单周期CPU控制器设计

2021年秋

本讲提要

- 指令执行方式和步骤
- 单周期CPU设计
  - 指令集：9条指令为例
  - 数据通路
  - 控制器

CPU设计

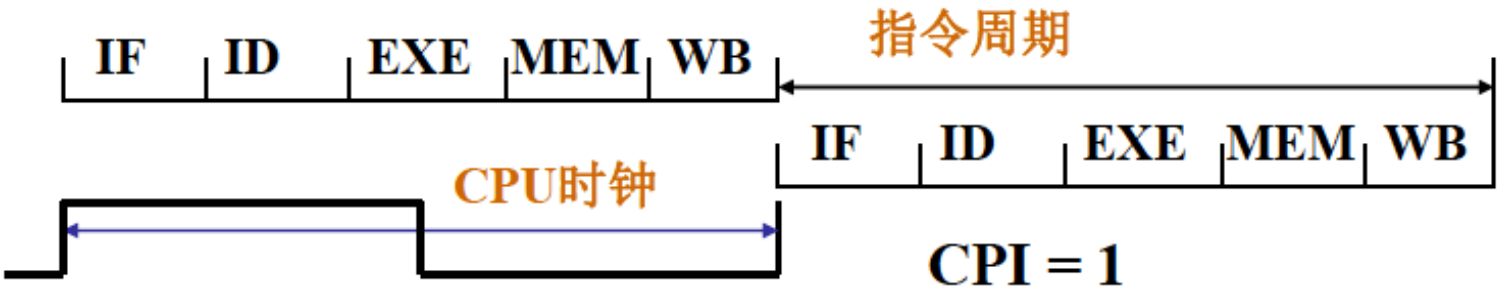
- 完成指令功能的主要部件
  - ALU
- 指令功能
  - 数据运算：算术、逻辑、移位
  - 数据移动： 内存到寄存器之间的数据拷贝
  - 流程控制：转移、调用/返回、中断
  - 其它： 优先级控制，虚拟内存
- 如何实现?
  - Datapath：实现数据的移动和运算
  - 控制器：指挥数据的移动和运算

每条指令的执行过程

- 第一步
  - 取指令（IF）
- 第二步
  - 指令译码（ID）
- 第三步
  - 执行指令（EXE）
- 第四步
  - 访问存储器（MEM）
- 第五步
  - 写回寄存器（WB）

指令执行步骤 —— 单周期CPU

- 计算机一条指令的执行时间被称为指令周期，一个CPU时钟时间被称为CPU周期(在某些计算机中，还可再把一个CPU周期区分为几个更小的步骤，称其为节拍)。执行每条指令平均使用的CPU周期个数被称为CPI
- 全部指令都选用一个CPU周期完成的系统被称为单周期CPU，指令串行执行，前一条指令结束后才启动下条指令。每条指令都用5个步骤的时间完成，控制各部件运行的信号在整个指令周期不变化。单周期CPU 用于早期计算机， 系统性能和资源利用率很低，相对当前技术变得不再实用。



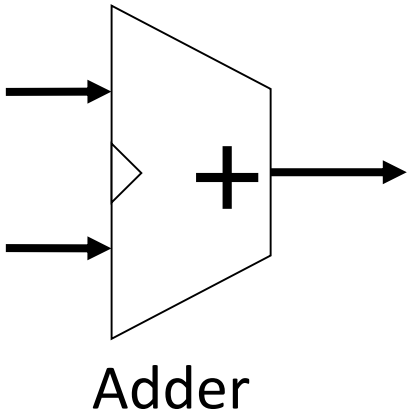
实现的指令集

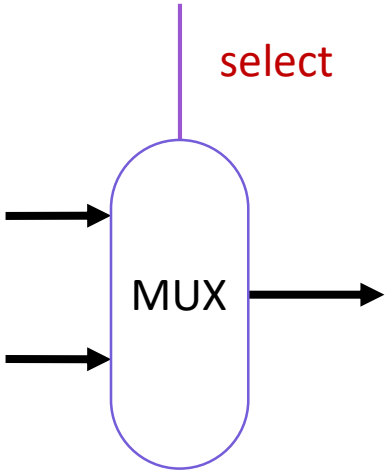
- 选取RISC-V指令中9条典型指令组成的子集
  - 访存指令：lw、sw
  - 算逻运算指令：add、sub、andi、auipc
  - 转移指令：beq、jal、jalr
- 解决三个主要问题
  - 数据通路设计
  - 控制信号设计
  - 执行时序设计
- 其他指令的实现原理可以从中体现
  - 上面9条指令覆盖了所有的RISC-V指令集的不同类型指令

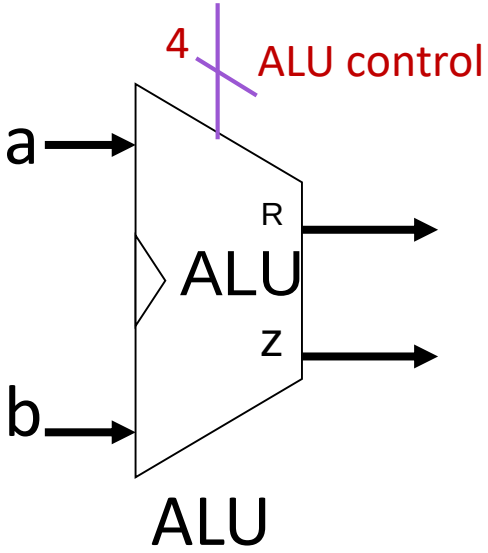
设计思路

- 指令的执行
  - 显然要设计一个时序逻辑电路
  - 一条指令用一个CPU周期完成
- 执行步骤的实现
  - 取指：从指令存储器中读指令（地址：PC）
  - 译码：读出一或两个源寄存器的值（寄存器组）
  - 运算：进行指令规定的运算（ALU）
  - 访存：读/写数据存储器
  - 写回：将结果写入目的寄存器
- 需要保存的值
  - PC、寄存器组、存储器

使用的组合逻辑部件

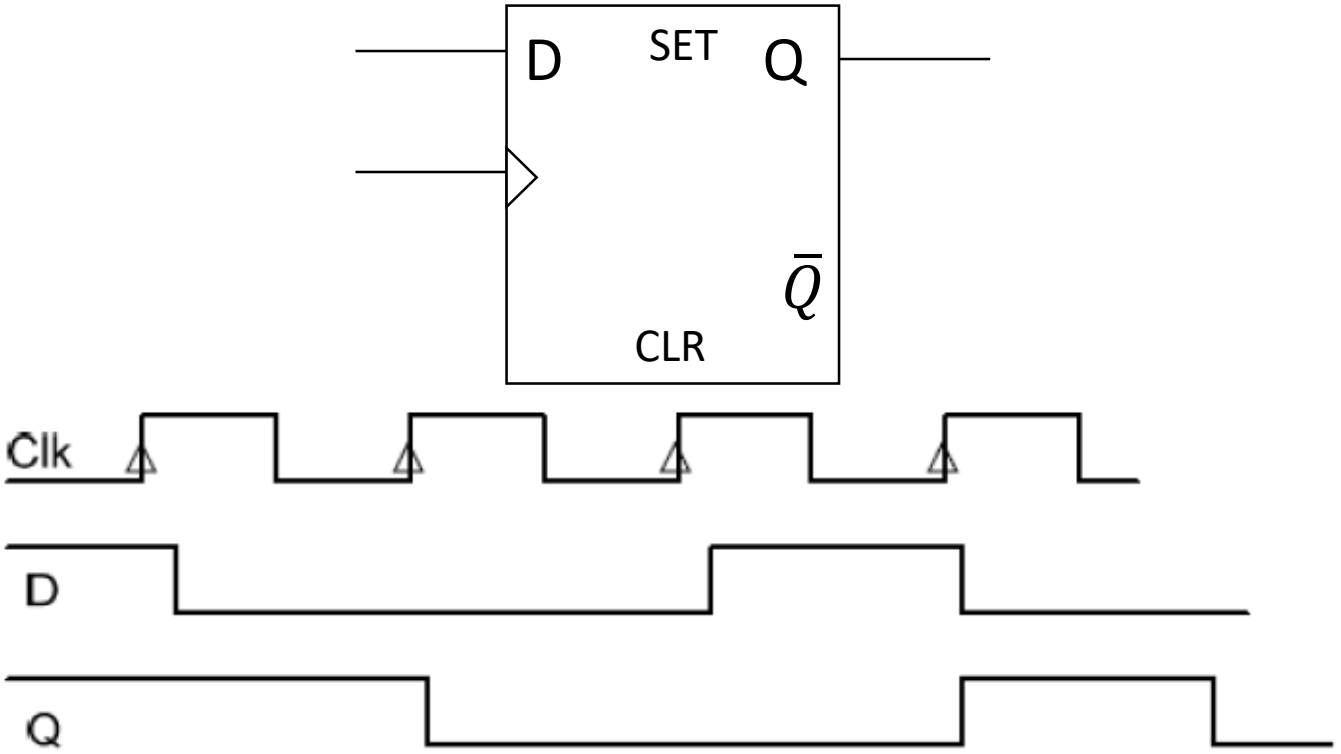




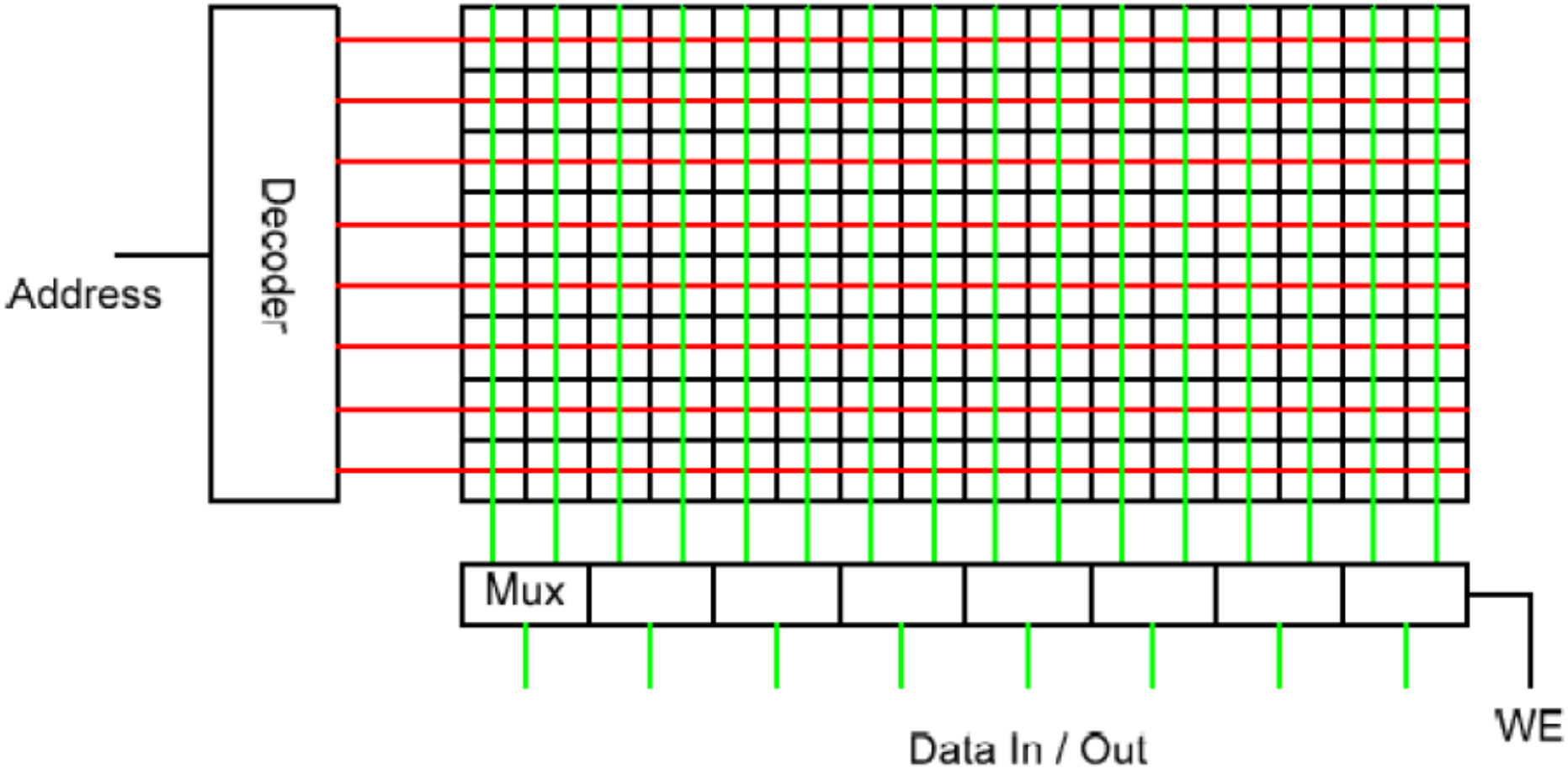


D触发器

- 在时钟的上升沿写入输入数据
- 一直保持到下一个上升沿

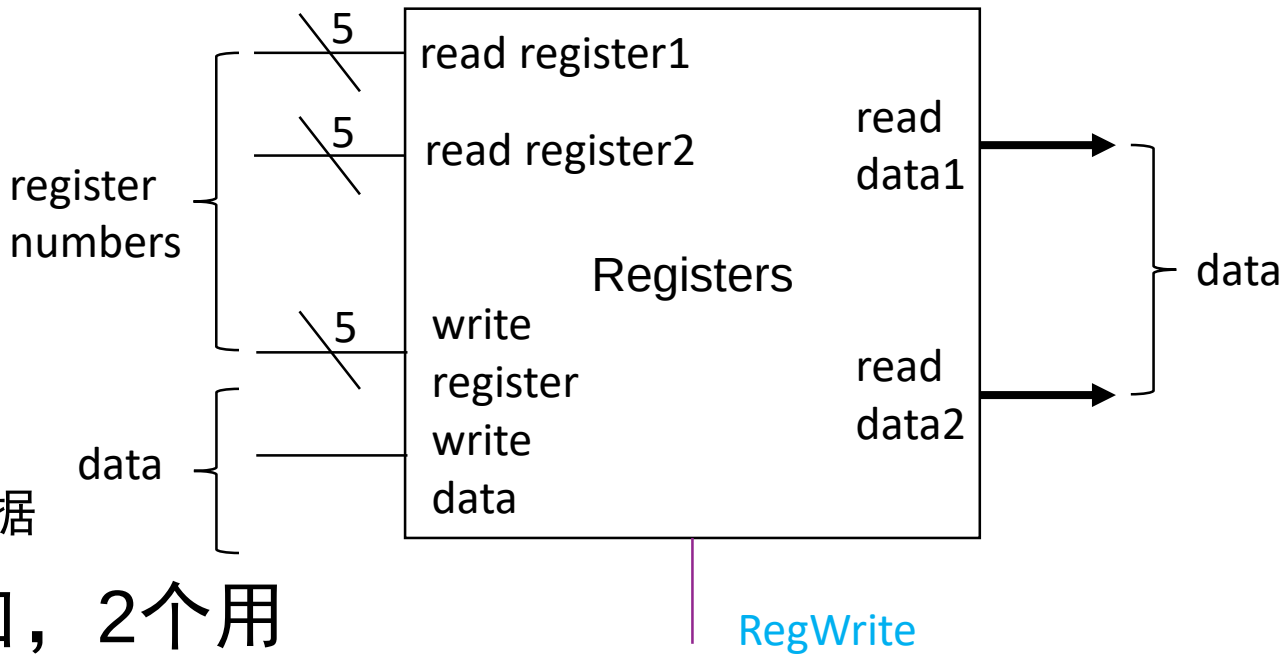


存储器



寄存器组

- 接口简单
  - 输入
    - 地址
    - 写入数据
    - 写信号
  - 输出
    - 读出的数据

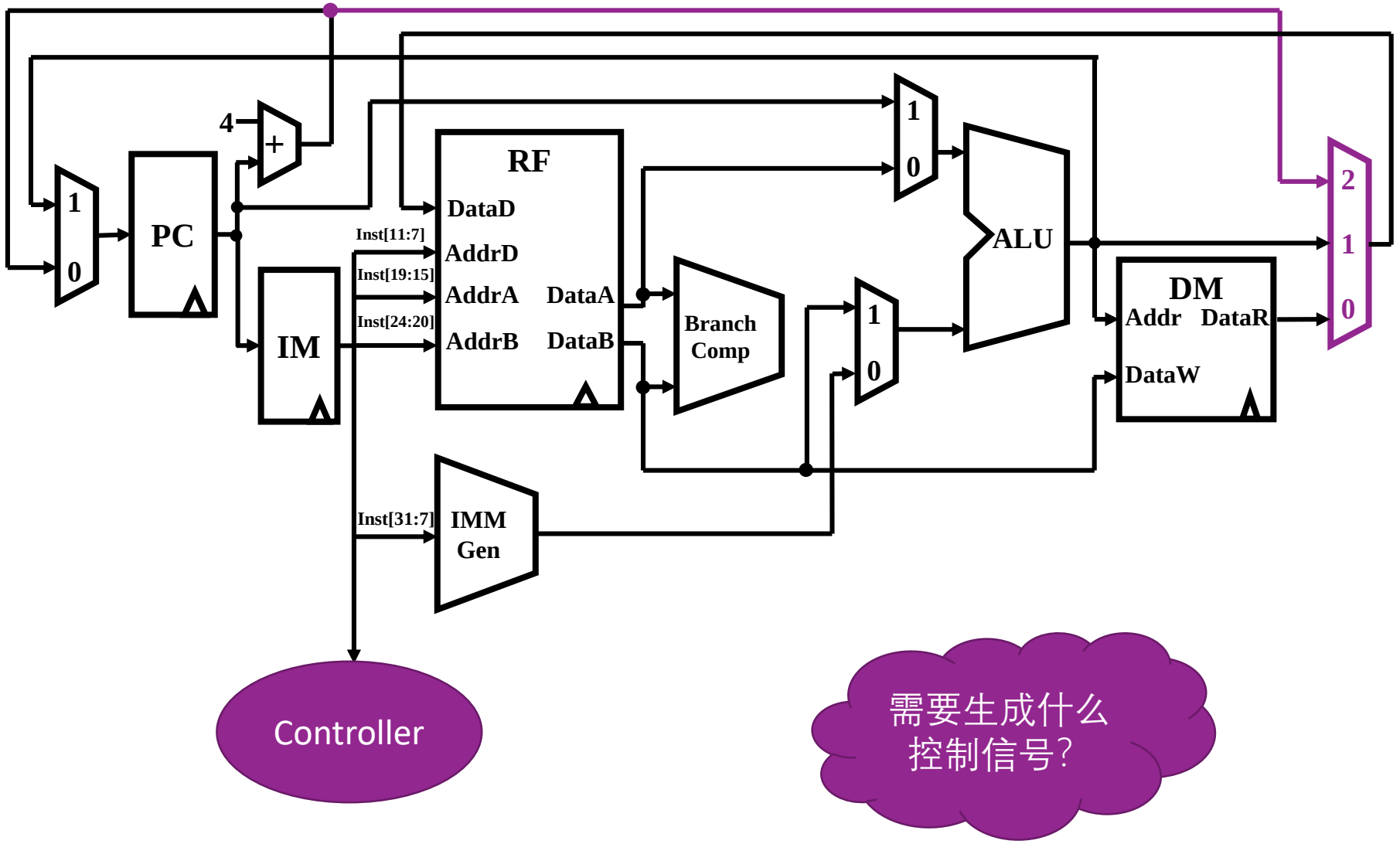


- 3个地址端口，2个用来读，1个用来写
- 每个时钟周期可以完成3次访问

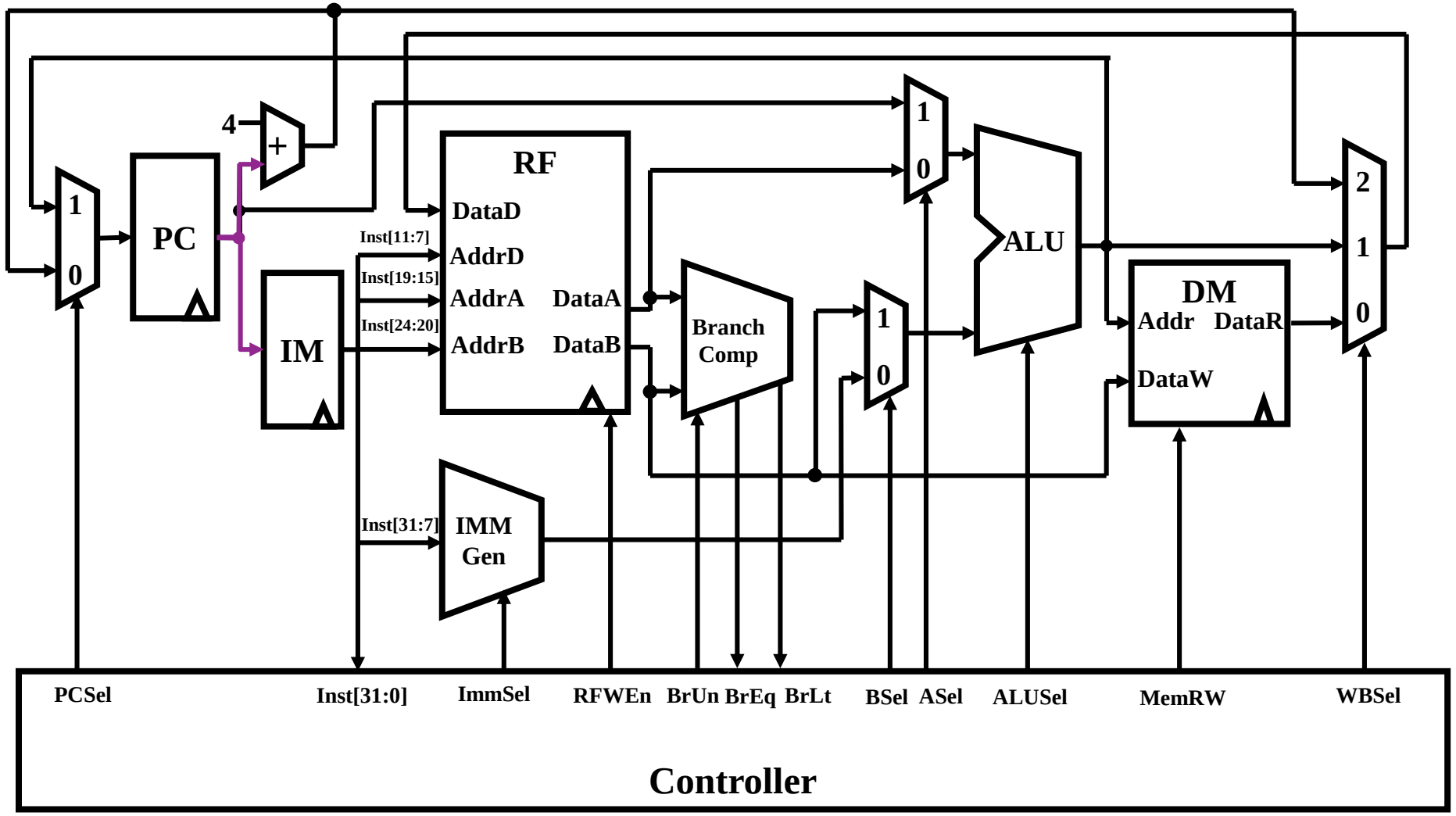
时序设计

- 每条指令占用一个时钟周期
  - 取指令后分析指令，并给出整个执行期间的全部信号
  - 不需要状态信息，在时钟的结束的边沿写入结果
- 控制对象
  - ALU的运算
  - 寄存器组和存储器的写入
  - 多路选通器
- 时钟周期开始时读取指令
  - 与具体指令无关

初步的Datapath

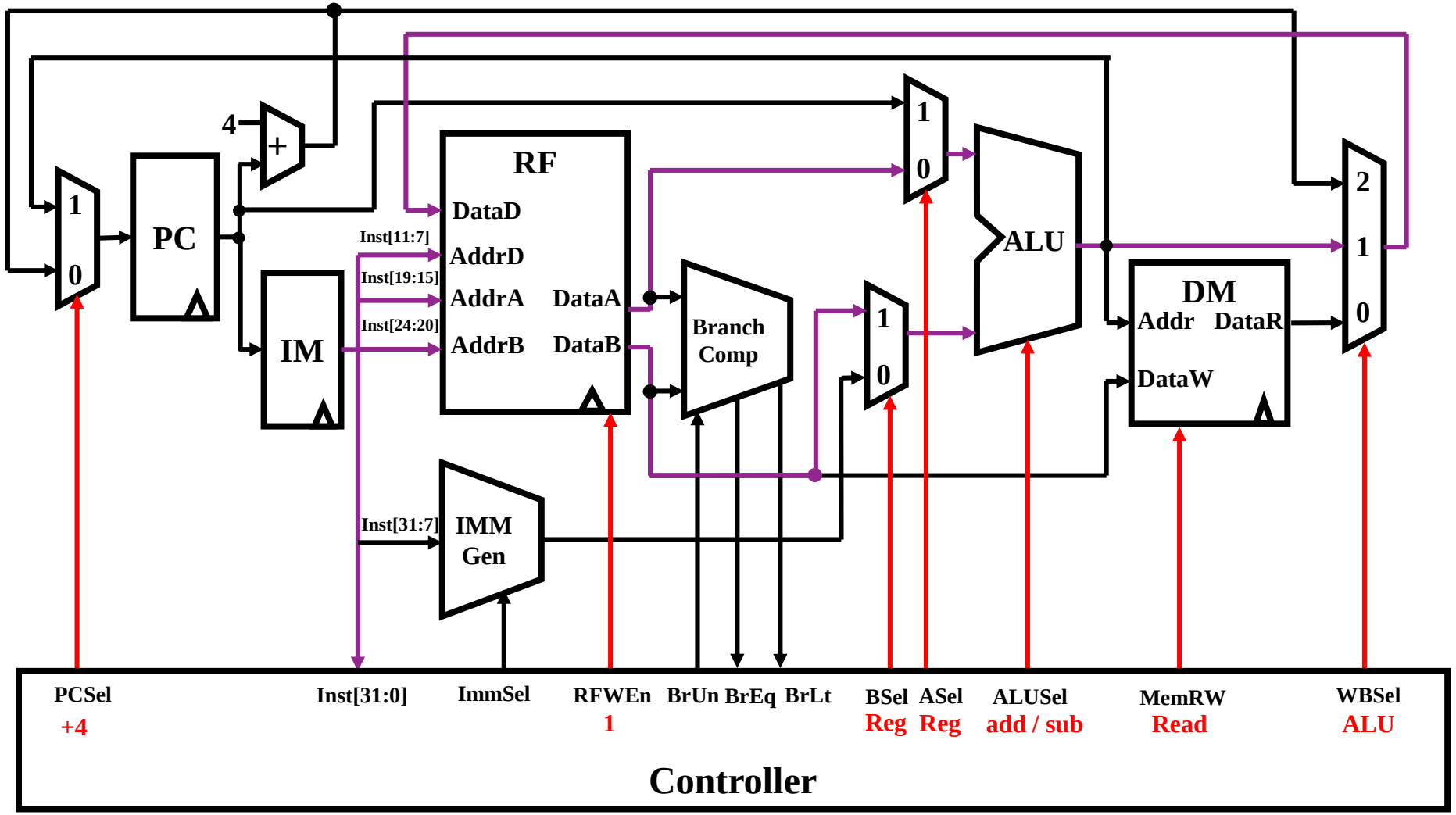


时钟周期开始时的控制


4
+
1
0
PC
IM
RF
DataD
AddrD
AddrA
DataA
AddrB
DataB
Inst[11:7]
Inst[19:15]
Inst[24:20]
Inst[31:7]
IMM
Gen
Branch
Comp
1
0
1
0
ALU
DM
Addr
DataR
DataW
2
1
0
PCSel
Inst[31:0]
ImmSel
RFWEn
BrUn
BrEq
BrLt
BSel
ASel
ALUSel
MemRW
WBSel
Controller

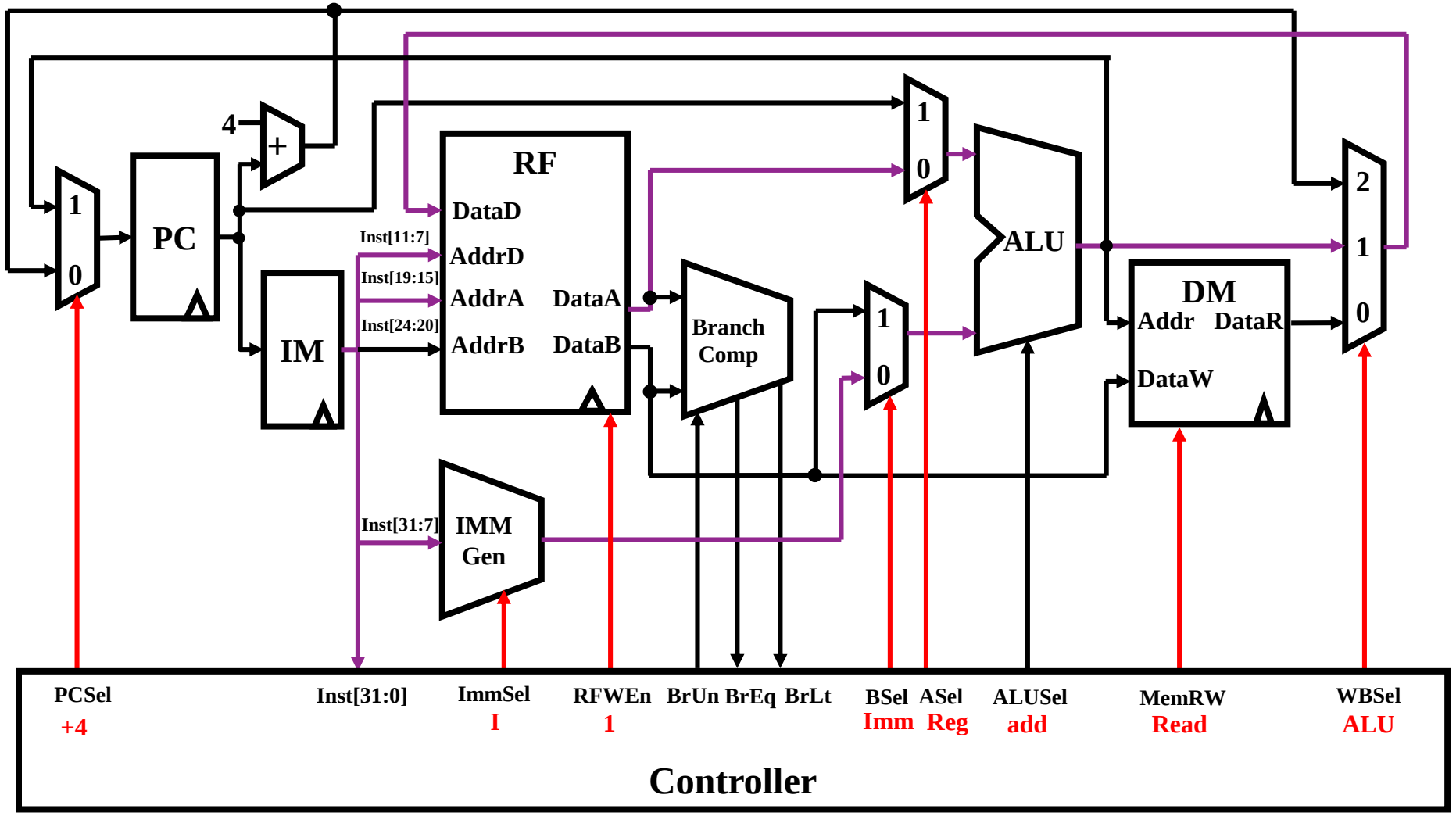
算术逻辑运算指令的控制

```
Add rd r1 r2
Sub rd r1 r2
```



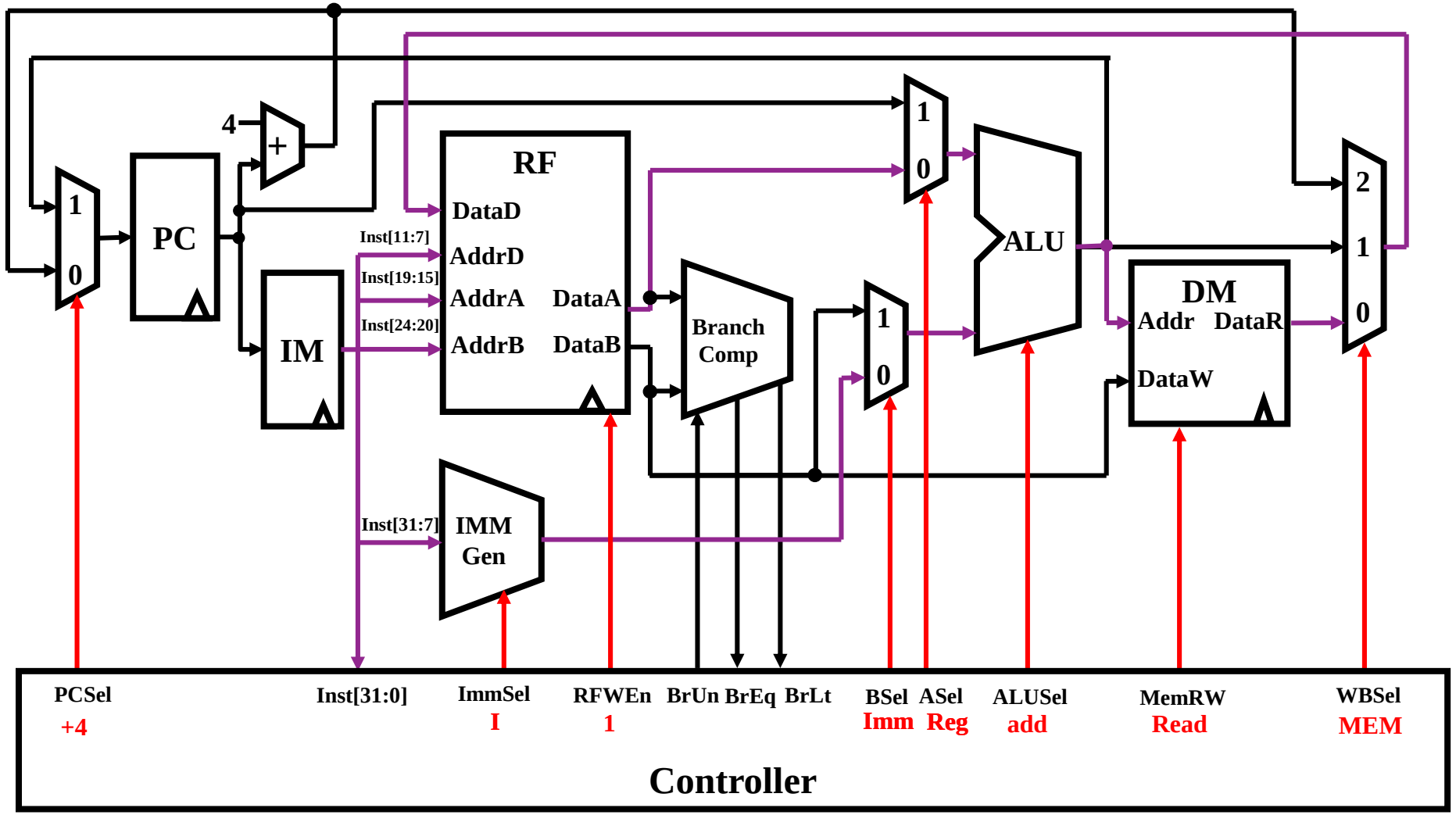
算术逻辑运算指令的控制

```
Addi rd r1 imm
```



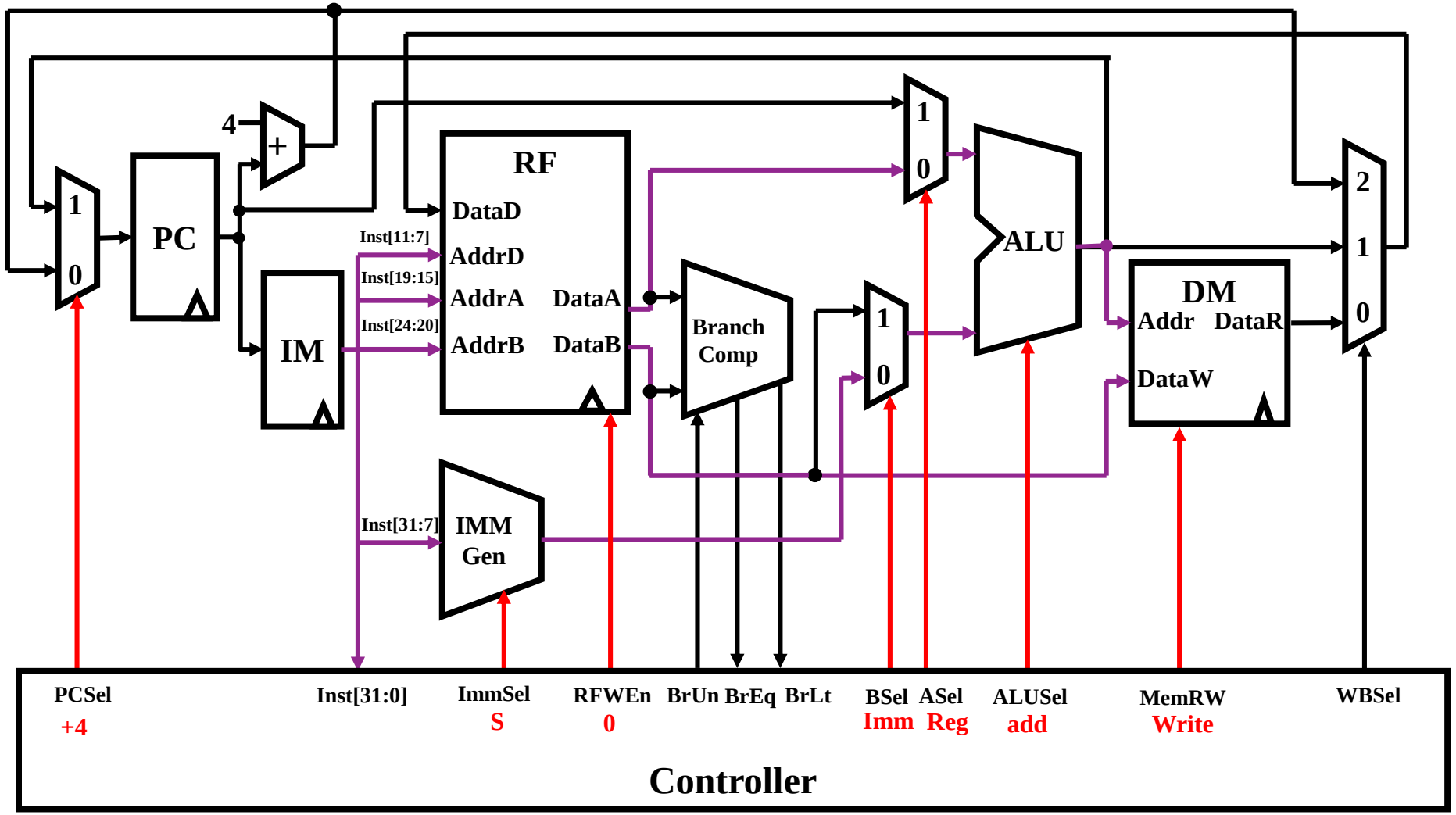
Load指令

```
lw rd, rs1, imm
```



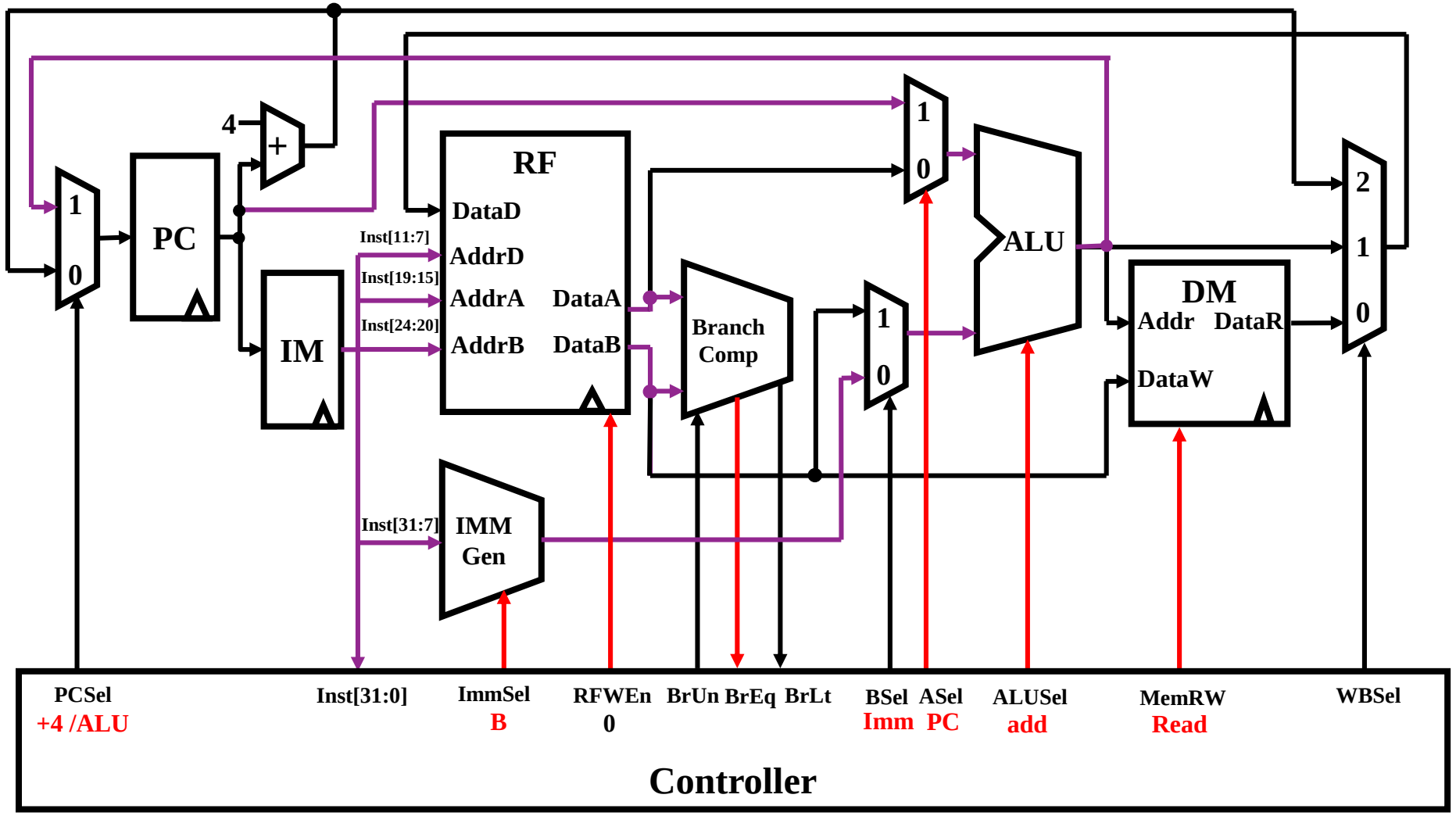
Store指令

```
sw rs2, rs1, imm
```



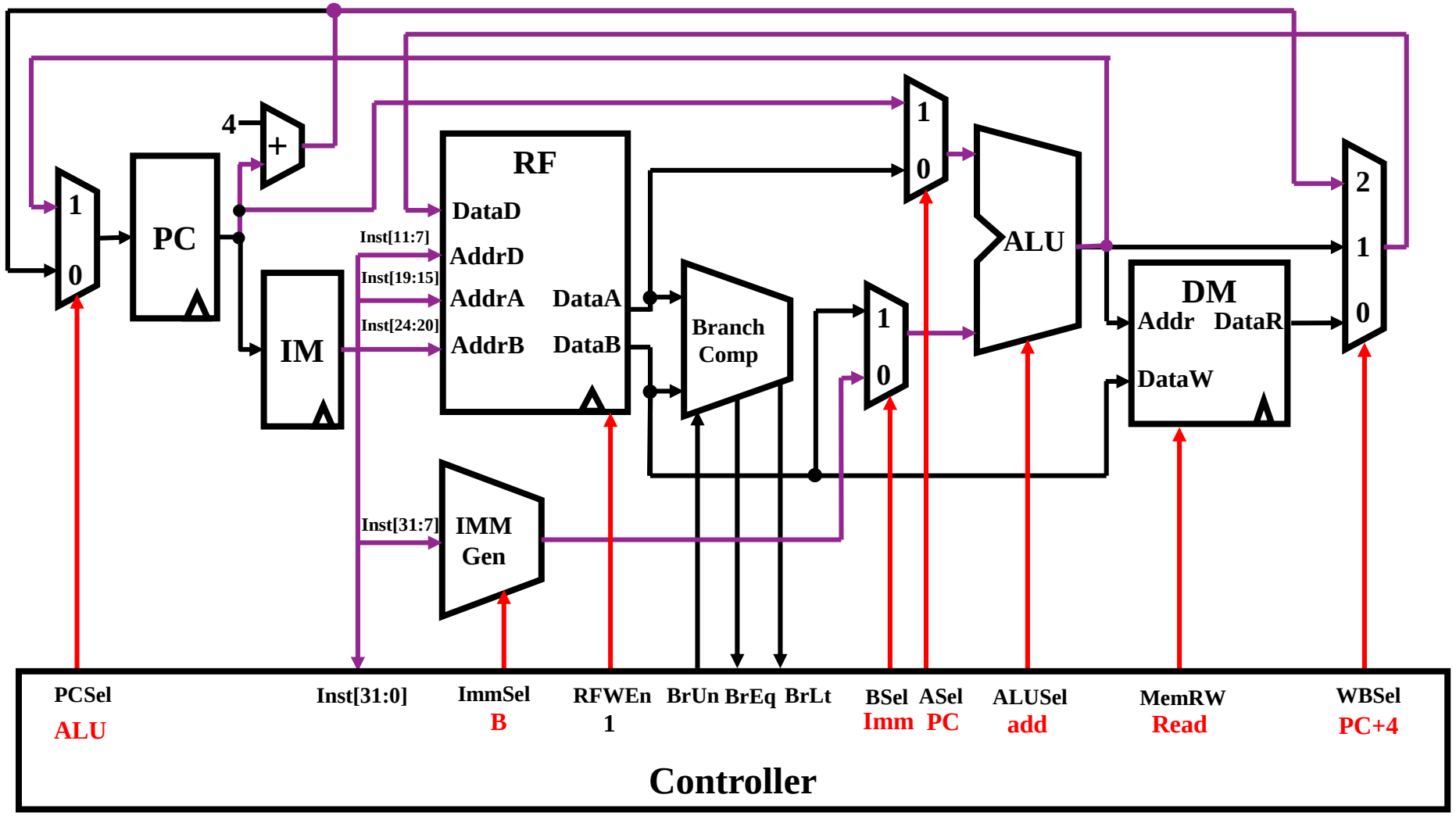
Beq指令

beq rs1, rs2, label



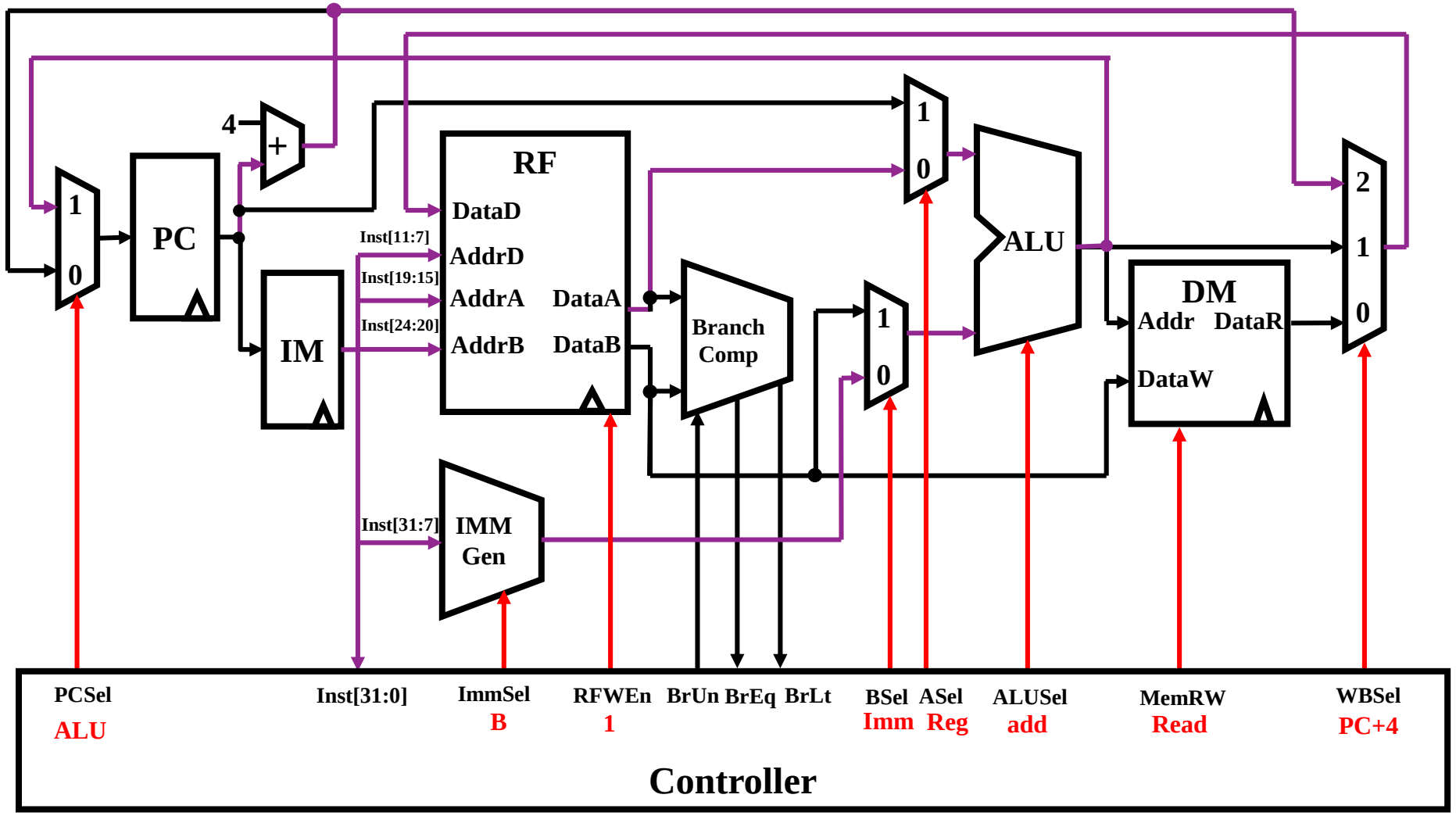
Jal指令

```
jal rd, imm
```



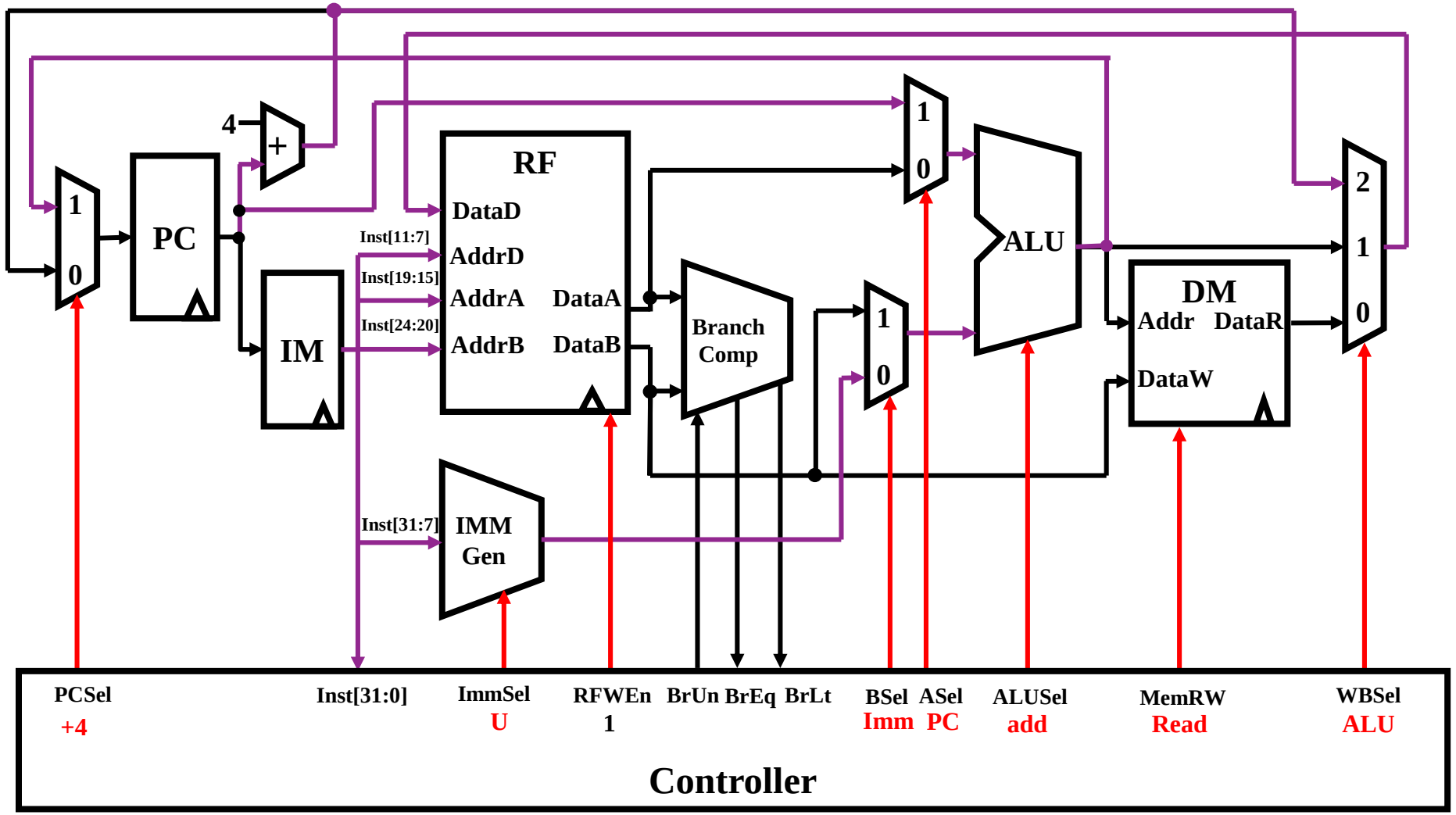
Jalr指令

```
jalr rd, rs, imm
```



auipc指令

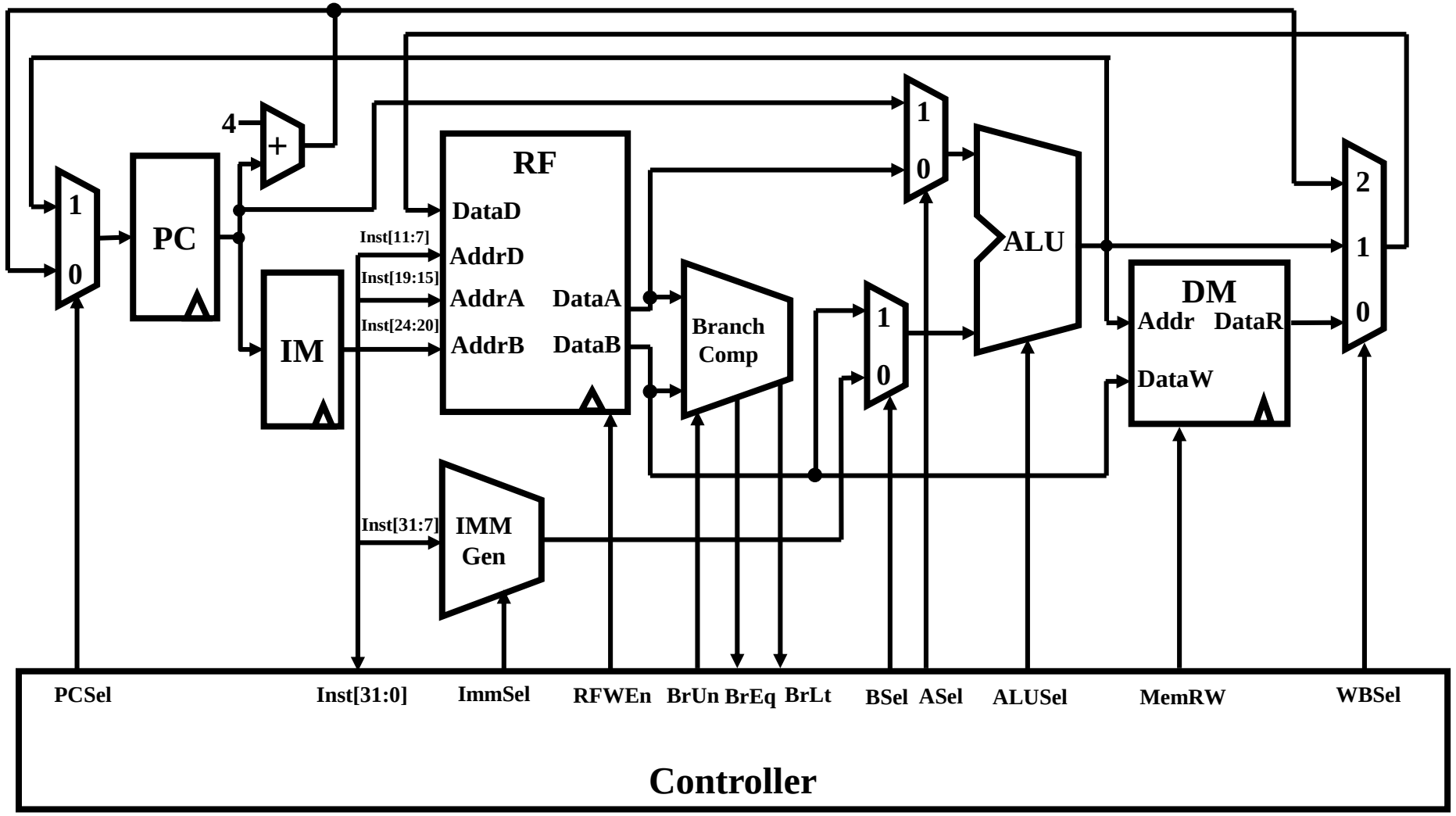
```
auipc rd, imm
```



信号整理

inst[31:0]	BrEq	BrLT	PCSel	ImmSel	BrUn	ASel	BSel	ALUSel	MemRW	RegWEn	WBSel
add	-	-	+4	-	-	Reg	Reg	Add	Read	1	ALU
sub	-	-	+4	-	-	Reg	Reg	Sub	Read	1	ALU
(R-R Op)	-	-	+4	-	-	Reg	Reg	(Op)	Read	1	ALU
addi	-	-	+4	I	-	Reg	Imm	Add	Read	1	ALU
lw	-	-	+4	I	-	Reg	Imm	Add	Read	1	Mem
sw	-	-	+4	S	-	Reg	Imm	Add	Write	0	-
beq	0	-	+4	B	-	PC	Imm	Add	Read	0	-
	1	-	ALU	B	-	PC	Imm	Add	Read	0	-
bne	0	-	ALU	B	-	PC	Imm	Add	Read	0	-
	1	-	+4	B	-	PC	Imm	Add	Read	0	-
blt	-	1	ALU	B	0	PC	Imm	Add	Read	0	-
bltu	-	1	ALU	B	1	PC	Imm	Add	Read	0	-
jalr	-	-	ALU	I	-	Reg	Imm	Add	Read	1	PC+4
jal	-	-	ALU	J	-	PC	Imm	Add	Read	1	PC+4
auipc	-	-	+4	U	-	PC	Imm	Add	Read	1	ALU

完整的单周期CPU


4
+
1
0
PC
IM
RF
DataD
AddrD
AddrA
DataA
AddrB
DataB
Inst[11:7]
Inst[19:15]
Inst[24:20]
Inst[31:7]
IMM
Gen
Branch
Comp
1
0
1
0
ALU
DM
Addr
DataR
DataW
2
1
0
PCSel
Inst[31:0]
ImmSel
RFWEn
BrUn
BrEq
BrLt
BSel
ASel
ALUSel
MemRW
WBSel
Controller

单周期CPU特点

- 优点
  - 每条指令占用一个时钟周期
  - 逻辑设计简单，时序设计也简单
- 缺点
  - 各组成部件的利用率不高
    - 维持有效信号
  - 时钟周期应满足执行时间最长指令的要求
    - Load指令
- CPI =1

小结

- 单周期CPU设计
  - 全部控制信号
  - 无需状态信息
  - 控制信号生成：组合逻辑电路
    - 输入：inst, breq, brlt
    - 输出：完成指令功能所需要的全部控制信号

阅读和思考

- 阅读
- 思考
  - 单周期CPU设计中是否存在什么不足？
  - 可以从哪几个方面对其进行改进？
- 实践
  - 设计能实现ThinPAD RISC-V指令系统的数据通路(单周期)，并列出各硬件模块所需要的控制信号。

谢谢

inst[31:0]	BrEq	BrLT	PCSel	ImmSel	BrUn	ASel	BSel	ALUSel	MemRW	RegWEn	WBSel
add	-	-	+4	-	-	Reg	Reg	Add	Read	1	ALU
sub	-	-	+4	-	-	Reg	Reg	Sub	Read	1	ALU
(R-R Op)	-	-	+4	-	-	Reg	Reg	(Op)	Read	1	ALU
addi	-	-	+4	I	-	Reg	Imm	Add	Read	1	ALU
lw	-	-	+4	I	-	Reg	Imm	Add	Read	1	Mem
sw	-	-	+4	S	-	Reg	Imm	Add	Write	0	-
beq	0	-	+4	B	-	PC	Imm	Add	Read	0	-
	1	-	ALU	B	-	PC	Imm	Add	Read	0	-
bne	0	-	ALU	B	-	PC	Imm	Add	Read	0	-
	1	-	+4	B	-	PC	Imm	Add	Read	0	-
blt	-	1	ALU	B	0	PC	Imm	Add	Read	0	-
bltu	-	1	ALU	B	1	PC	Imm	Add	Read	0	-
jalr	-	-	ALU	I	-	Reg	Imm	Add	Read	1	PC+4
jal	-	-	ALU	J	-	PC	Imm	Add	Read	1	PC+4
auipc	-	-	+4	U	-	PC	Imm	Add	Read	1	ALU