Carleton University
Department of Systems and Computer Engineering
SYSC-2003 Midterm 2, Winter 2010

Name : _____ **Mark :** _____ / 45

Student Number : _____

Interpretation of the exam text is a part of the evaluation. Do not ask questions unless you believe there is an error. Anything you need: make an assumption and write it clearly.

Tip: in all programming exercises, you can earn part marks by writing the skeleton of the program even if you can't write all the logic.

Tip 2: the points given to each exercise should be used as a hint on how much time to spend on them; do not spend long time for exercises with few marks (your answers there are supposed to be brief).

Question 1 [14 marks] Arrays, Subroutines, C programming, Recoup exercise for Midterm 1

Consider the following C program, which reads two vectors (*v1* and *v2*) of a given SIZE, and computes the dot product of the two of them, storing the results in the *result* vector (reminder: Dot Product definition: if a = [a1, a2, ... , an] and b = [b1, b2, ... , bn], then a.b = a1.b1+a2.b2+...+an.bn).

```
#define SIZE 10

void main(void) {

      char v1[SIZE], v2[SIZE];   // elements are 8-bit long
      int i=0, OK, result;

      // Fill in v1 and v2 with data using the read_data() function

      for (i=0; i<SIZE-1; i++)  {
            v1[i] = read_data();// reads an integer value from a file
      }

      for (i=0; i<SIZE-1; i++)      // a different way to fill in v2;
            v2[i] = read_data();

      OK = dot_product(v1, v2, result, SIZE);
      // execute the dot product of the two vectors
}
```

a) [4 marks] The function

```
unsigned char dot_product (char &array1, char &array2, int &result,
                  char SizeOfArrays);
```

takes two input arrays (each containing 8-bit elements), computes the dot product, and stores the result into the `result` variable. The function also returns a Boolean value (1 if there was an overflow, 0 if the computation was completed without problems). The argument `SizeOfArrays` defines the size of the arrays. The function uses 2 local variables (*int mul* for intermediate multiplications, and *unsigned char count*, used as a counter). **Draw the C Stack Frame for this subroutine.**

b) [10 marks] Write the code of the subroutine in part a) into Assembly (based on the Stack Frame you wrote in part a). Be very careful in using the C policies for the subroutine. Hint: you will obtain partial marks even if you do not remember all the details of the C Policies (2 marks for C Policies).

Question 2 [11 marks] General concepts about Interrupts

a) [2 marks] What is a LOCAL Maskable interrupt? What is the difference with a GLOBAL mask? How do you enable/disable LOCAL maskable interrupt?

b) [3 mark] What is a non-maskable interrupt? How it is detected? What is the difference with a maskable interrupt?

_____

_____

_____

_____

c) [6 marks] Complete the steps of the interrupt service cycle (if you do not remember exactly, the idea is for you to explain the important concepts of this hardware mechanism):

1. Save CPU status in the stack. The following registers MUST be saved:

   _____
   (order is NOT important)

2. Globally disable other maskable interrupts (Set the I mask)

3. _____

4. _____

5. Execute the ISR

If you need, use this space to draw a diagram on how this works, using the numbers to identify the different steps (non-mandatory; if you need to draw a picture to explain yourself better, do it here).

Question 3 [20 marks] Programming Parallel I/O and the Interrupts

a) [4 marks] Port K is organized as follows:

Port K	7	6	5	4	3	2	1	0
	x	x	relay	buzzer	LED4	LED3	LED2	LED1

Write a program in C to **turn on all the LEDs (and then off), one at a time, and in decreasing order**. That is, the sequence is: turn on Led4, Led3+4, Led2+3+4, Led1+2+3+4; then, you turn them off in the opposite order: Led1+2+3+4, Led2+3+4, Led3+4, Led4, all OFF). Once you turn on a led, it remains on until you turn it off in the second cycle. Do not introduce any delays (do not worry about being able to see the leds on this exercise).

b) [5 marks] Complete the following code for a RTI (Real-Time Interrupt):

```
      ___________      ; Install Timer ISR on an Axiom Board
                             ; That uses NOICE (HARDWARE
                               ;  Interrupt Address: $FFF0)
      FDB  ___________

      org $1000        ; On the Axiom board

numInterrupts    db $01
dummy            db $00


      org $4000
main:
      lds #$3f80             ; setup stack pointer
      bset CRGINT,$80        ; ____________________________
      bset RTICTL,$7F        ; Prescale Factor = 16 x 2^16
     _________               ; globally enable interrupts

      // HERE YOU HAVE TO COMPLETE EXERCISE E) (*)
      bra main

Timer_ISR:
      inc dummy
      ldaa #8
      cmpa dummy
      bne return
      movb #0, dummy
      inc numInterrupts

return:
      bset CRGFLG, $80     ; Acknowledge interrupt
      ___________
```

c) [2 marks] Explain why we use a dummy counter, and what is it used for. Why do we need this counter?

_____

_____

_____

_____

d) [3 marks] How often is *numInterrupts* incremented? Explain TWO ways to do this TWICE AS FAST.

e) [6 marks total] The program in part b) contains a line with a comment marked as (*). We want to call a function called *display_time*. This function will receive *numInterrupts* (passed as an argument), and will compute and display the number of Minutes and Seconds from the start of the program (ignore Hours; every hour, the Minutes are reset to 0).

i) [2 marks] Write the Assembly Code to call this subroutine. The Subroutine will be written in C.

ii) [4 marks] Write the C code for *display_time*.