```
    .i_en(),
    .i_reset(),

    // Operating mode (HIGH=single-step)
    .i_mode(),
    // Single-step pulse
    .i_step(),

    // UART comms
    .i_rx(),
    .q_tx(),
    .q_tx_byte(),

    // Debug outputs
    .q_debug_instr_clk(),
    .q_debug()
);
```

### 7.2.2 PRCO304 Compiler Reference Guide

#### Building the Compiler

To build the compiler (`cli` front-end and `libprco` back-end), run the following commands:

```
cd prco304
mkdir build && cd build
cmake ..
cmake --build .
```

If you wish to build the compiler's own standard library run the following command as root/administrator to install the sources and header files:

```
cmake --build . --target install
```

#### Command Line Interface (CLI) Arguments

**Name**

`cli` - compile a program into executable machine code for the PRCO304 processor.

**Synopsis**

`cli [OPTION]... -i{FILE}`

**Description**

`-d` Dump output machine code to a file

`-D{bits}` Select debug printing level. Example of use: `-D0xFF` to enable all debug bits.

`-i{file}` Pass the input file to the compiler. Example of use: `-i code.prco`.

`-O{0-1}` Enable optimisation levels. `0` = no optimisations, `>0` = constant folding and unreachable code elimination.

`-m{arch}` Pass the target architecture to the compiler. Deprecated.

### 7.2.3 PRCO304 Emulator Reference Guide

**Name**

`emu` - Disassemble and emulate PRCO304 processor programs.

**Synopsis**

```
emu [OPTION]... -i{FILE}
```

**Description**

`-i` Input machine code file. 1 instruction word per line. CRLF/LF accepted.

`-D{bits}` Select debug printing level. Example of use: `-D0xFF` to enable all debug bits.

## Example Output

```
Disassembly of Input:
Instruction,        MC,    Tag, Comment
----------------------------------------
ADDI  $-1,  Sp      4fff   0    (null)
  SW  Bp,   +0(Sp)  16e0   0    (null)
MOV   Bp,   Sp      1ee0   0    (null)
SUBI  $+1,  Sp      5f01   0    (null)
MOVI  $62,  Ax      2062   0    (null)
  SW  Ax,   -1(Bp)  10df   0    (null)
  LW  Ax,   -1(Bp)  08df   0    (null)
WRITE Ax,   UART1   9800   0    (null)
MOVI  $65,  Ax      2065   0    (null)
  SW  Ax,   -1(Bp)  10df   0    (null)
  LW  Ax,   -1(Bp)  08df   0    (null)
WRITE Ax,   UART1   9800   0    (null)
MOVI  $6e,  Ax      206e   0    (null)
  SW  Ax,   -1(Bp)  10df   0    (null)
  LW  Ax,   -1(Bp)  08df   0    (null)
WRITE Ax,   UART1   9800   0    (null)
MOVI  $20,  Ax      2020   0    (null)
  SW  Ax,   -1(Bp)  10df   0    (null)
  LW  Ax,   -1(Bp)  08df   0    (null)
WRITE Ax,   UART1   9800   0    (null)
MOV   Sp,   Bp      1fc0   0    (null)
  LW  Bp,   +0(Sp)  0ee0   0    (null)
ADDI  $+1,  Sp      4f01   0    (null)
HALT                9000   0    (null)

Initial Memory layout:
00    01    02    03    04    05    06    07    08
=====================================================================
4fff  16e0  1ee0  5f01  2062  10df  8df   9800  2065  10df  8df   9800  206e  10df  8df
9800  2020  10df  8df   9800  1fc0  ee0   4f01  9000  00    00    00    00    00    00
00    00    00    00    00    00    00    00    00    00    00    00    00    00    00
00    00    00    00    00    00    00    00    00    00    00    00    00    00    00
00    00    00    00    00    00    00    00    00    00    00    00    00    00    00
00    00    00    00    00    00    00    00    00    00    00    00    00    00    00
00    00    00    00    00    00    00    00    00    00    00    00    00    00    00
00    00    00    00    00    00    00    00    00    00    00    00    00    00    00
00    00    00    00    00    00    00    00    00    00    00    00    00    00    00
00    00    00    00    00    00    00    00    00    00    00    00    00    00    00
00    00    00    00    00    00    00    00    00    00    00    00    00    00    00
00    00    00    00    00    00    00    00    00    00    00    00    00    00    00
00    00    00    00    00    00    00    00    00    00    00    00    00    00    00
00    00    00    00    00    00    00    00    00    00    00    00    00    00    00
```

```
00    00    00    00    00    00    00    00    00    00    00    00    00    00    00
00    00    00    00    00    00    00    00    00    00    00    00    00    00    00
00    00    00    00    00    00    00    00    00    00    00    00    00    00    00

Executed Instructions:
PC,  Instruction,       MC,    Tag, Comment, Registers
---------------------------------------------------------------------
                                              00 00 00 00 00 00 00 ff
0x00 ADDI  $-1,  Sp     4fff   0   (null)
                                              00 00 00 00 00 00 00 fe
0x01   SW  Bp,   +0(Sp) 16e0   0   (null)
                                              SW $00, mem[fe]
0x02 MOV   Bp,   Sp     1ee0   0   (null)
                                              00 00 00 00 00 00 fe fe
0x03 SUBI  $+1,  Sp     5f01   0   (null)
                                              00 00 00 00 00 00 fe fd
0x04 MOVI  $62,  Ax     2062   0   (null)
                                              62 00 00 00 00 00 fe fd
0x05   SW  Ax,   -1(Bp) 10df   0   (null)
                                              SW $62, mem[00]
0x06   LW  Ax,   -1(Bp) 08df   0   (null)
                                              LW mem[00], $62
                                              62 00 00 00 00 00 fe fd
0x07 WRITE Ax,   UART1  9800   0   (null)
                                              PORT 0
                                              UART <- 'b' 0x62
0x08 MOVI  $65,  Ax     2065   0   (null)
                                              65 00 00 00 00 00 fe fd
0x09   SW  Ax,   -1(Bp) 10df   0   (null)
                                              SW $65, mem[00]
0x0a   LW  Ax,   -1(Bp) 08df   0   (null)
                                              LW mem[00], $65
                                              65 00 00 00 00 00 fe fd
0x0b WRITE Ax,   UART1  9800   0   (null)
                                              PORT 0
                                              UART <- 'e' 0x65
0x0c MOVI  $6e,  Ax     206e   0   (null)
                                              6e 00 00 00 00 00 fe fd
0x0d   SW  Ax,   -1(Bp) 10df   0   (null)
                                              SW $6e, mem[00]
0x0e   LW  Ax,   -1(Bp) 08df   0   (null)
                                              LW mem[00], $6e
                                              6e 00 00 00 00 00 fe fd
0x0f WRITE Ax,   UART1  9800   0   (null)
                                              PORT 0
                                              UART <- 'n' 0x6e
0x10 MOVI  $20,  Ax     2020   0   (null)
                                              20 00 00 00 00 00 fe fd
0x11   SW  Ax,   -1(Bp) 10df   0   (null)
                                              SW $20, mem[00]
0x12   LW  Ax,   -1(Bp) 08df   0   (null)
                                              LW mem[00], $20
                                              20 00 00 00 00 00 fe fd
```

```
0x13 WRITE Ax,   UART1   9800   0   (null)
                                             PORT 0
                                             UART <- ' ' 0x20
0x14 MOV   Sp,   Bp      1fc0   0   (null)
                                             20 00 00 00 00 00 fe fe
0x15   LW  Bp,   +0(Sp)  0ee0   0   (null)
                                             LW mem[fe], $00
                                             20 00 00 00 00 00 00 fe
0x16 ADDI  $+1,  Sp      4f01   0   (null)
                                             20 00 00 00 00 00 00 ff
0x17 HALT                9000   0   (null)

End memory contents:
00      01      02      03      04      05      06      07      08
=======================================================================================
4fff    16e0    1ee0    5f01    2062    10df    8df     9800    2065
10df    8df     9800    206e    10df    8df     9800    2020    10df
8df     9800    1fc0    ee0     4f01    9000    00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      00      00      00      00      00      00      00      00
00      20      00

Final Registers:
20 00 00 00 00 00 00 ff

UART tx buf:
ben
```

The output of the emulator starts with the disassembly of the processors input. As the instructions are reconstructed from the 16-bit instruction words, we lose additional information displayed by the compiler when this program was originally compiled, such as inline comments explaining the code-generation routine used (last column) and any assembler tags (second-last column) used by the assembler (see *libprco/arch/prco_isa.h:72*).

Next follows the initial memory layout of the processor. This shows where each instruction word is placed in memory. The emulator limits the displayed memory to the first 255 words of memory.

Following this is the list of executed instructions. Instruction execution starts at PC (program counter) `0x00`. On the right side, the contents of each memory is displayed. For complex instructions such as `LW/SW` and `WRITE` additional information is printed to verify correct ALU operation.

### 7.2.4 PRCO304 Compiler Continuous Integration Tests

```
-- GCC detected, adding compile flags
-- GCC detected, adding compile flags
-- GCC detected, adding compile flags
-- Configuring done
-- Generating done
-- Build files have been written to: /mnt/e/XilinxVM/prco304/prco_compiler/lbuild
[ 75%] Built target libprco
[ 87%] Built target cli
[100%] Built target emu
/mnt/e/XilinxVM/prco304/prco_compiler/test
Running test    ./tests/binary_ops_1.prco...     PASSED
Running test    ./tests/binary_ops_2.prco...     PASSED
Running test    ./tests/control_for_1.prco...    PASSED
Running test    ./tests/control_for_2.prco...    PASSED
Running test    ./tests/control_for_3.prco...    FAILED          Expected 1, got 1
Running test    ./tests/control_if_1.prco....    FAILED

/travis-ci.sh: line 17:  4301 Segmentation fault      (core dumped)
../lbuild/cli/cli -i $1 -d -D0x0002

Running test    ./tests/control_if_2.prco...     PASSED
Running test    ./tests/control_if_2.prco...     FAILED          Expected 32, got 1
Running test    ./tests/control_while_1.prco...  PASSED
Running test    ./tests/control_while_2.prco...  PASSED
Running test    ./tests/control_while_3.prco...  FAILED          Expected 5, got 1
Running test    ./tests/control_while_4.prco...  PASSED
Running test    ./tests/foo.prco...              PASSED
Running test    ./tests/funcs_1.prco...          PASSED
Running test    ./tests/funcs_2.prco...          PASSED
Running test    ./tests/ports_uart_1.prco...     PASSED
Running test    ./tests/strings_1.prco...        PASSED
Running test    ./tests/strings_2.prco...        PASSED
Running test    ./tests/strings_3.prco...        FAILED          Expected 1, got 1
Running test    ./tests/vars_1.prco...           PASSED
Running test    ./tests/vars_2.prco...           PASSED

17/21 passed.
```