Отчёта по лабораторной работе №7

Архитектура компьютера

Еремина Оксана Андреевна НКАбд-02-23

Содержание

Список таблиц

Список иллюстраций

1 Цель работы

Целью данной лабораторной работы является освоение арифметческих инструкций языка ассемблера NASM.

2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлы листинга
3. Выполнение заданий для самостоятельной работы

3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов: • условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия. • безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

Флаг – это бит, принимающий значение 1 («флаг установлен»), если выполнено некоторое условие, и значение 0 («флаг сброшен») в противном случае. Флаги работают независимо друг от друга, и лишь для удобства они помещены в единый регистр — регистр флагов, отражающий текущее состояние процессора

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

4 Выполнение лабораторной работы

1. Реализация переходов в NASM

Создаю каталог для программам лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm. (рис.1)

```
oaeremina@oaeremina:~$ mkdir ~/work/arch-pc/lab07
oaeremina@oaeremina:~$ cd ~/work/arch-pc/lab07
oaeremina@oaeremina:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Рис. 4.1: Создание папки и файла

Ввожу в файл lab7-1.asm текст программы. (рис.2)

```
%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. 4.2: Содержание файла lab7-1.asm

Создаю исполняемый файл и проверяю работу программы. (рис.3)

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

Рис. 4.3: Выподнение программы

Изменяю код программы так, чтобы она сначала выводила ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. (рис.4)

```
 1 %include 'in_out.asm' ; подключение внешнего файла
 2 SECTION .data
 3 msg1: DB 'Сообщение № 1',0
 4 msg2: DB 'Сообщение № 2',0
 5 msg3: DB 'Сообщение № 3',0
 6 SECTION .text
 7 GLOBAL _start
 8 _start:
 9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintLF ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintLF ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintLF ; 'Сообщение № 3'
21 _end:
22 call quit ; вызов подпрограммы завершения
```

Рис. 4.4: Изменение программы

Создаю исполняемый файл и проверяю работу программы. (рис.5)

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

Рис. 4.5: Выподнение программы

Изменяю код программы так, чтобы она сначала выводила ‘Сообщение № 3’, потом ‘Сообщение № 2’, ‘Сообщение № 1’ и завершала работу. (рис.6)

```
 1 %include 'in_out.asm' ; подключение внешнего файла
 2 SECTION .data
 3 msg1: DB 'Сообщение № 1',0
 4 msg2: DB 'Сообщение № 2',0
 5 msg3: DB 'Сообщение № 3',0
 6 SECTION .text
 7 GLOBAL _start
 8 _start:
 9 jmp _label3
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintLF ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintLF ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintLF ; 'Сообщение № 3'
21 jmp _label2
22 _end:
23 call quit ; вызов подпрограммы завершения
```

Рис. 4.6: Изменение программы

Создаю исполняемый файл и проверяю работу программы. (рис.7)

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

Рис. 4.7: Выподнение программы

Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Ввожу код программы в созданный файл. (рис.8)

```
%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:
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 iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рис. 4.8: Содержание файла lab7-2.asm

Создаю исполняемый файл и провяю его работу для разных значений B (1; 5). (рис.9)

```
oaeremina@oaeremina:~/work/arch-pc/lab07$ touch lab7-2.asm
oaeremina@oaeremina:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
oaeremina@oaeremina:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
oaeremina@oaeremina:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 1
Наибольшее число: 50
oaeremina@oaeremina:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
oaeremina@oaeremina:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
oaeremina@oaeremina:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
```

Рис. 4.9: Выполнение программы

2. Изучение структуры файлы листинга

Создаю файл листинга для программы из файла lab7-2.asm, знакамлюсь с его форматом и содержимым файла листинга, она показан ниже. (рис.10)

```
1  1                                %include 'in_out.asm'
2  2                            <1> ;---------------   slen  -------------------
3  3                            <1> ; Функция вычисления длины сообщения
4  4                            <1> slen:
5  5 00000000 53                <1>     push    ebx
6  6 00000001 89C3              <1>     mov     ebx, eax
7  7                            <1>
8  8                            <1> nextchar:
9  9 00000003 803800            <1>     cmp     byte [eax], 0
10 10 00000006 7403             <1>     jz      finished
11 11 00000008 40               <1>     inc     eax
12 12 00000009 EBF8             <1>     jmp     nextchar
13 13                           <1>
14 14                           <1> finished:
15 15 0000000B 29D8             <1>     sub     eax, ebx
16 16 0000000D 5B               <1>     pop     ebx
17 17 0000000E C3               <1>     ret
18 18                           <1>
19 19                           <1>
20 20                           <1> ;---------------  sprint  -------------------
21 21                           <1> ; Функция печати сообщения
```

Рис. 4.10: Структура и содержимое файла листинга

Открываю файл с программой lab7-2.asm и в выбранной мной инструкции с двумя операндами удаляю выделенный операнд. (рис.11)

```
26 mov [max],ecx ; 'max = A'
27 ; ---------- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 jg check_B ; если 'A>C', то переход на метку 'check
30 mov eax, [C] ; иначе 'ecx = C'
31 mov [max],ecx ; 'max = C'
32 ; ---------- Преобразование 'max(A,C)' из символа в
33 check_B:
34 mov eax, max
35 call atoi ; Вызов подпрограммы перевода символа в ч
36 mov [max],eax ; запись преобразованного числа в `ma
37 ; ---------- Сравниваем 'max(A,C)' и 'B' (как числа
38 mov ecx,[max]
39 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 jg fin ; если 'max(A,C)>B', то переход на 'fin',
```

Рис. 4.11: Изменение файла

Выполняю трансляцию с получением файла листинга. На выходе я не получаю ни одного файла из-за ошибки:инструкция mov (единственная в коде содер-

жит два операнда) не может работать, имея только один операнд, из-за чего нарушается работа кода.(рис.12)

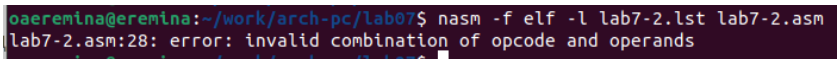

Рис. 4.12: Ошибка в работе кода

3. Выполнение заданий для самостоятельной работы

Пишу программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбираю из табл. 7.5. Мой вариант под номером 17, поэтому мои значения - 26, 12 и 68.

```
%include 'in_out.asm'
section .data
msg db "Наименьшее число: ",0h
A dd '26'
B dd '12'
C dd '68'
section .bss
min resb 10
section .text
global _start
_start:
mov ecx,[A]
mov [min], ecx
cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [min], ecx
check_B:
mov eax, min
call atoi
mov [min],eax
mov ecx,[min]
```

Рис. 4.13: Код прогрраммы

Создаю исполняемый файл и проверяю работу программы. (рис.14)

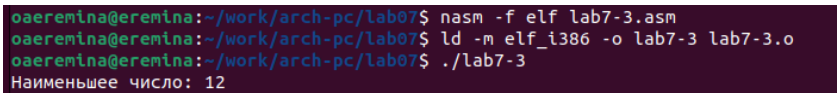

Рис. 4.14: Выподнение программы

Код программы:

%include ‘in_out.asm’

```
section .data
msg db “Наименьшее число:”,0h
A dd ‘26’
B dd ‘12’
C dd ‘68’
section .bss
min resb 10
section .text
global _start
_start:
mov ecx,[A] ; ‘ecx = A’
mov [min],ecx ; ‘min = A’
cmp ecx,[C] ; Сравниваем ‘A’ и ‘C’
jg check_B
mov ecx,[C] ; иначе ‘ecx = C’
mov [min],ecx ; ‘min = C’
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в min
mov ecx,[min]
cmp ecx,[B] ; Сравниваем ‘min(A,C)’ и ‘B’
jl fin
mov ecx,[B]
mov [min],ecx
fin:
mov eax, msg
call sprint ; Вывод сообщения ‘Наименьшее число:’
mov eax,[min]
```

```
call iprintLF ; Вывод ‘min(A,B,C)’
call quit ; Выход
```

5 Вывод

При выполнении данной лабораторной работы я изучила команды условного и безусловного перехода, приобрела навык написания программ с использованием переходов и познакомилась с назначением и структурой файла листинга.

Список литературы

Архитектура ЭВМ