Лабораторна робота № 2

Технологія роботи з системними засобами при створенні програм

Мета роботи: ознайомитися з технологією роботи створення асемблерної програми.

Порядок роботи:

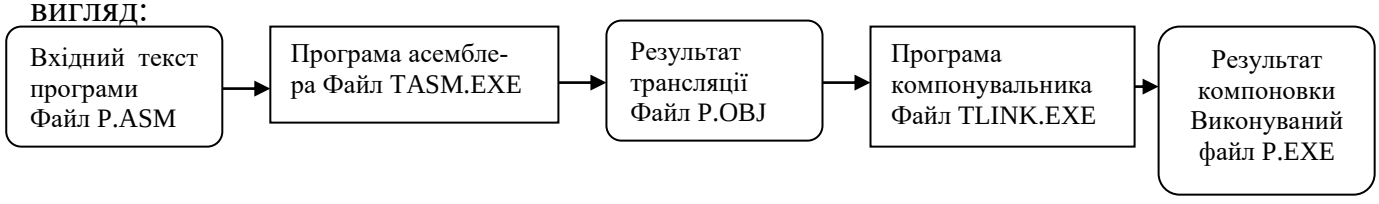
1. Створити код асемблерної програми в середовищі текстового редактора Блокнот.
2. Виконати трансляцію створеної програми.
3. Проаналізувати лістинг програми, визначити поля лістингу, віднайти відповідні сегменти програми, виправити помилки. У разі виявлення помилок повторити трансляцію програми та отримати об'єктний модуль програми.
4. Викликати компонувальника та отримати виконуваний модуль програми.
5. Запустити виконуваний модуль на виконання.
6. Скопіювати лістинг трансляції та результат роботи програми у командному рядку у лабораторну роботу.
7. Результати надати викладачу.

Теоретична частина

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. Процес підготовки програми до виконання має такий вигляд:



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 (програма-асемблер). 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].
Пам'ятати формат запуску в командному рядку асемблера tasm.exe необов'язково. Для отримання швидкої довідки про нього достатньо запустити tasm.exe без параметрів. В квадратних дужках позначені необов'язкові параметри. Обов'язковим є лише ім'я вхідного файла. Цей файл повинен знаходитися на диску, обов'язково мати розширення *.asm. За іменем вхідного файла через кому можна задати імена об'єктного файла, лістингу та перехресних посилань. Якщо ці імена не задати, то відповідно ці файли не будуть створені.
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).

Файл лістингу містить номер рядка тексту програми, код асемблера вхідної програми, а також розширену інформацію про цей код. Для кожної команди асемблера вказуються її машинний (об'єктний) код і зсув в кодовому сегменті.
Крім того, в кінці лістингу TASM формує таблиці з інформацією про мітки і сегменти, що використовуються в програмі. Якщо є помилки або сумнівні ділянки коду, то TASM включає в кінець лістингу повідомлення про них.
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:
/x – 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 /x /v prog+prog1+prog2** або вказати повний шлях до цих файлів.
Для отримання виконуваного модуля треба запустити **tlink /x /v pr.obj,** в результаті буде отриманий модуль **pr.exe**.

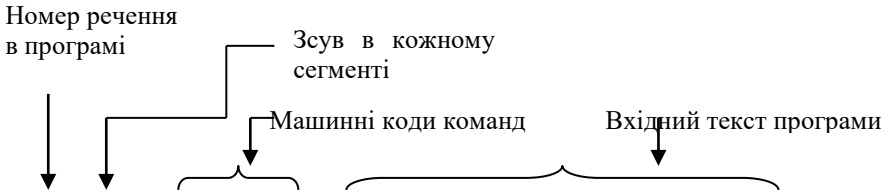
Виконання роботи

Створити файл:

```
text    segment                 ; Початок сегмента команд
assume  CS:text, DS:data        ;  Сегментний регістр CS вказує на сегмент
                                        команд
                                ; сегментний регістр DS - на сегмент даних
begin:mov   AX, data            ; Адресу сегмента спочатку завантажуємо в AX,
      mov   DS, AX              ;а потім переносимо з AX в DS
      mov   AH, 09h             ; Функція DOS 9h означає виведення на екран
      mov   DX, offset mesg;    Адреса повідомлення, що виводиться, повинно
                                        бути в DX
      int   21h                 ; Виклик DOS - функція переривання
      mov   AH, 4Ch             ; Функція 4Ch - завершення програми
      mov   Al, 0               ; Код 0 - код успішного завершення
      int   21h                 ; Виклик DOS
text  ends                      ;Кінець сегмента
data  segment                   ; Початок сегменту даних
mesg  db "NACHINAEM ! $"; Текст, що виводиться
data  ends                      ; Кінець сегменту даних
stk   segment stack             ; Початок сегменту стека
      db 256 dup (0)            ; Резервуємо 256 байт для стеку
stk   ends                      ; Кінець сегменту стека
      end begin                 ; Кінець тексту програми з точкою входу
```

Відтранслювати створений файл.

```
Turbo Assembler   Version 4.1    02/25/08 21:44:27          Page 1
Privet.asm
```




```
1  0000                 text    segment      ; Початок сегмента команд
2                       assume  CS:text, DS:data     ; Сегментний регістр CS
                                                     вказує на сегмент команд
3                                            ; сегментний регістр DS - на сегмент даних
4  0000  B8 0000s    begin: mov     AX, data   ; Адресу сегмента спочатку
                                                     завантажуємо в AX,
5  0003  8E D8                 mov     DS, AX  ;  а потім переносимо з AX в  DS
6  0005  B4 09                 mov     AH, 09h ;   Функція DOS 9h означає
                                                     виведення  на екран
7  0007  BA 0000r              mov     DX, offset mesg; Адреса повідомлення, що
```

```
                                                виводиться, повинно бути в DX
 8  000A  CD 21              int      21h ;      Виклик DOS – функція
                                                переривання
 9  000C  B4 4C              mov      AH, 4Ch ;  Функція 4Ch – завершення
                                                програми
10 000E  B0 00              mov      Al, 0   ; Код 0 - код успішного завершення
11 0010  CD 21              int      21h     ; Виклик DOS
12 0012                     text     ends      ; Кінець сегмента команд
```

Розмір в байтах сегменту команд

```
13 0000                     data     segment        ; Початок сегменту даних
```

Коди символів, що утворюють повідомлення

```
14 0000  4E 41 43 48 49 4E 41+      mesg     db     "NACHINAEM ! $"; Текст,
                                                    що виводиться
15          45 4D 20 21 20 24
16 000D                     data     ends      ; Кінець сегменту даних
```

Розмір в байтах сегменту даних

```
17 0000                     stk      segment stack   ; Початок сегменту стека
18 0000  0100*(00)          db       256 dup    (0)  ; Резервуємо 256 байт для
                                                     стеку
19 0100                     stk      ends       ; Кінець сегменту стека
```

Розмір в байтах сегменту стека

```
20                          end begin    ;      Кінець тексту програми з
                                                точкою входу
```

Turbo Assembler Version 4.1 02/25/08 21:44:27 Page 2
Symbol Table

Symbol Name	Type	Value
??DATE	Text	"02/25/08"
??FILENAME	Text	"Privet "
??TIME	Text	"21:44:27"
??VERSION	Number	040A
@CPU	Text	0101H
@CURSEG	Text	STK
@FILENAME	Text	PRIVET
@WORDSIZE	Text	2
BEGIN	Near	TEXT:0000

MESG Byte DATA:0000

Groups & Segments Bit Size Align Combine Class

DATA 16 000D Para none
STK 16 0100 Para Stack
TEXT 16 0012 Para none

Створити об’єктний файл та його виконати.

Вивчити структуру програми, розуміти що означає кожний оператор, розуміти зміст лістингу.