Lab06 多周期MIPS CPU设计

PB15000102 王嵩超

实验目的

设计一个多时钟周期的MIPS CPU，要求能够执行斐波拉契数列的汇编程序。

相对上一个实验的改进

- 对于存储器位宽为32，而PC=PC+4是以8位为位宽的写法，在上一次实验中为了图简便，将此式写为`PC=PC+1`。本次实验进一步统一地址的表示。内部地址的计算完全按照原有的8位位宽地址来。只是在取数存数时，将地址除以4。这样PC自增的表达式为：`PC=PC+4`。
- 运算单元可以先后复用：先用来计算地址（R型指令），后用来计算自增的PC（个人觉得大材小用，这使得我上次实验写的PC之后的表达式全要重写！而且流水线设计中PC的自增模块肯定是独立单元，不会用别的运算指令用到的ALU）。
- 没有按照PPT接线图那样另外增添mem_data寄存器。增添寄存器的代价是需要等待一个时钟使值写入寄存器。
- instruction寄存器：因为一条指令的执行过程中可能会再次访存，此时m_dout的改变会影响到instruction。所以用寄存器暂存指令比较保险。

> 对于本实验中的其他各寄存器暂存输出（都用寄存器来阻断）表示不太理解。寄存器更多地是为流水线而准备的。

- 对PC跳转、分支、还是正常自增的逻辑现在放到了control里面。在上一个实验中是在PCModule中实现的。
- 使用同步存储器，对状态机的修改：

  增加状态：S5plus、S8plus、S11plus

  都是为了为S0的取指做好准备。要使S0执行时，PC已经为正确的PC，这样S0执行后即可得到指令字。
- 默认的数据段0x00002000远超IP核支持的地址范围。将数据段起始地址改为：0x00000100

S0的控制信号：PCWrite，实际上在S1才起作用。故PC要等到S1才能被修改。

各个之前状态发出的信号，在下一状态要及时去除。以免发生意想不到的错误。

仿真结果

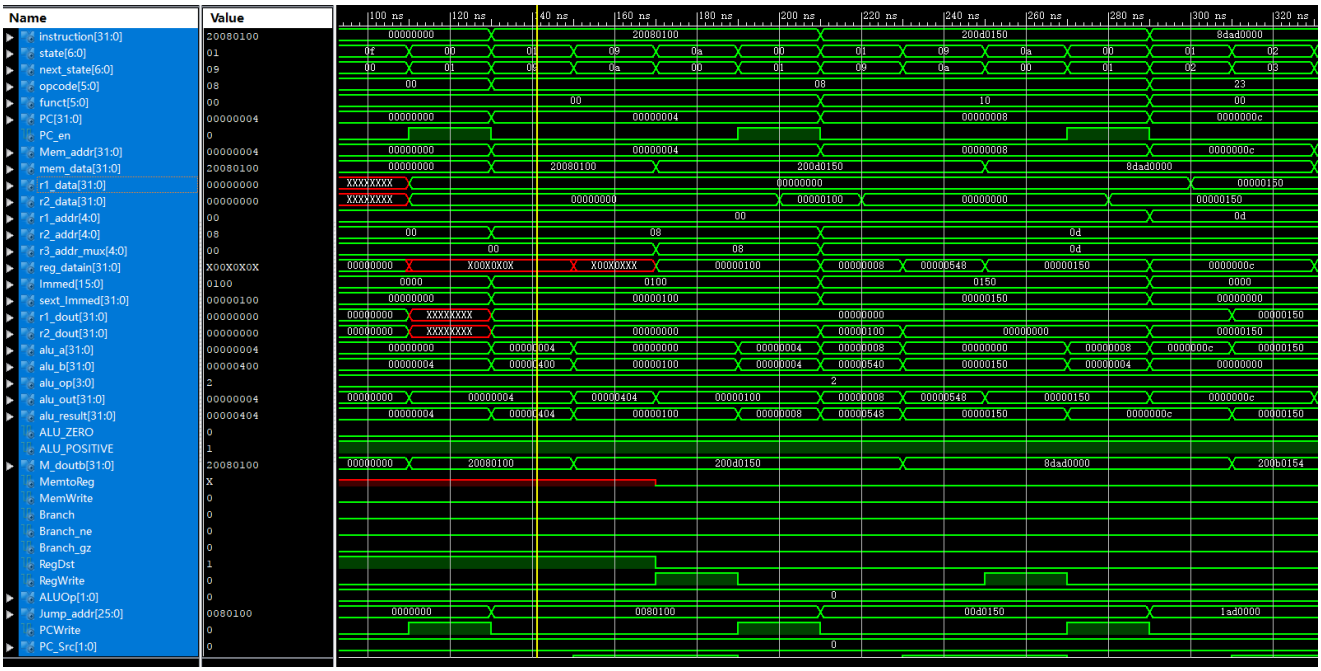
开始计算前的内存布局：

	0	1	2	3
0x0	537395456	537723216	-1918042112	537592148
0x4	-1922367488	537657684	-1920204796	-1391788032
0x8	-1391722492	564789246	-1928658944	-1928593404
0xC	23875616	-1391853560	554172420	556400639
0x10	488701945	134217745	0	0
0x14	0	0	0	0
0x18	0	0	0	0
0x1C	0	0	0	0
0x20	0	0	0	0
0x24	0	0	0	0
0x28	0	0	0	0
0x2C	0	0	0	0
0x30	0	0	0	0
0x34	0	0	0	0
0x38	0	0	0	0
0x3C	0	0	0	0
0x40	3	3	0	0
0x44	0	0	0	0
0x48	0	0	0	0
0x4C	0	0	0	0
0x50	0	0	0	0
0x54	20	3	3	0
0x58	0	0	0	0
0x5C	0	0	0	0
0x60	0	0	0	0

运行完毕时的内存布局：

	0	1	2	3
0x0	537395456	537723216	-1918042112	537592148
0x4	-1922367488	537657684	-1920204796	-1391788032
0x8	-1391722492	564789246	-1928658944	-1928593404
0xC	23875616	-1391853560	554172420	556400639
0x10	488701945	134217745	0	0
0x14	0	0	0	0
0x18	0	0	0	0
0x1C	0	0	0	0
0x20	0	0	0	0
0x24	0	0	0	0
0x28	0	0	0	0
0x2C	0	0	0	0
0x30	0	0	0	0
0x34	0	0	0	0
0x38	0	0	0	0
0x3C	0	0	0	0
0x40	3	3	6	9
0x44	15	24	39	63
0x48	102	165	267	432
0x4C	699	1131	1830	2961
0x50	4791	7752	12543	20295
0x54	20	3	3	0
0x58	0	0	0	0
0x5C	0	0	0	0
0x60	0	0	0	0

前两条指令执行的波形图：



源代码

文件太多，代码位于github上：

https://github.com/songchaow/MIPS_CPU_Design/tree/master/Lab06_Multicycle_CPU