Отчет по лабораторной работе №8

Дисциплина: архитектура компьютера

Шония Ника Гигловна

Содержание

Список таблиц

Список иллюстраций

Цель работ

Изучение навыков написания программ с использованием циклов и обработкой аргументов командной строки. # Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки
3. Задание для самостоятельной работы

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. На рис. 8.1 показана схема организации стека в процессоре. Стек имеет вершину, адрес последнего добавленного элемента, который хранится в ре- гистре 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).

Выполнение лабораторной работы

1. Реализация циклов в NASM Создаю каталог для программам лабораторной работы № 8, перехожу в него и создаю файл lab8-1.asm

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

Ввожу в файл lab8-1.asm текст программы из листинга 8.1. Создаю исполняемый файл и проверяю его работу

```
 GNU nano 6.2     /home/nikashoniya/work/arch-pc/lab08/lab8-1.asm *
%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
```

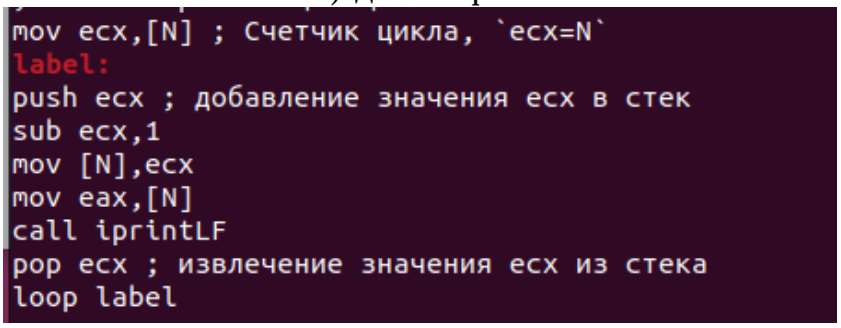
```
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-
1 lab8-1.o
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
```

Изменяю текст программы добавив изменение значение регистра ecx в цикле

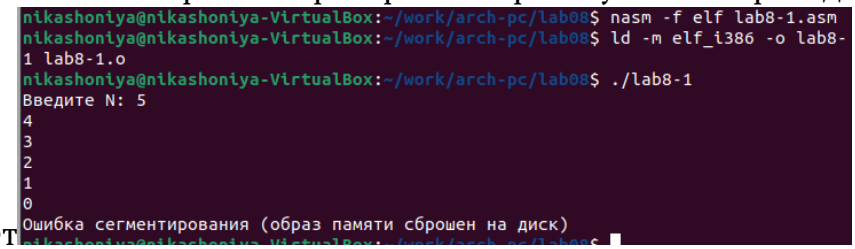
```
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
```

Создаю исполняемый файл и проверяю его работу. Число проходов цикла не

совпадает. Вношу изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop



. Создаю исполняемый файл и проверяю его работу. Число проходов цикла совпадает



2. Обработка аргументов командной строки Рассмотрим программу, которая выводит на экран аргументы командной строки в файле lab8-2.asm.

```
 GNU nano 6.2     /home/nikashoniya/work/arch-pc/lab08/lab8-2.asm *
%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
```

Создаю исполняемый файл и запускаю его, указав аргументы

```
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab08$ nasm -f
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab08$ ld -m e
2 lab8-2.o
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab08$ ./lab8-
мент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab08$
```

В файле lab8-3.asm в каталоге ~/work/arch-pc/lab08 и ввожу в него текст программы

```
 GNU nano 6.2     /home/nikashoniya/work/arch-pc/lab08/lab8-3.asm *
%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
```

Создаю исполняемый файл и запускаю его, указав аргументы

```
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/la
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/la
3 lab8-3.o
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/la
Результат: 47
```

Измените текст программы для вычисления произведения аргументов ко-

мандной строки.

```
 GNU nano 6.2     /home/nikashoniya/work/arch-pc/lab08/lab8-3.asm *
SECTION .data
msg db "Результат: ",0h
SECTION .text
global _start
_start:
pop ecx       ; Извлекаем из стека в `ecx` количество аргументов (первое значени
pop edx       ; Извлекаем из стека в `edx` имя программы (второе значение в стек
sub ecx,1     ; Уменьшаем `ecx` на 1 (количество аргументов без названия програм
mov esi, 1    ; Используем `esi` для хранения промежуточных произведений
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 ebx,esi   ;`ebx = esi`
mul ebx       ; умножаем промежуточное произведение на след. аргумент `esi=esi*e
mov esi,eax   ;`esi = eax`
loop next     ; переход к обработке следующего аргумента (переход на метку `next
_end:
mov eax, msg  ; вывод сообщения "Результат: "
```

Создаю исполняемый файл и проверяю его работу

```
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab08$
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab08$
3 lab8-3.o
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab08$
Результат: 54600
```

3. Задание для самостоятельной работы Вывожу результат программы, которая находит сумму значений функции ⊠(⊠) для ⊠ = ⊠1, ⊠2, …, ⊠⊠,

```
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-
4 lab8-4.o
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4
результат: 158
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab08$ ./lab8-4 5 6 7 8
результат: 398
```

Выводы

Были получены по организации циклов и работе со стеком на языке NASM.

Список литературы

Мой репозиторий: https://github.com/NikaShoniya/study_2023-2024_arch-pc