

**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА  
ШЕВЧЕНКА ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА  
КОМП'ЮТЕРНИХ СИСТЕМ**

**Звіт з лабораторної роботи №4**

Роботу виконав:  
студент 3-го курсу  
напряму підготовки: «Комп'ютерна інженерія»  
спеціалізація: «Системний адміністратор»  
Костюченко Данило Олександрович

## Хід роботи

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

```
lagranje:~/workspace $ uname -m  
x86_64
```

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

Встановлено

Створіть окремий каталог, який будете використовувати для виконання лабораторної роботи

```
lagranje:~/workspace $ mkdir lab4  
lagranje:~/workspace $ cd lab4  
lagranje:~/workspace/lab4 $ wget http://tilde.slu.kiev.ua/cs/asm/defs.h  
lagranje:~/workspace/lab4 $ wget 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.

```
lagranje:~/workspace/lab4 $ as -o exit.o -c exit.s  
lagranje:~/workspace/lab4 $ ld -static -o exit exit.o  
lagranje:~/workspace/lab4 $ ./exit  
lagranje:~/workspace/lab4 $
```

## 2. 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.

Код Makefile:

Демонстрація роботи:

```
[M] /README.m ×  makefile × +
1 ASFLAGS = -c -g --gdwarf-2
2 LDFLAGS = -static
3 DEPS = defs.h
4
5 all:exit
6 exit.o: exit.s
7     as $(ASFLAGS) -o exit.o -c exit.s
8 exit: exit.o
9     ld $(LDFLAGS) -o exit exit.o
10 .PHONY: clean
11
12 clean:
13     rm exit *.o

lagranje:~/workspace/lab4 $ make exit
ld -static -o exit exit.o
lagranje:~/workspace/lab4 $ ls
defs.h  exit*  exit.o  exit.s  makefile
lagranje:~/workspace/lab4 $ ./exit
lagranje:~/workspace/lab4 $ make clean
rm exit *.o
lagranje:~/workspace/lab4 $ ls
defs.h  exit.s  makefile
lagranje:~/workspace/lab4 $ make all
as -c -g --gdwarf-2 -o exit.o -c exit.s
ld -static -o exit exit.o
lagranje:~/workspace/lab4 $ ./exit
lagranje:~/workspace/lab4 $ ls
defs.h  exit*  exit.o  exit.s  makefile
lagranje:~/workspace/lab4 $ █
```

**3. Навички відлагоджування** Завантажте одержаний виконуваний файл у відлагоджувач за допомогою команди: `gdb ./exit` Встановіть точку зупинки на початок програми (мітка `_start`): `run` 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 `c` Програма працюватиме до наступної точки зупинки або до повного чи аварійного завершення. Для перегляду адресного простору процесу скористайтесь командою `x`, наприклад: `x/16gx 0x12345678`

```
lagranje:~/workspace/lab4 $ gdb ./exit
GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.3) 7.7.1
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./exit...done.
```

```
(gdb) break _start
```

```
Breakpoint 1 at 0x400078: file exit.s, line 7.
```

```
(gdb) run
```

```
Starting program: /home/ubuntu/workspace/lab4/exit
```

```
Breakpoint 1, _start () at exit.s:7
```

```
7          movq $SYS_EXIT, %rax
```

```
(gdb) info registers
```

rax	0x0	0
rbx	0x0	0
rcx	0x0	0
rdx	0x0	0
rsi	0x0	0
rdi	0x0	0
rbp	0x0	0x0
rsp	0x7fffffffef200	0x7fffffffef200
r8	0x0	0
r9	0x0	0
r10	0x0	0
r11	0x0	0
r12	0x0	0
r13	0x0	0
r14	0x0	0
r15	0x0	0
rip	0x400078	0x400078 <_start>
eflags	0x202	[ IF ]
cs	0x33	51

```
---Type <return> to continue, or q <return> to quit---
```

```

(gdb) x/16gx 0x400078
0x400078 <_start>:      0x480000003cc0c748      0x050f00000000c7c7
0x400088:      0x0000000000000000      0x000000020000002c
0x400098:      0x0000000000008000      0x0000000000400078
0x4000a8:      0x0000000000000010      0x0000000000000000
0x4000b8:      0x0000000000000000      0x000000020000004d
0x4000c8:      0x0000000001080000      0x0000000000400078
0x4000d8:      0x0000000000400088      0x2f00732e74697865
0x4000e8:      0x7562752f656d6f68      0x6b726f772f75746e
(gdb) █

```

### 3. Індивідуальні завдання

#### *Потоковий шифр Цезаря*

Створіть програму, яка побайтово читає стандартний потік введення, додає до значення байту число 13 (із переповненням) та записує у стандартний потік виведення.

Псевдокод

```

byte b;
main() {
    while(read(stdin, 1, &b) > 0) {
        b += 13;
        write(stdout, 1, &b);
    }
    exit(0);
}

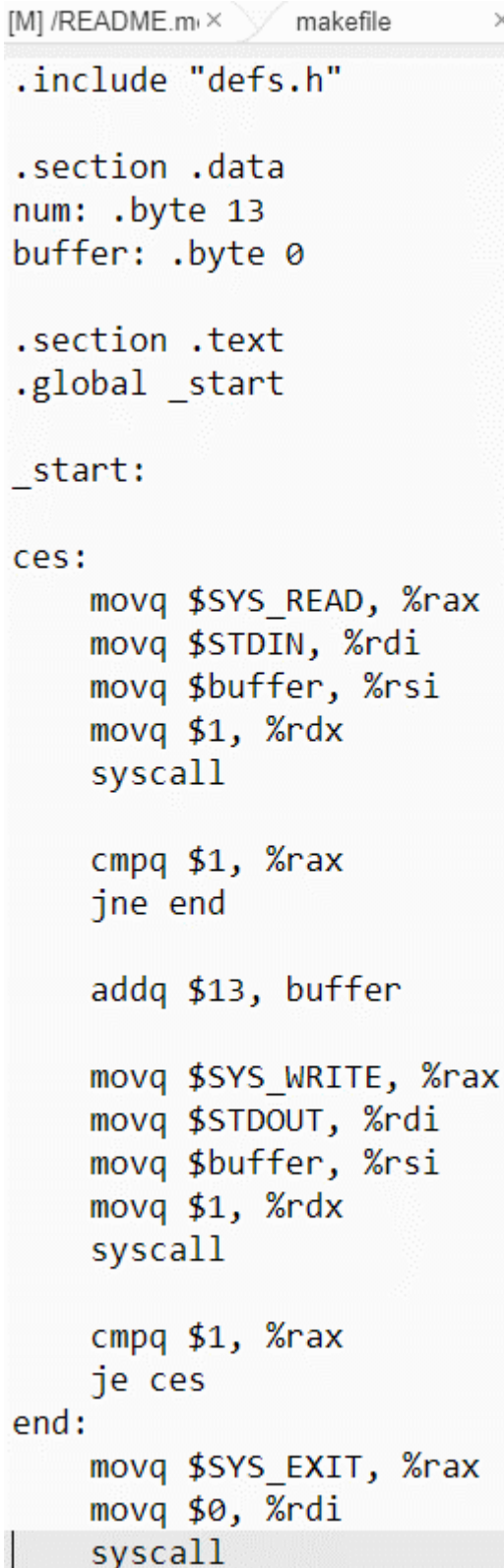
```

## Перевірка

Продемонструйте результат обробки вашою програмою такого тексту:

;X\_\_bJbe\_W

Код ceasar.s:



```
.include "defs.h"

.section .data
num: .byte 13
buffer: .byte 0

.section .text
.global _start

_start:

ces:
    movq $SYS_READ, %rax
    movq $STDIN, %rdi
    movq $buffer, %rsi
    movq $1, %rdx
    syscall

    cmpq $1, %rax
    jne end

    addq $13, buffer

    movq $SYS_WRITE, %rax
    movq $STDOUT, %rdi
    movq $buffer, %rsi
    movq $1, %rdx
    syscall

    cmpq $1, %rax
    je ces
end:
    movq $SYS_EXIT, %rax
    movq $0, %rdi
    syscall
```

Демонстрація програми:

```
lagranje:~/workspace/lab4 $ as -o ceasar.o -g -c ceasar.s
lagranje:~/workspace/lab4 $ ld -static -g -o ceasar ceasar.o
lagranje:~/workspace/lab4 $ ./ceasar
;X__bJbe_W
HelloWorld
```

**Висновок:** в даній лабораторній роботі я ознайомився з Асемблером та навчився створювати Makefile. Під час написання програм на Асемблері було помітно, що я працював на дуже низькому рівні абстракції для якого є свої плюси та мінуси.

**Плюси:**

1. Можливість писати швидкий у виконанні та високооптимізований код.
2. Усвідомлення базових парадигм програмування.

**Мінуси:**

1. Складність розробки сучасних програм.
2. Високі часові витрати на розробку.