CSC 3210 Computer organization and programming

Chunlan Gao

Georgia State University

Syllabus

- Check the Icollege!!
- https://educate.elsevier.com/book/details/9780128203316
- Syllabus-3210-Spring25-ACS15

Chapter1

Computer Abstractions and Technology

What is computer?

A **mechanism** that does two things:

- It **directs** the processing of data and it **performs** the actual processing of data.
- It does both of these things in response to a **computer program**.

Computer Program

```
#include<stdio.h>
int main()
{
// Variable declaration
int a, b, sum;

// Take two numbers as input from the user
scanf("%d %d", &a, &b);

// Add the numbers and assign the value to some variable
so that the
// calculated value can be used else where
sum = a + b;

// Use the calculated value
printf("%d\n", sum);

return 0;
// End of program
}
```

The Computer Revolution

1. Progress in computer technology

Continuous and significant advancements in computer hardware and software over time. The development of faster, more efficient, and specialized components has driven technological progress.

- **Underpinned by domain-specific accelerators**

Much of this progress has been made possible by the introduction of specialized accelerators designed for specific tasks. These accelerators allow computers to perform complex operations (e.g., AI tasks, graphics rendering, and scientific computations) much faster than general-purpose processors.

The Computer Revolution

- **2. Makes novel applications feasible**

Some new computer technology have enabled the development of entirely new applications that were previously impractical or impossible. Some examples are:

- **Computers in automobiles**
  Modern cars are equipped with powerful computers that manage various functions such as engine control, navigation systems, autonomous driving, and in-car entertainment.
- **Cell phones**
  Smartphones are essentially compact computers, enabling tasks like internet browsing, multimedia playback, and mobile applications, all made possible by advancements in computing technology.

The Computer Revolution

- **Human Genome Project**
  This was a large-scale scientific project aimed at mapping and understanding the entire human genome. It required enormous computational power, and its success was driven by advancements in specialized hardware and software.
- **World Wide Web**
  The internet as we know it today became feasible due to progress in computing technology, which enabled fast data transmission, web servers, and browsers to operate efficiently.
- **Search Engines**
  The development of search engines, such as Google, relies on highly optimized hardware and algorithms capable of indexing and retrieving vast amounts of data in milliseconds.

Classes of Computers

Personal computers

- General purpose, variety of software
- Subject to cost/performance tradeoff

Sever Computers

- Network based
- High capacity, performance, reliability
- Range from small servers to building sized

Classes of Computers

Supercomputers

- Type of server
- High-end scientific and engineering calculations
- Highest capability but represent a small fraction of the overall computer market

Embedded computers

- Hidden as components of systems
- Stringent power/performance/cost constraints

What Will We Learn???

- How programs are translated into the machine language
    - And how the hardware executes them
- The hardware/software interface
- What determines program performance
    - And how it can be improved
- How hardware designers improve performance
- What is parallel processing

The Performance?

- Algorithm
  - Determines number of operations executed
- Programming language, compiler, architecture
  - Determine number of machine instructions executed per operation
- Processor and memory system
  - Determine how fast instructions are executed
- I/O system (including OS)
  - Determines how fast I/O operations are executed

Great Ideas To Help

There are Seven Great Ideas in Computer Architecture, which are fundamental principles that guide the design and optimization of computer systems. Here's an explanation of each idea:

- **1. Use Abstraction to Simplify Design**

Abstraction hides the complexity of lower-level details, allowing developers to focus on higher-level logic. For example, high-level programming languages abstract away hardware-level operations.

2. Make the Common Case Fast

In most systems, certain operations occur much more frequently than others. Optimizing these common operations can improve overall performance more than focusing on rare cases.

3. Performance via Parallelism

Parallelism involves performing multiple operations simultaneously. For instance, multi-core processors execute tasks in parallel to increase throughput.

Great Ideas To Help-con

- **4. Performance via Pipelining**

Pipelining breaks tasks into stages and executes different stages of multiple tasks simultaneously. In a CPU, instructions are divided into stages like fetch, decode, and execute, allowing multiple instructions to be processed concurrently.

- **5. Performance via Prediction**

Prediction techniques, such as branch prediction, anticipate future operations to minimize delays. For example, a CPU may predict the outcome of a branch and continue execution without waiting.

- **6. Hierarchy of Memories**

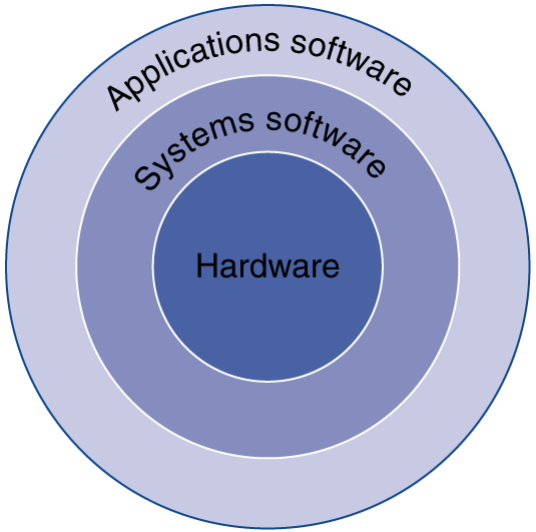
Memory is organized into a hierarchy (registers, caches, main memory, disk), with faster, smaller memories closer to the processor. Exploiting this hierarchy reduces latency and improves performance.

- **7. Dependability via Redundancy**

Redundancy involves adding extra components or information to improve reliability. For example, RAID uses redundant storage to prevent data loss, and error correction codes (ECC) are used to detect and correct memory errors.

How does a program work?

- Application software
  - Written in high-level language
- System software
  - Compiler: translates HLL code to machine code
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- Hardware
  - Processor, memory, I/O controllers



Levels of Program code

- High-level language
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
  - Easy for people understand
- Assembly language
  - Textual representation of instructions
- Hardware representation
  - Binary digits (bits)
  - Encoded instructions and data