Computer Architecture Concepts

Design of Arithmetic Unit, ISA, CISC/RISC, Pipelines, Hazards, Scheduling, and More

Contents

- Arithmetic Unit Design
- Instruction Set Architecture
- CISC & RISC Scalar Processors
- Linear & Nonlinear Pipeline Processors
- Instruction Pipeline Design
- Pipeline Hazards
- Instruction Scheduling (Scoreboarding & Tomasulo's Algorithm)

Design of Arithmetic Unit

- Performs add, subtract, multiply, divide
- Components: ALU, shifters, adders, multipliers
- Fixed-point vs Floating-point design
- Examples: Carry Lookahead Adder, Booth Multiplier

Instruction Set Architecture (ISA)

- Interface between hardware & software
- Defines instruction formats, addressing modes, data types
- Registers & memory model

CISC Scalar Processors

- Complex Instruction Set Computer
- Rich instruction set, multiple cycles per instruction
- Pros: Easier compiler design, powerful instructions
- Cons: Slower execution, complex hardware

RISC Scalar Processors

- Reduced Instruction Set Computer
- Simple instructions, single-cycle execution
- Pros: Faster, pipeline-friendly
- Cons: More instructions needed per program

Linear Pipeline Processors

- Instructions executed in sequence of stages
- Example stages: IF → ID → EX → MEM → WB
- Improves throughput

Nonlinear Pipeline Processors

- Multiple execution paths (branching, parallel units)
- More flexible but harder to control
- Used in superscalar designs

Instruction Pipeline Design

- Break instruction into smaller tasks
- Allows parallel execution
- Improves CPU performance

Pipeline Hazards

- Structural Hazards – resource conflict
- Data Hazards – dependency issues (RAW, WAR, WAW)
- Control Hazards – due to branching

Instruction Scheduling

- Scoreboarding – centralized control to handle hazards
- Tomasulo's Algorithm – register renaming & reservation stations

Branch Handling Techniques

- Branch prediction
- Delayed branching
- Speculative execution

Arithmetic Pipeline

- Pipeline for arithmetic operations (e.g., floating point)
- Stages: multiplication, normalization, rounding
- Used in scientific computing

Conclusion

- Pipeline improves performance
- RISC vs CISC tradeoffs
- Instruction scheduling & branch handling critical for efficiency

References

- Computer Architecture: A Quantitative Approach – Hennessy & Patterson
- Computer Organization and Architecture – Stallings