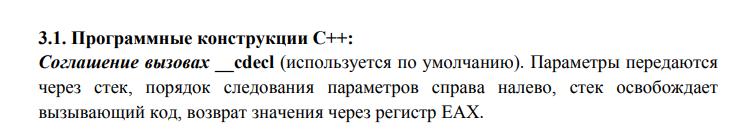
1. mov (move) - инструкция, используемая для перемещения данных между регистрами процессора и памятью. Она может быть использована для загрузки значений в регистры, сохранения значений из регистров в память и копирования значений между регистрами.
2. push - инструкция, используемая для помещения значения на вершину стека. Она может использоваться для передачи параметров функции и сохранения значений регистров перед вызовом другой функции.
3. sub (subtract) - инструкция, используемая для выделения места на стеке. Она вычитает значение из указателя стека rsp и создает новую область памяти на стеке для хранения данных.
4. lea (load effective address) - инструкция, используемая для загрузки адреса в регистр. Она не загружает значение из памяти в регистр, а загружает сам адрес.
5. call - инструкция, используемая для вызова функции. Она помещает адрес возврата на стек и переводит управление на указанный адрес, где выполняется вызываемая функция.

**Вызов функции:**

**\_\_stdcall** — это стандартный способ передачи аргументов, при котором аргументы передаются через стек в обратном порядке. Этот способ используется по умолчанию для функций Windows API.

**\_\_cdecl** — это альтернативный способ передачи аргументов, при котором аргументы также передаются через стек, но в прямом порядке. Этот способ часто используется для обычных функций в C/C++.



**\_fastcall** — это еще один способ передачи аргументов в функции на x86-совместимых процессорах. В отличие от **\_\_stdcall** и **\_\_cdecl**, **\_fastcall** использует регистры процессора для передачи первых аргументов функции (обычно до трех аргументов).

**Cdecl:**

**mov dword ptr [rsp+18h], r8d** - сохраняет значение регистра **r8d** по адресу **[rsp+18h]**.

00007FF7107B19B0 mov dword ptr [rsp+18h],r8d

**mov dword ptr [rsp+10h], edx** - сохраняет значение регистра **edx** по адресу **[rsp+10h]**.

00007FF7107B19B5 mov dword ptr [rsp+10h],edx

**mov dword ptr [rsp+8], ecx** - сохраняет значение регистра **ecx** по адресу **[rsp+8]**.

00007FF7107B19B9 mov dword ptr [rsp+8],ecx

**push rbp** сохраняет значение регистра **rbp** на вершине стека.

00007FF7107B19BD push rbp

**push rdi** сохраняет значение регистра **rdi** на вершине стека

00007FF7107B19BE push rdi

Данная инструкция ассемблера **sub rsp,108h** выделяет 108 байтов (0x108 в шестнадцатеричной системе) памяти на стеке, сдвигая указатель стека **rsp** на 0x108 байтов вниз.

00007FF7107B19BF sub rsp,108h

**lea rbp, [rsp+20h]** загружает в регистр **rbp** адрес, вычисленный как **rsp+20h**.

00007FF7107B19C6 lea rbp,[rsp+20h]

Данная инструкция ассемблера **lea rcx, [\_\_536B1A64\_Call@cpp(07FF7107C4068h)]** загружает в регистр **rcx** адрес символической ссылки **\_\_536B1A64\_Call@cpp**, которая используется для обращения к функции **sum\_c**.

00007FF7107B19CB lea rcx,[\_\_536B1A64\_Call@cpp (07FF7107C4068h)]

Данная инструкция ассемблера **call \_\_CheckForDebuggerJustMyCode(07FF7107B13F7h)** вызывает функцию **\_\_CheckForDebuggerJustMyCode**, которая используется для проверки, находится ли отладчик в режиме "только мой код" (Just My Code) для текущего процесса.

00007FF7107B19D2 call \_\_CheckForDebuggerJustMyCode (07FF7107B13F7h)

int sum = 0;

Данная инструкция ассемблера **mov dword ptr [sum], 0** является операцией присваивания значения **0** переменной **sum**.

00007FF7107B19D7 mov dword ptr [sum],0

sum = a + b + c;

Данная инструкция ассемблера **mov eax, dword ptr [b]** загружает значение в регистр **eax** из ячейки памяти, адрес которой хранится в переменной **b**.

00007FF7107B19DE mov eax,dword ptr [b]

Данная инструкция ассемблера **mov ecx, dword ptr [a]** загружает значение в регистр **ecx** из ячейки памяти, адрес которой хранится в переменной **a**.

00007FF7107B19E4 mov ecx,dword ptr [a]

Данная инструкция ассемблера **add ecx, eax** складывает значения, хранящиеся в регистрах **eax** и **ecx,** и сохраняет результат в регистре **ecx**.

00007FF7107B19EA add ecx,eax

Данная инструкция ассемблера **mov eax, ecx** копирует значение, хранящееся в регистре **ecx**, в регистр **eax**.

00007FF7107B19EC mov eax,ecx

Данная инструкция ассемблера **add eax, dword ptr [c]** складывает значение, хранящееся по адресу, хранящемуся в переменной **c**, с значением, хранящимся в регистре **eax**.

00007FF7107B19EE add eax,dword ptr [c]   
Эта инструкция записывает значение, хранящееся в регистре eax, в переменную sum.

00007FF7107B19F4 mov dword ptr [sum],eax

return sum;

Эта инструкция ассемблера загружает значение, хранящееся в переменной **sum**, в регистр **eax**. Значение в регистре **eax** затем возвращается как результат выполнения функции.

00007FF7107B19F7 mov eax,dword ptr [sum]

**\_STDCALL:**

int \_stdcall mul(int& a, int b, int c)

{

00007FF7107B1940 mov dword ptr [rsp+18h],r8d

00007FF7107B1945 mov dword ptr [rsp+10h],edx

00007FF7107B1949 mov qword ptr [rsp+8],rcx

00007FF7107B194E push rbp

00007FF7107B194F push rdi

00007FF7107B1950 sub rsp,108h

00007FF7107B1957 lea rbp,[rsp+20h]

00007FF7107B195C lea rcx,[\_\_536B1A64\_Call@cpp (07FF7107C4068h)]

00007FF7107B1963 call \_\_CheckForDebuggerJustMyCode (07FF7107B13F7h)

int mul\_result = 1;

00007FF7107B1968 mov dword ptr [mul\_result],1

mul\_result = a \* b \* c;

00007FF7107B196F mov rax,qword ptr [a]

00007FF7107B1976 mov eax,dword ptr [rax]

00007FF7107B1978 imul eax,dword ptr [b]

00007FF7107B197F imul eax,dword ptr [c]

00007FF7107B1986 mov dword ptr [mul\_result],eax

return mul\_result;

00007FF7107B1989 mov eax,dword ptr [mul\_result]

}

**\_Fastcall:**

int \_fastcall sum\_f(int a, int b, int c)

{

00007FF7107B1A20 mov dword ptr [rsp+18h],r8d

00007FF7107B1A25 mov dword ptr [rsp+10h],edx

00007FF7107B1A29 mov dword ptr [rsp+8],ecx

00007FF7107B1A2D push rbp

00007FF7107B1A2E push rdi

00007FF7107B1A2F sub rsp,108h

00007FF7107B1A36 lea rbp,[rsp+20h]

00007FF7107B1A3B lea rcx,[\_\_536B1A64\_Call@cpp (07FF7107C4068h)]

00007FF7107B1A42 call \_\_CheckForDebuggerJustMyCode (07FF7107B13F7h)

int sum = a + b + c;

00007FF7107B1A47 mov eax,dword ptr [b]

00007FF7107B1A4D mov ecx,dword ptr [a]

00007FF7107B1A53 add ecx,eax

00007FF7107B1A55 mov eax,ecx

00007FF7107B1A57 add eax,dword ptr [c]

00007FF7107B1A5D mov dword ptr [sum],eax

return sum;

00007FF7107B1A60 mov eax,dword ptr [sum]

}

00007FF7107B1A63 lea rsp,[rbp+0E8h]

00007FF7107B1A6A pop rdi

00007FF7107B1A6B pop rbp