seL4 IPC Emulation Internals

To implement the seL4 IPC emulation, we need to:

- Pass the **emulated register set values** + the message registers.
- Follow the architecture specific calling conventions and the seL4 semantics.

For example to emulate **`x64_sys_send_recv`**: on x86_64:

```
x64_sys_recv(seL4_Word sys, seL4_Word src, seL4_Word *out_badge,
        seL4_Word *out_info, seL4_Word *out_mr0, seL4_Word *out_mr1,
        seL4_Word *out_mr2, seL4_Word *out_mr3, seL4_Word reply)
{
        register seL4_Word mr0 asm("r10");
        register seL4_Word mr1 asm("r8");
        register seL4_Word mr2 asm("r9");
        register seL4_Word mr3 asm("r15");
        MCS_REPLY_DECL;
        asm volatile(
                "movq   %%rsp, %%rbx    \n"
                "syscall                \n"
                "movq   %%rbx, %%rsp    \n"
                : "=D"(*out_badge),
                "=S"(*out_info),
                "=r"(mr0),
                "=r"(mr1),
                "=r"(mr2),
                "=r"(mr3)
                : "d"(sys),
                "D"(src)
                MCS_REPLY
                : "%rcx", "%rbx", "r11", "memory"
        );
}
```

RDI	stores	**syscall number**
RSI	stores	**message info**
RDX	stores	**capability pointer**
R10	stores	**message reigister 0**
R8	stores	**message reigister 1**
R9	stores	**message reigister 2**
R15	stores	**message reigister 3**
R12	stores	**reply** (only used in MCS configuration)
IPC Buffer	stores	**Other message registers**