Open Source Compilers for RISC-V
Past, Present and Future

Jeremy Bennett
Tool Chain Components (GNU)

Source code
- Ada ✓
- C/C++ ✓
- Fortran ?
- OpenMP ?

GCC ✓

GAS ✓

Compiler libraries
- libgcc ✓
- libstdc++v3 ✓

Object code

libc/libm

GDB ✓

CGEN ✓

Binutils ✓

objdump

disassemble
target sim

RV32GC ✓
RV64GC ✓

Newlib ✓
Glibc ✓
Musl ?

Complete ✓
In progress ?
Not available ✗
Tool Chain Components (Clang/LLVM)

- **Source code**
  - Ada
  - C/C++
  - Fortran
  - OpenMP
  - Complete
  - In progress
  - Not available

- **Clang/LLVM**

- **LLVM int. asm**

- **Object code**

- **TableGen**

- **Compiler libraries**
  - Newlib
  - BSD libc
  - Musl

- **libc/libm**

- **binutils**

- **llvm-objdump**

- **LLDB**

### Compilation Process
1. **Source code**
2. **Clang/LLVM**
3. **LLVM int. asm**
4. **Object code**
5. **Compiler libraries**
6. **libc/libm**
7. **binutils**
8. **llvm-objdump**
9. **LLDB**

### Compatibility
- **RV32GC** ✔
- **RV64GC** ✔

### Notes
- CompilerRT libc++ ✔?
- BSD libc ✔?
- Musl ✔?
- Ada ✔?
- C/C++ ✔
- Fortran ✔?
- OpenMP ✔?
- Complete ✔
- In progress ✔
Future Developments

- LLVM
  - Ada, Flang (?), D, Rust
  - machine inliner and shrinkwrap optimizations
  - bit manipulation and vector ISA extension support
  - decent CompilerRT implementation
- GCC
  - bit manipulation and vector ISA extension support
- Both compilers
  - `-march consistency!!!`
  - combined elimination: OpenTuner
  - fast emulation libraries
Deep Dive
Compiling for RV32 IoT Class Devices
Today’s Size Optimization Techniques

Embench data

- Os save-restore
- Os
- O1
- O2
- O3
- O3 unroll/inline

Small is good
Today’s Size Optimization Techniques

Embench data

- **-O3** unroll/inline
- **-Os** save-restore
- **-Oz**
- **-Oz** save-restore
- **-O1**
- **-O2**
- **-O3**

 GCC size
 LLVM size

Small is good
Today’s Size Optimization Techniques

Embench data

- Oz save-restore
- Oz
- Os save-restore
- Os
- O1
- O2
- O3
- O3 unroll/inline

GCC size
LLVM size

Small is good
Today’s Speed Optimization Techniques

Embench data

- O3 unroll/inline
- O3
- O2
- O1
- Os
- Os save-restore

GCC speed

Large is good
Today’s Speed Optimization Techniques

![Bar chart showing Embench data for different optimization levels: -Oz save-restore, -Oz, -Os save-restore, -Os, -O1, -O2, -O3, and -O3 unroll/inline. The chart compares GCC speed (red) and LLVM speed (yellow). The chart highlights that larger optimization levels generally result in better performance.](image-url)
Today’s Speed Optimization Techniques

Embench data

- -Oz save-restore
- -Oz
- -Os save-restore
- -Os
- -O1
- -O2
- -O3
- -O3 unroll/inline

GCC speed
LLVM speed

Large is good
Future Size Optimization Techniques

Embench data

- Os save-restore
RV32IMCB -Os save-restore
-Os combined elimination
-O2
RV32IMCB -O2
-O2 combined elimination for -Os

GCC size

Small is good
Future Size Optimization Techniques

Embench data

- Os save-restore
- RV32IMCB -Os save-restore
- Os combined elimination
- -O2
- RV32IMCB -O2
- -O2 combined elimination for -Os

GCC size
LLVM size

Small is good
Future Size Optimization Techniques

Embench data

- Os save-restore
- Os combined elimination
- O2
- RV32IMCB -O2
- O2 combined elimination for -Os

Small is good
The Detail Matters: Bit Manipulation

Code size with gcc -Os -msave-restore

RV32IMC
RV32IMBC
The Detail Matters: Bit Manipulation

Code size with gcc -Os -msave-restore

![Bar chart showing code size comparison between RV32IMC and RV32IMBC for various benchmarks.](chart.png)
How are we progressing?

Embench GCC code size

- Arm Cortex M4
- RI5CY RV32IMC
How are we progressing?

Embench GCC code size

- Arm Cortex M4
- RI5CY RV32IMC
Thank You

www.embecosm.com

Jeremy Bennett
jeremy.bennett@embecosm.com