CPSC-240 Computer Organization and Assembly Language

Chapter 11

Macros

Instructor: Yitsen Ku, Ph.D.
Department of Computer Science,
California State University, Fullerton, USA

CALIFORNIA STATE UNIVERSITY
FULLERTON™

Outline

- Single-Line Macros

- Multi-Line Macros

- Macro Definition

- Using a Macro

- Macro Example

- Debugging Macros

Macros

- An assembly language macro is a predefined set of instructions that can easily be inserted wherever needed.

- Once defined, the macro can be used as many times as necessary.

Single-Line Macros

Single-Line Macros

- Single-line macros are defined using the **%define** directive. The definitions work in a similar way to C/C++; so you can do things like:

  %define mulby4(x) shl x, 2

- And, then use the macro by entering:

  mulby4 (rax)

- in the source, which will multiply the contents to the **rax** register by 4 (via shifting two bits).

Multi-Line Macros

Multi-Line Macros

- Multi-line macros can include a varying number of lines (including one).

- The multi-line macros are more useful and the following sections will focus primarily on multi-line macros.

Macro Definition

Macro Definition

- Before using a multi-line macro, it must first be defined. The general format is as follows:

    %macro <name> <number of arguments>

    ; [body of macro]

    %endmacro

- The arguments can be referenced within the macro by **%<number>**, with **%1** being the first argument, and **%2** the second argument, and so forth.

- In order to use labels, the labels within the macro must be prefixing the label name with a **%%**.

Macro Definition

- This will ensure that calling the same macro multiple times will use a different label each time.

- For example, a macro definition for the absolute value function would be as follows:

%macro abs 1

    cmp %1, 0

    jge %%done

    neg %1

%%done:

%endmacro

Using a Macro

- Before using a multi-line macro, it must first be defined. The general format is as follows:

  %macro <name> <number of arguments>

  ; [body of macro]

  %endmacro

- The arguments can be referenced within the macro by **%<number>**, with **%1** being the first argument, and **%2** the second argument, and so forth.

- In order to use labels, the labels within the macro must be prefixing the label name with a **%%**.

Using a Macro

- In order to use or "invoke" a macro, it must be placed in the code segment and referred to by name with the appropriate number of arguments.

- Given a data declaration as follows:

  qVar dq 4

- Then, to invoke the "abs" macro (twice):

  mov eax, -3

  abs eax

  abs qword [qVar]

Using a Macro

- The list file will display the code as follows (for the first invocation):

27 00000000 B8FDFFFFFF   mov eax, -3

28   abs eax

29 00000005 3D00000000 <1> cmp %1, 0

30 0000000A 7D02   <1> jge %%done

31 0000000C F7D8   <1> neg %1

32   <1> %%done:

Using a Macro

- The macro will be copied from the definition into the code, with the appropriate arguments replaced in the body of the macro, *each* time it is used.

- The **<1>** indicates code copied from a macro definition. In both cases, the **%1** argument was replaced with the given argument; **eax** in this example.

- Macros use more memory, but do not require overhead for transfer of control (like functions).

Macro Example

Macro Example

```
; Example Program to demonstrate a simple macro
;
*********************************************
**

; Define the macro
; called with three arguments:
; aver <lst>, <len>, <ave>
%macro    aver 3
    mov    eax, 0
    mov    ecx, dword [%2]           ; length
    mov    r12, 0
    lea    rbx, [%1]
```

Macro Example

```
%%sumLoop:
    add     eax, dword [rbx+r12*4]       ; get list[n]
    inc     r12
    loop    %%sumLoop
    cdq
    idiv    dword [%2]
    mov     dword [%3], eax
%endmacro
```

Macro Example

```
; ***********************************************
;
; Data declarations
section .data
; -----
; Define constants
EXIT_SUCCESS        equ        0        ; success code
SYS_exit            equ        60       ; code for terminate
```

Macro Example

; Define Data.

section .data

list1   dd   4, 5, 2, -3, 1

len1   dd   5

ave1   dd   0

list2   dd   2, 6, 3, -2, 1, 8, 19

len2   dd   7

ave2   dd   0

; **

;

Macro Example

```
section .text
global _start
_start:
; -----
; Use the macro in the program
    aver    list1, len1, ave1              ; 1st, data set 1
    aver    list2, len2, ave2              ; 2nd, data set 2
; -----
; Done, terminate program.
last:
    mov     rax, SYS_exit                  ; exit
    mov     rdi, EXIT_SUCCESS              ; success
    syscall
```

Debugging Macros

- The code for a macro will not be displayed in the debugger source window.

- In order to see the macro code, display the machine code window (**View → Machine Code Window**).

- In the window, the machine code for the instructions are displayed. The step and next instructions will execute the entire macro.

- In order to execute the macro instructions, the **stepi** and **nexti** commands must be used.

End of Chapter 11