Lecture 17: Instruction Set I

指令系统

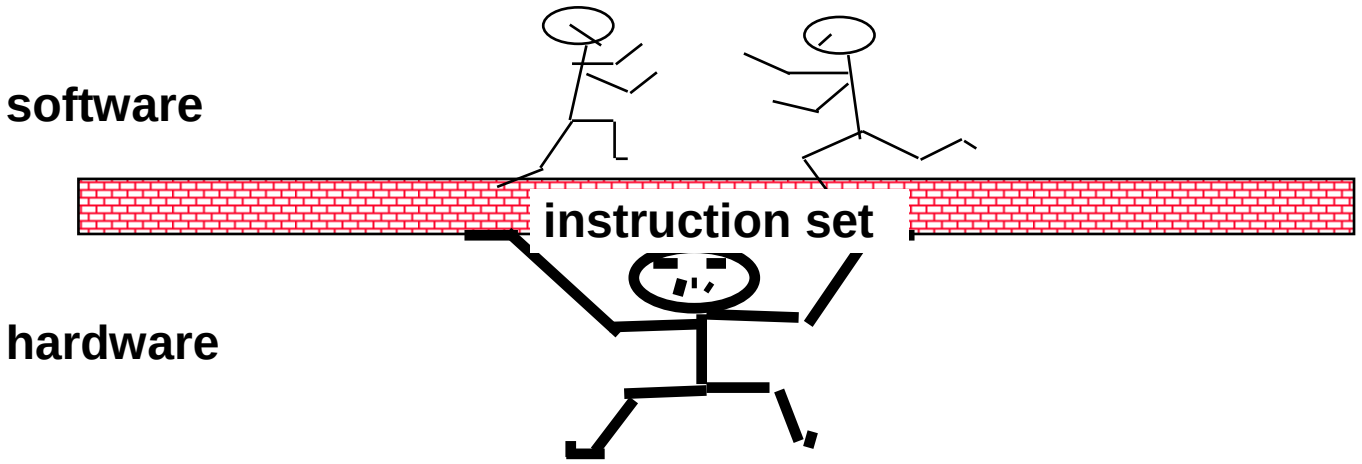
指令系统设计

主 要 内 容

- 一条指令必须指定的信息
- 指令中的地址码个数
- 指令系统设计的基本原则
- 指令类型
- 数据类型
  - 寄存器组织
  - 存储器组织
- 操作数的寻址方式
  - 立即 / 寄存器 / 寄存器间接 / 直接 / 间接 / 堆栈 / 偏移
- 操作码的编码
  - 定长编码法
  - 变长扩展编码法
- 条件码和标志寄存器
- 指令设计风格
- 指令系统举例

Instruction Set Design

- 指令系统处在软/硬件交界面，能同时被硬件设计者和系统程序员看到
- 硬件设计者角度：指令系统为CPU提供功能需求（易于硬件设计）
- 系统程序员角度：通过指令系统来使用硬件，要求易于编写编译器
- 指令系统设计的好坏还决定了：计算机的性能和成本



回顾：冯.诺依曼结构机器对指令规定：

- 用二进制表示，和数据一起存放在主存中
- 由两部分组成：操作码和操作数（或其地址码）
  - **Operation Code**: defines the operation type
  - **Operands**: indicate operation source and destination

一条指令须包含的信息

一条指令必须明显或隐含地包含以下信息：

操作码：指定操作类型

（操作码长度：固定 / 可变）

源操作数参照：一个或多个源操作数所在的地址

（操作数来源：主(虚)存/寄存器/I/O端口/指令本身）

结果值参照：产生的结果存放何处（目的操作数）

（结果地址：主(虚)存/寄存器/I/O端口）

下一条指令地址：下条指令存放何处

（下条指令地址 ：主(虚)存）

（正常情况隐含在PC中，改变顺序时由指令给出）

地址码字段的个数

据上述分析知,一条指令包含 1 个操作码和多个地址码

零地址指令

(1) 无需操作数　如：空操作 / 停机等

(2) 所需操作数为默认的　如：堆栈 / 累加器等

形式：

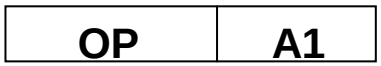
一地址指令

其地址既是操作数的地址，也是结果的地址

(1) 单目运算：如：取反 / 取负等

(2) 双目运算：另一操作数为默认的　如：累加器等

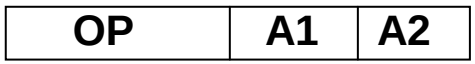
形式：



二地址指令（最常用）

分别存放双目运算中两个操作数，并将其中一个地址作为结果的地址。

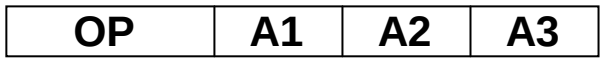
形式：



三地址指令（RISC风格）

分别作为双目运算中两个源操作数的地址和一个结果的地址。

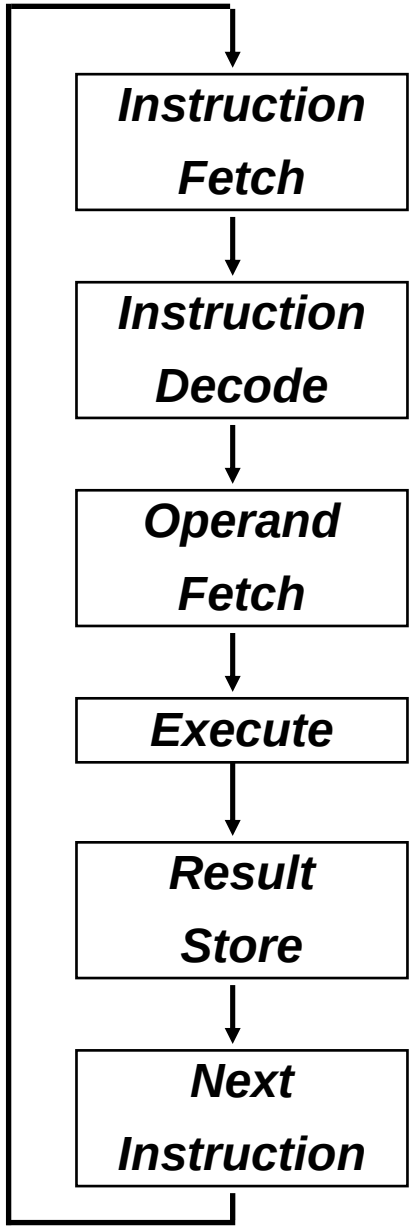
形式：



多地址指令

大中型机中用于成批数据处理的指令,如:向量 / 矩阵等（SIMD）

从指令执行周期看指令设计涉及的问题



Obtain instruction from program storage

指令地址、指令长度（定长/变长）

Determine required actions

指令格式、操作码编码、操作数类型

Locate and obtain operand data

地址码、寻址方式、操作数格式和存放方式

Compute result value or status

操作类型、标志或条件码

Deposit results in storage for later use

结果数据位置（目的操作数）

Determine successor instruction

下条指令地址（顺序 / 转移）

指令格式的设计

指令格式的选择应遵循的几条基本原则：

- 应尽量短
- 要有足够的操作码位数
- 指令编码必须有唯一的解释，否则是不合法的指令
- 指令字长应是字节的整数倍
- 合理地选择地址字段的个数
- 指令尽量规整

与指令集设计相关的重要方面

- 操作码的全部组成：操作码个数/种类/复杂度
- 数据类型：对哪几种数据类型完成操作
- 指令格式：指令长度/地址码个数/各字段长度
- 通用寄存器：个数/功能/长度
- 寻址方式：操作数地址的指定方式
- 下条指令的地址如何确定：顺序，PC+1；条件转移；无条件转移；……

一般通过对操作码进行不同的编码来定义不同的含义，

操作码相同时，再由功能码定义不同的含义！

Typical Operations(典型的操作)

Data Movement	Load (from memory) Store (to memory) memory-to-memory move register-to-register move push, pop (to/from stack)
Input/Output	input (from I/O device) output (to I/O device)
Arithmetic	integer (binary + decimal) or FP Add, Subtract, Multiply, Divide adc(带进位加)，sbb (带借位减)
Logical	not, and, or, set, clear
Shift	(arithmatic,logic,rotate)left/right shift
String	search, translate
Exec-Seq control	Jump, branch
CPU control	stop, sti(开中断), break
Subroutine Linkage	call, return
Interrupt	trap, interrupt return
Synchronization	test & set (atomic r-m-w)

操作数类型和存储方式

操作数是指令处理的对象，与高级语言数据类型对应，基本类型有哪些？

地址

被看成无符号整数，用来参加运算，以确定主(虚)存地址

数值数据

定点数(整数)：一般用二进制补码表示

浮点数(实数)：大多数机器采用IEEE754标准

十进制数：一般用NBCD码表示，压缩/非压缩

位、位串、字符和字符串

用来表示文本、声音和图像等

» 4 bits is a nibble（一个十六进制数字）

» 8 bits is a byte

» 16 bits is a half-word

» 32 bits is a word

逻辑(布尔)数据

按位操作（0-假 / 1-真）

Pentium & MIPS Data Type

- **Pentium**
  - 基本类型：
    - » 字节、字(16位)、双字(32位)、四字(64位)
  - 整数：
    - » 16位、32位、64位三种2-补码表示的整数
    - » 18位压缩8421 BCD码表示的十进制整数
  - 无符号整数（8、16或32位）
  - 近指针：32位段内偏移（有效地址）
  - 浮点数：IEEE 754（80位扩展精度浮点数寄存器）
- **MIPS**
  - 基本类型：
    - » 字节、半字(16位)、字(32位)、四字(64位)
  - 整数： 16位、32位、64位三种2-补码表示的整数
  - 无符号整数：（16、32位）
  - 浮点数：IEEE 754（32位/64位浮点数寄存器）

Addressing Modes（寻址方式）

- 什么是“寻址方式”?

  操作数指定方式。即：用来指定操作数或操作数所在位置的方法

- 地址码编码由操作数的寻址方式决定
- 地址码编码原则：

  指令地址码尽量短 —为什么?→ 目标代码短，省空间

  操作数存放位置灵活，空间应尽量大 → 利于编译器优化产生高效代码

  有效地址计算过程尽量简单 → 指令执行快

- 指令的寻址----简单

  正常：PC增值

  跳转 (jump / branch / call / return)：同操作数的寻址

- 操作数的寻址----复杂（想象一下高级语言程序中的操作数情况！！）

  操作数来源：寄存器 / 外设端口 / 主(虚)存 / 栈顶

  操作数结构：位 / 字节 / 半字 / 字 / 双字 / 一维表 / 二维表 /...

  通常寻址方式特指“操作数的寻址”

Addressing Modes

- 寻址方式的确定

  （1）在操作码中给定寻址方式

  如：MIPS指令，指令中仅有一个主(虚)存地址，且指令中仅有一、二种寻址方式。Load/store型机器指令属于这种情况。

  （2）有专门的寻址方式位

  如：X86指令，指令中有多个操作数，且寻址方式各不相同，需要各自说明寻址方式。

- 有效地址的含义

  操作数所在内存单元的地址，可通过指令计算得到

- 基本寻址方式

  立即 / 直接 / 间接 / 寄存器 / 寄存器间接 / 偏移 / 堆栈

- 基本寻址方式的算法及优缺点

  （见下页）

基本寻址方式的算法和优缺点

指令：OP A，R，.....

假设：A=地址字段值，R=寄存器编号，
EA=有效地址，(X)=地址X中的内容

方式	算法	主要优点	主要缺点
立即	操作数=A	指令执行速度快	操作数幅值有限
直接	EA=A	有效地址计算简单	地址范围有限
间接	EA=(A)	有效地址范围大	多次存储器访问
寄存器	操作数=(R)	指令执行快，指令短	地址范围有限
寄间接	EA=(R)	地址范围大	额外存储器访问
偏移	EA=A+(R)	灵活	复杂
堆栈	EA=栈顶	指令短	应用有限

偏移方式：将直接方式和寄存器间接方式结合起来。
有：相对 / 基址 / 变址三种 （见后面几页！）

问题：以上各种寻址方式下，操作数在寄存器中还是在存储器中？有没有可能在磁盘中？什么情况下，所取数据在磁盘中？

只有当操作数在存储器中时，才有可能“缺页”，此时操作数在磁盘中！