

# 计算机组成:软硬件接口

daydalek

2023 年 2 月 26 日

## 1 概要

### 1.1 衡量计算机系统的速度

如果时间来度量计算机的性能,则完成相同的计算任务,需要时间更少的计算机更快

使用CPU执行时间(CPU execution time) 它只表示CPU上花费的时间

使用系统性能(system performance)来表示空载系统的响应时间

并用属于CPU性能(CPU performance)的术语来表示用户CPU时间

### 1.2 相关公式

$$\text{程序的CPU执行时间} = \text{程序的CPU时钟周期数} \times \text{时钟周期时间} \quad (1)$$

$$\text{程序的CPU执行时间} = \text{程序的CPU时钟周期数} / \text{时钟频率} \quad (2)$$

这是因为时钟周期时间和时钟频率是相互倒数的. 一个程序需要的时钟周期数可写为

$$\text{程序的CPU时钟周期数} = \text{程序的指令数} \times \text{每条指令的平均时钟周期数} \quad (3)$$

术语CPI(Cycles Per Instruction)表示每条指令的平均时钟周期数

于是我们得到以下公式

$$\text{CPU时间} = \text{指令数} \times CPI \times \text{时钟周期时间} \quad (4)$$

## 2 指令

### 2.1 计算机硬件的操作数

MIPS体系架构中的一字为32位(MIPS32)或64位(MIPS64) 此处讨论MIPS32

#### 2.1.1 存储器操作数

由于MIPS只能操作寄存器中的数据，因此需要包含在存储器和寄存器之间传递数据的指令，这些指令叫做数据传送指令(Data Transfer Instructions)

为了访问存储器中的每一个字，指令需要给出存储器地址，将数据从存储器复制到寄存器的指令叫做取数指令(Load Instructions)，格式是lw+目标寄存器+ 用于寻址的常数和寄存器  
一个例子：

```
1      A[12]=h+A[8];  
2      -----  
3      lw $t0,32($s3) # $t0=A[8]  
4      add $t0,$t0,$s2 # $t0=h+A[8]  
5      sw $t0,48($s3) # store $t0 in A[12]
```

#### 2.1.2 立即数操作数

立即数操作数是指令中的常数，它们是在指令中直接给出的，而不是从存储器中读取的

例如要向S3中+4，可以这么写

```
1      addi $s3,$s3,4
```

相对从存储器提取常数，使用立即数操作更快

### 2.2 计算机中指令的表示

以这段代码为例

```
1      add $t0,$s1,$s2
```

其十进制表示为 0 17 18 8 0 32

以二进制表示是 000000 01001 01010 00000 01000 100000

指令的布局叫做指令格式(Instruction Format) 可以看出它是32位长的; 为了与汇编语言区分, 称其为机器语言(Machine Language), 指令序列称为机器码(Machine Code)

### 2.2.1 MIPS字段

为了简化, 把以上字段命名如下

op rs rt rd shamt funct

其中op是操作码, rs是源1寄存器, rt是源2寄存器, rd是目的寄存器, shamt是移位量, funct是功能码

它们的位数分别是6,5,5,5,5,6

然而, 这种安排存在着局限性, 它使得寻址范围极为有限。例如, 取字指令必须指定两个寄存器和一个常数在上述格式中, 如果地址使用其中的一个5位字段, 那么取字指令的常数就被限制在 $2^5$ (即32)之内. 这个常数通常用来从数组或数据结构中选择元素, 显然是不够的。所以, MIPS还引入另一种指令格式, 同样是32位长, 但是它的布局是 op rs rt constant or address 这种格式的指令称为I型指令, 它的字段分别是6,5,5,16

注意, 此处的rt含义已经发生变化, 它不再是源操作数寄存器, 而是目的寄存器

举个例子, 取字指令的格式是

```
1      lw $t0,32($s3)
```

它的机器码是 100011 10011 01000 0000000000100000

16位地址或常数码支持更大的基址偏移量, 从而支持更大范围的寻址