National University of Computer and Emerging Sciences

Laboratory Manual

for

Computer Organization and Assembly Language

Course Instructors

Lab Instructor(s)

Section

Semester

Department of Computer Science

COAL Lab 6 Manual

Objectives:

- Indirect Addressing
  - Indirect operands
  - Indexed operands
- Pointers
- Using DumpMEM
- Problems & Assignments

6.1 Indirect Addressing

Protected Mode	Real Mode

1. Indirect operand can be any 32-bit general-purpose register (**EAX**, **EBX**, **ECX**, **EDX**, **ESI**, **EDI**, **EBP**, and **ESP**) surrounded by brackets.
2. Example
`include irvine32.inc`
`.data`
`byteVal BYTE 10h`
`.code`
`Main proc`
`mov ESI,OFFSET byteVal`
`mov AL,[ESI] ;AL = 10h`
	1. Indirect operand can be any 16-bit general-purpose register (**AX**, **BX**, **CX**, **DX**, **SI**, **DI**, **BP**, and **SP**) surrounded by brackets.
	2. Example
	`include irvine16.inc`
	`.data`
	`byteVal BYTE 10h`
	`.code`
	`Main proc`
	`startup`
	`mov SI,OFFSET byteVal`
	`mov AL,[SI] ;AL = 10h`

The size of an operand may not be evident from the context of an instruction. The following instruction causes the assembler to generate an "operand must have size" error message:

```
inc [esi]   ; error: operand must have size
```

The PTR operator confirms the operand size:

```
inc BYTE PTR [esi]
```

6.2 Arrays Manipulation

Using Indirect Operands	Using Indexed Operands	

.data
 	
arrayW WORD
1000h,2000h,3000h
 	
.code
 	
mov esi,OFFSET arrayW
mov ax,[esi]
; AX = 1000h
add esi,2
mov ax,[esi]
; AX = 2000h
add esi,1
mov ax,[esi]
; AX = ???
	.code
	
	mov esi,0
	mov ax,[arrayW+esi]
	; AX = 1000h
	add esi,2
	mov ax,[arrayW+esi]
	; AX = 2000h
	add esi,1
	mov ax,[arrayW+esi]
		Using Displacement
		
		.code
		
		mov esi,OFFSET arrayW
		mov ax,[esi]
		; AX = 1000h
		mov ax,[esi+2]
		; AX = 2000h
		mov ax,[esi+4]
		; AX = 3000h

6.3 DumpMEM

The ***DumpMem*** procedure writes a range of memory to the console window in hexadecimal. Pass it the starting address in ESI, the number of units in ECX, and the unit size in EBX (1= byte, 2= word, 4=doubleword). The following sample call displays an array of 11 doublewords in hexadecimal:

```
.data
array DWORD 1,2,3,4,5,6,7,8,9,0Ah,0Bh

.code
main PROC

mov esi,OFFSET array        ; starting OFFSET
mov ecx,LENGTHOF array      ; number of units
mov ebx,TYPE array          ; doubleword format
call DumpMem
```

The following output is produced:

```
00000001  00000002  00000003  00000004  00000005  00000006  00000007
00000008 00000009 0000000A 0000000B
```

Problem(s) / Assignment(s)

Discussion & Practice **Estimated completion time: 1 hr, 30 mins**

Problem 6.1: *Fill the requested Register* **Estimated completion time:20 mins**

```
myBytes BYTE 10h,20h,30h,40h
myWords WORD 1Ah,2Bh,3Ch,4Dh,5Eh
myDoubles DWORD 2,4,6,8,5
myPointer DWORD myDoubles
```

Instructions	Registers
`mov esi, OFFSET myBytes`	
`mov al, [esi]`	**a. AL =**
`mov al, [esi+3]`	**b. AL =**
`mov esi, OFFSET myWords + 2`	
`mov ax, [esi]`	**c. AX =**
`mov edi, 8`	
`mov edx, [myDoubles + edi]`	**d. EDX =**
`mov edx, myDoubles[edi]`	**e. EDX =**
`mov ebx, myPointer`	
`mov eax, [ebx+4]`	**f. EAX =**
`mov eax,DWORD PTR myWords`	**g. EAX =**
`mov esi,myPointer`	
`mov ax,[esi+2]`	**h. AX =**
`mov ax,[esi+6]`	**i. AX =**
`mov ax,[esi-4]`	**j. AX =**

Problem 6.2 (a): *Fibonacci number generation* **Estimated completion time:15 mins**

Write a program that generate the first six Fibonacci number sequence (1, 1, 2, 3, 5, 8). Use an array named Fibonacci of word type. You may initialize the first two places of array with 1's and next four places with 0. Use the following rule and save results in the same array by **using indirect operand addressing**.

The rule to generate sequence is $F_n = F_{n-1} + F_{n-2}$. Call DumpMEM to display Fibonacci sequence.

(b):

Repeat a part using indexed operands without displacement.

Problem 6.3: *Array Manipulation* **Estimated completion time:20 mins**

Let us have an array,

A_array BYTE 1FH, 63H, 0ABH, 88H

1. Compute the sum for higher and lower bytes of each value (by ignoring carry bit).
2. Putting the result in another 4 elements DWORD size array namely B_array.
3. Use indexed operands addressing with displacement.

Display A_array and B_array on console.

You are done with your exercise(s), make your submission ☺