CS 273, FINAL EXAM, Fall 1997

Notes: Open Manuals and text; No calculators.
There are a total of 100 points. Exam will be graded out of 100.

1. (6 points) Show the S1 records the assembler translates the following code into. Just fill Xs in the checksum positions.

```
        org 00
fcb     $13
        org $f800
start   LDAA #$25
```

2. (4 points) Consider the command LSL 3,X. List the main thing that happens during each execution cycle.

3. (6 points) List the main differences between interrupt service routines and subroutines (include how they are called and what happens when they are called).

4. (5 points) Recall that when the board is in download mode, and reset is pressed, the Program Counter is loaded with the contents of a memory location in ROM, namely $BFFE$. $BFFE$ contains the address of a program in ROM. **What does this specific program do?** What is the last instruction of this program?

5. (2 points) What is the range of unsigned values that can be represented in N bits?

6. (2 points) How many different values can be represented in N bits?

7. (5 points) List all of the things that must be true for an SCI transmission interrupt to be raised.

8. (3 points) What value is placed in the PC when an SCI interrupt request is serviced?

9. (5 points) Suppose that OC3 interrupts have been properly setup, and an OC3 interrupt request is raised (and is the highest priority interrupt request pending). Exactly what would happen if you make the following error in your interrupt service routine? Please explain your answer.

```
OC3ISR:  LDD TOC3
         ADDD #2000
         STD TOC3
         CLI
         LDAA #%00010000
         STAA TFLG1
         ...
         RTI

...

         ORG $FFE2
         FDB OC3ISR
```

10. (4 points) Suppose that you want an OC3 interrupt to be generated every 4 ms (4×10^{-3}). What value should be added to TOC3? Assume that the clock rate driving the TCNT timer is 300 ns. (300×10^{-9}). **To get full credit, show your work.**

11. Consider the following code fragment, in which a branch instruction is missing.

```
        LDAA #$FF
loop    CMPA #$0F
        _____ end1
        DECA
        BRA loop
end1    ...
```

(a) (5 points)
What is the binary result of the subtraction performed by the CMPA instruction? Please show your work.

If we view this as a signed value, what decimal value is it?

If we view this as an unsigned value, what decimal value is it?

(b) (3 points) If the underline is filled with a BLS instruction, how many times is the loop executed?

(c) (3 points) If the underline is filled with a BLE instruction, how many times is the loop executed?

12. (6 points) Consider the following program fragment. For each instruction, state any changes to registers or memory. Be sure to indicate **all** locations whose values are changed. **Please use numerical addresses in your answer.**

```
        ORG $60
input   RMB 9
arg1    RMB 2

        ORG $F800

start   LDX #$20

        STX arg1

        LDAA #input

        STAA 3+input

        STX  4,X
```

13. (14 points) Following is an incomplete assembly listing. Hand assemble the remaining instructions and fill in the blanks. The places to fill in are marked by arrows.

```
                                org $70
0070                    input   rmb 30
008e                    cur     rmb 2

f800                            org  $F800
f800 9e ff              start   LDS  $00FF

f82e            <---            LDX  #input

f831            <---            STX  cur

f833            <---            LDAA #'p

f835            <---            STAA 0,X

f962 de 00                      LDX  sstate
f964 bd f9 a0           outer   jsr  getchar

f967            <---            PSHX

f968 18 38                      PULY
f96a c6 01                      LDAB #1

f96c c1 01              inner   CMPB #1

f96e            <---            BNE END1

f970 18 8c ff ff                CPY #$FFFF
f974 27 1e                      BEQ END1
f976 37                         PSHB
f977 18 e6 00                   LDAB 0,Y
f97a 11                         CBA
f97b 33                         PULB
f97c 26 11                      BNE  ELSE1
f97e           <---             LDX  1,Y
f981 c6 00                      LDAB #0
f983 9c 02                      CPX  fstate
f985 26 e5                      BNE  inner
f987 86 01                      LDAA #1
```

```
f989 97 14                         STAA bells
f98b de 00                         LDX  sstate
f98d 20 dd                         BRA  inner
f98f 18 ee 03             ELSE1    LDY  3,Y

f992             <---              bra  inner

f994 c1 01                END1     CMPB #1
f996 26 cc                         BNE  outer
f998 86 00                         LDAA #0
f99a 97 14                         STAA bells
f99c de 00                         LDX  sstate
```

14. (15 points) Write the following subroutine.

```
**************************************************************
*Subroutine max
*Below, (SP) means the contents of the stack pointer upon
*       first entering the subroutine
*Input: A, 2-byte address of an array of 1-byte, signed integers
*          (a call-by-reference parameter)
*         --  stored at (SP)+3 and (SP)+4
*       size, the size of the array, 1 byte (a call-by-value parameter)
*         -- stored at (SP)+5
*
*Returns:  -- the maximum element of the array;  the
*             returned value should be placed just under the
*             return address
*
****************************************************************************
```

15. (12 points) Pieces of hexmon are given to you on a separate handout. Feel free to also look at hexmon.asm in the green class notes.

Suppose that the command "w230045" is entered into the MAC (so that cmd-write-byte is jumped to). Assume that the stack pointer has been properly initialized, but that the stack is empty up to this point. Show the contents of the stack and the stack pointer at the points labeled A,B,C, and D. Just fill in the table below. **But show your work on the separate handout for partial credit.**

	Point A	Point B	Point C	Point D
00F6	-------	-------	-------	-------
00F7	-------	-------	-------	-------
00F8	-------	-------	-------	-------
00F9	-------	-------	-------	-------
00FA	-------	-------	-------	-------
00FB	-------	-------	-------	-------
00FC	-------	-------	-------	-------
00FD	-------	-------	-------	-------
00FE	-------	-------	-------	-------
00FF	-------	-------	-------	-------
SP	-------	-------	-------	-------

```
*** cmd-write-byte******
f86c cmd-write-byte    JSR     getbyte
                       PSHA
                                         <----  Point A (assume A contains
                                                $80)
                       JSR     getword
                       PULA
                       STAA    0,X
                       BRA     cmd_loop

** Subroutine getchar*****
f9b1 getchar           LDAA    SCSR
                       ANDA    #RDRF
                       BEQ     getchar
                       LDAA    SCDR
                       ANDA    #$7f
                       RTS

*** First line of subroutine getbyte*****
f9be getbyte          JSR     getchar
                                         <--- Point D, when getbyte called for
                                              the second time from getword
                                ...

****First part of subroutine getword****
f9d9 getword           JSR     getbyte
                                         <--- Point B
                       PSHA
                                         <--- Point C
                       JSR     getbyte
                       TAB
                              ...
```