Tentamenstof Computerarchitectuur

Deze lijst is gebaseerd op de *vijfde* editie van *"Computer Architecture: A Quantitative Approach"*. Tot de tentamenstof behoren alle hieronder genoemde delen uit het boek, tenzij anders aangegeven.

Chapter 1: Fundamentals of Quantitative Design and Analysis

- Section 1.1: Introduction
- Section 1.2: Classes of Computers
- Section 1.3: Defining Computer Architecture (overlap met Appendix A!)
- Section 1.4: Trends in Technology
- Section 1.5: Trends in Power and Energy in Integrated Circuits
- Section 1.8: Measuring, Reporting and Summarizing Performance
- Section 1.9: Quantitative Principles of Computer Design
- Section 1.11: Fallacies and Pitfalls
- Section 1.12: Concluding Remarks

Appendix A: Instruction Set Principles

- Section A.1: Introduction
- Section A.2: Classifying Instruction Set Architectures
- Section A.3: Memory Addressing
- Section A.4: Type and Size of Operands
- Section A.5: Operations in the Instruction Set
- Section A.6: Instructions for Control Flow
- Section A.7: Encoding an Instruction Set
- Section A.8: Crosscutting Issues: The Role of Compilers
  - Only subsections "The Structure of Recent Compilers", "Register Allocation" and "The Impact of Compiler Technology on the Architect's Decisions".
- Section A.11: Concluding Remarks

Appendix B: Review of Memory Hierarchy

- Section B.1: Introduction
- Section B.2: Cache Performance
- Section B.3: Six Basic Cache Optimizations
- Section B.6: Fallacies and Pitfalls
- Section B.7: Concluding Remarks

Chapter 2: Memory Hierarchy Design

- Section 2.1: Introduction (Note overlap Appendix B)
- Section 2.2: Ten Advanced Optimizations of Cache Performance
- Section 2.3: Memory Technology and Optimizations
- Section 2.6: Putting it All Together
  - No need to study the details of this Section, but read it to test your understanding of the material.
- Section 2.7: Fallacies and Pitfalls
- Section 2.8: Concluding Remarks

Appendix C: Pipelining: Basic and Intermediate Concepts

- Section C.1: Introduction
- Section C.2: The Major Hurdle of Pipelining – Pipeline Hazards
- Section C.3: How Is Pipelining Implemented?
- Section C.4: What Makes Pipelining Hard to Implement?
- Section C.5: Extending the MIPS Pipelining to Handle Multicycle Operations
- Section C.7: Crosscutting Issues
  - Under "Dynamically Scheduled Pipelines" you can find a discussion of "Scoreboarding". Note the overlap of this section with Section 3.5.

Chapter 3: Instruction-Level Parallelism and Its Exploitation

- Section 3.1: Instruction-Level Parallelism: Concepts and Challenges
- Section 3.2: Basic Compiler Techniques for Exposing ILP
- Section 3.4: Overcoming Data Hazards with Dynamic Scheduling
- Section 3.5: Dynamic Scheduling: Examples and the Algorithm
- Section 3.6: Hardware-Based Speculation
- Section 3.7: Exploiting ILP Using Multiple Issue and Static Scheduling
- Section 3.8: Exploiting ILP Using Dynamic Scheduling, Multiple Issue, and Speculation
- Section 3.9: Advanced Techniques for Instruction Delivery and Speculation
  - Only the subsection on "Increasing Instruction Fetch Bandwidth"
- Section 3.10: Studies of the Limitations of ILP
- Section 3.12: Multithreading: Exploiting Thread-Level Parallelism to Improve Uniprocessor Throughput
  - Skip "Effectiveness of Fine-Grained Multithreading on the Sun T1"
  - Skip "Effectiveness of Simultaneous Multithreading on Superscalar processors".
- Section 3.14: Fallacies and Pitfalls
- Section 3.15: Concluding Remarks

Chapter 4: Data-Level Parallelism in Vector, SIMD and GPU Architectures

- Section 4.1: Introduction
- Section 4.2: Vector Architecture
- Section 4.3: SIMD Instruction Set Extensions for Multimedia
- Section 4.4: Graphics Processing Units
- Section 4.5: Detecting and Enhancing Loop-Level Parallelism
  - Only the first part, skip "Finding Dependences" and onwards.
- Section 4.9: Concluding Remarks
  - Note that many of the GPU features "to be addressed in the near future" have by now been addressed.

Chapter 5: Thread-Level Parallelism

- Section 5.1: Introduction
- Section 5.2: Centralized Shared-Memory Architectures
- Section 5.4: Distributed Shared-Memory and Directory-Based Coherence
- Section 5.10: Concluding Remarks