

计算机设计与实践

流水线CPU设计与实现



HITSZ 实验与创新实践教育中心
Education Center of Experiments and Innovations, HITSZ

目录



设计要求

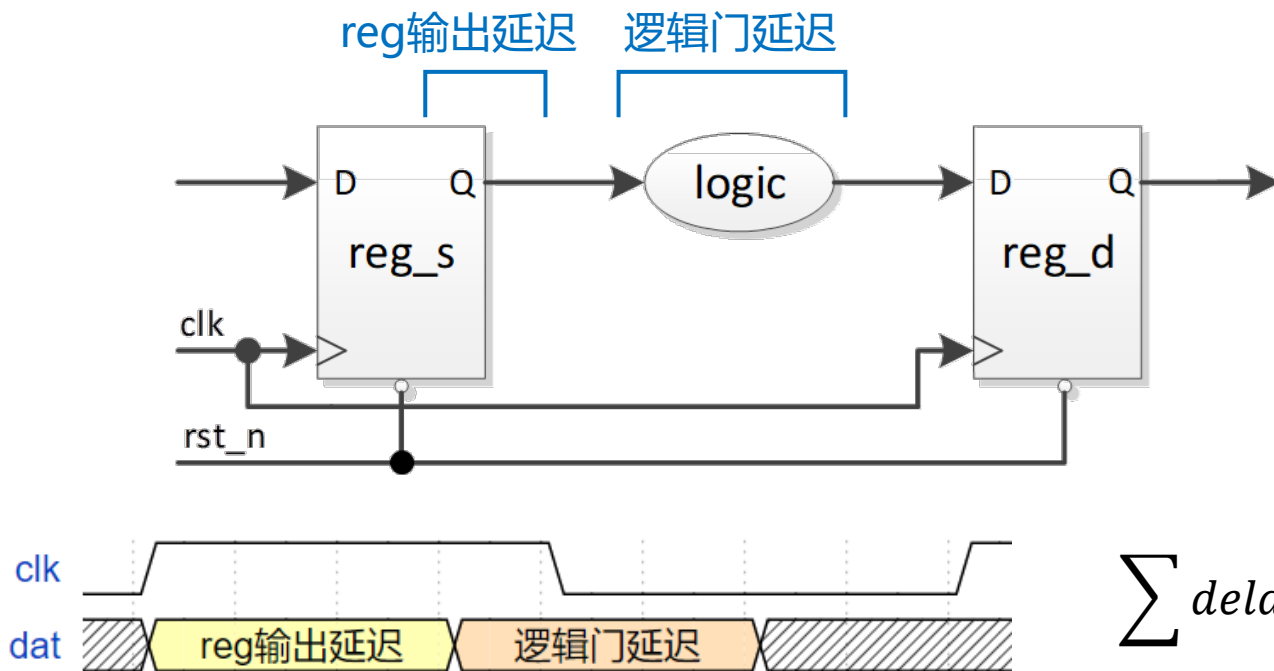
流水线概述

流水线设计

流水线实现

设计要求

- CPU频率的决定因素



$$\sum delay \leq T_{cycle}$$

设计要求

- 五级流水线 (至少)
- 哈佛结构, 分开的指令、数据接口
- 32个32位整数寄存器
- 最少支持24条常规指令 (可选支持其他指令)
- 具有32bit数据、地址总线宽度
- 支持外部中断、异常处理 (可选)

设计要求

可选指令

□ 比较指令 (RV32I)

SLT, SLTI, SLTU, SLTIU

□ 加载/存储指令 (RV32I)

LB, LH, SB, SH

□ 整数乘法指令 (RVM)

MUL, MULH, MULHU, MULHSU

□ 整数除法指令 (RVM)

DIV, DIVU, REM, REMU

□ CSR指令 (RV32I)

CSRRW, CSRRS, CSRRC, CSRRWI, CSRRSI, CSRRCI

目录



设计要求

流水线概述

流水线设计

流水线实现

流水线概述

- 流水线：将指令处理过程**拆分**为多个步骤，并通过多个硬件处理单元**并行执行**来加快指令执行速度
- 理想条件下，流水线CPU的指令执行时间：

$T_{\text{non-pipeline}}$ ：非流水线指令执行时间

T_{Pipeline} ：流水线指令执行时间

N ：流水线级数

$$T_{\text{Pipeline}} = \frac{T_{\text{non-pipeline}}}{N}$$

流水线概述

- 五级流水线

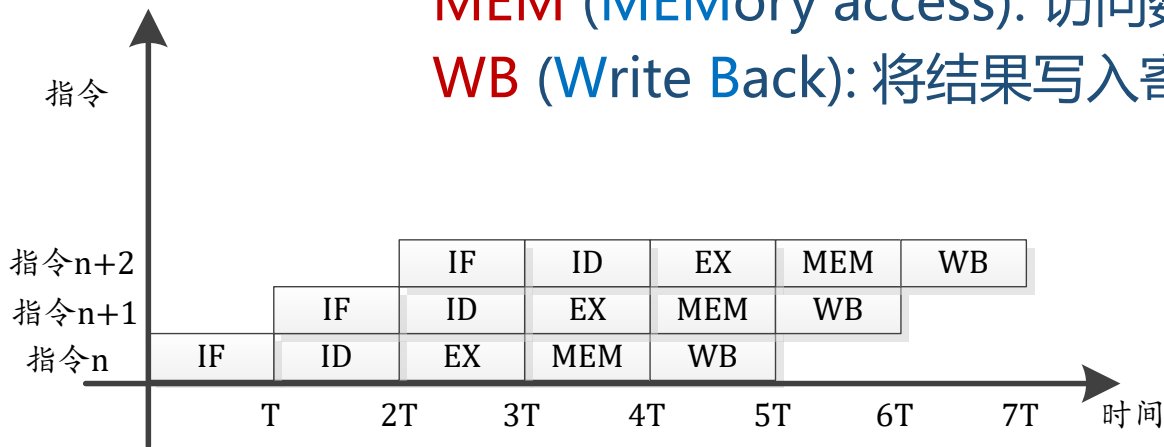
IF (Instruction **F**etch): 从存储器中取出指令

ID (Instruction **D**ecode): 读寄存器并译码指令

EX (**E**Xecute): 执行运算或计算地址

MEM (**M**EMory access): 访问数据存储器

WB (**W**rite **B**ack): 将结果写入寄存器



目录

设计要求

流水线概述

流水线设计

流水线划分和段寄存器

流水线冒险

异常与中断处理

基本外设

流水线实现

流水线设计-流水线划分

- 典型的流水线 (ARM处理器)

- ARM6/ARM7

三级流水线: (Fetch -> Decode -> Execute)

取指 (Fetch): 从寄存器装载一条指令

译码 (Decode): 识别被执行的指令, 并为下一个周期准备数据通路的控制信号

执行 (Execute): 处理指令并将结果写回寄存器

流水线设计-流水线划分

- **典型的流水线 (ARM处理器)**

- ARM8/ARM9

五级流水线: (Fetch -> Decode -> Execute -> Memory -> Write Back)

取指 (Fetch) : 从存储器中取出指令, 并将其放入指令流水线

译码 (Decode) : 指令被译码, 从寄存器堆中读取寄存器操作数

执行 (Execute) : 在ALU中计算结果。如果指令是Load或Store指令, 则在ALU中计算存储器的地址

访存 (Memory) : 如果需要则访问数据存储器, 否则缓冲一个时钟周期

回写 (Write Back) : 将指令的结果回写到寄存器堆

流水线设计-流水线划分

- 典型的流水线 (ARM处理器)

- ARM10

六级流水线: Fetch -> Issue -> Decode -> Execute -> Memory -> Write Back

取指 (Fetch) : 进行指令分支预测, 与指令的读取动作

分派 (Issue) : 判断是否为协处理器指令

译码 (Decode) : 指令被译码, 从寄存器堆中读取寄存器操作数

执行 (Execute) : 在ALU中计算结果。如果指令是Load或Store指令, 则在ALU中计算存储器的地址

访存 (Memory) : 如果需要则访问数据存储器, 否则缓冲一个时钟周期

回写 (Write Back) : 将指令的结果回写到寄存器堆

流水线设计-流水线划分

- 典型的流水线 (ARM处理器)

- ARM11

八级流水线: Fetch1->Fetch2->Decode->ISS->ALU/MAC/LD Pipeline->Write Back

取指1 (Fetch1) : 进行指令动态分支预测, 与指令的读取动作

取指2 (Fetch1) : 进行指令静态分支预测

译码 (Decode) : 指令被译码, 从寄存器堆中读取寄存器操作数

分派 (ISS) : 判断是否为协处理器指令

ALU Pipeline: Shifter -> ALU -> SAT

MAC Pipeline: MAC1 -> MAC2 -> MAC3

Load/Store Pipeline: LS Add -> DC1 -> DC2

回写 (Write Back) : 将指令的结果回写到寄存器堆

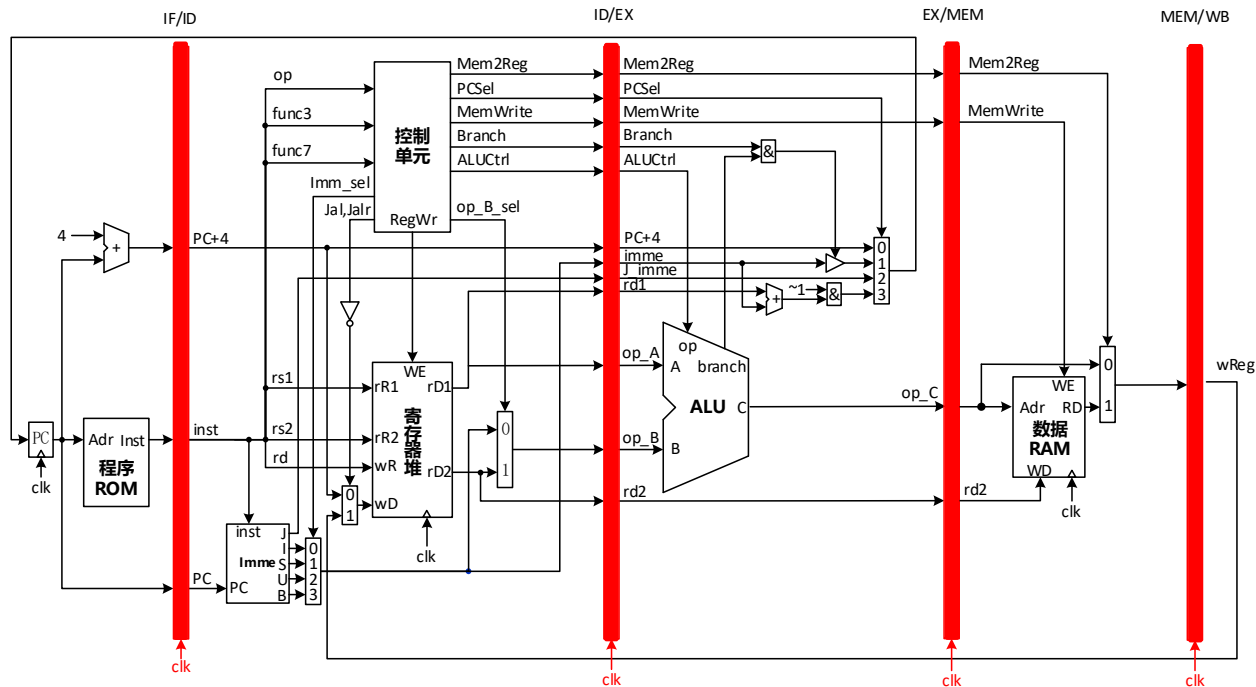
- Cotrex A8: 13级流水线 ...

2021年7月12日11时50分



流水线设计

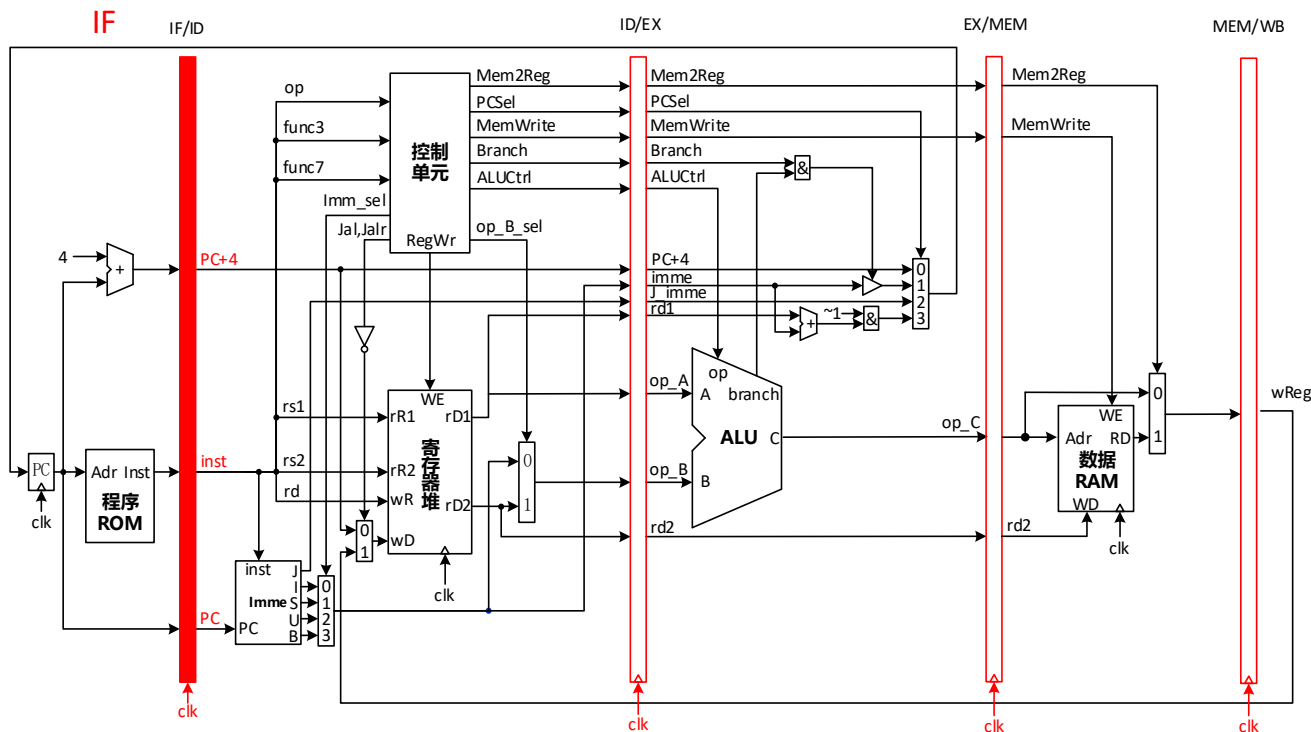
• 流水寄存器



- 切割组合逻辑，提升系统频率
- 保存该流水级输出数据信息，交给下一级处理，并共享给其他指令

流水线设计-流水寄存器

IF/ID流水寄存器



IF (instruction fetch): 从存储器中取出指令

流水线设计-流水寄存器

- IF/ID流水寄存器

IF/ID



PC: J型指令、B型指令

PC+4: J型指令

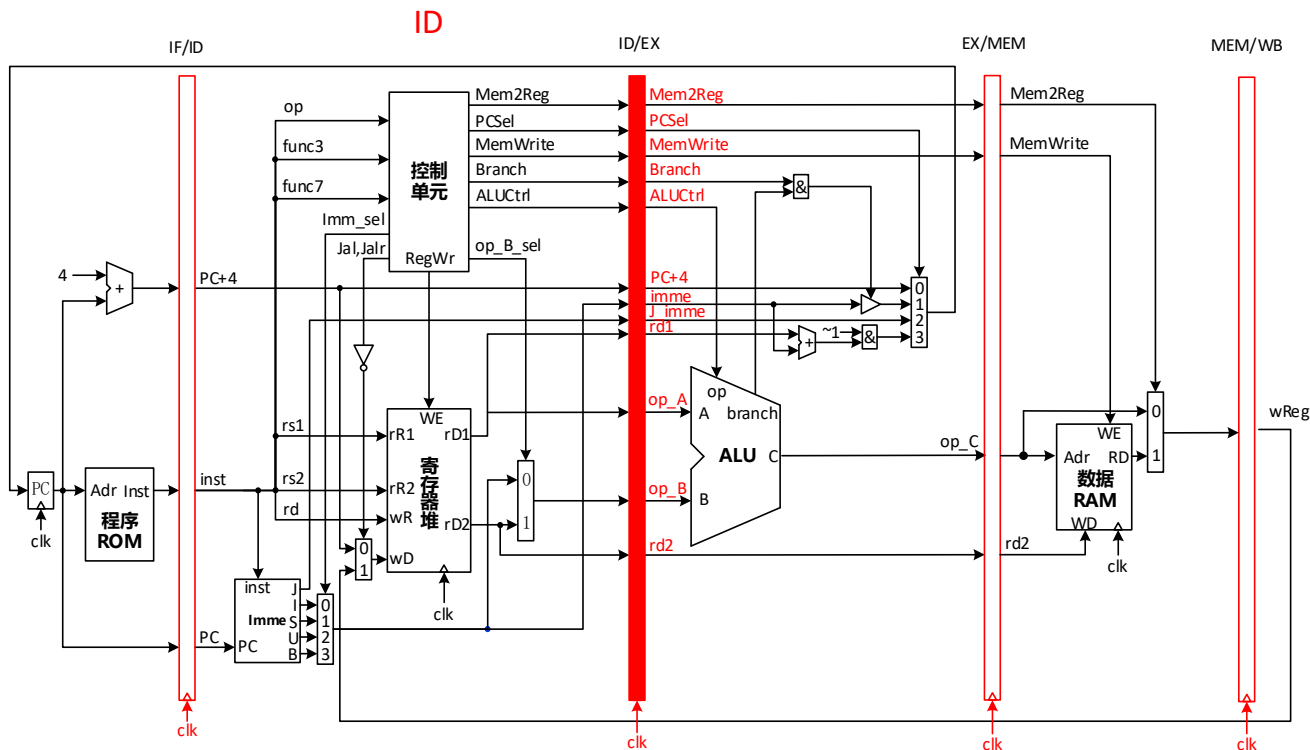
inst: 所有指令

RTL实现

```
16 always @ (posedge clk or negedge rst_n) begin
17     if (~rst_n) id_pc <= 32'h0;
18     else        id_pc <= if_pc;
19 end
20
21 always @ (posedge clk or negedge rst_n) begin
22     if (~rst_n) id_inst <= 32'h0;
23     else        id_inst <= if_inst;
24 end
```

流水线设计-流水寄存器

• ID/EX流水寄存器



ID (instruction decode): 读寄存器并译码指令

流水线设计-流水寄存器

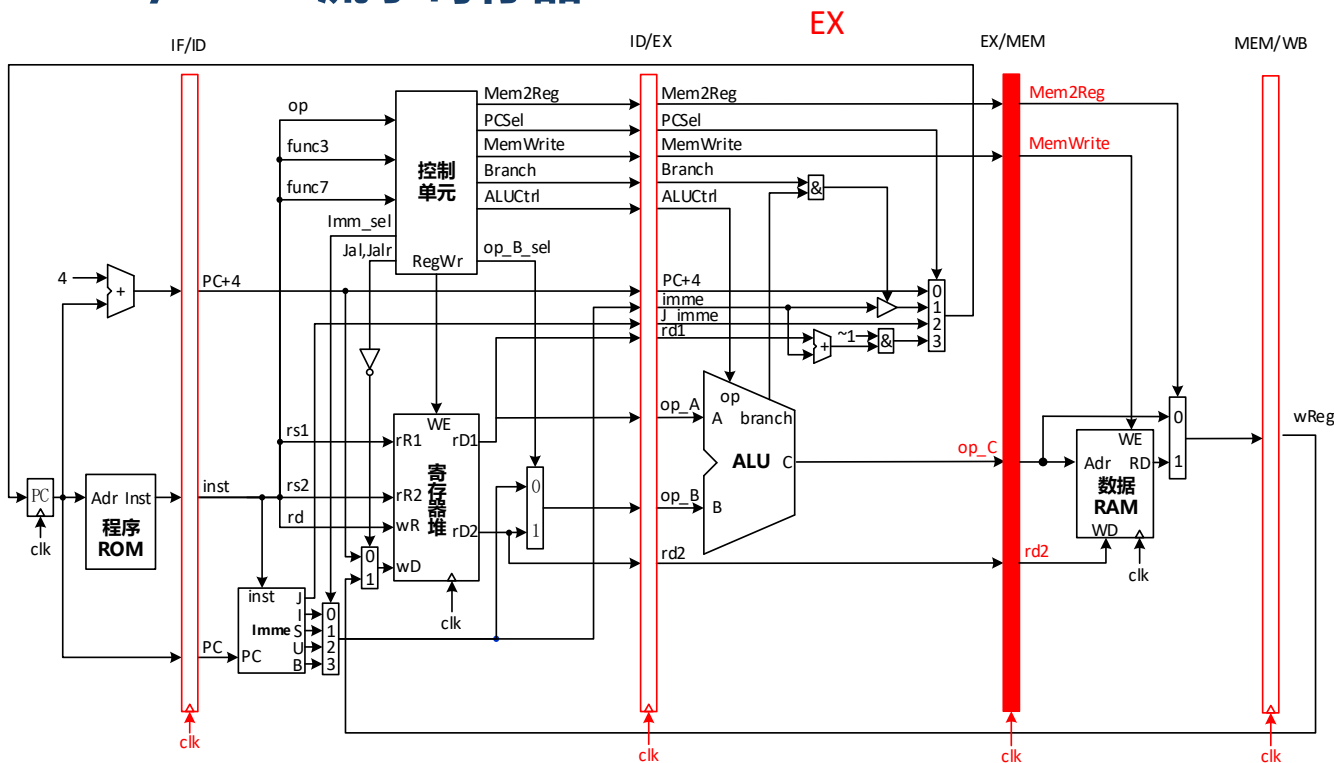
- ID/EX流水寄存器



ID (instruction
decode): 读寄存
器并译码指令

流水线设计-流水寄存器

EX/MEM流水寄存器

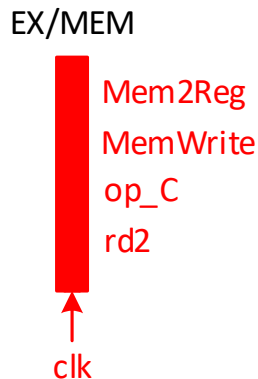


EX (execute):
执行操作或计算
地址

流水线设计-流水寄存器

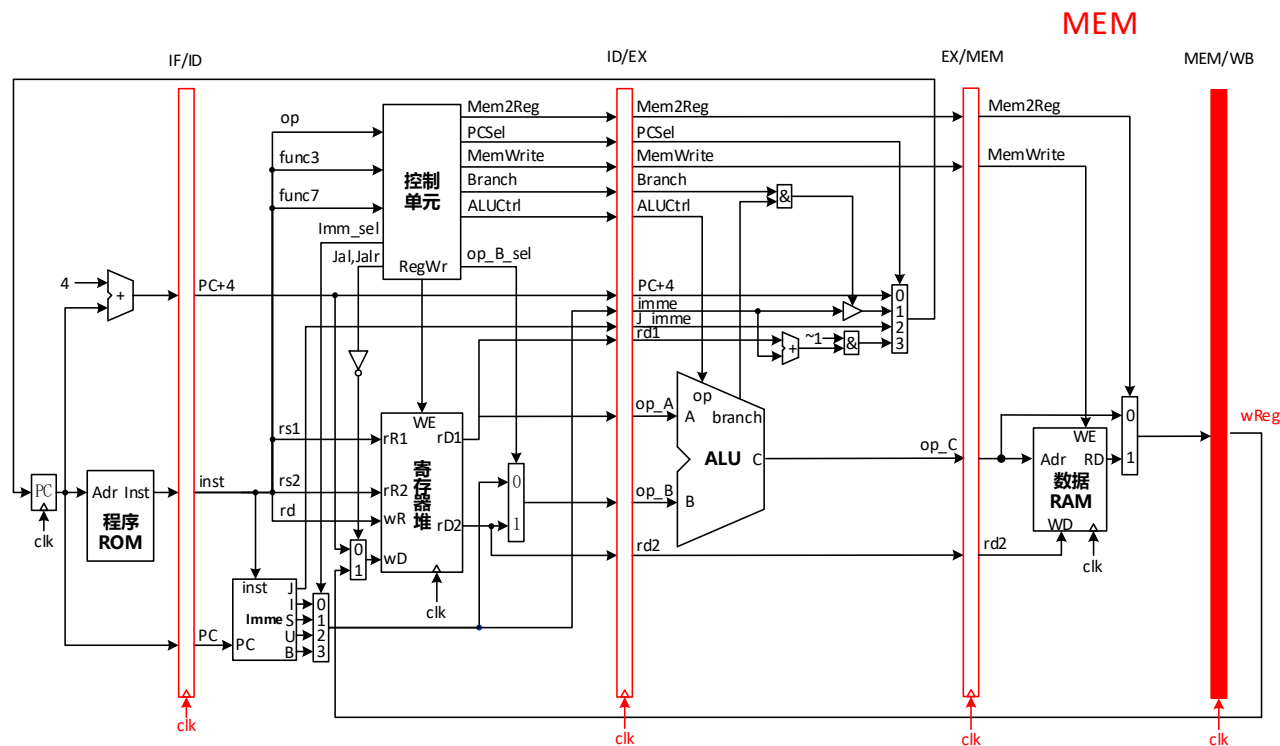
- EX/MEM流水寄存器

EX (execute):
执行操作或计算
地址



流水线设计-流水寄存器

MEM/WB流水寄存器



MEM (memory access): 访问数据存储器中的操作数

流水线设计-流水寄存器

- MEM/WB流水寄存器

MEM (memory access): 访问数据存储器中的操作数



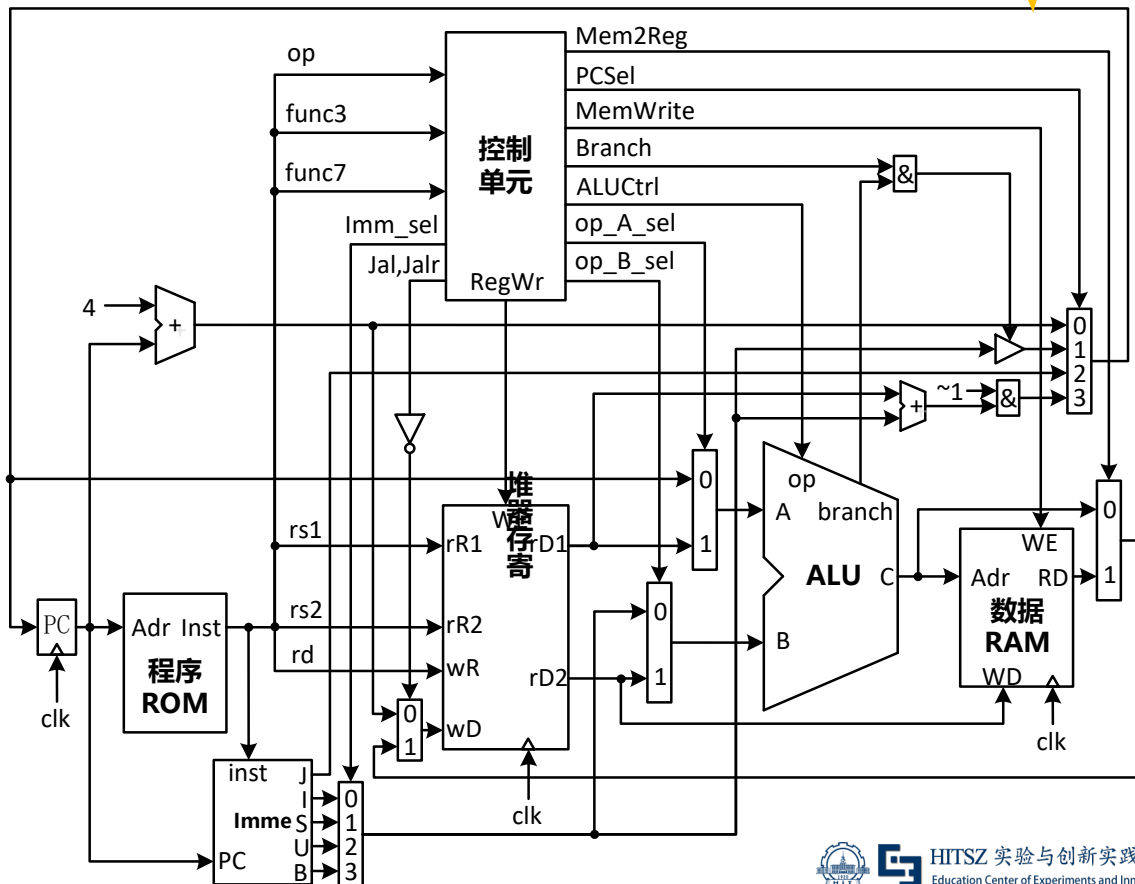
流水线设计-流水寄存器

单周期 → 流水线

How?



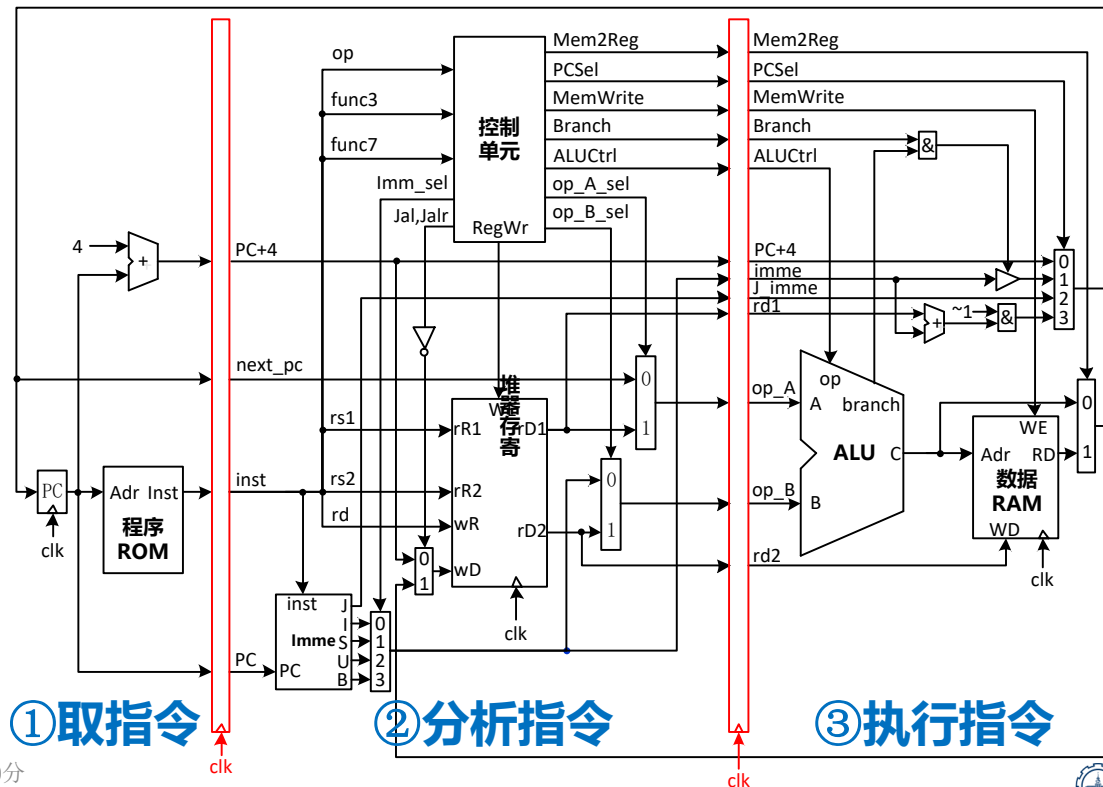
三级流水：
取指-分析-执行



流水线设计-流水寄存器

单周期 → 流水线

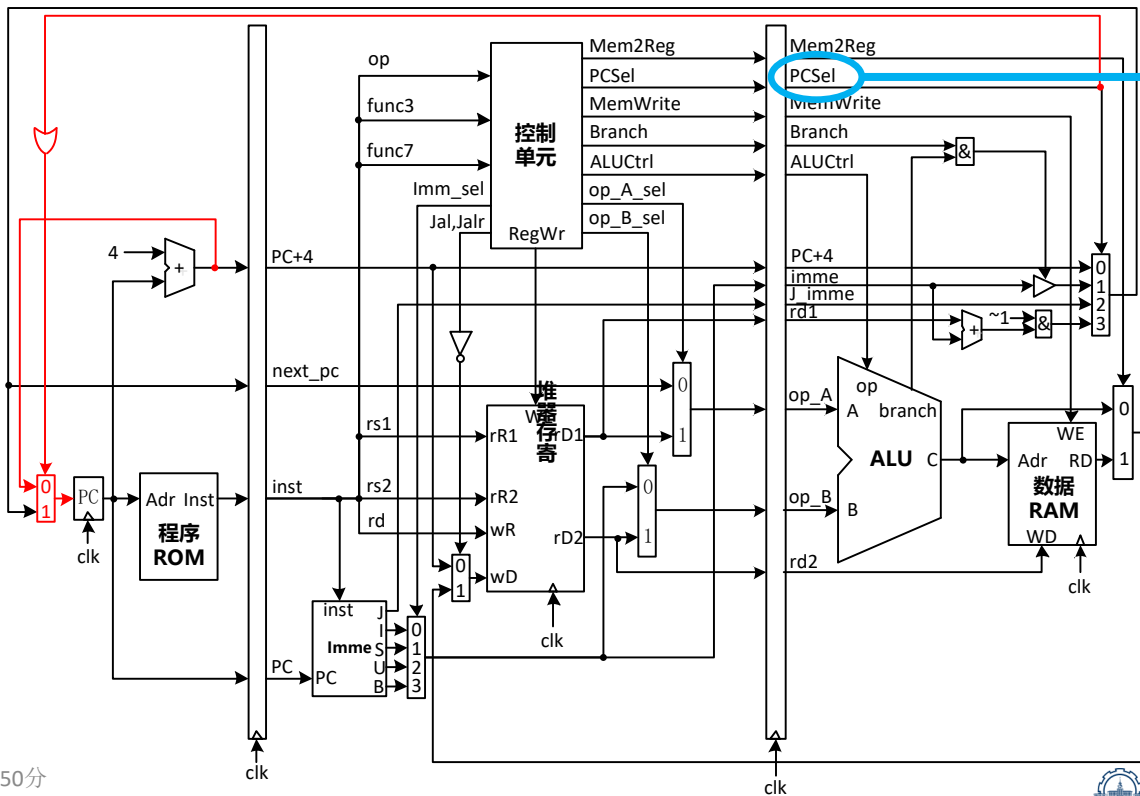
- Step1: 切分数据通路, 添加段寄存器 (红色为新增)



流水线设计-流水寄存器

单周期 → 流水线

- Step2: 修改写PC逻辑, 让流水线“流动”起来 (红色为新增)

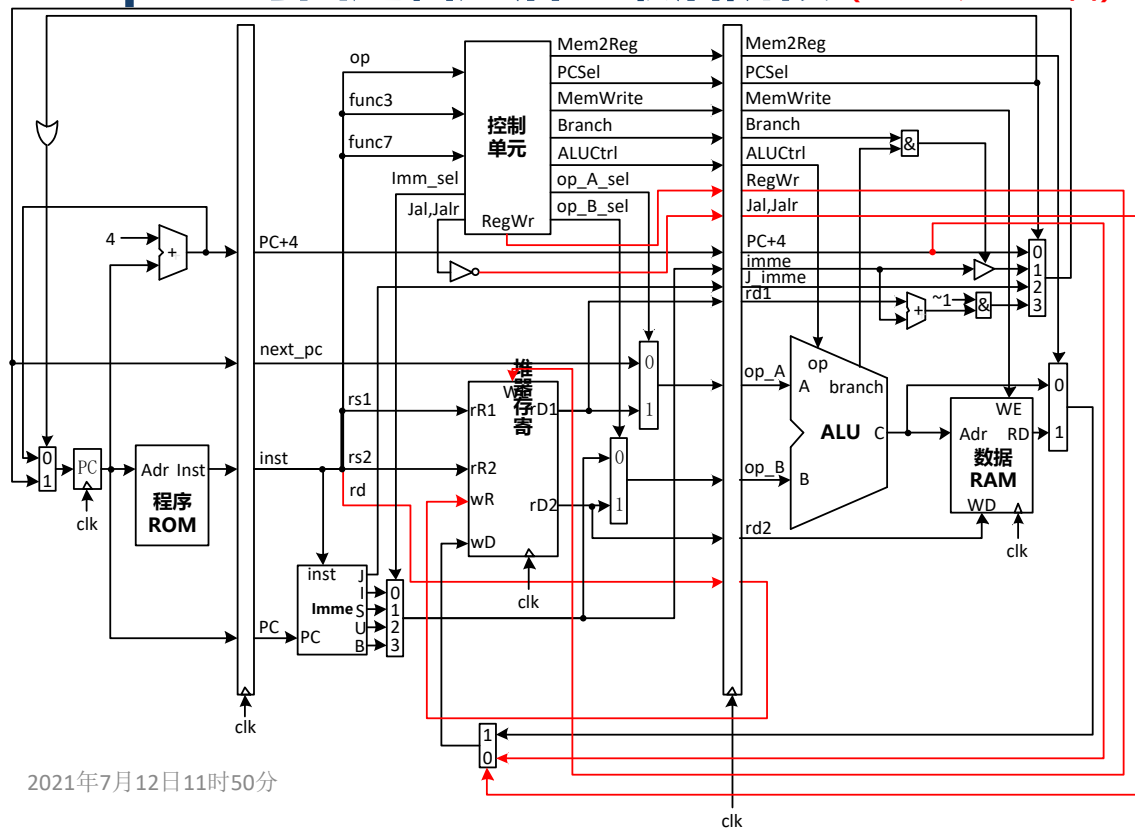


PCsel为1
表示跳转

流水线设计-流水寄存器

单周期 → 流水线

• Step3: 写回逻辑延后至最后阶段 (红色为新增)



➤ 段寄存器的必要性：
存储指令的**执行状态**

执行状态：
指令执行时所有信号的取值构成的集合

➤ 延后写回逻辑，才能得到正确的执行结果

目录

设计要求

流水线概述

流水线设计

流水线划分和段寄存器

流水线冒险

异常与中断处理

基本外设

流水线实现

流水线设计-冒险

冒险 (Hazard): 某些原因导致下一时钟周期无法执行下一条指令的现象

流水线冒险 { 结构冒险
数据冒险 (RAW, ~~WAR~~, ~~WAW~~)
控制冒险

思考

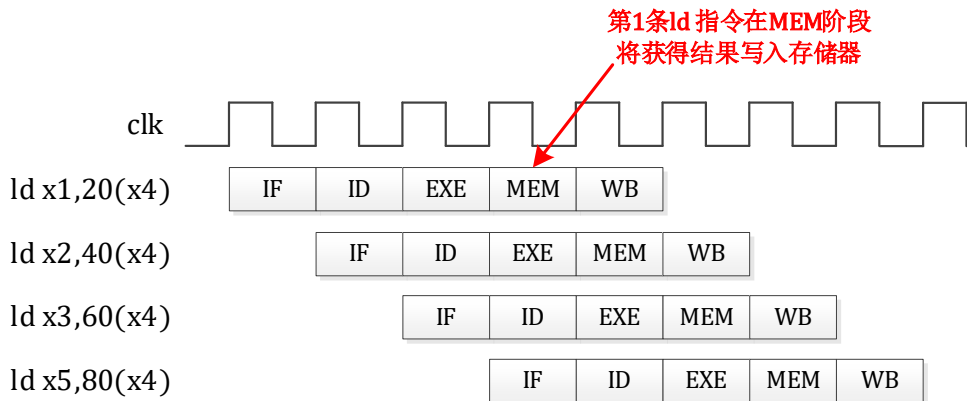
是否存在WAR和WAW?
为什么?

流水线设计-冒险

- 结构冒险

因缺乏硬件支持而导致指令不能在预定的时钟周期内执行的现象

流水线中只有一个存储器，数据与指令共享



解决方法:

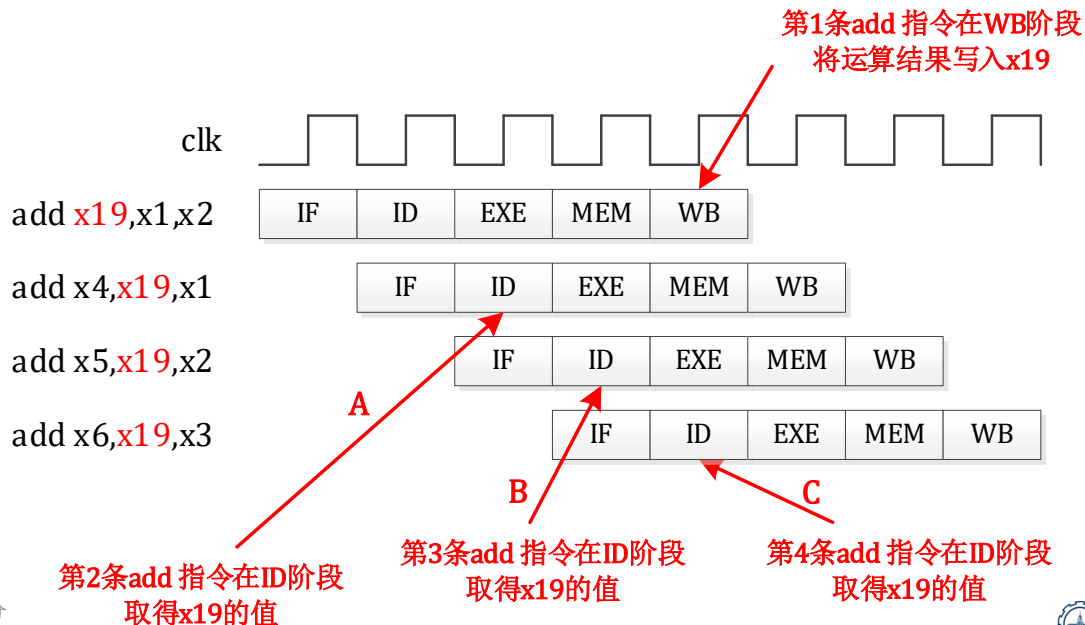
冯·诺依曼结构 -> 哈佛结构

流水线设计-冒险

- 数据冒险

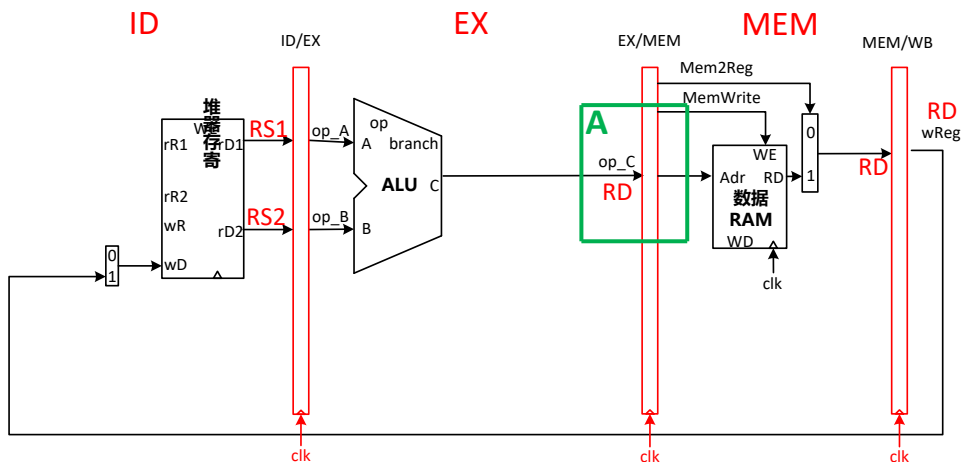
一条指令依赖于前面一条尚在流水线中的指令

因无法提供指令执行所需数据而导致指令不能在预期的时钟周期内执行的现象



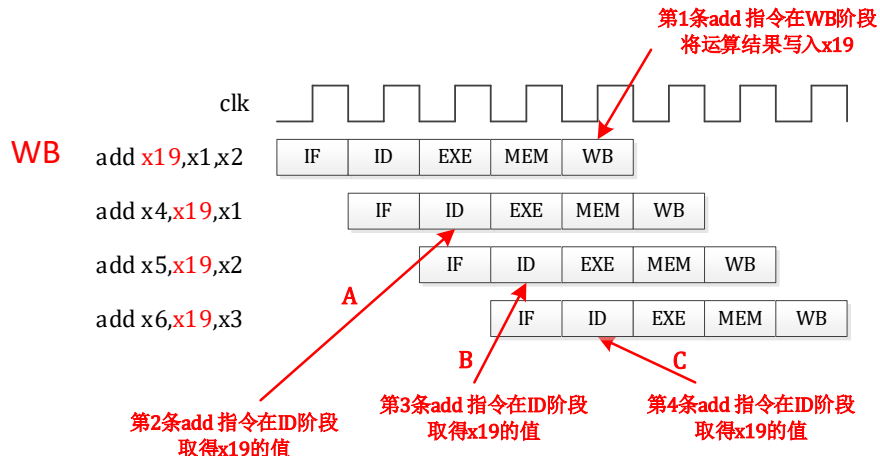
流水线设计-冒险

数据冒险的检测



注意: ID/EX.RS1表示rD1对应的寄存器号
ID/EX.RS2表示rD2对应的寄存器号
EX/MEM.RD表示op_C对应的寄存器号

2021年7月12日11时50分



情形A:

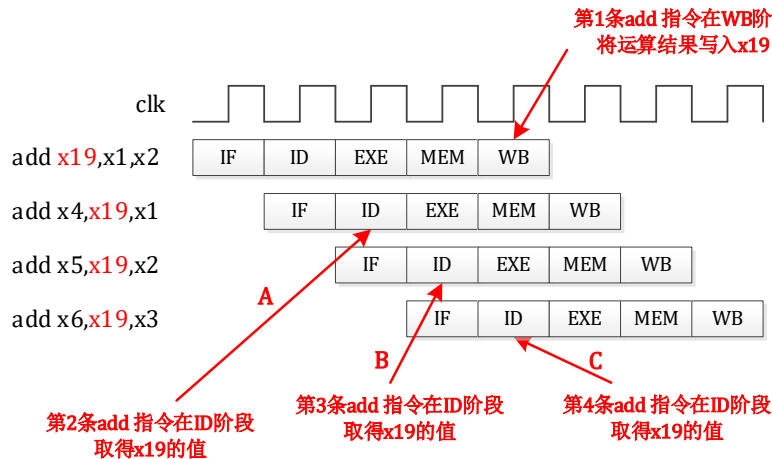
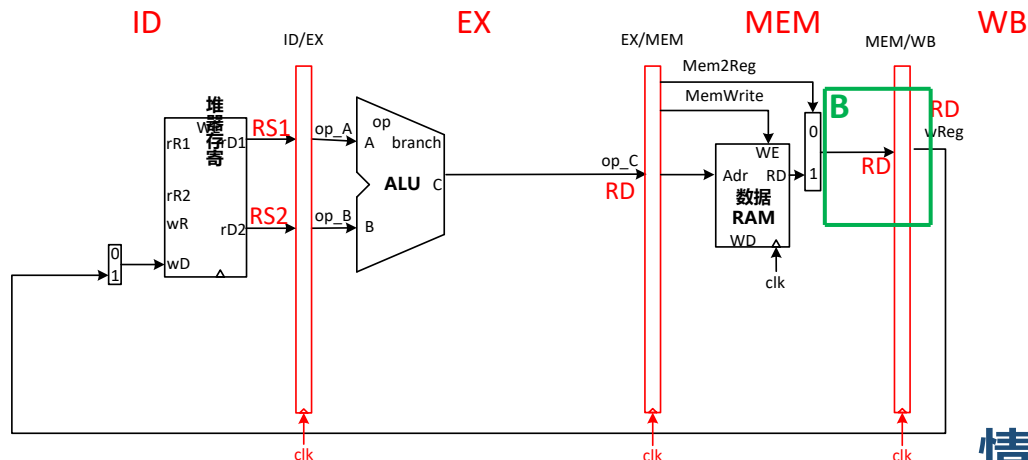
$ID/EX.RS1 = EX/MEM.RD = x19$

$ID/EX.RS2 = EX/MEM.RD = x19$



流水线设计-冒险

数据冒险的检测



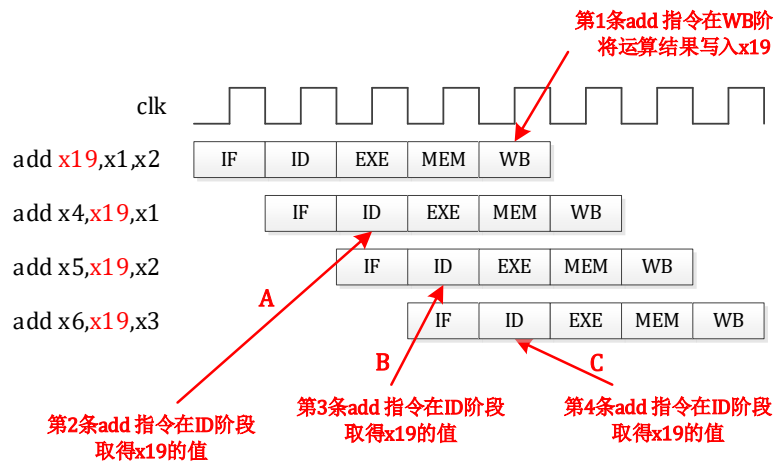
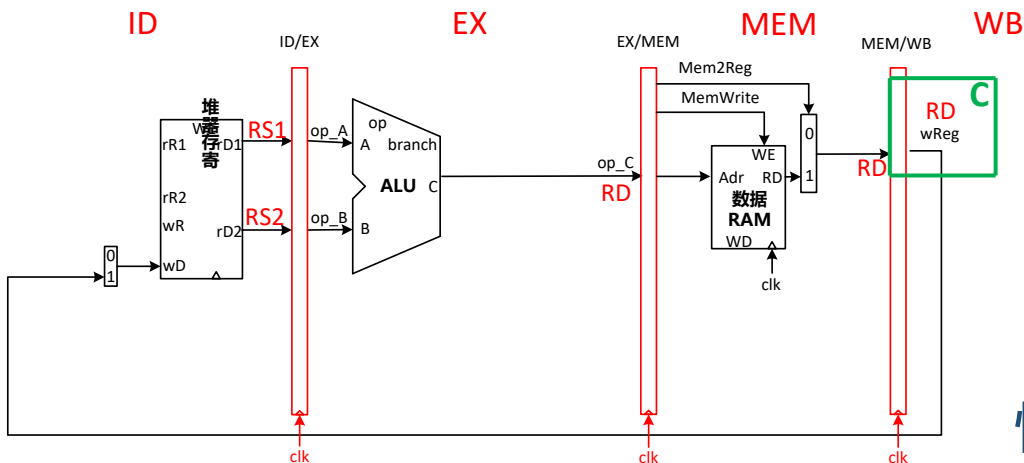
情形B:

$$\text{ID/EX.RS1} = \text{MEM/WB.RD} = \text{x19}$$

$$\text{ID/EX.RS2} = \text{MEM/WB.RD} = \text{x19}$$

流水线设计-冒险

数据冒险的检测



情形C:

$$\text{ID/EX.RS1} = \text{WB.RD} = \text{x19}$$

$$\text{ID/EX.RS2} = \text{WB.RD} = \text{x19}$$

流水线设计-冒险

- 数据冒险的检测

RTL实现

```
wire rs1_id_wb_con = (raddr1 == waddr) & we & re1;  
wire rs2_id_wb_con = (raddr2 == waddr) & we & re2;
```

情形C:

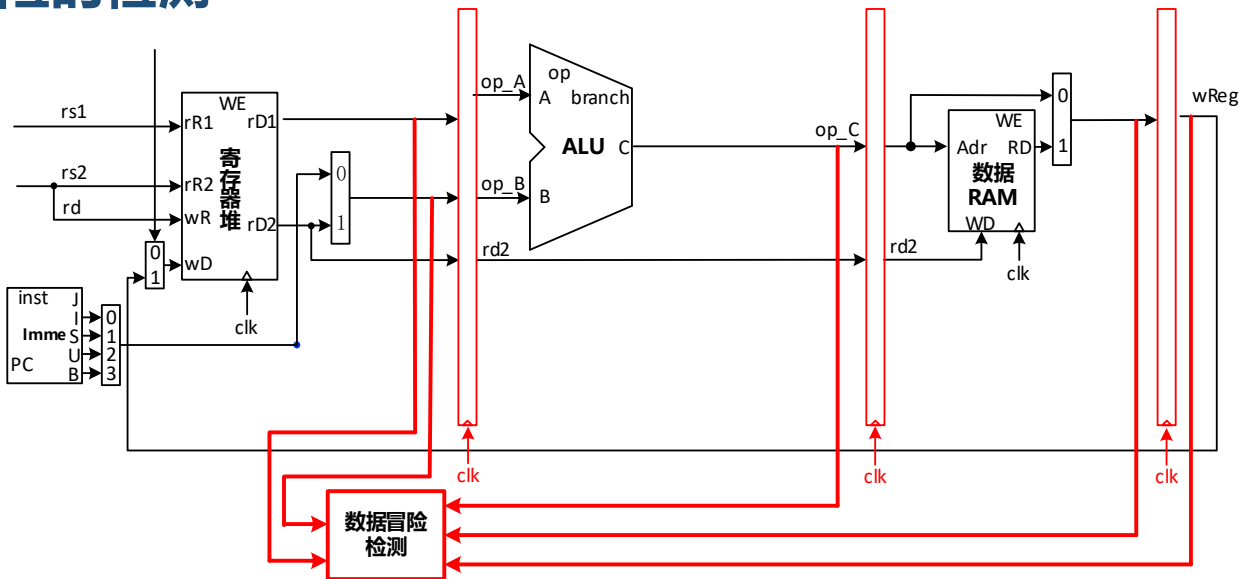
ID/EX.RS1 = WB.RD = x19

ID/EX.RS2 = WB.RD = x19

- ◆ U型、J型指令**没有源操作数寄存器**，不需读取RF，故不存在RAW
- ◆ 单周期的RF采用**异步读取**，可能造成**误判**
- ◆ 为了保证冒险检测的正确性，需添加“读使能”信号加以区分

流水线设计-冒险

- 数据冒险的检测



上图仅表示冒险检测逻辑的**相对**连接关系，**请同学们自行完成实际的电路连接！**

流水线设计-冒险

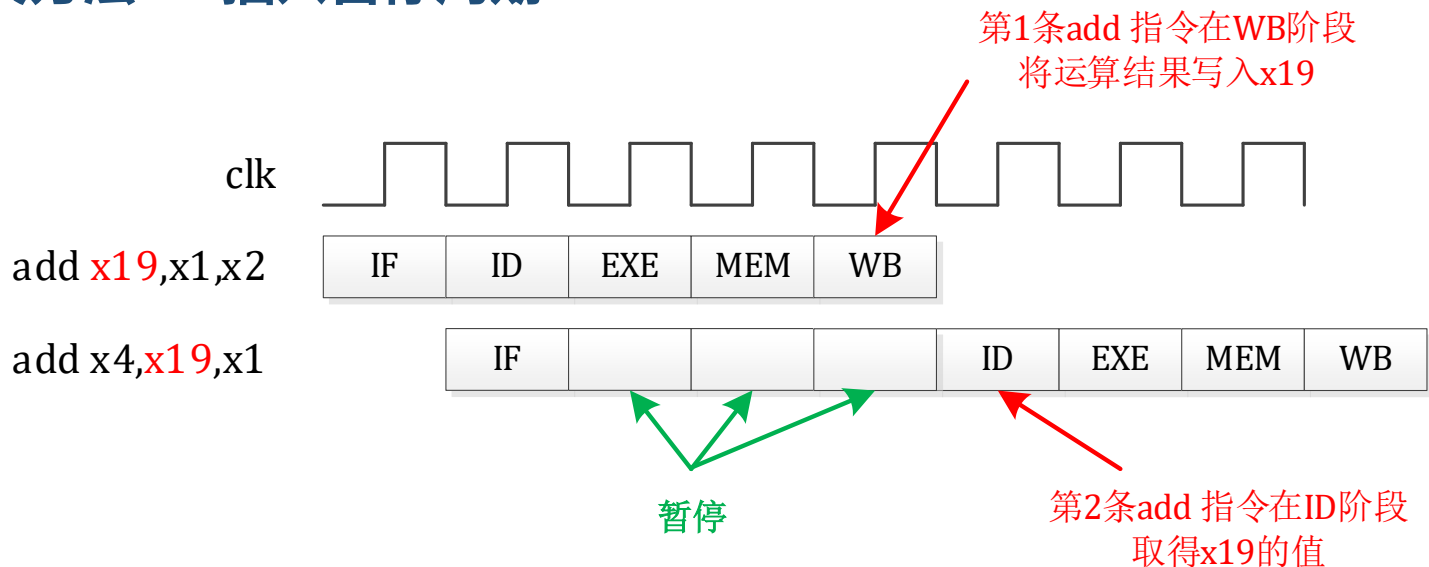
- **数据冒险解决方法**

- 1) 停顿/暂停
- 2) 数据前推/旁路/转发
- 3) 乱序执行

流水线设计-冒险

- 解决数据冒险

解决方法1：插入暂停周期



流水线设计-冒险

- 解决数据冒险

解决方法1：插入暂停周期

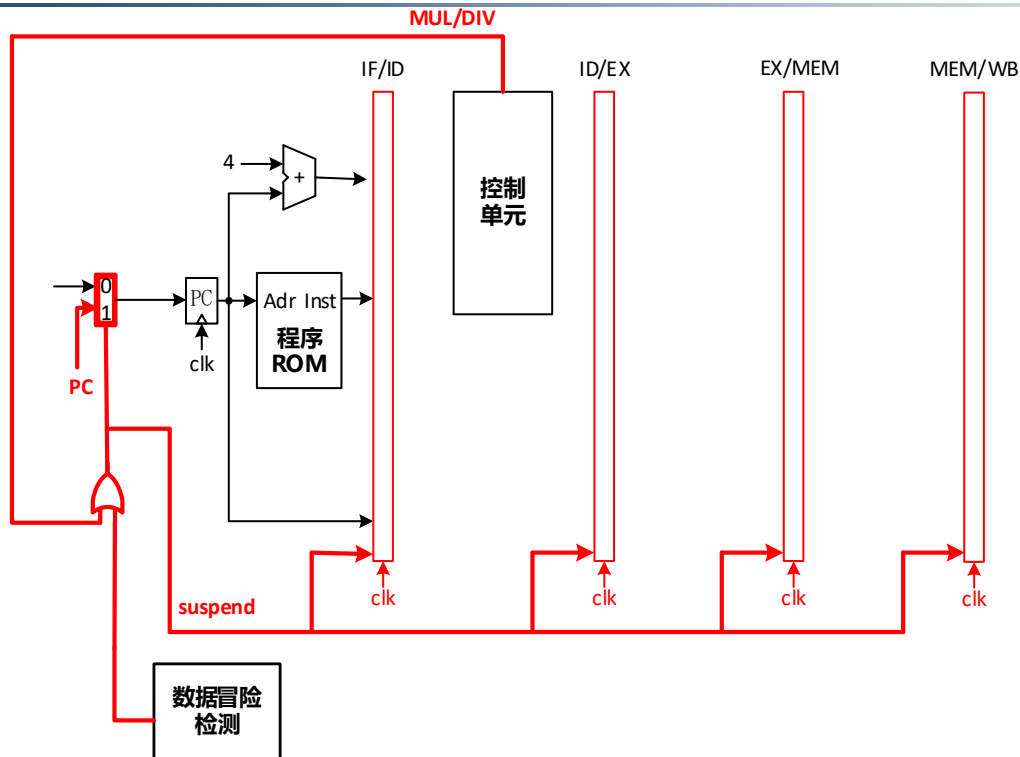
实现方法：

保持PC的值不变

保持流水线的各个段寄存器（IF/ID、ID/EX、EX/MEM、MEM/WB模块的输出）不变

流水线设计-冒险

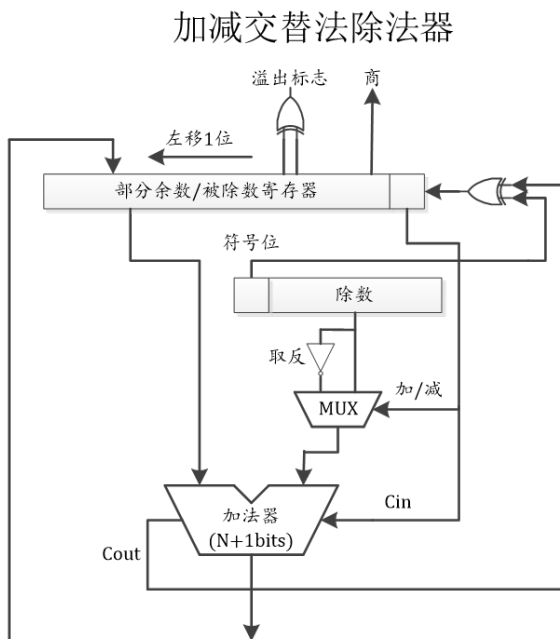
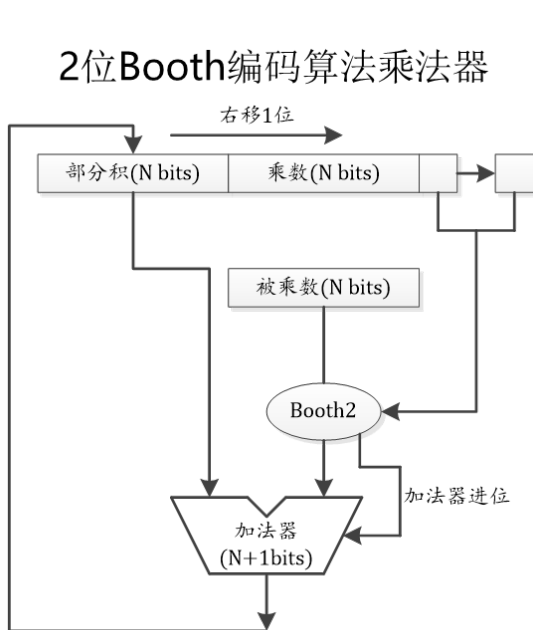
- 暂停机制



上图仅表示暂停逻辑的**相对**连接关系，请同学们自行完成实际的电路连接！

流水线设计-冒险

指令执行周期



操作	执行所需时钟数
除法	32
乘法	17
其余	1

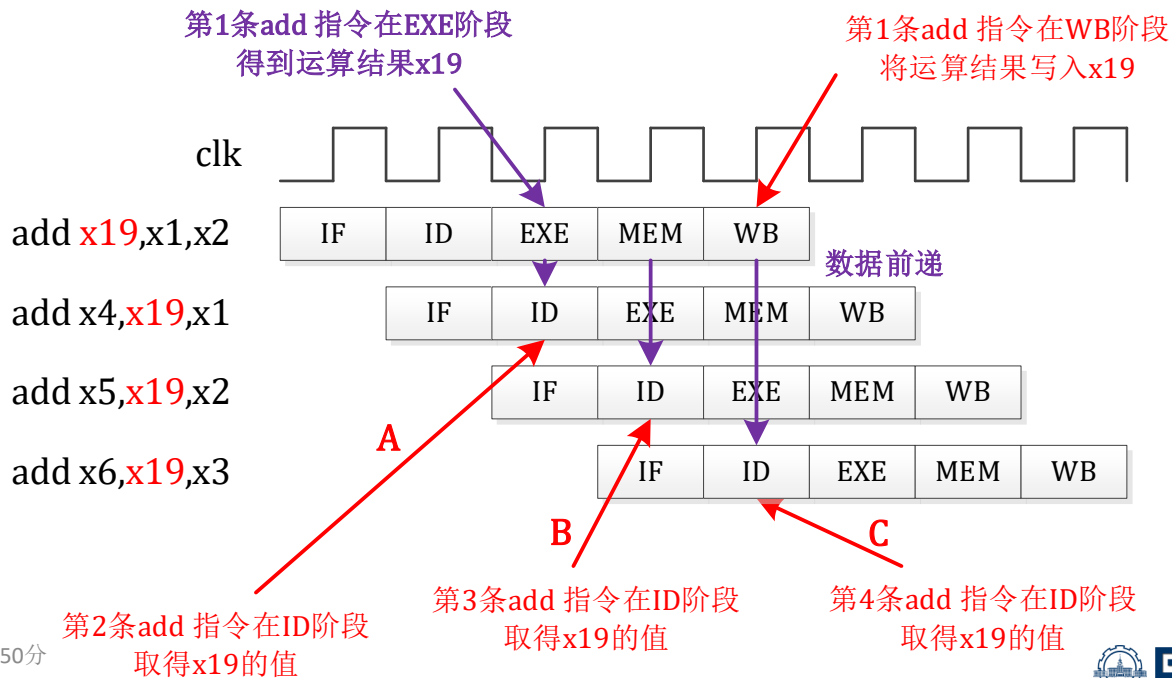


如果调用IP实现?

流水线设计-冒险

- 解决数据冒险

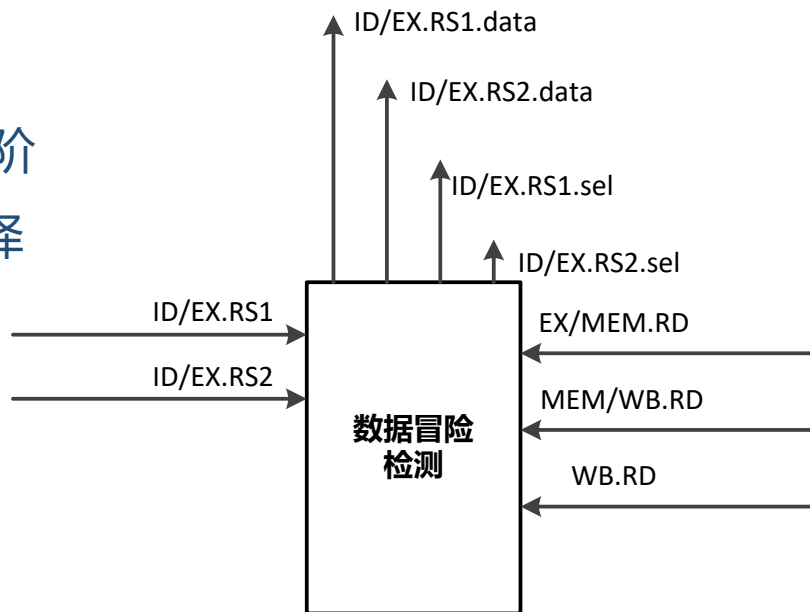
解决方法2：数据前推



流水线设计-冒险

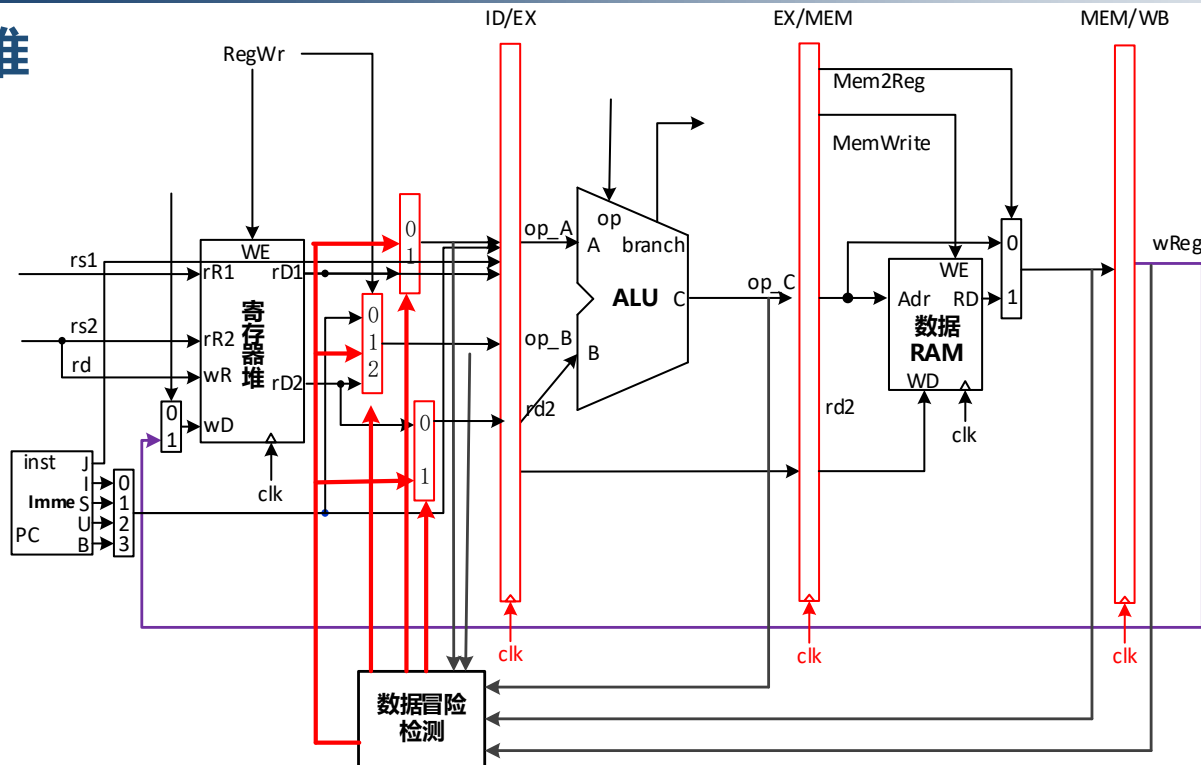
- 数据前推的逻辑

将EXE、MEM阶段的结果前推到ID阶段，参与ID阶段运算源操作数的选择



流水线设计-冒险

- 数据前推



上图仅表示数据前推逻辑的**相对**连接关系，**请同学们自行完成实际的电路连接！**

流水线设计-冒险

- 解决数据冒险

解决方法3：乱序执行法

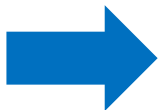
存在数据冒险

add **x2**,x1,x6

add x3,**x2**,x7

add x4,x1,x8

add x5,x1,x9



消除数据冒险

add **x2**,x1,x6

add x4,x1,x8

add x5,x1,x9

add x3,**x2**,x7

无关指令

流水线设计-冒险

- 解决数据冒险

解决方法3：乱序执行法

实现方法：

- ① 软件调度 (汇编器、编译器)
- ② 硬件前瞻执行 (分支预测、Tomasulo算法)

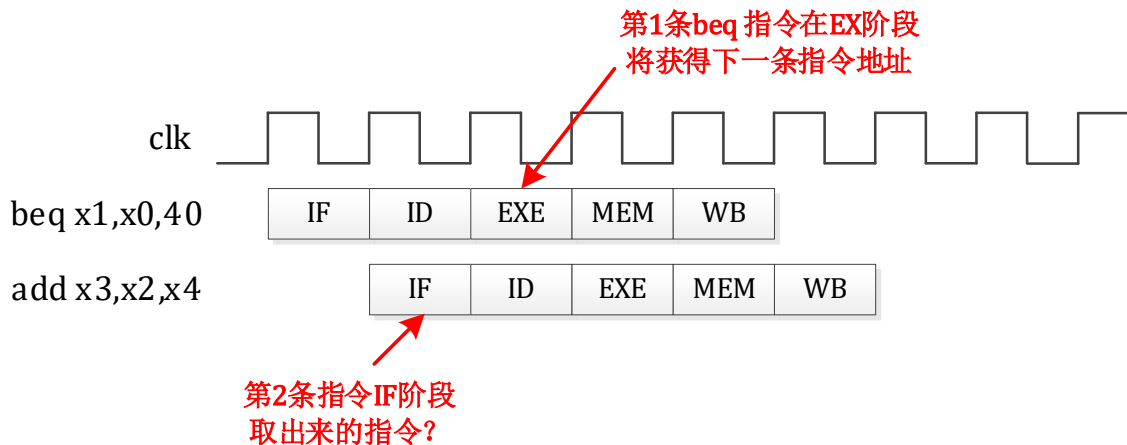
参考资料：《计算机系统结构教程(第2版)》 张晨曦

流水线设计-冒险

- 控制冒险

由于取到的指令并不是所需要的，或者指令地址的流向不是流水线所预期的，导致正确的指令无法在正确的时钟周期内执行的情况

分支指令后的IF段依赖于分支的结果，无法保证永远取到正确的指令



流水线设计-冒险

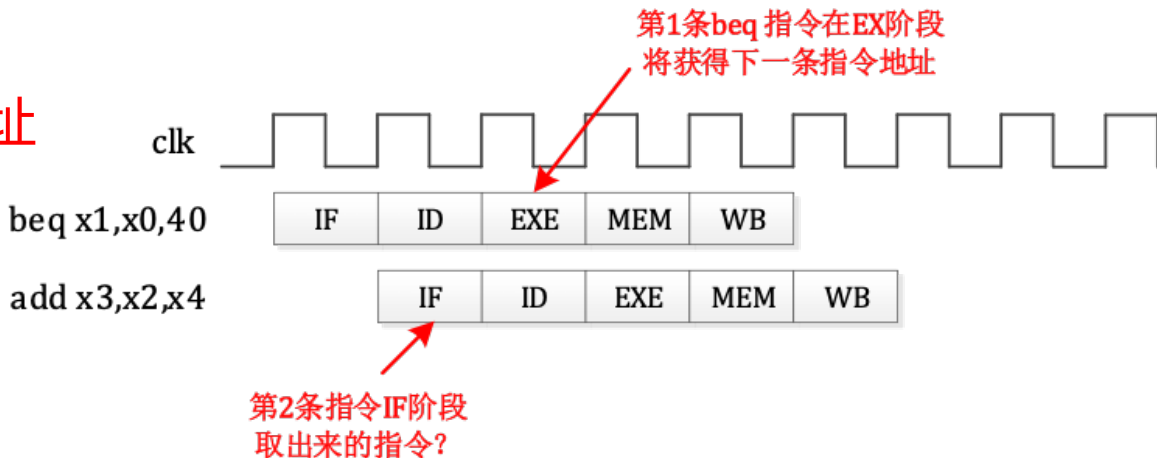
- 解决控制冒险

- 移动硬件，使得分支决定提前到ID阶段

- 需要两个操作提早发生

1) 预测方向

2) 计算分支目标地址



流水线设计-冒险

- 解决控制冒险

- 1) 静态预测方法

- 总是预测跳转 or 总是预测不跳转
 - BTFN预测 (Back Taken, Forward Not Taken)

优点：易实现、开销小；缺点：准确度较低；

流水线设计-冒险

2) 动态预测方法

□ 基于BHT的分支预测

采用分支历史表记录分支历史，并以此预测分支行为

BHT :

Tag	分支历史(2bit)
A0	H0
A1	H1
...	...
A _{k-1}	H _{k-1}

Tag —— 分支指令地址的一部分，类似于Cache的Tag

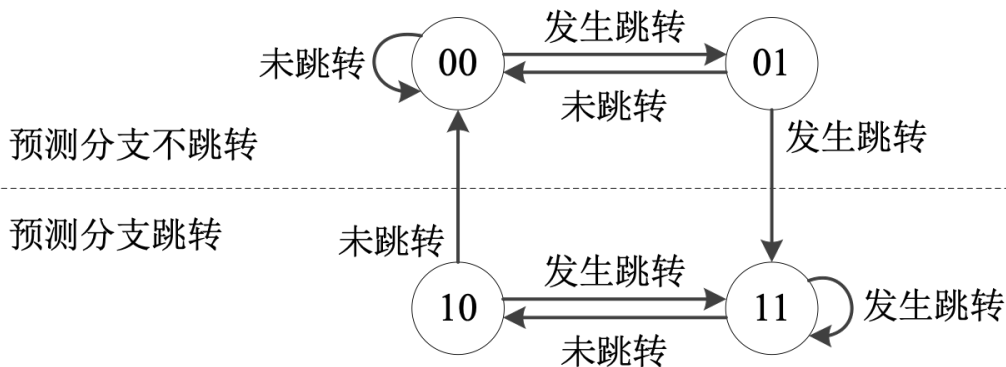
分支历史 —— 2bit饱和计数器

流水线设计-冒险

2) 动态预测方法

□ 基于BHT的分支预测

采用分支历史表记录分支历史，并以此预测分支行为



先用指令地址查BHT，再根据分支历史预测是否跳转

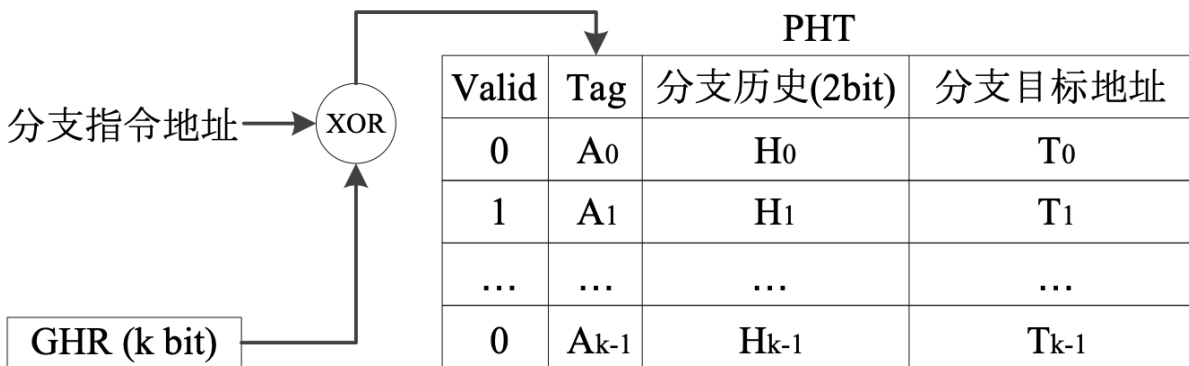
流水线设计-冒险

2) 动态预测方法

□ 基于全局历史的分支预测

BHT方法忽视了分支指令之前的关联性

使用GHR关联所有分支指令，使用PHT记录分支历史行为



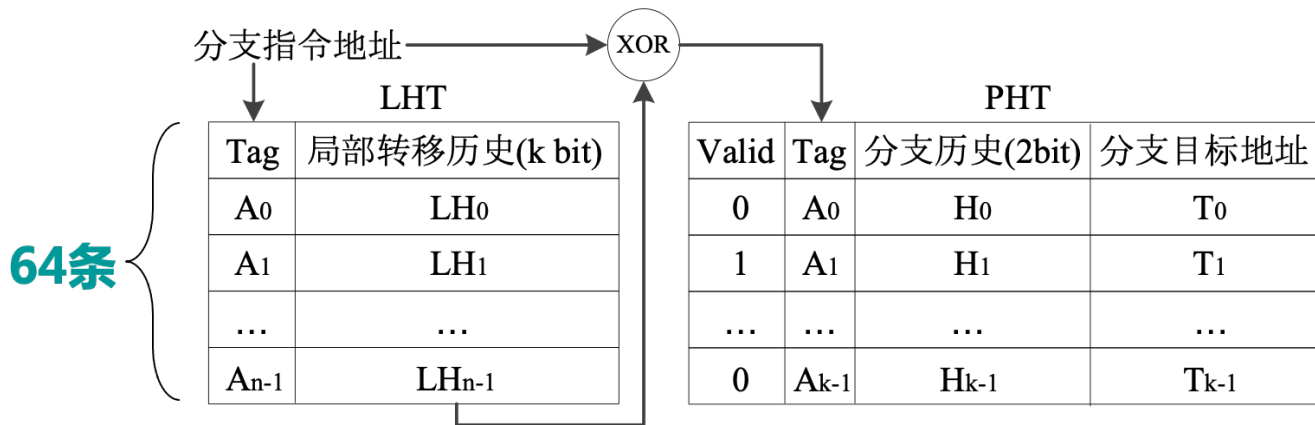
流水线设计-冒险

2) 动态预测方法

□ 基于局部历史的分支预测

全局历史方法过于“简单粗暴”

使用LHT关联局部的分支指令，使用PHT记录分支历史行为



目录

设计要求

流水线概述

流水线设计

流水线划分和段寄存器

流水线冒险

异常与中断处理

基本外设

流水线实现

流水线设计-异常与中断

- **异常 (Exception)**

异常机制：处理器核在顺序执行程序指令流的过程中，突然遇到了异常的事情而中止执行当前的程序，转而去处理该异常

- 1) **同步异常** (Synchronous Exception)

是指由于执行程序指令流或者试图执行程序指令流而造成的异常

- 2) **异步异常** (Asynchronous Exception)

是指那些产生原因不能够被精确定位于某条指令的异常

流水线设计-异常与中断

- 异常 (Exception)

相关寄存器

CSR地址	名称	描述
0x300	MSTATUS	状态寄存器 (Machine Status Register)
0x305	MTVEC	异常入口基地址寄存器 (Machine Trap-Vector Base-Address Register)
0x341	MEPC	异常PC 寄存器 (Machine Exception Program Counter)
0x342	MECAUSE	异常原因寄存器 (Machine Cause Register)

流水线设计-异常与中断

- **进入异常**

举例：add指令 add x1, x2, x1 的EX阶段发生了硬件故障

- IF阶段的指令：变为nop操作
- ID阶段的指令：增加新的逻辑部件，使得译码阶段的输出为0
- EX阶段的指令：需要保留x1原本的值
- 之前的指令，正常完成
- 设置CSR.MECAUSE和CSR.MSTATUS的寄存器值
- 转到例外处理程序CSR.MTVEC

流水线设计-异常与中断

- **退出异常**

- 1) 停止执行当前程序流，转而从CSR 寄存器CSR.MEPC定义的PC地址开始执行。
- 2) 更新CSR.MSTATUS。

流水线设计-异常与中断

- **中断 (Interrupt)**

中断机制：处理器核在顺序执行程序指令流的过程中突然被别的请求打断而中止执行当前的程序，转而去处理别的事情，待其处理完了别的事情，然后重新回到之前程序中中断的点继续执行之前的程序指令流

中断类型：

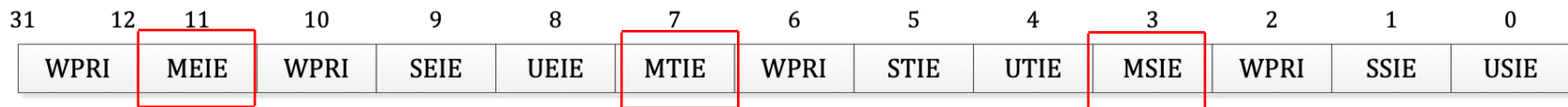
- ❑ 外部中断 (External Interrupt)
- ❑ 计时器中断 (Timer Interrupt)
- ❑ 软件中断 (Software Interrupt)
- ❑ 调试中断 (Debug Interrupt)

流水线设计-异常与中断

- 中断屏蔽

RISC-V的异常不可屏蔽，中断则可被屏蔽

RISC-V定义了CSR寄存器机器模式中断使能寄存器MIE（Machine Interrupt Enable Registers）——用于**控制中断的屏蔽**



MEIE域：控制**外部**中断的屏蔽

MTIE域：控制**计时器**中断的屏蔽

MSIE域：控制**软件**中断的屏蔽

流水线设计-异常与中断

- 中断等待

RISC-V定义了CSR寄存器中断等待寄存器MIP (Machine Interrupt Pending Registers) ——用于**查询中断的等待状态**



MEIP域：反映**外部**中断的等待状态

MTIP域：反映**计时器**中断的等待状态

MSIP域：反映**软件**中断的等待状态

流水线设计-异常与中断

- 中断优先级与仲裁

对于RISC-V架构而言，分为如下3种情况。

- ❑ 外部中断 (External Interrupt) 优先级**最高**
- ❑ 软件中断 (Software Interrupt) **其次**
- ❑ 计时器中断 (Timer Interrupt) **再次**

如果3种中断同时发生，按优先级响应

MCAUSE寄存器中将按此优先级顺序选择更新异常编号 (Exception Code) 的值

流水线设计-异常与中断

- **中断处理流程**

- ① 外设发出中断信号
- ② 保存此时的CSR寄存器 (MSTATUS、MEPC、MCAUSE)
- ③ 跳转到中断处理程序 (MTVEC)
- ④ 关闭其他中断响应使能 (不支持嵌套)
- ⑤ 然后处理中断 (过程中会清掉外设的中断)
- ⑥ 恢复CSR寄存器
- ⑦ 跳转PC跳回原来位置退出中断

目录

设计要求

流水线概述

流水线设计

流水线划分和段寄存器

流水线冒险

异常与中断处理

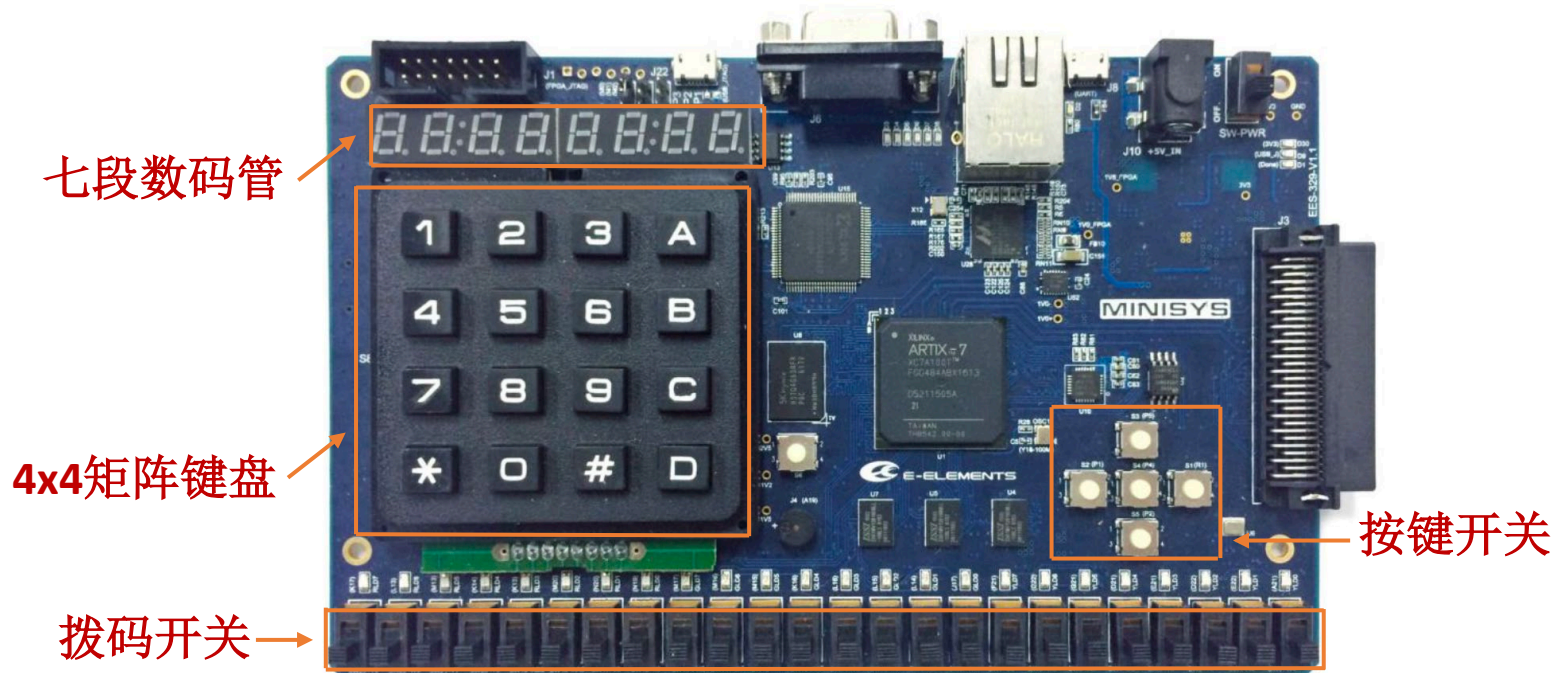
基本外设

流水线实现

流水线设计-外设I/O

- Minisys实验板

主芯片：XC7A100TFGG484-1

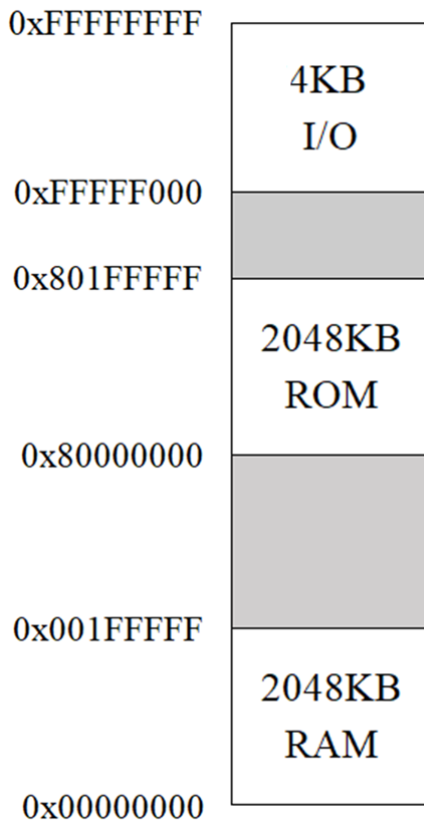


流水线设计-外设I/O

- 外设按统一编址访问

高4KB是I/O地址空间

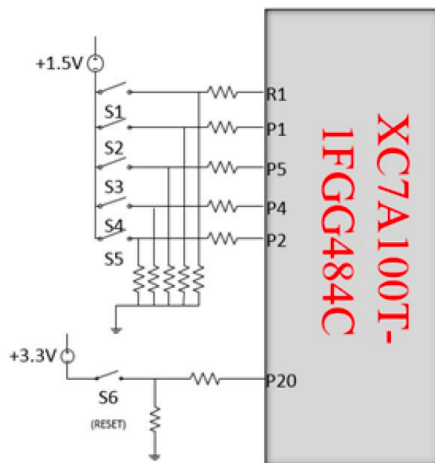
低256KB是DRAM地址空间



流水线设计-外设I/O

- 按键开关

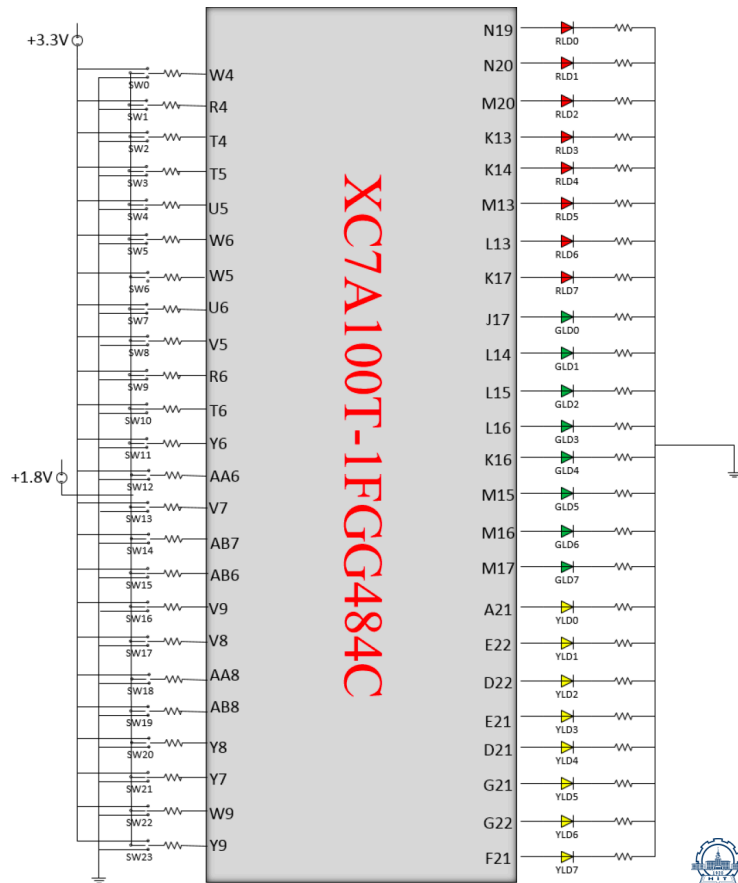
按键按下时，对应的FPGA输入为1，否则为0



流水线设计-外设I/O

• 拨码开关

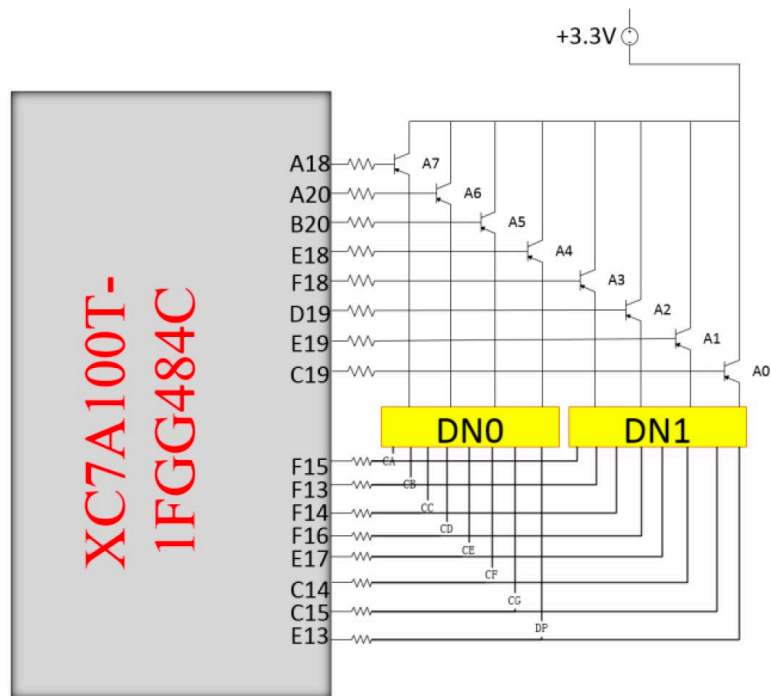
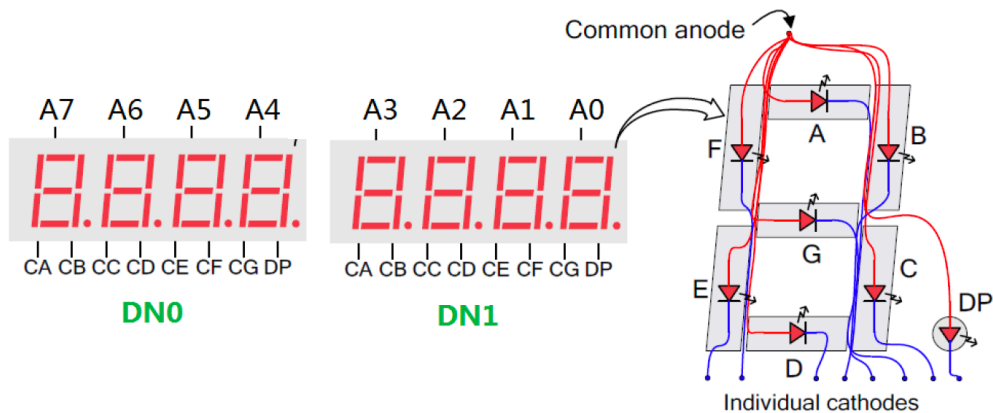
将拨码开关作为数据输入
当开关拨到下档时，表示
输入为0，否则为1



流水线设计-外设I/O

• 七段数码管

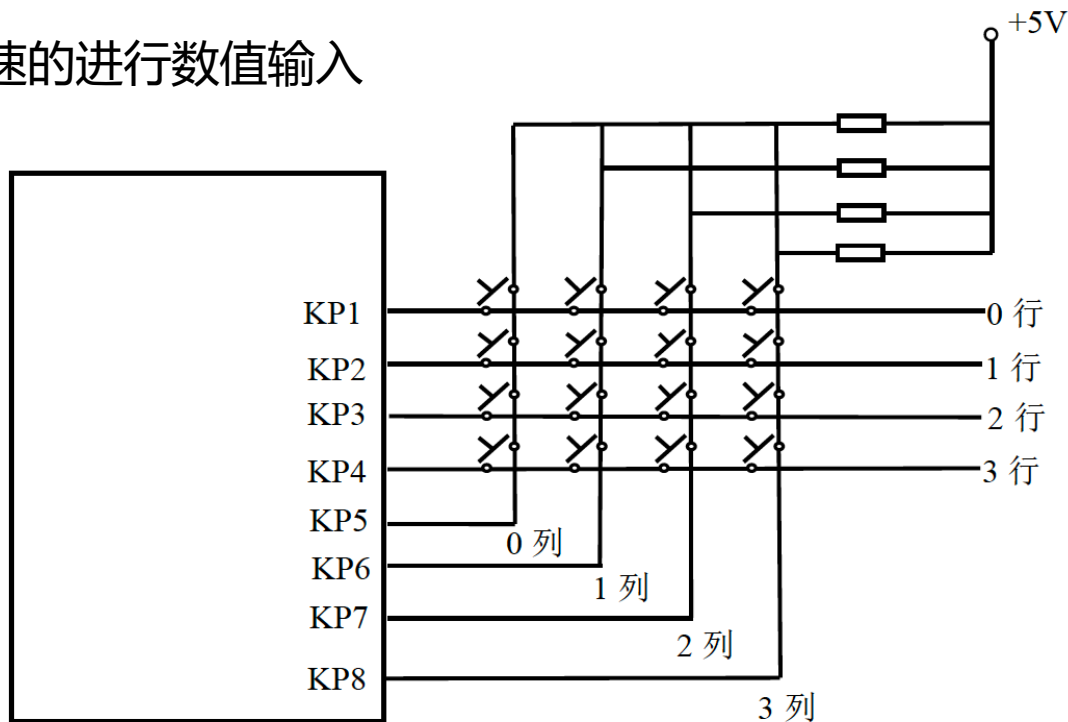
用以显示数据，或者要输出的结果



流水线设计-外设I/O

- 4×4 矩阵键盘

用以方便快速的进行数值输入



目录



设计要求

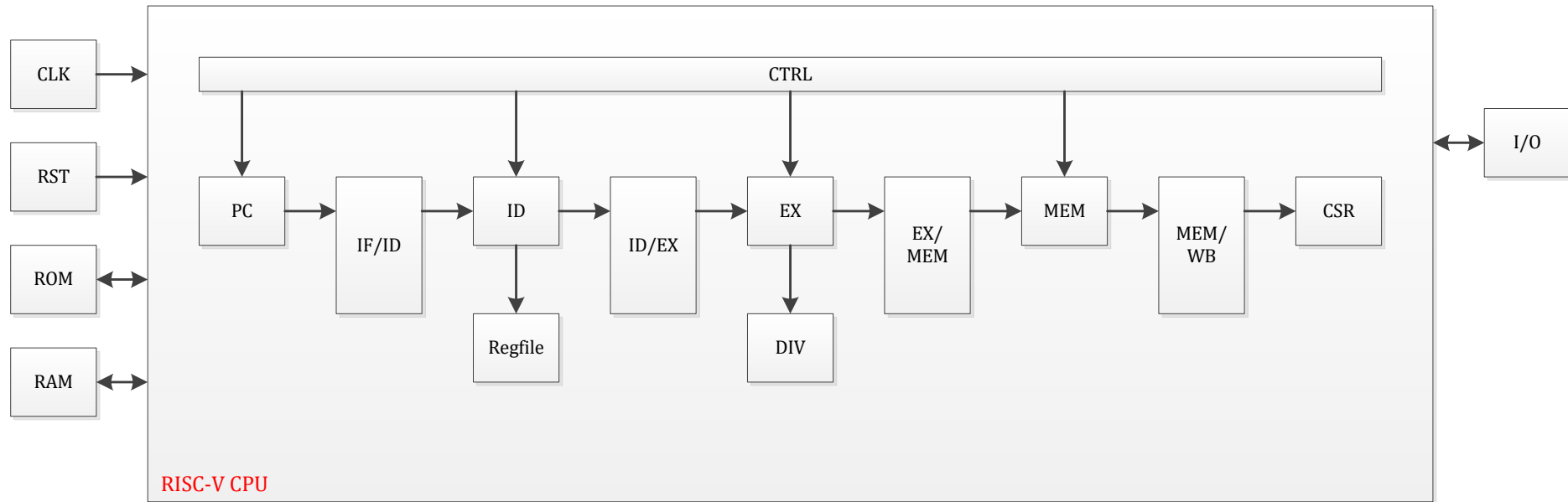
流水线概述

流水线设计

流水线实现

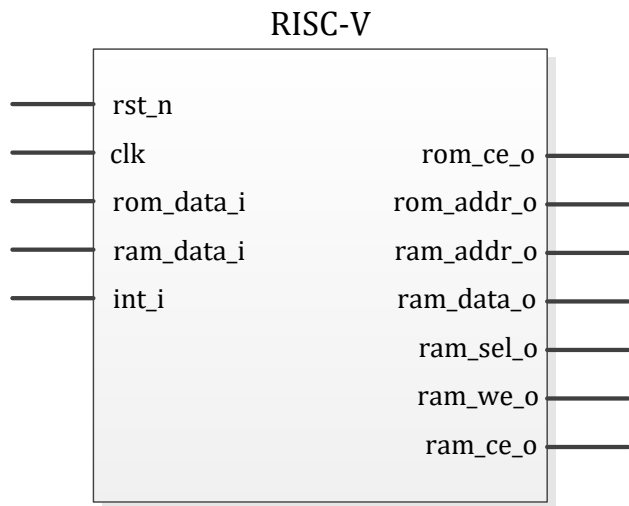
流水线实现

- 基于RISC-V CPU的SOC架构图



流水线实现

- 接口定义



Name	Width	I/O	Description
rst_n	1	I	复位信号（低有效）
clk	1	I	时钟信号
rom_data_i	32	I	从指令存储器取得的指令
rom_addr_o	32	O	输出到指令存储器的地址
rom_ce_o	1	O	指令存储器使能信号
ram_data_i	32	I	从数据存储器读取的数据
ram_data_o	32	O	访问的数据存储器的地址
ram_we_o	1	O	数据存储器的写使能信号
ram_sel_o	4	O	字节选择信号
ram_data_o	32	O	数据存储器的写数据
ram_ce_o	1	O	数据存储器的使能信号
int_i	1	I	外部硬件中断输入

谢 谢!



HITSZ 实验与创新实践教育中心
Education Center of Experiments and Innovations, HITSZ

2021年7月12日11时50分