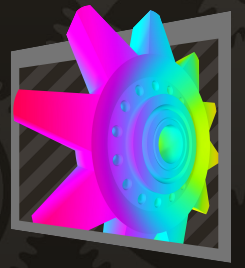
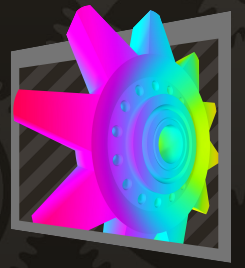


FEX-EMU

Fast(-er) x86 emulation for AArch64

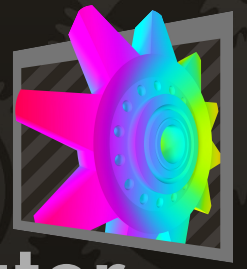
Ryan Houdek
@Sonicadvance1
IRC: HdkR

Who Am I?



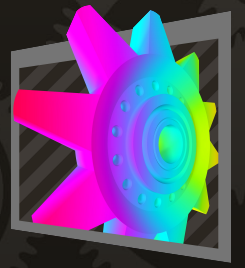
- Dolphin-Emu
  - GameCube, Wii, and Triforce emulator
- Ported some games to Android
  - Nothing worth mentioning
- FEX-Emu
  - It's this!

What is FEX-Emu?



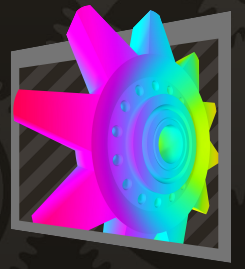
- Userspace mode x86 and x86-64 emulator
- Backwards compatibility for AArch64
- Fast enough to be usable for real gaming
- JITs!
- Lots of moving parts
  - Quickly improving
- MIT code license

What features will not be in FEX?



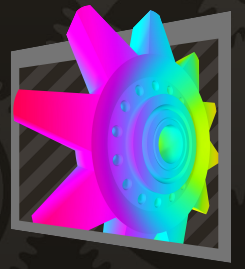
- Secure mode
- Hardware reference platform
- 16-bit x86
- 100% accurate
  - What to compare to even?

Other related software



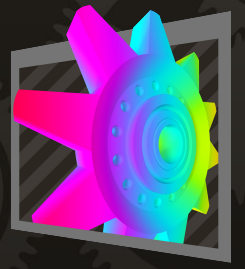
- QEMU-User
- Box86 and Box64
- Rosetta 2 in MacOS
- Microsoft XTA in Windows
- ElTechs Exagear
  - Popular in the Android space
- Intel Houdini

Interesting Problems



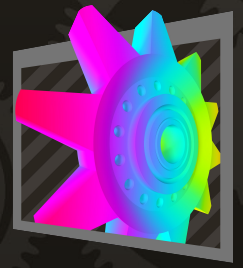
- CPU emulation scope is enormous
  - x86-64 instruction set is MASSIVE
- Linux wrapping is a daunting task
  - Lots of moving parts to get right
- Signals - Oh gods the signals
- Exception recovery

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

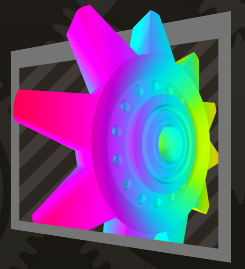


```
movups  xmm0, xmmword [rel zero]
movups  xmm1, xmmword [rel one]
paddq   xmm0, xmm1
hlt
```

```
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)
```

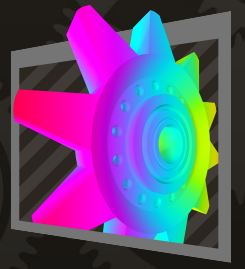
```
mov     x20, #0x13
movk    x20, #0x1, lsl #16
ldr     q4, [x20]
mov     x20, #0x23
movk    x20, #0x1, lsl #16
ldr     q17, [x20]
add     v16.2d, v4.2d, v17.2d
mov     x20, #0x13
movk    x20, #0x1, lsl #16
str     x20, [x28]
ldr     x0, [x28, #744]
mov     sp, x0
mov     x0, #0xd0e4
movk    x0, #0xf7fe,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

CPU Emulation



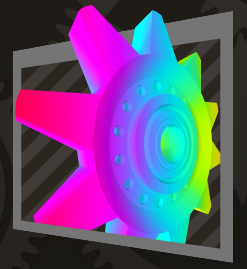
- Emulates x86 and x86-64 userspace
- Up to SSE4.1 feature-set
  - SSE4.2, AVX, AVX2 coming for latest games
- Intermediate representation JIT
  - IR caching, JIT code caching

CPU Emulation (cont.)



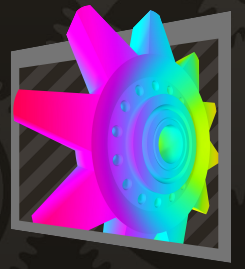
- Everything converts to a 64-bit process
- Cross ABI library interface thunking
  - 64-bit only, 32-bit is coming up
- ARM hardware losing 32-bit support

Linux Emulation/Wrapping



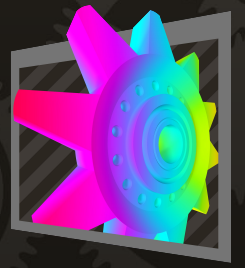
- Linux on Linux, not usually emulation
- Supports Linux 5.0 through 5.16
- Most syscalls and ioctls can passthrough
- Need to capture and emulate what doesn't
- Problems described Here

Hardware Assisted Emulation



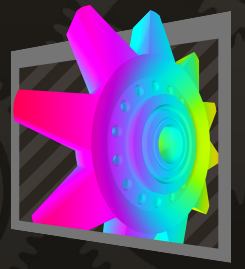
- AArch64 extensions
  - LSE, LSE2, RNG, RCPC, TME, SVE, SVE2 ...
- Float rounding alt. mode
- FPCR Exceptions
- 1Ghz virtual cycle counter
- Memcpy instructions
- x86-TSO memory model

Unit testing for correctness

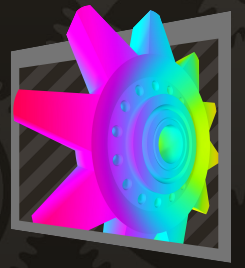


- Assembly tests
- IR tests
- GCC tests
- POSIX tests
- gVisor tests
- C/C++ tests

Future Endeavors

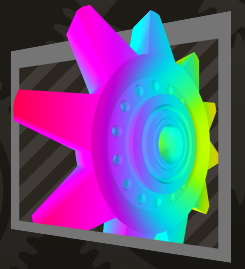


- Better/Faster code generation
- Fuzzing infrastructure
- More aggressive CI with Ampere Siryn
- More library thunking
- Self Modifying Code support
- Proton/Pressure-Vessel



Demo

Where to find us



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