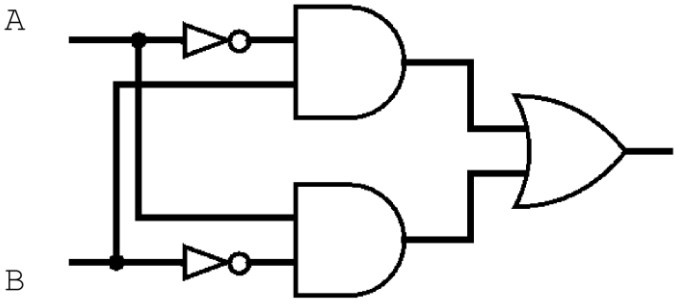
1. 体系结构基础：判断下列描述更符合 **CISC** 还是（早期）**RISC**

	CISC	**RISC**
指令机器码长度固定		✔
指令类型多、功能丰富	✔	
不采用条件码		✔
实现同一功能，需要的汇编代码较多		✔
译码电路复杂	✔	
访存模式多样	✔	
参数、返回地址都使用寄存器进行保存		✔
x86-64	✔	
MIPS		✔
广泛用于嵌入式系统		✔
已知某个体系结构使用add R1,R2,R3 来完成加法运算。当要将数据从寄存器S 移动至寄存器D 时，使用add S,#ZR,D 进行操作（#ZR 是一个恒为0 的寄存器），而没有类似于mov的指令。		✔
已知某个体系结构提供了 xlat 指令，它以一个固定的寄存器A 为基地址，以另一个固定的寄存器B 为偏移量，在A 对应的数组中取出下标为B 的项的内容，放回寄存器A 中。	✔	

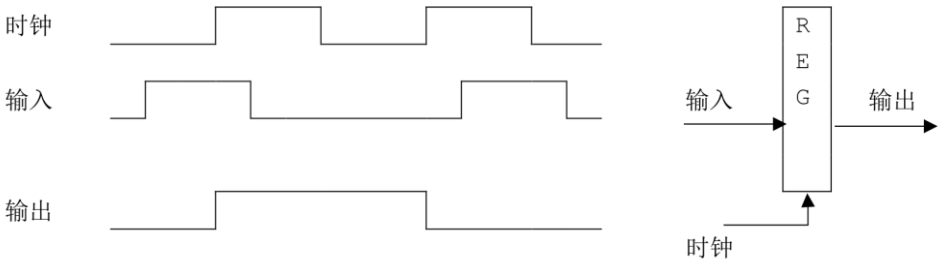
2. 写出下面电路的表达式



(!A&&B) || (!B&&A)

3. 下列寄存器在时钟上升沿锁存数据，画出输出的电平（忽略建立/保持时间）

答案



4. SEQ 模型：根据 Y-86 模型完成下表

		CALL Dest	JXX Dest
Fetch	icode:ifun	icode:ifun <- M_1[PC]	icode:ifun <- M_1[PC]
	rA,rB		
	valC	valC <- M_8[PC+1]	valC <- M_8[PC+1]
	valP	valP <- PC+9	valP <- PC+9
Decode	valA,srcA		
	valB,srcB	valB <- R[%rsp]	
Execute	valE	valE <- valB + (-8)	
	Cond Code		Cnd <- Cond(CC, ifun)
Memory	valM	M_8[%rsp] <- valP	
Write Back	dstE	R[%rsp] <- valE	
	dstM		
PC Update	PC	PC <- valC	PC <- Cnd? valC: valP

5. 已知 valA,valB 为从寄存器 rA,rB 中读出的值， valC 为指令中的常数值，valM 为访存得到的数据，valP 为 PC 自增得到的值，完成SEQ处理器中下面的HCL逻辑：

Stage: Execute

```
word aluA = [
    icode in { IRRMOVQ, IOPQ } : valA;
    icode in { IIRMOVQ, IRMMOVQ, IMRMOVQ } : valC;
    icode in { ICALL, IPUSHQ } : -8;
    icode in { IRET, IPOPQ } : 8;
];
```

Stage: PC Update

```
int new_pc = [
   icode == ICALL : valC;
   icode == IJXX && Cnd: valC;
   icode == IRET: valM;
   1: valP;
];
```