РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

дисциплина: Архитектура компьютера

Студент: Павличенко Родион Андреевич

Группа: НПИбд-02-24

МОСКВА

2024 г.

Цель работы

Цель работы : Познакомиться с базовыми инструкциями языка Ассемблер, отвечающими за основные арифметические операции

Выполнение лабораторной работы

Для начала выполнения лабораторной работы необходимо создать папку рабочего каталога и файл lab6-1.asm :

```
rapavlichenko@rapavlichenko:~$ mkdir ~/work/arch-pc/lab06
rapavlichenko@rapavlichenko:~$ cd ~/work/arch-pc/lab06
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ touch lab6-1.asm
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$
```

Создание рабочей директории и файла lab6-1.nasm для записи кода на языке Ассемблера

После этого, для более комфортной работы, запустим Midnight commanderьс:

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ mc
```

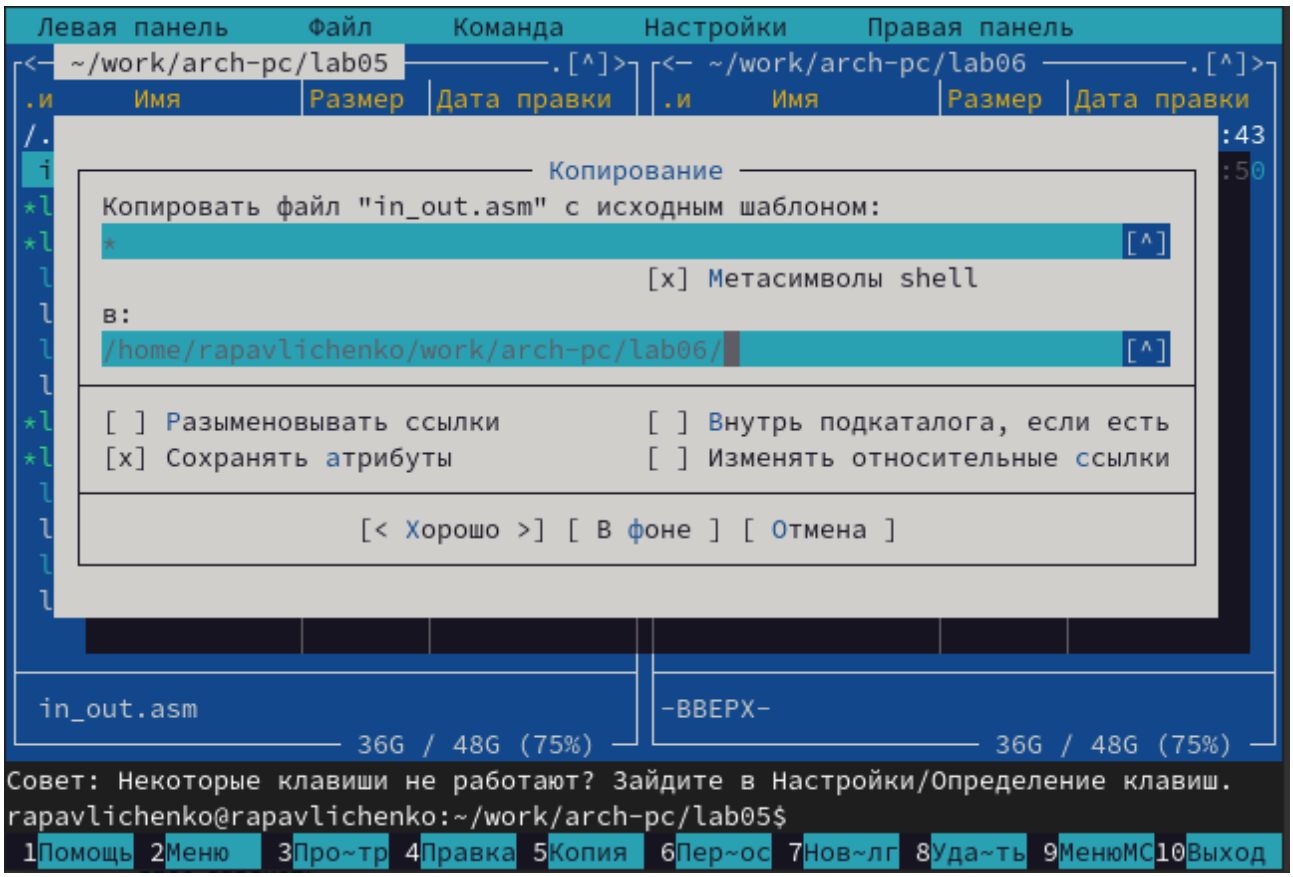
Запуск Midnight commander

Вставим в наш созданный файл код из листинга 6.1 с помощью команды F4 в MC :

```
  GNU nano 7.2    /home/rapavlichenko/work/arch-pc/lab06/lab6-1.asm    Изменён
%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
```

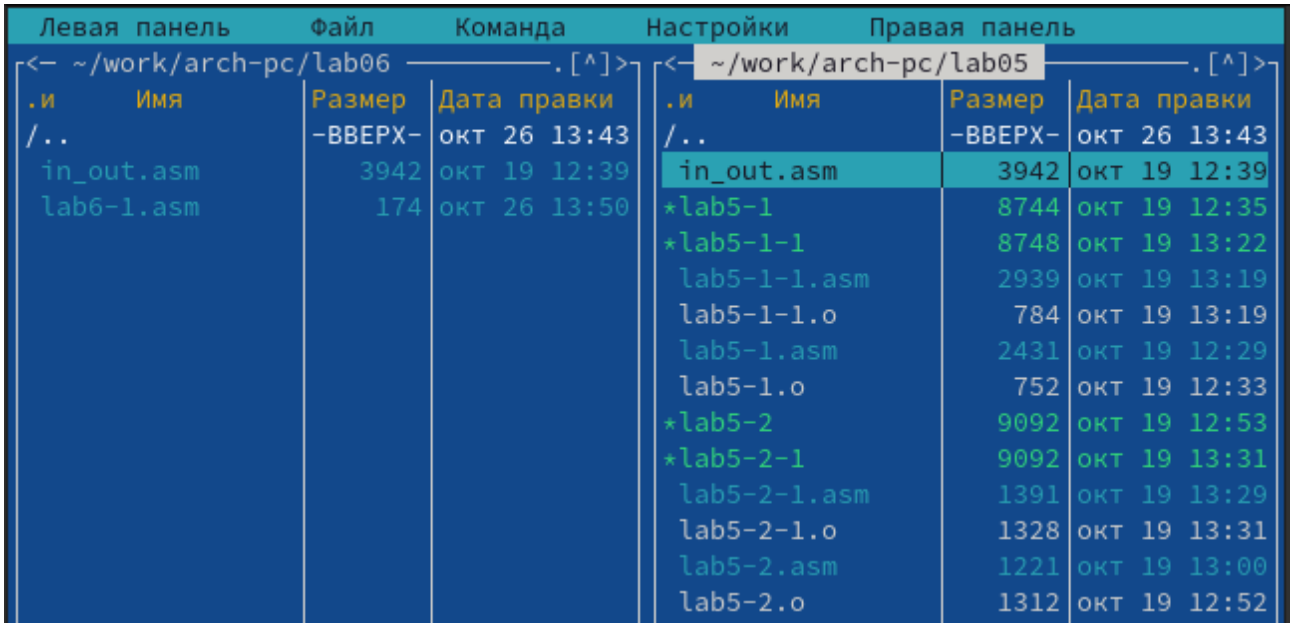
Запись кода из листинга в файл lab6-1.asm

Перед сборкой файла стоит учесть, что он использует сторонний файл in_out.asm. С помощью команды F5 скопируем его из каталога пятой лабораторной работы:



Копирование файла in_out.asm в рабочую директорию

Так будет выглядеть наша рабочая директория :



Вид каталога после перенесения файла in_out.asm

Теперь соберём наш файл в исполняемое приложение уже знакомыми инструментами, nasm и ld:

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
```

Сборка исполняемого файла из lab6-1.asm

Теперь запустим файл и посмотрим на результат:

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ./lab6-1
j
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$
```

Запуск исполняемого файла и результат вывода

Нам выводит символ j, однако это неправильный вывод. Наша цель - сложить 6 и 4, и получить в выводе число 10. Попробуем изменить наш файл:

```
 GNU nano 7.2          /home/rapavlichenko/work/arch-pc/lab06/lab6-1.asm
%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
```

Редактирование файла

Мы убрали кавычки у цифр, и теперь мы складываем уже не символы “6” и “4” (когда мы складываем символы, мы складываем их коды ASCII), а числа. Теперь попробуем собрать исполняемый исполняемый файл также, как собирали до этого, и запустим:

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ./lab6-1


rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$
```

Запуск исполняемого файла и результат вывода

Мы видим, что ничего не вывелось. Но так ли это? Когда мы вызываем команду sprintLF, она выводит не число 10, а символ с номером 10. Посмотрим на таблицу ASCII и увидим, что символ под номером 10 это символ перевода строки. Именно поэтому мы его не видим, мы видим просто новую строку. Теперь создадим второй файл под названием lab6-2.asm:

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$
```

Создание второго файла: lab6-2.asm

Теперь вставим в него код из листинга 6.2 :

```
 GNU nano 7.2          /home/rapavlichenko/work/arch-pc/lab06/lab6-2.asm
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

Запись кода из листинга в файл lab6-2.asm

Как мы видим, основное отличие заключается в том, что вместо sprintLF используется iprintLF. Соберём файл и запустим его, чтобы посмотреть, как изменится вывод :

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ./lab6-2
106
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$
```

Запуск исполняемого файла и результат вывода

Мы видим число 106. Так как цифры в коде указаны в кавычках, мы складываем их коды (54 и 52 в сумме дают 106). Теперь программа способна вывести число, а не символ ASCII с соответствующим номером. Теперь, если мы уберём кавычки у цифр, программа должна вывести 10. Убедимся в этом, сделав соответствующие изменения в коде:

```
 GNU nano 7.2          /home/rapavlichenko/work/arch-pc/lab06/lab6-2.asm
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Изменение файла lab6-2.asm

Соберём программу и запустим её:

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ./lab6-2
10
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$
```

Сборка исполняемого файла и результат работы программы

Как видим, программа действительно вывела число 10. Кроме операции iprintLF в файле in_out.asm есть операция iprint. Посмотрим, чем они отличаются. Заменим в коде iprintLF на iprint:

```
  GNU nano 7.2          /home/rapavlichenko/work/arch-pc/lab06/lab6-2.asm
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
```

Редактирование файла lab6-2.asm

Попробуем собрать программу и запустить её:

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ./lab6-2
10rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$
```

Сборка и результат работы отредактированного файла

Как видим, операция iprint не переносит на следующую строку, в отличие от iprintLF. В этом их разница. Теперь создадим третий файл:

```
10rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$
```

Создание третьего файла: lab6-3.asm

Он должен выводить значение функции (5*2+3)/3. Для этого вставим код из файла листинга 6.3:

```
 GNU nano 7.2          /home/rapavlichenko/work/arch-pc/lab06/lab6-3.asm
;------------------------------
; 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,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
```

Вставка кода из листинга в созданный ранее файл

Попробуем запустить этот программу, предварительно её собрав:

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$
```

Сборка файла lab6-3.asm и результат его работы

Полученный результат совпадает с результатом, указанным в лабораторной работе. Теперь изменим файл так, чтобы он вычислял значение выражения (4*6+2)/5. Для этого в коде заменим число 5 на 4, число 2 на 6, число 3 на 2, и второе число 3 на 5:

```
 GNU nano 7.2          /home/rapavlichenko/work/arch-pc/lab06/lab6-3.asm
;------------------------------
; 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
```

Редактирование файла lab6-3.asm

Соберём программу и запустим её:

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$
```

Повторная сборка уже изменённого файла lab6-3.asm и результат его работы

Пересчитав значение выражения вручную, убеждаемся, что вывод корректный. Теперь создадим файл variant.asm для вычисления варианта самостоятельной работы :

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$
```

Создание файла variant.asm для вычисления варианта для самостоятельной работы

Вставим в файл код из листинга 6.4, который вычисляет номер варианта по формуле (s mod 20) + 1, где s - номер студенческого билета:

```
  GNU nano 7.2          /home/rapavlichenko/work/arch-pc/lab06/variant.asm
------------------------------
; 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
```

Вставка кода из листинга в файл variant.asm

Соберём и запустим программу, указав номер студенческого билета. В моём случае это 112246838 :

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ nasm -f elf variant.asm
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246838
Ваш вариант: 19
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$
```

Сборка и запуск программы, а также результат выполнения

Программа вывела число 19. Действительно, ведь остаток от деления числа 112246838 на 20 равен 18. 1 + 8 = 19, соответственно.

Разберём работу кода, ответив на предложенные в лабораторной работе вопросы:

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?

За это отвечает 25-ая строчка call sprint, перед которой идёт строка mov eax,rem, которая перемещает строку с фразой в регистр eax, из которого мы считываем данные для вывода.

2. Для чего используется следующие инструкции?

mov ecx, x

mov edx, 80

call sread

Эти строки используются для того, чтобы записать данные в переменную x.

3. Для чего используется инструкция “call atoi”?

Для преобразования ASCII кода в число.

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

Напрямую за вычисление отвечают следующие строки:

div ebx

inc edx

Первая делит число x в регистре eax на значение регистра ebx (в нашем случае 20), а вторая прибавляет к значению регистра edx (куда сохранился остаток от деления в прошлой операции) единицу.

5. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?

Как уже было сказано в ответе на предыдущий вопрос, в регистр edx.

6. Для чего используется инструкция “inc edx”?

Для увеличения значения регистра edx на единицу.

7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?

За это отвечают строки:

mov eax,edx

call iprintLF

Первая строка переносит значение регистра edx в eax, а вторая вызывает операцию вывода значения регистра eax на экран.

Выполнение задания для самостоятельной работы

Теперь в качестве самостоятельной работы напишем код программы для вычисления выражения в варианте 2: (12x + 3)*5 . В предварительно созданном файле task9.asm впишем следующий код :

```
 GNU nano 7.2          /home/rapavlichenko/work/arch-pc/lab06/task9.asm
;------------------------------
; Программа вычисления варианта
;------------------------------
%include 'in_out.asm'
SECTION .data
msg: DB 'Выражение для вычисления: (12 * X + 3) * 5', 0
rem: DB 'Введите X: ', 0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintLF
mov eax, rem
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
```

Код требуемой программы

Попробуем собрать нашу программу и запустим код, указав в качестве x предложенные в лабораторной работе значения:

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ nasm -f elf task9.asm
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ld -m elf_i386 -o task9 task9.o
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ./task9
Выражение для вычисления: (12 * X + 3) * 5
Введите X: 1
75
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$ ./task9
Выражение для вычисления: (12 * X + 3) * 5
Введите X: 6
375
rapavlichenko@rapavlichenko:~/work/arch-pc/lab06$
```

Сборка исполняемого файла, запуск программы и проверка её корректной работы

Как видим, программа выводит правильные значения выражения.

Выводы

В результате выполнения лабораторной работы было получено представление о том, какие арифметические операции есть в языке Ассемблера, и как они работают. Были написаны программы, использующие в себе операции сложения, вычитания, умножения и деления.