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:

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

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 (y^3 + z^3 + a^3). 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 appears to make an array and passes it into a function call and returns a value.

```
public _main
_main proc near

argc= dword ptr 8
argv= dword ptr 0ch
envp= dword ptr 10h
;This program makes three variables, multiplies the first by the second variable
; X * Y
; X/2
; z = (x*y)-(x/2)
push ebp
mov ebp, esp
and esp, 0FFFFFFF0h ;Just aligning stack
sub esp, 20h ;Reserve space for local variables
call _main ;Calls the main function in c
mov dword ptr [esp+1ch], 3 ;X: Creates a variable and assigns a value of 3 to it
mov dword ptr [esp+18h], 5 ;Y: Creates a variable and assigns a value of 5 to it
mov dword ptr [esp+14h], 0 ;Z: Creates a variable and assigns a value of 0 to it
mov eax, [esp+1ch] ;Moves the first value into the accumulator
imul eax, [esp+18h] ;Multiplies the second by the first
mov edx, eax ;Moves the value into the edx register
mov eax, [esp+1ch] ;Moves the second value into the eax register
mov ecx, eax ;Stores that value into ecx (temp register)
shr ecx, 1fh ;Performs a bit shift to the right (to get the sign bit)
add eax, ecx ;Adds the sign bit to eax
sar eax, 1 ;Performs an arithmetic shift to the right
sub edx, eax ;Subtracts the result of X*y by x/2
mov eax, edx ;Move edx into eax
mov [esp+14h], eax ;Stores the value of eax into esp+14h variable
mov eax, [esp+14h] ;Stores the value of that variable into eax
mov [esp+4], eax ;Stores the value of that  variable into the first index of esp
mov dword ptr [esp], offset aD "%d" ;Prints the variable to console
call _printf
mov eax, 0 ;Clears eax
leave ;Close the program
retn ;Return value of main is 0
_main endp
```

```
;Creates an array and iterates through it to find the largest value
.text:00401500 push ebp
.text:00401501 mov ebp, esp
.text:00401503 and esp, 0FFFFFFF0h
.text:00401506 sub esp, 40h
.text:00401509 call ___main
.text:0040150E mov dword ptr [esp+18h], 0Ch  ;Array address = 12
.text:00401516 mov dword ptr [esp+1Ch], 0Fh  ;B = 15
.text:0040151E mov dword ptr [esp+20h], 0DDh ;C = 221
.text:00401526 mov dword ptr [esp+24h], 3    ;D = 3
.text:0040152E mov dword ptr [esp+28h], 1B0h ;E = 432
.text:00401536 mov dword ptr [esp+2Ch], 36h  ;F = 54
.text:0040153E mov dword ptr [esp+30h], 10h  ;G = 16
.text:00401546 mov dword ptr [esp+34h], 43h  ;H = 67
.text:0040154E mov dword ptr [esp+3Ch], 0    ;x = 0
.text:00401556 mov dword ptr [esp+38h], 0    ;y = 0
.text:0040155E jmp short loc_40157F
.text:00401560 ; ---------------------------------------------------------
.text:00401560
.text:00401560 loc_401560: ; CODE XREF: _main+84↓j
.text:00401560 mov eax, [esp+38h] ; eax <-- y
.text:00401564 mov eax, [esp+eax*4+18h] ;eax <-- (eax * 4) + Array address
.text:00401568 cmp eax, [esp+3Ch]  ; Compare eax to  x
.text:0040156C jle short loc_40157A ;If eax < x
.text:0040156E mov eax, [esp+38h] ;eax <- y
.text:00401572 mov eax, [esp+eax*4+18h] ;eax <- (eax*4) + Array address
.text:00401576 mov [esp+3Ch], eax ; x <-- eax
.text:0040157A
.text:0040157A loc_40157A: ; CODE XREF: _main+6C↑j
.text:0040157A add dword ptr [esp+38h], 1 ; Add 1 to y

.text:0040157F
.text:0040157F loc_40157F: ; CODE XREF: _main+5E↑j
.text:0040157F cmp dword ptr [esp+38h], 7 ;y < 7
.text:00401584 jle short loc_401560
.text:00401586 mov eax, [esp+3Ch] ; eax <-- x
.text:0040158A mov [esp+4], eax ;Index 1 <- eax
.text:0040158E mov dword ptr [esp], offset aD ; "%d"
.text:00401595 call _printf
.text:0040159A mov eax, 0
.text:0040159F leave
.text:004015A0 retn
.text:004015A0 _main endp
```

```
.text:00401500 push ebp
.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 ;int X = 100
.text:00401516 jmp loc_4015D6
.text:0040151B ;
-----------------------------------------------------------------------------
.text:0040151B
.text:0040151B loc_40151B: ; CODE XREF: _main+DE↓j

;Used to do a division by 100
;-------------------------------------------------
.text:0040151B mov ecx, [esp+1Ch] ;Move x into ecx
.text:0040151F mov edx, 51EB851Fh ;Moves 1374389535 into edx (USED TO DO DIVISIONS BY
100)
.text:00401524 mov eax, ecx ;moves ecx(x) into eax (the counter)
.text:00401526 imul edx ;Signed multiply edx by eax and stores the value in edx (x * edx)
.text:00401528 sar edx, 5 ;Shifts edx to the right by 5 (x * edx / 2^5)
.text:0040152B mov eax, ecx ;Stores ecx into eax
.text:0040152D sar eax, 1Fh ;retrieves the sign bit of x
.text:00401530 sub edx, eax ; (edx - x)
.text:00401532 mov eax, edx ; (eax <- (edx - x))
.text:00401534 mov [esp+18h], eax ;int y = eax /100
;-------------------------------------------------

;edx = eax * -100, ecx = y * -100 + x
;-----------------------------------------
.text:00401538 mov eax, [esp+18h] ;eax  = -100
.text:0040153C imul edx, eax, -64h ;edx = -100 * -100 = 10000
.text:0040153F mov eax, [esp+1Ch] ; eax = x
.text:00401543 lea ecx, [edx+eax] ;ecx = 10100
;-------------------------------------------------------

;z =  (y * -100 + x) / 10;
;-----------------------------------------------
.text:00401546 mov edx, 66666667h ;edx = 1717986919
.text:0040154B mov eax, ecx ;eax <- ecx (10100)
.text:0040154D imul edx ;eax (1717986919) * 10100
.text:0040154F sar edx, 2 ;edx = 429496729
.text:00401552 mov eax, ecx ; eax <- 10100
.text:00401554 sar eax, 1Fh
.text:00401557 sub edx, eax
.text:00401559 mov eax, edx
.text:0040155B mov [esp+14h], eax ;int z = ecx/10
;-----------------------------------------------

;-----------------------------------------------
.text:0040155F mov ecx, [esp+1Ch] ;ecx = x
.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 ;edx = x/10
.text:00401576 mov eax, edx ;eax = x/10
.text:00401578 shl eax, 2 ; eax = x/10 * 4
.text:0040157B add eax, edx ; eax = (x / 10 * 4) + x/10 => x / 10 * 5
.text:0040157D add eax, eax ; eax = 2*(x/10 * 5)
.text:0040157F sub ecx, eax ;ecx = x - (2*(x/10 * 5))
.text:00401581 mov eax, ecx; eax = x - (2*(x/10*5))
.text:00401583 mov [esp+10h], eax ; int a = eax

.text:00401587 mov eax, [esp+18h] ;eax = y
.text:0040158B imul eax, [esp+18h] ;eax = y * y
.text:00401590 imul eax, [esp+18h] ;eax = y* y* y
.text:00401595 mov edx, eax ;edx = y * y * y
```

```
.text:00401597 mov eax, [esp+14h] ;eax = z
.text:0040159B imul eax, [esp+14h] ;eax = z * z
.text:004015A0 imul eax, [esp+14h]; eax = z * z * z
.text:004015A5 add edx, eax ;edx = (y*y*y)+ (z*z*z)
.text:004015A7 mov eax, [esp+10h] ;eax = a
.text:004015AB imul eax, [esp+10h] ;eax = a * a
.text:004015B0 imul eax, [esp+10h] ;eax = a * a * a
.text:004015B5 add eax, edx        ;eax = (a * a * a) + (y*y*y) + (z*z*z)
.text:004015B7 cmp eax, [esp+1Ch] ; eax == x
.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
;-----------------------------------------

;Adds 1 to x
.text:004015D1
.text:004015D1 loc_4015D1: ; CODE XREF: _main+BB j
.text:004015D1 add dword ptr [esp+1Ch], 1

;Checks if x <= 999
.text:004015D6
.text:004015D6 loc_4015D6: ; CODE XREF: _main+16↑j
.text:004015D6 cmp dword ptr [esp+1Ch], 3E7h ;Checks if x < 999
.text:004015DE jle loc_40151B


.text:004015E4 mov eax, 0
.text:004015E9 leave
.text:004015EA retn
.text:004015EA _main endp
```

```
push    ebp
mov     ebp, esp
and     esp, 0FFFFFFF0h
sub     esp, 1B0h
call    ___main
mov     dword ptr [esp+1A8h], 7 ;int x = 7
mov     dword ptr [esp+1A4h], 64h ;int y = 100
mov     dword ptr [esp+1ACh], 0 ;int z = 0
jmp     short loc_401619

loc_401619:
mov     eax, [esp+1ACh] ; eax = z
cmp     eax, [esp+1A4h] ; If eax < y
jl      short loc_4015FC

mov     eax, [esp+1A8h] ;eax = x
mov     [esp+8], eax    ;int a = eax
mov     eax, [esp+1A4h] ;eax =  y
mov     [esp+4], eax    ;int b = eax
lea     eax, [esp+1B0h+var_19C]
mov     [esp], eax      ; int *
call    __Z5proc1Piii   ; proc1(int *,int a,int b)
mov     [esp+4], eax
mov     dword ptr [esp], offset aD ; "%d\n"
call    _printf
mov     eax, 0
leave
retn
_main endp

loc_4015FC:
mov     eax, [esp+1ACh] ;eax = z
lea     edx, [eax+1] ;edx = z + 1
mov     eax, [esp+1ACh] ; eax = z
mov     [esp+eax*4+14h], edx ;array[z] = edx
add     dword ptr [esp+1ACh], 1 ; z = z + 1
;-----------------------------------------

PROC1
var_10= dword ptr -10h ;c
var_C= dword ptr -0Ch
var_8= dword ptr -8 ;e
var_4= dword ptr -4 ;f
arg_0= dword ptr  8 ;Array
arg_4= dword ptr  0Ch ;y =100
arg_8= dword ptr  10h ;x = 7

push    ebp
mov     ebp, esp
sub     esp, 10h
mov     [ebp+var_C], 0 ;b = 0
mov     [ebp+var_10], 0;c = 0
mov     [ebp+var_4], 0 ;d = 0
jmp     loc_4015B7

loc_401520:
mov     [ebp+var_8], 1 ;ebp+var_8
jmp     short loc_40155E

loc_401529:
jmp     short loc_401538

loc_40152B:
mov     eax, [ebp+var_C] ;eax = b
add     eax, 1 ;eax = b + 1
cdq
idiv    [ebp+arg_4]      ; edx = eax / y
mov     [ebp+var_C], edx ; b = edx
```

```
loc_401538:
mov     eax, [ebp+var_C] ;eax = b
lea     edx, ds:0[eax*4]
mov     eax, [ebp+arg_0] ;eax = *array
add     eax, edx ;eax = eax + edx => *array + edx
mov     eax, [eax] ;eax = array[b]
test    eax, eax
jz      short loc_40152B

add     [ebp+var_8], 1 ;e = e + 1
mov     eax, [ebp+var_C] ;eax = b
add     eax, 1 ;eax = eax + 1
cdq
idiv    [ebp+arg_4] ; edx = eax / y
mov     [ebp+var_C], edx ;b = edx

loc_40155E:
mov     eax, [ebp+var_8] ;eax = var_8
cmp     eax, [ebp+arg_8] ;If eax < arg_8
jl      short loc_401529

jmp     short loc_401575

loc_401568:
mov     eax, [ebp+var_C] ;eax = b
add     eax, 1 ;eax = b + 1
cdq
idiv    [ebp+arg_4] ;eax = (b + 1) / y
mov     [ebp+var_C], edx ;b = (b + 1) / y

loc_401575:
mov     eax, [ebp+var_C] ;eax = b
lea     edx, ds:0[eax*4]
mov     eax, [ebp+arg_0] ;eax = *array
add     eax, edx ;eax = eax + edx => *array + edx
mov     eax, [eax] ;eax = array[b]
test    eax, eax
jz      short loc_401568

mov     eax, [ebp+var_C] ;eax = b
lea     edx, ds:0[eax*4]
mov     eax, [ebp+arg_0] ;eax = *array
add     eax, edx ;eax = eax + edx => *array + edx
mov     eax, [eax] ;eax = array[b]
mov     [ebp+var_10], eax ;c = eax
mov     eax, [ebp+var_C] ;eax = b
lea     edx, ds:0[eax*4]
mov     eax, [ebp+arg_0] ;eax = *array
add     eax, edx ;eax + eax + edx => *array + edx
mov     dword ptr [eax], 0 ;eax = 0
add     [ebp+var_4], 1 ;d = d++

loc_4015B7:
mov     eax, [ebp+var_4] ;eax = var_4
cmp     eax, [ebp+arg_4] ;if eax < arg_4
jl      loc_401520

mov     eax, [ebp+var_10] ;eax = c
leave
retn
__Z5proc1Piii endp
```