

Justin Shearson (jrs330)

EECS 349

Homework 2: Part 3

Question 1:

The program declares 3 variables (x, y & z). X and y are initialized to 3 & 5 respectively while z is initialized to 0. After the three variables are declared z is assigned to the value of:

$$z = (x * y) - \left(\frac{x}{2}\right)$$

Z is then printed to console with a value of 14.

Question 2:

The program declares an array with 8 elements (12,15,221,3,432,54,16,67) and two other variables which are initialized with a value of 0. Variable y is used as an index and x is used as a temporary value to store an element from the array. The element at the array index (y) is checked against x. If x is smaller than the array at index y, the element at index y is stored into x. The largest element in the array is then printed to console with a value of 432.

Question 3:

The function declares 4 variables (x, y, z, a, & b). X is initialized to a value of 100 and is used as a counter for a loop. Y is assigned to a value of x/100. Z is assigned to a value of (y* -100 + x) / 10. A is assigned to a value of (x - (2* (x/2))). B is assigned to a value of (y³ + z³ + a³). The program then checks if the value of x is equal to b. X is then incremented and loops until it reaches a value of 999.

Question 4:

Question 4 makes an array and passes it into a function call and returns a value of 49. The function runs 4 while loops that each perform mathematical equality checks as well as modulus division and then returns a value of 49.

```

1  public _main
2  _main proc near
3
4  argc= dword ptr 8
5  argv= dword ptr 0ch
6  envp= dword ptr 10h
7  ;This program makes three variables, multiplies the first by the second variable
8  ; X * Y
9  ; X/2
10 ; z = (x*y)-(x/2)
11 push ebp
12 mov ebp, esp
13 and esp, 0FFFFFFF0h ;Just aligning stack
14 sub esp, 20h ;Reserve space for local variables
15 call _main ;Calls the main function in c
16 mov dword ptr [esp+1ch], 3 ;X: Creates a variable and assigns a value of 3 to it
17 mov dword ptr [esp+18h], 5 ;Y: Creates a variable and assigns a value of 5 to it
18 mov dword ptr [esp+14h], 0 ;Z: Creates a variable and assigns a value of 0 to it
19 mov eax, [esp+1ch] ;Moves the first value into the accumulator
20 imul eax, [esp+18h] ;Multiplies the second by the first
21 mov edx, eax ;Moves the value into the edx register
22 mov ecx, [esp+1ch] ;Moves the second value into the eax register
23 mov ecx, eax ;Stores that value into ecx (temp register)
24 shr ecx, 1fh ;Performs a bit shift to the right (to get the sign bit)
25 add eax, ecx ;Adds the sign bit to eax
26 sar eax, 1 ;Performs an arithmetic shift to the right
27 sub edx, eax ;Subtracts the result of X*y by x/2
28 mov eax, edx ;Move edx into eax
29 mov [esp+14h], eax ;Stores the value of eax into esp+14h variable
30 mov eax, [esp+14h] ;Stores the value of that variable into eax
31 mov [esp+4], eax ;Stores the value of that variable into the first index of esp
32 mov dword ptr [esp], offset aD "%d" ;Prints the variable to console
33 call _printf
34 mov eax, 0 ;Clears eax
35 leave ;Close the program
36 ret 0 ;Return value of main is 0
37 _main endp

```

```

1  ;Creates an array and iterates through it to find the largest value
2  .text:00401500 push ebp
3  .text:00401501 mov ebp, esp
4  .text:00401503 and esp, 0FFFFFFF0h
5  .text:00401506 sub esp, 40h
6  .text:00401509 call ___main
7  .text:0040150E mov dword ptr [esp+18h], 0Ch ;Array address = 12
8  .text:00401516 mov dword ptr [esp+1Ch], 0Fh ;B = 15
9  .text:0040151E mov dword ptr [esp+20h], 0DDh ;C = 221
10 .text:00401526 mov dword ptr [esp+24h], 3 ;D = 3
11 .text:0040152E mov dword ptr [esp+28h], 1B0h ;E = 432
12 .text:00401536 mov dword ptr [esp+2Ch], 36h ;F = 54
13 .text:0040153E mov dword ptr [esp+30h], 10h ;G = 16
14 .text:00401546 mov dword ptr [esp+34h], 43h ;H = 67
15 .text:0040154E mov dword ptr [esp+3Ch], 0 ;x = 0
16 .text:00401556 mov dword ptr [esp+38h], 0 ;y = 0
17 .text:0040155E jmp short loc_40157F
18 .text:00401560 ; -----
19 .text:00401560
20 .text:00401560 loc_401560: ; CODE XREF: _main+84↓j
21 .text:00401560 mov eax, [esp+38h] ; eax <-- y
22 .text:00401564 mov eax, [esp+eax*4+18h] ;eax <-- (eax * 4) + Array address
23 .text:00401568 cmp eax, [esp+3Ch] ; Compare eax to x
24 .text:0040156C jle short loc_40157A ;If eax < x
25 .text:0040156E mov eax, [esp+38h] ;eax <- y
26 .text:00401572 mov eax, [esp+eax*4+18h] ;eax <- (eax*4) + Array address
27 .text:00401576 mov [esp+3Ch], eax ; x <-- eax
28 .text:0040157A
29 .text:0040157A loc_40157A: ; CODE XREF: _main+6C↑j
30 .text:0040157A add dword ptr [esp+38h], 1 ; Add 1 to y
31
32 .text:0040157F
33 .text:0040157F loc_40157F: ; CODE XREF: _main+5E↑j
34 .text:0040157F cmp dword ptr [esp+38h], 7 ;y < 7
35 .text:00401584 jle short loc_401560
36 .text:00401586 mov eax, [esp+3Ch] ; eax <-- x
37 .text:0040158A mov [esp+4], eax ;Index 1 <- eax
38 .text:0040158E mov dword ptr [esp], offset aD ; "%d"
39 .text:00401595 call _printf
40 .text:0040159A mov eax, 0
41 .text:0040159F leave
42 .text:004015A0 retn
43 .text:004015A0 _main endp

```

```

1  .text:00401500 push ebp
2  .text:00401501 mov ebp, esp
3  .text:00401503 and esp, 0FFFFFFF0h
4  .text:00401506 sub esp, 20h
5  .text:00401509 call ____main
6  .text:0040150E mov dword ptr [esp+1Ch], 64h ;int X = 100
7  .text:00401516 jmp loc_4015D6
8  .text:0040151B ;
-----
9  .text:0040151B
10 .text:0040151B loc_40151B: ; CODE XREF: _main+DE↓j
11
12 ;Used to do a division by 100
13 ;-----
14 .text:0040151B mov ecx, [esp+1Ch] ;Move x into ecx
15 .text:0040151F mov edx, 51EB851Fh ;Moves 1374389535 into edx (USED TO DO DIVISIONS BY
16 100)
17 .text:00401524 mov eax, ecx ;moves ecx(x) into eax (the counter)
18 .text:00401526 imul edx ;Signed multiply edx by eax and stores the value in edx (x * edx)
19 .text:00401528 sar edx, 5 ;Shifts edx to the right by 5 (x * edx / 2^5)
20 .text:0040152B mov eax, ecx ;Stores ecx into eax
21 .text:0040152D sar eax, 1Fh ;retrieves the sign bit of x
22 .text:00401530 sub edx, eax ; (edx - x)
23 .text:00401532 mov eax, edx ; (eax <- (edx - x))
24 .text:00401534 mov [esp+18h], eax ;int y = eax /100
25 ;-----
26 ;edx = eax * -100, ecx = y * -100 + x
27 ;-----
28 .text:00401538 mov eax, [esp+18h] ;eax = -100
29 .text:0040153C imul edx, eax, -64h ;edx = -100 * -100 = 10000
30 .text:0040153F mov eax, [esp+1Ch] ; eax = x
31 .text:00401543 lea ecx, [edx+eax] ;ecx = 10100
32 ;-----
33
34 ;z = (y * -100 + x) / 10;
35 ;-----
36 .text:00401546 mov edx, 66666667h ;edx = 1717986919
37 .text:0040154B mov eax, ecx ;eax <- ecx (10100)
38 .text:0040154D imul edx ;eax (1717986919) * 10100
39 .text:0040154F sar edx, 2 ;edx = 429496729
40 .text:00401552 mov eax, ecx ; eax <- 10100
41 .text:00401554 sar eax, 1Fh
42 .text:00401557 sub edx, eax
43 .text:00401559 mov eax, edx
44 .text:0040155B mov [esp+14h], eax ;int z = ecx/10
45 ;-----
46
47 ;-----
48 .text:0040155F mov ecx, [esp+1Ch] ;ecx = x
49 .text:00401563 mov edx, 66666667h
50 .text:00401568 mov eax, ecx
51 .text:0040156A imul edx
52 .text:0040156C sar edx, 2
53 .text:0040156F mov eax, ecx
54 .text:00401571 sar eax, 1Fh
55 .text:00401574 sub edx, eax ;edx = x/10
56 .text:00401576 mov eax, edx ;eax = x/10
57 .text:00401578 shl eax, 2 ; eax = x/10 * 4
58 .text:0040157B add eax, edx ; eax = (x / 10 * 4) + x/10 => x / 10 * 5
59 .text:0040157D add eax, eax ; eax = 2*(x/10 * 5)
60 .text:0040157F sub ecx, eax ;ecx = x - (2*(x/10 * 5))
61 .text:00401581 mov eax, ecx ; eax = x - (2*(x/10*5))
62 .text:00401583 mov [esp+10h], eax ; int a = eax
63
64 .text:00401587 mov eax, [esp+18h] ;eax = y
65 .text:0040158B imul eax, [esp+18h] ;eax = y * y
66 .text:00401590 imul eax, [esp+18h] ;eax = y* y* y
67 .text:00401595 mov edx, eax ;edx = y * y * y

```

```

68 .text:00401597 mov eax, [esp+14h] ;eax = z
69 .text:0040159B imul eax, [esp+14h] ;eax = z * z
70 .text:004015A0 imul eax, [esp+14h]; eax = z * z * z
71 .text:004015A5 add edx, eax ;edx = (y*y*y)+ (z*z*z)
72 .text:004015A7 mov eax, [esp+10h] ;eax = a
73 .text:004015AB imul eax, [esp+10h] ;eax = a * a
74 .text:004015B0 imul eax, [esp+10h] ;eax = a * a * a
75 .text:004015B5 add eax, edx ;eax = (a * a * a) + (y*y*y) + (z*z*z)
76 .text:004015B7 cmp eax, [esp+1Ch] ; eax == x
77 .text:004015BB jnz short loc_4015D1
78 .text:004015BD mov eax, [esp+1Ch]
79 .text:004015C1 mov [esp+4], eax
80 .text:004015C5 mov dword ptr [esp], offset aD ; "%d "
81 .text:004015CC call _printf
82 ;-----
83
84 ;Adds 1 to x
85 .text:004015D1
86 .text:004015D1 loc_4015D1: ; CODE XREF: _main+BB j
87 .text:004015D1 add dword ptr [esp+1Ch], 1
88
89 ;Checks if x <= 999
90 .text:004015D6
91 .text:004015D6 loc_4015D6: ; CODE XREF: _main+16↑j
92 .text:004015D6 cmp dword ptr [esp+1Ch], 3E7h ;Checks if x < 999
93 .text:004015DE jle loc_40151B
94
95
96 .text:004015E4 mov eax, 0
97 .text:004015E9 leave
98 .text:004015EA retn
99 .text:004015EA _main endp

```

```

1  push    ebp
2  mov     ebp, esp
3  and     esp, 0FFFFFFF0h
4  sub     esp, 1B0h
5  call    __main
6  mov     dword ptr [esp+1A8h], 7 ;int x = 7
7  mov     dword ptr [esp+1A4h], 64h ;int y = 100
8  mov     dword ptr [esp+1ACh], 0 ;int z = 0
9  jmp     short loc_401619
10
11 loc_401619:
12 mov     eax, [esp+1ACh] ; eax = z
13 cmp     eax, [esp+1A4h] ; If eax < y
14 jnl     short loc_4015FC
15
16 mov     eax, [esp+1A8h] ;eax = x
17 mov     [esp+8], eax ;int a = eax
18 mov     eax, [esp+1A4h] ;eax = y
19 mov     [esp+4], eax ;int b = eax
20 lea     eax, [esp+1B0h+var_19C]
21 mov     [esp], eax ; int *
22 call    __Z5proclPiii ; procl(int *,int a,int b)
23 mov     [esp+4], eax
24 mov     dword ptr [esp], offset aD ; "%d\n"
25 call    _printf
26 mov     eax, 0
27 leave
28 retn
29 __main endp
30
31 loc_4015FC:
32 mov     eax, [esp+1ACh] ;eax = z
33 lea     edx, [eax+1] ;edx = z + 1
34 mov     eax, [esp+1ACh] ; eax = z
35 mov     [esp+eax*4+14h], edx ;array[z] = edx
36 add     dword ptr [esp+1ACh], 1 ; z = z + 1
37 ;-----
38
39 PROC1
40 var_10= dword ptr -10h ;c
41 var_C= dword ptr -0Ch
42 var_8= dword ptr -8 ;e
43 var_4= dword ptr -4 ;f
44 arg_0= dword ptr 8 ;Array
45 arg_4= dword ptr 0Ch ;y =100
46 arg_8= dword ptr 10h ;x = 7
47
48 push    ebp
49 mov     ebp, esp
50 sub     esp, 10h
51 mov     [ebp+var_C], 0 ;b = 0
52 mov     [ebp+var_10], 0;c = 0
53 mov     [ebp+var_4], 0 ;d = 0
54 jmp     loc_4015B7
55
56 loc_401520:
57 mov     [ebp+var_8], 1 ;ebp+var_8
58 jmp     short loc_40155E
59
60 loc_401529:
61 jmp     short loc_401538
62
63 loc_40152B:
64 mov     eax, [ebp+var_C] ;eax = b
65 add     eax, 1 ;eax = b + 1
66 cdq
67 idiv     [ebp+arg_4] ; edx = eax / y
68 mov     [ebp+var_C], edx ; b = edx
69

```

```

70  loc_401538:
71  mov     eax, [ebp+var_C] ;eax = b
72  lea     edx, ds:0[eax*4]
73  mov     eax, [ebp+arg_0] ;eax = *array
74  add     eax, edx ;eax = eax + edx => *array + edx
75  mov     eax, [eax] ;eax = array[b]
76  test    eax, eax
77  jz      short loc_40152B
78
79  add     [ebp+var_8], 1 ;e = e + 1
80  mov     eax, [ebp+var_C] ;eax = b
81  add     eax, 1 ;eax = eax + 1
82  cdq
83  idiv    [ebp+arg_4] ; edx = eax / y
84  mov     [ebp+var_C], edx ;b = edx
85
86  loc_40155E:
87  mov     eax, [ebp+var_8] ;eax = var_8
88  cmp     eax, [ebp+arg_8] ;If eax < arg_8
89  jl      short loc_401529
90
91  jmp     short loc_401575
92
93  loc_401568:
94  mov     eax, [ebp+var_C] ;eax = b
95  add     eax, 1 ;eax = b + 1
96  cdq
97  idiv    [ebp+arg_4] ;eax = (b + 1) / y
98  mov     [ebp+var_C], edx ;b = (b + 1) / y
99
100 loc_401575:
101 mov     eax, [ebp+var_C] ;eax = b
102 lea     edx, ds:0[eax*4]
103 mov     eax, [ebp+arg_0] ;eax = *array
104 add     eax, edx ;eax = eax + edx => *array + edx
105 mov     eax, [eax] ;eax = array[b]
106 test    eax, eax
107 jz      short loc_401568
108
109 mov     eax, [ebp+var_C] ;eax = b
110 lea     edx, ds:0[eax*4]
111 mov     eax, [ebp+arg_0] ;eax = *array
112 add     eax, edx ;eax = eax + edx => *array + edx
113 mov     eax, [eax] ;eax = array[b]
114 mov     [ebp+var_10], eax ;c = eax
115 mov     eax, [ebp+var_C] ;eax = b
116 lea     edx, ds:0[eax*4]
117 mov     eax, [ebp+arg_0] ;eax = *array
118 add     eax, edx ;eax + eax + edx => *array + edx
119 mov     dword ptr [eax], 0 ;eax = 0
120 add     [ebp+var_4], 1 ;d = d++
121
122 loc_4015B7:
123 mov     eax, [ebp+var_4] ;eax = var_4
124 cmp     eax, [ebp+arg_4] ;if eax < arg_4
125 jl      loc_401520
126
127 mov     eax, [ebp+var_10] ;eax = c
128 leave
129 retn
130 __Z5proc1Piii endp

```