Лабораторная работа №7

Тема работы: Интеграция ассемблерных прерываний в проекты на C++.
Цель работы: Получить понимание принципов работы DOS-прерываний и методов их использования в программировании на языке C++.

Задания:

1. **Автоматическая блокировка экрана.**
   На стороне Assembler:
   Используйте прерывание таймера для определения периода неактивности пользователя.
   При достижении этого времени активируйте прерывание видеокарты для блокировки экрана.
   При блокировке экрана, позвольте проигрывать заданное пользователем аудио или анимацию.
   Прерывание клавиатуры используйте для разблокировки после ввода пароля.
   При каждой неудачной попытке ввода пароля, сохраняйте информацию (дата, время, возможно, последние нажатые клавиши) в отдельный лог-файл.
   На стороне C++:
   Интерфейс управления настройками блокировки экрана. Возможность выбора аудио или анимации для воспроизведения при блокировке. Вызов соответствующих функций ассемблера для обработки активности, блокировки экрана и ведения лог-файла.

   Используемые прерывания: int 10h, int 21h, int 1Ch, int 16h.

2. **Менеджер паролей.**
   На стороне Assembler:
   Реализация алгоритма поточного шифрования RC4, используя assembler-функции.
   Генератор случайных чисел: На базе системного таймера или других источников энтропии для создания случайных паролей.
   Пользователь может сохранять пароли и другую конфиденциальную информацию. Данные хранятся в зашифрованном виде. Реализуйте функцию генерации случайных паролей. Тайм-аут бездействия: Если менеджер паролей открыт и не используется в течение заданного времени, автоматически блокируйте его.
   На стороне C++:
   Главное меню и пользовательский интерфейс. Управление функциями ассемблера: добавление, удаление и редактирование записей; генерация пароля; шифрование и дешифрование данных. Логика тайм-аута бездействия.
   Используемые прерывания: int 16h, int 21h, int 1Ah.

3. **Утилита управления звуком.**
   Загрузите все необходимые драйверы и зависимости.
   На стороне Assembler:
   Определите, есть ли звуковая карта в системе. Если нет, выведите соответствующее сообщение и завершите программу.
   Отобразите пользователю основные параметры звука: громкость, баланс, другие параметры, которые может предоставлять ваша звуковая карта.
   Предложите пользователю несколько опций: изменить громкость, изменить баланс, выйти.
   Добавь простой эквалайзер и визуализацию.
   На стороне C++:
   Реализация меню с обработкой ошибок. Выводите параметры звука, полученные с помощью функций на ассемблере. Обрабатывайте команды пользователя и вызывайте соответствующие функции на ассемблере для управления звуком.
   Добавь простую визуализацию.

4. **Консольный файловый менеджер.**
   На стороне Assembler:
   Используйте прерывания видео для вывода на экран текущего каталога и содержимого директории. Отображайте только текущий путь и список файлов/папок в данной директории. Используйте прерывание клавиатуры для базовой навигации: вверх, вниз, выбор файла или папки. С помощью прерывания переходите в выбранный каталог или выходите на уровень выше. При выборе файла, отображайте его размер и дату последнего изменения. Эту информацию можно получить также через прерывание DOS (INT 21h).
   Добавить создание и открытие файлов.

   На стороне C++:
   Парсинг файлов в текущей директории.
   Логика управления.
   Вызов ассемблерных функций.
   Меню и интерфейс.
   Используемые прерывания: int 10h, int 16h, int 21h.

5. **Расширенный клавиатурный макро-менеджер.**
   На стороне Assembler:
   Необходимо создать макросы для комбинаций клавиш, выполняя ассемблерные функции в ответ на активацию макроса.
   Макросы:
   - F1 - вывод всех макросов и их описание.
   - При нажатии комбинации s + d вызывается ассемблерная функция, которая активирует PC Speaker для издания звукового сигнала определенной частоты и длительности.
   - Длительность звукового сигнала определяется по времени зажатия клавиш s + d(Например: клавиши были зажаты 30 секунд, после

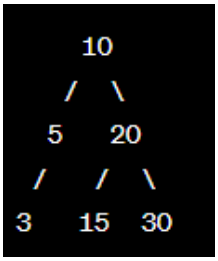
отпускания клавиш активируется звуковой сигнал протяженностью 30 секунд).

- При нажатии другой комбинации клавиш, например s + c, вызывается ассемблерная функция, которая изменяет цвет фона и/или текста в консоли.
- При нажатии комбинации, например s + r, текст на экране начинает "вращаться" (циклически сдвигаться влево или вправо).

На стороне C++:
В главной функции создайте цикл, который будет проверять состояние клавиатуры с использованием функции, написанной на ассемблере. При обнаружении комбинации клавиш вызывайте соответствующую функцию, написанную на ассемблере.
Логирование: Записывайте все действия пользователя и реакции программы на них в файл лога. Это может быть полезно для отладки или просто для истории использования.
Используемые прерывания: int 16h, int 10h, int 1Ah.

6. **Система уведомлений.**
   На стороне Assembler:
   Реализация функции мониторинга событий (например, проверка наличия устройства) с использованием DOS-прерываний. Вывод уведомления на экран при обнаружении события. Воспроизведение звукового сигнала с использованием PC Speaker. Обработка нажатий клавиш для выполнения действий из уведомления (например, отклонить подключение устройства).
   На стороне C++:
   Интерфейс управления и настройки мониторинга событий. Интерфейс для работы с уведомлениями, таким как управление ими и их настройки. Вызов соответствующих функций ассемблера для обработки событий и действий.
   Используемые прерывания: int 10h, int 21h, int 1Ch, int 11h, int 12h, int 16h.

7. **Бинарное дерево поиска с автоматическим бэкапом.**
   На стороне Assembler:
   Реализация алгоритмов для балансировки бинарного дерева поиска, чтобы обеспечить оптимальное время поиска.
   При каждой операции с деревом (добавление, удаление) автоматически создается резервная копия дерева в файле.
   Добавьте функционал восстановления дерева из последнего бэкапа.
   Реализацию всего дерева и его функций выполнить в assembler-коде.
   Сохранения дерева из файла надо произвести с использованием сериализации.
   На стороне C++:
   Управление функциями ассемблера: инициализация дерева, добавление и удаление элементов, балансировка дерева. Сериализация и десериализация дерева при сохранении и загрузке. Функции для создания бэкапов и восстановления из них.
   Загрузка дерева из файла надо произвести с использованием десериализации.



Используемые прерывания: int 16h, int 21h, int 1Ah.

8. **Создание простого графического редактора.**
   На стороне Assembler:
   Реализуйте функции для рисования линий, окружностей и прямоугольников на ассемблере.
   Добавьте возможность сохранения и загрузки изображений с использованием. Используйте int 21h для работы с файлами. Создайте собственный простой формат для сохранения изображений (например, сохранение каждого пикселя как структуры RGB). При сохранении изображения сначала сохраните его размеры, затем цвет каждого пикселя. При загрузке прочтите размеры изображения и затем восстановите изображение пиксель за пикселем.
   При нажатии клавиш “+” “-” изменять цвет.
   На стороне C++:
   Пользовательский интерфейс:
   - Меню
   - Выбор инструментов рисования (линии, окружности, прямоугольники)
   - Возможность загрузки и сохранения изображений.

   Управление вызовами функций Assembler:
   - Вызов функций рисования
   - Вызов функций сохранения и загрузки изображений
   - Обработка ошибок и исключений, связанных с работой с файлами или графикой.

   Используемые прерывания: int 10h, int 21h, int 16h.

9. **Текстовый редактор с шифрованием.**
   Реализуйте базовый текстовый редактор.
   На стороне Assembler:
   Добавьте функцию шифрования текста с помощью ассемблерного прерывания.
   Реализуйте функции сохранения и открытия зашифрованных файлов.
   Встроенное сообщение о нарушениях: Если кто-то пытается открыть зашифрованный файл без соответствующего ключа более определенного числа раз, программа автоматически уничтожает или блокирует файл. Предложите несколько различных методов шифрования, таких как Цезарь, подстановка, транспозиция и др(минимум 3 метода). Пользователь должен иметь возможность выбирать метод шифрования перед его применением. Методы должны быть реализованы как assembler-функции.
   На стороне C++:
   Меню и интерфейс пользователя. Управление функциями ассемблера: выбор метода шифрования, ввод ключа, чтение и запись файла и т.д. Логика по учету неудачных попыток доступа к файлу и последующего его уничтожения.

Используемые прерывания: int 16h, int 21h.

10. **Консольный менеджер процессов**
    На стороне Assembler:
    Показать все текущие выполняющиеся процессы с их идентификаторами, именами, статусами и потреблением ресурсов. Предоставлять возможность приостанавливать, возобновлять и завершать процессы. При выборе определенного процесса, пользователь должен увидеть детали о нем, такие как время запуска, путь к файлу, используемые ресурсы. Позволять фильтровать процессы по имени, ресурсам, а также сортировать их по различным критериям(1 любой критерий).
    На стороне C++:
    Логика управления.
    Вызов ассемблерных функций.
    Меню и интерфейс.
    Используемые прерывания: int 10h, int 16h, int 21h.

    Примечание:
    Важно понимать, что DOS не является многозадачной операционной системой в современном понимании, и управление процессами, как в Windows, невозможно. DOS позволяет работать только с одной задачей за раз. Для выполнения задачи в среде DOS, можно имитировать "процессы" как отдельные программы или загружаемые модули и создать программу на Turbo C++, которая бы имела возможность запускать эти "процессы", останавливать их и отображать информацию, основываясь на доступных данных.
    Лучше всего использовать windows 95, список процессов можно получить с помощью Win32API.

11. **Необходимо подсчитать корни биквадратного уравнения** вида ax^4+bx^2+c=0 (a, b, c - числа с плавающей точкой).
    На стороне Assembler:
    Ввод данных и посчитать дискриминант и корни. Обработайте возможные ошибки (например, отрицательный дискриминант, деление на ноль) с помощью прерываний. Также реализовать проверку на ввод. Передайте корни в стек для дальнейшего использования в C++. Промежуточные вычисления ошибки должны выводиться с помощью прерываний в консоль.
    На стороне C++:
    Извлеките корни из стека.
    Выведите корни в различных форматах (двоичный, десятичный, шестнадцатеричный).
    Используемые прерывания: int 04h, int 21h, int 0h.

12. **Графическая утилита для мониторинга системы.**
    На стороне Assembler:
    С помощью прерываний соберите информацию о системе: тип процессора, объем оперативной памяти, свободное пространство на диск. Выведите всю эту информацию.

На стороне C++:
Реализуйте "живые" диаграммы и графики, отображающие нагрузку на процессор, использование памяти и дисковой активности в реальном времени. Разработайте интерфейс, который будет регулярно обновляться на основе данных, полученных из функций ассемблера.
Обеспечьте отображение графиков и диаграмм, возможно, с использованием сторонних библиотек, на основе данных полученных из Assembler.
Используемые прерывания: int 15h, int 10h, int 21h.

13. **Тестирование скорости диска "пингом"**
Создайте инструмент, который будет "пинговать" ваш диск, записывая и затем считывая небольшие блоки данных для измерения задержек. Идея состоит в том, чтобы определить задержку (латентность) записи и чтения диска.
Запросите у пользователя размер блока данных для теста (например, 1 КБ, 10 КБ, 100 КБ). Запросите количество итераций (например, 100 пингов).
Запись на диск: Зафиксируйте текущее время с помощью прерывания таймера. Запишите блок данных указанного размера на диск. Зафиксируйте время после записи. Рассчитайте разницу между временами для определения времени записи.
Чтение с диска: Зафиксируйте текущее время. Прочтите блок данных с диска. Зафиксируйте время после чтения. Рассчитайте разницу между временами для определения времени чтения.
Рассчитайте среднее время записи и чтения за все итерации.
Используйте прерывания для обработки ошибок (например, недостаточно места на диске). Многопоточное тестирование: При запуске теста, создайте несколько потоков (например, 4), каждый из которых будет "пинговать" диск. Это позволит симулировать реальную среду, где диск может обрабатывать несколько операций одновременно(реализация на C++).

14. **Графическое представление функций.**
Разработайте систему, позволяющую пользователю вводить простейшие функции, и отображайте их графики с использованием прерываний видеокарты(на заданном отрезке).
Отображение графиков можно делать только в первой четверти.
На стороне Assembler:
Функции для переключения видеорежимов и отрисовки пикселей (используя INT 10h). Вычисление значения функции в заданной точке (в зависимости от введенной функции). Применение метода трапеций или Симпсона для численного интегрирования и вычисления площади под графиком.
На стороне C++:
Интерфейс для ввода функции пользователем. Вызов функций ассемблера для отрисовки графика функции. Вызов функций ассемблера для вычисления площади под графиком. Отображение результатов пользователю.
Используемые прерывания: int 10h, int 21h, int 16h.

15. **Интерактивный редактор изображений с обработчиком событий.**

Реализуйте простой интерактивный редактор изображений, который позволит пользователю выбирать и применять различные эффекты и трансформации к изображениям.
На стороне Assembler:
Для обработки пользовательского ввода и реакции на различные события используйте прерывания.
Поворот изображения(например, стрелки или клавиши "A" и "D" для поворота)
Используйте прерывание мыши: при двойном клике на определенную область изображения применяется выбранный фильтр или эффект.
Применяйте разные эффекты, такие как размытие, контраст, яркость, градация серого и другие(достаточно 3 эффектов).
Используйте прерывание таймера для создания анимированных эффектов, таких как постепенное изменение яркости или контраста.
На стороне C++:
Интерфейс редактора. Вызов ассемблерных функций для обработки изображений. Обработка пользовательского ввода (через прерывания) и вызов соответствующих функций редактирования. Загрузка и сохранение изображений.
Используемые прерывания: int 10h, int 33h, int 21h, int 1Ch.