Jennifer Wourms

HW2 Part 3

Question 1

|  |  |
| --- | --- |
| **Assembly Code** | **Interpretation** |
| 1. push ebp 2. mov ebp, esp 3. and esp, 0FFFFFFF0h 4. sub esp, 20h 5. call \_\_\_main 6. mov dword ptr [esp+1Ch], 3 7. mov dword ptr [esp+18h], 5 8. mov dword ptr [esp+14h], 0 9. mov eax, [esp+1Ch] 10. imul eax, [esp+18h] 11. mov edx, eax 12. mov eax, [esp+1Ch] 13. mov ecx, eax 14. shr ecx, 1Fh 15. add eax, ecx 16. sar eax, 1 17. sub edx, eax 18. mov eax, edx 19. mov [esp+14h], eax 20. mov eax, [esp+14h] 21. mov [esp+4], eax 22. mov dword ptr [esp], offset aD ; "%d" 23. call \_\_printf 24. mov eax, 0 25. leave 26. retn 27. \_main endp | Push base/frame pointer and copy stack pointer (esp) into ebp. Now ebp points toward function's stack frame.  Rounds stack pointer to nearest multiple of 16  subtract 20h to make room for local variables  run \_\_\_main function  store 3 into ptr address offset by 28  store 5 into ptr address offset by 24  store 0 into ptr address offset by 20  eax = 3  eax = 3 \* 5 = 15  edx = 15  eax = 3  ecx = 3  shift ecx right 31 bits (isolate signed bit) so ecx = 0 since it was not negative  eax = 3 + 0 = 3  shift eax right one place, eax = 1  edx = 15 - 1 = 14  eax = 14  ptr address offset by 20 = 14  eax = 14  ptr address offset by 4 = 14  "%d"  run \_\_printf function  eax = 0  esp = ebp, pop ebp. exit function  return to address at 24.  end program. |

**Output**:

14

Question 2

|  |  |
| --- | --- |
|  |  |
| push ebp  mov ebp, esp  and esp, 0FFFFFFF0h  sub esp, 40h  call \_\_\_main  mov dword ptr[esp+18h], 0Ch  mov dword ptr[esp+1Ch], 0Fh  mov dword ptr[esp+20h], 0DDh  mov dword ptr[esp+24h], 3  mov dword ptr[esp+28h], 1B0h  mov dword ptr[esp+2Ch], 36h  mov dword ptr[esp+30h], 10h  mov dword ptr[esp+34h], 43h  mov dword ptr[esp+3Ch], 0  mov dword ptr[esp+38h], 0  jmp short loc\_40157F  cmp dword ptr[esp+38h], 7  jle short loc\_401560  mov eax, [esp+38h]  mov eax, [esp+eax\*4+18h]  cmp eax, [esp+3Ch]  jle short loc\_40157A  mov eax, [esp+38h]  mov eax, [esp+eax\*4+18h]  mov [esp+3Ch], eax  add dword ptr[esp+38h], 1  cmp dword ptr[esp+38h], 7  jle short loc\_401560  mov eax, [esp+38h]  mov eax, [esp+eax\*4+18h]  cmp eax, [esp+3Ch]  jle short loc\_40157A  mov eax, [esp+38h]  mov eax, [esp+eax\*4+18h]  mov [esp+3Ch], eax  add dword ptr[esp+38h], 1  cmp dword ptr[esp+38h], 7  jle short loc\_401560  mov eax, [esp+38h]  mov eax, [esp+eax\*4+18h]  cmp eax, [esp+3Ch]  jle short loc\_40157A  mov eax, [esp+38h]  mov eax, [esp+eax\*4+18h]  mov [esp+3Ch], eax  add dword ptr[esp+38h], 1  cmp dword ptr[esp+38h], 7  jle short loc\_401560  mov eax, [esp+38h]  mov eax, [esp+eax\*4+18h]  cmp eax, [esp+3Ch]  jle short loc\_40157A  mov eax, [esp+38h]  mov eax, [esp+eax\*4+18h]  mov [esp+3Ch], eax  add dword ptr[esp+38h], 1  cmp dword ptr[esp+38h], 7  jle short loc\_401560  mov eax, [esp+38h]  mov eax, [esp+eax\*4+18h]  cmp eax, [esp+3Ch]  jle short loc\_40157A  mov eax, [esp+38h]  mov eax, [esp+eax\*4+18h]  mov [esp+3Ch], eax  add dword ptr[esp+38h], 1  cmp dword ptr[esp+38h], 7  jle short loc\_401560  mov eax, [esp+38h]  mov eax, [esp+eax\*4+18h]  cmp eax, [esp+3Ch]  jle short loc\_40157A  mov eax, [esp+38h]  mov eax, [esp+eax\*4+18h]  mov [esp+3Ch], eax  add dword ptr[esp+38h], 1  cmp dword ptr[esp+38h], 7  jle short loc\_401560  mov eax, [esp+38h]  mov eax, [esp+eax\*4+18h]  cmp eax, [esp+3Ch]  jle short loc\_40157A  mov eax, [esp+38h]  mov eax, [esp+eax\*4+18h]  mov [esp+3Ch], eax  add dword ptr[esp+38h], 1  cmp dword ptr[esp+38h], 7  jle short loc\_401560  mov eax, [esp+38h]  mov eax, [esp+eax\*4+18h]  cmp eax, [esp+3Ch]  jle short loc\_40157A  mov eax, [esp+38h]  mov eax, [esp+eax\*4+18h]  mov [esp+3Ch], eax  add dword ptr[esp+38h], 1  cmp dword ptr[esp+38h], 7  jle short loc\_401560  mov eax, [esp+3Ch]  mov [esp+4], eax  mov dword ptr[esp], offset aD; "%d"  call \_\_printf  mov eax, 0  leave  retn  endp | Push base/frame pointer and copy stack pointer (esp) into ebp. Now ebp points toward function's stack frame.  Rounds stack pointer to nearest multiple of 16  subtract 40h to make room for local variables  run \_\_\_main function  **ptr at offset 24 = 12**  **ptr at offset 28 = 15**  **ptr at offset 32 = 221**  **ptr at offset 36 = 3**  **ptr at offset 38 = 432**  **ptr at offset 44 = 54**  **ptr at offset 48 = 16**  **ptr at offset 52 = 67**  ptr at offset 60 = 0  ptr at offset 56 = 0  jump to loc\_40157F  compare 0 to 7  jump if less or equal (true) to loc\_401560  eax = 0  eax = 0 + 24 = 24  compare 24 to 0  jump if less or equal (false) to loc\_40157A  eax = 0  eax = 24  ptr at offset 60 = 24  ptr at offset 56 = 1  compare 1 to 7  jump if less or equal (true) to loc\_401560  eax = 1  eax = 4 + 24 = 28  compare 28 to 24  jump if less or equal (false) to loc\_40157A  eax = 1  eax = 28  ptr at offset 60 = 28  ptr at offset 56 = 2  compare 2 to 7  jump if less or equal (true) to loc\_401560  eax = 2  eax = 8 + 24 = 32  compare 32 to 28  jump if less or equal (false) to loc\_40157A  eax = 2  eax = 32  ptr at offset 60 = 32  ptr at offset 56 = 3  compare 3 to 7  jump if less or equal (true) to loc\_401560  eax = 3  eax = 12 + 24 = 36  compare 36 to 32  jump if less or equal (false) to loc\_40157A  eax = 3  eax = 36  ptr at offset 60 = 36  ptr at offset 56 = 4  compare 4 to 7  jump if less or equal (true) to loc\_401560  eax = 4  eax = 16 + 24 = 40  compare 40 to 36  jump if less or equal (false) to loc\_40157A  eax = 4  eax = 40  ptr at offset 60 = 40  ptr at offset 56 = 5  compare 5 to 7  jump if less or equal (true) to loc\_401560  eax = 5  eax = 20 + 24 = 44  compare 44 to 40  jump if less or equal (false) to loc\_40157A  eax = 5  eax = 44  ptr at offset 60 = 44  ptr at offset 56 = 6  compare 6 to 7  jump if less or equal (true) to loc\_401560  eax = 6  eax = 24 + 24 = 48  compare 48 to 44  jump if less or equal (false) to loc\_40157A  eax = 6  eax = 48  ptr at offset 60 = 48  ptr at offset 56 = 7  compare 7 to 7  jump if less or equal (true) to loc\_401560  eax = 7  eax = 28 + 24 = 52  compare 52 to 48  jump if less or equal (false) to loc\_40157A  eax = 7  eax = 52  **ptr at offset 60 = 52**  **ptr at offset 56 = 8**  compare 8 to 7  jump if less or equal (false) to loc\_401560  eax = 52  **ptr at offset 4 = 52**  assign ptr[esp] to print  call printf function  eax = 0  exit function  return to line below last call  end program |

**Output**

52Question 3

|  |  |
| --- | --- |
| **Assembly Code** | **Interpretation** |
| push ebp  mov ebp, esp  and esp, 0FFFFFFF0h  sub esp, 20h  call \_\_\_main  mov dword ptr[esp+1Ch], 64h  jmp loc\_4015D6  cmp dword ptr[esp+1Ch], 3E7h  jle loc\_40151B  mov ecx, [esp+1Ch]  mov edx, 51Eb851Fh  mov eax, ecx  imul edx  sar edx, 5  mov eax, ecx  sar eax, 1Fh  sub edx, eax  mov eax, edx  mov [esp+18h], eax  mov eax, [esp+18h]  imul edx, eax, -64h  mov eax, [esp+1Ch]  lea ecx, [edx+eax]  mov edx, 66666667h  mov eax, ecx  imul edx  sar edx, 2  mov eax, ecx  sar eax, 1Fh  sub edx, eax  mov eax, edx  mov [esp+14h], eax  mov ecx, [esp+1Ch]  mov edx, 66666667h  mov eax, ecx  imul edx  sar edx, 2  mov eax, ecx  sar eax, 1Fh  sub edx, eax  mov eax, edx  shl eax, 2  add eax, edx  add eax, eax  sub ecx, eax  mov eax, ecx  mov [esp+10h], eax  mov eax, [esp+18h]  imul eax, [esp+18h]  imul eax [esp+18h]  mov edx, eax  mov eax, [esp+14h]  imul eax, [esp+14h]  imul eax, [esp+14h]  add edx, eax  mov eax, [esp+10h]  imul eax, [esp+10h]  imul eax, [esp+10h]  add eax, edx  cmp eax, [esp+1Ch]  jnz short loc\_4015D1  add dword ptr[esp+1Ch], 1  cmp dword ptr[esp+1Ch], 3E7h  jle loc\_40151B  mov eax, 0  leave  retn  endp | Push base/frame pointer and copy stack pointer (esp) into ebp. Now ebp points toward function's stack frame.  Rounds stack pointer to nearest multiple of 16  subtract 20h to make room for local variables  run \_\_\_main function  ptr at offset 28 = 100  jump to loc\_4015D6  compare 100 to 999  if less than or equal (true), jump to loc\_40151B  ecx = 100  edx = 1374389535  eax = 100  edx = 137438953500  edx = 4294967296  eax = 100  eax = 0  edx = 4294967296  eax = 4294967296  ptr at offset 24 = 4294967296  eax = 4294967296  edx = 4294967296 \* (-100) = -429496729600  eax = 100  ecx = -425201762304  edx = 1717986919  eax = -425201762304  edx = 1717986919 \* -425201762304 = -7.3049107e+20  edx = 2305843009213693951  eax = -425201762304  eax = -198  edx = 2.305843e+18  eax = 2.305843e+18  ptr at offset 20 = 2.305843e+18  ecx = 100  edx = 1717986919  eax = 100  edx = 171798691900  edx = 42949672975  eax = 100  eax = 0  edx = 171798691900  eax = 171798691900  eax = 687194767600  eax = 858993459500  eax = 1.7179869e+12  ecx = -1.7179869e+12  eax = -1.7179869e+12  ptr at offset 16 = -1.7179869e+12  eax = 4294967296  eax = 1.8446744e+19  eax = 7.9228162e+28  edx = 1.8446744e+19  eax = 2.305843e+18  eax = 5.3169119e+36  eax = 1.2259964e+55  edx = 2.4519928e+55  eax = -1.7179869e+12  eax = 2.951479e+24  eax = -5.0706023e+36  eax = 2.4519928e+55  compare 2.4519928e+55 to 100  jump if not zero (true)  ptr at offset 28 = 101  compare 101 to 999  jump if less than or equal (true) to loc\_40151B  \*\*REPEAT WHOLE THING UNTIL ptr at offest 28 is 1000\*\*  eax = 0  leave function  return to line before last call (printf)  end program |

**Output**