Лабораторна робота №1
Введення та виведення тексту.

Мета роботи

Ознайомитись з синтаксисом та структурою програми на мові асемблера. Навчитися використовувати функції операційної системи MS DOS для введення та виведення тексту. Дослідити різницю між файлами .COM та .EXE.

Хід роботи

1. Постановка задачі та розробка алгоритму.
Перша лабораторна робота повинна дати перше знайомство з новою для студентів мовою програмування - асемблером. Асемблер - це мова необмежених можливостей, це мова, команди якої повністю відповідають
командам мікропроцесора, це мова, що відкриває шлях до створення найоптимальніших програм, що використовують всі можливості сучасної обчислювальної машини. Але незвичний синтаксис, незвичний підхід до програмування обумовлюють важке сприйняття цієї мови програмування на початкових стадіях. Тому перша лабораторна робота є дуже простою. При її виконанні студенти повинні навчитися використовувати функції
MS DOS для введення та виведення тексту - найважливіші функції, що дають можливість програмі спілкуватися з користувачем. Алгоритм поставленої перед нами задачі простий:

1. Введення строки з клавіатури.
2. Виведення тексту на екран.

За цим простим алгоритмом студенти повинні побудувати блок-схему.

2. Написання програми.

Скориставшись будь-яким текстовим редактором, що не використовує службових символів, необхідно за складеним алгоритмом написати програму на мові асемблера. Для виконання введення та виведення тексту слід користуватися функціями MS DOS (переривання 21h) 0Ah та 09h відповідно. Необхідно заздалегідь за допомогою директив виділення пам'яті підготувати буфер для зберігання строки, що вводиться та виводиться.

Слід пам'ятати, що за умовою нашої роботи програму треба написати та відкомпілювати двічі: спершу як програму .COM, а згодом як програму .EXE. Наприкінці роботи студенти порівняють ці дві програми і зроблять висновки.

Після написання програм, вони компілюються за допомогою вибраного компілятора (рекомендується TASM).

3. Тестування програми. Аналіз результатів.
Після написання програми необхідно перевірити її працездатність. При програмуванні на низькому рівні найкраще це зробити за допомогою дебагера, наприклад Turbo Debugger'a, що постачається разом з пакетами Turbo Assembler та Turbo C. За допомогою Turbo Debugger'a ми можемо крок за кроком виконати нашу програму, побачити, як вона розташовується в оперативній пам'яті, які події відбуваються в обчислювальній машині, як змінюються регістри, що відбувається на екрані комп'ютера. Якщо при роботі програми були помічені помилки, треба повернутися до стадії написання програми. По закінченню написання та тестування програм, студент повинен отримати дві програми, програму .COM і програму .EXE, що працюють за одним і тим самим алгоритмом і виконують одну й ту саму функцію. Студенти повинні порівняти розміри отриманих програм, зробити висновки, чому розміри програм різні. За допомогою дебагера проаналізувати

різницю у виконанні операційною системою EXE та COM програм. Поміркувати над особливостями програм .COM та .EXE.
Визначити в яких ситуаціях доцільніше використовувати EXE-програми, в яких - COM.

4. Висновки.
Перша лабораторна робота - не лише ознайомлення з синтаксисом та особливостями програм на мові асемблера, це ще й створення своєрідного скелету для написання всіх інших лабораторних робіт, що студент повинен виконати впродовж вивчення курсу. Програма майже кожної лабораторної роботи буде містити в собі ті самі функції введення та виведення, що містить в собі програма з першої лабораторної роботи.

5. Контрольні запитання.
1) Яка різниця між програмами COM та EXE?
2) Особливості написання та компіляції програми COM.
3) Особливості написання та компіляції програми EXE.
4) Функції введення та виведення операційної системи MS DOS.
5) Структура буферу вводу для функції 0Ah переривання 21h.

Текст програми 1 (COM)

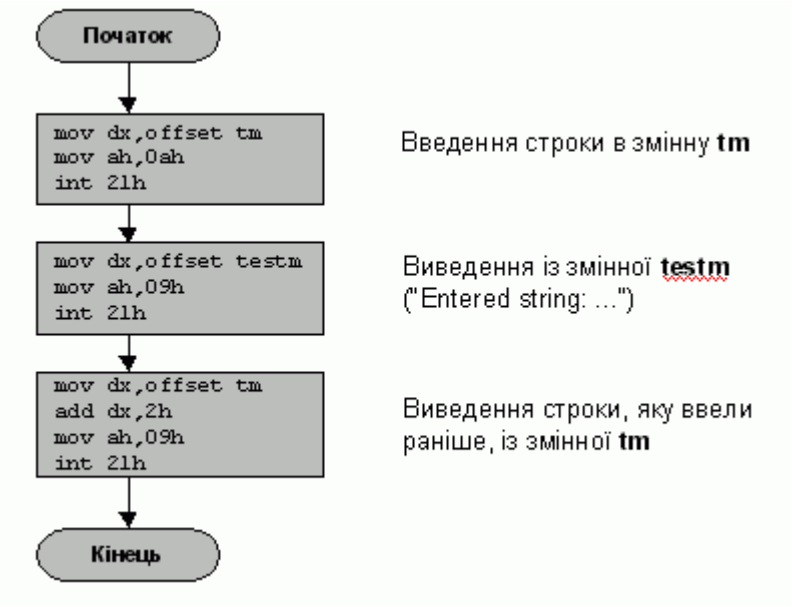
```
1     model tiny
2     .code
3     .startup
4
5           mov dx,offset tm
6           mov ah,0ah
7           int 21h
8           mov dx,offset testm
9           mov ah,09h
10          int 21h
11          mov dx,offset tm
12          add dx,2h
13          mov ah,09h
14          int 21h
15          ret
16
17    tm    db 255,255,255 dup("$")
18    testm db "Entered string: $"
19
20    End
```

Описання роботи програми
Як бачимо з блок-схеми, алгоритм є дуже простим і його можна умовно розбити на 3 частини.
Для цієї програми ми використовуємо 2 змінних: **testm** та **tm** (17-18). В змінну tm ми будемо вводити дані з клавіатури, а **testm** лише містить заданий текст.
Для написання програми достатньо знати, як використовувати дві функції DOS-івського переривання **21h**:
- функція **0ah** (введення з клавіатури)
- функція **09h** (виведення на екран з позиції курсору)

Вся програма побудована виключно на цих двох функціях і не потребує нічого, окрім ініціалізації цих самих функцій.

Працюють ці функції наступним чином:
1. В регістрі **DX** розміщується адрес зміщення змінної (в даному випадку – адрес першого символу строки). Це робиться за допомогою функції **offset** (5, 8, 11).
2. В регістрі **AH** розміщується порядковий номер потрібної функції (див. вище).
3. За допомогою оператора **int 21h** викликається DOS-івське переривання.

Побудувати програму з використанням цих функцій дуже легко.



Інші зауваження по коду програми:

- директива **model** (1) використовується компілятором. У випадку з **com**-файлами завжди використовується значення **tiny**, з **exe**-файлами будемо використовувати **tiny** або **small**, так як наші програми порівняно невеликі.

Модель	Кільк. сегментів коду, їх розмір	Кільк. сегментів даних, їх розмір
Small	1, <=64 Kb	1, <=64 Kb
Medium	Необмежено	1, <=64 Kb
Compact	1, <=64 Kb	Необмежено
Large	Необмежено	Необмежено
Huge	Необмежено	Необмежено

  - **Huge**, на відміну від **Large**, дозволяє створювати змінні розміром більше 64 Кб.
  - При використанні моделі **Tiny**, дані й код розміщуються в одному сегменті розміром до 64 Кб.

  Зауважимо, що більш великі моделі генерують 32-розрядні адреси, і тому потребують більше часу для виконання. Тому для таких невеликих програм, які ми будемо розглядати у лабораторних роботах, немає сенсу використовувати моделі більш великі, ніж **Small.**
- Символ "$" всередині строкових змінних є службовим символом і позначає кінець строки.
- (12) - Тут ми додаємо до адреси зміщення 2 байти, тому що у строках, введених з клавіатури, перші два байти використовуються для службової інформації, а за ними вже починається сама строка.
- Директива **.code** (2) визначає початок кодового сегменту.
- Директива **.startup** (3) автоматично генерує ініціалізацію сегментного регістру.
- Як бачимо, вся програма складається лише з одного кодового сегмента, так як маємо справу з **com**-файлом. Таким чином, **com**-файли більш зручні для менших за обсягом програм.

А чи знаєте ви, що…

…переривання – це операція, яка припиняє роботу програми, для того, щоб система могла виконати якісь дії. Наприклад, в даній роботі, при виклику функції **0ah** переривання **21h**, програма призупиниться, доки ми не введемо строку з клавіатури.

Текст програми 2 (EXE)

```
1     model small
2     .data
3     tm    db 255,255,255 dup("$")
4     testm db "Entered string: $"
5
```

```
6     .code
7     main  proc
8           mov ax,@data
9           mov ds,ax
10          mov dx,offset tm
11          mov ah,0ah
12          int 21h
13          mov dx,offset testm
14          mov ah,09h
15          int 21h
16          mov dx,offset tm
17          add dx,2h
18          mov ah,09h
19          int 21h
20          mov ax,4c00h
21          int 21h
22
23    main endp
24    end main
25    code ends
```

Як бачимо, **exe**-програма має декілька особливостей у написанні порівняно з **com**-програмою, але алгоритм залишається незмінним. Найголовніша відмінність полягає в тому, що ця програма складається з двох сегментів:

- сегменту даних, в якому зберігаються змінні та константи
- кодового сегменту, де зберігається сама програма

За допомогою директив **.data** та **.code** (2, 6), ми визначаємо відповідно сегменти даних та коду.

Отже, як бачимо, **exe**-файли більш зручні для написання більш серйозних за обсягом програм. В даному випадку, звісно, краще використовувати **com**-файли.

Додаткові завдання за варіантами:

0. Написати com-програму, яка виведе з введеного рядку на екран тільки непарні за порядком літери.
1. Написати exe-програму, яка виведе з введеного рядку на екран тільки непарні за порядком літери.
2. Написати com-програму, яка виведе з введеного рядку на екран тільки парні за порядком літери.
3. Написати exe-програму, яка виведе з введеного рядку на екран тільки парні за порядком літери.
4. Написати com-програму, яка запитає спочатку Ваше ім’я, після введення імені запитає прізвище, а тоді виведе: Hello, ВВЕДЕНЕ ПРІЗВИЩЕ ВВЕДЕНЕ ІМ’Я.
5. Написати exe-програму, яка запитає Ваше ім’я, після введення імені запитає прізвище, а тоді виведе: Hello, ВВЕДЕНЕ ПРІЗВИЩЕ ВВЕДЕНЕ ІМ’Я.
6. Написати com-програму, яка виведе на екран рядок **Hello, world!!!** стільки разів (кожна фраза починається з нового рядку), скільки дає сума ваших двох останніх цифр номеру залікової книжки (студентського квітку). Всі операції (додавання, передача значення) виконати на асемблері.
7. Написати exe-програму, яка виведе на екран рядок **Hello, world!!!** стільки разів (кожна фраза починається з нового рядку), скільки дає сума ваших двох останніх цифр номеру залікової книжки (студентського квітку). Всі операції (додавання, передача значення) виконати на асемблері.
8. Написати com-програму, яка виведе на екран заздалегідь записаний рядок (використовуємо змінні) у такому порядку. У першому рядку 1 літера з рядку, у другому 2, у третьому 3 і т.д. до останнього символа у рядку змінної.
9. Написати exe-програму, яка виведе на екран заздалегідь записаний рядок (використовуємо змінні) у такому порядку. У першому рядку 1 літера з рядку, у другому 2, у третьому 3 і т.д. до останнього символа у рядку змінної.