8086 assembler tutorial for beginners (part 10)

Macros

Macros are just like procedures, but not really. Macros look like procedures, but they exist only until your code is compiled, after compilation all macros are replaced with real instructions. If you declared a macro and never used it in your code, compiler will simply ignore it. **emu8086.inc** is a good example of how macros can be used, this file contains several macros to make coding easier for you.

Macro definition:

```
name    MACRO  [parameters,...]

            <instructions>

ENDM
```

Unlike procedures, macros should be defined above the code that uses it, for example:

```
MyMacro    MACRO  p1, p2, p3

     MOV AX, p1
     MOV BX, p2
     MOV CX, p3

ENDM

ORG 100h

MyMacro 1, 2, 3

MyMacro 4, 5, DX

RET
```

The above code is expanded into:

```
MOV AX, 00001h

MOV BX, 00002h

MOV CX, 00003h

MOV AX, 00004h

MOV BX, 00005h
```

```
MOV CX, DX
```

Some important facts about **macros** and **procedures**:

- When you want to use a procedure you should use **CALL** instruction, for example:

```
CALL MyProc
```

- When you want to use a macro, you can just type its name. For example:

```
MyMacro
```

- Procedure is located at some specific address in memory, and if you use the same procedure 100 times, the CPU will transfer control to this part of the memory. The control will be returned back to the program by **RET** instruction. The **stack** is used to keep the return address. The **CALL** instruction takes about 3 bytes, so the size of the output executable file grows very insignificantly, no matter how many time the procedure is used.
- Macro is expanded directly in program's code. So if you use the same macro 100 times, the compiler expands the macro 100 times, making the output executable file larger and larger, each time all instructions of a macro are inserted.
- You should use **stack** or any general purpose registers to pass parameters to procedure.
- To pass parameters to macro, you can just type them after the macro name. For example:

```
MyMacro 1, 2, 3
```

- To mark the end of the macro **ENDM** directive is enough.
- To mark the end of the procedure, you should type the name of the procedure before the **ENDP** directive.

Macros are expanded directly in code, therefore if there are labels inside the macro definition you may get "Duplicate declaration" error when macro is used for twice or more. To avoid such problem, use **LOCAL** directive followed by names of variables, labels or procedure names. For example:

```
MyMacro2    MACRO
        LOCAL label1, label2
```

```
        CMP  AX, 2
        JE label1
        CMP  AX, 3
        JE label2
        label1:
                 INC  AX
        label2:
                 ADD  AX, 2
ENDM


ORG 100h

MyMacro2

MyMacro2

RET
```

If you plan to use your macros in several programs, it may be a good idea to place all macros in a separate file. Place that file in **Inc** folder and use **INCLUDE *file-name*** directive to use macros. See **Library of common functions - emu8086.inc** for an example of such file.

<<< previous part <<< **>>> Next Part >>>**