push ebp

Push the base ptr onto the stack

Move the stack pointer onto the base

Aligns stack on 16 byte boundary

Subtracts space for local variables

Calls main

Moves 3 into esp+1C

Moves 5 into esp+18

Moves 0 into esp +14

Moves 3 into eax

Multiplies eax (3) by 5

Moves eax into edx, the data register

Moves 3 into eax

Moves eax into ecx, the counter register (3)

Shifts bits of ecx right by 31, so just biggest bit

Adds ecx to eax

Shift arithmetic right by 1, discard biggest bit, /2

Subtract eax from edx

Move edx into eax

Move eax into esp+14

Move esp+14 into eax?

Move eax into esp+4

Calls

Moves 0 into eax

Exit program

mov ebp, esp

and esp, 0FFFFFFF0h

sub esp, 20h

call \_\_\_main

mov dword ptr [esp+1Ch], 3

mov dword ptr [esp+18h], 5

mov dword ptr [esp+14h], 0

mov eax, [esp+1Ch]

imul eax, [esp+18h]

mov edx, eax

mov eax, [esp+1Ch]

mov ecx, eax

shr ecx, 1Fh

add eax, ecx

sar eax, 1

sub edx, eax

mov eax, edx

mov [esp+14h], eax

mov eax, [esp+14h]

mov [esp+4], eax

mov dword ptr [esp], offset aD ; "%d"

call \_printf

mov eax, 0

leave

retn

\_main endp

push ebp

Push base ptr onto stack

Move stack ptr onto base ptr

Aligns stack on 16 byte boundary

Decrement stack ptr

Call main

{12, 15, 221, 3, 432, 54, 16, 67}

X = 0

Eax = X

Eax = Xth integer in the array

Compare eax to last integer in array

If eax<=last int jump to loc

If greater, swaps the two values so that the last value in the array is the greatest value

Add 1 to x

Compare x to 7

If <7 jmp to loc 401560

Eax = last int in array

Prepare to print array

Call printf on array

Zero eax

Exit program

mov ebp, esp

and esp, 0FFFFFFFF0h

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+34h], 10h

mov dword ptr[esp+3Ch], 43h

mov dword ptr[esp+38h], 0

jmp short loc\_40157F

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

loc\_40157A:

add dword ptr[esp+38h],1

loc\_40157F

cmp dword ptr[esp+38h],7

jle short loc\_401560

mov eax,[esp+3Ch]

mov dword ptr[esp], offset aD ; “%d”

call \_printf

mov eax,0

leave

retn

endp

text:00401500 push ebp

Push base ptr onto stack

Move stack ptr onto base ptr

Aligns stack on 16 byte boundary

Decrement stack ptr

Call main

X = 100

Jump to 4015D6

Ecx = X

Edx = 1374389535

Eax = X

Multiply eax\*edx so x\*1374389535

Divide result of last calc by 32

Eax = x

Shift bits right by 31, so get sign

Subtract eax from edx (give it correct sign)

Eax = edx

Y = eax (result from calcs)

Move back into eax

Edx = Multiply result by -100

Eax, x

Ecx = memory at edx+eax

Edx = 1717986919

Eax = memory at edx+eax

Edx = edx\*eax

Divide this result by 4

Eax = ecx

Get sign of eax

Set sign of edx

Move number into eax

Z = eax

Ecx = x

Edx = 1717986919

Eax = x

Edx = edx\*eax

Divide edx by 4

Eax = x

Get sign of x

Set sign of result

Eax = edx

Multiply by 4

Add edx to eax

Add eax to itself

Subtract this from ecx

.text:00401501 mov ebp, esp

.text:00401503 and esp, 0FFFFFFF0h

.text:00401506 sub esp, 20h

.text:00401509 call \_\_\_main

.text:0040150E mov dword ptr [esp+1Ch], 64h

.text:00401516 jmp loc\_4015D6

.text:0040151B ; ---

.text:0040151B

.text:0040151B loc\_40151B

.text:0040151B mov ecx, [esp+1Ch]

.text:0040151F mov edx, 51EB851Fh

.text:00401524 mov eax, ecx

.text:00401526 imul edx

.text:00401528 sar edx, 5

.text:0040152B mov eax, ecx

.text:0040152D sar eax, 1Fh

.text:00401530 sub edx, eax

.text:00401532 mov eax, edx

.text:00401534 mov [esp+18h], eax

.text:00401538 mov eax, [esp+18h]

.text:0040153C imul edx, eax, -64h

.text:0040153F mov eax, [esp+1Ch]

.text:00401543 lea ecx, [edx+eax]

.text:00401546 mov edx, 66666667h

.text:0040154B mov eax, ecx

.text:0040154D imul edx

.text:0040154F sar edx, 2

.text:00401552 mov eax, ecx

.text:00401554 sar eax, 1Fh

.text:00401557 sub edx, eax

.text:00401559 mov eax, edx

.text:0040155B mov [esp+14h], eax

.text:0040155F mov ecx, [esp+1Ch]

.text:00401563 mov edx, 66666667h

.text:00401568 mov eax, ecx

.text:0040156A imul edx

.text:0040156C sar edx, 2

.text:0040156F mov eax, ecx

.text:00401571 sar eax, 1Fh

.text:00401574 sub edx, eax

.text:00401576 mov eax, edx

.text:00401578 shl eax, 2

.text:0040157B add eax, edx

.text:0040157D add eax, eax

.text:0040157F sub ecx, eax

.text:00401581 mov eax, ecx

Eax = ecx

A = eax

Eax = y

Eax = y^2

Eax = y^3

Edx = eax

Eax = Z

Eax = z^2

Eax = z^3

Edx = z^3 + y^3

Eax = A

Eax = A^2

Eax = A^3

Eax = edx + A^3

Compare eax = X

If they’re not equal jump to 4015D1

Eax = x

Setting print value to X

Prepare to print X

Add 1 to X

Compare X to 999

If x < 999 jump to 40151B

Else Eax = 0

Exit program

.text:00401583 mov [esp+10h], eax

.text:00401587 mov eax, [esp+18h]

.text:0040158B imul eax, [esp+18h]

.text:00401590 imul eax, [esp+18h]

.text:00401595 mov edx, eax

.text:00401597 mov eax, [esp+14h]

.text:0040159B imul eax, [esp+14h]

.text:004015A0 imul eax, [esp+14h]

.text:004015A5 add edx, eax

.text:004015A7 mov eax, [esp+10h]

.text:004015AB imul eax, [esp+10h]

.text:004015B0 imul eax, [esp+10h]

.text:004015B5 add eax, edx

.text:004015B7 cmp eax, [esp+1Ch]

.text:004015BB jnz short loc\_4015D1

.text:004015BD mov eax, [esp+1Ch]

.text:004015C1 mov [esp+4], eax

.text:004015C5 mov dword ptr [esp], offset aD ; "%d "

.text:004015CC call \_printf

.text:004015D1

.text:004015D1 loc\_4015D1

.text:004015D1 add dword ptr [esp+1Ch], 1

.text:004015D6

.text:004015D6 loc\_4015D6:

.text:004015D6 cmp dword ptr [esp+1Ch], 3E7h

.text:004015DE jle loc\_40151B

.text:004015E4 mov eax, 0

.text:004015E9 leave

.text:004015EA retn

.text:004015EA \_main endp