解决冒险的流水线实现

译码器的实现

因为整个数据冒险的处理都可以用 AT 法概括描述，这要求我们在译码的时候，不能只译码出指令信息，还需要译码出指令相关的 AT 信息。只有译码出了 AT 信息，才可以帮助我们进行流水线的决策。

当我们采用集中式译码的时候，AT 信息只在 D 级被译码了一次，但是同一个指令的 AT 值在不同的流水线阶段可能会发生变化，所以这就要求我们应当在流水线寄存器中完成流水级间的变化，比如某种实现的 $M\_T_{new} = max(E\_T_{new} - 1, 0)$ 。也可以不在流水线寄存器中完成，而是开辟一个新的功能单元完成。

对于分布式译码，因为 AT 信息在每个流水线都被译码，所以就不存在传递变化的问题，但是译码器就必须知道自己所在的流水级，才能给出对应的正确的 AT 值。

阻塞的实现

为了方便处理，**课程组要求阻塞是指将指令阻塞在 D 级。**

当一个指令到达 D 级后，我们需要将它的 T_{use} 值与后面每一级的 T_{new} 进行比较（当然还有 A 值的校验），当 $T_{use} < T_{new}$ 时，我们需要阻塞流水线。

阻塞的实现需要改造流水线寄存器和 PC ，我们需要让它们具有以下功能：

- 冻结 PC 的值；
- 冻结 D 级流水线寄存器的值；
- 将 E 级流水线寄存器清零（这等价于插入了一个 nop 指令）。

此外，还有一个考虑，就是复位信号和阻塞信号的优先级问题。请仔细设计信号的优先级来保证流水线的正确性。

内部转发的实现

GPR 是一个特殊的部件，它既可以视为 D 级的一个部件，**也可以视为 W 级之后的流水线寄存器。**基于这一特性，我们将对 GPR 采用**内部转发**机制。也就是说，当前 GPR 被写入的值会即时反馈到读取端上。

具体的说，当读寄存器时的地址与同周期写寄存器的地址相同时，我们将读取的内容改为写寄存器的内容，而不是该地址可以索引到的寄存器文件中的值。

转发的实现

当一个指令到达 D 级后，我们需要将它的 T_{use} 值与后面每一级的 T_{new} 进行比较（当然还有 A 值的校验），当 $T_{use} \geq T_{new}$ 时，我们需要进行转发。

为了实现转发，我们需要两种多路选择器 MUX，分别对应转发的供给者和需求者。

转发的供给者其实不需要考虑转发的需求者是谁，因为对于一条指令，他能提供的数据至多一种，他如果不是写指令，就不会提供数据，如果他是写指令，也要区分需要写的数据是否产生。如果是写指令，就通过一个 MUX 从流水线寄存器的输出里选择结果，比如 add 就会选择 ALUOut，lw 就会选择 DMOut ，此时衍生了一个问题，就是万一没有怎么办？比如在 EREG 中，就没有 DMOut ，这时就需要单独的设计了。这是第一种MUX，输入是流水线寄存器的输出，输出是当前指令对应的写数据。

转发的需求者应该是流水级的某个数据，而不是某个模块。例如在 D 级时，我们需要读出的数据就是 rs 对应的寄存器数据（rsOut）和 rt 对应的寄存器数据（rtOut），然后才是利用这些数据进行各种处理。所以我们转发的目的不应该是仅仅为模块提供正确数据，而应是把 rsOut 和 rtOut 换成正确的，经过转发后的数据 FW_rsOut ，FW_rtOut，这样就是两个 MUX。这是第二种 MUX，输入是各级的第一种 MUX 的输出，输出是当前正确（或者可以容忍的错误）的读数据。

控制器的实现

我们解决冒险需要进行 AT 值的比较判断，并需要根据判断的结果产生特定的控制信息。这些功能要求我们丰富我们的控制器，使其可以支持这些功能。

我们的控制器需要产生的信号包括但不限于**冻结信号，刷新信号，供给者选择器信号，需求者选择器信号**等。

实例

这里我们以 {add, lw, beq} 这三条指令为例来具体介绍 AT 法如何解决流水线数据冒险。

Tuse 和 Tnew 的分析

T_{use} 表示数据到了 D 级之后还需要多少个周期要使用，每个指令的 T_{use} 是固定不变的。

让我们先来分析这些指令的 T_{use}：

- 对于 add 指令，在 E 级使用 GPR[rs] 和 GPR[rt] 的值，因此 $rs\_T_{use} = rt\_T_{use} = 1$
- 对于 lw 指令，在 E 级使用 GPR[rs] 的值，因此 $rs\_T_{use} = 1$
- 对于 beq 指令，在 D 级使用 GPR[rs] 和 GPR[rt] 的值，因此 $rs\_T_{use} = rt\_T_{use} = 0$

T_{new} 表示数据还有多长时间产生，会随着数据的流水动态的减少。

让我们来分析这些指令在 E 级的 T_{new}：

- 对于 add 指令，结果在 E 级被计算出来，在 E 级还需要 1 个时钟周期才能将结果放到流水寄存器中，因此 $E\_T_{new} = 1$
- 对于 lw 指令，结果在 M 级从 DM 中取出，在 E 级时还需要 2 个时钟周期才能将结果放到流水寄存器中，因此 $E\_T_{new} = 2$
- 对于 beq 指令，不产生新数据，我们认为$E_T_{new} = 0$

当数据从一个流水级到下一个流水级时，在流水寄存器中需要更新 T_{new}，递推公式为 $T'_{new} = max\{T_{new} - 1, 0\}$，**注意 T_{new} 不会小于零**，据此我们可列出这些指令在不同流水级的 T_{new}：

指令	E_T_{new}	M_T_{new}	W_T_{new}
add	1	0	0
lw	2	1	0
beq	0	0	0

转发和暂停的分析

我们知道，当 $T_{use} \geq T_{new}$ 时，可以通过转发解决；当 $T_{use} < T_{new}$ 必须阻塞流水线。

根据上述的 T_{use} 和 T_{new} 的值，我们做出策略矩阵，其中 F 表示转发，S 表示暂停：

T_{use} \ T_{new}	E_T_{new}			M_T_{new}		W_T_{new}
	2	1	0	1	0	0
0	S	S	F	S	F	F
1	S	F	F	F	F	F

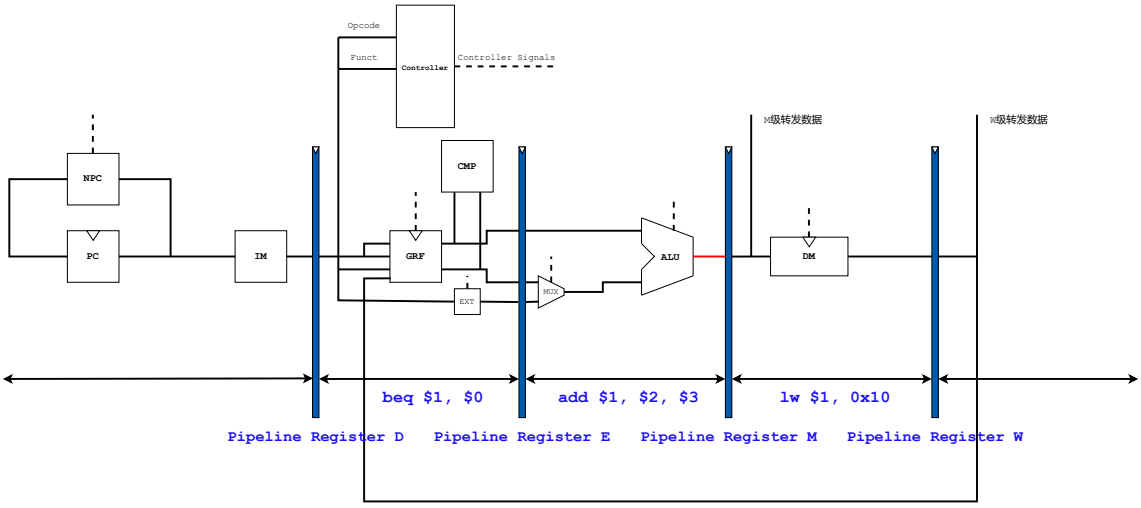
根据上表，可以看出只有四种情况需要阻塞：

- $E\_T_{new} = 2, T_{use} = 0$
- $E\_T_{new} = 1, T_{use} = 0$
- $M\_T_{new} = 1, T_{use} = 0$
- $E\_T_{new} = 2, T_{use} = 1$

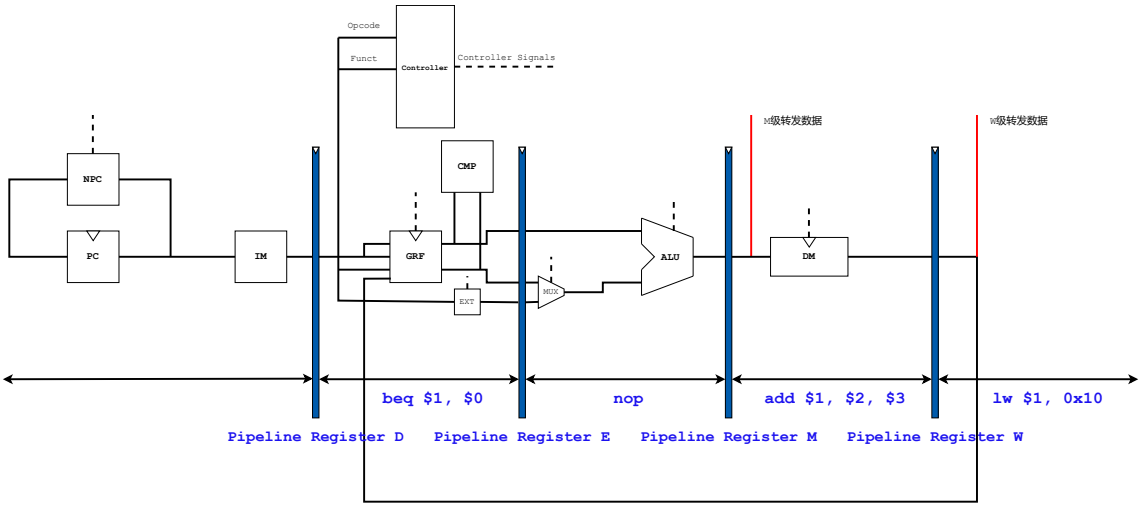
除了这四种，剩下的情况就是需要转发的了。在转发中，我们需要特别注意转发的优先级问题和 rt 域有效性问题。

转发优先级问题是当多个数据可以转发，选择哪个数据的问题，如下面这个指令序列：

```
lw      $1, 0x10
add     $1, $2, $3
beq     $1, $0
```



从上图可以看出，当 beq 在 D 级时，CMP 模块所需的 $1 寄存器的值在 add 指令中产生了一个新值（即上图中红线），并且这个新值无法通过转发的方式转发到 D 级，因此，我们此时不得不先暂停一个周期。



暂停一个周期后，我们发现 lw 和 add 两条指令产生了 $1 寄存器的两个新值（即上图中的两条红线）且两个数据都可以转发到 D 级。那么此时我们应当选用哪一级的转发数据转发到 D 级呢？但显然，我们应该用 M 级，也就是 add 产生的数据，因为他会覆盖掉前面 lw 产生的数据。也就是说，我们要选择流水线中靠前的“新鲜”的数据进行转发。

rt 域有效性问题是指有些指令的 rt 域不是用来表示读寄存器编号的，比如 j 指令没有 rt 域、ori 指令的 rt 域表示写入寄存器的编号，那么我们是否需要对这些指令进行特判呢？答案是不需要。对于 rt 域无效的指令，即使我们转发了相应的数据，也不会影响流水线的正确性，因此无需特判。

具体的实现

可能有些同学看了上面的讲述，在设计上仍有些疑惑，这里我们给出一个具体实现的例子，供有需要的同学参考。同学们完全可以有不一样的实现方式，满足提交要求即可，我们也鼓励大家给出更优秀的实现方式。

为解决流水线数据冒险，我们可以单独设计一个冒险控制模块，输入用于判断暂停和转发的 A 和 T 信号，输出暂停和转发的控制信号，下面让我们一起来分析该模块内部的逻辑。

对于暂停，我们可以根据 T_{use} 和 T_{new} 值的不同组合构造出 4 种暂停信号。当然除了上述 T 的条件，暂停时还需要满足 A 的条件（即读寄存器和写寄存器编号相同且不为 0、写寄存器信号有效）。以 rs 寄存器为例，我们可以写出暂停条件：

```
// 这里的 Stall_RS0_E2 对应 rs_Tuse = 0, E_Tnew = 2 的情况
// Tuse_RS0 表示 rs_Tuse == 0
// A1 表示指令中 [25:21] 域，A3 表示指令中 [15:11] 域
Stall_RS0_E2 = Tuse_RS0 & (Tnew_E == 2'b10) & (A1_D == A3_E) & (A1_D != 5'd0) &
RegWrite_E;
Stall_RS0_E1 = ......
......

Stall_RS = Stall_RS0_E2 | Stall_RS0_E1 | ......
```

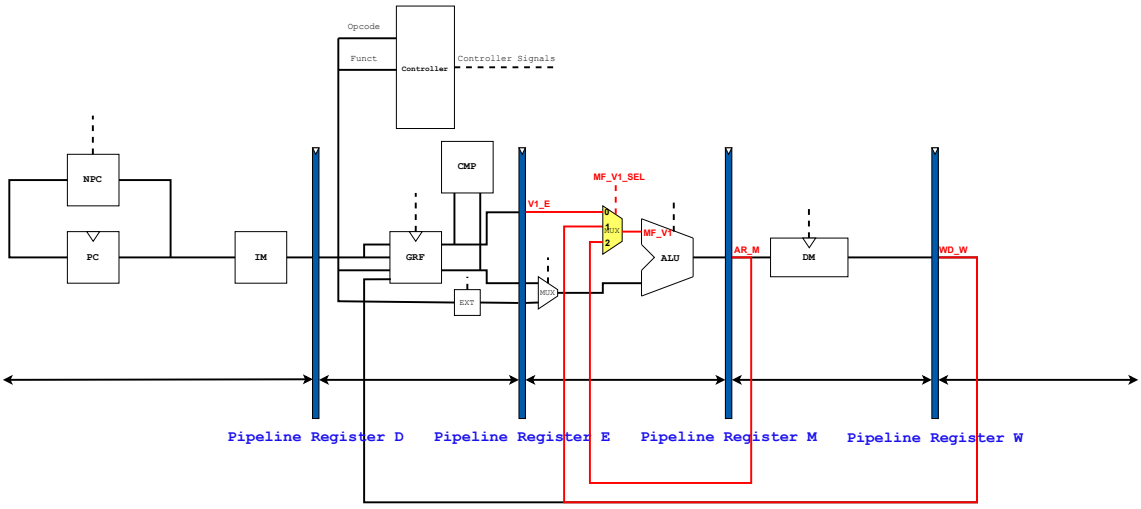
rt 寄存器同理，最后总的暂停信号把两个寄存器分别的暂停信号或起来即可。

简单起见，我们约定暂停只发生在 D 级，因此当暂停信号有效时，我们需要保持 D 级流水寄存器，清空 E 级流水寄存器。

对于转发，我们首先要在需要转发的点位放一个多路选择器，可以让 0 路对应原始数据，剩下的路按照数据优先级从低到高排列，然后利用一个转发控制信号选择正确的值。转发控制信号在冒险控制模块内生成，具体的判断条件是：读寄存器和写寄存器编号相同且不为 0、写寄存器信号有效（A 条件）以及转发流水级 $T_{new} = 0$（T 条件，表示此时数据已经准备好了）。以 E 级 ALU 中对应 GRF[rs] 的输入端为例：

```
// AR_M 表示 M 级流水寄存器中储存着的 ALU 结果，即 M 级转发数据
// WD_W 表示 W 级流水寄存器中将要写回到寄存器堆的值，即 W 级转发数据
MF_V1 = (MF_V1_SEL == 2'b10) ? AR_M :
        (MF_V1_SEL == 2'b01) ? WD_W : V1_E;

MF_V1_SEL = (A1_E == A3_M && A1_E != 5'd0 && Tnew_M == 2'b00 && RegWrite_M == 1'b1) ?
2'b10 :
            (A1_E == A3_W && A1_E != 5'd0 && Tnew_W == 2'b00 && RegWrite_W == 1'b1) ?
2'b01 : 2'b00;
```



至此，我们就基本解决了流水线数据冒险的核心问题，剩下的部分就留给聪明的你了，相信你一定能够顺利解决！

思考题

我们为什么要使用 GPR 内部转发？该如何实现？

思考题

我们转发时数据的需求者和供给者可能来源于哪些位置？共有哪些转发数据通路？