实验6 简单计算机系统-系统设计D-实验报告

电 25 吴晨聪 2022010311

# 6.1 搭建MIPS指令集add和sub指令的数据通路

和实验5搭建的精简指令集类似，主要的数据通路都类似，需要注意的点在于：1、数据位数发生了变化；2、指令格式发生了变化；3、controller新增了funct输入。所以需要做的就是在原有的基础上针对变化做些变更。

对于数据位数的变化，需要将rom、mux、regfile、ALU、controller和ram的位数都作相应的调整，同时由于ALU的计算位数是32位而imm的位数是16位，所以有必要在imm向ALU联通的路径上增加一个移位器，将imm的高16位补上0；对于指令结构的变化，需要对指令寄存器的输出进行相应的修改，op、Rs、Rt、Rd、imm、funct都按照指令的格式取相应的位数即可；由于MIPS指令的R型指令的指令类型不在op中区分，而是在funct中区分，所以需要在controller中新增输入funct，并将判断指令的格式的方式从原本的仅仅根据op变更为根据op和funct来判断。

如此完成了从原本指令集到MIPS指令集的变更，ALU内部的运算实现较为简单就不再赘述，如下图1所示是任务1的测试结果：

一張含有 文字, 螢幕擷取畫面, 多媒體軟體, 繪圖軟體 的圖片

AI 產生的內容可能不正確。

图1 任务1测试结果

前两条指令分别测试了加运算和减运算（124+236=360，13-1=12），运算结果均正确。此外还验证了跳转指令和立即数指令。先把16和17号regfile内容清空，再分别存入15和14，在进行BEQ指令（跳转不会成功），随后在执行BNE指令，此时成功跳转到了pcout=19，再对17号regfile的内容加1并JMP回最开始的跳转指令，此时16和17号regfile的值均为15是相同的，所以BEQ跳转成功，最终跳出循环。

经过验证，该CPU对于R指令的add和sub，I指令的addi、andi、beq和bne以及jmp指令都是可以运行的。以此为基础完成任务2的内容。

rom存入的机器码与汇编语言格式指令之间的关系如下表所示。

|  |  |  |
| --- | --- | --- |
| 行号 | 指令代码 | 机器码 |
| 0 | add $s0, $s1,$s2 | 0x02328020 |
| 1 | sub $t0, $t3, $t5 | 0x016d4022 |

# 6.2 用MIPS完成质数求和

在原本5.2任务的基础上进行改进，整体逻辑仍然是可行的，但是在细节上由于MIPS指令所用的数据位数都是足够的，所以无需采取多个RAM存储一个数据的方法，此外由于有足够的寄存器容量，所以所有的运算可以在寄存器内进行，省去了LW和SW的操作。

在如此改进之后得到的指令如文章末尾的附录1所示。做的改变是不采用RAM存储数据，而是直接将数据存储在寄存器regfile上，这样一方面节省了代码空间，还减少了计算时间。不用考虑位数的问题（计算的数据不涉及溢出的问题），所以直接进行加减运算即可，这对代码逻辑复杂度也有了很大的简化。

运行结果如下图2所示：可见仅仅是计算位数和寄存器大小的提升就能把原来的130行的代码缩减到30行，运算时间也从原来的0.7s提升到了现在的0.13s（模拟时钟累计时间）。如果利用devide指令来代替循环减判断整除，再利用求平方根方法来确定循环上界，则代码长度和运行时间还能得到进一步的优化。经Matlab检验，1~1149之内的素数之和为94056，结果正确。

rom存入的机器码与汇编语言格式指令之间的关系见Ex6.xlsx及rom2.hex。

一張含有 螢幕擷取畫面, 文字, 多媒體軟體, 繪圖軟體 的圖片

AI 產生的內容可能不正確。

图2 Matlab实现质数求和代码