EXPERIMENT NO: 01 **DATE:**

AIM: Introduction to IDE and Assembler directives.

Theory:

Integrsted Development Environment:

An integrated development environment (IDE) is a software suite that consolidates the basic tools developers need to write and test software. Typically, an IDE contains a code editor, a compiler or interpreter and a debugger that the developer accesses through a single graphical user interface (GUI). An IDE may be a standalone application, or it may be included as part of one or more existing and compatible applications.

What is Assembler?

A program for converting instructions written in low-level symbolic code (Assembly Language)into machine code.

Assembler Directives:

An assembler directive is a message to the assembler that tells the assembler something it needs to know in order to carry out the assembly process; for example, an assemble directive tess the assembler where a program is to be located in memory.

Some examles of Assembler Directives:

- **DB -** Defined Byte.
- **ORG -** Origin
- **End –** End the program
- **EQU -** Equate

DB:

The **DB** directive is the most widely used data directive in the assembler. It is used to define the 8-bit data. When DB is used to define data, the numbers can be in decimal, binary, hex, ASCII formats

ORG (origin):

The ORG directive is used to indicate the beginning of the address. The number that comes after ORG can be either in hex and decimal. If the number is not followed by H, it is decimal and the assembler will convert it to hex.

END:

This indicates to the assembler the end of the source (asm) file. The END directive is the last line of an 8051 program. Mean that in the code anything after the END directive is ignored by the assembler.

EQU (equate):

This is used to define a constant without occupying a memory location. The EQU directive does not set aside storage for a data item but associates a constant value with a data label. When the label appears in the program, its constant value will be substituted for the label. Assume that there is a constant used in many different places in the program, and the programmer wants to change its value throughout. By the use of EQU, one can change it once and the assembler will change all of its occurrences

Introduction to Micro vision-3 IDE for 8051:

The µVision3 IDE is a Windows-based software development platform that combines a robust editor, project manager, and make facility. µVision3 integrates all tools including the C compiler, macro assembler, linker/locator, and HEX file generator. µVision3 helps expedite the development process of your embedded applications by providing the following:

- Full-featured source code editor,
- Device database for configuring the development tool setting,
- Project manager for creating and maintaining your projects,
- Integrated make facility for assembling, compiling, and linking your embedded applications,
- Dialogs for all development tool settings,
- True integrated source-level Debugger with high-speed CPU and peripheral simulator,
- Advanced GDI interface for software debugging in the target hardware and for connection to Keil ULINK,
- Flash programming utility for downloading the application program into Flash ROM
- Links to development tools manuals, device datasheets & user's guides.

How to Create Project:

µVision3 is a standard Windows application and started by clicking on the program icon. About the Environment describes the different window areas of µVision3.

µVision3 includes a project manager which makes it easy to design applications for an ARM based microcontroller. You need to perform the following steps to create a new project:

Steps:

1. Select the Toolset (only required for ARM Projects).
2. Create Project File and Select CPU.
3. Project Workspace - Books.
4. Create New Source Files.
5. Add Source Files to the Project.
6. Create File Groups.
7. Set Tool Options for Target Hardware.
8. Configure the CPU Startup Code.

9. Build Project and Generate Application Program Code.
10. Create a HEX File for PROM Programming.
11. The section provides a step-by-step tutorial that shows you how to create a simple μVision3 project.

Conclusion:

Signature **MARKS**

EXPERIMENT NO: 02 **DATE:**

AIM: Write and execute 8051 Assembly language program using Arithmetic instructions. Verify the result for the same.

ASSEMBLY LANGUAGE PROAGRAMS:

PROGRAM 1: Write an 8051 assembly language program to add two 8 bit numbers stored in register R6 and R7 of bank 0. Store the result in register R5.

Lable	Instructions	Comments

Result:

Before Execution		After Execution	
Content of R6		Content of R6	
Content of R7		Content of R7	
Content of R5		Content of R5	

PROGRAM 2: Write an 8051 assembly language program to multiply two 8 bit numbers stored in register R6 and R7. Store the results in register R5(Lower Byte) and R4(Higher Byte).

Lable	Instructions	Comments

Result:

Before Execution		After Execution	
Content of R6		Content of R6	
Content of R7		Content of R7	
Content of R5		Content of R5	
Content of R4		Content of R4	

PROGRAM 3: Write an 8051 assembly language program to subtract contents of register R6 from R7.Store the result in register R5.

Lable	**Instructions**	**Comments**

Result:

Before Execution		**After Execution**	
Content of R6		**Content of R6**	
Content of R7		**Content of R7**	
Content of R5		**Content of R5**	

PROGRAM 4: Write an 8051 assembly language program to divide two 8 bit numbers stored in register R6 and R7. Store the result in register R5(Answer)and R4 (Remainder).

Lable	Instructions	Comments

Result:

Before Execution		**After Execution**	
Content of R6		**Content of R6**	
Content of R7		**Content of R7**	
Content of R5		**Content of R5**	
Content of R4		**Content of R4**	

Conclusion:

EXERSICE:

[1] Write assembly language program to add two 16 bit data stored at memory locations 60h-61h and 62-63h. Store result at location 82h(LSB) and 83h (MSB).

[2] Write and execute instructions to

(1) Transfer content 1234h to DPTR.

(2) Select register bank 1 and transfer content 45h to register R0 and 54h to R2.

[3] Write and execute program to subtract content of register R6 from register R7 and store result in register R0.

Signature **MARKS**

EXPERIMENT NO: 03 **DATE:**

AIM: Write and execute 8051 Assembly language programming for block data transfer between internal and external memory. Verify the result for the same.

ASSEMBLY LANGUAGE PROAGRAMS:

PROGRAM 1: Transfer block of data from the location 40h-4Bh to external memory location 2000h-200Bh. Verify the result.

Lable	Instructions	Comments

Result:

Before execution				After execution			
Internal Location	Content	External Location	Content	Internal Location	Content	External Location	Content

PROGRAM 2: Transfer block of data from the location 20h-2Bh to internal memory location 40h-4Bh in reverse order. Verify the result.

Lable	Instructions	Comments

Result:

Before execution		After execution	
Location	**Content**	**Location**	**Content**

Conclusion:

EXERSICE:

[1] Write program to arrange data stored in the location 20h-2Fh in ascending order.

Signature **MARKS**

EXPERIMENT NO: 04 **DATE:**

AIM: Write and execute 8051 Assembly language program for code conversion. Verify the result for the same.

ASSEMBLY LANGUAGE PROAGRAMS:

PROGRAM 1: Write a program to convert HEX number stored at 40H in to equivalent ASCII code. Store the result at 50H.

Lable	Instructions	Comments

RESULT:

Before Execution		After Execution	
40 H		**40 H**	
50 H		**50 H**	

PROGRAM 2: Write a program to convert a BCD number stored at ram location 30H into its equivalent binary number and store the result in internal ram location at 40H**.**

Lable	Instructions	Comments

Conclusion:

EXERCISE:

[1] Assume that register A has packed BCD. Write a program to convert packed BCD to two ASCII numbers and place them in R2 and R6.

Signature **MARKS**

EXPERIMENT NO: 05 **DATE:**

AIM: Write and execute 8051 Assembly language program for Separation of Odd/Even, Positive/Negative numbers from a set of data. Verify the result for the same.

ASSEMBLY LANGUAGE PROAGRAMS:

PROGRAM 1: Write a program to separate odd number from a set of ten 8-bit numbers stored in internal RAM starting from 30H and store odd numbers starting from 40H onwards.

Lable	Instructions	Comments

Result:

Before execution		After execution	
Location	Content	Location	Content

PROGRAM 2: Write a program to count positive and negetive number from a set of ten 8-bit numbers stored in internal RAM starting from 40H and store count of positive and negative numbers at location 50 and 51H respectively.

Lable	Instructions	Comments

Result:

Before execution		After execution	
Location	**Content**	**Location**	**Content**

Conclusion:

EXERCISE:

[1] Write a program to separate odd and even number from a set of ten 8-bit numbers stored in internal RAM starting from 30H and store even numbers starting from 40H and odd numbers starting from 40H.

Signature **MARKS**

EXPERIMENT NO: 06 **DATE:**

AIM: Write and execute 8051 Assembly language program for Timers in different modes. Verify the result for the same.

ASSEMBLY LANGUAGE PROAGRAMS:

PROGRAM 1: Write a program that blinks the LEDs connected with port-1 with 1 sec ON-OFF delay generated using Timer-0 in Mode 1.

Calculations for Delay:

Registers' Details: Draw TMOD and TCON

ASSEMBLY LANGUAGE PROAGRAM:

Lable	Instructions	Comments

PROGRAM 2: Write a program to generate square wave of 50% duty cycle having frequency 5 KHz at port pin P1.0 using timer 1 in mode 2.

Calculations for Delay:

Registers' Details: Draw TMOD and TCON

ASSEMBLY LANGUAGE PROAGRAM:

Lable	Instructions	Comments

Conclusion:

EXERSICE:

[1] Write an ALP to generate square wave of 75% duty cycle having frequency 10 kHz at port pin P0.1 using software delay. Crystal frequency is 11.0592 MHz.

Signature **MARKS**

EXPERIMENT NO: 07 **DATE:**

AIM: Write and execute 8051 C language program for I/O Port

C LANGUAGE PROAGRAMS:

PROGRAM 1: Write a C program that makes an LED ON when switch S1 is pressed and switched OFF LED when switch S2 is pressed.

RESULT:
When (Key=S1) P2.0 is pressed: **LED:**
When (Key=S2) P2.1 is pressed: **LED:**

PROGRAM 2: Write program to read switch connected at port pin P1.0, toggle it and send to port pin P1.1

Conclusion:

EXERSICE:

[1] Write C language program to continuously toggle pin P1.0 without disturbing other port pins.

[2] Write C language program to perform OR operation between port pin P1.0 and P1.1. Display result on port pin P1.2.

[3] Write C language program to read port P1, Compare content of port P1 with data 80h. If data at port P1 is greater than 80h, make port P0=0x00 and if data at port P1 is less than or equal to 80h, make port P0=0xFF.

Signature **MARKS**

EXPERIMENT NO: 08 **DATE:**

AIM: Write and execute 8051 Timers and Counters programming in embedded C for time delay. Verify the result for the same.

C LANGUAGE PROAGRAMS:

PROGRAM: Write an 8051 C program to toggle all bits of P2 continuously every 1 second. Use Timer 1 Mode 1 to create the delay.

Calculation:

Program:

Conclusion:

EXERCISE:

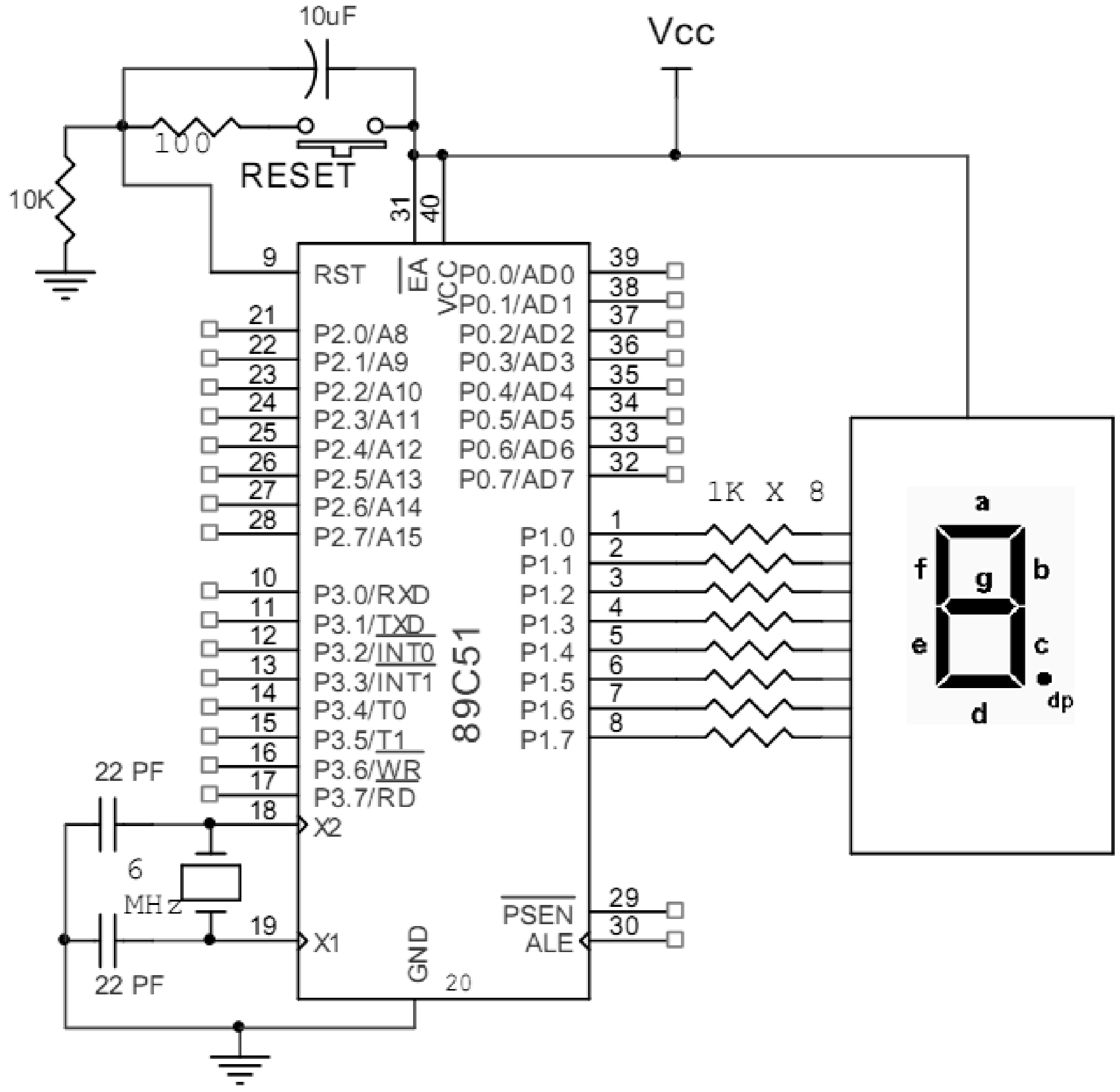
[1] Write an 8051 program to creat a frequency of 2500Hz on Pin 2.7. Use Timer 1 Mode 2 to creat the delay. Show related calculations.

Signature **MARKS**

EXPERIMENT NO: 09 **DATE:**

AIM: To study Interfacing of seven segment display with Port P1. Write a program to display number 0 to 9 on the seven segment display at the interval of 1 second. Verify the result for the same.

CIRCUIT DIAGRAM:



- Except 7 segment display interfacing, this circuit diagram shows minimum interfacing required for microcontroller to operate. That is, Reset circuit, Crystal oscillator, Vcc=5V, GND & EA' Pin.

Generation of Look-up Table:

PROAGRAM:

Conclusion:

EXERSICE:

[1] Write program to display count value F to 0 at the interval of 1 second on seven segment display connected at port P1. Use Timer 0 in Mode 1. Crystal frequency = 12 MHz.

Signature **MARKS**

EXPERIMENT NO: 10 **DATE:**

AIM: To study Interfacing of LCD with the microcontroller. Display your name on the LCD.

INTERFACING DIAGRAM:

Necessary Commands & Pins Description:

PROGRAM:

Conclusion:

EXERSICE:

[1] Modify LCD program to display two lines: “WELCOME TO ELE.” on first line and “FETR, ISROLI” on the second line. Execute program in your hardware.

Signature **MARKS**