CO 01 Computer Abstraction and Technology

1 Introduction

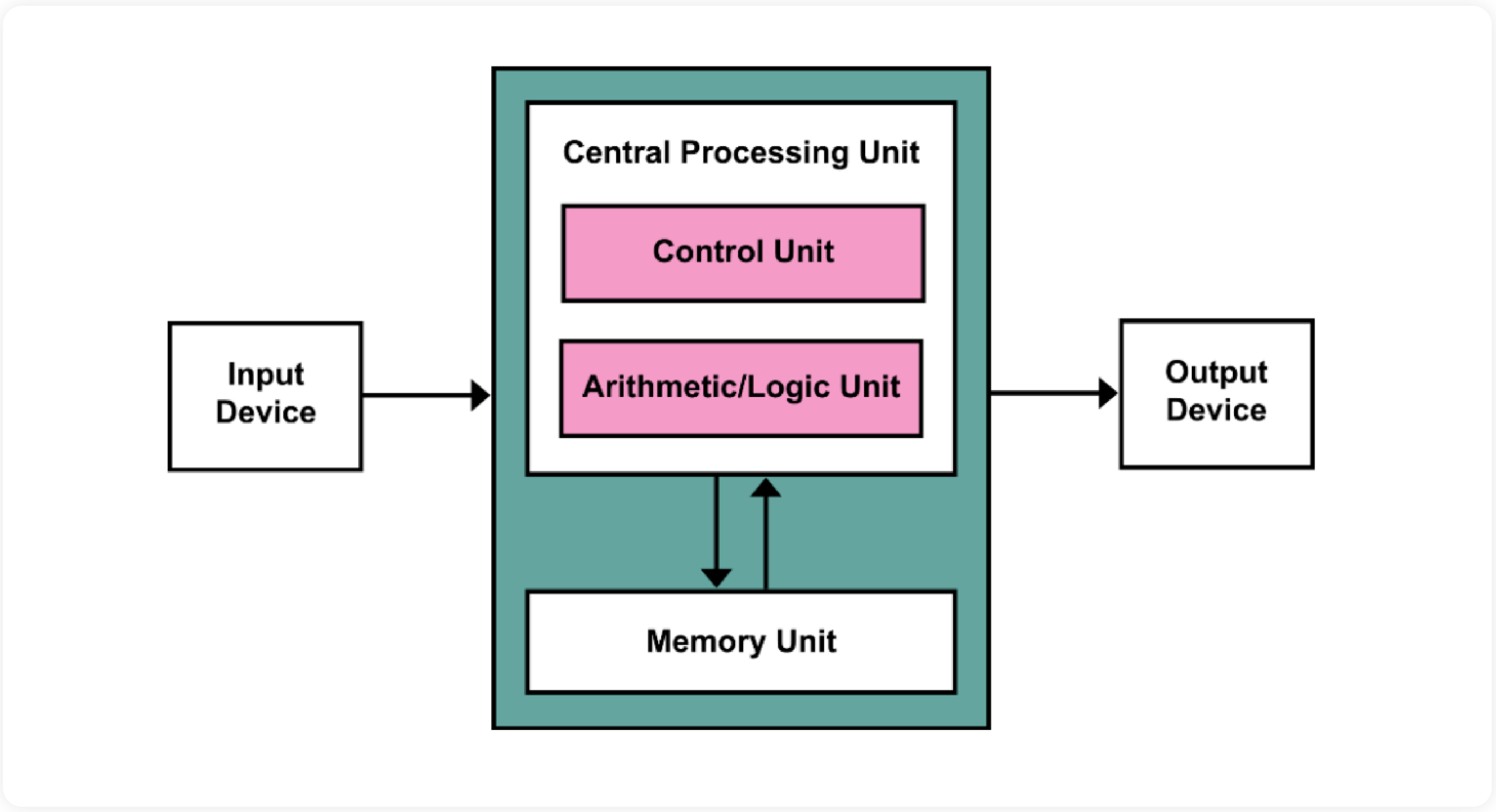
1.1 Some Key Words

- inst set
- datapath
- pipelines
- virtual memory
- cache

1.2 Structure of a Computer

- Computer/System
  - Software
  - Hardware
    - CPU
      - Control unit
      - Datapath
        - Path (MUX)
        - ALU: adder, multiplier
        - Registers
        - ...
    - Memory
    - I/O Interface

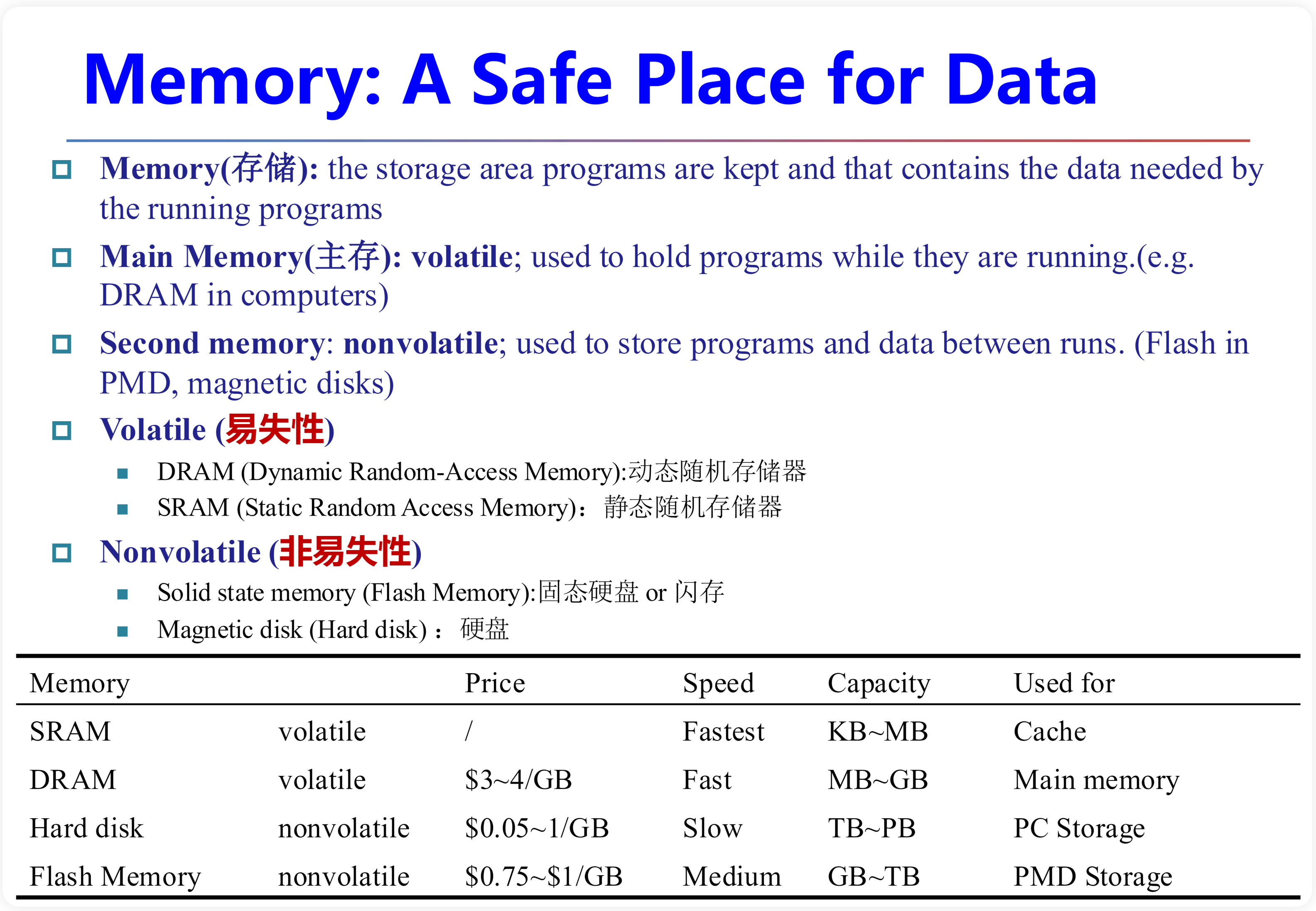
1.3 von Neumann Architecture



- 计算和存储分离
- 数据与指令保存在同一个存储器
- I/O 设备
- 指令集架构

2 Computer Organization

2.1 Memory



Memory: A Safe Place for Data

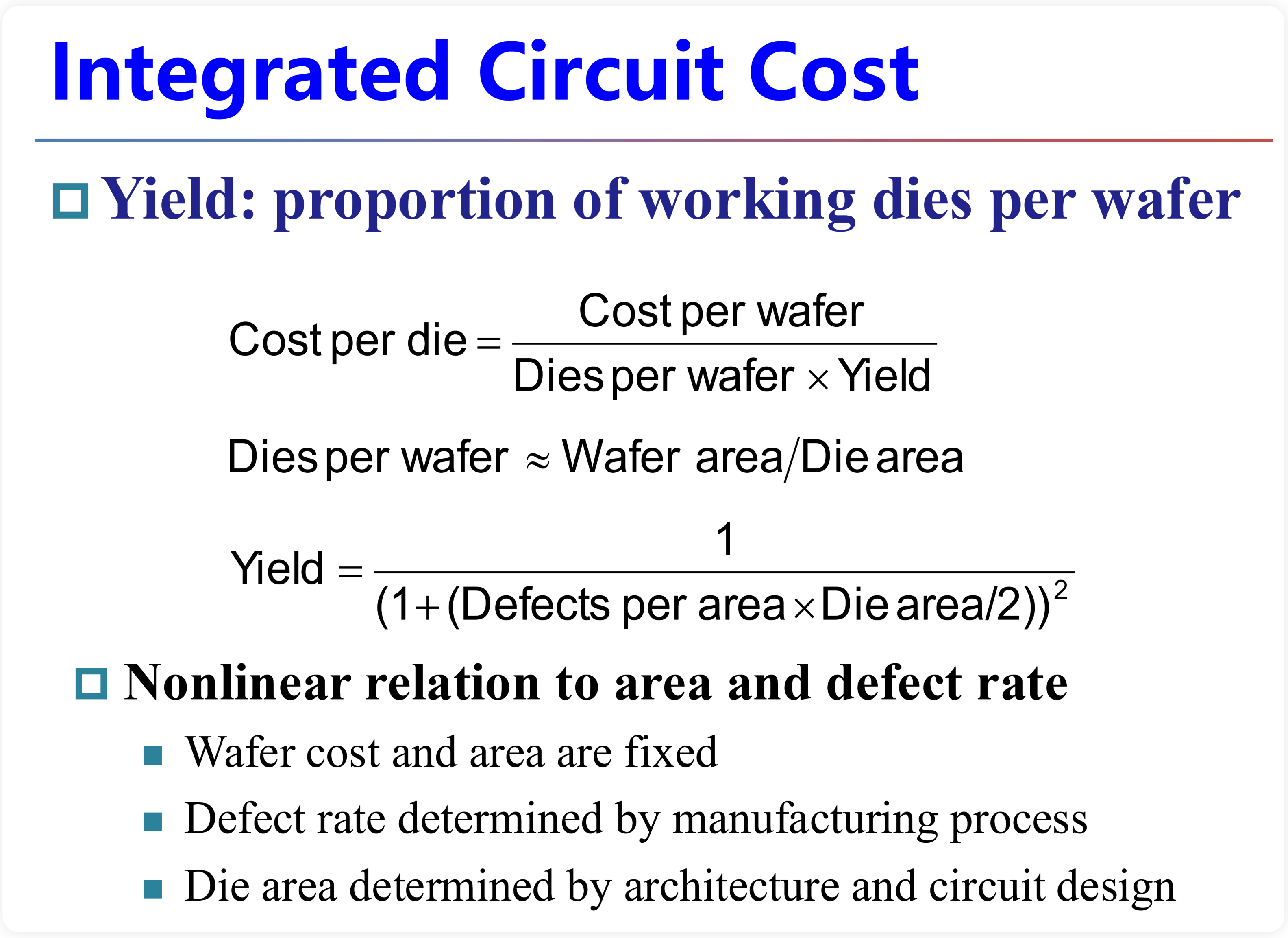
- **Memory(存储):** the storage area programs are kept and that contains the data needed by the running programs
- **Main Memory(主存): volatile**; used to hold programs while they are running.(e.g. DRAM in computers)
- **Second memory**: **nonvolatile**; used to store programs and data between runs. (Flash in PMD, magnetic disks)
- **Volatile (易失性)**
  - DRAM (Dynamic Random-Access Memory):动态随机存储器
  - SRAM (Static Random Access Memory)：静态随机存储器
- **Nonvolatile (非易失性)**
  - Solid state memory (Flash Memory):固态硬盘 or 闪存
  - Magnetic disk (Hard disk)：硬盘

Memory		Price	Speed	Capacity	Used for
SRAM	volatile	/	Fastest	KB~MB	Cache
DRAM	volatile	$3~4/GB	Fast	MB~GB	Main memory
Hard disk	nonvolatile	$0.05~1/GB	Slow	TB~PB	PC Storage
Flash Memory	nonvolatile	$0.75~$1/GB	Medium	GB~TB	PMD Storage

2.2 ISA

- 从高级硬件语言到底层硬件
- 只要支持同一套 ISA，硬件和软件就能配合

3 How to build processors?



Integrated Circuit Cost

- **Yield: proportion of working dies per wafer**

$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}$$

$$\text{Dies per wafer} \approx \text{Wafer area}/\text{Die area}$$

$$\text{Yield} = \frac{1}{(1+(\text{Defects per area} \times \text{Die area}/2))^2}$$

- **Nonlinear relation to area and defect rate**
  - Wafer cost and area are fixed
  - Defect rate determined by manufacturing process
  - Die area determined by architecture and circuit design

> 关于制造成本

4 Performance

4.1 Response Time and Throughput

- Response time/execution time *响应时间、执行时间，执行一个任务的时间是多长*

- Throughput (bandwidth) *吞吐率，单位时间能执行多少任务*
- Response time and throughput 的影响因素
  - 更好的处理器
  - 更多的处理器

4.2 Relative Performance

- $perf = 1/Exe\,time$
- *x is n times faster than y*

$$perf_x/perf_y = time_y/time_x = n$$

4.3 CPU Time

- Elapse Time 是总时间
- CPU Time 是除去 I/O 等其他因素的时间

4.4 Measuring Execution Time

- Clock period *时钟周期*
- Clock freq(rate) *时钟频率, cycles per sec*
- $$CPU\,Time = CPU\,Clock\,Cycles \times Clock\,Cycle\,Time = \frac{CPU\,CLock\,Cycles}{Clock\,Rate}$$
- 提升性能，可以减少周期数，也可以提升时钟频率

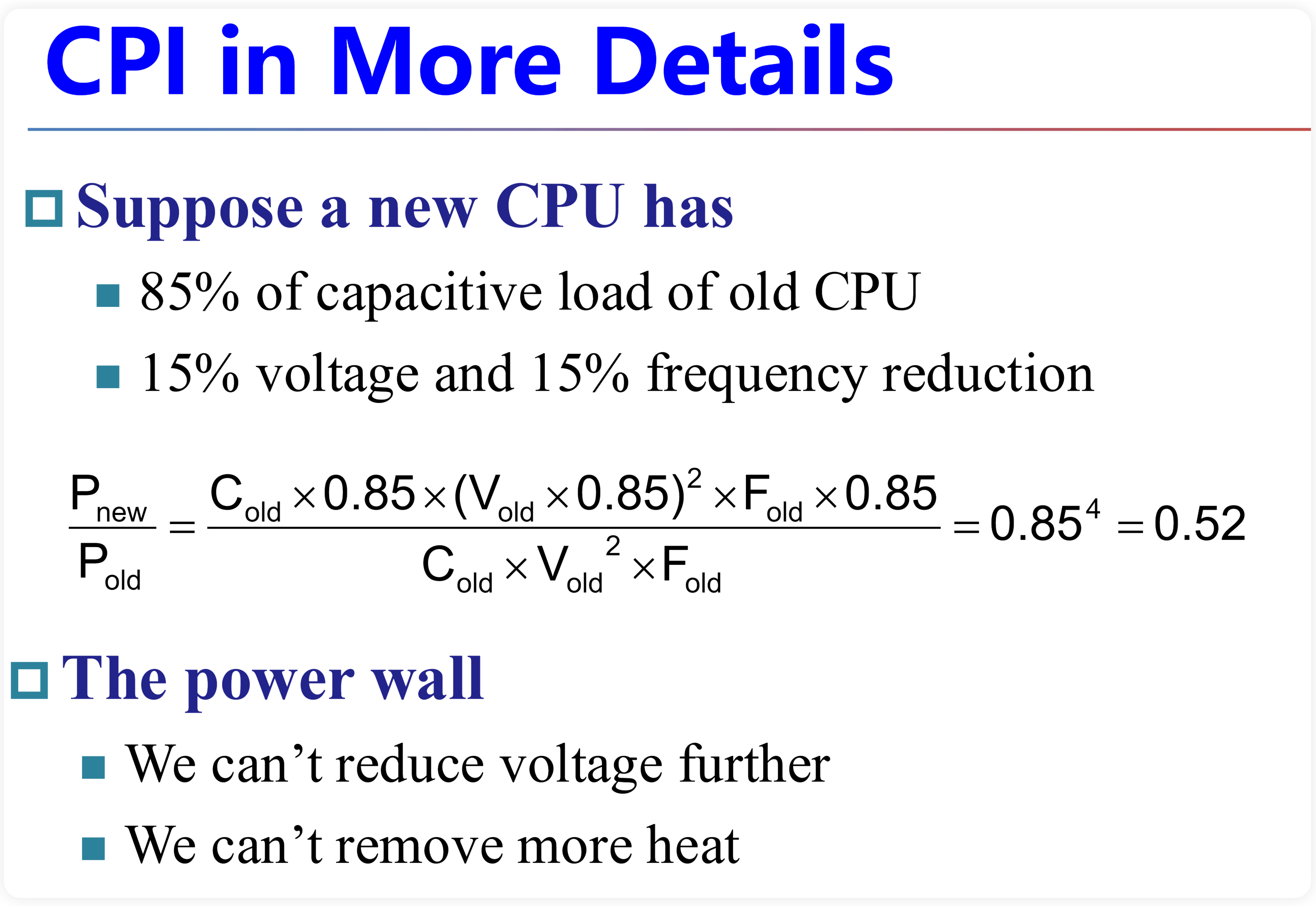
4.5 Inst Cnt and CPI

- $$Clock\,Cycles = Inst\,Cnt \times CPI$$
- $$CPU\,Time = Inst\,Cnt \times CPI \times Clock\,Cycle\,Time = \frac{Inst\,Cnt \times CPI}{Clock\,Rate}$$
- 同样的代码，如果 ISA 相同，那么 Inst Cnt 相同
- Weighted average CPI, 如果不同指令 CPI 不同，按照其 Inst Cnt 的比例分配权重

4.6 Perf depends on

1. Algo
2. Programming lang, compiler, architecture *ch.2, 3*
3. Processor and memory sys *ch.4, 5*
4. I/O system *ch. 6*

4.7 Power



CPI in More Details

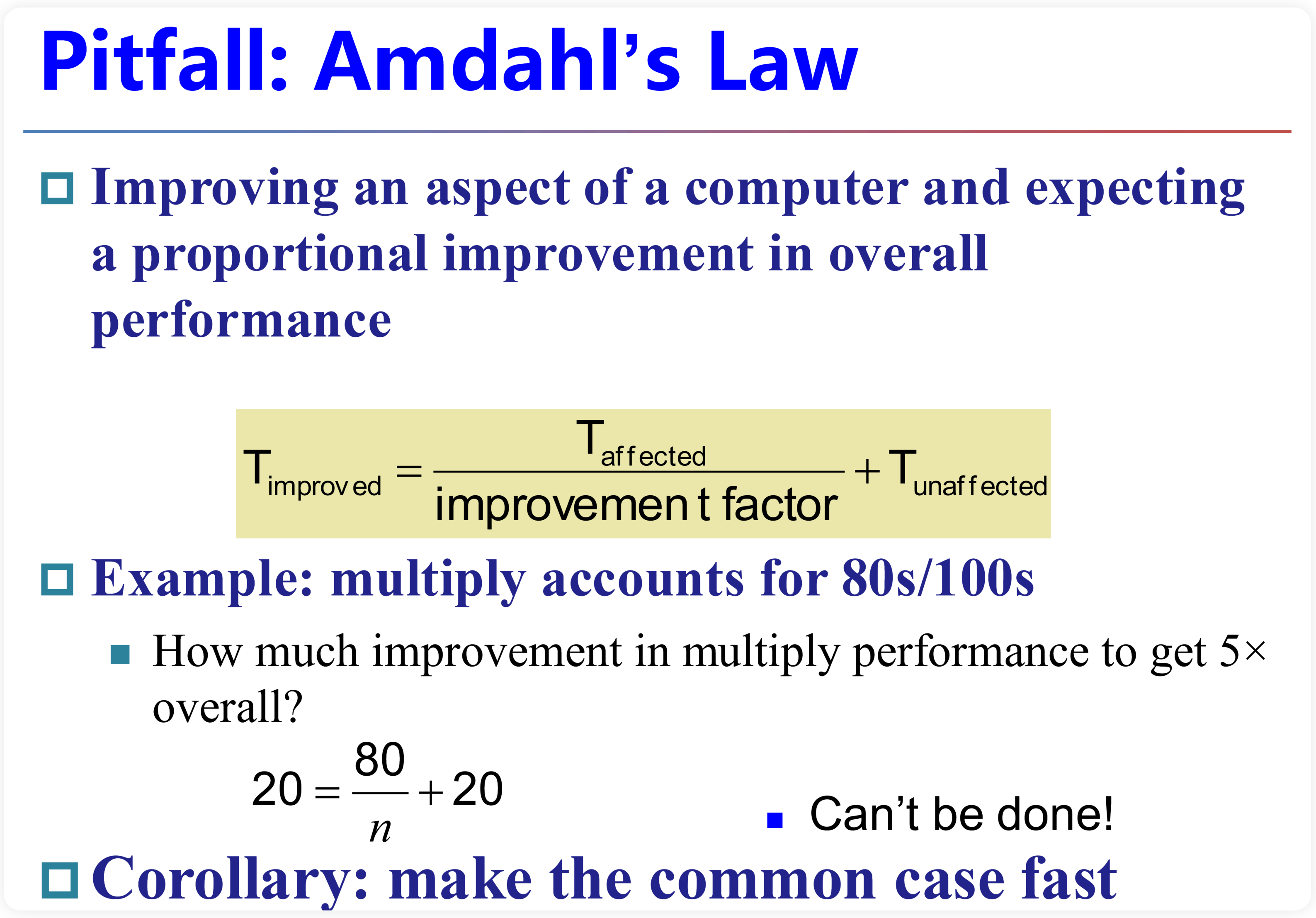
- Suppose a new CPU has
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction

$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}^2 \times F_{old}} = 0.85^4 = 0.52$$

- The power wall
  - We can’t reduce voltage further
  - We can’t remove more heat

> $P = C \times V^2 \times F$

4.8 Pitfall: Amdahl's Law



Pitfall: Amdahl’s Law

- Improving an aspect of a computer and expecting a proportional improvement in overall performance

$$T_{improved} = \frac{T_{affected}}{\text{improvement factor}} + T_{unaffected}$$

- Example: multiply accounts for 80s/100s
  - How much improvement in multiply performance to get 5× overall?

$$20 = \frac{80}{n} + 20$$

  - Can’t be done!
- Corollary: make the common case fast

> 提升部分性能对整体性能的优化具有上限

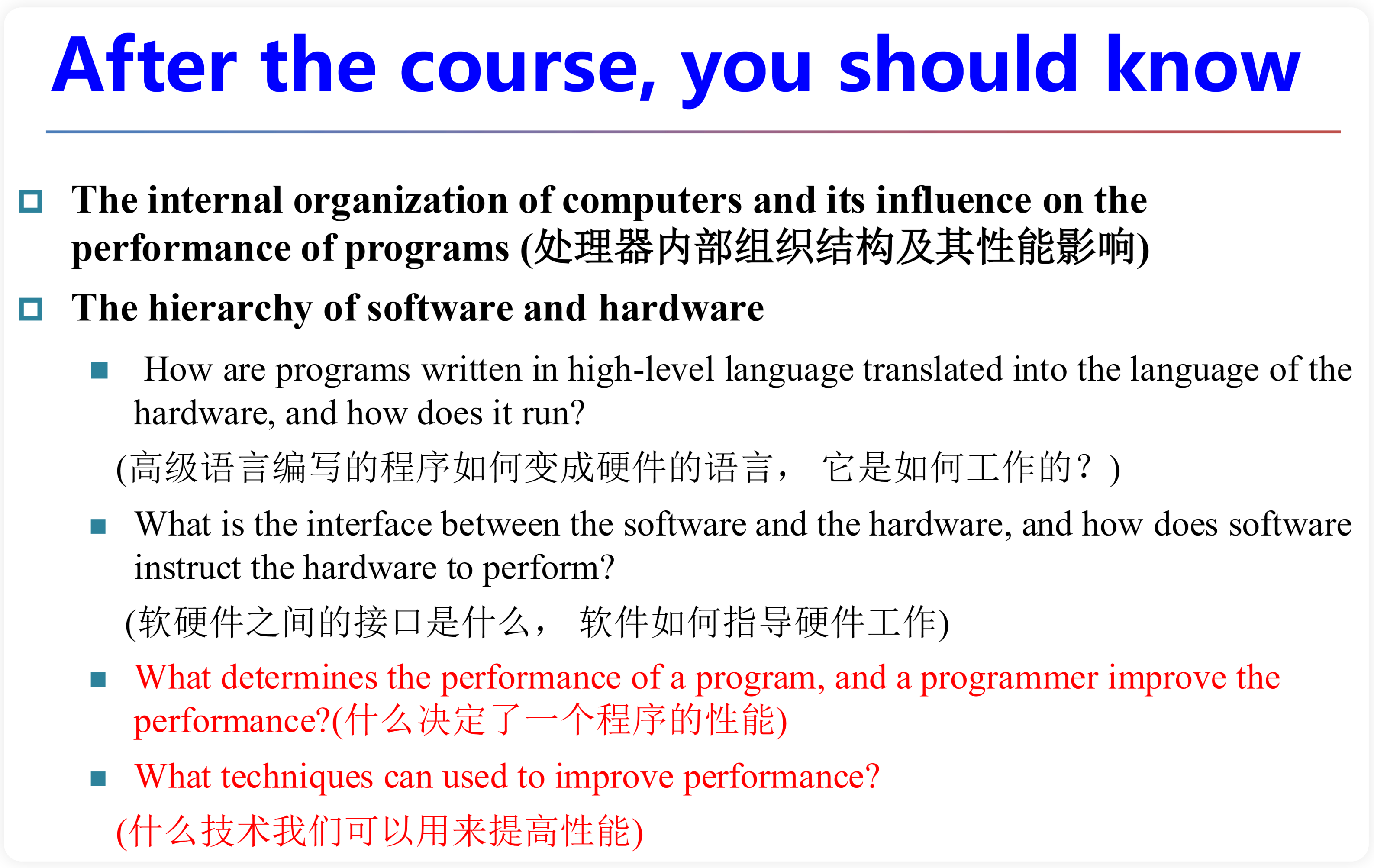
4.9 Pitfall: MIPS

- MIPS: Millions of Inst Per Second
- $$MIPS = \frac{Inst\ Cnt}{Exe\ Time \times 10^6} = \frac{Clock\ Rate}{CPI \times 10^6}$$

5 Eight Great Ideas

- Design for Moore's Law *设计紧跟摩尔定律*

    - Design for where it will be when **finishes** rather that design for where it starts.
- Use Abstraction to Simplify Design *采用抽象简化设计*
    - 层次化、模块化
    - e.g. ISA 作为标准
- Make the Common Case Fast *加速大概率事件*
    - 联系 Amdahl's Law
- Performance via Parallelism
    - multiprocessor
    - 增大位宽
- Performance via Pipelining
    - 最好情况下，每一个流程的时间都是均匀的
- Performance via Prediction
    - 分支预测
- Hierarchy of Memories *存储器层次*
    - Disk/Tape -> Main Memory(DRAM) -> L2-Cache(SRAM) -> L1-Cache(On-Chip) -> Registers
- Dependability via Redundancy *通过冗余提高可靠性*
    - 卡车的多个轮胎



After the course, you should know

- **The internal organization of computers and its influence on the performance of programs (处理器内部组织结构及其性能影响)**
- **The hierarchy of software and hardware**
    - How are programs written in high-level language translated into the language of the hardware, and how does it run?
      (高级语言编写的程序如何变成硬件的语言， 它是如何工作的？)
    - What is the interface between the software and the hardware, and how does software instruct the hardware to perform?
      (软硬件之间的接口是什么， 软件如何指导硬件工作)
    - What determines the performance of a program, and a programmer improve the performance?(什么决定了一个程序的性能)
    - What techniques can used to improve performance?
      (什么技术我们可以用来提高性能)