Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ярославский государственный технический университет»

Кафедра «Информационные системы и технологии»

Отчет по лаб/работе защищен

с оценкой \_\_\_\_\_\_\_\_\_

Преподаватель к.т.н Ю.В. Царёв

«18» мая 2022

**ПРОГРАММИРОВАНИЕ ОПЕРАЦИЙ ВВОДА-ВЫВОДА**

Отчет о лабораторной работе по курсу «Машино-зависимые языки программирования»

ЯГТУ 09.03.04-004 ЛР

|  |  |
| --- | --- |
|  | Отчет выполнил  студент группы ЦПИ-11    Аристов Д.В.  «18» мая 2022 |

2022

Цель работы: закрепление лекционного материала по командам прерывания на языке ассемблера и приобретение практических навыков использования этих команд в программах с операциями ввода-вывода.

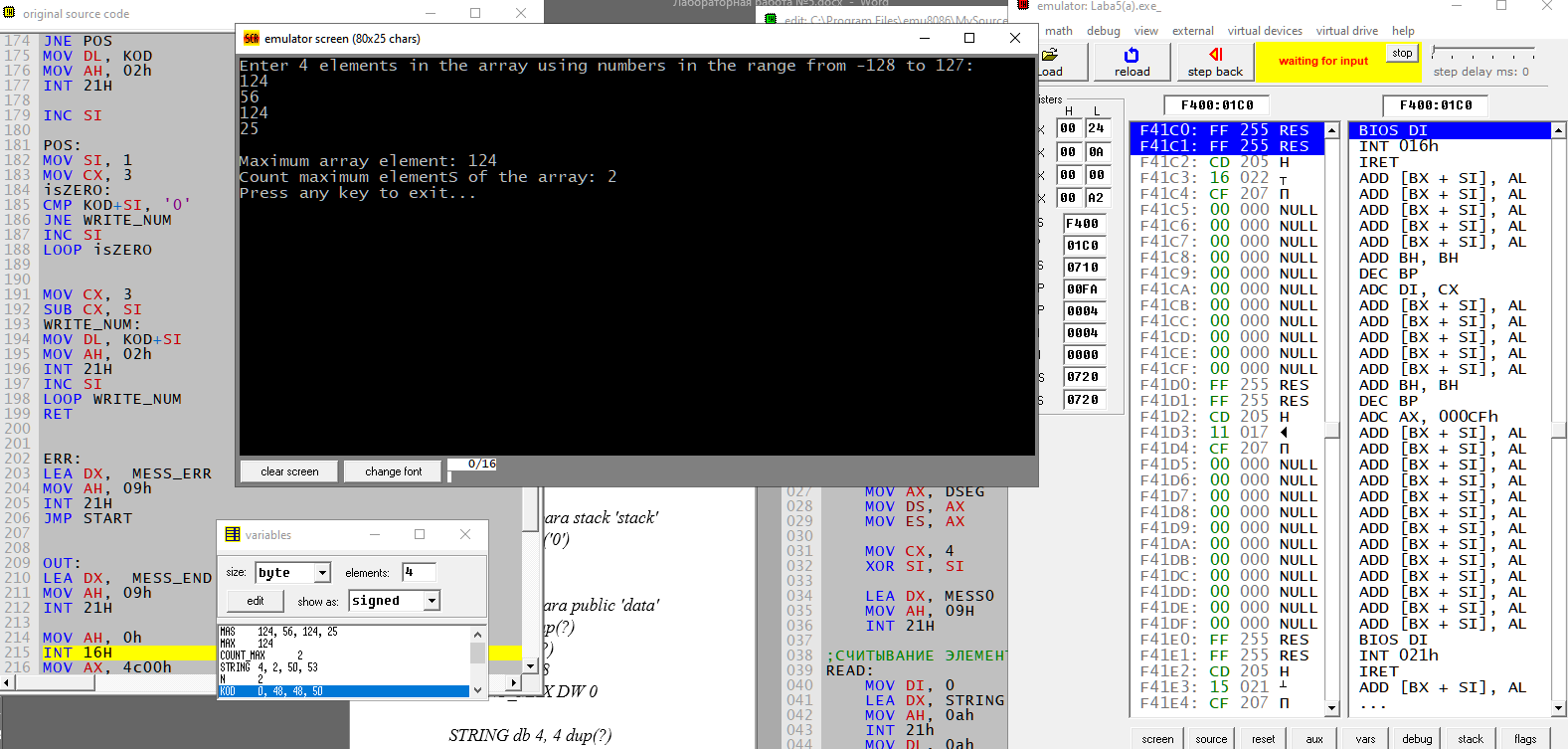
Задание: 4. Найти максимальный (минимальный) элемент двухмерного массива и подсчитать количество таких элементов:

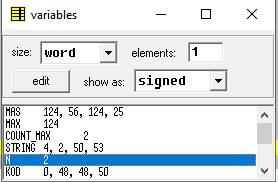
а) элементы массива однобайтовые;

б) элементы массива двухбайтовые.

Схема выполнения: в начале программы происходит считывание элементов и проверка на корректность введенных данных. Если введенный символ не число, то выводится сообщение об ошибке и предложение ввести все сначала. Далее происходит алгоритм поиска максимума(минимума) элементов массива. Затем подсчет количества максимальных(минимальных) элементов. Полученные результаты выводятся на экран посимвольно и учитывая знак.

a)





*Stack segment para stack 'stack'*

*DW 128 dup ('0')*

*Stack ends*

*Dseg segment para public 'data'*

*MAS DB 2 dup(?)*

*DB 2 dup(?)*

*MAX DB -128*

*COUNT\_MAX DW 0*

*STRING DB 5, 4 dup(?)*

*N DB ?*

*KOD DB 4 DUP(?)*

*MESS0 DB "Enter 4 elements in the array using numbers in the range from -128 to 127: ", 0DH,0AH, "$"*

*MESS1 DB 0DH,0AH,"Maximum array element: $"*

*MESS2 DB 0DH,0AH,"Count maximum elements of the array: $"*

*MESS\_END DB 0DH,0AH,"Press any key to exit... $"*

*MESS\_ERR DB "Error! Wrong number. Try again...", 0DH,0AH, "$"*

*Dseg ends*

*Cseg segment para public 'code'*

*Assume cs: cseg, ds: dseg, ss: stack;*

*START:*

*MOV AX, DSEG*

*MOV DS, AX*

*MOV ES, AX*

*MOV CX, 4*

*XOR SI, SI*

*LEA DX, MESS0*

*MOV AH, 09H*

*INT 21H*

*READ:*

*MOV DI, 0*

*LEA DX, STRING*

*MOV AH, 0ah*

*INT 21h*

*MOV DL, 0ah*

*MOV AH, 2*

*INT 21h*

*MOV DL, 0dh*

*MOV AH, 2*

*INT 21h*

*LEA SI, STRING+2*

*CMP [SI], 13*

*JE ERR*

*CMP BYTE PTR [SI], "-"*

*JNE POSITIVE*

*MOV DI, 1*

*INC SI*

*POSITIVE:*

*XOR AX, AX*

*MOV BL, 10*

*isNum:*

*MOV DL, [SI]*

*CMP DL, 13*

*JE END\_NUM*

*CMP DL, '0'*

*JB ERR*

*CMP DL, '9'*

*JA ERR*

*SUB DL, '0'*

*MUL BL*

*ADD AL, DL*

*INC SI*

*JMP isNum*

*END\_NUM:*

*CMP DI, 1*

*JNE END\_VALID\_NUM*

*NEG AL*

*END\_VALID\_NUM:*

*LEA SI, MAS*

*ADD SI, BP*

*MOV [SI], AL*

*INC BP*

*LOOP READ*

*XOR SI, SI*

*MOV CX, 4*

*CYCLE:*

*MOV AL, MAX*

*CMP MAS[SI], AL*

*JG isMAX*

*JMP NEXT\_EL*

*isMAX:*

*MOV AL, MAS[SI]*

*MOV MAX, AL*

*NEXT\_EL:*

*INC SI*

*LOOP CYCLE*

*MOV CX, 4*

*XOR SI, SI*

*FIND\_COUNT\_MAX:*

*MOV AL, MAS[SI]*

*CMP AL, MAX*

*JE INCREMENT*

*JMP NEXT*

*INCREMENT:*

*INC COUNT\_MAX*

*NEXT:*

*INC SI*

*LOOP FIND\_COUNT\_MAX*

*PRINT\_RESULT:*

*LEA DX, MESS1*

*MOV AH, 09h*

*INT 21H*

*MOV AL, MAX*

*MOV N, AL*

*CALL PRINT\_NUM*

*LEA DX, MESS2*

*MOV AH, 09h*

*INT 21H*

*MOV AX, COUNT\_MAX*

*MOV N, AL*

*MOV KOD, 0*

*CALL PRINT\_NUM*

*JMP OUT*

*PRINT\_NUM:*

*CMP N, 0*

*JGE NUM\_SET*

*NEG N*

*MOV KOD, '-'*

*NUM\_SET:*

*MOV BL, 10*

*MOV AL, N*

*MOV AH, 0*

*DIV BL*

*ADD AH, '0'*

*MOV KOD+3, AH*

*MOV AH, 0*

*DIV BL*

*ADD AX, '00'*

*MOV KOD+2, AH*

*MOV KOD+1, AL*

*CMP KOD, '-'*

*JNE POS*

*MOV DL, KOD*

*MOV AH, 02h*

*INT 21H*

*INC SI*

*POS:*

*MOV SI, 1*

*MOV CX, 3*

*isZERO:*

*CMP KOD+SI, '0'*

*JNE WRITE\_NUM*

*INC SI*

*LOOP isZERO*

*MOV CX, 3*

*SUB CX, SI*

*WRITE\_NUM:*

*MOV DL, KOD+SI*

*MOV AH, 02h*

*INT 21H*

*INC SI*

*LOOP WRITE\_NUM*

*RET*

*ERR:*

*LEA DX, MESS\_ERR*

*MOV AH, 09h*

*INT 21H*

*JMP START*

*OUT:*

*LEA DX, MESS\_END*

*MOV AH, 09h*

*INT 21H*

*MOV AH, 0h*

*INT 16H*

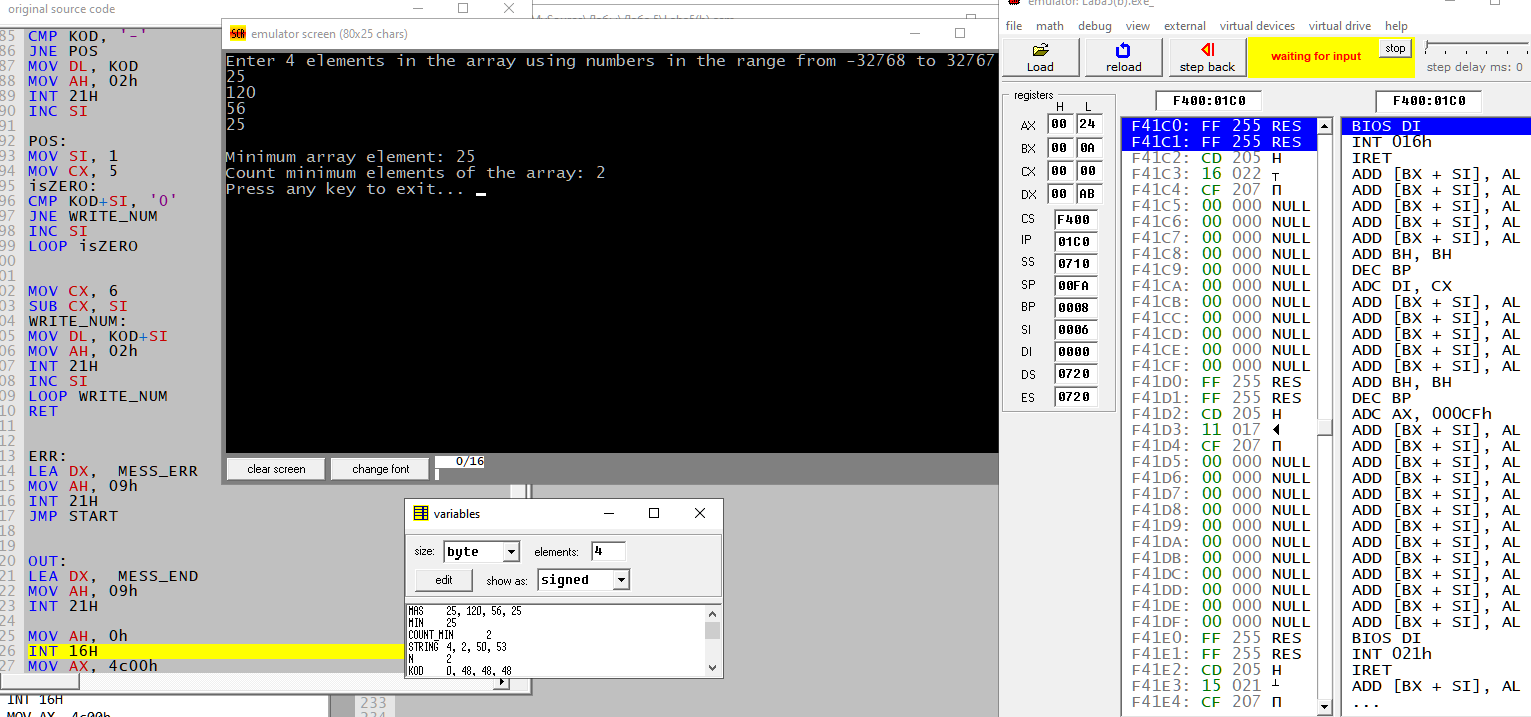
*MOV AX, 4c00h*

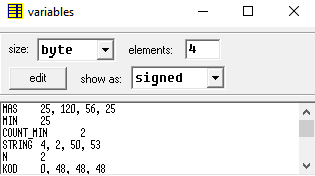
*INT 21H*

*Cseg ends*

*End START*

б)





*Stack segment para stack 'stack'*

*DW 128 dup ('0')*

*Stack ends*

*Dseg segment para public 'data'*

*MAS DW 2 dup(?)*

*DW 2 dup(?)*

*MIN DW 32767*

*COUNT\_MIN DW 0*

*STRING db 15, 16 dup(?)*

*N DW ?*

*KOD DB 6 DUP(?)*

*MESS0 DB "Enter 4 elements in the array using numbers in the range from -32768 to 32767: ", 0DH,0AH, "$"*

*MESS1 DB 0DH,0AH,"Minimum array element: $"*

*MESS2 DB 0DH,0AH,"Count minimum elements of the array: $"*

*MESS\_END DB 0DH,0AH,"Press any key to exit... $"*

*MESS\_ERR DB "Error! Wrong number. Try again...", 0DH,0AH, "$"*

*Dseg ends*

*Cseg segment para public 'code'*

*Assume cs: cseg, ds: dseg, ss: stack;*

*START:*

*MOV AX, DSEG*

*MOV DS, AX*

*MOV ES, AX*

*MOV CX, 4*

*XOR SI, SI*

*LEA DX, MESS0*

*MOV AH, 09H*

*INT 21H*

*READ:*

*MOV DI, 0*

*LEA DX, STRING*

*MOV AH, 0ah*

*INT 21h*

*MOV DL, 0ah*

*MOV AH, 2*

*INT 21h*

*MOV DL, 0dh*

*MOV AH, 2*

*INT 21h*

*LEA SI, STRING+2*

*CMP [SI], 13*

*JE ERR*

*CMP BYTE PTR [SI], "-"*

*JNE POSITIVE*

*MOV DI, 1*

*INC SI*

*POSITIVE:*

*XOR AX, AX*

*MOV BX, 10*

*PUSH CX*

*isNum:*

*MOV DL, [SI]*

*CMP DL, 13*

*JE END\_NUM*

*CMP DL, '0'*

*JB ERR*

*CMP DL, '9'*

*JA ERR*

*SUB DL, '0'*

*MUL BL*

*ADD AL, DL*

*INC SI*

*JMP isNum*

*END\_NUM:*

*POP CX*

*CMP DI, 1*

*JNE END\_VALID\_NUM*

*NEG AX*

*END\_VALID\_NUM:*

*LEA SI, MAS*

*ADD SI, BP*

*MOV [SI], AX*

*ADD BP, 2*

*LOOP READ*

*XOR SI, SI*

*MOV CX, 4*

*CYCLE:*

*MOV AX, MIN*

*CMP MAS[SI], AX*

*JL isMIN*

*JMP NEXT\_EL*

*isMIN:*

*MOV AX, MAS[SI]*

*MOV MIN, AX*

*NEXT\_EL:*

*ADD SI, 2*

*LOOP CYCLE*

*MOV CX, 4*

*XOR SI, SI*

*FIND\_COUNT\_MIN:*

*MOV AX, MAS[SI]*

*CMP AX, MIN*

*JE INCREMENT*

*JMP NEXT*

*INCREMENT:*

*INC COUNT\_MIN*

*NEXT:*

*ADD SI, 2*

*LOOP FIND\_COUNT\_MIN*

*PRINT\_RESULT:*

*LEA DX, MESS1*

*MOV AH, 09h*

*INT 21H*

*MOV AX, MIN*

*MOV N, AX*

*CALL PRINT\_NUM*

*LEA DX, MESS2*

*MOV AH, 09h*

*INT 21H*

*MOV AX, COUNT\_MIN*

*MOV N, AX*

*MOV KOD, 0*

*CALL PRINT\_NUM*

*JMP OUT*

*PRINT\_NUM:*

*CMP N, 0*

*JGE NUM\_SET*

*NEG N*

*MOV KOD, "-"*

*NUM\_SET:*

*MOV BX, 10*

*MOV DX, 0*

*MOV AX, N*

*DIV BX*

*ADD DL, '0'*

*MOV KOD+5, DL*

*MOV DL, 0*

*DIV BX*

*ADD DL, '0'*

*MOV KOD+4, DL*

*MOV DL, 0*

*DIV BX*

*ADD DL, '0'*

*MOV KOD+3, DL*

*MOV DL, 0*

*DIV BX*

*ADD DL, '0'*

*ADD AL, '0'*

*MOV KOD+2, DL*

*MOV KOD+1, AL*

*CMP KOD, '-'*

*JNE POS*

*MOV DL, KOD*

*MOV AH, 02h*

*INT 21H*

*INC SI*

*POS:*

*MOV SI, 1*

*MOV CX, 5*

*isZERO:*

*CMP KOD+SI, '0'*

*JNE WRITE\_NUM*

*INC SI*

*LOOP isZERO*

*MOV CX, 6*

*SUB CX, SI*

*WRITE\_NUM:*

*MOV DL, KOD+SI*

*MOV AH, 02h*

*INT 21H*

*INC SI*

*LOOP WRITE\_NUM*

*RET*

*ERR:*

*LEA DX, MESS\_ERR*

*MOV AH, 09h*

*INT 21H*

*JMP START*

*OUT:*

*LEA DX, MESS\_END*

*MOV AH, 09h*

*INT 21H*

*MOV AH, 0h*

*INT 16H*

*MOV AX, 4c00h*

*INT 21H*

*Cseg ends*

*End START*

Вывод: с помощью данной лабораторной работы я закрепил лекционный материал по командам прерывания на языке ассемблера и приобрёл практические навыки использования этих команд в программах с операциями ввода-вывода.