CPSC-240 Computer Organization and Assembly Language

Chapter 4

Program Format

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

CALIFORNIA STATE UNIVERSITY
FULLERTON™

Outline

- Comments
- Numeric Values
- Defining Constants
- Data Section
- BSS Section
- Text Section
- Example Program

Comments

Comments

- The semicolon (**;**) is used to note program comments. Comments (using the **;**) may be placed anywhere, including after an instruction.
- Any characters after the **;** are ignored by the assembler.
- This can be used to explain steps taken in the code or to comment out sections of code.

Numeric Values

Numeric Values

- Number values may be specified in decimal, hex, or octal. When specifying hex, or base-16 values, they are preceded with a **0x**. For example, to specify 127 as hex, it would be **0x7f**.
- When specifying octal, or-base-8 values, they are followed by a **q**. For example, to specify 511 as octal, it would be **777q**.
- The default radix (base) is decimal, so no special notation is required for decimal (base-10) numbers.

Defining Constants

Defining Constants

- Constants are defined with **equ**. The general format is:

<name> equ <value>

- The value of a constant cannot be changed during program execution.
- The constants are substituted for their defined values during the assembly process. As such, a constant is not assigned a memory location. This makes the constant more flexible since it is not assigned a specific type/size (byte, word, double-word, etc.). The values are subject to the range limitations of the intended use. For example, the following constant,

SIZE equ 10000

- could be used as a word or a double-word, but not a byte.

Data Section

Data Section

- The initialized data must be declared in the "section .data" section. There must be a space after the word 'section'. All initialized variables and constants are placed in this section. Variable names must start with a letter, followed by letters or numbers, including some special characters (such as the underscore, "_").
- Variable definitions must include the name, the data type, and the initial value for the variable. The general format is:

<variableName> <dataType> <initialValue>

Data Section

Declaration	
db	8-bit variable(s)
dw	16-bit variable(s)
dd	32-bit variable(s)
dq	64-bit variable(s)
ddq	128-bit variable(s)→integer
dt	128-bit variable(s)→float

Some simple examples include

```
section .data
    bVar    db    10                ; byte variable
    cVar    db    "H"               ; single character
    strng   db    "Hello World"     ; string
    wVar    dw    5000              ; 16-bit variable
    dVar    dd    50000             ; 32-bit variable
    arr     dd    100, 200, 300     ; 3 element array
    flt1    dd    3.14159           ; 32-bit float
    qVar    dq    1000000000        ; 64-bit variable
```

BSS Section

BSS Section

- Uninitialized data is declared in the "section .bss" section. There must be a space after the word 'section'. All uninitialized variables are declared in this section.
- Variable names start with a letter followed by letters or numbers including some special characters (such as the underscore, "_"). Variable definitions must include the name, the data type, and the count.
- The general format is:

<variableName> <resType> <count>

Data Type

Declaration	
resb	8-bit variable(s)
resw	16-bit variable(s)
resd	32-bit variable(s)
resq	64-bit variable(s)
resdq	128-bit variable(s)

Some simple examples include

```
section .bss
    bArr    resb    10      ; 10 element byte array
    wArr    resw    50      ; 50 element word array
    dArr    resd    100     ; 100 element double array
    qArr    resq    200     ; 200 element quad array
```

Text Section

text Section

- The code is placed in the "section .text" section. There must be a space after the word 'section'. The instructions are specified one per line and each must be a valid instruction with the appropriate required operands.
- The text section will include some headers or labels that define the initial program entry point. For example, assuming a basic program using the standard system linker, the following declarations must be included.

```
global _start
_start:
```

Some simple examples include

```
section .bss
    bArr    resb    10      ; 10 element byte array
    wArr    resw    50      ; 50 element word array
    dArr    resd    100     ; 100 element double array
    qArr    resq    200     ; 200 element quad array
```

Example Program

Some simple examples include (1)

```
; Simple example demonstrating basic program format and layout.
; Ed Jorgensen
; July 18, 2014
;
;*************************************************************

; Some basic data declarations
section .data
; -----
; Define constants
EXIT_SUCCESS    equ    0          ; successful operation
SYS_exit        equ    60         ; call code for terminate
```

Some simple examples include (2)

```
; -----
; Byte (8-bit) variable declarations
bVar1       db      17
bVar2       db      9
bResult     db      0
; -----
; Word (16-bit) variable declarations
wVar1       dw      17000
wVar2       dw      9000
wResult     dw      0
; -----
; Double-word (32-bit) variable declarations
dVar1       dd      17000000
dVar2       dd      9000000
dResult     dd      0
```

Some simple examples include (3)

```
; -----
; quadword (64-bit) variable declarations
qVar1      dq        170000000
qVar2      dq        90000000
qResult    dq        0
; ************************************************************
; Code Section
section    .text
global _start
_start:
; Performs a series of very basic addition operations
; to demonstrate basic program format.
```

Some simple examples include (4)

```
; ----------
; Byte example
; bResult = bVar1 + bVar2
mov     al, byte [bVar1]
add     al, byte [bVar2]
mov     byte [bResult], al
; ----------
; Word example
; wResult = wVar1 + wVar2
mov     ax, word [wVar1]
add     ax, word [wVar2]
mov     word [wResult], ax
```

Some simple examples include (5)

```
; ----------
; Double-word example
; dResult = dVar1 + dVar2
mov     eax, dword [dVar1]
add     eax, dword [dVar2]
mov     dword [dResult], eax
; ----------
; Quadword example
; qResult = qVar1 + qVar2
mov     rax, qword [qVar1]
add     rax, qword [qVar2]
mov     qword [qResult], rax
```

Some simple examples include (6)

```
;
*************************************************************
; Done, terminate program.
last:
mov     rax, SYS_exit           ; Call code for exit
mov     rdi, EXIT_SUCCESS       ; Exit program with success
syscall
```

Thanks