Ch1: Computer Abstractions and Technology

Introduction

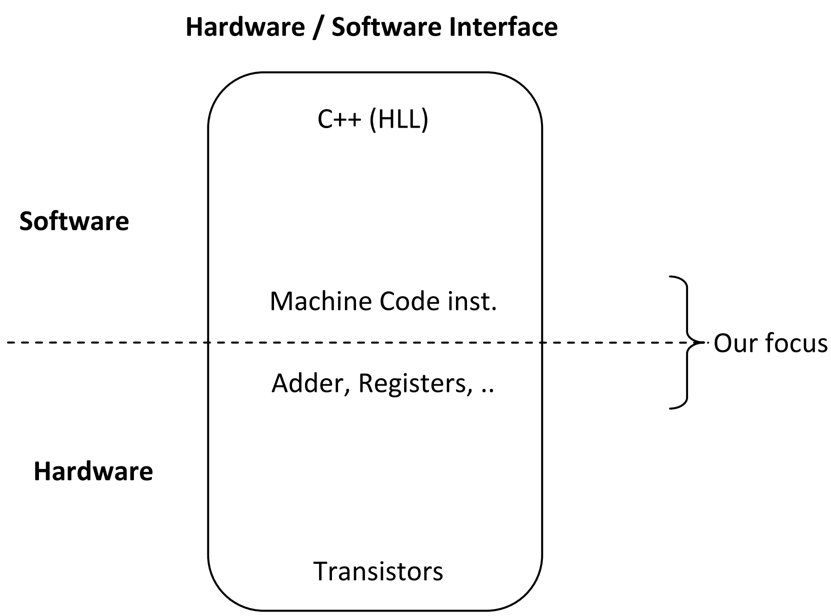
- This course is all about:
    - How computers work, basic principles.
    - How to analyze their performance.
    - How computers are designed and built.
    - Issues affecting modern processors (caches, pipelines, … etc.)

- Classes of computing applications and their characteristics:
    - Computers are used in three different classes of applications:
        1. Desktop computers: a computer designed for use by an individual, usually incorporating a graphics display, keyboard, and mouse.
        2. Servers: a computer used for running larger programs for multiple users often simultaneously and typically accessed only via a network.
            - Server types:
                - Mainframe
                - Minicomputer
                - Supercomputer: consist of hundreds to thousands of processors and usually gigabytes to terabytes of memory. (supercomputers are usually used for high-end scientific and engineering calculations. Ex: weather forecasting)
        3. Embedded computers: a computer inside another device. Used for running one predetermined application or collection of software.
            - Embedded computers include the microprocessors found in your car, the computers in a cell phone, or personal digital assistant.
- Why different processors? What is the difference between processors used in desktop, mobile, …. Etc?
    - Performance / Speed
    - Power consumption
    - Cost
    - General purpose / Special purpose
- Both hardware and software affect performance:
    - Algorithm determines number of source-level statements and the number of I/O operations executed.
    - Language / Compiler / Architecture determine the number of machine instructions for each source-level statement.
    - Processor / Memory determine how fast instructions are executed.
    - I/O system (hardware and operating system) determines how fast I/O operations may be executed.

- Computer Basic Components:
    - Inputs: writes instructions and data to memory.
    - Outputs: reads data from memory.
    - Memory: stores instructions and data.
    - Processor, which consists of:
        1. Datapath: processes data according to instructions.
        2. Control: commands the operations of input, output, memory and datapath according to instructions.

Levels of abstraction

- Impossible to understand computer components by looking every single transistor. Instead, abstraction is needed.
- Key ideas:
    - Both hardware and software are organized into hierarchical layers.
    - Hierarchical organization helps to cope with system complexity.
    - Lower-level details are hidden to offer a simple view at higher levels.
    - Interaction between levels occurs only through well-defined interface.

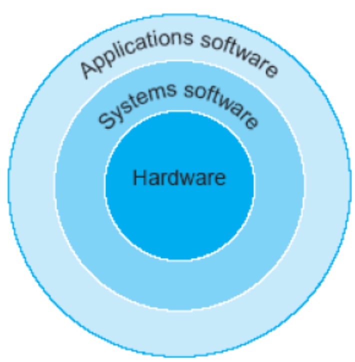


- How do computers work?
    - Need to understand abstractions such as:
        - Application software
        - System software
        - Assembly language
        - Machine language

        - Architectural issues: cache, pipelining, ….
        - Sequential logic, finite state machines
        - Combinational logic, arithmetic circuits
        - Boolean logic, 1s and 0s
        - Transistors used to build logic gates (CMOS)
        - Semiconductors / Silicon used to build transistors
        - Properties of atoms, electrons, and quantum dynamics
    - So much to learn!

Below your program

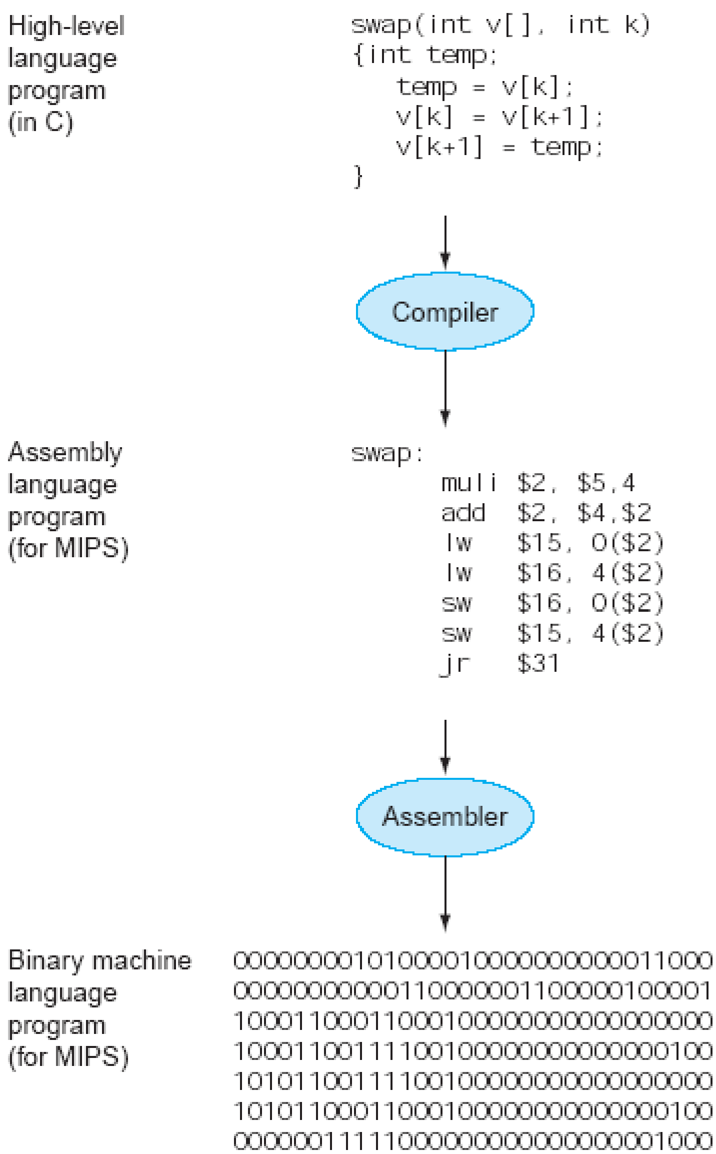
- Applications consist of hundreds of thousands to millions of lines of code.
- The hardware in a computer can only execute extremely simple low level instructions.
- To go from a complex application to the simple instructions involves several layers of software that interpret or translate high-level applications into simple computer instructions.
- System software: software that provides services that is commonly useful, including operating systems, compilers and assemblers.
    - Operating system: supervising program that manages the resources of a computer for the benefit of the programs that run on that machine.
    - Compiler: a program that translates high-level language statements into assembly language statements.



From a high-level language to the language of hardware

- To actually speak to an electronic machine, you need to send electrical signals.
- Computers are slaves to our commands, which are called instructions.
- Instructions, which are just collections of bits that the computer understands, can be thought of as a numbers.
- High-level programming language: a portable language (such as C, Java …) composed of words and algebraic notation that can be translated by a compiler into assembly language.

- Assembly language: a symbolic representation of machine instructions.
- Assembler: a program that translates a symbolic version of instructions into the binary version.
- Example:



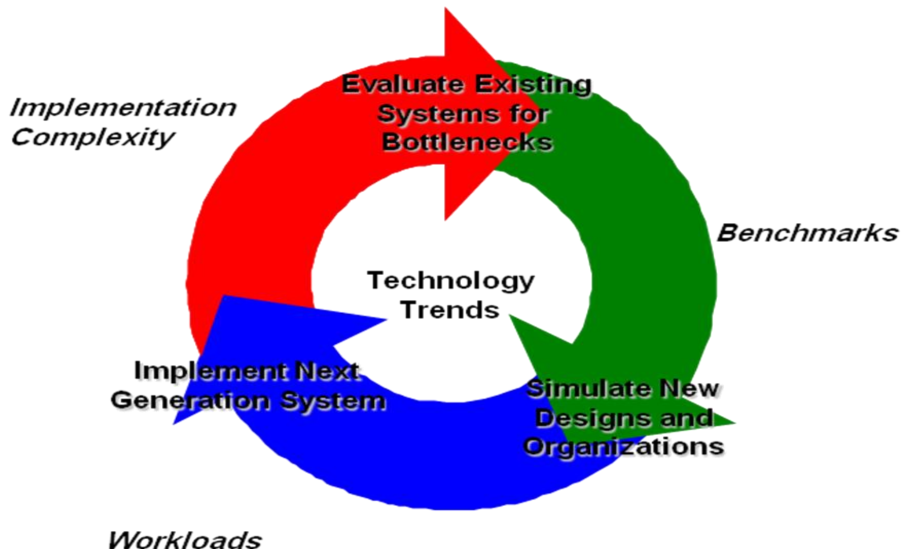
Computer Architecture

- A modern meaning of the term computer architecture covers three aspects of computer design:
    - Instruction Set Architecture (ISA),
    - Computer Organization and
    - Computer Hardware

- Instruction set architecture- ISA refers to the actual programmer-visible machine interface such as instruction set, registers, memory organization and exception handling.
    - Two main approaches:
        1. RISC (Reduced Instruction Set Computer) architecture
        2. CISC (Complex (and powerful) Instruction Set Computer) architecture
    - ISA is a boundary between HW and SW.
- Computer organization and computer hardware are two components of the implementation of a machine.
    - Computer organization includes the high-level aspects of a design, such as the memory system, the bus structure, and the design of the internal CPU (where arithmetic, logic, branching and data transfers are implemented)
    - Computer hardware refers to the specifies of a machine, included the detailed logic design and the packing technology of the machine.
- Computer Architecture = ISA + Organization + Hardware.
- Computer architecture Vs. computer organization
    - You could have the same instruction set architecture but different organizations (NEC5432 and NEC4122).
    - You could have the same ISA and same organization, but different HW implementations (Pentium Vs. Celeron).

Tasks of computer Architects

- Computer architects must design a computer to meet functional requirements as well as price, power, and performance goals. Often, they also have to determine what the functional requirements are, which can be a major task.
- Once a set of functional requirements has been established, the architect must try to optimize the design. There are three major application areas and their main requirements:
    - Desktop: focus on optimizing cost-performance as measured by a single user, with little regard for program size or power consumption.
        - Optimized for price-performance.
    - Server: focus on availability, scalability, throughput, and cost-performance.
    - Embedded: driven by price and often power issues, plus code-size is important.
        - Optimized for price, power, specialized performance.



- Benchmark: is special software to evaluate the performance of a particular system architecture.

RISC and CISC Architecture

- After 1985, any computer announced has been of RISC architecture.
- RISC ISA characteristics
    - All operations on data apply to data in registers and typically change the entire register.
    - The only operations that affect memory are load and store operations that move data from memory to a register or to memory from a register, respectively.
    - A small number of memory addressing modes.
    - The instruction formats are few in number with all instructions typically being one size.
    - Large number of registers.
- MIPS (stands for Microprocessor without Interlocked Pipeline Stages) processor is one of the first RISC processors.
- The main example of CISC processor is Intel IA_32 processors (in over 90% computers).
    - Intel IA_32 processors, from 80386 processor to Pentium IV today, and the next one to be introduced this or next year, are of CISC architecture.
- Intel IA_32 processors
    - Since 1995, Pentium processors consist of a front-end processor and a RISC-style processor.
    - The front-end processor fetches and decodes Intel IA_32 complex instructions and maps them into microinstructions.
    - A microinstruction is a simple instruction used in sequence to implement a more complex instruction. Microinstructions look very much as RISC instructions.
    - Then, the RISC-style processor executes microinstructions.

Comparing RISC and CISC

- MIPS is an example of RISC (reduced instruction set computer). Most existing processors are based on RISC because it is more promising.
- Another approach is CISC (complex instruction set computer):
    - Advantage:
        - provide more powerful operations
        - Reduce the number of instructions in a program.
    - Disadvantages:
        - Increase the time it takes to execute a program.
        - Make the processor hardware more complex and hence the speed of the processor slower.

RISC	CISC
Through quantitative measurements, choose only the most useful instructions and addressing modes.	Choose instructions and addressing modes that make the translation of high-level languages to assembly language simpler.
With few instructions and addressing modes, we can directly execute them in hardware.	Since we can have many instructions and addressing modes, we need a **microcode** (or **microprogrammed control**) to execute them in hardware.
A lot of chip space can be left for a large number of registers and cache memory.	We can have only few registers and small cache memory.
Compilers are more difficult to write.	Compilers are easier to write.
Assembly language programs are more difficult to write.	Assembly language programs are easier to write.

- IA-32 Overview
    - Complexity:
        - Instructions from 1 to 17 bytes long
        - one operand must act as both a source and destination
        - one operand can come from memory
        - complex addressing modes

          e.g., “base or scaled index with 8 or 32 bit displacement”