUS**
- **ADD/DATA**
- **WR**
- **DT/R**
- **DEN**

---
When the HOLD line goes high, it indicates to the processor that another master is requesting the bus access.

• The processor, after receiving the HOLD request, issues the hold acknowledge signal on HLDA pin, in the middle of the next clock cycle after completing the current bus cycle. At the same time, the processor floats the local bus and control lines. When the processor detects the HOLD line low, it lowers the HLDA signal. HOLD is an asynchronous input, and is should be externally synchronized.

• If the DMA request is made while the CPU is performing a memory or I/O cycle, it will release the local bus during T4 provided:
  1. The request occurs on or before T2 state of the current cycle.
  2. The current cycle is not operating over the lower byte of a word.
  3. The current cycle is not the first acknowledge of an interrupt acknowledge sequence.

5. Explain the operation of max mode of 8086? (Apr-2008, 12 Marks)

SIGNAL DESCRIPTION OF 8086:

The Microprocessor 8086 is a 16-bit CPU available in different clock rates and packaged in a 40 pin CERDIP or plastic package.

• The 8086 operates in single processor or multiprocessor configuration to achieve high performance. The pins serve a particular function in minimum mode (single processor mode) and other function in maximum mode configuration (multiprocessor mode).

• The 8086 signals can be categorised in three groups. The first are the signal having common functions in minimum as well as maximum mode.

• The second are the signals which have special functions for minimum mode and third are the signals having special functions for maximum mode.

The following signal descriptions are common for both modes.

• AD15-AD0:
  These are the time multiplexed memory I/O address and data lines.
  • Address remains on the lines during T1 state, while the data is available on the data bus during T2, T3, Tw and T4.
  • These lines are active high and float to a tristate during interrupt acknowledge and local bus hold acknowledge cycles.

• A19/S6, A18/S5, A17/S4, A16/S3:
  These are the time multiplexed address and status lines.
  • During T1 these are the most significant address lines for memory operations.
  • During I/O operations, these lines are low. During memory or I/O operations, status information is available on those lines for T2, T3, Tw and T4.
  • The status of the interrupt enable flag bit is updated at the beginning of each clock cycle.
  • The S4 and S3 combinedly indicate which segment register is presently being used
for memory accesses as in below fig.

• These lines float to tri-state off during the local bus hold acknowledge. The status line S6 is always low.
• The address bit are separated from the status bit using latches controlled by the ALE signal.

<table>
<thead>
<tr>
<th>S4</th>
<th>S3</th>
<th>Indication</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Alternate Data</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Stack</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Code or none</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Data</td>
</tr>
</tbody>
</table>

• **BHE /S7:**
  The bus high enable is used to indicate the transfer of data over the higher order (D15-D8) data bus as shown in table. It goes low for the data transfer over D15-D8 and is used to derive chip selects of odd address memory bank or peripherals. BHE is low during T1 for read, write and interrupt acknowledge cycles, whenever a byte is to be transferred on higher byte of data bus. The status information is available during T2, T3 and T4. The signal is active low and tristated during hold. It is low during T1 for the first pulse of the interrupt.

<table>
<thead>
<tr>
<th>BHE</th>
<th>A0</th>
<th>Indication</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Whole word</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Upper byte from or to even address</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Lower byte from or to even address</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>None</td>
</tr>
</tbody>
</table>

• **RD Read:**
  This signal on low indicates the peripheral that the processor is performing s memory or I/O read operation. RD is active low and shows the state for T2, T3, Tw of any read cycle. The signal remains tristated during the hold acknowledge.

• **READY:**
  This is the acknowledgement from the slow device or memory that they have completed the data transfer. The signal made available by the devices is synchronized by the 8284A clock generator to provide ready input to the 8086. The signal is active high.

• **INTR-Interrupt Request:**
  This is a triggered input. This is sampled during the last clock cycles of each instruction to determine the availability of the request. If any interrupt request is pending, the processor enters the interrupt acknowledge cycle.
  • This can be internally masked by resulting the interrupt enable flag. This signal is active high and internally synchronized.

• **TEST**
  This input is examined by a ‘WAIT’ instruction. If the TEST pin goes low, execution will continue, else the processor remains in an idle state. The input is
synchronized internally during each clock cycle on leading edge of clock.

• **CLK** - Clock Input:
The clock input provides the basic timing for processor operation and bus control activity. Its an asymmetric square wave with 33% duty cycle.

• **MN/MX**:
The logic level at this pin decides whether the processor is to operate in either minimum or maximum mode.

---

**THE FOLLOWING PIN FUNCTION ARE APPLICABLE FOR MAXIMUM MODE OPERATION OF 8086.**

---

• **S2, S1, S0** – Status Lines:
These are the status lines which reflect the type of operation, being carried out by the processor. These become active during T4 of the previous cycle and active during T1 and T2 of the current bus cycles.

---

<table>
<thead>
<tr>
<th>S2</th>
<th>S1</th>
<th>S0</th>
<th>Indication</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td><strong>Interrupt Acknowledge</strong></td>
</tr>
</tbody>
</table>
• **LOCK**: This output pin indicates that other system bus master will be prevented from gaining the system bus, while the LOCK signal is low.

  - The LOCK signal is activated by the ‘LOCK’ prefix instruction and remains active until the completion of the next instruction. When the CPU is executing a critical instruction which requires the system bus, the LOCK prefix instruction ensures that other processors connected in the system will not gain the control of the bus.
  - The 8086, while executing the prefixed instruction, asserts the bus lock signal output, which may be connected to an external bus controller.

• **QS₁, QS₀ – Queue Status:**

  These lines give information about the status of the code-prefetch queue. These are active during the CLK cycle after while the queue operation is performed.

  - This modification in a simple fetch and execute architecture of a conventional microprocessor offers an added advantage of pipelined processing of the instructions.
  - The 8086 architecture has 6-byte instruction prefetch queue. Thus even the largest (6-bytes) instruction can be prefetched from the memory and stored in the prefetch. This results in a faster execution of the instructions.
  - In 8085 an instruction is fetched, decoded and executed and only after the execution of this instruction, the next one is fetched.
  - By prefetching the instruction, there is a considerable speeding up in instruction execution in 8086. This is known as **instruction pipelining**.
  - At the starting the CS:IP is loaded with the required address from which the execution is to be started. Initially, the queue will be empty an the microprocessor starts a fetch operation to bring one byte (the first byte) of instruction code, if the CS:IP address is odd or two bytes at a time, if the CS:IP address is even.
  - The first byte is a complete opcode in case of some instruction (one byte opcode instruction) and is a part of opcode, in case of some instructions (two byte opcode instructions), the remaining part of code lie in second byte.
  - The second byte is then decoded in continuation with the first byte to decide the instruction length and the number of subsequent bytes to be treated as instruction data.
  - The queue is updated after every byte is read from the queue but the fetch cycle is initiated by BIU only if at least two bytes of the queue are empty and the EU may be concurrently executing the fetched instructions.
  - The next byte after the instruction is completed is again the first opcode byte of the next
instruction. A similar procedure is repeated till the complete execution of the program. • The fetch operation of the next instruction is overlapped with the execution of the current instruction. As in the architecture, there are two separate units, namely Execution unit and Bus interface unit.

• While the execution unit is busy in executing an instruction, after it is completely decoded, the bus interface unit may be fetching the bytes of the next instruction from memory, depending upon the queue status.

<table>
<thead>
<tr>
<th>QS&lt;sub&gt;1&lt;/sub&gt;</th>
<th>QS&lt;sub&gt;0&lt;/sub&gt;</th>
<th>Indication</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>No operation</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>First byte of the opcode from the queue</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Empty queue</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Subsequent byte from the queue</td>
</tr>
</tbody>
</table>

\textbf{RQ}/\overline{GT}, \overline{RQ}/\overline{GT} \textbf{Request/Grant}: These pins are used by the other local bus master in maximum mode, to force the processor to release the local bus at the end of the processor current bus cycle.

• Each of the pin is bidirectional with RQ/GT0 having higher priority than RQ/GT1.
• RQ/GT pins have internal pull-up resistors and may be left unconnected.

\textbf{Request/Grant sequence is as follows:}

1. A pulse of one clock wide from another bus master requests the bus access to 8086. 2. During T4(current) or T1(next) clock cycle, a pulse one clock wide from 8086 to the requesting master, indicates that the 8086 has allowed the local bus to float and that it will enter the ‘hold acknowledge’ state at next cycle. The CPU bus interface unit is likely to be disconnected from the local bus of the system.

3. A one clock wide pulse from the another master indicates to the 8086 that the hold request is about to end and the 8086 may regain control of the local bus at the next clock cycle. Thus each master to master exchange of the local bus is a sequence of 3 pulses. There must be at least one dead clock cycle after each bus exchange.

• The request and grant pulses are active low.
• For the bus request those are received while 8086 is performing memory or I/O cycle, the granting of the bus is governed by the rules as in case of HOLD and HLDA in minimum mode

\textbf{6. Explain in detail about Memory Management Unit (MMU) of 8086}

Memory Management Unit within a microprocessor converts the virtual memory address into a physical memory address. Virtual memory address is sometimes referred as logical memory address.
MMU can convert logical address into physical address in two ways.

1. Segment oriented approach:
   In this case, the logical memory space consists of memory segments of variable length. Each segment contains some data/program as is described by a "descriptor". The segment "descriptor" contains the base address of the segment, segment size and other attributes. The descriptors of all segments are stored in a "descriptor table" located in the physical memory. The logical address of a memory location contains two parts, viz, segment selector and offset. The segment selector points to the segment "descriptor" in the descriptor table. From the "descriptor", the base address of the segment is obtained. The offset part of the logical address is added to the segment base to generate physical address of the memory.

2. Page oriented approach:
   In page oriented approach, the logical address space is divided into pages of fixed length. A page has 4 Kbytes. Since the MMU deals with smaller chunk of memory (4 Kbytes), it is easier and faster to swap back and forth between the secondary storage and the physical memory.

7. Explain in detail about instruction set of 8086.(Nov-2008,12 Marks)

**Instruction Set**

8086 instruction set consists of the following instructions:

- Data moving instructions.
  - MOV
  - PUSH
  - POP
- XCHG
- XLAT
- Arithmetic – add, subtract, multiply, divide, increment, decrement, convert byte/word and compare.
  - ADD
  - ADC
  - INC
  - AAA
  - DAA
- Logic – AND, OR, exclusive OR, shift/rotate and test.
  - AND
  - OR
  - NOT
  - XOR
  - TEST
- String manipulation – load, store, move, compare and scan for byte/word.
  - LODS/LODSB/LODSW
  - STOS/STOSB/STOSW
  - MOV$S/ MOVSB/ MOVSN
  - CMPS/CMPSB/CMPSW
- Control transfer – conditional, unconditional, call subroutine and return from subroutine.
  - CALL
  - RET
  - JMP
  - J cond

6. Draw pin diagram of 8086
UNIT III MICROPROCESSOR PERIPHERAL INTERFACING

Introduction, Generation of I/O Ports, Programmable Peripheral Interface (PPI)-Intel 8255, Sample-and-Hold Circuit and Multiplexer, Keyboard and Display Interface, Keyboard and Display Controller (8279), Programmable Interval timers (Intel 8253, 8254), D-to-A converter, A-to-D converter, CRT Terminal Interface, Printer Interface.

PART-A

1. Write down the function of OBF in 8255.

   Output Buffer Full function determines whether the bidirectional data transfer is allowed or not.

2. Define PPI.

   The 8255A is a widely used, programmable, parallel I/O device. It can be programmed to transfer data under various conditions, from simple I/O to interrupt I/O.

3. Explain the advantages of PIC chips in microprocessor based systems.

   To increase the interrupt handling capacity of the microprocessor.

4. Explain the working of receiver part of USART.

   It manages all receiver related activities. Along with data reception, it does false start bit detection, parity error detection, framing error detection, sync detection and break detection.

5. What is key debouching?

   Key bouncing may cause multiple entries made for the same key. To overcome this problem after a key press is sensed the device is made to wait for few milliseconds. Then the key is checked again to ensure it is still pressed. If it is still pressed it is taken as a valid key press. This process is called keyboard debouching.

6. How much current is needed to drive an LED? Draw a typical driver circuit for it?(2)

   A current of between 5 and 30mA to light.

7. What is the count value needed to program the 8254 to generate a delay of 1 ms?

   Count-1000, frequency-1khz
8. **Name any two type of ADCs.**

   The different types of ADC are successive approximation ADC, Counter type ADC flash type ADC, integrator converters and voltage-to-frequency converters.

9. **For a A/D converter circuit why Vref should be stabilized supply.**

   To regulate the supply.

10. **Which is the fastest ADC and why?**

    Flash type ADC.

11. **What do you mean by Quantization error?**

    The difference between an analog wave and its digital representation. Also known as "quantization noise.".

12. **What is the difference between A/D and D/A converters?**

    Digital-to-analog conversion is to pull the samples from memory and convert them into an impulse train.

    An ADC is attempting to capture and convert a largely unknown signal into a known representation. In contrast, a DAC is taking a fully known, well-understood representation and "simply" generating an equivalent analog value.

    the challenge for an ADC is much greater than it is for a DAC. To get the most out of an ADC, especially a higher-performance one (speed or precision) takes a well-designed analog signal-conditioning input channel, often with an ADC driver carefully matched to the ADC itself.

    The DAC's life is much easier. But that relative ease shouldn't encourage complacency on the designer's part. It's too easy not to give the analog output of the DAC the attention it needs, regarding parameters such as slew rate, output drive (voltage, current, range) and protection against faults at its load. And that can lead to nasty circuit and system-level headaches, at both the prototype evaluation and in the field.

13. **Define the following terms for D/A converters:**

    i) Resolution
Resolution of a converter determines the degree of accuracy in conversion. It is equal to $1/2^n$.

ii) Accuracy

**Accuracy** is the degree to which information on a map or in a digital database matches true or accepted values. Accuracy is an issue pertaining to the quality of data and the number of errors contained in a dataset or map. In discussing a GIS database, it is possible to consider horizontal and vertical accuracy with respect to geographic position, as well as attribute, conceptual, and logical accuracy.

iii) Monotonicity

If a clock has monotonicity, then each successive time reading from that clock will yield a time further in the future than the previous reading.

iv) Conversion time

The time required by an analog to digital converter to fully convert an analog input sample.

14. List the features of 8279.

- It has built-in hardware to provide key debounce.
- It provides two output modes for display interface.
- It provides three input modes for keyboard interface.

15. Compare parallel and serial type of data transfer.

In parallel communication number of lines required to transfer data depend on the number of bits to be transferred.

In serial communication one bit is transferred at a time over a single line.

16. Why the number of out ports in the peripheral-mapped I/O is restricted to 256 ports?

The number of output ports in the peripheral I/O is restricted to 256 ports because the operand of the OUT instruction is 8-bits; it can have only 256 combinations.

17. What are the control signals necessary in the memory mapped I/O?

MEMR, MEMW
18. What happens when the 8085 execute the out instruction?

When the 8085 executes the out instruction, in the third machine cycle, it places the output port address on the low-order address bus, duplicates the same port address on the high-order bus, places the contents of the accumulator on the data bus and asserts the control signal WR.

PART-B

1. Draw the Block diagram and explain the operations of 8251 serial communication interface.

INTERFACING WITH INTEL 8251A (USART)

- The 8251A is a programmable serial communication interface chip designed for synchronous and asynchronous serial data communication.
- It supports the serial transmission of data.
- It is packed in a 28 pin DIP.

Block Diagram:

The functional block diagram of 8251A consists of five sections. They are:
- Read/Write control logic
- Transmitter
- Receiver
- Data bus buffer
- Modem control.

The functional block diagram is shown in fig:

Read/Write control logic:

- The Read/Write Control logic interfaces the 8251A with CPU, determines the functions of the 8251A according to the control word written into its control register.
- It monitors the data flow.

- This section has three registers and they are control register, status register and data buffer.
- The active low signals RD, WR, CS and C/D(Low) are used for read/write operations with these three registers.

- When C/D(low) is high, the control register is selected for writing control word or reading status word.
- When C/D(low) is low, the data buffer is selected for read/write operation.

- When the reset is high, it forces 8251A into the idle mode.

- The clock input is necessary for 8251A for communication with CPU and this clock does not control either the serial transmission or the reception rate.

Transmitter section:

- The transmitter section accepts parallel data from CPU and converts them into serial data.

- The transmitter section is double buffered, i.e., it has a buffer register to hold an 8-bit parallel data and another register called output register to convert the parallel data into serial bits.

- When output register is empty, the data is transferred from buffer to output register. Now the processor can again load another data in buffer register.

- If buffer register is empty, then TxRDY is goes to high.

- If output register is empty then TxEMPTY goes to high.

- The clock signal, TxC (low) controls the rate at which the bits are transmitted by the USART.

- The clock frequency can be 1,16 or 64 times the baud rate.

Receiver Section:

- The receiver section accepts serial data and convert them into parallel data

- The receiver section is double buffered, i.e., it has an input register to receive serial data and convert to parallel, and a buffer register to hold the parallel data.

- When the RxD line goes low, the control logic assumes it as a START bit, waits for half a bit time and samples the line again.

- If the line is still low, then the input register accepts the following bits, forms a character and loads it into the buffer register.
  - The CPU reads the parallel data from the buffer register.

- When the input register loads a parallel data to buffer register, the RxRDY line goes high.

- The clock signal RxC (low) controls the rate at which bits are received by the USART.

- During asynchronous mode, the signal SYNDET/BRKDET will indicate the break in the data transmission.
During synchronous mode, the signal SYNDET/BRKDET will indicate the reception of synchronous character.

MODEM Control:

- The MODEM control unit allows to interface a MODEM to 8251A and to establish data communication through MODEM over telephone lines.
- This unit takes care of handshake signals for MODEM interface.

INTERFACING WITH INTEL 8251A (USART)

- The 8251A can be either memory mapped or I/O mapped in the system.
- 8251A in I/O mapped in the system is shown in the figure.
- Using a 3-to-8 decoder generates the chip select signals for I/O mapped devices.
- The address lines A4, A5 and A6 are decoded to generate eight chip select signals (IOCS-0 to IOCS-7) and in this, the chip select signal IOCS-2 is used to select 8251A.
- The address line A7 and the control signal IO/M(low) are used as enable for decoder.
- The address line A0 of 8085 is connected to C/D(low) of 8251A to provide the internal addresses.
- The data lines D0 - D7 are connected to D0 - D7 of the processor to achieve parallel data transfer.
- The RESET and clock signals are supplied by the processor. Here the processor clock is directly connected to 8251A. This clock controls the parallel data transfer between the processor and 8251A.
- The output clock signal of 8085 is divided by suitable clock dividers like programmable timer 8254 and then used as clock for serial transmission and reception.
- The TTL logic levels of the serial data lines and the control signals necessary for serial transmission and reception are converted to RS232 logic levels using MAX232 and then terminated on a standard 9-pin D-type connector.
- In 8251A the transmission and reception baud rates can be different or same.

2. Draw the Block diagram of 8279 and explain the functions of each block.
KEYBOARD/DISPLAY CONTROLLER - INTEL 8279

The INTEL 8279 is specially developed for interfacing keyboard and display devices to 8085/8086/8088 microprocessor based system. The important features of 8279 are,

- Simultaneous keyboard and display operations.
- Scanned keyboard mode.
- Scanned sensor mode.
- 8-character keyboard FIFO.
- 1 6-character display.
- Right or left entry 1 6-byte display RAM.
- Programmable scan timing.

Block diagram of 8279:

- The functional block diagram of 8279 is shown.

- The four major sections of 8279 are keyboard, scan, display and CPU interface.
Keyboard section:

- The keyboard section consists of eight return lines RL0 - RL7 that can be used to form the columns of a keyboard matrix.
- It has two additional input: shift and control/strobe. The keys are automatically debounced.
- The two operating modes of keyboard section are 2-key lockout and N-key rollover.
- In the 2-key lockout mode, if two keys are pressed simultaneously, only the first key is recognized.
- In the N-key rollover mode simultaneous keys are recognized and their codes are stored in FIFO.
- The keyboard section also have an 8 x 8 FIFO (First In First Out) RAM.
- The FIFO can store eight key codes in the scan keyboard mode. The status of the shift key and control key are also stored along with key code. The 8279 generate an interrupt signal when there is an entry in FIFO. The format of key code entry in FIFO for scan keyboard mode is, 

![Keyboard section diagram](image)

- In sensor matrix mode the condition (i.e., open/close status) of 64 switches is stored in FIFO RAM. If the condition of any of the switches changes then the 8279 asserts IRQ as high to interrupt the processor.

Display section:

- The display section has eight output lines divided into two groups A0-A3 and B0-B3.
- The output lines can be used either as a single group of eight lines or as two groups of four lines, in conjunction with the scan lines for a multiplexed display.
- The output lines are connected to the anodes through driver transistor in case of common cathode 7-segment LEDs.
- The cathodes are connected to scan lines through driver transistors.
- The display can be blanked by BD (low) line.
- The display section consists of 16 x 8 display RAM. The CPU can read from or write into any location of the display RAM.

Scan section:

- The scan section has a scan counter and four scan lines, SL0 to SL3.
- In decoded scan mode, the output of scan lines will be similar to a 2-to-4 decoder.
In encoded scan mode, the output of scan lines will be binary count, and so an external decoder should be used to convert the binary count to decoded output.

The scan lines are common for keyboard and display.

The scan lines are used to form the rows of a matrix keyboard and also connected to digit drivers of a multiplexed display, to turn ON/OFF.

CPU interface section:

- The CPU interface section takes care of data transfer between 8279 and the processor.
- This section has eight bidirectional data lines DB0 to DB7 for data transfer between 8279 and CPU.
- It requires two internal address A =0 for selecting data buffer and A = 1 for selecting control register of 8279.
- The control signals WR (low), RD (low), CS (low) and A0 are used for read/write to 8279.
- It has an interrupt request line IRQ, for interrupt driven data transfer with processor.
- The 8279 require an internal clock frequency of 100 kHz. This can be obtained by dividing the input clock by an internal prescaler.
- The RESET signal sets the 8279 in 16-character display with two-key lockout keyboard modes.

Programming the 8279:

- The 8279 can be programmed to perform various functions through eight command words.

INTERFACING OF 8279 WITH 8085

In a microprocessor system, when keyboard and 7-segment LED display is interfaced using ports or latches then the processor has to carry the following task.

- Keyboard scanning
- Key debouncing
- Key code generation
- Sending display code to LED
- Display refreshing

3. WITH A NEAT DIAGRAM EXPLAIN HOW 8279 IS INTERFACED WITH 8085 AND USED FOR SERIAL (Nov 2008, 12 Marks)

INTERFACING OF 8279 WITH 8085

In a microprocessor system, when keyboard and 7-segment LED display is interfaced using ports or latches then the processor has to carry the following task.

- Keyboard scanning
- Key debouncing
- Key code generation
- Sending display code to LED
- Display refreshing

Interfacing 8279 with 8085 processor:

- A typical Hexa keyboard and 7-segment LED display interfacing circuit using 8279 is shown.
- The circuit can be used in 8085 microprocessor system and consist of 16 numbers of hexa-keys and 6 numbers of 7-segment LEDs.
- The 7-segment LEDs can be used to display six digit alphanumeric character.
- The 8279 can be either memory mapped or I/O mapped in the system. In the circuit shown is the 8279 is I/O mapped.
- The address line A0 of the system is used as A0 of 8279.
- The clock signal for 8279 is obtained by dividing the output clock signal of 8085 by a clock divider circuit.
- The chip select signal is obtained from the I/O address decoder of the 8085 system. The chip select signals for I/O mapped devices are generated by using a 3-to-8 decoder.
- The address lines A4, A5 and A6 are used as input to decoder.
- The address line A7 and the control signal IO/M (low) are used as enable for decoder.
- The chip select signal IOCS-3 is used to select 8279.
- The I/O address of the internal devices of 8279 are shown in table
The circuit has 6 numbers of 7-segment LEDs and so the 8279 has to be programmed in encoded scan. (Because in decoded scan, only 4 numbers of 7-segment LEDs can be interfaced)

In encoded scan the output of scan lines will be binary count. Therefore an external, 3-to-8 decoder is used to decode the scan lines SL0, SL1 and SL2 of 8279 to produce eight scan lines S0 to S7.

The decoded scan lines S0 and S1 are common for keyboard and display.

The decoded scan lines S2 to S5 are used only for display and the decoded scan lines S6 and S7 are not used in the system.

Anode and Cathode drivers are provided to take care of the current requirement of LEDs.

The pnp transistors, BC 158 are used as driver transistors.

The anode drivers are called segment drivers and cathode drivers are called digit drivers.

The 8279 output the display code for one digit through its output lines (OUT A0 to OUT A3 and OUT B0 to OUT B3) and send a scan code through, SL0- SL3.

The display code is inverted by segment drivers and sent to segment bus.

The scan code is decoded by the decoder and turns ON the corresponding digit driver. Now one digit of the display character is displayed. After a small interval (10 milli-second, typical), the display is turned OFF (i.e., display is blanked) and the above process is repeated for next digit. Thus multiplexed display is performed by 8279.

The keyboard matrix is formed using the return lines, RL0 to RL3 of 8279 as columns and decoded scan lines S0 and S1 as rows.

A hexa key is placed at the crossing point of each row and column. A key press will short the row and column. Normally the column and row line will be high.

During scanning the 8279 will output binary count on SL0 to SL3, which is decoded by decoder to make a row as zero. When a row is zero the 8279 reads the columns. If there is a key press then the corresponding column will be zero.

If 8279 detects a key press then it wait for debounce time and again read the columns to generate key code.

In encoded scan keyboard mode, the 8279 stores an 8-bit code for each valid key press. The keycode consist of the binary value of the column and row in which the key is found and the status of shift and control key.

After a scan time, the next row is made zero and the above process is repeated and so on. Thus 8279 continuously scan the keyboard.

---

4. Draw the block diagram of programmable interrupt controller and explain its operations. Mention the feature of 8259  (Apr-2006,12 Marks)

PROGRAMMABLE INTERRUPT CONTROLLER - INTEL 8259
FUNCTIONAL BLOCK DIAGRAM OF 8259:

It has eight functional blocks. They are,

1. Control logic
2. Read Write logic
3. Data bus buffer
4. Interrupt Request Register (IRR)
5. In-Service Register (ISR)
6. Interrupt Mask Register (IMR)
7. Priority Resolver (PR)
8. Cascade buffer.

The data bus and its buffer are used for the following activities.

1. The processor sends control word to data bus buffer through D0-D7.
2. The processor read status word from data bus buffer through D0-D7
3. From the data bus buffer the 8259 send type number (in case of 8086) or the call opcode and address (in case of 8085) through D0-D7 to the processor.

- The processor uses the RD (low), WR (low) and A0 to read or write 8259.
- The 8259 is selected by CS (low).
- The IRR has eight input lines (IR0-IR7) for interrupts. When these lines go high, the request is stored in IRR. It registers a request only if the interrupt is unmasked.
• Normally IR0 has highest priority and IR7 has the lowest priority. The priorities of the interrupt request input are also programmable.
• First the 8259 should be programmed by sending Initialization Command Word (ICW) and Operational Command Word (OCW). These command words will inform 8259 about the following:

1. Type of interrupt signal (Level triggered / Edge triggered).
2. Type of processor (8085/8086).
3. Call address and its interval (4 or 8)
4. Masking of interrupts.
5. Priority of interrupts.
6. Type of end of interrupts.

- The interrupt mask register (IMR) stores the masking bits of the interrupt lines to be masked. The relevant information is send by the processor through OCW.
- The in-service register keeps track of which interrupt is currently being serviced.
- The priority resolver examines the interrupt request, mask and in-service registers and determines whether INT signal should be sent to the processor or not.
- The cascade buffer/comparator is used to expand the interrupts of 8259.
- In cascade connection one 8259 will be directly interrupting 8086 and it is called master 8259.
- To each interrupt request input of master 8259 (IR0-IR7), one slave 8259 can be connected. The 8259s interrupting the master 8259 are called slave 8259s.
- Each 8259 has its own addresses so that each 8259 can be programmed independently by sending command words and independently the status bytes can be read from it.

FEATURES OF 8259:

1. It is programmed to work with either 8085 or 8086 processor.
2. It manage 8-interrupts according to the instructions written into its control registers.
3. In 8086 processor, it supplies the type number of the interrupt and the type number is programmable. In 8085 processor, the interrupt vector address is programmable. The priorities of the interrupts are programmable.
4. The interrupts can be masked or unmasked individually.
5. The 8259s can be cascaded to accept a maximum of 64 interrupts.

5. With a neat diagram explain how 8259 is interfaced with 8085 and cascading of 8259 (Apr 2008,12 Marks)

CASCADING OF 8259 & INTERFACING OF 8259 WITH 8085

CASCADING 8259:

- The cascade pins (CAS0, CAS1 and CAS2) from the master are connected to the corresponding pins of the slave.
- For the slave 8259, the SP (low) / EN (low) pin is tied low to let the device know that it is a slave.
- The SP (low) / EN (low) pin can be used as input or output signal.
- In non-buffered mode it is used as input signal and tied to logic-I in master 8259 and logic-0 in slave 8259.
- In buffered mode it is used as output signal to disable the data buffers while data is transferred from 8259A to the CPU.

- It requires two internal address and they are A = 0 or A = 1.
- It can be either memory mapped or I/O mapped in the system. The interfacing of 8259 to 8085 is shown in figure is I/O mapped in the system.
- The low order data bus lines D0-D7 are connected to D0-D7 of 8259.
- The address line A0 of the 8085 processor is connected to A0 of 8259 to provide the internal address.
- The 8259 require one chip select signal. Using 3-to-8 decoder generates the chip select signal for 8259.
- The address lines A4, A5 and A6 are used as input to decoder.
- The control signal IO/M (low) is used as logic high enables for decoder and the address line A7 is used as logic low enable for decoder.
- The I/O addresses of 8259 are shown in table.

<table>
<thead>
<tr>
<th>Binary Address</th>
<th>Decoder input/enable</th>
<th>Input to address pin of 8259</th>
<th>Hexa address</th>
</tr>
</thead>
<tbody>
<tr>
<td>For A0 of 8259 to be zero</td>
<td>0 0 0</td>
<td>0 x x x 0</td>
<td>00</td>
</tr>
<tr>
<td>For A0 of 8259 to be one</td>
<td>0 0 0</td>
<td>0 x x x 1</td>
<td>01</td>
</tr>
</tbody>
</table>

Note: Don't care "x" is considered as zero.

Working of 8259 with 8085 processor:

- First the 8259 should be programmed by sending Initialization Command Word (ICW) and Operational Command Word (OCW). These command words will inform 8259 about the following,
  1. Type of interrupt signal (Level triggered / Edge triggered).
  2. Type of processor (8085/8086).
  3. Call address and its interval (4 or 8)
  4. Masking of interrupts.
  5. Priority of interrupts.
  6. Type of end of interrupts.

- Once 8259 is programmed it is ready for accepting interrupt signal. When it receives an interrupt through any one of the interrupt lines IR0-IR7 it checks for its priority and also checks whether it is masked or not.
- If the previous interrupt is completed and if the current request has highest priority and unmasked, then it is serviced.
- For servicing this interrupt the 8259 will send INT signal to INTR pin of 8085.
- In response it expects an acknowledge INTA (low) from the processor.
- When the processor accepts the interrupt, it sends three INTA (low) one by one.
- In response to first, second and third INTA (low) signals, the 8259 will supply CALL opcode, low byte of call address and high byte of call address respectively. Once the processor receives the call opcode and its address, it saves the content of program counter (PC) in stack and load the CALL address in PC and start executing the interrupt service routine stored in this call address.
6. Explain the various modes of operation of programmable Peripheral Interface.

PROGRAMMABLE PERIPHERAL INTERFACE - INTEL 8255

Pins, Signals and internal block diagram of 8255:

- It has 40 pins and requires a single +5V supply.

PIN DESCRIPTION - INTEL 8255

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D₀ - D₇</td>
<td>Data lines</td>
</tr>
<tr>
<td>RESET</td>
<td>Reset input</td>
</tr>
<tr>
<td>CS</td>
<td>Chip select</td>
</tr>
<tr>
<td>RD</td>
<td>Read control</td>
</tr>
<tr>
<td>WR</td>
<td>Write control</td>
</tr>
<tr>
<td>A₀ - A₁</td>
<td>Internal address</td>
</tr>
<tr>
<td>PA₀ - PA₇</td>
<td>Port-A pins</td>
</tr>
<tr>
<td>PB₀ - PB₇</td>
<td>Port-B pins</td>
</tr>
<tr>
<td>PC₀ - PC₇</td>
<td>Port-C pins</td>
</tr>
<tr>
<td>Vcc</td>
<td>+5V</td>
</tr>
<tr>
<td>Vss</td>
<td>0V (GND)</td>
</tr>
</tbody>
</table>
• The internal block diagram of 8255 is shown in fig:

- The INTEL 8255 is a device used to parallel data transfer between processor and slow peripheral devices like ADC, DAC, keyboard, 7-segment display, LCD, etc.
- The 8255 has three ports: Port-A, Port-B and Port-C.
- Port-A can be programmed to work in any one of the three operating modes mode-0, mode-1 and mode-2 as input or output port.
- Port-B can be programmed to work either in mode-0 or mode-1 as input or output port.
- Port-C (8-pins) has different assignments depending on the mode of port-A and port-B.
- If port-A and B are programmed in mode-0, then the port-C can perform any one of the following functions.
  - As 8-bit parallel port in mode-0 for input or output.
  - As two numbers of 4-bit parallel ports in mode-0 for input or output.
  - The individual pins of port-C can be set or reset for various control applications.
- If port-A is programmed in mode-1/mode-2 and port-B is programmed in mode-1 then some of the pins of port-C are used for handshake signals and the remaining pins can be used as input/ output lines or individually set/reset for control applications.
- The read/write control logic requires six control signals. These signals are given below.
- RD (low): This control signal enables the read operation. When this signal is low, the microprocessor reads data from a selected I/O port of the 8255A.
- 2. WR (low): This control signal enables the write operation. When this signal goes low, the microprocessor writes into a selected I/O port or the control register.
- 3. RESET: This is an active high signal. It clears the control register and sets all ports in the input mode.
- 4. CS (low), A0 and A1: These are device select signals. They are,

<table>
<thead>
<tr>
<th>Internal Devices</th>
<th>A1</th>
<th>A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port A</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Port B</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Port C</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Control Register</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Interfacing of 8255 with 8085 processor:

- A simple schematic for interfacing the 8255 with 8085 processor is shown in fig.

The 8255 can be either memory mapped or I/O mapped in the system. In the schematic shown in above is I/O mapped in the system.
- Using a 3-to-8 decoder generates the chip select signals for I/O mapped devices.
- The address lines A4, A5 and A6 are decoded to generate eight chip select signals (IOCS-0 to IOCS-7) and in this, the chip select IOCS-1 is used to select 8255.
- The address line A7 and the control signal IO/M (low) are used as enable for the decoder.
- The address line A0 of 8085 is connected to A0 of 8255 and A1 of 8085 is connected to A1 of 8255 to provide the internal addresses.
- The data lines D0-D7 are connected to D0-D7 of the processor to achieve parallel data transfer.
- The I/O addresses allotted to the internal devices of 8255 are listed in table.

<table>
<thead>
<tr>
<th>Internal Device</th>
<th>Binary Address</th>
<th>Hexa Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port-A</td>
<td>0 0 0 1</td>
<td>10</td>
</tr>
<tr>
<td>Port-B</td>
<td>0 0 0 1</td>
<td>11</td>
</tr>
<tr>
<td>Port-C</td>
<td>0 0 0 1</td>
<td>12</td>
</tr>
<tr>
<td>Control Register</td>
<td>0 0 0 1</td>
<td>13</td>
</tr>
</tbody>
</table>

Note: Don't care "x" is considered as zero.
UNIT IV 8 BIT MICROCONTROLLER- H/W ARCHITECTURE, INSTRUCTION SET AND PROGRAMMING

Introduction to 8051 Micro-controller, Architecture, Memory organization, Special function registers, Port Operation, Memory Interfacing, I/O Interfacing, Programming 8051 resources, interrupts, Programmer’s model of 8051, Operand types, Operand addressing, Data transfer instructions, Arithmetic instructions, Logic instructions, Control transfer instructions, Programming

PART-A

1. What is Microcontroller?

A device which contains the microprocessor with integrated peripherals like memory, serial ports, parallel ports, timer/counter, interrupt controller, data acquisition interfaces like ADC,DAC is called microcontroller.

2. List the features of 8051 microcontroller.

The features are

1. single_supply +5 volt operation using HMOS technology.
2. 4096 bytes program memory on chip(not on 8031)
3. 128 data memory on chip.
4. Four register banks.
5. Two multiple mode, 16-bit timer/counter.
6. Extensive boolean processing capabilities.
7. 64 KB external RAM size

3. Name any four additional hardware features available in microcontrollers when compared to microprocessors.

Two multiple mode, 16 bit timers/counters, Four register banks, integrated Boolean processor.

4. List out the Hardware Resources available in 8051.

4096 bytes on chip program memory
128 bytes onchip data memory on chip

5. When 8051 is reset, all interrupts are disabled. How to enable these interrupts?

INTR must be kept low &ISR should be enable.
6. What is nested interrupts?

When the interrupt is acknowledged, it sets the corresponding bit in ISR.

7. How will you double the baud rate in 8051?

By adjusting the machine cycles.

8. Explain software and hardware methods to start and stop timers in 8051.

When a timer is used to measure time it is also called an "interval timer" since it is measuring the time of the interval between two events.

9. Give steps to program 8051 for serial data transfer.

IN
OUT
DATA
JMP
HLT

10. Write short notes on interrupt priority.

ISR-Interrupt service routine stores all the levels that are currently being serviced.

11. Write the vector address and priority sequence of 8051 interrupts

The interrupts are:

Vector address
External interrupt 0 : IE0 : 0003H
Timer interrupt 0 : TF0 : 000BH
External interrupt 1 : IE1 : 0013H
Timer Interrupt 1 : TF1 : 001BH
Serial Interrupt
Receive interrupt : RI : 0023H
Transmit interrupt: TI : 0023H

12. Explain the operating mode 0 of 8051 serial ports?
In this mode serial enters & exits through RXD, TXD outputs
the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud
rate is fixed at 1/12 the oscillator frequency.

13. Explain the operating mode2 of 8051 serial ports?

In this mode 11 bits are transmitted (through TXD) or received
(through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data
bit, & a stop bit (1). On transmit the 9th data bit (TB* in SCON) can be
assigned the value of 0 or 1. Or for eg,: the parity bit (P, in the PSW) could be
moved into TB8. On receive the 9th data bit go in to the RB8 in Special
Function Register SCON, while the stop bit is ignored. The baud rate is
programmable to either 1/32 or 1/64 the oscillator frequency.

14. Explain the mode3 of 8051 serial ports?

In this mode, 11 bits are transmitted (through TXD) or
received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable
9th data bit, & a stop bit (1). In fact, Mode3 is the same as Mode2 in all
respects except the baud rate. The baud rate in Mode3 is variable.
In all the four modes, transmission is initiated by any instruction
that uses SBUF as a destination register. Reception is initiated in Mode0 by
the condition RI=0 & REN=1. Reception is initiated in other modes by the incoming start bit if REN=1.

PART-B
1. Describe the architecture of 8051 with a neat diagram.

8051 employs Harvard architecture. It has some peripherals such as 32 bit digital I/O, Timers and Serial I/O. The basic architecture of 8051 is given in fig 5.1

Various features of 8051 microcontroller are given as follows.
8-bit CPU
16-bit Program Counter
8-bit Processor Status Word (PSW)
8-bit Stack Pointer
Internal RAM of 128 bytes
Special Function Registers (SFRs) of 128 bytes
32 I/O pins arranged as four 8-bit ports (P0 - P3)
Two 16-bit timer/counters : T0 and T1
Two external and three internal vectored interrupts
One full duplex serial I/O

In 8051, each instruction cycle has six states (S 1 - S 6 ). Each state has two pulses (P1 and P2)
128 bytes of Internal RAM Structure (lower address space)
Fig 5.3: Internal RAM Structure
2. EXPLAIN IN DETAIL ABOUT EXTERNAL MEMORY AND PROGRAM MEMORY.

The lower 32 bytes are divided into 4 separate banks. Each register bank has 8 registers of one byte each. A register bank is selected depending upon two bank select bits in the PSW register. Next 16 bytes are bit addressable. In total, 128 bits (16X8) are available in bitaddressable area. Each bit can be accessed and modified by suitable instructions. The bit addresses are from 00H (LSB of the first byte in 20H) to 7FH (MSB of the last byte in 2FH). Remaining 80 bytes of RAM are available for general purpose.

Internal Data Memory and Special Function Register (SFR) Map

![Internal Data Memory Map](image)

Fig 5.4 : Internal Data Memory Map

The special function registers (SFRs) are mapped in the upper 128 bytes of internal data memory address. Hence there is an address overlap between the upper 128 bytes of data RAM and SFRs. Please note that the upper 128 bytes of data RAM are present only in the 8052 family. The lower 128 bytes of RAM (00H - 7FH) can be accessed both by direct or indirect addressing while the upper 128 bytes of RAM (80H - FFH) are accessed by indirect addressing. The SFRs (80H - FFH) are accessed by direct addressing only. This feature distinguishes the upper 128 bytes of memory from the SFRs, as shown in fig 5.4.

SFR Map

The set of Special Function Registers (SFRs) contains important registers such as Accumulator, Register B, I/O Port latch registers, Stack pointer, Data Pointer, Processor Status Word (PSW) and various control registers. Some of these registers are bit addressable.

Address

Bit address of 'b' bit of register 'R' is

Address of register 'R' + b

where 0 ≤ b ≤ 7
Processor Status Word (PSW)  Address=D0H

| CY | AC | F0 | RS1 | RS0 | OV | - | P |

Fig 5.6: Processor Status Word

PSW register stores the important status conditions of the microcontroller. It also stores the bank select bits (RS1 & RS0) for register bank selection.

Interfacing External Memory

If external program/data memory is to be interfaced, they are interfaced in the following way.

Fig 6.1: Circuit Diagram for Interfacing of External Memory

External program memory is fetched if either of the following two conditions is satisfied.

1. $\overline{EA}$ (Enable Address) is low. The microcontroller by default starts searching for program from external program memory.

2. PC is higher than FFFH for 8051 or 1FFFH for 8052.
tells the outside world whether the external memory fetched is program memory or data memory. \( \text{EA} \) is user configurable. \( \text{PSEN} \) is processor controlled.

3. Explain the timers & counter of 8051.

Timers / Counters

8051 has two 16-bit programmable UP timers/counters. They can be configured to operate either as timers or as event counters. The names of the two counters are T0 and T1 respectively. The timer content is available in four 8-bit special function registers, viz, TL0, TH0, TL1 and TH1 respectively.

In the "timer" function mode, the counter is incremented in every machine cycle. Thus, one can think of it as counting machine cycles. Hence the clock rate is 1/12 th of the oscillator frequency. In the "counter" function mode, the register is incremented in response to a 1 to 0 transition at its corresponding external input pin (T0 or T1). It requires 2 machine cycles to detect a high to low transition. Hence maximum count rate is 1/24 th of oscillator frequency. The operation of the timers/counters is controlled by two special function registers, TMOD and TCON respectively. Timer Mode control (TMOD) Special Function Register: TMOD register is not bit addressable.

<table>
<thead>
<tr>
<th>Gate</th>
<th>C/T</th>
<th>M1</th>
<th>M0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timer 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timer 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Various bits of TMOD are described as follows - Gate: This is an OR Gate enabled bit which controls the effect of \( \text{INT1} \) on START/STOP of Timer. It is set to one (‘1’) by the program to enable the interrupt to start/stop the timer. If \( \text{TR1} \) in TCON is set and signal on \( \text{INT1} \) pin is high then the timer starts counting using either internal clock (timer mode) or external pulses (counter mode).

\( \overline{C/T} \): It is used for the selection of Counter/Timer mode.

<table>
<thead>
<tr>
<th>Mode</th>
<th>M1</th>
<th>M0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mode 1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Mode 2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Mode 3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Mode Select Bits:

M1 and M0 are mode select bits.

Timer/ Counter control logic:
Timer control (TCON) Special function register:

TCON is bit addressable. The address of TCON is 88H. It is partly related to Timer and partly to interrupt.

Fig 8.2 TCON Register

The various bits of TCON are as follows.

TF1 : Timer1 overflow flag. It is set when timer rolls from all 1s to 0s. It is cleared when processor vectors to execute ISR located at address 001BH.
TR1 : Timer1 run control bit. Set to 1 to start the timer / counter.
TF0 : Timer0 overflow flag. (Similar to TF1)
TR0 : Timer0 run control bit.
IE1 : Interrupt1 edge flag. Set by hardware when an external interrupt edge is detected. It is cleared when interrupt is processed.
IE0 : Interrupt0 edge flag. (Similar to IE1)
IT1 : Interrupt1 type control bit. Set/cleared by software to specify falling edge / low level triggered external interrupt.
IT0 : Interrupt0 type control bit. (Similar to IT1)

As mentioned earlier, Timers can operate in four different modes. They are as follows

Timer Mode-0:

In this mode, the timer is used as a 13-bit UP counter as follows.
The lower 5 bits of TLX and 8 bits of THX are used for the 13 bit count. Upper 3 bits of TLX are ignored. When the counter rolls over from all 0's to all 1's, TFX flag is set and an interrupt is generated. The input pulse is obtained from the previous stage. If TR1/0 bit is 1 and Gate bit is 0, the counter continues counting up. If TR1/0 bit is 1 and Gate bit is 1, then the operation of the counter is controlled by INTX input. This mode is useful to measure the width of a given pulse fed to INTX input.

Timer Mode-1:

This mode is similar to mode-0 except for the fact that the Timer operates in 16-bit mode.

Timer Mode-2: (Auto-Reload Mode)

This is a 8 bit counter/timer operation. Counting is performed in TLX while THX stores a constant value. In this mode when the timer overflows i.e. TLX becomes FFH, it is fed with the value stored in THX. For example if we load THX with 50H then the timer in mode 2 will count from 50H to FFH. After that 50H is again reloaded. This mode is useful in applications like fixed time sampling.

Timer Mode-3:

Timer 1 in mode-3 simply holds its count. The effect is same as setting TR1=0. Timer0 in mode-3 establishes TL0 and TH0 as two separate counters.
Control bits TR1 and TF1 are used by Timer-0 (higher 8 bits) (TH0) in Mode-3 while TR0 and TF0 are available to Timer-0 lower 8 bits(TL0).

4. Explain the interrupt structure of 8051.

Interrupts

8051 provides 5 vectored interrupts. They are -

\[\text{INT0}\]

TF0

\[\text{INT1}\]

TF1

RI/TI

Out of these, \[\text{INT0}\] and \[\text{INT1}\] are external interrupts whereas Timer and Serial port interrupts are generated internally. The external interrupts could be negative edge triggered or low level triggered. All these interrupt, when activated, set the corresponding interrupt flags. Except for serial interrupt, the interrupt flags are cleared when the processor branches to the Interrupt Service Routine (ISR). The external interrupt flags are cleared on branching to Interrupt Service Routine (ISR), provided the interrupt is negative edge triggered. For low level triggered external interrupt as well as for serial interrupt, the corresponding flags have to be cleared by software by the programmer.

The schematic representation of the interrupts is as follows -

Interrupt Vector Location
Each of these interrupts can be individually enabled or disabled by 'setting' or 'clearing' the corresponding bit in the IE (Interrupt Enable Register) SFR. IE contains a global enable bit EA which enables/disables all interrupts at once.

Interrupt Enable register (IE): Address: A8H

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EA</td>
<td></td>
<td>ET2</td>
<td>ES</td>
<td>ET1</td>
<td>EX1</td>
<td>ET0</td>
<td>EX0</td>
</tr>
</tbody>
</table>

- EX0 — interrupt (External) enable bit
- ET0 — Timer-0 interrupt enable bit
- EX1 — interrupt (External) enable bit
- ET1 — Timer-1 interrupt enable bit
- ES — Serial port interrupt enable bit
- ET2 — Timer-2 interrupt enable bit
- EA — Enable/Disable all

Setting '1' — Enable the corresponding interrupt

Setting '0' — Disable the corresponding interrupt
Priority level structure:

Each interrupt source can be programmed to have one of the two priority levels by setting (high priority) or clearing (low priority) a bit in the IP (Interrupt Priority) Register. A low priority interrupt can itself be interrupted by a high priority interrupt, but not by another low priority interrupt. If two interrupts of different priority levels are received simultaneously, the request of higher priority level is served. If the requests of the same priority level are received simultaneously, an internal polling sequence determines which request is to be serviced. Thus, within each priority level, there is a second priority level determined by the polling sequence, as follows.

**Interrupt Priority register (IP)**

<table>
<thead>
<tr>
<th>Source</th>
<th>Priority level</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE0</td>
<td>Highest</td>
</tr>
<tr>
<td>TF0</td>
<td></td>
</tr>
<tr>
<td>IE1</td>
<td>Lowest</td>
</tr>
<tr>
<td>TF1</td>
<td></td>
</tr>
<tr>
<td>RI+TI</td>
<td></td>
</tr>
</tbody>
</table>

Interrupt handling:

The interrupt flags are sampled at P2 of S5 of every instruction cycle (Note that every instruction cycle has six states each consisting of P1 and P2 pulses). The samples are polled during the next machine cycle (or instruction cycle). If one of the flags was set at S5P2 of the preceding instruction cycle, the polling detects it and the interrupt process generates a long call (LCALL) to the appropriate vector location of the interrupt. The LCALL is generated provided this hardware generated LCALL is not blocked by any one of the following conditions.

An interrupt of equal or higher priority level is already in progress.

The current polling cycle is not the final cycle in the execution of the instruction in progress.

The instruction in progress is RETI or any write to IE or IP registers.

When an interrupt comes and the program is directed to the interrupt vector address, the Program Counter (PC) value of the interrupted program is stored (pushed) on the stack. The required Interrupt Service Routine (ISR) is executed. At the end of the ISR, the instruction RETI returns the value of the PC from the stack and the originally interrupted program is resumed.
Reset is a non-maskable interrupt. A reset is accomplished by holding the RST pin high for at least two machine cycles. On resetting the program starts from 0000H and some flags are modified as follows:

<table>
<thead>
<tr>
<th>Register</th>
<th>Value(Hex) on Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>0000H</td>
</tr>
<tr>
<td>DPTR</td>
<td>0000H</td>
</tr>
<tr>
<td>A</td>
<td>00H</td>
</tr>
<tr>
<td>B</td>
<td>00H</td>
</tr>
<tr>
<td>SP</td>
<td>07H</td>
</tr>
<tr>
<td>PSW</td>
<td>00H</td>
</tr>
<tr>
<td>Ports P0-3 Latches</td>
<td>FFH</td>
</tr>
<tr>
<td>IP</td>
<td>XXX 00000 b</td>
</tr>
<tr>
<td>IE</td>
<td>0 XX 00000 b</td>
</tr>
<tr>
<td>TCON</td>
<td>00H</td>
</tr>
<tr>
<td>TMOD</td>
<td>00H</td>
</tr>
<tr>
<td>TH0</td>
<td>00H</td>
</tr>
<tr>
<td>TL0</td>
<td>00H</td>
</tr>
<tr>
<td>TH1</td>
<td>00H</td>
</tr>
<tr>
<td>TL1</td>
<td>00H</td>
</tr>
<tr>
<td>SCON</td>
<td>00H</td>
</tr>
<tr>
<td>SBUF</td>
<td>XX H</td>
</tr>
<tr>
<td>PCON</td>
<td>0 XXXX XXX b</td>
</tr>
</tbody>
</table>
5. **Explain the special function registers in 8051.**

Special Function Register (SFR) Memory

Special Function Registers (SFRs) are areas of memory that control specific functionality of the 8051 processor. For example, four SFRs permit access to the 8051's 32 input/output lines. Another SFR allows a program to read or write to the 8051's serial port. Other SFRs allow the user to set the serial baud rate, control and access timers, and configure the 8051's interrupt system.

When programming, SFRs have the illusion of being Internal Memory. For example, if you want to write the value "1" to Internal RAM location 50 hex you would execute the instruction:

```
MOV 50h,#01h
```

Similarly, if you want to write the value "1" to the 8051's serial port you would write this value to the SBUF SFR, which has an SFR address of 99 Hex. Thus, to write the value "1" to the serial port you would execute the instruction:

```
MOV 99h,#01h
```

As you can see, it appears that the SFR is part of Internal Memory. This is not the case. When using this method of memory access (its called direct address), any instruction that has an address of 00h through 7Fh refers to an Internal RAM memory address; any instruction with an address of 80h through FFh refers to an SFR control register.

SFRs

The 8051 is a flexible microcontroller with a relatively large number of modes of operations. Your program may inspect and/or change the operating mode of the 8051 by manipulating the values of the 8051's Special Function Registers (SFRs).

SFRs are accessed as if they were normal Internal RAM. The only difference is that Internal RAM is from address 00h through 7Fh whereas SFR registers exist in the address range of 80h through FFh.

Each SFR has an address (80h through FFh) and a name. The following chart provides a graphical presentation of the 8051's SFRs, their names, and their address.
As you can see, although the address range of 80h through FFh offer 128 possible addresses, there are only 21 SFRs in a standard 8051. All other addresses in the SFR range (80h through FFh) are considered invalid. Writing to or reading from these registers may produce undefined values or behavior.

SFR Types

As mentioned in the chart itself, the SFRs that have a blue background are SFRs related to the I/O ports. The 8051 has four I/O ports of 8 bits, for a total of 32 I/O lines. Whether a given I/O line is high or low and the value read from the line are controlled by the SFRs in green.

The SFRs with yellow backgrounds are SFRs which in some way control the operation or the configuration of some aspect of the 8051. For example, TCON controls the timers, SCON controls the serial port.

The remaining SFRs, with green backgrounds, are "other SFRs." These SFRs can be thought of as auxillary SFRs in the sense that they don't directly configure the 8051 but obviously the 8051 cannot operate without them. For example, once the serial port has been configured using SCON, the program may read or write to the serial port using the SBUF register.

SFR Descriptions

This section will endeavor to quickly overview each of the standard SFRs found in the above SFR chart map. It is not the intention of this section to fully explain the functionality of each SFR--this information will be covered in separate chapters of the tutorial. This section is to just give you a general idea of what each SFR does.
P0 (Port 0, Address 80h, Bit-Addressable):

This is input/output port 0. Each bit of this SFR corresponds to one of the pins on the microcontroller. For example, bit 0 of port 0 is pin P0.0, bit 7 is pin P0.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to a low level.

SP (Stack Pointer, Address 81h):

This is the stack pointer of the microcontroller. This SFR indicates where the next value to be taken from the stack will be read from in Internal RAM. If you push a value onto the stack, the value will be written to the address of SP + 1. That is to say, if SP holds the value 07h, a PUSH instruction will push the value onto the stack at address 08h. This SFR is modified by all instructions which modify the stack, such as PUSH, POP, LCALL, RET, RETI, and whenever interrupts are provoked by the microcontroller.

DPL/DPH (Data Pointer Low/High, Addresses 82h/83h):

The SFRs DPL and DPH work together to represent a 16-bit value called the Data Pointer. The data pointer is used in operations regarding external RAM and some instructions involving code memory. Since it is an unsigned two-byte integer value, it can represent values from 0000h to FFFFh (0 through 65,535 decimal).

PCON (Power Control, Addresses 87h):

The Power Control SFR is used to control the 8051’s power control modes. Certain operation modes of the 8051 allow the 8051 to go into a type of “sleep” mode which requires much less power. These modes of operation are controlled through PCON. Additionally, one of the bits in PCON is used to double the effective baud rate of the 8051’s serial port.

TCON (Timer Control, Addresses 88h, Bit-Addressable):

The Timer Control SFR is used to configure and modify the way in which the 8051’s two timers operate. This SFR controls whether each of the two timers is running or stopped and contains a flag to indicate that each timer has overflowed. Additionally, some non-timer related bits are located in the TCON SFR. These bits are used to configure the way in which the external interrupts are activated and also contain the external interrupt flags which are set when an external interrupt has occurred.

TMOD (Timer Mode, Addresses 89h):

The Timer Mode SFR is used to configure the mode of operation of each of the two timers. Using this SFR your program may configure each timer to be a 16-bit timer, an 8-bit autoreload timer, a 13-bit timer, or two separate timers. Additionally, you may configure the timers to only count when an external pin is activated or to count "events" that are indicated on an external pin.

TL0/TH0 (Timer 0 Low/High, Addresses 8Ah/8Ch):

These two SFRs, taken together, represent timer 0. Their exact behavior depends on how the timer is configured in the TMOD SFR; however, these timers always count up. What is configurable is how and when they increment in value.

TL1/TH1 (Timer 1 Low/High, Addresses 8Bh/8Dh):
These two SFRs, taken together, represent timer 1. Their exact behavior depends on how the timer is configured in the TMOD SFR; however, these timers always count up. What is configurable is how and when they increment in value.

P1 (Port 1, Address 90h, Bit-Addressable):

This is input/output port 1. Each bit of this SFR corresponds to one of the pins on the microcontroller. For example, bit 0 of port 1 is pin P1.0, bit 7 is pin P1.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to a low level.

SCON (Serial Control, Addresses 98h, Bit-Addressable):

The Serial Control SFR is used to configure the behavior of the 8051's on-board serial port. This SFR controls the baud rate of the serial port, whether the serial port is activated to receive data, and also contains flags that are set when a byte is successfully sent or received.

SBUF (Serial Control, Addresses 99h):

The Serial Buffer SFR is used to send and receive data via the on-board serial port. Any value written to SBUF will be sent out the serial port's TXD pin. Likewise, any value which the 8051 receives via the serial port's RXD pin will be delivered to the user program via SBUF. In other words, SBUF serves as the output port when written to and as an input port when read from.

P2 (Port 2, Address A0h, Bit-Addressable):

This is input/output port 2. Each bit of this SFR corresponds to one of the pins on the microcontroller. For example, bit 0 of port 2 is pin P2.0, bit 7 is pin P2.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to a low level.

IE (Interrupt Enable, Addresses A8h):

The Interrupt Enable SFR is used to enable and disable specific interrupts. The low 7 bits of the SFR are used to enable/disable the specific interrupts, where as the highest bit is used to enable or disable ALL interrupts. Thus, if the high bit of IE is 0 all interrupts are disabled regardless of whether an individual interrupt is enabled by setting a lower bit.

P3 (Port 3, Address B0h, Bit-Addressable):

This is input/output port 3. Each bit of this SFR corresponds to one of the pins on the microcontroller. For example, bit 0 of port 3 is pin P3.0, bit 7 is pin P3.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to a low level.

IP (Interrupt Priority, Addresses B8h, Bit-Addressable):

The Interrupt Priority SFR is used to specify the relative priority of each interrupt. On the 8051, an interrupt may either be of low (0) priority or high (1) priority. An interrupt may only interrupt interrupts of lower priority. For example, if we configure the 8051 so that all interrupts are of low priority except the serial interrupt, the serial interrupt will always be able to interrupt the system, even if another interrupt is currently
executing. However, if a serial interrupt is executing no other interrupt will be able to interrupt the serial interrupt routine since the serial interrupt routine has the highest priority.

PSW (Program Status Word, Addresses D0h, Bit-Addressable):

The Program Status Word is used to store a number of important bits that are set and cleared by 8051 instructions. The PSW SFR contains the carry flag, the auxiliary carry flag, the overflow flag, and the parity flag. Additionally, the PSW register contains the register bank select flags which are used to select which of the "R" register banks are currently selected.

ACC (Accumulator, Addresses E0h, Bit-Addressable):

The Accumulator is one of the most-used SFRs on the 8051 since it is involved in so many instructions. The Accumulator resides as an SFR at E0h, which means the instruction MOV A,#20h is really the same as MOV E0h,#20h. However, it is a good idea to use the first method since it only requires two bytes whereas the second option requires three bytes.

B (B Register, Addresses F0h, Bit-Addressable):

The "B" register is used in two instructions: the multiply and divide operations. The B register is also commonly used by programmers as an auxiliary register to temporarily store values.
PART-A

1. What is the jump range?

There are three forms of jump. They are

LJMP (Long jump)-address 16
AJMP (Absolute Jump)-address 11
SJMP (Short Jump)-relative address

2. Explain the addressing modes of 8051.

(i) register addressing
(ii) Direct byte addressing
(iii) Register indirect addressing
(iv) Immediate addressing
(v) register specific addressing
(vi) Index addressing

3. Identify the addressing mode used by each of the following instruction.

i) MOV A, R4
register addressing

ii) MOVC A, @A+DPTR
Index addressing

iii) SWAP A

register specific addressing

iv) MOV A, #30H

Immediate addressing

4. Explain PUSH and POP instructions in 8051.

PUSH-The stack pointer is incremented by one. The contents of the indicated variable is then copied into the internal RAM location addressed by the stack pointer.

POP-Reverse of PUSH operation

5. What are the instructions used to access external RAM.

MOVC A,@A+DPTR

MOVC A,@A+PC

6. What is key bounce? How it is achieved?

i) Mechanical switches are used as keys in most of the keyboards.

ii) When a key is pressed the contacts bounce back and forth and settle down

iii) only after a small time delay (about 20 ms). Even through a key is actuated once, it will appear to have been actuated several times. This problem is called key bouncing

7. Explain DAA instruction of 8051.

Decimal adjust accumulator for addition bytes

8. Explain rotate instructions of 8051.

RL A, RLC A, RR A, RRC A

9. Give the PSW setting for masking register bank 2 as default register bank in 8051 Microcontroller?
Selecting one of the 4 banks is done by setting or clearing the 2 bank select bits RB0 and RB1 in the PSW register. Registers are called R0 to R7 by default.

10. **How can you perform multiplication in 8051?**

MUL AB multiplies the unsigned eight bit integers in the Accumulator and REG B. The low order byte of the 16 Bit product is left in the accumulator, and the high order byte in B.

11. **If the product is 12. How can you perform addition in 8051?**

```
MOV A, #30H
ADD A, #50H
```

12. **Name any four bit manipulation instructions in 8051?**

```
ANL A, ORL A, XRL A, CLR A
```

13. **Write a program to subtract the contents of R1 of Bank 0 from the contents of R0 of Bank 2 using 8051?**

```
MOV PSW, #10
MOV A, R0
MOV PSW, #00
SUBB A, R1
```

14. **Write a program to subtract 2 8-bit numbers & exchange the digits using 8051?**

```
MOV A, #9F
MOV R0, #40
SUBB A, R0
SWAP A
```

15. **Write a program to swap two numbers using 8051?**

```
MOV A, #data
SWAP A
```

16. **Write a program to add 2 8-bit numbers using 8051?**
MOV A,#30H
ADD A,#50H

17. Write a program to find the 2’s complement using 8051?

    MOV A,R0
    CPL A
    INC A

18. Write program to load accumulator ,DPH,&DPL using 8051?

    MOV A,#30
    MOV DPH,A
    MOV DPL,A

19. Write about the jump statement?

    There are three forms of jump. They are
    LJMP(Long jump)-address 16
    AJMP(Absolute Jump)-address 11
    SJMP(Short Jump)-relative address

20. Write about CALL statement in 8051?

    There are two subroutine CALL instructions. They are
    LCALL(Long CALL)
    ACALL(Absolute CALL)

    Each increments the PC to the 1st byte of the instruction & pushes them in to the stack

21. List the addressing modes of 8051?

    Direct addressing
    Register addressing
Register indirect addressing.

Implicit addressing

Immediate addressing

Index addressing

Bit addressing

22. Write a program to mask the 0th &7th bit using 8051?

```assembly
MOV A,#data
ANL A,#81
MOV DPTR,#4500
MOVX @DPTR,A
LOOP SJMP LOOP
```

23. Write a program to perform multiplication of 2 nos using 8051?

```assembly
MOV A,#data 1
MOV B,#data 2
MUL AB
MOV DPTR,#5000
MOV @DPTR,A(lower value)
INC DPTR
MOV A,B
MOVX @DPTR,A
```

24. Explain the interrupts of 8051 microcontroller?

INT0, TF0, INT1, TF1, RI& TI
1. List out the Addressing Modes of 8051 Microcontroller.

8051 has four addressing modes.

1. Immediate Addressing:
   Data is immediately available in the instruction.
   For example -
   
   ADD A, #77; Adds 77 (decimal) to A and stores in A
   
   ADD A, #4DH; Adds 4D (hexadecimal) to A and stores in A
   
   MOV DPTR, #1000H; Moves 1000 (hexadecimal) to data pointer

2. Bank Addressing or Register Addressing:
   This way of addressing accesses the bytes in the current register bank. Data is available in the register specified in the instruction. The register bank is decided by 2 bits of Processor Status Word (PSW).
   For example -
   
   ADD A, R0; Adds content of R0 to A and stores in A

3. Direct Addressing:
   The address of the data is available in the instruction.
   For example -
   
   MOV A, 088H; Moves content of SFR TCON (address 088H) to A

4. Register Indirect Addressing:
   The address of data is available in the R0 or R1 registers as specified in the instruction.
   For example - MOV A, @R0 moves content of address pointed by R0 to A

External Data Addressing:
   Pointer used for external data addressing can be either R0/R1 (256 byte access) or DPTR (64kbyte access).
   For example -
   
   MOVX A, @R0; Moves content of 8-bit address pointed by R0 to A
MOVX A, @DPTR; Moves content of 16-bit address pointed by DPTR to A

External Code Addressing:
Sometimes we may want to store non-volatile data into the ROM e.g. look-up tables.

Such data may require reading the code memory.

This may be done as follows -

MOVX A, @A+DPTR; Moves content of address pointed by A+DPTR to A

MOVX A, @A+PC; Moves content of address pointed by A+PC to A

2. Explain the Serial Communication Interface in 8051.

Serial Interface
The serial port of 8051 is full duplex, i.e., it can transmit and receive simultaneously.

The register SBUF is used to hold the data. The special function register SBUF is physically two registers. One is, write-only and is used to hold data to be transmitted out of the 8051 via TXD. The other is, read-only and holds the received data from external sources via RXD. Both mutually exclusive registers have the same address 099H.

**Serial Port Control Register (SCON)**

Register SCON controls serial data communication.

Address: 098H (Bit address)

Mode select bits

SM2: multi processor communication bit
REN: Receive enable bit
TB8: Transmitted bit 8 (Normally we have 0-7 bits transmitted/received)
RB8: Received bit 8
TI: Transmit interrupt flag
RI: Receive interrupt flag
Power Mode control Register

Register PCON controls processor power down, sleep modes and serial data baud rate. Only one bit of PCON is used with respect to serial communication. The seventh bit (b7)(SMOD) is used to generate the baud rate of serial communication.

Address: 87H

SMOD: Serial baud rate modify bit
GF1: General purpose user flag bit 1
GF0: General purpose user flag bit 0
PD: Power down bit
IDL: Idle mode bit

Data Transmission

Transmission of serial data begins at any time when data is written to SBUF. Pin P3.1 (Alternate function bit TXD) is used to transmit data to the serial data network. TI is set to 1 when data has been transmitted. This signifies that SBUF is empty so that another byte can be sent.

Data Reception

Reception of serial data begins if the receive enable bit is set to 1 for all modes. Pin P3.0 (Alternate function bit RXD) is used to receive data from the serial data network. Receive interrupt flag, RI, is set after the data has been received in all modes. The data gets stored in SBUF register from where it can be read.

Serial Data Transmission Modes:

Mode-0: In this mode, the serial port works like a shift register and the data transmission works synchronously with a clock frequency of fosc /12. Serial data is received and transmitted through RXD. 8 bits are transmitted/received aty a time. Pin TXD outputs the shift clock pulses of frequency fosc /12, which is connected to the external circuitry for synchronization. The shift frequency or baud rate is always 1/12 of the oscillator frequency.
Fig 11.1 Data transmission/reception in Mode-0

Mode-1 (standard UART mode):

In mode-1, the serial port functions as a standard Universal Asynchronous Receiver Transmitter (UART) mode. 10 bits are transmitted through TXD or received through RXD. The 10 bits consist of one start bit (which is usually '0'), 8 data bits (LSB is sent first/received first), and a stop bit (which is usually '1'). Once received, the stop bit goes into RB8 in the special function register SCON. The baud rate is variable.

The following figure shows the way the bits are transmitted/received.

Fig 11.2 Data transmission format in UART mode
Bit time = 1/f baud

In receiving mode, data bits are shifted into the receiver at the programmed baud rate. The data word (8-bits) will be loaded to SBUF if the following conditions are true.

1. RI must be zero. (i.e., the previously received byte has been cleared from SBUF)
2. Mode bit SM2 = 0 or stop bit = 1.

After the data is received and the data byte has been loaded into SBUF, RI becomes one.

Mode-1 baud rate generation:

Timer-1 is used to generate baud rate for mode-1 serial communication by using overflow flag of the timer to determine the baud frequency. Timer-1 is used in timer mode-2 as an auto-reload 8-bit timer. The data rate is generated by timer-1 using the following formula.

\[
\text{f}_{\text{baud}} = \frac{2^{\text{SMOD} \times \text{fosc}}}{32 \times 12 \times [256 - (\text{TH1})]}
\]

Where,
SMOD is the 7th bit of PCON register
    fosc is the crystal oscillator frequency of the microcontroller

It can be noted that \( f_{osc} / (12 \times [256 - (TH1)]) \) is the timer overflow frequency in timer mode-2, which is the auto-reload mode.

If timer-1 is not run in mode-2, then the baud rate is,

\[
f_{\text{baud}} = \frac{2 \times \text{SMOD}}{32} \times \text{X (timer-1 overflow frequency)}
\]

Timer-1 can be run using the internal clock, \( f_{osc}/12 \) (timer mode) or from any external source via pin T1 (P3.5) (Counter mode).

Example: If standard baud rate is desired, then 11.0592 MHz crystal could be selected. To get a standard 9600 baud rate, the setting of TH1 is calculated as follows.

Assuming SMOD to be '0'

\[
9600 = \frac{2^0}{32} \times \frac{11.0592 \times 10^6}{12 \times (256-TH1)}
\]

Or,

\[
256-\text{TH1} = \frac{1}{32} \times \frac{11.0592 \times 10^6}{12 \times 9600} = 3
\]

Or,

\[
\text{TH1} = 256 \times 3 = 253 = \text{FDH}
\]

In mode-1, if SM2 is set to 1, no receive interrupt (RI) is generated unless a valid stop bit is received.

**Serial Data Mode-2 - Multiprocessor Mode:**

In this mode 11 bits are transmitted through TXD or received through RXD. The various bits are as follows: a start bit (usually '0'), 8 data bits (LSB first), a programmable 9\(^{th}\) (TB8 or RB8) bit and a stop bit (usually '1').

While transmitting, the 9\(^{th}\) data bit (TB8 in SCON) can be assigned the value '0' or '1'. For example, if the information of parity is to be transmitted, the parity bit (P) in PSW could be moved into TB8. On
reception of the data, the 9th bit goes into RB8 in 'SCON', while the stop bit is ignored. The baud rate is programmable to either 1/32 or 1/64 of the oscillator frequency.

\[ f_{\text{baud}} = \left(2^{\text{SMOD}} / 64\right) f_{\text{osc}}. \]

**Mode-3 - Multi processor mode with variable baud rate:**

In this mode 11 bits are transmitted through TXD or received through RXD. The various bits are: a start bit (usually '0'), 8 data bits (LSB first), a programmable 9th bit and a stop bit (usually '1').

Mode-3 is same as mode-2, except the fact that the baud rate in mode-3 is variable (i.e., just as in mode-1). \( f_{\text{baud}} = \left(2^{\text{SMOD}} / 32\right) \times \left( f_{\text{osc}} / \left(12 \times (256-\text{TH1})\right) \right). \)

This baudrate holds when Timer-1 is programmed in Mode-2.

3. **Explain 8051 Instruction set in detail.**

8051 has about 111 instructions. These can be grouped into the following categories:

1. Arithmetic Instructions
2. Logical Instructions
3. Data Transfer instructions
4. Boolean Variable Instructions
5. Program Branching Instructions

The following nomenclatures for register, data, address and variables are used while write instructions.

A: Accumulator

B: "B" register

C: Carry bit

Rn: Register R0 - R7 of the currently selected register bank

Direct: 8-bit internal direct address for data. The data could be in lower 128bytes of RAM (00 - 7FH) or it could be in the special function register (80 - FFH).

@Ri: 8-bit external or internal RAM address available in register R0 or R1. This is used for indirect addressing mode.

#data8: Immediate 8-bit data available in the instruction.
#data16: Immediate 16-bit data available in the instruction.

Addr11: 11-bit destination address for short absolute jump. Used by instructions AJMP & ACALL. Jump range is 2 kbyte (one page).

Addr16: 16-bit destination address for long call or long jump.

Rel: 2’s complement 8-bit offset (one - byte) used for short jump (SJMP) and all conditional jumps.

bit: Directly addressed bit in internal RAM or SFR

**Arithmetic Instructions**

<table>
<thead>
<tr>
<th>Mnemonics</th>
<th>Description</th>
<th>Bytes</th>
<th>Instruction Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD A, Rn</td>
<td>A ← A + Rn</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ADD A, direct</td>
<td>A ← A + (direct)</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ADD A, @Ri</td>
<td>A ← A + @Ri</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ADD A, #data</td>
<td>A ← A + data</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ADDC A, Rn</td>
<td>A ← A + Rn + C</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ADDC A, direct</td>
<td>A ← A + (direct) + C</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ADDC A, @Ri</td>
<td>A ← A + @Ri + C</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ADDC A, #data</td>
<td>A ← A + data + C</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>DA A</td>
<td>Decimal adjust accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DIV AB</td>
<td>Divide A by B</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>A ← quotient</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>B ← remainder</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEC A</td>
<td>A ← A - 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DEC Rn</td>
<td>Rn ← Rn - 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DEC direct</td>
<td>(direct) ← (direct) - 1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>DEC @Ri</td>
<td>@Ri ← @Ri - 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>INC A</td>
<td>A ← A + 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>INC Rn</td>
<td>Rn ← Rn + 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>INC direct</td>
<td>(direct) ← (direct) + 1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>INC @Ri</td>
<td>@Ri ← @Ri + 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>INC DPTR</td>
<td>DPTR ← DPTR + 1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>MUL AB</td>
<td>Multiply A by B</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>A ← low byte (A*B)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>B ← high byte (A* B)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mnemonics</td>
<td>Description</td>
<td>Bytes</td>
<td>Instruction Cycles</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------</td>
<td>-------</td>
<td>--------------------</td>
</tr>
<tr>
<td>SUBB A, Rn</td>
<td>A ← A - Rn - C</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SUBB A, direct</td>
<td>A ← A - (direct) - C</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>SUBB A, @Ri</td>
<td>A ← A - @Ri - C</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SUBB A, #data</td>
<td>A ← A - data - C</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

**Logical Instructions**

<table>
<thead>
<tr>
<th>Mnemonics</th>
<th>Description</th>
<th>Bytes</th>
<th>Instruction Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANL A, Rn</td>
<td>A ← A AND Rn</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ANL A, direct</td>
<td>A ← A AND (direct)</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ANL A, @Ri</td>
<td>A ← A AND @Ri</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ANL A, #data</td>
<td>A ← A AND data</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ANL direct, A</td>
<td>(direct) ← (direct) AND A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ANL direct, #data</td>
<td>(direct) ← (direct) AND data</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>CLR A</td>
<td>A ← 00H</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CPL A</td>
<td>A ← A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ORL A, Rn</td>
<td>A ← A OR Rn</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ORL A, direct</td>
<td>A ← A OR (direct)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ORL A, @Ri</td>
<td>A ← A OR @Ri</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ORL A, #data</td>
<td>A ← A OR data</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ORL direct, A</td>
<td>(direct) ← (direct) OR A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ORL direct, #data</td>
<td>(direct) ← (direct) OR data</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>RL A</td>
<td>Rotate accumulator left</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RLC A</td>
<td>Rotate accumulator left through carry</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RR A</td>
<td>Rotate accumulator right</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RRC A</td>
<td>Rotate accumulator right through carry</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SWAP A</td>
<td>Swap nibbles within Acumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XRL A, Rn</td>
<td>A ← A EXOR Rn</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XRL A, direct</td>
<td>A ← A EXOR (direct)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XRL A, @Ri</td>
<td>A ← A EXOR @Ri</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>XRL A, #data</td>
<td>A ← A EXOR data</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XRL direct, A</td>
<td>(direct) ← (direct) EXOR A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>XRL direct, #data</td>
<td>(direct) ← (direct) EXOR data</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Mnemonics</td>
<td>Description</td>
<td>Bytes</td>
<td>Instruction Cycles</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------------------------------------------</td>
<td>-------</td>
<td>-------------------</td>
</tr>
<tr>
<td>MOV A, Rn</td>
<td>A ← Rn</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MOV A, direct</td>
<td>A ← (direct)</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV A, @Ri</td>
<td>A ← @Ri</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MOV A, #data</td>
<td>A ← data</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV Rn, A</td>
<td>Rn ← A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MOV Rn, direct</td>
<td>Rn ← (direct)</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>MOV Rn, #data</td>
<td>Rn ← data</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV direct, A</td>
<td>(direct) ← A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV direct, Rn</td>
<td>(direct) ← Rn</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>MOV direct1, direct2</td>
<td>(direct1) ← (direct2)</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>MOV direct, @Ri</td>
<td>(direct) ← @Ri</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>MOV direct, #data</td>
<td>(direct) ← #data</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>MOV @Ri, A</td>
<td>@Ri ← A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MOV @Ri, direct</td>
<td>@Ri ← (direct)</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>MOV @Ri, #data</td>
<td>@Ri ← data</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV DPTR, #data16</td>
<td>DPTR ← data16</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>MOVC A, @A+DPTR</td>
<td>A ← Code byte pointed by A + DPTR</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>MOVC A, @A+PC</td>
<td>A ← Code byte pointed by A + PC</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>MOVC A, @Ri</td>
<td>A ← Code byte pointed by Ri 8-bit address)</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>MOVX A, @DPTR</td>
<td>A ← External data pointed by DPTR</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>MOVX @Ri, A</td>
<td>@Ri ← A (External data - 8bit address)</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>MOVX @DPTR, A</td>
<td>@DPTR ← A (External data - 16bit address)</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>PUSH direct</td>
<td>(SP) ← (direct)</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>POP direct</td>
<td>(direct) ← (SP)</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>XCH Rn</td>
<td>Exchange A with Rn</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XCH direct</td>
<td>Exchange A with direct byte</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>XCH @Ri</td>
<td>Exchange A with indirect RAM</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XCHD A, @Ri</td>
<td>Exchange least significant nibble of A with</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
that of indirect RAM

### Boolean Variable Instructions

<table>
<thead>
<tr>
<th>Mnemonics</th>
<th>Description</th>
<th>Bytes</th>
<th>Instruction Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR C</td>
<td>C-bit ←0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CLR bit</td>
<td>bit ←0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>SET C</td>
<td>C ←1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SET bit</td>
<td>bit ←1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>CPL C</td>
<td>C ←¬bit</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CPL bit</td>
<td>bit ←¬bit</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ANL C, /bit</td>
<td>C ← C . bit</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ANL C, bit</td>
<td>C ← C . bit</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ORL C, /bit</td>
<td>C ← C + bit</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ORL C, bit</td>
<td>C ← C + bit</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV C, bit</td>
<td>C ← bit</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV bit, C</td>
<td>bit ← C</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

### Program Branching Instructions

<table>
<thead>
<tr>
<th>Mnemonics</th>
<th>Description</th>
<th>Bytes</th>
<th>Instruction Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACALL addr11</td>
<td>PC + 2 →(SP) ; addr 11 → PC</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>AJMP addr11</td>
<td>Addr11 → PC</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>CJNE A, direct, rel</td>
<td>Compare with A, jump (PC + rel) if not equal</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>CJNE A, #data, rel</td>
<td>Compare with A, jump (PC + rel) if not equal</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>CJNE Rn, #data, rel</td>
<td>Compare with Rn, jump (PC + rel) if not equal</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>CJNE @Ri, #data, rel</td>
<td>Compare with @Ri A, jump (PC + rel) if not equal</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>DJNZ Rn, rel</td>
<td>Decrement Rn, jump if not zero</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>DJNZ direct, rel</td>
<td>Decrement (direct), jump if not zero</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>JC rel</td>
<td>Jump (PC + rel) if C bit = 1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>JNC rel</td>
<td>Jump (PC + rel) if C bit = 0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>JB bit, rel</td>
<td>Jump (PC + rel) if bit = 1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>JNB bit, rel</td>
<td>Jump (PC + rel) if bit = 0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>JBC bit, rel</td>
<td>Jump (PC + rel) if bit = 1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Instruction</td>
<td>Description</td>
<td>Cycles</td>
<td>Bytes</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>JMP @A+DPTR</td>
<td>A+DPTR → PC</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>JZ rel</td>
<td>If A=0, jump to PC+rel</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>JNZ rel</td>
<td>If A ≠ 0, jump to PC+rel</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>LCALL addr16</td>
<td>PC + 3 → (SP), addr16 → PC</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>LJMP addr16</td>
<td>Addr16 → PC</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>NOP</td>
<td>No operation</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RET</td>
<td>(SP) → PC</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>RETI</td>
<td>(SP) → PC, Enable Interrupt</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>SJMP rel</td>
<td>PC + 2 + rel → PC</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>JMP @A+DPTR</td>
<td>A+DPTR → PC</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>JZ rel</td>
<td>If A = 0, jump PC+rel</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>JNZ rel</td>
<td>If A ≠ 0, jump PC+rel</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>NOP</td>
<td>No operation</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>