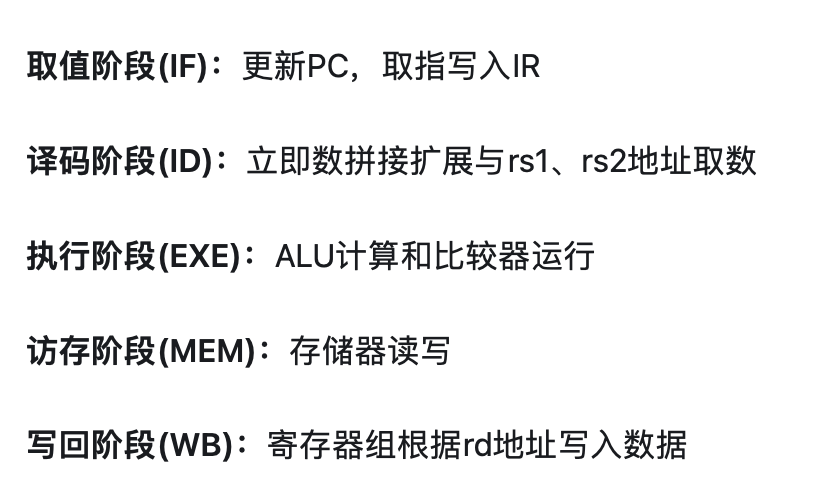
基于第一次会议的总结和规划

1、和CISC最大的不同是，RISC的算术指令不能直接在存储器上操作数据。因此RISC-ISA也被称为加载/存储结构。只使用加载指令和存储指令来访问存储器。

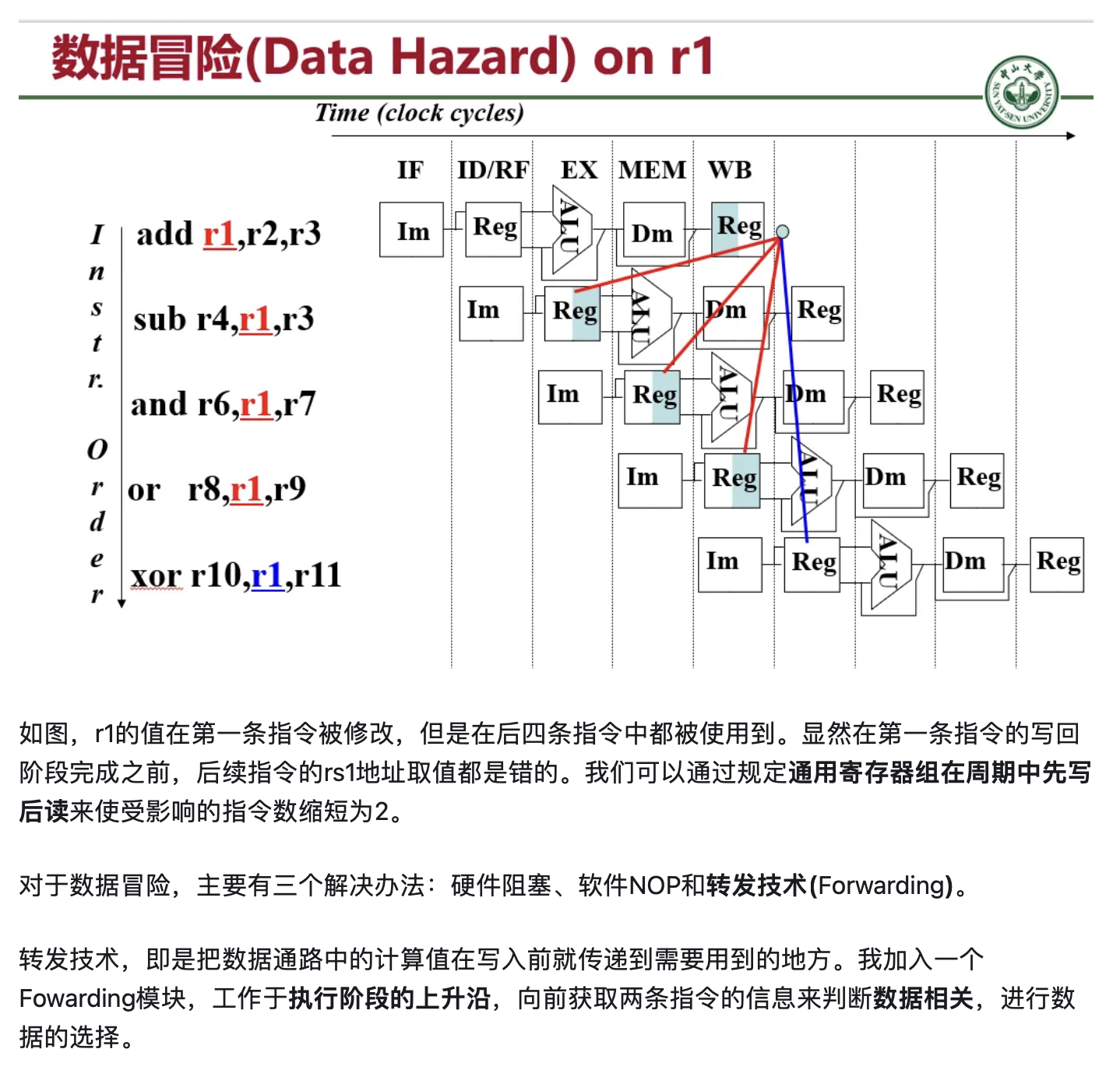
1.1所以RISC需要更多的寄存器，才能保存足够多的数据以保证吞吐量。

1.2本来的架构流水线为3级，取指、译码、执行。这个第三步其实是不严谨的，同时也完成了写回操作。因此，要在ALU的输出端新增一个输出寄存器，开辟新的写回级……

不妨直接采用图1[[1]](#footnote-1)的方案，将流水线直接分级为5级，一步到位。

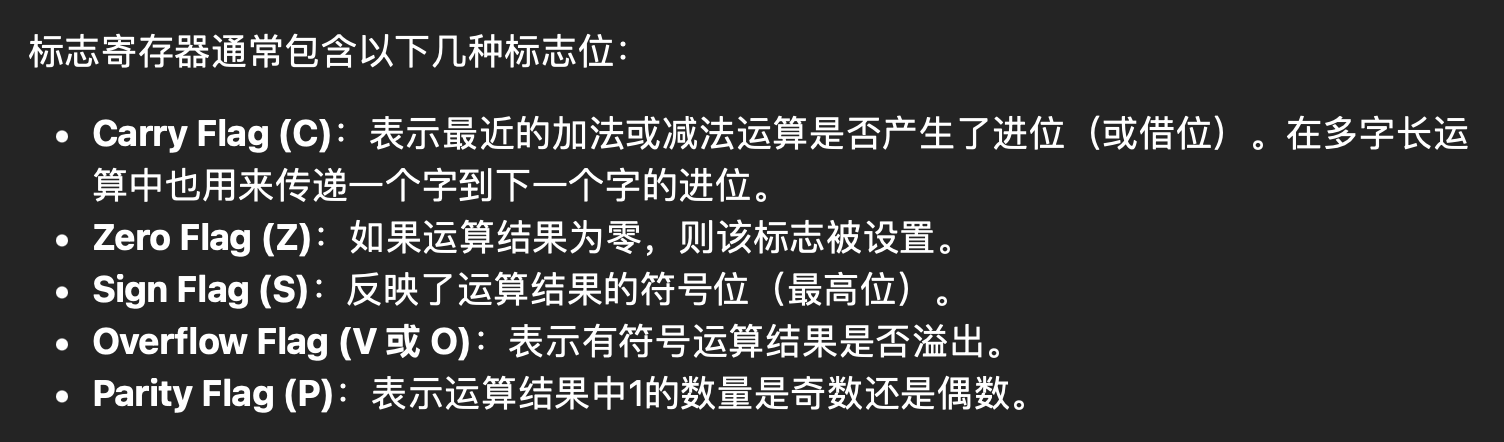
  
图1

这个结构的写回阶段和执行阶段差了2个周期，正如图2所说，会出现某个指令引用了前一次的运算结果，但前一次的结果值还未产生。所以，与之前的CPU的流水线有着很大的不同、难度大大提升，

  
图2

这个方案可以先不急着实现流水线功能，先将基本功能实现。

2、目前的CPU只能存取不超过255的数。也不能分析识别符合数和无符号数，因此我们需要数个标志寄存器的帮助，这样我们就能够操作大小超过1个字节的数。

  
图3

3、为了便于调试，6个数码管要可以根据开关变更显示的内容。

4、用git实现协同工作。

# 5、工作分配

5.1**周润家**：

5.1.1规划、调研、重新设计符合RISC的新指令集形式，在8位不够用的情形下可以考虑扩充指令集规模（或直接套用RISCV）。

5.1.2重新设计CPU架构（重绘框图）。

5.2**何天羽**：

5.2.1对目前的第一代CPU，增添一个输出寄存器，outputregister(CLK,seg,Reg\_output)，在第四个周期（写回）时根据Rd决定将Reg-output存入到指定的位置。（Rd在第四个周期而不是第三个周期发挥作用，因此Rd需要再作一个延迟）

5.2.2 目前的数码管输出为“PC，R1，RAM[R1]”，现在要让开关能够修改输出，比如可以让中间的两个数码管显示R0的值。等等。

5.3**严天誉**：

五个标志寄存器（参考图3）。Carry\_Flag(CLK,Reg\_a,Reg\_b,RamM,seg,Reg\_carry),根据输入输出的结果，Reg\_carry能够表示出是否有进位。

Zero\_Flag(CLK,Reg\_a,Reg\_b,RamM,seg,Reg\_zero)

Sign\_flag(CLK, ,Reg\_a,Reg\_b,RamM,seg,Reg\_sign)

Overflow\_flag(CLK, ,Reg\_a,Reg\_b,RamM,seg,Reg\_overflow)

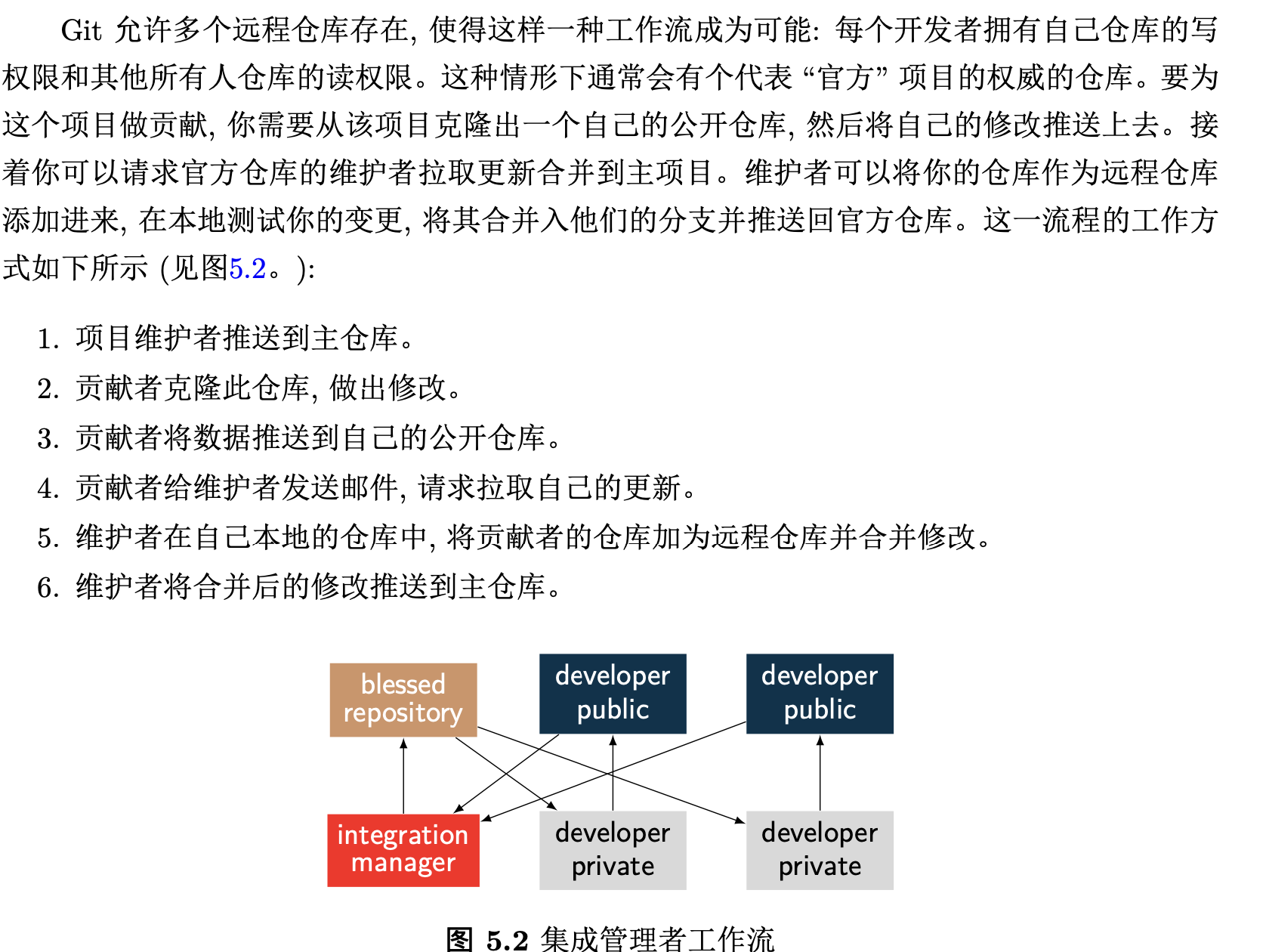
Parity\_flag(CLK, ,Reg\_a,Reg\_b,RamM,seg,Reg\_parity)

5.4**胡严**：

解决因为多了一个写回周期导致的流水线问题。

# 6、特别注意：

现在我们使用git进行协作。参考progit\_cn.pdf的5.1.2节 集成管理者工作流，如下截图所示。



从现在开始，胡严的仓库<https://github.com/hvdanyan/CPU-ON-FPGA>将作为主仓库。fork并clone主仓库之后，一定要先创建一个新的分支，并且在自己的分支上作修改。修改好后，要再次从主仓库里同步最新的主分支，然后把分支合并到主分支上。最后推送到自己的公开仓库上，并告知主仓库的管理者以便将修改同步到主仓库中。

# 7、参考资料

https://zhuanlan.zhihu.com/p/453232311

由胡严编写。

1. 图片的来源是最后的参考资料。图2也同样来自该链接。 [↑](#footnote-ref-1)