Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(**ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka

Department of Electrical & Electronics Engineering

LAB MANUAL

(2022-2023)

MICROCONTROLLER – 21EE43

(IPCC Course)

B.E. - IV Semester

Name: ____

USN: ____

Batch:________________________________Section: ______________

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(**ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka

Department of Electrical & Electronics Engineering

MICROCONTROLLER – 21EE43

B.E. - IV Semester

Version 3.0

June 2023

Prepared by:

Mrs. Radha B.N.
Assistant Professor
Dept. of EEE

Approved by:

V.C. Kumar
Professor & Head
Dept. of EEE

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(ISO 9001:2015 Certified Institution)
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

VISION

To create centers of excellence in education and to serve the society by enhancing the quality of life through value based professional leadership.

MISSION

- To provide high quality technical and professionally relevant education in a diverse learningenvironment.
- To provide the values that prepare students to lead their lives with personal integrity, professional ethics and civic responsibility in a global society.
- To prepare the next generation of skilled professionals to successfully compete in the diverseglobal market.
- To promote a campus environment that welcomes and honors women and men of all races, creeds and cultures, values and intellectual curiosity, pursuit of knowledge and academic integrity and freedom.
- To offer a wide variety of off-campus education and training programmes to individuals andgroups.
- To stimulate collaborative efforts with industry, universities, government and professional societies.
- To facilitate public understanding of technical issues and achieve excellence in the operations of the institute.

EEE- DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Vision

To Reach excellence in Electrical and Electronics Engineering education and to facilitate technically competent professionals in Electrical Sciences and allied fields with ethics to servethe society.

Mission

M1	To provide quality education to meet the modern needs in electrical & electronics engineering and allied fields.
M2	To empower each individual to apply knowledge and skills for the betterment of the society.
M3	To create centre of excellence in electrical sciences through industry-institute interactions and by adopting modern technology.
M4	To motivate research activities among students and faculty to meet the evolving needs ofthe society.

Program Educational Objectives(PEOs)	
PEO1	To exhibit strong knowledge in electrical sciences, mathematics and to analyze, apply, design and develop products of real time applications.
PEO2	To utilize technical knowledge, effective communication, leadership qualities and engaging with lifelong learning for the progress of Society.
PEO3	To facilitate a holistic academic environment and multidisciplinary approach for pursuing higher studies and to innovate through continuous research.

Program Specific Objectives(PSOs)	
PSO1	Analyse and apply principles of electrical science, mathematics and various techniques to evaluatedifferent circuits and to assess the performance of machines, transmission and distribution, protection mechanisms in power system.
PSO2	Design and development of electrical and electronics circuits, measuring instrumentsand their Testing, control systems and strategies for power electronics, digital electronics circuits andapplication of microcontrollers.
PSO3	Able to effectively communicate and work in a team with ethical attitude and to apply holisticknowledge in Design,development and demonstration of project.

Engineering Graduates will be able to (Program Outcomes)	
PO1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and engg. specialization to the solution of complex engineering problems.
PO2	**Problem analysis**: Identify, formulate, research literature, and analyze engineering problems to arrive at substantiated conclusions using first principles of mathematics, natural, and engineering sciences
PO3	**Design/development of solutions**: Design solutions for complex engineering problems and design system components, processes to meet the specifications with consideration for the public health and safety, and the cultural, societal, and environmental considerations
PO4	**Conduct investigations of complex problems**: Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid Conclusions
PO5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding ofthe limitations
PO6	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professionalengineering practice
PO7	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable Development
PO8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice
PO9	**Individual and team work**: Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings
PO10	**Communication**: Communicate effectively with the engineering community and with society at large. Be able to comprehend and write effective reports documentation. Make effective presentations, and give andreceive clear instructions
PO11	**Project management and finance**: Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments
PO12	PO 12. **Life-long learning Program**: Recognize the need for, and have the preparation and ability toengage in independent and life-long learning in the broadest context of technological change

Instruction to Candidates:

1. Come in formal dress code to Lab always.

2. Soon after entering the lab, enter “USN, Name, Time-in” in the Movement register.

3. Come prepared to the lab with relevant theory and logic about the programs to be executed

4. Before going out of the lab, Shut down the systems, Keep mouse, keyboard and chairs properly and enter “ time-out and put the signature” in movement register.

INDEX PAGE

MICROCONTROLLER – 21EE43(IPCC Course)

Sl. No.	Name of the Experiment	Date			Observation Marks (Max . 20)	Record Marks (Max. 10)	Signature (Student)	Signature (Faculty)
		Conduction	Repetition	Submission of Record				
01								
02								
03								
04								
05								
06								
07								
08								
09								
10								
11								
12								
Average								

Note: **If the student fails to attend the regular lab, the experiment has to becompleted in the same week. Otherwise manual/observation and record will be evaluated for 50% of maximum marks**

Microcontroller (21EE43) IPCC Lab Syllabus

Semester :	**IV**	**CIE : 50 Marks (30 theory + 20 Lab marks)**
IPCC Course Code :	**21EE43**	**SEE : 50 Marks**
Teaching hours and Practical Hours/week (L:T:P)	**3:0:2**	**Lab marks: 20** **Record +Observation: 15 marks** **Test : 05 marks**
Credits : 04		**Test Hours : 03**

Course Objectives:

- To explain writing assembly language programs for data transfer, arithmetic, Boolean and logical instructions.
- To explain writing assembly language programs for code conversions.
- To explain writing assembly language programs using subroutines for generation of delays, counters, configuration of SFRs for serial communication and timers.
- To perform interfacing of stepper motor and dc motor for controlling the speed.
- To explain generation of different waveforms using DAC interface

Course outcomes:

At the end of the IPCC course, the student will be able to:

- Write assembly language programs for data transfer, arithmetic, Boolean and logical instructions.
- Write ALP for code conversions.
- Write ALP using subroutines for generation of delays, counters, configuration of SFRs for serial communication and timers.
- Perform interfacing of stepper motor and dc motor for controlling the speed and Generate different waveforms using DAC interface.

Syllabus

Sl. No.		Experiments
1	Assembly programs	Data transfer – Program for block data movement, sorting, exchanging, finding largest element in an array
2		Arithmetic instructions: Addition, subtraction, multiplication and division. Square and cube operations for 16 bit numbers.
3		Counters
4		Boolean and logical instructions (bit manipulation).
5		Conditional call and return instructions.
6		Code conversion programs – BCD to ASCII, ASCII to BCD, ASCII to decimal, Decimal to ASCII, Hexa decimal to and Decimal to Hexa.
7		Programs to generate delay, Programs using serial port and on-chip timer/counters
8	Interfacing programs	Stepper motor interface.
9		DC motor interface for direction and speed control using PWM
10		Alphanumerical LCD panel interface.
11		Generate different waveforms: Sine, Square, Triangular, Ramp using DAC interface.
12		External ADC and Temperature control interface.
13		Elevator interface.

Note: For the **experiments 1 to 7, 8051 assembly programming** is to be used.

Note: Single chip solution for **interfacing 8051 is to be done with C Programs** for the **experiments 8 to 13**

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(**ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

Department of Electrical & Electronics Engineering

MICROCONTROLLER – 21EE43(IPCC Course)

Contents

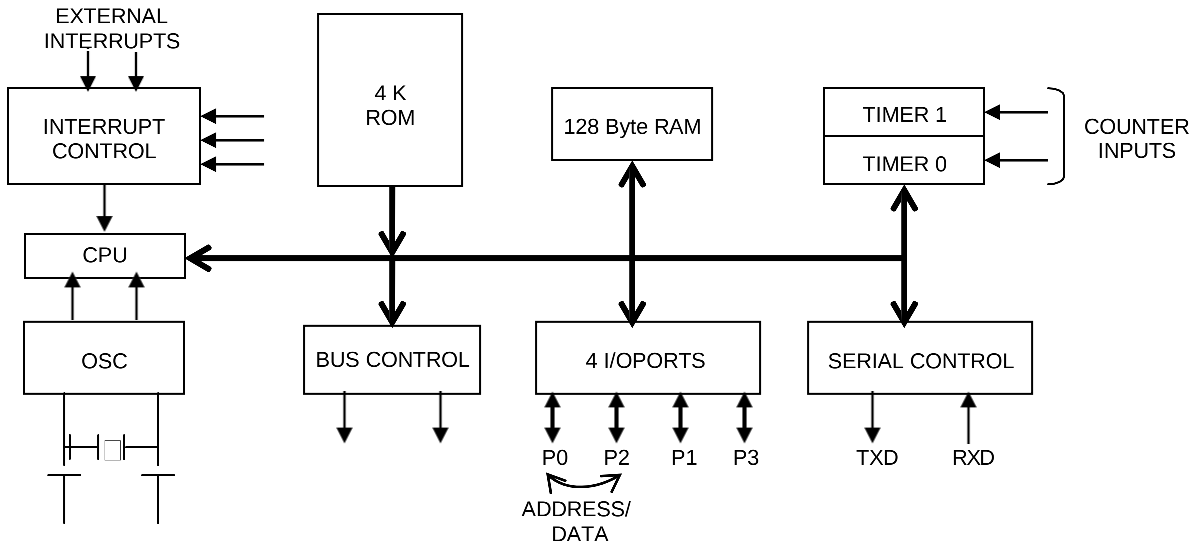
INTRODUCTION

MCS 8051 is an 8-bit single chip microcontroller with many built-in functions and is the core for all MCS-51devices.

The main features of the 8051 coreare:

- Operates with single Power Supply+5V.
- 8-bit CPU optimized for controlapplications.
- 16-bit program counter (PC) and 16-bit data pointer(DPTR).
- 8-bit program status word (PSW).
- 8-bit stack pointer(SP).
- 4K Bytes of On-Chip Program Memory (Internal ROM orEPROM).
- 128 bytes of On-Chip Data Memory (InternalRAM):
    - Four Register Banks, each containing 8 registers (R0 to R7) (Total 32registers).
    - 16 bytes of bit addressablememory.
    - 80 bytes of general-purpose data memory (Scratch PadArea).
- Special Function Registers (SFR) to configure/operatemicrocontroller.
- 32 bit bi-directional I/O Lines (4 ports P0 toP3).
- Two 16-bit timers/counters (T0 and T1).
- Full duplex UART (Universal AsynchronousReceiver/Transmitter).
- 6-source/5-vector interrupts (2 external and 3 internal) with two priority levels.
- On-Chip oscillator and clockcircuitry.

Figure below shows the general block diagram



General Block Diagram of 8051 Microcontroller Architecture

Special Function Registers:

***1. Timer Mode Control Register(TMOD)*:**

TMOD can be considered to be two duplicate 4-bit registers, each of which controls the action of one of the timers. The “Timer” or “Counter” function is selected by control bits C/T, and in different operating modes, which are selected by bit-pairs (M1, M0) in TMOD.

MSB LSB

GATE	C / T	M1	M0	GATE	C / T	M1	M0
Timer1				Timer 0			

GATE	Gating control when set. Counter “x” is enabled only while “INTx” pin is high and “TRx” control pin is set. When cleared Timer “x” is enabled whenever “TRx” control bit is set.
C/T	Timer or Counter Selector cleared for Timer operation (input from internal system clock.) Set for Counter operation (input from “Tx” input pin).
M1 M0	**OPERATI0N**
0 0	13-bit Timer/Counter 5-bits of “TLx” and 8-bits of “THx” are used.
0 1	16-bit Timer/Counter 8-bits of “TLx” and 8-bits of “THx” are cascaded.
1 0	8-bit auto-reload Timer/Counter “THx” holds a value which is to be reloaded into “TLx” each time it overflows.
1 1	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits. Timer/Counter 1 stopped.

***2. Timer Control Register (TCON)*:**

TCON has control bits and flags for the timers in the upper nibble, and control bits and flags for the external interrupts in lower nibble.

MSB LSB

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Bit	Symbol	Function
TCON.7	TF1	Timer 1 overflow flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine, or clearing the bit in software.
TCON.6	TR1	Timer 1 Run control bit. Set/cleared by software to turn Timer/Counter on/off.
TCON.5	TF0	Timer 0 overflow flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine, or by clearing the bit in software.
TCON.4	TR0	Timer 0 Run control bit. Set/cleared by software to turn Timer/Counter on/off.
TCON.3	IE1	Interrupt 1 Edge flag. Set by hardware when external interrupts

		edge detected. Cleared when interrupt processed.
TCON.2	IT1	Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
TCON.1	IE0	Interrupt 0 Edge flag. Set by hardware when external interrupts edge detected. Cleared when interrupt processed.
TCON.0	IT0	Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low Level triggered external interrupts.

3.Interrupt Enable (IE) Register:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EA	x	x	ES	ET1	EX1	ET0	EX0

Symbol	**Name and Function**
EA	Enable All. If 0, Disables all interrupts and no interrupt is acknowledged. If 1, each interrupt can be individually enabled or disabled by programming appropriate bit.
x	Reserved
x	-
ES	Enable Serial Interrupt. If 1, enables TI or RI to generate interrupt.
ET1	Enable Timer 1 interrupt. If 1, Enables the TF1 to generate the interrupt.
EX1	Enable External interrupt 1. If 1, Enables the INT1 to generate the interrupt.
ET0	Enable Timer 0 interrupt. If 1, Enables the TF0 to generate the interrupt.
EX0	Enable External interrupt 0. If 1, Enables the INT0 to generate the interrupt.

4. Interrupt Priority (IP) Register:

Each source of the interrupt can be individually programmed to be in either of the two priority levels. The priorities can be assigned to each interrupt by programming appropriate bits in the SFR Interrupt Priority Register.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
x	x	x	PS	PT1	PX1	PT0	PX0

Symbol	**Name and Function**
x	Reserved
x	Reserved
x	-
PS	Priority of Serial Interrupt. If 1, Priority of Serial Interrupt is higher.

PT1	Priority of Timer 1 interrupt. If 1, Priority of Timer 1 interrupt is higher.
PX1	Priority of External interrupt 1. If 1, Priority of the INT1 is higher.
PT0	Priority of Timer 0 interrupt. If 1, Priority of Timer 0 Interrupt is higher.
PX0	Priority of External interrupt 0. If 1, Priority of the INT0 is higher.

5. Serial Port Control Register (SCON):

The serial port control and status register is the Special Function Register **SCON.** This register contains not only the mode selection bits, but also the 9th data bit for transmit and receive (TB8 and RB8) and the serial port interrupt bits (TI and RI).

MSB							LSB
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Where SM0, SM1 specify the serial port mode, as follows:

SM0	SM1	Mode	Description	Baud Rate
0	**0**	**0**	shift register	f osc / 12
0	**1**	**1**	8-bit UART	Variable
1	**0**	**2**	9-bit UART	f osc / 64 or fosc /32
1	**1**	**3**	9-bit UART	variable

SM2	Enables the multiprocessor communication feature in Modes 2 and 3. In Mode 2 or 3, if SM2 is set to 1, then Rl will not be activated if the received 9th data bit (RB8) is 0. In Mode 1, if SM2=1 then RI will not be activated if a valid stop bit was not received. In Mode 0, SM2 should be0.
REN	Enables serial reception. Set by software to enable reception. Clear by software to disable reception.
TB8	The 9th data bit that will be transmitted in Modes 2 and 3. Set or clear by software as desired.
RB8	In Modes 2 and 3, is the 9th data bit that was received. In Mode 1, it SM2=0, RB8 is the stop bit that was received. In Mode 0, RB8 is not used.
TI	Transmit interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must be cleared by softwareonly.
RI	Receive interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM2). Must be cleared by software only.

STEPS TO CREATE AND COMPILE Keil µVision-3/4 PROJECT:

1. Double Click on the **µVision3/4** icon on thedesktop. 

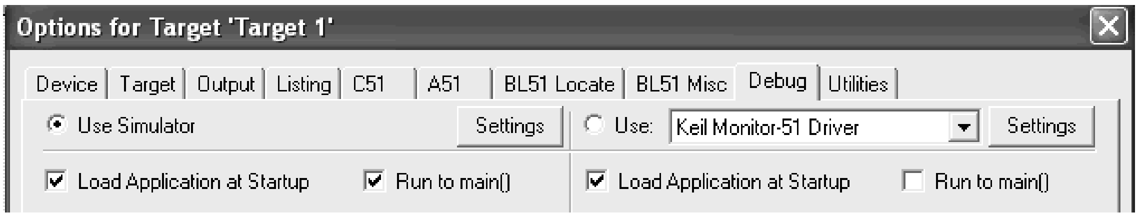
2. Close any previous projects that were opened using – **Project ->Close**.
3. Start **Project – New Project,** and select the CPU from the device database (Database-Atmel- AT89C51ED2 or AT89C51RD2 as per the board).On clicking **'OK'**, the following option is displayed. Choose**'No'**.

4. Create a source file (using **File->New**), type in the assembly or C program and save this (filename**.asm**/filename**.c**) and add this source file to the project using either one of the following two methods. (i)**Project->Manage->Components, Environment Books->addfiles->**browse to the required file **-> OK**

"OR" ii) right click on the Source Group in the Project Window and the **Add Files to Group** option.

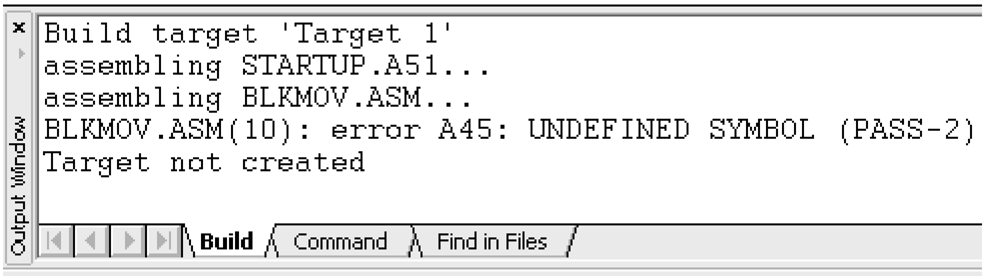
5. Set the Target options using ->**Project** – **Options for Target** opens the µ Vision2 **Options for Target – Target** configuration dialog. Set the **Xtal**(Crystal frequency)frequencyas11.0592MHz,andalsothe**OptionsforTarget**

– Debug – use either Simulator / Keil Monitor- 51 driver.



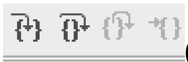
6. If **Keil Monitor- 51 driver is used click on Settings** -> COM Port settings select the COM Port to which the board is connected and select the baud rate as 19200 or 9600 (recommended). Enable **Serial Interrupt** option if the user application is not using on-chip UART, to stop programexecution.

7. Build the project; using **Project -> Build Project**. µVision translates all the user application and links. Any errors in the code are indicated by – “Target not created” in the Build window, along with the error line. Debug the errors. After an error free, to build go to Debugmode.



8. NowusercanenterintoDebugmodewith**Debug-Start/StopDebugsession** dialog. Or by clicking in the  icon.

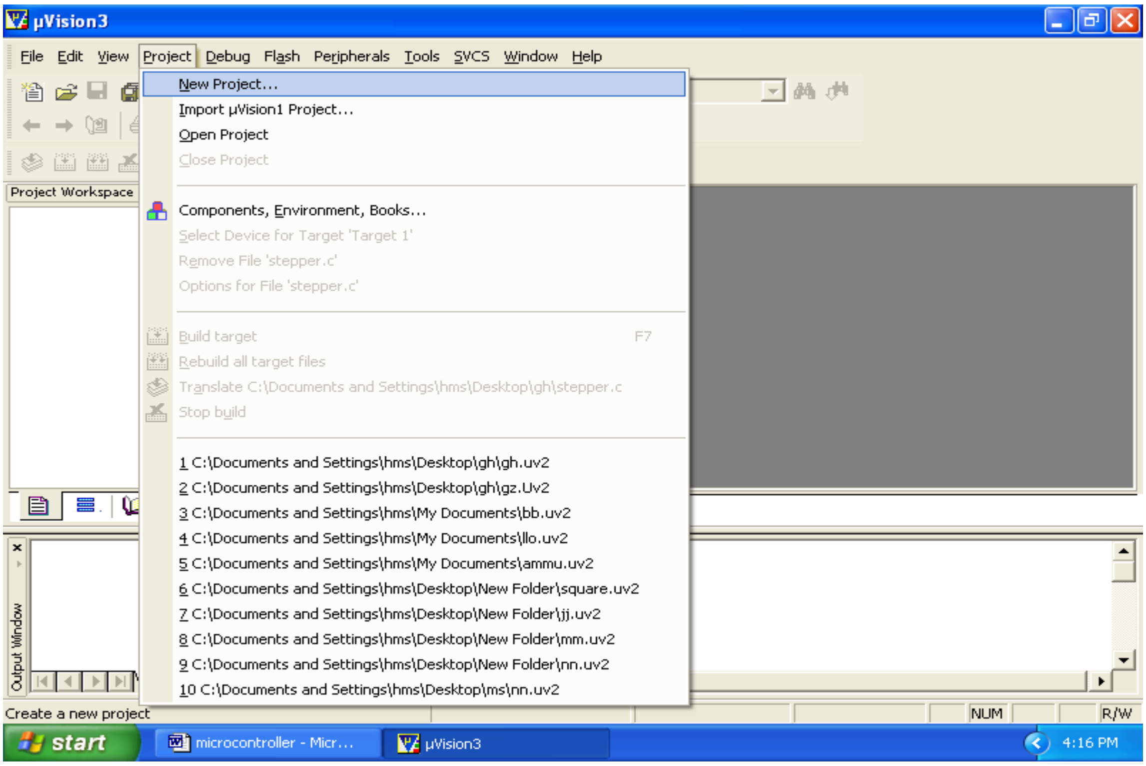
9. The program is run using the **Debug-Run** command & halted using **Debug-Stop Running.** Also the 

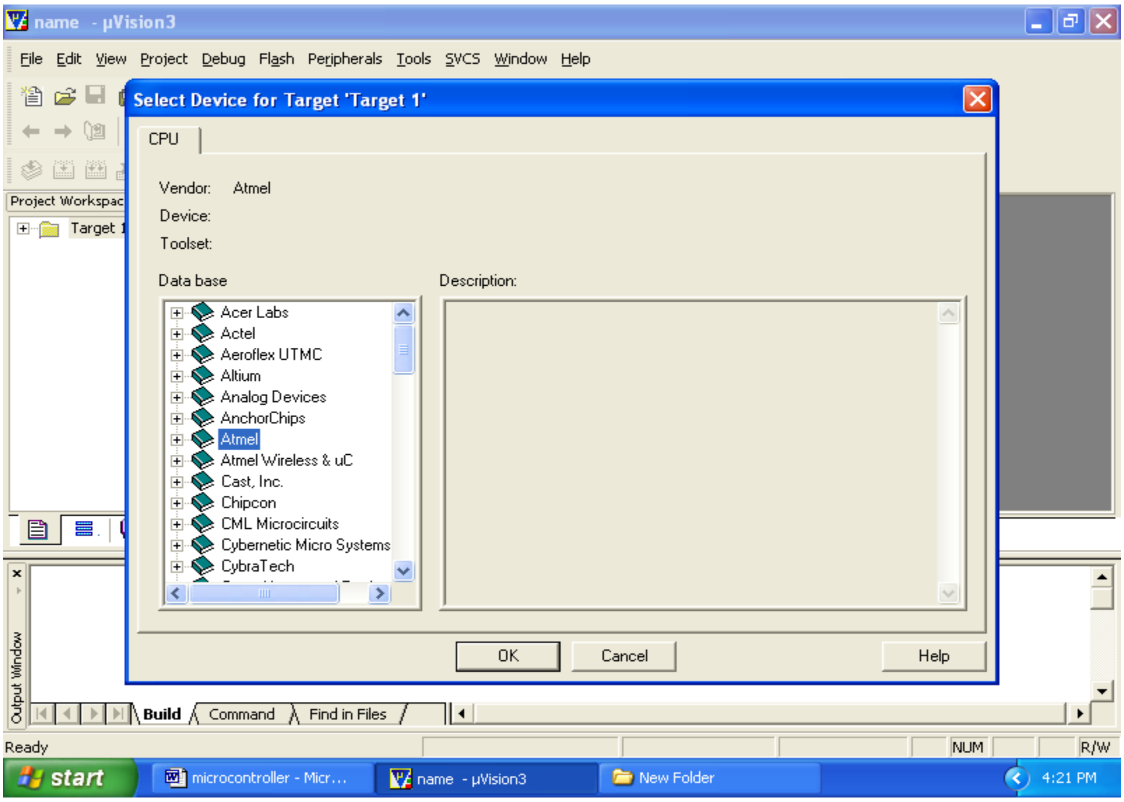
(reset, run, halt) icons can be used. Additional icons are  (step, step over, and step into, run tillcursor).

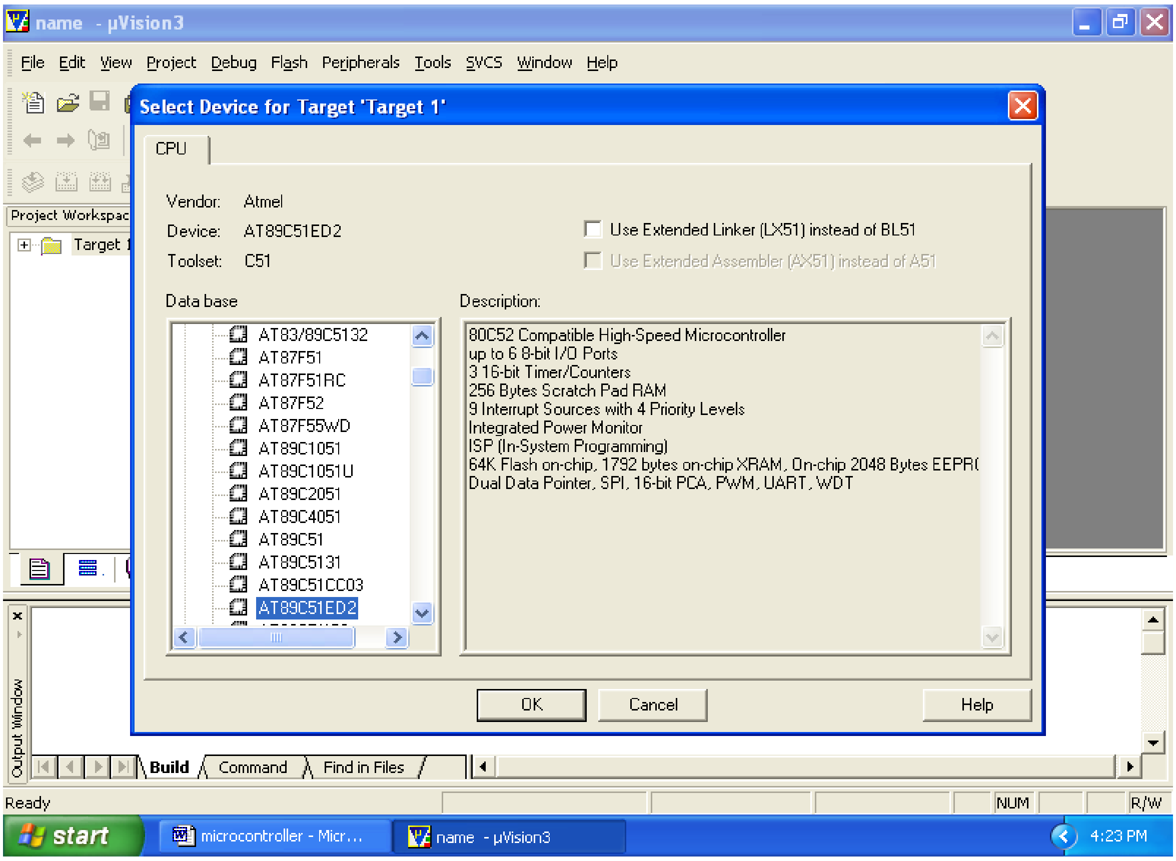
10. IfitisaninterfaceprogramtheoutputscanbeseenontheLCD,CRO,motor,led status, etc. If it is a part-A program, the appropriate memory window is opened using View -> memory window (for data RAM & XRAM locations), Watch window (for timer program), serial window, etc.

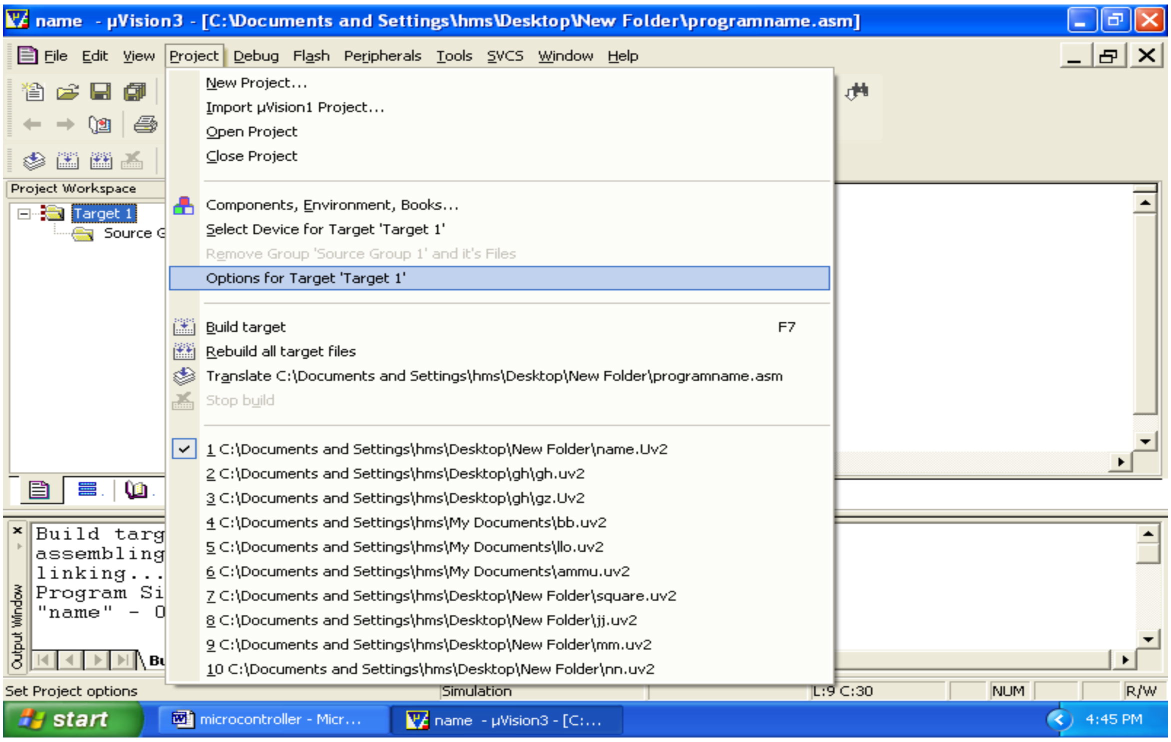
Note: To access data RAM area type address as D: 0020h. Similarly to access the DPTR region (XRAM-present on chip in AT89C51ED2) say 9000h location type in X: 09000H.

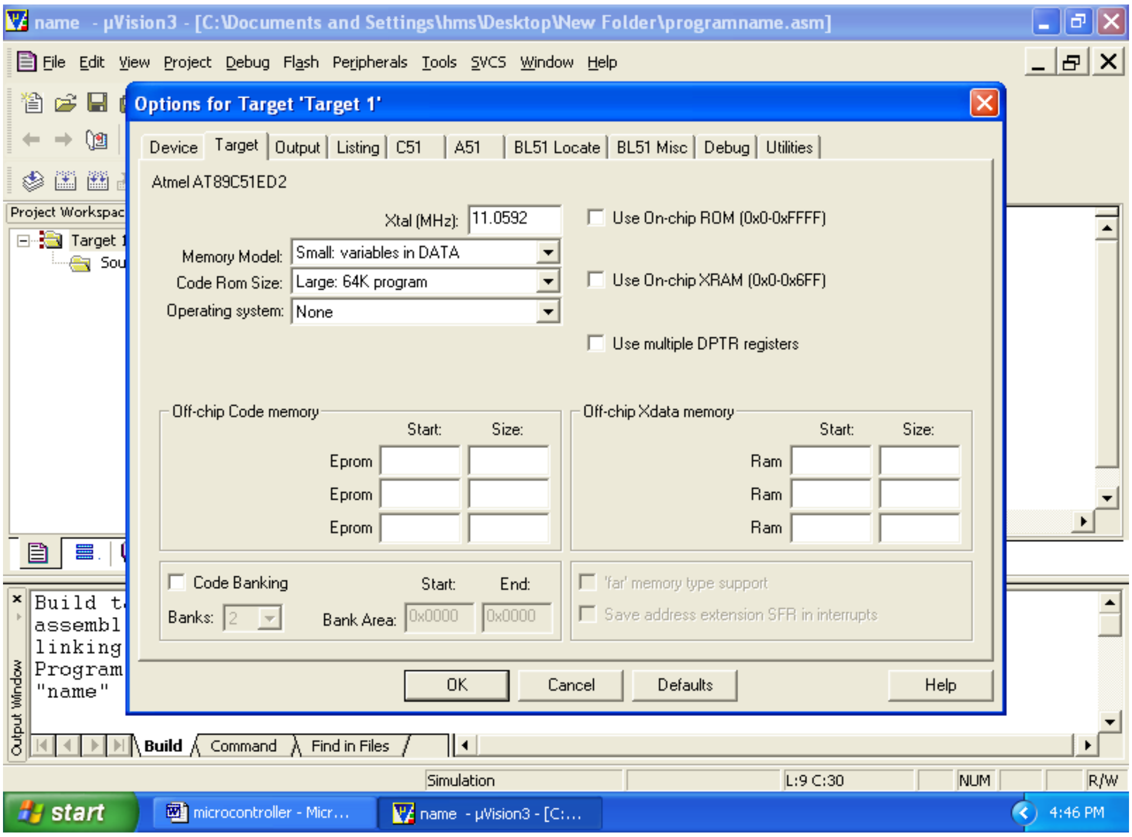
EXECUTION STEPS using KEIL µ vision:

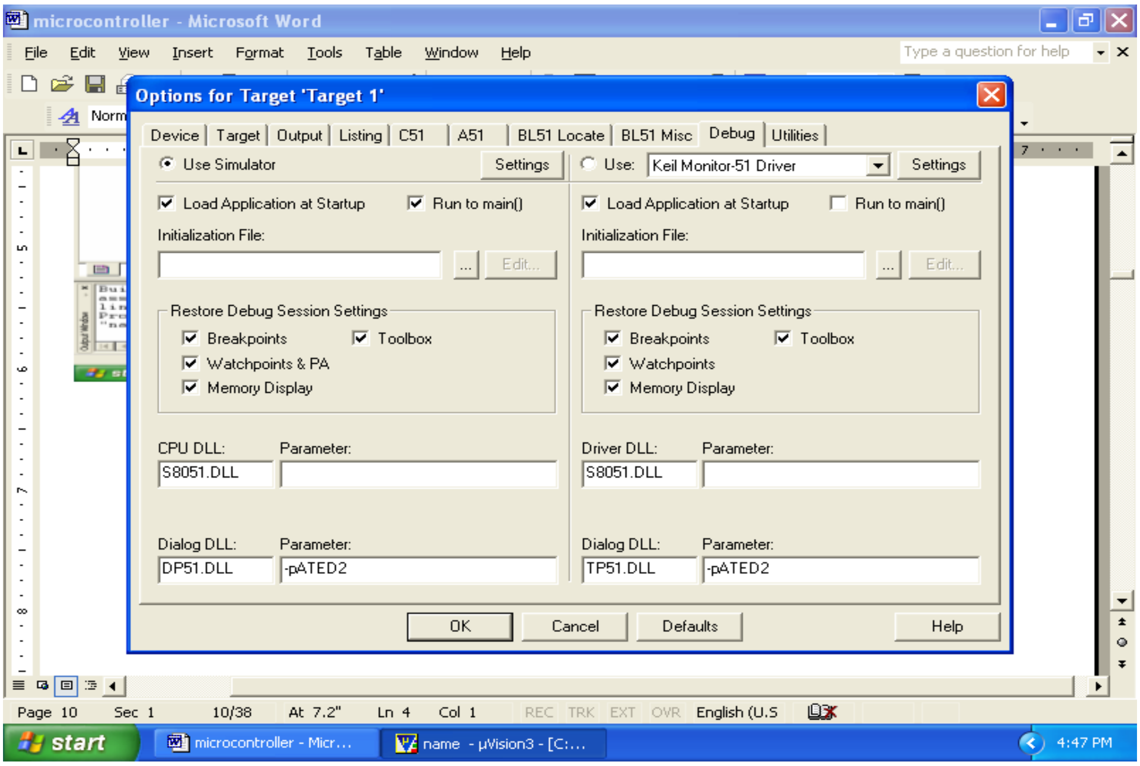


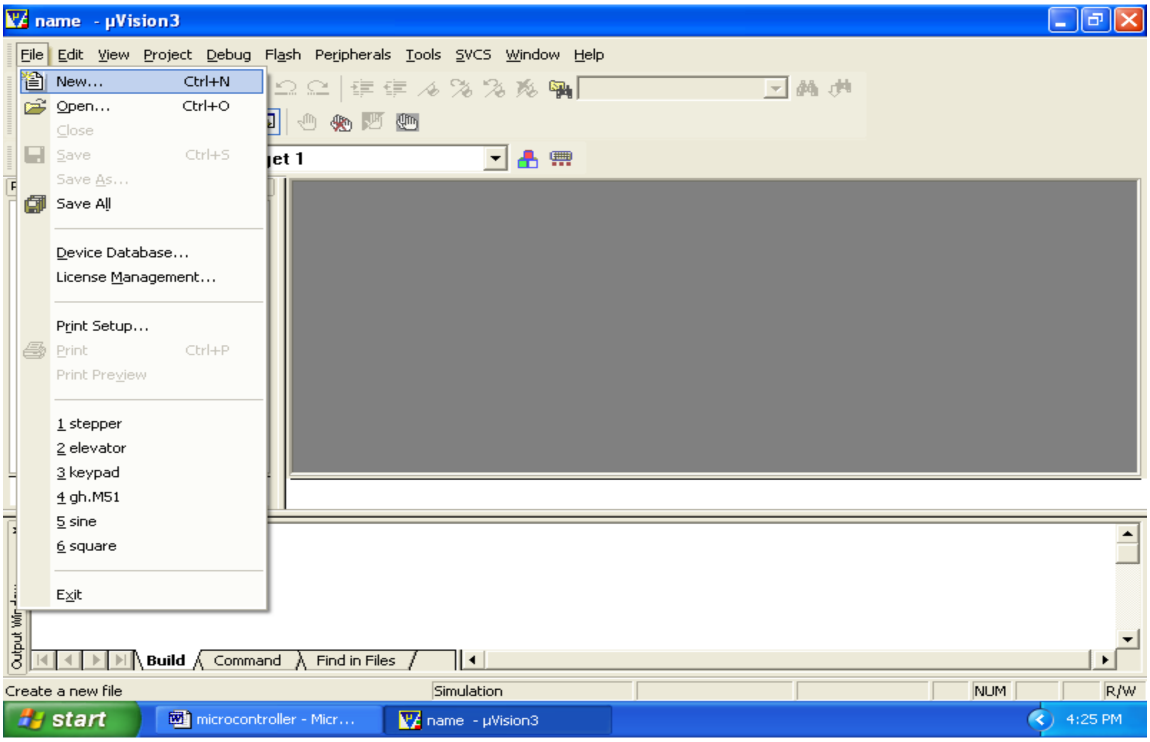


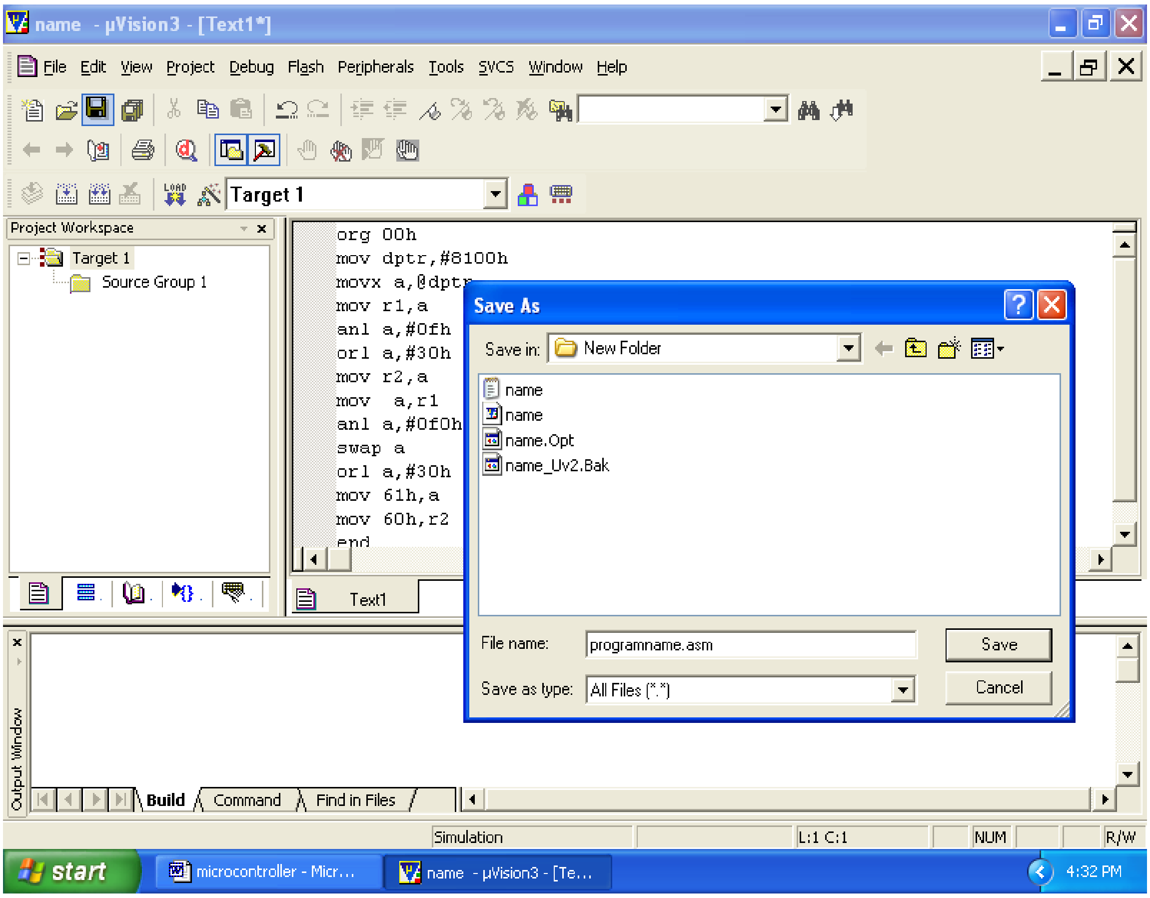

name - µVision3
Select Device for Target 'Target 1'
Vendor: Atmel
Device: AT89C51ED2
Toolset: C51
Use Extended Linker (LX51) instead of BL51
Use Extended Assembler (AX51) instead of A51
Data base
Description:
80C52 Compatible High-Speed Microcontroller
OK
Cancel
Help

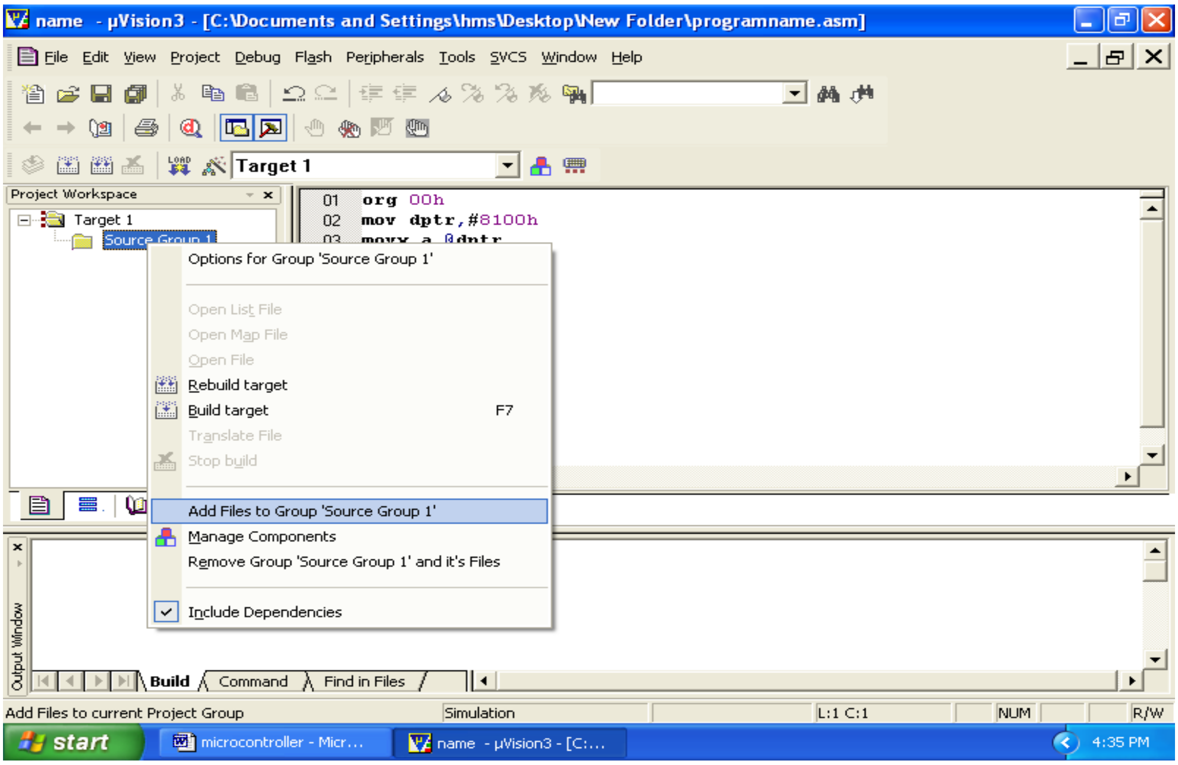

name - µVision3 - [C:\Documents and Settings\hms\Desktop\New Folder\programname.asm]
New Project...
Import µVision1 Project...
Open Project
Close Project
Components, Environment, Books...
Select Device for Target 'Target 1'
Remove Group 'Source Group 1' and it's Files
Options for Target 'Target 1'
Build target F7
Rebuild all target files
Translate C:\Documents and Settings\hms\Desktop\New Folder\programname.asm
Stop build

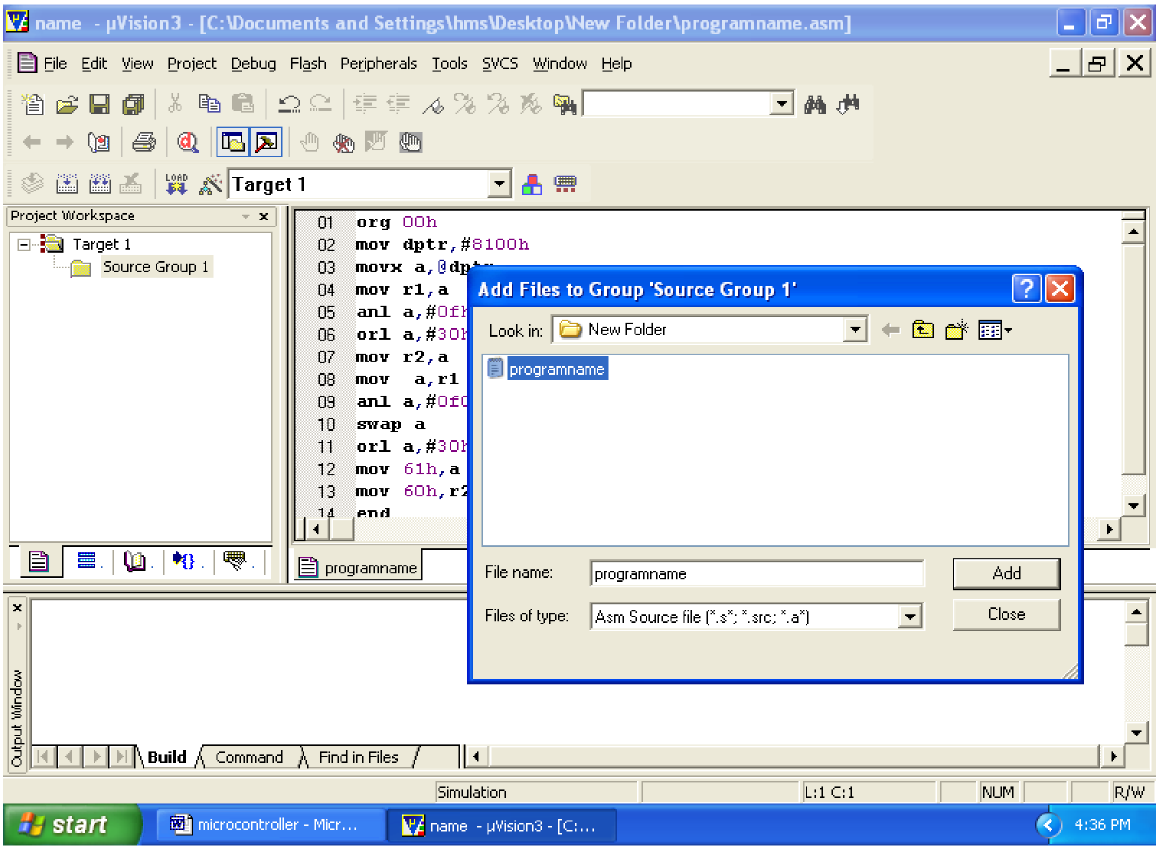


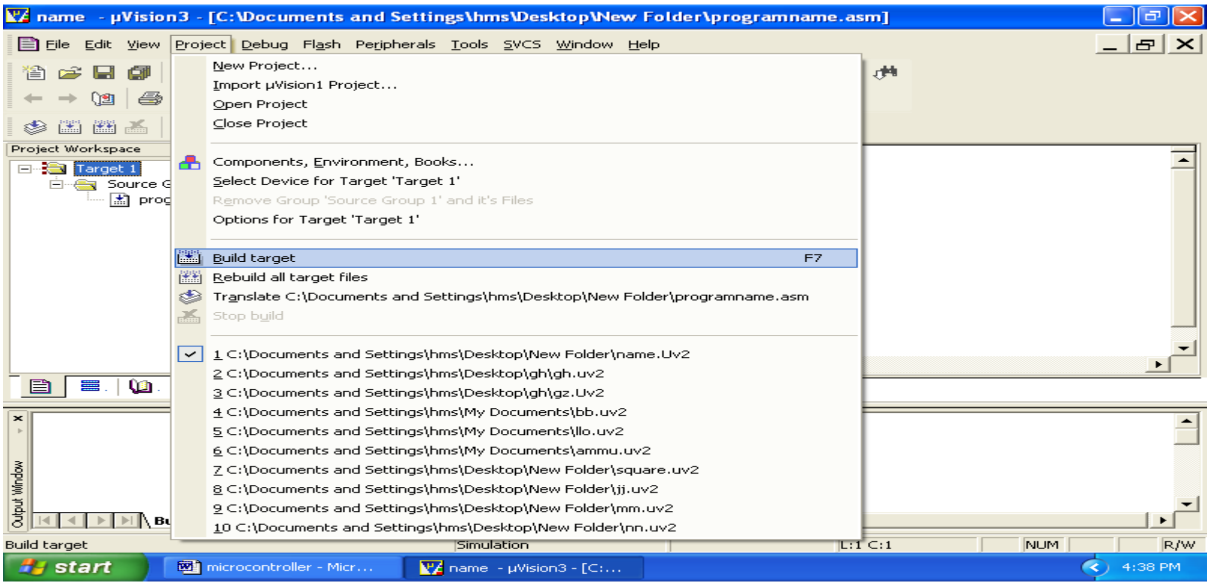


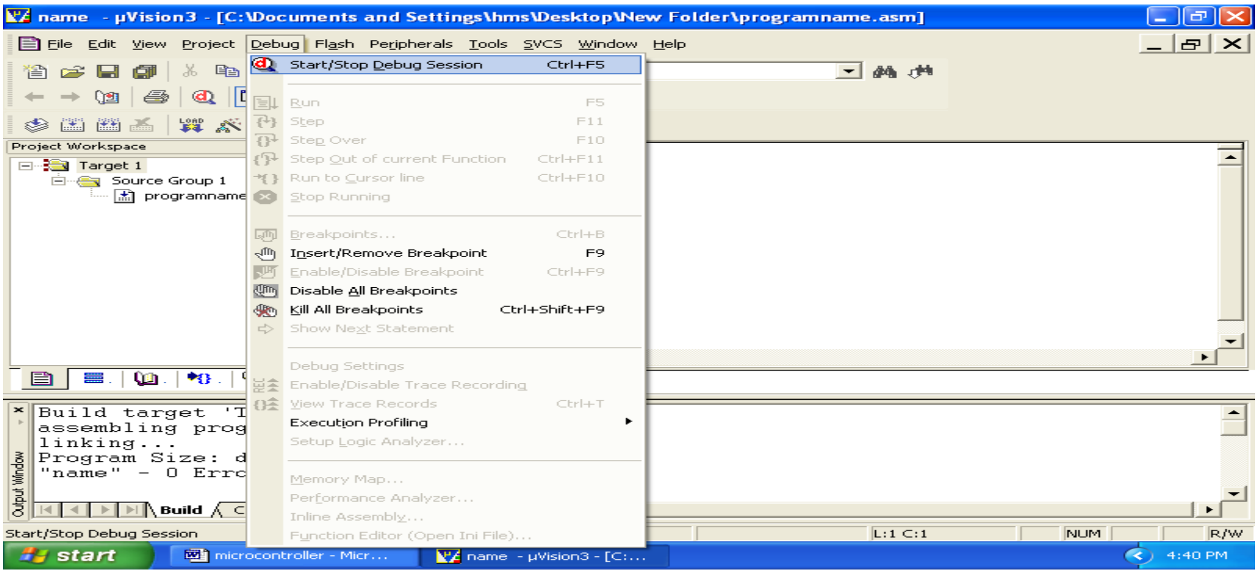

name - µVision3
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
New... Ctrl+N
Open... Ctrl+O
Close
Save Ctrl+S
Save As...
Save All
Device Database...
License Management...
Print Setup...
Print Ctrl+P
Print Preview
1 stepper
2 elevator
3 keypad
4 gh.M51
5 sine
6 square
Exit
Build Command Find in Files
Create a new file
Simulation
NUM
R/W
start
microcontroller - Micr...
name - µVision3
4:25 PM

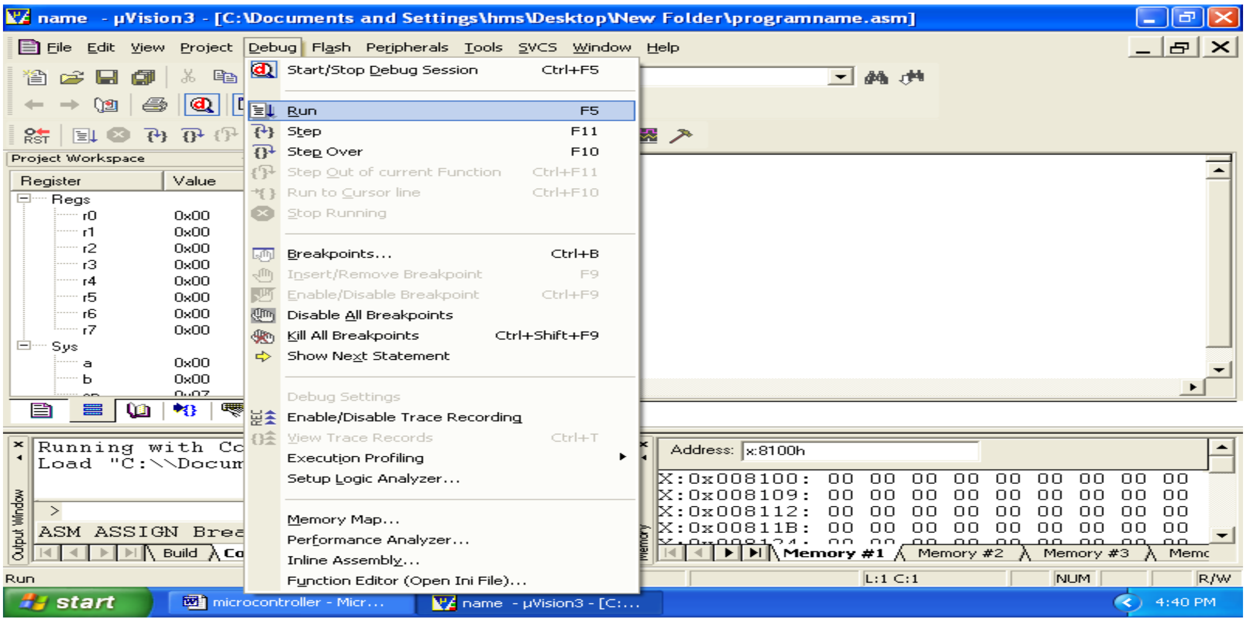

name - µVision3 - [Text1*]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Source Group 1
org 00h
mov dptr,#8100h
movx a,@dptr
mov r1,a
anl a,#0fh
orl a,#30h
mov r2,a
mov a,r1
anl a,#0f0h
swap a
orl a,#30h
mov 61h,a
mov 60h,r2
end
Save As
Save in: New Folder
name
name
name.Opt
name_Uv2.Bak
File name: programname.asm
Save
Save as type: All Files (*.*)
Cancel
Text1
Output Window
Build Command Find in Files
Simulation
L:1 C:1
NUM
R/W
start
microcontroller - Micr...
name - µVision3 - [Te...
4:32 PM


name - µVision3 - [C:\Documents and Settings\hms\Desktop\New Folder\programname.asm]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Source Group 1
01 org 00h
02 mov dptr,#8100h
Options for Group 'Source Group 1'
Open List File
Open Map File
Open File
Rebuild target
Build target F7
Translate File
Stop build
Add Files to Group 'Source Group 1'
Manage Components
Remove Group 'Source Group 1' and it's Files
Include Dependencies
Output Window
Build
Command
Find in Files
Add Files to current Project Group
Simulation
L:1 C:1
NUM
R/W
start
microcontroller - Micr...
name - µVision3 - [C:...
4:35 PM


name - µVision3 - [C:\Documents and Settings\hms\Desktop\New Folder\programname.asm]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Source Group 1
01 org 00h
02 mov dptr,#8100h
03 movx a,@dptr
04 mov r1,a
10 swap a
12 mov 61h,a
14 end
programname
Add Files to Group 'Source Group 1'
Look in: New Folder
programname
File name: programname
Files of type: Asm Source file (*.s*; *.src; *.a*)
Add
Close
Output Window
Build
Command
Find in Files
Simulation
L:1 C:1
NUM
R/W
start
microcontroller - Micr...
name - µVision3 - [C:...
4:36 PM


name - µVision3 - [C:\Documents and Settings\hms\Desktop\New Folder\programname.asm]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
New Project...
Import µVision1 Project...
Open Project
Close Project
Components, Environment, Books...
Select Device for Target 'Target 1'
Remove Group 'Source Group 1' and it's Files
Options for Target 'Target 1'
Build target F7
Rebuild all target files
Translate C:\Documents and Settings\hms\Desktop\New Folder\programname.asm
Stop build
1 C:\Documents and Settings\hms\Desktop\New Folder\name.Uv2
2 C:\Documents and Settings\hms\Desktop\gh\gh.uv2
3 C:\Documents and Settings\hms\Desktop\gh\gz.Uv2
4 C:\Documents and Settings\hms\My Documents\bb.uv2
5 C:\Documents and Settings\hms\My Documents\llo.uv2
6 C:\Documents and Settings\hms\My Documents\ammu.uv2
7 C:\Documents and Settings\hms\Desktop\New Folder\square.uv2
8 C:\Documents and Settings\hms\Desktop\New Folder\jj.uv2
9 C:\Documents and Settings\hms\Desktop\New Folder\mm.uv2
10 C:\Documents and Settings\hms\Desktop\New Folder\nn.uv2
Project Workspace
Target 1
Build target
Simulation
L:1 C:1
NUM
R/W
start
4:38 PM


name - µVision3 - [C:\Documents and Settings\hms\Desktop\New Folder\programname.asm]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Start/Stop Debug Session Ctrl+F5
Run F5
Step F11
Step Over F10
Step Out of current Function Ctrl+F11
Run to Cursor line Ctrl+F10
Stop Running
Breakpoints... Ctrl+B
Insert/Remove Breakpoint F9
Enable/Disable Breakpoint Ctrl+F9
Disable All Breakpoints
Kill All Breakpoints Ctrl+Shift+F9
Show Next Statement
Debug Settings
Enable/Disable Trace Recording
View Trace Records Ctrl+T
Execution Profiling
Setup Logic Analyzer...
Memory Map...
Performance Analyzer...
Inline Assembly...
Function Editor (Open Ini File)...
Project Workspace
Target 1
Source Group 1
programname
Build target 'T
assembling prog
linking...
Program Size: d
"name" - 0 Erro
Start/Stop Debug Session
start
4:40 PM


name - µVision3 - [C:\Documents and Settings\hms\Desktop\New Folder\programname.asm]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Start/Stop Debug Session Ctrl+F5
Run F5
Step F11
Step Over F10
Step Out of current Function Ctrl+F11
Run to Cursor line Ctrl+F10
Stop Running
Breakpoints... Ctrl+B
Insert/Remove Breakpoint F9
Enable/Disable Breakpoint Ctrl+F9
Disable All Breakpoints
Kill All Breakpoints Ctrl+Shift+F9
Show Next Statement
Debug Settings
Enable/Disable Trace Recording
View Trace Records Ctrl+T
Execution Profiling
Setup Logic Analyzer...
Memory Map...
Performance Analyzer...
Inline Assembly...
Function Editor (Open Ini File)...
Register Value
Regs
r0 0x00
r1 0x00
r2 0x00
r3 0x00
r4 0x00
r5 0x00
r6 0x00
r7 0x00
Sys
a 0x00
b 0x00
Running with Co
Load "C:\\Docum
ASM ASSIGN Brea
Address: x:8100h
X:0x008100: 00 00 00 00 00 00 00 00 00
X:0x008109: 00 00 00 00 00 00 00 00 00
X:0x008112: 00 00 00 00 00 00 00 00 00
X:0x00811B: 00 00 00 00 00 00 00 00 00
Memory #1 Memory #2 Memory #3
Run
start
4:40 PM

Programming Using 8051

Basic Programs

Example1: Program for addition of two 8 bit no's

```
Mov  r0,#82h      ; moves the immediate data 82h to r0register
Mov a,r0          ; moves content or data of r0 register toaccumalator
Mov  r1,#02h      ; movesthe immediate data02h to r1register
Mov  b,r1         ; moves the content or data of r1 register to bregister
Add a,b           ; adds accumulator data with b register data and stores
                    Output in accumalator
Mov 60h,a         ; store Output ( data in a) in the direct data address (60h)
end
```

Intermediate outputs to observe : r0= 82h ; a=82h ; r1=02h; b=02h; a=84h

Final Output: D:60h=84h

Example 2: Program for swap function (inter changing the nibbles)

```
Mov a,#21h
Mov 30h,a
Swap a        ; interchanging lower nibble to higher
Mov 31h,a
end
```

Intermediate outputs to observe: a= ; d:30h = ; a= ; d:31h =

Output: Initially a =21 After execution a =12

Example 3: Program for rotate operations

```
mov a,#21h
clr c
mov b,a
rl a              ; rotate accumulator by left
mov 30h,a
mov a,b
rlc a             ; rotate accumulator by left through carry
mov 31h,a
mov a,b
```

```
rr a            ; rotate accumulator by right
mov 32h,a
mov a,b
rrc a           ; rotate accumulator by right through carry
mov 33h,a
end
```

Output: Initiallya =21h

rl (d:30h)=42h

rlc(d:31h)=42h

rr(d:32h)=90h

rrc(d:33h)=10h

Example 4: Program to divide two 8-bit no’s

```
Mov r0 ,#12h        ; get first no. in r0
Mov a,r0            ; copy r0 value to accumulator
Mov r1 ,#05h        ; get second no in r1
Mov b,r1            ; copy r0 value to register b
Div ab              ; divide A by B
Mov 60h,a           ; Quotient value stored in 60h data location
Mov 61h,b           ; reminder value to 61h data location
```

Output: D:60h=

D:61h=

Example 5: program to multiply two 8-bit no’s

```
Mov r0 ,#12h        ; get first no in r0
Mov a,r0            ; copy r0 value to accumulator
Mov r1 ,#05h        ; get second no in r1
Mov b,r1            ; copy r0 value to register b
Mul ab              ; multiply A by B
Mov 60h,a           ; Output stored in 60h data location
```

Output: D:60h=5A

Example 6: Program AND, SWAP, ORoperations

```
 Mov r0 ,#12h       ; get first no inr0
Mov a,r0            ; copy r0 value toaccumulator
Anl a,#0F0h         ; mask lower bit
Mov 60h,a           ; store Output of AND operation in 60h data location
Mov  a,r0           ; copy r0 value toaccumulator
Swap a              ; exchange upper and lower nibbles ofacc
Mov 61h,a           ;store Output of AND operation in 61h data location
Mov  a,r0           ; copy r0 value toaccumulator
Orl a,0f0h          ; OR operation
Mov 62h,a           ;store Output of OR operation in 62h data location
 End
```

Output: D:60h=

D:61h=

D:62h=

Part-A

8051: Assembly Language Programs

General Procedure:

- Double click Kiel µvision
- Goto project Select ❼ Create Newproject
- Select Atmel AT89C51ED2 IDE from the Kielµvision
- SelectNewfile,EntertheprogramandSaveas(.asmin Assemblyand.cinC)andClick ❼ ok
- Add above file to the project created, build target , debug and run theprogram
- observe the result , by giving particular input beforeexecution.

1. Data transfer – Program for block data movement, sorting, exchanging, finding largest element in anarray.

a) Block transfer of data without overlap
b) Sorting ofdata
c) Block exchange ofdata
d) Finding largest number in thearray

1(a). Block transfer of data without overlap

```
        mov dptr,#9000h
        mov 30h,#00h
        mov 31h,#91h
        mov r7,#05h
  back: movx a,@dptr
        inc dptr
        mov  32h,dpl
        mov  33h,dph
        mov dpl,30h
        mov dph,31h
        movx  @dptr,a
        inc dptr
        mov 30h,dpl
      mov 31h,dph
        mov dpl,32h
        mov dph,33h
        djnz r7,back
         end
```

Output:

Before execution					
Source Memory Location	9000	9001	9002	9003	9004
Source Data	01	03	05	07	09
Destination Memory location	9100	9101	9102	9103	9104
Destination data	00	00	00	00	00
After execution					
Source Memory Location	9000	9001	9002	9003	9004
Source Data	01	03	05	07	09
Destination Memory location	9100	9101	9102	9103	9104
Destination data	01	03	05	07	09

Before execution					
Source Memory Location					
Source data					
Destination Memory location					
Destination data					
After execution					
Source Memory Location					
Source Data					
Destination Memory location					
Destination data					

1(b) Sorting (Ascending and descendingorder)

```
        mov   r0,#04h
        dec  r0
back3:  mov r1,00h
        mov dptr,#9000h
back1:  movx a,@dptr
        mov 7fh,a
         Inc dptr
        Movx a,@dptr
        cjne a,7fh,exc
        sjmp back2
   exc: jnc back2
        mov    r3,7fh
        xch a,r3
        movx @dptr,a
        mov a,r3
        movx @dptr,a
        inc dptr
back2:  djnz r1,back1
        djnz r0,back3
        sjmp $
        end
```

Output: for ascending order

Before execution					
Memory Location	9000	9001	9002	9003	9004
Data	05	02	08	03	01
After execution					
Memory Location	9000	9001	9002	9003	9004
Data	01	02	03	05	08

Before execution					
Memory Location					
Data					
After execution					
Memory Location					
Data					

Output: for Descending order decdpl

Before execution					
Memory Location	9000	9001	9002	9003	9004
Data	05	02	08	03	01
After execution					
Memory Location	9000	9001	9002	9003	9004
Data	08	05	03	02	01

Before execution					
Memory Location					
Data					
After execution					
Memory Location					
Data					

Note: Change the instruction **jnc back2** in the program to sort the data in ascending order to **jc back2** to sort the data in descending order.

Date:

1(c) Block exchange of data

```
      mov dptr,#9000h
      mov 30h,#00h
      mov 31h,#91h
      mov r7,#05h
back: movx a,@dptr
      mov 32h,dpl
      mov 33h,dph
      mov r4,a
      mov dpl,30h
      mov dph,31h
      movx a,@dptr
      xch a,r4
      movx @dptr,a
      inc dptr
      mov 30h,dpl
      mov 31h,dph
      mov dpl,32h
      mov dph,33h
      mov a,r4
      movx @dptr,a
      inc dptr
      djnz  r7,back
      end
```

Output:

Before execution					
Source Memory Location	9000	9001	9002	9003	9004
Source Data	01	02	03	04	05
Destination Memory location	9100	9101	9102	9103	9104
Destination data	06	07	08	09	10
After execution					
Source Memory Location	9000	9001	9002	9003	9004
Source Data	06	07	08	09	10

Before execution					
Source Memory Location					
Source Data					
Destination Memory location					
Destination data					
After execution					
Source Memory Location					
Source Data					
Destination Memory location					
Destination data					

1(d) Finding the Largest number in a given array:

```
        Mov dptr,#9000h
        mov r0,#05h
        dec r0
        movx a,@dptr
        mov 7fh,a
back2:  inc dptr
        movx a,@dptr
        cjne a,7fh,back1
        sjmp back3
back1:  jc back3
        mov 7fh,a
back3:  djnz r0,back2
        mov 77h,7fh
        end
```

Output:

Before execution					
Memory Location	9000	9001	9002	9003	9004
Data	05	02	08	03	01
After execution					
Data Location	D:77h	08			

Before execution					
Memory Location					
Data					
After execution					
Data Location	D:77h				

** For finding the **Smallest element** in a given array:

Note: Change the instruction **jc back3** in the program to find largest element in the array to **jnc back3** to find the smallest element in the array.

Output:

Before execution					
Memory Location	9000	9001	9002	9003	9004
Data	05	02	08	03	01
After execution					
Data Location	D:77h	01			

Before execution					
Memory Location					
Data					
After execution					
Data Location	D:88h				

2. Arithmetic instructions: Addition, subtraction, multiplication and division. Square and cube operations for 16 bitnumbers.

(a) Addition

b) Subtraction

(c) Multiplication

(d) Division

(e) Square of anumber

(f) Cube of anumber

2 (a) Addition of two 16 bit numbers:

```
mov dptr,#9001h
mov r0,#0ffh
mov r1,#0ffh
mov r2,#0ffh
mov r3,#0ffh
clr c
mova,r0
add a,r2
movx @dptr,a
dec dpl
mov a,r1
addc a,r3
movx @dptr,a
mov 00h,c
sjmp $
end
```

Output:

```
     r1r0
  +  r3r2
  -----------

---------------
```

2(b) Program for Subtraction of two 16 bit numbers:

```
mov dptr,#9001h // 5673-fc22
mov  r0,#73h
mov  r1,#56h
mov  r2,#22h
mov r3,#0fch
clr c
mov a,r0
subb a,r2
movx @dptr,a
dec dpl
mov a,r1
subb a,r3
movx @dptr,a
mov 00h,c end
```

Output:

```
     r1 r0          56  73  h
  -  r3 r2   ⟹      fc  22h
                  -------------

                  -------------
```

2(c) Multiplication of two 16 bit numbers:

```
Mov dptr,#9003h
mov r0,#23h
mov  r1,#41h
mov  r2,#41h
mov  r3,#32h
mov  a,r3
mov b,r1
mul ab
movx @dptr,a
mov r4,b
mov a,r3
mov b,r0
mul ab
add a,r4
mov r5,a
mov r4,b
mov a,r2
mov b,r1
mul ab
add a,r5
dec dpl
movx @dptr,a
mov a,b
addc a,r4
mov r4,a
mov a,r2
mov b,r0
mul ab
add a,r4
dec dpl
movx @dptr,a
dec dpl
mova,b
movx @dptr,a
end
```

Output: r0 r1 Xr2r3 ⇨ 23 41 X 41 32

----

2 (d) Division of 16 bit by 8 bit number:

```
        org 00h
        mov r0,40h
        mov r1,41h
        mov b,43h
        mov a,r0
        div ab
        mov 45h,a
        mova,b
        mov b,#0ah
        mul ab
        add a,r1
        movb,43h
        div ab
        mov 46h,a
here:   sjmp here
        end
```

Output: r1 r0 ÷b

2 (e) Find square of a number:

```
mov dptr,#9000h
movx a,@dptr
movb,a
mul ab
mov r0,a
mov dptr,#900eh
mov a,b
movx @dptr,a
inc dpl
mov a,r0
movx @dptr,a
end
```

Output: X : 900e h =$(accumulator)^2$

2(f) . Program to find cube of a number:

```
mov dptr,#9000h
movx a,@dptr
mov r0,a
mov b,a
mul ab
mov r1,b
mov b,r0
mul ab
mov dptr,#900e h
movx @dptr,a
mov r2,b
mov a,r1
movb,r0
mul ab
add a,r2
dec dpl
movx @dptr,a
dec dpl
mova,b
movx @dptr,a
end
```

Output: X : 900e h =(accumulator)^3

3. Counters (UP/DOWN)

3(a) Program for Binary up counter

```
          Mov dptr,#9000h
          mov     a,#00h
    next: movx @dptr,a
          acall delay
          inc     a
          jnz     next
    here:    sjmp here
    delay:   mov   r1,#0ffh
    loop1:  mov   r2,#0ffh
    loop2:  mov   r3,#0ffh
    loop3:  djnz   r3,loop3
            djnz   r2,loop2
            djnz   r1,loop1
            ret
            end
```

Output: x:9000h=00,01,02. ff

3(b). Program for Binary down counter

```
         mov dptr,#9000h
         mov a,#0ffh
next:    movx @dptr,a
         acall delay
         dec a
         jnz  next
          movx@ dptr, a
here:     sjmp   here
delay:movr1,#0ffh
        loop1:movr2,#0ffh
        loop2:movr3,#0ffh
       loop3:djnzr3,loop3
              djnz r2, loop2
               djnz r1,loop1
       ret
      end
```

Output: x:9000h=ff,fe,fd. ... 00

3(c) . Program for Decimal up counter

```
        Mov dptr,#9000h
        mov     a,#00h
next:   movx @dptr,a
        acall   delay
        add     a,#01h
        da      a
        jnz     next
here:   sjmp    here
delay:  mov     r1,#0ffh
loop1:  mov     r2,#0ffh
loop2:  mov     r3,#0ffh
loop3:  djnz    r3,loop3
        djnz    r2,loop2
        djnz    r1,loop1
        ret
        end
```

Output: x: 9000h=00,01,02. 99

3(d) Program for Decimal down counter

```
        Mov dptr,#9000h
        mov     a,#99h
next:   movx @dptr,a
        acall   delay
        add     a,#99h
        da      a
        jnz     next
        movx@dptr,a
here:   sjmp    here
delay:  mov     r1,#0ffh
loop1:  mov     r2,#0ffh
loop2:  mov     r3,#0ffh
loop3 : djnz    r3,loop3
        djnz    r2,loop2
        djnz    r1,loop1
        ret
        end
```

Output: x: 9000h=99,98,97… 00

4. Boolean and Logical instructions (BitManipulation):

4(a) Write an ALP to compare two eight bit numbers NUM1 and NUM2 stored in external memory locations 8000h and 8001h respectively. Reflect your result as: If NUM1<NUM2, SET LSB of data RAM location 2FH (bit address 78H). If NUM1>NUM2, SET MSB of location 2FH (bit address 7FH). If NUM1 = NUM2, then Clear both LSB & MSB of bit addressable memory location 2FH.

```
        mov dptr,#8000h
        movx a,@dptr
        mov r0,a
        incdptr
        movx a,@dptr
        clr c
        sub  a,r0
        jz equal
        jnc small
        setb 7fh
        sjmp end1
small:  setb 78h
        sjmp end1
equal:  clr 78h
        clr 7fh
 end1:  end
```

Result:

1) Before Execution: X: 8000h = & X: 8001 =
   After Execution: D: 02FH =
2) Before Execution: X: 8000h = & X: 8001 =
   After Execution: D: 02FH =
3) Before Execution: X: 8000h = & X: 8001 =
   After Execution: D: 02FH =

4(b) Write an assembly language program to count number of ones and zeros in a eight bit number.

```
        mov r1,#00h // to count number of 0s
        mov r2,#00h // to count number of 1s
        mov r7,#08h // counter for 8-bits
        mov a,#97h // data to count number of 1s and 0s
again: rlc a
        jc next
        inc r1
        sjmp here
 next: incr2
here: djnz r7,again
       end
```

Result:

Input: **Output:**

Number of zero's = r2 =

Number of one's = r1

4(c) Write an assembly language program to find whether given eight bit number is odd or even. If odd store 00h in accumulator. If even store FFh in accumulator.

```
       mov a,20h      // 20h=given number, to find is it even or odd
       jbacc.0,odd     //jump if direct bit is set i.e., if lower bit is1
                         then number is odd
       mov a,#0FFh
       sjmp next
odd: mov a,#00h
next:end
```

Result:

Input: **Output:**

20h: a:

4(d) Write an assembly language program to perform logical operations AND, OR, XOR on two eight bit numbers stored in internal RAM locations 21h, 22h.

```
mov a, 21h //do not use #, as data ram 21h is to be accessed
anl a, 22h     //logical andoperation
mov 30h, a //and operation result stored in 30h
mov a, 21h
orl a,22h      //logical or operation
mov 31h, a //or operation result stored in 31h
mov a,21h
xrl a,22h     //logical xoroperation
mov 32h,a // xor operation result stored in 32h
end
```

Result:

Before Execution: D:21H = D: 22H =

After Execution: D:30H= //ANDoperation

D: 31H= //OR operation

D: 32H= //XOR operation

4(e) Write a Program to check whether given number is palindrome or not. If palindrome store FFh in accumulator else store 00h inaccumulator.

```
      mov 30h,#81h
      mov r0,30h
      mov r1,#08h
      mov 31h,#00h
      clr c
back: mov a,30h
      rlc a
      mov  30h,a
      mov  a,31h
      rrc a
      mov 31h,a
      djnz r1,back
      cjne a,00h,npal
      mov a,#0ffh
      sjmp  next
npal: mov a,#00h
next: end
```

Result:

Input: **Output:**

5. Conditional call and returninstructions:

Ex 1: write a program to clear accumulator [a], then add 5 to the accumulator 20 times

```
        Mov a,#00h
        mov r4,#20
again:  add a,#05h
        mov 30h,a
        call delay
        djnz r4,again
        mov r5,a
delay:  mov  r1,#0ffh
  loop1:mov r2,#0ffh
  loop2:mov r3,#0ffh
  loop3:djnz r3,loop3
        djnz r2,loop2
        djnz r1,loop1
        ret
```

Output:

Ex 2: **write a program in which if R4 register contains the value 0. Then put 55H in R4 register**:

```
mov a,r4
jnz next
mov r4,#55h
next: mov a, r4
end
```

Output:

6. Code conversionprograms

a) **BCD toASCII**
b) **ASCII toBCD**
c) **ASCII toDecimal**
d) **Decimal toASCII**
e) **Hexa todecimal**
f) **Decimal toHexa**

6a) Program to convert a BCD number into ASCII code:

```
        mov dptr,#9000h
        movx a,@dptr
        mov r0,a
        swap a
        mov dptr,#900dh
        acall ascii
        mov a,r0
        acall ascii
        sjmp $
ascii:  anl a,#0fh
        add a,#30h
        movx @dptr,a
        inc dptr
        ret
        end
```

Result:

Before execution			
Memory Location	9000	900d	900e
Data	45	00	00
After execution			
Memory Location	9000	900d	900e
Data	45	34	35

Before execution			
Memory Location	9000	900d	900e
Data	97	00	00
After execution			
Memory Location	9000	900d	900e
Data	97		

6b) Program to convert a ASCII to BCD

```
mov a,#’4’
anl a,#0fh
swap a
mov b,a
mov a,#’7’
anl a,#0fh
orl a,b
```

Output: a=

6c) Program to convert a ASCII number into decimal

```
Mov dptr,#9000h
movx a,@dptr
Clr c
subb a,#30h
movx dptr,a
end
```

Result:

Before execution	
Memory Location	9000
Data	33
After execution	
Memory Location	9000
Data	03

Before execution	
Memory Location	9000
Data	97
After execution	
Memory Location	9000
Data	

Date:

6d) Program to convert decimal number to ASCII

```
mov dptr,#9000h
movx a,@dptr
add a,#30h
mov dptr,#900dh
movx @dptr,a
end
```

Result:

Before execution	
Memory Location	9000
Data	03
After execution	
Memory Location	9000
Data	33

Before execution	
Memory Location	9000
Data	63
After execution	
Memory Location	9000
Data	

6e) Program to convert Hex number to Decimal:

```
org 00h
mov a,#0a9h
mov b,#0ah
div ab
mov  r0,b
mov b,#0ah
div ab
mov r1,b
mov r2,a
end
```

Result: r0=01

r1=06

r2=09

6f) Program to convert decimal number to HEX:

```
mov dptr,#9000h
movx a,@dptr
mov r0,a
anl a,#0f0h
swap a
movb,#0ah
mul ab
mov r1,a
mov a,r0
anl a ,#0fh
adda,r1
movx @dptr,a
end
```

Result:

Before execution	
Memory Location	9000
Data	55
After execution	
Memory Location	9000
Data	37

Before execution	
Memory Location	9000
Data	99
After execution	
Memory Location	9000
Data	

7. Programs to generate delay, Programs using serial port and on-chiptimer/counters.

a) Program to configure 8051 microcontroller to transmit characters“ENTER YOUR NAME” to a PC using the serial port and display on the serial window.
b) Program to generate 1second delay continuously using on chiptimer.

Note: To use result of this program, after selecting DEBUG session in the main menu use View-> serial window #1. On running & halting the program, the data is seen in the serial window.
(11.0592MHz)/(12) by 32 before it is being used by the timer to set the baud rate.

To get 9600, 28800/3 is obtained by loading timer1 with -3 (i.e., FF – 3 = FD) for further clock division. For 2400 baud rate, 28800/12 => -12 = F4 in TH1

7 a) Program to configure 8051 microcontroller to transmit characters “ENTER YOUR NAME” to a PC using the serial port and display on the serial window

```
          mov tmod,#20h //setting Timer-1 in mode-2
          mov scon,#70h
          mov th1,#-3
          setb tr1
again:    mov r0,#03h
          mov dptr,#8000h
nextchar: movx a,@dptr
          acall transfer
          incdptr
          djnz r0,nextchar
          sjmp again
transfer: mov sbuf,a
wait:     jnb ti,wait
          clr ti
          ret
          end
```

RESULT:
Each time the program is executed, “ENTER YOUR NAME” will be displayed on the serial window.

Baud rate Calculation:

Crystal freq/ (12*32) = (11.0592MHz)/(12*32) = 28800.

Serial communication circuitry divides the machine cycle frequency

7b) Program to generate 1second delay continuously using on chip timer.

```
        mov tmod,#02h
        mov th0,#00h
        clr P1.0
        clr a
        setb tr0
again: mov r7,#0ffh
loop: mov r6,#14d
wait: jnb tf0, wait
        clr tf0
        djnz r6,wait
        djnzr7,loop
        cpl P1.0
        sjmp again
        end
```

RESULT:

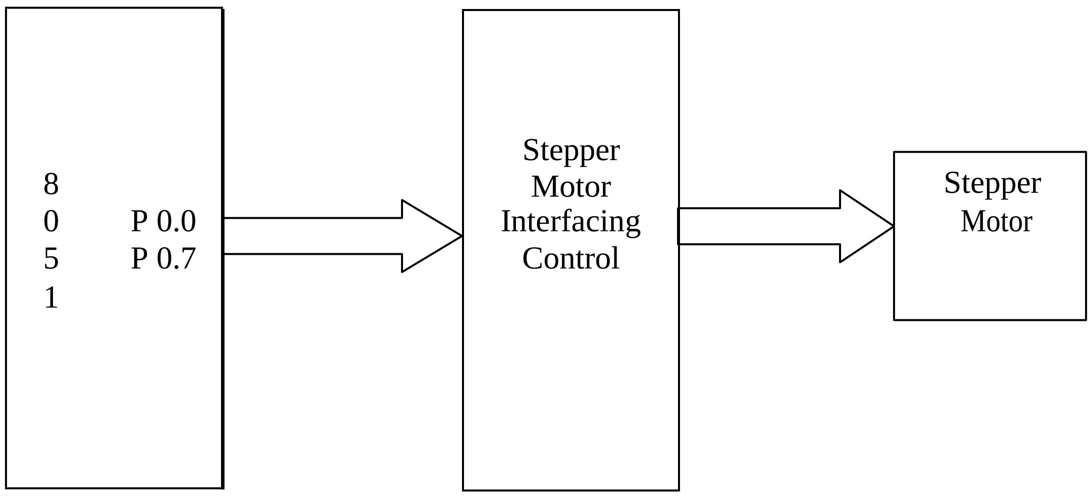
Accumulator A is incremented in binary from 00, 01,02…09,0A, 0B, …, 0F, 10,11, …FF every 1 second (for 33MHz clock setting & every 3 seconds for11.0598MHz)

Part –B

Interfacing Programs

8. Program for stepper motorinterface.

BlockDiagram:

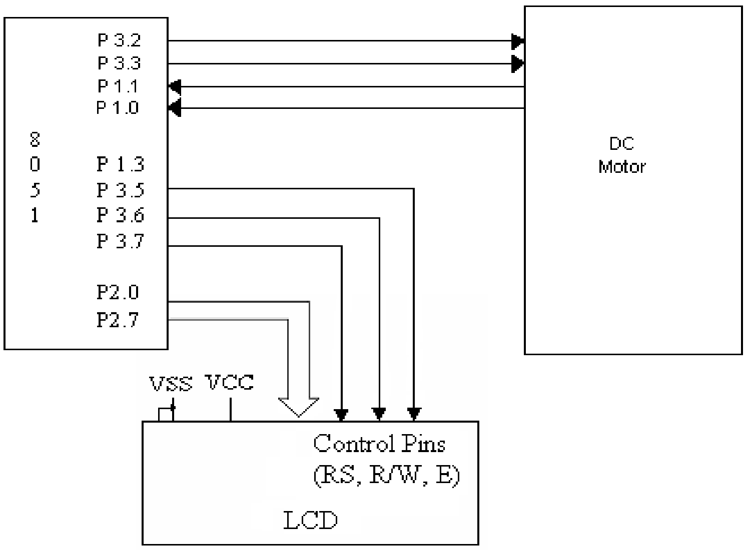



```
#include <REG51xD2.H>
void delay (unsigned int x)                    /* Delay Routine*/
{
 for(;x>0;x--);
 return;
}
main ( )
{
  unsigned char Val, i;
  P0=0x00;
while(1)
{
  Val = 0x11;
  for (i=0;i<4;i++)
              {
               P0 = Val;
               Val=Val<<1;                /* Val= Val>>1; for clockwise direction*/
               delay(500);
              }
 }
}
```

Output

9. Program for Dc motor interface for direction and speedcontrol using PWM.

BlockDiagram:

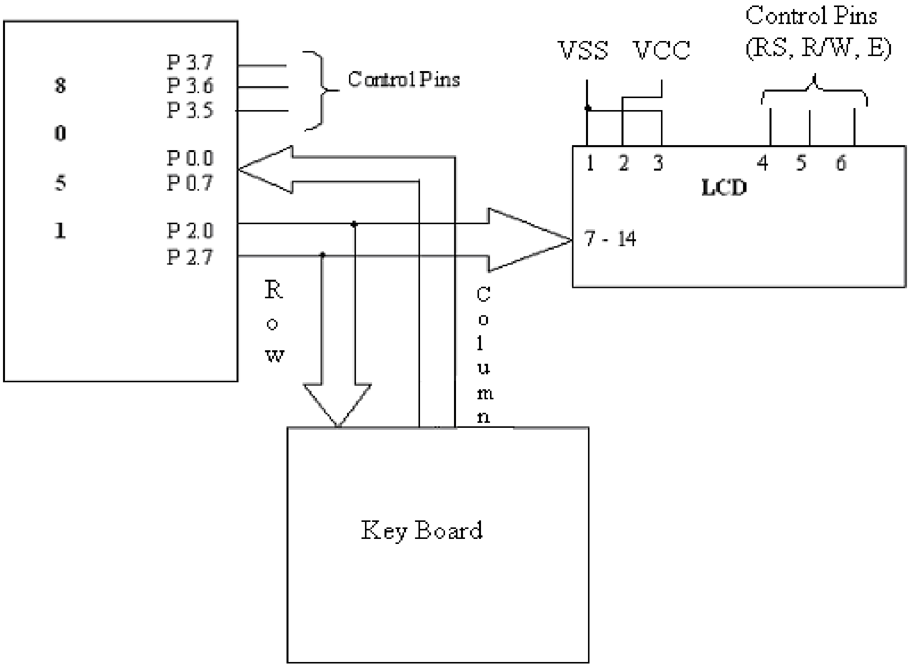


This program measures the motor speed and displays it on LCD
This Program uses Po for DAC data i.e. for speed increment or decrement

```
#include <REG51xD2.H>
Sbit inr= P3^2; //speed increment switch
sbit dcr= P3^3; //speed decrement switch
main()
{
    unsigned char i=0x80;
    P0 =0x7f;                 /*Run the motor at half speed.*/
while(1)
    { if (!inr)
        {while (!inr);
          if(i>10)
           i=i-10;          //increase the DC motor speed
          }
        if(!dcr)
        {
          while(!dcr);
           if(i<0xf0)
          i=i+10;           //decrease the DC motorspeed
        }
      P0=i;
    }
}
```

10. Program to interface Alphanumerical LCD panel and Hex keypad to 8051.

Block diagram :



LABEL ON THE KEYTOP	HEX CODE	LABEL ON THE KEYTOP	HEX CODE
0	0	-	0C
1	1	*	0D
2	2	/	0E
3	3	%	0F
4	4	AC	10
5	5	CE	11
6	6	CHK	12
7	7	=	13
8	8	MC	14
9	9	MR	15
.	0A	M	16
+	0B	M+	17

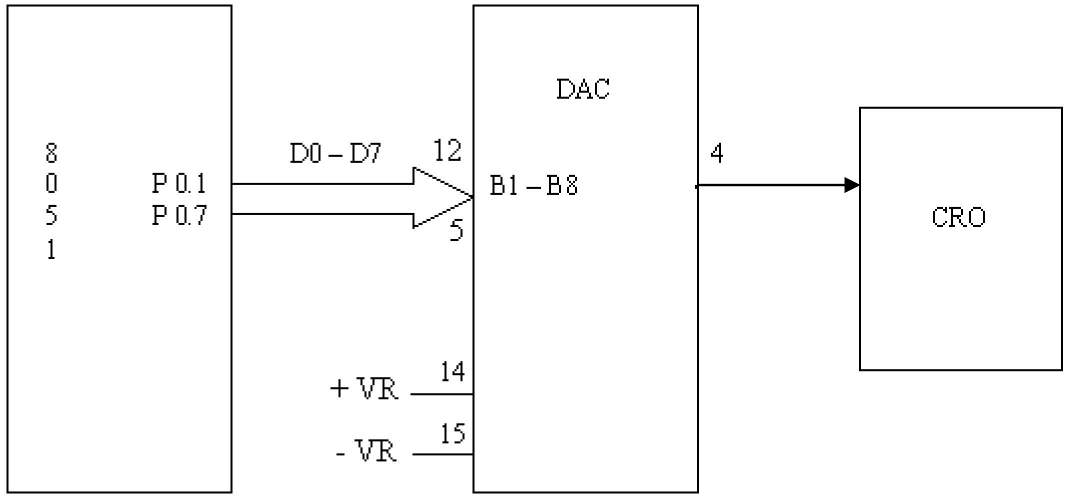
```
#include <REG51xD2.H>
#include "lcd.h"

unsigned char getkey();
void delay(unsigned int);

main()
{
  unsigned char key,tmp;
```

```
  InitLcd();                                      /* Initialise LCD*/
  WriteString("KeyPressed=");                     /* Display msg on LCD */
  while(1)
  {
   GotoXY(12,0);                                  /* Set Cursor Position */
        key= getkey();                            /* Call Getkey method*/
  }
}

unsigned char getkey()
{
 unsigned char i,j,k,indx,t;
 P2=0x00;                                 /* P2 as Output port */

 indx=0x00;                               /* Index for storing the first value of
                                             the scanline*/
 for(i=1;i<=8;i<<=1)                      /* for 4 scanlines*/
 {
        P1 = 0x0f&~i;                     /* write data to scanline*/
        t =P0;                            /* Read readlines connected to P0*/
        t =~t;
        if(t>0)                           /* If key press is true*/
        {
         delay(6000);                     /* Delay for bouncing*/
         for(j=0;j<=4;j++)               /* Check for 8 lines*/
         {
          t >>=1;
               if(t==0)                  /* if get pressedkey*/
               {
                k =indx+j;               /* Display that by converting to Ascii*/
                if(k >9)
           k+=0x37;
                else
           k+=0x30;
                WriteChar(k);
                return(indx+j);         /* Return index of the key pressed*/
               }
         }
        }
        indx+=0x04;                     /* If no key pressed increment index*/
 }
}

void delay(unsigned int x)             /* delay routine*/
{
 for(;x>0;x--);
}
```

Signature o Staff

11. (a) Program for dual DAC interfacing to generate square wave of frequency 'f'.

Block Diagram:



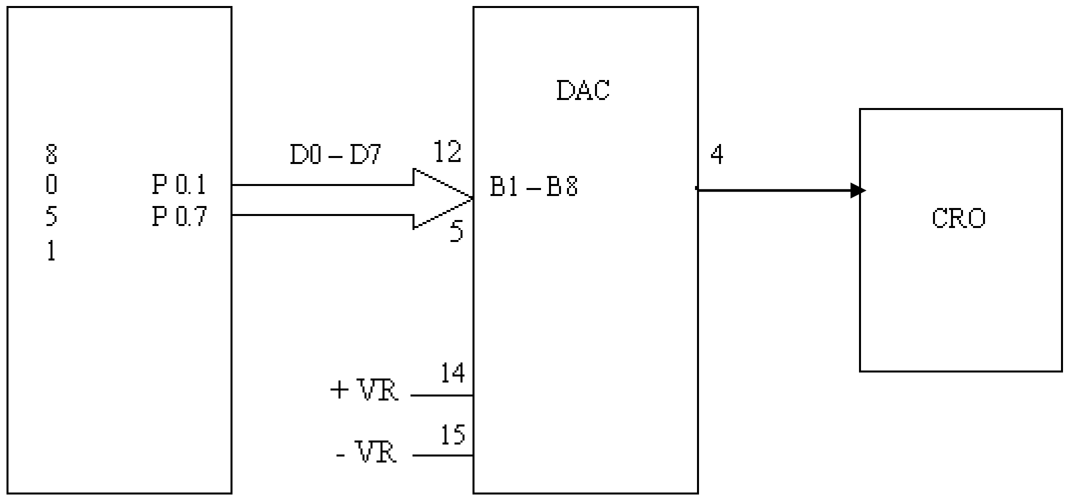

```
#include <REG51xD2.H>

sbit Amp=P3^3;                  /* Port line to change amplitude*/
sbitFre=P3^2;                   /* Port line to change frequency*/


void delay(unsigned int x)      /* delay routine*/
{
  for(;x>0;x--);
}

main()
{
 unsigned char on = 0x7f,off=0x00;
 unsigned int fre = 100;

 while(1)
 {
  if(!Amp)                      /* if user choice is to change amplitude*/
  {
    while(!Amp);                /* wait for key release */
        on+=0x08;               /* Increase the amplitude*/
  }

  if(!Fre)                      /* if user choice is to change frequency*/
  {
   if(fre>1000)                 /* if frequency exceeds 1000 reset to default */
        fre =100;
```

```
        while(!Fre);            /* wait for key release */
    fre+=50;                    /* Increase the frequency*/
   }
   P0=on;                                    /* write apmlitude to port*/
   delay(fre);
   P0=off;                                   /* clear port*/
   delay(fre);

  }
}
```

Date:

11(b). Program for dual DAC interfacing to generate ramp waveform.

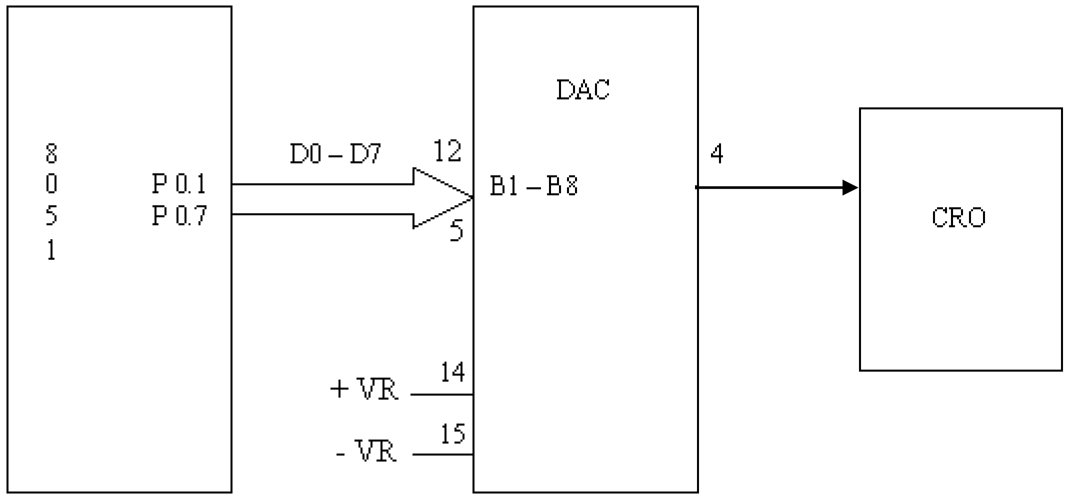
BlockDiagram:




```
#include

<REG51xD2.H>main()
{
  unsigned char i=0;

  P0=0x00;                              /* P0 as Output port */
  while(1)
  {
   {
    for(i=0;i<0xff;i++)                 /* Generate ON pulse */
     P0 =i;
 }

}
```

11(c) Program for dual DAC interfacing to generate triangular wave.

BlockDiagram:

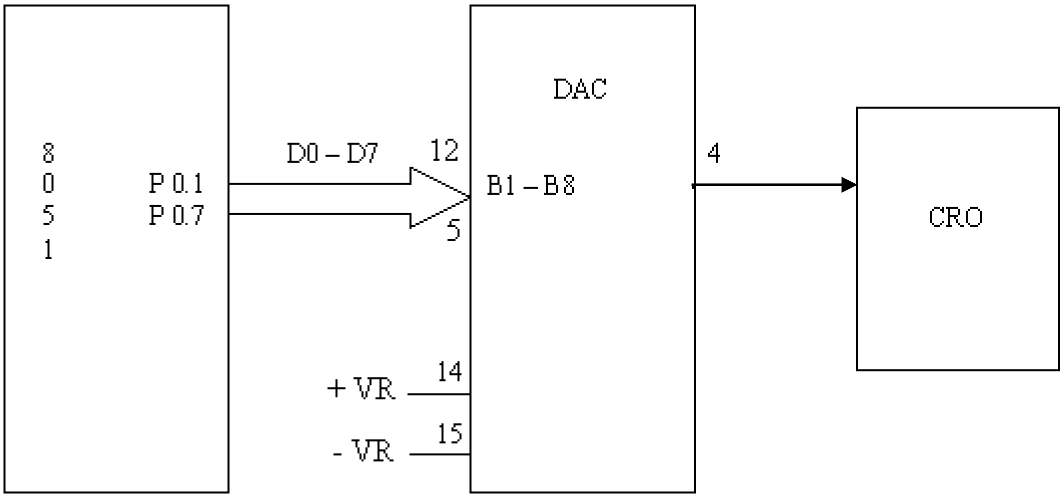



```
#include

<REG51xD2.H>main()
{
  unsigned char i=0;

  P0=0x00;                              /* P0 as Output port */
  while(1)
  {
    for(i=0;i<0xff;i++)                        /* Generate ON pulse */
     P0 =i;
          for(i=0xfe;i>0x00;i--)               /* Generate OFF pulse */
            P0 =i;
  }
}
```

Date:

11(d) Program for dual DAC interfacing to generate sine waveform.

Circuit Diagram:



,

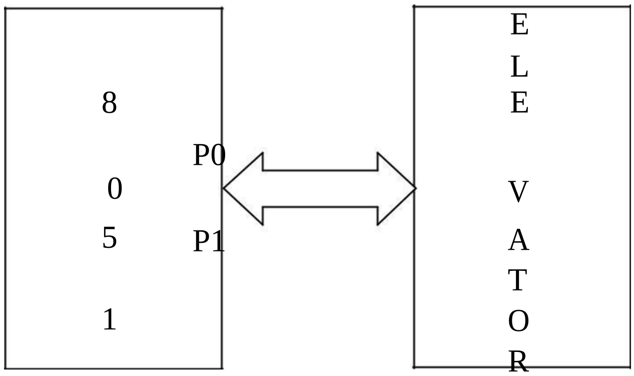
```
#include <RE51xD2.H>

void main( )
{
unsigned char i,
wave[36]={128,148,171,192,209,225,238,245,253,255,253,
245,238,225,209,192,171,128,104,82,64,43,28,15,07,01,00,01,07,15,28,43,64,82,104
};
P0 = 0x00;
while(1)
{
for (i==0; i<36; i++)
P0= wave[i];
}
}
```

12. External ADC and temperature control interface.

```
# include <at89c51xd2.h>
#include<intrins.h>
#include "lcd.h"
Unsigned int Adc;
unsigned char Low_adc,High_adc,relay; read_adc()

{

unsigned char status;
P2_3 = 1 ; // Start conversion of ADC
status = P1; //Read status of ADC
while((status & 0x01) != 0x01)

{

status = P1;

}

P2_2 =0;                // Enable outputs
P2_0 =0;               // Activate B1 to B8outputs
Low_adc =P0;           // Read lower byte of ADC and place in R0

P2_0 =1;               // Deactivate B1 to B8 outputs

P2_1 =0;               // Activate B9 to B12 and POL, over range
outputs High_adc=P0;// Read higher byte of ADC High_adc =
High_adc&0x0F;

P2_1 =1;               // deactivate B9 to B12 and POL, over range outputs

P2_2 =1;               // Disable outputs

P2_3 =0;                // Stop conversion of ADC

}
main()
{
 float  Temp,Vol,Res;
 unsigned char Temp1;
 unsigned charTemp2,Temp3;
P0 = 0xFF ; // Make port 0 as input
P2 = 0xFF ; // Make port 2 as high now the relay is on.
P1_1 = 0 ; // switch OFF relay
P2_3 = 0 ; // STOP conversion of ADC

relay = 10;
```

```
while(1)

{
read_adc(); //Read ADC
Adc = High_adc;
Adc<<= 8;
Adc = Adc | Low_adc;
if( (Adc> 0x656) && (relay!=0))       //IF greater than 0x0656 Switch OFFrelay

{

ClrLcd();
WriteString("RELAY OFF");
P1_1 = 0 ;
relay = 0;

}

else if ( (Adc< 0x5b9) &&(relay!=1))          //IF less than 0x05B9 Switch ONrelay

{

ClrLcd();
WriteString("RELAY ON");
P1_1 = 1 ;
relay = 1;

}

Vol =-((Adc/10)*0.000488); //voltage before amplifier
Res =((100*(1.8-Vol)-100*Vol)*100) /(100*Vol + 100*(1.8+Vol));
                                                          //Resistance Value

Res = Res - 100;
Temp = Res/ 0.384;
Temp1 = Temp;
Temp2 = 0x30 + (Temp1 / 0x0A);
Temp3 = 0x30 + (Temp1 % 0x0A);

GotoXY(0,1);

WriteString("Temperature ");
WriteChar(Temp2);
WriteChar(Temp3);
WriteString("'C");

}
}
```

13. Program for Elevator interface.



Theory:

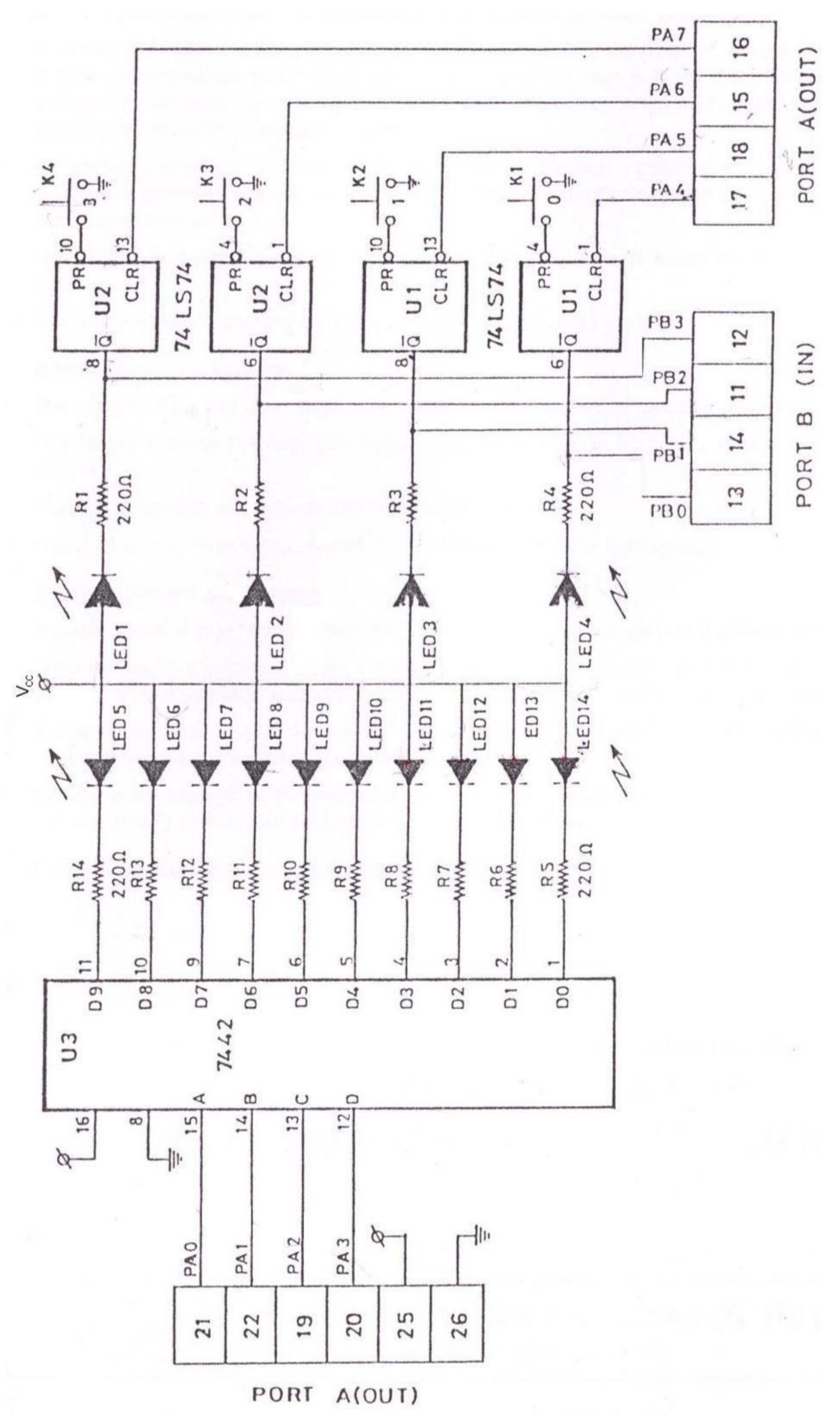
The operation of the elevator is as follows:

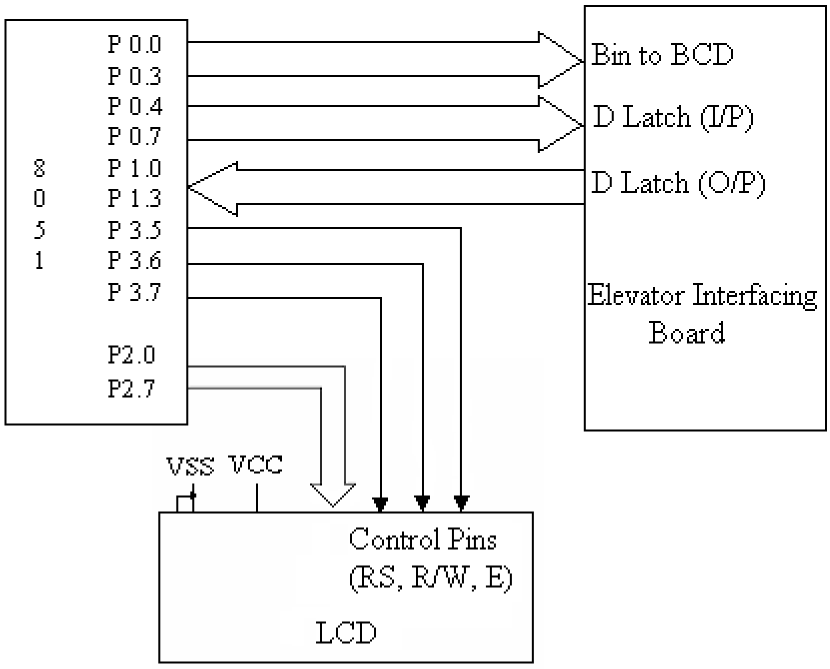
- Initially, the elevator is at ground floor.
- When the elevator reaches any floor, it stays at that floor until a request from another floor is made. When such a request is detected, it moves to that floor.
- The floor request are scanned in fixed order i.e., floors 0, 1, 2 and 3.

This interface simulates the control and operation of an elevator. Four floors assumed and for each floor a key and corresponding LED indicator are provided to serve as request buttons and request status indicator. The elevator itself is represented by a column of ten LEDs. The motion of elevator can be simulated by turning on successive LEDs one at a time. Te delay between turning off one LED and turning on the next LED can simulate the “speed” of the elevator. User can read the request status information through one port, reset the request indicators through another port and control the elevator (LED column) through another port.

Description of the Circuit

This interface has four keys, marked 0, 1, 2, and 3 representing the request buttons at the four floors. Pressing of key causes a corresponding Flip-Flop to be set. The outputs of the four Flip-flops can be read through port B (PBO, PBI, PB2 and PB3). Also, the status of these signals is reflected by a setoff 4 LEDs. The Flip-Flop can be rest (LEDs are cleared) through port A(PA54, PA5, PA6, and PA7). A column of 10 LEDs, representing the elevator can be controlled through Port A (PA0, PA1, PA2 and PA3). These port lines are fed to the inputs of the decoder 7442 whose outputs are used to control the on/off states of the LEDs which simulate the motion of the elevator.






```
#include <REG51D2.H>

void delay(unsigned int);
main()
{
 unsigned char Flr[9] = {0xff,0x00,0x03,0xff,0x06,0xff,0xff,0xff,0x09};
 unsigned char FClr[9] = {0xff,0x0E0,0x0D3,0xff,0x0B6,0xff,0xff,0xff,0x79};
 unsigned char ReqFlr,CurFlr = 0x01,i,j;
 P0 = 0x00;
 P0 = 0x0f0;
 while(1)
 {
        P1 = 0x0f;
        ReqFlr = P1 | 0x0f0;
        while(ReqFlr == 0x0ff)
          ReqFlr = P1 |0x0f0;          /* Read Request Floor from P1 */
        ReqFlr =~ReqFlr;
        if(CurFlr==ReqFlr)          /* If Request floor is equal to Current Floor*/
        {
          P0=FClr[CurFlr];               /* Clear Floor Indicator */
          continue;                             /* Go up to read again*/
        }

        else if(CurFlr>ReqFlr)          /* If Current floor is > request floor*/
```

```
        {

  i = Flr[CurFlr]-Flr[ReqFlr];      /* Get the no of floors to travel */
          j =Flr[CurFlr];
          for(;i>0;i--)                      /* Move the indicator down*/

          {
           P0 = 0x0f0|j;
           j--;
           delay(50000);
          }
        }
        else                          /* If Current floor is < request floor*/
        {
          i = Flr[ReqFlr] - Flr[CurFlr]; /* Get the no of floors to travel*/
          j =Flr[CurFlr];
          for(;i>0;i--)                      /* Move the indicator Up*/
          {
            P0 = 0x0f0 | j;
                j++;
            delay(50000);
          }

        }
        CurFlr=ReqFlr;                       /* Update Current floor*/
        P0=FClr[CurFlr];                     /* Clear the indicator*/
 }
}

void delay(unsigned int x)
{
  for(;x>0;x--);
          }
```

Question bank

Part A:

1. Write an assembly language program to transferN=_bytes of data from location A:______h to locationB:__________________h (without overlap) using8051
2. Write an assembly language program to exchange N=__________bytes of data from location A:_________h to locationB:__________h (without overlap) using8051
3. Write an assembly language program to sort an array of N=________h bytes of data in ascending /descending order using8051
4. Write an assembly language program to find largest number in a given array of ‘N’ elements using 8051 , where ,N=______________h
5. Write an assembly language program to perform addition of two 16 bit numbers using8051
6. Write an assembly language program to perform subtraction of two 16 bit numbers using8051
7. Write an assembly language program to perform multiplication of two 16 bit numbers using8051
8. Write an assembly language program to perform division of two 16 bit numbers using8051
9. Write an assembly language program to find square of a given numbers using8051
10. Write an assembly language program to find cube of a given numbers using8051
11. Write an assembly language program to count numbers fromN=____h to N=____h (Up counter/Down counter) using8051
12. Write an assembly language program to implement(display) an eight bit Up /Down binary(hex) counter on watch window using8051
13. Write an assembly language program to count number of one’s and zero’s in given 8 bit number using8051
14. Write an assembly language program to exhibit the usage of call and return instruction
15. Write an assembly language program to convert an 8 bit BCD number to ASCII using8051
16. Write an assembly language program to convert ASCII to an 8 bit BCD number to using8051
17. Write an assembly language program to convert ASCII to decimal using8051
18. Write an assembly language program to convert decimal to ASCII using8051
19. Write an assembly language program to convert Hexa decimal to decimal using 8051
20. Write an assembly language program to convert decimal to Hexa decimal using 8051
21. Write an assembly language program to generatedelayof_______seconds using 8051

Part B(using C program)

A. Write a program for stepper motor interface with8051
B. Write a program for DC motor interface with 8051 and control itsspeed
C. Write a program to interface LCD panel and hexa keypad to8051
D. Write a program for dual DAC interfacing to generate sinewave
E. Write a program for dual DAC interfacing to generate squarewave
F. Write a program for dual DAC interfacing to generate triangularwave
G. Write a program for dual DAC interfacing to generate rampwave
H. Write a program to interface ADC with8051
I. Write a program for elevator interface with8051

Viva Questions

1. What do you mean by Embedded System? Giveexamples.
2. Why are embedded Systemsuseful?
3. What are the segments of EmbeddedSystem?
4. What is EmbeddedController?
5. What isMicrocontroller?
6. List out the differences between Microcontroller andMicroprocessor.
7. How are Microcontrollers more suitable than Microprocessor for Real TimeApplications?
8. What are the General Features ofMicrocontroller?
9. Explain briefly the classification ofMicrocontroller.
10. Explain briefly the EmbeddedTools.
11. Explain the general features of 8051Microcontroller.
12. How many pins the 8051has?
13. Differentiate between Program Memory and DataMemory.
14. What is the size of the Program and Data memory?
15. Write a note on internal RAM. What is the necessity of register banks?Explain.
16. How many address lines are required to address 4K of memory? Show the necessary calculations.
17. What is the function ofaccumulator?
18. What are SFR’s? Explain briefly.
19. What is the program counter? What is itsuse?
20. What is the size of thePC?
21. What is a stack pointer(SP)?
22. What is the size ofSP?
23. What is the PSW? And briefly describe the function of itsfields.
24. What is the difference between PC andDPTR?
25. What is the difference between PC andSP?
26. What is ALE? Explain the functions of the ALE in8051.
27. Describe the 8051 oscillator andclock.
28. What are the disadvantages of the ceramicresonator?
29. What is the function of the capacitors in the oscillatorcircuit?
30. Show with an example, how the time taken to execute an instruction can becalculated.
31. What is the Data Pointer register? What is its use in the8051?
32. Explain how the 8051 implement the Harvard Architecture?
33. Explain briefly the difference between the Von Neumann andtheHarvard Architecture.
34. Describe in detail how the register banks areorganized.
35. What are the bit addressable registers and what is theneed?
36. What is the need for the general purpose RAMarea?
37. Write a note on the Stack and the StackPointer.
38. Why should the stack be placed high in internalRAM?
39. Explain briefly how internal and external ROM getsaccessed.
40. What are the different addressing modes supported by 8051 Microcontroller?
41. Explain the Immediate Addressing Mode.
42. Explain the Register AddressingMode.
43. Explain the Direct AddressingMode.
44. Explain the Indirect AddressingMode.
45. Explain the Code AddressingMode.
46. Explain in detail the Functional Classification of 8051 Instructionset
47. What are the instructions used to operatestack?
48. What are Accumulator specific transferinstructions?
49. What is the difference between INC and ADDinstructions?
50. What is the difference between DEC and SUBB instructions?
51. What is the use of OV flag in MUL and DIVinstructions?
52. What are single and two operandinstructions?
53. Explain Unconditional and Conditional JMP and CALL instructions.
54. Explain the different types of RETURNinstructions.
55. What is a softwaredelay?
56. What are the factors to be considered while deciding a softwaredelay?
57. What is a Machinecycle?

58. What is aState?
59. Explain the need for Hardware Timers and Counters?
60. Give a brief introduction onTimers/Counter.
61. What is the difference between Timer and Counteroperation?
62. How many Timers are there in8051?
63. What are the three functions ofTimers?
64. What are the different modes of operation oftimer/counter?
65. Give a brief introduction on the variousModes.
66. What is the count rate of timeroperation?
67. What is the difference between mode 0 and mode1?
68. What is the difference Modes 0,1,2 and 3?
69. How do you differentiate between Timers andCounters?
70. Explain the function of the TMOD register and its variousfields?
71. How do you control the timer/counteroperation?
72. What is the function of TF0/TF1bit
73. Explain the function of the TCON register and its variousfields?
74. Explain how the Timer/Counter Interrupts work.
75. Explain how the 8051 counts using Timers andCounters.
76. Explain Counting operation in detail in the 8051.
77. Explain why there is limit to the maximum external frequency that can becounted.
78. What’s the benefit of the auto-reloadmode?
79. Write a short note on Serial and Parallel communication and highlight their advantagesand disadvantages.
80. Explain Synchronous Serial DataCommunication.
81. Explain Asynchronous Serial DataCommunication.
82. Explain Simplex data transmission withexamples.
83. Explain Half Duplex data transmission withexamples.
84. Explain Full Duplex data transmission withexamples.
85. What is Baudrate?
86. What is aModem?
87. What are the various registers and pins in the 8051 required for Serial communication? Explainbriefly.
88. Explain SCON register and the variousfields.
89. Explain serial communication in general (synchronous and asynchronous). Also explain the use of the paritybit.
90. Explain the function of the PCON register during serial datacommunication.
91. How the Serial data interrupts aregenerated?
92. How is data transmitted serially in the 8051? Explainbriefly.
93. How is data received serially in the 8051? Explainbriefly.
94. What are the various modes of Serial Data Transmission? Explain each mode briefly.
95. Explain with a timing diagram the shift register mode in the8051.
96. What is the use of the serial communication mode 0 in the8051?
97. Explain in detail the Serial Data Mode 1 in the8051.
98. Explain how the Baud rate is calculated for the Serial Data Mode1.
99. How is the Baud rate for the Multiprocessor communication Modecalculated?
100. Explain in detail the Multiprocessor communication Mode in the8051.
101. Explainthesignificanceofthe9thbitintheMultiprocessorcommunication Mode.
102. Explain the Serial data mode 3 in the8051.
103. What are interrupts and how are they useful in Real TimeProgramming?
104. Briefly describe the Interrupt structure in the8051.
105. Explain about vectored and non-vectored interrupts ingeneral.
106. What are the five interrupts provided in the8051?
107. What are the three registers that control and operate the interrupts in8051?
108. DescribetheInterruptEnable(IE)specialfunctionregisteranditsvarious bits.
109. Describe the Interrupt Priority (IP) special function register and itsneed.
110. Explain in detail how the Timer Flag interrupts aregenerated.
111. Explain in detail how the Serial Flag interrupt isgenerated.
112. Explain in detail how the External Flag interrupts aregenerated.

113. What happens when a high logic is applied on the Resetpin?
114. Why the Reset interrupt is called a non-maskableinterrupt?
115. Why do we require a resetpin?
116. How can you enable/disable some or all theinterrupts?
117. Explainhowinterruptprioritiesareset?Andhowinterruptsthatoccur simultaneously arehandled.
118. WhatEventscantriggerinterrupts,andwheredotheygoaftergetting triggered?
119. What are the actions taken when an InterruptOccurs?
110. What are Software generated interrupts and how are they generated?
111.What is RS232 and MAX232?
112. What is the function of RS and E pins in anLCD?
113. What is the use of R/W pin in an LCD?
114.What is the significance of DAinstruction?
115.What is packed and unpackedBCD?
116. What is the difference between CY and OV flag?
117.When will the OV flag be set?
118.What is an ASCII code?

Instruction set

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS (Continued)				
INC	DPTR	Increment Data Pointer	1	24
MUL	AB	Multiply A & B	1	48
DIV	AB	Divide A by B	1	48
DA	A	Decimal Adjust Accumulator	1	12
LOGICAL OPERATIONS				
ANL	A,Rn	AND Register to Accumulator	1	12
ANL	A,direct	AND direct byte to Accumulator	2	12
ANL	A,@Ri	AND indirect RAM to Accumulator	1	12
ANL	A,#data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
ANL	direct,#data	AND immediate data to direct byte	3	24
ORL	A,Rn	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@Ri	OR indirect RAM to Accumulator	1	12
ORL	A,#data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct,#data	OR immediate data to direct byte	3	24
XRL	A,Rn	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL	A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12

Mnemonic		Description	Byte	Oscillator Period
LOGICAL OPERATIONS (Continued)				
RL	A	Rotate Accumulator Left	1	12
RLC	A	Rotate Accumulator Left through the Carry	1	12
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER				
MOV	A,Rn	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@Ri	Move indirect RAM to Accumulator	1	12
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	Rn,A	Move Accumulator to register	1	12
MOV	Rn,direct	Move direct byte to register	2	24
MOV	Rn,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,Rn	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@Ri	Move indirect RAM to direct byte	2	24
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@Ri,A	Move Accumulator to indirect RAM	1	12

Mnemonic		Description	Byte	Oscillator Period
DATA TRANSFER (Continued)				
MOV	@Ri,direct	Move direct byte to indirect RAM	2	24
MOV	@Ri,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
MOVX	A,@Ri	Move External RAM (8-bit addr) to Acc	1	24
MOVX	A,@DPTR	Move External RAM (16-bit addr) to Acc	1	24
MOVX	@Ri,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,Rn	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@Ri	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@Ri	Exchange low-order Digit indirect RAM with Acc	1	12

Mnemonic		Description	Byte	Oscillator Period
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to CARRY	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is Not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24
PROGRAM BRANCHING				
ACALL	addr11	Absolute Subroutine Call	2	24
LCALL	addr16	Long Subroutine Call	3	24
RET		Return from Subroutine	1	24
RETI		Return from interrupt	1	24
AJMP	addr11	Absolute Jump	2	24
LJMP	addr16	Long Jump	3	24
SJMP	rel	Short Jump (relative addr)	2	24

Mnemonic		Description	Byte	Oscillator Period
PROGRAM BRANCHING (Continued)				
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is Zero	2	24
JNZ	rel	Jump if Accumulator is Not Zero	2	24
CJNE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3	24
CJNE	A,#data,rel	Compare immediate to Acc and Jump if Not Equal	3	24

Mnemonic		Description	Byte	Oscillator Period
PROGRAM BRANCHING (Continued)				
CJNE	Rn,#data,rel	Compare immediate to register and Jump if Not Equal	3	24
CJNE	@Ri,#data,rel	Compare immediate to indirect and Jump if Not Equal	3	24
DJNZ	Rn,rel	Decrement register and Jump if Not Zero	2	24
DJNZ	direct,rel	Decrement direct byte and Jump if Not Zero	3	24
NOP		No Operation	1	12

Additional programs

(a) Logical operations:

```
    org 8000h
    mov r0, #0fh
    mov r1, #f0h
    mov r2, #66h
// And operation
    mov a,
    #ffhanl a, r0
    mov r3, a
// Oroperation
    mov a,
    #ffhorl a, r1
    mov r4, a
// Xor operation
    mov a, 03h
    mov a,
    #ffhxrl a, r2
    mov r5, a
    lcall 0003h
    end
```

Output:

b) Swap and rotateinstructions

```
   org9000h
// clear register A
    mov a, #0fh
    clra
    mov r0, a
//swap nibbles of register
    Amov a, #56h
    swap a
    mov r1, a
// Complement the bit of register A
    mov a, #66h
    cpl a
    mov r2, a
// Rotate the register contents towards right
    mov a, #63h
```

```
    rr a
    xrl a, r
    mov r3, a
// Rotate the register contents towards left
    mov a, #43h
    rl  axrl
    a, r
    mov r4, a
    lcall 0003h
    end
```

Output:

c) Bit manipulationoperations:

```
    org 9000h
    mov a, #0ffh
    clr c
// clear the carry flag
    anl c, acc.7
    mov r0, a
    setb c
// set the carry flag
    mov a, #00h
    orl c, acc.5
    mov r1, a
    mov a, #0ffh
    cplacc, 3
    mov r2, a
    lcall 0003h
    end
```

Output:

d) Program to generate a resultant byte whose 7th bit is given by b7=b2+b5+b6

```
        mov a, #86h
        mov r2, a
        anl a, #04
        rrca
        rrca
        rrca
        mov r3, a
        mov a, r2
        anla,#20
        rlca
        rlca
        mov r4, a
        mov a, r2
        anla,#40
        rlca
        orl a, r3
        orl a, r4
        mov p1,a
here:   sjmphere
        end
```

Output :

e) Program for subtraction of two 8 bit no's

```
Movr0,#12h        ; get first no inro
 Mova, r0         ; copy toaccumulator
 Movr1,#08h       ; get second no
Subba, r6         ; subtract accumulator with registerr6
Mov20h, a         ; store the Output
end
```

Output:acc=12h
r6=08h

D: 20h=4h