Computer Organization
CS 231-01

Exam 1 Review

Dr. William H. Robinson
September 20, 2004

http://eecs.vanderbilt.edu/courses/cs231/

History of Computers

• Zeroth Generation – Mechanical Computers
• First Generation – Vacuum Tubes
• Second Generation – Transistors
• Third Generation – Integrated Circuits
• Fourth Generation – VLSI

Topics

“I have never let my schooling interfere with my education.”

– Mark Twain
(1835-1910)

• Review, review, review
  – Ask questions if you have them
  – Computer Organization
  – Processors
  – Memory
  – Input/Output

Structured Computer Organization

Level 5: Problem-oriented language level
  Translation (compiler)

Level 4: Assembly language level
  Translation (assembler)

Level 3: Operating system machine level
  Partial interpretation (operating systems)

Level 2: Instruction set architecture level
  Interpretation (microprogram) or direct execution

Level 1: Microarchitecture level
  Hardware

Level 0: Digital logic level
The Big Picture

• Since 1946 (ENIAC I) all computers have had 5 components

RISC Design Principles

• All instructions directly executed by H/W
  – Eliminate overhead of interpretation

• Maximize rate instructions are issued
  – Utilize parallelism within program

• Instructions should be easy to decode
  – Quickly determine resources required

• Only loads and stores should reference memory
  – Memory access is longer and unpredictable

• Provide plenty of registers
  – Avoid memory access penalty for flushing data set

Pipelining Lessons

• Pipelining doesn’t help latency of single task, it helps throughput of entire workload

• Use different resources to execute multiple tasks simultaneously

• Potential speedup = Number pipe stages

• Time to “fill” pipeline and time to “drain” it reduces speedup

• Stall for Dependences

Addressing Objects: Endianness and Alignment

• Big Endian: address of most significant byte = word address (xx00 = Big End of word)
  – IBM 360/370, Motorola 68k, MIPS, SPARC, HP PA

• Little Endian: address of least significant byte = word address (xx00 = Little End of word)
  – Intel 80x86, DEC VAX, DEC Alpha (Windows NT)

Alignment: require that objects fall on address that is multiple of their size.
What is a Cache?
- Small, fast storage used to improve average access time to slow memory.
- Exploits spatial and temporal locality
- In computer architecture, almost everything is a cache!
  - Registers a cache on variables
  - First-level cache a cache on second-level cache
  - Second-level cache a cache on memory
  - Memory a cache on disk (virtual memory)

The Principle of Locality
- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time.
- Two Different Types of Locality:
  - **Temporal Locality** (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
  - **Spatial Locality** (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon (e.g., straightline code, array access)

Terminology
- **Hit** – memory location found in cache
- **Miss** – memory location not in cache
  - Compulsory
    - Cache is empty at the start of a program
  - Conflict
    - Another valid cache line is currently stored in the location
  - Capacity
    - Cache isn’t large enough to hold the entire working set of a program

Basic Cache Algorithm
- CPU read (memory reference)
  - yes (hit)
    - Line in cache?
      - Read bytes from cache
  - no (miss)
    - Read line from memory into cache
    - Extract desired bytes
    - Return data to CPU
**Terminology**

N-bit address

- **TAG field**
  - Indicates the address tag for comparison
- **LINE field**
  - Indicates the proper cache entry
- **WORD field**
  - Indicates which word referenced within a line
- **BYTE field**
  - Indicates a single byte, but not normally used

**Direct-Mapped Cache**

Read 0x0A72

<table>
<thead>
<tr>
<th>Entry</th>
<th>Valid</th>
<th>Tag</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>2</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>3</td>
<td>Y</td>
<td>0x0A</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>5</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>6</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>7</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

Read 0x2464

<table>
<thead>
<tr>
<th>Entry</th>
<th>Valid</th>
<th>Tag</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>2</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>3</td>
<td>Y</td>
<td>0x24</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>5</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>6</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>7</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

**Mean Access Time**

Given

- Cache access time, \( c \)
- Main memory access time, \( m \)
- Hit ratio, \( h \)

\[
\text{mean access time} = c + (1-h)m
\]
Reduce Miss Penalty with L2 Cache

- **L2 Equations**
  - \( AMAT = Average\ Memory\ Access\ Time \)
  - \( AMAT = Hit\ Time_{L1} + Miss\ Rate_{L1} \times Miss\ Penalty_{L1} \)
  - \( Miss\ Penalty_{L1} = Hit\ Time_{L2} + Miss\ Rate_{L2} \times Miss\ Penalty_{L2} \)
  - \( AMAT = Hit\ Time_{L1} + Miss\ Rate_{L1} \times (Hit\ Time_{L2} + Miss\ Rate_{L2} \times Miss\ Penalty_{L2}) \)

- **Definitions:**
  - Local miss rate — misses in this cache divided by the total number of memory accesses to this cache (Miss rate_{L2})
  - Global miss rate — misses in this cache divided by the total number of memory accesses generated by the CPU
  - Global Miss Rate is what matters

Hamming Codes

- First major class of binary codes designed for error correction
  - Originally used in error control for long-distance telephony
  - Encodes parity for groups of bits within data

General Idea: Code Vector Space

- Not every vector in the code space is valid
- Hamming Distance for the code (d_{min}): Minumum number of bit flips to turn one code word into another
- Number of errors that we can detect: \(d_{min} - 1\)
- Number of errors that we can fix: \(\frac{1}{2}(d_{min} - 1)\)

Hamming Codes

- First major class of binary codes designed for error correction
  - Originally used in error control for long-distance telephony
  - Encodes parity for groups of bits within data

Bit Numbering for Hamming Algorithm

Given an 8-bit data word to encode

- Data bits (remaining bits)
- Parity bits (powers of 2)
### Parity Bit Assignment

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Bit No. in Binary</th>
<th>Encoded by</th>
<th>Encodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 0 0 0 1</td>
<td>3, 5, 7, 9, 11</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0 0 0 1 0</td>
<td>3, 6, 7, 10, 11</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0 0 1 1 1</td>
<td>2, 1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0 1 0 0 0</td>
<td>5, 6, 7, 12</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0 1 0 1 1</td>
<td>4, 1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0 1 1 0 0</td>
<td>4, 2</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0 1 1 1 1</td>
<td>4, 2, 1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1 0 0 0 0</td>
<td>9, 10, 11, 12</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1 0 0 1 0</td>
<td>8, 1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1 0 1 0 0</td>
<td>8, 2</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1 0 1 1 1</td>
<td>8, 2, 1</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1 1 0 0 0</td>
<td>8, 4</td>
<td></td>
</tr>
</tbody>
</table>

### Method for Parity Check

- For parity bit 1
  - \( b_8 \oplus b_2 \oplus b_0 \oplus b_7 \oplus b_6 \oplus b_1 \)
  - Even parity \( \rightarrow 0 \)
  - Odd parity \( \rightarrow 1 \)

- For parity bit 2
  - \( b_2 \oplus b_6 \oplus b_7 \oplus b_10 \oplus b_11 \)

- For parity bit 4
  - \( b_4 \oplus b_5 \oplus b_6 \oplus b_7 \oplus b_{12} \)

- For parity bit 8
  - \( b_8 \oplus b_9 \oplus b_{10} \oplus b_{11} \oplus b_{12} \)

### IEEE Floating Point Standard 754

- Provided designers with a correct model
- Allowed FP data to be exchanged among different computer systems
- Defines three formats
  - Single precision (32 bits)
  - Double precision (64 bits)
  - Extended precision (80 bits)
    - Only occurs within FP units

### Disk Device Terminology

- Several platters, with information recorded magnetically on both surfaces (usually)
- Bits recorded in tracks, which in turn divided into sectors (e.g., 512 Bytes)
- Actuator moves head (end of arm, 1/surface) over track ("seek"), select surface, wait for sector rotate under head, then read or write
  - Cylinder: all tracks under heads

Adapted from John Kubiatowicz's CS 252 lecture notes. Copyright © 2003 UCB.
Formatting a Track

Fig. 2-19. A portion of a disk track. Two sectors are illustrated.

Redundant Arrays of (Inexpensive) Disks

- Files are "striped" across multiple disks
- Redundancy yields high data availability
  - Availability: service still provided to user, even if some components failed
- Disks will still fail
- Contents reconstructed from data redundantly stored in the array
  \[ \Rightarrow \] Capacity penalty to store redundant info
  \[ \Rightarrow \] Bandwidth penalty to update redundant info

Adapted from David Culler’s CS 252 lecture notes. Copyright © 2002 UCB.

RAID Levels 0 Through 5

Backup and parity drives are shaded

Summary

- Computer Organization is a GREAT class
- Dr. Robinson is a GREAT professor
- Main ideas
  - Levels of organization
  - Basic computer components
  - Implementing cache and error correction
- Study hard and good luck!