

# ISP RAS Projects on improving GCC for Itanium

Arutyun Avetisyan (arut@ispras.ru)  
Andrey Belevantsev (abel@ispras.ru)

Gelato Itanium Conference and Expo  
April 16-18, 2007  
San Jose, CA

# New instruction scheduler for GCC

- Based on the selective scheduling approach
  - focuses on VLIW architectures, but general enough for others
  - supports instruction cloning, speculative code motion, multiway branching, register renaming, forward substitution
- Provides an expandable framework
  - easy to support data and control speculation
  - easy to add more instruction transformations (e.g. mutation)
  - easy to tune for different targets
- Provides software pipelining on top of the scheduler
  - with a support for control-flow intensive/non-countable loops
  - can pipeline loop nests (from innermost to outermost)
  - supports control/data speculation during pipelining

## Project summary

- Requires a lot of development efforts
  - 15 months for basic implementation
  - 9 months for performance tuning
  - One of three biggest projects in the GCC backend
- Has a team of five
  - Consulting is provided by Vladimir Makarov, Red Hat
- Started in September 2005
  - Sources are published in January 2007
- Latest sources available in “sel-sched-branch” branch of the GCC repository
  - `svn co svn://gcc.gnu.org/svn/gcc/branches/  
sel-sched-branch`
- Aimed for inclusion in GCC 4.4

# Implementation highlights

- A flexible dependence analyzer infrastructure
  - Custom callbacks can be provided for dependence handling
  - Used in initialization, computing available instructions, checking the readiness of operands
  - Multiple dependence contexts can be handled
- Data initialization infrastructure
  - Supports on-the-fly initialization of scheduler data for newly created instructions and basic blocks
- Choosing the best instruction for scheduling
  - Uses interblock priorities to guide the choosing process
  - Uses the DFA lookahead engine of GCC
- Can work together with the existing scheduler
  - Dependence analysis and region finding are shared

## Current status

- All basic infrastructure is implemented
  - Bookkeeping support, register renaming, forward substitution
  - Software pipelining (innermost loops and loop nests)
  - Data and control speculation
  - Can be run before and after register allocation
- We are working on performance tuning
  - Better instruction priorities
  - Using of call saved registers for renaming
  - Rescheduling of pipelined loops for tighter schedules
  - Testing on SPEC CPU 2000 benchmarks
- The Itanium backend improvements are needed
  - Smarter placement of stop bits and alignment directives
  - Better interaction between the backend and the schedulers

## Results – small benchmarks

Benchmark	-O2	-O3 -funroll-loops	Benchmark	-O2	-O3 -funroll-loops
Whetstone	-0,50%	-1,94%	FLOPS	7,15%	3,46%
Tfftdp	1,62%	-1,65%		5,96%	8,82%
Sim	1,16%	0,00%		7,05%	9,81%
Shuffle	0,92%	0,68%		10,46%	12,57%
Linpackc	6,72%	3,89%	Hanoi: 29 disks	0,06%	0,04%
Heapsort	0,00%	5,17%	Fibonacci	-0,14%	-0,99%
Dhrystone 1.1	-1,51%	-1,19%	Queens	24,04%	6,85%
Dhrystone 2.1a	1,27%	2,28%	Scimark	0,00%	6,93%
c4 : Fhourstones 1.0	5,10%	-0,61%	FFT	2,61%	1,85%
Black Jack	0,95%	0,66%	SOR	1,55%	7,07%
Nsieve size=8191	9,71%	8,16%	Monte Carlo	0,04%	0,04%
FFT size=1000000	1,05%	0,13%	Sparse Matmult	-3,18%	8,70%
Blowfish	0,43%	-0,09%	LU	0,03%	9,17%

Speedup % to the old scheduler

## Results – SPEC

164.gzip	655	683	4,27%	168.wupwise	500	511	2,20%
175.vpr	803	795	-1,00%	171.swim	712	732	2,81%
176.gcc	X	X	X	172.mgrid	322	372	15,53%
181.mcf	699	687	-1,72%	173.applu	436	432	-0,92%
186.crafty	868	858	-1,15%	177.mesa	744	739	-0,67%
197.parser	676	670	-0,89%	178.galgel	691	705	2,03%
252.eon	692	707	2,17%	179.art	1760	1704	-3,18%
253.perlbnk	X	X	X	183.equake	447	444	-0,67%
254.gap	564	568	0,71%	187.facerec	390	393	0,77%
255.vortex	X	X	X	188.ampp	669	680	1,64%
256.bzip2	759	794	4,61%	189.lucas	854	832	-2,58%
300.twolf	1007	1013	0,60%	191.fma3d	273	273	0,00%
<b>GeoMean</b>	<b>737,25</b>	<b>743,31</b>	<b>0,82%</b>	200.sixtrack	182	184	1,10%
				301.apsi	507	515	1,58%
				<b>GeoMean</b>	<b>522,64</b>	<b>529,53</b>	<b>1,32%</b>

Speedup % to the old scheduler

# Improving alias analysis in GCC

- Providing more accurate info to the RTL level
  - Propagates alias information from Tree SSA to RTL
  - Uses it in the low-level RTL optimizers
  - Results in a few number of new disambiguations
  - More analysis is needed
    - Whether the saved information is precise enough
    - Does it stays intact throughout the RTL pipeline
- An extra alias analysis pass on RTL
  - Tracks “base + offset” computations relevant for Itanium
  - Currently works for acyclic regions (e.g. for scheduler)
- Further work on these projects is planned under guidance of Diego Novillo, Red Hat

# Improving modulo scheduling in GCC

- Make the modulo scheduling implementation to work on Itanium
  - Need to fix some code generation bugs
  - First patch is already published
- Improve data dependency analysis on RTL
  - Propagate the data dependency info from Tree SSA to RTL
  - Implement the verifier to check the consistency of saved data
  - Fix the problems found via the verifier in RTL optimizers
- Use the saved data in modulo scheduling
  - Test the implementation on hand-written tests and SPEC
- The project is sponsored by Intel Russia
  - Started in April 2007