计算机组织与体系结构

第七讲

计算机科学与技术学院

舒燕君

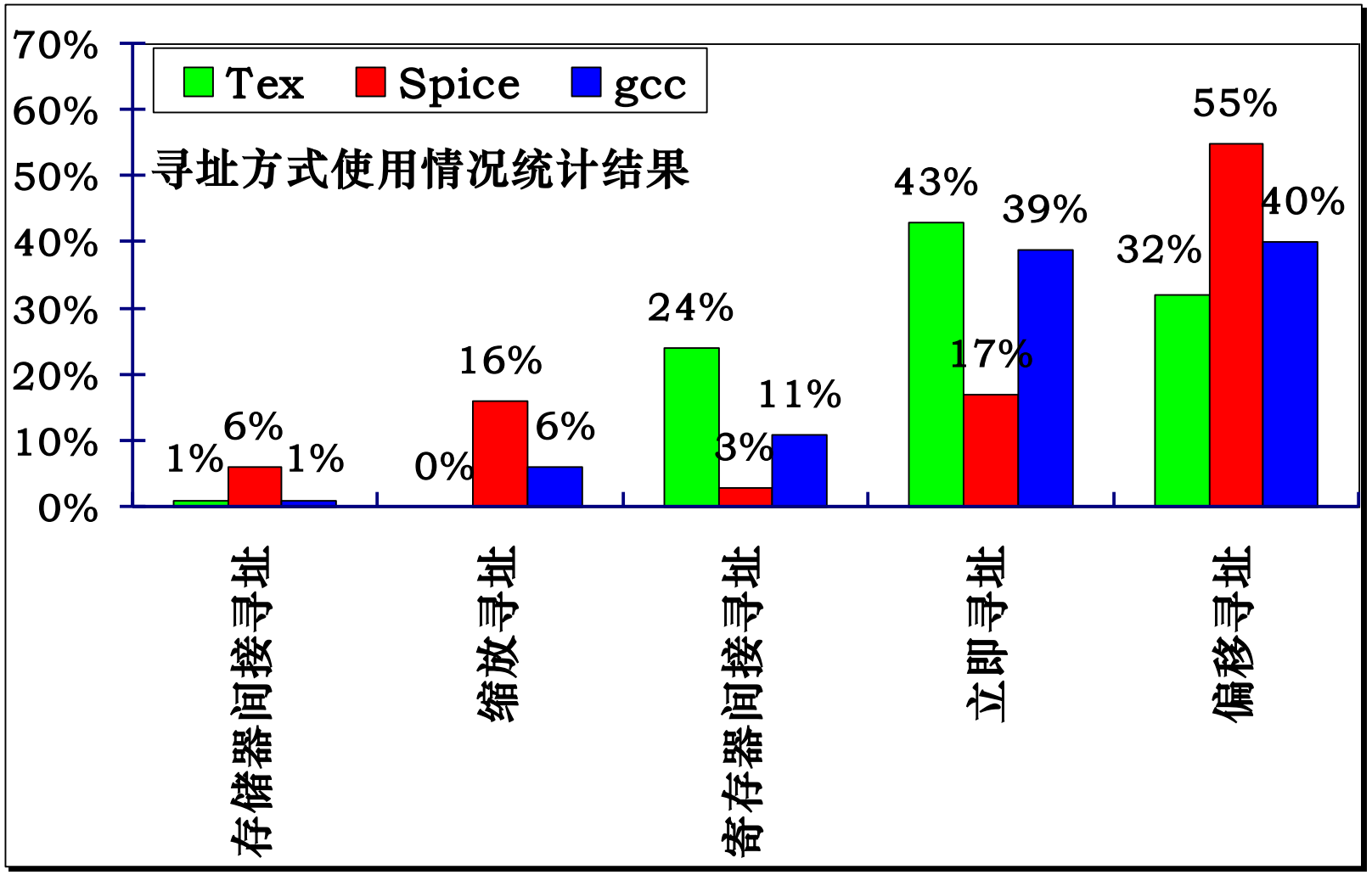
Recap

- 机器指令
  - ✓不同个数地址码形式
- 操作数类型和操作类型
  - ✓ 操作数的类型和操作数的表示
  - ✓ 操作数的存储方式
  - ✓ 操作的类型
- 寻址方式
  - ✓指令寻址：顺序、跳跃
  - ✓数据寻址：立即寻址、直接寻址、隐含寻址、间接寻址、寄存器寻址、寄存器间接寻址、基址寻址、变址寻址、相对寻址、堆栈寻址

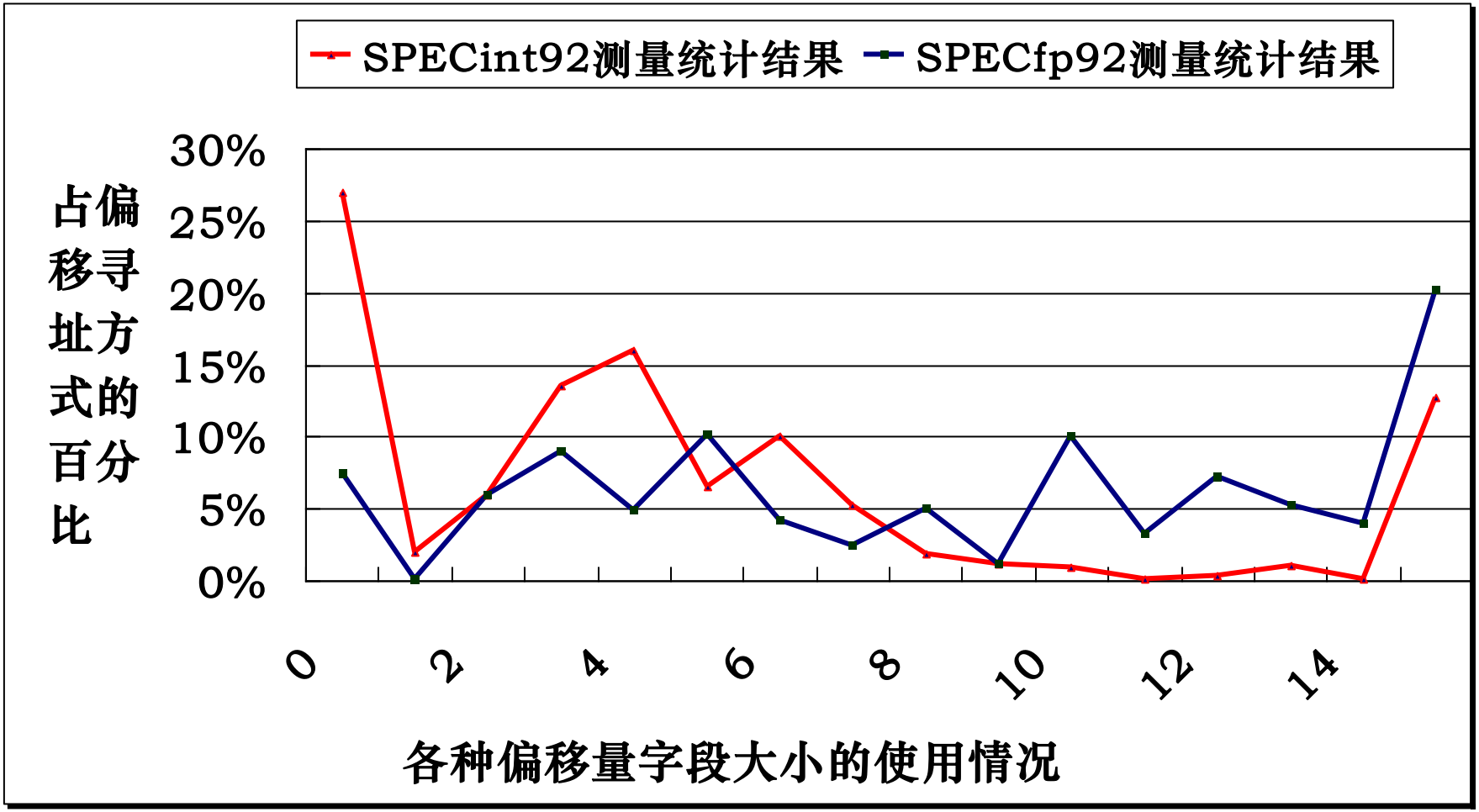
基本寻址方式优缺点

方式	算法	主要优点	主要缺点
立即寻址	操作数=A	无存储器访问	操作数幅值有限
直接寻址	EA=A	简单	地址范围有限
间接寻址	EA=（A）	大的地址范围	多重存储器访问
寄存器寻址	EA=R	无存储器访问	地址范围有限
寄存器间接寻址	EA=（R）	大的地址范围	额外存储器访问
相对/基址/变址寻址(偏移寻址)	EA=A+（R）	灵活	复杂
堆栈寻址	EA=栈顶	无存储器访问	应用有限

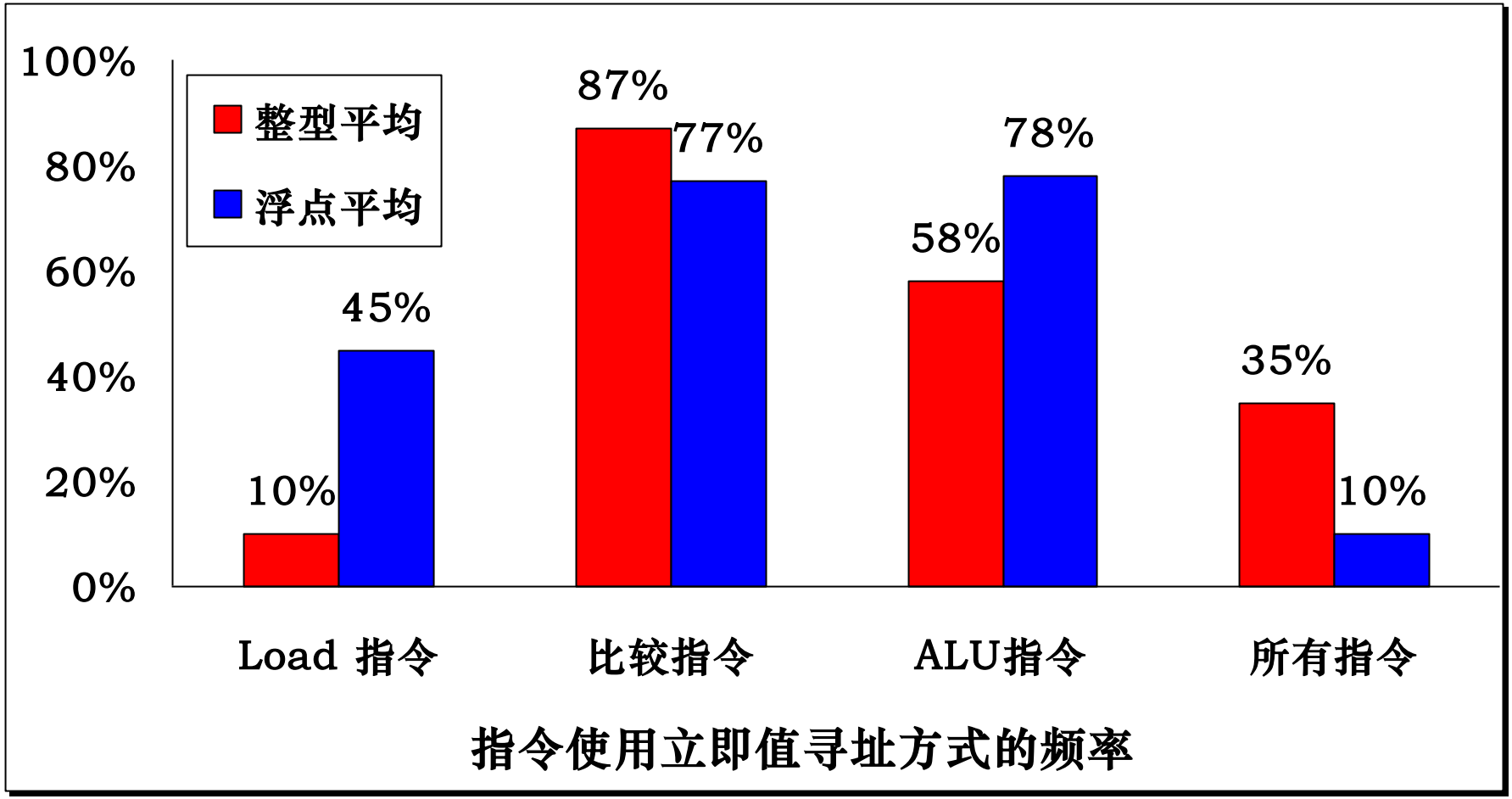
常用的一些操作数寻址方式



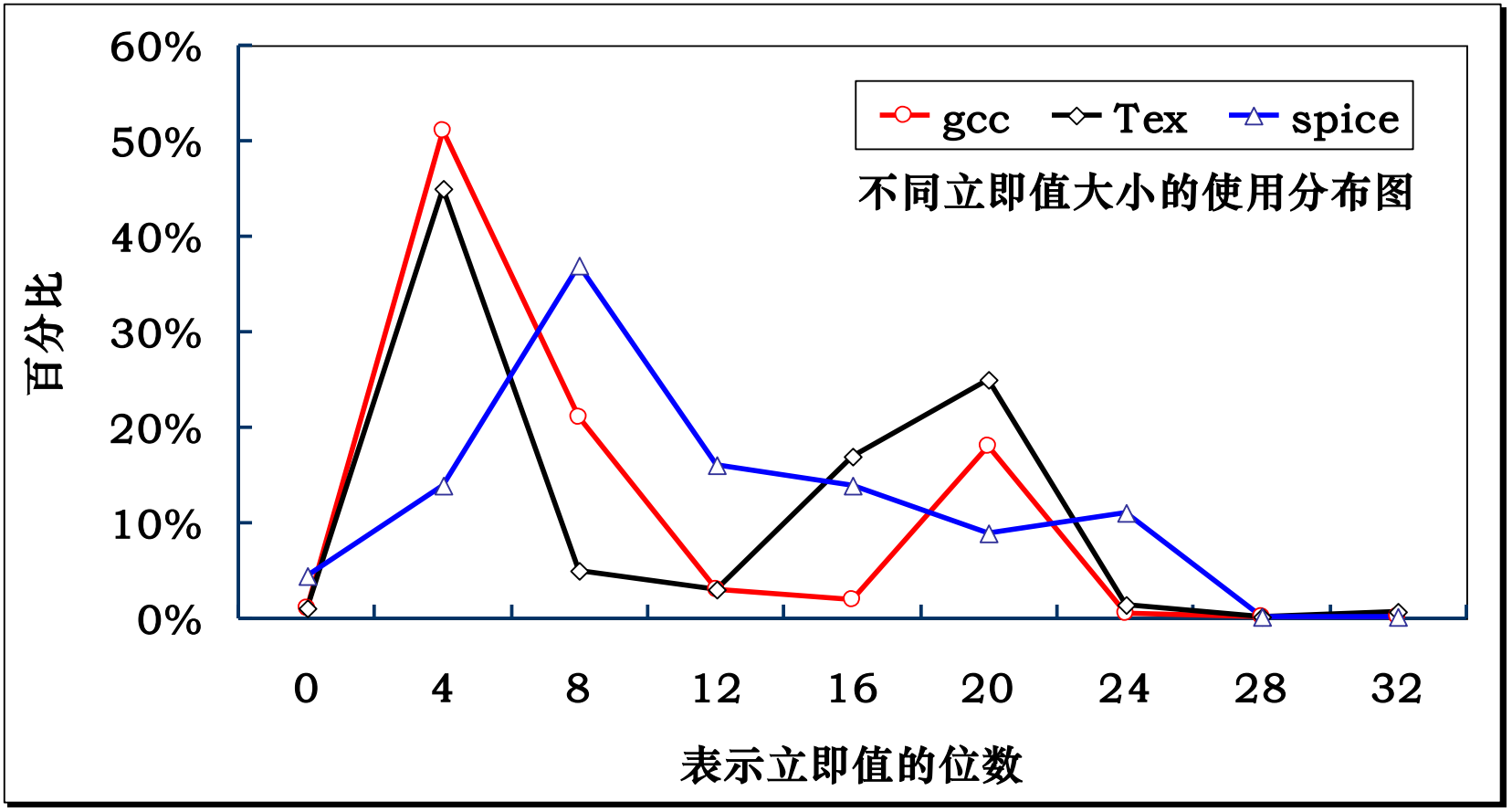
偏移寻址


SPECint92测量统计结果
SPECfp92测量统计结果
占偏移寻址方式的百分比
30%
25%
20%
15%
10%
5%
0%
0
2
4
6
8
10
12
14
各种偏移量字段大小的使用情况

立即寻址


100%
80%
60%
40%
20%
0%
整型平均
浮点平均
10%
45%
87%
77%
58%
78%
35%
10%
Load 指令
比较指令
ALU指令
所有指令
指令使用立即值寻址方式的频率

立即寻址


60%
50%
40%
30%
20%
10%
0%
百分比
gcc
Tex
spice
不同立即值大小的使用分布图
0
4
8
12
16
20
24
28
32
表示立即值的位数

第4章 指令系统

4.1 机器指令

4.2 操作数类型和操作类型

4.3 寻址方式

4.4 指令系统结构的分类

4.5 指令系统的设计与优化

4.6 指令系统的发展和改进

4.7 指令格式举例

4.4 指令集结构的分类

- 一般来说，可以从如下五个因素考虑对计算机指令集结构进行分类，即：
  - 在CPU中操作数的存储方法；
  - 指令中显式表示的操作数个数；
  - 操作数的寻址方式；
  - 指令集所提供的操作类型；
  - 操作数的类型和大小。

4.4 指令集结构的分类

- CPU中用来存储操作数的存储单元主要有：
  - 堆栈；
  - 累加器；
  - 一组寄存器。
- 指令中的操作数可以被明确地显式给出，也可以按照某种约定隐式地给出。

4.4 指令集结构的分类

- Z=X+Y表达式在这三种类型指令集结构上的实现方法

堆栈	累加器	寄存器 (寄存器-存储器)	寄存器 (寄存器-寄存器)
PUSH X PUSH Y ADD POP Z	LOAD X ADD Y Store Z	LOAD R1,X ADD R1,Y Store R1,Z	LOAD R1,X LOAD R2,Y ADD R3,R1,R2 Store R3,Z

4.4 指令集结构的分类

- 早期的大多数机器都是采用堆栈型或累加器型指令集结构，但是自1980年以来的大多数机器均采用的是寄存器型指令集结构。主要有三个方面的原因:
  - 集成电路技术飞速发展
  - 寄存器和CPU内部其它存储单元一样，要比存储器快
  - 对编译器而言，可以更容易有效地分配和使用寄存器

通用寄存器型指令集结构

- 通用寄存器型指令集结构的主要优点：
  - 在表达式求值方面，比其它类型指令集结构都具有更大的灵活性；
  - 寄存器可以用来存放变量；
    - 减少存储器的通信量，加快程序的执行速度（因为寄存器比存储器快）
    - 可以用更少的地址位来寻址寄存器，从而可以有效改进程序的目标代码大小。

通用寄存器型指令集结构

- 两种主要的指令特性能够将通用寄存器型指令集结构（GPR）进一步细分。
  - ALU指令到底有两个或是三个操作数？
  - 在ALU指令中，有多少个操作数可以用存储器来寻址，也即有多少个存储器操作数？

通用寄存器型指令集结构

ALU指令中存储器操作数的个数	ALU指令中操作数的最多个数	结构类型	机器实例
0	3	RR	MIPS, SPARC, Alpha, PowerPC, ARM
1	2	RM	IBM 360/370, Intel 80x86, Motorola 6800
1	3	RM	IBM 360/370
2	2	MM	VAX
3	3	MM	VAX

通用寄存器型指令集结构的分类

- 可以将当前大多数通用寄存器型指令集结构进一步细分为三种类型：
  - 寄存器－寄存器型

    (R－R： register-register)
  - 寄存器－存储器型

    (R－M： register-memory)
  - 存储器－存储器型

    (M－M： memory-memory)

(m，n)表示指令的n个操作数中有m个存储器操作数

三种通用寄存器型指令集结构的优缺点

- 寄存器－寄存器型（0,3）
  - 优点：
    - 指令字长固定，指令结构简洁，是一种简单的代码生成模型，各种指令的执行时钟周期数相近。
  - 缺点：
    - 与指令中含存储器操作数的指令系统结构相比，指令条数多，目标代码不够紧凑，因而程序占用的空间比较大。

三种通用寄存器型指令集结构的优缺点

- 寄存器－存储器型（1,2）
  - 优点：
    - 可以在ALU指令中直接对存储器操作数进行引用，而不必先用load指令进行加载，容易对指令进行编码，目标代码比较紧凑。
  - 缺点：
    - 由于有一个操作数的内容将被破坏，所以指令中的两个操作数不对称。在一条指令中同时对寄存器操作数和存储器操作数进行编码，有可能限制指令所能够表示的寄存器个数。指令的执行时钟周期因操作数的来源（寄存器或存储器）的不同而差别比较大。

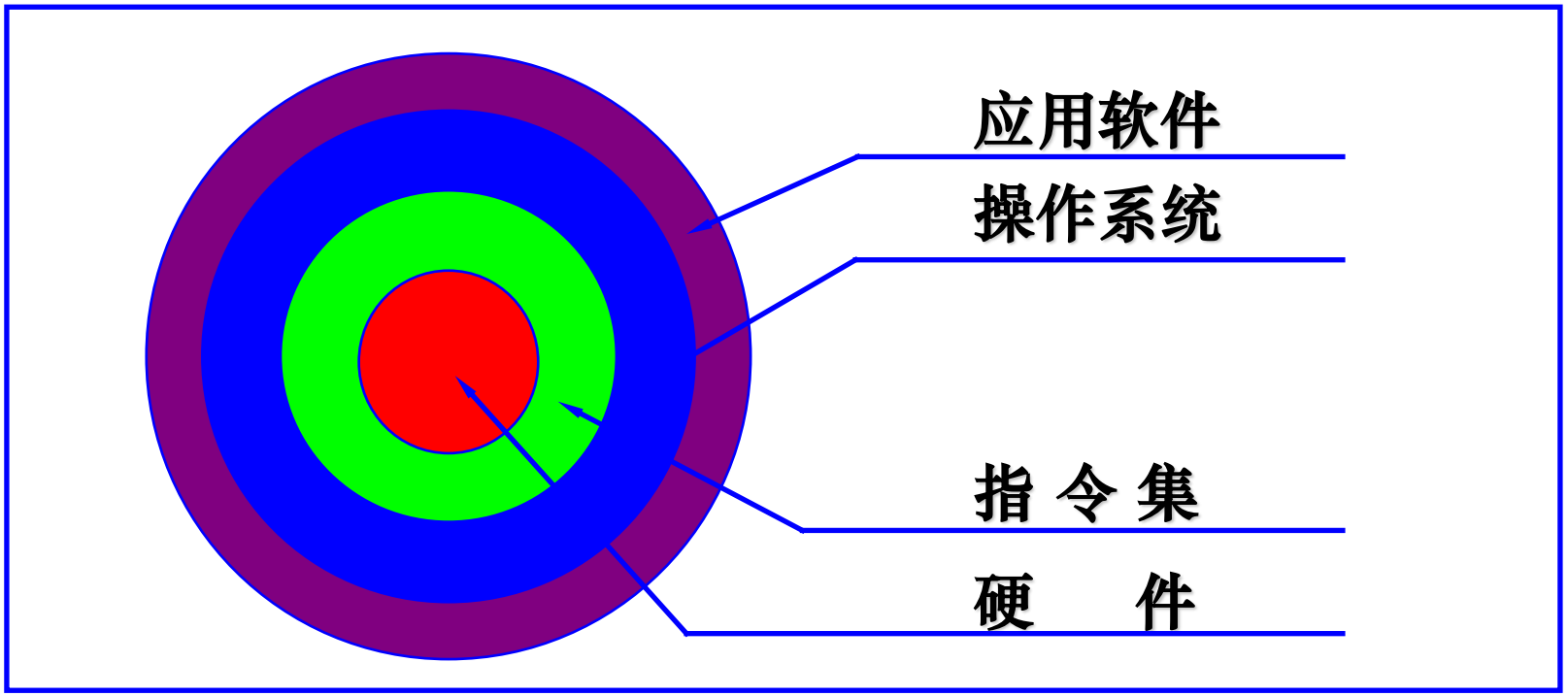
三种通用寄存器型指令集结构的优缺点

- 存储器—存储器型（（2，2）或（3，3））
  - 优点：
    - 目标代码最紧凑，不需要设置存储器来保存变量。
  - 缺点：
    - 指令字长变换很大，特别是3个操作数指令。而且每条指令完成的工作也差别很大。对存储器的频率访问会使存储器成为瓶颈。这种类型的指令系统现在已经不用了。

三种类型指令集结构的优缺点

指令集结构类型	优点	缺点
寄存器-寄存器（0，3）	指令字长固定，指令结构简洁，是一种简单的代码生成模型，各种指令的执行时钟周期数相近	与指令中含存储器操作数的指令系统结构相比，指令条数多，目标代码不够紧凑，因而程序占用的空间比较大
寄存器-存储器型（1，2）	可以在ALU指令中直接对存储器操作数进行引用，而不必先用load指令进行加载，容易对指令进行编码，目标代码比较紧凑	由于有一个操作数的内容将被破坏，所以指令中的两个操作数不对称。在一条指令中同时对寄存器操作数和存储器操作数进行编码，有可能限制指令所能够表示的寄存器个数。指令的执行时钟周期因操作数的来源（寄存器或存储器）的不同而差别比较大
存储器-存储器（2，2）或（3，3）	目标代码最紧凑，不需要设置存储器来保存变量	指令字长变换很大，特别是3个操作数指令。而且每条指令完成的工作也差别很大。对存储器的频率访问会使存储器成为瓶颈。这种类型的指令系统现在已经不用了

指令集结构设计概观



操作码	寻址方式	操作数	寻址方式	操作数

4.5 指令系统的设计和优化

4.5.1 指令系统设计的基本原则

4.5.2 控制指令

4.5.3 指令操作码的优化

4.5.1 指令系统设计的基本原则

1. 指令系统的设计
   - 首先考虑所应实现的基本功能，确定哪些基本功能应该由硬件实现，哪些功能由软件实现比较合适。
   - 包括
     - 指令的功能设计
     - 指令格式的设计

4.5.1 指令系统设计的基本原则

2. 在确定哪些基本功能用硬件来实现时，主要考虑3个因素：速度、成本、灵活性。

- 硬件实现的特点

  速度快、成本高、灵活性差

- 软件实现的特点

  速度慢、价格便宜、灵活性好

3. 对指令系统的基本要求

完整性、规整性、正交性、高效率、兼容性

4.5.1 指令系统设计的基本原则

- 完整性：在一个有限可用的存储空间内，对于任何可解的问题，编制计算程序时，指令系统所提供的指令足够使用。
  - 要求指令系统功能齐全、使用方便
  - 下表为许多指令系统结构都包含的一些指令类型
    - 前4类属于通用计算机系统的基本指令
    - 对于最后4种类型的操作，不同指令系统结构的支持大不相同 。

指令集操作的分类

算术和逻辑运算	整数的算术和逻辑操作：加、减、与、或等
数据传输	存数/取数
控制	分支、跳转、过程调用和返回、自陷等
系统	操作系统调用、虚拟存储器管理等
浮点	浮点操作：加、乘等
十进制	十进制加、十进制乘、十进制到字符的转换
字符串	字符串移动、字符串比较、字符串搜索等
图形	像素操作、压缩/解压操作等

4.5.1 指令系统设计的基本原则

- 规整性：主要包括对称性和均匀性。
  - 对称性：所有与指令系统有关的存储单元的使用、操作码的设置等都是对称的。
  - ✓例如，在存储单元的使用上，所有通用寄存器都要同等对待。在操作码的设计上，如果设置了A-B的指令，就应该也设置B-A的指令。
  - 均匀性：指对于各种不同的操作数类型、字长、操作种类和数据存储单元，指令的设置都要同等对待。
  - ✓例如，如果某机器有5种数据表示，4种字长，两种存储单元，则要设置5*4*2=40种同一操作的指令（如加法指令）。

4.5.1 指令系统设计的基本原则

- 正交性：在指令中各个不同含义的字段，如操作类型、数据类型、寻址方式字段等，在编码时应互不相关、相互独立。
- 高效率：指指令的执行速度快、使用频度高。
- 兼容性：主要是要实现向后兼容，指令系统可以增加新指令，但不能删除指令或更改指令的功能。

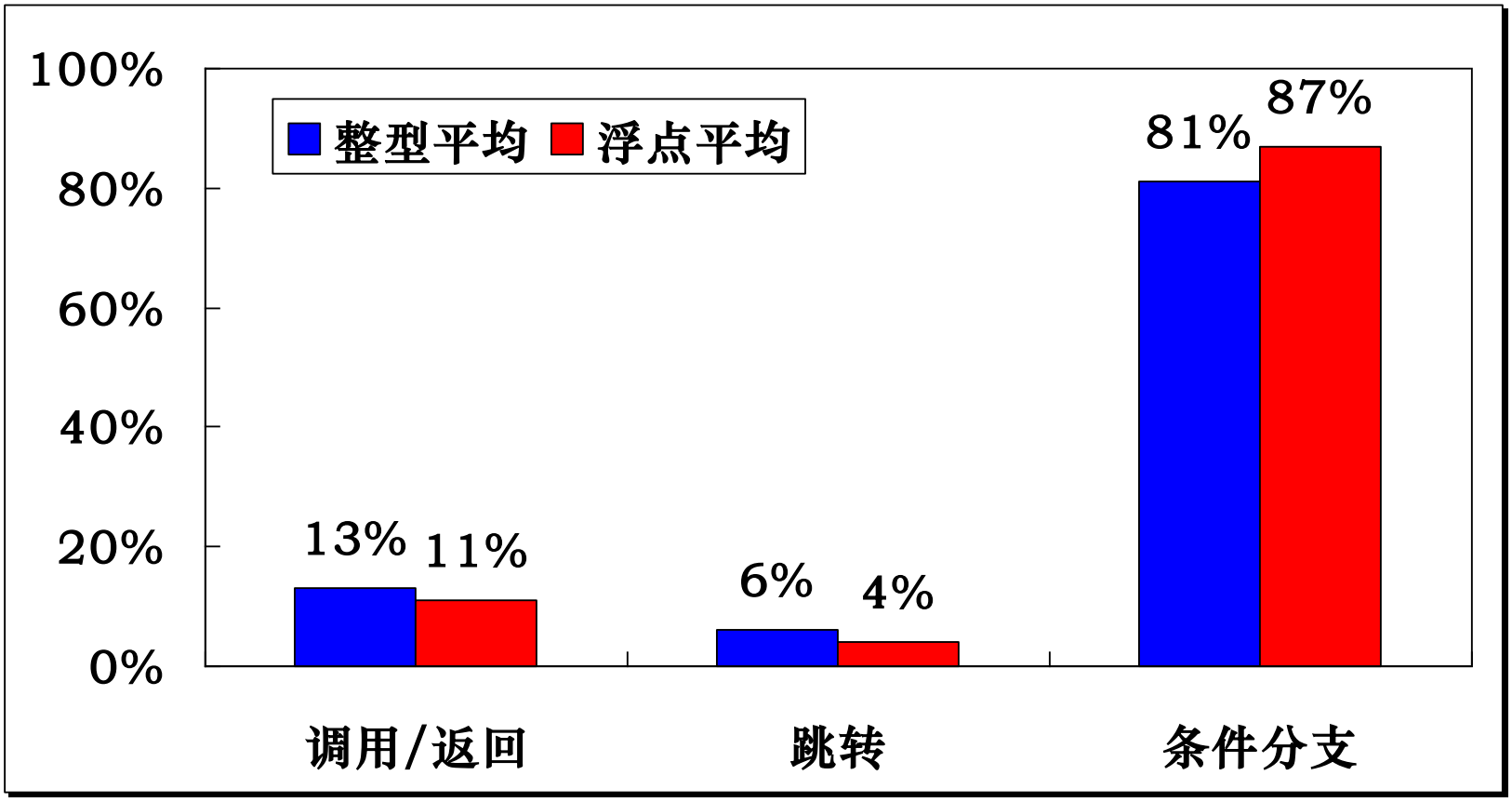
4.5.2 控制指令

- “跳转”（Jump）：当控制指令为无条件改变控制流时，我们称之为“跳转”。
- “分支”（Branch）：而当控制指令是有条件改变控制流时，我们称之为“分支”。

程序中控制流程的改变情况包括：

- 跳转（jump）；
- 条件分支（conditional branch）；
- 过程调用（call）；
- 过程返回（return）。

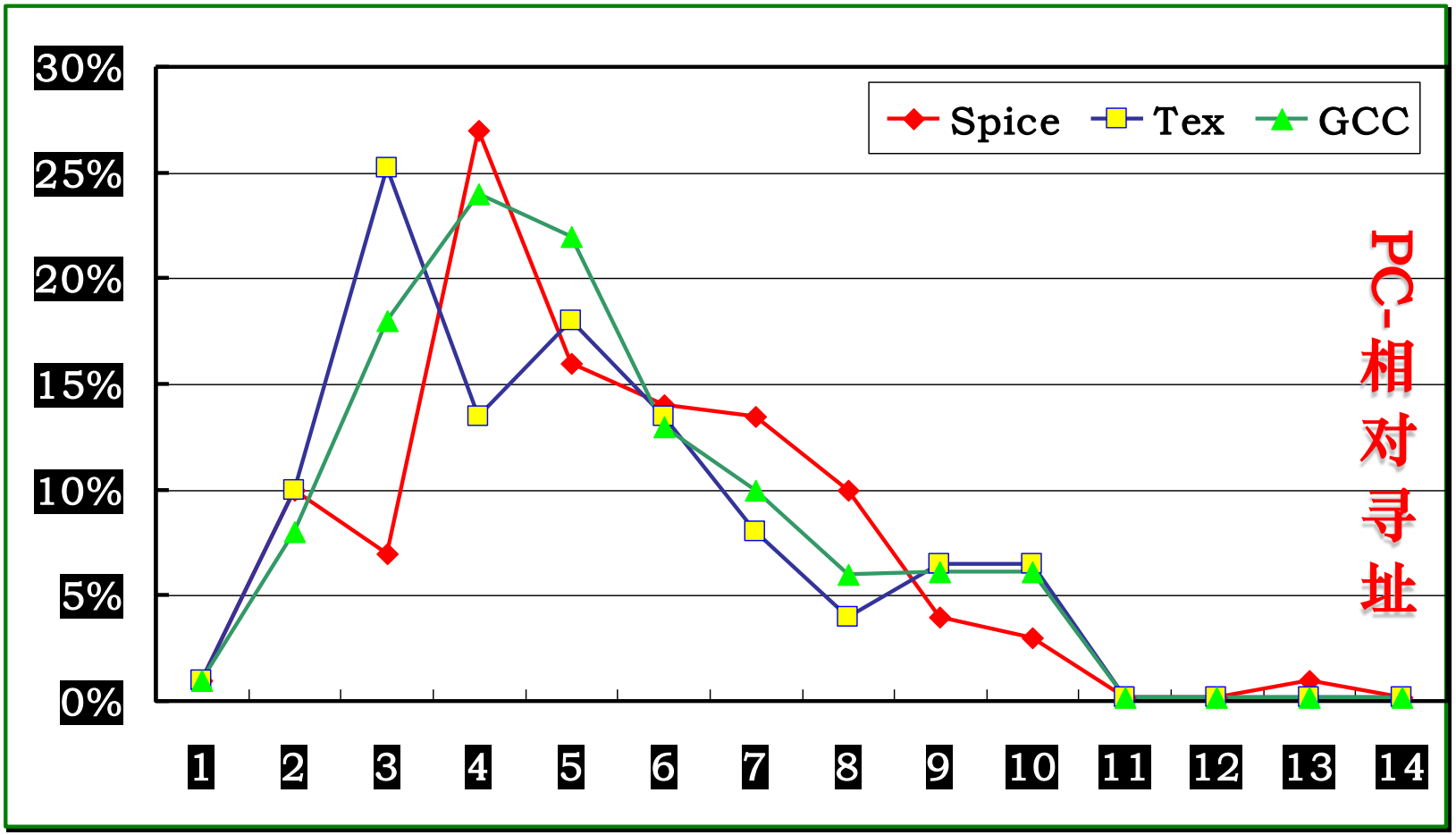
控制指令的使用频率



条件分支指令的表示

分支条件表示	优　点	缺　点
条件码（CC）：在程序的控制下，由ALU操作设置特殊的位	可以自由设置分支条件	必须从一条指令将分支条件信息传送到分支指令，所以CC是额外状态，条件码限制了指令执行顺序
条件寄存器：比较指令把比较结果放入任何一个寄存器，检测时就检测该寄存器	简单	占用了一个寄存器
比较分支：比较操作是分支指令的一部分，比较受限制	一条指令完成了两条指令的功能	分支指令的操作增多

分支目标地址的表示

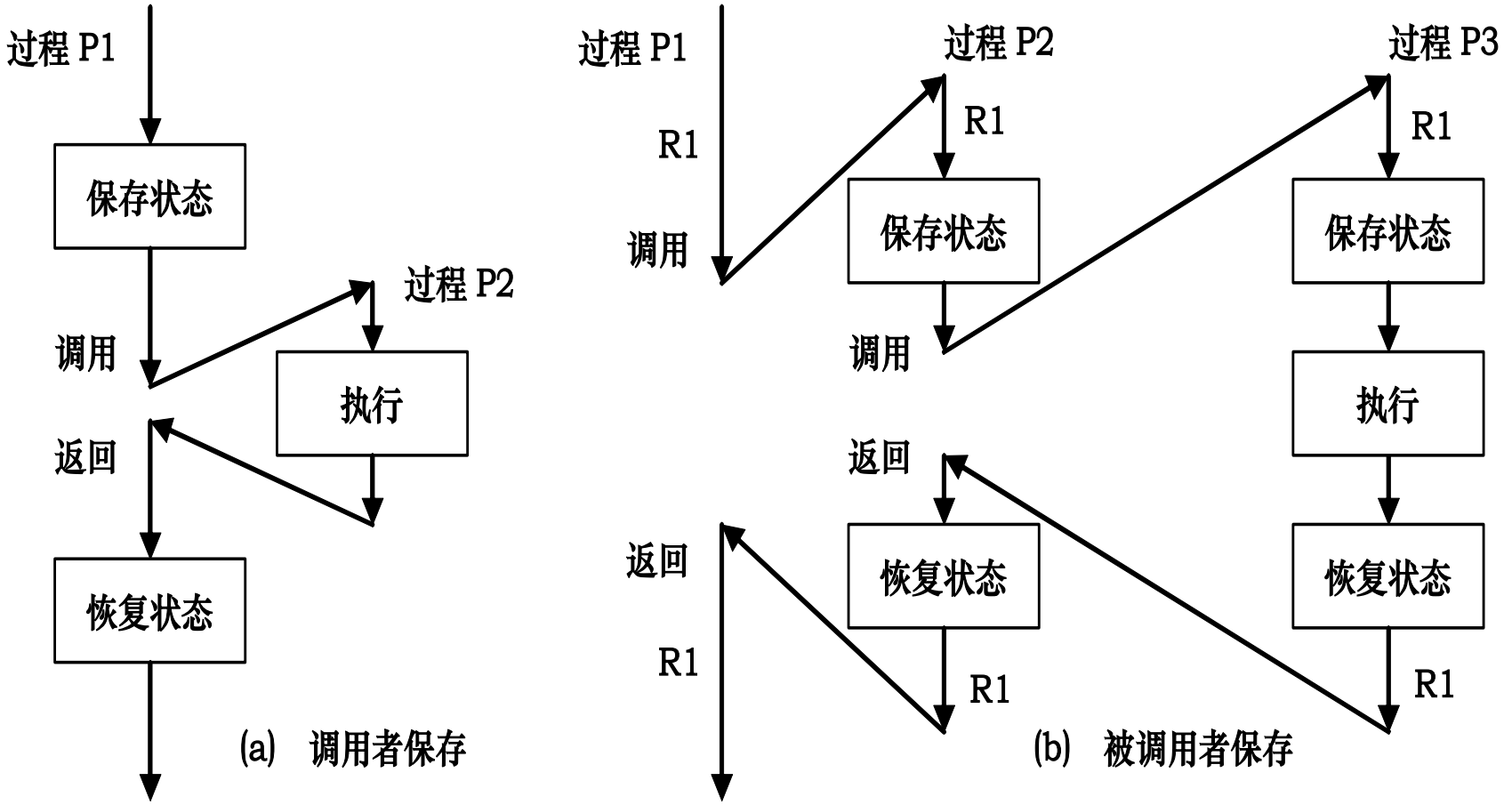


– 采用4～8位的偏移量字段（以指令字为单位）就能表示大多数控制指令的转移目标地址了

过程调用和返回的状态保存

- “调用者保存”（caller saving）方法：如果采用调用者保存策略，那么在一个调用者调用别的过程时，必须保存调用者所要保存的寄存器，以备调用结束返回后，能够再次访问调用者。
- “被调用者保存”（callee saving）方法：如果采用被调用者保存策略，那么被调用的过程必须保存它要用的寄存器，保证不会破坏过程调用者的程序执行环境，并在过程调用结束返回时，恢复这些寄存器的内容。

两种保存策略的比较


过程 P1
保存状态
过程 P2
调用
执行
返回
恢复状态
(a) 调用者保存
过程 P1
过程 P2
过程 P3
R1
R1
R1
调用
保存状态
保存状态
调用
执行
返回
返回
恢复状态
恢复状态
R1
R1
R1
(b) 被调用者保存

4.5.3 指令操作码的优化

- 指令由两部分组成：操作码、地址码
- 指令格式的设计

  确定指令字的编码方式，包括操作码字段和地址码字段的编码和表示方式。

- 指令格式的优化：如何用最短的位数来表示指令的操作信息和地址信息。

4.5.3 指令操作码的优化

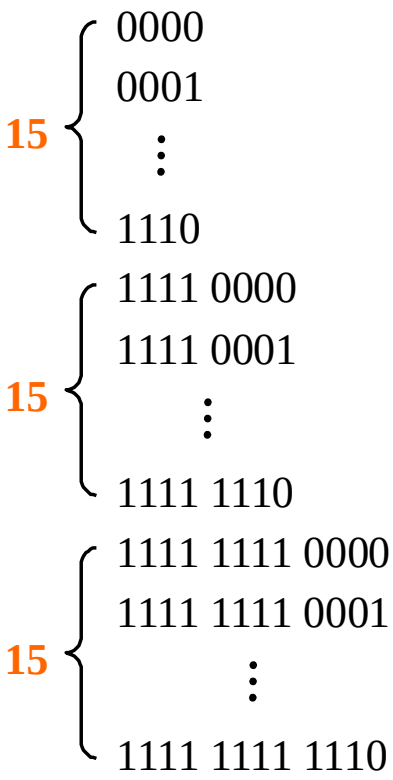
1. 等长扩展码

– 为了便于分级译码，一般都采用等长扩展码。（在早期的计算机上）

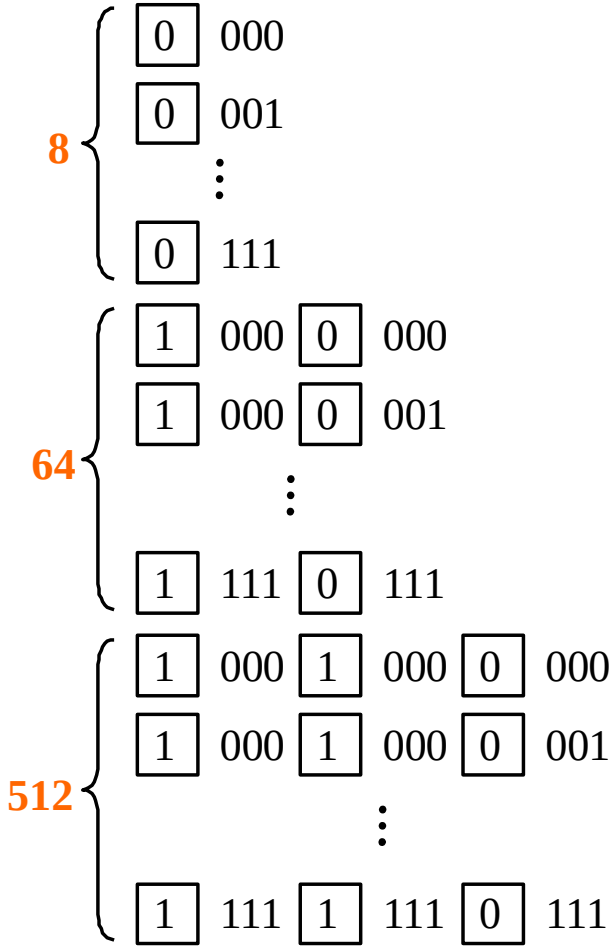
例如：15/15/15法和8/64/512法

- 选用哪种编码法取决于指令使用频度p_i的分布。若在头15种指令中p_i的值都比较大，但在后30种指令后急剧减少，则应选择15/15/15法；若p_i的值在头8种指令中较大，之后的64种指令的p_i值也不太低，则应选择8/64/512法。
- 衡量标准：看哪种编码法能使平均码长最短。

4.5.3 指令操作码的优化



15/15/15 编码法



8/64/512 编码法

4.5.3 指令操作码的优化

3. 定长操作码

- 固定长度的操作码：所有指令的操作码都是同一的长度（如8位）。

  许多计算机都采用（特别是RISC结构的计算机）

- 保证操作码的译码速度、减少译码的复杂度。
- 以程序的存储空间为代价来换取硬件实现上的好处。

4.6 指令系统的发展和改进

- 一个方向是强化指令功能，实现软件功能向硬件功能转移，基于这种指令集结构而设计实现的计算机系统称为复杂指令集计算机（CISC）。
- 八十年代发展起来的精简指令集计算机（RISC），其目的是尽可能地降低指令集结构的复杂性，以达到简化实现，提高性能的目的。

CISC指令集功能设计

- 面向目标程序增强指令功能
  - 提高运算型指令功能；
  - 提高传送指令功能；
  - 增加程序控制指令功能。
- 面向高级语言和编译程序改进指令系统
  - 增加对高级语言和编译系统支持的指令功能；
  - 高级语言计算机指令系统。

CISC指令集功能设计

- 面向操作系统的优化实现改进指令系统
  - 处理机工作状态和访问方式的切换；
  - 进程的管理和切换；
  - 存储管理和信息保护；
  - 进程的同步与互斥，信号灯的管理等。

RISC指令集功能设计

- CISC结构存在着如下缺点：
  1. 在CISC结构的指令系统中，各种指令的使用频率相差悬殊。据统计，有20％的指令使用频率最大，占运行时间的80％。也就是说，有80％的指令在20％的运行时间内才会用到。

RISC指令集功能设计

2. CISC结构指令系统的复杂性带来了计算机体系结构的复杂性。大量占用芯片面积，给VLSI设计造成很大困难。这不仅增加了研制时间和成本，而且还容易造成设计错误。
3. CISC结构的指令系统中，许多复杂指令需要很复杂的操作，因而运行速度慢。
4. 在CISC结构的指令系统中，由于各条指令的功能不均衡性，不利于采用先进的计算机体系结构技术（如流水技术）来提高系统的性能。

RISC指令集功能设计

执行频率排序	80X86指令	指令执行频率
1	Load	22%
2	条件分支	20%
3	比较	16%
4	Store	12%
5	加	8%
6	与	6%
7	减	5%
8	寄存器－寄存器间数据移动	4%
9	调用	1%
10	返回	1%
合　计		95%

RISC指令集功能设计

- 进行RISC计算机指令集结构的功能设计时，我们并不能简单地着眼于精简指令系统上，更重要的目的是使得计算机体系结构更加简单、更加合理和更加有效，克服CISC结构的缺点，使机器速度更快，程序运行时间缩短，从而提高计算机系统的性能。

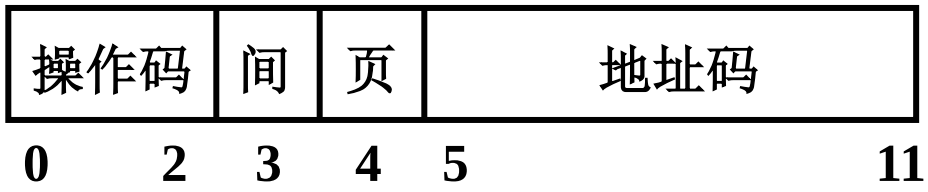
RISC指令集功能设计原则

- 选取使用频率最高的指令，并补充一些最有用的指令；
- 每条指令的功能应尽可能简单，并在一个机器周期内完成（采用流水线技术后）；
- 所有指令长度均相同；
- 只有load和store操作指令才访问存储器，其它指令操作均在寄存器之间进行；
- 大多数指令都采用硬连线技术；
- 以简单有效的方式支持高级语言；
- 充分利用流水线技术。

4.7 指令格式举例

PDP – 8　　指令字长固定 12 位

访存类指令



I/O 类指令

寄存器类指令

采用扩展操作码技术

PDP – 11

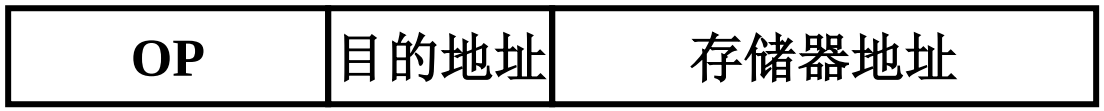
指令字长有 16 位、32 位、48 位三种

零地址 (16 位)

扩展操作码技术

一地址 (16 位)

二地址 R – R (16 位)

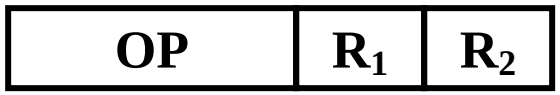


二地址 R – M (32 位)

二地址 M – M (48 位)

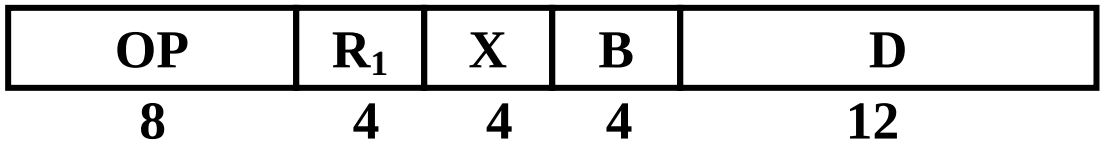
IBM 360

RR 格式



二地址 R – R

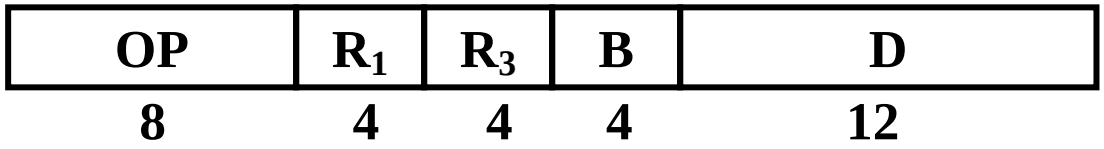
RX 格式



二地址 R – M

基址加变址寻址

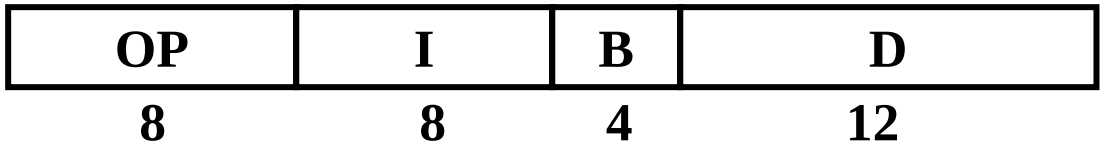
RS 格式



三地址 R – M

基址寻址

SI 格式



立即数 – M

基址寻址

SS 格式

二地址 M – M

基址寻址

Intel 8086

(1) 指令字长　1 ~ 6 个字节

INC　AX　　1 字节

MOV　WORD PTR[0204], 0138H　　6 字节

(2) 地址格式

零地址	NOP		1 字节	
一地址	CALL	段间调用	5 字节	
	CALL	段内调用	3 字节	
二地址	ADD AX，BX		2 字节	寄存器 – 寄存器
	ADD AX，3048H		3 字节	寄存器 – 立即数
	ADD AX，[3048H]		4 字节	寄存器 – 存储器

ARM

1 基本格式 <opcode>{<cond>}{S}<Rd><Rn><operand2>

<> 为必选项，{}为可选项
{} 决定指令执行条件域
{S} 决定指令执行是否影响当前程序状态寄存器CPSR的值

2 指令字长32位

cond	00	X	opcode	S	Rn	R_d	Shifter-operand
4	2	1	4	2	4	4	12

opcode：指令操作码
cond：指令执行条件
S：指令的操作是否影响CPSR的值
Rn：第一个操作数的寄存器地址
Rd：目标寄存器地址
Shifter_operand:第二个操作数