FEX-EMU

Interactions in an emulated environment

Ryan Houdek
@Sonicadvance1
IRC: HdkR

Who Am I?

- I’m in the Mesa IRC channels and elsewhere
- Emulators for more than a decade
- Started working on FEX-Emu in early 2019
- AArch64 snob

What is FEX-Emu?

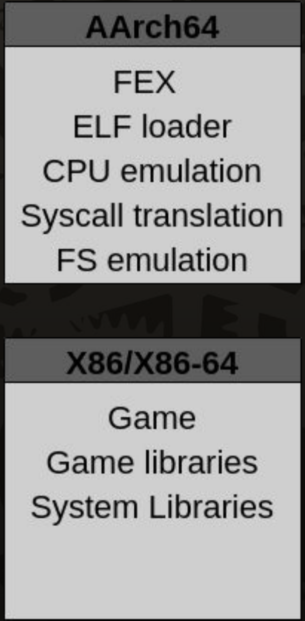
- Userspace mode x86 and x86-64 emulator
- Backwards compatibility on AArch64
- Fast enough to be usable for real gaming
- JITs!
- Lots of moving parts
  - Quickly improving
- MIT code license

Crossing the architecture boundary

- A couple of different styles to consider
  - Full emulation
  - Library “thunking”
- This changes where the ABI boundary gets crossed

Full library emulation

- Most compatible
- Syscalls jump to ARM64
- Libraries fully emulated
  - GL, Vulkan, X11, SDL, etc
- Tricky implications for drivers
- NVIDIA blob obviously won’t work

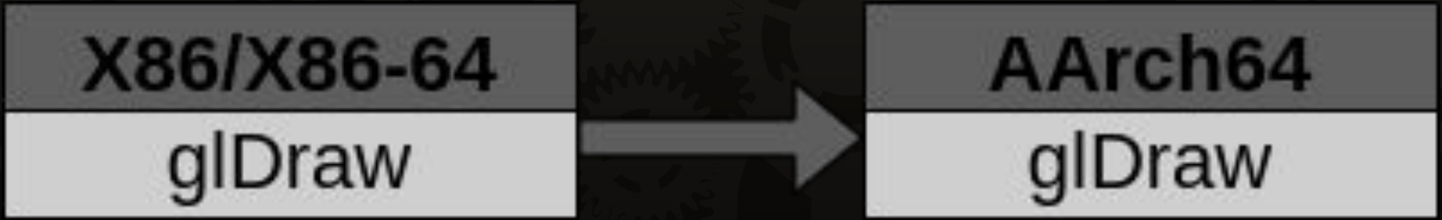


Emulated Mesa driver implications

- Drivers built for x86/x86-64 that were never tested before
  - Freedreno, Turnip, Panfrost, v3d, asahi, ...?
- What does this mean for Mesa devs?
  - Do *not* optimize for this case
  - But please keep arch specific optimizations in architecture checks
  - Thank you for accepting the fpmath=sse patch

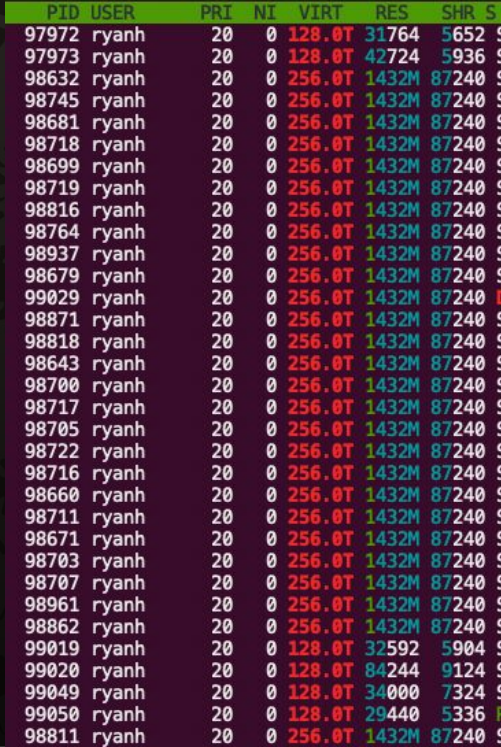
Library “thunking” implications

- What is library thunking?
  - Lighter than VIRGL
- How is this different?



Virtual address space problems

- 32-bit -> 64-bit transformation
  - Virtual address space is 4GB(!)
- 64-bit -> 64-bit transformation
  - 128TB of VA deleted!
- ARM supports BIG VA
  - 39, 48, 36*, 42*, 47*, 52*
- Kernel memory usage?



```
  PID USER      PRI  NI  VIRT   RES   SHR S
97972 ryanh      20   0 128.0T 31764  5652
97973 ryanh      20   0 128.0T 42724  5936
98632 ryanh      20   0 256.0T 1432M 87240
98745 ryanh      20   0 256.0T 1432M 87240
98681 ryanh      20   0 256.0T 1432M 87240
98718 ryanh      20   0 256.0T 1432M 87240
98699 ryanh      20   0 256.0T 1432M 87240
98719 ryanh      20   0 256.0T 1432M 87240
98816 ryanh      20   0 256.0T 1432M 87240
98764 ryanh      20   0 256.0T 1432M 87240
98937 ryanh      20   0 256.0T 1432M 87240
98679 ryanh      20   0 256.0T 1432M 87240
99029 ryanh      20   0 256.0T 1432M 87240
98871 ryanh      20   0 256.0T 1432M 87240
98818 ryanh      20   0 256.0T 1432M 87240
98643 ryanh      20   0 256.0T 1432M 87240
98700 ryanh      20   0 256.0T 1432M 87240
98717 ryanh      20   0 256.0T 1432M 87240
98705 ryanh      20   0 256.0T 1432M 87240
98722 ryanh      20   0 256.0T 1432M 87240
98716 ryanh      20   0 256.0T 1432M 87240
98660 ryanh      20   0 256.0T 1432M 87240
98711 ryanh      20   0 256.0T 1432M 87240
98671 ryanh      20   0 256.0T 1432M 87240
98703 ryanh      20   0 256.0T 1432M 87240
98707 ryanh      20   0 256.0T 1432M 87240
98961 ryanh      20   0 256.0T 1432M 87240
98862 ryanh      20   0 256.0T 1432M 87240
99019 ryanh      20   0 128.0T 32592  5904
99020 ryanh      20   0 128.0T 84244  9124
99049 ryanh      20   0 128.0T 34000  7324
99050 ryanh      20   0 128.0T 29440  5336
98811 ryanh      20   0 256.0T 1432M 87240
```

*Not really around in consumer class

** 57-bit x86-64 doesn't matter to us

The dangers of ioctls

- x86 and AArch64 encoding can and will mismatch
- Strong emphasis on DRM correctness
  - The less repacking the better
  - 38 currently require repacking
- Most DRM problems go away with thunks
- Will be taking a look at Asahi as soon as it calms down

X86 games an ARM GPU drivers

- Tiling bugs!
- Only GL 3.x? Time for Zink!
  - Lots of games requiring GL 4.x
- Vulkan 1.2+ is very important!
  - Proton, VKD3D-Proton, DXVK, Zink

Thunking implementation details

- Call shim function in library
- Pack arguments in to a struct on the stack
- Jump to Arm64 shim handler
- Unpack and call Arm64 function
- Return result
- Currently supported
  - OpenGL, Vulkan, GLX, X11, XCB, VDSO
- Clang tooling to help automate data packing

Video Decode? V4L2?

- VA-API?
- VDPAU?
- Vulkan Video?
- Software decoding?
- Needs investigation

Future plans

- Thunk optimizations
- Wine PE-conversion integration
- More performance
- More compatibility

Demo

Where to find us

- Website: https://fex-emu.com
- Discord: https://discord.gg/fexemu
- Github: https://github.com/FEX-Emu/FEX
- Twitter: https://twitter.com/FEX_Emu
- Reddit: https://reddit.com/r/FexEmu

Extra

CPU Emulation

- Emulates x86 and x86-64 userspace
- Up to SSE4.1 feature-set
  - SSE4.2, AVX, AVX2 coming for latest games
- Intermediate representation JIT
  - IR caching, JIT code caching

ASM to IR to Host code example

```
● movups   xmm0, xmmword [rel zero]
▲ movups   xmm1, xmmword [rel one]
■ paddq    xmm0, xmm1
☺ hlt
```

```
BeginBlock %ssa2(Invalid)
%ssa4(GPR0) i64 = EntrypointOffset #0x13
%ssa5(FPR0) i128 = LoadMemTSO %ssa4(GPR0) i64, %Invalid, #0x1, FPR, SXTX, #0x1          ●
%ssa6(GPR0) i64 = EntrypointOffset #0x23
%ssa7(FPRFixed1) i128 = LoadMemTSO %ssa6(GPR0) i64, %Invalid, #0x1, FPR, SXTX, #0x1     ▲
StoreRegister %ssa7(FPRFixed1) i128, #0x0, #0xa0, FPR, FPRFixed
%ssa9(FPRFixed0) i64v2 = VAdd %ssa5(FPR0) i128, %ssa7(FPRFixed1) i128                   ■
StoreRegister %ssa9(FPRFixed0) i64v2, #0x0, #0x90, FPR, FPRFixed
%ssa11(GPR0) i64 = EntrypointOffset #0x13
StoreContext %ssa11(GPR0) i64, #0x0, GPR
Break #0x3, #0x0                                                                        ☺
EndBlock %ssa2(Invalid)
```

```
  mov     x20, #0x13
  movk    x20, #0x1, lsl #16
● ldr     q4, [x20]
  mov     x20, #0x23
  movk    x20, #0x1, lsl #16
▲ ldr     q17, [x20]
■ add     v16.2d, v4.2d, v17.2d
  mov     x20, #0x13
  movk    x20, #0x1, lsl #16
  str     x20, [x28]
☺ ldr     x0, [x28, #744]
  mov     sp, x0
  mov     x0, #0xd0e4
  movk    x0, #0xf7fe, lsl #16
  movk    x0, #0x7f, lsl #32
  br      x0
```

Unit testing for correctness

- Assembly tests
- IR tests
- GCC tests
- POSIX tests
- gVisor tests
- C/C++ tests
- Functional thunk testing