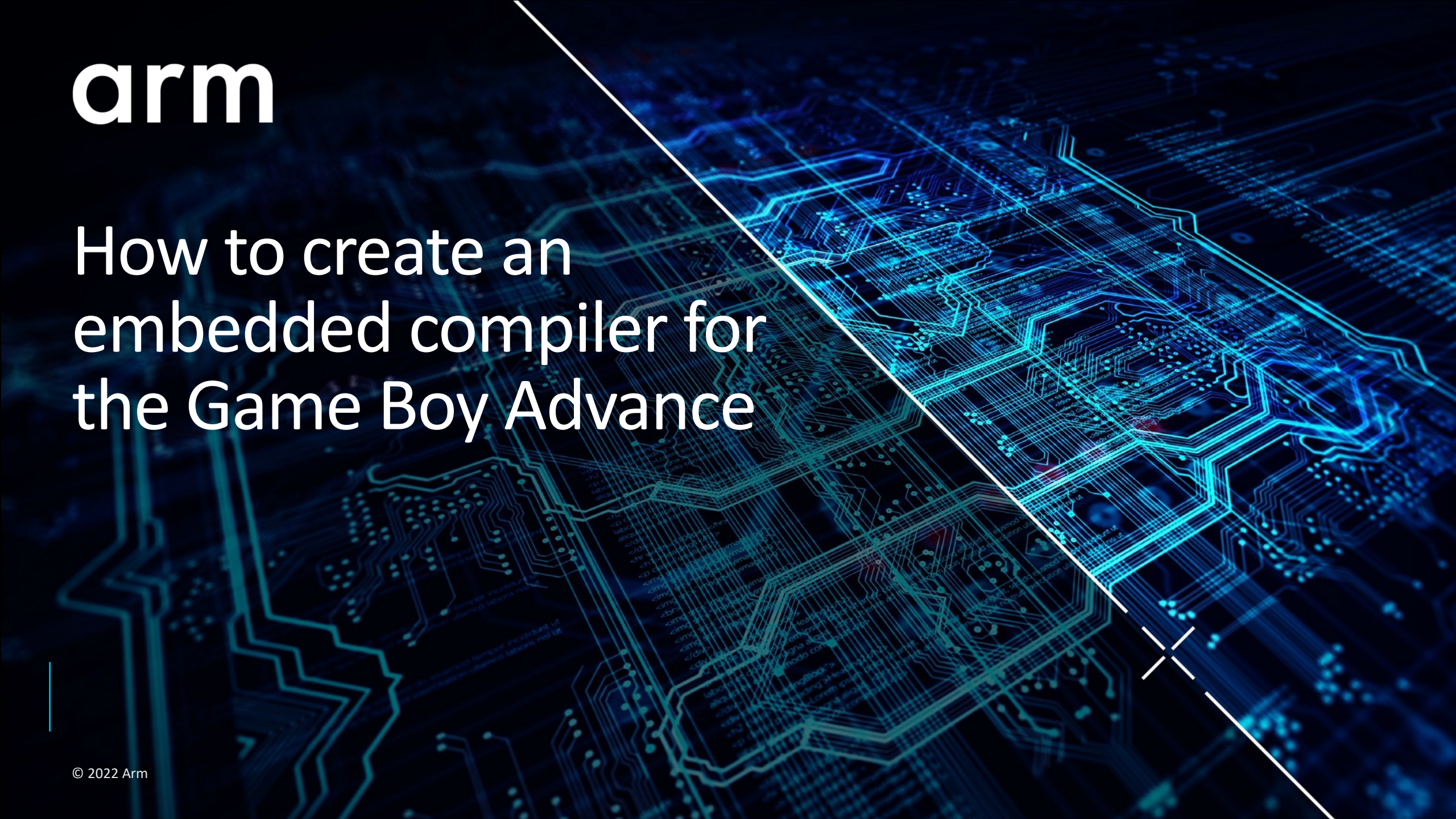


How to create an embedded compiler for the Game Boy Advance

Me

- Fairly obsessed with Game Boy Advance
- Made my first substantial program on it:
  - Bookreader: read tons of books on 240x160 screen
- Taught me assembly, made a toy compiler
- Got my first two jobs out of it
- All good things came from the GBA
- Got to work at Arm in LLVM Embedded team
- Wanted LLVM-based toolchain
- Has been tricky to find info

Talk about

- embedded compiler how
- State of the art
- Current LLVM embedded hot topics
- GCC toolchain compatibility issues
- What parts of LLVM needed/will need to be fixed up?

vehicle

- Game Boy Advance (GBA) toolchain
  - completely based on LLVM (no GCC)
  - Grafted on top of ‘LLVM Embedded Toolchain for Arm’ aka BMT + GBA essentials
  - Completely open source
  - https://github.com/stuij/gba-llvm-devkit
  - 1st release
- I worked on both the BMT and of course the GBA toolchain
- Some content relates to BMT, some to GBA toolchain,

Talk overview

- Short history of embedded LLVM
- Game Boy Advance (re)introduction
- embedded toolchain component overview
- Build setup
- Components: Compiler, LLD, etc..
- Example program
- Evaluation

Short history of LLVM embedded

- Traditionally LLVM effort has concentrated on hosted systems
- Up until 2020, mostly just Arm baremetal support, and not well maintained
- 2020 LLVM talk: LLVM in a Bare Metal Environment, Hafiz Abid Qadeer
- Now also support for RiscV, PowerPC, AArch64
- Community has grown, LLVM Embedded Toolchains Working group

Game Boy Advance overview

- Nintendo handheld, long awaited successor of Game Boy
- Released in March 2001, 82M units sold
- Graphics-wise Super Nintendo in your pocket, but totally different arch
- ARM7TDMI processor at 16.78 MHz, ARMv4T ISA, 32-bit ALU
- Arm + Thumb, 16bit instruction format
- made famous by Nokia 6110 (1998), 3210 (1999), 3310 (2000)
- Thought up by Dave Jagger, in trip to Japan in ;94
- 10 billion chips in total (200M in 2020)
- GBA details inline

GBA toolchain components

- LLVM compiler suite tools: clang/clang++, lld, lldb, etc..
- compiler-rt builtins
- LLVM libcxx: C++ library
- picolibc: embedded C library
- gba_cart.ld: linker script for GBA cart executables
- gba_crt0.s: startup code for the GBA
- libtonc: GBA library
- Apex Audio System: play music and sound effects on the GBA
- Grit: GBA graphics swiss army knife
- gbafix: fix up GBA headers

GBA internals

- Picture Processing Unit (PPU)
  - bitmap modes, tile modes
- Orig Game Boy hardware
  - No access, except for..
- Sound
  - Orig Game Boy Programmable Sound Chip (PSG)
    - Manipulate 4 waveforms, 4 channels
  - 2 channels to convert PGM format to DAC
- Link port with varous comm. Modes
- Bunch of memory we'll talk about later

Cmdline invocation

- grit bg.bmp -gB4 -Mw 8 -Mh 8 -pS -o bg -fa -pT1
- conv2aas AAS_Data
- clang --config armv4t-gba.cfg -Wl,-T,gba_cart.ld program.c bg.s AAS_Data.s -o program.elf
- llvm-objcopy -O binary program.elf program.gba
- gbafix program.gba

armv4t-gba.cfg

- --target=arm-none-eabi -mcpu=arm7tdmi
- -fno-exceptions
- -fno-rtti
- --sysroot <CFGDIR>/../lib/clang-runtimes/arm-none-eabi/armv4t
- -lcrt0-gba
- -D_LIBCPP_AVAILABILITY_HAS_NO_VERBOSE_ABORT

LLVM Components and layout

- Pure convention, following Hafiz Abid Qadeer
- bin/
  - compiler tools, gba tools
  - armv4t-gba.cfg
- lib/clang-runtimes/
  - multilib.yaml
  - arm-none-eabi/armv4t
    - lib/
      - crt0.o, Gba_cart.ld
      - libraries: compiler-rt, c libraries, C++ libraries, GBA libraries
    - include/
      - headers

Dependency graph

- GBA tools + config files
- compiler tools
  - c library (headers)
    - compiler-rt builtins
    - libunwind (headers)
      - C++abi
    - libcxx
      - Libcxxabi (also libunwind dep)
        - GBA libraries

Build setup

- Everything built from scratch, no GCC components
- GBA toolchain built on top of LLVM Embedded Toolchain for Arm
  - CMake, very composable
  - Piggyback on everything: CPACK, testing infra
- Theme: juggling to set the right lib and include between host and target
  - Create subproject for what needs to be built by ready toolchain
  - Don’t leak say CPPFLAGS
- -DCMAKE_SYSTEM_NAME=Generic
- When building for other arches, use LLVM Embedded Toolchain for Arm as example
- Loop through library variants

Configuration themes

- Set target\n
- Specify building for embedded
  - -DLIBCXX_ENABLE_STATIC=ON
  - CMAKE_SYSTEM_NAME=Generic
  - * -D*_BAREMETAL*=ON
- Feature options
- Where are tools, config files
- Where is target sysroot
- Test/build configuration

```
cmake ${CT_DIR} \
  -GNinja \
  -DCMAKE_TRY_COMPILE_TARGET_TYPE=STATIC_LIBRARY \
  -DCOMPILER_RT_BUILD_BUILTINS=ON \
  -DCOMPILER_RT_BUILD_SANITIZERS=OFF \
  -DCOMPILER_RT_BUILD_XRAY=OFF \
  -DCOMPILER_RT_BUILD_LIBFUZZER=OFF \
  -DCOMPILER_RT_BUILD_PROFILE=OFF \
  -DCMAKE_C_COMPILER_TARGET="armv4t-none-eabi" \
  -DCMAKE_ASM_COMPILER_TARGET="armv4t-none-eabi" \
```

- -DCMAKE_C_COMPILER=${BIN}/clang \
- -DCMAKE_AR=${BIN}/llvm-ar \
- -DCMAKE_NM=${BIN}/llvm-nm \
- -DCMAKE_RANLIB=${BIN}/llvm-ranlib \
- -DCOMPILER_RT_BAREMETAL_BUILD=ON \
- -DCOMPILER_RT_DEFAULT_TARGET_ONLY=ON \
- -DLLVM_CONFIG_PATH=${BIN}/llvm-config \
- -DCOMPILER_RT_INCLUDE_TESTS=OFF

- For c++;c++abi;libunwind we use the runtimes CMakelists.txt file
  - -DLIBCXX_ENABLE_MONOTONIC_CLOCK=OFF
  - -DLIBCXX_ENABLE_THREADS=OFF
  - -DLIBCXX_ENABLE_RANDOM_DEVICE=OFF
  - -DLIBCXX_ENABLE_RTTI=OFF
  - -DLIBCXX_ENABLE_WIDE_CHARACTERS=OFF
  - -DLIBUNWIND_ENABLE_THREADS=OFF
  - -DCMAKE_SYSTEM_NAME=Generic
  - -DCMAKE_TRY_COMPILE_TARGET_TYPE=STATIC_LIBRARY
  - -DLIBC_LINKER_SCRIPT=picolibcpp.ld
  - -DLIBCXXABI_BAREMETAL=ON

(cont..)

- -DLIBCXXABI_ENABLE_SHARED=OFF
- -DLIBCXXABI_ENABLE_ASSERTIONS=OFF
- -DLIBCXXABI_ENABLE_STATIC=ON
- -DLIBCXXABI_LIBCXX_INCLUDES="${INSTALL_DIR}/include/c++/v1"
- -DLIBCXXABI_USE_COMPILER_RT=ON
- -DLIBCXXABI_USE_LLVM_UNWINDER=ON

- Check talk by Hafiz Abid Qadeer for some more details and test setup

Individual components

- What to look out for
- GBA issues
- Hot topics

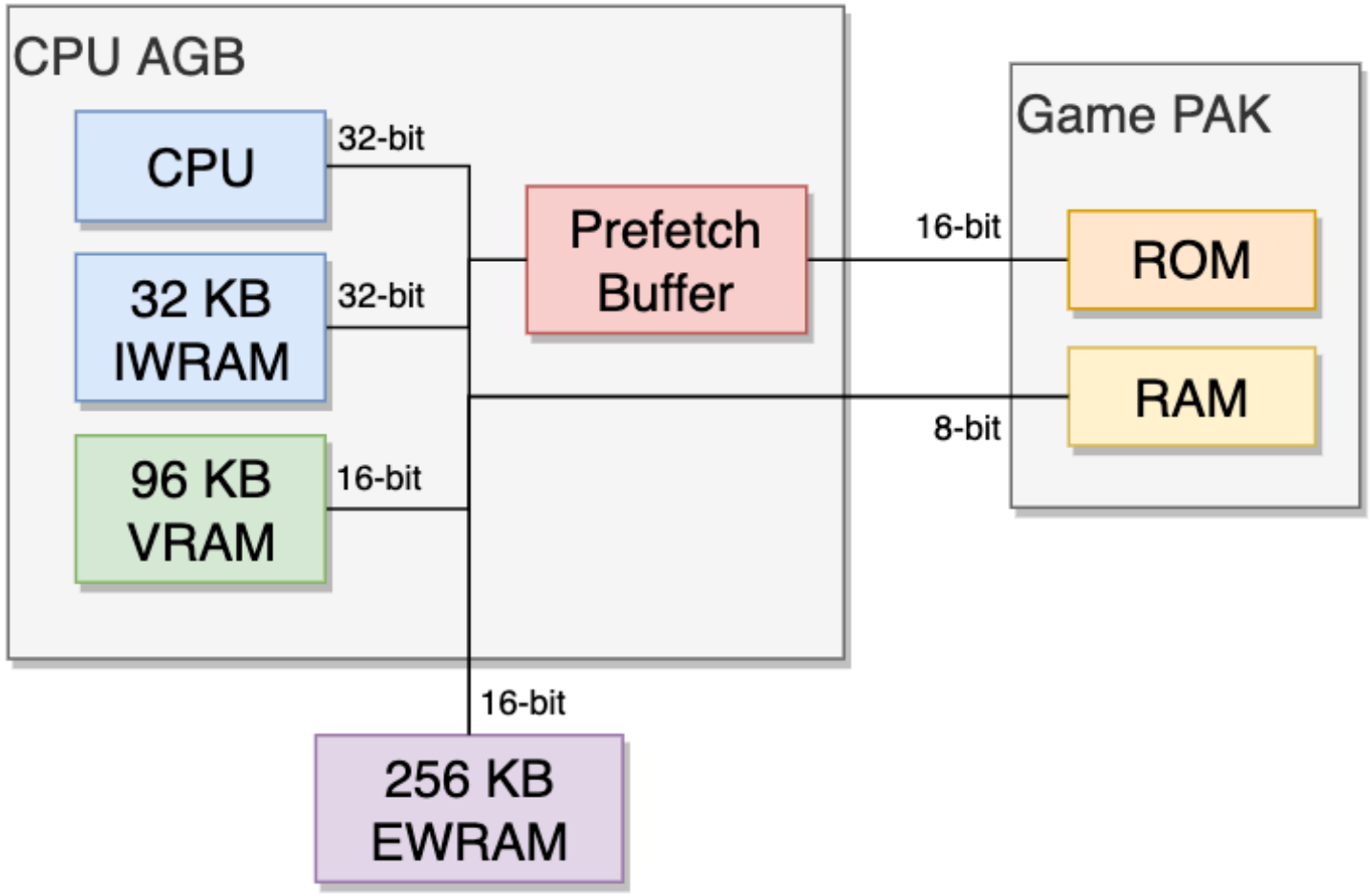
Clang

- Currently ARMv4T is the lowest supported Arm architecture in LLVM
- In fact for Arm, Clang defaults to ARMv4T and the ARM7TDMI is default CPU
- Bare metal file:
  - clang/lib/Driver/ToolChains/BareMetal.cpp
  - Adds things like –nostdsysteminc and handle multilib
- Old syntax

Multilib

- Solving issues:
  - need to build lots of libraries for variants.
  - Libraries don’t need to map to target triples.
- hardwiring library paths in C++ code not practical
- Can make build systems quite complex
- GCC has spec files
- Solution: more flexible option based on:
  - normalizing cmdline arguments
  - yaml files containing rules
  - Sets –sysroot –isystem -L
- Currently not working for GBA. Tiny bug when building for thumb. Can be fixed with matcher rule.

GBA memory layout (linker prepare)



Linker script / startup code

- Define memory regions (see previous slide)
- Copy data from rom to ram
- Zero initialize .bss
- Call routines from .init_array
- Call main

LLD / linker script / startup code

- For ARMv4T: Long branches and Arm/Thumb interworking was using BLX, which isn't supported on ARMv4(T). Also no MOVW/MOVT. So new thunks needed to cover all bases.
- For GBA:
  - Using DevkitPro linker script
  - Mostly worked as intended
  - LLD didn't accept overlay syntax
  - Catch C++ data sections
  - Clib incompatibilities
  - Add picolibc stubs

Linker improvements

- LTO
  - Respect code regions
  - still bunch of issues
    - code size can worsen
  - Todd Snider: goal is to implement API ... to honor linker script instructions.
- Code packing/distribute code over memory locations
- Compression
- debuggability
- place section at specific address: sys reg or io port
- overlays

Compiler-rt

- rt stands for runtime support
- builtins: low-level target-specific hooks. For Arm these are specified in the Arm ABI.
- libgcc drop-in replacement

- Also sanitizers, profiling, fuzzing, xray, scudo, BlocksRuntime, etc..
- Currently anything but builtins is turned off.
  - You get ubsan for free. It’s cheap.

- For ARMv4(T) there was no builtins target. Easy to fix by rearranging files so right subset would target.
- Demo time

clib

- Lots of choice: Newlib(-nano), Picolibc, LLVM Libc, etc..
- Choice for picolibc
  - BSD license
  - Newlib doesn’t have LLVM config
  - Responsive maintainers
  - More suitable for embedded development
- LLVM Libc
  - looks promising, but needs work done
  - Has seen embedded build success
  - Could scale from big iron to embedded. Talks underway.
  - Hardware Abstraction Layer
- For GBA development, can be convenient, but often inefficient.

CPP lib

- LLVM libcxx: Talks to make it more efficient for embedded.
- Butano, a popular GBA C++ engine, uses Embedded Template Library (ETL), optimized for embedded applications.
- Distributed as source

LLDB

- Debugger works in combination with mGBA emulator
- Also works from VSCode
  - But use CodeLLDB VSCode extension, not gba toolchain liblldb
  - Configure to attach to mGBA debugger stub
  - Currently there's a glitch when interrupting, but there's a fix out for that
- Shout-out to David Spickett for fixing some issues to get this to work.

GBA tools

- Grit
- Apex Audio System
- gbafix
- GBFS
- Maxmod
- Posprintf

Simple example

- Let’s paint the screen red

```
int main() {
  draw_screen();
  while(1) {};
  return 1;
}
```

Example continued

```
void draw_screen() {
  // set background mode 3 (bitmap) and turn on background 2
  *(unsigned long*) 0x4000000 = 0x403;

  unsigned short* vram = (unsigned short*) 0x6000000;
  // write red to every pixel on the 240x160 screen
  for(int x = 0; x < 240; x++)
    for(int y = 0; y < 160; y++)
      vram[x + y * 240] = 31;
}
```

CoreMark scores

compile settings	gcc	clang
rom arm	0.377644	0.348422
rom thumb	0.520429	0.411079
rom thumb unroll	0.563485	0.481064
rom thumb jumpth		0.492934
rom thumb more		0.527294
iwram thumb	1.521745	1.191095
iwram arm	1.899094	1.735995
iwram arm (hw)	1.899087	1.735989
iwram arm jumpth		1.950470
iwram arm more		2.098647

Evaluation

- Legend:
  - lots = -enable-dfa-jump-thread, -inline-threshold, -unroll-threshold

- Comparable:

- Resources
- repos
  - GBA-llvm-devkit -> https://github.com/stuij/gba-llvm-devkit
  - LLVM Embedded Toolchain for Arm (BMT) -> https://github.com/ARM-software/LLVM-embedded-toolchain-for-Arm
  - Runtimes ->
- Documentation
  - Multilib: https://clang.llvm.org/docs/Multilib.html
  - compile bare-metal compiler-rt: https://llvm.org/docs/HowToCrossCompileBuiltinsOnArm.html
  - LLVM in a Baremetal Environment -> https://www.youtube.com/watch?v=D9vCJwwTKaw
    Hafiz Abid Qadeer
- interact
  - Discord: #embedded-toolchains
  - Discourse/Monthly meets: LLVM Embedded Toolchains Working group: https://discourse.llvm.org/t/llvm-embedded-toolchains-working-group-sync-up

- Dave Jaggar, in Japan Jan ’94 visiting Nintendo. While there on skiing trip sketched solution on the back of a train napkin
  - Nintendo wanted a 32-bit CPU, but the memory footprint was too big.
  - Sharp, Arm’s 3rd licencee made their 8080/z80 cpu and also made all cartridge roms
  - GBA would be follow up product

GBA overview continued

- Ram: 32 KB internal, 256 KB external, 96 KB VRAM
- Game Boy HW inside, but only programmatic access to GB sound chip
- No own sound chip, mixing in software, output PCM samples to DAC
- Graphics: Picture Processing Unit (PPU)
  - Bg tile modes, bg bitmap modes, mode-7-like scaling
- Cart sizes up till 32MB (64MB video carts came out with mem remapping trickery)