计算机组织与体系结构

Computer Architectures

陆俊林

北京大学本科生主干基础课

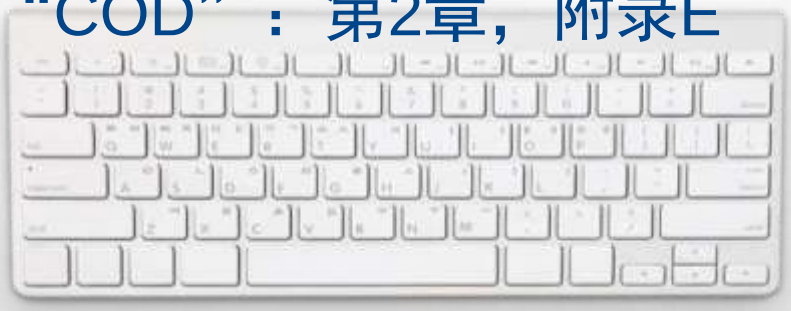
2018.3.21

第四讲 RISC和MIPS指令（1）

本讲要点

首先简介RISC兴起的历程，其次分析MIPS指令的设计原则和主要特点，然后按照指令格式分类讲解主要的MIPS指令，本讲主要分析R型和I型的典型指令，其他指令类型将在下一讲讲解。

阅读教材“COD”：第2章，附录E



主要内容

通过学习本课程
了解计算机的发展历程，理解计算机的组成原理，掌握计算机的设计方法

Ⅰ RISC的发展变迁

Ⅱ MIPS指令的主要特点

Ⅲ MIPS指令分类说明：R型

Ⅳ MIPS指令分类说明：I型

RISC的先驱，两位传奇人物

戴维·帕特森
David Patterson
1947年出生

约翰·亨尼西
John Hennessy
1953年出生

John Hennessy

- 1977年，进入斯坦福大学任职
- 1981年，领导RISC微处理器的研究小组
- 1984年，共同创立MIPS计算机系统公司
- 1989年~1999年，先后担任斯坦福大学计算机系统实验室主任、计算机系主任和工程学院院长等
- 2000年至2016年，任斯坦福大学校长
- 2018年，出任Alphabet董事长

约翰·亨尼斯
John Hennessy
1953年出生

IEEE Medal of Honor “for pioneering the RISC processor architecture and for leadership in computer engineering and higher education"

MIPS公司的商业兴衰

- 1984年，MIPS计算机系统公司成立
- 1988年，SGI公司在其计算机产品中采用MIPS处理器
- 1989年，MIPS第一次上市
- 1992年，SGI收购MIPS，更名为MIPS技术公司
- 1998年，MIPS再次上市
- 2012年，Imagination Technologies收购MIPS

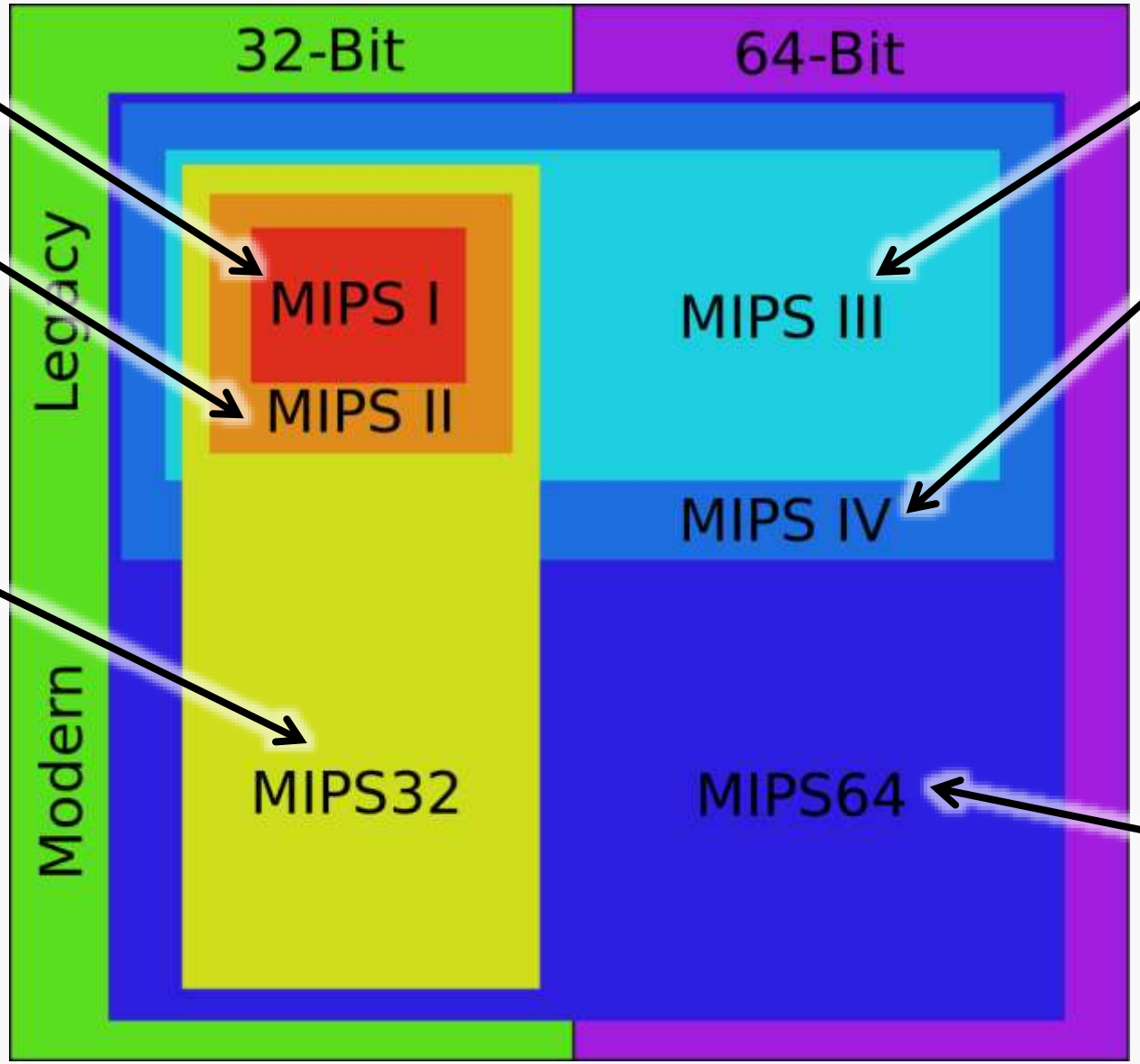
- MIPS处理器广泛应用的领域：
  - 数字电视、机顶盒、蓝光播放器、游戏机、网络设备等

MIPS指令的发展

1985年，R2000

1990年，R3000

1999年
以MIPS II为基础，
增加了MIPS
III/IV/V的部分特性



1992年，R4000
扩展到64位

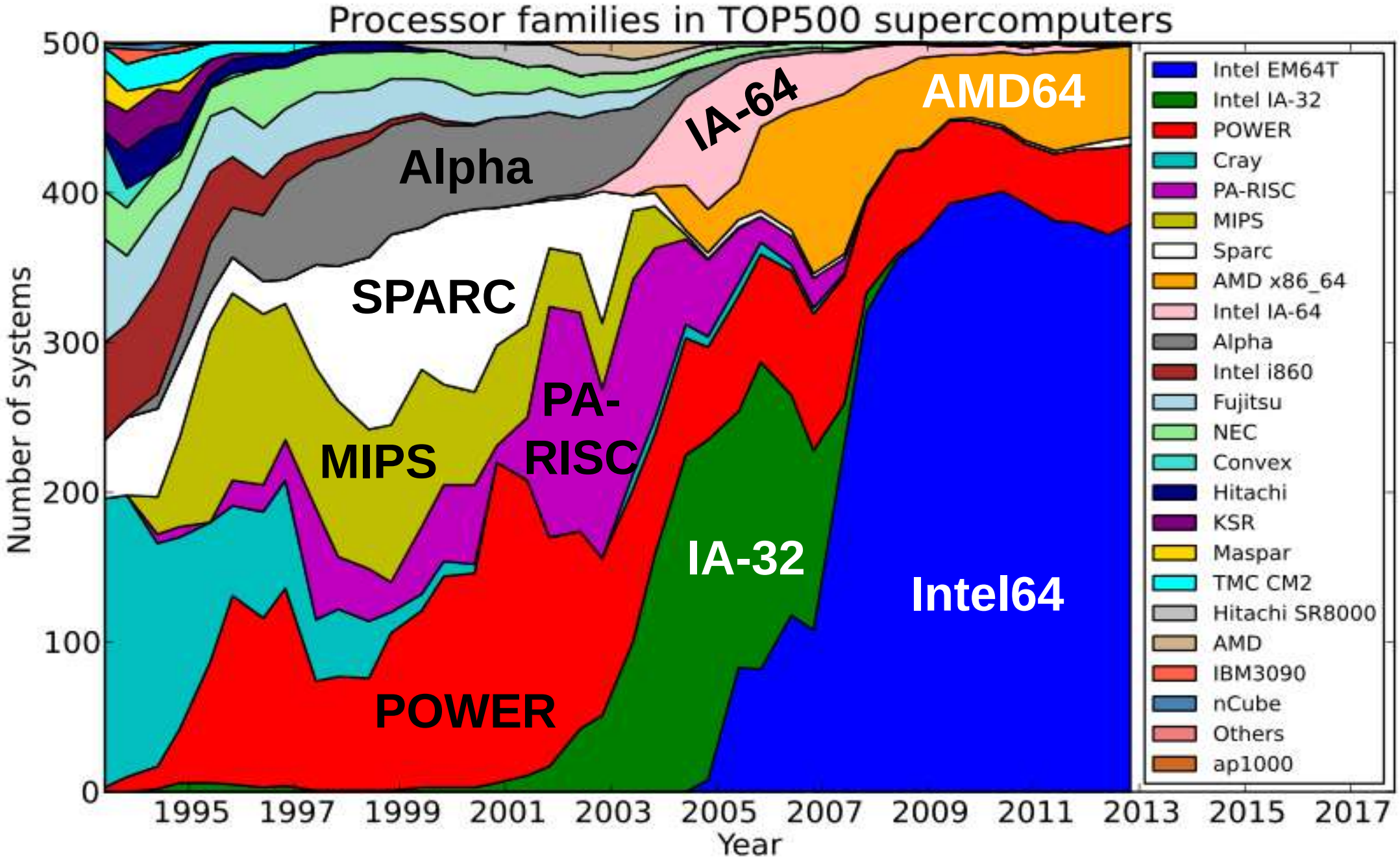
1994年，R8000

MIPS V
1996年

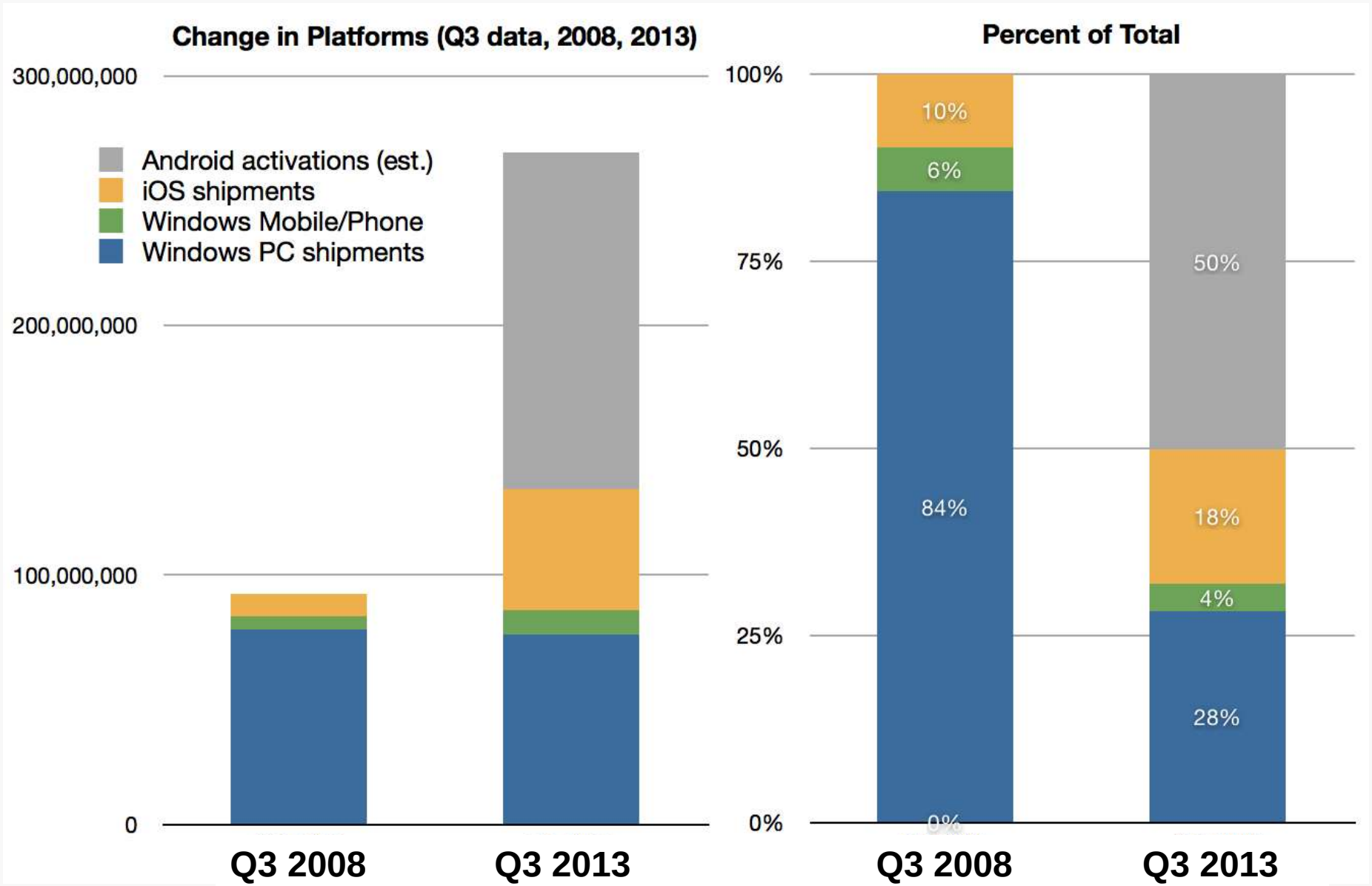
1999年
以MIPS V为基础

超级计算领域

CISC vs. RISC



个人计算领域 CISC vs. RISC



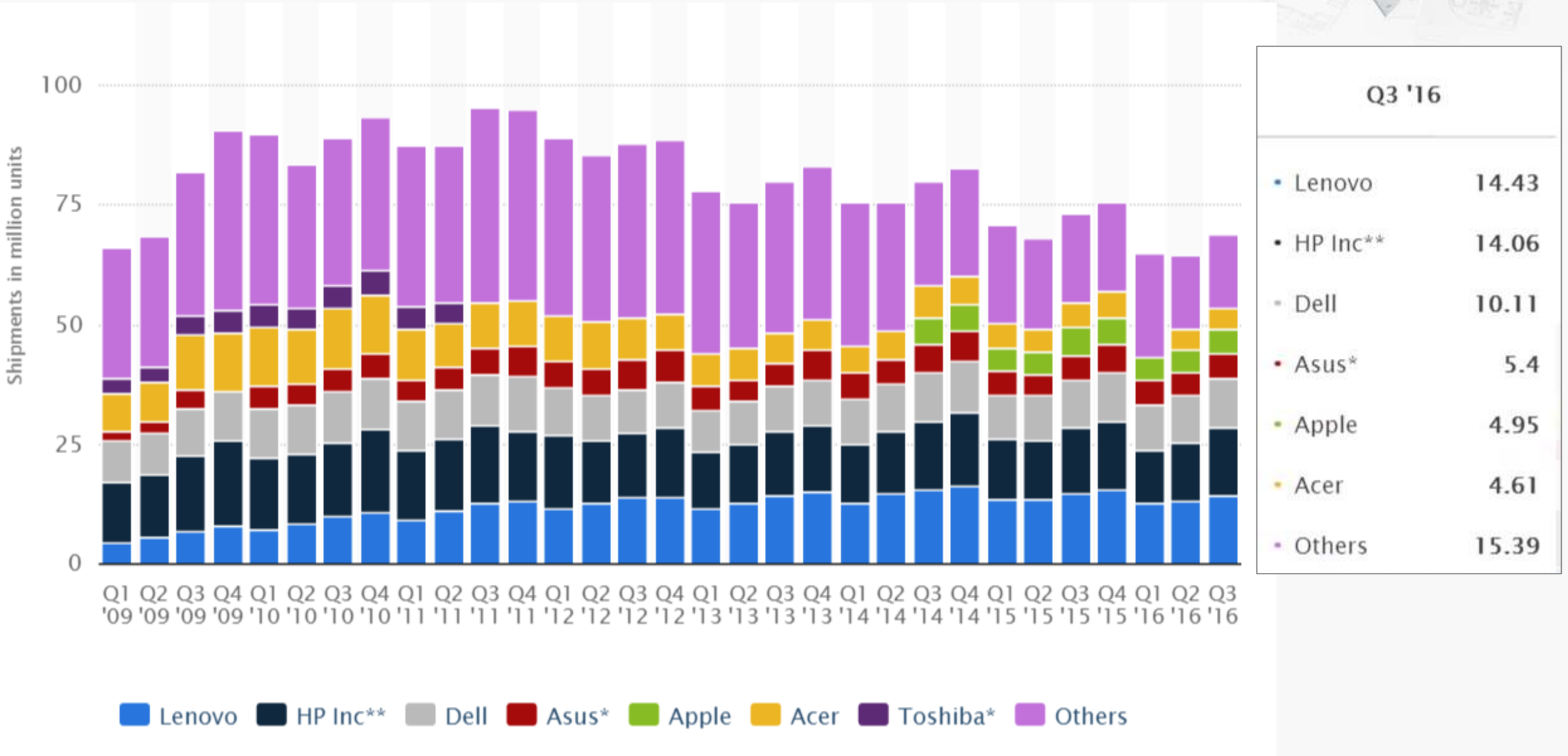
Android activations (est.)

iOS shipments

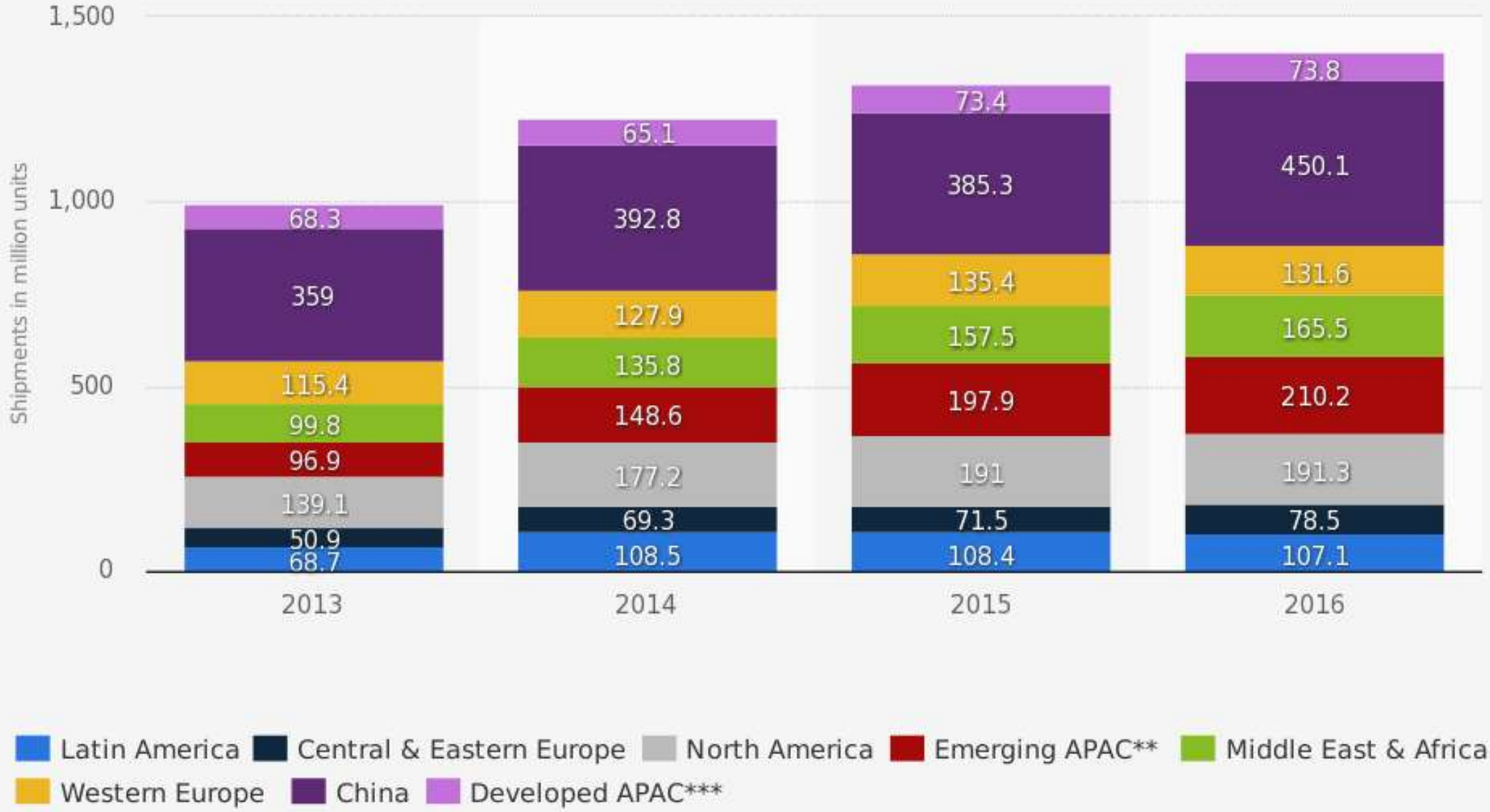
Windows Mobile/Phone

Windows PC shipments

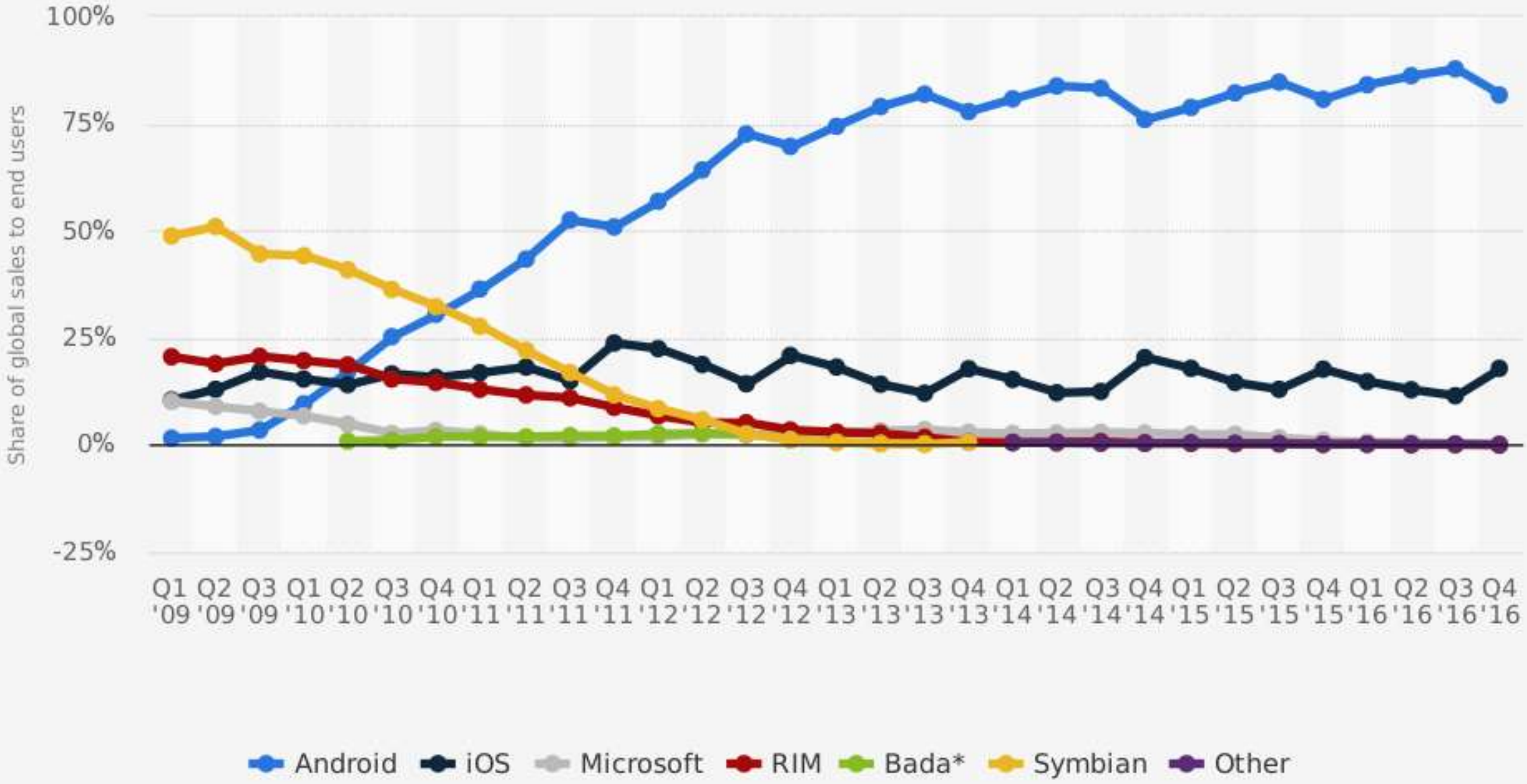
个人计算机的市场出货量（2009-2016）



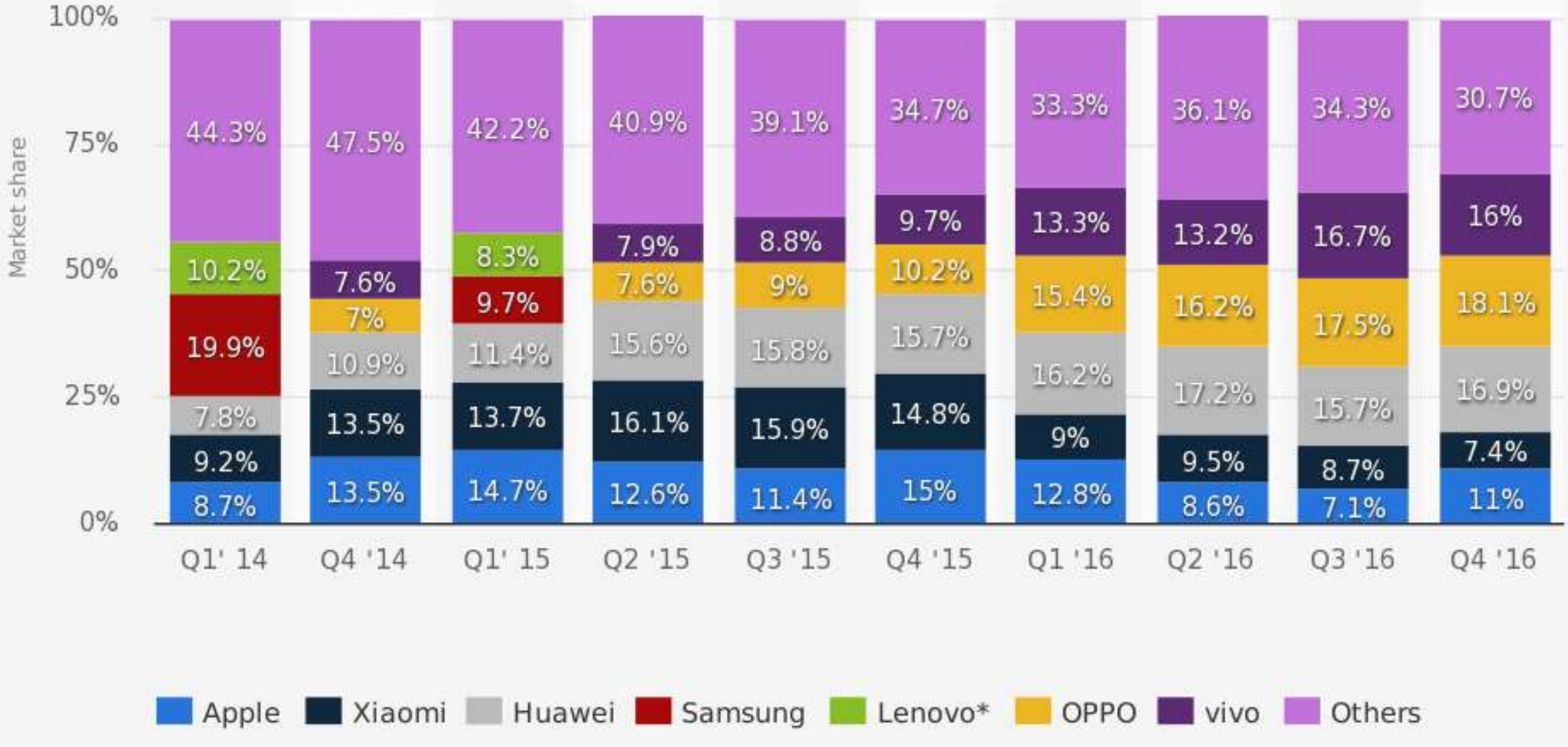
智能手机的市场出货量（2013-2016）



智能手机操作系统的市场份额（2009-2016）



中国的智能手机市场份额（2014-2016）



主要内容

通过学习本课程
了解计算机的发展历程，理解计算机的组成原理，掌握计算机的设计方法

MIPS的设计指导思想

- MIPS的全称
  - Microprocessor without Interlocked Piped Stages
- 主要关注点
  - 减少指令的类型
  - 降低指令复杂度
- 基本原则
  - A simpler CPU is a faster CPU.

MIPS指令示例

- 装载
  - 格式：`lw $8,($19)`
  - 操作：以19号寄存器的内容为地址，取出存储器中的32位数据，存入8号寄存器
- 加法
  - 格式：`add $10,$9,$8`
  - 操作：将8号和9号寄存器的内容相加，结果存入10号寄存器中
- 存储
  - 格式：`sw $10,32($19)`
  - 操作：将10号寄存器的内容存入存储器，地址为19号寄存器的内容加32

MIPS的通用寄存器（32个，每个都是32位宽）

编号	名称	用途	编号	名称	用途
0	$zero	The Constant Value 0	24-25	$t8-$t9	Temporaries
1	$at	Assembler Temporary	26-27	$k0-$k1	Reserved for OS Kernel
2-3	$v0-$v1	Values for Function Results and Expression Evaluation	28*	$gp	Global Pointer
4-7	$a0-a3	Arguments	29*	$sp	Stack Pointer
8-15	$t0-$t7	Temporaries	30*	$fp	Frame Pointer
16-23*	$s0-$s7	Saved Temporaries	31*	$ra	Return Address

* Preserved across a call

通用寄存器使用示例

- 以下指令与对应注释中的指令相同

```
lw   $t0,12($s3)
# lw   $8,12($19)

add $t0,$s2,$t0
# add $8,$18,$8

sw   $t0,40($s3)
# sw   $8,40($19)
```

编号	名称	用途
8-15	$t0-$t7	Temporaries
16-23	$s0-$s7	Saved Temporaries

MIPS指令示例

- 假设变量和寄存器的对应关系如下

```
f → $s0        g → $s1    h → $s2
i → $s3        j → $s4
```

f = (g + h) – (i + j)

```
add $t1, $s3, $s4
add $t2, $s1, $s2
sub $s0, $t2, $t1
```

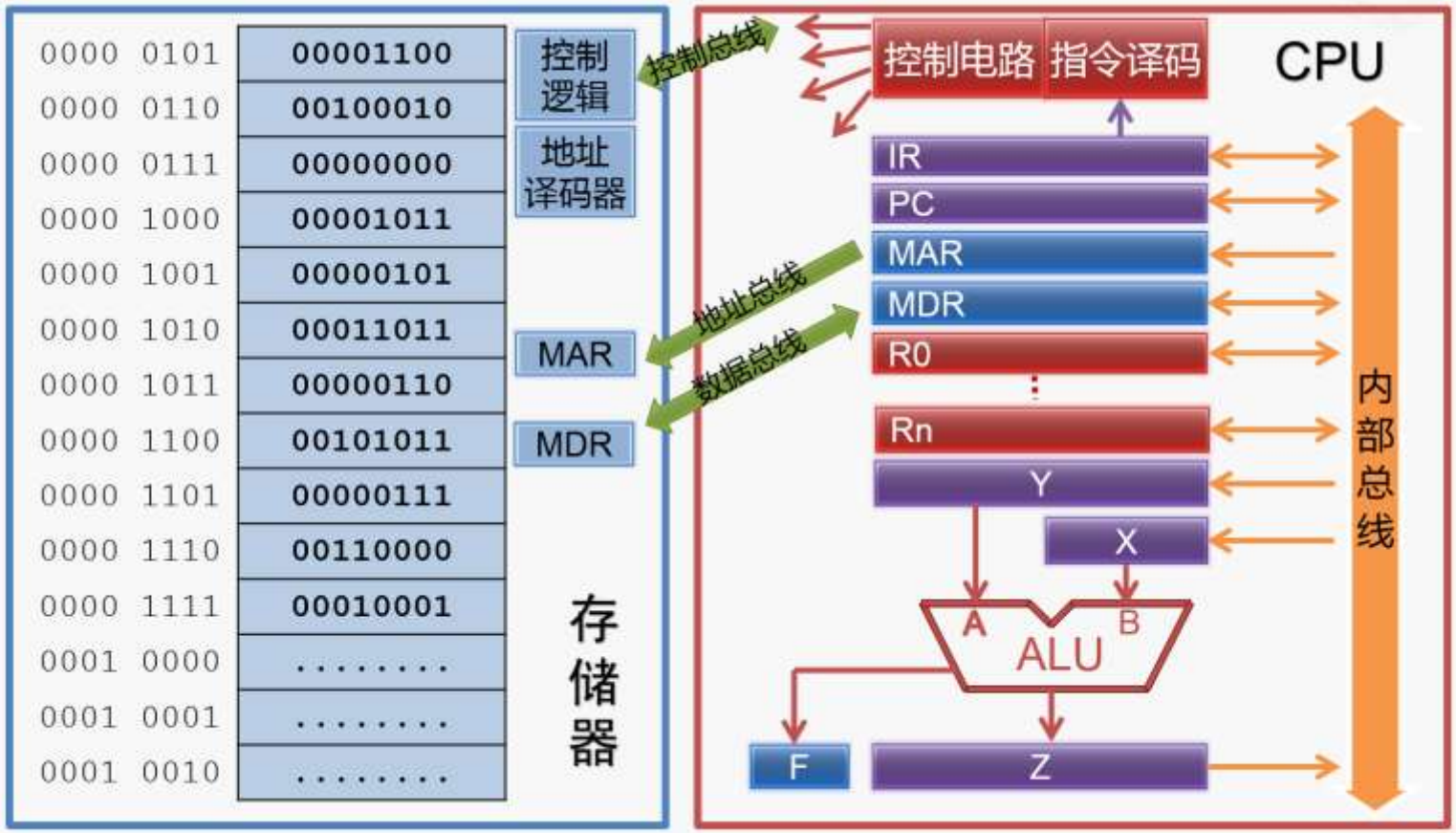
MIPS指令的主要特点（1）

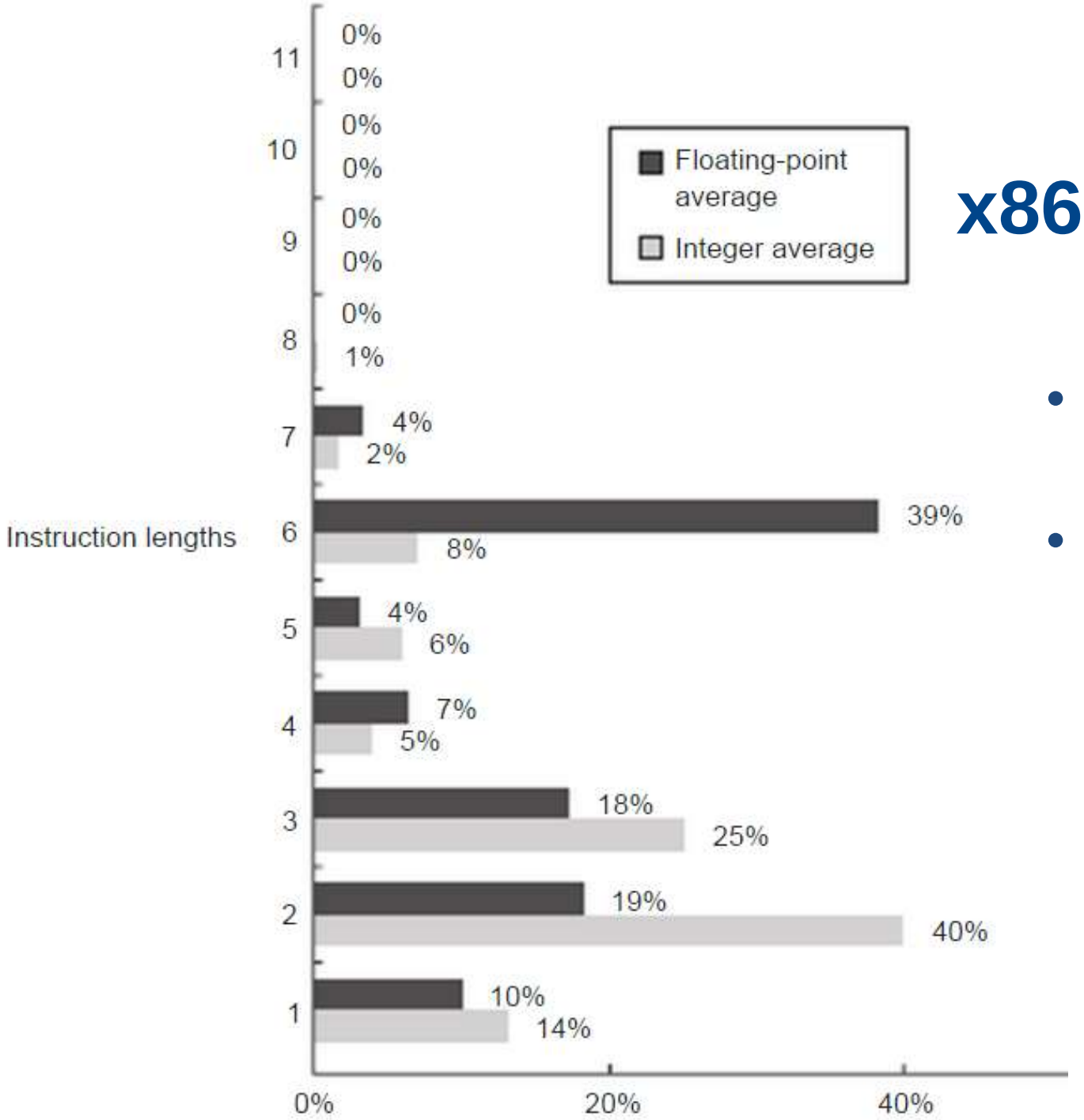
- 固定的指令长度（32-bit，即1 word）
  - 简化了从存储器取指令

x86指令

- 长度不确定
- 最短1个字节
- 长可达15个字节

模型机的CPU和存储器





x86指令不同长度的使用比例

- 整数指令平均长度2.8个字节
- 浮点指令平均长度4.1个字节

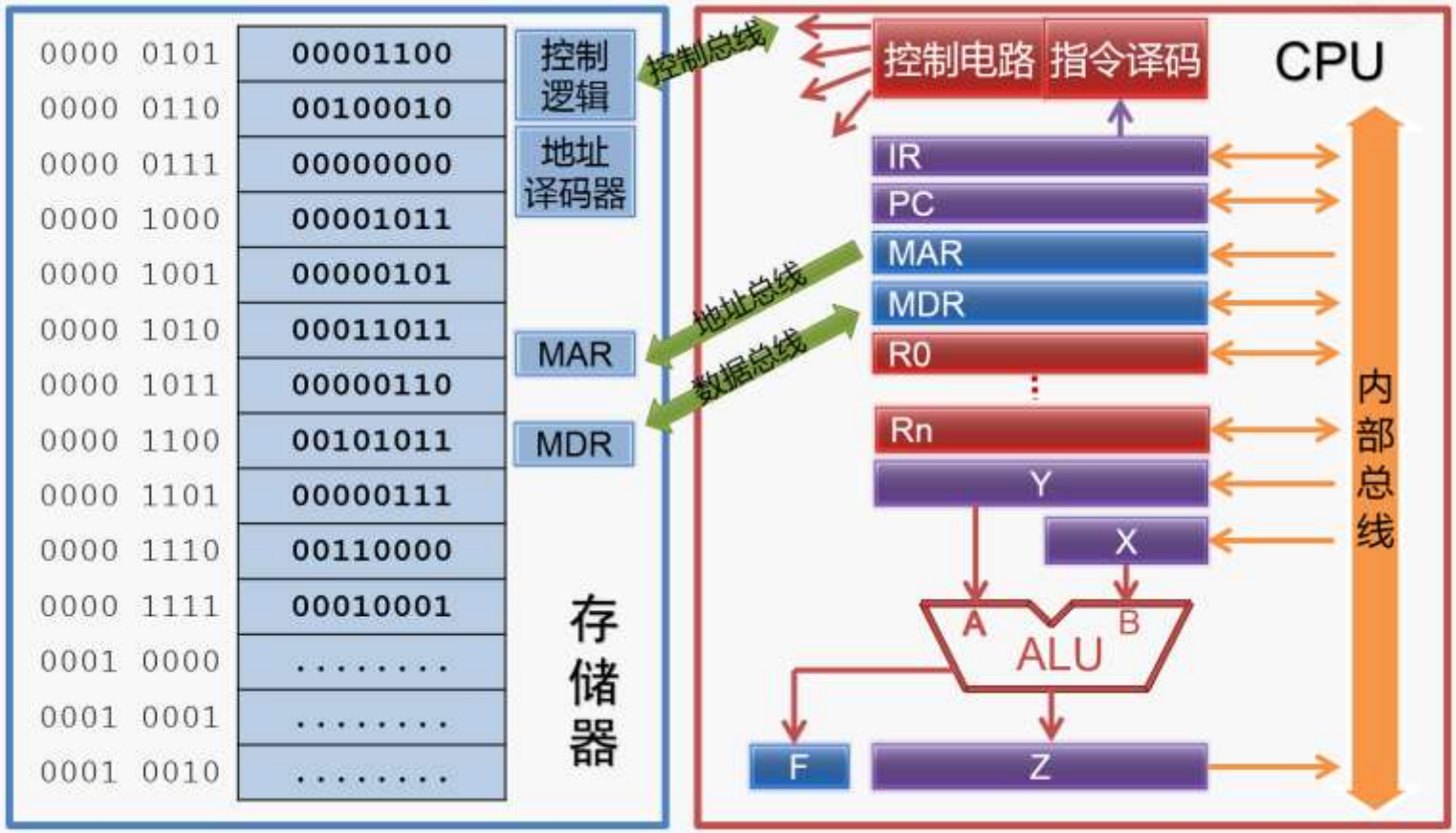
MIPS指令的主要特点（2）

- 只有Load和Store指令可以访问存储器

例如，不支持x86

指令的这种操作：

```
ADD AX,[3000H]
```

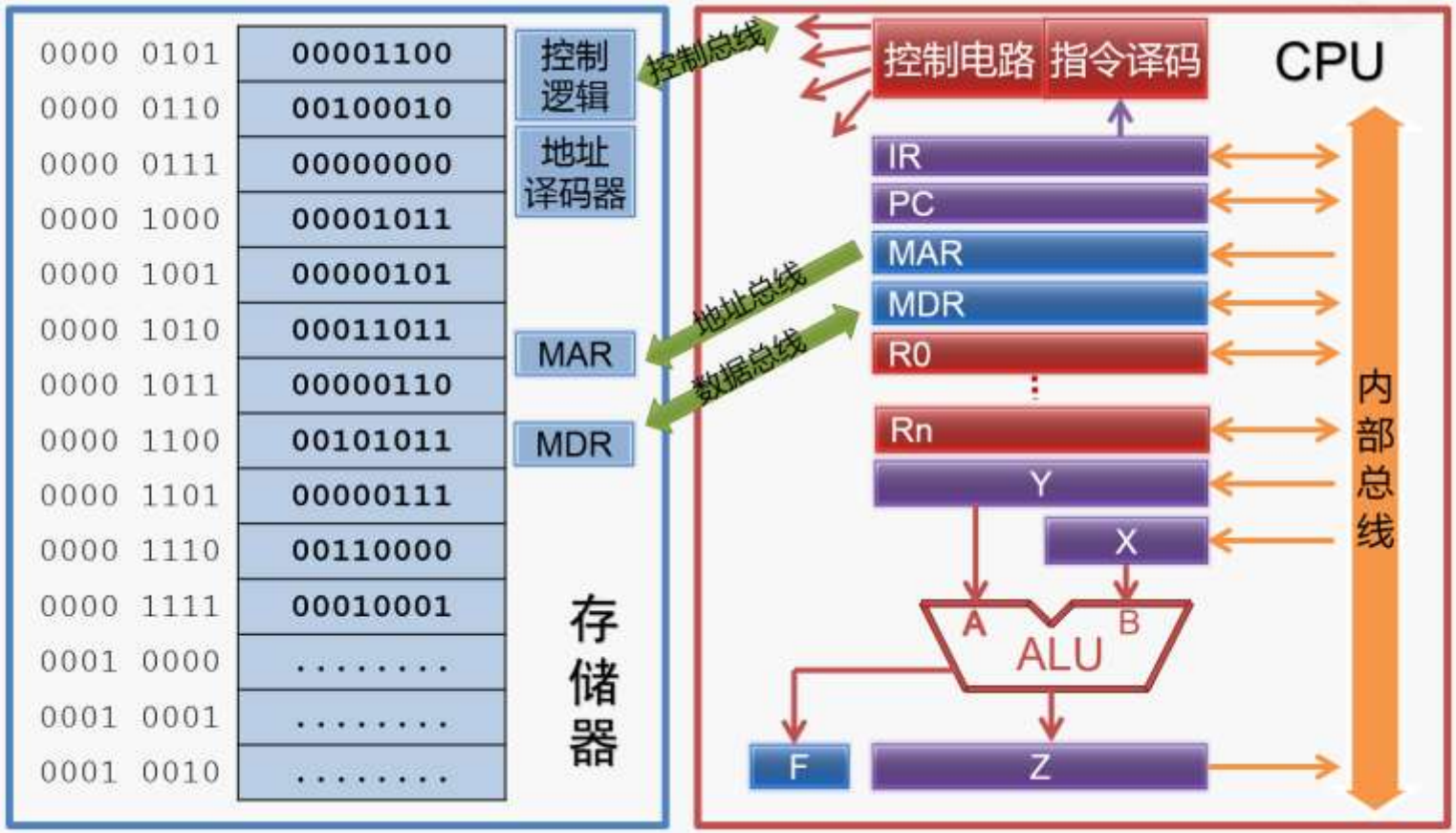


MIPS指令的主要特点（3）

- 简单的寻址模式
  - 简化了从存储器取操作数

```
lw $8,($19)
sw $10,32($19)
```

```
MOV EBX, 40
MOV AL, BL
MOV ECX, [1000H]
MOV [DI], AX
MOV DX, [BX+SI*2+200H]
```

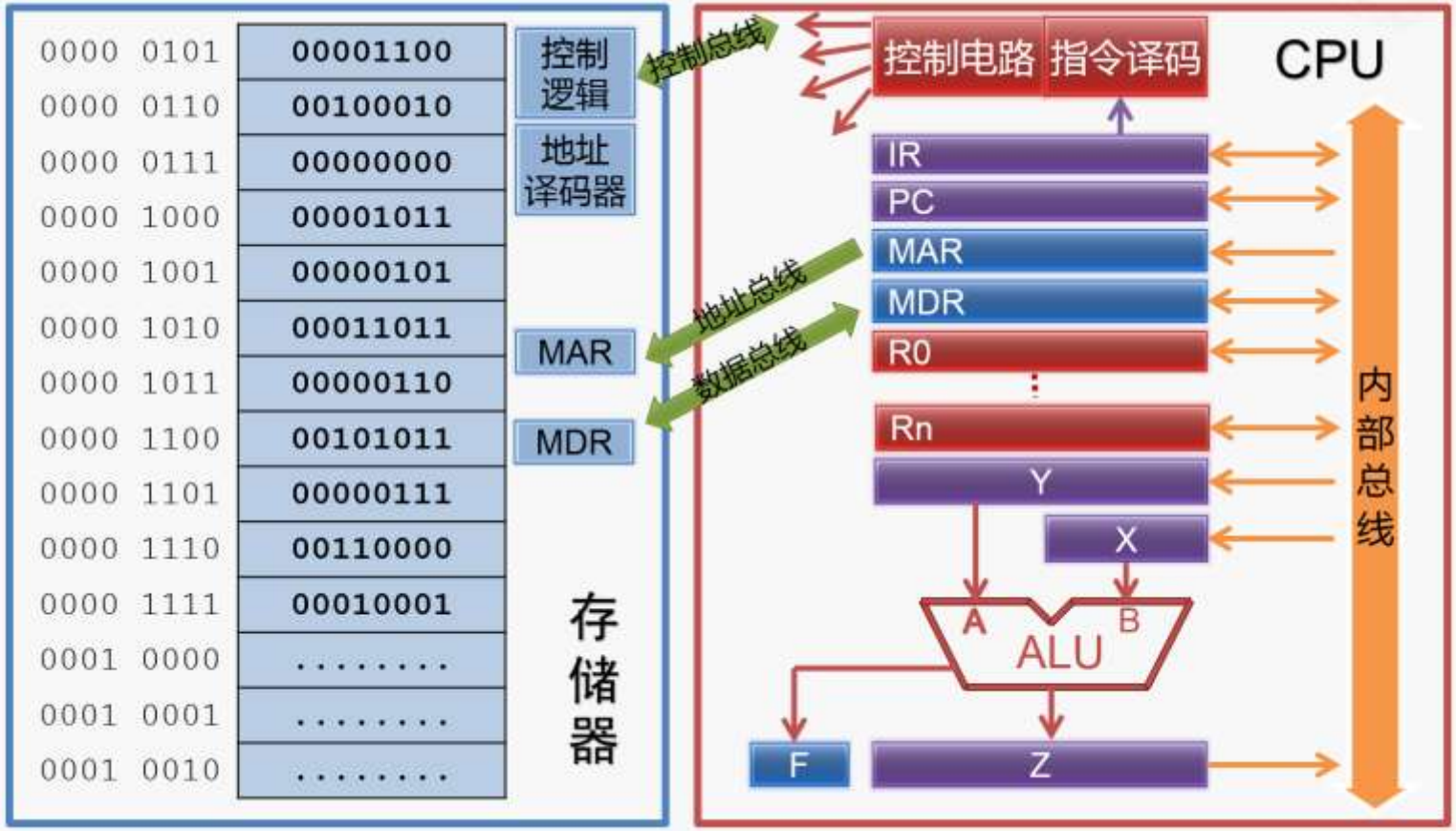


MIPS指令的主要特点（4）

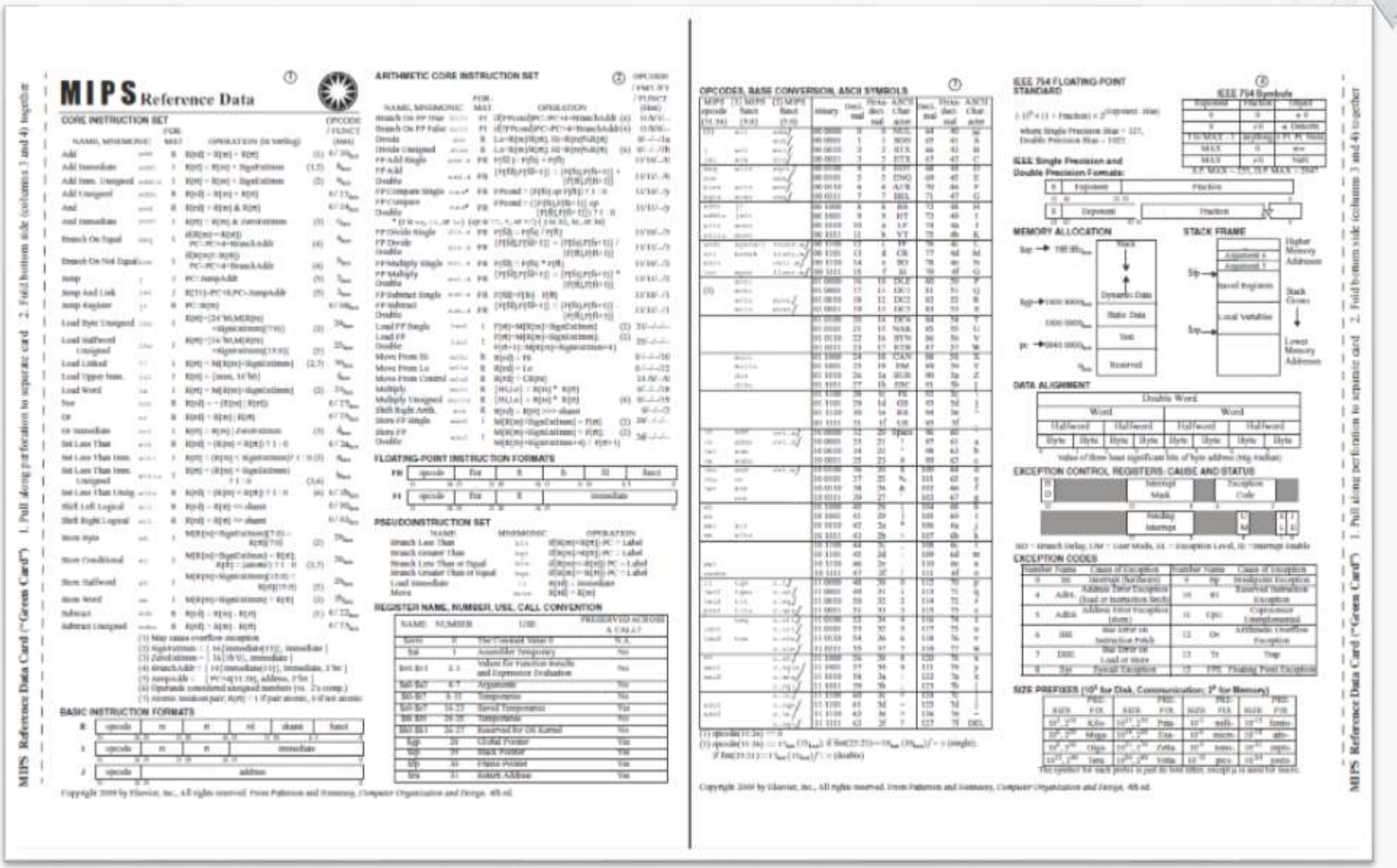
- 指令数量少，指令功能简单
  - 一条指令只完成一个操作，简化指令的执行过程

影响

- 处理器设计简单
- 处理器运行速度快
- 编程复杂
- 程序代码量大
- 需要优秀的编译器



MIPS指令



MIPS Reference Data

CORE INSTRUCTION SET

ARITHMETIC CORE INSTRUCTION SET

FLOATING-POINT INSTRUCTION FORMATS

PSEUDOINSTRUCTION SET

REGISTER NAME, NUMBER, USE, CALL CONVENTION

BASIC INSTRUCTION FORMATS

OPCODES, BASE CONVERSION, ASCII SYMBOLS

IEEE 754 FLOATING-POINT STANDARD

IEEE 754 Symbols

IEEE Single Precision and Double Precision Formats:

MEMORY ALLOCATION

STACK FRAME

DATA ALIGNMENT

EXCEPTION CONTROL REGISTERS: CAUSE AND STATUS

EXCEPTION CODES

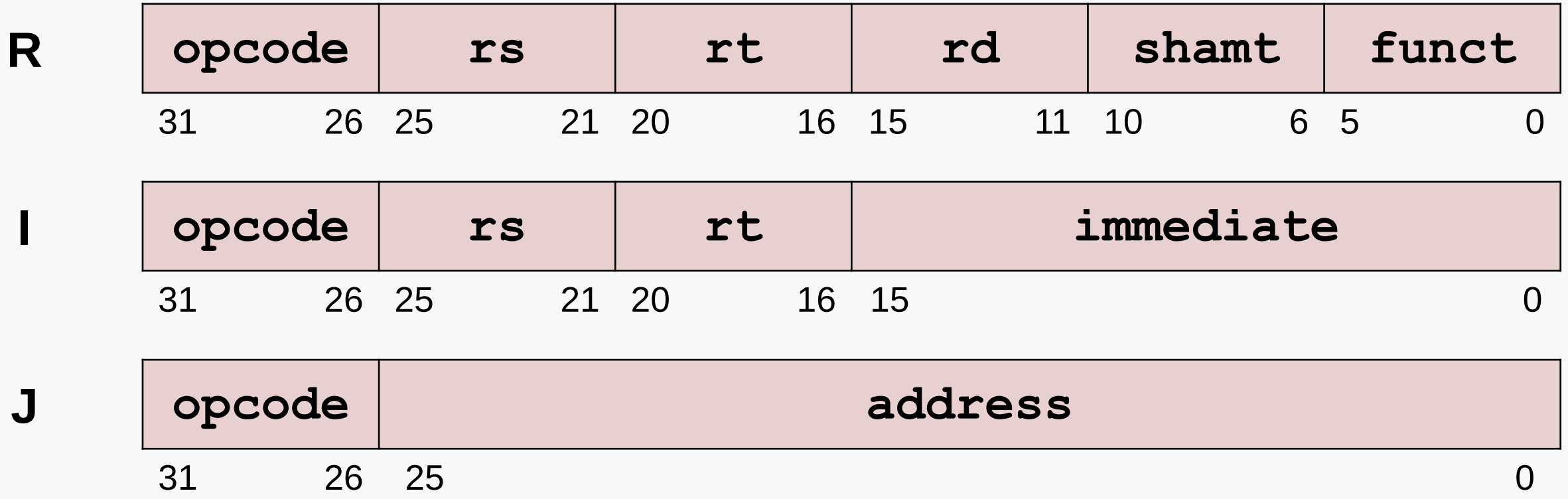
SIZE PREFIXES (10^3 for Disk, Communication; 2^{10} for Memory)

MIPS Reference Data Card ("Green Card") 1. Pull along perforation to separate card 2. Fold bottom side (columns 3 and 4) together

MIPS Reference Data Card ("Green Card") 1. Pull along perforation to separate card 2. Fold bottom side (columns 3 and 4) together

MIPS指令的基本格式

- R：Register，寄存器
- I：Immediate，立即数
- J：Jump，无条件转移



不同维度的指令分类（示例）

运算指令	add rd,rs,rt sll rd,rt,*shamt*	addi rt,rs,*imm* slti rt,rs,*imm*	/
访存指令	/	lw rt,*imm*(rs) sw rt,*imm*(rs)	/
分支指令	jr rs	beq rs,rt,*imm*	j *addr*
	R型指令	I型指令	J型指令

主要内容

通过学习本课程
了解计算机的发展历程，理解计算机的组成原理，掌握计算机的设计方法

Ⅰ RISC的发展变迁

Ⅱ MIPS指令的主要特点

Ⅲ MIPS指令分类说明：R型

Ⅳ MIPS指令分类说明：I型

不同维度的指令分类（示例）

运算指令	`add rd,rs,rt` `sll rd,rt,`*`shamt`*	`addi rt,rs,`*`imm`* `slti rt,rs,`*`imm`*	/
访存指令	/	`lw rt,`*`imm`*`(rs)` `sw rt,`*`imm`*`(rs)`	/
分支指令	`jr rs`	`beq rs,rt,`*`imm`*	`j `*`addr`*
	R型指令	I型指令	J型指令

R型指令的格式（1）

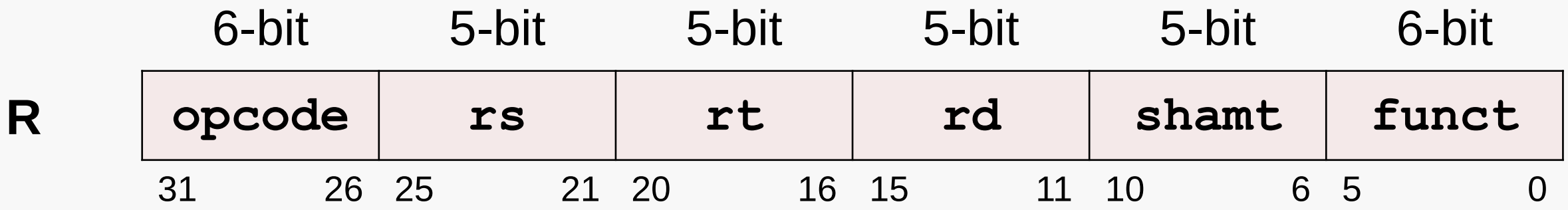
- R型指令格式包含6个域
  - 2个6-bit域，可表示0~63的数
  - 4个5-bit域，可表示0~31的数

用于指定指令的类型。对于所有R型指令，该域的值均为0

opcode

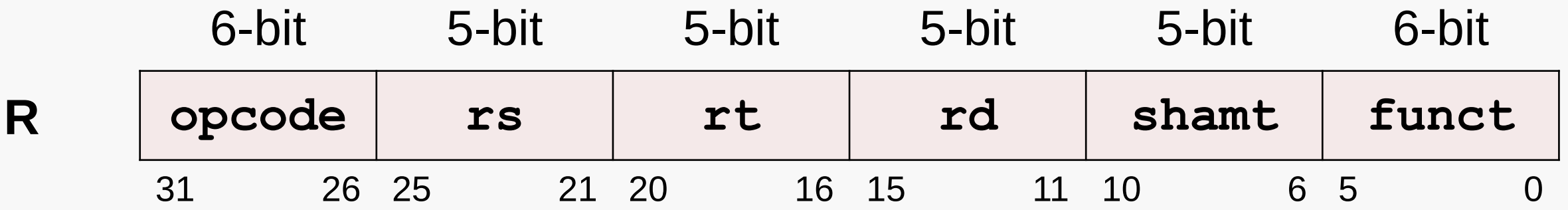
与opcode域组合，精确地指定指令的类型

funct



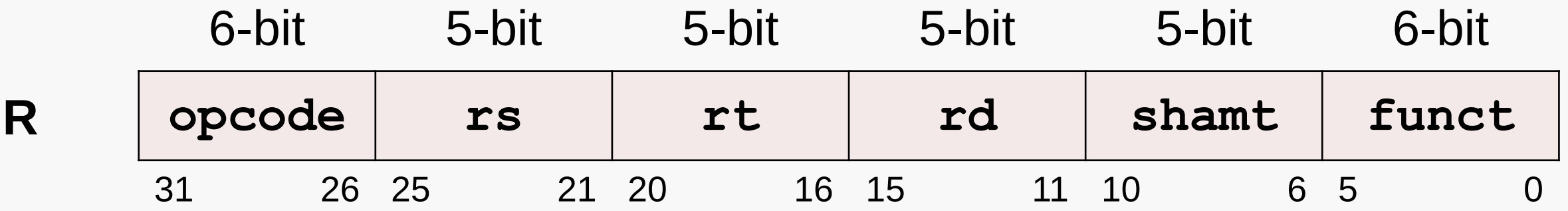
R型指令的格式（2）

- `rs` Source Register
  - 通常用于指定第一个源操作数所在的寄存器编号
- `rt` Target Register
  - 通常用于指定第二个源操作数所在的寄存器编号
- `rd` Destination Register
  - 通常用于指定目的操作数（保存运算结果）的寄存器编号
- 5-bit的域可表示0~31，对应32个通用寄存器



R型指令的格式（3）

- shamt　shift amount
  - 用于指定移位指令进行移位操作的位数
  - 5-bit的域可表示0~31，对于32-bit数，更多移位没有实际意义
  - 对于非移位指令，该域设为0



R型指令的编码示例（1）

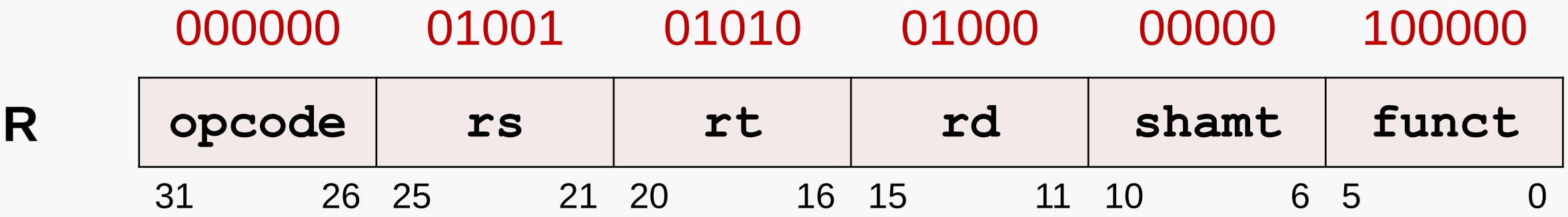
- **add $8,$9,$10**　# R[rd]=R[rs]+R[rt]
  - 查指令编码表得到：

  opcode = 0，funct = 32，shamt = 0（非移位指令）

  - 根据指令操作数得到：

  rd = 8（目的操作数），rs = 9（第一个源操作数）

  rt = 10（第二个源操作数）



R型指令的编码示例（2）

- **sll $8,$9,10** # R[rd]=R[rt]<<shamt
  - 查指令编码表得到：

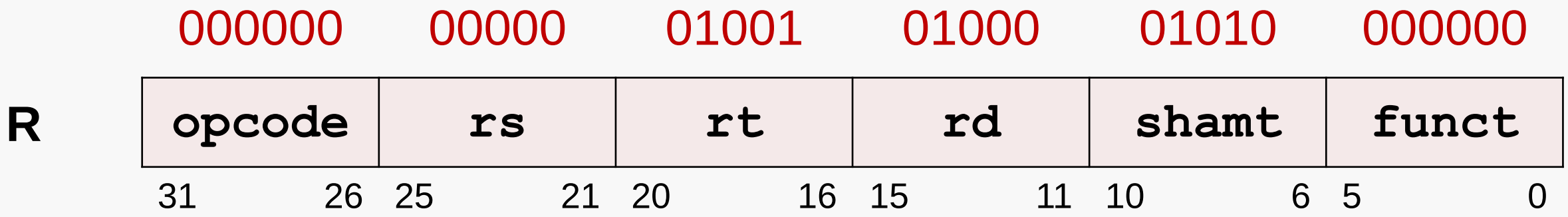
  opcode = 0， funct = 0， rs = 0 （未使用的寄存器）

  - 根据指令操作数得到：

  rd = 8 （目的操作数）

  rt = 9 （源操作数）

  shamt = 10（移位数）



主要内容

通过学习本课程
了解计算机的发展历程，理解计算机的组成原理，掌握计算机的设计方法

Ⅰ RISC的发展变迁

Ⅱ MIPS指令的主要特点

Ⅲ MIPS指令分类说明：R型

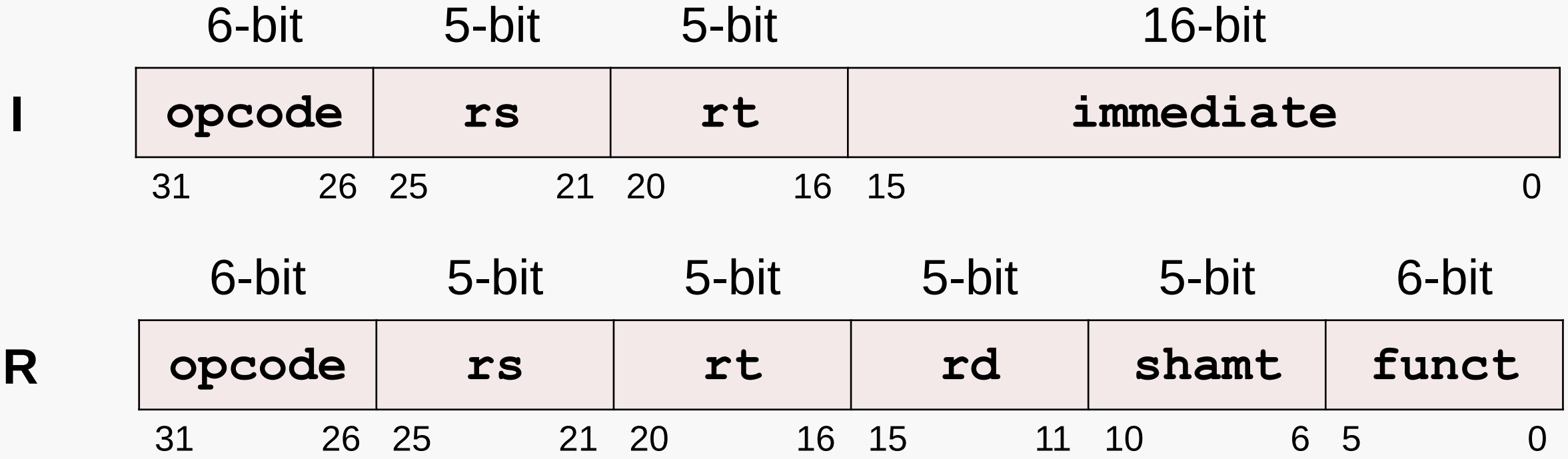
Ⅳ MIPS指令分类说明：I型

不同维度的指令分类（示例）

	R型指令	I型指令	J型指令
运算指令	add rd,rs,rt sll rd,rt,*shamt*	addi rt,rs,*imm* slti rt,rs,*imm*	/
访存指令	/	lw rt,*imm*(rs) sw rt,*imm*(rs)	/
分支指令	jr rs	beq rs,rt,*imm*	j *addr*

I型指令的格式（1）

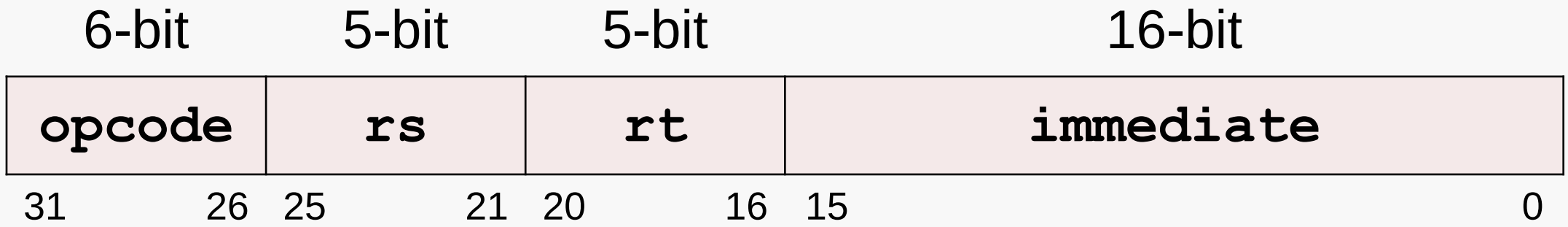
- R型指令只有一个5-bit域表示立即数，范围为0~31
- 常用的立即数远大于这个范围，因此需要新的指令格式
- I型指令的大部分域与R型指令相同



I型指令的格式（2）

- opcode
  - 用于指定指令的操作类型（但没有funct域）
- rs Source Register
  - 指定第一个源操作数所在的寄存器编号
- rt Target Register
  - 指定用于目的操作数（保存运算结果）的寄存器编号
  - 对于某些指令，指定第二个源操作数所在的寄存器编号

I

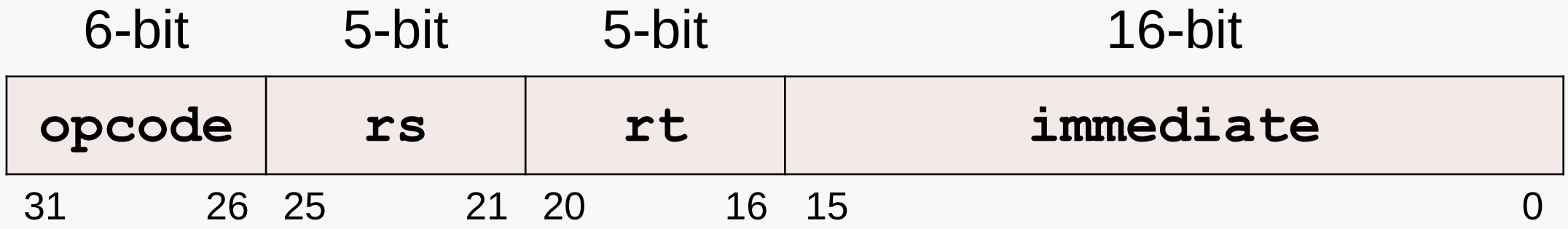


I型指令的格式（3）

- immediate
  - 16-bit的立即数，可以表示2^{16}个不同数值

  - 对于访存指令，如`lw rt,imm(rs)`
    通常可以满足访存地址偏移量的需求（-32768~+32767）
  - 对于运算指令，如`addi rt,rs,imm`
    无法满足全部需求，但大多数时候可以满足需求

I



I型指令的编码示例（1）

- **addi $21,$22,-50** # $21=$22+(-50)
  - 查指令编码表得到：

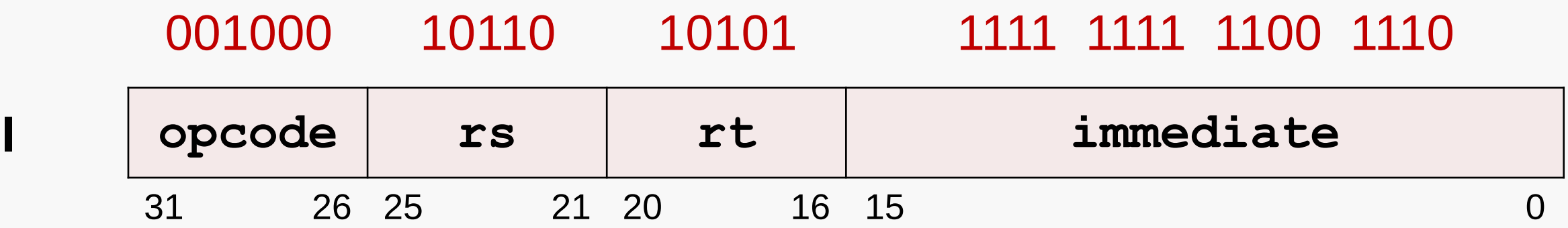
  opcode = 8

  - 分析指令得到：

  rs = 22 （源操作数寄存器编号）

  rt = 21 （目的操作数寄存器编号）

  immediate = -50 （立即数）



I型指令的编码示例（2）

- **lw $21,-50($22)** # $21=Mem[$22+(-50)]
  - 查指令编码表得到：

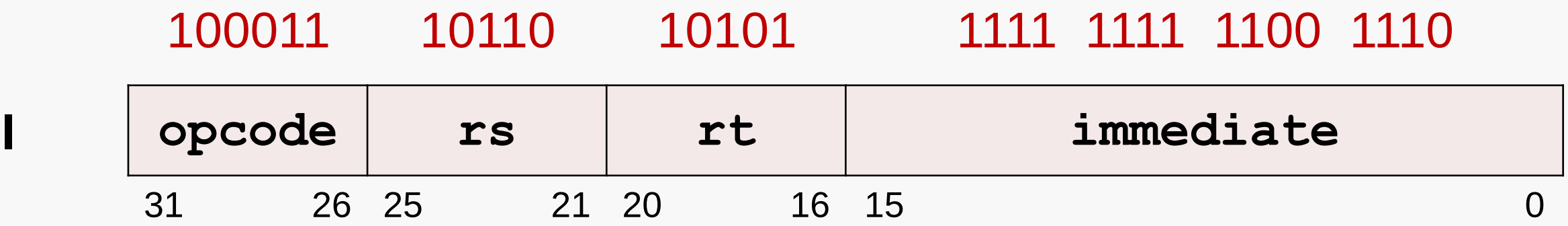
  opcode = 35

  - 分析指令得到：

  rs = 22 （源操作数寄存器编号）

  rt = 21 （目的操作数寄存器编号）

  immediate = -50 （立即数）



I型指令的编码示例（3）

- **slti $21,$22,-50** # $21=($22<(-50))?1:0
  - 查指令编码表得到：

  opcode = 10

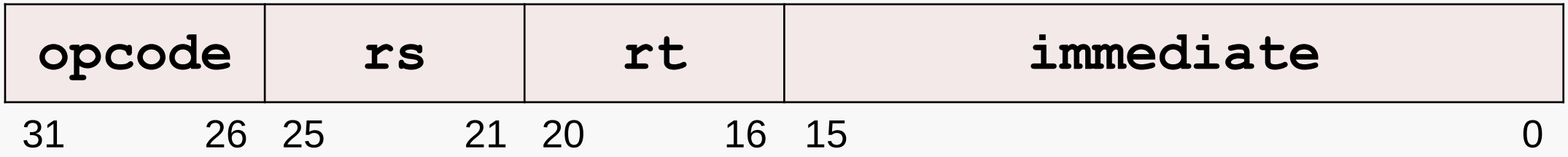
  - 分析指令得到：

  rs = 22 （源操作数寄存器编号）

  rt = 21 （目的操作数寄存器编号）

  immediate = -50 （立即数）

001010 10110 10101 1111 1111 1100 1110



本讲到此结束，谢谢

欢迎继续学习本课程

计算机组织与体系结构 Computer Architectures

主讲：陆俊林

北京大学 Peking University