CPE 325: Intro to Embedded Computer System

Lab04

Performing Specified Tasks on Strings in Assembly

Submitted by: Caleb Keller

Date of Experiment:_______09/10/2022______________

Report Deadline:__________09/14/2022______________

Demonstration Deadline: _________09/14/2022_________________

Introduction

Write a brief discussion on what the lab is about. (Use the tutorial and write in your own words. DO NOT copy text).

In this lab, I am writing assembly code to perform specific tasks on given strings. The first part of this project will be counting the number of digits as well as uppercase letters in a string and storing the amount of each to a specified register. For the second part, I will be converting all instances of lowercase letters in a string to uppercase and outputting the result in a specified output port.

Theory

Write short notes on each topic discussed in lab.

Topic 1: Assembler Directives

a) **Assembler directives:** assembly language directives tell the assembler to set the data and program at specific addresses in address space, allocate space for constants and variables, define synonyms, include additional files, etc. Some examples of these directives are: equate, origin, define space, define constant, and include.

Topic 2: Different Addressing Modes

a.) **Register**: the operand is contained in one of the CPU registers R0 to R15. It is the fastest addressing mode and needs the least memory. The data within the register can be accessed using word or byte instructions.

b.) **Indexed**: the address of the operand is the sum of the index and the contents of the register.

c.) **Symbolic**: the content of the addresses EDE/TONI are used for this operation. The source of destination address is computed as a difference from the PC and uses the PC in indexed addressing mode.

d.) **Absolute**: the contents of the fixed addresses are used for the operation. The SR is used in the indexed mode to create an absolute 0.

e.) **Indirect**: the registers are used as a pointer to the operand.

f.) **Immediate**: any immediate 8 or 16 bit constant can be used with the instruction. The PC is used in autoincrement mode to emulate this addressing mode.

g.) **Indirect with autoincrement**: the registers are used as a pointer to the operand. The registers are incremented afterwards – by 1 in byte mode, by 2 in word mode. For instance, I used this is my program to scan through each of the characters in the string.

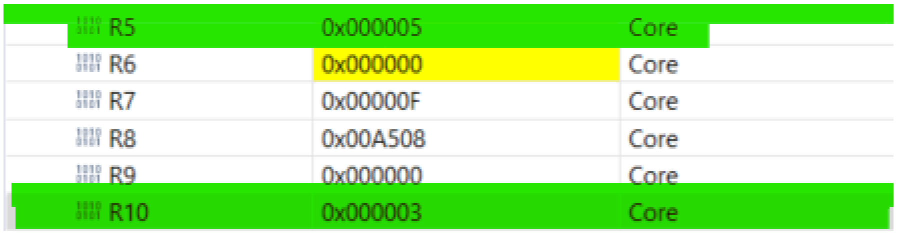
Results & Observation

Program 1:

Program Description:

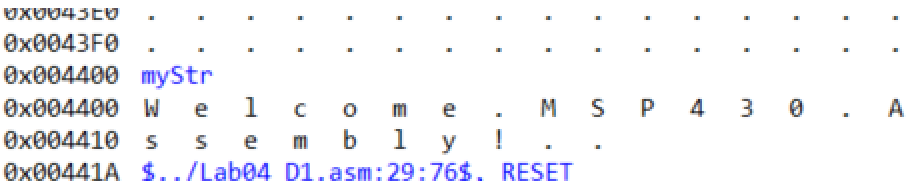
Explain your approach in solving the problem.

To perform the given tasks for this program, I began by writing code to test if the character in the string was a digit. If it was, I would then call a function to increment the amount of occurrences and placed the amount in R10. For the uppercase letters, I followed the same method and also called another function to increment a counter and place the uppercase value in R5.



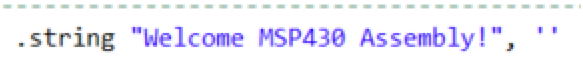
R5	0x000005	Core
R6	0x000000	Core
R7	0x00000F	Core
R8	0x00A508	Core
R9	0x000000	Core
R10	0x000003	Core

Digit Count Output in R10 and Uppercase Count Output in R5



```
0x0043E0  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
0x0043F0  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
0x004400  myStr
0x004400  W  e  l  c  o  m  e  .  M  S  P  4  3  0  .  A
0x004410  s  s  e  m  b  l  y  !  .  .
0x00441A  $../Lab04 D1.asm:29:76$. RESET
```

String in memory for program 1

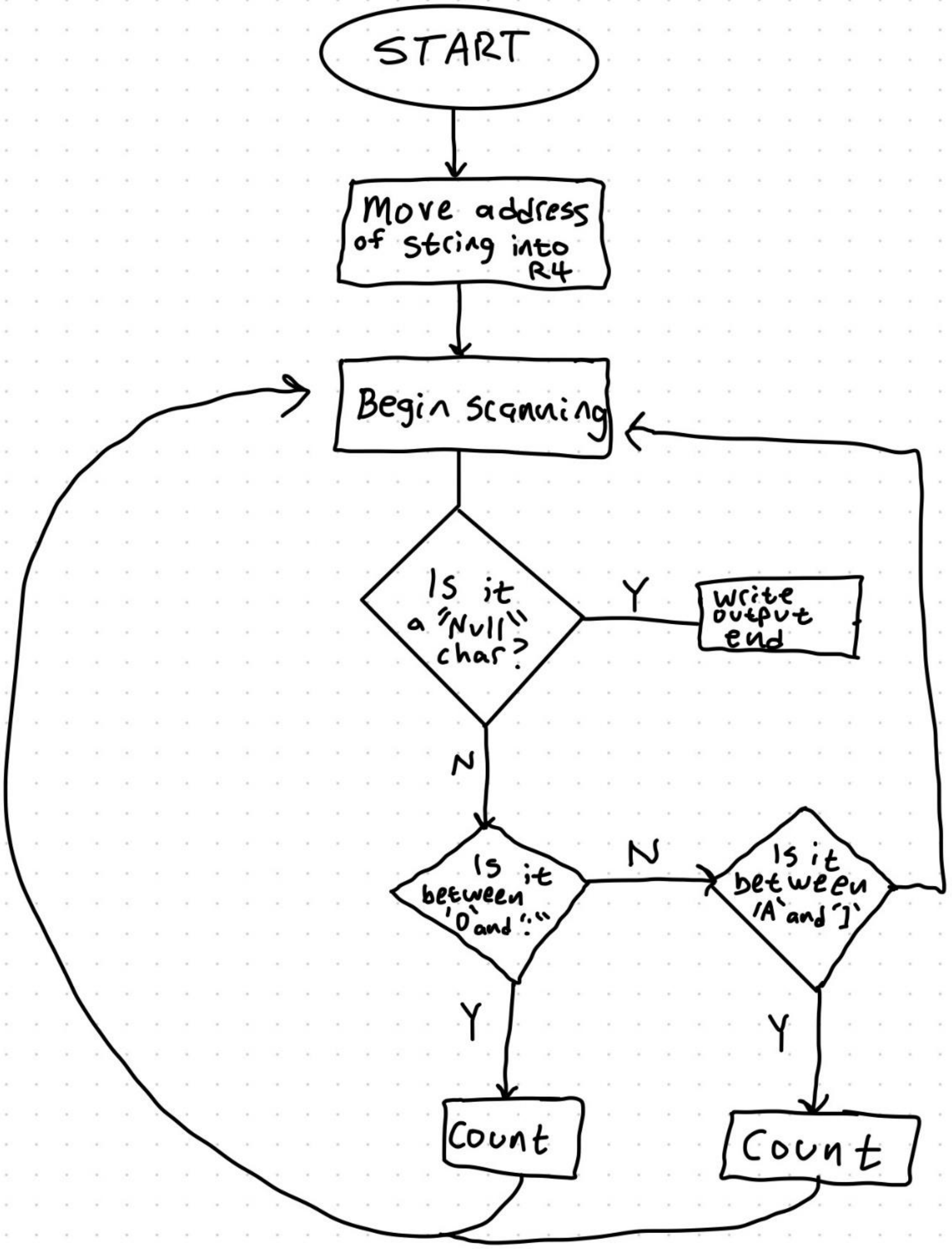


```
.string "Welcome MSP430 Assembly!", ''
```

Given string for program 1

Program 1 Flowchart

LAB 4 PART 1

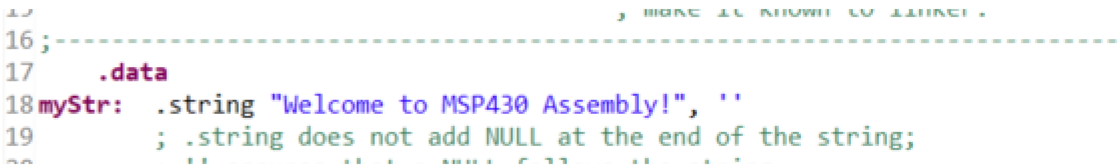

START
Move address of string into R4
Begin scanning
Is it a "Null" char?
Y
Write output end
N
Is it between '0' and ':'
N
Is it between 'A' and ']'
Y
Y
Count
Count

Program 2:

Program Description:

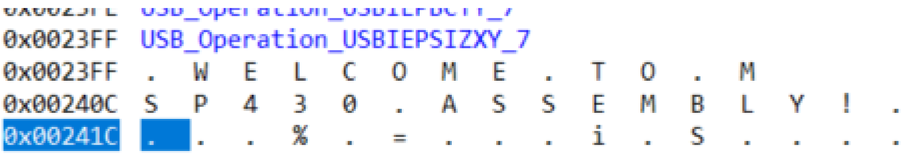
Explain your approach in solving the problem.

In this program, I implemented assembly code to change the lowercase letters in the string to uppercase. I also counted how many changes and kept track of them in R7. I followed a similar method as program 1, but I had to subtract 20 from the lowercase value in ASCII to match the uppercase value.



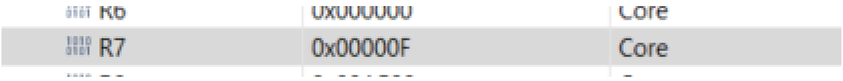
```
                                                  ; make it known to linker.
;-------------------------------------------------------------------------------
            .data
myStr:      .string "Welcome to MSP430 Assembly!", ''
            ; .string does not add NULL at the end of the string;
```

***Given string to perform tasks on"**



```
0x0023FE  USB_Operation_USBIEPBCTX_7
0x0023FF  USB_Operation_USBIEPSIZXY_7
0x0023FF  .  W  E  L  C  O  M  E  .  T  O  .  M
0x00240C  S  P  4  3  0  .  A  S  S  E  M  B  L  Y  !  .
0x00241C  .  .  .  %  .  =  .  .  .  i  .  S  .  .  .  .
```

Altered string in memory

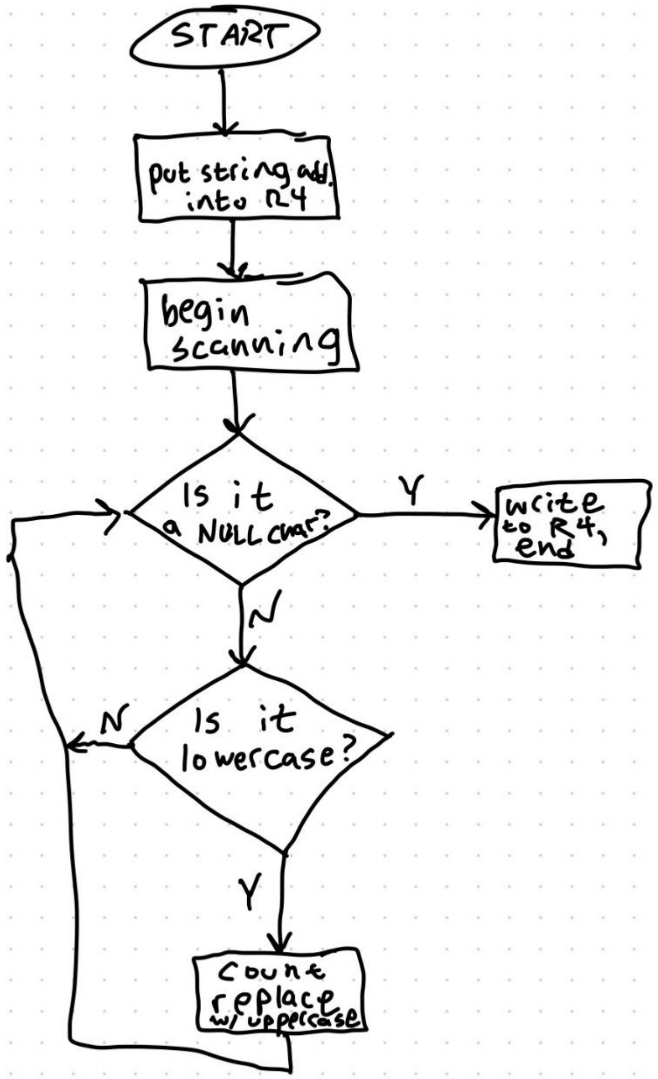


R6	0x000000	Core
R7	0x00000F	Core

Number of characters changed

Program 2 Flowchart

LAB 4 PART 2


START
put string add. into R4
begin scanning
Is it a NULL char?
Y
write to R4, end
N
Is it lowercase?
N
Y
count replace w/ uppercase

Conclusion

While working on this program, I ran into many challenges and issues. My most prevalent challenge was formatting the assembly code differently than what I am used to with C/C++. Assembly language is very surface level, so it took me some time to understand how to implement things correctly. I learned several things while working on these two programs, including how to use subroutines, as well as viewing completed code in the memory browser.

Appendix

Your first code goes here, if any. Make sure you use a 1X1 table for this.

(Note: Make sure the code is readable, have comments. Also reduce spacing between lines to avoid lengthy reports.

Program 1 source code

```
 ; ------------------------------------------------------------------------------
; File:          Lab04_D1.asm
; Description:   Counts the number of digits and characters in a given string
; Input:         The input string specified in myStr
; Output:        The port P1OUT displays the number of capital letters in the string
;                          The port P2OUT displays the number of digits in the
string
; Author(s):     Caleb Keller
; Date:          September 10, 2022
; Revised:       September 14, 2022
; -------------------------------------------------------------------------------
        .cdecls C, LIST, "msp430.h"     ; Include device header file
;--------------------------------------------------------------------------------
        .def    RESET                   ; Export program entry-point to
                                        ; make it known to linker.
;--------------------------------------------------------------------------------
myStr:  .string "Welcome MSP430 Assembly!", ''
        ; .string does not add NULL at the end of the string;
        ; '' ensures that a NULL follows the string.
        ; You can alternatively use .cstring "HELLO WORLD, I AM THE MSP430!"
        ; that adds a NULL character at the end of the string automatically.
;--------------------------------------------------------------------------------
        .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 stack pointer
        mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer
;--------------------------------------------------------------------------------
; Main loop here
;--------------------------------------------------------------------------------
main:   ;bis.b   #0FFh, &P1DIR           ; Do not output the result on port pins

        mov.w   #myStr, R4               ; Load the starting address of the string
into R4
```

```
        clr.b   R5                      ; Register R5 will serve as a counter for
digits
        clr.bR10                                    ; Register R10 will serve as
a counter for capitals

gnext:       mov.b   @R4+, R6                ; Get a new character
                    cmp.b        #0, R6                          ; is it the null
character?
                    jeq          lend                            ; if yes, go to end b/c
we are finished with string
             cmp.b  #'0', R6                   ; BEGIN DIGIT TEST
             jl     gnext                        ; if greater than or equal to 0
continue
             cmp.b  #':', R6
             jl           gnextLoop                   ; if less than : go to loop
to increment



                    cmp.b  #'A', R6                 ; BEGIN CAPITAL LETTER TEST
                    jl           gnext                           ;if greater than or
equal to A continue
                    cmp.b  #']', R6
                    jl           gnextLoop2                      ;if less than ] go to
loop to increment
                    jmp          gnext



gnextLoop:          inc.w  R10                         ; increment counter in R10
for digits
                           jmp gnext                            ; start again

gnextLoop2:         inc.w  R5                          ; increment counter in R5 for
capital letters
                           jmp gnext                            ; start again



lend:   mov.b   R10,&P1OUT              ; Write result in P1OUT (not visible on
port pins)
             mov.b  R5,&P2OUT

        bis.w   #LPM4, SR              ; LPM4
        nop                            ; Required only for debugger
;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
        .global __STACK_END
        .sect   .stack
;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
        .sect   ".reset"               ; MSP430 RESET Vector
        .short  RESET
        .end
```

Program 2 Source Code

```
; -----------------------------------------------------------------------------
; File:         Lab04_D2.asm
; Description:  Changes lowercase characters to uppercase in a given string
; Input:        The input string specified in myStr
; Output:       The port P3OUT displays the number of changes made in the string
; Author(s):    Caleb Keller
; Date:         September 10, 2022
; Revised:      September 14, 2022
; -----------------------------------------------------------------------------
        .cdecls C, LIST, "msp430.h"     ; Include device header file
;------------------------------------------------------------------------------
        .def    RESET                   ; Export program entry-point to
                                        ; make it known to linker.
;------------------------------------------------------------------------------
       .data
myStr:  .string "Welcome to MSP430 Assembly!", ''
        ; .string does not add NULL at the end of the string;
        ; '' ensures that a NULL follows the string.
        ; You can alternatively use .cstring "HELLO WORLD, I AM THE MSP430!"
        ; that adds a NULL character at the end of the string automatically.
;------------------------------------------------------------------------------
        .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 stack pointer
        mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer
;------------------------------------------------------------------------------
; Main loop here
;------------------------------------------------------------------------------
main:   bis.b   #0FFh, &P3DIR           ; output the result to P3 port pin

        mov.w   #myStr, R4               ; Load the starting address of the string
into R4

        clr.b   R7                       ; Register R5 will serve as a counter


gnext:       mov.b   @R4+, R6                 ; Get a new character
                    cmp.b  #0, R6                           ; is it the null character?
                    jeq          lend                             ; if yes, go to end b/c
we are finished with string
             cmp.b  #'a', R6                ; BEGIN LOWERCASE TEST
             jl           gnext                         ; if less than a continue
             cmp.b  #'{', R6
             jl           low2CAPS                  ; if less than { go to change
from lowercase to capital



low2CAPS:    inc.b  R7                              ; incrememnt counter in R7
                    sub.b  #0x20, R6                ; subtract decimal 20 from
lowercase character to get matching capital letter
                                                                 ; (ASCII table,
any lowercase - 20 = corresponding capital letter
                    mov.b  R6, -1(R4)               ; put adjusted string into
memory

             jmp    gnext                     ; go back to start again





lend:  mov.b  R7, &P3OUT
        bis.w   #LPM4, SR                   ; LPM4
        nop                                 ; Required only for debugger
;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
        .global __STACK_END
        .sect   .stack
;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
        .sect   ".reset"                    ; MSP430 RESET Vector
        .short  RESET
        .end
```

Make sure to submit a single pdf file only