Ryan Arvizu

EECS 349: HW 2 Part 3

24 October 2019

Q1 (code in 1.c):

Functionality –

8 – 20 = -12 ; sub esp, 20h

Starting at address esp28 – 32bit binary representation of 3 ; mov dword ptr [esp+1Ch], 3

Starting at address esp18 – 32bit binary representation of 5 ; mov dword ptr [esp+18h], 5

Starting at address esp14 – 32bit binary representation 0 ; mov dword ptr [esp+14h], 0

-12 to esp28 (replacing 3) ; mov eax, [esp+1Ch]

Multiply -12 by 5 = -60 ; imul eax, [esp+18h]

-60 in edx ; mov edx, eax

-60 in eax ; mov eax, [esp+1Ch]

-60 in ecx ; mov ecx, eax

Shift right -60, 31 -> 1 in ecx ; shr ecx, 1Fh

Add -60 and -1 = -61 ; add eax, ecx

Shift arithmetic right -61 - > -30; sar eax, 1

-60 – (-30) = -30 in edx ; sub edx, eax

Move -30 to eax ; mov eax, edx

Move -30 to [esp+14h] ; mov [esp+14h], eax

Move -30 to eax ; mov eax, [esp+14h]

Move -30 to [esp+4] ; mov [esp+4], eax

Print -30 ; mov dword ptr [esp], offset aD and call\_printf

Return 0

Q2 (code in 2.c):

Functionality –

Subtract 40 from esp

[esp+18] = 12

[esp+12] = 15

[esp+20] = 221

[esp+24] = 3

[esp+28] = 432

[esp+44] = 36

[esp+30] = 10

[esp+34] = 43

[esp+60] = 0

[esp+38] = 0

Go to loc\_40157F:

Compare 0, 7

0 < 7 so jump to loc\_401560

0 in eax

12 in eax

Compare 12, 0

12 > 0 so keep going

0 in eax

12 in eax

12 in [esp+60]

12 in eax

Print 12

Q3 (code in 3.c):

Functionality –

Subtract 20 from es

[esp+12] = 64

Go to loc\_4015D6

Compare [esp+12] and 999

If 64 < 999 go to loc\_40151B

Else return 0

Loc\_40151B:

Ecx is 12

Edx is 01010001 11101011 10000101 00011111

Eax is 12

Multiply 12 by edx so edx is 011 (3) and eax is 11010111 00001010 00111101 01110100

Edx shift right 5 -> 0

Eax is 12

Eax shift right 5 -> 0

Eax is 0

[esp+18] is eax

Eax is [esp+18]

Store -64\*0 in edx

Eax is [esp+12]

Put memory address of edx+eax (esp+12 + 0) into ecx

Edx is 66666667

Eax is ecx

Eax is memory address \* 66666667

Shift right edx by 2 -> 16666666

Shift right eax by 31

Edx = edx – eax

Eax = edx

[esp+14] = eax

Ecx = [esp+28]

Edx = 66666667

Eax = ecx

Eax = eax \* edx

Shift right edx by 2

Eax = ecx

Shift right eax by 31

Edx = edx – eax

Eax = edx

Shift left eax by 2

Eax = eax + edx

Eax = eax + eax

Ecx = ecx – eax

Eax = ecx

[esp+10] = eax

Eax = [esp+18]

Eax = eax \* [exp+18]

Eax = eax \* [exp+18]

Edx = edx + eax

Eax = [esp+10]

Eax = eax \* [exp+10]

Eax = eax \* [exp+10]

Eax = eax + edx

If (eax != e[esp+28]) then go to 4015D1 ([esp+28] += 1)

Else

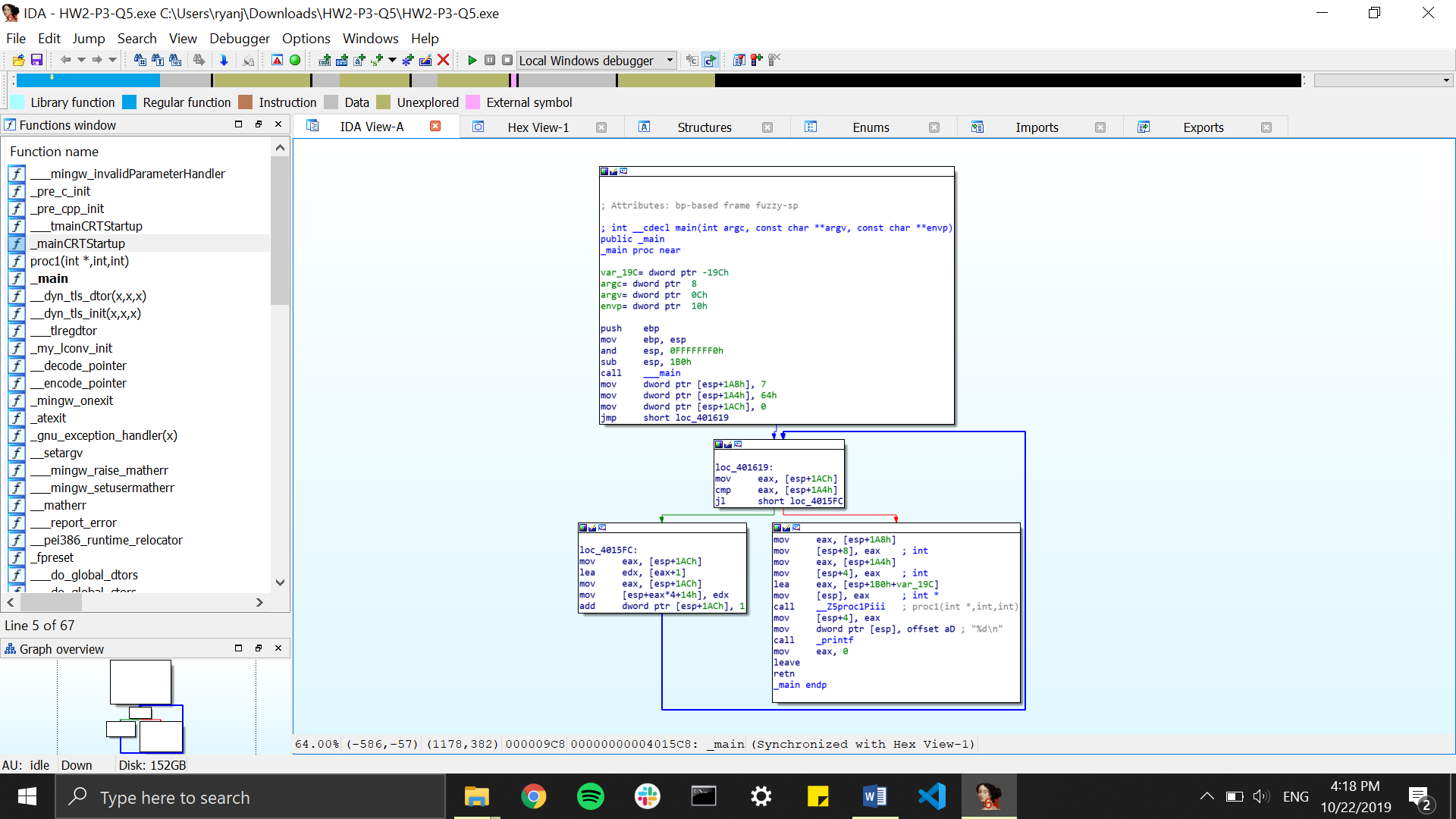
Eax = [esp+28]

Print eax

Return 0

Q4 (code in 4.c):

IDA Output



Functionality –

[esp+424] = 7

[esp+420]=64

[esp+428] = 0

Go to loc\_401619

a = [esp+428]

if (a < [esp+420]) { go to f1 }

else {

a = [esp+424]

[esp+8] = a;

A = [esp+420]

[esp+4] = a

a = memory address of [esp+432+var\_19C]

[esp] = a

Call proc1(int \*, int, int)

[esp+4] = a

Print a

Return 0;

f1:

a = [esp+420]

d = memory address of a + 1

a = [esp+420]

d = [esp+4\*a+14]

[esp+420] += 1