Отчёт по лабораторной работе №8

Выполнил студент НКАбд-01-22

Никита Михайлович Демидович

Содержание

Список иллюстраций

Список таблиц

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

8.3.1. Реализация переходов в NASM

1. Создайте каталог для программам лабораторной работы No 8, перейдите в него и создайте файл lab8-1.asm: mkdir ~/work/arch-pc/lab08 cd ~/work/arch-pc/lab08 touch lab8-1.asm
2. Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp. Введите в файл lab8-1.asm текст программы из предложенного листинга.

Создайте исполняемый файл и запустите его. Результат работы данной программы будет следующим: user@dk4n31:~$ ./lab8-1 Сообщение No 2 Сообщение No 3 user@dk4n31:~$ Таким образом, использование инструкции jmp _label2 меняет порядок ис- полнения инструкций и позволяет выполнить инструкции начиная с метки _label2, пропустив вывод первого сообщения. Инструкция jmp позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение No 2’, потом ‘Сообщение No 1’ и завершала работу. Для этого в текст программы после вывода сообщения No 2 добавим инструкцию jmp с меткой _label1 (т.е. переход к инструкциям вывода сообщения No 1) и после вывода сообщения No 1 добавим инструкцию jmp с меткой _end (т.е. переход к инструкции call quit). Измените текст программы в соответствии с листингом.

Измените текст программы добавив или изменив инструкции jmp, чтобы вывод программы был следующим: user@dk4n31:~$ ./lab8-1 Сообщение No 3 Сообщение No 2 Сообщение No 1 user@dk4n31:~$ 3. Использование инструкции jmp

приводит к переходу в любом случае. Од- нако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим 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.

Создайте исполняемый файл и проверьте его работу.

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 определен- ную точку программы в зависимости от проверки условия. • безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

8.2.1. Команды безусловного перехода.

Безусловный переход выполняется инструкцией jmp (от англ. jump – прыжок), которая включает в себя адрес перехода, куда следует передать управление: mp Адрес перехода может быть либо меткой, либо адресом области памяти, в которую предварительно помещен указатель перехода. Кроме того, в качестве операнда можно использовать имя регистра, в таком случае переход будет осу- ществляться по адресу, хранящемуся в этом регистре.

Тип операнда	Описание
`jmp label`	Переход на метку labe
`mp [label]`	Переход по адресу в памяти, помеченному меткой label

Тип операнда	Описание
`jmp eax`	Переход по адресу из регистра eax

В следующем примере рассмотрим использование инструкции jmp:

```
label:
    ...     ;
    ...     ; команды
    ...     ;
    jmp  label
```

Рис. 3.1: Использование инструкции jmp

8.2.2. Команды условного перехода

Как отмечалось выше, для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов.

8.2.2.1. Регистр флагов.

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. В следующей таблице указано положение битовых флагов в регистре флагов:

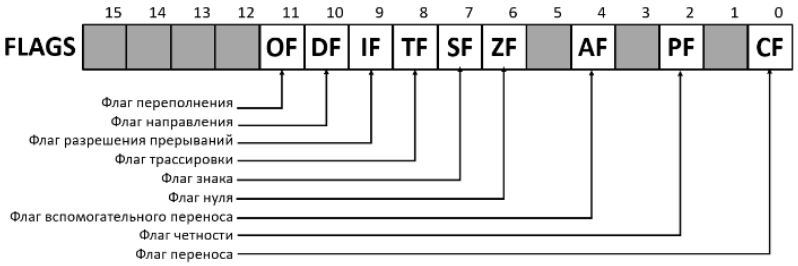

Рис. 3.2: Регистр флагов

Флаги состояния (биты 0, 2, 4, 6, 7 и 11) отражают результат выполнения арифметических инструкций, таких как ADD, SUB, MUL, DIV.

8.2.2.2. Описание инструкции cmp.

Инструкция cmp является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция cmp является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания: cmp , Команда cmp, так же как и команда вычитания, выполняет вычитание - , но результат вычитания никуда не записывается и единственным результатом команды сравнения является формирование флагов. Примеры:

Типы операндов	Мнемокод	Критенрий условного перехода $a \vee b$	Значения флагов	Коммента-рий
Любые	JE	$a = b$	ZF = 1	Переход если равно
Любые	JNE	$a \neq b$	ZF = 0	Переход если не равно
Со знаком	JL/JNGE	$a < b$	SF $\neq$ OF	Переход если меньше
Со знаком	JLE/JNG	$a \leq b$	SF $\neq$ OF или ZF = 1	Переход если меньше или равно
Со знаком	JG/JNLE	$a > b$	SF = OF и ZF = 0	Переход если больше
Со знаком	JGE/JNL	$a \geq b$	SF = OF	Переход если больше или равно
Без знака	JB/JNAE	$a < b$	CF = 1	Переход если ниже

Рис. 3.3: Описание инструкции cmp

8.2.2.3. Описание команд условного перехода.

Команда условного перехода имеет вид j label Мнемоника перехода связана со значением анализируемых флагов или со способом формирования этих флагов. В табл. 8.3. представлены команды условного перехода, которые обычно ставятся после команды сравнения cmp. В их мнемокодах указывается тот результат сравнения, при котором надо делать переход. Мнемоники, идентичные по своему действию, написаны в таблице через дробь (например, ja и jnbe). Программист выбирает, какую из них применить, чтобы получить более простой для понимания текст программы.

Инструкции условной передачи управления по результатам арифметического сравнения cmp a,b

Типы операндов	Мнемокод	Критенрий условного перехода $a \vee b$	Значения флагов	Комментарий
Без знака	JBE/JNA	$a \leq b$	CF = 1 или ZF = 1	Переход если ниже или равно
Без знака	JA/JNBE	$a > b$	CF = 0 и ZF = 0	Переход если выше
Без знака	JAE/JNB	$a \geq b$	CF = 0	Переход если выше или равно

Рис. 3.4: Инструкции условной передачи управления по результатам арифметического сравнения cmp a,b

Мнемокод	Значение флага для осуществления перехода	Мнемокод	Значение флага для осуществления перехода
JZ	ZF = 1	JNZ	ZF = 0

Рис. 3.5: Инструкции условной передачи управления по результатам арифметического сравнения cmp a,b

Примечание: термины «выше» («a» от англ. «above») и «ниже» («b» от англ. «below») применимы для сравнения беззнаковых величин (адресов), а термины «больше» («g» от англ. «greater») и «меньше» («l» от англ. «lower») используются при учёте знака числа. Таким образом, мнемонику инструкции JA/JNBE можно расшифровать как «jump if above (переход если выше) / jump if not below equal (переход если не меньше или равно)». Помимо перечисленных команд условного перехода существуют те, которые которые можно использовать после любых команд, меняющих значения флагов.

Мнемокод	Значение флага для осуществления перехода	Мнемокод	Значение флага для осуществления перехода
JZ	ZF = 1	JNZ	ZF = 0

Рис. 3.6: Инструкции условной передачи управления

Мне-мокод	Значение флага для осуществления перехода	Мне-мокод	Значение флага для осуществления перехода
JS	SF = 1	JNS	SF = 0
JC	CF = 1	JNC	CF = 0
JO	OF = 1	JNO	OF = 0
JP	PF = 1	JNP	PF = 0

Рис. 3.7: Инструкции условной передачи управления

В качестве примера рассмотрим фрагмент программы, которая выполняет умножение переменных a и b и если произведение превосходит размер байта, передает управление на метку Error. mov al, a mov bl, b mul bl jc Error

8.2.3. Файл листинга и его структура.

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. Ниже приведён фрагмент файла листинга.

```
10 00000000 B804000000          mov eax,4
11 00000005 BB01000000          mov ebx,1
12 0000000A B9[00000000]        mov ecx,hello
13 0000000F BA0D000000          mov edx,helloLen
14
15 00000014 CD80                int 80h
```

Рис. 3.8: Фрагмент файла листинга

Все ошибки и предупреждения, обнаруженные при ассемблировании, транслятор выводит на экран, и файл листинга не создаётся. Итак, структура листинга: • номер строки — это номер строки файла листинга (нужно помнить, что номер строки в файле листинга может не соответствовать номеру строки в файле с исходным текстом программы); • адрес — это смещение машинного кода от начала текущего сегмента; • машинный код представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности. (например, инструкция int 80h начинается по смещению 00000020 в сегменте кода; далее идёт машинный код, в который ассемблируется инструкция, то есть инструкция

int 80h ассемблируется в CD80 (в шестнадцатеричном представлении); CD80 — это инструкция на машинном языке, вызывающая прерывание ядра); • исходный текст программы — это просто строка исходной программы вме- сте с комментариями (некоторые строки на языке ассемблера, например, строки, содержащие только комментарии, не генерируют никакого ма- шинного кода, и поля «смещение» и «исходный текст программы» в таких строках отсутствуют, однако номер строки им присваивается).

4 Выполнение лабораторной работы

8.3.1. Реализация переходов в NASM.

Первым делом я создал каталог для программам лабораторной работы No 8, перешёл в него и создал файл lab8-1.asm (рис.9).

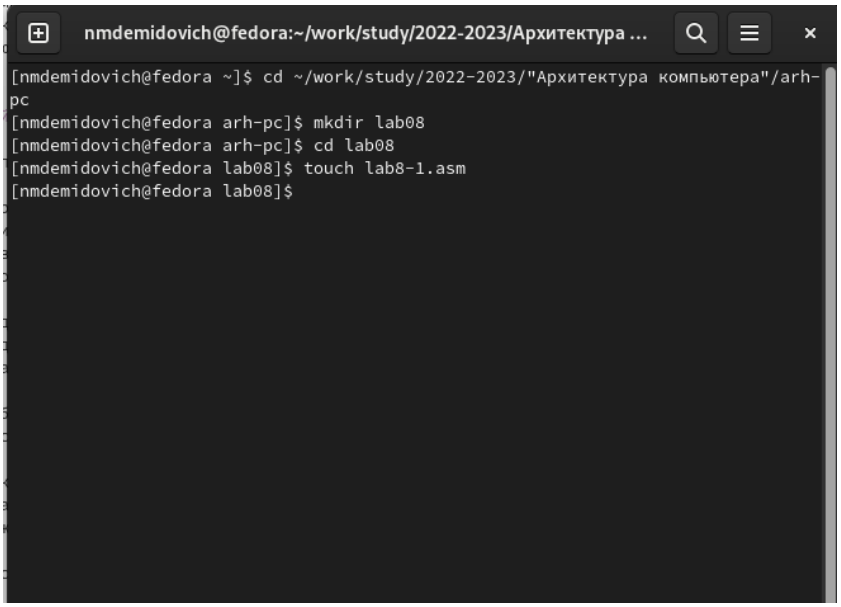

Рис. 4.1: Фрагмент файла листинга

Далее, для корректной работы я скопировал внешний файл в созданный каталог, ввёл текст программы с использованием инструкции jmp в текстовый файл lab8-1.asm, создал объектный файл и проверил работы программы (рис.10-12).

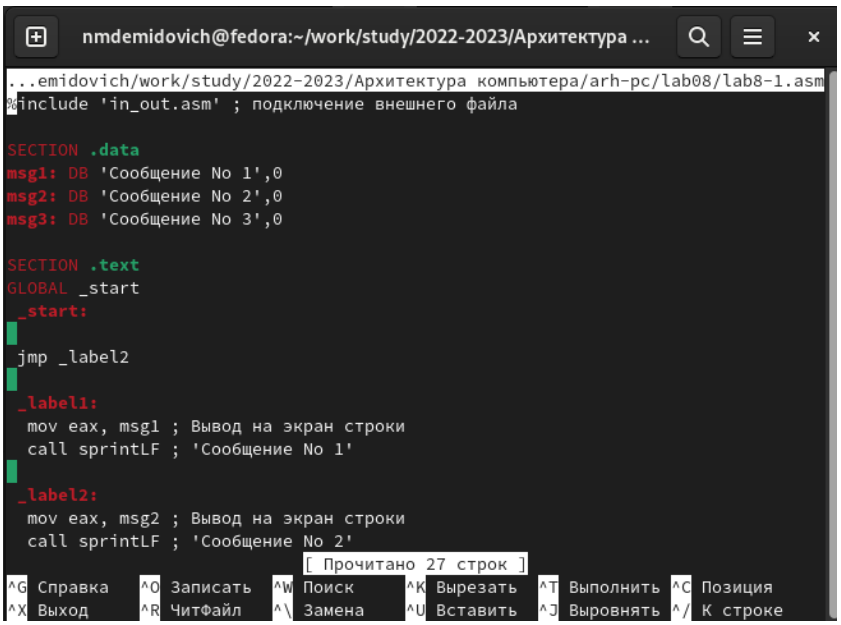

Рис. 4.2: Текст программы в Midnight Commander

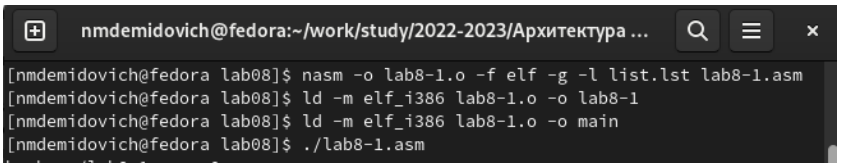

Рис. 4.3: Процесс создания программы

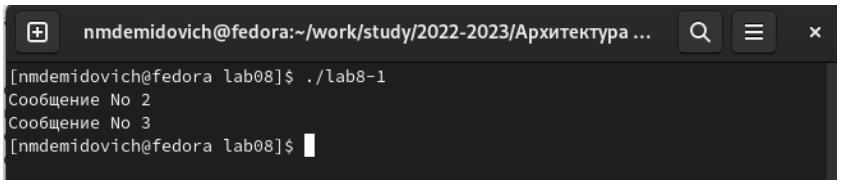

Рис. 4.4: Работа программы

Затем я изменил текст программы в соответствии с предложенным лестинингом и проверил её работу (рис.13-14).

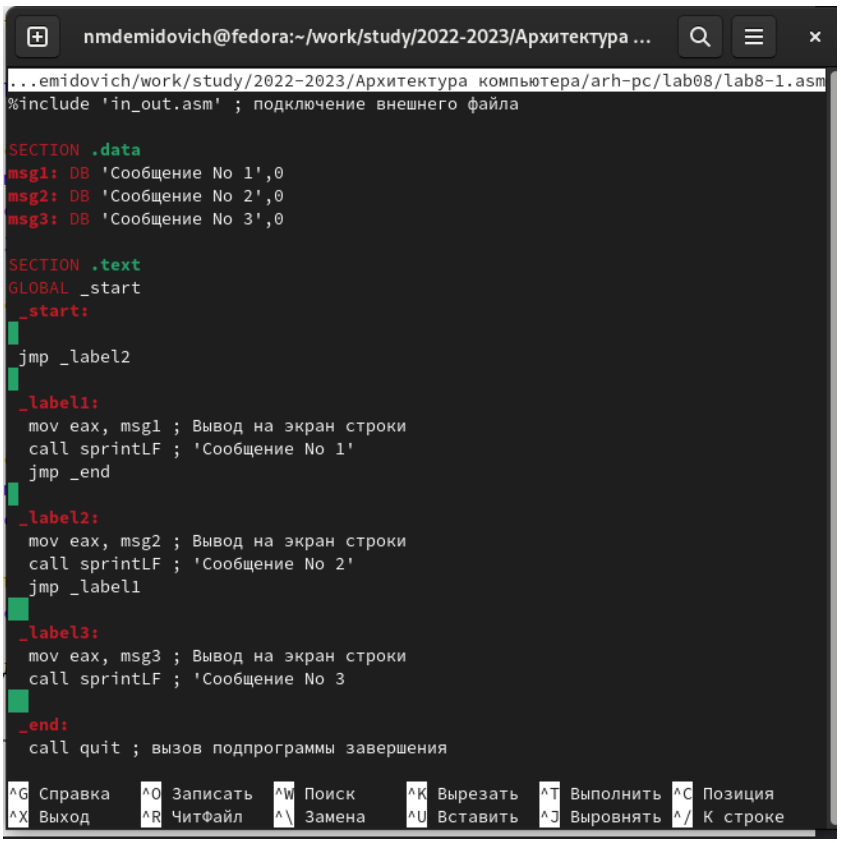

Рис. 4.5: Текст измененной программы в Midnight Commander

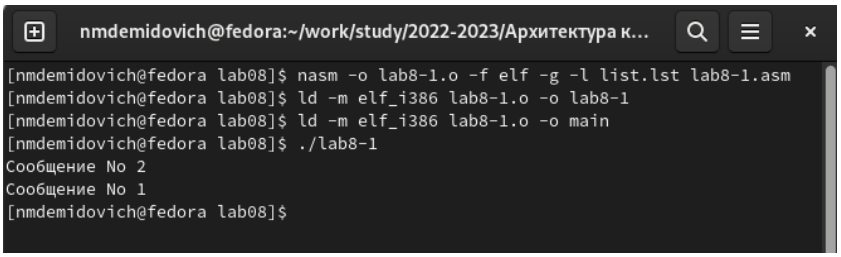

Рис. 4.6: Процесс создания и работа измененной программы

Затем я изменил текст программы таким образом, чтобы сообщения выводились в следующей последовательности: 3, 2, а затем 1 (рис.15).

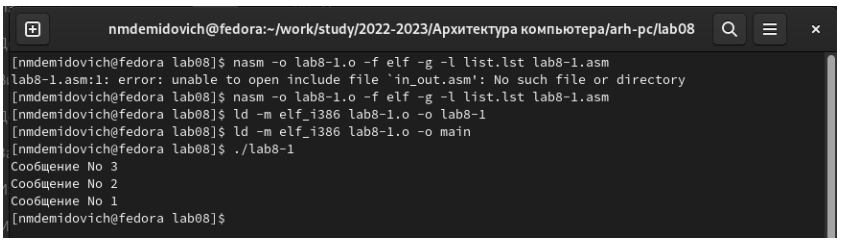

Рис. 4.7: Процесс создания и работа новой программы

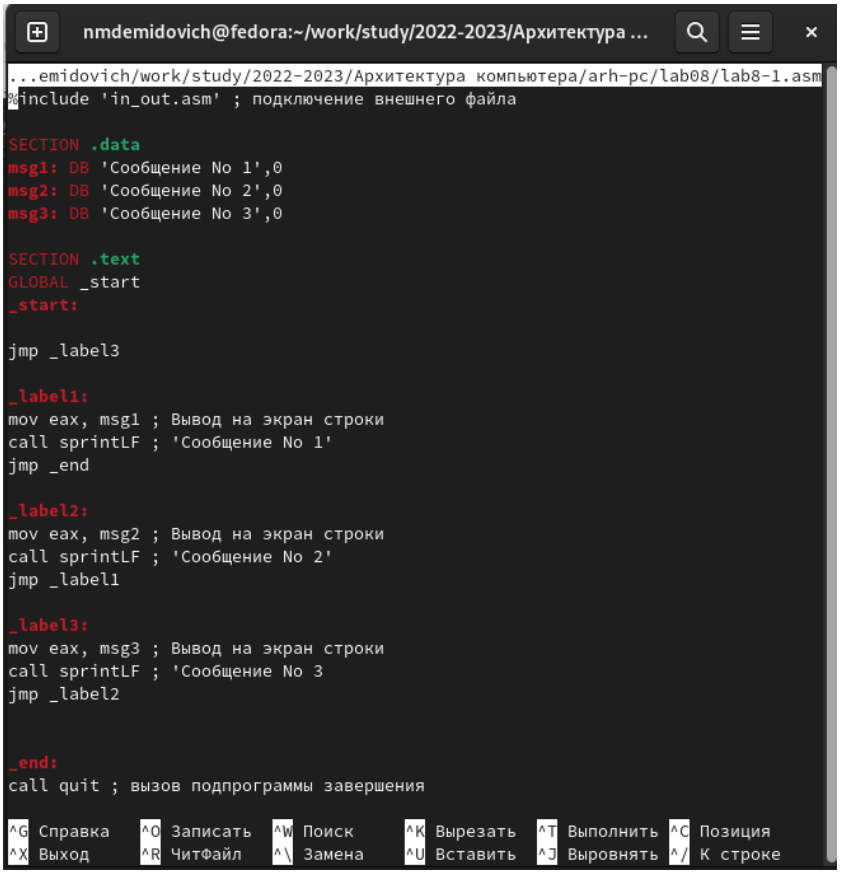

Рис. 4.8: Текст новой программы

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 3.1)

5 Выводы

В результате выполнения данной лабораторной работы, я изучил команды условного и безусловного переходов, приобретел навыки написания программ с использованием переходов и понакомился с назначением и структурой файла листинга.

Список литературы

Лабораторная работа №8 (Архитектура ЭВМ).