# CDA5155 Exam 2 Study Topics

Covers Appendix C, Chapter 3, Chapter 3 Extra, Static Pipelining paper, and Assignments 3, 4, and 5

1. Appendix C
    - 1.1. Classical Instruction Pipelining
    - 1.2. Limits of Pipeline Performance Improvements
    - 1.3. Structural Hazards
    - 1.4. Data Dependences
    - 1.5. Data Hazards
    - 1.6. Forwarding
    - 1.7. Instruction Scheduling
    - 1.8. Control Dependences
    - 1.9. Control Hazards
    - 1.10. Addressing Control Hazards
        - 1.10.1. Predict Not Taken
        - 1.10.2. Static and Semi-Static Branch Prediction
        - 1.10.3. Branch Prediction Buffers
            - 1.10.3.1. 1-Bit Branch Prediction Buffer
            - 1.10.3.2. 2-Bit Branch Prediction Buffer
    - 1.11. Exceptions
    - 1.12. Supporting Precise Exceptions
    - 1.13. Multiple Cycle Operations
    - 1.14. Complications Due to Multiple Cycle Operations

2. Chapter 3
    - 2.1. Compilation Techniques to Avoid Stalls
        - 2.1.1. Instruction Scheduling
        - 2.1.2. Loop Unrolling
    - 2.2. Enhanced Dynamic Branch Techniques
        - 2.2.1. Correlating Branch Prediction
        - 2.2.2. Gshare Predictor
        - 2.2.3. Tournament Branch Prediction
        - 2.2.4. Tagged Hybrid Predictors
        - 2.2.5. Branch Target Buffer
        - 2.2.6. Return Prediction
    - 2.3. Dynamic Scheduling and Out-of-Order Execution
        - 2.3.1. Scoreboarding Approach
        - 2.3.2. Nonspeculative Tomasulo Approach
        - 2.3.3. Speculative Tomasulo Approach
        - 2.3.4. Speculation Benefits and Costs

        2.3.5. Value Prediction
    2.4. Multiple Issue
        2.4.1. Statically Scheduled Superscalar Approach
        2.4.2. VLIW Approach
        2.4.3. Dynamically Scheduled Superscalar Approach
    2.5. Limits of Instruction-Level Parallelism
    2.6. Multithreading
        2.6.1. Differences between Threads and Processes
        2.6.2. Coarse-Grained Multithreading
        2.6.3. Fine-Grained Multithreading
        2.6.4. Simultaneous Multithreading

3. Chapter 3 extra
    3.1. Software Pipelining
    3.2. Using Conditional or Predicated Instructions

4. Static Pipelining
    4.1. Inefficiences with Traditional Instruction Pipelining
    4.2. Static Pipelining Datapath
        4.2.1. No Pipeline Registers
        4.2.2. Internal Registers Can Be Explicitly Accessed
        4.2.3. Compilation Process
        4.2.4. Exploiting Internal Registers to Perform Additional Optimizations
        4.2.5. Conclusions
            4.2.5.1. New Level of Compiler Optimizations Can Be Applied
            4.2.5.2. Performance, Code Size, and Energy Improvements Are Obtained