Lab: 10

Department of Computer Science

Iqra University Islamabad

Computer Organization and Assembly Language

Maqsood Ahmed

ID: 38186

Review Questions

1. **Destination Register Values (in hexadecimal):**
   - **a.** `mov ax, var1` **→ Invalid (addressing mode requires memory dereference)**
   - **b.** `movzx ax, var1` **→ 0xFC**
   - **c.** `movsx eax, var1` **→ 0xFFFFFFFC**
   - **d.** `mov ax, var2[2]` **→ 0x2000**
   - **e.** `mov bx, var3` **→ 0x0001**
   - **f.** `mov edx, [var3+4]` **→ 0x00000002**
   - **g.** `lea esi, var2` **→ 0x404002**
   - **h.** `mov al, [esi]` **→ 0x00 (assuming** `esi` **points to** `var2`**)**
   - **i.** `mov ax, [esi]` **→ 0x1000 (assuming** `esi` **points to** `var2`**)**
   - **j.** `mov eax, [esi]` **→ 0x00001000 (assuming** `esi` **points to** `var2`**)**
   - **k.** `inc [esi]` **→ Memory at** `esi` **incremented**

2. **Overflow flag with positive and negative integer addition:**
   - No
3. **NEG instruction setting the Overflow flag:**
   - Yes
4. **Both Sign and Zero flags set at the same time:**
   - No
5. **Any 16-bit general-purpose register for indirect addressing:**
   - No
6. **Any 32-bit general-purpose register for indirect addressing:**
   - Yes

Programming Exercises

1. **Program to Set and Clear Flags:**

```
TITLE Set and Clear Flags Example (SetClearFlags.asm)
.686
.MODEL flat, stdcall
.STACK 4096
INCLUDE Irvine32.inc

.code
main PROC
    ; Set and clear Carry flag
    clc             ; Clear Carry flag
    mov al, 0FFh
    add al, 1       ; Set Carry flag

    sub al, 1       ; Clear Carry flag
    ; Set and clear Zero and Sign flags
    mov al, 0
    sub al, 0       ; Set Zero flag

    mov al, 0
    sub al, 1       ; Set Sign flag

    ; Set and clear Overflow flag
    mov al, 7Fh
    add al, 1       ; Set Overflow flag

    mov al, 80h
    add al, 80h     ; Clear Overflow flag

    ; Set and clear both Carry and Overflow flags
    mov al, 7Fh
    add al, 81h     ; Set Carry and Overflow flags

    ; End of program
    exit
main ENDP
END main
```

2. **Fibonacci Sequence Program:**

```
TITLE Fibonacci Sequence (Fibonacci.asm)
.686
.MODEL flat, stdcall
.STACK 4096
INCLUDE Irvine32.inc

.data
fib DWORD 10 DUP(0)

.code
main PROC
    mov ecx, 8       ; Number of Fibonacci values to calculate
    mov eax, 1       ; First Fibonacci number
    mov ebx, 1       ; Second Fibonacci number
    mov [fib], eax
    mov [fib+4], ebx

L1:
    add eax, ebx
    mov [fib + ecx*4], eax
    xchg eax, ebx
    loop L1

    exit
main ENDP
END main
```

3. **Modify `SumArray.asm` to use Scale Factor:**

```
TITLE Summing an Array with Scale Factor (SumArrayScaled.asm)
.686
.MODEL flat, stdcall
.STACK 4096
INCLUDE Irvine32.inc
.data
intarray SWORD 5, 7, -3, 100, 0, -9, 10 DUP(-999)
sum      SWORD ?
.code
main PROC
    mov esi, 0
    mov ecx, LENGTHOF intarray
    mov ax, 0
L1:
    add ax, intarray[esi*2]
    inc esi
    loop L1
    mov sum, ax
    exit
main ENDP
END main
```

4. **Modify `CopyStr.asm` to Copy Characters in Reverse Order:**

```
TITLE Copy String in Reverse (CopyStrReverse.asm)
.686
.MODEL flat, stdcall
.STACK 4096
INCLUDE Irvine32.inc
.data
source  BYTE "This is the source string", 0
target  BYTE SIZEOF source DUP(0)
.code
main PROC
    mov esi, SIZEOF source - 2  ; Start from the end of the source
string
    mov edi, 0                  ; Start from the beginning of the
target string
    mov ecx, SIZEOF source - 1
L1:
    mov al, source[esi]
    mov target[edi], al
    dec esi
    inc edi
    loop L1

    exit
main ENDP
END main
```