Lab: 06

Department of Computer Science

Iqra University Islamabad

Computer Organization and Assembly Language

Maqsood Ahmed

ID: 38186

3.4 Lab Work: Multiple Initializers, Defining Strings, and the DUP Operator

This section explains how to use multiple initializers and the DUP operator to define arrays and strings in assembly language.

Multiple Initializers and DUP Operator Example:

```
assembly
TITLE Multiple Initializers        (MultipleInitializers.asm)

; Examples showing multiple initializers and the DUP operator

.686
.MODEL flat, stdcall
.STACK

INCLUDE Irvine32.inc

; ----------------- Byte Values ----------------
.data
list1   BYTE  10, 32, 41h, 00100010b
list2   BYTE  0Ah, 20h, 'A', 22h
array1  BYTE  8 DUP(0)      ; 8 bytes initialized to 0

greeting BYTE "Good afternoon", 0

; ----------------- Word Values ----------------
myList  WORD   1,2,3,4      ; array of words

; --------------- DoubleWord Values --------------
array2  DWORD   4 DUP(01234567h)

.code
main PROC
; No instructions to execute
       exit
main ENDP
END main
```

Memory Window Observations:

1. **Virtual Address of `myList`**: This can be found using the Watch window in the debugger.
2. **Virtual Address of `array2`**: Similar to `myList`.
3. **Byte Allocation:**
   - `myList`: Contains 4 words, each 2 bytes = 8 bytes.
   - `array2`: Contains 4 double words, each 4 bytes = 16 bytes.
4. **Byte Value Observations:**
   - Byte value at virtual address `00404018`: Can be observed directly from the debugger.
   - Byte value at virtual address `00404032`: Similarly observed.

Answers for the Questions

1. **Virtual Address of `myList`:** This can be checked in the Memory window.
2. **Virtual Address of `array2`:** This can be checked in the Memory window.
3. **Bytes allocated for `myList`:** 8 bytes (4 words * 2 bytes each).
4. **Bytes allocated for `array2`:** 16 bytes (4 double words * 4 bytes each).
5. **Byte value at virtual address `00404018`:** This will be one of the bytes in the memory, observed in the debugger.
6. **Byte value at virtual address `00404032`:** Another byte value, observed in the debugger.

Little Endian Order:

- The values are stored in little endian order, meaning the least significant byte is stored first.

Changing Display Format:

- For `myList`, change to Short Hex to view WORD values.
- For `array2`, change to Long Hex to view DWORD values.

Example Using Long Hex Format for `array2`

- Virtual address and four double word values:
  - `00404020`: 67 45 23 01
  - `00404024`: 67 45 23 01
  - `00404028`: 67 45 23 01
  - `0040402C`: 67 45 23 01

3.6 Lab Work: Data Related Operators

Operators.asm Example:

```
TITLE Operators (File: Operators.asm)
; Demonstration of TYPE, LENGTHOF, SIZEOF, OFFSET, and PTR operators

.686
.MODEL flat, stdcall
.STACK

INCLUDE Irvine32.inc

.data
byte1    BYTE  10,20,30,40
array1   WORD  30 DUP(?),0,0
array2   WORD  5 DUP(3 DUP(?))
array3   DWORD 01234567h,2,3,4
digitStr BYTE  '12345678',0
```

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

.code
main PROC
       ; Demonstrating TYPE operator
       mov al, TYPE byte1
       mov bl, TYPE array1
       mov cl, TYPE array3
       mov dl, TYPE digitStr

       ; Demonstrating LENGTHOF operator
       mov eax, LENGTHOF array1
       mov ebx, LENGTHOF array2
       mov ecx, LENGTHOF array3
       mov edx, LENGTHOF digitStr

       ; Demonstrating SIZEOF operator
       mov eax, SIZEOF array1
       mov ebx, SIZEOF array2
       mov ecx, SIZEOF array3
       mov edx, SIZEOF digitStr

       ; Demonstrating OFFSET operator
       mov eax, OFFSET byte1
       mov ebx, OFFSET array1
       mov ecx, OFFSET array2
       mov edx, OFFSET array3
       mov esi, OFFSET digitStr
       mov edi, OFFSET myArray

       ; Demonstrating PTR operator
       mov al, BYTE PTR array3
       mov bx, WORD PTR array3
       mov cx, WORD PTR myArray
       mov edx, DWORD PTR myArray

       exit
main ENDP
END main
```

Debugging Observations:

1. **TYPE Operator:**
   - `al` = 1 (BYTE)
   - `bl` = 2 (WORD)
   - `cl` = 4 (DWORD)
   - `dl` = 1 (BYTE)
2. **LENGTHOF Operator:**
   - `eax` = 32 (for `array1`)

- `ebx` = 15 (for `array2`)
- `ecx` = 4 (for `array3`)
- `edx` = 9 (for `digitStr`)

3. **SIZEOF Operator:**
   - `eax` = 64 (bytes for `array1`)
   - `ebx` = 30 (bytes for `array2`)
   - `ecx` = 16 (bytes for `array3`)
   - `edx` = 9 (bytes for `digitStr`)
4. **OFFSET Operator:**
   - `eax`, `ebx`, `ecx`, `edx`, `esi`, `edi` - Virtual addresses for each variable, observed in the debugger.
5. **PTR Operator:**
   - `al` - First byte of `array3`
   - `bx` - First word of `array3`
   - `cx` - First word of `myArray`
   - `edx` - First dword of `myArray`