Laboratory Manual

for

Computer Organization and Assembly Language

|  |  |
| --- | --- |
| Course Instructors |  |
| Lab Instructor(s) |  |
| Section |  |
| Semester |  |

**Department of Computer Science**

# COAL Lab 6Manual

|  |
| --- |
| 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-purposeregister (**EAX**, **EBX**, **ECX**, **EDX**, **ESI**, **EDI**, **EBP**, and **ESP**) surrounded by brackets. 2. Example   **include irvine32.inc**  **.data**  **byteVal BYTE 10h**  **.code**  **Main proc**  **movESI,OFFSET byteVal**  **movAL,[ESI] ;AL = 10h** | 1. Indirect operand can be any 16-bit general-purposeregister (**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**  **movSI,OFFSET byteVal**  **movAL,[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]**

**6.2Arrays Manipulation**

|  |  |  |
| --- | --- | --- |
| **Using Indirect Operands** | **Using Indexed Operands** | |
| **.data**  **arrayW WORD 1000h,2000h,3000h**  **.code**  **movesi,OFFSET arrayW**  **movax,[esi] ; AX = 1000h**  **addesi,2**  **movax,[esi] ; AX = 2000h**  **addesi,1**  **movax,[esi]**  **; AX = ???** | **.code**  **movesi,0**  **movax,[arrayW+esi] ; AX = 1000h**  **addesi,2**  **movax,[arrayW+esi] ; AX = 2000h**  **addesi,1**  **movax,[arrayW+esi]** | ***Using Displacement***  **.code**  **movesi,OFFSET arrayW**  **movax,[esi]**  **; AX = 1000h**  **movax,[esi+2]**  **; AX = 2000h**  **movax,[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**

**movesi,OFFSET array ; starting OFFSET**

**movecx,LENGTHOF array ; number of units**

**movebx,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*  **myBytes BYTE 10h,20h,30h,40h**  **myWords WORD 8Ah,3Bh,72h,44h,66h**  **myDoubles DWORD 1,2,3,4,5**  **myPointer DWORD myDoubles** | **Estimated completion time:20 mins** |

|  |  |
| --- | --- |
| **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*  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 Fn= Fn-1+Fn-2. Call DumpMEM to display Fibonacci sequence.  **(b):**  Repeat a part using indexed operands without displacement. | **Estimated completion time:15 mins** |

|  |  |
| --- | --- |
| **Problem 6.3:** *Array Manipulation*  Let us have an array,  A\_array WORD 10FFH, 6323H, 0ABCDH, 828H   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. | **Estimated completion time:20 mins** |

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

***Submission Guidelines:***

* Attach your .asm files of all programs and examples by your slate account
* Once you have attached all the files, click on the ‘Submit’ button.