Программирование цикла. Обработка аргументов командной строки.

Лабораторная работа №8

Губайдуллина Софья Романовна

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) Выполнение самостоятельной работы.

3 Теоретическое введение

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. Для стека существует две основные операции: • добавление элемента в вершину стека (push); • извлечение элемента из вершины стека (pop).

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.Существует ещё две команды для добавления значений в стек. Это команда pusha, которая помещает в стек содержимое всех регистров общего назначения в следующем порядке: ax, cx, dx, bx, sp, bp, si, di. А также команда pushf, которая служит для перемещения в стек содержимого регистра флагов. Обе эти команды не имеют операндов.

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. Аналогично команде записи в стек существует команда popa, которая восстанавливает из стека все регистры общего назначения, и команда popf для перемещения значений из вершины стека в регистр флагов.

Для организации циклов существуют специальные инструкции. Для всех инструкций максимальное количество проходов задаётся в регистре ecx. Наиболее простой является ин- струкция loop. Она позволяет организовать безусловный цикл. Иструкция 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

1) Реализация циклов в NASM.

Начинаю выполнение лабораторной работы с создания папки lab08 и файла lab8-1.asm (рис. ??).

```
srgubayjdullina@dk5n52 ~ $ mkdir ~/work/arch-pc/lab08
srgubayjdullina@dk5n52 ~ $ cd ~/work/arch-pc/lab08
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ touch lab8-1.asm
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $
```

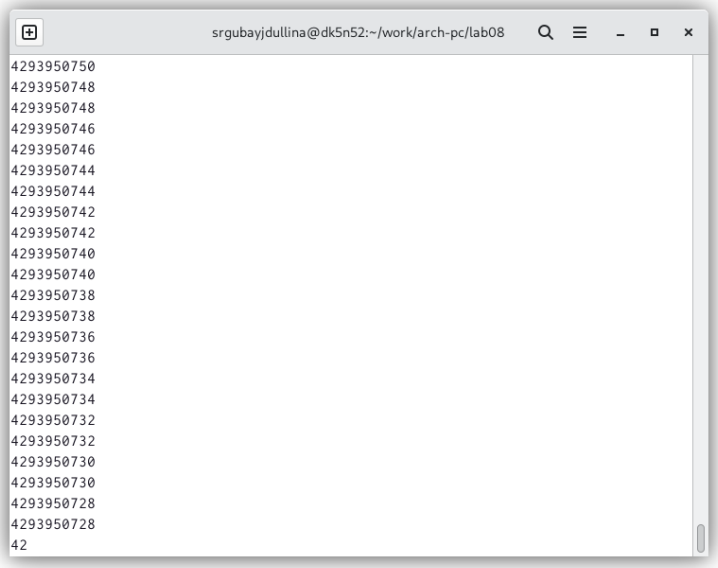
Создание рабочей папки и файла lab08 и lab8-1.asm

Непосредственно в новый файл lab8-1.asm записываю листинг 8.1 и запускаю его работу, создав исполняемый файл (рис. ??). Данный пример показывает, что использование регистра ecx в теле цилка loop может привести к некорректной работе программы.

```
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 7
7
6
5
4
3
2
1
```

Создание исполняемого файла lab8-1.asm и проверка его работы

Далее изменяю текст листинга в файле lab8-1.asm, добавив изменение значения регистра ecx в цикле. Создаю новый исполняемый файл, проверяю работу программы и получаю бесконечный цикл (число проходов цикла бесконечно). (рис. ??)



Проверка работы нового lab8-1.asm с циклом loop

Теперь меняю файл lab8-1.asm, добавляя команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop. После редактирования листинга программы проверяю правильность - создание исполняемого файла и проверка работы (рис. ??)

```
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 7
6
5
4
3
2
1
0
```

Проверка работы lab8-1.asm с добавлением стэка в цикл

2) Обработка аргументов командной строки.

Создаю lab8-2.asm и ввожу в файл текст программы листинга для считывания значения аргументов. Проверяю правильность работы (рис. ??) и вижу, что программа обработала все введенные с клавиатуры аргументы.

```
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ ./lab8-2
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ ./lab8-2 4 56 12
4
56
12
```

Создание исполняемого файла и проверка работы файла lab8-2.asm

Создаю файл lab8-3.asm - новый файл в котором я буду работать. Теперь непосредственно в lab8-3.asm ввожу листинг программы по вычислению суммы аргументов командной строки. Создаю исполняемый файл и проверяю корректность работы программы (рис. ??)

```
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ touch lab8-3.asm
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ mc

srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ ./lab8-3 43 87 184 34 11
Результат: 359
```

Проверка работы программы по вычислению суммы аргументов в lab8-3.asm

Далее по заданию мне следует изменить листинг программы в файле lab8-3.asm таким образом, чтобы программа вместо суммы считала произведение вводимых аргументов. Для этого я создала новый файл lab8-3.asm и ниже представлен листинг заданной программы:

%include ‘in_out.asm’ SECTION .data msg db “Результат:”,0 SECTION .text global _start _start: pop ecx ; Извлекаем из стека в ecx количество ; аргументов (первое значение в стеке) pop edx ; Извлекаем из стека в edx имя программы ; (второе значение в стеке) sub ecx,1 ; Уменьшаем ecx на 1 (количество ; аргументов без названия программы) mov esi,1 ; Используем esi для хранения next: cmp ecx,0h ; проверяем, есть ли еще аргументы jz _end ; если аргументов нет выходим из цикла ; (переход на метку _end) pop eax ; иначе извлекаем следующий аргумент из стека call atoi ; преобразуем символ в число mul esi mov esi, eax loop next ; переход к обработке следующего аргумента _end: mov eax, msg ; вывод сообщения “Результат:” call sprint mov eax, esi ; записываем в регистр eax call iprintLF ; печать результата call quit ; завершение программы

Создаю исполняемый файл, проверяю корректность введенной программы (рис. ??)

```
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ ./lab8-3 3 4 7
Результат: 84
```

Корректность программы по вычислению произведения вводимых аргументов

3) Выполнение самостоятельной работы.

Теперь перехожу к выполнению самостоятельной работы. Для этого создаю новый файл lab8-4.asm, вводя туда листинга программы по вычислению значения функции. Мой вариант - 20, функция - 3(10+x). Ниже представлен листинг программы:

%include ‘in_out.asm’ SECTION .data msg db “Результат:”,0 SECTION .text global _start _start: pop ecx ; Извлекаем из стека в ecx количество ; аргументов (первое значение в стеке) pop edx ; Извлекаем из стека в edx имя программы ; (второе значение в стеке) sub ecx,1 ; Уменьшаем ecx на 1 (количество ; аргументов без названия программы) mov esi, 0 ; Используем esi для хранения ; промежуточных сумм next: cmp ecx,0h ; проверяем, есть ли еще аргументы jz _end ; если аргументов нет выходим из цикла ; (переход на метку _end) pop eax ; иначе извлекаем следующий аргумент из стека call atoi ; преобразуем символ в число mov edi,3 add eax,10 mul edi add esi,eax loop next ; переход к обработке следующего аргумента _end: mov eax, msg ; вывод сообщения “Результат:” call sprint mov eax, esi ; записываем сумму в регистр eax call iprintLF ; печать результата call quit ; завершение программы

Для того чтобы проверить правильность написания программы, создаю исполняемый файл для lab8-4.asm, ввожу аргументы и сверяю с выведенным результатом. Программа работает корректно (рис. ??)

```
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ ./lab8-4 1
Результат: 33
srgubayjdullina@dk5n52 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3
Результат: 108
```

Работа программы по вычислению значения от заданной функции

5 Выводы

Выполняя лабораторную работу, я приобрела новые навыки написания программ с использованием циклов и обработкой аргументов командной строки.