编译器分析和优化人
日常belike

Shrink Wrap介绍
-在GCC和LLVM中的过去和现在

邱吉 qiuji@iscas.ac.cn
20250705

提纲

- 考古
- 论文介绍
- Prologue和Epilogue上还有什么优化
- Prologue和Epilogue上优化面临的问题

LLVM上最早的SW支持（2009）

by John Mosby <ojomojo@gmail.com>

Shrink Wrapping - RFC and initial implementation

Project Infrastructure LLVM Dev List Archives

John_Mosby Feb 2009

Hello LLVMdev,

I have been working with LLVM for just over a year now, mainly in the area of compilation for HDLs like SystemVerilog and SystemC.

Most of this work dealt with translation to LLVM IR, representing concurrent languages with LLVM and using LLVM analyses and transforms
for compiling onto proprietary simulation acceleration hardware. All of this work used the C back end exclusively, since I wanted a transparent
and easily debuggable flow.

To learn more about the code generator, I decided to try implementing shrink wrapping, a reasonably self-contained back end transformation pass.

I now have a preliminary implemenation of shrink wrapping, done as an option to prologue/epilogue insertion under the switch --shrink-wrap.
It is limited to X86 presently since that is the only target I have access to at the moment.

Feb 2009

1 / 19
Feb 2009

Mar 2009

- 以附件的形式提交了patch文件
- Chris合入了
- John在2010年以后就不活跃了

https://discourse.llvm.org/t/shrink-wrapping-rfc-and-initial-implementation/12381

LLVM上最早的SW支持和删除（2009&2013）

by John Mosby <ojomojo@gmail.com>

```
commit a1be2dcd6398299c267005c679755d925f8435c4
Author: John Mosby <ojomojo@gmail.com>
Date:   Fri Mar 27 06:09:40 2009 +0000

    Shrink wrapping in PEI: initial release. Finishing development, enable with --shrink-wrap.

    llvm-svn: 67828
```

2013年移除了最初的SW代码. “unused and untested”， 据说是按照Fred Chow的论文来实现的

```
commit dbec9d9b2a6ce38d21e3ab2a60b0252d51f4d384
Author: Rafael Espindola <rafael.espindola@gmail.com>
Date:   Thu Oct 31 14:07:59 2013 +0000

    Remove the --shrink-wrap option.

    It had no tests, was unused and was "experimental at best".

    llvm-svn: 193749
```

“ It seems that this algorithm has been avoided because of the compile time impact that was not worth compared to the performance improvement.”

LLVM上的第二次SW支持(2015.04)

by Quentin Colombet quentin.colombet@gmail.com (apple, google, meta)

reviews.llvm.org/D9210

Phabricator

This is an archive of the discontinued LLVM Phabricator instance.

Differential › D9210

Add a shrink-wrapping pass to improve the placement of prologue and epilogue.

Closed Public

Authored by qcolombet on Apr 22 2015, 3:50 PM.

Details

Reviewers qcolombet
grosbach

Commits rG61b305edfd86: [ShrinkWrap] Add (a simplified version) of shrink-wrapping.

SUMMARY

Hi,

@Jim , putting you as reviewer as you seemed to have touched PEI more than most people :).

This patch introduces a new pass that computes the safe point to insert the prologue and epilogue of the function.
The interest is to first safe points that are cheaper than the entry and exits blocks.

- Context **

Currently we insert the prologue and epilogue of the method/function in the entry and exits blocks. Although this is correct, we can do a better job when those are not immediately required and insert them at less frequently executed places.
The job of the shrink-wrapping pass is to identify such places.

- Motivating example **

Let us consider the following function that perform a call only in one branch of a if:
define i32 @f(i32 %a, i32 %b) {

- Quentin目前还一直活跃
- 还在关注SW的实现和优化

https://reviews.llvm.org/D9210

LLVM上的第二次SW支持(2015.04)-discussion

Heads-Up: Shrink-wrapping and TargetFrameLowering API changes

Project Infrastructure ■ LLVM Dev List Archives

qcolombet May 2015

May 2015

1 / 1
May 2015

Hi all,

The shrink-wrapping pass landed in r236507. This email gives a brief survey on how you can use it and for out-of-tree target, how you have to update the TargetFrameLowering to be able to compile.

To know more about the goal of the shrink-wrapping pass, look at the description in http://reviews.llvm.org/D9210 1 . The short story is that shrink-wrapping provides a cheaper placement, in terms of frequency, of the prologue and epilogue code sequence.

** Out-of-Tree Target: TargetFrameLowering API Changes **

This section helps the maintainers of Out-of-Tree targets to update their code to the latest APIs. It also describes why those changes are necessary and thus, may be useful for anyone that want to use the shrink-wrapping pass. The shrink-wrapping pass is disabled by default (see *How Can I Use Shrink-Wrapping?* for more details).

With the introduction of shrink-wrapping some APIs in **TargetFrameLowering** needed to be updated to accommodate the fact, that the prologue block is not necessarily the entry block of the function.

May 2015

https://discourse.llvm.org/t/heads-up-shrink-wrapping-and-targetframelowering-api-changes/36503

LLVM上的第二次SW支持-commit

```
commit 61b305edfd861d27726d7b0a9cdffd18d6423cdb
Author: Quentin Colombet <qcolombet@apple.com>
Date:   Tue May 5 17:38:16 2015 +0000

    [ShrinkWrap] Add (a simplified version) of shrink-wrapping.

    This patch introduces a new pass that computes the safe point to insert the
    prologue and epilogue of the function.
    The interest is to find safe points that are cheaper than the entry and exits
    blocks.

    As an example and to avoid regressions to be introduce, this patch also
    implements the required bits to enable the shrink-wrapping pass for AArch64.


    ** Context **

    Currently we insert the prologue and epilogue of the method/function in the
    entry and exits blocks. Although this is correct, we can do a better job when
    those are not immediately required and insert them at less frequently executed
    places.
    The job of the shrink-wrapping pass is to identify such places.
```

关于性能的测试和讨论

[Shrink-Wrapping] Request For Benchmarking: X86 and AArch64

Project Infrastructure ■ LLVM Dev List Archives

qcolombet May 2015

May 2015

1 / 8

May 2015

Hi,

Shrink-wrapping capabilities, i.e., better placement of prologue and epilogue sequences, landed in r236507 but are not yet enabled by default.

Since r236507 AArch64 is shrink-wrapping ready, meaning we can turn the pass on for this target.
I’ve done the same for X86 in r 238293.

Now, I need your help to test and benchmark how shrink-wrapping perform on those targets.

The goal is to decide whether or not the support is good enough to be enabled by default.

** How Can I Test/Benchmark It? **

Add (-mllvm) -enable-shrink-wrap on your command line or patch the XXXConfigPass to set EnableShrinkWrap to true.
Note the -enable-shrink-wrap=<bool> takes precedence over whatever is set for EnableShrinkWrap.

Jul 2015

https://discourse.llvm.org/t/shrink-wrapping-request-for-benchmarking-x86-and-aarch64/36799

后续的改进

Shrink wrapping vs SplitCSW @2017.5

by Kit Barton <kbarton@ca.ibm.com> for PPC ISA

lists.llvm.org/pipermail/llvm-dev/2017-May/112623.html

[llvm-dev] RFC: Shrink wrapping vs SplitCSR

Kit Barton via llvm-dev llvm-dev at lists.llvm.org
Tue May 2 19:54:11 PDT 2017

- Previous message: [llvm-dev] Permissions for llvm-mirror – Setting up Libc++ Appveyor builders
- Next message: [llvm-dev] RFC: Shrink wrapping vs SplitCSR
- **Messages sorted by:** [date] [thread] [subject] [author]

```
Hi all,

We've seen several examples recently of performance opportunities on
POWER if we can improve the location of save/restore code for
callee-saved registers. Both Nemanja and myself have discussed this with
several people, and it seems that there are two possibilities for
improving this:

  1. Extend shrink wrapping to make the analysis of callee-saved
     registers more precise.
  2. Focus on enabling and (possibly) improving SplitCSR.

I would like opinions from people on the preferred way to proceed.

I am leaning toward improving shrink wrapping, at least as a short-term
solution. However, I fully admit that this is because I am familiar with
the shrink wrapping code and completely naive about SplitCSR and what
work would be necessary to get this working well.

My proposal would be to implement the flow sensitive analysis described
by Fred Chow (PLDI '88) and make the necessary extensions in shrink
wrapping to handle multiple save/restore points. At that point we can do
an evaluation to understand the improvements it provides and the impact
on compile time. Once we have these results, we can look at the best way
to enable it (e.g., option, target opt-in, higher opts, etc.).

Thoughts?

Kit
```

- 但最后没有提交代码

https://discourse.llvm.org/t/rfc-shrink-wrapping-vs-splitcSW/44793

Shrink-wrapping improvement @2017.5

by Francis Visoiu Mistrih <fvisoiumistrih at apple.com> thegameg 没合并

reviews.llvm.org/D41765

Phabricator

This is an archive of the discontinued LLVM Phabricator instance.

Differential › D41765

[CodeGen] Provide an advanced shrink-wrapping interface

Needs Review | Public

Authored by thegameg on Jan 5 2018, 7:24 AM.

Details

Reviewers MatzeB, qcolombet, kbarton, gberry, junbuml

SUMMARY

In order to extend shrink-wrapping we need to make some changes to the interface.

When trying to extend shrink-wrapping I came across the following limitations in terms of interface:

- We cannot have multiple save points.
- We cannot have multiple restore points.
- We cannot choose which registers are being saved / restored for each block.
- We cannot separate the Prologue / Epilogue emission from the callee saved register storing / loading.
- We cannot reuse the same algorithm to shrink-wrap anything else than CSRs and stack usage.

The goal here would be to provide an interface that would allow us to develop shrink-wrapping extensions in-tree, and being able to switch between implementations without actually having to handle both interfaces in PEI.

In order to be able to simply re-use this, we need to know which basic blocks **use** which element (as an input), and which basic blocks are the best suited for saving / restoring each element (as an output).

reviews.llvm.org/D30808

Phabricator

This is an archive of the discontinued LLVM Phabricator instance.

Differential › D30808

shrink-wrap: implement more advanced algorithm

Needs Review | Public

Authored by thegameg on Mar 9 2017, 8:46 PM.

Details

Reviewers MatzeB

SUMMARY

THIS IS A WORK IN PROGRESS.

Putting this up for early review.

This pass is an improvement of the current shrink-wrapping pass, based, this time, on the dataflow analysis described in "Minimizing Register Usage Penalty at Procedure Calls – Fred C. Chow" [1]. The aim of this improvement is to remove the restriction that the current shrink-wrapping pass is having, which is having only one save / restore point.

https://discourse.llvm.org/t/rfc-shrink-wrapping-improvement/45834
https://reviews.llvm.org/D36109
https://reviews.llvm.org/D30808 里面有大量的关于trade off和算法的讨论
https://reviews.llvm.org/D41765

但是，Quentin说：

qcolombet added a comment. Mar 15 2017, 7:36 AM

Hi Francis,

Out of curiosity, what is the compile time impact of this early version compared to the performance gain?

I am not aware of the improvement that Lupo and Wilken did to the algorithm, but back in 2009, we worked on Fred's shrink-wrapping and the compile time impact was just not worth the improvement.
You can have a look at 378553cb for instance. Thus, I am wondering if we are heading in the same direction and if yes, if instead iterative improvements of the current approach wouldn't be more productive.

Cheers,
–Quentin

我对 Lupo 和 Wilken 对算法的改进并不知情，但在 2009 年，我们曾一起研究 Fred 的收缩包装，编译时间的影响根本不值得所带来的改进。例如，你可以看看 378553cb。因此，我想知道我们是否朝着相同的方向发展，如果是的话，当前方法的迭代改进是否会更具生产力。

2020年的讨论

by Jimmy_Zhongduo_Lin <jimmy.zhongduo.lin@huawei.com>

RFC] Shrink-wrapping improvement

Project Infrastructure ■ LLVM Dev List Archives

Jimmy_Zhongduo_Lin May 2020

Hi,

Sorry for bringing back such an old thread. I am interested in the latest status of the shrink wrapping pass in LLVM. I have found some active work back in 2017 from Francis Visoiu Mistrih and Kit Barton from the following links. However, it seems that all the work suddenly stops and the improvement for the existing shrink wrapping did not make it into the trunk. Is this work abandoned?

https://lists.llvm.org/pipermail/llvm-dev/2017-August/116131.html

http://lists.llvm.org/pipermail/llvm-dev/2017-May/112687.html

https://reviews.llvm.org/D41765 1

Thanks,

Jimmy

Reply

May 2020

1 / 6
May 2020

Oct 2020

- Francis给了回复，提到了unwind
- 没下文了

https://discourse.llvm.org/t/rfc-shrink-wrapping-improvement/55333

时间来到了2024，@RISC-V

2024年来自syntacore的讨论和修改@2024.4

Elizaveta Noskova

hrink-wrap-save-restore-points-splitting/83581

Shrink Wrap Save/Restore Points Splitting

Code Generation Common Infrastructure

enoskova-sc 1 Dec 2024

Motivation

Currently ShrinkWrap pass produces maximum one Save and one Restore point, if managed to find.
The Save point is chosen as NCD of BBs, in which some of the Calee Saved Registers are defined or used (or frame index operand / frame managing instr contained).
The NCPD of described blocks become Restore point.

But this is obviously not optimal.
Consider the following function with early exit – https://godbolt.org/z/vejzEjb4a 3 .
In first BB we use only s1:

```
        add     a3, s1, a5
        addw    a3, a3, a4
        bge     a3, a1, .LBB0_6
```

but save all of the used calee saved registers:

Dec 2024

1 / 15
Dec 2024

Back

10d ago

https://discourse.llvm.org/t/shrink-wrap-save-restore-points-splitting/83581

syntacore提交的代码

```
commit 4dd103e9c65de7d3dbf12e76fbb72724127ec325
Author: Elizaveta Noskova <159026035+enoskova-sc@users.noreply.github.com>
Date:   Wed Apr 3 11:22:43 2024 +0300

    [CodeGen][ShrinkWrap] Clarify StackAddressUsedBlockInfo meaning (#80679)
```

```
commit 3088c316994f078833cba11086b6c5cb29df2aae
Author: Elizaveta Noskova <159026035+enoskova-sc@users.noreply.github.com>
Date:   Wed Jan 22 11:55:02 2025 +0300

    [llvm] Add NCD search on Array of basic blocks (NFC) (#119355)

    Shrink-Wrap points split Part 2.
    RFC:
    https://discourse.llvm.org/t/shrink-wrap-save-restore-points-splitting/83581

    Part 1: https://github.com/llvm/llvm-project/pull/117862
    Part 3: https://github.com/llvm/llvm-project/pull/119357
    Part 4: https://github.com/llvm/llvm-project/pull/119358
    Part 5: https://github.com/llvm/llvm-project/pull/119359
```

但是，Quentin说：

qcolombet Dec 2024

Just to give a little bit of perspective on why I haven't really push shrink-wrapping towards individual save/restore points: some tools or mechanisms can stop working because of this (like unwinding) on some systems.

My word of caution is doing a smart(er) shrink-wrapping is easy (relatively speaking), playing nice with the rest of the ecosystem may be hard. Mileage may vary 🙂 .

Reply

只是想给大家一点背景，说明为什么我没有真正推进收缩包装朝着个别保存/恢复点发展：在某些系统上，一些工具或机制可能会因为这个而无法正常工作（比如展开）。
我的警告是，做一个更智能的收缩包装相对简单，但与整个生态系统其他部分良好配合可能非常困难。效果因情况而异。

另一种从Register allocation角度的思路@2024.3

by Mikhail Gudim <mgudim@ventanamicro.com> (就是Jeff Law的那个公司)

[RISCV][WIP] Let RA do the CSR saves. #90819

Open **mgudim** wants to merge 2 commits into `llvm:main` from `mgudim:save_csr_in_ra`

Conversation 69 | Commits 2 | Checks 4 | Files changed 23

mgudim commented on May 2, 2024 Contributor

We turn the problem of saving and restoring callee-saved registers efficiently into a register allocation problem. This has the advantage that the register allocator can essentialy do shrink-wrapping on per register basis. Currently, shrink-wrapping pass saves all CSR in the same place which may be suboptimal. Also, improvements to register allocation / coalescing will translate to improvements in shrink-wrapping.

In `finalizeLowering()` we copy all callee-saved registers from a physical register to a virtual one. In all return blocks we copy do the reverse.

mgudim requested review from **asb**, **preames**, **mshockwave**, **topperc**, **pcwang-thead** and **wangpc-pp** last year

https://github.com/llvm/llvm-project/pull/90819

对上面两种思路的评估

by michaelmaitland Michael Maitland 之前在sifive，今年4月去了meta

Ventana的思路

Syntacore的思路

benchmark	% improvement from #90819	% improvement from #119359
520.onmetpp_r	failed to build; excluded from geomean	-0.5053852089
500.perlbench_r	2.779585667	0.001528011602
502.gcc_r	8.57668599	-0.01639500151
505.mcf_r	0.7296558265	0.0004807892291
523.xalancbmk_r	1.623262317	-0.3166050354
525.x264_r	0.9712921909	qemu error; excluded from geomean
531.deepsjeng_r	0.4709669223	1.035302603
541.leela_r	3.800597897	-0.0001516787202
557.xz_r	-2.771439081	qemu error; excluded from geomean
geomean	1.977243131	0.02738473831

@mgudim 's numbers seem to be in line with what he was observing. @enoskova-sc do these numbers look correct for your side?

https://discourse.llvm.org/t/shrink-wrap-save-restore-points-splitting/83581/7

SW在RISC-V上的支持

[RISCV] Allow shrink wrapping for RISC-V @2019.05

by Lewis-Revill @Embecosm

Differential > D62190

[RISCV] Allow shrink wrapping for RISC-V

Closed Public

Authored by lewis-revill on May 21 2019, 5:39 AM.

Details

Reviewers asb
luismarques

Commits rGcd800f3b226b: [RISCV] Allow shrink wrapping for RISC-V

SUMMARY

Enabling shrink wrapping requires ensuring the insertion point of the epilogue is correct for MBBs without a terminator, in which case the instruction to adjust the stack pointer is the last instruction in the block.

```
commit cd800f3b226b25142f233beca846715fc601809b
Author: lewis-revill <lewis.revill@embecosm.com>
Date:   Tue Jan 14 18:59:11 2020 +0000

    [RISCV] Allow shrink wrapping for RISC-V

    Enabling shrink wrapping requires ensuring the insertion point of the
    epilogue is correct for MBBs without a terminator, in which case the
    instruction to adjust the stack pointer is the last instruction in the
    block.

    Differential Revision: https://reviews.llvm.org/D62190
```

https://reviews.llvm.org/D62190

GCC上支持RISC-V SW@2022.10

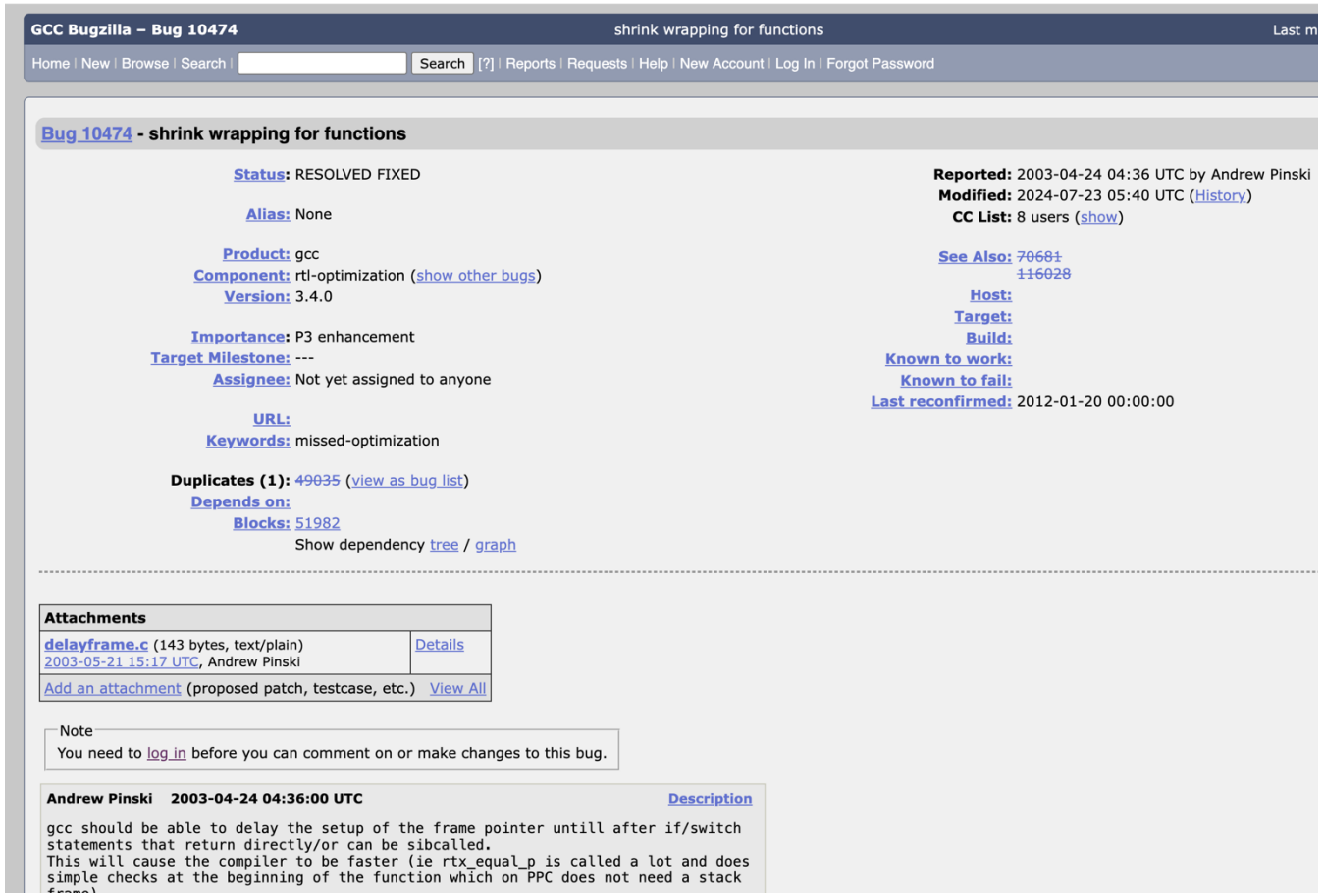
by Manolis Tsamis <manolis.tsamis@vrull.eu>

```
> >                                  # dynamic instructions
> >                     w/o shrink-wrap   w/ shrink-wrap      reduction
> > 500.perlbench_r     1265716786593     1262156218578     3560568015   0.28%
> > 500.perlbench_r      779224795689      765337009025    13887786664   1.78%
> > 500.perlbench_r      724087331471      711307152522    12780178949   1.77%
> > 502.gcc_r            204259864844      194517006339     9742858505   4.77%
> > 502.gcc_r            244047794302      231555834722    12491959580   5.12%
> > 502.gcc_r            230896069400      221877703011     9018366389   3.91%
> > 502.gcc_r            192130616624      183856450605     8274166019   4.31%
> > 502.gcc_r            258875074079      247756203226    11118870853   4.30%
> > 505.mcf_r            662653430325      660678680547     1974749778   0.30%
> > 520.omnetpp_r        985114167068      934191310154    50922856914   5.17%
> > 523.xalancbmk_r      927037633578      921688937650     5348695928   0.58%
> > 525.x264_r           490953958454      490565583447      388375007   0.08%
> > 525.x264_r          1994662294421     1993171932425     1490361996   0.07%
> > 525.x264_r          1897617120450     1896062750609     1554369841   0.08%
> > 531.deepsjeng_r     1695189878907     1669304130411    25885748496   1.53%
> > 541.leela_r         1925941222222     1897900861198    28040361024   1.46%
> > 548.exchange2_r     2073816227944     2073816226729           1215   0.00%
> > 557.xz_r             379572090003      379057409041      514680962   0.14%
> > 557.xz_r             953117469352      952680431430      437037922   0.05%
> > 557.xz_r             536859579650      536456690164      402889486   0.08%
> >                    18421773405376    18223938521833   197834883543   1.07%  totals
> >
```

https://gcc.gnu.org/pipermail/gcc-patches/2022-October/603824.html

SW在GCC上的支持

GCC的SW支持bugzilla history



GCC Bugzilla – Bug 10474 shrink wrapping for functions Last m

Home | New | Browse | Search | Search [?] | Reports | Requests | Help | New Account | Log In | Forgot Password

Bug 10474 - shrink wrapping for functions

Status: RESOLVED FIXED
Alias: None
Product: gcc
Component: rtl-optimization (show other bugs)
Version: 3.4.0
Importance: P3 enhancement
Target Milestone: ---
Assignee: Not yet assigned to anyone
URL:
Keywords: missed-optimization
Duplicates (1): 49035 (view as bug list)
Depends on:
Blocks: 51982
Show dependency tree / graph

Reported: 2003-04-24 04:36 UTC by Andrew Pinski
Modified: 2024-07-23 05:40 UTC (History)
CC List: 8 users (show)
See Also: 70681 116028
Host:
Target:
Build:
Known to work:
Known to fail:
Last reconfirmed: 2012-01-20 00:00:00

Attachments

delayframe.c (143 bytes, text/plain) Details
2003-05-21 15:17 UTC, Andrew Pinski

Add an attachment (proposed patch, testcase, etc.) View All

Note
You need to log in before you can comment on or make changes to this bug.

Andrew Pinski 2003-04-24 04:36:00 UTC Description

gcc should be able to delay the setup of the frame pointer untill after if/switch statements that return directly/or can be sibcalled.
This will cause the compiler to be faster (ie rtx_equal_p is called a lot and does simple checks at the beginning of the function which on PPC does not need a stack

- 最早是在2003年有patch

https://gcc.gnu.org/bugzilla/show_bug.cgi?id=10474

提纲

- 考古
- 论文介绍
- Prologue和Epilogue上还有什么优化
- Prologue和Epilogue上优化面临的问题

Fred Chow @PLDI1988

- **Minimizing register usage penalty at procedure calls**
- https://www.bilibili.com/video/BV1bW4y1S76T from 00:48 to 01:31
- 这篇文章包含两个不同的分配方法:
  - A one-pass inter-procedural register allocation scheme based on processing the procedures in a depth-first traversal of the call graph is presented.
  - A separate technique uses data flow analysis to optimize the placement of the save/restore code for registers within individual procedures. （这个才是Shrink wrapper）

提纲

- 考古
- 论文介绍
- Prologue和Epilogue上还有什么优化
- Prologue和Epilogue上优化面临的问题

Prologue和Epilogue可以做的优化

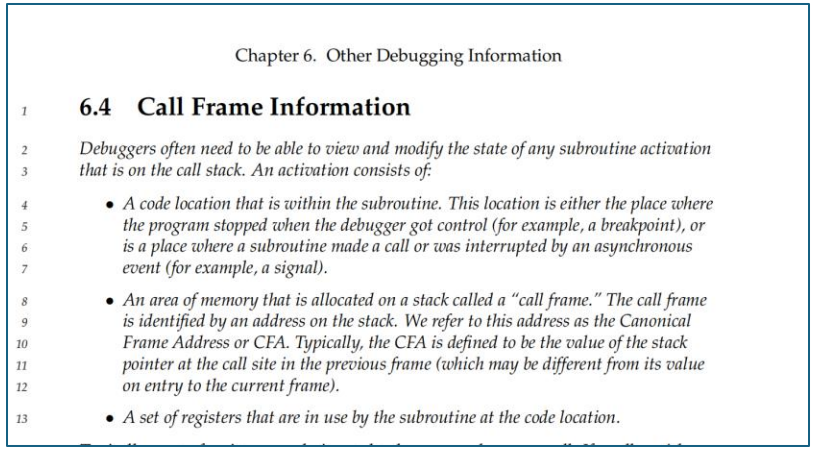
- 使用libcall消除重复的save/restore序列 –msave-restore

```
commit 07f7c00208b393296f8f27d6cd3cec2b11d86fd8
Author: lewis-revill <lewis.revill@embecosm.com>
Date:   Tue Feb 11 21:23:03 2020 +0000

    [RISCV] Add support for save/restore of callee-saved registers via libcalls

    This patch adds the support required for using the __riscv_save and
    __riscv_restore libcalls to implement a size-optimization for prologue
    and epilogue code, whereby the spill and restore code of callee-saved
    registers is implemented by common functions to reduce code duplication.

    Logic is also included to ensure that if both this optimization and
    shrink wrapping are enabled then the prologue and epilogue code can be
    safely inserted into the basic blocks chosen by shrink wrapping.

    Differential Revision: https://reviews.llvm.org/D62686
```

提纲

- 考古
- 论文介绍
- Prologue和Epilogue上还有什么优化
- Prologue和Epilogue上优化面临的问题

Shrink Warp有什么副作用



Chapter 6. Other Debugging Information

6.4 Call Frame Information

Debuggers often need to be able to view and modify the state of any subroutine activation that is on the call stack. An activation consists of:

- *A code location that is within the subroutine. This location is either the place where the program stopped when the debugger got control (for example, a breakpoint), or is a place where a subroutine made a call or was interrupted by an asynchronous event (for example, a signal).*
- *An area of memory that is allocated on a stack called a "call frame." The call frame is identified by an address on the stack. We refer to this address as the Canonical Frame Address or CFA. Typically, the CFA is defined to be the value of the stack pointer at the call site in the previous frame (which may be different from its value on entry to the current frame).*
- *A set of registers that are in use by the subroutine at the code location.*

- Debug tool的unwinding会出现问题
- sane stack traces in the debugger. Unwinding won't also work.
- 涉及的知识“DWARF Debugging Format Standard”
  - https://dwarfstd.org/doc/Debugging%20using%20DWARF-2012.pdf
- 可能的代码膨胀（多点S/R下会出现）：
  - https://gist.github.com/Lapshin/5b7bd6144327766239e631bd9a3e8ef9
  - see also： https://godbolt.org/z/P3WfTszYn
  - 不过这是个嵌入汇编的极端例子

调试信息的输出

- .cfi_def_cfa_offset RealStackSize
- “As comment 6 on that GCC bug mentions, disabling shrink-wrapping is part of what's necessary to make sure GCC sets up the frame pointer at the top of the function itself, not just inside some if that needs the prologue.”

Q&A Discussion