Sum the File Report

I don't know how the Hamming Proj folder got in the Lab_Assignment_2 branch in github, just ignore it.

Explanation of Code:

Start by opening and reading in file into buffer.

load:

Loads in one character at a time checking if there is a newline. If there is a newline it jumps to add_(single/double/triple/quadruple) depending on how many characters have been loaded. If no endline is found, jumps to end_of_file

End_of_file:

Jumps to add_(single/…) based on which characters are empty.

Add_(single/double/triple/quadruple):

Adds ones place to sum, multiplies 10s place by 10 if 2+ digits, adds to sum, and so on.

If the integer counter is 0 it jumps to initialize, else jumps to array_insert

Initialize:

Sets num_ints equal to sum

Array_insert:

Puts sum into array

Sum_array:

Sums array

Write:

(Trys to)Print thousands, hundreds, tens, ones places

Code:

```
section .data
        pathname DD "/afs/umbc.edu/users/r/y/ry02253/home/cmpe310/Assignment2/randomInt100.txt"

section .bss
buffer:    resb 1024
char:      resb 1
sum:    resb 2
array:  resb 2000
num_ints:       resb 4
file_over resb 1

section .text
        global _start
_start:
        xor esp, esp
        xor ecx, ecx
        mov eax, 5
        mov ebx, pathname
        mov ecx, 0
        int 0x80
        jmp read
read:
        mov ebx, eax
        mov eax, 3
        mov ecx, buffer
        mov edx, 1024
        int 0x80
        jmp loop_start


loop_start:
        xor eax, eax
        xor ebx, ebx
        xor ecx, ecx
        xor edx, edx
        xor esp, esp
        xor ebp, ebp
        jmp load

load:
        mov byte dl, [buffer + esp]
        inc esp
        mov byte cl, [buffer + esp]
        inc esp
        cmp cl, 10
        je add_single
        mov byte bl, [buffer + esp]
        inc esp
        cmp bl, 10
        je add_double
-UU-:**-  F1  sumfile.asm    Top   L50    (Assembler) ----------------------------------------------
```

```
        mov byte ah, [buffer + esp]
        inc esp
        cmp ah, 10
        je add_triple
        mov byte al, [buffer + esp]
        inc esp
        cmp al, 10
        je add_quadruple
        mov dh, 1
        mov [file_over], dh
        xor dh, dh
        jmp end_of_file


add_single:
        sub dl, 48
        add [sum], edx
        cmp ebp, 0
        je initialize
        jmp array_insert

add_double:
        sub cl, 48
        add [sum], ecx
        mov ecx, edx
        sub ecx, 48
        mov eax, 10
        mul ecx
        add [sum], eax
        cmp ebp, 0
        je initialize
        jmp array_insert

add_triple:
        sub ebx, 48
        add [sum], ebx
        mov ebx, edx
        mov eax, 10
        sub ecx, 48
        sub ebx, 48
        mul ecx
        add [sum], eax
        mov eax, 100
        mul ebx
        add [sum], eax
        cmp ebp, 0
        je initialize
        jmp array_insert

add_quadruple:
-UU-:**-  F1  sumfile.asm    24%   L51    (Assembler) --------
```

```
add_quadruple:
        shr eax, 8
        sub eax, 48
        sub ebx, 48
        sub ecx, 48
        sub edx, 48
        add [sum], eax
        mov eax, 10
        mov ch, dl
        mul ebx
        add [sum], eax
        mov bl,ch
        mov eax, 100
        mul ecx
        add [sum], eax
        mov eax, 1000
        mul ebx
        add [sum], eax
        cmp ebp, 0
        je initialize
        jmp array_insert

end_of_file:
        mov esi, [num_ints]
        cmp cl, 0
        je add_single
        cmp bl, 0
        je add_double
        cmp ah, 0
        je add_triple
        cmp al, 0
        je add_quadruple
        jmp sum_initialize

initialize:
        mov ax, [sum]
        mov [num_ints], ax
        mov edx, [num_ints]
        xor eax, eax
        xor ebx, ebx
        xor ecx, ecx
        xor edx, edx
        mov [sum], eax
        inc ebp
        jmp load

array_insert:
        mov eax, [sum]
        mov [array + 2*ebp], eax
        inc ebp
-UU-:**-  F1  sumfile.asm    45%   L129   (Assembler) --------------------
```

```
array_insert:
        mov eax, [sum]
        mov [array + 2*ebp], eax
        inc ebp
        xor eax, eax
        xor ebx, ebx
        xor ecx, ecx
        xor edx, edx
        mov [sum], eax
        cmp ebp, [num_ints]
        jg sum_initialize
        mov eax, [file_over]
        cmp eax, 0
        jne sum_initialize
        jmp load

sum_initialize:
        xor eax, eax
        xor ebx, ebx
        xor ecx, ecx
        xor edx, edx
        mov edx, [num_ints]
        jmp sum_array


sum_array:
        xor ebx, ebx
        mov byte bh, [array + (2*ecx) + 1]
-UU-:**-  F1  sumfile.asm    55%   L143   (Assembler) ----------
```

```
sum_array:
        xor ebx, ebx
        mov byte bh, [array + (2*ecx) + 1]
        mov byte bl, [array + (2*ecx)]
        add eax, ebx
        inc ecx
        cmp ecx, edx
        jg write
        jmp sum_array

write:
        mov [sum], eax
        mov ebx, [sum]
        xor ecx, ecx
        jmp thousands

thousands:
        cmp ebx, 1000
        jle printchar
        sub ebx, 1000
        inc ecx

hundreds:
        cmp ebx, 100
        jle printchar
        sub ebx, 100
        inc ecx
-UU-:**-  F1  sumfile.asm    65%   L168   (Assembler) ----------
```

```
hundreds:
        cmp ebx, 100
        jle printchar
        sub ebx, 100
        inc ecx

tens:
        cmp ebx, 10
        jle printchar
        sub ebx, 10
        inc ecx

ones:
        cmp ebx, 0
        je printchar
        dec ebx
        inc ecx

printchar:
        mov ebp, ebx
        mov ebx, 1
        mov eax, 4
        add ecx, 48
        mov [char], ecx
        mov ecx, char
        mov edx, 1
        int 0x80
        mov ebx, ebp
        xor ecx, ecx
-UU-:**-  F1  sumfile.asm    76%   L195   (Assembler) ----
```

```
printchar:
        mov ebp, ebx
        mov ebx, 1
        mov eax, 4
        add ecx, 48
        mov [char], ecx
        mov ecx, char
        mov edx, 1
        int 0x80
        mov ebx, ebp
        xor ecx, ecx
        cmp ebx, 0
        je end
        cmp ebx, 1000
        jl hundreds
        cmp ebx, 100
        jl tens
        cmp ebx, 10
        jl ones
        jmp end

end:
        mov eax, 6
        int 0x80
        mov eax, 1
        int 0x80
```

```
-UU-:**-  F1  sumfile.asm    Bot    L231    (Assembler Ovwrt) ----------------
```

Output:

```
[ry02253@linux5 Assignment2]$ sumfile
4[ry02253@linux5 Assignment2]$emacs sumfile.asm
```

GDB screenshots:

Final sum of array

```
(gdb) info registers
rax            0x11e3             4579
rbx            0x32               50
rcx            0x65               101
rdx            0x64               100
rsi            0x64               100
rdi            0x0                0
rbp            0x65               0x65
rsp            0x129              0x129
r8             0x0                0
r9             0x0                0
r10            0x0                0
r11            0x0                0
r12            0x0                0
r13            0x0                0
r14            0x0                0
r15            0x0                0
rip            0x401223           0x401223 <write+7>
eflags         0x202              [ IF ]
cs             0x33               51
ss             0x2b               43
ds             0x0                0
es             0x0                0
fs             0x0                0
gs             0x0                0
fs_base        0x0                0
gs_base        0x0                0
(gdb)
```

Start of array summing:

Rax is total

Rbx is current int

```
rax            0x55                85
rbx            0x55                85
rcx            0x1                 1
rdx            0x64                100
rsi            0x64                100
rdi            0x0                 0
rbp            0x65                0x65
rsp            0x129               0x129
r8             0x0                 0
r9             0x0                 0
r10            0x0                 0
r11            0x0                 0
r12            0x0                 0
r13            0x0                 0
r14            0x0                 0
r15            0x0                 0
rip            0x401214            0x401214 <sum_array+20>
eflags         0x206               [ PF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
fs_base        0x0                 0
gs_base        0x0                 0
(gdb) c
Continuing.

Breakpoint 1, sum_array () at sumfile.asm:177
177             add eax, ebx
(gdb) info registers
rax            0x62                98
rbx            0xd                 13
rcx            0x2                 2
rdx            0x64                100
rsi            0x64                100
rdi            0x0                 0
rbp            0x65                0x65
rsp            0x129               0x129
r8             0x0                 0
```

```
rax            0xbc                188
rbx            0x5a                90
rcx            0x3                 3
rdx            0x64                100
rsi            0x64                100
rdi            0x0                 0
rbp            0x65                0x65
rsp            0x129               0x129
r8             0x0                 0
r9             0x0                 0
r10            0x0                 0
r11            0x0                 0
r12            0x0                 0
r13            0x0                 0
r14            0x0                 0
r15            0x0                 0
rip            0x401214            0x401214 <sum_array+20>
eflags         0x202               [ IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
fs_base        0x0                 0
gs_base        0x0                 0
(gdb) c
Continuing.

Breakpoint 1, sum_array () at sumfile.asm:177
177             add eax, ebx
(gdb) info registers
rax            0xdc                220
rbx            0x20                32
rcx            0x4                 4
rdx            0x64                100
rsi            0x64                100
rdi            0x0                 0
rbp            0x65                0x65
rsp            0x129               0x129
r8             0x0                 0
r9             0x0                 0
r10            0x0                 0
r11            0x0                 0
r12            0x0                 0
```

```
rax            0xe0                224
rbx            0x4                 4
rcx            0x5                 5
rdx            0x64                100
rsi            0x64                100
rdi            0x0                 0
rbp            0x65                0x65
rsp            0x129               0x129
r8             0x0                 0
r9             0x0                 0
r10            0x0                 0
r11            0x0                 0
r12            0x0                 0
r13            0x0                 0
r14            0x0                 0
r15            0x0                 0
rip            0x401214            0x401214 <sum_array+20>
eflags         0x212               [ AF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
fs_base        0x0                 0
gs_base        0x0                 0
(gdb) c
Continuing.

Breakpoint 1, sum_array () at sumfile.asm:177
177             add eax, ebx
(gdb) info registers
rax            0x134               308
rbx            0x54                84
rcx            0x6                 6
rdx            0x64                100
rsi            0x64                100
rdi            0x0                 0
rbp            0x65                0x65
rsp            0x129               0x129
r8             0x0                 0
r9             0x0                 0
r10            0x0                 0
r11            0x0                 0
r12            0x0                 0
r13            0x0                 0
```