Лабораторная работа №8

НКАбд-06-23

Улитина Мария Максимовна

Содержание

1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

2 Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки
3. Написание программы для вычисления суммы функций

3 Теоретическое введение

3.1 Организация стека

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды. Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров. Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается. Для стека существует две основные операции:

• добавление элемента в вершину стека (push);

• извлечение элемента из вершины стека (pop).

3.1.1 Добавление элемента в стек

Команда push размещает значение в стеке, т.е. помещает значение в ячейку памяти, на которую указывает регистр esp, после этого значение регистра esp увеличивается на 4. Данная команда имеет один операнд — значение, которое необходимо поместить в стек.

3.1.2 Извлечение элемента из стека

Команда pop извлекает значение из стека, т.е. извлекает значение из ячейки памяти, на которую указывает регистр esp, после этого уменьшает значение регистра esp на 4. У этой команды также один операнд, который может быть регистром или переменной в памяти. Нужно помнить, что извлечённый из стека элемент не стирается из памяти и остаётся как “мусор”, который будет перезаписан при записи нового значения в стек.

3.2 Инструкции организации

Для организации циклов существуют специальные инструкции. Для всех инструкций максимальное количество проходов задаётся в регистре ecx. Наиболее простой является инструкция loop. Она позволяет организовать безусловный цикл. Инструкция loop выполняется в два этапа. Сначала из регистра ecx вычитается единица и его значение сравнивается с нулём. Если регистр не равен нулю, то выполняется переход к указанной метке. Иначе переход не выполняется и управление передаётся команде, которая следует сразу после команды loop.

4 Выполнение лабораторной работы

Создадим каталог для программ лабораторной работы, перейдём в него и создадим файл lab8-1.asm (рис. 1).

```
mmulitina@ubuntu:~$ mkdir  ~/work/arch-pc/lab08
mmulitina@ubuntu:~$ cd ~/work/arch-pc/lab08
mmulitina@ubuntu:~/work/arch-pc/lab08$ touch lab8-1.asm
```

Figure 1: Создание каталога

Введём в файл lab8-1.asm текст программы (рис. 2).

```
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
```

Figure 2: Текст программы

Проверим работу программы (рис. 3).

```
mmulitina@ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
mmulitina@ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
mmulitina@ubuntu:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 3
3
2
1
```

Figure 3: Работа программы

Изменим текст программы (рис. 4).

```
label:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
loop label


call quit
```

Figure 4: Изменение программы

Проверим работу программы (рис. 5).

```
4286700994
4286700992
4286700990
4286700988
4286700986
4286700984
4286700982
4286700980
4286700978
4286700976
```

Figure 5: Работа программы

Снова изменим текст программы (рис. 6).

```
; ------ Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
push ecx ; добавление значения ecx в стек
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx ; извлечение значения ecx из стека
loop label


call quit
```

Figure 6: Изменение программы

Проверим работу программы (рис. 7).

```
mmulitina@ubuntu:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
4
3
2
1
0
```

Figure 7: Работа программы

Создадим файл lab8-2.asm (рис. 8).

```
mmulitina@ubuntu:~/work/arch-pc/lab08$ touch lab8-2.asm
```

Figure 8: Создание файла

Запустим программу, указав аргументы (рис. 9).

```
mmulitina@ubuntu:~/work/arch-pc/lab08$ ./lab8-2 arg1 arg2 'arg3'
arg1
arg2
arg3
```

Figure 9: Работа программы

Создадим файл lab8-3.asm, введём в него необходимый текст программы и запустим его (рис. 10).

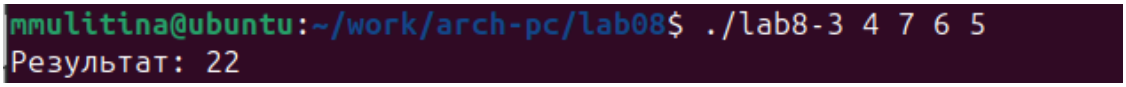

Figure 10: Работа программы

4.1 Задания для самостоятельной работы

Напишем программу для варианта №3 и проверим на разных наборах значений (рис. 11).

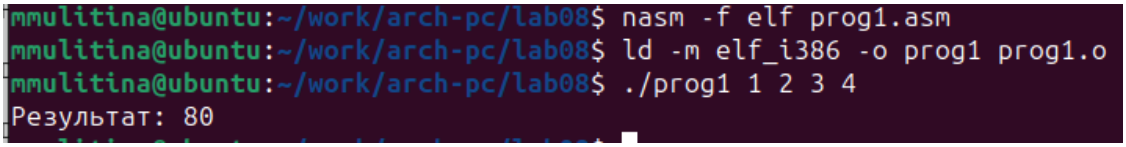

Figure 11: Работа программы

(рис. 12).

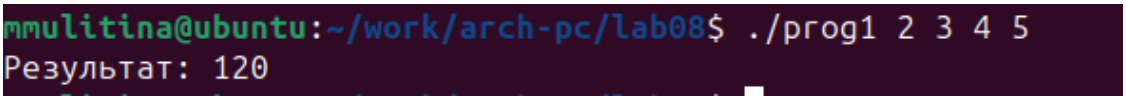

Figure 12: Работа программы

5 Выводы

В процессе выполнения работы были приобретены навыки написания программ с использованием циклов и обработкой аргументов командной строкию

6 Список литературы

Архитектура ЭВМ. Лабораторная работа №8.