Concepts Introduced

- classes of computers
- great architecture ideas
- software levels
- computer components
- performance measures
- technology trends

Classes of Computers

- personal computers (PCs)
  - intended for a single user at a stationary location
  - notebooks and workstations
  - emphasize good performance to single users at low cost
- servers
  - assessed by other computers to provide computation and/or data
  - typically only accessed via a network
  - greater computing, storage, and I/O capacity
  - emphasis on performing well under large workloads with enhanced dependability
- embedded computers
  - computers contained in other devices
  - usually a small number of predetermined applications
  - emphasis on cost and low power

Classes of Computers (cont.)

- personal mobile devices (PMDs)
  - battery-powered wireless devices with multimedia user interfaces
  - smart phones and tablets
  - reliance on touch screens
  - emphasis on cost and energy efficiency
- large clusters/warehouse-scale computers (WSCs)
  - large collections of servers connected by a network to act as a single powerful computer
  - scalability and availability handled through the network

Great Architecture Ideas

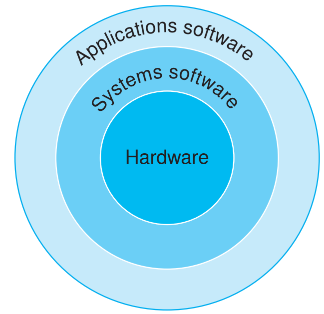
- **Design for Moore's law.**
  - Changes in computers are largely driven by Moore's Law, which states the number of transistors on a chip doubles every 18-24 months.
  - Architects have to anticipate where technology will be when the design is completed.
- **Use abstraction to simplify design.**
  - Abstraction is used to represent the design at different levels.
  - Lower-level details can be hidden to provide simpler models at higher levels.
- **Make the common case fast.**
  - Identify the common case and try to improve it.
  - Most cost efficient method to obtain improvements.
- **Improve performance via parallelism.**
  - Improve performance by performing operations in parallel.
  - There are many levels of parallelism.

Great Architecture Ideas (cont.)

- **Improve performance via pipelining.**
  - Break tasks into stages so that multiple tasks can be simultaneously performed in different stages.
  - Commonly used to improve instruction throughput.
- **Improve performance via prediction.**
  - Sometime faster to assume a particular result than waiting until the result is known.
  - Known as speculation and is used to guess results of branches.
- **Use a hierarchy of memories.**
  - Make the fastest, smallest, and most expensive per bit memory the first level accessed and the slowest, largest, and cheapest per bit memory the last level accessed.
  - Allows most of the accesses to be caught at the first level and be able to retain most of the information at the last level.
- Improve dependability via redundancy.
  - Include redundant components that can both detect and often correct failures.
  - Used at many different levels.

Layers of a Computer

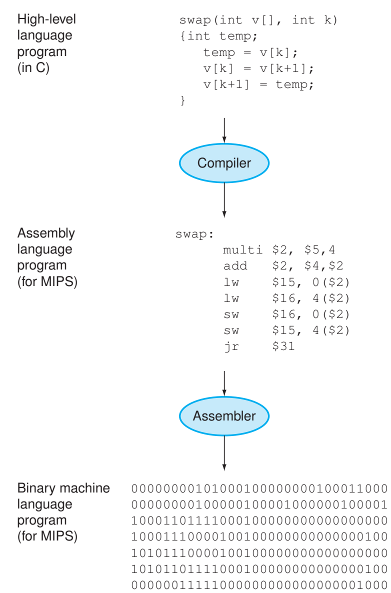
- Computers are organized into layers.
  - Applications are software programs invoked by a user.
  - System software provides useful services.
    - An operating system (1) handles I/O, (2) allocates storage and memory, and (3) allows multiple applications to share a computer.
    - Compilers translate applications written in a high-level language to instructions a machine can execute.



Program Levels and Translation

- program levels
  - high level language - level at which programmers develop applications
  - assembly language - symbolic representation of instructions
  - machine language - binary representations of instructions that can be executed by a processor
- translating between program levels
  - compiler - Translates a high-level language program file into an assembly language file.
  - assembler - Translates an assembly language file into a machine language file.
  - linker - Translates machine language files into a single executable file that can be loaded into memory and executed.

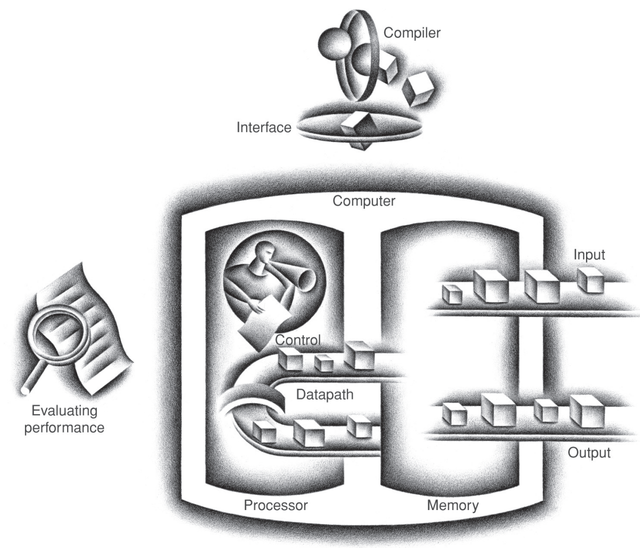
Example of Translating a C Program



What's in a Computer?

- processor - Performs the actions specified by the machine instructions of a program.
  - datapath - Portion of the processor that performs the arithmetic and logical operations.
  - control - Portion of the processor that commands the datapath, memory system, and I/O devices.
- memory system - Storage area where programs and data are kept.
- input devices - Mechanisms through which the computer receives external information.
- output devices - Mechanisms through which the computer conveys its results.

Organization of a Computer



Hardware/Software Abstractions

- The instruction set architecture (ISA) is the programmer visible instruction set that is the boundary between the hardware and the software.
  - operations - includes data transfer, arithmetic/logical, floating-point, and transfers of control
  - data types and sizes of operands - most processors include 8-bit (*char* and *unsigned char*), 16-bit (*short* and *unsigned short*), 32-bit (*int*, *unsigned int*, *float*), 64-bit (*long long*, *unsigned long long*, and *double*)
  - addressing modes - constants, registers, and ways to access memory
  - encoding - how machine instructions are represented in binary
- An ISA enables the development of many hardware implementations of varying cost and performance that can run identical software.
- The application binary interface (ABI) includes the ISA and the operating system (OS) interfaces and defines a standard for portability of executables across computers.

Memory

- Volatile (primary) memory loses information when power is turned off and is used to hold data and instructions associated with applications while they are running on a processor.
  - Main memory consist of dynamic random access memory (DRAM) chips. Each access to DRAM takes the same amount of time.
  - Cache memory consists of static random access memory (SRAM) that is generally on the same chip as the processor.
- Nonvolatile (secondary) memory retains information without power and is used to hold data and programs between runs.
  - Magnetic disks are used in PCs, servers, and WSCs.
  - Flash memory is used in PMDs.

Input and Output

- Input devices are the mechanisms for a processor to obtain external data. Input devices include the keyboard, mouse, touchscreen, microphone, image scanner, webcam, etc.
- Output devices are the mechanisms for conveying data to a user. Output devices include monitors, printers, speakers.
- Secondary storage (disks, flash memory) are sometimes considered I/O devices.

Communicating with Other Computers

- Networks connect computers allowing them to share data.
  - A local area network (LAN) is designed to connect computers within a relatively small area, such as a single building. Ethernet is a commonly used LAN.
  - LANs can be connected with switches to provide routing services.
  - Wide area networks (WANs) support communication across a continent, are based on optical fibers, and are the backbone of the Internet.
- Most PMDs, servers, and even PCs today are connected through a network in some way.

Steps for Executing a Program

1. Input device loads the machine code from the executable.
2. The machine code is stored in memory.
3. Processor fetches an instruction.
4. Control decodes the instruction.
5. Datapath executes the instruction.
6. If application not complete, then go to step 3.

Gauging Performance

- factors affecting the performance of a computer system
  - architecture
  - hardware implementation of the architecture
  - compiler for the architecture
  - operating system

Performance Terms

- Latency (response time) is the time between the start and completion of an event.
- Bandwidth (throughput) is the total amount of work done in a given period of time.

Performance Equations

- Performance has an inverse relationship to execution time.

$$Performance = \frac{1}{Execution\_Time}$$

- Comparing the performance of two machines can be accomplished by comparing execution times.

$$Performance_X > Performance_Y$$

$$\frac{1}{Execution\_Time_X} > \frac{1}{Execution\_Time_Y}$$

$$Execution\_Time_Y > Execution\_Time_X$$

N Times Faster

- Often people state that a machine X is n times faster than a machine Y. What does this mean?

$$\frac{Performance_X}{Performance_Y} = n = \frac{Execution\_Time_Y}{Execution\_Time_X}$$

- If machine X takes 20 seconds to perform a task and machine Y takes 2 minutes to perform the same task, then machine X is how many times faster than machine Y?

Measures of Clock Speed

- clock periods
  - millisecond (ms) - 10^{-3} of a second
  - microsecond (μs) - 10^{-6} of a second
  - nanosecond (ns) - 10^{-9} of a second
  - picosecond (ps) - 10^{-12} of a second
  - femtosecond (fs) - 10^{-15} of a second
- clock rates
  - kilohertz (KHz) - 10^{3} cycles per second
  - megahertz (MHz) - 10^{6} cycles per second
  - gigahertz (GHz) - 10^{9} cycles per second
  - terahertz (THz) - 10^{12} cycles per second
  - petahertz (PHz) - 10^{15} cycles per second

Measures of Data Size

- bit - Binary digIT
- nibble - four bits
- byte - eight bits
- word - four bytes (32 bits) on many embedded/mobile processors and eight bytes (64 bits) on many desktops and servers
- kibibyte (Kib) [kilobyte (Kb)] - 2^{10} (1,024) bytes
- mebibyte (Mib) [megabyte (Mb)] - 2^{20} (1,048,576) bytes
- gibibyte (Gib) [gigabyte (Gb)] - 2^{30} (1,073,741,824) bytes
- tebibyte (Tib) [terabyte (Tb)] - 2^{40} (1,099,511,627,776) bytes
- pebibyte (Pib) [petabyte (Pb)] - 2^{50} (1,125,899,906,842,624) bytes

CPU Time

- CPU time ignores I/O and the time for executing other processes.
- CPI stands for cycles per instruction.

$$CPU\_time = CPU\_clock\_cycles * clock\_cycle\_time = \frac{CPU\_clock\_cycles}{clock\_rate}$$

$$CPI = \frac{CPU\_clock\_cycles}{instruction\_count}$$

$$CPU\_time = Instruction\_count * CPI * clock\_cycle\_time$$

- CPI cannot be looked up in a manual as it can be affected by many external events.
- CPU time really needs to be measured and it can vary somewhat on each execution.

Amdahl's Law

- Amdahl's Law states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.
- Amdahl's Law depends on two factors:
  - The fraction of the time the enhancement can be exploited.
  - The improvement gained by the enhancement while it is exploited.

$$execution\_time_{new} = execution\_time_{old} * (1 - fraction_{enhanced} + \frac{fraction_{enhanced}}{speedup_{enhanced}})$$

$$speedup_{overall} = \frac{execution\_time_{old}}{execution\_time_{new}} = \frac{1}{(1 - fraction_{enhanced}) + \frac{fraction_{enhanced}}{speedup_{enhanced}}}$$

- If the speed of a CPU is improved by a factor of 5 and the CPU requires 40% of the machines execution time, then what is the overall speedup?

Trends in Implementation Technology

- Transistor count on a chip is increasing by about 40% to 55% a year, or doubling every 18 to 24 months (Moore's law).
- DRAM capacity per chip is increasing by about 25% to 40% a year, doubling every two to three years.
- Flash capacity per chip is increasing by about 50% to 60% a year, doubling recently about every 1.5 years. Flash memory is 15 to 20 times cheaper per byte than DRAM.
- Disk density is increasing about 40% per year, doubling every two years. Disks per byte are 15 to 25 times cheaper than flash.

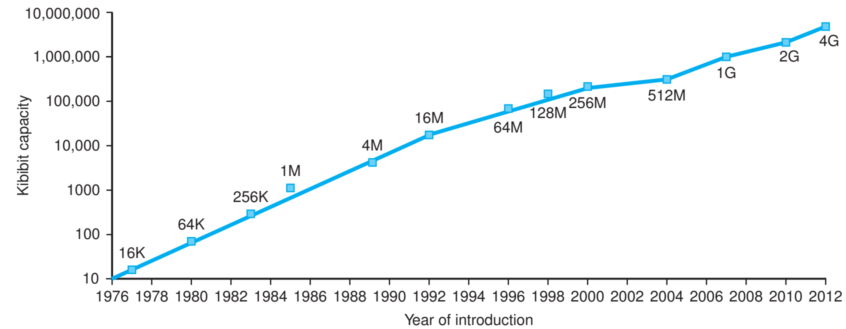
Benefits of Increasing Transistors on a Chip

- Increasing the number of transistors per chip has benefits.
  - Reduces chip manufacturing cost since less material is being used and it improves yield as die sizes decrease.
  - Improves performance since there is less distance for electricity to travel, which means the rate of executing machine instructions can increase.
- The table below estimates the improvement in the processor performance/cost ratio over the last 60+ years.

Year	Technology used in computers	Relative performance/unit cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit	900
1995	Very large-scale integrated circuit	2,400,000
2013	Ultra large-scale integrated circuit	250,000,000,000

Benefits of Increasing Transistors on a Chip (cont.)

- DRAM chips are also made of transistors.
- Increasing the number of transistors on a DRAM chip directly improves DRAM capacity as shown in the figure below.

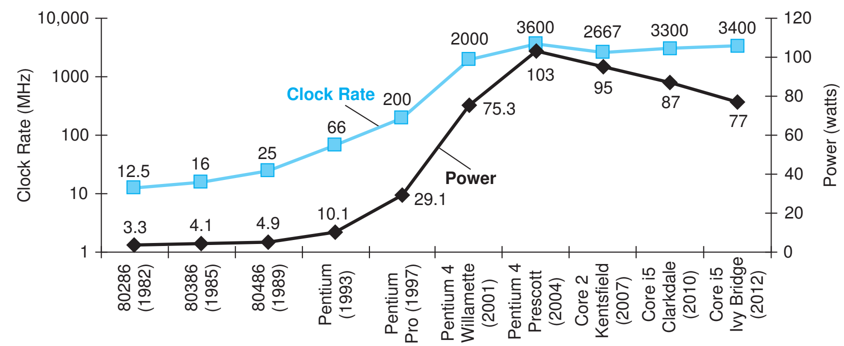


Effects of Dramatic Growth in Processing Performance

- Enhanced capability available to users.
- Led to new classes of computers.
- Led to dominance of microprocessor based computers.
- Allows programmers to trade performance for productivity.
- Nature of applications are also changing.

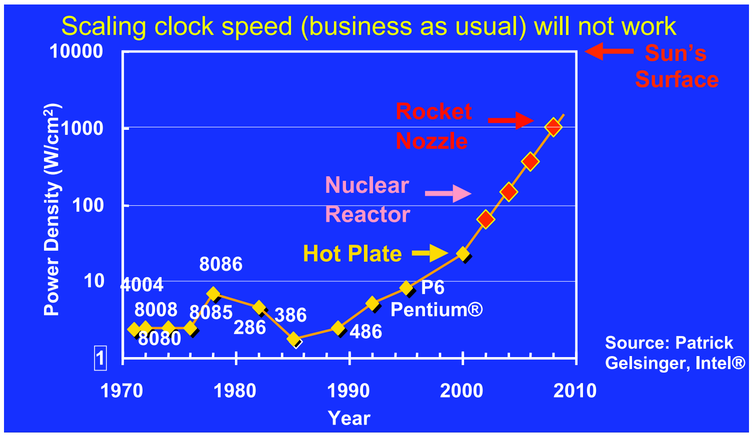
Need for Energy Efficient Processors

- Extend battery life for mobile systems.
- Reduce heat dissipation for general-purpose processors.
- Energy cost for computing is increasing.



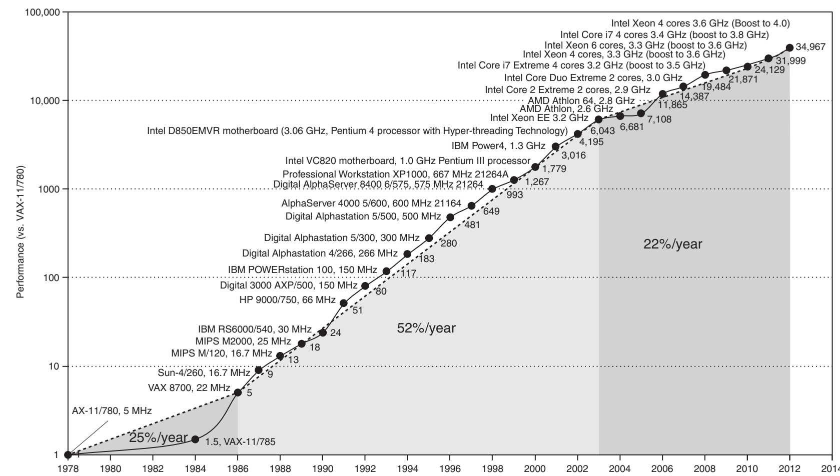
Hitting the Power Wall

- Clock rates have not kept increasing due to thermal constraints.



Processor Performance Improvements

- Performance improvements are slowing down in recent years.



Fallacies and Pitfalls

- Fallacy: a commonly held misbelief
  - Computers at low utitilization use little power.
  - Designing for performance and designing for energy are unrelated goals.
- Pitfall: an easily made mistake
  - Expecting the improvement of one aspect of a computer to increase overall performance by an amount proportional to the size of the improvement.
  - Using a subset of the performance equation as a performance metric.