Date: 11/14/2021

CSCI-212

ASSIGNMENT-7

Ex 14.2

1.4v @ lowerCase.s
@ prompts user to enter alphabetic character,
@ converts all uppercase to lowercase and
@ shows the result.

@ Defining my Raspberry Pi
.cpu cortex-a53
.fpu neon-fp-armv8
.syntax unified @modern syntax

@ Constant for assembler
.equ nBytes, 50 @amount of memory
@ for string

@ Constant program data
.section .rodata
.align 2
prompt:
.asciz "Enter some alphabetic
characters: "

@ The program
.text
.align 2
.global main
.type main, %function
main:
sub sp, sp, 16 @ space for saving
@ regs

Date: 11 / 14/2021

```
@(keeping 8-byte sp
@ align)
str r4, [sp, -4]!  @ save r4
str fp, [sp, -8]!  @      fp
str lr, [sp, -12]! @      lr
add fp, sp, 12     @ set four frame pointer

mov r0, nBytes  @ get memory from heap
bl malloc
mov r4, r0      @ pointer to new memory

ldr r0, =prompt1Addr  @ prompt user
bl writeStr

mov r0, r4      @ get user input
mov r1, nBytes  @ limit input size
bl readStr

mov r0, r4      @ convert to lowercase
bl toLower

mov r0, r4      @ echo user input
bl writeStr

mov r0, r4      @ free heap memory
bl free

mov r0, 0       @ return 0;
ldr r4, [sp, 4]  @ restore r4
ldr fp, [sp, 8]  @      fp
ldr lr, [sp, 12] @      lr
add sp, sp, 16   @ restore sp
bx lr            @ return
```

P.T.O

Date: 11/14 Tue

prompthdr:
- word prompt

@ tolower.s
@ Converts all alpha characters to lowercase
@ Calling sequence:
@ r0 <- address of string to be written
@ bl tolower
@ returns number of characters written

@ Define my Raspberry Pi
.cpu cortex-a53
.fpu neon-fp-armv8
.syntax unified @ modern syntax

@ Useful source code constants
.equ lowerMask, 0x20
.equ NUL, 0

@ The code
.text
.align 2
.global tolower
.type tolower, %function
tolower:
sub sp, sp, 16 @ space for saving
@ regs
str r4, [sp, 0] @ save r4
str r5, [sp, 4] @ r5
str fp, [sp, 8] @ fp
str lr, [sp, 12] @ lr
add fp, sp, 12 @ set our frame
@ pointer

P_T_0

Date: 11/14/2021

```
    mov r4, r0      @ r4 = string pointer
    mov r5, #0      @ r5 = count
whileLoop:
    ldrb r3, [r4]   @ get a char
    cmp r3, NUL     @ end of string?
    beq allDone     @ yes, all done

    orr r3, r3, lowerMask   @ convert to
                            @ lowercase
    strb r3, [r4]   @ update string

    add r4, r4, 1   @ increment pointer
                    @ var
    add r5, r5, 1   @ count++
    b whileLoop     @ back to top
allDone:
    mov r0, r5      @ return count;
    ldr r4, [sp, 0] @ restore r4
    ldr r5, [sp, 4] @       r5
    ldr fp, [sp, 8] @       fp
    ldr lr, [sp, 12] @      lr
    add sp, sp, 16  @ restore sp
    bx lr           @ return
```

Lx 14.4

1. Ans. It will work if we copy the assembly language code for hexToInt into a file and then assemble it with:

as --gstabs -o hexToInt2.o hexToInt 2.s

Then we gcc to link the object files with the file containing the main function:

gcc -g -o hexConvert hexToInt2.o writeHex.o readIn.o hexConvert.c

Date: 11 / 04 / 2021

2 Ans. It works for both cases because all alphabetic characters are numerically higher than the numeral characters in the ASCII code.

3. Ans.
```
@ hexConvert2.s
@ Prompts user for hex number and converts it
@ to an int.

@ Define my Raspberry Pi
        .cpu    cortex-a53
        .fpu    neon-fp-armv8
        .syntax unified         @ modern syntax

@ Constant for assembler
        .equ    maxChars, 9     @ max input chars
        .equ    theString, -16  @ for input string
        .equ    locals, 16      @ space for local
                                @ vars

@ Constant program data
        .section .rodata
        .align  2
prompt:
        .asciz  "Enter up to 32-bit hex
                 number: "
display:
        .asciz  "The integer is: %i\n"

@ The program
        .text
        .align  2
        .global main
        .type   main, %function
```

Date: 4 / 14 / 2021

main:

```
sub   sp, sp, 8      @space for fp, lr
str   fp, [sp, 0]    @save fp
str   lr, [sp, 4]    @ and lr
add   fp, sp, 4      @set our frame pointer
sub   sp, sp, locals @ for local vars

ldr   r0, promptAddr @prompt user
bl    writeStr

add   r0, fp, theString  @place for user
                         @ input
mov   r1, maxChars   @ limit input size
bl    readStr

add   r0, fp, theString  @ user input
bl    henGetInt          @convert it

mov   r3, r0         @result returned in r0
ldr   r0, displayAddr @ show user the
                      @ result
bl    printf          @ from < standard
                      @ lib.

mov   r0, 0          @return 0;
add   sp, sp, locals @ deallocate local
                     @ var
ldr   fp, [sp, 0]    @restore caller fp
ldr   lr, [sp, 4]    @          lr
add   sp, sp, 8      @          and sp
bx    lr             @ return
```

promptAddr:
    .word prompt
displayAddr:
    .word display

P.T.O

papergrid
Date: 11/14/2024

4. Ans:

@ addition.s
@ Prompts user for two hex numbers and adds
@ them

@ Defines my Raspberry Pi
.cpu cortex-a53
.fpu neon-fp-armv8
.syntax unified @ modern syntax

@ Constant for assembler
.equ maxChars, 9 @ max input chars
.equ inString1, -24 @ for 1st input
@ string
.equ inString2, -36 @ for 2nd input
@ string
.equ outString, -52 @ for output
@ string
.equ locals, 40 @ space for local
@ vars

@ Constant program data
.section .rodata
.align 2
prompt:
.asciz "Enter up to 32-bit hex
number: "
display:
.asciz "Their sum is: "

@ The program
.text
.align 2
.global main
.type main, %function

Date: 11/14/2021

main:

```
sub   sp, sp, 16    @space for saving regs
str   r4, [sp,0]    @save r4
str   r5, [sp,4]    @      r5
str   fp, [sp, 8]   @      fp
str   lr , [sp, 12] @      lr
add   fp, sp, 12    @set our frame pointer
sub   sp, sp, locals @for local vars

ldr   r0, promptaddr @prompt user
bl    writestr

add   r0, fp, instring1 @1st input
mov   r1, maxChars @limit input size
bl    readstr
add   r0, fp, instring1 @user input
bl    hexToInt    @convert it
mov   r4, r0      @1st int

ldr   r0 promptaddr @prompt for 2nd
bl    writestr

add   r0, fp, instring2 @2nd input
mov   r1, maxChars @limit input size
bl    readstr
add   r0, fp, instring2 @user input
bl    hexToInt    @convert it
mov   r5, r0      @2nd int

add   r1, r4, r5  @add the two ints
add   r0, fp, outstring @place for result
                  @-bt
bl    intToHex    @convert to hex
                  @string
```


P.T.O

```
ldr r0, displayAddr  @ show user result
bl writeStr

add r0, fp, outString
bl writeStr

bl newline           @ looks nicer

mov r0, 0            @ return 0;
add sp, sp, locals   @ deallocate local
                     @ vars
ldr r4, [sp, 0]      @ restore r4
ldr r5, [sp, 4]      @      r5
ldr fp, [sp, 8]      @      fp
ldr lr, [sp, 12]     @      lr
add sp, sp, 16       @ restore sp
bx lr                @ return

promptAddr:
        .word prompt
displayAddr:
        .word display
```

Ex. 14.7

1. Ans: Only writing the main function here because of a lack of time.

```
@ incrementDec.s
@ Prompts user for unsigned decimal number
@ and adds 1 to it.

@ Define my Raspberry Pi
```


P.T.O

Date: 11/14/2021

. cpu cortex-a53
. fpu neon-fp-armv8
. syntax unified @ modern syntax

@ Constants for assembler
. equ maxchars, 11 @ max input chars
. equ inputstring, -16 @ for input string
. equ outputstring, -28 @ for output string
. equ locals, 32 @ space for local vars

@ Constant program data
. section .rodata
. align 2

prompt:
. ascii "Enter an unsigned number up to 4294967294: "

@ The program
. text
. align 2
. global main
. type main, %function

main:
sub sp, sp, 8 @ space for fp, lr
str fp, [sp, 0] @ save fp
str lr, [sp, 4] @ save lr
add fp, sp, 4 @ set our frame pointer
sub sp, sp, locals @ for local vars

ldr r0, promptaddr @ prompt user
bl writestr

P.T.O

Date: 11 / 14 / 202

```
add  r0, fp, inputString @ place for user
                         @ input
mov  r1, maxChars  @ limit input size
bl   readStr

add  r0, fp, inputString @ user input
bl   userTextInt   @ convert it

add  r1, r0, 1     @ increment user's num-
                   @ - ber
add  r0, fp, outputString
bl   intToAscii

add  r0, fp, outputString
bl   writeStr
bl   newLine

mov  r0, 0         @ return 0;
add  sp, sp, locals @ deallocate
                    @ local vars
ldr  fp, [sp, 0]   @ restore caller
                   @ fp
ldr  lr, [sp, 4]   @ lr
add  sp, sp, 8     @ and sp
bx   lr            @ return
```

promptAdder:

- word prompt

2. Ans

@ intToAscii.s
@ Converts an int to the corresponding unsigned
@ decimal text string.
@ Calling sequence:

P.T.O

Date: 11/12/2021

```
@       r0 <- address of place to store string
@       r1 <- int to convert
@       ld uintToAsc

@ Define my Raspberry Pi
        .cpu cortex-a53
        .fpu neon-fp-armv8
        .syntax unified     @modern syntax

@ Constant for assembler
        .equ    tempString, -40 @for temp string
        .equ    locals, 16      @space for local vars
        .equ    zero, 0x30      @ ascii 0
        .equ    NUL, 0

@ The program
        .text
        .align 2
        .global uintToAsc
        .type uintToAsc, %function
uintToAsc:
        sub     sp, sp, 24      @space for saving regs
        str     r4, [sp, 0]     @ save r4
        str     r5, [sp, 4]     @      r5
        str     r6, [sp, 8]     @      r6
        str     r7, [sp, 12]    @      r7
        str     fp, [sp, 16]    @      fp
        str     lr, [sp, 20]    @      lr
        add     fp, sp, 20      @ set our frame
                                @ pointer
        sub     sp, sp, locals  @ for local vars

        mov     r4, r0          @ caller's string pointer
```


P.T.O

```
adr   r5, fp, tempString  @tempString
mov   r7, 10              @decimal constant

mov   r0, NUL             @end of C string
stob  r0, [r5]
add   r5, r5, 1           @move to char stora-
                          @ - ge

mov   r0, zero            @assume the int is 0
stob  r0, [r5]
mov   r6, r1              @int to convert
beq   copyback            @zero is special cas

convertloop:
      cmp   r6, 0         @end of int?
      beq   copy          @yes, copy for caller
      ndiv  r0, r6, r7    @no, div to get quot-
                          @ - int
      mls   r2, r0, r7, r6 @the rest (remainder
                          @ - er)
      mov   r6, r0        @the quotient
      add   r2, r2, zero  @convert to numeral
      stob  r2, [r5]
      add   r5, r5, 1     @next char position
      b     convertloop

copy:
      sub   r5, r5, 1     @last char stored locally

copyloop:
      ldob  r0, [r5]      @get local char
      stob  r0, [r4]      @store the char for
                          @ caller
      cmp   r0, NUL       @end of local string?
      beq   allDone       @yes, we're done
      add   r4, r4, 1     @no, next caller
                          @ location
```

Date: 11/14/2024

sub r5, r5, 1 @ next local char
b copyLoop

allDone:
strb r0, [r4] @ end of string
add sp, sp, locals @ deallocate local
@ vars
ldr r4, [sp, 0] @ restore r4
ldr r5, [sp, 4] @ r5
ldr r6, [sp, 8] @ r6
ldr r7, [sp, 12] @ r7
ldr fp, [sp, 16] @ fp
ldr lr, [sp, 20] @ lr
add sp, sp, 24 @ sp
bx lr @ return

5.4cy @ incrementSigned.s
@ Prompts user for signed decimal number and adds
@ 1 to it.

@ Define my Raspberry Pi
.cpu cortex-a53
.fpu neon-fp-armv8
.syntax unified @ modern syntax

@ Constant program data
.section .rodata
.align 2
prompt:
.asciz "Enter a signed integer between
-2147483648 and +2147483646: "

@ The program
.text

P.T.O

```
.align 2
.global main
.type main, %function
main:
    sub  sp, sp, 8      @space for fp, lr
    str  fp, [sp, 0]    @ save fp
    str  lr, [sp, 4]    @   and lr
    add  fp, sp, 4      @ set our frame
                        @ pointer

    ldr  r0, promptAddr @prompt user
    bl   writeStr

    bl   getDecInt      @convert N

    add  r0, r0, 1      @increment user's number
    bl   putDecInt      @print result
    bl   newline

    mov  r0, 0          @ return 0
    ldr  fp, [sp, 0]    @ restore caller fp
    ldr  lr, [sp, 4]    @      lr
    add  sp, sp, 8      @   and sp
    bx   lr             @ return

promptAddr:
    .word prompt

@ getDecInt.s
@ gets signed decimal integer from keyboard.
@ calling sequence:
@      bl getDecInt
@ returns equivalent int
```

Date: 11/14/2021

@ Defining my Raspberry Pi
.cpu cortex-A53
.fpu neon-fp-armv8
.syntax unified @ modern syntax

@ Constants for assembler
.equ maxChars, 12 @ max input chars
.equ inputString, -20 @ for input string
.equ locals, 8 @ space for local
@ vars

@ Useful source code constants
.equ POS, 0
.equ NEG, 1

@ The program
.text
.align 2
.global getDecInt
.type getDecInt, %function
getDecInt:
sub sp, sp, 16 @ space for saving regs
@ (keeping 8-byte sp
@ align)
str x4, [sp, 4] @ save x4
str fp, [sp, 8] @ fp
str lr, [sp, 12] @ lr
add fp, sp, 12 @ set our frame pointer
sub sp, sp, locals @ for the string

add x0, fp, inputString @ place to
@ store input
mov x1, maxChars @ limit input
@ length

bl readStr

P.T.O

```
        add  r0, fp, inputstring   @ input string
        mov  r4, POS               @ assume positive int

        ldrb r1, [r0]              @ get char
        cmp  r1, '-'               @ minus sign?
        bne  checkPlus             @ no, check for plus
                                   @ sign
        mov  r4, NEG               @ yes, flag as neg
        add  r0, r0, 1             @ go to the number
        b    convert               @ and convert it
checkPlus:
        cmp  r1, '+'               @ plus sign?
        bne  convert               @ no, we treat the
                                   @ number
        add  r0, r0, 1             @ go to the number
convert:
        bl   numberToInt

        cmp  r4, POS               @ positive int?
        beq  allDone               @ yes, we're done
        mvn  r0, r0                @ no, complement it
        add  r0, r0, 1             @ and finish negate
allDone:
        add  sp, sp, locals        @ deallocate local
                                   @ var
        ldr  r4, [sp, #]           @ restore r4
        ldr  fp, [sp, 8]           @       fp
        ldr  lr, [sp, 12]          @       lr
        add  sp, sp, 16            @       sp
        bx   lr                    @ return

@ putInt.s
@ Converts an int to the corresponding signed
```


P.T.O

@ decimal text string
@ Calling sequence:
@ r0 <- int to print
@ bl putDecInt

@ Define my Raspberry Pi
.cpu cortex-a53
.fpu neon-fp-armv8
.syntax unified @ modern syntax

@ Constants for assembler
.equ decString, -20 @ for string
.equ locals, 8 @ space for local vars

@ useful source code constants
.equ FP, 0
.equ NEGBIT, 0x80000000

@ the program
.text
.align 2
.global putDecInt
.type putDecInt, %function
putDecInt:
sub sp, sp, 16 @ space for saving regs
@ (keeping 8-byte stack align)
str r4, [sp, 4] @ save r4
str fp, [sp, 8] @ fp
str lr, [sp, 12] @ lr
add fp, sp, 12 @ set our frame pointer
sub sp, sp, locals @ space for the string

add r4, fp, decString @ place to store
@ string (-20)

```
mov r2, '+'     @ assume positive
strb r3, [r4]
tst r0, NEGBIT  @ negative int?
beq positive    @ no, go on
mov r2, '-'     @ yes, need to negate
strb r2, [r4]
mvn r0, r0      @ complement
add r0, r0, 1   @ two's complement
positive:
    mov r2, r0      @ set to convert
    add r0, r4, 1   @ skip over sign char
    bl uint2dec

    add r0, fp, decString   @ string to write
    bl writeStr

    add sp, sp, locals      @ deallocate local
                            @ vars
    ldr r4, [sp,4]  @ restore r4
    ldr fp, [sp,8]  @       fp
    ldr lr, [sp,12] @       lr
    add sp, sp, 16  @       sp
    bx lr           @ return
```