

1. SEQ 模型：根据 Y-86 模型完成下表

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

2. 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
<pre>word aluA = [icode in { IRRMOVQ, IOPQ } : valA; icode in { IIRMOVQ, IRMMOVQ, IMRMOVQ } : valC; icode in { ICALL, IPUSHQ } : -8; icode in { IRET, IPOPQ } : 8;];</pre>
Stage: PC Update
<pre>int new_pc = [icode == ICALL : valC; icode == IJXX && Cnd: valC; icode == IRET: valM; 1: valP;];</pre>

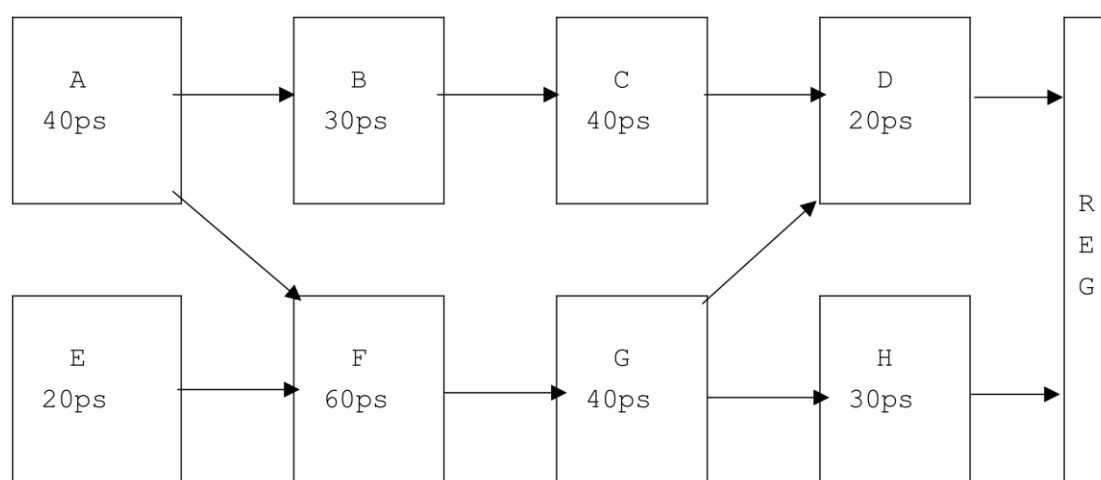
3. 判断以下说法是否正确

(×)	(1) 流水线的深度越深，总吞吐率越大，因此流水线应当越深越好。
(✓)	(2) 流水线的吞吐率取决于最慢的流水级，因此流水线的划分应当尽量均匀。
(✓)	(3) 假设寄存器延迟为 20ps，那么总吞吐率不可能达到或超过 50 GIPS。
(×)	(4) 数据冒险总是可以只通过转发来解决。
(✓)	(5) 数据冒险总是可以只通过暂停流水线来解决。

4. 一条三级流水线，包括延迟为50ps，100ps，100ps 的三个流水级，每个寄存器的延迟为10ps。那么这条流水线的总延迟是__330__ps，吞吐率是__9.09__GIPS。

注意：总延迟为110*3ps，而不是50+100+100+10*3ps；吞吐量1000/110=9.09GIPS

5. A~H 为8个基本逻辑单元，下图中标出了每个单元的延迟，以及用箭头标出了单元之间的数据依赖关系。寄存器的延迟均为10ps。



- (1) 计算目前的电路的总延迟 **40+60+40+30+10=180ps**
- (2) 通过插入寄存器，可以对这个电路进行流水化改造。现在想将其改造为两级流水线，为了达到尽可能高的吞吐率，问寄存器应插在何处？获得的吞吐率是多少？

插在BC、FG之间。1000/110=9.09GIPS

- (3) 现在想将其改造为三级流水线，问最优改造所获得的吞吐率是多少？

插在AB、AF、EF、BC、FG 之间。1000/(40+30+10)=12.5GIPS

6. 一个只使用流水线暂停、没有数据前递的Y86流水线处理器，为了执行以下的语句，至少需要**停顿**多少个周期？一共需要运行多少个周期？

<pre> irmovq \$1, %rax irmovq \$2, %rbx addq %rax, %rcx addq %rbx, %rdx halt </pre>	<pre> rrmovl %eax, %edx mrmovl (%ecx), %eax addl %edx, %eax halt </pre>	<pre> irmovl \$0x40, %eax mrmovl (%eax), %ebx subl %ebx, %ecx halt </pre>
答案		
停顿2周期	停顿3周期	停顿6周期
运行4+5+2=11周期	运行4+4+3=11周期	运行4+4+6=14周期

7. 考虑Y86中的ret与jXX指令。jXX总是预测分支跳转。

- (1) 写出流水线需要处理ret的条件（ret对应的常量为IRET）：

IRET in {D_icode, E_icode, M_icode}

(2)发现(1)中的条件后,流水线寄存器应如何设置?(选填stall, bubble, normal)

	Fetch	Decode	Execute	Memory	Writeback
ret	stall	bubble	normal	normal	normal

(3)写出流水线需要处理jXX分支错误的条件(jXX对应的常量为IJXX):

(E_icode == IJXX && !e_Cnd)

(4)发现(3)中的条件后,流水线寄存器应如何设置?(选填stall, bubble, normal)

	Fetch	Decode	Execute	Memory	Writeback
JXX错误	stall	bubble	bubble	normal	normal

(5)写出流水线需要处理load/use hazard(加载/使用冒险)的条件:

E_icode in { IMRMOVQ, IPOPOPQ } && E_dstM in { d_srcA, d_srcB }

(6)发现(5)中的条件后,流水线寄存器应如何设置?(选填stall, bubble, normal)

	Fetch	Decode	Execute	Memory	Writeback
load/use	stall	stall	bubble	normal	normal

(7)在Y86流水线中,是否存在一组指令序列可能同时满足上面的三个条件?是否可能同时出现上面的两种条件?

不可能同时满足三个条件;

(load/use hazard和jXX分支错误不同时出现,在E_icode上有冲突)

ret和load/use hazard可以同时满足

ret和branch misprediction可以同时满足

(8)考察以下两组指令序列,写出发现冒险时应该如何设置流水线寄存器。

...	je L1				
popq %rsp	addq %rdx, %rax				
ret	.L1:				
	ret				
	Fetch	Decode	Execute	Memory	Writeback
load/use+ret	stall	stall	bubble	normal	normal
	Fetch	Decode	Execute	Memory	Writeback
jXX+ret	(any)	bubble	bubble	normal	normal

(9)根据以上各小问,补全以下pipe.hcl中的控制逻辑。

```
bool F_stall =
    # Conditions for a load/use hazard
    E_icode in { IMRMOVQ, IPOPOPQ } &&
    E_dstM in { d_srcA, d_srcB } ||
    # Stalling at fetch while ret passes through pipeline
    IRET in { D_icode, E_icode, M_icode };

bool D_bubble =
    # Mispredicted branch
    (E_icode == IJXX && !e_cnd) ||
    # Stalling at fetch while ret passes through pipeline
    # but not condition for a load/use hazard
    !(E_icode in { IMRMOVQ, IPOPOPQ } &&
```

```
E_dstM in { d_srcA, d_srcB }) &&  
IRET in { D_icode, E_icode, M_icode };
```

大题部分的答案请查阅往年期中题答案（存在网盘里）