UNIVERSITY OF VICTORIA
EXAMINATIONS APRIL 1995
COMPUTER ARCHITECTURE AND ASSEMBLY LANGUAGE
C SC 230 - S01

NAME:_____ **REG. NO.** _____

DURATION : 3 hours

INSTRUCTOR: D. Wessels

TO BE ANSWERED ON THE 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 QUESTION PAPER HAS 10 PAGES OF QUESTIONS PLUS THE COVER PAGE AND 2 PAGES OF APPENDICES: 13 PAGES TOTAL

INSTRUCTIONS:
All answers are to be written on the paper. The exam is not open book. However, you are allowed one sheet (8.5" x 11") of notes, double sided, and the Motorola reference card to assist you. **ANSWER ANY NINE OF THE TEN QUESTIONS.** If you answer all 10 questions your best 9 marks will be totalled. Familiarity with THINK© C and MAS BasePak is assumed.

Introduction and notes.
Please be precise but brief, and use point form for written answers; you are not expected to write essays and the time might be better spent on other questions. I strongly recommend you read the entire exam through from beginning to end before beginning to answer the questions.

The appendix pages at the back of the exam may be removed, please try to leave the rest of the exam stapled together however.

ANWER ANY 9 OF THE 10 QUESTIONS TOTAL MARKS: 180

1	2	3	4	5	6	7	8	9	10	TOTAL
20	20	20	20	20	20	20	20	20	20	180

Question 1 [20 marks]
[15] (i) Provide brief but descriptive comments for the following MC68000 assembly language program (do **not** simply echo the instructions!).

```
        xref    decin,stop

start:  lea     list,a0

        jsr     decin

        move.w  d0,d1

        subi.w  #1,d1

        asl.w   #1,d1

        lea     0(a0,d1.w),a0

loop:   cmpi.w  #10,d0

        bge     cont

        move.w  2(a0),(a0)+

        addi.w  #1,d0

        bra     loop

cont:   jsr     stop

        data
list:   ds.w    10
        end
```

[5] (ii) Describe in a few short sentences the effect of the above program.

Question 2 [20 marks]
Suppose we are given an MC68000 processor that runs at 25 MHz. That is, each clock cycle takes 40 nanoseconds. A bus cycle takes 4 clock cycles to complete, and each instruction requires 2 clock cycles to execute in addition to any required bus cycles.

Using this information, plus the number of reads and writes required during the fetch/decode/execute cycle, compare the execution times of the following two code segments.

```
Segment1:  clr.l   d1
           lea     array,a0
           move.l  #9,d0
loop:      add.l   (a0)+,d1
           dbra    d1,loop


Segment2:  move.l  #0,d1
           lea     array,a0
           move.l  #0,d0
loop:      add.l   (a0),d1
           adda.l  #4,a0
           addi.l  #1,d0
           cmpi.l  #10,d0
           blt     loop
```

Question 3 [20 marks]
Translate the following C program into a complete MC68000 assembly language program.

```
void  main()
{
    int i;
    int  array[20];

    i = 1;
    array[0] = 1;
    while (i < 20) {
        array[i] = 2 * array[i-1];
        if ((i % 2) == 0) printf("%ld \n", array[i]);
        i = i+1;
    }
}
```

Note: "%" is the C modulo operator, i.e. x % 4 gives the remainder of x / 4.

Question 4 [20 marks]

[10] (i) Translate the following MC68000 instruction into machine code, including all extension words if any are needed. (The relevant reference pages from the Ford and Topp appendices are provided in appendix 1 of the exam.)

MOVE.W 4(A3),-(A2)

[10] (ii) Translate all the extension words for the following instructions (you do not need to translate the opcode words).

(a) LEA 8(A3),A1

(b) ADDI.L #-4,$0FC00FC

(c) SUBA.B $D4,A3

(d) MOVE.L -2(A4),-(A4)

(e) ADD.L (A0)+,D4

Question 5 [20 marks]
Suppose we are given the following set of subroutine calling conventions:

- first, parameters are pushed on the stack in "right-to-left" order
- next, the number of parameters is pushed on the stack as a long integer (not counting the return value in the number of parameters)
- next, one longword of space is pushed on the stack for the return value (this space is pushed even if the subroutine does not return anything)
- the calling routine is responsible for cleaning the stack once the subroutine call completes

[15] (i) Provide the MC68000 assembly language code necessary to implement the following subroutine call using the specified conventions, given that a, b, and y are long integers. You are only required to provide code for the calling routine, not the foo subroutine.

```
y = foo(a, b)
```

[5] (ii) Draw and clearly label a picture of the stack at entrance to the foo subroutine described above (i.e. after the jsr instruction completes but prior to any linking within the foo subroutine).

Question 6 [20 marks]

[10] (i) In a typical two-pass assembler, what types of error checking are typically **not** performed on the first pass and why?

[10] (ii) In the Macintosh, labels are translated into operands using the indirect-with-offset addressing mode using either the PC or A5 as the base register. In some other MC68000 systems, labels are translated into absolute addresses using the PLC. If position independent code is desired, briefly what are the ramifications of these two approaches with respect to linking and loading?

Question 7 [20 marks]

[8] (i) What is the main difference between statically position independent code and dynamically relocatable code?

[12] (ii) Assuming TABLE is coded as an offset from the program counter, the following code segment is statically position independent. Show that it is **not**, however, dynamically relocatable.

```
        CLR.L    D1
        MOVEQ    #3,D0
        LEA      TABLE,A0
FOO:    ADD.W    (A0)+,D1
        DBRA     D0,FOO
```

Question 8 [20 marks]
Suppose we are given a system in which memory is partitioned into pages of one kilobyte each, and there is a memory management unit which uses a memory map (as discussed in class) to translate 16 bit logical addresses into 12 bit physical addresses.

[3] (i) How many logical pages of memory are there?

[3] (ii) How many of these pages are resident in physical memory?

[14] (iii) Provide the current memory map given that all the following holds at the moment. (You only need to list the entries for pages that are currently in physical memory.)

logical address $A8F4 corresponds to physical address $0F4
logical address $AA00 corresponds to physical address $200
logical address $ABA3 corresponds to physical address $BA3
logical address $0E48 corresponds to physical address $E48
logical address $84A3 corresponds to physical address $4A3
logical address $A848 corresponds to physical address $848
logical address $0C00 corresponds to physical address $C00
logical address $87F4 corresponds to physical address $7F4

Question 9 [20 marks]

[10] (i) Describe the basic communication protocol for the MC68000 read word cycle, clearly giving the values on the upper and lower data strobes, address strobe, data acknowledgement line, read/write line, as well as indicating the status of data on the data and address buses and the function control lines.

[10] (ii) Suppose there is a DMA (direct memory access) unit capable of controlling the bus in an MC68000 system. Given the three signal lines

BG	(bus grant, goes from the CPU to the DMA)
BR	(bus request, goes from the DMA to the CPU)
BGA	(bus grant acknowledge, goes from the DMA to the CPU)

try modifying the protocol of part (i) above to provide a protocol for exchanging control of the bus between the CPU and the DMA.

Question 10 [20 marks]

[3] (i) What is the significance of a signal being asserted on the BERR pin in the midst of the fetch/decode/execute cycle?

[3] (ii) What is the significance of a signal being asserted on the BERR pin in the midst of the interrupt acknowledgement cycle?

[5] (iii) Describe the main differences between internal and external interrupts with respect to detecting and identifying the interrupts.

[9] (iv) In the MC68000 processor, no instructions are implemented (in hardware) with opcode words of the form $Axxx or $Fxxx. How might an assembler and special service routine be used to take advantage of this to "add instructions" to the basic MC68000 instruction set?

-- THE END --

Appendix 1

MOVE — Move Data from Source to Destination

Operation : Source , Destination

Assembler Syntax : **MOVE <ea>, <ea>**

Attributes : Size = (Byte, Word, Long)

Description : Moves the content of the source to the destination location. The data is examined as it is moved, and the condition codes set accordingly. The size of the operation may be specified as byte, word, or long.

Condition Codes :

X	N	Z	V	C
-	*	*	0	0

- **N** Set if the result is negative. Cleared otherwise.
- **Z** Set if result is zero. Cleared otherwise.
- **V** Always cleared.
- **C** Always cleared.
- **X** Not affected.

Instruction Format :

15	14	13 12	11 10 9	8 7 6	5 4 3	2 1 0
0	0	Size	Destination Register	Destination Mode	Source Mode	Source Register

Instruction Fields :

Size field - Specifies the size of the operand to be moved:

- 01 - byte operation
- 11 - word operation
- 10 - long operation

Destination Effective Address field - Specifies the destination location. Only data alterable addressing modes are allowed as shown :

Addr. Mode	Mode	Register
Dn	000	reg. number:Dn
An	-	-
(An)	010	reg. number:An
(An)+	011	reg. number:An
-(An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An

Addr. Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	-	-
(d_{16},PC)	-	-
(d_8,PC,Xn)	-	-

Source Effective Address field - Specifies the source operand. All addressing modes are allowed as shown:

Addr. Mode	Mode	Register
Dn	000	reg. number:Dn
An*	001	reg. number:An
(An)	010	reg. number:An
(An)+	011	reg. number:An
-(An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An

Addr. Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d16,PC)	111	010
(d8,PC,Xn)	111	011

* For byte size operation, address register direct is not allowed.

Notes : 1. MOVEA is used when the destination is an address register. Most assemblers automatically make this distinction.
2. MOVEQ can also be used for certain operations on data registers.