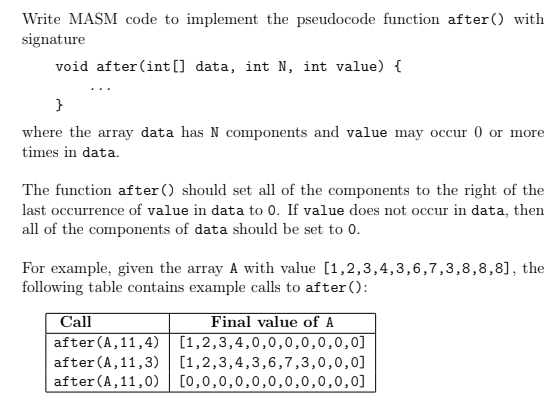
Lab Exam AA 8/12/2017



**main.asm**

EXTERNDEF after:near

.data

N = 11

M = N-1

givenArray DWORD 1,2,3,4,3,6,7,3,8,8,8

givenValue DWORD 3

message BYTE 'Press OK to continue',0

caption BYTE 'Welcome to CA296',0

.code

main:nop

push M

push N

push offset givenArray

push givenValue

call after

invoke writeArray, addr givenArray, N

finish:

invoke MessageBox,0,ADDR message,ADDR caption,MB\_OK

invoke ExitProcess,0

end main

**after.asm**

.586

.model flat,stdcall

.stack 4096

EXTERNDEF after:near

.code

after:

push ebp

mov ebp, esp

sub esp, 4 ; creating space for our i -will be in ebp-4

mov ecx, [ebp+20] ; move len-1 to ecx

mov [ebp-4], ecx ; move len-1 to index space

go\_through:

mov ecx, [ebp-4] ; move index to ecx

cmp ecx, 0 ; when index less than 0 we are done

jl go\_home

shl ecx, 2 ; multiply index by 4

add ecx, [ebp+12] ; add index to address

mov ebx, ecx ; put address of last value into ebx

mov eax, [ebx] ; value in ith position of array into eax

cmp eax, [ebp+8] ; compare with givenValue

jne switcheroo ; if not it we need to put 0 in that space

jmp go\_home

switcheroo:

mov eax, 0 ; put 0 in eax

mov [ebx], eax ; put 0 into array slot in position i

dec DWORD ptr[ebp-4] ; i -= 1

jmp go\_through

go\_home:

mov eax, [ebp+12] ; mov array into eax

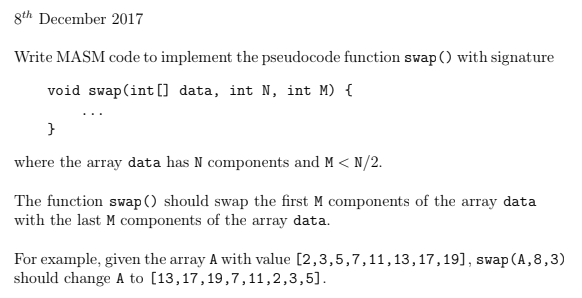
mov esp, ebp ; do reserse of start

pop ebp ; pop the top of stack (return address) to ebp

ret 16 ; return by number of items pushed in main\*4

end

Lab Exam BB 8/12/2017



**main.asm**

; Data & Code

;

EXTERNDEF swap:near

.data

N = 8

M = 3

givenArray DWORD 2,3,5,7,11,13,17,19

message BYTE 'Press OK to continue',0

caption BYTE 'Welcome to CA296',0

.code

main:nop

invoke writeArray, addr givenArray, N

push M

push N

push offset givenArray

call swap

invoke writeArray, addr givenArray, N

finish:

invoke MessageBox,0,ADDR message,ADDR caption,MB\_OK

invoke ExitProcess,0

end main

**swap.asm**

EXTERNDEF swap:near

.code

swap: push ebp ; push return address

mov ebp, esp

sub esp, 4 ; creating space for our i

mov eax, 0

mov [ebp-4], eax ; this is our i index - start at 0

mov ecx, [ebp+12] ; put length into ecx

sub ecx, [ebp+16] ; j-M = starting point from right

mov [ebp-8], ecx ; put j index (len-M) into ebp-8

from\_left:

mov ecx, [ebp-4] ; move i into ecx

cmp ecx, [ebp+16] ; cmp i with M

je go\_home

shl ecx, 2 ; i\*4

add ecx, [ebp+8] ; add i\*4 to address of first num in array

mov ebx, ecx ; put address of value into ebx

from\_right:

mov ecx, [ebp-8] ; mov j into ecx

shl ecx, 2 ; j\*4

add ecx, [ebp+8] ; add j\*4 to address in array-Mth position

switch\_em:

mov eax, [ebx] ; put ith value at [ebx] into temp(eax)

mov edx, [ecx] ; put jth value into temp (edx)

mov [ecx], eax ; put ith value into j position

mov [ebx], edx ; put jth value into i position

inc DWORD ptr[ebp-4] ; i+=1

inc DWORD ptr[ebp-8] ; j+=1

jmp from\_left

go\_home:

mov eax, [ebp+8] ; for main to access it, we must put array into eax

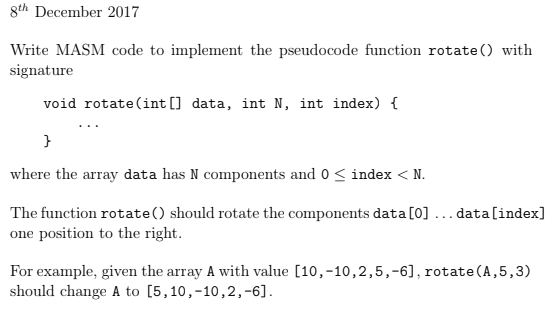
mov esp, ebp ; do reverse of start

pop ebp ; pop top of stack (return address) to ebp

ret 12 ; return number of items pushed in main\*4

end

Lab Exam CC 8/12/2017

****

**main.asm**

; Data & Code

;

EXTERNDEF rotate:near

.data

N = 5

index = 3

givenArray DWORD 10,-10,2,5,-6

message BYTE 'Press OK to continue',0

caption BYTE 'Welcome to CA296',0

.code

main:nop

invoke writeArray, addr givenArray, N

push index

push N

push offset givenArray

call rotate

invoke writeArray, addr givenArray, N

finish:

invoke MessageBox,0,ADDR message,ADDR caption,MB\_OK

invoke ExitProcess,0

end main

**rotate.asm**

EXTERNDEF rotate:near

.code

rotate:

push ebp ; save return address

mov ebp, esp ; pointers align

sub esp, 4

mov eax, 0 ; put 0 into eax

mov [ebp-4], eax ; our i = 0 (stored in ebp-4)

mov eax, [ebp+16] ; put given index into eax

mov [ebp-8], eax ; put j (givenIndex) in ebp-8

from\_j:

mov ecx, [ebp-8] ; put j into ecx

shl ecx, 2 ; j\*4

add ecx, [ebp+8] ; add j\*4 to offset array to get value at jth position

mov ebx, ecx ; move this value's address to ebx

from\_i:

mov ecx, [ebp-4] ; move index into ecx

shl ecx, 2 ; multiply it by 4

add ecx, [ebp+8] ; add i\*4 to offset array to get value at ith position

first\_move:

mov eax, [ebx] ; put jth value at [ebx] into temp1 (eax)

mov edx, [ecx] ; put ith value at [ecx] into temp2 (edx)

mov [ecx], eax ; put jth value into array[0]

switcheroo:

inc DWORD ptr[ebp-4] ; i+=1

mov ebx, [ebp-4] ; put i into ebx

cmp ebx, [ebp+16] ; when i > given index, go\_home

jg go\_home

shl ebx, 2 ; i\*4

add ebx, [ebp+8] ; ebx = A[i]

mov eax, [ebx] ; eax = value in A[i]

mov [ebx], edx ; put prev i into A[i]

mov edx, eax ; put our new prev into edx

jmp switcheroo

go\_home:

mov eax, [ebp+8] ; for main to access it, we must put array into eax

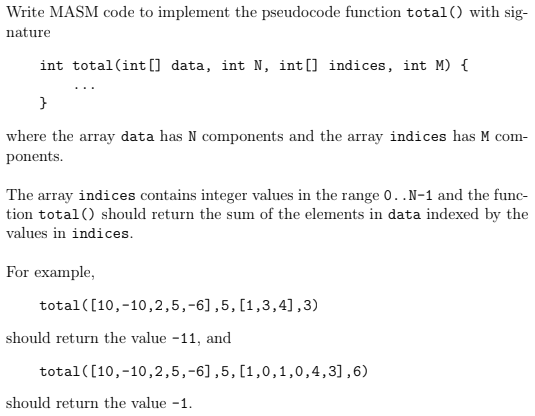
mov esp, ebp ; do reverse of start

pop ebp ; pop top of stack (return address) into ebp

ret 12 ; return number of items pushed in main \*4

end

Lab Exam AA 9/12/2016



**main.asm**

EXTERNDEF total:near

.data

lenGiven = 5

lenIndex = 3

givenArray DWORD 10,-10,2,5,-6

indexes DWORD 1,3,4

message BYTE 'Press OK to continue',0

caption BYTE 'Welcome to CA296',0

.code

main:nop

push lenGiven

push lenIndex

push offset givenArray

push offset indexes

call total

finish:

invoke writeInteger, eax

invoke MessageBox,0,ADDR message,ADDR caption,MB\_OK

invoke ExitProcess,0

end main

**total.asm**

.586

.model flat,stdcall

.stack 4096

EXTERNDEF total:near

.data

sum DWORD 0 ; using this for function only

.code

total:

push ebp

mov ebp, esp

sub esp, 4 ; make room for i

mov edx, 0

mov [ebp-4], edx ; this will be our i

go\_through:

mov eax, [ebp-4] ; put i into eax

cmp eax, [ebp+16] ; while i<len(indexes)

je go\_home ; if equal - done

shl eax, 2 ; i\*4

add eax, [ebp+8] ; add i\*4 to offset indexes

mov ebx, [eax] ; put ith value into ebx

; this is now j (our index for givenArray)

shl ebx, 2 ; j\*4

add ebx, [ebp+12] ; add j\*4 to offset givenArray

mov ecx, [ebx] ; put value in ebx into temp (ecx)

add sum, ecx

inc DWORD ptr[ebp-4] ; i+=1

jmp go\_through

go\_home:

mov eax, sum ; put sum into eax

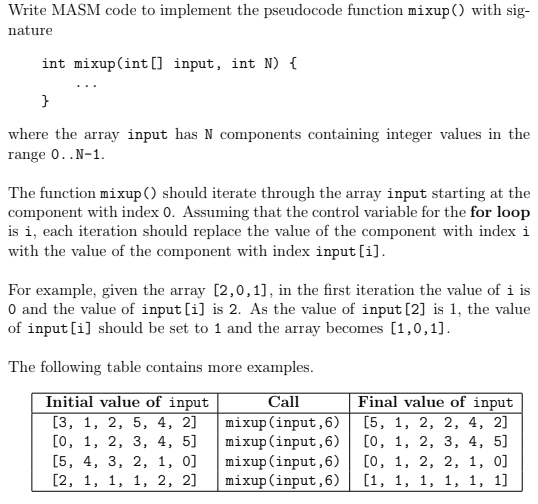
mov esp, ebp

pop ebp

ret 16

end

Lab Exam BB 9/12/2016



**main.asm**

EXTERNDEF mixup:near

.data

givenArray DWORD 3,1,2,5,4,2

M DWORD 6

message BYTE 'Press OK to continue',0

caption BYTE 'Welcome to CA296',0

.code

main:nop

push M

push offset givenArray

call mixup

finish:

invoke writeArray, eax , M

invoke MessageBox,0,ADDR message,ADDR caption,MB\_OK

invoke ExitProcess,0

end main

**mixup.asm**

.586

.model flat,stdcall

.stack 4096

EXTERNDEF mixup:near

.data

.code

mixup:

push ebp

mov ebp, esp

sub esp, 4

mov edx, 0

mov [ebp-4], edx ; i = 0

go\_through:

mov eax, [ebp-4] ; eax = i

cmp eax, [ebp+12] ; while eax != len(array)

je go\_home

shl eax, 2 ; i \*4

add eax, [ebp+8] ; i += offset array

mov ebx, [eax] ; put ith value into ebx

swap\_em:

mov ecx, ebx ; store value into ecx, ebx becomes j index

shl ebx, 2 ; j\*4

add ebx, [ebp+8] ; add j\*4 to offset array to get value at jth position

mov edx, [ebx] ; put jth value into temp (edx)

mov [eax], edx ; put jth value into i position

inc DWORD ptr[ebp-4] ; inc i

jmp go\_through

go\_home:

mov eax, [ebp+8] ; put array into eax

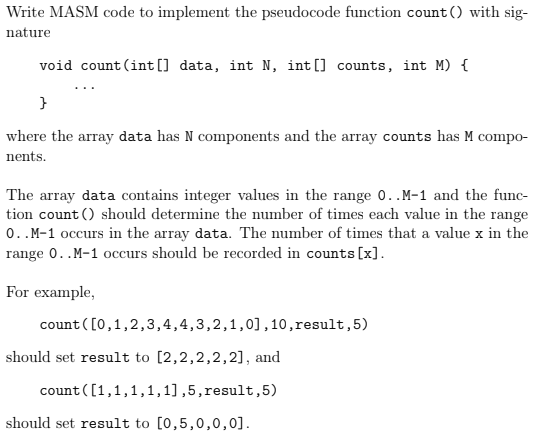
mov esp, ebp ; do reserse of start

pop ebp ; pop the top of stack (return address) to ebp

ret 8 ; return by number of items pushed in main\*4

end

Lab Exam CC 9/12/2016



**main.asm**

EXTERNDEF count:near

.data

len\_giv = 10

len\_res = 5

givenArray DWORD 0,1,2,3,4,4,3,2,1,0

resultArray DWORD len\_res DUP(0)

message BYTE 'Press OK to continue',0

caption BYTE 'Welcome to CA296',0

.code

main:nop

invoke writeArray, ADDR givenArray, len\_giv

push len\_giv

push len\_res

push offset givenArray

push offset resultArray

call count

finish:

invoke writeArray, addr resultArray, len\_res

invoke MessageBox,0,ADDR message,ADDR caption,MB\_OK

invoke ExitProcess,0

end main

**count.asm**

.586

.model flat,stdcall

.stack 4096

EXTERNDEF count:near

.data

.code

count:

push ebp

mov ebp, esp

sub esp, 4 ; make room for i

mov edx, 0 ; edx = 0

mov [ebp-4], edx ; put i into ebp-4

mov [ebp-8], edx ; put j into ebp-8

go\_through:

mov eax, [ebp-4] ; mov i into eax

cmp eax, [ebp+20] ; while i < len(givenArray)

je go\_home

shl eax, 2 ; i\*4

add eax, [ebp+12] ; add i\*4 to offset given

mov ebx, [eax] ; put value in eax into ebx

shl ebx, 2 ; this is our j now (j\*4)

add ebx, [ebp+8] ; add j\*4 to offset resultArr

inc DWORD ptr[ebx] ; increment value in this position

inc DWORD ptr[ebp-4]

jmp go\_through

go\_home:

mov eax, [ebp+8] ; move resultArray to eax

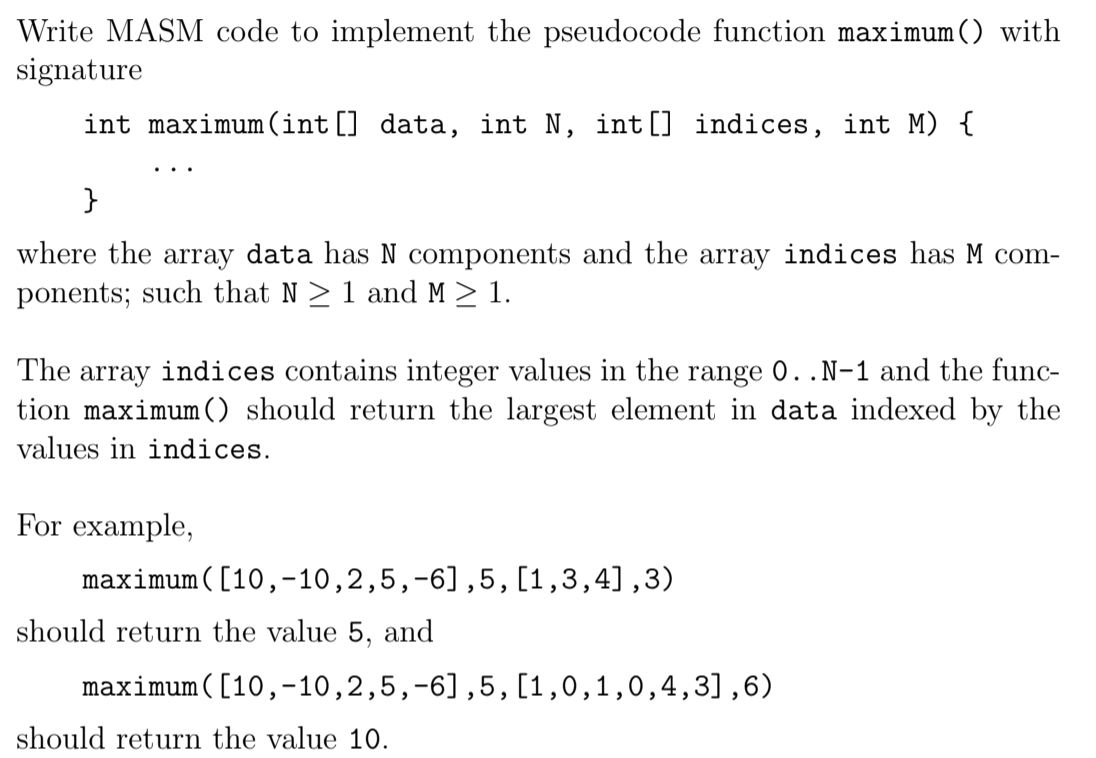
mov esp, ebp

pop ebp

ret 16

end

Resit Lab Exam 9/08/2017



main.asm

EXTERNDEF maximum:near

.data

givenArr DWORD 10,-10,2,5,-6

indexArr DWORD 1,0,1,0,4,3

lenGiven DWORD 5

lenIndex DWORD 6

message BYTE 'Press OK to continue',0

caption BYTE 'Welcome to CA296',0

.code

main:nop

invoke writeArray, addr givenArr, lenGiven

push lenGiven

push lenIndex

push offset givenArr

push offset indexArr

call maximum

finish:

invoke writeInteger, eax

invoke MessageBox,0,ADDR message,ADDR caption,MB\_OK

invoke ExitProcess,0

end main

maximum.asm

EXTERNDEF maximum: near

.data

.code

maximum:

push ebp

mov ebp, esp

sub esp, 8 ; create space for i and maximum

mov eax, 0 ; eax = 0

mov [ebp-4], eax ; i = 0

find\_max:

mov ebx, [ebp-4] ; ebx is our i

cmp ebx, [ebp+16] ; while i < len(indexArr)

je go\_home ; if i == len(indexArray) we are done

shl ebx, 2 ; i\*4

add ebx, [ebp+8] ; add i to indexArr address to get indexArr[i]

mov ecx, [ebx] ; ecx = value at indexArray -- j index

shl ecx, 2 ; j\*4

add ecx, [ebp+12] ; add j to givenArr address to get givenArr[j]

mov edx, [ecx] ; put value at givenArr[j] into edx

cmp [ebp-4], eax ; if i == 0

je switcheroo ; this is our first max, and needs to be saved. else:

cmp edx, [ebp-8] ; see which is largest

jg switcheroo ; if larger, we need to put it in

; else:

inc DWORD ptr[ebp-4] ; i+=1

jmp find\_max ; go back to get next value

switcheroo:

mov [ebp-8], edx ; put max into our maximum space

inc DWORD ptr[ebp-4] ; i+=1

jmp find\_max ; go back to get next value

go\_home:

mov eax, [ebp-8] ; put maximum in eax to send back

mov esp, ebp ; reset pointers to original positions

pop ebp ; pop the top of the stack to ebp

ret 16 ; return by (number of pushes in main)\*4

end