

计算机组成原理

实验五 流水线CPU设计

2022春季

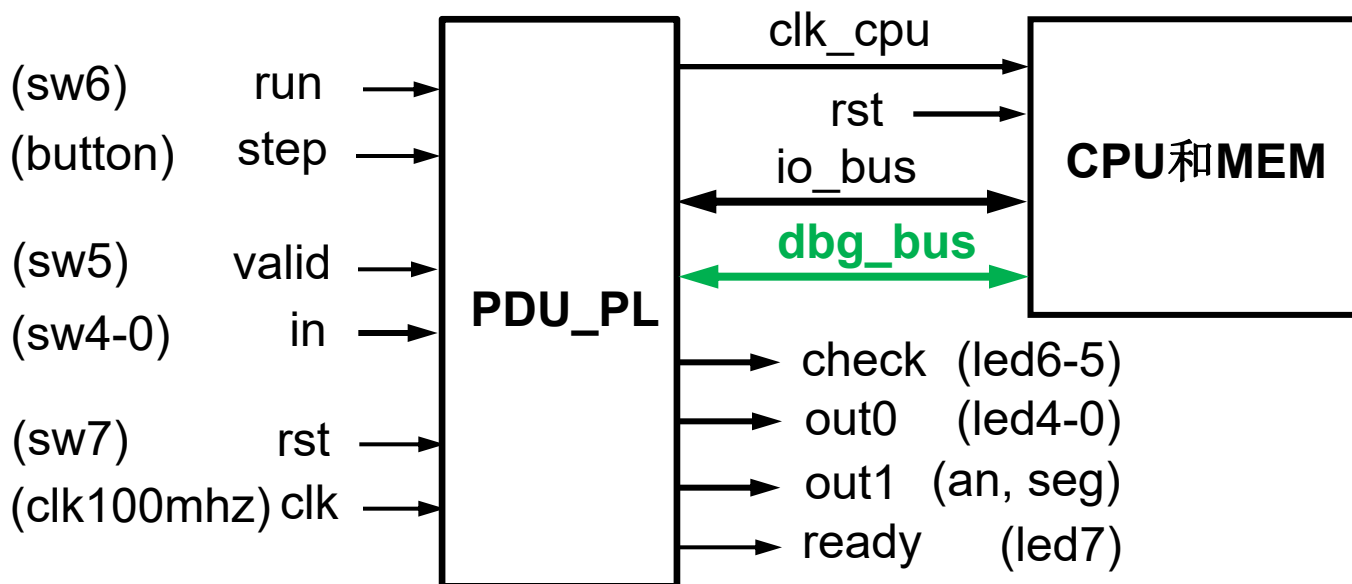
zjx@ustc.edu.cn

实验目标

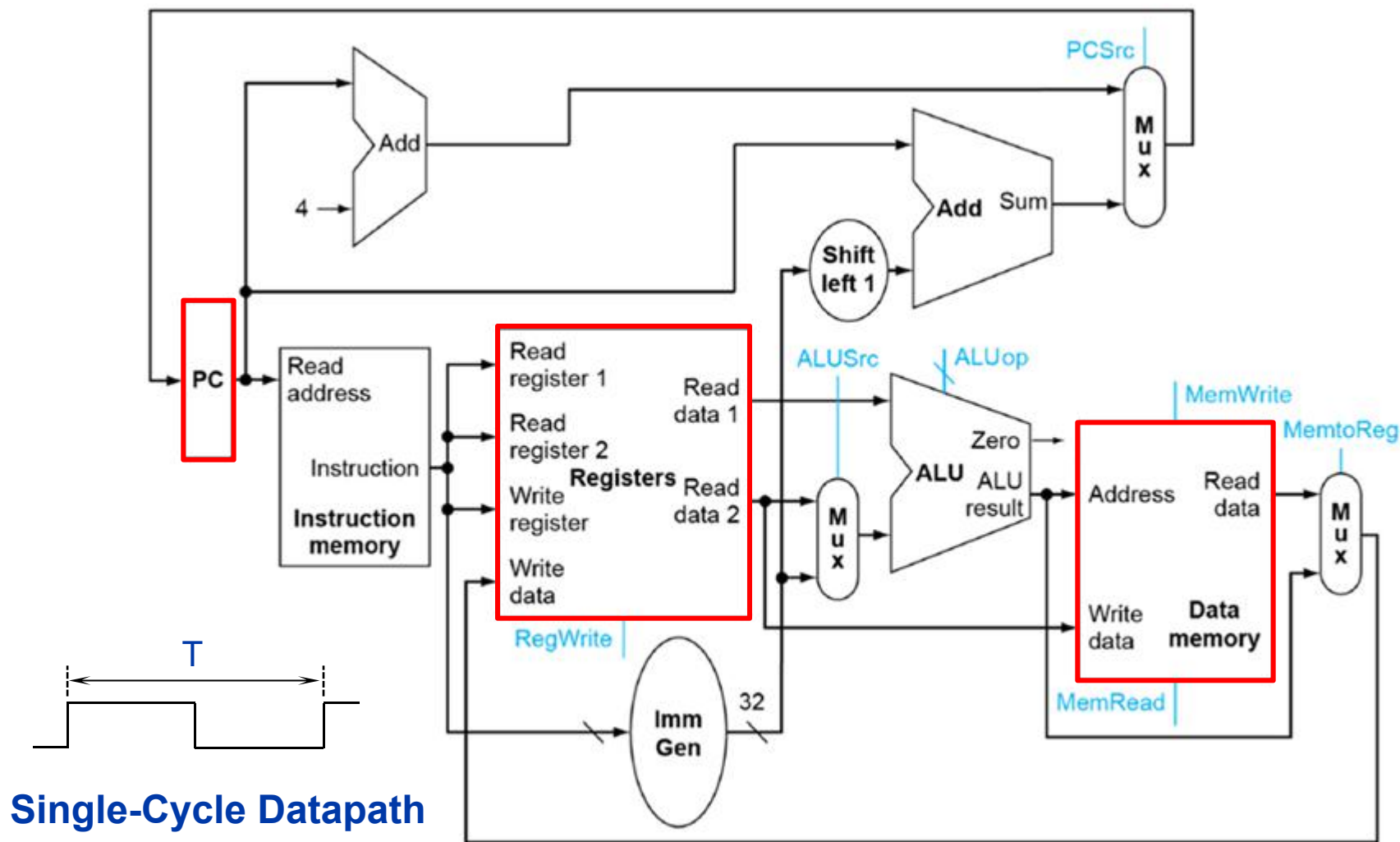
- 理解流水线CPU的结构和工作原理
- 掌握流水线CPU的设计和调试方法，特别是流水线中数据相关和控制相关的处理
- 熟练掌握数据通路和控制器的设计和描述方法

实验内容

- 设计实现5级流水线的RISC-V CPU
 - 能够执行6条指令：**add, addi, lw, sw, beq, jal**
 - 指令存储器和数据存储器均使用分布式存储器，容量均为256x32位，数据存储器地址为0x0000_0000 ~ 0x0000_2ffff，指令存储器地址为0x0000_3000 ~ 0x0000_3ffc。
 - 寄存器堆和数据存储器均增加一个读端口用于调试

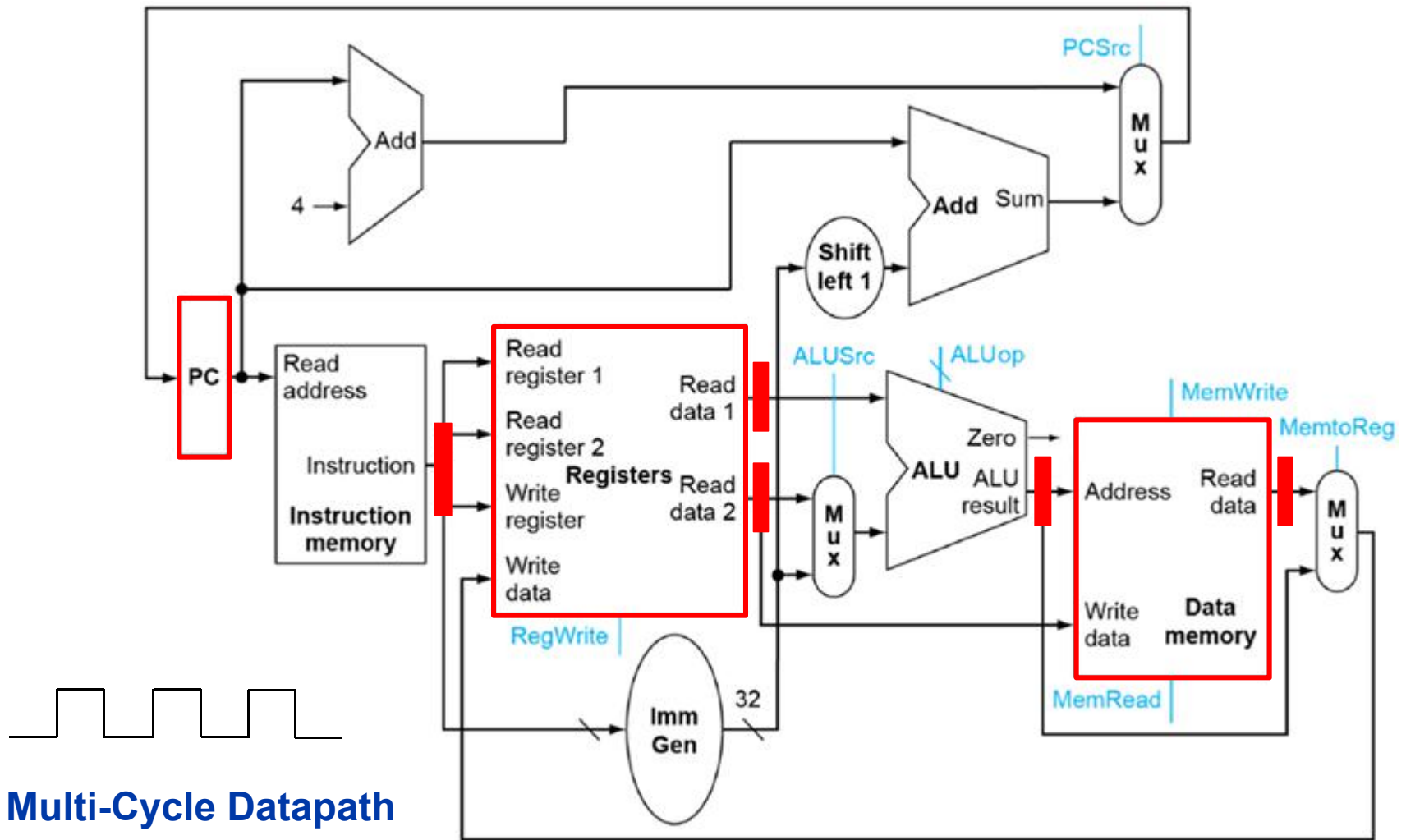


单周期CPU数据通路



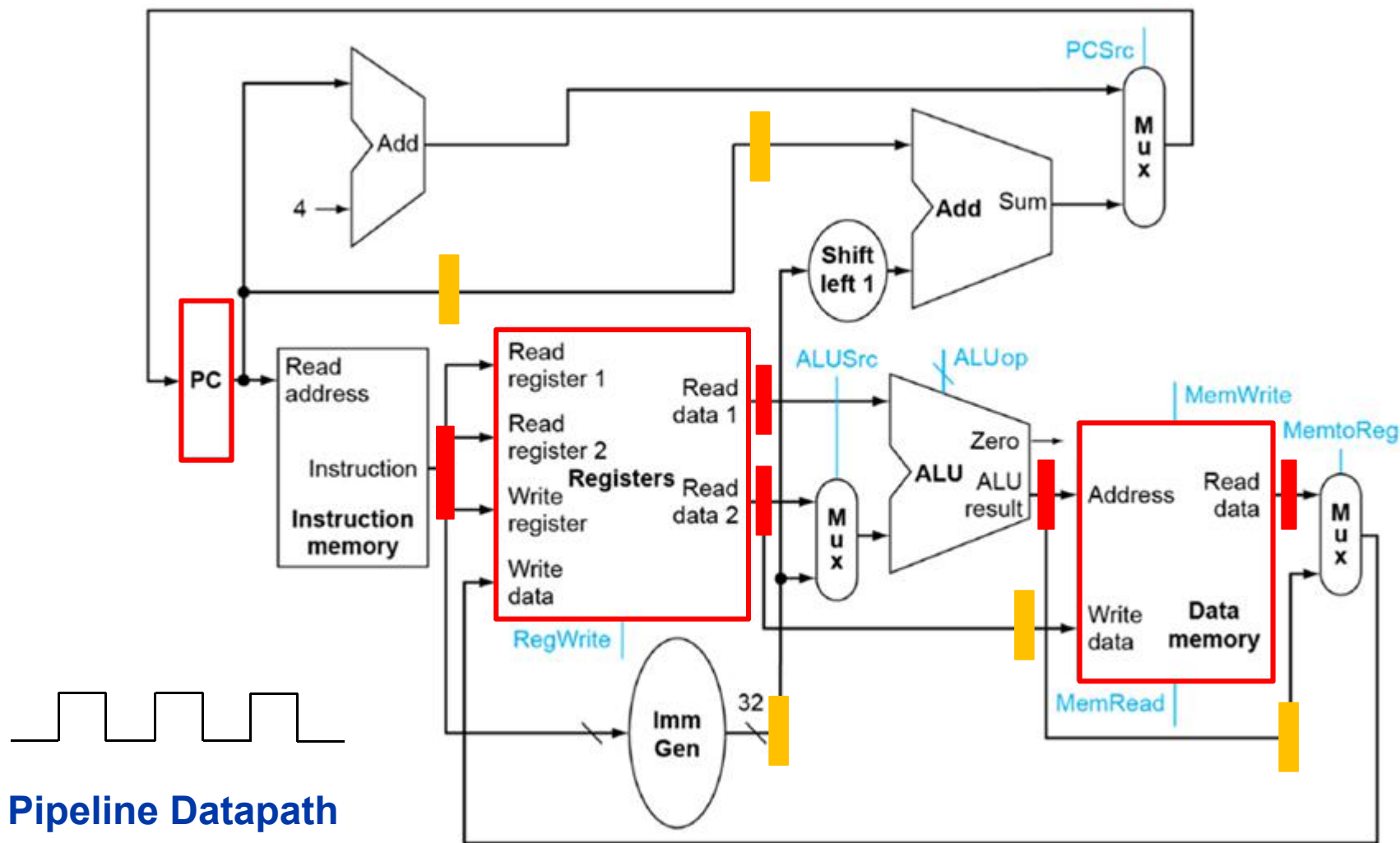
Single-Cycle Datapath

多周期CPU数据通路

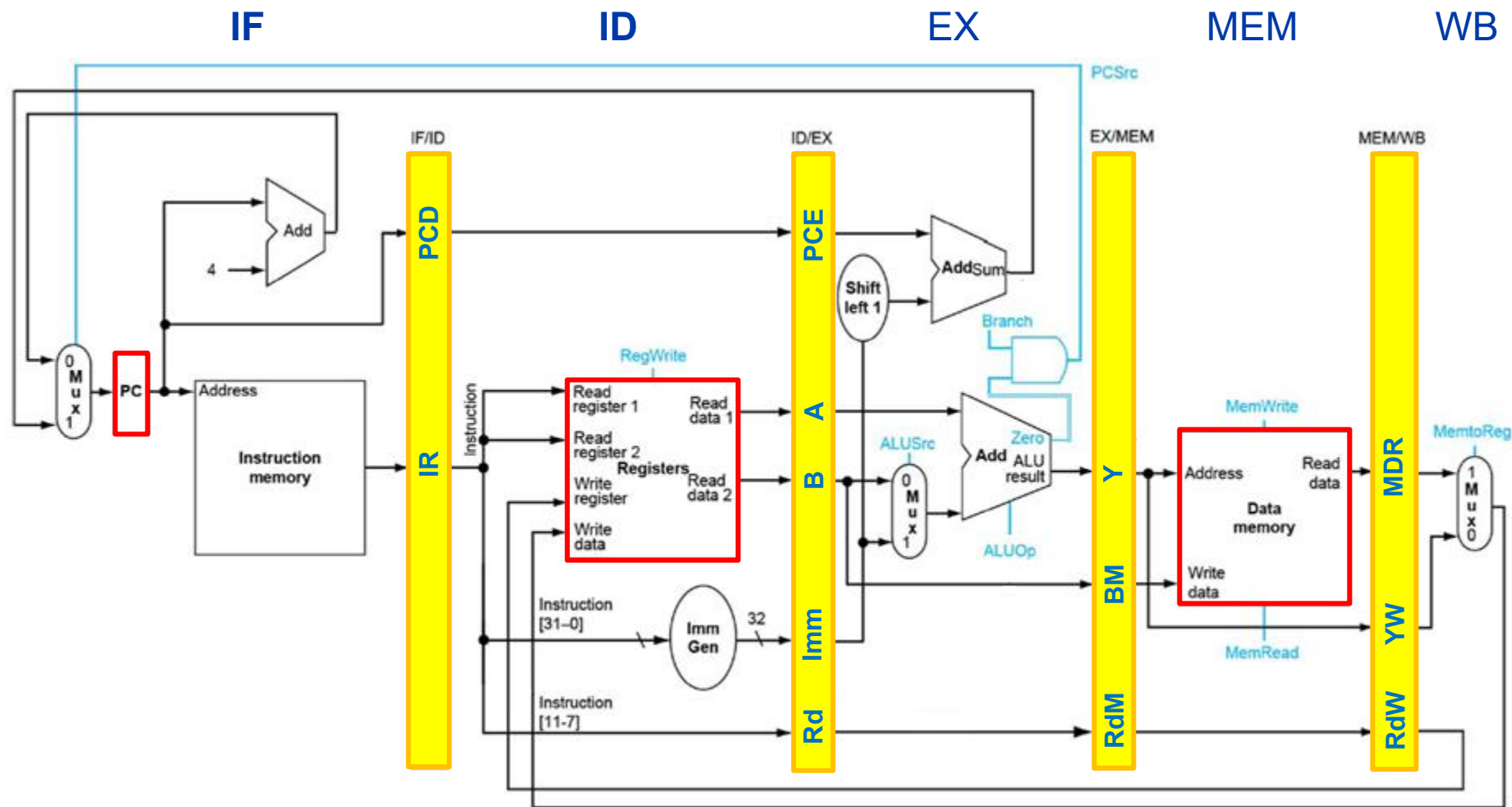


Multi-Cycle Datapath

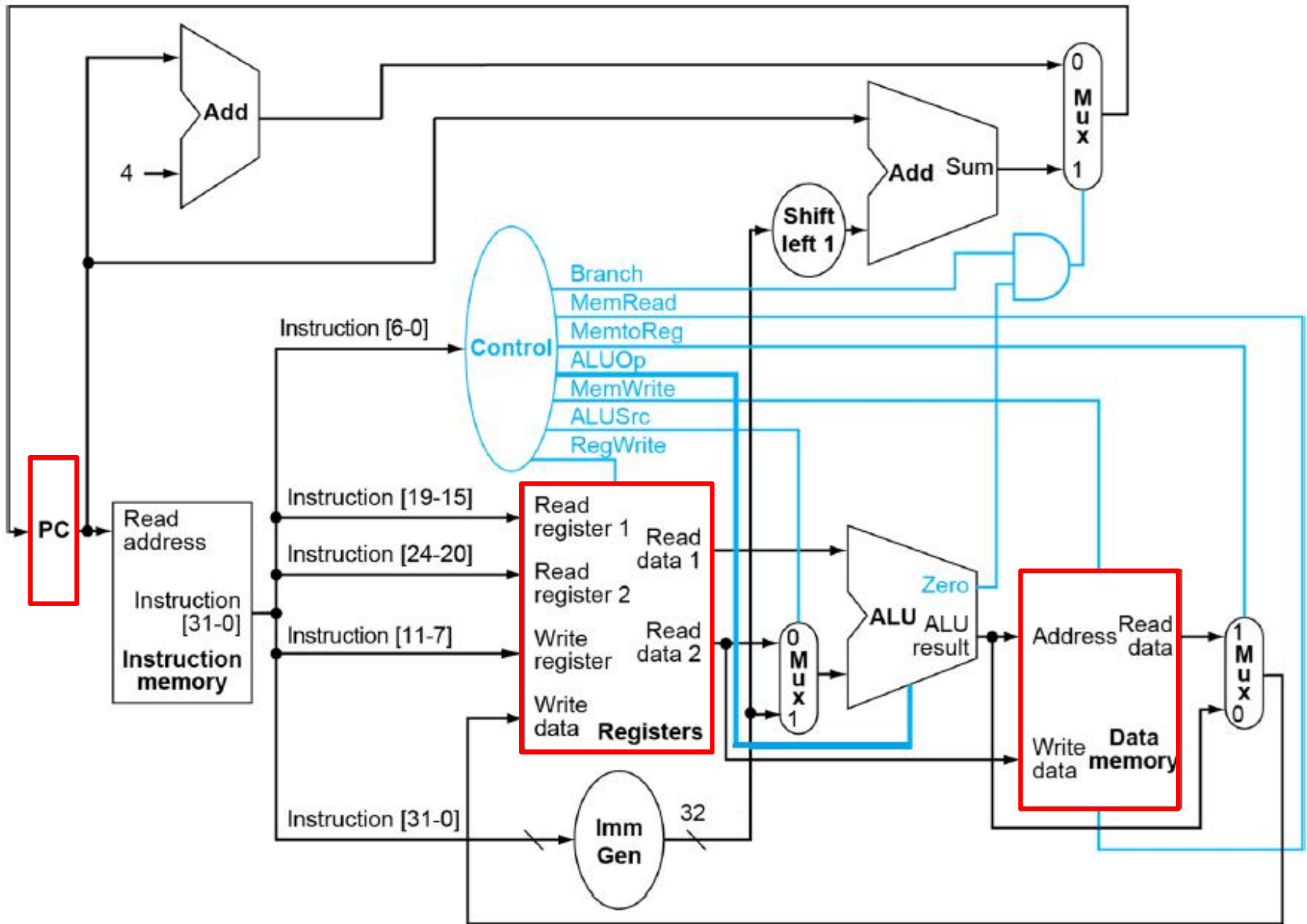
流水线CPU数据通路



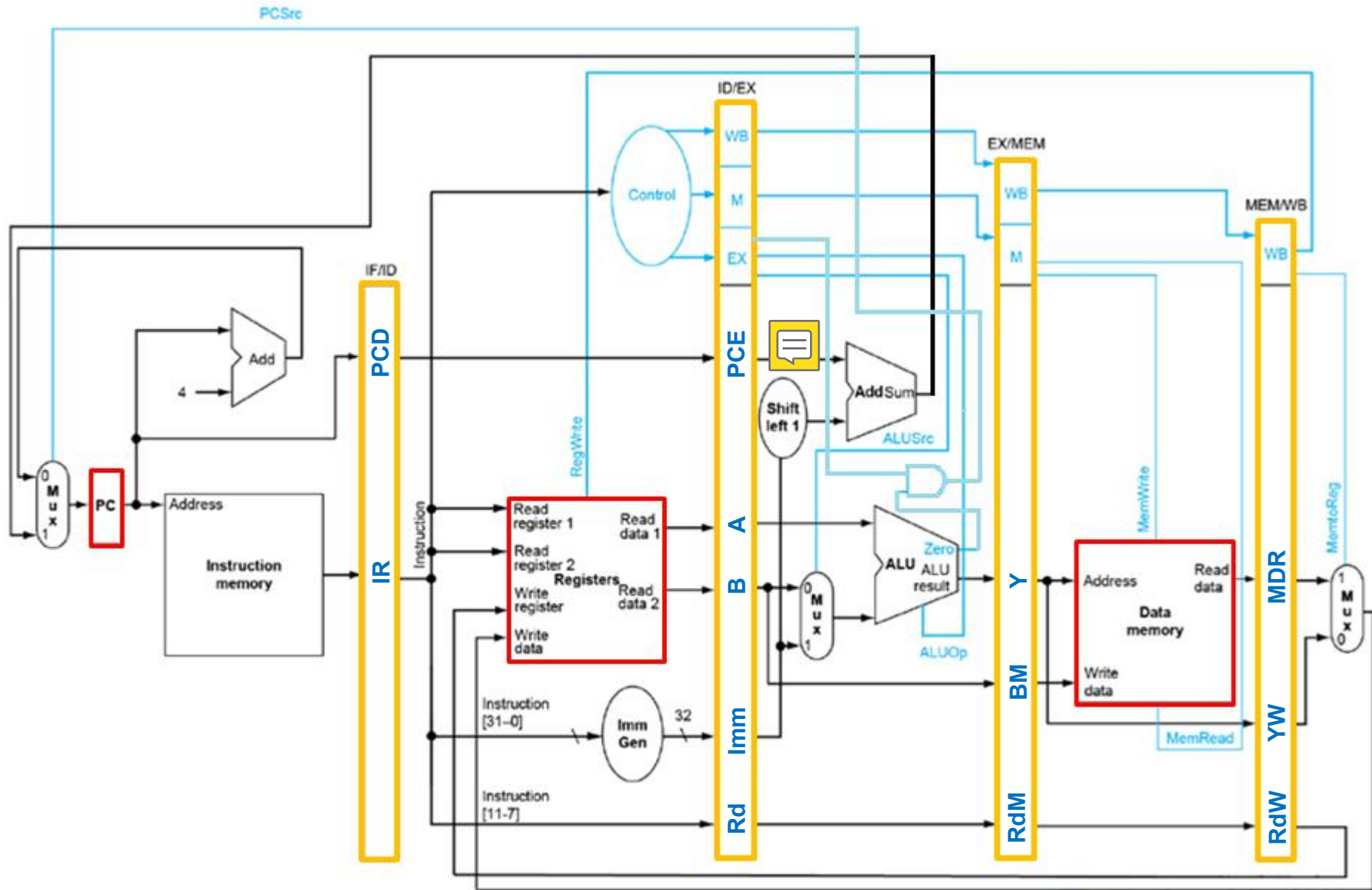
流水线CPU数据通路



单周期CPU数据通路+控制器



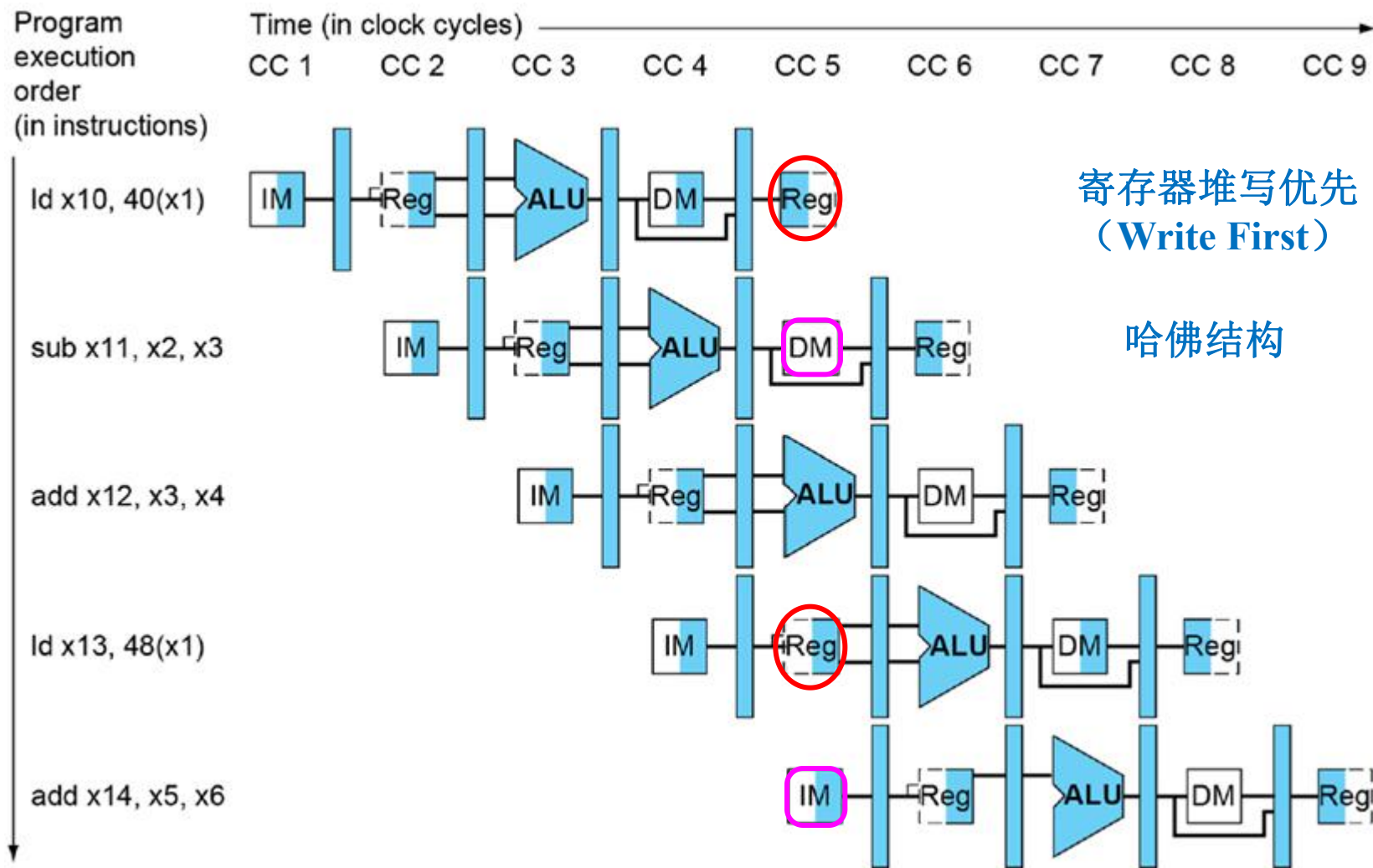
流水线CPU数据通路+控制器



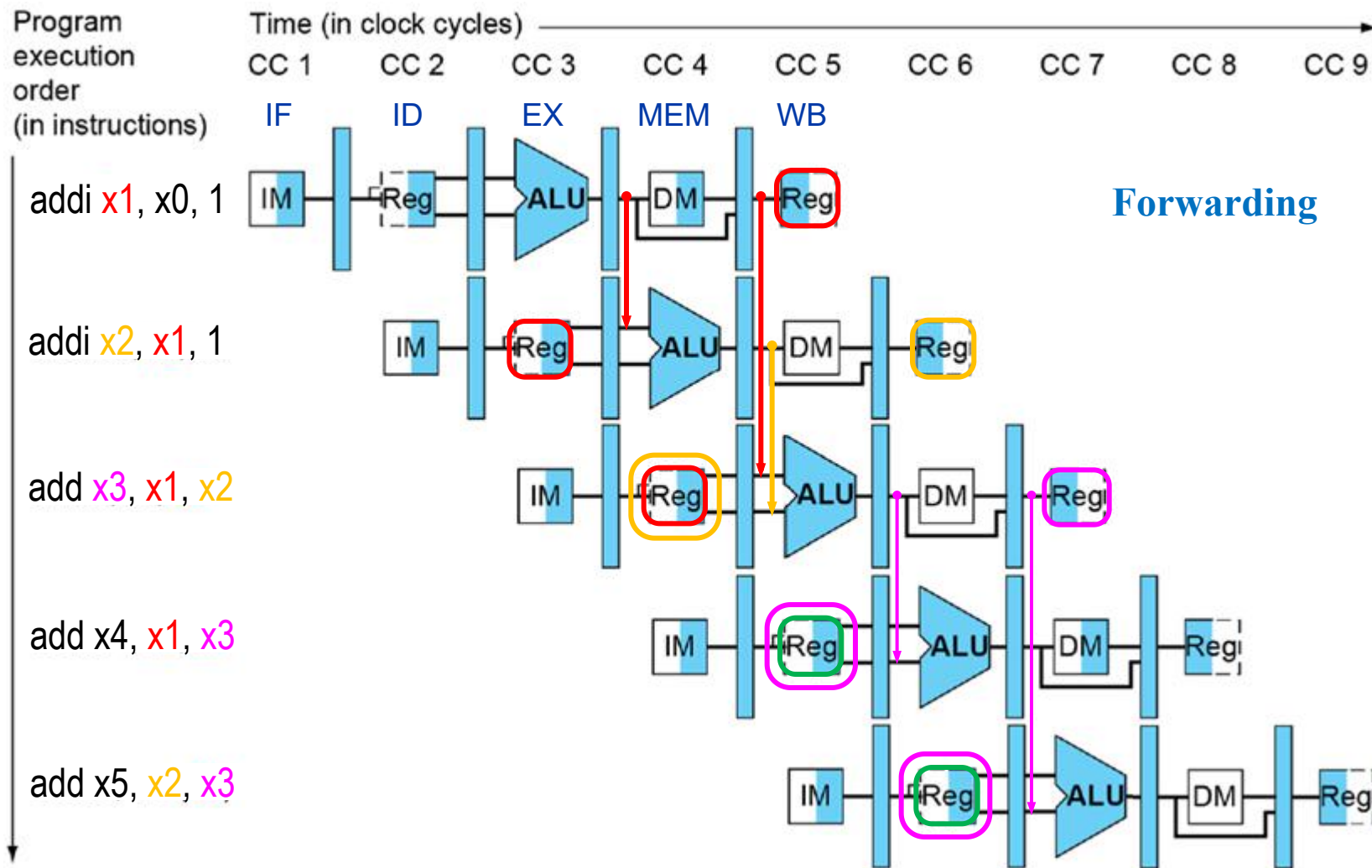
流水线相关及其处理

- **结构相关：当多条指令执行时竞争使用同一资源时**
 - 存储器相关处理：哈佛结构（指令和数据存储器分开）
 - 寄存器堆相关处理：同一寄存器读写时，写优先（Write First）
- **数据相关：当一条指令需要等待前面指令的执行结果时**
 - 数据定向（Forwarding）：将执行结果提前传递至之前流水段
 - 加载-使用相关（Load-use hazard）：阻止紧随Load已进入流水线的指令流动（Stall），向后续流水段插入空操作（Bubble）
- **控制相关：当遇到转移指令且不能继续顺序执行时**
 - 清除（Flush）紧随转移指令已进入流水线的指令
 - 从转移目标处取指令后执行

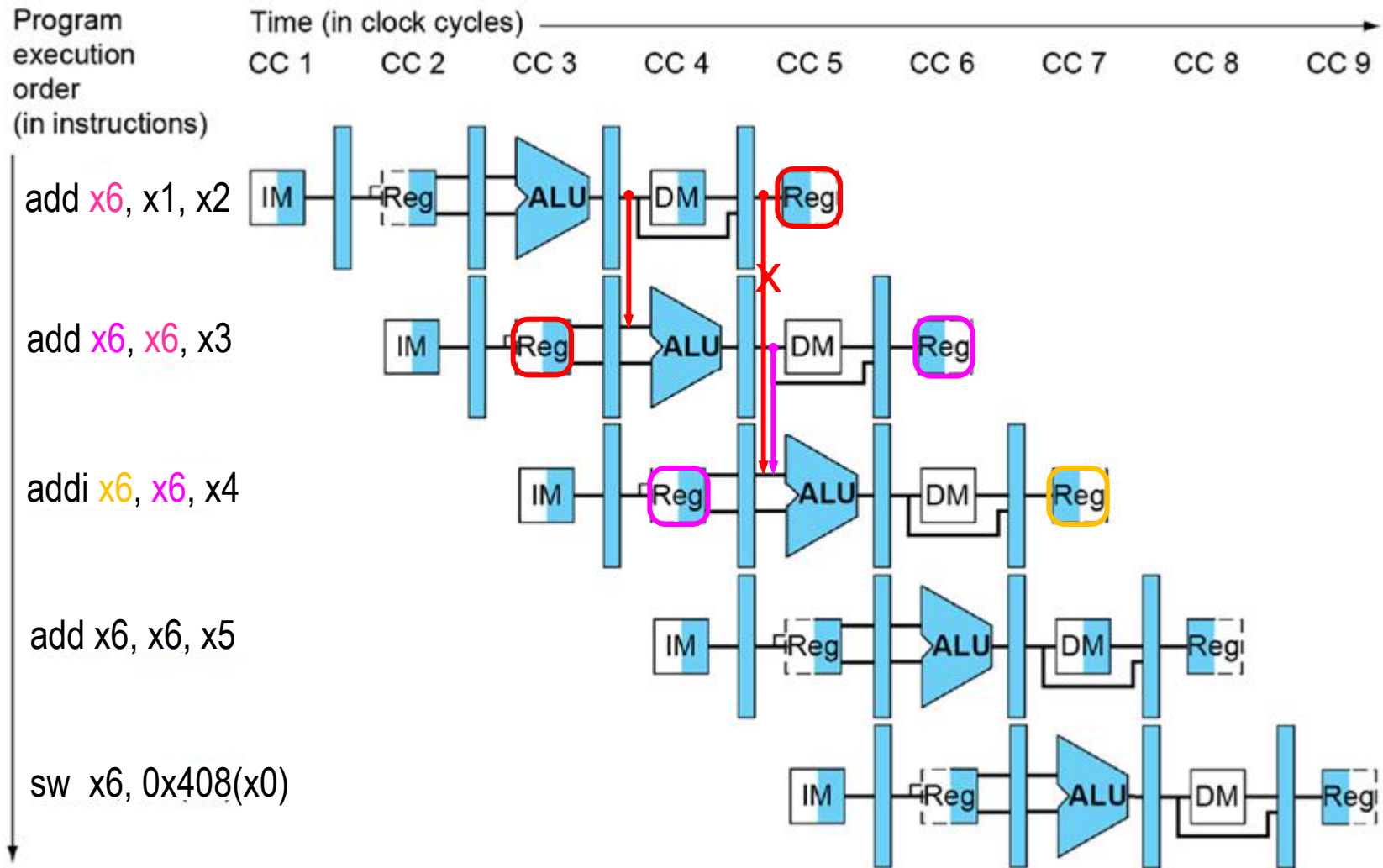
流水线相关：结构相关



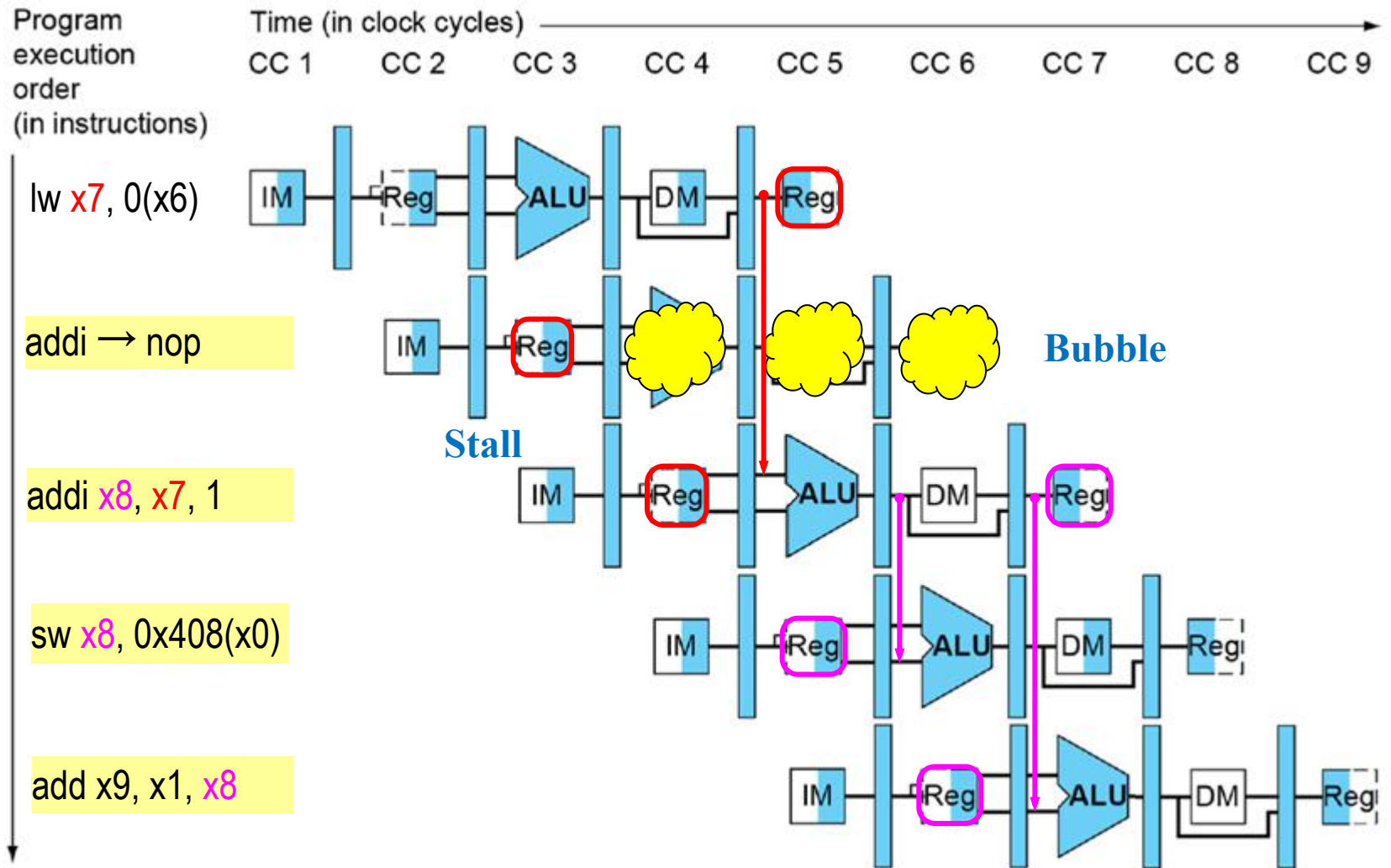
流水线相关：数据相关



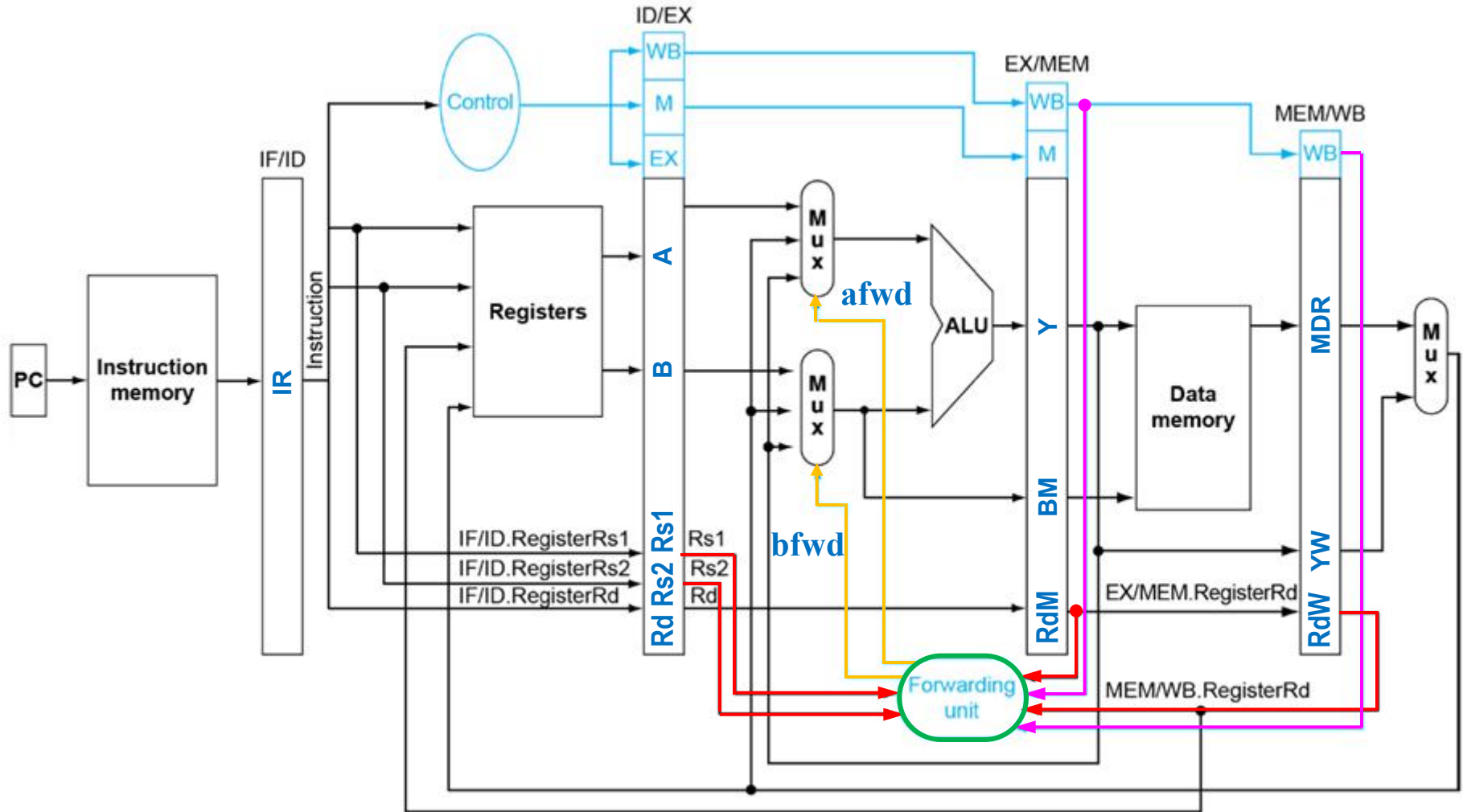
流水线相关：数据相关 (续1)



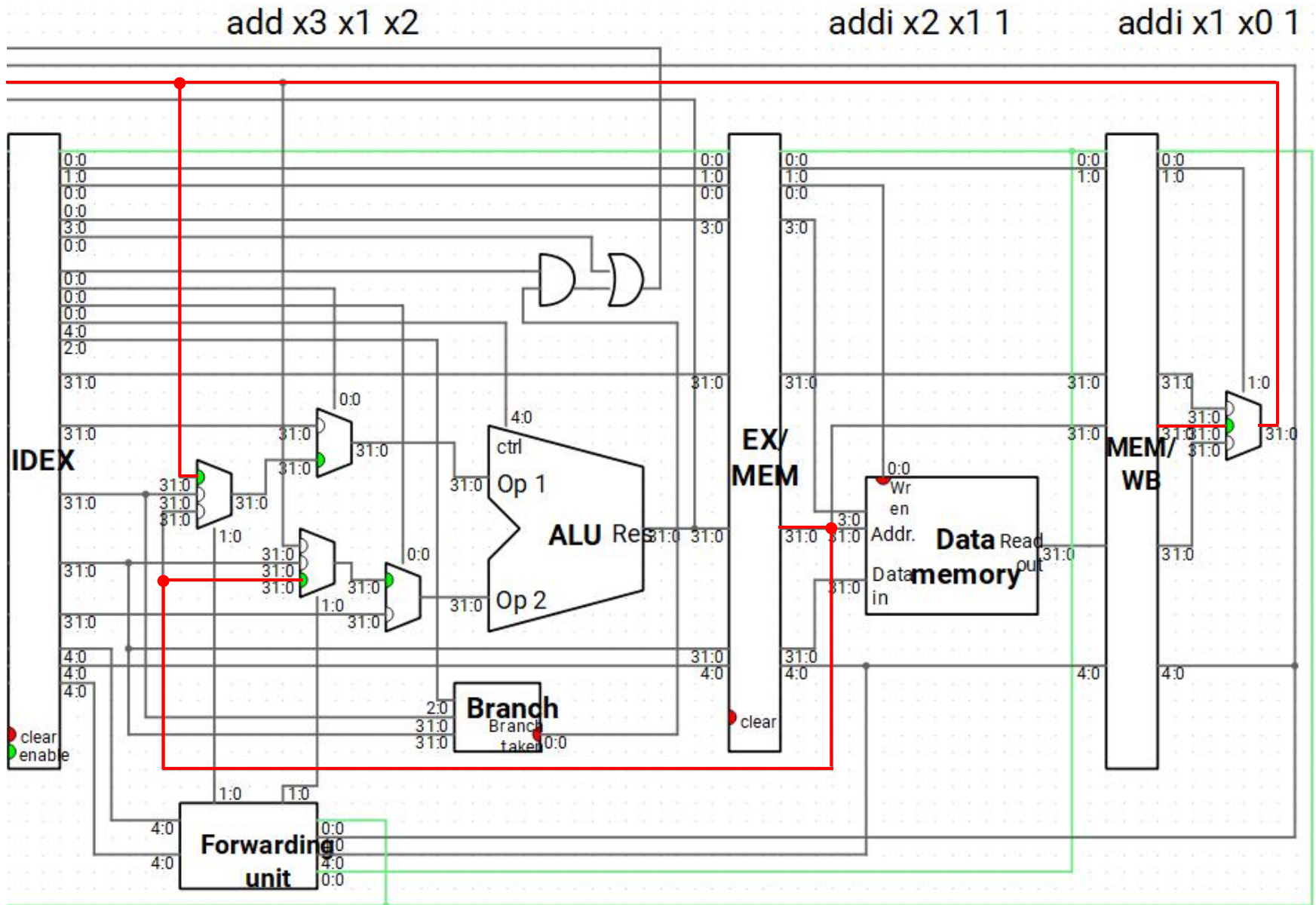
数据相关 (续2)



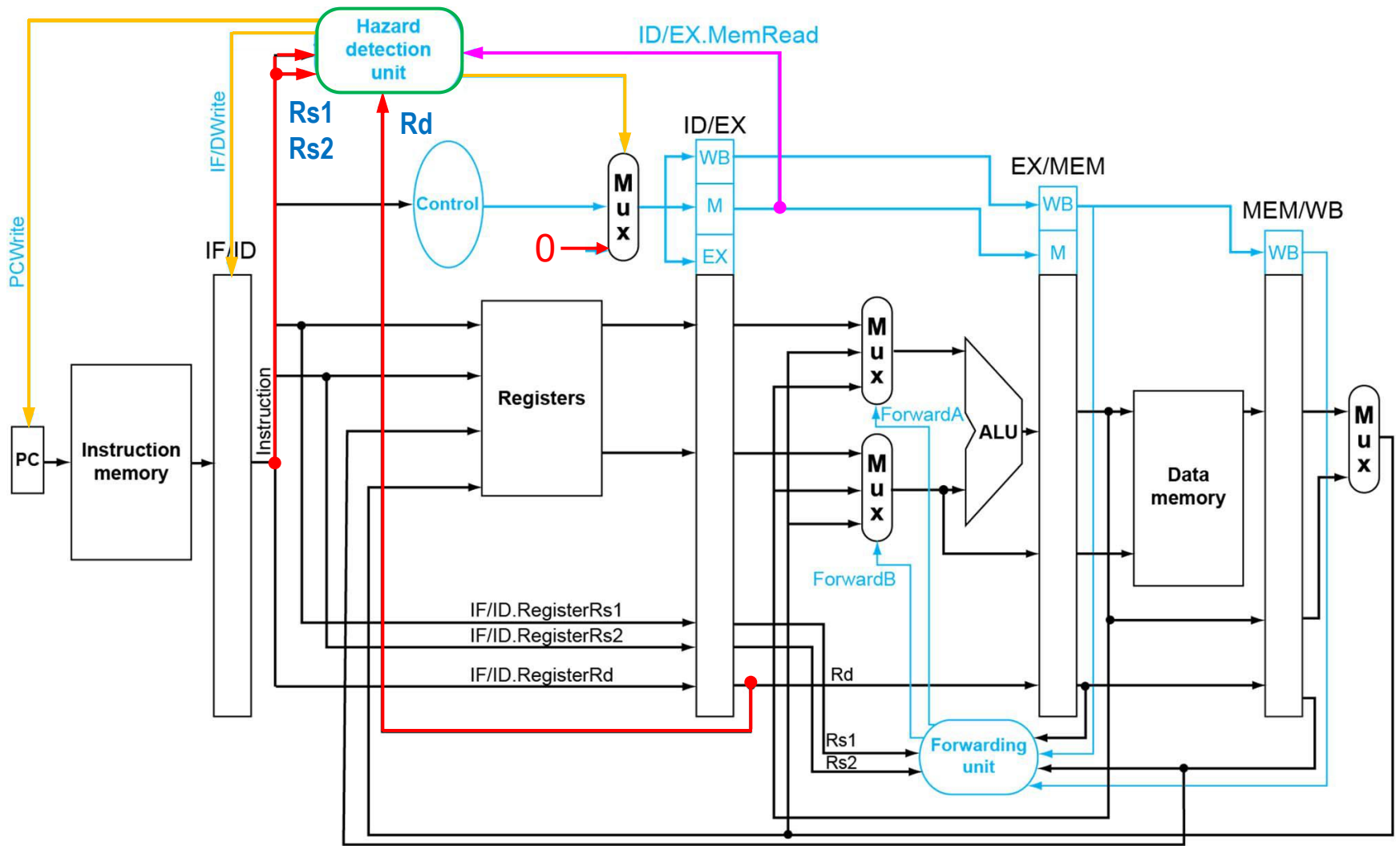
Forwarding



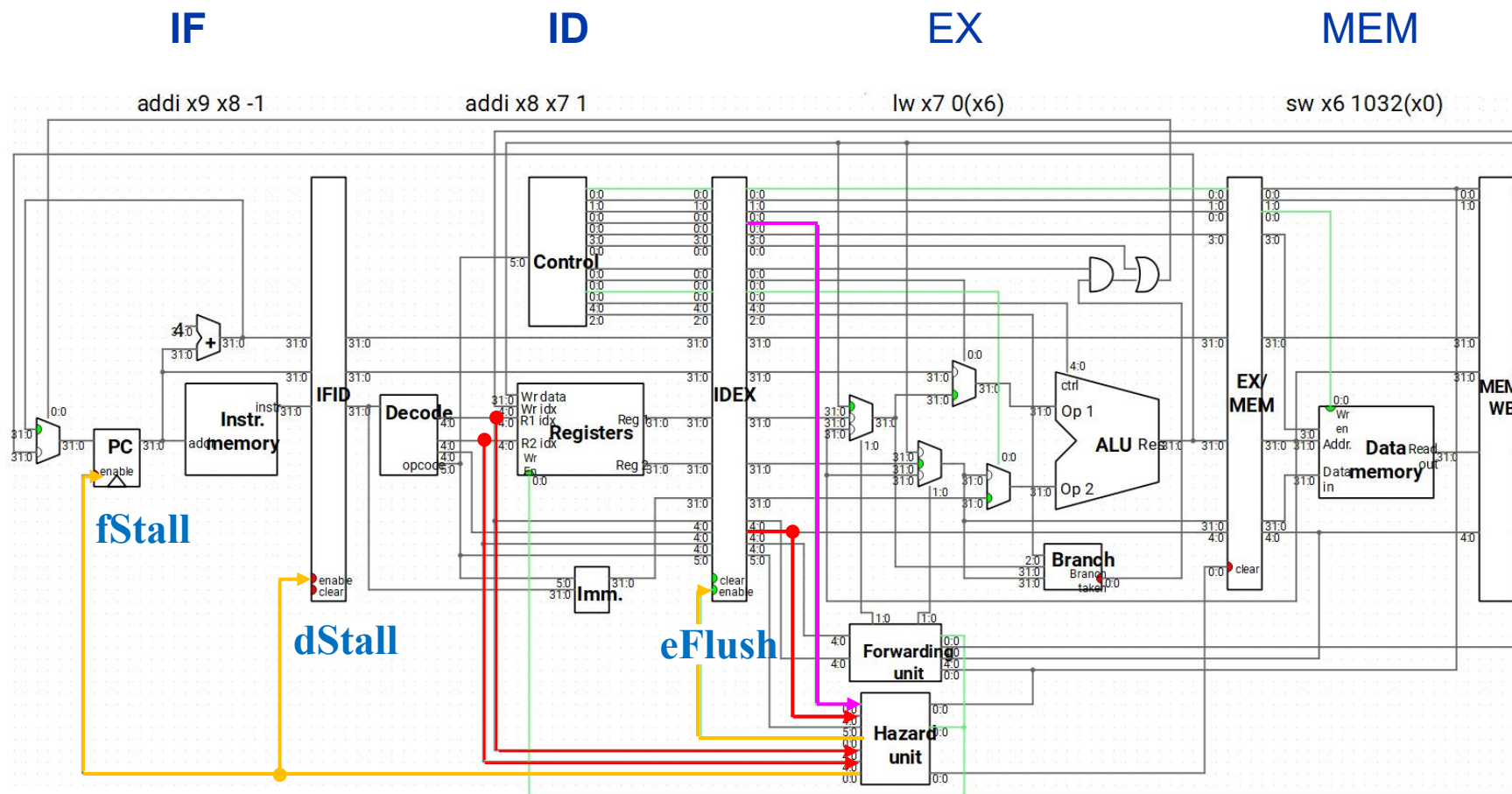
Forwarding: Ripes



Load-Use Hazard



Load-Use Hazard: Ripes



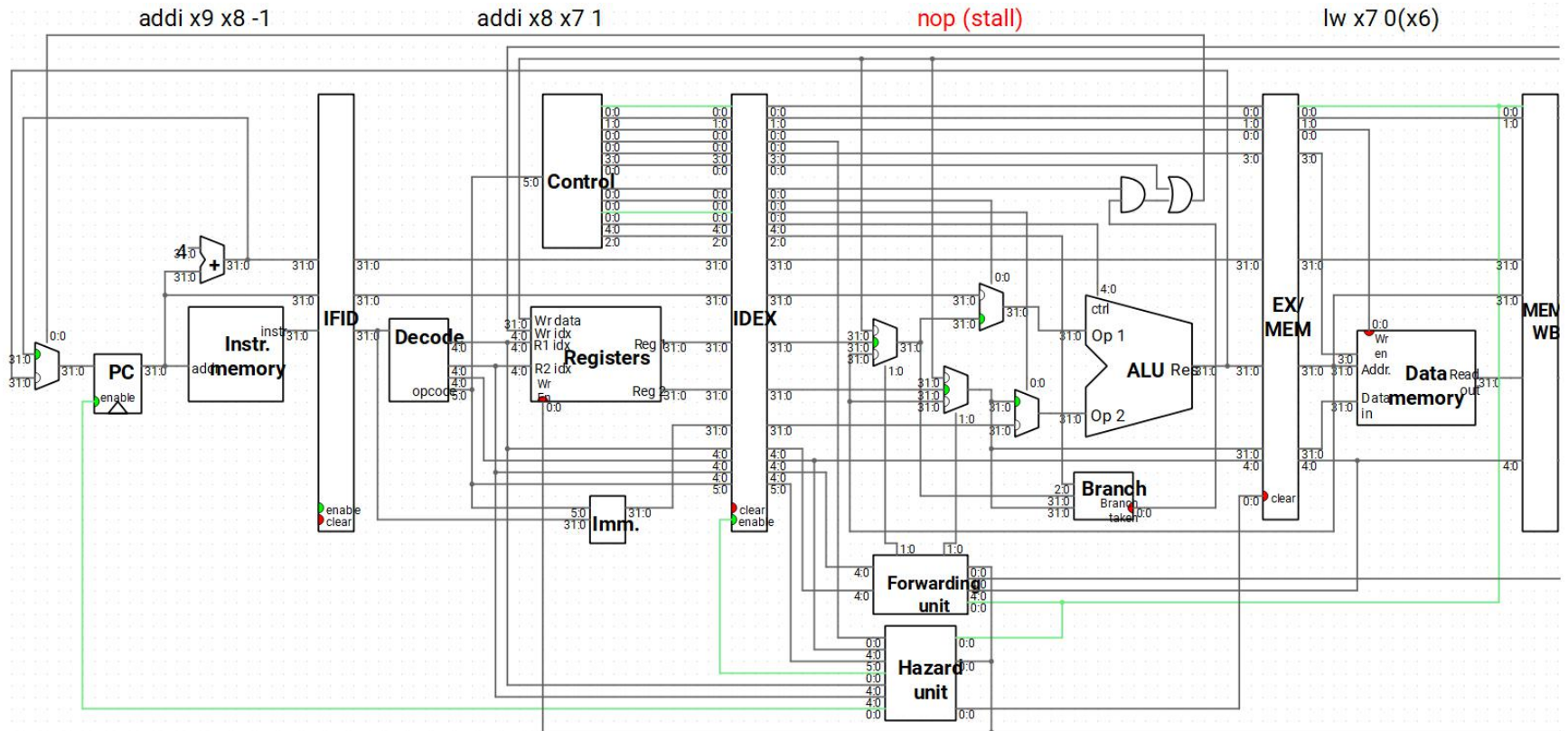
Load-Use Hazard: Ripes

IF

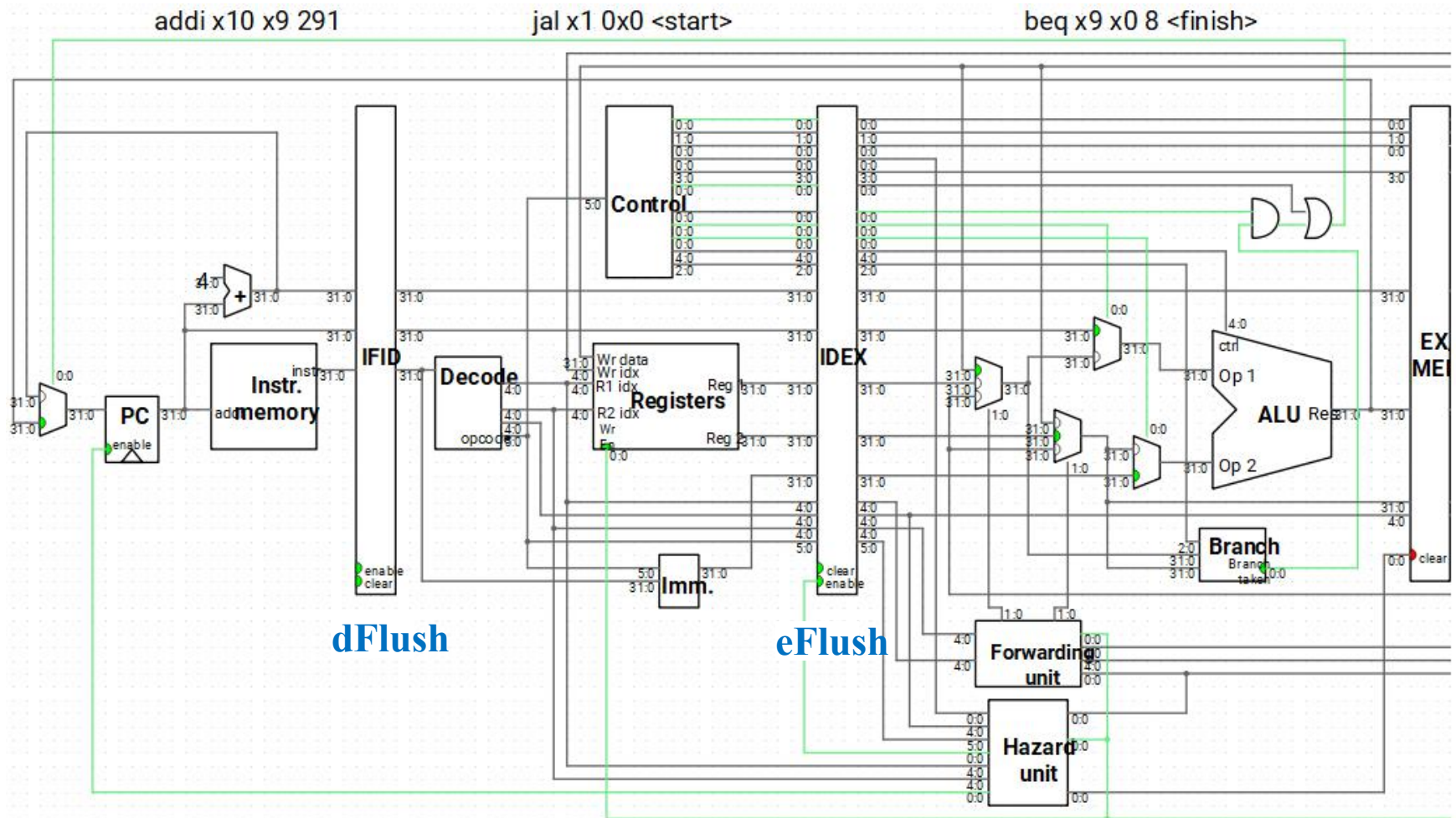
ID

EX

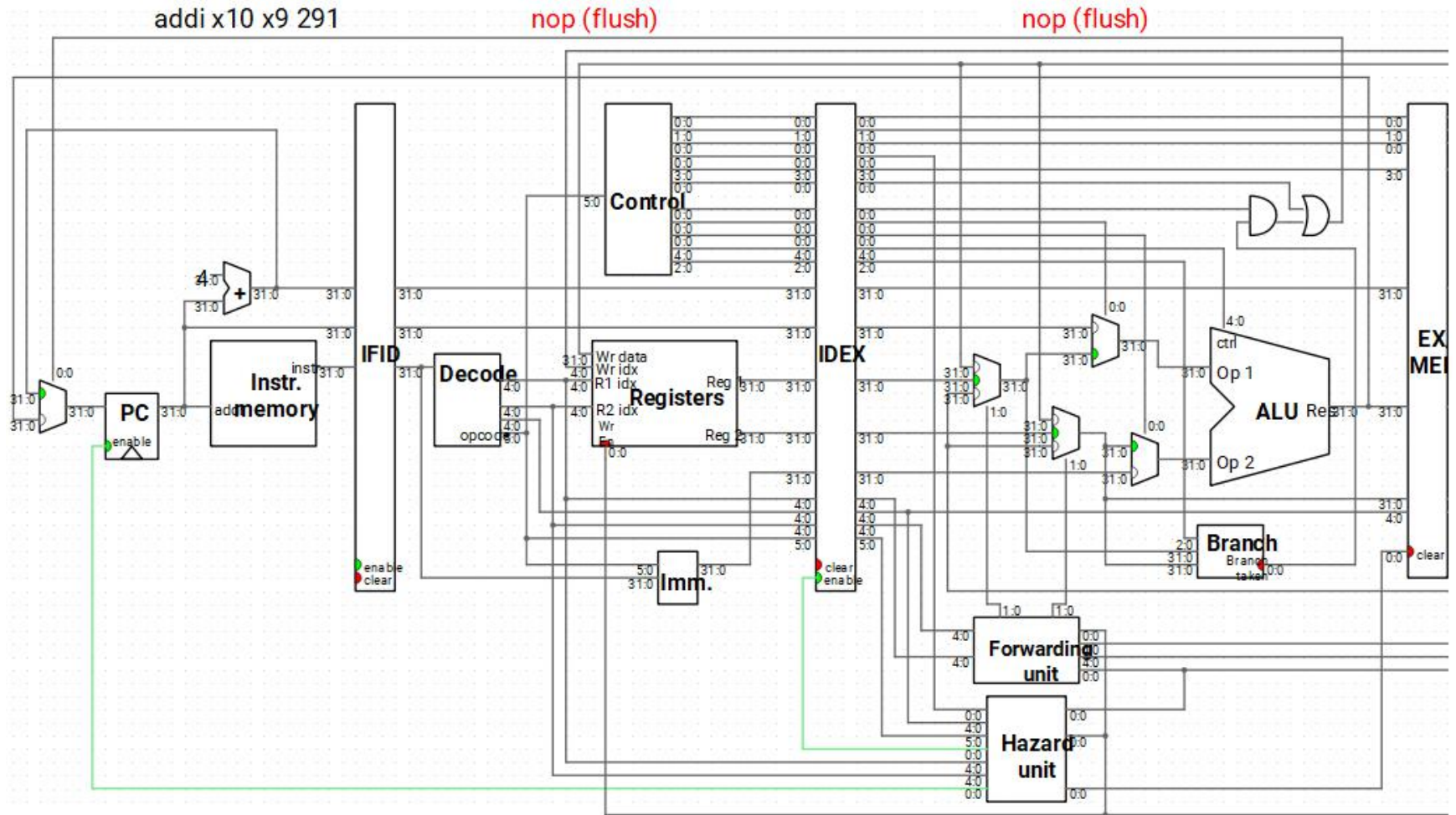
MEM



Branch Hazard: Ripes

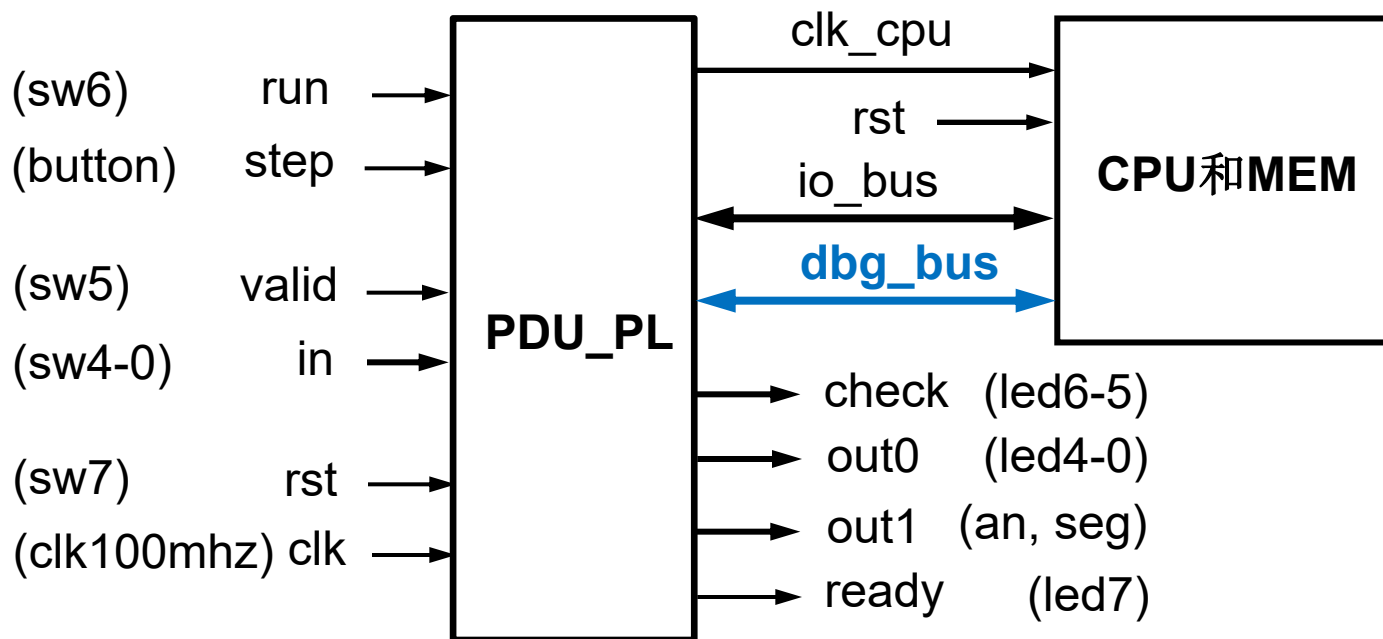


Branch Hazard: Ripes



流水线处理器调试单元

- **PDU_PL (Processor Debug Unit for Pipe-Line)**
 - 控制CPU的运行方式: `run = 1` 连续运行, `0` 单步运行
 - 管理外设 (开关`sw`、指示灯`led`、数码管`seg`), 显示CPU运行结果和数据通路状态



IO_BUS信号

- CPU运行时访问外设的总线（地址、数据和控制）信号
 - io_addr: I/O外设的地址
 - io_din: CPU接收来自外设sw的输入数据
 - io_dout: CPU向外设led和seg输出的数据
 - io_we: CPU向外设led和seg输出时的使能信号

存储器映射外设的端口地址表

存储器地址	I/O_addr	输出端口名	输入端口名	外设
0x0000_0400	0	out0	-	led4-0
0x0000_0404	1	ready	-	led7
0x0000_0408	2	out1	-	an, seg
0x0000_040C	3	-	in	sw4-0
0x0000_0410	4	-	valid	sw5

DBG_BUS信号

- 调试时将寄存器堆和数据存储器内容，以及流水段寄存器内容导出显示
 - m_rf_addr: 存储器(MEM)或寄存器堆(RF)的调试读口地址
 - rf_data: 从RF读出的数据
 - m_data: 从MEM读出的数据
 - 流水段寄存器
 - PC/IF/ID: pcin, pc, pcd, ir
 - ID/EX: pce, a, b, imm, rd, ctrl
 - EX/MEM: y, bm, rdm, ctrlm
 - MEM/WB: yw, mdr, rdw, ctrlw

Ctrl:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
fstall	dstall	dflush	eflush	0	0	a_fwd		0	0	b_fwd		0	rf_wr	wb_sel	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	m_rd	m_wr	0	0	jal	br	0	0	a_sel	b_sel	alu_op			

CPU运行方式

- **run = 1: 连续运行**
 - PDU向CPU输出连续时钟信号clk_cpu
 - CPU通过I/O_BUS访问外设
 - 输入端口: in, valid
 - 输出端口: out0, out1, ready
- **run = 0: 单步运行（每次执行一条指令）**
 - 每按动step一次, PDU产生一个周期的clk_cpu
 - 执行外设输入指令前, 应先设置好valid或in后, 再按动step
 - 执行任何指令后, led和数码管(an, seg)显示当前程序运行结果
 - 随后可以通过改变valid和in查看寄存器堆、存储器和**流水段寄存器(Pipe-Line Register, PLR)**的内容

外设使用说明表

- 利用vld和in选择需要查看的信息，chk指示out0和out1上显示信息的类型

SW						btn	led			seg	说明
7	6	5	4~2	1	0		7	6~5	4~0		
rst	run	vld	in			step	rdy	chk	out0	out1	
			ah_m	pre	next						
↑	-	-	-	-	-	-	1	00	0x1F	0x12..78	复位
x	1	vld	in			-	rdy	00	out0	out1	连续运行
	0	vld	in			↑	rdy	00	out0	out1	单步运行
		↑↓	-	↑↓	↑↓	x	rdy	01	a_rf	d_rf	查看寄存器堆
			ah_m					10	al_m	d_m	查看存储器
			-					11	a_plr	d_plr	查看流水段寄存器


查看寄存器堆和数据存储器

- 寄存器堆




- 利用vld (sw5)切换chk (led6~5) = 01
- 使用pre (sw1)和next (sw0)修改寄存器堆调试读端口地址a_rf (led4~0), out1(seg)显示读出的数据

- 数据存储器

- 利用vld (sw5)切换chk (led6~5) = 10 
- 通过sw4~2设置数据存储器调试读端口高位地址ah_m (sw4~2)
- 使用pre (sw1)和next (sw0)修改数据存储器调试读端口低位地址al_m (led4~0), out1(seg)显示读出的数据

查看流水段寄存器

- 利用vld (sw5)切换chk (led6~5) = 11 
- 使用pre (sw1)和next (sw0)分别修改流水段寄存器地址 ah_plr和al_plr, out1 (seg)显示对应寄存器数据(d_plr)

ah_plr (led4~3)	al_plr (led2~0)	d_plr (seg)	ah_plr (led4~3)	al_plr (led2~0)	d_plr (seg)
00 (PC/IF/ID)	x00	pc	10 (EX/MEM)	x00	y
	x01	pcd		x01	bm
	x10	ir		x10	rdm
	x11	pcin		x11	ctrlm
01 (ID/EX)	000	pce	11 (MEM/WB)	x00	yw
	001	a		x01	mdr
	010	b		x10	rdw
	011	imm		x11	ctrlw
	100	rd			
	101	ctrl			

CPU_PL模块接口

```
module cpu_pl (  
    input clk,  
    input rst,  
  
    //IO_BUS  
    output [7:0] io_addr,      //led和seg的地址  
    output [31:0] io_dout,     //输出led和seg的数据  
    output io_we,              //输出led和seg数据时的使能信号  
    input [31:0] io_din,       //来自sw的输入数据  
  
    //Debug_BUS  
    input [7:0] m_rf_addr,     //存储器(MEM)或寄存器堆(RF)的调试读口地址  
    output [31:0] rf_data,     //从RF读取的数据  
    output [31:0] m_data,      //从MEM读取的数据  
  
    //PC/IF/ID 流水段寄存器
```

CPU_PL模块接口(续)

```
output [31:0] pc,  
output [31:0] pcd,  
output [31:0] ir,  
output [31:0] pcin,
```

// ID/EX 流水段寄存器

```
output [31:0] pce,  
output [31:0] a,  
output [31:0] b,  
output [31:0] imm,  
output [4:0] rd,  
output [31:0] ctrl,
```

// EX/MEM 流水段寄存器

```
output [31:0] y,
```

```
output [31:0] bm,  
output [4:0] rdm,  
output [31:0] ctrlm,
```

// MEM/WB 流水段寄存器

```
output [31:0] yw,  
output [31:0] mdr,  
output [4:0] rdw,  
output [31:0] ctrlw
```

```
);
```

PDU_PL模块接口

<pre>module pdu_pl (input clk, input rst, //选择CPU工作方式 input run, input step, output clk_cpu, //输入sw的端口 input valid, input [4:0] in, //输出led和seg的端口 output [1:0] check, //led6-5:查看类型 output [4:0] out0, //led4-0</pre>	<pre> output [2:0] an, //8个数码管 output [3:0] seg, output ready, //led7 //IO_BUS input [7:0] io_addr, input [31:0] io_dout, input io_we, output [31:0] io_din, //Debug_BUS output [7:0] m_rf_addr, input [31:0] rf_data, input [31:0] m_data, //PC/IF/ID 流水段寄存器</pre>
---	---

PDU_PL模块接口(续)

```
input [31:0] pc,  
input [31:0] pcd,      //pc_id  
input [31:0] ir,      //inst_id  
input [31:0] pcin,  
  
//ID/EX 流水段寄存器  
input [31:0] pce,      //pc_ex  
input [31:0] a,        //rf_rdata1_ex  
input [31:0] b,        //rf_rdata2_ex  
input [31:0] imm,      //imm_gen_out_ex  
input [4:0] rd,        //rf_waddr_ex  
input [31:0] ctrl,  
  
//EX/MEM 流水段寄存器  
input [31:0] y,        //alu_out_mem
```

```
input [31:0] bm,      // reg_b_mem  
input [4:0] rdm,      // rf_waddr_mem  
input [31:0] ctrlm,  
  
//MEM/WB 流水段寄存器  
input [31:0] yw,      //alu_out_wb  
input [31:0] mdr,      //Mem_rdata_wb  
input [4:0] rdw,      //rf_waddr_wb  
input [31:0] ctrlw  
);
```


实验步骤

1. 修改Lab4寄存器堆模块，使其满足写优先(Write First)，即在对同一寄存器读写时，写数据可立即从读数据输出

2.设计完整的流水线CPU：

- 1) 设计无数据和控制相关处理的流水线CPU，将CPU和PDU连接，加载no_hazard_test.coe至指令存储器（自测）。
- 2) 设计仅有数据相关处理的流水线CPU，将CPU和PDU连接，加载data_hazard_test.coe至指令存储器（自测）。
- 3) 设计有数据和控制相关处理的流水线CPU，将CPU和PDU连接，分别加载hazard_test.coe 和fib_test.coe至指令存储器（最终检查）。

The End