Отчёт по лабораторной работе №8

Дисциплина: архитектура компьютера

Байдина Елизавета Дмитриевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Приобрести навыки написания программ с использованием циклов и обработкой аргументов командной строки.

2 Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки
3. Задание для 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. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров.

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.

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.

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 обратном 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 извлечь из стека количество аргументов, а затем циклично для каждого аргумента выполнить логику программы.

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM

Создаю каталог для программам лабораторной работы № 8 (рис. 4.1)

```
edbayjdina@dk3n35 ~ $ mkdir ~/work/arch-pc/lab08
edbayjdina@dk3n35 ~ $ cd ~/work/arch-pc/lab08
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ touch lab8-1.asm
```

Рис. 4.1: Создание каталога, переход в него и создание файла

Копирую файл in_out.asm из загрузок в соответствующую папку для дальнейшей работы (рис. 4.2)

```
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ ls
in_out.asm  lab8-1.asm
```

Рис. 4.2: Копирование файла

Ввожу в lab8-1.asm текст программы из листинга 8.1 (рис. 4.3)

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

Рис. 4.3: Текст программы листинга 8.1

Создаю исполняемый файл и запускаю его (рис. 4.4)

```
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 5
5
4
3
2
1
```

Рис. 4.4: Создание и запуск файла

Изменение текста программы из листинга 8.1 (рис. 4.5)

```
%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
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
```

Рис. 4.5: Изменение текста программы из листинга 8.1

Создаю изменённый исполняемый файл и запускаю его (рис. 4.6)

```
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 6
5
3
1
```

Рис. 4.6: Запуск изменённого файла

Изменяю текст программы так, чтобы программа работала корректно (рис. 4.7)

```
%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
mov [N],eax
; ------ Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
abel:
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
```

Рис. 4.7: Изменение текста программы

Создаю исполняемый файл и запускаю его (рис. 4.8)

```
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 6
5
4
3
2
1
0
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $
```

Рис. 4.8: Создание файла и запуск

4.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm (рис. 4.9)

```
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ touch lab8-2.asm
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $
```

Рис. 4.9: Создание файла

Ввожу текст программы из листинга 8.2 в файл (рис. 4.10)

```
%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
```

Рис. 4.10: Текст программы листинга 8.2

Создаю исполняемый файл и запускаю его (рис. 4.11)

```
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент2 'аргумент 3'
аргумент1
аргумент2
аргумент 3
```

Рис. 4.11: Создание файла и запуск

Создаю файл lab8-3.asm(рис. 4.12)

```
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ touch lab8-3.asm
```

Рис. 4.12: Создание файла

Ввожу текст программы из листинга 8.3 в файл (рис. 4.13)

```
%include 'in_out.asm'
SECTION .data

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
```

Рис. 4.13: Текст пограммы из листинга 8.3

Создаю исполняемый файл и запускаю его.И вывожу сумма аргументов (рис. 4.14)

```
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47
```

Рис. 4.14: Создание файла и запуск

Изменяю текст рограммы так,чтобы она выводла произвдение аргументов (рис. 4.15)

```
%include 'in_out.asm'
SECTION .data

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
```

Рис. 4.15: Изменение программы

Создаю исполняемый файл и запускаю его (рис. 4.16)

```
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
134520832
```

Рис. 4.16: Создание файла и запуск

4.3 Задание для самостоятельной работы

Создаю файл, для программы в рамках смостоятельной работы (рис. 4.17)

```
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ touch lab8-4.asm
```

Рис. 4.17: Создание файла

Пишу программу для своего тринадцатого варианта (рис. 4.18)

lab8-1.asm × lab8-2.asm

```
%include 'in_out.asm'
SECTION .data
prim DB 'f(x)=12x-7',0
otv DB 'Результат: ',0
SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi,0
mov eax,prim
call sprintLF
next:
cmp ecx,0
jz _end
mov ebx,12
pop eax
call atoi
mul ebx
add eax,-7
add esi,eax
loop next
_end:
mov eax,otv
call sprint
mov eax,esi
call iprintLF
call quit
```

Рис. 4.18: Написание программы

Запускаю исполняемый файл (рис. 4.19)

```
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
```

Рис. 4.19: Запуск файла

Запускаю исполняемый файл (рис. 4.20)

```
edbayjdina@dk3n35 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3 4
f(x)=12x-7
Результат: 92
```

Рис. 4.20: Запуск файла

5 Выводы

В ходе выполнения данной лабораторной работы были приобретены навыки написания программ с использованием циклов и обработкой аргументов командной строки.