ART II

New Member Assimilation Guide
Who are we?

- LMG
  - Linaro Android Runtime Team (ART)
- 7 engineers @ ARM
- 2 engineers @ Spreadtrum
What do we do?

- Linaro ART Tip (Upstream)
  - Focus on improving ART for the **next** Android release

- Linaro ART Stable (Linaro)
  - Focus on improving ART for the **current** Android release
Development Cycle

- Task Selection
- Development
- Internal Code Review
- Upstream Code Review
- Testing
- Benchmarking
Development Cycle

- Jira
- C++, ASM, Java
- Internal Gerrit
- Upstream Gerrit
- Jenkins
- Jenkins & Lava
Task Selection
Task Selection

- Jira component
  - ART-Tip, ART-Stable within the LMG project

- Jira labels
  - For classifying the ART subsystems

- Fix version
  - For identifying our current focus

- Kanban
  - With daily scrum meetings (Scrumban)
  - Works well when working on fast-paced upstream projects!
<table>
<thead>
<tr>
<th>Todo</th>
<th>In Development</th>
<th>Review</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMG-1051 Investigate that irreducible loops work.</td>
<td>LMG-742 Implement a SlowPath sharing mechanism.</td>
<td>LMG-749 Extend the intrinsic list.</td>
<td>LMG-749 Extend the intrinsic list.</td>
</tr>
<tr>
<td>None</td>
<td>ARTAndroidN, ARTOptimisations</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>LMG-1135 Investigate dex_cache_arrays and pc_relative from the sight of irreducible loops</td>
<td>LMG-1056 Implement a mechanism for compressing Stack Maps containing Inline Info.</td>
<td>LMG-1010 Implement ARM64 support for SystemArrayCopy().</td>
<td>None</td>
</tr>
<tr>
<td>None</td>
<td>ARTAndroidN, ARTOptimisations</td>
<td>None</td>
<td>ARTAndroidN, ARTOptimisations</td>
</tr>
<tr>
<td>LMG-1036 Investigate Aarch64 Assembler Interrupts.</td>
<td>LMG-1050 Optimize String Intrinsics.</td>
<td>LMG-988 Linker support for multi-arch library paths</td>
<td>None</td>
</tr>
<tr>
<td>None</td>
<td>ARTAndroidN, ARTInvestigation, ARTRuntime</td>
<td>None</td>
<td>ARTAndroidN, ARTOptimisations</td>
</tr>
<tr>
<td>LMG-1089 Benchmark arm64 fast interpreter.</td>
<td>LMG-1047 Investigate Multi-image issues</td>
<td>LMG-1049 Implement ARM64 Integer/LongBitCount</td>
<td>None</td>
</tr>
<tr>
<td>None</td>
<td>ART Optimizing Compiler Improvements.</td>
<td>None</td>
<td>ARTAndroidN, ARTOptimisations</td>
</tr>
<tr>
<td>LMG-749 Extend the intrinsic list.</td>
<td>ART Optimizing Compiler Improvements.</td>
<td>LMG-989Negate bitwise operations simplification</td>
<td>None</td>
</tr>
<tr>
<td>None</td>
<td>ARTAndroidN, ARTRuntime</td>
<td>None</td>
<td>ARTAndroidN, ARTOptimisations</td>
</tr>
</tbody>
</table>

**Development Cycle**

**Increasing Priority**

**Kanban**

**Linaro Connect**

**Bangkok 2016**
## Kanban

<table>
<thead>
<tr>
<th>To Do</th>
<th>In Development</th>
<th>Review</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 of 106</td>
<td>5 of 22</td>
<td>5 of 21</td>
<td>3 of 34</td>
</tr>
</tbody>
</table>

### Development - 17 issues

<table>
<thead>
<tr>
<th>Issue</th>
<th>Description</th>
<th>Assigned</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMG-1051</td>
<td>Investigate that irreducible loops work ...</td>
<td>None</td>
<td>ARTAndroidN, ARTOptimisations</td>
</tr>
<tr>
<td>LMG-1135</td>
<td>Investigate <code>dex_cache_arrays</code> and <code>pc_relative</code> from the sight of Irreducible loops</td>
<td>None</td>
<td>ARTAndroidN, ARTOptimisations</td>
</tr>
<tr>
<td>LMG-1036</td>
<td>Investigate Arm64 Assembler Interpreter</td>
<td>None</td>
<td>ART, ARTAndroidN, ARTInvestigation</td>
</tr>
<tr>
<td>LMG-1089</td>
<td>Benchmark arm64 fast interpreter</td>
<td>None</td>
<td>ARTAndroidN, ARTOptimisations</td>
</tr>
<tr>
<td>LMG-749</td>
<td>Extend the intrinsics list</td>
<td>None</td>
<td>ARTAndroidN, ARTOptimisations</td>
</tr>
<tr>
<td>LMG-1064</td>
<td>Add support for AArch32 FP intrinsics</td>
<td>None</td>
<td>ARTAndroidN, ARTOptimisations</td>
</tr>
<tr>
<td>LMG-1050</td>
<td>Optimize String Intrinsics</td>
<td>None</td>
<td>ARTAndroidN, ARTInvestigation, ARTOptimisations</td>
</tr>
<tr>
<td>LMG-1047</td>
<td>Investigate Multi-image issues</td>
<td>None</td>
<td>ARTAndroidN, ARTRuntime</td>
</tr>
<tr>
<td>LMG-1056</td>
<td>Implement a mechanism for compressing Stack Maps containing Inline Info.</td>
<td>None</td>
<td>ARTAndroidN, ARTOptimisations</td>
</tr>
<tr>
<td>LMG-1010</td>
<td>Implement ARM64 support for SystemArrayListCopy</td>
<td>None</td>
<td>ARTAndroidN, ARTOptimisations</td>
</tr>
<tr>
<td>LMG-988</td>
<td>Linker support for multi-arch library paths</td>
<td>None</td>
<td>ARTAndroidN</td>
</tr>
<tr>
<td>LMG-1049</td>
<td>Implement ARM64 Integer/LongBitCount</td>
<td>None</td>
<td>ARTAndroidN, ARTOptimisations</td>
</tr>
<tr>
<td>LMG-989</td>
<td>Negated bitwise operations simplification</td>
<td>None</td>
<td>ARTAndroidN, ARTOptimisations</td>
</tr>
<tr>
<td>LMG-1065</td>
<td>Investigate CLANG build failure and propose fix</td>
<td>None</td>
<td>ARTAndroidN, ARTInvestigation</td>
</tr>
</tbody>
</table>

---

Linaro Connect
Bangkok 2016
Development
Development

- Take a look at Scott Wakeling’s BKK16-302
- **Develop**
  - Google’s coding style (take a look at the surrounding code)
  - Simple is better!
- **Test**
  - Gtests, Java tests, Checker tests
- **Benchmark**
  - ART-testing framework (write a new benchmark)
- Questions & guidance
  - #linaro-art IRC
Internal Code Review
Internal Code Review

- 2 Gerrit instances
  - **Public code review** (ART-Testing patches)
  - **Private code review** (ART patches)
- All ART patches ready for internal review
  - **Should pass:** `mma test-art-host && mma test-art-target`
- All ART patches ready for upstream review
  - +2 from Linaro ART Leads!
  - +1 from Linaro ART!
  - +1 from Linaro Automation?!
  - Have replies to all comments
Testing
Working with Upstream?
Working with Upstream?

Cambridge, UK
Wednesday 20:00
Light Rain Showers

Precipitation: 59%
Humidity: 84%
Wind: 15 mph

Bangkok, Thailand
Thursday 03:00
Clear with periodic clouds

Precipitation: 0%
Humidity: 65%
Wind: 3 mph
Testing

- Thanks Daniel, Milosz, Vishal and automation@linaro.org!
- Tests types
  - Run on every Gerrit patch submission
  - Run on every tree update (every 6hrs)
- Build monitors
  - Linaro ART Tip
  - Linaro ART Stable
- Every test
  - Has a short description
  - Stores useful logs and pinned-manifest.xml
Test: **ARTHostTest**

- **Description**
  - mma test-art-host
  - gtests, java tests, checker tests
  - Builds on host, runs on host
  - Generates [Gtests results](#)

- **Trigger**
  - Every patch submission
  - Every tree update

- **When should be run?**
  - On every patch submitted for review (automatically triggered)
Test: **ARTTargetTest**

- **Description**
  - `mma test-art-target` for ARM and ARM64
  - `gtests`, `java tests`, `checker tests`
  - Builds on host, runs on target
  - Tests run in a chroot like environment

- **Trigger**
  - Every patch submission
  - Every tree update

- **When should be run?**
  - On every patch submitted for review (automatically triggered)
Test: ARTBootImageMeasurements

● Description
  ○ Builds `boot.oat` using `dex2oat` for all the ART targets
  ○ Builds on host, runs on host
  ○ Generates compile time and size plots

● Trigger
  ○ Every tree update
  ○ Topic trigger
    
    repo upload art/ --dest=master/IncludeARTBootImageMeasurements

● When should be run?
  ○ Concerned about compile time and size
Test: **ARTCodeCoverage**

- **Description**
  - Builds `boot.oat` using `dex2oat` for all the ART targets
  - Runs the host `gtest`
  - Builds on host, runs on host
  - Generates code coverage reports

- **Trigger**
  - Every tree update
  - Topic trigger
    ```bash
    repo upload art/ --dest=master/IncludeARTCodeCoverage
    ```

- **When should be run?**
  - Concerned about code paths exercised
Test: ARTBootToGUI-aosp_arm64

- **Description**
  - Builds the aosp_arm64 target and runs it in the emulator
  - Builds on host, runs on host
  - Generates build and boot time plots

- **Trigger**
  - Every tree update
  - Topic trigger
    - `repo upload art/ --dest=master/IncludeBootToGUI`

- **When should be run?**
  - Concerned about booting to GUI
How to debug a test failure?

- Look at the test trigger
  - Tree update or patch trigger?

- Take a look at
  - Test status
  - Parsed console output
  - The build logs

- Run the tests locally
  - Grab the pinned-manifest.xml
  - Run the test scripts locally (WIP)
Example: **ARTTargetTest** Failure

- **Test status:** 145-alloc-tracking-stress failed
- **Trigger type:** Tree Update
- **Parsed console output**
  test-art-target-run-test-default failed (ARM/ARM64)
  test-art-target-run-test-optimizing failed (ARM/ARM64)
- **Build logs** (only need to take a look at default and optimizing logs)
  

  ######################## info
  # Regression test for b/18661622
  ######################## diffs
  --- expected.txt  2016-03-02 10:00:23.446375718 +0000
  +++ output.txt   2016-03-02 12:15:51.386021604 +0000
  @@ -1 +1 @@
  -Finishing
  +Aborted
  ########################
Example: **ARTTargetTest** Failure

- What next?
  - Grab the `pinned-manifest.xml`
  - Reproduce the failure locally!
  - Fix the failure (see more info in Scott Wakeling’s [BKK16-302](#))
Benchmarking
Benchmarking

- Thanks Daniel, Milosz, Vishal and automation@linaro.org!
- Jenkins & LAVA
  - Benchmarking on Nexus 9 (Nexus 5x soon)
- Runs on every patch submitted for review and every tree update
- ART Reports - sent by email
  - Every patch (posted to Gerrit), week, month
- ART Graphs
- Using ART-testing as a workload
  - Performance, compile time and size
ART Reports

<table>
<thead>
<tr>
<th>build</th>
<th>change</th>
<th>patchset</th>
</tr>
</thead>
<tbody>
<tr>
<td>#412 / linaro-art-tip-build-nexus9-MicroBenchmarks</td>
<td>518</td>
<td>10</td>
</tr>
<tr>
<td>#411 / linaro-art-tip-build-nexus9-MicroBenchmarks</td>
<td>562</td>
<td>6</td>
</tr>
<tr>
<td>#410 / linaro-art-tip-build-nexus9-MicroBenchmarks</td>
<td>563</td>
<td>7</td>
</tr>
</tbody>
</table>
ART Graphs

Branch: master
Benchmark: SystemArrayCopy

Benchmark results for branch: master

Wednesday, Mar 2, 03:43:13.865
- ArrayCopyCharBufferedReadSmall_32: 244 390.906
Benchmark: Nexus9-MicroBenchmarks

- **Description**
  - Jenkins builds a Nexus 9 image and sends it over to Lava
  - LAVA runs the benchmarks for ARM and ARM64
  - LAVA stores the results in the database and sends reports

- **Trigger**
  - Every patch submission
  - Every tree update

- **When should be run?**
  - On every patch submitted for review (automatically triggered)
Example: **Nexus9-MicroBenchmarks** Run

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSieve / NSieveAccess_32 (10)</td>
<td>7089996.35</td>
</tr>
<tr>
<td>threading / ThreadRing_32 (10)</td>
<td>930114790.40</td>
</tr>
<tr>
<td>revcomp / RevComp_32 (10)</td>
<td>9947498.50</td>
</tr>
<tr>
<td>HashMapBench / TestHashMap_32 (10)</td>
<td>7817846.05</td>
</tr>
<tr>
<td>MathPartialSums / MathPartialSums_32 (10)</td>
<td>13273360.70</td>
</tr>
<tr>
<td>StringEquals / Equals_32 (10)</td>
<td>75116.98</td>
</tr>
<tr>
<td>MathSpectralNorm / MathSpectralNorm_32 (10)</td>
<td>3501096.58</td>
</tr>
<tr>
<td>DeltaBlue / DeltaBlue_32 (10)</td>
<td>3634497.30</td>
</tr>
</tbody>
</table>

- **Gerrit Patch**
- **Jenkins build**
- **System Image**
- **ART Report**
Future Work on Testing & Benchmarking

- **Testing**
  - More stable tests!
  - Ability to run Jenkins scripts locally
  - More tests - Clang sanitizers, dex fuzzing, Valgrind tests, etc.

- **Benchmarking**
  - More devices - Nexus 5x, 6p, future devices
  - Compilation time and sizes for various apks
  - Benchmark profiles
Upstream Code Review
Upstream Code Review

- Always use two Android trees!
  - A Linaro tree - for internal review
  - An AOSP tree - for external review

- All patches submitted upstream should
  - Have a +2 in the internal review
  - Small fixes **do not need to be re-reviewed** internally
  - Reworks and new features **need to be re-reviewed** internally!
  - Have Linaro ART Leads added as reviewers upstream
Thanks!