



Міністерство освіти і науки України  
Національний технічний університет України  
“Київський політехнічний інститут імені Ігоря Сікорського”  
Факультет інформатики та обчислювальної техніки  
Кафедра інформатики та програмної інженерії

## Лабораторна робота №5

### Системне програмне забезпечення

Виконав  
студент групи ІТ-03:

Чабан А.Є.

Перевірив:

Стельмах О.П.

Київ 2021

**Тема:** макрозасоби мови асемблер.

**Завдання:**

Скласти програму на нижче наведені завдання:

- 1) переписати програму комп'ютерного практикуму No 2 з використанням макросів;
- 2) переписати програму комп'ютерного практикуму No 3 з використанням макросів;
- 3) переписати одну програму (на вибір викладача) комп'ютерного практикуму No 4 з використанням макросів.

**Код програми:**

**Л/р №2:**

```
STSEG SEGMENT PARA STACK 'STACK'
DB 64 DUP ( 'STACK' )
STSEG ENDS

INIT_MACROS MACRO
PUSH DS
MOV AX, 0
PUSH AX
MOV AX, DSEG
MOV DS, AX
LEA DX, INPUT
MOV AH, 9
INT 21H
LEA DX, COMP
MOV AH, 10
INT 21H
ENDM

SOLVE MACRO
CALL CHECKFUNC
TEST DX, 7
JNZ ERRPOP
IMUL MULTIPLIER
JO ERRPOP
CALL PRINTFUNC
LEA DX, OUTPUT
```

```

MOV AH, 9
INT 21H
LEA DX, RESULT
MOV AH, 9
INT 21H
JMP END_MAIN
ERRPOP:
LEA DX, ERROR
MOV AH, 9
INT 21H

ENDM

DSEG SEGMENT PARA PUBLIC 'DATA'
COMP DB 6, ?, 6 DUP ( '?' )
INPUT DB 13, 10, '$'
ERROR DB 13, 10, 'Input must include only numbers, or have result less than +-
32k. $'
OUTPUT DB 13, 10, '$'
MULTIPLIER DW 23
ERRCHECK DW 10
RESULT DB 7 DUP ( '?' )
DSEG ENDS

CSEG SEGMENT PARA PUBLIC 'CODE'
MAIN PROC FAR
ASSUME CS: CSEG, DS: DSEG, SS: STSEG

INIT_MACROS ; инициализируем и считываем с клавиатуры
SOLVE

END_MAIN:
RET
MAIN ENDP

CHECKFUNC PROC NEAR ; функция проверки перед умножением
LEA SI, COMP+1 ; comp + 1, т.к сначала записан максимальный размер COMP DB 6, ?,
потом реальный
MOV CX, 0
XOR AX, AX
XOR BX, BX
MOV DX, 0

```

```
MOV CL, [SI]
INC SI
MOV DI, 0
MOV BL, [SI]
CMP BL, '-' ; если 1й символ -, то идем по лейблу
JNE DIGIT_LOOP
MOV DI, 1
DEC CX
INC SI
```

```
DIGIT_LOOP:
MOV BL, [SI]
INC SI
CMP BL, '0'
JB ERROR_MSG
CMP BL, '9'
JA ERROR_MSG

SUB BL, '0'
MUL ERRCHECK
JO ERROVRFL
ADD AX, BX
JO ERROVRFL
LOOP DIGIT_LOOP
MOV DX, 0
TEST DI, DI
JZ END_PROC
NEG AX
JMP END_PROC
ERROR_MSG:
ERROVRFL:
MOV DL, 7
END_PROC:
RET
CHECKFUNC ENDP
```

```
PRINTFUNC PROC NEAR ; вывод
LEA SI, RESULT
XOR CX, CX
OR AX, AX
JNS NOT_NEG
MOV DL, '-'
MOV [SI], DL
INC SI
```

```

NEG AX

NOT_NEG:
XOR DX, DX
DIV ERRCHECK
ADD DL, '0'
PUSH DX
INC CX
TEST AX, AX
JNZ NOT_NEG
WRITE_LOOP:
POP AX
MOV [SI], AL
INC SI
LOOP WRITE_LOOP
MOV DL, '$'
MOV [SI], DL
RET
PRINTFUNC ENDP

CSEG ENDS
END MAIN

```

### Л/р №3:

```

STSEG SEGMENT PARA STACK 'STACK'
DB 64 DUP ( 'STACK' )
STSEG ENDS

INIT_MACROS MACRO
PUSH DS
MOV AX, 0
PUSH AX
MAINST:
MOV AX, DSEG
MOV DS, AX
LEA DX, INP_MSG

MOV AH, 9
INT 21H
LEA DX, COMP
MOV AH, 10
INT 21H
ENDM

```

```

MINUS_CHECK MACRO
LEA SI, COMP+1
MOV CX, 0
XOR AX, AX
XOR BX, BX
MOV DX, 0
MOV CL, [SI]
INC SI
MOV DI, 0
MOV BL, [SI]
CMP BL, '-'

ENDM

DSEG SEGMENT PARA PUBLIC 'DATA'
INP_MSG DB 13, 10, '$'
COMP DB 7, ?, 7 DUP ( '?' )
ERROR DB 13, 10, 'Input must include only numbers, or have result less than
32k.$'
LEFTOVER DB 13, 10, 'Leftover: $'
OUTPUT DB 13, 10, 'Total: $'
RESUL DB 7 DUP ( '?' )
MYVALUE DW 2 DUP ( '?' )
ERRCHECK DW 10
DSEG ENDS

CSEG SEGMENT PARA PUBLIC 'CODE'
MAIN PROC FAR
ASSUME CS: CSEG, DS: DSEG, SS: STSEG
INIT_MACROS

CALL CHECKFUNC
TEST DX, 7
JNZ ERRPOP
CALL CALCUL
TEST DX, 7
JNZ ERRPOP
PUSH CX
CALL PRINTFUNC
LEA DX, OUTPUT
MOV AH, 9
INT 21H

```

```
LEA DX, RESULT
MOV AH, 9
INT 21H
POP AX
TEST AX, AX
JZ END_MAIN
CALL PRINTFUNC
LEA DX, LEFTOVER
MOV AH, 9
INT 21H
```

```
LEA DX, RESULT
MOV AH, 9
INT 21H
JMP END_MAIN
ERRPOP:
LEA DX, ERROR
MOV AH, 9
INT 21H
END_MAIN:
MOV AH, 8
INT 21H
CMP AL, 27
JNE MAINST
MOV AH, 9
INT 21H
RET
MAIN ENDP
```

```
CHECKFUNC PROC NEAR
MINUS_CHECK
JNE DIGIT_LOOP
MOV DI, 1
DEC CX
INC SI
```

```
DIGIT_LOOP:
MOV BL, [SI]
INC SI
CMP BL, '0'
JB ERROR_MSG
CMP BL, '9'
JA ERROR_MSG
```

```
SUB BL, '0'
MUL ERRCHECK
JO ERROR_OVERFLOW
ADD AX, BX
JO ERROR_OVERFLOW
LOOP DIGIT_LOOP
MOV DX, 0;
TEST DI, DI
JZ END_PROC
NEG AX
JMP END_PROC
ERROR_MSG:
ERROR_OVERFLOW:
MOV DL, 7
END_PROC:
RET
CHECKFUNC ENDP

PRINTFUNC PROC NEAR
LEA SI, RESULT
XOR CX, CX
OR AX, AX
JNS NOT_NEG
MOV DL, '-'
MOV [SI], DL
INC SI
NEG AX
NOT_NEG:
XOR DX, DX
DIV ERRCHECK
ADD DL, '0'
PUSH DX
INC CX
TEST AX, AX
JNZ NOT_NEG
WRITE_LOOP:
POP AX
MOV [SI], AL
INC SI
LOOP WRITE_LOOP

MOV DL, '$'
MOV [SI], DL
RET
```



```
PRINTFUNC ENDP
```

```
CALCUL PROC NEAR
```

```
MOV CX, 0
```

```
CMP AX, 9
```

```
JL LESS
```

```
CMP AX, 9
```

```
JG MORE
```

```
MOV AX, 1
```

```
XOR DX, DX
```

```
JMP EXIT_PROC
```

```
LESS: ;  $(x^3 - 2x^2 + 1) / (x^2 + 1)$ 
```

```
MOV MYVALUE, AX
```

```
MOV BX, AX
```

```
IMUL BX
```

```
JO OVERJMP
```

```
MOV CX, AX
```

```
ADD CX, 1
```

```
JO OVERJMP
```

```
PUSH CX
```

```
PUSH AX
```

```
MOV AX, BX
```

```
IMUL BX
```

```
MOV BX, AX
```

```
MOV AX, 2
```

```
IMUL BX
```

```
JO OVERJMP
```

```
POP BX
```

```
PUSH AX
```

```
MOV AX, BX
```

```
IMUL MYVALUE
```

```
JO OVERJMP
```

```
MOV BX, AX
```

```
POP AX
```

```
SUB BX, AX
```

```
MOV AX, BX
```

```
JO OVERJMP
```

```
JO OVERJMP
```

```
ADD AX, 1
```

```
JO OVERJMP
```

```
MOV DX, 0
```

```

POP BX
CWD
IDIV BX
JO OVERJMP
MOV CX, DX

MOV DX, 0
JMP EXIT_PROC

MORE: ; (x^2 - 8x + 6) / (x^2+1)
MOV BX, AX
IMUL BX
JO OVERJMP
MOV CX, AX
ADD CX, 1
JO OVERJMP
PUSH CX
PUSH AX
MOV AX, 8
IMUL BX
JO OVERJMP
POP BX
SUB BX, AX
MOV AX, BX
JO OVERJMP
ADD AX, 6
JO OVERJMP
MOV DX, 0
POP BX
CWD
IDIV BX
JO OVERJMP
MOV CX, DX

MOV DX, 0
JMP EXIT_PROC
OVERJMP:
MOV DX, 7
EXIT_PROC:
RET
CALCUL ENDP
CSEG ENDS
END MAIN

```

#### Л/р №4:

```
STSEG SEGMENT PARA STACK 'STACK'
DB 64 DUP ( 'STACK' )
STSEG ENDS

INIT_MACROS MACRO
MOV AX, DSEG
MOV DS, AX
LEA DI, ARRONE_INPUT
ENDM

PRINT MACRO
PUSH AX
MOV AH,9
XCHG DX,DI
INT 21H
XCHG DX,DI
POP AX
ENDM

DSEG SEGMENT PARA PUBLIC 'DATA'
ARRONE_INPUT DB 'Enter array: ',13,10,'$'
NUMONE_INPUT DB '$'
INPSEND DB 'Input: $'
SORTFINAL DB 'Sort: $'
SORTMAX DB 'Max number: $'
SORTMIN DB 'Min number: $'
SORTNUMS DB '$'
ERRWARN DB 'Input must include only numbers, or have result less than 32k. $'
ARRTWO_INPUT DB 'Enter second array:',13,10,'$'
ARRTWO_ROWONE DB 'First row: ',13,10,'$'
ARRTWO_ROW TWO DB 'Second row: ',13,10,'$'
FINDTARGET DB 'Enter number to find through i,j: ', 13, 10, '$'
ARR_I DB 'I: $'
ARR_J DB 'J: $'
SORTLENGTH DB 'Error. Minimum length required: $'
SORTSUM DB 'Array sum: $'
SORTRES DB 'Result: ',13,10,'$'
SORTERROR DB 'ERROR! $'
SORTEEMPTY DB 'Input must include numbers. $'
SORTBADINPUT DB 'Input must include only numbers. $'
SORTOVERFLOW DB 'Your input goes beyond 32k. $'
SORTNOTFOUND DB 'NOT FOUND! $'
ENDLINE DB 13,10,'$'
```

```

IS_NEGATIVE DB 0
BUFFER DB 9 DUP('?')
ARRAY_MAX DB 16
ARRAY_MIN DB 1
ARRAY_SIZE DB 0
VAR DW 0
NUM DW 0
ARRAY DW 16 DUP('?')
I DW 0
J DW 0
DSEG ENDS

CSEG SEGMENT PARA PUBLIC 'CODE'
MAIN PROC FAR
ASSUME CS: CSEG, DS: DSEG, SS: STSEG
INIT_MACROS
PRINT
CALL INPUT_ARRAY
LEA DI, INPSEND
PRINT
CALL PRINT_ARRAY

CALL PRINT_ENDLINE
CALL SUM_NUMBERS
CALL BUBBLE_SORT
LEA DI, SORTFINAL
PRINT
CALL PRINT_ARRAY
CALL PRINT_ENDLINE
CALL MAX_NUMBER
CALL MIN_NUMBER
SECOND_ARR_INPUT:
INPUT_FIRST_ROW:
LEA DI, ARRTWO_ROWONE
PRINT
MOV VAR, 0
MOV ARRAY, 0
MOV ARRAY_SIZE, 0
MOV ARRAY_MAX, 8
CALL INPUT_ARRAY
INPUT_SECOND_ROW:
LEA DI, ARRTWO_ROW TWO
PRINT
MOV CL, ARRAY_SIZE
MOV ARRAY_MIN, CL

```

```
MOV ARRAY_MAX, CL
MOV ARRAY_SIZE, 0
CALL INPUT_ARRAY

CALL PRINT_ENDLINE
FIND_ELEMENT:
LEA DI, FINDTARGET
PRINT
LEA DI, NUMONE_INPUT
CALL INPUT_STR
TEST AL, AL
JZ EXIT_MAIN
CALL STR_TO_WORD
JC EXIT_MAIN
CALL FIND_INDEX
JMP FIND_ELEMENT
EXIT_MAIN:
CALL EXIT_PROGRAM
RET
MAIN ENDP
```

```
; ADDITIONAL PROCEDURES
```

```
FIND_INDEX PROC
XOR CX, CX
XOR SI, SI
MOV NUM, AX
MOV CL, ARRAY_MIN
MOV AX, CX
MOV CX, 2
MUL CX
MOV CX, AX
MOV VAR, AX
```

```
FIND_ELEMENT_LOOP:
MOV DX, NUM
CMP DX, ARRAY[SI]
JE NUMBER_FOUND
JNE FIND_ELEMENT_STEP
NUMBER_FOUND:
MOV AX, SI
XOR BX, BX
MOV BX, 2
DIV BL
MOV J, AX
```

```

CMP VAR, SI
JLE SECOND_ROW
FIRST_ROW:
MOV I, 0
JMP PRINT_INDEXES
SECOND_ROW:
MOV I, 1
XOR BX, BX
MOV BL, ARRAY_MIN
SUB J, BX
PRINT_INDEXES:
LEA DI, ARR_I
PRINT
MOV AX, I
CALL PRINT_NUM

CALL PRINT_ENDLINE
LEA DI, ARR_J
PRINT
MOV AX, J
CALL PRINT_NUM
CALL PRINT_ENDLINE
JMP FIND_ELEMENT_STEP
FIND_ELEMENT_STEP:
ADD SI, 2
LOOP FIND_ELEMENT_LOOP
RET
FIND_INDEX ENDP


INPUT_ARRAY PROC
PUSH CX
PUSH SI
XOR SI, SI
XOR CX, CX
ARRAY_INPUT_LOOP:
MOV CH, ARRAY_SIZE
MOV CL, ARRAY_MAX
CMP CL, CH
JE EXIT_ARRAY_INPUT_LOOP
CLC
LEA DI, NUMONE_INPUT
INPUT_LOOP:
XOR AX, AX

```

```

CALL INPUT_STR
TEST AL, AL
JZ ON_ENTER_TAP
CALL STR_TO_WORD
JNC STEP
JC RE_ENTER
RE_ENTER:
CALL PRINT_ENDLINE
LEA DI, ERRWARN
PRINT
JMP INPUT_LOOP
NOT_ENOUGH_NUMBERS:
CALL PRINT_ENDLINE
LEA DI, SORTLENGTH
PRINT
XOR AX, AX
MOV AL, ARRAY_MIN
CALL PRINT_NUM
JMP RE_ENTER
ON_ENTER_TAP:
MOV CH, ARRAY_SIZE
MOV CL, ARRAY_MIN
CMP CH, CL
JNB EXIT_ARRAY_INPUT_LOOP
CLC
JMP NOT_ENOUGH_NUMBERS

STEP:
CALL PRINT_ENDLINE
MOV SI, VAR
MOV ARRAY[SI], AX
INC ARRAY_SIZE
ADD VAR, 2
JMP ARRAY_INPUT_LOOP
EXIT_ARRAY_INPUT_LOOP:
CALL PRINT_ENDLINE
POP SI
POP CX
RET
INPUT_ARRAY ENDP

SUM_NUMBERS PROC

```

```
PUSH CX
PUSH BX
PUSH AX
XOR CX, CX
MOV CL, ARRAY_SIZE
LEA BX, ARRAY
XOR AX, AX
SUM_NUMBERS_LOOP:
ADD AX, [BX]
JC OVERFLOW
JO OVERFLOW
ADD BX, 2

LOOP SUM_NUMBERS_LOOP
LEA DI, SORTSUM
PRINT
CALL PRINT_NUM
CALL PRINT_ENDLINE
JMP SUM_NUMBERS_EXIT
OVERFLOW:
LEA DI, SORTOVERFLOW
CALL PRINT_ENDLINE
PRINT
CALL PRINT_ENDLINE
JMP SUM_NUMBERS_EXIT
SUM_NUMBERS_EXIT:
POP AX
POP BX
POP CX
RET
SUM_NUMBERS ENDP
```

```
BUBBLE_SORT PROC
PUSH CX
PUSH SI
PUSH BX
PUSH DX
PUSH AX
XOR CX, CX
XOR SI, SI

XOR BX, BX
XOR AX, AX
```



```

XOR DX, DX
CMP ARRAY_SIZE, 1
JE BUBBLE_SORT_EXIT
CLC
MOV CL, ARRAY_SIZE
DEC CX
OUTER_LOOP:
MOV BX, CX
MOV SI, 0
INNER_LOOP:
MOV AX, ARRAY[SI]
MOV DX, ARRAY[SI+2]
CMP AX, DX
JL NOSWAP
MOV ARRAY[SI],DX
MOV ARRAY[SI+2],AX
NOSWAP:
ADD SI, 2
DEC BX
JNZ INNER_LOOP
LOOP OUTER_LOOP
BUBBLE_SORT_EXIT:
POP AX
POP DX

POP BX
POP SI
POP CX
RET
BUBBLE_SORT ENDP

```

```

MIN_NUMBER PROC
LEA DI, SORTMIN
PRINT
MOV AX, ARRAY[0]
CALL PRINT_NUM
CALL PRINT_ENDLINE
RET
MIN_NUMBER ENDP

```

```

MAX_NUMBER PROC

```

```
LEA DI, SORTMAX
PRINT
PUSH SI
XOR SI, SI
MOV SI, VAR
MOV AX, ARRAY[SI-2]
CALL PRINT_NUM
CALL PRINT_ENDLINE
POP SI
RET
MAX_NUMBER ENDP
```

```
PRINT_ARRAY PROC
PUSH CX
PUSH SI
XOR CX, CX
XOR SI, SI
MOV CL, ARRAY_SIZE
PRINT_ARRAY_LOOP:
XOR AX, AX
MOV AX, ARRAY[SI]
CALL PRINT_NUM
LEA DI, SORTNUMS
PRINT
ADD SI, 2
LOOP PRINT_ARRAY_LOOP
POP SI
POP CX
RET
PRINT_ARRAY ENDP
```

```
INPUT_STR PROC
MOV AH, 0AH
MOV [BUFFER], 7
MOV BYTE[BUFFER+1], 0
LEA DX, BUFFER
INT 21H
MOV AL, [BUFFER+1]
ADD DX, 2
RET
INPUT_STR ENDP
```

```
STR_TO_WORD PROC
PUSH CX
PUSH SI
PUSH DI
PUSH BX
PUSH DX
MOV BX,DX
MOV BL,[BX]
CMP BL,'-'
JE NEGATIVE_NUMBER
JMP POSITIVE_NUMBER
POSITIVE_NUMBER:
MOV IS_NEGATIVE, 0
JMP UNSIGNED_STR_TO_WORD
NEGATIVE_NUMBER:
INC DX
DEC AL
MOV IS_NEGATIVE, 1
JMP UNSIGNED_STR_TO_WORD
UNSIGNED_STR_TO_WORD:
MOV SI, DX
MOV DI, 10
XOR CX, CX
MOV CL,AL
XOR AX, AX

XOR BX, BX
UNSIGNED_STR_TO_WORD_LOOP:
MOV BL,[SI]
INC SI
CMP BL,'0'
JL INCORRECT_SYMBOL_ERROR
CMP BL,'9'
JG INCORRECT_SYMBOL_ERROR
SUB BL,'0'
MUL DI
JC OVERFLOW_ERROR
ADD AX, BX
JC OVERFLOW_ERROR
LOOP UNSIGNED_STR_TO_WORD_LOOP
CMP IS_NEGATIVE, 1
JE MAKE_NEGATIVE
```

```
CLC
JMP STR_TO_WORD_EXIT
MAKE_NEGATIVE:
NEG AX
CLC
JMP STR_TO_WORD_EXIT
INCORRECT_SYMBOL_ERROR:
LEA DI, SORTBADINPUT
CALL ERROR_HANDLER
JMP STR_TO_WORD_EXIT
```

```
OVERFLOW_ERROR:
LEA DI, SORTOVERFLOW
CALL ERROR_HANDLER
JMP STR_TO_WORD_EXIT
STR_TO_WORD_EXIT:
POP CX
POP SI
POP DI
POP BX
POP DX
RET
STR_TO_WORD ENDP
```

```
PRINT_ENDLINE PROC
LEA DI, ENDLINE
PRINT
RET
PRINT_ENDLINE ENDP
```

```
ERROR_HANDLER PROC
PRINT
STC
RET
ERROR_HANDLER ENDP
```

```
EXIT_PROGRAM PROC
MOV AH, 4CH
INT 21H
RET
```

```
EXIT_PROGRAM ENDP
```

```
PRINT_NUM PROC
```

```
PUSH AX
```

```
PUSH BX
```

```
PUSH DX
```

```
PUSH CX
```

```
MOV BX, AX
```

```
OR BX, BX
```

```
JNS POSITIVE
```

```
MOV AH,2
```

```
MOV DL, '-'
```

```
INT 21H
```

```
NEG BX
```

```
POSITIVE:
```

```
MOV AX, BX
```

```
XOR CX, CX
```

```
MOV BX, 10
```

```
PRINT_NUM_LOOP:
```

```
XOR DX, DX
```

```
DIV BX
```

```
ADD DL, '0'
```

```
PUSH DX
```

```
INC CX
```

```
TEST AX, AX
```

```
JNZ PRINT_NUM_LOOP
```

```
OUTPUT_LOOP:
```

```
MOV AH,2
```

```
POP DX
```

```
INT 21H
```

```
LOOP OUTPUT_LOOP
```

```
POP CX
```

```
POP DX
```

```
POP BX
```

```
POP AX
```

```
RET
```

```
PRINT_NUM ENDP
```

```
CSEG ENDS
```

```
END MAIN
```

**Висновок:**

Отже, під час виконання даної лабораторної роботи я навчився писати макроси та ознайомився з ними на практиці.