Міністерство освіти і науки України   
Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Кафедра автоматизації проектування енергетичних процесів і систем

Лабораторна робота №2 з дисципліни «Операційні системи»  
**Технологія роботи з системними засобами при створенні програм**

Виконав студент групи ТР-12

Каркушевський Владислав

Перевірила д.т.н., проф. Левченко Л. О.

КИЇВ 2022

**Мета роботи**: ознайомитися з технологією роботи створення асемблерної програми.

**Теоритичні відомості**

Асемблери зазвичай виконують два або більше проходів по тексту програми при трансляції. При першому проході асемблер проглядає усю вхідну програму та створює таблицю символів, яка містить імена та мітки, що зустрічаються в програмі. При другому проході асемблер використовує таблицю символів, в якій відома довжина кожної команди та її відносне розташування, а також формує об’єктний код для кожної інструкції. Після цього асемблер створює об’єктний файл (\*.obj), файл лістингу (\*.lst) та файл перехресних посилань (\*.crt).

Процес підготовки та відладки програми на мові Асемблер включає такі етапи: підготовка у текстовому редакторі Notepad файлу за шаблоном \*.asm, його трансляцію, компоновку, відладку програми за допомогою інтерактивного відладчика. Трансляція вхідного тексту програми полягає у перетворенні речень вхідної мови у коди машинних команд та виконується за допомогою програми TASM. В результаті трансляції створюється утворюється об’єктний файл з розширенням \*.obj.

Компоновка об’єктного файлу виконується за допомогою компонувальника (редактора зв’язків), який під’єднує до файлу основної програми файли з підпрограмами, встановлює зв’язки між ними та перетворює формат об’єктного файлу у виконуваний \*.exe, який завантажується у оперативну пам’ять та виконується

Для виконання трансляції використовується пакет TASM.EXE. В командному рядку ця програма запускається наступним чином: **TASM.EXE [ключі] ім’я\_вхідного\_файлу [, ім’я\_об’єктного\_файла] [, ім’я\_файла\_лістингу] [, ім’я\_файла\_перехресних\_посилань].**

Ключі – це режими роботи транслятора.

При запуску транслятора треба використовувати два ключа: /la - виведення розширеного варіанту лістингу транслятора, /zi – отримання повної інформації для відладчика. Наприклад, **tasm /la /zi pr.asm** Таким чином, результатом роботи транслятора є створення трьох модулів: **\*.lst** (лістингу), **\*.crf** (таблиці перехресних посилань: таблиці символічних імен змінних, які використовуються в програмі, та таблиці відносних посилань, в якій вказується у якому операторі визначено ім’я і де зустрічається), **\*.obj** (об’єктного), (Замість \* - ім’я\_вхідного\_файлу).

Запуск компоновщика (редактора зв’язку) здійснюється в командному рядку: **TLINK [ключі] список об’єктних модулів [, ім’я завантажувального модуля]**

Ключі можна подивитися просто запустивши tlink без параметрів.

При запуску компоновщика треба використовувати два ключа:

/х – не створювати файл з розширенням \*.map (подавляється формування файлу лістингу компоновки, в якому відображається карта завантаження, без цього файла можна обійтись)

/v – передає у завантажувальний файл символьну інформацію, яка дозволяє відладчику TD виводити на екран повний текст вхідної програми, включаючи мітки, коментарі та ін.

Список об’єктних модулів – це обов’язковий параметр, файли розділені пропуском або знаком плюс: tlink /х /v prog+prog1+prog2 або вказати повний шлях до цих файлів.

Для отримання виконуваного модуля треба запустити tlink /х /v pr.obj, в результаті буде отриманий модуль pr.exe

**Формат оператора в асемблері**. Структура асемблерного оператора відображає структуру відповідної машинної команди. Асемблерні оператори складаються з чотирьох полів:

**мітка операція(команда/директива) операнди ; коментар.**

**Директиви** – спеціальні інструкції - команди, які вказують самому асемблеру, як організовувати різні секції програми, де розташовуються дані, а де команди, які дозволяють створювати макровизначення, вибирати тип використовуваного процесора, організовувати зв'язки між процедурами і т.д.

За замовчуванням Асемблер **не розрізняє** великі і малі літери.

Директива **SEGMENT** починає новий сегмент, директива **ENDS** завершує його. Дозволяється починати новий сегмент, потім починати сегмент даних, потім переходити знову до текстового сегменту і т.д.

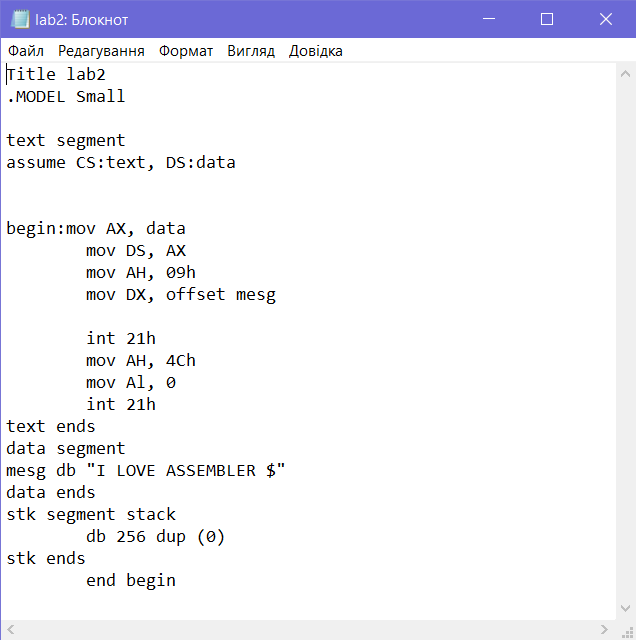
**Інші директиви**

Директива **ASSUME** повідомляє компілятору, що зазначений у ній сегмент необхідно зв’язати з іменем сегмента або групи. В якості сегментних регістрів застосовуються регістри CS, DS, ES, SS.

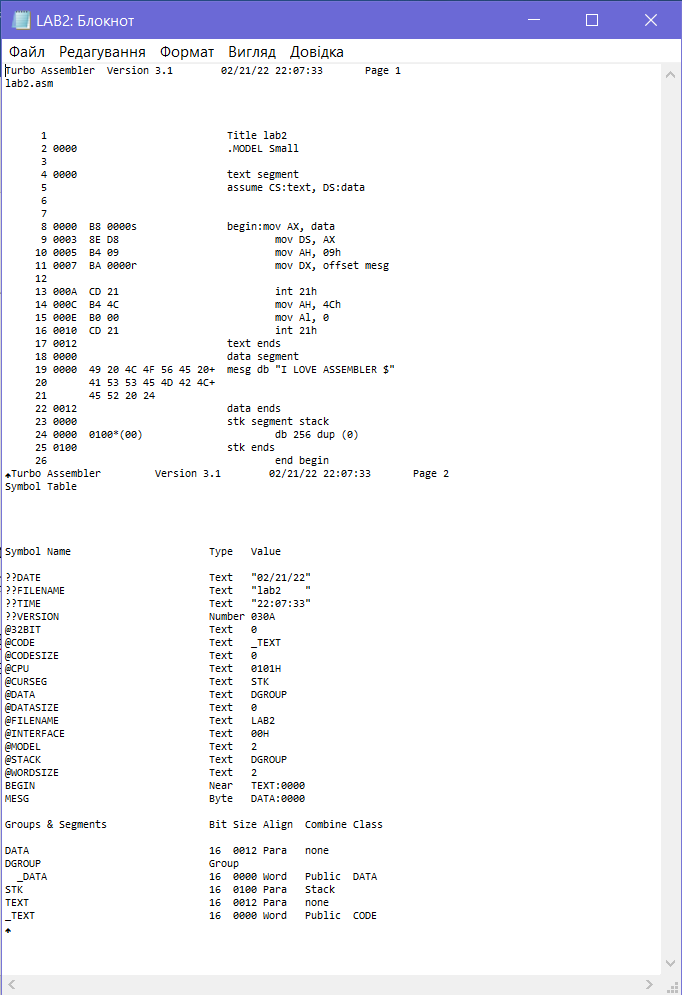
**MOV–** команда пересилання даних. Копіює вміст джерела до приймача, джерело не змінюється.

**Результати виконання роботи**

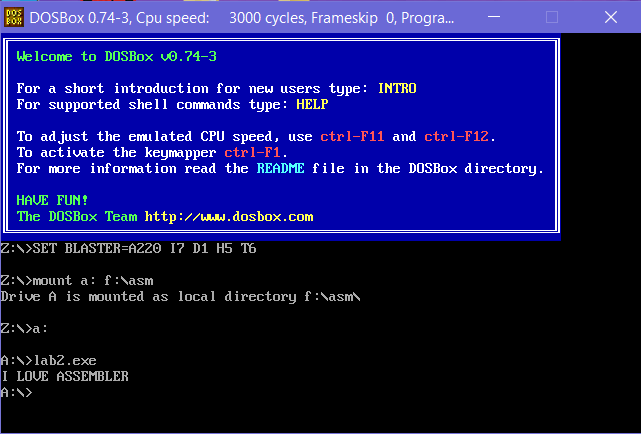
**1)** Створюємо код асемблерної програми в середовищі текстового редактора Блокнот.

****

**2)** Виконуємо трансляцію створеної програми

****

**3)** . Викликаємо компонувальника, отримуємо виконуваний модуль програми, та запускаємо виконуваний модуль на виконання

****

**Висновок:**

На цій лабораторній роботі ми ознайомитися з технологією роботи створення асемблерної програми. Вивчили формат оператора в асамблері, познайомилися з новими директивами. Створили код асемблерної програми у блокноті, виконали трансляцію, викликали компонувальника, отримали виконуваний модуль програми, та запустили виконуваний модуль на виконання. Результатом є виведений текст в консолі.