计算机体系结构--

指令系统的设计

zhaofangbupt@163.com

北京郵電大學

BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

zhaofangbupt@163.com

主要内容

- 1. 指令系统结构的分类
- 2.操作数的类型和数据表示
- 3.寻址方式
- 4. 指令系统的设计和优化
- 5. 指令系统的发展和改进
- 6. MIPS/DLX指令系统结构

zhaofangbupt@163.com

计算机指令系统设计与优化

- 指令系统是从程序设计者看到的机器的主要属性，是软、硬件的主要界面
  - 指令系统是计算机系统结构的主要组成部分
  - 指令系统是软件和硬件分界面的一个主要标志
  - 指令系统是软件和硬件之间互相沟通的桥梁
  - 指令系统与软件之间的语义差距越来越大
- 指令系统的设计
  - 指令的功能（操作类型、具体操作内容）设计
  - 指令格式的设计

zhaofangbupt@163.com

指令系统设计的基本原则

- 指令系统结构的功能设计：
  - 确定软、硬件功能分配，即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
- 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

zhaofangbupt@163.com

指令系统设计的基本原则

- 设计和确定指令系统主要应考虑的因素：
  - 有利于满足系统的基本功能
  - 有利于优化机器的性能价格比
  - 有利于指令系统今后的发展和改进
- 指令的选择和设计不是孤立的：
  - 运算类指令与数据表示的选择密切相关
  - 地址码的设计与寻址方式有着直接联系
- 指令系统的设计由编译程序设计人员与系统结构设计人员配合完成

zhaofangbupt@163.com

指令系统设计的基本原则

- 设计新指令系统的一般步骤：
  - 初拟指令的分类和具体的指令
  - 向上设计：试编出相应的编译程序
  - 模拟、测试，确定指令系统操作码及寻址方式的效能
  - 软硬件实现的选择：
    - 高频出现的指令串用硬件方式实现
    - 频度很低的指令操作改成用软件方式实现

zhaofangbupt@163.com

指令系统设计的基本原则

指令类型

- 非特权型指令
  - 供应用程序员使用，也可供系统程序员使用
    - 算术逻辑运算、数据传送、浮点运算、字符串、十进制运算、控制转移及系统控制等
- 特权型指令
  - 仅供系统程序员使用，用户无权使用
  - 用户仅由访管指令使用特权指令
    - “启动I/O”、停机等待、存储管理保护、控制系统状态、诊断等子类

zhaofangbupt@163.com

指令系统设计的基本原则

- 指令系统设计的原则
  - 支持编译系统高效、简易地将源程序翻译成目标代码
- 对指令系统的基本要求
  - 完整性
  - 规整性（包括对称性和匀称性）
  - 正交性
  - 高效率
  - 兼容性

zhaofangbupt@163.com

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，指令集所提供的指令足够用
  - 要求：
    - 指令集功能齐全
    - 使用方便

zhaofangbupt@163.com

指令系统设计的基本原则

操作类型	实　例
算术和逻辑运算	算术运算和逻辑操作：加，减，乘，除，与，或等
数据传输	load，store
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，减，乘，除，比较等
十进制	十进制加，十进制乘，十进制到字符的转换等
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

zhaofangbupt@163.com

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**：所有与指令集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
  - **均匀性**：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

zhaofangbupt@163.com

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、寻址方式等，在编码时应互不相关，相互独立
- **高效率**：
  - 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，但不能删除指令或者更改指令的功能

zhaofangbupt@163.com

指令系统设计的基本原则

- 站着系统结构设计者角度还希望：
  - 指令码密度适中
    - 高密度指令：强功能复合指令
    - 优点：减少程序长度、访存次数、Cache、虚存访问调度次数、程序运行时间
    - 缺点：指令系统复杂，硬件实现困难
  - 适应性
    - 当工艺技术发展变化时，指令系统仍可以方便地用硬件来实现也能方便实现

zhaofangbupt@163.com

指令系统设计的基本原则

- 设计原则总结：
  - 要有利于满足系统的基本功能，有利于优化机器的性能价格比，有利于指令系统今后的发展和改进
  - 要考虑非特权型和特权型指令
  - 设计新指令系统要按一般步骤进行
  - 运算类指令的功能要与数据表示的选择密切相关
  - 指令中地址码的设计要与寻址方式有着直接联系
  - 同时协调兼顾编译程序设计者和系统结构设计者两者的要求

zhaofangbupt@163.com

指令系统设计的基本原则

- 在设计指令集结构时，有两种截然不同的设计策略，产生了两类不同的计算机系统：
  - CISC（复杂指令集计算机）
    - 增强指令功能，把越来越多的功能交由硬件来实现，并且指令的数量也是越来越多
  - RISC（精简指令集计算机）
    - 尽可能地把指令集简化，不仅指令的条数少，而且指令的功能也比较简单

zhaofangbupt@163.com

指令系统的设计

- 指令系统的设计包含的内容：
  - 指令操作功能
    - 指令的功能设计
  - 指令的类型
  - 指令的格式
    - 指令的操作码和地址码
  - 操作数的访问方式
    - 寻址方式

zhaofangbupt@163.com

指令系统的功能设计

- 通用计算机系统的5类基本指令
  - 数据传送类指令
  - 运算类指令
  - 程序控制指令
  - 输入输出指令
  - 处理机控制和调试指令

zhaofangbupt@163.com

指令系统的功能设计

- 数据传送类指令
  - 由如下三个主要因素决定：
    - 数据存储设备的种类：M、R、堆栈
    - 数据单位：字、字节、数据块等
    - 采用的寻址方式　4-20种
  - 指令种类：有8种
    - 通用寄存器 ⇒ 通用寄存器/主存储器/堆栈
    - 主存储器 ⇒ 通用寄存器/主存储器/堆栈
    - 堆栈 ⇒ 通用寄存器/主存储器

zhaofangbupt@163.com

指令系统的功能设计

- 数据运算类指令：
  - 考虑四个因素的组合：
    - 操作种类：加、减、乘、除、与、或、非、异或、比较、移位、检索、转换、匹配、清除、置位等
    - 数据表示：定点、浮点、逻辑、十进制、字符串、定点向量等
    - 数据长度：字、双字、半字、字节、位、数据块等
    - 数据存储设备：通用寄存器、主存储器、堆栈等

zhaofangbupt@163.com

指令系统的功能设计

- 加法类指令举例：
  - 寄存器-寄存器型的定点单字长加法指令
  - 寄存器-寄存器型的定点双字长加法指令
  - 寄存器-寄存器型的定点半字加法指令
  - 寄存器-寄存器型的字节加法指令
  - 寄存器-寄存器型的浮点单字长加法指令
  - 寄存器-寄存器型的浮点双字长加法指令
  - 寄存器-寄存器型的单字长逻辑加法指令
  - 寄存器-寄存器型的定点向量加法指令
  - 寄存器-寄存器型的浮点向量加法指令
- 考虑了存储设备，还有RS、SS型、堆栈型等

zhaofangbupt@163.com

指令系统的功能设计

- 对于移位指令，要组合以下三个因素：
  - 移位方向：左移(L)、右移(R)
  - 移位种类：算术移位(A)、逻辑移位(L)、循环移位(R)
  - 移位长度：单字长(S)、双字长(D)
  - 组合起来共有：3×2×2＝12种，清除相同指令，应该10种，分别是：
    - SLAS(SRAS) 单字长算术左移/右移
    - SLLS(SRLS) 单字长逻辑左移，或单字长算术左移
    - SLRS(SRRS) 单字长循环左移/右移

zhaofangbupt@163.com

指令系统的功能设计

- SLAD(SRAD) 双字长算术左移/右移
- SLLD(SRLD) 双字长逻辑左移，或双字长算术左移
- SLRD(SRRD)双字长循环左移/右移

➢ 一般有两个操作数，第二个给出移位的位数

□ 位操作指令：置位、清位、位测试、找位等

□ 字符串指令：比较、查找、匹配、转换等

zhaofangbupt@163.com

指令系统的功能设计

- 程序控制类指令
  - 控制指令：用来改变控制流
    - 跳转：无条件改变控制流时，称之为跳转指令
    - 分支：有条件改变控制流时，则称之为分支指令
  - 能够改变控制流的指令：
    - 分支
    - 跳转
    - 过程调用
    - 过程返回

zhaofangbupt@163.com

指令系统的功能设计

- 程序控制指令的使用频度：
  - 针对load-store型指令集结构的机器，基准程序为SPEC CPU2000

指令类型	使用频度	
	整型平均	浮点平均
调用/返回	19%	8%
跳转	6%	10%
分支	75%	82%

zhaofangbupt@163.com

指令系统的功能设计

□常用的3种表示分支条件的方法及其优缺点：

名 称	检测分支条件的方法	优 点	缺 点
条件码（CC）	检测由ALU操作设置的一些特殊的位（即CC）	可以自由设置分支条件	条件码是增设的状态。而且它限制了指令的执行顺序，因为要保证条件码能顺利地传送给分支指令
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
比较与分支	比较操作是分支指令的一部分，通常这种比较是受到一定限制的	用一条指令（而不是两条）就能实现分支	当采用流水方式时，该指令的操作可能太多，在一拍内做不完

zhaofangbupt@163.com

指令系统的功能设计

转移目标地址的表示

- 最常用的方法
  - 在指令中提供一个偏移量，由该偏移量和程序计数器（PC）的值相加而得出目标地址（PC相对寻址）
- 优点
  - 有效地减少表示该目标地址所需要的位数
  - 位置无关（代码可被装载到主存的任意位置执行）
- 关键：确定偏移量字段的长度
  - 模拟结果表明：采用4～8位的偏移量字段

zhaofangbupt@163.com

指令系统的功能设计

- 程序过程调用和返回指令：
  - 除了要改变控制流之外，还要保存机器状态
  - 至少也得保存返回地址（放在专用的链接寄存器或堆栈中）
    - 过去有些指令系统结构提供了专门的保存机制来保存许多寄存器的内容
    - 现在较新的指令系统结构则要求由编译器生成load和store指令来保存或恢复寄存器的内容

zhaofangbupt@163.com

指令系统的功能设计

- 程序控制类指令举例：
  - BEQ(BNEQ)　等于/不等于零转移
  - BLS(BGT)　小于/大于转移
  - BLEQ(BGEQ) 小于/大于等于转移，或不大/小于转移
  - BLSU(BGTU) 不带符号小于/大于转移
  - BLEQU(BGEQU)　不带符号小于/大于等于转移，或不带符号不大于/不小于转移
  - BCC(BCS)　没有进位/有进位转移
  - BVC(BVS)　没有溢出/有溢出转移

zhaofangbupt@163.com

指令系统的功能设计

- 程序过程调用和返回指令举例：
  - CALL　　转入子程序
  - RETURN　　从子程序返回，本身可以带有条件
- 中断控制指令：
  - 开中断、关中断
  - 改变屏蔽
  - 中断返回
  - 自陷等

zhaofangbupt@163.com

指令系统的功能设计

- 输入输出指令：
  - 启动、停止、测试、控制设备，数据输入/输出操作等
    - 采用单一的直接寻址方式
  - 在多用户或多任务环境下，输入输出指令属于特权指令
  - 也可以不设置输入输出指令，输入输出设备与主存储器共用同一个零地址空间

zhaofangbupt@163.com

指令系统的功能设计

- 处理机控制和调试指令：
  - 处理机状态切换指令
    - 处理机至少有两个或两个以上状态：管态、用户态
  - 硬件和软件的调试指令
    - 硬件调试指令：钥匙位置、开关状态的读取寄存器和主存单元的显示等
    - 软件调试指令：断点的设置、跟踪，自陷井指令等

zhaofangbupt@163.com

指令的组成

- 指令 = 操作码 + 地址码
  - 操作码主要包括两部分内容：
    - 操作种类
    - 操作数描述
      - 数据的类型、进位制、数据字长
  - 地址码通常包括三部分内容：
    - 操作对象的地址
    - 地址的附加信息
    - 寻址方式

zhaofangbupt@163.com

指令格式的优化

- 确定指令字编码方式
  - 如何用最短的位数表示指令的操作信息和地址信息
  - 主要目标：
    - 节省程序存储空间
    - 指令格式尽量规整，简化译码
    - 优化后执行速度不降低
  - 包括：
    - 操作码的优化
    - 地址码的优化

zhaofangbupt@163.com

操作码的优化

- 操作码的三种编码方法：
  - 固定长度：
    - 优点：规整性好，解码简单
    - 缺点：空间大，没有体现各指令的特点
  - Huffman编码：
    - 优点：空间小
    - 缺点：规整性不好，解码复杂
  - 扩展编码：
    - 折衷方案

zhaofangbupt@163.com

操作码的优化

- 不同编码方式对存储空间的影响
  - 改进操作码编码方式能够节省程序存储空间
    - 例如：Burroughs公司的B-1700机

操作码 编码方式	整个操作系统所用 指令的操作码总位数	改进的 百分比
8位定长编码	301,248	0
4-6-10扩展编码	184,966	39%
Huffman编码	172,346	43%

zhaofangbupt@163.com

操作码的优化

定长操作码

- 定长编码是指所有指令的操作码长度都是相等的
- 如果需要编码的操作码有n个，那么，定长操作码的位数最少需要 $\lceil \log_2 n \rceil$ 位
  - 例如：用指令字中第一个字节（8位）表示操作码
- 优点：简化硬件设计和减少指令译码时间
- 缺点：浪费了信息量
- RISC（精简指令系统计算机）采用这种编码方法

zhaofangbupt@163.com

操作码的优化

哈夫曼（Huffman）编码

- 霍夫曼压缩概念的基本思想：
  - 当各种事件发生的概率不均等时，采用优化技术对发生概率最高的事件用最短的位数（时间）来表示（处理），而对出现概率较低的允许用较长的位数（时间）来表示（处理），以达到平均位数减少的目的
- 用于代码压缩、程序压缩、空间压缩和时间压缩
- 需要知道每种指令在程序中出现的概率（使用频度）

zhaofangbupt@163.com

操作码的优化

【例1】某模型机指令使用频度举例

指令	指令使用频度pi	指令	指令使用频度pi
I1	0.40	I5	0.04
I2	0.30	I6	0.03
I3	0.15	I7	0.03
I4	0.05		

- **信息源熵**：信息源包含的平均信息量

$$H = -\sum p_i \log_2 p_i \qquad H = -\sum_{i=1}^{7} p_i \log_2 p_i = 2.17$$

zhaofangbupt@163.com

操作码的优化

- 各种编码的平均码长和信息冗余：
  - 编码优化的程度用平均码长来评价，平均码长定义为：

$$l = \sum_{i=1}^{n} p_i l_i$$

  - 信息冗余量：$\dfrac{\text{操作码的实际平均长度} - H}{\text{操作码的实际平均长度}}$
  - 定长码表示：需 $\lceil \log_2 n \rceil = 3$ 位
    - 定长码信息冗余量：

$$\frac{\text{实际平均码长} - H}{\text{实际平均码长}} = \frac{3 - 2.17}{3} \approx 28\%$$

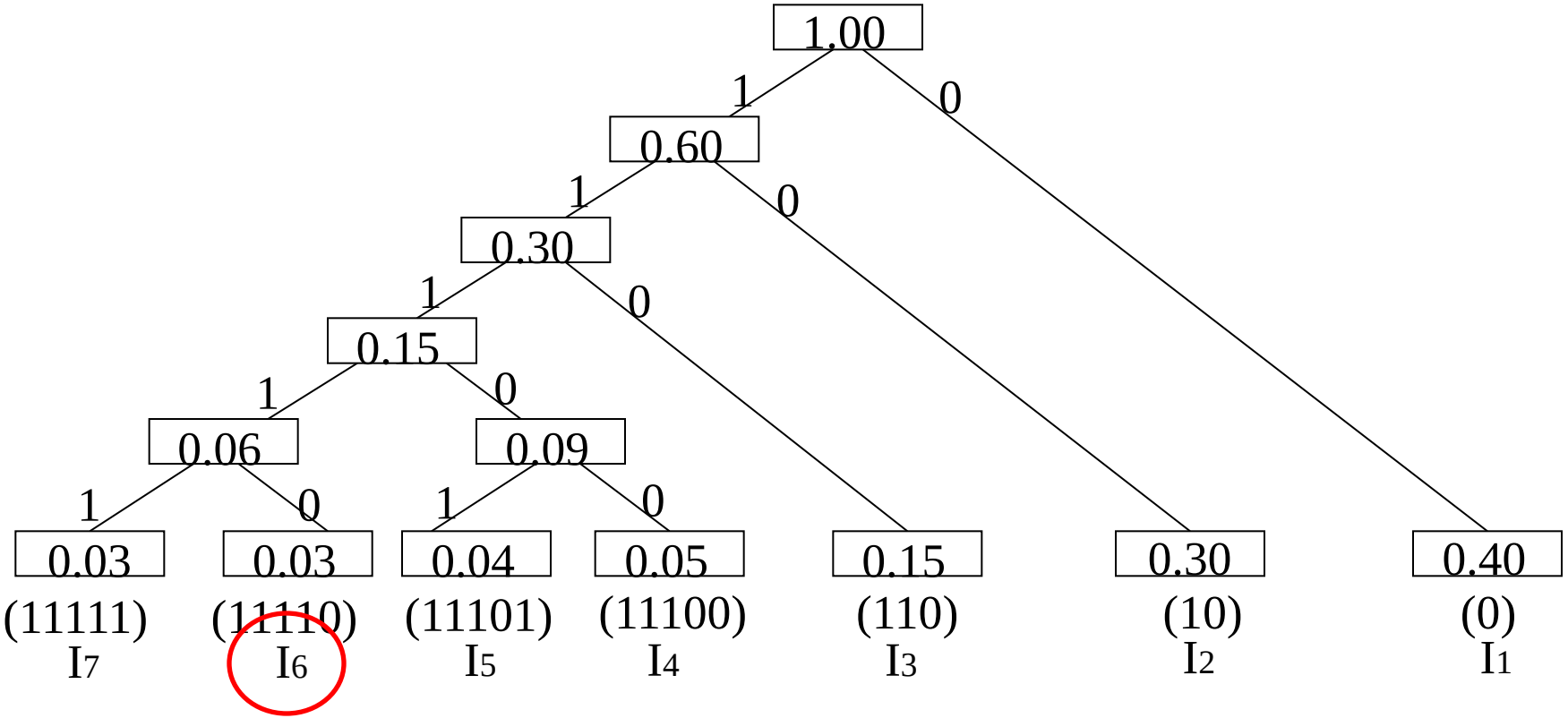
zhaofangbupt@163.com

操作码的优化

- 霍夫曼编码方法：
  - 霍夫曼树的构建：
    - 从小到大排序
    - 最小两个合并
    - 重复上述过程
    - 只剩一个结束
  - 添加霍夫曼编码方法：
    - 根结点开始，左边分支编"1"，右边分支编"0
    - 哈夫曼树不是唯一的

zhaofangbupt@163.com

操作码的优化



zhaofangbupt@163.com

操作码的优化

- 最终的结果
  - Huffman用四种长度：1位、2位、3位、5位
    - 0，10，110，11100，11101，11110，11111
  - 信息熵计算结果：

$$平均码长 = \sum_{i=1}^{7} p_i * l_i = 2.20位$$

$$信息余量 = 0.0136 = 1.36\%$$

$$长度个数 = 4$$

zhaofangbupt@163.com

操作码的优化

- Huffman操作码的主要缺点：
  - 操作码长度很不规整，硬件译码困难
  - 与地址码共同组成固定长的指令比较困难
- 扩展编码法：
  - 由固定长操作码与Huffman编码法相结合形成
    - 减少平均长度
    - 方便译码

zhaofangbupt@163.com

操作码的优化

- 操作码的哈夫曼编码及扩展操作码编码

指令	频度Pi	Op哈夫曼编码	OP长度l_i	扩展操作码	OP长度l_i
I1	0.40	0	1	00	2
I2	0.30	10	2	01	2
I3	0.15	110	3	10	2
I4	0.05	11100	5	1100	4
I5	0.04	11101	5	1101	4
I6	0.03	11110	5	1110	4
I7	0.03	11111	5	1111	4

zhaofangbupt@163.com

操作码的优化

- 扩展操作码编码设计
  - 界于定长二进制编码和霍夫曼编码间的编码方式
  - 操作码长度不是定长的，但只有有限几种码长
  - 利用高概率的用短码、低概率的用长码表示的霍夫曼压缩思想，然后分别用定长码来实现
    - 使操作码平均长度缩短，以降低信息冗余
  - 利用霍夫曼编码中短码不可能是长码的前缀，从而保证了解码的惟一性和实时性

zhaofangbupt@163.com

操作码的优化

- 实现性能比较

编码方式	整个操作系统所用的操作码总位数	改进百分比%
定长8位	301 248位	0
4-6-10位	184 966位	39
霍夫曼法	172 346位	43

- 实现方式：
  - 等长扩展码
    - 4-8-12位等长扩展
    - 8/64/512扩展法

操作码的优化–等长扩展

操作码编码	说明
0000 0001 …… 0111	4位长度的操作码 共8种
1000 0000 1000 0001 …… 1111 0111	8位长度的操作码 共64种
1000 1000 0000 1000 1000 0001 …… 1111 1111 0111	12位长度的操作码 共512种

zhaofangbupt@163.com

操作码的优化

操作码编码	说　明
0000 0001 …… 1110	4 位长度的 操作码共 15 种
1111 0000 1111 0001 …… 1111 1110	8 位长度的 操作码共 15 种
1111 1111 0000 1111 1111 0001 …… 1111 1111 1110	12 位长度的 操作码共 16 种

等长 15/15/15……扩展法

操作码编码	说　明
0000 0001 …… 0111	4 位长度的 操作码共 8 种
1000 0000 1000 0001 …… 1111 0111	8 位长度的 操作码共 64 种
1000 1000 0000 1000 1000 0001 …… 1111 1111 0111	12 位长度的操 作码共 512 种

等长 8/64/512……扩展法

zhaofangbupt@163.com

操作码的优化

编码方式的选择

- 选用哪种编码方法取决于指令使用频度Pi的分布
  - 若Pi值在头15种指令中都比较大，但在30种指令后急剧减少，则宜选15/15/15法
  - 若Pi值在头8种指令中较大，之后的64种指令的pi值也不太低时，则宜选8/64/512法
- 衡量标准是看哪种编码法能使平均码长∑PiLi最短
- 扩展标志不同，还可有其他许多种扩展方案

zhaofangbupt@163.com

操作码的优化

- 扩展操作码的总结：
  - 扩展操作码必须遵守短码不能是长码的前缀
    - 扩展操作码的编码不惟一，平均码长也不惟一
  - 因此，需要对各扩展方案进行比较：
    - 找出一种平均码长尽可能短
    - 码长种类数不能过多，从而便于优化实现

zhaofangbupt@163.com

操作码的优化

- 定长操作码
  - 前提：
    - 随着计算机存储器空间的日益加大
    - 保证操作码的译码速度、减少译码的复杂度
  - 许多计算机（特别是RISC结构）都采用了固定长度的操作码
  - 以持续的存储空间为代价来换取硬件实现上的好处

zhaofangbupt@163.com

地址码的优化

- 地址码只需要给出操作数和结果的地址
  - 地址码的长度：
    - 地址码个数
    - 寻址方式
    - 存储设备（R、M和堆栈等）的编址方式
    - 寻址空间大小

zhaofangbupt@163.com

地址码的优化

地址码个数

- 指令的地址码个数：三个地址、二个地址、一个地址及零地址
  - 三地址指令格式：

OP	A_1	A_2	A_3

  - 二地址指令格式：

OP	A_1	A_2

zhaofangbupt@163.com

地址码的优化

➢ 一地址指令格式：

OP	A

- 只有目的操作数的单操作数指令：指令中只给出一个目的地址A，A既是操作数的地址，又是操作结果的存放地址
- 隐含约定目的地址的双操作数指令：源操作数按指令给出的源地址A读取，另一个操作数隐含在CPU的累加器AC中，运算结果也将存放在AC中

zhaofangbupt@163.com

地址码的优化

- 零地址指令格式：

OP

  - 不需要操作数的指令：空操作指令、停机指令等
  - 所需操作数是隐含指定的：计算机中对堆栈操作的运算指令

- 指令格式中采用隐地址能有效减少地址数，缩短了指令码的长度
  - 会增加指令译码的复杂性
- 上述指令格式是一般情况，并非每台计算机都有

zhaofangbupt@163.com

指令字格式的优化

不同地址个数指令的特点及适用场合

地址数目	程序的长度	程序存储量	程序执行速度	适用场合
三地址	最短	最大	一般	向量，矩阵运算为主
二地址	较短	很大	很低	一般不宜采用
一地址	较长	较大	较快	连续运算，硬件结构简单
零地址	最长	最小	最低	嵌套，递归，变量较多
二地址寄存器型	一般	最小	最快	多累加器，数据传送较多

zhaofangbupt@163.com

地址码的优化

- 缩短地址码的方法
  - 虚拟存储器系统的采用：
    - 采用寄存器直接寻址，或寄存器间接寻址均可缩短地址码长度
    - 采用基址或变址寻址方式缩短地址码长
    - 采用存储器间接寻址方式缩短地址长度

zhaofangbupt@163.com

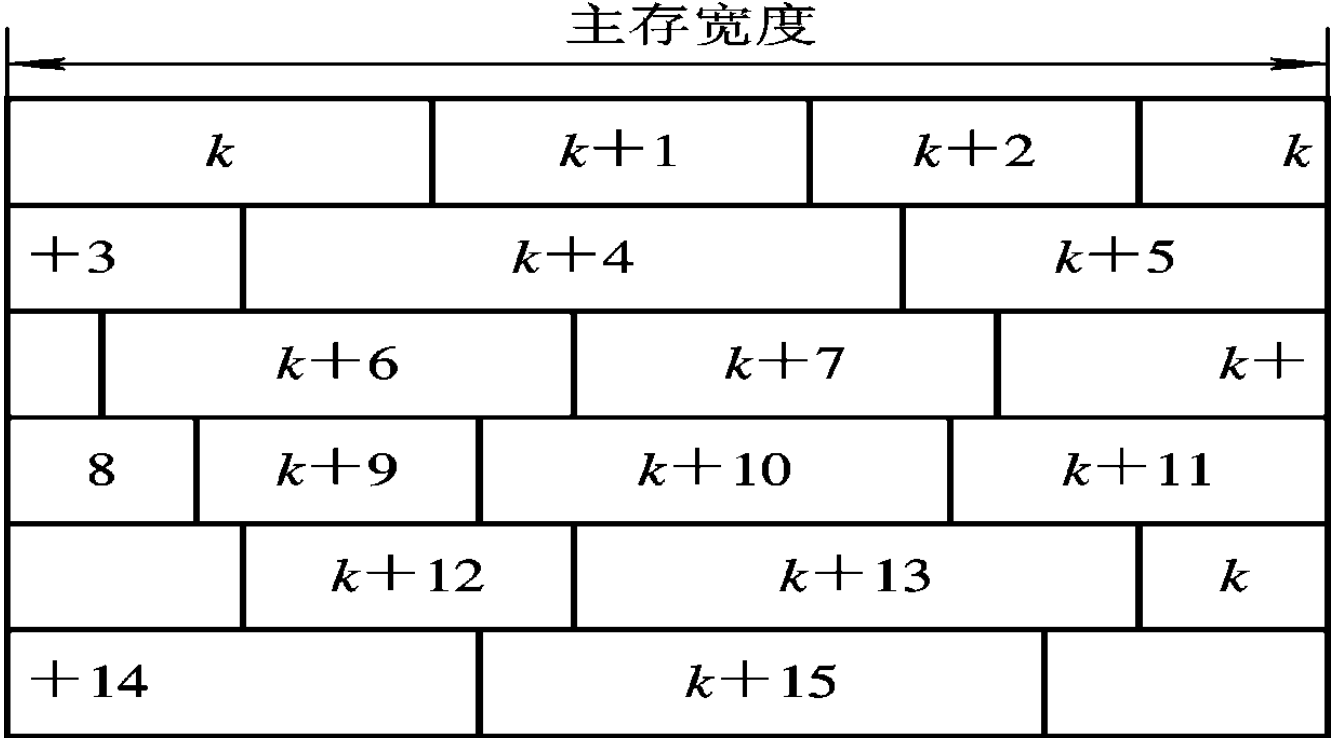
指令字格式的优化

- 确定操作码字段和地址码字段的大小及其组合形式，以及各种寻址方式的编码方法
  - 目标：
    - 缩短指令字，程序总位数才会减少
  - 设计原则：
    - 尽可能地增加寄存器数目和寻址方式类型
    - 充分考虑寄存器字段和寻址方式字段对指令平均字长的影响，以及它们对目标代码大小的影响
    - 设计出的指令字格式能够在具体实现中容易处理

zhaofangbupt@163.com

指令字格式的优化

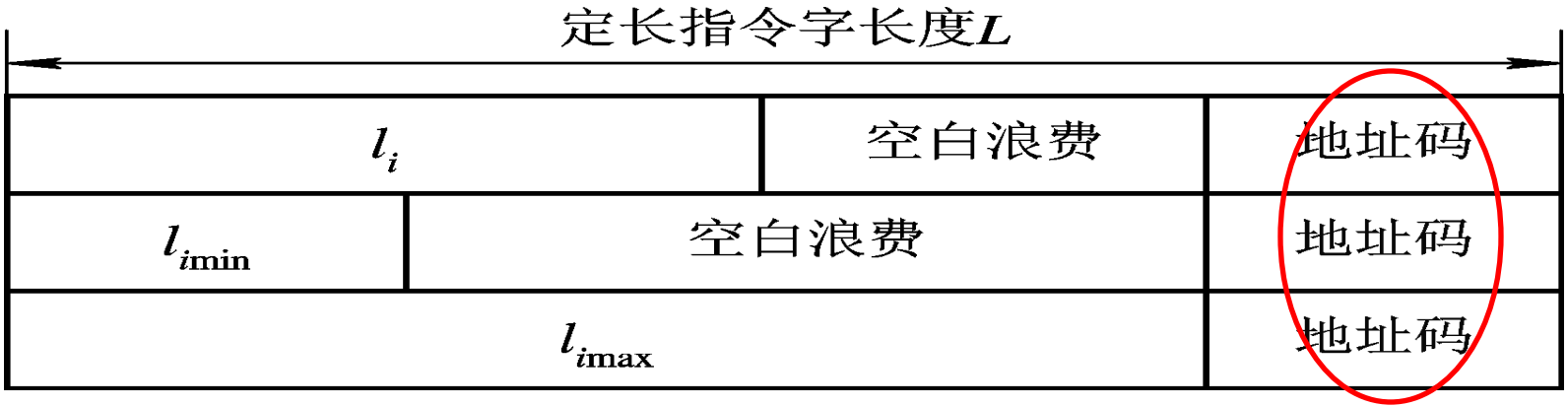
- 任意长指令字在按位编址主存中存贮的情况



zhaofangbupt@163.com

指令字格式的优化

- 等长地址码情况：

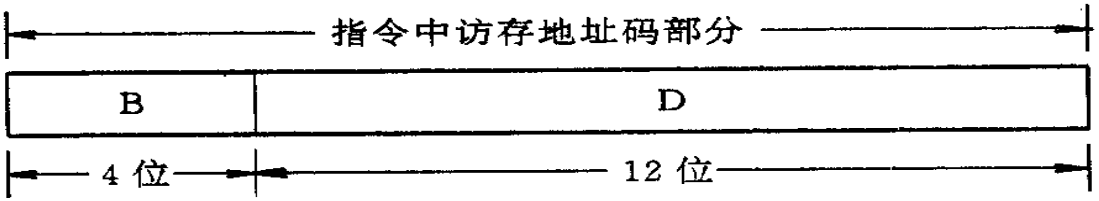


- 等长地址码发挥不出操作码优化表示的作用

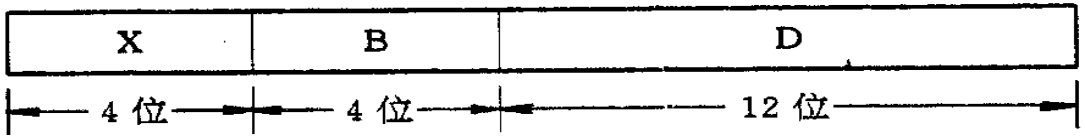
zhaofangbupt@163.com

指令字格式的优化

- 缩短指令中地址码的方式：
  - 采用基址寻址形式



  - 在指令中同时采用基址和变址形式



  - 访存地址空间分段管理

段　号	段内地址

zhaofangbupt@163.com

指令字格式的优化

- 采用地址个数可变和/或地址码长度可变
  - 同一种地址采用多种地址形式和长度
  - 用空白处存放直接操作数或常数

寄存器-寄存器型	操作码		R	R
寄存器-存储器型	操作码	访存地址S		R
带直接操作数	操作码	直接操作数	R	R

zhaofangbupt@163.com

指令字格式的优化

- 寻址方式的表示方法
  - 与操作码一起编码方法
    - 适合：处理机采用load-store结构，寻址方式只有很少几种
  - 设置专门的地址描述符，由地址描述符表示相应操作数的寻址方式
    - 适合：处理机具有多种寻址方式，且指令有多个操作数

zhaofangbupt@163.com

指令字格式的优化

- 指令字格式优化需要考虑因素
  - 机器中寄存器的个数和寻址方式的数目对指令平均字长的影响以及它们对目标代码大小的影响
  - 所设计的指令格式便于硬件处理，特别是流水实现
  - 指令字长应该是字节（8位）的整数倍，而不能是随意的位数
- 指令集的3种编码格式
  - 变长编码格式、定长编码格式、混合型编码格式

zhaofangbupt@163.com

指令字格式的优化

变长编码格式

- 优点：
  - 当指令系统的寻址方式和操作种类很多时，这种编码格式是最好的
  - 用最少的二进制位来表示目标代码
- 缺点：
  - 可能会使各条指令的字长和执行时间相差很大

操作码	地址描述符 1	地址码 1	…	地址描述符 n	地址码 n

zhaofangbupt@163.com

指令字格式的优化

定长编码格式

- 将操作类型和寻址方式一起编码到操作码中
- 当寻址方式和操作类型非常少时，这种编码格式非常好
- 可以有效地降低译码的复杂度，提高译码的速度
- 大部分RISC的指令集均采用这种编码格式

操作码	地址码 1	地址码 2	地址码 3

zhaofangbupt@163.com

指令字格式的优化

混合型编码格式

- 提供若干种固定的指令字长
- 以期达到既能够减少目标代码长度又能降低译码复杂度的目标

操作码	地址描述符	地址码

操作码	地址描述符 1	地址描述符 2	地址码

操作码	地址描述符	地址码 1	地址码 2

zhaofangbupt@163.com

指令字格式的优化

- 指令字格式优化的措施概括如下：
  - 采用扩展操作码，根据指令频度分布选择编码，以缩短操作码平均码长
  - 采用多种寻址方式，以缩短地址码的长度，在有限地址长度内提供更多的地址信息
  - 采用多种地址制，以增强指令的功能，缩短程序长度，加快程序的执行速度
  - 在同种地址制内再采用多种地址形式
  - 维持指令字按整数边界存储，使用多种指令字长度

zhaofangbupt@163.com

内容小结

- 指令系统的设计与优化
  - 指令系统设计的原则
    - 指令系统设计的基本要求
  - 指令系统的功能设计
    - 控制指令功能性设计（重点）
  - 指令操作码的优化（重点）
    - 各种编码方式及其优缺点
  - 指令字格式的设计及优化（重点）
    - 地址码的优化及指令格式

zhaofangbupt@163.com

内容小结

□知识要点

完整性	规整性	正交性
定长编码	霍夫曼编码	扩展编码
可变长度指令编码	固定长度指令编码	混合型指令编码

zhaofangbupt@163.com

练习题

- 1. 指令集结构设计所涉及的内容有哪些？
- 2. 为了对编译器设计提供支持，在进行指令系统设计时，应考虑哪些问题？
- 3. 通常有哪几种指令格式？简述其适用范围。

Thank You !

zhaofangbupt@163.com

北京郵電大學

BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS