МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное учреждение
высшего образования

«КУРГАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра автоматизации производственных процессов

# МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к комплексу лабораторных работ по дисциплине
«Вычислительные машины, системы и сети»
для студентов специальностей 15.03.04 «Автоматизация технологических процессов и производств» и 27.03.04 «Управление в технических системах»

Курган 2016

Кафедра автоматизации производственных процессов

Дисциплина: «Вычислительные машины, системы и сети»

Составил: ст. преподаватель  И.В. Скобелев

Утверждены на заседании кафедры «7»  апреля  2016 г.

Рекомендованы методическим советом университета
«17»  декабря  2015 г.

## СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Современный специалист в области автоматизации производства должен обладать достаточными знаниями по использованию средств вычислительной техники в организации и управлении технологическими процессами. Низкоуровневое программирование позволяет четко усвоить принципы работы вычислительных машин и систем, а также их функциональных блоков, более рационально использовать их вычислительную мощность на конкретном производстве, с учетом его особенностей.

**Целью** проведения лабораторных работ является изучение студентами организации и принципов функционирования памяти, микропроцессора, организации ввода – вывода, а также приобретение навыков низкоуровневого программирования на языке ассемблера.

Для выполнения лабораторных работ потребуется:

**TASM** или другой транслятор с языка ассемблера;

**TLINK** или другой компоновщик для построения исполняемого модуля;

**TD –** Турбоотладчик.

Для изучения использовании ассемблерных вставок может понадобиться Microsoft Visual Studio 6.0 или Microsoft Visual Studio 2010.

## Лабораторная работа №1
## Изучение команд языка Ассемблера и применение ассемблерных вставок в программе на C++

продолжительность 4 часа.

**Цель работы: Изучить использование некоторых команд языка Ассемблера. Научиться применять системный отладчик Debug. Научиться вставлять ассемблерный код в программу на языке C++.**

Выполнение работы

1. Вызовите в командной строке системный отладчик debug. Задайте команду _a и введите несколько ассемблерных команд, изменяющих содержимое регистров (Например с помощью команды MOV загрузить регистры ax и bx числами, выполнить сложение этих чисел).
2. Пошагово выполните загруженный фрагмент ассемблерной программы, следя за содержимым регистров с помощью команды _t. Посмотрите, какие еще есть возможности у системного отладчика.
3. Создайте проект на базе диалогового окна, обычным образом, как и во всех предыдущих лабораторных работах. В окне предусмотрите следующие поля редактирования: два поля для ввода значений, два поля

вывода значений из регистров, например, вх и евх, еще два поля для вывода значений массивов и одна кнопка для выполнения действий.

4. Для кнопки создать метод, в котором будет использована ассемблерная вставка. К полям редактирования создайте переменные.
5. Напишите программный код для исследования ассемблерных команд и доступа к переменным и массивам программы на языке VC++. Можно использовать пример, рассмотренный на лекции.
6. Изучите с помощью созданной программы способы получения данных из VC++ в ассемблерной вставке и передачи результатов вычислений в программу на VC++, способы согласования длины переменных и способы доступа к элементам массивов.
7. **Напишите отчет о проделанной работе. В отчете алгоритм решения задачи изобразите в виде блок-схемы.**

**Пояснения к выполнению работы**

Главная проблема при написании ассемблерных вставок - правильное понимание типов данных и их длины.

**Ассемблерный код для исследования в Debug**

```
mov bx,45
mov bl,2
  mov bh,4
add  bx,17
  mov ax,56
```

**Ассемблерная вставка**

```
void CAsm1Dlg::OnButton1()
{  short  a1,a2,a3,a5;
short ma[10];
int r1;
char s2,is[10]="A2b4C6L89";
for (int i=0; i<10; i++)
ma[i]=i*i;
UpdateData(1);
a1=m_a1;
a2=m_a2;

  __asm
  {   lea ecx,is
     lea edx,ma
      mov ebx,0
      mov bl,byte ptr a1
  mov bh,byte ptr a2
  mov  a3,bx
```

```
    mov r1,ebx
    add cx,a1
    mov bl, [ecx]
    mov s2,bl
    mov ax,a1
    add ax,a1
    add dx,ax
    mov bx,[edx]
    mov a5,bx
    }
     m_reg32=r1;
     m_rs=a3        ;
     m_s=s2;
     m_a5=a5;
     UpdateData(0);
  }
```

Контрольные вопросы

1. Для чего нужны ассемблерные вставки?
2. Для чего используется язык ассемблера и чем он отличается от алгоритмического языка?
3. Как в ассемблерной вставке получить доступ к переменным языка С?
4. Как в ассемблере можно организовать доступ к элементам массива?
5. Почему в ассемблерной вставке использована конструкция add ax,a1 add dx,ax?
6. Почему в ассемблерной вставке использована конструкция mov bl,byte ptr a1?

## Лабораторная работа №2
## Изучение программирования на языке Ассемблера

Продолжительность 8 часа.

**Цель работы: Научится разрабатывать и реализовывать простейшие программы на языке Ассемблера. Получить практические навыки работы по использованию транслятора tasm и редактора связей tlink. Научится использовать команды языка Ассемблера.**

Выполнение работы

1. Написать простейшую программу типа: Hello Word.
2. Оттранслировать и выполнить .

3. Усовершенствовать полученную во втором пункте программу, вывод двух строчек, например строчку “Hello Word” и с новой строки строчку «Good Bye!»
4. Рассмотреть программы выводящие различные мелодии, с помощью программирования системного динамика.
5. Изменить звучание программы.
6. **Написать отчет о проделанной работе.**

**Пояснения к выполнению работы:**

Пример программы типа Hello Word.

```
AStack   SEGMENT STACK
DW 32 DUP(?) ;Отвести под стек 32 слова
AStack   ENDS
AData    SEGMENT
Hello    DB 'Good Bye! $';Строка сообщения,
распечатывается
AData    ENDS
EData    SEGMENT
Bye      DB 'До свидания $';Строка сообщения, существует,
но
EData    ENDS            ;не распечатывается
ACode    SEGMENT
ASSUME CS:ACode, DS:AData, SS:Astack, ES: EData
WriteMsg     PROC NEAR    ;Функция 9 прерывания 21
         mov AH, 9    ;распечатывает сообщение, начальный
         int 21h  ;адрес которого DS:[DX]
         ret      ;Возврат в главную процедуру
WriteMsg     ENDP
Main     PROC FAR ;Главная процедура
         push DS      ;Сохранение адреса (адрес
         sub AX, AX   ;первой команды PSP) возврата
         push AX  ;в DOS
         mov AX, AData
         mov DS, AX   ;Загрузить в DS адрес сегмента
данных
         mov DX, OFFSET Hello ;Получить в DX смещение на
строку
                  ;сообщения
         call WriteMsg    ;Вызов процедуры распечатки
сообщения
         ret      ;Возврат в DOS
Main     ENDP
ACode    ENDS
END      Main
```

**Еще один пример**

```
.MODEL SMALL
        .STACK 100h
        .DATA
    Message  DB 'Hello!',13,10,'$'
        .CODE
        mov    ax,@Data
        mov    dx,ax         ; установить регистр DS таким
                             ; образом, чтобы он указывал
                             ; на сегмент данных
        mov    ah,9              ; функция DOS вывода строки
        mov    dx,OFFSET Message    ; ссылка на сообщение "Привет!"
        int    21h               ; вывести "Привет!" на экран
        mov    ah,4ch            ; функция DOS завершения
                                 ; программы
        int    21h               ; завершить программу
        END
```

**Пример программы, использующей таймер**

```
. masm
model small
.stack 100h
.data
note    dw
4832,6833,5423,6087,7240,8127,8127,4062,4559,4559,2415,1,4062,4559,5423,6
833,4559,4832,4832,9669,9669,4823,5423,5423,6087
dlit dw 4,4,4,4,8,4,4,4,6,2,4,4,8,4,4,6,2,4,4,8,4,4,4,4,4,4,8,4,4,6,2,4,4
.code                           ; Открываем кодовый сегмент
Begin:

    mov ax, @data
    mov ds, ax
    sub ax,ax
    mov si,0

    Mov al, 10110110B       ; Записываем управляющее слово в регистр
управления таймера
    out 43h,al              ; которому соответствует порт 43h
    in al,61h               ; Разрешаем динамику воспроизводить звуки и
записываем в  ключ 1
    or al,3             ; Для этого в порт 61h в первые два младших бита
записываем 11
    out 61h,al

play:
    mov ax, note[si]        ; Косвенная адресация
    out 42h, al             ; Заносим коэффициент деления в порт 42h
```

```
    mov al,ah               ;
    out 42h,al              ;
    mov ax,dlit[si]
    add ax,ax
    mov di,ax               ; Заносим в регистр SI величину задержки в тиках
таймера
    mov ah, 0               ; Считываем содержимое счётчика таймера
    int 1ah                 ; Прерывание необходимое для работы с таймером
    mov bx, dx              ; Переносим младший байт счётчика в регистр Bx
    add bx, di              ; Добавляем величину задержки в тиках таймера к
значению счётчика
next:                       ; Цикл
    int 1ah                 ; Прерывание, необходимое для того, чтобы
прочитать содержимое счётчика
    cmp dx, bx              ; Сравниваем содержимое регистра bx со
значением счётчика таймера
    jne next
    add si,2                ; Добавляем к индексному регистру слово
    cmp si,50               ; Условия выхода
    je exit             ;
    jmp play

exit:
    in al, 61h              ; Запрещаем динамику воспроизводить звуки
    and al, 11111100B       ;
    out 61h, al             ;
    mov ax,4c00h            ; Стандартный
    int 21h                 ; выход
end begin                       ; Конец программы
```

Контрольные вопросы

1. Для чего нужна программа **tasm?**
2. Для чего нужна программа **tlink?**
3. В каком редакторе можно набирать исходный текст на Ассемблере?
4. Какой тип файла получается после трансляции?
5. Как получить исполняемый файл?
6. Для чего нужен сегмент стека?
7. Для чего используются сегменты данных?
8. В каком сегменте записывается программа на ассемблере?

## Лабораторная работа №3
## Изучение арифметических операций над байтами и словами.

Продолжительность 8 часа.

**Цель работы: приобретение навыков использования арифметических команд при написании ассемблерных программ и вставок, получение представления об особенностях обработки данных разных размерностей и режимах доступа к данным при выполнении арифметических операций.**

Выполнение работы

1. Написать простейшую программу, либо, на языке ассемблера либо используя ассемблерную вставку, используя знания полученные в 1 и 2 лабораторной работе.
2. Выполнить сложение 10 и 27; полученный результат записать в соответствующую ячейку памяти.
3. Выполнить вычитание 10 и 27; полученный результат переслать в соответствующую ячейку памяти. Изменить знак второго числа (27) и снова выполнить операцию вычитания 10 и -27. Выполнить умножение 10 и -27 с учетом знака; результат записать в соответствующую ячейку памяти. Выполнить умножение 10 и -27 без учета знака.
4. Выполнить деление 325 и 15; полученные результаты записать в соответствующие ячейки памяти.
5. **Написать отчет о проделанной работе.**

**Пояснения к выполнению работы**

Пример программы сложения:

```
s_s segment stack "stack"  ;начало сегмента стека
 dw 12 dup(?)                    ; зарезервировано в памяти 24 ячейки
s_s ends                          ;конец сегмента стека
d_s segment              ;начало сегмента данных
 aa dw 5435h,4531h; данные, т.е. числа 5435h, 4531h записаны
                                ;по адресу aa и aa+2 соответственно, т.к. они
                                ;определены как слова
 s1 dw 2h           ; по адресу s1 записано число 2
 sum dw ?           ;любое данное записано по адресу sum (это метка)
d_s ends         ;конец сегмента данных
c_s segment              ;начало сегмента кода
 assume ss:s_s,ds:d_s,cs:c_s;
begin:                   ;начало программы
 mov ax,d_s
 mov ds,ax
 mov ax,aa         ;пересылка в регистр ax содержимого, находящегося
                          ;по адресу aa, т.е. числа 5435h
 add ax,aa+2     ;сложить число, которое находится в регистре ax с
                         ;содержимым, находящимся по адресу aa+2, т.е.числа
                         ;4531h, результат записывается в ax
 jno kof             ;перейти, если нет переполнения (OF=0)
```

```
mov ax,aa        ;если OF=1 - переполнение, выбрать опять число
add ax,s1        ;и сложить его с другим
kof: mov sum,ax ;переслать содержимое ax, т.е. результат в ячейку
                 ;памяти по адресу sum
mov ah,4ch       ;для правильного завершения программы необходимо
                 ;переслать в регистр ah 4ch
int 21h          ;и вызвать прерывание равное 21h
c_s ends         ;конец сегмента кода
end begin        ;конец программы
```

Контрольные вопросы

1. Как зарезервировать ячейки для хранения данных?
2. Команда для записи данных в ячейку.
3. Для чего используются регистры?
4. Назовите регистры процессора.
5. Напишите команду для записи данных в регистр.
6. Напишите команду сложения чисел в регистрах .
7. Напишите команду сложения чисел в регистре и ячейке памяти.

## Лабораторная работа №4
## Изучение логических операций и операции сдвига над данными

Продолжительность 8 часа.

**Цель работы: приобретение навыков использования логических команд и команд сдвига при написании ассемблерных программ и вставок, получение представления об особенностях обработки данных и режимах доступа к данным при выполнении логических и сдвиговых операций.**

Выполнение работы

1. Написать простейшую программу либо на языке ассемблера, либо, используя ассемблерную вставку, используя знания полученные в 1 и 2 лабораторной работе.
2. Определить однобайтовое число в двоичной системе счисления. Переписать его в регистр, установить 2 любых бита в единицу, инвертировать все, сбросить 3 любых бита.
3. Полученный результат продублировать в другом регистре, сложить получившиеся значения по модулю два.
4. Выполнить проверку, является ли полученный результат четным числом. Если да, то переписать его в регистр DH, иначе – в регистр DL.
5. Используя команды линейного сдвига, умножить сначала значение регистра DH или DL, в зависимости от результата предыдущей операции, на 4, а потом разделить на 2.

6. Используя команды циклического сдвига, в регистре BL получить значение третьего бита полученного числа, а в регистре BH – значение пятого бита.
7. **Написать отчет о проделанной работе**

Контрольные вопросы

1. Для чего нужны команды сдвига?
2. Какие команды сдвига Вы знаете?
3. Что такое циклический сдвиг?

## Лабораторная работа №5
## Изучение команд условного и безусловного переходов. Организация ветвлений и циклов в программе

Продолжительность 4 часа.

**Цель работы: приобретение навыков использования команд условного и безусловного переходов, циклов при написании ассемблерных программ и вставок, получение представления об особенностях обработки данных, команд и режимах доступа к данным при организации ветвлений и циклов.**

Выполнение работы:

1. Написать простейшую программу либо на языке ассемблера, либо, используя ассемблерную вставку, используя знания полученные в 1 и 2 лабораторной работе.
2. Определить два числа в шестнадцатеричной системе счисления, размером в один байт каждое.
3. Зарезервировать однобайтовую ячейку для хранения наибольшего общего делителя (НОД) двух чисел с произвольным первоначальным значением.
4. Используя команды переходов и цикла, найти НОД двух чисел, описанных в сегменте данных.
5. Полученный результат поместить в соответствующую ячейку памяти.
6. Используя команды циклического сдвига, переходов и цикла подсчитать количество единиц в НОД.
7. Полученное значение поместить в регистр DL.
8. **Написать отчет о проделанной работе**

Контрольные вопросы

1. Для чего используются команды безусловного перехода?
2. Назовите команды безусловного перехода.
3. Для чего используются команды условного перехода?

4. Продемонстрируйте использование команд условного перехода.
5. Как организовать цикл в ассемблерной программе?

## Лабораторная работа №6
## Изучение использование стека и подпрограмм. Организация внутрисегментных и межсегментных переходов.

продолжительность 8 часа.

**Цель работы: приобретение навыков использования команд безусловного перехода для организации внутрисегментных и межсегментных переходов и стека при написании ассемблерных программ и вставок, получение представления об особенностях обработки данных, команд и режимах доступа к данным при организации переходов и использовании стека.**

Выполнение работы

1. Написать простейшую программу либо на языке ассемблера либо используя ассемблерную вставку, используя знания полученные в 1 и 2 лабораторной работе.
2. Описать сегмент стека, в котором зарезервировать 30 ячеек, занятых нулями.
3. Описать два сегмента данных: в первом определить массив из семи однобайтовых чисел; во втором – определить массив из семи однобайтовых элементов, первоначально занятых нулями, а также две однобайтовые ячейки для хранения минимального и максимального элементов массива соответственно.
4. Программы нахождения минимального, максимального элементов массива, а также реверсирования массива оформить в виде процедур ближнего вызова.
5. Реверсирование массива реализовать с использованием стека.
6. Результаты работы каждой процедуры поместить в другой сегмент данных в соответствующие ячейки. При этом переход в другой сегмент необходимо выполнить один раз в конце основной программы, после чего переписать в него данные из соответствующих регистров.
7. **Написать отчет о проделанной работе.**

### Порядок оформления отчета

Отчет оформляется по результатам выполнения лабораторной работы на стандартных листах бумаги формата А4 с помощью редактора Word. Оформление отчета является важной частью лабораторной работы. При работе над отчетом студент должен осмыслить ход выполнения работы и научиться документировать полученные результаты.

Отчет должен содержать блок-схемы алгоритмов, оформленные по ГОСТу, необходимые структуры данных и распечатки программных кодов, разработанные студентом.

В отчете могут быть приведены тестовые данные, использовавшиеся студентом для проверки правильности работы программы. По результатам проверки программы должны быть сделаны выводы. Если в работе получились не те результаты, которые ожидались, в отчете должны быть отмечены причины, по которым не удалось получить желаемые результаты.

Допускается оформление одного отчета по всему курсу, выполненных в течение семестра лабораторных работ.

Пример оформления титульного листа для лабораторной работы приведен в Приложении.

**Список рекомендуемой литературы**


1. Таненбаум Э. Архитектура компьютера: 5-е изд. СПб.: Питер, 2012.
2. Орлов С. А., Цилькер Б. Я. Организация ЭВМ и систем: учебник для вузов. – 2-е изд. – СПб.: Питер, 2011.
3. Голенкова Ж.К., Заболоцкий А.В., Мархасин М.Л. и др. Руководство по архитектуре IBM PC AT/ Под общ. ред. М.Л.Мархасина. - Мн.: ООО «Консул», 1992.-949с.: ил.
4. Кулаков В. Программирование на аппаратном уровне: Специальный справочник. – СПб.: Питер, 2001.- 496 с.: ил.
5. Левкин Г.Н., Левкина В.Е. Введение в схемотехнику ПЭВМ IBM PC/AT. - М.: Изд-во МПИ, 1991..
6. Дженнингс. Ф. Практическая передача данных. Модемы, сети и протоколы. -М.: Мир, 1989.
7. Григорьев В.Л. Видеосистемы ПК фирмы IBM. - М.: Радио и связь, 1993.- 192 с.: ил.
8. Новиков Ю.В., Калашников О.А., Гуляев С.Э. Разработка устройств сопряжения для персонального компьютера типа IBM PC: Практическое пособие. -М.: ЭКОМ, 2000. – 224 с.
9. Юров В. Assembler. – СПб.: Питер, 2001.- 624 с.: ил.

**Приложение**

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

КУРГАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра автоматизации производственных процессов

**ОТЧЕТ**

по лабораторным работам
по дисциплине
«Вычислительные машины сети и системы»

Выполнил:
студент гр. 3014 Иванов И.И.

Проверил:
преподаватель Скобелев И.В.

Курган 2016

Скобелев Игорь Вадимович

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ**

к комплексу лабораторных работ по дисциплине

«Вычислительные машины, системы и сети»

для студентов специальностей 15.03.04 «Автоматизация технологических процессов и производств» и 27.03.04 «Управление в технических системах»

Авторская редакция

Подписано к печати | Формат 60x84 1/16 | Бумага 65 г/м$^2$
Печать цифровая | Усл.печ.л. 1,0 | Уч.-изд.л. 1,0
Заказ | Тираж 25 | Не для продажи

Редакционно-издательский центр КГУ.

640000, г. Курган, ул. Советская, 63/4.

Курганский государственный университет.