Austin Bumbalough
CPE 325-08
Lab 04
09/23/2019

Lab 04 Solution

Part 1

In Lab 4 part 1, I created an assembly program to count the number of mathematical operators in the string "Do 42+53/76%8=2*8-32+71 & you can sleep." The string is located at memory address 0x3100.

```
0x003100  testStr
0x003100  D   o   .   4   2   +   5   3   /   7   6   %   8   =   2
0x00310F  *   8   -   3   2   +   7   1   .   &   .   y   o   u   .
0x00311E  c   a   n   .   s   l   e   e   p   .   .
0x003129  opChars
0x003129  %   &   *   +   -   /   <   >   =   ^   ~   .   .
0x003136  $../part1_solution.asm:42:74$, RESET
0x003136  1   @   .   1   .   @   .   Z   .   .   .   C   .   C   4
0x003145  @   .   1
0x003148  getNext
0x003148  5   @   )   1   v   D   F   .   .   $
0x003152  testOp
0x003152  w   E   G   .   .   '   G   .   .   #   .   S   .   .   .
0x003161  ?
0x003162  stringEnd
0x003162  :   @   .   .   .   ?
0x003168  $isr_trap.asm:48:59$, __TI_ISR_TRAP
0x003168  2   .   .   .   .   ?   .   C   .   .   .   .   .   .   .
0x003177  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
0x003186  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
0x003195  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
0x0031A4  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
0x0031B3  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
```

After processing the string, the number of operator charcters is stored in a variable located in RAM at memory address 0x1100.

```
0x001100  0008 5349 4920 2053 2041 494D 4558 2D44 4143 4553 5320 525
0x001118  4E49 2E47 0000 4011 D820 0080 0450 0C11 5108 6040 00E0 000
0x001130  23FE 531A 23FC 4303 3FFF 2000 4020 0840 04CA 2834 8680 210
0x001148  8402 0290 4040 0003 4080 4000 0000 4411 4404 4082 0281 504
0x001160  0000 1005 0742 4000 2088 B000 0E00 0021 0000 0010 0026 008
0x001178  8100 0820 4051 0008 0800 0809 031C 0180 1100 A102 02D4 015
0x001190  0000 0004 02A4 0210 0090 0050 0080 2140 024C 0005 2100 028
0x0011A8  0000 0C25 2C08 1400 83C2 1408 A041 0003 8180 0412 0000 600
0x0011C0  AC40 1010 4004 0800 1800 1180 8051 0040 1400 0088 0030 042
0x0011D8  A00C 0004 8802 0804 A764 3140 0090 0000 07C0 0120 6803 084
0x0011F0  0000 40D0 3830 8880 0008 0100 0440 0089 FF7F BFFF DBDF FBF
0x001208  1F5B EFFB 9E8E BF6D E7BB E17F 6BBF EEFF FB75 D7FF 7BDB B7F
0x001220  EFFF EF7C DBFF 7FBF 5CDF 7CCF FFC7 F2F6 FFFD 1D9F 5B5B ACE
0x001238  FEF2 ECFF FD5F FFFD F63F DB7F 6FE6 BF3E DFDD 5ED9 F97C F95
0x001250  EBFF 5C5D CD9F 7BFE 5FA7 FBC8 3EFD FBF7 AD6D FDAD 9FEF FBD
0x001268  DF77 9FE7 7ECF 7DEF FBDD FEFB 55FF FBD7 EB7F EBFF FF7E 8FF
0x001280  FFDC DFFF FAFF BF13 F75E FFF9 AFEF FFF4 FD5B EE1F 9FDB E7D
0x001298  FCFB FEEE FFF3 FFBF BDF5 3FFF 3BFB 7FF3 FCFA EFF6 FB7F 55F
0x0012B0  FFC7 FFFF BF7F EFF5 F9FE FFF7 F75F 7EEB 7FC7 FBED FFFF FFB
0x0012C8  EE2D EFCF EBBB CDF3 BFE9 FFFF F4DF F7D1 FFF1 FDBF 33BF FD9
0x0012E0  7FB7 BBFD EEFB FDDE DF7F DEDF 3EBF FFCF 7F3C 8FB7 DEFF BBF
0x0012F8  F7BF EDDE BEDF FBD7 0000 C020 0802 0600 6020 2000 4AC0 4C0
0x001310  2C00 0001 0264 0019 0002 0802 0040 2138 0000 4002 0208 120
```

Source Code

```
            .cdecls C,LIST,"msp430.h"       ; Include device header file
            .align 2                                    ; Align to words

testStr:        .cstring "Do 42+53/76%8=2*8-32+71 & you can sleep." ; Program input string
```

```
opChars:        .cstring "%&*+-/<>=^~"          ; Allocates 11 bytes for math operator characters
                        .data
charCount:      .byte 0

;-------------------------------------------------------------------------------
            .def    RESET                   ; Export program entry-point to
                                            ; make it known to linker.
;-------------------------------------------------------------------------------
            .text                           ; Assemble into program memory.
            .retain                         ; Override ELF conditional linking
                                            ; and retain current section.
            .retainrefs                     ; And retain any sections that have
                                            ; references to current section.

;-------------------------------------------------------------------------------
RESET:       mov.w   #__STACK_END,SP         ; Initialize stackpointer
             mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer


;-------------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------------

                clr.w SR                        ; Clear status register
                clr.w R8                        ; Clear R8 to store the matched character count

                mov.w #testStr, R4              ; Move address of first test string character into R4
getNext:        mov.w #opChars, R5              ; Move address of first operator character into R5
                mov.b @R4+, R6                  ; Move next test character into R6 and increment R4 to next character address
                cmp.b #0, R6                    ; Check if the current test character is the null character
                jeq stringEnd                   ; If the current test character is null character, jump to program end
testOp:         mov.b @R5+, R7                  ; Move next operator charcter into R7 and increment R5 to next operator character
                cmp.b #0, R7                    ; Check if all operator characters have been tested
                jeq getNext                     ; All operator characters have been compared to current test character
                cmp.b R6, R7                    ; Check if current test character matches current operator character
                jne testOp                      ; If not match, get next operator character and repeat
                inc.w charCount                 ; If the characters match, increment the counter register and
                jmp getNext                     ; Get next operator charcter and repeat
stringEnd:      mov.w #charCount, R10           ; Move memory address of counter variable to R10
                jmp $                           ; End of program, unconditional jump to current location


;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
            .global __STACK_END
            .sect   .stack

;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
            .sect   ".reset"                ; MSP430 RESET Vector
            .short  RESET
```

Part 2

In lab 4 part 2, I created an assembly program to interpret a string containing a mathematical expression and evaluate the result. Valid operands are single digit numbers and valid operators are '+' and '-'. The result of evaluating the string "1+2+4-6" is displayed on the MSP 430 port 2.

Name	Value	Description
P2OUT	0x01	Port 2 Output [Memory Mapped]
P7	0	P7
P6	0	P6
P5	0	P5
P4	0	P4
P3	0	P3
P2	0	P2
P1	0	P1
P0	1	P0

Source Code

```
            .cdecls C,LIST,"msp430.h"       ; Include device header file
            .align 2                                        ; Align to words

;-------------------------------------------------------------------------------
            .def    RESET                   ; Export program entry-point to
                                            ; make it known to linker.

;-------------------------------------------------------------------------------
; Program variables stored in RAM (.data)
;-------------------------------------------------------------------------------

evalString:     .cstring "1+2+4-6"

;-------------------------------------------------------------------------------
            .text                           ; Assemble into program memory.
            .retain                         ; Override ELF conditional linking
                                            ; and retain current section.
            .retainrefs                     ; And retain any sections that have
                                            ; references to current section.

;-------------------------------------------------------------------------------
RESET:       mov.w   #__STACK_END,SP         ; Initialize stackpointer
             mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer


;-------------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------------

                clr.w SR                        ; Clear status register
                clr.w R7                        ; Clear result register R7
                clr.w R8                        ; Clear operator flag register R8
                bis.b #0xFF, P2DIR              ; Set port 2 to output
                mov.w #evalString, R4           ; Move string start address to R4
                jmp getOperand

getOperator:    mov.b @R4+, R6                  ; Get next Operator
                cmp.b #0, R6                    ; Check for string end
                jeq printResult
                cmp.b #'+', R6                  ; Check if operator is +
                jeq setAdd
                cmp.b #'-', R6                  ; Check if operator is -
                jeq setSub
convOp:         cmp.b #'+', R6                  ; Compare operator and set operation flag
                jeq setAdd
                cmp.b #'-', R6
                jeq setSub
setAdd:         mov.b #0, R8                    ; Set operation flag to 0
                jmp getOperand
setSub:         mov.b #1, R8                    ; Set operation flag to 1
                jmp getOperand
getOperand:     mov.b @R4+, R5                  ; Move next operand to R5
charToDec:      cmp.b #'0', R5                  ; Convert operand from ascii to decimal
                jeq ascii0
                cmp.b #'1', R5
                jeq ascii1
                cmp.b #'2', R5
                jeq ascii2
                cmp.b #'3', R5
                jeq ascii3
                cmp.b #'4', R5
                jeq ascii4
                cmp.b #'5', R5
                jeq ascii5
                cmp.b #'6', R5
                jeq ascii6
                cmp.b #'7', R5
                jeq ascii7
                cmp.b #'8', R5
                jeq ascii8
                cmp.b #'9', R5
                jeq ascii9
ascii0:         mov.b #0, R5
```

```
                jmp evaluate
ascii1:         mov.b #1, R5
                jmp evaluate
ascii2:         mov.b #2, R5
                jmp evaluate
ascii3:         mov.b #3, R5
                jmp evaluate
ascii4:         mov.b #4, R5
                jmp evaluate
ascii5:         mov.b #5, R5
                jmp evaluate
ascii6:         mov.b #6, R5
                jmp evaluate
ascii7:         mov.b #7, R5
                jmp evaluate
ascii8:         mov.b #8, R5
                jmp evaluate
ascii9:         mov.b #9, R5
                jmp evaluate
evaluate:       cmp #0, R8                  ; Check operation flag and perform operation
                jeq regAdd
                cmp #1, R8
                jeq regSub
regAdd:         add.b R5, R7                ; Add R5 to R7
                jmp getOperator
regSub:         sub.b R5, R7                ; Subtract R5 from R7
                jmp getOperator
printResult:    mov.b R7, P2OUT             ; Display R7 on Port 2
                jmp $

;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
            .global __STACK_END
            .sect   .stack

;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
            .sect   ".reset"                ; MSP430 RESET Vector
            .short  RESET
```

Part 3 (Bonus)

In lab 4 part 3, I created an assembly program to convert all lowercase characters in a string to uppercase. The input string is located in RAM at memory address 0x1100.

```
0x001100  T  h  i  s  .  i  s  .  a  .  M  i  x  e  d
0x00110F  -  C  a  s  e  .  s  t  r  i  n  g  .  .  .
0x00111E  .  @  .  .  .  .  P  .  .  .  .  Q  @  `  .
0x00112D  .  .  .  .  #  .  S  .  #  .  C  .  ?  .  .
0x00113C  .  @  @  .  .  .  4  (  .  .  .  !  .  .  .
0x00114B  .  @  @  .  .  .  @  .  @  .  .  .  D  .  D
0x00115A  .  @  .  .  D  P  .  .  .  .  B  .  .  @  .
0x001169  .  .  .  .  .  !  .  .  .  .  .  &  .  .  .
0x001178  .  .  .  .  Q  @  .  .  .  .  .  .  .  .  .
0x001187  .  .  .  .  .  .  .  P  .  .  .  .  .  .  .
0x001196  .  .  .  .  P  .  .  .  @  !  L  .  .  .  .
0x0011A5  !  .  .  .  .  %  .  .  ,  .  .  .  .  .  .
0x0011B4  A  .  .  .  .  .  .  .  .  .  .  `  @  .  .
0x0011C3  .  .  @  .  .  .  .  .  .  Q  .  @  .  .  .
0x0011D2  .  .  0  .  ,  .  .  .  .  .  .  .  .  .  d
0x0011E1  .  @  1  .  .  .  .  .  .  .  .  .  h  E  .
0x0011F0  .  .  .  @  0  8  .  .  .  .  .  .  @  .  .
0x0011FF  .  .  .  .  .  .  .  .  .  [  .  .  .  .  .
0x00120E  m  .  .  .  .  .  .  k  .  .  u  .  .  .  .
0x00121D  {  .  .  .  .  |  .  .  .  .  .  .  \  .  |
0x00122C  .  .  .  .  .  .  .  .  [  [  .  .  .  .  .
0x00123B  .  _  .  .  .  ?  .  .  .  .  o  >  .  .  .
0x00124A  .  ^  |  .  U  .  .  .  ]  \  .  .  .  {  .
0x001259     .  .  .  >  .  .  m  .  .  .  .  .  .  .
```

The original string is modified such that the output string is stored at the same location.

```
0x001100  T  H  I  S  .  I  S  .  A  .  M  I  X  E  D
0x00110F  -  C  A  S  E  .  S  T  R  I  N  G  .  .  .
0x00111E  .  @  .  .  .  .  P  .  .  .  .  Q  @  `  .
0x00112D  .  .  .  .  #  .  S  .  #  .  C  .  ?  .  .
0x00113C  .  @  @  .  .  .  4  (  .  .  .  !  .  .  .
0x00114B  .  @  @  .  .  .  @  .  @  .  .  .  D  .  D
0x00115A  .  @  .  .  D  P  .  .  .  .  B  .  .  @  .
0x001169  .  .  .  .  .  !  .  .  .  .  .  &  .  .  .
0x001178  .  .  .  .  Q  @  .  .  .  .  .  .  .  .  .
0x001187  .  .  .  .  .  .  .  P  .  .  .  .  .  .  .
0x001196  .  .  .  .  P  .  .  .  @  !  L  .  .  .  .
0x0011A5  !  .  .  .  .  %  .  .  ,  .  .  .  .  .  .
0x0011B4  A  .  .  .  .  .  .  .  .  .  .  `  @  .  .
0x0011C3  .  .  @  .  .  .  .  .  .  Q  .  @  .  .  .
0x0011D2  .  .  0  .  ,  .  .  .  .  .  .  .  .  .  d
0x0011E1  .  @  1  .  .  .  .  .  .  .  .  .  h  E  .
0x0011F0  .  .  .  @  0  8  .  .  .  .  .  .  @  .  .
0x0011FF  .  .  .  .  .  .  .  .  .  [  .  .  .  .  .
0x00120E  m  .  .  .  .  .  .  k  .  .  u  .  .  .  .
0x00121D  {  .  .  .  .  |  .  .  .  .  .  .  \  .  |
0x00122C  .  .  .  .  .  .  .  .  [  [  .  .  .  .  .
0x00123B  .  _  .  .  .  ?  .  .  .  .  o  >  .  .  .
0x00124A  .  ^  |  .  U  .  .  .  ]  \  .  .  .  {  .
0x001259     .  .  .  >  .  .  m  .  .  .  .  .  .  .
```

Source Code

```
            .cdecls C,LIST,"msp430.h"       ; Include device header file
            .align 2                                        ; Align to words

;-------------------------------------------------------------------------------
            .def    RESET                   ; Export program entry-point to
                                            ; make it known to linker.

;-------------------------------------------------------------------------------
; Program variables stored in RAM (.data)
;-------------------------------------------------------------------------------

                        .data
testString:     .cstring "This is a Mixed-Case string."

;-------------------------------------------------------------------------------
            .text                           ; Assemble into program memory.
            .retain                         ; Override ELF conditional linking
                                            ; and retain current section.
            .retainrefs                     ; And retain any sections that have
                                            ; references to current section.

;-------------------------------------------------------------------------------
RESET:       mov.w   #__STACK_END,SP         ; Initialize stackpointer
             mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer


;-------------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------------

               mov.w #testString, R4           ; Move string start address to R4
getNext:       mov.b @R4, R5                   ; Move first character into R5
               cmp.b #0, R5                    ; Check for string end
               jeq stringEnd
               cmp #97, R5                     ; Check if R5 contains a lower case character
               jl incString
               cmp #123, R5
               jge incString
               sub.b #32, 0(R4)                ; Subtract 32 to convert lower case to upper case
               jmp incString
incString:     inc.w R4
               jmp getNext
stringEnd:     jmp $

;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
            .global __STACK_END
            .sect   .stack
```

```
;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
            .sect   ".reset"                ; MSP430 RESET Vector
            .short  RESET
```