# 1 Цель работы

Изучение основ разработки ОС, принципов низкоуровневого взаимодействия с аппаратным обеспечением, программирования системной функциональности и процесса загрузки системы.

# 2 Ход работы

Работа выполняется в **ОС Windows**  
Транслятор ассемблера для загрузчика: **YASM, синтаксис: Intel**  
Компилятор ядра: **Microsoft C complier**  
Задание для реализации: **CalcOS**

Загрузчик взаимодействует с пользователем с помощью клавиатуры. Используется прерывание 0x16, для считывания символа клавиатуры. После того, как символ –цифра, соответствующая определённому цвету- будет считан, происходит загрузка данных «data» в регистры и передача управления ядру ОС. Считанная цифра остаётся на экране до тех пор, пока загруженное ядро не считает информацию с экрана, не запишет нужное значение цвета и не сотрёт всю информацию с экрана.

Далее, выполняется основная часть кода ядра ОС, которая подразумевает собой взаимодействие с пользователем при помощи консоли. Ядро поддерживает такие команды, как “info”, “help”, “shutdown”, “expr \*”, где \* - математическое выражение.

Весь выводимый в окне текст будет представлен в цвете, указанном при загрузке ОС.

# 3 Реализация загрузчика

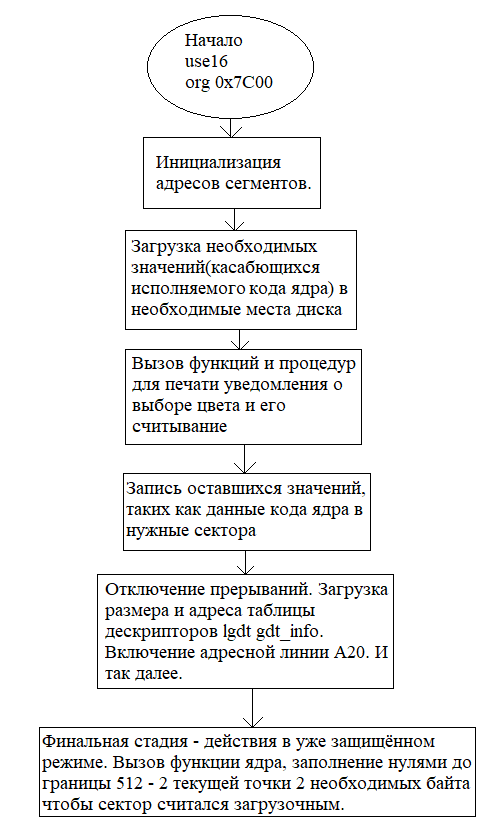
При запуске системы процессор Intel находится в реальном режиме, который нужно будет изменить на защищённый после передачи нужных аргументов в регистры и других дополнительных действий.

Первым из наиболее важных действий является указание адреса загрузки кода(в данном случае, команда org 0x7C00). Далее происходит запись необходимых значений в регистры, для того, чтобы выполнить такие важные мероприятия, как, например, сохранение адреса сегмента кода,

сохранение этого адреса как начало сегмента данных и так далее.

После этого производится ряд действий, переводящих процессор в защищённый режим. К таковым относятся: отключение прерываний, загрузка размера и адреса таблицы дескрипторов и так далее.

Процесс загрузки ядра заканчивается заполнением нулями сектора до границы 512 – 2 и записью в эти последние два байта значений, 0х55 и 0хАА.



# 4 реализаця ядра

В начале, считывается цифра, введённая пользователем на этапе загрузки системы, сохраняется нужный цвет последующего вывода, экран очищается и выводится окно приветствия.

Основной функцией данной операционной системы является вычисление математических выражений. Для этого используется команда “expr”. Вычисляется значение, написанное после данного выражения.

Талица функций, использующихся в ядре.

|  |  |
| --- | --- |
| void processing(char \*str) | Первая в очереди вызываемых для обработки выражения функция. Принимает на вход целую строку, начиная с выражения “expr”. Сначала производит «базовые» проверки (пустое выражение и т.п.), а после вызывает функции «углубленной» проверки. |
| char find\_error\_mult\_div(char \*str, int len) | Производит поиск неправильных знаковых записей. |
| char incorrect\_input(char \*str, int len) | Производит поиск неверных символов. НЕ цифр и НЕ знаков. |
| char\* delete\_extra(char \*str, int len) | Производит удаление лишних «+» и «-»(унарных). |
| int digit\_to\_integer(char \*str, int len) | Перевод «строкового» числа в знаечниие тпа int. |
| void integer\_to\_digit(int integer, char \*temp\_char) | Оперцаия, обратная предыдущей. |
| void slide(int \*mass, int result, int place, int size) | Сдвиг значений в массиве. |
| int calculate(char \*str, int len) | Собсвтенно, сама функция вычисления выражения, вызывающая другие функции. |

# 6 ПРИмер работы

