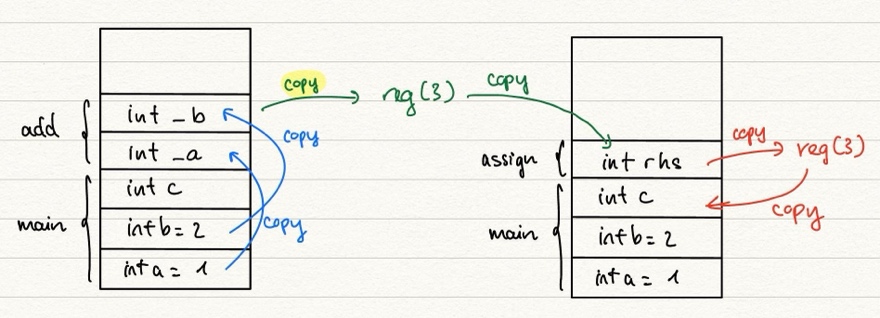
**Copy Constructor**

Say we want to implement the following functions:

Text

Description automatically generatedThen the following will happen on the stack: (Notice the copy actions)

+) The return value from a function (add or rhs) is **registered** (to somewhere in the CPU)

+) There is copy actions done while passing by value. We also do copy to register.

+) Then, when assigning and passsing values, copy is done.

Text

Description automatically generated**?** So what if copy is done on Objects instead of integers?

When we do c = a + b;

Diagram

Description automatically generated with medium confidence\*\*Since it's possible that classes exceeds the memory of register when returning, compilers normally **make a copy of the return object (maybe on stack), and let register store the address of that object.**

\*\* If we change the parameters of operator+ and operator= functions to be pass by reference, the memory for r is no longer needed. However, if we do c = a + b, we are passing by reference the temporary object. Thus, we MUST declare as const (not changing this temp object) in order to bind it with c.

|  |
| --- |
| **!!! NOTE: We can change the return type of operator= to be Complex&, since we would then avoid creation of "copy of C", and can reference directly to C.**  **However, we CANNOT do the same thing for operator+, since "copy of temp" is a copy of a temporary value that lives on stack, and will get deleted later 🡪 Dangerous** |

1. **Implementation**

So, the following scenarios calls the copy constructor:

+) Pass by value

+) Return by value

+) When instantiating one object from another of the same type

MyClass a(b);

+) **When we declare & initialize an object synchronously**

* !!!! NOT when Complex b; b = a; (this is operator=)

Complex a = b; (since this is considered similar to Complex a(b))

MyClass ( const MyClass& other);

**+) Must pass the object by reference!**

**+) Must use const for the parameter (explained above)**

**+) The default copy constructor is member-wise copy** 🡪 shallow copy, problems when have pointers!

1. **Use of Const for binding**

Text, letter

Description automatically generated

🡪 For operator+, **NO** use const. For operator=, use pass by reference and **use const**