COD WEEK6

1. T1

> 在本题中将讨论流水线如何影响处理器的时钟周期。假设数据通路的各个流水级的延迟如下：
>
> | IF | ID | EX | MEM | WB |
> | --- | --- | --- | --- | --- |
> | 250ps | 350ps | 150ps | 300ps | 200ps |
>
> 同时，假设处理器执行的指令分布如下：
>
> | ALU/Logic | Jump/Branch | Load | Store |
> | --- | --- | --- | --- |
> | 45% | 20% | 20% | 15% |

1.1 （1）

> 在流水化和非流水化的处理器中，时钟周期分别是多少？

1. 流水化：max=350ps，所以时钟周期为350ps
2. 非流水化：
   - ALU/Logic: IF，ID，EX ，WB 阶段，因此延迟为 250ps + 350ps + 150ps + 200ps=950ps
   - Jump/Branch: IF，ID ， EX 阶段，因此延迟为 250ps + 350ps + 150ps = 750ps
   - Load:所有阶段，因此延迟为 250ps + 350ps + 150ps + 300ps + 200ps = 1250ps
   - Store:IF，ID，EX，MEM因此延迟为 250ps + 350ps + 150ps + 300ps =1050ps

aver：$45\% \times 950ps + 20\% \times 750ps + 20\% \times 1250ps + 15\% \times 1050ps = 977.5ps$

1.2 （2）

> 在流水化和非流水化的处理器中，对于 ld 指令的延迟分别是多少？

1. 流水化：ld指令包含所有阶段：延迟为 250ps + 350ps + 150ps + 300ps + 200ps = 1250ps
2. 非流水化：ld指令包含所有阶段：延迟为 250ps + 350ps + 150ps + 300ps + 200ps = 1250ps

1.3 （3）

> 如果我们将数据通路中的一个流水级拆成两个新的流水级，每一个新的流水级的延迟是原来的一半，那么我们将拆分哪一级？新处理器的时钟周期是多少？

- 拆分延迟最大的一级，在此题中即为ID，拆分后，每个新阶段的延迟为175ps
- 新时钟周期为max=300ps

1.4 （4）

> 假设没有停顿或冒险，数据存储器的利用率如何？

- 数据存储器在 Load 和 Store 指令的 MEM 阶段被使用，因此利用率为20% + 15% = 35%

1.5 （5）

> 假设没有停顿或冒险，寄存器堆的写端口利用率如何？

- 寄存器堆的写端口在 WB 阶段被使用，需要WB阶段的指令有 ALU/Logic、Load指令，因此利用率为45% + 20% = 65%

2. T2

> 对于如下的 RSIC-V 的汇编片段：

```
sd x29,12(x16)
ld x29,8(x16)
sub x17,x15,x14
beqz x17,label
add x15,x11,x14
sub x15,x30,x14
```

假设我们修改流水线使得其只有一个存储器（存放指令和数据）。在这种情况下，每次程序在另一个指令访问数据的同一周期内获取指令时，都会存在结构冒险。

2.1 （1）

> 请画出流水线图，说明以上代码会在何处停顿

	IF	ID	EXE	MEM	WB
1	sd				
2	ld	sd			
3	sub	ld	sd		
4	**nop**	sub	ld	**sd**	
5	**nop**	**nop**	sub	**ld**	sd
6	beqz	**nop**	**nop**	sub	ld
7	add	beqz	**nop**	**nop**	sub
8	sub	add	beqz	**nop**	**nop**
9		sub	add	beqz	**nop**
10			sub	add	beqz
11				sub	add

sd和ld需要访问存储器，而同一周期内还需要取指令，因此会产生停顿

2.2 （2）

> 是否可通过重排代码来减少因结构冒险而导致的停顿次数？

不能

2.3 （3）

> 该结构冒险必须用硬件来解决吗？我们可以通过在代码中插入 NOP 指令来消除数据冒险，对于结构冒险是否可以相同处理？请解释原因。

必须硬件，对于结构冒险，有可能CPU读取不到指令，因此插入了NOP也无法读取到，只能在硬件层面上进行处理，以确保资源的正确分配和利用

2.4 （4）

> 在典型程序中，大约需要为该结构冒险产生多少时钟周期的停顿？（使用以下指令分布）

R-type/I-type (non-ld)	ld	sd	beq
52%	25%	11%	12%

Load 和 Store 会产生结构冒险，占 36%，因此，产生 36% 的时钟周期的停顿

3. T3

> 如果我们改变 load/store 指令格式，使用寄存器（不需要立即数偏移）作为访存地址，这些指令就不再需要使用 ALU 。这样的话，MEM 阶段和 EX 阶段就可以重叠，流水级数变为四级。

3.1 (1)

> 流水级数的减少会影响时钟周期吗？

不一定

时钟周期取决于流水级中延迟最大的阶段。 在EX 和 MEM 阶段重叠后，如果重叠后的流水级延迟仍然不是最大延迟，那么时钟周期将不受影响。如果重叠后的流水级延迟比其他阶段的延迟大，那么时钟周期可能会受到影响

3.2 (2)

> 这样的变化可能会提高流水线的性能吗？

可能会提高流水线的性能

1. 减少了流水线阶段的数量，从而减小了冒险发生的可能性
2. 通过使用寄存器作为访存地址，减少了对ALU的需求，可以提高指令的吞吐量和执行速度

3.3 (3)

> 这样的变化可能会降低流水线的性能吗？

可能会降低流水线的性能

1. Load/Store 指令不再使用立即数偏移， 可能需要额外的指令来计算偏移，有可能降低性能
2. 如果寄存器的数量有限，可能会限制并行执行，从而降低流水线的性能

4. T4

> 考虑如下循环：
>
> ```
> Loop:
> ld x10,0(x13)
> ld x11,8(x13)
> add x12,x10,x11
> subi x13,x13,16
> bnez x12,Loop
> ```
>
> 如果使用完美的分支预测（即没有控制冒险带来的流水线停顿），流水线中没有使用延迟槽，采用硬件前递解决数据冒险，分支指令在 EX 阶段判断是否跳转。

4.1 （1）

> 给出该循环中前两次循环的流水线执行图

	IF	ID	EXE	MEM	WB
1	ld1				
2	ld2	ld1			
3	add	ld2	ld1		
4	subi	add	ld2	ld1	
5	bnez	subi	add	ld2	ld1
6	ins1	bnez	subi	**add**	ld2
7	ins2	ins1	bnez	**subi**	add
8	ld1	**nop**	**nop**	**bnez**	subi
9	ld2	ld1	**nop**	**nop**	**bnez**
10	add	ld2	ld1	**nop**	**nop**
11	subi	add	ld2	**ld1**	**nop**
12	bnez	subi	add	ld2	ld1

4.2 （2）

> 标注出没有进行有用操作的流水级。当流水线全负荷工作时，所有五个流水级都在进行有用的操作的情况多久会出现一次？（从 subi 指令进入 IF 阶段开始计算，到 bnez 指令进入 IF 阶段结束）

- 标注如上图红色
- 6个周期出现一次