Debugging and profiling embedded Linux/CRIS systems with QEMU

Edgar E. Iglesias <edgar@axis.com>
Talk

• Why
• CRIS & ETRAX
  • Quick overview
• QEMU
  • Overview
  • Debugging & Profiling features
• Summary
• Questions
Why

• We didn't have an emulator
  • Fast
  • Easy to use and extend
  • Powerful debug capabilities
• QEMU - fun hobby project
CRIS

- **Code Reduced Instruction Set**
  - ISA designed for small footprint.
  - GNU toolchain (binutils, GCC, GDB).

- **CRISv8 (1999)**
  - Designed to be small and for low power consumption.
  - 2-stage pipeline @ 100Mhz.
  - uClinux (no MMU).

- **CRISv10 (2000)**
  - Standard Linux (with MMU).

- **CRISv32 (2004)**
  - 5-stage pipeline @ 200Mhz.
CRIS

- Code Reduced Instruction Set
  - ISA designed for small footprint.
  - GNU toolchain (binutils, GCC, GDB).
- CRISv8 (1999)
  - Designed to be small and for low power consumption.
  - 2-stage pipeline @ 100Mhz.
  - uClinux (no MMU).
- CRISv10 (2000)
  - Standard Linux (with MMU).
  - 5-stage pipeline @ 200Mhz.
CRISv32

- 32-bit RISC architecture
  - Variable length (16bit) insn encoding.
- 5-stage pipeline
  - Load+operate
  - Enforces dependencies by interlocks.
    - 2-stage Multiplier shares stage with MEM
    - Auto-increment has no regforwarding
    - load/store multiples lack regforwarding
  - Delayed branches
- MMU / TLB
  - 16 segments (linear or 8Kb paged).
  - 8-bit ASID, 64 entries.
- L1 Cache
  - 2 x 16Kb 2way VIPT.
  - Coherent (buggy).
- No performance counters
ETRAX

• **Ethernet Token Ring AXis**
  • Family of networking chips
  • Lot's of I/O
    • SCSI, IDE
    • Ethernet, TokenRing
    • USB, Parallel ports, Serial ports
    • Etc..
  • Print Servers, Storage Servers, Scan Servers, Network Cameras, Network Video Servers etc.

• **ARTPEC**
  • AXIS family of video processing chips.
AXIS Communications

- Video surveillance
  - Cameras, Video encoders, Decoders, SW etc

- Early with embedded linux
“QEMU is a generic and open source machine emulator and virtualizer.”
“QEMU is a generic and open source machine emulator and virtualizer.”
QEMU

• **System emulation**
  • Emulates a complete machine.
  • Cross run unmodified OS/Firmware.
  • Can also emulate boot-roms including different bootstrap methods.

• **Linux-user emulation**
  • Emulates the target processor.
  • Cross run linux programs.
  • Syscalls run natively on the host (through an argument translator).
QEMU

• **System emulation**
  - Emulates a complete machine.
  - Cross run unmodified OS/Firmware.
  - Can also emulate boot-rooms and including different bootstrap methods.

• **Linux-user emulation**
  - Emulates the target processor.
  - Cross run Linux programs.
  - Syscalls run natively on the host (through an argument translator).
QEMU

Does not continuously interpret guest ISA. Instead it translates guest machine code into host code.

- Fetching only done at translation time.
- Instruction decoding only done at translation time.
- Basic optimization at translation time.
- Lazy Condition Code flags evaluation.
Dynamic Translation

• On demand translation of instruction sequences from target to host ISA. The result is referred as a Translation Block.

• Translation is done through a portable intermediate generic code generator, Tiny Code Generator (TCG).
Tiny Code Generator

- Per CPU target translators translate guest code into TCG operations.

- TCG runs generic optimization passes.
  - Basic stuff, Regalloc, liveness analysis etc.

- TCG backends emit host machine code.
Translation

CRIS:
move.d $r9, $r10
ret
addq 3, $r10
...

TCG:
mov_i32 $r10, $r9
movi_i32 cc_x, $0x0
mov_i32 cc_result, $r10
---
mov_i32 btarget, $srp
movi_i32 tmp0, $0xffffffff
and_i32 btarget, btarget, tmp0
movi_i32 btaken, $0x1
...

TCG x86 backend:
mov 0x24(%ebp), %eax
mov 0x6c(%ebp), %edx
...
TCG Helpers

• Subroutine calls from TB
  • TCG needs to writeback and reload the CPUState around the call due to aliasing and helper side-effects.
  • PURE | CONST helpers avoid wb & reloading.
  • Nice if you can easily identify a complex target instruction sequence.

• Compiled by host compiler
  • Optimization cost mostly taken at QEMU compile time.
Lazy CC evaluations

• CRIS has implicit updates
  • Emulators need to evaluate the condition code flags after every insn.

• Lazy evaluation
  • Save operation and operands.
  • Evaluate when there is a dependency to the flags.
QEMU TCG

• **Target ports (TCG translators):**
  - Alpha, ARM, CRIS, MIPS, m68k, PPC, SH, SPARC32/64, x86 and x86_64.

• **Host ports (TCG backends):**
  - ARM, HPPA, PPC, SPARC32, x86 and x86_64.
QEMU IO

• Memory accesses
  • No cache models.
  • No bus transfer models.
  • Limited bus topology modeling.

• SoftMMU
  • QEMU fast TLB caches slower guest TLB.
  • I faults taken between TB's.
  • D faults abort and retranslate the current TB with extra info to find the actual guest insn that caused the exception.

• Interrupts
  • Taken between TB's.
QEMU Peripherals

- Interrupt controllers
- DMA units
- Flash memories (NOR/NAND)
- Networking
  - Flexible ways to connect to the host.
  - Support for DMA and PHY control.
- IDE / SCSI controllers
- Serial ports
- Graphic adapters
- Audio adapters
- More..
QEMU Peripherals

- Provide registration function
- Register callbacks for control register access
- Combinational logic
- Timers
- Interrupts
- QEMU I/O
  - Networking
  - Serial ports
  - IPC
  - etc...
QEMU Boards

- Instantiate CPU cores
- Define Address map
- Wire up all the devices
- Load kernel/OS images
Debugging and Profiling CRIS

- Builtin GDB stub
- Execution traces
- L1 Cache model
- Processor pipeline model
- Interrupt latency tracker
- Kcachegrind compatible statistics
- Track peripheral programming inefficiencies and errors
Built-in GDB stub

- Non-intrusive
- Controllable from first executed insn
- HW Breakpoints
- HW Watchpoints
- VM time stops while halted
- Configurable interrupts while single-stepping
- Experimental patch for tracepoints
CRIS Cache

• **L1 cache model**
  - Controller and tag memories.
  - Does not include the data path/memories.
  - Snoops on other bus masters.

<table>
<thead>
<tr>
<th>Index</th>
<th>Tag</th>
<th>Valid</th>
<th>Dirty</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>x</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>x</td>
</tr>
<tr>
<td>...</td>
<td>x</td>
</tr>
</tbody>
</table>
CRIS Cache

- **QEMU Cache tag memories**
  - Not really bound by size.
  - Extended with debug info.
    - Virtual Address for the access.
    - Virtual PC for the access.
    - One dirty bit per line word.
- Connected to GDB
  - Stop execution on cache-miss (misspoints)

<table>
<thead>
<tr>
<th>Index</th>
<th>Tag</th>
<th>Valid</th>
<th>Dirty</th>
<th>VPC</th>
<th>Vaddr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>0</td>
<td>000..</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>1</td>
<td>010..</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>x</td>
<td>1</td>
<td>110..</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>x</td>
<td>0</td>
<td>000..</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CRIS Cache

- Track wasted writeback cycles
  - Due to fragmented store patterns
CRIS Cache

- Track wasted writeback cycles
  - Due to fragmented store patterns
- Reorganize global data
  - __read_mostly attribute
- Reorganize structures
CRIS Cache

- Cache snoops on DMA accesses
  - Incoherence warnings.
  - TODO: Debugger breakpoints
CRIS Pipeline

- Work in progress
- Intra TB
  - Computed at translation time.
  - Fast but not all locked cycles are seen.
- No branch prediction
- Logs PC address and symbol name
Interrupt Latency

- Track IRQ masking (CPU line).
- Log long paths.
  - Time estimate based on core frequency, instruction count, interlock cycles and cache statistics.
- Helps reducing:
  - Interrupt latency
  - Jitter

c0010156 (badcode) -> c0010338 (badcode) lr=c0010330 9632 insns 10398 cycles 41592ns
Interrupt Latency

{
    unsigned long flags;

    spin_lock_irq_save(&lock1, flags);

    /* code. */
    If (something) {
        spin_lock_irq_save(&lock2, flags);
        /* More critical code. */
        spin_unlock_irqrestore(&lock2, flags);
    }
    /* code. */

    spin_unlock_irqrestore(&lock1, flags);
}
Kcachegrind

- Instruction count per function
  - Instructions with interrupts masked
- Cycle estimate per function
  - Cache model
  - Pipeline model
- No callgraphs (TODO)
Peripheral Programming

Warn for control register programming errors and inefficiencies:

- Duplex mismatch MAC / PHY.
- Illegal combinations/setups.
- Unnecessary control register accesses.
- Enforce reserved fields
Peripheral Programming

Simplified view

```
<table>
<thead>
<tr>
<th>ARM926 Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 Cache</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA</td>
</tr>
<tr>
<td>DMA</td>
</tr>
</tbody>
</table>

| DRAM |

| Bridge |

<table>
<thead>
<tr>
<th>Peripheral</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM</td>
</tr>
</tbody>
</table>
```
Peripheral Programming

```
struct ram_entry
{
    u16 ctrl;
    u16 pos;
    ...
};

volatile struct ram_entry *e = SOME_ADDRESS;
if (e->ctrl & 1)    { ... }
if (e->ctrl & 2)    { ... }
if (e->ctrl & 4)    { ... }
... more...

• Compiler emits loads/stores resulting in deep bus transfers for every access to ctrl!
```
Axis Devices

- ETRAX-FS
  - Bare FS virtual machine
  - Axis Devboard 88
- ARTPEC-3
  - P3301, Q7401
- ARTPEC-4
  - Prototype of virtual machine
- ARTPEC-B
  - M3011
Axis Devices

- ETRAX-FS / ARTPEC-3
  - CRIS core
  - L1 Cache
  - MMU
  - PIC
  - Timers
  - DMA
  - Ethernet with PHY models
  - Asynch Serial Ports
  - PIO (NAND flashes)
  - NOR flashes
  - GPIO
    - Temperature sensors (i2c)
Axis Devices

• ARTPEC-B
  • ARM926 core
  • MMU
  • PIC
  • Timers
  • Ethernet with PHY models
  • Asynch Serial Ports
  • NAND Controller
  • GPIO
  • Stub for RASC interface
    • Just enough to boot.
Future work

- Emulate media sources
  - Image and Audio pipelines.
  - Codecs.
- Linux aware debugging
  - Track kernel memory allocations.
  - Kernel modules debuginfo.
  - Track user-space processes in system emulation.
- TLB profiler
Summary

- Early access to hardware
  - Initial testing
- Debugging
  - Early boot code
  - Cache incoherence
  - Debug the debug code
- Profiling
  - Interrupt latency
  - Improve cache performance
  - Avoid interlock cycles in hot loops
- Testing (future)
  - FW up/downgrades
  - Instrumentation
    - I/O Stimulus
Questions

Thanks for listening

URL: git://repo.or.cz/qemu/cris-port.git
edgar@axis.com