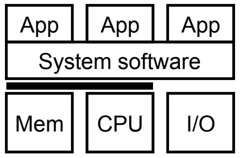
CIS 371
Computer Organization and Design

Unit 4: Performance & Multicycle

This Unit

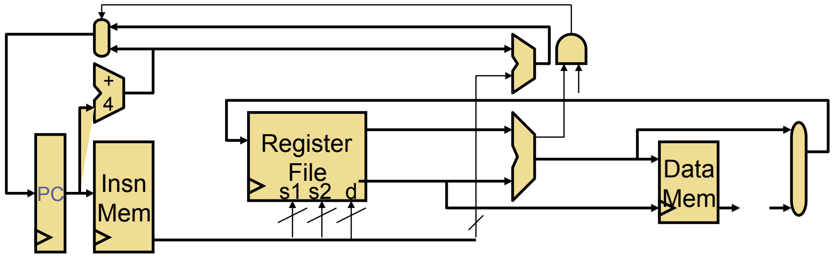


- Multicycle datapath
- Clock vs CPI
- CPU performance equation
- Performance metrics
- Benchmarking

Readings

- P&H
  - Chapter 1.4 (for performance discussion)

240 → 371



- CIS 240: build something that works
- CIS 371: build something that works "well"
  - "well" means "high-performance" but also cheap, low-power, etc.
  - Mostly "high-performance"
  - So, what is the performance of this?
  - What is performance?

CPU Performance Equation

- Multiple aspects to performance: helps to isolate them
- Latency = seconds / program =
  - (insns / program) * (cycles / insn) * (seconds / cycle)
  - **Insns / program**: dynamic insn count = f(program, compiler, ISA)
  - **Cycles / insn**: CPI = f(program, compiler, ISA, micro-arch)
  - **Seconds / cycle**: clock period = f(micro-arch, technology)
- For low latency (better performance) minimize all three
  - – Difficult: often pull against one another
  - Example we have seen: RISC vs. CISC ISAs
    - ± RISC: low CPI/clock period, high insn count
    - ± CISC: low insn count, high CPI/clock period

MIPS (performance metric, not the ISA)

- Factor out dynamic insn count, CPU equation becomes…
  - Latency: seconds / insn = (cycles / insn) * (seconds / cycle)
  - Throughput: **insns / second** = (insns / cycle) * (cycles / second)
- **MIPS** (millions of insns per second): insns / second * 10^{-6}
  - **Cycles / second**: clock frequency (in MHz)
  - Example: CPI = 2, clock = 500 MHz (2ns period), what is MIPS?
    - 0.5 * 500 MHz * 10^{-6} = 250 MIPS
- MIPS is okay for micro-architects
  - Typically work in one ISA/one compiler, treat insn count as fixed
- Not okay for general public
  - Processors with different ISAs/compilers have incomparable MIPS
  - Wait, it gets worse…

Mhz (MegaHertz) and Ghz (GigaHertz)

- 1 Hertz = 1 cycle per second
  1 Ghz is 1 cycle per nanosecond, 1 Ghz = 1000 Mhz
- Micro-architects often ignore instruction count…
- … but general public (mostly) also ignores CPI
  - Equates clock frequency with performance!!
- Which processor would you buy?
  - Processor A: CPI = 2, clock = 5 GHz
  - Processor B: CPI = 1, clock = 3 GHz
  - Probably A, but B is faster (assuming same ISA/compiler)
- Classic example
  - 800 MHz PentiumIII faster than 1 GHz Pentium4!
  - Same ISA and compiler!
- **Meta-point: danger of partial performance metrics!**

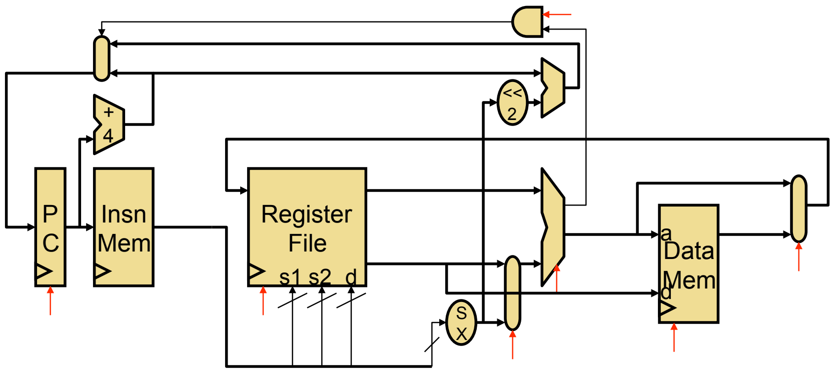
System Performance

- Clock frequency implies processor “core” clock frequency
  - Other system components have their own clocks (or not)
  - E.g., increasing processor clock doesn’t accelerate memory
- Example
  - Processor A: CPI_{CORE} = 1, CPI_{MEM} = 1, proc. clock = 500 MHz (2ns)
  - What is the speedup if we double processor clock frequency?
  - Base: CPI = 2 → IPC = 0.5 → MIPS = 250
  - New: CPI = **3** → IPC = 0.33 → MIPS = 333
    - Clock *= 2 → CPI_{MEM} *= 2
  - Speedup = 333/250 = **1.33** << 2
- What about an infinite clock frequency?
  - Only a 2X (factor of 2) speedup
  - Example of Amdahl’s Law

Amdahl's Law

- Literally: total speedup limited by non-accelerated piece
  - Example: can optimize 50% of program A
    - Even "magic" optimization that makes this 50% disappear…
    - …only yields a 2X speedup
- For consumers: buy a balanced system
- For microarchitects: build a balanced system
  - **MCCF (Make Common Case Fast)**
  - Focus your efforts on things that matter

Single-Cycle Datapath Performance

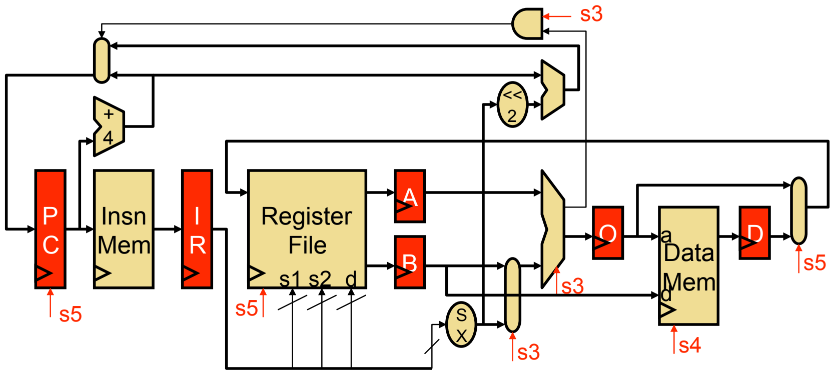


- Goes against make common case fast (MCCF) principle
  + Low CPI: 1
  – Long clock period: to accommodate slowest instruction
    - Especially if multiply/divide are included

Multi-cycle Operations

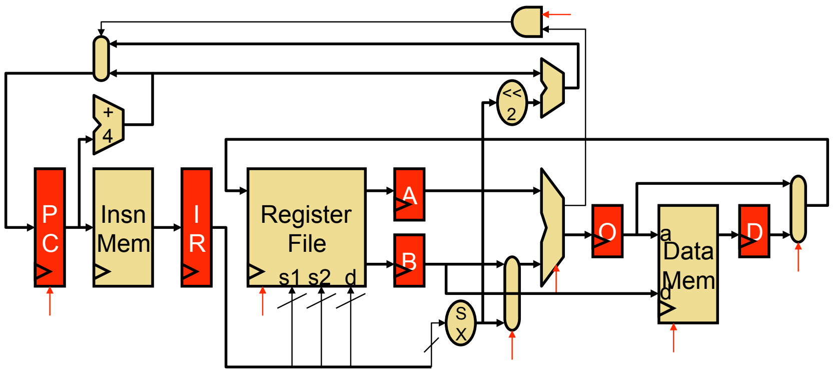
- Let's allow long-latency operations take many cycles
- Calculation assumptions:
  - Most instructions take 10 nanoseconds (ns)
  - But multiply instruction takes 40ns
  - Multiplies are 10% of all instructions
- Single-cycle datapath: 40ns clock period, 1 CPI
  - 40ns per instruction
  - 1/40ns = 0.025 billion instructions per second = **25 MIPS**
- Multi-cycle datapath: 10ns clock period
  - Average CPI = (90% * 1) + (10% * 4) = 1.3
  - 13ns per instruction
  - 1/13ns = 0.77 billion instructions per second = **77 MIPS**
- Multi-cycle is 3 times (or 200%) faster than single-cycle

Fine-Grained Multi-Cycle Datapath



- **Multi-cycle datapath**: attacks high clock period
  - Cut datapath into multiple stages (5 here), isolate using FFs
  - Finite state machine (FSM) control "walks" insns through
  + Insns can skip stages and exit early (memory ops vs alu ops)

Multi-Cycle Datapath Performance



- Opposite performance split of single-cycle datapath
  - + Short clock period
  - – High CPI

Multicycle Performance

- Assumptions
  - 30% loads, 5ns
  - 10% stores, 5ns
  - 50% adds, 4ns
  - 10% multiplies, 20ns
- Single-cycle datapath: 20ns clock period, 1 CPI
  - 20ns per instruction or **50 MIPS**
- Simple multi-cycle datapath: 5ns clock
  - CPI = (90% * 1) + (10% * 4) = 1.3
  - 6.5ns per instruction or **153 MIPS**
- Fine-grained multi-cycle datapath: 1ns clock
  - CPI = (30% * 5) + (10% * 5) + (50% * 4) + (10% * 20) = 1.5 + 0.5 + 2 + 2 = 6 CPI
  - 6ns per instruction or **166 MIPS**

Processor Performance and Workloads

- Q: what does performance of a chip mean?
- A: nothing, there must be some associated workload
  - **Workload**: set of tasks someone (you) cares about
- **Benchmarks**: standard workloads
  - Used to compare performance across machines
  - Either are or highly representative of actual programs people run
- **Micro-benchmarks**: non-standard non-workloads
  - Tiny programs used to isolate certain aspects of performance
  - Not representative of complex behaviors of real applications
  - Examples: binary tree search, towers-of-hanoi, 8-queens, etc.

SPEC Benchmarks

- SPEC: Standard Performance Evaluation Corporation
  - http://www.spec.org/
  - Consortium that collects, standardizes, and distributes benchmarks
  - Suites for CPU, Java, I/O, Web, Mail, OpenMP (multithreaded), etc.
  - Updated every few years: so companies don't target benchmarks
  - Post **SPECmark** results for different processors
    - 1 number that represents performance for entire suite
  - CPU 2006: 29 CPU-intensive C/C++/Fortran programs
    - "integer": bzip2, gcc, perl, hmmer (genomics), h264, etc.
    - "floating-point": wrf (weather), povray, sphynx3 (speech), etc.
- TPC: Transaction Processing Council
  - Like SPEC, but for web/database server workloads
  - Much heavier on memory, I/O, network, than on CPU
  - Doesn't give you the source code, only a 'description'

SPECmark

- Reference machine: Sun SPARC 10
- Latency SPECmark
  - For each benchmark
    - Take odd number of samples: on both machines
    - Choose median
    - Take latency ratio (Sun SPARC 10 / your machine)
  - Take "geometric mean" of ratios over all benchmarks
- Throughput SPECmark
  - Run multiple benchmarks in parallel on multiple-processor system

- Recent SPECmark latency leaders
  - SPECint: Intel 2.3 GHz Core2 Extreme (3119)
  - SPECfp: IBM 2.1 GHz Power5+ (4051)

Mean (Average) Performance Numbers

- **Arithmetic**: (1/N) * $\Sigma_{P=1..N}$ Latency(P)
  - For units that are proportional to time (e.g., latency)

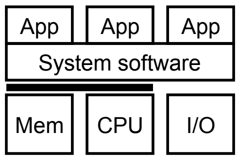
- You can add latencies, but not throughputs
  - Latency(P1+P2,A) = Latency(P1,A) + Latency(P2,A)
  - Throughput(P1+P2,A) != Throughput(P1,A) + Throughput(P2,A)
    - 1 mile @ 30 miles/hour + 1 mile @ 90 miles/hour
    - Average is **not** 60 miles/hour
- **Harmonic**: N / $\Sigma_{P=1..N}$ 1/Throughput(P)
  - For units that are inversely proportional to time (e.g., throughput)

- **Geometric**: $\sqrt[N]{\Pi_{P=1..N}}$ Speedup(P)
  - For unitless quantities (e.g., speedups)

How Can We Make Common Case Fast?

- If we don't know what CC is?
- How is CPI actually measured?
  - Execution time: time (Unix): wall clock / CPU + system
  - CPI = CPU time / (clock frequency * dynamic insn count)
- How is dynamic insn count measured?
  - Hardware event counters
- More useful is CPI breakdown (CPI_{CPU}, CPI_{MEM}, etc.)
  - So we know what performance problems are and what to fix
  - Hardware event counters:
    + Accurate
    – Can't measure everything or evaluate modifications
  - Cycle-level micro-architecture simulation: e.g., SimpleScalar
    + Measure exactly what you want, evaluate potential fixes
    – Burden of accuracy is on the simulator writer

Summary



- Multicycle datapath
- Clock vs CPI
- CPU performance equation
- Performance metrics
- Benchmarking