ЛАБОРАТОРНАЯ РАБОТА №7
ПЛАН

По дисциплине: Разработка ПО для встроенных систем
Тема занятия: **Простой вывод данных на экран**
Цель занятия: Научиться выводить данные на экран с помощью функции 09h прерывания 21h
Количество часов: 2

Содержание работы:

Составить программу вывода любого десятичного числа (размером до 2-х байт) на экран. При выводе числа на экран необходимо учесть, что символы на экране отображаются в ASCII-кодах. Число задать в коде программы.

Функция вывода на экран: ah=09h прерывания int 21h.

Дополнительная информация.

Переменная - это единица программных данных, имеющая символическое имя.

Большинство ассемблерных программ начинается с определения данных, которыми они будут оперировать. Распределение ячеек памяти и присвоение им идентификаторов осуществляется с помощью директив

db (Define Byte) - определить байт
dw (Define Word) - определить слово
dd (Define Doubleword) - определить двойное слово
dq (Define Quadword) - определить 4 слова
dt (Define Tenbyte) - определить 10 байтов.

Операторы распределения данных имеют следующий формат:

Имя	**db**	**нач.значение, {нач.значение}, ...**
Имя	**dw**	**нач.значение, {нач.значение}, ...**
Имя	**dd**	**нач.значение, {нач.значение}, ...**
Имя	**dq**	**нач.значение, {нач.значение}, ...**
Имя	**dt**	**нач.значение, {нач.значение},** ...

Например, оператор

alpha dw 0Ah - резервирует слово памяти, присваивает ему идентификатор alpha и заносит в него код 000A;

string db 'Привет' - резервирует 6 байт памяти и заносит в них строку символов и присваивает этой строке идентификатор string.

Другие операторы:

- **.** (точка) - ссылка на элемент структуры;
- **:** (двоеточие) - переопределение сегмента;
- **[]** (угловые скобки) - косвенная адресация;
- **?** - неинициализированное значение;
- **число DUP (значение)** - повторяющееся значение.

Например, оператор

Addr DD 20 DUP (?) - резервирует место для 20 полных адресов и присваивает этому массиву идентификатор Addr.

слева направо. Например, A+B-C - это (A+B)-C.

Команда пересылки: ***MOV dst, src -*** т.е. первым указывается операнд-получатель, а вторым – операнд источник. Одним из операндов обязательно должен быть регистр. { ***dst = src}***

Команды сложения: ***ADD dst, src -*** т.е. первый операнд складывается со вторым, и результат операции замещает первый операнд. { ***dst += src}***

Команда вычитания: ***SUB dst, src -*** т.е. второй операнд вычитается из первого и результат операции замещает первый операнд. { ***dst -= src}***

Команда сравнения ***CMP op1,op2 -*** команда сравнения выполняет вычитание второго операнда из первого, но нигде не запоминает результат операции и влияет только на состояние флажков.

Команда умножения: ***MUL src -*** команда умножения беззнаковых целых чисел MUL выполняет умножение адресуемого операнда на содержимое аккумулятора. При операции над байтами функции аккумулятора выполняет регистр ***AL***, а 16-битный результат операции помещается в регистр ***AX***. При операции над словами функции аккумулятора выполняет регистр ***AX***, а произведение длиной 32 бита формируется в регистрах ***DX*** (старшее слово) и ***AX*** (младшее слово).

Команда деления: ***DIV src -*** команда деления беззнаковых чисел DIV производит деление содержимого аккумулятора и его расширения на содержимое адресуемого операнда.

При делении 16-битного делимого на 8-битный делитель делимое помещают в регистр ***AX***. В результате выполнения операции частное формируется в регистре ***AL***, а остаток - в ***AH***. При делении 32-битного делимого на 16-битный делитель старшая часть делимого помещается в регистр ***DX***, а младшая - в ***AX***. В результате выполнения операции частное формируется в регистре ***AX***, а остаток - в ***DX***.

При делении на 0 автоматически происходит прерывание и переход к специальной программе обработки.

Команда безусловных переходов: ***JMP метка -*** Команда дальнего безусловного перехода реализует прямой и косвенный межсегментнные переходы.

Команды условных переходов: ***Jxx метка***

1) Команды для работы с беззнаковыми числами:

JA/JNBE - переход, если больше;
JAE/JNB/JNC - переход, если больше или равно;
JB/JNAE/JC - переход, если меньше;
JBE/JNA - переход, если меньше или равно.

2) Команды для работы со знаковыми числами:

JG/JNLE - переход, если больше;
JGE/JNL - переход, если больше или равно;
JL/JNGE - переход, если меньше;
JLE/JNG - переход, если меньше или равно;
JNS - переход, если больше нуля;
JS - переход, если меньше нуля.

3) Команды, общие для знаковых и беззнаковых чисел:

JE/JZ - переход, если равно / переход, если ноль;
JNE/JNZ - переход, если не равно / переход, если не ноль;
JNO - переход, если нет переполнения;
JO - переход, по переполнению.

4) Прочие команды:

JCXZ - переход, если содержимое регистра CX равно нулю;

JNP/JPO - переход при отсутствии четности;
JP/JPE - переход по четности.

Команда прерывания: ***INT type*** - вызов прерывания с номером type (от 0 до 255). Команда программного прерывания INT вызывает программу обработки, определяемую типом прерывания, помещенным в регистр ***AH.***

```
    .model small
    .286
    .stack 100h
    .data
a   dw   123
b   db   '    ',10,13,'$'
c   db   10
    .code
start:
    mov ax,@data
    mov ds,ax

    mov si,2
    mov ax, a
k1: div c
    add ah,30h
    mov [b+si],ah
    dec si
    mov ah,0
    cmp al,0
    jne k1
    mov dx, offset b
    mov ah,9h
    int 21h
    mov ah,4ch
    int 21h
    end start
```