Agenda

● Introduction on LCG ART Team
● Our accomplishments
  ○ Compiler optimisations
  ○ New architecture features enablement
  ○ Tools & testing improvements for ART
● Q & A
LCG ART Team

- A team of 5 engineers
- Most active contributors (non googler) to upstream ART project in past 12 months:
  - Linaro ART team: 133 patches merged
  - All others: 110 patches merged
- Our current focus:
  - 64-bit compiler performance and code size improvements.
  - Enabling ARM new architecture features in ART (e.g. Armv8.x).
  - Various testing improvements
- The LCG-390 initiative card
  - [https://projects.linaro.org/browse/LCG-390](https://projects.linaro.org/browse/LCG-390)
  - Includes all our current work and plans
ART Compiler Optimisations

● **Goal**
  ○ Make sure ARM backend delivers best performance for ART
  ○ Benefit our Linaro members and upstream.

● **All compiler optimisation work based on our deep-dive performance studies**
  ○ Our recent findings:
    ■ Strength reduction for HDiv/HRem operations
    ■ Array bound checks elimination improvements
    ■ Jump threading optimization opportunities
    ■ … and more.
  ○ Please attend Evgeny’s talk LVC20-311 for more detailed update.
ART Compiler Optimisations

- Optimisation highlights:
  - ART stackmap optimisation:
    - [https://android-review.googlesource.com/c/platform/art/+/870770](https://android-review.googlesource.com/c/platform/art/+/870770)
    - Saves 5% code size of boot.oat!
  - GetChars intrinsic NEON implementation
    - [https://android-review.googlesource.com/c/platform/art/+/1114515](https://android-review.googlesource.com/c/platform/art/+/1114515)
    - Up to 80% speedup on long strings getChars (Pixel3).
  - SIMD support in load store elimination:
    - [https://android-review.googlesource.com/c/platform/art/+/1217312](https://android-review.googlesource.com/c/platform/art/+/1217312)
  - Combine LSR+ASR into ASR for Int32 HDiv/HRem
    - [https://android-review.googlesource.com/c/platform/art/+/1308074](https://android-review.googlesource.com/c/platform/art/+/1308074)
Enable Armv8.x Features in ART

- In recent years, we’ve seen Android phones with ARMv8.2+ features starting to appear in the market. For our members/partners, it is important that the latest Arm architectural features are fully exploited in their products.

- ART compiler can optimise Android APIs and Java/Kotlin code for users with Armv8.x instructions, and can make big performance difference.
Enable Armv8.x Features in ART

- Optimise Android APIs with **ARMv8.2-FP16** instructions:

  android.util.Half.toFloat(short h)
  android.util.Half.toHalf(float f)
  android.util.Half.floor()
  android.util.Half.ceil()
  android.util.Half.round()

  ```java
  // Original Java implementation
  public static float toFloat(short h) {
    int bits = h & 0xffff;
    int s = bits & SIGN_MASK;
    int e = (bits >>> EXPONENT_SHIFT) & SHIFTED_EXPONENT_MASK;
    int m = (bits ) & SIGNIFICAND_MASK;
    int outE = 0;
    int outM = 0;
    if (e == 0) {
      if (m != 0) {
        float o = Float.intBitsToFloat(FP32_DENORMAL_MAGIC + m);
        o -= FP32_DENORMAL_FLOAT;
        return s == 0 ? o : -o;}} else {
    outM = m << 13;
    if (e == 0x1f) {
      outE = 0xff;
      if (outM != 0) {outM |= FP32_QNAN_MASK;}} else {outE = e - EXPONENT_BIAS + FP32_EXPONENT_BIAS;}
  }
  int out = (s << 16) | (outE << FP32_EXPONENT_SHIFT) | outM;
  return Float.intBitsToFloat(out);
  }
  ```
Enable Armv8.x Features in ART

● Optimise Android APIs with **ARMv8.2-FP16** instructions:
  
  ```java
  android.util.Half.toFloat(short h)
  android.util.Half.toHalf(float f)
  android.util.Half.floor()
  android.util.Half.ceil()
  android.util.Half.round()
  ...
  
  ;; inlined by ART as intrinsic:
  FMOV H31, W1 ; W1 == h
  FCVT S0, H31 ; return S0
  ```

● 9 Android Half APIs are optimised with ARMv8.2-FP16 instructions
  ○ for more details:
  ○ [https://android-review.googlesource.com/q/owner:usama.arif@linaro.org](https://android-review.googlesource.com/q/owner:usama.arif@linaro.org)
Enable Armv8.x Features in ART

- Optimise Java dot product loop with **ARMv8.2-UDOT/SDOT**

```plaintext
// byte[] a; byte[] b;
for (int i = 0; i < size; i++) {
    sum += a[i] * b[i];
}
```

- Implemented in ART auto vectorization.
- Armv8.2 dotprod CPU feature detection and testing also implemented.
- For more details:
  
  https://android-review.googlesource.com/c/platform/art/+/1352360
Enable Armv8.x Features in ART

- Optimise Java loops with ARM SVE
  - Most exciting feature this year.
  - Get ART prepared for future Android platforms with ARM SVE features.
  - No SVE real hardware for Android yet (in 2020)
    - functionality/optimisation development based on VIXL simulator.

- Current ART SVE patches:
  - CPU feature detection in ART (DONE)
  - SVE codegen support in ART (DONE)
  - SVE predicated vector instructions support in ART (DONE)
  - Update AOSP/VIXL with SVE support (UPSTREAM)
  - [https://android-review.googlesource.com/q/owner:artem.serov@linaro.org+sve](https://android-review.googlesource.com/q/owner:artem.serov@linaro.org+sve)
Enable Armv8.x Features in ART

- What SVE optimisation looks like in ART:
  - Optimised by auto vectorization in ART compiler.

```java
private void DiamondLoop(int[] a){
    for (int i = 0; i < LENGTH; i++){
        if (a[i] % 2 == 0) {
            a[i] = a[i] + 1;
        }
    }
}
```

Thanks to SVE's predicated execution!

```
__ Mov(w0, 0);
__ Dup(z0.VnS(), 1);
__ Bind(&label_loop);
__ Whilelt(p0.VnS(), w0, w2);
__ B(eq, &exit);

__ Add(x3, x1, kJavaArrayDataOffset);
__ Ld1w(z1.VnS(), p0.Zeroing(), SVEMemOperand(x3, x0, LSL, 2));
__ And(z2.VnS(), p0.Merging(), z1.VnS(), z0.VnS());
__ Cmpeq(p1.VnS(), p0.Zeroing(), z2.VnS(), z0.VnS());
__ Add(z1.VnS(), p1.Merging(), z1.VnS(), z0.VnS());
__ St1w(z1.VnS(), p1, SVEMemOperand(x3, x0, LSL, 2));
__ Incw(w0);
__ B(&label_loop);
__ Bind(&exit);
__ Mov(w0, 0);
__ Ret();
```
ART Development & Testing Improvements

● Performance testing & analysis improvements
  ○ Profiling ART benchmarks with simpleperf
    ■ https://android-review.linaro.org/c/linaro-art/art-build-scripts/+/21222
  ○ The perf2cfg tool
    ■ annotates CFG file with profiling information from simpleperf data files
    ■ shows method/block/instruction level perf profile in c1visualizer
    ■ https://android-review.googlesource.com/q/owner:emma.lagier@linaro.org

● Armv8.x feature testing improvements:
  ○ ART checker test improvements: introduced conditional statements support
    ■ Enhanced ART testing optional Armv8.x features
    ■ E.g. Dotprod, SVE instructions.
  ○ https://android-review.googlesource.com/q/owner:fabio.rinaldi@linaro.org
ART Development & Testing Improvements

- **VIXL simulator**
  - Part of VIXL project.
  - Support all instructions emitted by the VIXL assembler.
  - Pure software emulation.
  - Improves testing & debugging new ARMv8.x ISA features, algorithm development in ART.

- **Standalone VIXL simulator**
  - Try SVE on VIXL simulator now:

- **ART integration patch upstreamed:**
  - [https://android-review.googlesource.com/c/platform/art/+/1177334](https://android-review.googlesource.com/c/platform/art/+/1177334)
Summary

● We are a team of very happy & active ARTists
  ○ https://android-review.googlesource.com/q/author:*linaro.org+project:platform/art

● Our focus in 2020:
  ○ ART performance and code size improvements
  ○ Enabling ARM new architecture features in ART (e.g. SVE)
  ○ ART development & testing improvements

● Suggestions/feedback/new assignees to LCG ART team are welcome!
Thank you

Accelerating deployment in the Arm Ecosystem