Name:______________________________________ Box:__________

Test 1 EC331 Embedded Systems (100 Point Maximum) Fall 2007 (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.	RTI	9	0
b.	SWI	2	9
c.	JSR $4060	0	2
d.	JSR [$4060,X]	2	2
e.	MOVW 5,X, 2,Y	2	2
f.	INC 5,X	1	1
g.	INC [5,X]	3	1
h.	MOVB #4, $3800	0	1
i.	PSHB	0	1
j.	LSL $3800	1	1
k.	LDY $1234, X	2	0
l.	LEAY $1234, X	0	0
m.	TARG: BRSET $40,X,$20,TARG	1	0
n.	BCLR $1000,$F0	1	1
o.	BSET $1234,X,$20	1	1

2. (39 points – 1 point per blank) Assuming the instructions below are executed in sequence, fill in the blanks below:

(A)
```
LDAA #$98
ADDA #$89
```
1 21

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

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

Register A contains $ 21

(B)
```
DAA
```
$21
+ 66
1 87

After this DAA instruction executes, Register A contains $ 87

and now the Carry condition code flag must be C = 1

(C)
```
LDAA #$75
ADDA #$C7
```
1 3C

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

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

Register A contains $ 3C

(D)
```
LDAA #$85
SUBA #$6C
```
0 19

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

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

Register A contains $ 19

(E)
```
LDAA #$4F
SUBA #$ED
```
1 62

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

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

Register A contains $ 62

(F)
```
LDD   #$ABCD
SUBD #$4BCE
```
0 5FFF

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

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

Register D contains $ 5FFF

(G)
```
LDAA #$C2
CMPA #$C2
```
00

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

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

Register A contains $ C2

(H)
```
LDX #$0123
LEAX $0321,X
TFR  X, D
ADDD #%0010100100000101
```
After the ADDD instruction executes, the condition code (CCR) flags are:

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

Register D contains $ 2D49

0123
0321
444
2905
2D49

3. (13 Points – 1 pt per blank) Given the following address map in an M68HC12-based system, fill in the blanks:

Address	Contents
$0020	$DE
$0021	$02
$0022	$34
$0023	$02
$0024	$02
$0025	$35
$0041	$12
$0042	$34
$0043	$20
$0044	$00
$0045	$12
$0205	$10
$0206	$24
$0234	$00
$0235	$23
$0236	$00
$0237	$21
$0238	$05
$0239	$39
$02DE	$35
$02E0	$01
$02E1	$A5
$02E2	$36
$02E3	$FE
$1004	$89
$1005	$FE
$1024	$45
$1025	$67
$3437	$20
$3438	$00
$3439	$20
$343A	$02
$343B	$78
$3734	$37
$3735	$02

A. The following two instructions are executed:

LDX $21 ←

LDD 2,+X ←

X =#0234

D =#00 21 , X = X+2
= #236

A B

Now register "A" contains $ 00

register "B" contains $ 21

register "X" contains $ 0236

B. The following two instructions are executed

LDY #$0236

LDX 1,Y+

Now register "Y" contains $ 0237 and register "X" contains $ 0021

C. The following instructions are

LDX $234

LDX -3,X

LDY $21

LDAA 2,Y

LDAB [2,Y]

0023

x = DE02

0234

A = 00

B = 02

Now X contains $ DE02 and D contains $ 0002

(recall that D = A:B)

D. The following four instructions are executed:

LDS #$3F00

LDY #$1234

PSHY

PUL~~A~~B

PUL~~B~~A

PSHB

LEAY $4321,Y

34

12

A = 12

B = 34

PSH B ⇒ SP = 3EFF

4321

1234

5555

Now accumulator register "Y" contains $ 5555 "S" contains $ 3EFF "D" contains $ 3412

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

```
START:      LDAA #4
            CLRB
            LDX #$0234
LOOP1:      ADDB 1,X+
            DBNE  A,LOOP1
            STAB $4000
LOOP2:      BRA LOOP2
```

00 A = 4 x = 235

23 A = 3

00 A = 2

21 A = 1

44

After the STAB instruction is executed, what is in A and X, and what is stored at location $4000 ?

A = $ 00 X = $ 0238 ($4000) = $ 44 (Contents of address $4000)

4. (18 points --- 2 pts per blank) Fill in the TEN blanks in the calling program "FINDCHAR_TEST" and the subroutine "FINDCHAE" below. Subroutine FINDCHAR is called by

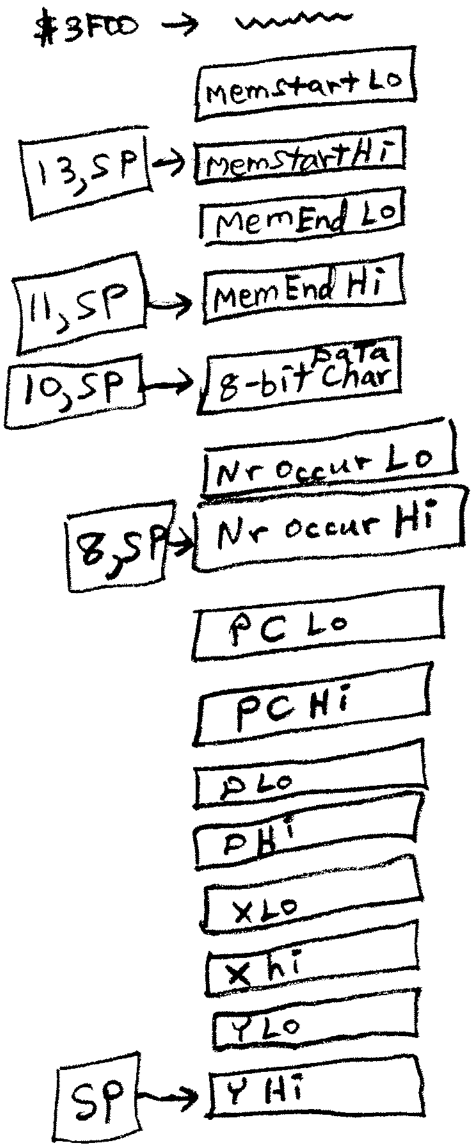
(1) pushing a 16-bit Memory Start address on the stack.
(2) pushing a 16-bit Memory End address on the stack.
(3) pushing an 8-bit data byte that is to be searched for.
(4) pushing a 16-bit RAM address which, upon return from the subroutine, will hold the number of times this 8-bit data byte is found between the Memory Start address and the Memory End address. (The contents of the Memory Start address are included in the search, but the contents of the Memory End address are NOT included in the search.)

The input arguments must be cleaned off of the stack after returning to the main program. Subroutine FINDCHAR must NOT disturb the values in the registers D, X, and Y back in the calling program. (*Hint: first make a map of the stack after the PSHY executes in subroutine FINDCHAR.*)

```
     XDEF FINDCHAR_TEST
      ABSENTRY FINDCHAR_TEST
      ORG $3800
NR_OCCURRENCES:
      DS.W 1
      ORG $4000
MEMSTART:  DC.B "This is a test to count the number of occurrences of the lower case letter e"
MEMEND:    DC.B 0
FINDCHAR_TEST:
      LDS #$3F00
      LDX #MEMSTART
      PSHX
      LDX # MEMEND                      ;Blank 1
      PSHX
      LDAA #'e'
      PSHA                              ;Blank 2
      LDX #NR_OCCURRENCES
      PSHX
      BSR FINDCHAR
      LEAS 7,SP                         ;Blank 3
STOP_HERE:  BRA STOP_HERE

FINDCHAR:
      PSHD
      PSHX
      PSHY
      LDX  13 ,SP                       ;Blank 4
      LDY #0
      LDAA  10 ,SP                      ;Blank 5
NOT_DONE:  CMPA 1,X+                    ;Blank 6
      BNE NOT_FOUND
      INY                               ;Blank 7
NOT_FOUND:
      CPX  11 ,SP                       ;Blank 8
      BNE NOT_DONE
      STY [ 8,SP ]                      ;Blank 9
      PULY
      PULX
      PULD
      RTS

      ORG $FFFE
      DC.W FINDCHAR_TEST
```


$3F00 →
MemStart Lo
13,SP → MemStart Hi
MemEnd Lo
11,SP → MemEnd Hi
10,SP → 8-bit Data Char
Nr Occur Lo
8,SP → Nr Occur Hi
PC Lo
PC Hi
D Lo
D Hi
X Lo
X hi
Y Lo
SP → Y Hi