Отчёт по лабораторной работе 6

Архитектура компьютера

Ходжамедов Давуд НБИбд-02-23

Содержание

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

# 2 Выполнение лабораторной работы

1. Создал каталог для программам лабораторной работы № 6, перешел в него и создал файл lab6-1.asm.
2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения, записанные в регистр eax.

В данной программе в регистр eax записывается символ 6 (mov eax,‘6’), в регистр ebx символ 4 (mov ebx,‘4’). Далее к значению в регистре eax прибавляем значение регистра ebx (add eax,ebx, результат сложения запишется в регистр eax). Далее выводим результат. Так как для работы функции sprintLF в регистр eax должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра eax в переменную buf1 (mov [buf1],eax), а затем запишем адрес переменной buf1 в регистр eax (mov eax,buf1) и вызовем функцию sprintLF.

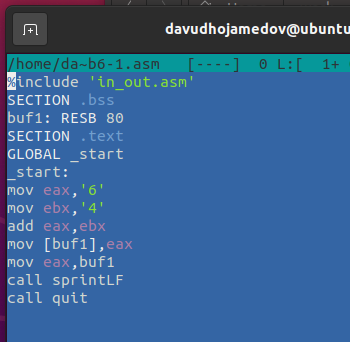


Figure 1: Программа в файле lab6-1.asm

Привожу код программы в отчете

%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

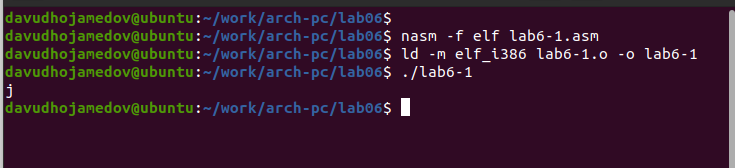


Figure 2: Запуск программы lab6-1.asm

В данном случае при выводе значения регистра eax мы ожидаем увидеть число 10. Однако результатом будет 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.

1. Далее изменяю текст программы и вместо символов, запишем в регистры числа.

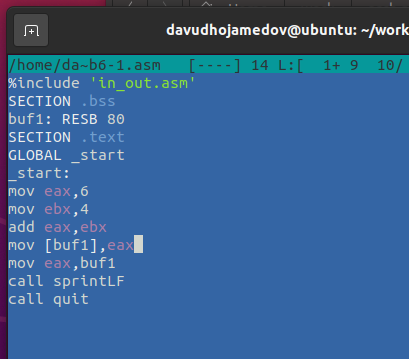


Figure 3: Программа в файле lab6-1.asm

Привожу код программы в отчете

%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

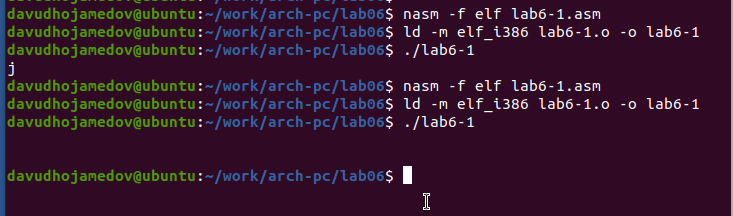


Figure 4: Запуск программы lab6-1.asm

Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Это символ конца строки (возврат каретки). В консоле он не отображается, но добавляет пустую строку.

1. Как отмечалось выше, для работы с числами в файле in\_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразовал текст программы с использованием этих функций.

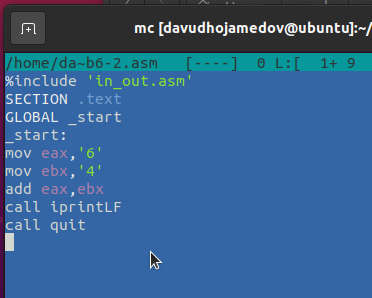


Figure 5: Программа в файле lab6-2.asm

Привожу код программы в отчете

%include 'in\_out.asm'  
SECTION .text  
GLOBAL \_start  
\_start:  
mov eax,'6'  
mov ebx,'4'  
add eax,ebx  
call iprintLF  
call quit

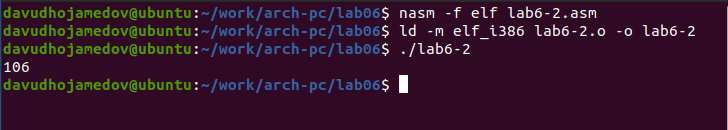


Figure 6: Запуск программы lab6-2.asm

В результате работы программы мы получим число 106. В данном случае, как и в первом, команда add складывает коды символов ‘6’ и ‘4’ (54+52=106). Однако, в отличии от прошлой программы, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

1. Аналогично предыдущему примеру изменим символы на числа.

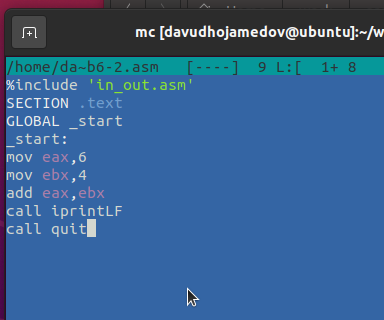


Figure 7: Программа в файле lab6-2.asm

Привожу код программы в отчете

%include 'in\_out.asm'  
SECTION .text  
GLOBAL \_start  
\_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprintLF  
call quit

Функция iprintLF позволяет вывести число и операндами были числа (а не коды символов). Поэтому получаем число 10.

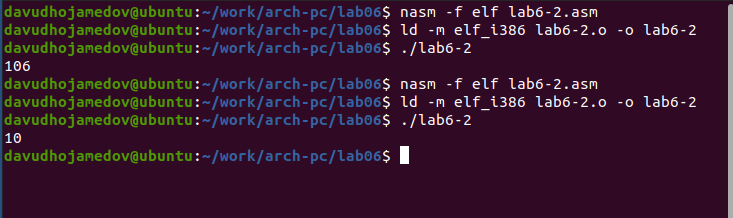


Figure 8: Запуск программы lab6-2.asm

Заменил функцию iprintLF на iprint. Создал исполняемый файл и запустил его. Вывод отличается тем, что нет переноса строки.

Привожу код программы в отчете

%include 'in\_out.asm'  
SECTION .text  
GLOBAL \_start  
\_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprint  
call quit

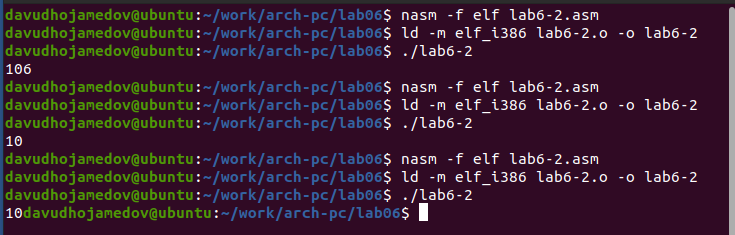


Figure 9: Запуск программы lab6-2.asm

1. В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения

* .

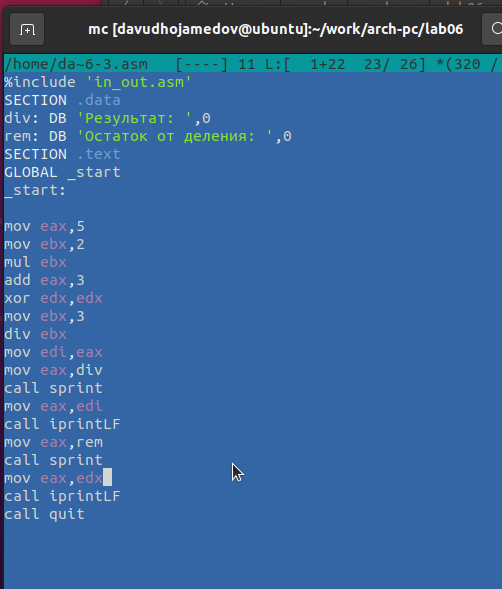


Figure 10: Программа в файле lab6-3.asm

Также размещаю код программы в отчете.

%include 'in\_out.asm'  
SECTION .data  
div: DB 'Результат: ',0  
rem: DB 'Остаток от деления: ',0  
SECTION .text  
GLOBAL \_start  
\_start:  
  
mov eax,5  
mov ebx,2  
mul ebx  
add eax,3  
xor edx,edx  
mov ebx,3  
div ebx  
mov edi,eax  
mov eax,div  
call sprint  
mov eax,edi  
call iprintLF  
mov eax,rem  
call sprint  
mov eax,edx  
call iprintLF  
call quit

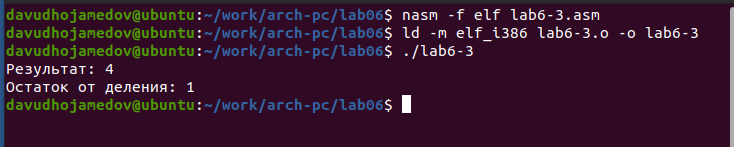


Figure 11: Запуск программы lab6-3.asm

Изменил текст программы для вычисления выражения . Создал исполняемый файл и проверил его работу.

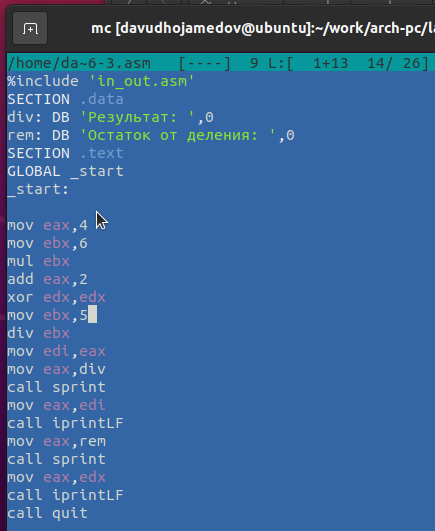


Figure 12: Программа в файле lab6-3.asm

Также размещаю код программы в отчете.

%include 'in\_out.asm'  
SECTION .data  
div: DB 'Результат: ',0  
rem: DB 'Остаток от деления: ',0  
SECTION .text  
GLOBAL \_start  
\_start:  
  
mov eax,4  
mov ebx,6  
mul ebx  
add eax,2  
xor edx,edx  
mov ebx,5  
div ebx  
mov edi,eax  
mov eax,div  
call sprint  
mov eax,edi  
call iprintLF  
mov eax,rem  
call sprint  
mov eax,edx  
call iprintLF  
call quit

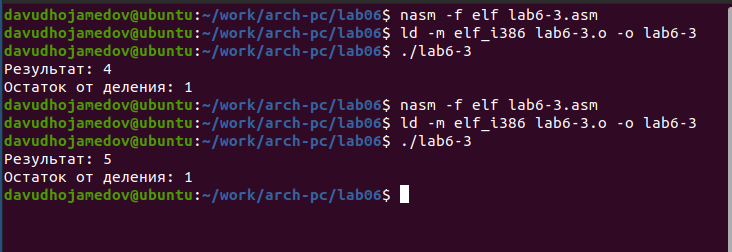


Figure 13: Запуск программы lab6-3.asm

1. В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета.

В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры. Как отмечалось выше ввод с клавиатуры осуществляется в символьном виде и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого может быть использована функция atoi из файла in\_out.asm.

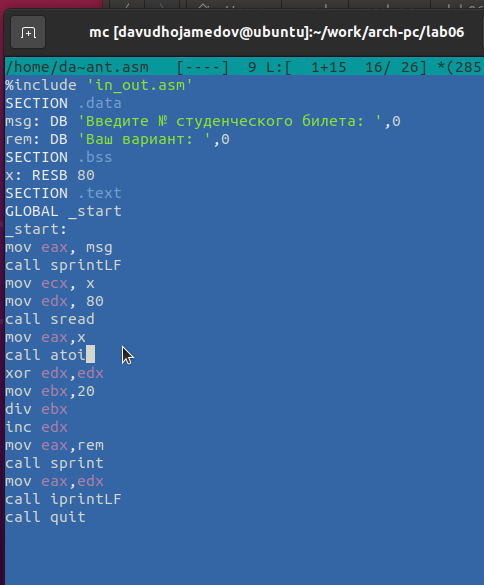


Figure 14: Программа в файле variant.asm

Также размещаю код программы в отчете.

%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  
call atoi  
xor edx,edx  
mov ebx,20  
div ebx  
inc edx  
mov eax,rem  
call sprint  
mov eax,edx  
call iprintLF  
call quit

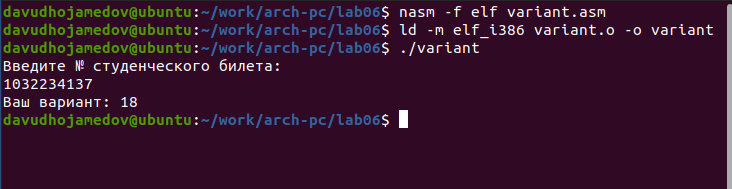


Figure 15: Запуск программы variant.asm

ответы на вопросы

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

* mov eax,rem – перекладывает в регистр значение переменной с фразой ‘Ваш вариант:’
* call sprint – вызов подпрограммы вывода строки

1. Для чего используется следующие инструкции?

nasm   
mov ecx, x   
mov edx, 80   
call sread

Считывает значение студбилета в переменную Х из консоли

1. Для чего используется инструкция “call atoi”?

Эта подпрограмма переводит введенные символы в числовой формат.

1. Какие строки листинга отвечают за вычисления варианта?

xor edx,edx  
mov ebx,20  
div ebx  
inc edx

Здесь происходит деление номера студ билета на 20. В регистре edx хранится остаток, к нему прибавляется 1.

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

регистр edx

1. Для чего используется инструкция “inc edx”?

по формуле вычисления варианта нужно прибавить единицу

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

mov eax,edx – результат перекладывается в регистр eax

call iprintLF – вызов подпрограммы вывода

1. Написать программу вычисления выражения y = f(x). Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x, вычислять заданное выражение в зависимости от введенного x, выводить результат вычислений. Вид функции f(x) выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений x1 и x2 из 6.3.

Получили вариант 18 -

для



Figure 16: Программа в файле task.asm

Также размещаю код программы в отчете.

%include 'in\_out.asm'  
SECTION .data  
msg: DB 'Введите X ',0  
rem: DB 'выражение = : ',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`  
add eax,10  
mov ebx,3  
mul ebx  
sub eax,20  
mov eax,rem  
call sprint  
mov eax,ebx  
call iprintLF  
call quit

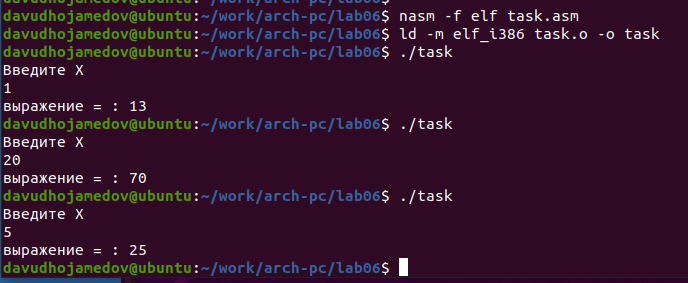


Figure 17: Запуск программы task.asm

# 3 Выводы

Изучили работу с арифметическими операциями.