Отчет по лабораторной работе №7

Дисциплина: Архитектура компьютера

Клименко Алена Сергеевна

Содержание

Список иллюстраций

Список таблиц

# 1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

# 2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлов листинга
3. Самостоятельное написание программ по материалам лабораторной работы

# 3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов: • условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия. • безусловный переход – выполнение передачи управления в определенную точку про- граммы без каких-либо условий.

# 4 Выполнение лабораторной работы

## 4.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы №7, перехожу в него, создаю файл lab7-1.asm и проверяю его наличие. (рис. 1).

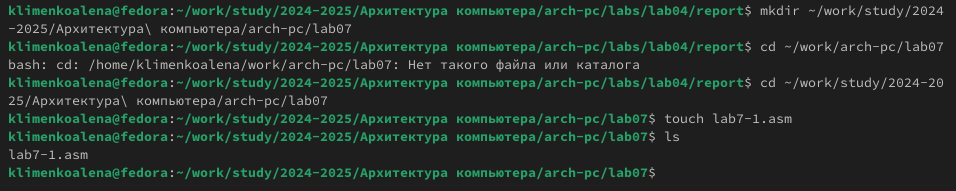


Рис. 1: Создание каталога и файла для программы

Заполняю в файл lab7-1.asm код из листинга 7.1, чтобы посмотреть как будет работать jmp. (рис. 2).

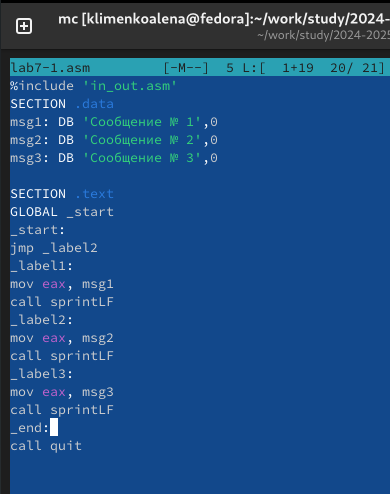


Рис. 2: код с импользованием jmp

Создаю исполняемый файл и запускаю его. Результат совпадает с тем, что находится в лабораторной работе. (рис. 3).

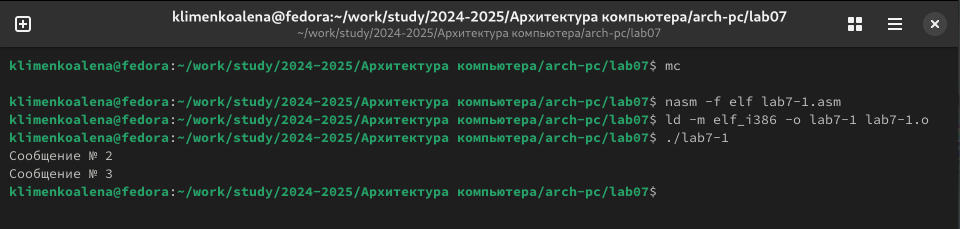


Рис. 3: работа кода

После изменения кода в файле запускаю его. (рис. 4).

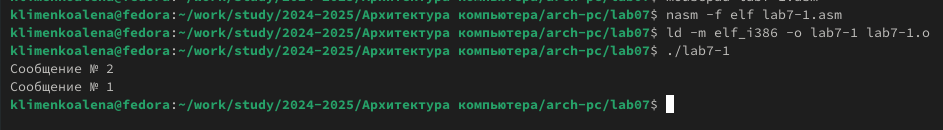


Рис. 4: работа кода

Далее меняю код, чтобы он выводил то, что указано в задании. (рис. 5).

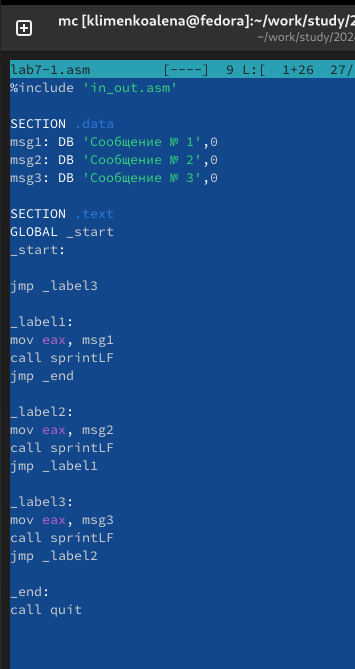


Рис. 5: код

%include 'in\_out.asm'  
  
SECTION .data  
msg1: DB 'Сообщение № 1',0  
msg2: DB 'Сообщение № 2',0  
msg3: DB 'Сообщение № 3',0  
  
SECTION .text  
GLOBAL \_start  
\_start:  
  
jmp \_label3  
  
\_label1:  
mov eax, msg1  
call sprintLF  
jmp \_end  
  
\_label2:  
mov eax, msg2  
call sprintLF  
jmp \_label1  
  
\_label3:  
mov eax, msg3  
call sprintLF  
jmp \_label2  
  
\_end:  
call quit

Проверяю корректность написания и вывода. (рис. 6).

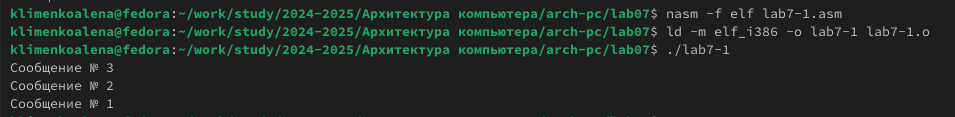


Рис. 6: работа кода

Создаю файл lab7-2.asm и после изучения листинга 7.3 ввожу код в файл. (рис. 7).

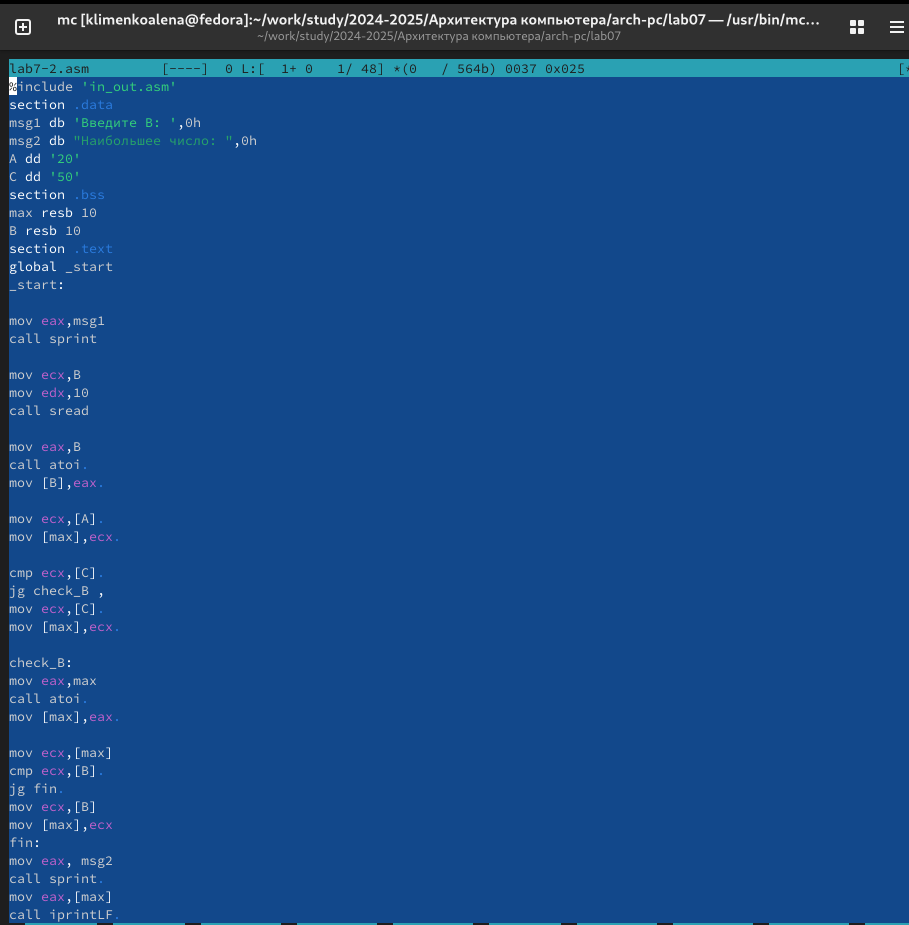


Рис. 7: код

Несколько раз проверяю корректность работы. (рис. 8).

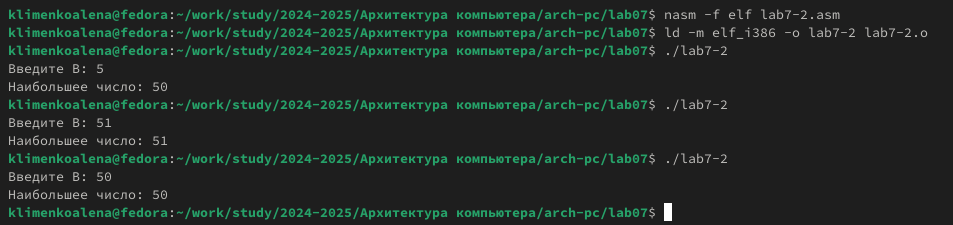


Рис. 8: работа кода

## 4.2 Изучение структуры файла листинга

Открываю файл листинга с помощью текстового редактора mousepad. (рис. 9).

В листинге есть три столбца, не считая первый, в котором просто номер строки. Первый отвечает за адрес строки в файле asm, номер строки указан в шестнадцетеричной системе. Во втором столбце находится машинный код. В третьем находится исходный текст программы. Рассмотрю это на конкретных строках кода, например, 8,9,10 8,9,10 - номер строки в листинге 00000003, 00000006, 00000009 - обозначают на какой строке кода находится команда, последовательность не постоянна потому что в коде есть пустые строки 803800, 7403, 40 - обозначение команды машинным кодом cmp… ,0, jz finished, inc eax - текст программы

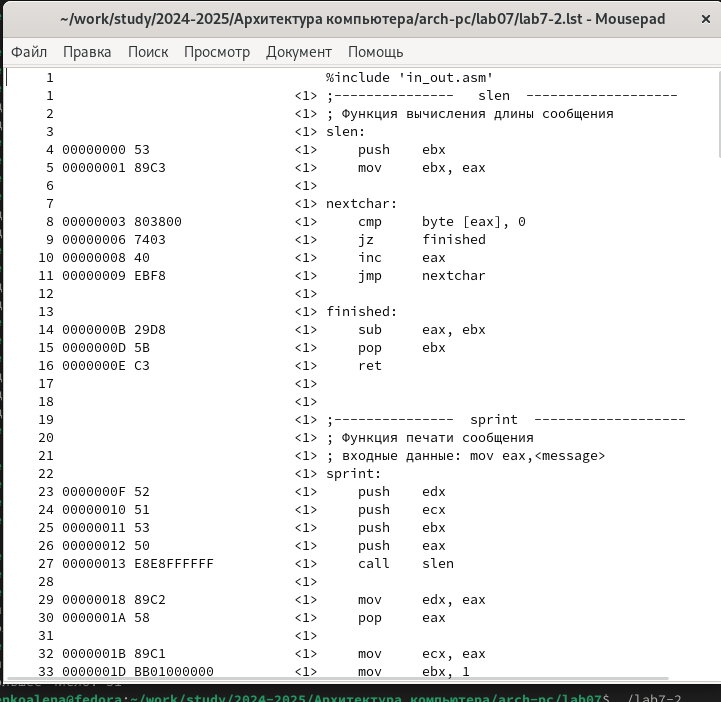


Рис. 9: Проверка файла листинга

Убираю один операнд. (рис. 10).

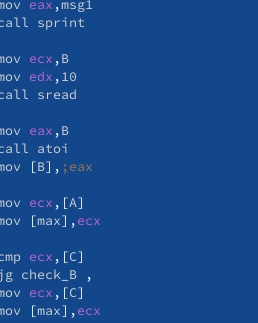


Рис. 10: Удаление операнда из программы

В листинге добавляетс отображение ошибки. (рис. 11).

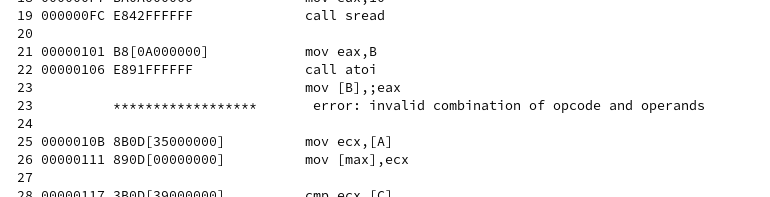


Рис. 11: Просмотр ошибки в файле листинга

## 4.3 Задания для самостоятельной работы

Как я поняла я должна решить такой же вариант как и в 6 лабораторной работе, это - 2 вариант Написала программу для нахождения наименьшего числа из 3. (рис. 12). во время написания кода, у меня возникли сомнения надо ли вводить число B с клавиатуры, поэтому я сделала два варианта написания кода. (рис. 13).

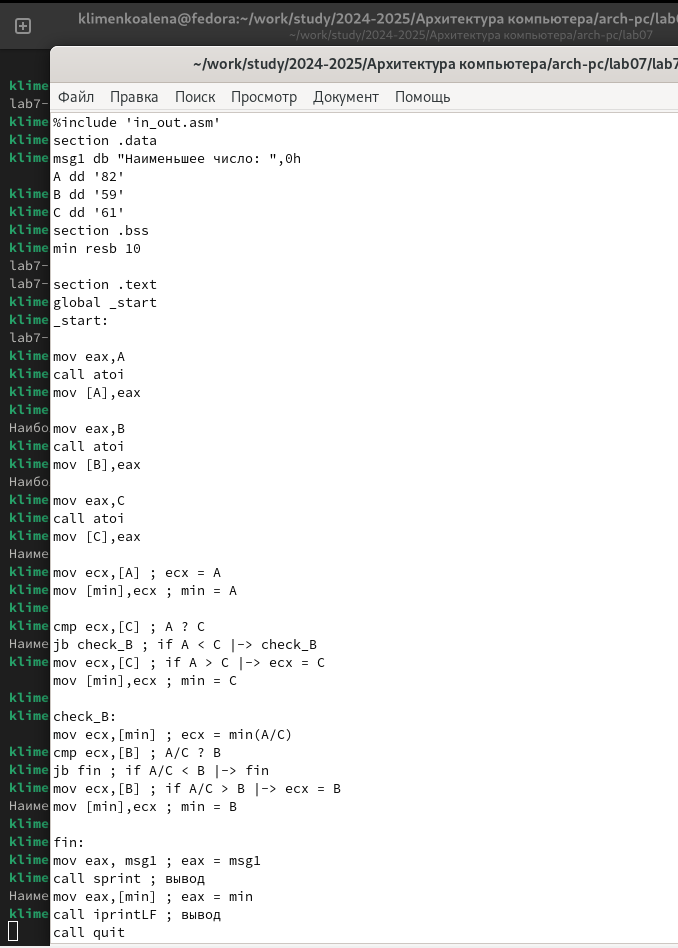


Рис. 12: первый вариант кода

Код первой программы:

%include 'in\_out.asm'  
section .data  
msg1 db "Наименьшее число: ",0h  
A dd '82'  
B dd '59'  
C dd '61'  
section .bss  
min resb 10  
  
section .text  
global \_start  
\_start:  
  
mov eax,A  
call atoi   
mov [A],eax   
  
mov eax,B  
call atoi   
mov [B],eax   
  
mov eax,C  
call atoi   
mov [C],eax   
  
mov ecx,[A] ; ecx = A  
mov [min],ecx ; min = A  
  
cmp ecx,[C] ; A ? C  
jb check\_B ; if A < C |-> check\_B  
mov ecx,[C] ; if A > C |-> ecx = C  
mov [min],ecx ; min = C  
  
check\_B:  
mov ecx,[min] ; ecx = min(A/C)  
cmp ecx,[B] ; A/C ? B  
jb fin ; if A/C < B |-> fin  
mov ecx,[B] ; if A/C > B |-> ecx = B  
mov [min],ecx ; min = B  
  
fin:  
mov eax, msg1 ; eax = msg1  
call sprint ; вывод  
mov eax,[min] ; eax = min  
call iprintLF ; вывод  
call quit

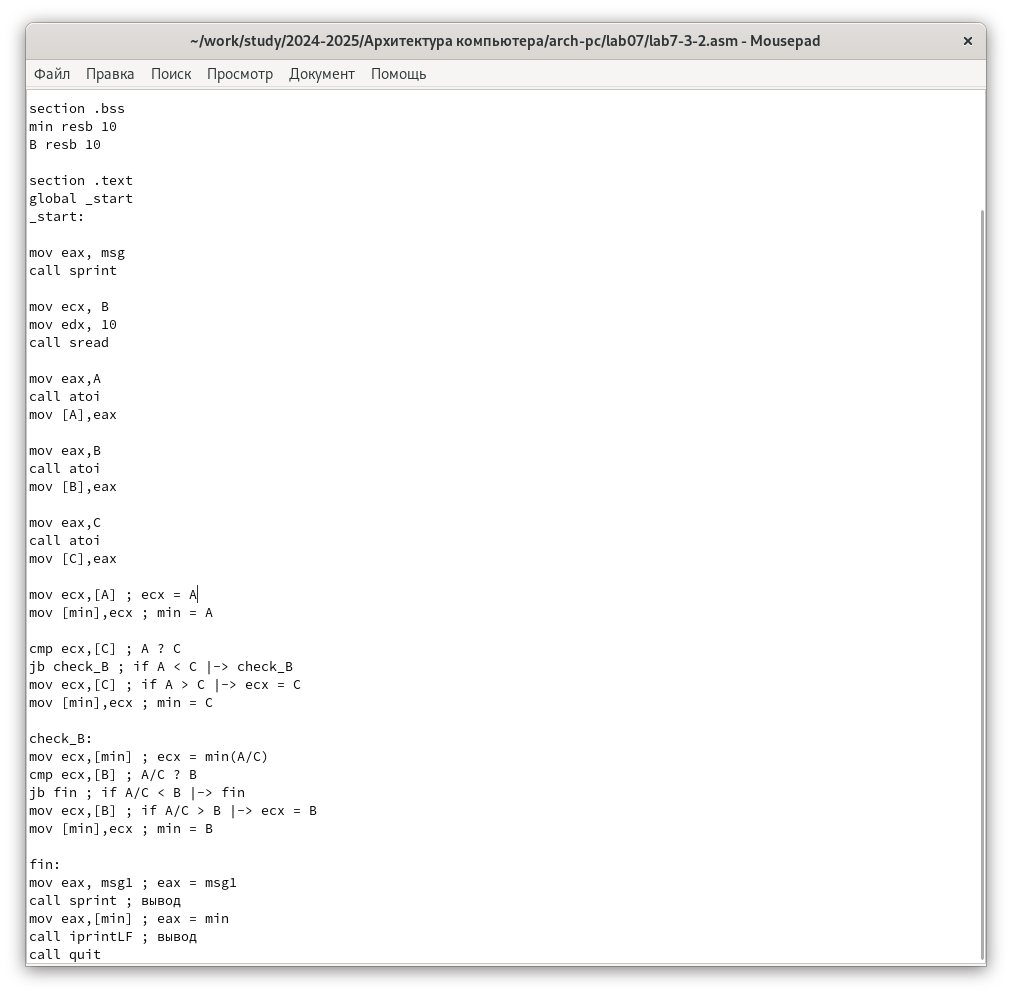


Рис. 13: второй вариант кода

%include 'in\_out.asm'  
section .data  
msg db 'Введите B: ', 0h  
msg1 db 'Наименьшее число: ',0h  
A dd '82'  
  
C dd '61'  
  
section .bss  
min resb 10  
B resb 10  
  
section .text  
global \_start  
\_start:  
  
mov eax, msg  
call sprint  
  
mov ecx, B  
mov edx, 10  
call sread  
  
mov eax,A  
call atoi   
mov [A],eax   
  
mov eax,B  
call atoi   
mov [B],eax   
  
mov eax,C  
call atoi   
mov [C],eax   
  
mov ecx,[A] ; ecx = A  
mov [min],ecx ; min = A  
  
cmp ecx,[C] ; A ? C  
jb check\_B ; if A < C |-> check\_B  
mov ecx,[C] ; if A > C |-> ecx = C  
mov [min],ecx ; min = C  
  
check\_B:  
mov ecx,[min] ; ecx = min(A/C)  
cmp ecx,[B] ; A/C ? B  
jb fin ; if A/C < B |-> fin  
mov ecx,[B] ; if A/C > B |-> ecx = B  
mov [min],ecx ; min = B  
  
fin:  
mov eax, msg1 ; eax = msg1  
call sprint ; вывод  
mov eax,[min] ; eax = min  
call iprintLF ; вывод  
call quit

Проверяю корректность работы кода первого варианта. (рис. 14).

Первая программа самостоятельной работы

Рис. 14: Первая программа самостоятельной работы

Проверяю корректность работы кода второго варианта. (рис. 15).

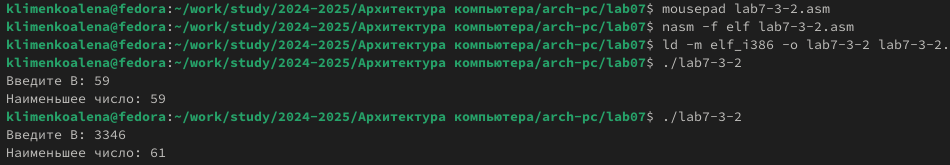


Рис. 15: Вторая программа самостоятельной работы

Написала программу, которая для двух введенных с клавиатуры значение вычисляет требуемое значение и выводит результат. (рис. 16).



Рис. 16: код

Код второй программы:

%include 'in\_out.asm'  
SECTION .data  
msg1: DB 'Введите значение переменной x: ',0  
msg2: DB 'Введите значение переменной a: ',0  
res: DB 'Результат: ',0  
SECTION .bss  
x: RESB 80  
a: RESB 80  
SECTION .text  
GLOBAL \_start  
\_start:  
  
  
mov eax, msg1  
call sprint  
mov ecx, x  
mov edx, 80  
call sread  
mov eax,x  
call atoi  
mov edi,eax   
  
mov eax, msg2  
call sprint  
mov ecx, a  
mov edx, 80  
call sread  
mov eax,a  
call atoi  
mov esi,eax  
  
  
cmp esi,edi  
jg var2 ; a > x |-> var2  
  
mov eax,x  
call atoi  
  
add eax, -1  
mov edi,eax ; edi = aex  
jmp fin  
  
var2:  
mov eax,a  
call atoi  
add eax, -1  
mov edi, eax ; edi = aex  
  
  
fin:  
mov eax,res ; eax = res  
call sprint ; строка  
mov eax,edi ; eax = edi  
call iprintLF  
call quit

Проверяю корректность работы кода. (рис. 17).

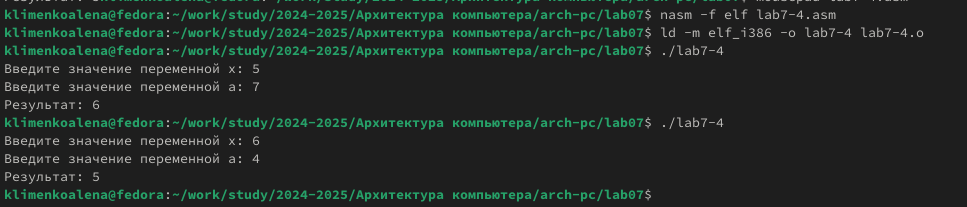


Рис. 17: Вторая программа самостоятельной работы

# 5 Выводы

При выполнении лабораторной работы я изучила команды условных и безусловных переходов, а также приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файлов листинга.

# Список литературы

1. [Курс на ТУИС](https://esystem.rudn.ru/course/view.php?id=112)
2. [Лабораторная работа №7](https://esystem.rudn.ru/pluginfile.php/2089087/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%967.%20%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B%20%D0%B1%D0%B5%D0%B7%D1%83%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%B8%20%D1%83%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%BF%D0%B5%D1%80%D0%B5%D1%85%D0%BE%D0%B4%D0%BE%D0%B2%20%D0%B2%20Nasm.%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%B2%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9.pdf)