Лабораторная работа №7

Команды безусловного и условного переходов в NASM. Программирование ветвлений

Студент: Зайцев Андрей Алексеевич

---

1 Цель работы

Изучить команды безусловного (`jmp`) и условного переходов (`je`, `jne`, `jg`, `jl` и др.) в языке ассемблера NASM, а также приобрести навыки написания программ с ветвлениями и анализа листингов ассемблерных программ.

---

2 Задание

1. Написать программу на языке ассемблера NASM, демонстрирующую работу команды безусловного перехода `jmp` (файл `lab7-1.asm`).

2. Написать программу для определения максимального значения из трёх целочисленных переменных `A`, `B`, `C`, используя условные переходы (файл `lab7-2.asm`). Переменные `A` и `C` задать в сегменте данных, `B` вводить с клавиатуры.

3. Выполнить трансляцию программы `lab7-2.asm` с получением файла листинга `lab7-2.lst`, проанализировать его структуру.

4. **Самостоятельная работа 1.** Написать программу `min3.asm` для поиска минимального значения из трёх чисел `a`, `b`, `c`, заданных в сегменте данных.

5. **Самостоятельная работа 2.** Написать программу `fx.asm` для вычисления значения функции

$$

f(x) =

\begin{cases}

3a, & a < 3,\\

x + 1, & a \ge 3,

\end{cases}

$$

где параметры `x` и `a` вводятся с клавиатуры.

---

3 Теоретическое введение

Ветвление в ассемблерных программах на x86 реализуется с помощью:

- инструкции **сравнения** `cmp`, которая вычитает второе значение из первого и устанавливает флаги процессора (ZF, SF, OF и др.);

- инструкции **безусловного перехода** `jmp`, изменяющей значение счётчика команд без проверки флагов;

- инструкций **условного перехода** (`je`, `jne`, `jg`, `jge`, `jl`, `jle` и др.), которые выполняют переход в зависимости от состояния флагов.

Типичный шаблон использования условного перехода:

```asm

cmp eax, ebx   ; сравнение двух значений
```

jg label   ; переход, если eax > ebx (signed)

```

В лабораторной работе используется библиотека `in_out.asm`, предоставляющая процедуры:

- `sprint`, `sprintLF` — вывод строки;
- `sread` — ввод строки с клавиатуры;
- `atoi` — преобразование строки в целое число;
- `iprint`, `iprintLF` — вывод целого числа;
- `quit` — завершение программы (вызов `int 0x80` с кодом 1).

---

## 4 Выполнение лабораторной работы

### 4.1 Программа lab7-1.asm — безусловные переходы

Программа `lab7-1.asm` выводит три сообщения на экран и демонстрирует безусловные переходы `jmp` между метками.

**Листинг программы `lab7-1.asm`:**

```asm
; lab7-1.asm
; Демонстрация безусловных и условных переходов

```
%include "in_out.asm"

section .data
msg1 db 'Сообщение 1', 0
msg2 db 'Сообщение 2', 0
msg3 db 'Сообщение 3', 0

section .text
global _start

_start:
   mov eax, msg1
   call sprintLF

   jmp label2   ; безусловный переход

label1:
   mov eax, msg2
   call sprintLF
   jmp finish

label2:
   mov eax, msg3
   call sprintLF

   jmp label1   ; переход к label1
```

finish:

call quit

```

### 4.2 Программа lab7-2.asm — максимум из трёх чисел

Программа `lab7-2.asm` вычисляет максимальное значение из трёх целочисленных переменных `A`, `B`, `C`.

Значения `A` и `C` жёстко заданы в сегменте данных (`A = 20`, `C = 50`), значение `B` вводится пользователем с клавиатуры. Для определения максимума используются инструкция `cmp` и условные переходы.

**Листинг программы `lab7-2.asm`:**

```asm

; lab7-2.asm

; Определение и вывод максимума из трех чисел A, B, C

%include "in_out.asm"

section .data

msgB db 'Введите B: ',0

msgR db 'Наибольшее число: ',0

A dd 20

```
C dd 50

section .bss
Bstr resb 10
Bval resd 1
max resd 1

section .text
global _start

_start:
   ; ----- запрос B -----
   mov eax, msgB
   call sprint

   mov ecx, Bstr
   mov edx, 10
   call sread

   ; ----- B → число -----
   mov eax, Bstr
   call atoi
   mov [Bval], eax

   ; ----- max = A -----
   mov eax, [A]
```

```
    ; ----- сравнение с C -----
    mov ebx, [C]
    cmp eax, ebx
    jge compare_B
    mov eax, ebx

compare_B:
    ; ----- сравнение с B -----
    mov ebx, [Bval]
    cmp eax, ebx
    jge done
    mov eax, ebx

done:
    mov [max], eax

    ; ----- вывод результата -----
    mov eax, msgR
    call sprint

    mov eax, [max]
    call iprintLF

    call quit
```

```

### 4.3 Анализ листинга lab7-2.lst

При трансляции с ключом `-l` был получен файл листинга `lab7-2.lst`. В нём для каждой строки исходного кода указаны:

1. номер строки в листинге;

2. адрес команды в памяти;

3. машинный код инструкции в шестнадцатеричном виде;

4. исходный текст строки программы.

Анализ листинга позволяет отследить соответствие между ассемблерным кодом и машинными командами, а также облегчает поиск ошибок.

### 4.4 Самостоятельная работа 1 — программа min3.asm

Для закрепления навыков условных переходов написана программа `min3.asm`, которая находит минимальное значение из трёх целых чисел `a`, `b`, `c`, заданных в сегменте данных (`a = 52`, `b = 33`, `c = 40`).

**Листинг программы `min3.asm`:**

```asm
%include "in_out.asm"

section .data

msg_res db 'Минимальное значение: ',0
```
```

```
a dd 52
b dd 33
c dd 40

section .bss
min resd 1      ; переменная для хранения минимума

section .text
global _start

_start:
   ; начальный минимум = a
   mov eax, [a]
   mov ebx, [b]
   mov ecx, [c]

   ; сравнение c b
   cmp eax, ebx
   jle check_c
   mov eax, ebx

check_c:
   ; сравнение c c
   cmp eax, ecx
   jle save
```

```
    mov eax, ecx

save:
    mov [min], eax        ; сохранили минимум

    ; вывод строки
    mov eax, msg_res
    call sprint

    ; вывод числа
    mov eax, [min]
    call iprintLF

    call quit
```

4.5 Самостоятельная работа 2 — программа fx.asm

Вариант 8: требуется реализовать вычисление функции

$$
f(x) =
\begin{cases}
3a, & a < 3,\\
x + 1, & a \ge 3.

\end{cases}

$$

Значения `x` и `a` вводятся пользователем с клавиатуры. После ввода программа сравнивает `a` с числом 3 и в зависимости от результата выбирает одну из двух ветвей:

- при `a < 3` вычисляется `f = 3 * a`;

- при `a >= 3` вычисляется `f = x + 1`.

Листинг программы `fx.asm`:

```asm
; fx.asm — вычисление f(x) для варианта 8
; f(x) = 3a,    если a < 3
; f(x) = x + 1, если a >= 3

%include "in_out.asm"

section .data
msgX db 'Введите x: ',0
msgA db 'Введите a: ',0
msgF db 'Значение f(x): ',0

section .bss
x_str resb 10
a_str resb 10
```

```
x_val resd 1
a_val resd 1
f_val resd 1

section .text
global _start

_start:
    ; --- ввод x ---
    mov eax, msgX
    call sprint
    mov ecx, x_str
    mov edx, 10
    call sread
    mov eax, x_str
    call atoi
    mov [x_val], eax

    ; --- ввод a ---
    mov eax, msgA
    call sprint
    mov ecx, a_str
    mov edx, 10
    call sread
    mov eax, a_str
    call atoi
```

```
    mov [a_val], eax

    ; --- вычисление f(x) ---
    mov eax, [a_val]
    cmp eax, 3
    jl  less3         ; если a < 3 → ветка 3a

    ; ветка a >= 3: f = x + 1
    mov eax, [x_val]
    add eax, 1
    jmp save

less3:
    ; ветка a < 3: f = 3a
    mov eax, [a_val]
    imul eax, eax, 3

save:
    mov [f_val], eax

    ; --- вывод результата ---
    mov eax, msgF
    call sprint
    mov eax, [f_val]
    call iprintLF
```

```
    call quit
```

---

5 Выводы

В ходе выполнения лабораторной работы были изучены команды безусловного (`jmp`) и условного переходов в языке ассемблера NASM. На примере нескольких программ были реализованы линейные и разветвляющиеся алгоритмы: вывод сообщений, поиск максимума и минимума из трёх чисел, вычисление кусочно-заданных функций.

Был получен практический опыт:

- использования инструкций `cmp` и условных переходов;
- организации ветвлений и выбора нужной ветки алгоритма по условию;
- работы с библиотекой ввода-вывода `in_out.asm`;
- анализа листингов ассемблерных программ и поиска ошибок по сообщению транслятора.

Полученные навыки могут быть использованы при дальнейшем изучении архитектуры ЭВМ и низкоуровневого программирования.

Список литературы

1. Методические указания к лабораторной работе №7 «Команды безусловного и условного переходов в NASM. Программирование ветвлений».

2. NASM — The Netwide Assembler. Официальная документация.

3. Таненбаум Э. Архитектура компьютера. — СПб.: Питер, 2013.