S J P N Trust's

Hirasugar Institute of Technology, Nidasoshi

Inculcating Values, Promoting Prosperity

Approved by AICTE, Recognized by Govt. of Karnataka and Affiliated to VTU Belagavi.

Accredited at 'A' Grade by NAAC

Programmes Accredited by NBA: CSE, ECE, EEE & ME

Department of Electrical & Electronics Engineering

Microcontroller Laboratory

Manual

18EEL57

<u>Lab Incharge</u>

Prof. M. P. Yanagimath

<u>Lab Instructor</u>

Shri. V. M. Mutalik

Department of Electrical & Electronics Engineering

VISION

To be the centre of excellence in teaching and learning to produce the competent & socially responsible professionals in the domain of Electrical & Electronics Engineering.

MISSION

To educate students with core knowledge of Electrical & Electronics Engineering by developing problem solving skills, professional skills and social awareness to excel in their career.

Microcontroller Laboratory

SEMESTER –V

Course objectives:

1. To explain writing assembly language programs for data transfer, arithmetic, Boolean and Logical instructions.
2. To explain writing assembly language programs for code conversions.
3. To explain writing assembly language programs using subroutines for generation of delays, Counters, configuration of SFRs for serial communication and timers.
4. To perform interfacing of stepper motor and dc motor for controlling the speed.
5. To explain generation of different waveforms using DAC interface.

Experiments

Note: For the experiments 1 to 6, 8051 assembly programming is to be used.

1.	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.

	Note: Single chip solution for interfacing 8051 is to be with C Programs for the following experiments.
8	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.

Course outcomes:

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

1. Write assembly language programs for data transfer, arithmetic, Boolean and logical instructions.
2. Write ALP for code conversions.
3. Write ALP using subroutines for generation of delays, counters, configuration of SFRs for serial communication and timers.
4. Perform interfacing of stepper motor and dc motor for controlling the speed.
5. Generate different waveforms using DAC interface.
6. Work with a small team to carryout experiments using microcontroller concepts and prepare reports that present lab work.

Introduction to Microcontroller 8051

The most universally employed set of microcontrollers come from the 8051 family. 8051 Microcontrollers persist to be an ideal choice for a huge group of hobbyists and experts. The original 8051 microcontroller was initially invented by Intel. The two other members of this 8051 family are-

- 8052-This microcontroller has 3 timers & 256 bytes of RAM. Additionally it has all the features of the traditional 8051 microcontroller. 8051 microcontroller is a subset of 8052 microcontroller.
- 8031 - This microcontroller is ROM less, other than that it has all the features of a traditional 8051 microcontroller. For execution an external ROM of size 64K bytes can be added to its chip.

8051 microcontroller brings into 2 different sorts of memory such as - NV- RAM, UV - EPROM and Flash.

8051 is the basic microcontroller to learn embedded systems projects.

FEATURES OF 8051

8051 microcontroller is an eight bit microcontroller. It is available in 40 pin DIP package. It has 4kb of ROM (on-chip programmable space) and 128 bytes of RAM space which is inbuilt, if desired 64KB of external memory can be interfaced with the microcontroller. There are four parallel 8 bits ports which are easily programmable as well as addressable. An on- chip crystal oscillator is integrated in the microcontroller which has crystal frequency of 12MHz. In the microcontroller there is a serial input/output port which has 2 pins. Two timers of 16 bits are also incorporated in it; these timers can be employed as timer for internal functioning as well as counter for external functioning.

The microcontroller comprise of 5 interrupt sources namely- Serial Port Interrupt, Timer Interrupt 1, External Interrupt 0, Timer Interrupt 0, External Interrupt 1.

The programming mode of this micro-controller includes GPRs (general purpose registers), SFRs (special function registers) and SPRs (special purpose registers).

INTERNAL ARCHITECHURE OF 8051 MICRO-CONTROLLER

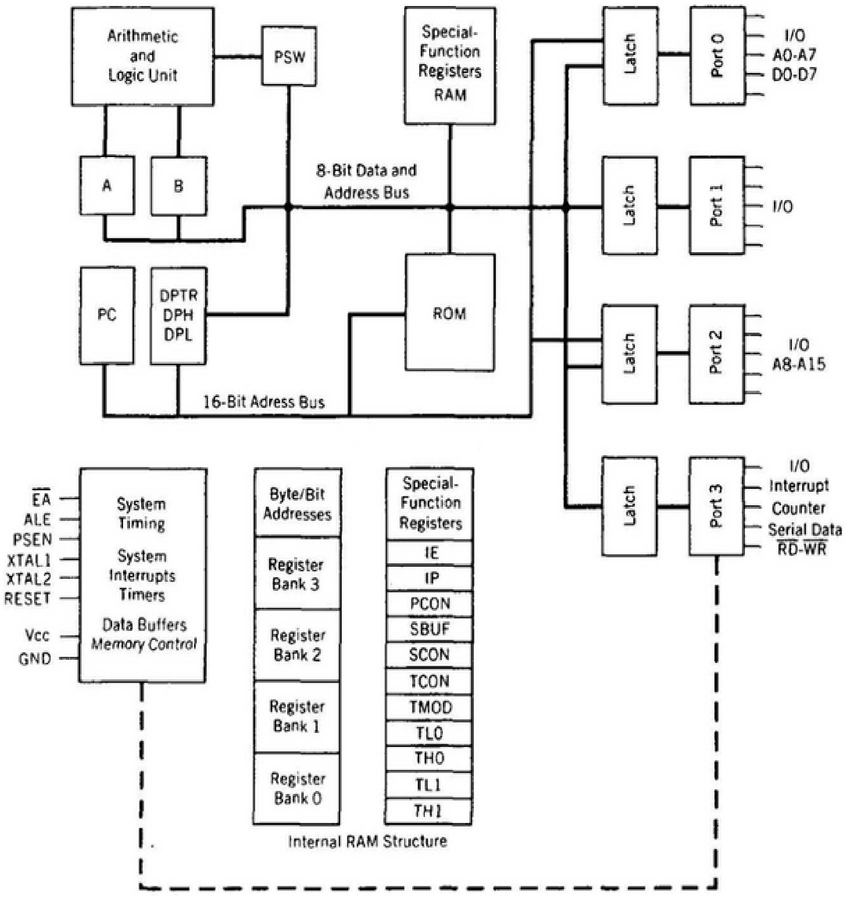


Fig. 1 Internal Architecture of 8051 Micro-Controller

1. ALU

All arithmetic and logical functions are carried out by the ALU. Addition, subtraction with carry, and multiplication come under arithmetic operations. Logical AND, OR and exclusive OR (XOR) come under logical operations.

2. Program Counter (PC)

A program counter is a 16-bit register and it has no internal address. The basic function of program counter is to fetch from memory the address of the next instruction to be executed. The PC holds the address of the next instruction residing in memory and when a command is encountered, it produces that instruction. This way the PC increments automatically, holding the address of the next instruction.

3. Registers

Registers are usually known as data storage devices. 8051 microcontroller has 2 registers, namely Register A and Register B. Register A serves as an accumulator while Register B functions as a general purpose register. These registers are used to store the output of mathematical and logical instructions. The operations of addition, subtraction, multiplication and division are carried out by Register A. Register B is usually unused and comes into picture only when multiplication and division functions are carried out by Register A. Register A also involved in data transfers between the microcontroller and external memory.

PIN DIAGRAM OF 8051 MICRO-CONTROLLER

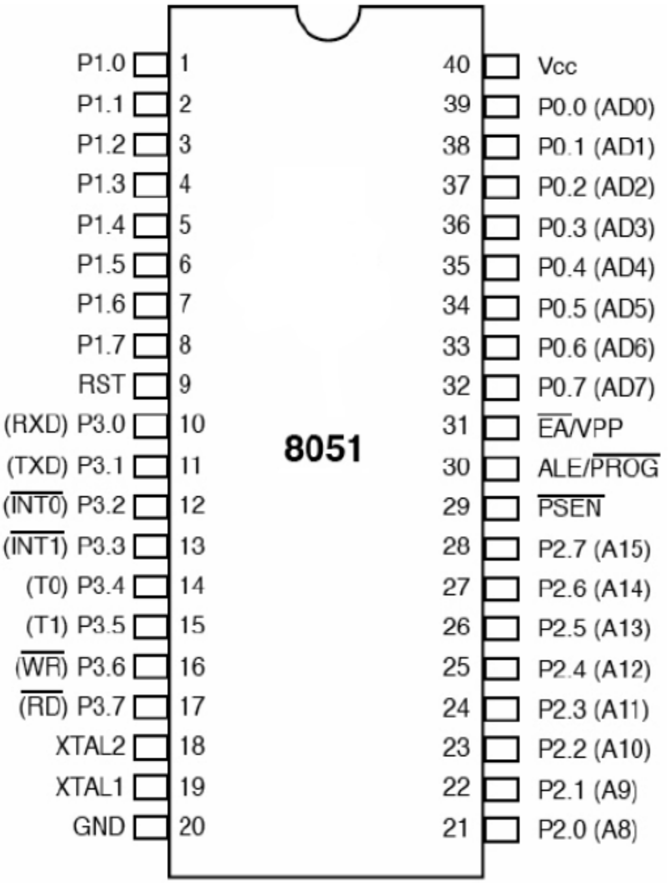


Fig. 2 Pin Diagram of 8051

PINOUT DESCRIPTION

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

Pin 9: RS A logic one on this pin disables the microcontroller and clears the contents of most registers. In other words, the positive voltage on this pin resets the microcontroller. By applying logic zero to this pin, the program starts execution from the beginning.

Pins10-17: Port 3 Similar to port 1, each of these pins can serve as general input or output. Besides, all of them have alternative functions:

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

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

Pin 12: INT0 Interrupt 0 input.

Pin 13: INT1 Interrupt 1 input.

Pin 14: T0 Counter 0 clock input.

Pin 15: T1 Counter 1 clock input.

Pin 16: WR Write to external (additional) RAM.

Pin 17: RD Read from external RAM.

Pin 18, 19: XTAL$_2$, XTAL$_1$ are internal oscillator input and output pins. A quartz crystal which specifies operating frequency is usually connected to these pins. Instead of it, miniature ceramics resonators can also be used for frequency stability. Later versions of microcontrollers operate at a frequency of 0 Hz up to over 50 Hz.

Pin 20: GND Ground.

Pin 21-28: Port 2 If there is no intention to use external memory then these port pins are configured as general inputs/outputs. In case external memory is used, the higher address byte, i.e. addresses A8-A15 will appear on this port. Even though memory with capacity of 64Kb is not used, which means that not all eight port bits are used for its addressing, the rest of them are not available as inputs/outputs.

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

Pin 30: ALE Prior to reading from external memory, the microcontroller puts the lower address byte (A0-A7) on P0 and activates the ALE output. After receiving signal from the ALE pin, the external register (usually 74HCT373 or 74HCT375 add-on chip) memorizes the state of P0 and uses it as a memory chip address. Immediately after that, the ALU pin is returned its previous logic state and P0 is now used as a Data Bus. As seen, port data multiplexing is performed by means of only one additional (and cheap) integrated circuit. In other words, this port is used for both data and address transmission.

Pin 31: EA By applying logic zero to this pin, P2 and P3 are used for data and address transmission with no regard to whether there is internal memory or not. It means that even there is a program written to the microcontroller, it will not be executed. Instead, the program written to external ROM will be executed. By applying logic one to the EA pin, the microcontroller will use both memories, first internal then external (if exists).

Pin 32-39: Port 0 Similar to P2, if external memory is not used, these pins can be used as general inputs/outputs. Otherwise, P0 is configured as address output (A0-A7) when the ALE pin is driven high (1) or as data output (Data Bus) when the ALE pin is driven low (0).

Pin 40: VCC +5V power supply.

Introduction to Microcontroller Lab

The main objective of introducing Microcontroller Laboratory in the Curriculum is to provide in-depth knowledge of 8051 and Assemble Language Programming. This laboratory will make students to build real time applications by working with assembly and C-language programs.

Software tools used in Microcontroller Laboratory

1. Keil uVision 3
2. Flash Magic

Hardware Kits used in Microcontroller Laboratory

1. Microcontroller- 89C61x2 Flash Kits
2. DC Motor Interface
3. LCD & Keyboard
4. Temperature Measurement Interface
5. Elevator Interface
6. Stepper motor interface
7. Dual DAC interface
8. 8-bit ADC interface

Application of Microcontroller in Day to Day Life

1. Light sensing & controlling devices
2. Temperature sensing and controlling devices
3. Fire detection & safety devices
4. Industrial instrumentation devices
5. Process control devices

Application of Microcontroller in Industrial Control Devices:

1. Industrial instrumentation devices
2. Process control devices

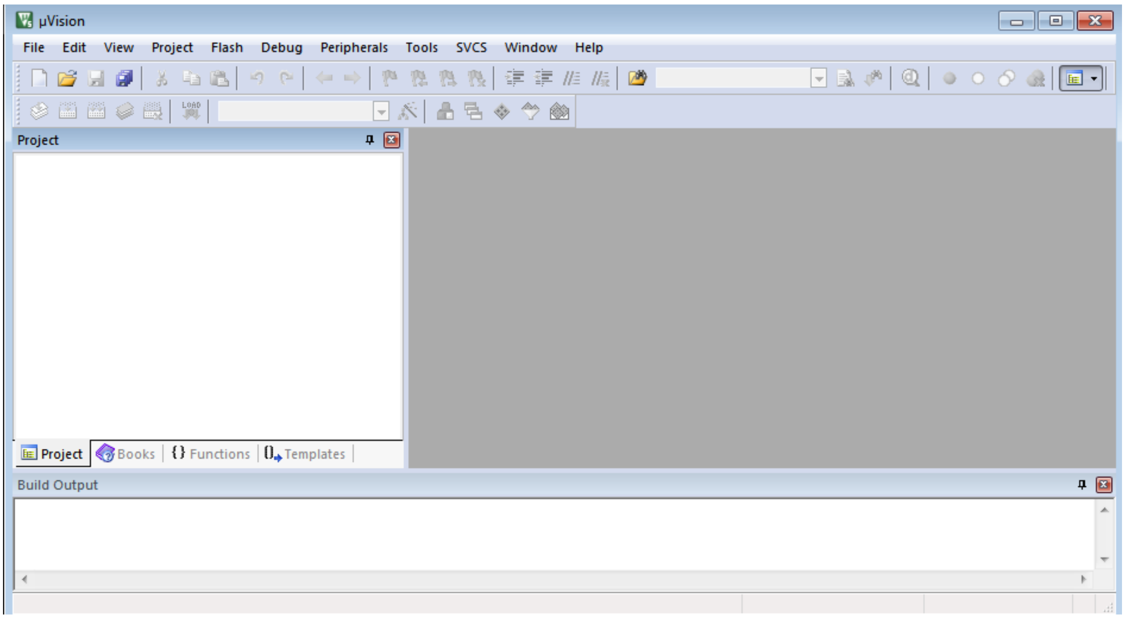
Tools used in Microcontroller Lab

1. Keil Micro vision 3

Procedure to start up with Keil Micro Vision 3

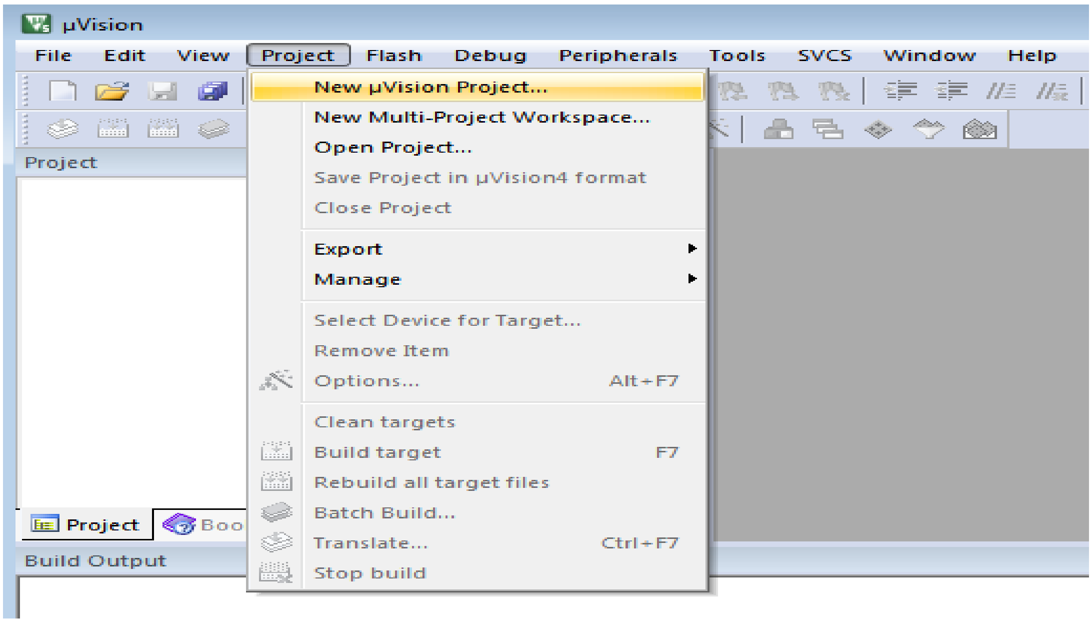
a) Starting Micro vision 3

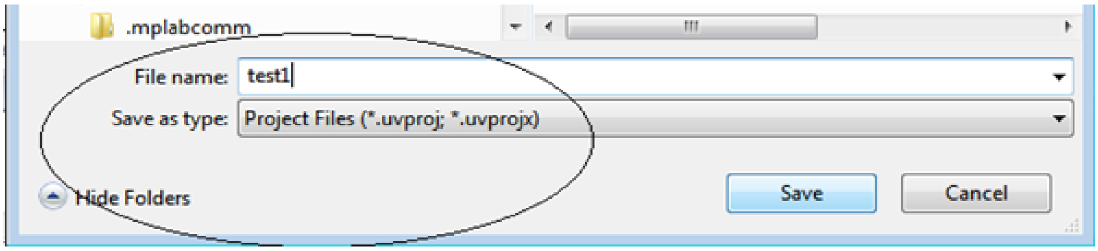
Click on keil Micro Vision icon on the desktop


µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Project
Project Books {} Functions 0→ Templates
Build Output

b) Loading a project into Micro Vision 3

Click on Project menu, Select Close Project if any Projects are Present or Select New Project from the drop – down menu. Enter the filename and Click on Save.

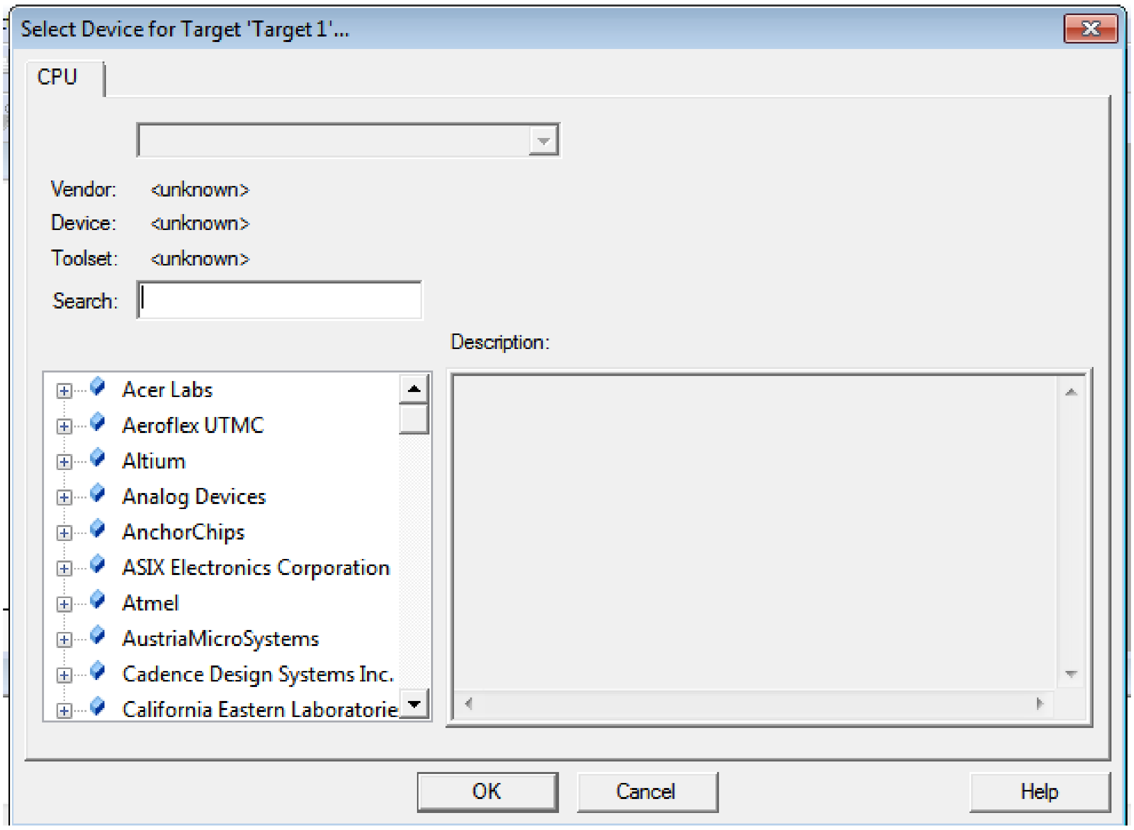




File name: test1

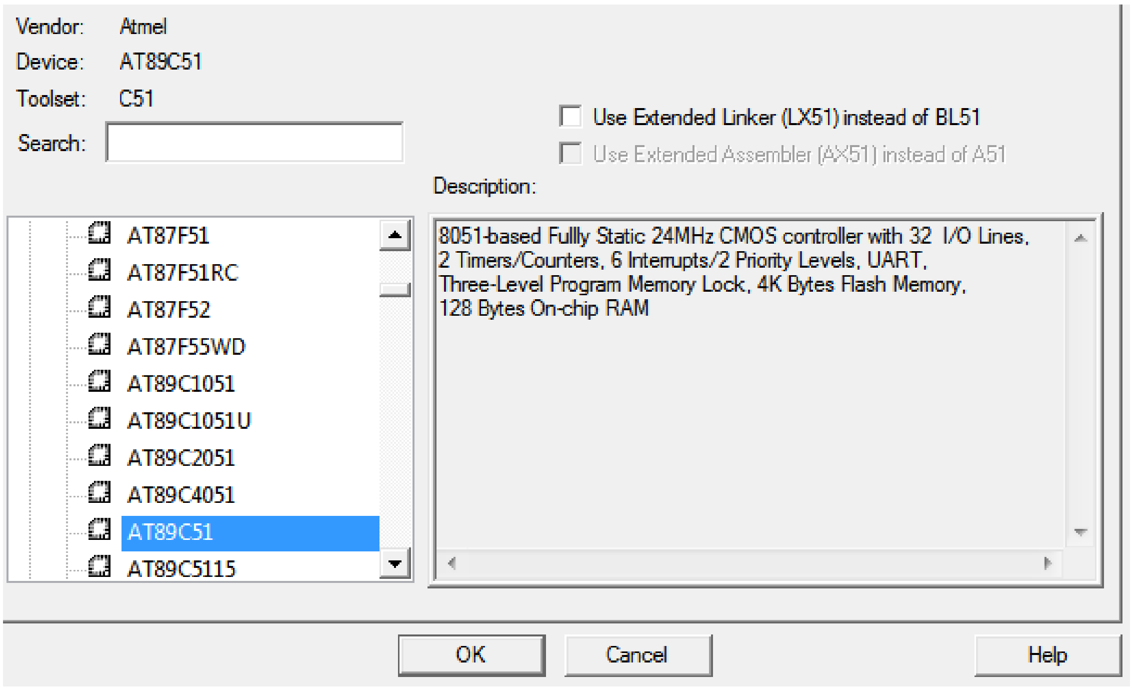
Save as type: Project Files (*.uvproj; *.uvprojx)

Double Click on ATMEL from the wizard then select AT89c51 and Press OK



Select Device for Target 'Target 1'...

Vendor: <unknown>
Device: <unknown>
Toolset: <unknown>
Search:

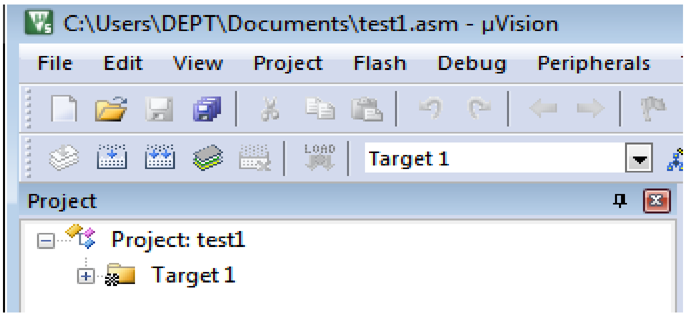


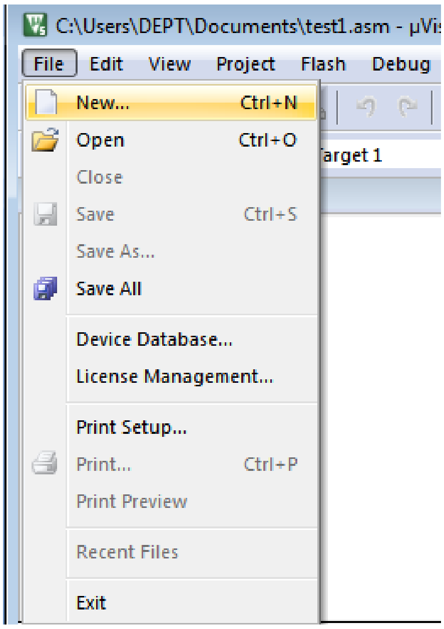
Vendor: Atmel
Device: AT89C51
Toolset: C51
Search:

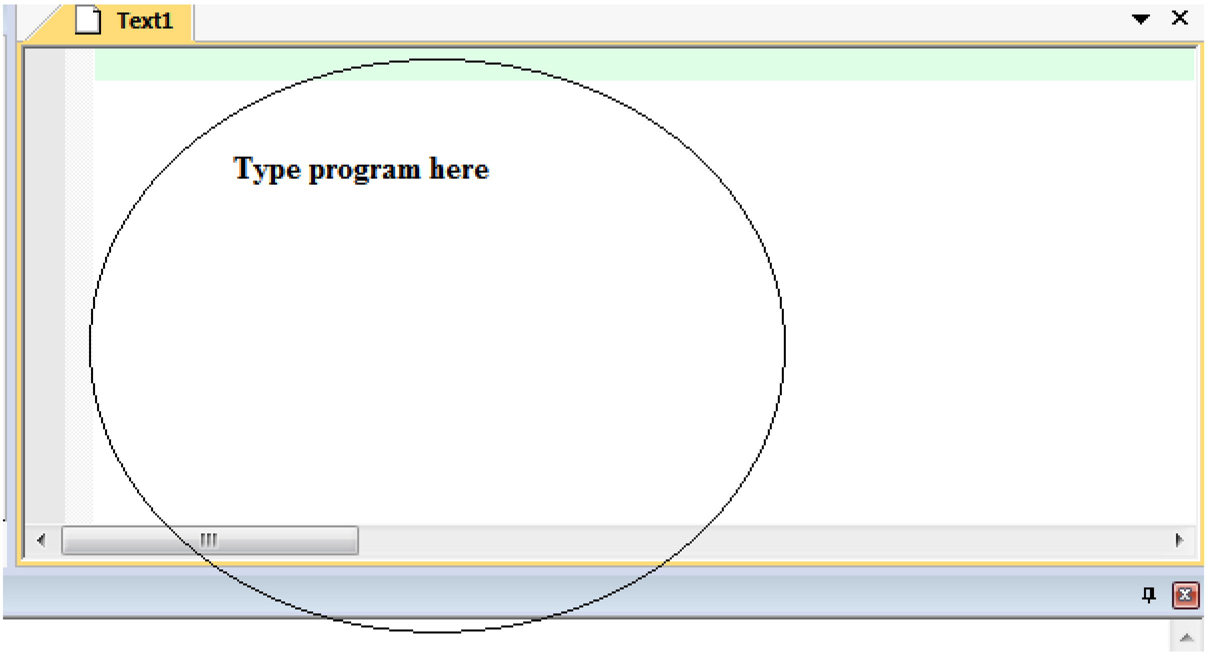
□ Use Extended Linker (LX51) instead of BL51
□ Use Extended Assembler (AX51) instead of A51

Description:
8051-based Fullly Static 24MHz CMOS controller with 32 I/O Lines, 2 Timers/Counters, 6 Interrupts/2 Priority Levels, UART, Three-Level Program Memory Lock, 4K Bytes Flash Memory, 128 Bytes On-chip RAM

Micro vision 3 will load 8051 Microcontroller Projects file and Display as :

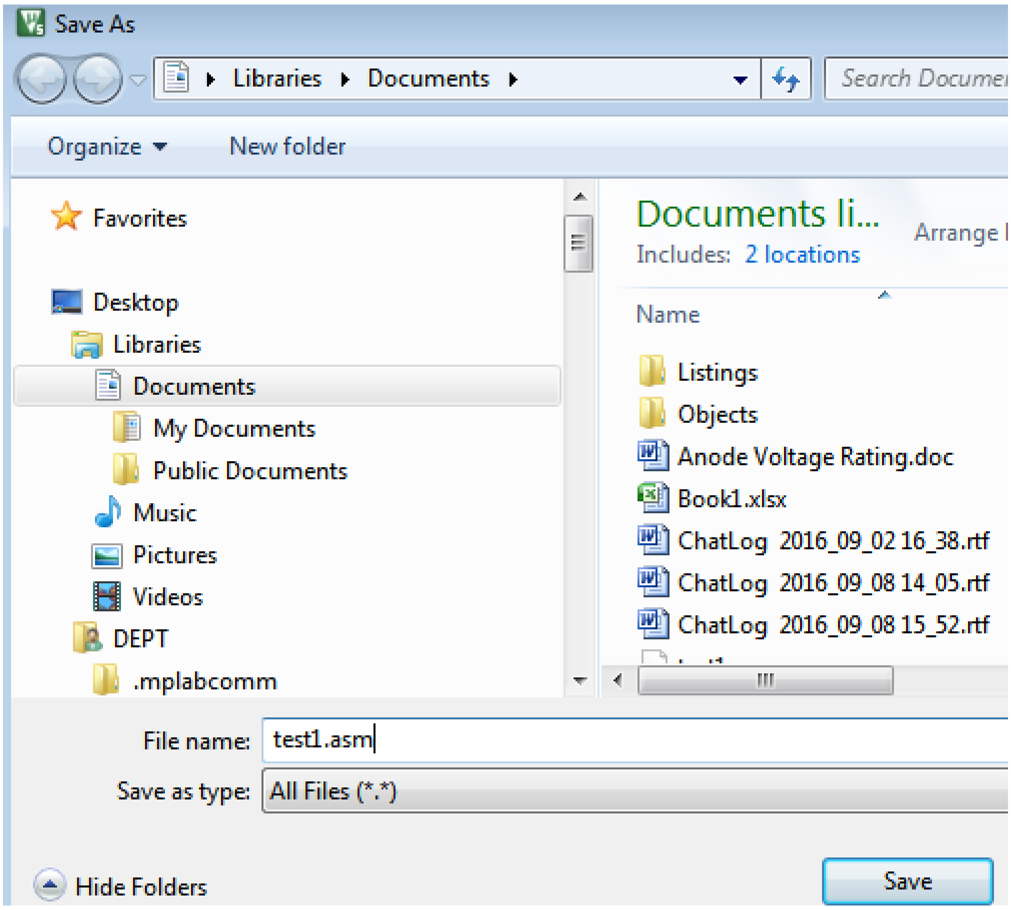

C:\Users\DEPT\Documents\test1.asm - µVision
File Edit View Project Flash Debug Peripherals
Target 1
Project
Project: test1
Target 1





c) Editing and Assembling

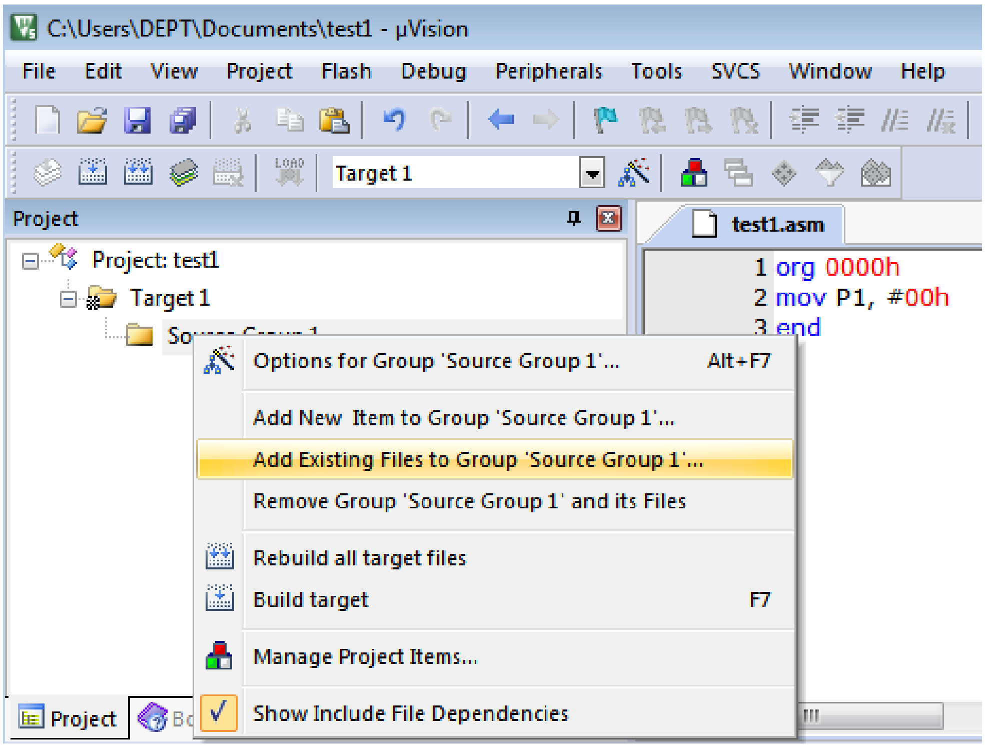
Type the program in the work space window. Now save the file and right click source group 1



File name: test1.asm

Save as type: All Files (*.*)

Select Add files to group source group Let the files be in ASM. Select the corresponding file from the list and click OK.

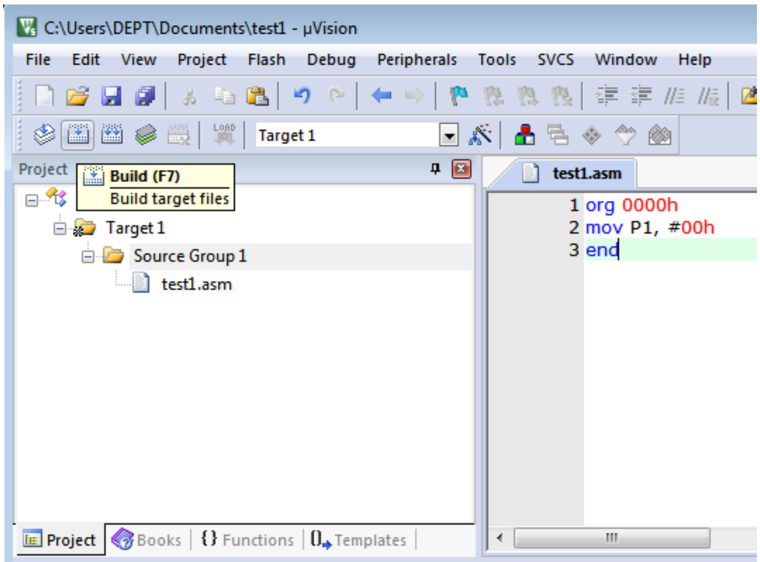


C:\Users\DEPT\Documents\test1 - µVision
File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
Target 1
Project
Project: test1
Target 1
Source Group 1
test1.asm
Options for Group 'Source Group 1'... Alt+F7
Add New Item to Group 'Source Group 1'...
Add Existing Files to Group 'Source Group 1'...
Remove Group 'Source Group 1' and its Files
Rebuild all target files
Build target F7
Manage Project Items...
Show Include File Dependencies
Project

```
org 0000h
mov P1, #00h
end
```

To assemble select build target, if no error(s) are found the output window will display.

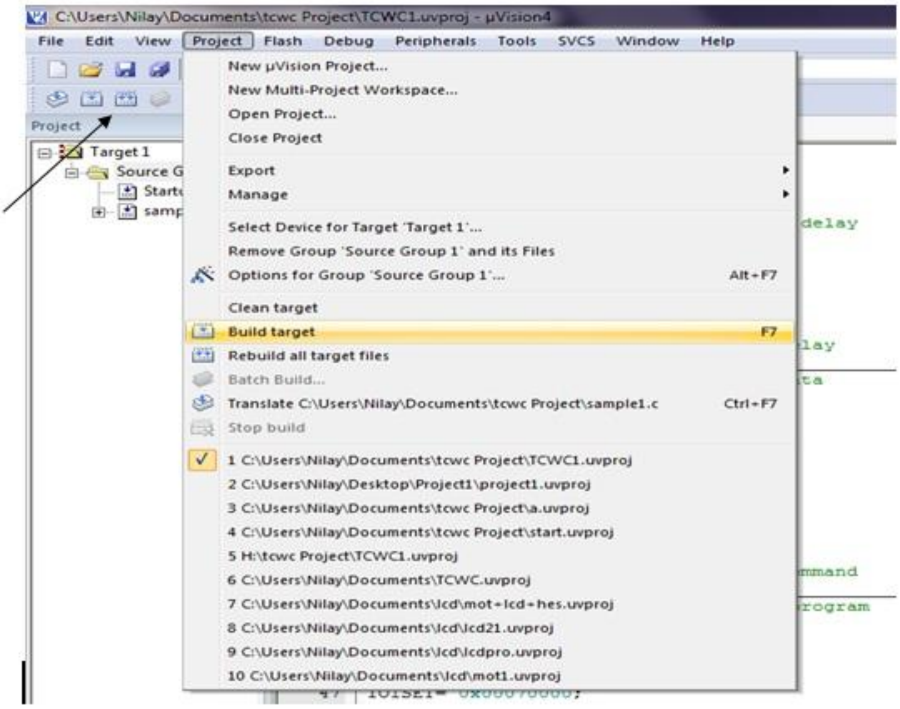
(0) Error(s) , (0)Warning (s).



```
org 0000h
mov P1, #00h
end
```

If error(s) are found then select Rebuild Target and then the Programmer will find it easy to correct the error(s).

d) Debugging
To debug Click on debug button.


C:\Users\Nilay\Documents\tcwc Project\TCWC1.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Project
Target 1
Source G
Start
samp
New µVision Project...
New Multi-Project Workspace...
Open Project...
Close Project
Export
Manage
Select Device for Target 'Target 1'...
Remove Group 'Source Group 1' and its Files
Options for Group 'Source Group 1'... Alt+F7
Clean target
Build target F7
Rebuild all target files
Batch Build...
Translate C:\Users\Nilay\Documents\tcwc Project\sample1.c Ctrl+F7
Stop build
1 C:\Users\Nilay\Documents\tcwc Project\TCWC1.uvproj
2 C:\Users\Nilay\Desktop\Project1\project1.uvproj
3 C:\Users\Nilay\Documents\tcwc Project\a.uvproj
4 C:\Users\Nilay\Documents\tcwc Project\start.uvproj
5 H:\tcwc Project\TCWC1.uvproj
6 C:\Users\Nilay\Documents\TCWC.uvproj
7 C:\Users\Nilay\Documents\lcd\mot+lcd+hes.uvproj
8 C:\Users\Nilay\Documents\lcd\lcd21.uvproj
9 C:\Users\Nilay\Documents\lcd\lcdpro.uvproj
10 C:\Users\Nilay\Documents\lcd\mot1.uvproj
delay
lay
ta
mmand
rogram

For memory display, select Memory window icon under View option. Enter the bytes(s) at memory window (address). Now Click on Run button to run the program continuously. After debugging ends the value will be stored in registers memory and will also be displayed in memory window.

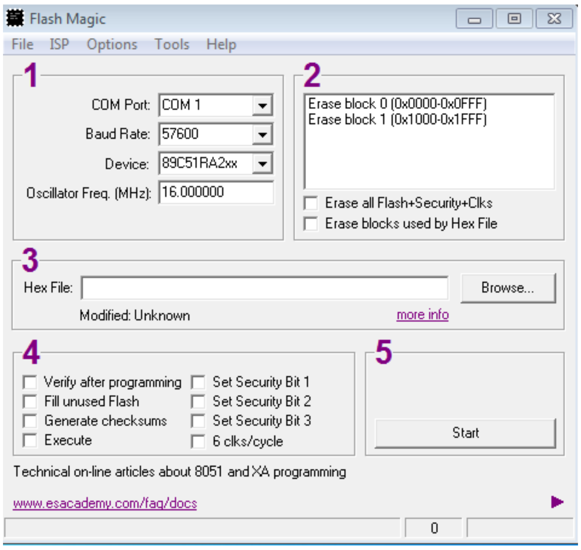
2. Flash Magic

Generally we use ISP (In System Programming) when it comes to micro controllers with flash memory. One of the basic software for such purpose is FLASH MAGIC .

Steps:

1. Click on the icon "flash magic".

2. Following window will appear.



COM Port: COM 1
Baud Rate: 57600
Device: 89C51RA2xx
Oscillator Freq. (MHz): 16.000000

Erase block 0 (0x0000-0x0FFF)
Erase block 1 (0x1000-0x1FFF)

□ Erase all Flash+Security+Clks
□ Erase blocks used by Hex File

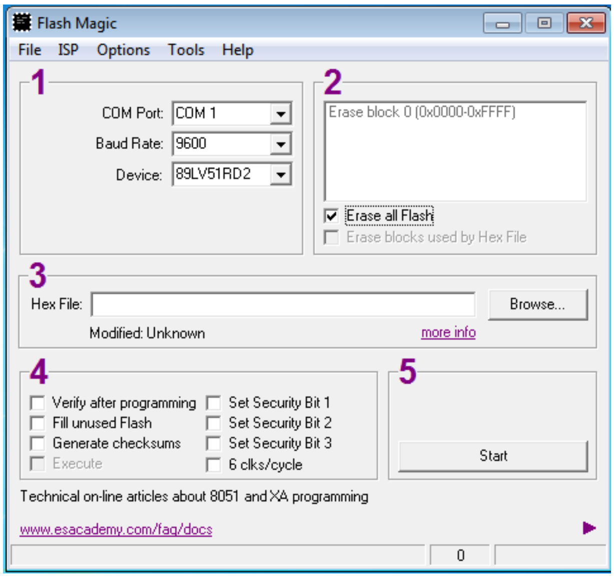
Hex File:
Modified: Unknown more info

□ Verify after programming □ Set Security Bit 1
□ Fill unused Flash □ Set Security Bit 2
□ Generate checksums □ Set Security Bit 3
□ Execute □ 6 clks/cycle

Technical on-line articles about 8051 and XA programming
www.esacademy.com/faq/docs
0

3. Now in this window select the following

COM Port: COM 1
Baud Rate: 9600
Device: 89LV51RD2
Select the option "Erase all Flash".



COM Port: COM 1
Baud Rate: 9600
Device: 89LV51RD2

Erase block 0 (0x0000-0xFFFF)

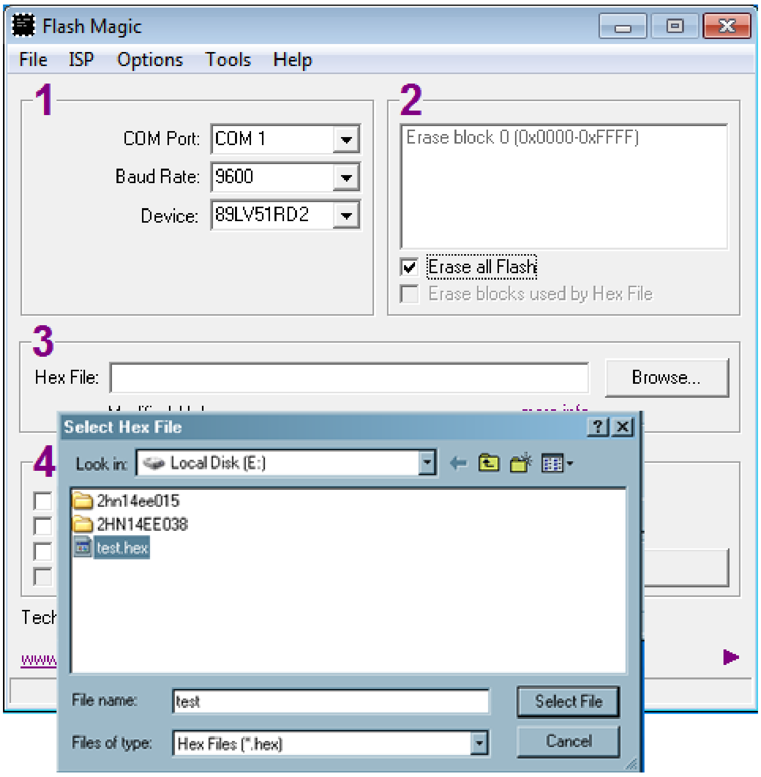
☑ Erase all Flash
□ Erase blocks used by Hex File

Hex File:
Modified: Unknown more info

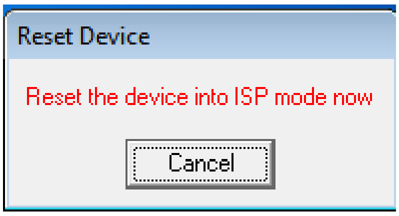
□ Verify after programming □ Set Security Bit 1
□ Fill unused Flash □ Set Security Bit 2
□ Generate checksums □ Set Security Bit 3
□ Execute □ 6 clks/cycle

Technical on-line articles about 8051 and XA programming
www.esacademy.com/faq/docs
0

4. Click on “browse” and select the .hex file to be loaded on the chip.

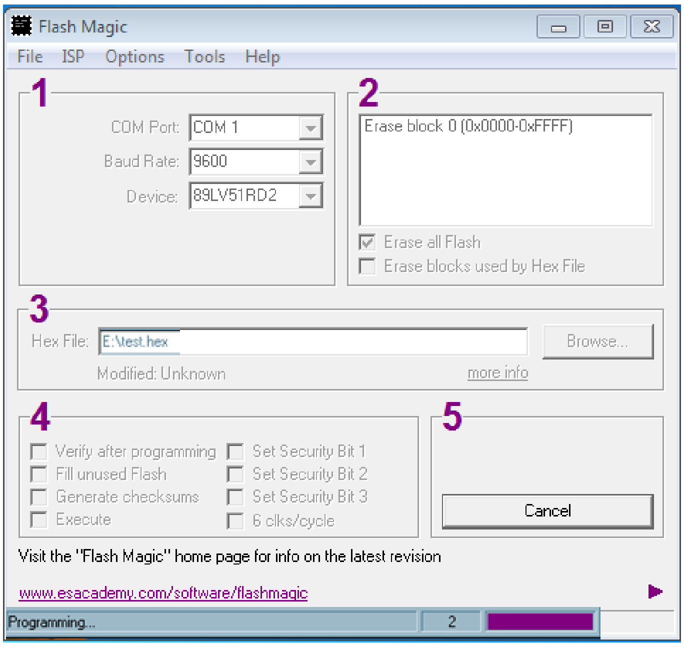


5. Click on “start” and following window will appear and it will ask “reset the device into ISP mode now”.

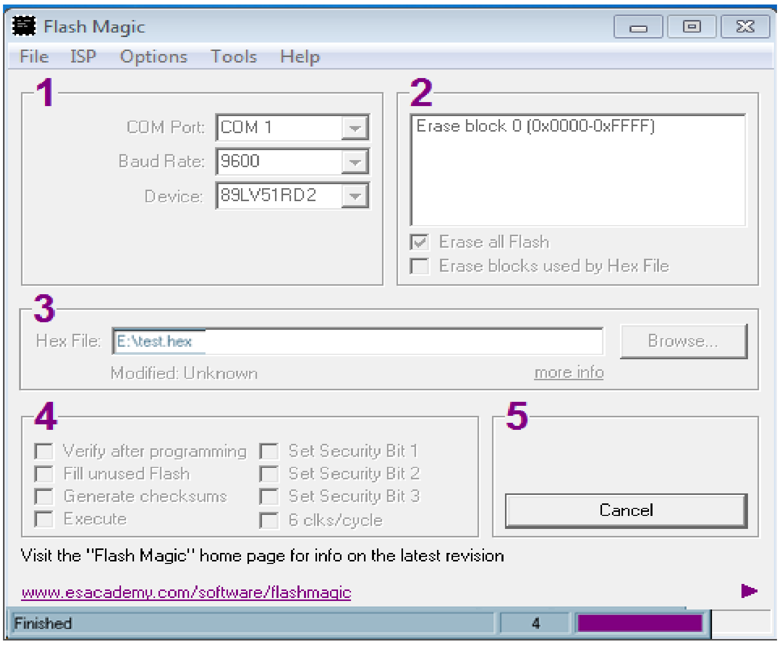


6. Now press the “reset” switch which is on flash board.

7. Now the software starts programming the device.



8. After completion of loading .hex file into chip, it will show as “finished”.



COM Port: COM 1

Baud Rate: 9600

Device: 89LV51RD2

Erase block 0 (0x0000-0xFFFF)

☑ Erase all Flash

□ Erase blocks used by Hex File

Hex File: E:\test.hex

Modified: Unknown

□ Verify after programming

□ Fill unused Flash

□ Generate checksums

□ Execute

□ Set Security Bit 1

□ Set Security Bit 2

□ Set Security Bit 3

□ 6 clks/cycle

Visit the "Flash Magic" home page for info on the latest revision

www.esacademy.com/software/flashmagic

Finished 4

9. Now press “reset” switch from flash board and the device starts executing the program loaded in it.

Sl. No.	List of Experiments
1	**Data transfer – Program for block data movement, sorting, exchanging, finding largest element in an array**
1a)	Write an ALP to move a block of data from one internal memory location to other.
1b)	Write an ALP to move a block of data from one external memory location to other
1c)	Write an ALP to exchange a block of data from one internal memory location to other.
1d)	Write an ALP to exchange a block of data from one external memory location to other
1e)	Write an ALP to find smallest number in the array.
1f)	Write an ALP to find largest number in an array.
1g)	Write an ALP to arrange numbers in ascending order.
1h)	Write an ALP to arrange numbers in descending order.
2	**Arithmetic instructions: Addition, subtraction, multiplication and division. Square and cube operations for 16 bit numbers.**
2a)	Write an ALP to find addition of two 8 bit numbers.
2b)	Write an ALP to find subtraction of two 8 bit numbers.
2c)	Write an ALP to find multiplication of two 8 bit numbers.
2d)	Write an ALP to find division of two 8 bit numbers.
2e)	Write an ALP to square of a 8 bit numbers.
2f)	Write an ALP to cube of a 8 bit numbers.
2g)	Write an ALP to find addition of two 16 bit numbers.
2h)	Write an ALP to find subtraction of two 16 bit numbers.
2i)	Write an ALP to square of a 16 bit numbers.
3	**Counter**
3a)	Write an ALP to generate Hex up counter.
3b)	Write an ALP to generate Hex down counter.
3c)	Write an ALP to generate BCD up counter
3d)	Write an ALP to generate BCD down counter.
4	**Boolean and logical instructions (bit manipulation).**

4a)	Write an ALP to compute the following.
	IF X=0; THEN NUM1 (AND) NUM2, IF X=1; THEN NUM1 (OR) NUM2,
	IF X=2; THEN NUM1 (XOR) NUM2, ELSE RES =00, RES IS 23H LOCATION
	Using logical instructions in byte level.

5	**Conditional call and return instructions.**
5a)	Write a program to toggle all the bits of port 1 continuously by sending the values 55H and AAH using call and return instructions.
5b)	Write an ALP to find factorial of a number using call and return instructions.

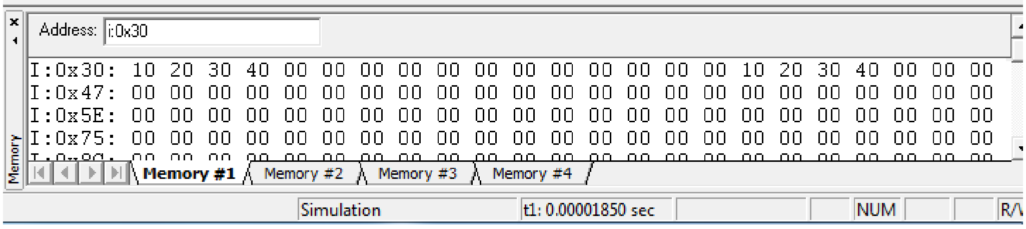
Sl. No.	List of Experiments
6	**Code conversion programs – BCD to ASCII, ASCII to BCD, ASCII to decimal, Decimal to ASCII, Hexa decimal to and Decimal to hexa.**
6a)	Write an ALP to convert hexadecimal number to decimal number.
6b)	Write an ALP to convert decimal number to hexadecimal number.
6c)	Write an ALP to convert packed BCD number to ASCII number.
6d)	Write an ALP to convert ASCII number to BCD number
7	**Programs to generate delay, Programs using serial port and on-chip timer/counters.**
7a)	Write an ALP to toggle the content of port 0 continuously using timer delay in between.
7b)	Write an ALP to transmit characters to a PC HyperTerminal using the serial port and display on the serial window.
Interfacing	
8	**Stepper motor interface.**
8a)	Write a C program to rotate stepper motor in clockwise/anticlockwise direction.
9	**DC motor interface for direction and speed control using PWM.**
9a)	Write a C program to show the on off control of DC motor.
10	**Alphanumerical LCD panel interface.**
10a)	Write a C program to send letters to LCD using delays.
11	**Generate different waveforms: Sine, Square, Triangular, Ramp using DAC interface.**
11a)	Write a C program to generate Square wave using DAC interface to 8051.
11b)	Write a C program to generate Ramp wave using DAC interface to 8051.
11c)	Write a C program to generate triangular wave using DAC interface to 8051.
11d)	Write a C program to generate Sine wave using DAC interface to 8051.
12	**ADC Interface**
12a)	Write a C program to interface ADC to measure temperature.
13	**Elevator interface.**
13a)	Write a C program to show control and operation of elevator using 8051.

Sl No.	Experiment Name
1	**Data transfer – Program for block data movement, sorting, exchanging, finding largest element in an array**
1a)	Write an ALP to move a block of data from one internal memory location to other.
1b)	Write an ALP to move a block of data from one external memory location to other
1c)	Write an ALP to exchange a block of data from one internal memory location to other.
1d)	Write an ALP to exchange a block of data from one external memory location to other
1e)	Write an ALP to find smallest number in the array.
1f)	Write an ALP to find largest number in an array.
1g)	Write an ALP to arrange numbers in ascending order.
1h)	Write an ALP to arrange numbers in descending order.

1a)	Write an ALP to move a block of data from one internal memory location to other.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV R0,#30H	;r0=30h i.e. initial block memory location
	MOV R1,#40H	;r1=40h i.e. block memory location where data has to transfer
	MOV R2,#05H	; load counter as 05h (n) in register r2
NEXT:	MOV A,@R0	;copy the contents of memory location pointed by register r0 into A
	MOV @R1,A	;copy the contents of register A into memory location pointed by reg r1
	INC R0	;increment register r0
	INC R1	;increment register r1
	DJNZ R2, NEXT	;decrement register r2 if not equal to zero jump to next
	LCALL 0003H	;end of asm file

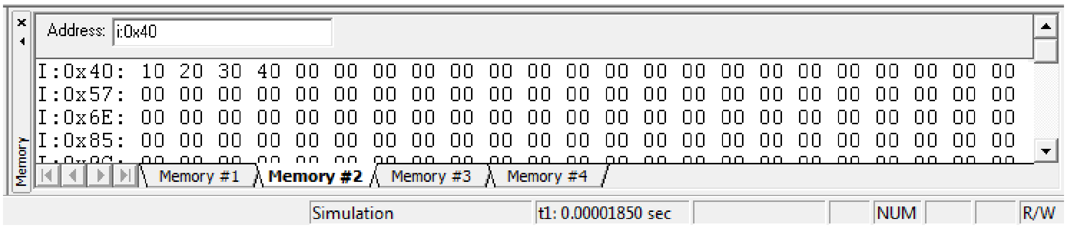
Before Execution



Address: i:0x30

```
I:0x30: 10 20 30 40 00 00 00 00 00 00 00 00 00 00 00 00 10 20 30 40 00 00 00
I:0x47: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x5E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x75: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution



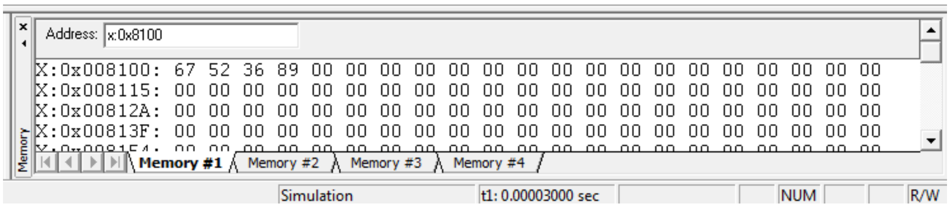
Address: i:0x40

```
I:0x40: 10 20 30 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x57: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x6E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x85: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

1b)	Write an ALP to move a block of data from one external memory location to other

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV R0,#04H	;load the counter as 04h (n) in register r0
	MOV R1,#81H	;higher byte of initial block=81h in register r1
	MOV R2,#85H	;higher byte of memory location where data has to transfer=85h in reg r2
	MOV R3,#00H	;lower byte of both initial and final block=00h in register r3
UP:	MOV DPH,R1	;DPH=contents of register r1
	MOV DPL,R3	;DPL=contents of register r3
	MOVX A,@DPTR	;copy the contents of memory location pointed by DPTR into reg A
	MOV DPH,R2	;dph=contents of register r2
	MOVX @DPTR,A	;copy the contents of reg A into memory location pointed by reg dptr
	INC R3	;increment register r3
	DJNZ R0,UP	;decrement register r0 if not equal to zero jump to up
	LCALL 0003H	;end of asm file

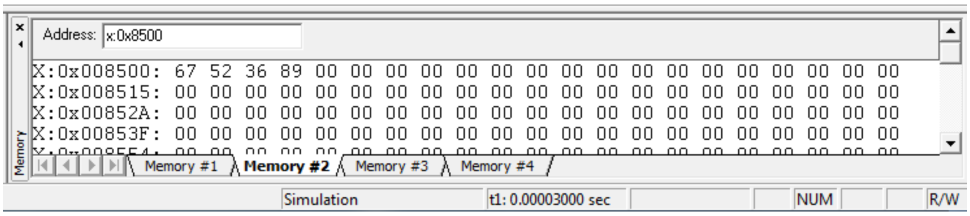
Before Execution



Address: x:0x8100

```
X:0x008100:  67  52  36  89  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
X:0x008115:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
X:0x00812A:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
X:0x00813F:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
```

After Execution



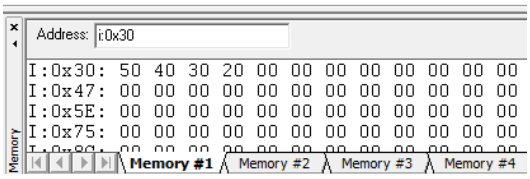
Address: x:0x8500

```
X:0x008500:  67  52  36  89  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
X:0x008515:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
X:0x00852A:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
X:0x00853F:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
```

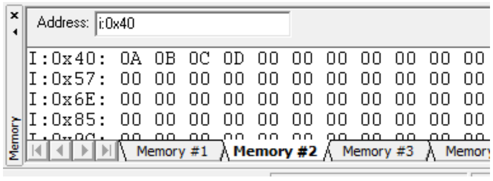
1c)	Write an ALP to exchange a block of data from one internal memory location to other.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV R0,#30H	;r0=30h i.e. initial block memory location
	MOV R1,#40H	;r1=40h i.e. block memory location where data has to transfer
	MOV R4,#04H	; load counter as 04h (n) in register r4
UP:	MOV A,@R0	;copy the contents of memory location pointed by register r0 into A
	MOV R6,A	;copy the contents of register A into memory location pointed by reg r6
	MOV A,@R1	;copy the contents of memory location pointed by register r1 into reg A
	MOV @R0,A	;copy the contents of register A into memory location pointed by reg r0
	MOV A,R6	;copy the contents of r6 in register A
	MOV @R1,A	;copy the contents of register A into memory location pointed by reg r1
	INC R0	;increment register r0
	INC R1	;increment register r1
	DJNZ R4,UP	;decrement register r4 if not equal to zero jump to up
	LCALL 0003H	;end of asm file

Before Execution

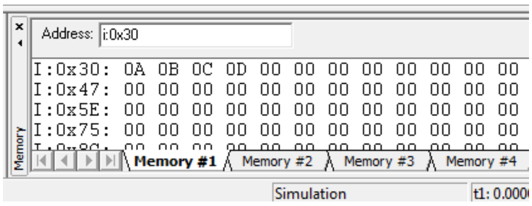

Address: i:0x30

```
I:0x30: 50 40 30 20 00 00 00 00 00 00 00 00
I:0x47: 00 00 00 00 00 00 00 00 00 00 00 00
I:0x5E: 00 00 00 00 00 00 00 00 00 00 00 00
I:0x75: 00 00 00 00 00 00 00 00 00 00 00 00
```

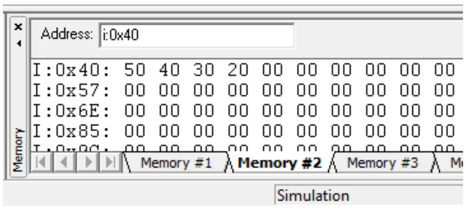

Address: i:0x40

```
I:0x40: 0A 0B 0C 0D 00 00 00 00 00 00 00
I:0x57: 00 00 00 00 00 00 00 00 00 00 00
I:0x6E: 00 00 00 00 00 00 00 00 00 00 00
I:0x85: 00 00 00 00 00 00 00 00 00 00 00
```

After Execution


Address: i:0x30

```
I:0x30: 0A 0B 0C 0D 00 00 00 00 00 00 00 00
I:0x47: 00 00 00 00 00 00 00 00 00 00 00 00
I:0x5E: 00 00 00 00 00 00 00 00 00 00 00 00
I:0x75: 00 00 00 00 00 00 00 00 00 00 00 00
```

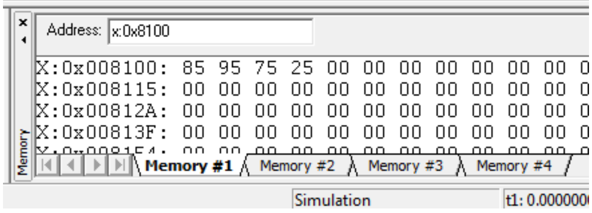

Address: i:0x40

```
I:0x40: 50 40 30 20 00 00 00 00 00 00
I:0x57: 00 00 00 00 00 00 00 00 00 00
I:0x6E: 00 00 00 00 00 00 00 00 00 00
I:0x85: 00 00 00 00 00 00 00 00 00 00
```

1d)	Write an ALP to exchange a block of data from one external memory location to other

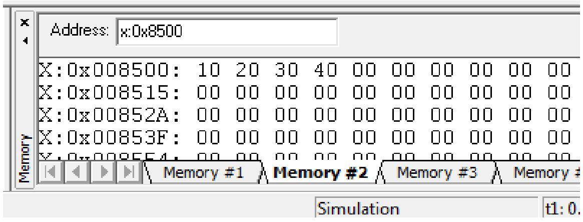
Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV R0,#04H	;load the counter as 04h (n) in register r0
	MOV R1,#81H	;load higher byte of initial block=81h in register r1
	MOV R2,#85H	;load higher byte of memory location where data has to transfer i.e 85h in reg r2
	MOV R3,#00H	;load lower byte of both initial and final block=00h in register r3
UP:	MOV DPH,R1	;DPH=contents of register r1
	MOV DPL,R3	;DPL=contents of register r3
	MOVX A,@DPTR	;copy the contents of memory location pointed by DPTR into reg A
	MOV B,A	; move the content of a into b register
	MOV DPH,R2	;DPH=contents of r2
	MOVX A,@DPTR	;store the contents of register a in memory location pointed by dptr
	MOV DPH,R1	;DPH=contents of r1
	MOVX @DPTR,A	;store the content of a into memory location pointed by r1
	MOV DPH,R2	;DPH=contents of r2
	MOV A,B	;move the content of B into A register
	MOVX @DPTR,A	;copy the contents of register a into memory location pointed by reg dptr
	INC R3	;increment register r3
	DJNZ R0,UP	;decrement register r0 if not equal to zero jump to up
	LCALL 0003H	;end of asm file

Before Execution



Address: x:0x8100

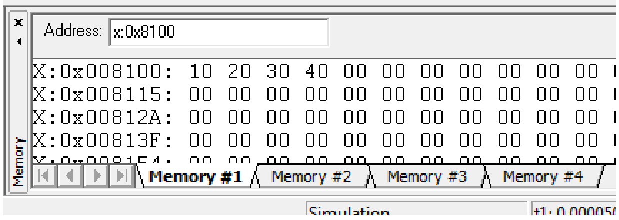
```
X:0x008100: 85 95 75 25 00 00 00 00 00 00 00 0
X:0x008115: 00 00 00 00 00 00 00 00 00 00 00 0
X:0x00812A: 00 00 00 00 00 00 00 00 00 00 00 0
X:0x00813F: 00 00 00 00 00 00 00 00 00 00 00 0
```



Address: x:0x8500

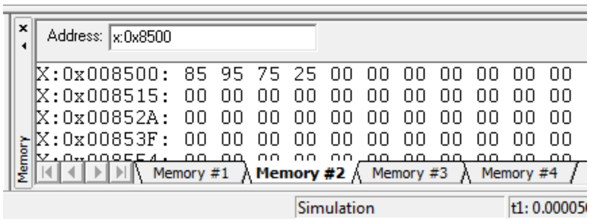
```
X:0x008500: 10 20 30 40 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00
```

After Execution



Address: x:0x8100

```
X:0x008100: 10 20 30 40 00 00 00 00 00 00 00
X:0x008115: 00 00 00 00 00 00 00 00 00 00 00
X:0x00812A: 00 00 00 00 00 00 00 00 00 00 00
X:0x00813F: 00 00 00 00 00 00 00 00 00 00 00
```



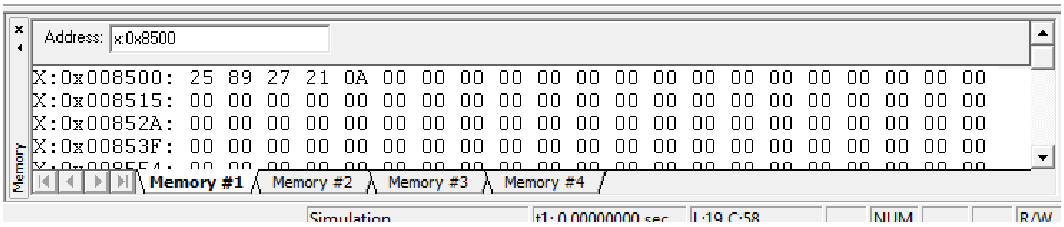
Address: x:0x8500

```
X:0x008500: 85 95 75 25 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00
```

1e)	Write an ALP to find smallest number in the array.

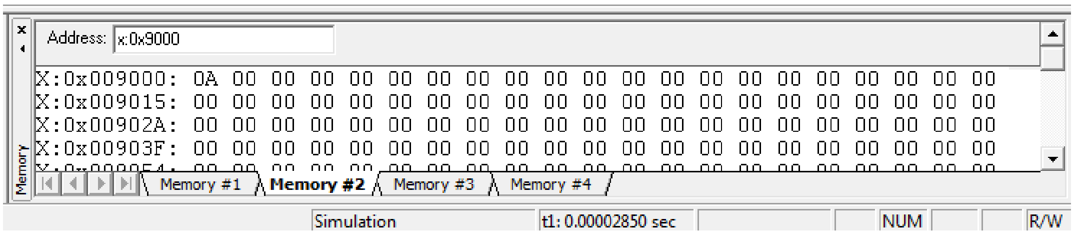
Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV DPTR,#8500H	;dptr=8500h
	MOV R0,#04H	;load counter as 04h (n-1) in register r0
	MOVX A,@DPTR	;contents of memory location pointed by DPTR are copied in reg A
	MOV B,A	; move the content of A into B register
AGAIN:	INC DPTR	; increment DPTR
	MOVX A,@DPTR	;contents of memory location pointed by DPTR are copied in reg A
	CJNE A,B,NEXT	;compare a and b if not equal jump to next
	AJMP SKIP	;absolute jump to skip
NEXT:	JNC SKIP	;jump if no carry to skip
	MOV B,A	;if no carry then copy contents of register A in B
SKIP:	DJNZ R0,AGAIN	;decrement register r0 if not equal to zero jump to again
	MOV A,B	;copy the contents of register B in A
	MOV DPTR,#9000H	;dptr=9000h
	MOVX @DPTR,A	;store the smallest number in 9000h
	LCALL 0003H	;end of asm file

Before Execution



```
Address: x:0x8500
X:0x008500: 25 89 27 21 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution

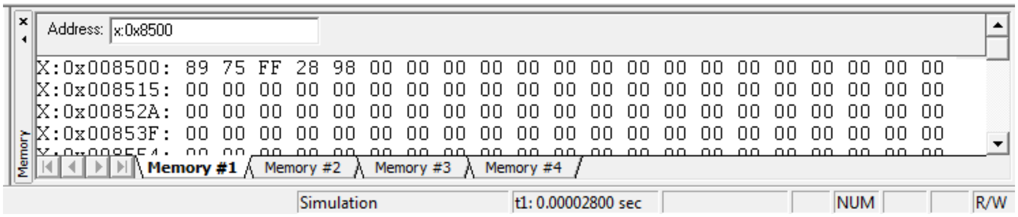


```
Address: x:0x9000
X:0x009000: 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x009015: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00902A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00903F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

1f)	Write an ALP to find largest number in an array.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV DPTR,#8500H	;dptr=8500h
	MOV R0,#04H	;load counter as 04h (n-1) in register r0
	MOVX A,@DPTR	;contents of memory location pointed by DPTR are copied in reg A
	MOV B,A	; move the content of A into B register
AGAIN:	INC DPTR	; increment DPTR
	MOVX A,@DPTR	contents of memory location pointed by DPTR are copied in reg A
	CJNE A,B,NEXT	;compare a and b if not equal jump to next
	AJMP SKIP	;absolute jump to skip
NEXT:	JC SKIP	;jump if carry to skip
	MOV B,A	;if no carry then copy contents of register A in B
SKIP:	DJNZ R0,AGAIN	;decrement register r0 if not equal to zero jump to again
	MOV A,B	;copy the contents of register B in A
	MOV DPTR,#9000H	;dptr=9000h
	MOVX @DPTR,A	;store the largest number in 9000h
	LCALL 0003H	;end of asm file

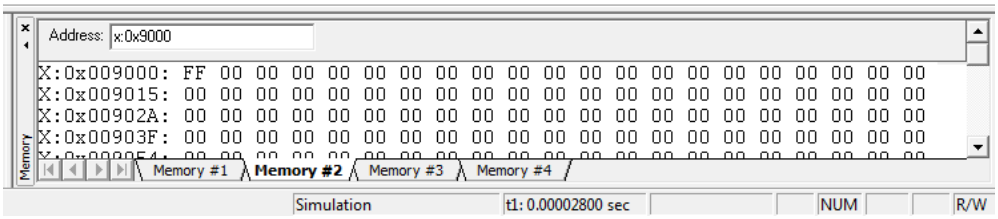
Before Execution



Address: x:0x8500

```
X:0x008500: 89 75 FF 28 98 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution



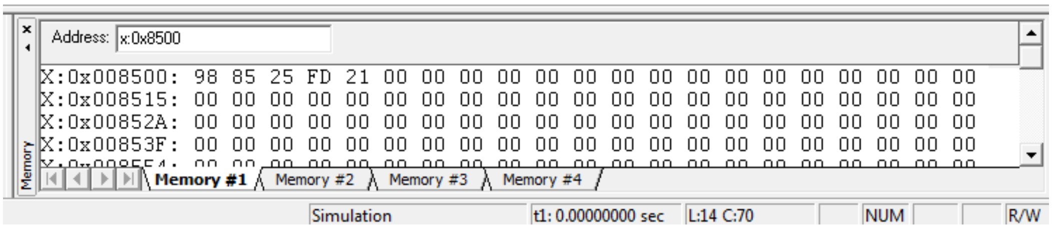
Address: x:0x9000

```
X:0x009000: FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x009015: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00902A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00903F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

1g)	Write an ALP to arrange numbers in ascending order.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV R0,#04H	;load the outer counter as 04h (n-1) in register r0
AGAIN:	MOV DPTR,#8500H	;dptr=8500h
	MOV R1,#04H	;load the inner counter as 04h (n-1) in register r1
BACK:	MOV R2,DPL	;dptr=contents of r2
	MOVX A,@DPTR	;copy the contents of memory location pointed by reg DPTR into reg A
	MOV B,A	;copy the contents of register A in register B
	INC DPTR	;increment the dptr
	MOVX A,@DPTR	;copy the contents of memory location pointed by dptr into register a
	CJNE A,B,NEXT	;compare a and b if equal jump to next
	AJMP SKIP	;absolute jump to skip
NEXT:	JNC SKIP	;jump if no carry to skip
	MOV DPL,R2	;DPL=contents of register r2
	MOVX @DPTR,A	;copy the contents of reg A into memory location pointed by reg DPTR
	INC DPTR	;increment DPTR
	MOV A,B	;copy the contents of register B in register A
	MOVX @DPTR,A	;copy the contents of register a into memory location pointed by DPTR
SKIP:	DJNZ R1, BACK	;decrement register r1 if not equal to zero jump to Back
	DJNZ R0,AGAIN	;decrement register r0 if not equal to zero jump to again
	LCALL 0003H	;end of asm file

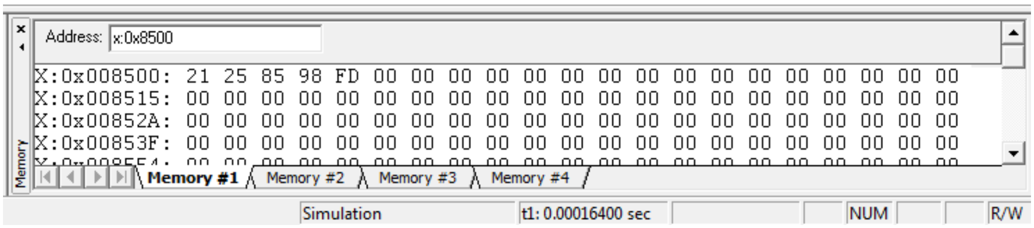
Before Execution



Address: x:0x8500

```
X:0x008500: 98 85 25 FD 21 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution



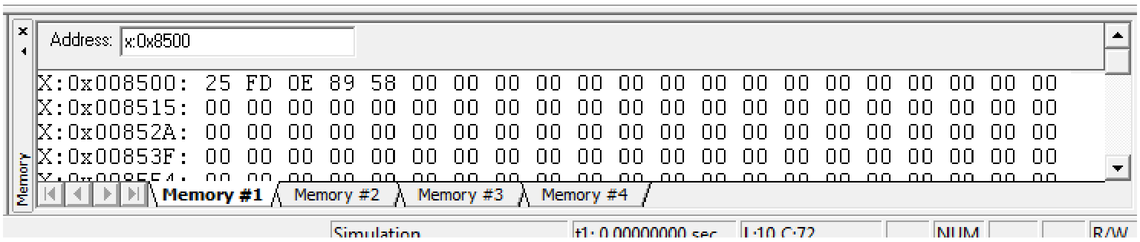
Address: x:0x8500

```
X:0x008500: 21 25 85 98 FD 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

1h)	Write an ALP to arrange numbers in descending order.

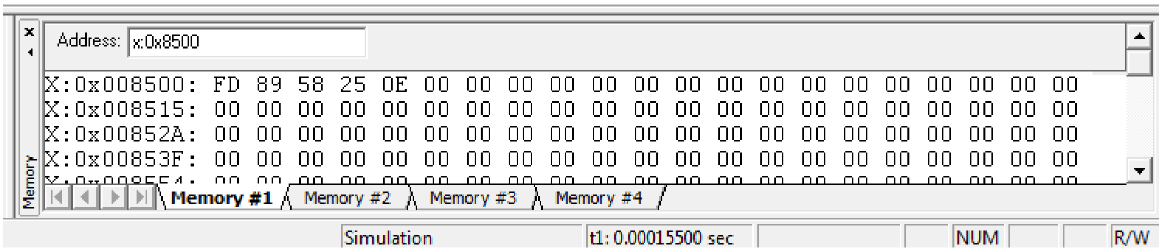
Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV R0,#04H	;load the outer counter as 04h (n-1) in register r0
AGAIN:	MOV DPTR,#8500H	;dptr=8500h
	MOV R1,#04H	;load the inner counter as 04h (n-1) in register r1
BACK:	MOV R2,DPL	;dptr=contents of r2
	MOVX A,@DPTR	;copy the contents of memory location pointed by DPTR into reg A
	MOV B,A	;copy the contents of register A in register B
	INC DPTR	;increment the dptr
	MOVX A,@DPTR	;copy the contents of memory location pointed by reg DPTR into reg A
	CJNE A,B,NEXT	;compare a and b if equal jump to next
	AJMP SKIP	;absolute jump to skip
NEXT:	JC SKIP	;jump if carry to skip
	MOV DPL,R2	;DPL=contents of register r2
	MOVX @DPTR,A	;copy the contents of reg A into memory location pointed by reg DPTR
	INC DPTR	;increment DPTR
	MOV A,B	;copy the contents of register B in register A
	MOVX @DPTR,A	;copy the contents of register a into memory location pointed by DPTR
SKIP:	DJNZ R1, BACK	;decrement register r1 if not equal to zero jump to Back
	DJNZ R0,AGAIN	;decrement register r0 if not equal to zero jump to again
	LCALL 0003H	;end of asm file

Before Execution


Address: x:0x8500

```
X:0x008500: 25 FD 0E 89 58 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution


Address: x:0x8500

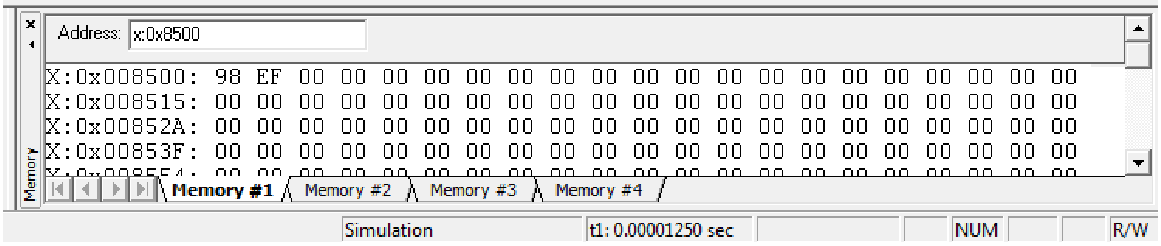
```
X:0x008500: FD 89 58 25 0E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

2	**Arithmetic instructions: Addition, subtraction, multiplication and division. Square and cube operations for 16 bit numbers.**
2a)	Write an ALP to find addition of two 8 bit numbers.
2b)	Write an ALP to find subtraction of two 8 bit numbers.
2c)	Write an ALP to find multiplication of two 8 bit numbers.
2d)	Write an ALP to find division of two 8 bit numbers.
2e)	Write an ALP to square of a 8 bit numbers.
2f)	Write an ALP to cube of a 8 bit numbers.
2g)	Write an ALP to find addition of two 16 bit numbers.
2h)	Write an ALP to find subtraction of two 16 bit numbers.
2i)	Write an ALP to square of a 16 bit numbers.

2a)	Write an ALP to find addition of two 8 bit numbers.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV DPTR,#8500H	;dptr=contents of 8500h
	MOVX A,@DPTR	;get the lower byte from memory location 8103h in register A
	MOV R0,A	;store in register r0
	INC DPTR	;increment dptr
	MOVX A,@DPTR	;get the lower byte from memory location 8101h in register A
	ADDC A, R0	;add first lower byte and second lower byte
	JNC NEXT	;jump if no carry to here
	INC R5	;if carry then increment register r5
NEXT :	MOV DPTR,#9001H	;DPTR=9001h
	MOVX @DPTR,A	;dptr=store the lower byte in memory location 9101h
	MOV A,R5	;store the contents of r5(carry) in register A
	DEC DPL	;decrement DPL i.e. DPTR=9000h
	MOVX @DPTR,A	;store the carry in memory location in 9000h
	LCALL 0003H	;end of asm file

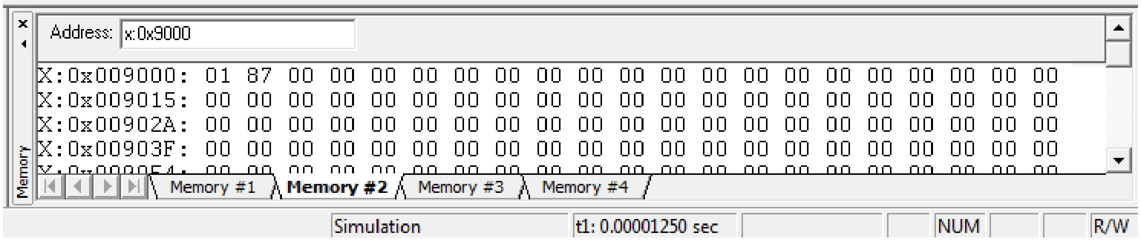
Before Execution



Address: x:0x8500

```
X:0x008500: 98 EF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution



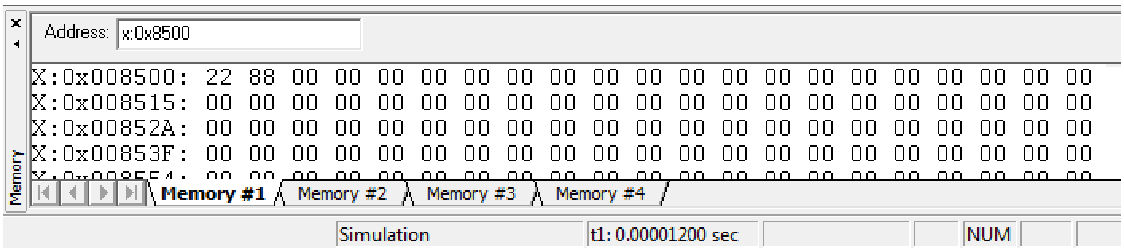
Address: x:0x9000

```
X:0x009000: 01 87 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x009015: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00902A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00903F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

2b)	Write an ALP to find subtraction of two 8 bit numbers.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV DPTR,#8500H	;dptr=contents of 8500h
	MOVX A,@DPTR	;get the lower byte from memory location 8500h in register A
	MOV R0,A	;store in register r0
	INC DPTR	; increment dptr
	MOVX A,@DPTR	;get the lower byte from memory location 8501h in register A
	SUBB A,R0	;sub content of r0 from a
	JNC NEXT	;jump if no carry to here
	INC R5	;if carry then increment register r5
NEXT:	MOV DPTR,#9001H	;DPTR=9001h
	MOVX @DPTR,A	;dptr=store the lower byte in memory location 9001h
	MOV A,R5	;store the contents of r5(carry) in register A
	DEC DPL	;decrement DPL i.e. DPTR=9100h
	MOVX @DPTR,A	;store the carry in memory location in 9100h
	LCALL 0003H	;end of asm file

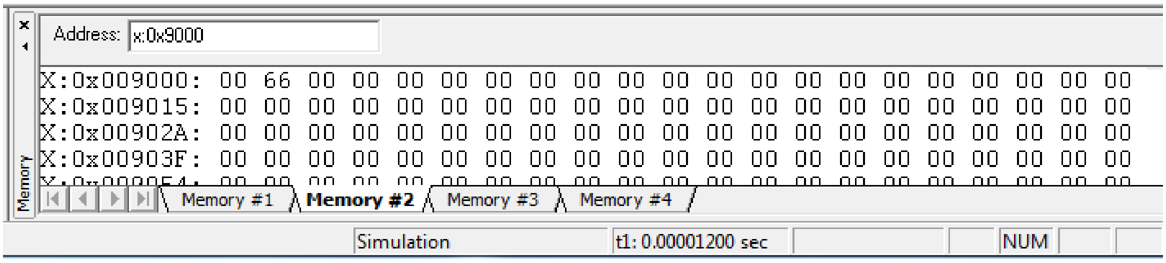
Before Execution



Address: x:0x8500

```
X:0x008500: 22 88 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution



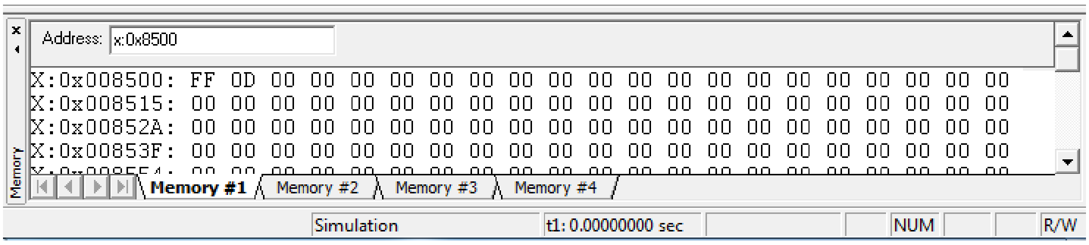
Address: x:0x9000

```
X:0x009000: 00 66 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x009015: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00902A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00903F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

2c)	Write an ALP to find multiplication of two 8 bit numbers.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV DPTR,#8500H	;dptr=8500h
	MOVX A,@DPTR	;store the number in register a from memory location 8500h
	MOV B,A	;copy the number in register B
	INC DPTR	;dptr=8501h
	MOVX A,@DPTR	; store the number in register a from memory location 8501h
	MUL AB	;multiply A and B
	INC DPTR	;increment DPTR
	INC DPTR	;increment DPTR
	MOVX @DPTR,A	;store the higher byte in memory location in 8501h
	MOV A,B	;copy the contents of register b in register a
	DEC DPL	;decrement DPL
	MOVX @DPTR,A	;store the lower byte in memory location 8502h
	LCALL 0003H	;end of asm file

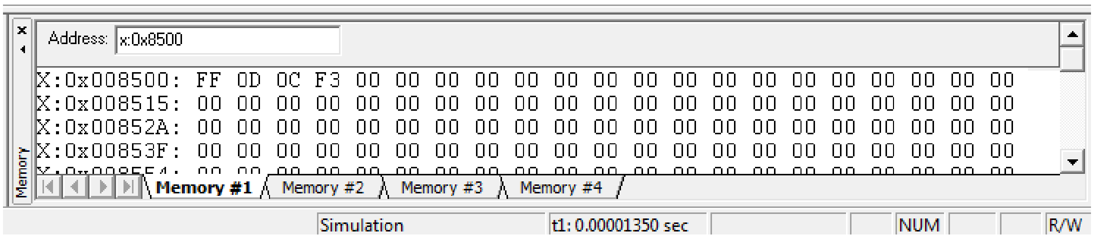
Before Execution



Address: x:0x8500

```
X:0x008500: FF 0D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution



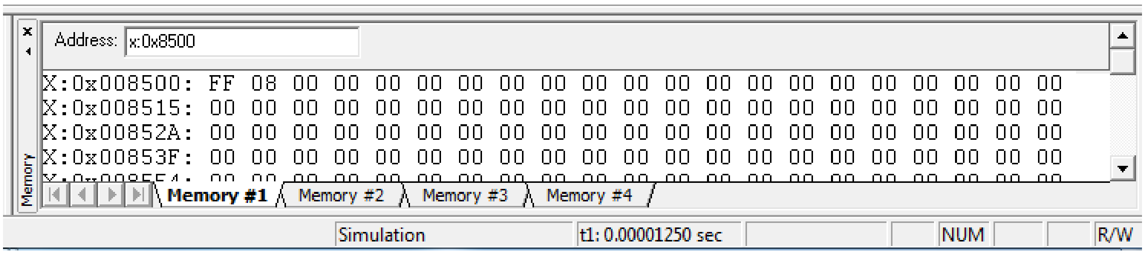
Address: x:0x8500

```
X:0x008500: FF 0D 0C F3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

2d)	Write an ALP to find division of two 8 bit numbers.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV DPTR,#8501H	;dptr=8501h
	MOVX A,@DPTR	;store the number in register a from memory location 8500h
	MOV B,A	;copy the number in register B
	DEC DPL	; dptr=8500h
	MOVX A, @DPTR	store the number in register a from memory location 8501h
	DIV AB	;divide A by B
	MOV DPTR, #9500H	;increment DPTR
	MOVX @DPTR,A	;store the quotient in memory location in 9500h
	MOV A,B	;store the reminder in register b
	INC DPTR	;increment DPTR
	MOVX @DPTR,A	;store the reminder in memory location 9501h
	LCALL 0003H	;end of asm file

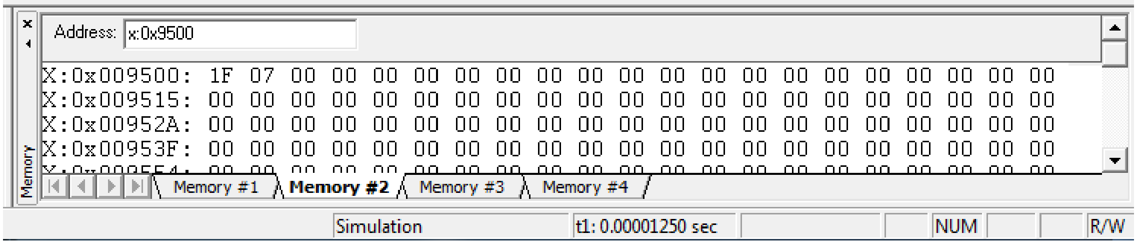
Before Execution



Address: x:0x8500

```
X:0x008500: FF 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution



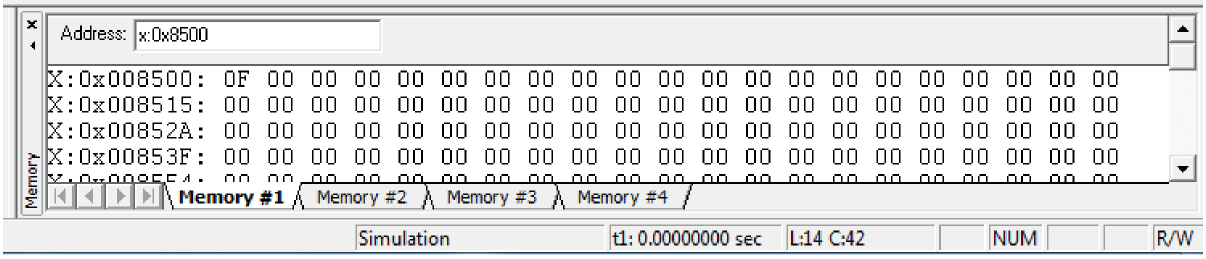
Address: x:0x9500

```
X:0x009500: 1F 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x009515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00952A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00953F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

2e)	Write an ALP to square of a 8 bit numbers.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV DPTR,#8500H	;dptr=8500h
	MOVX A,@DPTR	;store the number in register a from memory location 8500h
	MOV B,A	;copy the number in register B
	MUL AB	;multiply A and B
	INC DPTR	;increment DPTR
	MOVX @DPTR,A	;store the higher byte in memory location in 8501h
	MOV A,B	;copy the contents of register b in register a
	INC DPTR	;increment DPTR
	MOVX @DPTR,A	;store the lower byte in memory location 8502h
	LCALL 0003H	;end of asm file

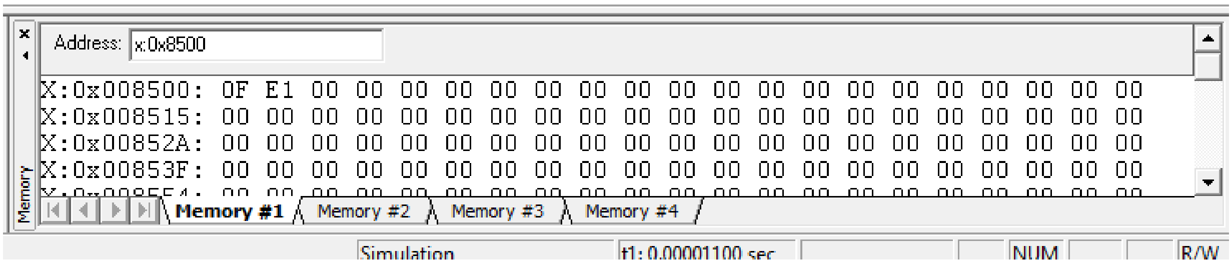
Before Execution



Address: x:0x8500

```
X:0x008500: 0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution



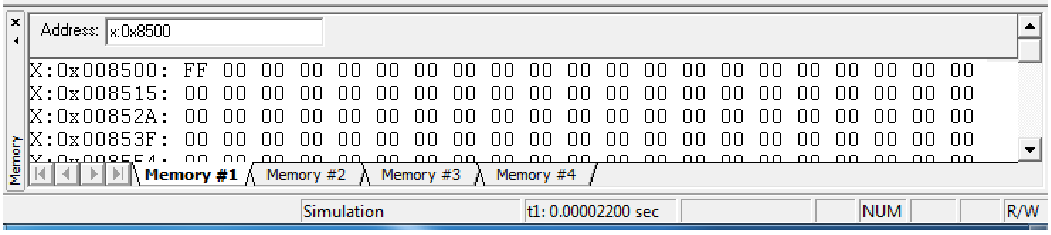
Address: x:0x8500

```
X:0x008500: 0F E1 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

2f)	Write an ALP to cube of a 8 bit numbers.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV DPTR,#8500H	;dptr=8500h
	MOVX A,@DPTR	;store the number in register a from memory location 8500h
	MOV B,A	;copy the number in register B
	MOV R3,A	;save in register r3
	MUL AB	;multiply A and B
	MOV R4,B	;save the first lower byte in r4
	MOV B,R3	;load once again register B with number which has to cube
	MUL AB	;multiply first higher byte stored in register A with B
	MOV DPTR,#9002H	;Load DPTR by 9002H
	MOVX @DPTR,A	; store the result in DPTR
	MOV R2,B	;save the second lower byte in r2
	MOV A,R4	;store the first lower byte in register in a from r4
	MOV B,R3	;store the contents of register in B
	MUL AB	;multiply register A and B
	ADD A,R2	;add register A and register r2
	DEC DPL	; decrement dptr
	MOVX @DPTR,A	;store the higher byte in memory location in 9001h
	MOV A,#00H	;a=00h
	ADDC A,B	;add register A and B with carry
	DEC DPL	;decrement dpl
	MOVX @DPTR,A	;copy the contents of A in memory location pointed by DPTR
	LCALL 0003H	;end of asm file

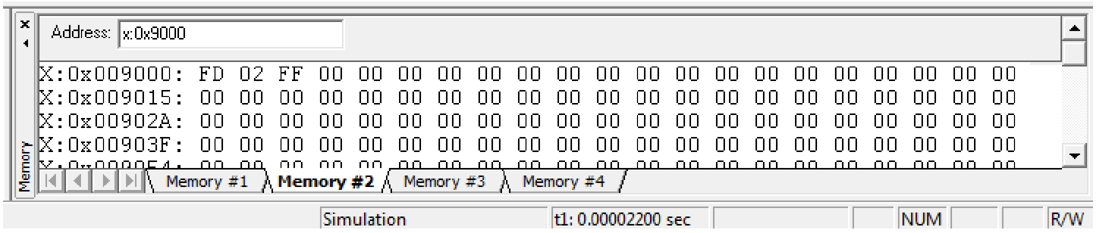
Before Execution



Address: x:0x8500

```
X:0x008500: FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution



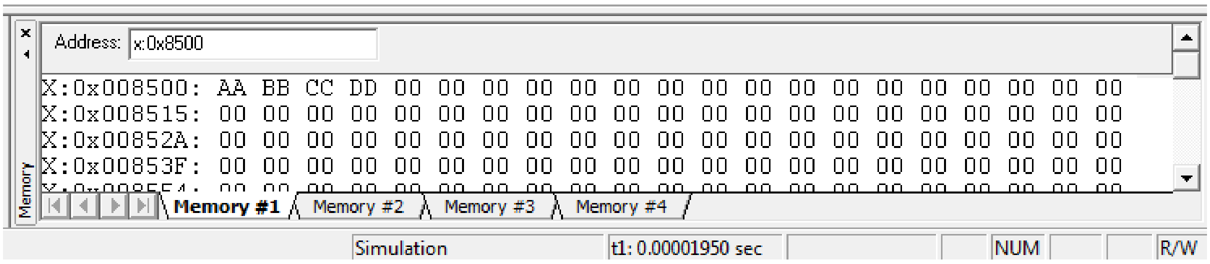
Address: x:0x9000

```
X:0x009000: FD 02 FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x009015: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00902A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00903F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

2g)	Write an ALP to find addition of two 16 bit numbers.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV DPTR,#8503H	;dptr=contents of 8503h
	MOVX A,@DPTR	;get the lower byte from memory location 8503h in register A
	MOV R1,A	;store content of A in register r1
	MOV DPTR,#8501H	;dptr=contents of 8501h
	MOVX A,@DPTR	;get the lower byte from memory location 8501h in register A
	ADD A, R1	;A=A+R1
	MOV DPTR,#9002H	; DPTR= 9002h
	MOVX @DPTR,A	; 9002H=A
	MOV DPTR,#8502H	; DPTR= 8502h
	MOVX A,@DPTR	;A= content of 8502h
	MOV R2,A	; R2=A
	MOV DPTR,#8500H	; DPTR= 8500h
	MOVX A,@DPTR	; A=content of 8500H
	ADDC A, R2	;A=A+R2+C
	JNC NEXT	;jump if no carry to next
	INC R7	;if carry then increment register r7
NEXT:	MOV DPTR,#9001H	;DPTR=9001h
	MOVX @DPTR,A	;dptr=store the lower byte in memory location 9001h
	DEC DPL	;decrement DPL i.e. DPTR=9000h
	MOV A,R7	; A=R7
	MOVX @DPTR,A	;store the carry in memory location in 9000h
	LCALL 0003H	;end of asm file

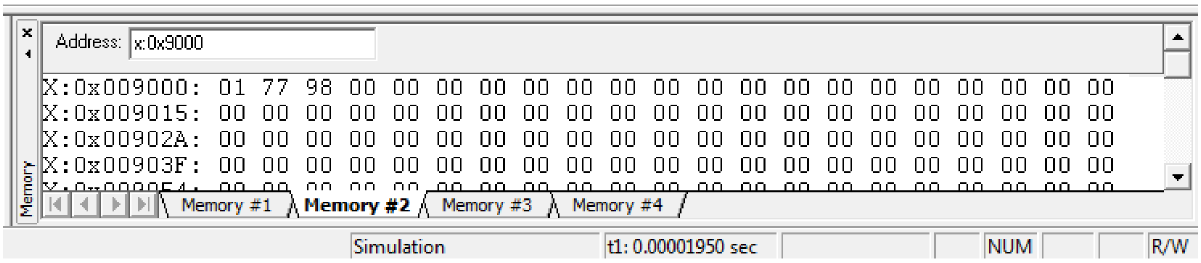
Before Execution



Address: x:0x8500

```
X:0x008500: AA BB CC DD 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution



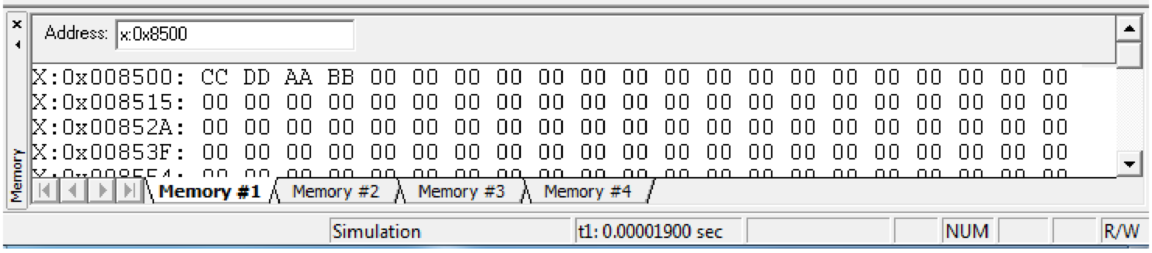
Address: x:0x9000

```
X:0x009000: 01 77 98 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x009015: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00902A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00903F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

2h)	Write an ALP to find subtraction of two 16 bit numbers.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV DPTR,#8503H	;dptr=contents of 8503h
	MOVX A,@DPTR	;get the lower byte from memory location 8503h in register A
	MOV R1,A	;store content of A in register r1
	MOV DPTR,#8501H	;dptr=contents of 8501h
	MOVX A,@DPTR	;get the lower byte from memory location 8501h in register A
	SUBB A, R1	;A=A-R1
	MOV DPTR,#9002H	; DPTR= 9002h
	MOVX @DPTR,A	; 9002H=A
	MOV DPTR,#8502H	; DPTR= 8502h
	MOVX A,@DPTR	;A= content of 8502h
	MOV R2,A	; R2=A
	MOV DPTR,#8500H	; DPTR= 8500h
	MOVX A,@DPTR	; A=content of 8500H
	SUBB A, R2	;A=A+R2+C
	JNC NEXT	;jump if no carry to next
	INC R7	;if carry then increment register r7
NEXT:	MOV DPTR,#9001H	;DPTR=9001h
	MOVX @DPTR,A	;dptr=store the lower byte in memory location 9001h
	DEC DPL	;decrement DPL i.e. DPTR=9000h
	MOV A,R7	; A=R7
	MOVX @DPTR,A	;store the carry in memory location in 9000h
	LCALL 0003H	;end of asm file

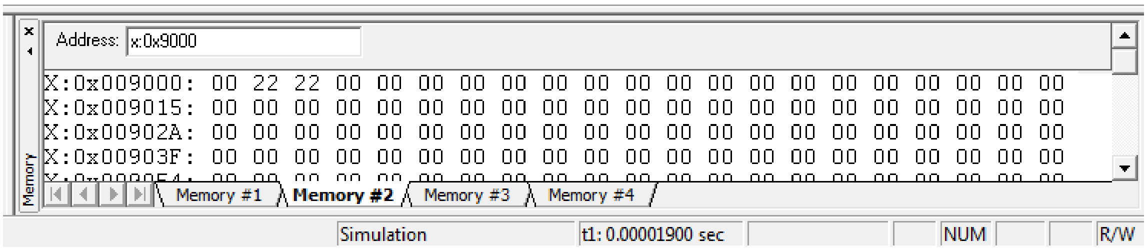
Before Execution



Address: x:0x8500

```
X:0x008500: CC DD AA BB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution



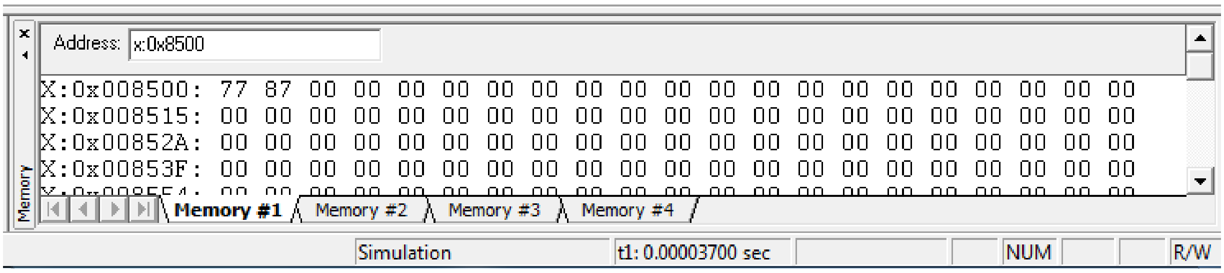
Address: x:0x9000

```
X:0x009000: 00 22 22 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x009015: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00902A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00903F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

2i)	Write an ALP to square of a 16 bit numbers.

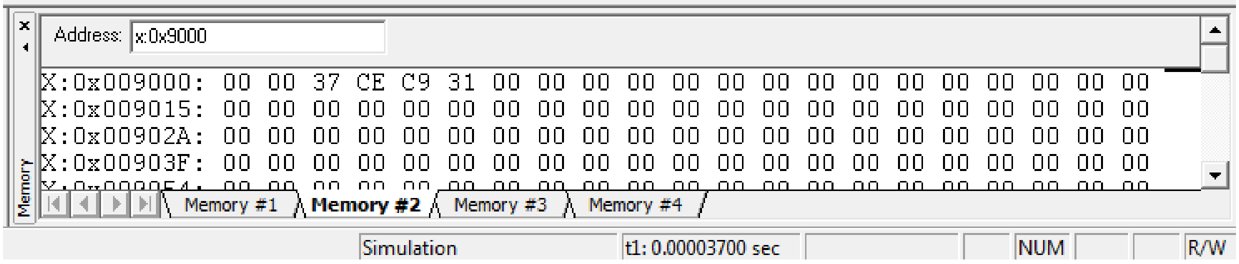
Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV DPTR, #8501H	;dptr=8501h
	MOVX A, @DPTR	;store the number in register a from memory location 8501h
	MOV R0, A	; store the content reg a in r0
	DEC DPL	; dpl=dpl-1
	MOVX A, @DPTR	;store the number in register a from memory location 8500h
	MOV R1, A	; store the content reg a in r1
	MOV A, R0	; store the content reg r0 in a
	MOV B,A	; store the content reg a in b
	MUL AB	; a=axb
	MOV DPTR, #9005H	;dptr=9005h
	MOVX @DPTR, A	;store the number in register a from memory location 9005h
	MOV R2, B	; store the content reg b in r2
	MOV A, R0	; store the content of reg r0 in a
	MOV B, R1	; store the content reg r1 in b
	MUL AB	; a=axb
	ADD A, R2	; add content r2 with a
	MOV R3, A	; store the content of reg a in r3
	MOV R2, B	; store the content of reg b in r2
	CLR A	; a=0
	ADDC A, R2	; a=a+c+r2
	MOV R2,A	
	MOV A, R0	; store the content of reg r0 in a
	MOV B, R1	; store the content of reg r1 in b
	MUL AB	; a=axb
	MOV R7, A	; store the content of reg a in r7
	MOV R4, B	; store the content of reg b in r2
	MOV A, R1	; store the content of reg r1 in a
	MOV B, R1	; store the content of reg r1 in b
	MUL AB	; a=axb
	ADD A, R2	; a=a+r2
	MOV R6, A	; store the content of reg a in r6
	MOV A,B	; store the content of reg b in a
	MOV R5,A	; store the content of reg a in r5
	MOV A, R3	; store the content of reg r3 in a
	ADD A, R7	;a=a+r7
	DEC DPL	;dpl=dpl-1
	MOVX @DPTR, A	;store the number in register a from memory location
	MOV A, R4	; store the content of reg r4 in a
	ADDC A, R6	; a=a+c+r6
	DEC DPL	;dpl=dpl-1
	MOVX @DPTR, A	;store the number in register a from memory location
	MOV A, R5	; store the content of reg r5 in a
	DEC DPL	;dpl=dpl-1
	MOVX @DPTR,A	;store the number in register a from memory location
	LCALL 0003H	end

Before Execution


Address: x:0x8500

```
X:0x008500: 77 87 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution


Address: x:0x9000

```
X:0x009000: 00 00 37 CE C9 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x009015: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00902A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00903F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

3	**Counter**
3a)	Write an ALP to generate Hex up counter.
3b)	Write an ALP to generate Hex down counter.
3c)	Write an ALP to generate BCD up counter
3d)	Write an ALP to generate BCD down counter.

3a)	Write an ALP to generate Hex up counter.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV A,#00H	;load the accumulator with 00H
UP:	LCALL DELAY	;call delay
	INC A	; increment accumulator by 1
	CJNE A,#40H,UP	;compare accumulator with 40h if not equal jump up
	LCALL 0003H	; end
DELAY:	MOV R0,#0FFH	; load r0 by FFH
BACK1:	MOV R1,#0FFH	; load r1 by FFH
BACK:	MOV R2,#0FFH	; load r2 by FFH
HERE:	DJNZ R2,HERE	; decrement r2 if not equal to zero jump here
	DJNZ R1, BACK	;decrement r1 if not equal to zero jump Back
	DJNZ R0,BACK1	;decrement r0 if not equal to zero jump Back1
	RET	;return to main

3b)	Write an ALP to generate Hex down counter.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV A,#020H	;load the accumulator with 20H
UP:	LCALL DELAY	;call delay
	DEC A	; decrement accumulator by 1
	CJNE A,#00H,UP	;compare accumulator with 00H if not equal jump up
	LCALL 0003H	; end
DELAY:	MOV R0,#0FFH	; load r0 by FFH
BACK1:	MOV R1,#0FFH	; load r1 by FFH
BACK:	MOV R2,#0FFH	; load r2 by FFH
HERE:	DJNZ R2,HERE	; decrement r2 if not equal to zero jump here
	DJNZ R1, BACK	;decrement r1 if not equal to zero jump Back
	DJNZ R0,BACK1	;decrement r0 if not equal to zero jump Back1
	RET	;return to main

3c)	Write an ALP to generate BCD up counter

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV A,#00H	;load the accumulator with 00H
UP:	LCALL DELAY	;call delay
	ADD A,#01H	; add accumulator with 0 1H
	DA A	;decimal adjust accumulator
	CJNE A,#30H,UP	;compare accumulator with 30h if not equal jump up
	LCALL 0003H	; end
DELAY:	MOV R0,#0FFH	; load r0 by FFH
BACK1:	MOV R1,#0FFH	; load r1 by FFH
BACK:	MOV R2,#0FFH	; load r2 by FFH
HERE:	DJNZ R2,HERE	; decrement r2 if not equal to zero jump here
	DJNZ R1, BACK	;decrement r1 if not equal to zero jump Back
	DJNZ R0,BACK1	;decrement r0 if not equal to zero jump Back1
	RET	;return to main

3d)	Write an ALP to generate BCD down counter.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV A,#30H	;load the accumulator with 30H
UP:	LCALL DELAY	;call delay
	ADD A,#99H	; add accumulator with 99H
	DA A	;decimal adjust accumulator
	CJNE A,#00H,UP	;compare accumulator with 00h if not equal jump up
	LCALL 0003H	; end
DELAY:	MOV R0,#0FFH	; load r0 by FFH
BACK1:	MOV R1,#0FFH	; load r1 by FFH
BACK:	MOV R2,#0FFH	; load r2 by FFH
HERE:	DJNZ R2,HERE	; decrement r2 if not equal to zero jump here
	DJNZ R1, BACK	;decrement r1 if not equal to zero jump Back
	DJNZ R0,BACK1	;decrement r0 if not equal to zero jump Back1
	RET	;return to main

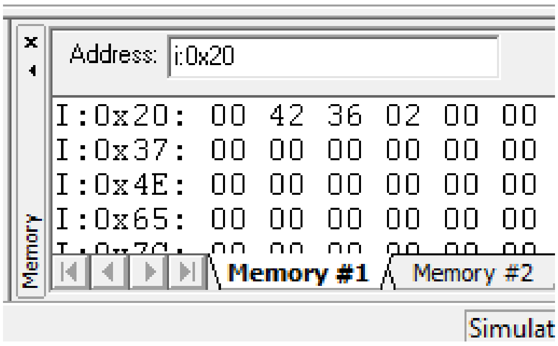
4	**Boolean and logical instructions (bit manipulation).**

4a)	Write an ALP to compute the following.
	IF X=0; THEN NUM1 (AND) NUM2, IF X=1; THEN NUM1 (OR) NUM2,
	IF X=2; THEN NUM1 (XOR) NUM2, ELSE RES =00, RES IS 23H LOCATION
	Using logical instructions in byte level.

4a)	Write an ALP to compute the following.
	IF X=0; THEN NUM1 (AND) NUM2, IF X=1; THEN NUM1 (OR) NUM2,
	IF X=2; THEN NUM1 (XOR) NUM2, ELSE RES =00, RES IS 23H LOCATION
	Using logical instructions in byte level.

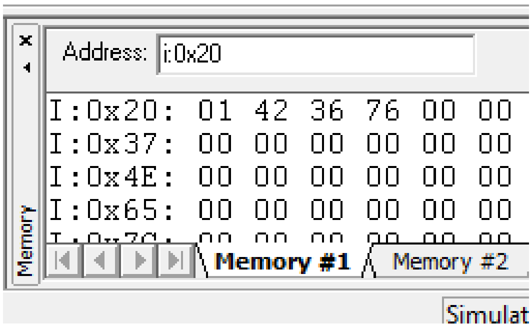
Label	Opcode and Operands	Comments
	ORG 0000H	
	LJMP 8000H	
	ORG 8000H	
	MOV A, 20H	; Store the content of 20H in A
	MOV R1, A	; Store the value of A in R1
	MOV A, 21H	; Store the content of 21H in A
	CJNE R1, #0, CKOR	; compare the content of R1 with zero, if yes go to ckor otherwise go to next instruction
	ANL A, 22H	; A=A and content of 22H
	MOV 23H, A	; move the content of A in 23H
	SJMP END1	;End
CKOR:	CJNE R1, #1, CKXOR	; if R1 is not equal to 1 go to ckxor, otherwise go to next instruction
	ORL A, 22H	; Or the content of A with content of 22H
	MOV 23H, A	; move the content of A in 23H
	SJMP END1	;end
CKXOR:	CJNE R1, #2H, OTHER	; if R1 is not equal to 2 go to other, otherwise go to next instruction
	XRL A, 22H	; Ex-or the content of A with content of 22H
	MOV 23H, A	; move the content of A in 23H
	SJMP END1	;end
OTHER:	CLR A	; A=0
END1:	MOV 23H, A	; move the content of A in 23H
HERE:	SJMP HERE	;unconditional jump to here.
	END	;end

When X=0



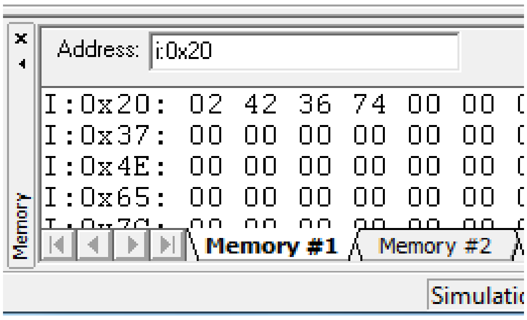
```
Address: i:0x20
I:0x20: 00 42 36 02 00 00
I:0x37: 00 00 00 00 00 00
I:0x4E: 00 00 00 00 00 00
I:0x65: 00 00 00 00 00 00
```

When X=1



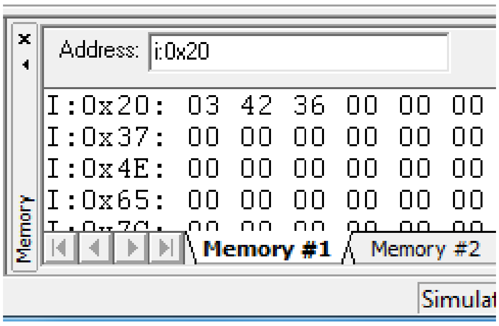
```
Address: i:0x20
I:0x20: 01 42 36 76 00 00
I:0x37: 00 00 00 00 00 00
I:0x4E: 00 00 00 00 00 00
I:0x65: 00 00 00 00 00 00
```

When X=2



```
Address: i:0x20
I:0x20: 02 42 36 74 00 00
I:0x37: 00 00 00 00 00 00
I:0x4E: 00 00 00 00 00 00
I:0x65: 00 00 00 00 00 00
```

When X=3



```
Address: i:0x20
I:0x20: 03 42 36 00 00 00
I:0x37: 00 00 00 00 00 00
I:0x4E: 00 00 00 00 00 00
I:0x65: 00 00 00 00 00 00
```

5.	**Conditional call and return instructions.**
5a)	Write a program to toggle all the bits of port 1 continuously by sending the values 55H and AAH using call and return instructions.
5b)	Write an ALP to find factorial of a number using call and return instructions.

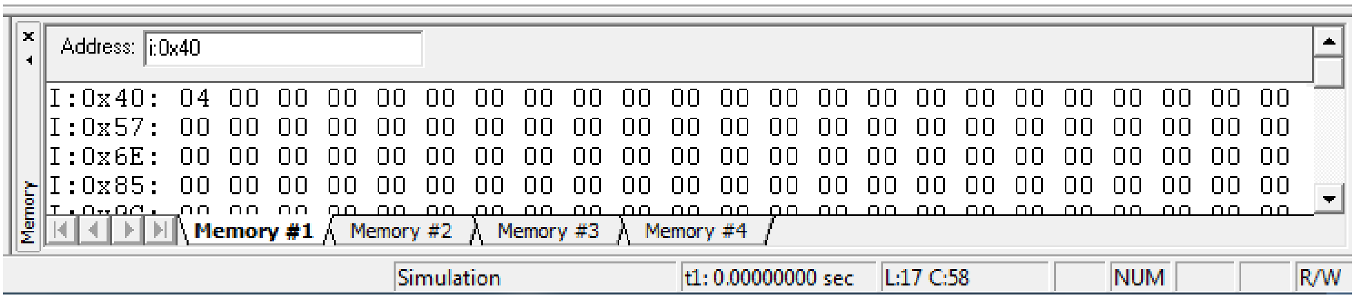
5a)	Write a program to toggle all the bits of port 1 continuously by sending the values 55H and AAH using call and return instructions.

Label	Opcode and Operands	Comments
	ORG 0000H	
	LJMP 8000H	
	ORG 8000H	
	MOV R5, #05H	; r5=05h
GO:	MOV A, #55H	;a=55h
	MOV P1, A	; ;store the number in from memory location
	LCALL 300H	; Long call to 300h i.e delay program
	MOV A, #0AAH	; move the content aaH to A
	MOV P1, A	; move the content of A in P1
	LCALL 300H	; Long call to 300h i.e delay program
	DJNZ R5, GO	; R5=R5-1, if R5 is not equal to zero then jump to go
	LCALL 0003H	; Long call to 300h i.e delay program
	ORG 300H	; Start delay program
	MOV R0, #0FFH	; R0=FFH
HERE:	MOV R1, #0FFH	
BACK1:	MOV R2, #0FFH	; R1=FFH
BACK:	DJNZ R2, BACK	; R2=R2-1, if R2 is not equal to zero then jump to back
	DJNZ R1, BACK1	; R1=R1-1, if R1 is not equal to zero then jump to back1
	DJNZ R0, HERE	; R0=R0-1, if R2 is not equal to zero then jump to here
	RET	; Return to main program

5b)	Write an ALP to find factorial of a number using call and return instructions.

Label	Opcode and Operands	Comments
	ORG 0000H	
	LJMP 8000H	
	ORG 8000H	
	MOV A, 40H	;Move the content of 40H in A
	MOV R0, A	; Move the content of A in R0
	LCALL 9000H	;call 9000h
	LCALL 0003H	; call 3000h
	ORG 9000H	; Originate 9000h
FACT:	CJNE R0, #01H, NEXT	; if R0 is not equal to 1 then jump to next
	RET	; Return to main program
NEXT:	DEC R0	; R0=R0-1
	MOV B, R0	; Move the content of R0 in b
	MUL AB	;A=AxB
	MOV 41H, A	; Move the content of A in R5
	LJMP 9000H	; Go to 9000h
	END	; end

Before Execution

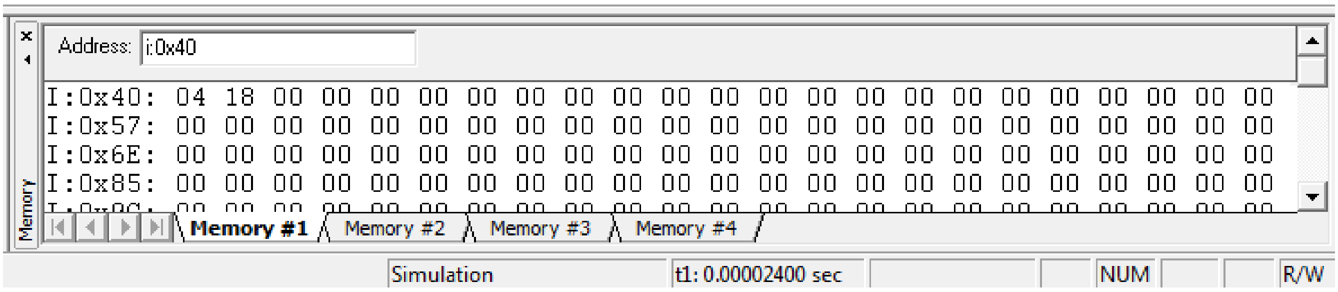


Address: i:0x40

```
I:0x40: 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x57: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x6E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x85: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Simulation | t1: 0.00000000 sec | L:17 C:58 | NUM | R/W

After Execution



Address: i:0x40

```
I:0x40: 04 18 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x57: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x6E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x85: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

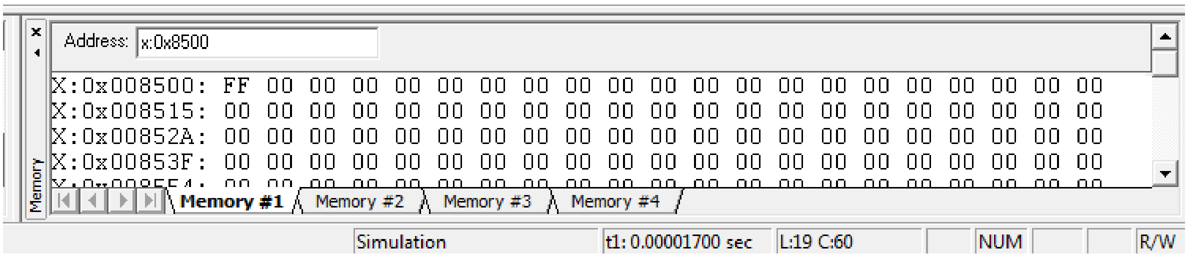
Simulation | t1: 0.00002400 sec | NUM | R/W

6	**Code conversion programs – BCD to ASCII, ASCII to BCD, ASCII to decimal, Decimal to ASCII, Hexa decimal to and Decimal to hexa.**
6a)	Write an ALP to convert hexadecimal number to decimal number.
6b)	Write an ALP to convert decimal number to hexadecimal number.
6c)	Write an ALP to convert packed BCD number to ASCII number.
6d)	Write an ALP to convert ASCII number to BCD number

6a)	Write an ALP to convert hexadecimal number to decimal number.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV DPTR,#8500H	;dptr=8500h
	MOVX A,@DPTR	;copy the contents of memory location pointed by DPTR into register A
	MOV B,#64H	;B=64H
	DIV AB	;division A/B
	INC DPTR	;increment DPTR=8501H
	MOVX @DPTR,A	;copy the contents of memory location pointed by DPTR into register A
	MOV A,B	; move the content of B into A register
	MOV B,#0AH	;Store the number 0AH into register B
	DIV AB	; division A/B
	INC DPTR	; increment DPTR=8502H
	MOVX @DPTR,A	; store the contents of register A in memory location pointed by DPTR
	INC DPTR	;increment DPTR=8503H
	MOV A,B	;move the content of B into A register
	MOVX @DPTR,A	; store the contents of register A in memory location pointed by DPTR
	LCALL 0003H	;end of asm file

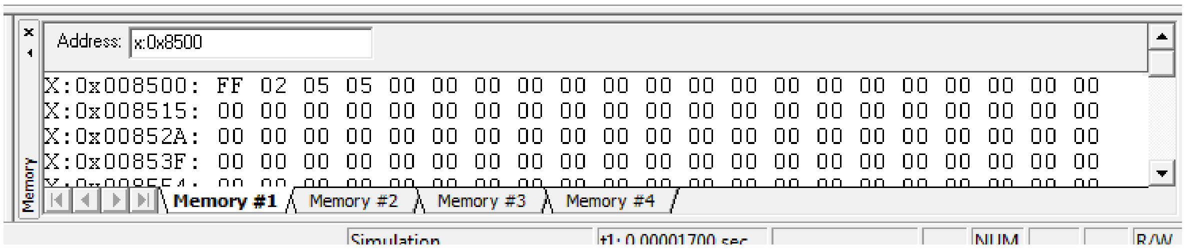
Before Execution



Address: x:0x8500

```
X:0x008500: FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution



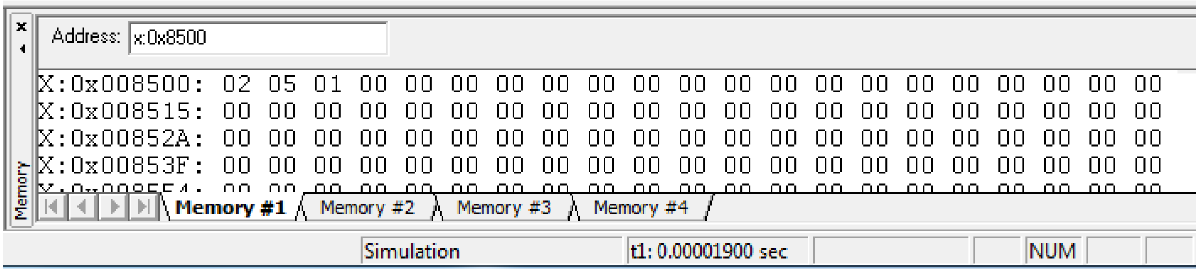
Address: x:0x8500

```
X:0x008500: FF 02 05 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

6b)	Write an ALP to convert decimal number to hexadecimal number.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV DPTR,#8500H	;dptr=8500h
	MOVX A,@DPTR	;copy the contents of memory location pointed by DPTR into reg A
	MOV B,#64H	;B=64H
	MUL AB	;multiply A and B
	MOV R2,A	;store the contents of A in register r2
	INC DPTR	;increment DPTR=8501H
	MOVX A ,@DPTR	;copy the contents of memory location pointed by DPTR into register A
	MOV B,#0AH	;Store the number 0AH into register B
	MUL AB	; division A/B
	MOV B,R2	;store the contents of R2 in register B
	ADD A,B	;add register A and B
	MOV B,A	;store the contents of A in register B
	INC DPTR	; increment DPTR=8502H
	MOVX A,@DPTR	; copy the contents of memory location pointed by DPTR into register A
	ADD A,B	;add register A and B
	INC DPTR	;increment DPTR=8503H
	MOVX @DPTR,A	; store the contents of register A in memory location pointed by DPTR
	LCALL 0003H	;end of asm file

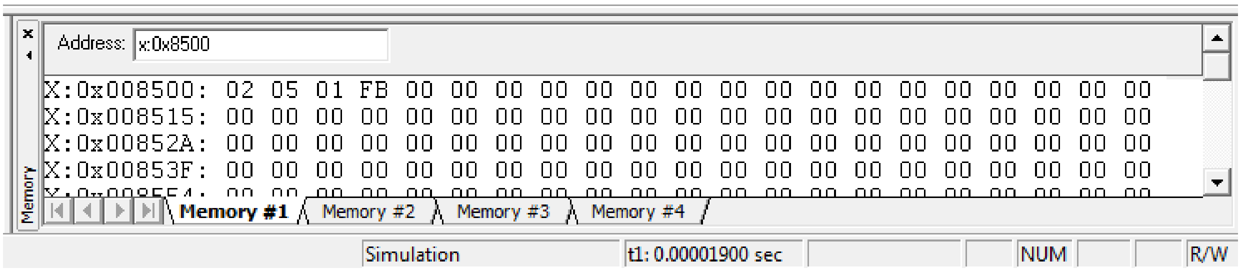
Before Execution



Address: x:0x8500

```
X:0x008500: 02 05 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution



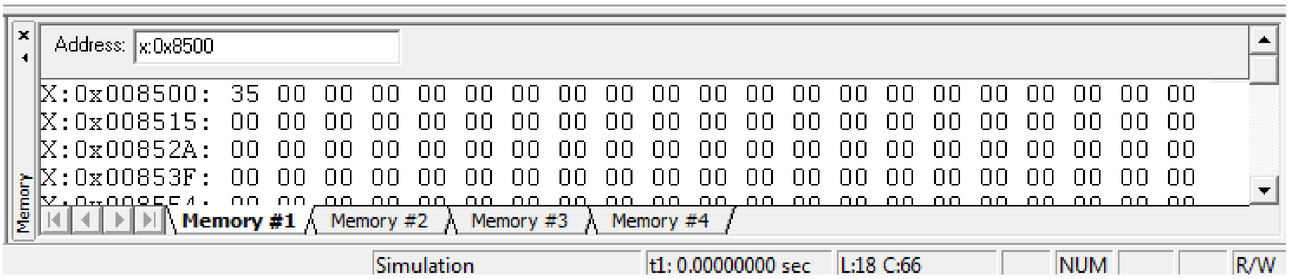
Address: x:0x8500

```
X:0x008500: 02 05 01 FB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

6c)	Write an ALP to convert packed BCD number to ASCII number.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV DPTR,#8500H	;dptr=8500h
	MOVX A,@DPTR	;store the contents of memory location pointed by DPTR into register A
	MOV R0,A	;move the content of A to r0.
	ANL A,#0FH	;make logical AND function with register A and immediate data 0FH
	ORL A,#30H	;make logical OR function with register A and immediate data 30H
	INC DPTR	;increment DPTR
	MOVX @DPTR,A	;save the result in memory location 8501H
	MOV A,R0	;get the once again BCD number in register A
	ANL A,#0F0H	;make logical AND function with reg A and immediate data 0F0h
	SWAP A	;swap the contents of register A
	ORL A,#30H	;make logical OR function with reg A and immediate data 30H
	INC DPTR	;increment DPTR
	MOVX @DPTR,A	;save the result in memory location 8502H
	LCALL 0003H	;end of asm file

Before Execution

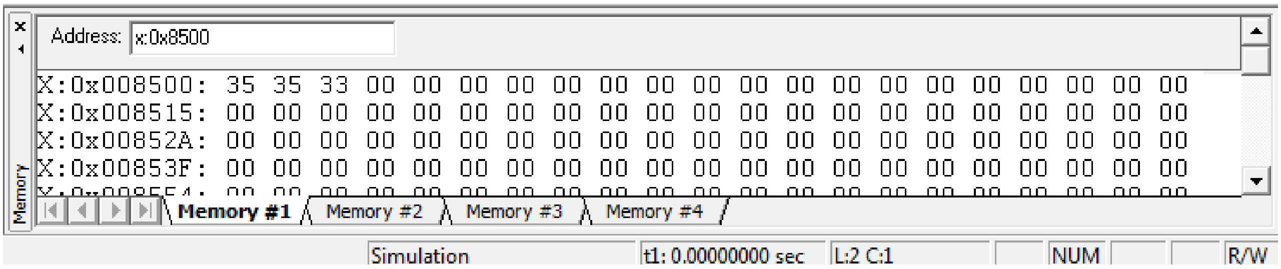


Address: x:0x8500

```
X:0x008500: 35 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Simulation | t1: 0.00000000 sec | L:18 C:66 | NUM | R/W

After Execution



Address: x:0x8500

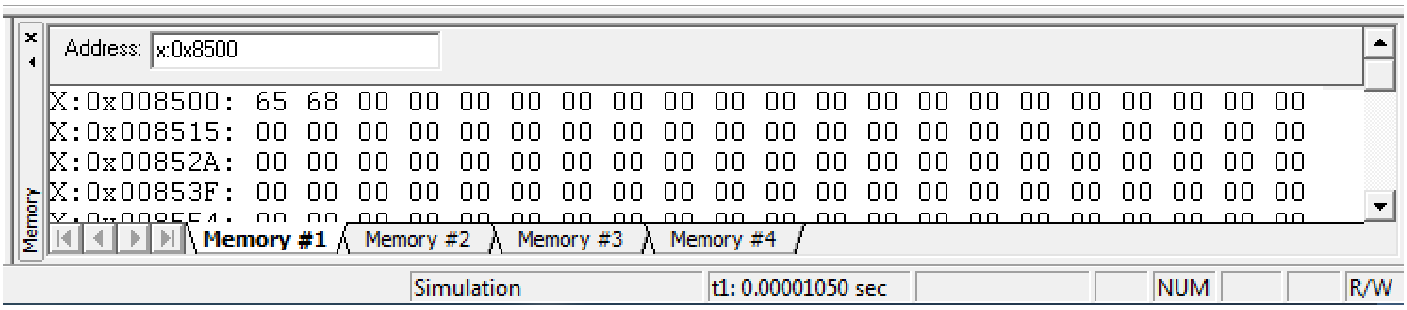
```
X:0x008500: 35 35 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Simulation | t1: 0.00000000 sec | L:2 C:1 | NUM | R/W

6d)	Write an ALP to convert ASCII number to BCD number

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV DPTR,#8500H	;dptr=8500h
	MOVX A,@DPTR	;store the contents of memory location pointed by DPTR into reg A
	ANL A,#0FH	;make logical AND function with register A and immediate data 0FH
	MOV B,A	;store in register B
	INC DPTR	;increment DPTR
	MOVX A,@DPTR	;get the second ASCII number in reg A from memory location 8501H
	ANL A,#0FH	;make logical AND function with reg A and immediate data 0F0h
	SWAP A	;swap the contents of register A
	ORL A,B	;make logical OR function with register A and B
	INC DPTR	;increment DPTR =8502H
	MOVX @DPTR,A	;save the result(BCD Number.) in memory location 8502H
	LCALL 0003H	;end of asm file

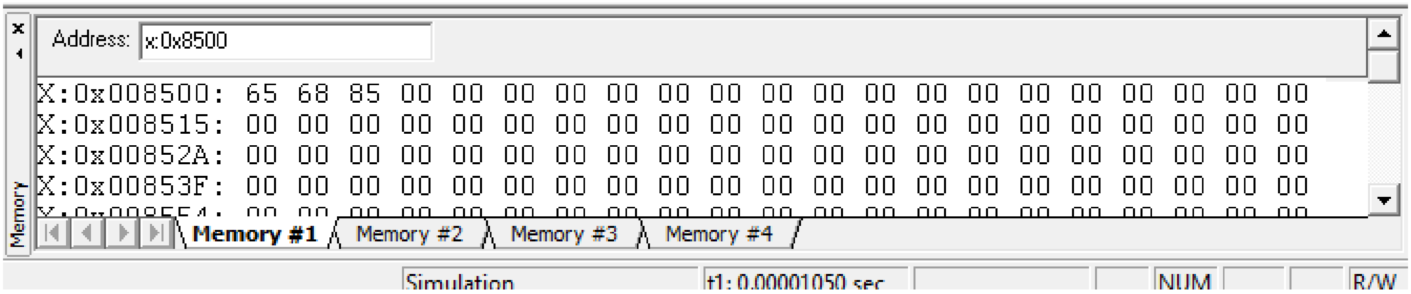
Before Execution



Address: x:0x8500

```
X:0x008500: 65 68 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution



Address: x:0x8500

```
X:0x008500: 65 68 85 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x008515: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00852A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X:0x00853F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

7	**Programs to generate delay, Programs using serial port and on-chip timer/counters.**
7a)	Write an ALP to toggle the content of port 0 continuously using timer delay in between.
7b)	Write an ALP to transmit characters to a PC HyperTerminal using the serial port and display on the serial window.

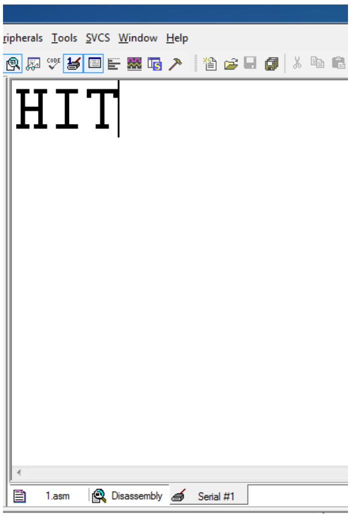
7a)	Write an ALP to toggle the content of port 0 continuously using timer delay in between.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV R0, #0AH	; R0=0AH
GO:	MOV A, #55H	; A=55H
	MOV P0, A	; Move the content of A in P0
	ACALL DELAY	; Call delay program
	MOV A, #0AAH	; A=AAh
	MOV P0, A	; Move the content of A in P0
	ACALL DELAY	; Call delay program
	DJNZ R0, GO	; R0=R0-1, if R0 is not equal to zero then jump to go
	LCALL 0003H	; End of main program
DELAY:	MOV TMOD, #01H	; Load TMOD
START:	MOV TL0, #00H	; Load TL0
	MOV TH0, #00H	;Load TH0
	SETB TR0	; TR0=1;
HERE:	JNB TF0, HERE	;if TF0 is not equal 1 then jump to here
	CLR TR0	; TR0=0
	CLR TF0	;TF0=0
	RET	;Return to main program

7b)	Write an ALP to transmit characters to a PC HyperTerminal using the serial port and display on the serial window.

Label	Opcode and Operands	Comments
	ORG 0000H LJMP 8000H ORG 8000H	
	MOV TMOD, #20H	; TMOD= 20H
	MOV TH1, #-3	;TH1=-3H
	MOV SCON, #50H	;SCON= 50H
	SETB TR1	;TR1=1;
	MOV A, #’H’	;Load letter ‘H’ in A
	ACALL TRANS	;Call transmit program
	MOV A, #’I’	;Load letter ‘I’ in A
	ACALL TRANS	;Call transmit program
	MOV A, #’T’	;Load letter ‘T’ in A
	ACALL TRANS	;Call transmit program
	LCALL 0003H	;Stop the program
TRANS:	MOV SBUF, A	; Load SBUF with letter stored in A
HERE:	JNB TI, HERE	; if TI is not equal 1 jump to here
	CLR TI	; Clear TI
	RET	; Return
	END	; End

After Execution



```
HIT
```

8	**Stepper motor interface.**
8a)	Write a C program to rotate stepper motor in clockwise/anticlockwise direction.

8a)	Write a C program to rotate stepper motor in clockwise/anticlockwise direction.

```
#include<reg51.h>

sbit sw=P1^6;

void delay(unsigned int );

void main()
{
sw=1;
while(1)
{
if(sw==0)
{
P2=0x66;
delay(22000);
P2=0x33;
delay(22000);
P2=0x99;
delay(22000);
P2=0xcc;
delay(22000);
}
else
{
P2=0xcc;
delay(22000);
P2=0x99;
delay(22000);
P2=0x33;
delay(22000);
P2=0x66;
delay(22000);
}
}
}

void delay(unsigned int itime)
{
unsigned int I, j;
for(i=0;i<=itime;i++);
for(j=0;j<=6000;j++);
}
```

9	**DC motor interface for direction and speed control using PWM.**
9a)	Write a C program to show the on off control of DC motor.

9a)	Write a C program to show the on off control of DC motor.

```
#include<reg51.h>

sbit ok=P2^4;

void turn_onoff(unsigned int);

void  main(void)
{
TCON=0;
TMOD=0x20;
while(1)
{
ok=0;
turn_onoff(50000);
ok=1;
turn_onoff(50000);
}
}

void turn_onoff(unsigned int)
{
unsigned int i;
for(i=0;i<itime;i++)
{
TL0=0x00;
TH0=0xff;
TR0=1;
while(!TF0);
{
TF0=0;
TR0=0;
}
}
```

10	**Alphanumerical LCD panel interface.**
10a)	Write a C program to send letters to LCD using delays.

10a)	Write a C program to send letters to LCD using delays.

```
#include<reg51.h>

sbit rs=P2^4;
sbit rw=P2^5;
sbit en=P2^6;

void main()
{
void lcdcmd(unsigned char value);
void lcddata (unsigned char value);
void delay(unsigned int itime);

lcdcmd(0x38);
delay(250);
lcdcmd(0x0c);
delay(250);
lcdcmd(0x01);
delay(250);
lcdcmd(0x80);
delay(250);
lcddata(‘H’);
delay(250);
lcddata(‘I’);
delay(250);
lcddata(‘T’);
delay(250);
lcddata(‘-’);
delay(250);
lcddata(‘E’);
delay(250);
lcddata(‘E’);
delay(250);
}
```

```
void lcdcmd(unsigned char value)
{
P0=value;
rs=0;
rw=0;
en=1;
delay(1);
en=0;
}


void lcddata(unsigned char value)
{
P0=value;
rs=1;
rw=0;
en=1;
delay(1);
en=0;
}


void delay(unsigned int itime)
{
unsigned int i. j;
for(i=0;i<itime;i++)
for(j=0;j<1275;j++);
}
```

11	**Generate different waveforms: Sine, Square, Triangular, Ramp using DAC interface.**
11a)	Write a C program to generate Square wave using DAC interface to 8051.
11b)	Write a C program to generate Ramp wave using DAC interface to 8051.
11c)	Write a C program to generate triangular wave using DAC interface to 8051.
11d)	Write a C program to generate Sine wave using DAC interface to 8051.

11a)	Write a C program to generate Square wave using DAC interface to 8051.

```
#include<reg51.h>
void delay(unsigned int );

void main(void)
{
while(1)
{
P1=0x00;
delay(250);
P1=0xff;
delay(250);
}
}


void delay(unsigned itime)
{
unsigned int i;
for(i=0;i<=itime;i++);
}
```

11b)	Write a C program to generate Ramp wave using DAC interface to 8051.

```
#include<reg51.h>
void delay(unsigned int );

void main(void)
{
unsigned int i;
while(1)
{
for(i=0;i<255;i++)
{
P1=i;
delay(2);
}
}
}


void delay(unsigned itime)
{
unsigned int i;
for(i=0;i<=itime;i++);
}
```

11c)	Write a C program to generate triangular wave using DAC interface to 8051.

```
#include<reg51.h>
void delay(unsigned int );

void main(void)
{
unsigned int  i,j;
while(1)
{
for(i=0;i<255;i++)
{
P1=i;
delay(2);
}
for(j=255;j>0;j--)
{
P1=j;
delay(2);
}
}
}


void delay(unsigned itime)
{
unsigned int i;
for(i=0;i<=itime;i++);
}
```

11d)	Write a C program to generate Sine wave using DAC interface to 8051.

```
#include<reg51.h>

sfr  dacdata=0x90;

void main(void)
{
unsigned char sine_value[12]={128, 192,238,255,238,192,128,64,17,0,17,64};
unsigned int x;
while(1)
{
for(x=0;x<12;x++)
dacdata=sine_value[x];
}
}
```

12	**Elevator interface.**
12a)	Write a C program to show control and operation of elevator using 8051.

12a)	Write a C program to show control and operation of elevator using 8051.

```
#include<reg51.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;
ReqFlr=~ReqFlr;
if(CurFlr==ReqFlr)
{
P0=FClr[CurFlr];
continue;
}
else if (CurFlr>ReqFlr)
{
i=Flr[CurFlr]-Flr[ReqFlr];
j=Flr[CurFlr];
for(;i>0;i--)
{
P0=0x0f0|j;
j--;
delay(50000);
}
}
else
{
i=Flr[ReqFlr]-Flr[CurFlr];
j=Flr[CurFlr];
for(;i>0;i--)
{
P0=0x0f0|j;
j++;
delay(50000);
}
}
CurFlr=ReqFlr;
P0=FClr[CurFlr];
}
}
void delay(unsigned int x)
{
for(;x>0;x--);
}
```

13	**ADC Interface**
13a)	Write a C program to interface ADC to measure temperature.

13a)	Write a C program to interface ADC to measure temperature.

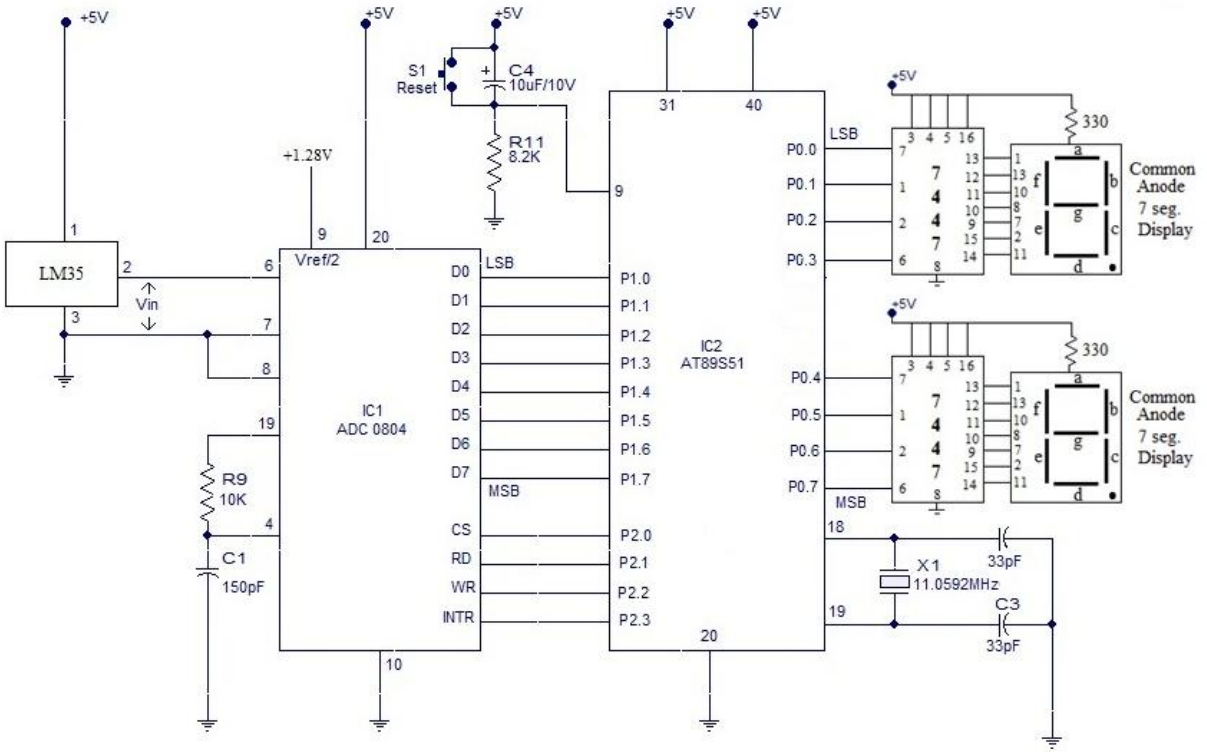
```
#include<reg51.h>
sbit cs=P2^0;
sbit rd=P2^1;
sbit wr=P2^2;
sbit intr=P2^3;

void delay( int itime);

main()
{
int  result;
P1=255;
P2=0;
P0=0;
while(1)
{
cs=0;
wr=0;
delay(50);
wr=1;
while(intr!=0);
cs=0;
rd=1;
delay(50);
rd=0;
result=P1;
P0=(((result/10)*16)+(result%10));
}
}

void delay(int itime)
{
int i,j;
for(i=0;i<itime<i++)
for(j=0;j<1275;j++);
}
```

Circuit Diagram



As per the datasheet LM35 gives output of 10mV per degree centigrade of temperature.

Example:

Room Temperature in Degrees	28 Decimal				1C Hex			
LM 35 Output (DC)	280mV=0.28V							
ADC 0804 Output	1C Hex							
	0	0	0	1	1	1	0	0
Conversion Formula	(28/10 X 16) + (28%20) Decimal				(1C%0A) X 10) + (1C%0A) Hex			
Conversion Formula Result	32+8= 40 Decimal				20+8= 28 Hex			
Output of Port P0 is given to 7447 display driver to display it on 7 segment display	2 (MSB)				8 (LSB)			
	0	0	1	0	1	0	0	0
7-segment display