Міністерство освіти і науки України

Національний університет «Запорізька Політехніка»

Кафедра програмних засобів

ЗВІТ

з лабораторної роботи №3

з дисципліни «Архітектура Комп’ютера та Низькорівневе Програмування»

На тему «ТИПИ ДАНИХ. ВИВЕДЕННЯ СИМВОЛІВ ТА СИМВОЛЬНИХ РЯДКІВ»

Варіант №20

Виконав:

Студент групи КНТ-122 О. А. Онищенко

Прийняли:

Ст. Викладач О. І. Качан

Доцент А. Є. Казурова

2023

Текст завдання

1. У текстовому редакторі створити початковий код на мові асемблера для exe-програми з наступними вимогами до коду:

- у сегменті даних сформувати змінні різноманітних типів та розмірів, ініціалізувати їх різними способами та значеннями;

- серед сформованих змінних мають бути також символи, символьні рядки, масиви з 1/2/4-байтовими елементами різних розмірностей (1/2/3);

- реалізувати виведення до консолі будь-якого друкованого символу;

- реалізувати виведення до консолі символьних рядків;

- продемонструвати виведення до консолі керуючих символів (CR/LF/BS або інших);

- виведення символів до консолі реалізовувати функціями 21h-переривання (2 та 9 обов'язково, іншими - на вибір);

- потрібні блоки програмного коду сформувати до модулів (програмних процедур);

- у програмному коді використовувати різні методи адресації до даних;

- для реалізації виведення символів та рядків, продемонструвати роботу простої інструкції LOOP;

- обов'язково у потрібних місцях коду прописувати коментарі.

2. Над створеним початковим кодом програми виконати процеси компіляції, лінкування та позбавитись від помилок (якщо вони є).

3. Виконати запуск програми у консолі з демонстрацією результатів роботи. Перенаправити потік даних з консолі до текстового файлу з наступною перевіркою коректності виведених керуючих символів.

Код програми

```
.model small    ; set program model as small
.stack 100h     ; set stack size to 100h

cseg segment para public 'code'     ; declare code segment
    assume cs:cseg, ss:sseg, es:nothing ; set segment register to
corresponding ones

start:  ; declare program entry point
    assume ds:dseg  ; set data segment register
    mov bx, dseg    ; add data segment to bx register
    mov ds, bx  ; set ds register to bx register

    call main   ; call main function

    mov ah, 04Ch     ; exit to OS
    mov bl, 06Ch     ; set error code to 108 in hex
    int 21h     ; call interrupt

main proc near  ; declare main function
    ; output new line
    lea dx, new_line
    call outputString

    ; output a byte character
    mov dl, my_byte
    ; use null terminator to output the character itself
    add dl, '$'
    call outputSymbol
    ; output new line
    lea dx, new_line
    call outputString

    ; output new line
    lea dx, new_line
    call outputString

    ; initialize counter to 5
    mov cx, 5
    ; initialize pointer to my_array
    mov si, offset my_array

    ; loop 5 times
    loop_start:
        ; load byte at memory address pointed to by si into dl
        mov dl, [si]
```

```
        ; add ASCII value of $ to dl
        add dl, '$'
        ; output resulting character to console
        call outputSymbol
        ; load memory address of _space string into dx
        lea dx, _space
        ; output space character to console
        call outputString

        ; increment pointer to point to next byte in array
        inc si
        ; repeat loop until cx reaches 0
        loop loop_start

    ; output new line
    lea dx, new_line
    call outputString

    ; initialize CX register with the number of characters in the message
string
    mov cx, 8
    ; initialize SI register with the offset of the message string
    mov si, offset message

    ; loop through each character in the message string and print it
    print_message:
        mov dl, [si]     ; load the current character into DL register
        call outputSymbol   ; print the character in DL register
        lea dx, _space   ; load the offset of the space character into DX
register
        call outputString   ; print the space character
        inc si           ; move to the next character in the message
string
        loop print_message  ; repeat until all characters have been
printed

    ; output new line
    lea dx, new_line
    call outputString

    ; output new line
    lea dx, new_line
    call outputString
    ; output horizontal line
    lea dx, horizontal_line
    call outputString
    ; output new line
    lea dx, new_line
```

```
    call outputString
    ; output my name
    lea dx, my_string
    call outputString
    ; output new line
    lea dx, new_line
    call outputString
    ; output my group
    lea dx, student_group
    call outputString
    ; output new line
    lea dx, new_line
    call outputString
    ; output new line
    lea dx, new_line
    call outputString
    ; output current year
    lea dx, current_year
    call outputString
    ; output new line
    lea dx, new_line
    call outputString
    ; output horizontal line
    lea dx, horizontal_line
    call outputString
    ; output new line
    lea dx, new_line
    call outputString

    ret
main endp   ; end main function

outputString proc near
    ; clear the AX register
    sub ax, ax
    ; set AH to 09h to indicate that we want to output a string
    mov ah, 09h
    ; call interrupt 21h to output the string
    int 21h
    ; return from the procedure
    ret
outputString endp

outputSymbol proc near
    ; clear the AX register
    sub ax, ax
    ; set AH to 02h to indicate that we want to output a character
    mov ah, 02h
```

```
    ; call interrupt 21h to output the character
    int 21h
    ; return from the procedure
    ret
outputSymbol endp
cseg ends   ; end code segment

sseg segment para stack 'stack'     ; declare stack segment
    db 256 dup(?)       ; reserve memory for stack
sseg ends   ; end stack segment

dseg segment para public 'data'     ; declare data segment
    ; define a space character and initialize it
    _space db ' ', '$'

    ; define a new line character and initialize it
    new_line db 0Dh, 0Ah, '$'

    ; define a byte and initialize it with a value
    my_byte db 10h

    ; define a word and initialize it with a value
    my_word dw 1234h

    ; define a dword and initialize it with a value
    my_dword dd 5678h

    ; define necessary string  and initialize them with corresponding
values
    my_string db '      Oleh Onyshchenko', '$'
    student_group db '          KHT-122', '$'
    horizontal_line db '------------------------------', '$'
    current_year db '           2023', '$'
    message db 'Assembly', '$'

    ; define an array of bytes and initialize it with values
    my_array db 1, 2, 3, 4, 5

    ; define an array of words and initialize it with values
    my_matrix dw 1, 2, 3, 4, 5

    ; define a 2-dimensional array of doublewords and initialize it with
values
    my_dword_matrix dd 1, 2, 3, 4
                dd 5, 6, 7, 8
                dd 9, 10, 11, 12
dseg ends   ; end data segment
```

```
end start   ; end program execution
```

Приклад роботи