РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8.

дисциплина: *Архитектура компьютеров*

Студент: Подхалюзина Виолетта Михайловна

Группа: НКАбд-04-24

МОСКВА

2024 г.

Оглавление

1 Цель работы

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

2 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.

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 Выполнение лабораторной работы

3.1 Начало работы

Я создаю каталог для программ лабораторной работы №8, перехожу в него и создаю файл lab8-1.asm

```
podkhalyuzina_violetta_113224676@violetta-Mint:~$ mkdir ~/work/arch-pc/lab08
podkhalyuzina_violetta_113224676@violetta-Mint:~$ cd ~/work//arch-pc/lab08
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ touch lab8-1.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ ls
lab8-1.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$
```

Далее я ввела в файл lab8-1.asm текст программы из листинга 8.1. Создала исполняемый файл и проверила его работу

```
 GNU nano 7.2
;-------------------------------------------------
; Программа вывода значений регистра 'ecx'
;-------------------------------------------------
%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
```

```
.и
/..
 in_out.asm
 lab8-1.asm
```

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$
```

Далее я изменила текст программы, добавив изменение значение регистра ecx в цикле. Создала исполняемый файл и проверила его работу

```
 GNU nano 7.2
;-------------------------------------------------
; Программа вывода значений регистра 'ecx'
;-------------------------------------------------
%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 ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`
call quit
```

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$
```

Можем заметить, что теперь регистр ecx на каждой итерации уменьшается на 2

значения, количество итераций уменьшается вдвое.

После я внесла изменения в текст программы, добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop

```
 GNU nano 7.2
;-------------------------------------------------
; Программа вывода значений регистра 'ecx'
;-------------------------------------------------
%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:
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
```

Создала исполняемый файл и проверила его работу

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$
```

После этого мы можем сказать, что количество итераций совпадает введенному N, однако произошло смещение выводимых чисел на -1

Я создала файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввела в него текст

программы из листинга 8.2. Создала исполняемый файл и запустила его, указав аргументы

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ touch lab08-2.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ ls
in_out.asm  lab08-2.asm  lab8-1  lab8-1.asm  lab8-1.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$
```

```
  GNU nano 7.2
%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
```

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ nasm -f elf lab08-2.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab08-2 lab08-2.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ ./lab08-2 2 5 'test'
2
5
test
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$
```

Создала файл lab8-3.asm в каталоге ~/work/archpc/lab08 и ввела в него текст программы из листинга 8.3. Создала исполняемый файл и запустила его, указав аргументы

```
 GNU nano 7.2
%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
```

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ touch lab8-3.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ nano lab8-3.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$
```

Изменила текст программы из листинга 8.3 для вычисления произведения аргументов командной строки

```
 GNU nano 7.2
%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
; 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
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 сумму в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы
```

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ ./lab8-3 5 5 4
Результат: 100
```

3.2 Самостоятельная работа

```
 GNU nano 7.2
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
form db "Формула 3x - 1",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
mov eax, form
call sprintLF
mov ebx, 3
next:
cmp ecx,0h
jz _end
pop eax
call atoi
mul ebx
sub eax, 1
add esi,eax
loop next
_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
```

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ nasm -f elf sWork.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ ld -m elf_i386 -o sWork sWork.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$ ./sWork 1 2 3
Формула 3x - 1
Результат: 15
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab08$
```

Вывод: в результате выполнения данной лабораторной работы я приобрела навыки написания программ с использованием циклов а также научился обрабатывать аргументы командной строки.

4 Контрольные вопросы для самопроверки

1. Команда loop используется для организации циклов в ассемблере. Она уменьшает значение регистра ecx на единицу, после чего проверяет его значение. Если ecx не равен нулю, выполнение программы переходит на указанную метку, продолжая цикл. Если ecx равен нулю, выполнение передается следующей инструкции после команды loop.

2. Цикл можно организовать без использования команды loop, применяя команды условного перехода, такие как cmp и jnz.

3. Стек — это структура данных, работающая по принципу LIFO. Основные операции: добавление элемента в стек (push) и извлечение элемента из стека (pop). Стек используется для хранения адресов возврата, передачи аргументов, локальных переменных и временных данных.

4. Данные извлекаются из стека в обратном порядке их добавления (по принципу LIFO). Команда push уменьшает указатель стека esp, добавляя элемент, а команда pop увеличивает указатель esp, извлекая элемент.

5 Список литературы

1. GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/.

2. GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/.

3. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander. org/.

4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.

5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658.

6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.

7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.

8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.

9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.

10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.

11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.

12. Расширенный ассемблер: NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/.

13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.

14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL:

http://www.stolyarov.info/books/asm_unix.

15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).

16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).