Carleton University
Department of Systems and Computer Engineering
SYSC-2003 Assignment 2, Winter 2012

Due Date: February 5th, at 7:00 pm (electronic submission)

You must submit the files identified below <u>using the electronic Submit application</u>. The submission process will be canceled at the deadline. No assignments will be accepted via email or on disk.

Write the following programs in Assembly, assemble them, and run them step by step in the simulator. Unless stated otherwise, use 16-bit integer numbers. The solution should end up in the specified register(s), memory or stack.

EVERY PROGRAM IN THIS ASSIGNMENT MUST END USING THE BRA * INSTRUCTION. COMMENT YOUR SOURCE CODE EXHAUSTIVELY.

Exercises 1 – 3 and 7 MUST first be developed on the Simulator, but the version you submit MUST RUN IN THE BOARD (i.e., you should run test them in the board after testing them in the simulator). If you do not succeed in running the program on the board, submit a version that runs on the simulator. We encourage you to run the rest of the exercises on the board too. CLEARLY IDENTIFY with a big comment at the beginning of your file, if your program runs on the board or the simulator.

In most embedded platforms, the microcontroller boards are not provided with display devices. Being unable to print strings during the execution of a program is inconvenient. The HC12 board we are using is provided with a Monitor program with debugging and printing facilities that make use of the serial communication channels of the PC and the board to send strings through the communication lines.

Programming these communication devices is complex, hence, you will be provided with a source file (*sci0api.asm*, found in the RESOURCES webpage, in the *include* folder) that must be <u>included</u> in your program (see how to do this the template included as an example). The file contains the implementation of two subroutines (do not worry about subroutines details yet).

void setbaud (register int baudRate)
```
;Sets up SCI0 with baud rate specified in register D
; Must be called once, before any prints
;Note: Simulator automatically matches to any baud rate
BAUD9600    EQU   0052   ;9600 Divider  (8MHz ECLK) Used for Mon12
BAUD19K     EQU   0026   ;19200 Divider (8MHz ECLK) Used for NoICE
```

putChar_sc0:
```
; Takes a character in B and displays it
```

You must use the subroutine to display the final result that must be stored on register B for every exercise (the result will be interpreted as ASCII, unless you use the Bonus exercise first). Print the message to the PC. Rude messages will result in a zero for the assignment. When you display the results, you will see the character corresponding to the ASCII value of your result.

1. [3 marks] (**assign21.asm; assign21.s19**) Write a program that computes Z = (Y + 7) - X. The result must be stored in register B, and you must display the results using the subroutines before, as explained. The three numbers must be in fixed memory locations, and they must be initialized as follows: Z=0; Y=45; X=4.

2. [4 marks] (**assign22.asm; assign22.s19**) Write a program that computes the following formula: Z = (X - 2Y) * 3. Here, X and Y are 8-bit unsigned integers. The result of Z should be stored in register D. The numbers must be originally stored in fixed memory locations. Submit a version where the initial data is: X = 20, Y = 2 (test it with different values; we will check the correctness of the program with various values).

3. [6 marks] (**assign23.asm; assign23.s19**) Write a program to compute the first 10 elements in the Fibonacci series. The numbers should be stored in the D register. (further details at http://en.wikipedia.org/wiki/Fibonacci_number). Test it with different values, but submit a version that stops after the first 10 numbers.

4. [6 marks] (**assign24.asm; assign24.s19**) Write a program that takes a 16 bit variable, with the following value:

   VAR = 1010 1010 0111 1101

   In the first step, toggle all bits. In a second step, count how many 1s do you have.

   You can only use Boolean/shift operators to manipulate this variable in this exercise (you can use other functions for loops, control flow, manipulating other variables, etc.; but VAR data should only be modified using Boolean/Shift).

5. [5 marks] (**assign25.asm; assign25.s19**) Write a program to traverse an array of bytes, and computes the Sum (integer division; ignore the remainder) of the elements in the array. The array ends when you find number 255 (which is not counted). Store the result in D. Submit a version with the following array data:

   Array = { 47, 121, 114, 34, 44, 117, 33, 124, 255 }

6. [6 marks] (**assign26.asm; assign26.s19**) Write a program compare two strings. The strings are named *firstString* and *secondString*, they are 10 bytes long each, and they are defined in your program as a constant value. The program will return a value of zero if the 2 strings are different, and a value of 1 if they are both exactly the same. Display both strings one byte at a time.

7. [BONUS: 5 marks] (**assign27.asm; assign27.s19**) In most exercises above the result obtained is an integer number. Integers cannot be displayed directly: they must be converted to ASCII first.

   a. Write a program that converts integer-to-ascii. The program receives a 16-bit integer (for instance, 255) and converts it into a null-terminated string (for instance, “255”, i.e., the ascii “2” followed by ascii “5” followed by ascii “5”, and terminated by ascii “0”). Your code can be organized as a single program, or it can use subroutines. If you use subroutines, the data can be a global variable, or it can be passed by reference on the stack. THIS PROGRAM MUST RUN ON THE BOARD.

   b. Combine the program in part a. with Exercise 1 above, and show the real result on the terminal. To do so, you can use the following subroutine, also available in *sci0api.asm*:

   void putStr_sc0 (register char[] &message)
   ;Given a start address in Y, outputs string until it finds null

Assignment 2 Marking Criteria (30 marks + 5 bonus):

An assignment will receive zero if it is late OR any of the directions are not followed (**including late submission or lack of comments**).

Assignments will receive marks as per the previous marking scheme, if your code assembles and runs correctly when run by the TA. If your code does not assemble, you will receive a mark of zero for that exercise. In each case, the solution should demonstrate what is being asked for (i.e. it should be a full program with sample data, ready to execute and illustrate your solution).