50条指令报告

CPU功能

- 实现了50条指令所要求的全部功能
- **额外增加了分支预测，使得引入分支延时槽后所有分支指令都没有阻塞周期**

实现思路

主要部件的思路已在10条指令的报告中阐述，这里主要分析新增部件的思路。

分支预测

由于程序默认开启了分支延时槽，且我的程序判断分支在ID阶段，因此若ID阶段的beq等分支指令需要等待前一条指令的运算结果时，则需要阻塞一个周期。利用这个周期的办法是，**若beq等分支指令有数据依赖，则直接判定分支会跳转，等到延时槽指令装载后，则可以让beq在EX阶段重新计算正确的分支方向并取指和修改下一条指令的IP；若没有数据依赖，则直接在ID阶段计算出正确的分支方向并修改IP。** 使用此方法使得所有分支指令均不再有阻塞周期。

乘除法操作器

模块实例化中，传入的两个操作数是进入ALU时的两个操作数：

```
MultiplicationDivisionUnit MDU(.reset(reset), .clock(clock),
.operand1(nextEXtoMEM.RsValue), .operand2(nextEXtoMEM.ALUInput), .operation(IDtoEX.signals.Mduop),
.start(IDtoEX.signals.MduStart & (IDtoEX.signals.Mduop > MDU_WRITE_LO)), .busy(Mdubusy), .dataRead(Mduresult));
```

在冒险检测单元判断是否因为乘除法器而阻塞：

```
if ((Mdubusy & IDtoEX.signals.MduStart) == 1 )
     Mduflush = 1;
else
     Mduflush = 0;
```

相对应地，如果阻塞了就要将指令阻塞在EX级：

```
PipelineRegister IDEX(.reset(reset | flush), .clock(clock), .in(Mduflush ? nextEXtoMEM : nextIDtoEX), .out(IDtoEX) );
PipelineRegister EXMEM(.reset(reset | Mduflush), .clock(clock), .in(nextEXtoMEM), .out(EXtoMEM) );
```

乘除法的结果在后续的使用与其他运算结果基本一致，不详细展开论述。

内存对齐

在ControlUnit中发送MemSize和RegSize表示读写的大小，然后根据这个大小进行相应的调整，下面代码以读取指令为例：

```
    always @(*) begin
        nextMEMtoWB = EXtoMEM;
        if(EXtoMEM.signals.MemoryToReg) begin
            if(EXtoMEM.signals.MemSize == 'b01) begin //LB  or LBU
                if(EXtoMEM.signals.MemExt) begin //LB
                    nextMEMtoWB.RdValue = {{24{tempB[7]}}, tempB};
                end
                else begin //LBU
                    nextMEMtoWB.RdValue = {{24'b0}, tempB};
                end
            end
            else if(EXtoMEM.signals.MemSize == 'b10) begin //LH  or LHU
                if(EXtoMEM.signals.MemExt) begin //LH
                    nextMEMtoWB.RdValue = {{16{tempH[15]}}, tempH};
                end
                else begin //LHU
                    nextMEMtoWB.RdValue = {{16'b0}, tempH};
                end
            end
            else if(EXtoMEM.signals.MemSize == 'b11) //LW
                nextMEMtoWB.RdValue = readResult;
        end
    end
```

测试结果

以第一次syscall指令结束前的所有输出为标准，经过手工对比发现同mars生成结果完全一致。

各测试样例所需周期数如下图：

	A	B
1	名称	指令数/周期数（第一次遇到syscall）
2	add-1	40011/40017
3	add-2	25013/25019
4	add-4	22515/22521
5	add-8	21265/21271
6	floyd	8162/10167
7	gcd-hardmul	268/411
8	gcd-softmul	255/262
9	lfsr	8016/8021
10	pointer-chasing	11354/11370

随机测试的360个样例也与标准输出一致：