

Лаборатору Отчет No7

ДЭВИД МАЙКЛ ФРАНСИС

Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

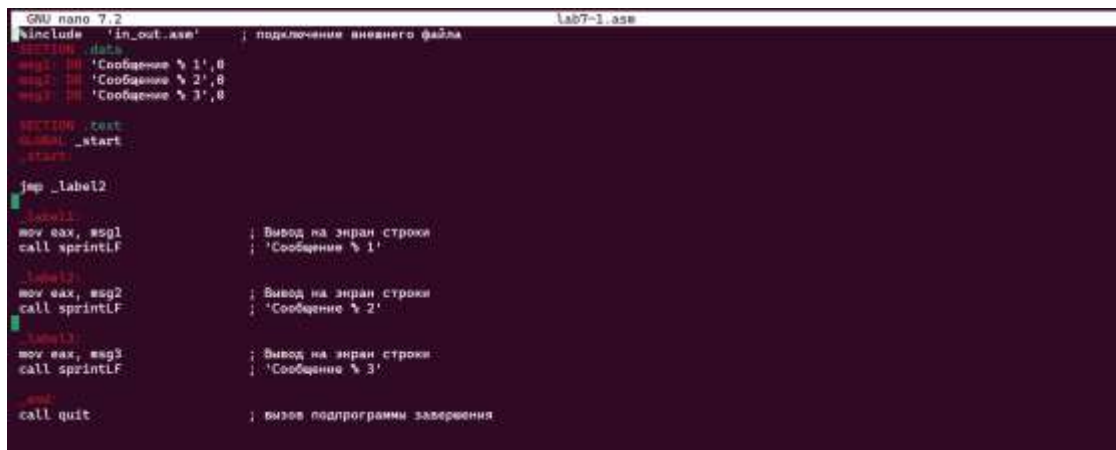
Выполнение лабораторной работы

Реализация переходов в NASM

С помощью утилиты mkdir создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №7. Перехожу в созданный каталог с помощью утилиты cd.

С помощью утилиты touch создаю файл lab7-1.asm

Открываю созданный файл lab7-1.asm, вставляю в него программу вывода значения регистра eax



```
GNU nano 2.2.8 lab7-1.asm
#include "in_out.asm" ; подключение внешнего файла

section .data
msg1 db 'Сообщение %1',8
msg2 db 'Сообщение %2',8
msg3 db 'Сообщение %3',8

section .text
global _start
_start:
;
jmp _label2

_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение %1'

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение %2'

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение %3'

quit:
call quit ; вызов подпрограммы завершения
```

Screenshot1

Создаю исполняемый файл программы и запускаю его.



```
(base) m1@DESKTOP-14ENTC01:~/Desktop/asm/asm7/asm7 $ nasm -f elf lab7-1.asm
(base) m1@DESKTOP-14ENTC01:~/Desktop/asm/asm7/asm7 $ ld -m elf_i386 -o lab7-1 lab7-1.o
(base) m1@DESKTOP-14ENTC01:~/Desktop/asm/asm7/asm7 $ ./lab7-1
Сообщение %2
Сообщение %3
```

Screenshot2

Программа с использованием инструкции jmp

Откройте файл lab7-1.asm и добавьте инструкции JMP

[illegible]

Screenshot3

Создаю исполняемый файл программы и запускаю его.

```
(base) # cd /usr/src/linux-headers-$(uname -r)/arch/x86/include/asm/unistd.h
(base) # nano lab7-1.asm
(base) # ld -m elf_i386 -o lab7-1 lab7-1.o
(base) # ./lab7-1
```

Screenshot4

Создаю новый файл lab6-2.asm с помощью утилиты touch. Я ввожу текст другой программы в файл для условного jmp

```

include 'in_out.asm'
section
    .data
    msg1 db 'Введите B: ',0h
    msg2 db 'Наибольшее число: ',0h
    A dd '20'
    C dd '50'
section .bss
    max resb 10
    B resb 10

section
    .text
global _start
_start:
;----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
;----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
;----- Преобразование 'B' из символа в число
mov eax,B
call atoi
mov [B],eax
;----- Записываем 'A' в переменную 'max'
mov ecx,[A]
mov [max],ecx
;----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C]
;----- Сравниваем 'A' и 'C'
jg check_B
;----- если 'A>C', то переход на метку 'check_B',
mov ecx,[C]
;----- иначе 'ecx = C'
mov [max],ecx
;----- Преобразование 'max(A,C)' из символа в число
;check_B
mov eax,max
call atoi
mov [max],eax
;----- Запись преобразованного числа в 'max'
;----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B]
;----- Сравниваем 'max(A,C)' и 'B'
jg fin
;----- если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B]
;----- иначе 'ecx = B'
mov [max],ecx
;----- Вывод результата
;fin
mov eax, msg2
call sprint
mov eax,[max]
call iprintLF
call quit
; Выход

```

Создаю исполняемый файл программы и запускаю его.

```

(base) ~/Desktop-1927C01:~/Desktop-1927C01 $ nasm -f elf lab7-2.asm
(base) ~/Desktop-1927C01:~/Desktop-1927C01 $ ld -m elf_i386 -o lab7-2 lab7-2.o
(base) ~/Desktop-1927C01:~/Desktop-1927C01 $ ./lab7-2
Введите B: 21
Наибольшее число: 50

```

Screenshot7

Изучение структуры файлы листинга

Я создал файл листинга для файла lab7-2.asm с помощью команды - nasm -f elf -l lab7-2.lst lab7-2.asm Я открыл файл lab7-2.lst с помощью команды nano lab7-4.lst

```

GNU nano 7.2 lab7-2.lst
1      #include "in_out.asm"
2      ;----- strlen -----
3      ; Функция вычисления длины сообщения
4      strlen:
5          push    ebx
6          mov     ebx, eax
7
8      nextchar:
9          cmp     byte [eax], 0
10         jz       finished
11         inc     eax
12         jmp     nextchar
13
14     finished:
15         sub     eax, ebx
16         pop     ebx
17         ret
18
19     ;----- sprintf -----
20     ; Функция печати сообщения
21     ; входные данные: mov eax, <message>
22     sprintf:
23         push    edx
24         push    ecx
25         push    ebx
26         push    eax
27         call    strlen
28         ;
29         mov     edx, eax
30         pop     eax
31         ;
32         mov     ecx, eax
33         mov     ebx, 1

```

Screenshot8

Ответы на вопросы по программе

- В строке 5:Эта инструкция сохраняет текущее значение регистра EBX в стек. Обычно это делается, чтобы сохранить значение регистра перед его изменением внутри функции. Машинный код 53 представляет операцию помещения значения регистра EBX в стек.

00000000 53 **push ebx**

- В строке 8:Эта инструкция сравнивает значение, хранящееся по адресу памяти, на который указывает регистр EAX, с 0. Если значение равно 0, устанавливается флаг нуля (Zero Flag, ZF) в процессоре, который может использоваться для условий перехода. Машинный код 803800 кодирует операцию сравнения, указывая, что она работает с байтовым значением.

00000003 803800 **cmp byte [eax], 0**

- В строке 14:та инструкция вычитает значение регистра EBX из значения регистра EAX, сохраняя результат в EAX. Эта операция может использоваться, например, для вычисления длины строки, вычитая начальный адрес (EBX) из текущего адреса (EAX). Машинный код 29D8 указывает операцию вычитания (SUB) между регистрами EAX и EBX.

0000000B 29D8 **sub eax, ebx**

Когда я удалил операнд из инструкции с двумя операндами и попробовал в файле lab7-4.asm . Когда я попытался создать файл .lst, это не сработало. Я получил ошибки из-за того, что файл стал недействительным

```
(base) mik@DESKTOP-14ENPC0: ~/Laboratory-work/Assemblary-компьютера/asm-с/лаб7/lab7$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:15: error: invalid combination of opcode and operands
```

Screenshot9

Выполнение заданий для самостоятельной работы

1. Создаю файл lab7-3.asm с помощью утилиты touch. Открываю созданный файл для редактирования, и написал программу для нахождения наименьшей из трех целочисленных переменных a, b, c. Я выбрал вариант 1

```
GNU nano 7.2 lab7-3.asm
#include "in_out.asm" ; Подключаем внешний файл для вывода и завершения

SECTION .data
a db 17 ; Заменяю значением 'a' из любого варианта
b db 23 ; Заменяю значением 'b' из любого варианта
c db 45 ; Заменяю значением 'c' из любого варианта
smallest db 0 ; Переменная для хранения минимального значения
msg1 db "Наименьшее значение: ", 0

SECTION .text
global _start

_start:
; Загружаем 'a' в %eax и 'b' в %ebx
mov al, [a] ; Загружаем значение 'a' в %al
mov bl, [b] ; Загружаем значение 'b' в %bl
cmp al, bl ; Сравниваем 'a' с 'b'
jle .check_c ; Если 'a' <= 'b', переходим к проверке 'c'
mov al, bl ; Иначе, минимальное значение - это 'b'

.check_c:
; Сравниваем текущее минимальное значение (%eax) с 'c'
mov bl, [c] ; Загружаем значение 'c' в %bl
cmp al, bl ; Сравниваем минимальное значение с 'c'
jle .store_smallest ; Если минимальное значение <= c, переходим к сохранению
mov al, bl ; Иначе, минимальное значение - это 'c'

.store_smallest:
; Сохраняем минимальное значение в переменной 'smallest'
mov [smallest], al ; Сохраняем минимальное значение в памяти

; Переходим к выводу результата
jmp _print_result

;max
mov eax, max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max], eax ; запись преобразованного числа в 'max'
;----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx, [max]
cmp ecx, [B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx, [B] ; иначе 'ecx = B'
mov [max], ecx
;----- Вывод результата
;fin
mov eax, msg2
call sprint ; Вывод сообщения "Наибольшее число: "
mov eax, [max]
call iprintf ; Вывод 'max(A, B, C)'
call quit ; Выход
```

Создаю и запускаю исполняемый файл. Результаты получили 17

```
(base) mik@DESKTOP-14ENPC0: ~/Laboratory-work/Assemblary-компьютера/asm-с/лаб7/lab7$ nasm -f elf lab7-3.asm
(base) mik@DESKTOP-14ENPC0: ~/Laboratory-work/Assemblary-компьютера/asm-с/лаб7/lab7$ ld -m elf_i386 -o lab7-3 lab7-3.o
(base) mik@DESKTOP-14ENPC0: ~/Laboratory-work/Assemblary-компьютера/asm-с/лаб7/lab7$ ./lab7-3
Наименьшее значение: 17
```

Screenshot12

2. Создаю файл lab7-4.asm с помощью утилиты touch. Открываю созданный файл для редактирования, и написал программу, которая для значений x и a, введенных с клавиатуры, вычисляет значение данной функции f(x) и выводит результат вычислений. Я выбрал вариант 1

```

GNU nano 7.2 Lab7-4.asm
#include "in_out.asm" ; Подключаем внешний файл для вывода и завершения

SECTION .data
msg1 db "Введите значение x: ", 0
msg2 db "Введите значение a: ", 0
msg3 db "Результат вычисления f(x): ", 0
f_x db 0 ; Переменная для хранения результата функции

SECTION .bss
x read 1 ; Буфер для хранения значения x
a read 1 ; Буфер для хранения значения a

SECTION .text
GLOBAL _start

_start:
; Ввод значения x
mov eax, msg1
call sprintf
call read_int ; Чтение значения x
mov [x], eax ; Сохраняем значение x

; Ввод значения a
mov eax, msg2
call sprintf
call read_int ; Чтение значения a
mov [a], eax ; Сохраняем значение a

; Загружаем значения x и a
mov eax, [x] ; Загружаем x в eax
jmp _print_result

; Если x <= a, f(x) = 15
mov byte [f_x], 15 ; Устанавливаем f_x = 15

_print_result:
; Выводим сообщение: "Результат вычисления f(x):"
mov eax, msg3
call sprintf

; Выводим результат f(x)
movzx eax, byte [f_x] ; Загружаем результат из f_x в eax
call iprintlf

; Завершаем программу
call quit

; Чтение целого числа с клавиатуры
read_int:
; Используем системный вызов для чтения строки
mov eax, 3 ; Системный вызов для чтения из stdin (0 - стандартный ввод)
mov ebx, 0 ; Стандартный ввод
mov ecx, x ; Адрес буфера для ввода
mov edx, 4 ; Размер буфера (4 байта для одного целого числа)
int 0x80 ; Вызов системного вызова

; Преобразуем строку в число
; Будем считать, что пользователь вводит число в виде строки, а мы его конвертируем
mov eax, [x] ; Загружаем строку в eax
sub eax, '0' ; Преобразуем символ в число (для простоты только для 1 цифры)
ret

```

Создаю и запускаю исполняемый файл

```

(base) miniforger@INERTOS: ~/bin/asm $ name -f elf lab7-4.asm
(base) miniforger@INERTOS: ~/bin/asm $ ld -m elf_i386 -o lab7-4 lab7-4.o
(base) miniforger@INERTOS: ~/bin/asm $ ./lab7-4
Введите значение x: 2
Введите значение a: 1
Результат вычисления f(x): 96

```

Screenshot15

Выводы

Выполняя эту работу, я научился писать программы с использованием переходов и немного узнал о файле .lst

Ссылка на официальный сайт [Github](#)