CH5 指令集体系结构

一、指令集体系结构

指令集体系结构（ISA，也称指令系统），是对处理器硬件细节的抽象描述，即**设计规范**，定义了处理器能够做什么，也是系统级程序员所能看到的处理器的属性。

ISA包括：

> 指令功能的编码
>
> 基本数据类型
>
> 寄存器
>
> 寻址模式
>
> 存储管理机制
>
> 异常和中断处理
>
> 运行时环境和运行级别控制

ISA为上层提供了使用硬件的规范和接口。

二、汇编语言

1. 汇编语言与机器语言

汇编语言：便于人阅读，包括汇编指令，伪指令（标签），宏指令等

机器语言：机器可读的01序列，也叫机器指令

汇编语言和机器语言是一一对应的关系。

设计准则：

- 简单设计有利于规整化
- 加速常用功能
- 越小越快
- 好的设计需要折中

2. 概念

操作数：

> 寄存器：使用$开头
>
> 存储器：用于存放常用变量（寄存器放不下）
>
> 立即数：可以被指令立即访问（16位补码）

MIPS基于字节寻址。

3. 基本指令

`a = b + c` → `add a, b, c`

`a = b - c` → `sub a, b, c`

读字寻址 → `lw $s0, 5($t1)` 将$t1保存的地址+5得到的地址下保存的数据移动到$s0中。

写字寻址 → `sw $t4, 0x7($0)` 将$t4下保存的数存储在$0下保存的地址下。

字节访问：`lb  rb`

半字访问：`lh  rh`

立即数加法：`addi $s0, $s0, 4`（没必要使用subi）

4. 字节序

- 小端：低地址存放低位
- 大端：低地址存放高位

```
s = 0x23456789; // 原始数据
b = 0x23456789; // 大端序
s = 0x89674523; // 小端序
```

三、机器语言

1. 概念

指令的二进制表示形式。

RISC中，指令为定长编码

MIPS使用32位指令

- R（寄存器）类型指令
- I（立即数）类型指令
- J（跳转）类型指令

2. R指令

三个操作数：rs、rt（源寄存器）、rd（目标寄存器）

op：操作码，R型指令操作码为0

funct：功能码，由funct决定操作类型

shamt：仅用于移位操作，存储移位的位数

op(6) + rs(5) + rt(5) + rd(5) + shamt(5) + funct(6)

3. I指令

三个操作数：rs、rt（寄存器操作数）、imm（16位补码立即数）

op：操作码

- 每个I指令都具有一个对应的op
- 指令的操作由op字段决定

op(6) + rs(5) + rt(5) + imm(16)

4. J指令

op：操作码

26位的跳转地址

op(6) + addr(26)

5. 机器指令译码

机器指令以操作码开头，根据机器代码后面的部分如何进行解释

6. 程序的存储

程序指令存储在存储器中，使用**程序计数器（PC）**存储当前正在执行指令在内存中的地址。

7. 常用MIPS汇编指令

逻辑运算

> R类型指令：and, or, xor, nor（同或）
>
> I类型指令：andi, ori, xori（立即数为16位补码）
>
> 不提供not指令，not A = A nor 0
>
> 不提供nori指令，用其他指令代替

跳转指令

> beq 等于时跳转(je)
>
> bne　不等于时跳转(jne)
>
> j　　强制跳转

有符号数比较

> R类型：slt
>
> I类型：slti（符号位扩展）

无符号数比较

> R类型：sltu
>
> I类型：sltiu（符号位扩展）

有符号数数据装载（符号位扩展）

> 装入字：lw
>
> 装入半字：lh
>
> 装入字节：lb

无符号数数装载（0扩展）

> 装入字：lwu（没有这个指令，因为不需要扩展）
>
> 装入半字：lhu

> 装入字节：lbu

存储数据

> 存储字：sw
>
> 存储半字：sh
>
> 存储字节：sb

四、单周期MIPS32处理器的设计

1. 特点

- 每条指令均在一个时钟周期内完成
- 32个32位寄存器，哈佛结构（程序存储器和数据存储器分开），小端模式，支持23条指令
- 数据通路：完成对指令中的操作数的运算、存储等处理工作
- 控制通路：从数据通路中接收指令，并对其进行翻译以告知数据通路如何处理

在各个记忆部件之间添加组合逻辑电路，在控制单元的控制下根据当前电路的状态计算出电路的新状态。

2. 工作步骤

- 取指令
- 译码——从寄存器取操作数，另一个操作数进行符号扩展
- 执行——计算访存地址
- 访存/写回——从数据存储器取回数据，写入寄存器文件
- 更新PC——计算下一条指令的地址

3. 控制单元

RegWrite：指令是否将结果写回寄存器

RegDst：目的寄存器由**rt**字段确定还是**rd**字段确定

ALUSrc：源操作数**B**是来自寄存器还是立即数

MemWrite：指令是否要写存储器

MemtoReg：指令运行结果来自**ALU**还是存储器

ALUCtrl：**ALU**操作码，决定其进行哪种运算

Branch：标识当前指令是否是分支指令

Jump：标识当前指令是否是跳转指令

常见指令及其控制参数列表：

	RegWrite	RegDst	ALUSrc	MemWrite	MemtoReg	ALUCtrl	Branch	Jump
lw	1	0	1	0	1		0	0
sw	0	X	1	1	X		0	0
addi	1	0	1	0	0		0	0
and	1	1	0	0	0		0	0
or	1	1	0	0	0		0	0
add	1	1	0	0	0		0	0
sub	1	1	0	0	0		0	0
slt	1	1	0	0	0		0	0
beq	0	X	0	0	X		1	0
j	0	X	X	0	X		X	1

RegDst：1目标字段由rt确定，0目标字段由rd确定

RegWrite：寄存器写入使能端（仅在需要写回寄存器时需要考虑）

ALUSrc：0从寄存器文件选择srcB，1从SignImm选择srcB

ALUControl：控制ALU操作符

MemtoReg：0ALU结果写入寄存器，1内存写入寄存器（仅在需要写入寄存器时考虑）

MemWrite：1向存储器中写入文件

Branch：判断是否含有分支结构

jump：判断是否需要跳转

4. 单周期处理器的性能分析

CPU Time = $\frac{Time}{program}$=$\frac{Instructions}{Program}\times\frac{Cycles}{Instruction}\times\frac{Time}{Cycle}$

$$T_c = t_{pcq\_PC} + t_{mem} + \max(t_{RFread}, t_{sext} + t_{mux}) + t_{ALU} + t_{mem} + t_{mux} + t_{RFsetup}$$
$$= t_{pcq\_PC} + 2t_{mem} + t_{RFread} + t_{mux} + t_{ALU} + t_{RFsetup}$$

sext：位扩展模块

mux：复用器模块

五、期末大题

- 卡诺图
- 延迟
- 复用器
- 状态机