控制冲突和异常

2021年秋

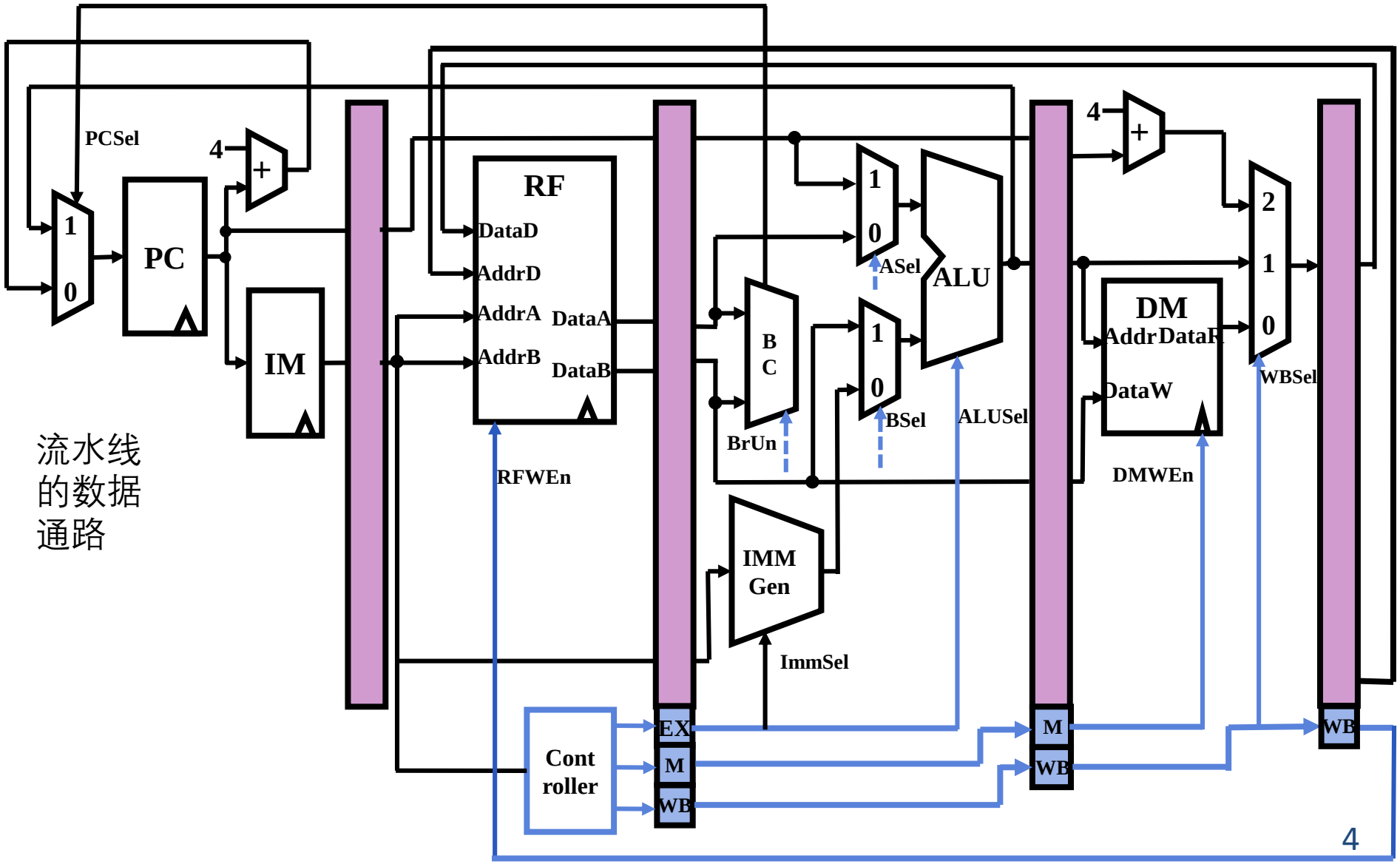
本讲概要

- 指令流水的“冲突”问题
- 控制冲突及对应方案
- 异常及处理

流水线的实现原理

- 指令流水的简单实现
  - 每一条指令的实现至多需要5个时钟周期。这5个时钟周期如下：
    - 取指令周期（**IF**）
    - 指令译码 / 读寄存器周期（**ID**）
    - 执行 / 有效地址计算周期（**EX**）
    - 存储器访问 / 分支完成周期（**MEM**）
    - 写回周期（**WB**）
  - 不同类型的指令在以上5个时钟周期中进行的操作各不相同。

流水线的实现



流水线的数据通路

流水线的冲突

- 结构冲突
  - 是指令在重叠执行的过程中，硬件资源满足不了指令重叠执行的要求而产生的冲突
- 数据冲突
  - 是指在同时重叠执行的几条指令中，一条指令依赖于前面指令执行结果数据，但是又在指定的数据源中得不到时发生的冲突。
- 控制冲突
  - 是指流水线中的分支指令或者其他需要改写PC的指令造成的冲突

结构冲突和数据冲突

- 结构冲突
  - 在同一时刻，由于指令重叠执行，不同指令争用同一硬件资源。
    - 暂停流水线
    - 重复设置资源（把资源克隆成多份）
- 数据冲突
  - 由于指令重叠执行，造成指令间执行步骤的相对时间发生变化，后续指令无法在预期的位置得到正确的源操作数
    - 暂停流水线
    - 数据旁路：检测、传送（给出最新鲜的数据）
    - 优化编译器
    - 动态调度

数据冲突和相应解决方法

- 数据冲突的动态调度
  - 这种方法是由硬件动态调整指令执行顺序以减少暂停的影响，能够简化编译器设计。
  - 动态调度并不能真正消除数据冲突，但它能在出现数据冲突时尽量避免出现处理器暂停。而静态调度方法则是尽量通过分离有冲突问题的指令使它们不会导致冲突，从而减少暂停的影响。
  - 动态调度的主要思想：
    - 指令顺序发射——乱序执行——指令乱序流出
  - 动态调度的问题：异常处理的不精确性。在采用动态调度方法的处理机中，在某条指令产生异常情况时，有可能出现其后面的指令已经执行完成的情况，这样异常处理是不精确的。

控制冲突

- 流水线的控制冲突是因为程序执行转移类指令而引起的冲突。转移类指令如无条件转移、条件转移、子程序调用、中断等，它们属于分支指令，执行中可能改变程序的方向，从而造成流水线断流。
- 数据冲突影响到的仅仅是本条指令附近少数几条指令，所以称为局部冲突。而控制冲突影响的范围要大得多，它会引起程序执行方向的改变，使流水线损失更多的性能，所以称为全局冲突。
- 控制冲突会使流水线的连续流动受到破坏。当执行条件转移指令时，有两种可能结果：
  - 如发生转移，将程序计数器PC的内容改变成转移目标地址；
  - 如不发生转移，只将PC加上一个增量，指向下一条指令的地址。

控制冲突

- 数据冲突
  - 由于寄存器数据的缺失引发
- 控制冲突
  - 由于PC的缺失引发
  - 条件转移和无条件转移指令
- 控制冲突对性能影响更大
  - IF在指令流水的第一个阶段

控制冲突

	R/I - Type	LW	SW	Br	Jal	Jalr
IF	use	use	use	use	use	use
ID	produce	produce	produce			
EX				produce	produce	produce
MEM						
WB						

- 所有指令都要在IF阶段使用PC
- 对转移指令，至少要到EXE阶段才能得到正确的PC
  - 暂停流水线一个或者两个周期

控制冲突的处理

- 暂停流水线
  - 直到有了正确的转移地址
  - 造成性能的降低
- 预测分支不成功
  - 顺序执行下一条指令
  - 预测失败后要清除错误启动的指令
- 预测分支成功
  - 更复杂一些，因为要计算转移目的地址
  - 预测失败后要清除错误启动的指令
- 动态预测
  - 硬件根据上次分支的结果进行本次预测

暂停流水线

- 控制冲突简单解决方法
  - 一旦发现分支指令就暂停流水线，即暂停该指令之后的所有指令，直到分支指令达到MEM段确定了新的PC值为止。

控制冲突和相应解决方法

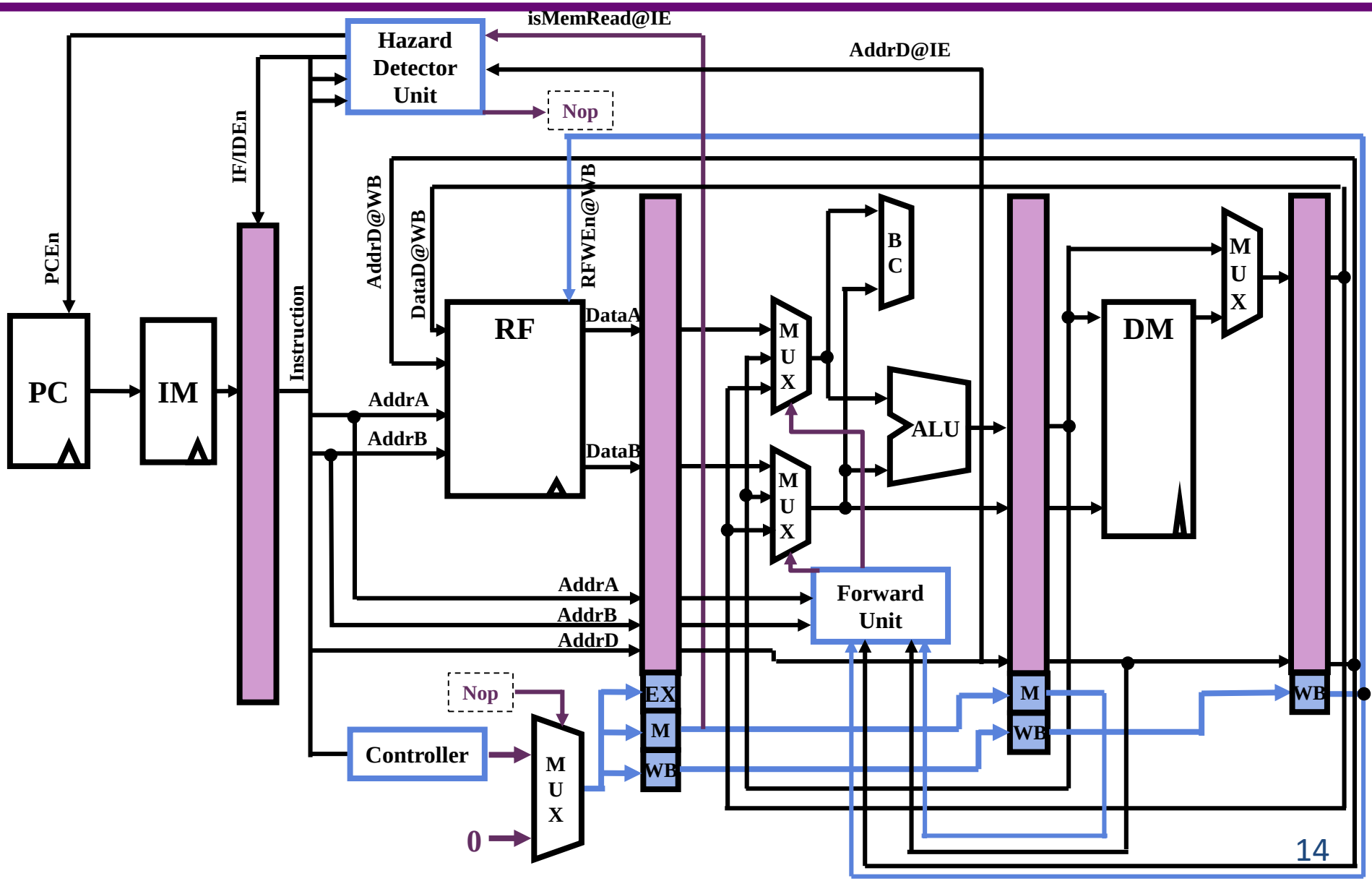
简单方法处理时空图

指令	时钟									
	1	2	3	4	5	6	7	8	9	10
分支指令	IF	ID	EX	MEM	WB					
分支后继指令		IF	Stall	Stall	IF	ID	EX	MEM	WB	
分支后继指令+1						IF	ID	EX	MEM	WB
分支后继指令+2							IF	ID	EX	MEM
分支后继指令+3								IF	ID	EX
分支后继指令+4									IF	ID
分支后继指令+5										IF

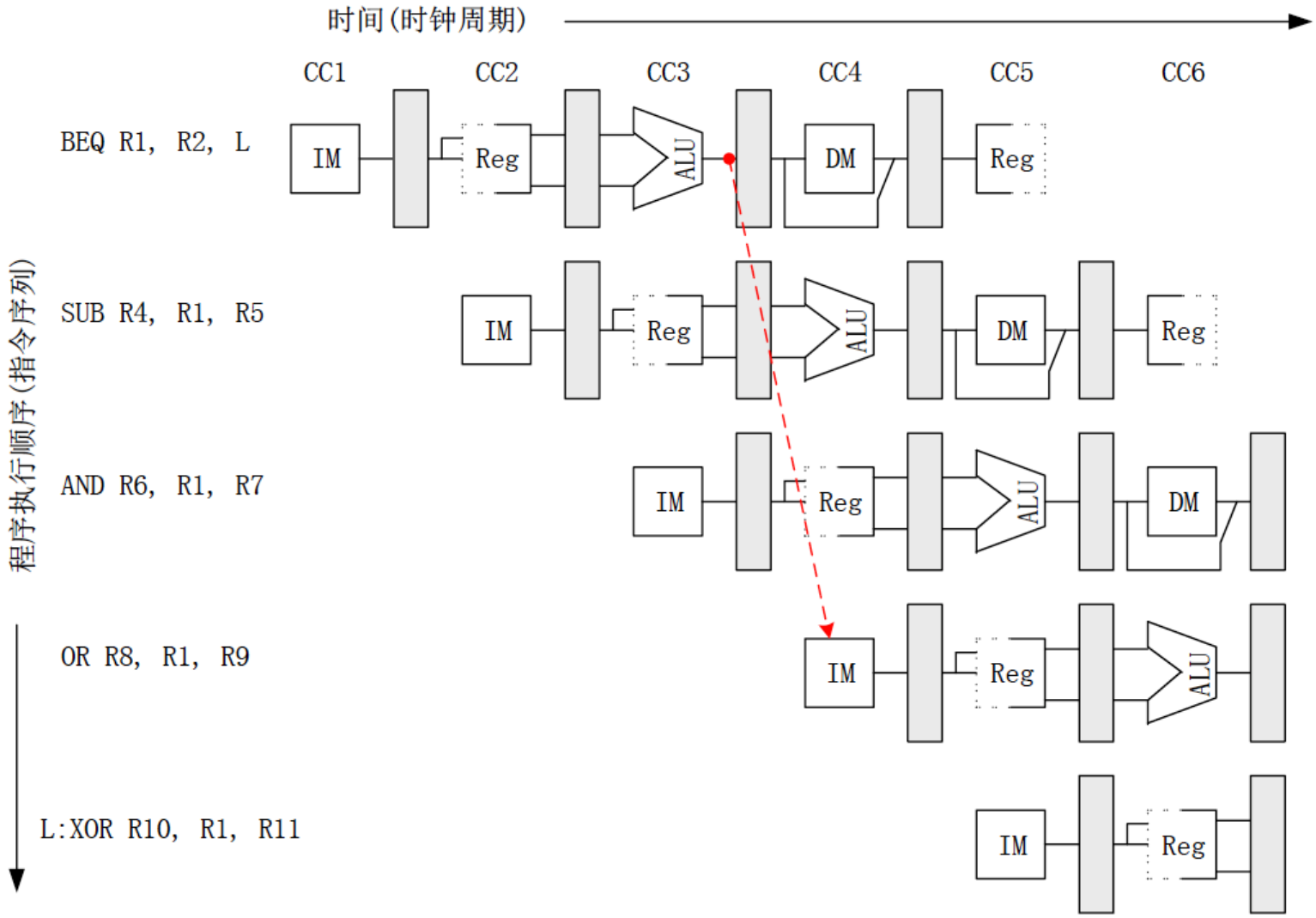
减少流水线处理分支指令时暂停周期数的方案

- 在流水线中尽早判断出分支转移是否成功；
- 尽早计算出分支成功转移时的PC值（如分支的目标地址）。

支持流水的CPU（加上Data Forwarding）

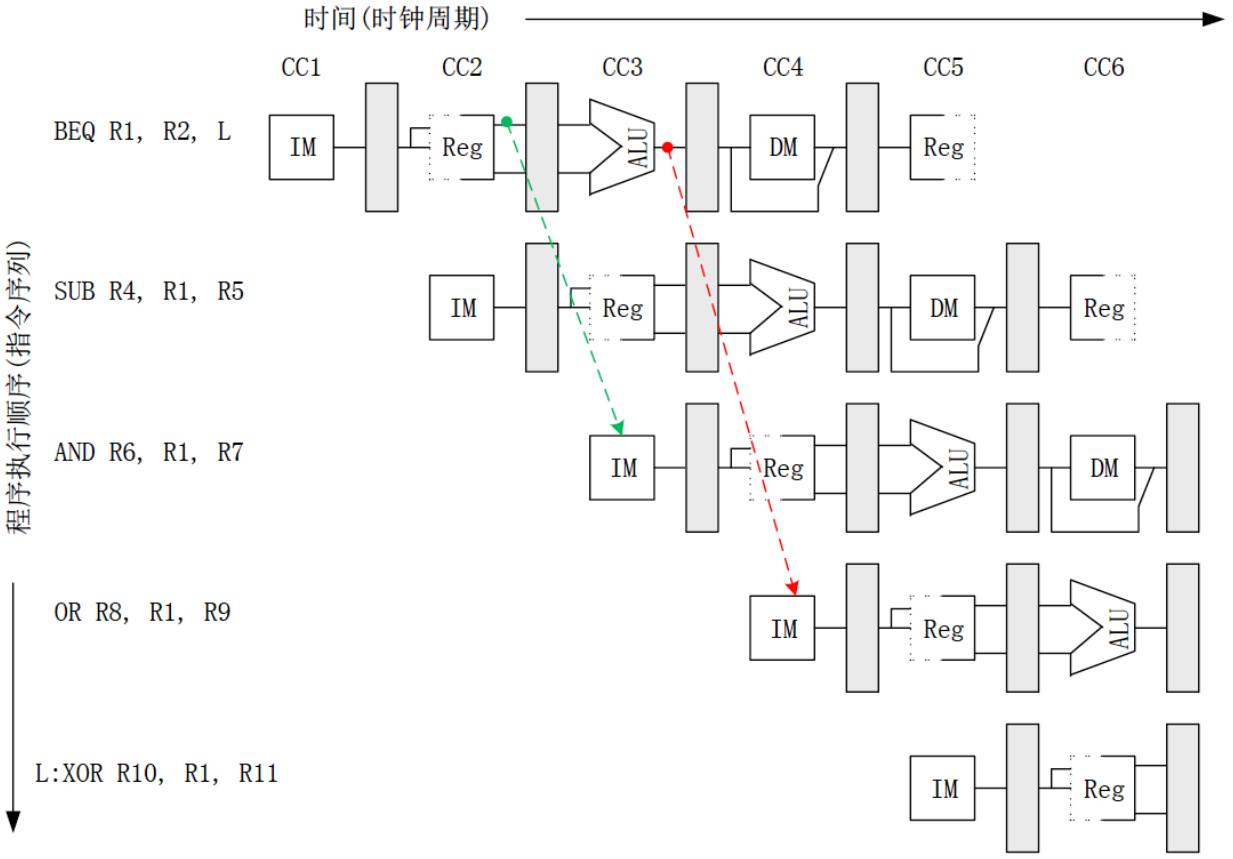


暂停流水线


时间(时钟周期)
CC1
CC2
CC3
CC4
CC5
CC6
BEQ R1, R2, L
SUB R4, R1, R5
AND R6, R1, R7
OR R8, R1, R9
L:XOR R10, R1, R11
程序执行顺序(指令序列)
IM
Reg
ALU
DM
Reg

暂停流水线

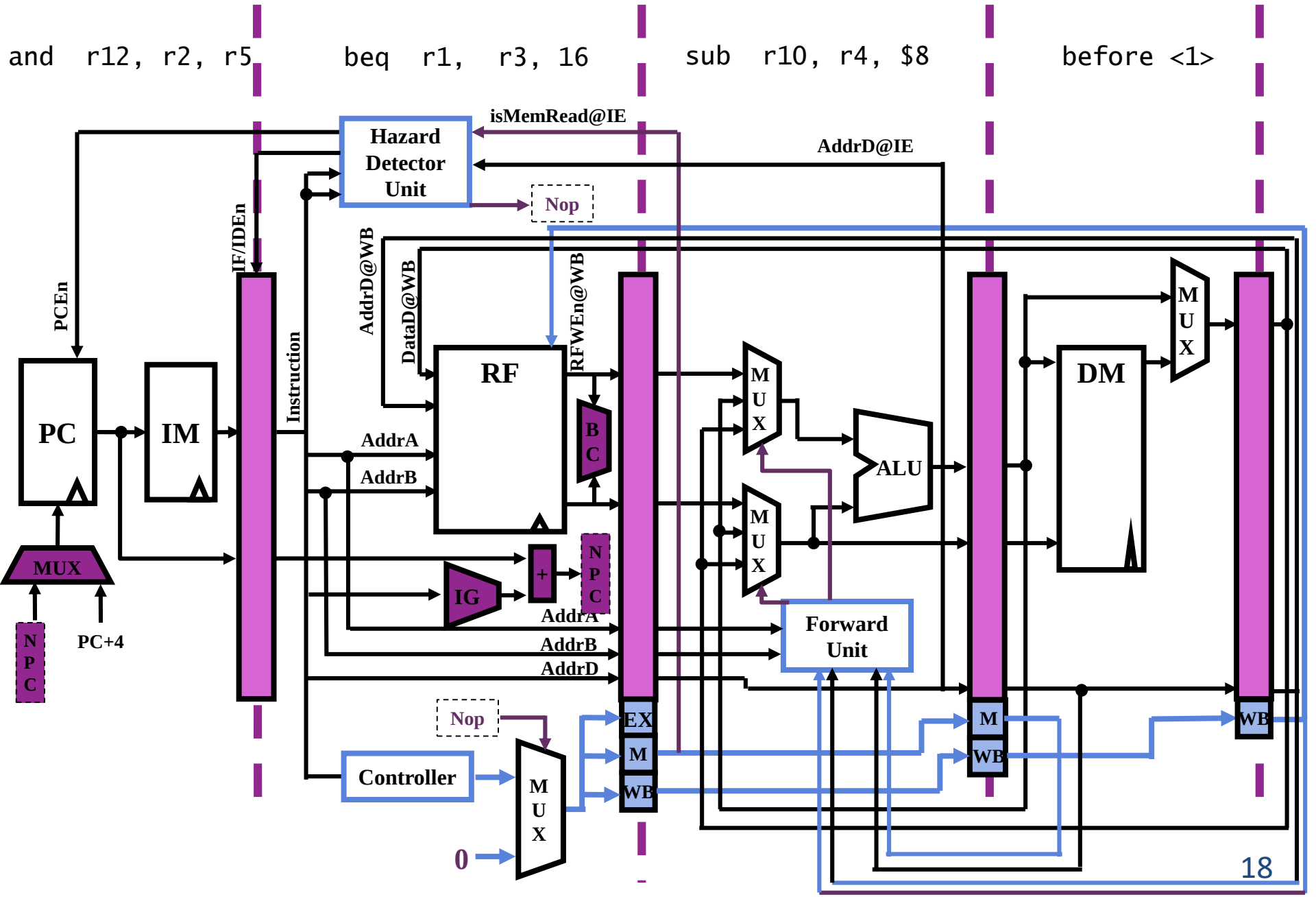
- 减少暂停周期数
  - 在流水线中尽早判断出分支转移是否成功；
  - 尽早计算出分支成功转移时的PC值（如分支目标地址）。



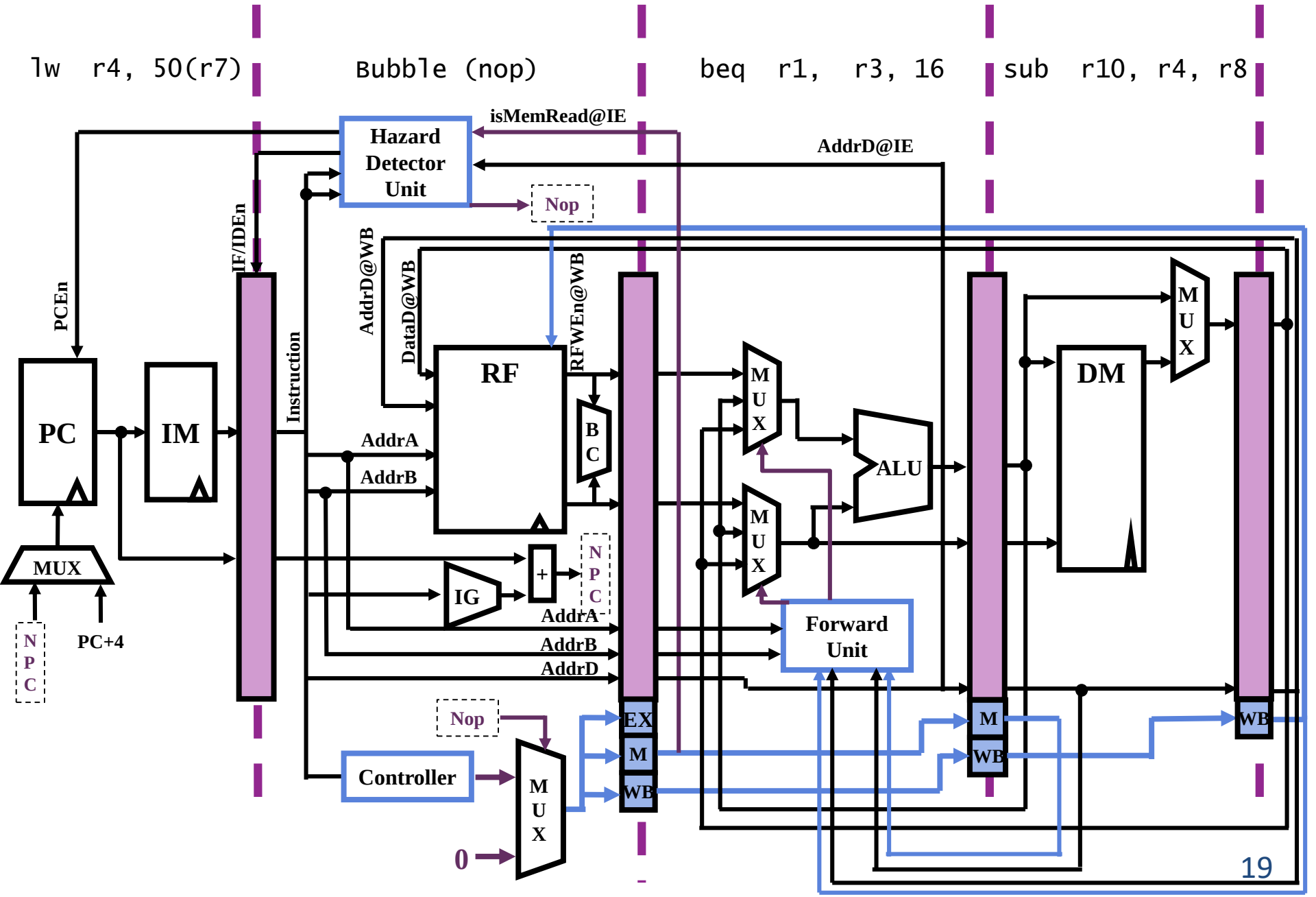
减少分支延迟

- 硬件在ID阶段就确定需要的信息，增加：
  - 目标地址加法器
  - 寄存器比较器
- 例子程序：Branch Taken
  - ```
    36:  sub  r10, r4, r8
    40:  beq  r1,  r3, 32
    44:  and  r12, r2, r5
    48:  or   r13, r2, r6
    52:  add  r14, r4, r2
    56:  slt  r15, r6, r7
         ...
    72:  lw   r4, 50(r7)
    ```
    ```

Branch Taken



Branch Taken



增加的部件

- 数据通路
  - 增加比较器，完成源操作数的比较
  - 调整加法器到ID段，完成转移地址的计算
- 控制信号
  - 根据比较结果，若相等则转移
    - PCsrc选择转移目标地址(PC ←目的地址)
    - 清除IF/ID段寄存器的值(IF/ID寄存器← NOP)
    - 当前指令的控制信号为0(ID/EX寄存器← 0)
  - 若不等，则顺序执行
    - PCsrc选择顺序执行(PC ← PC+4)
    - 保留IF/ID段寄存器的值(IF/ID寄存器← BEQ的后一条指令)
    - 当前指令的控制信号为0(ID/EX寄存器← 0)
- 数据旁路??

分支预测

- 减少流水线分支开销的方法
  - 预测分支转移失败的方法
    - 译码到一条分支指令指令时，就像分支指令就是一条普通的指令那样，流水线继续取后续的指令，并且允许分支指令后续指令在流水线中正常流动。
    - 发现预测错误后，要清除已执行指令的影响
  - 预测分支转移成功的方法
    - 一旦完成分支指令的译码并且计算出了分支的目标地址，就假设分支转移成功，并且开始在分支目标地址处取指令执行。
    - 发现预测错误后，要清除已执行指令的影响
  - 分支延迟（延迟转移）的方法

控制冲突和相应解决方法

预测分支转移的方法

指令	时钟									
	1	2	3	4	5	6	7	8	9	10
分支指令i	IF	ID	EX	MEM	WB					
指令i+1		IF	ID	EX	MEM	WB				
指令i+2			IF	ID	EX	MEM	WB			
指令i+3				IF	ID	EX	MEM	WB		
指令i+4					IF	ID	EX	MEM	WB	

(a)分支转移失败

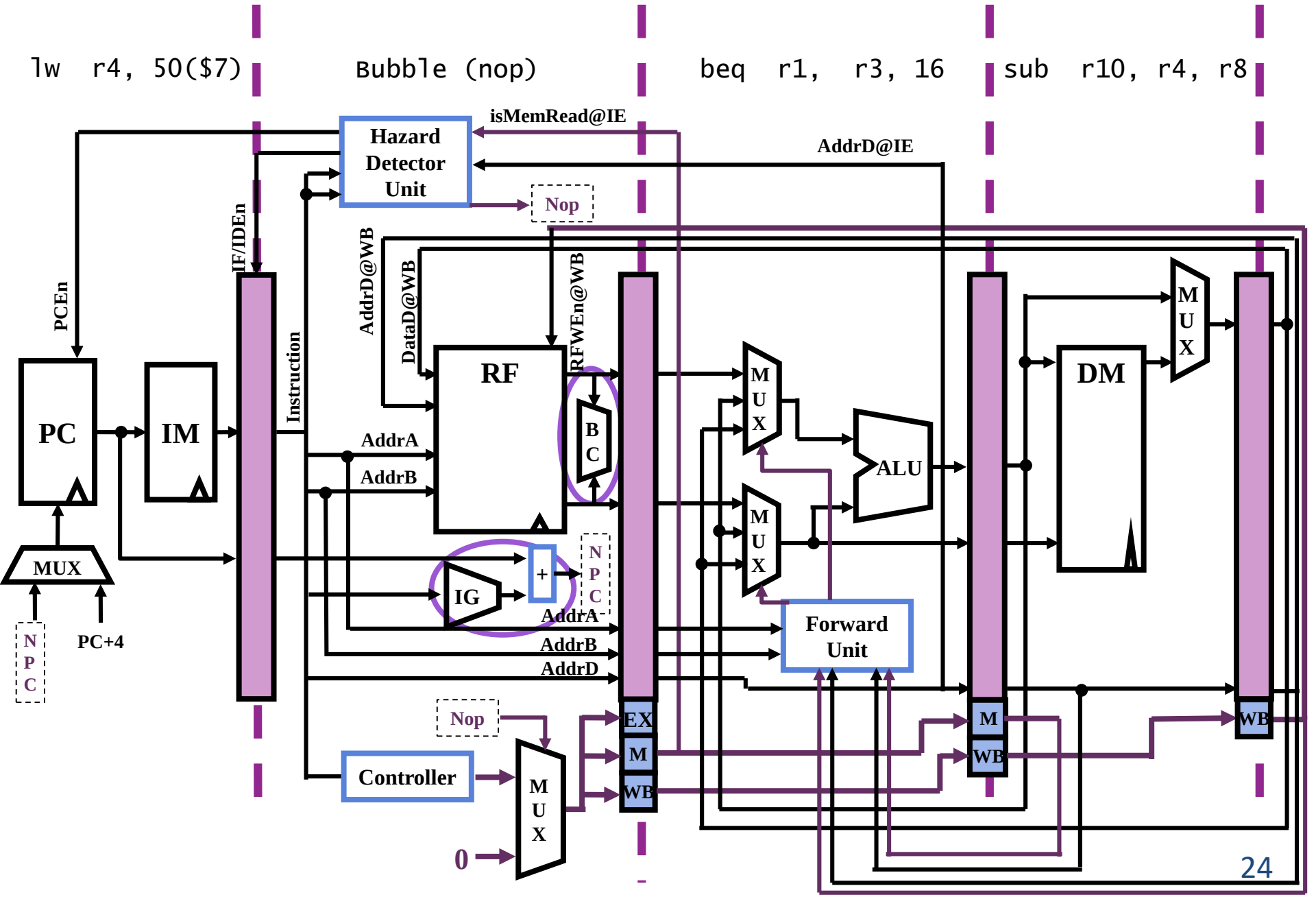
指令	时钟									
	1	2	3	4	5	6	7	8	9	10
分支指令i	IF	ID	EX	MEM	WB					
指令i+1		IF	Idle	Idle	Idle					
分支目标			IF	ID	EX	MEM	WB			
分支目标+1				IF	ID	EX	MEM	WB		
分支目标+2					IF	ID	EX	MEM	WB	

(b)分支转移成功

预测分支失败

- 转移不会发生
- 继续执行下一条指令
- 如果正确，则不会在性能上有任何损失
- 如果错误，则要消除影响

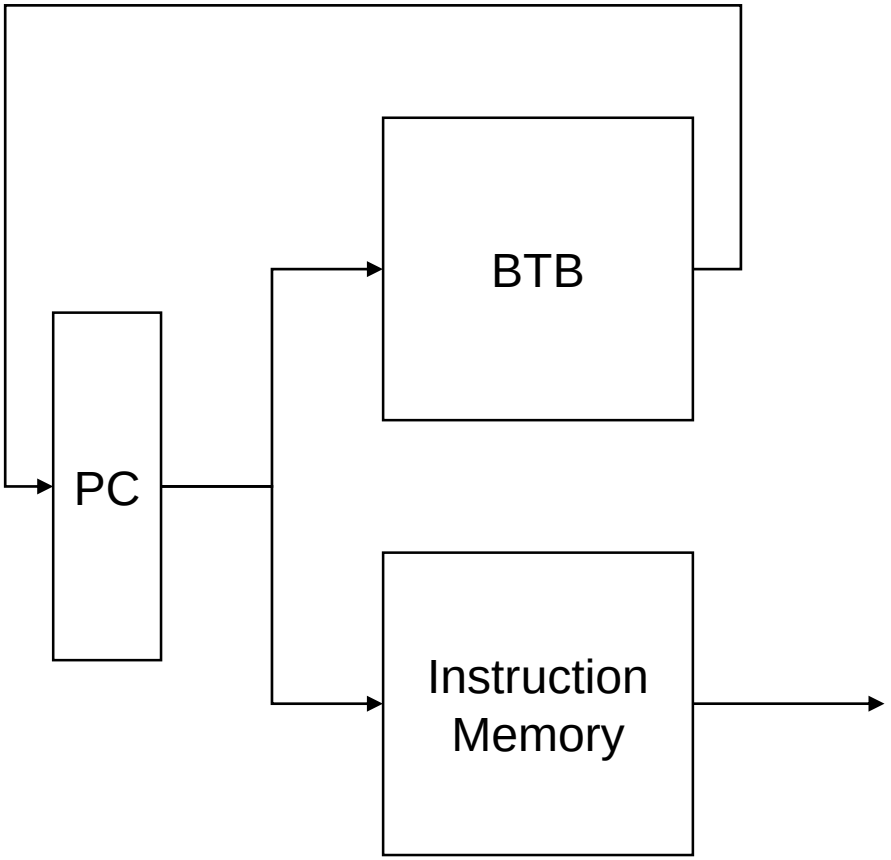
可实现预测功能的CPU



控制冲突的动态调度

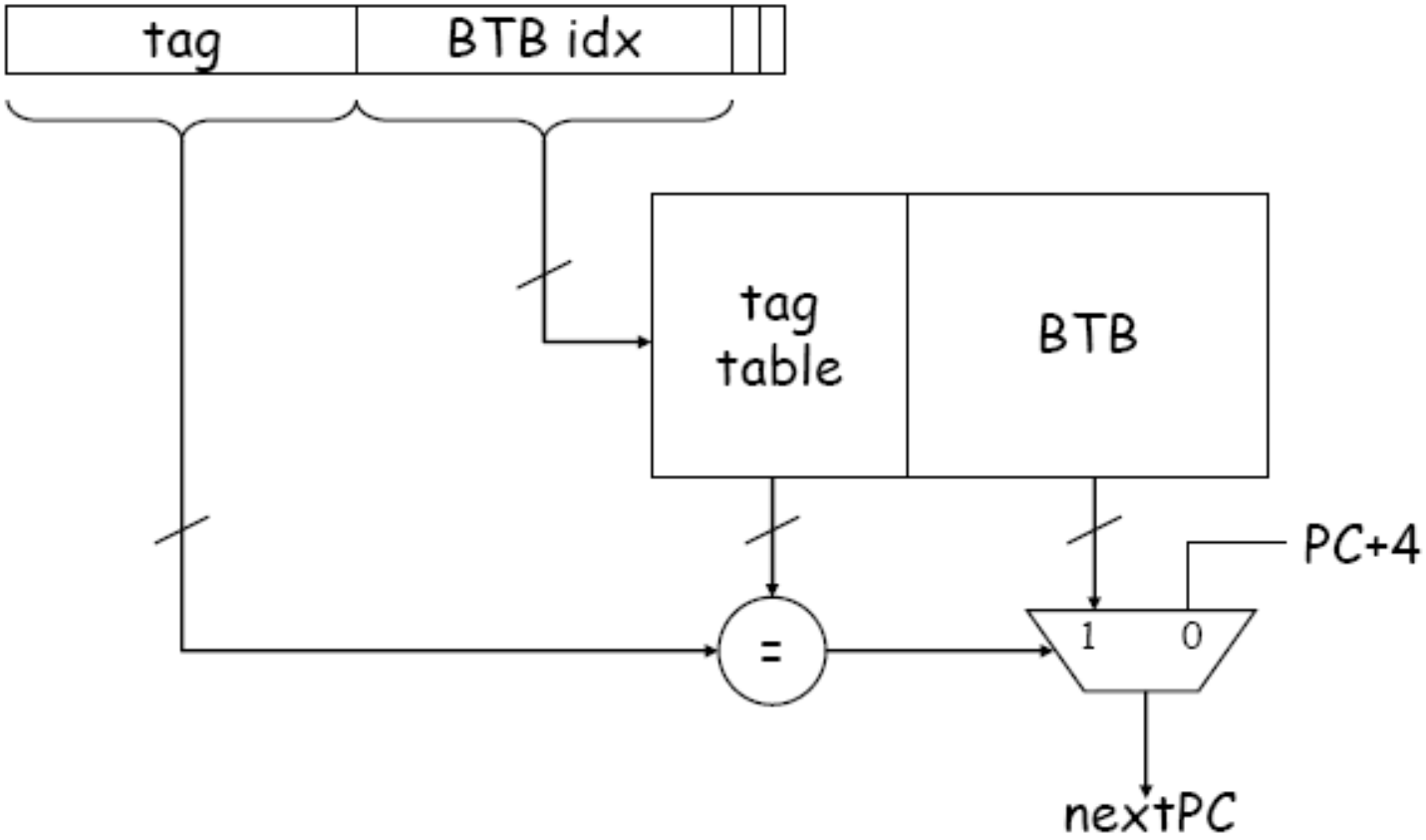
- 分支目标缓冲技术（***Branch Target Buffer***）
  - 缓冲区的每一项内容被用来预测分支转移是否成功，并且根据实际的分支情况对内容进行修改。这种方法是基于如下的考虑：如果本次分支转移成功了，那么预测下一次分支转移也成功，例如一个循环体就是这种情况。
  - 将分支转移成功的分支指令的地址和它的分支目标地址都放到一个缓冲区中保存起来，缓冲区以分支指令的地址作为标志；在取指令阶段，所有的指令地址都与保存的标志作比较，如果相同，就认为本条指令是分支指令，而且认为它分支转移成功，同时它的分支目标（下一条指令）地址就是保存在缓冲区中的分支目标地址。

实现预测的方法



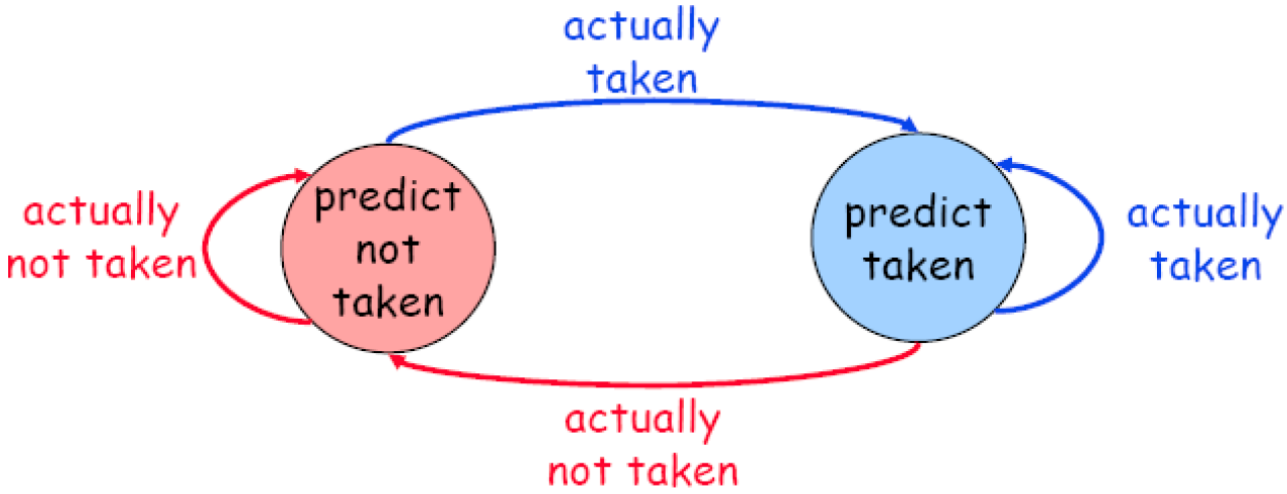
- 增加分支目的地址存储（Branch Target Buffer）
- 首次遇到新的PC，保存它的下一条指令的地址
  - PC + 4　　alu/lw/sw
  - PC + offset　　BEQ/JAL
  - ??　　JAL
- 机理
  - 程序运行的局部性

减少BTB容量



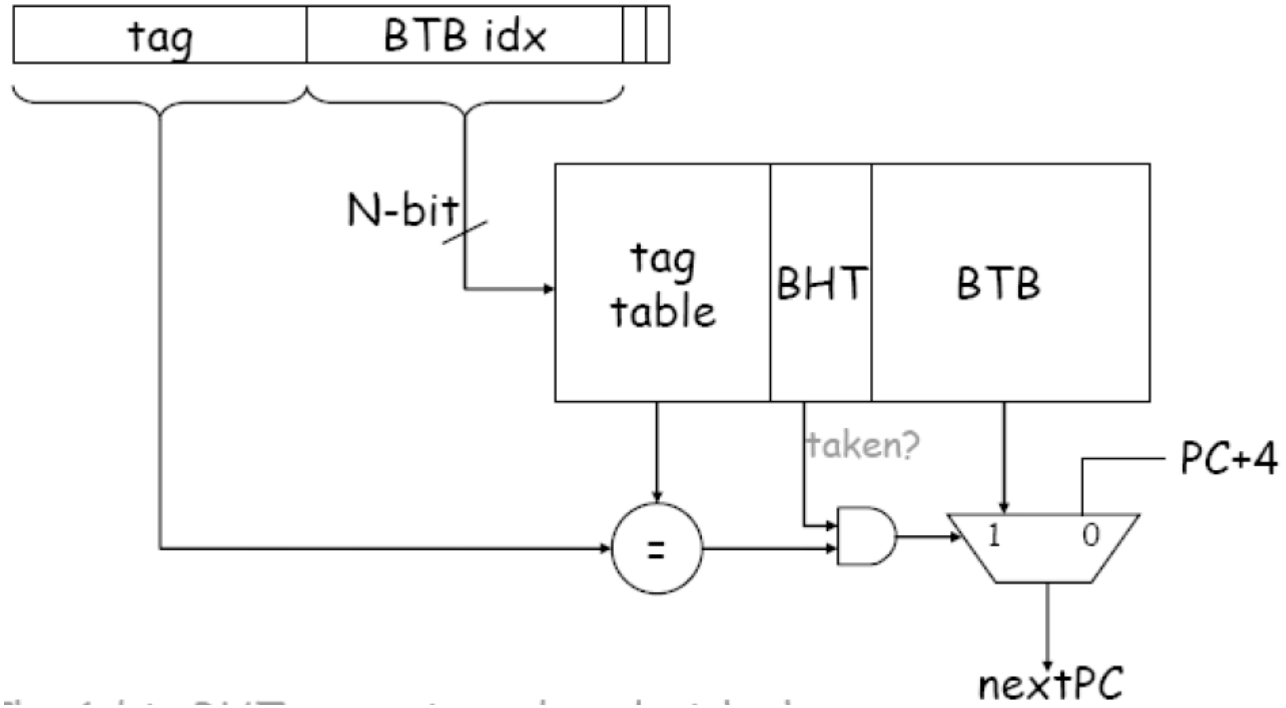
- 只保存转移指令的PC值

动态预测



- 统计规律表明，如果上次发生转移，则本次发生转移的概率更大
- 反之亦然
- 提高预测准确率到90%以上

动态预测



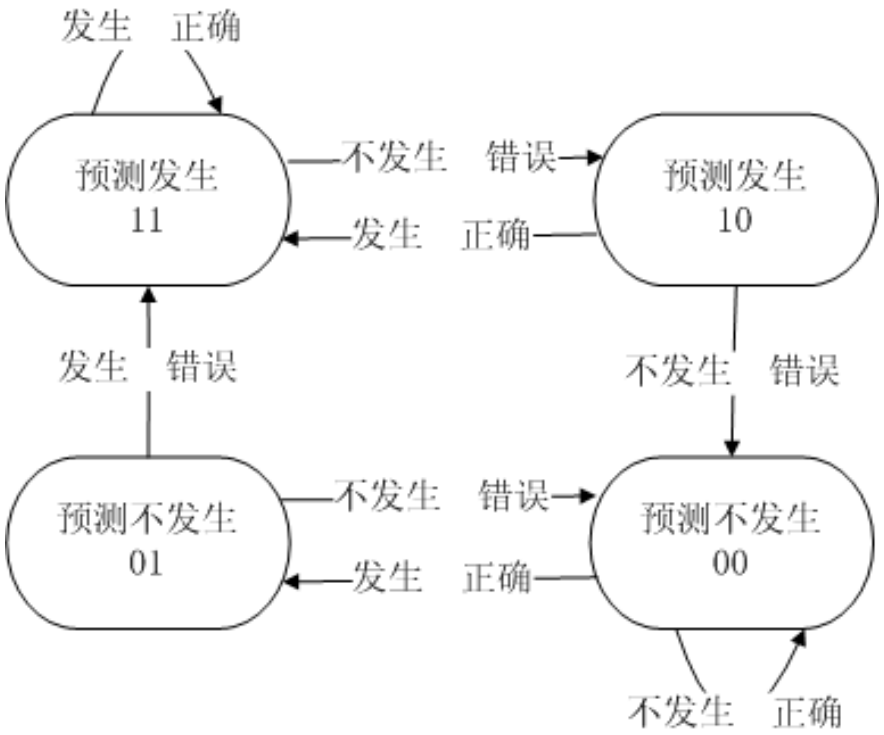
- 增加1位BHT位，实现动态预测
  - BHT: Branch history table

程序设计举例

```
.section .text
.globl _start
_start:
    addi t0, x0, 0x1
    addi t1, x0, 0x0
    addi t3, x0, 0xa
loop:
    add t1, t1, t0
    add t0, t0, 0x1
    add t3, t3, -1
    bne t3, x0, loop
    nop
    jr ra
    nop
```

- 静态预测
  - 顺序执行
  - 准确率：1/10
- 动态预测
  - 第1次错误
  - 第2~9次正确
  - 第10次错误
- 准确率：8/10

动态预测（2位预测）



- 连续两次预测错误时才改变预测方向
- 对多重循环，可进一步提高预测准确率

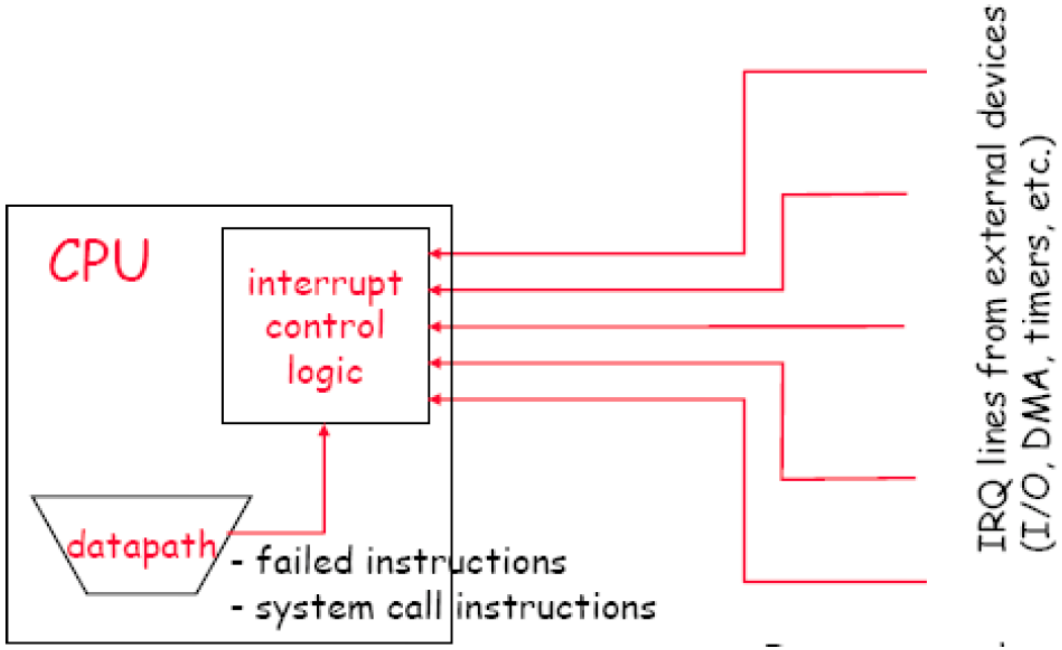
异常

- 什么是异常？
  - CPU运行时会遇到突发的不正常事件
    - 指令执行过程中发生错误
      - 取指令、指令译码、计算、访存
    - 外部设备提出服务请求
    - 多进程运行时与其他进程发生资源冲突
- 中断程序正常执行流程的事件
  - 来自CPU，称为异常
  - 来自外部设备，称为中断

异常响应和处理

- 很难在程序中进行处理
  - 时间、空间的不确定性
- 对程序“透明”处理
  - 程序正常运行
  - 由硬件检测发生异常或中断
  - “透明”地自动转换到服务程序进行响应和处理
    - 保存现场
    - 启动服务程序
  - 返回到程序执行
    - 恢复现场

异常处理的实现

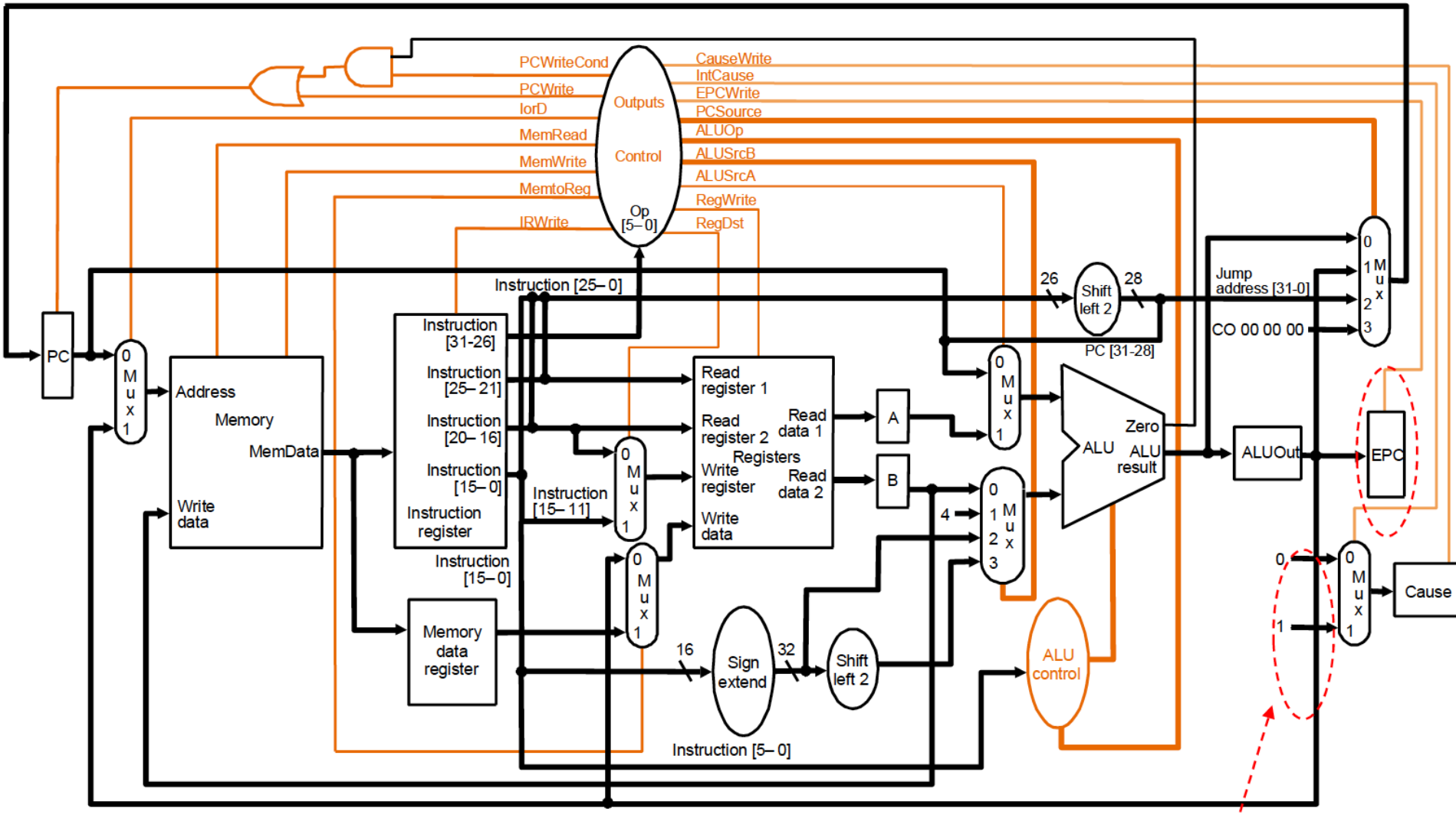


- 增加异常原因寄存器，保存异常的原因
  - 错误的指令地址、错误的操作码、错误的运算结果、错误的数据地址
  - 外部中断请求编号
- 发现异常
  - 增加一个步骤，检查中断寄存器和异常原因寄存器

异常处理的实现

- 保存现场
  - mepc
  - 当前的状态等（状态寄存器）
- 转异常处理程序（中断服务）
  - 根据异常原因，寻址处理程序的入口（PC的来源）
  - 跳转到处理程序执行（与转移指令相同）
- 处理完成后，返回主程序执行

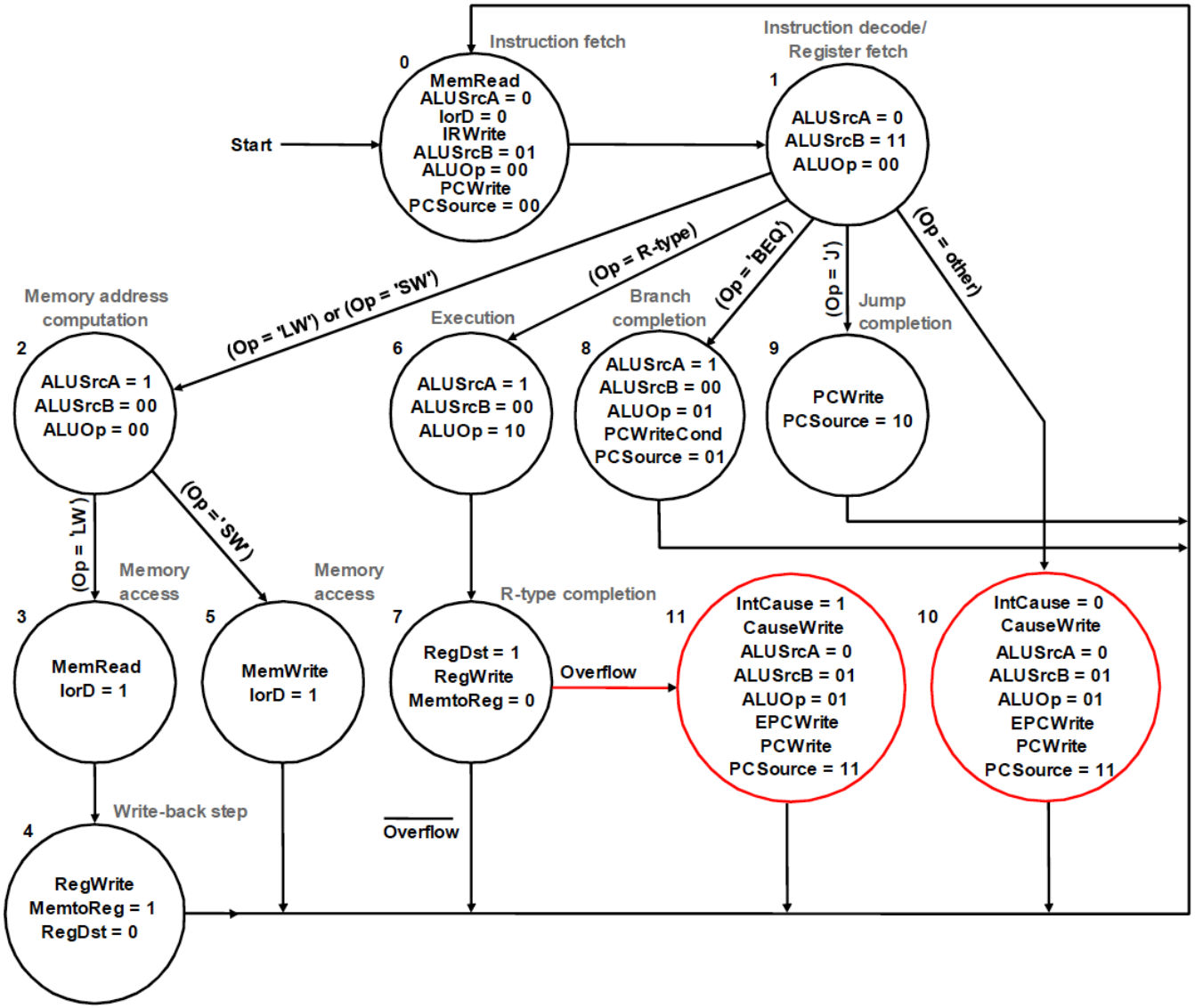
多周期的异常处理



异常类型

多周期的异常处理

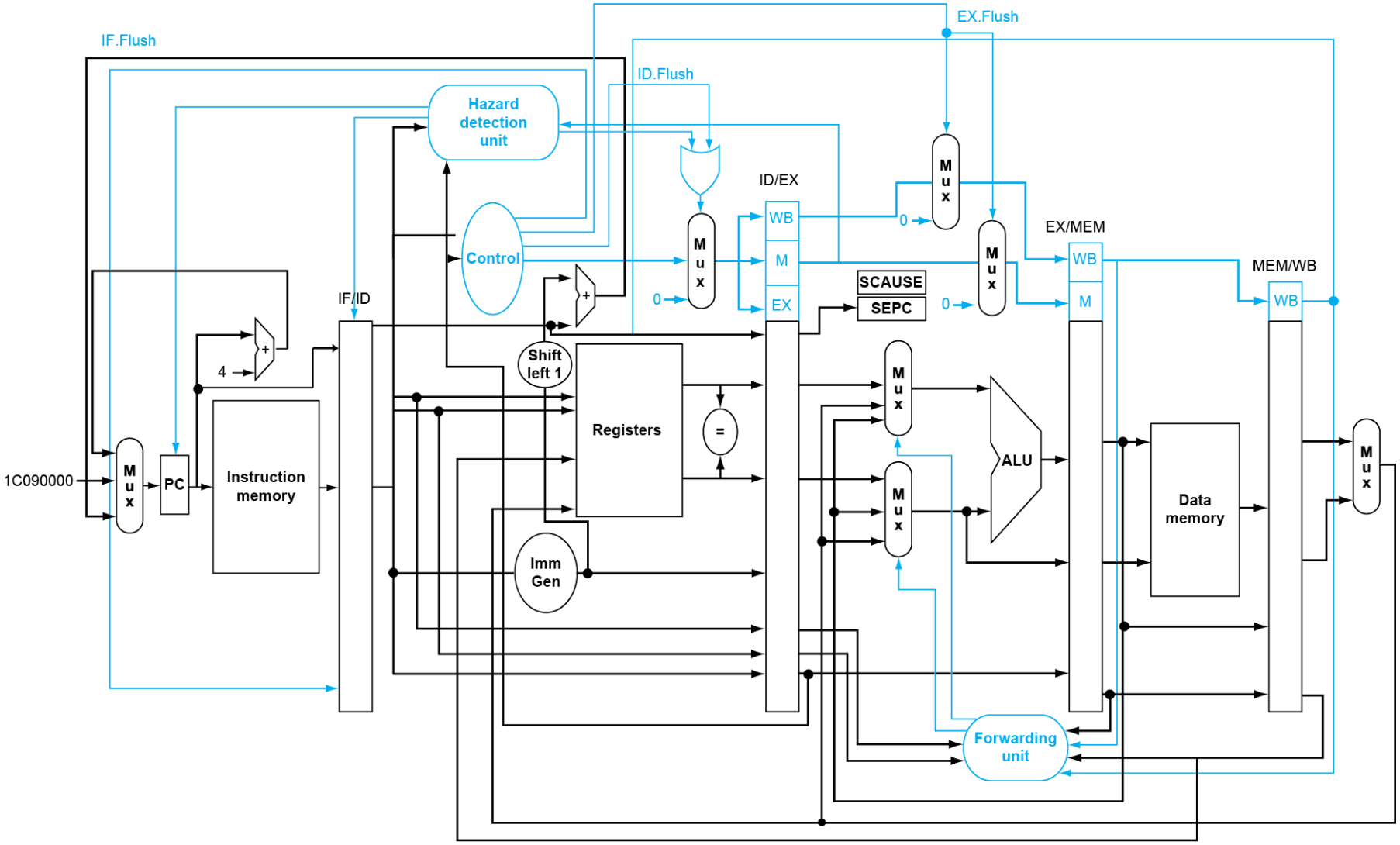
增加一个检查异常（中断）是否发生的步骤



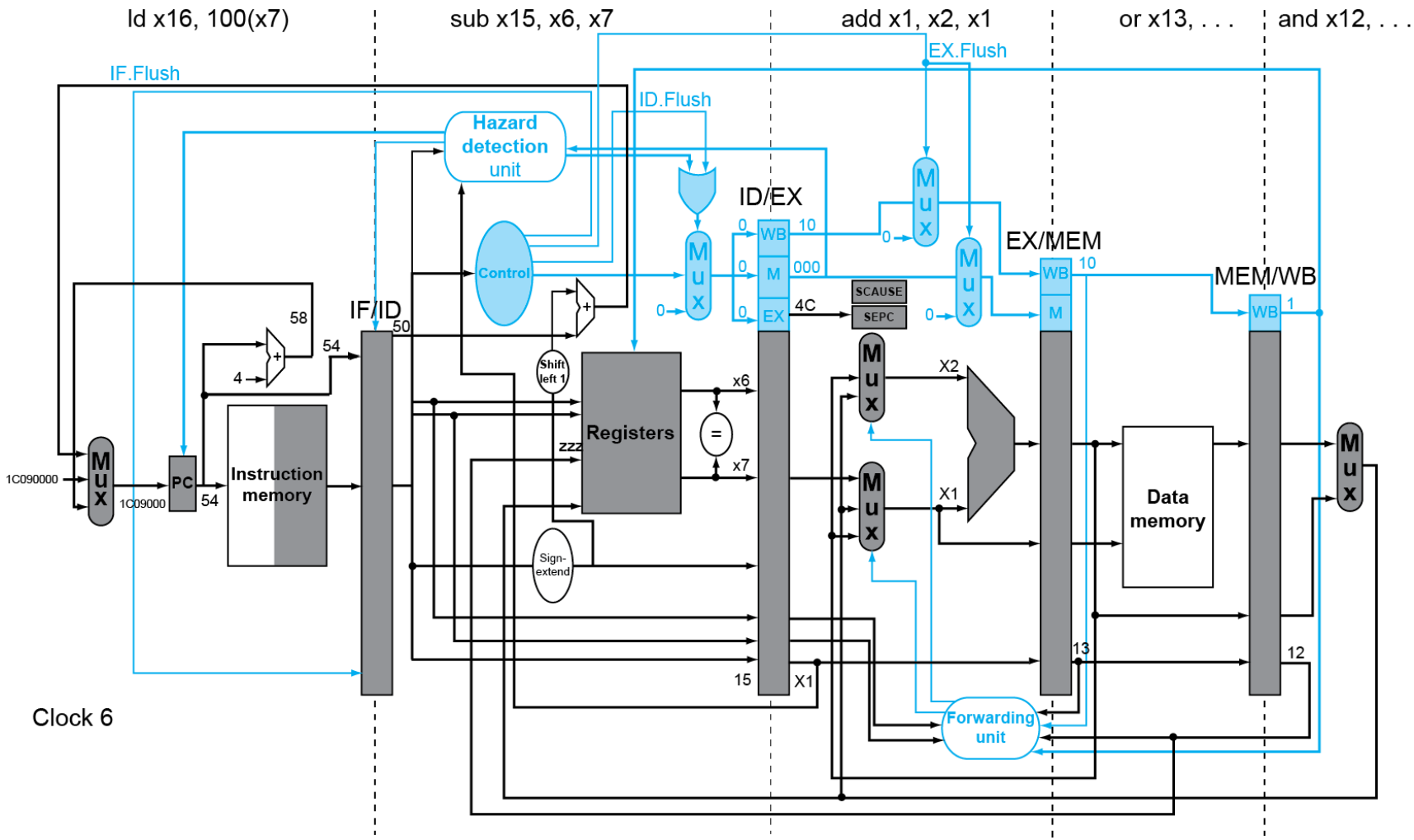
流水线CPU异常处理

- 处理要求
  - 指令流水线中有5条指令正在执行
  - 判断发生异常的位置
  - 保留发生异常的现场
  - 执行异常处理程序
- 硬件
  - mepc
  - mcause

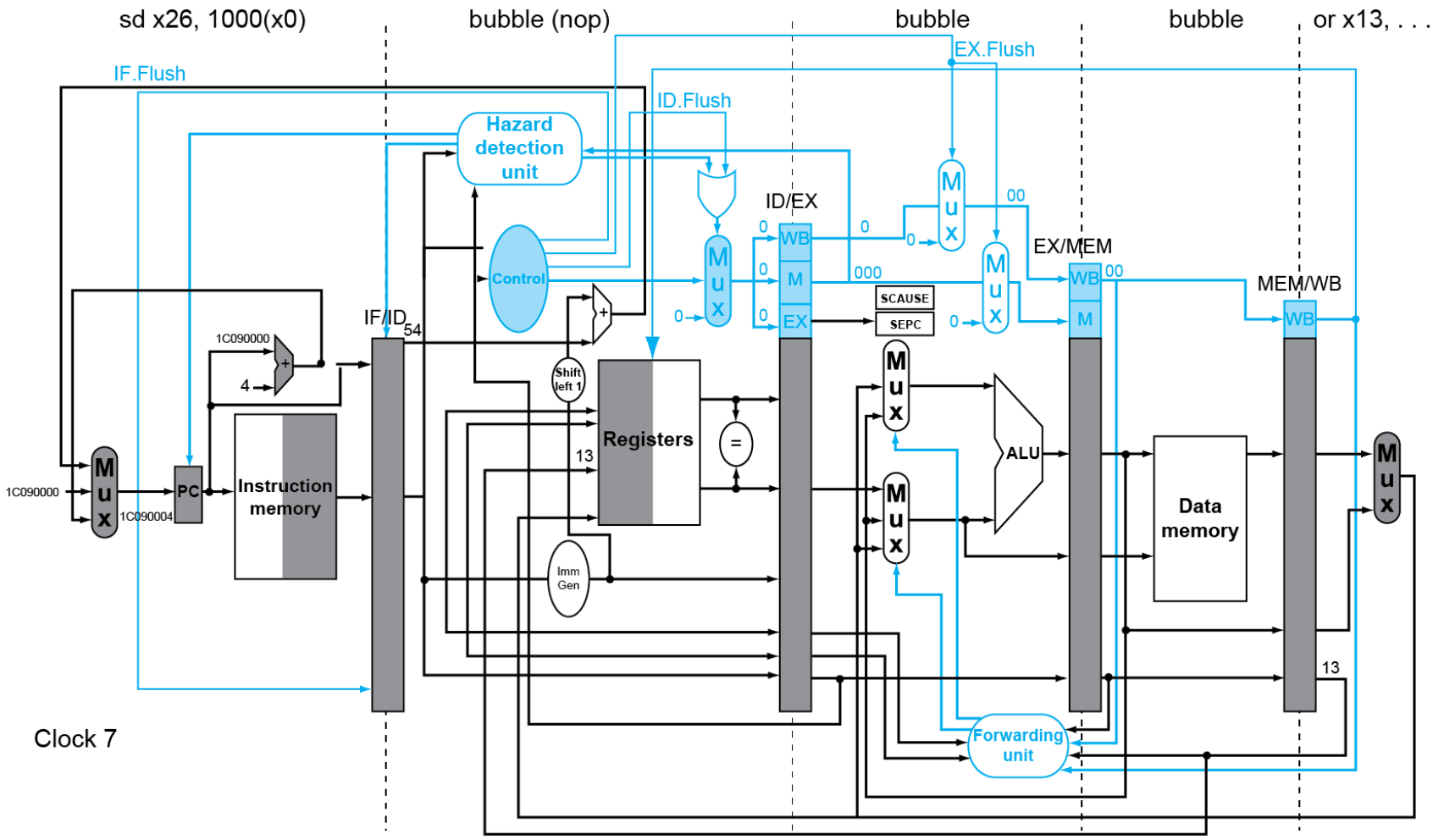
流水线CPU的异常处理



异常示例


ld x16, 100(x7)
sub x15, x6, x7
add x1, x2, x1
or x13, . . .
and x12, . . .
IF.Flush
EX.Flush
ID.Flush
Hazard detection unit
Control
ID/EX
EX/MEM
MEM/WB
IF/ID
WB
M
EX
SCAUSE
SEPC
4C
Shift left 1
Registers
Sign-extend
Instruction memory
Data memory
Forwarding unit
PC
Mux
ALU
1C090000
1C09000
Clock 6

异常示例


sd x26, 1000(x0)
bubble (nop)
bubble
bubble
or x13, . . .
IF.Flush
ID.Flush
EX.Flush
Hazard detection unit
Control
ID/EX
EX/MEM
MEM/WB
WB
M
EX
SCAUSE
SEPC
IF/ID
54
1C090000
4
1C090004
PC
Instruction memory
Shift left 1
Registers
13
Imm Gen
ALU
Data memory
Forwarding unit
Clock 7

精确处理和非精确处理

- 精确异常处理
  - mepc中保存有发生异常指令的地址
  - 操作系统处理简单
  - 指令流水情况下实现比较复杂
- 非精确异常处理
  - mepc中保存当前PC或者近似的PC
  - 由操作系统处理

小结

- 冲突问题及解决方案
  - 结构冲突
  - 数据冲突
  - 控制冲突
- 控制冲突
  - 全局冲突
  - 综合解决方案
    - 硬件、编译器
    - 猜测、尝试
- 异常
  - 发现和响应

阅读和思考

- 阅读

- 思考
  - 与1位预测位比较，2位预测位对于何种循环程序准确率高？极端情况下是否会降低预测准确率？
  - 流水情况下，实现中断的具体方案。
- 实践
  - 根据每组的指令系统，设计大实验数据通路，每组提交1份到网络学堂。

谢谢

可实现预测功能的CPU