Name: Solution Box:__________

Test 1 EC331 Embedded Systems (100 Point Maximum) Spring 2008 (KEH)

Closed notes, open CPU12 Manual - 100 points max. 60 minutes
"Fill in the Blank"/"Multiple Choice" Questions

This is an objective test. You must have exactly the correct answer to each question for credit. (No partial credit given) All questions on this test apply to the M68HC12 microcontroller.

1. (30 points – 1 point per blank) Fill in the chart below, indicating how many bytes must be READ from memory and how many bytes must be WRITTEN to memory by each instruction AFTER THE INSTRUCTION HAS BEEN FETCHED.

	Assembly Code	# Bytes Read From Memory	# Bytes Written to Memory
	LDX #$2A	0	0
	LDX $2A	2	0
	ADDA $4000	1	0
	STD $12, X	0	2
	RTS	2	0
a.	LDX 3, X	2	0
b.	LEAX 3, X+	0	0
c.	LDX $0834	2	0
d.	LDX #$0834	0	0
e.	MOVB 5, +X, 2, -Y	1	1
f.	DEC 5, -X	1	1
g.	DEC [5, X]	3	1
h.	MOVW #4, $0800	0	2
i.	PSHX	0	2
j.	ROR $0800	1	1
k.	ROR [$0800, X]	3	1
l.	JSR $1234, X	0	2
m.	JSR [$1234, X]	2	2
n.	BCLR $0FFF, $F0	1	1
o.	TARG: BRCLR $250, X, $20, TARG	1	0

2. (38 points – 1 point per blank) Assuming the instructions below are executed in sequence, fill in the blanks below:

(A)

```
      1
LDAA #$98
ADDA #$79
①C  11
```

After this ADDA instruction executes, the condition code (CCR) flags are:

H = 1 N = 0 Z = 0 V = 0 C = 1

Register A contains $11

(B)

```
DAA
    6C
```

After this DAA instruction executes, Register A contains $77

(C)

```
LDAA #$E5
ADDA #$C7
①C  AC
```

After this ADDA instruction executes, the condition code (CCR) flags are:

H = 0 N = 1 Z = 0 V = 0 C = 1

Register A contains $AC

(D)

```
   0↷ 1↷
LDAA #$85
SUBA #$5B
      2A
```

After this SUBA instruction executes, the condition code (CCR) flags are:

N = 0 Z = 0 V = 1 C = 0

Register A contains $2A

(E)

```
   ①↷①↷
LDAA #$43
SUBA #$CD
      76
```

After this SUBA instruction executes, the condition code (CCR) flags are:

N = 0 Z = 0 V = 0 C = 1

Register A contains $76

(F)

```
   0↷ 1↷
LDD  #$ABCD
SUBD #$5DCB
      4E02
```

After the SUBD instruction executes, the condition code (CCR) flags are:

N = 0 Z = 0 V = 1 C = 0

Register D contains $4E02

(G)

```
      ①↷
LDAA #$A5
CMPA #$C2
        E3
```

After the CMPA instruction executes, the condition code (CCR) flags are:

N = 1 Z = 0 V = 0 C = 1

Register A contains $A5 ← CMPA does not alter "A"

(H)

```
LDX  #$0123
LEAX $0123, X
TFR  X, D
ADDD #$FDBA
      1 1
      0246
      FDBA
①C    0000
```

After the ADDD instruction executes, the condition code (CCR) flags are:
N = 0 Z = 1 V = 0 C = 1
Register D contains $0000

3. (14 Points – 1 pt per blank) Given the following address map in an M68HC12-based system, fill in the blanks:

Address	Contents
$0820	$DE
$0821	$08
$0822	$34
$0823	$02
$0824	$02
$0825	$35
$0826	$00
$0827	$24
$0828	$20
$0829	$00
$082A	$12
$082B	$10
$082C	$24
$082D	$00
$082E	$23
$082F	$00
$0830	$21
$0831	$05
$0832	$08
$0833	$35
$0834	$08
$0835	$40
$0836	$08
$0837	$2E
$0838	$08
$0839	$20
$083A	$45
$083B	$67
$083C	$20
$083D	$00
$083E	$20
$083F	$02
$0840	$78
$0841	$37
$0842	$02

A. The following two instructions are executed:

```
LDX $832      x = 0835
LDD 8, X+     x = 083D
```

now register "A" contains $ 40

now register “B” contains $ 08

now register “X” contains $ 083D

B. The following two instructions are executed

```
LDY  #$0832     x = 0832
LDX  5,+Y       Y = 0837, x = 2E08
```

Now register “Y” contains $ 0837 and register “X” contains $ 2E08

C. The following instructions are

```
LDX  $834      x =  0840
LDX  -3,X      x =  0020
LDY  $821      Y =  0834
LDAA 2,Y       A =  08
LDAB [2,Y]     B =  23
```

Now X contains $ 0020 and D contains $ 0823

D. The following four instructions are executed:

```
LDS  #$1000
LDY #$0836
PSHY
PULB           B = 08
PULA           A = 36
PSHY
LDY   2,Y      Y = 0820
LEAX 2,Y       x = 0822
```

```
      1000 · --
      FFF   36
SP -> FFE   08
```

Now "Y" contains $ 0820 “S” contains $ 0FFE “D” contains $ 3608 “X” contains $ 0822

E. Assume the memory map above, and that he following program fragment is executed from location START:

```
START:      LDY #3
            LDD #0
            LDX #$0820
LOOP1:      ADDD 2,+X
            DBNE  Y, LOOP1
            STD $0800
LOOP2:      BRA LOOP2
```

```
  3402
  0235
  0024
  ----
  365B
```

After the BRA instruction is executed, indicate the contents of Y, X, and RAM locations $800 and $0801 ?

Y = $ 0000 X = $ 0826 ($800) = $ 36 ($801) = $ 5B

4. (18 points --- 2 pts per blank) Subroutine “ToUpper” converts lower case letters found in an ASCII string (in RAM) to upper

case (capital) letters. Recall that lower case letters "a, b, ... z" are represented by the ASCII codes $61, $62, …$7A; while the upper case letters "A, B, … Z" are represented by the ASCII codes $41, $42, … $5A. This ASCII string must be null-terminated, which means that it must end with the NULL ASCII character, whose value is $00.

Subroutine "ToUpper" is called by

1) Pushing the (16-bit) starting address of a null-terminated ASCII string (stored in RAM) on the stack.
2) Pushing a (16-bit) RAM address which, upon return from the subroutine, will hold the number of characters that were changed from lower case to upper case.
3) Calling the routine using a JSR or BSR instruction.

Upon return, the null-terminated ASCII string (which must be in RAM) will have been converted to all upper case (capital) letters. The input arguments must be removed from the stack after returning to the main program. Subroutine "ToUpper" must be written so that upon return to the calling program, the values that were in registers X, Y, and D before this subroutine was called are not changed. *First construct a memory map of the stack (to the right of the code below) just after the registers have been preserved on the stack in Subroutine "ToUpper"*, then fill in the NINE missing blanks in the code for subroutine "ToUpper" and its calling test program "ToUpperTest", which are shown below.

```
                        XDEF ToUpperTest
                        ABSENTRY ToUpperTest
                        ORG $0800
STRING_START            DC.B "This is a TEST to Convert an ASCII STRING TO all Upper Case Characters", 0
NR_LOWERCASE            DS.W  1
        ;Note: after the program below has been run to location "STOP_HERE", location NR_LOWERCASE
        ;should contain the value 35 (in decimal), since there 35 lower case letters need to be
        ;converted from lower  to upper case. Also, the null-terminated ASCII string at location
        ;STRING_START will be converted to all upper case letters.
              ORG $4000
ToUpperTest:  LDS #$1000
              LDX #STRING_START
              PSHX
              LDX #NR_LOWERCASE
              PSHX
              BSR ToUpper
              LEAS ____4____ , SP          ;***BLANK 1              addr String Lo
STOP_HERE:    BRA STOP_HERE                                SP+10 → addr String Hi
ToUpper:      PSHD                                                  addr Lowercase NR Lo
              PSHX                                          SP+8 → addr Lowercase NR Hi
              PSHY                                                  PC Lo
              LDX ___10___ , SP            ;***BLANK 2               PC Hi
              LDY #0                                                D Lo
BACKAGN:      LDAA 0,X                                              D Hi
              BEQ DONE_STRING                                       X Lo
NOT_DONE:     CMPA #$61                                             X Hi
              BLO LC_NoT_FOUND             ;***BLANK 3               Y Lo
              CMPA #$7A                                   SP → Y Hi
              BHI LC_NOT_FOUND             ;***BLANK 4
              SuBA_____________ #$20       ;***BLANK 5
              STAA ___0,X___               ;***BLANK 6
               INY
LC_NOT_FOUND: INX
              BRA BACKAGN
DONE_STRING:STY [ ___8___ , SP ]           ;***BLANK 7
              PULY
              PULX
              PULD____________             ;***BLANK 8
              rts_____________             ;***BLANK 9
              ORG $FFFE
              DC.W ToUpperTest
```