Київський національний університет імені Тараса Шевченка
радіофізичний факультет

Лабораторна робота № 4

Тема: Асемблер

Роботу виконав
студент 3 курсу
Комп’ютерної інженерії
Веремій Юрій

Київ 2019

Силка на git https://github.com/uayura/koputer_sistem/tree/master/lab4

Хід виконання:

1. Підготовка середовища розробки

```
[root@g00-s00 lab4]# http://tilde.slu.kiev.ua/cs/asm/exit.s^C
[root@g00-s00 lab4]# as -o exit.o -c exit.s
[root@g00-s00 lab4]# ld -static -o exit exit.o
[root@g00-s00 lab4]# ls
defs.h  exit  exit.o  exit.s
[root@g00-s00 lab4]# ./exit
[root@g00-s00 lab4]#
```

2. Автоматизація збірки

y@g00-s00:/home/y/lab4

```
OBJ = env_my.s

wildcards := *.s *.as *.asm

SRCS := $(basename$(wildcard$(wildcards)))
OBJS := $(SRCS:.*=.o)

ASFLAGS = -c -g --gwarf-2
LDFLAGS = -static

lab: $(OBJ)
        nasm -f elf64 $(OBJ)
        ld $(LDFLAGS) -o lab $(OBJ).0

all: $(SRCS)

$(SRCS): $(OBJS)
        as $(ASFLAGS) -o $(OBJS) -c $(SRCS)
        ld $(LDFLAGS) -o lab $(OBJS)

.PHONY: all clean help

clean:
        rm -f $(SRCS) $(OBJS)

help:
        @echo 'do stuff'


~
~
~
"Makefile" 29L, 444C
```

```
[root@g00-s00 task3]# ls
env_my.asm  Makefile
[root@g00-s00 task3]# make
nasm -f elf64 -o  env_my.o env_my.asm
ld -static -o env_my env_my.o
[root@g00-s00 task3]# ls
env_my  env_my.asm  env_my.o  Makefile
[root@g00-s00 task3]#
```

3. Навички відлагоджування

Завантажте одержаний виконуваний файл у відлагоджувач за допомогою команди:

Встановіть точку зупинки на початок програми (мітка _start):

Запустіть програму

Після зупинки виконання програми перегляньте вміст регістрів:

```
[root@g00-s00 lab4]# as -o exit.o -c exit.s
[root@g00-s00 lab4]# ld -static -o exit exit.o
[root@g00-s00 lab4]# ls
defs.h  env_my.asm  exit  exit.o  exit.s  makefile  Makefile  task3
[root@g00-s00 lab4]# gdb ./exit
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-114.el7
Copyright (C) 2013 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-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/y/lab4/exit...(no debugging symbols found)...done.
(gdb) b _start
Breakpoint 1 at 0x400078
(gdb) run
Starting program: /home/y/lab4/./exit

Breakpoint 1, 0x0000000000400078 in _start ()
(gdb) i r
rax            0x0      0
rbx            0x0      0
rcx            0x0      0
rdx            0x0      0
rsi            0x0      0
rdi            0x0      0
rbp            0x0      0x0
rsp            0x7fffffffe5b0   0x7fffffffe5b0
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
ss             0x2b     43
ds             0x0      0
es             0x0      0
fs             0x0      0
gs             0x0      0
(gdb)
```

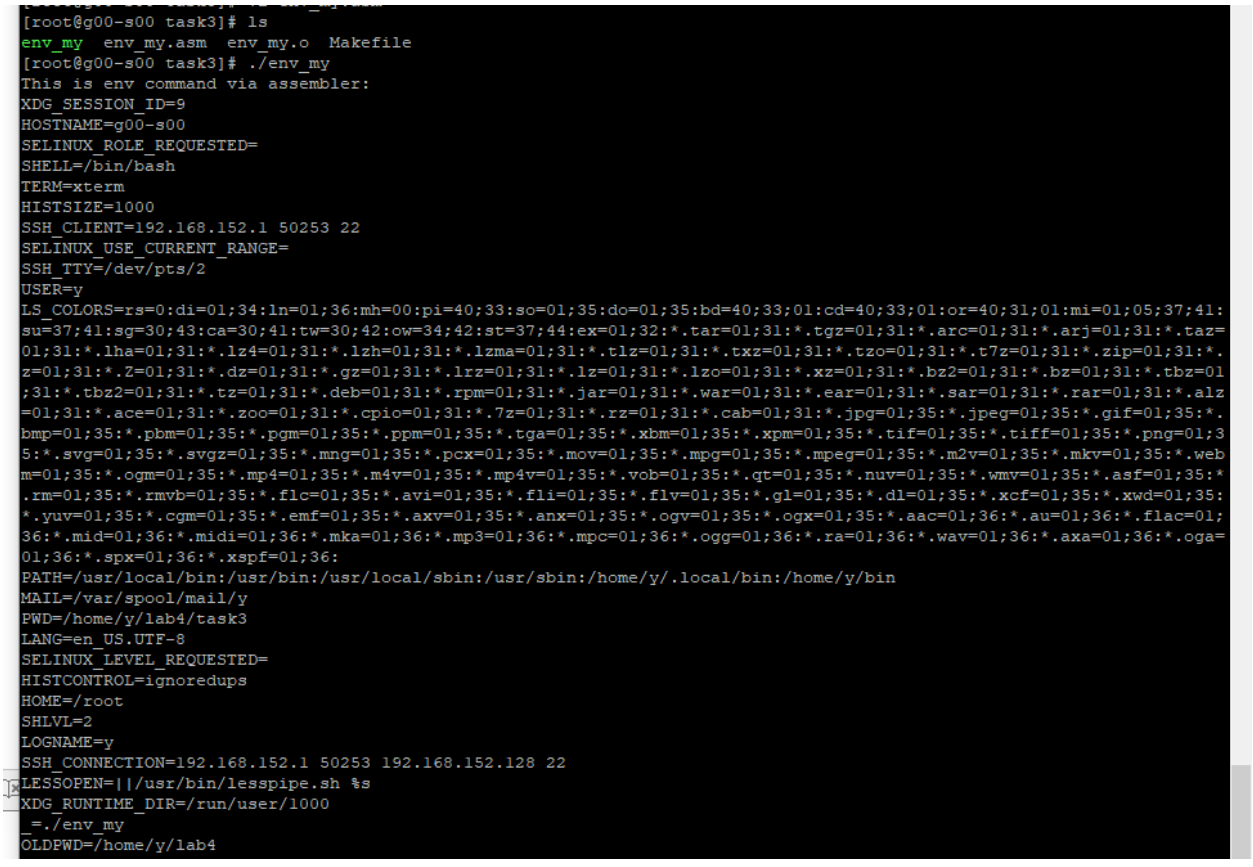
4. Індивідуальне завдання

Код:

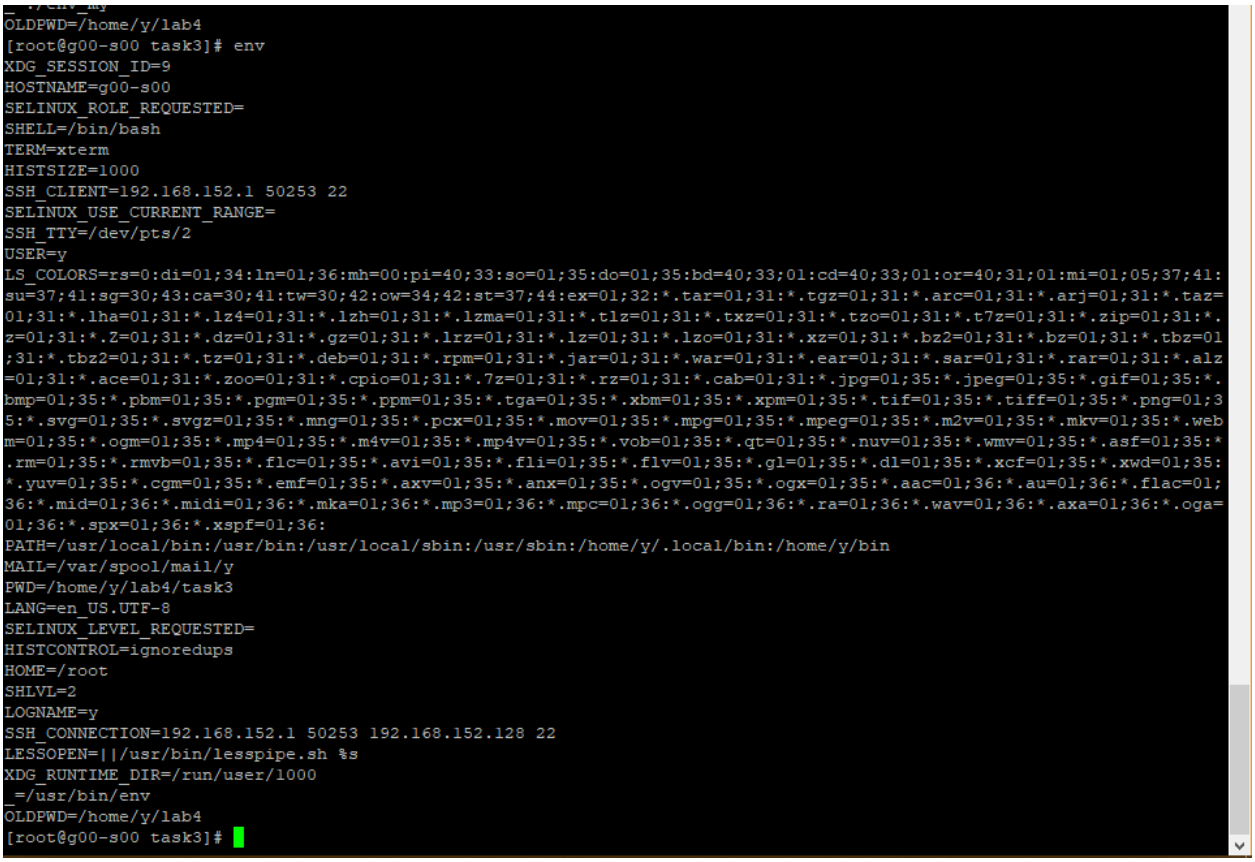
```
;macro for
printing
text.
example:
'print info,
info_length'
                %macro print 2
mov rax, SYS_WRITE
mov rdi, STDOUT
mov rsi, %1      ;first argument (info)
mov rdx, %2      ;second argument {info_length)
syscall
                %endmacro

                section .data
;defined Linux System Calls for x64
%define SYS_WRITE 1
%define STDOUT 1
%define SYS_EXIT 60

newline db 10, 0 ;newline implementation in NASM
nl_len: equ $-newline       ;length of new line

msg db "This is env command via assembler: ", 10, 0
                ;message with new line
msg_len: equ $-msg    ;calculated length of message

                section .bss
envp: resq 1    ;variable for strings of env command

                section .text
global _start
_start:
print msg, msg_len

mov rbx, qword [rsp]  ; argc = *(%rsp)
lea rcx, [rsp + rbx*8 + 16] ;needed offset of cmd args
;rcx = %rsp +8 * (argc + 2)
mov qword [envp], rcx ; **envp = rcx

loop: ;while (envp != NULL)
mov rcx, [envp]         ;temp var for transfering value
mov rsi, qword [rcx]  ;p = *envp
mov rdi, rsi          ;temp for output
xor rdx, rdx          ;len = 0

;loop to count length of each row of environmnet array
string_count:         ;while (*p != '\0')
cmp byte [rdi], 0     ;check of LSB
je output
inc rdi               ;p++
```

```
        inc rdx                 ;len++
        jmp string_count

        output: ;printing string (%rdi is p, %rdx is length)
        mov rax, SYS_WRITE
        mov rdi, STDOUT
        syscall

        add qword [envp], 8     ;offset to next element of env
        mov r8, [envp]          ;temp var for check
        cmp qword [r8], 0
        je end
        cmp qword [envp], 0
        je end

        print newline, nl_len
        jmp loop

        end: ;exit from program
        print newline, nl_len
        mov rax, SYS_EXIT
        xor rdi, rdi
        syscall
```

Виконання:



Порівняння з оригінальною командою:



Висновок: У результаті виконання даної лабораторної роботи було проведено ознайомлення з мовою програмування Асемблер, а також було створено програму на даній мові у відповідності до варіанту.