Elizabeth House

T00738901

08/13/2024

COMP 2131 – Intro to Computer Systems

Programming Assignment #3

References:

<https://www.tutorialspoint.com/assembly_programming/assembly_system_calls.htm>

<https://asm.sourceforge.net/articles/linasm.html>

<https://www.educative.io/answers/what-are-macros-in-assembly-language>

<https://www.geeksforgeeks.org/how-to-measure-time-taken-by-a-program-in-c/>

Step 1:

A screen shot of a computer program

Description automatically generated

\*\* I set up Windows Subsystem for Linux to be able to run and test this since I don’t have access to a Linux machine. I hope this is acceptable. \*\*

I also couldn’t get the macro working with 3 separate lines for some reason, so I have it in 1 line with escape sequences. I know this will probably get me points off, but it at least works.

Step 2:

*The do-while loop appears to be the most efficient*

*The goto is 2nd most efficient*

*The while and for loops are both tied for least efficient*

**Do-While:**

**Number of Registers Used:** Esp, edx, ecx, eax, ebp – ***5 registers***

**Number of Jumps:** jle, jne – ***2 jumps***

**Total Number of Operations:** ***23 total operations***

LFB12: movl, movl, movl – 3 operations

L2: addl, addl, cmpl, jle, ret – 5 operations

LFB13: pushl, movl, andl. subl, call, movl – 6 operations

L5: addl, cmpl, jne, movl, movl, call, movl, leave, ret – 9 operations

**While:**

**Number of Registers Used:** esp, edx, ecx, eax, ebp – ***5 registers***

**Number of Jumps:** jg, jge, jne – ***3 jumps***

**Total Number of Operations: *27 total operations***

LFB12: movl, movl, cmpl, jg, movl – 5 operations

L3: addl, addl, cmpl, jge, ret – 5 operations

L4: movl, ret – 2 operations

LFB13: pushl, movl, andl, subl, call, movl – 6 operations

L7: addl, cmpl, jne, movl, movl, call, movl, leave, ret – 9 operations

**For:**

**Number of Registers Used:** esp, edx, ecx, eax, ebp – ***5 registers***

**Number of Jumps:** jg, jge, jne – ***3 jumps***

**Total Number of Operations: *27 total operations***

LFB12: movl, movl, cmpl, jg, movl – 5 operations

L3: addl, addl, cmpl, jge, ret – 5 operations

L4: movl, ret – 2 operations

LFB13: pushl, movl, andl, subl, call, movl – 6 operations

L7: addl, cmpl, jne, movl, movl, call, movl, leave, ret – 9 operations

**Goto:**

**Number of Registers Used:** esp, edx, ecx, eax, ebp – ***5 registers***

**Number of Jumps:** jg, jge, jne – ***3 jumps***

**Total Number of Operations: *25 total operations***

LFB12: movl, movl, movl, cmpl, jg – 5 operations

L5: addl, addl, cmpl, jge – 4 operations

L1: ret – 1 operation

LFB13: pushl, movl, andl, subl, call, movl – 6 operations

L7: addl, cmpl, jne, movl, movl, call, movl, leave, ret – 9 operations

Step 3:

I wasn’t sure exactly what was required here so I made the sum of an array of 50 elements but did different numbers of iterations on it to show the performance improvements (I used the string length graph as a guideline)

A graph with blue and purple lines

Description automatically generated

**Output below**

100,000 iterations

A screen shot of a computer program

Description automatically generated

200,000 iterations

A screen shot of a computer program

Description automatically generated

300,000 iterations

A screen shot of a computer program

Description automatically generated

400,000 iterations

A screen shot of a computer program

Description automatically generated

500,000 iterations

A screen shot of a computer program

Description automatically generated