**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования «Московский авиационный институт (национальный исследовательский университет)»**

## Факультет №3 «Системы управления, информатика и электроэнергетика»

Кафедра №[316 «Системное моделирование и автоматизированное проектирование](http://mai.ru/education/control/316/)»

# Отчет по лабораторной работе №1

По курсу: “Архитектура ЭВМ и систем”

Выполнил Вакурин А.О

Студент группы: 3-3ВТИ-4ДБ-009-14

Проверил: Бутко А.О

Москва 2017

**Оглавление**

[Введение 3](#_Toc465721296)

[Основная часть 4](#_Toc465721296)

[Задача 1 7](#_Toc465721296)

[Задача 2 8](#_Toc465721296)

[Задача 3 9](#_Toc465721296)

[Задача 4 10](#_Toc465721296)

[Задача 5 11](#_Toc465721296)

[Задача 6 12](#_Toc465721296)

[Задача 7 13](#_Toc465721296)

[Задача 8 14](#_Toc465721296)

[Литерутура 16](#_Toc465721296)

**Введение**

Электронно-вычислительные машины (ЭВМ), или, как их теперь чаще называют, компьютеры, - одно из самых удивительных творений человека. В узком смысле ЭВМ - это приспособления, выполняющие разного рода вычисления или облегчающие этот процесс. Простейшие устройства, служащие подобным целям, появились в глубокой древности, несколько тысячелетий назад. По мере развития человеческой цивилизации они медленно эвоционировали, непрерывно совершенствуясь. Однако только в 40-е годы нашего столетия было положено начало созданию компьютеров современной архитектуры и с современной логикой. Именно эти годы можно по праву считать временем рождения современных вычислительных машин.

В своем историческом докладе, опубликованном в 1945 году, Джон фон Нейман выделил и детально описал пять ключевых компонентов того, что ныне называют " архитектурой фон Неймана " современного компьютера.

Чтобы компьютер был и эффективным, и универсальным инструментом, он должен включать следующие структуры: центральное арифметико-логическое устройство (АЛУ), центральное устройство управления (УУ), " дирижирующее " операциями, запоминающее устройство, или память, а также устройства ввода-вывода информации.

Фон Нейман отмечал, что эта система должна работать с двоичными числами, быть электронным, а не механическим устройством и выполнять операции последовательно, одну за другой.

Принципы, сформированные фон Нейманом, стали общепринятыми и положены в основу как больших ЭВМ первых поколений, так и более поздних мини- и микро-ЭВМ. И хотя в последнее время идут активные поиски вычислительных машин, построенных на принципах, отличных от классических, большинство компьютеров построено согласно принципам, определенным Нейманом.

**Основная часть**

Для прямого использования возможностей и особенностей конкретной вычислительной системы необходимо применять низкоуровневое программирование. Это язык программирования близкий к программированию непосредственно в [машинных кодах](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) используемого реального или виртуального (например, Java, Microsoft .NET) процессора. Для обозначения машинных команд обычно применяется мнемоническое обозначение. Это позволяет запоминать команды не в виде последовательности двоичных нулей и единиц, а в виде осмысленных сокращений слов человеческого языка (обычно английских).

Для того чтобы писать программы на этом уровне, необходимо знать архитектуру аппаратной части системы:

* структуру и функционирование системы в целом;
* организацию оперативной памяти;
* состав внешних устройств, их адреса и форматы регистров;
* организацию и функционирование процессора,
* состав и форматы его регистров, способы адресации,
* систему команд;
* систему прерываний и т. д.

При этом не следует забывать, что вычислительная система – это совокупность не только аппаратных, но и программных средств, и особенности имеющегося программного обеспечения (операционной системы) оказывают существенное влияние на разработку программ.

В истории развития программирования существовали три разновидности низкоуровневых языков, последовательно сменивших друг друга:

* машинный код;
* мнемокод;
* ассемблер.

В машинных кодах программа представляется в виде последовательности чисел, являющихся кодами команд процессора, адресами оперативной памяти, номерами регистров процессора и внешних устройств и т. д. Фрагмент последовательности кодов команд микропроцессора Intel 8086 для добавления двухбайтового слова с адресом 36 к слову с адресом 32:

Адрес

Код

0000 A1

0001 20

Очевидно, что в таком виде создание сколько-нибудь сложной программы является весьма трудной задачей. Для решения этой задачи были разработаны так называемые мнемокоды, которые и явились основой для современных ассемблеров. Они вместо чисел позволяли использовать мнемонические (символьные) имена, отражающие смысл выполняемой команды. Приведенная выше последовательность команд в мнемонической записи:

Mov ax ,ds:[32]

Add ax,ds:[36]

Mov ds:[32],ax

В этом фрагменте уже виден смысл команд, из которых складывается сложение двух слов памяти. Ассемблер отличается от своего предшественника – мнемокода обширным набором директив транслятора, существенно упрощающих процесс кодирования программы, в первую очередь директивами оформления программы в виде логически законченных элементов и макросредствами.

В наших лабораторных работах будет использоваться язык программирования – ассемблер. Ассемблер — компилятор с языка ассемблера в команды машинного языка. На сегодняшний день ассемблер как язык программирования, предназначенный для создания программ, используется крайне редко. Это связано с тем, что этот процесс является чрезвычайно трудоемким и получающиеся программы являются системно-зависимыми, их нельзя свободно переносить между компьютерами, имеющими различные аппаратные архитектуры и операционные системы. Поэтому на сегодняшний день ассемблер изучают в учебных целях, так, как только он дает полное представление о реальном устройстве и функционировании аппаратуры и операционной системы. Сейчас разработка программ на ассемблере применяется в основном в программировании небольших микропроцессорных систем (микроконтроллеров), как правило, встраиваемых в какое-либо оборудование. Очень редко возникает потребность использования ассемблера в разрабатываемых программах в тех случаях, когда необходимо, например, получить наивысшую скорость выполнения определенного участка программы, выполнить операцию, которую невозможно реализовать средствами языков высокого уровня, либо уместить программу в память ограниченного объема (типичное требование для загрузчиков операционной системы). Под каждую архитектуру процессора и под каждую ОС или семейство ОС существует свой ассемблер. Есть также так называемые кросс-ассемблеры, позволяющие на машинах с одной архитектурой (или в среде одной ОС) ассемблировать программы для другой целевой архитектуры или другой ОС и получать исполняемый код в формате, пригодном к исполнению на целевой архитектуре или в среде целевой ОС. Язык ассемблера— тип языка программирования низкого уровня. Команды языка ассемблера один в один соответствуют командам процессора и представляют собой удобную символьную форму записи (мнемокод) команд и аргументов. Язык ассемблера обеспечивает связывание частей программы и данных через метки, выполняемое при ассемблировании (для каждой метки).

**Задача 1**

Листинг программы

text segment 'code' ; начало описания сегмента команд

assume CS:text, DS:text ; начало инициализации сегментных регистров

begin: ; метка начала программы

mov ax,text ; загрузка адресса сегмента команд в регистр ds через ax. из ax в ds

mov ds,ax ;

mov ah,09h ; загрузка номера функции для вывода текста на экран. 09 функция

mov dx,offset message; загрузка переменной для вывода

int 21h ; выполнение функции

mov ah,4ch; загрузка номера функции выхода из программы

mov al,00h ; загрузка кода выхода. программа успешно завершена. код 0

int 21h ; выполнение функции

message db 'Hello world!$'; объявление переменной message. $-конец строки

text ends ; конец описания сегмента команд

end begin; указание метки конца программы

Результат работы программы представлен на рисунке 1.

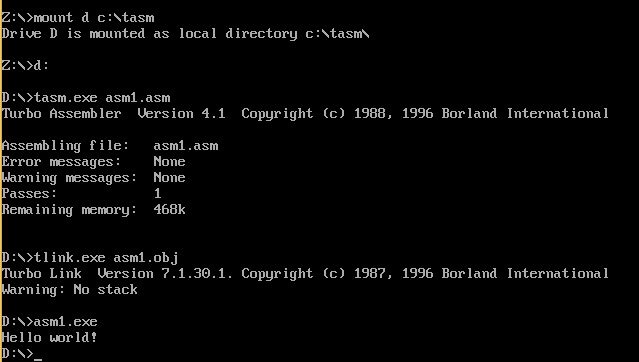
****

Рис1. Задача 1

**Задача 2**

Листинг программы

text segment 'code' ; начало описания сегмента команд

assume CS:text, DS:text ; начало инициализации сегментных регистров

begin: ; метка начала программы

mov ax,data ; загрузка адресса сегмента команд в регистр ds через ax. из ax в ds

mov ds,ax ; ^

mov ah,09h ; загрузка номера функции для вывода текста на экран. 09 функция

mov dx,offset message ; загрузка переменной для вывода

int 21h ; выполнение функции

mov ax,4c00h ; загрузка номера функции выхода из программы

int 21h ; выполнение функции

text ends ;

data segment ;

message db 'Hello world!$' ; объявление переменной message. $-конец строки

data ends; конец описания сегмента команд

end begin ; указание метки начала программы

Результат работы программы представлен на рисунке 2.



Рис2. Задача 2

**Задача 3**

Листинг программы

text segment 'code' ; начало описания сегмента команд

assume CS:text, DS:text ; начало инициализации сегментных регистров

begin: ; метка начала программы

mov ax,data ; загрузка адреса сегмента команд в регистр ds через ax. из ax в ds

mov ds,ax ; ^

mov ah,09h ; загрузка номера функции дл¤ вывода текста на экран. 09 функци¤

mov dx,offset message; загрузка переменной дл¤ вывода

int 21h ; выполнение функции

mov ax,4c00h ; загрузка номера функции выхода из программы

int 21h ; выполнение функции 4с

text ends ; конец описание сегмента команд

data segment ; начало описание сегмента данных

message db 9,'Hello world!',10,13 ; 9-табуляция. db-текстовая переменная

db 9,'Welcome fellas',10,13 ; 1 строчка имеет 5 строчек текста.

db 9,'This is a multiline',10,13 ; строка, разделенная табуляцией и переходом на новую строчку

db 9,'formated text',10,13 ; 10,13-переход на новую строчку.

db 9,'diplaying program$' ; объявление переменной message. $-конец строки

data ends ; конец описание сегмента данных

stk segment stack 'stack'; начало описание сегмента стека

dw 128 dup(0); выделение памяти 128 ячеек типа "ворд". все ячейки заполнены нулем

stk ends ; конец описание сегмента стека

end begin; указание метки конца программы

Результат работы программы представлен на рисунке 3.

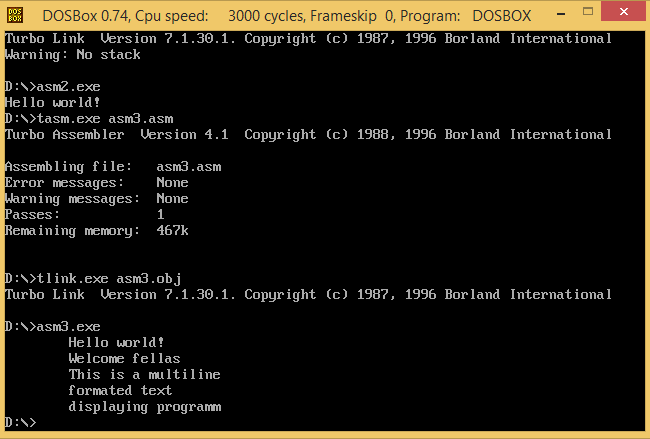


Рис3. Задача 3

**Задача 4**

Листинг программы

text segment 'code' ; начало описания сегмента команд

assume CS:text, DS:text ; начало инициализации сегментных регистров

begin: ; метка начала программы

mov ax,data ; загрузка адреса сегмента команд в регистр ds через ax. из ax в ds

mov ds,ax

mov ah,09h ; загрузка номера функции для вывода текста на экран. 09 функции

mov dx,offset message ; загрузка переменной для вывода

int 21h ; выполнение функции

mov ax,4c00h ; загрузка номера функции выхода из программы

int 21h ; выполнение функции 4с

text ends ; конец описания сегмента команд

data segment ; начало описания сегмента данных

message db 80\*25 dump(' '),10,13 ; массив 80\*25 для очистки экрана

db 9,'Hello world!',10,13 ; 9-табул¤ци¤. db-текстовая переменная

db 9,'Welcome fellas',10,13 ; 1 строчка имеет 5 строчек текста.

db 9,'This is a multiline',10,13 ; строка разделенная табул¤цией и переходом на новую строчку

db 9,'formated text',10,13 ; 10,13-переход на новую строчку.

db 9,'diplaying program$' ; объявление переменной message. $-конец строки

data ends ; конец описания сегмента данных

stk segment stack 'stack' ; начало описания сегмента стека

dw 128 dup(0); выделение памяти 128 ячеек типа "ворд". все ячейки заполнены нулем

stk ends ; конец описания сегмента стека

end begin; указание метки конца программы

Результат работы программы представлен на рисунке 4.

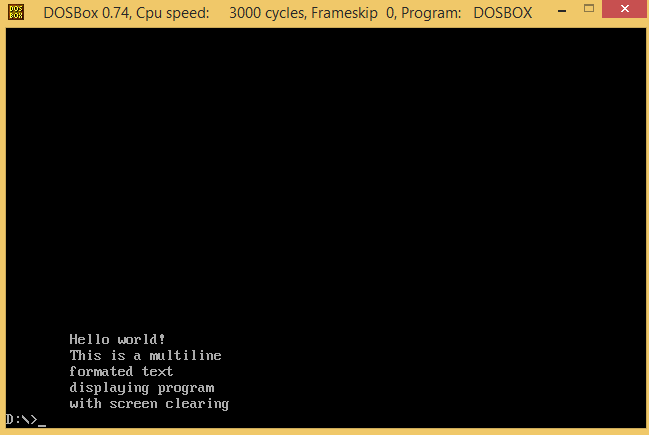


Рис4. Задача 4

**Задача 5**

Листинг программы

text segment 'code' ; начало описания сегмента команд

assume CS:text, DS:data ; начало инициализации сегментных регистров

begin: ; метка начала программы

mov ax,data ; загрузка адреса сегмента команд в регистр ds через ax. из ax в ds

mov ds,ax ; ^

mov cx,223 ; установка числа итерация для цикла

mov si,0 ; установка начального индекса массива

mov al,32 ; загрузка начального символа массива

fill: ; метка начала цикла

mov symbol[si],al ; загрузка символа в массив на позицию si

inc al ; увеличение номера символа на 1

inc si ; увеличение номера позиции на 1

loop fill ; закрытие цикла

mov ah,40h ; загрузка номера функции вывода на стандартное устройство

mov bx,1 ; установка номера устройства. 1-экран

mov cx,223 ; количество символов дл¤ вывода на экран

mov dx,offset symbol ; загрузка адреса переменной, из которой символы будут выводиться

int 21h ; выполнение функции номер 40

mov ax,4c00h ; выход из программы

int 21h ; выполнение функции 4c

text ends ; конец описания сегмента команд

data segment ; начало описания сегмента данных

symbol db 223 dup ('\*') ; массив из 223 элементов, заполненных \*

data ends ; конец описания сегмента данных

stk segment stack 'stack' ; начало описания сегмента стека

dw 128 dup(0) ; выделение памяти 128 ячеек типа "ворд". все ячейки заполнены нулем

stk ends ; конец описание сегмента стека

end begin ; указание метки конца программы

Результат работы программы представлен на рисунке 5.

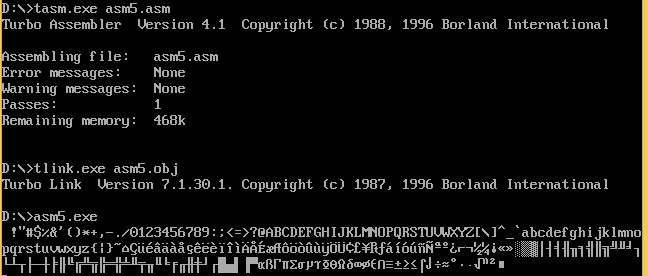


Рис5. Задача 5

**Задача 6**

Листинг программы

text segment 'code' ; начало описания сегмента команд

assume CS:text, DS:data ; начало инициализации сегментных регистров

begin: ; метка начала программы

mov ax,data ; загрузка адресса сегмента команд в регистр ds через ax. из ax в ds

mov ds,ax ; ^

mov ah,02h ; загрузка номера функции вывода символа на экран

mov dl, '>' ; загрузка символа для вывода

int 21h ; выполнение функции 02

mov ah,3fh ; загрузка номера функции, чтение данных с клавиатуры

mov bx,0 ; установка минимального количества символов

mov cx,128 ; установка максимального количества символов

mov dx,offset inbuf ; установка адреса переменной, куда будет записываться символ

int 21h ; выполнение символа 3f

mov ah,40h ; загрузка номера функции дл¤ вывода на стандартное устройство

mov bx,1 ; загрузка номера устройства. 1-экран

mov cx,actlen ; количество символов для вывода

mov dx,offset inbuf ; загрузка адреса переменной для вывода

int 21h ; выполнение функции

mov ax,4c00h ; выход из программы

int 21h ; выполнение функции выхода из программы

text ends ; конец описания сегмента команд

data segment ; начало описания сегмента данных

actlen dw 0 ; переменна¤ типа "ворд"

inbuf db 128 dup ('\*') ; массив из 128 символов дл¤ чтени¤ данных с клавиатуры, заполненных \*

data ends ; конец описания сегмента данных

stk segment stack 'stack' ; начало описания сегмента стека

dw 128 dup(0); выделение памяти 128 ячеек типа "ворд". все ячейки заполнены нулем

stk ends ; конец описания сегмента стека

end begin ; указание метки конца программы

Результат работы программы представлен на рисунке 6

****

Рис6. Задача 6

**Задача 7**

Листинг программы

text segment 'code' ; начало описания сегмента команд

assume CS:text, DS:data ; начало инициализации сегментных регистров

begin: ; метка начала программы

mov ax,data ; загрузка адреса сегмента команд в регистр ds через ax. из ax в ds

mov ds,ax

input: ; метка начала ввода символов

mov ah,09h ; загрузка номера функции вывода на экран

mov dx,offset promt ; загрузка адреса переменной дл¤ вывода на экран

int 21h ; выполнение функции 09

mov ah,01h ; загрузка номера функции 01 чтени¤ символа с клавиатуры

int 21h ; выполнение функции 01

cmp al,'9' ; сравнение введенного символа с 9

ja error ; переход на метку error, если больше, чем 9

cmp al,'0' ; сравнение введенного символа с 0

jb error; переход на метку error, если меньше 0

mov si,18 ; загрузка индекса дл¤ записи введенного символа

mov inbuf(ai),al ; запись введенного символа в переменную на заданную позицию

mov ah,40h ; установка номера функции для вывода на стандартное устройство

mov bx,1 ; загрузка номера устройства. 1-экран

mov cx,19; загрузка количества символов для вывода

mov dx, offset inbuf ; загрузка адреса переменной

int 21h ; выполнение функции 40

mov ax,4c00h ; выход из программы

int 21h ; выполнение функции выхода из программы

error: ; начало блока обработки ошибки

mov sh,09h; вывод переменной message с помощью функции 09

mov dx,offset massage; загрузка адреса переменной

int 21h ; выполнение функции 09

jmp input ; переход на метку ввода символа

text ends ; конец описания сегмента команд

data segment ; начало описания сегмента данных

message db 10,13,'Incorrect symbol$' ; сообщение об ошибке

pront db 10,13,'Enter Symbol 0-9>$' ; запрос на ввод символа 0-9

inbuf db 10,13,'Entered symbol >\*' ; куда будет записан введенный символ вместо \*

data ends ; конец описание сегмента данных

stk segment stack 'stak' ; начало описания сегмента стека

dw 128 dup(0) ; выделение памяти 128 ячеек типа "ворд". все ячейки заполнены нулем

stk ends ; конец описание сегмента команд

end begin ; метка указание конца программы

Результат работы программы представлен на рисунке 7.

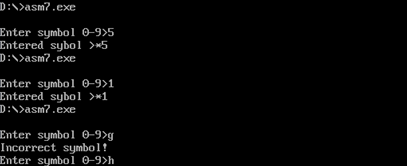


Рис7. Задача 7

**Задача 8**

Листинг программы

text segment 'code' ; начало описания сегмента команд

assume CS:text, DS:text ; начало инициализации сегментных регистров

begin: ; метка начала программы

mov ax,data ; загрузка адреса сегмента команд в регистр ds через ax. из ax в ds

mov ds,ax ; ^

input: ; метка начала ввода символов

mov ah,09h ; загрузка номера функции вывода на экран

mov dx,offset promt ; загрузка адреса переменной для вывода на экран

int 21h ; выполнение функции 09

mov bx,0 ; установка начальной позиции для ввода символов в элемент

pass: ; начало блока считывания символов

mov ah,08h ; установка номера функции 08 считывания символов без отображения

int 21h ; выполнение функции 08

cmp al,13 ; сравнение введенного символа с кнопкой Enter

je compare ; если равно, то переход на метку compare

mov string[bx],al ; если не перешли, то выполняется запись введенного символа на позицию bx

inc bx ; увеличение позиции на 1

jmp pass ; переход на метку pass для считыания очередного символа

compare ; начала блока сравнения введенной строки с шаблоном

push ds ; инициализация сегментных регистров

pop es

mov si,offset string ; загрузка адреса введенной строки

mov di,offset password; загрузка адреса шаблона

cld ; установка направления обработки

mov cx,bx ; загрузка количества символов для сравнения

repe cmpsb ; сравнения символов

je pass2 ; переход на метку pass2, если все символы равны

error: ; начало блока обработки ошибки

mov ah,09h ; вывод переменной message с помощью функции 09

mov dx,offset massage ; загрузка адреса переменной

int 21h ; выполнение функции 09

jmp input ; переход на метку input для ввода символов

pass2: ; блок вывода сообщения по сравнению пароля

mov ah,09h ; вывод сообщения с помощью функции 09

mov dx,offset message ; загрузка адреса переменной

int 21h ; выполнение функции 09

mov ax,4x00h ; выход из программы

int 21h ; выполнение функции выхода

text ends ; конец описания сегмента команд

data segment ; начало описания сегмента данных

passok bd 10,13,'Password accepted!$' ; сообщение пароль принят

message db 10,13,'Incorrect password!$' ; ссобщение об ошибке

promt db 10,13,'Enter password>$' ; запрос на ввод пароля

string db 80 dup (?) ; переменная, в которую считывается пароль

password db 'password' ; шаблон, с которым сравнивается пароль

data ends ; конец описания сегмента данных

stk segment stack 'stak' ; начало описания сегмента стека

dw 128 dup(0) ; выделение памяти 128 ячеек типа "ворд". все ячейки заполнены нулем

stk ends ; конец описания сегмента команд

end begin ; метка указания конца программы

Результат работы программы представлен на рисунке 8.

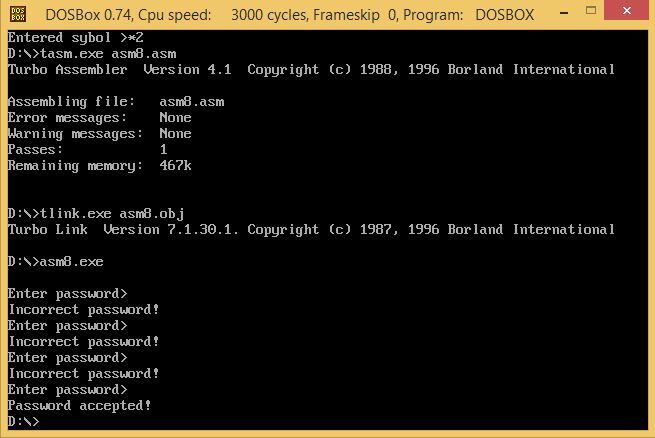


Рис8. Задача8

**Литература**

1. Аблязов, Р. Программирование на ассемблере на платформе x86-64 / Р. Аблязов. - М.: Книга по Требованию, 2011. - 302 c.  
2. Брэй, Барри Применение микроконтроллеров PIC18. Архитектура, программирование и построение интерфейсов с применением С и ассемблера (+ CD-ROM) / Барри Брэй. - М.: МК-Пресс, Корона-Век, 2008. - 576 c.  
3. Галисеев, Г. В. Ассемблер для Win 32. Самоучитель / Г.В. Галисеев. - М.: Вильямс, 2007. - 368 c.  
4. Жуков, А. Ассемблер / А. Жуков, А. Авдюхин. - М.: БХВ-Петербург, 2012. - 448 c.  
5. Иванов, В. Б. Программирование микроконтроллеров для начинающих. Визуальное проектирование, язык C, ассемблер (+ CD-ROM) / В.Б. Иванов. - М.: Корона-Век, МК-Пресс, 2010. - 176 c.  
6. Калашников, О. А. Ассемблер — это просто. Учимся программировать (+ CD-ROM) / О.А. Калашников. - М.: БХВ-Петербург, 2011. - 336 c.  
7. Калашников, О. Ассемблер - это просто. Учимся программировать / О. Калашников. - М.: БХВ-Петербург, 2011. - 336 c.