CS2208b Assignment 4

Issued on: Thursday, March 12, 2020

Due by: 11:55 pm on Thursday, March 19, 2020

For this assignment, only an electronic submission (***attachments***) at owl.uwo.ca is required.

- Attachments must include:
  - o ***ONE pdf*** file (named `report2.pdf`) that has one flowchart.
  - o ***ONE Text*** file (named `question1.s`) that has softcopy of the assembly source program you wrote.
- So, in total, you will submit 1 + 1 = 2 files (`report2.pdf` and `question1.s`)
- **Failure to follow the above format may cost you 10% of the total assignment mark.**

Late assignments are strongly discouraged

- 10% will be deducted from a late assignment (up to 24 hours after the due date/time)
- After 24 hours from the due date/time, late assignments will receive a zero grade.

In this assignment, you will use the *micro Vision ARM simulator* by *Keil*, which is an ***MS Windows***-based software, to develop the required programs in this assignment. The simulator (version 4) has been installed on *all PCs at GEN labs*, ***except*** NCB-105.

The *Keil micro Vision* simulator may also be installed on your Windows PC. You just need to download it from OWL and install it.

Programming Style

The programming style is very important in assembly language. It is expected to do the following in your programs:

- Using the EQU directive to give a symbolic name to a numeric constant to make it more readable.
- Applying neat spacing and code organization:
  - o Assembly language source code should be arranged in three columns: *label*, *instruction*, and *comments*:
    - ▪ the *label* field starts at the beginning of the line,
    - ▪ the *instruction* field (opcodes + operands) starts at the next TAB stop, and
    - ▪ the *comments* are aligned in a column on the right.
- Using appropriate label names.
- Commenting on *each* assembly line
- Commenting on *each* logical part of your code.

Great Ways to Lose Marks

- Not grouping your lines into logical ideas
- Not appropriately using whitespace
- Not bothering to comment your code
- Commenting the code by just stating what you're doing, instead of why, e.g.,
```
MOV r0, #5    ;move 5 into r0
```
- Not paying attention to the programming style (see the previous paragraph)
- Not optimizing your code by using unnecessary assembly instructions. The more instructions in your program, the less your mark will be.
- Handing in your code as soon as it assembles, without testing and validating your code
- Not using proper flowchart symbols
- Not following the flowchart rules

QUESTION 1 (100 marks)

A string is an array representing a sequence of characters. To store a string of *n* characters in your program, you need to set aside *n+1* bytes of memory. This allocated memory will contain the characters in the string, plus one extra special character—the *null* character—to mark the end of the string. The *null* character is a byte whose bits are all zeros (`0x00`). The actual string consists of any group of characters, which none of them can be the *null* character.

Draw a *detailed flowchart* and write an ARM assembly language *program* to copy a *null*-terminated **`STRING1`** to a *null*-terminated **`STRING2`**, after removing any occurrences of the word "***the***" (case sensitive) in **`STRING1`**. I.e., if **`STRING1`** is `“`**`the`**` woman and `**`The`**` man said `**`the`**`”` then **`STRING2`** would become, `“ woman and `**`The`**` man said ”`. However, if **`STRING1`** is `“and `**`the`**`y took brea`**`the`**`”` then **`STRING2`** would become `“and `**`the`**`y took brea`**`the`**`”` without any change. You can assume that **`STRING2`** will be *less than* 128 characters.
Your code should be highly optimized. Use as few instructions as possible (as little as 30 assembly instructions only, NOT including any assembly directives or data definitions)!!.

Define the data of this program in a separate DATA area.

Define the strings as follow:

```
STRING1 DCB "and the man said they must go"   ;String1
EoS     DCB 0x00                              ;end of string1
STRING2 space 0x7F                            ;just allocating 127 bytes
```

`More test cases:`

```
"the the   the 123    the" ➔ "     123    "
"the, the   the 123    the." ➔ "the,     123    the."
"" ➔ ""
"the" ➔ ""
"The" ➔ "The"
"them   the   the1" ➔ "them      the1"
"4the  the 4the  The the the1" ➔ "4the   4the  The  the1"
```