Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Мишина Анастасия Алексеевна

Содержание

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

# 2 Выполнение лабораторной работы

Для начала создадим каталог для программ 7-ой лабораторной работы, перейдем в нее и создадим файл lab7-1.asm (рис. 1).

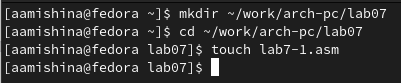


Рис. 1: Создание каталога и файла

Вводим текст программы из листинга 7.1 в наш файл. Копируем из загрузок файл in\_out.asm в каталог lab07 (рис. 2). Создадим и запустим исполняемый файл (рис. 3). Видим в результате символ “j” (сумму ASCII кодов символов 4 и 6).

Программа lab7-1.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 sprintLF  
call quit

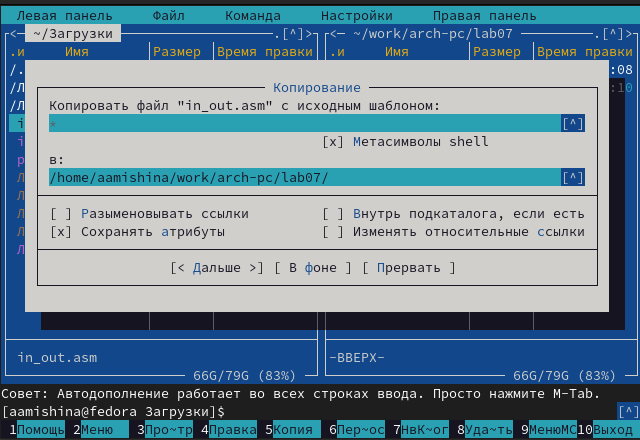


Рис. 2: Копирование файла in\_out.asm

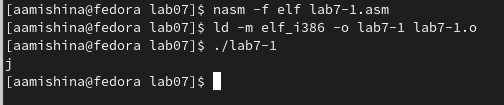


Рис. 3: Работа с файлом lab7-1.asm

Меняем символы на числа (рис. 5), пересоздаем исполняемый файл и запускам его (рис. 6). Теперь программа выведет символ с кодом 10.

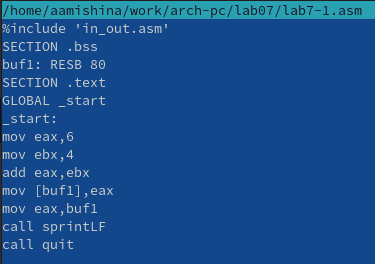


Рис. 4: Редактирование файла lab7-1.asm

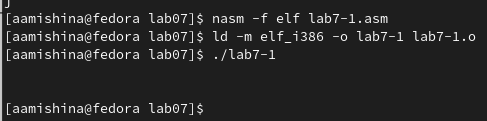


Рис. 5: Работа с измененным файлом lab7-1.asm

Пользуясь таблицой ASCII определяем, что код 10 соответствует символу перевода строки. Данный символ не отображается на экране, однако мы можем заметить пустую строку.

Создаем файл lab7-2.asm в каталоге ~/work/arch-pc-lab07 и вводим в него текст программы (листинг 7.2) (рис. 6). В результате выводится число 106 (сумма кодов символов ‘6’ и ‘4’, преобразованная в число).

Программа lab7-2.asm:

%include 'in\_out.asm'  
SECTION .text  
GLOBAL \_start  
\_start:  
mov eax,'6'  
mov ebx,'4'  
add eax,ebx  
call iprintLF  
call quit

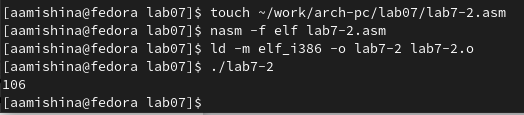


Рис. 6: Работа с файлом lab7-2.asm

Аналогично действиям выше меняем символы на числа (рис. 7). Создаем и запускаем измененный исполняемый файл. Теперь программа складывает сами числа, а не их коды, результатом является число 10 (рис. 8).

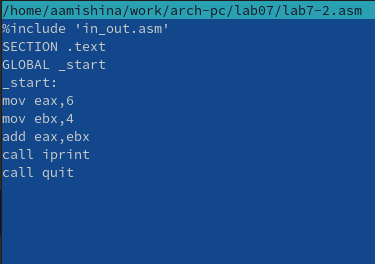


Рис. 7: Редактирование файла lab7-2.asm

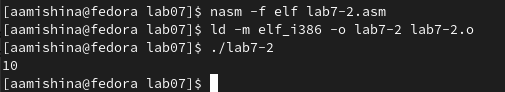


Рис. 8: Работа с измененным файлом lab7-2.asm

Поменяем функцию iprintLF на iprint. Пересоздаем исполняемый файл, заметим, что теперь нет символа перевода строки (рис. 9).

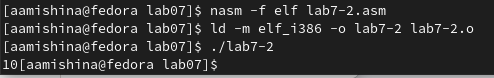


Рис. 9: Запуск измененного файла

Создаем файл lab7-3.asm, вводим в него текст программы из листинга 7.3 для вычисления следующего выражения: (5\*2+3)/3. Создаем и запускаем исполняемый файл (рис. 10).

Программа lab7-3.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, 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

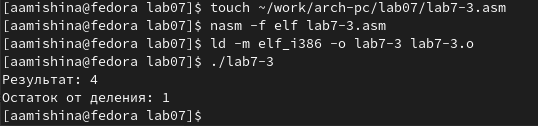


Рис. 10: Работа с файлом lab7-3.asm

Изменяем код для подсчета выражения: (4\*6+2)/5. Проверяем работу программы (рис. 11).

;--------------------------------  
; 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

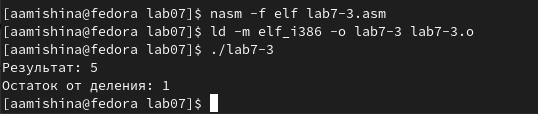


Рис. 11: Работа с измененным файлом lab7-3.asm

Теперь создадим файл variant.asm для вычисления варианта по номеру студенческого билета. Вводим в новый файл текст программы из листинга 7.4. Создаем и запускаем исполняемый файл, вводим туда свой номер студ. билета, получаем вариант 13 (рис. 12). Аналитически также получаем вариант 13.

Программа variant.asm:

;--------------------------------  
; 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 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

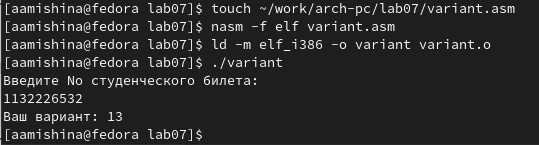


Рис. 12: Вычисление варианта

Ответы на вопросы по программе: 1. Строки, отвечающие за вывод сообщение “Ваш вариант:”

mov eax,rem  
call sprint

1. Конструкция “mov ecx, x” используется для того, чтобы положить в регистр ecx адрес х. Конструкция “mov edx, 80” нужна, чтобы записать в регистр edx длину вводимого сообщения. “call sread” позволяет вызвать подпроргамму из внешего файла для ввода сообщения с клавиатуры.
2. Инструкция “call atoi” используется для преобразования ASCII-кода символа в целое число и записи результата в регистр eax.
3. За вычисление варианта отвечают следующие строки кода:

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

1. При выполнении инструкции “div edx” остаток от деления записывается в регистр edx.
2. Инструкция “inc edx” позволяет увеличить значение регистра edx на 1.
3. Строки, отвечающие за вывод на экран результата вычислений:

mov eax,edx  
call iprintLF

# 3 Выполнение заданий самостоятельной работы

Согласно моему варианту (13) мне следовало запрограммировать подсчет функции (8х+6)\*10. Создаю файл mytask.asm, пишу код, создаю и запускаю исполняемый файл. Проверяю значения при х = 1 и х = 4. Программа отрабатывает верно (рис. 13).

Программа mytask.asm:

;--------------------------------  
; Программа вычисления выражения  
;--------------------------------  
%include 'in\_out.asm' ; подключение внешнего файла  
SECTION .data  
msg: DB 'Введите х для подсчета выражения (8х+6)\*10: ',0  
res: DB 'Результат: ',0  
  
SECTION .bss  
x: RESB 80  
  
SECTION .text  
GLOBAL \_start  
\_start:  
  
mov eax, msg  
call sprint  
  
mov ecx, x  
mov edx, 80  
call sread  
  
mov eax, x  
call atoi  
  
mov ebx, 8  
mul ebx  
  
add eax, 6  
  
mov ebx, 10  
mul ebx  
  
mov edi, eax  
  
mov eax, res  
call sprint  
  
mov eax, edi  
call iprintLF  
  
call quit ; вызов подпрограммы завершения

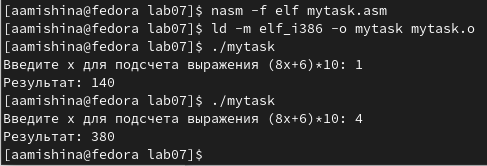


Рис. 13: Моя программа

# 4 Выводы

В ходе выполнения данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM. Вся моя работа была записана и прокомментирована мной в данной лабораторной.