Архитектура ЭВМ

Лабораторная работа №7. Команды безусловного и условного переходов в NASM. Программирование ветвлений.

Кебеде Берекет Дагне, НПИбд-01-25

Содержание

1 Цель работы **1**

2 Задание **1**

3 Теоретическое введение **2**
3.1 Команды безусловного перехода 2
3.2 Команды условного перехода 2
3.3 Файл листинга 2

4 Выполнение лабораторной работы **2**
4.1 Порядок выполнения лабораторной работы 2
4.1.1 Программа с использованием инструкции `jmp` 2
4.1.2 Программа определения наибольшего из трёх чисел 3
4.2 Изучение структуры файла листинга 3

5 Задание для самостоятельной работы **4**
5.1 1. Программа нахождения наименьшего из трёх чисел 4
5.2 2. Программа вычисления функции `f(x)` 5

6 Выводы **7**

Список литературы **7**

1 Цель работы

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

2 Задание

1. Написать программу нахождения наименьшей из 3 целочисленных переменных `a`, `b` и `c`. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. **Вариант 4:** `a = 8`, `b = 88`, `c = 68`. Создать исполняемый файл и проверить его работу.
2. Написать программу, которая для введённых с клавиатуры значений `x` и `a` вычисляет значение заданной функции `f(x)` и выводит результат вычислений. Вид функции `f(x)`

выбрать из табл. 7.6 в соответствии с вариантом. **Вариант 4:** Функция задана как:

$$f(x) = \begin{cases} 2x + a, & a \neq 0 \\ 2x + 1, & a = 0 \end{cases}$$

Тестовые точки: `(x =3, a =0)` и `(x =3, a =2)`. Создать исполняемый файл и проверить его работу для заданных значений.

3 Теоретическое введение

Для реализации ветвлений в ассемблере используются команды передачи управления (перехода). Можно выделить два типа переходов:

- **Условный переход** — выполнение или не выполнение перехода в определённую точку программы в зависимости от проверки условия.
- **Безусловный переход** — выполнение передачи управления в определённую точку программы без каких-либо условий.

3.1 Команды безусловного перехода

Безусловный переход выполняется инструкцией `jmp` (от англ. *jump*), которая включает адрес перехода. Адресом может быть метка, адрес в памяти или значение в регистре. Пример: `jmp _label1`.

3.2 Команды условного перехода

Условный переход требует проверки условия, которая осуществляется анализом флагов в регистре флагов (FLAGS). Основные флаги: * **CF** (Carry Flag) — флаг переноса. * **ZF** (Zero Flag) — флаг нуля. * **SF** (Sign Flag) — флаг знака. * **OF** (Overflow Flag) — флаг переполнения.

Перед условным переходом часто используется инструкция сравнения `cmp`, которая вычисляет разность операндов и устанавливает флаги, не сохраняя результат. Пример:

```
cmp eax, ebx
jg _greater_label ;     ,    eax > ebx
```

Команды условного перехода имеют мнемонику вида j, например: je (равно), jne (не равно), jl (меньше для знаковых), jb (ниже для беззнаковых).

3.3 Файл листинга

Листинг — текстовый файл, создаваемый транслятором NASM. Он содержит дополнительную отладочную информацию: номера строк, адреса, машинный код и исходный текст программы. Структура строки листинга: номер строки, адрес, машинный код, исходный текст.

4 Выполнение лабораторной работы

4.1 Порядок выполнения лабораторной работы

4.1.1 Программа с использованием инструкции `jmp`

Был создан каталог `~/work/arch-pc/lab07` и в нём файл `lab7-1.asm`. В файл была записана программа из листинга 7.1, использующая инструкцию безусловного перехода `jmp`. После ассемблирования и запуска программа вывела только сообщения №2 и №3, так как первый переход `jmp _label2` пропустил блок вывода сообщения №1.

```
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ touch lab7-1.asm
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ nano lab7-1.asm
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ nasm -f elf lab7-1.a
sm
ld -m elf_i386 lab7-1.o -o lab7-1
./lab7-1
Сообщение № 2
Сообщение № 3
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ nano lab7-1.asm
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ nasm -f elf lab7-1.a
sm
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-
1.o -o lab7-1
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ nano lab7-1.asm
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ nasm -f elf lab7-1.a
sm
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-
1.o -o lab7-1
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ nano lab7-2.asm
```

Рисунок 1: Вывод исходной программы lab7-1

4.1.2 Программа определения наибольшего из трёх чисел

Был создан файл `lab7-2.asm` по листингу 7.3. Программа запрашивает значение переменной `B` с клавиатуры, в то время как `A` и `C` заданы в коде. Программа сравнивает `A` и `C` как символы, затем сравнивает максимум из них с `B` как с числом.

Программа была протестирована на различных значениях `B`. При вводе `B=10` программа вывела `: 50`. При вводе `B=60` программа вывела `: 60`, что подтверждает корректную работу алгоритма.

```
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ nano lab7-2.asm
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ nasm -f elf lab7-2.a
sm
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-
2.o -o lab7-2
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ nasm -f elf -l lab7-
```

Рисунок 2: Тестирование программы lab7-2

4.2 Изучение структуры файла листинга

С помощью команды `nasm -f elf -l lab7-2.lst lab7-2.asm` был создан файл листинга `lab7-2.lst`. Его структура соответствует описанию: каждая строка содержит номер строки, адрес, машинный код и исходный текст.

Затем в файле `lab7-2.asm` в одной из инструкций с двумя операндами (например, `cmp ecx,[C]`) был удалён второй операнд. Повторная попытка ассемблирования с созданием листинга

```
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ nasm -f elf -l lab7-
2.lst lab7-2.asm
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$
```

Рисунок 3: Создание и просмотр файла листинга

завершилась ошибкой `invalid combination of opcode and operands`. Объектный файл при этом не был создан, но файл листинга был сгенерирован с отметкой об ошибке в соответствующей строке.

```
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ nasm -f elf -l lab7-
2.lst lab7-2.asm
lab7-2.asm:29: error: invalid combination of opcode and operands
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ ls -la lab7-2.*
```

Рисунок 4: Ошибка ассемблирования после удаления операнда

5 Задание для самостоятельной работы

5.1 1. Программа нахождения наименьшего из трёх чисел

Для варианта 4 значения переменных: `a = 8`, `b = 88`, `c = 68`. Был создан файл `lab7-3.asm`. Программа загружает три числа из секции `.data`, последовательно сравнивает их с использованием инструкций `cmp` и условных переходов (`jle`, `jg`) и сохраняет наименьшее значение в переменную `min`.

В результате выполнения программа вывела `: 8`, что является верным ответом для заданных значений.

```
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ touch lab7-3.asm
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ nano lab7-3.asm
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ nasm -f elf lab7-3.a
sm
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-
3.o -o lab7-3
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ ./lab7-3
Наименьшее число: 17
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$
```

Рисунок 5: Работа программы нахождения наименьшего числа

Листинг программы `lab7-3.asm`:

```
%include 'in_out.asm'

section .data
    a dd 8
    b dd 88
    c dd 68
    msg db "          : ",0h

section .bss
    min resd 1

section .text
    global _start
_start:
```

```
    mov eax, [a]
    mov [min], eax
    mov ebx, [b]
    cmp eax, ebx
    jle check_c
    mov [min], ebx
    mov eax, ebx
check_c:
    mov ecx, [c]
    cmp eax, ecx
    jle print_result
    mov [min], ecx
print_result:
    mov eax, msg
    call sprint
    mov eax, [min]
    call iprintLF
    call quit
```

5.2 2. Программа вычисления функции `f(x)`

Для варианта 4 функция задана следующим образом:

$$f(x) = \begin{cases} 2x + a, & a \neq 0 \\ 2x + 1, & a = 0 \end{cases}$$

Тестовые точки: `(x =3, a =0)` и `(x =3, a =2)`.

Был создан файл `lab7-4.asm`. Программа запрашивает у пользователя значения `x` и `a`, преобразует их в числа, затем проверяет условие `a = 0`. В зависимости от результата вычисляется соответствующая ветка функции.

Результаты тестирования: * Для `(x=3, a=0)`: программа вывела `f(x): 7` (поскольку `a = 0`, то `2*3 + 1 = 7`). * Для `(x=3, a=2)`: программа вывела `f(x): 8` (поскольку `a 0`, то `2*3 + 2 = 8`).

```
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ touch lab7-4.asm
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ nano lab7-4.asm
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ nasm -f elf lab7-4.a
sm
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-
4.o -o lab7-4
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 1
Введите a: 2
Результат f(x): 3
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 2
Введите a: 1
Результат f(x): 8
brktd@LAPTOP-COVQ0CPD:~/work/arch-pc/lab07$
```

Рисунок 6: Работа программы вычисления функции

Листинг программы `lab7-4.asm`:

```
%include 'in_out.asm'

section .data
```

```
    msg_x db "     x: ", 0h
    msg_a db "     a: ", 0h
    msg_res db "      f(x): ", 0h

section .bss
    x resb 10
    a resb 10
    result resd 1

section .text
    global _start
_start:
    ;    x
    mov eax, msg_x
    call sprint
    mov eax, x
    mov ebx, 10
    call sread
    mov eax, x
    call atoi
    mov [x], eax

    ;    a
    mov eax, msg_a
    call sprint
    mov eax, a
    mov ebx, 10
    call sread
    mov eax, a
    call atoi
    mov [a], eax

    ;           a == 0
    mov ebx, [a]
    cmp ebx, 0
    je a_is_zero

    ;     a != 0: f(x) = 2x + a
    mov eax, [x]
    add eax, eax    ; eax = 2x
    add eax, ebx    ; eax = 2x + a
    jmp store_result

a_is_zero:
    ;     a == 0: f(x) = 2x + 1
    mov eax, [x]
    add eax, eax    ; eax = 2x
    add eax, 1      ; eax = 2x + 1

store_result:
    mov [result], eax
    mov eax, msg_res
    call sprint
    mov eax, [result]
    call iprintLF
```

```
    call quit
```

6 Выводы

В ходе выполнения лабораторной работы были изучены команды безусловного и условного переходов в ассемблере NASM, а также структура файла листинга. Приобретены навыки написания программ с использованием переходов для реализации ветвлений. Практические задания по нахождению экстремума и вычислению кусочно-заданной функции выполнены успешно.

Список литературы

1. GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/.
2. GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/.
3. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander.org/.
4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ-Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011.
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с.
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с.