Lecture 10: Memory System
-- Memory Technology

CSE 564 Computer Architecture Summer 2017

Department of Computer Science and Engineering
Yonghong Yan
yan@oakland.edu
www.secs.oakland.edu/~yan
Topics for Memory Systems

• Memory Technology and Metrics
  – SRAM, DRAM, Flash/SSD, 3-D Stack Memory, Phase-change memory
  – Latency and Bandwidth, Error Correction
  – Memory wall

• Cache
  – Cache basics
  – Cache performance and optimization
  – Advanced optimization
  – Multiple-level cache, shared and private cache, prefetching

• Virtual Memory
  – Protection, Virtualization, and Relocation
  – Page/segment, protection
  – Address Translation and TLB
Topics for Memory Systems

• Parallelism (to be discussed in TLP)
  – Memory Consistency model
    • Instructions for fence, etc
  – Cache coherence
  – NUMA and first touch
  – Transactional memory (Not covered)

• Implementation – (Not Covered)
  – Software/Hardware interface
  – Cache/memory controller
  – Bus systems and interconnect
Acknowledgement

• Based on slides prepared by: Professor David A. Patterson Computer Science 252, Fall 1996, and edited and presented by Prof. Kurt Keutzer for 2000 from UCB

• Some slides are adapted from the textbook slides for Computer Organization and Design, Fifth Edition: The Hardware/Software Interface
The Big Picture: Where are We Now?

- Memory system
  - Supplying data on time for computation
  - Term memory include circuits for storing data
    - Cache (SRAM)
    - Scratchpad (SRAM)
    - RAM (DRAM)
    - etc
## Technology Trends

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

### DRAM

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

1000:1! 2:1!
Who Cares About the Memory Hierarchy?

Processor-DRAM Memory Gap (latency) → Memory Wall

“Moore’s Law”

- Processor 60%/yr. (2X/1.5yr)
- Memory 9%/yr. (2X/10 yrs)

Processor-Memory Performance Gap: (grows 50% / year)


Time

Performance

Who Cares About the Memory Hierarchy?
The 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 ⇒ -5X
The Goal: illusion of large, fast, cheap memory

- 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
An Expanded View of the Memory System

Speed: Fastest
Size: Smallest
Cost: Highest

Fastest
Smallest
Highest

Slowest
Biggest
Lowest

Diagram showing the expanded view of the memory system with Processor, Control, Datapath, Memory, and Memory blocks.
Why hierarchy works

• The Principle of Locality:
  – Program access a relatively small portion of the address space at any instant of time.

![Graph showing probability of reference over address space]
Typical Memory Reference Patterns

Address

Instruction fetches

Stack accesses

Data accesses

n loop iterations

subroutine call

argument access

subroutine return

vector access

scalar accesses

Time
Locality

• Principle of Locality:
  – Programs tend to reuse data and instructions near those they have used recently, or that were recently referenced themselves
  – Spatial locality: Items with nearby addresses tend to be referenced close together in time
  – Temporal locality: Recently referenced items are likely to be referenced in the near future

Locality Example:
  • Data
    – Reference array elements in succession (stride-1 reference pattern): Spatial locality
    – Reference sum each iteration: Temporal locality
  • Instructions
    – Reference instructions in sequence: Spatial locality
    – Cycle through loop repeatedly: Temporal locality

```c
int sum = 0;
for (int i = 0; i < n; i++)
  sum += a[i];
return sum;
```
Locality Example

• **Claim:** Being able to look at code and get qualitative sense of its locality is key skill for professional programmer

• **Question:** Does this function have good locality?

```c
int sumarrayrows(int a[M][N])
{
    int i, j, sum = 0;
    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```
Locality Example

- **Question:** Does this function have good locality?

```c
int sumarraycols(int a[M][N])
{
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];

    return sum;
}
```
Locality Example

- **Question:** Can you permute the loops so that the function scans the 3-d array \(a[]\) with a stride-1 reference pattern (and thus has good spatial locality)?

```c
int sumarray3d(int a[M][N][N])
{
    int i, j, k, sum = 0;

    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++)
            for (k = 0; k < M; k++)
                sum += a[k][i][j];

    return sum;
}
```
Memory Hierarchy of a 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.
Memory Hierarchy in Real

(a) Memory hierarchy for server
- Register reference
- Level 1 Cache reference
- Level 2 Cache reference
- Level 3 Cache reference
- Memory reference
- I/O bus
- Disk storage
- Disk memory reference

(b) Memory hierarchy for a personal mobile device
- CPU
- Registers
- Register reference
- Level 1 Cache reference
- Level 2 Cache reference
- Memory reference
- Storage
- Flash memory reference

Size:
- Register reference: 1000 bytes
- Level 1 Cache: 64 KB
- Level 2 Cache: 256 KB
- Level 3 Cache: 4–16 GB
- Memory reference: 4–16 TB

Speed:
- Register: 300 ps
- Level 1: 1 ns
- Level 2: 3–10 ns
- Level 3: 10–20 ns
- Memory: 50–100 ns
- Disk: 5–10 ms

Size:
- CPU Registers: 500 bytes
- Level 1 Cache: 64 KB
- Level 2 Cache: 256 KB
- Memory: 256–512 MB
- Storage: 4–8 GB

Speed:
- CPU Registers: 500 ps
- Level 1: 2 ns
- Level 2: 10–20 ns
- Memory: 50–100 ns
- Storage: 25–50 us
Intel i7 (Nahelem)

- Private L1 and L2
  - L2 is 256KB each. 10 cycle latency

- 8MB shared L3. ~40 cycles latency
Area
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)

- Virtual memory
  - Virtual layer between application address space to physical memory
  - Not part of the physical memory hierarchy
Technology vs Architectures

- Technology determines the raw speed
  - Latency
  - Bandwidth
  - It is material science

- Architecture put them together with processors
  - So physical speed can be achieved in real
Memory Technology

- **Static RAM (SRAM)**
  - 0.5ns – 2.5ns, $2000 – $5000 per GB

- **Dynamic RAM (DRAM)**
  - 50ns – 70ns, $20 – $75 per GB

- **3-D stack memory**
- **Solid state disk**

- **Magnetic disk**
  - 5ms – 20ms, $0.20 – $2 per GB

**Ideal memory:**
- Access time of SRAM
- Capacity and cost/GB of disk
Memory 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

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

### 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

#### 6-Transistor SRAM Cell

- Replaced with pullup to save area
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. 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 (4 Mbit)

Square root of bits per RAS/CAS
Main Memory Performance

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

- 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
- 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

D1 available
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
Independent Memory Banks

• How many banks?
  number banks  number clocks to access word in bank
  – For sequential accesses, otherwise will return to original bank before it has next word ready

• Increasing DRAM => fewer chips => harder to have banks
  – 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)
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
Advanced DRAM Organization

• Bits in a DRAM are organized as a rectangular array
  – DRAM accesses an entire row
  – Burst mode: supply successive words from a row with reduced latency

• Double data rate (DDR) DRAM
  – Transfer on rising and falling clock edges

• Quad data rate (QDR) DRAM
  – Separate DDR inputs and outputs
3-D Stack Memory

• High Bandwidth Memory
• Hybrid Memory Cube

Unveiling Details of Knights Landing
(Next Generation Intel® Xeon Phi™ Products)

Platform Memory: DDR4 Bandwidth and Capacity Comparable to Intel® Xeon® Processors

Compute: Energy-efficient IA cores
  • Microarchitecture enhanced for HPC
  • 3X Single Thread Performance vs Knights Corner
  • Intel Xeon Processor Binary Compatible

On-Package Memory:
  • up to 16GB at launch
  • 5X Bandwidth vs DDR4

Jointly Developed with Micron Technology

2nd half ’15
1st commercial systems
3+ TFLOPS In One Package
Parallel Performance & Density
Flash Storage

- Nonvolatile semiconductor storage
  - $100\times - 1000\times$ faster than disk
  - Smaller, lower power, more robust
  - But more $$/GB$$ (between disk and DRAM)
Flash Types

• NOR flash: bit cell like a NOR gate
  – Random read/write access
  – Used for instruction memory in embedded systems

• NAND flash: bit cell like a NAND gate
  – Denser (bits/area), but block-at-a-time access
  – Cheaper per GB
  – Used for USB keys, media storage, …

• Flash bits wears out after 1000’ s of accesses
  – Not suitable for direct RAM or disk replacement
  – Wear leveling: remap data to less used blocks
Disk Storage

- Nonvolatile, rotating magnetic storage
Disk Sectors and Access

- Each sector records
  - Sector ID
  - Data (512 bytes, 4096 bytes proposed)
  - Error correcting code (ECC)
    - Used to hide defects and recording errors
  - Synchronization fields and gaps

- Access to a sector involves
  - Queuing delay if other accesses are pending
  - Seek: move the heads
  - Rotational latency
  - Data transfer
  - Controller overhead
Disk Access Example

• Given
  – 512B sector, 15,000rpm, 4ms average seek time, 100MB/s transfer rate, 0.2ms controller overhead, idle disk

• Average read time
  – 4ms seek time
    + \( \frac{1}{2} \div (15,000/60) \) = 2ms rotational latency
    + \( 512 \div 100\text{MB/s} \) = 0.005ms transfer time
    + 0.2ms controller delay
    = 6.2ms

• If actual average seek time is 1ms
  – Average read time = 3.2ms
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.