Today’s Topics

- Macros
- Brief overview of Chapter 9, e.g.,
  - MOVSB…etc.
  - LODSB, STOSB…etc.
  - REP Prefix
Study Guide

• Section 10.2: Macros

Quickly browse Chapter 9 and Sections 10.1, 10.3, and 10.4

Introducing Macros

• A macro\(^1\) is a named block of assembly language statements.
• Once defined, it can be invoked (called) one or more times.
• During the assembler's preprocessing step, each macro call is expanded into a copy of the macro.
• The expanded code is passed to the assembly step, where it is checked for correctness.

\(^1\)Also called a macro procedure.
Defining Macros

- A macro must be defined before it can be used.
- Parameters are optional.
- Each parameter follows the rules for identifiers. It is a string that is assigned a value when the macro is invoked.

Syntax:

```
macroName MACRO [parameter-1, parameter-2,...]  
statement-list  
ENDM
```

mPutChar Macro

Writes a single character to standard output.

Definition:

```
mPutchar MACRO char  
push eax  
mov al,char  
call WriteChar  
pop eax  
ENDM
```

Invocation:

```
.code  
mPutchar 'A'
```

Expansion:

```
1 push eax  
1 mov al,'A'  
1 call WriteChar  
1 pop eax
```

viewed in the listing file
Invoking Macros

- Each parameter is replaced by its corresponding argument when the macro is expanded.
- When a macro expands, it generates assembly language source code.
- Arguments are treated as simple text by the preprocessor.

Invalid Argument

- If you pass an invalid argument, the error is caught when the expanded code is assembled.
- Example:
  
  ```
  .code
  mPutchar 1234h
  push eax
  mov al,1234h ; error!
  call WriteChar
  pop eax
  ```
Blank Argument

- If you pass a blank argument, the error is also caught when the expanded code is assembled.
- Example:

```assembly
.code
m_putchar

1  push eax
1  mov al,
1  call WriteChar
1  pop eax
```

mGotoXY

The mGotoXY macro sets the console cursor position by calling the Gotoxy library procedure.

```assembly
m_Gotoxy MACRO X:REQ, Y:REQ
  push edx
  mov  dh,Y
  mov  dl,X
  call Gotoxy
  pop  edx
ENDM
```

The REQ next to X and Y identifies them as required parameters.
mWriteStr Macro (1 of 2)

Provides a convenient way to display a string, by passing the string name as an argument.

```
mWriteStr MACRO buffer
    push edx
    mov edx,OFFSET buffer
    call WriteString
    pop edx
ENDM
```

```
data
str1 BYTE "Welcome!",0
.code
mWriteStr str1
```

mWriteStr Macro (2 of 2)

The expanded code shows how the str1 argument replaced the parameter named buffer:

```
mWriteStr MACRO buffer
    push edx
    mov edx,OFFSET buffer
    call WriteString
    pop edx
ENDM
```

```
1 push edx
1 mov edx,OFFSET str1
1 call WriteString
1 pop edx
```
mWrite

The mWrite macro writes a string literal to standard output. It is a good example of a macro that contains both code and data.

```asm
mWrite MACRO text
    LOCAL string
    .data ;; data segment
    string BYTE text,0 ;; define local string
    .code ;; code segment
    push edx
    mov edx,OFFSET string
    call Writestring
    pop edx
ENDM
```

The LOCAL directive prevents string from becoming a global label.

Nested Macros

- Nested macro: a macro invoked by another macro.

```asm
mWriteLn MACRO text
    mWrite text
    call Crlf
ENDM

mWriteLn "My Sample Macro Program"
```

nesting level
What I won’t cover in this course and why:
• Structure (easy for self-study)
• Conditional-Assembly Directives (easily confused with “dot directives” (e.g., .IF) for beginners)

• Chapter 9 (Strings and Arrays) are also not covered.

Except the following slides to let you get a feeling of what they are about.
MOVSB, MOVSW, and MOVSD

- The MOVSB, MOVSW, and MOVSD instructions copy data from the memory location pointed to by ESI to the memory location pointed to by EDI.

```assembly
.data
source DWORD 0FFFFFFFFh
target DWORD ?
.code
    mov esi,OFFSET source
    mov edi,OFFSET target
    MOVSD
```

LODSB, LODSW, and LODSD

- Use LODSB, LODSW, and LODSD if you want to process the data before moving it.

```assembly
.data
array 1,2,3,4,5,6,7,8,9
dest  9 DUP(?)
.code
    mov esi,OFFSET array
    mov edi,OFFSET dest
    mov ecx,LENGTHOF array
    cld
L1:  LODSB
    or al,30h
    STOSB
    loop L1
```
Using a Repeat Prefix

• REP (a repeat prefix) can be inserted just before MOVSB, MOVSW, or MOVSD.
• ECX controls the number of repetitions
• Example: Copy 20 doublewords from source to target

```assembly
.data
source DWORD 20 DUP(?)
target DWORD 20 DUP(?)
.code
cld ; direction = forward
mov ecx,LENGTHOF source ; set REP counter
mov esi,OFFSET source
mov edi,OFFSET target
REP MOVSD
```