第三次作业

4.1

4.1 考虑如下指令：

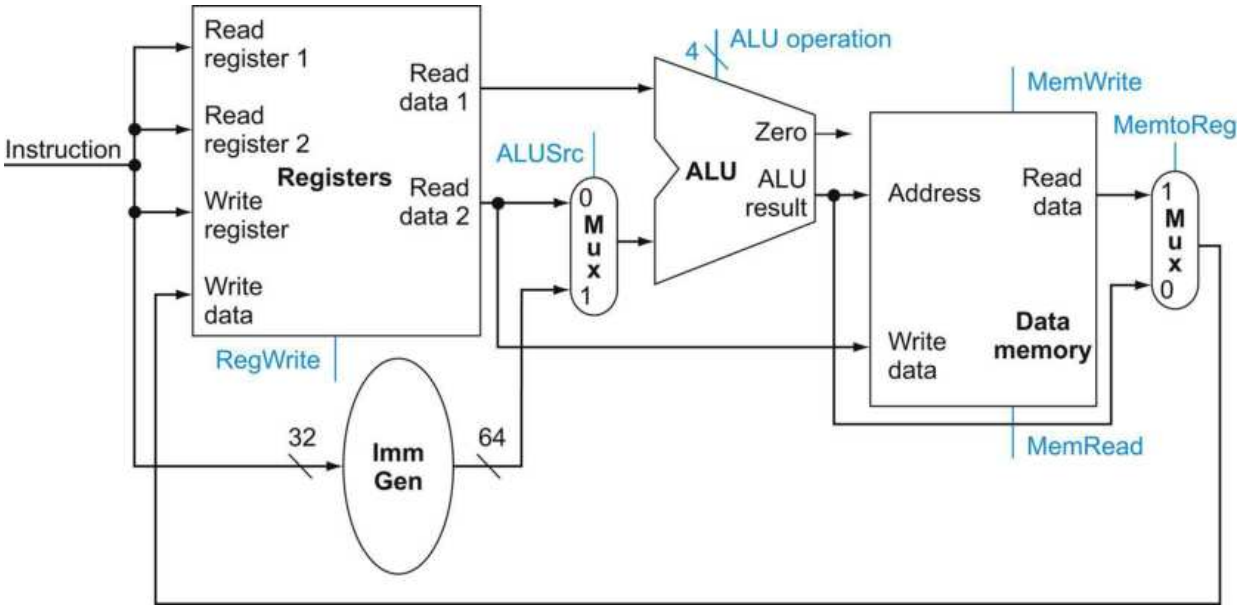
指令：`and rd, rs1, rs2`

解释：`Reg[rd] = Reg[rs1] AND Reg[rs2]`

4.1.1 [5]<4.3>对于上述指令，图 4-10 中的控制信号各是什么数值？

4.1.2 [5]<4.3>对于上述指令，将用到哪些功能单元？

4.1.3 [10]<4.3>对于上述指令，哪些功能单元不产生任何输出？哪些功能单元的输出不会被用到？



4.1.1

4.1.1 [5]<4.3>对于上述指令，图 4-10 中的控制信号各是什么数值？

`RegWrite`	`ALUSrc`	`ALU operation`	`MemWrite`	`MemRead`	`MemtoReg`
1	0	0000(AND)	0	0	0

注意审题，是**AND**不是**ADD**，有的同学给的**ALU Op**是**0010(ADD)**

4.1.2

4.1.2 [5]<4.3>对于上述指令，将用到哪些功能单元？

除了`Imm Gen`和`Data memory`以外的所有功能单元

4.1.3

4.1.3 [10]<4.3>对于上述指令，哪些功能单元不产生任何输出？哪些功能单元的输出不会被用到？

哪些功能单元不产生任何输出

两个答案：

1. `Data memory`不产生任何输出
2. 所有部件均产生输出

对于第一种答案`Data memory`，是因为有`MemRead`信号的存在，当该信号为0时，`Read data`输出没有输出，即高阻态。其它的功能单元主要都是组合逻辑，实际上都会产生输出，只不过这些输出没有被选择器选择。

哪些功能单元的输出不会被用到

`Imm Gen`的输出未被用到。

4.7

4.7 假设用来实现处理器数据通路的各功能模块延迟如下所示：

I-Mem / D-Mem	Register File	Mux	ALU	Adder	Single gate	Register Read	Register Setup	Sign extend	Control
250ps	150ps	25ps	200ps	150ps	5ps	30ps	20ps	50ps	50ps

其中，寄存器读延迟指的是，时钟上升沿到寄存器输出端稳定输出新值所需的时间。该延迟仅针对 PC 寄存器。寄存器建立时间指的是，寄存器的输入数据稳定到时钟上升沿所需的时间。该数值针对 PC 寄存器和寄存器堆。

4.7.1 [5] <4.4> R 型指令的延迟是多少？比如，如果想让这类指令工作正确，时钟周期最少为多少？

4.7.2 [10] <4.4> ld 指令的延迟是多少？仔细检查你的答案，许多学生会在关键路径上添加额外的寄存器。

4.7.3 [10] <4.4> sd 指令的延迟是多少？仔细检查你的答案，许多学生会在关键路径上添加额外的寄存器。

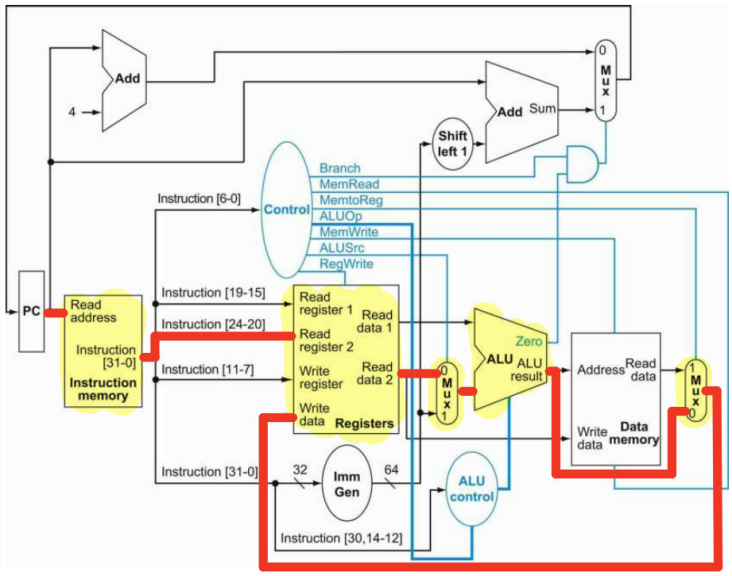
4.7.4 [5] <4.4> beq 指令的延迟是多少？

4.7.5 [5] <4.4> I 型指令的延迟是多少？

4.7.6 [5] <4.4> 该 CPU 的最小时钟周期是多少？

4.7.1

4.7.1 [5] <4.4> R 型指令的延迟是多少？比如，如果想让这类指令工作正确，时钟周期最少为多少？



延迟(最小时钟周期):

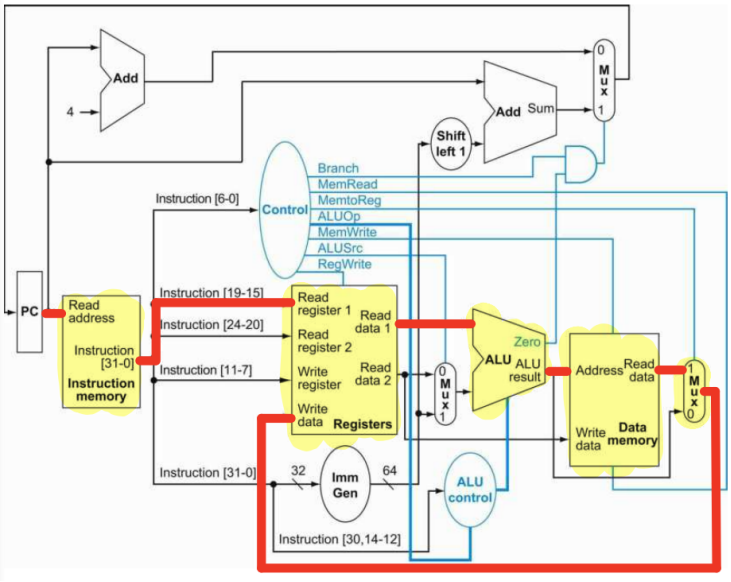
$$30(PC_{Read}) + 250(IMem) + 150(RF) + 25(MUX_{ALUSrc}) + 200(ALU) + 25(MUX_{MemtoReg}) + 20(RF_{setup}) = 700ps$$

有的同学会问读寄存器堆的时候需不需要考虑register setup的时间，由于题目没有说清楚，所以考虑不考虑都算正确（但我倾向于不考虑，所以这题以及之后的延迟计算都没有加上读寄存器的setup时间）。

另外`Imm Gen`和Register File是同时收到指令的各个部分的，所以两者是并行的。

4.7.2

4.7.2 [10] <4.4> ld 指令的延迟是多少？仔细检查你的答案，许多学生会在关键路径上添加额外的寄存器。

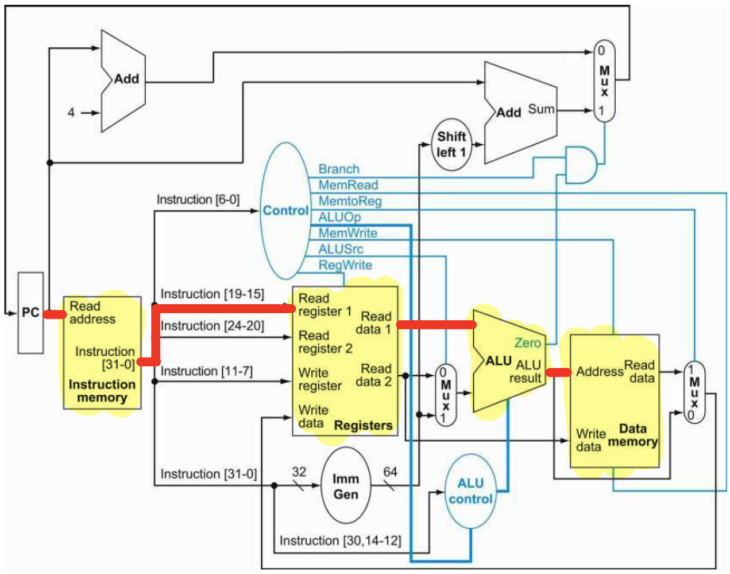


延迟:

$$30(PC_{Read})+150(IMem)+150(RF)+200(ALU)+250(DMem)+25(MUX_{MemtoReg})+20(RF_{setup})=925ps$$

4.7.3

4.7.3 [10] <4.4> sd 指令的延迟是多少？仔细检查你的答案，许多学生会在关键路径上添加额外的

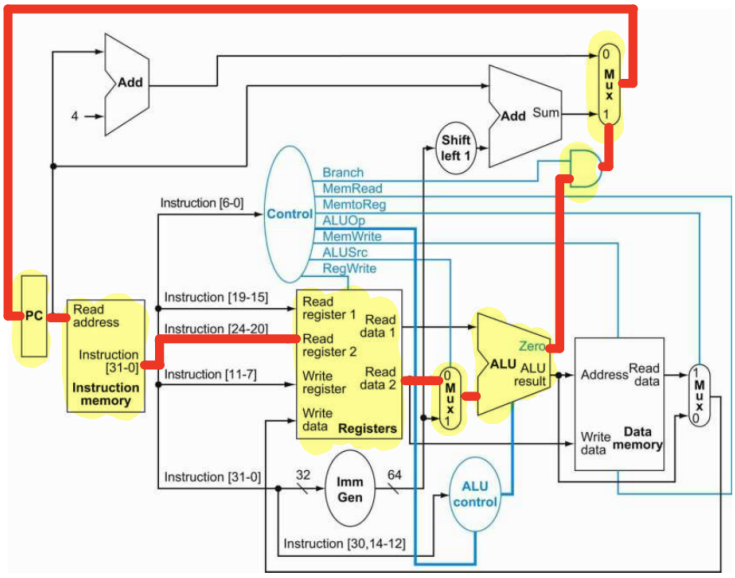


延迟:

$$30(PC_{Read})+250(IMem)+150(RF)+200(ALU)+250(DMem)=880ps$$

4.7.4

4.7.4 [5] <4.4> beq 指令的延迟是多少？



延迟:

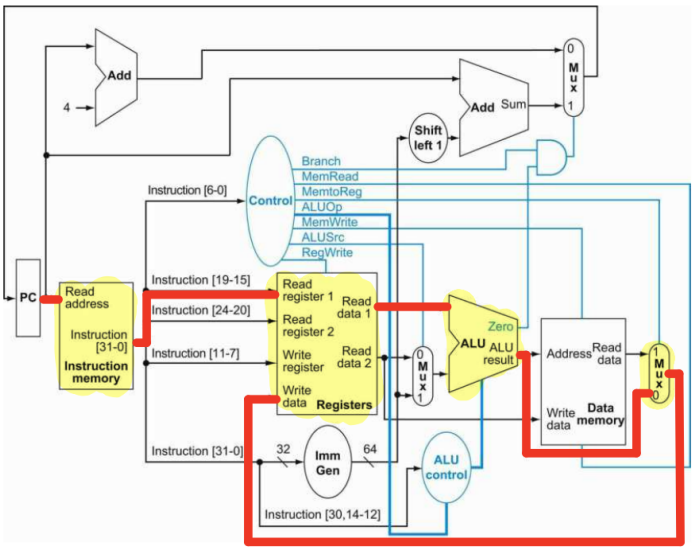
$$30(PC_{Read})+250(IMem)+150(RF)+25(MUX_{ALUSrc})+200(ALU)+5(AND_{gate})+25(MUX_{PCSrc})+20(PC_{Regsetup})=705ps$$

4.7.5

4.7.5 [5] <4.4> I型指令的延迟是多少?

I-type	lb	0000011	000	n.a.
	lh	0000011	001	n.a.
	iw	0000011	010	n.a.
	id	0000011	011	n.a.
	ibu	0000011	100	n.a.
	ihu	0000011	101	n.a.
	iwu	0000011	110	n.a.
	addi	0010011	000	n.a.
	slli	0010011	001	000000
	xori	0010011	100	n.a.
	srli	0010011	101	000000
	srai	0010011	101	010000
	ori	0010011	110	n.a.
	andi	0010011	111	n.a.
	jalr	1100111	000	n.a.

ADDI的关键路径示例



load类型和lw延迟相同

立即数的算数运算比R-type延迟小一个MUX

如果按照I-type里耗时最长的指令(load操作指令)来计算延迟，那么I-type延迟为925ps

不过书本上**I-type**在单周期里并没有说**lb**等指令，所以如果结果为**675ps**也正确

4.7.6

4.7.6 ［5］<4.4>该 CPU 的最小时钟周期是多少?

最小时钟周期按照最长指令延迟(ld)来计算，为925ps

PS:

鉴于有的同学可能认为**MUX**要等两个输入都稳定了才能输出，所以**4.7.2/4.7.3/4.7.6**写了**950/905/950**的也不扣分，但是并不提倡，因为对于实际硬件来说，并不需要等到两者输入都稳定才能输出。

C.2

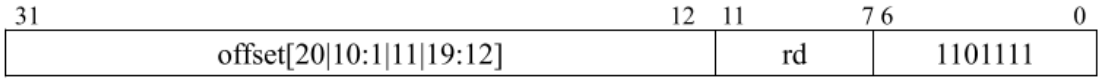
C.2 [5] <§C.3> We wish to add the instruction `jal` (jump and link). Make any necessary changes to the datapath or to the control signals if needed. You can photocopy figures to make it faster to show the additions. How many product terms are required in a PLA that implements the control for the single-cycle datapath for `jal`?

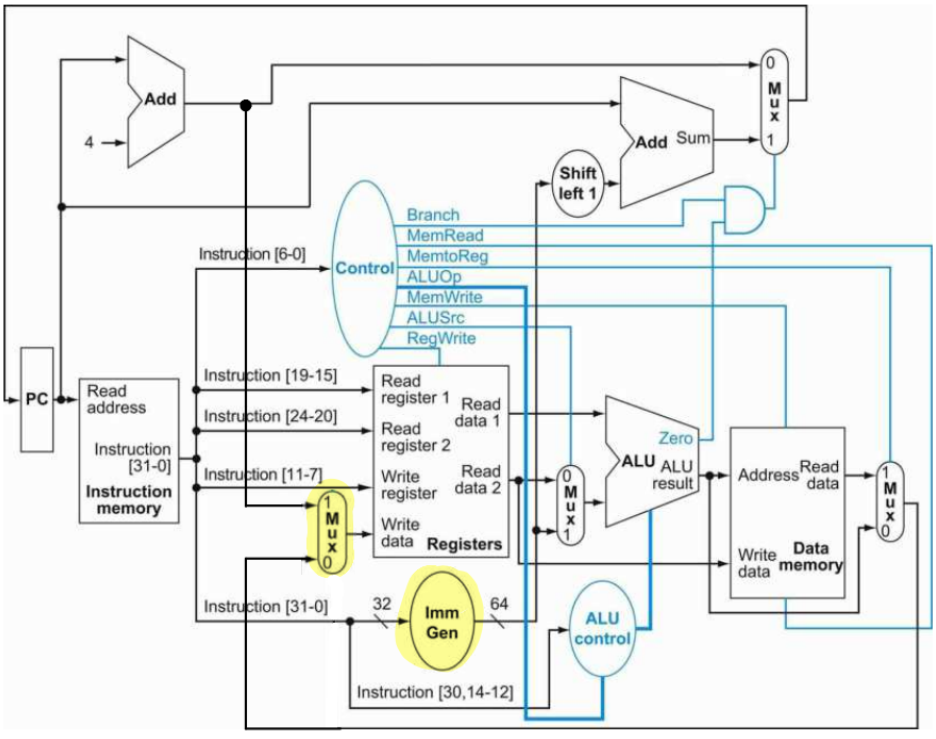
jal rd, offset　　x[rd] = pc+4; pc += sext(offset)

跳转并链接 *(Jump and Link)*. J-type, RV32I and RV64I.

把下一条指令的地址(*pc*+4)，然后把*pc* 设置为当前值加上符号位扩展的*offset*。*rd* 默认为 x1。

压缩形式：**c.j** offset; **c.jal** offset





可以修改原有的ImmGen，通过control发出的信号来生成对应的立即数。另外需要增加将PC写入RF的部分。

老师**PPT 37**页的图不全，缺少**PC+4**写回**Register File**的部分

I=7'b1101111

增加一个MUX signal和一个Imm Gen signal。

例如可以写为

$$MUX_{RFdata} = ImmGen_{control} = I[0] \wedge I[1] \wedge I[2] \wedge I[3] \wedge \neg I[4] \wedge I[5] \wedge I[6]$$

RV指令结构

32-bit Instruction Formats

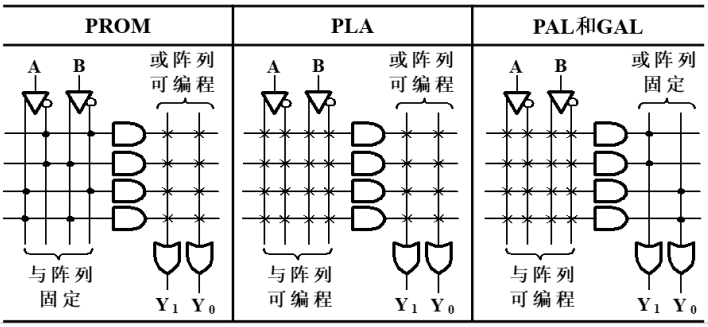
	31	30 25	24 21	20	19 15	14 12	11 8	7	6 0
R	funct7		rs2		rs1	funct3	rd		opcode
I	imm[11:0]				rs1	funct3	rd		opcode
S	imm[11:5]		rs2		rs1	funct3	imm[4:0]		opcode
SB	imm[12]	imm[10:5]	rs2		rs1	funct3	imm[4:1]	imm[11]	opcode
U	imm[31:12]						rd		opcode
UJ	imm[20]	imm[10:1]	imm[11]		imm[19:12]		rd		opcode

可以看到rs2始终为Inst[24:20]，rd始终为Inst[11:6]，所以不需要在通过RegDst信号选择。这点和**MIPS**不同。

PLA

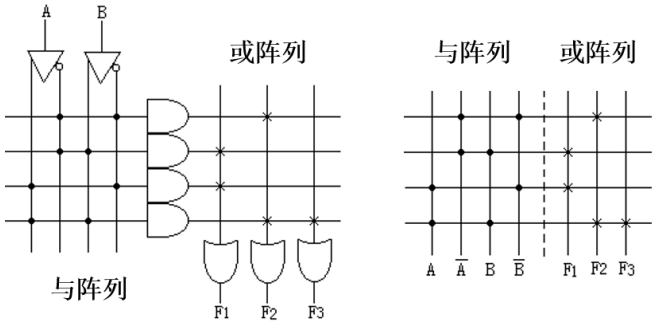
低密度PLD的与、或阵列结构

- **PROM: Programmable Read Only Memory,** 可编程只读存储器
- **PLA: Programmable Logic Array,** 可编程逻辑阵列
- **PAL: Programmable Array Logic,** 可编程阵列逻辑
- **GAL: Gate Array Logic,** 门阵列逻辑



示例—PROM实现组合逻辑

$F1 = \overline{A}\cdot B + A\cdot\overline{B}$ $F2 = \overline{A}\cdot\overline{B} + A\cdot B$ $F3 = A\cdot B$



In a PLA, each output is the logical OR of one or more minterms. A *minterm*, also called a *product term*, is simply a logical AND of one or more inputs. The inputs can be thought of as the address for indexing the PLA, while the minterms select which of all possible address combinations are interesting. A minterm corresponds to a

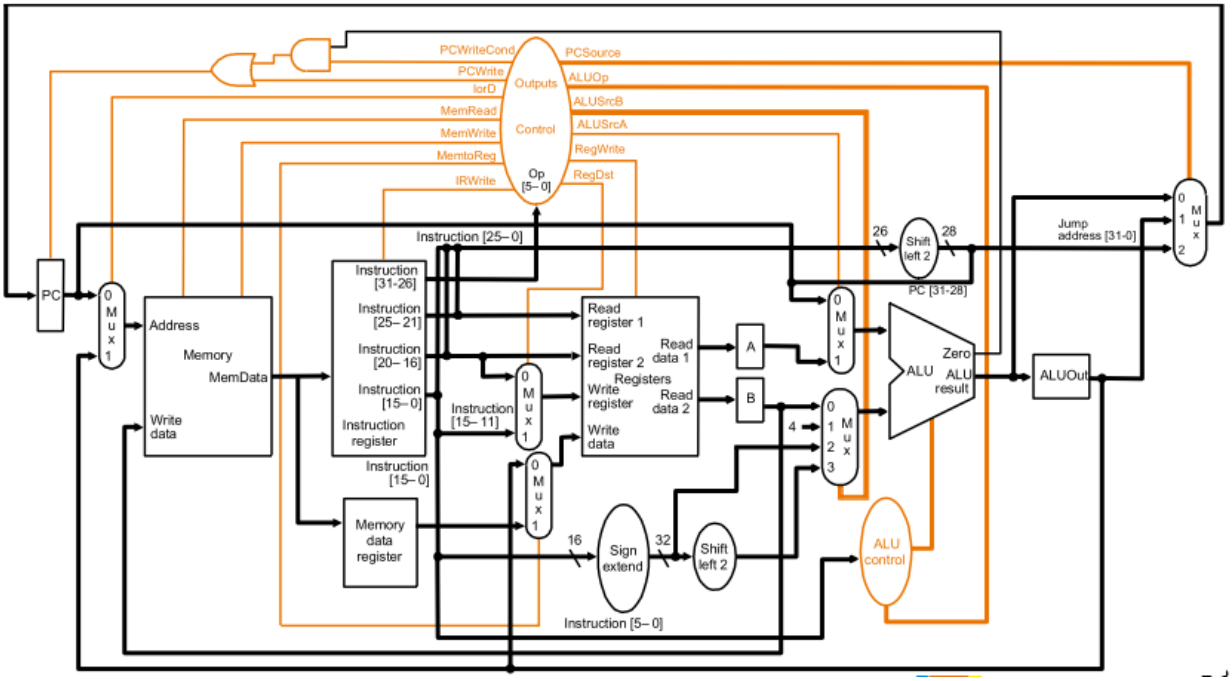
这样一共需要2个乘积项

本题答案不唯一，言之有理即可

分析多周期功能部件

> 分析RISC-V的五种类型指令(R/I/S/B/J)或MIPS三种类型指令(R/I/J)的多周期设计方案中每个周期所用到的功能部件

数据通路(MIPS/RISC-V)



MIPS

分析 MIPS 三种类型指令的多周期设计方案中每个周期所用到的功能部件。

阶段\指令类型	R	I	Beq	J
取指	PC, Add, Control, Mem, IR			
译码	IR, Control, regs	IR, Control, regs Sign extend		Control, PC
执行	ALU, Control	ALU, Control	ALU, Control, Branch, PC	
访存	Control	Mem, MDR, Control		
回写	Control, regs	Control, regs		

如果有同学认为**R-type**应该在访存段写入寄存器堆，**J-type**在**EX**段跳转，也都是正确的。

RISC-V

Format	Instruction	Opcode	Funct3	Funct6/7
R-type	add	0110011	000	0000000
	sub	0110011	000	0100000
	sll	0110011	001	0000000
	xor	0110011	100	0000000
	srl	0110011	101	0000000
	sra	0110011	101	0000000
	or	0110011	110	0000000
	and	0110011	111	0000000
	lr.d	0110011	011	0001000
	sc.d	0110011	011	0001100
I-type	lb	0000011	000	n.a.
	lh	0000011	001	n.a.
	iw	0000011	010	n.a.
	id	0000011	011	n.a.
	ibu	0000011	100	n.a.
	ihu	0000011	101	n.a.
	iwu	0000011	110	n.a.
	addi	0010011	000	n.a.
	slli	0010011	001	000000
	xori	0010011	100	n.a.
	srli	0010011	101	000000
	srai	0010011	101	010000
	ori	0010011	110	n.a.
	andi	0010011	111	n.a.
	jalr	1100111	000	n.a.
S-type	sb	0100011	000	n.a.
	sh	0100011	001	n.a.
	sw	0100011	010	n.a.
	sd	0100011	111	n.a.
SB-type	beq	1100111	000	n.a.
	bne	1100111	001	n.a.
	blt	1100111	100	n.a.
	bge	1100111	101	n.a.
	bltu	1100111	110	n.a.
	bgeu	1100111	111	n.a.
U-type	lui	0110111	n.a.	n.a.
UJ-type	jal	1101111	n.a.	n.a.

RISC-V和MIPS对应

RISC-V	**MIPS**
R-type	R-type
I-type(load类型)	I-type
I-type(imm类型)	R-type 在译码阶段增加Sign extend
I-type(jalr)	不管了吧。
S-type	I-type 去掉访存阶段的MDR 去掉回写阶段所有
B-type	Beq
J-type	J