

Application Note: AN10009

# How to debug a program at the machine instruction level

This application note is a short how-to on programming/using the xTIMEcomposer tools. It shows how to debug a program at the machine instruction level.

---

## Required tools and libraries

This application note is based on the following components:

- xTIMEcomposer Tools - Version 14.0.0

## Required hardware

Programming how-tos are generally not specific to any particular hardware and can usually run on all XMOS devices. See the contents of the note for full details.

## 1 How to debug a program at the machine instruction level

As well as working at the source level, XGDB can be used to debug programs at the level of the machine instructions. As an example, consider the following program:

```
int main() {
    int i, j = 0;
    for (i = 0; i < 5; ++i) {
        j += i;
    }
    return 0;
}
```

Compile the above program ensuring that debug is enabled (-g):

## 2 From xTIMEcomposer Studio

Create a new debug configuration via *Run->Debug Configurations->xCORE Applications*. Set a breakpoint on *main* and start debugging. Execution will now break when *main* is reached. Open both the *Disassembly* and the *Debug* views. In the *Debug* view toolbar, click on the *Instruction Stepping Mode* button. When in this mode, a single step will now correspond to a single machine instruction and not a source line. This can be seen by stepping a few times and watching the progress in the *Disassembly* view.

### 3 From the command line

Start XGDB, connect to the simulator and set a breakpoint on *main*. Execution will therefore suspend when the *main* is reached. The *stepi* command can be used to step a single machine instruction, and the *disassemble* command can be used to see the disassembly of the code close to the current PC location:

```
> xgdb a.xe
..etc..
(gdb) connect -s
0xfffffc070 in ?? ()
(gdb) b main
Breakpoint 1 at 0x100b0: file debugging_at_the_instruction_level.xc, line 9.
(gdb) r
...etc...
Breakpoint 1, main () at debugging_at_the_instruction_level.xc:9
9     int i, j = 0;
(gdb) disassemble
Dump of assembler code for function main:
0x000100ac <main+0>:      entsp (u6)      0x4
0x000100ae <main+2>:      ldc (ru6)       r0, 0x0
0x000100b0 <main+4>:      stw (ru6)       r0, sp[0x0]
0x000100b2 <main+6>:      stw (ru6)       r0, sp[0x1]
0x000100b4 <main+8>:      bu (u6)         0x7
0x000100b6 <main+10>:     ldw (ru6)       r0, sp[0x1]
0x000100b8 <main+12>:     ldw (ru6)       r1, sp[0x0]
0x000100ba <main+14>:     add (3r)        r0, r1, r0
0x000100bc <main+16>:     stw (ru6)       r0, sp[0x0]
0x000100be <main+18>:     ldw (ru6)       r0, sp[0x1]
0x000100c0 <main+20>:     add (2rus)      r0, r0, 0x1
0x000100c2 <main+22>:     stw (ru6)       r0, sp[0x1]
0x000100c4 <main+24>:     ldw (ru6)       r0, sp[0x1]
0x000100c6 <main+26>:     ldc (ru6)       r1, 0x5
0x000100c8 <main+28>:     lss (3r)        r0, r0, r1
0x000100ca <main+30>:     bt (ru6)        r0, -0xb
0x000100cc <main+32>:     bu (u6)         0x0
0x000100ce <main+34>:     ldc (ru6)       r0, 0x0
0x000100d0 <main+36>:     stw (ru6)       r0, sp[0x2]
0x000100d2 <main+38>:     ldw (ru6)       r0, sp[0x2]
0x000100d4 <main+40>:     retsp (u6)      0x4
End of assembler dump.
(gdb) stepi
10    for (i = 0; i < 5; ++i) {
(gdb) stepi
0x000100b4 10        for (i = 0; i < 5; ++i) {
(gdb) stepi
0x000100c4 10        for (i = 0; i < 5; ++i) {
(gdb) stepi
0x000100c6 10        for (i = 0; i < 5; ++i) {
(gdb) stepi
0x000100c8 10        for (i = 0; i < 5; ++i) {
...etc...
```

its use. Xmos Ltd. makes no representation that the Information, or any particular implementation thereof, is or will be free from any claims of infringement and again, shall have no liability in relation to any such claims.