**Лабораторна робота № 4**
**Завантаження та виконання програм DOS.**
**Організація програм *.EXE та *.COM**

**Мета роботи**: ознайомитися зі структурою виконуваних програм *.EXE та *.COM та їх образом в пам’яті.

**Порядок роботи:**

1. Ознайомитися зі структурою програми *.EXE та образом такої програми у пам’яті.
2. Ознайомитися зі структурою програми *.COM та образом такої програми у пам’яті.
3. Засвоїти особливості створення виконуваних програм *.EXE та *.COM.
4. Підготувати .EXE та .COM програми для виведення на екран повідомлення HELLO WORD! з варіантом опису даних:
   msg db "Hello Word!", 0Dh, 0Ah, '$'
   0Dh – символ повернення каретки (Carriage Return),
   0Ah - символ переведення рядка
5. Продемонструвати роботу .EXE – та .COM-програм, пояснити розподіл пам’яті для цих програм.
6. У DEBUG за допомогою команди D CS:0000 ви маєте можливість переглянути машинний код програми для exe-програми.
7. У DEBUG за допомогою команди D DS:100 ви маєте можливість переглянути дані.

**Теоретична частина**

Операційна система MS DOS передбачає два типи виконуваних програм, які мають розширення *.COM та *.EXE .

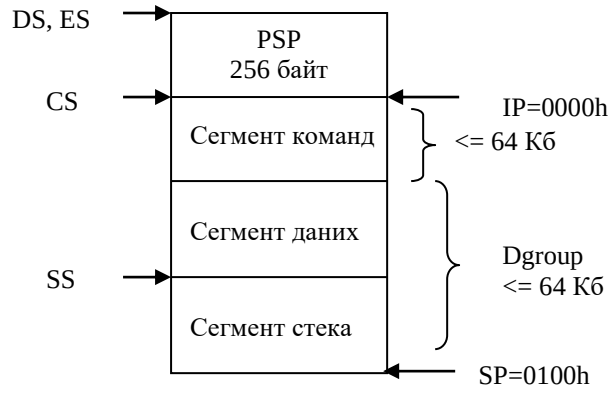
Перед завантаженням в оперативну пам’ять COM- та EXE-програм DOS визначає спеціальну область оперативної пам’яті розміром 256 (100h) байт - сегментну адресу, яка називається префіксом програмного сегменту (PSP – Program Segment Prefics). PSP може використовуватися в програмі для визначення імен файлів та параметрів з командного рядка, які вводяться при запуску програми на виконання, обсягу допустимої пам'яті, змінних оточення системи та ін.

При завантаженні програми в оперативну пам’ять DOS ініціалізує як мінімум три сегментних регістри: CS, DS, SS (додатковим може бути ES). Код та дані переміщуються з файлу на диску в оперативну пам’ять, а адреси цих сегментів заносяться у CS та DS відповідно. Сегмент стека або виділяється в області, що вказана в програмі, або співпадає (якщо він явно не описаний в програмі) з самим першим сегментом програми. Адреса сегменту стека розташовується в регістрі SS. Програма може мати декілька кодових сегментів та сегментів даних і в процесі виконання за допомогою спеціальних команд здійснюється переключення між ними.

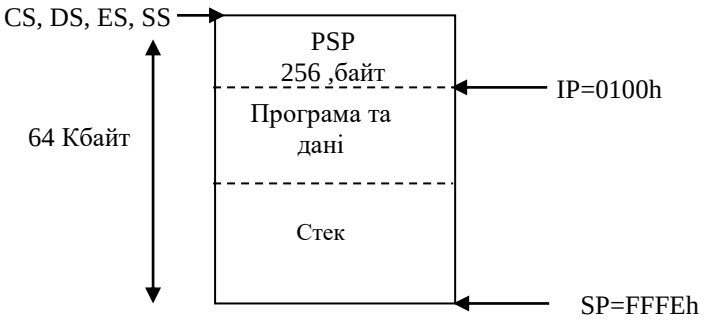
Для того, щоб адресувати одночасно два сегменти даних, наприклад, для виконання операції пересилання з однієї області пам’яті в іншу, можна використовувати регістр додаткового сегмента ES. Кодовий сегмент та сегмент стеку завжди визначається вмістом своїх регістрів (CS та SS), в кожний момент виконання програми завжди використовується якийсь один кодовий сегмент та один сегмент стека.

Усі сегменти можуть використовувати різні області пам’яті, а можуть частково або повністю перекриватися. Кодовий сегмент повинен обов’язково описуватися в програмі, усі інші сегменти можуть бути відсутніми. У цьому випадку при завантаженні програми в оперативну пам’ять DOS ініціює регістри DS та ES значенням адреси префікса програмного сегмента PSP. Регістр SS при цьому ініціалізується значенням сегменту, що знаходиться одразу за PSP, тобто першого сегменту програми. При цьому слід враховувати, що стек «зростає вниз» (при розміщенні у стек вміст регістра SP, що вказує на вершину стека, зменшується, а при зчитуванні зі стека - збільшується). Тому при розміщенні у стек будь-яких значень вони можуть затерти PSP, в зв’язку з цим слід завжди сегмент стеку описувати в програмі явно та задавати для нього розмір, достатній для нормальної роботи. У вказівник команд IP завантажується зсув точки входу в програму (вибирається з операнда директиви END), у вказівник стека SP – зсув кінця сегмента стека.

Образ пам’яті програми типу *.EXE має такий вигляд:




Образ пам’яті програми типу *.COM має такий вигляд:



Основні відмінності цих програм полягають у наступному:

*програми типу *.COM* (двійковий образ коду та даних програми)

- складаються тільки з одного сегменту, в якому розміщується і код програми, і дані, і стек;
- файл COM-формату не містить переміщуваних адрес;
- COM-файл завантажується, починається з адреси PSP:0100h;
- CS, DS, ES, SS вказують на PSP;
- SP вказує на кінець сегмента PSP (зазвичай 0FFFEh);
- IP містить 100h (перший байт модуля) в результаті команди JMP PSP:100h;
- розмір програми типу *.COM не може перевищувати 64 Кбайт;
- використовується *модель пам'яті TINY*;
- образ COM-файла зчитується з диску та розташовується в пам'яті, починаючи PSP:0100h;
- програма генерує стек автоматично, тому в самій асемблерній програмі стек має бути відсутнім; усі дані повинні бути визначені у сегменті коду;

*в програмах типу *.EXE* міститься спеціальний заголовок, за допомогою якого завантажувач виконує налаштування посилань на сегменти у завантаженому модулі

- використовуються окремі сегменти і для коду, і для даних, і для стеку;
- розмір програми типу *.EXE не має жорстких обмежень, тому що програми може мати будь-яку кількість сегментів команд та даних;
- EXE-файл завантажується, починається з адреси PSP:0100h;
- DS та ES вказують на початок PSP;
- CS, IP, SS, SP ініціалізуються значеннями, зазначеними в заголовку EXE;
- використовується *модель пам'яті SMALL*, яка передбачає розташування структурних частин програми у двох сегментах: сегменту кода програми (64 Кбайт) та сегменту даних і стека , що належать одній групі Dgroup (64 Кбайт).

Завершення програми можна виконати декількома способами:

- через функцію 4CH (EXIT) переривання 21H в будь-який момент, незалежно від значень регістрів;
- через функцію 00H переривання 21H або переривання INT 20H, у випадку коли CS вказує PSP.

Програми типу *.EXE та *.COM розрізняються форматом вхідного тексту, процедурою підготовки виконуваного файлу, а також форматами завантажувальних файлів.

Модель пам'яті неявно задає атрибути спрощених директив, що визначають дії компоновщика Turbo Linker при формуванні виконуваного файла програми. Коротка характеристика спрощеної моделі пам'яті SMALL наведена у таблиці:

*Сегменти спрощеної моделі пам'яті SMALL*

| *Директива* | *Ім'я* | *Вирівнювання* | *Поєднання* | *Клас* | *Група* |
|---|---|---|---|---|---|
| Codeseg | _Text | Word | Public | 'Code' | |
| Dateseg | _Data | Word | Public | 'Data' | Dgroup |
| Stack | Stack | Para | Stack | 'Stack' | Dgroup |

**Ім'я** – ідентифікатор конкретного сегменту, що використовується при призначенні адреси та поєднанні з іншими сегментами.

**Вирівнювання –** вказує граничні значення на початок сегмента. В процесі асемблювання, якщо поточна позиція на початку сегмента не задовольняє встановленому атрибуту, лічильник адреси збільшується на відповідну величину, зміщуючи початок сегменту в область старших адрес (*word* – початок сегменту повинен мати парну адресу; *para* парна адреса, яка є кратною параграфу, тобто 16 біт).

**Поєднанн**я – встановлює правила поєднання декількох сегментів з однаковимим іменем. Параметр *Public* послідовно поєднує сегменти з однаковим іменем в один великий сегмент, що має адресу першого з поєднуючих сегментів.

**Клас –** виконує роль категорії сегмента. Усі сегменти однакового класу під час роботи компоновщика розташовуються один за одним у пам'яті.

**Група –** дозволяє здійснити доступ до даних з усіх сегментів, що знаходяться в групі, за допомогою завантаження адреси групи у сегментний регістр**.**

Використання директиви *Model* дозволяє використовувати службові ідентифікатори, за допомогою яких можна отримати адреси сегментів, що використовуються:

*@code* – 16-розрядна адреса сегмента коду;

*@data* – 16-розрядна адреса сегмента даних типу near;

*@stack* – 16-розрядна адреса сегмента стека.

Наприклад,
```
MOV AX, @DATA
MOV DS, AX
```
ініціалізація сегмента даних.

Для створення EXE-програми необхідно виконати наступні команди:

**TASM /LA /ZI ім'я файлу.asm**

**TLINK /X /V ім'я файлу.obj**

**ім'я файлу.exe**

При створенні програми *.COM необхідно виконання двох умов:

- вхідний текст програми повинен мати відповідний формат з використанням мінімальної моделі пам'яті;
- необхідно після компоновки отримати виконуваний файл з розширенням *.COM.

При використанні пакета TASM при виклику компоновщика необхідно вказати ключ /T:

**TASM /Z /N ім'я файлу.asm**

Ключ **/z** дозволяє *виведення на екран рядків вихідного тексту програми*, в яких асемблер виявив помилки (без цього ключа пошук помилок довелося б проводити з лістингу трансляції).

**/n** *пригнічує виведення в лістинг таблицю ідентифікаторів* (переліку символічних позначень в програмі), за рахунок чого трохи зменшується інформативність лістингу, але скорочується його розмір.

**TLINK /X /T ім'я файлу.obj**

Ключ **/x** означає не створити файл карти *map*.файл.

Ключ **/t** означає створити виконуваний *com*.файл.

**ім'я файлу.com**

При використанні програм типу *.com при виклику асемблера ключ /ZI не використовується, аналогічно при виклику компоновщика ключ /V також не використовується. Ці ключі є неприйнятними для програм типу *.com.

## Шаблон 1 програми для com-файла

```
; Шаблон COM -програми.
; сегмент кода.
; Модель пам'яті tiny - код і дані розташовуються в одному
сегменті

.MODEL TINY
.CODE
ORG 100H
program:
;----

;----
RET

<Опис даних>

END program
```

## Шаблон 2 програми для com-файла

Типова структура COM-програми аналогічна структурі EXE-програми, з тією лише різницею, що COM-програма містить лише один сегмент – сегмент коду, який включає інструкції процесора, директиви і описи змінних.

```
;Визначення сегмента кода
CODE SEGMENT
ASSUME    CS:CODE,DS:CODE,SS:CODE
ORG   100H    ;Початок для COM-програми, резервування місця під
      стек
;Визначення підпрограми
PROC1     PROC
     . . .          ;Текст піпрограми
PROC1     ENDP
```

```
START:
     . . .          ;Текст програми
     MOV  AH,4CH    ;Оператори завершеня програми
     INT  21H
;===== Data =====
BUF  DB   6         ;Визначення змынної типу Byte
. . .               ;Визначення інших даних
CODE ENDS
END  START
```

## Шаблон 3 програми для сом-файла

```
     ; Виведення заголовка (TITLE) програми у другому рядку кожної
     ; сторінки лістингу
     TITLE     ASCOM     COM-ПРОГРАМА ПЕРЕМІЩЕННЯ ТА ДОДАВАННЯ
     ;Вирівнювання адреси сегмента кода на межу параграфа – адреса
     ;націло ділиться на 16 (10H)
     CODESEG   SEGMENT   PARA 'Code'
     ;Регістри CS, DS, SS, ES повинні містити адресу початку
     ;сегмента кода
          ASSUME    CS:CODESEG, DS:CODESEG, SS:CODESEG, ES:CODESEG
     ;Початок для COM-програми, резервування місця під стек
     ORG  100H
     ;Значення 100H розміщено в регістр IP, тому наступне речення
     ;після ORG повинно бути виконуваною командою
Begin:    JMP MAIN; JMP – команда переходу на мітку MAIN
;Якщо дані записати після інструкцій,
; то потреби у операторі JMP немає
;-------------------------------------------------------
DD1  DW   220       ; Визначення даних
DD2  DW   115
DD3  DW   ?
;-------------------------------------------------------------
MAIN      PROC      NEAR ;Процедура MAIN
          MOV       AX, DD1   ; В регістр AX переслати 220
          ADD       AX, DD2   ;Додати 115 до AX
          MOV       DD3, AX   ;Переслати суму в DD3
          MOV       AX, 4C00H ; Завершити роботу
          INT       21H
MAIN      ENDP      ;Кінець процедури MAIN
CODESEG   ENDS      ;Кінець сегмента кода
          END  BEGIN
```

## Шаблон 1 програми для exe-файла

```
Шаблон EXE -програми.
; Сегмент даних + сегмент кода.
; Модель пам’яті small - код розташовується в одному сегменті,
; а дані та стек – в іншому.

.MODEL SMALL
.DATA
```

```
.CODE
program:
MOV AX,@DATA ;Ініціалізація сегмента DS
MOV DS,AX    ;адресою сегмента даних
;----

;----
MOV AX, 4C00H
INT 21H
END program
```

Для невеликих EXE-програм з трьохсегментною структурою типова наступна структура:

**Шаблон 2 програми для exe-файла**

```
;Визначення сегмента стека
STAK      SEGMENT STACK
          DB   256 DUP (?)
STAK      ENDS
;Визначення сегмента даних
DATA      SEGMENT
SYMB      DB '#'    ;Опис змінної з іменем SYMB
                    ;типу Byte і зі значенням «#»
. . .               ;Визначення інших змінних
DATA      ENDS
;Визначення сегмента коду
CODE      SEGMENT
ASSUME    CS:CODE,DS:DATA,SS:STAK
;Регістр CS містить адресу початку сегмента коду CODE
;Регістр DS містить адресу початку сегмента даних DATA
;Регістр SS містить адресу початку сегмента стека STAK
;Визначення підпрограми
PROC1     PROC
     . . .          ;Текст підпрограми
PROC1     ENDP
START:              ;Точка входу в програму START
     XOR  AX,AX     ;Обнулення регістру AX
     MOV  BX, data  ;Обов'язкова ініціалізація
     MOV  DS,BX     ;регістра DS на початку програми
     CALL PROC1     ;Приклад виклику підпрограми
     . . .          ;Текст програми
     MOV  AX,4C00H  ;Оператори завершення програми
     INT  21H       ;Повернення у DOS
CODE      ENDS
END  START
```

```
; hello-1.asm
; Виводить на екран повідомлення "'Hello World!" і завершується
.model     small  ; модель пам’яті, яка використовується для EXE-файлів
.stack  100h      ; сегмент стеку розміром 256 байт
; У сегменті даних
.data
message db "Hello World! $'' ; рядок для виведення тексту та
                                символи управління
; У сегменті команд
.code    ; початок сегменту кода
start:
  mov ax, @data
  mov ds, ax
  mov ah, 09h ; номер функції DOS: виведення на екран - в АН
записується код функції
  mov dx,offset message ; зсув до змінної у сегменті даних  пишемо
в DX
  int 21h ; виклик системної функції DOS: вивести рядок на екран
  mov  ax, 4c00h  ; код успішного завершення програми та
вивантаження її з пам’яті
  int 21h
end  start ; кінець програми
```

```
; hello-1.asm
; Виводить на екран повідомлення "'Hello World!" і завершується
.model tiny  ; модель пам’яти, яка використовується для COM
.code    ; початок сегменту коду
org  100h ; початкове значення лічильник - 100h
start:  mov ah, 09h ; номер функції DOS: виведення на екран - в АН
записується код функції
mov dx, offset message ; зсув до змінної у сегменті даних  пишемо
в DX
int  21h ; виклик системної функції DOS: вивести рядок на екран
ret  ; завершення COM-програми, вихід в MS DOS
message db  "Hello World!",0Dh,0Ah,'$' ; рядок для виведення
                                        символи управління
; 0dh – Return (повернення  каретки), 0ah – Line feed (переведення
рядка)
End start   ; кінець програми
```