GDB for RISC-V
Extending Support for Bare Metal Multi-core Debugging

Jeremy Bennett
Andrew Burgess
GDB in the RISC-V World

Debug Spec Task Group

Technical Committee

SW Tool Chain
Upstream GDB

• Committed upstream 6 March 2018
  - basic bare metal support

• Combined effort of several engineers:
  - Andrew Burgess, Tim Newsome, Albert Ou, Darius Rad
  - official maintainers are Andrew Burgess & Palmer Dabbelt

• Nightly regression testing
  - against GDBsim and remote GDBserver (wrapping GDBsim)
  - range of architectures from RV32IM thru’ RM64IMFDC
  - >99% pass rate on all architectures
### GDB Regression Tests (Sim Subset)

<table>
<thead>
<tr>
<th></th>
<th>RV32IM</th>
<th>RV32IMFC</th>
<th>RV64IM</th>
<th>RV64IMFDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASS</td>
<td>35,936</td>
<td>36,050</td>
<td>35,965</td>
<td>36,067</td>
</tr>
<tr>
<td>FAIL</td>
<td>309</td>
<td>190</td>
<td>303</td>
<td>194</td>
</tr>
<tr>
<td>XPASS</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XFAIL</td>
<td>30</td>
<td>30</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>KPASS</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>KFAIL</td>
<td>53</td>
<td>53</td>
<td>51</td>
<td>51</td>
</tr>
<tr>
<td>UNRESOLVED</td>
<td>12</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>UNTESTED</td>
<td>231</td>
<td>231</td>
<td>229</td>
<td>229</td>
</tr>
<tr>
<td>UNSUPPORTED</td>
<td>326</td>
<td>326</td>
<td>326</td>
<td>326</td>
</tr>
</tbody>
</table>
# GDB Regression Tests (GDBserver Subset)

<table>
<thead>
<tr>
<th></th>
<th>RV32IM</th>
<th>RV32IMFC</th>
<th>RV64IM</th>
<th>RV64IMFDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASS</td>
<td>35,746</td>
<td>35,856</td>
<td>35,793</td>
<td>35,896</td>
</tr>
<tr>
<td>FAIL</td>
<td>287</td>
<td>169</td>
<td>261</td>
<td>151</td>
</tr>
<tr>
<td>XPASS</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XFAIL</td>
<td>30</td>
<td>30</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>KPASS</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>KFAIL</td>
<td>53</td>
<td>53</td>
<td>51</td>
<td>51</td>
</tr>
<tr>
<td>UNRESOLVED</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>UNTESTED</td>
<td>232</td>
<td>232</td>
<td>230</td>
<td>230</td>
</tr>
<tr>
<td>UNSUPPORTED</td>
<td>318</td>
<td>318</td>
<td>318</td>
<td>318</td>
</tr>
</tbody>
</table>
Upstream GDB Next Steps

- Adding XML target description support
  - initial work removed for now, aim for OpenOCD compatibility
- Memory map support
- Remote I/O support
- Adding non-DWARF stack unwinding
- Upstreaming a GDB simulator——possibly CGEN based
  - see Mary Bennett's talk/poster later
- Linux application debugging
  - both native and via Linux gdbserver application
GDB Historic View of Multicore Debug

- Multiple inferiors OK
  - only one active at once
  - can handle `fork()`
- Address space per process
- Threads share an address space
- All-stop and non-stop execution
GDB Future View of Multicore Debug

- Multiple active inferiors
  - each a concurrent flow of control
- Each inferior associated with an address space
  - which memory can it see?
- Each inferior associated with a program space
  - which symbol table relates to its code?
  - breakpoints across multiple inferiors with the same program space
- All-stop and non-stop execution
  - inferiors not just threads
Current Status

• Upstream GDB supports multiple concurrent inferiors
  - works for native/Linux and single remote target
  - generic work by Pedro Alves and others to support on multiple remote targets

• GDB for RISC-V has this functionality
  - tested with 36-core RV64IMC system
  - working on a public version based on PULP

• More work is needed for complex address spaces
  - for example where some memory is shared with other inferiors
Thank You

www.embecosm.com