### 各个Stage的工作

Pipeline有多种可能的实现，表 1说明了本模拟器机制下各个Stage的工作。

表 1 各个Stage的工作

|  |  |
| --- | --- |
| Stage | 工作 |
| IF | 获取指令并解析。更新pc寄存器，对于Branch指令需做出分支预测。 |
| ID | 读取src寄存器或Forwarding数据，获取正确的右值或发起Stall。 |
| EX | 执行计算或分支判断，在分支预测失败时修改pc寄存器，并发起Bubble。 |
| MEM | 对于Load和Store指令，执行内存读写。 |
| WB | 将指令结果写入dst寄存器。 |

### 指令集类的设计

对于每一条指令，需要五个执行函数pc\_modify、inst\_decode、execute、mem\_access、write\_back。为了用统一的代码调度不同指令的执行函数，利用C++多态特性，将指令封装为类，继承自统一的父类Inst，并将五个执行函数设计为Inst类的虚函数。相同类别下的不同指令的一些执行函数是相同的，利用继承机制实现去重。

为了实现Forwarding机制，在Inst类中添加了forward虚函数，用于提供Forwarding数据（或发起Stall）。并对有读寄存器需求的指令类添加get\_fwd函数,以请求Forwarding数据。

### Parser的设计

获取二进制代码后，利用Inst::parse函数解析opcode，判断指令大类，然后将二进制代码传入RTypeInst::parse、ITypeInst::parse等函数进一步解析。

### 简化Pipeline Register

在Pipeline运行的过程中，一些数据被存入Pipeline Register，以使得IF以外的四个Stage可以获得运行所需的数据，这些数据主要包括

1. 除IF以外的Stage正在运行的指令
2. （某个Stage正在运行Branch指令或JALR）当前指令的pc、预测的下一指令pc
3. （某个Stage正在运行需要寄存器读取的指令）ID阶段读取到的右值
4. （某个Stage正在运行需要计算的指令）EX阶段的计算结果
5. （某个Stage正在运行Load指令）MEM阶段内存读取结果

在软件模拟Pipeline时，为了简化代码，提高运行速度，将这些数据视作指令类的成员变量。创建Inst \* inst[5]数组，每个Stage通过Inst \*指向的对象了解正在执行的指令类型，并获得执行指令所需的数据。

### Forwarding

利用两个函数forward和get\_fwd实现了Forwarding机制。其中，forward函数提供Forwarding数据，get\_fwd函数请求Forwarding数据。

bool Inst::forward(Stage stage, unsigned src, unsigned & rval)

当前指令所处的Stage为stage，请求Forwarding的指令需要从编号为src的寄存器中读取右值，当前指令提供的Forwarding数据存入rval，返回的bool类型值表示是否发生了Forwarding。在Load指令的forward函数中，若stage==EX，则发起Stall。

void SrcInst::get\_fwd(unsigned src, unsigned & rval)

当前指令需要从编号为src的寄存器中读取右值，该右值应存入rval。该函数会依次向处于EX、MEM、WB的指令请求Forwarding，一旦某一请求成功，函数结束。

### 分支预测

Predictor类实现了二级自适应分支预测器，分支预测准确率实验结果如表 2所示。

表 2 分支预测准确率

|  |  |
| --- | --- |
| 测试数据 | 准确率 |
| array\_test1 | 54.55% |
| array\_test2 | 50.00% |
| basicopt1 | 98.89% |
| bulgarian | 94.50% |
| expr | 76.58% |
| gcd | 61.67% |
| hanoi | 86.48% |
| lvalue2 | 66.67% |
| magic | 82.20% |
| manyarguments | 80.00% |
| multiarray | 68.52% |
| naive | 未执行分支 |
| pi | 84.51% |
| qsort | 91.74% |
| queens | 81.01% |
| statement\_test | 66.83% |
| superloop | 95.19% |
| tak | 75.70% |