*Отчет по домашнему заданию №2*
*Архитектура вычислительных систем*
*Иванова Виолетта БПИ-195*

**Вариант №11. Вывести новый массив без первого положительного числа в начальном массиве.**

В первом блоке я задаю все необходимые мне константы и переменные.

```
;-----------------------------Some useful data-------------------------------------
section '.data' data readable writable

        strVecSize   db 'enter size of the vector: ', 0
        strIncorSize db 'Incorrect size of vector A = %d', 10, 0
        strVecElemI  db '[%d]? ', 0
        strScanInt   db '%d', 0
        strVecElemOut  db '[%d] = %d', 10, 0

        vec_size     dd 0
        i            dd ?
        tmp          dd ?
        tmpStack     dd ?
        vec_a         rd 100
        vec_b         rd 100
```

Следующий блок отвечает за вызов процедур в порядке их надобности.

```
;-----------------------------------Sections of code-------------------------------------
section '.code' code readable executable
start:
; 1) vector A input
        call VectorInput
; 2) get vector B (without first positive)
        call CreateVectorB

; 3) vector B output
        call VectorOut
finish:
        call [getch]

        push 0
        call [ExitProcess]
```

Первой вызывается процедура из 4 блоков. Программа запрашивает у пользователя размер вектора, при неправильном вводе программа завершает свою работу, при корректном вводе программа запрашивает у пользователя каждый элемент вектора построчно.

```
;--------------------------------Getting vector A--------------------
VectorInput:
        push strVecSize
        call [printf]
        add esp, 4

        push vec_size
        push strScanInt
        call [scanf]
        add esp, 8

        mov eax, [vec_size]
        cmp eax, 0
        jg  getVector
; if bad enter
        push vec_size
        push strIncorSize
        call [printf]
        push 0
        call [ExitProcess]

; else everything OK
getVector:
        xor ecx, ecx
        mov ebx, vec_a
getVecLoop:
        mov [tmp], ebx
        cmp ecx, [vec_size]
        jge endInputVector


        mov [i], ecx
        push ecx
        push strVecElemI
        call [printf]
        add esp, 8

        push ebx
        push strScanInt
        call [scanf]
        add esp, 8

        mov ecx, [i]
        inc ecx
        mov ebx, [tmp]
        add ebx, 4
        jmp getVecLoop
endInputVector:
        ret
```

После этой процедуры программа вызывает процедуру создания и составления нового вектора. Этот этап состоит из пяти блоков. Первый инициализирует счётчик и первое значение вектора. После этого программа проходит по всем значением массива до первого положительного, записывая их в новый вектор. Если найден положительный

элемент, то программа пропускает его и переходит в другой цикл, в котором положительные элементы больше не ищутся. Если создание вектора завершилось в первом цикле (вектор без положительных чисел), то количество элементов в векторе = количеству элементов в начальном векторе. Иначе – количество элементов уменьшается на единицу.

```
;-----------------------------Creating new vector B----------------
CreateVectorB:
        xor ecx, ecx                ; ecx = 0
        mov ebx, vec_a              ; ebx = &vec_a
        mov esi, vec_b              ; esi = &vec_b
        jmp CompareLoop
CompareLoop:
        mov eax, [ebx]
        mov [tmp], ebx
        cmp ecx, [vec_size]
        je endOfCompareLoop

        mov [i], ecx
        cmp eax, 0
        jg  ifPositive

        mov [esi], eax
        add esi, 4
       add  ebx, 4
        inc ecx
        jmp CompareLoop
ifPositive:
       cmp ecx, [vec_size]
       je endOfCompareLoopG

       add  ebx, 4
       mov eax, [ebx]
       mov [esi], eax
       add esi, 4

       inc ecx
       jmp  ifPositive
endOfCompareLoopG:
       dec [vec_size]
       ret
endOfCompareLoop:
        ret
```

Далее программа выводит новый вектор в консоль. Этот этап состоит из трёх блоков.

```
;--------------------------------Printing new Vector B---------------

VectorOut:
        mov [tmpStack], esp
        xor ecx, ecx
        mov ebx, vec_b
        jmp PrintInfo

putVecLoop:
        mov [tmp], ebx
        cmp ecx, [vec_size]
        je endOutputVector
        mov [i], ecx

        ; output element
        push dword [ebx]
        push ecx
        push strVecElemOut
        call [printf]

        mov ecx, [i]
        inc ecx
        mov ebx, [tmp]
        add ebx, 4
        jmp putVecLoop
endOutputVector:
        mov esp, [tmpStack]
        ret
```

```
        ret
;-------------------------------Including HeapApi----------------

section '.idata' import data readable
    library kernel, 'kernel32.dll',\
            msvcrt, 'msvcrt.dll',\
            user32,'USER32.DLL'

include 'api\user32.inc'
include 'api\kernel32.inc'
    import kernel,\
           ExitProcess, 'ExitProcess',\
           HeapCreate,'HeapCreate',\
           HeapAlloc,'HeapAlloc'
  include 'api\kernel32.inc'
    import msvcrt,\
           printf, 'printf',\
           scanf, 'scanf',\
           getch, '_getch'
```