Лабораторная работа №7

Архитектура компьютера

Кучмар София Игоревна

Содержание

1 Цель работы

Эта работа направлена на изучение команд условного и безусловного переходов, приобретение навыков написания программ с использованием инструкции jmp, знакомство с назначением и структурой файла листинга, команд условного перехода, инструкции cmp.

2 Задание

Данная работа посвящена практическому освоению ассемблера NASM. Будут изучены основы работы адресацией в NASM, освоены арифметические операции в NASM, целочисленное сложение add, целочисленное вычитание sub, команды инкремента и декремента, команда изменения знака операнда neg, основными директивами ассемблера, команды умножения mul и imul и будет написана программу для вычисления выражений с использованием инструкции jmp и программа, которая определяет и выводит на экран наибольшую/ наименьшую из 3 целочисленных переменных: A,B и C. Будет подключен внешний файл in_out.asm с функциями ввода и вывода данных.

3 Выполнение лабораторной работы

Создадим каталог для программам лабораторной работы № 7, перейдём в него и создадим файл lab6-1.asm(рис. 3.1).

Рис. 3.1: Создание каталога и файла в нём

Рассмотрим пример программы с использованием инструкции jmp (рис. 3.2).

```
/home/sikuchmar/work/arch-pc/lab07/lab7-1.asm
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.2: Вводим программу

Создадим исполняемый файл и запустим его (рис. 3.3).

```
sikuchmar@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
sikuchmar@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
sikuchmar@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
sikuchmar@vbox:~/work/arch-pc/lab07$
```

Рис. 3.3: Запуск файла

Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию jmp с меткой _label1 и после вывода сообщения № 1 добавим инструкцию jmp с меткой _end(рис. 3.4).

```
/home/sikuchmar/work/arch-pc/lab07/lab7-1.asm
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
jmp _end
label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.4: Изменение текст программы

Создадим исполняемый файл и запустим его (рис. 3.5)

```
sikuchmar@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
sikuchmar@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
sikuchmar@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рис. 3.5: Запуск файла

Изменим текст программы добавив или изменив инструкции jmp, чтобы вывод программы был следующим: сначала ‘Сообщение № 3’, ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. (рис. 3.6).

```
/home/sikuchmar/work/arch-pc/lab07/lab7-1.asm
%include 'in_out.asm' ; подключение внешнего файла

SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0

SECTION .text
GLOBAL _start
_start:
jmp _label3 ; Начинаем с вывода msg3

_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF
jmp _label2

_end:
call quit
```

Рис. 3.6: Изменение текст программы

Создадим исполняемый файл и запустим его (рис. 3.7).

```
sikuchmar@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
sikuchmar@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
sikuchmar@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 3.7: Запуск файла

Создадим файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 и создадим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных

переменных: A,B и C. Значения для A и C задаются в программе, значение B вводиться с клавиатуры. (рис. 3.8).

```
/home/sikuchmar/work/arch-pc/lab07/lab7-2.asm
%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 ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ---------- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ---------- 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 ; 'max = C'
; ---------- Преобразование 'max(A,C)' из символа в число
check_B:
```

Рис. 3.8: Создание программы

Создадим исполняемый файл и проверим его работу для разных значений B (рис. 3.9).

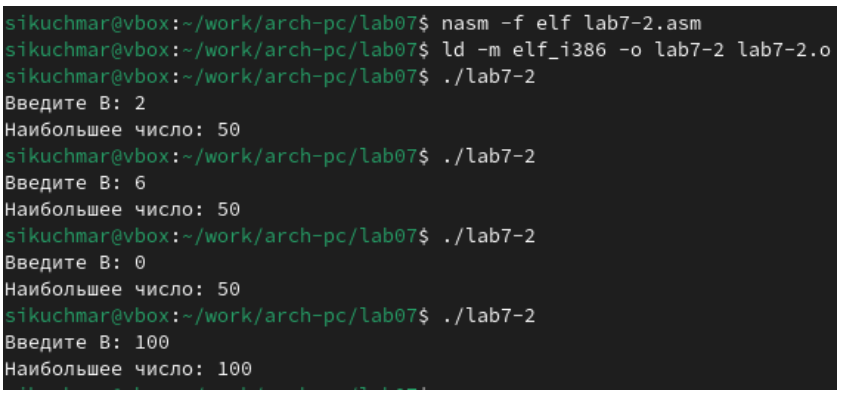

Рис. 3.9: Запуск файла

Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ -l и задав имя файла листинга в командной строке. Создадим файл листинга для программы из файла lab7-2.asm(рис. 3.10). Откроем файл листинга lab7-2.lst с помощью любого текстового редактора. Внимательно ознакомиться с его форматом и содержимым. Например: Строка 8: cmp byte [eax], 0 сравнивает байт по адресу eax с нулем, проверяя конец строки (нуль-терминатор).

Строка 14: sub eax, ebx вычисляет длину строки, вычитая начальный адрес (ebx) из конечного (eax).

Строка 27: call slen вызывает функцию slen для определения длины строки.

Давайте удалим один операнд из инструкции в строке 27. Например, удалим slen. В результате будут созданы два файла lab7-2.o : объектный файл, содержащий машинный код и lab7-2.lst: листинговый файл.В листинговом файле в строке 27 мы увидим сообщение об ошибке от ассемблера. Ассемблер не сможет корректно собрать программу, так как инструкция call требует операнда.

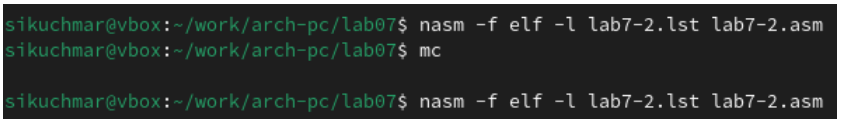

Рис. 3.10: Создание lst файлов

Напишем программу нахождения наименьшей из 3 целочисленных перемен-

ных a,b и c. (рис. 3.11).

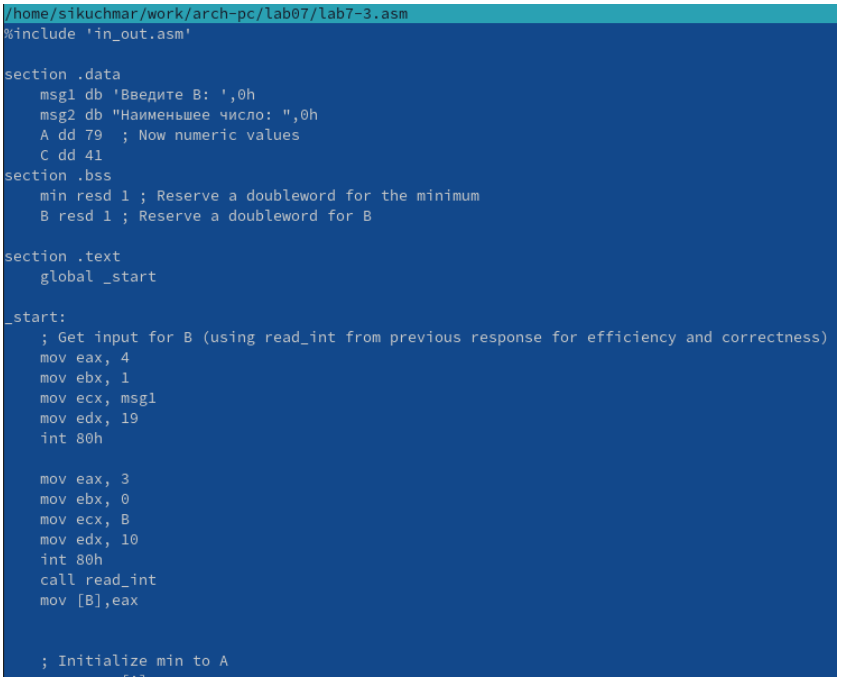

Рис. 3.11: Создание программы

Создадим исполняемый файл и запустим его для значений Варианта 6: a=79, b=83, c=41 (рис. 3.12).

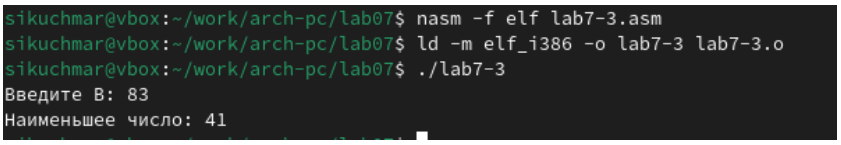

Рис. 3.12: Запуск файла

Напишем программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции x+a, при x не = a, 5x, при x=a и выводит результат вычислений. (рис. 3.13).

```
/home/sikuchmar/work/arch-pc/lab07/lab7-4.asm
%include 'in_out.asm'

section .data
  msg1 db 'Введите A: ',0h
  msg2 db 'Введите X: ',0h
  msg3 db "Результат: ",0h
  five dd 5

section .bss
  fin resd 1 ; Use resd for integer results
  A resd 1  ; Use resd for integers
  X resd 1  ; Use resd for integers

section .text
  global _start

_start:
  ; Input A
  mov eax, 4
  mov ebx, 1
  mov ecx, msg1
  mov edx, 19
  int 80h

  mov eax, 3
  mov ebx, 0
  mov ecx, A
  mov edx, 10
  int 80h
  call read_int ; efficient integer input
  mov [A], eax
```

Рис. 3.13: Создание программы

Создадим исполняемый файл и запустим его для значений Варианта 6: x1=2, a1=1, x2=2, a2=1 (рис. 3.14).

```
sikuchmar@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
sikuchmar@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
sikuchmar@vbox:~/work/arch-pc/lab07$ ./lab7-4
Введите A: 2
Введите X: 2
Результат: 4
sikuchmar@vbox:~/work/arch-pc/lab07$ ./lab7-4
Введите A: 1
Введите X: 2
Результат: 10
```

Рис. 3.14: Запуск файла

4 Выводы

В рамках данной работы были успешно освоены основы работы с ассемблером NASM. Были освоены разные операции с переменными, такие как сравнение и вычисление наибольшего и наименьшего в NASM и создание программ для вычисления выражений.