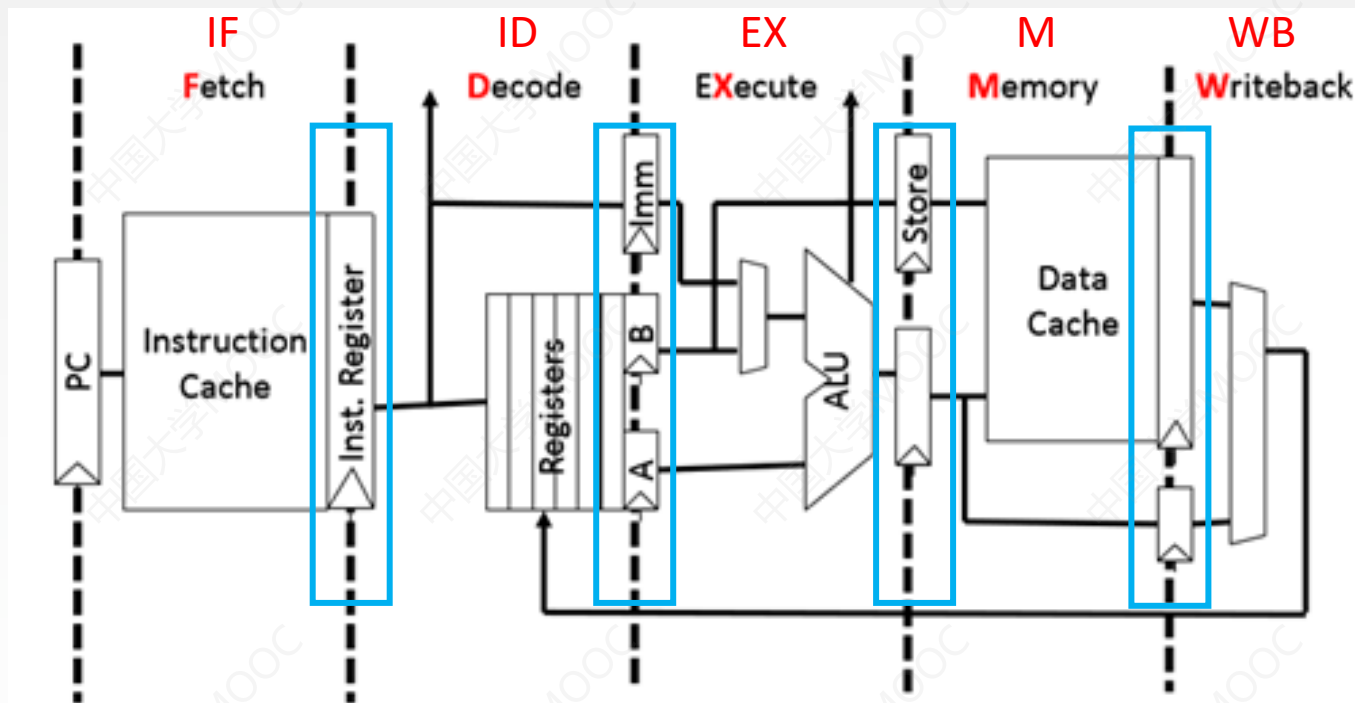


本节内容

五段式指令 流水线

机器周期的设置



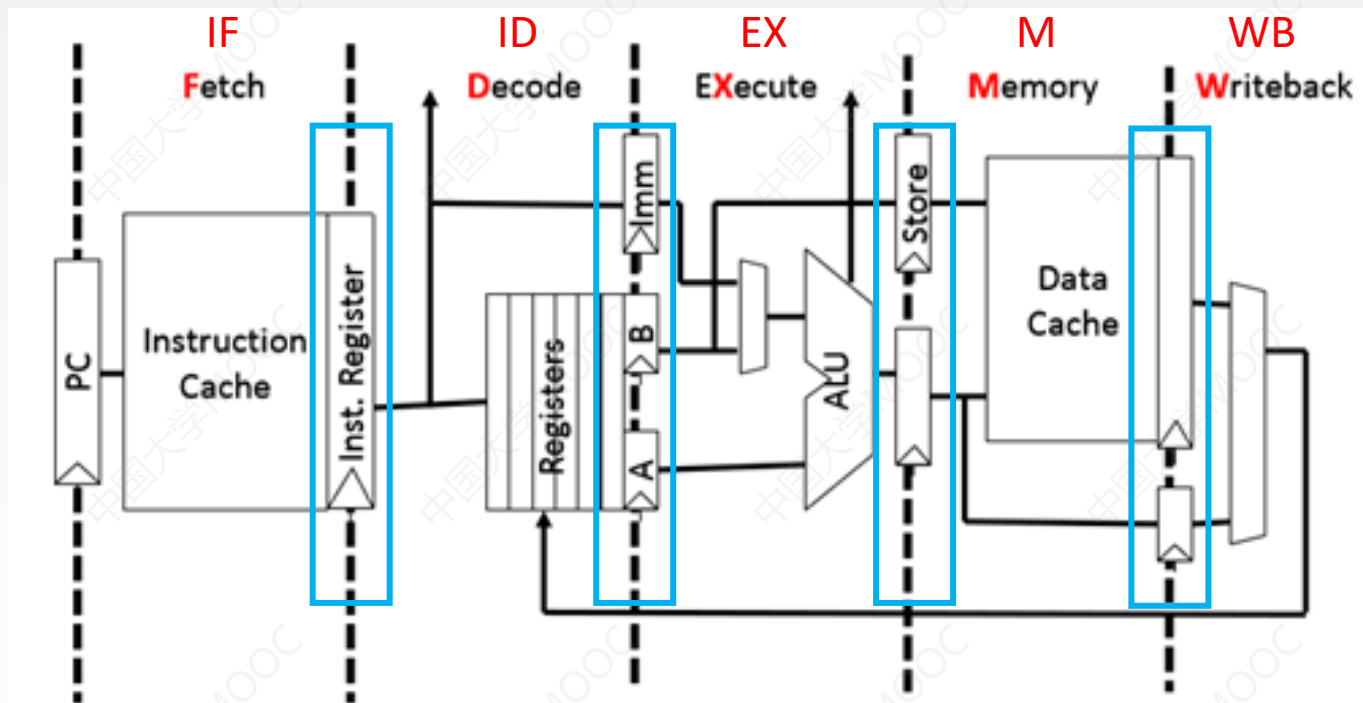
流水线每一个功能段部件后面都要有一个缓冲寄存器，或称为锁存器，其作用是保存本流水段的执行结果，提供给下一流水段使用。

各部件实际耗时： 100ns 80ns 70ns 50ns 50ns

为方便流水线的设计，将每个阶段的耗时取成一样，以最长耗时为准。
即此处应将机器周期设置为100ns。

理想情况下，每个机器周期（功能段）只消耗一个时钟周期。

五段式指令流水线



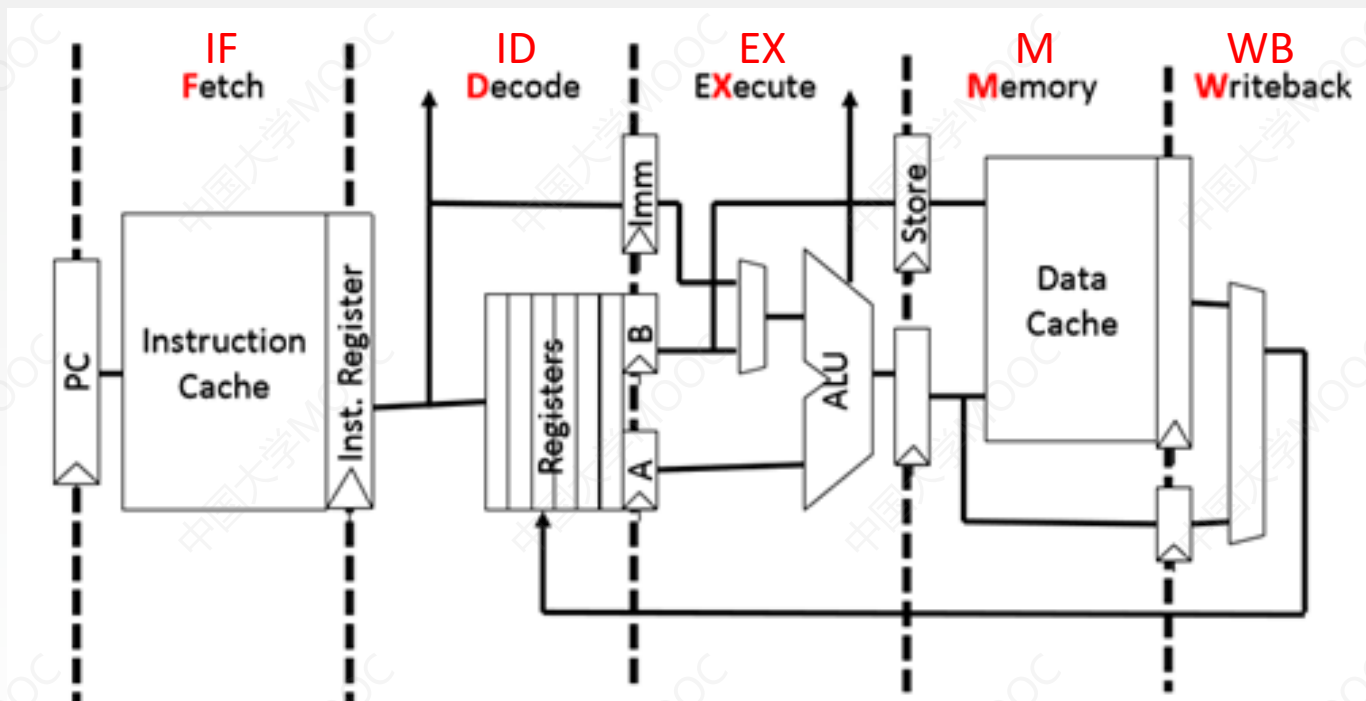
①IF取指 → ②ID译码&取数 → ③EX 执行 → ④M访存 → ⑤WB写回寄存器

考试中常见的五类指令：

- 运算类指令、LOAD指令、STORE指令、条件转移指令、无条件转移指令



运算类指令的执行过程



注:

R_s 指源操作数 (source)

R_d 指目的操作数 (destination)

运算类指令

IF: 根据PC从指令Cache取指令至IF段的锁存器

ID: 取出操作数至ID段锁存器

EX: 运算, 将结果存入EX段锁存器

M: 空段

WB: 将运算结果写回指定寄存器

①IF取指 → ②ID译码&取数 → ③EX 执行 → ④M访存 → ⑤WB写回寄存器

运算类指令举例

加法指令 (两个寄存器相加):

加法指令 (寄存器与立即数相加):

算数左移指令:

指令的汇编格式

ADD R_s, R_d

ADD #996, R_d

SHL R_d

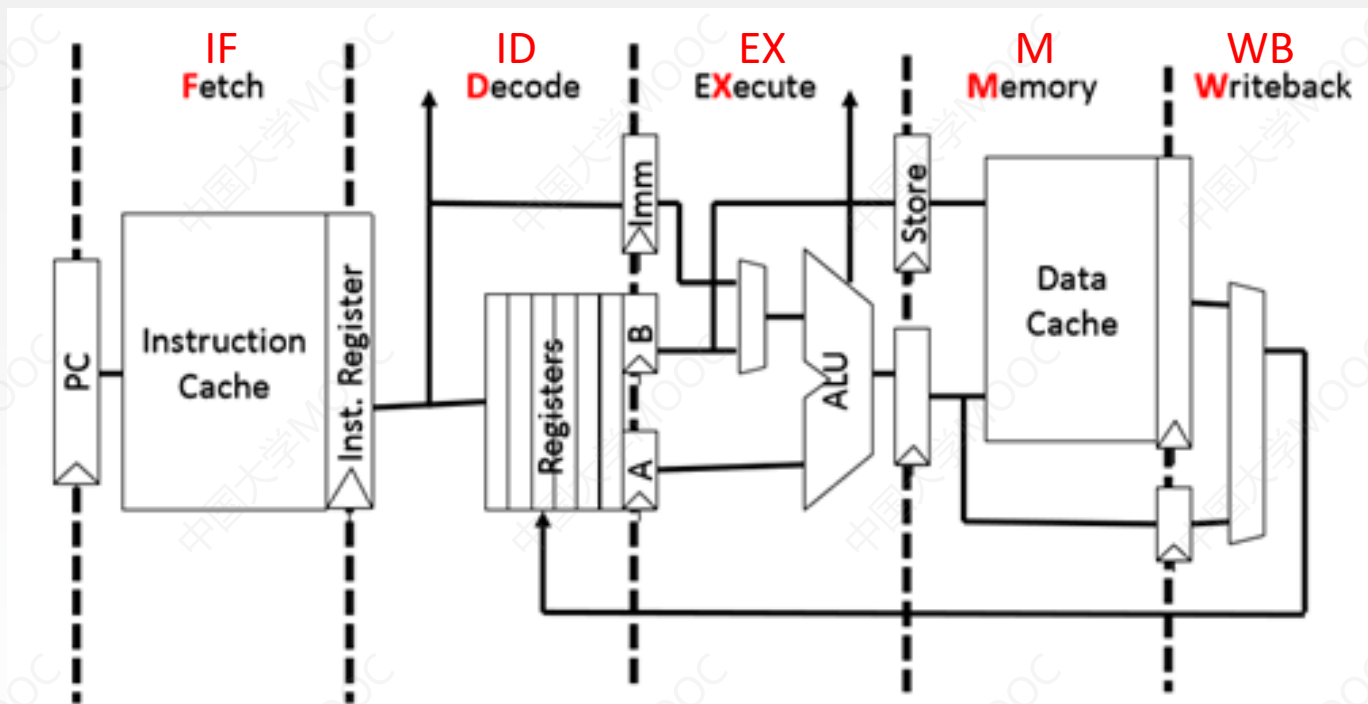
功能

$(R_s) + (R_d) \rightarrow R_d$

$996 + (R_d) \rightarrow R_d$

$(R_d) \lll 2 \rightarrow R_d$

LOAD指令的执行过程



LOAD指令

IF: 根据PC从指令Cache取指令至IF段的锁存器

ID: 将基址寄存器的值放到锁存器A，将偏移量的值放到Imm

EX: 运算，得到有效地址

M: 从数据Cache中取数并放入锁存器

WB: 将取出的数写回寄存器

①IF取指 → ②ID译码&取数 → ③EX 执行 → ④M访存 → ⑤WB写回寄存器

指令的汇编格式

LOAD Rd,996(Rs)

或简写为:

LOAD Rd,mem

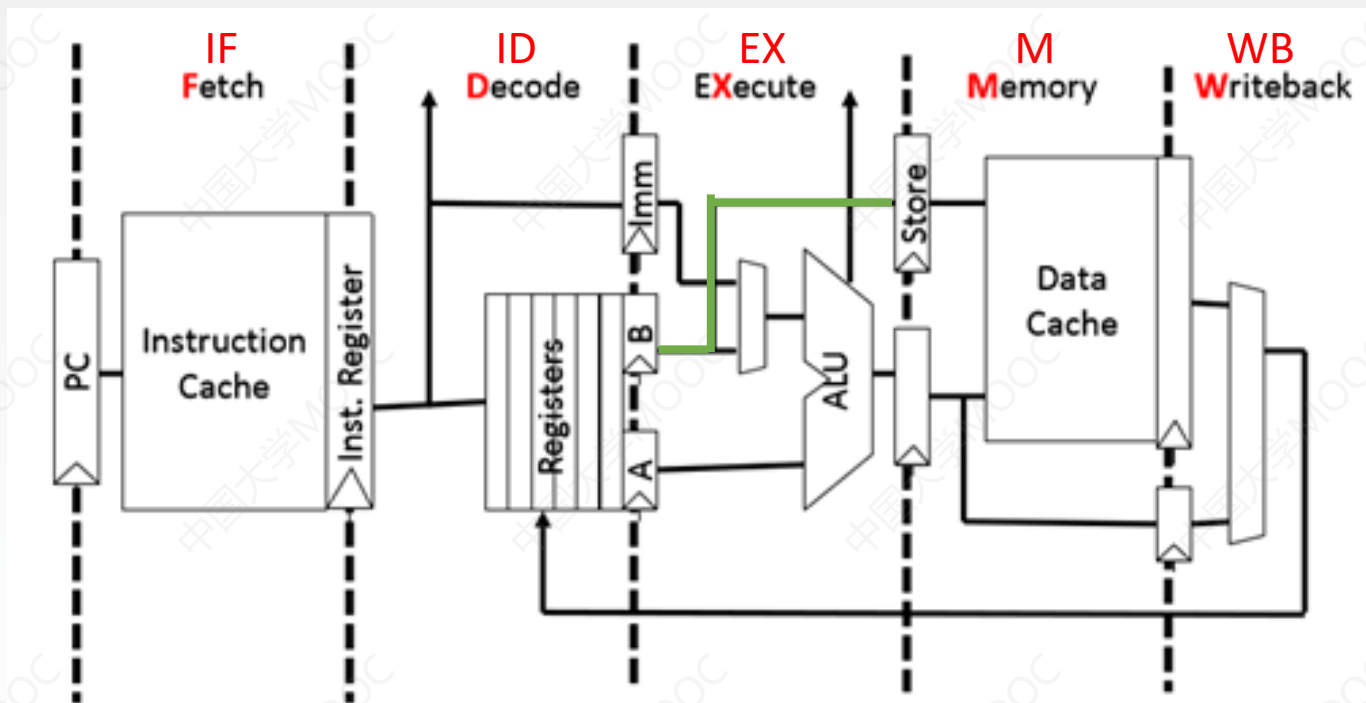
功能

$(996+(Rs)) \rightarrow Rd$

$(mem) \rightarrow Rd$

通常，RISC处理器只有“取数LOAD”和“存数STORE”指令才能访问主存

STORE指令的执行过程



STORE指令

IF: 根据PC从指令Cache取指令至IF段的锁存器

ID: 将基址寄存器的值放到锁存器A，将偏移量的值放到Imm。将要存的数放到B

EX: 运算，得到有效地址。并将锁存器B的内容放到锁存器Store。

M: 写入数据Cache

WB: 空段

①IF取指 → ②ID译码&取数 → ③EX 执行 → ④M访存 → ⑤WB写回寄存器

指令的汇编格式

STORE Rs,996(Rd)

或简写为:

STORE Rs,mem

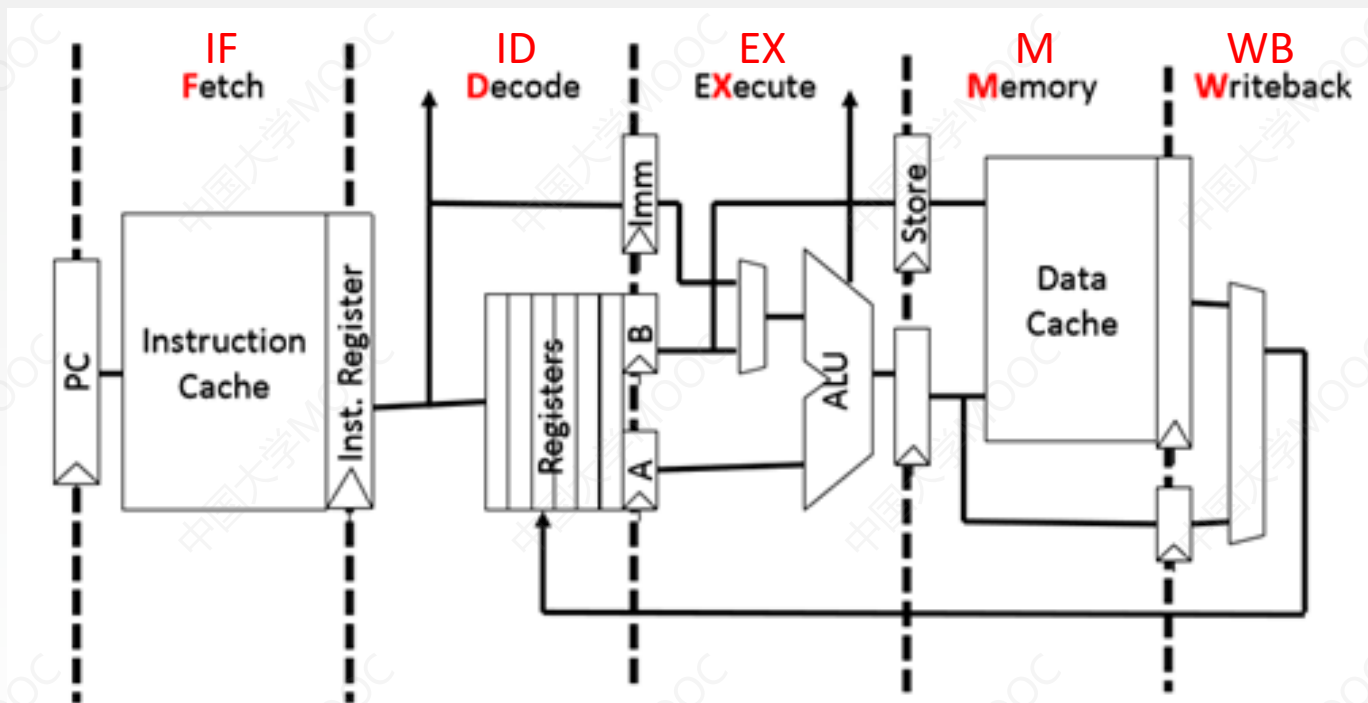
功能

$Rs \rightarrow (996 + (Rd))$

$Rs \rightarrow (mem)$

条件转移指令的执行过程

转移类指令常采用相对寻址



条件转移指令

IF: 根据PC从指令Cache取指令至IF段的锁存器

ID: 进行比较的两个数放入锁存器A、B；偏移量放入 Imm

EX: 运算，比较两个数

M: 将目标PC值写回PC（左图没画全）

WB: 空段

很多教材把写回PC的功能段称为“WrPC段”，其耗时比M段更短，可安排在M段时间内完成

①IF取指 → ②ID译码&取数 → ③EX 执行 → ④M访存 → ⑤WB写回寄存器

注：通常在IF段结束止之后PC就会自动 + “1”

指令的汇编格式

功能

beq Rs, Rt, #偏移量

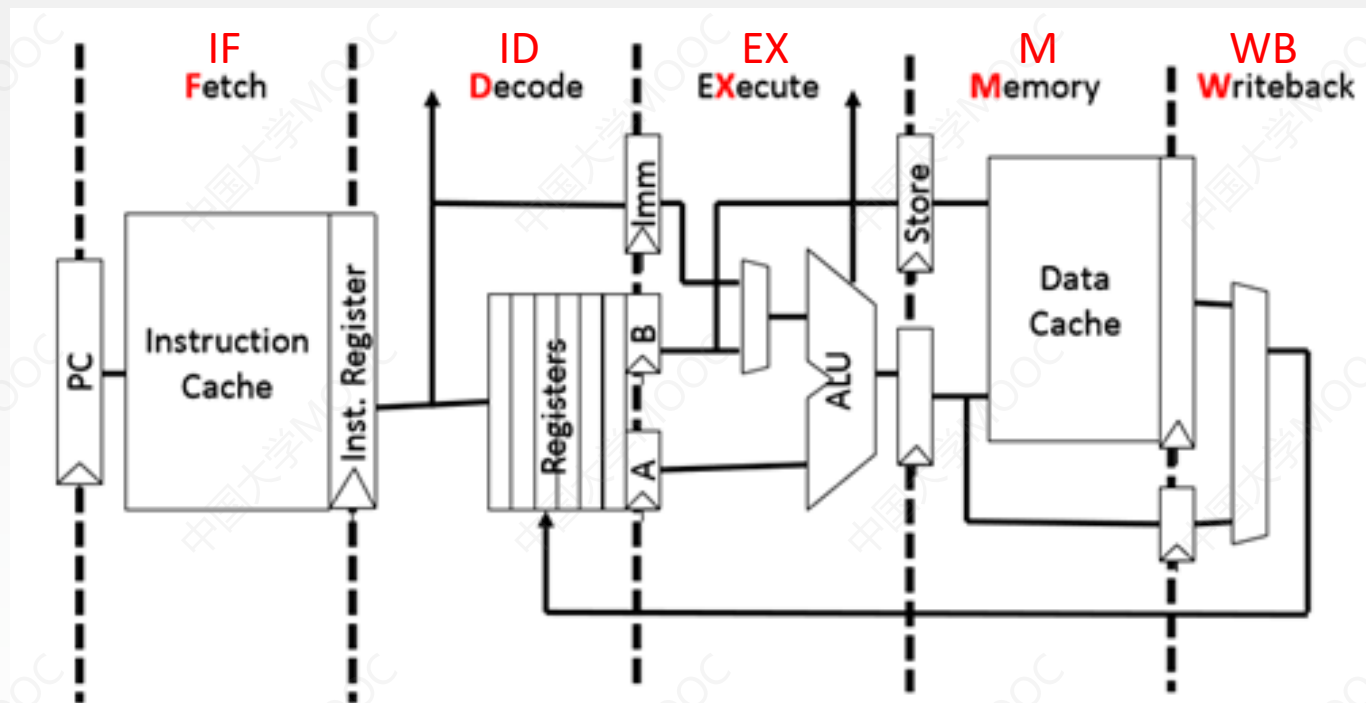
若(Rs)=(Rt),则 $(PC)+指令字长+(偏移量 \times 指令字长) \rightarrow PC$; 否则 $(PC)+指令字长 \rightarrow PC$

bne Rs, Rt, #偏移量

若(Rs)!(Rt),则 $(PC)+指令字长+(偏移量 \times 指令字长) \rightarrow PC$; 否则 $(PC)+指令字长 \rightarrow PC$

无条件转移指令的执行过程

转移类指令常采用相对寻址



无条件转移指令

IF: 根据PC从指令Cache取指令至IF段的锁存器

ID: 偏移量放入 Imm

EX: 将目标PC值写回PC (左图没画全)

M: 空段

WB: 空段

①IF取指 → ②ID译码&取数 → ③EX 执行 → ④M访存 → ⑤WB写回寄存器

指令的汇编格式

`jmp #偏移量`

功能

$(PC) + \text{指令字长} + (\text{偏移量} \times \text{指令字长}) \rightarrow PC$

“WrPC段”耗时比EX段更短，可安排在EX段时间内完成。WrPC段越早完成，就越能避免控制冲突。当然，也有的地方会在WB段时间内才修改PC的值

例题

例题. 假设某指令流水线采用“按序发射，按序完成”方式，没有采用转发技术处理数据相关，并且同一寄存器的读和写操作不能在同一个时钟周期内进行。若高级语言程序中某赋值语句为 $x=a+b$ ， x 、 a 和 b 均为int型变量，它们的存储单元地址分别表示为 $[x]$ 、 $[a]$ 和 $[b]$ 。该语句对应的指令序列及其在指令流中的执行过程如下图所示。

I1 LOAD R1, [a] $M[a] \rightarrow R1$
I2 LOAD R2, [b] $M[b] \rightarrow R2$
I3 ADD R1, R2 $(R1) + (R2) \rightarrow R2$
I4 STORE R2, [x] $(R2) \rightarrow M[x]$

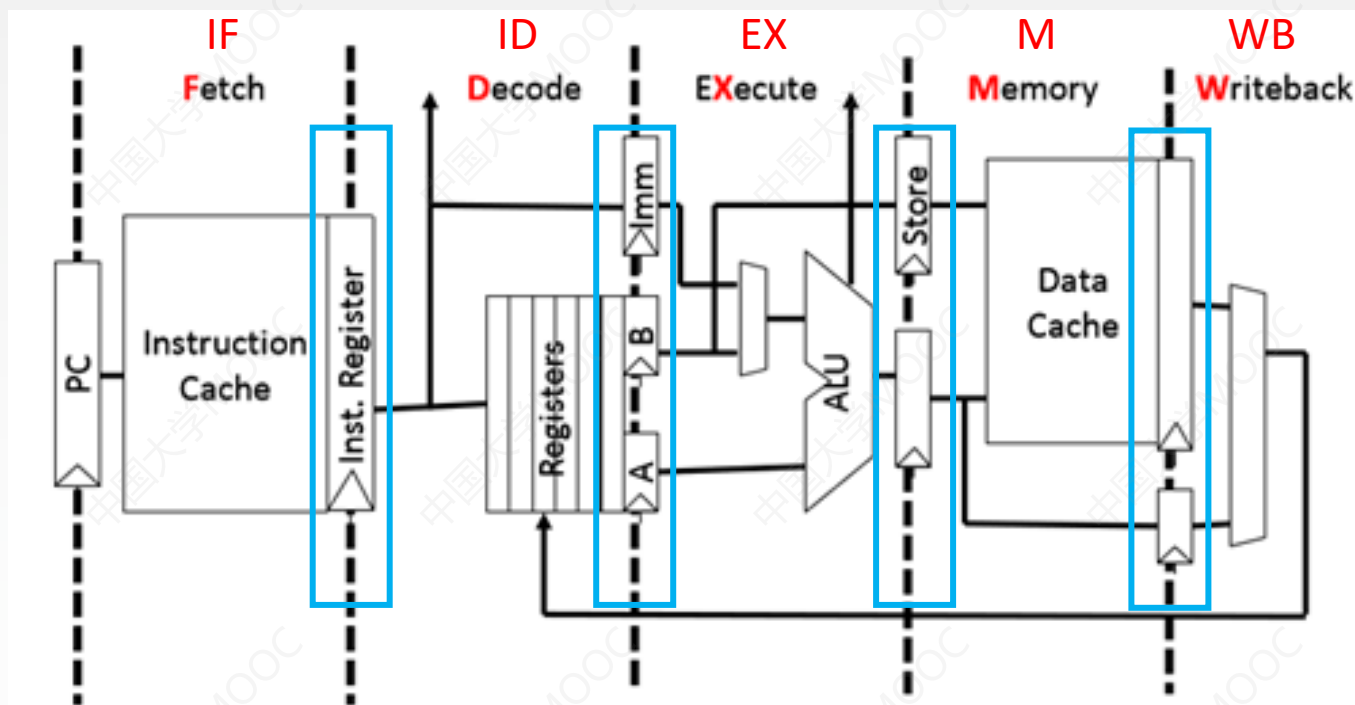
则这4条指令执行过程中I3的ID段和I4的IF段被阻塞的原因各是什么？

| | 时间单元 | | | | | | | | | | | | | |
|----------------|------|----|----|----|----|----|----|----|---|----|----|----|----|----|
| 指令 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| I ₁ | IF | ID | EX | M | WB | | | | | | | | | |
| I ₂ | | IF | ID | EX | M | WB | | | | | | | | |
| I ₃ | | | IF | | | | ID | EX | M | WB | | | | |
| I ₄ | | | | | | | IF | | | | ID | EX | M | WB |

I3与I1和I2存在数据相关；

I4的IF段必须在I3进入ID段后才能开始，否则会覆盖IF段锁寄存器的内容

五段式指令流水线



①IF取指 → ②ID译码&取数 → ③EX 执行 → ④M访存 → ⑤WB写回寄存器

只有上一条指令进入ID段后，下一条指令才能开始IF段，否则会覆盖IF段锁存器的内容