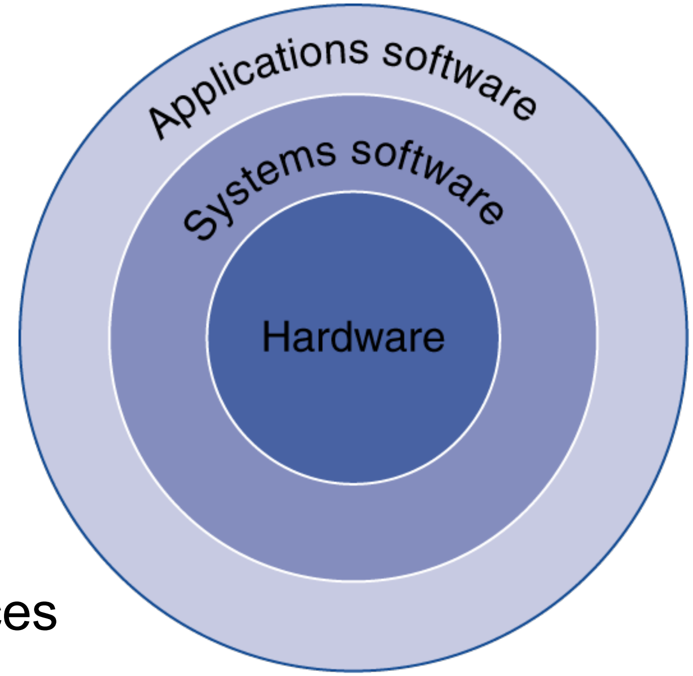
Computer Abstraction

Software
Hardware

Computer Abstraction

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



SW abstraction: Levels of Program Code

High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

Compiler

Assembly language program (for MIPS)

```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

Assembler

Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

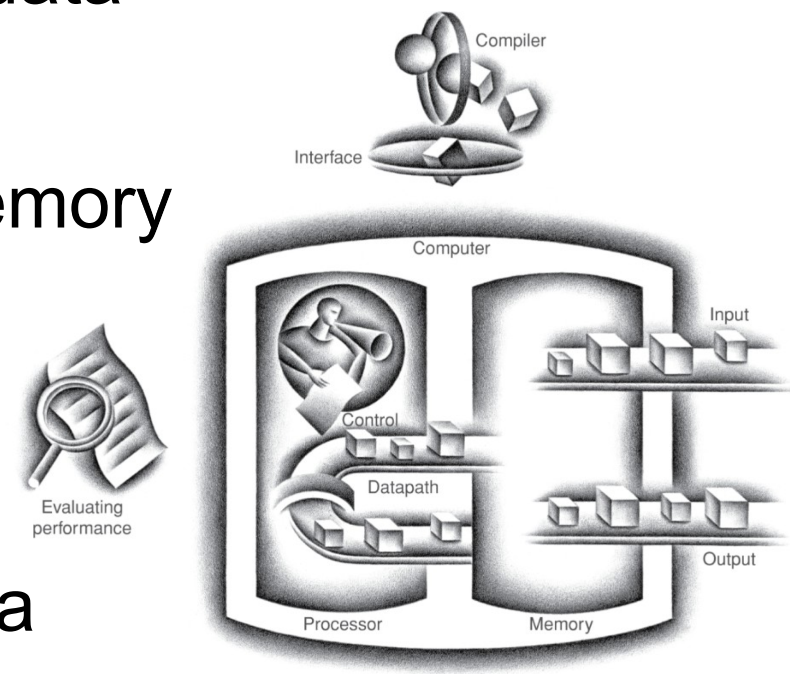
- High-level language
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- Assembly language
  - Textual representation of instructions
- Machine language
  - Binary digits (bits)
  - Encoded instructions and data

Instruction Set Architecture (ISA)

- The ISA is the key interface between the hardware and low-level software
  - Defines everything needed to write a machine language program
    - Data types, machine state, addressing mode, instruction set, I/O model
- Any ISA may have different implementations of varying cost and performance

Hardware Operation Overview

- Datapath:
  - performs operations on data
- Control:
  - sequences datapath, memory access
- Cache memory
  - Small fast memory for immediate access to data



Computer Performance

Objectives

- Define key metrics used for measuring computer performance
- Understand how computer performance can be improved
- Identify the challenges for improving the computer performance
- Present how the computer design is changed to accommodate these challenges

Key Performance Metrics

- **When we say computer A is better than Computer B?**
- ***Response time***
  - Also referred to as execution time
  - How long does it take to complete a task?
- ***Throughput***
  - Total work done per unit time
    - e.g., tasks/transactions/… per hour
  - Focus of servers

Discussion

- How are response time and throughput affected by
  - Replacing the processor with a faster version?
  - Adding more processors?

Relative Performance

- Define **Performance = 1/Execution Time**
- "X is n time faster than Y"

$$\text{Performance}_X / \text{Performance}_Y$$
$$= \text{Executiontime}_Y / \text{Executiontime}_X = n$$

- Example: time taken to run a program
  - 10s on A, 15s on B. How much faster is A than B?

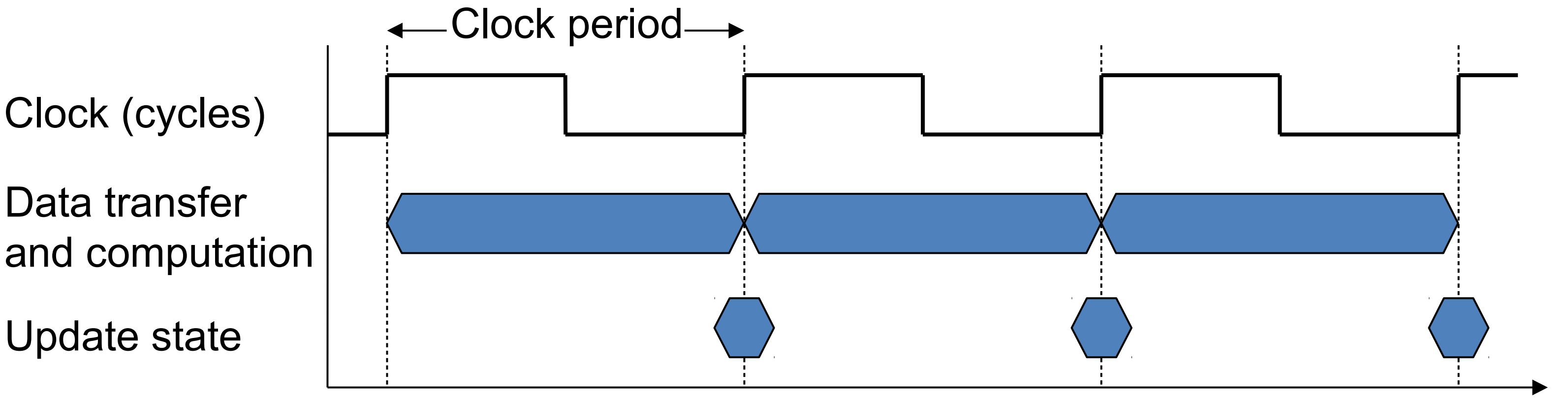
- Execution Time_B / Execution Time_A = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

Measuring Performance

- Elapsed time
  - Total time to complete a task, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines ***system performance***
- CPU time
  - Time spent processing a given job
    - Does not include: I/O time, other jobs' shares
  - Comprises user CPU time and system CPU time
- Different programs are affected differently by CPU and system performance

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock



- ***Clock period***: duration of a clock cycle
  - E.g., 250ps = 0.25ns = 250×10^{-12}s
- ***Clock frequency (rate)***: cycles per second
  - E.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

CPU Performance

- CPU execution time for a program

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$
$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate
- Hardware designer must often trade off clock rate against cycle count

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles
- How fast must Computer B clock rate be?

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles
- How fast must Computer B clock rate be?

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6s}$$

$$\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A$$

$$= 10s \times 2GHz = 20 \times 10^9$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6s} = \frac{24 \times 10^9}{6s} = 4GHz$$

Instruction Count and Cycles Per Instruction

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- Instruction Count for a program
  - Determined by program, ISA and compiler
- Average cycles per instruction
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

$$\begin{aligned}\text{CPU Time}_A &= \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A \\ &= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}\end{aligned}$$

A is faster…

$$\begin{aligned}\text{CPU Time}_B &= \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B \\ &= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}\end{aligned}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$

…by this much

Performance BIG Picture

- ***Time is the only complete and reliable measure of performance***

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Performance depends on
  - Algorithm: affects IC, possibly CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - Instruction set architecture: affects IC, CPI, T_c

Reading

- Sections 1.6, 1.7, 1.8, 1.9