15-740/18-740
Computer Architecture
Lecture 6: Performance

Prof. Onur Mutlu
Carnegie Mellon University
Fall 2011, 9/23/2011

Review of Last Lectures

- ISA Tradeoffs
  - Semantic gap, instruction length, uniform decode, register count, addressing modes

- **"Row Buffer Locality-Aware Data Placement in Hybrid Memories"** SAFARI Tech Report, 2011.
  - Provide a review for next Friday

- Project discussion

Today

- More ISA tradeoffs
- Performance metrics and evaluation
- Pipelining basics

Other ISA-level Tradeoffs

- Load/store vs. Memory/Memory
- Condition codes vs. condition registers vs. compare&test
- Hardware interlocks vs. software-guaranteed interlocking
- VLIW vs. single instruction vs. SIMD
- 0, 1, 2, 3 address machines (stack, accumulator, 2 or 3-operands)
- Precise vs. imprecise exceptions
- Virtual memory vs. not
- Aligned vs. unaligned access
- Supported data types
- Software vs. hardware managed page fault handling
- Granularity of atomicity
- Cache coherence (hardware vs. software)
- …

Programmer vs. (Micro)architect

- Many ISA features designed to aid programmers
- But, complicate the hardware designer’s job

- Virtual memory
  - vs. overlay programming
  - Should the programmer be concerned about the size of code blocks?
- Unaligned memory access
  - Compile/programmer needs to align data
- Transactional memory?
- VLIW vs. SIMD? Superscalar execution vs. SIMD?

Transactional Memory

THREAD 1

```
enqueue (Q, v) {
  Node_t node = malloc(…);
  node->val = v;
  node->next = NULL;
  acquire(lock);
  if (Q->tail)
    Q->tail->next = node;
  else
    Q->head = node;
  Q->tail = node;
  release(lock);
}
```

```
begin-transaction
…
enqueue (Q, v); //no locks
…
end-transaction
```

THREAD 2

```
enqueue (Q, v) {
  Node_t node = malloc(…);
  node->val = v;
  node->next = NULL;
  acquire(lock);
  if (Q->tail)
    Q->tail->next = node;
  else
    Q->head = node;
  Q->tail = node;
  release(lock);
}
```

```
begin-transaction
…
enqueue (Q, v); //no locks
…
end-transaction
```

Transactional Memory

- A transaction is executed atomically: ALL or NONE

- If there is a data conflict between two transactions, only one of them completes; the other is rolled back
  - Both write to the same location
  - One reads from the location another writes

- Herlihy and Moss, "Transactional Memory: Architectural Support for Lock-Free Data Structures," ISCA 1993.

ISA-level Tradeoff: Supporting TM

- Still under research
- Pros:
  - Could make programming with threads easier
  - Could improve parallel program performance vs. locks. Why?

- Cons:
  - Complexity
  - What if it does not pan out?
  - All future microarchitectures might have to support the new instructions (for backward compatibility reasons)

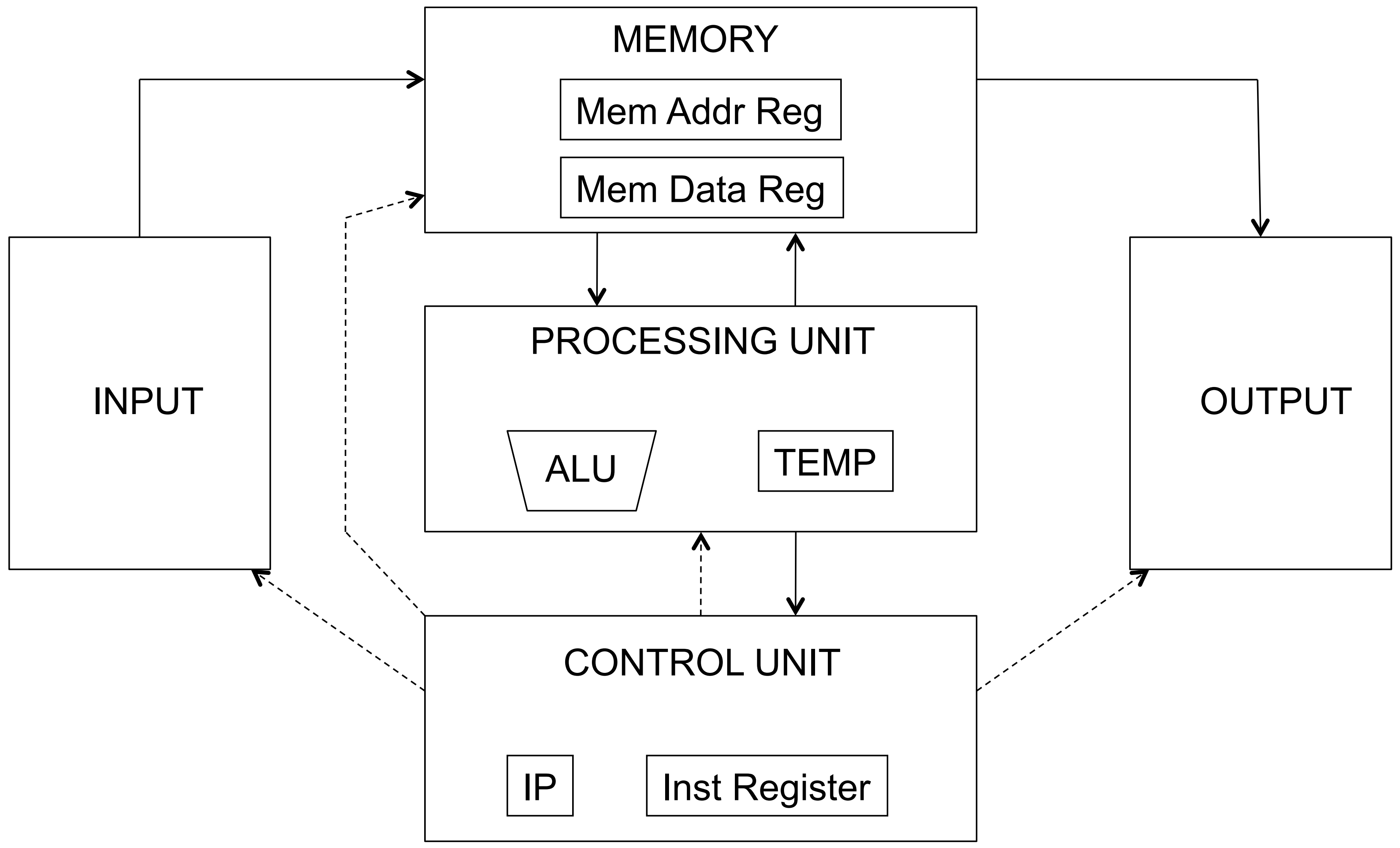
- How does the architect decide whether or not to support TM in the ISA? (How to evaluate the whole stack)

ISA-level Tradeoffs: Instruction Pointer

- Do we need an instruction pointer in the ISA?
  - Yes: Control-driven, sequential execution
    - An instruction is executed when the IP points to it
    - IP automatically changes sequentially (except control flow instructions)
  - No: Data-driven, parallel execution
    - An instruction is executed when all its operand values are available (data flow)
    - Dennis, ISCA 1974.

- Tradeoffs: MANY high-level ones
  - Ease of programming (for average programmers)?
  - Ease of compilation?
  - Performance: Extraction of parallelism?
  - Hardware complexity?

The Von-Neumann Model

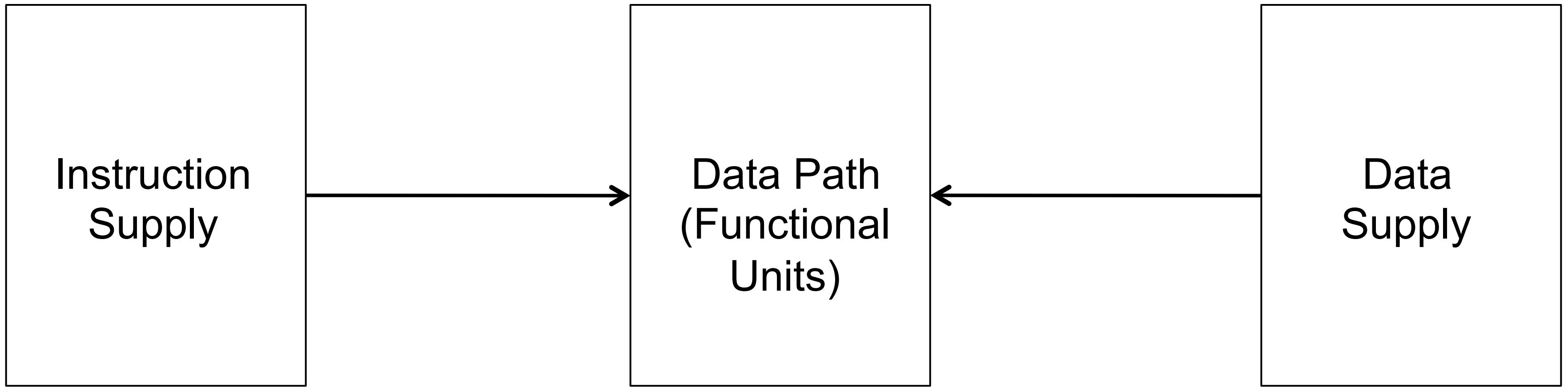


The Von-Neumann Model

- Stored program computer (instructions in memory)
- One instruction at a time
- Sequential execution
- Unified memory
  - The interpretation of a stored value depends on the control signals

- All major ISAs today use this model
- Underneath (at uarch level), the execution model is very different
  - Multiple instructions at a time
  - Out-of-order execution
  - Separate instruction and data caches

Fundamentals of Uarch Performance Tradeoffs



- Zero-cycle latency (no cache miss)
- No branch mispredicts
- No fetch breaks

- Perfect data flow (reg/memory dependencies)
- Zero-cycle interconnect (operand communication)
- Enough functional units
- Zero latency compute?

- Zero-cycle latency
- Infinite capacity
- Zero cost

We will examine all these throughout the course (especially data supply)

How to Evaluate Performance Tradeoffs

$$\text{Execution time} = \frac{\text{time}}{\text{program}}$$

$$= \frac{\text{\# instructions}}{\text{program}} \times \frac{\text{\# cycles}}{\text{instruction}} \times \frac{\text{time}}{\text{cycle}}$$

Algorithm Program ISA Compiler	ISA Microarchitecture	Microarchitecture Logic design Circuit implementation Technology

Improving Performance

- Reducing instructions/program
- Reducing cycles/instruction (CPI)
- Reducing time/cycle (clock period)

Improving Performance (Reducing Exec Time)

- Reducing instructions/program
  - More efficient algorithms and programs
  - Better ISA?

- Reducing cycles/instruction (CPI)
  - Better microarchitecture design
    - Execute multiple instructions at the same time
    - Reduce latency of instructions (1-cycle vs. 100-cycle memory access)

- Reducing time/cycle (clock period)
  - Technology scaling
  - Pipelining

Improving Performance: Semantic Gap

- Reducing instructions/program
  - Complex instructions: small code size (+)
  - Simple instructions: large code size (--)

- Reducing cycles/instruction (CPI)
  - Complex instructions: (can) take more cycles to execute (--)
    - REP MOVS
    - How about ADD with condition code setting?
  - Simple instructions: (can) take fewer cycles to execute (+)

- Reducing time/cycle (clock period)
  - Does instruction complexity affect this?
    - It depends

Other Performance Metrics: IPS

- Machine A: 10 billion instructions per second
- Machine B: 1 billion instructions per second
- Which machine has higher performance?

- Instructions Per Second (IPS, MIPS, BIPS)

$$\frac{\text{\# of instructions}}{\text{cycle}} \times \frac{\text{cycle}}{\text{time}}$$

  - How does this relate to execution time?
  - When is this a good metric for comparing two machines?
    - Same instruction set, same binary (i.e., same compiler), same operating system
    - Meaningless if "Instruction count" does not correspond to "work"
      - E.g., some optimizations add instructions, but do not change "work"

Other Performance Metrics: FLOPS

- Machine A: 10 billion FP instructions per second
- Machine B: 1 billion FP instructions per second
- Which machine has higher performance?

- Floating Point Operations per Second (FLOPS, MFLOPS, GFLOPS)
  - Popular in scientific computing
  - FP operations used to be very slow (think Amdahl's law)
- Why not a good metric?
  - Ignores all other instructions
    - what if your program has 0 FP instructions?
  - Not all FP ops are the same

Other Performance Metrics: Perf/Frequency

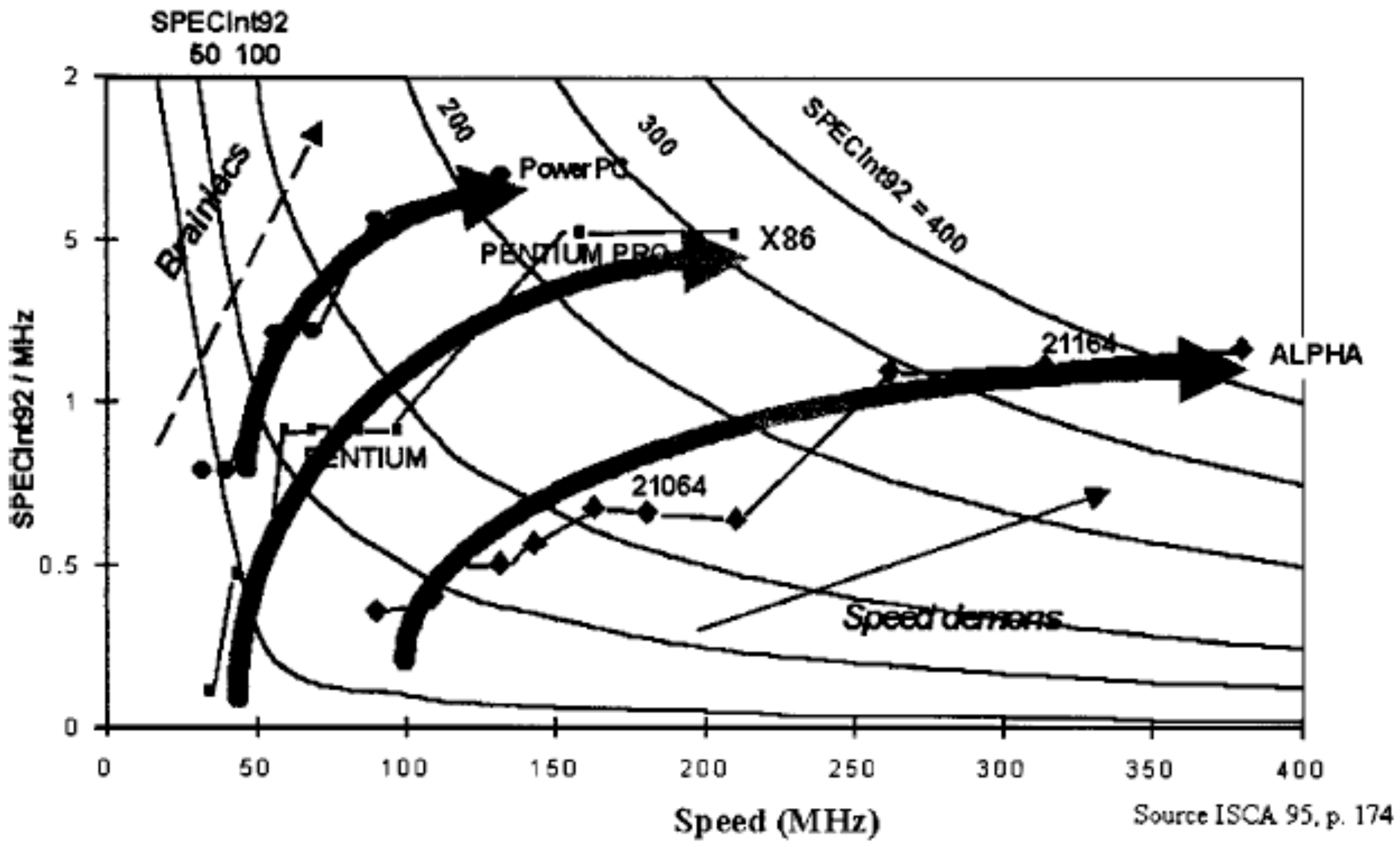
- SPEC/MHz
- Remember $\text{Execution time} = \dfrac{\text{time}}{\text{program}} = \dfrac{1}{\text{Performance}}$
- Performance/Frequency

$$= \frac{\dfrac{\text{time}}{\text{cycle}}}{\dfrac{\text{\# instructions}}{\text{program}} \times \dfrac{\text{\# cycles}}{\text{instruction}} \times \dfrac{\text{time}}{\text{cycle}}}$$

$$= 1 / \left\{ \frac{\text{\# cycles}}{\text{program}} \right\}$$

- What is wrong with comparing only “cycle count”?
  - Unfairly penalizes machines with high frequency
- For machines of equal frequency, fairly reflects performance assuming equal amount of “work” is done
  - Fair if used to compare two different same-ISA processors on the same binaries

An Example Use of Perf/Frequency Metric



- Ronen et al, IEEE Proceedings 2001