Making ARM the Best Platform for Android

Xue Liang Zhong
Linaro LMG ART Team
Agenda

● Who we are
● What we’ve contributed to ART
● Develop high quality patches upstream
● Q & A
Who are we

- Subset of Linaro LMG
- 7 engineers @ ARM (Cambridge)
- 2 engineers @ Spreadtrum (Beijing & Tianjin)
- 1 engineer @ Mediatek (Hsinchu)
What we’ve contributed to ART

- VIXL32
- ARM optimizations to ART optimizing compiler
- Tooling improvements
VIXL32

- What’s VIXL?
  - Programmatic assembler & disassembler: [https://git.linaro.org/arm/vixl.git/about/](https://git.linaro.org/arm/vixl.git/about/)
  - Adopted by ART compiler for its ARM backends: ARM64 and ARM32.
  - Offers simple assembler syntax.

<table>
<thead>
<tr>
<th>ARM assembly written in VIXL</th>
<th>Actual code generated</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__ Add(r1, r0, 0x12345678);</code></td>
<td><code>mov r1, #22136</code></td>
</tr>
<tr>
<td></td>
<td><code>movt r1, #4660</code></td>
</tr>
<tr>
<td></td>
<td><code>add r1, r0, r1</code></td>
</tr>
</tbody>
</table>

- Offers extra features for easier & safer coding:
  - Veneers for branch
  - Literal pools for constant values

- VIXL32 offers complete ARMv8 Aarch32 ISA support
  - Full NEON support (which the old arm backends lacks of)
  - New 32-bit instructions: floating-point rounding, conversion, etc.

- VIXL32 is now the default backend for ART ARM32
Instruction scheduler for ARM Backends

- Aim to help little core based ARM platforms:
  - Cortex-A53/A32/A7/etc.

- Avoids execution delays on ARM CPUs by:
  - Building scheduling graph
  - Reordering instructions aggressively
  - Hiding execution dependencies

- Performance improvements:
  - Math benchmark: over 10% performance gain
  - 2% to 4% improvements for various workloads:
    - algorithm, benchmarkgame, caffeinemark, and stanford

- [https://android-review.googlesource.com/#/c/201950/](https://android-review.googlesource.com/#/c/201950/)
Various improvements to ART

- **Intrinsics**
  - Thanks to VIXL32, more and more library calls (e.g. java.Math) are optimized using advanced ARM instructions (e.g. vrintn, vrintp, vrintm, etc.)

- **Instruction simplifier**
  - ARM specific simplifier optimizations: ShiftOperands, MultiplyAccumulate, ArrayAccess, etc.

- **32-bit specific performance analysis & optimizations:**
  - Including: intrinsics, simplification, improved code generation, scheduler, etc.

- **Actively introducing & improving tests for ART:**
  - gtests/checker tests
  - Valgrind tests for Android devices: fixed memory leaks in ART
  - DexFuzzer tests
Tooling improvements for ART ARM backends

- Introducing JIT code profiling for ART.
  - Works with standard PERF tool with 4.1+ kernel
  - ART JIT mode now supports generating: /data/misc/trace/jit-PID.jump
  - ‘perf-inject’ processed perf.data file
    - Perf report example
      ```bash
      $ perf record -k mono dalvikvm -Xcompiler-option -g <classpath> Test
      $ perf inject -i perf.data -o perf.data.jitted
      $ perf annotate -i perf.data.jitted
      ```
      ```bash
      5067356647016 : PERF_RECORD_MMAP2 [0x7fb70d20000(0x5e7000)]: /system/lib64/libart.so
      5068487571183 : PERF_RECORD_MMAP2 [0x7fad3ddd30(0x118)]: /data/misc/trace/jitted-16753-10.so
      5067368247016 : PERF_RECORD_SAMPLE(IP, 0x2):0x7fad3ddd40
      ```
  - Command workflow example
    ```bash
    $ perf record -k mono dalvikvm -Xcompiler-option -g <classpath> Test
    $ perf inject -i perf.data -o perf.data.jitted
    $ perf annotate -i perf.data.jitted
    ```
    ```bash
    5067356647016 : PERF_RECORD_MMAP2 [0x7fb70d20000(0x5e7000)]: /system/lib64/libart.so
    5068487571183 : PERF_RECORD_MMAP2 [0x7fad3ddd30(0x118)]: /data/misc/trace/jitted-16753-10.so
    5067368247016 : PERF_RECORD_SAMPLE(IP, 0x2):0x7fad3ddd40
    ```

- Perf report example

<table>
<thead>
<tr>
<th>Percent</th>
<th>Source code &amp; Disassembly of jitted-16753-10.so for cpu-clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.24</td>
<td>7c: sub w6, w5, #0x1</td>
</tr>
<tr>
<td>9.33</td>
<td>80: add w7, w2, #0xc</td>
</tr>
<tr>
<td>19.73</td>
<td>84: ldr w8, [x7,x6,lsl #2]</td>
</tr>
<tr>
<td>19.30</td>
<td>88: cmp w8, w4</td>
</tr>
<tr>
<td>0.00</td>
<td>8c: b.le a4 &lt;void Sort.insertion(int[])+0x64&gt;</td>
</tr>
</tbody>
</table>
Tooling improvements for ART ARM backends

- Works even better with ARM DS5 studio.
  - ART JIT code analysis with DS5/Streamline
Tooling improvements for ART ARM backends

- Works even better with ARM DS5 studio.
  - ART AOT code analysis with DS5/Streamline
Develop high quality patches upstream

- Linaro internal gerrit & Linaro CI integration
- ART performance benchmarking and reports
- Working on Linaro Android tree
Linaro internal gerrit & CI integration

- High coding standards for all patches

- Gerrit code review integrated with Linaro CI.
  - Trigger tests easily from Gerrit.
    - Host tests
    - Target tests
    - ART Benchmarking tests
    - VIXL Tests
  - Workflow example -->
  - Great THANKS to
    - Linaro automation team!
ART performance benchmarking and reports

- **art-testing benchmark suite**
  - Includes different kinds benchmarks to represent real-life workloads.
    - Algorithms (CryptoMD5, DeltaBlue, Dhrystone, Richards, etc)
    - Caffeinemark
    - Math (NBody, Cordic, SpectralNorm, etc)
    - and more ...

- **art-reports.linaro.org:**
  - Keeping track of improvements & regressions of each patch

A significant performance change captured by art-reports in FEB 2017, showing that *instruction scheduling* merged upstream.
Working with Linaro Android tree

- **ART Build Scripts**
  - Maintained by Linaro ART team
  - Abstract frequent changes to Android build system
  - Replicate how Upstream and Linaro CI tests
  - “ANDROID ROOT”
    - Offers chroot-like side install of binaries to /data/local/tmp
    - Mitigates broken Tip

```
$ scripts/tests/test_art_host.sh
...
...
INFO: scripts/tests/test_art_host.sh Finished!
INFO: Host Test took: 0h 55m 45s
SUCCESS: Test Summary:
build_host TOOK: 0h 25m 11s PASSED
gtest TOOK: 0h 6m 43s PASSED
optimizing TOOK: 0h 7m 43s PASSED
interpreter TOOK: 0h 8m 21s PASSED
jit TOOK: 0h 7m 45s PASSED
```
Resources

- ARTNewStarter
  - https://wiki.linaro.org/Internal/AndroidART/Introduction

- LMG Reference Library
  - https://wiki.linaro.org/LMG/Engineering

- ART JIT in Android N
  - https://www.youtube.com/watch?v=TCJLFqhC1VE

- ART 32-bit Performance Analysis
  - https://www.youtube.com/watch?v=a3K_HrwWzXY&t=4340s

- VIXL: A Programmatic Assembler & Disassembler for AArch32
  - https://www.youtube.com/watch?v=a3K_HrwWzXY&t=3485s

- Linaro ART CI Wiki
  - https://wiki.linaro.org/Internal/LMG/ART-CI
Thank You

#BUD17

For further information: [www.linaro.org](http://www.linaro.org)

BUD17 keynotes and videos on: [connect.linaro.org](http://connect.linaro.org)