汇编语言程序设计

第1讲 指令集与计算机系统结构

参考教材

1. 《深入理解计算机系统》（Computer Systems： A Programmer‘s Perspective），第二版或第三版均可
   - 第2、3章
   - 课程ppt的部分素材也来源于该书作者的课程网站

2. SEE MIPS RUN （MIPS体系结构透视）第二版，2008

总分

- 考试 （60%） + 作业（10%） + 实验（30%）

Outline

- 课程介绍
- 汇编语言与计算机系统结构
- 典型指令集初步介绍

课程定位

“汇编课的应用二进制接口（ABI）内容简化了编译系统相关内容的讲解难度……”

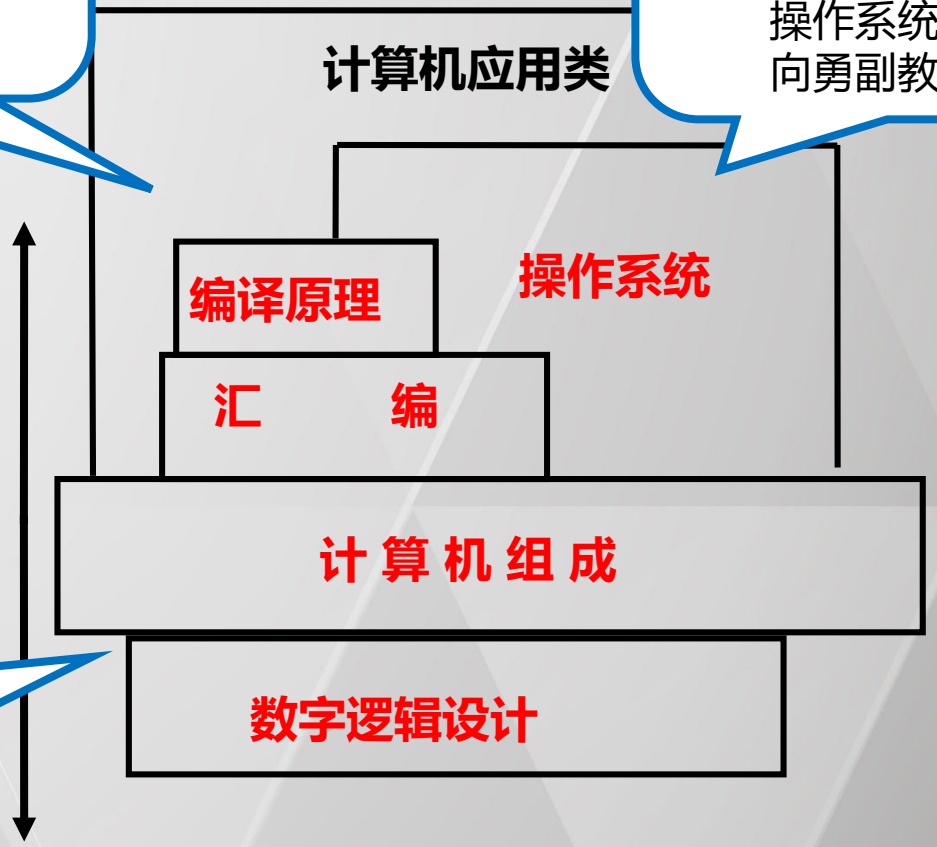
编译原理主讲
王生原副教授

“原先的学生不熟悉课程中涉及到的汇编指令，需要重新学习，现在这方面上手很快……”

操作系统课程主讲
向勇副教授

与计算机组成原理、编译原理、操作系统、数字逻辑设计等组成计算机系统核心课程

- **<u>汇编语言程序设计</u>与计算机组成原理作为软硬件界面起到“承上启下”的作用**



“汇编指令内容可以直接用于本课程的处理器设计……”

计算机组成原理主讲 刘卫东教授

为后续课程打下指令集、汇编编程以及微体系结构入门的基础。

看待计算机系统的不同角度

- 传统角度：从计算机构造者的角度入手
- **本课程角度：从编程者、使用者的角度入手**
- 其它角度：对于构建新型存算合一计算系统的启示
  - 补充内容

国外著名大学一二年级课程特点

MIT　CMU　Stanford　UC Berkeley

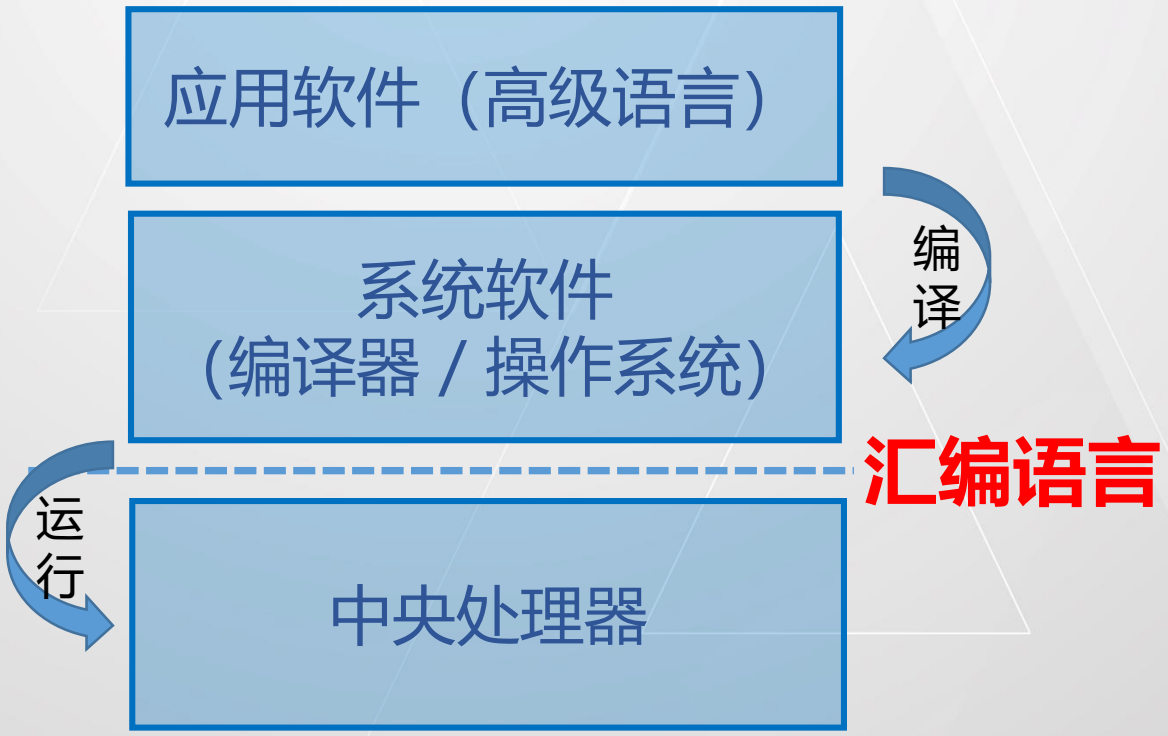
学期	MIT-EECS	MIT-CSE	CMU	Stanford	UCB
1 秋	18.01.微积分 2	18.01.微积分 2	21-120 Differential & Integral Calculus 10	以下 1 秋总计 18	MATH 1A (4)：微积分
	8.01 物理 1	8.01 物理 1	21-127 Concepts of Mathematics 10	MATH41(5)	
			15-122 Principles of Imperative Computation 10 命令式计算原理	THINK（4）+WRITING（4）编程方法学 程序结构	CS 10(4)：The Beauty and Joy of Computing 导论课
			15-128:Freshman Immigration Course 1	CS106A(5) Programming Methodology	Reading & Composition A (4)
			15-131:Great Practical Ideas in Computer Science (optional) 计算机科学中的实践思维	以下 1 冬总计 14	L&S Breadth(3)
			76-101:Interpretation and Argument 9	MATH42(5)	
			99-10x:Computing @ Carnegie Mellon 3	PHYSICS(4) 数据结构	
1 春	18.02 微积分 2	18.02 微积分 2	15-150 Principles of Functional Programming 10 函数式编程原理 程序/软件结构	CS106B(5) Programming Abstractions 编程抽象	EL ENG 16A(4) Designing Information Devices and Systems I
	8.02 物理 2	8.02 物理 2	15-251 Great Theoretical Ideas in Computer Science 12 计算机科学中的理论思维	以下 1 春总计 17	L&S Breadth(3)
	6.01 EECS 导论	6.01 EECS 导论	21-122 Integration, Differential Equations, and Approximation 10	MATH 选(5)+ PHYSICS43 (4) +ENGR 40M(5) Introductory Electronics +Intro Sem(3)	CS 61A(4) 程序结构 The Structure and Interpretation of Computer Programs
			1 人文选+1 科学选：总计 18		Reading & Composition A (4)
2 秋	18.03 微分方程	18.06 线性代数	21-241 Matrices and Linear Transformations 10	以下 2 秋总计 19	EL ENG 16B (4) Designing Information Devices and Systems II
	6.041/6.042 概率论/离散数学 数据结构	6.042 离散数学	15-213 Introduction to Computer Systems 12 计算机系统导论 机器结构	CS103(5) Mathematical Foundations of Computing +CS107(5) Computer Organization and Systems 机器结构	CS 61B(4) 数据结构 Data Structures
	CS Fundation		15-221 Technical Communication for Comp- uter Scientists 9 数据结构	语言(5)+写作(4) 计算机组织与系统	L&S Breadth(4)
			1 科学选+1 自选：总计 18	以下 2 冬总计 15	L&S Breadth(3)
2 春	6.02 EECS 导论 机器结构	6.02 EECS 导论	15-210 Parallel and Sequential Data Structures and Algorithms 12 数据结构与算法	语言(5)	CS 61C(4) 机器结构 Machine Structures
	EE Foundation 软件结构	CS Foundation	1 CS 选+1 人文选+1 科学选+1 自选：总计 36	CS109(5) Introduction to Probability for Computer Scientists 计算机系统原理 +CS110(5) Principles of Computer Systems	CS 70(4) Discrete Mathematics and Probability Theory
	CS [illegible]	CS [illegible]		CS [illegible] Compilers 自选(4)+人文(4)	[illegible]+人文(4)

课程名字虽然不同，但课程内涵一致：思维与结构

美国一流大学相关必修课情况

	MIT	UC Berkeley	stanford	CMU
课程名称	Computation Structures	Great Ideas in Computer Architecture （Machine Structures ）	Computer Organization and Systems（COS）	Introduction to Computer System（ICS）
开设专业*	EE、CS	CS	CS	CS
课程描述	门电路→功能部件→单周期和流水线 CPU；C 语言→汇编→指令→过程→进程；并行、性能评价	C 语言→汇编→指令；应用级并行→数据级并行→线程级并行→指令级并行→寄存器传送级硬件描述	C 语言→汇编→指令→微体系结构；编译→链接→装入→执行；程序性能优化、存储器结构与管理、并发和多线程、网络编程	C 语言→汇编→指令→微体系结构；编译→链接→装入→执行；程序性能优化、存储器结构与管理、并发和多线程、网络编程
教材**	未指定（讲义）	C"K&R"+ COD "P&H" + WSC"B&H"	PP"B&O"+ C"K&R"	APP"B&O"+ C"K&R",
模型机	自定义（RISC）	MIPS（RISC）	IA32	IA32
助教人数	12	7	15	未列出
先行课程或要求	了解编程（函数式）和数据结构基础，具备电子技术基础知识	了解和掌握编程（函数式）和数据结构基础，具备电子技术基础知识	了解和掌握编程和数据结构基础，具备电子技术基础知识	了解和掌握编程（函数式）和数据结构基础，具备电子技术基础知识
实验内容	共 8 个实验，涉及门电路特性、ALU、图灵机、汇编、处理器设计、鼠标中断方式 I/O 等	共 12 个实验，4 个大作业，涉及到 Debug、EC2、MapReduce，MIPS 汇编、数据级并行、线程级并行、Cache、虚拟存储管理等，大作业和实验成绩占 60%	共 8 个实验和 7 个大作业，涉及数据的表示、堆区分配、过程调用和栈的构成及使用、溢出、编译工具和编译优化等，大作业和实验成绩占 60%	共 7 个实验，涉及数据的表示、Cache、缓冲区溢出、过程调用及栈的构成与使用、堆的分配、代理设置等，实验成绩占 50%
实验手段	各类模拟器	编程、云计算平台、模拟器	编程	编程
相关后继课程或教学内容	“数字系统设计”和“计算机体系结构及系统”等，涉及 CPU 设计实验和体系结构方面的模拟实验	“数字系统设计”和“计算机体系结构及工程”等，涉及 CPU 设计实验和体系结构方面的模拟实验，前者是同时为 CS 和 EE 的学生开设的课程	后继课程为“Digital Systems II”，教材为 COD"P&H"，主要是流水线 CPU 和存储系统设计，实验为用 DHL 设计 CPU（CS 学生不需要）	CS 学生可选 ECE 开设的课程“Introduction to Computer Architecture”，教材为 COD"P&H"，主要是流水线 CPU 和存储系统设计，实验为用 Verilog 语言设计 CPU

具体讲什么?



计算机系统层次结构
(system hierarchy)

核心内容

- 高级语言程序（C语言）在机器层面的表示与运行

基本目标

- 将程序的执行与计算机的工作过程紧密联系起来
- 为后续课程，如编译原理、计算机组成原理等提供先导知识

```c
int array[2] = {1, 2};
int main()
{
    int val = sum(array, 2);
    return val;
}
```
main.c

编译

```
0000000000000000 <array>:
 0:   01 00                   add    %eax,(%rax)
 2:   00 00                   add    %al,(%rax)
 4:   02 00                   add    (%rax),%al
0000000000000000 <main>:
 0:   48 83 ec 08             sub    $0x8,%rsp
 4:   be 02 00 00 00          mov    $0x2,%esi
 9:   bf 00 00 00 00          mov    $0x0,%edi
                      a: R_X86_64_32 array
 e:   e8 00 00 00 00          callq  13 <main+0x13>
                      f: R_X86_64_PC32 sum-0x4
 13:   48 83 c4 08            add    $0x8,%rsp
 17:   c3                     retq
```
main.o

汇编指令

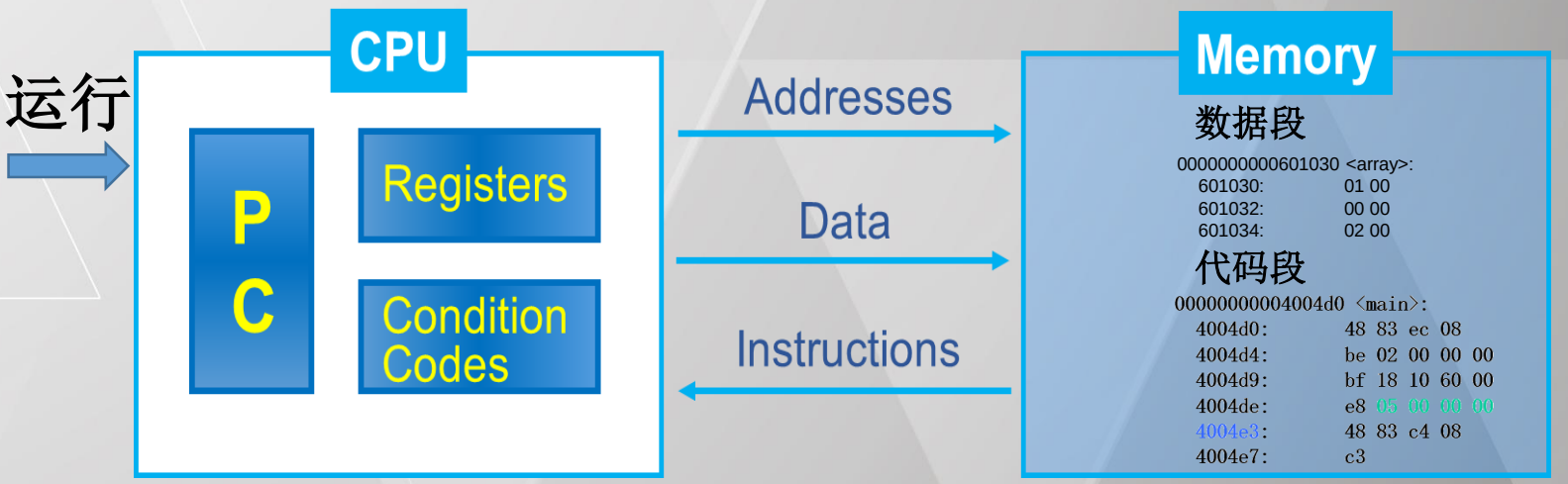
机器指令

链接

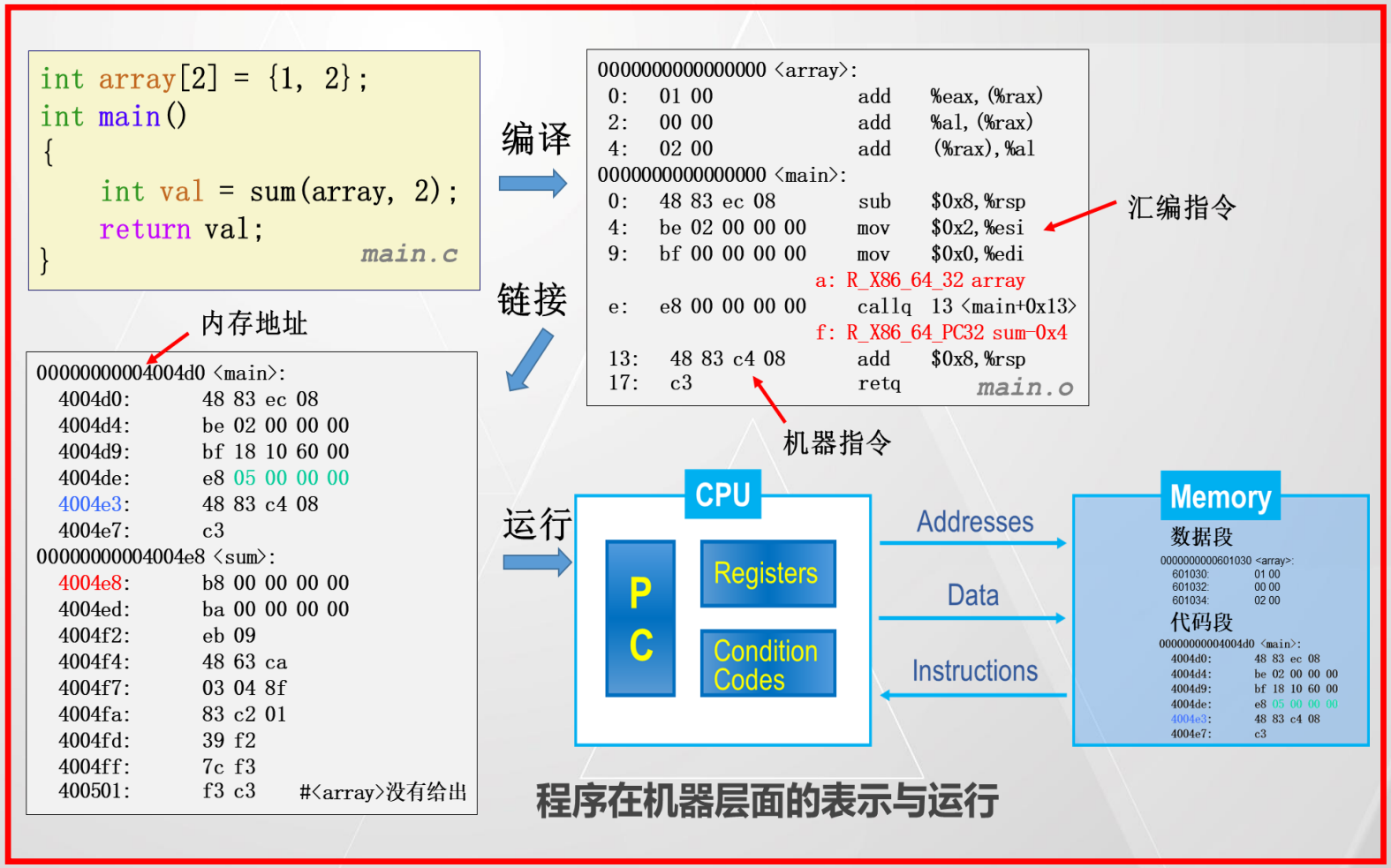
内存地址

```
00000000004004d0 <main>:
  4004d0:       48 83 ec 08
  4004d4:       be 02 00 00 00
  4004d9:       bf 18 10 60 00
  4004de:       e8 05 00 00 00
  4004e3:       48 83 c4 08
  4004e7:       c3
00000000004004e8 <sum>:
  4004e8:       b8 00 00 00 00
  4004ed:       ba 00 00 00 00
  4004f2:       eb 09
  4004f4:       48 63 ca
  4004f7:       03 04 8f
  4004fa:       83 c2 01
  4004fd:       39 f2
  4004ff:       7c f3
  400501:       f3 c3      #<array>没有给出
```

运行



程序在机器层面的表示与运行



（看上去）一个程序占据了一个处理器以及一块完整的内存空间

- 在编译、操作系统、处理器共同支持下实现
- 课程讲解这儿涉及到的基本概念与过程
- 完整的实现要到后续课程中学习

C程序在硬件层面的表示

```
int array[2] = {1, 2};
int main()
{
    int val = sum(array, 2);
    return val;
}
```
main.c

编译

```
0000000000000000 <array>:
 0:   01 00                   add    %eax,(%rax)
 2:   00 00                   add    %al,(%rax)
 4:   02 00                   add    (%rax),%al
0000000000000000 <main>:
 0:   48 83 ec 08             sub    $0x8,%rsp
 4:   be 02 00 00 00          mov    $0x2,%esi
 9:   bf 00 00 00 00          mov    $0x0,%edi
                      a: R_X86_64_32 array
 e:   e8 00 00 00 00          callq  13 <main+0x13>
                      f: R_X86_64_PC32 sum-0x4
 13:   48 83 c4 08            add    $0x8,%rsp
 17:   c3                     retq
```
main.o

汇编指令

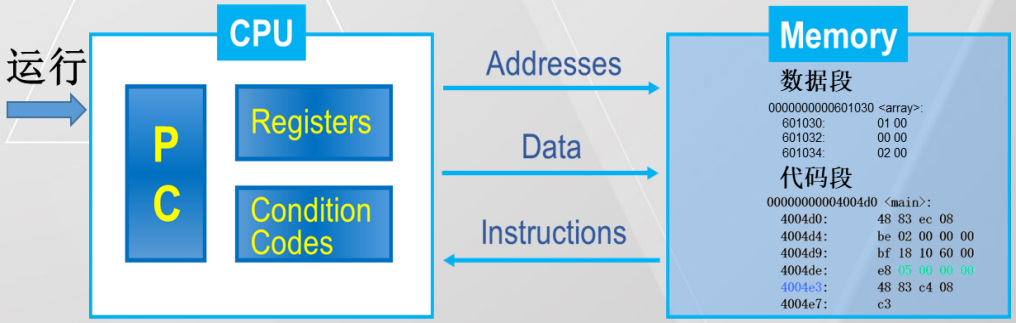
机器指令

链接

内存地址

```
00000000004004d0 <main>:
  4004d0:       48 83 ec 08
  4004d4:       be 02 00 00 00
  4004d9:       bf 18 10 60 00
  4004de:       e8 05 00 00 00
  4004e3:       48 83 c4 08
  4004e7:       c3
00000000004004e8 <sum>:
  4004e8:       b8 00 00 00 00
  4004ed:       ba 00 00 00 00
  4004f2:       eb 09
  4004f4:       48 63 ca
  4004f7:       03 04 8f
  4004fa:       83 c2 01
  4004fd:       39 f2
  4004ff:       7c f3
  400501:       f3 c3          #<array>没有给出
```

运行



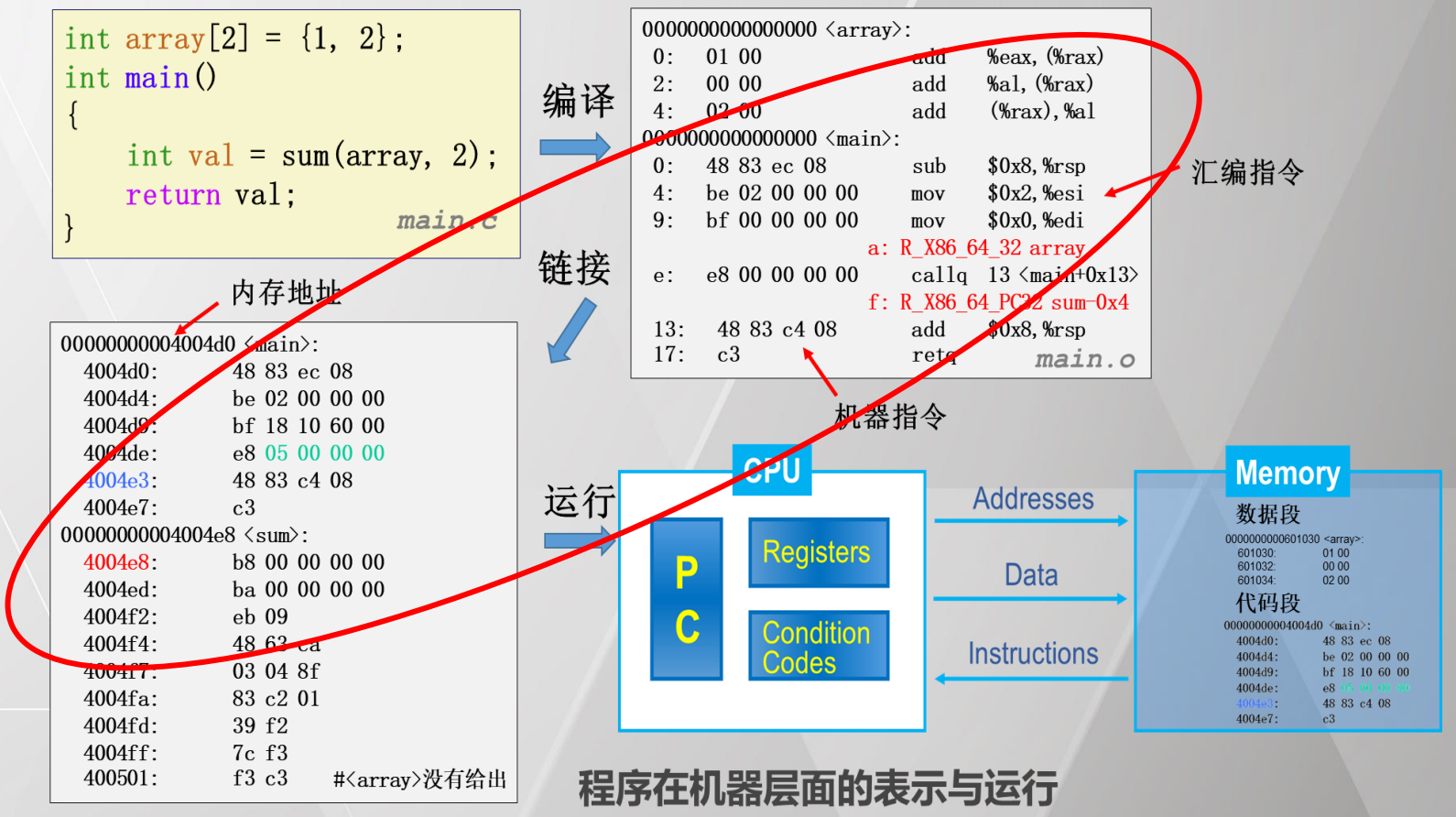
程序在机器层面的表示与运行

- 数据
  - 整数（第二讲）
  - 浮点数（第三讲）
  - 数组、结构（第八讲）
- 代码
  - 基本概念/基本指令/寻址方式（第五讲）
  - 程序控制流与相关指令（第六讲）
  - 函数调用与相关指令（第七讲）

编译器如何工作由后续课程讲授

C程序在硬件层面的表示

- 数据/代码的内存地址定位
  - 链接（第九讲）
- 数据/代码的内存布局
  - 栈、堆等各类数据段以及代码段的layout（第十讲）
  - 缓冲区溢出等（第十讲）
- 讲解基本调试工具（GDB）的使用



程序在机器层面的表示与运行

```c
int array[2] = {1, 2};
int main()
{
    int val = sum(array, 2);
    return val;
}
```
main.c

编译

```
0000000000000000 <array>:
 0:   01 00                   add    %eax,(%rax)
 2:   00 00                   add    %al,(%rax)
 4:   02 00                   add    (%rax),%al
0000000000000000 <main>:
 0:   48 83 ec 08             sub    $0x8,%rsp
 4:   be 02 00 00 00          mov    $0x2,%esi
 9:   bf 00 00 00 00          mov    $0x0,%edi
                      a: R_X86_64_32 array
 e:   e8 00 00 00 00          callq  13 <main+0x13>
                      f: R_X86_64_PC32 sum-0x4
 13:   48 83 c4 08            add    $0x8,%rsp
 17:   c3                     retq
```
main.o

汇编指令

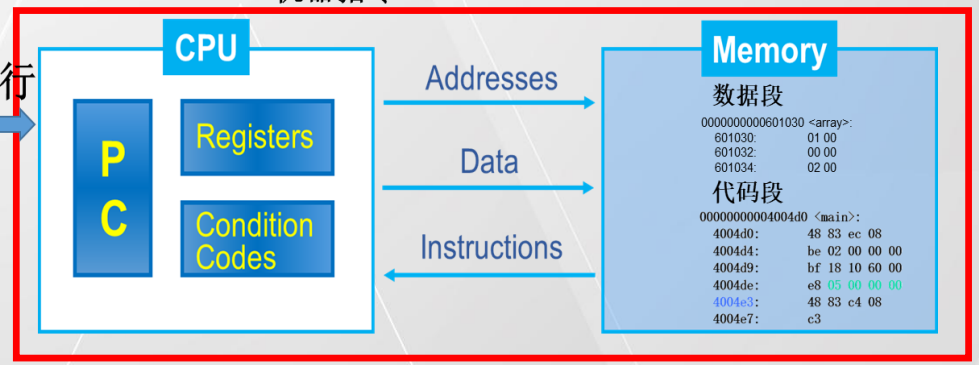
链接

内存地址

```
00000000004004d0 <main>:
  4004d0:       48 83 ec 08
  4004d4:       be 02 00 00 00
  4004d9:       bf 18 10 60 00
  4004de:       e8 05 00 00 00
  4004e3:       48 83 c4 08
  4004e7:       c3
00000000004004e8 <sum>:
  4004e8:       b8 00 00 00 00
  4004ed:       ba 00 00 00 00
  4004f2:       eb 09
  4004f4:       48 63 ca
  4004f7:       03 04 8f
  4004fa:       83 c2 01
  4004fd:       39 f2
  4004ff:       7c f3
  400501:       f3 c3          #<array>没有给出
```

机器指令

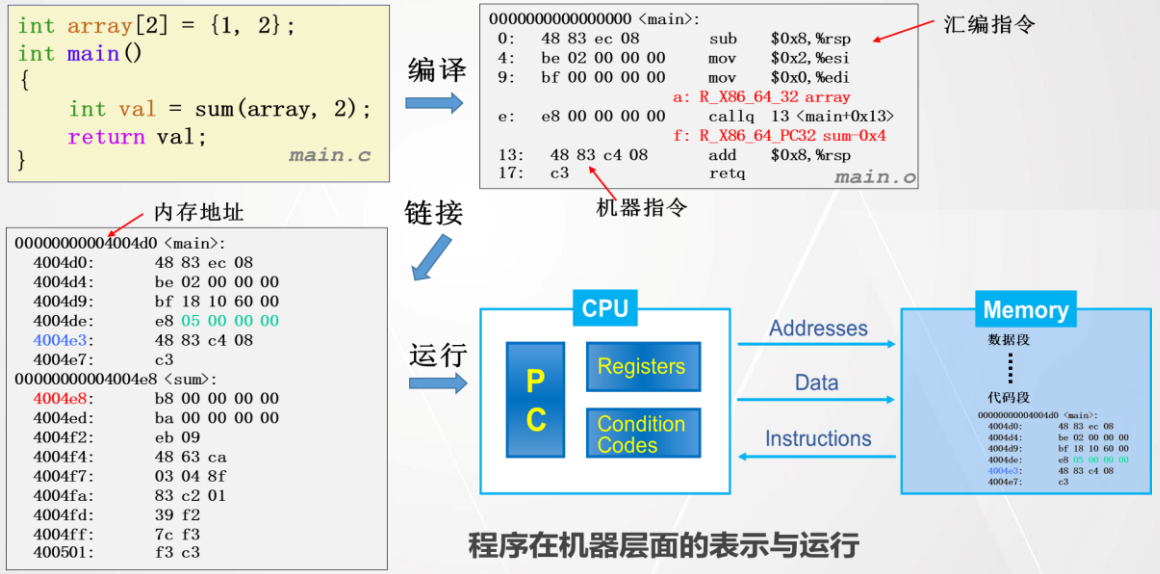
运行



程序在机器层面的表示与运行

在硬件层面的运行

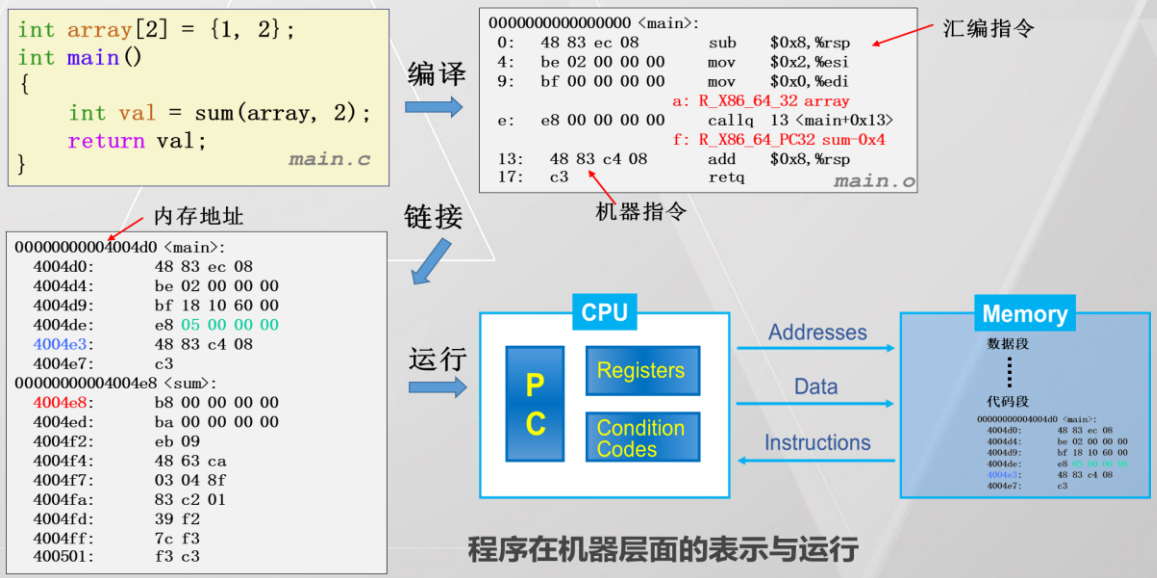
- 线性内存地址的基本概念（第四讲）
- 初步介绍下处理器流水线的概念（第六讲）
  - 具体在组成原理等后续课程

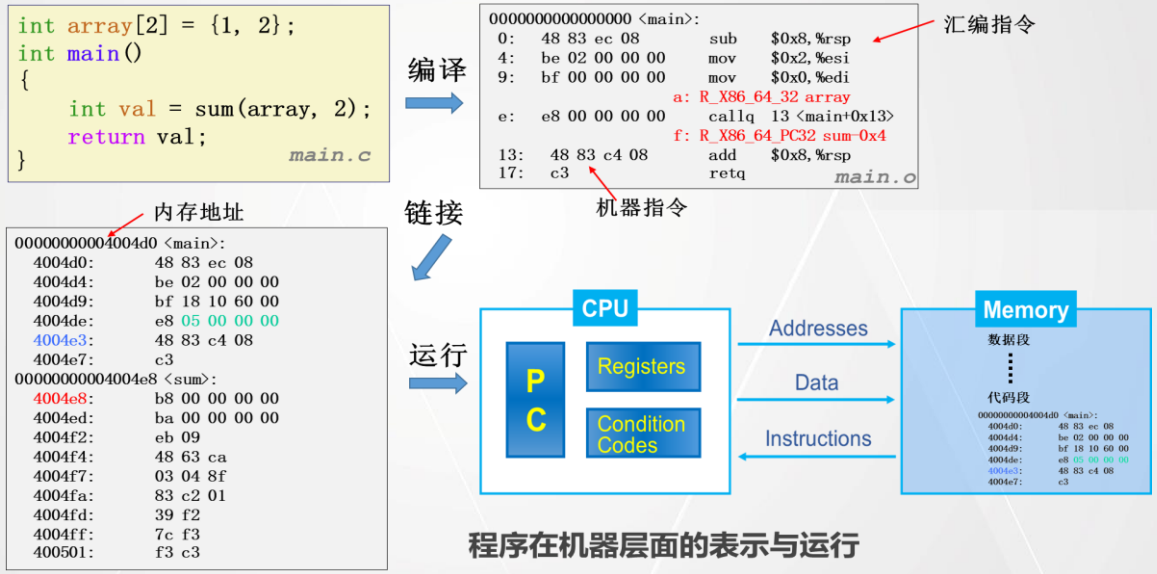


（看上去）一个程序占据了一个处理器以及一块完整的内存空间。

但是实际情况要远为复杂！

怎么做到的？

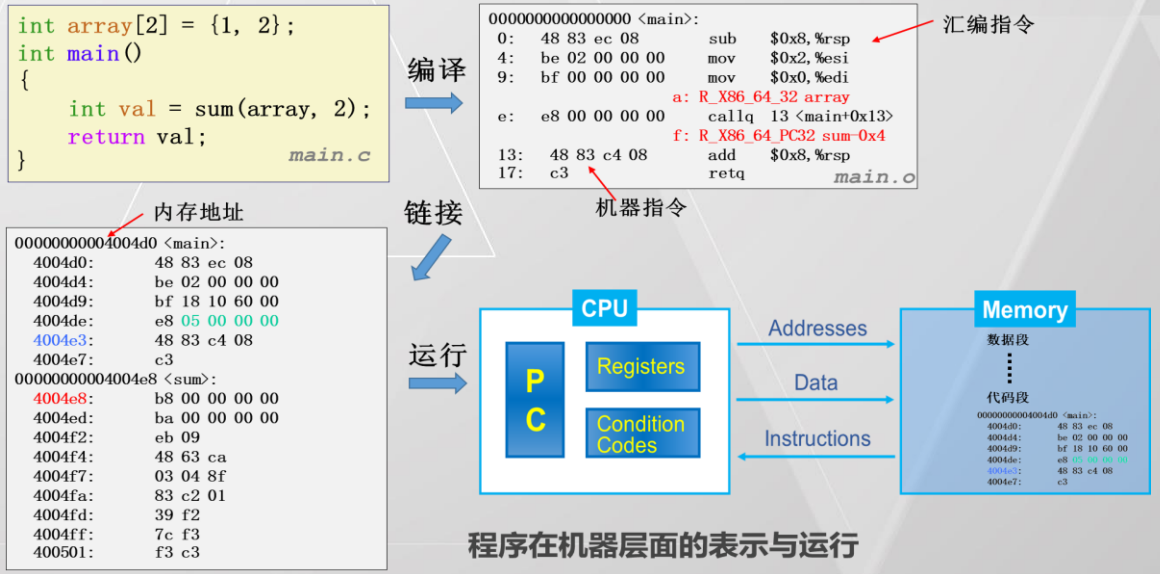




（看上去）一个程序占据了一个处理器以及一块完整的内存空间，但是实际情况要远为复杂！ 怎么做到的？

关系到计算机系统的两个重要概念：

- 虚存——在有限物理内存前提下设计出连续的、相互独立的虚拟内存
- 异常——各个任务切换的重要机制（当然异常还有很多其他作用）



- 虚存——在有限的物理内存前提下设计出连续的、独立的虚拟内存
- 异常——各个任务切换的重要机制（当然异常还有很多其他作用）

与系统课组结合，采用MIPS 32指令集（逐步更新到RISC-V）

- MIPS 32处理器结构与指令集初步（第十二讲）
- MIPS32指令集与编程（第十三讲）
- MIPS32异常处理（第十四讲）
- 虚存与MIPS 32内存管理（第十五讲）

Outline

- 课程介绍
- 汇编语言与计算机系统结构
- 典型指令集初步介绍

计算机系统结构

计算机系统结构（中国计算机科学技术百科全书第一版的定义）

- 计算机系统的物理或者硬件结构、各部分组成的属性以及这些部分的相互联系
- 系统软件开发人员看到的计算机系统的功能行为和概念结构
- 计算机系统的结构与实现（计算机组成）

计算机系统结构

其它定义：对计算机系统中各级之间界面的划分和定义，以及对各级界面上、下的功能进行分配

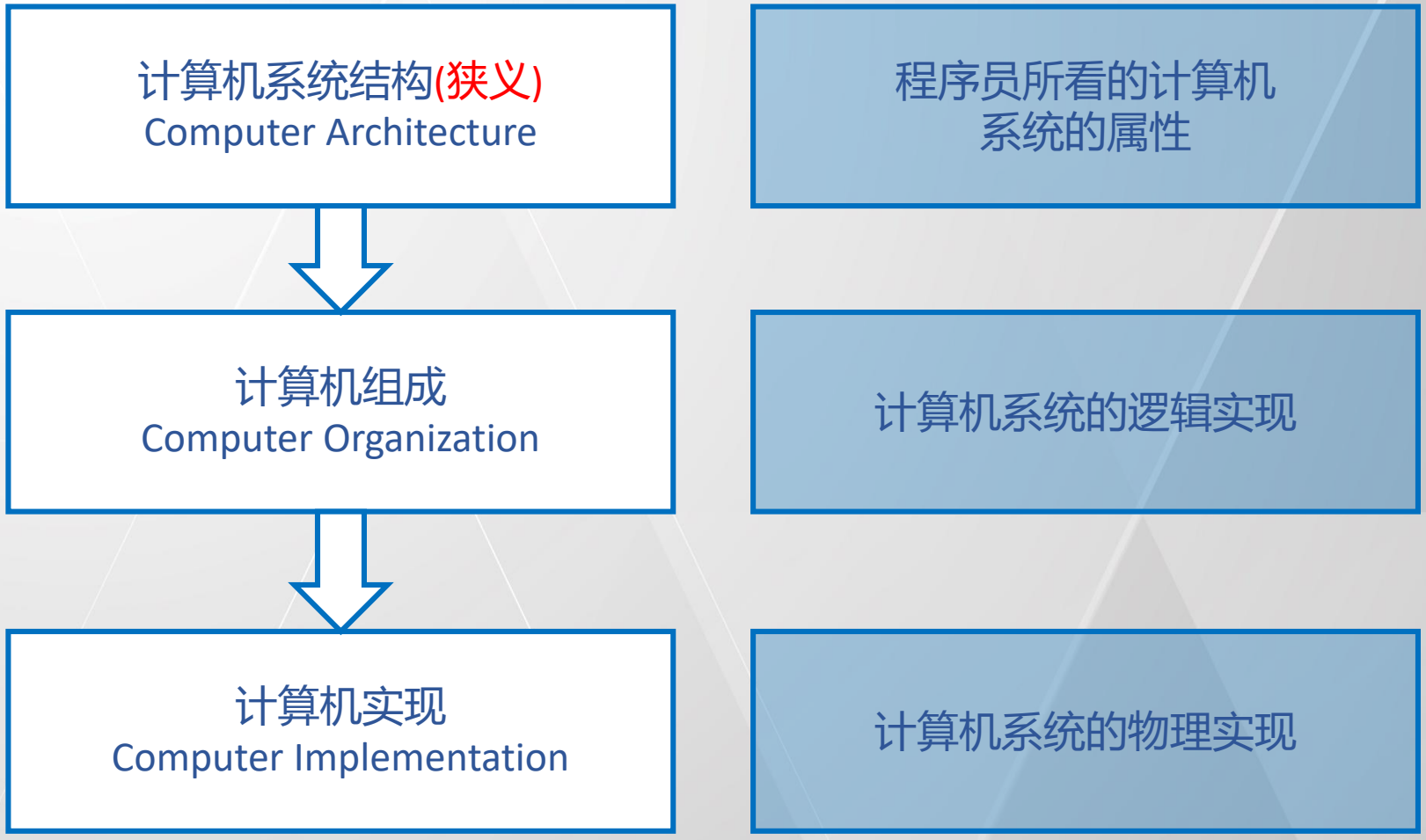
- 1964年，IBM/360系列机的总设计工程师G.M.Amdahl、G.A.Blaauw、F.P.Brooks等人提出。也称体系结构

- 是从系统程序员的角度所看到的系统的属性，是概念上的结构和功能上的行为
  - 程序员：系统程序员（包括：**汇编语言**、机器语言、编译程序、操作系统、虚拟机等）
  - 看到的：编写出能在机器上正确运行的程序所必须了解到的

- 它不同于数据流程和控制的组织，不同于逻辑设计以及物理实现方法

一般认为这是一种狭义的定义

计算机系统结构



计算机系统结构是研究计算机系统自身的学科

范围（广义）

指令系统（ Instruction set architecture, or ISA 或指令集体系结构)

机器语言，还包括机器字长、内存地址模式、处理器寄存器等程序员可见的系统状态、数据格式等

微体系结构（Micro-architecture)

如何实现指令集

处理器内部的实现，包括流水线、处理部件、缓存等内容

计算机系统

计算机系统里的其它硬件，包括总线、交换开关(Switch)、内存控制器、DMA控制器等

其他内容

虚拟化、计算机集群等等

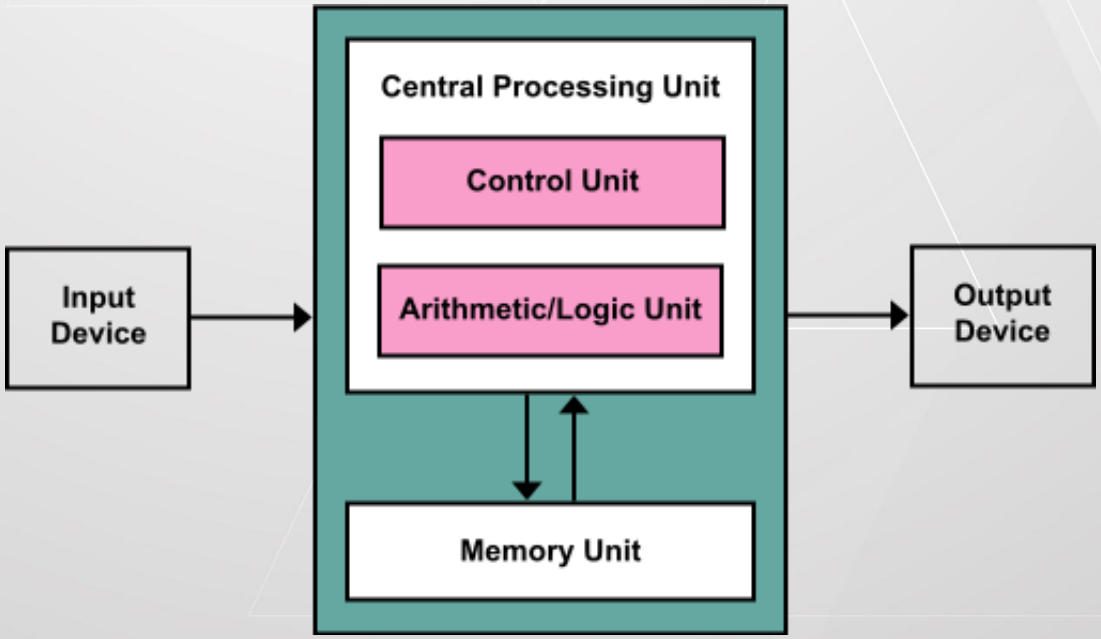
指令系统

指令系统（汇编语言可以看做是它的一种助记符）

计算机处理器对外提供的主要接口与规格

软硬件的分界

系统程序员看到的计算机的主要属性



A design architecture for an electronic digital computer with parts consisting of a **processing unit** containing **an arithmetic logic unit and processor registers, a control unit containing an instruction register and program counter**, **a memory** to store both data and instructions, external mass storage, and input and output mechanisms.

John Von Neumann，1903－1957 (From wiki)

Long live the von Neumann architecture!

指令系统分类

CISC （复杂指令系统，Complex Set Instruction Computer ）

面向高级语言，缩小机器指令系统与高级语言语义差距

指令条数多，寻址方式多变

单条指令功能相对复杂

代表：X86

RISC（精简指令系统， Reduced instruction set computer)

通常只支持常用的能在一个周期内完成的操作（80：20原则）

简单而统一的指令格式

只有LOAD和STORE指令可以访问存储器，简单的寻址方式

较多的寄存器

指令条数相对较少，依赖于编译器产生高效的代码

处理器微体系结构相对简单，运行频率高

代表：MIPS / ARM / PowerPC / RISC-V

指令系统分类

CISC与RISC走向融合

X86处理器内部采用类似RISC的micro-op

出于兼容性考虑，其指令集一直属于CISC

经典的RISC指令集也日益扩展、复杂化

PowerPC指令集(RISC)有超过230多条指令，也很复杂

主要的区分标准——Load / store with (without) other operations?

通用指令系统的共性——图灵完备性

- 最简单的指令集

```
subneg a, b, c ; Mem[b] = Mem[b] - Mem[a]
               ; if (Mem[b] < 0) goto c
```

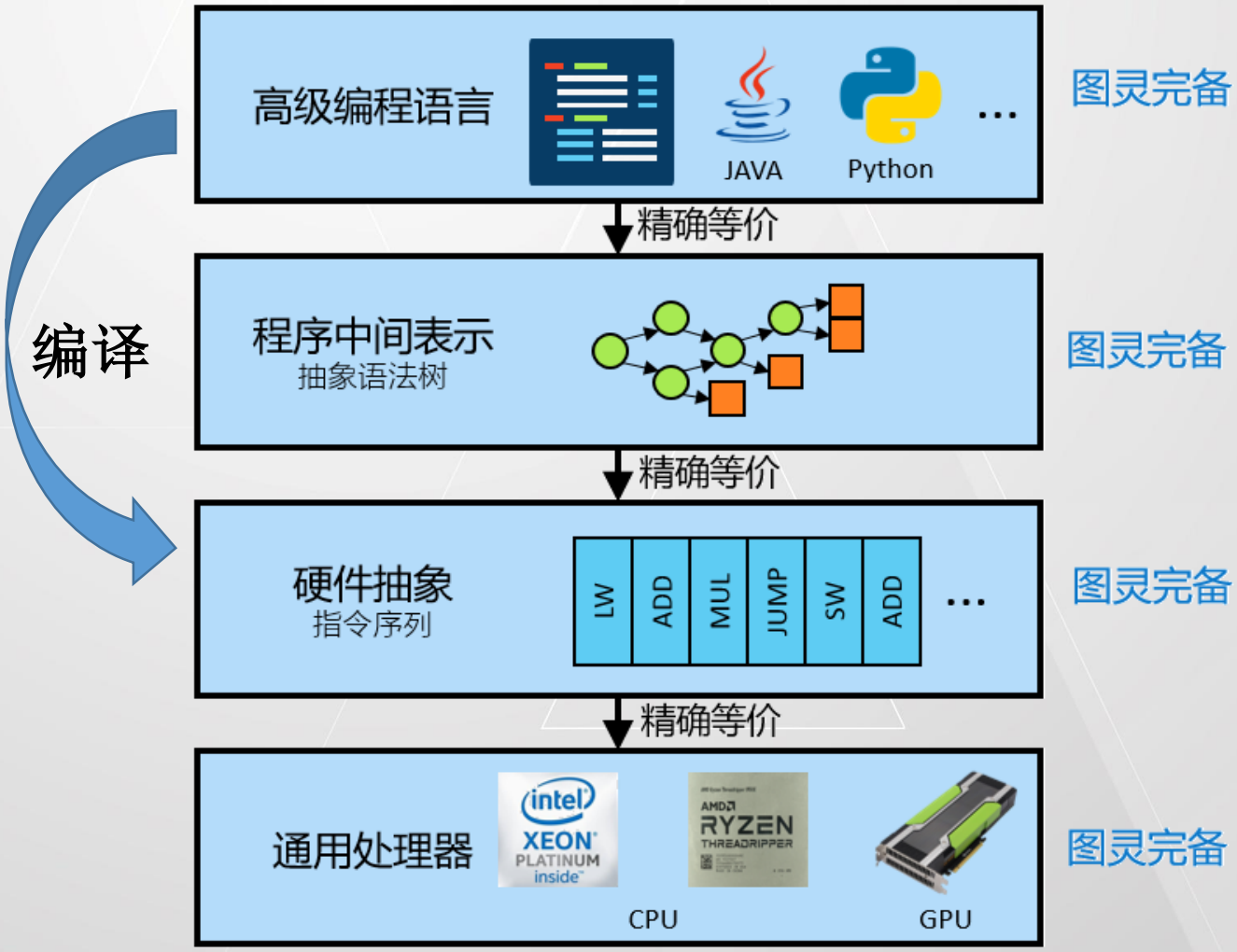
Carbon nanotube computer. NATURE. Volume 501, pp.526–530 (26 September 2013)

- 实用的指令集
  - X86指令集
    - 1000多条*
  - RISC-V基本指令集
    - 40条指令
    - 包含整数的基本计算、Load/ Store和控制流

*Intel® 64 and IA-32 Architectures Software Developer’s Manual.

- Statefulness Principle
The instruction set must be stateful by state and a next state function
- State Change Principle
There must be an instruction to modify state
- Next State Change Principle
There must be an instruction to modify next state function

通用计算机与编程语言的图灵完备性



- 软硬件解耦合的计算机层次结构是通用计算机迅猛发展的体系结构基础
- 图灵完备性是这一层次结构的可行性基石

Outline

- 课程介绍
- 汇编语言与计算机系统结构
- 典型指令集初步介绍

X86指令集

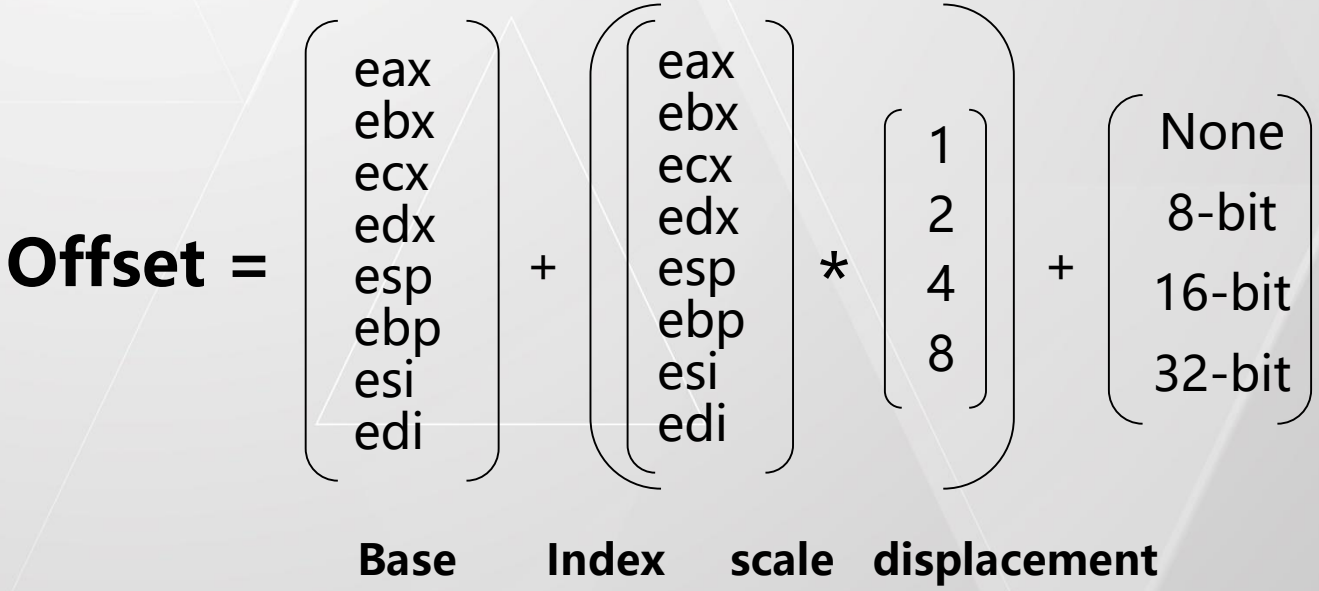
X86指令集的基本特色

向下兼容

变长指令

1-15 字节，多数为2-3字节长度

多种寻址方式（可访问不对齐内存地址）



X86-32指令集为例

X86指令集的基本特色

指令集的通用寄存器个数有限

X86-32系统下拥有8个通用寄存器（x86-64扩展到16个）

至多能有一个操作数在内存中，另一个操作数为立即数或者寄存器

x86-32/64 General Purpose Registers

%rax	%eax	%r8	%r8d
%rdx	%edx	%r9	%r9d
%rcx	%ecx	%r10	%r10d
%rbx	%ebx	%r11	%r11d
%rsi	%esi	%r12	%r12d
%rdi	%edi	%r13	%r13d
%rsp	%esp	%r14	%r14d
%rbp	%ebp	%r15	%r15d

为何X86指令集长久不衰?

商业上的成功是其主要原因（生态环境）

技术上注重向下兼容

一个反例是Itanium（IA-64， Explicitly Parallel Instruction Computing ），技术创新但是无法兼容，已“死亡” （2021年7月29日，Intel正式停止出货安腾9700系列处理器）

X86指令集的缺点?

向下兼容导致指令集越来越大、越复杂

类RISC内核，采用micro-op模式进行翻译，使得功耗相对增大，这导致其在注重低功耗的领域不易占优势

对很多领域而言，资源利用率低

在高性能计算领域，普遍使用的300余条X86指令中，大致只有80余条是被科学计算所需要的（美国劳伦斯伯克利国家实验室的研究， 2009）

MIPS指令集

经典的RISC指令集

MIPS I、MIPS II、MIPS III、MIPS IV到MIPS V，嵌入式指令体系MIPS16、MIPS32到MIPS64的发展已经十分成熟

为充分利用处理器的流水线结构，其设计思想是使得各个指令的流水线分段较为均匀

分段一致，每段的操作时延相差不多，以提高主频

尽量利用软件办法避免流水线中的控制相关问题

实例：Branch Delay Slot

RISC提出者之一John Hennessy创办了MIPS公司；本世纪初，ARM与MIPS两者发展几乎不分伯仲（1999年以前MIPS架构是世界上用得最多的处理器，按片数计算的话）

经典的RISC指令集

以寄存器为中心（32个），只有Load/Store指令访问内存，所有的计算类型的指令均从寄存器堆中读取数据并把结果写入寄存器堆中。
MIPS32还定义了32个浮点寄存器

MIPS32指令集的指令格式非常规整，所有的指令长度一定，而且指令操作码在固定的位置上。

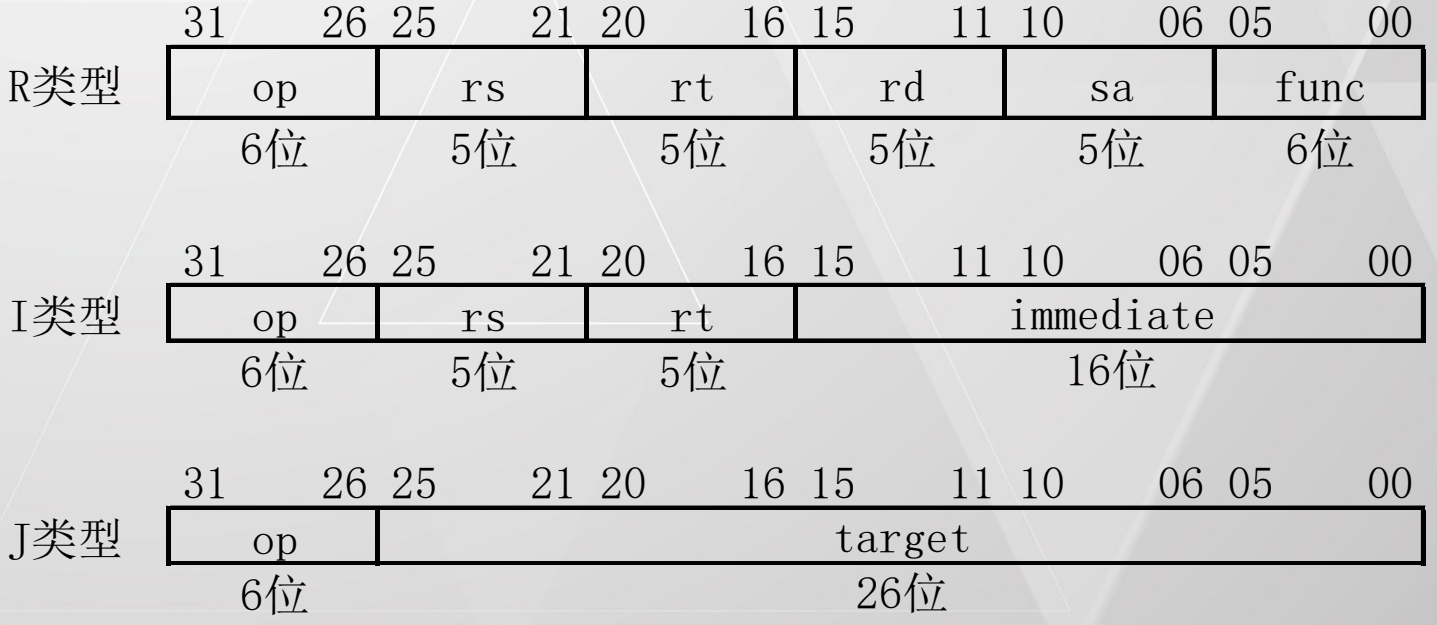
MIPS指令的寻址方式非常简单，每条指令的操作也较简单

MIPS32™的指令格式只有3种

R（register）类型的指令从寄存器堆中读取两个源操作数，计算结果写回寄存器堆

I（immediate）类型的指令使用一个16位的立即数作为源操作数

J（jump）类型的指令使用一个26位立即数作为跳转的目标地址（target address）



Load和Store指令都为立即数（I-type）类型，用来在存储器和通用寄存器之间的储存和装载数据。MIPS指令集只有该类指令能访问内存，而其它指令操作寄存器

该类指令只有基址寄存器的值加上扩展的16位有符号立即数一种寻址模式，数据的存取方式可以是字节（byte）、半字（half-word）、字（word）和双字（Double word）

MIPS扩展指令集

MIPS-3D， 浮点SIMD 用于三维几何处理（MIPS64架构下）

Vertex transformation (matrix multiplication)

Clip-check (compare and branch)

Transform to screen coordinates (perspective division using reciprocal)

Lighting : infinite and local (normalization using reciprocal square root)

采用MIPS64浮点运算单元和双单精度数据类型。

PS (paired-single，双单精度)操作可对64位寄存器中的两个32位浮点值进行运算，从而提供2路SIMD (单指令多数据)能力

MIPS扩展指令集

MIPS16e——16位指令，指令集被压缩，代码存储容量要求减小，从而降低系统成本；相比MIPS32，利用MIPS16e编译的应用程序平均减小30%

32个通用寄存器中有8个可用于MIPS16e模式

与MIPS32一起使用时，支持8位、16位和32位数据类型；与MIPS64一起使用时，支持8位、16位、32位和64位数据类型

MIPS16e 和 MIPS32/64之间的模式切换支持：通过一条特殊的跳转指令来实现模式切换的软件控制

RISC-V指令集

很像MIPS的指令集

与MIPS近乎同源（David Patterson教授主导，RISC的发明人之一）

模块化与增量型 ISA——

RV32I (64I)：32个通用寄存器，40条指令（整数计算、load/store、条件分支、无条件跳转、杂项指令；RV64I额外增加了12条）

RV32M：添加了整数乘法和除法指令

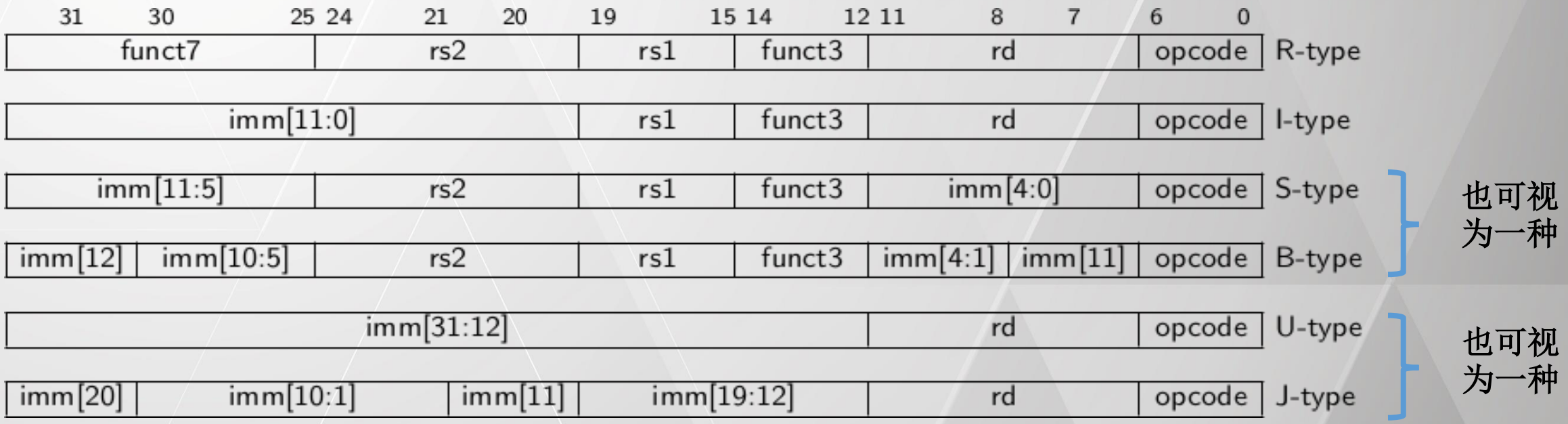
RV32A：原子操作指令

RV32F：单精度浮点数指令

RV32D：双精度浮点数指令

RV32V：向量指令

RISC-V指令格式


31 30 25 24 21 20 19 15 14 12 11 8 7 6 0
funct7 rs2 rs1 funct3 rd opcode R-type
imm[11:0] rs1 funct3 rd opcode I-type
imm[11:5] rs2 rs1 funct3 imm[4:0] opcode S-type
imm[12] imm[10:5] rs2 rs1 funct3 imm[4:1] imm[11] opcode B-type
imm[31:12] rd opcode U-type
imm[20] imm[10:1] imm[11] imm[19:12] rd opcode J-type
也可视为一种
也可视为一种

与MIPS指令集的对比*

	MIPS32	RV32I
简洁性	立即数支持零扩展及符号扩展；一些算术指令会造成溢出异常	立即数只进行符号扩展；算术指令不抛出异常
指令集和具体实现的分离	Branch Delay Slot；乘除使用单独的HI，LO寄存器	分支指令没有延迟槽；乘除指令通用寄存器
易于编程/编译/链接	内存数据必须对齐、不规则的数据寻址模式	内存数据可以不对齐、PC 相对的数据寻址模式

*参考了RISC-V 手册。http://crva.ict.ac.cn/documents/RISC-V-Reader-Chinese-v2p1.pdf

本讲心得

计算机体系结构的层次化设计理念

- 汇编指令（指令集体系结构）是软硬件分界线，体现了冯.诺依曼结构核心概念
  - 以指令为中心的执行模式
  - 存算分离执行模式
- 软硬件去耦合 / 软硬件独立发展是IT至今繁荣的体系结构基础
- 兼容性考虑是计算机体系结构发展的关键因素之一