Рубежное тестирование 2
Ассемблер

1. **Общая структура программы под Windows (любой вариант).**

```
.386
.MODEL Flat, STDCALL
.DATA
  <Ваши инициализируемые данные>
  ......
.DATA?
  <Ваши не инициализируемые данные>
  ......
.CONST
  <Ваши константы>
  ......
.CODE
<метка>:
  <Ваш код>
  ......
end <метка>
```

Используемые библиотеки и файлы

Для подключения некоторых Windows-функций используются следующие файлы и библиотеки:

- *windows.inc*
- *user32.inc*
- *kernel32.inc*
- *user32.lib*
- *kernel32.lib*
- *comdlg32.lib*
- *и др.*

Ниже приведены примеры некоторых WinAPI функции:

1. int WINAPI **MessageBox**(
   _In_opt_ HWND hWnd,
   _In_opt_ LPCTSTR lpText,
   _In_opt_ LPCTSTR lpCaption,
   _In_ UINT uType
   ); - *показать диалоговое окно с сообщением*

2. HANDLE WINAPI **CreateFile**(
   _In_ LPCTSTR lpFileName,
   _In_ DWORD dwDesiredAccess,
   _In_ DWORD dwShareMode,
   _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes,

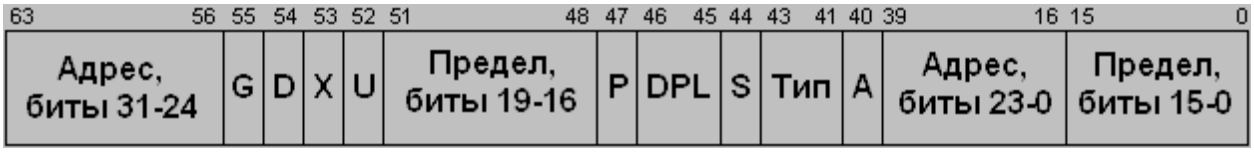
_In_ DWORD dwCreationDisposition,
_In_ DWORD dwFlagsAndAttributes,
_In_opt_ HANDLE hTemplateFile
); - *создает или открывает файл или устройство ввода/вывода.*

3. BOOL WINAPI **WriteFile**(
_In_ HANDLE hFile,
_In_ LPCVOID lpBuffer,
_In_ DWORD nNumberOfBytesToWrite,
_Out_opt_ LPDWORD lpNumberOfBytesWritten,
_Inout_opt_ LPOVERLAPPED lpOverlapped
); - *пишет данные в файл с места, обозначенного указателем позиции в файле*

4. BOOL WINAPI **CloseHandle**(
_In_ HANDLE hObject
); **-** *функция закрывает дескриптор открытого объекта*

*Пример вызова функции **MessageBox (“сообщение об ошибке”)**:*
invoke MessageBox, NULL, addr *MsgBoxTextEW*, addr *MsgBoxCaption*, 010h

2. Формат дескриптора сегмента



Базовый адрес – 32 битное поле базового адреса сегмента занимает 2, 3, 4 и 7 байты дескриптора, определяет любой начальный адрес сегмента в линейном адресном пространстве 4 Гбайт.

Предел – 20 битное поле предела (limit) или граница сегмента. Предел определяет размер сегмента в байтах минус 1.

Байт S дескриптора сегмента содержит информацию о правах доступа: **бит P** установлен в состояние 1, когда сегмент находится в оперативной памяти. При P=0 содержимое сегмента записывается на диск.

Двухбитное поле уровня привилегий дескриптора **DPL** определяет уровень привилегий сегмента и имеет значение от 0 (наивысший уровень привилегий) до 3 (наименьший уровень). Привилегии входят составной частью в механизм защиты процессора.

S – системный бит или бит сегмента в дескрипторах сегментов памяти всегда установлен в 1, а конкретное назначение сегмента описывается полем типа.

Тип – трехбитное поле определяет целевое использование сегмента, задавая допустимые операции в сегменте.

Поле типа определяет правила доступа к сегментам

Если бит S = 0, дескриптор описывает системный объект, как являющийся, так и не являющийся сегментом памяти. В этом случае биты 0 – 3 байта доступа определяют тип системных дескрипторов

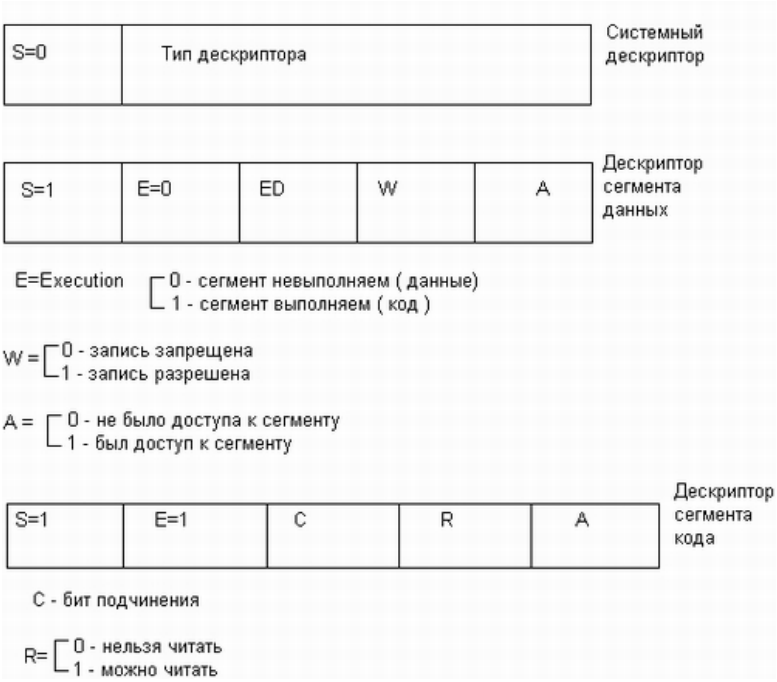
A – бит доступа или обращения. Процессор автоматически устанавливает его в 1, когда осуществляется обращение к тому сегменту памяти, который определяется данным дескриптором (ОС использует его при свопинге).

D – бит размера, если = 0 , 16 битные операнды, если – 1, то 32 битные операнды, в том числе и в стеке.

Бит 53 – зарезервирован.

Бит пользователя U предназначен для использования системными программистами по их усмотрению, процессор игнорирует это бит.

3. Как отличить сегмент кода от сегмента данных?



4. Системные регистры GDTR, IDTR, LDTR и TR. Их назначение и отличие.

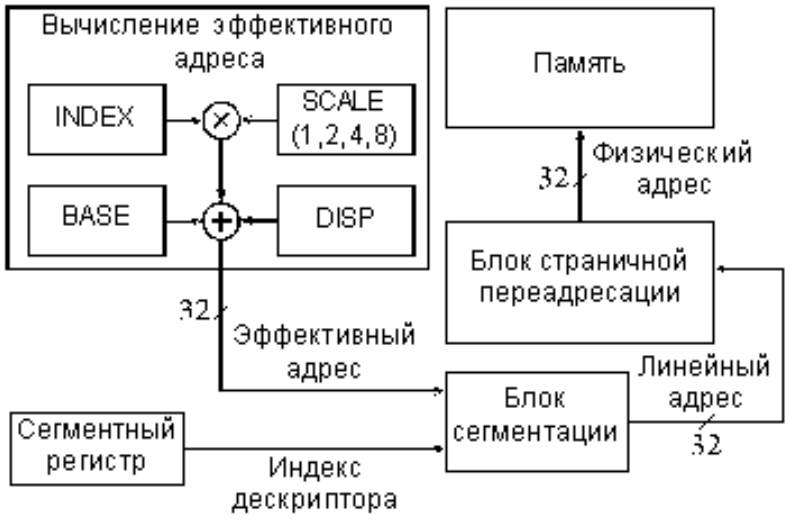
В архитектуре x86 есть три вида дескрипторных таблиц:

- Глобальная дескрипторная таблица (*Global Descriptor Table, GDT*);
- Локальная дескрипторная таблица (*Local Descriptor Table, LDT*);
- Таблица векторов прерываний (*Interrupt Descriptor Table, IDT*)**;**

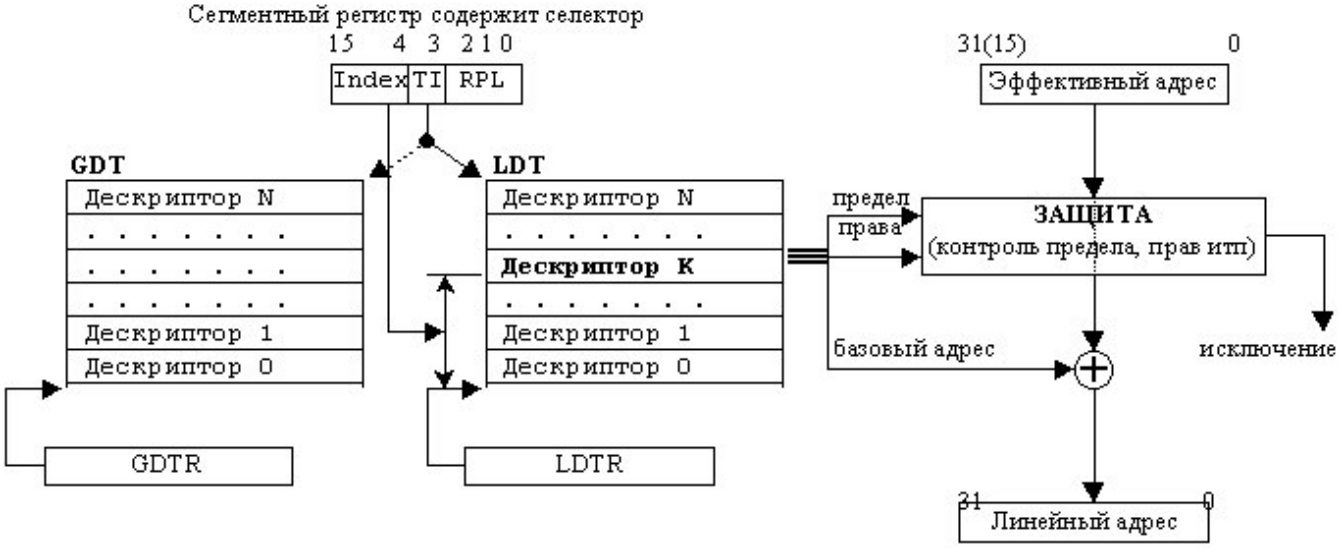
- ❑ Глобальная дескрипторная таблица является общей для всех процессов. Её размер и расположение в физической памяти определяются регистром **GDTR.**
- ❑ В отличие от **GDT**, **LDT** может быть много (соответственно количеству задач). Каждая задача может иметь свою. На расположение таблицы текущей задачи указывает регистр **LDTR.**

- Таблица прерываний глобальна. Размещение в физической памяти определяется регистром **IDTR.**
- **TR** (Task Register) - это 16-разрядный системный регистр, который хранит селектор дескриптора TSS текущей задачи.

5. Приведите схему формирования линейного адреса данных.



Вариант 2



TI (Table Indicator) – выбирает дескрипторную таблицу (TI=0 : GDT; TI=1 : LDT)
RPL (Requested Privilege Level) – запрашиваемый уровень привилегий

6. Защита по привилегиям. Уровни привилегий. Приведите пример привилегированных команд (2-3)

В защищённом режиме допускается одновременное выполнение нескольких прикладных (пользовательских) программ, но они изолированы друг от друга таким образом, что ошибки в одной из них не влияют на другие. В однопроцессорной системе в любой момент времени выполняется только одна программа, но возможность быстрого переключения с одной задачи на другую создаёт иллюзию одновременности.

Процессор поддерживает 4 уровня привилегий:

- **0** (наивысший) - ядро операционной системы;
- **1** - операционная система;
- **2** - системы программирования;
- **3** - прикладные (пользовательские) программы.

Примеры:

```
hlt              ; Останов процессора
clts             ; Сброс флажка переключённой задачи
lgdt, lidt. lldt ; Загрузка регистров дескрипторных таблиц
ltr              ; Загрузка регистра задачи
lmsw             ; Загрузка слова состояния машины
```

7. Кратко сформулируйте назначение подчинённых сегментов и шлюзов вызова.

Передача управлений между уровнями привилегий.Как было указано выше, передача управления командами CALLили JMPвозможна только в пределах одного уровня привилегий. Для преодоления данного ограничения (передачи управления между уровнями привилегий) применяются два способа:

- применение подчиненных сегментов кода;
- применение специальных дескрипторов - шлюзов (вентилей) вызова.

Подчиненные сегменты кода позволяют обращаться к себе на любом уровне привилегий, так как они подчиняются тому уровню привилегий, который передается им командами CALLили JMP. Поэтому здесь отсутствуют проблемы согласования уровней. Однако при использовании подчиненных сегментов передача управления разрешается только в сторону повышения привилегий (внутрь колец, CPL> DPL), что связано с проблемой правильной организацией возврата к исходной программе.

Шлюз вызова позволяет передавать управление с одного уровня привилегий на другой. Также шлюз вызова полезен при передаче управления между 16- и 32-разрядным кодом.

На рис. 4-1 приведен формат дескриптора шлюза вызова.

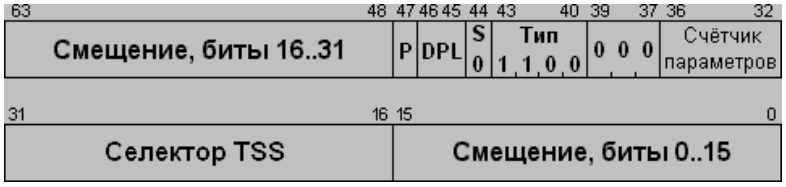


Рисунок 4-1. Формат дескриптора шлюза вызова

Этот дескриптор можно размещать в GDT и LDT, но не в IDT. Дескриптор шлюза вызова выполняет 6 функций:

- Определяет сегмент кода, к которому будет доступ.
- Определяет точку входа в этом сегменте.

- Определяет уровень привилегий, который должна иметь вызывающая процедура.
- При смене стека, он указывает число необязательных параметров, копируемых процессором из одного стека в другой.
- Определяет размер значений, записываемых в целевой стек: 16-разрядный шлюз использует 16-разрядные значения, 32-разрядный - 32-разрядные.
- Определяет присутствие шлюза вызова.