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

|  |  |
| --- | --- |
| **Assembly Code** | **Interpretation** |
| 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  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  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  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  add dword ptr[esp+38h], 1  cmp dword ptr[esp+38h], 7  jle short loc\_401560  mov eax, [esp+3Ch]  mov eax, [esp+eax\*4+18h]  cmp eax, [esp+3Ch]  jle short loc\_40157A  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 40 = 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 = ptr[offset 24] = 12  compare 0 to 0  jump if less or equal (true) to loc\_40157A  ptr at offset 56 = 1  compare 1 to 7  jump if less or equal (true) to loc\_401560  eax = 1  eax = ptr[offset 28] = 15  compare 15 to 0  jump if less or equal (false) to loc\_40157A  eax = 1  eax = 15  ptr at offset 60 = 15  ptr at offset 56 = 2  compare 2 to 7  jump if less or equal (true) to loc\_401560  eax = 2  eax = ptr[32] = 221  compare 221 to 15  jump if less or equal (false) to loc\_40157A  eax = 2  eax = 221  ptr at offset 60 = 221  ptr at offset 56 = 3  compare 3 to 7  jump if less or equal (true) to loc\_401560  eax = 3  eax = ptr[36] = 3  compare 3 to 221  jump if less or equal (true) to loc\_40157A  ptr at offset 56 = 4  compare 4 to 7  jump if less or equal (true) to loc\_401560  eax = 4  eax = ptr[40] = 432  compare 432 to 221  jump if less or equal (false) to loc\_40157A  eax = 4  eax = 432  **ptr at offset 60 = 432**  ptr at offset 56 = 5  compare 5 to 7  jump if less or equal (true) to loc\_401560  eax = 5  eax = ptr[44] = 54  compare 54 to 432  jump if less or equal (true) to loc\_40157A  ptr at offset 56 = 6  compare 6 to 7  jump if less or equal (true) to loc\_401560  eax = 6  eax = ptr[48] = 16  compare 16 to 432  jump if less or equal (true) to loc\_40157A  ptr at offset 56 = 7  compare 7 to 7  jump if less or equal (true) to loc\_401560  eax = 7  eax = ptr[52] = 67  compare 67 to 432  jump if less or equal (true) to loc\_40157A  ptr at offset 56 = 8  compare 8 to 7  jump if less or equal (false) to loc\_401560  eax = 432  **ptr at offset 4 = 432**  set up 432 to print  call printf function  eax = 0  exit function  return to line below printf call  end program |

**Output**

432Question 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 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 or equal (true) jump to loc\_40151B  ecx = 100  edx =1374389535  eax = 100  edx = 137438953500  edx = 4294967296  eax = 100  eax = 0  edx = 4294967296 - 0 = 4294967296  eax = 4294967296  **ptr at offset 24 = 4294967296**  eax = 4294967296  edx = 4294967296 \* -100 = -429496729600  eax = 100  ecx = -4294967296 + 100 = -4294967196  edx = 1717986919  eax = -4294967196  edx = -4294967196 \* 1717986919 = -7.3786975e+18  edx = -1844674375000000000  eax = -4294967196  eax = 1  edx = -1844674375000000000 - 1 = -1844674375000000001  eax = -1844674375000000001  **ptr at offset 20 = -1844674375000000001**  ecx = 100  edx = 1717986919  eax = 100  edx = 171798691900  edx = 42949672975  eax = 100  eax = 0  edx = 42949672975 - 0 = 42949672975  eax = 42949672975  eax = 171798691900  eax = 171798691900 + 42949672975 = 214748364875  eax = 214748364875 + 214748364875 = 429496729750  ecx = 100 - 429496729750 = -42949672875  eax = -42949672875  **ptr at offset 16 =** -**42949672875**  eax = 4294967296  eax = 4294967296 \* 4294967296 = 1.8446744e+19  eax = 1.8446744e+19 \* 4294967296 = 7.9228162e+28  edx = 7.9228162e+28  eax = -1844674375000000001  eax = -1844674375000000001 \* -1844674375000000001 = 3.4028235e+36  eax = 3.4028235e+36 \* -1844674375000000001 = -6.2771013e+54  edx = -6.2771013e+54  eax = -42949672875  eax = -42949672875 \* -42949672875 = 1.8446744e+21  eax = 1.8446744e+21 \* -42949672875 = -7.9228162e+31  eax = -6.2771013e+54  compare -6.2771013e+54 to 100  **jump if not zero (true) to loc\_4015D1**  **ptr at offset 28 = 101**  compare 101 to 999  if less or equal (true) jump to jle loc\_40151B  \*\*\*REPEAT\*\*\*  ecx = 101  edx = 1374389535  eax = 101  edx = 1374389535 \* 101 = 138813343035  edx = 4337916969  eax = 101  eax = 0  edx = 4337916969 - 0 = 4337916969  eax = 4337916969  **ptr at offset 24 = 4337916969**  eax = 4337916969  edx = 4337916969 \* -100 = -433791696900  eax = 101  ecx = -433791696900 + 101 = -433791696799  edx = 1717986919  eax = -433791696799  edx = -433791696799 \* 1717986919 = -7.4524846e+20  edx = -2305843009213693952  eax = -433791696799  eax = 1  edx = -2305843009213693952 - 1 = -2305843009213693953  eax = -2305843009213693953  **ptr at offset 20 = -2305843009213693953**  ecx = 101  edx = 1717986919  eax = 101  edx = 171798691900  edx = 42949672975  eax = 101  eax = 0  edx = 42949672975  eax = 42949672975  eax = 171798691900  eax = 171798691900 + 42949672975 = 214748364875  eax = 214748364875 + 214748364875 = 429496729750  ecx = 101 - 429496729750 = -429496729649  eax = -429496729649  **ptr at offset 16 =** -**429496729649**  eax = 4337916969  eax = 4337916969 \* 4337916969 = 1.8817524e+19  eax = 1.8817524e+19 \* 4337916969 = 8.1628857e+28  edx = 8.1628857e+28  eax = -2305843009213693953  eax = -2305843009213693953 \* -2305843009213693953 = 5.316912e+36  eax = 5.316912e+36 \* -2305843009213693953 = -1.2259964e+55  edx = 8.1628857e+28 + -1.2259964e+55 = -1.2259964e+55  eax = -429496729649  eax = -429496729649 \* -429496729649 = 1.8446744e+23  eax = 1.8446744e+23 \* -429496729649 = -7.9228162e+34  eax = -1.2259964e+55  compare -1.2259964e+55 to 101  if not zero (true) jump to loc\_4015D1  **ptr at offset 28 = 102**  compare 102 to 999  if less or equal (true) jump to loc\_40151B  \*\*\*REPEAT\*\*\*  this repeats until eax = [esp+1Ch]  the program then prints out whatever that eax is, which is the some index between 100 and 999  eax = 0  exit function  return to last call  end program |

**Output**

Question 4

|  |  |
| --- | --- |
| **Assembly Code** | **Interpretation** |
| mov dword ptr [esp+1A8h], 7  mov dword ptr [esp+1A4h], 64h  mov dword ptr [esp+1ACh], 0  jmp short loc\_401619  mov eax, [esp+1ACh]  cmp eax, [esp+1A4h]  jl short loc\_4015FC  mov eax, [esp+1ACh]  lea edx, [eax+1]  mov eax, [esp+1ACh]  mov [esp+eax\*4+14h], edx  add dword ptr [esp+1ACh], 1  mov eax, [esp+1ACh]  cmp eax, [esp+1A4h]  jl short loc\_4015FC  mov eax, [esp+1ACh]  lea edx, [eax+1]  mov eax, [esp+1ACh]  mov [esp+eax\*4+14h], edx  add dword ptr [esp+1ACh], 1  mov eax, [esp+1ACh]  cmp eax, [esp+1A4h]  jl short loc\_4015FC  cmp eax, [esp+1A4h]  jl short loc\_4015FC  mov eax, [esp+1A8h]  mov [esp+8], eax  mov eax, [esp+1A4h]  mov [esp+4], eax  lea eax, [esp+1B0h+var\_19C]  mov [esp], eax  call \_\_Z5proc1Piii   |  | | --- | | mov [ebp+var\_C], 0  mov [ebp+var\_10], 0  mov [ebp+var\_4], 0  jmp loc\_401587  mov eax, [ebp+var\_4]  cmp eax, [ebp+arg\_4]  jl loc\_401520  mov [ebp+var\_8], 1  jmp short loc\_40155E  mov eax, [ebp+var\_8]  cmp eax, [ebp+arg\_8]  jl short loc\_401529  jmp short loc\_401538  mov eax, [ebp+var\_C]  lea edx, ds:0[eax\*4]  mov eax, [ebp+arg\_0]  add eax, edx  mov eax, [eax]  test eax, eax  jz short loc\_40152B  add [ebp+var\_8], 1  mov eax, [epb+var\_C]  add eax, 1  cdq  idiv [ebp+arg\_4]  mov eax, [ebp+var\_8]  cmp eax, [ebp+arg\_8]  jl short loc401529  jmp short loc\_401575  mov eax [ebp+var\_C]  lea edx, ds:0[eax\*4]  mov eax, [ebp+arg\_0]  add eax, edx  mov eax, [eax]  test eax, eax  jz short loc\_401568  mov eax, [ebp+var\_C]  lea edx, ds:0[eax\*4]  mov eax, [ebp+arg\_0]  add eax, edx  mov eax, [eax]  mov [ebp+var\_10], eax  mov eax, [ebp+var\_C]  lea edx, ds:0[eax\*4]  mov eax, [ebp+arg\_0]  add eax, edx  mov dword ptr [eax], 0  add [ebp+var\_4], 1  mov eax, [ebp+var\_4]  cmp eax, [ebp+arg\_4]  jl loc\_401520  jz short loc\_401568  mov eax, [ebp+var\_C]  add eax, 1  cdq  idiv [ebp+arg\_4]  mov [ebp+var\_C], edx  mov eax [ebp+var\_C]  lea edx, ds:0[eax\*4]  mov eax, [ebp+arg\_0]  add eax, edx  mov eax, [eax]  test eax, eax  jz short loc\_401568  mov eax, [ebp+var\_C]  lea edx, ds:0[eax\*4]  mov eax, [ebp+arg\_0]  add eax, edx  mov eax, [eax]  mov [ebp+var\_10], eax  mov eax, [ebp+var\_C]  lea edx, ds:0[eax\*4]  mov eax, [ebp+arg\_0]  add eax, edx  mov dword ptr [eax], 0  add [ebp+var\_4], 1 |   mov [esp+4], eax  mov dword ptr [esp], offset aD  call \_\_printf  mov eax, 0  leave  retn  \_\_main endp | **ptr at offset 424 = 7**  **ptr at offset 420 = 100**  **ptr at offset 428 = 0**  jump to loc\_401619  eax = 0  compare 0 to 100  if less (true) jump to loc\_4015FC  eax = 0  edx = 1  eax = 0  **ptr at offset 20 = 1**  **ptr at offset 428 = 1**  eax = 1  compare 1 to 100  if less (true) jump to loc\_4015FC  eax = 1  edx = 2  eax = 1  **ptr at offset 24 = 2**  **ptr at offset 428 = 2**  eax = 2  compare 2 to 100  if less (true) jump to loc\_4015FC  \*\* REPEAT UNTIL eax = 100\*\*  at this point:  **ptr at offset 416 = 100**  **ptr at offset 428 = 100**  compare 100 to 100  if less (false) jump to loc\_4015FC  eax = 7  **ptr at offset 8 = 7**  eax = 100  **ptr at offset 4 = 100**  eax = 432 + (-412) = 20  **ptr = 20**   |  | | --- | | **ptr at offset -12 = 0**  **ptr at offset -16 = 0**  **ptr at offset -4 = 0**  jump to loc\_401587  eax = 0  compare 0 to 12  if less (true) jump to loc\_401520  **ptr at offset -8 = 1**  jump to loc\_40155E  eax = 1  compare 1 to 16  if less (true) jump to loc\_401529  jump to loc\_401538  eax = 0  edx = 0  eax = 8  eax = 8  mov eax = 1  test 1 1 = 1  if zero (false) jump to loc\_40152B  **ptr at offset -8 = 2**  eax = 0  eax = 1  \*\*convert double to quad\*\*  eax = 0  eax = 2  compare 2 to 16  if less (true) jump to loc401529  \*\*\*REPEAT until [ebp+var\_8] = 16\*\*\*  at this point **ptr at offset -8 = 16**  if less (false) jump to loc401529  jump to loc\_401575  \*\*\* This is the same process as before\*\*\*  we already know that this is false so we do not jump to loc\_401568  eax = 0  edx = 0  eax = 8  eax = 8  eax = 1  **ptr at offset -16 = 1**  eax = 0  edx = 0  eax = 8  eax = 8  **ptr = 0**  **ptr at offset -4 = 1**  eax = 1  compare 1 to 12  if less (true) jump to loc\_401575  \*\*\* REPEAT until eax = 12\*\*\*  jump if zero (true) to loc\_401568  FALSE | |

**Output**

Unsure... but it will print out some integer...