

# COMP4007: 并行处理和体系结构

## 第二章: 并行计算机体系结构

授课老师: 王强、施少怀  
助 教: 刘虎成、林稳翔

哈尔滨工业大学 (深圳)

# 内容大纲

- ▶ 现代计算机体系结构
- ▶ Flynn分类法
- ▶ 共享内存系统
- ▶ 分布式内存系统
- ▶ 向量化方法及SIMD



# 现代计算机体系结构

## ▶ CPU

- ▶ 中央处理单元
- ▶ 负责运行程序

## ▶ 内存 (or RAM)

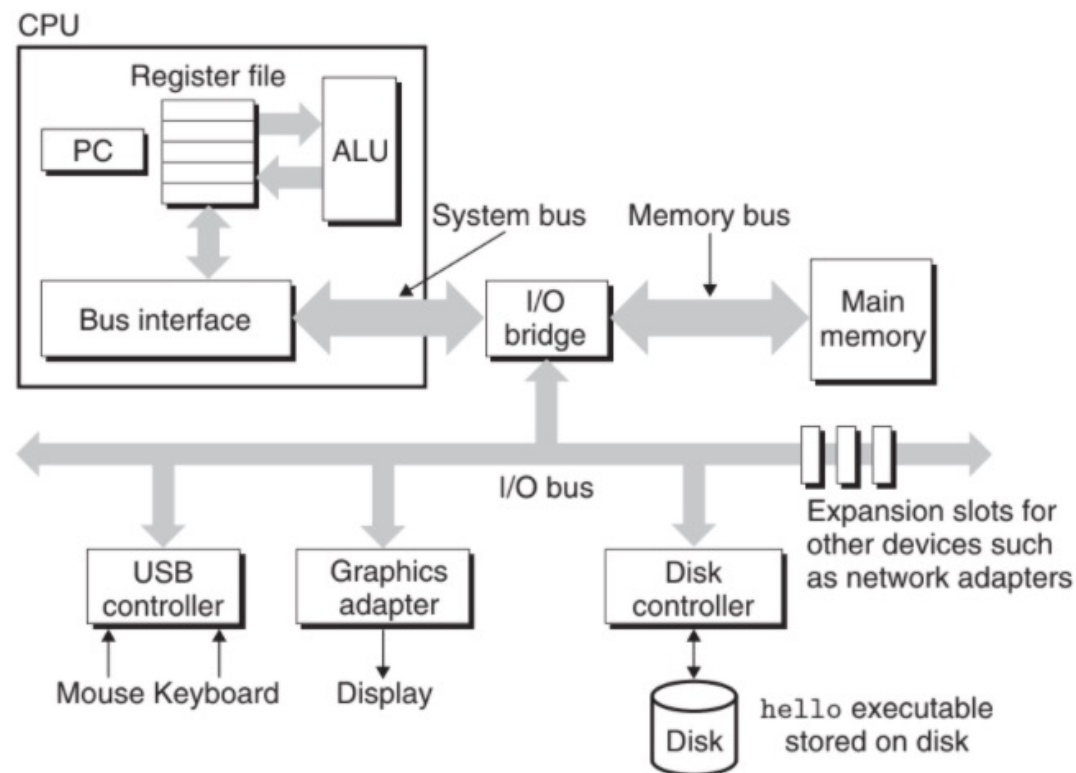
- ▶ 随机存取存储器
- ▶ 程序和数据的短暂存储

## ▶ 总线

- ▶ 数据的传输

## ▶ 输入输出 (I/O) 设备

- ▶ 磁盘：长期存储
- ▶ 鼠标、键盘、显示器

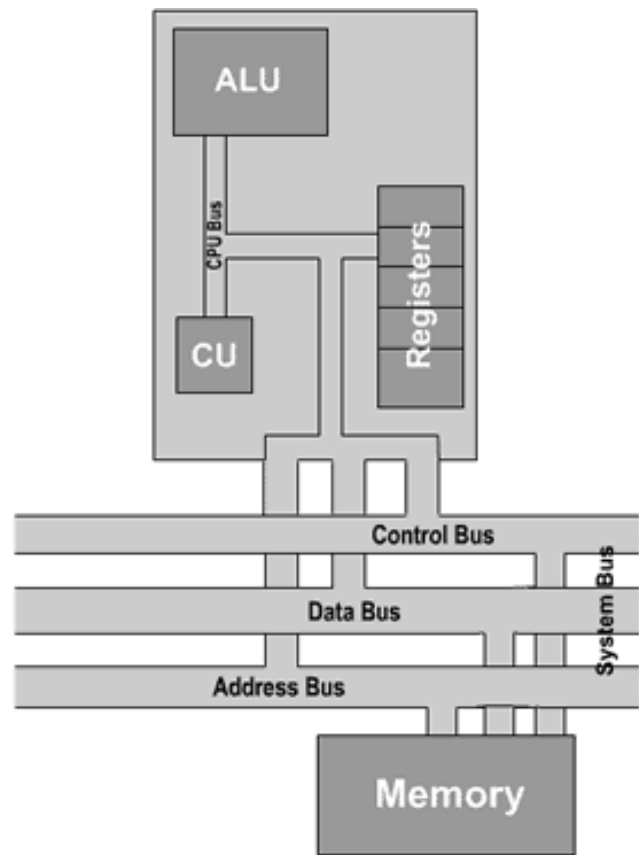


典型计算机系统的硬件架构 [1]

# 处理器

- ▶ 算术逻辑单元 (ALU)
  - ▶ 处理算术和逻辑操作
- ▶ 控制单元 (CU)
  - ▶ 定向指令进出处理器
  - ▶ 向 ALU 发送控制信号
- ▶ 寄存器
  - ▶ 指令寄存器
  - ▶ 程序计数器 (PC)
  - ▶ 通用寄存器

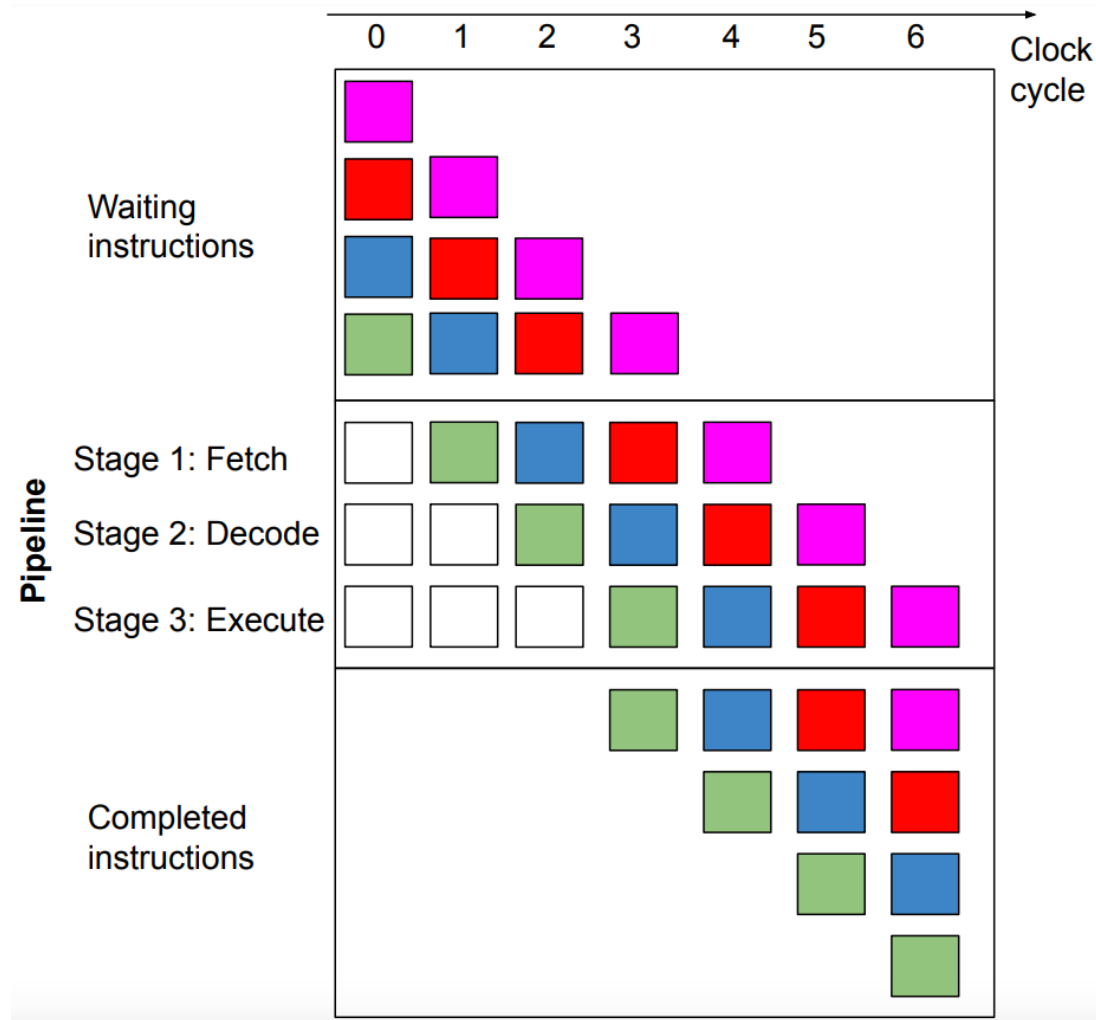
von Neumann architecture



Source: <http://computerscience.chemeketa.edu/cs160Reader/ComputerArchitecture/Processor.html>

# 指令周期与流水线

- ▶ 每一条指令都有下列周期
  - ▶ 获取指令
    - ▶ 从 PC 获取下一条指令的内存地址
  - ▶ 分析指令
    - ▶ 放置在指令寄存器中的指令被 CU 解释
  - ▶ 执行指令
    - ▶ ALU 执行算术或逻辑函数
- ▶ 流水线
  - ▶ 处于不同周期的不同指令可以同时执行



# 存储层次结构



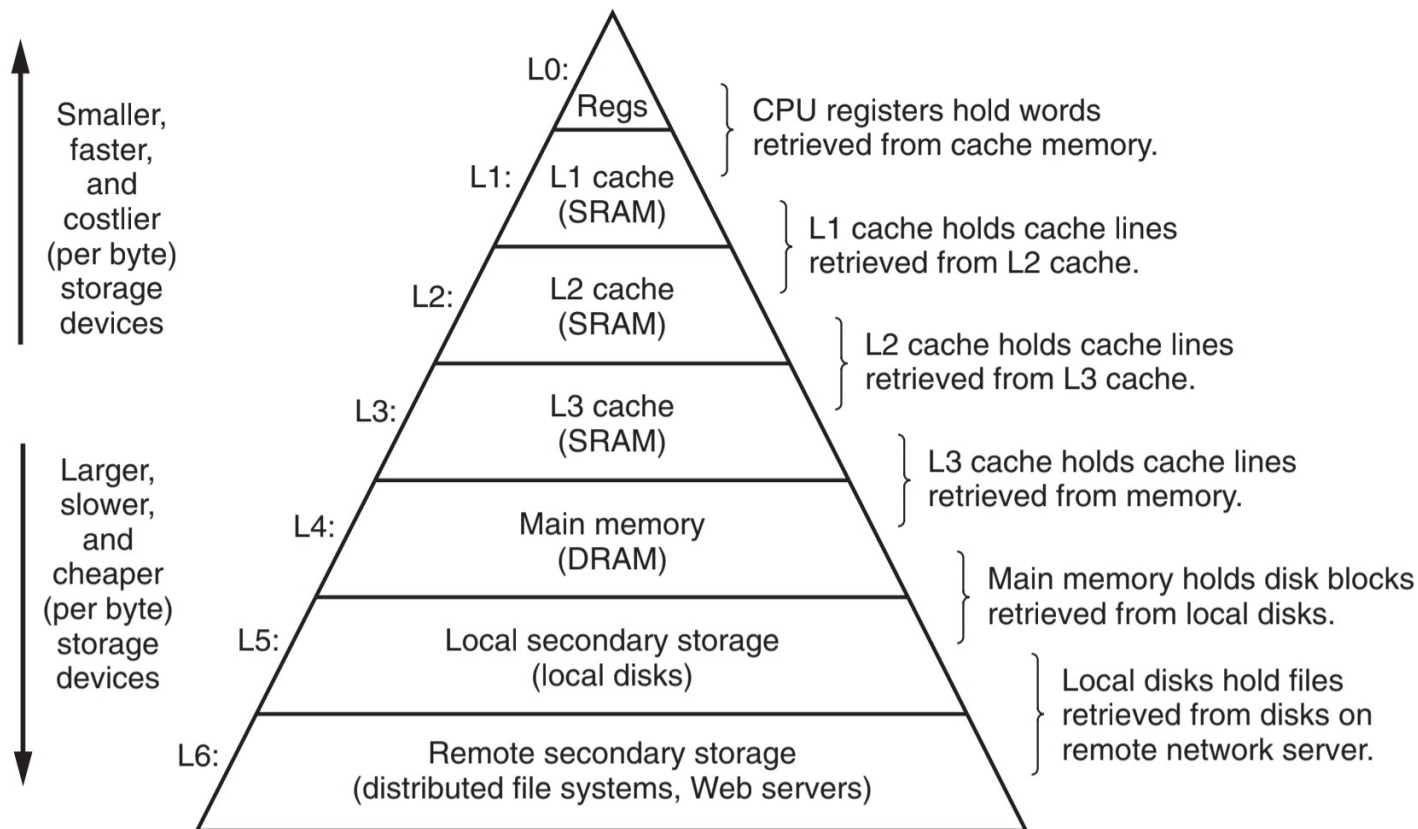
- ▶ 通过减少数据移动来保持处理器持续工作

- ▶ 高速存储非常昂贵

- ▶ L0 (Registers): 1ns, KB
- ▶ L1, L2, L3: 10ns, MB
- ▶ Main memory: 100ns, GB
- ▶ Disk: 10ms, TB
- ▶ Remote: 10sec, PB

- ▶ 动态随机存取存储器 (DRAM)

- ▶ Double Data Rate (DDR)
- ▶ High Bandwidth Memory (HBM)

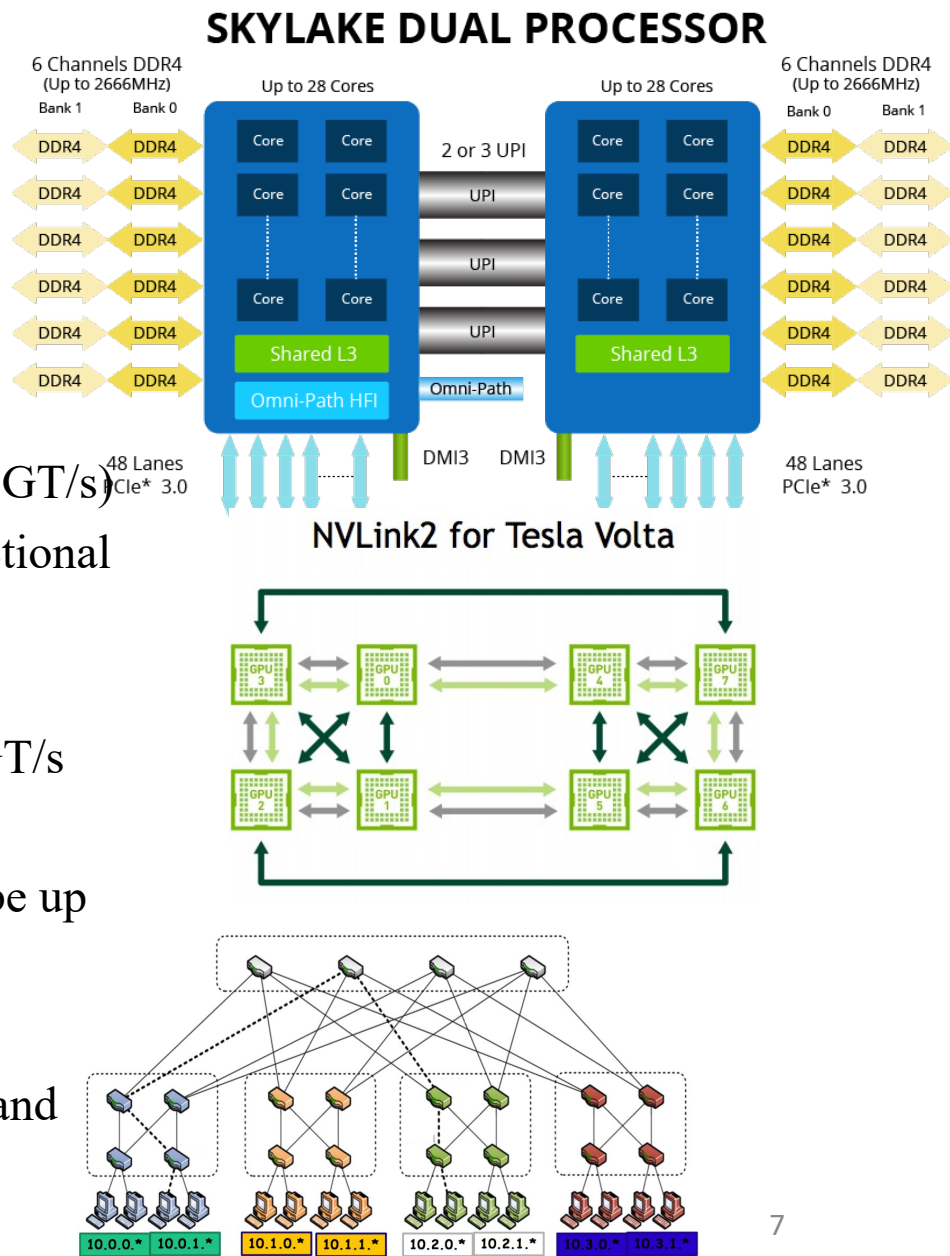


Memory hierarchy of modern computers [1]

# 所有计算机都在并行化

## 从手机到超级计算机

- 手机: 如 Apple A9, ARMv8-A dual-core
- 桌面电脑: 如 Intel Core i3, 2 Cores
- 服务器
  - 多核 CPU: 如 Intel(R) Xeon(R) Gold 6230 CPU: 20 Cores
  - 多 CPU: 如 Intel(R) Xeon(R) Gold 6230, 20
    - QPI: Intel QuickPath Interconnect (Unidirectional Speed: 6.4 GT/s)
    - UPI (starting at 2017): Intel Ultra Path Interconnect (Unidirectional Speed: 10.4 GT/s)
  - 多 GPU:
    - GPU 或其他附加设备使用 PCIe 总线: 如 PCIe3.0x16 (8 GT/s per lane, ~1GB/s per lane)
    - Nvidia GPU 也可使用 NVLink 总线: 如 NVLink 3.0 (can be up to 96 lanes) for Nvidia A100 GPUs (50 Gbit/s per lane)
- 集群
  - 多台服务器通过高速互联 (如 Mellanox 100 Gbit/s EDR InfiniBand ConnectX-5) 连接在一起



# 并行硬件的Flynn分类法, 1966



- ▶ Michael J. Flynn: 斯坦福大学名誉教授

<p>SISD</p> <p>Single instruction stream</p> <p>Single data stream</p> <p>No parallelism!</p>	<p>(SIMD)</p> <p>Single instruction stream</p> <p>Multiple data stream</p> <p>Popular: SSE, AVX, GPU</p>
<p>MISD</p> <p>Multiple instruction stream</p> <p>Single data stream</p> <p>Uncommon</p>	<p>(MIMD)</p> <p>Multiple instruction stream</p> <p>Multiple data stream</p> <p>Popular: multi-core, cluster</p>



- ▶ 指令与数据都是并行处理
- ▶ 单个控制单元从存储获取一条指令流 (IS)
- ▶ 控制单元产生相应的控制信号引导单个处理部件 (PE) 操作单个数据流 (DS)

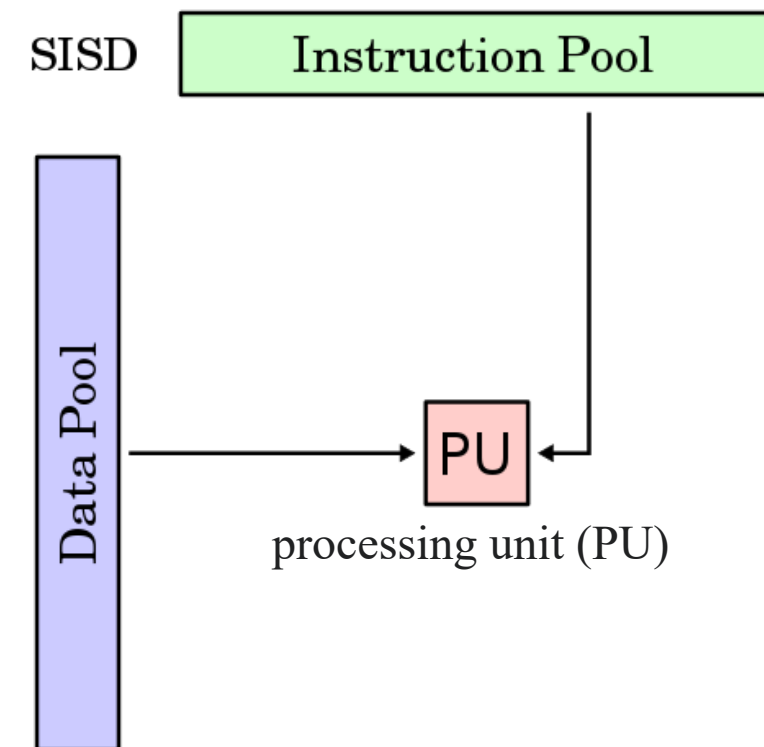


Image credit: [https://en.wikipedia.org/wiki/Flynn%27s\\_taxonomy](https://en.wikipedia.org/wiki/Flynn%27s_taxonomy)

# SIMD

- ▶ 通过数据并行将数据分配给处理器实现并行
- ▶ 对多组数据应用相同的指令
- ▶ SIMD 架构的典型例子
  - ▶ Intel x86 CPUs: MMX, SSE, and AVX
    - ▶ MMX: MultiMedia eXtension, 诞生于 1997
      - ▶ 一条指令可以被同时应用于 2 个 32-bit、4 个 16-bit 或 8 个 8-bit 整型数据
    - ▶ SSE: Streaming SIMD Extensions, 诞生于 1999
      - ▶ 一条 SSE 指令可以完成 4 个单精度 或 2 个双精度运算
    - ▶ AVX (Advanced Vector Extensions), AVX2, AVX-512, 2008
      - ▶ 一条 AVX-256 指令可以完成 8 个单精度 或 4 个双精度运算
      - ▶ 一条 AVX-512 指令可以完成 16 个单精度 或 8 个双精度运算
  - ▶ GPU: Graphics Processing Unit

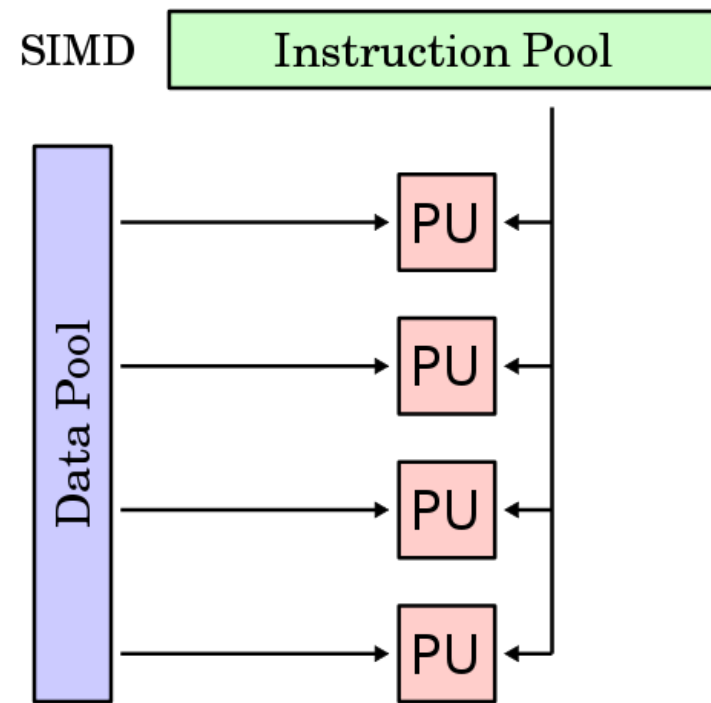


Image credit: [https://en.wikipedia.org/wiki/Flynn%27s\\_taxonomy](https://en.wikipedia.org/wiki/Flynn%27s_taxonomy)

# 对比示例: SISD vs. SIMD

## ▶ SISD

- ▶ 传统模式
- ▶ 一次操作产生一个结果

64-bit a

+

64-bit b

---

64-bit c

64-bit a3	64-bit a2	64-bit a1	64-bit a0
-----------	-----------	-----------	-----------

+

64-bit b3	64-bit b2	64-bit b1	64-bit b0
-----------	-----------	-----------	-----------

---

64-bit c3	64-bit c2	64-bit c1	64-bit c0
-----------	-----------	-----------	-----------

## ▶ SIMD

- ▶ 例如 Sandy Bridge (如 2 代酷睿): AVX (256 bits)  
或 Cascade Lake (如 i9-10980XE): AVX-512 (512 bits)
- ▶ 一次操作 (256-bit AVX) 产生  $256/64 = 4$  results (double-precision, 64bits)

# MIMD

- ▶ 支持在多个数据流上执行多个同步指令流
- ▶ 它是集成了多个完全独立的处理单元或核心，每一个都有自己的控制单元与 ALU
- ▶ 共享内存系统
  - ▶ 每一个处理器都可以利用所有内存
- ▶ 分布式内存系统
  - ▶ 通过互联网络连接

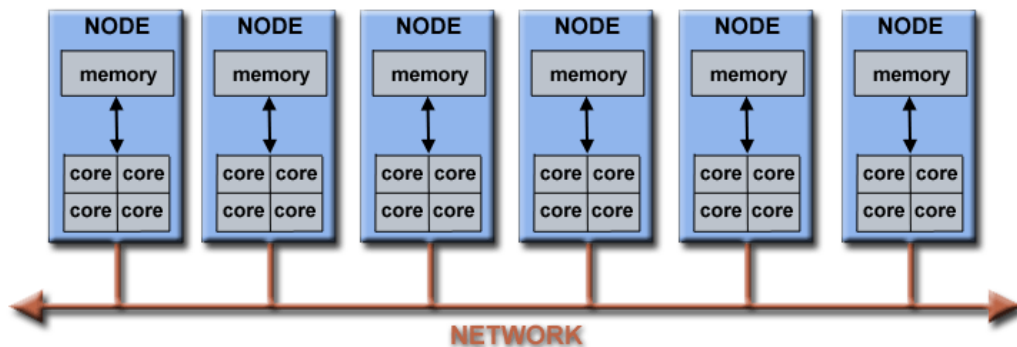


Image credit: <https://hpc.llnl.gov/training/tutorials/introduction-parallel-computing-tutorial>

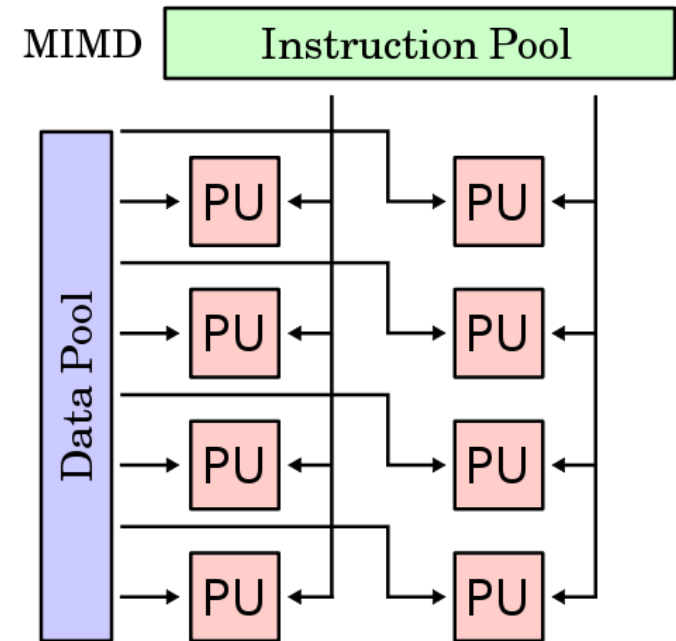


Image credit: [https://en.wikipedia.org/wiki/Flynn%27s\\_taxonomy](https://en.wikipedia.org/wiki/Flynn%27s_taxonomy)

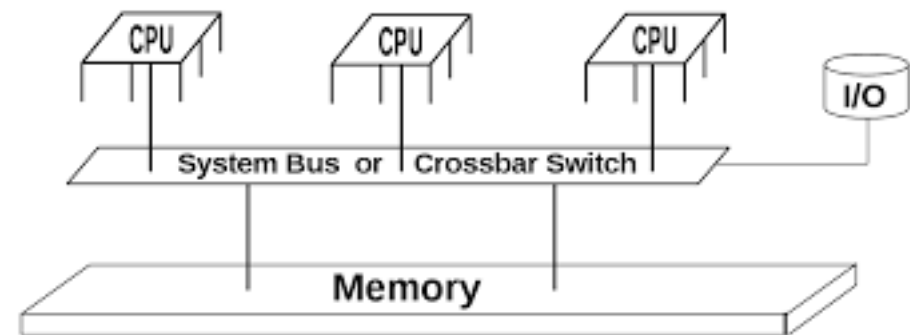
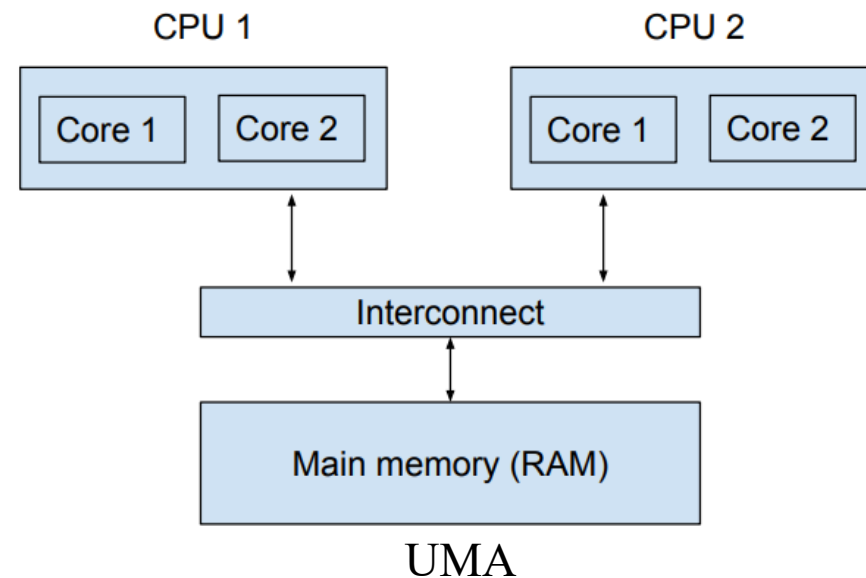


Image credit: [https://en.wikipedia.org/wiki/File:Shared\\_memory.svg](https://en.wikipedia.org/wiki/File:Shared_memory.svg)

# 共享内存系统

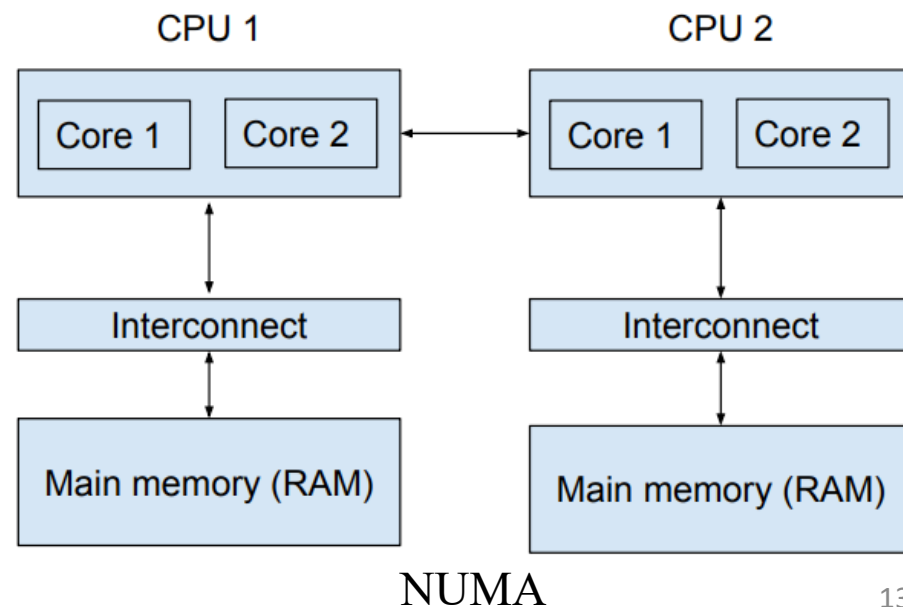
## ▶ 统一内存访问 (UMA)

- ▶ 所有处理器可直接访问主内存
- ▶ 每个处理器具有同等的内存访问延迟和访问速度
- ▶ 如Sun Starfire servers、Compaq alpha server 以及 HP-v series
- ▶ 低扩展性
  - ▶ 随着 CPU 核心数增加，硬件上为它们提供共享内存变得困难



## ▶ 非一致性内存访问 (NUMA)

- ▶ 每个处理器可通过特定硬件 (如 QPI, UPI) 访问彼此的部分主内存
- ▶ 访存的延迟取决于处理器的放置位置
- ▶ 如 BBN, TC-2000, SGI Origin 3000, Cray



# 分布式内存系统



- ▶ 一系列的处理节点通过网络进行互联

- ▶ 网络的构成部件

- ▶ 接线
    - ▶ 交换机
    - ▶ 网卡 (NIC)

- ▶ 网络通信速度影响因素

- ▶ 连接速度
    - ▶ 路由
    - ▶ 网络拓扑

- ▶ Fat tree, Bcube, Torus, ...



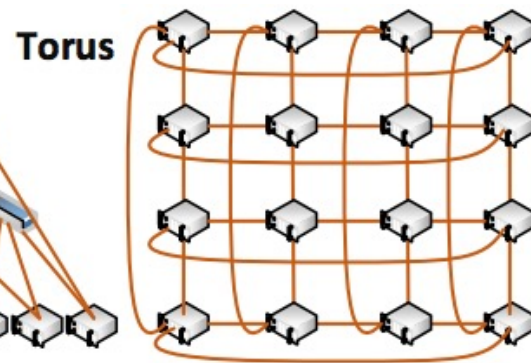
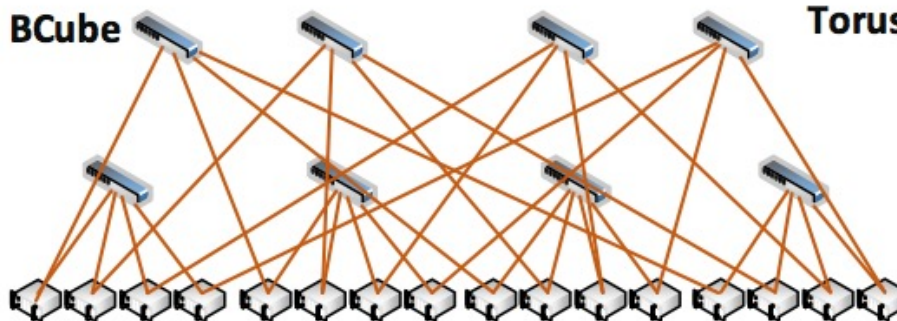
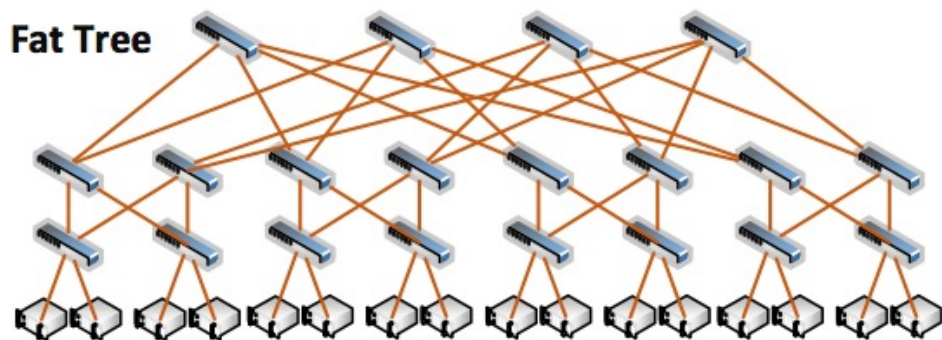
10G DAC Copper Cable



NIC Card



Image credit [1]



# 向量及SIMD概要

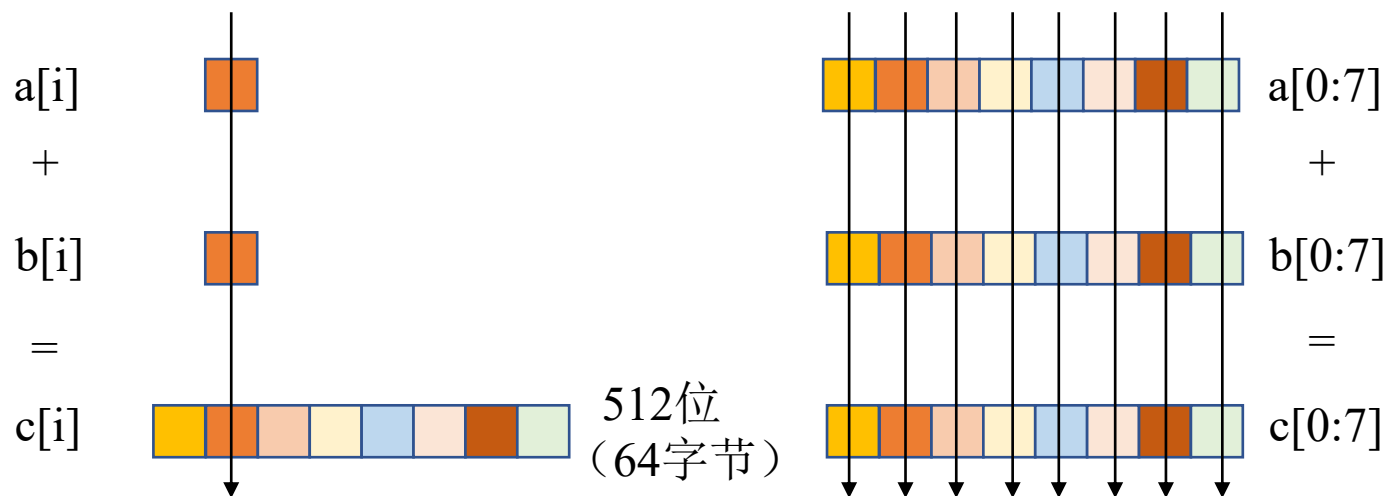
- ▶ 通过处理器特殊的向量单元，可以同时加载和操作多个数据元素

- ▶ 向量化术语

- ▶ 向量通道
- ▶ 向量宽度
- ▶ 向量长度
- ▶ 向量指令集

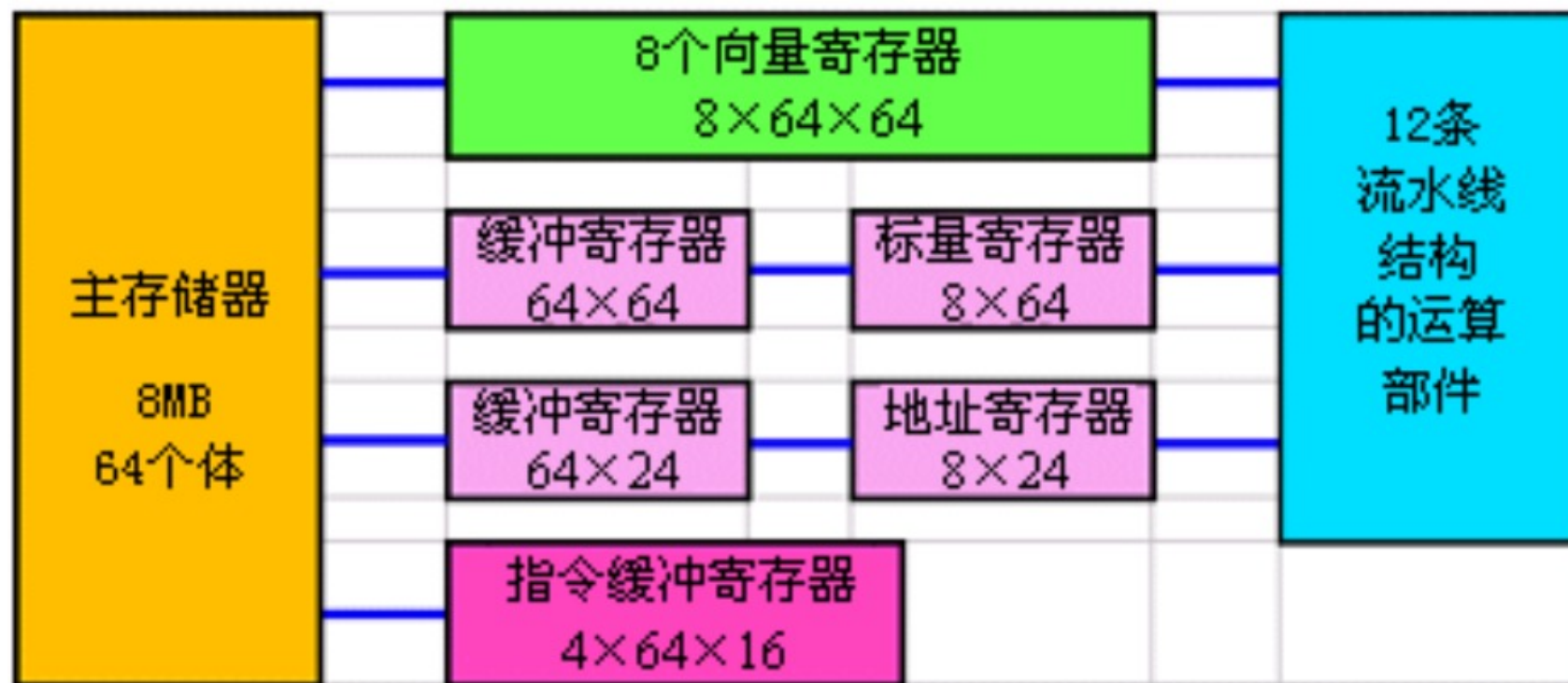
- ▶ 向量化的实现方式

- ▶ 软件：向量指令由编译器生成
- ▶ 硬件：指令与处理器的向量单元进行匹配





# 最早的向量机Cray-1（1976）

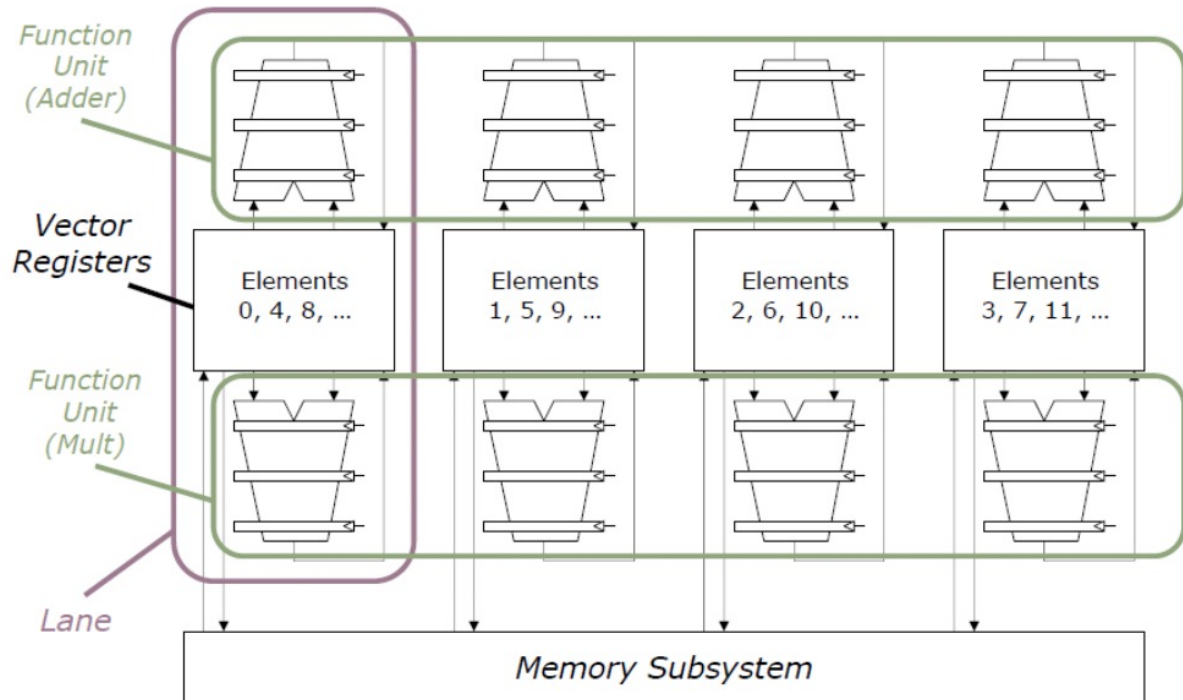


- ✓ “存储器-存储器”结构优化为“寄存器-寄存器”结构
- ✓ 运算部件所需要的操作数从向量寄存器中读取，运算的中间结果也写到向量寄存器中
- ✓ 操作数，数值都存储在寄存器中读写

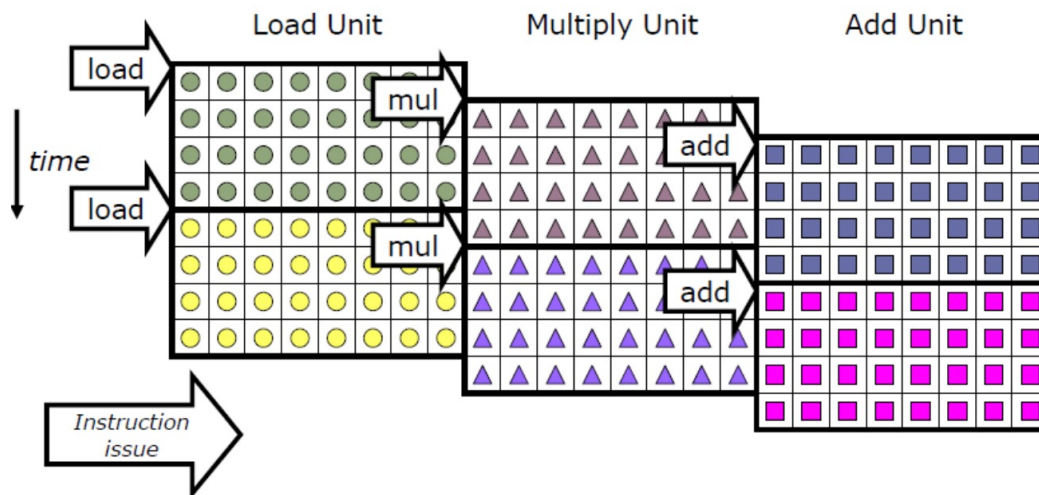


# 向量运算执行的特点

- ✓ 使用深度流水执行元素操作
- ✓ 简化深度流水的控制（因为一个向量内元素是独立的）



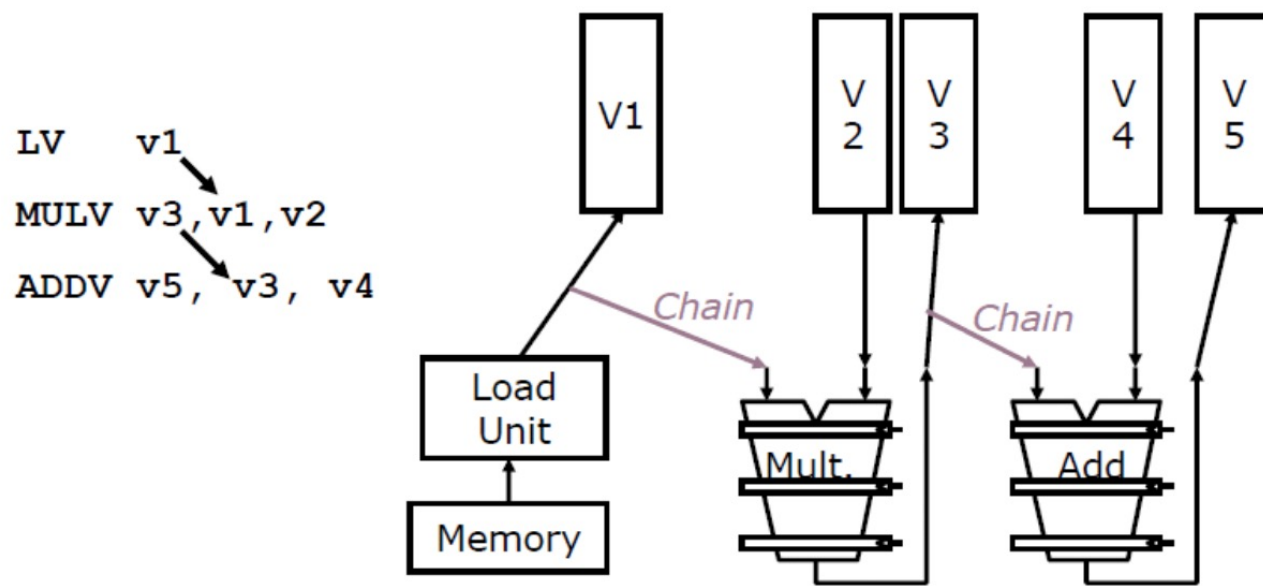
- ✓ 多个向量指令可以重叠执行



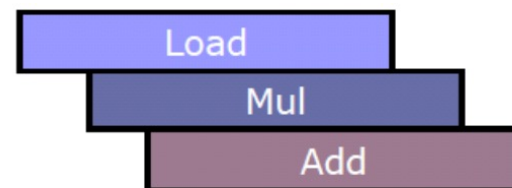
# 向量运算执行的特点

## 链化执行

- ✓ 如果没有链，在开始下一个以来的指令是，必须等待结果中的最后一个元素被写入。
- ✓ 有了链化执行，只要写过中第一个元素被写入就可以开始下一个依赖操作。



无链化执行

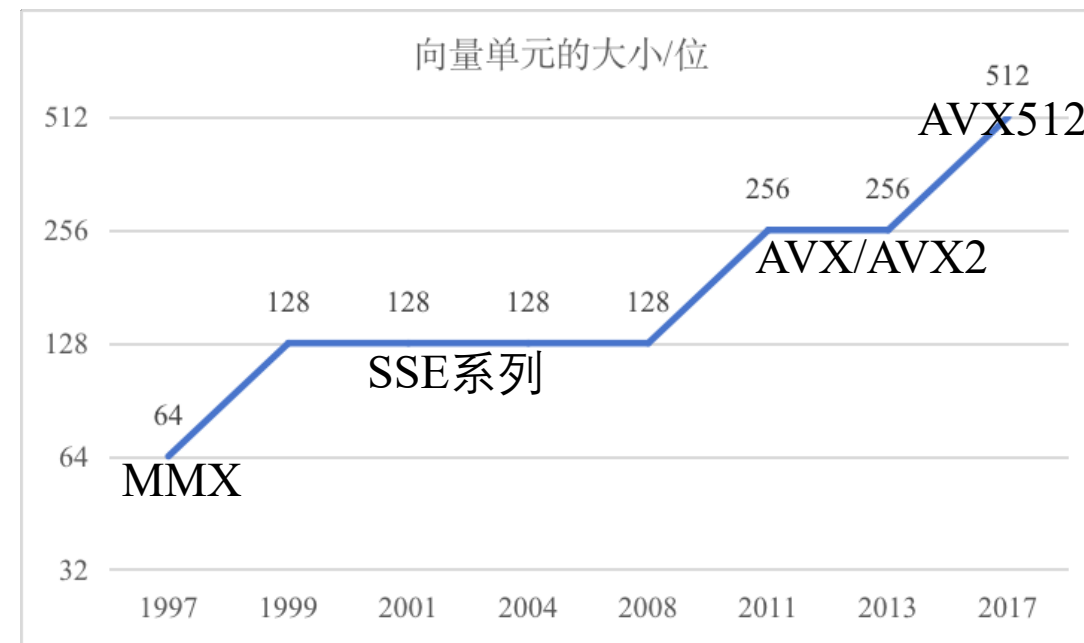


有链化执行

# 向量化的硬件趋势

过去10年，向量硬件的发展极大地完善了向量功能

发行版	功能
SSE（单指令多数数据流扩充）	Intel首个提供带有单精度支持的浮点运算的向量单元
SSE2	增加对双精度的支持
AVX（高级向量扩展）	提供两倍向量长度，AMD在其具有竞争力的硬件中添加了融合乘加（FMA）运算向量指令，有效地使某些环路的性能翻倍
AVX2	Intel为其向量处理器添加了乘积累加运算FMA
AVX512	首次出现在Knights Landing处理器上。并在2017年进入多核处理器主线硬件产品阵容。作为向量硬件架构的持续改进，从2018年起，Intel和AMD已经实现了AVX512的多个版本





# 向量化方法

## ➤ 优化软件库

- Intel处理器: MKL提供了BLAS、LAPACK、SCALAPACK、FFT和Sparse Solvers
- GPU处理器: CUBLAS、CUFFT和CUTLASS

## ➤ 自动向量化

- `__restrict__` 关键字
- 编译器标志: `-ftree-vectorize -fopt-info-vec-optimized`

## ➤ 在编译器中使用hints

- `#pragma omp simd`
- 同样的需要加入编译器标志

## ➤ 向量内在函数

- kahan sum向量库 (Intel和AMD x86的向量库)

## ➤ 汇编指令

- `vmovapd`, `vaddsd`, `vsubsd`, `vxorpd` ...

- ▶ Thomas Sterling, Matthew Anderson and Maciej Brodowicz (2018), “High Performance Computing: Modern Systems and Practices,” Morgan Kaufmann, **Chapter 2**. [PDF: <https://www.sciencedirect.com/book/9780124201583/high-performance-computing>]
- ▶ Duncan, R. (1990). “A survey of parallel computer architectures”. *Computer*, 23(2), 5-16. [PDF: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=44900>]



# 总结

- ▶ 计算机体系结构
  - ▶ CPU
  - ▶ 内存
  - ▶ 指令周期
  - ▶ 指令流水线
- ▶ 并行计算机体系结构
  - ▶ Flynn 分类
    - ▶ SISD、SIMD、MISD 以及 MIMD
  - ▶ 共享内存与分布式内存系统
  - ▶ 网络拓扑
- ▶ 向量化方法及SIMD
  - ▶ 常见硬件结构
  - ▶ 常用向量化方法