	<- Your section (F01 for the 9:30 class; F02 for the 10:30 class.)

FAMILY NAME	GIVEN NAME	STUDENT ID

UNIVERSITY OF VICTORIA

EXAMINATION DECEMBER 1997

CSC230 COMPUTER ARCHITECTURE AND ASSEMBLY LANGUAGE
INSTRUCTOR: M.H. VAN EMDEN SECTIONS F01 AND F02

PLEASE WRITE YOUR SECTION NUMBER, NAMES AND STUDENT ID AT THE TOP OF THIS SHEET AS INDICATED.

PLEASE WRITE YOUR ANSWERS ON THE PAPER. DURATION: THREE HOURS

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

THIS EXAMINATION PAPER HAS 10 PAGES IN ADDITION TO THIS COVER PAGE.

ATTEMPT EVERY QUESTION. WRITE YOUR ANSWERS IN THE SPACES PROVIDED. USE THE BACKS OF THE PAGES FOR ROUGH WORK.

THIS IS AN OPEN-BOOK EXAMINATION: YOU MAY USE THE TEXT, THE LAB MANUAL, NOTES, AND A CALCULATOR.

Question #		Out of
1		5
2		5
3		10
4		10
5		10
6		10
7		10
8		15
9		25
Total		100

CSC 230
F01, F02

Question 1. (5 marks)

In the program below, determine whether, in the places indicated, the carry flag is set and whether the overflow flag is set. Write your answers in the space provided.

```
P    equ $ff
Q    equ $81
R    equ $01
S    equ $7f

     ldaa #P
     adda #P
;(a) Carry flag set? _____ Overflow flag set? _____

     ldaa #Q
     adda #Q
;(b) Carry flag set? _____ Overflow flag set? _____

     ldaa #R
     adda #R
;(c) Carry flag set? _____ Overflow flag set? _____

     ldaa #Q
     adda #S
;(d) Carry flag set? _____ Overflow flag set? _____

     ldaa #R
     adda #S
;(e) Carry flag set? _____ Overflow flag set? _____
```

Question 2 (5 marks)

How many microseconds does it take to execute the following code segment (assume a 2 MHz clock):

```
        bset TMSK2,x $10
        cli
        ldd TCNT,x
lab     addd #OC2DLY
        std TOC2,x
        ldaa #CLEAR
        staa TFLG2,x
```

Question 3 (10 marks)

Consider the following code segment:

```
                                    ins
                                    ins
                                    ins
                                    ins
```

(a) How many cycles does it take to execute?

(b) How many bytes does it occupy?

(c) Can you find an instruction sequence that has the same effect on the stack pointer that is faster? If so, show it in the space below.

Question 4 (10 marks)

The program segment below is intended to place the value of TCNT in the accumulators A and B.

```
;Assume the PR1 and PR0 bits of the TMSK1 register are 00.

REGBAS  equ $1000     ;base address of I/O register block
TCNTH   equ $0e       ;offset of TCNTH from REGBAS
TCNTL   equ $0f       ;offset of TCNTL from REGBAS

     ldx #REGBAS
     ldaa TCNTH,x     ;copy upper byte of main timer to A
     ldab TCNTL,x     ;copy lower byte of main timer to B
```

(a)
What values will be in accumulators A and B after the above three instructions if TCNT contains $21FF when the ldx instruction is completed?

(b)
Modify the above program so that it correctly obtains the contents of TCNT.

Question 5 (10 marks)

Complete the following instructions so that they clear the OC4F flag bit.

(a)

```
ldx  #$1000
ldaa #$_______
staa ___,x
```

(b)
This flag could also have been cleared by keeping the first instruction and by replacing the second and third by a single instruction. Show the result below.

```
ldx  #$1000

_____________________________
```

Question 6 (10 marks)

Consider a computer with six-bit addresses. It is equipped with a direct-mapped cache with eight slots. Successive memory references are to addresses $3f, $03, $2a, $2b, $2c, $00, $22, $34, $0f, and $2a.

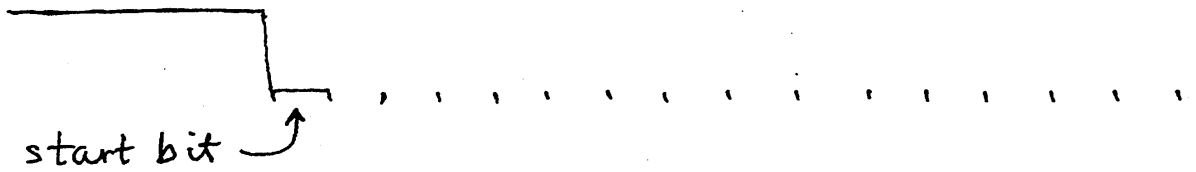
Show the state of the cache by filling out the table below:

Slot	Validity bit	Tag	Content
000			
001			
010			
011			
100			
101			
110			
111			

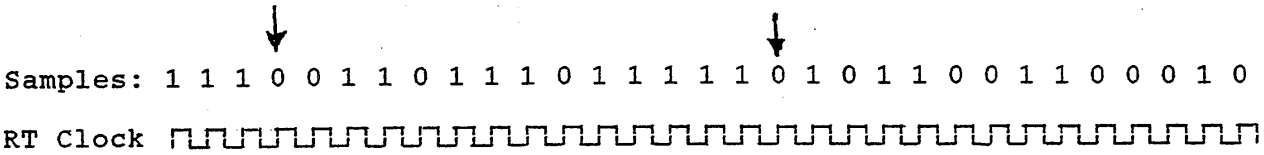
Question 7 (10 marks)

(a)
The letter q (hex code 71) is to be transmitted in the dataframe shown below by the SCI subsystem in the format with 7 databits, one parity bit, and 2 stop bits.

Complete the data frame.



(b) A stop bit has been identified by the SCI subsystem. The samples immediately following the stop bit are the following:



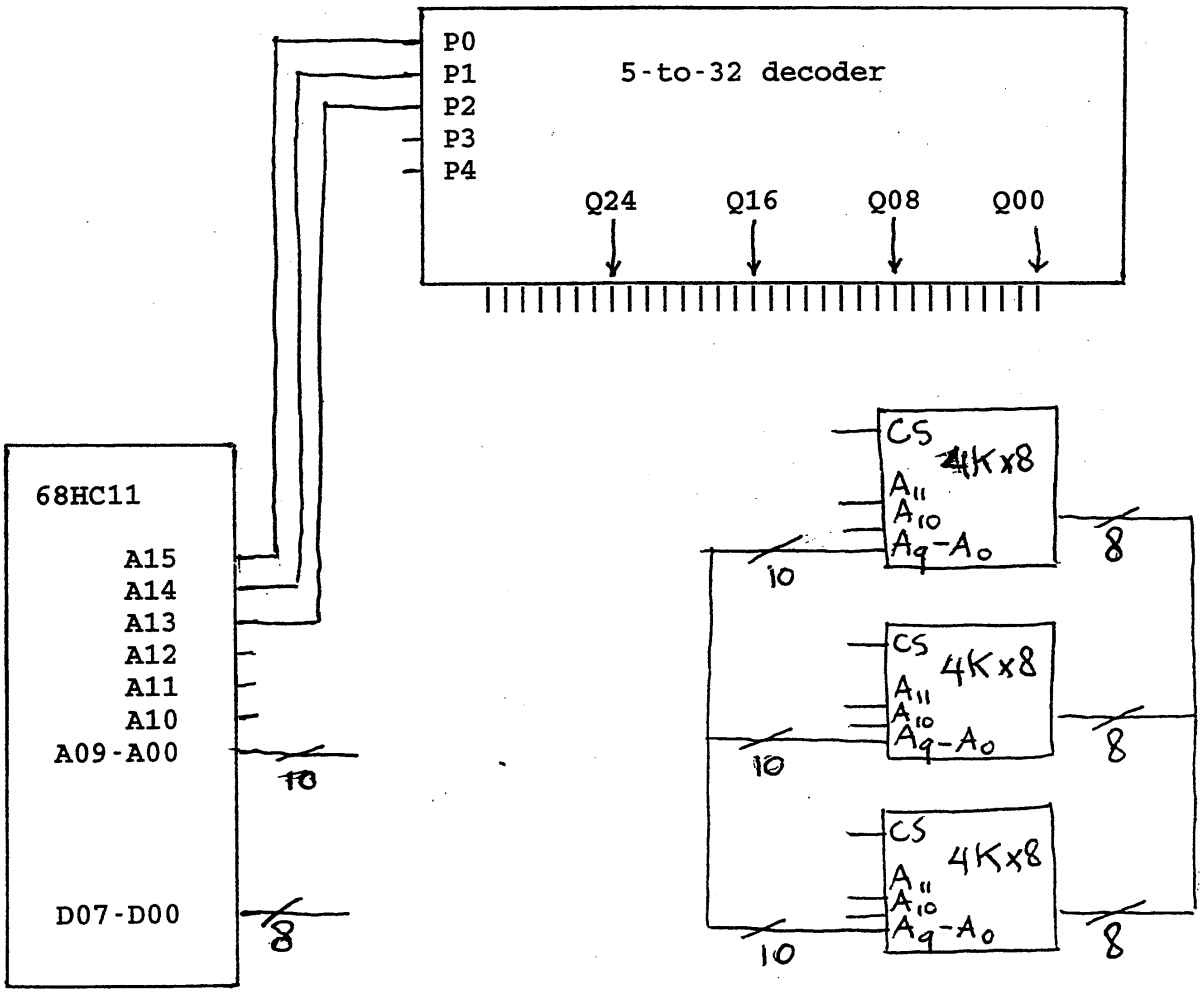
For each of the two zeros indicated by an arrow in the above samples, determine if it is the starting sample of a start bit. If it is not, explain why not.

Question 8 (15 marks)

The 68HC11 is being equipped with three memory modules, each of which contains 4K bytes. In addition to a 16-bit address bus, there is a separate 8-bit data bus.

A 5-to-32 decoder must be used because no smaller one is available.' The inputs to the decoder have been labelled P0,...,P4. The outputs of the decoder are labelled Q00,...,Q31. All 32 output pins are shown, but because of space limitations only the labels Q24, Q16, Q08, and Q00 have been printed.

Complete the diagram below in such a way that the three address blocks starting at addresses $3000, $4000, and $7000 are implemented. (You don't have to know about partial decoding. If you do and you wonder whether you should use it, the answer is: no.)



Question 9 (25 marks)

Below you find a listing of the multitasking template where the source of interrupts is the real time interrupt. In this listing, make the following modifications:

(a) In the current version the first process to execute is process 1. Change the code so that process 3 is the first to execute.

(b) Make OC2 the source of interrupts instead of the real time interrupt. Make the time interval between interrupts equal to 25 milliseconds.

(c) In the given listing, all processes have identical code, and this code is unnecessarily duplicated for each process. Indicate in the listing below how you would modify it so that all three processes share the code that is now used by process 1. Cross out all lines that become superfluous.

Change as little as possible to meet the above requirements.

```
-----------------------------------------
MAX     equ 10000       ; max num times to increment sum
REGBAS  equ $1000
TMSK2   equ $24         ;offset of TMSK2 from REGBAS
TFLG2   equ $25         ;offset of TFLG2 from REGBAS
RTIF    equ $40         ;mask to select RTI enable bit

;data, process chain, process code, process stacks
        org $8000
;application data
sum     rmb 2           ;to be incremented many times
;process chain
sys     rmb 2           ;variable for system stack pointer
curr    rmb 2           ;variable for current process
;should = proc1, proc2, or proc3
proc1   fdb proc2       ;stack pointers for three processes ...
        rmb 2
proc2   fdb proc3
        rmb 2
proc3   fdb proc1
        rmb 2           ;... circularly chained together
;start of code for process 1
start1  ldx #MAX        ;increments sum MAX times
start1b cpx #$0000
        beq start1a
        ldy sum
        iny
        sty sum
        dex
        bra start1b
start1a bra start1a     ;loops forever
        rmb $300        ;room for stack for process 1
```

```
sp1     rmb 1
;stack pointer sp1 points to one address lower than top of stack
        rmb 9           ;room for interrupt stack frame of 9 bytes
;start of code for process 2
start2  ldx #MAX        ;increments sum MAX times
start2b cpx #$0000
        beq start2a
        ldy sum
        iny
        sty sum
        dex
        bra start2b
start2a bra start2a     ;loops forever
        rmb $300        ;room for stack for process 2
sp2     rmb 1
;stack pointer sp2 points to one address lower than top of stack
        rmb 9           ;room for interrupt stack frame of 9 bytes
;start of code for process 3
start3  ldx #MAX        ;increments sum MAX times
start3b cpx #$0000
        beq start3a
        ldy sum
        iny
        sty sum
        dex
        bra start3b
start3a bra start3a     ;loops forever
        rmb $300        ;room for stack for process 3
sp3     rmb 1
;stack pointer sp3 points to one address lower than top of stack
        rmb 9           ;room for interrupt stack frame of 9 bytes
enddata rmb 1
;Unused last byte to place label. Check if address less than $9000

        org $00eb       ;install mtk ...
        jmp mtk         ;... as handler for RTI

;main program
        org $9000
        sei             ;disable interrupts
        lds #$dfff      ;set SP to system stack
        ldx #0
        stx sum
        ldx #proc1      ;execution starts ...
        stx curr        ;... with process 1

        ldy #sp1        ;set up ...
        ldx #proc1
        sty 2,x
        ldy #sp2
        ldx #proc2
        sty 2,x
```

```
        ldy #sp3
        ldx #proc3
        sty 2,x         ;... process chain
;initialize stack frame for process 1
        ldab #$80       ;value for CCR. I bit is clear.
        ldx #sp1
        stab 1,x        ;offset is CCR location in process stack frame
        ldy #start1
        sty 8,x         ;offset is PC location in process stack frame
;initialize stack frame for process 2
        ldab #$80       ;value for CCR. I bit is clear.
        ldx #sp2
        stab 1,x        ;offset is CCR location in process stack frame
        ldy #start2
        sty 8,x         ;offset is PC location in process stack frame
;initialize stack frame for process 3
        ldab #$80       ;value for CCR. I bit is clear.
        ldx #sp3
        stab 1,x        ;offset is CCR location in process stack frame
        ldy #start3
        sty 8,x         ;offset is PC location in process stack frame

        ldx #REGBAS     ;enable ...
        ldaa #RTIF
        staa TMSK2,x    ;... Real Time Interrupt

        jmp disp        ;dispatch process

;Interrupt service routine for Real Time interrupt

mtk     tsx
        dex             ;IX holds stack pointer of interrupted process
        ldy curr        ;IY is proc1, proc2, or proc3
        stx 2,y
;stack pointer of interrupted process stored in process chain
        lds sys         ;SP points to systems stack
;kernel can do some managing here; is free to use stack
        ldx curr        ;IX holds one of proc1, proc2, or proc3
        ldx 0,x         ;IX holds next of proc2, proc3, or proc1
        stx curr        ;curr points to next process
disp    tsx             ;save system ...
        dex
        stx sys         ;... stack pointer

        ldx #REGBAS     ;clear ...
        ldaa #RTIF
        staa TFLG2,x    ;... the RTI flag

        ldx curr        ;load stack pointer from ...
        lds 2,x         ;... location in process chain pointed to by curr
        rti
```

- END -