1 №8

1.1 :

1.1.1

1.1.2

1 1 2 1 3 1 4 2 4.1
NASM 2 4.2 5 4.3 8 5 10 6
10

1.2 1

. ## 2
1. NASM. 2. . 3. .
3 — , LIFO («Last In —
First Out» « — »).
. (ss, bp, sp) .
. ,
. ,
, esp ( ).
. , , .
, — . push , . .
, esp, esp 4.
— , . pop
, . . , esp,
esp 4. , .
, “ ”,
. .
ecx. loop.
. ## 4 ### 4.1 NASM
№ 8, lab8-1.asm. (.1).

```
emshekhavcov@emshekhavcov:~$ mkdir ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/lab08
emshekhavcov@emshekhavcov:~$ cd work/study/2023-2024/Архитектура\ компьютера/arch-pc/lab08
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ touch lab8-1.asm
```

lab8-1.asm 8.1. (.2).

```
Открыть                                    lab8-1.asm
                                           ~/work/study/2023-2024/Архитектура компьютер

%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
```

. (.3).

```
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab8-1.asm
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 l
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
```

N 1 . , ecx

. (.4).

```
; ----- 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 ; Вывод значения `N`
loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`
call quit
```

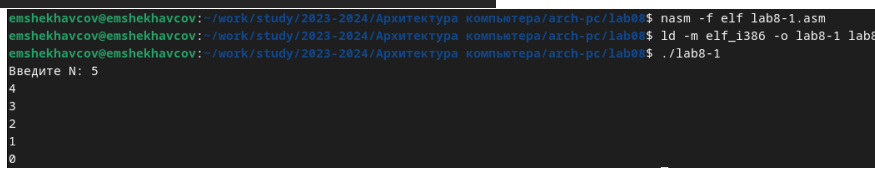
. (.5).

```
4294526570
4294526568
4294526566
4294526564
4294526562
4294526560
4294526558
4294526556
4294526554
4294526552
4294526550
4294526548
4294526546
4294526544
4294526542
4294526540
4294526538
4294526536
4294526534
4294526532
429452653
```

.

, push pop loop. (.6).

```
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; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`
call quit
```



.(.7).

N-1

0 . ### 4.2 lab8-2.asm

~/work/arch-pc/lab08 8.2. (.8).

Открыть • lab8-2.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/

```
%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
```

```
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ touch lab8-2.asm
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab8
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./lab8-2 аргумен
аргумент1
аргумент2
аргумент3
```

, . (.9). 4 , 2 , 3, - “2” . , , . lab8-3.asm ~/work/archpc/lab08 8.3. (.10).

Открыть • lab8-3.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08

```
%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`
```

```
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ touch lab8-3.a
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ nasm -f elf la
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./lab8-3 5 5 1
Результат: 20
```

, . (.11). 8.3 . (.12).

Открыть — lab8-3.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08

```
%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
imul 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
```

```
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./lab8-3 2 2 10
Результат: 40
```

, . (.13).

4.3 , f(x) = 15x + 2 (11) x = x1, x2, ..., xn. x =

Открыть • task.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08

```
%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
imul eax, 15 ; умножаем x на 15
add eax, 2 ; добавляем 2
add esi,eax ; добавляем значение функции для
; конкретного аргумента к промежуточной сумме
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
```

x1, x2, …, xn.(.14).

x = x1, x2, …, xn. (.15).

```
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ touch task.asm
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ nasm -f elf task.asm
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o task task.o
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./task
Результат: 0
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./task 1 2 3 4
Результат: 158
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./task 4 10
Результат: 214
emshekhavcov@emshekhavcov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$
```

. : %include ‘in_out.asm’ SECTION .data msg db “ :”,0 SECTION .text global _start _start: pop ecx ; ecx ; ( ) pop edx ; edx ; ( ) sub ecx,1 ; ecx 1 (;) mov esi, 0 ; esi ; next: cmp ecx,0h ; , jz _end ; ; (_end) pop eax ;

call atoi ;

imul eax, 15 ; x 15 add eax, 2 ; 2 add esi,eax ;
; loop next ; _end:
mov eax,msg ; “ :” call sprint mov eax,esi ;
`eax` call iprintLF ; call quit ; ## 5

,
. ## 6 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 . — (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 . — ISBN 978-1491941591. 7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php. 8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 . — ISBN 9781784396879. 9. . ., . . . — . : , 2018. 10. . ., . . ASSEMBLER. — . : - , 2017. 11. . . . — . : , 2016. 12. : NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/. 13. ., ., . UNIX. — 2- . — , 2010. — 656 . — ISBN 978-5-94157-538-1. 14. . NASM Unix. — 2- . — . : , 2011. — URL: http://www.stolyarov.info/books/asm_unix. 15. . . — 6- . — . : , 2013. — 874 . — (Computer Science). 16. ., . . — 4- . — . : ,2015. — 1120 . — (Computer Science).