Лабораторная работа № 5
Программирование ветвлений

Цель работы: Практическое овладение навыками разработки программного кода на языке Ассемблер. Изучение команд условного и безусловного перехода. Исследование организации переходов.

Задачи: Разработка простой программы использующей операторы передачи управления и приемов программирования арифметических выражений, содержащих разветвления.

Порядок выполнения работы

1. Создать рабочую папку для текстов программ на ассемблере и записать в нее файлы tasm.exe, tlink.exe, rtm.exe и td.exe. из пакета tasm, а также файл с исходным текстом программы на ассемблере, который сохранить с именем prog5.asm.
2. Составить программу арифметических и логических действий над целыми переменными и константами.
3. **Сформировать меню, предлагающее пользователю сделать выбор значения диапазона. Для ввода символа используется функция DOS 01h (при вводе символ отображается на экране).**
4. Создать загрузочный модуль, загрузить его в отладчик и выполнить программу в пошаговом режиме.

Содержание отчета:

1. Цель работы.
2. Постановка задачи.
3. Листинг программы.
4. Пояснения к программе (окно DOS, содержащие сформированное меню) .
5. Вывод.

Задание

1. В центре чистого экрана: сформировать меню отражающее суть задания и предлагающее пользователь ввести символ 1,2 или 3 для расчета соответствующего выражения при заданных значениях переменных a и b.

2. Вывести по углам экрана:
   - Фамилию автора программы;
   - Номер группы;
   - Факультет;
   - Символ ‘!’, повторив его 5 раз.
3. Задать видеоатрибуты, определяющие цвет символов и цвет фона при выводе символов на экран. Выбор цветового оформления зависит от вкуса.

Теоретическая часть

В лабораторной работе №1 приведена программа вывода строки на экран. Вывод строки в этой программе осуществляется посредством обращения к функции операционной системы (ОС). Кроме этой функции ОС содержит большое количество других функций, которые можно использовать при разработке программ.

Для вывода строки на экран использовалась функция **№9** прерывания **№21h.**

Прерывание - специальный набор готовых процедур, постоянно доступный для использования.

Общий алгоритм применения функций любого прерывания

1. Поместить номер вызываемой функции в регистр ah;
2. Подготовить, если необходимо, входные данные для функции;
3. Вызвать прерывание;
4. Сохранить или обработать выходные данные

Пример 1: Вывести на экран символ '$'

```
Mov ah, 02  ;номер функции
mov dl, '$' ;входные данные
int 21h     ;вызов прерывания
```

Функции BIOS для работы с экраном

В работе № 2 были рассмотрены различные системные функции DOS вывода на экран символьной информации. Однако возможности DOS весьма ограничены: она не имеет функций для изменения цвета выводимых символов и позиционирования курсора. Кроме того, в DOS отсутствуют средства формирования графических изображений.

Все возможности видеосистемы компьютера можно реализовать с помощью видеофункций BIOS прерывания **int 10h.** Прерывание **int 10h** обеспечивает: смену видеорежима (текстовый или графический); вывод символьной и текстовой информации; смену шрифтов, настройку цветовой палитры, работу с графическим изображением. Программирование видеосистемы с помощью средств BIOS более громоздко, однако большие возможности и высокая скорость вывода обуславливают широкое использование этого метода в прикладных программах.

Прерывание BIOS 10H (обмен данными с дисплеем).

Это прерывание обеспечивает выполнение 16-и операций с дисплеем. Выбор операции производится в зависимости от значения регистра AH. Операции с дисплеем можно разделить на 5 групп:

1. Операции интерфейса

- AH = 00h установка видеорежима
- AH = 01h установка конфигурации курсора
- AH = 02h установка позиции курсора
- AH = 03h получение положения курсора
- AH = 04h чтение положения светового пера
- AH = 05h выбор активной страницы
- AH = 06h прокрутка активной страницы (экрана) вверх
- AH = 07h прокрутка активной страницы (экрана) вниз

2. Операции обработки символов.

- AH = 08h чтение символа и его атрибута в текущей позиции курсора
- AH = 09h запись символа и атрибута в текущую позицию курсора
- AH = 0Ah запись символа в текущую позицию курсора

3. Операции графического интерфейса.

- AH = 0BH задание палитры цветов
- AH = 0CH изображение точки
- AH = 0DH чтение точки (позиция, атрибут)

4. Операция вывода в режиме телетайпа - позволяет использовать монитор, как простой терминал (вывод на экран с перемещением курсора)

AH = 0EH

5. Операция чтения в видеорежиме

AH = 15 0FH получение видеорежима

Примечание. Смотри файл **Справочный материал по функциям BIOS**

Пример 2: Позиционирование курсора.

```
mov ah,2   ; запрос на установку курсора
mov dh,8   ; номер строки - 8
mov dl,12  ; номер столбца - 12
mov bh,0   ; номер страницы
int 10H
```

Пример 3: Запись символа в позицию курсора

Записывает символ ASCII кода в текущую позицию курсора. Символ принимает атрибут, установленный ранее для этой позиции. После вывода курсор следует сместить к следующей позиции функцией 02h.

```
 mov AL, 'a'          ;выводимый символ (или его порядковый номер в таблице.)
mov bl, 10001100b     ;атрибут – ярко-красный мигающий
mov CX, 1             ;коэффициент повторения
int 10h
```

Пример 4: Очистка экрана.

Для очистки экрана используется прокрутка, т.к. при этой операции появляющиеся на экране строки заполняются пробелами.

```
mov ax,0600H    |mov ah,6 ; прокрутка вверх
                |mov al,0 ; строки, появляющиеся внизу, заполняются
                |пробелами. 0 - все окно заполняется пробелами
mov bh, 3fh     ;байт атрибута
mov cx,0000     |mov ch,0 ;номер строки верхнего левого угла окна
                |mov cl,0 ; номер столбца верхнего левого угла окна

mov dx,184FH    |mov dh,18H ;номер строки нижнего правого угла окна
                |mov dl,4FH;номер столбца нижнего правого угла |окна
int 10H
```

Режим цветного текста

Из рис. 1 следует, что каждый символ может принимать любой из 16 возможных цветов, определяемых сочетанием младших 4-х битов. Биты 4-6 байта атрибутов задают цвет фона под данным символом. Последний бит 7, в зависимости от режима видеоадаптера, определяет либо яркость фона под данным символом (тогда фон также может принимать 16 разных цветов), либо мерцание символа (устанавливается DOS по умолчанию).

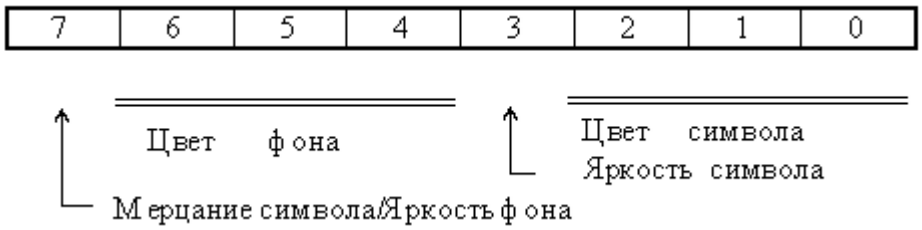

Рис.1 Структура байта атрибутов

При загрузке машины устанавливается стандартная палитра, коды цветов которой приведены в табл. 1. Рассмотрим некоторые примеры. Так, в режиме мерцания значение старшего полубайта атрибута 8h обозначает не серый фон, а чёрный при мерцающем символе, цвет которого по-прежнему определяется младшим полубайтом; значение старшего полубайта 0Ch – красный фон при мерцающем символе. Переключение назначения бита 7 осуществляется подфункцией 03h функции 10h прерывания int 10h.

Таблица 1

Коды цветов стандартной палитры

Двоичное число	Шестнадца-теричное число	Цвет	Двоичное число	Шестнадца-теричное число	Цвет
0000	0h	Чёрный	1000	8h	Серый
0001	1h	Синий	1001	9h	Голубой
0010	2h	Зелёный	1010	0Ah	Салатовый
0011	3h	Бирюзовый	1011	0Bh	Светло-бирюзовый
0100	4h	Красный	1100	0Ch	Розовый
0101	5h	Фиолетовый	1101	0Dh	Светло-фиолетовый
0110	6h	Коричневый	1110	0Eh	Жёлтый
0111	7h	Белый	1111	0Fh	Ярко- белый

Пример программы формирование меню

Простая программа, которая выводит меню и предлагает пользователю сделать выбор. Для ввода символа используется функция DOS 01h (при вводе символ отображается на экране). В зависимости от введённого символа осуществляется переход на нужный участок кода.

```
1
2
3  ;-------------------------------------------------------
4  menu     db '1 - Print hello',13,10
5           db '2 - Print go away',13,10
6           db '0 - Exit',13,10,'$'
7  select   db 13,10,'Select>$'
8  hello    db 13,10,'Hello!',13,10,13,10,'$'
9  go_away  db 13,10,'Go away!',13,10,13,10,'$'
10 ;-------------------------------------------------------
11     mov ah,09h          ;\
12     mov dx,menu         ; > Вывод меню
13     int 21h             ;/
14
15 select_loop:
16     mov ah,09h          ;\
17     mov dx,select       ; > Вывод строки 'Select>'
18     int 21h             ;/
19
20     mov ah,01h          ;Функция DOS 01h - ввод символа
21     int 21h             ;Введённый символ помещается в AL
22
23     cmp al,'1'          ;Сравнение введённого символа с '1'
24     je c1               ;Переход, если равно
25     cmp al,'2'          ;Сравнение введённого символа с '2'
26     je c2               ;Переход, если равно
27     cmp al,'0'          ;Сравнение введённого символа с '0'
28     je exit             ;Переход, если равно
29     jmp select_loop     ;Безусловный переход
30 c1:
31     mov ah,09h          ;\
32     mov dx,hello        ; > Вывод строки 'Hello'
33     int 21h             ;/
34     jmp start           ;Безусловный переход
35 c2:
36     mov ah,09h          ;\
37     mov dx,go_away      ; > Вывод строки 'Go away'
38     int 21h             ;/
39     jmp start           ;Безусловный переход
40 exit:
41     mov ax,4C00h        ;\
42     int 21h             ;/ Завершение программы
43
44
```