第一题

① 对上述指令而言，图 1 中的控制单元要产生哪些控制信号？

	指令	解释
(a)	`add Rd,Rs,Rt`	`Reg[Rd] = Reg[Rs] + Reg[Rt]`
(b)	`lw Rt,Offs(Rs)`	`Reg[Rt] = Mem[Reg[Rs] + Offs]`

	RegDst	RegWrite	MemRead	ALUMux	MemWrite	ALUOp	RegMux	Branch
(a)	1	1	0	0(Reg)	0	Add	0	0
(b)	0	1	1	1(imm)	0	Add	1	0

ALUMux代表参与运算的数据，**ALUMux = 1**时由**imm**参与运算，否则由**Reg[Rt]**参与运算

RegMux代表输入寄存器堆的数据，**RegMux = 1**时由**Alu**的输出的数据输入，**RegMux = 0**时由**Memory**的数据输入

② 对上述指令而言，要用到哪些功能单元？

(a) PC、指令存储器、寄存器堆、ALU, 控制器, PCplus4加法器

(b) PC、指令存储器、寄存器堆、ALU、数据存储器、控制器

③ 哪些功能单元会产生输出，但输出不会被以上指令用到？对以上指令而言，哪些功能单元不产生任何输出？

	产生输出但没有用到	没有输出
(a)	分支加法器	数据存储器
(b)	分支加法器，寄存器堆的 $RD2$	无

④ 对一条 MIPS 的与指令 (And) 而言，关键路径是什么？

(a) 指令存储器、寄存器堆、多选器、ALU、多选器、寄存器堆

(b) 指令存储器、寄存器堆、多选器、ALU、多选器、寄存器堆

⑤ 对一条 MIPS 的装载指令 (LW) 而言，关键路径是什么？

(a) 指令存储器、寄存器堆、多选器、ALU、数据存储器、多选器、寄存器堆

(b) 指令存储器、寄存器堆、多选器、ALU、数据存储器、多选器、寄存器堆

⑥ 对一条 MIPS 的相等则分支指令 (BEQ) 而言，关键路径是什么？

(a) 指令存储器、寄存器堆、多选器、ALU、多选器

(b) 指令存储器、寄存器堆、多选器、ALU、多选器

第二题

① 对上述指令而言，哪些已有的单元还可以被使用？

(a) PC、指令存储器、寄存器堆、ALU, 控制器, PCplus4加法器

(b) PC、指令存储器、寄存器堆、ALU, 控制器, PCplus4加法器

② 对上述指令而言，还需要增加哪些功能单元？

(a) 寄存器堆加一个 `Rx` 输入接口和 `RD3` 输出接口，ALU 加一个运算接口和三数据运算功能

(b) 在 ALU 增加移位功能和 `shamt` 输入接口，以便直接和 `Rt` 参与运算

③ 为了支持这些指令，需要在控制单元增加哪些信号？

(a) 在 $AluCtrl$ 增加一个 `add3` 的编码，指定 ALU 运算方式

(b) 在 $AluCtrl$ 增加一个 `sll` 的编码，指定 ALU 运算方式

④ 改进前后的时钟周期分别是多少？

数据通路关键路径：指令存储器 $\rightarrow$ 寄存器堆 $\rightarrow$ 多选器 $\rightarrow$ ALU $\rightarrow$ 数据存储器 $\rightarrow$ 多选器 $\rightarrow$ 寄存器堆

$T = 400 + 200 + 30 + 120 + 350 + 30 + 200 = 1330\ ps$

(a) 加法器不在关键路径，没有改变

(b) 寄存器在关键路径，时钟周期得 $1330 + 100 \times 2 = 1530\ ps$

⑤ 改进后将获得多大的加速比？

(a) 由于没有改变最短时钟周期，所以加速比为 1

(b) $\frac{1330}{1530} = 0.915$

⑥ 比较改进前后的性能 价格比，进行这样的改进是否有意义？

(a) 由于性能没有任何改变，而且也提高了成本，所以没有意义

$$\frac{\text{新成本}}{\text{旧成本}} = \frac{1000 + 2 \times 30 + 3 \times 10 + 100 + 200 + 2000 + 500 + 2 \times 20}{1000 + 2 \times 30 + 3 \times 10 + 100 + 200 + 2000 + 500} = 1.010$$

$$\frac{\text{性能}}{\text{价格比}} = \frac{1.000}{1.010} = 0.990$$

(b) 性能下降且成本增高，所以也没有意义

$$\frac{\text{新成本}}{\text{旧成本}} = \frac{3890 + 200}{3890} = 1.051$$

$$\frac{\text{性能}}{\text{价格比}} = \frac{0.915}{1.051} = 0.871$$

第三题

① 如果处理器只需做连续取指这一件事，那么时钟周期是多少？

(a) $400\ ps$

(b) $500\ ps$

② 考虑一个与图 3 类似的数据通路，但是假设处理器只需处理无条件相对跳转指令，那么时钟周期是多少？

(a) 指令存储器 $\rightarrow$ 符号扩展 $\rightarrow$ 左移两位 $\rightarrow$ ALU $\rightarrow$ 多选器

$400 + 20 + 2 + 120 + 30 = 572\ ps$

(b) 指令存储器 $\rightarrow$ 符号扩展 $\rightarrow$ 左移两位 $\rightarrow$ ALU $\rightarrow$ 多选器

$500 + 90 + 20 + 180 + 100 = 890\ ps$

③ 同样考虑 一个与图 3 类似的数据通路， 但这次假设只需处理有条件相对跳转指令，那么时钟周期是多少？ 请注意图 3 中 ALU 的零输出端不是与数据存储器连接，该输出与选择 PC 值来源的多选器的控制有关

(a) 指令存储器 $\rightarrow$ 寄存器堆 $\rightarrow$ 多选器 $\rightarrow$ ALU $\rightarrow$ 多选器

$400 + 200 + 30 + 120 + 30 = 780\ ps$

(b) 指令存储器 $\rightarrow$ 寄存器堆 $\rightarrow$ 多选器 $\rightarrow$ ALU $\rightarrow$ 多选器

$500 + 220 + 100 + 180 + 100 = 1100\ ps$

④ 哪些类型的指令需要该单元？

(a) 所有指令

(b) 有关读出和写入数据存储器的指令，例：`lw,sw`

⑤ 对哪些类型的指令而言，该单元位于关键路径上？

(a) 所有指令都不会在他的关键路径上，因为指令存储器的延迟比加法器慢

(b) 有关读出和写入数据存储器的指令，例：`lw,sw`

⑥ 假设仅需支持 `beq` 指令和 `add` 指令，讨论该单元的延迟变化对处理器时钟周期的影响。假设其他单元的延迟不变。

(a) 当加法器的的延迟大于指令存储器，则必定影响 CPU 的时钟周期

(b) 由于数据存储器不在 `beq` 和 `add` 的使用范围，所以该元件的延迟不影响时钟周期

第四题

① 如果仅需支持 ALU 类指令（如 `add`、`and` 等），处理器的时钟周期是多少

(a) 指令存储器 → 寄存器堆 → 多选器 → ALU → 多选器 → 寄存器堆

$400 + 200 + 30 + 120 + 30 + 200 = 980\ ps$

(b) 指令存储器 → 寄存器堆 → 多选器 → ALU → 多选器 → 寄存器堆

$500 + 220 + 100 + 180 + 100 + 220 = 1320\ ps$

② 如果仅需支持 `lw` 类指令，时钟周期是多少？

(a) 指令存储器 → 寄存器堆 → 多选器 → ALU → 数据存储器 → 多选器 → 寄存器堆

$400 + 200 + 30 + 120 + 350 + 30 + 200 = 1330\ ps$

(b) 指令存储器 → 寄存器堆 → 多选器 → ALU → 数据存储器 → 多选器 → 寄存器堆

$500 + 220 + 100 + 180 + 1000 + 100 + 220 = 2320\ ps$

③ 如果必须支持 `add`、`beq`、`lw` 和 `sw` 指令，时钟周期是多少？

`lw` 指令的关键路径最长，所以时钟周期是

(a) $1330\ ps$

(b) $2320\ ps$

④ 数据存储器平均用了多少时钟周期？

(a) $20\% + 10\% = 30\%$

(b) $35\% + 15\% = 50\%$

⑤ 符号扩展电路的输入平均用了多少时钟周期？在未用到该输入的其他时间，符号扩展电路在做什么？

(a) $15\% + 20\% + 20\% + 10\% = 65\%$

(b) $5\% + 15\% + 35\% + 15\% = 70\%$

⑥ 如果可以将数据通路上某个单元的延迟减少 10%，应该减少哪个单元的延迟？改进后整个处理器的加速比是多少？

`lw` 指令的关键路径最长，所以时钟周期是

(a) 指令存储器的延迟最长，所以将他的延迟减少 10% 至 $360\ ps$ 可以使整个处理器的时钟周期从 $1330\ ps$ 变成 $1290\ ps$，加速比为

$$\frac{1330}{1290} = 1.031$$

(b) 数据存储器的延迟最长，所以将他的延迟减少 10% 至 $900\ ps$ 可以使整个处理器的时钟周期从 $2320\ ps$ 变成 $2220\ ps$，加速比为

$$\frac{2320}{2220} = 1.045$$

第五题

① 设这样测试处理器的缺陷：先给 PC 、寄存器堆、数据和指令存储器中设置一些值（可以自己选择），执行一条指令，然后读出 PC 、寄存器堆和存储器中的值；最后检查这些值以判断处理器中是否存在缺陷。你能设计这样一个方案检查该信号上是否有固定为 0 缺陷吗？

(a) 利用 `ori $t0, $0, 0x0080` ，如果输出信号第 7 位有固定为 0 的缺陷，则 `$t0` 的值是 0 而不是 `0x00000080` 。

(b) 首先存储器的数据都清 0，若 `lw $t0, 4($0)` 使 `$t0` 写入的是 4 而不是 0，则表示 `MemtoReg` 信号有固定为 0 的缺陷。

② 条件同第 ① 问，但是这次检查固定为 1 缺陷。你能只设计一个测试方案同时检查固定为 0 缺陷和固定为 1 缺陷吗？如果可以，请解释如何实现；如果不能，请说明理由。

(a) 利用 `ori $t0, $0, 0` ，如果输出信号第 7 位有固定为 1 的缺陷，则 `$t0` 的值是 `0x00000080` 而不是 `0x00000000`

(b) 这个信号的固定为 1 缺陷不能准确检查出来，因为所有能够将 `MemtoReg` 信号设置为 0 的指令都会同时将 `MemRead` 信号设置为 0，这样会导致写入到寄存器中的数据是不确定的，有可能最后出现在寄存器中的数据是与原来寄存器中的数据相同的，从而检测不出来固定为 1 缺陷。

③ 如果我们知道一个处理器在该信号上有一个固定为 1 缺陷，它还能用吗？为了使这个处理器仍然可用，我们必须将原来能在正常 $MIPS$ 处理器上运行的程序做一些变换，使之可以在这个处理器上运行。假设指令存储器和数据存储器都很大，足够容纳变换后的程序。提示：将因为该缺陷不能用的指令替换为一系列能用的指令，这一系列指令与原指令功能相同。

(a) 可以避开指令存储器输出信号固定为 1 的缺陷，找到在输出信号第 7 位上为 0 的所有指令，然后将这些指令中的立即数进行相应的替换。

(b) `MemtoReg` 是 `load` 指令的必要信号，若这个信号固定为 1 会一直让数据存储器的数据传到寄存器堆内。

④ 条件同第 ① 问，这次检测控制信号 `MemRead` 是否存在上表中的缺陷？

(a) `MemRead` 的缺陷不能准确检测出来，因为在 `load` 数据传输还得经过 `MemtoReg` 才能将数据写入寄存器堆，所以 `MemRead` 固定为 1 也不会影响 CPU 的运作。

(b) `MemRead` 的缺陷不能准确检测出来，因为指令 `[31:26]` 都为 0 的指令都是 R 指令，不是 `load` 指令，所以 `MemRead` 为 0 也是必要的，因此指令的第 `[31:26]` 位全为 0，则固定为 0，不影响 CPU 的运作。

⑤ 条件同第 ① 问，这次检测控制信号 `Jump` 是否存在上表中的缺陷？

(a) 只需放一条非跳转指令例如 `add $1,$0,$1` 在指令存储器的第一个位置，若执行后的 PC 值不为 $PC+4$ 则表示 `Jump` 信号存在固定为 1 缺陷。

(b) `Jump` 的缺陷不能准确检测出来，因为指令 `[31:26]` 都为 0 的指令都是 R 指令，不是 `load` 指令，所以 `MemRead` 为 0 也是必　　要的，因此指令的第 `[31:26]` 位全为 0，则固定为 0，不影响 CPU 的运作。

⑥ 使用第 ① 问中描述的测试方案，可以一次对几个不同的信号进行测试，但一般来说不可能同时测试到所有信号。试着设计一系列方案对所有多选器输出的上表中的缺陷进行测试 五个多选器输出的每一位都要测试到。尽量使用较少的测试方案。

(a) `RegDst` 信号若固定为 1 则利用 `ori $1, $0, 0x1000` 若写入的字在 `$2` 而不是 `$1` 则表示 `RegDst` 存在固定为 1 的缺陷；只需放一条非跳转指令例如 `add $1,$0,$1` 在指令存储器的第一个位置，若执行后的 PC 值不为 $PC+4$ 则表示 `Jump` 信号存在固定为 1 缺陷；只需放一条非分支且满足分支的指令例如 `add $1,$0,$0` 在指令存储器的第一个位置，若执行后的 PC 值不为 $PC+4$ 则表示 `Branch` 信号存在固定为 1 缺陷； `MemtoReg` 固定为 1 的缺陷不能准确检测，因为所有能够将 `MemtoReg` 信号设置为 0 的指令都会同时将 `MemRead` 信号设置为 0， 导致写入的数据是不确定的 `AluSrc` 固定为 1 的缺陷可以用 `add $1,$0,$0` 测出，若写入 `$1` 的值为 `0x00000820` 而不是 0 则表示 `AluSrc` 存在固定为 1 的缺陷。

(b) `RegDst` 信号若在指令 `[31:26]` 位为 0 固定为 0，则可用 `add $1,$2,$3` 测出，若写入的值出现在 `$3` 而不是 `$1` 则表示 `RegDst` 存在此缺陷。其他信号则测不出来，因为 `Jump,Branch,MemtoReg,AluSrc` 的指令 `[31:26]` 位都不完全为 0

第六题

① 为了避免增加图 4 中数据通路的关键路径长度，留给控制单元产生 `MemWrite` 信号的时间有多少？

(a) 从指令存储器到数据存储器的关键路径为：指令存储器、寄存器堆、多选器、ALU、数据存储器、多选器、寄存器堆。

时钟周期为 $400 + 200 + 30 + 120 + 350 + 30 + 200 = 1330\ ps$

要在下一个时钟上升沿到来前产生 `MemWrite` 信号，所以 $1330 - 400 - 350 = 580\ ps$

(b) 从指令存储器到数据存储器的关键路径为：指令存储器、寄存器堆、多选器、ALU、数据存储器、多选器、寄存器堆。

时钟周期为 $500 + 220 + 100 + 180 + 1000 + 100 + 220 = 2320\ ps$

要在下一个时钟上升沿到来前产生 `MemWrite` 信号，所以 $2320 - 500 - 1000 = 820\ ps$

② 图 4 中哪个控制信号最不关键，控制单元需要在多长时间内产生该信号以避免其成为关键路径？

`Jump` 信号，因为 `Jump` 信号在时钟周期的最后才会用到，并且在同一个类型的 `MemWrite` 和 `RegWrite` 中的元件延迟最短，所以 `Jump` 具有最长的松弛时间。

(a) 松弛时间为：$1330 - 400 - 30 = 900\ ps$

(b) 松弛时间为：$2320 - 500 - 100 = 1720\ ps$

③ 图 4 中哪个控制信号最关键，控制单元需要在多长时间内产生该信号以避免其成为关键路径？

(a) `AluOp` 信号，因为 `AluOp` 信号必须在数据到达 ALU 前产生，所以松弛时间为 $200 + 30 - 50 = 180\ ps$

(b) `AluSrc` 信号，因为 `AluSrc` 必须在 `Read Data 2` 和 `imm32` 数据来之前到达多选器，所以松弛时间为 $220\ ps$

④ 处理器的时钟周期为多少？

(a) 因为 `MemWrite` 信号延迟超过 $580\ ps$，所以时钟周期需增加 $710 - 580 = 130\ ps$，因此时钟周期为 $1330 + 130 = 1460\ ps$

(b) 因为 `MemWrite` 信号延迟超过 $820\ ps$，所以时钟周期需增加 $1500 - 820 = 680\ ps$，因此时钟周期为 $2320 + 680 = 3000\ ps$

⑤ 如果你可以加速控制信号的产生，但加快一个控制信号 5 ps 的代价是处理器成本增加 1 元。那么为了最大化性能你会加速哪些控制信号？这种性能改进的最小代价是多少？

各个控制信号的周期

	RegDst	Jump	Branch	MemRead	MemtoReg	ALUOp	MemWrite	ALUSrc	RegWrite
松弛时间	700 ps	900 ps	870 ps	350 ps	700 ps	180 ps	580 ps	200 ps	730 ps
a	720 ps	730 ps	600 ps	400 ps	700 ps	200 ps	710 ps	200 ps	800 ps
延迟	20 ps	-170 ps	-270 ps	50 ps	0 ps	20 ps	130 ps	0 ps	70 ps

	RegDst	Jump	Branch	MemRead	MemtoReg	ALUOp	MemWrite	ALUSrc	RegWrite
松弛时间	1500 ps	1720 ps	1620 ps	500 ps	1500 ps	135 ps	820 ps	220 ps	1600 ps
b	1600 ps	1600 ps	1400 ps	500 ps	1400 ps	400 ps	1500 ps	400 ps	1700 ps
延迟	100 ps	-120 ps	-220 ps	0 ps	-100 ps	265 ps	680 ps	180 ps	100 ps

只需加快影响时钟周期的控制信号

	对时钟周期有影响的信号	代价
a	`RegDst(+20 ps)` `MemRead(+50 ps)` `ALUOp(+20 ps)` `MemWrite(+130 ps)` `RegWrite(+70 ps)`	$\frac{290}{5} = 58$
b	`RegDst(+100 ps)` `ALUOp(+135 ps)` `MemWrite(+680 ps)` `AluSrc(+180 ps)` `RegWrite(+100 ps)`	$\frac{1195}{5} = 239$

⑥ 如果一个处理器的成本已经很高，那么我们需要在维持处理器性能的同时降低其成本，而不是像第 ⑤ 问中所作的那样为提高它的性能而买单。如果你可以使用更慢的逻辑来实现对信号的控制，并且单个控制信号每减慢 5 ps ，处理其成本就可以节省 1 元，那么在保持处理器性能的同时，你会减慢哪些控制信号，

最大正值的信号决定了周期，为了减慢不影响时钟周期的控制信号，可以将所有的信号都减慢到具有跟该信号相同的值

	RegDst	Jump	Branch	MemRead	MemtoReg	ALUOp	MemWrite	ALUSrc	RegWrite
减慢数值(a)	110 ps	300 ps	400 ps	80 ps	130 ps	110 ps	0 ps	130 ps	60 ps
减慢数值(b)	580 ps	800 ps	900 ps	680 ps	780 ps	545 ps	0 ps	500 ps	580 ps