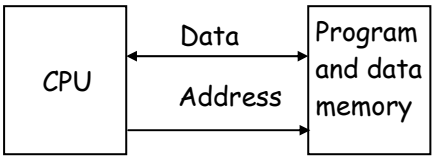
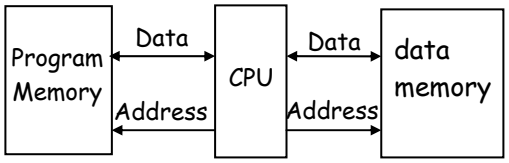
T.Y. Diploma : Sem. V
[EJ/EN/ET/EX/DE/IS/IC/IE/EV/MU]
Microcontroller

Time : 3 Hrs.] Prelim Question Paper Solution **[Marks : 100**

Q.1 (a) Attempt any THREE of the following: **[12]**

Q.1 (a) (i) State the difference between Harvard and Von Neumann architecture with suitable diagram. **[4]**

(A)

	Harvard architecture	**Von Neumann architecture**
(i)		
(ii)	The Harvard architecture uses physically separate memories for their instructions and data.	The Van Neumann architecture uses single memory for their instructions and data.
(iii)	Requires separate and dedicated buses for memories for instructions and data.	Requires single bus for instructions and data
(iv)	Its design is complicated.	Its design is simpler
(v)	Instructions and data can be fetched simultaneously as there is separate buses for instruction and data which increasing operation bandwidth.	Instructions and data have to be fetched in sequential order limiting the operation bandwidth.
(vi)	Vectors and pointers, variables program segments and memory blocks for data and stacks have different addresses in the program.	Program segments and memory blocks for data and stacks have separate sets of addressed.

Q.1 (a) (ii) List features of 8051. **[4]**

(A) Features of 8051 micro controller are as follows:

(1) 8- bit data bus and 8- bit ALU.
(2) 16- bit address bus – can access maximum 64KB of RAM and ROM.
(3) On- chip RAM -128 bytes (Data Memory)
(4) On- chip ROM – 4 KB (Program Memory)
(5) Four 8-bit bi- directional input/output ports Four 8-bit bi- directional input/ output ports.
(6) Programmable serial ports i.e. One UART (serial port)
(7) Two 16- bit timers- Timer 0& Timer 1
(8) Works on crystal frequency of 11.0592 MHz.
(9) Has power saving and idle mode in microcontroller when no operation is performed.
(10) Six interrupts are available: Reset, Two interrupts Timers i.e. Timer 0 and Timer 1, two external hardware interrupts- INT0 and INT1, Serial communication interrupt for both receive and transmit.

Q.1 (a) (iii) Draw and explain TCON reg. **[4]**

(A) TCON: TIMER/COUNTER CONTROL REGISTER. BIT ADDRESSABLE.

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 (TCON.7): Timer 1 overflow flag. Set by hardware when the Timer/Counter 1 overflows. Cleared by hardware as processor vectors to the interrupt service routine.

TR1 (TCON.6): Timer 1 run control bit. Set/cleared by software to turn Timer/Counter 1ON/OFF.

TF0 (TCON.5): Timer 0 overflow flag. Set by hardware when the Timer/Counter 0 overflows. Cleared by hardware as processor vectors to the service routine.

TR0 (TCON.4): Timer 0 run control bit. Set/cleared by software to turn Timer/Counter 0 ON/OFF.

IE1 (TCON.3): External Interrupt 1 edge flag. Set by hardware when External Interrupt edge is detected. Cleared by hardware when interrupt is processed.

IT1 (TCON.2): Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.

IE0 (TCON.1): External Interrupt 0 edge flag. Set by hardware when External Interrupt edge detected. Cleared by hardware when interrupt is processed.

IT0 (TCON.0): Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.

Q.1 (a) (iv) Explain following instructions : [4]

(1) XCH A, @ Ri
(2) CJNE A, direct rel.
(3) SJMP addr
(4) LJMP addr.

(A) (1) XCH A, @ Ri

Exchange the contents of the accumulator with the contents of the memory location pointed by the 8 bit register in the instruction.

Example: XCH A, @R1

Exchange the contents of the accumulator with the contents of the memory location pointed by the R1 register.

(2) CJNE A, direct rel.

Compare the contents of the accumulator with the 8 bit data directly mentioned in the instruction and if they are not equal then jump to the relative address mentioned in the instruction.

Example: CJNE A, 04H, UP

Compare the contents of the accumulator with the 04H mentioned in the instruction and if they are not equal then jump to the line of instruction where UP label is mentioned.

(3) SJMP addr

PC_{new} PC_{old} +addr

The program control jumps max 128 bytes backwards or max 127 bytes forward with respect to the current PC. Next instruction is executed from new PC.

(4) LJMP addr.

The program counter is loaded with the 16-bit long address given in 2nd and 3rd byte of the instruction. Jump can take place anywhere in the 64KB memory space.

PC ⇦ addr
Example: LJMP 3245H
PC ⇦ 3245

The next instruction executes from the memory location 3245H.

Q.1 (a) (v) What is BUS? Explain different types of buses used in 8051. [4]

(A) A Bus is a set of physical connection used for communication between CPU and peripherals. There are three types of buses Address Bus, Data Bus and Control Bus

(1) Address Bus : The address bus is unidirectional over which the microcontroller sends an address code to the memory or input/output. The size of the address bus is specified by the number of bits it can handle. The more bits there are in the address bus, the more memory locations a microcontroller can access. A 16-bit address bus is capable of addressing (64k) addresses.

(2) Data Bus : The data bus is bi—directional on which data or instruction codes are transferred into the microcontroller or on which the result of on operation or computation is sent out from the microcontroller to the memory or input/output. Depending on the particular microcontroller, the data bus can handle 8-bit or 16-bit data.

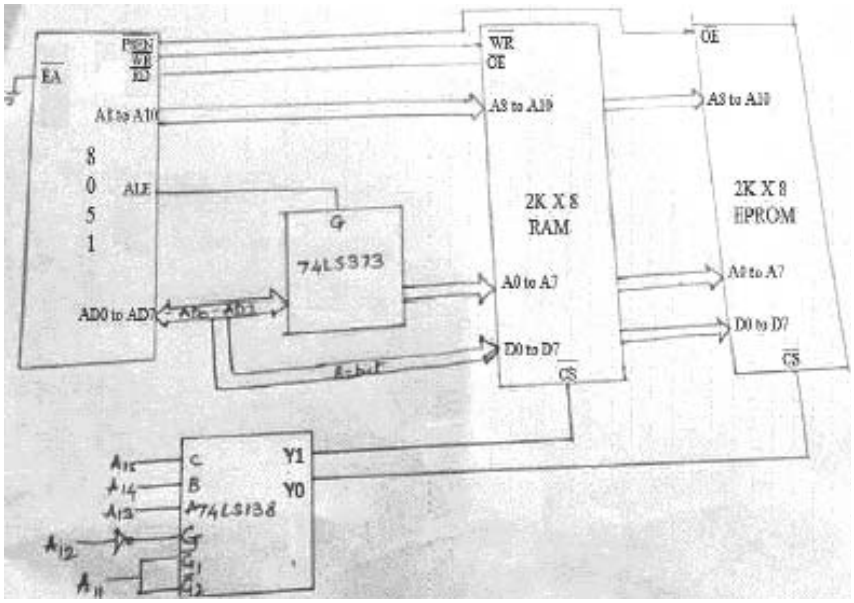
(3) Control Bus : The control bus is used by the microcontroller to send out or receive timing and control signals like read and write in order to co- ordinate and regulate its operation and to communicate with other devices i.e. memory or input/output.

Q.1 (b) Attempt any ONE of the following : [6]

Q.1 (b) (i) Draw the interfacing of 2KB of EPROM & 2kB of RAM to 8051 along with its mem map. [6]

(A)

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	ADDR
Start addr of EPROM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
End addr of EPROM	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	07FFH
Start addr of ROM	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000H
End addr of ROM	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	27FFH



Q.1 (b) (ii) Write an assembly language program to exchange ten bytes of data from source location 40H to destination location 60H, for 8051 microcontrolller. [6]

(A)

ORG 0000H	; Program from 0000H
MOV R3, #10	; Initialize Byte counter
MOV R0, #40H	; Initialize memory pointer for source array
MOV R1, #60H	; Initialize memory pointer for destination array ; Therefore R0 ---> Source pointer R1 ---> destination pointer
UP: MOV A, @R0	; Read data from SOURCE array
MOV B, @R1	; Read data from DESTINATION array
MOV @R1, A	; Write data of source array into destination array
MOV @R0, B	; Write data of destination array into source array
INC R0	; Increment source memory pointer by 1
INC R1	; Increment destination memory pointer by 1
DJNZ R3, UP	; Decrement byte counter by 1 ; Is it zero? No, jump to UP
HERE: SJMP HERE	
END	;Stop

Q.2 Attempt any FOUR of the following : [16]

Q.2 (a) Describe the function of following pins of 8051 [4]

(i) $\overline{\text{PSEN}}$ **(ii) ALE** **(iii)** $\overline{\text{EA}}$ | VPP **(iv) RST**

(A) (i) $\overline{\text{PSEN}}$

- PSEN stands for "program store enable." The read strobe for external Program Memory is the signal PSEN (Program Store Enable). In an 8031-based system in which an external ROM holds the program code, this pin is connected to the OE pin of the ROM.
- In other words, to access external ROM containing program code, the 8031/51 uses the PSEN signal. This read strobe is used for all external program fetches. PSEN is not activated for internal fetches.

(ii) ALE

- ALE stands for address latch enable. It is an output pin and is active high for latching the low byte of address during accesses to external memory.
- The ALE pin is used for demultiplexing the address and data by connecting to the G pin of the 74LS373 chip.

(iii) $\overline{\text{EA}}$ | VPP

- EA which stands for external access is pin number 31 in the DIP packages. It is an input pin and must be connected to either Vcc or GND. In other words, it cannot be left unconnected.
- The lowest 4K (or SK or 16K) bytes of Program Memory can be either in the on-chip ROM or in an external ROM. This selection is made by strapping the EA (External Access) pin to either VCC or Vss.
- In the 4K byte ROM devices, if the pin is strapped to Vcc, then program fetches to addresses 0000H through OFFFH are directed to the internal ROM. Program fetches to addresses 1000H through FFFFH are directed to external ROM.
- If the pin is strapped to Vss, then all program fetches are directed to external ROM. The ROMless parts must have this pin externally strapped to VSS to enable them to execute properly.

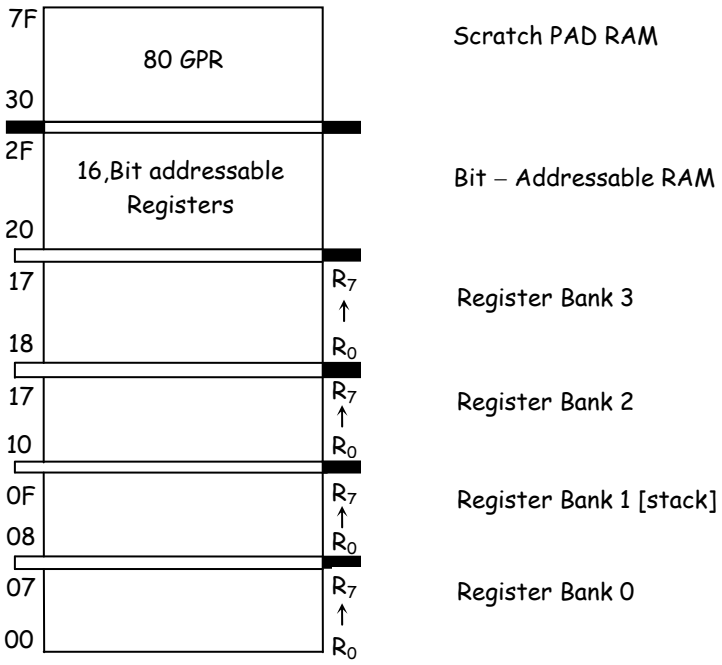
(iv) RST

- Pin 9 is the RESET pin. It is an input and is active high (normally low). Upon applying a high pulse to this pin, the microcontroller will reset and terminate all activities.
- This is often referred to as a power-on reset. Activating a power-on reset will cause all values in the registers to be lost. It will set program counter to all 0s.

- In order for the RESET input to be effective, it must have a minimum duration of two machine cycles. In other words, the high pulse must be high for a minimum of two machine cycles before it is allowed to go low.

Q.2 (b) Draw internal RAM organization of 8051 and explain. [4]

(A) Internal RAM Memory



RAM Memory :

There are 128 bytes of RAM in 8051 assigned address 00 to 7FH.

- The 128 bytes are divided into 3 different groups as follows :
  (i) A total of 32 bytes from location 00 to 1F hex are set aside for register banks and the stacks.
  (ii) A total of 16 bytes from locations 20H to 2FH are set aside for bit addressable read/write memory.
  (iii) A total of 80 bytes from location 30H to 7FH are used for read and write storage called as scratch pad.

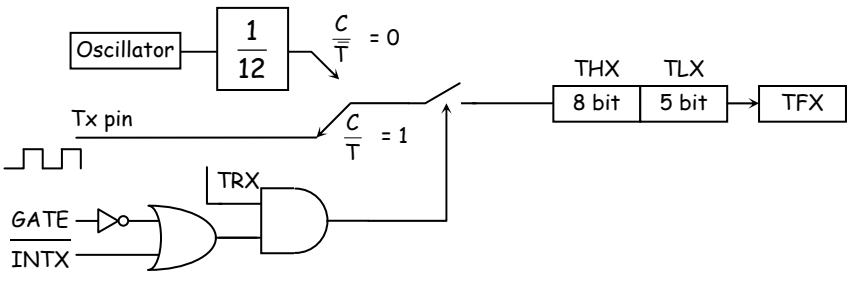
Q.2 (c) Explain different timer modes in 8051. [4]

(A) Timer Modes of 8051 Microcontroller

8051 microcontroller has 4 timer modes. Timer modes can be selected with the help of TMOD register.

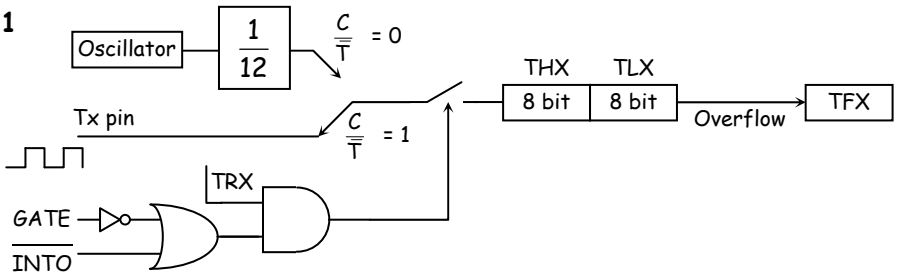
M_1	M_0	Timer Mode
0	0	Mode 0
0	1	Mode 1
1	0	Mode 2
1	1	Mode 3

Mode 0



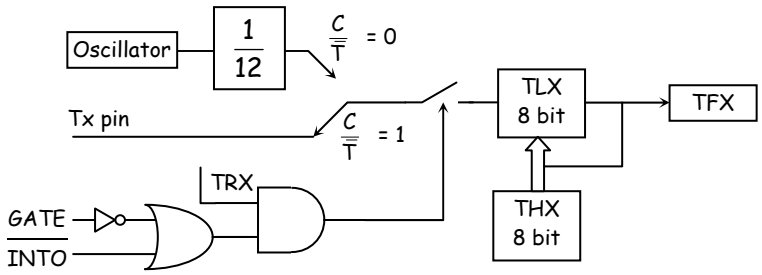
In mode 0, timer register is configured as 13 bit register, 8 bit for THX and 5 bit for TLX. Timer is enable when TRX = 1 and gate = 0 then register can hold values between 0000H to 1FFFH in TH and TL.
When register value reaches its maximum it roles over to 0000H, when the register overflow it sets timer interrupt flag TF one or zero.

Mode 1



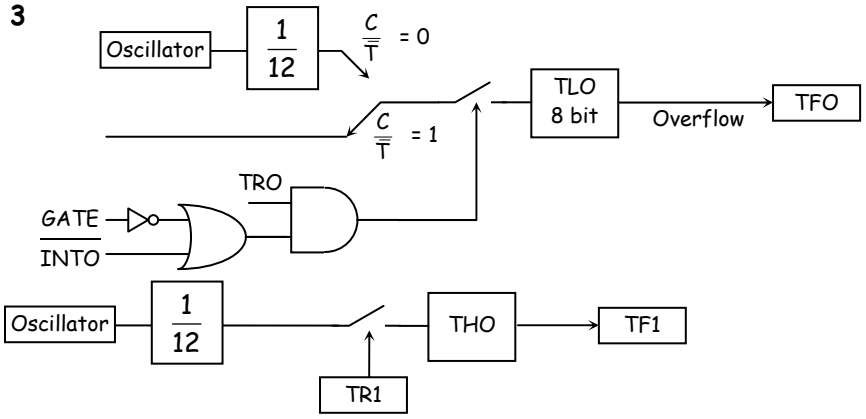
In mode 1, timer register is configured as 16 bit register where 8 bit for THX and 8 bit for TLX and gate = 0, then registers can hold value between 0000H to FFFFH in TH and TL.

Mode 2



The operation is some for timer 0 and timer 1. In this mode, timer register is configured as TLX. When TLX overflows from FFH to 00H it sets the flag TFX and it loads THX. The reload leaves THX unchanged. In this mode timer 1 or timer 0 supports the automatic reload operation. The timer control logic is same as mode 0 or mode 1.

Mode 3



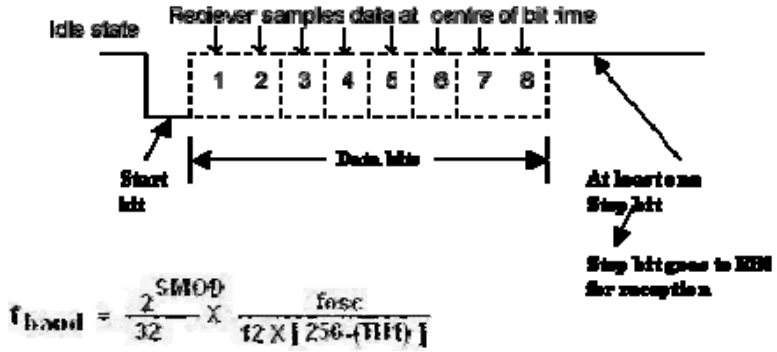
Timer 0 in mode 3 TL0 and TH0 are 2 separate registers as shown in figure. TL0 uses gate TR0, INT0 and TF0 control bit of timer 0 and TH0 uses TR1 and TF1 control bit of timer 1. When TL0 overflows from FFH to 00H, then it sets the timer flag TF0. If control bit TR1 is set, TH0 receives the timer clock. When counter TH0 overflows from FF to 00 then it sets the flag TF1.
Timer 1 may still be used in mode 0/1/2 but it neither interrupt the processor nor sets the flag. When timer 0 is in mode 3, timer 1 can be used for baud rate generation in serial communication.

Q.2 (d) Explain different serial communication modes of 8051. [4]

(A) 8051 micro controller communicate with another peripheral device through RXD and TXD pin of port3.controller have four mode of serial communication.

1. **Serial Data Mode-0 (Baud Rate Fixed)**
   In this mode, the serial port works like a shift register and the data transmission works synchronously with a clock frequency of $f_{osc}/12$. 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 f_{osc} /12, which is connected to the external circuitry for synchronization. The shift frequency or baud rate is always 1/12 of the oscillator frequency.

2. **Serial Data Mode-1 (standard UART mode)(baud rate is variable)**
   In mode-1, the serial port functions as a standard Universa 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**.



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

3. **Serial Data Mode-2 Multiprocessor (baud rate is fixed)**
   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 9th (TB8 or RB8)bit and a stop bit (usually '1'). While transmitting, the 9th 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.
   $f_{baud} = (2^{SMOD}/64)\ f_{osc}$

4. **Serial Data Mode-3 - Multi processor mode(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).
   $f_{baud} = (2^{SMOD}/32) * (f_{osc} / 12 (256-TH1))$

Q.2 (e) Draw and explain the format of SCON reg. [4]

(A)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0	SCON.7	Serial port mode specifier
SM1	SCON.6	Serial port mode specifier.
SM2	SCON.5	Used for multiprocessor communication (Makeit0.)
REN	SCON.4	Set/cleared by software to enable/disable reception.
TB8	SCON.3	Not widely used.
TB8	SCON.2	Not widely used.
TI	SCON.1	Transmit interrupt flag. Set by hardware at the beginning of the stop Bit in mode 1. Must be cleared by software.

RI	SCON.0	Receive interrupt flag. Set by hardware halfway through the stop bit time in mode 1. Must be cleared be cleaned by ft

Note: Make SM2, TB8 and RB8 = 0.

SM0	SM1	
0	0	Serial Mode0
0	1	Serial Mode1, 8-bit data, 1 stop bit, 1 start bit
1	0	Serial Mode2

SM2 : SM2 is the D5 bit of the SCON register.
This bit enables the multiprocessing capability of the 8051.. Make SM2 = 0 since we are not using the 8051 in a multiprocessor or environment.
This bit enables the multiprocessing capability of the8051. MakeSM2=0 since we are not using the 8051 in a multiprocessor environment.
REN: The REN (receive enable) bit is D4 of the SCON register. The REN bit is also referred to as SCON.4 since SCON is a bit addressable register.
When the REN =1, it allows the 8051 to receive data on the RxD pin of the 8051. As a result, if we want the8051 to both transfer and receive data, REN must be set to 1.
By making REN=0, the receiver is disabled. Making REN=1 or REN=0can be achieved by the instructions "SETBSCON.4" and "CLR SCON.4", respectively.
This bit can be used to block any serial data reception and is an extremely important bit in the SCON register.
TB8: TB8 (transfer bit 8) is bit D3 of SCON. It is used for serial modes 2 and 3.
WemakeTB8=0 since it is not used in our applications.
RB8: RB8 (receive bit 8) is bit D2 of the SCON register. In serial mode1, this bit gets copy of the stop bit when an 8-bitdata is received. This bit (as is the case for TB8) is rarely used any more. In all our applications wewillmakeRB8=0.Like TB8, the RB8 bit is also used in serial modes 2and 3.
TI: TI (transmit interrupt)is bit D1 of the SCON register. This is an extremely important flag bit in the SCON register.
When the 8051 finishes the transfer of the 8-bit character, it raises the T1flagto indicate that it is ready to transfer another byte. The TI bit is raised at the beginning of the stop bit.
RI: RI(receive interrupt)is theD0 bit of the SCON register. This is another extremely important flag in the SCON register.
When the 8051 receives data serially via RxD, it gets rid of the start and stop bits and places the byte in the SBUF register.
Then it raises the RI flag bit to indicate that a byte has been received and picked up before it is lost. RI is raised halfway through the stop bit.

Q.2 (f) Explain following directing with example **[4]**

(i) ORG **(ii) EQU**
(iii) DB **(iv) Code**

(A) Assembler directives :

(i) ORG : The ORG directive is used to indicate the starting address. It can be used only when program counter needs to be changed. It can be decimal or hex.
e.g. 2000H; set program counter 2000

(ii) DB (Define byte) : DB directive is used to define 8 bit data. It initialized memory with 8 bit value. It can be decimal, binary or in ASCII formats.
e.g. (1) DB 40H ; Hex
(2) DB 01011100B ; binary
(3) DB 16 ; Decimal
(4) DB 'HELLO' ; ASCII

(iii) EQU (or SET) : These directives assigns numerical value or register name to the specified symbol name. EQU is used to define constant without storing it in memory. Symbol defined EQU should not be redefined whereas SET allows redefinition of symbol.

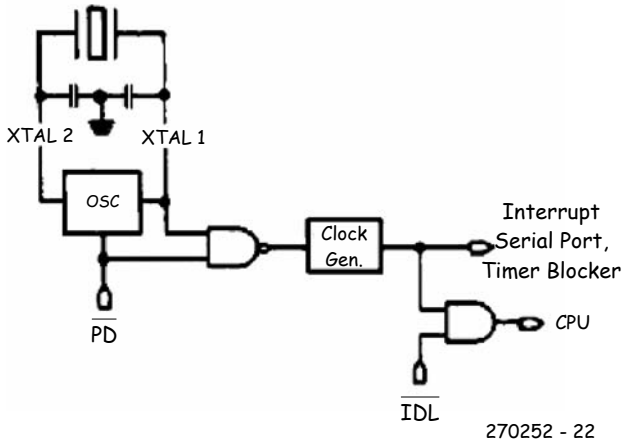
e.g. (1) SET R1 ; use R1 as pointer
(2) EQU R1 ; use R1 as a counter

(iv) END : This indicates END of program to assembler any text after END is ignored, if END is missing assembler generates error message.

Q.3 Attempt any FOUR of the following : [16]

Q.3 (a) Draw and explain power saving options of 8051. [4]

(A)



Format of PCON:

PCON: POWER CONTROL REGISTER. NOT BIT ADDRESSABLE.

SMOD	–	–	–	GF1	GF0	PD	IDL

SMOD Double baud rate bit. If Timer 1 is used to generate baud rate and SMOD = 1, the baud rate is double when the Serial Port is used in modes 1, 2, or 3.

– Not implemented, reserved for future use.*
– Not implemented, reserved for future use*
– Not implemented, reserved for future use*

GF1 General purpose flag bit.
GFQ General purpose flag bit.
FD Power Down bit. Setting this bit activates Power Down operation in the 80C51BH.
IDL Idle Mode bit. Setting this bit activates Idle Mode operation in the 80C51BH.

IDLE MODE

In the Idle mode, the internal clock signal is gated off to the CPU, but not to the Interrupt, Timer, and Serial Port functions.

The CPU status is preserved in its entirety, the Stack Pointer, Program Counter, Program Status Word, Accumulator, and all other registers maintain their data during Idle. The port pins hold the logical state they had at the time idle mode was activated. ALE and PSEN hold at logic high levels.

There are two ways to terminate the idle mode.

i) Activation of any enabled interrupt will cause PCON.O to be cleared and idle mode is terminated.
ii) Hard ware reset: that is signal at RST pin clears IDEAL bit IN PCON register directly. At this time, CPU resumes the program execution from where it left off.

POWER DOWN MODE

An instruction that sets PCON.1 causes that to be the last instruction executed before going into the Power Down mode. In the Power Down mode, the on-chip oscillator is stopped. With the clock frozen, all functions are stopped, but the on-chip RAM and Special Function Register are maintained held. The port pins output the values held by their respective SFRS. ALE and PSEN are held low.

Termination from power down mode: an exit from this mode is hardware reset.
Reset defines all SFRs but doesn't change on chip RAM.

Q.3 (b) State the function of editor, assembler, compiler and linker. [4]

(A) **(i) Editor:** An editor is a program which helps you to construct your assembly language program in right format so that the assembler will translate it correctly to machine language. So, you can type your program using editor. This form of your program is called as source program and extension of program must be .asm or .src depending on which assembler is used. The DOS based editor such as EDIT, WordStar, and Norton Editor etc. can be used to type your program.

(ii) Assembler: An assembler is programs that translate assembly language program to the correct binary code for each instruction i.e. machine code and generate the file called as Object file with extension .obj and list file with extension .lst extension.
Some examples of assembler are ASEM-51, Kiel's A51, AX 51 and C51, Intel PL/M-51 etc.

(iii) Compiler: Instructions in assembly language are represented in the form of meaningful abbreviations, and the process of their compiling into executable code is left over to a special program on a PC called compiler.

(iv) Linker: A linker is a program, which combines, if requested, more than one separately assembled object files into one executable program, such as two or more programs and also generate .abs file and initializes it with special instructions to facilitate its subsequent loading the execution.
Some examples of linker are ASEM-51 BL51, Keil u Vision Debugger, LX 51 Enhanced Linker etc.

Q.3 (c) Write assembly language programe to transfer msg 'MIC' serially at the band rate of 4800 with 8 data bits and 1 stop bit do it continuously. [4]

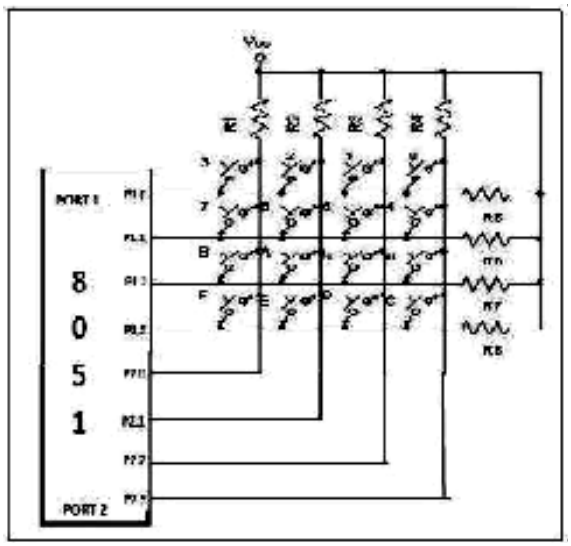
(A)

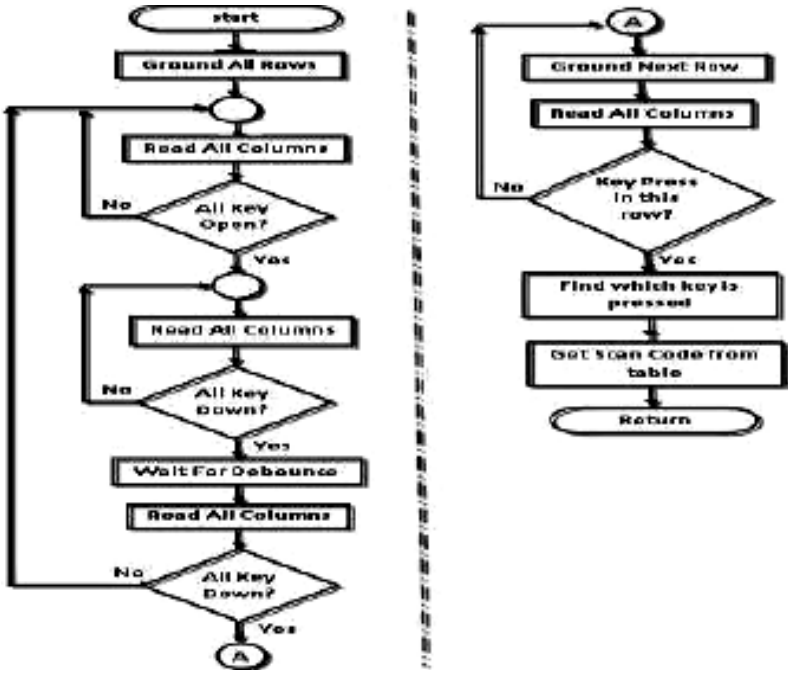
```
        ; Initialization
        MOV TMOD, #20H ;        timer1, mode2
        MOV TH1, #-6            ; 4800 baud
        MOV SCON, #50H          ; 8 bit data , 1 stop bit
        ; Begin to transmit
        SETB TR1 ; start Timer 1
AGAIN:  MOV A, # 'M'            ; transfer 'M'
        CALL TRANS
        MOV A, #'I'             ; transfer 'I'
        CALL TRANS
        MOV A, #'C'             ; transfer 'C'
        CALL TRANS
        SJMP AGAIN              ; keep doing it
        ; Serial transmitting subroutine
TRANS:  MOV SBUF, A             ; load SBUF
HERE:   JNB TI, HERE            ; wait for the last bit to transfer
        CLR TI                  ; get ready for the next byte
        RET
```

NOTE: Program may change. Student can also use the other logic.
Please check the logic and understanding of students.)

Q.3 (d) Draw interfacing diagram showing 4 × 4 matrix keyboard connection to Port 2 and port 1 of 8051. Draw flow chart to detect pressed key. **[4]**

(A)

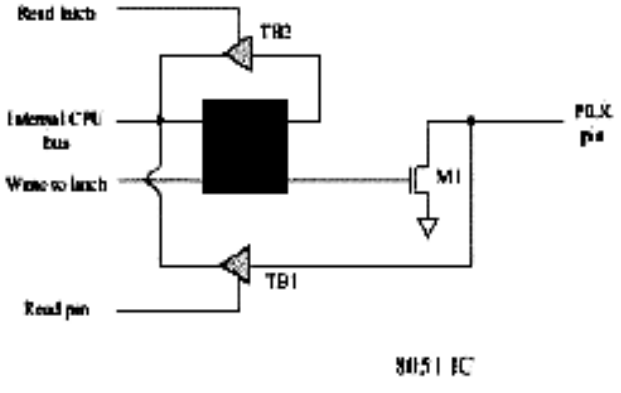




Q.3 (e) Draw the circuit diagram at port 0 of 8051 and describe its functions. **[4]**

(A)

A Pin of Port 0



Port 0: It may be used as input/output or bidirectional low order address and data bus for external memory. It does not have internal pull up resistors.

Q.4 (a) Attempt any THREE of the following: **[12]**

Q.4 (a) (i) List all alternate functions of port3 of 8051. **[4]**

(A)

Pin	Name	Alternate Function
P3.0	RXD	Serial input line
P3.1	TXD	Serial output line
P3.2	$\overline{INT0}$	External interrupt 0
P3.3	$\overline{INT1}$	External interrupt 1
P3.4	T0	Timer 0 external input
P3.5	T1	Timer 0 external input
P3.6	$\overline{WR}$	External data memory wirte strobe
P3.7	$\overline{RD}$	External data memory read strobe

Q.4 (a) (ii) What values should be loaded in TH1 of 8051 microcontroller to obtain 4800 Baud rate ? Assume crystal frequency = 11.0592 MHz. Give answer in both decimal and Hex. **[4]**

(A) The count to be loaded in TH1 to have a baud rate of 4800 can be calculated as follows

TH1=256d- [(K*oscillator frequency)/(384d*baud rate)]

Where, SMOD = 0 then K = 1 and SMOD = 1 then K = 2

Therefore,

TH1 = 256 - [(1*11.0592*10^6)/(384*4800)]

TH1 = 250d

TH1 = 0FAH

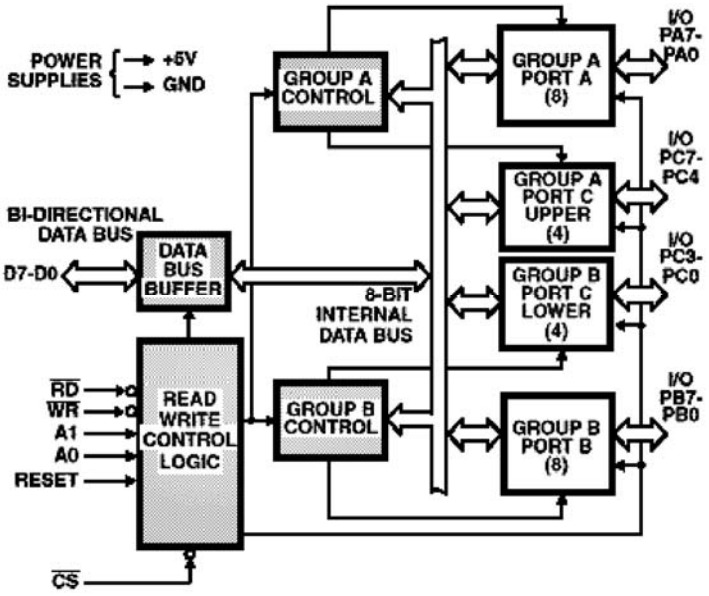
Q.4 (a) (iii) Write an assembly language program for 8051 microcontroller to add five 8-bit numbers. stored in internal RAM from 20H onwards. Store the result at 30H **[4]**

(A) Program for addition of five 8 bit nos.

```
CLR PSW.3; Select register Bank 0
CLR PSWW.4;
MOV R0, # 05H                ; Initialize byte counter
MOV R1, #20H                 ; Initialize memory pointer
MOVA, #00H                   ; Clear Accumulator
UP: ADD A @R1                ; Add accumulator with number from array
INC R1; Increment memory pointer
DJNZ R0, UP                  ; Decrement byte counter,
                             ; if byte counter ≠0
                             ; Then go to UP
MOV30H, A                    ; Store result in internal memory
HERE: SJMP HERE              ; Stop
```

Q.4 (a) (iv) Draw Block diagram of 8255. Discuss the operating modes of it. [4]

(A)



The following is an overview of each of the three available modes:

1) MODE 0 (Basic Input/Output):
   - Two eight bit ports and two 4 bit ports
   - Any port can be either input or output
   - Outputs are latched
   - Inputs are not latched
   - 16 different Input/output configurations are possible in this mode.

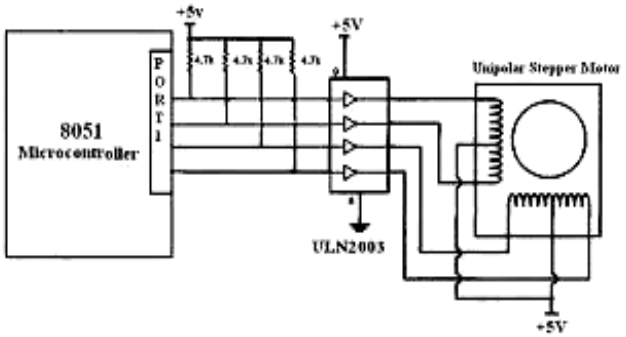
2) MODE 1 (Strobed Input/output):
   - Two Groups (Group A and Group B)
   - Each group contains one 8 bit data port and (1) 4 bit control/data port
   - The 8 bit data port can be either input or output
   - Both inputs and outputs are latched
   - The 4 bit port is used for control and status of the 8 bit data port

3) MODE 2 (Strobed Bi–directional Bus I/O):
   - Used in Group A only
   - One 8 bit bi–directional bus port (Port A) and one (5) bit control port (port C)
   - Both inputs and outputs are latched
   - The 5 bit control port (Port C) is used for control and status for the (8) bit, bi-directional bus port (Port A)

Q.4 (b) Attempt any ONE of the following : [6]

Q.4 (b) (i) Draw interfacing diagram of stepper motor with 8051 microcontroller and write ALP to rotate in anticlockwise direction through 180°. Assume step angle 1.8°. [6]

(A)



Step no	Winding A	Winding B	Winding C	Winding D	anticlockwise
1	1	0	0	1	↑
2	1	1	0	0	
3	0	1	1	0	
4	0	0	1	1	

Calculation for no. of steps:

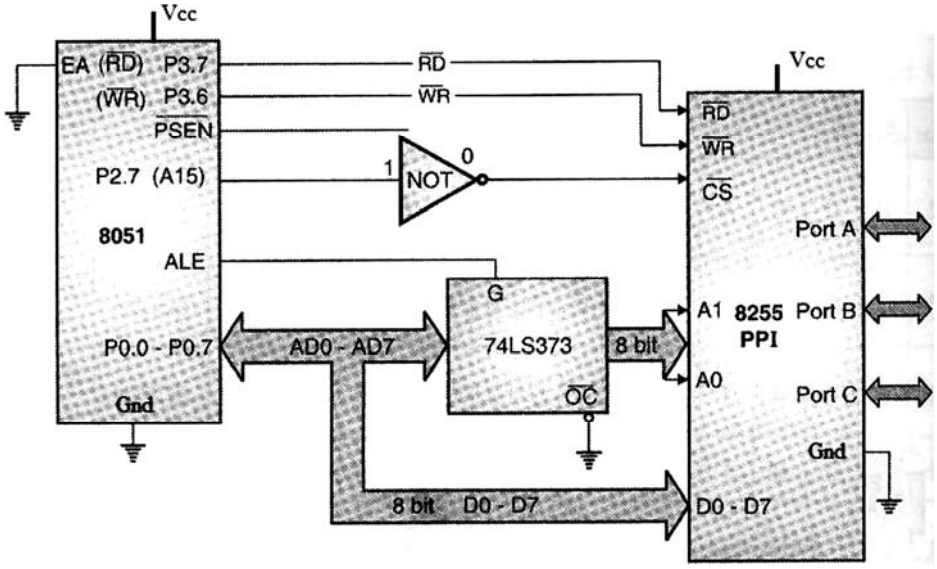
$$\textbf{Count} = \frac{\text{Required Rotation Angle}}{1.8\,\text{degree} * \text{no. of steps}}$$

= 180/1.8*4 steps=100/4=25

```
PROGRAM:
        MOV A, #66H          ; load step sequence
        MOV R0, # 25         ; count for 4 steps
BACK:   MOV P1, A            ; issue sequence to motor
AGAIN:RL A                   ; rotate left anticlockwise
        ACALL DELAY          ; wait
        DJNZ R0, AGAIN
        SJMP BACK            ; keep going
DELAY                        ; delay subroutine.
        MOV R2, #100
H1 :    MOV R3, #255
H2 :    DJNZ R3, H2
        DJNZ R2, H1
        RET
```

Q.4 (b) (ii) Draw interfacing diagram of 8051 microcontroller with 8255. State I/O port address and control word register address. [6]

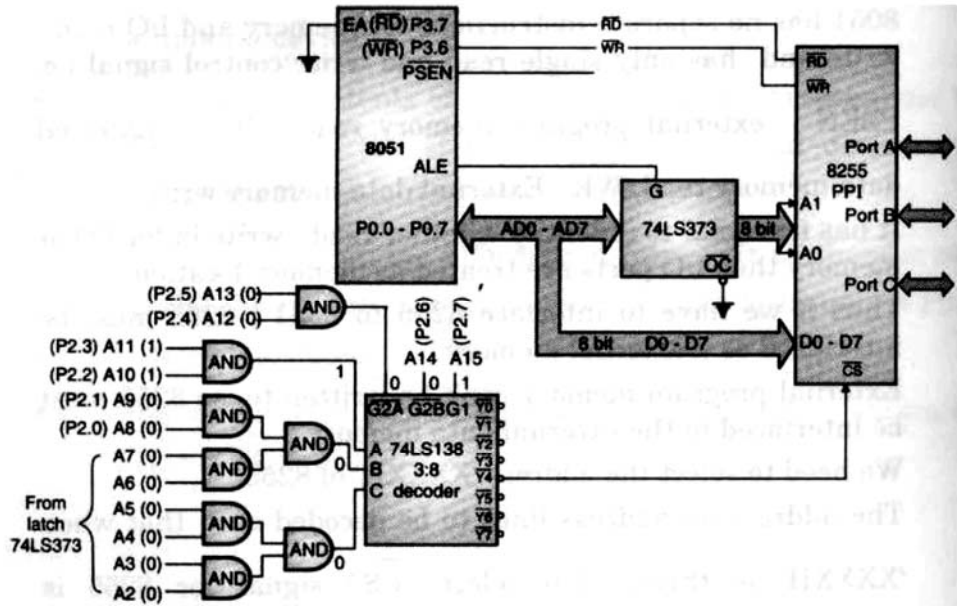
(A) (Note: Interfacing and address mapping using absolute decoding should also be considered)



Address Mapping:

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Add	Register
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000H	Port A
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	8001H	Port B
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	8002H	Port C
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	8003H	Control Word

OR



Address Mapping:

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Add	Register
1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	8C00H	Port A
1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	8C01H	Port B
1	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	8C02H	Port C
1	0	0	0	1	1	0	0	0	0	0	0	0	0	1	1	8C03H	Control Word

Q.5 Attempt any FOUR of the following : **[16]**

Q.5 (a) Differentiate between linear and absolute address decoding. **[4]**

(A)

	Absolute Decoding	Linear Decoding
1)	It is also called as full decoding as all the	It is also called as partial decoding as all

	address lines are used for decoding.	address lines are not used for decoding.
2)	It is used in large memory systems.	It is used in small system.
3)	Hardware required for decoding logic is more.	Hardware used for decoding logic is eliminated.
4)	Multiple addresses are not generated.	Multiple address are generated.

Q.5 (b) Write an assembly language programme for 8051 microcontroller to turn on led connected to P1.7 on the occurrence of $\overline{\text{INT0}}$ and turn off LED after some delay. **[4]**

(A)
```
    MOV A, # 80H          ; Store 80H in A
    HERE: JNB P3.2 HERE   ; wait for INT0 interrupt
BACK: MOV P1, A           ; move 80 to P1 to glow the LED connected
                            to P1.7
    ACALL DELAY           : wait for some time
    MOV A, #00H           ; turn off the LED
    SJMP BACK
```

Delay Program []
```
AGAIN: MOV R3, #255
    DJNZ R3, AGAIN
    RET
```

Q.5 (c) Describe any four selection factor at microcontroller. **[4]**

(A) The selection of microcontroller depends upon the type of application. The following factors must be considered while selecting the microcontroller.

(i) Word length: The word length of microcontroller is either 8, 16 or 32 bit. As the word length increases, the cost, power dissipation and speed of the microcontroller increases.

(ii) Power dissipation: It depends upon various factors like clock frequency, speed, supply voltage, VLSI technology etc. For battery operated embedded systems, we must use low power microcontrollers.

(iii) Clock frequency: The speed of an embedded system depends upon the clock frequency. The clock frequency depends upon the application.

(iv) Instruction Set: On the basis of instructions microcontrollers are classified into two categories 1. CISC 2. RISC.
CISC system improves software flexibility. Hence it is used in general purpose systems. RISC improves speed of the system for the particular applications.

(v) Internal resources: The internal resources are ROM, RAM, EEPROM, FLASH, ROM, UART, TIMER, watch dog timer, PWM, ADC, DAC, network interface, wireless interface etc. It depends upon the application for which microcontroller is going to be used.

(vi) I/O capabilities: The number of I/O ports, size and characteristics of each I/O port, speed of operation of the I/O port, serial port or parallel ports. These are the considerations needed to ascertain.

Q.5 (d) List priority at interrupts along with their vector add. How it can be changed. **[4]**

(A) All the 5 interrupts of 8051 has got different priorities. Interrupts are serviced according to it's priority order. From the table above, you can see that INT0 has the highest priority of 1 and Timer 0 comes next with priority value 2. The order of priority works like this – consider a case where two interrupts are raised at the same time – one from INT0 and another from Timer 1 interrupt. In such a case, processor would serve the interrupt according to it's priority. In our case INT0 is of high priority (priority order 1)and Timer 1 interrupt is of low priority (priority order 4). So processor will execute ISR of INT0 first and then later, after finishing ISR of INT0, processor will begin executing ISR of Timer 1 interrupt.
The priority of the interrupts can be modified by programming the contents of IP register.

Q.5 (e) Explain different addressing modes with one example each

(A) There are a number of addressing modes available to the 8051 instruction set, as follows:

1. Immediate Addressing mode
2. Register Addressing mode
3. Direct Addressing mode
4. Register Indirect addressing mode
5. Relative Addressing mode
6. Indexed Addressing mode

1) **Immediate Addressing mode:**

Immediate addressing simply means that the operand (which immediately follows the Instruction op. code) is the data value to be used.

For example the instruction:
MOV A, #25H; Load 25H into A

Move the value 25H into the accumulator. The # symbol tells the assembler that the immediate addressing mode is to be used.

2) Register Addressing Mode:

One of the eight general-registers, R0 to R7, can be specified as the instruction Operand. The assembly language documentation refers to a register generically as Rn.
An example instruction using register addressing is:
ADD A, R5; add the contents of register R5 to contents of A (accumulator)
Here the contents of R5 are added to the accumulator. One advantage of register addressing is that the instructions tend to be short, single byte instructions.

3) Direct Addressing Mode:

Direct addressing means that the data value is obtained directly from the memory location specified in the operand.
For example consider the instruction:
MOV R0, 40H; Save contents of RAM location 40H in R0.
The instruction reads the data from Internal RAM address 40H and stores this in theR0. Direct addressing can be used to access Internal RAM, including the SFR registers.

4) Register Indirect Addressing Mode:

Indirect addressing provides a powerful addressing capability, which needs to be appreciated.
An example instruction, which uses indirect addressing, is as follows:
MOV A, @R0; move contents of RAM location whose address is held by R0 into A
Note the @ symbol indicated that the indirect addressing mode is used. If the data is inside the CPU, only registers R0 & R1 are used for this purpose.

5) Relative Addressing Mode:

This is a special addressing mode used with certain jump instructions. The relative address, often referred to as an offset, is an 8-bit signed number, which is automatically added to the PC to make the address of the next instruction. The 8-bitsigned offset value gives an address range of + 127 to –128 locations.
Consider the following example:
SJMP LABEL_X
An advantage of relative addressing is that the program code is easy to relocate in memory in that the addressing is relative to the position in memory.

6) Absolute addressing Mode:
Absolute addressing within the 8051 is used only by the AJMP (Absolute Jump) and ACALL (Absolute Call) instructions.

7) Long Addressing Mode:
The long addressing mode within the 8051 is used with the instructions LJMP and LCALL. The address specifies a full 16 bit destination address so that a jump or a call can be made to a location within a 64KByte code memory space (216 = 64K).
An example instruction is:
LJMP 5000h; full 16 bit address is specified in operand

8) Indexed Addressing Mode:
With indexed addressing a separate register, either the program counter, PC, or the data pointer DTPR, is used as a base address and the accumulator is used as an offset address. The effective address is formed by adding the value from the base address to the value from the offset address. Indexed addressing in the 8051 is used with the JMP or MOVC instructions. Look up tables are easy to implement with the help of index addressing.

Consider the example instruction:
MOVC A, @A+DPTR
MOVC is a move instruction, which moves data from the external code memory space. The address operand in this example is formed by adding the content of the DPTR register to the accumulator value. Here the DPTR value is referred to as the base address and the accumulator value us referred to as the index address.
Consider the example instruction:
MOVC A, @A+DPTR
MOVC is a move instruction, which moves data from the external code memory space. The address operand in this example is formed by adding the content of the DPTR register to the accumulator value. Here the DPTR value is referred to as the base address and the accumulator value us referred to as the index address.

Q.6 Attempt any FOUR of the following : [16]

Q.6 (a) Write assembly language program to generate continuous sq. wave of 2KHz an P1.4 using timer 0. Assume crystal freq. = 11.0592 MHz. [4]

(A) Crystal frequency = 12 MHz
I/P clock = (11.059 X 10^6)/12 = 1000000 = 921.58KHz
T_{in} = 1.085µ sec
For 2 kHz square wave
F_{out} = 2 kHz
T_{out} = 1/ 2X 103
T_{out} = 0.5msec = 500us
Consider half of it = T_{out} = 250µ sec
N = T_{out} / T_{in} = 250/1.085 = 230.41
65536 – 231 = $(65305)_{10}$ = $(FF19)_{16}$

Program :

```
MOV TMOD, # 01H          ; Set timer 0 in Mode 1, i.e., 16 bit timer
L2:  MOV TLO, # 19H      ; Load TL register with LSB of count MOV
THO, # FFH               ; load TH register with MSB of count SETB TRO
     ; start timer 0
L1: JNB TFO, L1          ; poll till timer roll over
CLR TRO                  ; stop timer 0
CPL P1.4                 ; complement port 1.4 line to get high or low
```

```
CLR TFO          ; clear timer flag 0
SJMP L2          ; re-load timer with count as mode 1 is not
                   auto reload
```

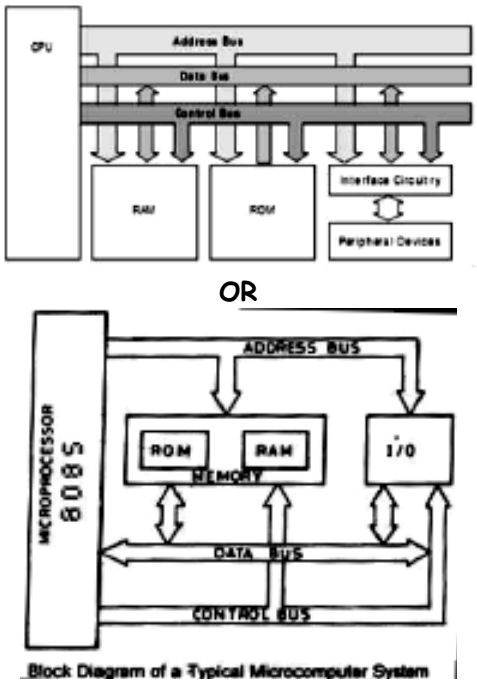
Q.6 (b) Write an assembly language program to arrange ten numbers in an ascending order. **[4]**

(A) **NOTE:** Program may change. Student can also use the other logic.
Please check the logic and understanding of students.)

```
        MOV R0, #0AH          ; Initialize the pass counter
UP1:    MOV DPTR, #3000H      ; Initialize the memory pointer
        MOV R1, #0AH          ; Initialize the byte counter
UP:     MOV R2, DPL           ; Save the lower byte address
        MOVX A, @DPTR         ; Read number from array
        MOV 0F0H, A           ; Copy number to B register
        INC DPTR              ; Increment memory pointer
        MOVX A,@DPTR          ; Read next number from array
        CJNE A , 0F0, DWN     ; Compare number with next number
        AJMP SKIP
DWN:    JNC SKIP              ; If number> next number then go to SKIP
        MOV DPL, R2           ; Else exchange the number with next number
        MOVX @DPTR, A
        INC DPTR
        MOV A, 0F0H
        MOVX @DPTR, A
        SKIP: DJNZ R1 UP      ; Decrement byte counter if not zero, go to UP
        DJNZ R0 UP1           ; Decrement pass counter if not zero, go to UP1
LOOP:   AJMP LOOP             ; Stop
```

Q.6 (c) Sketch block diagram of microcomputer and explain. **[4]**

(A)



Block Diagram of a Typical Microcomputer System

A microcomputer has three basic blocks: a central processing unit (CPU), a memory unit, and an input/output unit. The CPU executes all the instructions and performs arithmetic and logic operations on data. The CPU of the microcomputer is called the "microprocessor." The

microprocessor is typically a single VLSI (Very Large-Scale Integration) chip that contains all the registers, control unit, and arithmetic/ logic circuits of the microcomputer.

A memory unit stores both data and instructions. The memory section typically contains ROM and RAM chips. The ROM can only be read and is nonvolatile, that is, it retains its contents when the power is turned off. A ROM is typically used to store instructions and data that do not change. For example, it might store a table of codes for outputting data to a display external to the microcomputer for turning on a digit from 0 to 9.One can read from and write into a RAM. The RAM is volatile; that is, it does not retain its contents when the power is turned off. A RAM is used to store programs and data that are temporary and might change during the course of executing a program.

An I/0 (Input/Output) unit transfers data between the microcomputer and the external devices via I/0 ports (registers). The transfer involves data, status, and control signals.

The microcomputer's system bus contains three buses, which carry all the address, data, and control information involved in program execution. These buses connect the microprocessor (CPU) to each of the ROM, RAM, and I/O chips so that information transfer between the microprocessor and any of the other elements can take place.

Q.6 (d) Draw format of IE SFR and describe each bit. **[4]**

(A) IE: INTERRUPT ENABLE REGISTER, BIT ADDRESSABLE.
If the bit it 0, the corresponding interrupt is disbled. If the bit is 1, the corresponding interrupt is enabled.

EA	––	ET2	ES	ET1	EX1	ET0	EX0

EA	IE.7	Disables all in interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
––	IE.6	Not implemented, reserved for future use.
ET2	IE·5	Enable or disable the Timer 2 overflow or capture interrupt (8052 only)
ES	IE·4	Enable or disable the serial port interrupt.
ET1	IE·3	Enable or disable the Timer 1 overflow interrupt.
EX1	IE·2	Enable or disable External Interrupt 1.
ET0	IE·1	Enable or disable the Timer 0 overflow interrupt.
EX0	IE·0	Enable or disable External Interrupt 0.

* User software should not write is to reserved bits. These bits may be used in future MCS–51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

Q.6 (e) A switch is connected to P1.0 and LED to P2.7. Write a program to get the status of the switch and saved it to LED. **[4]**

(A) NOTE: Program may change. Student can also use the other logic Please check the logic and understanding of students.

```
NEXT: MOV A,P1        ;Read key status from P1
ANL A,#01H            ;check P1.0 bit of port 1
CJNEA,#01H,DN         ;If not pressed then NEXT
MOV P2,A              ;Make LED ON
```

```
ACALL Delay          ; Wait for sometime
SJMP NEXT
DN: MOV P2,A         ;Make LED OFF
ACALL Delay          ;Wait for sometime
SJMP NEXT
```

❑ ❑ ❑ ❑ ❑