Tour of Computer Systems

Karthik Dantu
Ethan Blanton
Computer Science and Engineering
University at Buffalo
kdantu@buffalo.edu

Some portions of this lecture are borrowed from the CMU 15-213 and UNL CSCE 284H

Administrivia

- PA1 handout due today
- PA1 – Conway's Game of Life

Systems Knowledge is Power!

- **Systems Knowledge**

  How hardware (processors, memories, disk drives, network infrastructure) plus software (operating systems, compilers, libraries, network protocols) combine to support the execution of application programs

  How you as a programmer can best use these resources

- **Useful outcomes from taking CSE 220**

  Become more effective programmers

  - Able to find and eliminate bugs efficiently
  - Able to understand and tune for program performance

  Prepare for later "systems" classes in CS, CE,

  - Operating Systems, Networks, Computer Architecture, Embedded Systems, Computer Security, etc.

Important to Know How Things Work

- **Why do I need to know this stuff?**
  - Abstraction is good, but don't forget reality
- **Most CS courses emphasize abstraction**
  - Abstract data types
  - Asymptotic analysis
- **These abstractions have limits**
  - Especially in the presence of bugs
  - Need to understand details of underlying implementations
  - Sometimes the abstract interfaces don't provide the level of control or performance you need

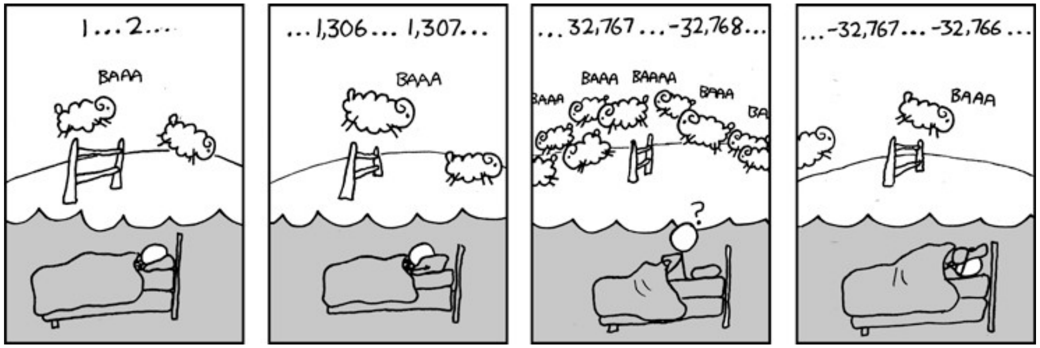
Great Reality #1

- **Example 1: Is $x^2 \geq 0$?**

  Floats: Yes!

  ints:

  ```
  40000 * 40000 --> 1600000000
  50000 * 50000 --> ?
  ```



- **Example 2: Is (x + y) + z = x + (y + z)?**

  Unsigned & Signed Ints: Yes!

  Floats:

  ```
  (1.0e20 + -1.0e20) + 3.14 --> 3.14
  1.0e20 + (-1.0e20 + 3.14) --> ??
  ```

Computer Arithmetic

- **Does not generate random values**
  - Arithmetic operations have important mathematical properties
- **Cannot assume all "usual" mathematical properties**
  - Due to finiteness of representations
  - Integer operations satisfy "ring" properties
    - Commutativity, associativity, distributivity
  - Floating point operations satisfy "ordering" properties
    - Monotonicity, values of signs
- **Observation**
  - Need to understand which abstractions apply in which contexts
  - Important issues for compiler writers and serious application programmers

CSE 191

Great Reality #2

- **Chances are, you'll never write programs in assembly**
  - Compilers are much better & more patient than you are
- **But: Understanding assembly is key to machine-level execution model**
  - Behavior of programs in presence of bugs
    - High-level language models break down
  - Tuning program performance
    - Understand optimizations done / not done by the compiler
    - Understanding sources of program inefficiency
  - Implementing system software
    - Compiler has machine code as target
    - Operating systems must manage process state
  - Creating / fighting malware
    - x86 assembly is the language of choice!

Great Reality #3: Memory Matters

- **Memory is not unbounded**
  - It must be allocated and managed
  - Many applications are memory dominated
- **Memory referencing bugs especially pernicious**
  - Effects are distant in both time and space
- **Memory performance is not uniform**
  - Cache and virtual memory effects can greatly affect program performance
  - Adapting program to characteristics of memory system can lead to major speed improvements

Memory Referencing Errors

- C and C++ do not provide any memory protection
  - Out of bounds array references
  - Invalid pointer values
  - Abuse of `malloc`/`free`
- Can lead to nasty bugs
  - Whether or not bug has effect depends on the compiler
  - Action at a distance
    - Corrupted object logically unrelated to one accessed
    - Effect of bug may first be observed long after it occurred
- How do I deal with this?
  - Don't – program in Java, Lisp and ML
  - Use/develop tools to detect memory errors (`valgrind`)

Memory Bug Example

```
int main() {
      long int a[2];
      double d = 3.14;
      a[2] = 1073741824;
      printf("d=%.15g",d);
      exit(0);
}
```

	Alpha	MIPS	Linux
-g	5.30498947741318e-315	3.1399998664856	3.14
-O	3.14	3.14	3.14

Memory Bug – Therac 25

- Computer controlled radiation therapy machine
- Six accidents between 1985 and 1987

  100 times the recommended dose of radiation

  Concurrent programming errors

https://medium.com/swlh/software-architecture-therac-25-the-killer-radiation-machine-8a05e0705d5b

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Toyota Acceleration (2009-11)

- Unintended acceleration
- ~9 million vehicles recalled
- “Stack overflow”
- Toyota fined $1.2B for “concealing safety defects”

Toyota "Unintended Acceleration" Has Killed 89

A 2005 Toyota Prius, which was in an accident, is seen at a police station in Harrison, New York, Wednesday, March 10, 2010. The driver of the Toyota Prius told police that the car accelerated on its own, then lurched down a driveway, across a road and into a stone wall. (AP Photo/Seth Wenig) / AP PHOTO/SETH WENIG

Unintended acceleration in Toyota vehicles may have been involved in the deaths of 89 people over the past decade, upgrading the number of deaths possibly linked to the massive recalls, the government said Tuesday.

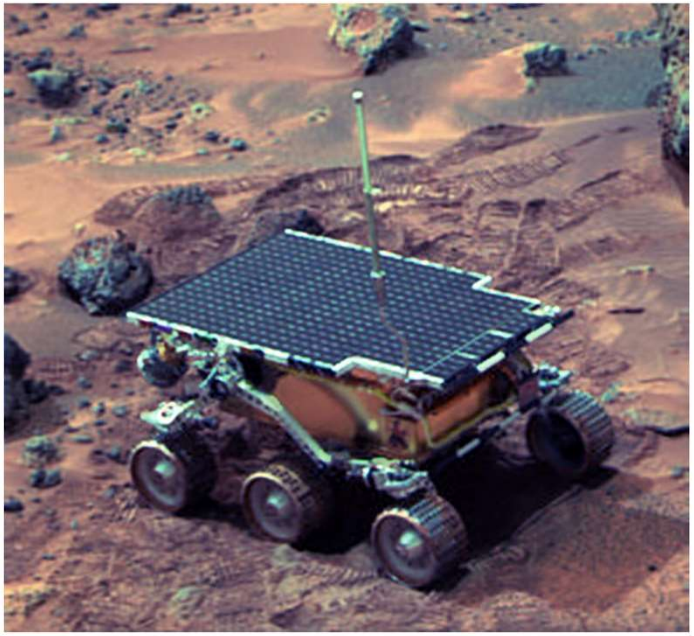
The National Highway Traffic Safety Administration said that from 2000 to mid-May, it had received more than 6,200 complaints involving sudden acceleration in Toyota vehicles. The reports include 89 deaths and 57 injuries over the same period. Previously, 52 deaths had been suspected of being connected to the problem. http://www.cbsnews.com/news/toyota-unintended-acceleration-has-killed-89/

https://users.ece.cmu.edu/~koopman/pubs/koopman14_toyota_ua_slides.pdf

Mars Pathfinder (1997)

- Frequently locked up and stopped responding
  - Automatic reboots

- Priority inversion in "parallel" software



https://www.rapitasystems.com/blog/what-really-happened-to-the-software-on-the-mars-pathfinder-spacecraft

Great Reality #4: Performance is more than Asymptotic Complexity

- **Constant factors matter too!**
- **And even exact op count does not predict performance**
  - Easily see 10:1 performance range depending on how code written
  - Must optimize at multiple levels: algorithm, data representations, procedures, and loops
- **Must understand system to optimize performance**
  - How programs compiled and executed
  - How to measure program performance and identify bottlenecks
  - How to improve performance without destroying code modularity and generality

Memory Performance Example

```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (i = 0; i < 2048; i++)
    for (j = 0; j < 2048; j++)
      dst[i][j] = src[i][j];
}
```

4.3ms

```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (j = 0; j < 2048; j++)
    for (i = 0; i < 2048; i++)
      dst[i][j] = src[i][j];
}
```

81.8ms

2.0 GHz Intel Core i7 Haswell

- Hierarchical memory organization
- Performance depends on access patterns
  - Including how step through multi-dimensional array

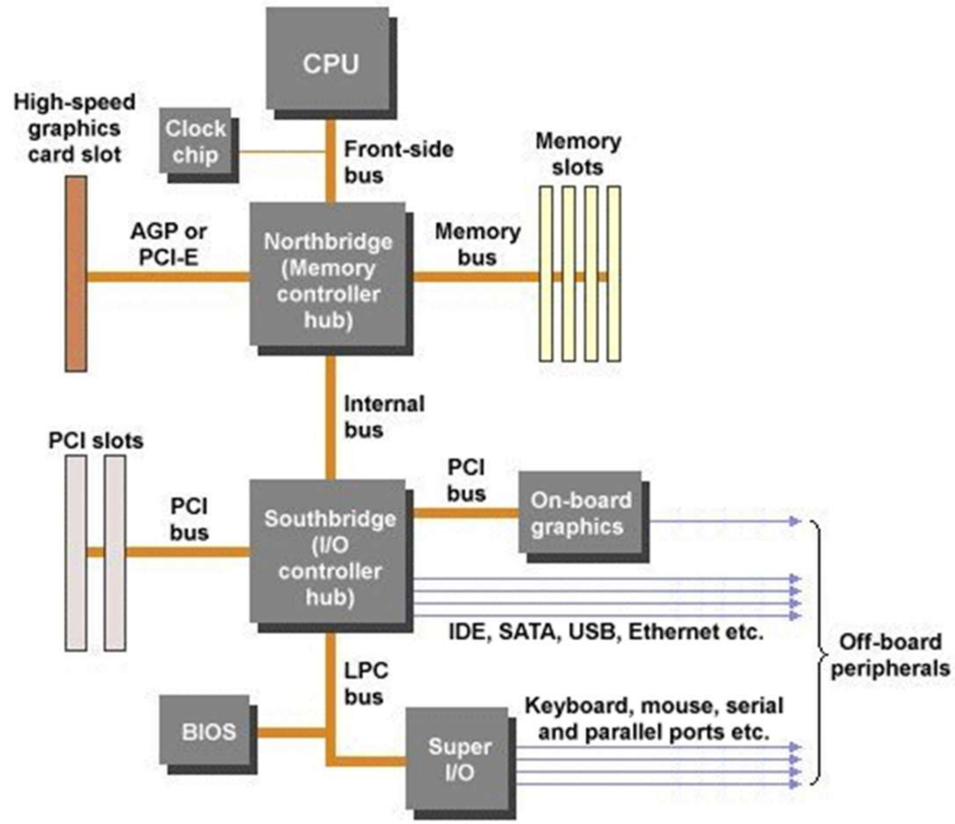
Great Reality #5: Computers do more than execute programs

- **They need to get data in and out**
  - I/O system critical to program reliability and performance

- **They communicate with each other over networks**
  - Many system-level issues arise in presence of network
    - Concurrent operations by autonomous processes
    - Coping with unreliable media
    - Cross platform compatibility
    - Complex performance issues

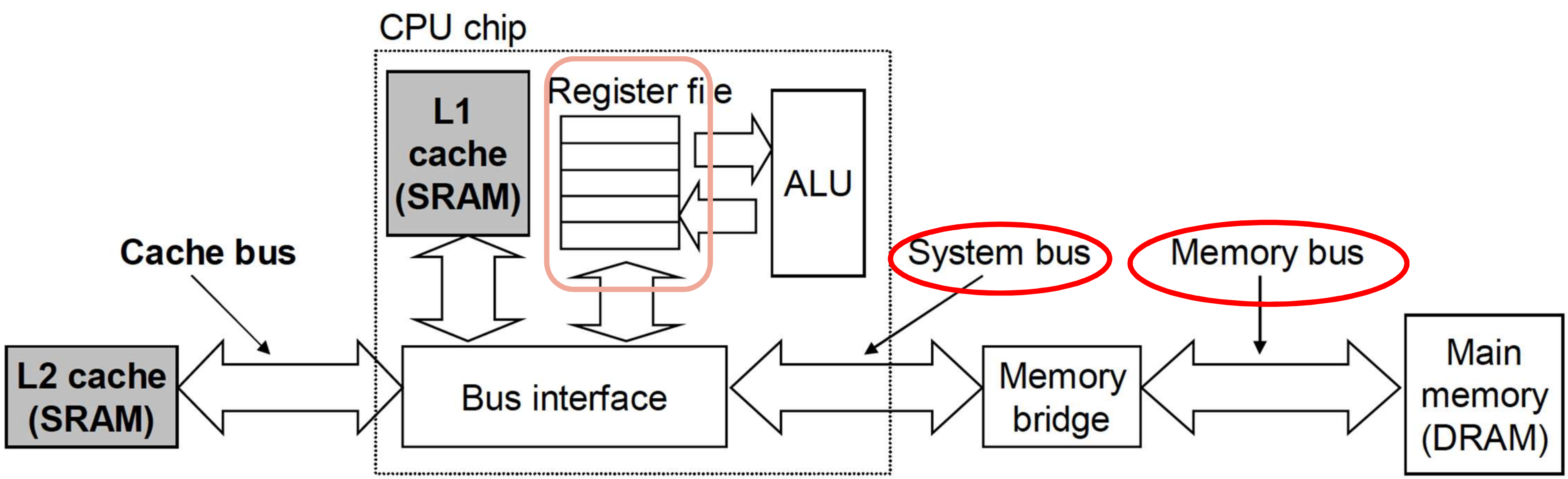
What is a Computer?

- CPU, Memory, I/O
- Connected via the FSB and other buses
- Memory bus width determines access width
- Internal bus speeds (not CPU) determines overall speed
- Several layers of complexity



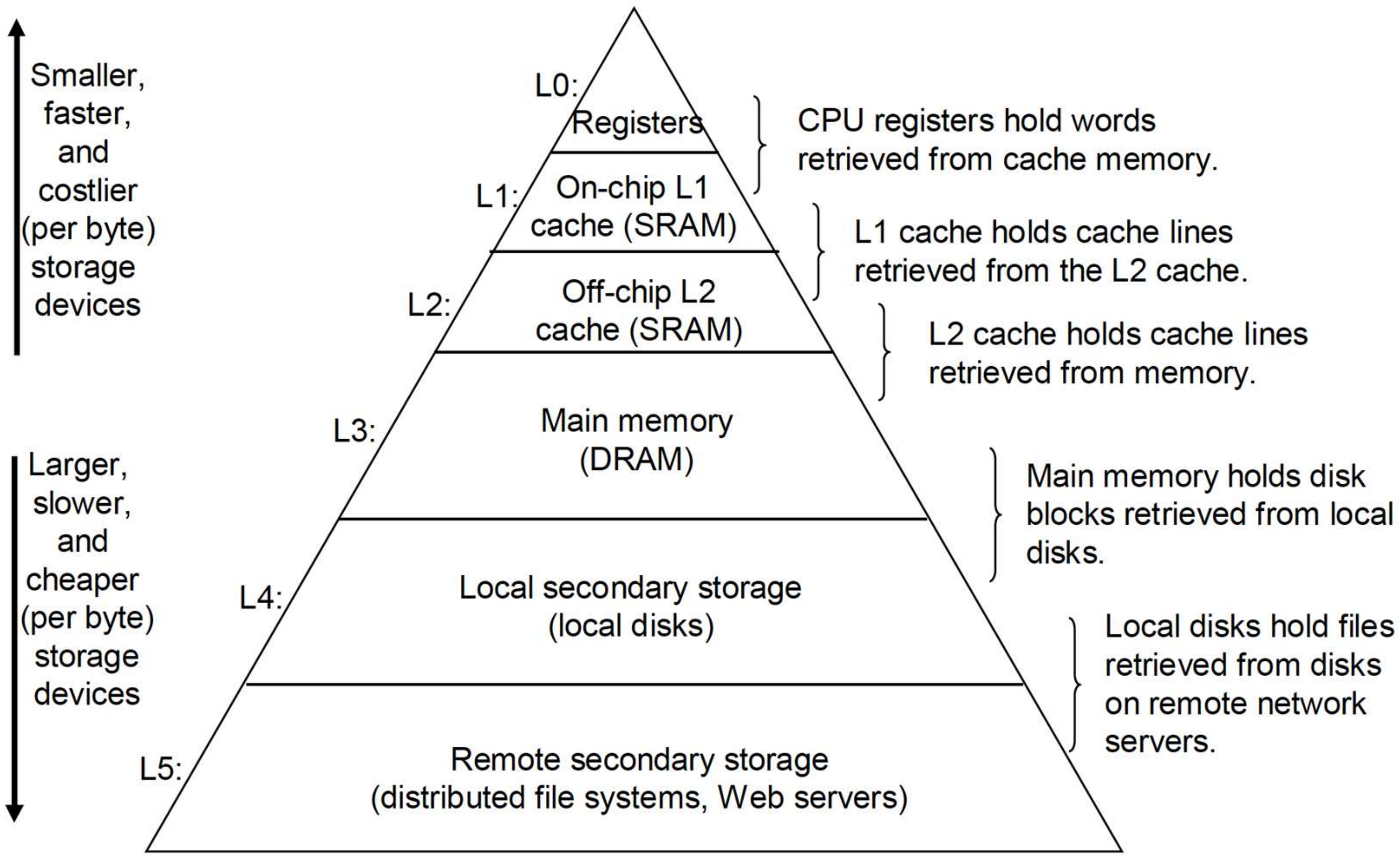
http://www.technologyuk.net/computing/computer-hardware/motherboard.shtml

Modern CPU

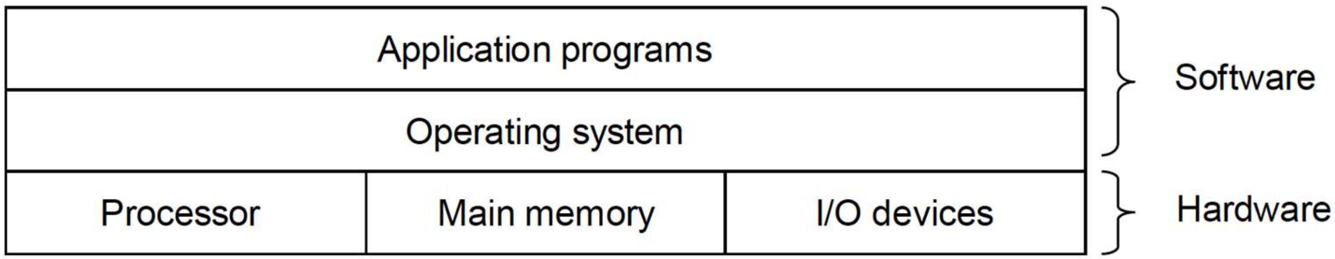


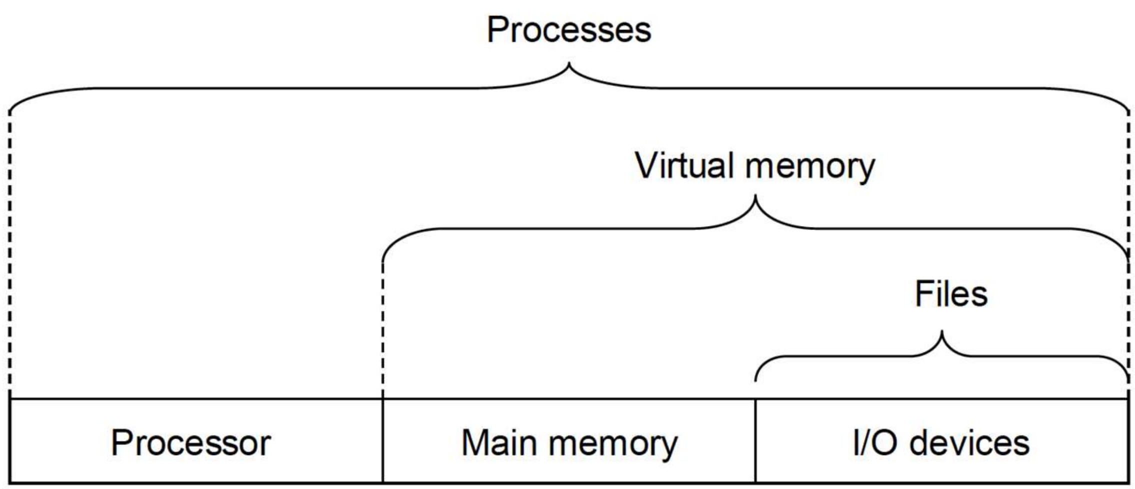
- Bus widths are fixed and determine access speed, addressable range and overall system speed
- Fixed number of registers
- Address/data bus widths might be different

Memory Hierarchy



OS Abstracts HW





Summary

- The computer system is more than just hardware
- We have to understand both the hardware and systems interfaces to properly understand and use a computer
- Next class – how numbers are represented!

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Required Reading

- BO Chapter 1