Shri Ramdeobaba College of Engineering and Management, Nagpur
Department of Computer Science and Engineering
(Artificial Intelligence & Machine Learning)
Semester-V

Course Code: CAP303 Course: Microcontroller Design Lab

LIST OF EXPERIMENT
Session: 2022-23

Lab No.	Experiment
Lab 0	Study and familiarization of 8051 Microcontroller trainer kit and software tool.
Lab 1	Write an assembly language program i) to perform addition of two 16 bit number and ii) to perform subtraction of two 16 bit number and
Lab 2	Write an assembly language program i) To perform block move operation from one location to another location (Internal RAM) ii) To exchange two blocks of data bytes using 8051
Lab 3	Write an assembly language program i) To find smallest number from the given array ii) To find largest number from the given array
Lab 4	Write an assembly language program i) To arrange the number in the ascending order ii) To arrange the number in the descending order
Lab 5	Write an assembly language program i) To exchange the lower nibble of data present in external memory 6000H and 6001H. ii) To count the number of 1's and 0's of 8 bit data stored in location 6000H.
Lab 6	Write an assembly language program/Embedded C program i) To ON-OFF all the LED's of port P0 continuously with some delay in between. ii) To toggle all the LED's of port P0 continuously with some delay in between.
Lab 7	Write an assembly language program/Embedded C program to print "WELCOME CSE-AIML" on LCD of 8051 Microcontroller development board.
Lab 8	Write an assembly language program/Embedded C program to display decimal number 0 - 9 on seven-segment display of 8051 Microcontroller development board.
Lab 9	Write an Assembly Language Program to perform addition of two 32 bit numbers stored at memory location using ARM board FRDM-L25Z.
Lab 10	Write an Assembly Language Program to perform subtraction of two 32 bit numbers stored at memory location using ARM board FRDM-L25Z.
Lab 11	Mini Project based on 8051 Microcontroller.

Internal Distribution [25 Marks]

♣ Execution [Internal Exam]	: 10 marks
♣ Viva-Voce [Internal Exam]	: 05 marks
♣ Mini Project	: 05 marks
♣ Lab Journal	: 05 marks

Dr. A. J. Agrawal
HOD, CSE

Experiment-0

Aim:- Study and familiarization of 8051 Microcontroller trainer kit and software tool.

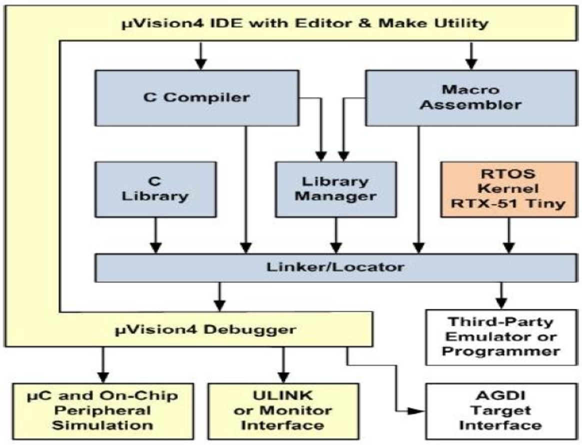
Software used:- Keil µVision5/Flash Magic

Hardware used:-NVIS 5001A trainer kit

Theory:-

The C51 development tool chains are designed for the professional software developer, but any level of programmer can use them to get the most out of the 8051 microcontroller architecture.

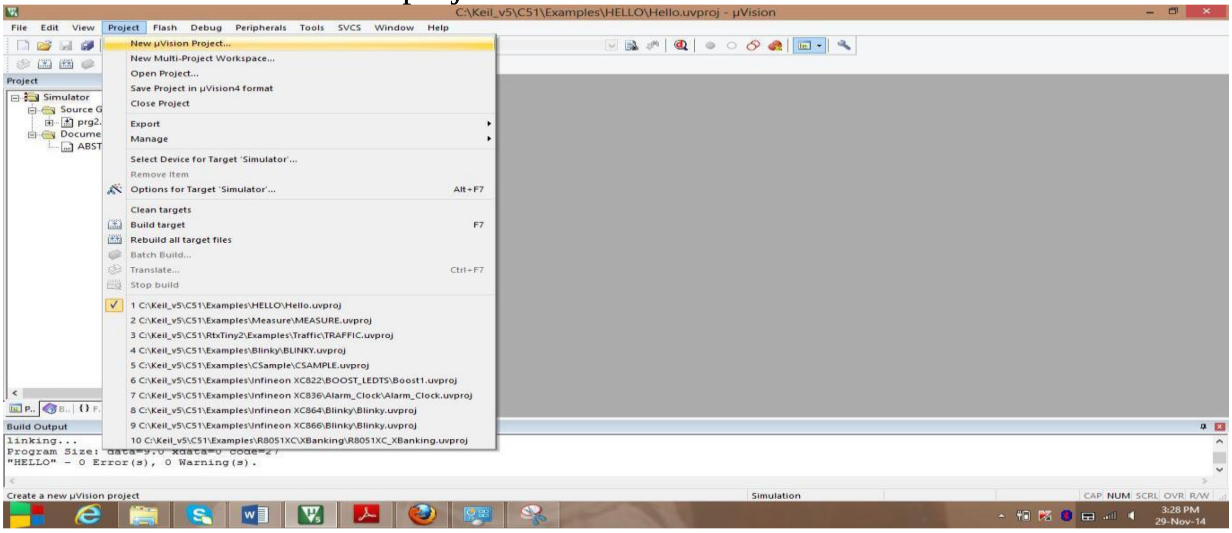
The µVision IDE is a window-based software development tool that combines project management and a rich-featured editor with interactive error correction, option setup, make facility, and on-line help. Use µVision to create source files and organize them into a project that defines your target application.


µVision4 IDE with Editor & Make Utility
C Compiler
Macro Assembler
C Library
Library Manager
RTOS Kernel RTX-51 Tiny
Linker/Locator
µVision4 Debugger
Third-Party Emulator or Programmer
µC and On-Chip Peripheral Simulation
ULINK or Monitor Interface
AGDI Target Interface

µVision Integrated Development Environment (IDE)

Procedure:- (How to use Keil uVision)

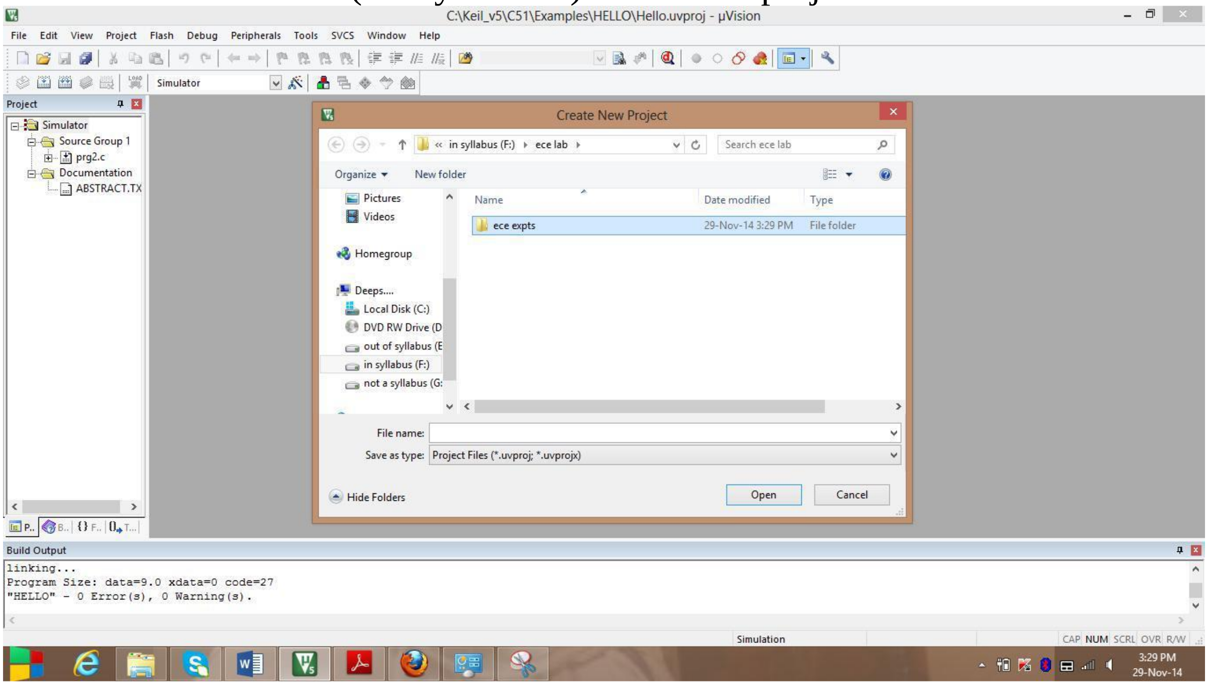
1. Create new u vision project


C:\Keil_v5\C51\Examples\HELLO\Hello.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
New µVision Project...
New Multi-Project Workspace...
Open Project...
Save Project in µVision4 format
Close Project
Export
Manage
Select Device for Target 'Simulator'...
Remove Item
Options for Target 'Simulator'... Alt+F7
Clean targets
Build target F7
Rebuild all target files
Batch Build...
Translate... Ctrl+F7
Stop build
1 C:\Keil_v5\C51\Examples\HELLO\Hello.uvproj
2 C:\Keil_v5\C51\Examples\Measure\MEASURE.uvproj
3 C:\Keil_v5\C51\RtxTiny2\Examples\Traffic\TRAFFIC.uvproj
4 C:\Keil_v5\C51\Examples\Blinky\BLINKY.uvproj
5 C:\Keil_v5\C51\Examples\CSample\CSAMPLE.uvproj
6 C:\Keil_v5\C51\Examples\Infineon XC822\BOOST_LEDTS\Boost1.uvproj
7 C:\Keil_v5\C51\Examples\Infineon XC836\Alarm_Clock\Alarm_Clock.uvproj
8 C:\Keil_v5\C51\Examples\Infineon XC864\Blinky\Blinky.uvproj
9 C:\Keil_v5\C51\Examples\Infineon XC866\Blinky\Blinky.uvproj
10 C:\Keil_v5\C51\Examples\R8051XC\XBanking\R8051XC_XBanking.uvproj
Project
Simulator
Source G
prg2.
Docume
ABST
Build Output
Simulation
CAP NUM SCRL OVR R/W
3:28 PM
29-Nov-14

```
linking...
Program Size: data=9.0 xdata=0 code=27
"HELLO" - 0 Error(s), 0 Warning(s).
```

Create a new µVision project

2. Select the folder (newly created) to save the project



C:\Keil_v5\C51\Examples\HELLO\Hello.uvproj - µVision

Create New Project

« in syllabus (F:) › ece lab ›

Name	Date modified	Type
ece expts	29-Nov-14 3:29 PM	File folder

File name:

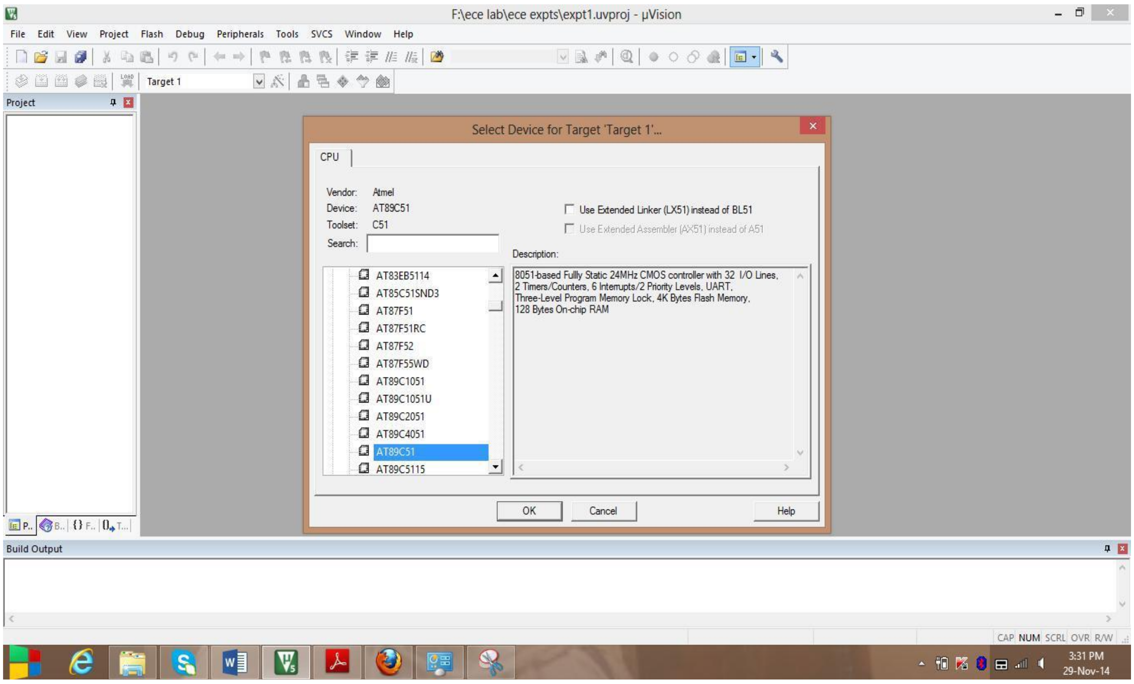
Save as type: Project Files (*.uvproj; *.uvprojx)

Build Output

```
linking...
Program Size: data=9.0 xdata=0 code=27
"HELLO" - 0 Error(s), 0 Warning(s).
```

3. Save the project

4. Select the vendor “Atmel” and device “AT89C51”



F:\ece lab\ece expts\expt1.uvproj - µVision

Select Device for Target 'Target 1'...

Vendor: Atmel

Device: AT89C51

Toolset: C51

Search:

□ Use Extended Linker (LX51) instead of BL51

□ Use Extended Assembler (AX51) instead of A51

Description:

8051-based Fully 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

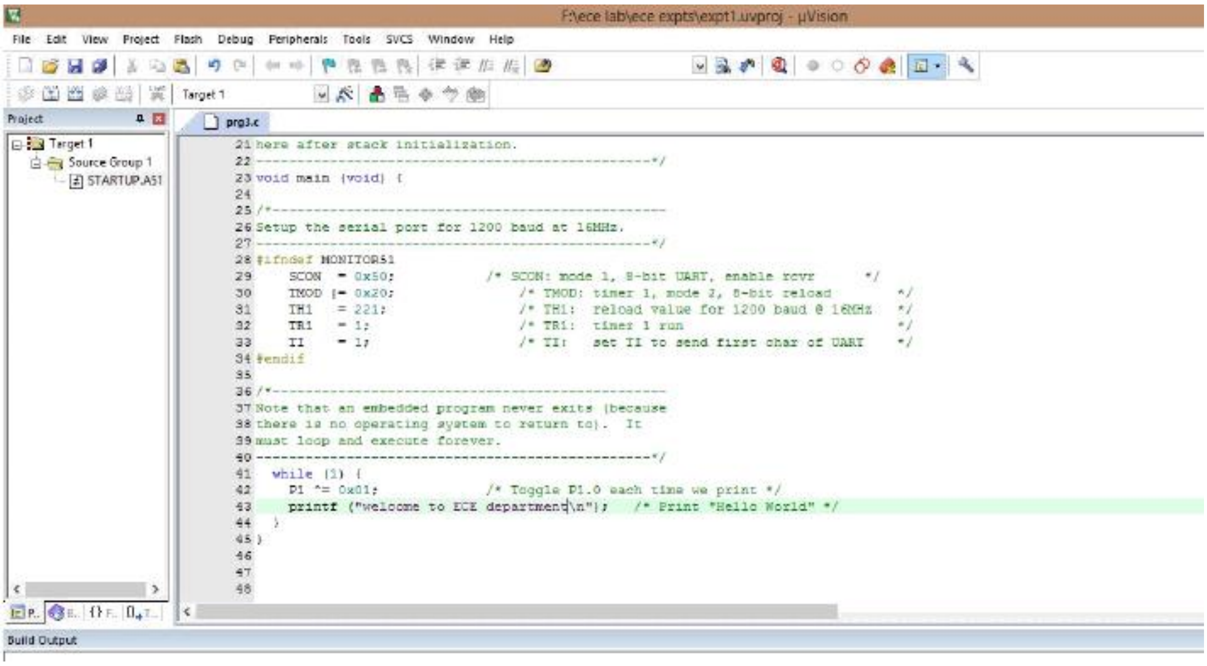
5. Addition of STARTUP.A51 to project folder- click Yes.

6. Click on file->New.

7. Text file generated to write actual program.

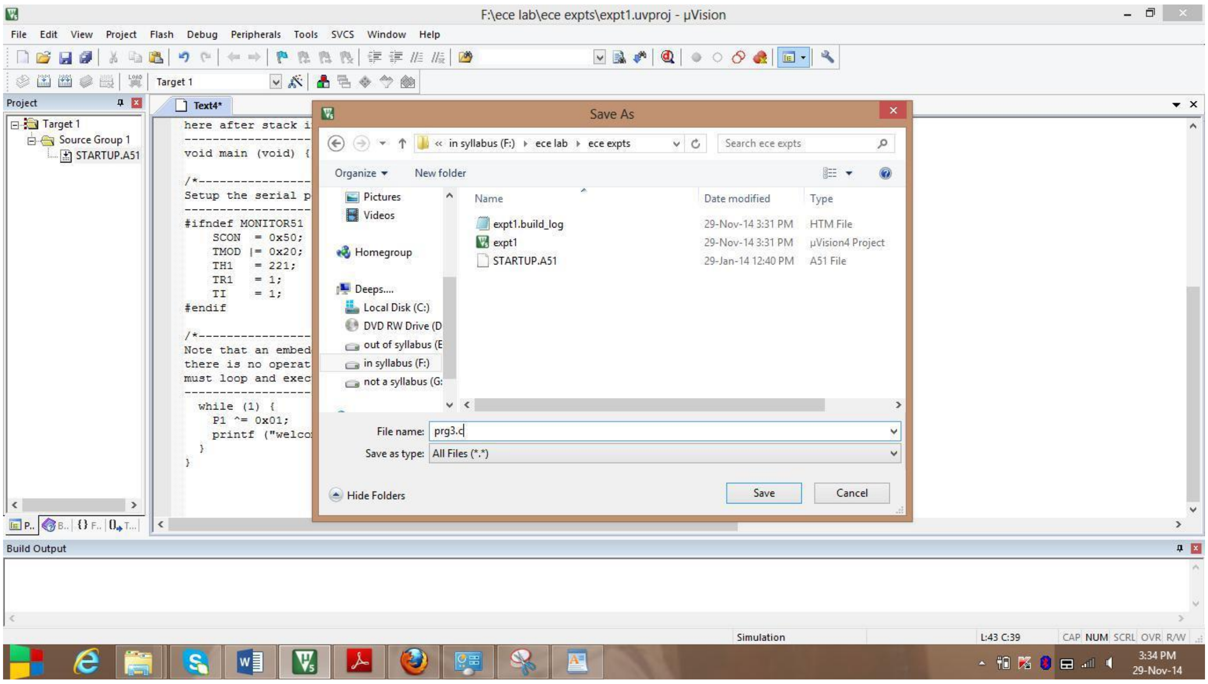
8. Write a program and save it by the extension .c

Example:- The program to print “welcome to ECE department” is written


F:\ece lab\ece expts\expt1.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Source Group 1
STARTUP.A51
prg3.c
Build Output

```
here after stack initialization.
------------------------------------------------*/
void main (void) {

/*------------------------------------------------
Setup the serial port for 1200 baud at 16MHz.
------------------------------------------------*/
#ifndef MONITOR51
    SCON  = 0x50;           /* SCON: mode 1, 8-bit UART, enable rcvr      */
    TMOD |= 0x20;               /* TMOD: timer 1, mode 2, 8-bit reload        */
    TH1   = 221;                /* TH1:  reload value for 1200 baud @ 16MHz   */
    TR1   = 1;                  /* TR1:  timer 1 run                          */
    TI    = 1;                  /* TI:   set TI to send first char of UART    */
#endif

/*------------------------------------------------
Note that an embedded program never exits (because
there is no operating system to return to).  It
must loop and execute forever.
------------------------------------------------*/
  while (1) {
    P1 ^= 0x01;             /* Toggle P1.0 each time we print */
    printf ("welcome to ECE department\n");   /* Print "Hello World" */
  }
}
```

9. The program is saved as prg3.c



Name	Date modified	Type
expt1.build_log	29-Nov-14 3:31 PM	HTM File
expt1	29-Nov-14 3:31 PM	µVision4 Project
STARTUP.A51	29-Jan-14 12:40 PM	A51 File

File name: prg3.c

Save as type: All Files (*.*)

```
here after stack i
-----------------
void main (void) {

/*---------------
Setup the serial p
-----------------
#ifndef MONITOR51
    SCON  = 0x50;
    TMOD |= 0x20;
    TH1   = 221;
    TR1   = 1;
    TI    = 1;
#endif

/*---------------
Note that an embed
there is no operat
must loop and exec
-----------------
  while (1) {
    P1 ^= 0x01;
    printf ("welco
  }
}
```

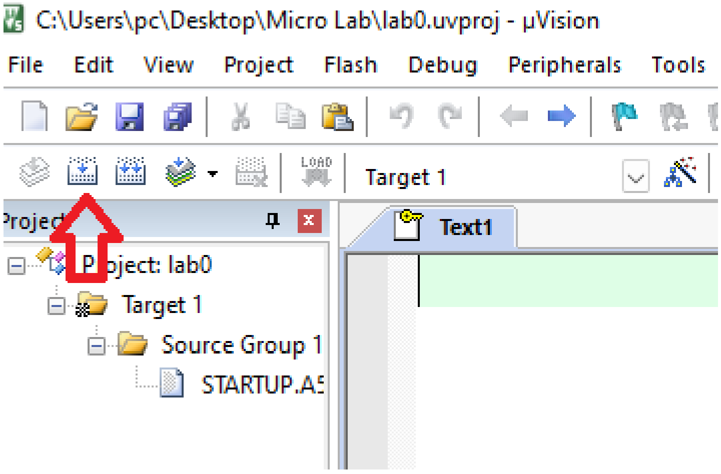
10. Prg3.c is to be added to Source Group1

Right click on source group1 ->Add existing files to group “source group1”.

Select prg3.c file-> click on add button.

Now prg3.c is added to Source Group 1.

11.Build the target. Click on this icon.


C:\Users\pc\Desktop\Micro Lab\lab0.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools
Target 1
Project
Text1
Project: lab0
Target 1
Source Group 1
STARTUP.A5

12. Debugging the target
Click on debug->start/stop debug session.
New window evaluation mode appeared. Press ok.
Again click on debug-> run .

To check output:-

1. On UART:-
   Click on view-> serial window->UART#1.
2. On port:-
   Click on peripherals ->I/O port-> port1.

Stop debugging process after checking the output.

Experiment-1

Aim:- Write an assembly language program to perform addition and subtraction of 16 bit number using Keil µVision tool.

Apparatus/Tools Required:-

Sr. No.	Apparatus/Tools	Description	Quantity
1	PC/Laptop	---	1
2	Keil µVision4	IDE Tool	1
3	NVIS5001 A	8051 Trainer Kit	1

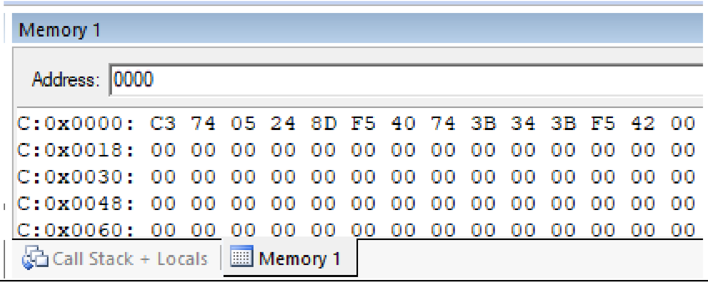
Program:- (16 Bit Addition)

Memory address	Instruction	Opcode	Label	Description
-------	ORG 0000H	--		Start your code from 0000H
0000H	CLR C	C3		Make CY=0
0001H, 0002H	MOV A,#05H	74, 05		Load A with low byte E7H
0003H, 0004H	ADD A,#8DH	24, 8D		Add low byte with 8DH and A=74H, CY=1
0005H, 0006H	MOV 40H, A	F5, 40		Save the low byte sum at 40H
0007H, 0008H	MOV A, #3BH	74, 3B		Load A with high byte 3BH
0009H, 000AH	ADDC A,#3BH	34, 3B		Add with carry (3B+3B+1)
000BH	MOV 42H,A	F5, 42		Save the high byte sum at 42H
	END			End your assembly program.

Flowchart:- (Student will draw flowchart as per code on blank page)

Input:-

First 16 Bit Number	3B05 H
Second 16 Bit Number	3B8D H



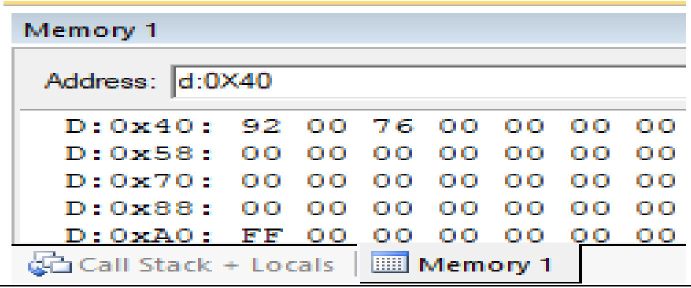
Memory 1

Address: 0000

```
C:0x0000: C3 74 05 24 8D F5 40 74 3B 34 3B F5 42 00
C:0x0018: 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Output:-

Addition of two 16 Bit number	7692 H



Memory 1

Address: d:0X40

```
D:0x40: 92 00 76 00 00 00 00
D:0x58: 00 00 00 00 00 00 00
D:0x70: 00 00 00 00 00 00 00
D:0x88: 00 00 00 00 00 00 00
D:0xA0: FF 00 00 00 00 00 00
```

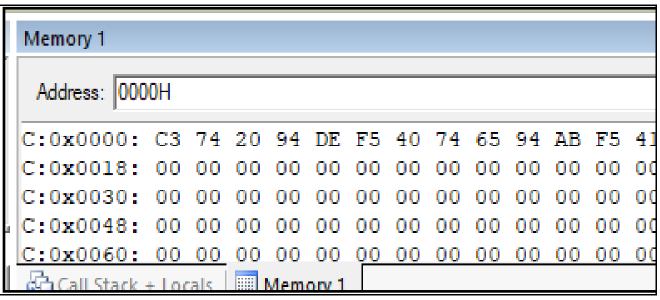
Program:- (16 Bit Subtraction)

Memory address	Instruction	Opcode	Label	Description
-------	ORG 0000H			
0000H	CLR C	C3		Make CY=0
0001H, 0002H	MOV A,#20H	74,20		Lower Byte of Operand 1 In A
0003H, 0004H	SUBB A,#DEH	94,DE		Subtract Lower Byte of Operand 2 With A
0005H, 0006H	MOV 40H,A	F5,40		Stores LSB of Result In 40H
0007H, 0008H	MOV A,#65H	74,65		Higher Byte of Operand 2 In A
0009H, 000AH	SUBB A,#ABH	94,AB		Subtract With Higher Byte of Operand 1
000BH	MOV 41H,A	F5,41		Stores MSB of Result In 41H
-------	END			

Flowchart:- (Student will draw flowchart on blank page)

Input:-

First 16 Bit Number	AB20 H
Second 16 Bit Number	65DE H

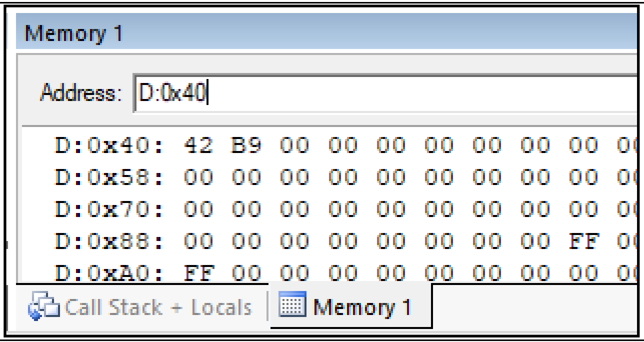


Address: 0000H

```
C:0x0000: C3 74 20 94 DE F5 40 74 65 94 AB F5 41
C:0x0018: 00 00 00 00 00 00 00 00 00 00 00 00 0
C:0x0030: 00 00 00 00 00 00 00 00 00 00 00 00 0
C:0x0048: 00 00 00 00 00 00 00 00 00 00 00 00 0
C:0x0060: 00 00 00 00 00 00 00 00 00 00 00 00 0
```

Output:-

Subtraction of two 16 Bit number is	B942 H



Address: D:0x40

```
D:0x40: 42 B9 00 00 00 00 00 00 00 0
D:0x58: 00 00 00 00 00 00 00 00 00 0
D:0x70: 00 00 00 00 00 00 00 00 00 0
D:0x88: 00 00 00 00 00 00 00 00 FF 0
D:0xA0: FF 00 00 00 00 00 00 00 00 0
```

Conclusion:-

Experiment-2

Aim:- Write an assembly language program

i) To perform block move operation from one location to another location (Internal RAM)

Apparatus/Tools Required:-

Sr. No.	Apparatus/Tools	Description	Quantity
1	PC/Laptop	---	1
2	Keil µVision4	IDE Tool	1
3	NVIS5001 A	8051 Trainer Kit	1

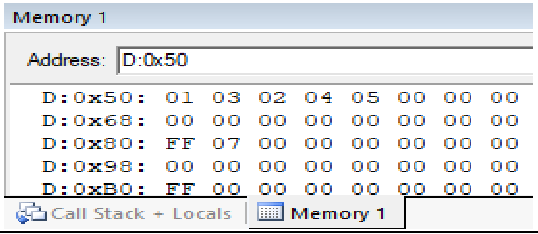
Program:-

Memory address	Instruction	Opcode	Description
0000, 0001	MOV R0,#50H	78, 50	Initialize the source memory pointer
0002, 0003	MOV R1,#68H	79, 68	Initialize the destination memory pointer
0004, 0005	MOV R2,#05H	7A, 05	Initialize Iteration counter
0006	BACK : MOV A,@R0	E6,	Get the data from source memory pointer
0007	MOV @R1,A	F7	Store the data into destination memory pointer
0008	INC R0	08	Increment the source memory pointer
0009	INC R1	09	Increment the destination memory pointer
000A, 000B	DJNZ R2, BACK	DA, FA	Decrement iteration count and if it // is not zero, go to relative Address

Flowchart:- (Student will draw flowchart on blank page)

Input:- (Before Execution)

50H= 01
51H= 03
52H= 02
53H= 04
54H= 05



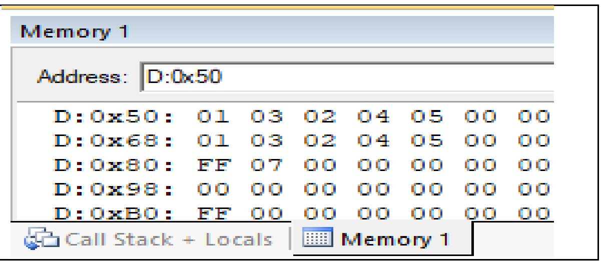
Memory 1

Address: D:0x50

```
D:0x50: 01 03 02 04 05 00 00 00
D:0x68: 00 00 00 00 00 00 00 00
D:0x80: FF 07 00 00 00 00 00 00
D:0x98: 00 00 00 00 00 00 00 00
D:0xB0: FF 00 00 00 00 00 00 00
```

Output:- (After Execution)

68H= 01
69H= 03
70H= 02
71H= 04
72H= 05



Memory 1

Address: D:0x50

```
D:0x50: 01 03 02 04 05 00 00
D:0x68: 01 03 02 04 05 00 00
D:0x80: FF 07 00 00 00 00 00
D:0x98: 00 00 00 00 00 00 00
D:0xB0: FF 00 00 00 00 00 00
```

Conclusion:-

Experiment-2

Aim:- Write an assembly language program

ii) To exchange two blocks of data bytes using 8051

Apparatus/Tools Required:-

Sr. No.	Apparatus/Tools	Description	Quantity
1	PC/Laptop	---	1
2	Keil µVision4	IDE Tool	1
3	NVIS5001 A	8051 Trainer Kit	1

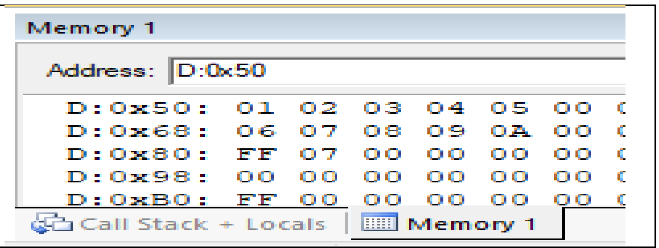
Program:-

Memory address	Instruction	Opcode	Description
0000,0001	MOV R0,#50H	78, 50	Initialize the source memory pointer
0002,0003	MOV R1,#68H	79, 68	Initialize the destination memory pointer
0004,0005	MOV R2,#05H	7A, 05	Initialize Iteration counter
0006	BACK: MOV A,@R0	E6	Get the data from source memory pointer
0007	XCH A,@R1	C7	Exchange data
0008	MOV @R0,A	F6	Store the data into source memory pointer
0009	INC R0	08	Increment the source memory pointer
000A	INC R1	09	Increment the destination memory pointer
000B, 000C	DJNZ R2, BACK	DA, F9	Decrement iteration count and if it is not zero
	END		

Flowchart:- (Student will draw flowchart on blank page)

Input:-(Before Execution)

50H=01	68H=06
51H=02	69H=07
52H=03	70H=08
53H=04	71H=09
54H=05	72H=0A

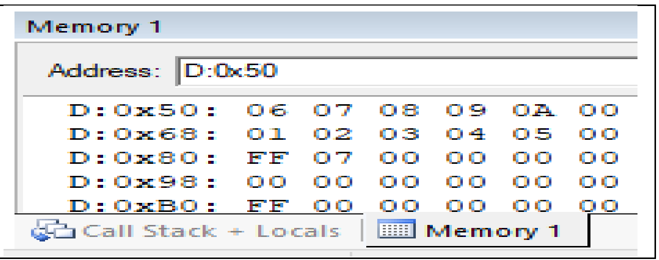

Memory 1

Address: D:0x50

```
D:0x50:  01  02  03  04  05  00
D:0x68:  06  07  08  09  0A  00
D:0x80:  FF  07  00  00  00  00
D:0x98:  00  00  00  00  00  00
D:0xB0:  FF  00  00  00  00  00
```

Output:- (After Execution)

50H=06	68H=01
51H=07	69H=02
52H=08	70H=03
53H=09	71H=04
54H=0A	72H=05


Memory 1

Address: D:0x50

```
D:0x50:  06  07  08  09  0A  00
D:0x68:  01  02  03  04  05  00
D:0x80:  FF  07  00  00  00  00
D:0x98:  00  00  00  00  00  00
D:0xB0:  FF  00  00  00  00  00
```

Conclusion:-

Experiment-3

Aim:- Write an assembly language program

a) To find largest number from the given array

Apparatus/Tools Required:-

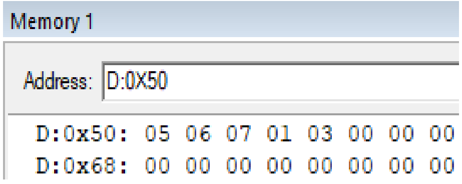
Sr. No.	Apparatus/Tools	Description	Quantity
1	PC/Laptop	---	1
2	Keil µVision4	IDE Tool	1
3	NVIS5001 A	8051 Trainer Kit	1

Program:-

Memory address	Instruction	Opcode	Description
0000	MOV R0,#50H	78,50	Initialize the source memory pointer
0002	MOV R2,#05H	7A,05	Initialize Iteration counter
0004	MOV B, @R0	86,F0	Use B Register to store largest value and initialize it to the first value
0006	BACK: MOV A,@R0	E6	Get the data from source memory pointer and Load into accumulator
0007	CJNE A,B,LOOP	B5	Compare the data if not equal, go to relative address(LOOP)
000A	LOOP: JC LOOP1	F0,00	If carry generates, go to relative address LOOP1
000C	MOV B,A	40,02	Store larger value into B-register
000E	LOOP1:INC R0	F5	Increment the source memory pointer
000F	DJNZ R2,BACK	F0	Decrement iteration count and if it is not zero,
0011	MOV 60H,B	08,DA	Store the largest value into memory location 60H
	END		

Flowchart:- (Student will draw flowchart on blank page)

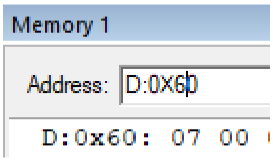
Input:-



Address: D:0X50

```
D:0x50: 05 06 07 01 03 00 00 00
D:0x68: 00 00 00 00 00 00 00 00
```

Output:-



Address: D:0X60

```
D:0x60: 07 00
```

Result:-

Conclusion:-

Experiment-3

Aim:- Write an assembly language program

a) To find smallest number from the given array

Apparatus/Tools Required:-

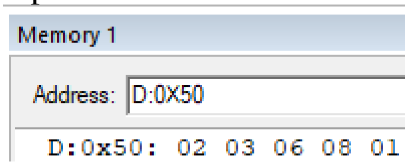
Sr. No.	Apparatus/Tools	Description	Quantity
1	PC/Laptop	---	1
2	Keil µVision4	IDE Tool	1
3	NVIS5001 A	8051 Trainer Kit	1

Program:-

Memory address	Instruction	Opcode	Description
0000	MOV R0,#50H	78,50	Initialize the source memory pointer
0002	MOV R2,#05H	7A,05	Initialize Iteration counter
0004	MOV B, @R0	86,F0	Mov 50H data to Register B
0006	BACK:MOV A,@R0	E6,B5	Mov 50H data to Register A
0007	CJNE A,B,LOOP	F0,00,50	Compare A and B data
000A	LOOP: JNC LOOP1	07,F5	If carry not generates, go to relative address LOOP1
000C	MOV B,A	F0,08	Mov data from A to B
000E	INC R0	DA	Increment R0
000F	DJNZ R2,BACK	F5,80	Decrement counter till zero
0011	SJMP NEXT	03, 08	Short jump on NEXT
0013	LOOP1: INC R0	DA	Inc source memory address
0014	DJNZ R2,BACK	F0, 85	Decrement R2
0016	NEXT: MOV 60H,B	F0, 60	Mov data on B to 60H memory
	END		

Flowchart:- (Student will draw flowchart on blank page)

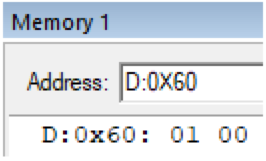
Input:-



Address: D:0X50

```
D:0x50: 02 03 06 08 01
```

Output:-



Address: D:0X60

```
D:0x60: 01 00
```

Result:-

Conclusion:-

Experiment-4

Aim:- Write an assembly language program

i) To arrange the numbers in ascending order.

Apparatus/Tools Required:-

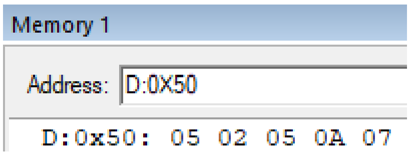
Sr. No.	Apparatus/Tools	Description	Quantity
1	PC/Laptop	---	1
2	Keil µVision4	IDE Tool	1
3	NVIS5001 A	8051 Trainer Kit	1

Program:-

Memory address	Instruction	Opcode	Description
	ORG 0000H		
0000	MOV R2, #05H	7A, 05	Mov 05 in register R2
0002	DEC R2	1A	Decrement R2
0003	BACK1: MOV R0, #50H	78, 50	Load R0 with 50H
0005	MOV R1, #51H	79, 51	Load R1 with 51H
0007	MOV A, R2	EA	Mov R2 data in A
0008	MOV R3, A	FB	Mov A data in R3
0009	BACK: MOV A,@R0	E6	Mov data on 50H in A
000A	MOV B,@R1	87, F0	Mov data on 51H in B
000C	CJNE A, B, LOOP	B5, F0, 00	Compare A & B
000F	LOOP: JC LOOP1	40, 03	Jump if carry is set
0011	MOV @R0,B	A6, F0	Mov B data on 50H
0013	MOV @R1, A	F7	Mov A data on 51H
0014	LOOP1: INC R0	08	Increment R0
0015	INC R1	09	Increment R0
0016	DJNZ R3, BACK	DB, F1	Decrement R3 if not zero jump Back
0018	DJNZ R2, BACK1	DA, E9	Decrement R2 if not zero jump Back1
	END		

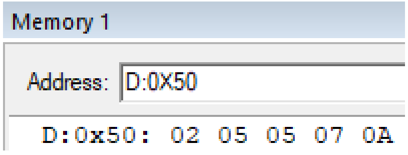
Flowchart:- (Student will draw flowchart on blank page)

Input:-



```
Memory 1
Address: D:0X50
D:0x50: 05 02 05 0A 07
```

Output:-



```
Memory 1
Address: D:0X50
D:0x50: 02 05 05 07 0A
```

Result:-

Conclusion:-

Experiment-4

Aim:- Write an assembly language program

ii) To arrange the numbers in descending order.

Apparatus/Tools Required:-

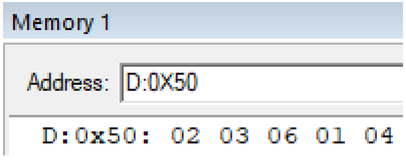
Sr. No.	Apparatus/Tools	Description	Quantity
1	PC/Laptop	---	1
2	Keil µVision4	IDE Tool	1
3	NVIS5001 A	8051 Trainer Kit	1

Program:-

Memory address	Instruction	Opcode	Description
	ORG 0000H		
0000	MOV R2, #05H	7A, 05	Mov 05 in register R2
0002	DEC	1A	Decrement R2
0003	BACK1: MOV R0, #50H	78, 50	Load R0 with 50H
0005	MOV R1, #51H	79, 51	Load R1 with 51H
0007	MOV A, R2	EA	Mov R2 data in A
0008	MOV R3, A	FB	Mov A data in R3
0009	BACK: MOV A,@R0	E6	Mov data on 50H in A
000A	MOV B,@R1	87, F0	Mov data on 51H in B
000C	CJNE A, B, LOOP	B5, F0, 00	Compare A & B
000F	LOOP: JNC LOOP1	50, 03	Jump if not carry
0011	MOV @R0,B	A6, F0	Mov B data on 50H
0013	MOV @R1, A	F7	Mov A data on 51H
0014	LOOP1: INC R0	08	Increment R0
0015	INC R1	09	Increment R0
0016	DJNZ R3,	DB, F1	Decrement R3 if not zero jump Back
0018	DJNZ R2, BACK1	DA, E9	Decrement R2 if not zero jump Back1
	END		

Flowchart:- (Student will draw flowchart on blank page)

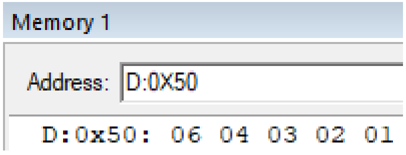
Input:-



Address: D:0X50

```
D:0x50: 02 03 06 01 04
```

Output:-



Address: D:0X50

```
D:0x50: 06 04 03 02 01
```

Result:-

Conclusion:-

Experiment-5

Aim:- Write an assembly language program

i) To exchange the lower nibble of data present in external memory 6000H and 6001H.

Apparatus/Tools Required:-

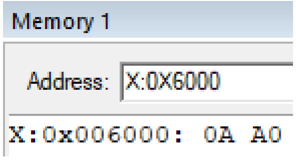
Sr. No.	Apparatus/Tools	Description	Quantity
1	PC/Laptop	---	1
2	Keil µVision4	IDE Tool	1
3	NVIS5001 A	8051 Trainer Kit	1

Program:-

Memory address	Instruction	Opcode	Description
	ORG 0000H		Set program counter 0000H
0000	MOV DPTR, #6000H	90, 60, 00	; Copy address 6000H to DPTR
0003	MOVX A, @DPTR	E0	; Copy contents of 6000H to A
0004	MOV R0,#45H	78, 45	; Load pointer, R0=45H
0006	MOV @R0,A	F6	; Copy contents of A to internal data RAM pointed by R0
0007	INC DPL	05, 82	; Increment pointer
0009	MOVX A,@DPTR	E0	; Copy contents of 6001H to A
000A	XCHD A,@R0	D6	; Exchange lower nibble of A with RAM pointed by R0
000B	MOVX @DPTR,A	F0	; Copy contents of A to 6001H
000C	DEC DPL	15, 82	; Decrement pointer
000E	MOV A,@R0	E6	Copy contents of internal data RAM pointed by R0 to A
000F	MOVX @DPTR,A	F0	; Copy contents of A to data RAM pointed by DPTR
	END		

Flowchart:- (Student will draw flowchart on blank page)

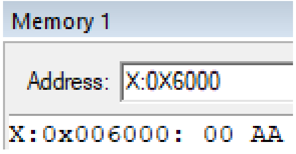
Input:-



Address: X:0X6000

```
X:0x006000: 0A A0
```

Output:-



Address: X:0X6000

```
X:0x006000: 00 AA
```

Result:-

Conclusion:-

Experiment-5

Aim: - Write an assembly language program

ii)To count the number of 1’s and 0’s of 8-bit data stored in location 6000H.

Apparatus/Tools Required: -

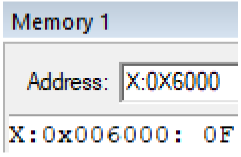
Sr. No.	Apparatus/Tools	Description	Quantity
1	PC/Laptop	---	1
2	Keil µVision4	IDE Tool	1
3	NVIS5001 A	8051 Trainer Kit	1

Program: -

Memory address	Instruction	Opcode	Description
	ORG 0000H		Set program counter 0000H
0000	MOV DPTR, #6000H	90, 60, 00	Copy address 6000H to DPTR
0003	MOVX A, @DPTR	E0	; Copy number to A
0004	MOV R0,#08	78, 08	; Copy 08 in R0
0006	MOV R2,#00	7A, 00	; Copy 00 in R2
0008	MOV R3,#00	7B, 00	; Copy 00 in R3
000A	CLR C	C3	; Clear carry flag
000B	BACK: RLC A	33	; Rotate contents of A through carry fl ag
000C	JC NEXT	40, 03	; If CF = 1, branch to next
000E	INC R2	0A	; If CF = 0, increment R2
000F	AJMP NEXT2	01, 12	
0011	NEXT: INC R3	0B	; If CF = 1, increment R3
0012	NEXT2: DJNZ R0,BACK	D8, F7	; Repeat until R0 is zero
	END		

Flowchart:- (Student will draw flowchart on blank page)

Input:-

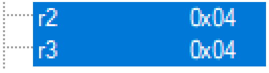


Memory 1

Address: X:0X6000

```
X:0x006000: 0F
```

Output:-



```
r2    0x04
r3    0x04
```

Result:-

Conclusion:-

Experiment-6

Aim: - Write an assembly language program

i) To ON-OFF all the LED’s of port P0 continuously with some delay in between.

Apparatus/Tools Required: -

Sr. No.	Apparatus/Tools	Description	Quantity
1	PC/Laptop	---	1
2	Keil µVision4	IDE Tool	1
3	Ultra 8051 Board	8051 Trainer Kit	1

Program: -

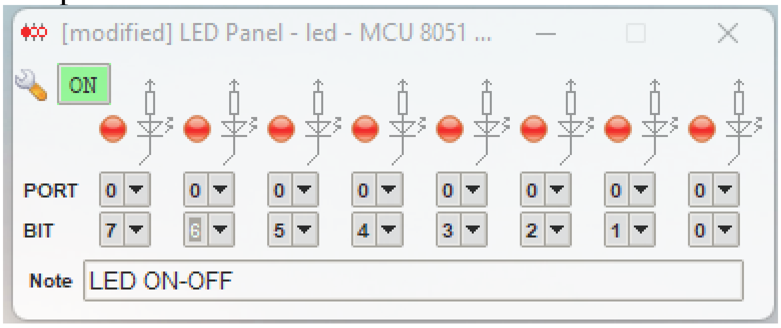
Memory address	Instruction	Opcode	Description
	ORG 0000H		
0000	LOOP: MOV A, #00H	74, 00	Load Accumulator with 00
0002	MOV P0, A	F5, 80	Mov accumulator data on port P0
0004	ACALL DELAY	11, 0E	Call Delay
0006	MOV A, #0FFH	74, FF	Load Accumulator with FF
0008	MOV P0, A	F5, 80	Mov accumulator data on port P0
000A	ACALL DELAY	11, 0E	Call Delay
000C	SJMP LOOP	80, F2	Short jump on LOOP
000E	DELAY: MOV R7,#50	7F, 32	Mov R7 with 50
0010	H1: MOV R2,#255H	7A, 55	Mov 255 to R2
0012	H2: MOV R3,#255H	7B, 55	Mov 255 to R3
0014	L:DJNZ R3,L	DB, FE	Dec R3
0016	DJNZ R2,H2	DA, FA	Dec R2
0018	DJNZ R7,H1	DF,F6	Dec R7
001A	RET	22	Return
	END		End of Program

Flowchart: - (Student will draw flowchart on blank page)

Input: -

A	FFH
A	00H

Output: -



[modified] LED Panel - led - MCU 8051 ...

PORT	0	0	0	0	0	0	0	0
BIT	7	6	5	4	3	2	1	0

Note LED ON-OFF

Result: -

Conclusion: -

Experiment-6

Aim: - Write an assembly language program

ii) To toggle all the LEDs of port P0 continuously with some delay in between.

Apparatus/Tools Required: -

Sr. No.	Apparatus/Tools	Description	Quantity
1	PC/Laptop	---	1
2	Keil µVision4	IDE Tool	1
3	Ultra 8051 Board	8051 Trainer Kit	1

Program: -

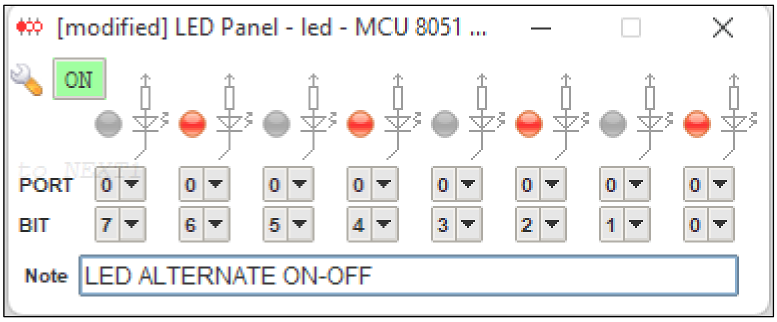
Memory address	Instruction	Opcode	Description
	ORG 0000H		
0000	LOOP: MOV A, #55H	74, 55	Load Accumulator with 55
0002	MOV P0, A	F5, 80	Mov accumulator data on port P0
0004	ACALL DELAY	11, 0E	Call Delay
0006	MOV A, #0AAH	74, AA	Load Accumulator with AA
0008	MOV P0, A	F5, 80	Mov accumulator data on port P0
000A	ACALL DELAY	11, 0E	Call Delay
000C	SJMP LOOP	80, F2	Short jump on LOOP
000E	DELAY: MOV R7,#50	7F, 32	Mov R7 with 50
0010	H1: MOV R2,#255H	7A, 55	Mov 255 to R2
0012	H2: MOV R3,#255H	7B, 55	Mov 255 to R3
0014	L:DJNZ R3,L	DB, FE	Dec R3
0016	DJNZ R2,H2	DA, FA	Dec R2
0018	DJNZ R7,H1	DF,F6	Dec R7

Flowchart: - (Student will draw flowchart on blank page)

Input: -

A	AA
A	55

Output: -



[modified] LED Panel - led - MCU 8051 ...

PORT	0	0	0	0	0	0	0	0
BIT	7	6	5	4	3	2	1	0

Note LED ALTERNATE ON-OFF

Result: -

Conclusion: -

Experiment-7

Aim:- Write an assembly language program/Embedded C program to print “WELCOME CSE-AIML” on LCD of 8051 Microcontroller development board.

Apparatus/Tools Required: -

Sr. No.	Apparatus/Tools	Description	Quantity
1	PC/Laptop	---	1
2	Keil µVision4	IDE Tool	1
3	Ultra 8051 Board	8051 Trainer Kit	1

Program:-

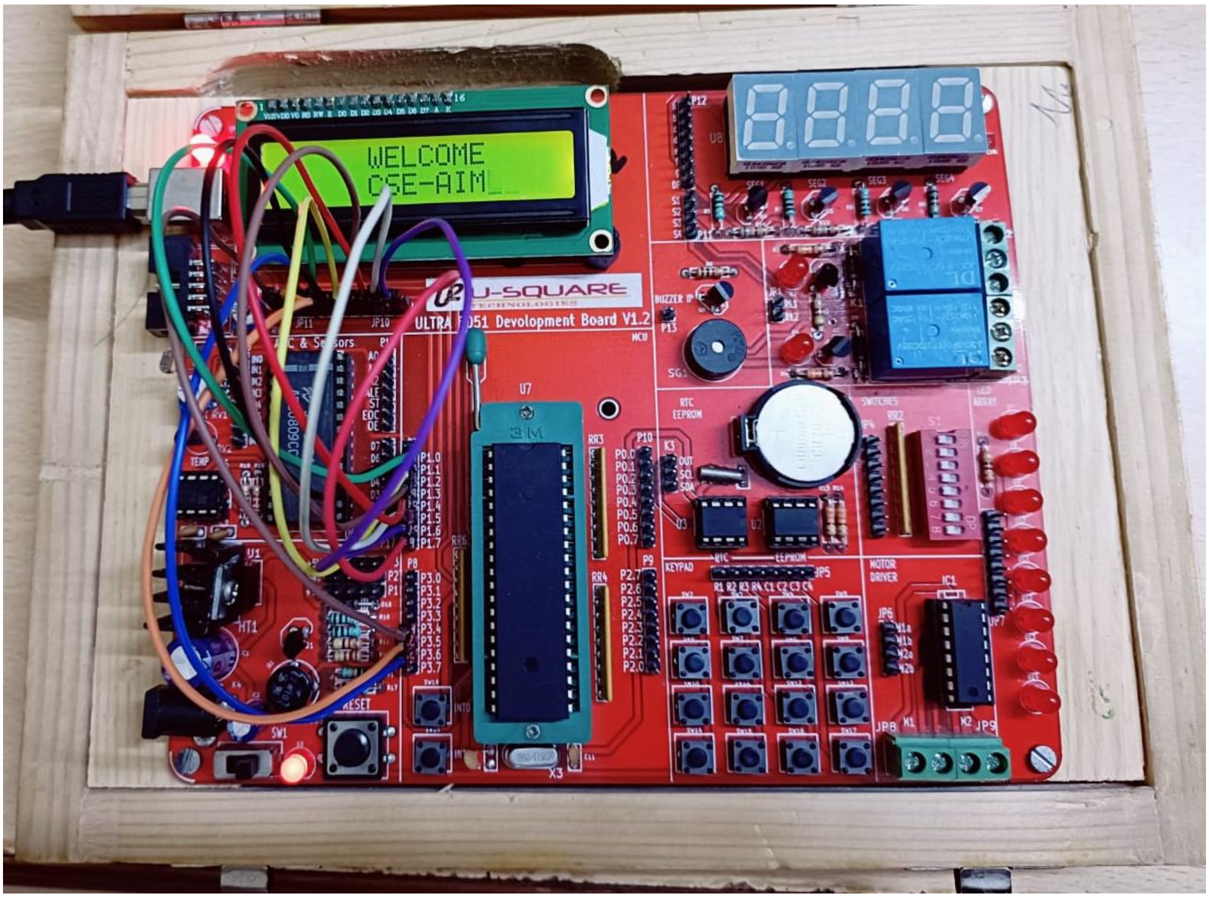
```
        ORG   0000H
        MOV   SP, #70H
        MOV   PSW, #00H

LCD_IN:     MOV   A, #38H    ; init. LCD 2 lines, 5x7 matrix
        LCALL       COMNWRT            ;call command subroutine
        LCALL       DELAY              ;give LCD some time
        MOV   A, #0EH            ;dispplay on, cursor on
        LCALL       COMNWRT            ;call command subroutine
        LCALL       DELAY              ;give LCD some time
        MOV     A, #01           ;clear LCD
        LCALL       COMNWRT            ;call command subroutine
        LCALL       DELAY              ;give LCD some time
        MOV     A, #06H          ;shift cursor right
        LCALL       COMNWRT            ;call command subroutine
        LCALL       DELAY              ;give LCD some time
        MOV     A, #85H          ;cursor at line 1 postion 1
        LCALL       COMNWRT            ;call command subroutine
        LCALL       DELAY              ;give LCD some time
        MOV     A, #'W'          ;display letter W
        LCALL       DATAWRT            ;call display subroutine
        LCALL       DELAY              ;give LCD some time
        MOV     A, #'E'          ;display letter E
        LCALL       DATAWRT            ;call display subroutine
        LCALL       DELAY
        MOV     A, #'L'          ;display letter L
        LCALL       DATAWRT            ;call display subroutine
        LCALL       DELAY              ;give LCD some time
        MOV     A, #'C'          ;display letter C
        LCALL       DATAWRT            ;call display subroutine
        LCALL   DELAY
        MOV     A, #'O'          ;display letter O
        LCALL       DATAWRT            ;call display subroutine
        LCALL       DELAY              ;give LCD some time
        MOV     A, #'M'          ;display letter M
        LCALL       DATAWRT            ;call display subroutine
        LCALL       DELAY               ;give LCD some time
        MOV    A, #'E'          ;display letter E
        LCALL       DATAWRT             ;call display subroutine
        LCALL       DELAY               ;give LCD some time

        MOV    A, #0C5H    ;cursor at line 1 postion 1
        LCALL       COMNWRT             ;call command subroutine
        LCALL       DELAY               ;give LCD some time

        MOV    A, #'C'          ;display letter C
        LCALL       DATAWRT             ;call display subroutine
        LCALL       DELAY               ;give LCD some time
        MOV    A, #'S'          ;display letter S
        LCALL       DATAWRT             ;call display subroutine
        LCALL       DELAY
        MOV    A, #'E'          ;display letter E
        LCALL       DATAWRT             ;call display subroutine
        LCALL       DELAY               ;give LCD some time
        MOV  A, #'-'            ;display letter -
        LCALL       DATAWRT
   LCALL      DELAY         ;call display subroutine
        MOV     A, #'A'                 ;display letter A
        LCALL       DATAWRT             ;call display subroutine
        LCALL       DELAY               ;give LCD some time
        MOV     A, #'I'                 ;display letter I
        LCALL       DATAWRT             ;call display subroutine
        LCALL       DELAY
        MOV     A, #'M'                 ;display letter M
        LCALL       DATAWRT             ;call display subroutine
        LCALL       DELAY
        MOV     A, #'L'                 ;display letter L
        LCALL       DATAWRT             ;call display subroutine
        LCALL       DELAY

        AJMP  LCD_IN

COMNWRT:LCALL READY             ;send command to LCD
        MOV  P1, A              ;copy reg A to port 1
        CLR  P3.4               ;RS=0 for command
        CLR  P3.5               ;R/W=0 for write
        SETB P3.6               ;E-1 for high pulse
        ACALL       DELAY       ;give LCD some time
        CLR  P3.6               ;E=0 for H-to-L pulse
        RET

DATAWRT:LCALL  READY            ;write data to LCD
        MOV  P1, A              ;copy reg A to port1
        SETB P3.4               ;RS=1 for data
        CLR  P3.5               ;R/W=0 for write
        SETB P3.6               ;E=1 for high pulse
        ACALL       DELAY       ;give LCD some time
        CLR  P3.6               ;E=0 for H-to-L pulse
        RET
```

```
READY:CLR  P3.4
       SETBP3.5
       RET

DELAY:       MOV   R3, #255       ;50 or higher for fast CPUs
HERE2:       MOV   R4, #255       ;R4=255
HERE:  DJNZ  R4, HERE             ;stay untill R4 becomes 0
       DJNZ  R3, HERE2
       RET

       END
```

Flowchart:- (Student will draw flowchart on blank page)

Input:- WELCOME
CSE-AIML

Output:-



Result:-

Conclusion:-

Experiment-8

Aim:- Write an assembly language program

ii)

Apparatus/Tools Required:-

Sr. No.	Apparatus/Tools	Description	Quantity
1	PC/Laptop	---	1
2	Keil µVision4	IDE Tool	1
3	Ultra 8051 Board	8051 Trainer Kit	1

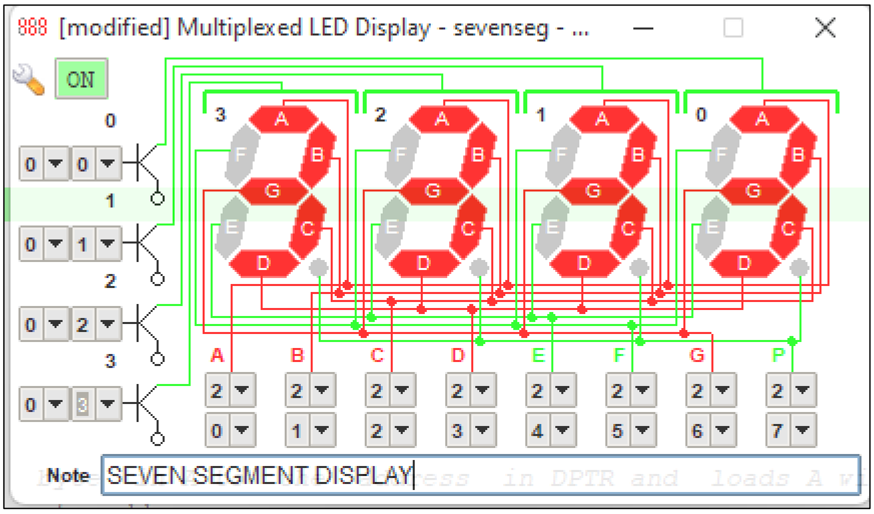
Program:-

```
ORG   0000H ;initial starting address
MOV   DPTR,#CC_PATTERNS ;  loads  the  adress  Lookup Table
START: MOV          A,#0FFH        ; initial value of accumulator
        MOV         B,A
        MOV         R0,#0AH ;Register R0 initialized as counter
LOOP:  MOV A,B
        INC A
       MOV          B,A
       ACALL   DISPLAY ;
       MOV          P0,A
       ACALL        DELAY          ; calls the delay of the timer
       DEC          R0     ;Counter R0 decremented by 1
       MOV          A,R0   ; R0 moved to accumulator to check if it is zero in next
instruction.
      JZ       START          ;Checks accumulator for zero and jumps to START. Done to
check if counting has been finished.
      SJMP  LOOP

DELAY:        MOV  R3, #255        ;50 or higher for fast CPUs
HERE2:        MOV  R4, #255        ;R4=255
HERE: DJNZ  R4, HERE               ;stay untill R4 becomes 0
      DJNZ  R3, HERE2
      RET
DISPLAY:
      MOVC          A,@A+DPTR ;  adds  the  byte  in A to  the  address  in DPTR and
loads A with  data  present  in  the  resultant  address
      RET

CC_PATTERNS:
DB 0c0h,0f9h,0a4h,0b0h,099h,092h,082h,0f8h,080h,090h,0 ; COMMON ANODE
CONFIGURATION

      END
```

Flowchart:- (Student will draw flowchart on blank page)

Input:-0c0h,0f9h,0a4h,0b0h,099h,092h,082h,0f8h,080h,090h,0 ; COMMON ANODE CONFIGURATION

Output:-



[modified] Multiplexed LED Display - sevenseg - ...

Note: SEVEN SEGMENT DISPLAY

Result:-

Conclusion:-

Experiment-9

Aim:- Write an Assembly Language Program to perform addition of two 32 bit numbers stored at memory location using ARM board FRDM-L25Z.

Apparatus/Tools Required:-

Sr. No.	Apparatus/Tools	Description	Quantity
1	PC/Laptop	---	1
2	Keil µVision5	IDE Tool	1

Program:-

Instruction	**Description**
AREA MYCODE, CODE, READONLY	
value1 EQU 0x1FFFF000	Value1 first memory location (first data)
value2 EQU 0x1FFFF010	Value 2 second memory location(second data)
value3 EQU 0x1FFFF020	Value 3 third memory location (result)
__main	
LDR R0,=value1	Load register R0 with value1
LDR R1,=value2	Load register R1 with value2
LDRB R3,[R0]	Load R3 with data present at location of value1
LDRB R4,[R1]	Load R4 with data present at location of value2
ADDS R5,R4,R3	Add R3 and R4 and store result in R5
LDR R0,=value3	Load R0 with value 3
STRB R5,[R0]	Stored R5 result data at value3 location
HERE B HERE	Continue in same loop
END	

Flowchart:- (Student will draw flowchart on blank page)

Input:-

0x1FFFF000	13
0x1FFFF010	12

Output:-

0x1FFFF020	25

Result:-

Conclusion:-

Experiment-10

Aim:- Write an Assembly Language Program to perform subtraction of two 32 bit numbers stored at memory location using ARM board FRDM-L25Z.

Apparatus/Tools Required:-

Sr. No.	Apparatus/Tools	Description	Quantity
1	PC/Laptop	---	1
2	Keil µVision5	IDE Tool	1

Program:-

Instruction	**Description**
AREA MYCODE, CODE, READONLY	
value1 EQU 0x1FFFF000	Value1 first memory location (first data)
value2 EQU 0x1FFFF010	Value 2 second memory location(second data)
value3 EQU 0x1FFFF020	Value 3 third memory location (result)
__main	
LDR R0,=value1	Load register R0 with value1
LDR R1,=value2	Load register R1 with value2
LDRB R3,[R0]	Load R3 with data present at location of value1
LDRB R4,[R1]	Load R4 with data present at location of value2
SUBS R5,R4,R3	Subtract R3 from R4 and store result in R5
LDR R0,=value3	Load R0 with value 3
STRB R5,[R0]	Stored R5 result data at value3 location
HERE B HERE	Continue in same loop
END	

Flowchart:- (Student will draw flowchart on blank page)

Input:-

1FFFF000	04
1FFFF010	08

Output:-

0x1FFFF020	04

Result:-

Conclusion:-