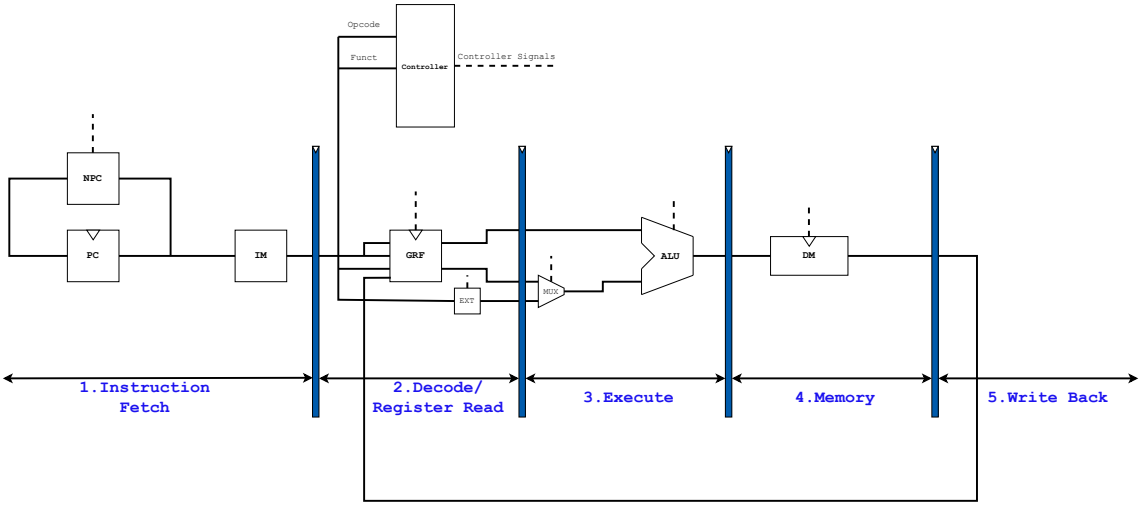
数据通路

流水线阶段

流水线技术是提高 CPU 吞吐量的有效手段。通过将单周期处理器分解成 5 个流水线阶段来构成流水线处理器。5 级流水的 CPU 在每个流水阶段都可以执行 1 条指令，那么同时就有 5 条指令并行，这大大提高了 CPU 的吞吐量。

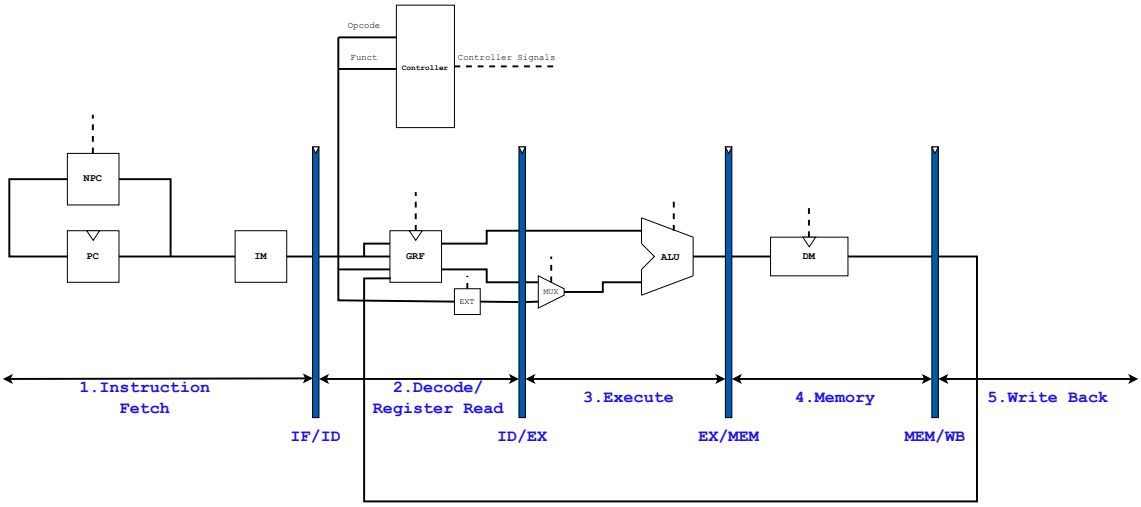
我们 CPU 为的 5 级流水线阶段为：

阶段	简称	功能概述
取指阶段（Fetch）	F	从指令存储器中读取指令
译码阶段（Decode）	D	从寄存器文件中读取源操作数并对指令译码以便得到控制信号
执行阶段（Execute）	E	使用 ALU 执行计算
存储阶段（Memory）	M	读或写数据存储器
写回阶段（Writeback）	W	将结果写回到寄存器文件



流水线寄存器

我们通过**流水线寄存器**将单周期处理器的数据通路划分为 5 个阶段。我们通过在两个阶段之间加入寄存器，来保存在前一周期中的上一阶段所传来的数据，同时在后一周期中为下一阶段提供数据。无论是数据还是信号，都需要在寄存器中进行保存，直至不再需要。

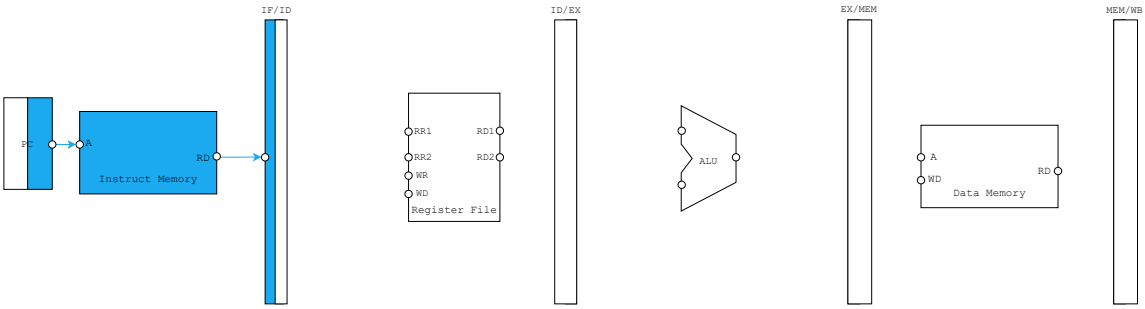


我们流水线寄存器以其提供数据的流水级的简称命名，如 D 级流水线寄存器的前一级为 F 级，而后一级为 D 级。

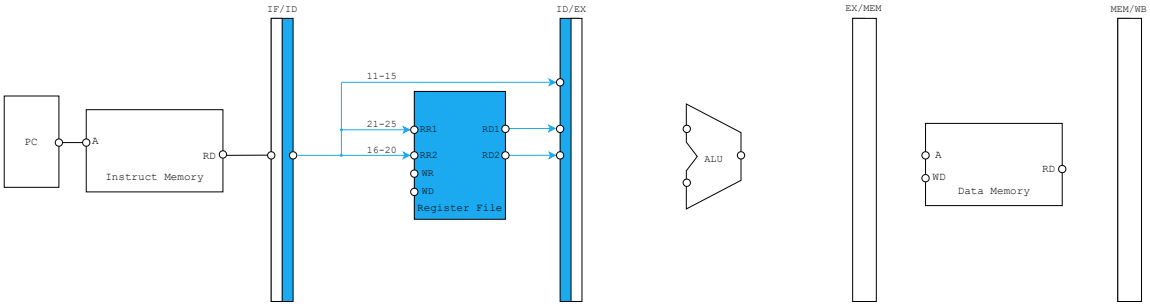
下面我们对 P5 中涉及的指令进行分析，以此讲解流水线寄存器的工作原理和需要保存哪些数据：

R 型算术运算指令：add、sub

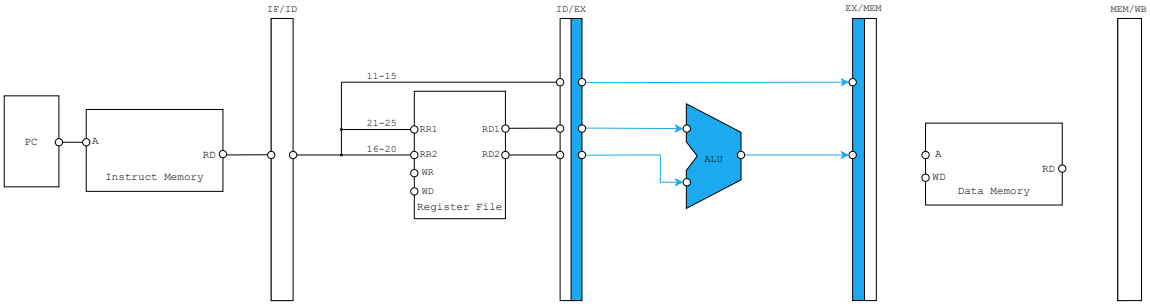
Fetch 阶段：从 PC 寄存器取地址，根据地址从 IM 获得指令编码，在下一周期存入 D 级寄存器。



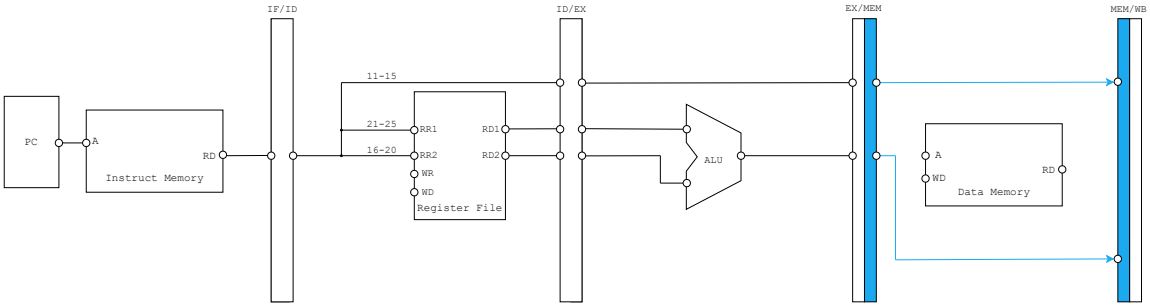
Decode 阶段：从 D 级寄存器取出指令，根据相应寄存器 rs、rt 地址获取其中数据，与 rd 地址一起在下一周期存入 E 级寄存器中。



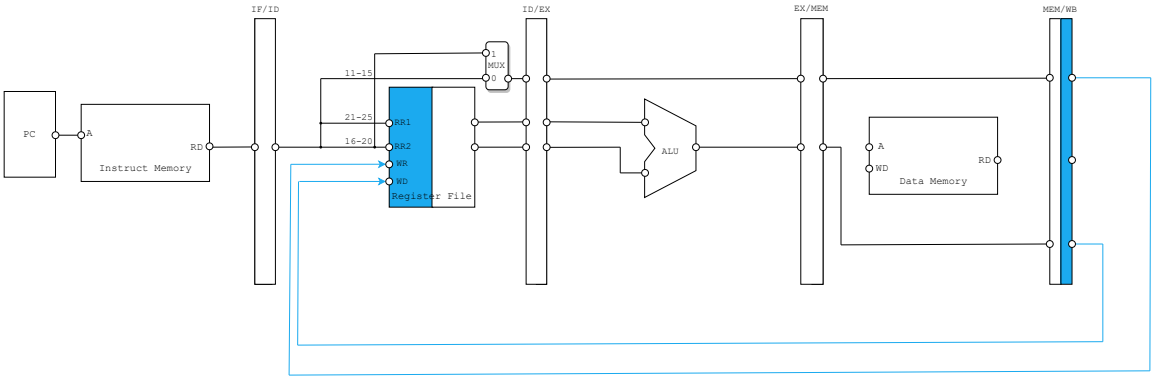
Execute 阶段：从 E 级寄存器获得上一阶段从 RF 中取得的值，经过 ALU 计算，下一周期将存入 M 级寄存器中，rd 地址继续流水。



Memory 阶段：rd 地址与 ALU 结果继续流水。



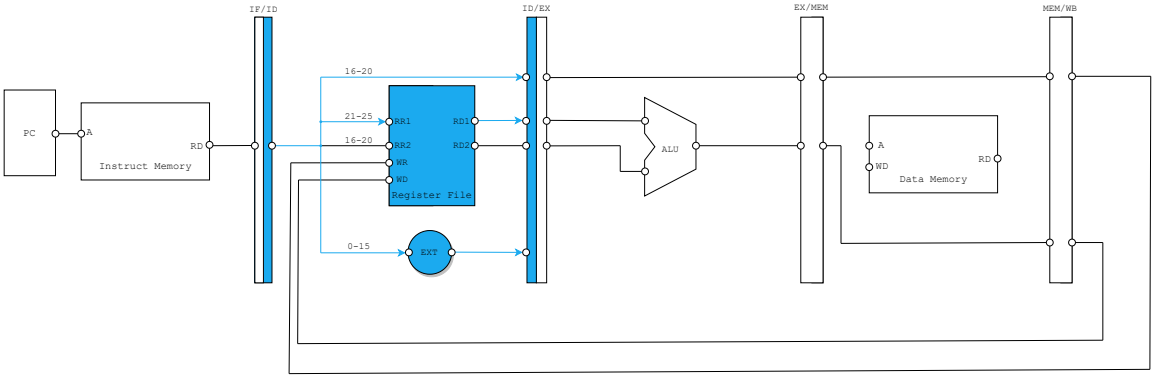
Writeback 阶段：W 级寄存器为 RF 提供 rd 地址与 WD，将相应数据写入 rd 中。



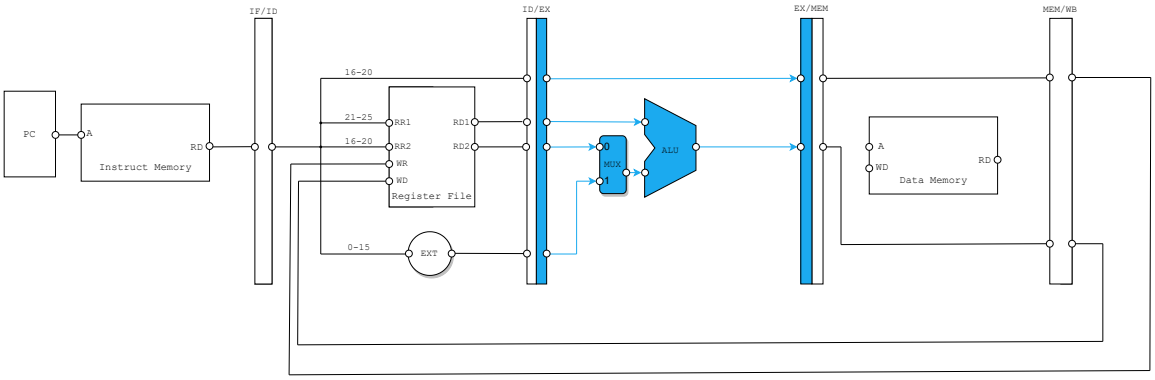
带有立即数的 I 型算数、逻辑运算指令：ori、lui

这类指令与之前的 R 型算数、逻辑运算指令大致相似，不同之处在于 D 和 E 阶段的处理。

Decode 阶段：将指令低 16 位取出经过 EXT 存入 E 级寄存器。



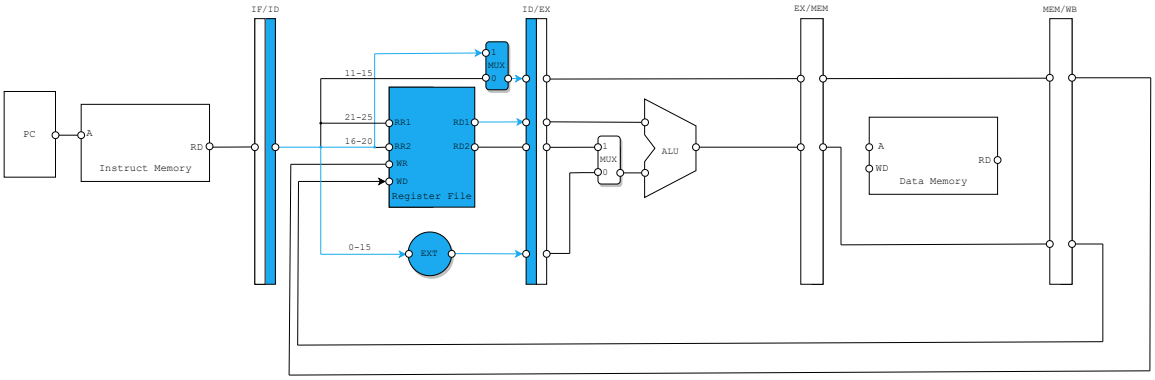
Execute 阶段：E 级寄存器提供 EXT 的结果，经过多路选择器提供给 ALU 的 B 端。



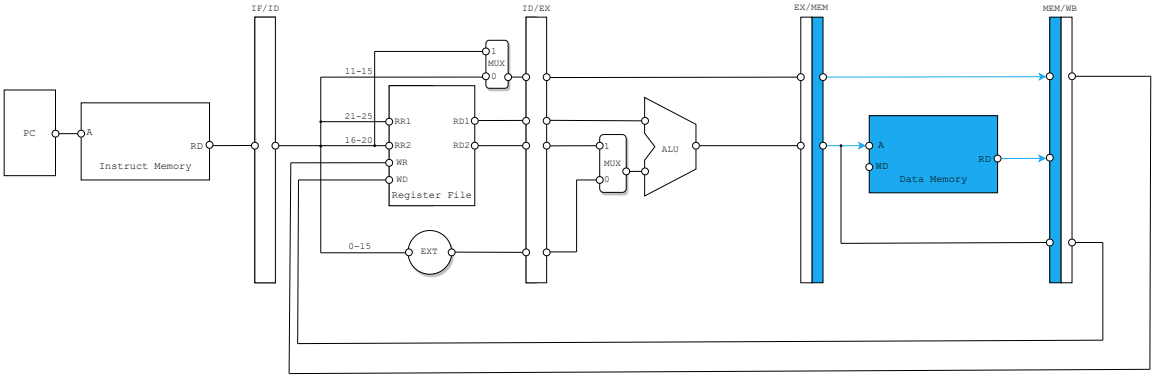
内存访问指令：lw

访存时写入寄存器变为 rt，需要在 ID 级增加一个 MUX，同时，在 M 级需要 ALU 的输出来指定 DM 的地址获得数据，这个数据随着流水线传到 W 阶段写入 RF，因此在 W 阶段需要增设一个 MUX。

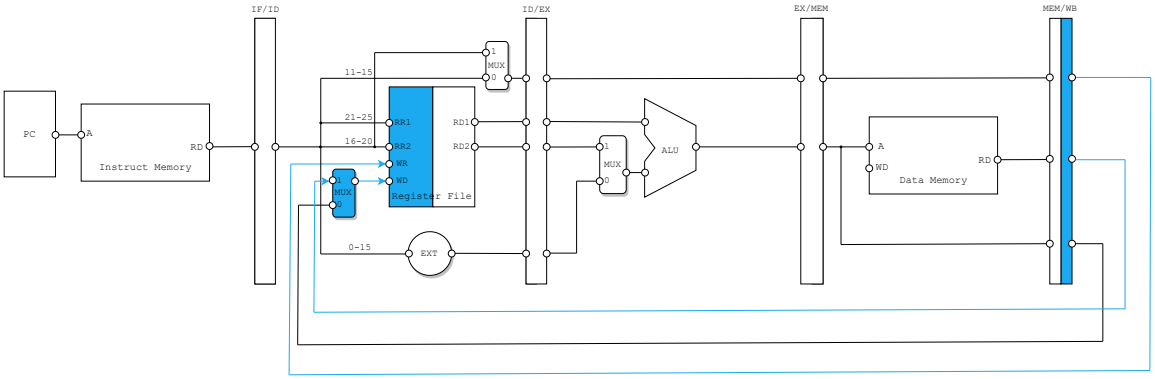
Execute 阶段：将 rt 作为目的寄存器存入 ID/EX 寄存器。



Memory 阶段：根据 ALU 计算出的地址仿存，并将数据存入 MEM/WB 寄存器。



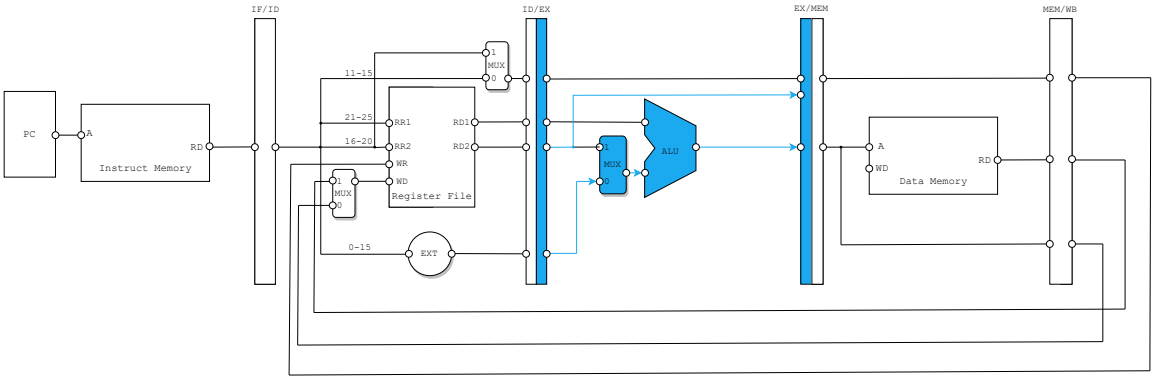
Writeback 阶段：通过多选器将数据写入 rt 寄存器。



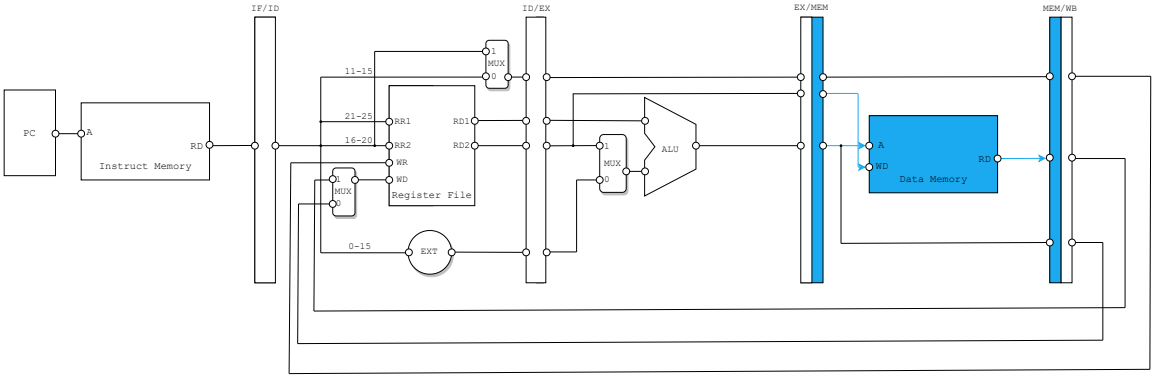
内存写入指令：sw

内存的写入在 M 级需要增加为 DM 提供的写入数据，并且在 W 阶段没有实质性的数据通路，即无需写回到 RF。

Execute 阶段：通过 rs 寄存器的数据与立即数计算地址。

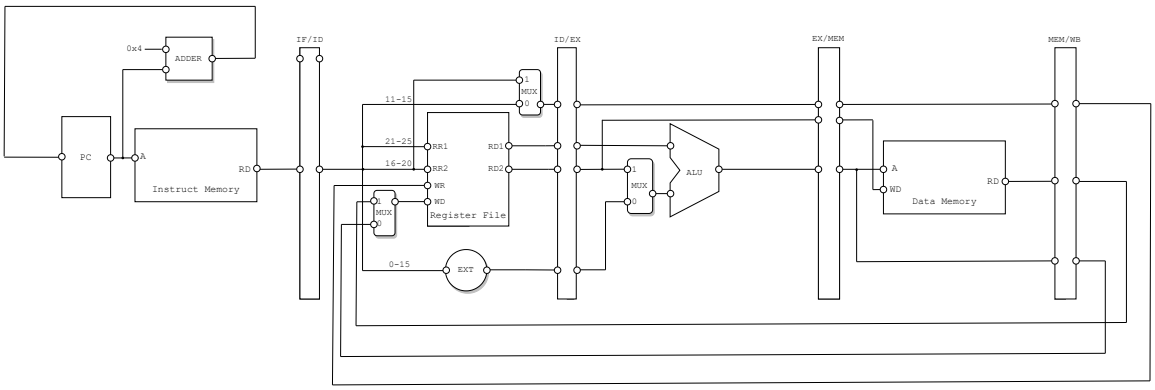


Memory 阶段：将 rt 寄存器的数据写入 ALU 计算出的地址。



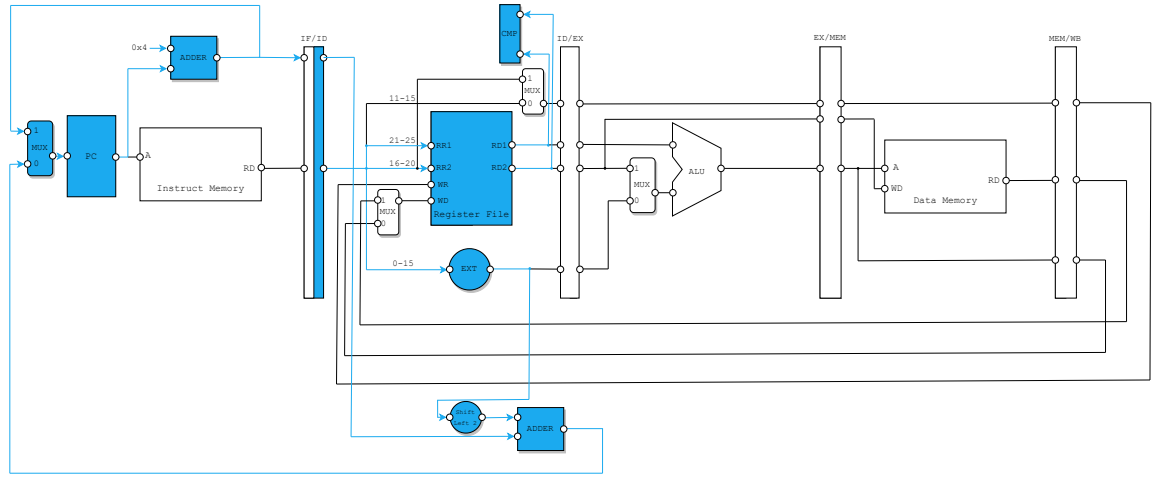
跳转 B 型指令：beq

在分析跳转指令开始之前，我们首先需要考虑 PC 寄存器，由于指令是顺序执行的，因此我们需要不断为 PC 寄存器加 4。



完善 PC 寄存器的实现后，我们开始正式分析 beq 这个指令，beq 需要在取出 rs、rt 寄存器的值后进行比较，然后决定是否跳转到新的 PC 值，新的 PC 的值可以通过 ID 级有符号扩展后，左移两位，再与 PC + 4 相加得到。

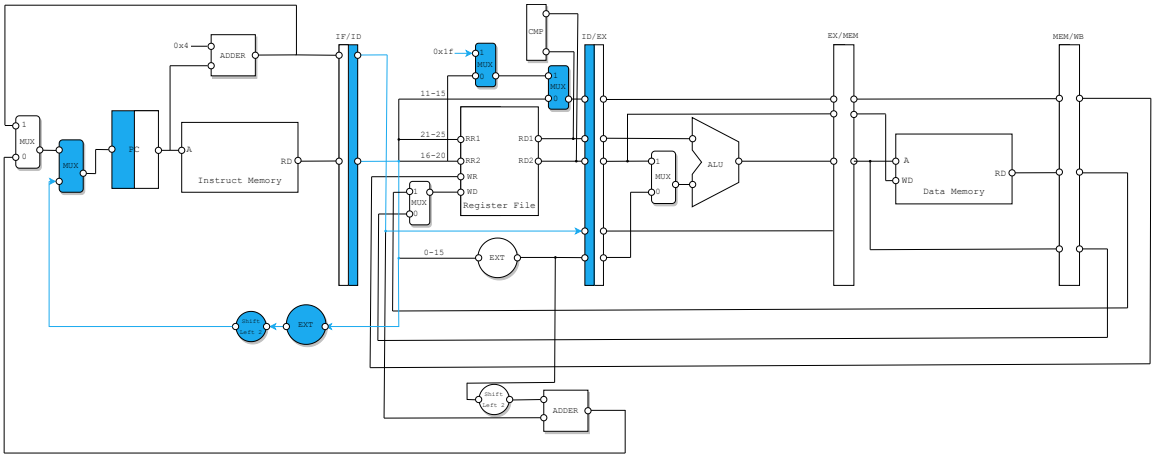
Decode 阶段：比较读取出的 rs 和 rt 寄存器的数据，若相等，则通过多选器将拓展后的立即数作为下一周期的 PC 写入 PC 寄存器。



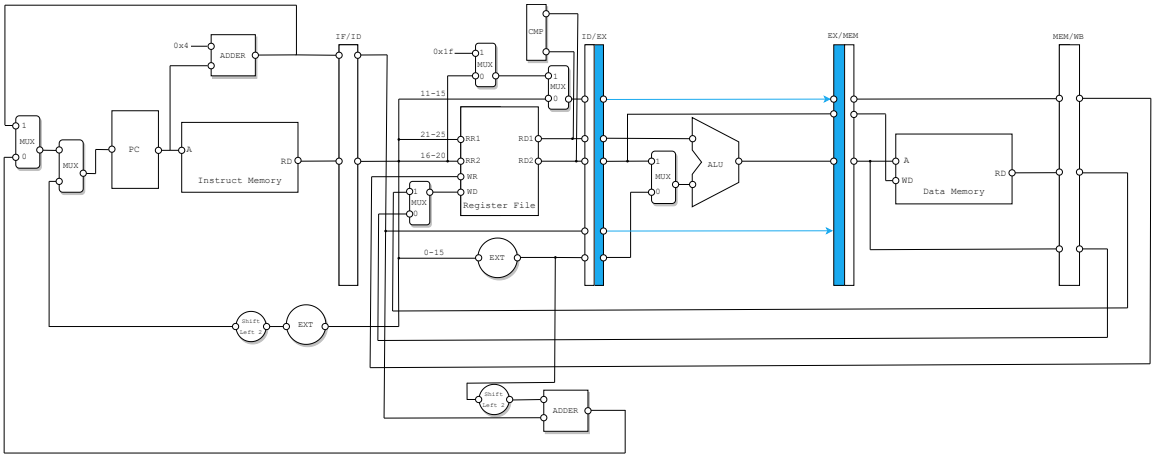
跳转 J 型指令：jal

J 型跳转指令同样无条件的可以在 D 级就修改 PC 寄存器的值，但需要在 ID 级增加一个扩展 26 位立即数的部件，同时，jal 需要写入 ra 寄存器，PC + 8 的值需要流水至 W 级写入 RF。

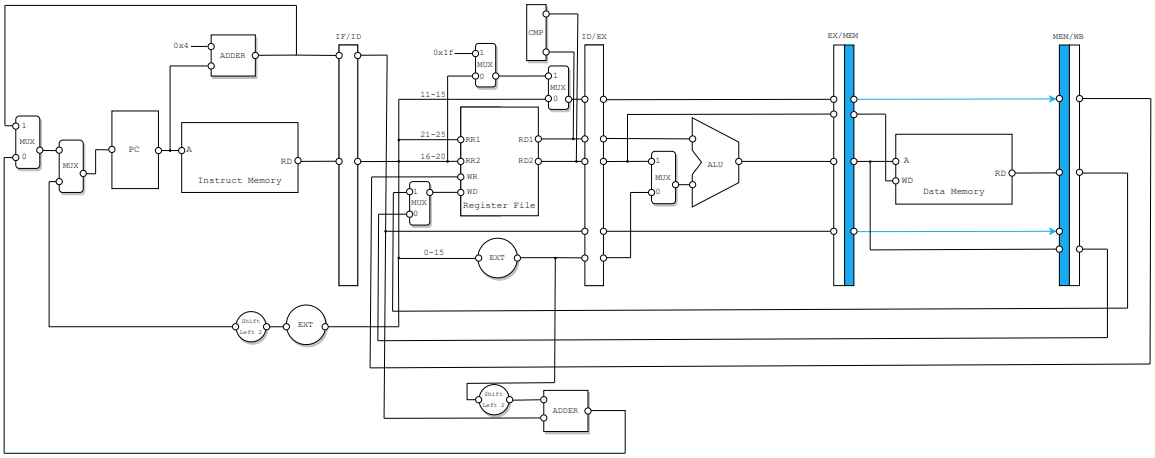
Decode 阶段：通过多选器将拓展后的立即数作为下一周期的 PC 写入。



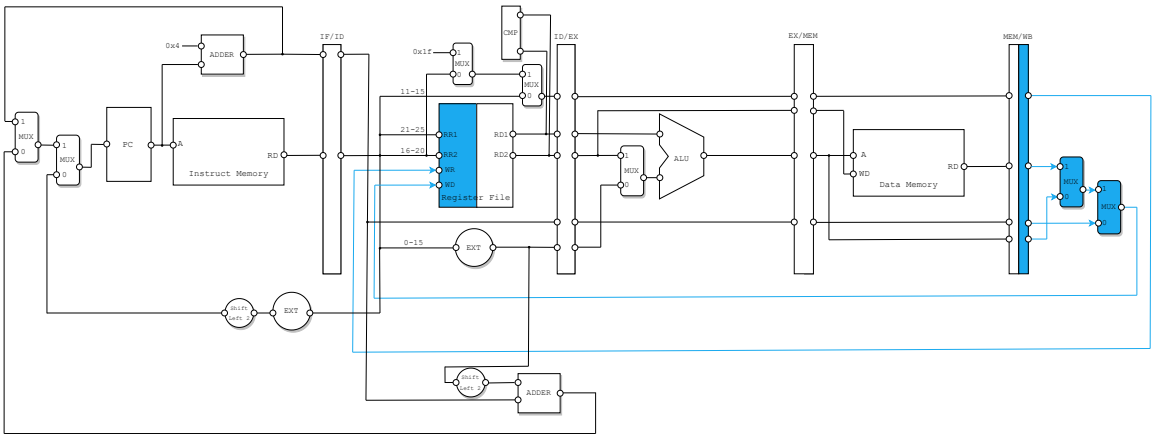
Execute 阶段：将跳转前的 PC 存入 EX/MEM 寄存器。



Memory 阶段：将跳转前的 PC 存入 MEM/WB 寄存器。



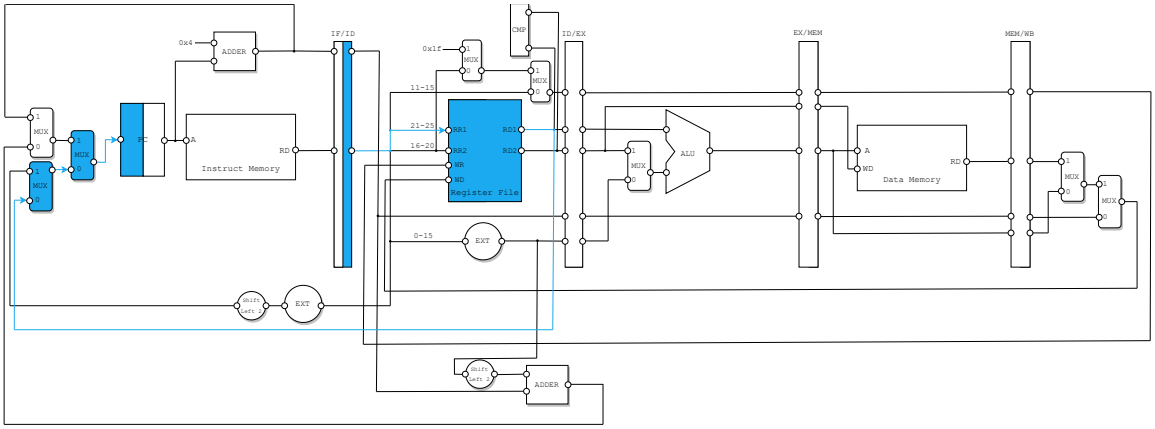
Writeback 阶段：将跳转前的 PC + 8 写入 31 号 ra 寄存器。



跳转 R 型指令：jr

R 型跳转指令将 rs 写入 PC 寄存器，可以在 D 级就修改 PC 寄存器，因此只需在 D 级增加一个 MUX。

Decode 阶段：通过多选器将 rs 寄存器的数据作为下一周期的 PC 写入。



流水数据的选择

一般而言，我们需要流水的数据只有一个衡量标准，就是我们在其后的流水阶段中需不需要这个数据，比如说 ALU 的计算结果，有的会被写回寄存器文件中，所以我们需要流水这个数据，而 rs 对应的寄存器值，在 M 级和 W 级并没有用到，所以就可以不再流水（仅一般情况）。我们可以这样说，**每个流水线寄存器都保存着一条指令完成后续操作所需要的的全部信息。**

当一个数据被选择成为了流水数据，那么其在 CPU 中就可能存在多个值。比如 E, M, W 级均会有 ALU 的计算结果（三者并不相同），在编程的时候应当使用流水阶段名前缀将其区分开。

思考题

在课上测试时，我们需要你现场实现新的指令，对于这些新的指令，你可能需要在原有的数据通路上做哪些扩展或修改？提示：你可以对指令进行分类，思考每一类指令可能修改或扩展哪些位置。