Keiran Berry

CENG320L

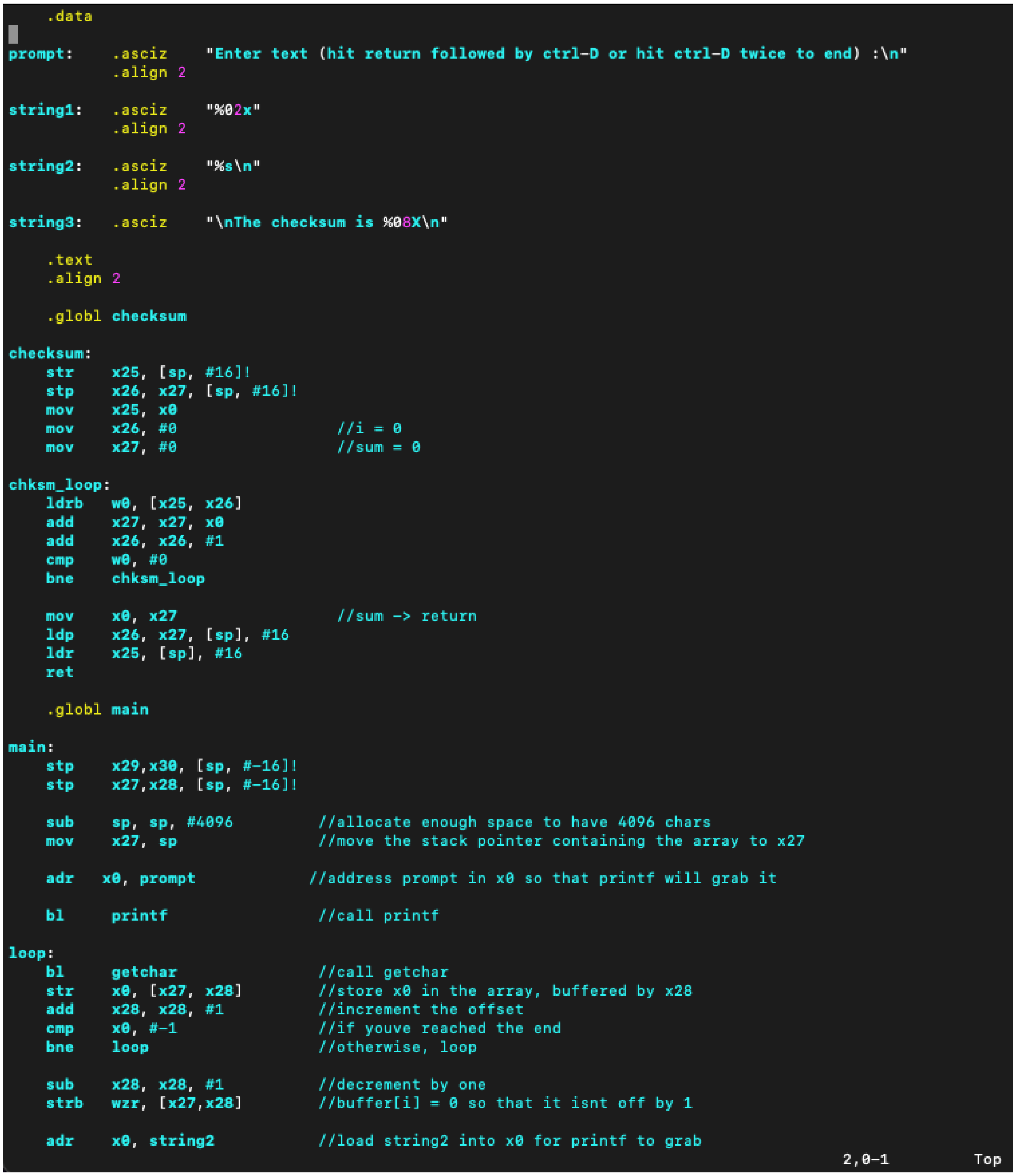
Lab 4 Report

29 September, 2022

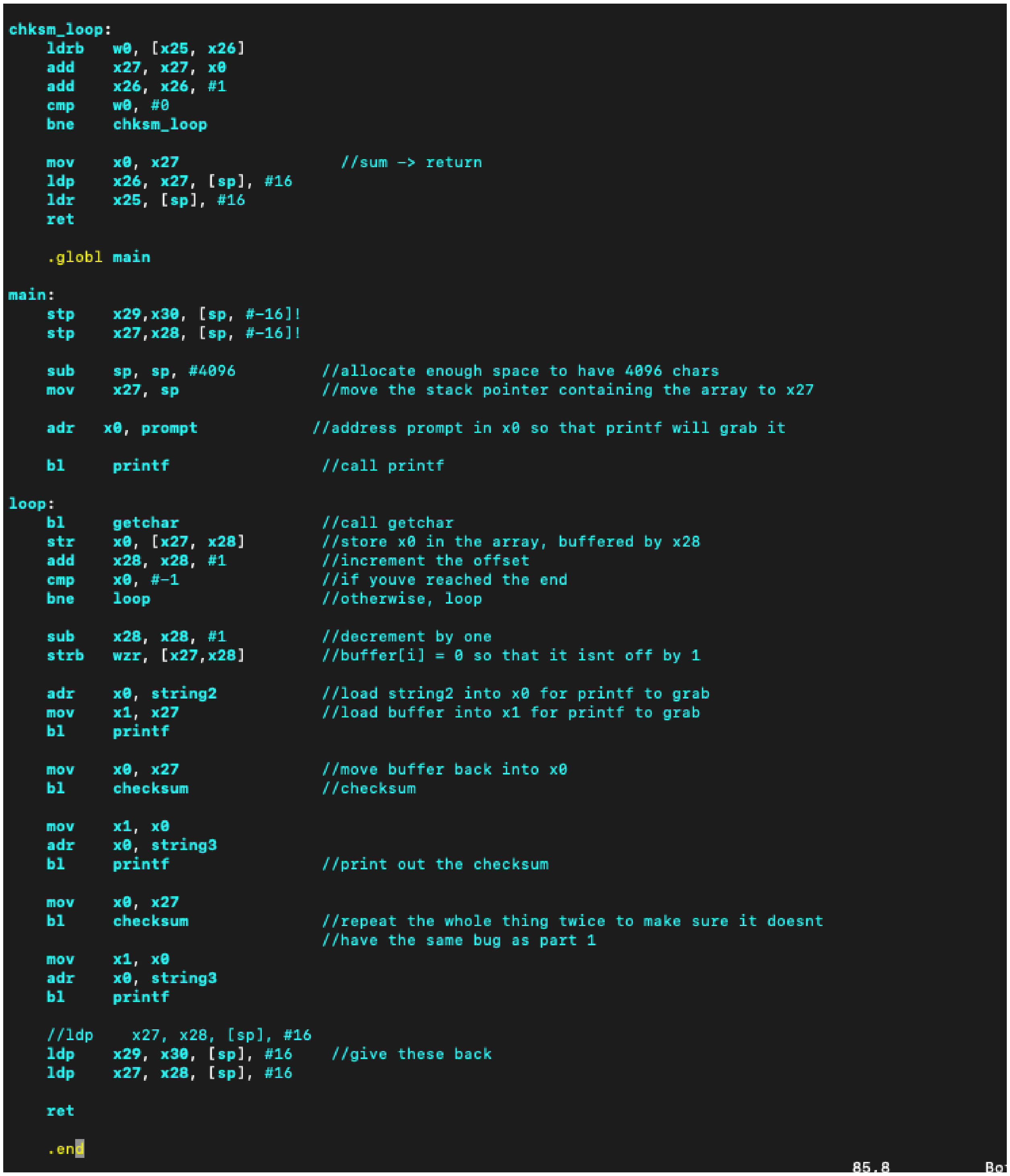
The goal of this lab was to find a bug in existing C code, and practice converting C code into assembly language. The bug in the first implementation of the C code has to do with the variables being global. Because checksum is called twice, the checksum will be correct the first time and then will be doubled the second time. Since it is a global variable, it will not be cleared at all and will just add the second pass through on top of the original solution.

The static portion of the lab was not required, but if I were to code up that portion I would implement the array by declaring buffer as a .byte, initializing it to 0, then using a .skip 4096. To implement it using the stack, I subtracted 4096 from the stack pointer and then moved the stack pointer into a non-volatile register. To implement the checksum function using the stack, I initialized both i and the sum to 0 inside the checksum function, then looped through addition until it was done. This was done through using a bne statement, where if it were not finished it would go back to the loop and if it were then it would finish the function.

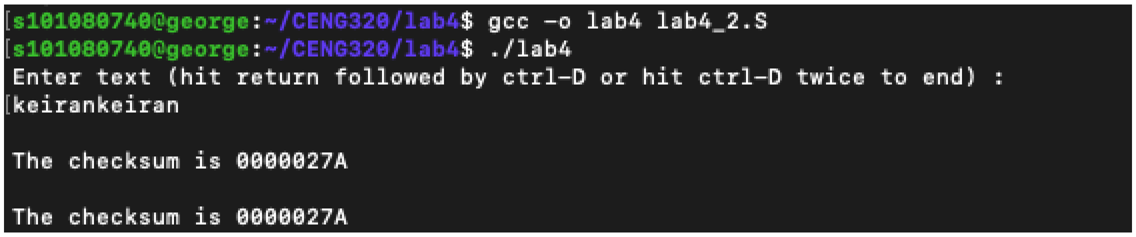
I ran into many issues with this lab, including compilation errors and the array overstepping by one, which was solved by decrementing the array by one before setting the final element to 0. This lab taught me a lot about loops and syntax in assembly, and I now understand better the equivalents of C code in assembly, making me more comfortable programming in assembly in the future.



```
    .data

prompt:     .asciz    "Enter text (hit return followed by ctrl-D or hit ctrl-D twice to end) :\n"
            .align 2

string1:    .asciz    "%02x"
            .align 2

string2:    .asciz    "%s\n"
            .align 2

string3:    .asciz    "\nThe checksum is %08X\n"

    .text
    .align 2

    .globl checksum

checksum:
    str    x25, [sp, #16]!
    stp    x26, x27, [sp, #16]!
    mov    x25, x0
    mov    x26, #0                //i = 0
    mov    x27, #0                //sum = 0

chksm_loop:
    ldrb   w0, [x25, x26]
    add    x27, x27, x0
    add    x26, x26, #1
    cmp    w0, #0
    bne    chksm_loop

    mov    x0, x27                //sum -> return
    ldp    x26, x27, [sp], #16
    ldr    x25, [sp], #16
    ret

    .globl main

main:
    stp    x29,x30, [sp, #-16]!
    stp    x27,x28, [sp, #-16]!

    sub    sp, sp, #4096        //allocate enough space to have 4096 chars
    mov    x27, sp              //move the stack pointer containing the array to x27

    adr   x0, prompt           //address prompt in x0 so that printf will grab it

    bl     printf               //call printf

loop:
    bl     getchar              //call getchar
    str    x0, [x27, x28]       //store x0 in the array, buffered by x28
    add    x28, x28, #1         //increment the offset
    cmp    x0, #-1              //if youve reached the end
    bne    loop                 //otherwise, loop

    sub    x28, x28, #1         //decrement by one
    strb   wzr, [x27,x28]       //buffer[i] = 0 so that it isnt off by 1

    adr    x0, string2          //load string2 into x0 for printf to grab
                                                                        2,0-1         Top
```



```
chksm_loop:
    ldrb   w0, [x25, x26]
    add    x27, x27, x0
    add    x26, x26, #1
    cmp    w0, #0
    bne    chksm_loop

    mov    x0, x27                //sum -> return
    ldp    x26, x27, [sp], #16
    ldr    x25, [sp], #16
    ret

    .globl main

main:
    stp    x29,x30, [sp, #-16]!
    stp    x27,x28, [sp, #-16]!

    sub    sp, sp, #4096         //allocate enough space to have 4096 chars
    mov    x27, sp               //move the stack pointer containing the array to x27

    adr   x0, prompt            //address prompt in x0 so that printf will grab it

    bl     printf                //call printf

loop:
    bl     getchar               //call getchar
    str    x0, [x27, x28]        //store x0 in the array, buffered by x28
    add    x28, x28, #1          //increment the offset
    cmp    x0, #-1               //if youve reached the end
    bne    loop                  //otherwise, loop

    sub    x28, x28, #1          //decrement by one
    strb   wzr, [x27,x28]        //buffer[i] = 0 so that it isnt off by 1

    adr    x0, string2           //load string2 into x0 for printf to grab
    mov    x1, x27               //load buffer into x1 for printf to grab
    bl     printf

    mov    x0, x27               //move buffer back into x0
    bl     checksum              //checksum

    mov    x1, x0
    adr    x0, string3
    bl     printf                //print out the checksum

    mov    x0, x27
    bl     checksum              //repeat the whole thing twice to make sure it doesnt
                                 //have the same bug as part 1
    mov    x1, x0
    adr    x0, string3
    bl     printf

    //ldp    x27, x28, [sp], #16
    ldp    x29, x30, [sp], #16    //give these back
    ldp    x27, x28, [sp], #16

    ret

    .end
                                                                                         85,8          Bo
```



```
[s101080740@george:~/CENG320/lab4$ gcc -o lab4 lab4_2.S
[s101080740@george:~/CENG320/lab4$ ./lab4
 Enter text (hit return followed by ctrl-D or hit ctrl-D twice to end) :
[keirankeiran

 The checksum is 0000027A

 The checksum is 0000027A
```