Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Шония Ника Гигловна

Содержание

Список таблиц

Список иллюстраций

Цель работы

Цель данной лабораторной работы - освоение арифметческих инструкций языка ассемблера NASM. # Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM

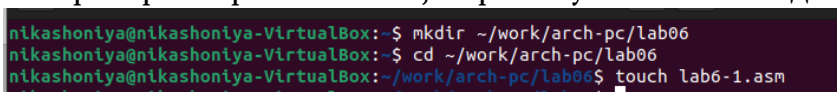
Теоретическое введение

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.

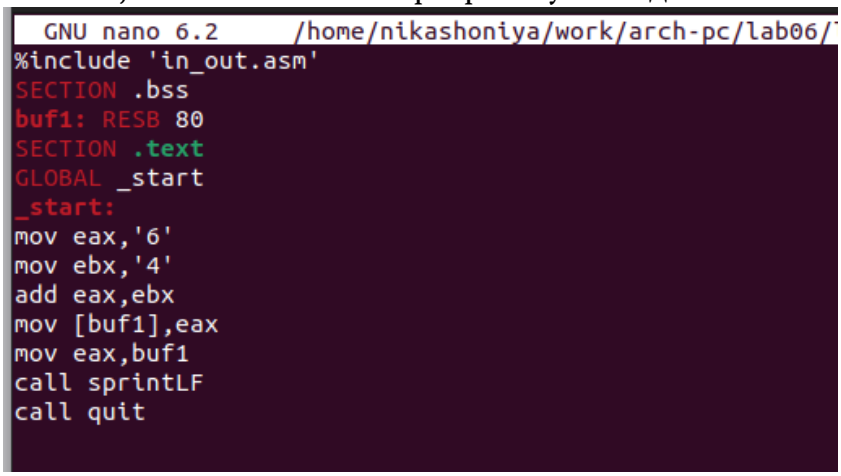
Регистровая адресация - операнды хранятся в регистрах и в команде используются им
Непосредственная адресация - значение операнда задается непосредственно в команде
Адресация памяти - операнд задает адрес в памяти. В команде указывается символиче

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 # Выполнение лабораторной работы

1. Символьные и численные данные в NASM Создаю каталог для программам лабораторной работы № 6, перехожу в него и создаю файл lab6-1.asm

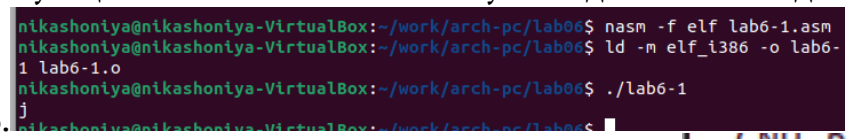


Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения

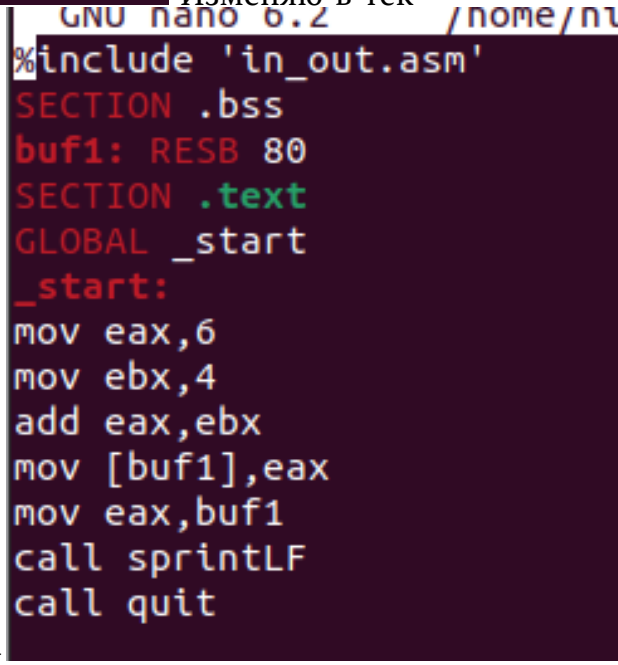


регистра eax

Создаю исполняемый файл программы и запускаю его. Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.



Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4



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

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

nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab06$
```

Создаю новый файл lab6-2.asm с помощью утилиты touch

```
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab06/lab6-2.asm
```

Ввожу в файл текст другойпрограммы для вывода значения регистра eax

```
 GNU nano 6.2      /home/nikashoniya/work/arch-pc
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

Создаю и запускаю исполняемый файл lab6-2. Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.

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

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4

```
 GNU nano 6.2          /home/nik
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Создаю и запускаю новый исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

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

Заменяю в тексте программы функцию iprintLF на iprint

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

Создаю и запускаю новый исполняемый файл. Вывод не изменился, потому что символ переноса строки не отображался, когда программа исполнялась с функцией iprintLF, а iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF.

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

2. Выполнение арифметических операций в NASM Создаю файл lab6-3.asm с помощью утилиты touch

```
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab06$ touch ~/work/arch-pc/la
b06/lab6-3.asm
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab06$
```

Ввожу в созданный файл текст программы для вычисления значения выражения f(x) = (5 * 2 + 3)/3

```
GNU nano 6.2     /home/nikashoniya/work/arch-pc/lab06/lab6-3.asm
%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
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
```

Создаю исполняемый файл и запускаю его

```
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab06$ nasm -f el
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_
3 lab6-3.o
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab06$
```

Изменяю программу так, чтобы она вычисляла значение выражения f(x) = (4 * 6 + 2)/5

```
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,4 ; EAX=5
mov ebx,6 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
```

Создаю и запускаю новый исполняемый файл. Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

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

Создаю файл variant.asm с помощью утилиты touch

```
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab
b06/variant.asm
```

Ввожу в файл текст программы для вычисления варианта задания по номеру

```
%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
```

студенческого билета

Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 11

```
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/la
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/la
t variant.o
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/la
Введите № студенческого билета:
1132236110
Ваш вариант: 11
```

Ответы на вопросы:

3. За вывод сообщения “Ваш вариант” отвечают строки кода:mov eax,rem call sprint
4. Инструкция mov ecx, x используется, чтобы положить адрес вводимой строки x в регистр ecx mov edx, 80 - запись в регистр edx длины вводимой строки call sread - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры
5. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax
6. За вычисления варианта отвечают строки: xor edx,edx ; обнуление edx для корректной работы div mov ebx,20 ; ebx = 20 div ebx ; eax = eax/20, edx - остаток от деления inc edx ; edx = edx + 1
7. При выполнении инструкции div ebx остаток от деления записывается в регистр edx
8. Инструкция inc edx увеличивает значение регистра edx на 1
9. За вывод на экран результатов вычислений отвечают строки: mov eax,edx

```
call iprintLF
```

Выводы

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

Список литературы

Архитектура ЭВМ Мой репозиторий: https://github.com/NikaShoniya/study_2023-2024_arch-pc