Code Size Optimization

Bin Cheng (bin.cheng@arm.com)

Zhenqiang Chen (zhenqiang.chen@linaro.org)

Agenda

- Background
- Reduce spilling
- Code layout optimization
- Other opportunities
- Library optimizations

Background

- More and more MCU/embedded application developments base on GNU tools.
- Code size optimization is not as active as performance optimization in gcc.
- ARM Shanghai toolchain team focus on GNU Tools for ARM Embedded Processors
  - https://launchpad.net/gcc-arm-embedded

Key factors for code size

- Spilling
  - Additional load/store
- Code layout
  - Additional jump/long jump
- Misc others

Reduce spilling

- Register pressure directed optimizations
  - Hoist
    - In trunk, improved 0.1-0.2% for ARM/MIPS.
  - Loop2_invariant
    - -fno-move-loop-invariants.
    - In process. Expected to reduce > 0.2% for ARM/MIPS/PPC/X86

Register pressure changes

```
    A = …
    B = …
L:
    … = A
    … = B
    I = invariant
    … = I
```

```
    A = …
    B = …

L:
    … = A
    I = B
    … = I
```

```
    A = …
    B = …
L:
    I = A op B
    … = I
```

➔

```
A = …
B = …
I = invariant
L:
    … = A
    … = B
    … = I
```

➔

```
A = …
B = …
I = B
L:
    … = A
    … = I
```

➔

```
A = …
B = …
I = A op B
L:
    … = I
```

Code layout optimization

- Options to optimize code layout include
  - Basic block reordering
  - Short-circuit
  - ifconvert

Basic block reordering

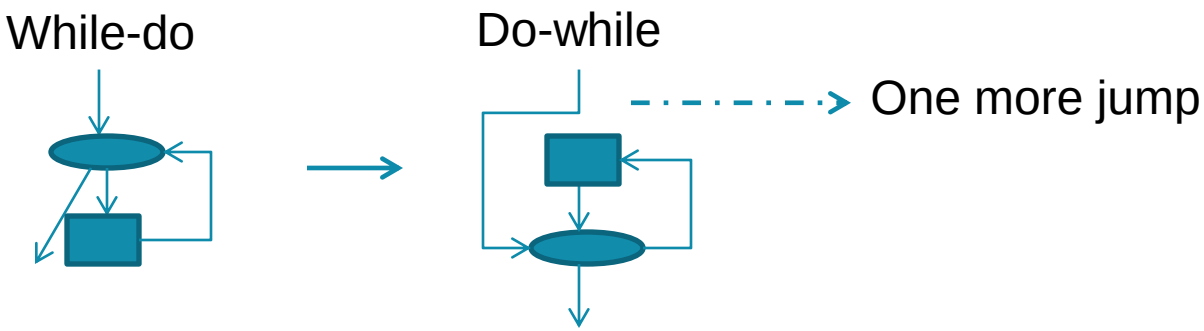
- Bbro
  - Disabled in 4.7 for –Os.
  - Implemented new heuristics for –Os (in trunk).
    - More fall through
    - Less long jump
    - Improved 0.2%-0.3% for ARM/MIPS/PPC/X86

Bbro examples

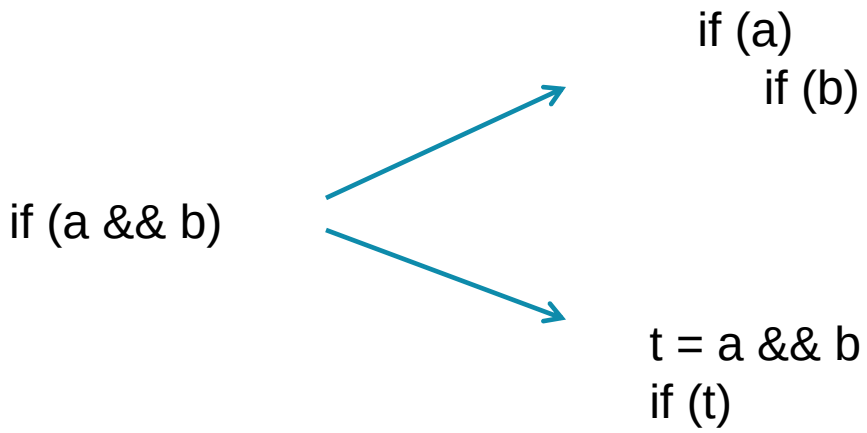
```
  A
 / \
B  C
 \ /
  D
```

For performance:
A -> B -> D … C or
A -> C -> D … B

For size:
A -> B -> C -> D or
A -> C -> B -> D



Short-circuit



Which one is better?

ifconvert

```
if (test) goto over; // x not
live
x = a;
goto label;
over:
...
```

➔ **find_if_case_1**

```
x = a;
if (!test) goto label;
over:
…
```

One jump can be saved

```
x = a;
if (test)
   x = b;
```

➔ PRE

```
if (test)
   x = b;
else
  x = a;
```

One more jump is needed

Other opportunities

- Tune inline parameters
  - MAX_INLINE_INSNS_SINGLE: 400
  - MAX_INLINE_INSNS_AUTO: 40
  - MAX_INLINE_INSNS_RECURSIVE: 450
  - MAX_INLINE_INSNS_RECURSIVE_AUTO: 450
- Cross-jump for code segments with only little difference.

Other opportunities (cont.)

- Optimizations after RA
  - Postreload
    - Add NOTES to indicate several loads from stack are from one spilled register.
  - Rename
  - Hard register propagation
    - Is “r4 = r3 - 0” a copy?
    - Inter-block

Other opportunities (cont.)

- RA heuristics
  - Re-materialization
  - Register copy
  - Cost model
- LRA is coming to replace reload.

Library optimizations

- Configure for size
  - E.g. --enable-target-optspace
- Simplify functionalities
- Link as needed

Summary

- Lots of opportunities for code size.

- Challenges
  - rtx_cost
  - RA

References

- Code size benchmark: CSiBE
  - http://www.inf.u-szeged.hu/csibe/
- EEMBC
  - http://www.eembc.org/
- C++ benchmarks
  - http://gcc.opensuse.org/c++bench/
- GNU Tools for ARM Embedded Processors
  - https://launchpad.net/gcc-arm-embedded

Thank you!