

Geneva GCC Improvement Workshop – Jan 26 and 27, 2005

"Improving the GNU Compiler Collection (GCC) on Itanium"

(summarized and synthesized from flipchart notes, discussions, and presentations; **supplement** to slides;
NOTE: recorded comments may represent an individual view and not a Gelato consensus)

Contents

Overview	2
Next Steps.....	3
Attendees	4
Day 1 – Meeting Introduction	4
Day 1 – Kalyan Muthukumar (Intel)	4
Day 1 – Mark K. Smith (Gelato)	5
Day 1 – Michael Matz (SUSE)	5
Day 1 – General Discussion Led by Al Stone (HP)	6
GCC Improvement Goals and Constraints.....	6
Possible Optimizations	8
Day 2 – Arutyun Avetisyan (RAS).....	9
Day 2 – Wenguang Chen (Tsinghua U).....	9
Day 2 – Bob Kidd (UIUC).....	9
Day 2 – Shin-Ming Liu (HP)	10
Day 2 – Shin Yee Chung (IHPC)	10
Day 2 – General Discussion Led by Al Stone (HP)	11
Improvement 1: SMS/Rotating Registers – Stage 1 window 3/05-6/05 (Mark Davis – Intel).....	11
Improvement 2: Memory Disambiguation – Stage 2 window 6/05-8/05 (Shin-Ming Liu – HP).....	12
Improvement 3: Superblock Scheduling – Stage 2 window 6/05-8/05 (Wen-mei Hwu – UIUC).....	13
Other Toolchain Issues.....	13

Geneva GCC Improvement Workshop – Jan 26 and 27, 2005

"Improving the GNU Compiler Collection (GCC) on Itanium"

(summarized and synthesized from flipchart notes, discussions, and presentations; **supplement** to slides)

<http://www.gelato.org/community/workshop/gcc/index.php>

Overview

Many have dismissed Itanium as viable platform based only on GCC-compiled application performance. It is generally thought that improving GCC performance on Itanium will accelerate Itanium's adoption in the broader computing community beyond HPC.

This workshop brought together members of the Gelato Federation and the GCC community interested in improving GCC (the GNU Compiler Collection) on Itanium 2 (tm) processors. An "improved GCC" is a GCC that produces faster running code for Itanium 2 (tm), ideally in shorter compilation times.

The initial goals for the workshop were to develop and encourage an active Gelato forum for discussing issues related to improving GCC on Itanium, understand work in progress that could be applied to improving GCC, and examine ways to cooperate/collaborate in improving GCC.

The workshop accomplished the following:

- identified specific upstream improvements to GCC that could potentially boost performance on Itanium 2 (tm),
- broke down the top three upstream improvements thought most likely to produce significant benefit into specific tasks, in order to gain insight into the expertise and levels of effort required to implement these improvements,
- allowed attendees to exchange information on their relevant work that might be leveraged into improvement implementations,
- discussed how to work with the community and get the changes accepted upstream
- identified possible project managers and implementers for each of the three recommended GCC improvements.

The top three possible improvements were selected based on metrics of:

(1) difficulty

- technical
- level of effort
- political ramifications

(2) and benefit

- runtime integer (INT) performance
- runtime floating point (FP) performance
- leveragability of the changes

It was noted that many improvements will help GCC in general, even if Itanium has the most to gain.

It was decided that the best performance metric to use would be runtime INT performance, given that GCC is the system compiler on GNU/Linux distributions, and would therefore be most visible. In addition, special consideration was given to how these changes might integrate into typical GCC release schedules, based on the way the community development process works and the desire to respect that process.

Key to the discussion was also the awareness that we must be able to concretely measure the progress made by improvements to GCC. It quickly became clear that collecting data from SPECint is a "given"; that is, that the results from SPECint are an essential part of the metrics needed to determine progress. It

Geneva GCC Improvement Workshop – Jan 26 and 27, 2005

"Improving the GNU Compiler Collection (GCC) on Itanium"

(summarized and synthesized from flipchart notes, discussions, and presentations; **supplement** to slides)

<http://www.gelato.org/community/workshop/gcc/index.php>

also became clear that attendees were very interested in the Gelato Vanilla benchmark suite of applications as a means to obtain an even better measure of progress in improving INT performance. The Gelato Vanilla suite includes common, everyday applications and provides a better measure of real world performance gains achieved by GCC improvement.

The top three recommended improvements to GCC were:

	est. INT gain	est. FP gain
superblock scheduling	11%	6%
rotating register support	2%	30%
memory disambiguation	12%	

The following individuals agreed to be possible project managers (PM) or implementors (IMP):

	PM	IMP
superblock scheduling	Wen-mei Hwu (UIUC)	Bob Kidd (UIUC)
rotating register support	Mark Davis (Intel)*	Tsinghua U
memory disambiguation	Shin-Ming Liu (HP)	RAS?

* Shortly after the meeting, Intel agreed to take on the PM role for the rotating register work.

Al Stone from HP agreed to handle the community relations role which includes introducing the proposed changes to the GCC community, provide critical technical introductions, and raise awareness about the workshop and the initial improvement projects in the general community.

Prior to the meeting, Duraid Madina, from University of Tokyo, provided his thoughts on improving GCC. http://www.gelato.org/pdf/Workshops/geneva05/duraid_gcc_workshop_jan05.pdf.

Attendees were also asked to review some of the existing lists of recommended optimizations:

- <http://gcc.gnu.org/projects/ia64.html>
- <http://www.gelato.org/software/wishlist.php>

as well as information on current GCC improvements in progress:

- <http://www.gccsummit.org/2004/2004-GCC-Summit-Proceedings.pdf>
- <http://www.linux.org.uk/~ajh/gcc/gccsummit-2003-proceedings.pdf>

Next Steps

Each project manager will further evaluate specific areas to get a better handle on the resources and time duration of the tasks.

Members of the group were highly encouraged to submit papers to the upcoming GCC Summit to raise visibility of GCC improvement for Itanium 2 (tm) undertaken by this group. Deadline for abstract submission is March 1, 2005. <http://www.gccsummit.org/2005/cfp.php>

Geneva GCC Improvement Workshop – Jan 26 and 27, 2005

"Improving the GNU Compiler Collection (GCC) on Itanium"

(summarized and synthesized from flipchart notes, discussions, and presentations; **supplement** to slides)

<http://www.gelato.org/community/workshop/gcc/index.php>

Attendees

Arutyun Avetisyan (RAS)	Sverre Jarp (CERN)	Gerald Pfeifer (SUSE)
Michel Benard (HP)	Ping-Hui Kao (HP)	Suresh Rao (Intel)
Wenguang Chen (Tsinghua U)	Bob Kidd (UIUC)	Mark K. Smith (Gelato)
Shin Yee Chung (IHPC)	Shin-Ming Liu (HP)	Al Stone (HP)
Patrick Demichel (HP)	Michael Matz (SUSE)	Weimin Zheng (Tsinghua U)
Wen-mei Hwu (UIUC)	Kalyan Muthukumar (Intel)	

Attendee details: http://www.gelato.org/pdf/Workshops/geneva05/attendees_gcc_workshop_jan05.pdf

Attendee group photo: <http://www.crhc.uiuc.edu/~smithmk/group.JPG>

Day 1 – Meeting Introduction

http://www.gelato.org/pdf/Workshops/geneva05/workshop_intro_smith.pdf

Mark K. Smith (Gelato) presented a very brief overview of the Gelato Federation and laid out the background, concerns, goals, and objectives for the workshop.

Day 1 – Kalyan Muthukumar (Intel)

http://www.gelato.org/pdf/Workshops/geneva05/compiling_itanium_muthu.pdf

Muthu presented a comprehensive set of slides detailing the potential performance gains that could be achieved by various techniques in GCC. These performance gains were for icc (the Intel Itanium 2 (tm) compiler), but can be generally extended to GCC in order to obtain a performance estimate for the various optimization techniques.

Intel stands ready to contribute knowledge about various compiler techniques to improve performance, but will not share any code due to IP issues. To that end, Muthu brought several papers to share with the workshop attendees.

http://www.gelato.org/pdf/Workshops/geneva05/disambiguation_micro2001_lavery.pdf

http://www.gelato.org/pdf/Workshops/geneva05/icc_generator_2000_intel.pdf

http://www.gelato.org/pdf/Workshops/geneva05/rot_reg_prefetch.pdf

http://www.gelato.org/pdf/Workshops/geneva05/swp_early_exits_intel.pdf

http://www.gelato.org/pdf/Workshops/geneva05/wavefront_scheduling_ieee1999.pdf

icc spec2000 performance gains for various optimization techniques

	INT gain	FP gain
Software pipelining	2%	30%
Global/superblock scheduling	11%	6%
Memory disambiguation	12%	
Predication		3%
Data prefetching	5%	41%
Rotating registers	2%	30%

20% to 30% noops are typical for Itanium – These numbers are AI's rule of thumb for GCC-generated code on ia64. There are gaps in the bundling algorithms that seem to result in noops occupying slots in some of the bundles where useful work could actually be done instead.

Geneva GCC Improvement Workshop – Jan 26 and 27, 2005

"Improving the GNU Compiler Collection (GCC) on Itanium"

(summarized and synthesized from flipchart notes, discussions, and presentations; **supplement** to slides)

<http://www.gelato.org/community/workshop/gcc/index.php>

Day 1 – Mark K. Smith (Gelato)

http://www.gelato.org/pdf/Workshops/geneva05/vanilla_gelato.pdf

Mark presented a very brief overview of the Gelato Federation/Intel Vanilla project. The Gelato Vanilla project will make highly-optimized Itanium binaries available for critical Linux utilities. The project aims to offer "recipes" for compiling applications including compiler options and performance analysis. The Gelato Vanilla suite includes common, everyday applications and provides a better measure of real-world performance gains achieved by GCC improvement. <http://co.gelato.org/vanilla> (web site still under construction).

Day 1 – Michael Matz (SUSE)

http://www.gelato.org/pdf/Workshops/geneva05/gcc_development_suse.pdf

Michael Matz from SUSE gave an overview of recent and future GCC development. See slides for more details an <http://gcc.gnu.org/gcc-4.0/changes.html> for additional details on GCC 4.0 features and projects in progress.

Developments:

User visible

- Precompiled headers
- Much closer to ISO C++
- Better performing libstdc++, also closer to ISO C++
- fvisibility
- character set support in C/C++

Internal

- Variable tracking
- Reorder functions
- Superblock formation
- Unit-at-a-time

Recent Developments

- Realistic code size estimates
- Improved PGO
- Tree-SSA
- LNO (loop nest optimizer)
- Fortran 95 frontend
- Infrastructure

Future and WIP

- Integrate rest of LNO + autovect
- IPO + Infrastructure
- Optimizer improvements
- Restructuring GIMPLE / RTL interaction
- Region based compilation
- C++
- Objective-C++

Geneva GCC Improvement Workshop – Jan 26 and 27, 2005

"Improving the GNU Compiler Collection (GCC) on Itanium"

(summarized and synthesized from flipchart notes, discussions, and presentations; **supplement** to slides)

<http://www.gelato.org/community/workshop/gcc/index.php>

Java
Mono?

Michael and Gerald not only provided keen technical insight into GCC, they provided valuable advice on how to work with the GCC community to ensure that changes have the greatest chance of being accepted back into the project. In addition, they educated workshop attendees about the current GCC development cycle. This information was key to determining what improvements could be done within the time frame of the next major GCC release, 4.1.

- GCC development process: <http://gcc.gnu.org/develop.html>
- GCC: submitting a patch: <http://gcc.gnu.org/contribute.html>
<http://gcc.gnu.org/cvs.html>

Michael and Gerald suggested that branches could be created that were improvement specific, especially when the needed changes look to be invasive – e.g., a rotating register branch. They also stressed that reviewing the branches already created could be useful, too; someone may have already started a similar line of research or investigation and it's always better to collaborate, if at all possible. Common sense is also called for – branches are clearly not needed for 20-line algorithm changes.

For reporting bugs, it is always most helpful to attach sample code to expose problems; if it is not possible to release the original code used to find the bug due to confidentiality restrictions, please make every effort to create sample code that exposes the bug. Shin-Ming Liu of HP and Suresh Rao of Intel indicated that both companies have extensive test suites that they intend to apply to GCC in the very near future. The goal of this work would be to ensure the correctness of GCC, first and foremost. Wherever possible, both companies were highly encouraged to share their test cases – not just the results – with the GCC community in order to improve the compiler for all Itanium 2 users. Indeed, some of the test cases may be completely processor-independent, and those are especially welcomed in the community.

Day 1 – General Discussion Led by Al Stone (HP)

This time was used to identify key improvement areas for GCC on Itanium 2 (tm).

GCC Improvement Goals and Constraints

First and foremost, improve run-time performance and code quality; these seemed to be the consensus goals.

Benefiting the GCC infrastructure would be good for the community and Itanium 2 (tm) for the long term.

In choosing what to work on, though, we need to consider "difficulty":

- How hard is an optimization technically?
- How much effort will that work take?
- What political limitations will we run into?

Stated repeatedly -- whatever work we do needs to be done within the community framework to be successful. Openness, transparency, and cooperation are absolutely essential.

Geneva GCC Improvement Workshop – Jan 26 and 27, 2005

"Improving the GNU Compiler Collection (GCC) on Itanium"

(summarized and synthesized from flipchart notes, discussions, and presentations; **supplement** to slides)

<http://www.gelato.org/community/workshop/gcc/index.php>

How Do We Tell When We're Done?

There was a lot of discussion here that was not completely captured.

The net result appeared to be agreement that we need to:

1. Define a set of benchmarks for measuring performance.
2. Capture baseline values.
3. Compare results to other compilers (e.g., icc was mentioned more than once :).

Much discussion led us to agreement that using SPECint and the Vanilla Suite from Gelato are our primary metrics. AI agreed to capture the baseline values soon. While SPECfp was also interesting, we seemed to come to the conclusion to use SPECint since it would be the benchmark most visible to Itanium 2 (tm) users, and that visible progress was essential.

Community, Community, Community

Cooperation with and discussions with the GCC community were stressed again and again. The importance of working *_with_* the community cannot be overstated -- throwing massive patches "over the wall" will never, ever work.

Work within the development stages for GCC; e.g., when stage 1 opens, be ready. Contribute during stage 2 and 3. GCC 4.0 is currently closed. GCC 4.1 stage 1 will be open from ~mid March, 2005.

Predicted GCC 4.1 Release Cycle:

GCC 4.1 Stage	Description	Time window*
Stage 1	Invasive changes	mid March to mid June, 2005
Stage 2	Modifications	mid June to mid August, 2005
Stage 3	Bug fixes	mid August to mid October, 2005

* *These are at best educated guesses based on past experience.*

It generally takes 18 months for changes to propagate out to the end-user enterprise.

This led to a discussion of political limitations: Gerald summed these up pretty concisely – “Don't break other platform, and you'll be safe.”

Community relationship is **key** to getting changes accepted.

- Be active on the GCC mailing list: <http://gcc.gnu.org/lists.html>
- Become a maintainer
- Attend and/or present at the GCC Summit in June
- Contribute code
- Interface with the community as an individual, not as a company; respect is accorded to individuals based on their technical merit, independent of who they actually work for.

Geneva GCC Improvement Workshop – Jan 26 and 27, 2005

"Improving the GNU Compiler Collection (GCC) on Itanium"

(summarized and synthesized from flipchart notes, discussions, and presentations; **supplement** to slides)

<http://www.gelato.org/community/workshop/gcc/index.php>

Possible Optimizations

NOTE: Muthu's presentation drove a lot of the content here. The possible optimizations were voted upon by the workshop attendees indicated in the "Votes" column. The "GCC Stage" indicates in which GCC development stage the potential optimizations could be integrated.

Votes	GCC stage	Description
13	1	Improve SMS/Rotating Registers
9	2	Memory disambiguation (array DD analysis, too)
3	2	Data prefetching (Shin Yee Chung expressed interest in pursuing this topic further)
0	1	Autovectorization (mostly FP improvements)
5	1	Superblock scheduling -- recovery code generation?
0	2	Instruction bundling
2	1	Control speculation; this is coupled with Instruction Scheduling, which could go in as late as stage 2, and which RAS is looking at already
2	2	Backend improvements (Al Stone and his colleagues are investigating some of these) -- better use of immediates -- machine model/-mtune options -- fp register costs
3	1	Aggressive inlining (RAS is currently investigating) -- IPO library support (Bob Kidd is interested in looking at this topic)
		General tuning (more of compile-time, than run-time issue, that people would look at as changes were being made -- more of an opportunistic item, than specific plan)

Geneva GCC Improvement Workshop – Jan 26 and 27, 2005

"Improving the GNU Compiler Collection (GCC) on Itanium"

(summarized and synthesized from flipchart notes, discussions, and presentations; **supplement** to slides)

<http://www.gelato.org/community/workshop/gcc/index.php>

Day 2 – Arutyun Avetisyan (RAS)

Current Project: Improving GCC's instruction scheduler for Itanium – Jan 15 to July 15, 2005

http://www.gelato.org/pdf/Workshops/geneva05/gcc_scheduler_ras.pdf

Arutyun's group at RAS has started a project to improve GCC's instruction scheduler for Itanium. The goals of the project are to implement control/data speculation support and solve the alias analysis problem by propagating alias information from TREE level to the scheduler.

To date the team has implemented and tested a patch for using GCC probability information. Currently a data speculation and an alias information patch are each being designed and implemented.

Arutyun and his team will present a report on their progress at the Gelato meeting in May, 2005. They are also thinking about submitting a paper to the GCC Summit in June.

Proposed Projects/Collaborations with Others:

http://www.gelato.org/pdf/Workshops/geneva05/project_proposal_ras.pdf

Arutyun's presentation covered ideas for improving GCC on Itanium. See slides for additional information.

ISP RAS proposed projects:

- (1) Implement aggressive instruction scheduler (addresses ILP features) – 4 to 5 engineer-years
- (2) Analyze GCC pitfalls on Itanium 2 (tm) with SPEC benchmark – 6 engineer months
- (3) Tune software pipelining for Itanium 2 (tm) – 6 engineer months
- (4) Make inlining for Itanium 2 (tm) more aggressive (addresses big caches) – 3-4 engineer months

Potential collaboration projects:

- (1) Implement autovectorization (addresses multimedia instructions) – 1 engineer year
- (2) Implement interprocedural optimizations framework

Day 2 – Wenguang Chen (Tsinghua U)

http://www.gelato.org/pdf/Workshops/geneva05/ia64_tsinghua.pdf

Wenguang outlined related work at Tsinghua University that could be leveraged for improving GCC on Itanium. Their main expertise is in code generation optimization and GCC front-end work. Tsinghua's current projects are:

- Code Generation Optimization for ORC
- OpenMP for ORC including GCC front-end extension to support OpenMP directives

Tsinghua proposed they could port optimization methods in ORC to GCC, specifically improve the current software pipelining implementation in GCC. Additionally, they could provide OpenMP support for GCC.

Day 2 – Bob Kidd (UIUC)

http://www.gelato.org/pdf/Workshops/geneva05/openimpact_kidd.pdf

Bob referred to his presentation he gave at Gelato meeting at Tsinghua University in October of 2004.

The IMPACT team's experience implementing an interprocedural IR may be helpful to attempts to do the same in GCC. They recently updated their high level IR with an interprocedural symbol table. This database-like structure allows efficient access to any symbol in the program at any time. They also completed an IR-level linker (ld on steroids). In addition to emulating ld, this component attempts to do a source level merge of code that would normally only be linked at a binary level. This involves merging

Geneva GCC Improvement Workshop – Jan 26 and 27, 2005

"Improving the GNU Compiler Collection (GCC) on Itanium"

(summarized and synthesized from flipchart notes, discussions, and presentations; **supplement** to slides)

<http://www.gelato.org/community/workshop/gcc/index.php>

types, unifying structure layout where discrepancies exist, and even fixing incorrect code to preserve the behavior of the original binary linking.

They can also offer their experience with superblock scheduling. As this already exists in GCC to an extent, this is a more reasonable starting point than the interprocedural IR. They have implemented superblocks and predication in IMPACT, although recovery code is still unsupported. They are willing to contribute to GCC's superblock implementation.

Day 2 – Shin-Ming Liu (HP)

Shin-Ming works with Steve Ellcey. Shin-Ming's group is converting an HP-UX internal compiler test suite to GCC. He indicated that he is interested in presenting on the converted test suite at the May Gelato meeting. Additionally, he would like to work with Gelato and Intel (Suresh) on the Gelato Vanilla project.

Day 2 – Shin Yee Chung (IHPC)

http://www.gelato.org/pdf/Workshops/geneva05/algorithms_chung.pdf

As the performance gap between the processor and the memory widens, the cache, as a fast memory buffer, becomes an important hardware resource to bridge the gap. Due to rapid hardware replacements in the industry, one cannot afford to perform extensive manual fine-tuning for each new hardware cycle. IHPC proposed Adaptive Cache-Oblivious algorithms which optimize the cache performance without detailed information about the cache configurations and perform run-time self-tuning without manual fine-tuning. Interestingly, their experiments show the execution time of the Adaptive Cache-Oblivious Matrix Transposition is comparable to the conventional tiling implementation on multiple platforms with different hardware configurations.

Geneva GCC Improvement Workshop – Jan 26 and 27, 2005

"Improving the GNU Compiler Collection (GCC) on Itanium"

(summarized and synthesized from flipchart notes, discussions, and presentations; **supplement** to slides)

<http://www.gelato.org/community/workshop/gcc/index.php>

Day 2 – General Discussion Led by Al Stone (HP)

This time was used focus in on the three improvement areas for GCC on Itanium 2 (tm) identified on Day 1 of the workshop.

Improvement 1: SMS/Rotating Registers – Stage 1 window 3/05-6/05 (Mark Davis – Intel)

What would it take to get this work done?

NOTE: After the meeting, Intel volunteered to start to build more details into this plan – Mark Davis is the PM (Project Manager). Tsinghua U expressed interest as possible implementers of this technology.

- (1) Contact Haifa developers to see where they are and what they have planned (subscribe and then cc GCC list).
- (2) Verify correctness of current SMS work (Suresh can help here)
 - test suite
 - benchmark
 - collect baseline benchmark data (use Gelato Vanilla)
- (3) Compare SMS static performance to icc, aCC (HPUX compiler) -- i.e., examine assembler output.
 - Fix bugs
 - Repeat

Estimate: items 1-3 would take from 3-6 EM (Engineer-Months)

- (4) Itanium infrastructure additions (this is stuff that may have to go in during stage1, and this part could possibly happen in parallel with the prior three steps). So, for rotating registers:
 - (a) data dependency graph
 - (b) scheduling that is rotating register aware
 - (c) register allocation on rotating registers only
 - (d) add in all the Itanium 2 (tm) specific things

Estimate: item 4 would take ~12 EM

- (5) Disable unrolling of modulo scheduled loop kernels (use rotating registers as needed for SMS)

Estimate: item 5 would take ~1 EM

- (6) Interface to existing register allocator

Estimate: item 6 would take ~1 EM

References:

Wen-mei Hwu presentation:

http://www.gelato.org/pdf/Workshops/geneva05/modulo_scheduling_hwu.pdf

Dan Lavery's Ph.D. thesis:

<http://www.crhc.uiuc.edu/IMPACT/papers/phd-thesis-daniel-lavery.html>
<http://www.crhc.uiuc.edu/IMPACT/ftp/report/phd-thesis-daniel-lavery.pdf>

Geneva GCC Improvement Workshop – Jan 26 and 27, 2005

"Improving the GNU Compiler Collection (GCC) on Itanium"

(summarized and synthesized from flipchart notes, discussions, and presentations; **supplement** to slides)

<http://www.gelato.org/community/workshop/gcc/index.php>

Improvement 2: Memory Disambiguation – Stage 2 window 6/05-8/05 (Shin-Ming Liu – HP)

What would it take to get this work done?

NOTE: Shin-Ming volunteered to start examining how to build more details into this plan -- he's the putative PM (Project Manager).

This effort is best to have one group to work on it contiguously. We could define the roadmap. But, the progress is highly dependent on the implementer. We need to establish interactive development model or treat it as a black box and monitor the performance progress.

- (1) Verify existing implementation – check on participants, make sure result gets used by code generator
 - Effort in RAS funded by HP, to analyze the current implementation and fix up loopholes
 - Gelato Vanilla Suite –O performance for progress monitoring
 - Use Gelato Vanilla Suite to identify missing opportunities

Estimate: 1-3 EM

- (2) Check with existing structure-aliasing branch (Dan Berlin/Zadeck)

Estimate: 1 EM

- (3) Incorporate structure-aliasing branch, same process as step 1

- Effort in RAS funded by HP

- (4) Intra procedural enhancements

- Pick enhancement items from Dan Lavery's paper

Estimate: 6-9 EM

- (5) Inter procedural -- "huge-r work"

- Cannot be done unless compilation model changed

- (6) Array DD information/analysis (see Sebastian Pop to understand his current work in this area) -- "huge work"

- Pending on Autovectorization phase stabilized

- (7) Provide aliasing probabilities

References:

Dan Lavery's slides from Kalyan Muthukumar's presentation

http://www.gelato.org/pdf/Workshops/geneva05/compiling_itanium_muthu.pdf

Dan Lavery's paper

http://www.gelato.org/pdf/Workshops/geneva05/disambiguation_micro2001_lavery.pdf

Geneva GCC Improvement Workshop – Jan 26 and 27, 2005

"Improving the GNU Compiler Collection (GCC) on Itanium"

(summarized and synthesized from flipchart notes, discussions, and presentations; **supplement** to slides)

<http://www.gelato.org/community/workshop/gcc/index.php>

Improvement 3: Superblock Scheduling – Stage 2 window 6/05-8/05 (Wen-mei Hwu – UIUC)

What would it take to get this work done?

NOTE: Wen-mei volunteered to start examining how to build more details into this plan -- he's the PM (Project Manager).

(1) Verify existing implementation

Estimate: 2 EM

(2) DAG building -- may already be done?

(3) Superblock loops

Estimate: 1 EM

(4) Investigate tail duplication heuristics

Estimate: 1 EM

(5) Speculation models (will require results to convince folks of it's usability)

Estimate: 1 EM

(6) KAPI libs in machine description (as an -foption or -moption?); there was some discussion on re-using work done in gas, at least as a starting point.

Other notes:

First, implement things without recovery code, then, implement recovery code. Later on, introduce data speculation.

References:

KAPI - KAPI is an API that allows other tools to read and interpret the knobsfile format. A knobsfile is a data file that describes various microarchitecture details.

<ftp://download.intel.com/software/opensource/kapi/kapi.pdf>

Wen-mei Hwu presentation:

http://www.gelato.org/pdf/Workshops/geneva05/speculation_hwu.pdf

Other Toolchain Issues

These are items that concern GCC on some level, and would definitely be part of a larger effort on the GNU toolchain on Itanium 2 (tm), though not perhaps pertinent to an Itanium 2 (tm) GCC effort:

- valgrind: `_please_` get it working on Itanium 2 (tm)!
- gdb: continues to have problems, needs lots of TLC
- MPI tools?
- annotated assembler: there's quite a bit of interest in assembler output carefully annotated to reflect the original source, regardless of where the instructions got moved to.