Carleton University

Department of Systems and Computer Engineering

SYSC 3006 (Computer Organization) summer 2020

Lab / Assignment 6 – Answers file

Student Name: ID#:

Part 1 – Fragment 1 [2-mark/5]

1. [2-mark] Complete the Fragment1SRC code by replacing all occurrences of "***" with the necessary details. Do not add any additional instructions. Then Submit your completed (working) Fragment1SRC code.

```
     B   Main
ExampleString           ; start of packed string
       DCD    #0x53595343   ; 'S'=0x53    'Y'=0x59    'S'=0x53    'C'=0x43
       DCD    #0x20333030
       DCD    #0x36210000   ; 1st 00 is the null-terminator, 2nd 00 is just padding to fill the word
Main

; R0 = address of the packed string (LEA Notes Towards Software P87 or slide 250 )
  LEA    R0, [ ExampleString ]
; R1 = index of the character in the packed string ... test with several values
  MOV    R1, #5

; the following code puts the indexed character into the least significant byte of R0,
;    and clears the remainder of R0

; ----   start of block to copy into  Fragment 2   ----

; R2 = offset to word containing indexed character
  LSR    R2, R1, #2         ; divide index by 4
; R3 = word containing indexed character
  LDR    R3, [ R0, R2 ]

; R1 = digit offset of indexed character in R3  (digit offset = 0, 1, 2 or 3)
  AND    R1, R1, #3

; need to shift character from current position to least significant byte
;   need to shift 8 bits for each digit position
; R1 = number of digit shifts needed
  MOV    R4, #3
  SUB    R1, R4, R1
  BEQ    DoneShiftLoop    ; if 0 shifts needed, then done shifting
; shift character into least significant byte
ShiftLoop
  LSR    R3, R3, #8
  SUB    R1, R1, #1
  BNE    ShiftLoop
DoneShiftLoop
; R3 now has indexed character in least significant byte,
;   but may have additional characters in higher bits
; R0 = indexed character with higher bits cleared
  AND    R0, R3, #0xFF ; or #255

; ----   end of block to copy into  Fragment 2   ----
  DCD  #0xFFFFFFFF         ; stop
```

Part 2 – Fragment 2 [3-mark/5]

1. [2.5-mark] Complete the Fragment2SRC code by replacing all occurrences of "***" with the necessary details. Do not add any additional instructions. Then Submit your completed (working) Fragment2SRC code

```
    MOV R13, #0x800       ; initialize stack pointer
    B Main

ExampleString        ; start of packed string
      DCD    #0x53595343   ; 'S'=0x53    'Y'=0x59    'S'=0x53    'C'=0x43
      DCD    #0x20333030
      DCD    #0x36210000   ; 1st 00 is the null-terminator, 2nd 00 is just padding to fill the word

;  char   CharAt ( &( PackedString[] ), uint charIndex  )
;  accepts    R0 = address of PackedString[] (by reference),
;             R1 = charIndex (by value)
;  returns    R0 = PackedString[ charIndex ]
CharAt
  PUSH { R1,R2,R3,R4, R14}    ; save registers (SYSC 3006 Register Preservation convention)

; ----   start of block copied from Fragment 1   ----

; R2 = offset to word containing indexed character
  LSR   R2, R1, #2    ; divide index by 4
; R3 = word containing indexed character
  LDR   R3, [ R0, R2 ]

; R1 = digit offset of indexed character in R3  (digit offset = 0, 1, 2 or 3)
  AND   R1, R1, #3

; need to shift character from current position to least significant byte
;   need to shift 8 bits for each digit position
; R1 = number of digit shifts needed
  MOV   R4, #3
  SUB   R1, R4, R1
  BEQ   DoneShiftLoop   ; if 0 shifts needed, then done shifting
; shift character into least significant byte
ShiftLoop
  LSR   R3, R3, #8
  SUB   R1, R1, #1
  BNE   ShiftLoop
DoneShiftLoop
; R3 now has indexed character in least significant byte,
;   but may have additional characters in higher bits
; R0 = indexed character with higher bits cleared
  AND   R0, R3, #0xFF ; or #255

; ----   end of block copied from Fragment 1   ----

  POP { R1,R2,R3,R4, R15 } ;restore registers and return (SYSC 3006 Register Preservation convention)


Main

; R2 will be used as a character index into ExampleString
; R2 = 0
  MOV  R2, #0

; for ( R0 = CharAt( &(ExampleString), R2 ); R0 != null; R0 = CharAt( &(ExampleString), ++R2 ) )
TestFor
  ; set up to call CharAt
```

```
  ; get address of ExampleString
        LEA  R0, [ ExampleString ]
  ; get character index
    MOV  R1, R2

    BL   CharAt    ; R0 = CharAt( &(ExampleString), R2 )

         CMP  R0, #0    ; if character = null ... then done for loop
         BEQ  DoneFor

    ADD  R2, R2, #1
    B    TestFor   ; get and test next char

DoneFor
; at this point, R2 contains the number of characters in ExampleString

  DCD  #0xFFFFFFFF        ; stop
```

2. [0.25-mark] Describe the difference between the Bcc and BLcc instructions, and why the difference is important.

   BLcc is used to call subroutines, it saves the return address in R14 (the Link Register) before branching (R14 <- R15).

3. [0.25-mark] Briefly describe the programming conventions associated with subroutines that are used in this course.

   R0 – R3 will be used to pass parameter arguments during invocation. R0 will be used for the first (leftmost) parameter, R1 for the second parameter, etc. Using only 4 (max.) registers to pass arguments limits the number of parameters that a subroutine can have. This course will not be concerned with how to pass more than 4 arguments (it would involve using some memory to store the extra arguments). R0 will also be used to return the return value.

Must be submitted on cuLearn, locate (Assignment 6 submission) and follow instructions. Submission exact deadline (date and time) is displayed clearly within the Assignment 6 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