Study Notes: Computer Architecture - Final Review

Von Neumann Model

- **Components:** Processor, Memory, Storage
- **Process:**
  - Instruction Fetch
  - Instruction Decode
  - Execution (ALU, Branch/Jump, Memory Operations)
  - Program Counter
  - Registers
- **Performance:** Execution time depends on how programs are loaded into memory.

CPU Performance Equation

- **Execution Time = (Instructions / Program) x (Cycles / Instruction) x (Seconds / Cycle)**
- **Goal:** Reduce instructions, cycles, and time.

Improving Performance

- **Lowering Cycle Time (CT):**
  - **Programmer:** Algorithms with fewer operations, wise use of instructions (vector, specialized), facilitate compiler optimizations.
  - **Compiler:** Inline functions, loop unrolling, common sub-expression elimination, constant propagation, loop invariant code motion.
- **Lowering Instruction Count (IC):**
  - Memory operations (caching, TLB).
  - Compiler: Use registers as often as possible.
  - Hardware: Pipelining, branch prediction, SuperScalar, OoO execution.
  - Programmer: Reduce data dependencies, make code more predictable.
- **Lowering Cycles Per Instruction (CPI):**
  - Maximum speedup.
  - Make the common case fast.
  - Optimization is a moving target.
  - Exploit parallelism.
  - Single-core performance matters.
  - Don't hurt the most common case too much.

Amdahl's Law

- `Speedup_enhanced(f, s) = 1 / ((1 - f) + (f / s))`
- `Speedup_max(f, ∞) = 1 / (1 - f)`
- Focus on optimizing the most frequent parts of the code.

Memory Hierarchy

- **Levels:** Registers, L1 Cache, L2 Cache, L3 Cache, DRAM, Storage

- **Characteristics:** Speed (fastest to slowest), Size (smallest to largest)
- **Locality:**
  - **Temporal:** Repeating access to the same data.
  - **Spatial:** Accessing data in nearby memory locations.

Cache Operation

- Processor sends memory access request to L1 cache.
- **Hit:** Return data (Read), Update L1 and set DIRTY (Write).
- **Miss:**
  - Empty block: Place data there.
  - No empty block: Select a victim (LRU policy).
  - Dirty victim: Write back to lower level.
  - Fetch requesting block from lower level and place in cache.

Cache-Friendly Code

- **C = ABS** (Capacity = Associativity x Block Size x Number of Sets)
- **Miss Types:**
  - **Compulsory:** First-time access.
  - **Conflict:** Insufficient associativity.
  - **Capacity:** Working set exceeds capacity.
- **Remedies:**
  - Condense data layout.
  - Prefetch.
  - Tiling.
  - Padding.
  - Loop fission/interchange.

Virtual Memory

- Abstraction of memory space for programs.
- OS and hardware map virtual addresses to physical addresses.
- Organized into pages.
- Each process has its own page table.

Address Translation

- Virtual Address -> Page Table -> Physical Address
- **TLB (Translation Look-aside Buffer):** Cache for page table entries.
- **Virtually Indexed, Physically Tagged Cache:** Index cache using virtual address, tag with physical address.

Pipelining

- Overlapping execution of multiple instructions.
- **Hazards:**
  - **Structural:** Resource conflicts.

    - **Control:** Branch/jump instructions.
    - **Data:** Data dependencies.
- **Solutions:**
    - Stalls, forwarding, branch prediction, dynamic scheduling.

Branch Prediction

- **Local Predictor:** Every branch has its own state.
- **Global History (GH) Predictor:** States associated with history of branch outcomes.
- **Tournament Predictor:** Hybrid approach, chooses between local and global predictors.

Data Hazards

- Instruction needs result from a prior instruction still in the pipeline.
- **Solutions:**
    - Stalls.
    - Data forwarding.
    - Dynamic scheduling.
    - Reorder instructions.

Register Renaming

- Map architectural registers to physical registers.
- Eliminate false dependencies (WAR, WAW).

Speculative Execution

- Execute instructions before knowing if they should be executed.
- **Reorder Buffer (ROB):** Store results of speculative instructions.
- **Commit/Retire:** Present results to program when non-speculative.

Superscalar

- Fetch/decode/issue multiple instructions per cycle.
- **Fetch Width:** Instructions fetched per cycle.
- **Issue Width:** Instructions issued per cycle.

Parallelism

- **Instruction-Level Parallelism (ILP):** Pipelining, OoO, Superscalar.
- **Data-Level Parallelism (DLP):** SIMD instructions.
- **Thread-Level Parallelism (TLP):** Multicore/SMT processors.

Simultaneous Multithreading (SMT)

- Multiple threads share processor resources.

Chip Multiprocessors (CMP)

- Multiple cores on a single chip.

Cache Coherency & Consistency

- **Coherency:** Guarantees processors see the same value for a variable.
- **Consistency:** All threads see data changes in the same order.

Alternative Parallel Architectures

- **Vector Processing:** Operate on vectors of data.
- **GPU (Graphics Processing Unit):** Massively parallel architecture for graphics and computation.

Power/Energy/Dark Silicon

- **Moore's Law:** Transistor density doubles approximately every two years.
- **Power:** Direct contributor to heat.
- **Energy:** Power x Execution Time.
- **Dark Silicon:** Some transistors must be turned off to stay within power budget.

Modern Processors

- Heterogeneous CMPs (P-cores/E-cores).
- Domain-specific accelerators (TPUs, Neural Engines).

Glossary

- **ALU:** Arithmetic Logic Unit. Performs arithmetic and logical operations.
- **CPI:** Cycles Per Instruction. Average number of clock cycles required for each instruction.
- **CT:** Cycle Time. Duration of a single clock cycle.
- **DRAM:** Dynamic Random-Access Memory. Main system memory.
- **IC:** Instruction Count. Number of instructions executed by a program.
- **LRU:** Least Recently Used. Cache replacement policy.
- **OoO:** Out-of-Order execution. Instructions executed based on data dependencies, not program order.
- **PC:** Program Counter. Register holding the address of the next instruction to be executed.
- **SIMD:** Single Instruction, Multiple Data. Instruction set architecture that performs the same operation on multiple data points simultaneously.
- **SMT:** Simultaneous Multithreading. Allows multiple independent threads of execution to better utilize the resources provided by modern processors.
- **TLB:** Translation Lookaside Buffer. A memory cache that is used to reduce the time it takes to access a user memory location.
- **TPU:** Tensor Processing Unit. A custom-developed accelerator designed to speed up machine learning workloads.