

计算机组成和体系结构

第四讲

四川大学网络空间安全学院

2020年3月16日

封面来自alphacoder.com，作者Samim Hasan

版权声明

课件中所使用的图片、视频等资源版权归原作者所有。

课件原创内容采用 [创作共用署名—非商业使用—相同方式共享4.0国际版许可证\(Creative Commons BY-NC-SA 4.0 International License\)](#) 授权使用。

Copyright@四川大学网络空间安全学院计算机组成与体系结构课程组，2020



上期内容回顾

- 计算机模型2
 - 内存概述
 - 体系结构
 - 交叉存储器
 - 中断及中断处理
- 计算机模型3: MARIE机器
 - 体系结构
 - 指令集
 - 寄存器传输表示
 - 汇编语言
 - 控制单元的实现
 - 现实世界中的指令集

本期学习目标

本期学习目标

- 回顾1-3讲的内容

本期学习目标

- 回顾1-3讲的内容
- 梳理各部分内容之间的联系

本期学习目标

- 回顾1-3讲的内容
- 梳理各部分内容之间的联系
- 作业题目分析

中英文缩写对照表

英文缩写	英文全称	中文全称
ALU	Arithmetic-Logic Unit	算数逻辑单元
CPU	Central Processing Unit	中央处理器
ISA	Instruction Set Architecture	指令集架构
RAM	Random Access Memory	随机访问内存
RTL	Register Transfer Language	寄存器传输语言
RTN	Register Transfer Notation	寄存器传输表示

1-3周课程小结

*What we talk about when we talk about computer
organization and architecture.*

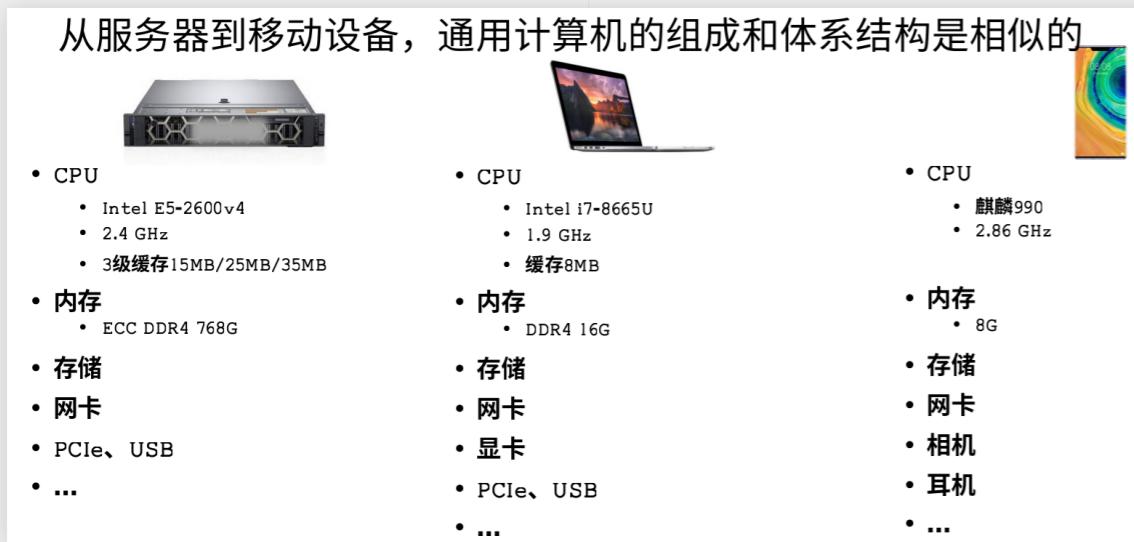
前三周学习的内容在计算机组成和体系结构这门课程中的作用

*What we talk about when we talk about computer
organization and architecture.*

前三周学习的内容在计算机组成和体系结构这门课程中的作用
以及它们之间的联系

初识计算机系统

- 认识计算机系统的主要组件：三个主要构成部分
- 计算机系统各组件的性能认识：各组成部分的主要性能指标
 - 计算机系统中数字表达的工具：利用前缀乘数简化性能指标的表示



表示容量、速度等的前缀乘数

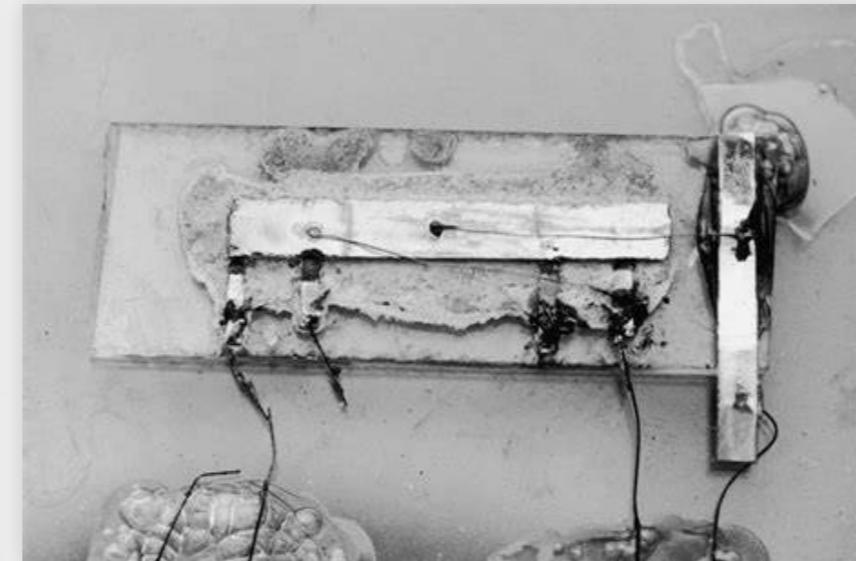
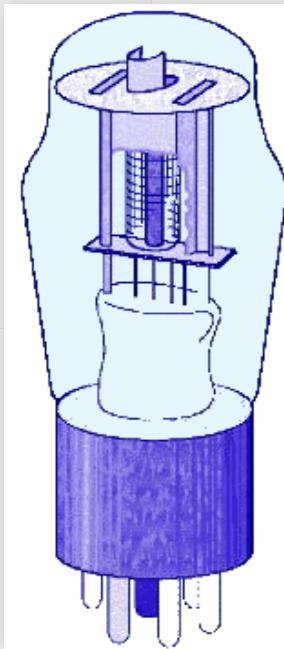
- Kilo- (K) = 1 thousand = 10^3 and 2^{10}
- Mega- (M) = 1 million = 10^6 and 2^{20}
- Giga- (G) = 1 billion = 10^9 and 2^{30}
- Tera- (T) = 1 trillion = 10^{12} and 2^{40}
- Peta- (P) = 1 quadrillion = 10^{15} and 2^{50}
- Exa- (E) = 1 quintillion = 10^{18} and 2^{60}
- Zetta- (Z) = 1 sextillion = 10^{21} and 2^{70}
- Yotta- (Y) = 1 septillion = 10^{24} and 2^{80}

表示容量、速度等的前缀乘数

- Milli- (m) = 1 thousandth = 10^{-3}
- Micro- (μ) = 1 millionth = 10^{-6}
- Nano- (n) = 1 billionth = 10^{-9}
- Pico- (p) = 1 trillionth = 10^{-12}
- Femto- (f) = 1 quadrillionth = 10^{-15}
- Atto- (a) = 1 quintillionth = 10^{-18}
- Zepto- (z) = 1 sextillionth = 10^{-21}
- Yocto- (y) = 1 septillionth = 10^{-24}

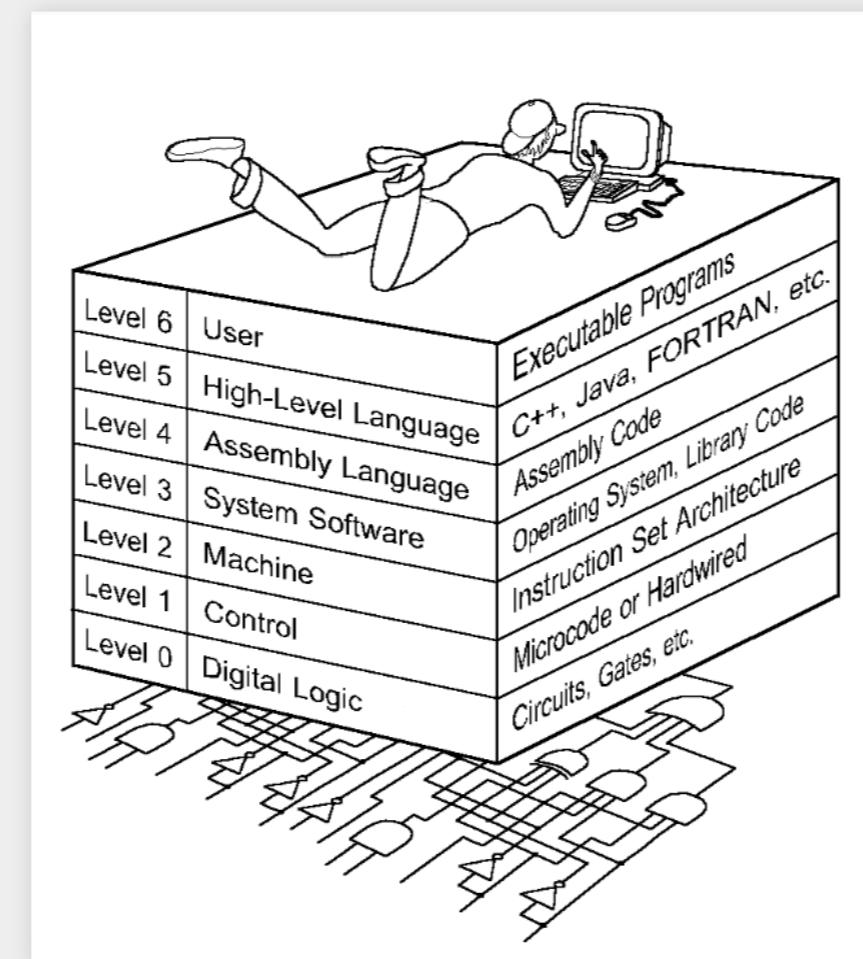
计算机发展历史

- 了解CPU实现的发展过程
 - 本质不变：数字逻辑电路
 - 按照材料发展：电子管到晶体管
 - 按照集成度发展：晶体管到集成电路到超大规模集成电路
- 发展规律：摩尔定律和罗克定律



计算机层次化结构

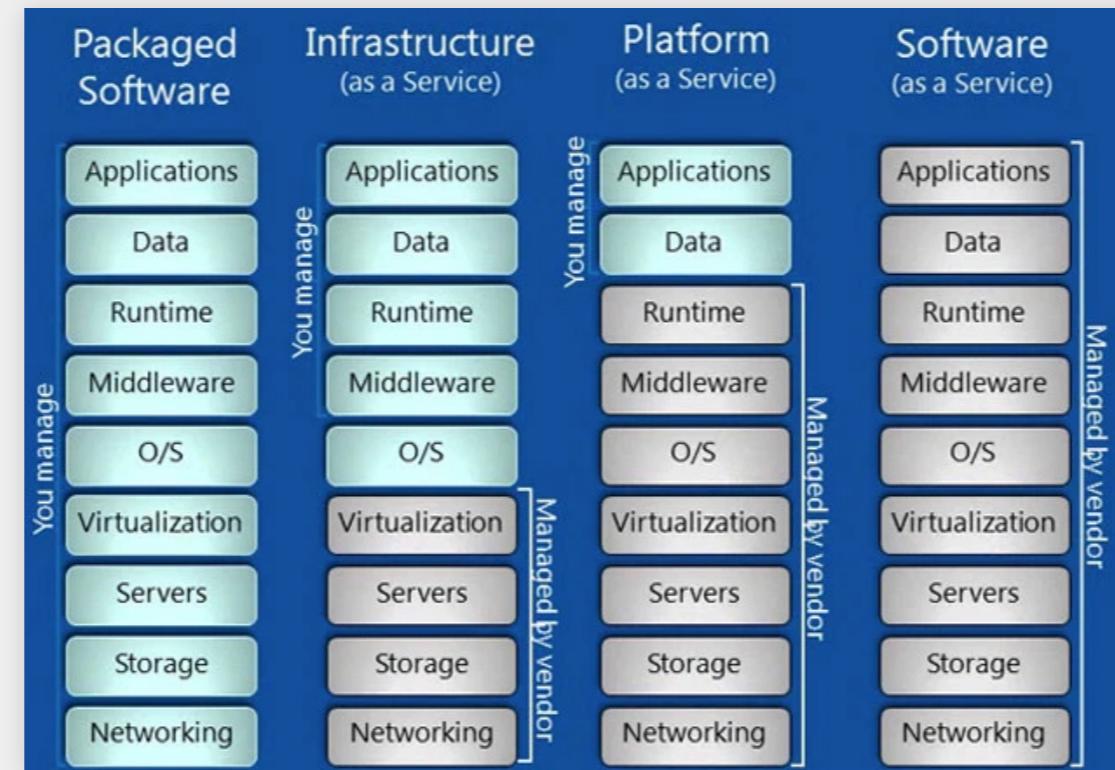
- 计算机系统整体的逻辑抽象
 - 展现了本课程内容在计算机系统中的位置
- 分解系统复杂性的方法：抽象



图片来源：Linda Null and Julia Lobur, *Computer Organization and Architecture*, 4th Edition

云计算

- 技术发展对于计算机组成的影响
 - 网络技术、虚拟化技术、管理平台
- 层次化结构的一个应用

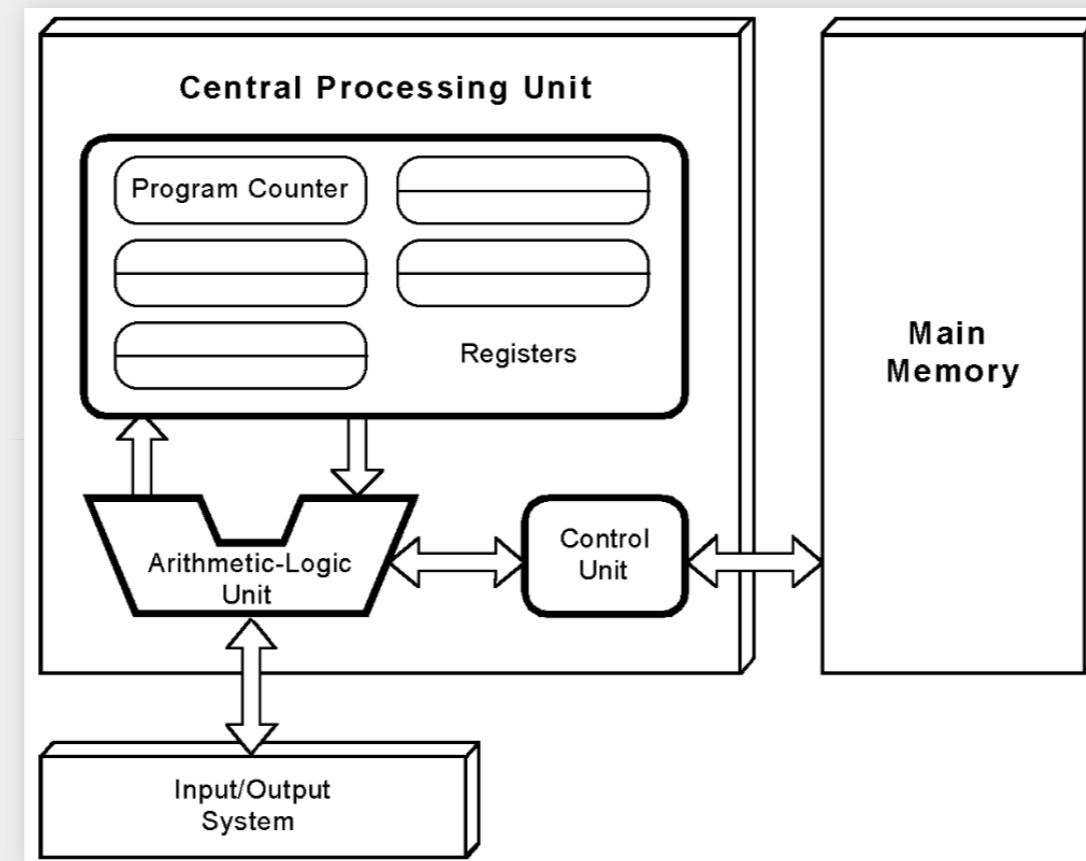


图片来源: <https://venturebeat.com/2011/11/14/cloud-iaas-paas-saas/>

*实际上最左边这一列也有对应的云计算服务，称为MaaS(Metal-as-a-service)。

冯—诺依曼模型

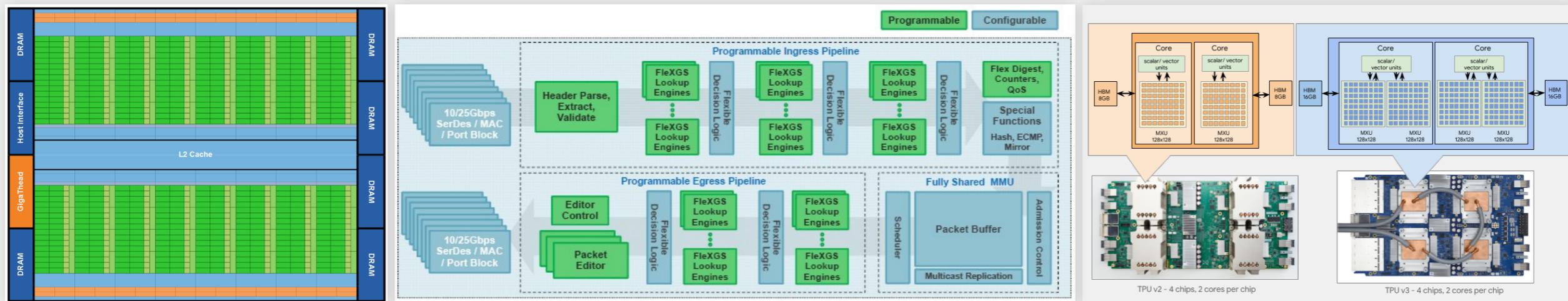
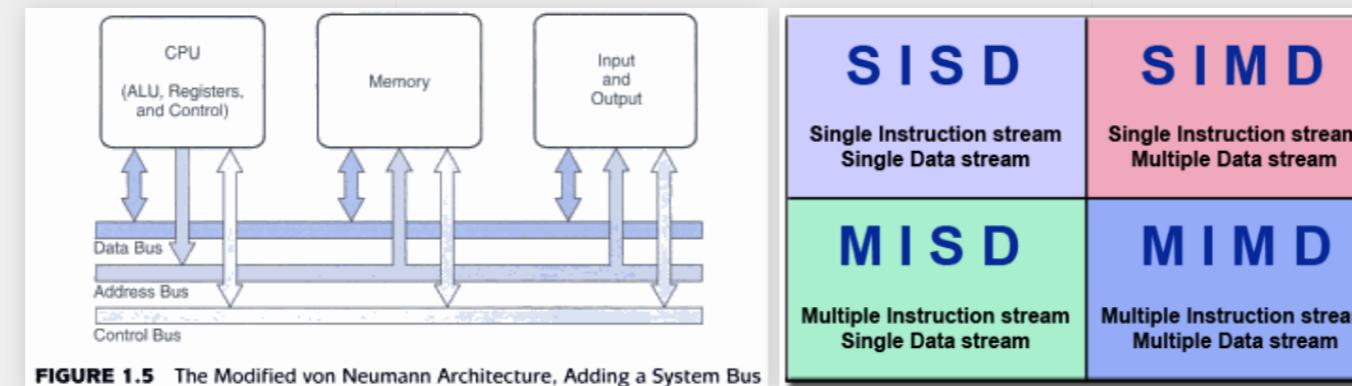
- 三种主要组成部分的连接方式是本课程中的计算机组成的基础，也是现实中大部份计算机组成的基础
- 存储程序和取-译码-执行周期是现代计算机依然延用的设计



图片来源： Linda Null and Julia Lobur, *Computer Organization and Architecture*, 4th Edition

非冯-诺依曼模型

- 对冯诺依曼模型的改进
- 功能主导体系结构设计的实例

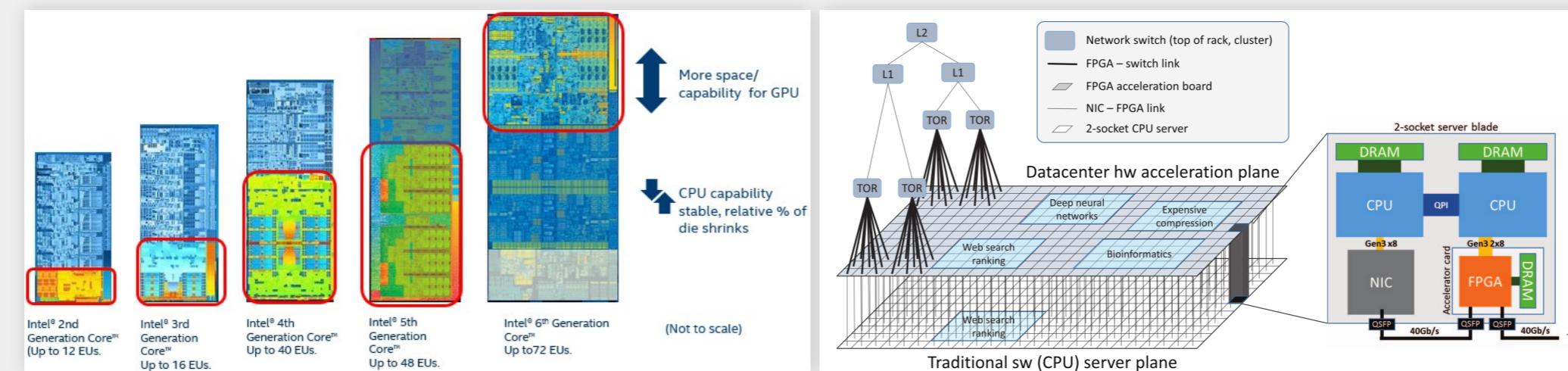


图片来源：Linda Null and Julia Lobur, *Computer Organization and Architecture*, 4th Edition, Nvidia, NVIDIA's Next Generation CUDA Compute Architecture: Fermi,

<https://www.nextplatform.com/2017/08/04/making-mainstream-ethernet-switches-malleable/>, <https://cloud.google.com/tpu/docs/system-architecture>

并行计算

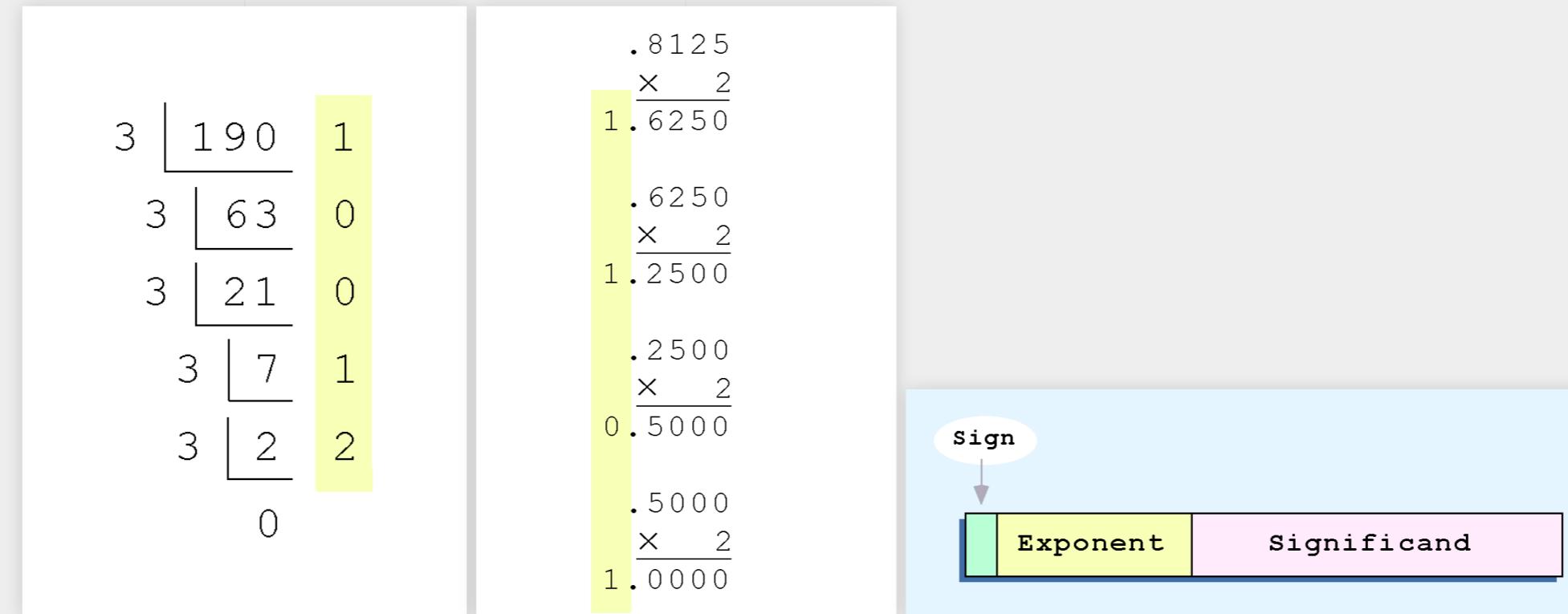
- 非冯诺依曼模型的实例
- “水平扩展”的设计思想



图片来源：<http://www.tech-faq.com/wp-content/uploads/images/Quad-Core-Processor.jpg>, <https://servermarketinglibrary.intel.com/wp-content/uploads/assets/s2600wt2-angle.png>,
http://pacman.cs.tsinghua.edu.cn/~cwg/papers_cwg/a56-lin.pdf, https://software.intel.com/sites/default/files/managed/f7/f8/cpugpu_0.jpg, <https://www.usenix.org/conference/nsdi18/presentation/firestone>

数据表示

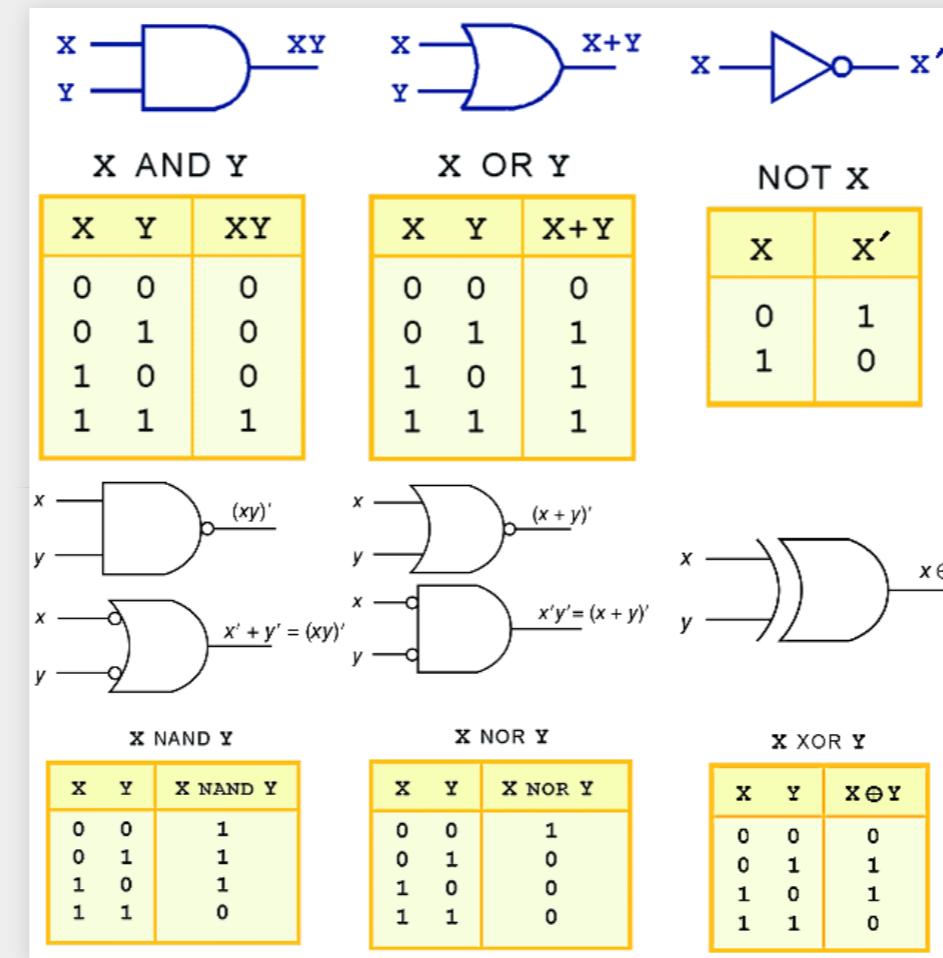
- 计算机内部如何**高效**存储和表示不同类型的数据
 - 无符号整数、有符号整数、浮点数
- 工具：进制转化、补码、移码、科学记数法、算术运算方法*



图片来源：Linda Null and Julia Lobur, *Computer Organization and Architecture*, 4th Edition

布尔代数

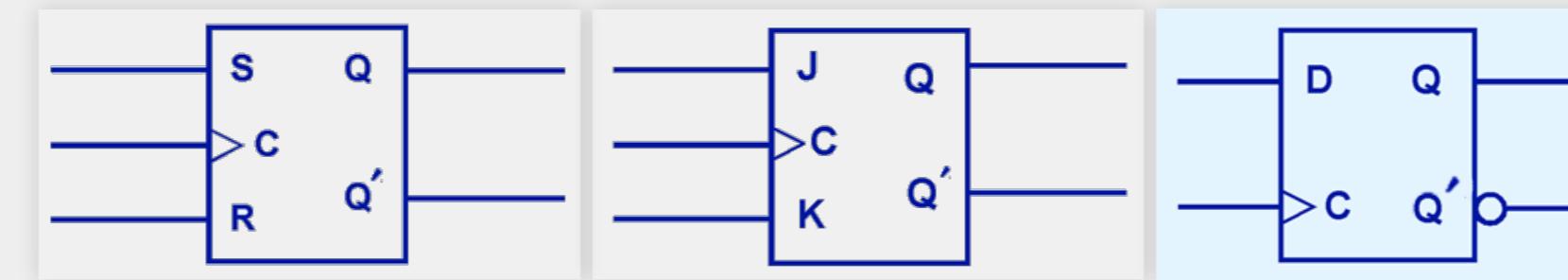
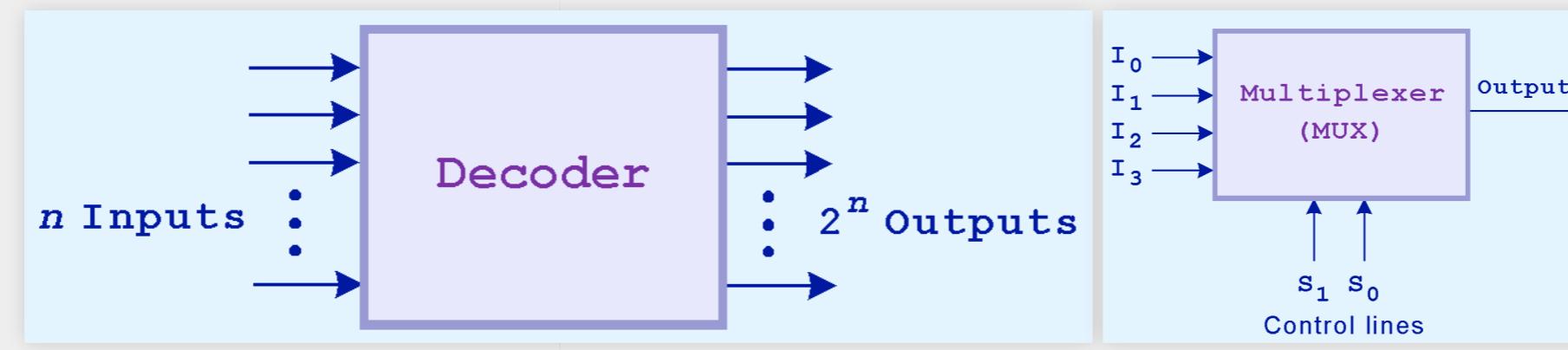
- 逻辑操作(布尔代数)的运算法则
- 逻辑操作和数字逻辑电路实现的对应关系



图片来源：Linda Null and Julia Lobur, *Computer Organization and Architecture*, 4th Edition

常见的数字逻辑电路和时序电路

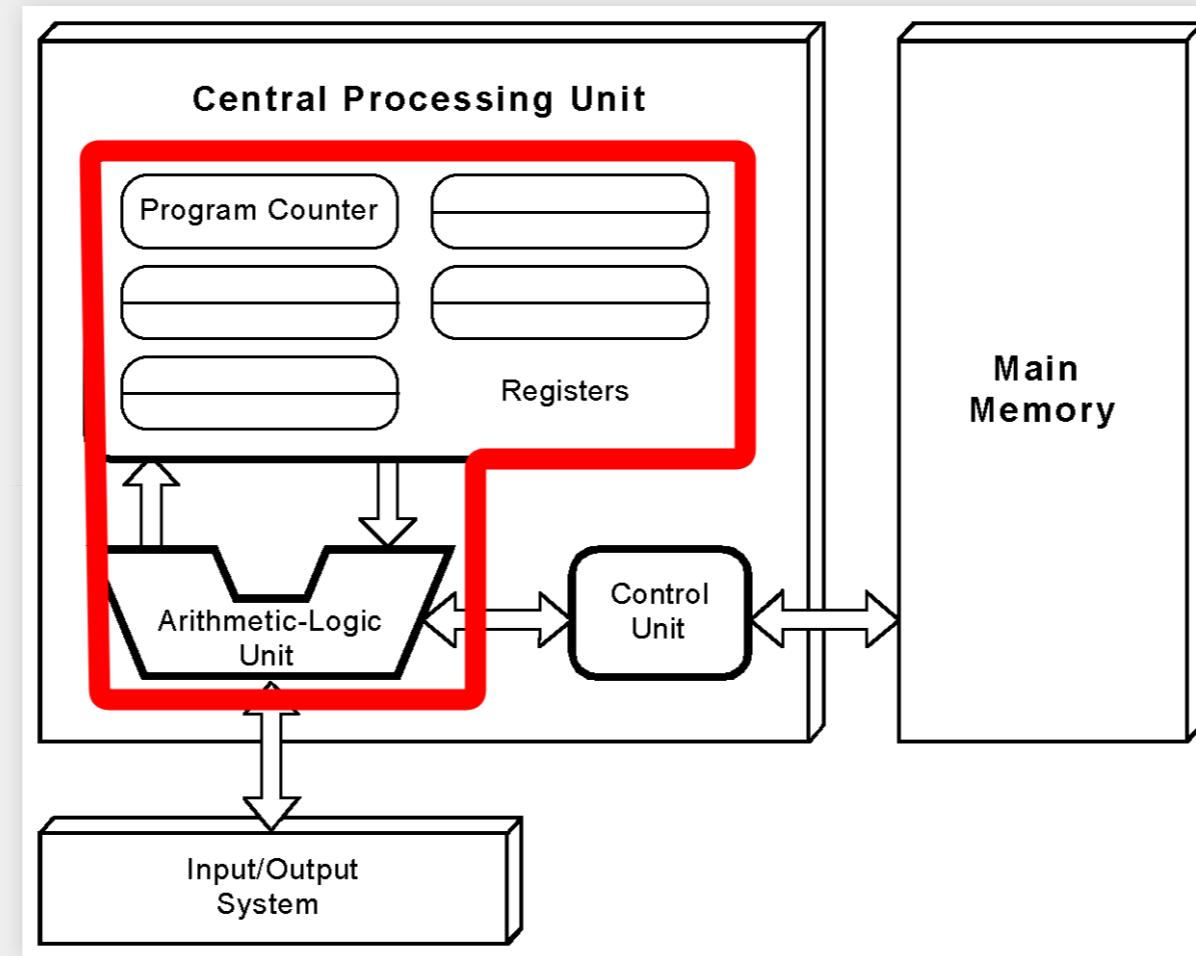
- CPU电路实现的基础元件
- 体现了抽象的思想：关注实现的功能而不是内部的电路实现



图片来源：Linda Null and Julia Lobur, *Computer Organization and Architecture*, 4th Edition

CPU组织结构

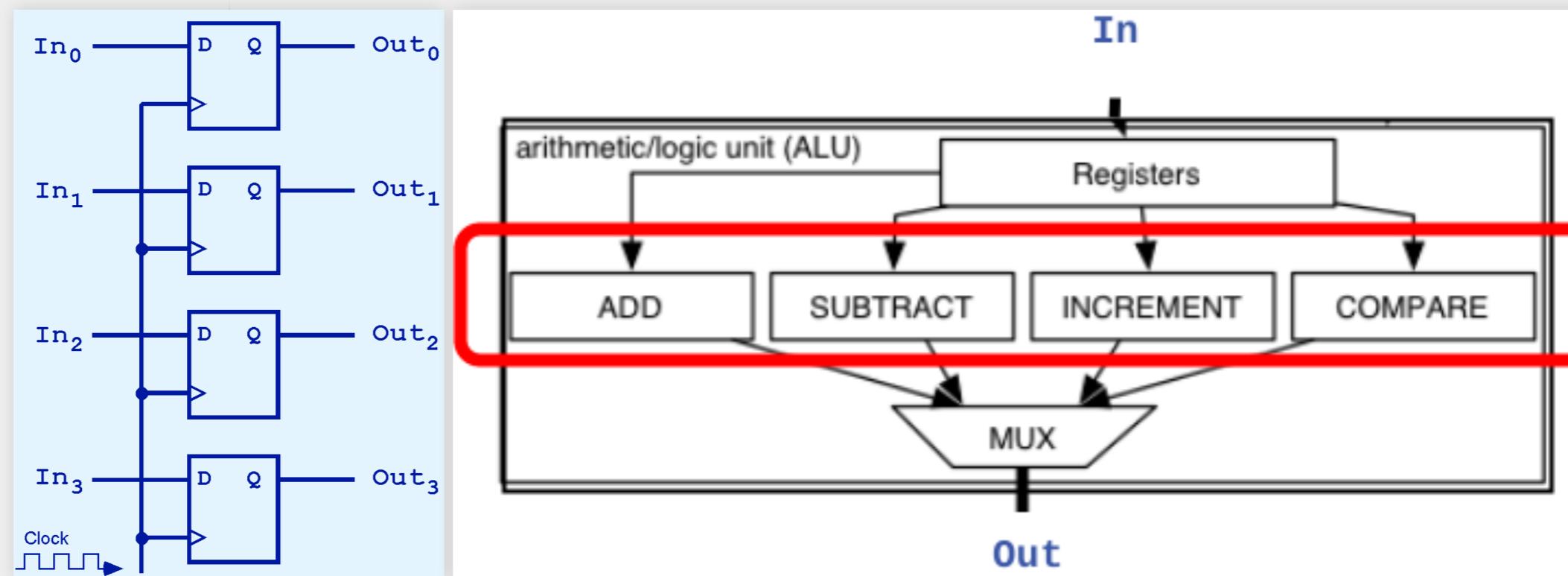
- 计算机组成细化到CPU的组成
- 这个基本组成是现代CPU内部组成的基础



图片来源：Linda Null and Julia Lobur, *Computer Organization and Architecture*, 4th Edition

寄存器、算术逻辑单元、控制单元

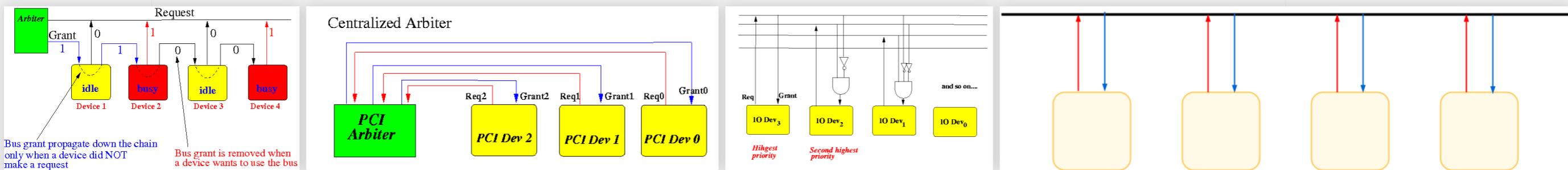
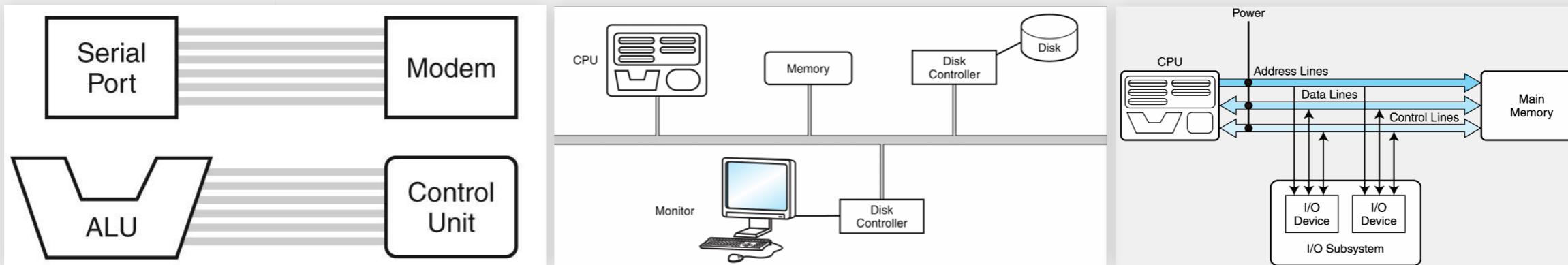
- CPU的组成进一步细化
 - 寄存器的**功能**和**实现**
 - 算术逻辑单元**功能**和**实现**
 - 控制单元的**功能**
- 通用的CPU组成部分



图片来源：Linda Null and Julia Lobur, *Computer Organization and Architecture*, 4th Edition

总线

- CPU的组成进一步细化为总线的功能和实现
- 通用的CPU组成部分



图片来源：Linda Null and Julia Lobur, *Computer Organization and Architecture*, 4th Edition, <http://www.mathcs.emory.edu/~cheung/Courses/355/Syllabus/5-bus/bus-arbiter.html>,
<http://www.mathcs.emory.edu/~cheung/Courses/355/Syllabus/5-bus/bus-arbiter.html>, <http://www.mathcs.emory.edu/~cheung/Courses/355/Syllabus/5-bus/bus-arbiter.html>

时钟和I/O子系统

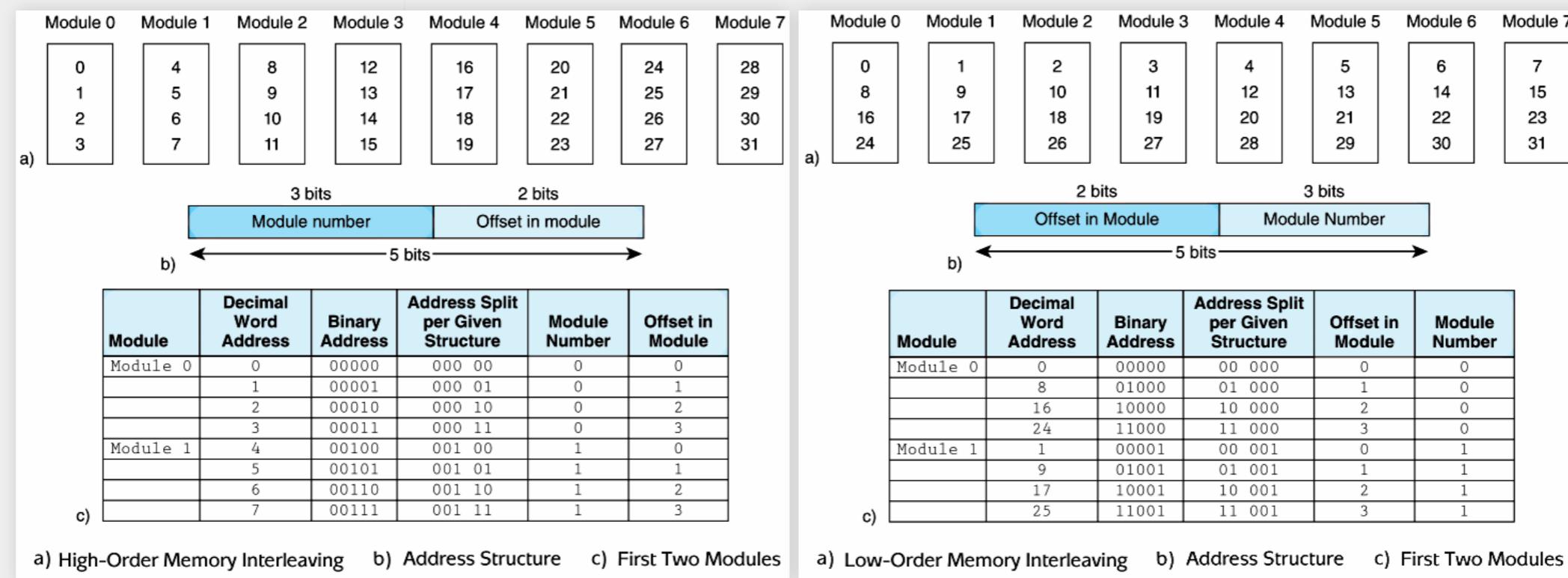
- CPU的组成进一步细化
 - 时钟的功能
 - I/O的功能和常见的实现方式
- 通用的CPU组成部分
- 工具：CPU时间计算公式(分析推导影响程序运行速度因素)

$$\text{CPU Time} = \frac{\text{seconds}}{\text{program}} = \frac{\text{instructions}}{\text{program}} \times \frac{\text{avg. cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

图片来源： Linda Null and Julia Lobur, *Computer Organization and Architecture*, 4th Edition

内存

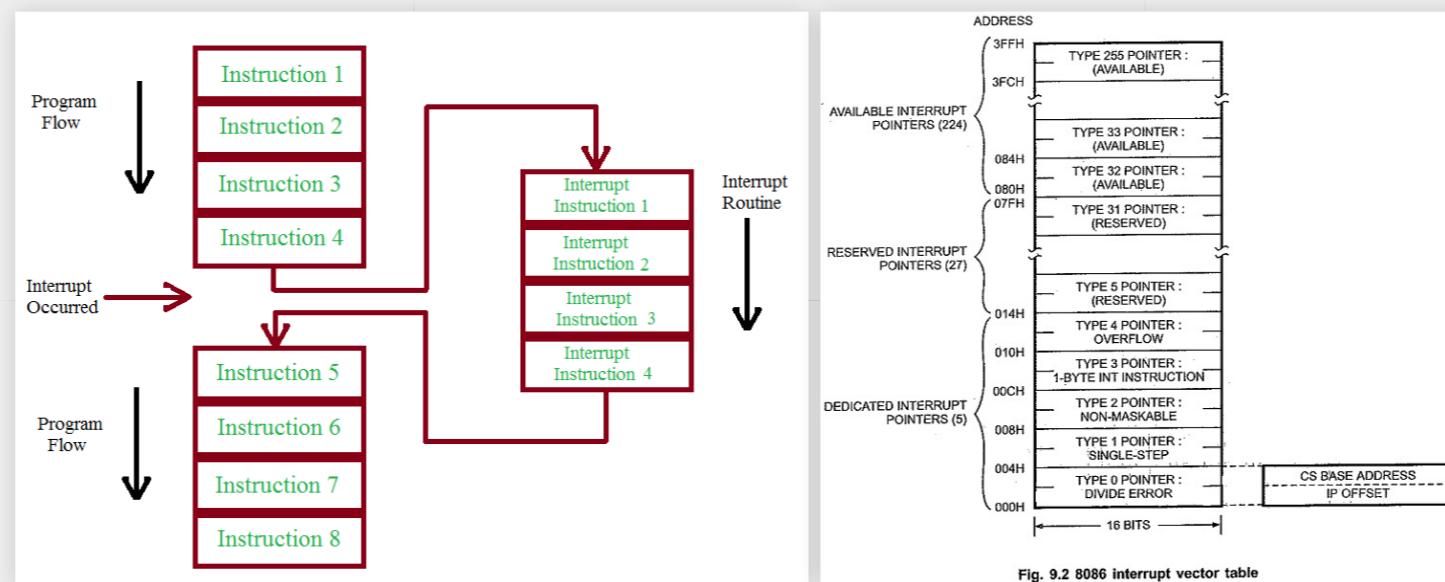
- 计算机组成为进一步细化为内存的功能和实现*
- 计算机体系结构中内存的功能
- 工具：不同编址方法和交叉编址方法
 - 连接方式决定寻址方法，寻址方法决定编址方法



图片来源：Linda Null and Julia Lobur, *Computer Organization and Architecture*, 4th Edition

中断

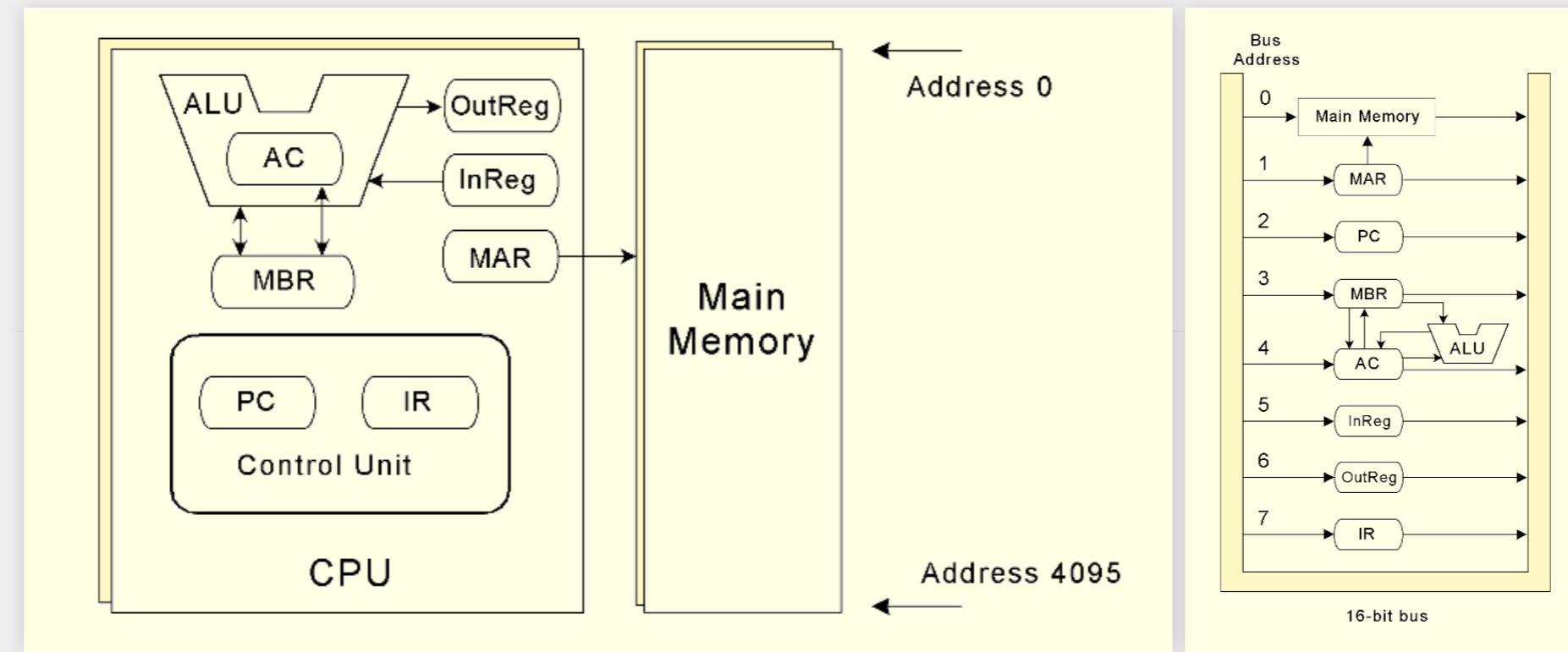
- 中断是实际计算机系统中CPU与I/O通信的实现机制
- 也是现代CPU设计中必须考虑的功能
- 通过指令集影响到操作系统的实现



图片来源：<https://www.electronicshub.org/arm-interrupt-tutorial/> 图片来源：<https://www.eeguide.com/8086-interrupt/>

MARIE

- 给出了上述各计算机组成部分的一个实例
- 引出更高层次的抽象：指令集和汇编语言
- 从MARIE的体系结构设计中，我们可以一窥CPU的设计过程



图片来源：Linda Null and Julia Lobur, *Computer Organization and Architecture*, 4th Edition

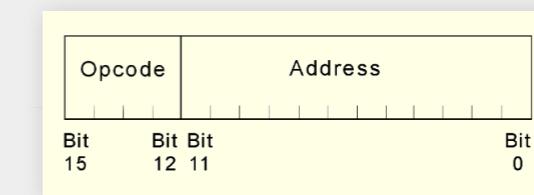
MARIE指令集

- 帮助了解计算机系统指令集的特点以及实现方式
- 工具：指令编码、寄存器传输表示
 - 指令集决定了指令编码，指令编码决定了可寻址内存大小

指令	RTN
JNS x	MBR←PC MAR←x M[MAR]←MBR MBR←x AC←1 AC←AC+MBR PC←AC
LOAD x	MAR←x MBR←M[MAR] AC←MBR
STORE x	MAR←x MBR←AC M[MAR]←MBR
ADD x	MAR←x MBR←M[MAR] AC←AC + MBR
SUBT x	MAR←x MBR←M[MAR] AC←AC - MBR

指令	RTN
INPUT	AC←InReg
OUTPUT	OutReg←AC
HALT	-
SKIPCOND	IF x[11-10] = 00 AND AC < 0 THEN PC←PC + 1 ELSE IF x[11-10] = 01 AND AC = 0 THEN PC←PC + 1 ELSE IF x[11-10] = 10 AND AC > 0 THEN PC←PC + 1
JUMP x	PC←x
CLEAR	AC←0
ADDI x	MAR←x MBR←M[MAR] MAR←MBR MBR←M[MAR] AC←AC + MBR

指令	RTN
JUMPI x	MAR←x MBR←M[MAR] PC←MBR
LOADI x	MAR←x MBR←M[MAR] MAR←MBR MBR←M[MAR] AC←MBR
STOREI x	MAR←x MBR←M[MAR] MAR←MBR MBR←AC M[MAR]←MBR



图片来源：Linda Null and Julia Lobur, Computer Organization and Architecture

指令执行过程和控制单元实现

- 根据指令集定义控制层逻辑
- 冯诺依曼模型的一个实例
- 工具：按模块的控制信号分析、加入信号模式的寄存器传输表示

信号模式	微操作
$P_3 P_2 P_1 P_0 T_3$	$\text{MAR} \leftarrow x$
$P_4 P_3 T_4 M_R$	$\text{MBR} \leftarrow M[\text{MAR}]$
$C_r A_0 P_5 T_5 L_{\text{alt}}$	$\text{AC} \leftarrow \text{AC} + \text{MBR}$

汇编语言

- 展示了汇编语言与机器语言的共同点和区别点
- 工具：(汇编)编译器

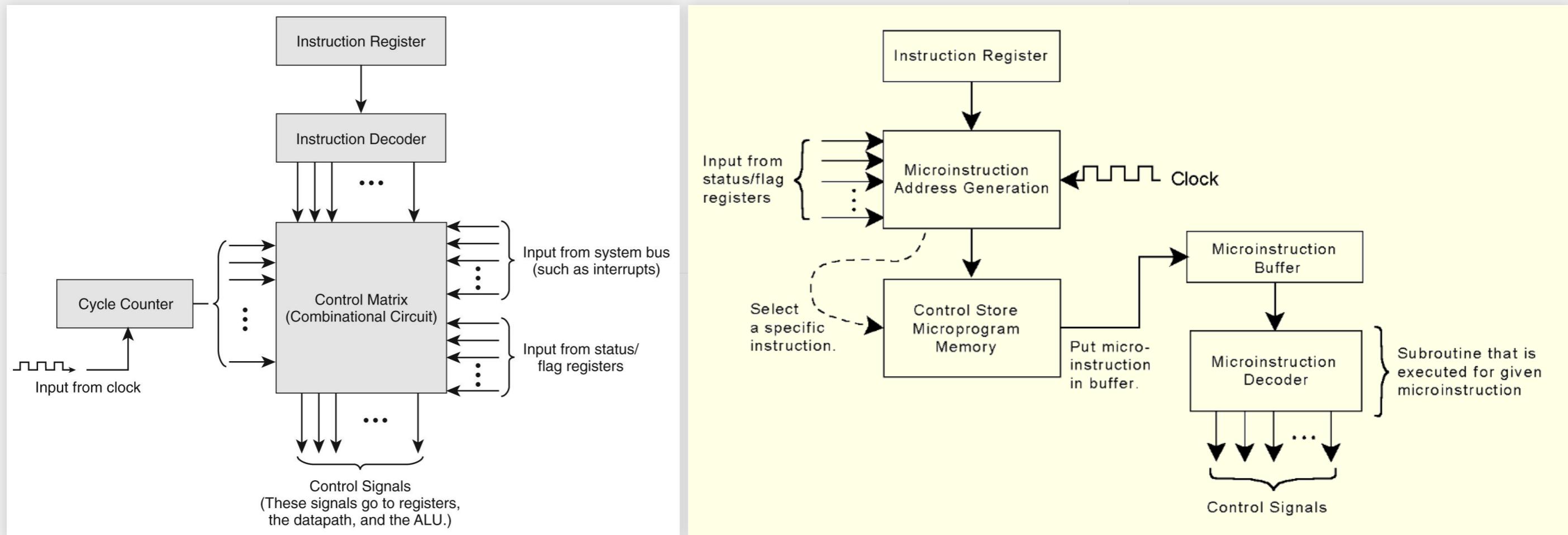
Address	Instruction	
100	Load	X
101	Add	Y
102	Store	Z
103	Halt	
104 X,	DEC	35
105 Y,	DEC	-23
106 Z,	HEX	0000

1 1 0 4
3 1 0 5
2 1 0 6
7 0 0 0
0 0 2 3
F F E 9
0 0 0 0

图片来源：Linda Null and Julia Lobur, *Computer Organization and Architecture*, 4th Edition

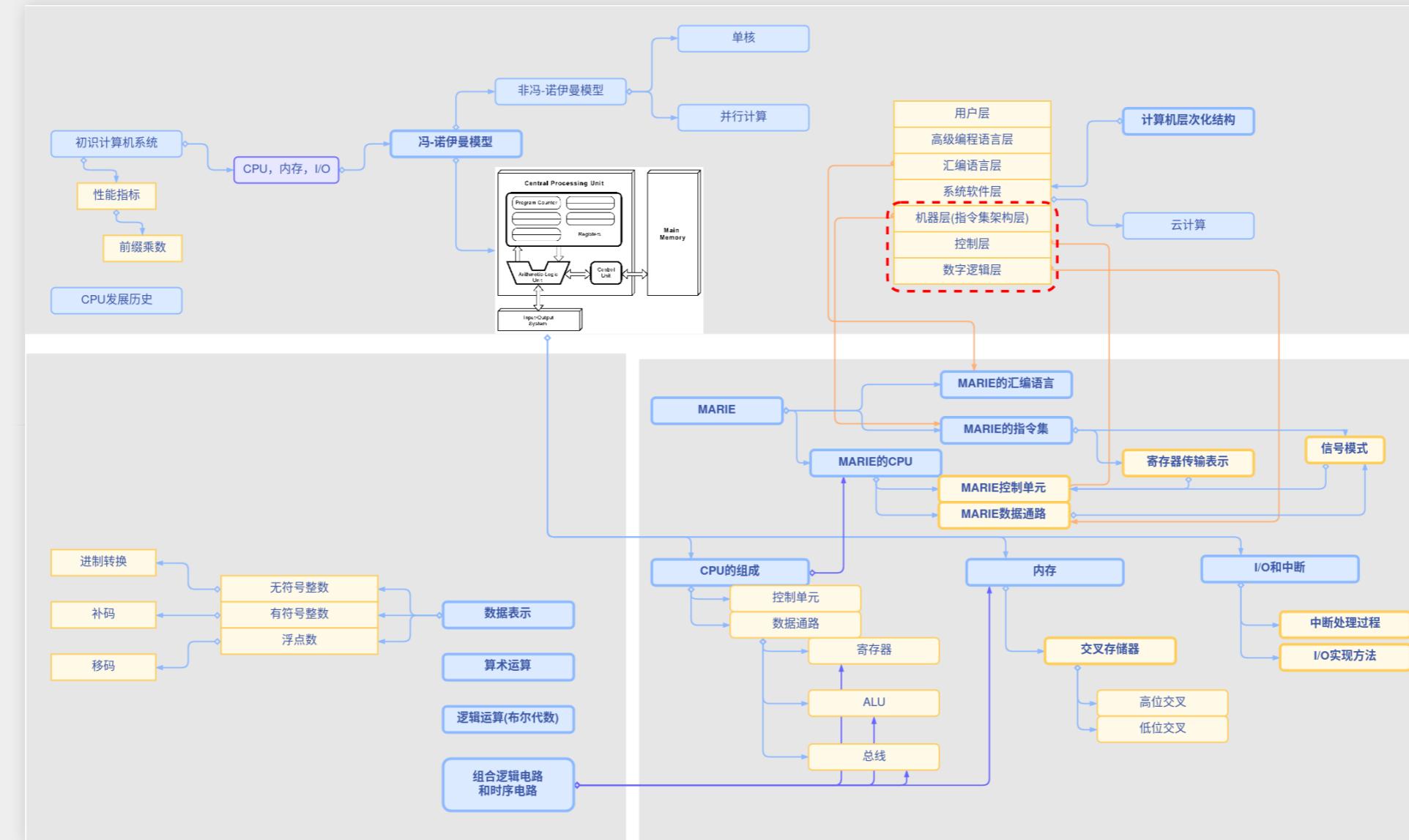
控制单元的实现

- 对应了计算机层次化结构中控制层的实现



图片来源：Linda Null and Julia Lobur, *Computer Organization and Architecture*, 4th Edition

各部分内容之间的联系





第四讲结束



Q & A