Calling Conventions

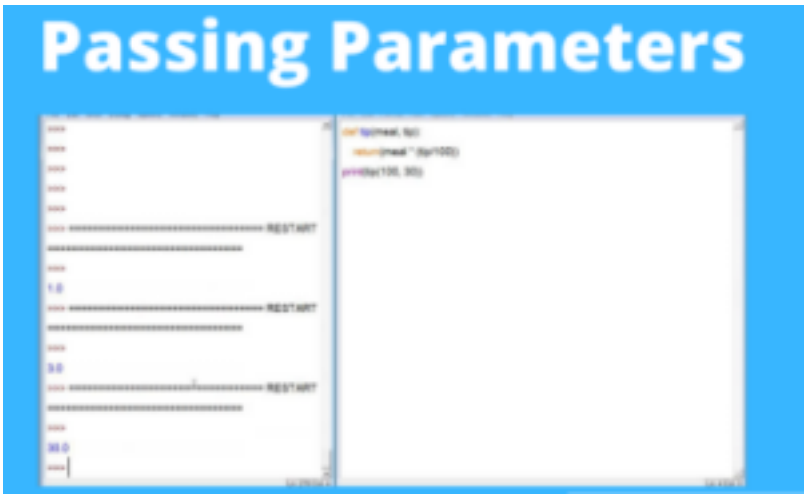
Here is a simplified explanation of the **C** and **STDCALL** calling conventions.

==========

C calling convention:

==========

Parameter passing: Parameters are pushed onto the stack in reverse order.



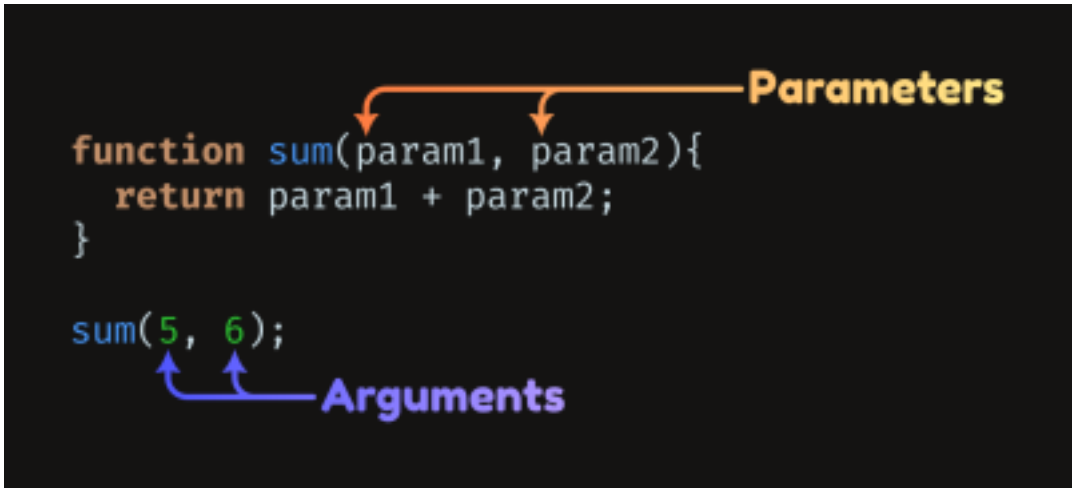
Stack cleanup: The caller is responsible for cleaning up the stack after the function returns.

==========

STDCALL calling convention:

==========

Parameter passing: Parameters are pushed onto the stack in reverse order.



Stack cleanup: The called function is responsible for cleaning up the stack after it returns.

In other words, with the C calling convention, the caller has to tell the called function how many parameters it is passing.

With the STDCALL calling convention, the called function knows how many parameters it is receiving because it is responsible for cleaning up the stack.

Here is an example of a function call using the C calling convention:

```
int AddTwo(int a, int b) {
  return a + b;
}

int main() {
  int a = 5;
  int b = 6;

  ;Push the parameters onto the stack in reverse order.
  push(b);
  push(a);

  ;Call the AddTwo function.
  call AddTwo;

  ;Add the size of the parameters to the stack pointer to clean up the stack.
  add esp, 8;

  ;Store the return value in a variable.
  int result = eax;

  ; ...
}
```

Here is an example of a function call using the STDCALL calling convention:

```
int AddTwo(int a, int b) {
  return a + b;
}

int main() {
  int a = 5;
  int b = 6;

  ;Push the parameters onto the stack in reverse order.
  push(b);
  push(a);

  ;Call the AddTwo function.
  call AddTwo;

  ;...
}
```

As you can see, the only difference between the two calling conventions is who is responsible for cleaning up the stack. With the C calling convention, the caller is responsible. With the STDCALL calling convention, the called function is responsible.

The STDCALL calling convention is used by the Windows API, so it is important to be familiar with it if you are writing programs that call Windows API functions.

==========

Useless stuff you don't need:

==========

There are many different calling conventions, each with its own advantages and disadvantages. Some common calling conventions include:

C calling convention: This calling convention is used by the C and C++ programming languages. It is simple to implement, but it can be

inefficient for functions with many parameters.

STDCALL calling convention: This calling convention is used by the Windows API. It is more efficient than the C calling convention for functions with many parameters, but it is more complex to implement. x64 calling convention: This calling convention is used by 64-bit x86 processors. It is similar to the STDCALL calling convention, but it has been optimized for 64-bit performance.

Pascal calling convention: This calling convention is used by the Pascal programming language. It is similar to the C calling convention, but it passes the first parameter in a register instead of on the stack.

FORTRAN calling convention: This calling convention is used by the FORTRAN programming language. It is different from the other calling conventions in that it passes all parameters in registers, rather than on the stack. In addition to these general-purpose calling conventions, there are also many specialized calling conventions that are used for specific purposes.

For example, there are calling conventions for operating system calls, library functions, and even specific types of functions, such as floating-point functions.

The number of calling conventions that exist depends on the specific processor architecture and programming language. However, there are a few common calling conventions that are used by most processors and

programming languages.