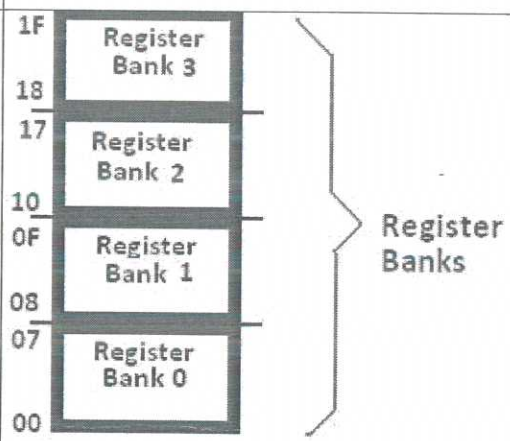
Scoring Indicators

COURSENAME: MICROCONTROLLER AND APPLICATIONS

COURSECODE: 4041 QID: 2103230126

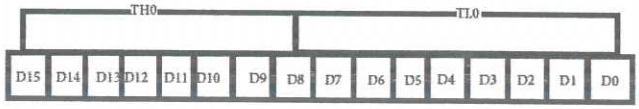
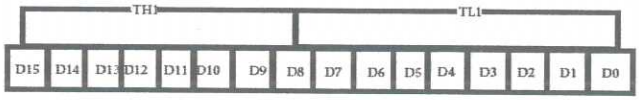
Q. No	Scoring Indicators	Split score	Sub Total	Total score
	PARTA			**9**
I.1	RAM – 128 bytes ROM – 4K	2*0.5	1	
I.2	Program counter (PC) and data pointer (DPTR)	2*0.5	1	
I.3	Register indirect addressing mode	1	1	
I.4	The CPU takes a certain number of clock cycles to execute an instruction. These clock cycles are referred to as machine cycles.	1	1	
I.5	RETI	1	1	
I.6	TCON register	1	1	
I.7	The baud rate is the rate at which information is transferred in a communication channel. If the information unit is one baud (one bit), then the bit rate and the baud rate are identical.	1	1	
I.8	1. The ability to display numbers, characters and graphics. 2. Declining prices. 3. Ease of programming for characters and graphics. 4. Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD.	2*0.5	1	
I.9	Output voltage	1	1	

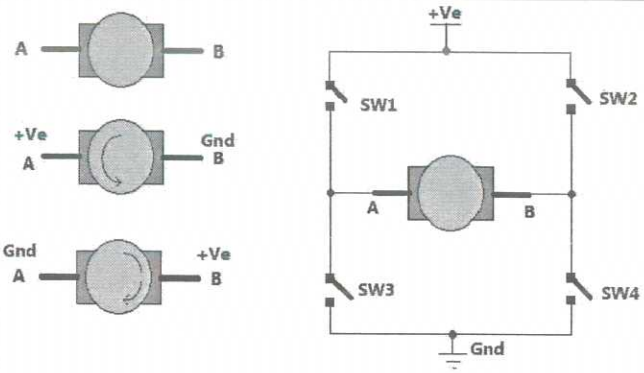
	PARTB			24
II.1	• 8-bit CPU with registers A and B • 16-bit program counter (PC) and data pointer (DPTR) • 8-bit program status word (PSW) • 8-bitstack pointer (SP) • 4 K internal ROM • 128 bytes internal RAM • 32 I/O pins arranged as four 8-bit ports • Two 16-bit timer/counters: T0 and T1 • Full duplex serial data receiver/transmitter: SBUF • Control registers: TCON, TMOD, SCON, PCON, IP, IE • 2 external and 3 internal interrupt sources • Oscillator and clock circuits	6*0.5	3	
II.2	 The 8051 microcontroller consists of four register banks, such as Bank0, Bank1, Bank2, Bank3 which are selected by the PSW (Program Status Word) register. These register banks are present in the internal RAM memory of the 8051 microcontroller, and are used to process the data when the microcontroller is programmed.	Fig(1.5) + exp(1.5)	3	

II.3	PC and DPTR are 16bit registers. Each is used to hold the address of a byte in memory. Program instruction bytes are fetched from locations in memory that are addressed by the PC. PC is automatically incremented after every instruction byte is fetched and may also be altered by certain instructions. DPTR register is made up of two 8-bit registers DPH and DPL, which is used to furnish memory addresses for internal and external code access and external data access.	1.5+1.5	3	

II.4	MOV A, #0FFH
	MOV P2, A
	BACK: MOV A, P2
	MOV P1, A
	SJMP BACK
		3	3	

II.5	MOV: MOV is an internal data transfer instruction. It is the data transfer between registers and between registers and general storage area. MOVX: MOVX is a data transfer instruction between external data memory (external RAM) and accumulator A. Because the internal and external RAM addresses overlap, different commands are needed to distinguish them. MOVC: MOVC is a data transfer instruction between the accumulator and the program storage area. It has a letter "C" more than the MOV instruction. This "C" means "Code", which means "code", which is the data transfer instruction between the code area (program storage area) and A. It can be used for data transfer between the internal program storage area (internal ROM) and A, and can also be used for data transmission between the external program storage area (external ROM) and A. Because the program memory area and outside are addressed uniformly, one instruction is enough.	3	3	

II.6	<table><tr><th>Interrupts</th><th>Memory Location</th><th>Pin</th><th>Flag Clearing</th></tr><tr><td>Reset</td><td>0000</td><td>9</td><td>Auto</td></tr><tr><td>Timer0</td><td>000B</td><td></td><td>Auto</td></tr><tr><td>Timer1</td><td>001B</td><td></td><td>Auto</td></tr><tr><td>INT0</td><td>0003</td><td>12</td><td>Auto</td></tr><tr><td>INT1</td><td>0013</td><td>13</td><td>Auto</td></tr><tr><td>Serial com</td><td>0023</td><td></td><td>Cleared by programmer</td></tr></table>	3	3	
II.7	 Fig. Timer0  Fig. Timer1	1.5+ 1.5	3	
II.8	In 8051, the oscillator output is divided by 12 using a divide by 12 network and then fed to the Timer as the clock signal. That means for an 8051 running at 12MHz, the timer clock input will be 1MHz. That means the timer advances once in every 1uS and the maximum time delay possible using a single 8051 timer is (2^16) x (1μS) = 65536μS. Delays longer than this can be implemented by writing up a basic delay program using timer and then looping it for a required number of time.	3	3	
II.9	**SBUF (Serial buffer register)** It is an 8-bit register and is used for serial communication of data in 8051 microcontroller. Whatever data is required to be transmitted via TXD line must be placed in the SBUF register. Similarly the received data via RXD line is saved in SBUF register. When data is written to SBUF register then it is framed	3	3	

	in b/w start and stop bit before it is transmitted via TXD line and similarly during reception of data start and stop bits are removed and actual data bits are extracted from the received frame and then it is placed in the SBUF register.			
II.10	 The circuit connection used to control or change the direction or rotation of the DC motor is called H – bridge. The input supply Vcc and the ground are made available to both the pins of the motor through four SPST(Single Pole Single Throw) switch. If switches SW1 and SW4 are closed and SW2 and SW3 are open, current from Vcc will travel through the motor from left to right and results in the clockwise motion of the motor. If SW2 and SW3 are closed and SW1 and SW4 are open, the previous input polarity gets interchanged making the current to flow through motor from right to left resulting the motor to rotate in anticlockwise direction.	Fig(1)+ exp(2)	3	
	PART C			**42**
III	Program Memory (ROM) Program Memory (ROM) is used for permanent saving program (CODE) being executed. The memory is read only. Depending on the settings made in compiler, program memory may also used to store a constant variable. The 8051 executes programs stored in program memory only. Code memory type specifier is used to refer to program memory.	Fig(2)+ Exp(5)	7	

8051 memory organization allows external program memory to be added. How does the microcontroller handle external memory depend on the pin EA logical state.

1. Internal Program Memory (4KB) i.e. from 0000H to 0FFFH + External Program Memory (60KB) i.e. from 1000H to FFFFH. This mode is selected by making EA = 1.

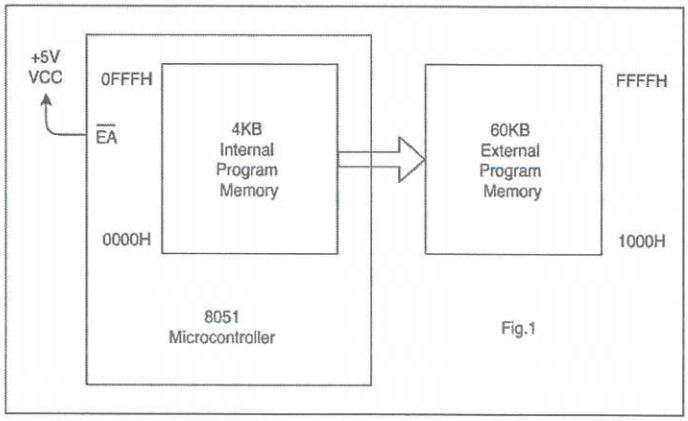


Fig.1

2. Total External Program Memory (64KB) i.e., over the entire range of 0000H to FFFFH. This mode is selected by making EA = 0.

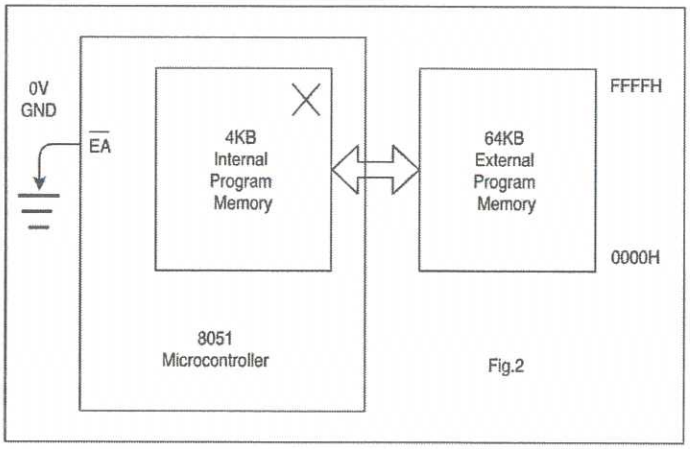
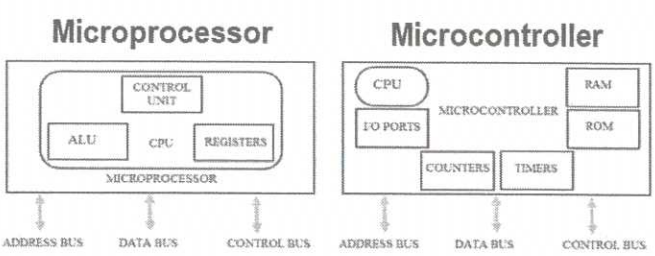


Fig.2

<table>
<tr>
<td>IV</td>
<td>


Microprocessor vs Microcontroller by EEEPROJECT.COM
<table>
<tr><th>Microprocessor</th><th>Microcontroller</th></tr>
<tr><td>a. Microprocessors are widely used in computer systems.</td><td>a. Microcontroller is widely used in embedded systems.</td></tr>
<tr><td>b. It has only a CPU embedded into it.</td><td>b. It has a CPU, a fixed amount of RAM, ROM and other peripherals all embedded on it.</td></tr>
<tr><td>c. In case of microprocessors we have to connect all the components externally so the circuit becomes large and complex.</td><td>c. As all the components are internally connected in microcontroller so the circuit size is small.</td></tr>
<tr><td>d. It consumes more power.</td><td>d. It consumes less power than a microprocessor.</td></tr>
<tr><td>e. It has very less internal register storage so it has to rely on external storage. So all memory operations are carried out using memory based external commands which results in high processing time.</td><td>e. It has many registers so processing time is less.</td></tr>
</table>
</td>
<td>Fig(2)+ Exp(5)</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>The ability to rotate accumulator register data is useful to allow examination of individual bits of a byte. Register A can be rotated one bit position to left or right with or without including carry flag in the rotation.
RL A – Rotate Accumulator left
RLC A - Rotate Accumulator left through the carry
RR A – Rotate Accumulator right
RRC A - Rotate Accumulator right through the carry</td>
<td>7</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>VI</td>
<td>There are two kinds of jump instructions:
1. Unconditional jump instructions: upon their execution a jump to a new location from where the program continues execution is executed.
a. LJMP(long jump) - 3-byte instruction, The first byte is the opcode, Second and third</td>
<td>3.5+ 3.5</td>
<td>7</td>
<td></td>
</tr>
</table>

bytes represent the 16-bit target address, Any memory location from 0000 to FFFFH

b. SJMP (short jump) - 2-byte instruction, The first byte is the opcode, The second byte is the relative target address– 00 to FFH(Forward +127 and backward -128 bytes from the current PC)

2. **Conditional jump instructions:** a jump to a new program location is executed only if a specified condition is met. Otherwise, the program normally proceeds with the next instruction. All conditional jumps are short jumps: The address of the target must within -128 to +127 bytes of the contents of PC .

8051 Conditional Jump Instructions

Instruction	Action
JZ (Jump Zero)	Jump if A=0
JNZ (Jump no zero)	Jump if A≠0
DJNZ Rn,target	Decrement and jump if byte≠0
CJNE A,byte,target	Compare A with byte and jump if not equal (A≠byte)
CJNE reg,#data,target	Compare reg. with #data and jump if not equal (byte ≠ #data)
JC (Jump carry)	Jump if CY=1
JNC (Jump no carry)	Jump if CY=0
JB (Jump bit)	Jump if bit=1
JNB (Jump no bit)	Jump if bit=0
JBC (jump bit clear bit)	Jump if bit=1 and clear bit

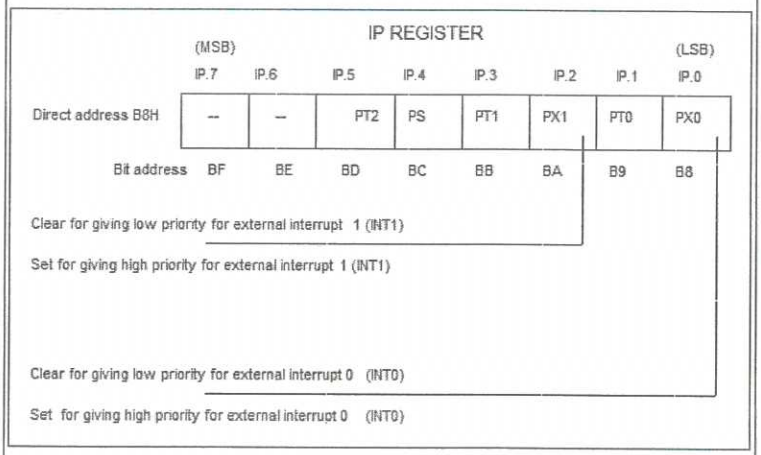
VII	MOV DPTR, #8500H
	MOVX A, @DPTR
	MOV B, #64H
	DIV AB
	INC DPTR
	MOVX @DPTR, A
	MOV A, B
	MOV B, #0AH
	DIV AB
	INC DPTR
	MOVX @DPTR, A
	MOV A, B
	INC DPTR
	MOVX @DPTR, A
	HERE: SJMP HERE
		7	7	

VIII	ORG 0000H
	CLR C
	MOV A, #0ACH
	MOV B, #89H
	ADD A, B
	MOV R0, A
	MOV A, #13H
	MOV B, #43H
	ADDC A,B
	MOV R1, A
	END
		7	7	

IX		**Priority**	**Interrupt source**	**Intr. bit / flag**	3.5+ 3.5	7	
		1	External Interrupt 0	INT0			
		2	Timer Interrupt 0	TF0			
		3	External Interrupt 1	INT1			
		4	Timer Interrupt 1	TF1			
		5	Serial communication interrupt	(TI/RI)			
		6	Timer 2(8052 only)	TF2			

When the 8051 is powered up the priorities assigned according to above table. The priority scheme is an internal polling sequence in which the 8051 polls the interrupts in the sequence listed in table and responds accordingly.

Priority to the interrupt can be assigned by using the interrupt priority register (IP). Upon power-up reset, the IP register contains all 0s. To give a higher priority to any of the interrupts we make the corresponding bit in the IP register high.



PT2-Timer2 interrupt priority bit (8052 only)

	PS-serial port interrupt priority bit PT1- Timer1 interrupt priority bit PX1-external interrupt1 priority bit PT0- Timer0 interrupt priority bit PX0-external interrupt0 priority bit			
X	Mode 0 (13-Bit Timer Mode) Both Timer 1 and Timer 0 in Mode 0 operate as 8-bit counters (with a divide-by-32 prescaler). Timer register is configured as a 13-bit register consisting of all the 8 bits of TH1 and the lower 5 bits of TL1. The upper 3 bits of TL1 are indeterminate and should be ignored. Setting the run flag (TR1) does not clear the register. The timer interrupt flag TF1 is set when the count rolls over from all 1s to all 0s. Mode 0 operation is the same for Timer 0 as it is for Timer 1. Mode 1 (16-Bit Timer Mode) Timer mode "1" is a 16-bit timer and is a commonly used mode. It functions in the same way as 13-bit mode except that all 16 bits are used. TLx is incremented starting from 0 to a maximum 255. Once the value 255 is reached, TLx resets to 0 and then THx is incremented by 1. As being a full 16-bit timer, the timer may contain up to 65536 distinct values and it will overflow back to 0 after 65,536 machine cycles. Mode 2 (8 Bit Auto Reload) Both the timer registers are configured as 8-bit counters (TL1 and TL0) with automatic reload. Overflow from TL1 (TL0) sets TF1 (TF0) and also reloads TL1 (TL0) with the contents of Th1 (TH0), which is preset by software. The reload leaves	7	7	

TH1 (TH0) unchanged.

The benefit of auto-reload mode is that you can have the timer to always contain a value from 200 to 255. If you use mode 0 or 1, you would have to check in the code to see the overflow and, in that case, reset the timer to 200. In this case, precious instructions check the value and/or get reloaded. In mode 2, the microcontroller takes care of this. Once you have configured a timer in mode 2, you don't have to worry about checking to see if the timer has overflowed, nor do you have to worry about resetting the value because the microcontroller hardware will do it all for you. The auto-reload mode is used for establishing a common baud rate.

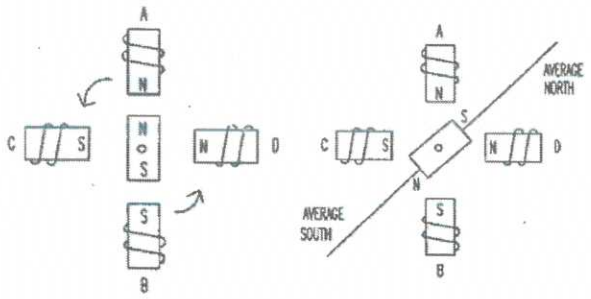
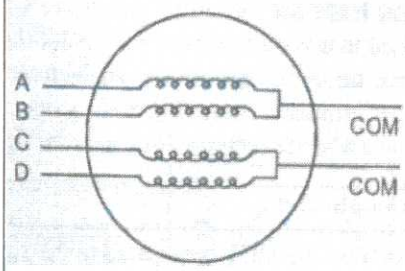
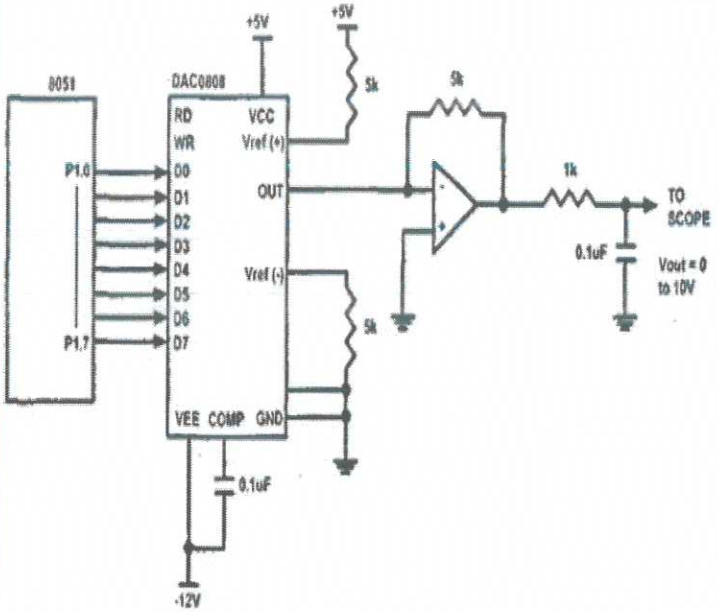
Mode 3 (Split Timer Mode)

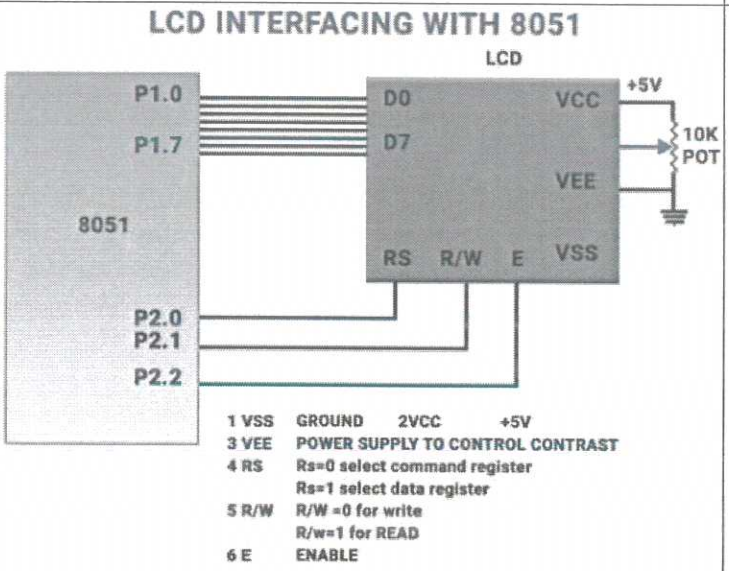
Timer mode "3" is known as **split-timer mode**. When Timer 0 is placed in mode 3, it becomes two separate 8-bit timers. Timer 0 is TL0 and Timer 1 is TH0. Both the timers count from 0 to 255 and in case of overflow, reset back to 0. All the bits that are of Timer 1 will now be tied to TH0.

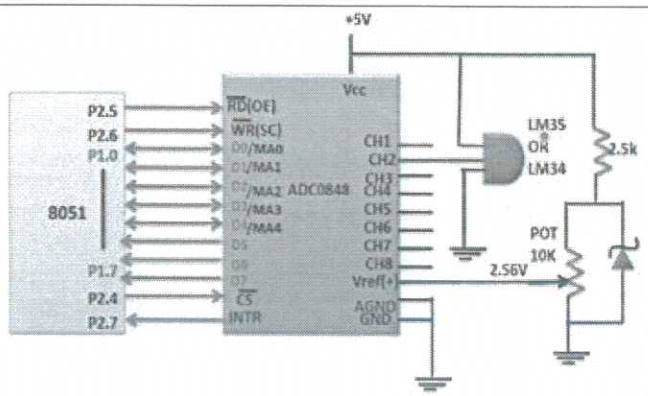
When Timer 0 is in split mode, the real Timer 1 (i.e. TH1 and TL1) can be set in modes 0, 1 or 2, but it cannot be started/stopped as the bits that do that are now linked to TH0. The real timer 1 will be incremented with every machine cycle.

XI	A stepper motor is a widely used device that translates electrical pulses into mechanical movement. They have a permanent magnet rotor (also called the shaft) surrounded by stator. The most common stepper motor has four stator windings that are paired with a center tapped common as shown. This type is referred to as four-phase or unipolar stepper motor. The center tap allows a change or current direction in each of the two coils when a winding is grounded, thereby resulting in polarity change of the stator. The stepper motor shaft runs freely, the stepper motor shaft moves in a fixed repeatable increment, which allows one to move it to a precise position. This repeatable fixed movement is possible as a result of basic magnetic theory where poles of the same polarity repel and opposite poles attract. The direction of rotation is dictated by stator poles. The stator poles are determined by the current sent through the wire coils. As the direction of the current is changed, the polarity is also changed causing reverse rotation of the rotor. Clockwise  Counter clockwise 	Fig(3)+ exp(4)	7	

step	A	B	C	D
1	1	0	0	1
2	1	1	0	0
3	0	1	1	0
4	0	0	1	1

	Stepper Motor Diagram Rotor Alignment   			
XII	 The digital data to be converted to analog is transferred from microcontroller to the DAC through port1, i.e., P1.0 is	Fig(4)+ exp(3)	7	

	connected to D0, P1.1 to D1 and so on. Pin 16(COMP) needs a 0.1 uF disc capacitor between this pin and pin 3. +5V is given to Vcc and -12V is given to VEE. +Vref and –Vref are connected as shown in the diagram. At the output stage an opamp is used. Ideally is we connect an output pin to a resistor, it convert this current to voltage. But practically this creates inaccuracy. Hence the output current is isolated by connecting it to an opamp in inverting mode by 5K feedback resistor. Using this DAC we can generate various waveforms like sine, triangle, ramp and square waves. DAC may also be used to drive DC motors and other electrical devices.			
XIII	 The PORT 1(8 bits, P1.0 to P1.7) of the microcontroller is connected to the data pins of LCD (D0 to D7), it is used for transferring data/commands to the LCD. The P2.0 pin is connected to the RS(register select) pin of LCD and RS pin is the input pin used to select command code register(RS=0) or data register (RS=1).	Fig(4)+ exp(3)	7	

	The P2.1 pin is connected to the R/W(input) pin of LCD and it helps to write information to the LCD(R/W=0) or read information(R/W=1). The P2.2 pin of PORT2 is connected to the enable pin of LCD, which is used by theLCD to latch information presented to its data pins D0 to D7.			
XIV	 The LM35 is a temperature sensor whose output voltage is linearly proportional to Celsius temperature. The LM35 comes already calibrated hence requires no external calibration. It outputs 10mV for each degree of Celsius temperature. LM35 sensor produces voltage corresponding to temperature. This voltage is converted to digital (0 to 256) by ADC0848 and it is fed to 8051 microcontroller. 8051 microcontroller converts this digital value into temperature in degree Celsius. Then this temperature is converted into ascii form which is suitable for displaying.	Fig(4)+ exp(3)	7	