计算机系统结构

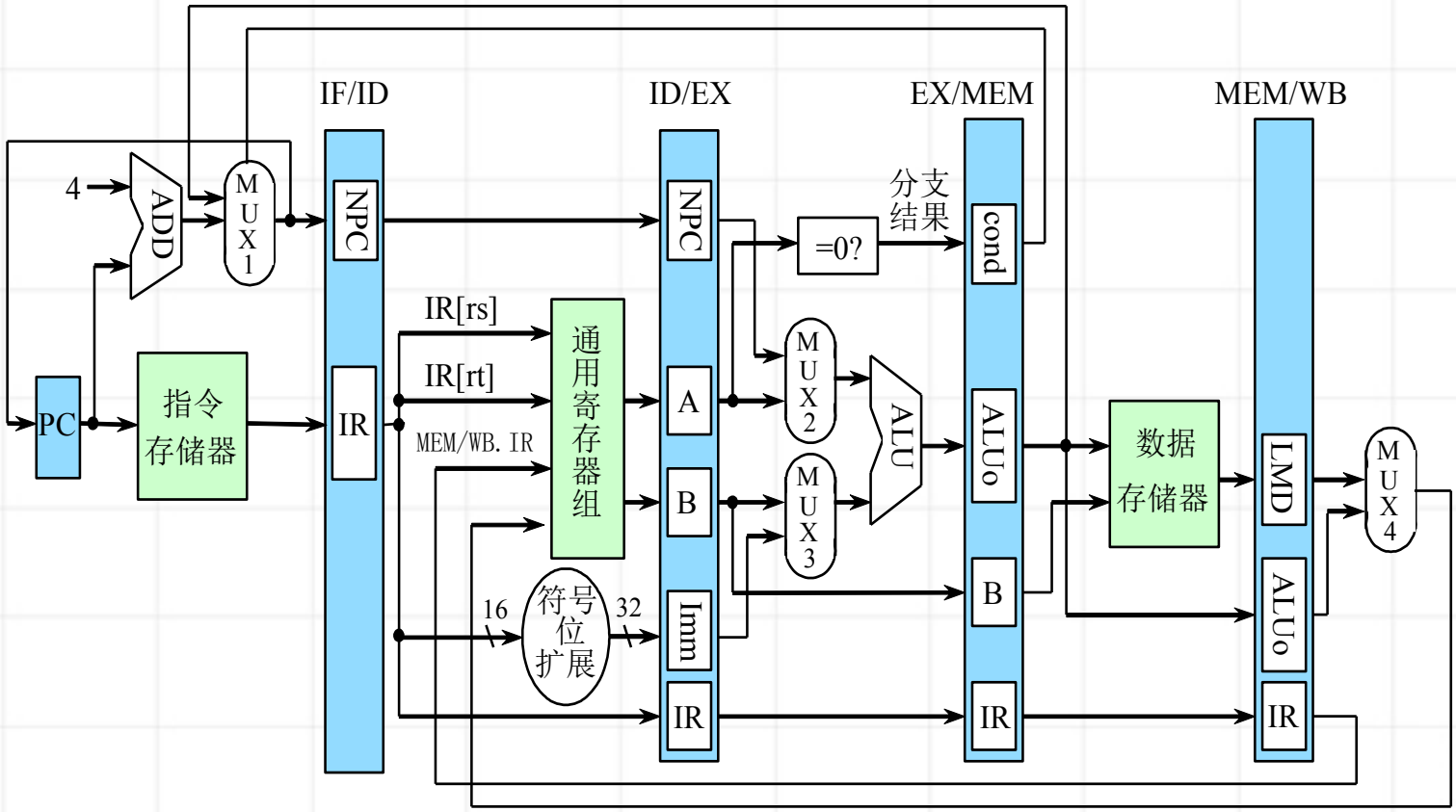
第九讲流水线的实现（2）

冯丹

武汉光电国家研究中心

华中科技大学
HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

1.MIPS的流水化



流水实现的数据通路

2.增加的多路选择器MUX1

- MUX1的功能

```
if （（EX/MEM.IR[op]==“分支指令”）&
EX/MEM.cond）{
        MUX1_output＝EX/MEM.ALUo
};
else
        MUX1_output＝PC+4; //MUX1_output表示
MUX1的输出
```

3.增加的多路选择器MUX2

- MUX2的功能

```
if （ID/EX.IR[op]==“分支指令”）{
        MUX2_output＝ID/EX.NPC
};
else
        MUX2_output＝ID/EX.A; //MUX2_output表示MUX2的输出
```

4.增加的多路选择器MUX3

- MUX3的功能

```
if （ID/EX.IR[op]==“寄存器－寄存器型ALU指令”）
{
        MUX3_output＝ID/EX.B
};
else
        MUX3_output＝ID/EX.Imm; //MUX3_output表示MUX3的输出
```

5.增加的多路选择器MUX4与MUX5

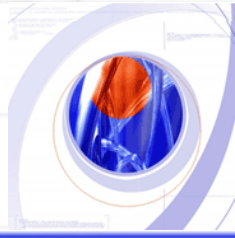
- MUX4的功能

```
if （MEM/WB.IR[op]==“load”）{
        MUX4_output＝MEM/WB.LMD
};
else
        MUX4_output＝MEM/WB.ALUo //MUX4_output表示MUX4的输出
```

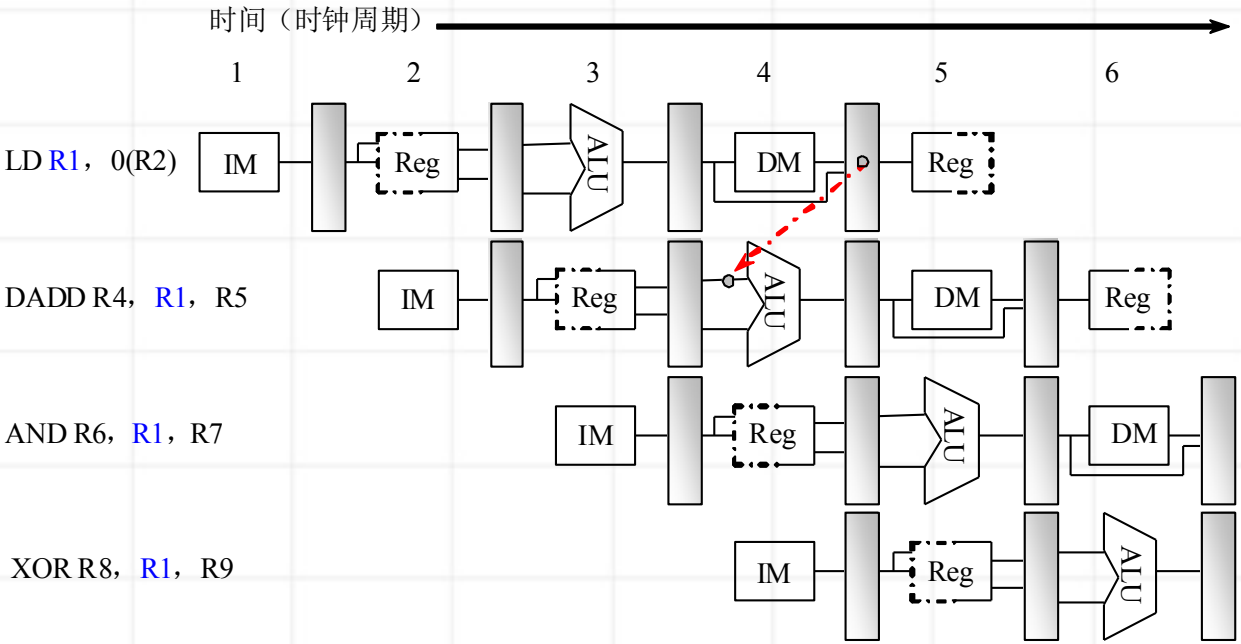
- 目标寄存器编号怎么确定？MUX5
  - MEM/WB.IR[rd]和MEM/WB.IR[rt]二选一
  - 寄存器－寄存器型ALU指令：选择MEM/WB.IR[rd]
  - 寄存器－立即数型ALU指令和load指令：选择MEM/WB.IR[rt]

6.数据冲突

- 所有的数据冲突均可以在ID段检测到
  - 如果存在数据冲突，就在相应的指令流出ID段之前将之暂停。
  - 完成该工作的硬件称为流水线的互锁机制。
- 在ID段确定需要什么样的定向，并设置相应的控制
  - 降低流水线的硬件复杂度。（不必挂起已经改变了机器状态的指令）
- 也可以在使用操作数的那个时钟周期的开始检测冲突和确定必需的定向
- 检测冲突是通过比较寄存器地址是否相等来实现的



7. Load引起的流水线冲突实例



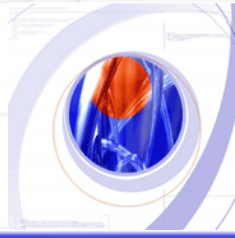
8. 实现load互锁（1）——检测

- 由于使用load的结果而引起的流水线互锁称为load互锁
- 在ID段检测是否存在RAW冲突（这时load指令在EX段）

ID/EX中的操作码 （ID/EX.IR[op]）	IF/ID中的操作码 （IF/ID.IR[op]）	比较的操作数字段
load	RR ALU	ID/EX.IR[rt]=IF/ID.IR[rs]
load	RR ALU	ID/EX.IR[rt]=IF/ID.IR[rt]
load	load、store ALU立即数或分支	ID/EX.IR[rt]=IF/ID.IR[rs]

9. 实现load互锁（2）——停顿

- 若检测到RAW冲突，流水线互锁机制必须在流水线中插入停顿，并使当前正处于IF段和ID段的指令不再前进
  - 将ID/EX.IR中的操作码改为全0（全0表示空操作）
  - IF/ID寄存器的内容回送到自己的入口



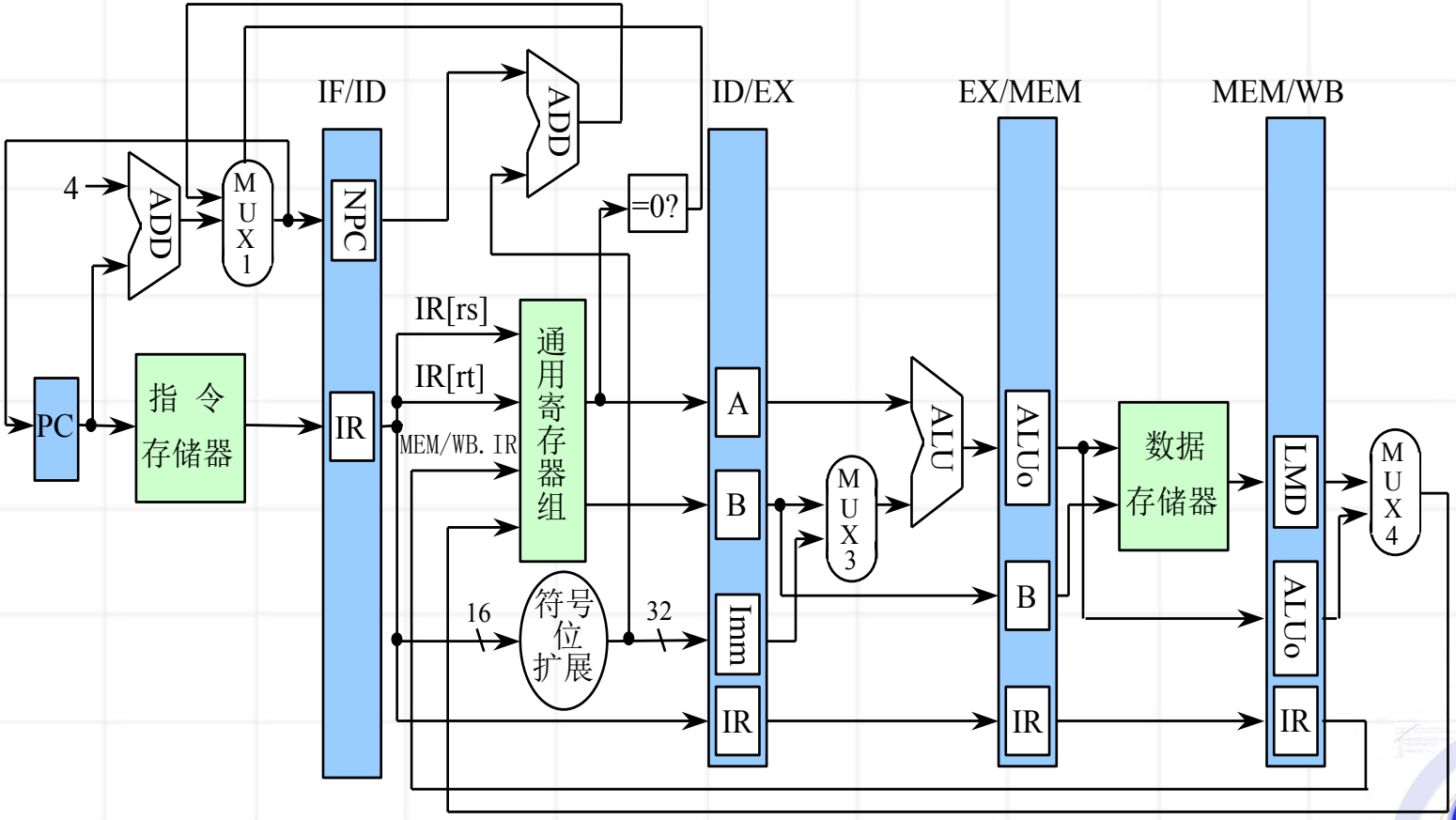
10.定向逻辑

- 要考虑的情况更多
- 通过比较流水寄存器中的寄存器地址来确定
- 例如：
  - 若：(ID/EX.IR.op==RR ALU)&（EX/MEM.IR.op==RR ALU）&（ID/EX.IR[rt]==EX/MEM.IR[rd]）
  - 则：EX/MEM.ALUo定向到ALU的下面一个输入
- 又如：
  - 若：（ID/EX.IR[op]==RR ALU）&（MEM/WB.IR[op]==load）&（ID/EX.IR[rt]==MEM/WB.IR[rt]）
  - 则：把MEM/WB.LMD定向到ALU的下面一个输入

11.控制冲突

- 分支指令的条件测试和分支目标地址计算是在EX段完成，对PC的修改是在MEM段完成。
- 它所带来的分支延迟是3个时钟周期。
- 减少分支延迟（把上述工作提前到ID段进行）
  - 在ID段增设一个加法器：计算分支目标地址
  - 把条件测试“=0？”的逻辑电路移到ID段
  - 这些结果直接回送到IF段的MUX1

12. 改进后对分支指令的处理



13.取指令周期（IF）

– 分支指令的操作

IF/ID.IR ← Mem[PC]；

IF/ID.NPC, PC ← (

if((IF/ID[op]= =branch)&(Regs[IF/ID.IR[rs]] = = 0)) {

IF/ID.NPC + (IF/ID.IR16)16 ## （IF/ID.IR16..31 << 2）

}

else {

PC+4}

);

14.指令译码/读寄存器周期（ID）

- 分支指令的操作

  ID/EX.A ←Regs[IF/ID.IR[rs]]；

  ID/EX.B← Regs[IF/ID.IR[rt]]；

  ID/EX.IR ← IF/ID.IR；

  ID/EX.Imm ← （ IF/ID.IR16 ）16 ## IF/ID. IR16..31；

谢谢大家