МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»



**Лабораторна робота №1**

з курсу «Системне програмування»

**Варіант 14**

Виконав студент

групи КНз-2

Чалий Михайло

Перевірив

Мазур В.В.

Львів 2014

# Мета роботи

Навчитись працювати з асемблером MASM, TASM

# Теоретичні відомості

Асемблер є мовою програмування низького рівня Він забезпечує написання, відладку та виконання програм на рівні близькому до машинних команд. Основна область застосування Асемблера - написання високоефективних програм системного характеру (драйверів, динамічна відеографіка, обробка сигналів в режимі реального часу, тощо).

Програмування на Асемблері базується на системі програмування, яка забезпечує перехід від початкового файлу з текстом програми до завантажуваного файлу, готового до виконання (інтерпретації мікропроцесором).

Реалізація програми на Асемблері охоплює такі етапи:

- написання програми на мові Асемблера, ввід та оформлення її у вигляді початкового файлу за допомогою текстового редактора (наприклад, Nогtоn Соmmander, FAR, Блокнот (шрифт – Terminal) );

- трансляція початкового файлу в об'єктний файл;

- редагування зв'язків на основі декількох об'єктних файлів і формування завантажуваного файлу для інтерпретації мікропроцесором.

Така схема забезпечує незалежне асемблювання кількох початкових файлів з подальшим їх об'єднанням на рівні об'єктних файлів в один завантажуваний. Вказаний підхід реалізує концепції модульного програмування (розробка та реалізація програм по частинах - модулях). Відповідно розглядають початковий модуль, об'єктний модуль та завантажуваний модуль. Відомі і простіші схеми асемблювання, які забезпечують, перехід, від рівня початкових файлів до завантажуваних.

**2.1. Структура програми на Асемблері**

Програма. на Асемблері складається з модулів. Модулі складаються з сегментів. Сегменти складаються з процедур, команд та директив.

Початковий програмний модуль складається з операторів. Виділяються оператори трьох типів: команди, директиви, коментар. Оператори команд визначають операції, які виконує процесор. Директиви вказують Асемблеру та Редактору зв'язків як оформляти (об'єднувати) команди та модулі в програму. Директиви також забезпечують розподіл пам'яті та визначення даних.

Оператори команд складаються з символічного імені, коду операції, поля операндів та коментаря. Код операції визначає команду і відрізняє її від директиви. При написанні програми на Асемблері символічне ім’я ставиться перед кодом операції через двокрапку і називається міткою (внутрішньою). Мітка визначає адресу команди в оперативній пам'яті, що може використовуватись, наприклад, операторами переходу.

Операнди (два, один, або відсутні) визначають дані над якими виконуються операції процесором. Операнди відділяються один від одного комою, а від коду команди - пропуском.

Коментар відділяється від полів команди та операндів пропуском та крапкою з комою. Він не приймає участі в асемблюванні і служить лише для документування програми.

Оператори директив складаються з символічного імені, коду псевдооперації, поля операндів та коментаря. Структура директиви аналогічна структурі команди. Код псевдооперації визначає зміст директиви. Операндів директиви може бути декілька і вони розділяються комами, а від коду псевдооперації відділяються пропуском.

Символічне ім'я, яке стоїть на початку директиви розподілу пам'яті (директиви визначення) називається змінною. Символічне ім'я директиви відділяється від коду пропуском, а не двокрапкою, як мітка. Директива

Коментар записується через пропуск та крапку з комою.

Програма складається з модулів, які можуть бути паралельно розроблені та реалізовані. Кожний модуль складається з операторів, які обробляються (транслюються) Асемблером. Для транслятора признаком кінця операторів модуля є директива - ЕND (Кінець). Крім того операнд директиви END визначає точку входу програми.

Кожний модуль поділяється на окремі частини директивами сегментації, які визначають початок (SEGMENT) та кінець (ENDS) сегмента. Кожен із сегментів (типів сегментів) має певні функції і забезпечує структуризацію програм. Всі сегменти поіменовані (SSEG - стековий сегмент, CSEG - програмний сегмент, DSEG - сегмент даних, ESEG - додатковий сегмент даних). Директива ASSUME зв'язує довільні імена сегментів з сегментними регістрами. Сегментні регістри містять базові адреси відповідних сегментів і використовуються при обчислені фізичних адрес даних і команд.

# Лістінг t1.asm

; Constants

REC\_WIDTH equ 75

REC\_HEIGHT equ 45

REC\_COLOR equ 5

sseg segment stack

db 256 dup(?)

sseg ends

dseg segment

start\_x dw ?

exit\_message db "Press any key to exit...", "$"

dseg ends

cseg segment

assume ss:sseg, cs:cseg, ds:dseg

start:

jmp main

main:

mov ax,seg dseg

mov ds,ax

; VGA Mode

mov ax,0012h

int 10h

; Mouse

mov ax,0

int 33h

; Show cursor

mov ax,1

int 33h

ms:

mov ax,3 ; Read mouse state

int 33h

test bx,001b ; Test if left button not pressed

jz ms

mov ax,2 ; Hide cursor

int 33h

mov start\_x,cx ; X

mov si,cx

add si,REC\_WIDTH

mov di,dx ; Y

add di,REC\_HEIGHT

; Draw rect

mov ah,0ch ; Pixel output mode

mov bh,0 ; Page

mov al,REC\_COLOR

draw\_line:

mov cx,start\_x ; Start from start\_x

draw\_pixel:

int 10h

inc cx ; Next column

cmp cx,si

jl draw\_pixel

inc dx ; Next row

cmp dx,di

jl draw\_line

; Print exit message

mov ah,09

lea dx,exit\_message

int 21h

; Wait user input

mov ah,08h

int 21h

; Clean view port(FreeDOS)

mov ax,0003h ; Text mode

int 10h

; Exit all good

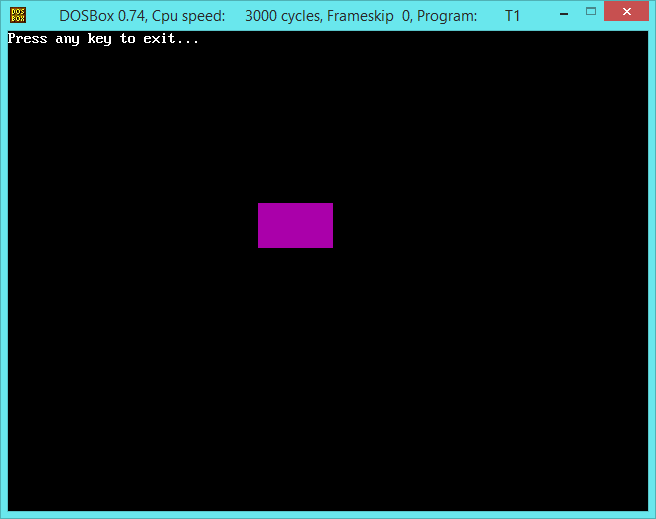
mov ax,4c00h

int 21h

cseg ends

end start

### Результат



# Висновок

В результаті виконання роботи навчився працювати з асемблером MASM, TASM, JWASM(MASM сумісний)