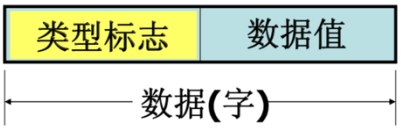
Chapter2 数据表示与指令系统

计算：信息熵、平均码长、操作码编码（定长、哈夫曼、等长）
重点：
自定义数据表示
指令操作码编码（计算）
指令格式优化方法
指令系统的改进

2.1 数据表示

计算机常用三类数据：用户定义的数据、系统数据、指令数据
计算机系统结构研究的首要问题：在所有的数据类型中，哪些用硬件实现，哪些用软件实现，并研究他们的实现方法。
数据表示：能够被硬件直接识别和指令系统直接调用的数据类型
数据表示以外的数据类型，一般都是**数据结构**要研究的内容



数据表示与数据结构的关系：
参考前面的定义
数据结构是通过软件映像成机器所具有的各种数据表示实现
-数据结构通过软件变成数据 表示
数据表示是数据结构的成员
数据表示为数据结构提供不同程度的支持
数据结构和数据表示都是数据结构类型的子集
首要问题实际上是软硬件的取舍问题

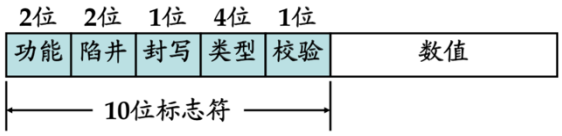


图 在R2巨型机中带标志符的数据表示方法

高级数据表示：
***自定义数据表示：**数据本身表明数据类型
*目的：*缩短机器语言与高级语言对于数据属性的说明之间的语义差距
带标志符的数据表示：高级语言中用说明语句 如 int，
机器语言中用操作码指明操作数，带符号加法，无符号加法
编译程序负责将高级语言的说明转化为机器语言指令操作码

机器中设置带标志符的数据表示，标志符对于程序员透明
优缺点：提高操作码通用性，便于硬件检查和完成数据转换，简化编译程序
/可能会使程序占用空间增加，单指令的执行速度会降低。
应用广泛

数据描述符：支持向量、数组、记录等数据，减少标识符所占的空间->一个数组只要一个类型说明

描述符和数据分开存放，访问数据必须先访问描述符，描述符被看作程序一部分
*优点：*描述符树形链接，能描述复杂数据结构，为向量、数组等数据结构提供一定支持，简化编译，比变址法更快形成元素地址。

***向量数组数据表示：**

向量：n 个数据的数组

特点：每个数据叫数组的元素，每个数据类型、数据表示、操作都是相同的，各数据之间独立无关

*优点：*快速形成元素地址、数据块预取，一条指令实现整个向量处理，节省存储

***堆栈数据表示：**堆栈数据结构在编译和子程序调用中经常用到，大小机器基本都有堆栈数据表示。

通用寄存器机型对于堆栈数据结构的支持比较差：

操作机器指令少，功能单一
堆栈在存储器中，访问速度慢
通常用来保存返回地址，少量寄存器用来传参数

堆栈机器：

由若干高速寄存器组成的硬件堆栈
堆栈操作指令丰富，功能强
支持高级语言编译，简化编译，缩小机器语言与高级语言的语义差距（比如中缀表达式转化为 逆波兰表达式，方便堆栈操作）
支持子程序嵌套和递归调用
存储效率高

引入数据表示的原则：

缩短程序运行时间
减少 CPU 和主存通信量
通用性（能否高效支持多种数据结构）和利用率（硬件设备利用是否充分）

寻址方式：

寻址技术：寻找操作数以及其他信息地址的技术，是软硬件的主要分界面。
研究内容：编址方式、寻址方式和定位方式

编址单位：字编址、字节编址、位编址、块编址
编址单位 = 访问单位
通常访问单位>编址单位

编址方式：分类编址、统一编址、隐含编址

分类编址：不同部件有自己的编址空间，比如 RAM 和 IO 分别都有一个 0 地址
*优点：*指令短、地址形成简单、主存编址空间大
*缺点：*指令中需要有区分部件的标志或约定
统一编址：每个部件统一成一个地址空间，比如内存和 IO 的统一编址
*优点：*简化指令系统
*缺点：*一定程度上地址形成复杂化，因为需要用不同地址区间区分部件
隐含编址：事先约定的编址方式隐含寻址，比如寄存器，cache，无 0 地址空间
*优点：*不需要地址计算，速度快，比如 MIPS 寄存器直接给寄存器号访问
*缺点：*有时候会使指令的设计不规范。

寻址方式： 显式:指令中额外指出寻址方式子段 / **隐式**：指令操作码字段隐含约定方式

大部分计算机采用分类编址方式
三类寻址方式：
面向寄存器寻址方式
面向主存
面向堆栈

逻辑地址：编写程序所使用的地址（符号地址）
主存物理地址：程序在主存中的实际地址

程序在主存中的定位技术：
直接定位：程序装入主存之前，指令和数据的主存物理地址确定
静态再定位：通过装入程序，将逻辑地址转化为物理地址，程序执行过程中，物理地址不改变
动态再定位：执行指令的时候，才形成物理地址

信息分布：
如果主存宽度为 64 位，则一个存储周期内可以访问 8 字节(64bit)
任意存储：
存储跨边界的时候需要两个存储周期才能访问到，速度慢，详见组成原理部分
按字节的整数倍存储：
优点：访问速度快，一个存储周期就可以访问到
缺点：存储空间浪费

指令系统的设计和优化：
内容：指令功能、格式设计

原则：有利于提升机器性价比、有利于指令系统的发展和改进
指令组成：操作码+地址码 （地址码可能有多个，对应多个操作数）
操作码优化目的：能够表达所有指令的同时，操作码子段所占用的位数最少，从而减少程序所占存储空间
操作码编码方式：

1.固定长度编码
2.Huffman 编码
3.扩展编码

*计算整理：信息熵，信息冗余量，实际平均码长

指令字格式优化：

地址码优化：间接寻址、编址寻址、寄存器间接寻址。
多种地址机制：零地址、123 地址

指令系统：CISC 和 RISC

指令系统发展和改变方向：

1.增强指令系统的功能->实现软件功能硬化（CISC，Complex…）
2.简化指令系统->机器指令系统精简（RISC，Reduced Instruction Set Computer）

CISC：增强指令功能从而减少程序指令条数

面向目标程序优化实现->减少存储空间，提高程序运行效率（减少访存）

1. **哈夫曼思想：**

高频度指令：增强功能，加快执行速度，缩短指令长度，增设新指令替代
低频度指令：功能合并到高频指令或者在新的系列中取消
静态使用频度：程序中出现指令的统计百分比->存储空间
动态使用频度：程序执行中使用的指令统计百分比->执行时间

2. **增设强功能复合指令**

*两种方法共同特点：不删改原有指令系统，增加少量新指令，向后兼容

面向高级语言优化实现：缩小高级语言与机器语言的差异，加快编译速度

1.统计使用频度来改进指令->高频语句设专门指令来加快编译和执行速度
2.面向编译、优化代码生成来改进指令->规整：①对称：寄存器同等对待 ②.均匀：不同数据类型等设置相同的指令

缩小各种语言的语义差异：->增大解释的比重，减少翻译的比重

1.指令系统通用
2.缩小差异
3.机器设计多种指令系统，多种系统结构，动态切换（微程序）
4.发展高级语言计算机：不需要编译，直接或间接执行高级语言

面向操作系统的优化：缩短 OS 与系统结构语义差距，减少 OS 所占的时间和空间

统计使用频度来进行改进
增设专用于 OS 的新指令
用硬件或固件实现 OS 的某些功能->使用频繁的子程序硬化或固化
专门的处理机完成 OS，实现功能分布处理结构->IO 处理机

----------End of CISC----------

RISC：简化计算机指令，减少每条指令的周期数

*2080 规律：CISC 中 20%的指令占了 80%的处理时间

CISC 的问题：

a. 指令系统庞大
b. 指令执行速度低
c. 编译程序本身复杂
d. 很多指令使用频率不高

助记符	指令格式	指令功能	实例
LB	LB rt , offset (base)	加载字节	LB a0, 4 (a1)
LBU	LBU rt , offset (base)	加载无符号字节	LBU t0 , 7 (t3)
SB	SB rt , offset (base)	存储字节	SB a0 , 3 (a3)

RISC 的特点：

a. 统一格式的指令
b. 减少指令和寻址种类
c. 大部分指令单周期内完成
d. Load/Store 结构？->见 MIPS 的 load 和 sotre 指令
e. 注重编译优化->三地址指令格式、较多寄存器、对称的指令格式

设计基本原则：

1.只选择使用频度很高的指令，增加少量支持操作系统和高级语言最有用的指令，<100 条
2.较少寻址方式种类：<2
3.简化指令格式：长度相同，2 种以内
4.指令都在一个周期内完成
5.扩大通用寄存器个数，>32，减少访存，指令操作都在寄存器内进行
6.大部分指令都采用硬联控制实现
7.使用精简指令优化设计编译程序，通过简单方式支持高级语言

*缺点：*汇编难写，指令功能太过简单，对浮点运算和虚拟存储器支持强大但不理想

RISC 关键技术：

a.重叠寄存器窗口技术：参数通过公共（重叠）部分传送

b.流水线技术：本条指令执行与下一条指令预取相重叠

c.延迟转移技术：找到不一条不相关的指令来防止流水线断流

d.指令取消技术：

1.向后转移：（循环程序）

2.向前转移：if…then

3.隐含转移技术：if 条件取反，如果成立，则取消指令

e.指令流调整技术：变量重命名消除数据相关-> 相同的变量只占用一个空间（见编译）

f.优化编译系统设计的技术：优化寄存器，减少相关

----------End of RISC----------