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

(a) The 8-bit two's complement representation of 3_{10} is 11111101_2. 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 1 and 2 bit errors and the correction of 1 bit errors. True False

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

(e) On-processor memory can share addresses with off-processor memory. True False

(f) An arithmetic shift preserves the sign of a numeric value. True False

(g) A processor must have a hardware stack in order to support a jump to subroutine. True False

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

(i) The 6811 is an example of a RISC design. True False

(j) The PowerPC is an example of a RISC design. True False

2. (4 marks) For the single precision IEEE floating point representation for -25.25_{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)

3. (8 marks) Consider the following program:

```
        org     $C000
numb    rmb     2
start   lds     #$FF
        ldd     #1024
        std     numb
loop    ldx     #numb
        pshx
        jsr     div8
        ins
        ins
        ldd     numb
        cpd     #32
        bhs     loop
        stop
```

(continued on next page)

(a) Write the subroutine "div8" so the parameter passed to it is divided by eight each time "div8" is executed. This should be an integer division which returns the quotient without rounding. Note that the subroutine accepts one parameter on the stack. Registers need not be protected.

(b) How many times is subroutine "div8" 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 ASCIII 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 255 followed by a blank, then D contain $00FF 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 array the "arr" is placed in the location
;labelled "total".
        org     $8000
arr     fcb     121,144,169,75,38,205
total   rmb     2
        org     $9000
        lds     #$FF
        ldx     #arr
        ldy     #6
        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     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 program memory. 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 in memory in the order given. 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 array the "arr" is placed in the location
;labelled "total".
        org     $8000
arr     fcb     121,144,169,75,38,205
total   rmb     2
        org     $9000
        lds     #$FF

        _____________
        _____________
        jsr     sum
        _____________
        _____________
        _____________
        _____________
        _____________
        _____________
        stop

; sum receives three parms in program memory: 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
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 IC2. You are to write a subroutine called WAIT which accepts one parameter in accumulator A. The subroutine is to poll IC2 until the button has been pressed and released (a falling edge on IC2). Once the falling edge occurs, your subroutine is to wait the number of milliseconds specified by the value in ACCA, maximum value 25. Your subroutine returns after the specified time has elapsed. Your subroutine should protect the accumulator and index registers it uses. Use OC2 for timing the required delay. Use polling for OC2. Assume 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 and the project. 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 1/100'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   $24
TFLG1    EQU   $23
IC3F     EQU   $01
OC2F     EQU   $40

; Timing control so watch counts in hundredths of a second
SLICE    EQU   _______
TIMECNT  EQU   _______

; 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   $C000
START    LDS   #$FF
         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    LDD   _______      / -> STATE 1 SO SET TOC2
        ADDD  _______
        STD   TOC2,X
        BSET  ___________  / ENABLE OC2 INTERRUPT
        CLR   COUNT
        RTI
IC32    BCLR  TMSK1,X $40  / -> STATE 2 SO DISABLE OC2 INTERRUPT
        RTI
;=============================================================
; Time is kept by counting oc2 interrupts spaced SLICE cycles
; apart.  It takes TIMECNT slices to make 1/100 of a second.
; This handler affects the global variables COUNT and TIME.

OC2     _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        _________________
        RTI
```

8. (9 marks)
   (a) What is the purpose of the start bit in asynchronous serial communication?

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

   (c) Draw the bit pattern that would be sent for the ASCII character W ($57) assuming 7 bits, even parity with one stop bit. Draw vertical lines to separate the bits.

9. (6 marks)
   Briefly explain three significant differences between RISC and CISC designs:

10. (7 marks)
    (a) What is the advantage of having separate fixed-point and floating point processors on the Power-PC?

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

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

Happy Holidays