

**CSCI 51A/E    MID TERM #1**  
**(slightly modified)**

John Zenor, Spr 1995  
50 minutes

Open book, notes. NO calculators.

1. The following are lines from an assembly language program. Provide the answers after each instruction in the blanks provided. All answers are to be in hex, leading zeroes are NOT necessary. Remember, all of these instructions run in sequence as part of one program.

; ASCII codes for selected letters

; A \$41    E \$45    H \$48    L \$4C    O \$4F    R \$52    T \$54  
; blank (space) = \$20

MOVE.L	#\$12345678,D0	D0 = ___ \$12345678 ___	
ADD.L	#\$FF000000,D0	D0 = \$_____	A.
		C-bit in CCR = _____	B.
		V-bit in CCR = _____	C.
		N-bit in CCR = _____	D.
		Z-bit in CCR = _____	E.
		X-bit in CCR = _____	F.
MOVE.L	#\$12345678,D0	D0 = ___ \$12345678 ___	
SUB.L	#\$80000000,D0	D0 = \$_____	G.
		C-bit in CCR = _____	H.
		V-bit in CCR = _____	I.
		N-bit in CCR = _____	J.

Z-bit in CCR = \_\_\_\_\_ K.

X-bit in CCR = \_\_\_\_\_ L.

MOVEA.L #S12345678,A0 A0 = \$12345678

ADDA.W #S24,A0 D0 = \$\_\_\_\_\_ M.

C-bit in CCR = \_\_\_\_\_ N.

V-bit in CCR = \_\_\_\_\_ O.

N-bit in CCR = \_\_\_\_\_ P.

Z-bit in CCR = \_\_\_\_\_ Q.

X-bit in CCR = \_\_\_\_\_ R.

MOVEA.L #S12345678,A0 A0 = \$12345678

MOVEA.W #S7FFF,A0 A0 = \$\_\_\_\_\_ S.

MOVEA.L #S12345678,A0 A0 = \$12345678

MOVEA.W #S8123,A0 A0 = \$\_\_\_\_\_ T.

MOVE.L #S12345678,D0 D0 = \$12345678

MOVE.L #S1A,D0 D0 = \_\_\_\_\_ U.

C-bit in CCR = \_\_\_\_\_ V.

V-bit in CCR = \_\_\_\_\_ W.

N-bit in CCR = \_\_\_\_\_ X.

Z-bit in CCR = \_\_\_\_\_ Y.

X-bit in CCR = \_\_\_\_\_ Z.

MOVE.L #S12345678,D0 D0 = \$12345678

ADD.B Z,D0 D0 = \$\_\_\_\_\_ A1.

```

MOVE.L  #$12345678,D0  D0 =     $12345678    
MOVE.W  X+4,D0          D0 = $                          B1.

MOVE.L  #$12345678,D0  D0 =     $12345678    
MOVE.W  X+6,D0          D0 = $                          C1.

```

; For the following questions, assume that the data area below begins in \$1000,  
; and that the machine we are using allows absolute addressing modes  
; without using A5 for addressing data.

```

LEA     Z,A0            A0 =                           D1.
MOVE.L  #$12345678,D0  D0 =     $12345678    
MOVE.W  (A0),D0        D0 = $                          E1.

MOVEA.L #Y,A0          A0 = $                          F1.

LEA     X,A0            A0 = $                          G1.

CLR.L   D0
MOVE.W  W+2,D0         D0 = $                          H1.

```

```

ORG     $1000
X:      DC.B   -2,$12,-3,13
Y:      DC.W   -6,$A,$7
Z:      DC.L   $FEDCBA98
W:      DC.B   'HELLO AGAIN. THIS IS NOT QUITE THE END'
END

```

2. Draw a line through any illegal instructions in the following program segment.

```

ADD.W   #5,D0
ADDI.W  #5,D0
ADDQ.W  #5,D0

```

ADD.W #5,A0  
ADDQ.W #5,A0  
ADDQ.W #9,D0  
ADD.W #9,XX  
ADDI.W #9,XX  
ADDA.L #9,A0  
ADDA.B #9,A0

.....  
XX DC.W 5,6,9

3. Answer the following questions about number systems:

A. What is the largest positive number that may be stored in 24 bits.

Give result as an expression containing a power of two. \_\_\_\_\_

B. What is the most negative number that may be stored in 24 bits.

Give result as an expression containing a power of two. \_\_\_\_\_

C. How many different numbers may be stored in 24 bits.

Give result as an expression containing a power of two. \_\_\_\_\_

D. After an addition of two two's-complement signed numbers, what bit in the CCR register tells you if the answer fit, and is correct? \_\_\_\_\_

E. After an addition of two unsigned numbers, what bit in the CCR register tells you if the answer fit, and is correct? \_\_\_\_\_

F. Assume that you have just compared two addresses using a CMP instruction. What is the correct conditional branch instruction to branch if the result was greater than or equal to zero? \_\_\_\_\_

G. Assume that you have just compared two two's-complement numbers using a CMP instruction. What is the correct conditional branch instruction to branch if the result was less than zero? \_\_\_\_\_

4. Fill in the blanks to make this a correct assembly language program that will execute the loop body ten times.

; Part A: My first illustrious incomplete program

```
START:  CLR.L    D0
LOOPIT:  CMP.L    _____,D0  ;FILL IN THE BLANK
        BGT     LPEXIT
; some code in the body of the loop
```

```
        ADDQ.L  #1,D0
        BRA     LOOPIT
```

LPEXIT:

; Part B: My second illustrious incomplete program. Note that it is much shorter!

```
START:  MOVE.L  _____,D0  ;FILL IN THE BLANK
LOOPIT:
; some code in the body of the loop
```

```
        SUB.L   #1,D0
        BGE    LOOPIT
```

LPEXIT: