

Improving GCC for IA64: future directions

Arutyun Avetisyan, ISP RAS
January 26-27, 2005, Geneva

Features of IA64 architecture

- Large memory cache
(L1 - 32KB, L2 - 256KB, L3 - up to 9MB)
- Large register file
(128 integer and floating-point registers, 64 predication registers, 8 branch registers)
- Features for exposing ILP
(speculation, predication, register rotation)
- Prefetching and branch prediction
- Multimedia instructions
(compatible with MMX and Streaming SIMD extensions)

ISP RAS proposed projects:

1. Implement aggressive instruction scheduler (addresses ILP features)
2. Analyze GCC pitfalls on IA64 with SPEC benchmark
3. Tune software pipelining for IA64
4. Make inlining for IA64 more aggressive (addresses big caches)

Potential collaboration projects:

1. Implement autovectorization (addresses multimedia instructions)
2. Implement interprocedural optimizations framework

GCC instruction scheduler: current status

- Uses simplified dominator path based approach
- Speculation and predication is not supported
- Set of instruction transformations useful for exploiting ILP is not supported
(register renaming, forward substitution, instruction cloning/mutation)
- Software pipelining is done as a separate pass

ISP RAS project #1.

Instruction scheduler: implement aggressive approach

Proposed instruction scheduler should:

- Use DAG approach
(e.g., selective scheduling by Moon and Ebcioğlu)
- Support set of instruction transformations useful for exploiting ILP
- Support for speculation and predication
- Perform software pipelining as part of the scheduler

Resources needed: 4-5 man-years

ISP RAS project #1. Instruction scheduler: requirements from other platforms

New instruction scheduler should:

- Be able to exploit different forms of speculation
(e.g., support non-volatile instructions of Fujitsu processors)
- Be configurable for other architectures
(e.g., disable instruction cloning to avoid code blowout on embedded platforms)
- Be able to use profile information

ISP RAS project #2. Analyzing GCC pitfalls: using SPECINT benchmark

- Thorough analysis of GCC pitfalls compiling for IA64 was never performed
- Reference software should be chosen (SPEC benchmark is proposed)
- Reference compiler should be chosen (Intel compiler is proposed)
- Outcome: a list of common GCC pitfalls, possible improvements for them

Resources: 6 man-months

ISP RAS project #3. Software pipelining: tuning existing implementation

- Software pipelining is implemented as Swing Modulo Scheduling (SMS)
- The pass aims to reduce register pressure, which is not suitable for IA64

Proposed project: fix current SMS implementation by disabling its efforts of reducing register pressure on IA64

Resources: 6 man-months

ISP RAS project #4. Inlining framework: tuning existing implementation

- Inlining is implemented in GCC as interprocedural framework (unit-at-a-time compilation mode)
- It is tuned for x86 and x86_64, limiting maximal size of the function to be inlined

Proposed project: tune current inlining framework for IA64, benefiting from IA64 cache size

Resources: 3-4 man-months

Collaboration project #1. Autovectorization: enhancing current implementation

- Autovectorization is being implemented for GCC in LNO branch
- First version is already in GCC CVS
- Currently supports vectorizing of simple loops
- Mainly tuned for PowerPC platform

Proposed project: enhance current implementation with support for IA64, participate in further development

Resources: 1 man-year for IA64 support

Collaboration project #2. Interprocedural optimizations: implementing a framework

- Interprocedural optimizations (IPO) improve code by $\sim 15-45\%$
- GCC has no support for any inter-module optimizations
- Requires redesign of GCC compilation process and changes in internal representation (tree-ssa)
- Requires new optimizations and modifications of old ones

The project requires energy of the whole community involved.