

Name: Solution

Box: \_\_\_\_\_

**Test 1 EC331 Embedded Systems (100 Point Maximum) Fall 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. (32 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. ADDD \$804	<u>2</u>	<u>0</u>
b. ADDD #\$804	<u>0</u>	<u>0</u>
c. JSR \$4060	<u>0</u>	<u>2</u>
d. JSR [\$4060,X]	<u>2</u>	<u>2</u>
e. MOVW A,X, B,Y	<u>2</u>	<u>2</u>
f. INC 5,X	<u>1</u>	<u>1</u>
g. INC [5,X]	<u>3</u>	<u>1</u>
h. MOVB #4, \$0800	<u>0</u>	<u>1</u>
i. PSHD	<u>0</u>	<u>2</u>
j. LSL [6,SP]	<u>3</u>	<u>1</u>
k. LDY \$1234, X	<u>2</u>	<u>0</u>
l. LEAY \$1234, X	<u>0</u>	<u>0</u>
m. TARG: BRCLR D,X,\$20,TARG	<u>1</u>	<u>0</u>
n. BCLR \$0800,\$F0	<u>1</u>	<u>1</u>
o. BSET \$0800,Y,\$F0	<u>1</u>	<u>1</u>
p. RTI	<u>9</u>	<u>0</u>

Read 2-byte  
addr from  
"5,X", then  
Read data  
byte at  
that addr

Read  
indicated  
memory  
Byte

Write  
modified  
byte  
back to  
RAM

9 bytes  
Unstacked: PC, X, Y, D, CCR

2. (20 points – 0.5 point per blank) Assuming the instructions below are executed in sequence, fill in the blanks below:

(A) LDAA #\$79  
 ADDA #\$58



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

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

Register A contains \$ D1

(B) DAA

After this DAA instruction executes, Register A contains \$ 37

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

(C) LDAA #\$E4  
 ADDA #\$C7



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

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

Register A contains \$ AB

(D) LDAA #\$83  
 SUBA #\$6B

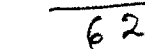


After this SUBA instruction executes, the condition code (CCR) flags are:

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

Register A contains \$ 18

(E) LDAA #\$4F  
 SUBA #\$ED

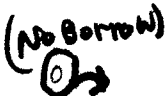


After this SUBA instruction executes, the condition code (CCR) flags are:

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

Register A contains \$ 62

(F) LDD #BAD0  
 SUBD #\$4BCE



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

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

Register D contains \$ 6F02

(G) LDAA #\$AD  
 CMPA #\$AD

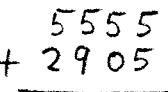


After the CMPA instruction executes, the condition code (CCR) flags are:

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

Register A contains \$ AD

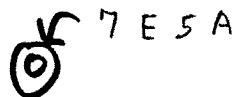
(H) LDX #\$1234  
 LEAX \$4321,X  
 TFR X, D



ADDD #%0010100100000101 After the ADDD instruction executes, the condition code (CCR) flags are:

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

Register D contains \$ 7E5A



3. ( 30 Points – 1.75 pts 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

A. The following two instructions are executed:  
 LDX \$0021  $X = 0234$   
 LDD 2,X+  $D = 0023$   
 Now A = \$ 00 B = \$ 23 X = \$ 0236

B. The following two instructions are executed  
 LDY #\$0236  $Y = 0236$   
 LDX 2,Y  
 Now X = \$ 0539 and Y = \$ 0238

C. The following instructions are  
 LDX \$0234  $X = 0023$   
 LDY -2,X  
 Now X = \$ 0023 and Y = \$ 0234  
 LDY \$0021  $Y = 00234$   
 LDAA 2,Y  
 LDAB [2,Y]

Now A = \$ 00 and B = \$ 02

D. The following four instructions are executed:  
 sp → 34  
 → 12  
 sp → 34

LDS #\$1000  
 LDY #\$1234  
 PSHY  
 PULB  
 PULA  
 PSHY  
 PSHA  
 LEAY \$6543,Y

1234  
 6543  
 7777

Now Y = \$ 7777 S = \$ FFD D = \$ 3412 (\$FFF) = \$ 34 (\$FFD) = \$ 34

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

START: LDAA #3  
 CLRB  
 LDX #\$0234  
 LOOP1: ADDB 1,X  
 DBNE A,LOOP1  
 STAB \$0800  
 LOOP2: BRA LOOP2

23  
 00  
 21  
 44

After the STAB instruction is executed, what is in A and X, and what is stored at location \$0800 ?

A = \$ 00 X = \$ 0237 (\$0800) = \$ 44

4. (18 points --- 2 pts per missing program blank. Note: the stack map entries will not be graded, but you will no credit the this entire problem if the stack map is not filled in!) Fill in the nine blanks in the calling program "FINDCHAR\_TEST" and the subroutine "FINDCHAR" that appear below. Subroutine FINDCHAR is called by doing the following in the calling program:

- (1) pushing a 16-bit Memory Start address of a "null-terminated" ASCII string on the stack.
- (2) pushing an 8-bit data byte (ASCII Code) that is to be searched for within the null-terminated ASCII string.
- (3) pushing the address of a RAM word 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 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.

Begin by filling in a map of the stack after the PSHY executes in subroutine FINDCHAR. See the right side of the page below:

```

XDEF FINDCHAR_TEST
ABSENTRY FINDCHAR_TEST

ORG $800
NR_OCCURRENCES: DS.W 1

ORG $4000
STRINGSTART: DC.B "This is a test to count the occurrences of the lower case letter e", 0
    
```

```

FINDCHAR_TEST:
LDS #$1000
LDX #STRINGSTART ;Blank 1
PSHX
LDAA #'e'
PSHA
LDX #NR_OCCURRENCES
PSHX
BSR FINDCHAR
LEAS 5,SP ;Blank 2
STOP_HERE: BRA STOP_HERE

FINDCHAR:
PSHD
PSHX
PSHY
LDX 11,SP ;Blank 3
LDY #0
LDAA 10,SP ;Blank 4
NOT_DONE: TST 0,X
BEQ DONE ;Blank 5
CMPA 1,X+
BNE NOTFOUND ;Blank 6
INY
NOT_FOUND: BRA NOT_DONE ;Blank 7
DONE: STY [8,SP] ;Blank 8
PULY
PULX
PULD
RTS ;Blank 9
    
```

Put Your Stack Map here:  
(Not all the blanks will be filled in.)

Addr	Contents
\$1000	---
\$0FFF	STR START Adr Low
\$0FFE	STR START Adr HI
\$0FFD	Char To Find
\$0FFC	Result Adr Low
\$0FFB	Result Adr HI
\$0FFA	PC LOW
\$0FF9	PC HI } RTN Adrn
\$0FF8	D LOW
\$0FF7	D HI
\$0FF6	X LOW
\$0FF5	X HI
\$0FF4	Y LOW
\$0FF3	Y HI
\$0FF2	_____
\$0FF1	_____
\$0FF0	_____

Pound Sign essential!

Clean the 5 input arguments off stack

"Square brackets" needed so result written to the address contained on stack!