Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(**NAAC Accredited & ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572216. Karnataka.

Department of Electrical & Electronics Engineering

BEE403-IPCC LAB MANUAL

(2025-2026)

Microcontroller

Practical component of IPCC

B.E. - IV Semester

Name: ____

U S N: ____

Batch: ____

Semester & section: ____________________________

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

Department of Electrical & Electronics Engineering

Microcontroller Lab

Version 4.0

Feb- 2026

Prepared by:
Mrs. Rekha D K
Assistant Professor

Reviewed& Approved by:
Mrs. Radha B N
Assistant Professor& Head,
Dept. of EEE

Channabasaveshwara Institute of Technology

(AffiliatedtoVTU,Belgaum&Approvd by AICTE, NewDelhi)

(NAAC Accredited & ISO 9001:2015 Certified Institution)

NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

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

INSTITUTION MISSION
1. To provide high quality technical and professionally relevant education in a diverse learning environment. 2. To provide the values that prepare students to lead their lives with personal integrity, professional ethics and civic responsibility in a global society. 3. To prepare the next generation of skilled professionals to successfully compete in the diverse global market. 4. 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. 5. To offer a wide variety of off-campus education and training programmes to individuals and groups. 6. To stimulate collaborative efforts with industries, universities, government and professional societies. 7. To facilitate public understanding of technical issues and achieve excellence in the operations of the institute.

QUALITY POLICY

Our organization delights customers (students, parents and society) by providing value added quality education to meet the national and international requirements. We also provide necessary steps to train the students for placement and continue to improve our methods of education to the students through effective quality management system, quality policy and quality objectives.

Partnering in Academic Excellence

Channabasaveshwara Institute of Technology

(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)

(NAAC Accredited & ISO 9001:2015 Certified Institution)

NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

DEPARTMENT VISION
To establish a centre of excellence in Electrical and Electronics Engineering education and to foster the development of technically proficient professionals in Electrical Science and related fields while instilling a strong sense of ethics to serve the society efficiently.

DEPARTMENT MISSION	
M1	To provide competent human resources, and to ensure that our students receive top-notch education and mentorship, enabling them to excel in electrical and electronics engineering and allied fields.
M2	To provide quality infrastructure, and to create an environment conducive to innovative learning and research, empowering our students to explore the frontiers of Electrical Sciences and related disciplines.
M3	To foster strong collaborations with industry and research institutions, and to facilitate the exchange of knowledge and ideas, allowing our students and faculty to remain at the cutting edge of technological advancements and practical applications in the field..
M4	To emphasize social responsibility and professional ethics in our curriculum and community engagement, and to prepare our graduates to be conscientious leaders who use their expertise to benefit society, making a positive impact through their work in Electrical Sciences and allied fields..

SYLLABUS

Sl. No.		Experiments
1	Assembly programs	Arithmetic instructions: Addition, subtraction, multiplication and division. Square And cube operations for 16 bit numbers.
2	Assembly programs	Data transfer – Program for block data movement, sorting, exchanging, finding largest element in an array
3	Assembly programs	Up/Down BCD/ Binary Counters
4	Assembly programs	Boolean and logical instructions (bit manipulation).
5	Assembly programs	Code conversion programs – BCD to ASCII, ASCII to BCD, ASCII to decimal, Decimal to ASCII, Hexa decimal to and Decimal to Hexa.
6	Assembly programs	
7		DC motor interface for direction and speed control using PWM
8	Interfacing programs	Stepper motor interface for direction and speed control.
9	Interfacing programs	Alphanumerical LCD panel interface.
10	Interfacing programs	Generate different waveforms: Sine, Square, Triangular, Ramp using DAC Interface.

Note: For the **experiments 1 to 6, 8051 assembly programming** is to be used and Single chip solution for **interfacing 8051 is to be done with C Programs** for the **experiments 7 to 10**

Course outcomes:

At the end of the 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.
- Generate different waveforms using DAC interface.
- Work with a small team to carryout experiments using microcontroller concepts and prepare reports that present lab work.

INDEX PAGE

Sl.No.	Name of the Program /Experiment	Date			Manual marks (10)	Record marks (05)	Signature (Student)	Signature (faculty)
		Conduction	Repetition	Submission of Record				
Average								

Note: If the student fail to attend the regular lab, the program execution has to be completed in the same week. Otherwise the evaluation will be done for 50% of the maximum marks

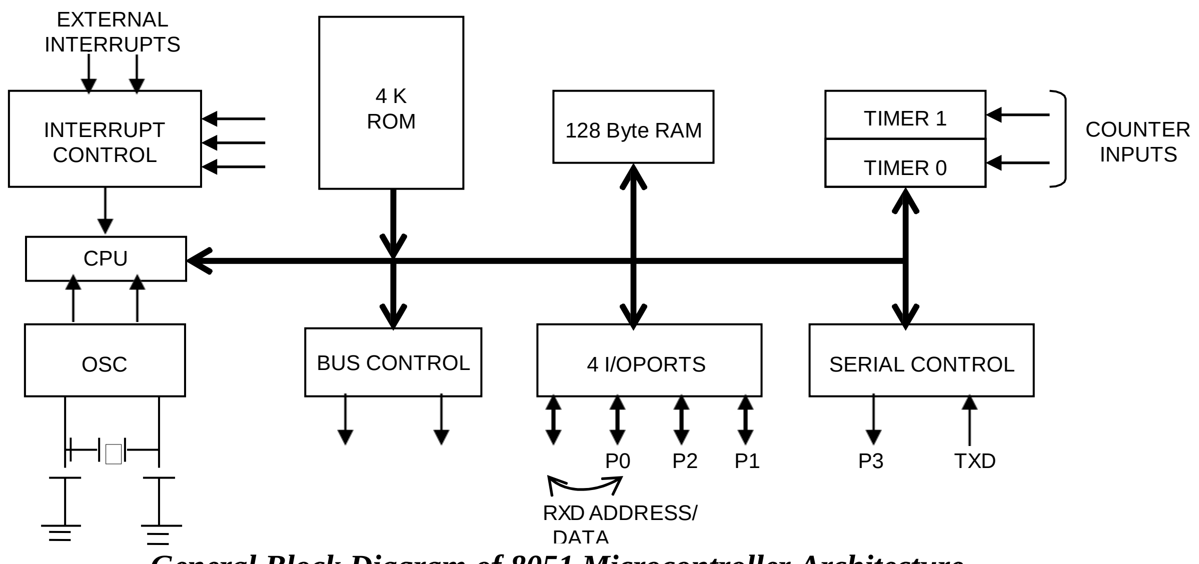
INTRODUCTION

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 core are:

- Operates with single Power Supply+5V.
- 8-bit CPU optimized for control applications.
- 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 ore PROM).
- 128 bytes of On-Chip Data Memory (Internal RAM):
    - Four Register Banks, each containing 8 registers (R0 to R7) (Total 32registers).
    - 16 bytes of bit addressable memory.
    - 80 bytes of general-purpose data memory (Scratch Pad Area).
- Special Function Registers (SFR) to configure/operate microcontroller.
- 32 bit bi-directional I/O Lines (4 ports P0 toP3).
- Two 16-bit timers/counters (T0 and T1).
- Full duplex UART (Universal Asynchronous Receiver/Transmitter).
- 6-source/5-vector interrupts (2 external and 3 internal) with two priority levels.
- On-Chip oscillator and clock circuitry.

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:

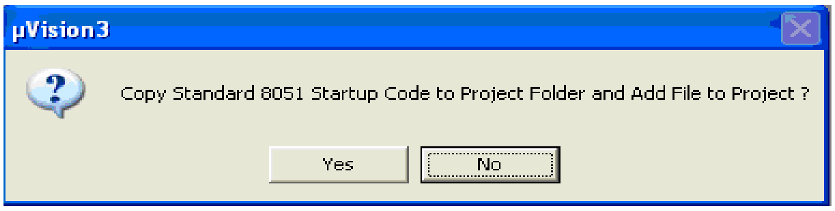
SM0	**SM1**	**Mode**	**Description**	**Baud Rate**
0	**0**	**0**	shift register	f osc / 12
0	**1**	**1**	8-bit UART	Variable
1	**0**	**2**	9-bit UART	f osc / 64 or fosc /32
1	**1**	**3**	9-bit UART	variable

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 the desktop.

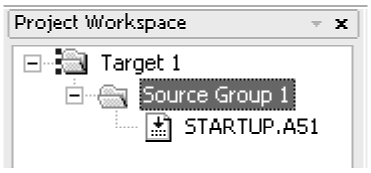
2. Close any previous projects that were opened using – **Project ->Close**.
3. Start **Project – New Project,** and select the CPU from the device database (Database-Atmel- AT89C51ED2 or AT89C51RD2 as per the board).On clicking **'OK'**, the following option is displayed. Choose **'No'**.



Copy Standard 8051 Startup Code to Project Folder and Add File to Project ?

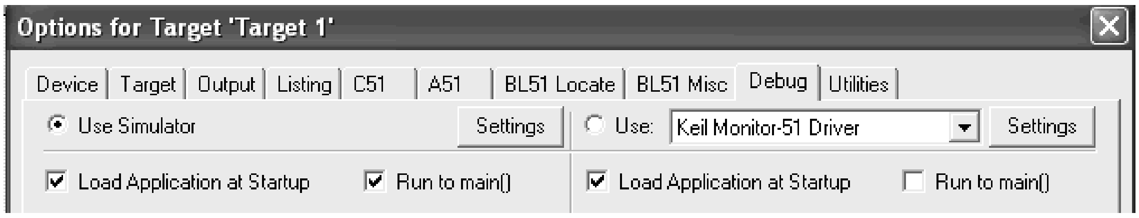
4. Create a source file (using **File->New**), type in the assembly or C program and save this (filename**.asm**/filename**'s**) and add this source file to the project using either one of the following two methods. (i)**Project->Manage->Components, Environment Books->add files->**browse to the required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

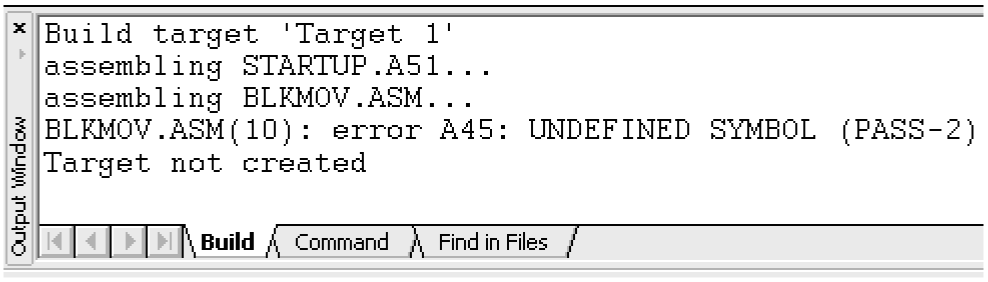
   – Debug – use either Simulator / Keil Monitor- 51 driver.



● Use Simulator ○ Use: Keil Monitor-51 Driver
☑ Load Application at Startup ☑ Run to main() ☑ Load Application at Startup ☐ Run to main()

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.



```
Build target 'Target 1'
assembling STARTUP.A51...
assembling BLKMOV.ASM...
BLKMOV.ASM(10): error A45: UNDEFINED SYMBOL (PASS-2)
Target not created
```

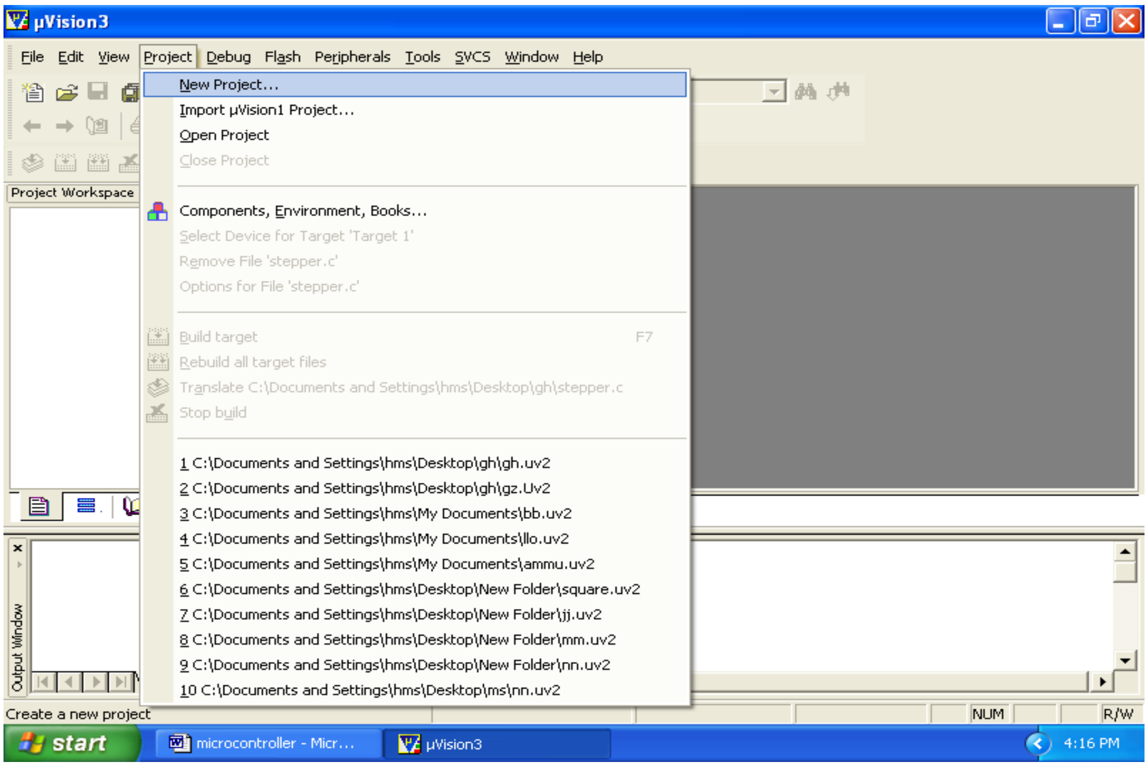
8. NowusercanenterintoDebugmodewith**Debug-Start/StopDebugsession** dialog. Or by clicking in the icon.

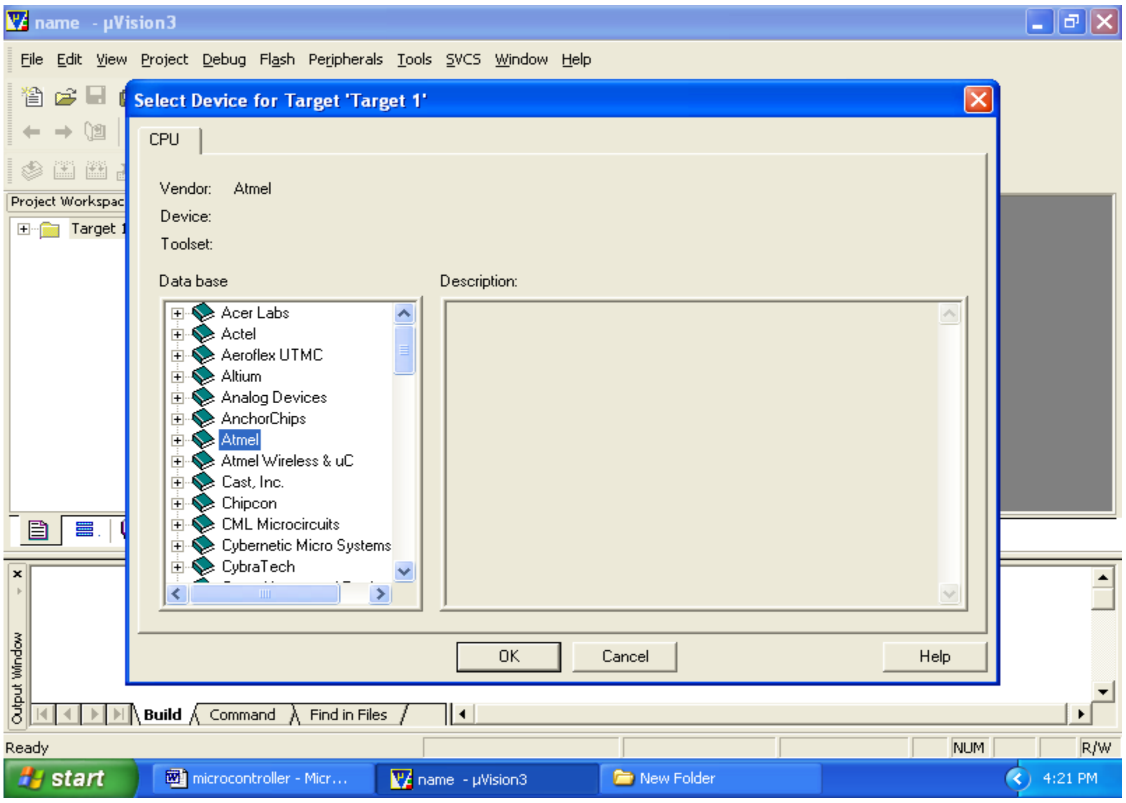
9. The program is run using the **Debug-Run** command & halted using **Debug-Stop Running.** Also the (reset, run, halt) icons can be used. Additional icons are (step, step over, and step into, run tillcursor).

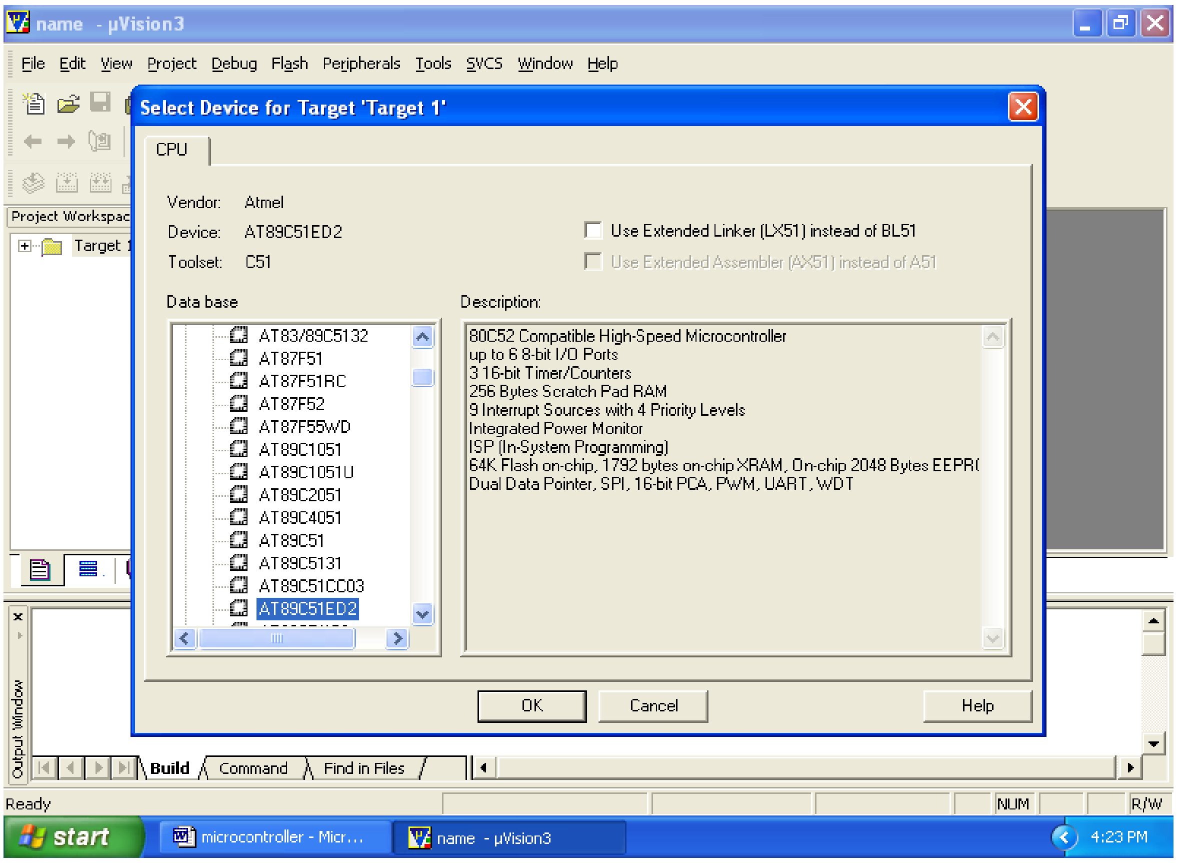
10. IfitisaninterfaceprogramtheoutputscanbeseenontheLCD,CRO,motor,led status, etc. If it is a part-A program, the appropriate memory window is opened using View -> memory window (for data RAM & XRAM locations), Watch window (for timer program), serial window, etc.

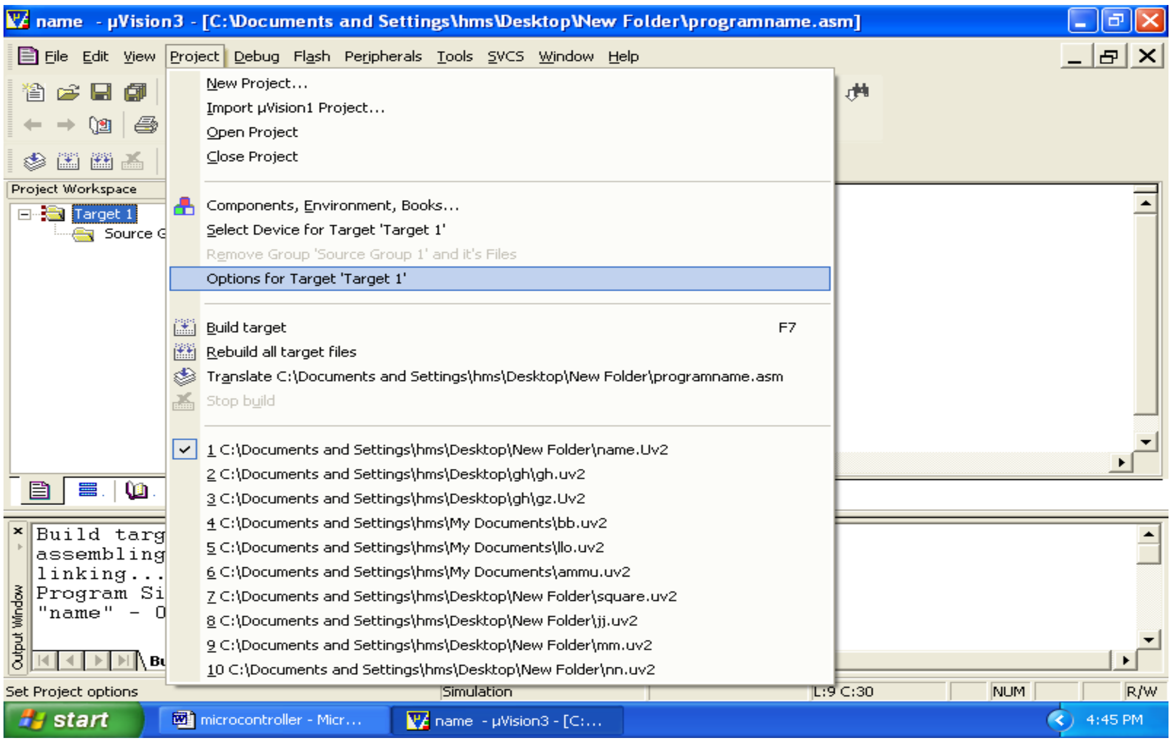
Note: To access data RAM area type address as D: 0020h. Similarly to access the DPTR region (XRAM-present on chip in AT89C51ED2) say 9000h location type in X: 09000H.

EXECUTION STEPS using KEIL µ vision:

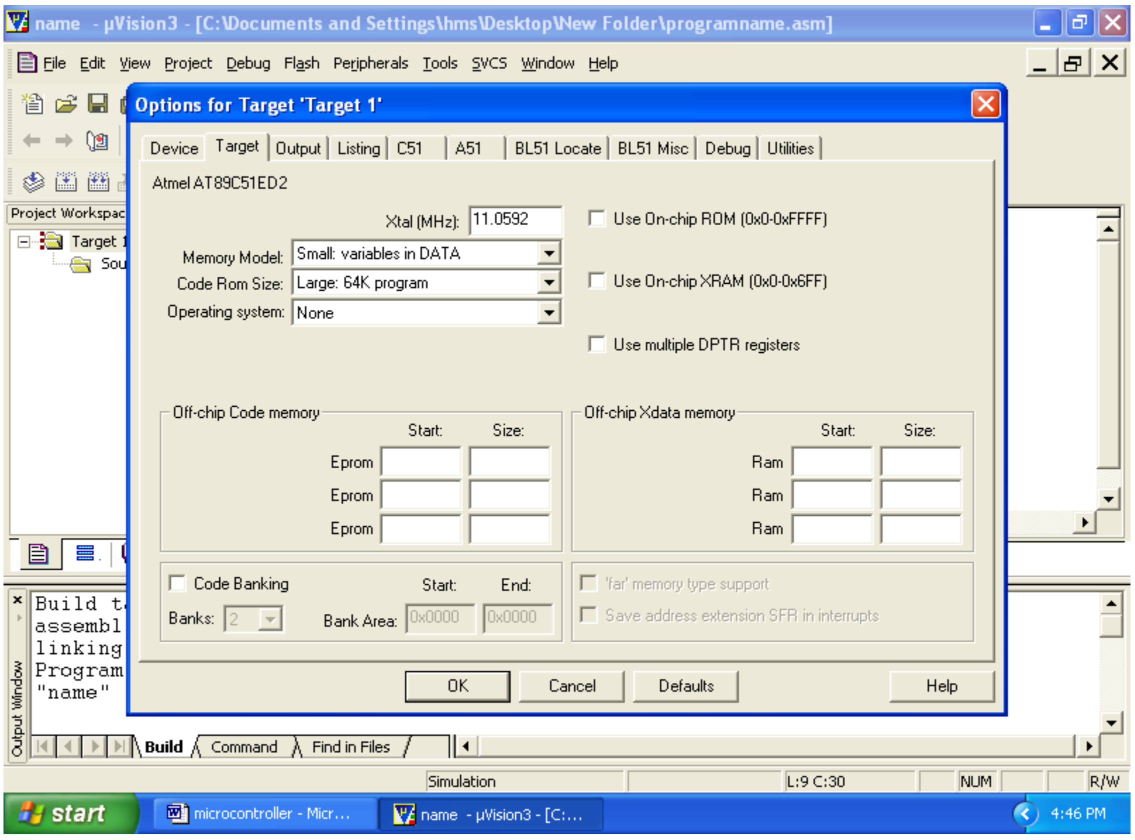

µVision3
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
New Project…
Import µVision1 Project…
Open Project
Close Project
Components, Environment, Books…
Select Device for Target 'Target 1'
Remove File 'stepper.c'
Options for File 'stepper.c'
Build target F7
Rebuild all target files
Translate C:\Documents and Settings\hms\Desktop\gh\stepper.c
Stop build
1 C:\Documents and Settings\hms\Desktop\gh\gh.uv2
2 C:\Documents and Settings\hms\Desktop\gh\gz.Uv2
3 C:\Documents and Settings\hms\My Documents\bb.uv2
4 C:\Documents and Settings\hms\My Documents\llo.uv2
5 C:\Documents and Settings\hms\My Documents\ammu.uv2
6 C:\Documents and Settings\hms\Desktop\New Folder\square.uv2
7 C:\Documents and Settings\hms\Desktop\New Folder\jj.uv2
8 C:\Documents and Settings\hms\Desktop\New Folder\mm.uv2
9 C:\Documents and Settings\hms\Desktop\New Folder\nn.uv2
10 C:\Documents and Settings\hms\Desktop\ms\nn.uv2
Project Workspace
Output Window
Create a new project
NUM
R/W
start
microcontroller - Micr…
µVision3
4:16 PM




name - µVision3
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Select Device for Target 'Target 1'
CPU
Vendor: Atmel
Device: AT89C51ED2
Toolset: C51
Use Extended Linker (LX51) instead of BL51
Use Extended Assembler (AX51) instead of A51
Data base
AT83/89C5132
AT87F51
AT87F51RC
AT87F52
AT87F55WD
AT89C1051
AT89C1051U
AT89C2051
AT89C4051
AT89C51
AT89C5131
AT89C51CC03
AT89C51ED2
Description:
80C52 Compatible High-Speed Microcontroller
up to 6 8-bit I/O Ports
3 16-bit Timer/Counters
256 Bytes Scratch Pad RAM
9 Interrupt Sources with 4 Priority Levels
Integrated Power Monitor
ISP (In-System Programming)
64K Flash on-chip, 1792 bytes on-chip XRAM, On-chip 2048 Bytes EEPR
Dual Data Pointer, SPI, 16-bit PCA, PWM, UART, WDT
OK
Cancel
Help
Build
Command
Find in Files
Ready
NUM
R/W
start
microcontroller - Micr...
name - µVision3
4:23 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
Source G
Set Project options
Simulation
L:9 C:30
NUM
R/W
start
microcontroller - Micr...
name - µVision3 - [C:...
4:45 PM

```
Build targ
assembling
linking...
Program Si
"name" - 0
```


name - µVision3 - [C:\Documents and Settings\hms\Desktop\New Folder\programname.asm]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Options for Target 'Target 1'
Device Target Output Listing C51 A51 BL51 Locate BL51 Misc Debug Utilities
OK Cancel Defaults Help
Build Command Find in Files
start microcontroller - Micr... name - µVision3 - [C:...
4:46 PM

Atmel AT89C51ED2

Xtal (MHz): 11.0592

Memory Model: Small: variables in DATA

Code Rom Size: Large: 64K program

Operating system: None

☐ Use On-chip ROM (0x0-0xFFFF)

☐ Use On-chip XRAM (0x0-0x6FF)

☐ Use multiple DPTR registers

Off-chip Code memory — Start: Size: — Eprom, Eprom, Eprom

Off-chip Xdata memory — Start: Size: — Ram, Ram, Ram

☐ Code Banking — Start: End:

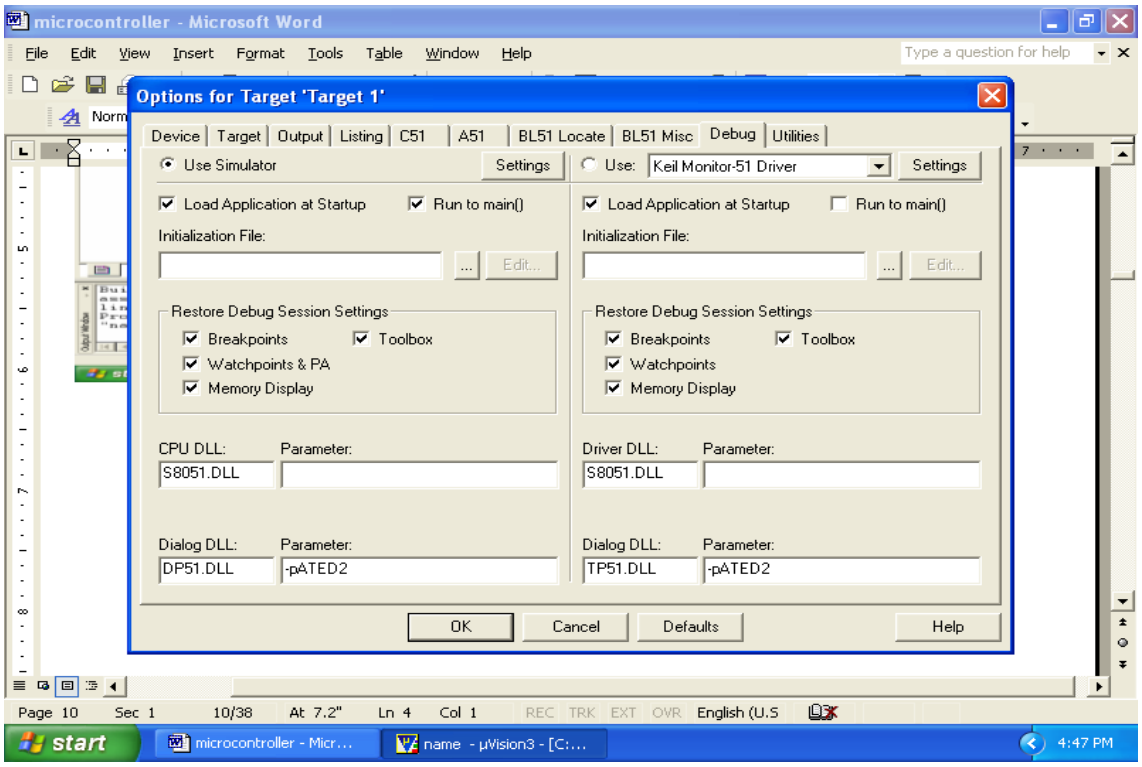
Banks: 2 — Bank Area: 0x0000 0x0000

☐ 'far' memory type support

☐ Save address extension SFR in interrupts

Build target
assembling
linking
Program
"name"

Simulation L:9 C:30 NUM R/W


microcontroller - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Type a question for help
Options for Target 'Target 1'
Device Target Output Listing C51 A51 BL51 Locate BL51 Misc Debug Utilities
Settings
Edit...
OK Cancel Defaults Help
start microcontroller - Micr... name - µVision3 - [C:...
4:47 PM

(●) Use Simulator | ( ) Use: Keil Monitor-51 Driver

☑ Load Application at Startup ☑ Run to main() | ☑ Load Application at Startup ☐ Run to main()

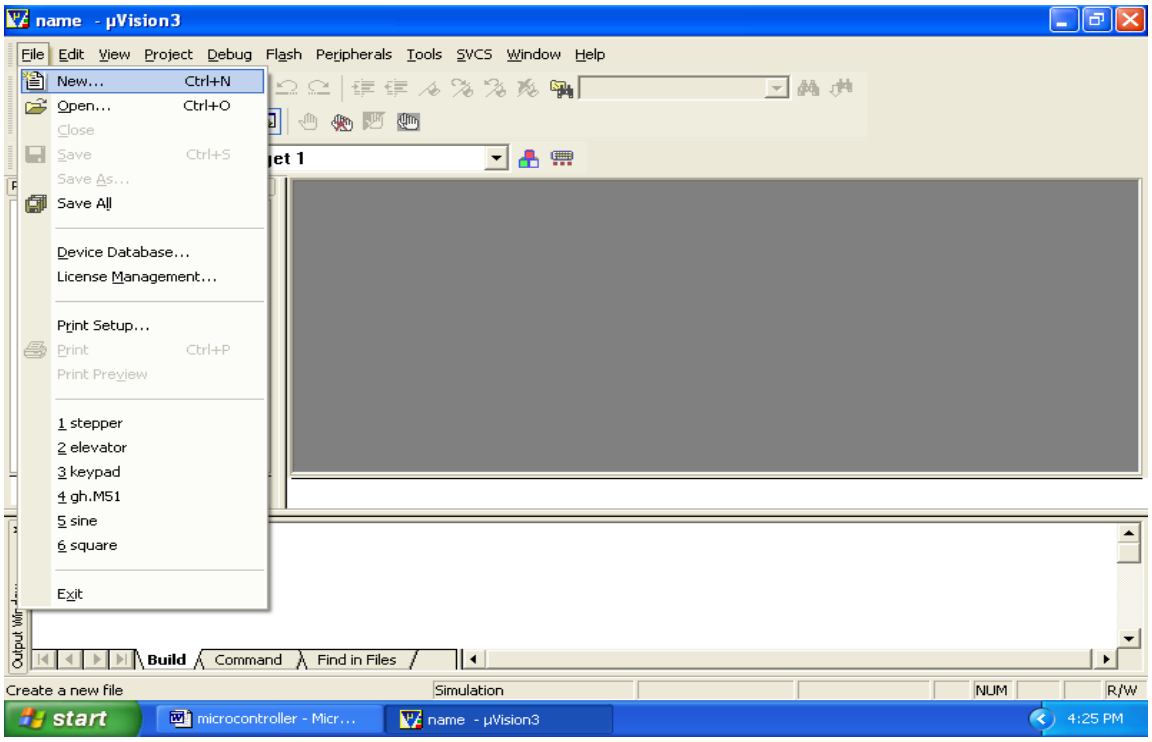
Initialization File: | Initialization File:

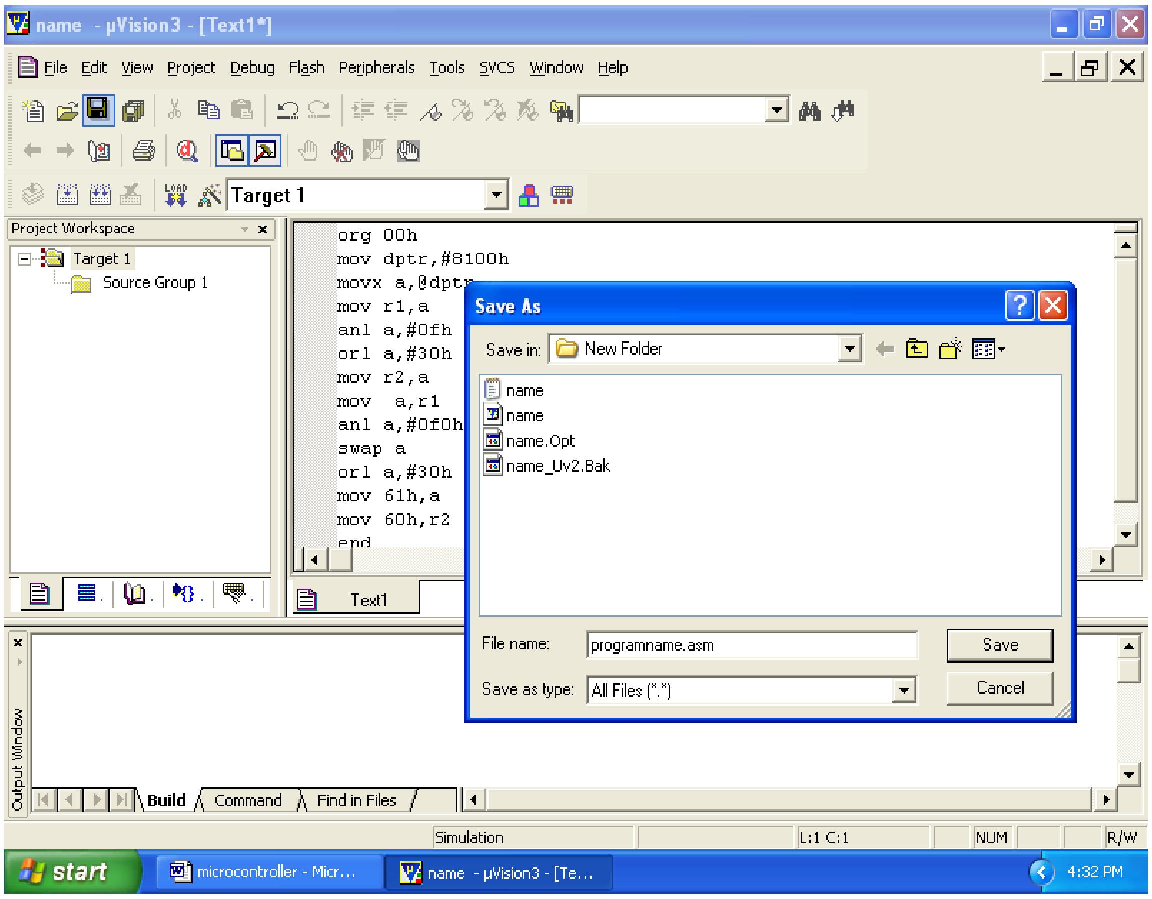
Restore Debug Session Settings: ☑ Breakpoints ☑ Toolbox ☑ Watchpoints & PA ☑ Memory Display | Restore Debug Session Settings: ☑ Breakpoints ☑ Toolbox ☑ Watchpoints ☑ Memory Display

CPU DLL: S8051.DLL Parameter: | Driver DLL: S8051.DLL Parameter:

Dialog DLL: DP51.DLL Parameter: -pATED2 | Dialog DLL: TP51.DLL Parameter: -pATED2

Page 10 Sec 1 10/38 At 7.2" Ln 4 Col 1 REC TRK EXT OVR English (U.S



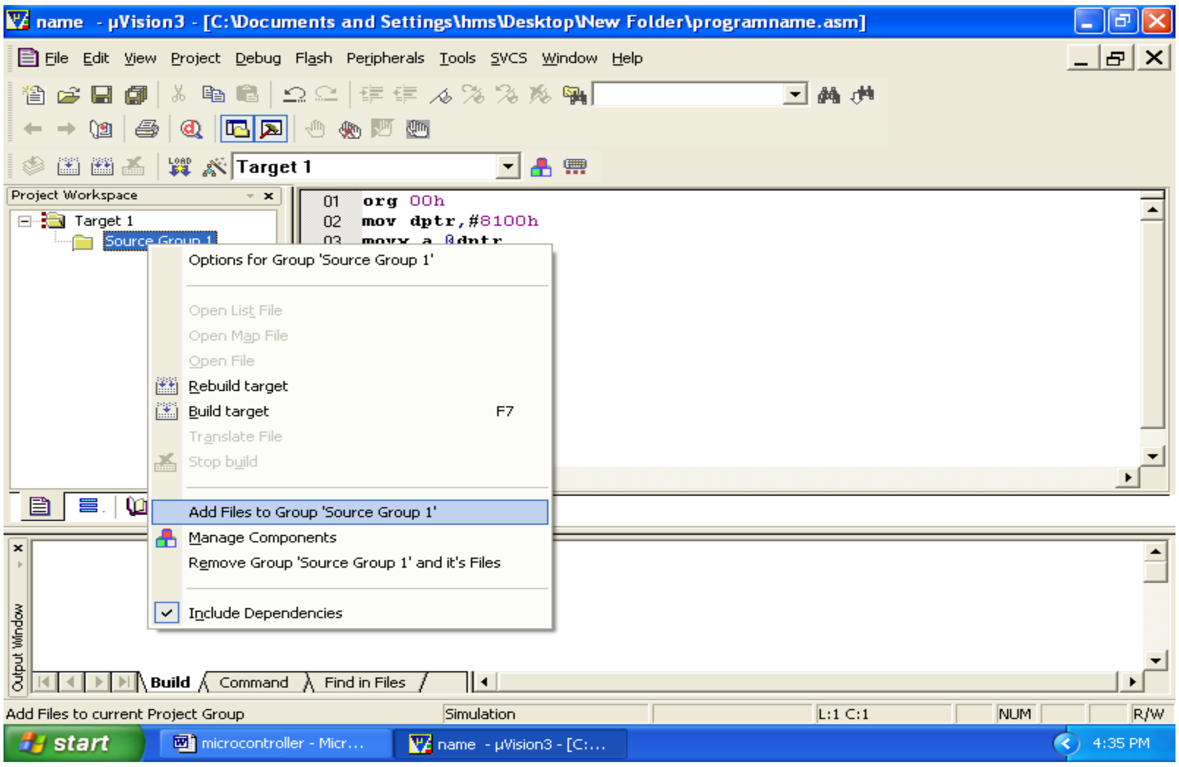


```
org 00h
mov dptr,#8100h
movx a,@dptr
mov r1,a
anl a,#0fh
orl a,#30h
mov r2,a
mov  a,r1
anl a,#0f0h
swap a
orl a,#30h
mov 61h,a
mov 60h,r2
end
```

Save in: New Folder

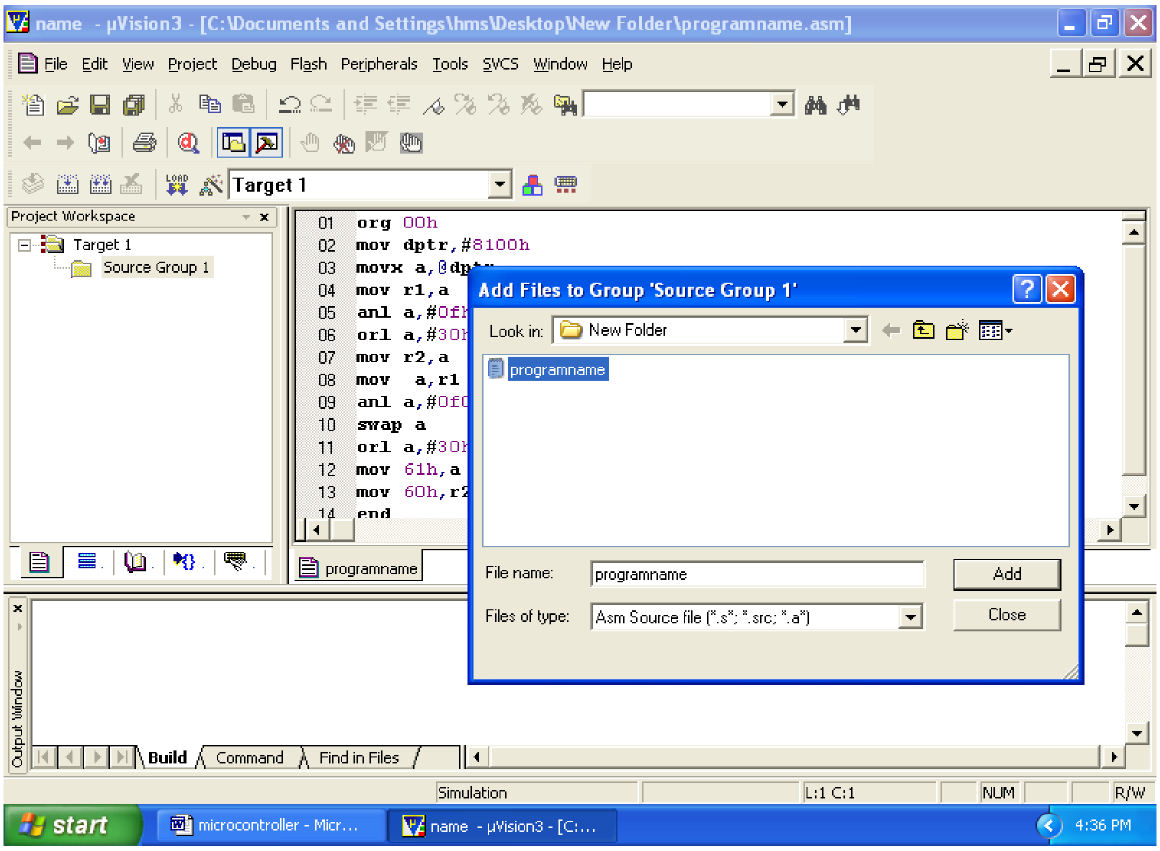
File name: programname.asm

Save as type: All Files (*.*)


name - µVision3 - [C:\Documents and Settings\hms\Desktop\New Folder\programname.asm]

```
org 00h
mov dptr,#8100h
movx a,@dptr
```

Add Files to current Project Group


name - µVision3 - [C:\Documents and Settings\hms\Desktop\New Folder\programname.asm]

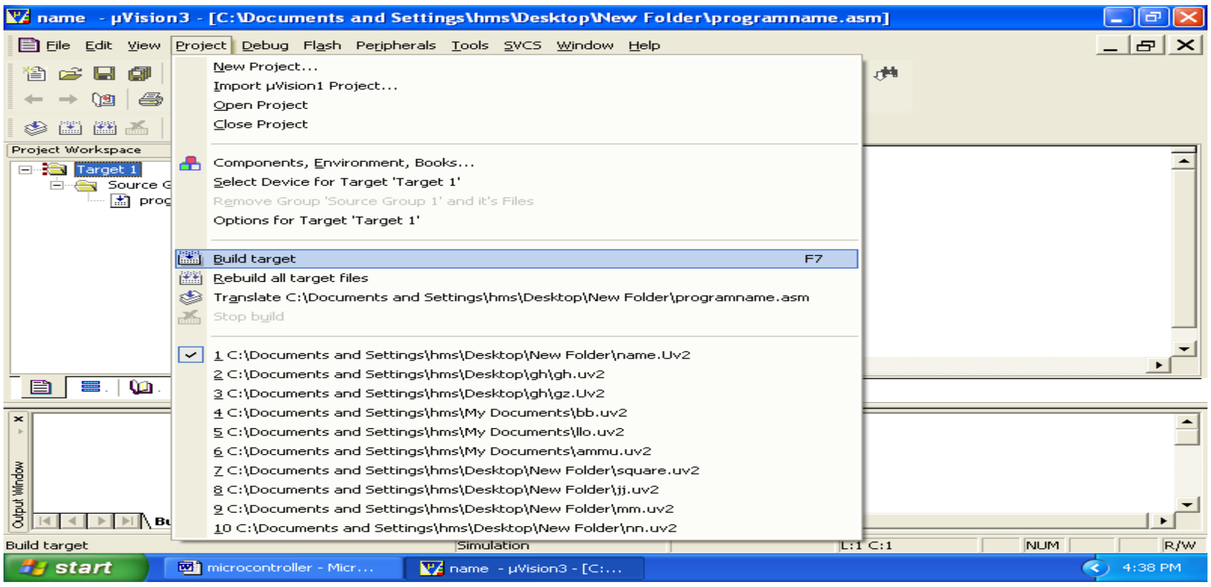
```
org 00h
mov dptr,#8100h
movx a,@dp
mov r1,a
anl a,#0f
orl a,#30
mov r2,a
mov  a,r1
anl a,#0f
swap a
orl a,#30
mov 61h,a
mov 60h,r2
end
```

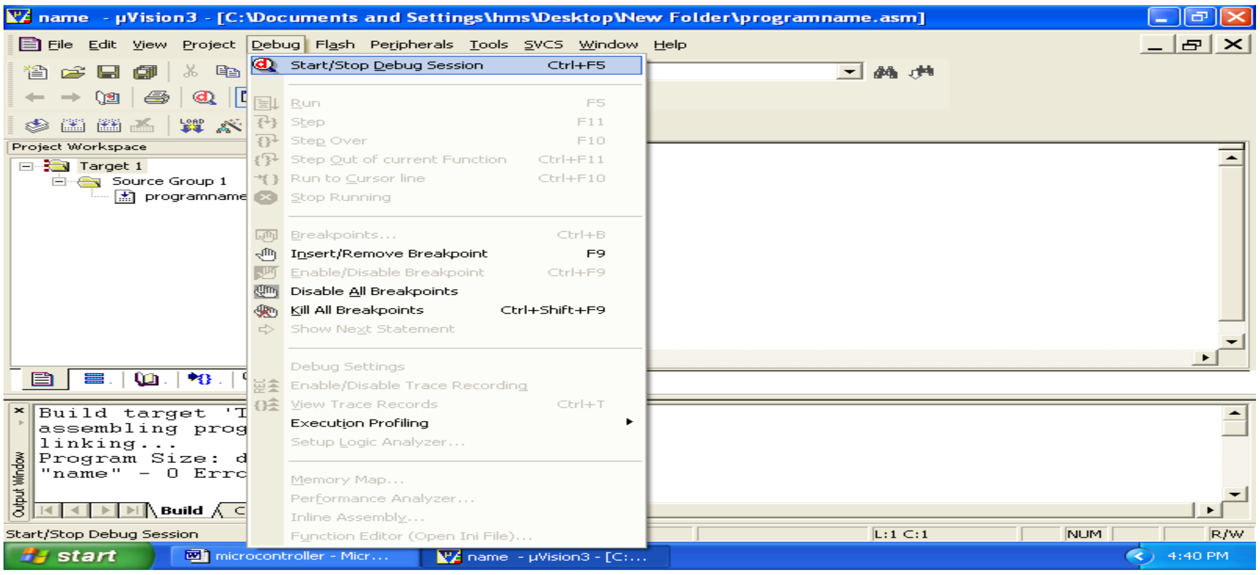
Add Files to Group 'Source Group 1'

Look in: New Folder

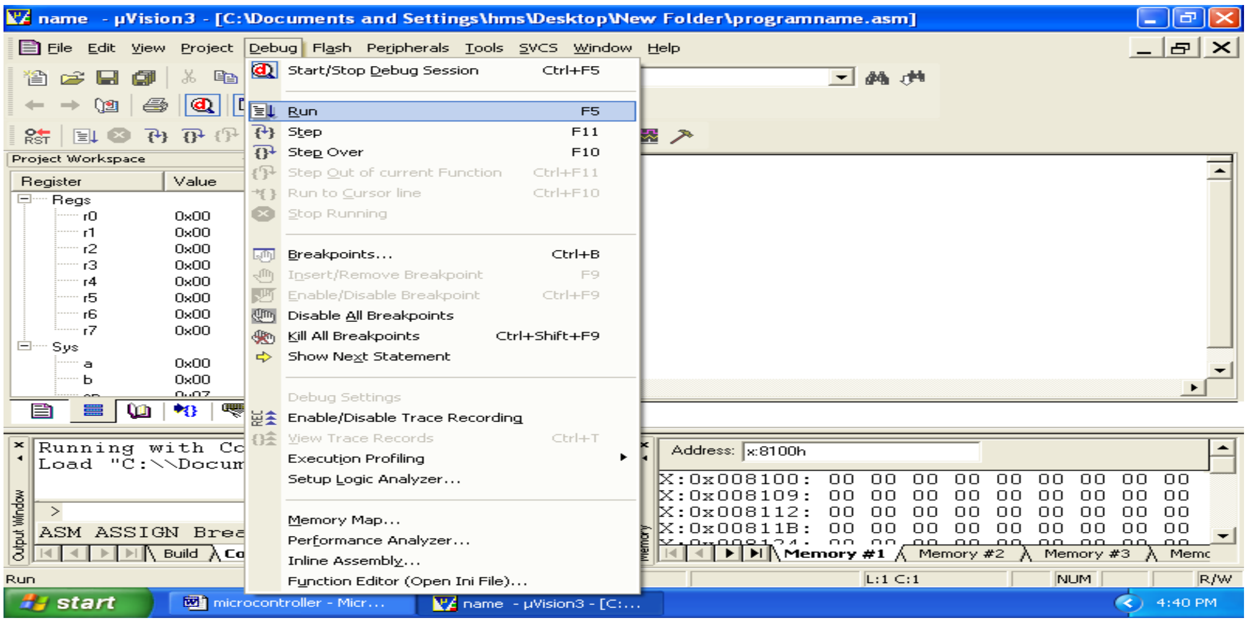
File name: programname

Files of type: Asm Source file (*.s*; *.src; *.a*)


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
microcontroller - Micr...
name - µVision3 - [C:...
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
Start/Stop Debug Session
L:1 C:1
NUM
R/W
start
microcontroller - Micr...
name - µVision3 - [C:...
4:40 PM

```
Build target 'T
assembling prog
linking...
Program Size: d
"name" - 0 Erro
```


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
Run
L:1 C:1
NUM
R/W
start
microcontroller - Micr...
name - µVision3 - [C:...
4:40 PM

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
```

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 to accumulator
mov r1,#02h        ; movesthe immediate data02h to r1register
mov b,r1           ; moves the content or data of r1 register to b register
add a ,b           ; adds accumulator data with b register data and stores
Output in accumulator
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          ; 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
```

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

Output: Initially a =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
Mov61h, b               ; reminder value to 61h data location
```

Output: D: 60h=
D: 61h=

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

```
Movr0, #12h             ; get first no inr0
Mova, r0                ; copy r0 value to accumulator
Movr1, #05h             ; get second no in r1
Movb, r1                ; copy r0 value to register b
Mul ab                  ; multiply A byB
Mov60h,a                ; Output stored in 60h data location
```

Output: D: 60h=5A

Example 6: Program AND, SWAP, OR operations

```
 Mov r0 ,#12h               ; get first no inr0
Mova,r0                     ; copy r0 value to accumulator
Anla,#0F0h                  ; mask lower bit
Mov60h,a                    ; store Output of AND operation in 60h data location
Mov  a,r0                   ; copy r0 value to accumulator
Swapa                       ; exchange upper and lower nibbles of  acc
Mov61h,a                    ;store Output of AND operation in 61h data location
Mov  a,r0                   ; copy r0 value to accumulator
Orla,0f0h                   ; OR operation
Mov62h,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 µ vision
- Go to project Select ❼ Create New project
- Select Atmel AT89C51ED2 IDE from the Kiel vision
- Select New file, Enter the program and Save as(.asm in Assembly and .c in C)and Click ❼ ok
- Add above file to the project created, build target , debug and run the program
- observe the result , by giving particular input before execution.

1. 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
adda,r2
movx
@dptr,adecdp
l
mov a,r1
addc a,r3
movx
@dptr,amov
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,adecdp
l
mov a,r1
subb a,r3
movx@dptr,a
mov 00h,c
end
```

Output:

```
     r1r0              56  73  h
  -   r3r2   ⇒          fc  22h
                     -------------

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

2(c) Multiplication of two 16 bit numbers:

```
movdptr,#9003h
mov r0,#23h
mov r1,#41h
mov r2,#41h
mov r3,#32h
mov a,r3
mov b,r1
mulab
movx
@dptr,amov
r4,b
mov a,r3
mov b,r0
mulab
add a,r4
mov r5,a
mov r4,b
mov a,r2
mov b,r1
mulab
add a,r5
decdpl
movx@dptr,amo
va,b
addca,r4
mov r4,a
mov a,r2
mov b,r0
mulab
add a,r4
decdpl
movx@dptr,a
decdpl
mova,b
movx @dptr,a
end
```

Output: r0 r1 Xr2r3 $\Longrightarrow$ 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,bmov
            b,#0ah
            mulab
            add a,r1
            movb,43h
            div abmov
            46h,a
here:       sjmphere
           end
```

Output: r1 r0 ÷b

2 (e) Find square of a number:

```
mov dptr,#9000h
movx
a,@dptrmovb,a
mulabm
ovr0,a
movdptr,#900eh
mova,b
movx@dptr,ai
ncdpl
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,@dptrmov r0,a
movb,a
mulabm
ov r1,b
mov b,r0
mulab
mov dptr,#900e h
movx
@dptr,amov r2,b
mov a,r1
movb,r0
mulab
add a,r2
decdpl
movx@dptr,a
decdpl
mova,b
movx @dptr,a
end
```

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

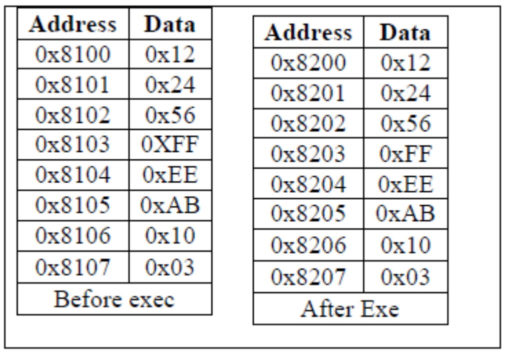
2.Data transfer – Program for block data movement, sorting, exchanging, finding largest element in an array.

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

1(a). Block transfer of data without overlap

```
    MOV R0, #08H              ; initialize the count
    MOV R1, #81H              ; initialize the source memory location higher byte
    MOV R2, #82H               ; initialize the destination memory location higher byte
    MOV R3, #00H               ; initialize the destination & source location lower byte
    BACK: MOV DPH, R1    ; get the source memory location address to DPTR
    MOV DPL, R3
    MOVX A, @DPTR          ; get the data from source memory to Accumulator
    MOV DPH, R2                ; get the destination memory location address to DPTR
    MOVX @DPTR, A          ; copy the accumulator content to destination memory
    INC R3                          ; increment to next source and destination memory
    DJNZ R0, BACK            ; decrement count. If count! =0 go to label "BACK"
    SJMP $
END
```

Outcome:



Address	Data
0x8100	0x12
0x8101	0x24
0x8102	0x56
0x8103	0XFF
0x8104	0xEE
0x8105	0xAB
0x8106	0x10
0x8107	0x03
Before exec	

Address	Data
0x8200	0x12
0x8201	0x24
0x8202	0x56
0x8203	0xFF
0x8204	0xEE
0x8205	0xAB
0x8206	0x10
0x8207	0x03
After Exe	

2(b) Sorting (Ascending and descending order)

```
ORG 0000H
MOV R1, #04H              ; initialize the step count
L1: MOV A, R1             ; move the count to accumulator
MOV R2, A                 ; move accumulator content to R2 (comparison)
MOV DPTR, #5100H          ; Initialize the external memory location
L2: MOVX A,@DPTR          ; get the data from memory to accumulator
MOV B,A                   ; move the accumulator content to B register
INC DPTR                  ; increment the external memory location.
MOVX A, @DPTR             ; get the data from memory to accumulator
CJNE A, B, L3             ; compare accumulator content and B register content, if not equal Jump to label 'L3'
SJMP L5                   ;    short jump to label L5
L3: JC L4                 ; If A& B are not equal, then check CY! =1(A<B) If CY =1(A>B) jump to label 'L4'
SJMP L5                   ; short jump to label L5
L4: XCH A,B               ;Exchange A & B
MOVX @DPTR, A             ; move accumulator content to external memory
INC DPTR                  ; increment the external memory location
L5: DJNZ R2, L2           ; decrement comparison count, if count! =0 then jump to L2
DJNZ R1, L1               ; decrement step count, if count! =0 then jump to label 'L1'
SJMP $
END
```

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.

2(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
        movxa,@dptrxch
        a,r4
        movx@dptr,ai
        ncdptr
        mov 30h,dpl
        mov 31h,dph
        mov dpl,32h
        mov dph,33h
        mov a,r4
        movx@dptr,ai
        ncdptr
        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
Destination Memory location	9100	9101	9102	9103	9104

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
            movxa,@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:88h				

** 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.Counters (UP/DOWN)

3(a) Program for Binary up counter

```
             movdptr,#9000h
             mov     a,#00h
   next:      movx@dptr,a

              acalldelay
              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

```
              movdptr,#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

```
              movdptr,#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,@dptrmov r0,a
        incdptr
        movx
        a,@dptrclr c
        sub  a,r0jz
        equal jnc
        small setb
        7fh
        sjmpend1
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
 sjmphere
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
        anla, 22h      //logical andoperation
        mov 30h, a //and operation result stored in 30h
        mov a, 21h
        orla,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
        rlca
        mov30h,a
        mova,31h
        rrca
        mov 31h,a
        djnz r1,back
        cjnea,00h,npal
        mov a,#0ffh
        sjmp next
npal: mov a,#00h
next: end
```

Result:

Input: **Output:**

2. 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:

5. Code conversion programs

a) **BCD to ASCII**

b) **ASCII to BCD**

c) **ASCII to Decimal**

d) **Decimal to ASCII**

e) **Hexa to decimal**

f) **Decimal to Hexa**

5a) Program to convert a BCD number into ASCII code:

```
          mov dptr,#9000h
          movx
          a,@dptrmov r0,a
          swap a
          mov dptr,#900dh
          acallascii
          mov a,r0
          acallascii
          sjmp $
ascii: anl a,#0fh
          add a,#30h
          movx
          @dptr,aincdpt
          r
          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		

5b) Program to convert a ASCII to BCD

```
mov a,#'4'
anl a,#0fh
swap a
movb,amo
v a,#'7'
anl a,#0fh
orla,b
```

Output: a=

5c) Program to convert a ASCII number into decimal

```
movdptr,#9000h
movx
a,@dptrclrc
subb  a,#30h
movxdptr,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	

5d) 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	

5e) Program to convert Hex number to Decimal:

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

Result: r0=01
r1=06
r2=09

5f) Program to convert decimal number to HEX:

```
mov dptr,#9000h
movx
a,@dptrmov r0,a
anl a,#0f0h
swap a
movb,#0ah
mulabmov
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	

6. Programs to generate delay, Programs using serial port and on- chip timer/counters.

g) Program to configure 8051 microcontroller to transmit characters "ENTER YOUR NAME" to a PC using the serial port and display on the serial window.

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

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.

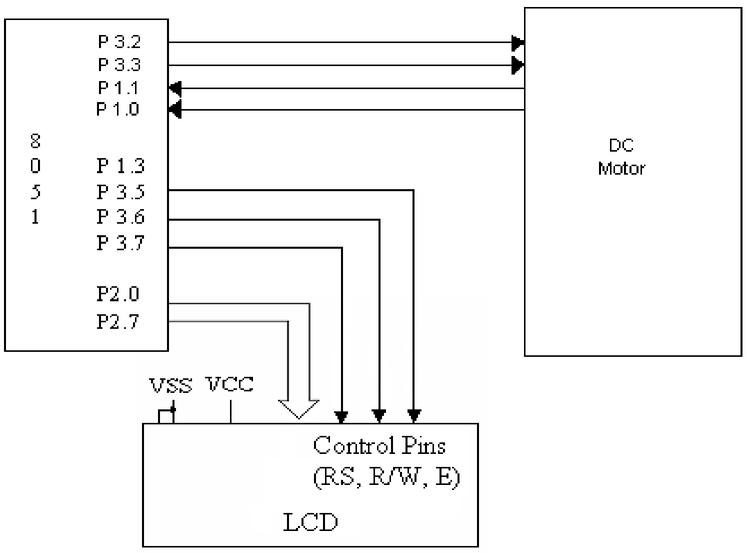
To get 9600, 28800/3 is obtained by loading timer1 with -3 (i.e., FF – 3 = FD) for further clock division. For 2400 baud rate, 28800/12 => -12 = F4 in TH1

Part –B

Interfacing Programs

7. Program for Dc motor interface for direction and speed control using PWM.

Block Diagram:

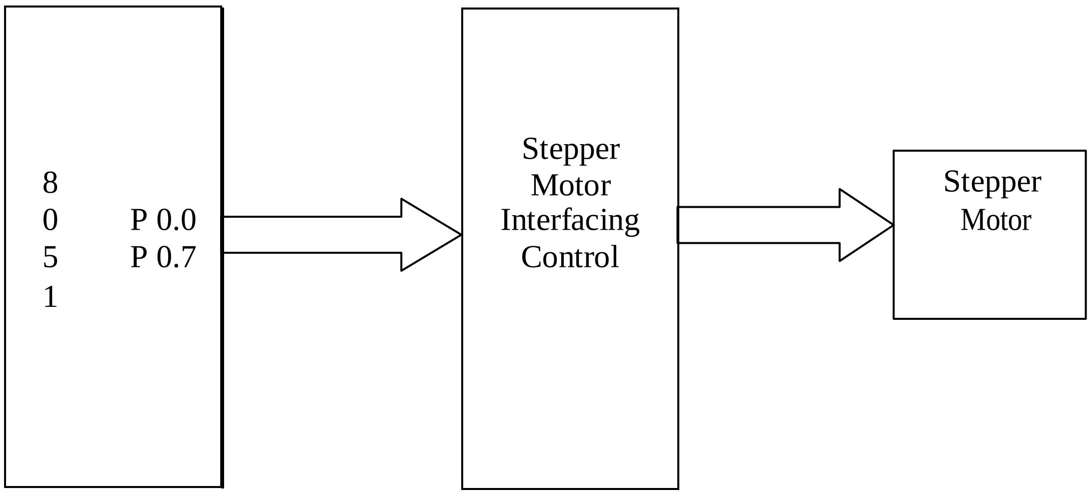


This program measures the motor speed and displays it on LCD
This Program uses Po for DAC data i.e. for speed increment or decrement

```
#include <REG51xD2.H>
sbitinr=  P3^2;  //speed  increment  switch
sbitdcr=  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;
    }
}
```

3. Program for stepper motor interface.

Block Diagram:

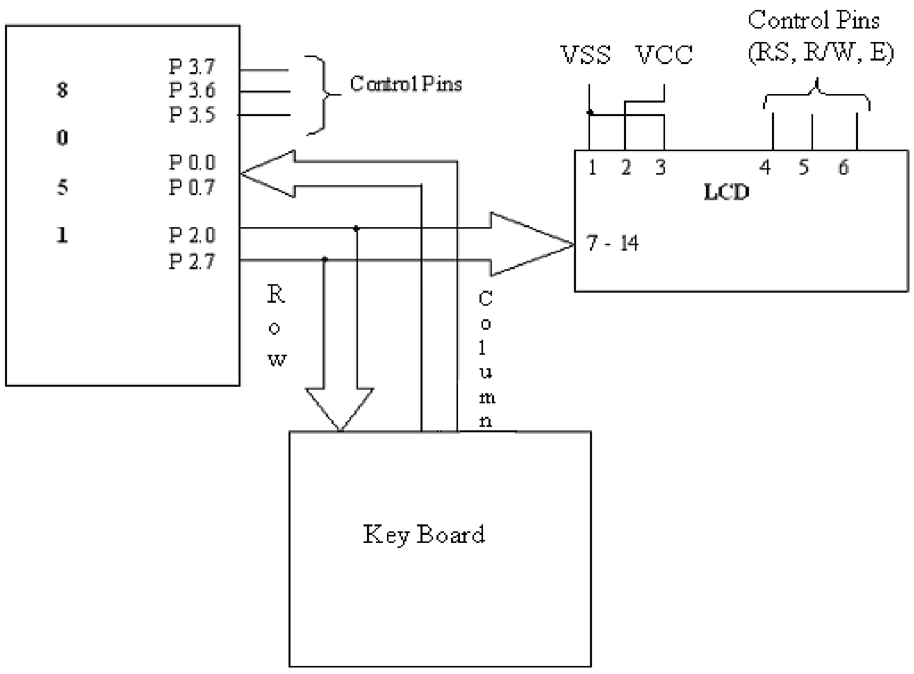


```
#include <REG51xD2.H>
void delay (unsignedint x)                    /* Delay Routine*/
{
 for(;x>0;x--);
 return;
}
main ( )
{
  unsigned char Val, i;
  P0=0x00;
while(1)
{
  Val = 0x11;
  for (i=0;i<4;i++)
             {
              P0 = Val;
              Val=Val<<1;                /* Val= Val>>1;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

Output

9. Program 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

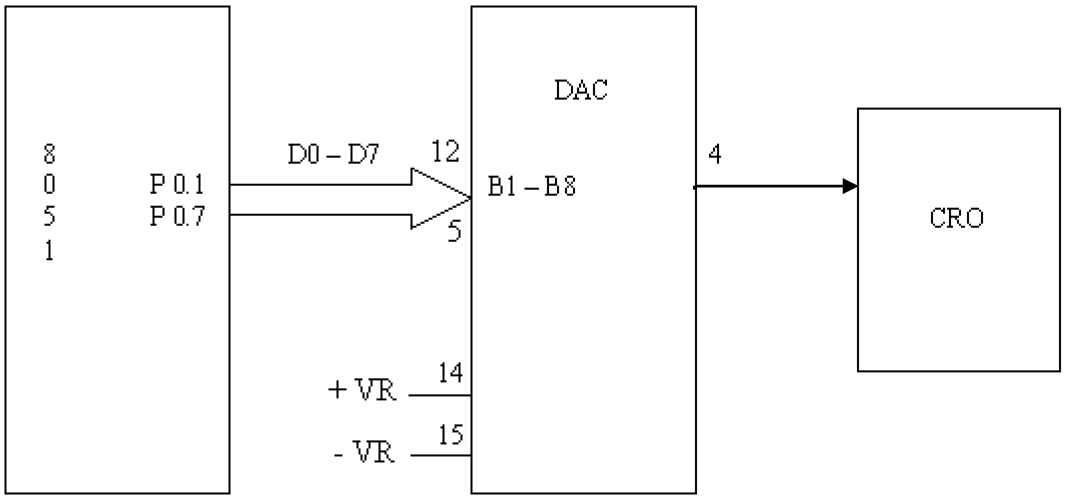
```
#include <REG51xD2.H>
#include "lcd.h"

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(unsignedint x)                /* delay routine*/
{
 for(;x>0;x--);
}
```

Signature o Staff

10(a) Program for dual DAC interfacing to generate square wave of frequency ‘f’.

Block Diagram:

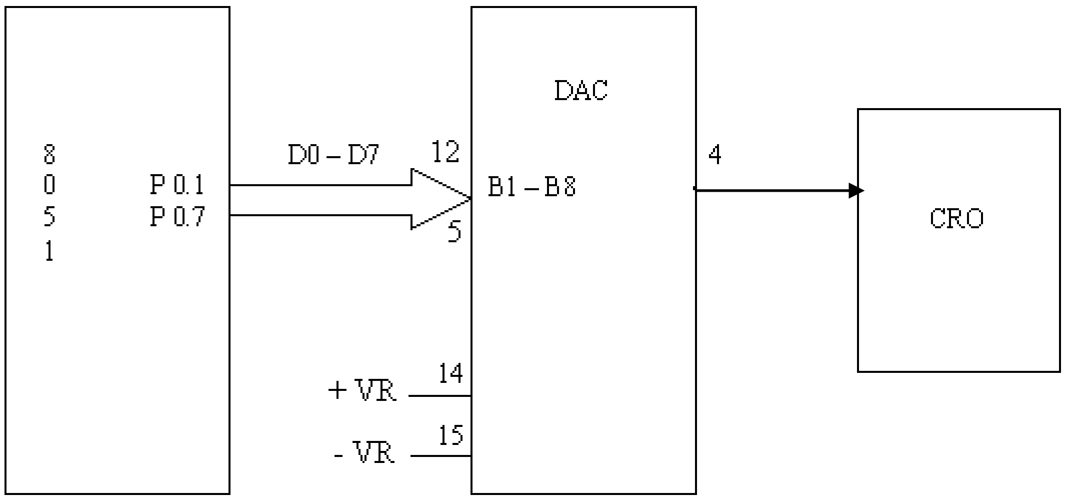


```
#include <REG51xD2.H>

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


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

main()
{
 unsigned char on = 0x7f,off=0x00;
 unsigned intfre = 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:

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

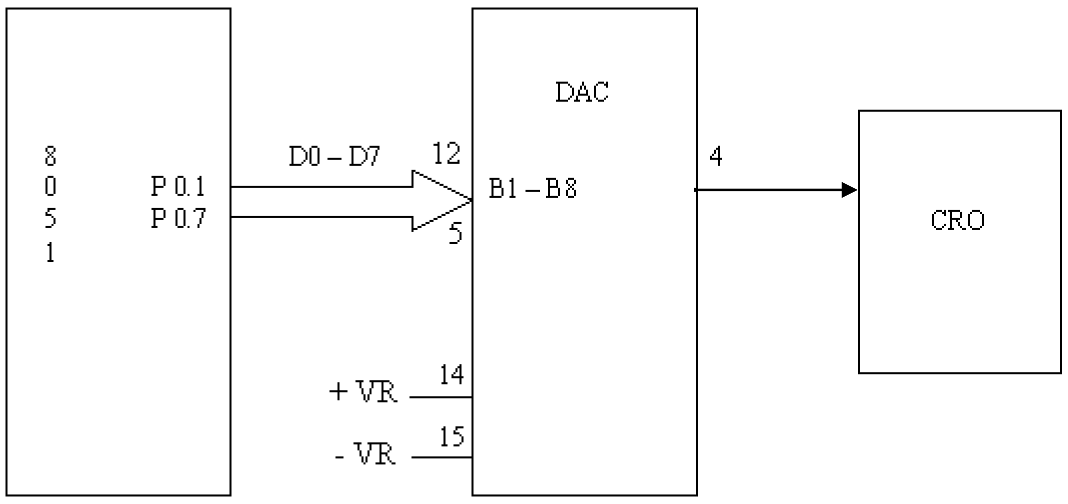
BlockDiagram:



```
#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;
 }

}
```

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

BlockDiagram:

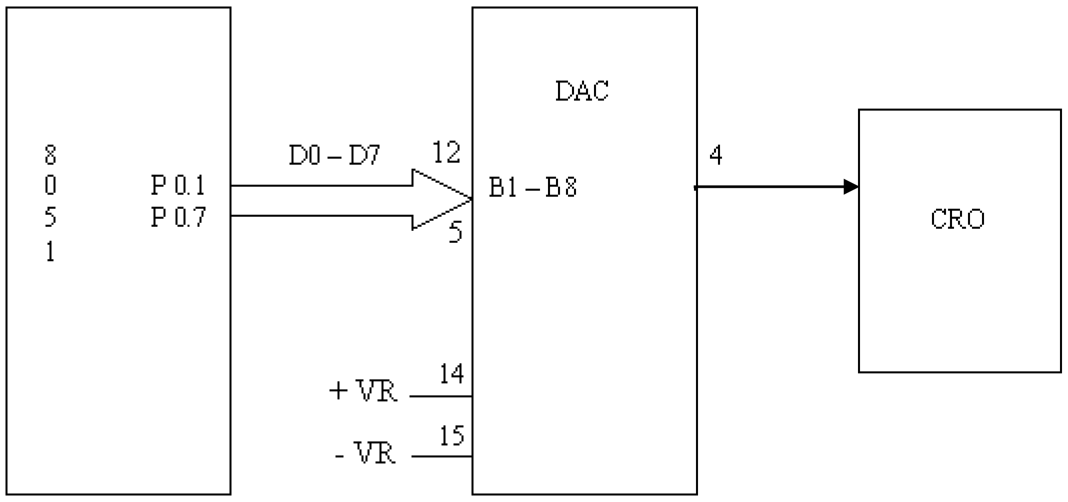


```
#include

<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:

10(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<12; i++)
P0= i;
}
}
```

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?

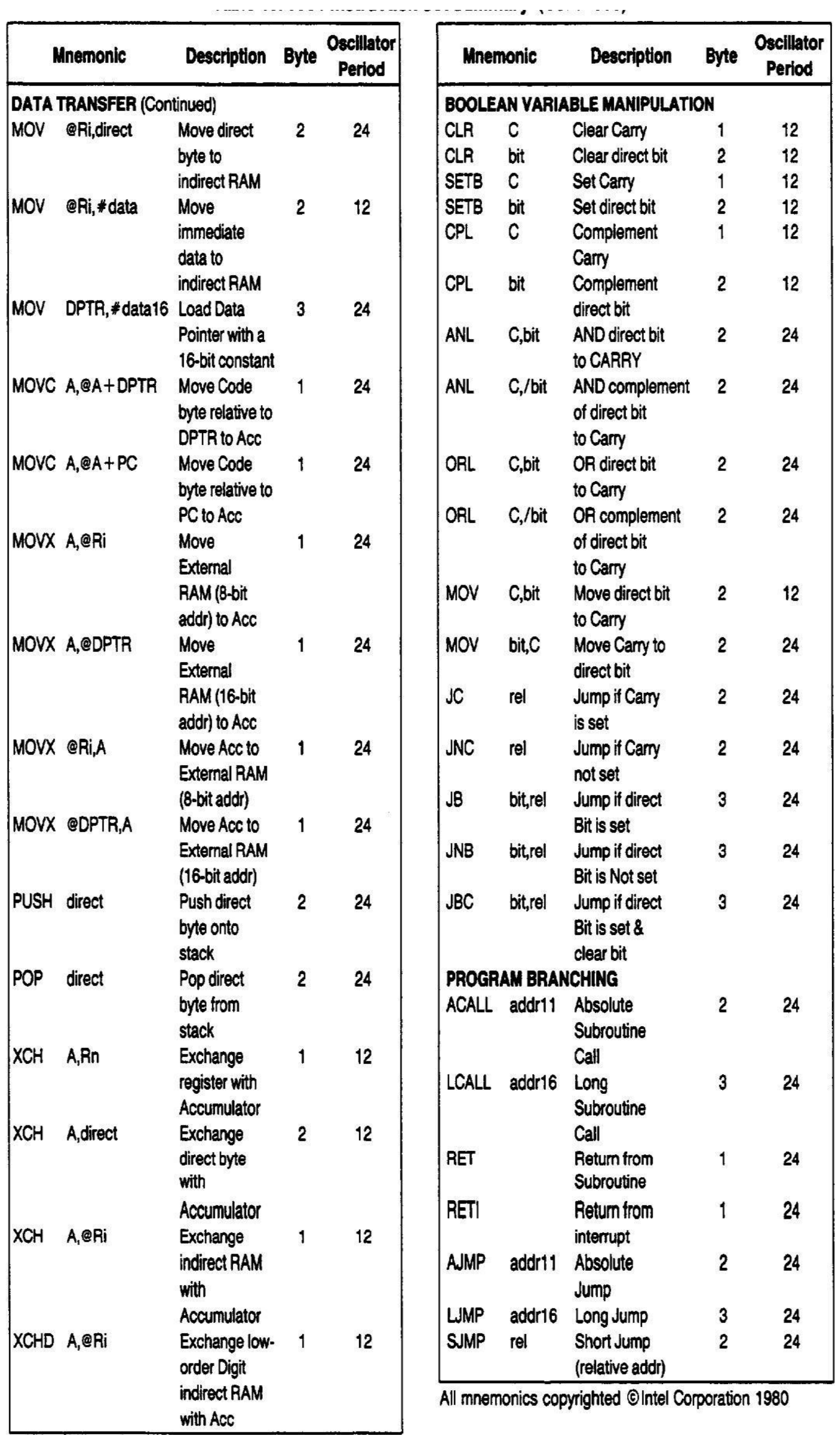
58. What is aState?
59. Explain the need for Hardware Timers and Counters?
60. Give a brief introduction onTimers/Counter.
61. What is the difference between Timer and Counteroperation?
62. How many Timers are there in8051?
63. What are the three functions ofTimers?
64. What are the different modes of 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

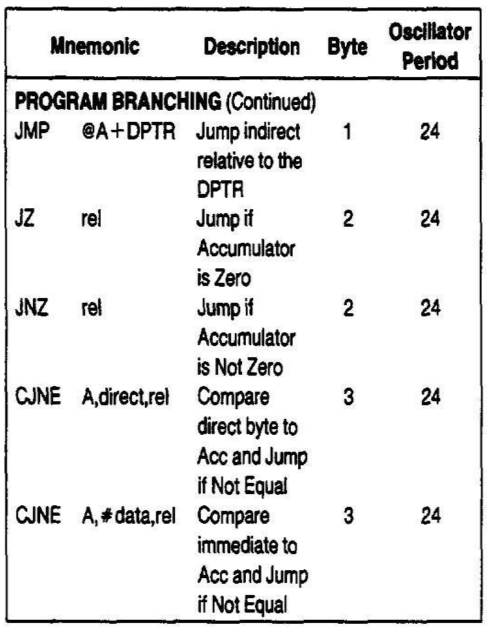
Mnemonic		Description	Byte	Oscillator Period
LOGICAL OPERATIONS (Continued)				
RL	A	Rotate Accumulator Left	1	12
RLC	A	Rotate Accumulator Left through the Carry	1	12
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	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



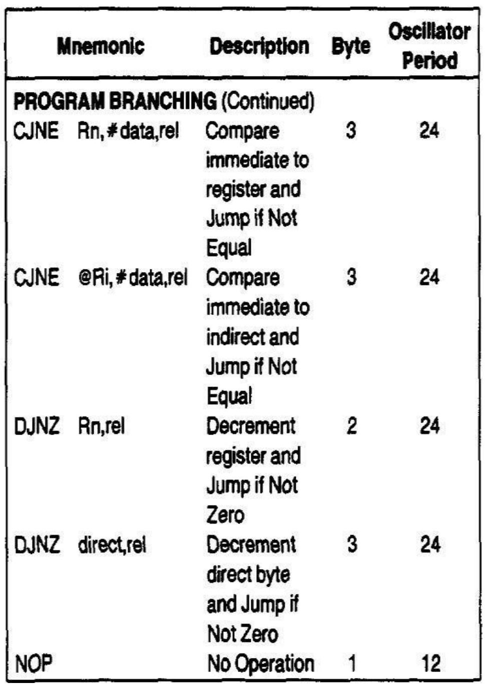
Mnemonic		Description	Byte	Oscillator Period
DATA TRANSFER (Continued)				
MOV	@Ri,direct	Move direct byte to indirect RAM	2	24
MOV	@Ri,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	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

All mnemonics copyrighted ©Intel Corporation 1980



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
```