KLS Vishwanathrao Deshpande Institute of Technology

(Accredited by NAAC with "A" Grade)
(Approved by AICTE, New Delhi, Affiliated to VTU, Belagavi)
(Recognized Under Section 2(f) by UGC, New Delhi)
Udyog Vidya Nagar, Haliyal - 581 329, Dist.: Uttara Kannada
Phone: 08284 - 220861, 220334, 221409, Fax: 08284 - 220813
www.klsvdit.edu.in | principal@klsvdit.edu.in | hodece@klsvdit.edu.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

University / Model Question Paper
Scheme & Solution

Faculty Name	:	RAHUL. C. M.
Course Name	:	8051 Microcontroller
Course Code	:	BEC405A.
Year of Question Paper	:	2023 - 24 (Model)
Date of Submission	:	19/08/2024

Rahul. C. M
Faculty Member

HoD

Head of the Department
Dept. of Electronics & Communication Engg.

19/8/24
Dean (Acad.)

BEC405A

Model Question Paper-II with effect from 2023-24 (CBCS Scheme)

USN | | | | | | | | | | |

Fourth Semester B.E. Degree Examination
8051 MICROCONTROLLER

TIME: 03 Hours **Max. Marks: 100**

Note: Answer any **FIVE** full questions, choosing at least **ONE** question from each **MODULE**.

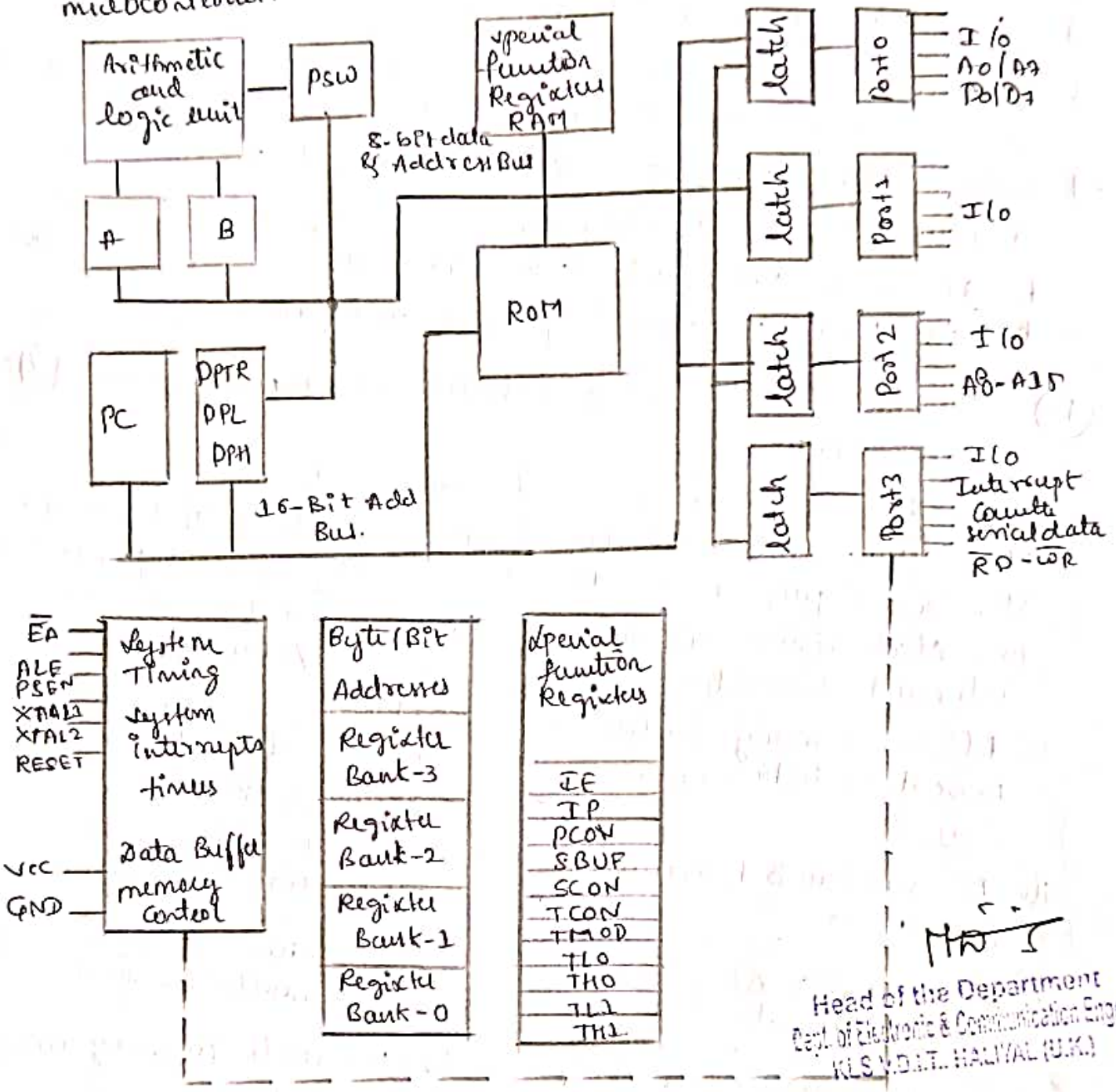
		Module -1	*Bloom's Taxonomy Level	Marks
Q.01	a	With a neat architecture diagram explain the architectural features of 8051 microcontroller.	L2	8
	b	Bring out the difference between Microprocessor and Microcontroller.	L1	4
	c	With simple block diagram explain the features of 8051 microcontroller.	L2	8
		OR		
Q.02	a	Interface 4k bytes RAM and 8k bytes ROM to 8051 microcontroller in such a way that starting address of RAM is 1000H and ROM is C000H.	L2	8
	b	With function of each pin explain the pin layout of 8051 microcontroller	L2	8
	c	Define microcontroller, mention its applications.	L1	4
		Module-2		
Q. 03	a	Explain 5 different addressing modes with examples	L1	8
	b	Write an ALP to convert a packed BCD number into two ASCII numbers. Store the result in R5 and R6 respectively	L3	6
	c	List and explain bit level logical instructions in 8051.	L2	6
		OR		
Q.04	a	Define assembler directives. With example explain all the assembler directives supported by 8051 microcontroller.	L2	8
	b	Write an ALP to convert a Binary number to packed BCD number (hexadecimal to decimal). The binary number is stored at 40h location. Store the converted packed BCD number at 50h and 51h internal RAM location.	L3	6
	c	With neat diagram explain the range of JUMP instructions	L2	6
		Module-3		
Q. 05	a	Explain the bit contents of TCON and TMOD registers.	L1	8
	b	Write an assembly language program to transfer multi-byte data serially with 9600 baud rate.	L3	8
	c	Explain how timers are programmed in mode 1.	L2	4
		OR		
Q. 06	a	Explain the bit pattern of SCON register with diagram	L2	6
	b	Assuming XTAL frequency as 11.0592Mhz, write a program to generate 4Khz square wave on P2.1. Use timer 0 in model show all the calculations	L3	8
	c	Explain RS232 in serial communication using 8051 microcontroller	L2	6
		Module-4		
Q. 07	a	Explain how interrupt priority can be changed using IP register. Also explain the default priorities assigned to interrupts in 8051 microcontroller.	L2	8
	b	Explain how programming of external hardware interrupts is done in 8051 microcontrollers with a code.	L2	8
	c	Explain programming of timer interrupts.	L1	4
		OR		
Q. 08	a	Explain the bit contents of IE register.	L1	4
	b	Write a C program using interrupts to generate a square wave on port pin P1.2 of 1kHz using timer-0 in mode 2.	L3	8
	c	Explain programming of serial communication interrupts	L2	8
		Module-5		

Q. 09	a	With neat diagram write an assembly language program to interface DAC to 8051 microcontroller.	L3	10
	b	Write a program to display "HELLO WORLD" by interfacing LCD display to 8051 microcontroller	L3	10
OR				
Q. 10	a	With neat diagram write an assembly language program to interface ADC0804 to 8051 microcontroller.	L3	10
	b	With neat diagram explain the interfacing of stepper motor using 8051 microcontroller.	L3	10

Module-01.

(1a) With a neat diagram explain the architecture features of [8M] 8051 microcontroller.

⟹ The below figure shows the architecture of classic 8051 microcontroller.



Head of the Department
Dept. of Electronics & Communication Engg
KLS V.D.I.T. HALIYAL (U.K.)

Features of 8051

An 8-bit ALU with A and B Registers, 8-bit PSW.
16-bit address and 8-bit data Bus.
16-bit Program Counter (PC) and 16-bit data pointer (DPTR)
8-bit stack pointer (SP) initial default value is 07h
SFR: TCON, TMOD, SCON, PCON, SBUF, IP and IE etc.
Two 16-bit timers/counters T0 and T1.

Rohit C. M.

Two external interrupts INT0 and INT1 and 3-internal interrupts T0, T1 & SI.
Full duplex UART serial interface.
32 I/O pins arranged as 04 8-bit ports :- P0-P3.
Special bit manipulatid instructions
Internal ROM of 4KB, 8751 - EPROM : 8951 - EEPROM 80
31 - 0 bytes. Extendable upto 64KB.
Internal RAM of 128 bytes, Extendable up to 64KB.
* four register banks, each containing 8-registers (32 bytes)
* 16- bytes bit addressable memory
* 80- bytes of general-purpose data memory.

(1b) Bring out the difference between Microprocessor and Microcontroller. [4M]

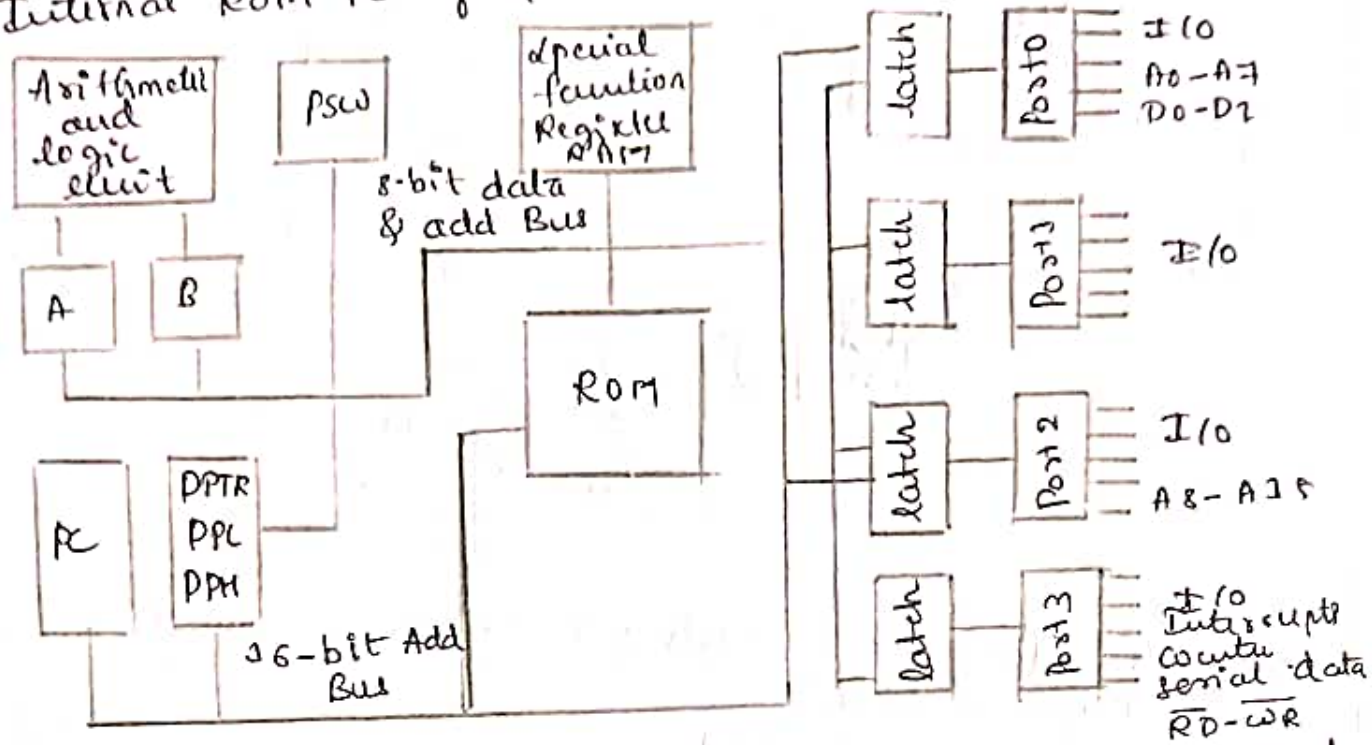
Microprocessor	Microcontroller
(i) Microprocessor contain ALU SP, General purpose registers, PC, clock timing circuit, interrupt circuit.	(i) Microcontroller contains the circuitry of microprocessor RAM, ROM I/O devices, Timers / counters etc.
(ii) It has a many instruction move data between memory & CPU	(ii) It has a few instruction to move data between memory & CPU.
(iii) It has few bit handling instruction	(iii) It has many bit handling instruction
(iv) less number of pins are multifunctional	(iv) More number of pins are multifunctional.
(v) Single memory for data and code	(v) Separate memory map for data & code.

(1c) With a simple block diagram explain the features of 8051 microcontroller.

⟹ # The 8051 contains 8-bit arithmetic & logic unit to perform arithmetic operations like addition, subtraction, multiplication division & logical operations like AND, OR Clear etc using registers A, B, SP and PSW, DPTR & PC

Internal RAM is of 128-bytes organized into 3 distinct areas with 8- Registers in each block.
16-bytes of data addressable area.
General purpose RAM from 30h to 7Fh. [8M]

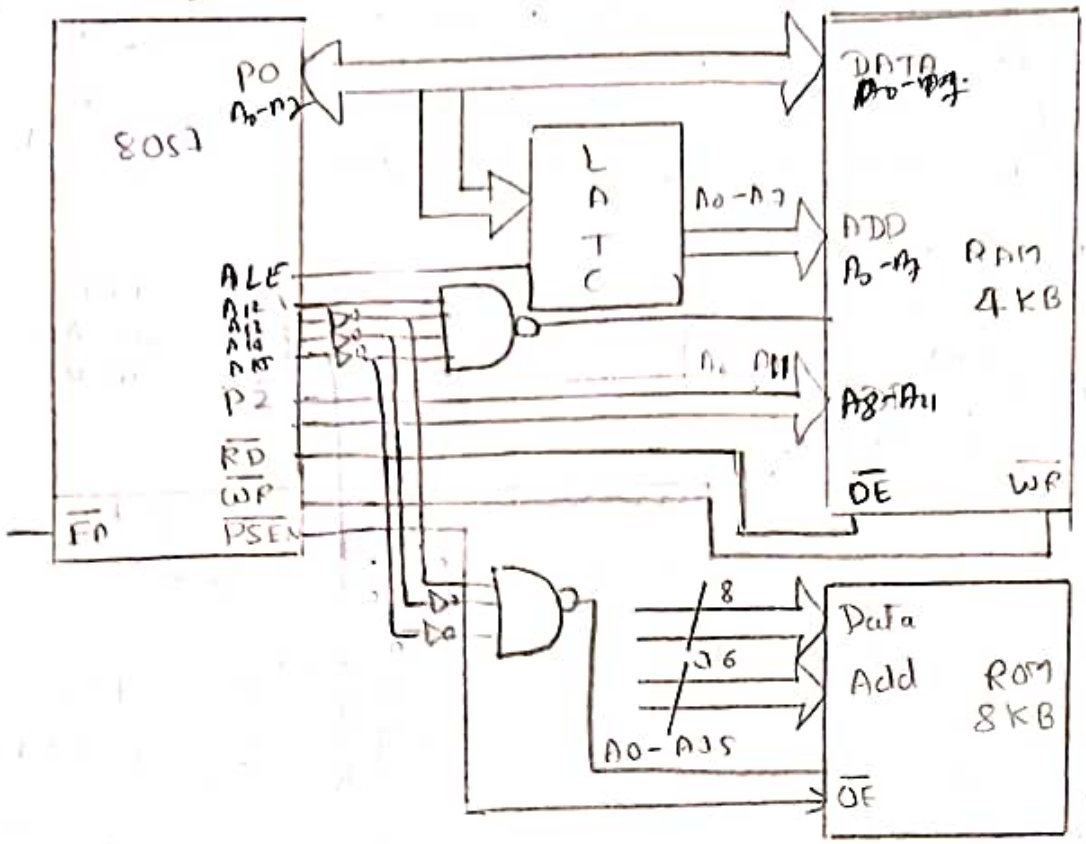
* Internal Rom is of 4KB.



Registers: Accumulator is a 8-bit general register used for transfer data and arithmetic operations
A and B register is used for arithmetic operations like multiplication division & for data storage.
SP is stack pointer register which holds the address of the TOP of stack.
I/O ports: 8051 has 32 I/O pin configured as 4-ports ie port0, port1, port2 and port3.
Full duplex serial data receiver / transmitter
Two external & internal interrupt sources.

(2a) Interface 4K bytes RAM and 8k bytes ROM to 8051 microcontroller in such a way that starting address of RAM is 1000h and ROM is C000h. [8M]

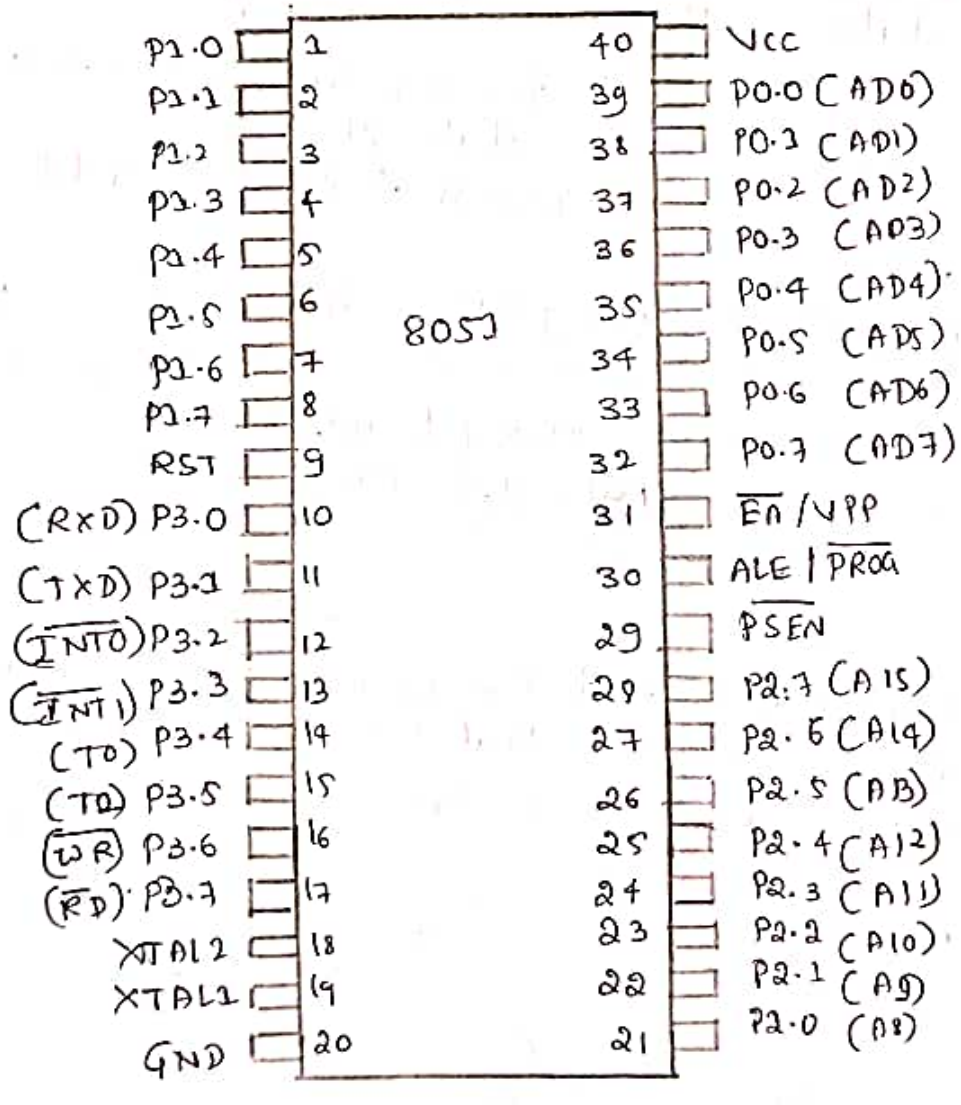
Roll C.M



(26) with a function of each pin explain the pin layout of 8051 microcontroller.

[8M]

⇒ Pin diagram of 8051.



Pins 1-8 : Port 1. Each of these can be configured as an input or an output

Pin 9 : RESET. A logic one on this pin disables the microcontroller and clears the contents of most registers.

Pin 10-17 : Port 3. Similar to port 2, each of these pins can serve as general input or output. Besides, all of them have alternative functions

Pin 10 : RXD: Serial asynchronous communication input or serial synchronous communication output.

Pin 11 : TXD: Serial asynchronous communication output or serial synchronous communication clock output

Pin 12 : INT0: External interrupt 0 input

Pin 13 : INT1: External interrupt 1 input.

Pin 14 : T0, Counter 0 clock input

Pin 15 : T1, counter 1 clock input

Pin 16 : WR. write to external RAM.

Pin 17 : RD. Read from external RAM.

Pin 18, 19 : XTAL2, XTAL1, Internal oscillator input and output. A quartz crystal which specifies operating frequency is usually is usually connected to these pins.

Pin 20 : GND, Ground.

Pin 21-28 : Port 2. If there is a intention to use external memory then these port pins are configured as general input / outputs.

Pin 29 : PSEN. If external ROM is used for storing program then a logic zero (0) appears on it every time the microcontroller reads a byte from memory.

Pin 30 : ALE. prior to reading from external memory, the microcontroller puts the lower address byte (A0-A7) on P0 and activates the ALE output.

Pin 31 : EA. By applying logic zero to pins this pin, P2 and P3 are used for data and address transmission with no regard to whether is internal memory or not.

Pin 32-39 : Port 0: Similar to P2, If external memory is not used, these pins can be used as general I/O.

Pin 40 : VCC, +5V power supply.

(2C) Define microcontroller, mention its applications. [4M]

Rahul C. M

⇒ A microcontroller is a programmable digital processor with ne-cessary peripherals. It is a small computer on a single inte-grated chip with one or more CPU's along with memory and I/O peripherals.

Application of Microcontroller:

* Mobile phones, Automobiles, keyboard controllers, watches etc
* The prime use of microcontroller is to control the operator of the machine using a fixed program that is stored in ROM & that does not change over the lifetime of the system.
* It is also meant to read, perform limited calculations on that & control its environment based on those calcula-tions.

Module - 02.

(3a) Explain 5 different addressing modes with examples. [8M]

→ (i) Immediate Addressing mode:

The mnemonic for immediate data is # sign. Immediate addressing modes transfers immediate addressing modes data to registers (R0 to R7 from the currently selected bank) register A & DPTR.

Ex: MOV R0, #00h
MOV R1, #01h
MOV P1, #05h

(ii) Register Addressing mode:

In register addressing mode both source & destination regi-ster size should match, The data can be moved between A & Rn but movement of data between Rn is invalid.

Ex: MOV A, R0
MOV R2, A
MOV DPTR, #1234h
MOV R4, DPH.

(iii) Direct Addressing mode:

In direct addressing modes, the entire, 128 bytes of internal RAM can be accessed. (i.e 00 to 7Fh) Above 7Fh (i.e 80h to FFh) only SFR can be accessed by using their names or by their access.

Ex: MOV A, add
MOV Rn, add
MOV add, Rn
MOV add, A

(iv) Indirect addressing mode: In the indirect addres mode, a register is used as pointer to the data ie it uses register R0 & R1 as a data pointer to hold the address of internal RAM locations from (00h to 7Fh) The mnemonic symbol used for indirect addressing is the '@' sign.

Ex: MOV A, @R0 ; Copy the content of the address in R0 to the A register.

MOV @R1, #35h ; Copy the number 35h to the address in R1.

MOV @R0, A ; Copy the content of A register to the address in R1 register.

(5b) Write an ALP to convert a packed BCD number into two ASCII numbers store the result in R5 & R6 respectively. [6M]

→
```
MOV A, #29h
MOV R5, A
ANL A, #0Fh
ORL A, #30h
MOV R6, A
MOV A, R5
ANL A, #0F0h
RR A
RR A
RR A
RR A
ORL A, #30h
MOV R2, A
End
```

OR SWAP A.

Head of the Department
Dept. of Electronic & Communication Engg.
K.L.S V.D.I.T., HALIYAL (U.K.)

(5c) List and explain bit level logical instruction in 8051. The bit-level Boolean logical opcodes operate on any addressable RAM or SFR bit. The carry flag (C) in the PSW special-function register is the destination for most of the opcodes because the flag can be tested and the program flow changed using instructions covered in [6M]

K.L.S VDIT HALIYAL

The Boolean bit-level operations are as follows.

Mnemonic	operation.
ANL C,b	AND C and the addressable bit, put the result in C
ANL C,/b	AND C and the complement of the addressed bit, put the result in C; the addressable bit is not altered.
ORL C,b	OR C and the addressed bit; put the result in C
ORL C,/b	OR C and the complement of the addressed bit, put the result in C; the addressed bit is not altered
CPL C	Complement the C flag
CPL b	Complement the addressed bit
CLR C	Clear the C flag to 0
CLR b	Clear the addressed bit to 0
MOV C,b	Copy the addressed bit to the C flag
MOV b,C	Copy the C flag to the addressed bit
SETB C	Set the C flag to 1
SETB b	Set the addressed bit to 1

(4a) Define assembler directives, with example explain all the assembler directives supported by 8051 microcontroller. [8M]

→ "Assembler directive tells the assembler to do something other than creating the machines code for an instruction".

(i) ORG (origin): The ORG directive is used to indicate the starting address. It can be used only when the program counter needs to be changed. The number that comes after ORG can be either in hexadecimal or in decimal.

```
Ex  org 0000h   ; set PC to 0000
```

(ii) EQU (equate):
This directive is used to define a constant without occupying a memory location, The EQU directive does. EQU directive does not set aside storage for a storage for a storage for data item but associated a constant value with a data label so that content the label appears in program.

```
Eg:  COUNT  EQU  25
     MOV    R3, #COUNT
```

(iii) DBC Define Byte): The DB directive is used to define an 8-bit data, which initializes memory with 8-bit values.
* The number can be in decimal, binary, hexadecimal or in ASCII formats. for decimal the 'D' after the decimal number is optional, but for binary & hexadecimal, 'B' & are

Eg:
```
DATA 1 : DB04h ; Hex
DATA 2 : DB01011100B ; binary
DATA 3 : DB48 ; decimal
DATA 4 : DB 'go corona' ; ASCII
```

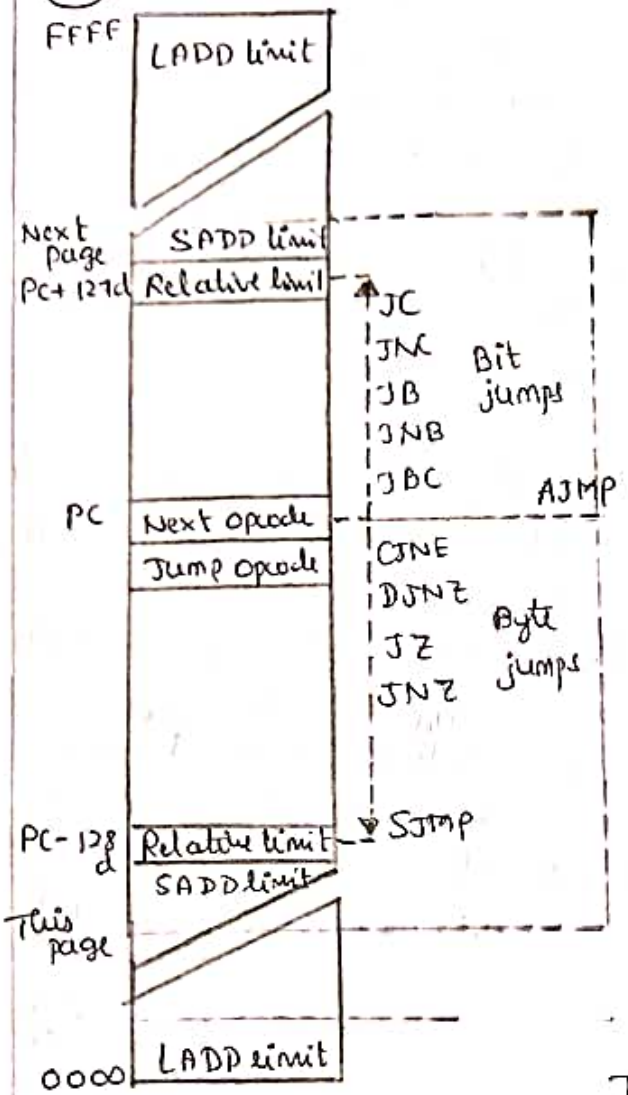
(iv) END:
The "End" directive signals the end of the assembly module. It indicates the end of the program to the assembler. Any text in the assembly file the appears after the END directive is ignored. If the END statement is missing the assembler will generate an error message.

(4b) Write an ALP to convert a Binary number to Packed BCD number (hexa to decimal). The binary number is stored at 40h location & store the converted packed BCD number at 50h & 51h internal RAM location. [6M]

```
org 0000h
mov A, #40h
ANL A, #0Fh
mov 50h, A
mov A, 40h
ANL A, #0F0h
SWAP A
mov 51h, A
end
```


R.C.M.

(4C) With neat diagram explain the range of jump instructions.



→ The difference of two address from the address in the program where jump or call is located is called the range of jump or call [6M]

Bit Jumps

Bit jumps all operates according to the status of the carry flag in the PSW or the status of any bit-addressable location. All bit jumps are relative to the program counter.

Mnemonic — Operation

JC radd → Jump relative if the carry flag is set to 1

JNC radd → Jump relative if the carry flag is reset to 0

JB b, radd → Jump relative if addressable bit is set to 1

JNB b, radd → Jump relative if addressable bit is reset to 0

JBC b, radd → Jump relative if addressable bit is set, and clear the addressable bit to 0

Byte Jumps : Byte jumps - jump instructions that test bytes of data - behave as bit jumps. If the condition that is tested is true, the jump is taken; if the condition is false, the instruction after the jump is executed. All byte jumps are relative to the program counter.

CJNE A, add, radd

CJNE A, #n, radd

CJNE Rs, #n, radd

CJNE @Rp, #n, radd

DJNZ Rs, radd

DJNZ add, radd

JZ radd

JNZ radd.

Module - 03

5a) Explain the bit contents of TCON and TMOD registers. [6M]

(i) TCON Register.

* Bit addressable as TCON.0 to TCON.7.

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

Bit	Symbol	function
7	TF1	* Timer-1 overflow flag * Set when rools from all 1's and 0.
6	TR1	* Timer-1 control bit * Set to 1 by program to enable timer to count
5	TF0	* Timer-0 overflow flag * Set when timer rolls from all 1's to 0
4	TR0	* Timer-0 control bit * Set to 1 by program to enable timer-to counter.
3	IE1	* External interrupt-1 edge flag * Set to 1 when a high to low edge signal is received on $\overline{INT1}$.
2	IT1	* External interrupt-1 signal type control bit * Set to 1 by program to enable external interrupt-1 to be triggered by a falling edge signal.
1	IE0	* External interrupt 0 edge flag * Set to 1 when a high to low edge signal is received on $\overline{INT0}$.
0	TI0	* External interrupt 0 signal type control bit.

(ii) TMOD Register.

7	6	5	4	3	2	1	0
Gate	C/$\overline{T}$	M1	M0	Gate	C/$\overline{T}$	M1	M0
Timer 1				Timer 0			

Prof. C. M.

Bit	Symbol	Function
7/3	Gate	Gating control when set. The timer / counter is enabled only while the INTx pin is high and the TRx control pin is set.
6/2	C/$\bar{T}$	Timer or counter selected cleared for timer operation. set for counter operation
5/1	M1	Timer/counter operating mode select bit 1
4/0	M0	Timer/counter operating mode select bit 0

M1	M0	mode	operating mode
0	0	mode-0	13-bit timer/counter
0	1	mode-1	16-bit timer/counter
1	0	mode-2	8-bit Auto reload T/C
1	1	mode-3	Split timer mode

(5b) Write an ALP to transfer multi-byte data serially with 9600 baud rate. [8M]

⟹

```
        org 0000h
        MOV TMOD,#20h
        MOV TH1,#-3
        MOV SCON,#50h
        SETB TR1
        MOV TR0,#04
        MOV R1,#100
LABEL:  MOV A,@R0
        INC R0
        ACALL TRANS
        DJNZ R1,LABEL
HERE:   END HERE
TRANS:  CLR TI
        RET
        END
```

(5c) Explain how timers are programmed in mode-1 [4M]

[4M]

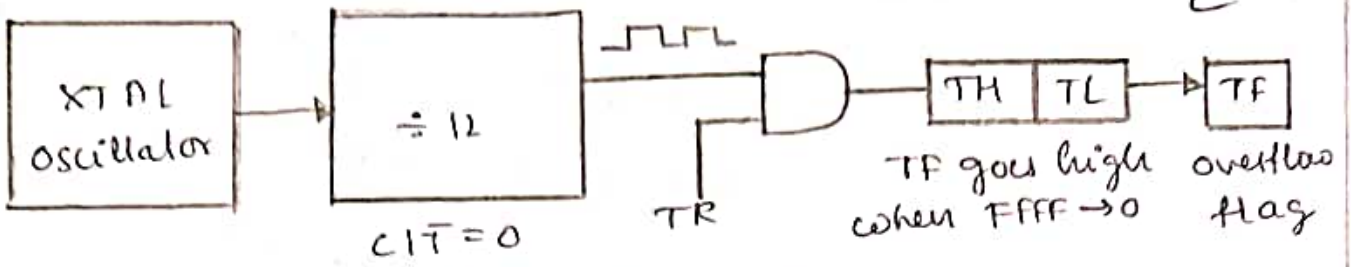


fig: mode-1 programming.

The following are the characteristics and operations of mode1

1. It is a 16-bit timer, therefore, it allows value of 0000h to FFFFh to be loaded into the timer's register TL and TH.
2. After TH and TL are loaded with a 16-bit initial value, the timer must be started. This is done by "setb TR0" for timer 0 and "setb TR1" for timer 1.
3. After the timer started it starts to count up. It counts up until it reaches its limit of FFFFh. when it rolls over the from FFFFh to 0000h. it sets high a flag bit called TF (timer flag). This timer flag can be monitored when this timer flag is raised, one option coould be to stop the timer with the instructions 'CLR TR0' or 'CLR TR1' for T0 and T1 respectively.
4. After the timer reaches its limit and rolls over, in order to repeat the process the registers TH and TL must be reloaded with the original value and TF must be reset to 0.

(6a) Explain the bit pattern of SCON register with diagram.

[6M]

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0 SCON.7 Serial port mode specifier

SM1 SCON.6 Serial port mode specifier

R.K.C.R.

SM2	SCON.5	used for multiprocessor communication. (Make it 0).
REN	SCON.4	Set/Cleared by software to enable/disable reception.
TB8	SCON.3	Not widely used
RB8	SCON.2	Not widely used.
TI	SCON.1	Transmit interrupt flag.
RI	SCON.0	Receive interrupt flag.

SM0 SM1 : These are D7 and D6 of the SCON register, respectively. They can take following combinations.

SM0	SM1	
0	0	Serial mode-0
0	1	Serial mode-1, 8-bit data, 1 stop bit, 1 start bit
1	0	Serial mode-2
1	1	Serial mode-3

(6b) Assuming XTAL frequency as 11.0592MHz. write a program in mode 1 to generate 4kHz square wave on P2.1 use timer 0 [8M]

$\longrightarrow T = \frac{1}{f} = \frac{1}{4k} = 0.25 \times 10^{-3}$ Sec.

$\frac{0.25 \times 10^{-3}}{2} = 0.125$ ms.

$0.125m = (65536 - n) 1/11.0592 \times 10^6 \div 12$

$n = 65420$

$n = FF8Ch$

```
org 0000h
mov TMOD, #01h
back: mov TH0, #0FFh
mov TL0, #08Ch
setb TR0
L1: JNB TF0, L1
CLR TR0
CLR TF0
CPL P2.1
SJMP back
RET
END.
```

(6C) Explain RS232 in serial communication using 8051 micro - controller. [4M]

⇒ # To allow compatibility among data communication equipment made by various manufacturers, an interfacing standard called RS232.

Handshaking signals.

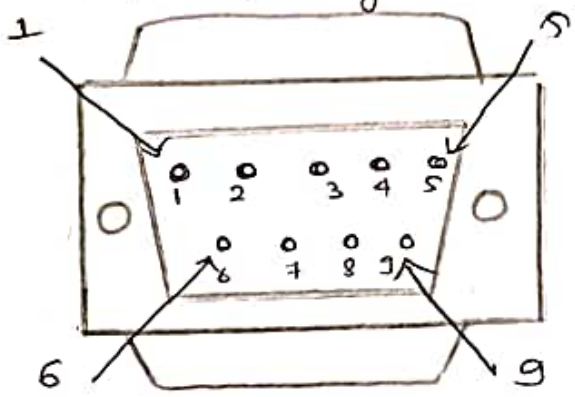
(1) DTR (Data terminal ready)

when a terminal is turned on, after going through a self-test, it sends out signal DTR to indicate that it a ready for communication. If there is something wrong with the com port, this signal not be activated.

It is an active low signal & can be used to inform the modem that computer is alive & ticking.

(2) DSR (data set ready).

when DCE (modem) is turned on and has gone through the self-test, it asserts DSR to indi-cate that it is ready to Communicate



(3) RTS (Request to send)

when the DTE device (such as a PC) has a byte to transmit, it asserts RTS to signal the modem that it has a byte of data to transmit. RTS is an active-low output from the DTE and an input to the modem.

(4) CTS (clear to send)

In response to RTS, when the modem has room for it to-ring the data it is to receive, it sends out signal. CTS to the DTE (PC) to indicate that it can receive the data now. This input signal to the DTE is used by the start transmission.

(5) DCD (Carrier detect, or DCD, data carrier detector).

The modem asserts signal DCD to inform the DTE (PC) that a valid carrier has been detected and that contact between it and the other modem is established. Therefore, DCD is output from the modem and an input to the PC (DTE)

Rohit.C.N.

⑥ RI (ring indicator).

An output from the modem (DCE) and an input to a PC (DTE) indicates that the telephone is ringing. It goes on and off in synchronization with the ringing sound.

Module - 04

⑦a Explain how interrupt priority can be changed using IP register. Also explain the default priorities assign to interrupts in 8051 microcontroller. [8M]

⟹ table: 8051 interrupt priority upon reset

Highest to lowest Priority	
External interrupt 0	INT0
Timer interrupt 0	TF0
External interrupt 1	INT1
Timer interrupt 1	TF1
Serial Communication	(RI + TI)

we can alter the squence of above table by assiguing a higher priority to anyone of the interrupts. This is done by programming a register called IP (Interrupt priority).

figure below shows the bits of the IP register. Upon power-up reset, the IP register contains all 0s, making the priority sequence based on above table. To give a higher priority to any of the interrupts, we make the corresponding bit in the IP register High

D7	D6	D5	D4	D3	D2	D1	D0
—	—	PT2	PS	PT1	PX1	PT0	PX0

priority bit = 1 assigns high priority . Priority bit = 0 assigns low priority

— IP.7 Reserved

— IP.6 Reserved.

PT2	IP.5	Timer 2 interrupt priority bit (8052 only)
PS	IP.4	Serial port interrupt priority bit
PT1	IP.3	Timer 1 interrupt priority bit
PX1	IP.2	External interrupt 1 priority bit
PT0	IP.1	Timer 0 interrupt priority bit
PX0	IP.0	External interrupt 0 priority bit

(16) Explain how programming of external hardware interrupts is done in 8051 Microcontroller with a code [8M]

⇒ # There are only two external hardware interrupts in the 8051 : INT0 and INT1. They are located on pins P3.2 and P3.3 of port 3, respectively. The interrupt vector table locations 0003h and 0013h are set aside for INT0 and INT1 respectively.

They are enable and disabled using the IE register.

There are two types of activation for the external hardware interrupts : (1) level triggered.
(2) Edge triggered.

Example : Generate a square wave on all pins of P0 which is half the frequency of signal applied at INT0 pin (P3.2)

```
        org 0000h
        Ljmp main
        org 0003h
        CPL P0
        RETI
        org 0080h
main:   Setb TCON.0
        mov IE, #81h ; Enable hardware interrupt INT0
HERE:   SJMP HERE
        END
```

Ref. C.M

(7C) Explain programming of timer interrupts. [4M]

→ Roll-over timer flag and interrupt:

In polling TF, we have to wait until the TF is raised. The problem with this method is that the microcontroller is tied down while waiting for TF to be raised and cannot do anything else. Using interrupts solves this problem and avoids tying down the microcontroller.

If the timer interrupt in the IE register is enable -d, whenever the timer roll over, TF is raised, and the microcontroller is interrupted in whatever it is doing, and jumps to the interrupt vector table to ser -vice the ISR. In this way, the microcontroller can do other things until it is notified that the timer has rolled over.

(8a) Explain the bit contents of IE register. [4M]

D7							D0
EA	-	ET2	ES	ET1	EX1	ET0	EX0

EA IE.7 Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clear -ing its enable bit.

\- IE.6 Not implemented, reserved for future use *

ET2 IE.5 Enables or disables Timer 2 overflow or capture interrupt

ES IE.4 Enables or disables the serial port interrupt.

ET1 IE.3 Enables or disable Timer 1 overflow interrupt.

EX1 IE.2 Enables or disables external interrupt 1.

ET0 IE.1 Enables or disables Timer 0 overflow interrupt.

EX0 IE.0 Enables or disables external interrupt 0.

(8b) Write a C-program using interrupts to generate a square wave on port pin P2.2 of 1KHz using timer-0 in mode-02

⟹
```
#include<reg51.h>
sbit mybit = P2^2;
void timer0() interrupt
{ unsigned char i;
while (1)
{ for (i=0; i<10; i++)
{ mybit = ~mybit;
 }
 }
 }
void main ( )
{ TMOD = 0X02;
  TH0 = 0XD2h;
  IE = 0X82;
  TR0 = 1;
 }
```

$f = 1\,KHz$, $T = \frac{1}{1k} = 1ms$

$\frac{1ms}{2} = 0.5ms$ [8M]

$\frac{0.5\,ms}{1.085\,\mu s} = (256 - x)$

$\therefore x = D2\,h$

(8c) Explain programming of serial communication interrupts.

⟹ RI and TI flags and interrupts.

TI (transfer interrupt) is raised when the last bit of the framed data, the stop bit, is transferred, indicating that the SBUF register is ready to transfer the next byte.

RI (Received interrupt), is raised when the entire frame of the data, including the stop bit, is received.

In the polling method, we wait for the flag (TI or RI) to be raised; while we wait, we cannot do anything else.

In the interrupt method, we all notified when the 8051 has received a byte or is ready to send the next byte; we can do other things while the serial communication needs are served.

Rahul. C. N

In 8051 only one interrupt is set aside for serial communication. This interrupt is used to both send and receive data.

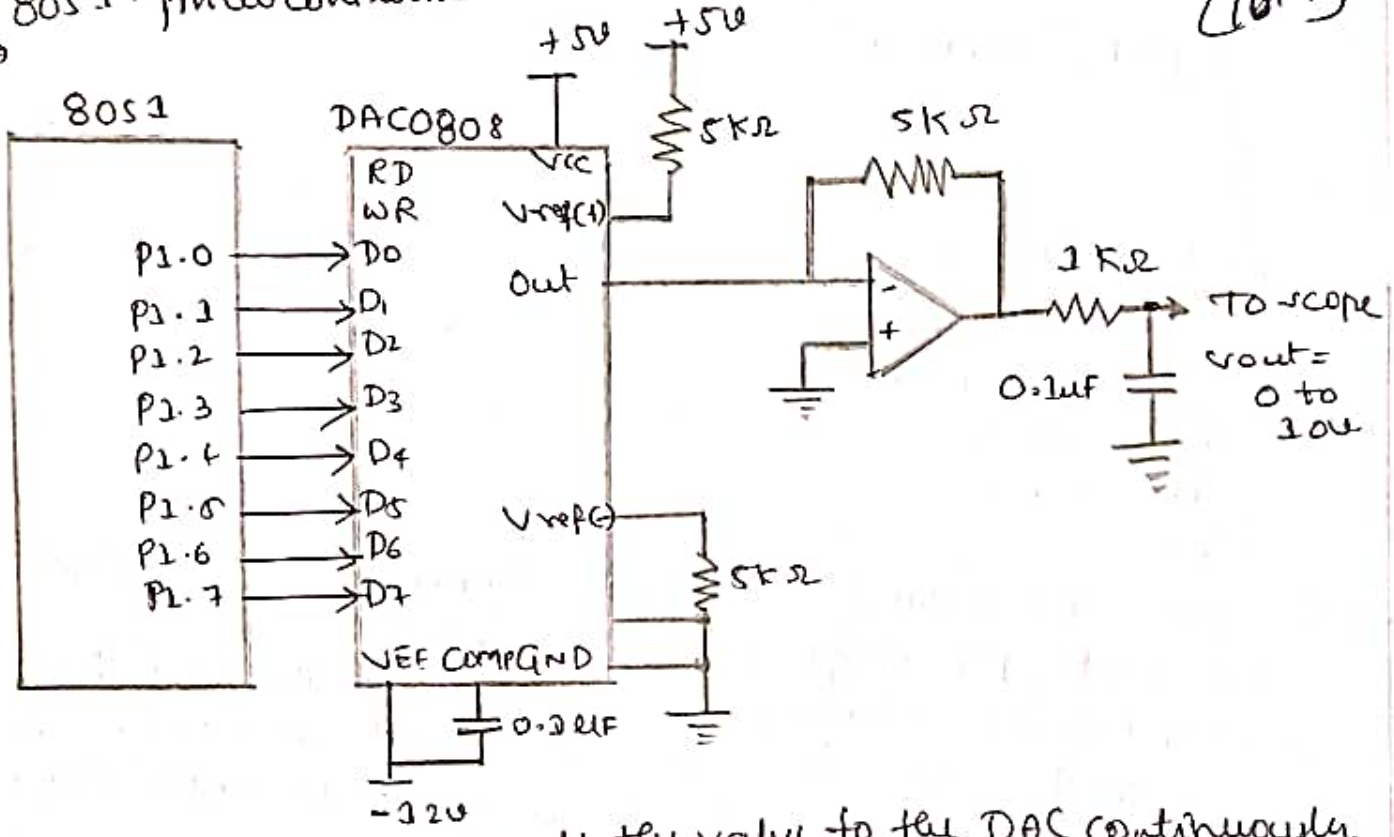
If the interrupt bit in the IE register (IE.4) enabled when RI or TI is raised the 8051 gets interrupted and jumps to memory address location 0023h to execute the ISR.

* Clearing RI and TI before the RETI instruction

Module - 05

(9a) With neat diagram write an ALP to interface DAC to 8051 microcontroller [8M]

⟹



The below program sends the value to the DAC continuously (in an infinite loop) to produce saw tooth wave.

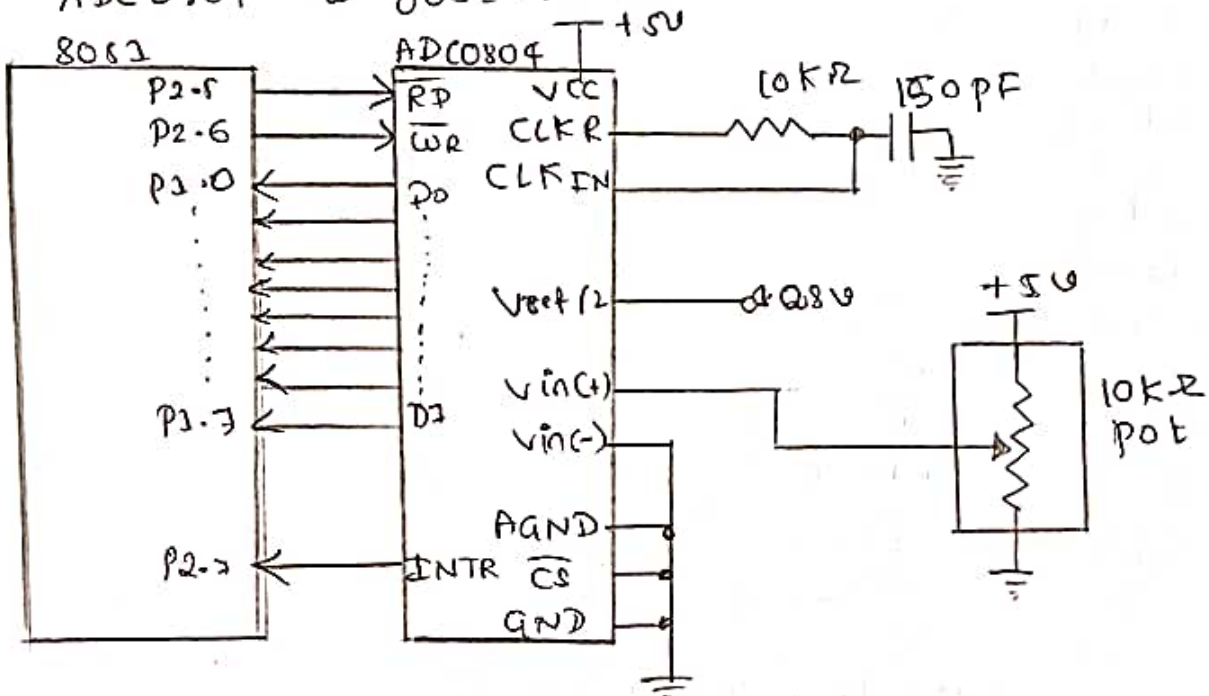
```
        ORG 0000h
again:  MOV DPTR, #TABLE
        MOV R2, #COUNT
Back:   CLR A
        MOVC A, @A+DPTR
        MOV P1, A
        INC DPTR
        DJNZ R2, Back
        SJMP again

        ORG 300
TABLE:  DB 128, 192, 238, 255, 238,
           192
        DB 128, 64, 17, 0, 17, 64,
           128
```

(9b) Write a program to display "HELLO WORLD" by interfacing LCD display to 8051 microcontroller. [10M]

```
#include<reg51.h>
sfr ldata = 0x90;
sbit rs = P2^0;
sbit rw = P2^1;
sbit en = P2^2;
void lcdcmd (unsigned char);
void lcddata (unsigned char);
void msdisplay (unsigned char);
void main()
{
 unsigned char i;
 unsigned char arr[] = "HELLO WORLD";
  lcdcmd = 0x38;
  msdelay (250);
  lcdcmd = 0x0E;
  msdelay (250);
  lcdcmd = 0x01;
  msdelay (250);
  lcdcmd = 0x06;
  msdelay (250);
  lcdcmd = 0x86;
  msdelay (250);
for (i=0; i<11; i++)
{ lcddata (arr[i]);
 }
 msdelay (250);
 void lcdcmd (unsigned char value)
 { ldata = value;
    rw = 0;
    rs = 0;
    en = 1;
   msdelay ms display(1);
   en=0;
   return;
   }
void lcddata (unsigned char value)
 { ldata = value;
    rw = 1;
    rs = 0;
```

```
    en=1;
    msdisplay(1);
    en=0;
    return;
}
void msdisplay(unsigned char item)
{unsigned value x, y;
for(x=0; x<item*10; x++)
  for(y=0; y<1275; y++);
  }
```

(10a) with a neat diagram write an ALP to interface ADC0804 to 8051 uC. [10M]



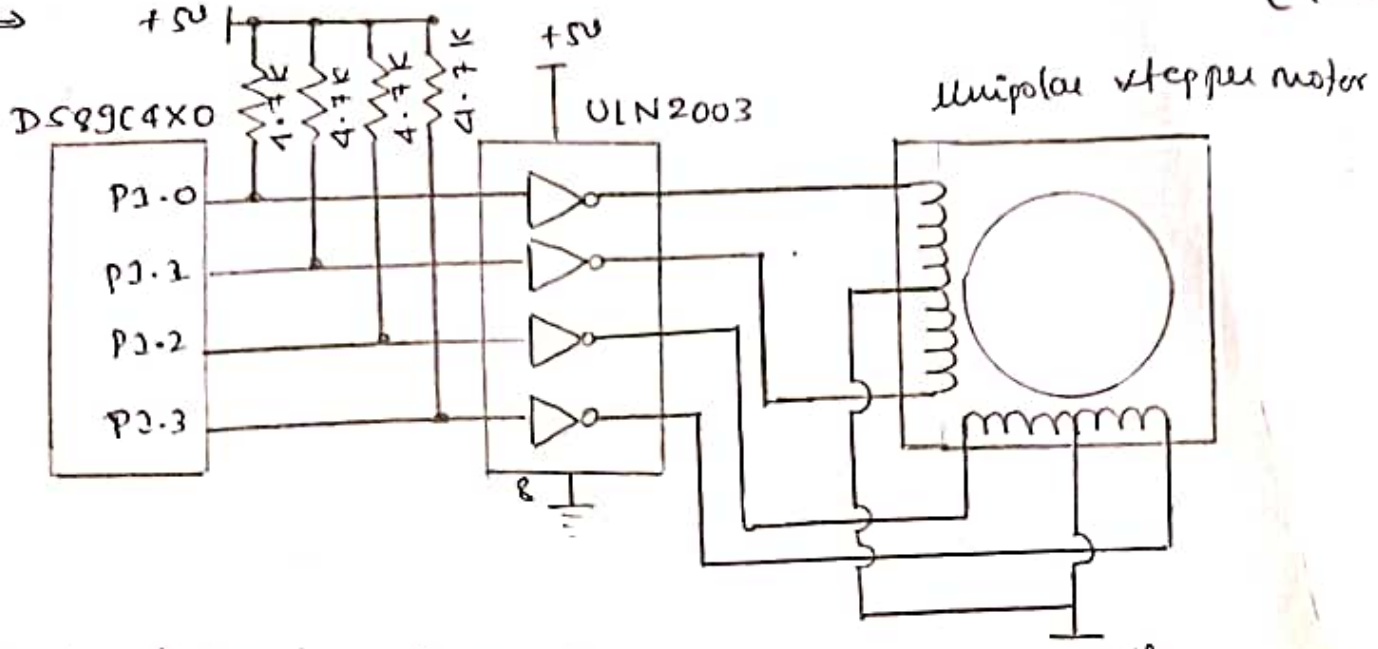
*The following program monitors the INTR pin and brings an analog input into register A. It then calls hex-to-ASCII conversion and data display subroutines.

```
RD      BIT  P2.5
WR      BIT  P2.6
INTR    BIT  P2.7
MYDATA  EQU  P1
MOV     P1, #0FFH
Setb    INTR
Back:CLR    WR
     Setb   WR
HERE:JB     INTR, HERE
```

```
CLR   RD
MOV   A, MYDATA
ACALL CONVERSION
ACALL DATA_DISPLAY
Setb  RD
JUMP  Back.
```

(10b) With neat diagram explain the interfacing of stepper motor using 8051 microcontroller. [10M]

⟹



The following steps show the 8051 connection to the step-per motor and its programming.

(1) Use an ohmmeter to measure the resistance of the leads. This should identify which COM leads are connected to which winding leads

(2) The common wire(s) are connected to the positive side the motor power supply. In many motors +5V is sufficient.

(3) The four leads of the stator winding are controlled by four bits of the 8051 port (P1.0 - P1.3). However, since the 8051 lacks sufficient current to drive the stepper motor windings, we must use a driver such as the ULN2003 to energize the stator. One reason that using ULN2003 is preferable to use of transistors as drivers is that ULN2003 has an internal diode to take care of Back EMF.

```
        mov A, #66h
back:   mov P1, A
        RR  A
        ACALL Delay
        SJMP back
        ....

Delay   mov R2, #100
H1:     mov R3, #255
H2:     DJNZ R3, H2
        DJNZ R2, H1
        RET
```