Make sure Fiasco is started with `-serial_esc` and Qemu with `-serial_stdio` (both are the default in this repository).

You can enter JDB by

- Pressing escape at any time during the execution
- Including this code:

```c
#include <l4/sys/kdebug.h>

// somewhere in your code
enter_kdebug("message");
```

For that your process needs the JDB capability (`jdb = L4.Env.jdb` in Lua).

It is normal for one CPU to run at 100% in JDB (it polls for input).
JDB – commands

- Most importantly: \texttt{h} – help
- \texttt{JS} – resize JDB to match terminal size
- \texttt{Q} – list kernel objects
  - Navigate with cursor keys
  - Select an object with enter for more information
  - For tasks & threads: \( S = \) address space, \( C = \) cpu, \( R = \) ref count
  - For IPC gates: \( L == \) label, \( D = \) owning thread
- \texttt{Esc} – Leave menus like the above
- \texttt{g} – Continue running.
JDB – commands (2)

- **lp/lr** – list all/ready threads
- In detailed thread view (after selecting a thread in Q, lp, lr):
  - Space – disassembly
- **dt<task-id><address>** – memory dump
  - Space switches modes (big endian, little endian, ASCII)
  - e allows to edit the memory
  - u gives disassembly
- **X** – play tetris
IPC logging

- JDB can log all IPCs, i.e. log system calls
- I* – turn on IPC log
- IR+ – turn on result log
- T – view trace buffer (after running your code)
- Output format:

  \[
  \text{ipc: } \text{THR_ID TYPE -> [C:CAP_DEST] DID=DEST_ID} \backslash
  L=\text{LABEL [TAG] (MSG1, MSG2) TO=TIMEOUT} \backslash
  \text{THR_ID answ [TAG] L=FROM err=ERR.no} \backslash
  (\text{ERR.str}) (MSG1,MSG2)
  \]

  Here MSG1 and MSG2 are the first two words of the message. The answ lines are threads receiving (not necessarily answers).
Debugging with GDB

- Launch Qemu with -s to start GDB stub
- Connect from GDB with `target remote localhost:1234`
- Consider passing -S to Qemu: With that it’ll only boot after you type `continue in gdb`
- Problems:
  - You will be stepping through kernel code without debugging information.
  - You can load debugging information for a binary as usual, but GDB won’t know which address space you are in.
  - You can’t switch binaries while running GDB.