

Отчёт по лабораторной работе №6
Простейший вариант

Метвалли Ахмед Фарг Набеев

Содержание

1 Цель работы

2 Теоретическое введение

3 Ответы на вопросы по программе

4 Выполнение самостоятельной работы

5 Выводы

Список литературы

Ошибка! Закладка не определена

Список иллюстраций

2.1 Создание каталога, переход в него, создание файла и его открытие

2.2 Ввод программы

2.3 Компиляция исходного файла и текста, передача файла компоновщику

2.4 Редактирование программы

2.5 Компиляция файла и передача файла компоновщику

2.6 Создание файла

2.7 Ввод программы

2.8 Создание файла, ввод программы, ввод студенческого

2.9 Вариант 17

4.1 С/р

4.2 Проверка

Список таблиц

1 . Цель работы

Цель данной лабораторной работы - освоение арифметических инструкций языка ассемблера NASM.

2.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 имена этих регистров, например: `mov ax,bx`. - Непосредственная адресация – значение операнда задается непосредственно в команде, Например: `mov ax,2`. - 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 символ- ном виде. Кодирование этой информации производится согласно кодовой табли- це 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, на- пример, 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 ситу- ация 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 Создал каталог lab06 перешел в него и создал файл lab6-1.asm и открыл его (рис. 2.1).

```

~$ mkdir ~/work/arch-pc/lab06
~$ cd ~/work/arch-pc/lab06
~/work/arch-pc/lab06$ touch lab6-1.asm
~/work/arch-pc/lab06$

```

Рис. 2.1: Создание каталога, переход в него, создание файла и его открытие Ввел

программу в файл (рис. 2.2).

```

#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
66 Демидова А. В.
Архитектура ЭВМ
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 iprintf ; 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 iprintf ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения

```

Рис. 2.2: Ввод программы

Скомпилировал исходный файл передал файл компоновщику (рис. 2.3).

```

ahmad-farg@ahmadfarg-VirtualBox:~$ mkdir ~/work/arch-pc/lab06
ahmad-farg@ahmadfarg-VirtualBox:~$ cd ~/work/arch-pc/lab06
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab06$ touch lab6-1.asm
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab06$ █

```

Рис. 2.3: Компиляция исходного файла и текста, передача файла компоновщику

Редактировал программу в файле lab6-1.asm (рис. 2.4).

```
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab06$
```

Рис. 2.4: Редактирование программы

Скомпилировал исходный файл передал файл компоновщику (рис. 2.5)

Рис. 2.5: Компиляция файла и передача файла компоновщику

Создание файла lab6-2 в том же каталоге (рис. 2.6).

```
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab06$ ./lab6-4
Введите x:
2
Результат: 16
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab06$ ./lab6-4
Введите x:
8
Результат: 100
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab06$
```

Рис. 2.6: Создание файла

Ввел программу в файл lab6-2.asm (рис. 2.7).

```
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i
386 -o lab6-2 lab6-2.o
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab06$ ./lab6-2
10
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab06$
```

mc [ahmad-farg@ahmadfarg-VirtualBox]:~...

Left	File	Command	Options
<-	~/work/arch-pc/lab06	-.[^]>	<- ~/work/a
.n	Name	Size	Modify time
/..	P--DIR	ek 21 16:53	/..
lab6-1.asm	0	ek 21 16:54	lab6-1.asm

```
%include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
moc ebx,'4'
add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintf
call quit
```

```
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab06$ touch lab6-2.asm
```

Рис. 2.7: Ввод программы

Создал файл variant.asm в том же каталоге ввел программу, затем ввел номер своего студенческого билета и узнал свой вариант-17. (рис. 2.8)

Рис. 2.8: Создание файла, ввод программы, ввод студенческого

Узнал номер своего варианта (рис. 2.9).

Рис. 2.9: Вариант 4

3. Ответы на вопросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки кода: `mov eax,rem` `call sprint`
2. Инструкция `mov esx, x` используется, чтобы положить адрес вводимой строки `x` в регистр `esx` `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

4.Выполнение самостоятельной работы

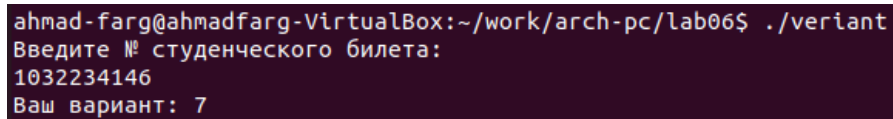
```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax,x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
68 Демидова А. В.
xor edx,edx
mov ebx,20
div ebx
inc edx
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```

В файле variant.asm очистил предыдущую программу и написал новую программу для выполнения самостоятельной работы (рис. 4.1).

Рис. 4.1: С/р

Проверил правильность программы(рис. 4.2).

Рис. 4.2: Проверка



```
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032234146
Ваш вариант: 7
```

4.Выводы

Освоил арифметические инструкции языка ассемблера NASM