Лабораторная работа No8

Программирование цикла. Обработка аргументов командной строки

Четвергова Мария Викторовна

Содержание

# 1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки. Целью работы является приобретение навыков работы с написанием программ с использованием циклов и обработкой аргументов командной строки. Использование полученных знаний на практике.

# 2 Теоретическое введение

8.2.1. 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. На рисунке показана схема организации стека в процессоре. 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 указа- тель стека уменьшается, а при извлечении — увеличивается. Для стека существует две основные операции:

• добавление элемента в вершину стека (push);

• извлечение элемента из вершины стека (pop).

8.2.1.1. 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.

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

8.2.1.2. 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.

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.

8.2.2. Инструкции организации циклов 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, типичная структура которого имеет следующий вид. 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.

# 3 Выполнение лабораторной работы

Реализация циклов в NASM

Создайте каталог для программам лабораторной работы No 8, перейдите в него и создайте файл lab8-1.asm:

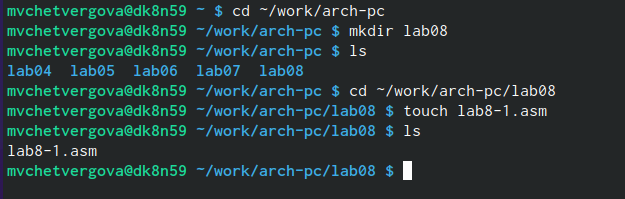
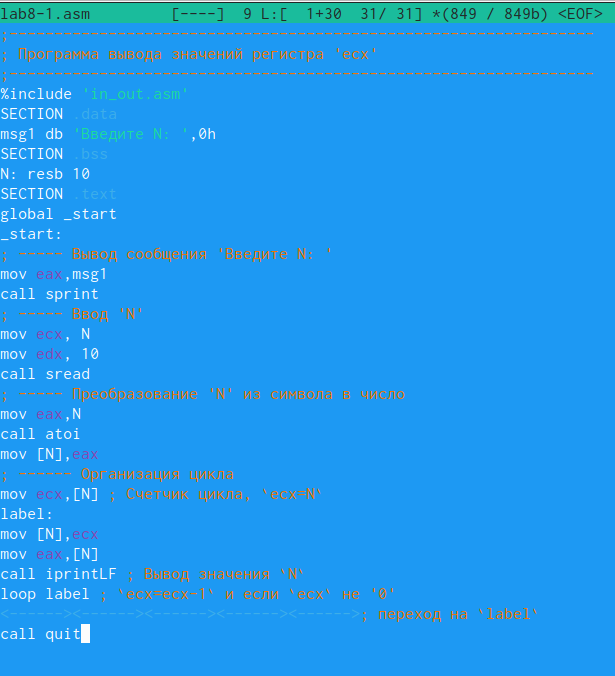


Рис.1 Создание каталога и файла для работы

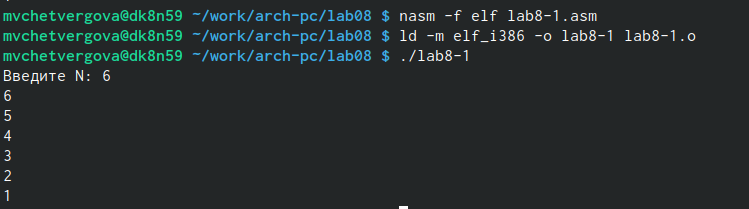
При реализации циклов в NASM с использованием инструкции loop необходимо помнить о том, что эта инструкция использует регистр ecx в качестве счетчика и на каждом шаге уменьшает его значение на единицу. В качестве примера рассмотрим программу, которая выводит значение регистра ecx. Внимательно изучите текст программы (Листинг 8.1). Листинг 8.1. Программа вывода значений регистра ecx

;—————————————————————– ; Программа вывода значений регистра ‘ecx’ ;—————————————————————– %include ‘in\_out.asm’ SECTION .data msg1 db ‘Введите N:’,0h 88 Демидова А. В. Архитектура ЭВМ 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 mov [N],eax ; —— Организация цикла mov ecx,[N] ; Счетчик цикла, ecx=N label: mov [N],ecx mov eax,[N] call iprintLF ; Вывод значения N loop label ; ecx=ecx-1 и если ecx не ‘0’ ; переход на label call quit

Введите в файл lab8-1.asm текст программы из листинга 8.1. Создайте исполняемый файл и проверьте его работу.



Ввод листинга программы

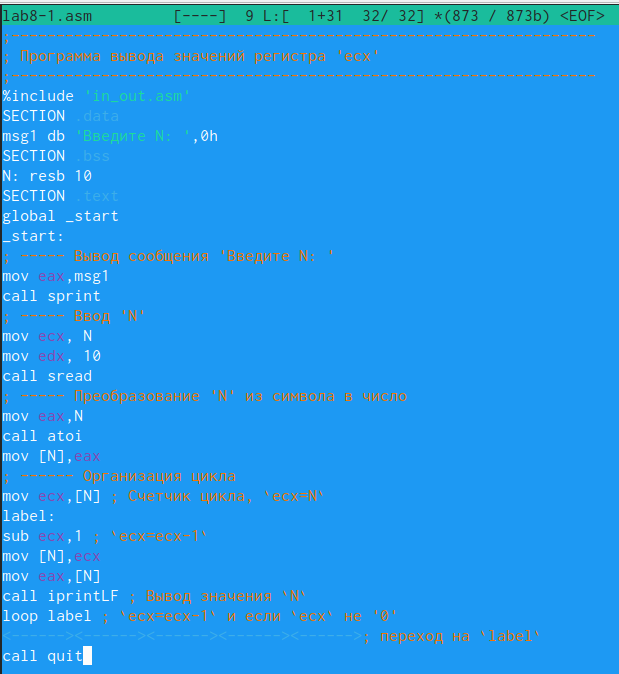


Проверка работы данной программы

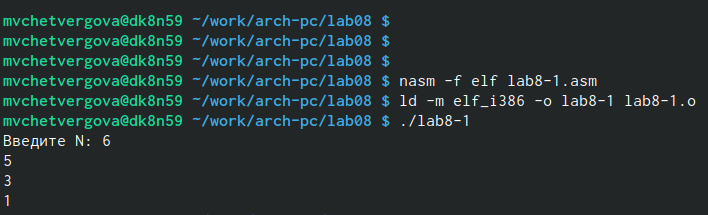
Вывод: Данный пример показывает, что использование регистра ecx в теле цилка loop может привести к некорректной работе программы. Изменим текст программы добавив изменение значение регистра ecx в цикле:

label: sub ecx,1 ; ecx=ecx-1 mov [N],ecx mov eax,[N] call iprintLF Демидова А. В. 89 Архитектура ЭВМ loop label

Создайте исполняемый файл и проверьте его работу.



Изменение листинга программы

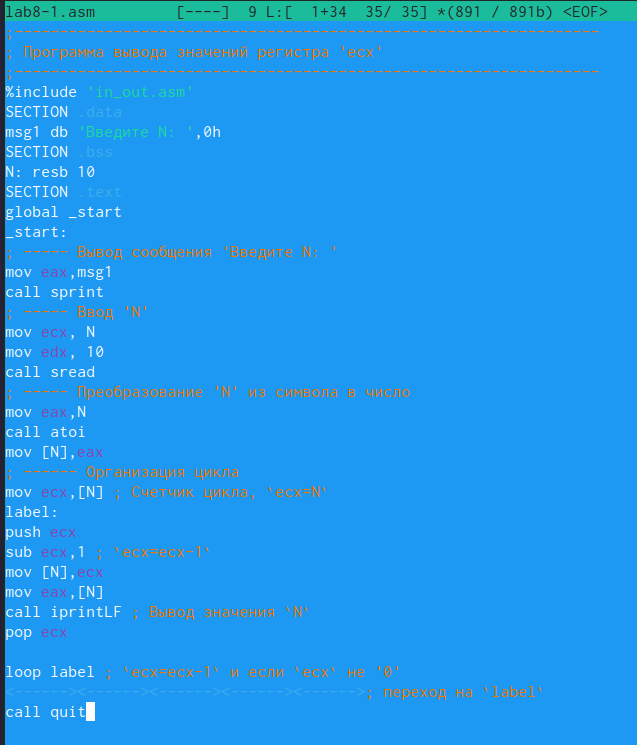


Проверка работы изменённой программы

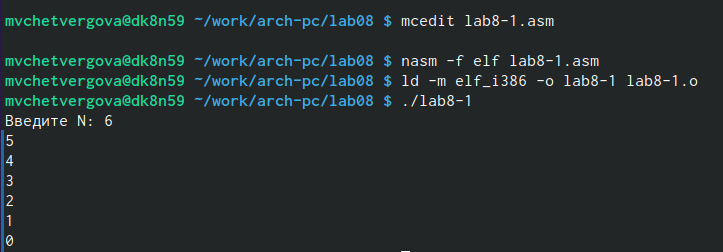
Какие значения принимает регистр ecx в цикле? Соответствует ли число проходов цикла значению 𝑁 введенному с клавиатуры? Ответ: Регистр ecx принимает значение

Для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. Внесём изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла

loop: 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



Изменение листинга программы



Проверка работы изменённой программы

Создайте исполняемый файл и проверьте его работу. Соответствует ли в данном случае число проходов цикла значению 𝑁 введенному с клавиатуры? Да, число проходов совпадает. Программа проводит 6 операций во время выполнения кода: работает с числами 6 с переходом в 0, 5 с переходом в 4, 4 с переходом в 3, 3 с переходом в 2, 2 с переходом в 1 и 1 с переходом в 0. После прохода с единицей получается 0 и программа прекращает действие.

8.3.2. Обработка аргументов командной строки

При разработке программ иногда встает необходимость указывать аргументы, которые будут использоваться в программе, непосредственно из командной строки при запуске программы.

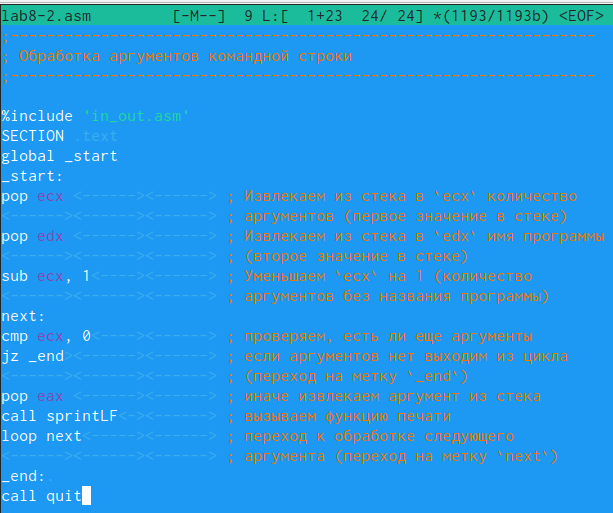
При запуске программы в NASM аргументы командной строки загружаются в стек в обрат- ном порядке, кроме того в стек записывается имя программы и общее количество аргументов. Последние два элемента стека для программы, скомпилированной NASM, – это всегда имя программы и количество переданных аргументов.

Таким образом, для того чтобы использовать аргументы в программе, их просто нужно извлечь из стека. Обработку аргументов нужно проводить в цикле. Т.е. сначала нужно из- влечь из стека количество аргументов, а затем циклично для каждого аргумента выполнить логику программы. В качестве примера рассмотрим программу, которая выводит на экран аргументы командной строки. Внимательно изучите текст программы (Листинг 8.2).

Листинг 8.2. Программа выводящая на экран аргументы командной строки

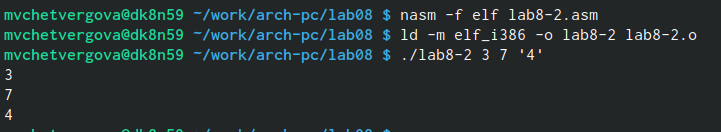
;—————————————————————– ; Обработка аргументов командной строки ;—————————————————————– %include ‘in\_out.asm’ SECTION .text global \_start \_start: pop ecx ; Извлекаем из стека в ecx количество ; аргументов (первое значение в стеке) pop edx ; Извлекаем из стека в edx имя программы ; (второе значение в стеке) sub ecx, 1 ; Уменьшаем ecx на 1 (количество ; аргументов без названия программы) next: cmp ecx, 0 ; проверяем, есть ли еще аргументы jz \_end ; если аргументов нет выходим из цикла ; (переход на метку \_end) pop eax ; иначе извлекаем аргумент из стека call sprintLF ; вызываем функцию печати loop next ; переход к обработке следующего ; аргумента (переход на метку next) \_end: call quit

Создайте файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введите в него текст про- граммы из листинга 8.2.



текст листинга в файле

Создайте исполняемый файл и запустите его, указав аргументы:

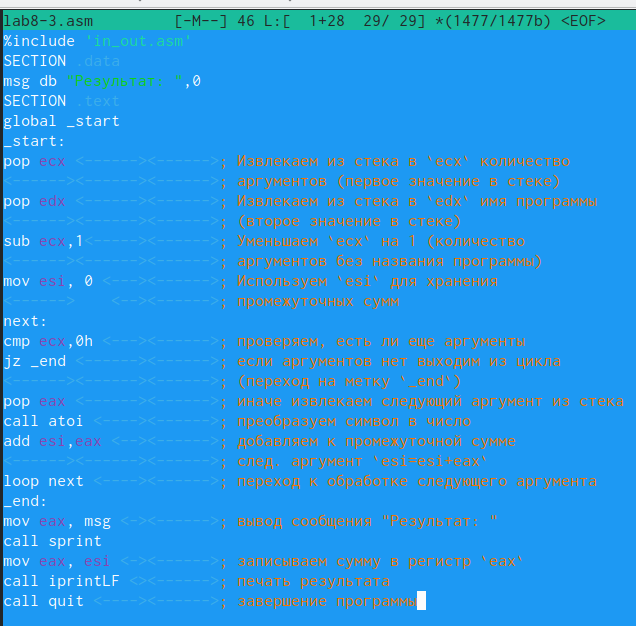


Запуск программы с указанием аргументов

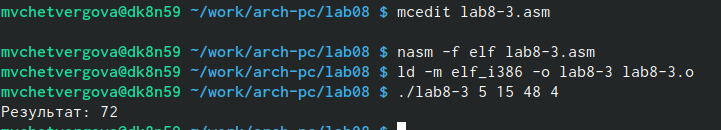
Сколько аргументов было обработано программой? Программа обработала 3 аргумента и вывела их на экран. Рассмотрим еще один пример программы которая выводит сумму чисел, которые пере- даются в программу как аргументы. Создайте файл lab8-3.asm в каталоге и введите в него текст программы из листинга 8.3.

Листинг 8.3. Программа вычисления суммы аргументов командной строки %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 add esi,eax ; добавляем к промежуточной сумме ; след. аргумент esi=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

Создайте исполняемый файл и запустите его, указав аргументы. Пример результата работы программы:

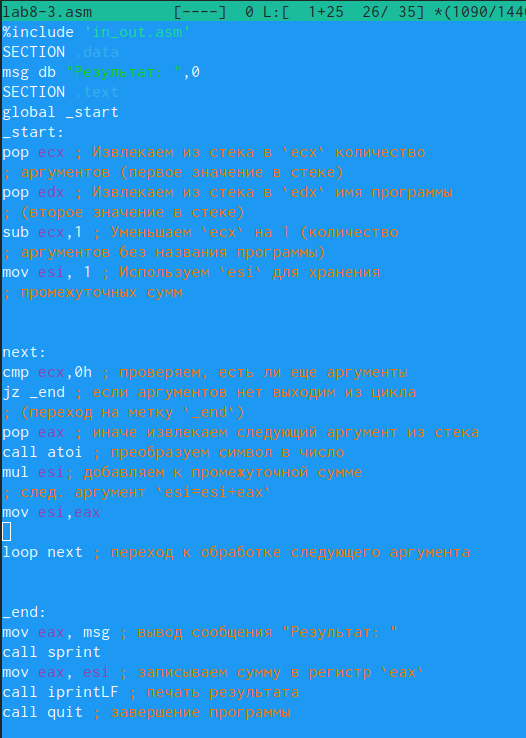


текст листинга в файле

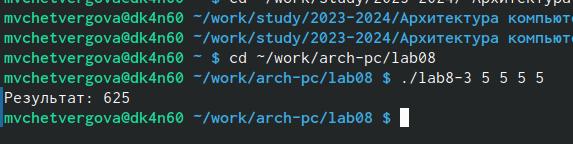


запуск программы

Измените текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.



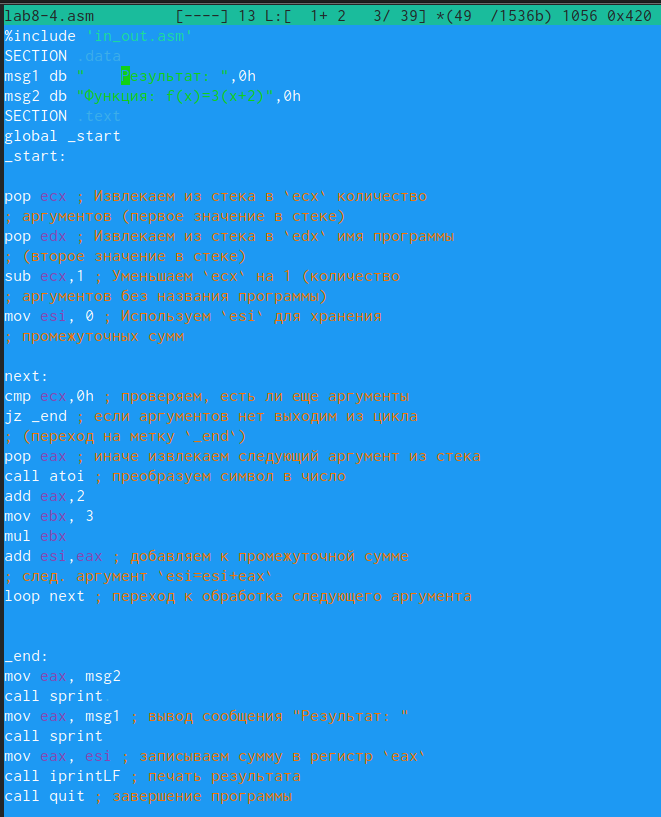
Изменение листинга программы



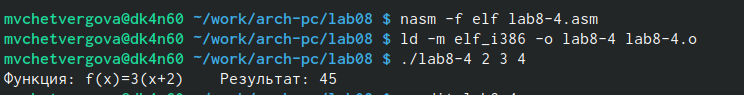
запуск программы

8.4. Задание для самостоятельной работы

1. Напишите программу, которая находит сумму значений функции f(x) для x = x1 + x2 + … + xn т.е. программа должна выводить значение f(x1) + f(x2) + … + f(xn). Значения xi передаются как аргументы. Вид функции f(x) выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы No 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах x = x1 + x2 + … + xn.



Листинг данной программы



запуск программы

# 4 Выводы

В ходе выполнения лабораторной работы номер 8, мы приобрели навыки написания программ с использованием циклов и обработкой аргументов командной строки. Целью работы является приобретение навыков работы с написанием программ с использованием циклов и обработкой аргументов командной строки. Использовали полученные знания на практике.