MEASURING PERFORMANCE

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

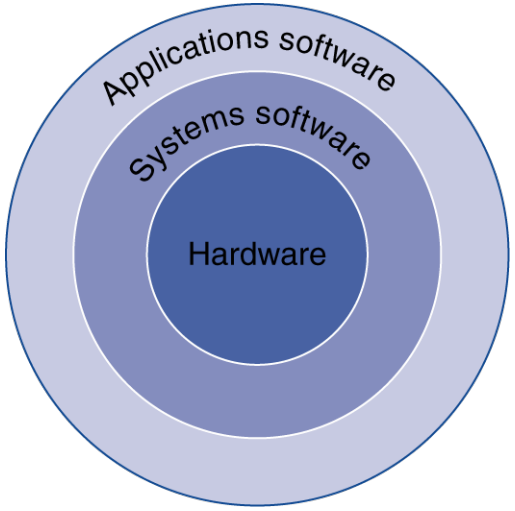
University of Utah

Overview

- This lecture
  - Levels of program code
  - Chips process
  - Performance
  - Design principles

Below Your Program

- 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 and sharing resources
- Hardware
  - Processor, memory, I/O controllers



Levels of Program Code

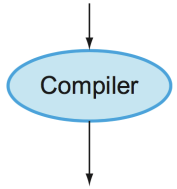
- High-level language
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability

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

Levels of Program Code

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

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




```
swap:
      multi $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
```

Levels of Program Code

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

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

Compiler

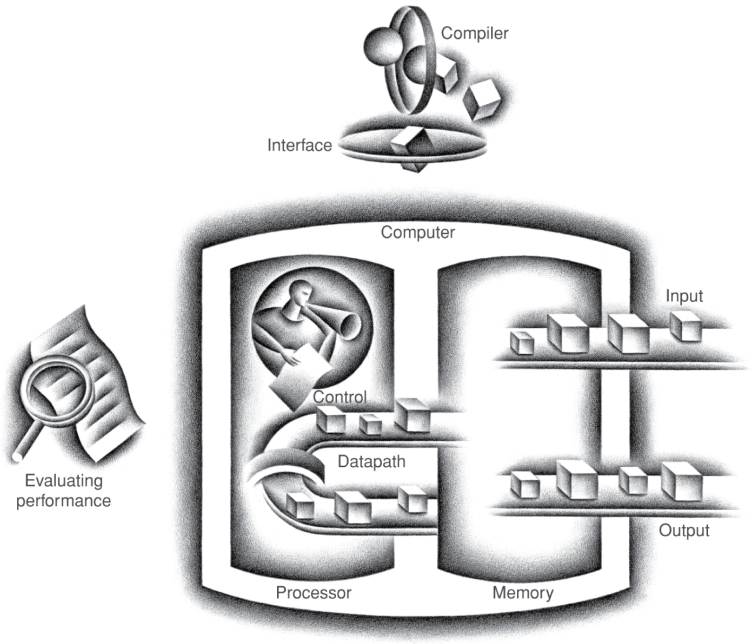
```
swap:
      multi $2, $5,4
      add   $2, $4,$2
      lw    $15, 0($2)
      lw    $16, 4($2)
      sw    $16, 0($2)
      sw    $15, 4($2)
      jr    $31
```

Assembler

```
00000000101000100000000100011000
00000000100000100001000000100001
10001101111000100000000000000000
10001110000100100000000000000100
10101110000100100000000000000000
10101101111000100000000000000100
00000011111000000000000000001000
```

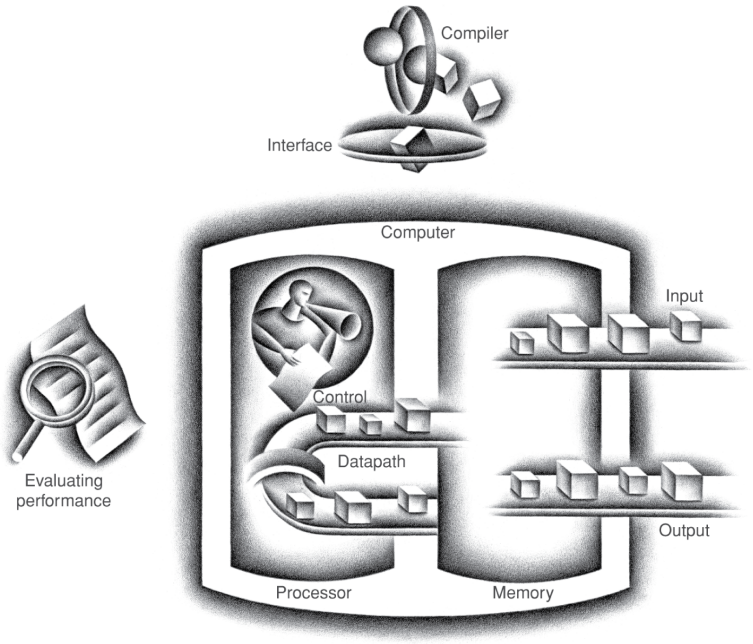
Components of Computer

- Same components for all kinds of computer
  - Desktop, server, embedded
- Input/output includes
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers



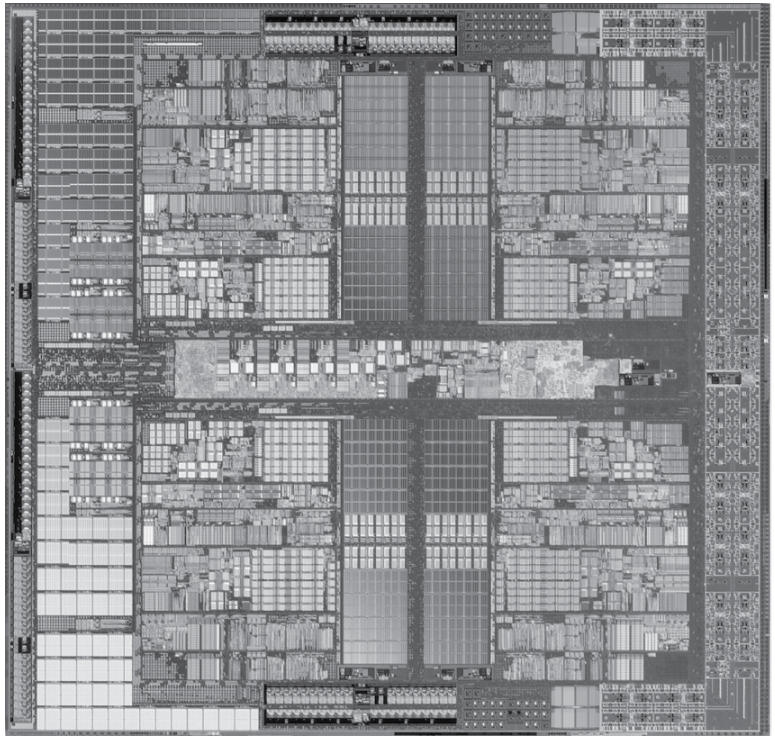
Inside Processor (CPU)

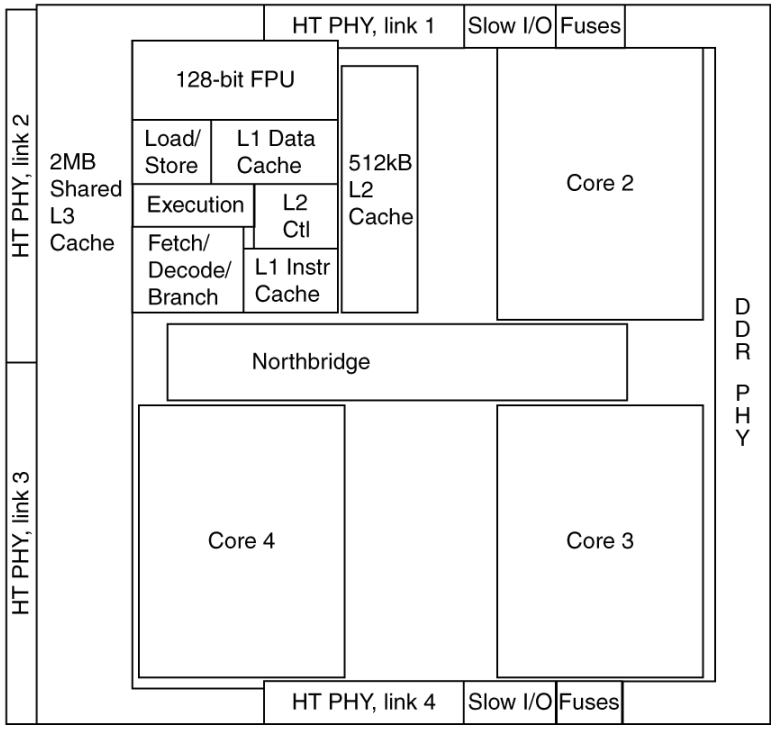
- Datapath
  - performs operations on data
- Control
  - sequences datapath, memory, ...
- Cache memory
  - Small fast SRAM memory for immediate access to data



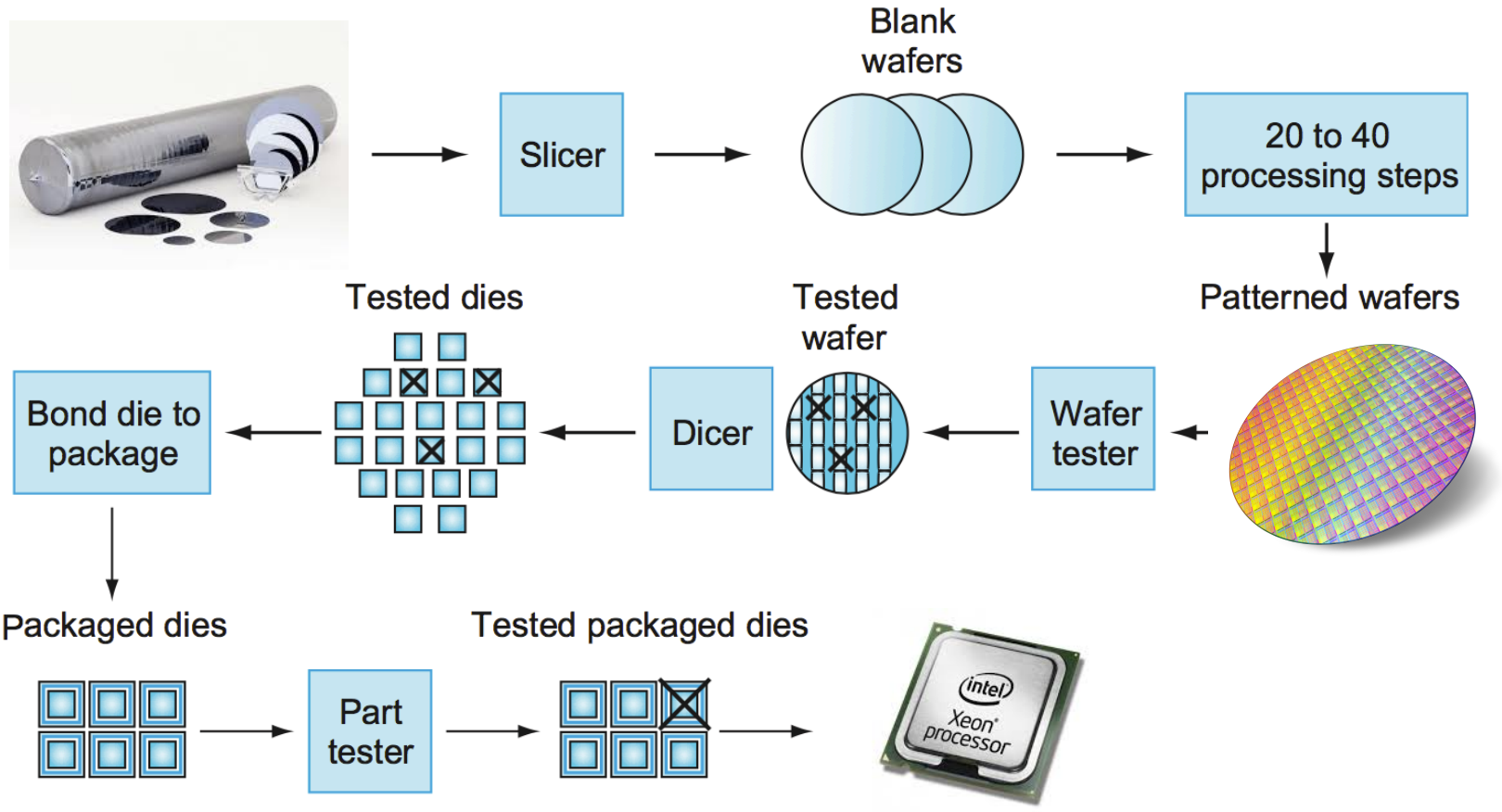
Inside Processor (CPU)

- AMD Barcelona: four processor cores





The Chip Manufacturing Process



Measuring Performance

- How to measure performance?
  - Latency or response time
    - The time between start and completion of an event (e.g., milliseconds for disk access)
  - Bandwidth or throughput
    - The total amount of work done in a given time (e.g., megabytes per second for disk transfer)

Measuring Performance

- How to measure performance?
  - Latency or response time
    - The time between start and completion of an event (e.g., milliseconds for disk access)
  - Bandwidth or throughput
    - The total amount of work done in a given time (e.g., megabytes per second for disk transfer)
- Which one is better? latency or throughput?

Measuring Performance

- Which one is better (faster)?

- Delay=10m
- Capacity=4p

- Delay=30m
- Capacity=30p

Measuring Performance

- Which one is better (faster)?

Car	Bus
Delay=10m	Delay=30m
Capacity=4p	Capacity=30p
Throughput=0.4PPM	Throughput=1PPM

It really depends on your needs (goals).

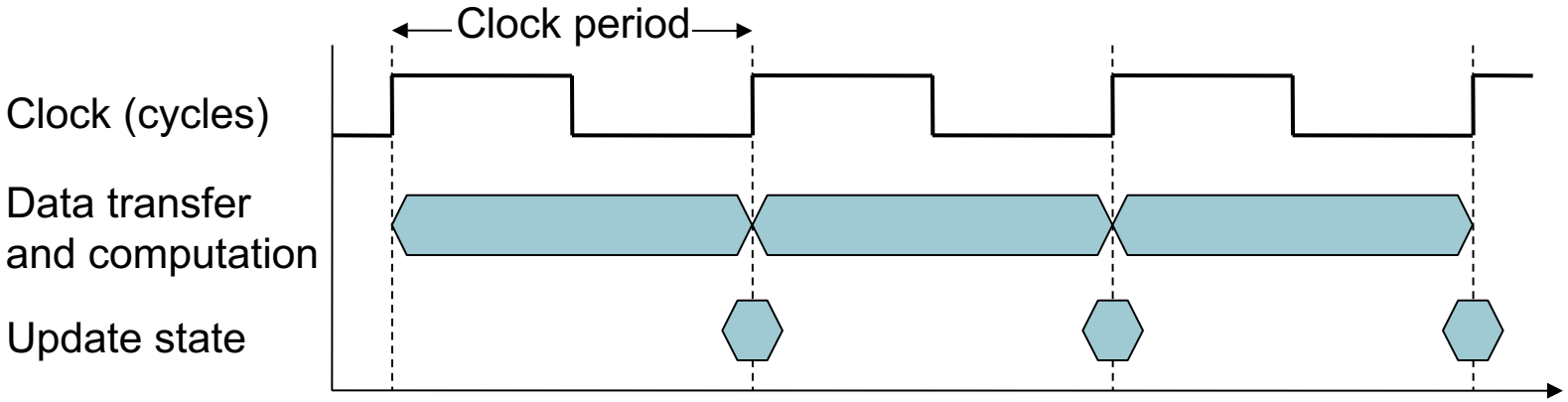
Measuring Execution Time

- Elapsed time
  - Total response time, including all aspects
  - Processing, I/O, OS overhead, idle time
  - Determines system performance

- CPU time
  - Time spent processing a given job
  - Discounts I/O time, other jobs' shares
  - Comprises user CPU time and system CPU time
  - Different programs are affected differently by CPU and system performance

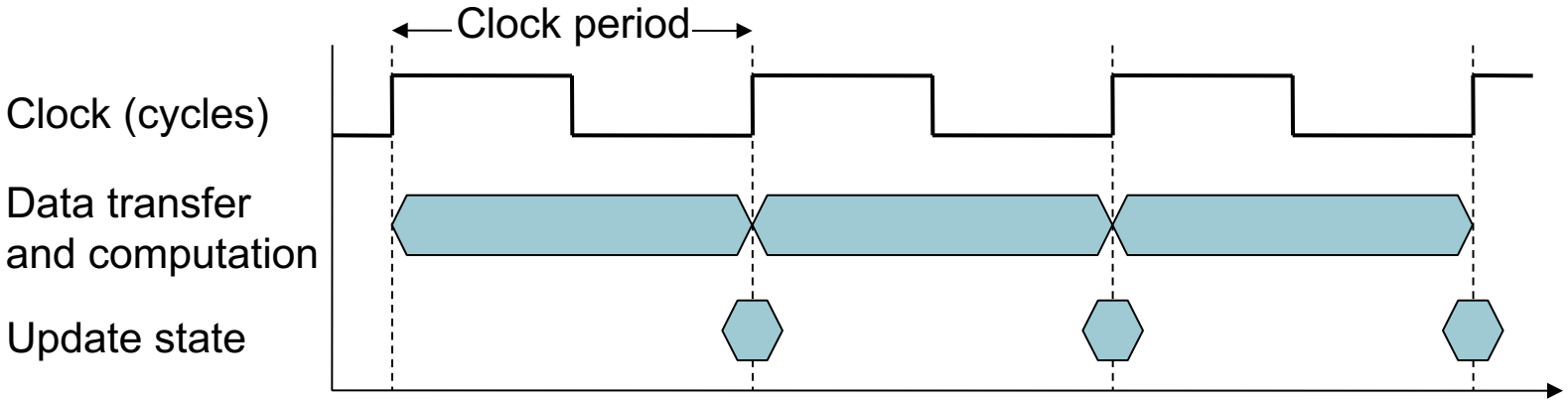
Clocking and Cycle Time

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



Clocking and Cycle Time

- 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

The Processor Performance

- Clock cycle time (CT = 1/clock frequency)
  - Influenced by technology and pipeline
- Cycles per instruction (CPI)
  - Influenced by architecture
  - IPC may be used instead (IPC = 1/CPI)
- Instruction count (IC)
  - Influenced by ISA and compiler
- CPU time = IC x CPI x CT
- Performance = 1/Execution Time

Speedup vs. Percentage

- Speedup = old execution time / new execution time
- Improvement = (new performance - old performance)/old performance
- My old and new computers run a particular program in 80 and 60 seconds; compute the followings
  - speedup
  - percentage increase in performance
  - reduction in execution time

Speedup vs. Percentage

- Speedup = old execution time / new execution time
- Improvement = (new performance - old performance)/old performance
- My old and new computers run a particular program in 80 and 60 seconds; compute the followings
  - speedup = 80/60
  - percentage increase in performance = 33%
  - reduction in execution time = 20/80 = 25%

Principles of Computer Design

- Designing better computer systems requires better utilization of resources
  - Parallelism
    - Multiple units for executing partial or complete tasks
  - Principle of locality (temporal and spatial)
    - Reuse data and functional units
  - Common Case
    - Use additional resources to improve the common case