Отчет по лабораторной работе №6

Архитектура компьютеров

Арутюнян Эрик Левонович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM

2 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес опе- ранда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации. Существует три основных способа адресации: • Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: mov ax,bx. • Непосредственная адресация – значение операнда задается непосредственно в ко- манде, Например: mov ax,2. • Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию. Например, определим переменную intg DD 3 – это означает, что задается область памяти размером 4 байта, адрес которой обозначен меткой intg. В таком случае, команда mov eax,[intg] копирует из памяти по адресу intg данные в регистр eax. В свою очередь команда mov [intg],eax запишет в память по адресу intg данные из регистра eax. Также рассмотрим команду mov eax,intg В этом случае в регистр eax запишется адрес intg. Допустим, для intg выделена память начиная с ячейки с адресом 0x600144, тогда команда mov eax,intg аналогична команде mov eax,0x600144 – т.е. эта команда запишет в регистр eax число 0x600144.

3 Выполнение лабораторной работы

Создал каталог для программ лабораторной работы № 6, перешел в него и создал файл lab-1.asm (рис. 3.1).

Рис. 3.1: Создание файлов

Введите в файл lab6-1.asm текст программы из листинга 6.1.(рис. 3.2).

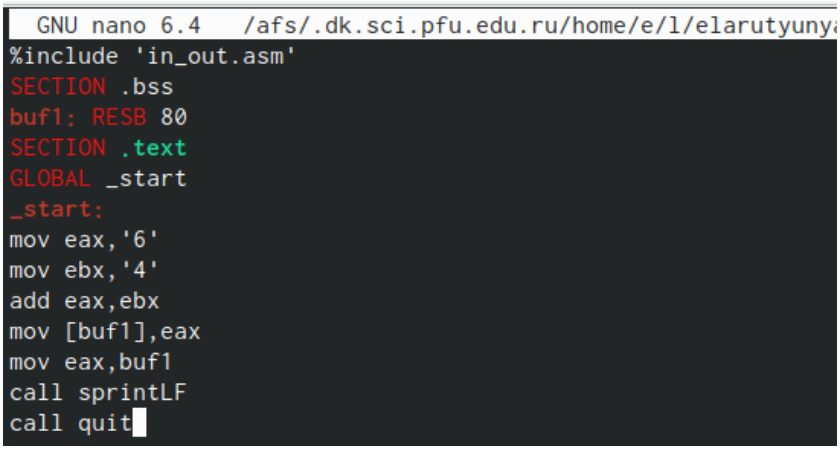

Рис. 3.2: Ввод текста

Создал исполняемый файл и запустил его (рис. 3.3).

```
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ d -m elf_i386 -o lab6-1 lab6-1.o
bash: d: команда не найдена
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ ./lab6-1
j
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $
```

Рис. 3.3: Запуск файла

Изменил исполняемый файл заменив строки (рис. 3.4).

```
  GNU nano 6.4    /afs/.d
%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. 3.4: Замена строк в файле

Запустил файл в котором заменил строки (рис. 3.5).

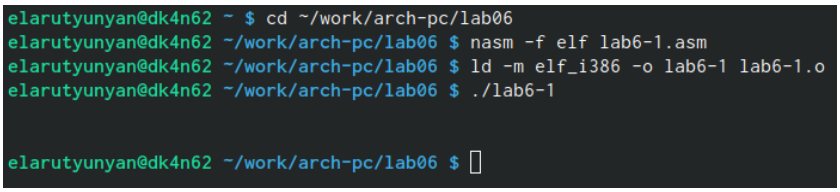

Рис. 3.5: Запуск файла

Создайте файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 (рис. 3.6).

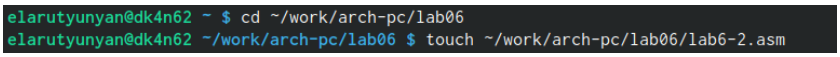

Рис. 3.6: Создание файла

Ввел в созданный файл текст (рис. 3.7).

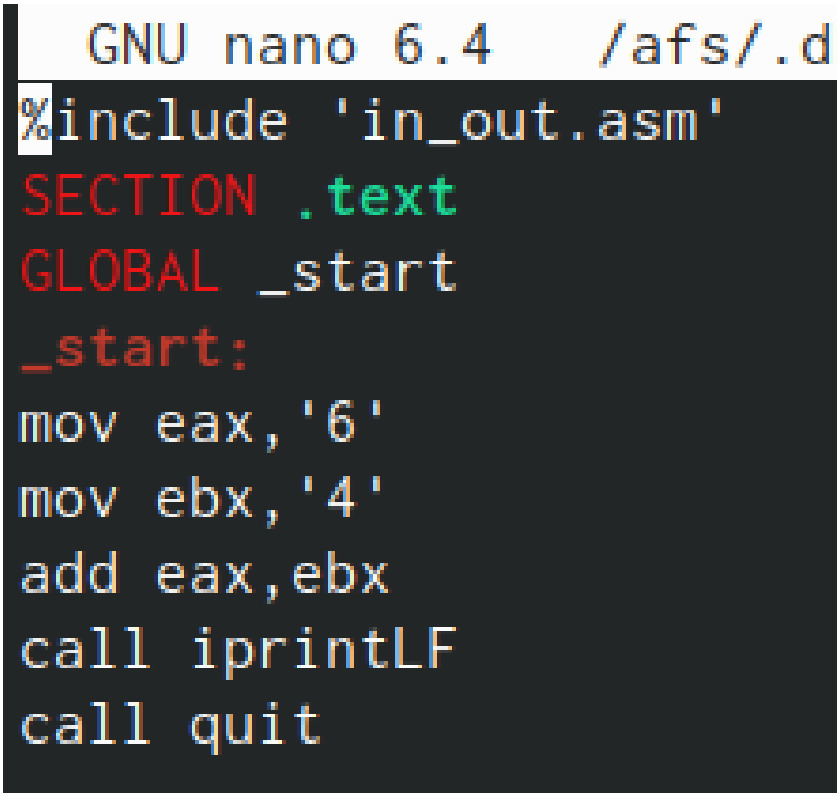

Рис. 3.7: Ввод текста

Далее запустил программу и получил результат 106 (рис. 3.8).

```
elarutyunyan@dk4n62 ~ $ cd ~/work/arch-pc/lab06
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ ./lab6-2
106
```

Рис. 3.8: Запуск и результат

Аналогично предыдущему примеру изменил символы на числа. Потом заменил строки на другие (рис. 3.9).

```
  GNU nano 6.4    /af
%include 'in_out.asm
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 3.9: Замена строк

Далее запустил программу и получил результат 10 (рис. 3.10).

```
elarutyunyan@dk4n62 ~ $ cd ~/work/arch-pc/lab06
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ ./lab6-2
10
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $
```

Рис. 3.10: Запуск и результат

Далее заменил в том же файле функцию iprintLF на iprint(рис. 3.11).

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

Рис. 3.11: Замена функции

Запустил измененную программу и получил результат 10 в той же строке и поняд чем отличается предыдущая программа от этой (рис. 3.12).

```
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ ./lab6-2
10elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $
```

Рис. 3.12: Запуск новой программы

Создайте файл lab6-3.asm в каталоге ~/work/arch-pc/lab06 (рис. 3.13).

```
elarutyunyan@dk4n62 ~ $ cd ~/work/arch-pc/lab06
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-3.asm
```

Рис. 3.13: Создание файла

Далее ввел в него код для вычисления выражения (рис. 3.14).

```
;-----------------------------
; 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
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 3.14: Ввод кода

После этого я запустил этот файл и получил результат (рис. 3.15).

```
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $
```

Рис. 3.15: Запуск файла и получение результата

Измените текст программы для вычисления выражения ⊠(⊠) = (4 ⊠ 6 + 2)/5. (рис. 3.16).

```
 GNU nano 6.4   /afs/.dk.sci.pfu.edu.ru/home/e/l/ela
;-----------------------------
; 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
call iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 3.16: Изменение текста программы

Далее запустил измененный текст и получил желаемый результат (рис. 3.17).

```
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $
```

Рис. 3.17: Запуск измененного файла

Создал файл variant.asm в каталоге ~/work/arch-pc/lab06 (рис. 3.18).

```
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/variant.asm
elarutyunyan@dk4n62 ~/work/arch-pc/lab06 $
```

Рис. 3.18: Создание файла variant.asm

Внимательно изучил текст программы из листинга 6.4 и ввел в файл variant.asm. (рис. 3.19).

```
;-----------------------------
; Программа вычисления варианта
;-----------------------------
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите No студенческого билета: ',1132232869
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
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```

Рис. 3.19: Ввод текста в файл variant.asm

Запустил файл и узнал номер своего варианта (рис. **??**).

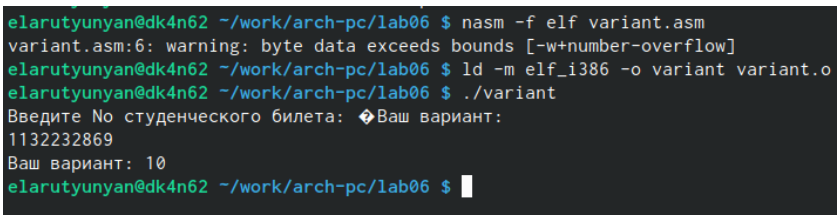

Рис. 3.20: Вариант 10

#Ответы на вопросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки кода: mov eax,rem call sprint
2. 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
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 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
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 на экран результатов вычислений отвечают строки: mov eax,edx call iprintLF #Самостоятельная работа

4 Вывод

Освоил арифметические инструкции языка NASM

Список литературы