Лабораторна робота №2
Заміщення рядкових літер прописними.

Мета роботи

Ознайомитись з командами умовного те безумовного переходу, арифметичними діями в асемблері, отримати уяву про побудову циклів, поглибити знання функцій для введення і виведення тексту, закріпити навички роботи зі строковими величинами.

Хід роботи

1. Постановка задачі та розробка алгоритму.

Під час проведення цієї лабораторної роботи ми можемо використовувати елементи програми створеної під час виконання першої лабораторної роботи. Але тепер наше завдання ускладнюється. Нам потрібно не лише ввести і вивести текст, а й обробити його належним чином перед тим як вивести його на екран. Зокрема нам потрібно у введеному рядку замінити всі рядкові літери на прописні. Щоб це зробити, ми повинні проаналізувати кожен символ введеного рядку і якщо він потрапляє в діапазон "маленьких" літер, замінити його на відповідну "велику" літеру.
Для спрощення задачі будемо працювати з літерами лише англійського алфавіту. Уважно проглянувши таблицю ASCII, помітимо, що кожна "велика" літера англійського алфавіту має код на 20h менший за відповідну "маленьку" літеру. Тобто для заміщення рядкової літери на прописну нам досить відняти від значення її коду число 20h.

Алгоритм поставленої перед нами задачі буде наступний:
1. Вводимо рядок з клавіатури.
2. Встановлюємо вказівник на перший символ рядку.
3. Перевіряємо, чи потрапляє вказаний символ в діапазон "рядкових" літер.
4. Якщо попадає - віднімаємо від його коду значення 20h.
5. Якщо поточний символ - останній, переходимо до пункту 8.
6. Переміщаємо покажчик на наступний символ рядку.
7. Повертаємося до пункту 3.
8. Виведення тексту на екран.

За цим алгоритмом студенти повинні побудувати блок-схему, а згодом реалізувати програму на мові асемблера.

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 найкраще користуватися функціями MS DOS (переривання 21h) 0Ah та 09h відповідно. Щоб проаналізувати кожен символ рядку, доцільніше занести в регістр BX адресу першого символу, а згодом збільшувати значення BX на 1, що перейти до наступного символу. Для перевірки попадання символу в діапазон “маленьких” літер можемо скористуватись командою порівняння CMP і, виконуючи необхідні умовні переходи, замінити “маленьку” літеру на “велику”, або залишити її без змін. Слід зазначити, що оскільки ми працюємо не з числами, а з символами, потрібно використовувати лише ті команди умовних переходів, що оперують беззнаковими величинами. Для заміни “маленької” літери на “велику” досить скористуватися командою SUB і відняти від коду літери значення 20h.
Контролювати кінець рядку буде не складно, тому що при зчитуванні рядка ми отримуємо і кількість введених літер. Ми можемо виділити якийсь регістр процесор, наприклад CX, під лічильник, зменшувати його при переході до кожного наступного символу і, коли лічильник стане дорівнювати 0, завершити обробку рядка і вивести його на екран.
Після написання програм, вони компілюються за допомогою обраного компілятора (рекомендується TASM).

3. Тестування програми. Аналіз результатів.

Після написання програми необхідно, скориставшись дебагером перевірити її працездатність. Треба звернути увагу на правильність всіх умовних та безумовних переходів, прослідкувати, щоб не виникало нескінченних циклів, звернути увагу, чи правильно ідентифікуються "маленькі" літери, чи коректно працює заміна цих літер на "великі".
Також важливим моментом є правильна адресація рядка. Ми вносимо зміни в рядок, який розташовано в оперативній пам'яті. Треба слідкувати, щоб ми ні в якому разі не вийшли за межі нашого рядка, інакше є ймовірність, що ми можемо пошкодити данні інших програм або операційної системи.
Якщо при роботі програми були помічені помилки, треба повернутися до стадії написання програми.
По закінченню написання, компіляції та тестування програм, студент повинен отримати працюючу програму .COM. В нашому випадку нема потреби робити програму EXE, тому що ми не працюємо з великими об'ємами даних і код нашої програми також не великий. Але для тренування та збільшення навичок, можна внести незначні зміни в текст програми і відкомпілювати її як програму EXE.

4. Висновки.
Виконана лабораторна робота – це перша спроба за допомогою команд умовних та безумовних переходів створити програму зі складною логічною структурою. Також ця програма дозволяє закріпити навички по обробці текстових рядків і може мати практичне використання в майбутньому.

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

```
model tiny
.code
.startup
     mov dx, offset Sos ; встановлюємо вказівник на строку
     mov ah,0Ah
     int 21h ; введення строки

     mov cx,[bx + 1] ; встановлюємо лічильник
  lst: ; початок циклу
     mov di,cx
      cmp Sos[di+1],60h ; перевірка, чи літера маленька
      jl next ; якщо літера велика – йдемо далі
      sub Sos[di+1],20h ; якщо маленька – змінюємо її
   next:
      loop lst ; перехід на початок циклу
      mov dx, offset Sos + 2 ; кінець циклу, встановити вказівник на рядок
      mov ah,9
      int 21h ; виведення строки
      ret
   Sos     db 20 dup($) ; змінна-рядок
      end
```

Функції для введення та виведення на екран, переривання, та більшість операторів, які тут використовуються, ми детально розглянули в першій роботі.
Тому зосередимо увагу на нових моментах в цій роботі: а саме на **циклах** (loop) та **умовних переходах** (в нашому випадку jl – перехід, якщо менше). Нагадаємо що в інших мовах програмування ці функції реалізуються операторами відповідно **for, while** та **if … then … else**.

Розглянемо, як працюють команди loop та jl. Принцип їх роботи дуже простий.

Команда **loop** переходить на задану мітку, кожного разу зменшуючи значення лічильника (в якості котрого використовується регістр CX, в який ми заносимо необхідну кількість повторень). Цей цикл продовжується, доки значення лічильник не стане рівним нулю. **Зверніть увагу**, що на відміну від інших мов програмування, лічильник відраховує значення в зворотному порядку, до нуля, а не від нуля.
Зрозуміло, що в нашому випадку початкове значення лічильника буде дорівнювати кількості символів у введеному рядку.

Команда **jl** – переходить на задану мітку, якщо до цього функція порівняння **cmp** повернула ознаку "менше" (про це див. далі детальніше). В іншому випадку програма просто переходить на наступну команду.
Окрім цієї команди існують зокрема jg (перехід, якщо більше), je (якщо рівно), jne (якщо не рівно) та ін. Також існує команда безумовного переходу – **jmp.**

Використовуючи ці функції, блок-схему, та знання з першої роботи, можна легко написати програму.

А чи знаєте ви, що…

Регістри-прапорці (flag registers) використовуються в мові Асемблер для забезпечення умовних переходів. В цій роботі ми безпосередньо використовуємо регістр знаку останньої операції **SF**. В ньому зберігається знак останньої операції. Тобто після виконання команди cmp там буде "-", якщо значення буде менше, "+" – якщо більше.
Саме це значення використовує команда **jl**. Перехід, якщо менше, здійснюватиметься тоді, коли в регістрі SF буде знак "-".

Таблиця ASCII…

Нижче вказана **частина** таблиці ASCII, що містить символи з кодами від 20h до 7Fh.

```
   0 1 2 3 4 5 6 7 8 9 A B C D E F

2    ! " # $ % & ' ( ) * + , - . /
3  0 1 2 3 4 5 6 7 8 9 : ; < = > ?
4  @ A B C D E F G H I J K L M N O
5  P Q R S T U V W X Y Z [ \ ] ^ _
6  ` a b c d e f g h i j k l m n o
7  p q r s t u v w x y z { | } ~
```

Якщо ви помітили, всі маленькі літери починаються з символу 61h, причому різниця між відповідними великими та маленькими літерами = 20h. На цьому грунтується робота нашої програми.

Додаткові завдання за варіантами:

0. Написати com-програму, яка виведе на екран запитання **“Are you ОК?”** І представить на вибір відповіді:

Y) **ОК!**

N) **Bad.**

У разі введення **Y** чи **y** - видасться рядок – **“I am glad”**, **N** або **n** – рядок **“So pitty...”,** будь-яка інша літера чи цифра **– “Do not worry. Be happy”**

1. Написати exe-програму, яка виведе на екран запитання **“How are you?”** І представить на вибір відповіді:

1) **ОК!**

2) **Bad.**

У разі введення **Y** чи **y** - видасться рядок – **“I am glad”**, **N** або **n** – рядок **“So pitty...”,** будь-яка інша літера чи цифра **– “Do not worry. Be happy”**

2. Написати com-програму, яка виведе на екран запитання **“Input string”** . Після введяння рядку необхідно всі прописні літери перетворити на рядкові.
3. Написати exe-програму, яка виведе на екран запитання **“Input string”** . Після введяння рядку необхідно всі прописні літери перетворити на рядкові.
4. Написати com-програму, яка виведе на екран запитання **“Input string”** . Після введяння рядку необхідно всі прописні літери перетворити на рядкові, а рядкові на прописні.
5. Написати exe-програму, яка виведе на екран запитання **“Input string”** . Після введяння рядку необхідно всі прописні літери перетворити на рядкові, а рядкові на прописні.
6. Написати com-програму, яка виведе на екран запитання **“Input string”** . Після введяння рядку необхідно перетворити першу і останню літери рядку на рядкові, а всі інші рядкові на прописні.
7. Написати exe-програму, яка виведе на екран запитання **“Input string”** . Після введяння рядку необхідно перетворити першу і останню літери рядку на рядкові, а всі інші рядкові на прописні.
8. Написати com-програму, яка виведе на екран запитання **“Input string”** . Після введяння рядку необхідно перетворити першу і останню літери рядку на прописні, а всі інші прописні на рядкові.
9. Написати exe-програму, яка виведе на екран запитання **“Input string”** . Після введяння рядку необхідно перетворити першу і останню літери рядку на прописні, а всі інші прописні на рядкові.