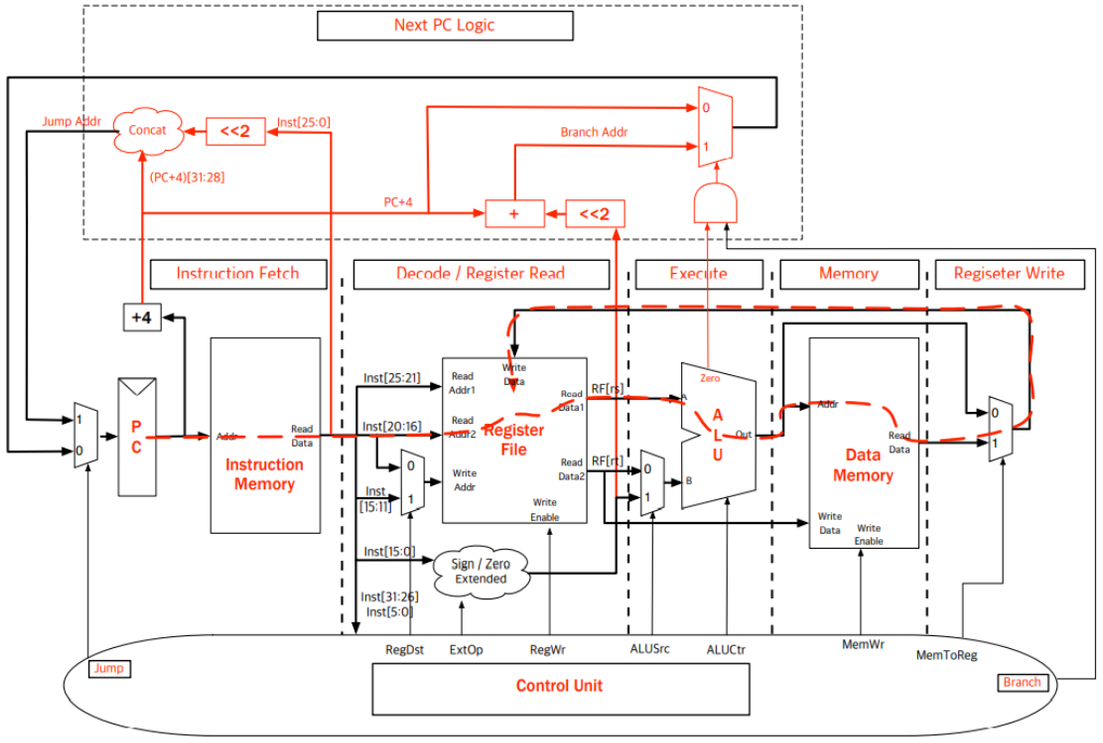
处理器设计（作业）

1. 单周期处理器控制逻辑

这是一个单周期处理器的数据通路（single cycle CPU diagram）：



在下表中填写出上图中各个控制信号的数值：

Instrs.			Control Signals						
			RegDst	ExtOp	ALUSrc	ALUCtr	MemWr	MemtoReg	RegWr
add			1	X	0	0010	0	0	1
ori			0	0	1	0001	0	0	1
lw			0	1	1	0010	0	1	1
sw			X	1	1	0010	1	X	0
beq			X	1	0	0110	0	X	0
j			X	X	X	XXXX	0	X	0

这个表格给出了算术逻辑单元每个操作的 ALUCtr 值：

Operation	AND	OR	ADD	SUB	SLT	NOR
ALUCtr	0000	0001	0010	0110	0111	1100

2. 单周期处理器的性能分析

时钟分析方法：

- 每个状态元件的输入信号必须在时上升沿之前稳定下来。
- 关键路径（critical path）：电路中状态元件之间最长的延迟路径。
- $t_{clk} \geqslant t_{clk\text{-}to\text{-}q} + t_{CL} + t_{setup}$，其中 t_{CL} 是组合逻辑中的关键路径
- 如果我们把寄存器放在关键路径上，我们可以通过减少寄存器之间的逻辑量来缩短周期。

电路元件的延时如下所示：

Element	Register clk-to-q	Register Setup	MUX	ALU	Mem Read	Mem Write	RegFile Read	RegFile Setup
Parameter	$t_{clk\text{-}to\text{-}q}$	t_{setup}	t_{mux}	t_{ALU}	$t_{MEMread}$	$t_{MEMwrite}$	t_{RFread}	$T_{RFsetup}$
Delay(ps)	30	20	25	200	250	200	150	20

关于硬件中的时钟的一些术语说明：

- 时钟（CLK）：使系统同步的稳定方波
- 启动时间（setup time）：在时钟边沿之前，输入必须稳定的时间
- 保持时间（hold time）：在时钟边沿之后，输入必须稳定的时间
- “CLK-to-Q”延迟（“CLK-to-Q” delay）：从时钟边沿测量，改变输出需要多长时间
- 周期（period）=最大延迟=“CLK-to-Q”延迟+CL 延迟+启动时间
- 时钟频率=1/周期（即周期的倒数）

回答问题：

1) 用到关键路径（critical path）的指令是哪一条？

涉及到内存的指令时间显然更长。就写数据 sw 和读数据 lw 而言，lw 比 sw 多了一步写回寄存器的操作，且读内存的延迟高于写内存，因此时间更长。所以用到关键路径的指令应该是 lw。

2) 最小时钟周期 t_{clk}是多少？最大时钟频率 f_{clk}是什么？假设 $t_{clk\text{-}to\text{-}q}$ > 保持时间(hold time).

最小时钟周期：

lw 使用了一次选路器，一次 ALU 运算，一次读寄存器，一次读内存，一次写寄存器。

$$t_{CL} = t_{RFread} + t_{mux} + t_{ALU} + t_{MEMread} + t_{RFsetup} = 150\,\text{ps} + 25\,\text{ps} + 200\,\text{ps} + 250\,\text{ps} + 20\,\text{ps} = 645\,\text{ps}$$

$$t_{clk} = t_{clk-to-q} + t_{CL} + t_{setup} = 30\,\text{ps} + 645\,\text{ps} + 20\,\text{ps} = 695\,\text{ps}$$

最大时钟频率：

$$f_{clk} = 1/t_{clk} = 1/695\,\text{ps} = 1.44 \times 10^{9}\,\text{Hz} = 1.44\,\text{GHz}$$

3. 流水线处理器设计（Pipelined CPU Design）

现在，我们将使用流水线方法来优化一个单周期处理器。流水线虽然增加了单个任务的延迟，但它可以减少时钟周期，提高吞吐量。 在流水线处理器中，多条指令重叠执行，体现了指令级并行性。

为了设计流水线，我们已经将单周期处理器分成五个阶段，在每两个阶段之间增加流水段寄存器。

接下来进行性能分析:
我们将使用与上一题相同的时钟参数:

Element	Register clk-to-q	Register Setup	MUX	ALU	Mem Read	Mem Write	RegFile Read	RegFile Setup
Parameter	$t_{clk\text{-}to\text{-}q}$	t_{setup}	t_{mux}	t_{ALU}	$t_{MEMread}$	$t_{MEMwrite}$	t_{RFread}	$T_{RFsetup}$
Delay(ps)	30	20	25	200	250	200	150	20

回答问题:

1) 这个五阶段流水线处理器的最小时钟周期长度和最大时钟频率分别是多少?

最小时钟周期:

在 MEM 阶段读内存时，花费的时间最长。因此:

$$t_{clk} = t_{clk-to-q} + t_{MEMwrite} + t_{setup} = 30\,\text{ps} + 200\,\text{ps} + 20\,\text{ps} = 250\,\text{ps}$$

最大时钟频率:

$$f_{clk} = 1/t_{clk} = 1/250\,\text{ps} = 4\,\text{GHz}$$

2) 相比于单周期处理器，性能加速比（speed up）是多少？为什么加速比会小于 5?

加速比为 $speed\ up = \dfrac{695\,\text{ps}}{250\,\text{ps}} = 2.78$。

因为五级流水中所需时间最长的阶段（Mem 阶段）比总时间的五分之一要长，而这一阶段需要作为流水处理器的时间周期的长度。

4. 控制冒险（Control Hazard）

遇到 branch 和 jump 指令的时候会发生控制冒险。我们可以通过暂停流水线来解决。但是，由于分支条件是在执行阶段计算的，流水线需要停顿三个周期。我们可以在寄存器读取阶段增加一个分支比较器，并引入一个转移延迟槽（delayed slot），使分支语句(branch)后的指令总是会被执行。

问题:

考虑填充转移延迟槽，我们需要重新排列以下几组指令，如果实在找不到指令填充延迟槽，你可能需要插入一条 nop 指令。

Set 1	Reordered set 1	Set 2	Reordered Set 2
addiu $t0, $t1, 5	addiu $t0, $t1, 5	addiu $t0, $t1, 5	addiu $t0, $t1, 5
ori $t2, $t3, 0xff	beq $t0, $s0, label	ori $t2, $t3, 0xff	ori $t2, $t3, 0xff
beq $t0, $s0, label	ori $t2, $t3, 0xff	beq $t0, $t2, label	beq $t0, $t2, label
lw $t4, 0($t0)	lw $t4, 0($t0)	lw $t4, 0($t0)	nop
			lw $t4, 0($t0)

5. 转移延迟槽

考虑以下两种设计:

- 第一种设计为每一条 branch 指令设计两个转移延迟槽（delay slots)，但不使用转移预测（branch prediction),而是在编译时调度可用的指令填充转移延迟槽。假设其中30%的 branch 指令在编译时能找到指令将两个延迟槽填满，60%的 branch 指令在编译时只能找到指令填充一个延迟槽，剩下 10%的 branch 指令的两个延迟槽在编译时无法填充。
- 第二种设计不采用转移延迟槽 (delay slots)，而是采用转移预测 (Branch Prediction)。转移预测错误的开销（mis-prediction penalty)是 3 个周期。Branch 指令本身需要花一周期执行，但如果转移预测错误，就会增加 3 个额外周期的开销。

如果需要第二种设计能达到第一种设计的性能，转移预测的准确度应该为多少？

使用转移延迟槽时，30%的概率不增加额外周期开销，60%的概率增加一个额外周期开销，10%的概率增加两个额外周期开销。平均增加 60%+2*10%=0.8 个周期开销。

如果采用转移预测，那么想要达到同样的性能，那么预测失败的概率应该为 0.8/3=26.7%
因此预测的准确度应该为 73.3%才能使二者有相同的性能。

6. 指令调度

假定在一个有转发（forwarding）功能的五段流水线中执行以下程序段，则可以怎样调整以下指令序列使其性能达到最好?

```
1    lw    $2, 100($6)
2    add   $2, $2, $3
3    lw    $3, 200($7)
4    add   $6, $4, $7
5    sub   $3, $4, $6
6    lw    $2, 300($8)
7    beq   $2, $8, Loop
```

具有 forwarding 功能的流水线只需要注意避免 Read-use 的 hazard 就能规避数据冒险。

```
1    lw    $2,  100($6)
2    add   $6,  $4,  $7
3    add   $2,  $2,  $3
4    lw    $3,  200($7)
5    lw    $2,  300($8)
6    sub   $3,  $4,  $6
7    beq   $2,  $8,  Loop
```

7. 中断

- 当MIPS处理器在执行一条除法指令时，发生了除数为0异常（exception）。那么此时处理器就要进行中断处理。在中断处理过程中，最开始的一部分工作由硬件完成，描述一下:

1）中断处理开始的阶段，硬件需要完成哪些工作，保存哪些状态？

需要在EPC中保存发生异常的指令的地址;
设置STATUS寄存器的一个控制位SR(EXL)，以强迫CPU进入内核态，关闭中断;
在Cause寄存器中保存异常产生的原因;
CPU将PC的值设为80000180H。

2）如果中断处理程序（interrupt handler）需要读寄存器R5,R6,R7， 写寄存器R5,R8,R10,那么中断处理程序应该在一开始保留哪几个寄存器的值？

应该保留R5,R6,R7,R8,R10

3）ERET（中断返回）指令会触发硬件完成哪些动作？

将EPC中的内容移入PC中，并将SR(EXL)清零，允许中断响应，并进入用户态。

如果一条指令在执行阶段，即发生了“指令地址错误“异常，又发生了”ALU运算溢出“异常，那么这条指令被中断时，原因寄存器（cause register）中记录的中断原因，应该是哪一个？

应该是“指令地址错误异常”