Assembly Language Programming

Status Flags

The status flags reflect the outcomes of arithmetic and logical operations performed by the CPU.

- The carry flag (CF) is set when the result of an unsigned arithmetic operation is too large to fit into the destination.
- The overflow flag (OF) is set when the result of a signed arithmetic operation is too large or too small to fit into the destination.
- The sign flag (SF) is set when the result of an arithmetic or logical operation generates a negative result.
- The zero flag (ZF) is set when the result of an arithmetic or logical operation generates a result of zero.

Assembly Programs

We are going to run assembly programs from (http://www.kipirvine.com/asm/) using Visual Studio. I have downloaded all of the example programs and placed them in CS430 Pub. Copy them onto your local machine and start up Visual Studio.

The first program we are going to run is below. Copy this into the Project_Sample project in the examples folder. Run the program. Let's talk about what this program does.

```
TITLE Add and Subtract

; This program
; Last update: 06/01/2006

INCLUDE Irvine32.inc

.code
main PROC

     mov  eax,10000h
     add  eax,40000h
     sub  eax,20000h
     call DumpRegs

     exit
main ENDP
END main
```

What's the difference between the previous program and this one:

```
TITLE Add and Subtract, Version 2         (AddSub2.asm)

; This program adds and subtracts 32-bit integers
; and stores the sum in a variable.
; Last update: 06/01/2006

INCLUDE Irvine32.inc

.data
val1     dword  10000h
val2     dword  40000h
val3     dword  20000h
finalVal dword  ?

.code
main PROC

     mov  eax,val1            ; start with 10000h
     add  eax,val2            ; add 40000h
     sub  eax,val3            ; subtract 20000h
     mov  finalVal,eax        ; store the result (30000h)
     call DumpRegs            ; display the registers

     exit
main ENDP
END main
```

Data Transfer Instructions

The MOV instruction copies from a source operand to a destination operand. The following rules must be observed:

1. Both operands must be the same size.
2. Both operands cannot be memory operands.
3. CS, EIP, and IP cannot be destination operands.
4. An immediate value cannot be moved to a segment register.

MOVZX Instruction

This copies the contents of a source operand into a destination operand and zero extends the value to 16 or 32 bits.

```
movzx ax, 10001111b
```

MOVSX Instruction

This copies the contents of a source operand into a destination operand and sign extends the value to 16 or 32 bits.

```
movsx ax, 10001111b
```

XCHG Instruction

This instruction exchanges the contents of two operands. Operands must be the same size, and cannot be immediate. Why?

```
xchg ax, bx
xchg ah, al
xchg var1, bx
```

What are the values of the registers and the variables after each group of instructions in the following program?

```
TITLE Data Transfer Examples       (Moves.asm)

; Chapter 4 example. Demonstration of MOV and
; XCHG with direct and direct-offset operands.
; Last update: 06/01/2006

INCLUDE Irvine32.inc
.data
val1  WORD 1000h
val2  WORD 2000h

arrayB BYTE  10h,20h,30h,40h,50h
arrayW WORD  100h,200h,300h
arrayD DWORD 10000h,20000h

.code
main PROC

     mov    bx,0A69Bh
     movzx  eax,bx
     movzx  edx,bl
     movzx  cx,bl

     mov   bx,0A69Bh
     movsx eax,bx
     movsx edx,bl
     mov  bl,7Bh
     movsx cx,bl

     mov  ax,val1
     xchg ax,val2
     mov  val1,ax

     mov al,arrayB
     mov al,[arrayB+1]
     mov al,[arrayB+2]

     mov ax,arrayW
     mov ax,[arrayW+2]

     mov eax,arrayD
     mov eax,[arrayD+4]
     mov eax,[arrayD+TYPE arrayD]

     exit
main ENDP
END main
```

Arithmetic Instructions

Let's investigate arithmetic instructions. As well as ADD and SUB, there are:

- INC, DEC instructions
- NEG instruction

Flags affected by Addition and Subtraction

- The Carry flag indicates unsigned integer overflow. For example, if an instruction has an 8-bit destination operand but the instruction generates a result larger than 11111111 binary, the Carry flag is set.
- The Overflow flag indicates signed integer overflow. For example, if an instruction has a 16-bit destination operand but it generates a negative result smaller than -32,768 decimal, the Overflow flag is set.
- The Zero flag indicates that an operation produced zero. For example, if an operand is subtracted from another of equal value, the Zero flag is set.
- The Sign flag indicates that an operation produced a negative result. If the most significant bit of the destination operand is set, the Sign flag is set.
- The Parity flag counts the number of 1 bits in the least significant byte of the destination operand.
- The Auxiliary flag is sent when a 1 bit carries out of position 3 in the least significant byte of the destination operand.

Example Program:

```
TITLE  Addition and Subtraction        (AddSub3.asm)

; Chapter 4 example. Demonstration of ADD, SUB,
; INC, DEC, and NEG instructions, and how
; they affect the CPU status flags.
; Last update: 06/01/2006
INCLUDE Irvine32.inc

.data
Rval   SDWORD ?
Xval   SDWORD 26
Yval   SDWORD 30
Zval   SDWORD 40

.code
main PROC
     ; INC and DEC
     mov  ax,1000h
     inc  ax
     dec  ax

     mov  eax,Xval
     neg  eax
     mov  ebx,Yval
     sub  ebx,Zval
     add  eax,ebx
     mov  Rval,eax

     mov  cx,1
     sub  cx,1
     mov  ax,0FFFFh
     inc  ax

     mov  cx,0
     sub  cx,1
     mov  ax,7FFFh
     add  ax,2

     mov  al,0FFh
     add  al,1

     mov  al,+127
     add  al,1
     mov  al,-128
     sub  al,1

     exit
main ENDP
END main
```

1. Indicate whether or not each of the following instructions is valid.

```
a.   add ax,bx
b.   add dx,bl
c.   add ecx,dx
d.   sub si,di
e.   add bx,90000
f.   sub ds,1
g.   dec ip
h.   dec edx
i.   add edx,1000h
j.   sub ah,126h
k.   sub al,256
l.   inc ax,1
```

2. What will be the value of the Carry flag after each of the following instruction sequences has executed?

```
a.  mov ax,0FFFFh
    add ax,1
b.  mov bh,2
    sub bh,2
c.  mov dx,0
    dec dx
d.  mov al,0DFh
    add al,32h
e.  mov si,0B9F6h
    sub si,9874h
f.  mov cx,695Fh
    sub cx,A218h
```

3. What will be the value of the Zero flag after each of the following instruction sequences has executed?

```
a.    mov ax,0FFFFh
      add ax,1
b.    mov bh,2
      sub bh,2
c.    mov dx,0
      dec dx
d.    mov al,0DFh
      add al,32h
e.    mov si,0B9F6h
      sub si,9874h
f.    mov cx,695Fh
      add cx,96A1h
```

4. What will be the value of the Sign flag after each of the following instruction sequences has executed?

```
a.    mov ax,0FFFFh
      sub ax,1
b.    mov bh,2
      sub bh,3
c.    mov dx,0
      dec dx
d.    mov ax,7FFEh
      add ax,22h
e.    mov si,0B9F6h
      sub si,9874h
f.    mov cx,8000h
      add cx,A69Fh
```

5. What will be the values of the Carry, Sign, and Zero flags after the following instructions have executed?

```
mov ax,620h
sub ah,0F6h
```

6. What will be the values of the Carry, Sign, and Zero flags after the following instructions have executed?

```
mov ax,720h
sub ax,0E6h
```

7. What will be the values of the Carry, Sign, and Zero flags after the following instructions have executed?

```
mov ax,0B6D4h
add al,0B3h
```

8. What will be the values of the Overflow, Sign, and Zero flags after the following instructions have executed?

```
mov bl,-127
dec bl
```

9. What will be the values of the Carry, Overflow, Sign, and Zero flags after the following instructions have executed?

```
mov cx,-4097
add cx,1001h
```

10. What will be the values of the Carry, Overflow, Sign, and Zero flags after the following instructions have executed?

```
        mov ah,-56
        add ah,-60
```