**Команды управления и циклы**

1. **Переходы**

**Переход** – это изменение порядка выполнения команд в программе. Существуют несколько видов переходов

**Ветвление** представляет собой структуру, которая позволяет программе выбирать один из нескольких путей выполнения на основе определенного условия. Для реализации ветвлений используются такие операторы, как if, switch и другие подобные конструкции.

**Безусловный** **переход** - это инструкция, которая передает выполнение программы к другому участку кода без проверки каких-либо условий.

**Условные переходы** осуществляются лишь тогда, когда выполняется заданное условие.

**Переходы по смещению** представляют собой такие переходы, при которых адрес перехода определяется исходя из текущего местоположения программы. Эти переходы могут быть как безусловными, так и условными.

1. **Циклы**

**Циклы** дают возможность повторять определенные операции несколько раз. Это особенно полезно для обработки последовательностей данных, а также для выполнения действий до тех пор, пока не будет выполнено определенное условие.

**Цикл с фиксированным количеством повторений в C++**

for (int i = 0; i < 5; ++i) {

std::cout << i << std::endl;

}

**Интерационный цикл в C++**

int i = 0;

while (i < 5) {

std::cout << i << std::endl;

i++;

}

**Цикл смешанного типа в C++**

int i = 0;

for (;;) {

if (i >= 5) break;

std::cout << i << std::endl;

i++;

}

**Мультипликативный цикл в C++**

int main() {

int product = 1;

for (int i = 1; i <= 5; i++) {

product \*= i;

std::cout << "Факториал " << i << ": " << product << std::endl;

}

return 0;

}

1. **Переадресация**

**Переадресация** — это способ изменения адреса следующей выполняемой команды.

**Переадресация с использованием констант и восстановление (на языке ассемблера)**

mov eax, 0x12345678 ; Загрузка константы в регистр EAX

jmp eax ; Переход по адресу, хранящемуся в EAX

**Косвенная адресация (на языке ассемблера)**

mov eax, [ebx] ; Косвенная адресация через регистр EBX

**Автоинкремент и автодекремент (на языке ассемблера)**

**Автоинкремент:**

inc eax ; Увеличить значение в регистре EAX на 1

**Автодекремент:**

dec ebx ; Уменьшить значение в регистре EBX на 1

**Стек (на языке ассемблера)**

push eax ; Поместить значение из EAX в стек

pop ebx ; Извлечь верхнее значение из стека в EBX

**Индексный регистр (на языке ассемблера)**

mov esi, offset array ; Установить ESI на начало массива

add esi, 4 ; Переместиться на 4 байта вперед (индекс 1)

mov eax, [esi] ; Загрузить значение элемента массива в EAX

1. **Продвинутые команды управления циклами**

**Продвинутые команды управления циклами позволяют гибче управлять выполнением циклических конструкций.**

**Комбинированные команды: управление и индексирование (на языке ассемблера)**

mov ecx, 5 ; Число итераций

mov edi, offset arr ; Указатель на массив

loop\_label:

inc dword ptr [edi] ; Увеличение элемента массива

add edi, 4 ; Переход к следующему элементу

loop loop\_label ; Продолжение цикла

**Управление с инкрементом индекса (на языке ассемблера)**

mov ecx, 5 ; Число итераций

mov edi, offset array ; Начальный адрес массива

mov eax, 0 ; Результат

loop\_label:

add eax, [edi] ; Добавить элемент массива к EAX

add edi, 4 ; Инкремент индекса на размер int (4 байта)

loop loop\_label ; Повторять, пока ECX не станет равен нулю

**Управление с использованием счетчика (на языке ассемблера)**

mov ecx, 5 ; Счётчик цикла

mov eax, 0 ; Аккумулятор

loop\_label:

add eax, ecx ; Добавить значение счётчика к аккумулятору

dec ecx ; Декремент счётчика

jnz loop\_label ; Повторить, если ECX не ноль

**Управление с индексированием и счетчиком (на языке ассемблера)**

mov ecx, 5 ; Счётчик цикла

mov edi, offset arr ; Индекс массива

loop\_label:

mov eax, [edi] ; Загрузить элемент массива

add edi, 4 ; Следующий элемент (инкремент индекса)

dec ecx ; Декремент счётчика

jnz loop\_label ; Повторить, если ECX не ноль