FINAL CLASS!
My next job?
Announcements

- Review Session Helpful? Solutions out soon.
- Request Topics review.
- A single handwritten sheet of notes allowed for finals. (Confirmation on this soon.)
- No electronic devices.
Today?
FINISHED WITH EXAMS?

LOOK RIGHT HERE
CS 354 ~ What to take away?

Ganesh Kumar . May 6, 2016
General

**Everything** is represented as a sequence of bits (0s and 1s):

- Your executable
- Your images
- Your browser application
- Your pdfs
- ...
- Everything!

What they actually represent...

- Depends on **context**
- How we choose to **interpret** them.
More specifically,

- I have a piece of data that is \texttt{0100 0001}. What is it?

- What is \texttt{01001010101101101011010101010101010100}?

  Are we going to split it into 8 bit groups? Or split it into 32 bit groups?

**Special Mention**: Quantum Computing
Compilation System

- hello.c
  - Source program (text)
  - Pre-processor (cpp)
  - Modified source program (text)
  - hello.i
  - Compiler (cc1)
  - hello.s
  - Assembly program (text)
  - Assembler (as)
  - hello.o
  - Relocatable object programs (binary)
  - Linker (ld)
  - hello
    - Executable object program (binary)
C - Pointers

Operators

&varX  - ADDRESS OF variable varX.

*varY  - VALUE AT address varY.  (Indirection operator)

How do you define a char pointer?

char  ch = 'y';
char  * ptr = &ch;

This right here is not an indirection operator. It is just pointer declaration syntax;
Data Representation

Endianness

Integer - 0x12 34 56 78 - 4 bytes.

<table>
<thead>
<tr>
<th>Addr</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x100</td>
<td>78</td>
</tr>
<tr>
<td>0x101</td>
<td>56</td>
</tr>
<tr>
<td>0x102</td>
<td>34</td>
</tr>
<tr>
<td>0x103</td>
<td>12</td>
</tr>
</tbody>
</table>

Little Endian

<table>
<thead>
<tr>
<th>Addr</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x100</td>
<td>12</td>
</tr>
<tr>
<td>0x101</td>
<td>34</td>
</tr>
<tr>
<td>0x102</td>
<td>56</td>
</tr>
<tr>
<td>0x103</td>
<td>78</td>
</tr>
</tbody>
</table>

Big Endian
Data Representation

4 bit datatype - 0000 to 1111

Unsigned Representation

No specific bits to denote sign. So value goes from DEC 0 to 15.

Signed Representation

MSB allocated for sign.

So value ranges from

<table>
<thead>
<tr>
<th>BIN</th>
<th>0111</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC</td>
<td>7</td>
<td>-8</td>
</tr>
</tbody>
</table>
Assembly

- How to read the x86 Instruction Sheet.
- Registers
- Control Flags and Conditional Jumps.
  - cmp and test instructions followed by jumps.
- Function Stack Frames!
  - All the space associated with a function goes away after it returns i.e. Popping it off the stack.
-
Memory

- Not all storage technology are created equal.
- Some are fast and expensive and some are relatively slow and inexpensive.
- Use a fast memory to serve as a *staging area* for data from a slower one - **CACHING!**
Memory Hierarchy

- **L0:** CPU registers hold words retrieved from cache memory.
- **L1:** L1 cache (SRAM)
  - L1 cache holds cache lines retrieved from L2 cache.
- **L2:** L2 cache (SRAM)
  - L2 cache holds cache lines retrieved from L3 cache.
- **L3:** L3 cache (SRAM)
  - L3 cache holds cache lines retrieved from memory.
- **L4:** Main memory (DRAM)
  - Main memory holds disk blocks retrieved from local disks.
- **L5:** Local secondary storage (local disks)
  - Local disks hold files retrieved from disks on remote network servers.
- **L6:** Remote secondary storage (distributed file systems, Web servers)
Cache

Design a cache for me!

(S, E, B, m)
Locality

Temporal Locality

Accessing the same memory location over and over again—Good Thing!

Spatial Locality

Accessing the memory in sequence (stride-1 reference pattern) is a good thing as well.

Tip: Algorithm design does not consider the physical limitations of memory.
Virtual Memory

Why Virtual Memory?

- Memory Protection
- Easy Memory Management
- Use more space that what is physically available in the physical DRAM memory.
Virtual Memory

Important Task

Virtual Address → Physical Address

How?

- Memory Management Unit (In-charge of doing this!)
- Page Tables (Software entity)
- Translation Lookaside Buffer (Separate cache)
Dynamic Memory Allocation

Huge space of unallocated memory - The Heap!

Why an allocator?

- Like any shared resource, access to this resource has to be controlled.

- See Tragedy of the Commons. (Shared resource - Road. Controller - Cops and Traffic Rules).

- So, we need a Dynamic Memory Allocator.

- ... with a lot of rules and control mechanisms - Headers, Alignment Restrictions, Maintenance.
Exceptions

Abrupt change in a processor’s control flow.

Different Kinds

- **Interrupts** (not under our control ~ Asynchronous)
  - Press Ctrl + C!
- **System Calls** (we cause the abrupt change)
  - Write to display..
- **Faults** (we cause this)
  - Accessing a page that is not in the main memory.
- **Abort** (we cause this)
Context Switching

Switch from executing one process to another... while storing the status (or context) of the switched out process.

Why?
CPU should not wait for a slow task that a process needs to be performed. (Relative times - 1 sec for a CPU is 10 months for a Hard Disk)

Process has two modes - Kernel and User

Needs to be in kernel mode before performing a context switch. Why?
Linking

Hope this is still fresh in memory!
Goodbye
&
Good Luck!