UNIVERSITY OF VICTORIA
EXAMINATIONS APRIL 1999

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 **TEN** 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 BOOK EXAMINATION. YOU MAY REFER TO THE TEXT MICROPROCESSORS AND MICROCOMPUTERS: HARDWARE AND SOFTWARE (FOURTH EDITION) OR ANY OTHER SINGLE TEXT, AND THE ASSEMBLER AN-D SIMULATOR GUIDES. NO OTHER AIDS, E.G. COURSE NOTES OR CALCULATORS, ARE **PERMITTED**.

QUESTION	MAX. MARK	STUDENT'S MARK
1	10	
2	4	
3	8	
4	15	
5	13	
6	12	
7	16	
8	7	
9	7	
10	6	
11	2	
TOTAL	100	

1. (10 marks) Circle the correct answer for each of the following:

(a) The d-bit two's complement representation of 7_{10} is 11111001~. True False

(b) In 2's complement arithmetic, overflow occurs when the result value is too large for the number of data bits available. True False

(c) A Hamming code allows the detection of up to 3 bit errors and the correction of 1 and 2 bit errors. True False

(d) On the 6811, the external address and data buses are synchronous. True False

(d) Memory-mapped I/O eliminates the need for dedicated I/O instructions on a processor. True False

(f) A processor must have a hardware stack in order to support a jump to subroutine as a single instruction. True False

(g) Unless explicitly programmed by the user, maskable interrupts can not interrupt each other on the 6811. True False

(h) The 6811 is an example of a CISC design. True False

(i) The PENTIUM is an example of a RISC design. True False

(j) The PENTIUM II uses SIMD techniques to implement MMX True False

2. (4 marks) For the single precision IEEE floating point representation for -12.125_{10}

What is the value of the sign bit: ____

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

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

3. (8 marks) Consider the following program:

```
        org   $C00
numb    rmb   2
start   lds   #$FF
        ldd   #1027
        std   numb
loop    ldx   #numb
        pshx
        jsr   div4
        ins
        ins
        Idd   numb
        cpd   #32
        bhs   loop
        stop
```

(continued on next page)

(a) Write the subroutine "div4" 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 accepts one parameter on the stack. Your subroutine should be as efficient as possible. Registers should be protected.

(b) How many times is subroutine “div4” executed by the program above?

4. (15 marks) You are to write a subroutine called CONVERT with no parameters that inputs a decimal number by reading ASCII characters from the keyboard using the monitor routine INCHAR (address $FFCD). Recall that INCHAR returns one character in register A each time it is called. You can assume the user only types decimal digits and a single terminating blank. Your subroutine CONVERT is to return with the value read in register D. For example, if the user typed 257 followed by a blank, then D contain $0101 on return from CONVERT. Your subroutine should protect registers X and Y. The decimal digits have ASCII codes $30 - $39.

5. (13 marks)

Consider the following program:

```
;The sum of the elements of the array "arr" is placed in the location
;labelled "total".
        org     $8000
arr     fcb     121,124,169,85,38
total   rmb     2
        org     $9000
        lds     #$01FF
        ldx     #arr
        ldy     #5
        jsr     sum
        std     total
        stop

;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     Idd     #0
loop    cpy     #0
        beq     end
        addb    0,x
        adca    #0
        inx
        dey
        bra     loop
end     rts
```

(a) What is the content of accumulator A 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 in order 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. Parameter (1) is to be on top. You are to make the additions necessary to the main program and the subroutine.

You are also to add any instructions necessary to ensure all accumulator and index registers used by the subroutine are protected.

(continued on next page)

```
;The sum of the elements of the array "arr" is placed in the location
;labelled "total".
        org     $8000
arr     fcb     121,124,169,85,38
total   rmb     2
        org     $9000
        lds     #$01FF

        _____
        _____
        _____
        _____
        _____
        jsr     sum
        _____
        _____
        _____
        _____
        _____
        _____
        stop

; sum receives three parms on the stack: the address of an array (on top)
; the number of values to be totaled and the address of the location
; to hold the result
sum     _____
        _____
        _____
        _____
        _____
        _____
        _____
        ldd     #0
loop    cpy     #0
        beq     end
        addb    0,x
        adca    #0
        inx
        dey
        bra     loop
end     _____
        _____
        _____
        _____
        _____
        rts
```

6. (12 marks)

A push-button is connected as an input to IC3. You are to write a subroutine that polls IC3 until the button has been pressed (a rising edge on IC3). Once the rising edge occurs, your subroutine is to light a LED at Port A bit 5 for 1 second. After the one second, your subroutine turns off the LED and returns. Your subroutine should protect the accumulator and index registers it uses. Use OC3 for timing the one second. Use polling for a 2MHz processor clock and recall there are 1000 milliseconds in a second. Use the usual control register names such *as TFLG1, TOC2 etc. without defining them.

Show any initialization needed outside the subroutine in the space provided.

Initialization outside subroutine:

Subroutine:

7. (16 marks)

Complete the following stop watch program so it behaves as described. Note it is not the same as the one you did for assignment 5. The routines ZERO and NDISPLAY are not shown for brevity. You do **not** have to write them. Assume the processor has a 2MHz clock.

```
; This is a stop watch program that times in 1/10's of a second.
; The watch is in one of three states:
;      state 0 stopped with the time at 0 (initial state)
;      state 1 running
;      state 2 stopped with the time held at the time it was stopped

; The watch starts in state 0.  A rising edge on IC3 takes the watch
; to the next state in order 0 -> 1 -> 2 -> 0 etc.  So the first IC3
; event starts the watch.  A second one stops it, and a third one
; resets it to 0.

; OC2 interrupts are used for timing.

REGBASE  EQU   $1000
TCNT     EQU   $0E
TIC3     EQU   $14
TOC2     EQU   $18
TCTL2    EQU   $21
TMSK1    EQU   $22
TMSK2    EQU   524
TFLG1    EQU   $23
IC3F     EQU   $01
OC2F     EQU   $40

; Timing control so watch counts in tenths of a second
SLICE    ___ EQU
T I M E C N T  _ E Q U

, Global variables
COUNT   RMB   1     / OC2 INTERRUPT COUNT
TIME     RMB  2     / TIME IN 0.1 SECS
STATE   RMB   1     / WATCH STATE

; Interrupt jump table entries
        _____________________
        _____________________
        _____________________
        _____________________

        ORG   $C000
START   LDS   #$01FF
        LDX   #REGBASE
        CLR   STATE      / WATCH IS INIT STOPPED
        JSR   ZERO       / AND ZEROED
        LDAA  _______    / SET IC3 EDGE TYPE TO RISING
        STAA  _______
        LDAA  _______    / ENABLE IC3 INTERRUPTS
        STAA  _______
                         / ENABLE INTERRUPTS
```

(continued on next *page)*

```
LOOP    LDY   TIME      / DISPLAY TIME
        PSHY
        JSR   NDISPLAY
        INS
        INS
        BRA   LOOP
;===============================================================
; IC3 INTERRUPT

IC3     LDX   #REGBASE
        LDAA  _______     / CLEAR IC3 EVENT FLAG
        STAA  _______
        INC   STATE       / ADVANCE STATE
        LDAA  STATE
        CMPA  #1
        BEQ   IC31
        CMPA  #2
        BEQ   IC32
        CLR   STATE       / -> STATE 0 SO CLEAR TIMER
        JSR   ZERO
        RTI
IC31    L     D    D      / -> STATE 1 SO SET TOC2
        A   D  D   D
        STD   TOC2,X
        BSET  __________  / ENABLE OC2 INTERRUPT
        CLR   COUNT
        RTI
IC32    BCLR TMSK1,X ___ / -> STATE 2 SO DISABLE OC2 INTERRUPT
        RTI
;===============================================================
; Time is kept by counting oc2 interrupts spaced SLICE cycles
; apart.  It takes TIMECNT slices to make 1/10 of a second.
; This handler affects the global variables COUNT and TIME.

oc2     _________________
        _________________

        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________

        RTI
```

8. *(7* marks)

(a) What is the purpose of the start bit in asynchronous serial communication?

(b) What is the purpose of the parity bit in asynchronous serial communication?

(c) Draw the bit pattern that would be sent for the ASCII character X ($58) assuming 7 bits, even parity with one stop bit. Draw vertical lines to separate the bits.

9. *(7* marks)

(a) What is the advantage of having separate fixed-point and floating point processors in a CPU?

(b) Briefly explain how the compiler assists branch prediction on the Power-PC.

10. (6 marks)

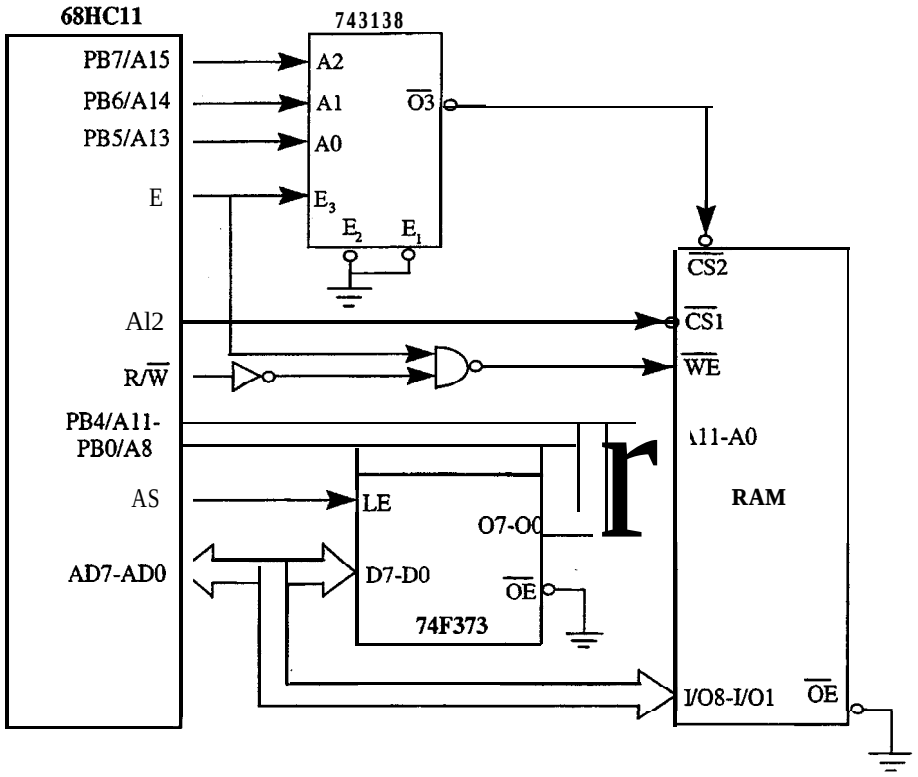
Consider the diagram on the next page (it is not identical to the one shown in class):

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

(b) What range of addresses does the RAh4 chip occupy?

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

(continued on next page)



11. (2) What does the term **superscalar** mean with respect to processor design?

***** End of Examination *****

Note: Course marks to date are posted on the web. Please check your marks and notify Dr. Miller of any errors or omissions by April IP.