Last time in Lecture 5

- Control hazards (branches, interrupts) are most difficult to handle as they change which instruction should be executed next
- Speculation commonly used to reduce effect of control hazards (predict sequential fetch, predict no exceptions)
- Branch delay slots make control hazard visible to software
- Precise exceptions: stop cleanly on one instruction, all previous instructions completed, no following instructions have changed architectural state
- To implement precise exceptions in pipeline, shift faulting instructions down pipeline to “commit” point, where exceptions are handled in program order
CPU-Memory Bottleneck

Performance of high-speed computers is usually limited by memory bandwidth & latency

- Latency (time for a single access)
  Memory access time >> Processor cycle time

- Bandwidth (number of accesses per unit time)
  if fraction $m$ of instructions access memory,
  $\Rightarrow 1+m$ memory references / instruction
  $\Rightarrow$ CPI = 1 requires $1+m$ memory refs / cycle
  (assuming MIPS RISC ISA)

Core Memory

- Core memory was first large scale reliable main memory
  - invented by Forrester in late 40s/early 50s at MIT for Whirlwind project

- Bits stored as magnetization polarity on small ferrite cores threaded onto 2 dimensional grid of wires

- Coincident current pulses on X and Y wires would write cell and also sense original state (destructive reads)

- Robust, non-volatile storage

- Used on space shuttle computers until recently

- Cores threaded onto wires by hand (25 billion a year at peak production)

- Core access time ~ $1\mu$s

DEC PDP-8/E Board, 4K words x 12 bits, (1968)
Semiconductor Memory, DRAM

- Semiconductor memory began to be competitive in early 1970s
  - Intel formed to exploit market for semiconductor memory

- First commercial DRAM was Intel 1103
  - 1Kbit of storage on single chip
  - charge on a capacitor used to hold value

- Semiconductor memory quickly replaced core in ‘70s

One Transistor Dynamic RAM

![Diagram of 1-T DRAM Cell]

- 1-T DRAM Cell
- word
- access transistor
- bit
- Storage capacitor (FET gate, trench, stack)
- TiN top electrode ($V_{REF}$)
- Ta$_2$O$_5$ dielectric
- W bottom electrode
- poly word line
- access transistor

![SEM image of 1-T DRAM Cell]
**DRAM Architecture**

- Bits stored in 2-dimensional arrays on chip
- Modern chips have around 4 logical banks on each chip
  - each logical bank physically implemented as many smaller arrays

**DRAM Packaging**

- DIMM (Dual Inline Memory Module) contains multiple chips with clock/control/address signals connected in parallel (sometimes need buffers to drive signals to all chips)
- Data pins work together to return wide word (e.g., 64-bit data bus using 16x4-bit parts)
DRAM Operation

Three steps in read/write access to a given bank

- **Row access (RAS)**
  - decode row address, enable addressed row (often multiple Kb in row)
  - bitlines share charge with storage cell
  - small change in voltage detected by sense amplifiers which latch whole row of bits
  - sense amplifiers drive bitlines full rail to recharge storage cells

- **Column access (CAS)**
  - decode column address to select small number of sense amplifier latches (4, 8, 16, or 32 bits depending on DRAM package)
  - on read, send latched bits out to chip pins
  - on write, change sense amplifier latches which then charge storage cells to required value
  - can perform multiple column accesses on same row without another row access (burst mode)

- **Precharge**
  - charges bit lines to known value, required before next row access

Each step has a latency of around 15-20ns in modern DRAMs

Various DRAM standards (DDR, RDRAM) have different ways of encoding the signals for transmission to the DRAM, but all share same core architecture

Double-Data Rate (DDR2) DRAM

200MHz Clock

[ Micron, 256Mb DDR2 SDRAM datasheet ]
Four-issue 2GHz superscalar accessing 100ns DRAM could execute 800 instructions during time for one memory access!

Typical Memory Reference Patterns

- Instruction fetches
- Stack accesses
- Data accesses
- n loop iterations
- subroutine call
- subroutine return
- argument access
- vector access
- scalar accesses
Common Predictable Patterns

Two predictable properties of memory references:

- **Temporal Locality**: If a location is referenced it is likely to be referenced again in the near future.

- **Spatial Locality**: If a location is referenced it is likely that locations near it will be referenced in the near future.

Memory Reference Patterns

**Multilevel Memory**

Strategy: **Reduce** average latency using small, fast memories called caches.

Caches are a mechanism to reduce memory latency based on the empirical observation that the patterns of memory references made by a processor are often highly predictable:

```
PC
...
loop: ADD r2, r1, r1 100
      SUBI r3, r3, #1 104
      BNEZ r3, loop 108
...
```

**Memory Hierarchy**

- **capacity:** Register << SRAM << DRAM  *why?*
- **latency:** Register << SRAM << DRAM  *why?*
- **bandwidth:**  on-chip >> off-chip  *why?*

On a data access:

- **hit** (data ∈ fast memory)  ⇒ low latency access
- **miss** (data ∉ fast memory)  ⇒ long latency access (DRAM)
Relative Memory Cell Sizes

- On-Chip SRAM in logic chip
- DRAM on memory chip

[ Foss, “Implementing Application-Specific Memory”, ISSCC 1996 ]

Management of Memory Hierarchy

- Small/fast storage, e.g., registers
  - Address usually specified in instruction
  - Generally implemented directly as a register file
    » but hardware might do things behind software’s back, e.g., stack management, register renaming

- Large/slower storage, e.g., memory
  - Address usually computed from values in register
  - Generally implemented as a cache hierarchy
    » hardware decides what is kept in fast memory
    » but software may provide “hints”, e.g., don’t cache or prefetch
CS152 Administrivia

• Quiz 1 Thursday in class (306 Soda)
  – Lectures 1-5, closed book, no calculators or computers

• Krste, special office hours, Wednesday 2/11, 2-3pm, 579 Soda Hall (Par Lab)
• Scott special office hours, Wednesday 2/11, 4-5pm, 711 Soda Hall
• Next week lecture 2/17 back in 320 Soda

Caches

Caches exploit both types of predictability:

– Exploit temporal locality by remembering the contents of recently accessed locations.

– Exploit spatial locality by fetching blocks of data around recently accessed locations.
Inside a Cache

Cache Algorithm (Read)

Look at Processor Address, search cache tags to find match. Then either

- Found in cache a.k.a. HIT
  - Return copy of data from cache

- Not in cache a.k.a. MISS
  - Read block of data from Main Memory
  - Wait ...
  - Return data to processor and update cache

Q: Which line do we replace?
### Placement Policy

- **Cache**:
  - Fully Associative: anywhere
  - (2-way) Set Associative: anywhere in set 0
  - Mapped: only into block 4

- **Memory**:


```
0 1 2 3 4 5 6 7 8 9
1 1 1 1 1 1 1 1 1 1
0 1 2 3 4 5 6 7 8 9
2 2 2 2 2 2 2 2 2 2
0 1 2 3 4 5 6 7 8 9
3 3 0 1
```

- **Set Number**
  - 0 1 2 3

- **Block Number**
  - 0 1 2 3 4 5 6 7 8 9

- block 12 can be placed only into block 4 (12 mod 8)

### Direct-Mapped Cache

- **Tag**: t
- **Index**: k
- **Block Offset**: b

- 2^k lines

- HIT

- Data Word or Byte
Direct Map Address Selection

*higher-order vs. lower-order address bits*

\[ \text{Index} \quad \text{Tag} \quad \text{Block Offset} \]

\[ V \quad \text{Tag} \quad \text{Data Block} \]

\[ 2^k \text{ lines} \]

HIT

Data Word or Byte

---

2-Way Set-Associative Cache

\[ \text{Tag} \quad \text{Index} \quad \text{Block Offset} \]

\[ V \quad \text{Tag} \quad \text{Data Block} \]

\[ V \quad \text{Tag} \quad \text{Data Block} \]

HIT

Data Word or Byte
Replacement Policy

In an associative cache, which block from a set should be evicted when the set becomes full?

- Random
- Least Recently Used (LRU)
  - LRU cache state must be updated on every access
  - true implementation only feasible for small sets (2-way)
  - pseudo-LRU binary tree often used for 4-8 way
- First In, First Out (FIFO) a.k.a. Round-Robin
  - used in highly associative caches
- Not Least Recently Used (NLRU)
  - FIFO with exception for most recently used block or blocks

*This is a second-order effect. Why?*

Replacement only happens on misses
Acknowledgements

- These slides contain material developed and copyrighted by:
  - Arvind (MIT)
  - Krste Asanovic (MIT/UCB)
  - Joel Emer (Intel/MIT)
  - James Hoe (CMU)
  - John Kubiatowicz (UCB)
  - David Patterson (UCB)

- MIT material derived from course 6.823
- UCB material derived from course CS252