Kingdom of Saudi Arabia
Royal Commission at Yanbu
University College – Yanbu
Department of ACS & AIT
Yanbu Al-Sinaiyah

المملكة العربية السعودية
الهيئة الملكية ينبع
الكلية الجامعية-ينبع
ينبع الصناعية

2ND Semester 2007-08

CS-203 HANDOUT (HO - 9)

Manipulating Variables and Arrays in 8086 Assembly

9.1 Declaring Variables in 8086 Assembly

When you declare an integer variable by assigning a label to a data allocation directive, the assembler allocates memory space for the integer. The variable's name becomes a label for the memory space. A variable is just a memory location in actual practice. Its for the programmer's ease that a variable name is given.

The following directives indicate the integer's size and value range:

Example

```
Data1  DB  87H
Data2  DW 88F8H
```

(Only the directives, which are underlined, are included in your course)

Directive	Description of Initializers
BYTE, **DB** (byte)	Allocates unsigned numbers from 0 to 255.
SBYTE (signed byte)	Allocates signed numbers from –128 to +127.
WORD, DW (word = 2 bytes)	Allocates unsigned numbers from 0 to 65,535 (64K).
SWORD (signed word)	Allocates signed numbers from –32,768 to +32,767.
DWORD, DD (doubleword = 4 bytes),	Allocates unsigned numbers from 0 to 4,294,967,295 (4 megabytes).
SDWORD (signed doubleword)	Allocates signed numbers from –2,147,483,648 to +2,147,483,647.
FWORD, DF (farword = 6 bytes)	Allocates 6-byte (48-bit) integers. These values are normally used only as pointer variables on the 80386/486

	processors.
QWORD, DQ (quadword = 8 bytes)	Allocates 8-byte integers used with 8087-family coprocessor instructions.
TBYTE, DT (10 bytes),	Allocates 10-byte (80-bit) integers if the initializer has a radix specifying the base of the number.

The amount, which these data types take, are given in the table below.

Data Type	Bytes
BYTE, SBYTE	1
WORD, SWORD	2
DWORD, SDWORD	4
FWORD	6
QWORD	8
TBYTE	10

9.1.1 Constants:

In an assembly language program, constants are defined through the use of the **EQU** directive.
These are the main points in using constant definitions:

- The EQU directive is used to assign a name to a constant.
- Use of constant names makes an assembly language easier to understand.
- No memory is allocated for a constant.
- Reference is made to a constant through immediate addressing mode.

Example:

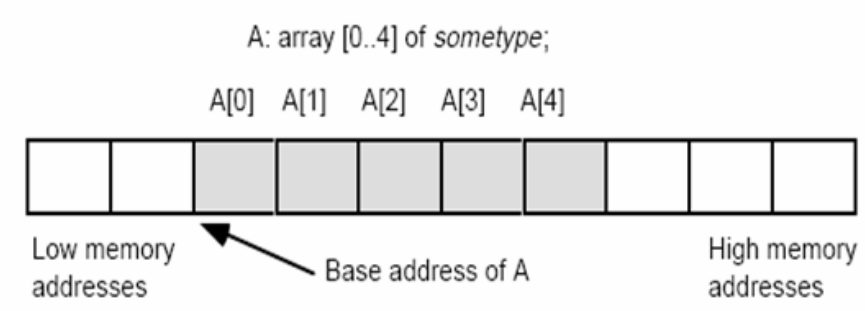
The following shows examples on the declaration of constants.

Constant Declarations

```
...
.data
        LF      EQU     0AH
        PROMPT  EQU     First Message
        MSG     DB      Ali Al-johani
.code
        ...
        MOV DL,LF ; LF means 0AH
        MOV DL, LF
        ...
```

9.2 Arrays in 8086

Arrays are probably the most commonly used data type. Its a data type whose members (elements) are all the same type. It consists of list of elements all of same data type. Different elements of array can be accessed through the addresses in the memory. For example, If i holds the starting address of first element of array then A[i] chooses the first element from array A. A[I+1] refers to the next element respectively.



The *base address* of an array is the address of the first element on the array and always appears in the lowest memory location. The second array element directly follows the first in memory; the third element follows the second, etc.

The following example shows how to declare an array of bytes (b_array) and an array of words (w_array):

Array declaration

```
b_array BYTE  1, 2, 3, 4
w_array WORD FFFFh, 789Ah, BCDEh
```

The following example shows legal multiple-line array declaration:

Multiple Line Byte Declaration

```
big     BYTE    21, 22, 23, 24, 25,26, 27, 28,
                10, 20, 30
```

An array may span more than one logical line, such as the array var1 in the example below, although a separate type declaration is needed in each logical line:

Multiple line array declaration

```
var1    BYTE 10, 20, 30
        BYTE 40, 50, 60
        BYTE 70, 80, 90
```

9.2.1 The DUP Operator:

The DUP operator is very often used in the declaration of arrays. This operator works with any of the data allocation directives.

In the syntax:

count DUP (initialvalue [[, initialvalue]]...)

The count value sets the number of times to repeat all values within parentheses. The initial value can be an integer, character constant, or another DUP operator, and must always appear within parentheses.
For example, the statement allocates the integer value 1 five times for a total of 5 bytes.:

The DUP Operator

```
barray  BYTE    5       DUP     (1)
is similar to

barray BYTE 1, 1, 1, 1, 1
```

The following examples show various ways for allocating data elements with the DUP operator:

Different uses of the DUP Operator

```
array    DWORD  10  DUP (1)            ;10 doublewords initialized to 1
buffer   BYTE   256 DUP (?)                          ;256-byte buffer
```

Another example

BigArray word 256 dup (0,1,2,3)

This array has 1024 elements, not 256. The n dup (xxxx) operand tells MASM to duplicate xxxx *n* times, not create an array with *n* elements. If xxxx consists of a single item, then the dup operator will create an *n* element array. Since there are four items in the parentheses above, the dup operator creates 256*4 or 1024 items in the array. The values in the array will initially be 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 ...

Example Program 1

```
.TITLE A program to calculate sum of five numbers
.MODEL SMALL
.STACK 200
.DATA
            DATAIN  DB 5, 6, 7, 8, 9
            SUM DB ?
.CODE
.STARTUP
            MOV CX, 05H
            MOV CX, OFFSET DATAIN
            MOV AL, 0
AGAIN:      ADD AL, [BX]
            INC BX
            DEC CX
            JNZ AGAIN
            MOV SUM, AL

.EXIT
```

END

Example Program 2

```
.TITLE A program that finds the highest number in a list of numbers and stores in DL
register
.MODEL SMALL
.STACK 200
.DATA
             DATA1 DB 2, 6, 1, 3, 5
.CODE
.STARTUP
            MOV CX, 5
            MOV BX, OFFSET DATA1
            SUB AL, AL
AGAIN:      CMP AL, [BX]
            JA NEXT
            MOV AL, [BX]
NEXT:       INC AX
            LOOP AGAIN
            MOV DL, AL
            .EXIT
            END
```

Example Program 3

```
.TITLE A program that add elements of two arrays and store it into a new array
.MODEL SMALL
.STACK 200
.DATA
                NUM1  DB  2, 6, 1, 3, 5
                NUM2  DB  3, 4, 6, 2, 4
                SUM    DB  5  DUP(0)
.CODE
.STARTUP
                MOV BX, OFFSET NUM1
                MOV SI, OFFSET NUM2
                MOV DI, OFFSET SUM
                MOV CX, 5
NEXT:           MOV AX, [BX]
                ADD AX, [SI]
                MOV [DI], AX
                INC SI
                INC BX
                INC DI
                LOOP NEXT

.EXIT
END
```

Example Test Questions

Exercise Program 1

Write a program which counts the number of zeros in a byte stored in AL register.

Exercise Program 2

Write a program, which searches for a value 3 in a list of given five numbers, suppose, 4, 3, 1, 9, 6
If the program can find the number present in the list then it subtracts 2 from it and saves in BL register, otherwise it adds 3 in it and stores in BL register.

Pseudocode

If 3 present in (4, 3, 1, 9, 6) then
BL = 3 – 2
Else
BL = 3 +3