Using Irvine Library

Read pp 153 – 156 from Textbook

Library Test #2: Random Integers

```
include Irvine32.inc

TAB = 9                               ; ASCII code for Tab

.code
main PROC
     call  Randomize                  ; init random generator
     call  Rand1
     call  Rand2
     exit
main ENDP
```

```
Rand1 PROC
; Generate ten pseudo-random integers.
    mov   ecx,10                    ; loop 10 times

L1: call  Random32                  ; generate random int
    call  WriteDec                  ; write in unsigned decim
    mov   al,TAB                    ; horizontal tab
    call  WriteChar                 ; write the tab
    loop  L1

    call  Crlf
    ret
Rand1 ENDP
```

```
Rand2 PROC
; Generate ten pseudo-random integers from -50 to +49
     mov   ecx,10                     ; loop 10 times

L1:  mov   eax,100                    ; values 0-99
     call  RandomRange                ; generate random int
     sub   eax,50                     ; values -50 to +49
     call  WriteInt                   ; write signed decimal
     mov   al,TAB                     ; horizontal tab
     call  WriteChar                  ; write the tab
     loop  L1

     call  Crlf
     ret
Rand2 ENDP
END main
```

```
3221236194   2210931702   974700167    367494257    2227888607
926772240    506254858    1769123448   2288603673   736071794
-34    +27    +38    -34    +31    -13    -29    +44    -48    -43
```

Library Test #3: Performance Timing

```
include Irvine32.inc

.data
OUTER_LOOP_COUNT = 3
startTime DWORD ?
msg1 byte "Please wait...",0dh,0ah,0
msg2 byte "Elapsed milliseconds: ",0

.code
```

```
innerLoop PROC
     push  ecx                          ; save current ECX value

     mov   ecx,0FFFFFFFh                ; set the loop counter
L1:  mul   eax                          ; use up some cycles
     mul   eax
     mul   eax
     loop  L1                           ; repeat the inner loop

     pop   ecx                          ; restore ECX's saved value
     ret
innerLoop ENDP

END main
```

Unsigned Multiply

Opcode	Mnemonic	Description
`F6 /4`	`MUL r/m8`	Unsigned multiply (AX = AL * r/m8).
`F7 /4`	`MUL r/m16`	Unsigned multiply (DX:AX = AX * r/m16).
`F7 /4`	`MUL r/m32`	Unsigned multiply (EDX:EAX = EAX * r/m32).

Description

Performs an unsigned multiplication of the first operand (destination operand) and the second operand (source operand) and stores the result in the destination operand. The destination operand is an implied operand located in register AL, AX or EAX (depending on the size of the operand); the source operand is located in a general-purpose register or a memory location. The action of this instruction and the location of the result depends on the opcode and the operand size as shown in the following table.

MUL Results

Operand Size	Source 1	Source 2	Destination
Byte	AL	r/m8	AX
Word	AX	r/m16	DX:AX
Doubleword	EAX	r/m32	EDX:EAX

The result is stored in register AX, register pair DX:AX, or register pair EDX:EAX (depending on the operand size), with the high-order bits of the product contained in register AH, DX, or EDX, respectively. If the high-order bits of the product are 0, the CF and OF flags are cleared; otherwise, the flags are set.

```
main PROC
     mov   edx,OFFSET msg1      ; "Please wait..."
     call  WriteString
; Save the starting time
     call  GetMSeconds
     mov   startTime,eax
; Start the outer loop
     mov   ecx,OUTER_LOOP_COUNT
L1:  call  innerLoop
     loop  L1
; Calculate the elapsed time
     call  GetMSeconds
     sub   eax,startTime
; Display the elapsed time
     mov   edx,OFFSET msg2       ; "Elapsed milliseconds: "
     call  WriteString
     call  WriteDec              ; write the milliseconds
     call  Crlf
     exit
main ENDP
```

```
Please wait....

Elapsed milliseconds: 4974
```