Name: Solution Box:

Test 1 EC331 Embedded Systems (100 Point Maximum) Fall 2009 (KEH)

Closed notes, open Huang Textbook Only - 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 9S12C128 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
	RTI	9	0
a.	INC 50,X	1	1
b.	ADDA #$84	0	0
c.	JSR $4060,X	0	2
d.	JSR [$4060,X]	2	2
e.	MOVW A,X, 2,-Y	2	2
f.	MOVW #1234, 2,Y-	0	2
g.	INC [50,X]	3	1
h.	MUL	0	0
i.	PULX	2	0
j.	LSR [6,SP]	3	1
k.	LDY $1234, X	2	0
l.	LEAY $1234, X	0	0
m.	TARG: BRSET A,X,$20,TARG	1	0
n.	BSET $0400,$F0	1	1
o.	BCLR $0400,Y,$F0	1	1
p.	SWI	0	9

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 #$89
  ----------
    02
```

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

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

Register A contains $ 02

(B)
```
  DAA   02
      + 66
  ----------
    1 0 6 8
```

After this DAA instruction executes, Register A contains $ 68

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

(C)
```
  LDAA #$D5
+ ADDA #$B7
  ----------
  (1) 8C
```

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

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

Register A contains $ 8C

(D)
```
  LDAA #$92
- SUBA #$6B
  ----------
    27
```

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

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

Register A contains $ 27

(E)
```
  LDAA #$3E
- SUBA #$ED
  ----------
    51
```

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

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

Register A contains $ 51

(F)
```
  LDD  #$DEAD
- SUBD #$BEEF
  ----------
    1FBE
```

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

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

Register D contains $ 1FBE

(G)
```
  LDAA #$AD
- CMPA #$35
  ----------
    78
```

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

N = 0 Z = 0 V = 1 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

```
  1234
+ 4321
  ----
  5555
+ 2905
  ----
0 7E5A
```

X = $5555

3. (30 Points – 1.75 pts per blank) Given the following address map in an 9S12C128-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 $0024 X = 0235

LDD 1,-X X = 0234, D = 0023

Now A = $ 00 B = $ 23 X = $ 0234

B. The following two instructions are executed

LDAA $02E0 A = 01

LDY #$0236 Y = 0236

LDX A,Y X = 2105

Now X = $ 2105 and Y = $ 0236

C. The following instructions are

LDX #36 = # $24

LDY -2,X Y = 3402

Now X = $ 0024 and Y = $ 3402

LDY $0024 Y = 0235

LDAA -1,Y A = 00

LDAB [-1,Y] B = 02

LEAX -1,Y X = 0234

Now D = $ 0002 and X = $ 0234

D. The following sequence of instructions are executed:

stack adr

LDS #$1000

LDY $1024 Y = 4567

PSHY 0FFE 45

PULA A = 45, S = 0FFF 0FFD 67

PULB B = 67, S = 1000

PSHY

PSHB

PULY Y = 6745, S = 0FFF

Now Y = $ 6745 S = $ 0FFF D = $ 4567 ($0FFF) = $ 67 ($0FFE) = $ 45

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

```
START:   LDAA #4
         CLRB
         LDX #$0239
LOOP1:   ADDB 1,X-
         DBNE A,LOOP1
         STAB $0400
LOOP2:   BRA LOOP2
```

39
05
21
+ 00
5F

After the STAB instruction is executed, what is in A and X, and what is stored at location $0400 ?

A = $ 00 X = $ 0235 ($0400) = $ 5F

4. (18 points --- 1.5 pts per missing program blank.) Subroutine "**String_Compare**"
Subroutine "String_Compare" compares the first N elements of two null-terminated ASCII strings, where N is the length of the shorter of the two strings. (A null-terminated string must end in the value $00.) The calling sequence follows:
(1) Push the ***starting address*** of "*null-terminated*" ASCII String1 on the stack.
(2) Push the ***starting address*** of "*null-terminated*" ASCII String2 on the stack.
(3) Push the ***address of a RAM word*** which, upon return from the subroutine, will hold the address of the element in String1 where the two strings disagree, or it will hold a value of 0 if the first N characters of the two strings are identical.
The input arguments *must be cleaned off* of the stack after returning to the main program. Subroutine **StringCompare** must NOT disturb the values in the registers D, X, and Y back in the calling program. Note: the stack map entries will not be graded, *but you will get no credit for the entire problem if the stack map is not filled in*!) ***Begin by filling in a map of the stack after the PSHY executes in subroutine* String_Compare.** ***See the right side of the page below.*** Then fill in the twelve blanks in the calling program "**String_Compare_Test**" and the subroutine "**String_Compare**" that appear below.

```
                XDEF String_Compare_Test
                ABSENTRY String_Compare_Test
                ORG $400
Mismatch_Address:  DS.W 1
                ORG $4000
STRING1:        DC.B "This is a test to compare two strings", 0
STRING2:        DC.B "This is a test to compare 2 strings",0   ;***After running, Mismatch_Address contains $401A***
String_Compare_Test:                                           ;     Put Your Stack Map here:
                LDS #$1000                                     ;(Not all the blanks will be filled in.)
                LDX  #STRING1                                  ; Addr        Contents
                PSHX                                           ;$1000        ---
                LDX  #STRING2                                  ;$0FFF        STR1 LO
                PSHX                                           ;$0FFE        STR1 HI
                LDX #Mismatch_Address                          ;$0FFD        STR2 LO
                PSHX                                           ;$0FFC        STR2 HI
                BSR String_Compare                             ;$0FFB        Mismatch Adr LO
                LEAS  6,SP              ;Blank 1               ;$0FFA        Mismatch Adr Hi
STOP_HERE:      BRA STOP_HERE                                  ;$0FF9        PC LO
                                                               ;$0FF8        PC HI
String_Compare: PSHD                                           ;$0FF7        D LO
                PSHX                                           ;$0FF6        D HI
                PSHY                                           ;$0FF5        X LO
                LDX   12  ,SP           ;Blank 2               ;$0FF4        X HI
                LDY   10  ,SP           ;Blank 3               ;$0FF3        Y LO
NextChar:       TST 0,X                                        ;$0FF2        Y HI
                BEQ NoMismatchFound     ;Blank 4               ;$0FF1        _____
                TST 0,Y                 ;Blank 5               ;$0FF0        _____
                BEQ NoMismatchFound                            ;$0FEF        _____
                LDAA  1  ,X+            ;Blank 6
                LDAB  1  ,Y+            ;Blank 7
                CBA
                BNE MismatchFound       ;Blank 8
                BRA NextChar
NoMismatchFound: LDX #0
                STX [8,SP]              ;Blank 9
                BRA DONE
MismatchFound:  DEX
                STX [8,SP]              ;Blank 10
DONE:
                PULY
                PULX                    ;Blank 11
                PULD
                RTS                     ;Blank 12
                ORG $FFFE
                DC.W String_Compare_Test
```

12,SP → ;$0FFE
10,SP → ;$0FFC
8,SP → ;$0FFA
SP → ;$0FF2

X points to STRING1

Y points to STRING2