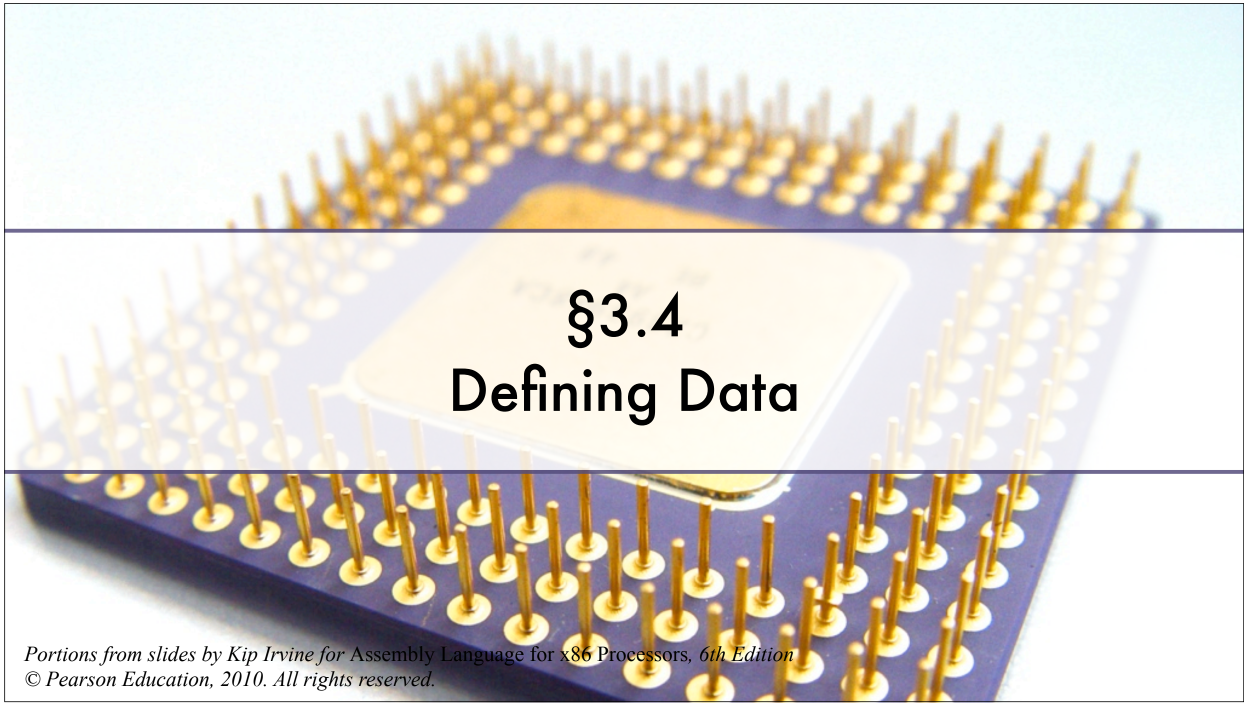


§3.4 Defining Data

Intrinsic Data Types

- BYTE, SBYTE – 8-bit unsigned, signed integer
- WORD, SWORD – 16-bit unsigned, signed integer
- DWORD, SDWORD – 32-bit unsigned, signed integer
- QWORD – 64-bit integer
- REAL4 – 4-byte IEEE short real (floating point)
- REAL8 – 8-byte IEEE long real (floating point)
- REAL10 – 10-byte IEEE extended real (floating point)

Data Definition Statement

- A data definition statement sets aside storage in memory for a variable.
- May optionally assign a name (label) to the data
- Syntax:

  [*name*] *directive initializer* [,*initializer*] . . .

  ```
  value1 BYTE 10
  ```

- All initializers become binary data in memory

Defining BYTE & SBYTE Data

Each of the following defines a single byte of storage:

```
value1 BYTE 'A'             ; character constant
value2 BYTE 0               ; smallest unsigned byte
value3 BYTE 255             ; largest unsigned byte
value4 SBYTE -128           ; smallest signed byte
value5 SBYTE +127           ; largest signed byte
value6 BYTE ?               ; uninitialized byte
```

MASM does not prevent you from initializing a BYTE with a negative value, but it's considered poor style.

If you declare a SBYTE variable, the debugger will display its value in decimal with a leading sign.

Defining Byte Arrays

Examples that use multiple initializers:

```
list1 BYTE 10,20,30,40
list2 BYTE 10,20,30,40
      BYTE 50,60,70,80
      BYTE 81,82,83,84
list3 BYTE ?,32,41h,00100010b
list4 BYTE 0Ah,20h,‘A’,22h
```

Defining Strings (1 of 2)

- A string is implemented as an array of bytes
  - For convenience, it is usually enclosed in quotation marks
  - It often will be null-terminated
- Examples:

```
str1 BYTE "Enter your name",0
str2 BYTE 'Error: halting program',0
str3 BYTE 'A','E','I','O','U'
greeting  BYTE "Welcome to the Encryption Demo program "
          BYTE "created by Kip Irvine.",0
```

Defining Strings (2 of 2)

- To continue a single string across multiple lines, end each line with a comma:

```
menu BYTE "Checking Account",0dh,0ah,0dh,0ah,
    "1. Create a new account",0dh,0ah,
    "2. Open an existing account",0dh,0ah,
    "3. Credit the account",0dh,0ah,
    "4. Debit the account",0dh,0ah,
    "5. Exit",0ah,0ah,
    "Choice> ",0
```

Using the DUP Operator

- Use DUP to allocate (create space for) an array or string. Syntax: *counter* DUP (*argument*)

Counter and *argument* must be constants or constant expressions

```
var1 BYTE 20 DUP(0)             ; 20 bytes, all equal to zero
var2 BYTE 20 DUP(?)             ; 20 bytes, uninitialized
var3 BYTE 4 DUP("STACK")       ; 20 bytes: "STACKSTACKSTACKSTACK"
var4 BYTE 10,3 DUP(0),20        ; 5 bytes
```

Defining WORD & SWORD Data

- Define storage for 16-bit integers
  - single value or array (multiple values)

```
word1  WORD  65535           ; largest unsigned value
word2  SWORD -32768          ; smallest signed value
word3  WORD  ?               ; uninitialized, unsigned
word4  WORD  "AB"            ; double characters
myList WORD  1,2,3,4,5       ; array of words
array  WORD  5 DUP(?)        ; uninitialized array
```

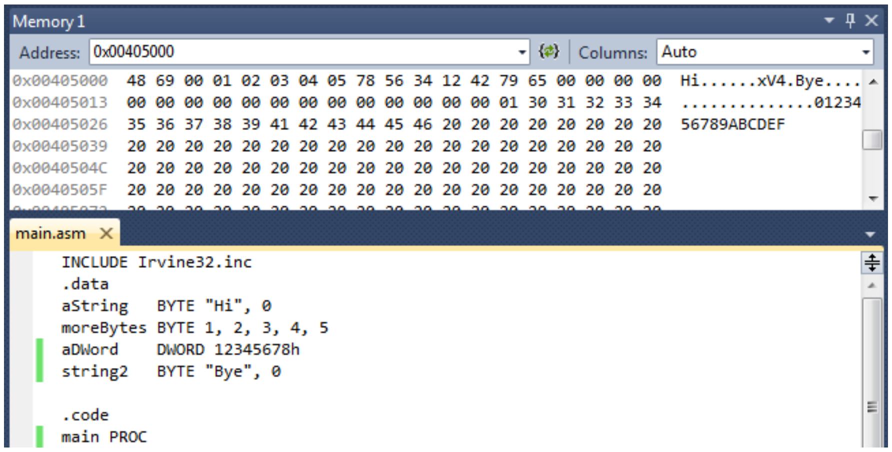
Defining DWORD & SDWORD Data

Storage definitions for signed and unsigned 32-bit integers:

```
val1 DWORD  12345678h         ; unsigned
val2 SDWORD -2147483648       ; signed
val3 DWORD  20 DUP(?)         ; unsigned array
val4 SDWORD -3,-2,-1,0,1      ; signed array
```

VS Memory Window

- Recall from Lab 2 how to use the Memory Window in the Visual Studio debugger



```
0x00405000  48 69 00 01 02 03 04 05 78 56 34 12 42 79 65 00 00 00 00  Hi......xV4.Bye....
0x00405013  00 00 00 00 00 00 00 00 00 00 00 00 00 01 30 31 32 33 34  .............01234
0x00405026  35 36 37 38 39 41 42 43 44 45 46 20 20 20 20 20 20 20 20  56789ABCDEF
0x00405039  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0x0040504C  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0x0040505F  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
```

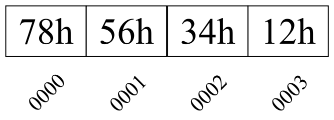
```
INCLUDE Irvine32.inc
.data
aString   BYTE "Hi", 0
moreBytes BYTE 1, 2, 3, 4, 5
aDWord    DWORD 12345678h
string2   BYTE "Bye", 0

.code
main PROC
```

Little Endian Order

- General purpose registers store 32-bit values; memory stores bytes
- For all data types larger than a byte:
  - The ***l***east significant byte is stored in the ***l***owest memory address
  - This is called ***l****ittle endian* byte ordering
- Example:

```
val1 DWORD 12345678h
```



Big Endian Order

- **x86 processors use little endian byte ordering**, but...
- Some other processors use *big endian,* where 12345678h would be stored as

12h	34h	56h	78h
0000	0001	0002	0003

- Big endian is also called *network byte order*
  - The Internet Protocol (IP) and many other protocols transfer 16- and 32-bit values in big-endian order, i.e., the most significant byte is transmitted first

Using Data in Memory (1 of 2)

- You know two versions of the mov instruction:
  - `mov` *register*, *immediate* — `mov eax, 5`
  - `mov` *register*, *register* — `mov eax, ebx`
- You can also move data to and from memory:

```
.data
myVar  DWORD  135
```

  - `mov` *register*, *memory* — `mov eax, myVar`
  - `mov` *memory*, *register* — `mov myVar, ebx`
  - `mov` *memory*, *immediate* — `mov myVar, 9876`
  - `mov` *immediate*, *memory* — **Q.** Is this possible?

Using Data in Memory (2 of 2)

```
TITLE Add and Subtract, Version 2            (AddSub2.asm)
; This program adds and subtracts 32-bit unsigned
; integers and stores the sum in a variable.
INCLUDE Irvine32.inc
.data
val1 DWORD 10000h
val2 DWORD 40000h
val3 DWORD 20000h
finalVal DWORD ?
.code
main PROC
    mov eax,val1                ; start with 10000h - load from memory into register
    add eax,val2                ; add 40000h - load operand from memory
    sub eax,val3                ; subtract 20000h - load operand from memory
    mov finalVal,eax            ; store the result (30000h) - store result operand
    call DumpRegs               ; display the registers
    exit
main ENDP
END main
```

Declaring Unitialized Data

- Use the .data? directive to declare an uninitialized data segment:

  `.data?`

- Within the segment, declare variables with "?" initializers:

  `smallArray DWORD 10 DUP(?)`

Advantage: the program's EXE file size is reduced.

Activity 6 Part I