

Отчёт по лабораторной работе

Дисциплина: архитектура компьютера

Назармамадов Умед Джамshedович

Содержание

1 Цель работы

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

2 Задание

1. Символьные и численные данные в NASM.
2. Выполнение арифметических операций в NASM.
3. Выполнение заданий для самостоятельной работы.

3 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. 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 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. Расширенная таблица ASCII состоит из двух частей. Первая (символы с кодами 0-127) является универсальной (см. Приложение.), а вторая (коды 128-255) предназначена для специальных символов и букв национальных алфавитов и на компьютерах разных

типов может меняться. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). По-этому, 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. Для выполнения лабораторных работ в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Это:

- `iprint` – вывод на экран чисел в формате ASCII, перед вызовом `iprint` в регистр `eax` необходимо записать выводимое число (`mov eax, ...`).
- `iprintLF` – работает аналогично `iprint`, но при выводе на экран после числа добавляет к символ перевода строки.
- `atoi` – функция преобразует ascii-код символа в целое число и записывает результат в регистр `eax`, перед вызовом `atoi` в регистр `eax` необходимо записать число (`mov eax, ...`).

4 Выполнение лабораторной работы

С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6. Перехожу в созданный каталог с помощью утилиты `cd`

```
udnazarmamadov@dk8n81 ~$ mkdir ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/lab06
udnazarmamadov@dk8n81 ~$ cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/lab06
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $
```

Создание директории

С помощью утилиты `touch` создаю файл `lab6-1.asm`.

```
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ touch lab6-1.asm
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ls
lab6-1.asm
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $
```

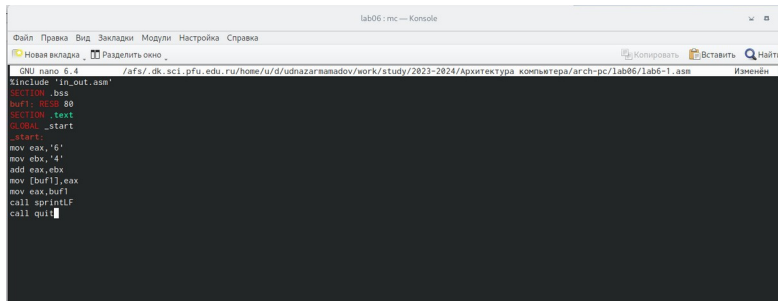
Создание файла

Копирую в текущий каталог файл `in_out.asm` с помощью утилиты `cp`.

```
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ cp ~/Зарпукки/in_out.asm in_out.asm
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ls
in_out.asm lab6-1.asm
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $
```

Создание копии файла

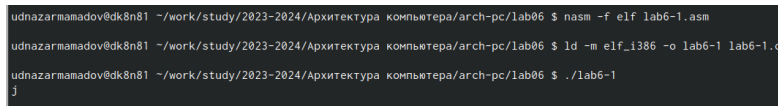
Открываю созданный файл `lab6-1.asm`, вставляю в него программу вывода значения регистра `eax`.



```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/u/d/udnazarmamadov/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06/lab6-1.asm
#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 sprintf
call quit
```

Редактирование файла

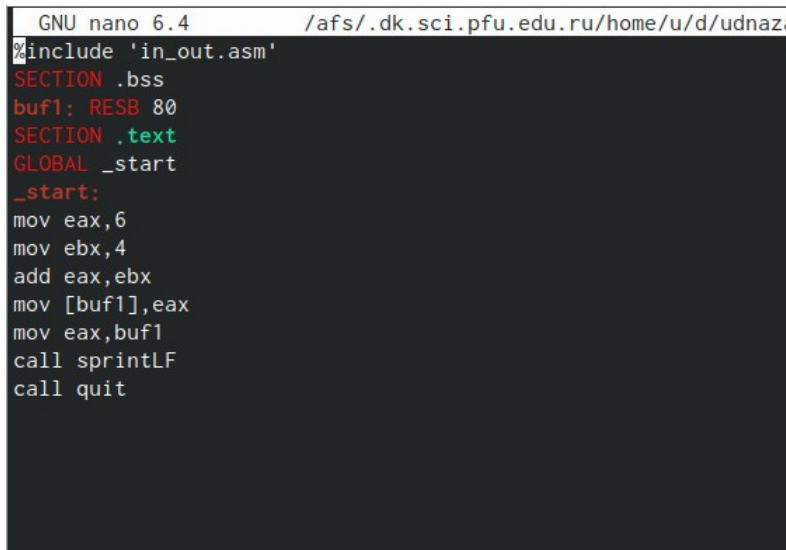
Создаю исполняемый файл программы и запускаю его. Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.



```
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ nasm -f elf lab6-1.asm
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ./lab6-1
j
```

Запуск файла

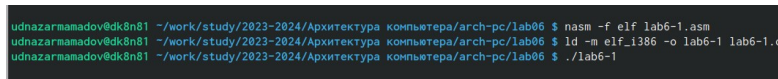
Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4



```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/u/d/udnazarmamadov/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06/lab6-1.asm
#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 sprintf
call quit
```

Редактирование файла

Создаю новый исполняемый файл программы и запускаю его.



```
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ nasm -f elf lab6-1.asm
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ./lab6-1
```

Запуск файла

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

```
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ touch lab6-2.asm
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $
```

Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax.

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/u/d/ud
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

Редактирование файла

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”.

```
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ nasm -f elf lab6-2.asm
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ./lab6-2
106
```

Запуск файла

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

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/u/d/udnazarmamadov/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06/lab6-2.asm
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Редактирование файла

Создаю и запускаю новый исполняемый файл.

```
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ nasm -f elf lab6-2.asm
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ./lab6-2
10
```

Запуск файла

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
```

Редактирование файла

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`.

```
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ nasm -f elf lab6-2.asm
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ./lab6-2
10
```

Запуск файла

Создаю файл `lab6-3.asm` с помощью утилиты `touch`.

```
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ touch lab6-3.asm
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $
```

Создание файла

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

```
lab06 : mc — Console
Файл Правка Вид Закладки Модули Настройка Справка
Новая вкладка Разделить окно Копировать
GNU nano 6.4 /afs/dk.sci.pfu.edu.ru/home/u/d/udnazarmamadov/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06/lab6-3.asm
;-----
; 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, EDI=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
```

Редактирование файла

Создаю исполняемый файл и запускаю его.

```
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ nasm -f elf lab6-3.asm
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Запуск файла

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

```
GNU nano 6.4 /afs/dk.sci.pfu.edu.ru/home/u/d/udnazarmamadov/work/study/2023-2024/Архитектура компьютера/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,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, EDI=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
```

Изменение программы

Создаю и запускаю новый исполняемый файл.

```
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ nasm -f elf lab6-3.asm
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
```

Запуск файла

Создаю файл variant.asm с помощью утилиты touch.

```
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ touch variant.asm
```

Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета.

```
; Программа вычисления варианта
;-----
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите No студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
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 sprintf
mov eax, edx
call iprintLF
call quit
```

Редактирование файла

Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант 6.

```
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ nasm -f elf variant.asm
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
udnazarmamadov@dk8n81 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1032225205
Ваш вариант: 6
```

Запуск файла

5 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
```


6 Выводы

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

Список литературы

1. Лабораторная работа №6.
2. Таблица ASCII.