Introduction

Compilers for High Performance Architectures

Josep Llosa, Josep Ramon Herrero, Marc González

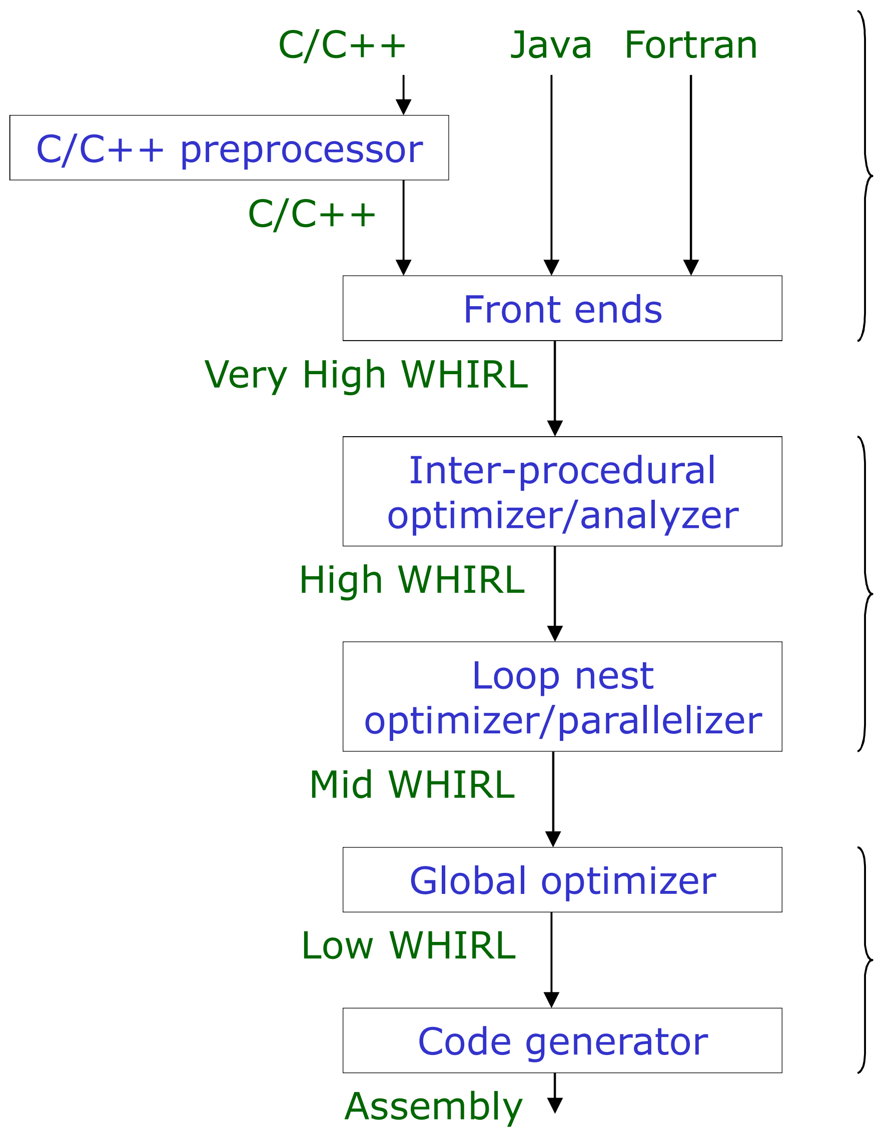
Course Outline

1. Introduction (we are now at it)
2. Instruction Level Parallelism Optimizations (Josep Llosa)
   - Instruction Level Parallelism
   - Machine Independent Optimizations
   - Instruction Scheduling
   - Register Allocation
3. Memory Hierarchy Optimizations (Josep Ramon Herrero)
   - Basic Concepts
   - Basic transformations
   - Combination of transformations
4. Thread Level Parallelism Optimizations (Marc González)
   - Thread level Parallelism
   - Analysis and detection of parallelism
   - Programming models
   - Parallel execution
   - Memory models

Awakening Some Interest in Compilers

- Traditional strength at UPC - Architecture
  - Superscalar processors, do everything in hardware
- Little on compilers
  - Hardware people have to understand compilers
    - No attention to compilers -> bad processor design
    - Architects with no compiler knowledge tend to introduce compiler unfriendly hardware features which cannot be effectively exploited
  - Compilers required to effectively exploit hardware features
    - Expose ILP for the processor (even superscalar)
    - Exploit memory locality
    - Parallelize code for multiprocessors/multicomputers
  - Optimizing applications
    - Understanding how the compiler works helps to optimize source level code

Structure of a real compiler: ORC



Front end
Traditional compilers class
(parsing, analysis, etc…)
NOT this course topic

Middle end
Covered in topics:
3) Memory optimizations
4) TLP optimizations

Back end
Covered in topic:
2) ILP optimizations

Structure of a real compiler: ORC

- IPA: Inter-procedural analyzer
  - Analyzes a routine and transfers the information beyond the scope of the routine
  - Some Optimizations:
    - Inlining
    - Cloning
    - Dead function/variable elimination
    - Constant propagation
- LNO: Loop nest optimizer/parallelizer
  - Performs high level transformation to better use cache memory and to enhance/expose thread level parallelism (parallelization only partially implemented)
  - Some Optimizations:
    - Loop fission
    - Loop fusion
    - Unroll & jam
    - Loop interchange
    - Loop peeling
    - Loop tiling
    - Data prefetching

Structure of a real compiler: ORC

- WOPT: Global Optimizer
  - Performs machine independent optimizations at the procedure level
  - Some Optimizations:
    - Dead code elimination
    - Partial redundancy elimination
    - Strength reduction
    - Induction variable elimination
    - Register promotion
- CG: Code Generator
  - Performs machine dependent optimizations and code generation/optimization
  - Some Optimizations:
    - If conversion
    - Software pipelining
    - Predication
    - Global instruction scheduling
    - Register allocation
    - Local instruction scheduling

Course Grades

What to do to pass the course:

- Attendance control: Attend all classes, missed classes mean a reduced score in participation.
  - Every missed class reduces 0.2 points the attendance score
  - 0 to 2 missed classes is acceptable -> no points deducted
  - 3 missed classes 0.6 will be deducted
  - 10 or more missed classes no attendance score
- Course presentation:
  - Each student must prepare a class on a particular topic (more on that later)
- Report:
  - Write a report on your topic
- Read other students reports:
  - Ask questions during other students presentations

- 40% deliverable of assignment
- 40% presentation of assignment
- 20% attendance and participation in class

About presentations

- Students will work in groups of 2 students
- Each group must prepare a 25 minutes class on a particular topic
  ⇒ 25 minutes is a hard limit you will be interrupted if you overextend
- There is a list of starting papers for each topic
  ⇒ Research other related papers
- Do NOT describe individual papers
  ⇒ Present the topic in a related & coherent way
- Don’t take everything you read as a fact
  ⇒ Be critical

About presentations

- The objective is NOT to show the professors that you know a lot about your topic

  ⇒ The objective is that the other students learn what you know

- Don’t explain everything you have read in 25 minutes

  ⇒ Limit the scope of you presentation to fit in 25 minutes you can make a short overview and explain in depth the most important aspects

- Presentations will be done at class time the last weeks of the course (exact schedule to be announced)

About Reports

- Each group must write a report on his topic
- Approximate size of report should be ~20 pages
  - This is not a hard limit, there is no problem if you can explain all you need in 15 pages or if you need 25 or 30, is just so you know the amount of work we expect (i.e. no need to write 100 pages ☺)
- If something does not fit in the class and you think is interesting, the report is the place to have it.

- Reports should be available before presentations start
  - Deadlines to be anounced
- All students HAVE TO read all the reports
- During presentations, students HAVE TO make questions related either to the presentation or to the report.

Assignment topics

- Instruction Level Parallelism Optimizations
  - Speculation & Predication
  - Acyclic code scheduling
  - Modulo scheduling for control intensive loops
  - Advanced register allocation
  - Scheduling & Register allocation
  - Instruction scheduling for clustered VLIW
- Memory Hierarchy Optimizations
  - Software prefetch
  - Improving instruction cache
  - Data cache optimizations
  - Compilation techniques for SIMD Architectures
- Thread Level Parallelism Optimizations
  - Parallelism detection
  - Dependence tests
  - Enabling parallelism
  - Parallelism execution

Speculation and Predication

- S. A. Mahlke, W. Y. Chen, R. A. Bringmann, R. E. Hank, W. W. Hwu, B. R. Rau, and M. S. Schlansker, “Sentinel Scheduling: A Model for Compiler-Controlled Speculative Execution,” ACM Transactions on Computer Systems, vol. 11, no. 4, pp. 376-408, Nov. 1993
- Chang et al. (1995). P. P. Chang, N. J. Warter, S. A. Mahlke, W. Y. Chen, and W. W. Hwu, “Three Architectural Models for Compiler-Controlled Speculative Execution,” IEEE Transactions on Computers, vol. 44, no. 4, pp. 481-494, April 1995.
- D. I. August, D. A. Connors, S. A. Mahlke, J. W. Sias, K. M. Crozier, B. Cheng, P. R.Eaton, Q. B. Olaniran, and W. W. Hwu, “Integrated, Predicated, and Speculative Execution in the IMPACT EPIC Architecture,” Proceedings of the 25th Annual International Symposium on Computer Architecture, pp. 227-237, July 1998.
- Y. Choi, A. Knies, L. Gerke, and T.-F. Ngai, “The Impact of If-Conversion and Branch Prediction on Program Execution on the Intel Itanium Processor,” Proceedings of the 34th Annual International Symposium on Microarchitecture, pp. 182-191, Dec. 2001

Acyclic code scheduling

- A new viewpoint on code generation for directed acyclic graphs S. Liao, K. Keutzer, S. Tjiang, S. Devadas January 1998 ACM Transactions on Design Automation of Electronic Systems (TODAES), Volume 3 Issue 1
- Avoidance and suppression of compensation code in a trace scheduling compiler
  Stefan M. Freudenberger, Thomas R. Gross, P. Geoffrey Lowney. July 1994, ACM Transactions on Programming Languages and Systems (TOPLAS), Volume 16 Issue 4
- Performance evaluation of instruction scheduling on the IBM RISC System/6000. David Bernstein, Doron Cohen, Yuval Lavon, Vladimir Rainish. December 1992. ACM SIGMICRO Newsletter , Proceedings of the 25th annual international symposium on Microarchitecture, Volume 23 Issue 1-2
- The impact of if-conversion and branch prediction on program execution on the Intel® Itanium¿ processor Youngsoo Choi, Allan Knies, Luke Gerke, Tin-Fook Ngai December 2001 Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture

Modulo scheduling for control intensive loops

- P. Tirumalai, M. Lee, M. Schlansker. Parallelization of loops with exits on pipelined architectures. Proceedings of the 1990 ACM/IEEE conference on Supercomputing.
- Daniel M. Lavery, Wen-mei W. Hwu. Modulo scheduling of loops in control-intensive non-numeric programs. Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture.
- Nancy J. Warter-Perez, Noubar Partamian. Modulo scheduling with multiple initiation intervals. Proceedings of the 28th annual international symposium on Microarchitecture.
- SangMin Shim, Soo-Mook Moon. Split-path enhanced pipeline scheduling for loops with control flows. Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture.

Advanced Register allocation.

- E. Eichenberger, Edward S. Davidson. Register allocation for predicated code. Proceedings of the 28th annual international symposium on Microarchitecture.
- Bernhard Scholz, Erik Eckstein. Register allocation for irregular architectures ACM SIGPLAN Notices , Proceedings of the joint conference on Languages, compilers and tools for embedded systems: software and compilers for embedded systems, Volume 37 Issue 7.
- Jinpyo Park, Soo-Mook Moon. Optimistic register coalescing. ACM Transactions on Programming Languages and Systems (TOPLAS), Volume 26 Issue 4.
- B. R. Rau, M. Lee, P. P. Tirumalai, M. S. Schlansker. Register allocation for software pipelined loops. SIGPLAN Not., volume 27, number 7, 1992, pages 283-299.

Scheduling & register allocation

- Register allocation with instruction scheduling. Shlomit S. Pinter. June 1993, ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1993 conference on Programming language design and implementation, Volume 28 Issue 6
- J. R. Goodman and W.-C. Hsu. Code scheduling and register allocation in large basic blocks. ICS '88: Proceedings of the 2nd international conference on Supercomputing, 1988, pp. 442-452.
- Javier Zalamea, Josep Llosa, Eduard Ayguadé, Mateo Valero. Modulo scheduling with integrated register spilling for clustered VLIW architectures. Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture.

Instruction scheduling for clustered VLIW

- Giuseppe Desoli. Instruction Assignment for Clustered VLIW DSP Compilers: A New Approach. HP Laboratories Cambridge, HPL-98-13 February, 1998
- Emre Özer, Sanjeev Banerjia, Thomas M. Conte. Unified assign and schedule: a new approach to scheduling for clustered register file microarchitectures. Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture
- Jesus Sanchez, Antonio Gonzalez. The Effectiveness of Loop Unrolling for Modulo Scheduling in Clustered VLIW Architectures. International Conference on Parallel Processing (ICPP'00), 2000.
- Javier Zalamea, Josep Llosa, Eduard Ayguadé, Mateo Valero. Modulo scheduling with integrated register spilling for clustered VLIW architectures. Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture.

Software prefetch

- D. Callahan, K. Kennedy, A. Porterfield. Software prefetching. ACM SIGARCH Computer Architecture News, Volume 19 , Issue 2, April 1991.
- TC Mowry, MS Lam, A Gupta. Design and evaluation of a compiler algorithm for prefetching. ACM SIGPLAN Notices Volume 27 , Issue 9, September 1992.
- Mowry, T. C. 1998. Tolerating latency in multiprocessors through compiler-inserted prefetching. ACM Trans. Comput. Syst. 16, 1 (Feb. 1998), 55-92.
- Chi-Keung Luk and Todd C. Mowry. Automatic Compiler-Inserted Prefetching for Pointer-Based Applications. In IEEE Transactions on Computers, Vol. 48, No. 2, February 1999

Improving Instruction Cache

- McFarling, S. 1989. Program optimization for instruction caches. SIGARCH Comput. Archit. News 17, 2 (Apr. 1989), 183-191.
- McFarling, S.Procedure Merging with Instruction Caches. PLDI 1991, 71-79
- Karl Pettis and Robert C. Hansen. 1990. Profile guided code positioning. SIGPLAN Not. 25, 6 (Jun. 1990), 16–27. DOI:https://doi.org/10.1145/93548.93550
- WW Hisu, PP Chang. Achieving High Instruction Cache Performance With An Optimizing Compiler. The 16th Annual International Symposium on Computer Architecture, June 1989.

Data Cache Optimizations

- Michael E. Wolf and Monica S. Lam. 1991. A data locality optimizing algorithm. In *Proceedings of the ACM SIGPLAN 1991 conference on Programming language design and implementation* (PLDI '91). ACM, New York, NY, USA, 30-44.
- Michael E. Wolf, Dror E. Maydan, and Ding-Kai Chen. 1996. Combining loop transformations considering caches and scheduling. In *Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture* (MICRO 29). IEEE Computer Society, Washington, DC, USA, 274-286.
- Kathryn S. McKinley, Steve Carr, and Chau-Wen Tseng. 1996. Improving data locality with loop transformations. *ACM Trans. Program. Lang. Syst.* 18, 4 (July 1996), 424-453.
- Steve Carr and Ken Kennedy. 1994. Improving the ratio of memory operations to floating-point operations in loops. *ACM Trans. Program. Lang. Syst.* 16, 6 (November 1994), 1768-1810.
- Steve Carr, Kathryn S. McKinley, and Chau-Wen Tseng. 1994. Compiler optimizations for improving data locality. *SIGPLAN Not.* 29, 11 (November 1994), 252-262.
- Steve Carr. 1996. Combining Optimization for Cache and Instruction-Level Parallelism. In *Proceedings of the 1996 Conference on Parallel Architectures and Compilation Techniques* (PACT '96). IEEE Computer Society, Washington, DC, USA, 238-.

Compilation techniques for SIMD Architectures

- Krall A, Lelait S. Compilation techniques for multimedia processors. International Journal of Parallel Programming 2000;28(4):347–361.
- Pryanishnikov, I., Krall, A. and Horspool, N. (2007), Compiler optimizations for processors with SIMD instructions. Softw: Pract. Exper., 37: 93–113.
- Peng Wu; Eichenberger, AE.; Wang, A, "Efficient SIMD code generation for runtime alignment and length conversion," Code Generation and Optimization, 2005. CGO 2005. International Symposium on , vol., no., pp.153,164, 20-23 March 2005
- Alexandre E. Eichenberger, Peng Wu, and Kevin O'Brien. 2004. Vectorization for SIMD architectures with alignment constraints. In Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation (PLDI '04). ACM, New York, NY, USA, 82-93.
- Dorit Nuzman and Ayal Zaks. 2008. Outer-loop vectorization: revisited for short SIMD architectures. In Proceedings of the 17th international conference on Parallel architectures and compilation techniques (PACT '08). ACM, New York, NY, USA, 2-11.

Parallelism Detection

- "Efficient and Precise Array Access Analysis", Y. Paek, J. Hoeflinger, D Padua ACM Transactions on Programming Languages and Systems (TOPLAS) Volume 24 , Issue 1 (January 2002) Pages: 65 - 109
- "Efficient Interprocedural Analysis for Program parallelization and restructuring", Z. Li and P. Yew SIGPLAN, June 1988.
- "A Technique for Summarizing Data Access and Its Use in Parallelism Enhancing Transformations" V. Balasundaram, K. Kennedy Conference on Programming Language Design and Implementation Proceedings of the ACM SIGPLAN 1989 Conference on Programming language design and implementation Portland, Oregon, United States Pages: 41 – 53

Dependence Tests

- Banerjee/Wolfe test
  - M.Wolfe, U.Banerjee, "Data Dependence and its Application to Parallel Processing", Int. J. of Parallel Programming, Vol.16, No.2, pp.137-178, 1987
- Range test
  - William Blume and Rudolf Eigenmann. Non-Linear and Symbolic Data Dependence Testing, IEEE Transactions of Parallel and Distributed Systems, Volume 9, Number 12, pages 1180-1194, December 1998.
- Omega test
  - William Pugh. The Omega test: a fast and practical integer programming algorithm for dependence. Proceedings of the 1991 ACM/IEEE Conference on Supercomputing,1991
- I Test
  - Xiangyun Kong, David Klappholz, and Kleanthis Psarris, "The I Test: A New Test for Subscript Data Dependence," Proceedings of the 1990 International Conference on Parallel Processing, Vol. II, pages 204-211, August 1990.

Enabling Parallelism

- Peng Tu and D. Padua. Automatic Array Privatization. Languages and Compilers for Parallel Computing. Lecture Notes in Computer Science 768, U. Banerjee, D. Gelernter, A. Nicolau, and D. Padua (Eds.), Springer-Verlag, 1994.
- Zhiyuan Li, Array Privatization for Parallel Execution of Loops, Proceedings of the 1992 ACM International Conference on Supercomputing
- B. Pottenger and R. Eigenmann. Idiom Recognition in the Polaris Parallelizing Compiler. ACM Int. Conf. on Supercomputing (ICS'95), June 1995. (Extended version: Parallelization in the presence of generalized induction and reduction variables. www.ece.ecn.purdue.edu/~eigenman/reports/1396.pdf)

Parallelism Execution

- Yuri Dotsenko John Mellor-Crummey, François Cantonnet, Tarek El-Ghazawi, Ashrujit Mohanti, Yiyi Yao, Daniel Chavarría-Miranda, "An evaluation of global address space languages: co-array fortran and unified parallel C", Proceedings of the 10th ACM SIGPLAN symposium on Principles and practice of parallel programming (PPoPP2005)
- Chen Ding, Xipeng Shen, Kirk Kelsey, Chris Tice, Ruke Huang, and Chengliang Zhang, "Software Behavior Oriented Parallelization", PLDI2007
- Milind Kulkarni, Keshav Pingali, Bruce Walter, Ganesh Ramanarayanan, Kavita Bala, L. Paul Chew, "Optimistic Parallelism Requires Abstractions", PLDI2007
- E. Ayguadé, A. Duran, J. Hoeflinger, F. Massaioli, X. Teruel. "An Experimental Evaluation of the New OpenMP Tasking Model", In Proceedings of the 20th International Workshop on Languages and Compilers for Parallel Computing Urbana, USA. October 2007
- Alexandre E. Eichenberger, Kathryn O'Brien, Kevin K. O'Brien, Peng Wu, Tong Chen, Peter H. Oden, Daniel A. Prener, Janice C. Shepherd, Byoungro So, Zehra Sura, Amy Wang, Tao Zhang, Peng Zhao, and Michael Gschwind," Optimizing Compiler for the CELL Processor", PACT 2005