

某计算机字长16位，标志寄存器Flag中的ZF、SF和OF分别是零标志、符号标志和溢出标志，采用双字节定长指令字。假定该计算机中有一条bgt (大于零转移)指令，其指令格式如下：第一个字节指明操作码和寻址方式，第二个字节为偏移地址Imm8，其功能是：

若 $(ZF+(SF\oplus OF)=0)$ 则 $PC=PC+2+Imm8$ 否则 $PC=PC+2$

完成如下要求并回答问题：

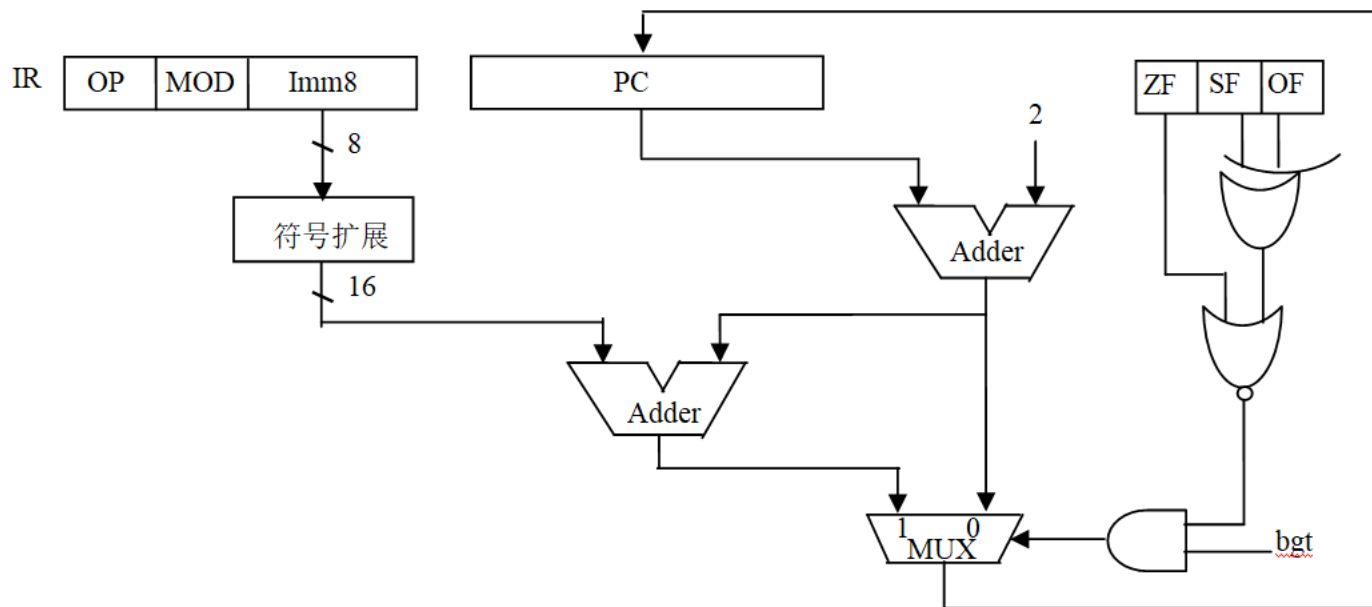
(1) 该计算机存储器的编址单位是多少位？**因为顺序执行时PC的增量是2，且每条指令占2个字节，所以编址单位是字节。**

补充思考：bgt指令执行的是带符号整数比较还是无符号整数比较？偏移地址Imm8是补码表示，那么转移目标地址的范围是什么？

根据“大于”的条件判断表达式 $(ZF+(SF\oplus OF)=0)$ ，可以看出该bgt指令实现的是带符号整数比较。因为无符号整数比较大小时，其判断表达式中应该有借位标志CF，而没有溢出标志OF和符号标志SF。

偏移地址Imm8为补码表示，说明转移目标指令可能在bgt指令之前，也可能在bgt指令之后。计算转移目标地址时，偏移量为Imm8，而不是 $Imm8\times 2$ ，说明Imm8是相对地址，而不是相对指令条数。Imm8占8位，即范围为-128~127，但因为采用双字节定长指令字，所以偏移量不可能是奇数，因此转移目标地址的范围是 $PC+2+(-128)\sim PC+2+126$ ，也即转移目标地址的范围是相对于bgt指令的后（地址更小，负跳）126个单元到前（地址更大，正跳）128个单元之间，用指令条数来衡量的话，就是相对于bgt指令的后63条指令到前64条指令之间。

(2) 实现bgt指令的数据通路如下图所示。



补充：

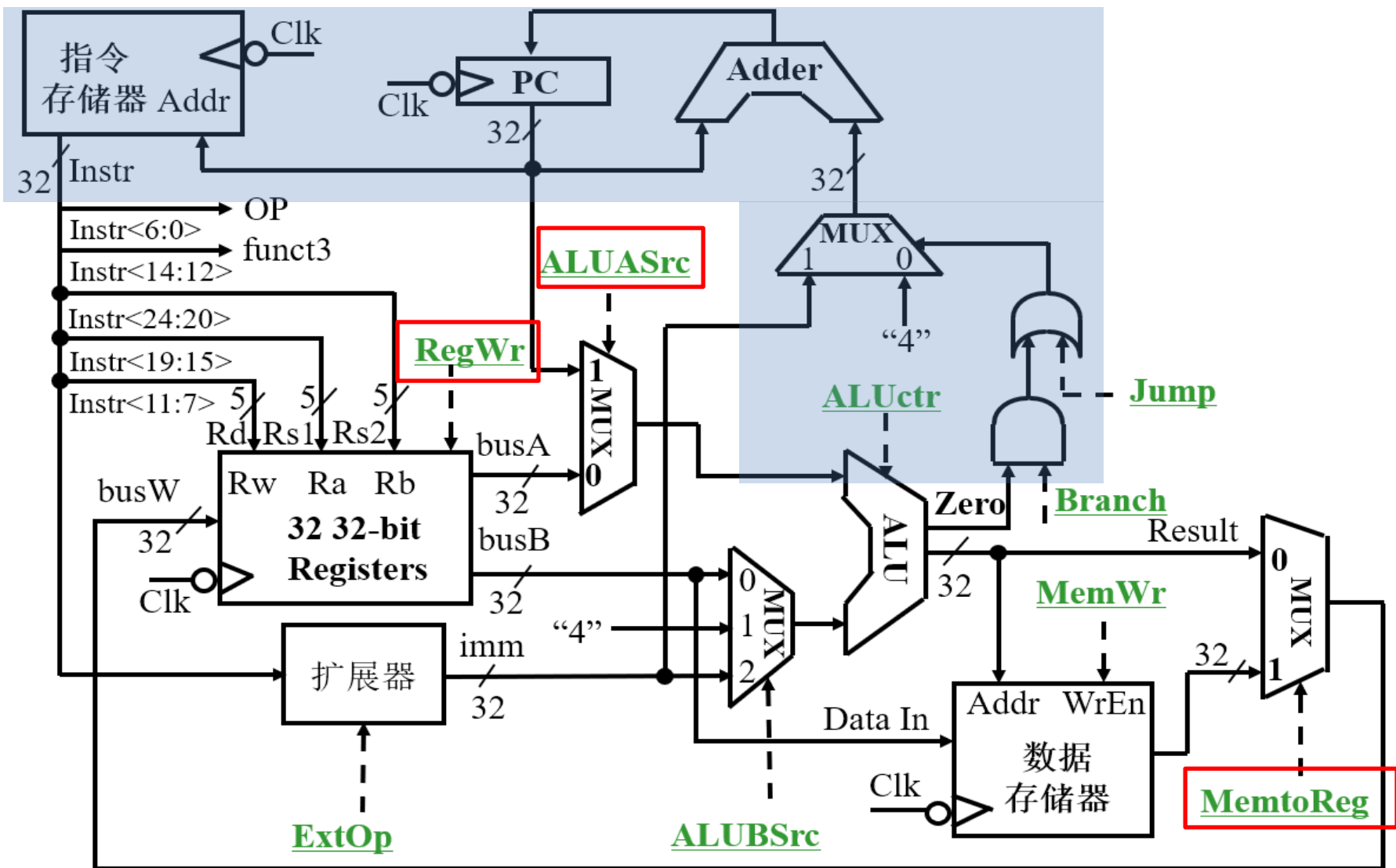
(判断无符号数大于bgtu的条件： $CF=0$ 且 $ZF=0$)

(“小于”的条件是什么？)

$ZF=0$ 且 $CF=1$ ：无符号， $ZF=0$ 且 $SF \oplus OF=1$ ：有符号)

假定图8.18给出的单周期数据通路对应的控制逻辑发生错误，使得控制信号RegWr、ALUASrc、Branch、Jump、MemWr、MemtoReg中某一个在任何情况下总是为0，则该控制信号为0时哪些指令不能正确执行？要求分别对每个控制信号进行讨论。

- 若RegWr=0，则所有需写结果到寄存器的指令（如：R-型指令、I-型运算类指令、load, jal）都不能正确执行，因为寄存器不发生写操作。
- 若ALUASrc=0，则J型指令（jal）可能不能正确执行，当指令中的目标寄存器rd不是x0（0号寄存器）时，需要将PC+4（返回地址）存入rd中，但因为ALUASrc=0，使得PC不能作为ALU的A输入端，因而在ALU中不能完成PC+4的计算。
- 若Branch=0，则branch类（B型）指令可能出错，因为即使条件满足了也不会发生跳转。
- 若Jump=0，则J型指令（jal）出错，因为无法发生跳转。
- 若MemWr=0，则store类型指令出错，因为存储器不能写入所需数据。
- 若MemtoReg=0，则load类型指令发生错误，因为不能选择将存储器读出的内容送目的寄存器。



6 若要在RV32I指令集中增加一条swap指令（功能是实现两个寄存器内容的互换），可以有两种方式。一种是采用伪指令（即软件）方式，这种情况下，当执行到swap指令时，用若干条已有指令构成的指令序列来代替实现；另一种做法是直接改动硬件来实现swap指令，这种情况下，当执行到swap指令时，则可直接在swap指令对应的数据通路（硬件）上执行。

（1）写出用伪指令方式实现“swap rs, rt”时的指令序列（提示：伪指令对应的指令序列中不能使用其他额外寄存器，以免破坏这些寄存器的值）。

xor rs, rs, rt

xor rt, rs, rt

xor rs, rs, rt

（2）假定用硬件实现swap指令时会使每条指令的执行时间增加10%，则swap指令在程序中占多大的比例才值得用硬件方式来实现，而不是用（1）中给出的伪指令方式实现？

假定占比为 x （ $0 \leq x \leq 1$ ），其他指令比例为 $1-x$ ，则用硬件实现该指令时，程序执行时间为原来的 $1.1 \times (x + 1 - x) = 1.1$ 倍。用软件实现该指令时，程序执行时间为原来的 $3x + 1 - x = 2x + 1$ 倍。因此，当 $1.1 < 2x + 1$ 时，硬件实现才有意义，由此可知， $x > 0.05$ （5%）

(3) 采用硬件方式实现时，在不对通用寄存器组进行修改的情况下，能否在单周期数据通路中实现**swap**指令？对于多周期数据通路的情况又怎样？

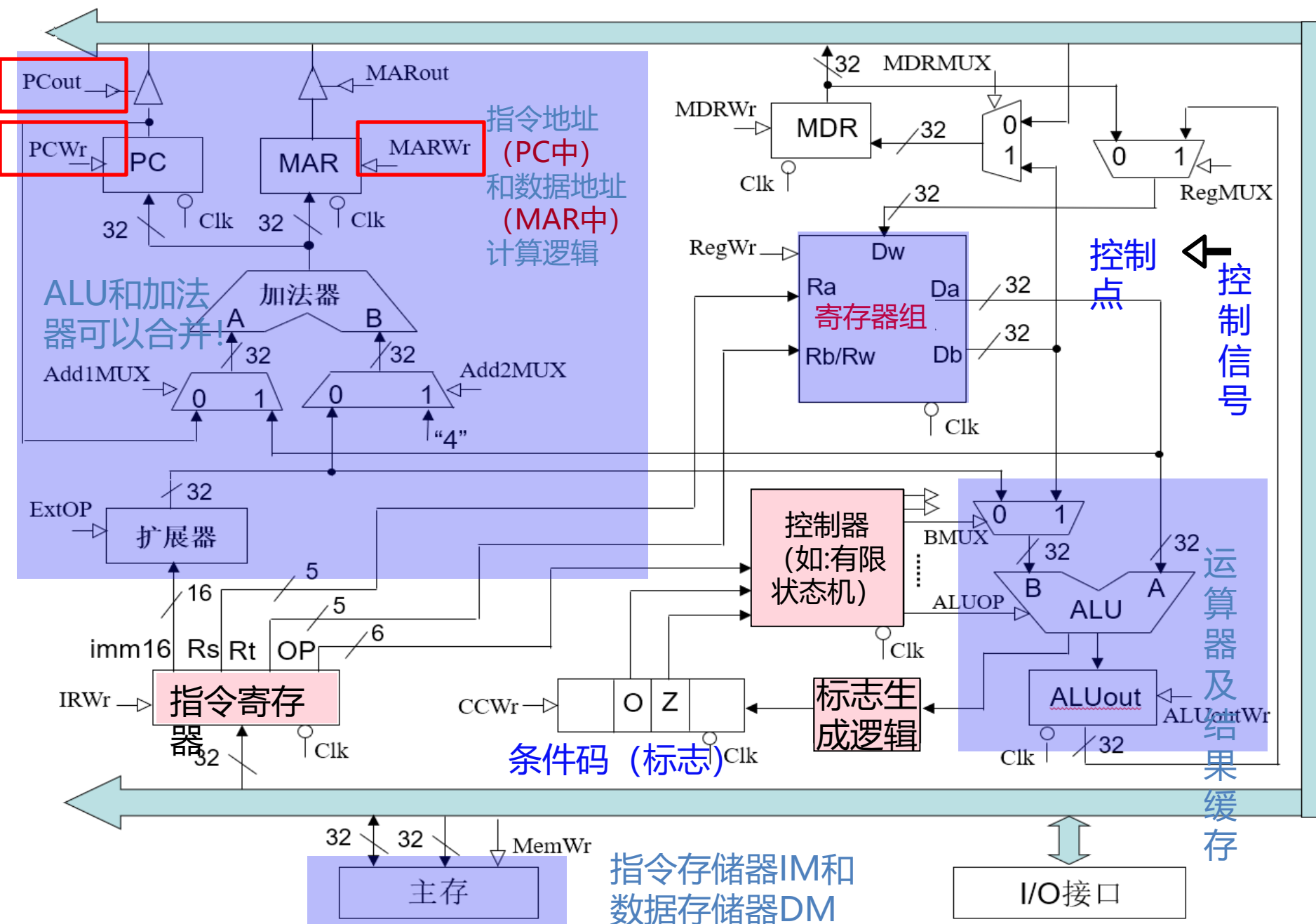
在单周期数据通路中，所有指令的执行都在一个时钟周期内完成，数据总是在时钟边沿被写入寄存器堆，即本条指令执行的结果总是下条指令开始（即下个时钟到来）时，才开始被写到通用寄存器组中，因此一个时钟周期只能写一次寄存器。而**swap**指令执行过程中需要多次写寄存器，在不对通用寄存器组进行修改的情况下，无法在单周期数据通路中实现**swap**指令；

对于多周期数据通路，一个指令周期可以有多个时钟周期，因而可以多次写寄存器，因此，在不对通用寄存器组进行修改的情况下，**swap**指令可以在多周期数据通路中实现。

9

- 假定多周期数据通路对应的有限状态机控制器发生错误，使得控制信号PCWr、MARWr、RegWr、BMUX、PCout中某一个在任何情况下总是为1，则该控制信号为1时哪些指令不能正确执行？要求分别对每个控制信号进行讨论。
- 若PCWr=1，则除了第一条指令，后续所有指令都不能正确执行，因为每个时钟到来都会更新PC（第一条指令还没有执行完，PC就更新为其后面相隔若干条指令的地址了，则取出的第二条指令就已经是错误的了）。
- 若MARWr=1，只要MARout一直是正确的，那么所有指令都能正确执行。
- 若RegWr=1，无法正确执行所有指令。因为每个时钟周期都可能会向不该写的寄存器里写入错误的数值，而且无法恢复。
- 若BMUX=1，则I-型运算类指令执行错误，因为扩展器的输出无法作为ALU输入端B的输入。
- 若PCout=1，则一旦执行到load或store指令（需要通过总线传输存储器地址），就会发生冲突（地址总线上同时接受到32位的pc和32位的数据地址）和不确定的错误（拿不到正确的32位的数据地址），导致load和store无法正确执行。

简单指令系统对应的多周期CPU



11

11.假定在一个5级流水线处理器中，各主要功能单元的操作时间为：存储单元-200ps；ALU和加法器-150ps；通用寄存器组的读口或写口-50ps。请问：

(1) 若执行阶段EX所用的ALU操作时间缩短20%，则能否加快流水线执行速度？如果能的话，能加快多少？如果不能的话，为什么？

不能。因为指令流水线的执行速度取决于最慢的功能部件所用时间，最慢的是存储器，只有缩短了存储器的操作时间才可能加快流水线速度。

(2) 若ALU操作时间增加20%，对流水线的性能有何影响？

ALU操作时间延长20%时，变为了180ps，比存储器所用时间200ps还小，因此，对流水线性能没有影响。

(3) 若ALU操作时间增加40%，对流水线的性能又有何影响？

ALU操作时间延长40%时，变为了210ps，比存储器所用时间200ps大，因此，在不考虑流水段寄存器延时的情况下，流水线的时钟周期从200ps变为210ps，流水线执行速度降低了 $(210-200)/200=5\%$ 。

13

假定最复杂的一条指令所用的组合逻辑分成6个部分，依次为A~F，其延迟分别为80ps、30ps、60ps、50ps、70ps、10ps。在这些组合逻辑块之间插入必要的流水段寄存器就可实现相应的指令流水线，寄存器延迟为20ps。理想情况下，以下各种方式所得到的时钟周期、指令吞吐率和指令执行时间各是多少？应该在哪里插入流水段寄存器？

(1) 插入一个流水段寄存器，得到一个两级流水线。

(1) 两级流水线的平衡点在C和D之间，其前面一个流水段的组合逻辑延时为 $80+30+60=170\text{ps}$ ，后面一个流水段的组合逻辑延时为 $50+70+10=130\text{ps}$ 。最长功能段延时为170ps，加上流水段寄存器延时20ps，因而时钟周期为190ps，理想情况下，指令吞吐率为每秒钟执行 $1/190\text{ps}=5.26\text{G}$ 条指令。每条指令在流水线中的执行时间为 $2\times 190=380\text{ps}$ 。

(2) 插入两个流水段寄存器，得到一个三级流水线。

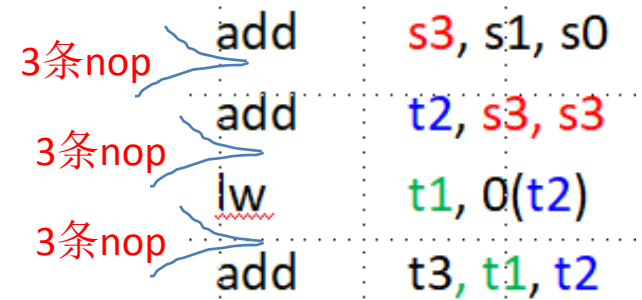
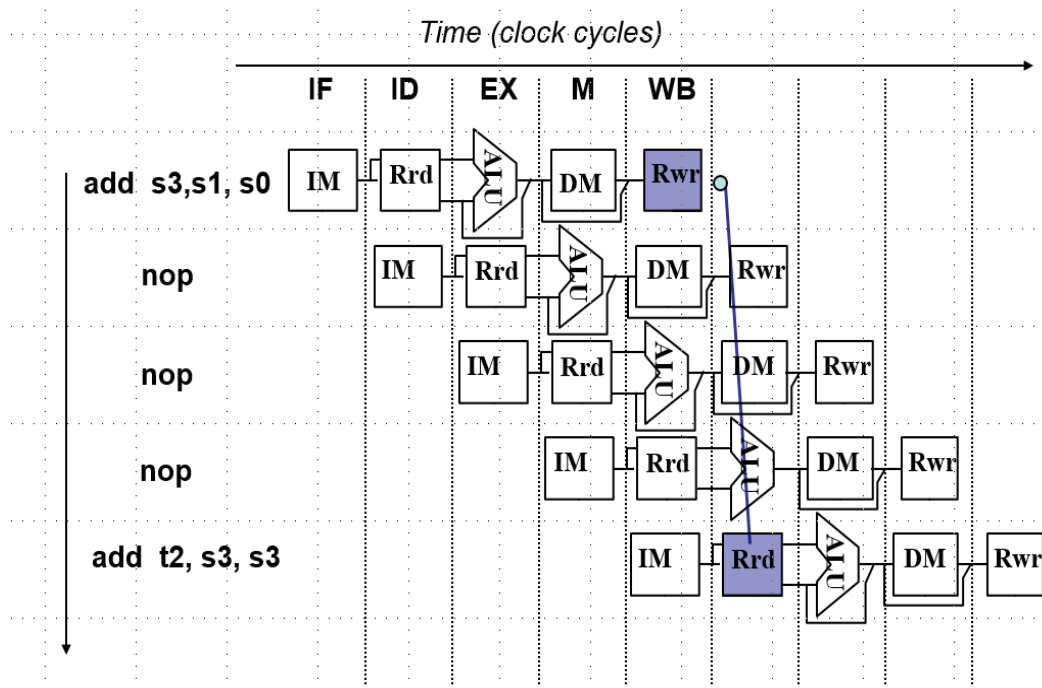
(2) 两个流水段寄存器分别插在B和C、D和E之间，这样第一个流水段的组合逻辑延时为 $80+30=110\text{ps}$ ，中间第二段的延时为 $60+50=110\text{ps}$ ，最后一个段延时为 $70+10=80\text{ps}$ 。这样，每个流水段所用时间都按最长延时调整为 $110+20=130\text{ps}$ ，故时钟周期为130ps，指令吞吐率为每秒钟执行 $1/130\text{ps}=7.69\text{G}$ 条指令，每条指令在流水线中的执行时间为 $3\times 130=390\text{ps}$ 。

- (3) 插入三个流水段寄存器，得到一个4级流水线。
- (3) 三个流水段寄存器分别插在A和B、C和D、D和E之间，这样第一个流水段的组合逻辑延时为80ps，第二段延时为30+60=90ps，第三段延时为50ps，最后一段延时为70+10=80ps。这样，每个流水段都以最长延时调整为90+20=110ps，故时钟周期为110ps，指令吞吐率为每秒钟执行 $1/110\text{ps}=9.09\text{G}$ 条指令，每条指令在流水线中的执行时间为 $4\times 110=440\text{ps}$ 。
- (4) 吞吐量最大的流水线。
- (4) 因为各功能部件对应的组合逻辑中最长延时为80ps，所以，流水线的时钟周期肯定比80ps+20ps=100ps长。为了达到最大吞吐率，时钟周期应该尽量短，因此，最合理的划分方案应该按照每个时钟周期为100ps来进行。根据每个功能部件所用时间可知，流水线至少按5段来划分，分别把流水线寄存器插入在A和B、B和C、C和D、D和E之间，这样各段的组合逻辑延时为80ps、30ps、60ps、50ps和80ps。其中，最后一个延时80ps是E和F两个阶段的时间相加而得到的。这样时钟周期为100ps，指令吞吐率为每秒钟执行 $1/100\text{ps}=10\text{G}$ 条指令，每个指令的执行时间为 $5\times 100=500\text{ps}$ 。
- 通过对上述4种情况进行分析，可以得出以下结论：划分的流水段多，时钟周期就变短，指令执行吞吐率就变高，而相应的额外开销（即插入的流水段寄存器的延时）也变大，使得一条指令的执行时间变长。

14

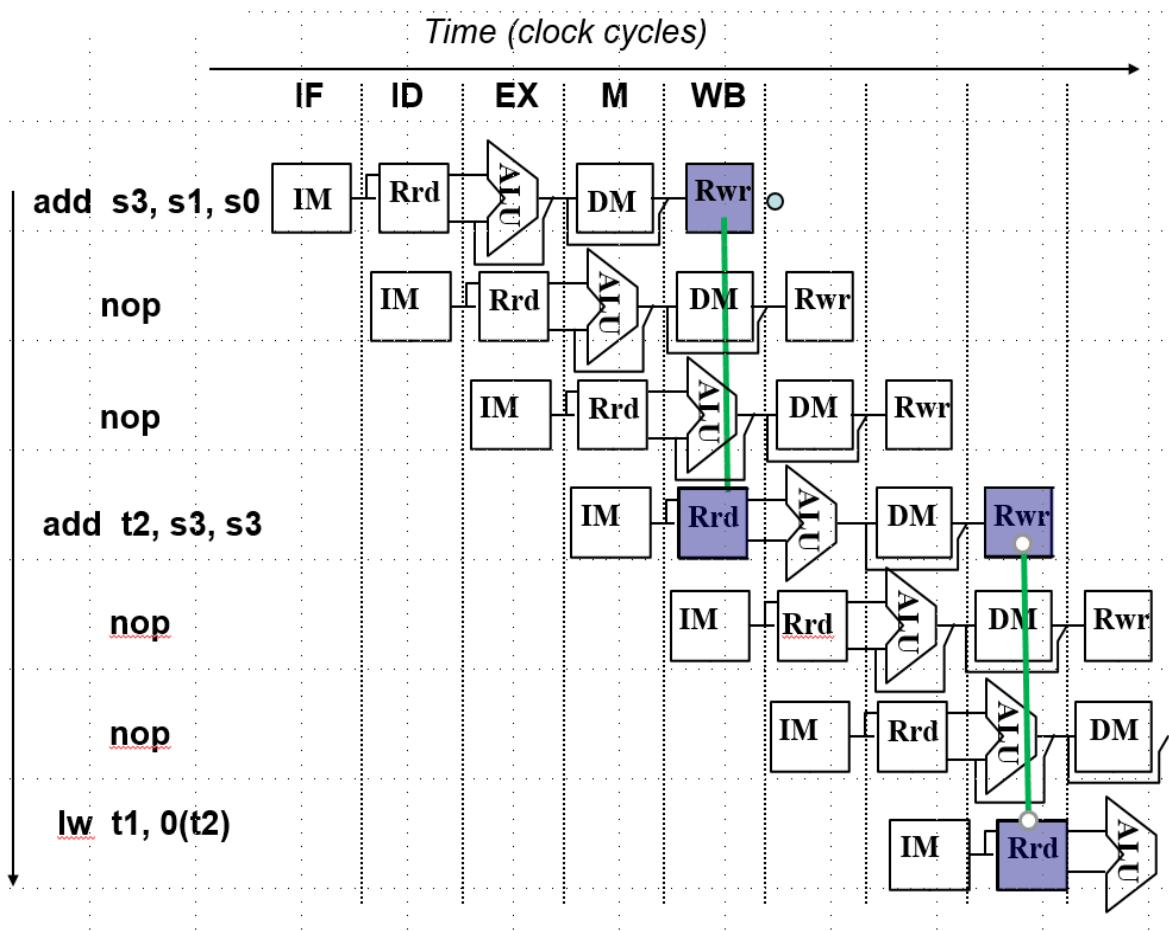
- 以下指令序列中，哪些指令对之间会发生数据相关？假定采用“取指、译码/取数、执行、访存、写回”5段流水线方式，
 - `add s3, s1, s0`
 - `add t2, s3, s3`
 - `lw t1, 0(t2)`
 - `add t3, t1, t2`
- 那么不用“转发”技术的话，需要在发生数据相关的指令前加入几条nop指令才能使这段程序避免数据冒险？
- 如果采用“转发”是否可以完全解决数据冒险？不行的话，需要在发生数据相关的指令前加入几条nop指令才能使这段RV32I程序不发生数据冒险？
- （问题如果改成“会阻塞几个时钟周期？” ， 答案如何？）

- 第1和第2条指令、第2和第3条指令、第2条和第4条指令、第3条和第4条指令之间发生数据相关。
- 若不采用“转发”技术，则为避免相邻两条指令之间的数据冒险，需要在它们之间插入3条nop指令。（或阻塞3个时钟周期）



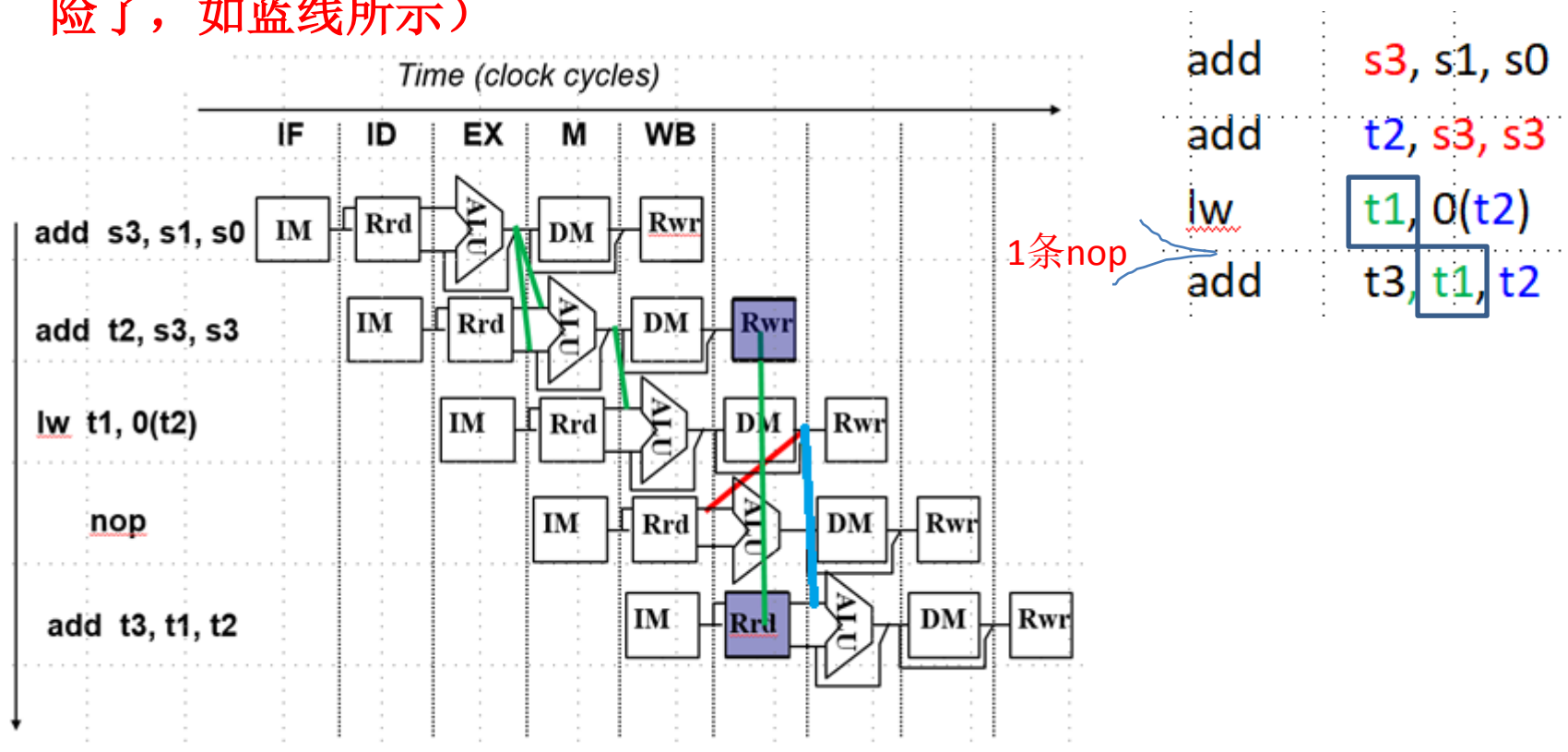
寄存器不采用前半周期写后半周期读。

如果寄存器采用前半周期写后半周期读。则在相邻两条数据相关的指令之间插入两条nop指令即可。（或阻塞2个时钟周期）



2条nop	add	s3, s1, s0
2条nop	add	t2, s3, s3
2条nop	lw	t1, 0(t2)
2条nop	add	t3, t1, t2


- 可以采用“转发”技术解决第1和第2条指令、第2和第3条指令、第2和第4条指令之间的数据冒险（下图中的绿线）；但是，因为第3和第4条指令之间是Load-use数据冒险，因而不能通过转发技术解决（参见下图中的红线），需要在第3条指令后加一条nop指令（加入nop或阻塞1个时钟周期之后，lw和最后一个add之间就可以通过转发解决冒险了，如蓝线所示）



17

- 假定在一个带“转发”逻辑的5段流水线中执行以下RV32I程序段，应怎样调整指令序列使其性能达到最好？

```
1  lw    s2, 100(s6)
2  add   s2, s2, s3
3  lw    s3, 200(s7)
4  add   s6, s4, s7
5  sub   s3, s4, s6
6  lw    s2, 300(s8)
7  beq   s2, s8, Loop
```



因为采用“转发”技术，所以，只要针对load-use数据冒险进行指令序列调整。

第1和第2条指令、第6和第7条指令之间存在load-use数据冒险，所以，可将与第2和第3条指令无关的第4条指令插入第2条指令之前；将与第6和第7条无关的第5条指令移动到第6条指令之后。调整顺序后的指令序列如下

```
1          lw    s2, 100(s6)
4          add   s6, s4, s7
2          add   s2, s2, s3
3          lw    s3, 200(s7)
6          lw    s2, 300(s8)
5          sub   s3, s4, s6
7          beq   s2, s8, Loop
```


18

- 在一个采用“取指、译码/取数、执行、访存、写回”的5段流水线中，若检测相减结果是否为“零”的操作在执行阶段进行，则分支延迟损失时间片（即分支延迟槽）为多少？
- 以下一段RV32I指令序列中，在考虑数据转发的情况下，哪些指令执行时会发生流水线阻塞？各需要阻塞几个时钟周期？

```
1  loop: add t1, s3, s3
2          add t1, t1, t1
3          add t1, t1, s6
4          lw  t0, 0(t1)
5          bne t0, s5, Exit
6          add s3, s3, s4
7          j   loop
```

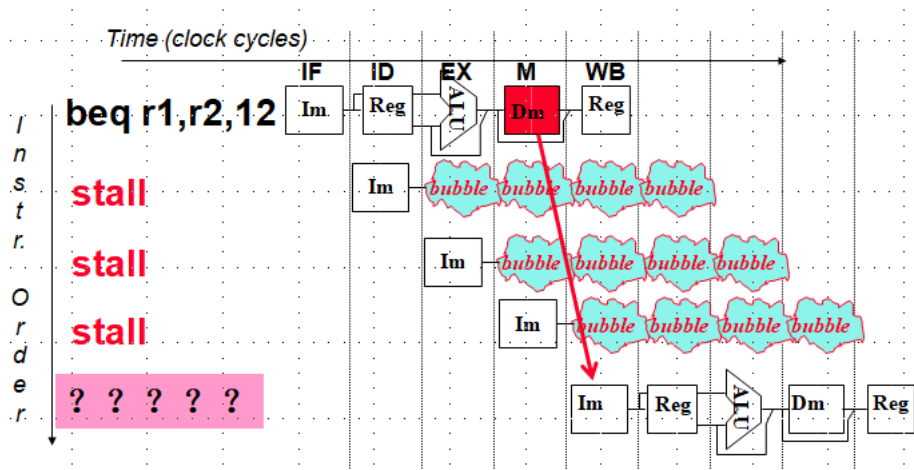
Exit:

（1）在采用数据转发技术的情况下，第4条和第5条指令之间为Load-use冒险，无法通过转发技术解决，此时第5条bne指令的执行被阻塞一个时钟。

（2）第5条指令是分支指令bne，由于该指令执行时条件检测结果不能马上得到，因此其后add指令的执行被阻塞，阻塞的时钟周期数等于分支延迟损失时间片（2）。

（如果采用静态分支预测，总是预测not taken，且预测成功，则阻塞几个时钟？——预测成功则无需阻塞）

（3）第7条指令为无条件跳转指令jal（j loop是其伪指令），假定“j loop”指令更新PC的操作在“执行（EX）”阶段进行，则其后指令的执行也将被阻塞2个时钟周期；假定更新PC的操作在“译码（ID）”阶段进行，则只需要阻塞1个时钟周期。



回忆课堂内容——若检测结果是否为“零”并更新PC的操作在“Mem”阶段进行，则分支延迟损失时间片（分支延迟槽）为3。

本题——若检测结果是否为“零”并更新PC的操作提前到在“执行（EX）”阶段进行，则分支延迟损失时间片（分支延迟槽）变为2。

