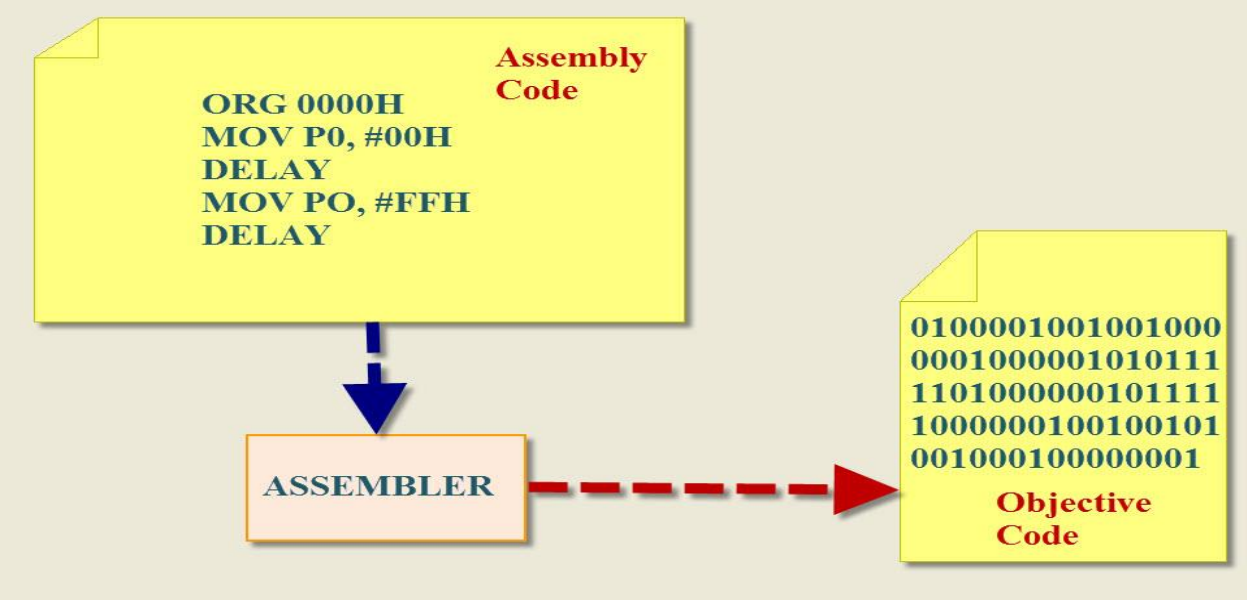
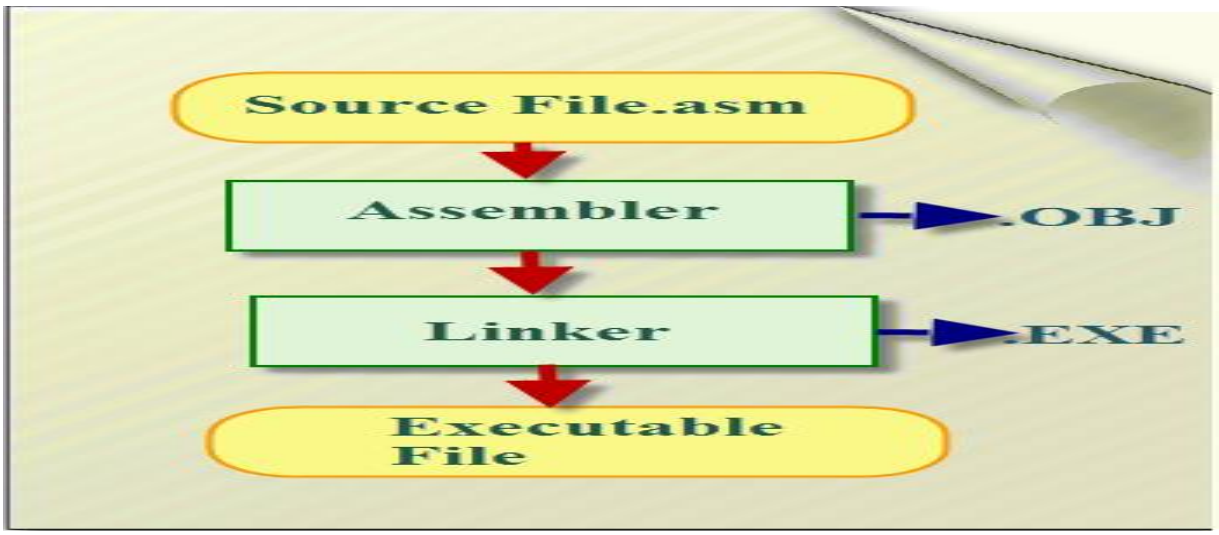
8051 Basic Programming

The assembly language is a low-level programming language used to write program code in terms of mnemonics. Even though there are many high-level languages that are currently in demand, assembly programming g language is popularly used in many applications. It can be used for direct hardware manipulations. It is also used to write the 8051 programming code efficiently with less number of clock cycles by consuming less memory compared to the other high-level languages.



8051 Programming in Assembly Language

The assembly language is a fully hardware related programming language. The embedded designers must have sufficient knowledge on hardware of particular processor or controllers before writing the program. The assembly language is developed by mnemonics; therefore, users cannot understand it easily to modify the program.



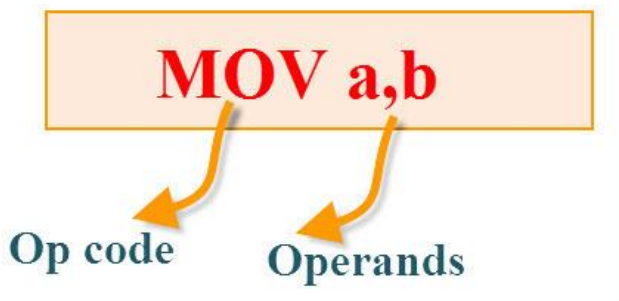
8051 Programming in Assembly Language

Assembly programming language is developed by various compilers and the "keiluvison" is best suitable for microcontroller programming development. Microcontrollers or processors can understand only binary language in the form of '0s or 1s'; An assembler converts the assembly language to binary language, and then stores it in the microcontroller memory to perform the specific task.

Rules of Assembly Language

- The assembly code must be written in upper case letters
- The labels must be followed by a colon (label:)
- All symbols and labels must begin with a letter
- All comments are typed in lower case
- The last line of the program must be the END directive

The assembly language mnemonics are in the form of op-code, such as MOV, ADD, JMP, and so on, which are used to perform the operations.



Op-code: The op-code is a single instruction that can be executed by the CPU. Here the op-code is a MOV instruction.
Operands: The operands are a single piece of data that can be operated by the op-code. Example, multiplication operation is performed by the operands that are multiplied by the operand.
Syntax: MUL a,b;

The Elements of an Assembly Language Programming:

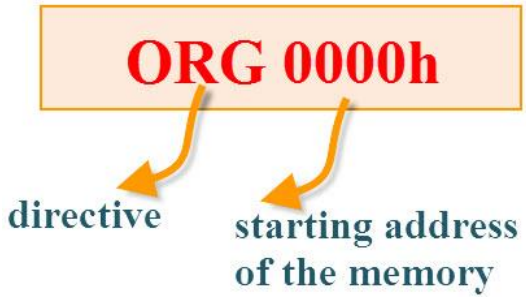
- Assembler Directives
- Instruction Set
- Addressing Modes

Assembler Directives:
The assembling directives give the directions to the CPU. The 8051 microcontroller consists of various kinds of assembly directives to give the direction to the control unit. The most useful directives are 8051 programming, such as:

- ORG
- DB
- EQU
- END

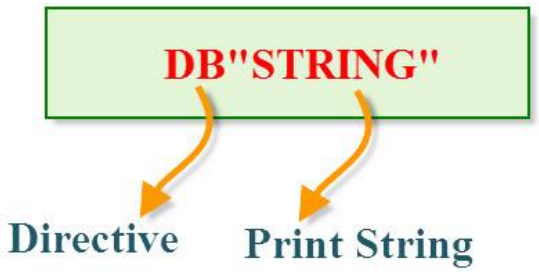
ORG(origin): This directive indicates the start of the program. This is used to set the register address during assembly. For example; ORG 0000h tells the compiler all subsequent code starting at address 0000h.

Syntax: ORG 0000h



DB(define byte): The define byte is used to allow a string of bytes. For example, print the “EDGEFX” wherein each character is taken by the address and finally prints the “string” by the DB directly with double quotes.
Syntax:
ORG 0000h

MOV a, #00h
————-
————-
DB”EDGEFX”



EQU (equivalent): The equivalent directive is used to equate address of the variable.
Syntax:
reg equ,09h
——————
——————
MOV reg,#2h

END:The END directive is used to indicate the end of the program.
Syntax:
reg equ,09h
——————
——————
MOV reg,#2h
END

What is arithmetic in assembly language?

Arithmetic operators expect numeric operands and produce a numeric result. They are most frequently used in sub-expressions. The following top-level expression means that the quantity of item C in the solution must be the same as the sum of the quantities of items A and B.

What is arithmetic operations in microcontroller?

The arithmetic instructions define the set of operations performed by the processor Arithmetic Logic Unit (ALU). The arithmetic instructions are further classified into binary, decimal, logical, shift/rotate, and bit/byte manipulation instructions.

Which registers of 8051 are used for mathematical arithmetic operation?

The **Accumulator or Register A** is the most important and most used 8051 **Microcontroller SFRs**. The Register A is located at the address E0H in the SFR memory space. The Accumulator is used to hold the data for almost all the ALU Operations.

Which register in 8051 microcontroller is used for arithmetic and logical operations?

Registers

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.

What are arithmetic operations in programming?

Arithmetic Operation - is a function which can perform one of the following tasks: adding, subtracting, multiplying and dividing. Operator - a programming device that performs a function on one or more inputs, for example arithmetic operators (+,-,/,*)

What are the advantages of arithmetic operations?

A main advantage of arithmetic operators is that the process is very simple and therefore fast. Logical operators are often used to combine two (mostly binary) images. In the case of integer images, the logical operator is normally applied in a bitwise way.

Arithmetic Instruction Set:

The arithmetic instructions perform the basic operations such as:

- Addition
- Multiplication
- Subtraction
- Division

Addition:

```
ORG                                                    0000h
MOV R0, #03H          // move the value 3 to the register R0//
MOV A, #05H           // move the value 5 to accumulator A//
Add A, 00H            // addA value with R0 value and stores the result inA//
END
```

Multiplication:

```
ORG                                                    0000h
MOV R0, #03H          // move the value 3 to the register R0//
MOV A, #05H           // move the value 5 to accumulator A//
MUL A, 03H            // Multiplied result is stored in the Accumulator A //
END
```

Subtraction:

```
ORG 0000h
MOV R0, #03H     // move the value 3 to register R0//
MOV A, #05H      // move the value 5 to accumulator A//
SUBB A, 03H      // Result value is stored in the Accumulator A //
END
```

Division:

```
ORG 0000h
MOV R0, #03H     // move the value 3 to register R0//
MOV A, #15H      // move the value 5 to accumulator A//
DIV A, 03H       // final value is stored in the Accumulator A //
END
```

Sum of 8-bit Numbers Stored in Memory

```
ORG 00H
MOV R0, #50H   ;get memory location in memory pointer R0
MOV R1, #51H   ;get memory location on memory pointer register R1
MOV A,@R0      ;get content of memory location 50H to accumulator
ADD A,@R1      ;add content of A with content of memory location 51H and store result in A
MOV R0,#52H    ;get 52H to memory pointer R0
MOV@R0,A       ;copy content of A to memory location 52H
END
```

ADD 16-bit Numbers

Program

```
ORG 00H
MOV DPTR,#2040H  ; get 2040H into DPTR
MOV A,#2BH       ;get lower byte of second 16-bit number on accumulator
MOV R0,#20H      ;get higher byte of second 16-bit number on accumulator
ADD A,82H        ;[A]+[DPL]
MOV 82H,A        ;save result of lower byte addition
MOV A,R0         ;get higher byte of second number in A
ADDC A,83H       ;[A]+[DPH]
MOV 83H,A        ;Save result of higher byte addition
END
```