Macros

Objectives

- Macros
- Directives
  - **LOCAL**
  - **Conditional**
  - **INCLUDE**

What Are Macros?

- **A macro is a set of instructions identified with a unique name**
- **Wherever the name of the macro occurs in a program, it is replaced (by the assembler) with its associated set of instructions. This process is called macro expansion.**
- **Macros have the same purposes as procedures. However, they are different.**

Why Macros

Simplify and reduce the amount of repetitive code

- **reduce errors caused by repetitive coding.**
- **make it easier to program**
- **Macro may be reused (create a macro library)**
- **make it easier to maintain**

Macro Format

```
MacroName MACRO [parameter List]
          ...
          Instructions
          ...
          ENDM
```

Example

Define a macro that initializes the segment registers for .EXE programs.

```
Init    MACRO
        MOV AX,@data
        MOV DS,AX
        MOV ES,AX
        ENDM
```

Example

Define a macro that terminates a .EXE program.

```
Terminate MACRO
          MOV AX, 4C00H
          INT 21H
          ENDM
```

Implement Macros

Example: a program to display a string on screen.

```
                page  60, 132
    TITLE       EXAMPLE (EXE)
    Init        MACRO               ; Macro definition
                MOV AX, @data
                MOV DS, AX
                MOV ES, AX
                ENDM
    Terminate   MACRO               ; Macro definition
                MOV AX, 4C00H
                INT 21H
                ENDM
```

Example

```
          .Model SMALL
          .Data
MSG       DB    'Test macros',13,10,'$'
          .CODE
Main      PROC  FAR
          Init                ;Macro reference
          LEA   DX, MSG
          MOV   AH, 09H
          INT   21H
          Terminate           ;Macro reference
Main      ENDP
          END
```

Example

- **Define a macros that displays a message on the screen.**
- **Need to pass the message as a parameter.**

```
Prompt   MACRO    STRG
         LEA      DX, STRG
         MOV      AX, 09H
         INT      21H
         ENDM
```

Example

```
        .Model SMALL
        .Data
MSG     DB    'Test macros',13,10,'$'
        .CODE
Main    PROC  FAR
        Init              ;Macro reference
        Prompt MSG
        Terminate         ;Macro reference
Main    ENDP
        END
```

Macro Parameter Passing

- When the assembler expands macro `Prompt`, it replaces every occurrence of the string `STRG` in the macro definition with the string provided in the macro reference
- parameters used with macro are dummy parameters: they are replaced textually during macro expansion
- register can be used as a parameter in a call

Example

Define a macro that multiplies the contents of AX by an unsigned immediate value and leaves the product in DX:AX

Example: **AX * 7**

```
MOV     BX,7
MUL     BX
```

```
Mult    MACRO     Const
        MOV  BX, Const
        MUL  BX
        ENDM
```

Example

```
            .Model SMALL
            .Data
Data1       DW          2310H
            .CODE
Main        PROC        FAR
            Init
            Mult        7
            Mult        CX
            Mult        Data1
            Terminate
Main        ENDP
            END
```

Example

- **Need to modify Mult so that it multiply AX by a signed or unsigned immediate.**
- **Note: need to MUL with unsigned multiplier and IMUL with signed multiplier.**

```
Mult    MACRO Oper const
        MOV   BX,const
        Oper  BX
        ENDM
```

Example

```
Mult   MUL  7
```

will be expanded into:

```
MOV BX, 7
MUL BX
```

```
Mult   IMUL -12
```

will be expanded into

```
MOV  BX,-12
IMUL BX
```

Labels and the LOCAL Directive

- **Define a macro that computes the absolute value of a general register or memory location and leaves the result as the new value for that register or memory location.**
- **Algorithm:**

  if original value is positive

  leave it as is

  else (if original value is negative)

  negate it.

Labels and the LOCAL Directive

```
ABS         MACRO   Num
            CMP     Num, 0
            JGE     ENDABS
            NEG     Num
ENDABS:
            ENDM
```

- What is the problem with above macro?

Labels and the LOCAL Directive

Problem with labels defined inside macros

-- if macro is referenced more than once in a program, the same label will occur several times in the same program. This will generate an error because labels are supposed to be unique.

Example

```
ABS   AX
ABS   BX
```

will be expanded into:

```
                CMP         AX, 0
                JGE         ENDABS
                NEG         AX
ENDABS:         CMP         BX, 0
                JGE         ENDABS
                NEG         BX
ENDABS:
                ...
```

Labels and the LOCAL Directive

- **Use the LOCAL directive**

```
ABS             MACRO     num
                LOCAL     ENDABS
                CMP       num, 0
                JGE       ENDABS
                NEG       num
ENDABS:
                ENDM
```

When macro is expanded, ENDABS label will be replaced by an unused label ??0000-??FFFF

Example

```
ABS   AX
ABS   BX
```

will be expanded into:

```
                CMP         AX, 0
                JGE         ??0000
                NEG         AX
        ??0000:
                CMP         BX, 0
                JGE         ??0001
                NEG         BX
        ??0001:
                ...
```

Including Macros in a Library

- **Commonly, users save their macro definition in a library and store it in a disk file.**
- **You may use any editor to define your macro. Save in an ASCII file.**
- **To use a macro from the library, use the INCLUDE directive to include the library file:**

```
INCLUDE   Library-name
```

INCLUDE Directive

```
        .Model SMALL
        INCLUDE   A:\MacroLib
        .Data
MSGDB   'Test macros', 13, 10, '$'
        .CODE
MAIN    PROC      FAR
        Init
        Prompt    MSG
        Terminate
MAIN    ENDP
        END       MAIN
```

The INCLUDE directive

- **The assembler will replace the INCLUDE line with the contents of the referenced library file.**

Conditional Directives

- **General format:**

```
IFxxx       (condition)
    ...
ELSE    (this is optional)
    ...
ENDIF
```

Conditional Directives

- **IF const_expr**
- **IFE const_expr**
- **IFDEF symbol_name**
- **IFNDEF symbol_name**
- **IFB <arg>**
- **IFNB <arg>**

Conditional Directives

`IF const_expr`

if expr evaluates to non-zero assemble the statements within the conditional block.

`IFE const_expr`

if expr evaluates to zero assemble the statements within the conditional block.

`IFDEF symbol`

assemble conditional block if the symbol name is defined in the program or has been declared as `EXTRN`

Conditional Directives

`IFNDEF const_expr`

assemble conditional block if symbol name is NOT defined in program and has NOT been declared as EXTRN

`IFB <arg>`

assemble conditional block if arg is blank

`IFNB <arg>`

assemble conditional block if arg is not blank

Example

- **Define a macro for dividing AX by a 1-byte in memory. Test divisor before division.**

```
ChkDIV   MACRO Divisor
         LOCAL ENDDIV
         CMP   Divisor,0
         JE    ENDDIV
         CMP   AH,divisor
         JNB   ENDDIV
         DIV   Divisor
ENDDIV:
         ENDM
```

Example

```
ChkDIV   MACRO   divisor
         LOCAL   ENDDIV
         IFNDEF  Divisor
         ;Macro expansion terminated
         EXITM
         ENDIF
         CMP     Divisor, 0
         JE      ENDDIV
         CMP     AH,divisor
         JNB     ENDDIV
         DIV     Divisor
ENDDIV:
         ENDM
```

Example

- **All INT 21H calls require a function number in AH and some require a value in DX. Define a macro that loads AH, conditionally loads DX, and calls**

  INT 21H

Example

```
INT21   MACRO   Function  Strg
        MOV AH, Function
        IFNB    <Strg>
        LEA     DX, Strg
        ENDIF
        INT 21H
        ENDM
```