Carleton University

Department of Systems and Computer Engineering

SYSC 3006 (Computer Organization) summer 2020

Lab / Assignment 5 – Answers file

Student Name: ID#:

Part 1 – Fragment 1 [3-mark/5]

Questions about Fragment1 SRC.txt (included in lab 5.zip)

1. [0.25-mark] What is the high-level objective (purpose) of the code fragment? Explain the objective in terms of the net effect of the fragment on the variables it modifies.
   The fragment adds 10 to each element in the array (whose base address is labelled by Arr).

2. [0.25-mark] Write C-like pseudocode that accomplishes the same objective (see lab statement for more details)
   For (R3 = (Arr_Size -1); R3 >= 0; R3--) { Arr[(R3)] += 10; } /* Arr is the address of first array element, and R3 is an index into the array (from Arr + 2 to Arr + 0) */

3. [1-mark] Starting after the SkipOverVariables declaration, add comments to the instructions that document what is being done ... the comments should be at the level of the pseudocode objective, not at the RTL level. For example, consider the instruction:

MOV R4, # 1

An RTL-level comment for the instruction might be: "; move #1 into R4", which is accurate but says nothing about the net programming objective (i.e. why is loading #1 useful in the context of the program's objective?). A more appropriate comment might be: "; R4 = address of Arr_Size".

```
B SkipOverVariables

       ; Arr is an array of 3 words
Arr_Size DCD #3
Arr
   DCD #20 ; first (0 - th)element of Arr = 20
   DCD #-4; second (1 - th)element of Arr = -4
   DCD #0 ; third (2 - th)element of Arr = 0

SkipOverVariables
MOV R2, Arr             ; R2 = address of Arr (0 - th element).
LDR R3, [ Arr_Size ]    ; R3 = 3 (content at address Arr_Size).
CMP R3, #0              ; Is the array empty? ...
BEQ Done                ; ... If yes, then end the program.
SUB R3, R3, #1          ; Subtract 1 from array size in R3.
                        ; This is done in anticipation of
                        ; offset addressing later.

Loop                    ; Starting a for loop
LDR R5, [R2, R3 ]       ; R5 = contents at beginning (base) of array,
                        ; plus offset.
                        ; This initially will give the last element in
                        ; the array.
ADD R5, R5, #10         ; Add ten to this element ...
STR R5, [R2, R3]        ; ... then store at the same location.
SUB R3, R3, #1          ; Decrement the loop index.
BPL Loop                ; Do the same to the previous element in
array,
                      ; but only if the result of subtraction is >=
0.

Done
   DCD #0xFFFFFFFF ; breakpoint instruction
```

4. [0. 5-mark] When the fragment is executed, how many instructions will be executed (including the breakpoint instruction)?

   22? Nope, 21 because the breakpoint instruction (#0xFFFFFFFF) is only fetched not executed:
   - 1 instruction for initial branch
   - 5 instructions for SkipOverVariables procedure
   - 5 instructions x 3 iterations of for-loop
   - 0 instructions for breakpoint (fetched, but never executed).

TOTAL = 1 + 5 + 5x3 + 0 = **21 instructions executed**

5. [0.5-mark] When assembled, how many words of memory will the fragment occupy?
   The assembled Fragment1 requires 16 words of memory (this includes variables and instructions), 1 initial B + 5 DCD + 10 instructions = 16 words.

6. [0.5-mark]Assemble and run Fragment 1. To validate running the fragment, submit here after the contents of Main Memory RAM before and after executing the fragment. (Hint: right-click on RAM Save Image ...).
   Before execution:

```
v2.0 raw
80F00004 00000003 00000014 FFFFFFFC 00000000 23200002
333FFFFA 57300000 80100006 22330001 32523000 2155000A
36523000 22330001 806FFFFB FFFFFFFF
```

   After execution:

```
v2.0 raw
80f00004 00000003 0000001e 00000006 0000000a 23200002
333ffffa 57300000 80100006 22330001 32523000 2155000a
36523000 22330001 806ffffb ffffffff
```

   Note: Saved RAM image does not add zeros to the left. Here I did extend zeros to the left manually for better representation. Student should not be penalized for not doing so.

Part 2 – Fragment 2 [2-mark/5]

1. [1.5-mark] complete the code by replacing all occurrences of "***" with the necessary details and execute the processing for the data values in the template. Do not add additional instructions. Submit your completed (working) SRC fragment. This part of the lab will be easier to complete in the lab if some options for the "***" entries have been considered prior to arriving for the lab.

```
    B  SkipOverVariables

    ; Arr is an array of 5 words
Arr_Size     DCD #5
Arr
             DCD #3 ; first (0 - th) element of Arr
             DCD #-4
             DCD #0
             DCD #-8
             DCD #6

SkipOverVariables

; for ( R11 = 0; R11 < Arr_size; R11++ )
; R10 = Arr_Size
             LDR R10, [ Arr_Size ]
             MOV R11, #0 ; R11 is index into array, index = 0

for_test  ; test whether to enter loop
             CMP R11, R10
             BHS end_for     ; if fail test, then finished for loop

; {  ; start of for loop body
; if ( Arr[ R11 ] < 0 )
; for access to Arr: R9 = address of Arr

             MOV R9, Arr
             LDR R5, [ R9 , R11 ] ; R5 = Arr[ R11 ]
             CMP R5, #0
             BGE end_if

; { Arr[ R11 ] = abs( Arr[ R11 ] ) ; abs is absolute value
; need value 0 for calculating abs
             MOV R6, #0 ; R6 = 0
             SUB R5, R6, R5 ; initial value in R5 is negative: R5 = 0 - R5 = abs( R5
             )
             STR R5, [ R9, R11 ] ; store Arr[ R11 ]
; }
end_if

; } ; end of for loop body
; adjust Arr index
             ADD R11, R11, #1
             B for_test      ;[or BAL]
end_for
    DCD #0xFFFFFFFF ; breakpoint instruction
```

2. [0.5-mark] Assemble and run Fragment 2. To validate running the fragment, submit here after the contents of Main Memory RAM before and after executing the fragment. (Hint: right-click on RAM → Save Image ...)..

Before execution:

```
v2.0 raw
80F00006 00000005 00000003 FFFFFFFC 00000000 FFFFFFF8
00000006 33AFFFF9 23B00000 47BA0000 80300009 23900002
3259B000 57500000 80B00003 23600000 02565000 3659B000
21BB0001 80FFFFF5 FFFFFFFF
```

After execution:

```
v2.0 raw
80f00006 00000005 00000003 00000004 00000000 00000008
00000006 33AFFFF9 23B00000 47BA0000 80300009 23900002
3259B000 57500000 80B00003 23600000 02565000 3659B000
21BB0001 80FFFFF5 FFFFFFFF
```

Note: Saved RAM image does not add zeros to the left. Here I did extend zeros to the left manually for better representation. Student should not be penalized for not doing so.

Submission deadline

Must be submitted on cuLearn, locate (Assignment 5 submission) and follow instructions. Submission exact deadline (date and time) is displayed clearly within the Assignment 5 submission on cuLearn.

Note: If you have any question please contact your respective group TA (see TA / group information posted on cuLearn) or use Discord class server.

Good Luck