

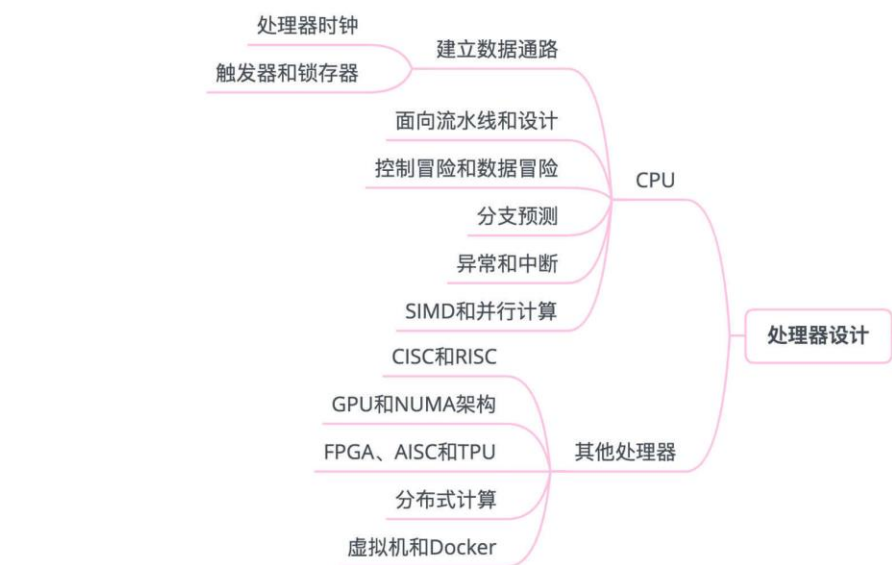
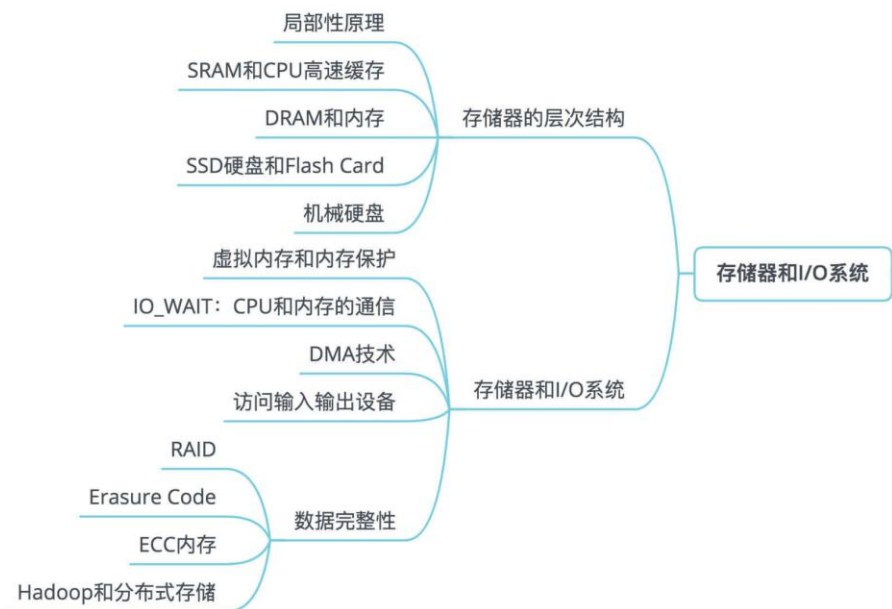
Welcome

数据科学与大数据技术

计算机系统基础

上海体育学院经济管理学院

Wu Ying



计算机组成原理知识地图

计算机的基本组成



计算机的指令和运算



章号	内容	课程				
		①	②	③	④	⑤
1	计算机系统概述	√	√	√	√	√
2	数据的机器级表示与处理	√	√	√	√	√
3	程序的转换及机器级表示	√	√	√		√
4	程序的链接	√	√			√
5	程序的执行	√		√	√	√
6	层次结构存储系统	√	√	√	√	
7	异常控制流	√	√			
8	I/O 操作的实现	√	√			
附录 A	数字逻辑电路基础	√	√	√	√	√

120学时

80学时

60-80学时

<64 学时



bilibili | 搜索

crash course

搜索

综合 视频 99+ 番剧 0 影视 0 直播 0 专栏 80 话题 0 用户 2

综合排序

最多点击

最新发布

最多弹幕

最多收藏



全部时长

10分钟以下

10-30分钟

30-60分钟

60分钟以上

全部分区

动画

番剧

国创

音乐

舞蹈

游戏

知识

科技

运动

汽车

生活

美食

动物园

鬼畜

时尚

资讯

娱乐

影视

纪录片

电影

电视剧

收起 ^



【计算机科学速成课】[40集全/精校] - Crash Course

189.4万 2018-03-29

CrashCourse字幕组



【英语口语听力】Crash Course·文学 (全四季)

7.2万 2020-02-20

YouTube口语精选



Crash Course10分钟速成经济学

15.5万 2018-07-30

每日学习视频搬运工



[合集]10分钟速成课：社会学[Crash Course

21.4万 2017-03-04

ZireHao



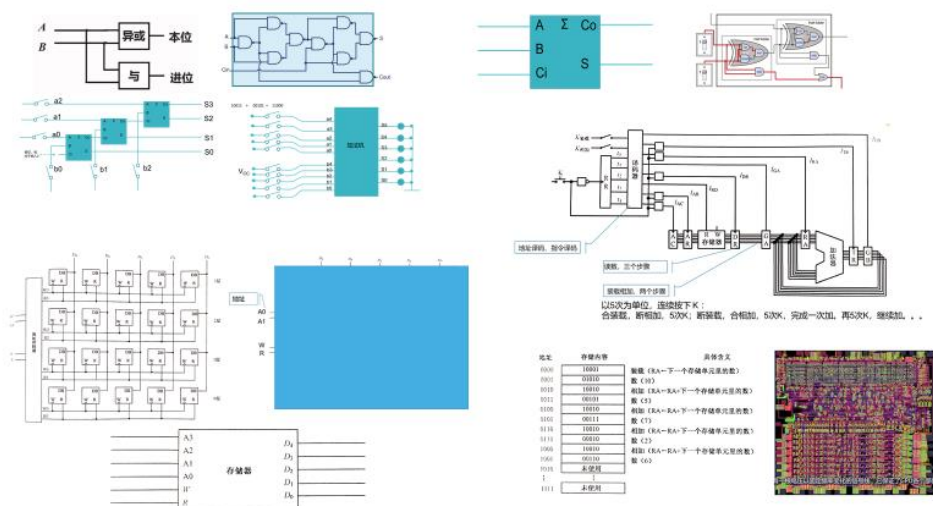
#十分钟速成课 - 合集 - 文学 (1-4季全) #Crash

11.6万 2019-10-30

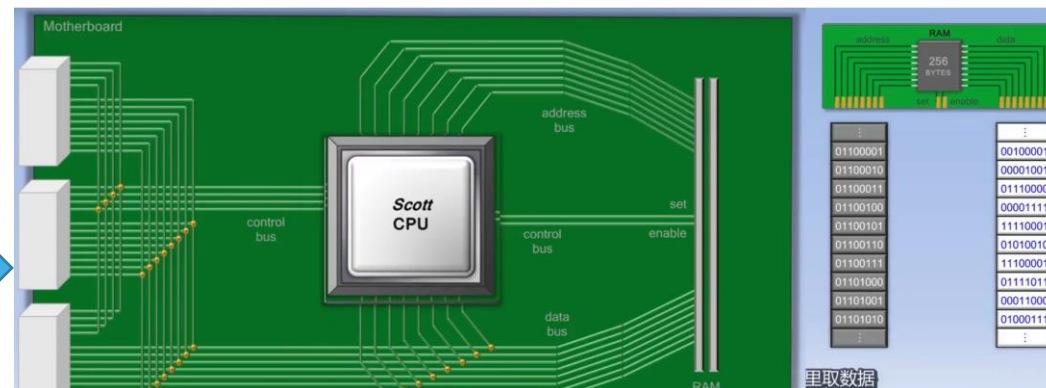
岚_0v0

CPU小结10'





抽象



抽象：

隐藏底层的细节和复杂性，为另一个层面提供新的能力

回顾要点

理解计算机

1. 个人计算机的硬件组成
2. 存储程序 原理
3. 可编程
4. 冯·诺依曼体系结构
5. 总线
6. 0 - 1 序列 指挥电路动作
7. 高级语言、汇编语言、机器码
8. 计算机发展阶段、特点以及应用
9. 总线概述、总线分类、系统总线分类、总线仲裁
10. 位、字节、地址、存储单元、地址总线位数与寻址范围

理解运算

1. 电路怎样实现运算，与或非逻辑运算
2. 传统逻辑、布尔代数、香农开关
3. CPU的演化
4. 与主存一起完成自动加法计算

深入 CPU 和 主存

1. 一条指令的执行过程
2. 程序，多条指令的连续执行
3. 冯·诺依曼机的基本工作原理
4. 指令和机器码（指令的分类、格式）
5. 模型机
6. 指令周期（Instruction Cycle）、机器周期（CPU/Machine Cycle）、时钟周期（Clock Cycle）
7. 微操作
8. 抽象

超星练习


计算机系统基础 – 指令集、计算机系统的抽象层次

本节课主要内容

01. 回顾要点
02. 指令集、指令集体系结构
03. 计算机系统的抽象层次
04. 软件分类



指令集 (指令系统)



Instruction Set

LOAD a number from RAM into the CPU

ADD two numbers together

STORE a number from the CPU back out to RAM

COMPARE one number with another

JUMP IF *Condition* to another address in RAM

JUMP to another address in RAM

OUTPUT to a device such as a monitor

INPUT from a device such as a keyboard

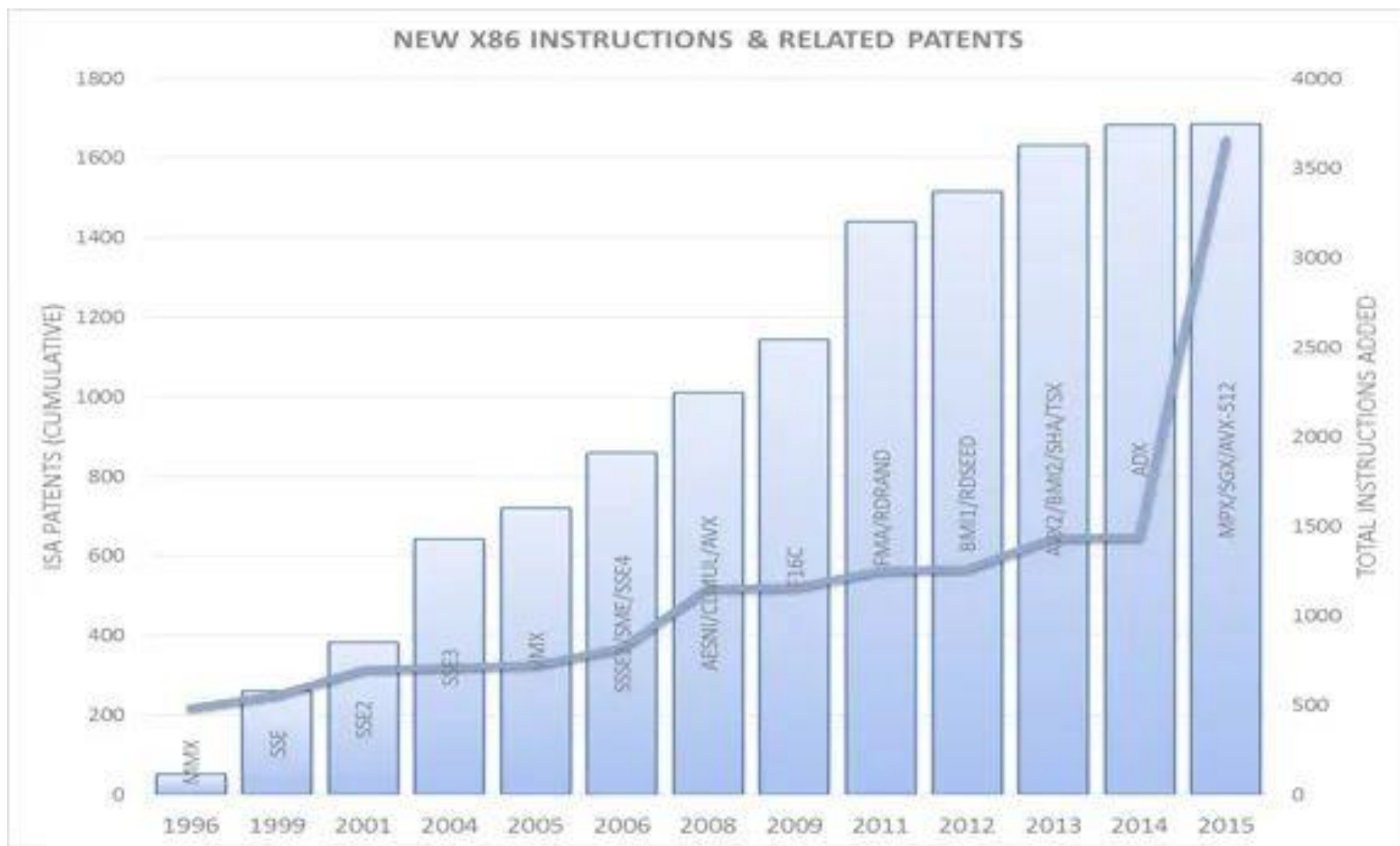
...	...
01100001	LOAD
01100010	9
01100011	IN
01100100	Keyboard
01100101	COMPARE
01100110	JUMP IF =
01100111	10100001
01101000	OUT
01101001	Monitor
01101010	"G"
...	...

指令集：CPU所能识别的全部指令的集合，包括指令格式、寻址方式和数据形式

每一种新型的CPU在设计时就规定了一系列与其硬件电路相配合的指令系统

常见的指令集有：Intel的x86, EM64T, MMX, SSE, SSE2, SSE3, SSSE3 (Super SSE3), SSE4A, SSE4.1, SSE4.2, AVX, AVX2, AVX-512, VMX等指令集；AMD的x86, x86-64, 3D-Now指令集

指令集 (指令系统)

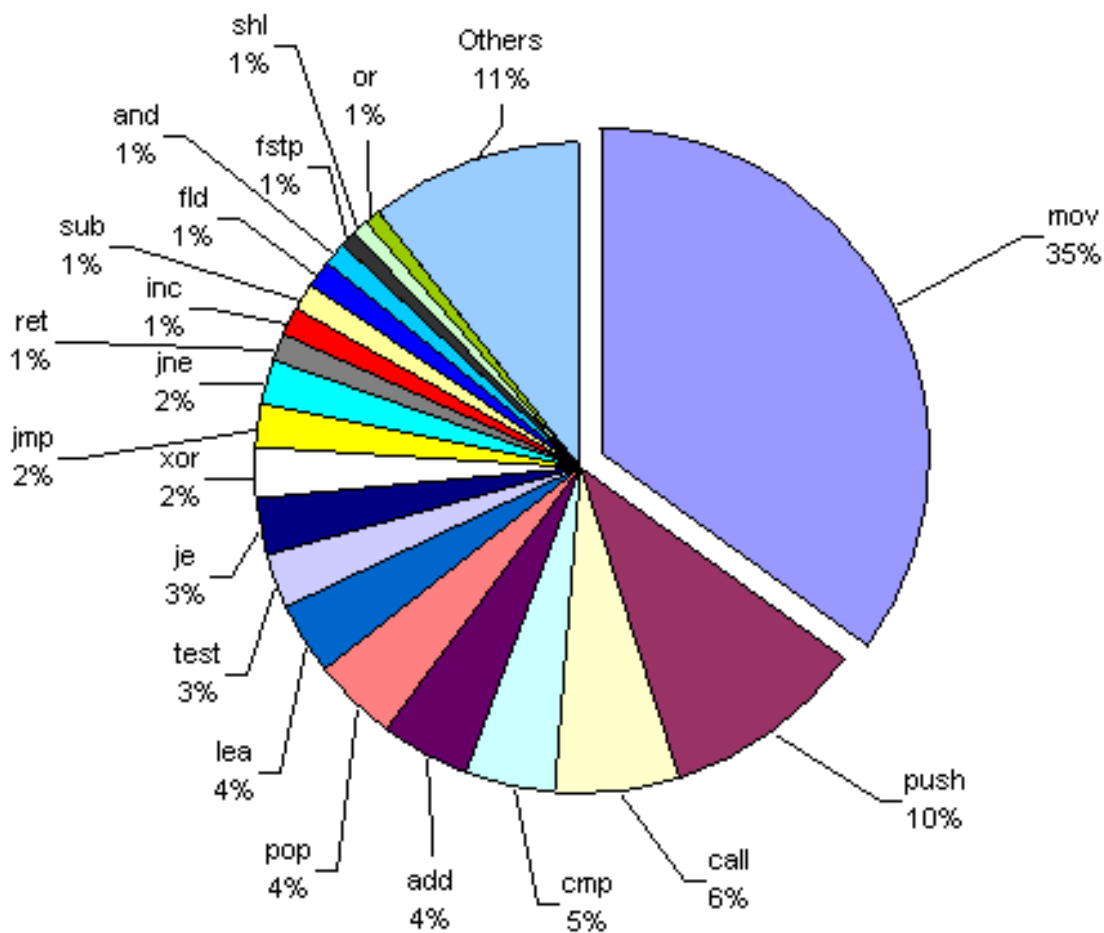


Intel官方给出的指令个数及相关专利的发展趋势，从中可以看出，在40年的发展历史中，由不足200条指令到今天的超过1600条指令。

指令分类

- ① 算术类指令
- ② 数据传输类指令
- ③ 逻辑类指令
- ④ 条件分支类指令
- ⑤ 无条件跳转指令

Top 20 instructions of x86 architecture



Instruction Set

LOAD a number from RAM into the CPU

ADD two numbers together

STORE a number from the CPU back out to RAM

COMPARE one number with another

JUMP IF *Condition* to another address in RAM

JUMP to another address in RAM

OUTPUT to a device such as a monitor

INPUT from a device such as a keyboard

...	...
01100001	LOAD
01100010	9
01100011	IN
01100100	Keyboard
01100101	COMPARE
01100110	JUMP IF =
01100111	10100001
01101000	OUT
01101001	Monitor
01101010	"G"
...	...

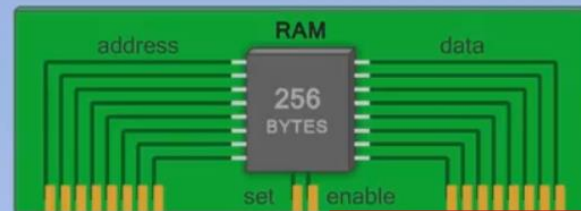


Scott
CPU

control
bus

control
bus

data
bus



...
01100001
01100010
01100011
01100100
01100101
01100110
01100111
01101000
01101001
01101010
...

...
00100001
00001001
01110000
00001111
11110001
01010010
11100001
01111011
00011000
01000111
...

CPU在设计时就规定了一系列与其硬件电路相配合的指令系统

硬件电路，硬件

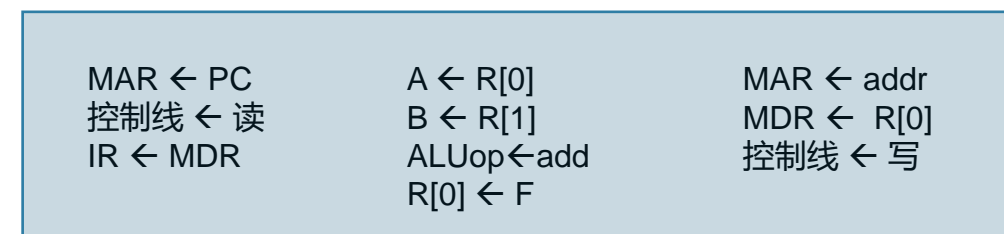
接口部分：指令系统 ISA

程序，指令组合，软件

...
LOAD
9
IN
Keyboard
COMPARE
JUMP IF =
10100001
OUT
Monitor
"G"
...

回顾一下关于指令的脉络

每一个微操作命令都对应一个硬件逻辑电路（组合逻辑型控制单元）



抽象

LOAD R0,[6] ADD R0,R1 STORE [7],R0

主存地址	指令符号表示
0000	指令 I1 : LOAD R0, [6]
0001	指令 I2 : MOV R1, R0
0010	指令 I3 : LOAD R0, [5]
0011	指令 I4 : ADD R0, R1
0100	指令 I5 : STORE [7], R0
0101	2
0110	3
0111	
1000	

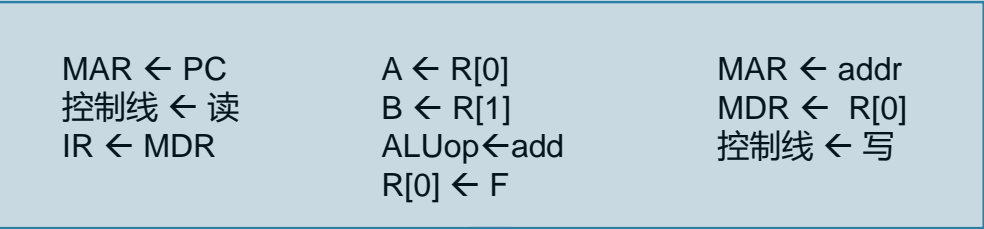
机器码表示
1110 0110
0000 0100
1110 0101
0001 0001
1111 0111

格式	4 位	2 位	2 位	功能说明
R 型	op	rt	rs	$R[rt] \leftarrow R[rt] \text{ op } R[rs]$ 或 $R[rt] \leftarrow R[rs]$
M 型	op	addr		$R[0] \leftarrow M[addr]$ 或 $M[addr] \leftarrow R[0]$

书P5 图1.2 定长指令格式

指令的格式?

每一个微操作命令都对应一个硬件逻辑电路
(组合逻辑型控制单元)



抽象



LOAD R0,[6] ADD R0,R1 STORE [7],R0

STA M:

- [1] AD(IR) --> MAR: 控制信号C₅有效, IR指令中的地址码字段送给MAR。
- [2] 1 --> W: 使能寄存器被写状态。
- [3] ACC --> MDR: 控制信号C₁₂有效, 将要写入存储器中M地址的数据写入MDR。
- [4] MDR --> MM(MAR): C₁,C₂控制信号有效, MDR中的数据写往寄存器M的M地址之上。

ADD M:

- [1] AD(IR) --> MAR: 控制信号C₅有效, IR指令中的地址码字段送给MAR。
- [2] 1 --> R: 使能寄存器被读状态。
- [3] MM(MAR) --> MDR : C₁,C₂控制信号有效, 存储器之上的MAR地址的内容通过数据总线传输到MDR中。
- [4] (ACC) + (MDR) --> ACC: 控制信号C₇, C₆有效, 将读到的数据与ACC中的内容送到ALC相加, C₈控制信号有效, 将ALU相加的结果送入ACC中。

表 10.1 操作时间表

工作周 期标记	节拍	状态条件	微操作命令信号	CLA	COM	SHR	CSL	STP	ADD	STA	LDA	JMP	BAN
FE (取指)	T ₀		PC→MAR	1	1	1	1	1	1	1	1	1	1
			1→R	1	1	1	1	1	1	1	1	1	1
	T ₁		M(MAR)→MDR	1	1	1	1	1	1	1	1	1	1
			(PC) + 1→PC	1	1	1	1	1	1	1	1	1	1
	T ₂		MDR→IR	1	1	1	1	1	1	1	1	1	1
			OP(IR)→ID	1	1	1	1	1	1	1	1	1	1
		I	I→IND						1	1	1	1	1
		\overline{I}	I→EX	1	1	1	1	1	1	1	1	1	1
IND (间接 寻址)	T ₀		Ad(IR)→MAR						1	1	1	1	1
			I→R						1	1	1	1	1
	T ₁		M(MAR)→MDR						1	1	1	1	1
			MDR→Ad(IR)						1	1	1	1	1
		\overline{IND}	I→EX						1	1	1	1	1
	T ₀		Ad(IR)→MAR						1	1	1		
			1→R						1		1		
			1→W							1			
			M(MAR)→MDR						1		1		



指令集和指令集体系结构 (ISA)

指令格式——规定了 指令 在硬件上对应的二进制比特流

格式	4 位	2 位	2 位	功能说明
R 型	op	rt	rs	$R[rt] \leftarrow R[rt] \text{ op } R[rs]$ 或 $R[rt] \leftarrow R[rs]$
M 型	op	addr		$R[0] \leftarrow M[addr]$ 或 $M[addr] \leftarrow R[0]$

主存地址	指令符号表示	机器码表示
0000	指令 I1 : LOAD R0, [6]	1110 0110
0001	指令 I2 : MOV R1, R0	0000 0100
0010	指令 I3 : LOAD R0, [5]	1110 0101
0011	指令 I4 : ADD R0, R1	0001 0001
0100	指令 I5 : STORE [7], R0	1111 0111
0101	2	
0110	3	
0111		
1000		



MIPS指令集

32位指令

指令类型	6位	5位	5位	5位	5位	6位	解释
R	opcode	rs	rt	rd	shamt 位移量	funct 功能码	算术操作、逻辑操作
I	opcode	rs	rt	address/immediate 地址/立即数			数据传输、条件分支、 立即数操作
J	opcode	target address 目标地址					无条件跳转

MIPS的指令种类有：

- 1、算术运算。
- 2、逻辑运算。
- 3、数据传送。
- 4、条件转移。
- 5、无条件跳转。
- 6、特殊指令。
- 7、例外指令。
- 8、协处理器指令。
- 9、系统控制协处理器指令。

MIPS指令集，精简指令集RISC，所有指令都是32位，指令格式简单，X86指令长度不是固定的



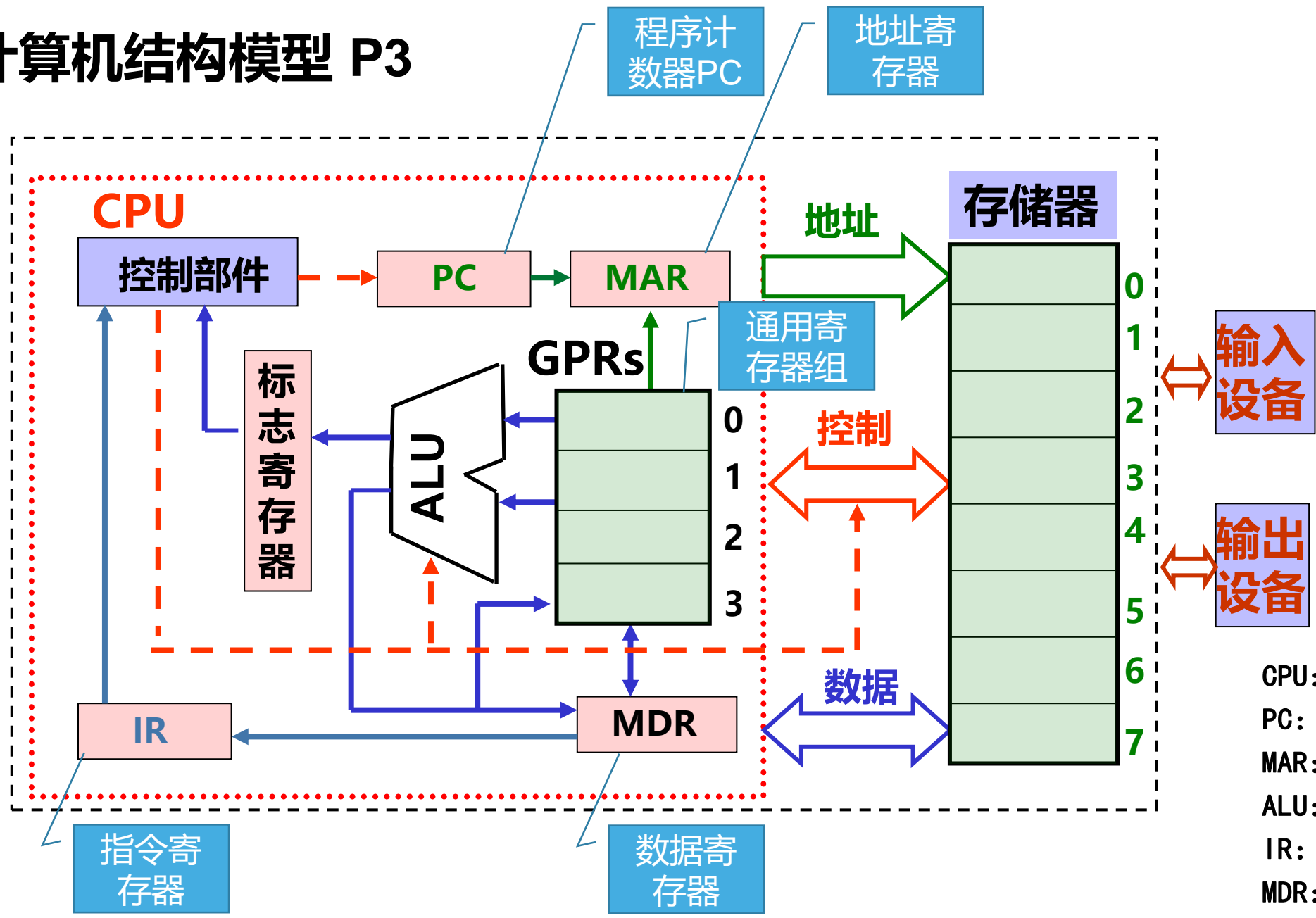
指令集和指令集体系结构 (ISA)

【P13 下半面】 【P89 3.1.2 第一段】

ISA指 Instruction Set Architecture，即指令集体系结构，有时简称为指令系统，是一种规约 (Specification)，**规定了如何使用硬件**，内容包括：

- ① 可执行的指令的集合，包括**指令格式、操作种类**以及每种操作对应的操作数的相应规定；
- ② 指令可以接受的**操作数的类型**；
- ③ 操作数或其地址所能存放的寄存器组的结构，包括每个**寄存器的名称、编号、长度和用途**；
- ④ 操作数所能存放的**存储空间的大小和编址方式**；
- ⑤ 操作数在存储空间存放时按照**大端还是小端方式存放**；
- ⑥ 指令获取操作数的方式，即**寻址方式**；
- ⑦ 指令执行过程的控制方式，包括**程序计数器 (PC)、条件码定义**等

现代计算机结构模型 P3



书例P5 模型机假设

- ① 字长：8位
- ② 4个通用寄存器：r0 – r3，编号为 0 - 3
- ③ 有16个存储单元，编号为 0 – 15
- ④ 每个主存单元和CPU中的ALU、通用寄存器组、IR、MDR的宽度都是 8 位，PC 和 MAR 4 位
- ⑤ 连接CPU和主存的总线中，地址线 4 位，数据线 8 位，若干位控制线（包括读写命令线）
- ⑥ 该模型机采用 8 位定长指令，即每条指令长度为 8 bit

Instruction Set

LOAD a number from RAM into the CPU

ADD two numbers together

STORE a number from the CPU back out to RAM

COMPARE one number with another

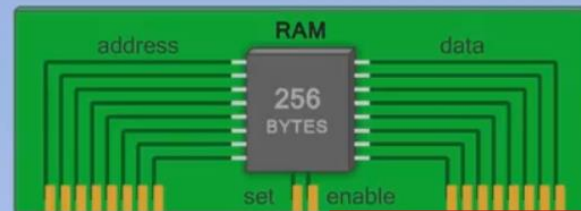
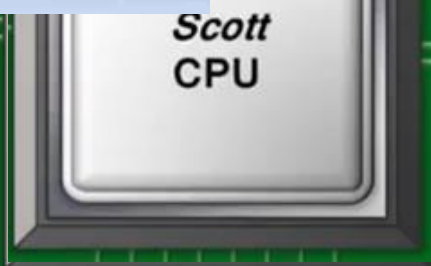
JUMP IF *Condition* to another address in RAM

JUMP to another address in RAM

OUTPUT to a device such as a monitor

INPUT from a device such as a keyboard

...	...
01100001	LOAD
01100010	9
01100011	IN
01100100	Keyboard
01100101	COMPARE
01100110	JUMP IF =
01100111	10100001
01101000	OUT
01101001	Monitor
01101010	"G"
...	...



...
01100001
01100010
01100011
01100100
01100101
01100110
01100111
01101000
01101001
01101010
...

...
00100001
00001001
01110000
00001111
11110001
01010010
11100001
01111011
00011000
01000111
...

...
LOAD
9
IN
Keyboard
COMPARE
JUMP IF =
10100001
OUT
Monitor
"G"
...

CPU在设计时就规定了一系列与其硬件电路相配合的指令系统

硬件电路, 硬件

接口部分: 指令系统 ISA

程序, 指令组合, 软件

应用程序

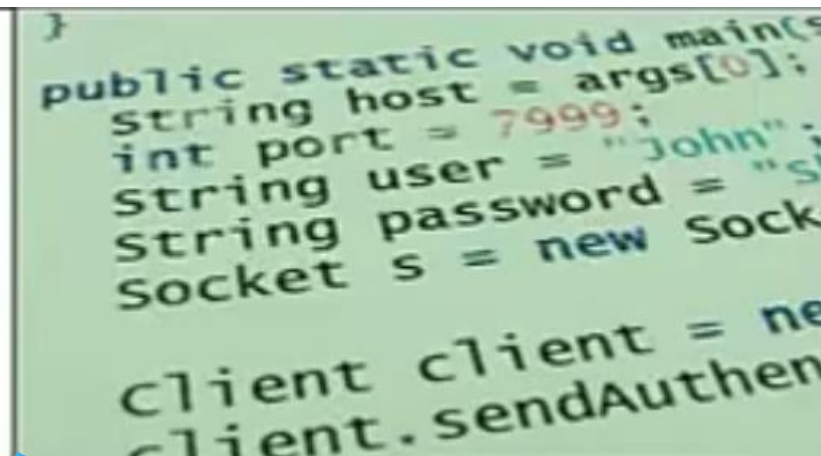
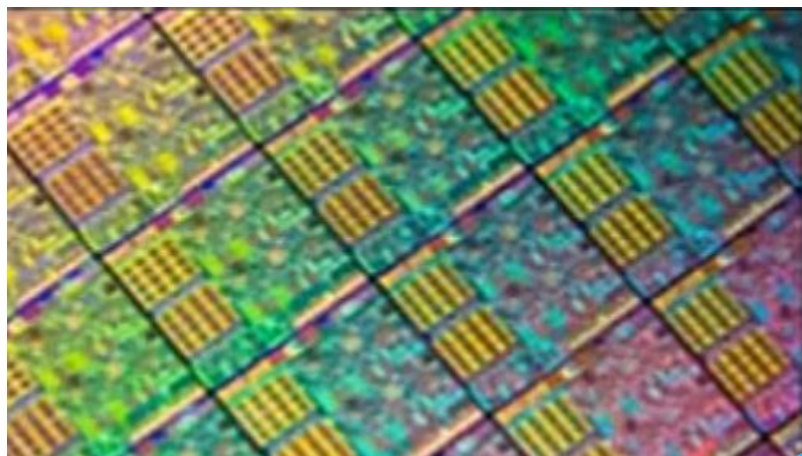
指令集体系结构

计算机硬件

指令集和指令集体系结构 (ISA)

hardware

software



$Z = X + Y$

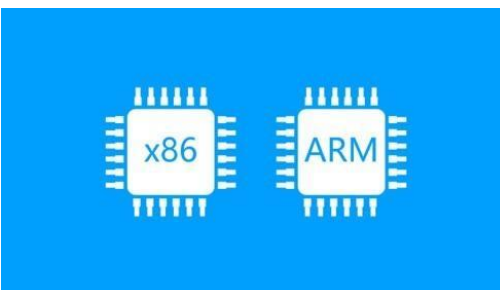
指令集架构

复杂指令集架构 CISC

Complex Instruction Set Computers

INTEL的 IA-32

AMD AMD64



精简指令集架构 RISC

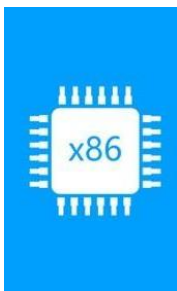
Reduced Instruction Set Computers

ARM

MIPS

【P94-95 3.2 第一段】

- **x86**是Intel开发的一类处理器体系结构的泛称
 - 包括 Intel 8086、80286、i386和i486等，因此其架构被称为 “x86”
 - 由于数字并不能作为注册商标，因此，后来使用了可注册的名称，如Pentium、PentiumPro、Core 2、Core i7等
 - 现在Intel把32位x86架构的名称x86-32改称为**IA-32**
- 由AMD首先提出了一个**兼容IA-32指令集的64位版本**
 - 扩充了指令及寄存器长度和个数等，更新了参数传送方式
 - AMD称其为AMD64，Intel称其为Intel64（不同于IA-64。Intel64,64位超长指令字，不兼容IA-32）
 - 命名为 “**x86-64**”，有时也**简称为x64（兼容IA-32）**

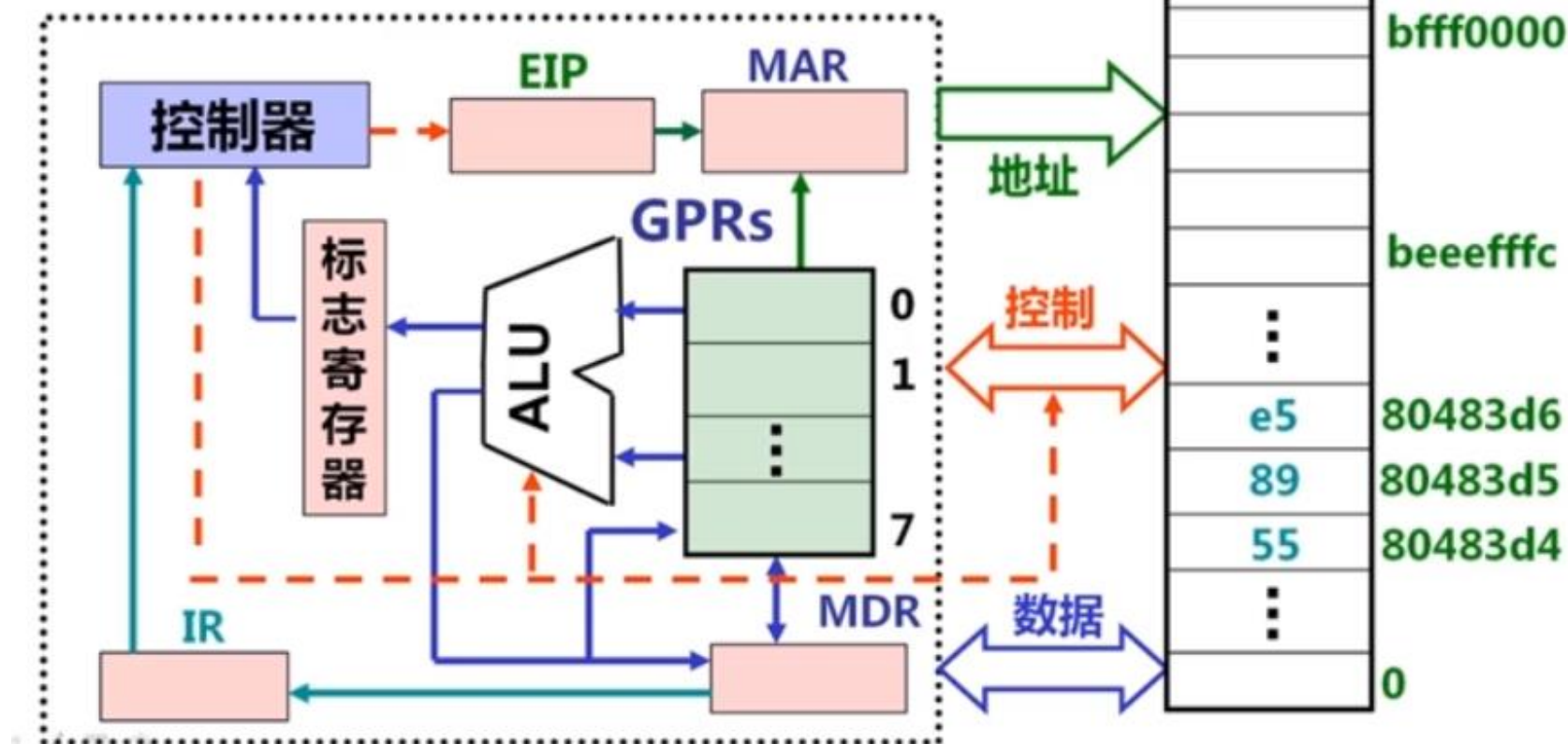


复杂指令集架构 CISC

第三章主要介绍IA-32

IA-32 模型机

8个GPR (0~7) , 一个EFLAGS, PC为EIP
可寻址空间4GB (编号为0~0xFFFFFFFF)
指令格式变长, 操作码变长, 指令由若干字段
(OP、Mod、SIB等) 组成



第三章主要介绍IA-32

IA-32支持的数据类型及格式 *【P95-97】*

ISA-32从ISA-16发展而来，
16位——字

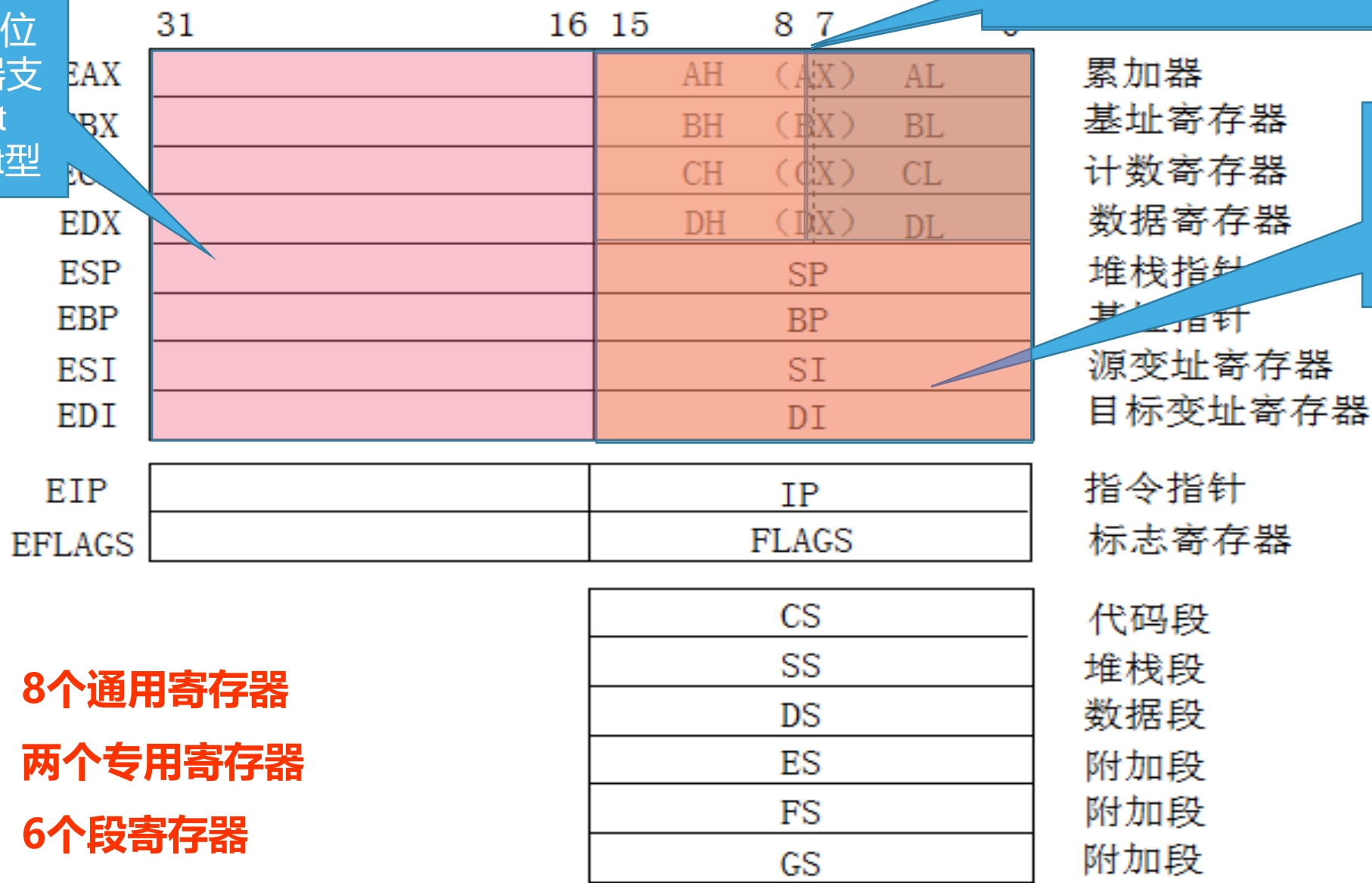
C 语言声明	Intel 操作数类型	汇编指令长度后缀	存储长度 (位)
(unsigned) char	整数 / 字节	b	8
(unsigned) short	整数 / 字	w	16
(unsigned) int	整数 / 双字	l	32
(unsigned) long int	整数 / 双字	l	32
(unsigned) long long int	-	-	2×32
char *	整数 / 双字	l	32
float	单精度浮点数	s	32
double	双精度浮点数	l	64
long double	扩展精度浮点数	t	80 / 96

long double 实际长度为80
位，分配了12B

IA-32的定点整数寄存器的组织【P95-97】

8个32位寄存器支持int longint型

8个8位寄存器支持char型



8个16位寄存器支持short型

8个通用寄存器
两个专用寄存器
6个段寄存器

具体到每一种型号的芯片，核心的构建可能不同

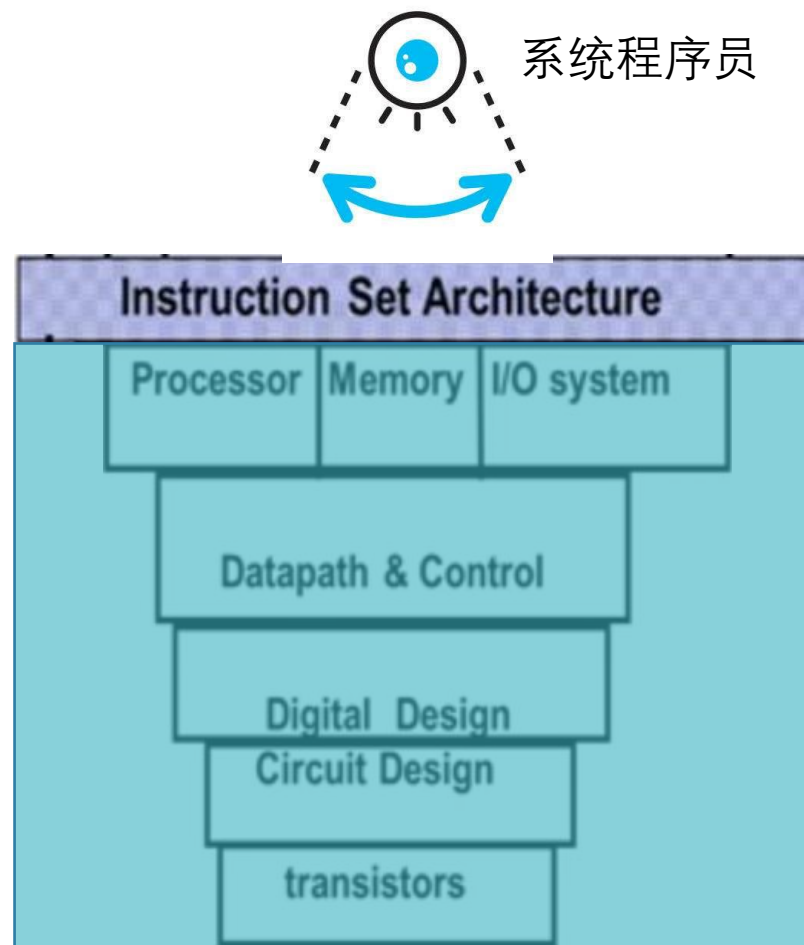
第三章主要介绍IA-32，非大纲内容，详见袁老师慕课

指令集体系结构 ISA

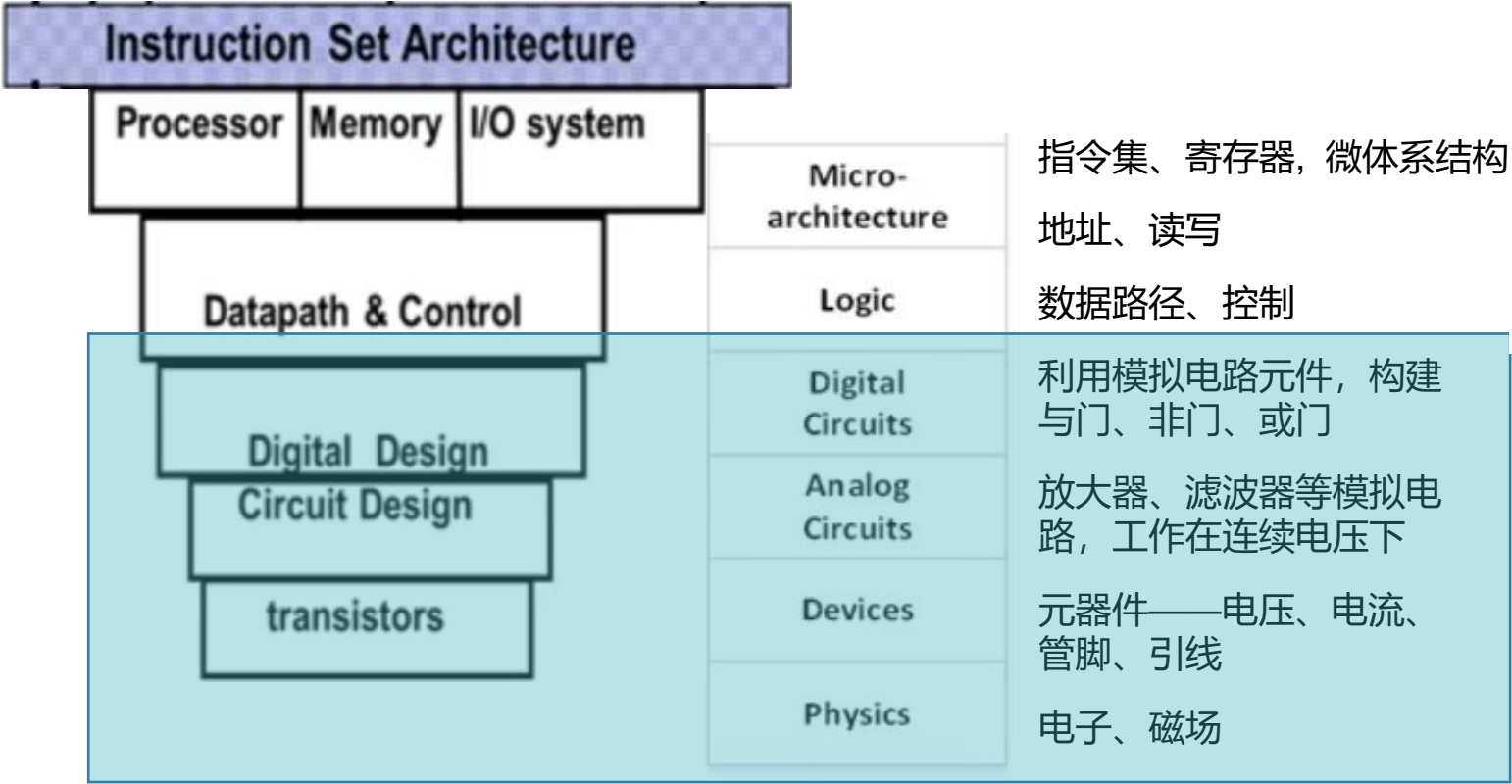
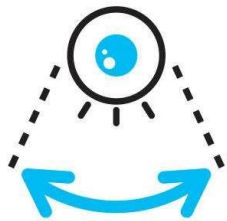
计算机硬件之上的抽象层，对使用硬件的软件屏蔽了底层的实现细节，将物理上的计算机硬件抽象成为一个逻辑上的虚拟计算机，称为**机器语言级虚拟机** 【P89 3.1.2 第一段】

系统程序员所看到的机器，是配置了指令系统的机器——**机器语言、机器、虚拟机**

【P18 1.3末尾】

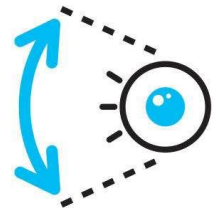


系统程序员

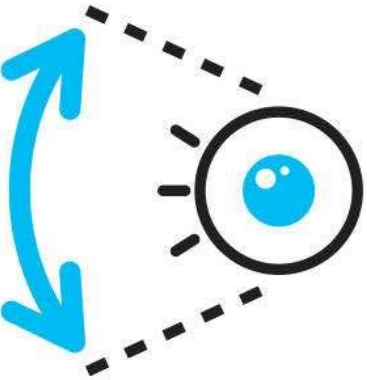


物理构成角度

指令集和指令集体系结构 (ISA)



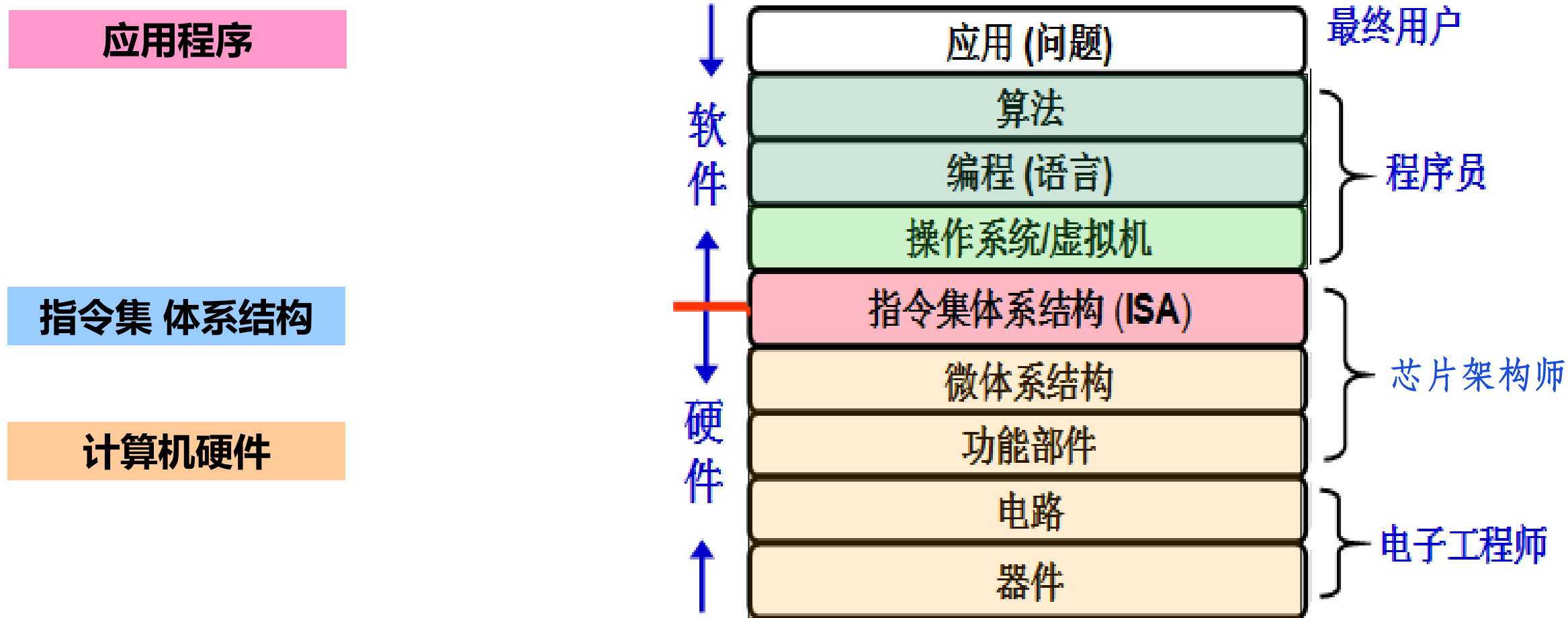
芯片系统架构师



电子工程师

微电子学

1.3.3 计算机系统的不同用户



计算机系统基础 – 指令集、计算机系统的抽象层次

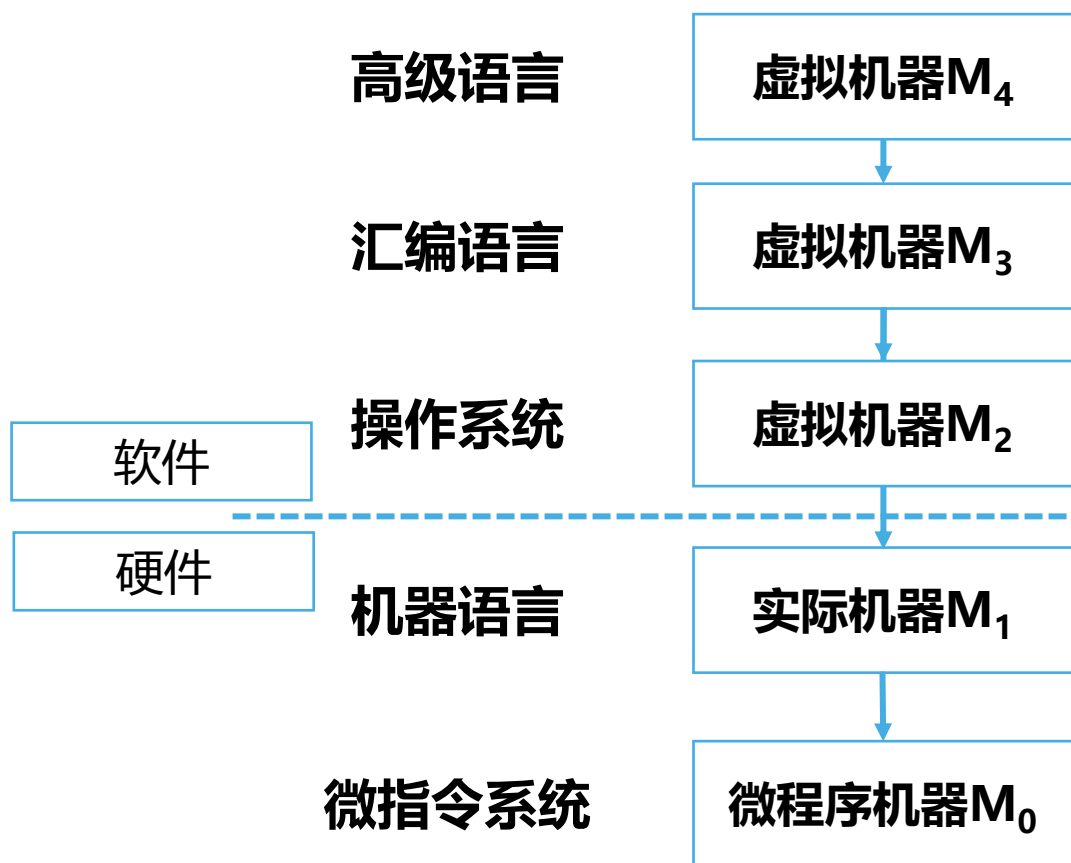
本节课主要内容

01. 回顾要点
02. 指令集、指令集体系结构
03. 计算机系统的抽象层次
04. 软件分类



计算机系统的抽象层次结构

从软件工程师的角度



用编译程序翻译成汇编语言

用汇编程序翻译成机器语言

用机器语言解释操作系统

由微指令解释机器指令

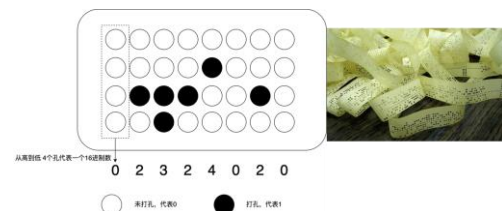
由硬件直接执行微指令

```
tmp = a[i];
a[i] = a[i+1];
a[i+1] = tmp;
```

```
lw $8, 0($2)
lw $9, 4($2)
sw $9, 0($2)
sw $8, 4($2)
```

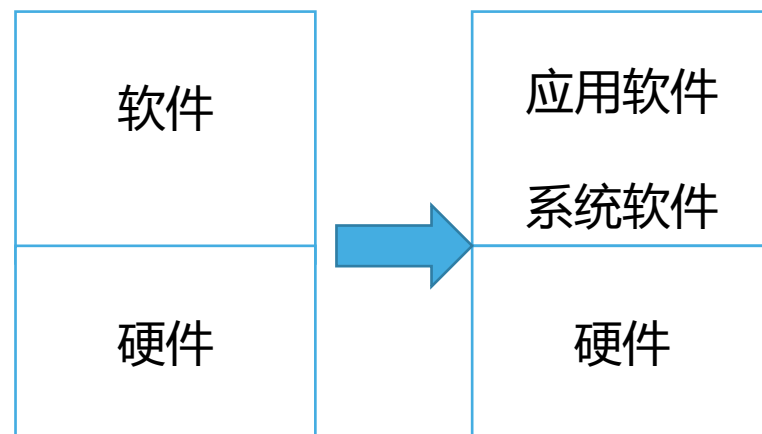
```
100011 00010 01000 0000000000000000
100011 00010 01001 0000000000000100
101011 00010 01001 0000000000000000
101011 00010 01000 0000000000000100
```

```
..., EXTop=1,ALUSelA=1,ALUSelB=11,ALUop=add,
IorD=1,Read,MemtoReg=1,RegWr=1, ...
```

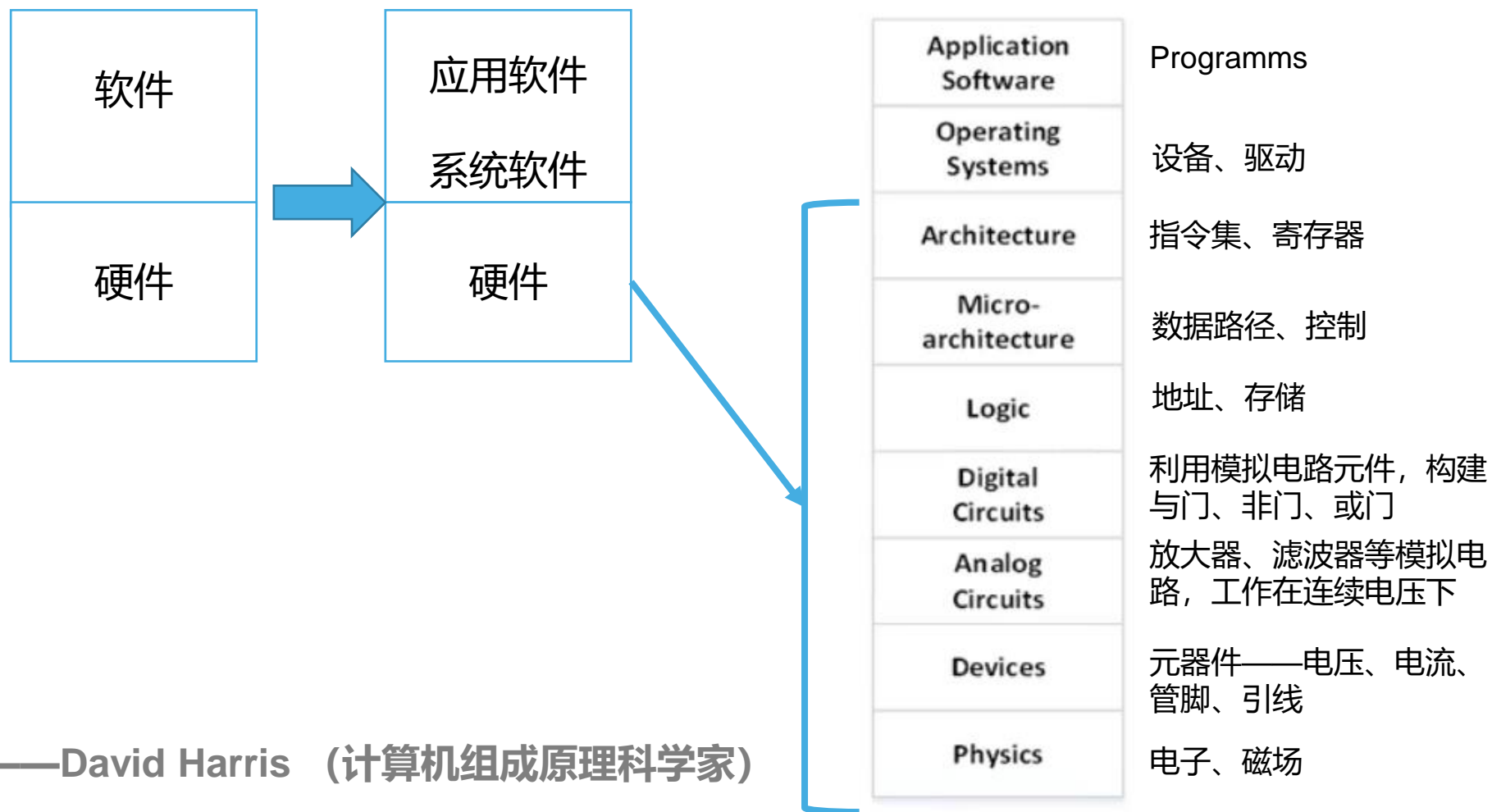


计算机系统的抽象层次结构

隐藏系统当中不重要的细节



计算机系统的简单层次结构



- 抽象 ——David Harris (计算机组成原理科学家)

隐藏系统当中不重要的细节

从物理构成角度

软件系统

代码	计算机软件类别	说明	代码	计算机软件类别
10000	系统软件		60000	应用软件
11000	操作系统	包括实时、分时、颁布式、智能等操作系统	61000	科学和工程计算软件
			61500	文字处理软件
12000	系统实用程序		62000	数据处理软件
13000	系统扩充程序	包括操作系统的扩充、汉化	62500	图形软件
14000	网络系统软件		63000	图象处理软件
19900	其它系统软件		64000	应用数据库软件
30000	支持软件		65000	事务管理软件
31000	软件开发工具		65500	辅助类软件
32000	软件评测工具		66000	控制类软件
33000	界面工具		66500	智能软件
34000	转换工具		67000	仿真软件
35000	软件管理工具		67500	网络应用软件
36000	语言处理程序		68000	安全与保密软件
37000	数据库管理系统		68500	社会公益服务软件
38000	网络支持软件		69000	游戏软件
39900	其它支持软件		69900	其它应用软件

- **系统软件（管理整个计算机系统）**

操作系统、网络系统、编译系统等

- **支持软件（服务性的系统软件）**

开发工具、界面工具、语言处理程序、数据库管理系统、网络支持软件等

- **应用软件(按任务需要编制的各类程序)**

文字处理软件、图像处理软件、辅助设计软件、游戏软件等

系统软件 – 操作系统



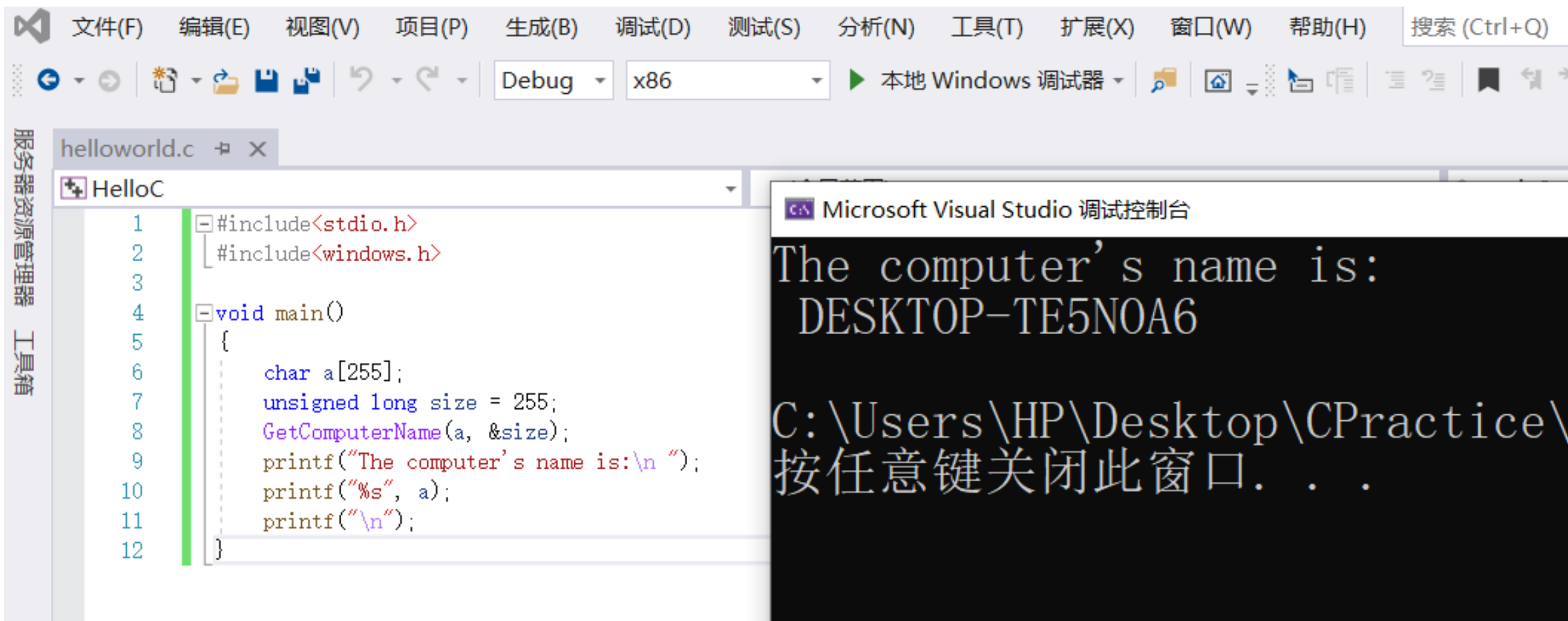
迷糊老师

编程加油站系列

UP主M的宝石被抢了



说一群劫匪抢了UP主M的一块宝石



计算机名、域和工作组设置

计算机名: DESKTOP-TE5NOA6

计算机全名: DESKTOP-TE5NOA6

计算机描述:

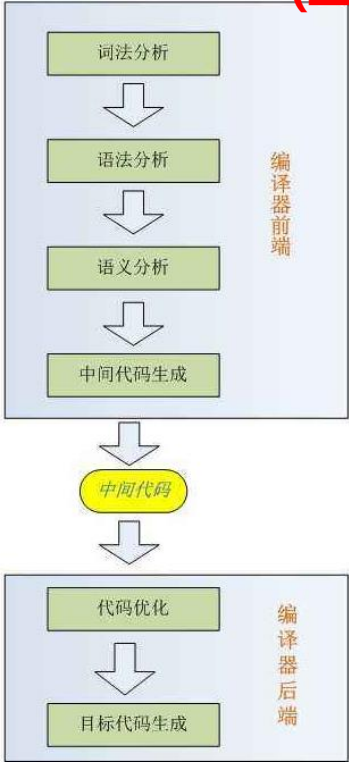
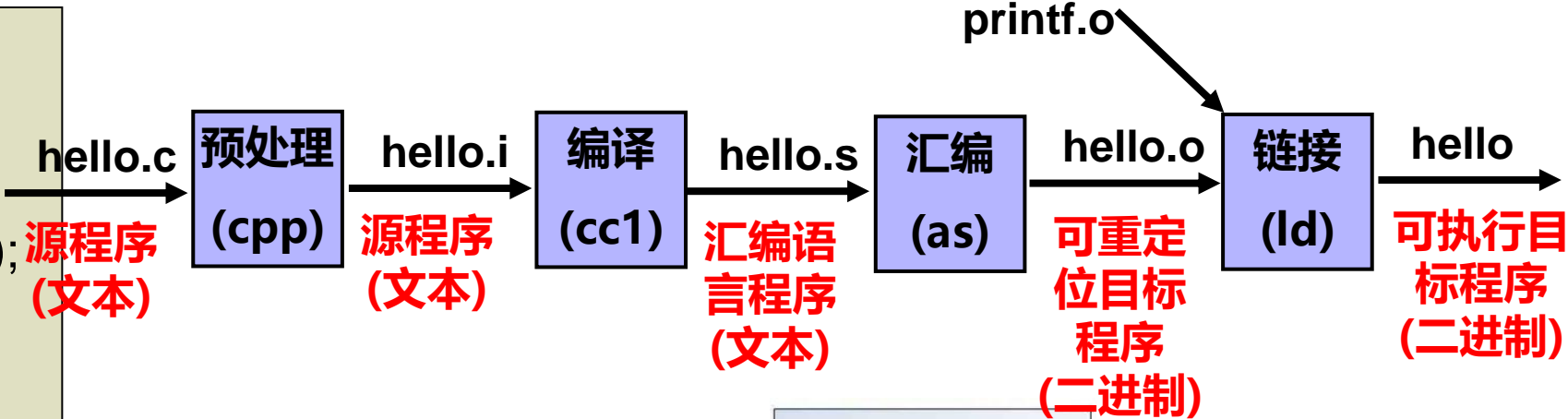
工作组: WORKGROUP

Windows 激活

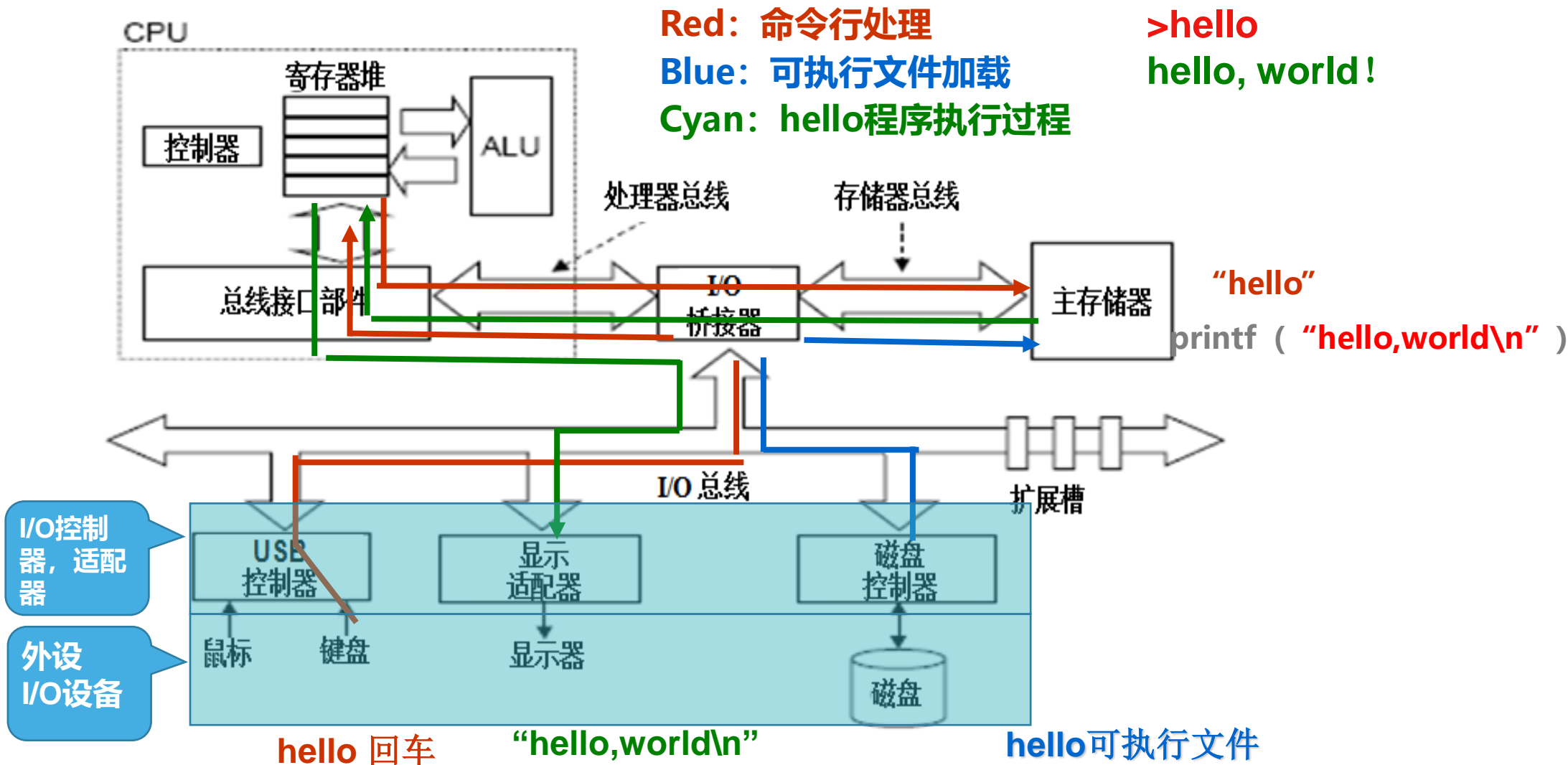
系统软件 – 编译

```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
    return 0;
}
```



可执行文件的启动和执行



所有过程在CPU执行指令所产生的控制信号的作用下进行
所有的数据流动都通过总线、I/O桥接器进行，传输之前缓存在存储部件中

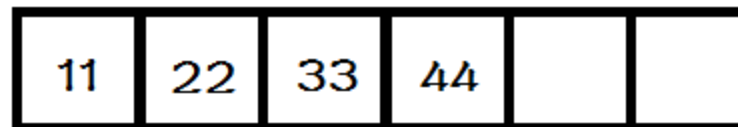
指令、数据 内存分区域存放

主存地址	指令符号表示	机器码表示
0000	指令 I1 : LOAD R0, [6]	1110 0110
0001	指令 I2 : MOV R1, R0	0000 0100
0010	指令 I3 : LOAD R0, [5]	1110 0101
0011	指令 I4 : ADD R0, R1	0001 0001
0100	指令 I5 : STORE [7], R0	1111 0111
0101	2	
0110	3	
0111		
1000		

11 22 33 44

尾端 44

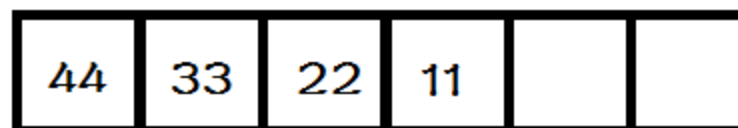
高尾端 (大端)



0 1 2 3

低地址 -----> 高地址

低尾端 (小端)



0 1 2 3

低地址 -----> 高地址

利用union特性测试大端、小端

- union的存放顺序是所有成员从低地址开始，利用该特性可测试CPU的大端、小端方式

```
int main()
{
    union NUM
    {
        int a;
        char b;
    } num;
    num.a = 0x12345678;
    if (num.b == 0x78)
        printf("小端\n");
    else
        printf("大端\n");

    printf("num.b = 0x %X\n", num.b);
    return 0;
}
```

