Dylan Orebaugh

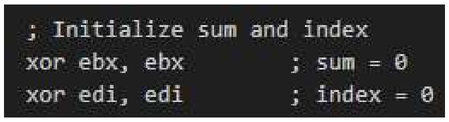
CMPE310

Lab Assignment 2

For this Project, I started with being able to read every line of the file I was trying to read. This was done during the lab. It was a pretty simple process. All I had to do was open the file, read the line, turn the integers into ascii so that it can be printed, and then finally close the file at the very end. If there was an error reading the file or the file didn't exist, The code would run the error function that tells the user that the file doesn't exist. By the end, I was able to properly implement code that was able to read every line of a file and print it out. The next step was the much harder part

After I was able to read every line of the file, I had to find a way to add up every single line and print out the sum of all of the integers in the file. First I had to think of a way to add up all of the integers. I did this by making an array to be able to store the integers. I Then set the "sum" variable and the "index" to zero by doing this,

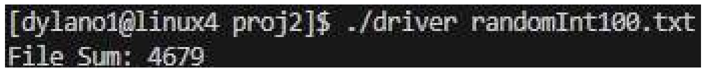
Figure 1:



```
; Initialize sum and index
xor ebx, ebx        ; sum = 0
xor edi, edi        ; index = 0
```

Then I would check if all of the numbers were read. If they were, I would jump to the function that would print the sum. For adding all of the integers up in the file, I would add whatever was in the array next, to a register. Once I got to the end of the file, I would finally print out the Sum of said file. Of course after I was done reading the file, I would close the file like you would for any project that includes a file. I code my project in a way that you can insert any file with integers via the command line and it will be able to print out the sum. Here are a few example of me running my code,

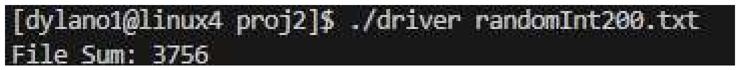
Figure 2:



```
[dylano1@linux4 proj2]$ ./driver randomInt100.txt
File Sum: 4679
```

This was the file given to us in the lab.

Figure 3:



```
[dylano1@linux4 proj2]$ ./driver randomInt200.txt
File Sum: 3756
```

This was a file that I made myself.

In conclusion, This was a pretty simple project but it did have its difficulties because we are coding in assembly and that is never an easy task. Compared to other languages, reading files is a much more complicated process but in the end i was able to figure it out.

My code:

```
section .data
    fmt_input   db "%d", 0          ; Format string for fscanf
    fmt_output  db "Sum: %d", 10, 0   ; Format string for printf
    error_msg   db "Error opening file", 10, 0
    array       times 1000 dd 0       ; Array to store integers
    mode        db "r", 0             ; Read mode for fopen


section .bss
    file_ptr    resd 1                ; File pointer
    count       resd 1              ; Number of integers

    sum         resd 1              ; Store sum


section .text

    global main

    extern fopen, fscanf, printf, fclose


main:

    ; Extract command-line argument (filename)

    mov eax, [esp+8]   ; Get pointer to argv[1]

    test eax, eax      ; Check if filename was provided

    jz error           ; If NULL, print error and exit


    ; Open the file

    push mode          ; Push file mode "r"

    push eax           ; Push filename

    call fopen

    add esp, 8         ; Clean up stack

    cmp eax, 0

    je error           ; If fopen fails, print error and exit

    mov [file_ptr], eax ; Save file pointer


    ; Read the first line to get the number of integers
    push dword [file_ptr]

    push count

    push fmt_input

    call fscanf

    add esp, 12


    ; Initialize sum and index

    xor ebx, ebx        ; sum = 0

    xor edi, edi        ; index = 0


read_loop:

    cmp edi, [count]    ; Check if all numbers read

    jge done_reading

    push dword [file_ptr]

    lea eax, [array + edi * 4]  ; Get address of array[index]

    push eax

    push fmt_input

    call fscanf

    add esp, 12

    add ebx, [array + edi * 4]

    inc edi

    jmp read_loop
```

```
done_reading:
    mov [sum], ebx      ; Store sum

    ; Print sum
    push ebx
    push fmt_output
    call printf
    add esp, 8

    ; Close the file
    push dword [file_ptr]
    call fclose
    add esp, 4

    ; Exit
    mov eax, 1          ; syscall for exit
    xor ebx, ebx
    int 0x80

error:
    push error_msg
    call printf
    add esp, 4
```

```
mov eax, 1

mov ebx, -1

int 0x80
```