体系结构 第二章

by xjc

第二章 计算机指令集结构

2.1 指令集结构的分类

2.1.1 指令集结构的分类依据及结果

1. 区分不同指令集结构的主要因素： CPU中 用于存放*操作数* 的 *存储单元的类型*
2. 3种存放操作数的存储单元的类型
    1. 堆栈
    2. 累加器
    3. 通用寄存器组
3. 指令集结构的分类
    1. 堆栈结构
    2. 累加器结构
    3. 通用寄存器结构/load-store结构（寄存器-存储器结构RM结构，寄存器-寄存器结构RR结构）

2.1.2 通用寄存器结构详解

特点： 1）现代指令集结构的主流；2）在灵活性和提高性能方面具有明显的优势：访问块，便于编译器使用，可存放变量，减少对存储器的访问，用更少的地址位寻址
细分类型： 1）寄存器-寄存器型（RR）；2）寄存器-存储器型（RM）；3）存储器-存储器型（MM）

2.2 寻址方式

10种寻址方式，要求掌握
有几种方式容易错，多看看

寻址方式	代码示例	含义
寄存器寻址	Add R4, R3	Regs[R4]←Regs[R4]+Regs[R3]
立即数寻址	Add R4, #3	Regs[R4]←Regs[R4]+3
偏移寻址	Add R4, 100(R1)	Regs[R4]←Regs[R4]+Mem[100+Regs[R1]]
寄存器间接寻址	Add R4, (R1)	Regs[R4]← Regs[R4] + Mem[Regs[R1]]

寻址方式	代码示例	含义
索引寻址	Add R3, (R1+R2)	Regs[R3]← Regs[R3] + Mem[Regs[R1]+Regs[R2]]
直接寻址（绝对寻址）	Add R1, (1001)	Regs[R1]←Regs[R1] + Mem[1001]
存储器间接寻址	Add R1, @(R3)	Regs[R1]←Regs[R1]+Mem[Mem[Regs[R3]]
自增寻址	Add R1, (R2)+	Regs[R1]←Regs[R1]+Mem[Regs[R2]] Regs[R2]= Regs[R2] + d
自减寻址	Add R1, -(R2)	Regs[R2] = Regs[R2] - d Regs[R1]←Regs[R1] + Mem[Regs[R2]]
缩放寻址	Add R1, 100(R2),[R3]	Regs[R1]←Regs[R1]+Mem[100 + Regs[R2] + Regs[R3]]

2.3 指令集结构的功能设计

2.3.0 指令集结构功能设计概念

1. 功能设计的思路：确定软硬件功能分配
2. 考虑因素：速度、成本、灵活性
3. 对指令集的基本要求：完整性、规整性、高效率、兼容性
4. 两种设计策略：CISC（复杂指令集计算机）和RISC（精简指令集计算机）

2.3.1 CISC指令集结构的功能设计

(1) 设计目标

- 强化指令功能，功能交由硬件实现
- 减少程序的指令条数
- 以达到提升性能的目的

(2) 增强指令功能的方法

1. 面向*目标程序*增强指令功能
2. 面向*高级语言*的优化实现改进指令集
3. 面向*操作系统*的优化实现改进指令集

2.3.2 RISC指令集结构的功能设计

(1) CISC存在的问题

1. 各种指令的*使用频度相差悬殊*
2. 指令集庞大，指令条数多，功能复杂，导致*控制器硬件非常复杂*

3. 指令CPI大，*执行速度慢*
4. 指令功能复杂导致*规整性不好*，不利于采用*流水技术*提高性能

（2）RISC指令集功能设计原则

1. 指令数少且简单
2. 指令格式简单统一，寻找方式少
3. 保证指令可在单个周期内完成
4. 采用load-store架构，只针对寄存器进行运算
5. 指令尽量用硬件逻辑实现
6. 基于寄存器，为编译器提供更大的优化空间
7. 可充分利用pipeline流水技术来挖掘ILP（指令并行性）

（3）早期的RISC微处理器特点（了解）

1. 采用load-store结构
2. 指令字长为32位
3. 采用高效的流水技术

2.3.3 控制指令

(1) 功能：改变控制流
(2) 类型：跳转（无条件）、分支（有条件）
(3) 能够改变控制流的指令：分支、跳转、过程调用、过程返回
(4) 使用频度：改变控制流的大部分是*分支指令*
(5) 表示分支条件的方法：条件码（ALU设置）、条件寄存器、比较与分支
(6) 转移目标地址的表示：在指令中添加偏移量，与PC相加得到目标地址
(7) 过程调用与返回：

- 保存机器状态（保存返回地址）
- 以前有专门的保存机制，现在使用load/store保存和恢复

2.4 操作数的类型和大小

2.4.1 操作数的设计概念

(1) 数据表示：计算机*硬件*能够直接识别、*指令集*可以直接调用的数据类型
(2) 数据结构：由*软件*进行处理和实现的各种数据类型

2.4.2 表示操作数类型和大小

（1）表示操作数类型

方法一：指令*操作码指定*操作数类型（常用方法）
方法二：*数据中包含指示*操作数类型的

（2）表示操作数大小

主要的大小：字节（8位）半字（16位）字（32位）双字（64位）

2.5 指令格式的设计

2.5.1 指令格式设计基本概念

指令的组成：操作码+地址码
指令格式的设计：确定*指令字的编码方式*，含*操作码字段+地址码字段*的编码与表示方式

2.5.2 操作码编码方式

（1）Huffman编码法：操作码*变长编码*，减少了操作码平均位数，但不规整
（2）固定长度：保证操作码译码速度

2.5.3 寻址方式表示

（1）寻址方式编码于*操作码*中
（2）设置专门的*地址描述符*

2.5.4 指令格式设计考虑因素

（1）**指令和目标代码大小**：寄存器个数、寻址方式数目对*指令平均字长和目标代码大小*的影响
（2）**硬件处理/流水**：指令格式 需要便利 硬件处理，尤其是*流水处理*
（3）**指令字长限制**：指令字长为*字节（8位）的整数倍*，不能随意设计

2.5.5 指令集的3种编码格式

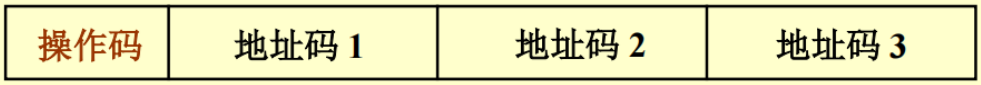
（1）变长编码格式

1. **应用范围：**指令集寻址方式、操作种类很多时，效果最好
2. **优点：***最少二进制位*表示目标代码
3. **缺点：**每条指令字长、执行时间差距大

操作码	地址描述符 1	地址码 1	…	地址描述符 n	地址码 n

（2）定长编码格式

1. **特点：** 操作类型+寻址方式 编码到 操作码中
2. **应用：** 寻址方式、操作种类很少时，效果好
3. **优点：** 降低译码复杂度，提高效率
4. **RISC：** RISC使用这种



（3）混合型编码格式

1. **特点：** 变长和定长的结合，提供若干固定指令字长
2. **优点：** 一举两得，减少目标代码长度，也能降低译码复杂度

2.6 MIPS指令集结构（理解，无需精准记忆）

2.6.1 MIPS的寄存器

寄存器类型	个数	位数	符号	功能	备注
通用寄存器(GPRs)	32	64位	R0~R31	存放整数	R0永远为0
浮点数寄存器(FPRs)	32	64位	F0~F31	存放浮点数	存放单精度(32位时，只用一半)
特殊寄存器	/	/	/	/	可与通用寄存器交换数据

2.6.2 MIPS的数据表示

（1）整数

- 字节（8位）
- 半字（16位）
- 字（32位）
- 双字（64位）

（2）浮点数

- 单精度浮点（32位）
- 双精度浮点（64位）

（3）扩展备注

字节、半字、字（小于等于32位）装入64位寄存器时，进行*零扩展/符号扩展*，此后按照64位整数运算

2.6.3 MIPS数据寻址方式

寻址类型	特殊功能
立即数寻址 偏移量寻址	均为16位
寄存器间接寻址	0作为偏移量
绝对寻址	16位，把R0作为基址寄存器
存储器	字节寻址，64位
存储器访问	必须边界对齐

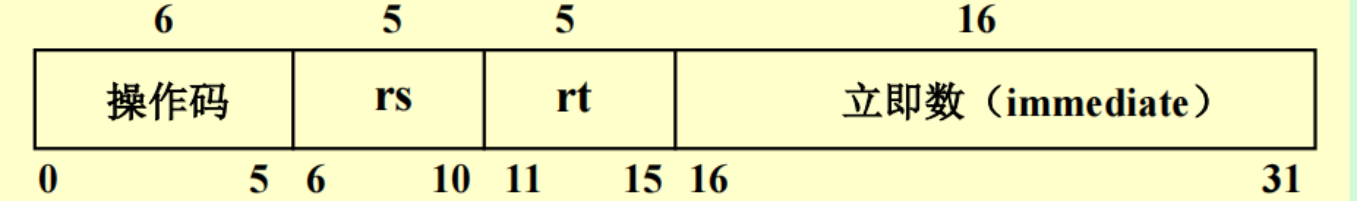
2.6.4 MIPS的指令格式

（1）特点

1. 寻址方式编码到操作码种
2. 所有指令均为32位
3. 操作码占6位
4. 3种指令格式

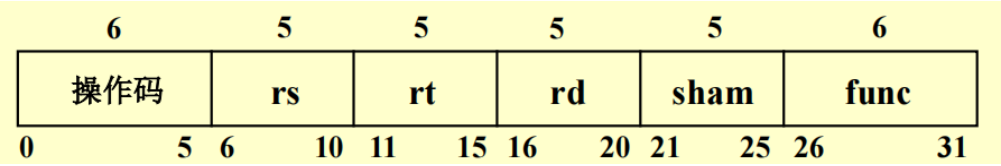
（2）I类指令

包含类型： load/store指令、立即数指令、分支指令、寄存器跳转指令、寄存器连接跳转指令
立即数： 字段为16位



（3）R类指令

包含类型： ALU指令、专用寄存器读/写指令、move指令



（4）J类指令

包含类型：跳转指令、跳转并连接指令、自陷指令、异常返回指令

低26位为偏移量

2.6.5 MIPS的操作

（1）MIPS的四种指令

1. load/store
2. ALU
3. 分支与跳转
4. 浮点操作

（2）各种指令的具体解释

3. load和store指令

指令举例	指令名称	含 义
LD R2，20(R3)	装入双字	Regs[R2]←$_{64}$ Mem[20+Regs[R3]]
LW R2，40(R3)	装入字	Regs[R2]←$_{64}$ (Mem[40+Regs[R3]]$_0$)^{32} ## Mem[40+Regs[R3]]
LB R2，30(R3)	装入字节	Regs[R2]←$_{64}$ (Mem[30+Regs[R3]]$_0$)^{56} ## Mem[30+Regs[R3]]
LBU R2，40(R3)	装入无符号字节	Regs[R2]←$_{64}$ 0^{56} ## Mem[40+Regs[R3]]
LH R2，30(R3)	装入半字	Regs[R2]←$_{64}$ (Mem[30+Regs[R3]]$_0$)^{48} ## Mem[30+Regs[R3]]## Mem[31+Regs[R3]]
L.S F2，60(R4)	装入半字	Regs[F2]←$_{64}$ Mem[60+Regs[R4]] ## 0^{32}
L.D F2，40(R3)	装入双精度浮点数	Regs[F2]←$_{64}$ Mem[40+Regs[R3]]
SD R4，300(R5)	保存双字	Mem[300+Regs[R5]]←$_{64}$ Regs[R4]
SW R4，300(R5)	保存字	Mem[300+Regs[R5]]←$_{32}$ Regs[R4]
S.S F2，40(R2)	保存单精度浮点数	Mem[40+Regs[R2]]←$_{32}$ Regs[F2] $_{0..31}$
SH R5，502(R4)	保存半字	Mem[502+Regs[R4]]←$_{16}$ Regs[R5] $_{48..63}$

1. LD = LOAD DOUBLE
2. LW = LOAD WORD
3. LB = LOAD BYTE
4. LBU = LOAD BYTE UNSIGNED
5. LH = LOAD HALF WORD
6. L.S = LOAD SINGLE FLOAT
7. L.D = LOAD DOUBLE FLOAT

1. SD = STORE DOUBLE
2. SW = STORE WORD
3. SB = STORE BYTE
4. SBU = STORE BYTE UNSI
5. SH = STORE HALF WORD
6. S.S = STORE SINGLE FL
7. S.D = STORE DOUBLE FL

4. ALU指令

寄存器-寄存器型（RR型）指令或立即数型

算术和逻辑操作：加、减、与、或、异或和移位等

指令举例	指令名称	含义
DADDU R1, R2, R3	无符号加	Regs[R1]← Regs[R2]+ Regs[R3]
DADDIU R4, R5, #6	加无符号立即数	Regs[R4]← Regs[R5]+6
LUI R1, #4	把立即数装入到一个字的高16位	Regs[R1]← 0^{32} ## 4 ## 0^{16}
DSLL R1, R2, #5	逻辑左移	Regs[R1]← Regs[R2]<<5
DSLT R1, R2, R3	置小于	If(Regs[R2]< Regs[R3]) Regs[R1]← 1 else Regs[R1]←0

1. DADDU = DATA ADD UNSIGNED
2. DADDIU = DATA ADD IMMEDIATE UNSIGNED
3. LUI = LOAD UNSIGNED IMMEDIATE
4. DSLL = DATA SHIFT LOGIC LEFT
5. DSLT = DATA SET IF LITTLE

2.6.6 MIPS的控制指令

指令举例	指令名称	含义
J name	跳转	PC $_{36\cdots63}$← name<<2
JAL name	跳转并链接	Regs[R31]←PC+4；PC $_{36\cdots63}$←name<<2； ((PC+4)－2^{27})≤name＜((PC+4)+2^{27})
JALR R3	寄存器跳转并链接	Regs[R31]←PC+4；PC← Regs[R3]
JR R5	寄存器跳转	PC← Regs[R5]
BEQZ R4，name	等于零时分支	if(Regs[R4]== 0) PC←name ； ((PC+4)－2^{17})≤name＜((PC+4)+2^{17})
BNE R3，R4，name	不相等时分支	if(Regs[R3]!= Regs[R4]) PC←name ((PC+4)－2^{17})≤name＜((PC+4)+2^{17})
MOVZ R1，R2，R3	等于零时移动	if(Regs[R3]==0) Regs[R1]← Regs[R2]

由*一组跳转和一组分支指令*来实现控制流的改变

（1）跳转指令

4种类型

1. **确定目标地址的方式：**
   - 26位偏移量左移2位，替换PC的28位
   - 寄存器给出目标地址（间接跳转）
2. **跳转是否链接：**
   - 简单跳转（直接给PC）
   - 跳转并链接（目标地址给PC，返回地址给R31）

（2）分支指令

分支条件由指令确定
分支的目标地址：16为带符号偏移量 左移2位 与PC相加得到

2.6.7 MIPS的浮点操作

1. **浮点指示：**操作码指示操作数的精度，后缀S表示单精度，后缀D表示双精度
2. **浮点操作：**加减乘除，分别有单精度和双精度指令
3. **浮点数比较指令**：根据比较结果设置浮点状态寄存器，并进行分支