CS2208a Assignment 3

Issued on: Monday, October 28, 2013

Due by: 11:55 pm on Tuesday, November 5, 2013

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

- Attachments must include:
  - ***ONE pdf*** file that has the two flowcharts, program documentations, and any related communication.
  - ***Text*** soft copy of the assembly programs that you wrote for each question (*one program attachment per question*), i.e., ***TWO assembly*** program files in total.
- So, in total, you will submit 1 + 2 = 3 files.
- **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 MC-342 and MC-08 labs.

The simulator may also be installed on your PC. If you have not installed *Keil micro Vision 4 simulator* on your PC yet, you will need to download and install it from https://www.keil.com/download/product/

Note that during this month (October 2013), *Keil* has released a new version of its *MDK-ARM simulator* (*Keil micro Vision 5*). The *MDK-5* installation process has changed from a single monolithic installer to a set of installers.

So, you need to download *MDK-Core Version 5* from https://www.keil.com/download/product/ as well as the *Legacy support for ARM7 & ARM9 devices* from http://www2.keil.com/mdk5/legacy

The author of our text book has provided a *Quick Guide to Using the Keil ARM Simulator*. If you wish, you can access it at http://www.alanclements.org/usingkeilsimulator.html

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

- Using macros for the constants in your program to make it more readable.
- Applying neat spacing and code organization:
  - 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 each assembly line

Great Ways to Lose Marks

- Not grouping your lines into logical ideas
- Not using any whitespace at all
- Not bothering to comment
- 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)
- Handing it in as soon as it assembles without testing and/or trying to break your code

QUESTION 1 (50 marks)

0
13800
15073
8

Most goods sold in U.S. and Canadian stores are marked with a Universal Product Code (UPC). The meanings of the digits underneath the bar code (from left to right) are:

- First digit: type of item,
- First group of five digits: manufacturer,
- Second group of five digits: product, and
- Final digit: check digit, used to help identify an error in the preceding digits.

To compute the check digit,

- Add the first, third, fifth, seventh, ninth, and eleventh digits
- Add the second, fourth, sixth, eighth, and tenth digits
- Multiply the first sum by 3 and add it to the second sum
- The check digit is the digit which, when added to the above sum, produces a sum that is multiple of 10
- Subtract 1 from the total
- Compute the remainder when the adjusted total is divided by 10
- Subtract the remainder from 9

Example for UPC 0 13800 15073 8:

- First sum: 0 + 3 + 0 + 1 + 0 + 3 = 7
- Second sum: 1 + 8 + 0 + 5 + 7 = 21
- Multiplying the first sum by 3 and adding the second yields 42
- Subtracting 1 gives 41
- Remainder upon dividing by 10 is 1
- Remainder is subtracted from 9
- Result is 8

Draw a *detailed flowchart* and write an ARM assembly language program to determine whether a string of 12 ASCII encoded digits stored in memory is a valid UPC or not. If valid, you should store 1 in R0, if not, you should store 0 in R0. **You code should be highly optimized, i.e., use as few number of instructions as possible.**

You may want to define the UPC string as follow:

```
UPC   DCB "013800150738"   ;UPC string
EoS   DCB 0x00             ;end of string
```

To test your program, you can use the following UPCs:

```
      0 60383 75557 7
      0 65633 45471 2
```

You can also get more UPC code from your own kitchen items.

HINT 1: You can implement the division operation using repeated subtraction.
HINT 2: To calculate 3 × Z, you can do so using only one ADD instruction with LSL#1 shift.
HINT 3: To load a byte to a register, use `LDRB` not `LDR`.

QUESTION 2 (50 marks)

Draw a *detailed flowchart* and write an ARM assembly language program to determine whether a string of *printable* ASCII encoded characters stored in memory is a palindrome (i.e., the letters in the string are the same from left to right as from right to left) or not. If palindrome, you should store 1 in R0, if not, you should store 0 in R0. **You code should be highly optimized, i.e., use as few number of instructions as possible.**

Ignore all *characters* that are not *letters*. You should also treat capital and small letters the same. For example, "*madam*", "*deleveled*", "*Noon*", "*He lived as a devil, eh?*", and "*Was it a car or a cat I saw?*" are palindrome strings. However, "*madam, I am Adam.*" is not a palindrome string.

A string can have *even* or *odd* number of characters and ends with character `0x00` (the ASCII code of the null character).

You may want to define the UPC string as follow:

```
UPC   DCB "He lived as a devil, eh?" ;UPC string
EoS   DCB 0x00                       ;end of string
```