CMPE310 LA2 Report – Johann Mission – 3/30/25

The goal of the assignment was given a txt file with numbers, one per line, to output those numbers and ultimately output the sum of the numbers.

My initial approach entailed reading the txt file by assigning a file descriptor and a buffer to different registers. This allowed me to output each number.

This would be done in a loop, as long as the line read in was not empty. Therefore, I would allocate a read loop, and output each number in that same loop.

In the end, I would have to also close the file.

My initial approach to summing the numbers was to assign a new register to hold a summation value. In the read loop, I would take the new number and add it to the register of the summation value, and right before closing the file I'd call printf on the sum.

However, when I got it to output, it was outputting garbage values. By debugging, I realized that the register I was using to output each individual number was holding an ASCII value not an int. This would let me output it but not sum it.

Therefore everytime I read a number from the file, I'd have to use a different loop to convert it from ASCII to int and then add that int to a summation register which holds int.

This parsing loop would identify one digit at a time , making sure it's a valid ASCII int, and add it to the temporary register. It thenwill check if the next character is another int, (double/triple digit), in that case it'll multiply by ten and push back to build the second digit. When there is no next number detected it breaks the loop for that value and adds it to the sum, where it will be outputted when the last line is reached.

Code (print5.asm)

```
section .data
    pathname: db "file2.txt", 0
    buffer: times 2048 db 0
    newline db 10, 0
    format_string: db "%s", 0
    sum_format: db "Sum: %ld", 10, 0

section .bss
    fd resb 4
    sum resq 1  ; Store total sum

section .text
    global main
    extern printf

main:
    ;xor edi,edi ; initialise edi as 0
    ;xor rdi,rdi
    ;; read only
    mov eax, 5
    mov ebx, pathname
    mov ecx, 0
    int 0x80
    mov [fd], eax

    xor rdi, rdi  ; Clear sum register
    mov [sum], rdi

read_loop:
    ; Read file and cout
    mov eax, 3
    mov ebx, [fd]
    mov ecx, buffer
    mov edx, 2048
    int 0x80

    cmp eax, 0
    jle close_file

    mov byte [buffer+eax], 0

    ; print each individual number using format string and print
    mov rdi, format_string
    mov rsi, buffer
    xor rax, rax
    call printf

```

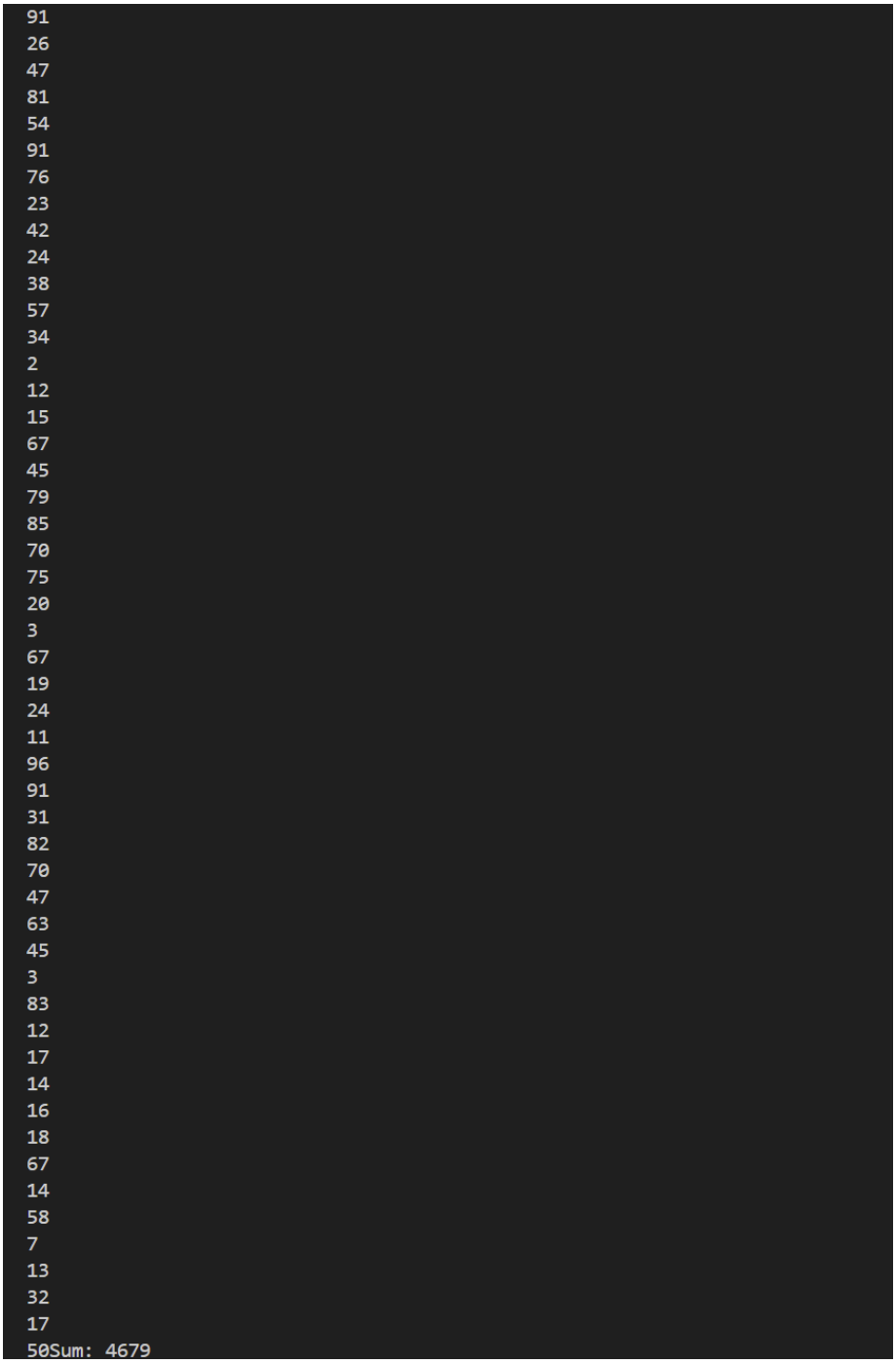
```
48      ; we need to add the number to sum. this will parse ASCII to int
49      mov rsi, buffer
50      call sum_numbers
51
52      jmp read_loop
53
54  ; initialise registers for where we're going to convert ascii to int
55  sum_numbers:
56      xor rax, rax  ;initialise rax ,the temp register we're going to put the
57      xor rdx, rdx  ; where we're going to put the digits
58
59  ;; ascii to int is done separately from printing, we will print this sum at the end.
60  parse_loop:
61      movzx rcx, byte [rsi]  ; get byte from buffer
62      test rcx, rcx          ; make sure its not null
63      jz sum_done            ;;
64
65      ;; make sure the number is a number
66      cmp rcx, '0'
67      jl skip_char           ; if the ascii is less than 0, skip this line
68      cmp rcx, '9'          ; if the ascii is greater than 9, skip this line
69      jg skip_char
70
71      sub rcx, '0'           ; Convert ASCII to integer
72      imul rax, rax, 10      ; Multiply current number by 10 for the second digit
73      add rax, rcx           ; Add new digit
74
75      jmp next_char
76
77  skip_char:
78      test rax, rax          ; Check if we have a number
79      jz next_char
80      add [sum], rax         ; Add current number to sum
81      xor rax, rax           ; reset
82
83  next_char:
84      inc rsi ;increment counter
85      jmp parse_loop  ;reloop
86
87  sum_done:
88      test rax, rax          ; Check if last number needs to be added
89      jz end_sum_numbers
90      add [sum], rax         ; Add last number
91
92  end_sum_numbers:
93      ret
```

```
94
95  close_file:
96      ; Print sum
97      mov rsi, [sum]
98      mov rdi, sum_format
99      xor rax, rax
100     call printf
101     ; close out
102     mov eax, 6
103     mov ebx, [fd]
104     int 0x80
105
106     mov eax, 1
107     xor ebx, ebx
108     int 0x80
109
```

Output with given randomInt file (renamed as file.txt)

Sum: 4679

All the numbers are outputted, just too big to include here



Using a custom input (file2.txt)

SSH FS - umbc.gl > cmpe310 > 2025-03-14 > file2.txt

```
449
3
30
2025
4
67
9
11
45
```

The sum of this by calculator is 2643.

Output:

```
50Sum: 4679
[m376@linux5 2025-03-14]$ nasm -f elf64 print5.asm -o print5.o
[m376@linux5 2025-03-14]$ gcc print5.o -o print5
[m376@linux5 2025-03-14]$ print5
449
3
30
2025
4
67
9
11
45Sum: 2643
[m376@linux5 2025-03-14]$
```