John Carroll

Reading Questions for Monday, October 13th

Section 7.4, skipping 7.4.3

-Questions: 7, 8, 9, 11, 14, 15, in 7.4.7 (pp. 255-256)

-Could you use MOVSX instead of CBW? CWD? CDQ?

-(Bonus) When Visual C++ compiles a C++ program to assembly language, the assembly code it generates only uses IMUL and IDIV, even for unsigned arithmetic. Why does this work, since they’re supposed to be signed arithmetic instructions?

**Questions:**

7. Show an example of sign extension before calling the IDIV instruction with a 16-bit operand.

.data

wordVal SWORD -5000

.code

mov ax,wordVal ; dividend, low

cwd ; extend AX into DX

mov bx,+256 ; divisor

idiv bx ; quotient AX = -19, rem DX = -136

8. What will be the contents of AX and DX after the following operation?

mov dx,0

mov ax,222h

mov cx,100h

mul cx

**DX:AX = 00022200h, CF = 1**

9. What will be the contents of AX after the following operation?

mov ax,63h ; dividend

mov bl,10h ; divisor

div bl ; **AL = 06h, AH = 03h**

11. What will be the contents of AX and DX after the following operation?

mov dx,500h ; not clear dividend, high

mov ax,4000h ; dividend, low

mov bx,10h ; divisor

div bx ;

5004000/10 = 500400  
DX:AX 05004000/ BX 0010 -> This will result in divide overflow – a CPU interrupt. This is because the quotient(500400h) will not fit in the AX register.

14. Implement the following C++ expression in assembly language, using 32-bit unsigned operands:  
val1 = (val2 \* val3) / (val4 - 3)

;(val2 \* val3)   
 mov eax,val2 ; left side  
 mov ebx,val3  
 mul ebx ; EDX:EAX = product

;(val4 - 3)  
 mov ebx, val4 ; right side  
 sub ebx,3

; divide EDX:EAX by EBX  
 div ebx ; final division  
 mov val1, eax

15. Implement the following C++ expression in assembly language, using 32-bit signed operands:  
val1 = (val2 / val3) \* (val1 + val2)

; (val2 / val3)  
mov eax, val2 ;left side  
mov ebx, val3  
cdq ; sign-extend dividend  
idiv ; EDX = remainder  
  
;(val1 + val2)  
mov ebx, val1 ;right side  
add ebx, val2  
  
; multiply the quotient by EBX   
imul ebx  
mov val1, eax

-Could you use MOVSX instead of CBW? CWD? CDQ?

You could for CBW, but not for CWD and CDQ because these two extend into DX:AX and EDX:EAX respectively.

-(Bonus) When Visual C++ compiles a C++ program to assembly language, the assembly code it generates only uses IMUL and IDIV, even for unsigned arithmetic. Why does this work, since they’re supposed to be signed arithmetic instructions?