RISC处理器设计

胡伟武

复　习（1）

- **计算机中数的表示**
  - **二进制，CMOS电路用电压高低表示　0　和　1**
  - **其它表示方法：超导计算机、量子计算机、分子计算机**
- **CMOS门电路**
  - **电路原理**
  - **延迟模型**

复 习（2）

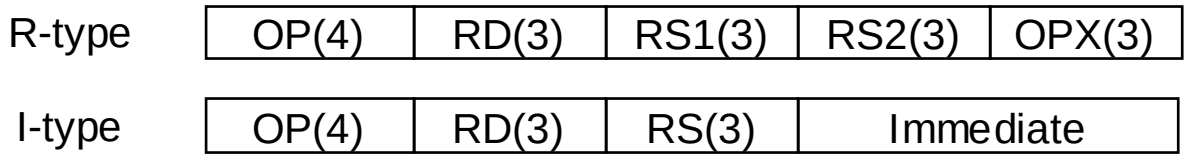
- **组合逻辑**
  - **逻辑表达式、真值表、卡诺图、逻辑图**
  - **常见逻辑电路：译码器、选择器、ALU电路**
- **时序逻辑**
  - **寄存器原理**
  - **时序逻辑电路的状态转换**

指令流水线

- **一个简单的CPU**
- **指令流水线的概念**
- **指令的相关性**

一个简单的CPU

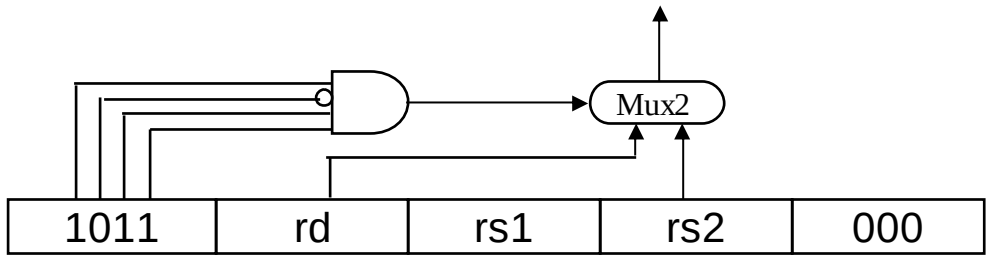
- **基本特征**
  - **RISC结构**
  - **16位指令及数据**
  - **8个通用寄存器，0号GPR恒为0。**
  - **一个定点ALU**
- **指令格式**
  - **寄存器型**
  - **立即数型**



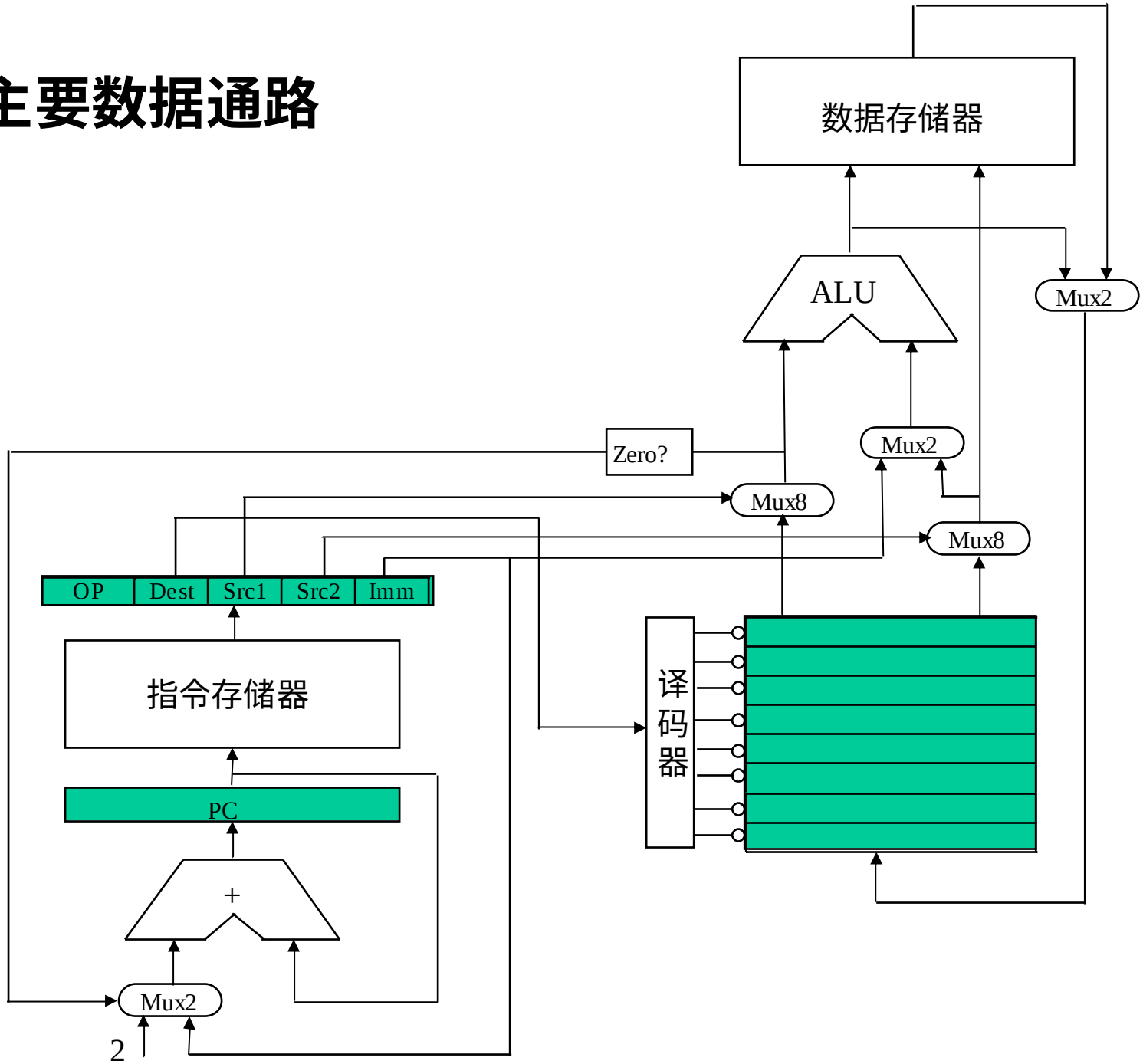
l 基本指令

ADD	0001	rd	rs1	rs2	000
SUB	0010	rd	rs1	rs2	000
AND	0011	rd	rs1	rs2	000
OR	0100	rd	rs1	rs2	000
NOT	0101	rd	rs1	rs2	000
SL	0110	rd	rs1	rs2	000
SR	0111	rd	rs1	rs2	000
SRU	1000	rd	rs1	rs2	000
ADDI	1001	rd	rs1	imm	
LD	1010	rd	base	offset	
ST	1011	rd	base	offset	
BZ	1100	rd	000	offset	

- **关于ST指令**
  - **在ST指令中，目标寄存器域rd保存的是要存储的源寄存器号。为此，我们对IR的输出做如下处理，即当操作码为ST时，rs2的输出来自rd。这个逻辑作为IR的一部分，以后不再专门指出。**



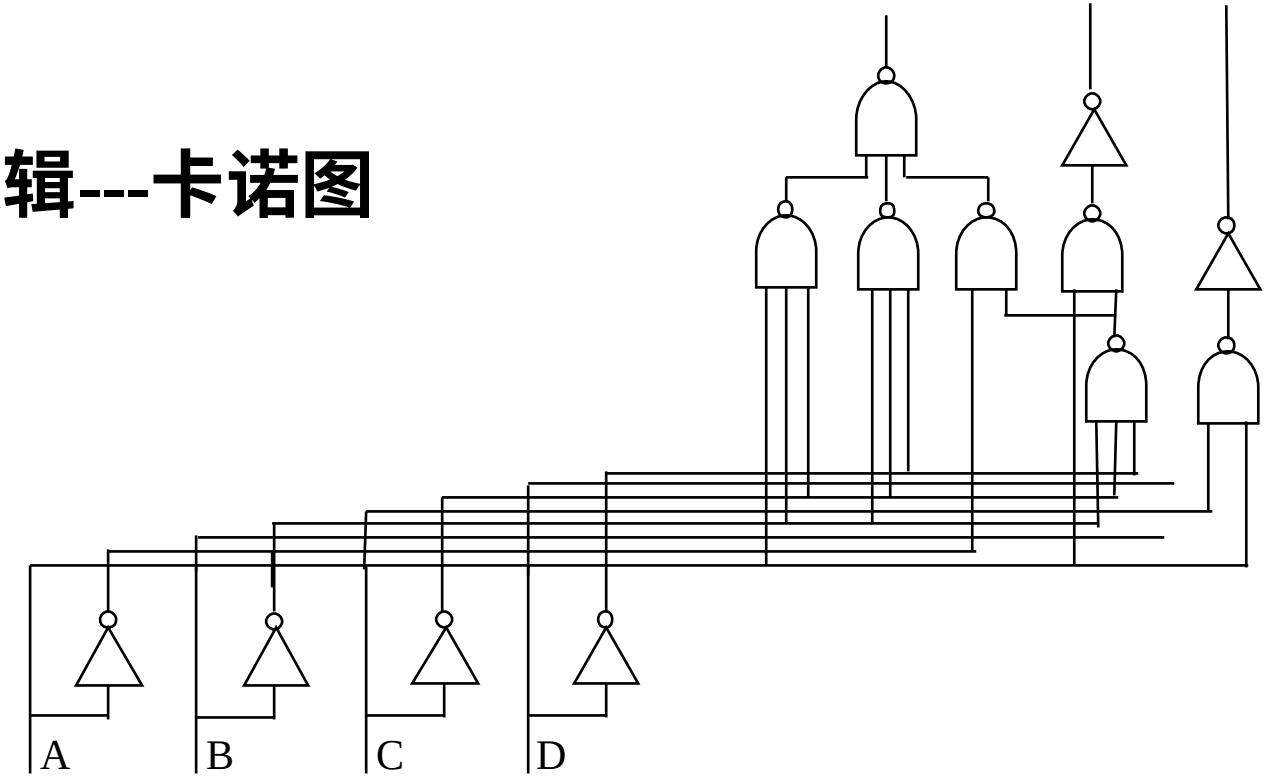
l 主要数据通路


数据存储器
ALU
Mux2
Zero?
Mux2
Mux8
Mux8
OP
Dest
Src1
Src2
Imm
指令存储器
译码器
PC
+
Mux2
2

l 主要控制逻辑---真值表

指令	OP	译码使能	操作数 2 选 1	结果 2 选 1	PC 2 选 1	ALU 控制
	0000	0	X	X	0	XXX
ADD	0001	1	0	0	0	000
SUB	0010	1	0	0	0	001
AND	0011	1	0	0	0	010
OR	0100	1	0	0	0	011
NOT	0101	1	X	0	0	100
SL	0110	1	0	0	0	101
SR	0111	1	0	0	0	110
SRU	1000	1	0	0	0	111
ADDI	1001	1	1	0	0	000
LD	1010	1	1	1	0	000
ST	1011	0	1	X	0	000
BZ	1100	0	1	X	zero	XXX
	1101	0	X	X	0	XXX
	1110	0	X	X	0	XXX
	1111	0	X	X	0	XXX

l 主要控制逻辑---卡诺图



^C \ ^A A	00	01	11	10	
00	0	1	0	1	^D
01	1	1	0	1	D
11	1	1	0	0	
10	1	1	0	1	^D
C	^B	B	^B		

译码器使能:

^AB+^AD+^AC+A^B^C+^BC^D

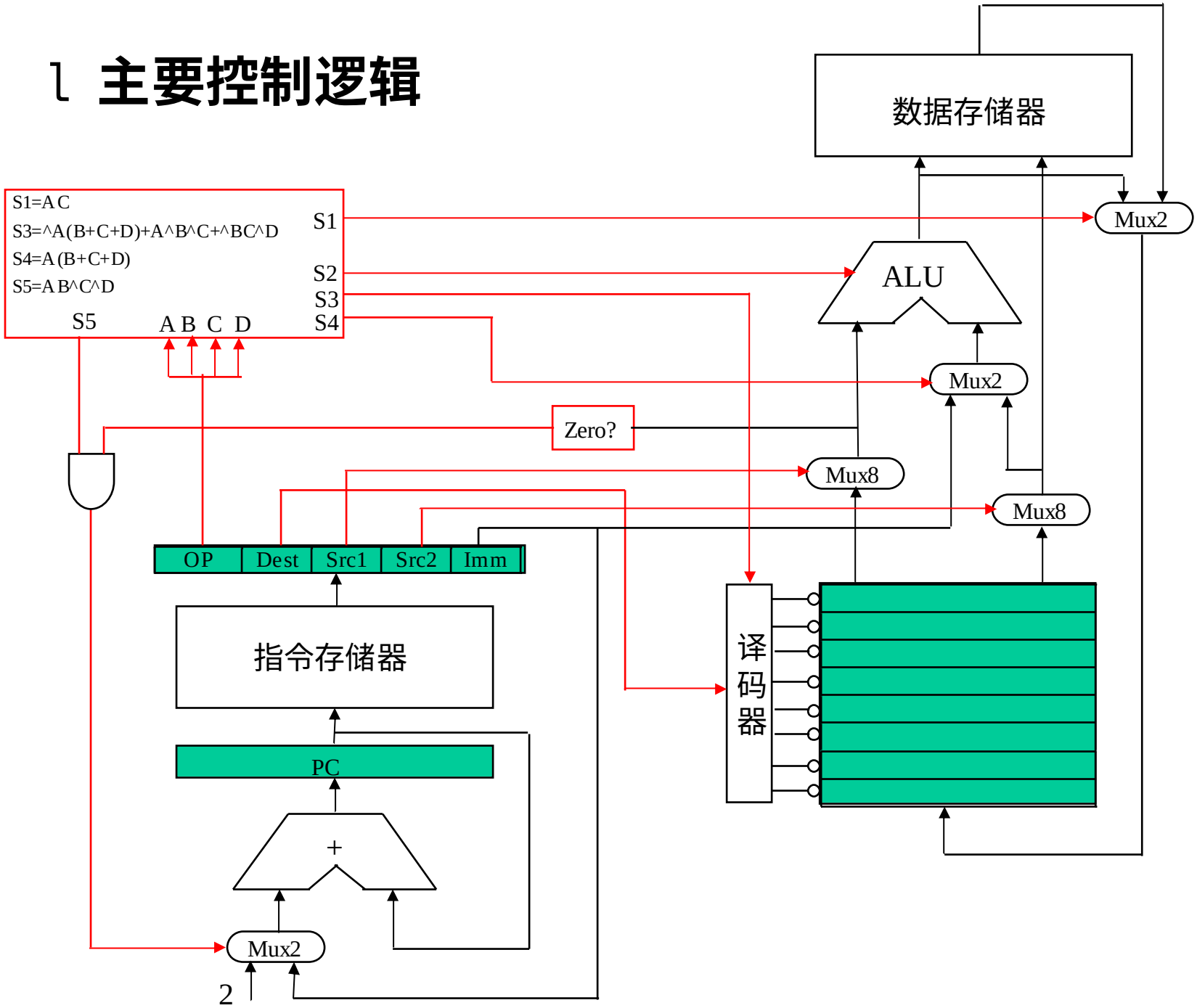
^C \ ^A A	00	01	11	10	
00	X	0	1	0	^D
01	0	X	X	1	D
11	0	0	X	1	
10	0	0	X	1	^D
C	^B	B	^B		

操作数2选1: AB+AD+AC

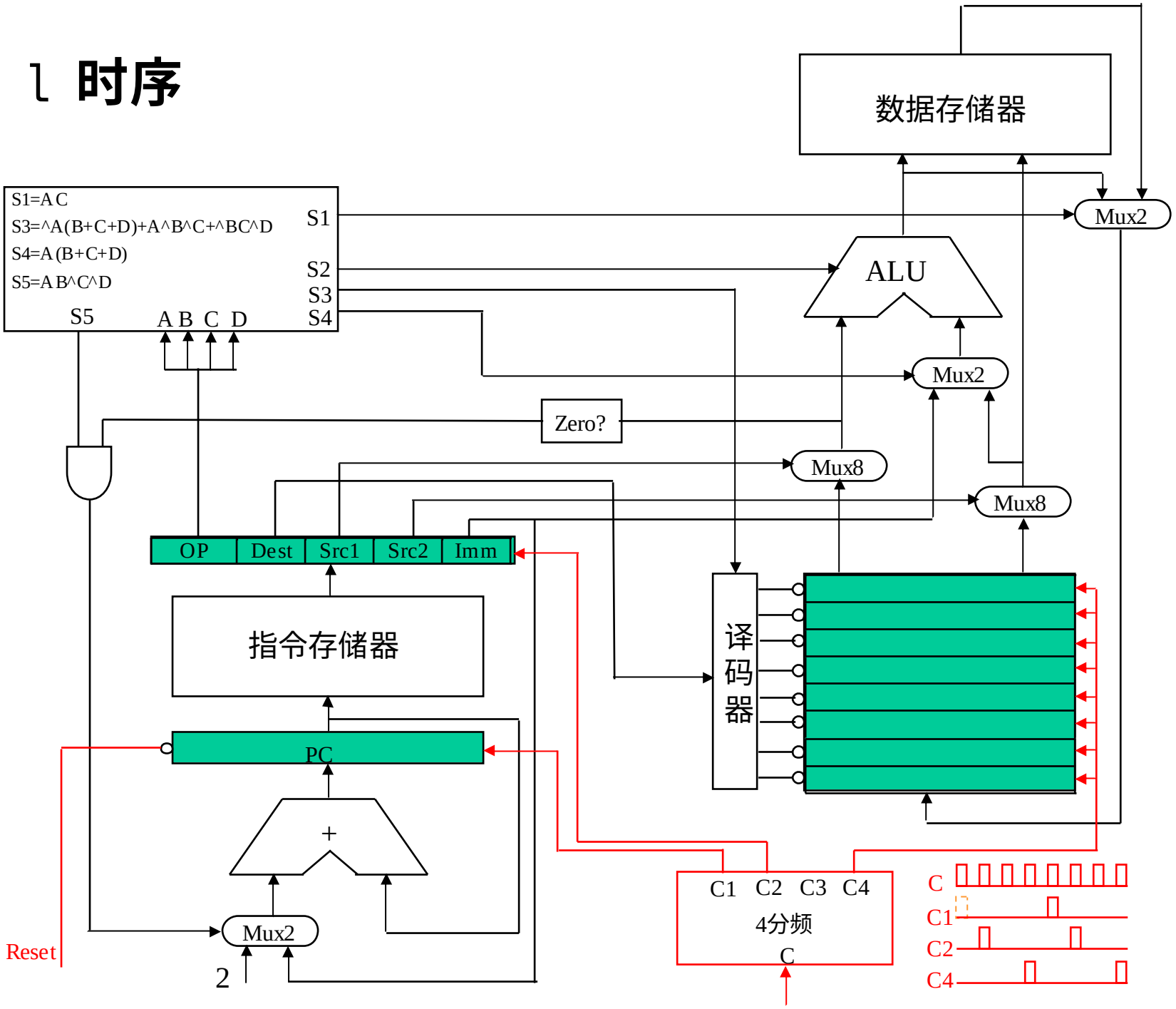
^C \ ^A A	00	01	11	10	
00	X	0	X	0	^D
01	0	0	X	0	D
11	0	0	X	X	
10	0	0	X	1	^D
C	^B	B	^B		

结果2选1: AC

l 主要控制逻辑


数据存储器
S1=A C
S3=^A(B+C+D)+A^B^C+^BC^D
S4=A(B+C+D)
S5=A B^C^D
S1
S2
S3
S4
S5
A B C D
Mux2
ALU
Mux2
Zero?
Mux8
Mux8
OP
Dest
Src1
Src2
Imm
指令存储器
译码器
PC
+
Mux2
2

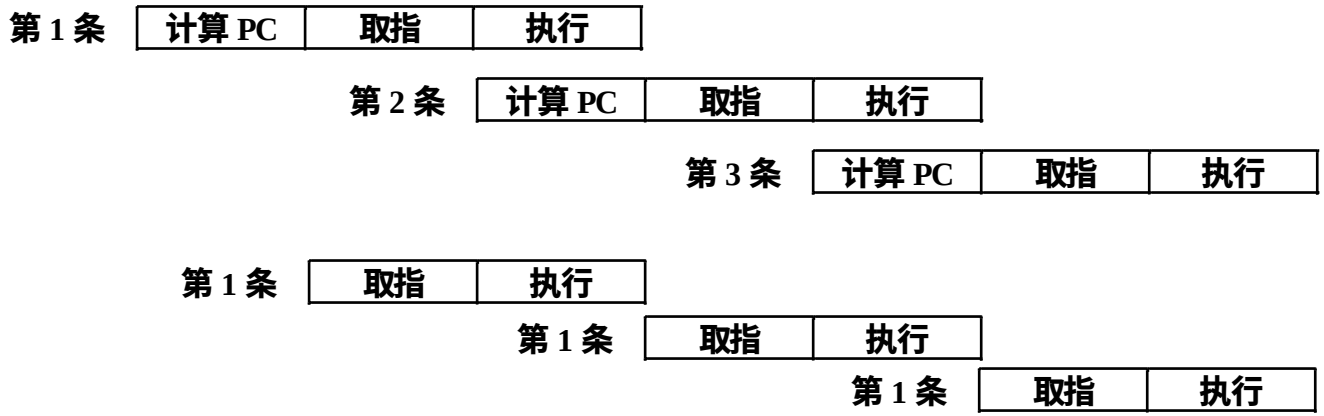
l 时序


数据存储器
S1=A C
S3=^A(B+C+D)+A^B^C+^BC^D
S4=A (B+C+D)
S5=A B^C^D
S1
S2
S3
S4
S5
A B C D
Mux2
ALU
Mux2
Zero?
Mux8
Mux8
OP
Dest
Src1
Src2
Imm
指令存储器
译码器
PC
+
Mux2
2
Reset
C1 C2 C3 C4
4分频
C
C
C1
C2
C4

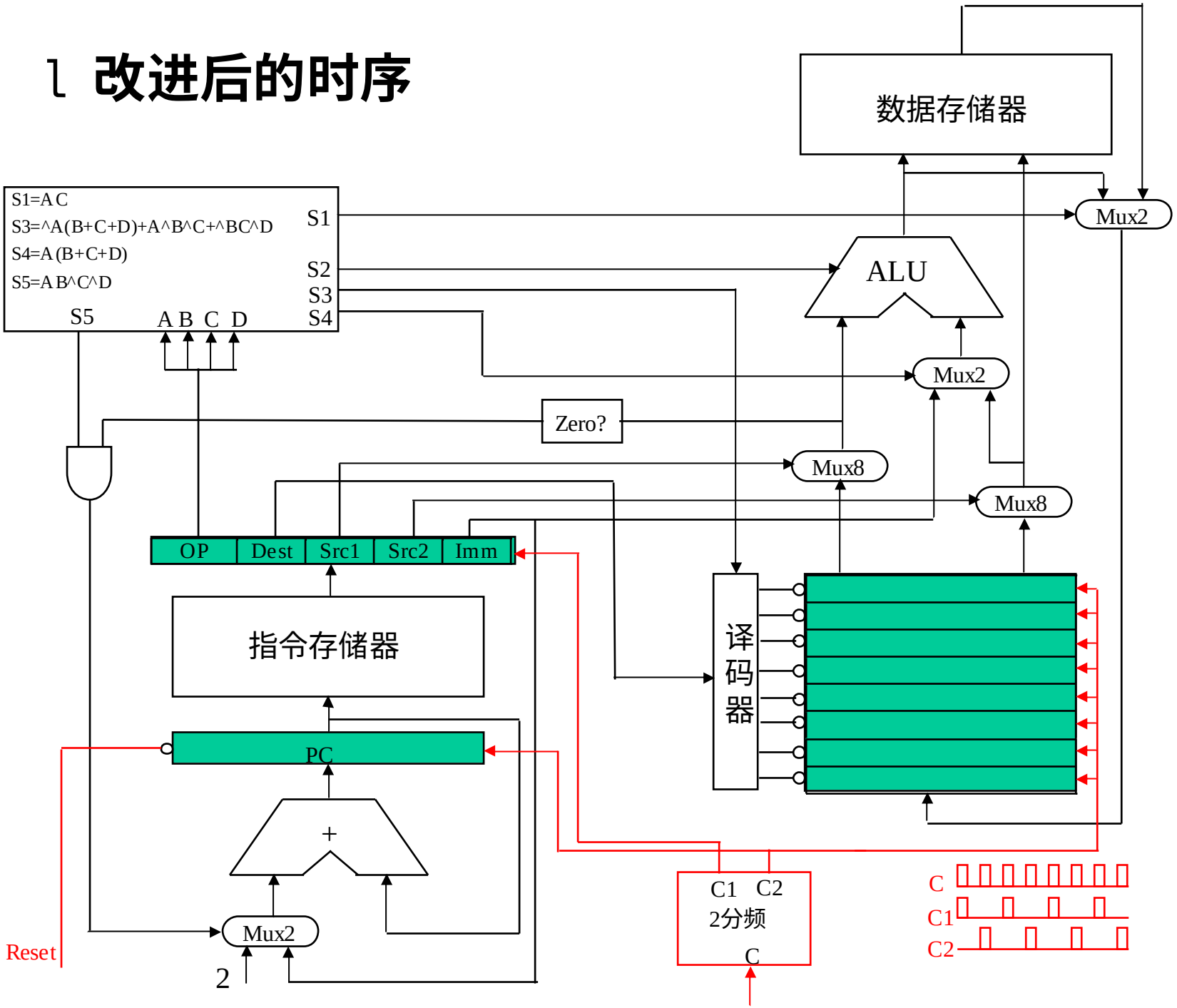
时序的改进

- 上述时序的三个步骤
  - 指令地址送到PC、取指到IR、计算结果到GPR

- 可以合并为两个
  - 计算下一条指令的PC和指令执行重叠
  - 可以把计算下一拍PC值作为指令执行的一部分
    - 转移指令的运算结果是PC的值

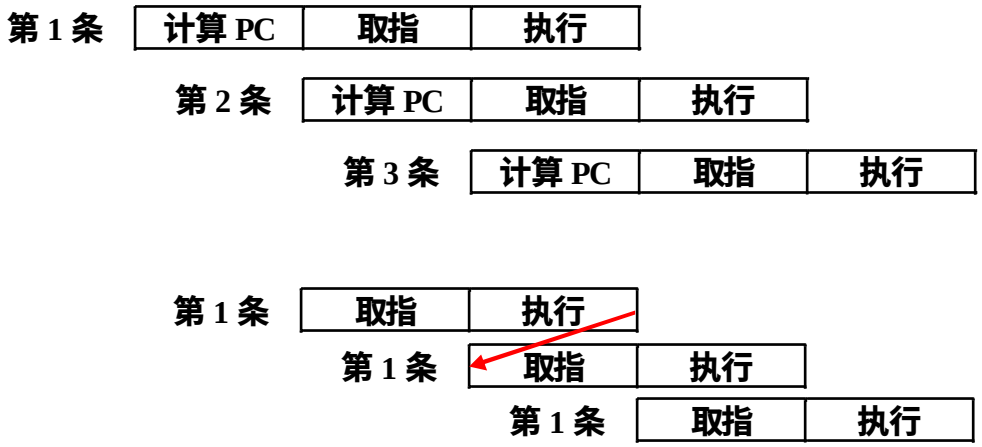


l 改进后的时序

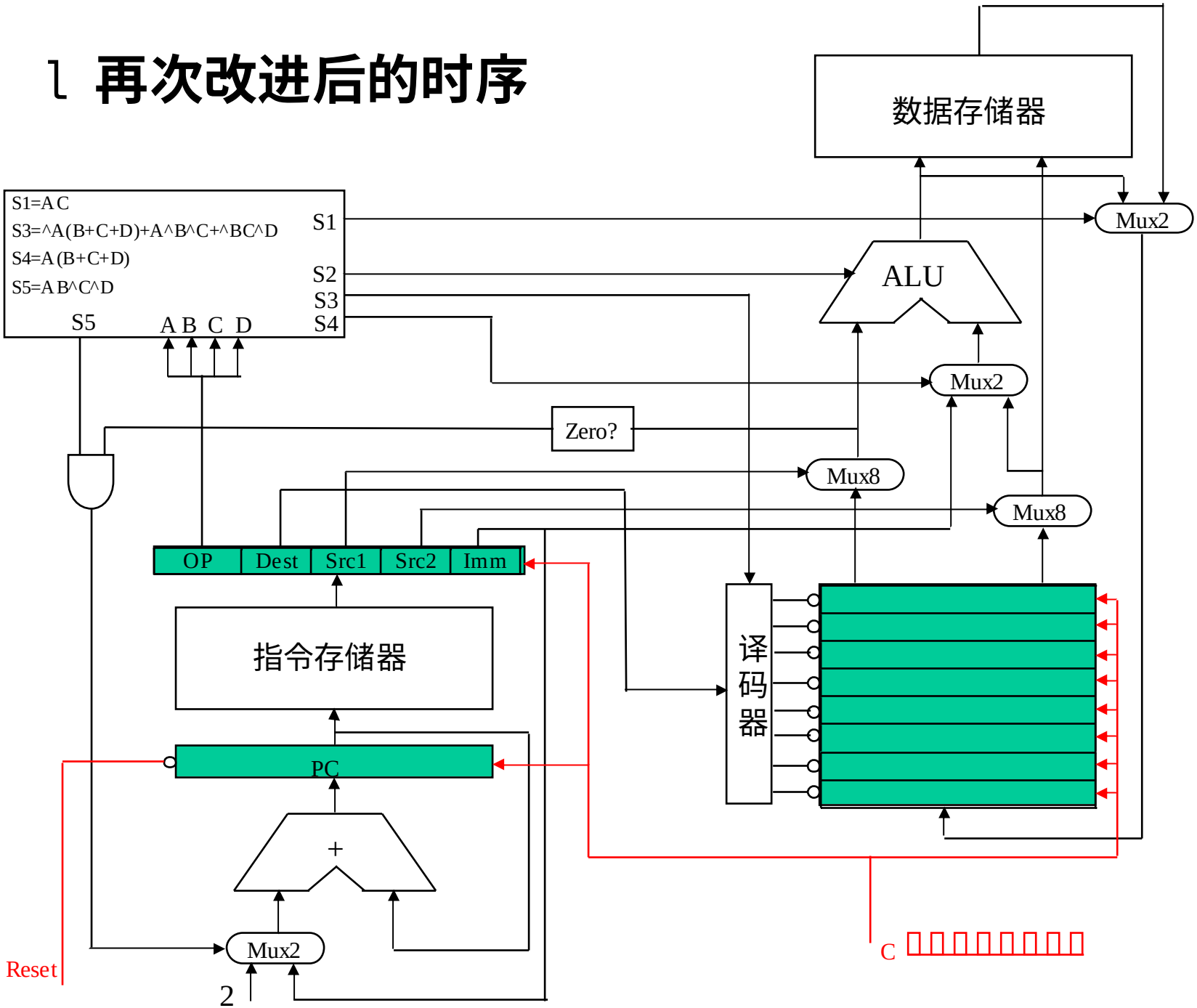

数据存储器
S1=A C
S3=^A(B+C+D)+A^B^C+^BC^D
S4=A (B+C+D)
S5=A B^C^D
S1
S2
S3
S4
S5
A B C D
Mux2
ALU
Mux2
Zero?
Mux8
Mux8
OP
Dest
Src1
Src2
Imm
指令存储器
译码器
PC
+
Mux2
2
Reset
C1 C2
2分频
C
C
C1
C2

能否把取指和运算也重叠？

- **大多数情况可以重叠**
  - **第n+1条指令执行时，第n条指令已经执行完，因此第n+1条指令可以用到第n条指令的结果。**
- **但有一个例外**
  - **如果第n+1条指令的取指也要用到第n条指令的结果，则第n+1条指令的取指必须等到第n条指令结束后才能执行。**
  - **正是Branch指令的情况，可以用delay slot解决这个问题**



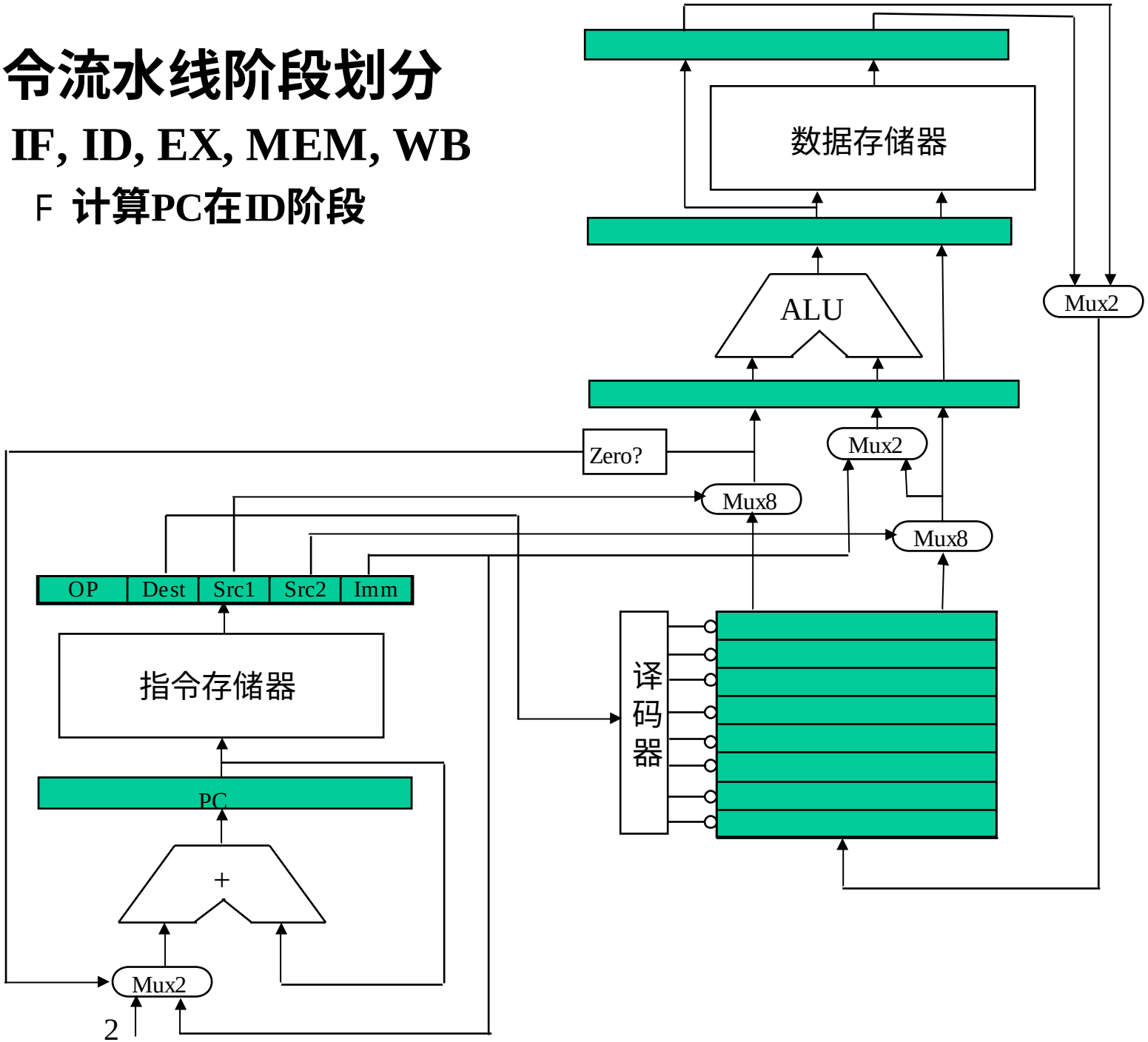
l 再次改进后的时序


数据存储器
S1=A C
S3=^A(B+C+D)+A^B^C+^BC^D
S4=A (B+C+D)
S5=A B^C^D
S1
S2
S3
S4
S5
A B C D
Mux2
ALU
Mux2
Zero?
Mux8
Mux8
OP
Dest
Src1
Src2
Imm
指令存储器
译码器
PC
+
Mux2
2
Reset
C

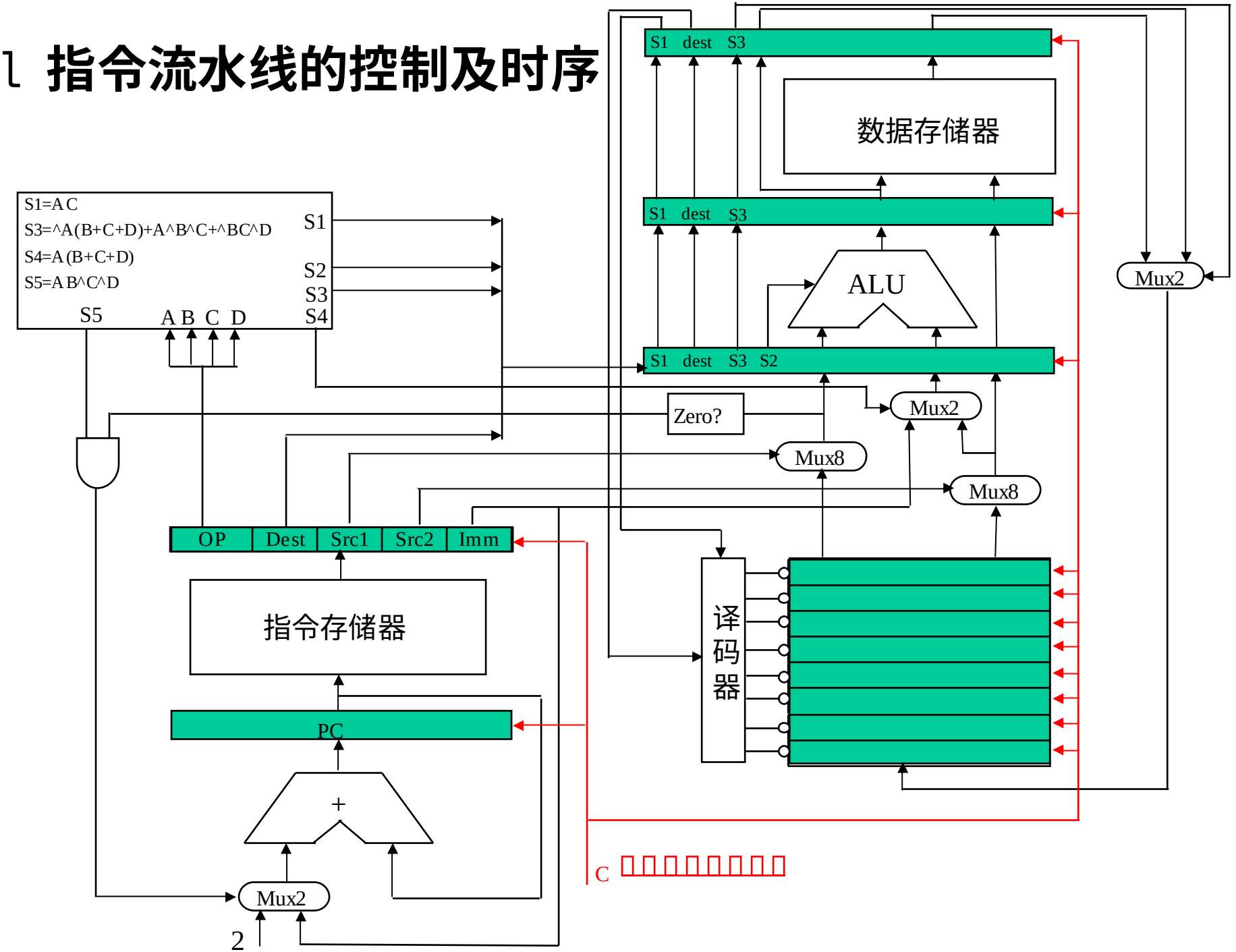
上述流水线的进一步改进

- **上述流水线是两级流水线**
  - **取指和执行**
  - **执行阶段做的事情较多**
    - **译码（包括读取寄存器的值）**
    - **运算（ALU操作）**
    - **访存（取数或存数）**
    - **写回到寄存器**
  - **时钟周期较长，一拍内必须做完上述四件事情**
- **可以把执行阶段再细分**
  - **执行阶段分成译码、运算、访存、写回**

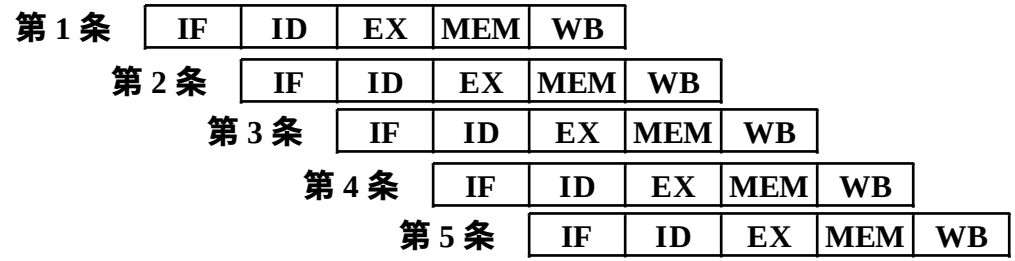
- 指令流水线阶段划分
  - IF, ID, EX, MEM, WB
    - 计算PC在ID阶段



l 指令流水线的控制及时序

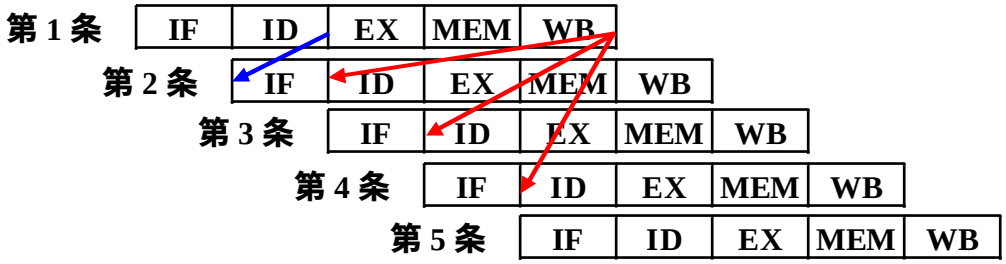


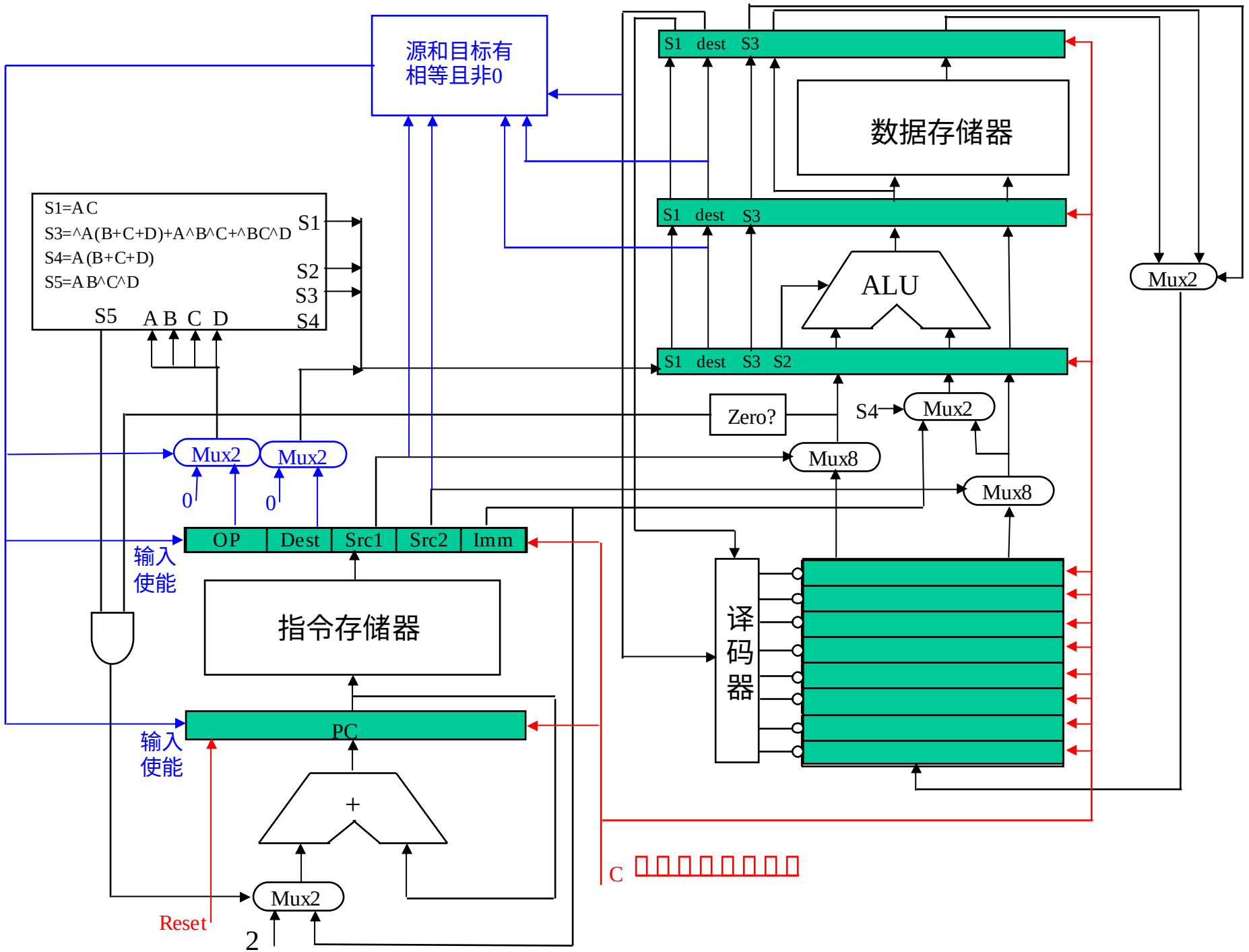
标准指令流水线


第 1 条
IF
ID
EX
MEM
WB
第 2 条
IF
ID
EX
MEM
WB
第 3 条
IF
ID
EX
MEM
WB
第 4 条
IF
ID
EX
MEM
WB
第 5 条
IF
ID
EX
MEM
WB

流水线的相关问题

- **指令相关的概念**
  - **流水线变深了，相关问题更为突出。**
  - **在流水线中，如果某指令的某个阶段必须等到它前面另一条指令的某个阶段后才能开始，则这两条指令存在相关**
  - **相关的指令要隔开足够远，否则后面的指令就必须等待**
  - **在我们的流水线中，可能发生如下相关**
    - **红线表示数据相关**
    - **蓝线表示转移相关（从中可看出转移指令在译码阶段执行的必要性）**
    - **还有结构相关（在我们的例子中不存在）**



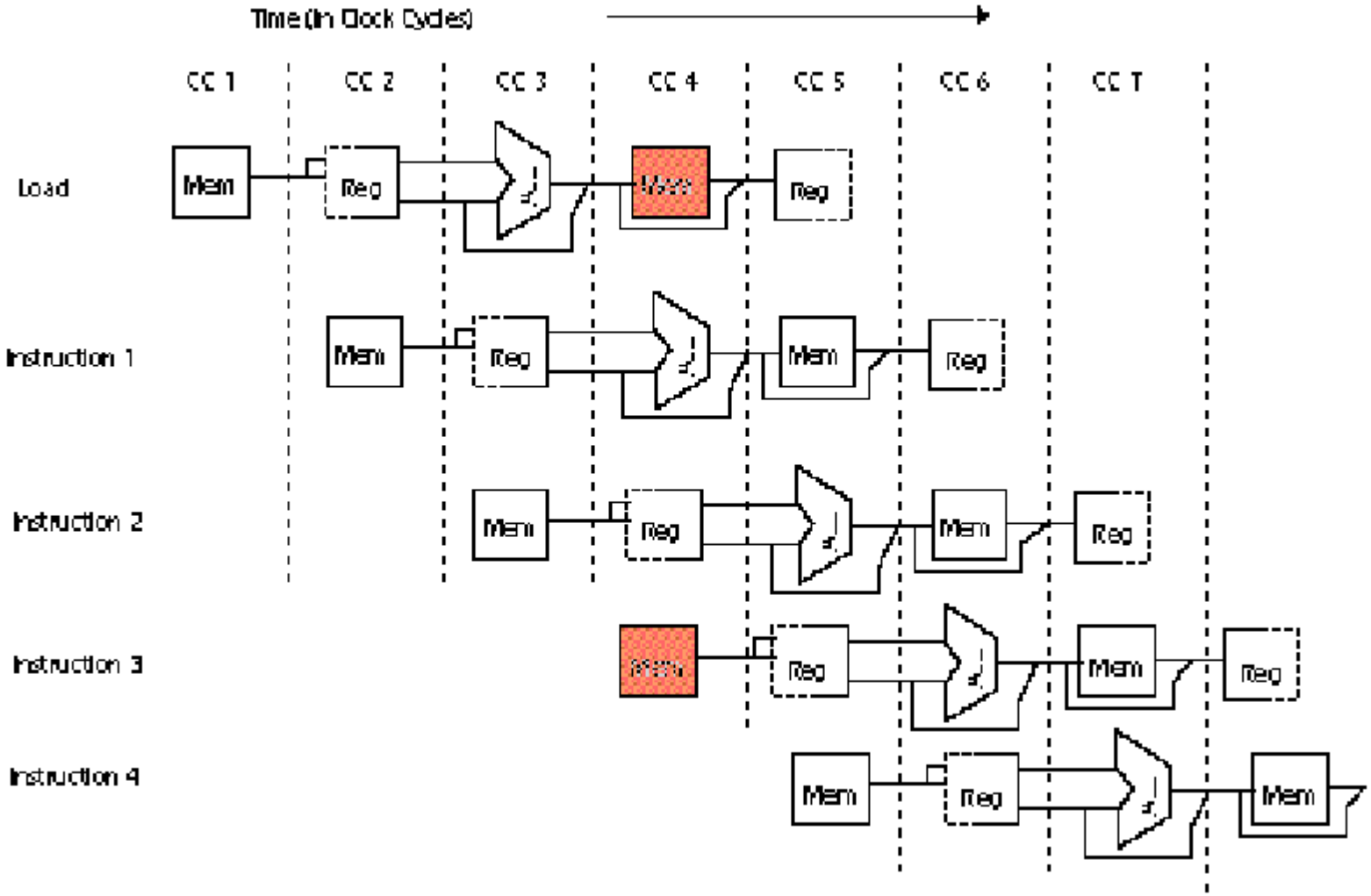

源和目标有
相等且非0
S1 dest S3
数据存储器
S1 dest S3
ALU
Mux2
S1=A C
S3=^A(B+C+D)+A^B^C+^BC^D
S4=A (B+C+D)
S5=A B^C^D
S1
S2
S3
S4
S5 A B C D
S1 dest S3 S2
Zero?
S4
Mux2
Mux2
Mux2
0
0
Mux8
Mux8
OP
Dest
Src1
Src2
Imm
输入
使能
指令存储器
译
码
器
PC
输入
使能
+
Mux2
Reset
2
C

指令流水线的相关

- **数据相关：使用同一个寄存器引起的相关**
  - **如后面的指令用到前面指令的结果**
- **控制相关：与PC有关的相关**
  - **每条指令取指用到PC，转移指令修改PC**
- **结构相关：资源冲突**
  - **多条指令同时使用一个功能部件**
- **相关引起流水线阻塞**

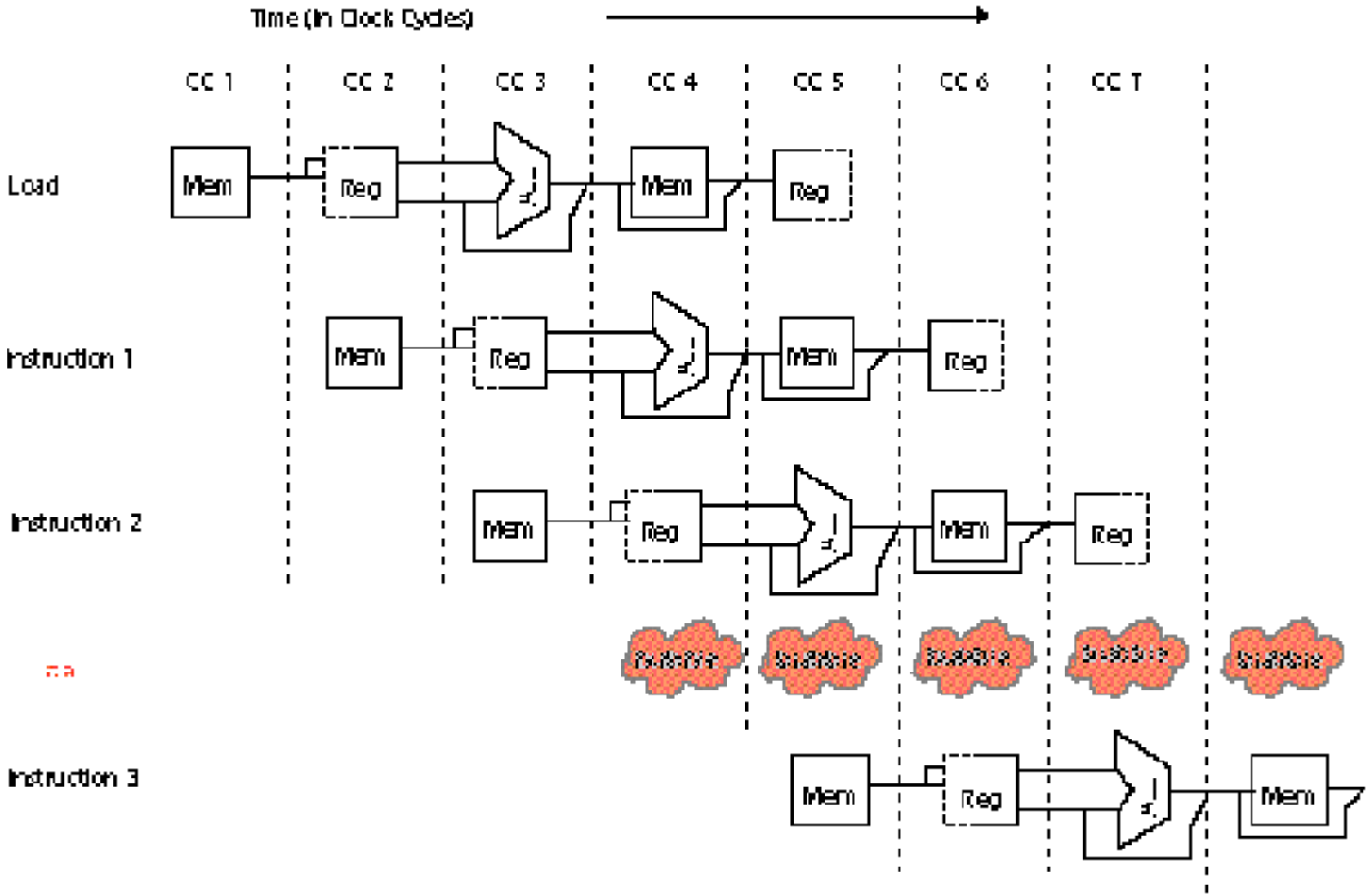
由访存引起的结构相关

- **取指和取数都要访存**



结构相关引起阻塞

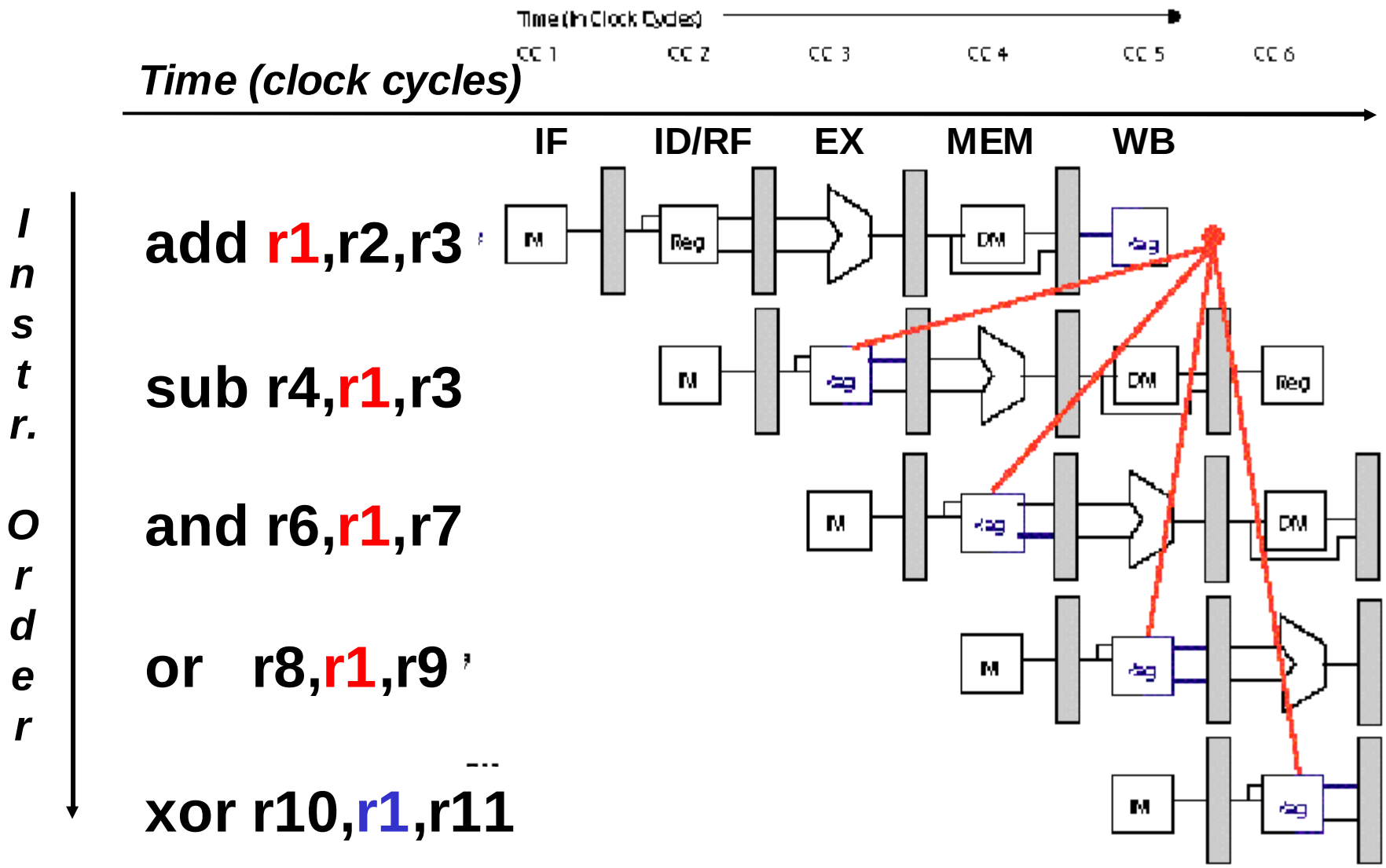
- **取指延迟一拍进行**



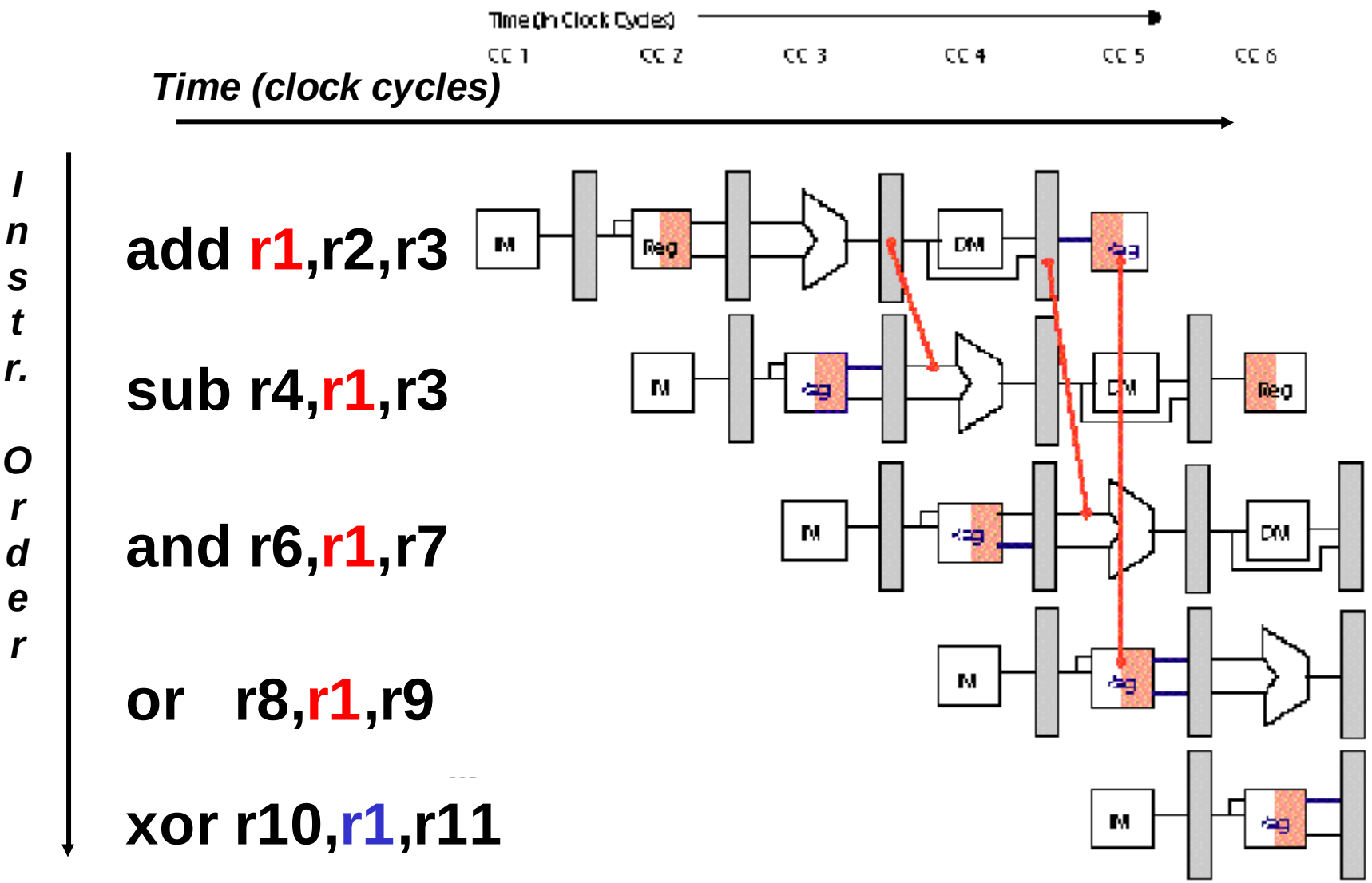
数据相关

- **RAW(Read After Write)**
  - **后面指令用到前面指令所写的数据**
- **WAW(Write After Write)**
  - **两条指令写同一个单元**
  - **在简单流水线中没有此类相关，因为不会乱序执行**
- **WAR(Write After Read)**
  - **后面指令覆盖前面指令所读的单元**
  - **在简单流水线中没有此类相关**
- **在动态流水线中会有WAR和WAW相关**

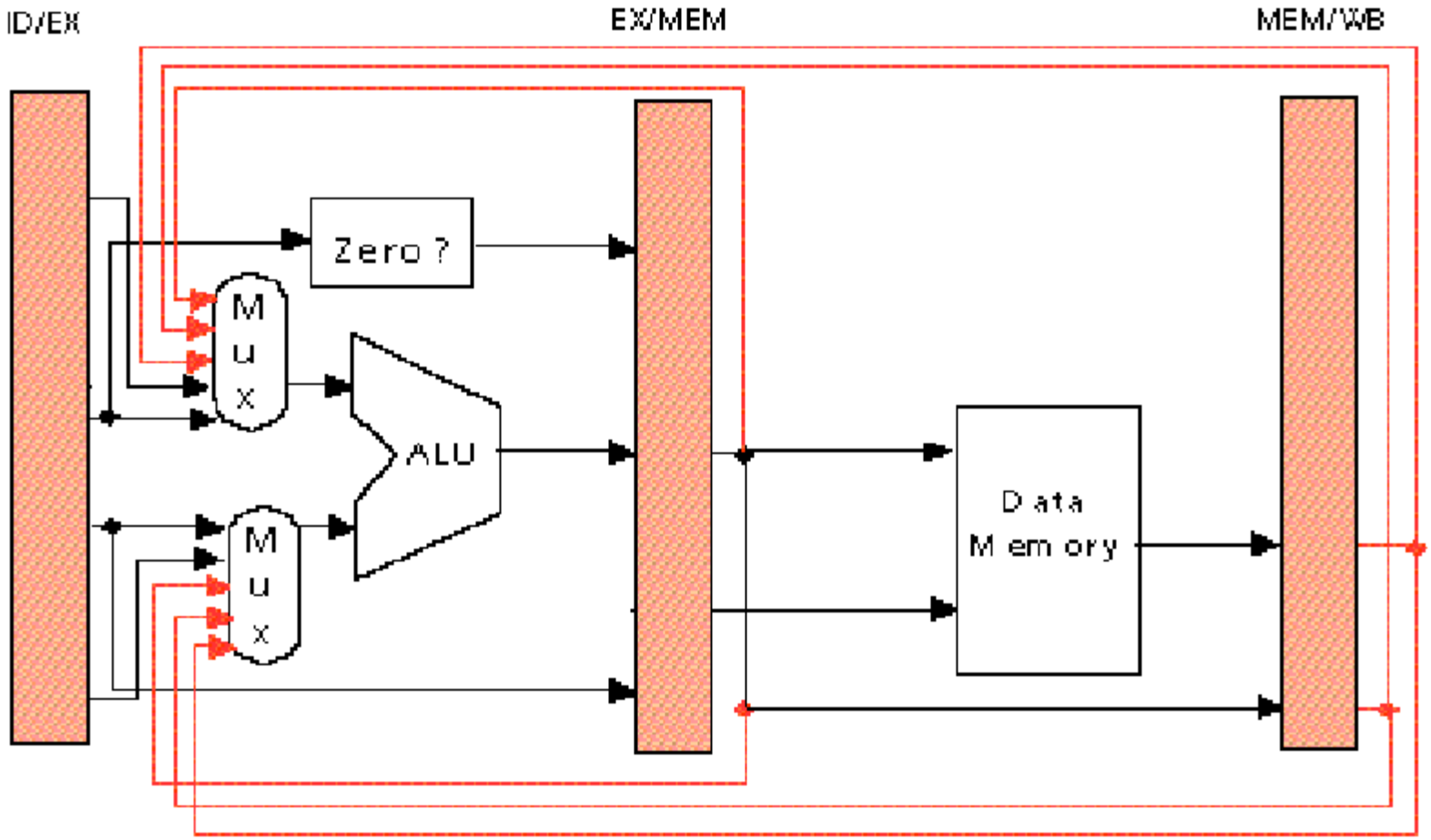
数据相关

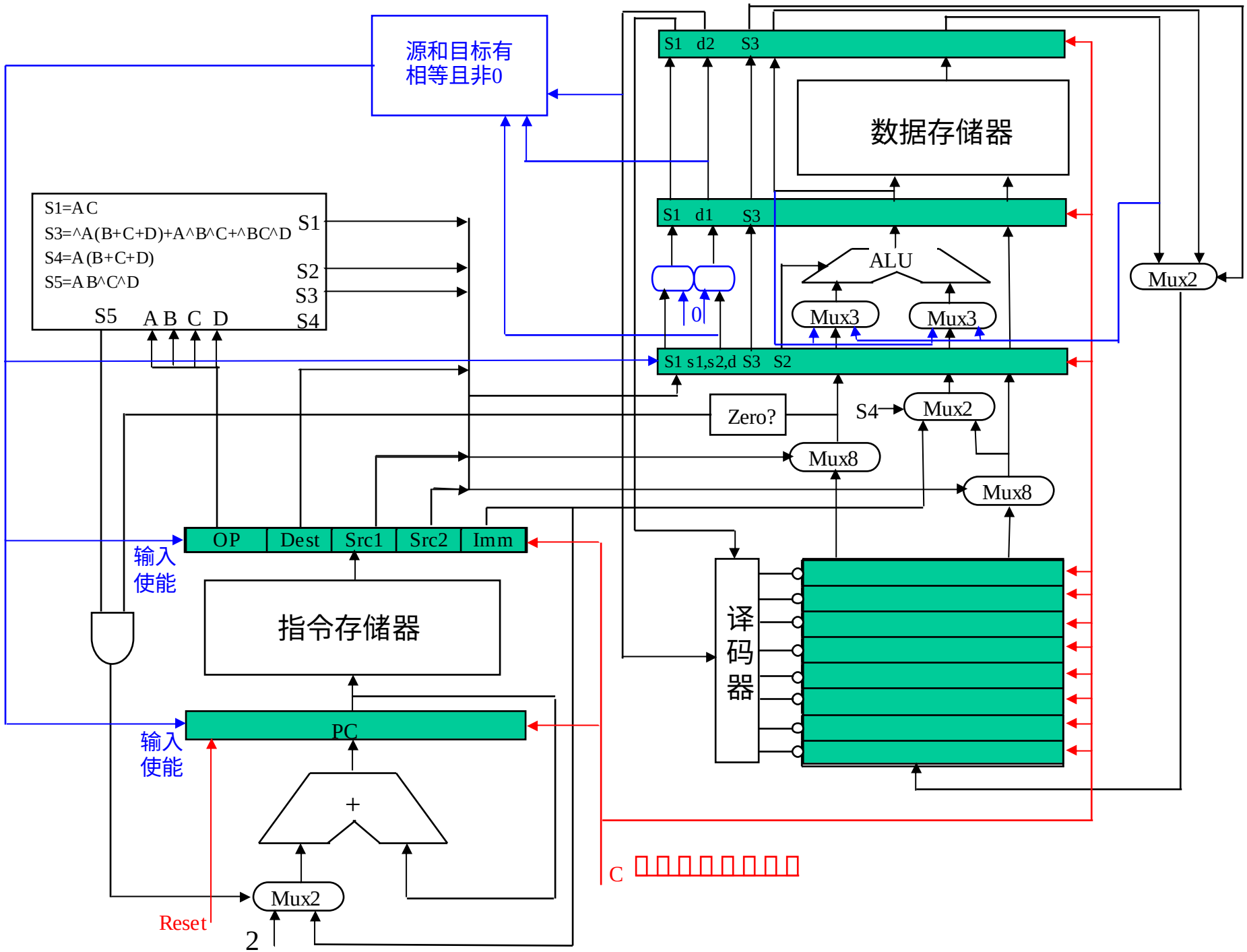

Time (In Clock Cycles)
CC 1
CC 2
CC 3
CC 4
CC 5
CC 6
Time (clock cycles)
IF
ID/RF
EX
MEM
WB
Instr. Order
add r1,r2,r3
sub r4,r1,r3
and r6,r1,r7
or r8,r1,r9
xor r10,r1,r11
IM
Reg
DM

解决RAW相关的Forwarding技术



Forwarding的数据通路

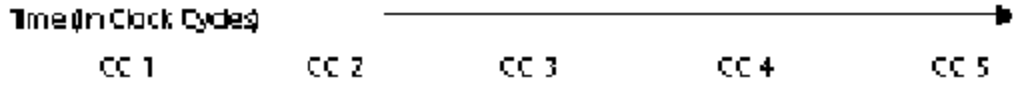



源和目标有
相等且非0
S1 d2 S3
数据存储器
S1 d1 S3
ALU
Mux2
Mux3
Mux3
0
S1=A C
S3=^A(B+C+D)+A^B^C+^BC^D
S4=A (B+C+D)
S5=A B^C^D
S1
S2
S3
S4
S5 A B C D
S1 s1,s2,d S3 S2
Zero?
S4
Mux2
Mux8
Mux8
OP Dest Src1 Src2 Imm
输入
使能
指令存储器
译
码
器
PC
输入
使能
+
Mux2
Reset
2
C

Forwarding的相关处理逻辑

- **以ALU左端的输入为例**
  - **当s1和前面两级的目标寄存器域d1和d2都不相等时，选择中间通路**
  - **当s1==d2时，选择右边通路**
  - **当s1==d1且当前操作不是LD时，选择左边通路**
  - **当s1==d1且当前操作是LD时，后面流水线暂停，往前面流水线送空操作**

Forwarding 情况下的数据相关



Time (clock cycles)

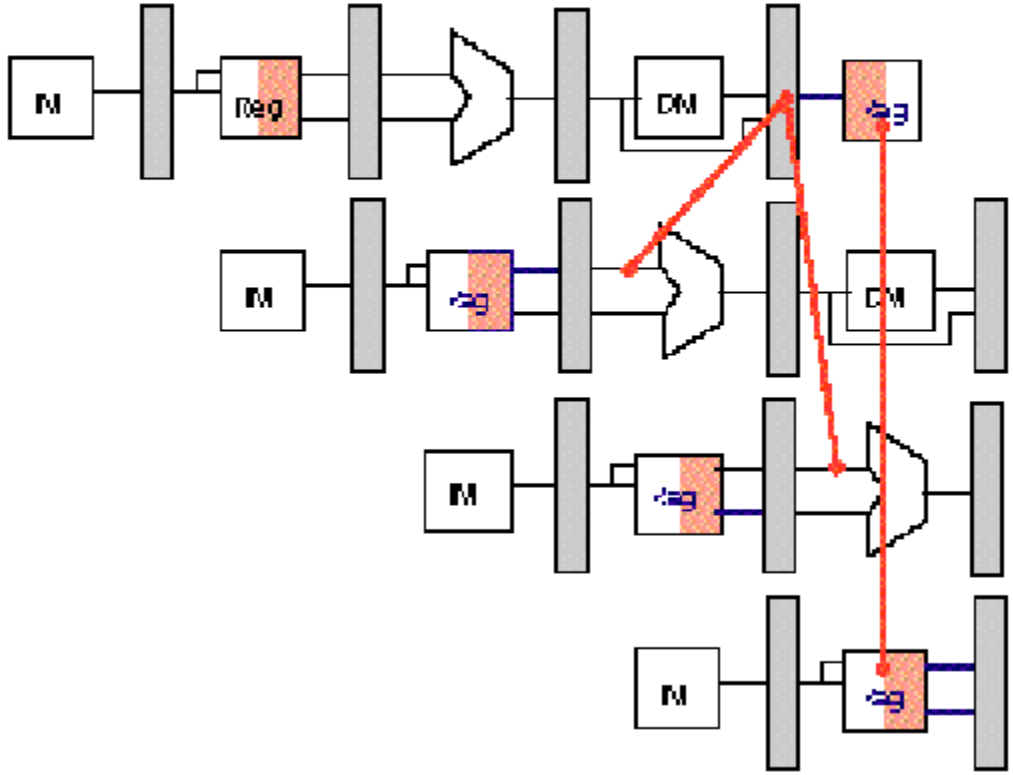
Instr. Order

lw r1, 0(r2)

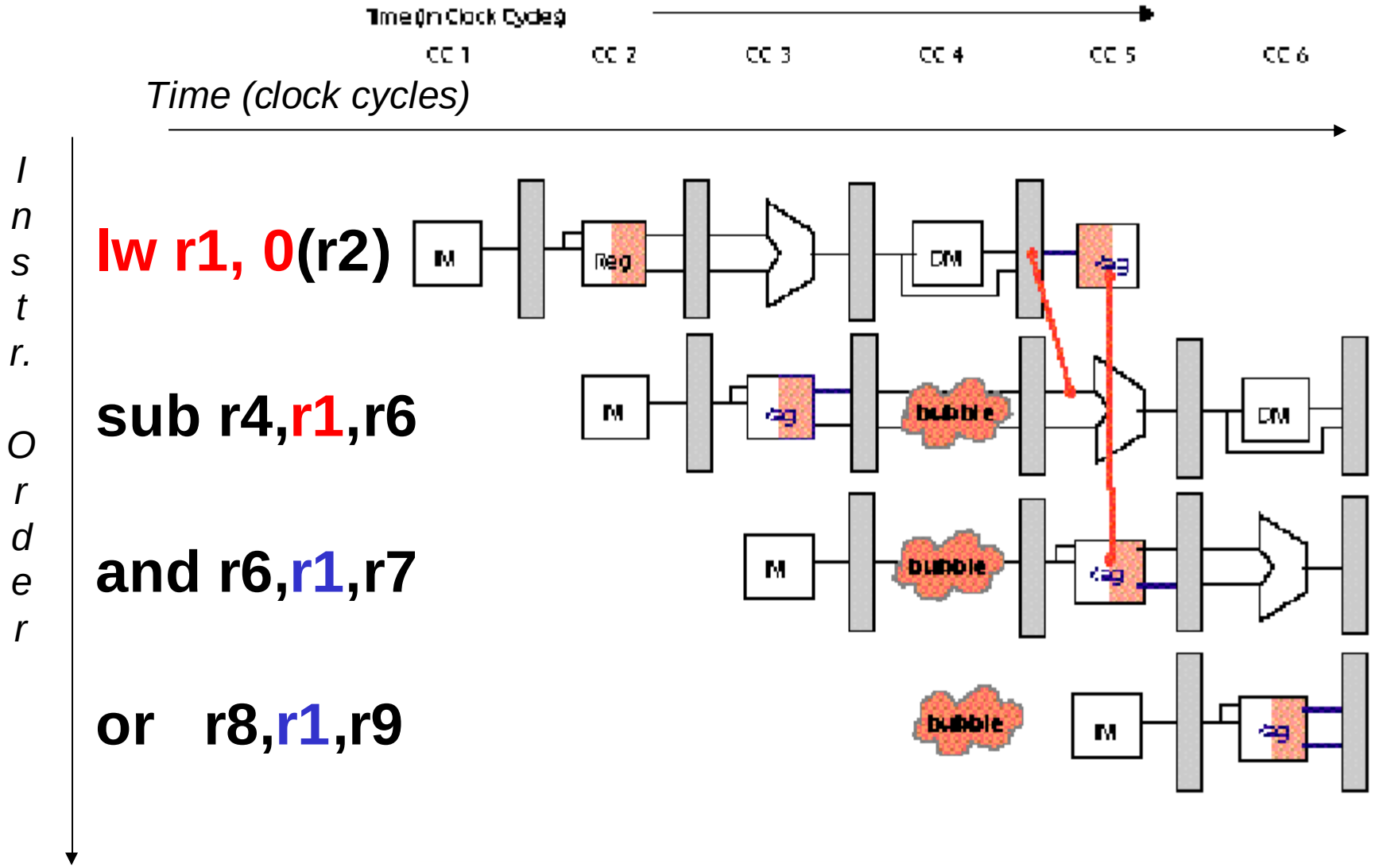
sub r4,r1,r6

and r6,r1,r7

or r8,r1,r9



相关引起的阻塞


Time (in Clock Cycles)
CC 1
CC 2
CC 3
CC 4
CC 5
CC 6
Time (clock cycles)
Instr. Order
lw r1, 0(r2)
sub r4,r1,r6
and r6,r1,r7
or r8,r1,r9
IM
Reg
DM
bubble

通过静态调度解决相关

如下程序段的优化和非优化代码

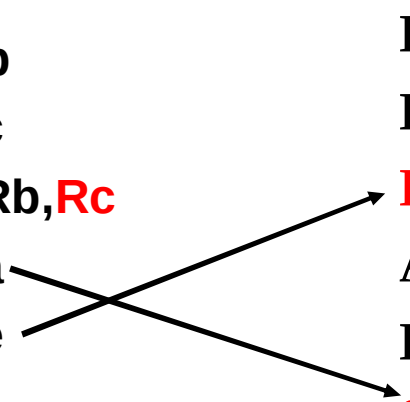
```
a = b + c;
d = e -f;
```

Slow code:

```
LW    Rb,b
LW    Rc,c
ADD   Ra,Rb,Rc
SW    a,Ra
LW    Re,e
LW    Rf,f
SUB   Rd,Re,Rf
SW    d,Rd
```

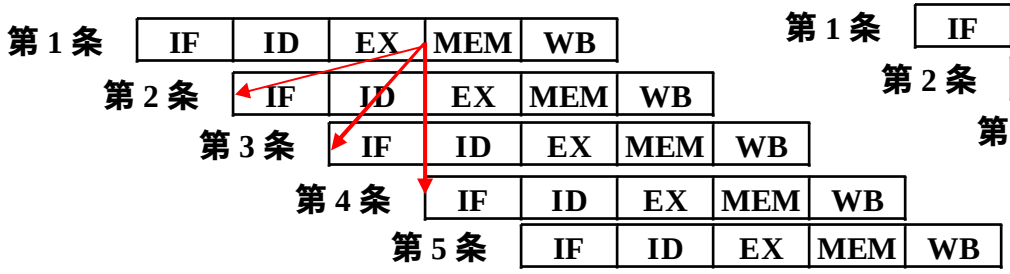
Fast code:

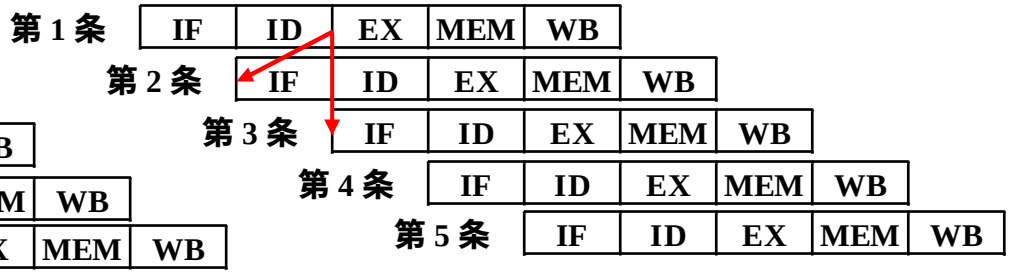
```
LW    Rb,b
LW    Rc,c
LW    Re,e
ADD   Ra,Rb,Rc
LW    Rf,f
SW    a,Ra
SUB   Rd,Re,Rf
SW    d,Rd
```



控制相关

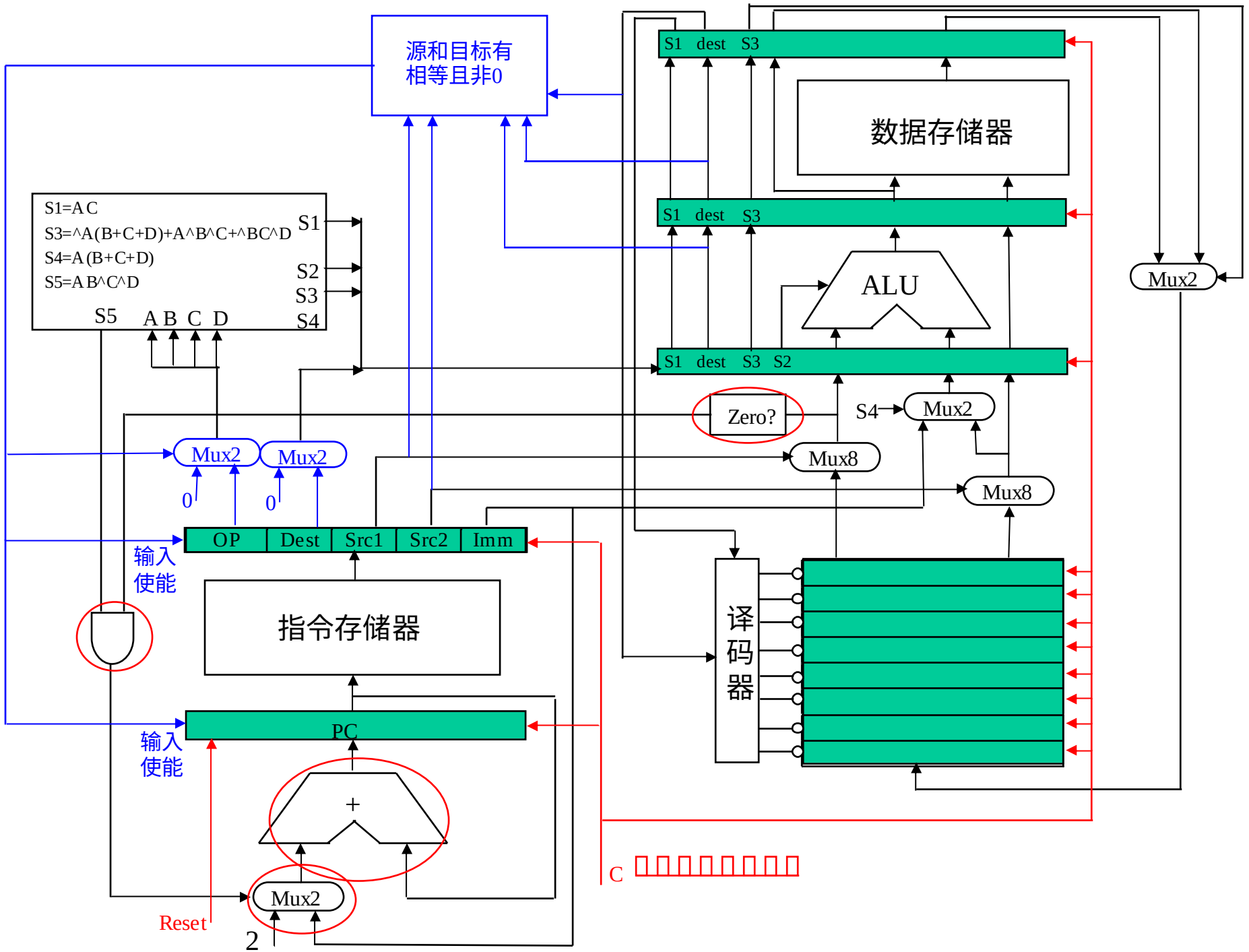
- **PC相关**
  - **转移指令计算的下一条指令地址在EX阶段计算**
  - **下一条指令等2拍**
  - **使用专门的地址运算部件把地址计算提前到译码阶段可以少等一拍**
  - **使用一个delay slot可以不用等待**
- **转移条件相关**
  - **如果在ID阶段执行转移指令时转移条件未确定还是要等待**





Delayed Branch

- **Where to get instructions to fill branch delay slot?**
  - **Before branch instruction**
  - **From the target address: only valuable when branch taken**
  - **From fall through: only valuable when branch not taken**
  - **Cancelling branches allow more slots to be filled**
- **Compiler effectiveness for single branch delay slot:**
  - **Fills about 60% of branch delay slots**
  - **About 80% of instructions executed in branch delay slots useful in computation**
  - **About 50% (60% x 80%) of slots usefully filled**
- **Delayed Branch downside: 7-8 stage pipelines, multiple instructions issued per clock (superscalar)**


源和目标有
相等且非0
S1 dest S3
数据存储器
S1 dest S3
ALU
Mux2
S1=A C
S3=^A(B+C+D)+A^B^C+^BC^D
S4=A (B+C+D)
S5=A B^C^D
S1
S2
S3
S4
S5 A B C D
S1 dest S3 S2
Zero?
S4
Mux2
Mux2
Mux2
0
0
Mux8
Mux8
OP Dest Src1 Src2 Imm
输入
使能
指令存储器
译
码
器
PC
输入
使能
+
Mux2
Reset
2
C

解决转移条件相关的方法

- **阻塞**
  - **等待直到转移条件确定**
- **用delay slot容忍延迟**
- **编译器优化**
  - **产生条件的指令和转移指令隔得足够远**
- **预测**
  - **预测转移成功：较精确，计算转移地址需要delay slot**
  - **预测转移不成功：直接用PC+4**
  - **复杂预测技术：精确、控制复杂**
  - **都需要预测错误时的恢复**

例外(Exception)与流水线

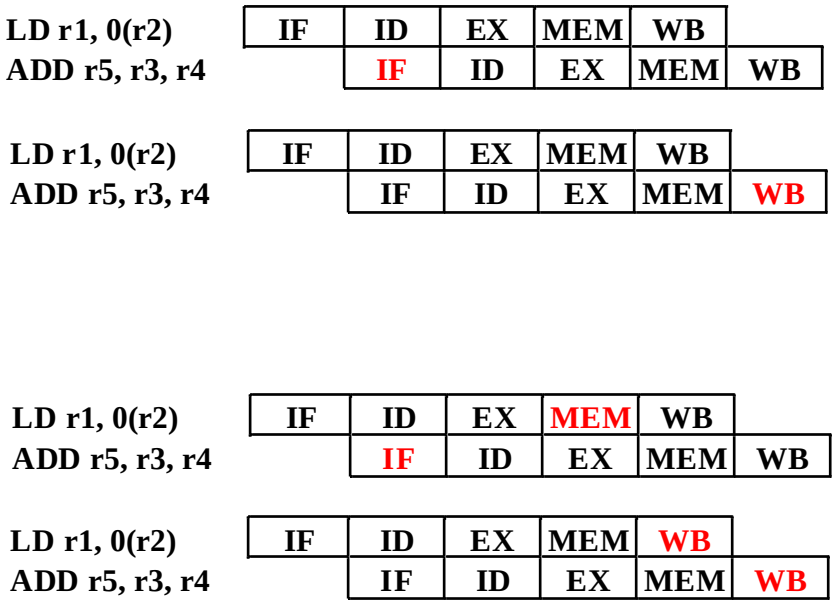
- **例外原因**
  - **I/O请求：外部中断**
  - **指令例外：用户请求中断**
    - **系统调用、断点、跟踪调试指令**
  - **运算部件**
    - **整数运算溢出、浮点异常**
  - **存储管理部件**
    - **访存地址不对齐、用户访问系统空间、TLB失效、缺页、存储保护错（写只读页）**
  - **保留指令错：未实现指令**
  - **硬件错**
  - **等等**

- 例外发生的流水阶段
  - 取指：访存例外
  - 译码：保留指令、中断指令如Trap、Syscall
  - 执行：整数溢出、浮点异常（如除零）等
  - 访存：访存例外
  - 其它：外部中断，可能在任何时候发生
- 例外特征
  - 同步与异步
  - 用户请求与系统强制
  - 可屏蔽与不可屏蔽
  - 指令内与指令间
  - 可恢复与结束

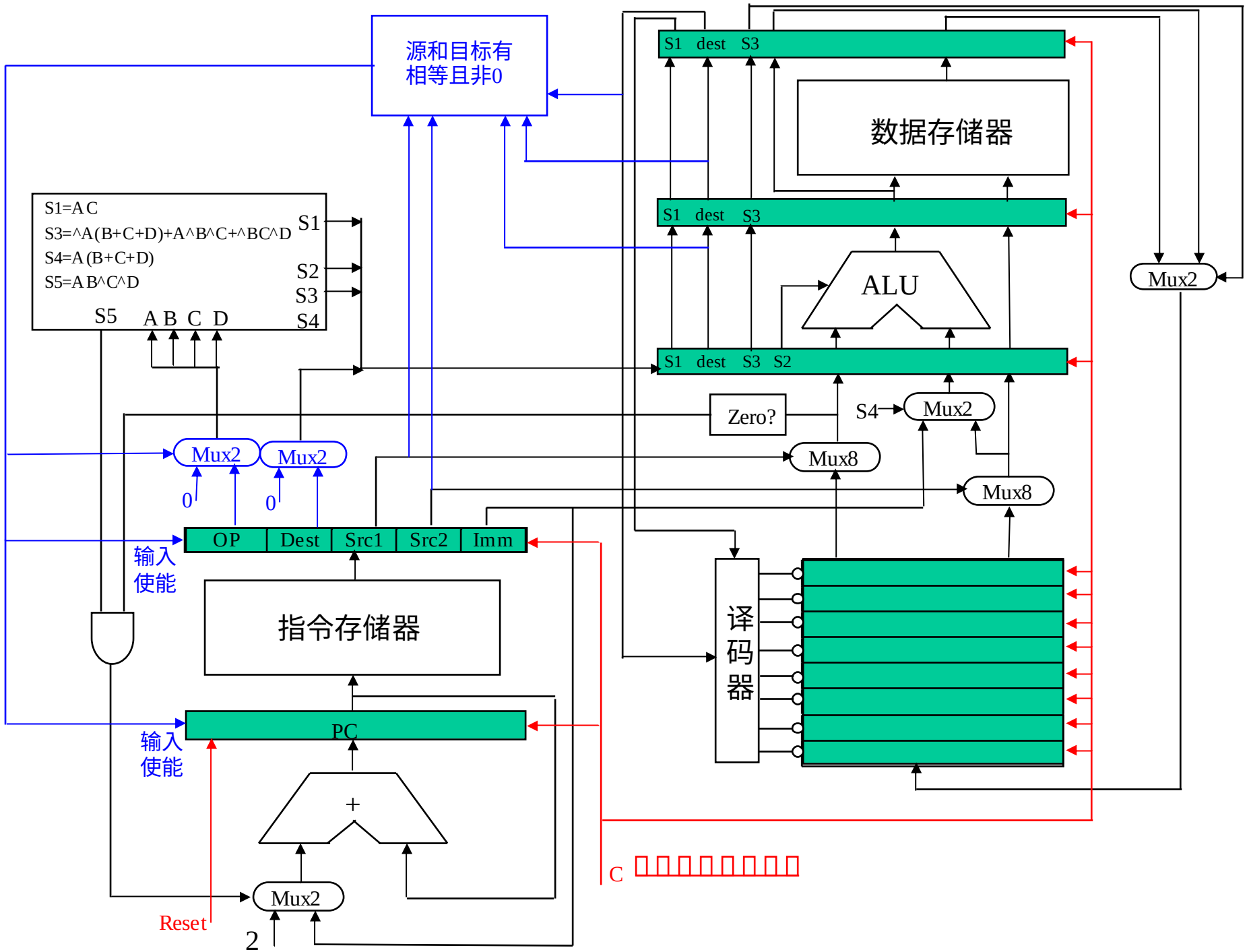
	同步/异步	用户/强制	可/不可屏蔽	指令内/间	可恢复/结束
I/O 请求	异步	强制	不可屏蔽	指令间	可恢复
系统调用	同步	用户	不可屏蔽	指令间	可恢复
跟踪调试	同步	用户	可屏蔽	指令间	可恢复
断点	同步	用户	可屏蔽	指令间	可恢复
整数溢出	同步	强制	可屏蔽	指令内	可恢复
浮点异常	同步	强制	可屏蔽	指令内	可恢复
缺页	同步	强制	不可屏蔽	指令内	可恢复
访存地址不齐	同步	强制	可屏蔽	指令内	可恢复
存储保护错	同步	强制	不可屏蔽	指令内	可恢复
非法指令	同步	强制	不可屏蔽	指令内	结束
硬件错	异步	强制	不可屏蔽	指令内	结束
电源错	异步	强制	不可屏蔽	指令内	结束

- 在前述例外中，指令内可恢复例外的处理比较困难，条件转移指令的delay slot又增加了中断处理的难度
- 精确中断：在处理中断时，发生中断指令前面的所有指令都执行完，中断指令后面的所有指令还未执行。精确中断是存储管理和IEEE运算规范的要求
  - 发生例外指令前面的指令继续执行完
  - 后面的指令不能修改机器状态，对运算状态字的修改可能在EX阶段进行
  - 多条指令发生例外

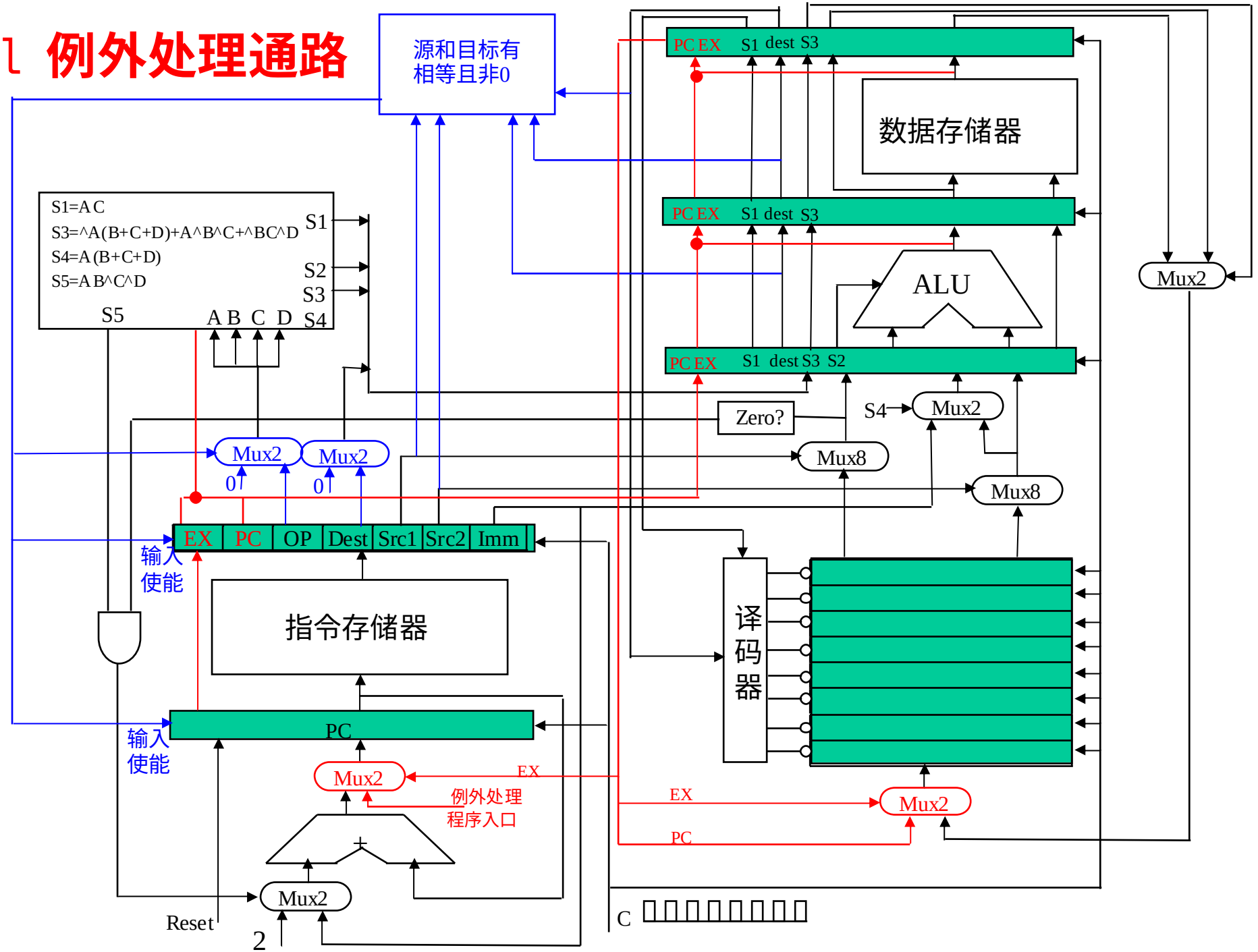
- 可以把每条指令的例外延迟到WB时再处理
  - 对机器状态的修改也在WB阶段进行
  - 对状态寄存器的修改从EX阶段延迟到WB阶段



- 简单流水线的例外处理
  - 任何以及流水发生例外时，在流水线中记录下发生例外的事件，直到WB阶段再处理
  - 如果在EX阶段要修改机器状态（如状态寄存器），保存下来直到WB阶段再修改。
  - 指令的PC值随指令流水前进到WB阶段中断处理专用
  - 外部中断作为IF的例外处理
  - 指定通用寄存器中的一个（如最后一个）为中断处理时保存PC值专用。
  - 当发生例外的指令在WB阶段时：
    - 保存该指令的PC（也在WB阶段），有些机器还保存其它状态
    - 置PC值为中断处理程序入口地址


源和目标有
相等且非0
S1 dest S3
数据存储器
S1 dest S3
ALU
Mux2
S1=A C
S3=^A(B+C+D)+A^B^C+^BC^D
S4=A (B+C+D)
S5=A B^C^D
S1
S2
S3
S4
S5 A B C D
S1 dest S3 S2
Zero?
S4
Mux2
Mux2
Mux2
0
0
Mux8
Mux8
OP
Dest
Src1
Src2
Imm
输入
使能
指令存储器
译
码
器
PC
输入
使能
+
Mux2
Reset
2
C

l 例外处理通路


源和目标有
相等且非0
PC EX S1 dest S3
数据存储器
PC EX S1 dest S3
ALU
PC EX S1 dest S3 S2
Mux2
S1=A C
S3=^A(B+C+D)+A^B^C+^BC^D
S4=A (B+C+D)
S5=A B^C^D
S1
S2
S3
S4
S5
A B C D
Zero?
S4
Mux2
Mux2
Mux2
0
0
Mux8
Mux8
EX PC OP Dest Src1 Src2 Imm
输入
使能
指令存储器
译
码
器
PC
输入
使能
Mux2
EX
例外处理
程序入口
EX
Mux2
PC
Mux2
Reset
2
C