厦门大学

《计算机组成原理》
课程实验报告

姓名：苏一涵

学院：信息学院

系：软件工程系

专业：软件工程

学号：36720232204041

2025 年 5 月 21 日

第 7 次实验　流水线 CPU 设计

1. 实验目的

(1) 了解理想流水线 MIPS 处理器设计的基本原理，在该流水线上运行测试程序。

(2) 了解气泡流水线 MIPS 处理器设计的基本原理，在该流水线上运行测试程序。

(3) 了解重定向流水线 MIPS 处理器设计的基本原理，在该流水线上运行测试程序。

(4) 了解动态分支预测流水线 MIPS 处理器设计的基本原理，在该流水线上运行测试程序。

2. 实验环境

(1) Windows 系统下运行 Logisim 软件（需安装 JDK）。

(2) MARS 4.5 汇编工具。

3. 实验内容

(1) 在理想流水线 MIPS 处理器上运行测试程序。

(2) 在气泡流水线 MIPS 处理器上运行测试程序。

(3) 在重定向流水线 MIPS 处理器上运行测试程序。

(4) 在动态分支预测流水线 MIPS 处理器上运行测试程序。

回答问题：

(1) 理想流水线测试程序 test1-1.asm 中， 下述指令：

addi $s0, $zero, 0

ori $s0,$s0, 0

sw $ s0, 0($s0)

这三条指令为何不存在相关性?

答：对三条指令进行分析如下：

addi $s0,$zero,0 是将 $s0 赋值为 0。

ori $s0,$s0,0 是对 $s0 进行逻辑或操作（0 OR 0），结果仍为 0，未改变 $s0 的值。

sw $s0,0($s0)是将 $s0 的值（0）存入对应存储单元。

虽然 ori $s0,$s0,0 读取了 addi $s0,$zero,0 的结果，但 ori 操作后 $s0 仍为 0，未影响 sw $s0,0($s0) 的执行结果（无论 $s0 是 addi 直接写入的 0，还是经 ori 操作后的 0，sw 指令的存储值均为 0）。

而由于 ori $s0,$s0,0 未改变 $s0 的值，未对后续 sw $s0,0($s0) 的执行结果产生实质影响，因此这三条指令不存在相关性。

(2) 理想流水线测试程序 test1-1.asm 中，

```
addi $v0,$zero,10
addi $s1,$zero,0
addi $s2,$zero,0
addi $s3,$zero,0
```

三条 addi 指令的作用是什么？为什么?

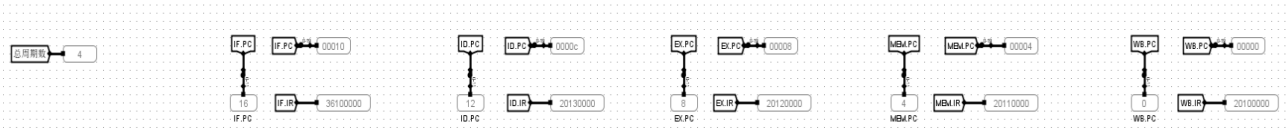
答：三条 addi 指令的作用是消除 syscall 指令与 addi $v0,$zero,10 指令之间的数据相关性，确保程序中所有指令均无相关性，以维持流水线的理想执行状态。

分析：addi $v0,$zero,10 指令给寄存器 $v0 赋值（用于设置 syscall 的停机操作），若 syscall 指令紧接其后，会形成数据相关（syscall 需读取 $v0 的值，即 RAW 相关）。

插入三条 addi 指令后，它们操作的寄存器（$s1、$s2、$s3）与 $v0 无关，且三条指令本身不依赖 addi $v0,$zero,10 的结果。这样，syscall 指令与 addi $v0,$zero,10 之间的数据相关被打破，使得程序中所有指令（包括 syscall）均无相关性，避免流水线因数据相关而停顿，保证流水线按理想状态（每周期执行一条指令）高效运行。

(3) 在理想流水线 MIPS 处理器的数据通路运行 test1-1.hex

- ①在数据通路的“指令存储器”中装入“test-1.hex”程序。按Ctrl+R，此时总周期数=0。
- ②接下去，按Ctrl+T二次，此时总周期数=1；再按Ctrl+T二次，此时总周期数=2；再按Ctrl+T二次，此时总周期数=3；再按Ctrl+T二次，此时总周期数=4。观看这期间IF.PC、IF.IR、ID.PC、ID.IR、EX.PC、EX.IR、MEM.PC、MEM.IR、WB.PC、WB.IR的变化情况，是不是与下一页的情况相同?

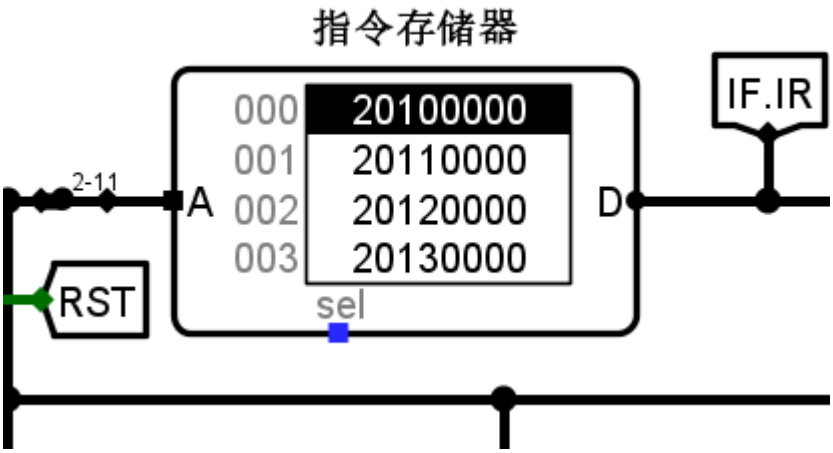


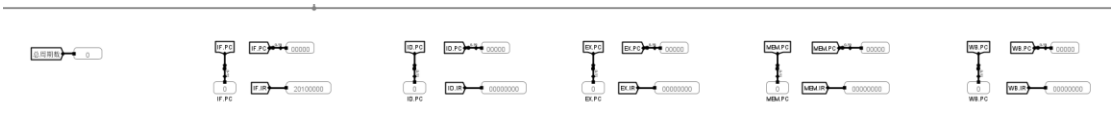
- ③接下去，按Ctrl+K，直到程序执行结束。观看总周期数是不是为21? 观看“数据存储器”0-3单元的内容是不是为0-3?

上文中下一页即为实验七课件第 17 页的显示结果，需结合自己实际验证结果，观察总周期数每次递增的时候，各级流水线寄存器值的变化，从而理解流水线寄存器的作用，可截屏佐证并伴文字说明。

答：验证如下

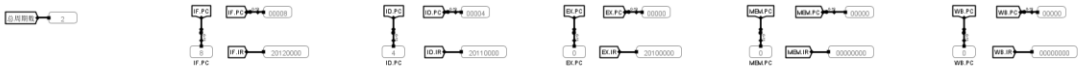
先加载 test1 的 hex 文件



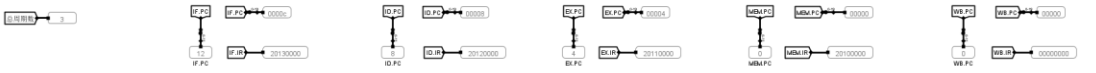


按 Ctrl+T 二次，此时总周期数=1，这期间 IF.PC、IF.IR、ID.PC、ID.IR、EX.PC、EX.IR、MEM.PC、MEM.IR、WB.PC、WB.IR 的变化情况如下

再按 Ctrl+T 二次，此时总周期数=2，这期间 IF.PC、IF.IR、ID.PC、ID.IR、EX.PC、EX.IR、MEM.PC、MEM.IR、WB.PC、WB.IR 的变化情况如下



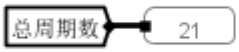
再按 Ctrl+T 二次，此时总周期数=3，这期间 IF.PC、IF.IR、ID.PC、ID.IR、EX.PC、EX.IR、MEM.PC、MEM.IR、WB.PC、WB.IR 的变化情况如下



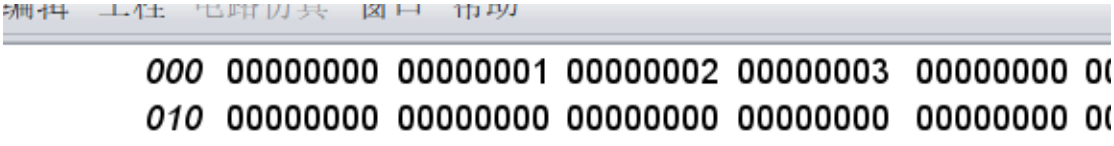
再按 Ctrl+T 二次，此时总周期数=4，这期间 IF.PC、IF.IR、ID.PC、ID.IR、EX.PC、EX.IR、MEM.PC、MEM.IR、WB.PC、WB.IR 的变化情况如下

接下去，按 Ctrl+K，直到程序执行结束

总周期数如下：



数据存储器 0-3 单元的内容为 0-3



由验证可以发现单周期 MIPS 处理器数据通路依次分成 5 个阶段：取指令 IF 阶段、译码取数 ID 段、指令执行 EX 段、访存 MEM 段、写回 WB 段。而改造成理想流水线，只需要加入流水寄存器部件即可。一共增加 IF/ID、ID/EX、EX/MEM、MEM/WB 共 4 个流水寄存器，数据通路被分为 5 段流水线。所有流水寄存器、程序计数器 PC、寄存器

堆、数据存储器均采用同一时钟进行同步，每来一个时钟就会有一条新的指令进入流水线取指令 IF 段，多个功能段并行执行，同一时刻有多条指令在流水线执行。ID 段译码生成该指令的所有控制信号，控制信号通过流水寄存器逐段向后传递，后段功能部件所需的控制信号不需要单独生成，只需要从流水寄存器获取即可。流水线数据通路由单周期数据通路改造而来，操作控制信号相同，因此流水线可以复用单周期处理器中的操作处理器。

(4) 在气泡流水线中，对于测试程序 test2-1.asm，如何消除掉控制相关的？提示：根据气泡流水线的下述控制信号并结合 test2-1.asm 中的某条分支指令（j,beq,bne,jal,jr）进行分析

```
IF/ID.CLR = BranchTaken                    #出现分支跳转时，清空IF/ID段流水寄存器
ID/EX.CLR = Flush = BranchTaken + DataHazzard    #出现分支跳转或数据相关时，清空ID/EX段流水寄存器
```

答：在气泡流水线中，每当分支指令确定跳转（BranchTaken = 1），通过清空 IF/ID 段流水寄存器（必要时清空 ID/EX 段），取消预取的无效指令，从而消除控制相关

如 j jmp_next1 指令：

```
j jmp_next1
```

当执行该指令且确定跳转（BranchTaken = 1）时，IF/ID.CLR 被置为 1，清空 IF/ID 段流水寄存器，使后续原本预取的 addi $s1,$zero,4 等指令不再继续执行，从而避免控制相关。

(5) Test2-1.asm 中的下述 nop 空操作的作用是什么？

```
jmp_func:
        addi $s1,$s1,-1
#       nop
#       nop
#       nop
        add $a0,$0,$s1                  #a0=s1
#       nop
#       nop
#       nop
        syscall                         #在数码管上从1F（32-1=31=1F）开始显示，接下去减1，直到显示0
        bne $s1,$zero,jmp_func
        jr $31                          #子程序返回指令
```

答：nop 空操作的作用是填充流水线延迟槽，避免数据相关。

在 addi $s1, $s1, -1 之后的 nop：addi 指令在流水线中执行时，结果不会立即写入寄存器（存在流水线阶段延迟）。若后续指令（如 add $a0, $0, $s1）立即使用 $s1 的值，会因数据未准备好而产生数据相关错误。nop 填充延迟槽，等待 addi 操作完成，确保 $s1 的值稳定后，add $a0, $0, $s1 再使用该值，避免数据相关导致的错误。

在 add $a0, $0, $s1 之后的 nop：同理，add 指令执行后，$a0 的值不会立即

准备好。nop 填充延迟槽，等待 add 操作完成，确保后续 syscall 指令使用 $a0 时，其值已正确更新，避免数据相关问题，保证程序正确执行。

(6) 针对 test2-2.asm，选择任意几条存在数据相关的指令，解释气泡流水线是如何消除这些相关性的?

答：以 sw $s1,0($s1) 和 lw $s2,0($s1) 以及 addi $s1,$s0,1 和 add $s0,$s0,$s1 为例，说明气泡流水线消除数据相关的过程：

1） sw $s1,0($s1) 和 lw $s2,0($s1)（RAW 相关）：

i：sw $s1,0($s1) 向内存地址 0($s1) 写入数据，lw $s2,0($s1) 紧接着从该地址读取数据。由于写操作未立即完成，存在数据相关（写后读）。

ii：气泡流水线通过控制信号 ID/EX.CLR = Flush = BranchTaken + DataHazzard，当检测到数据相关（DataHazzard = 1）时，清空 ID/EX 段流水寄存器。这使得 lw $s2,0($s1) 暂停执行，待 sw $s1,0($s1) 完成写操作后，再继续执行 lw $s2,0($s1)，从而消除数据相关。

2） addi $s1,$s0,1 和 add $s0,$s0,$s1（RAW 相关）：

i：addi $s1,$s0,1 向 $s1 写入新值，add $s0,$s0,$s1 需要读取 $s1 的新值。由于写操作未立即完成，存在数据相关（写后读）。

ii：气泡流水线检测到数据相关（DataHazzard = 1），ID/EX.CLR 被置位，清空 ID/EX 段流水寄存器。这使得 add $s0,$s0,$s1 暂停执行，待 addi $s1,$s0,1 完成写操作后，再继续执行 add $s0,$s0,$s1，消除数据相关。

综上，气泡流水线通过检测数据相关信号（DataHazzard），利用 ID/EX.CLR 清空对应流水寄存器，暂停后续相关指令执行，待前序指令完成写操作后，再继续执行，从而消除数据相关，确保程序正确运行。

(7) 针对测试程序 test2-3.asm 中的下述 5

```
and $s1,$s1,$s2      #s1 = s1 ^ s2   = 1 ^ 1          =1
sub $s2,$s1,$zero    #s2 = s1 - 0    = 1-0            =1
add $s3,$s1,$s1      #s3 = s1 + s1   = 1+1            =2
or $s4,$s5,$s1       #s4 = s5 | s1   = 101 | 001 = 101 =5
and $s5,$s6,$s1      #s5 = s6 ^ s1   = 101 ^ 001      =1
```

条指令，解释如何利用气泡流水线解决数据相关性的？插入的气泡数与教材上的是

否一致?

答:利用气泡流水线解决数据相关性的核心是通过硬件检测数据相关,当后续指令在 ID 段需要读取的寄存器值，其前面指令尚未写回（仍处于 EX 或 MEM 段）时，阻塞 IF 和 ID 段的执行，并在 EX 段插入气泡（空操作），直至前面指令完成寄存器写回，确保后续指令读取到正确值。以下针对 5 条指令逐一分析：

1）and $s1,$s1,$s2：该指令写回 $s1。

2）sub $s2,$s1,$zero：在 ID 段需读取 $s1，若此时 and 指令未完成写回（处于 EX 或 MEM 段），则阻塞 IF、ID 段，在 EX 段插入气泡。

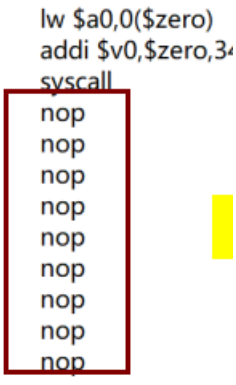
3）add $s3,$s1,$s1：在 ID 段需读取 $s1，若 and 指令未完成写回，同样阻塞相关段并在 EX 段插入气泡。

4）or $s4,$s5,$s1：在 ID 段需读取 $s1，若 and 指令未完成写回，检测到数据相关后，阻塞 IF、ID 段，在 EX 段插入气泡。

5）and $s5,$s6,$s1：在 ID 段需读取 $s1，若 and 指令未完成写回，依旧阻塞 IF、ID 段，在 EX 段插入气泡。

每处数据相关插入的气泡数与教材一致。以相邻指令数据相关（如 sub 与 and）为例，需先后插入两个气泡消除相关性。这是因为在流水线中，需等待前面指令经过 EX、MEM 段后，在 WB 段写回寄存器。插入气泡可避免错误取值，确保数据一致性

(8) 对于测试程序 fib_mips.asm，增加 nop 指令的目的是什么？ 尝试减少 nop 的数量，有何影响?



答：nop（无操作）指令用于填充流水线的延迟槽，确保前一条 syscall（数码管显示）指令完全执行完毕。由于 syscall 执行 34 号功能（数码管显示）需要一定时间完成硬件操作（如更新显示内容），若立即执行后续的 lw 指令，可能因前序操作未完成而导致数据读取错误或显示异常。nop 指令通过占用时钟周期，

为 syscall 操作争取足够时间，避免流水线冲突，保证程序逻辑的正确性。

若尝试减少 nop 的数量，可能会使 syscall 的数码管显示操作尚未完成时，后续 lw 指令就开始执行。这会导致以下影响：

i：显示错误：数码管可能未正确更新显示内容，出现乱码或错误数值。

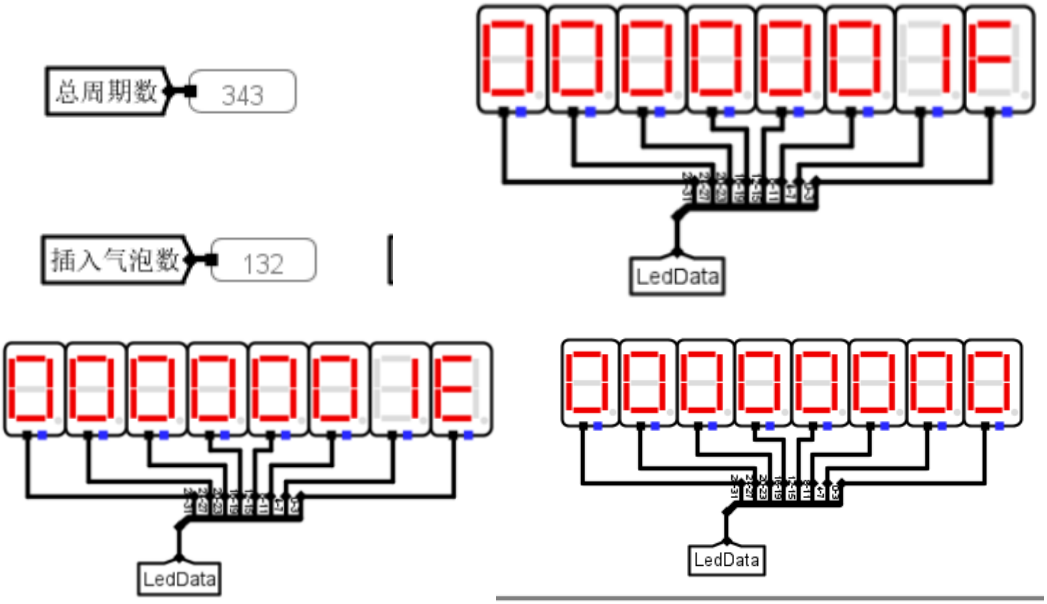
ii：数据冲突：流水线中前序操作未结束，后续指令读取数据时可能获取到无效值，破坏程序逻辑的正确性。

iii：系统不稳定：频繁的冲突可能导致程序运行异常，甚至崩溃。

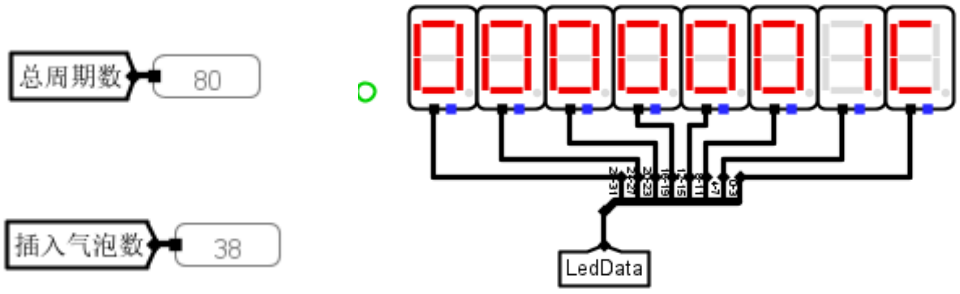
(9) 实验 PPT 第 28 页 要求（1）（2）

答：(1)在气泡流水线 MIPS 处理器的数据通路上运行 test2-1.hex、test2-2.hex、test2-3.hex、sum_mips.hex、fib_mips.hex、sort1_mips.hex、sort2_mips.hex 等程序，记录每个程序运行后的总周期数、插入气泡数。

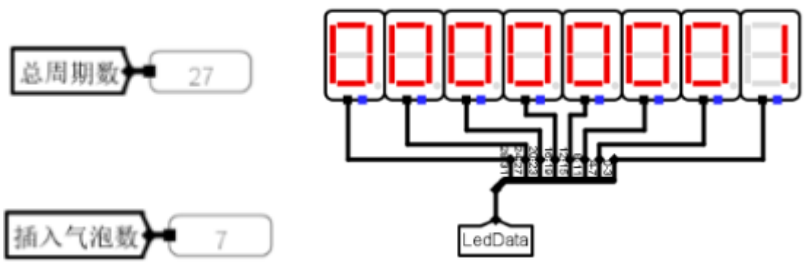
test2-1.hex：

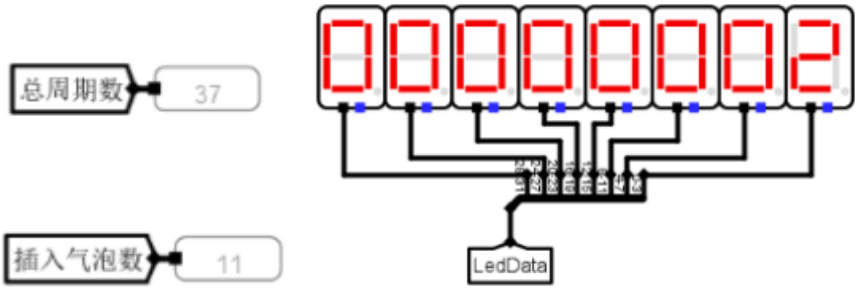


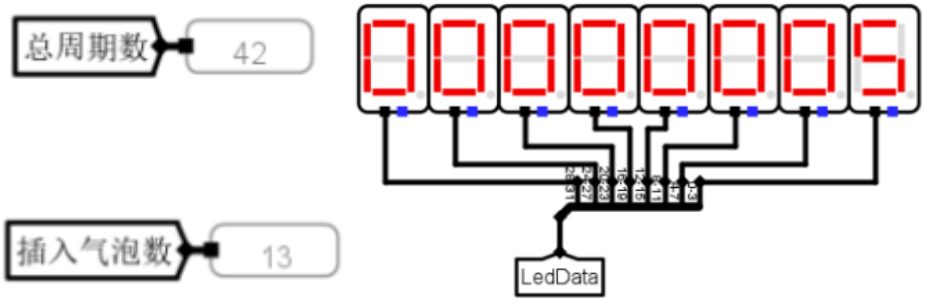
test2-2.hex：

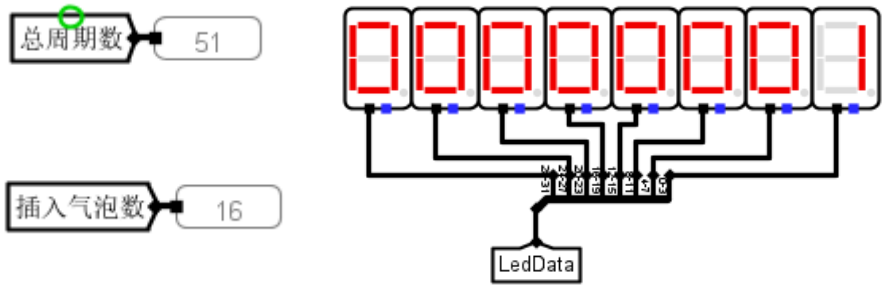


test2-3.hex:

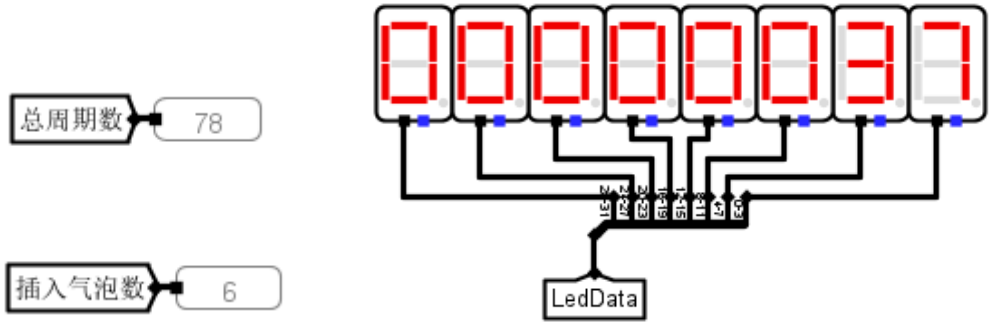




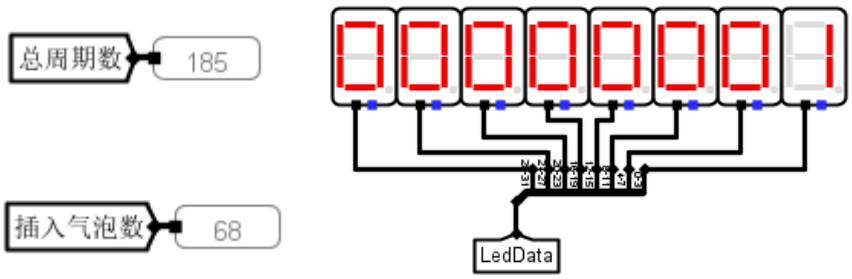


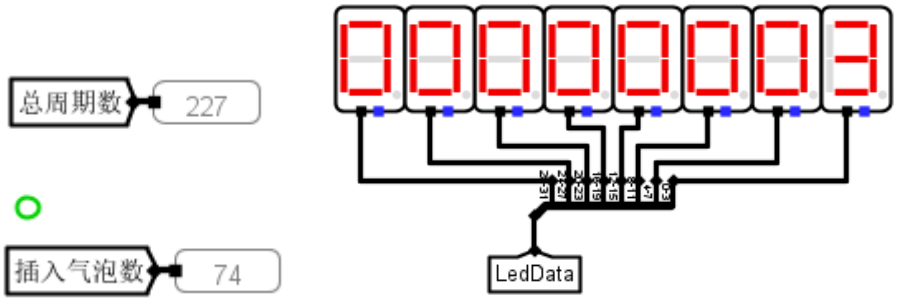


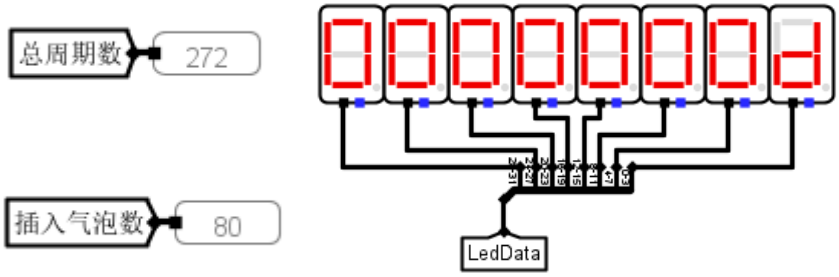
sum_mips.hex:

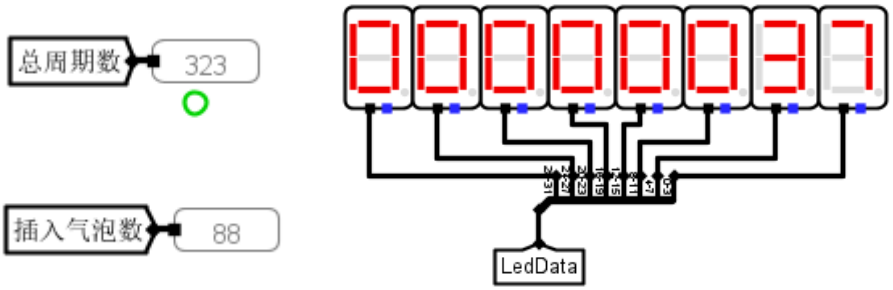


fib_mips.hex:

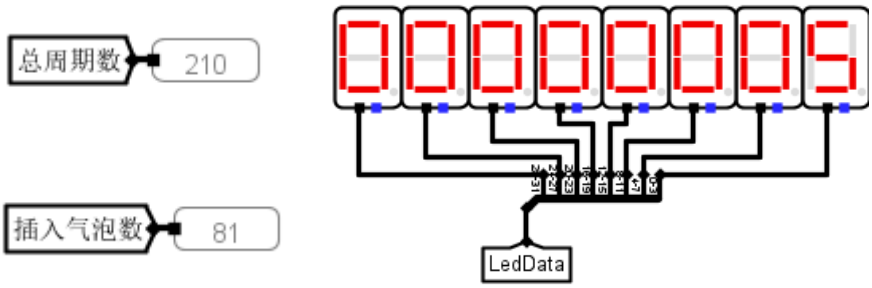


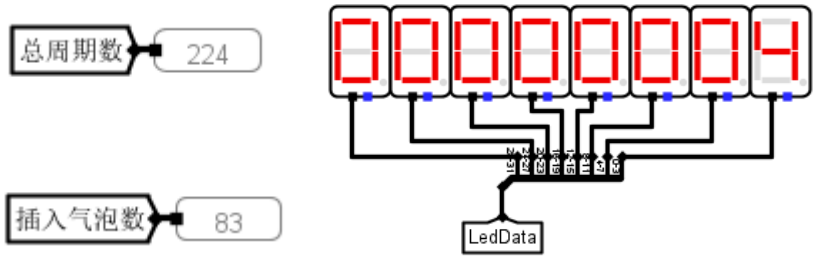


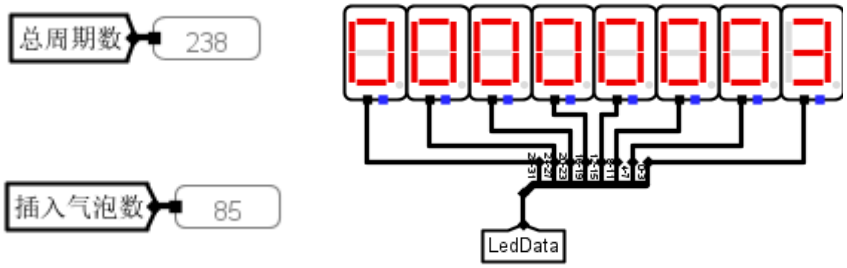


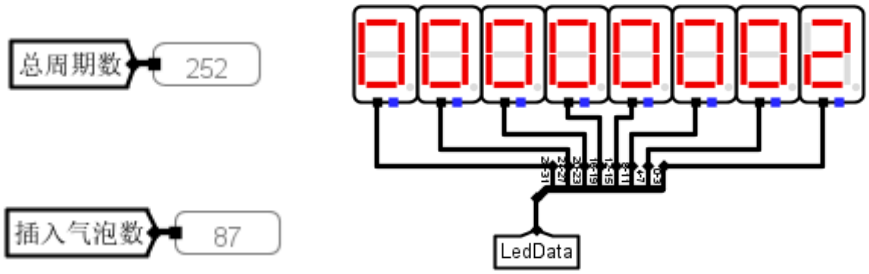


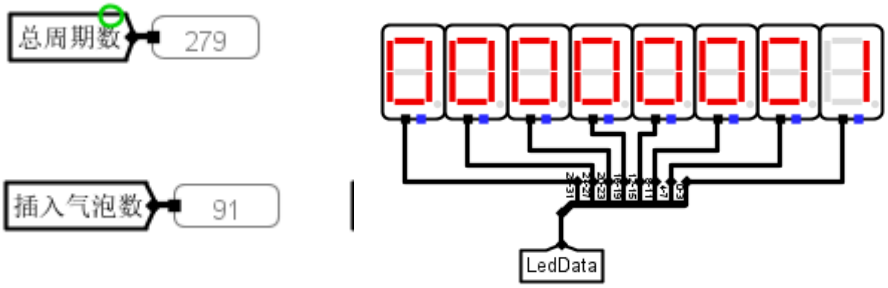
sort1_mips.hex:



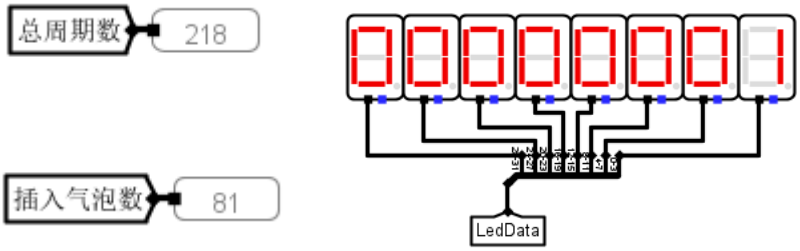


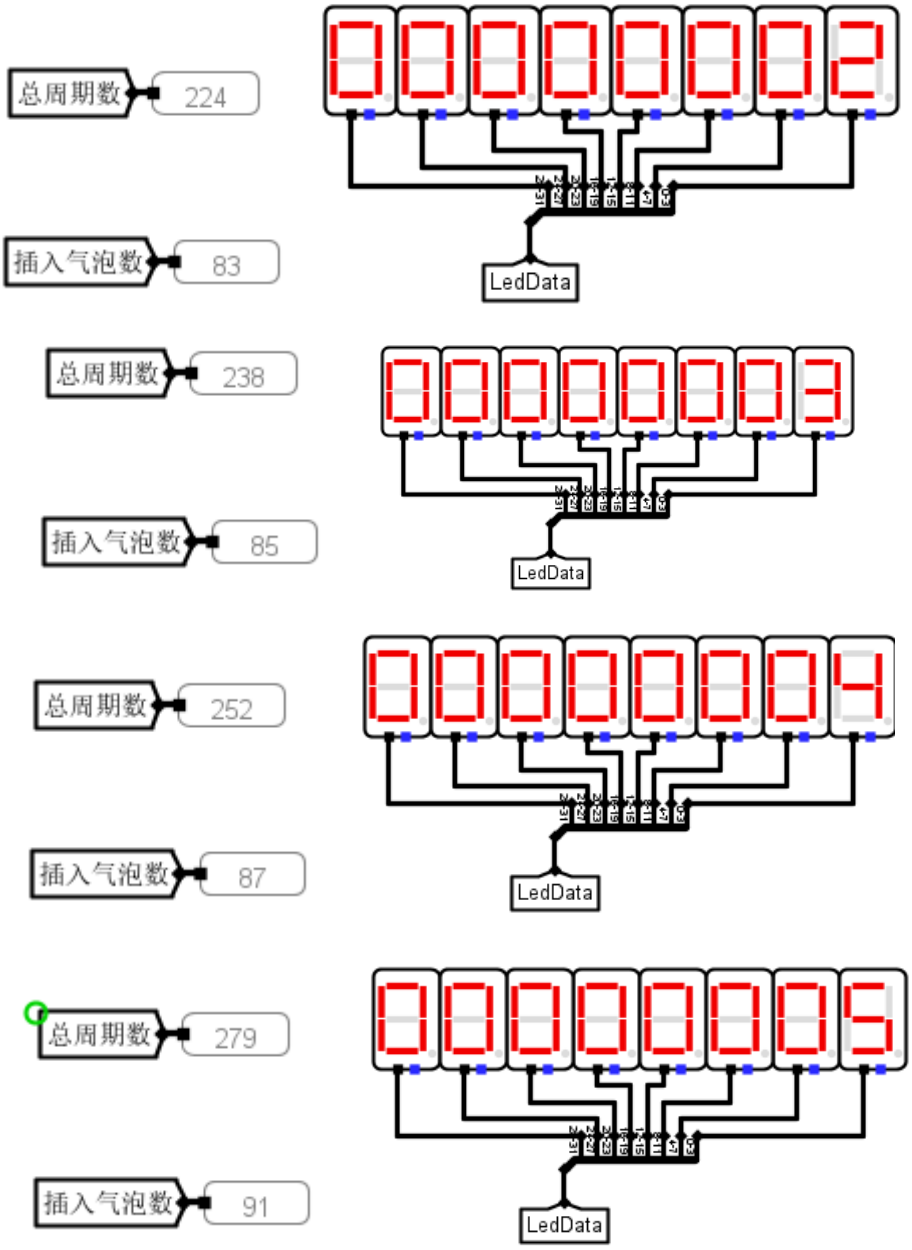






sort2_mips.hex:





（2）在气泡流水线 MIPS 处理器的数据通路上运行教材 P269 例 7.1 的程序（test2-4.hex），给出该程序运行后的时空图，并与教材上的图 7.20 进行比较，观测是否一致？

时空图如下，与教程上的一样

CLKS	取 指 令 IF	译码 ID	执行 EX	访 存 MEM	写回 WB
1	lw $5,4($1)				

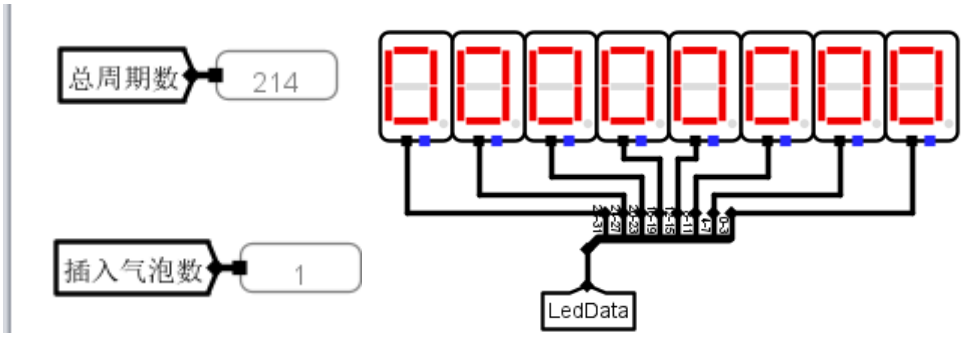
CLKS	取 指 令 IF	译码 ID	执行 EX	访 存 MEM	写回 WB
2	add $6,$5,$7	lw $5,4($1)			
3	sub $1,$2,$3	add $6,$5,$7	lw $5,4($1)		
4	sub $1,$2,$3	add $6,$5,$7		lw $5,4($1)	
5	sub $1,$2,$3	add $6,$5,$7			lw $5,4($1)
6	or $7,$6,$7	sub $1,$2,$3	add $6,$5,$7		
7	and $9,$7,$6	or $7,$6,$7	sub $1,$2,$3	add $6,$5,$7	
8	and $9,$7,$6	or $7,$6,$7		sub $1,$2,$3	add $6,$5,$7
9	Next Instr	and $9,$7,$6	or $7,$6,$7		sub $1,$2,$3
10	Next Instr	and $9,$7,$6		or $7,$6,$7	
11	Next	and			or

CLKS	取 指 令 IF	译码 ID	执行 EX	访 存 MEM	写回 WB
	Instr	$9,$7,$6			$7,$6,$7

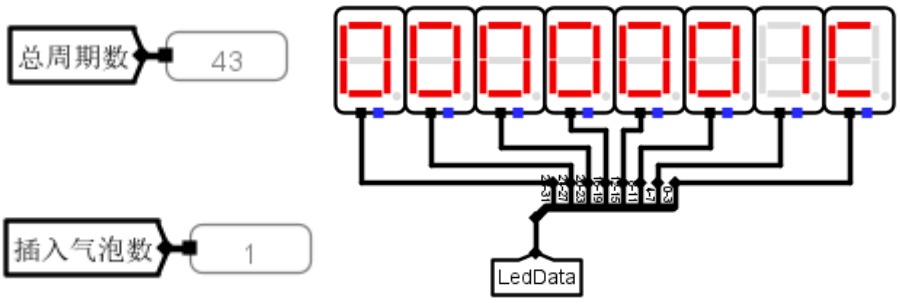
(10) 实验 PPT 第 42 页 要求（1）（2）

1） 在重定向流水线 MIPS 处理器的数据通路上运行 test3-1.hex、test3-2.hex、test3-3.hex、sum_mips.hex、fib_mips.hex、sort1_mips.hex、sort2_mips.hex 等程序，记录每个程序运行后的总周期数、插入气泡数，并与气泡流水线对应程序的总周期数、插入气泡数进行比较，得出什么结论？

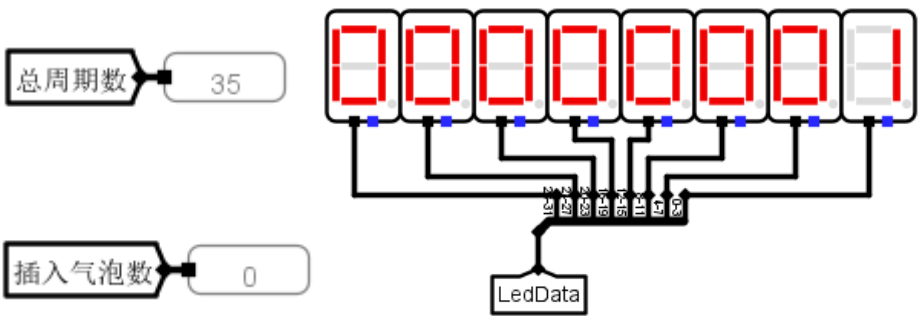
test3-1.hex：



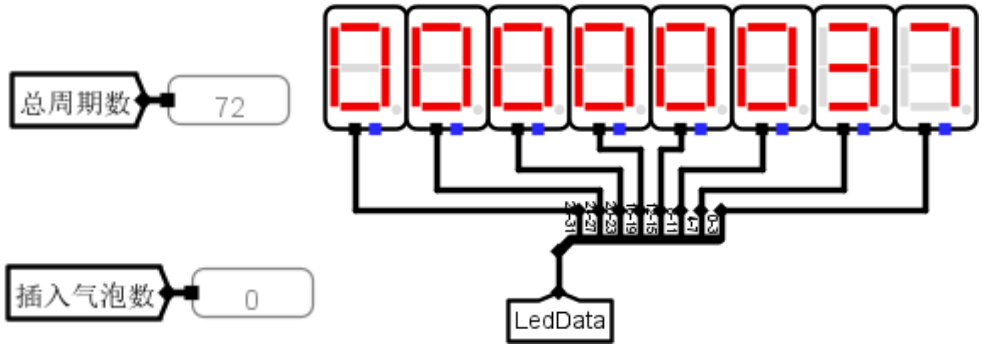
test3-2.hex：



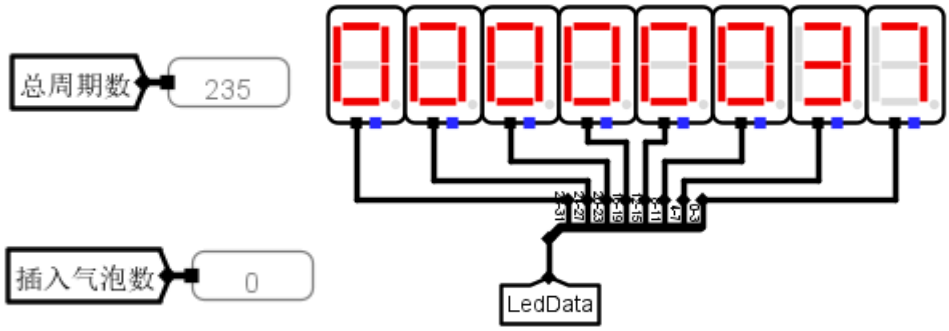
test3-3.hex：



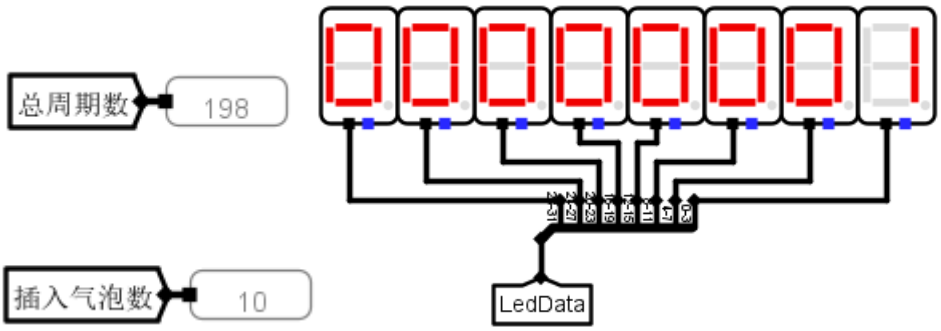
sum_mips.hex：



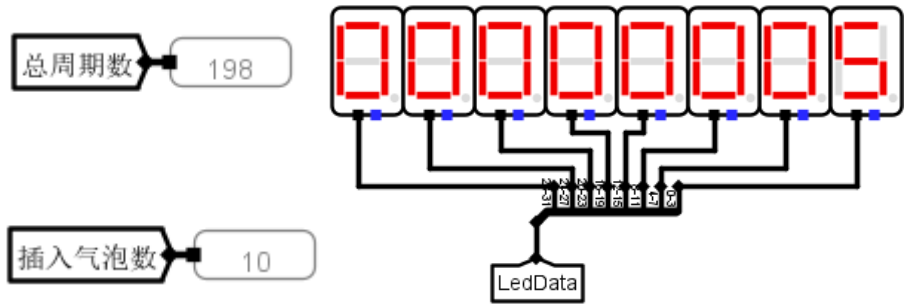
fib_mips.hex：



sort1_mips.hex：



sort2_mips.hex

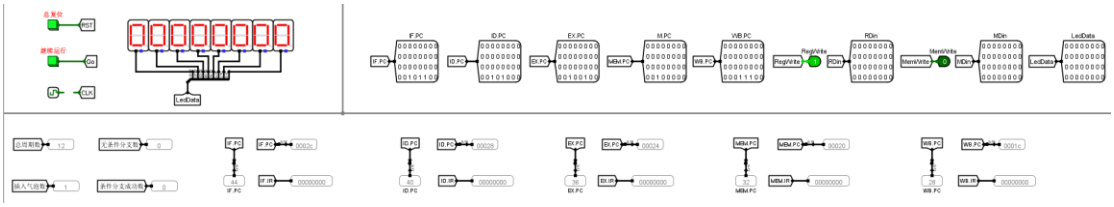


结论：重定向流水线的插入气泡和总周期数均小于气泡流水线，这是因为重定向方法无需插入气泡，可以解决大部分的数据相关问题，避免插入气泡引起的流水线性能下降，极大优化流水线性能

2） 在重定向流水线 MIPS 处理器的数据通路上运行教材 P275 例 7.2 的程序（test3-

4.hex)，给出该程序运行后的时空图，并与教材上的图 7.28 进行比较，观测是否一致

运行结果如下：



时空图如下：

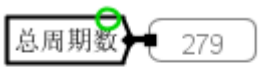
CLKs	取指令 IF	译码 ID	执行 EX	访存 MEM	写回 WB
1	lw $5,4($1)				
2	add $6,$5,$7	lw $5,4($1)			
3	sub $1,$2,$3	add $6,$5,$7	lw $5,4($1)		
4	sub $1,$2,$3	add $6,$5,$7		lw $5,4($1)	
5	or $7,$6,$7	sub $1,$2,$3	add $6,$5,$7		lw $5,4($1)
6	and $9,$7,$6	or $7,$6,$7	sub $1,$2,$3	add $6,$5,$7	
7	xor $5,$9,$3	and $9,$7,$6	or $7,$6,$7	sub $1,$2,$3	add $6,$5,$7
8	Next Instr	xor $5,$9,$3	and $9,$7,$6	or $7,$6,$7	sub $1,$2,$3

与课本一致

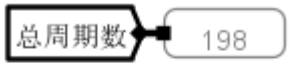
(11) 任选一个测试程序，基于气泡流水线和重定向流水线，结合其总周期数和插入气泡数以及电路，分析原因

答：选择降序排序程序进行分析。二者的总周期数和插入气泡数情况如下

气泡流水线：



重定向流水线：

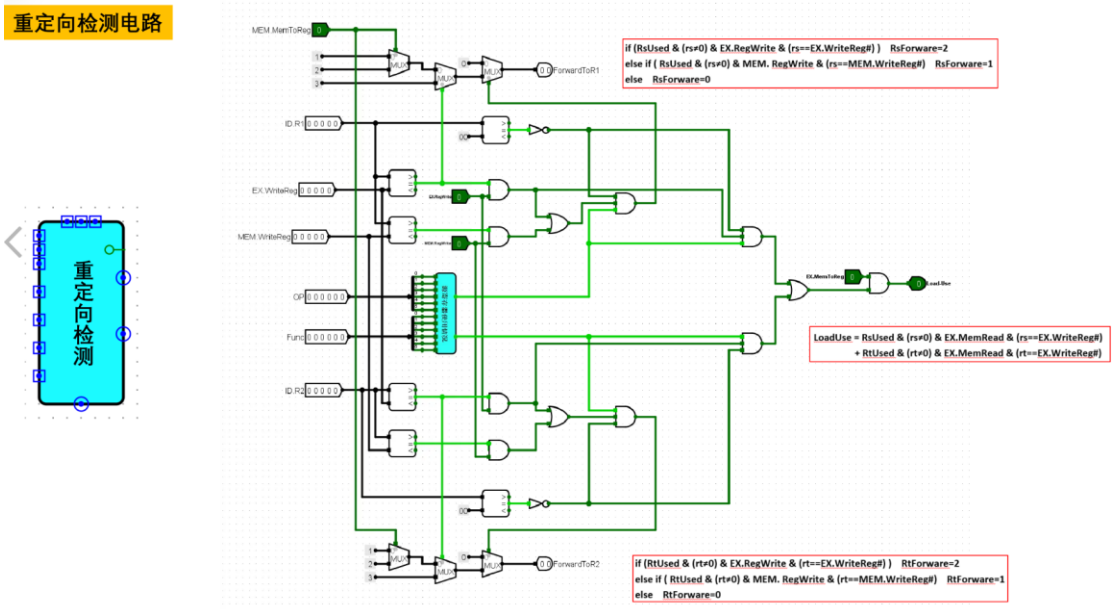


可以发现

①气泡流水线的总周期数通常较多。在存在数据相关或分支相关时，需要插入大量气泡（空操作指令）来解决冲突，使流水线停顿等待。例如在冒泡排序程序中，数据相关和分支指令频繁出现，导致在很多时钟周期都要插入气泡，增加了总周期数。插入气泡数也较多。只要检测到数据相关或分支相关，就会插入气泡。

原因是气泡流水线没有复杂的硬件检测和预测机制，只能通过插入气泡来保证指令执行的正确性。它不考虑指令之间的相关性具体情况，只要遇到相关问题就插入气泡，所以会导致总周期数增多，插入气泡数也较多，性能相对较低。

②重定向流水线的总周期数相对气泡流水线可能会减少。通过硬件逻辑检测数据相关和分支相关，一旦检测到相关，会暂停流水线并调整指令执行顺序。相比于气泡流水线盲目插入气泡，重定向流水线可以更精准地处理相关问题，在某些情况下可以减少不必要的停顿，从而减少总周期数。其插入气泡数较少。它不是像气泡流水线那样在很多情况下都插入气泡，而是通过调整指令执行顺序来解决相关问题，只有在必要时才插入少量气泡。原因是重定向流水线有专门的硬件检测机制，重定向检测电路



能够分析指令之间的相关性，然后根据具体情况进行处理。它可以在检测到相关时暂停流水

线，调整指令执行顺序，使得在一些情况下不需要插入气泡或者只插入很少的气泡，所以在性能上会比气泡流水线有所提升。

(12) 动态分支预测流水线，结合讲解视频，深度理解分支预测原理以及 BHT 是如何预测的，又是如何更新的?

答：分支预测原理：在流水线处理器中，遇到分支指令（如条件跳转指令）时，处理器需确定后续执行路径。若等分支指令执行完再确定下一条指令，会使流水线停顿，降低性能。分支预测技术就是在分支指令结果确定前，提前猜测其执行方向，让流水线持续运行，减少停顿时间。

BHT 预测过程：①记录历史信息：BHT 是动态分支预测核心部件，以表格形式存在，每条分支指令在其中有对应表项。表项记录该分支指令之前跳转情况，如最近几次跳转结果是“跳”或“不跳”。例如，对于频繁跳转的分支指令，其 BHT 表项会记录多次跳转信息。

②分析历史信息预测：处理器依据 BHT 中记录的历史跳转信息预测当前分支指令执行方向。若某分支指令过去多次跳转，预测当前也跳转；若过去多不跳转，预测当前也不跳转 。比如，一个循环中的分支指令，若之前每次循环都跳转，预测本次循环也跳转。

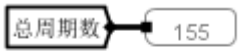
BHT 更新机制：①预测正确时：若分支指令实际执行结果与预测一致，BHT 会强化对应预测方向。例如采用两比特饱和计数器的 BHT，当前状态为“弱需要跳转”且实际跳转，状态更新为“强需要跳转”。这表明对该分支指令未来跳转的预测信心增强。

②预测错误时：若分支指令实际执行结果与预测不符，BHT 会调整预测方向。比如当前状态为“强需要跳转”但实际未跳转，状态变为“弱需要跳转”。通过这种调整，使 BHT 能根据分支指令实际执行情况不断修正预测，提高后续预测准确性。

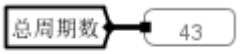
(13) 实验课件 P57 要求（1）

1） 在动态分支预测流水线 MIPS 处理器的数据通路上运行 test4-1.hex、test4-2.hex、test4-3.hex 、 test4-4.hex 、 sum_mips.hex 、 fib_mips.hex 、 sort1_mips.hex 、sort2_mips.hex 等程序，记录每个程序运行后的总周期数、插入气泡数，并与气泡流水线、重定向流水线对应程序的总周期数、插入气泡数进行比较，得出什么结论

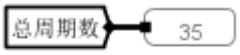
test4-1.hex：



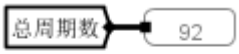
test4-2.hex



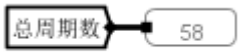
test4-3.hex：



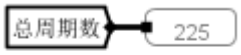
test4-4.hex：



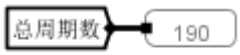
sum_mips.hex：



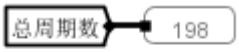
fib_mips.hex：



sort1_mips.hex：



sort2_mips.hex：



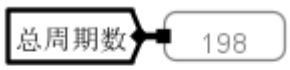
结论：相比于气泡流水线和重定向流水线，在增加了动态分支预测机制后，基本不需要插入气泡，只有少数 Load-Use 相关需要插入气泡，时钟周期也减少了，执行效率有较大提升。

(14) 针对降序排序程序，观察其在气泡流水线，重定向流水线以及动态分支预测流水线的性能表现（总周期数和插入气泡数等）有何不同，并给出原因（可绘制三种情况下的流水线时空图进行解释）

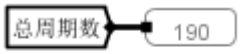
答：降序排序在三种流水线下的性能表现如下

气泡流水线：

重定向流水线：



动态分支预测流水线：



可以发现动态分支预测流水线的总周期数和插入气泡数是最少的，而气泡流水线是最多的。

气泡流水线的时空图如下：

clks	取指令 IF	译码 ID	执行 EX	访存 MEM	写回 WB
1	lw $s3,0($s0)				
2	lw $s4,0($s1)	lw $s3,0($s0)			
3	slt $t0,$s3,$s4	lw $s4,0($s1)	lw $s3,0($s0)		
4	beq $t0,$zero,sort_next	slt $t0,$s3,$s4	lw $s4,0($s1)	lw $s3,0($s0)	
5	气泡	beq $t0,$zero,sort_next	slt $t0,$s3,$s4	lw $s4,0($s1)	lw $s3,0($s0)
6	气泡	气泡	beq $t0,$zero,sort_next	slt $t0,$s3,$s4	lw $s4,0($s1)
7	sw $s3,0($s1)	气泡	气泡	beq $t0,$zero,sort_next	slt $t0,$s3,$s4
8	sw $s4,0($s0)	sw $s3,0($s1)	气泡	气泡	beq $t0,$zero,sort_next

9	addi $s1,$s1,-4	sw $s4,0($s0)	sw $s3,0($s1)	气泡	气泡
10	bne $s0,$s1,sort_loop	addi $s1,$s1,-4	sw $s4,0($s0)	sw $s3,0($s1)	气泡
11	气泡	bne $s0,$s1,sort_loop	addi $s1,$s1,-4	sw $s4,0($s0)	sw $s3,0($s1)
12	气泡	气泡	bne $s0,$s1,sort_loop	addi $s1,$s1,-4	sw $s4,0($s0)
13	addi $s0,$s0,4	气泡	气泡	bne $s0,$s1,sort_loop	addi $s1,$s1,-4
14	addi $s1,$zero,16	addi $s0,$s0,4	气泡	气泡	bne $s0,$s1,sort_loop
15	bne $s0,$s1,sort_loop	addi $s1,$zero,16	addi $s0,$s0,4	气泡	气泡
16	气泡	bne $s0,$s1,sort_loop	addi $s1,$zero,16	addi $s0,$s0,4	气泡
17	气泡	气泡	bne $s0,$s1,sort_loop	addi $s1,$zero,16	addi $s0,$s0,4
18	lw $a0,0($zero)	气泡	气泡	bne $s0,$s1,sort_loop	addi $s1,$zero,16
19	addi $v0,$zero,34	lw $a0,0($zero)	气泡	气泡	bne $s0,$s1,sort_loop

20	syscall	addi $v0,$zero,34	lw $a0,0($zero)	气泡	气泡
21	气泡	syscall	addi $v0,$zero,34	lw $a0,0($zero)	气泡
22	气泡	气泡	syscall	addi $v0,$zero,34	lw $a0,0($zero)
23	lw $a0,4($zero)	气泡	气泡	syscall	addi $v0,$zero,34
24	addi $v0,$zero,34	lw $a0,4($zero)	气泡	气泡	syscall
25	syscall	addi $v0,$zero,34	lw $a0,4($zero)	气泡	气泡
26	气泡	syscall	addi $v0,$zero,34	lw $a0,4($zero)	气泡
27	气泡	气泡	syscall	addi $v0,$zero,34	lw $a0,4($zero)
28	lw $a0,8($zero)	气泡	气泡	syscall	addi $v0,$zero,34
29	addi $v0,$zero,34	lw $a0,8($zero)	气泡	气泡	syscall
30	syscall	addi $v0,$zero,34	lw $a0,8($zero)	气泡	气泡
31	气泡	syscall	addi $v0,$zero,34	lw $a0,8($zero)	气泡
32	气泡	气泡	syscall	addi $v0,$zero,34	lw $a0,8($zero)
33	lw $a0,12($zero)	气泡	气泡	syscall	addi $v0,$zero,34
34	addi $v0,$zero,34	lw $a0,12($zero)	气泡	气泡	syscall
35	syscall	addi $v0,$zero,34	lw $a0,12($zero)	气泡	气泡

36	气泡	syscall	addi $v0,$zero,34	lw $a0,12($zero)	气泡
37	气泡	气泡	syscall	addi $v0,$zero,34	lw $a0,12($zero)
38	lw $a0,16($zero)	气泡	气泡	syscall	addi $v0,$zero,34
39	addi $v0,$zero,34	lw $a0,16($zero)	气泡	气泡	syscall
40	syscall	addi $v0,$zero,34	lw $a0,16($zero)	气泡	气泡
41	气泡	syscall	addi $v0,$zero,34	lw $a0,16($zero)	气泡
42	气泡	气泡	syscall	addi $v0,$zero,34	lw $a0,16($zero)
43	addi $v0,$zero,10	气泡	气泡	syscall	addi $v0,$zero,34
44	syscall	addi $v0,$zero,10	气泡	气泡	syscall

重定向流水线的时空图如下：

CLKs	取指令 IF	译码 ID	执行 EX	访存 MEM	写回 WB
1	lw $s3,0($s0)				
2	lw $s4,0($s1)	lw $s3,0($s0)			
3	slt $t0,$s3,$s4	lw $s4,0($s1)	lw $s3,0($s0)		
4	beq $t0,$zero,sort_next	slt $t0,$s3,$s4	lw $s4,0($s1)	lw $s3,0($s0)	
5	（暂停，检测相关）				（若条件满足，重定向执行）
6	sw $s3,0($s1)	beq $t0,$zero,sort_next	slt $t0,$s3,$s4	lw $s4,0($s1)	lw $s3,0($s0)
7	sw $s4,0($s0)	sw $s3,0($s1)	beq	slt $t0,$s3,$s4	lw $s4,0($s1)

			$t0,$zero,sort_next		
8	addi $s1,$s1,-4	sw $s4,0($s0)	sw $s3,0($s1)	beq $t0,$zero,sort_next	slt $t0,$s3,$s4
9	bne $s0,$s1,sort_loop	addi $s1,$s1,-4	sw $s4,0($s0)	sw $s3,0($s1)	beq $t0,$zero,sort_next
10	（暂停，检测相关）				（若条件满足，重定向执行）
11	addi $s0,$s0,4	bne $s0,$s1,sort_loop	addi $s1,$s1,-4	sw $s4,0($s0)	sw $s3,0($s1)
12	addi $s1,$zero,16	addi $s0,$s0,4	bne $s0,$s1,sort_loop	addi $s1,$s1,-4	sw $s4,0($s0)
13	bne $s0,$s1,sort_loop	addi $s1,$zero,16	addi $s0,$s0,4	bne $s0,$s1,sort_loop	addi $s1,$s1,-4
14	（暂停，检测相关）				（若条件满足，重定向执行）
15	lw $a0,0($zero)	bne $s0,$s1,sort_loop	addi $s1,$zero,16	addi $s0,$s0,4	bne $s0,$s1,sort_loop
16	addi $v0,$zero,34	lw $a0,0($zero)	bne $s0,$s1,sort_loop	addi $s1,$zero,16	addi $s0,$s0,4
17	syscall	addi $v0,$zero,34	lw $a0,0($zero)	bne $s0,$s1,sort_loop	addi $s1,$zero,16
18	（暂停，检测相关）				（若条件满足，重定向执行）
19	lw $a0,4($zero)	syscall	addi $v0,$zero,34	lw $a0,0($zero)	bne $s0,$s1,sort_loop
20	addi $v0,$zero,34	lw $a0,4($zero)	syscall	addi $v0,$zero,34	lw $a0,0($zero)
21	syscall	addi $v0,$zero,34	lw $a0,4($zero)	syscall	addi $v0,$zero,34
22	（暂停，检测相关）				（若条件满足，重定向执行）

23	lw $a0,8($zero)	syscall	addi $v0,$zero,34	lw $a0,4($zero)	syscall
24	addi $v0,$zero,34	lw $a0,8($zero)	syscall	addi $v0,$zero,34	lw $a0,4($zero)
25	syscall	addi $v0,$zero,34	lw $a0,8($zero)	syscall	addi $v0,$zero,34
26					（若条件满足，重定向执行）
27	lw $a0,12($zero)	syscall	addi $v0,$zero,34	lw $a0,8($zero)	syscall
28	addi $v0,$zero,34	lw $a0,12($zero)	syscall	addi $v0,$zero,34	lw $a0,8($zero)
29	syscall	addi $v0,$zero,34	lw $a0,12($zero)	syscall	addi $v0,$zero,34
30	（暂停，检测相关）				（若条件满足，重定向执行）
31	lw $a0,16($zero)	syscall	addi $v0,$zero,34	lw $a0,12($zero)	syscall
32	addi $v0,$zero,34	lw $a0,16($zero)	syscall	addi $v0,$zero,34	lw $a0,12($zero)
33	syscall	addi $v0,$zero,34	lw $a0,16($zero)	syscall	addi $v0,$zero,34
34	（暂停，检测相关）				（若条件满足，重定向执行）
35	addi $v0,$zero,10	syscall	addi $v0,$zero,34	lw $a0,16($zero)	syscall
36	syscall				（若条件满足，重定向执行）

动态分支流水线的时空图如下：

clks	取指令 IF	译码 ID	执行 EX	访存 MEM	写回 WB
1	lw $s3,0($s0)				
2	lw $s4,0($s1)	lw $s3,0($s0)			
3	slt $t0,$s3,$s4	lw $s4,0($s1)	lw $s3,0($s0)	预测：不跳转（假设）	

4	beq $t0,$zero,sort_next	slt $t0,$s3,$s4	lw $s4,0($s1)	lw $s3,0($s0)	
5	addi $s1,$s1,-4	beq $t0,$zero,sort_next	slt $t0,$s3,$s4	lw $s4,0($s1)	lw $s3,0($s0)
6	bne $s0,$s1,sort_loop	addi $s1,$s1,-4	beq $t0,$zero,sort_next	slt $t0,$s3,$s4	lw $s4,0($s1)
7	lw $a0,0($zero)	bne $s0,$s1,sort_loop	addi $s1,$s1,-4	beq $t0,$zero,sort_next	slt $t0,$s3,$s4
8	addi $v0,$zero,34	lw $a0,0($zero)	bne $s0,$s1,sort_loop	addi $s1,$s1,-4	beq $t0,$zero,sort_next
9		addi $v0,$zero,34	lw $a0,0($zero)	bne $s0,$s1,sort_loop	addi $s1,$s1,-4
10	预测错误，回滚				
11	addi $s0,$s0,4				
12	addi $s1,$zero,16	addi $s0,$s0,4			
13	bne $s0,$s1,sort_loop	addi $s1,$zero,16	addi $s0,$s0,4		
14	lw $s3,0($s0)	bne $s0,$s1,sort_loop	addi $s1,$zero,16	addi $s0,$s0,4	
15	lw $s4,0($s1)	lw $s3,0($s0)	bne	addi $s1,$zero,16	addi $s0,$s0,4

			$s0,$s1,sort_loop		
16	slt $t0,$s3,$s4	lw $s4,0($s1)	lw $s3,0($s0)	bne $s0,$s1,sort_loop	addi $s1,$zero,16
17	beq $t0,$zero,sort_next	slt $t0,$s3,$s4	lw $s4,0($s1)	lw $s3,0($s0)	bne $s0,$s1,sort_loop
18	addi $s1,$s1,-4	beq $t0,$zero,sort_next	slt $t0,$s3,$s4	lw $s4,0($s1)	lw $s3,0($s0)
19	bne $s0,$s1,sort_loop	addi $s1,$s1,-4	beq $t0,$zero,sort_next	slt $t0,$s3,$s4	lw $s4,0($s1)
20	lw $a0,0($zero)	bne $s0,$s1,sort_loop	addi $s1,$s1,-4	beq $t0,$zero,sort_next	slt $t0,$s3,$s4
21	addi $v0,$zero,34	lw $a0,0($zero)	bne $s0,$s1,sort_loop	addi $s1,$s1,-4	beq $t0,$zero,sort_next
22	预测错误，回滚				
23	addi $s0,$s0,4			重新取正确指令	
24	addi $s1,$zero,16	addi $s0,$s0,4		预测：不跳转（假设）	
25	bne $s0,$s1,sort_loop	addi $s1,$zero,16	addi $s0,$s0,4		
26	lw $a0,4($zero)	bne $s0,$s1,sort_loop	addi $s1,$zero,16	addi $s0,$s0,4	

27	addi $v0,$zero,34	lw $a0,4($zero)	bne $s0,$s1,sort_loop	addi $s1,$zero,16	addi $s0,$s0,4
28	syscall	addi $v0,$zero,34	lw $a0,4($zero)	bne $s0,$s1,sort_loop	addi $s1,$zero,16
29		syscall	addi $v0,$zero,34	lw $a0,4($zero)	bne $s0,$s1,sort_loop
30			syscall	addi $v0,$zero,34	lw $a0,4($zero)
31				syscall	addi $v0,$zero,34
32					syscall
33	lw $a0,8($zero)				
34	addi $v0,$zero,34	lw $a0,8($zero)			
35	syscall	addi $v0,$zero,34	lw $a0,8($zero)		
36		syscall	addi $v0,$zero,34	lw $a0,8($zero)	
37			syscall	addi $v0,$zero,34	lw $a0,8($zero)
38				syscall	addi $v0,$zero,34
39					syscall
40	lw $a0,12($zero)				

41	addi $v0,$zero,34	lw $a0,12($zero)			
42	syscall	addi $v0,$zero,34	lw $a0,12($zero)		
43		syscall	addi $v0,$zero,34	lw $a0,12($zero)	
44			syscall	addi $v0,$zero,34	lw $a0,12($zero)
45				syscall	addi $v0,$zero,34
46					syscall
47	lw $a0,16($zero)				
48	addi $v0,$zero,34	lw $a0,16($zero)			
49	syscall	addi $v0,$zero,34	lw $a0,16($zero)		
50		syscall	addi $v0,$zero,34	lw $a0,16($zero)	
51			syscall	addi $v0,$zero,34	lw $a0,16($zero)
52				syscall	addi $v0,$zero,34
53					syscall
54	addi $v0,$zero,10				
55	syscall	addi $v0,$zero,10			

由以上三个时空图可以推断，三种流水线中·动态分支预测流水线在预测准确时性能最优，重定向流水线次之，气泡流水线性能相对较差。但动态分支预测流水线如果预测错误会带来较大开销，而重定向流水线和气泡流水线虽然性能相对低一些，但实现相对简

单。

4. 实验报告提交

(1) 实验报告命名为：学号+姓名+第 7 次实验报告.pdf。

(2) 将实验报告上传到 course.xmu.edu.cn 上，第 7 次实验报告提交截止时间（2 周内）：2025 年 6 月 4 日晚上 24 点。