Review: Tomasulo With Reorder buffer:

- FP Op Queue
- Reorder Buffer
- Registers

Review: Branch Target Buffer (BTB)

- Address of branch index to get prediction AND branch address (if taken)
  - Must check for branch match now, since can’t use wrong branch address
  - Grab predicted PC from table since may take several cycles to compute

Review: Branch History Table

- BHT is a table of “Predictors”
  - Usually 2-bit, saturating counters
  - Indexed by PC address of Branch – without tags

- In Fetch state of branch:
  - BTB identifies branch
  - Predictor from BHT used to make prediction

- When branch completes:
  - Update corresponding Predictor
The Five Classic Components of a Computer

- Processor
- Control
- Datapath
- Memory
- Output

Today's Topics:
- Recap last lecture
- Locality and Memory Hierarchy
- Administrivia
- SRAM Memory Technology
- DRAM Memory Technology
- Memory Organization

The Big Picture: Where are We Now?

Technology Trends (from 1st lecture)

<table>
<thead>
<tr>
<th></th>
<th>Capacity</th>
<th>Speed (latency)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic</td>
<td>2x in 3 years</td>
<td>2x in 3 years</td>
</tr>
<tr>
<td>DRAM</td>
<td>4x in 3 years</td>
<td>2x in 10 years</td>
</tr>
<tr>
<td>Disk</td>
<td>4x in 3 years</td>
<td>2x in 10 years</td>
</tr>
</tbody>
</table>

Technology Trends (from 1st lecture)

<table>
<thead>
<tr>
<th></th>
<th>Year</th>
<th>Size</th>
<th>Cycle Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1980</td>
<td></td>
<td>64 Kb</td>
<td>250 ns</td>
</tr>
<tr>
<td>1983</td>
<td></td>
<td>256 Kb</td>
<td>220 ns</td>
</tr>
<tr>
<td>1986</td>
<td></td>
<td>1 Mb</td>
<td>190 ns</td>
</tr>
<tr>
<td>1989</td>
<td></td>
<td>4 Mb</td>
<td>165 ns</td>
</tr>
<tr>
<td>1992</td>
<td></td>
<td>16 Mb</td>
<td>145 ns</td>
</tr>
<tr>
<td>1995</td>
<td></td>
<td>64 Mb</td>
<td>120 ns</td>
</tr>
</tbody>
</table>

Who Cares About the Memory Hierarchy?

Processor-DRAM Memory Gap (latency)

Processor-Memory Performance Gap:
- "Moore's Law" (grows 50% per year)

"Less' Law?"

Fact:
- Large memories are slow
- Fast memories are small

How do we create a memory that is large, cheap and fast (most of the time)?
- Hierarchy
- Parallelism

The Goal: illusion of large, fast, cheap memory
Memory Hierarchy of a Modern Computer System

- By taking advantage of the principle of locality:
  - Present the user with as much memory as is available in the cheapest technology.
  - Provide access at the speed offered by the fastest technology.

Today's Situation: Microprocessor

- Rely on caches to bridge gap
- Microprocessor-DRAM performance gap
  - time of a full cache miss in instructions executed
    - 1st Alpha (7000): 340 ns / 5.0 ns = 68 clks x 2 or 136 instructions
    - 2nd Alpha (8400): 266 ns / 3.3 ns = 80 clks x 4 or 320 instructions
    - 3rd Alpha (t.b.d.): 180 ns / 1.7 ns = 108 clks x 6 or 648 instructions
  - 1/2X latency x 3X clock rate x 3X Instr/clock \( \Rightarrow -5X \)

Example: 1 KB Direct Mapped Cache with 32 B Blocks

- For a \( 2^{N} \) byte cache:
  - The uppermost (32 - N) bits are always the Cache Tag
  - The lowest M bits are the Byte Select (Block Size = \( 2^M \))

- Probability of reference
- Temporal Locality (Locality in Time):
  - Keep most recently accessed data items closer to the processor
- Spatial Locality (Locality in Space):
  - Move blocks consists of contiguous words to the upper levels
Example: Set Associative Cache

- **N-way set associative**: N entries for each Cache Index
  - N direct mapped caches operate in parallel

- **Example: Two-way set associative cache**
  - Cache Index selects a "set" from the cache
  - The two tags in the set are compared to the input in parallel
  - Data is selected based on the tag result

Memory Hierarchy: Terminology

- **Hit**: data appears in some block in the upper level (example: Block X)
  - Hit Rate: the fraction of memory access found in the upper level
  - Hit Time: Time to access the upper level which consists of RAM access time + Time to determine hit/miss

- **Miss**: data needs to be retrieved from a block in the lower level (Block Y)
  - Miss Rate = 1 - (Hit Rate)
  - Miss Penalty: Time to replace a block in the upper level + Time to deliver the block to the processor

- **Hit Time << Miss Penalty**

Recap: Cache Performance

- CPU time = (CPU execution clock cycles + Memory stall clock cycles) x clock cycle time

- Memory stall clock cycles =
  - (Reads x Read miss rate x Read miss penalty + Writes x Write miss rate x Write miss penalty)

- Memory stall clock cycles = Memory accesses x Miss rate x Miss penalty

- Different measure: AMAT
  - Average Memory Access time (AMAT) = Hit Time + (Miss Rate x Miss Penalty)

- Note: *memory hit time is included in execution cycles.*

Recap: Impact on Performance

- Suppose a processor executes at
  - Clock Rate = 200 MHz (5 ns per cycle)
  - Base CPI = 1.1
  - 50% arith/logic, 30% ld/st, 20% control

- Suppose that 10% of memory operations get 50 cycle miss penalty

- Suppose that 1% of instructions get same miss penalty

- CPI = Base CPI + average stalls per instruction
  
  1.1(cycles/ins) + [0.30 (DataMops/ins) x 0.10 (miss/DataMop) x 50 (cycle/miss)] + [1 (InstMop/ins) x 0.01 (miss/InstMop) x 50 (cycle/miss)]

  = (1.1 + 1.5 + .5) cycle/ins = 3.1

- 58% of the time the processor is stalled waiting for memory!

- AMAT=(1/1.3)x[1+0.01x50]+(0.3/1.3)x[1+0.1x50]=2.54
How is the hierarchy managed?

- Registers <-> Memory
  - by compiler (programmer?)
- cache <-> memory
  - by the hardware
- memory <-> disks
  - by the hardware and operating system (virtual memory)
  - by the programmer (files)

Memory Hierarchy Technology

- Random Access:
  - “Random” is good: access time is the same for all locations
  - DRAM: Dynamic Random Access Memory
    - High density, low power, cheap, slow
    - Dynamic: need to be “refreshed” regularly
  - SRAM: Static Random Access Memory
    - Low density, high power, expensive, fast
    - Static: content will last “forever” (until lose power)

- “Non-so-random” Access Technology:
  - Access time varies from location to location and from time to time
  - Examples: Disk, CDROM, DRAM page-mode access

- Sequential Access Technology: access time linear in location (e.g., Tape)

The next two lectures will concentrate on random access technology

- The Main Memory: DRAMs + Caches: SRAMs

Main Memory Background

- Performance of Main Memory:
  - Latency: Cache Miss Penalty
    - Access Time: time between request and word arrives
    - Cycle Time: time between requests
  - Bandwidth: I/O & Large Block Miss Penalty (L2)

- Main Memory is DRAM: Dynamic Random Access Memory
  - Dynamic since needs to be refreshed periodically (8 ms)
  - Addresses divided into 2 halves (Memory as a 2D matrix):
    - RAS or Row Access Strobe
    - CAS or Column Access Strobe

- Cache uses SRAM: Static Random Access Memory
  - No refresh (6 transistors/bit vs. 1 transistor)
  - Size: DRAM-SRAM - 4-8
  - Cost/Cycle time: SRAM/DRAM - 8-16

Random Access Memory (RAM) Technology

- Why do computer designers need to know about RAM technology?
  - Processor performance is usually limited by memory bandwidth
  - As IC densities increase, lots of memory will fit on processor chip
    - Tailor on-chip memory to specific needs
      - Instruction cache
      - Data cache
      - Write buffer

- What makes RAM different from a bunch of flip-flops?
  - Density: RAM is much denser
**Static RAM Cell**

6-Transistor SRAM Cell

- **Write:**
  1. Drive bit lines (bit=1, bit=0)
  2. Select row
- **Read:**
  1. Precharge bit and bit to Vdd or Vdd/2 => make sure equal!
  2. Select row
  3. Cell pulls one line low
  4. Sense amp on column detects difference between bit and bit

**Typical SRAM Organization: 16-word x 4-bit**

**Logic Diagram of a Typical SRAM**

- Write Enable is usually active low (WE_L)
- Din and Dout are combined to save pins:
  - A new control signal, output enable (OE_L) is needed
  - WE_L is asserted (Low), OE_L is disasserted (High)
    - D serves as the data input pin
  - WE_L is disasserted (High), OE_L is asserted (Low)
    - D is the data output pin
  - Both WE_L and OE_L are asserted:
    - Result is unknown. Don’t do that!!!
- Although could change VHDL to do what desire, must do the best with what you’ve got (vs. what you need)

**Typical SRAM Timing**

- Write Timing:
- Read Timing:
  - Read Access Time
  - Data Out
  - Read Address
  - Read Access Time
Problems with SRAM

- Six transistors use up a lot of area
- Consider a “Zero” is stored in the cell:
  - Transistor N1 will try to pull “bit” to 0
  - Transistor P2 will try to pull “bit bar” to 1
- But bit lines are precharged to high: Are P1 and P2 necessary?

1-Transistor Memory Cell (DRAM)

- Write:
  - 1. Drive bit line
  - 2. Select row
- Read:
  - 1. Precharge bit line to Vdd/2
  - 2. Select row
  - 3. Cell and bit line share charges
    - Very small voltage changes on the bit line
  - 4. Sense (fancy sense amp)
    - Can detect changes of ~1 million electrons
  - 5. Write: restore the value
- Refresh
  - 1. Just do a dummy read to every cell.

Classical DRAM Organization (square)

- Row and Column Address together:
  - Select 1 bit a time

DRAM logical organization (1 Mbit)

- Square root of bits per RAS/CAS
Control Signals (RAS_L, CAS_L, WE_L, OE_L) are all active low

Din and Dout are combined (D):
- WE_L is asserted (Low), OE_L is disasserted (High)
  - D serves as the data input pin
- WE_L is disasserted (High), OE_L is asserted (Low)
  - D is the data output pin

Row and column addresses share the same pins (A)
- RAS_L goes low: Pins A are latched in as row address
- CAS_L goes low: Pins A are latched in as column address
- RAS/CAS edge-sensitive

Every DRAM access begins at:
- The assertion of the RAS_L
- 2 ways to read: early or late v. CAS

Early Read Cycle: OE_L asserted before CAS_L
Late Read Cycle: OE_L asserted after CAS_L

Every DRAM access begins at:
- The assertion of the RAS_L
- 2 ways to write: early or late v. CAS

Early Wr Cycle: WE_L asserted before CAS_L
Late Wr Cycle: WE_L asserted after CAS_L
Key DRAM Timing Parameters

- \( t_{RAC} \): minimum time from RAS line falling to the valid data output.
  - Quoted as the speed of a DRAM
  - A fast 4Mb DRAM \( t_{RAC} = 60 \) ns

- \( t_{RC} \): minimum time from the start of one row access to the start of the next.
  - \( t_{RC} = 110 \) ns for a 4Mbit DRAM with a \( t_{RAC} \) of 60 ns

- \( t_{CAC} \): minimum time from CAS line falling to valid data output.
  - 15 ns for a 4Mbit DRAM with a \( t_{RAC} \) of 60 ns

- \( t_{PC} \): minimum time from the start of one column access to the start of the next.
  - 35 ns for a 4Mbit DRAM with a \( t_{RAC} \) of 60 ns

DRAM Performance

- A 60 ns \( t_{RAC} \) DRAM can
  - perform a row access only every 110 ns \( t_{RC} \)
  - perform column access \( t_{CAC} \) in 15 ns, but time between column accesses is at least 35 ns \( t_{PC} \).
    - In practice, external address delays and turning around buses make it 40 to 50 ns

- These times do not include the time to drive the addresses off the microprocessor nor the memory controller overhead.
  - Drive parallel DRAMs, external memory controller, bus to turn around, SIMM module, pins...
  - 180 ns to 250 ns latency from processor to memory is good for a “60 ns” \( t_{RAC} \) DRAM

Main Memory Performance

- **Simple**:
  - CPU, Cache, Bus, Memory same width (32 bits)

- **Wide**:
  - CPU/Mux 1 word; Mux/Cache, Bus, Memory N words (Alpha: 64 bits & 256 bits)

- **Interleaved**:
  - CPU, Cache, Bus 1 word; Memory N Modules (4 Modules); example is word interleaved

- DRAM (Read/Write) Cycle Time >> DRAM (Read/Write) Access Time
  - 2:1; why?

- DRAM (Read/Write) Cycle Time:
  - How frequent can you initiate an access?
  - Analogy: A little kid can only ask his father for money on Saturday

- DRAM (Read/Write) Access Time:
  - How quickly will you get what you want once you initiate an access?
  - Analogy: As soon as he asks, his father will give him the money

- DRAM Bandwidth Limitation analogy:
  - What happens if he runs out of money on Wednesday?
### Increasing Bandwidth - Interleaving

**Access Pattern without Interleaving:**
- Start Access for D1
- D1 available
- Start Access for D2

**Access Pattern with 4-way Interleaving:**
- Access Bank 0
- Access Bank 1
- Access Bank 2
- Access Bank 3
- We can Access Bank 0 again

### Main Memory Performance

**Timing model**
- 1 to send address,
- 4 for access time, 10 cycle time, 1 to send data
- Cache Block is 4 words
- **Simple M.P.** = 4 x (1+10+1) = 48
- **Wide M.P.** = 1 + 10 + 1 = 12
- **Interleaved M.P.** = 1+10+1 + 3 = 15

### How many banks?

- Number banks \( \geq \) number clocks to access word in bank
  - For sequential accesses, otherwise will return to original bank before it has next word ready

- **Increasing DRAM** \( \Rightarrow \) fewer chips \( \Rightarrow \) harder to have banks
  - Growth bits/chip DRAM : 50%-60%/yr
  - Nathan Myrvold M/S: mature software growth (33%/yr for NT) - growth MB/3 of DRAM (25%-30%/yr)

### Fewer DRAMs/System over Time

(from Pete MacWilliams, Intel)

<table>
<thead>
<tr>
<th>Year</th>
<th>DRAM Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>'86</td>
<td>4 MB</td>
</tr>
<tr>
<td>'89</td>
<td>16 MB</td>
</tr>
<tr>
<td>'92</td>
<td>64 MB</td>
</tr>
<tr>
<td>'96</td>
<td>256 MB</td>
</tr>
<tr>
<td>'99</td>
<td>1 Gb</td>
</tr>
<tr>
<td>'02</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory per DRAM growth @ 60% / year</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 MB</td>
</tr>
<tr>
<td>8 MB</td>
</tr>
<tr>
<td>16 MB</td>
</tr>
<tr>
<td>32 MB</td>
</tr>
<tr>
<td>64 MB</td>
</tr>
<tr>
<td>128 MB @ 25%-30% / year</td>
</tr>
<tr>
<td>256 MB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Minimum PC Memory Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 MB</td>
</tr>
<tr>
<td>8 MB</td>
</tr>
<tr>
<td>16 MB</td>
</tr>
<tr>
<td>32 MB</td>
</tr>
<tr>
<td>64 MB</td>
</tr>
<tr>
<td>128 MB</td>
</tr>
<tr>
<td>256 MB</td>
</tr>
</tbody>
</table>

Lec19.37

Lec19.38

Lec19.39

Lec19.40
Fast Page Mode Operation

° Regular DRAM Organization:
- N rows x N columns x M-bit
- Read & Write M-bit at a time
- Each M-bit access requires a RAS / CAS cycle

° Fast Page Mode DRAM:
- N x M "SRAM" to save a row

° After a row is read into the register:
- Only CAS is needed to access other M-bit blocks on that row
- RAS_L remains asserted while CAS_L is toggled

Key DRAM Timing Parameters

° t_RAC: minimum time from RAS line falling to the valid data output.
  - Quoted as the speed of a DRAM
  - A fast 4Mb DRAM t_RAC = 60 ns

° t_RC: minimum time from the start of one row access to the start of the next.
  - t_RC = 110 ns for a 4Mb DRAM with a t_RAC of 60 ns

° t_CAC: minimum time from CAS line falling to valid data output.
  - 15 ns for a 4Mb DRAM with a t_RAC of 60 ns

° t_PC: minimum time from the start of one column access to the start of the next.
  - 35 ns for a 4Mb DRAM with a t_RAC of 60 ns

DRAMs over Time

<table>
<thead>
<tr>
<th>DRAM Generation</th>
<th>1st Gen. Sample</th>
<th>’84</th>
<th>’87</th>
<th>’90</th>
<th>’93</th>
<th>’96</th>
<th>’99</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Size</td>
<td>1 Mb</td>
<td>4 Mb</td>
<td>16 Mb</td>
<td>64 Mb</td>
<td>256 Mb</td>
<td>1 Gb</td>
<td></td>
</tr>
<tr>
<td>Die Size (mm^2)</td>
<td>55</td>
<td>85</td>
<td>130</td>
<td>200</td>
<td>300</td>
<td>450</td>
<td></td>
</tr>
<tr>
<td>Memory Area (mm^2)</td>
<td>30</td>
<td>47</td>
<td>72</td>
<td>110</td>
<td>165</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Memory Cell Area (µm^2)</td>
<td>28.84</td>
<td>11.1</td>
<td>4.26</td>
<td>1.64</td>
<td>0.61</td>
<td>0.23</td>
<td></td>
</tr>
</tbody>
</table>

(from Kazuhiro Sakashita, Mitsubishi)

DRAM History

° DRAMs: capacity +60%/yr, cost ~30%/yr
  - 2.5X cells/area, 1.5X die size in ~3 years
° ’97 DRAM fab line costs $1B to $2B
  - DRAM only: density, leakage v. speed
° Rely on increasing no. of computers & memory per computer (60% market)
  - SIMM or DIMM is replaceable unit
  - computers use any generation DRAM
° Commodity, second source industry
  - high volume, low profit, conservative
  - Little organization innovation in 20 years
  - page mode, EDO, Synch DRAM
° Order of importance: 1) Cost/bit 1a) Capacity
  - RAMBUS: 10X BW, +30% cost => little impact
**DRAM v. Desktop Microprocessors Cultures**

<table>
<thead>
<tr>
<th>Standards</th>
<th>pinout, package, refresh rate, capacity, ...</th>
<th>binary compatibility, IEEE 754, I/O bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sources</td>
<td>Multiple</td>
<td>Single</td>
</tr>
<tr>
<td>Figures of Merit</td>
<td>1) capacity, 1a) $/bit</td>
<td>1) SPEC speed</td>
</tr>
<tr>
<td></td>
<td>2) BW, 3) latency</td>
<td>2) cost</td>
</tr>
<tr>
<td>Improve Rate/year</td>
<td>1) 60%, 1a) 25%,</td>
<td>1) 60%</td>
</tr>
<tr>
<td></td>
<td>2) 20%, 3) 7%</td>
<td>2) little change</td>
</tr>
</tbody>
</table>

**DRAM Design Goals**

° Reduce cell size 2.5, increase die size 1.5
° Sell 10% of a single DRAM generation
  • 6.25 billion DRAMs sold in 1996
° 3 phases: engineering samples, first customer ship (FCS), mass production
  • Fastest to FCS, mass production wins share
° Die size, testing time, yield => profit
  • Yield >> 60%
  (redundant rows/columns to repair flaws)

° DRAMs: capacity +60%/yr, cost –30%/yr
  • 2.5X cells/area, 1.5X die size in 3 years
° ’97 DRAM fab line costs $1B to $2B
  • DRAM only: density, leakage v. speed
° Rely on increasing no. of computers & memory per computer (60% market)
  • SIMM or DIMM is replaceable unit
  => computers use any generation DRAM
° Commodity, second source industry => high volume, low profit, conservative
  • Little organization innovation (vs. processors) in 20 years: page mode, EDO, Synch DRAM
° Order of importance: 1) Cost/bit 1a) Capacity
  • RAMBUS: 10X BW, +30% cost => little impact

° Commodity, second source industry => high volume, low profit, conservative
  • Little organization innovation (vs. processors)
  in 20 years: page mode, EDO, Synch DRAM
° DRAM industry at a crossroads:
  • Fewer DRAMs per computer over time
    • Growth bits/chip DRAM : 50%-60%/yr
    • Nathan Myrvold M/S: mature software growth
      (33%/yr for NT) - growth MB$/ of DRAM (25%-30%/yr)
  • Starting to question buying larger DRAMs?
Today's Situation: DRAM

- Intel: 30%/year since 1987; 1/3 income profit

Summary:

- Two Different Types of Locality:
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon.
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon.

- By taking advantage of the principle of locality:
  - Present the user with as much memory as is available in the cheapest technology.
  - Provide access at the speed offered by the fastest technology.

- DRAM is slow but cheap and dense:
  - Good choice for presenting the user with a BIG memory system

- SRAM is fast but expensive and not very dense:
  - Good choice for providing the user FAST access time.

---

### Summary: Processor-Memory Performance Gap “Tax”

<table>
<thead>
<tr>
<th>Processor</th>
<th>% Area</th>
<th>% Transistors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha 21164</td>
<td>37%</td>
<td>77%</td>
</tr>
<tr>
<td>StrongArm SA110</td>
<td>61%</td>
<td>94%</td>
</tr>
<tr>
<td>Pentium Pro</td>
<td>64%</td>
<td>88%</td>
</tr>
</tbody>
</table>

- 2 dies per package: Proc/ISA/D$ + L2S
- Caches have no inherent value, only try to close performance gap