UNIVERSITY OF VICTORIA
EXAMINATIONS APRIL 1998

C SC 230 Computer Architecture and Assembly Language

NAME (print): _____ REG NO. _____

SIGNATURE: _____ DURATION: 3 hours

INSTRUCTOR: D. M. Miller

TO BE ANSWERED ON THIS EXAMINATION PAPER.

STUDENTS MUST COUNT THE NUMBER OF PAGES IN THIS EXAMINATION PAPER BEFORE BEGINNING TO WRITE, AND REPORT ANY DISCREPANCY IMMEDIATELY TO THE INVIGILATOR.

THIS EXAMINATION HAS 9 PAGES PLUS THIS COVER PAGE.

ATTEMPT EVERY QUESTION. ANSWER IN THE SPACES PROVIDED (YOU DO NOT NECESSARILY HAVE TO USE ALL LINES PROVIDED AND MAY USE OTHER AREAS ON THE **FRONTS** OF THE PAGES IF NECESSARY. USE THE BACKS OF THE PAGES FOR ROUGH WORK.

THIS IS AN OPEN TEXT BOOK EXAMINATION. YOU MAY REFER TO
H.W. Huang: "MC68HC11: an Introduction" West Publishing, 1996.

NO OTHER AIDS, E.G. COURSE NOTES OR CALCULATORS, ARE PERMITTED.

QUESTION	MAX. MARK	STUDENT'S MARK
1	4	
2	8	
3	15	
4	13	
5	12	
6	13	
7	13	
8	3	
9	3	
10	7	
11	9	
TOTAL	100	

1. (4 marks)
For the single precision IEEE floating point representation for the value -17.5_{10}

What is the value of the sign bit: ____

What is the value stored for the exponent (in decimal): _____

What is the value stored for the mantissa (in binary): _____
(do not show trailing 0's)

2. (8 marks)
Consider the following program:

```
        org     $8000
count   rmb     2
start   lds     #$dfff
        ldd     #512
        std     count
loop    jsr     div4
        fdb     count
        ldd     count
        cpd     #32
        bhs     loop
        jmp     $e000
```

(a) Write the subroutine "div4" in as few instructions as possible so the parameter passed to it is divided by four each time "div4" is executed. This should be an integer division which returns the quotient without rounding. Note that the subroutine uses program memory parameter passing. Your subroutine does not have to protect the registers it uses.

_____

_____

_____

_____

_____

_____

_____

_____

_____

(b) How many times is subroutine "div4" executed? _____

3. (15 marks)
You are to write a complete 68HC11 assembly program that inputs characters one at a time. At the end of each line, i.e. when the user enters a carriage return ($0D), your program should display (on a new line) a message saying whether the line typed by the user is a palindrome or not.

Your program should convert all lower-case letters into upper-case letters before checking if the input line is a palindrome. Assume there is no punctuation, digits or other special characters in the input. Your program is to ignore blanks. There can be no more than 80 characters per line of input. Your program should terminate when the user types an empty line i.e. a carriage return with no characters before it.

Recall that a palindrome is a line that reads the same if the characters are reversed (ignoring case, punctuation and spaces). For example, *Able was I ere I saw Elba* is a palindrome as is *A man a plan a canal Panama.*

Use the monitor I/O routines documented on page 140 - 141 of the text.

4. (13 marks)
Consider the following program:

```
;The sum of the first 5 elements of array arr is placed in location
;labelled "total".
        org     $8000
arr     fcb     121,144,169,75,38,205
total   rmb     2
        org     $9000
        lds     #$dfff
        ldx     #arr
        ldy     #5
        jsr     sum
        std     total
        jmp     $e000

;Subroutine sum receives the number of elements to be totaled in register Y
;and the array address in register X. It leaves the result in register D.
sum     ldd     #0
loop    cpy     #0
        beq     end
        addb    0,x
        adca    #0
        inx
        dey
        bra     loop
end     rts
```

(a) What is the content of accumulator B on exit from subroutine "sum"? _____

(b) What is the content of index register X on exit from subroutine "sum"? _____

(c) The above program passes its parameters, both input and output, via registers. You are to modify it to pass the parameters via the stack. There are to be three parameters which are: (1) the address of the array of values; (2) the number of values to be totaled; (3) the address of the location where the result is to be placed. They are to be put onto the stack in the order given. You are to add the code necessary to remove the parameters from the stack following the jsr.

You are also to add any instructions necessary to ensure all registers used by the subroutine are protected *i.e.* upon return from the subroutine all registers (except the CCR) have the same contents they had when the jsr was executed.

(continued on next page)

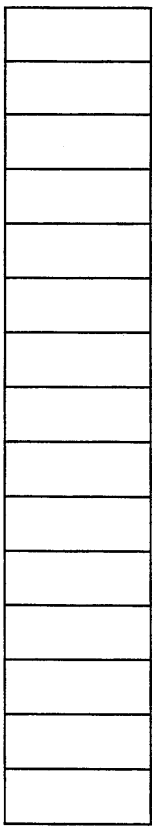
```
        org     $8000
arr     fcb     121,144,169,75,38,205
total   rmb     2
        org     $9000
        lds     #$dfff

        _____________
        _____________
        _____________
        _____________
        _____________
        _____________
        _____________
        jsr     sum
        _____________
        _____________
        _____________
        jmp     $e000
```

;Subroutine sum receives three parameters on the stack: the address of an
;array, the number of values to be totaled and the address of the location
;to hold the result

```
sum     _____________
        _____________
        _____________
        _____________
        _____________
        ldd     #0
        tsy
        ldx     ___,y
loop    tst     ___,y
        beq     end
        addb    0,x
        adca    #0
        inx
        dec     __,y
        ___     loop
end     ldx     __,y
        std     0,x
        _____________
        _____________
        _____________
        _____________
        _____________
        _____________
        rts
```

show the stack frame here



5. (12 marks)

A push-button is connected as an input to IC2. You are to write a subroutine called BCNT which accepts one parameter in accumulator A. The subroutine is to poll IC2 and is to return only after the button has been pressed and released the number of times specified by that parameter. For example, if the subroutine is called with 5 in A, the subroutine will not return until the button has been pressed and released 5 times. Your subroutine is to detect the button release as the external event. Your subroutine should not affect any registers except the CCR. You do not have to worry about de-bouncing the switch.

Show the necessary initialization related to this handling of IC2 in the space provided.

Initialization:

Subroutine:

6. (13 marks)
Frequently, a system has a task that must be executed at a fixed period. That is the situation here for a subroutine called UPDATE. You are to write 68HC11 code so that calls to UPDATE are as close to 5 seconds apart as possible and so that the time between calls never exceeds 5 seconds. Assume the E clock is 2 MHz.

You must use the real-time interrupt and it is your RTI handler that makes the call to UPDATE. Note that UPDATE has no parameters and returns via an RTS instruction with no return result and no registers except the CCR modified. UPDATE does not alter the I flag. You do not have to write any code for UPDATE.

VERY IMPORTANT: UPDATE can take up to 500 milliseconds to execute so you must allow for nested RTI interrupts i.e. an RTI can occur while the previous RTI is still being processed.

Initialization code required:

Real-time interrupt service routine:

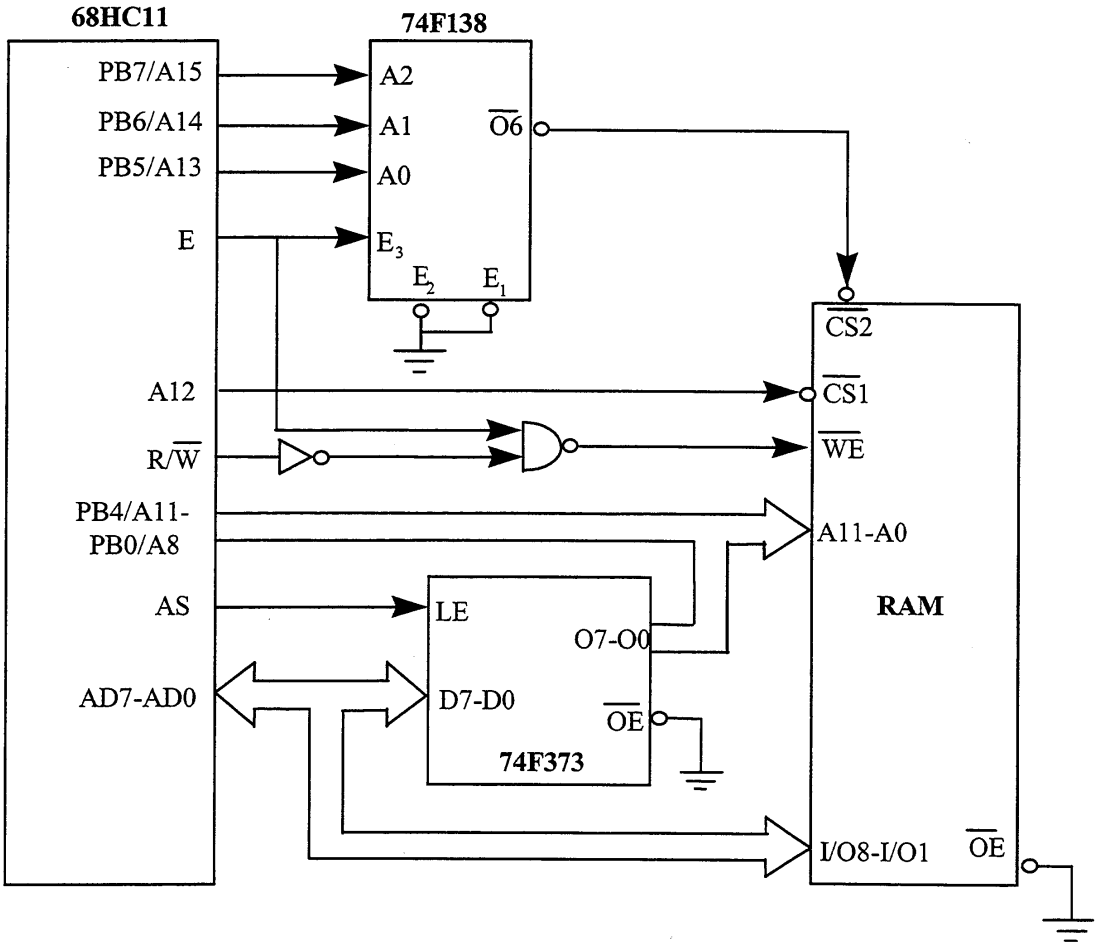
7. (13 marks)
Consider the diagram below (note it is **NOT** identical to the one in the text).

(a) How many bytes of memory does the RAM chip have? _____

(b) What range of addresses does the RAM chip occupy in the given diagram?

_____

(c) Show the changes necessary so that (a) the diagram represents the connection of a RAM chip of twice the number of bytes as the one given and (b) the RAM chip occupies the block of memory beginning at location $E000.



(d) Which component(s) would not be needed if the 68HC11 used had separate address and data busses?

_____

8. (3 marks)
Briefly explain how *branch prediction* works on the PowerPC:

9. (3 marks)
State three significant factors that distinguish *RISC* designs from *CISC* designs:

10. (7 marks)
(a) Briefly explain the difference between *write-through* and *write-back* with respect to cache management.

(b) Briefly explain what is meant by a *set associative cache*.

11. (9 marks)

(a) Briefly explain what is meant by *pipelining* and how it speeds up the execution of machine language instructions.

(b) Why does a conditional branch cause difficulties for pipelining whereas an unconditional branch does not?

(c) What does the term *superscalar* mean?

****** END OF EXAMINATION ******