QEMU internals

Chad D. Kersey

January 28, 2009

Where to get the source

```
svn co svn://svn.savannah.nongnu.org/qemu
```

Make sure you have the latest sources if you're reading along. A lot has changed since the previous release.

Functional simulation

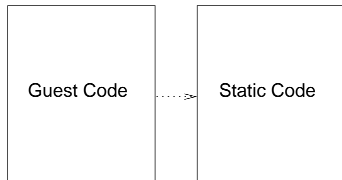
- Simulate what a processor does, not how it does it.
- Needs separate model for timing analysis (if needed).
- Faster than “cycle-accurate” simulators.
- Good enough to use applications written for another CPU.

QEMU system simulation

- QEMU simulates VGA, serial, and ethernet.
- `hw/*` contain all of the supported boards.
- Includes rather complete PC, Nokia N-series, PCI ultrasparc.
- Various development boards in varying levels of completion.

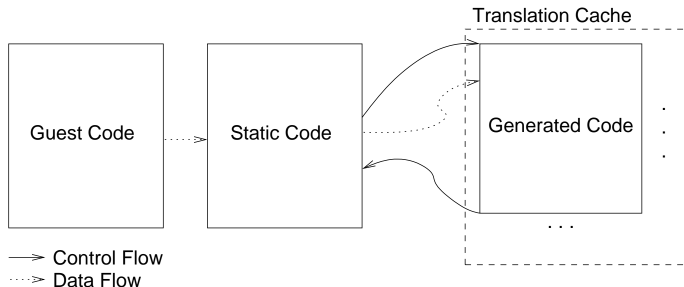
What dynamic translation isn't

- Interpreters execute instructions one at a time.
- Significant slowdown from constant overhead.
- Easier to write and debug than dynamic translators.



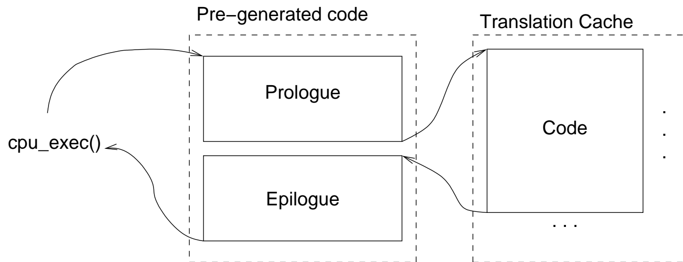
What dynamic translation is

- Dynamic translators convert code as needed.
- Try to spend most time executing in translation cache.
- Translate basic blocks as needed.
- Store translated blocks in code cache.

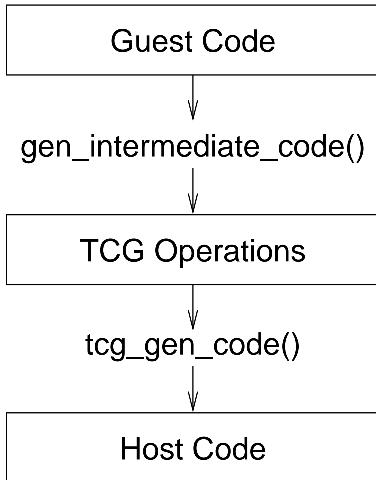


Getting into and out of the code cache

- cpu_exec() called each time around main loop.
- Program executes until an unchained block is encountered.
- Returns to cpu_exec() through epilogue.

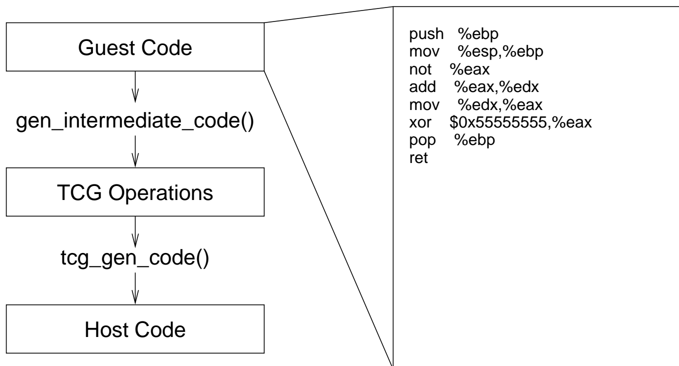


Portable dynamic translation

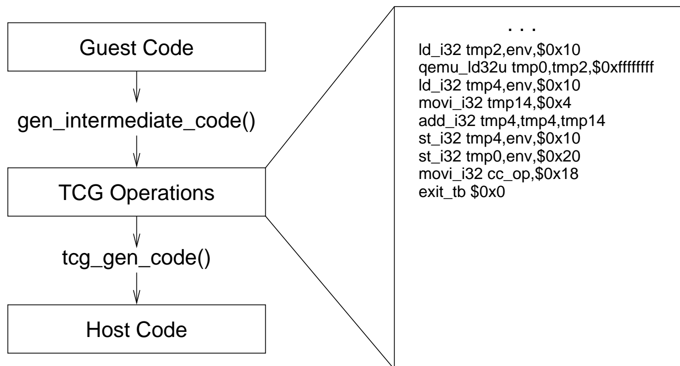


- QEMU uses an intermediate form.
- Frontends are in `target-*/`
- Backends are in `tcg/*/`
- Selected with preprocessor evil.

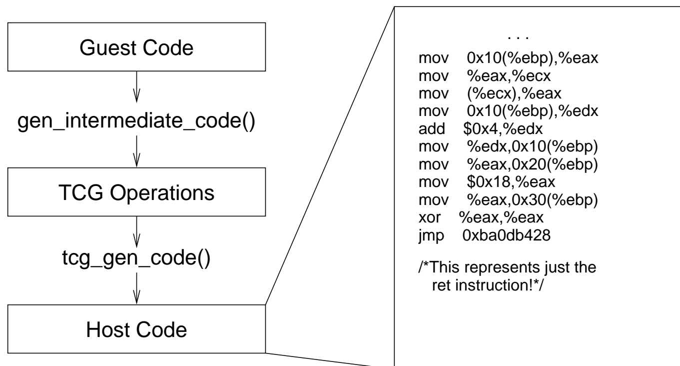
Portable dynamic translation: stage 1



Portable dynamic translation: stage 2

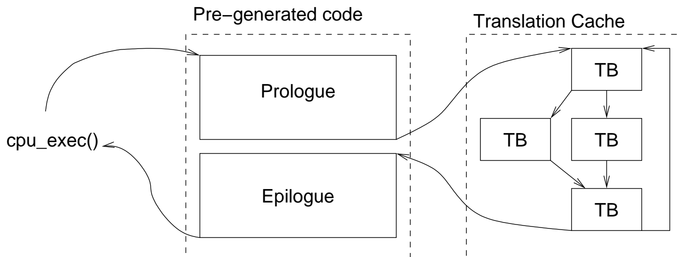


Portable dynamic translation: stage 3

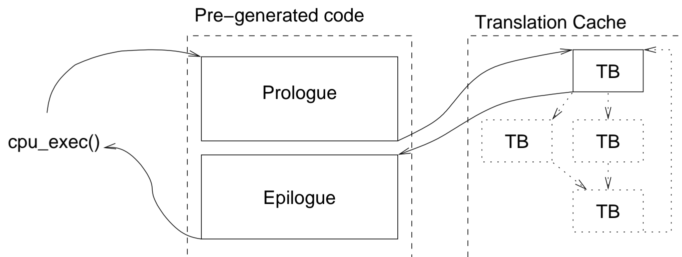


Basic block chaining

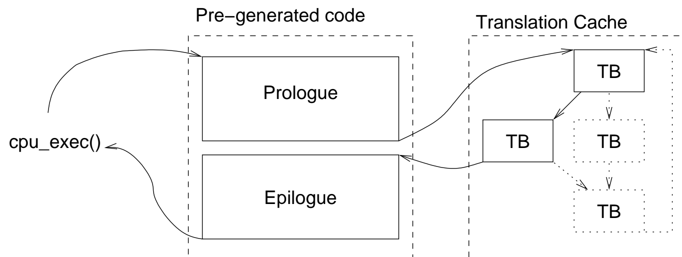
- Returning from code cache is slow.
- Solution: jump directly between basic blocks!
- Make space for a jump, follow by a return to the epilogue.
- Every time a block returns, try to chain it.



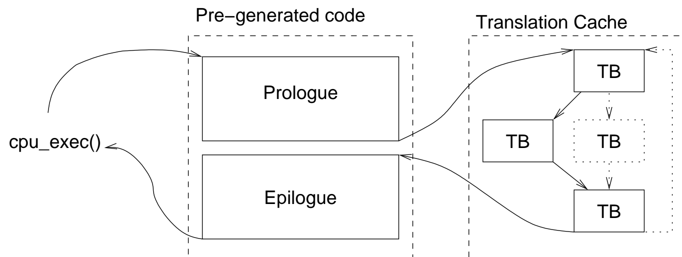
Basic block chaining: step 1



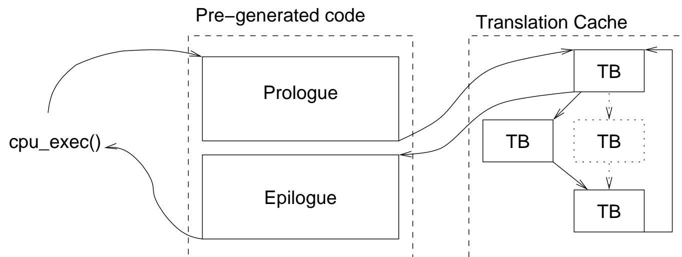
Basic block chaining: step 2



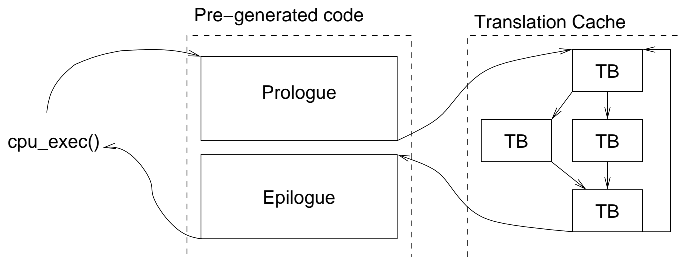
Basic block chaining: step 3



Basic block chaining: step 4

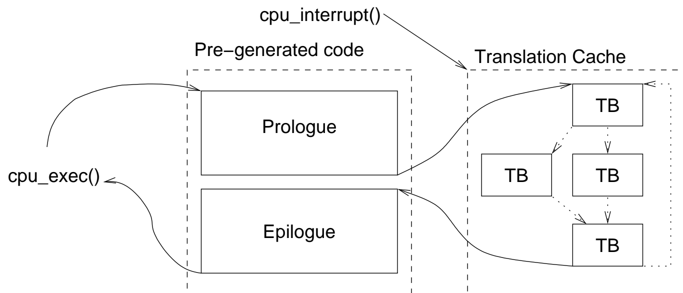


Basic block chaining: step 5



Unchain on interrupt

- Now how do we interrupt the processor?
- Have another thread unchain the blocks.



Code organization

- TranslationBlock structure in `translate-all.h`
- Translation cache is `code_gen_buffer` in `exec.c`
- `cpu-exec()` in `cpu-exec.c` orchestrates translation and block chaining.
- `target-*/translate.c`: guest ISA specific code.
- `tcg-*/*/`: host ISA specific code.
- `linux-user/*`: Linux usermode specific code.
- `vl.c`: Main loop for system emulation.
- `hw/*`: Hardware, including video, audio, and boards.

Ways to have fun

- Add extra instructions to an ISA.
- Generate execution traces to drive timing models.
- Try to integrate timing models.
- Retarget frontend or backend.
- Improve optimization, say, by retaining chaining across interrupts.

Acknowledgments

- QEMU by Fabrice Bellard: www.bellard.org/
- Current qemu-internals:
  http://bellard.org/qemu/qemu-tech.html
- Some graphics in these slides part of work funded by DOE grant.

Questions?