

**UNIVERSITY OF VICTORIA****EXAMINATIONS APRIL 2001****C SC 230 - Computer Architecture and Assembly Language (S01)**

NAME: \_\_\_\_\_ (Print)

REG. NUMBER \_\_\_\_\_

TIME: 3 hours

INSTRUCTOR: M. Serra

TOTAL MARKS: 130

**TO BE ANSWERED IN THE PAPER**

Question No.	Value	Mark	Question No.	Value	Mark
1	6		7	12	
2	9		8	20	
3	20		9	12	
4	10		10	18	
5	10		<b>TOTAL</b>	130	
6	13				

**INSTRUCTIONS:**

1. STUDENTS MUST COUNT THE NUMBER OF PAGES IN THIS EXAMINATION PAPER BEFORE BEGINNING TO WRITE, AND REPORT ANY DISCREPANCY IMMEDIATELY TO THE INVIGILATOR
2. The examination paper consists of 12 pages.
3. The exam is not open book. However, you are given, together with this paper, an Appendix with details necessary to answer the questions.
4. Please be precise but brief, and use point form.
5. It is strongly recommended that you read the entire exam through from beginning to end before starting to answer the questions.
6. Maximum number of marks available is 130 - part marks are available on all questions. The marks assigned to each question are printed within square brackets.

April 2001

Dept. of Computer Science - Univ. of Victoria

**Question 1 [6].** Match each Assembler Directive with its meaning:

Directive	Match to	Meaning	
ORG		Form constant byte	A
EQU		Form double byte constant	B
RMB		Form constant character string	C
FCB		Equate symbol to value	D
FCC		Reserve memory bytes	E
FDB		Set program counter to origin	F

**Question 2 [9].** The code below shows the differences in loading addresses or contents of variables. State exactly what is the effect of each instruction (see first example). This code segment was posted on the Web page in the examples. If an instruction refers to the *address* of a variable, please state both the address in hexadecimal (you know it from the memory allocation directives), and the semantics (e.g. does it load the address or the content of a variable?).

```

LIST      ORG      $0000
LIST2     FDB      $A1,$B2,$C3,$D4,$E5
I         FCB      $A6,$B7,$C8,$D9
J         RMB      1
TEMP      RMB      1
TEMP2     EQU      $AA
TEMP3     EQU      $BB
TEMP3     EQU      $ABCD

```

```

ORG      $C000

```

**ANSWERS HERE**

```

LDAA     #TEMP      ;ACC A = $AA

```

```

LDX      #TEMP3     ;

```

```

STAA     I          ;

```

```

LDAA     #TEMP2     ;

```

```

STAA     J          ;

```

```

LDX      I          ;

```

```

LDX      #I         ;

```

```

LDX      LIST       ;

```

```

LDX      #LIST      ;

```

```

LDAA     LIST2      ;

```

April 2001

Dept. of Computer Science - Univ. of Victoria

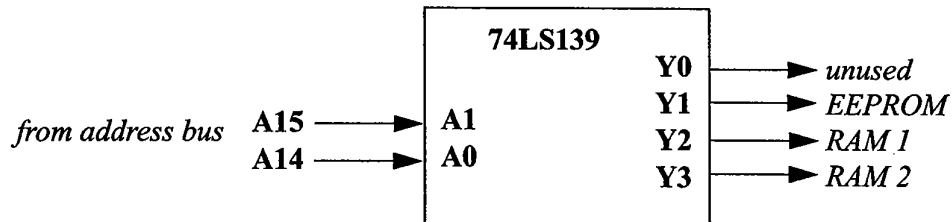
**Question 3 [20].** Write structured MC68HC11 code for the following design given in pseudo-code:

```
IF R = S
  THEN
    WHILE T < W
      DO
        DECREMENT W
        R = 2 + R
      END DO
    END WHILE
  ELSE
    R = 2 + S
  END IF
```

Assume R, S, T, and W are 8-bit unsigned binary numbers, that storage has been allocated in the program by the following code, and that R, S, T and W are initialized to some value in some other part of the program:

R	RMB 1	T	RMB 1
S	RMB 1	W	RMB 1

**Question 4 [10].** Below is the schematic design of a decoder used to connect the CPU and external memory, in this case 1 EEPROM chip and 2 RAM chips, and to provide appropriate addressing. By examining the diagram, state:



(a) what are the address spaces allocated to each component? Give the range of addresses, in hexadecimal, including any unused area.

unused: \_\_\_\_\_

EEPROM: \_\_\_\_\_

RAM1: \_\_\_\_\_

RAM2: \_\_\_\_\_

(b) how big is the address space allocated to each component? (Note that it is the same for each).

\_\_\_\_\_

**Question 5 [10].**

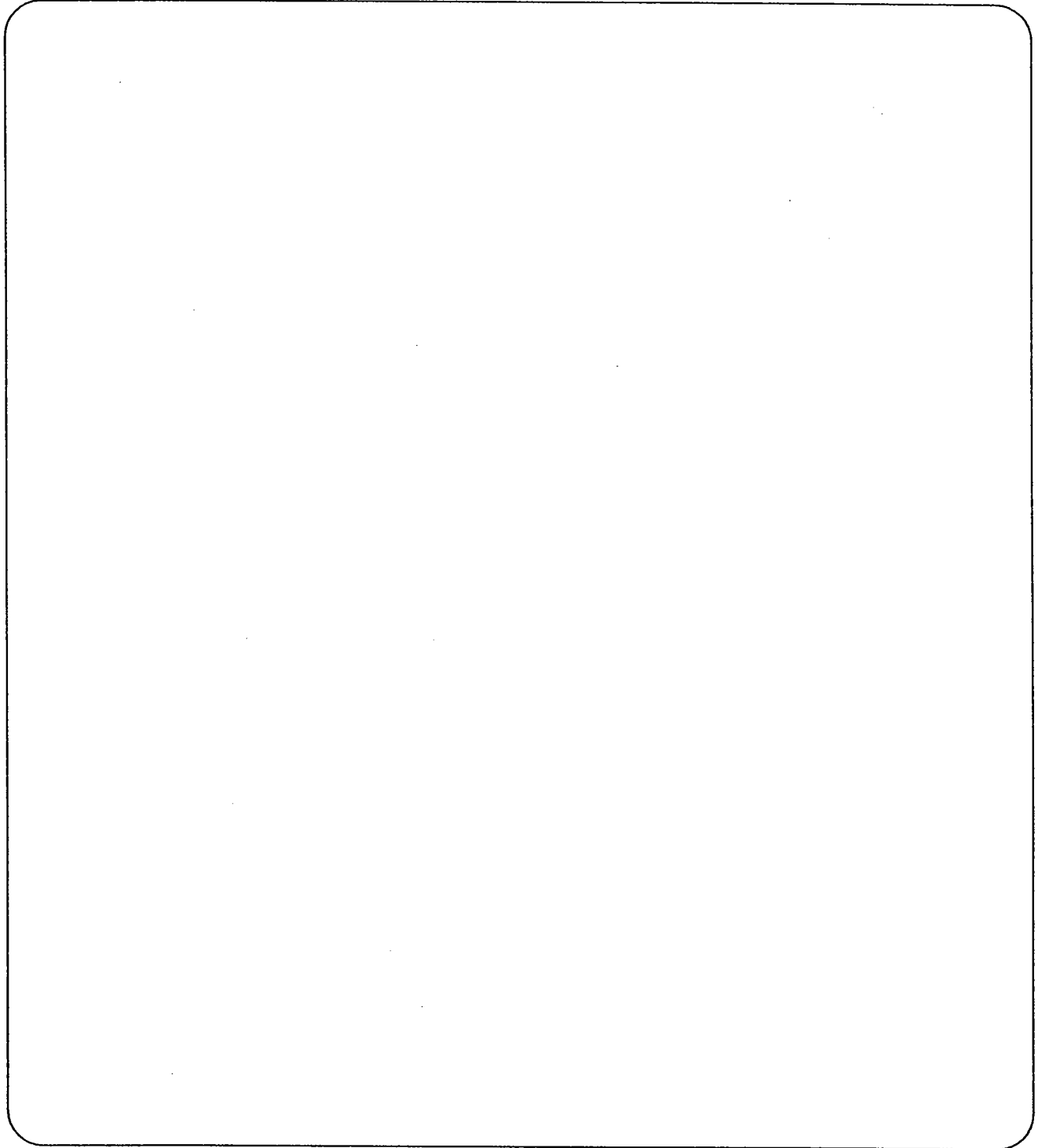
(a) Assuming a 2's complement representation, convert the following numbers from decimal to 8-bit binary and to its equivalent hexadecimal:

Decimal	Binary	Hexadecimal
25 <sub>10</sub>		
-26 <sub>10</sub>		

(b) Given the 4-bit hexadecimal number, state its decimal equivalent according to the assumption of its representation listed in each heading:

Hexadecimal	Unsigned	2's complement	Signed magnitude
A <sub>16</sub>			
6 <sub>16</sub>			

**Question 6 [13].** Give a picture of the general organization of a microprocessor-based system. In particular show registers, memory, various elements of the CPU, bus(es), peripherals, external communications, etc. Assume either memory- mapped I/O or isolated I/O and state which. Be clear, precise and complete. Show the main blocks only, do not go overboard on details, yet label everything accurately.



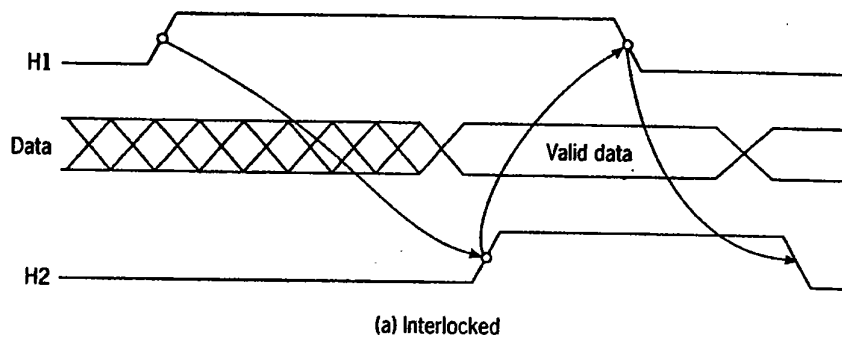
**Question 7 [12].**

(a) [2] Describe the main function of a DMA.

(b) [2] State what peripheral interfaces are.

(c) [2] State two main features of a RISC Architecture.

(d) [6] You have seen the picture below in class. It describes the timing events for an *input handshaking protocol* using *interlocked mode*. Describe the protocol in point form, following the diagram.



**Question 8 [20].**

You must set up the templates in Assembly Language for calling functions and returning from functions in a C program to help your friendly compiler writer. The general conventions are as follows:

- All parameters are to be passed through the stack.
- All parameters are to be passed by reference (i.e. their address).
- Storage is also created on the stack for whatever local variables are declared inside the function.

**TEMPLATE (C Calling Conventions).**

The caller pushes the arguments in right-to left order, then calls the function. When the function returns, it is the caller's responsibility to remove the arguments from the stack. After pushing the input arguments, but before issuing the actual call, the caller pushes the *address* of the location where the return value is to be placed. The function must copy this address and store the result at the location it points to. The address is considered a hidden argument to the function and it is the caller's responsibility to remove it eventually from the stack.

Use the pseudo-code below for a general function as an example for the template, so it can be marked more accurately.

```

FUNCTION TEST (R,S: INTEGER) -> T: INTEGER; {2 input parameters of type integer,
                                             one return parameter of type integer}

    VAR T,W: INTEGER; {locals to function TEST}
    {.....
    code
    assume here that somewhere accumulator A will contain the return value}
    T = return value;
END;
```

In the high level language, the caller issues something like:

```
Q = TEST ( K,J)
```

You must provide a template for the conventions in both the calling routine and the called function. The template must include:

- (i) for the calling routine:
  - (a) the instructions for pushing all parameters on the stack
  - (b) the calling instruction
  - (c) the instructions, after the function returns, to retrieve the function result
- (ii) in the called function:
  - (a) the instructions to allocate and deallocate local storage on the stack
  - (b) the instructions to save and restore local registers
  - (c) the instructions inside the function to copy the parameters from the stack into local registers
- (iii) the instructions to clean up the stack wherever appropriate (in caller or function called)
- (v) comment your code if you want me to understand what you are doing and give you any marks at all
- (vi) integers are assumed to be 8-bit long (only here!) with 16-bit addresses.

April 2001

Dept. of Computer Science - Univ. of Victoria



**Template for caller**

**Template for function TEST**

**April 2001**

**Dept. of Computer Science - Univ. of Victoria**

**Question 9 [12].**

The following code was discussed in the labs. Fill in the missing parts.

; Example IO\_32.txt

;Use the input capture to measure the period (in clock cycles) of a signal  
;connected to IC2. Store the result in memory locations \$C500 and \$C501.  
;Use the input capture interrupt capability.

-----

**; TIMER AND PORT DEFINITIONS**

```
REGBASE    EQU    $1000
PORTA      EQU    $0
TCTL2      EQU    $21
TMSK1      EQU    $22
TIC2       EQU    $12
TFLG1      EQU    $23
```

-----

**; CONTROL DEFINITIONS**

```
IC2R       EQU    %_____ ; SET FOR RISING EGDE
IC2        EQU    %_____ ; IC2F IN TFLG1
```

-----

**; PROGRAM VARIABLES**

```
SBASE      EQU    $1FF
```

-----

**; GENERAL VARIABLES**

```
PERIOD     EQU    $C500
COUNT     RMB    1                ; WHICH EDGE?
```

-----

**; INTERRUPT JUMP TABLE ENTRY**

```
ORG _____
```

```
JMP _____
```

-----

**; MAIN PROGRAM**

```
MAIN       ORG    $C000
           LDS    #SBASE
           LDY    #REGBASE
```

April 2001

Dept. of Computer Science - Univ. of Victoria

```

;-----
; INITIALIZATION

START      SEI                                ; DISABLE INTERRUPTS
           LDAA _____                    ; CLEAR FLAG IC2F
           STAA _____
           BSET TMSK1,Y IC2                  ; ENABLE IC3 INT
           LDAA _____                    ; SET IC2 RISING EDGE
           STAA _____
           CLR  COUNT
           CLI                                ; ENABLE INTERRUPTS
LOOP       BRA  LOOP                          ; DO ANYTHING HERE

;-----
; IC2 INTERRUPT HANDLER

IC2INT     LDAA _____                    ; CLEAR EVENT FLAG!
           STAA _____
           TST  COUNT
           BNE  SECOND
           LDD  _____                    ; LOAD TIME OF CAPTURE
           STD  PERIOD                        ; AND STORE IT
           INC  COUNT
           BRA  DONE
SECOND     LDD  _____                    ; LOAD TIME OF CAPTURE
           SUBD PERIOD                        ; CALCULATE PERIOD
           STD  PERIOD
           CLR  COUNT
DONE       RTI                                ; RETURN FROM INTERRUPT
           END

```

**Question 10 [18]. Answer True or False:**

You feel confident that you are able to design and implement a traffic light controller	
Interrupts are automatically masked when one gets to the interrupt service routine in the MC68HC11	
When the MC68HC11 processes an interrupt service routine all registers, except the CCR, are saved on the stack	
The SEI instruction is used to globally unmask interrupts in the MC68HC11	
The Jump Table location for the address of the interrupt service routine for TOC3 is \$00D9 in the MC68HC11	
XIRQ and IRQ have the same priority in the MC68HC11	
The CCR contains bits to define interrupt priorities in the MC68HC11	
If the TOF bit is set when the timer interrupts are enabled and interrupts unmasked, an interrupt will occur immediately in the MC68HC11	
The TOI bit in the TMSK2 register is set when the TCNT register overflows in the MC68HC11	
The IC2F bit in register TFLG1 is set when Pin 2 of Port A receives a signal transition and the appropriate input capture edge has been programmed in TCTL2 in the MC68HC11	
The single precision IEEE floating point standard format has different representations for +0 and -0.	
In 2's complement addition, overflow can only occur when adding 2 large negative numbers.	
If TCTL2 contains '00000110' it implies that IC1 has its capture disabled, IC2 will capture on a rising edge, and IC3 will capture on a falling edge in the MC68HC11	
RISC processor are designed with few instructions in the instruction set and many registers	
The MC68HC11 is a RISC processor	
The timer overflow flag is set when TCNT moves from \$FFFF to \$0000 and it is cleared automatically in the next cycle in the MC68HC11	
The SPARC processor, commonly found in Sun workstations, is a CISC processor.	
When the I bit in the condition code register is set to 1, interrupts are enabled in the MC68HC11	

T H E E N D

April 2001

Dept. of Computer Science - Univ. of Victoria