**Київський національний університет імені Тараса Шевченка**

**факультет радіофізики, електроніки та комп’ютерних систем**

**Комп’ютерні системи**

Лабораторна робота № 4

Тема: «Дослідження збірок та автоматизація збірок, написання програм на асемблері»

Роботу виконав

студент 3 курсу

спеціальності “КІ-СА”

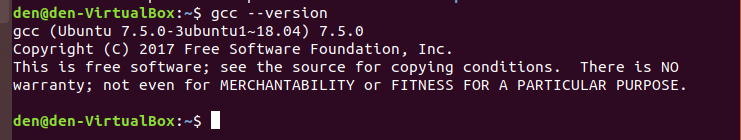
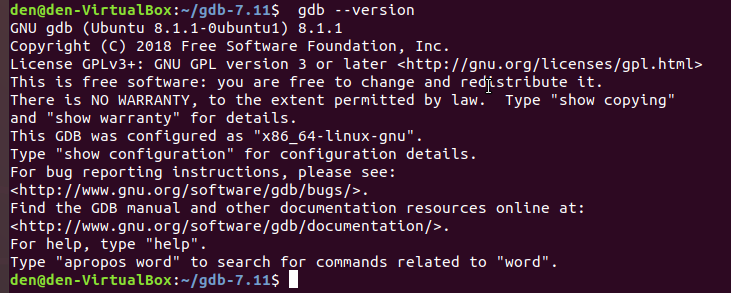
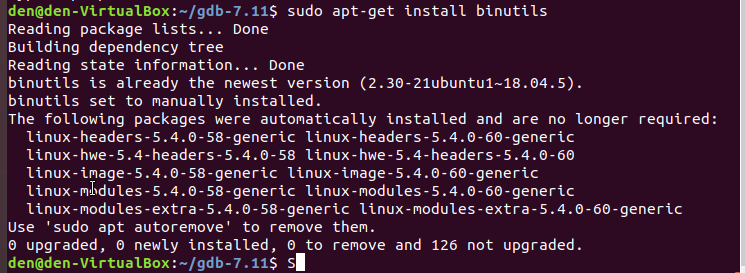
Ситниченко Денис Вікторович

Київ 2021

**Хід виконання роботи:**

## **Підготовка середовища розробки**

Для виконання лабораторної роботи вам знадобиться комп'ютер (віртуальний або фізичний) архітектури AMD64/EM64T із встановленим дистрибутивом ОС Linux (будь-яким).

На систему необхідно встановити GCC, GDB, GNU Make та GNU Binutils.

Створіть окремий каталог, який будете використовувати для виконання лабораторної роботи.



Завантажте в нього файл із символами та програму-заготовку:

* [defs.h](http://tilde.slu.kiev.ua/cs/asm/defs.h)
* [exit.s](http://tilde.slu.kiev.ua/cs/asm/exit.s)

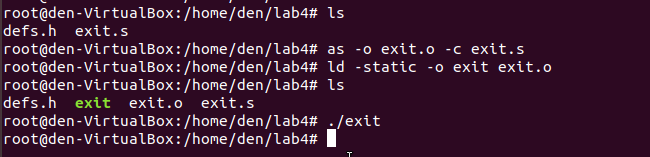




Виконайте асемблювання програми-заготовки та зв'язування:

* as -o exit.o -c exit.s
* ld -static -o exit exit.o

Пересвідчіться у тому, що виконуваний файл працездатний. Програма повинна нічого не робити і не виводити жодних помилок.



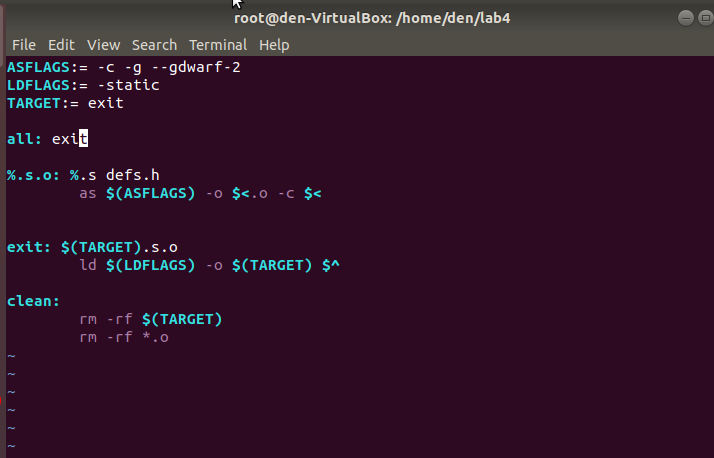
## **Автоматизація збірки**

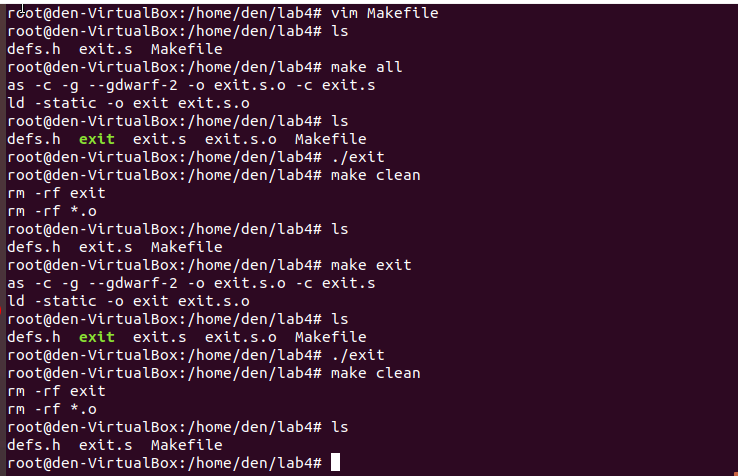
Створіть Makefile, який за командою make exit та make all виконає збірку, а за командою make clean очистить об'єктні та виконувані файли.

Модифікуйте Makefile так, щоб опції асемблера та лінкера задавалися змінними ASFLAGS та LDFLAGS.

Додайте опцію асемблера для генерації відлагоджувальних символів DWARF.

Використайте шаблонні правила так, щоб можна було збирати декілька асемблерних файлів в окремі виконувані файли. Це знадобиться при виконанні індивідуального завдання.



Демонстрація роботи:

## **Навички відлагоджування**

Завантажте одержаний виконуваний файл у відлагоджувач за допомогою команди:

gdb ./exit

Встановіть точку зупинки на початок програми (мітка \_start):

b \_start

Запустіть програму

run

Після зупинки виконання програми перегляньте вміст регістрів:

info registers або i r

Переходьте до виконання наступної команди:

next або n

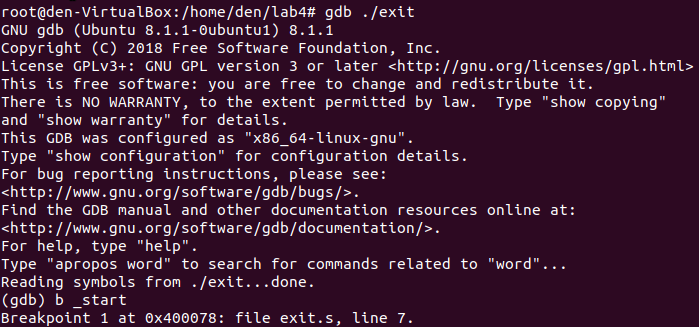
Для виходу із режиму покрокового виконання використовуйте команду

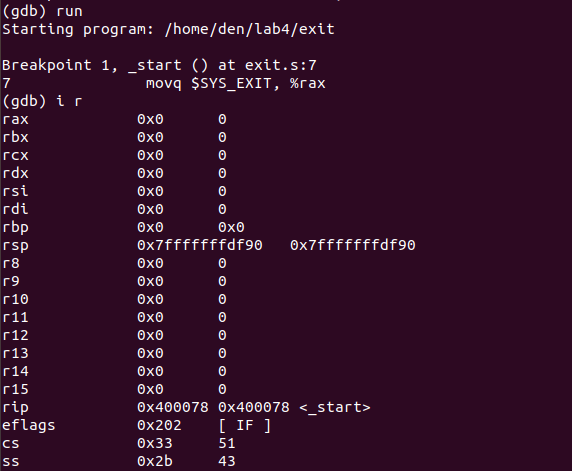
continue або с

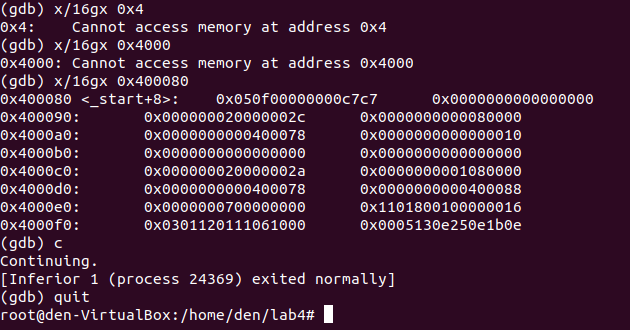
Програма працюватиме до наступної точки зупинки або до повного чи аварійного завершення.

Для перегляду адресного простору процесу скористайтесь командою x, наприклад:

x/16gx 0x12345678







## **Індивідуальні завдання**



## **Правда про своє оточення**

Створіть програму, яка виводить вміст змінних оточення власного процесу на стандартний потік виведення.

## **Додаткова довідка**

Розміщення параметрів командного рядка та змінних оточення на стеку

* 0(%rsp)  - argc
* 8(%rsp)  - argv[0] - name of executable
* ...  - argc-1 arguments
* NULL - end of arguments
* ...  - envp[0] - environment
* …
* NULL - end of environment

## **Псевдокод**

int len;

byte \*p;

const char \*newline = "\n";

main() {

    int argc = \*(%rsp)

    char \*\*envp = %rsp + 8 \* (argc + 2);

    while(envp != NULL) {

        len = 0;

        p = \*envp;

        while(\*p != '\0') {

            p++;

            len++;

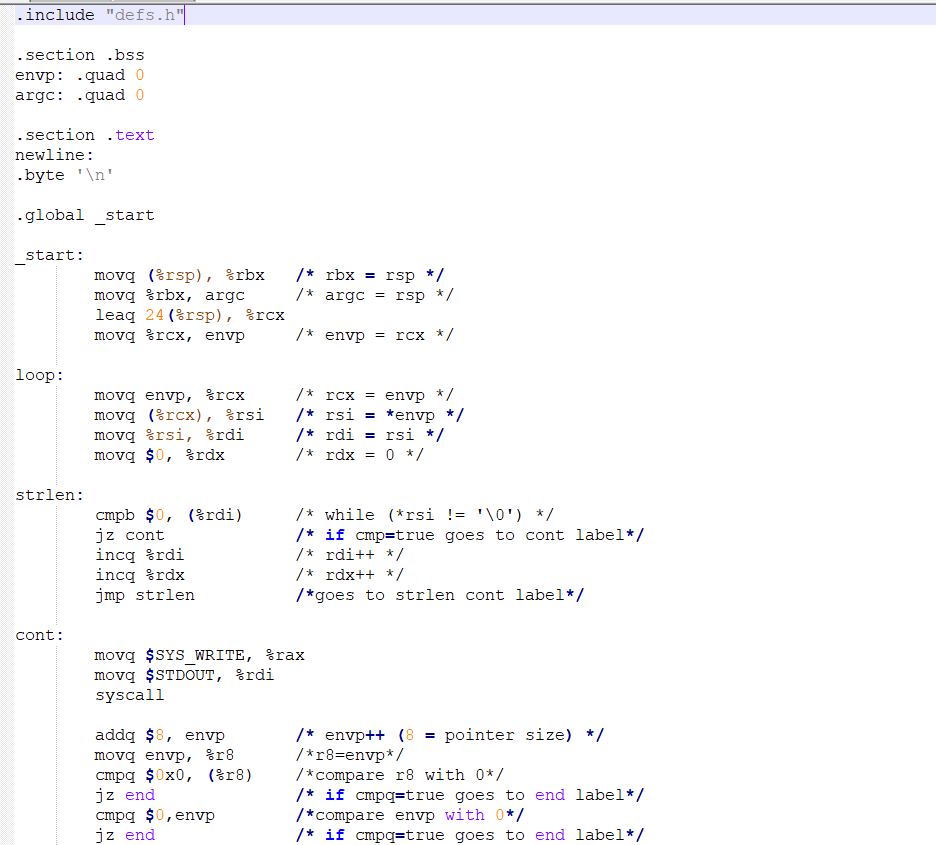
        }

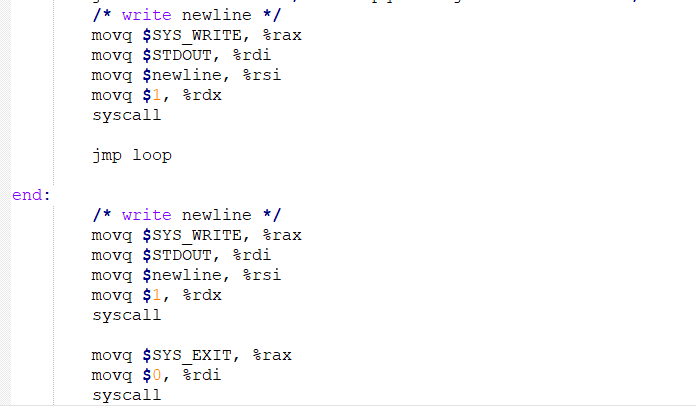
        write(stdout, len, \*envp);

        write(stdout, 1, newline);

    }

}

**Код програми:**

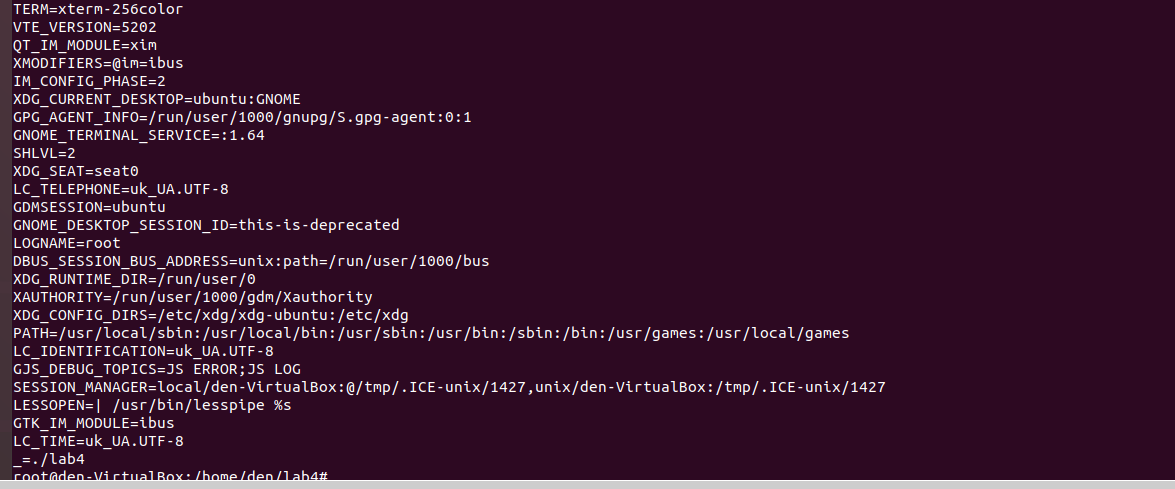
****

## **Перевірка**

Порівняйте результат виконання із результатом команди env.

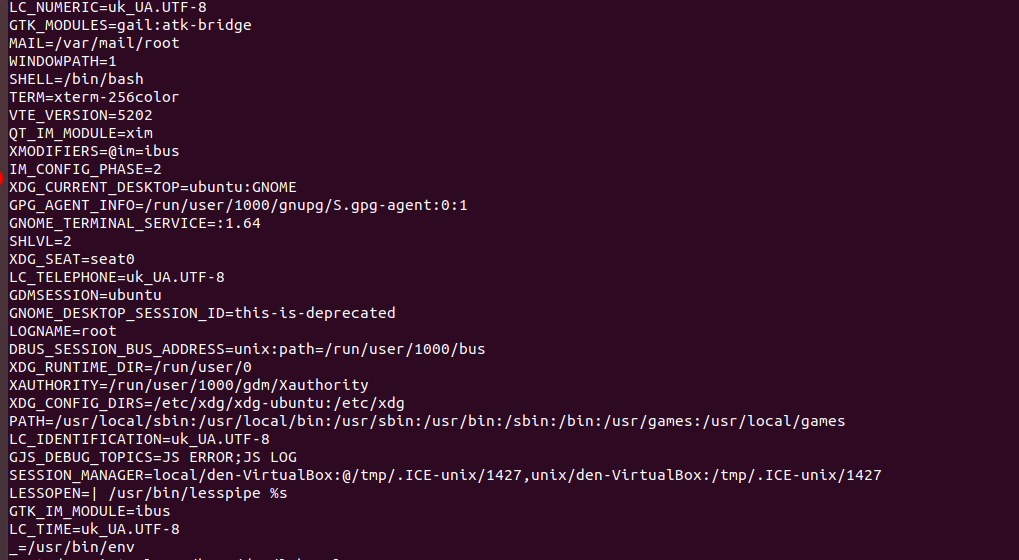
Вивід програми:





Вивід env:





Як видно зі скріншотів, вивід ідентичний

## **Висновок:** було отриманно навички написання програм на х64 асемблері та автоматизації створення збірок за допомогою Makefile.