

3 семестр. [ЭУМК] Архитектура компьютеров

Вы зашли под именем [Никита Иванов](#) ([Выход](#))

[В начало](#) ► [ПИ 3. АК](#) ► Модуль 1. «Базовые принципы архитектуры» ► [Лабораторная работа №2. Структура программы на языке...](#)

Настройки

[Управление курсом](#)

[Настройки моего профиля](#)

Навигация

[В начало](#)

■ [Моя домашняя страница](#)

[Страницы сайта](#)

[Мой профиль](#)

[Текущий курс](#)

[ПИ 3. АК](#)


[Участники](#)

[ЭУМК «Архитектура компьютеров»](#)


[Модуль 1. «Базовые принципы архитектуры»](#)


 [On-line лекции](#)

 [1. Введение](#)


 [2. Архитектура 32-битных Intel-совместимых микропр...](#)


 [3. Синтаксис языка Ассемблера](#)


 [4. Система команд микропроцессора Intel 80x86](#)


 [20.10.2020 - Видеозапись лекции по системе команд ...](#)

 [5. Подпрограммы](#)

 [Лабораторная работа №1. Принципы фон Неймана](#)

 [Лабораторная работа №2. Структура программы на языке...](#)

 [Лабораторная работа №3. Арифметические операции в ...](#)

 [Лабораторная работа №4. Ветвления и циклы в языке ...](#)

Лабораторная работа №2. Структура программы на языке Ассемблера

Лабораторная работа №2

Структура программы на языке Ассемблера

Задание 1. Ознакомиться с примером программы на языке Ассемблера, представленной ниже. Набрать в редакторе [ASM Editor](#) эту программу, скомпилировать и запустить программу, убедиться в её работоспособности:

```
.486
.model flat, stdcall
option casemap :none ; чувствительность к регистру букв в идентификаторе
include windows.inc
include kernel32.inc
includelib kernel32.lib

.data
    messageString db "Hello, World!!!"
    inputBuffer db 0

.data?
    inputHandle dd ?
    outputHandle dd ?
    numberOfChars dd ?

.code
entryPoint:

    push STD_INPUT_HANDLE ; передача параметра в функцию
    call GetStdHandle      ; вызов системной функции
    mov inputHandle, EAX   ; сохранение результата функции

    push STD_OUTPUT_HANDLE
    call GetStdHandle
    mov outputHandle, EAX

    push NULL              ; 5-ый параметр функции, системная ко
    push offset numberOfChars ; 4-ый параметр функции, адрес пере
    push 15                ; 3-ий параметр функции, целочисленна
    push offset messageString ; 2-ой параметр функций, адрес массив
    push outputHandle       ; 1-ый параметр функции, дескриптор с
    call WriteConsole

    push NULL
    push offset numberOfChars
    push 1
    push offset inputBuffer
    push inputHandle
    call ReadConsole

    push 0
    call ExitProcess

end entryPoint
```



Задание 2. Ознакомиться с приведённым ниже описанием системных функций, используемых в примере выше, и с общими правилами вызова системных функций в языке Ассемблера. На основе полученных знаний модифицировать приведённый выше пример таким образом, чтобы программа считывала введенную пользователем строку и выводила её в фигурных скобках.

Краткая теория:

В приведённом примере используется три вида инструкций языка Ассемблера, это команды push, call и mov. Команда push используется для работы с сегментом стека приложения, подробнее действие данной команды будет рассмотрено позже, в данном приложении эта команда используется для предварительной передачи параметров в функцию **перед** вызовом самой функции, который осуществляется командой call. При этом параметры функции, описанные в прототипе этой функции, записанном на языке C++, при вызове на языке Ассемблера передаются в **обратном** порядке. То есть первым с помощью команды push передаётся последний параметр функции (последний в её записи на языке C++). Команда mov в данном примере присваивает переменной, которая является первым операндом команды, значение регистра EAX, который является вторым операндом этой команды. Делается это для получения значения, возвращаемого системной функцией, так как все системные функции операционной системы Windows возвращаемое значение помещают именно в этот регистр.

Рассмотрим функции, используемые в приведённом примере:

Функция GetStdHandle

Возвращает дескриптор стандартного потока (ввода, вывода или ошибок) консоли. Дескриптор - это 32-битное целое беззнаковое число, по которому операционная система находит системные объекты, к которым относятся и потоки ввода/вывода консоли.

Прототип данной функции на языке C++ выглядит так:

```
HANDLE WINAPI GetStdHandle(DWORD nStdHandle);
```

Параметры функции:

nStdHandle

Входной обязательный параметр, который определяет, дескриптор какого именно потока необходимо получить. Может принимать значение одной из целочисленных констант:

STD_INPUT_HANDLE = -10	Поток ввода, по умолчанию ввод с клавиатуры в консоли.
STD_OUTPUT_HANDLE = -11	Поток вывода, по умолчанию вывод на экран в консоль.
STD_ERROR_HANDLE = -12	Поток ошибок, по умолчанию вывод на экран в консоль.

Возвращаемое значение:

В случае успеха возвращается дескриптор запрошенного потока, который затем может использоваться для ввода или вывода данных, используя функции ReadConsole или WriteConsole. В случае ошибки функция возвращает специальное значение, равное константе INVALID_HANDLE_VALUE. В случае, если ошибки не произошло, но приложение не может иметь доступ к консоли, возвращается нулевое значение, равное константе NULL.

Функция WriteConsole

Записывает символы строкового буфера в указанный поток вывода.

Прототип данной функции на языке C++ выглядит так:

```
BOOL WINAPI WriteConsole(  
    HANDLE hConsoleOutput,  
    const VOID *lpBuffer,  
    DWORD nNumberOfCharsToWrite,  
    LPDWORD lpNumberOfCharsWritten,  
    LPVOID lpReserved  
);
```

Параметры функции:

hConsoleOutput

Входной обязательный параметр — дескриптор потока вывода, в который будут выведены символы.

LpBuffer

Входной обязательный параметр — адрес начала строкового буфера, содержимое которого будет выведено в поток.

nNumberOfCharsToWrite

Входной обязательный параметр — количество символов в буфере, которое необходимо вывести в поток.

LpNumberOfCharsWritten

Выходной обязательный параметр — адрес 32-битной ячейки памяти, в которую функция запишет количество фактически выведенных в поток символов.

LpReserved

Зарезервированный параметр. Должен быть равен NULL.

Возвращаемое значение:

В случае успеха возвращается значение true, иначе false.

Функция ReadConsole

Считывает символы из указанного потока ввода в строковый буфер. При этом считанные символы удаляются из потока ввода.

Прототип данной функции на языке C++ выглядит так:

```
BOOL WINAPI ReadConsole(  
    HANDLE hConsoleInput,  
    LPVOID lpBuffer,  
    DWORD nNumberOfCharsToRead,  
    LPDWORD lpNumberOfCharsRead,  
    LPVOID pInputControl  
);
```

Параметры функции:

hConsoleInput

Входной обязательный параметр — дескриптор потока ввода, из которого будут прочитаны символы.

LpBuffer

Входной обязательный параметр — адрес начала строкового буфера, в который будут введены символы, прочитанные из потока.

nNumberOfCharsToRead

Входной обязательный параметр — количество символов, которое необходимо прочитать из потока. Данное количество должно быть не

больше размера самого буфера. При этом если пользователь ввёл с клавиатуры большее количество символов, чем указанное данным параметром, то введённая строка будет урезана до `nNumberOfCharsToRead` символов, остальные символы в буфер помещены не будут.

LpNumberOfCharsRead

Выходной обязательный параметр — адрес 32-битной ячейки памяти, в которую функция запишет количество фактически прочитанных из потока символов.

pInputControl

Входной необязательный параметр — адрес структуры, в которой можно указать дополнительные параметры потока, такие как символ окончания операции чтения. Данный параметр указывается для чтения символов в кодировке Unicode. В нашем случае можно всегда оставлять его равным NULL.

Возвращаемое значение:

В случае успеха возвращается значение `true`, иначе `false`.

Функция ExitProcess

Завершает приложение, в котором она вызывается.

Прототип данной функции на языке C++ выглядит так:

```
VOID WINAPI ExitProcess(UINT uExitCode);
```

Параметры функции:

uExitCode

Код завершения приложения, возвращаемый операционной системе.

Возвращаемое значение:

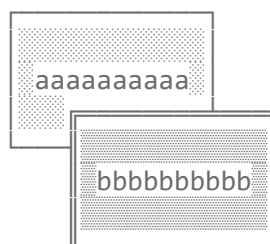
Функция не возвращает никакого значения.

Примечание:

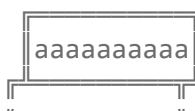
В рассмотренных функциях `WriteConsole` и `ReadConsole` используется явное указание размера буфера, то есть не используются строки с завершающим нулевым символом. Это нужно учитывать при операциях чтения строк из консоли. Если дальнейшая обработка прочитанной строки будет требовать завершающего нуля, то после операции чтения из потока в ячейку памяти с адресом `lpBuffer+[lpNumberOfCharsRead]` необходимо будет явно занести значение 0. Здесь квадратные скобки обозначают обращение к содержимому ячейки памяти с адресом `lpNumberOfCharsRead`. Записанное здесь выражение в таком виде на языке Ассемблера записано быть не может, правильная запись будет ясна после рассмотрения методов адресации.

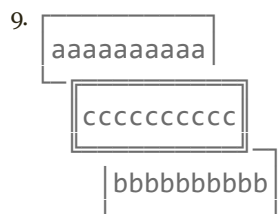
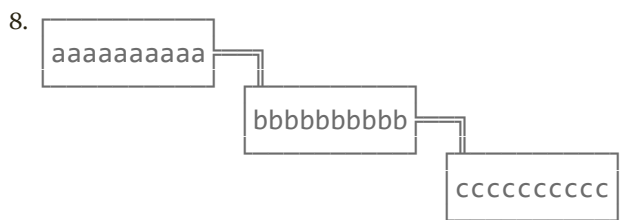
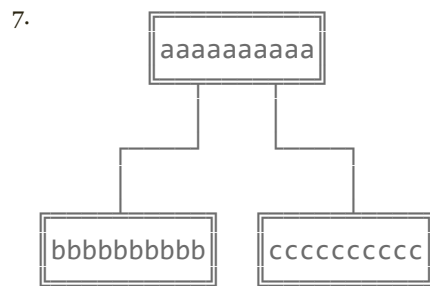
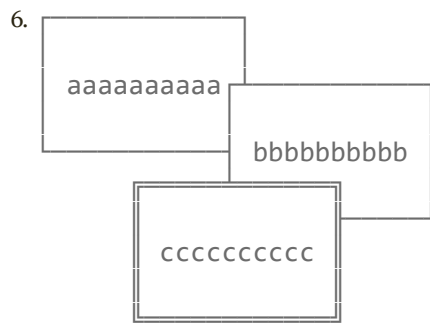
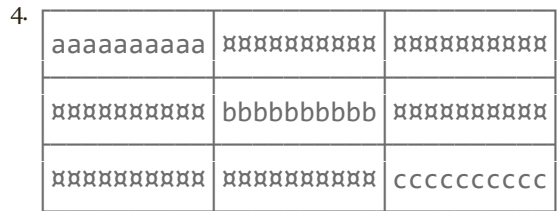
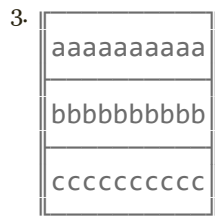
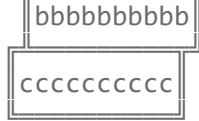
Задание 3. На языке Ассемблера создать консольное приложение, которое считывает три введённых пользователем строки, усекает или дополняет пробелами каждую строку до 10 символов, после чего выводит текст в соответствии с заданием Вашего варианта:

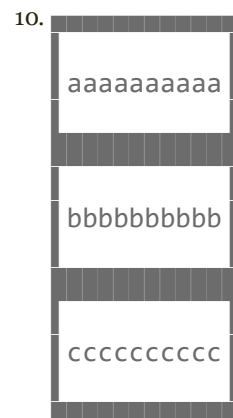
1. CCCCCCCCCC



2.







Во всех вариантах первая введенная пользователем строка обозначена как "aaaaaaaaaa", вторая — как "bbbbbbbbbb", и третья — как "cccccccccc".

Коды символов, с помощью которых необходимо строить предложенные схемы, можно найти в следующей таблице:

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_		⓪	⓫	♥	♦	♣	♠					♂	♀		♪	♠
1_	▶	◀	↕	!!	Ⓜ	§	—	±	↑	↓	→		⌒	↔	▲	▼
2_		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3_	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4_	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5_	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6_	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7_	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
8_	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
9_	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
A_	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
B_	␣	␣	␣		┤	├	┐	┌	└	┘	┙	┚	┛	├	┝	┞
C_	␣	␣	␣	┤	├	┐	┌	└	┘	┙	┚	┛	├	┝	┞	┟
D_	␣	␣	␣	┤	├	┐	┌	└	┘	┙	┚	┛	├	┝	┞	┟
E_	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
F_	Ё	ё	Є	€	İ	ï	ÿ	ÿ	°	·	·	√	№	¤	■	

В данной таблице крайний левый столбец содержит старшие цифры шестнадцатеричного кода символа, верхняя строка содержит младшие цифры шестнадцатеричного кода символа, а на пересечении соответствующих строки и столбца находится сам символ.

Замечание: рассмотрим операцию чтения строки из консоли

```
push NULL
push offset numberOfChars
push 1000
push offset inputBuffer
push inputHandle
call ReadConsole
```

где переменные описаны следующим образом

```
inputBuffer dd 1000 dup ( " ")
inputHandle dd ?
numberOfChars dd ?
```

При таком чтении если с клавиатуры будет прочитано более 10 символов, то при использовании вывода с помощью функции WriteConsole можно будет ограничить количество выводимых символов третьим параметром. Однако при вводе трёх символов, допустим, строки "abc" пользователь нажимает клавишу Enter для завершения ввода, и в строку на 4-ую и 5-ую позицию будут записаны коды символов, соответствующие этой клавише (символ с кодом 13 и символ с кодом 10). И хоть при объявлении строки inputBuffer мы резервируем 1000 пробельных символов (с кодом 32), при выводе первых 10 символов такой строки будут выведены 3 символа, введенных пользователем ("abc"), затем символ перехода на новую строку (код 13), затем символ возврата каретки (код 10), после чего оставшиеся 5 пробельных символов (код 32). Для того, чтобы избежать вывода символов с кодами 13 и 10, нужно на их место записать пробельные символы (с кодом 32). Добиться этого можно с помощью переменной numberOfChars. Для этого можно занести в регистр EBX адрес введенной строки inputBuffer, в регистр EAX занести количество прочитанных символов (фактически, длину строки inputBuffer). А затем в ячейки памяти с адресами EBX + EAX - 1 и EBX + EAX - 2 занести код пробельного символа

```
mov EBX, offset inputBuffer
mov EAX, numberOfChars
mov byte ptr [ EBX + EAX - 1 ], " "
mov byte ptr [ EBX + EAX - 2 ], " "
```



Состояние ответа

Состояние ответа на задание	Ответ на задание должен быть представлен вне сайта
Состояние оценивания	Не оценено

Вы зашли под именем [Никита Иванов](#) ([Выход](#))