Отчет по лабораторной работе №6

Дисциплина: Архитектура компьютера

Серебрякова Дарья Ильинична

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Цель данной лабораторной работы - освоение арифметческих инструкций языка ассемблера NASM

2 Задания

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти.

1. Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: mov ax,bx.
2. Непосредственная адресация – значение операнда задается непосредственно в команде, Например: mov ax,2.
3. Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут

представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в nasm

Командой mkdir создала каталог для программ лабораторной работы № 6, перешла в него и создала файл lab6-1.asm (рис. 4.1).

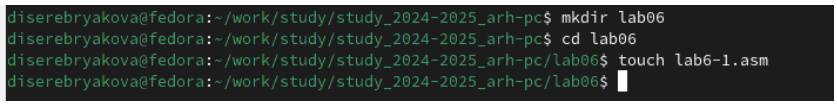

Рис. 4.1: Создание каталога

Копирую в текущий каталог файл in_out.asm с помощью утилиты cp, т.к. он будет использоваться в других программах (рис. 4.2).

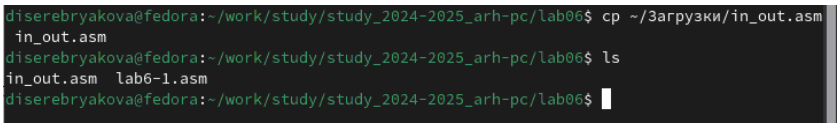

Рис. 4.2: Копия файла в каталог

Открываю созданный файл lab7-1.asm, вставляю в него программу вывода значения регистра eax (рис. 4.3).

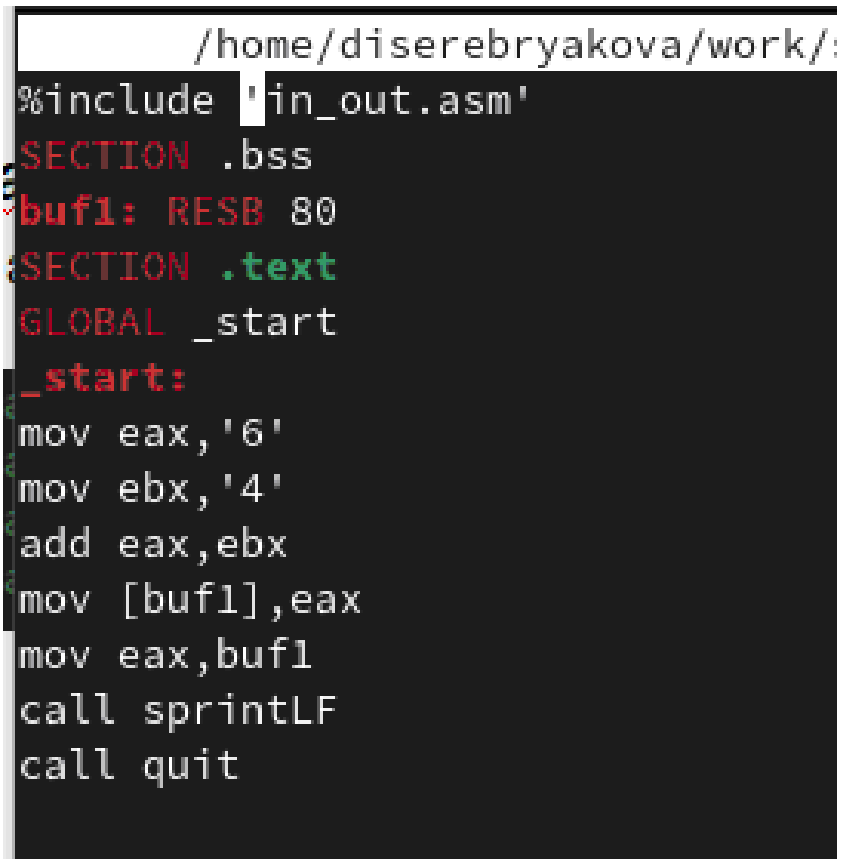

Рис. 4.3: Программа вывода значения eax

Создаю исполняемый файл программы и запускаю его. Вывод программы: символ j. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа 4 – 00110100 (52). Команда add eax,ebx запишет в регистр eax сумму кодов – 01101010 (106), что в свою очередь является кодом символа j (рис. 4.4).

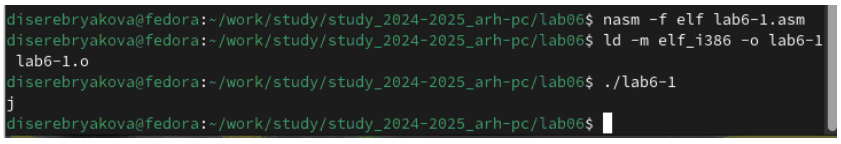

Рис. 4.4: Запуск исполняемого файла

Далее изменю текст программы и вместо символов, запишу в регистры числа 6 и 4 (рис. 4.5).

```
/home/diserebryakova/work/
%include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintLF
call quit
```

Рис. 4.5: Редактирование текста программы

Создаю исполняемый файл и запускаю его. Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран (рис. 4.6).

```
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/lab06$ nasm -f elf lab6-1.asm
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/lab06$ ld -m elf_i386 -o lab6-1
 lab6-1.o
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/lab06$ ./lab6-1


diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/lab06$
```

Рис. 4.6: Запуск новой программы

Командой touch создаю файл lab6-2.asm в каталоге lab06 и ввожу в него текст программы из предложенного листинга 6.2 (рис. 4.7).

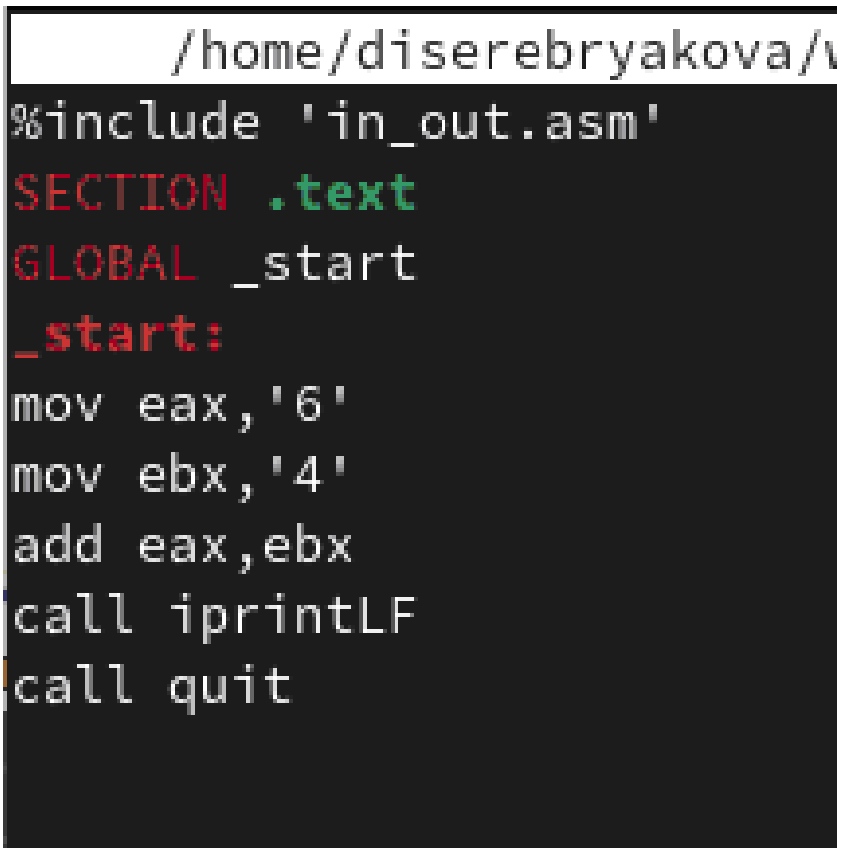

Рис. 4.7: Создание программы по предложенному листингу

Создаю исполняемый файл и запускаю его. В результате работы программы получила число 106. В данном случае, как и в первом, команда add складывает коды символов ‘6’ и ‘4’ (54+52=106). Однако, в отличии от программы из листинга 6.1, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число (рис. 4.8).

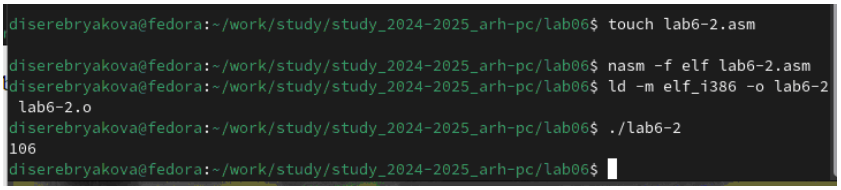

Рис. 4.8: Сравнение команд

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4. Создаю исполняемый файл и запускаю его. Теперь программа складывает не

соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10 (рис. 4.9).

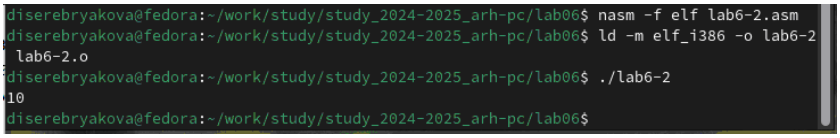

Рис. 4.9: Запуск программы

Заменяю в тексте программы функцию iprintLF на iprint. Создаю и запускаю новый исполняемый файл. Вывод изменился, потому что iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF (рис. 4.10).

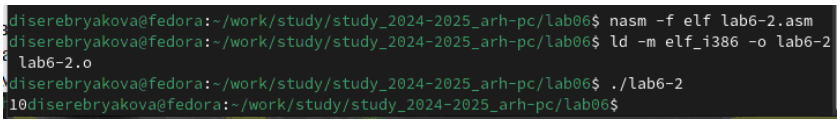

Рис. 4.10: Сравнение вариантов вывода

4.2 Выполнение арифметических операций в nasm

Командой touch создала файл lab6-3.asm (рис. 4.11).

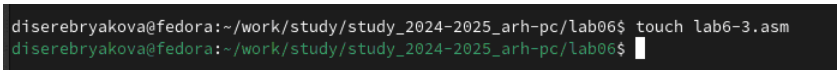

Рис. 4.11: Создание пустого файла

Изучила текст программы листинга 6.3 и ввела его в lab6-3.asm (рис. 4.12).

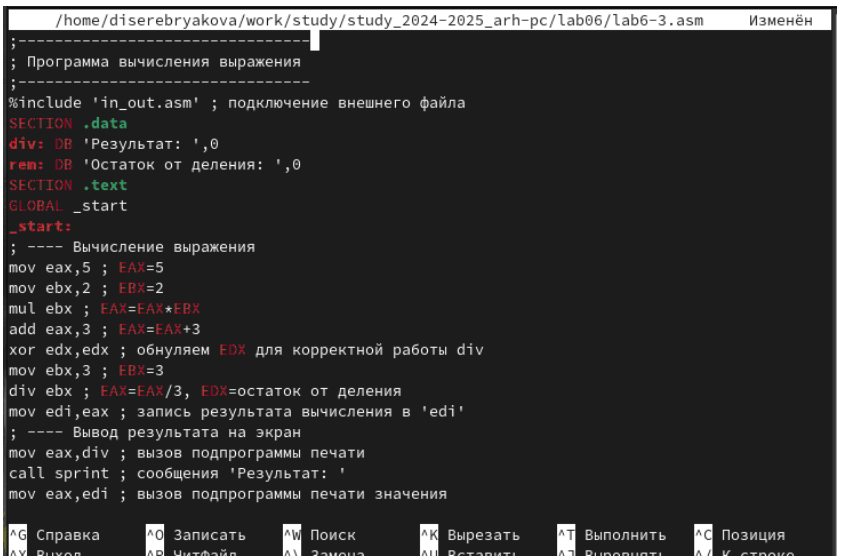

Рис. 4.12: Текст программы из предложенного листинга

Создаю исполняемый файл и запускаю его (рис. 4.13).

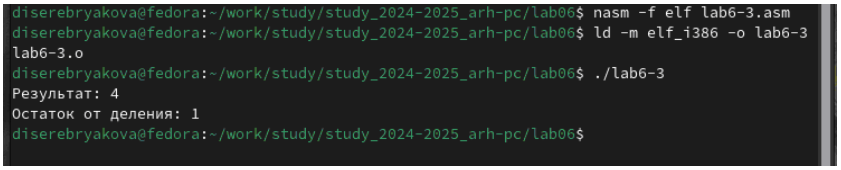

Рис. 4.13: Создание и запуск исполняемого файла

Изменяю текст программы для вычисления выражения f(x) = (4 6 + 2)/5 (рис. 4.14).

```
/home/diserebryakova/work/study/study_2024-2025_arh-pc/lab06/
;-----------------------------
; Программа вычисления выражения
;-----------------------------
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,4 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=5
div ebx ; EAX=EAX/5, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
```

Рис. 4.14: Программа для вычисления предложенного выражения

Создаю исполняемый файл и запускаю его. Результат вычислений изменился (рис. 4.15).

```
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/lab06$ nasm -f elf lab6-3.asm
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/lab06$ ld -m elf_i386 -o lab6-3
lab6-3.o
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/lab06$
```

Рис. 4.15: Запуск измененной программы

Командой touch создаю файл variant.asm в каталоге lab06. Внимательно изучаю текст программы из листинга 6.4 и ввожу его в только что созданный файл (рис. 4.16).

```
     /home/diserebryakova/work/study/study_2024-2025_arh-pc/lab06/varia
;-----------------------------
; Программа вычисления варианта
;-----------------------------
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax,x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
xor edx,edx
mov ebx,20
div ebx
inc edx
```

Рис. 4.16: Новая программа из предложенного листинга

Создаю исполняемый файл и запускаю его. Ввожу номер своего студенческого билета и вижу, что мой вариант 14 (рис. 4.17).

```
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/lab06$ nasm -f elf variant.asm
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/lab06$ ld -m elf_i386 -o variant
  variant.o
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/lab06$ ./variant
Введите № студенческого билета:
1132246733
Ваш вариант: 14
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/lab06$
```

Рис. 4.17: Нахождение варианта для выполнения задания

4.3 Ответы на вопросы

1. За вывод на экран сообщения ‘Ваш вариант:’ отвечают следующие строки (рис. 4.18).

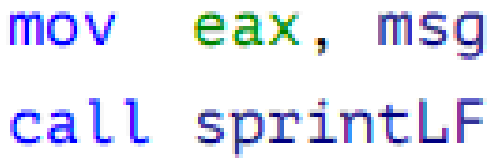

Рис. 4.18: Вывод сообщения на экран

2. Указанные инструкции отвечают за то, чтобы положить адрес вводимой строки x в регистр ecx, далее записать в регистр edx длину вводимой строки (80) и call sread - вызвать подпрограммы из внешнего файла, обеспечивающую ввод сообщения с клавиатуры

3. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax

4. За вычисление варианта отвечают следующие строки: (рис. 4.19).

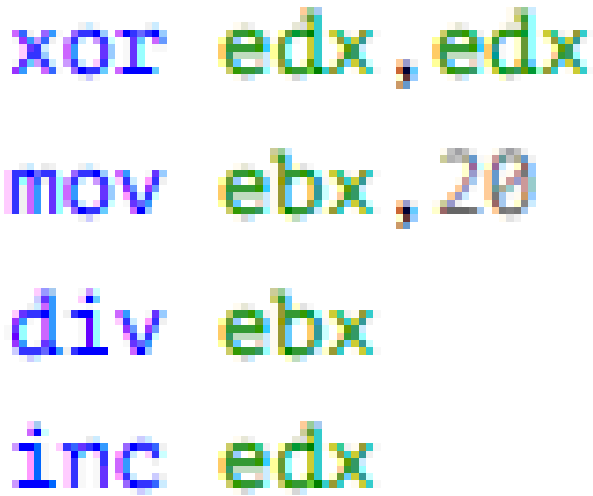

Рис. 4.19: Вычисление варианта

5. При выполнении инструкции div ebx остаток от деления записывается в регистр edx

6. Инструкция inc edx увеличивает значение регистра edx на 1

7. За вывод результатов вычислений на экран отвечают следующие строки: (рис. 4.20).

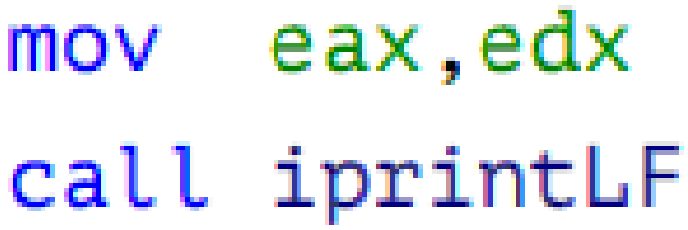

Рис. 4.20: Вывод результатов вычисления

5 Выполенение заданий для самостоятельной работы

Командой touch создаю файл lab6-4.asm для выполнения задания (рис. 5.1).

```
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/lab06$ touch lab6-4.asm
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/lab06$
```

Рис. 5.1: Создание файла для выполнения задания

Захожу в только что созданный файл и пишу в нем программу для вычисления заданного выражения (мой вариант – 14) (рис. 5.2).

```
/home/diserebryakova/work/study/study_2024-2025_arh-pc/lab06/lab6-4.asm
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax,x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx, 2 ;
div ebx ;
add eax, 8; eax = eax + 8 = x/2 + 8
mov ebx,3
mul ebx; EAX=EAX*EBX = (x/2+8)*3
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 5.2: Программа для вычисления выражения

Создаю исполняемый файл и запускаю его. Проверяю работу программы на двух предложенных значениях. Результат вычислений программы совпадает с результатом вычисления выражения вручную, значит программа написана верно. Задание выполнено (рис. 5.3).

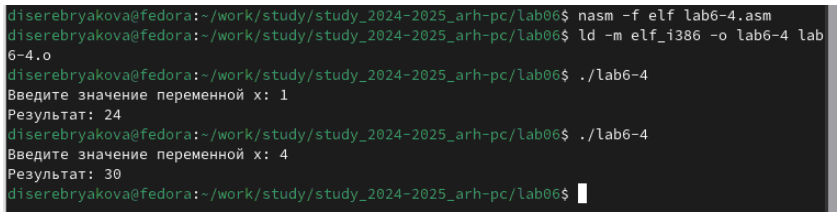

Рис. 5.3: Проверка работы программы

6 Выводы

При выполнении данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM

Список литературы

1. Лабораторная работа 6