

Author: Dominic Fantauzzo

ONID ID number: 934-556-579

Date: 10/21/2023

OSU email address: fantauzd@oregonstate.edu

Course number/section: CS 271, 400

Project Number: 2 – Debugging Lab

## **Part 1:**

**Question 1:** The current value (in Hex) is 0xffffffff

**Question 2:** Carry: clear, Overflow: clear, Zero: clear

## Screenshot 1: Part 1, Questions 1-2

Registers

EAX = FFFFFFFF EBX = 00267000 ECX = 00790000 EDX = 0040643A ESI = 004010AA EDI = 004010AA EIP = 004036C5 ESP = 0019FF74 EBP = 0019FF80 EFL = 00000202

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 0 AC = 0 PE = 0 CY = 0

2.

Disassembly

```
56 JMP _EndFile ; Quit
57
58 ; Attempt to Print data from file as ASCII characters (hopefully no NULLs)
59 _validRead:
60 MOV EDX, OFFSET fileBuffer
61 CALL WriteString
62 CALL CrLf
63
64 _EndFile:
65
66 Invoke ExitProcess, 0 ; exit to operating system
67 main ENDP
68
69 greetUser PROC
70 MOV EDX, OFFSET greeting
71 CALL WriteString
72 MOV EAX, 064h
73 CALL WriteHex
74 CALL CrLf
75 CALL CrLf
76 RET
```

Watch

Name	Value	Type
EAX	0xffffffff	unsigned int

Breakpoints

Name	Labels	Condition	Hit Count
DebugLab.asm, line 66		break always	

## Part 2:

Question 1: *shown below*

## Screenshot 2: Part 2, Question 1

Registers

EAX = 00000579 EBX = 00253000 ECX = 004010AA EDX = 00406000 ESI = 004010AA EDI = 004010AA EIP = 004036E0 ESP = 0019FF70 EBP = 0019FF80 EFL = 00000202

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 0 AC = 0 PE = 0 CY = 0

Disassembly

```
60 MOV EDX, OFFSET fileBuffer
61 CALL WriteString
62 CALL CrLf
63
64 _EndFile:
65
66 Invoke ExitProcess,0 ; exit to operating system
67 main ENDP
68
69 greetUser PROC
70 MOV EDX, OFFSET greeting
71 CALL WriteString
72 MOV EAX, 064h
73 CALL WriteHex
74 CALL CrLf ≤ 1ms elapsed
75 CALL CrLf
76 RET
77 greetUser ENDP
78 ; (insert additional procedures here)
79
80 END main
```

Diagnostics Tools

Diagnostics session: 0 seconds (45 ms selected)

44.1ms 44.2ms

Events

Process Memory

CPU (% of all processors)

Summary Events Memory Usage CPU Usage

Show Events (6 of 6)

Memory Usage

Take Snapshot

Enable heap profiling (affects performance)

CPU Usage

Record CPU Profile

Watch 1

Name	Value	Type
EAX	0x00000579	unsigned int

Breakpoints

Name	Labels	Condition	Hit Count
DebugLab.asm, line 30		break alive...	

Call Stack Breakpoints Exception Settings Command Window Immediate Window Output

Ready

Type here to search

74°F Clear 8:37 PM 10/21/2023

## Part 3:

**Question 1:** The instruction at the `_validName` code label is `mov`. It has a destination operator, `eax`, and a source operator `fileHandle`.

**Question 2:** The memory address is `0040368A`

**Question 3:**

The value in EIP represents the memory address of the next instruction that will be executed. We can track this value by using the registers window when debugging. The leftmost value on any given line represents the memory address for that instruction (when in disassembly, as shown below). Hence, the leftmost value on any given line is fed into the EIP register when that line is the next to be executed. Therefore, the relationship between the value in EIP and the leftmost value on any given line is significant because we, as the programmer, can use the debugger to compare the leftmost value to EIP and determine which line the code will execute next. This gives us the ability to see if our flow of control is executing as expected, especially when debugging branching and looping structures. If we run into bugs, we can utilize this relationship to step through code and find where EIP does not behave as expected. Pinpointing where EIP does not match our expected leftmost value can allow us to fix issues more quickly.

### Screenshot 3: Part 3, Question 1-2

The screenshot displays the Visual Studio Code interface with a debugger window open. The main window shows assembly code for a program, with the instruction `MOV EAX, fileHandle` at address `0040368A` highlighted by a red circle. The registers window at the top shows the current state of the CPU registers, including `EAX = FFFFFFFF`, `EBX = 003FD000`, `ECX = 006C0000`, `EDX = 006C0000`, `ESI = 004010AA`, `EDI = 004010AA`, `EIP = 00403677`, `ESP = 0019FF74`, `EBP = 0019FF80`, and `EFL = 00000246`. The status bar at the bottom indicates the current file is `DebugLab.asm` and the current thread is `[0x3690] Main Thread`.

The assembly code is as follows:

```
JMP _EndFile ; Quit
00403688 jmp _validName+3Bh (04036C5h)

; Attempt to Read Input File to Memory
_validName:
MOV EAX, fileHandle
0040368A mov eax, dword ptr [fileHandle (040604Ah)]
MOV EDX, OFFSET fileBuffer
0040368F mov edx, offset fileBuffer (040604Eh)
MOV ECX, MAX_FILE_SIZE
00403694 mov ecx, 3E8h
CALL ReadFromFile
00403699 call _ReadFromFile@0 (04010B4h)
MOV bytesRead, EAX
0040369E mov dword ptr [bytesRead (0406436h)], eax
JNC _validRead ; Jump if No Error in Reading File
004036A3 jae _validName+2Ch (04036B6h)
MOV EDX, OFFSET fileReadError
004036A5 mov edx, offset fileReadError (040648Fh)
CALL WriteString ; Notify User of File Read Error
```

The Watch window on the left shows the variable `EAX` with a value of `0xffffffff` and type `unsigned int`. The Breakpoints window on the right shows two breakpoints set at `DebugLab.asm, line 30` and `DebugLab.asm, line 38`, both with the condition `break always`.

The Diagnostic Tools window on the right shows the current session's performance metrics, including a timeline of events and memory usage. The CPU usage is shown as a bar chart, and the memory usage is shown as a bar chart. The CPU usage is currently at 0%, and the memory usage is at 0%.

## Part 4:

**Question 1:** My last three OSU ID digits are 579. The 580<sup>th</sup> byte (index 579) of TestText.txt, as an ASCII character, is s. (0x73)

**Question 2:** *see below*

## Screenshot 4: Part 4, Question 1

The screenshot displays the Visual Studio Code interface with the following components:

- Memory Window:** Shows a memory dump starting at address 0x00406295. A red circle highlights the address 0x00406295. The dump contains hexadecimal data and ASCII text, including a paragraph about the enemy that devastated the Imperium.
- Disassembly Window:** Shows assembly code for 'DebugLab.asm'. The code includes instructions like `CALL WriteString`, `CALL CrLf`, `JMP _EndFile`, and `MOV EAX, fileHandle`. A red circle highlights the instruction `MOV EAX, fileHandle` at line 47.
- Watch Window:** Shows variables `EAX` (value 0x00000449, type unsigned int), `fileHandle` (value 0x00000110, type unsigned long), and `INVALID_HANDLE_VALUE` (value identifier "INVALID\_HANDLE\_VALUE" is undefined).
- Breakpoints Window:** Shows two breakpoints: `DebugLab.asm, line 47` (break always (currently 0)) and `DebugLab.asm, line 51` (break always (currently 1)).
- Diagnostic Tools Window:** Shows a summary of diagnostics, including Events, Memory Usage, and CPU Usage.



## Part 5:

**Question 1:** As the OpenInputFile procedure returns the file handle in the EAX register, which is then moved to the fileHandle identifier, the file handle of TestText.txt in my execution of the lab program must be 0x00000108.

The screenshot shows the Visual Studio 2022 IDE during a debugging session. The main window displays assembly code for a program. The code includes a function `fileReadError` and a `main` procedure. The `main` procedure calls `greetUser`, attempts to open an input file, and checks if the file path is correct. The `Watch` window shows the values of `EAX` and `fileHandle`, both set to `0x00000108`. The `Breakpoints` window shows a breakpoint set at `DebugLab.asm, line 34`. The `Diagnostic Tools` window shows the `Events` tab, `Process Memory` usage, and `CPU Usage`.