РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук
Кафедра компьютерных и информационных наук

ОТЧЕТ

по лабораторной работе № 7

дисциплина: Архитектура компьютеров и операционные системы

Студент: Дьяконова Софья
Номер студенческого билета: 1132220829
Группа: НКАбд-01-22

МОСКВА 2022 г

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
3. Выполнение заданий для самостоятельной работы

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 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.

Ход работы:

1. С помощью утилиты mkdir создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №7. Перехожу в созданный каталог с помощью утилиты cd.

```
[sadjyakonova@localhost ~]$ mkdir /home/sadjyakonova/work/study/2022-2023/"Архит
ектура компьютера"/arh-pc/lab07
```

рис.1. создание директории

2. С помощью утилиты touch создаю файл lab7-1.asm .

```
[sadjyakonova@localhost arh-pc]$ touch lab07-1.asm
[sadjyakonova@localhost arh-pc]$ ls
CHANGELOG.md  lab06        labs      prepare             README.md
config        lab07        LICENSE   README.en.md        template
COURSE        lab07-1.asm  Makefile  README.git-flow.md
```

рис.2. Создание файла

3. Копирую в текущий каталог файл in_out.asm с помощью утилиты cp, т.к. он будет использоваться в других программах .
4. Открываю созданный файл lab7-1.asm, вставляю в него программу вывода значения регистра eax.

```
[sadjyakonova@localhost lab07]$ nasm -f elf lab07-1.asm
[sadjyakonova@localhost lab07]$ ld -m elf_i386 -o lab07-1 lab07-1.o
[sadjyakonova@localhost lab07]$ /lab07-1
bash: /lab07-1: Нет такого файла или каталога
[sadjyakonova@localhost lab07]$ ./lab07-1
j
```

рис.3. Запуск исполняемого файла

5. 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.
6. Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4

7. 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.

```
[sadjyakonova@localhost lab07]$ ./lab07-1c
```

рис.4. Запуск исполняемого файла

8. Создаю новый файл lab7-2.asm с помощью утилиты touch.
9. Ввожу в файл текст другойпрограммы для вывода значения регистра eax.
10. Создаю и запускаю исполняемый файл lab7-2. 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”.
11. Заменяю в тексте программы в файле lab7-2.asm символы “6” и “4” на числа 6 и 4.

```
[sadjyakonova@localhost lab07]$ touch lab07-2.asm
[sadjyakonova@localhost lab07]$ nasm -f elf lab07-2.asm
[sadjyakonova@localhost lab07]$ ld -m elf_i386 -o lab07-2 lab07-2.o
[sadjyakonova@localhost lab07]$ ./lab07-2
106
```

рис.5. Запуск исполняемого файла lab07-2.asm

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

```
[sadjyakonova@localhost lab07]$ nasm -f elf lab07-2.asm
[sadjyakonova@localhost lab07]$ ld -m elf_i386 -o lab07-2 lab07-2.o
[sadjyakonova@localhost lab07]$ ./lab07-2
10
```

рис.6. Запуск исполняемого файла lab07-2.asm

13. Заменяю в тексте программы функцию iprintLF на iprint.
14. 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.
15. Создаю файл lab7-3.asm с помощью утилиты touch.
16. Ввожу в созданный файл текст программы для вычисления значения выражения f(x) = (5 * 2 + 3)/3.

17. Создаю исполняемый файл и запускаю его.

```
[sadjyakonova@localhost lab07]$ touch lab07-3.asm
[sadjyakonova@localhost lab07]$ nasm -f elf lab07-3.asm
[sadjyakonova@localhost lab07]$ ld -m elf_i386 -o lab07-3 lab07-3.o
[sadjyakonova@localhost lab07]$ ./lab07-3
Результат: 4
Остаток от деления: 1
```

рис.7. Запуск исполняемого файла lab07-3.asm

18. Изменяю программу так, чтобы она вычисляла значение выражения f(x) = (4 * 6 + 2)/5
19. Создаю и запускаю новый исполняемый файл. Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

```
[sadjyakonova@localhost lab07]$ nasm -f elf lab07-3.asm
[sadjyakonova@localhost lab07]$ ld -m elf_i386 -o lab07-3 lab07-3.o
[sadjyakonova@localhost lab07]$ ./lab07-3
Результат: 5
Остаток от деления: 1
```

рис.8. Запуск исполняемого файла lab07-3.asm

20. Создаю файл variant.asm с помощью утилиты touch.
21. Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета.
22. Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 10.

```
[sadjyakonova@localhost lab07]$ touch variant.asm
[sadjyakonova@localhost lab07]$ mousepad variant.asm
bash: mousepad: команда не найдена...
[sadjyakonova@localhost lab07]$ nasm -f elf variant.asm
[sadjyakonova@localhost lab07]$ ld -m elf_i386 -o variant variant.o
[sadjyakonova@localhost lab07]$ ./variant
Введите No студенческого билета:
1132220829
Ваш вариант: 10
```

рис.9. Запуск исполняемого файла variant.asm

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
```

Задания для самостоятельной работы:

1. Создаю файл lab7-4.asm с помощью утилиты touch
2. Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения (11 + x) * 2 - 6. Это выражение было под вариантом 10.
3. Создаю и запускаю исполняемый файл.

```
[sadjyakonova@localhost lab07]$ touch lab07-4.asm
[sadjyakonova@localhost lab07]$ nasm -f elf lab07-4.asm
[sadjyakonova@localhost lab07]$ ld -m elf_i386 -o lab07-4 lab07-4.o
[sadjyakonova@localhost lab07]$ ./lab07-4
Введите значение переменной x: 3
Результат: 22[sadjyakonova@localhost lab07]$ ./lab07-4
```

рис.10. Запуск исполняемого файла

4. Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе.

```
Результат: 22[sadjyakonova@localhost lab07]$ ./lab07-4
Введите значение переменной x: 11
Результат: 38[sadjyakonova@localhost lab07]$
```

рис.11. Запуск исполняемого файла

```
%include 'in_out.asm' ; подключение внешнего файла

SECTION .data ; секция инициированных данных

msg: DB 'Введите значение переменной x: ',0

rem: DB 'Результат: ',0

SECTION .bss ; секция не инициированных данных

x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный размер - 80 байт

SECTION .text ; Код программы

GLOBAL _start ; Начало программы

_start: ; Точка входа в программу

; ---- Вычисление выражения

mov eax, msg ; запись адреса выводимиого сообщения в eax

call sprint ; вызов подпрограммы печати сообщения

mov ecx, x ; запись адреса переменной в ecx

mov edx, 80 ; запись длины вводимого значения в edx

call sread ; вызов подпрограммы ввода сообщения
```

```
mov eax,x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
add eax,11; eax = eax+11 = x + 11
mov ebx,2 ; запись значения 2 в регистр ebx
mul ebx; EAX=EAX*EBX = (x+11)*2
add eax,-6; eax = eax-6 = (x+11)*2-6
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprint ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
```

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

Список литературы:

1. Лабораторная работа No7. Таблица ASCI