



**《计算机组成原理》**

**课程实验报告**

姓名：陈澄

学院：信息学院

系：软件工程

专业：软件工程

学号：32420212202930

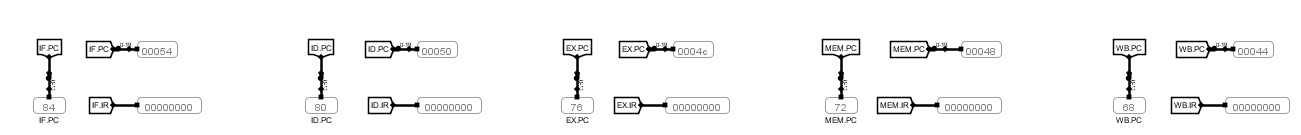
2023年6月1日

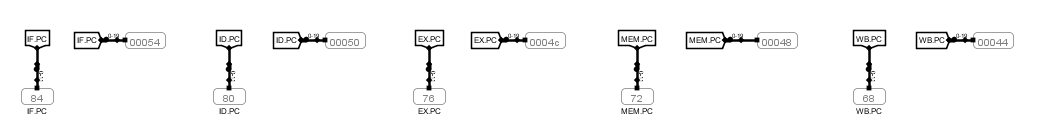
**第6次实验 流水线CPU设计**

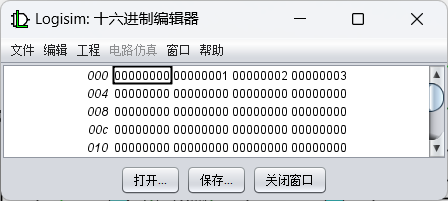
1. **实验目的**
2. **掌握理想流水线MIPS处理器设计的基本原理，在该流水线上运行测试程序。**
3. **掌握气泡流水线MIPS处理器设计的基本原理，在该流水线上运行测试程序。**
4. **掌握重定向流水线MIPS处理器设计的基本原理，在该流水线上运行测试程序。**
5. **掌握动态分支预测流水线MIPS处理器设计的基本原理，在该流水线上运行测试程序。**
6. **实验环境**
7. **Windows系统下运行Logisim软件（需安装JDK）。**
8. **MARS 4.5汇编工具。**
9. **实验内容**
10. **在理想流水线MIPS处理器上运行测试程序。**

**1)在理想流水线MIPS处理器的数据通路上运行测试程序test.hex。**

**运行结果：**







**2)分析理想流水线MIPS处理器的数据通路电路。**

理想流水线 MIPS 处理器的数据通路电路可以分为五个主要部分：取指、译码、执行、访存和写回。具体各部分的作用和电路组成如下：

取指：负责从指令存储器中取出指令，并将其送入下一个阶段。电路组成包括指令存储器和程序计数器（PC）。程序计数器用于存储下一条指令的地址，指令存储器根据程序计数器中的地址读取指令并进行输出。取指阶段的输出是下一条指令的二进制编码。

译码：负责对指令进行解码，确定指令的操作类型、源寄存器、目的寄存器等。电路组成包括寄存器文件（Regfile）和译码器。译码器将指令的操作码解码为控制信号，控制信号用于后续阶段的运算器、数据选择器等控制。寄存器文件用于存储和读取指令中的寄存器编号以及对应的数据。

执行：负责执行指令的运算操作。电路组成包括算术逻辑单元（ALU）和数据选择器。通过获取来自译码阶段的控制信号，数据选择器可以从寄存器文件或者立即数中选择数据作为 ALU 的操作数。ALU 负责对所选取的数据进行运算，根据操作码完成加减乘除以及各种逻辑运算等计算操作。

访存：负责执行访问内存的操作，包括读取和写入内存。电路组成包括数据存储器（Data Memory）和地址生成器。地址生成器根据寄存器文件的输出和控制信号，产生要访问的内存地址。数据存储器根据地址和控制信号，从内存中读取或者写入数据。

写回：负责将操作结果写入寄存器文件。电路组成包括写回选择器和寄存器文件。写回选择器根据译码阶段的控制信号，确定执行阶段的哪个阶段输出的数据写入寄存器文件。通过控制信号，寄存器文件将数据写入指定的目标寄存器中。

**3)分析理想流水线MIPS处理器的4个流水寄存器电路（IF/ID、ID/EX、EX/MEM、**

**MEM/WB）。**

IF/ID 流水寄存器：该寄存器用于存储取指阶段（IF）的输出，并将其传递给译码阶段（ID）。具体电路组成包括一个寄存器和两个锁存器。当指令从取指阶段发出后，被存入 IF/ID 寄存器的第一个锁存器；当指令到达译码阶段后，被读取并存入 IF/ID 寄存器的第二个锁存器。IF/ID 寄存器的主要作用是在译码阶段之前，保存取指阶段的输出，避免了指令执行时的数据冒险。

ID/EX 流水寄存器：该寄存器用于存储译码阶段（ID）的输出，并将其传递给执行阶段（EX）。具体电路组成包括一个寄存器和两个锁存器。当指令从译码阶段发出后，被存入 ID/EX 寄存器的第一个锁存器；当指令到达执行阶段后，被读取并存入 ID/EX 寄存器的第二个锁存器。ID/EX 寄存器的主要作用是在执行阶段之前，保存译码阶段的输出，避免数据冒险。

EX/MEM 流水寄存器：该寄存器用于存储执行阶段（EX）的输出，并将其传递给访存阶段（MEM）。具体电路组成包括一个寄存器和两个锁存器。当指令从执行阶段发出后，被存入 EX/MEM 寄存器的第一个锁存器；当指令到达访存阶段后，被读取并存入 EX/MEM 寄存器的第二个锁存器。EX/MEM 寄存器的主要作用是在访存阶段之前，保存执行阶段的输出，避免数据冒险。

MEM/WB 流水寄存器：该寄存器用于存储访存阶段（MEM）的输出，并将其传递给写回阶段（WB）。具体电路组成包括一个寄存器和两个锁存器。当指令从访存阶段发出后，被存入 MEM/WB 寄存器的第一个锁存器；当指令到达写回阶段后，被读取并存入 MEM/WB 寄存器的第二个锁存器。MEM/WB 寄存器的主要作用是在写回阶段之前，保存访存阶段的输出，避免数据冒险。

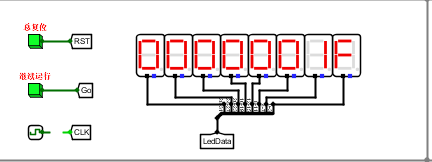
1. **在气泡流水线MIPS处理器上运行测试程序。**

**1)在气泡流水线MIPS处理器的数据通路上运行test1.hex、test2.hex、test3.hex、**

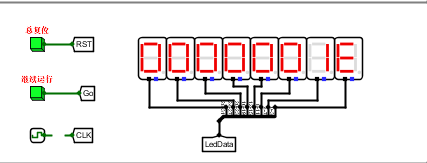
**fib\_mips.hex、sum\_mips.hex、sort1\_mips.hex、sort2\_mips.hex等程序，记录每个程序运行后的总周期数、插入气泡数。**

**test1.hex**

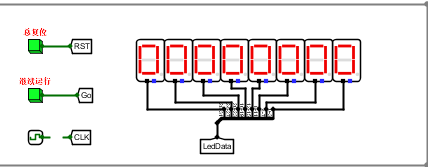
**运行后一段时间数码管显示1F（31）**



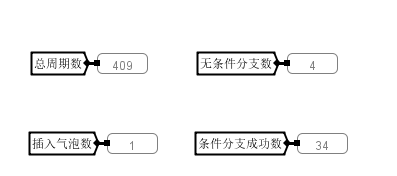
**继续运行逐渐减少**



**最后为0**

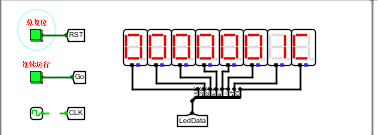


**总周期数，插入气泡数如下：**

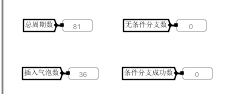


**test2.hex**

**运行结束后，数码管显示1C**

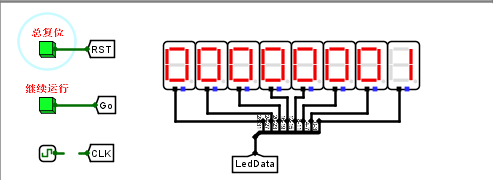


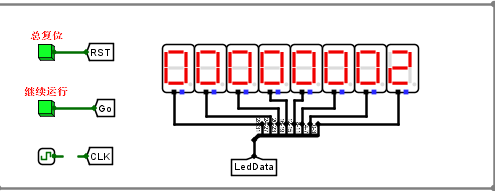
**总周期数，插入气泡数如下：**

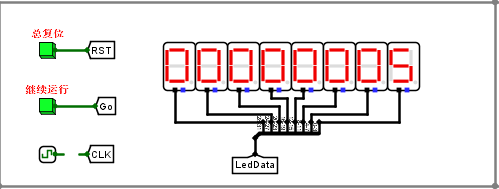


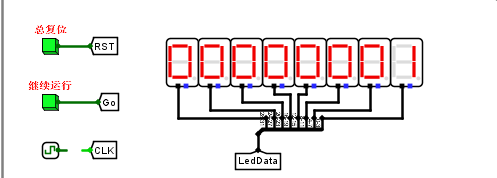
**test3.hex**

**数码管依次显示1,1,2,5,1**







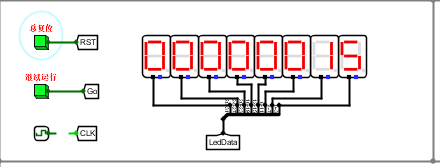


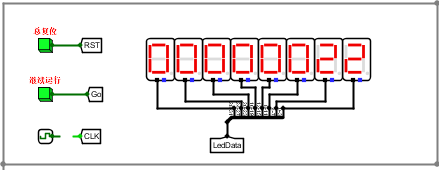
**总周期数，插入气泡数如下：**



**fib\_mips.hex**

**运行后依次显示0,1,1,2,3,5,8,D,15,22,37**



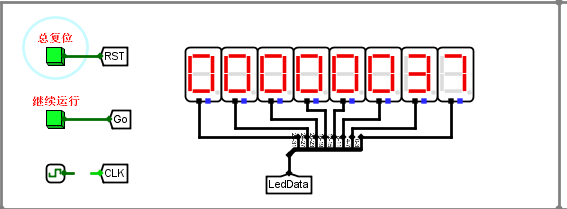


**总周期数和插入气泡数如下：**



**sum\_mips.hex**

**运行后数码管显示37**

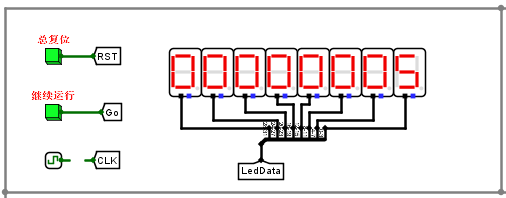


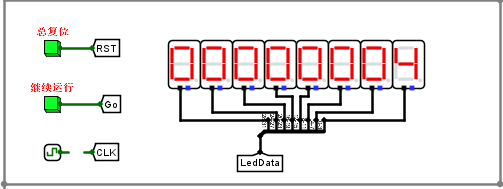
**总周期数和插入气泡数如下：**



**sort1\_mips.hex**

**运行后依次显示5,4,3,2,1**



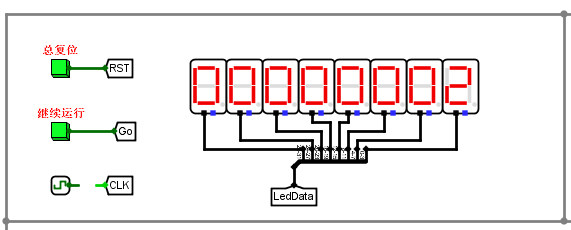


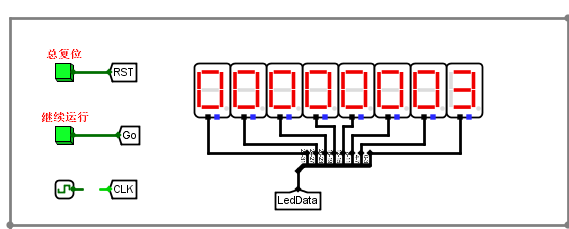
**总周期数和插入气泡数如下：**



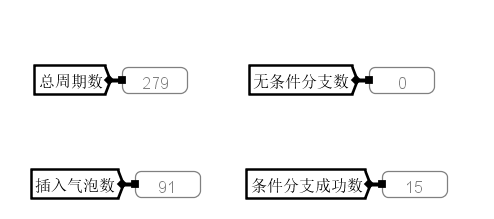
**sort2\_mips.hex**

**运行后依次显示1,2,3,4,5**



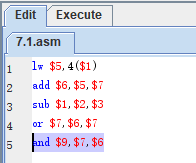


**总周期数和插入气泡数如下：**

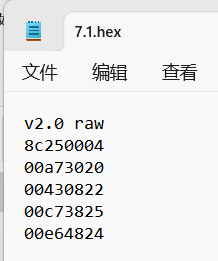


**2)在气泡流水线MIPS处理器的数据通路上运行教材P269例7.1的程序，给出该程序运行后的时空图，并与教材上的图7.20进行比较，观测是否一致？**

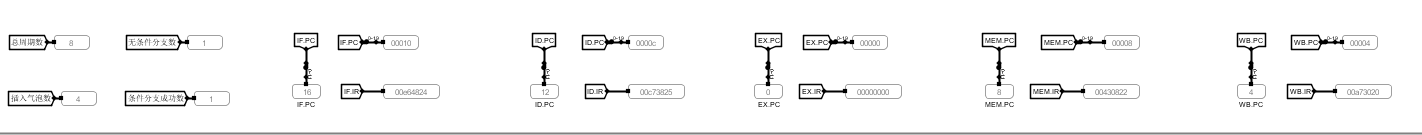
**程序如下：**



**所得hex文件：**



**（下图为第8个周期）**



**依据logisim电路所得时空图：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CLKs** | **IF** | **ID** | **EX** | **MEM** | **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** | **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** |  | **and $9,$7,$6** | **or $7,$6,$7** |  | **sub $1,$2,$3** |
| **10** |  | **and $9,$7,$6** |  | **or $7,$6,$7** |  |
| **11** |  | **and $9,$7,$6** |  |  | **or $7,$6,$7** |
| **12** |  |  | **and $9,$7,$6** |  |  |
| **13** |  |  |  | **and $9,$7,$6** |  |
| **14** |  |  |  |  | **and $9,$7,$6** |

**前11个周期与教材所给时空图完全相符：**



**3)分析气泡流水线MIPS处理器的数据通路电路。**

取指阶段：取指阶段主要由指令 Cache、指令译码器、程序计数器（PC）和 PC+4 加法器等组成。其中，指令 Cache 负责存储指令，指令译码器将指令转换为CPU内部的指令格式（例如，将操作码解析为对应的操作类型），PC 存储当前正在执行的指令地址。气泡流水线MIPS处理器的取指阶段与理想流水线MIPS处理器基本相同，但是会在指令相关性分析后产生气泡（nop）插入到流水线中。这样做是为了避免出现数据冒险或控制冒险等情况，提高指令执行的正确性和稳定性。

译码阶段：译码阶段负责将执行阶段需要的指令操作码和操作数从指令中提取出来，并发送到下一阶段用于执行指令。此阶段包含指令寄存器（IR）、通用寄存器、ALU 控制器等组件，其中指令寄存器存储上条指令，通用寄存器存储数据，ALU 控制器用于控制 ALU 进行特定操作。在气泡流水线MIPS处理器的译码阶段，如果前一条指令与当前指令存在相关性，则需要等待前一条指令全部完成，即插入一个气泡，保证指令的正确性，因此该阶段可能会产生气泡。

执行阶段：执行阶段是指令被实际执行的阶段。该阶段由 ALU、ALU 控制器、通用寄存器、加法器、移位器等组成。在这个阶段，ALU 控制器控制 ALU 执行指定操作，通用寄存器存储指令的操作数，加法器和移位器则用于地址计算。如果前面的指令与当前指令存在相关性，则会插入气泡，等待前面的指令执行完毕，避免数据冒险的发生。

访存阶段：访存阶段主要用于从内存中读取或写入数据。该阶段包含数据 Cache、存储器地址计算器、存储器数据寄存器以及存储器数据选择器等组件。存储器地址计算器计算存储器的实际地址，存储器数据寄存器存储要写入或已读取的数据，存储器数据选择器则控制从哪个寄存器中选择数据输出到总线上。开销较大的指令会在前面的阶段产生气泡，来等待前面的指令执行完毕，保证数据的正确性。

写回阶段：写回阶段将操作结果写回到通用寄存器中，同时也可能会更新程序计数器。该阶段包含通用寄存器、写回数据选择器和多路选择器等组件。写回数据选择器用于从 ALU 或存储器数据寄存器中选择正确的数据，多路选择器可以将多个数据路径连接到通用寄存器中。如果前面的指令与当前指令存在相关性，则会插入气泡，等待前面的指令执行完毕，避免数据冒险的发生。

**4)分析气泡流水线MIPS处理器的数据相关检测电路。**

数据相关检测电路主要使用比较器来比较前一条指令的目标寄存器和当前指令的源操作数的地址是否相等，以此判断是否存在数据相关性。

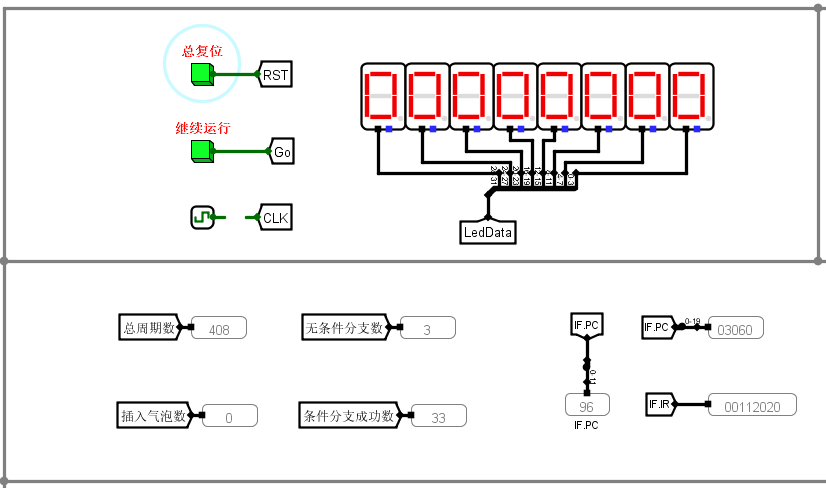
1. **在重定向流水线MIPS处理器上运行测试程序。**

**1)在重定向流水线MIPS处理器的数据通路上运行test1.hex、test2.hex、test3.hex、**

**fib\_mips.hex、sum\_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？**

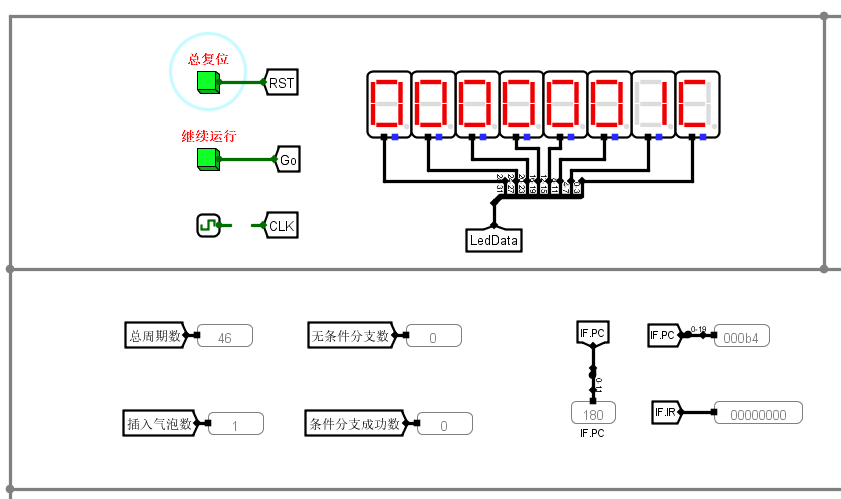
**test1.hex**

**对比气泡流水线对应程序总周期数少1，插入气泡数少1**



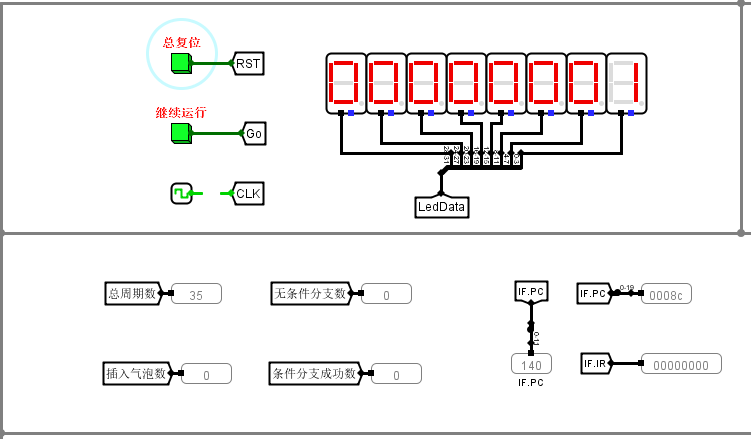
**test2.hex**

**对比气泡流水线对应程序总周期数少35，插入气泡数少35**



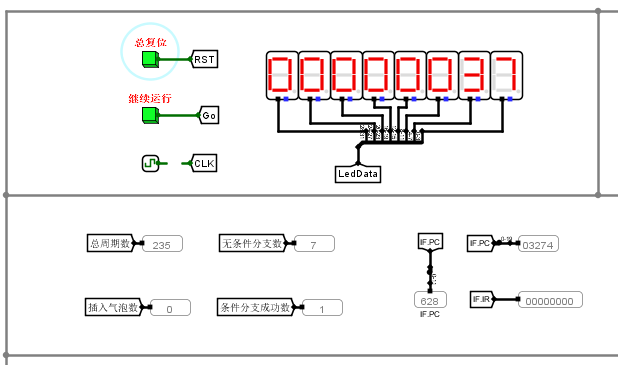
**test3.hex**

**对比气泡流水线对应程序总周期数少16，插入气泡数少16**



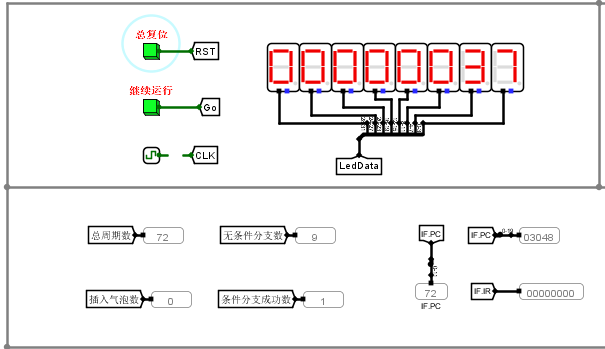
**fib\_mips.hex**

**对比气泡流水线对应程序总周期数少88，插入气泡数少88**



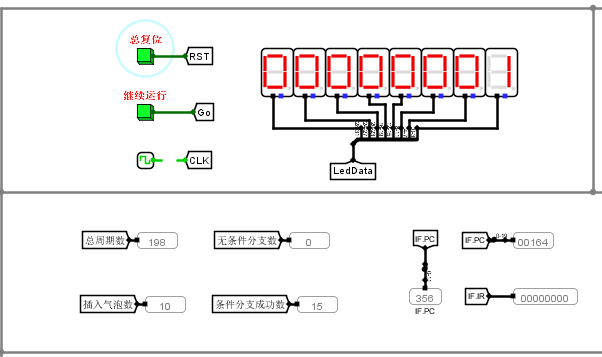
**sum\_mips.hex**

**对比气泡流水线对应程序总周期数少6，插入气泡数少6**



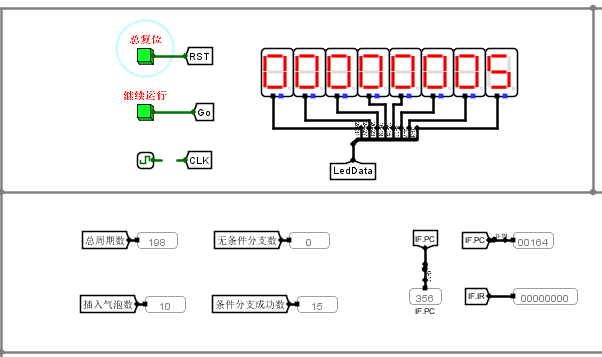
**sort1\_mips.hex**

**对比气泡流水线对应程序总周期数少81，插入气泡数少81**



**sort2\_mips.hex**

**对比气泡流水线对应程序总周期数少81，插入气泡数少81**



**综上，重定向流水线MIPS处理器，对比气泡型流水线MIPS处理器，其插入气泡数大量减少，从而大大提升性能。**

**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，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？**

**依据logisim电路所得时空图：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CLKs** | **IF** | **ID** | **EX** | **MEM** | **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** |  | **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** |
| **9** |  |  |  | **and $9,$7,$6** | **or $7,$6,$7** |
| **10** |  |  |  |  | **and $9,$7,$6** |

**与教材所给图基本吻合**



**3)分析重定向流水线MIPS处理器的数据通路电路。**

取指阶段：与气泡型流水线MIPS处理器相同，取指阶段也是从指令内存中读取指令，然后将指令送往译码阶段。不同的是，由于重定向的存在，取指阶段需要额外考虑分支指令等可能会改变PC值的情况。在重定向流水线中，取指阶段的PC值来源于分支逻辑电路中计算出的目标地址，而非指令顺序加1得到的下一条指令的地址。

译码阶段：与气泡型流水线MIPS处理器相同，译码阶段也是将指令进行解码，生成相应的控制信号和寄存器操作数。在重定向流水线中，译码阶段需要考虑上一条指令是否由于数据相关性被暂停流水线，因此需要实现数据相关检测电路，并根据检测结果，更新当前指令的控制信号和操作数。同时，如果上一条指令正在运行的是访存阶段，那么该阶段的结果需要通过转发回传给当前指令的两个源操作数，以便在执行阶段中正确使用这些操作数。

执行阶段：与气泡型流水线MIPS处理器相同，执行阶段是指令的实际执行阶段。不同的是，在重定向流水线中，执行阶段需要额外考虑数据相关性和异常情况，因此需要实现数据关联检测电路和异常处理电路。当数据相关性发生时，执行阶段需要等待前面的指令完成后才能继续执行，否则会导致错误的结果。如果发生异常，执行阶段也需要停止执行，并将控制权传递给异常处理电路进行处理。

访存阶段：与气泡型流水线MIPS处理器相同，访存阶段也是针对特定指令（如lw、sw等）从内存中读取或写入数据的阶段。在重定向流水线中，访存阶段同样需要解决数据相关性问题，在执行阶段完成后进行访存操作之前，需要等待前面指令的执行阶段和访存阶段完成。除此之外，在访存阶段还需要处理数据重定向的情况，即当前指令的结果需要被传递给前面的指令，以便前面的指令在执行阶段中使用这个结果，来避免产生气泡等待。

写回阶段：与气泡型流水线MIPS处理器相同，写回阶段是将执行完毕的结果写回到目标寄存器中的阶段。在重定向流水线中，写回阶段需要考虑数据重定向的情况，即如果当前指令的结果需要被转发给前面的指令，那么就不能直接写回到目标寄存器，而是需要通过转发网将结果传递给前面的指令，以便前面的指令在执行阶段中正确使用这个结果。

**4)分析重定向流水线MIPS处理器的重定向检测电路。**

重定向检测电路对当前指令的源寄存器操作数与前一条指令的目标寄存器进行比较，以判断当前指令是否依赖于前面指令的结果。如果当前指令依赖于前面指令的结果，则暂停当前指令的执行，等待前面的指令完成并将结果写回，然后再重新启动当前指令的执行。这个过程通过转发网来实现，将前面指令的结果传递给当前指令。当发生数据相关性时，转发网会将前面指令的结果传递给当前指令使用，从而避免了执行气泡等待的情况，提高了流水线的利用率和性能。

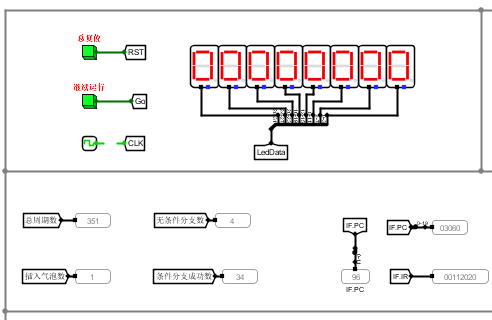
1. **在动态分支预测流水线MIPS处理器上运行测试程序。**

**1）在动态分支预测流水线MIPS处理器的数据通路上运行test1.hex、test2.hex、test3.hex、test4.hex、fib\_mips.hex、sum\_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？**

**test1.hex**

**对比重定向MIPS处理器，总周期数少57，插入气泡数多1**

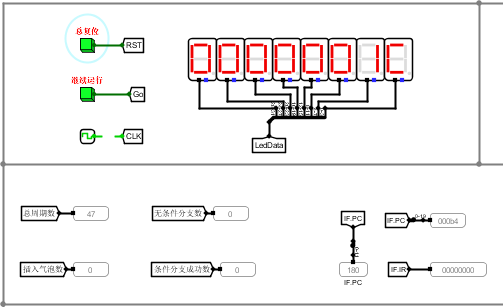
**对比气泡型MIPS处理器，总周期数少58，插入气泡数相同**



**test2.hex**

**对比重定向MIPS处理器，总周期数多1，插入气泡数少1**

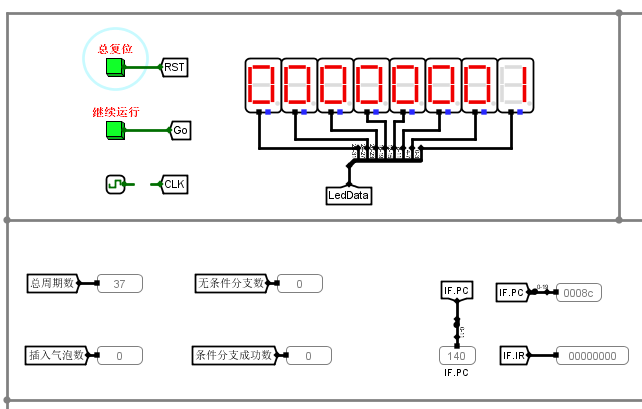
**对比气泡型MIPS处理器，总周期数少34，插入气泡数少36**



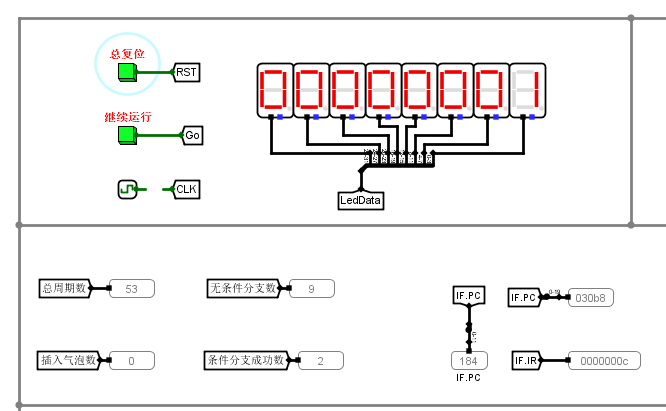
**test3.hex**

**对比重定向MIPS处理器，总周期数多2，插入气泡数不变**

**对比气泡型MIPS处理器，总周期数少14，插入气泡数少16**



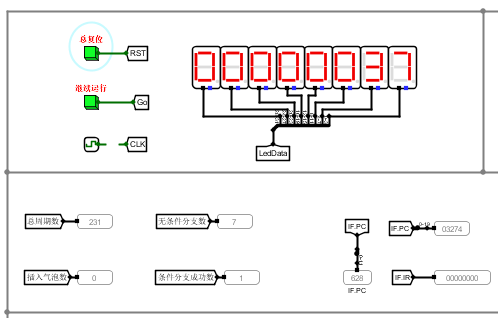
**test4.hex**



**fib\_mips.hex**

**对比重定向MIPS处理器，总周期数少4，插入气泡数不变**

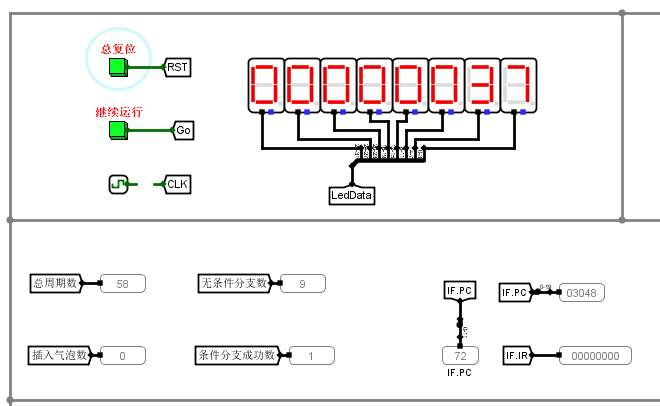
**对比气泡型MIPS处理器，总周期数少92，插入气泡数少88**



**sum\_mips.hex**

**对比重定向MIPS处理器，总周期数少14，插入气泡数不变**

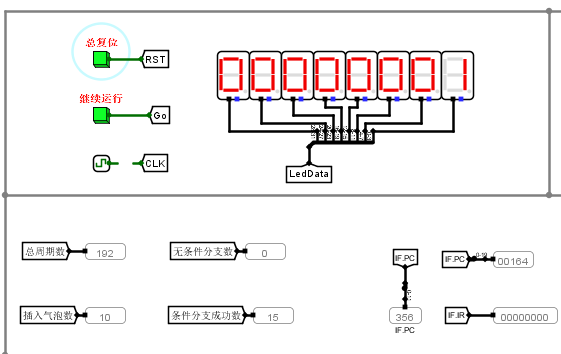
**对比气泡型MIPS处理器，总周期数少20，插入气泡数少6**



**sort1\_mips.hex**

**对比重定向MIPS处理器，总周期数少6，插入气泡数不变**

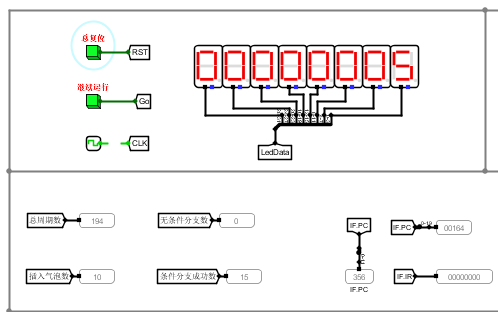
**对比气泡型MIPS处理器，总周期数少87，插入气泡数少81**



**sort2\_mips.hex**

**对比重定向MIPS处理器，总周期数少4，插入气泡数不变**

**对比气泡型MIPS处理器，总周期数少85，插入气泡数少81**



**综上，动态分支预测流水线MIPS处理器相对于气泡型流水线MIPS处理器总周期数和插入气泡数都有大幅减少，但是相比于重定向型流水线MIPS处理器总周期数和插入气泡数却不一定减少，这一定程度上取决于分支预测是否成功。**

**2)分析动态分支预测流水线MIPS处理器的数据通路电路。**

取指阶段：与重定向流水线MIPS处理器和气泡型流水线MIPS处理器类似，动态分支预测流水线MIPS处理器在取指阶段通过PC寄存器传递下一条指令的地址。但是，为了更好地实现分支预测和分支延迟槽等功能，动态分支预测流水线MIPS处理器还配备了分支预测器和跳转目标缓存这两个单元，用于提前预测分支指令的执行结果并缓存跳转指令的目标地址，从而加快分支指令的执行速度和准确性。

译码阶段：在译码阶段，动态分支预测流水线MIPS处理器会从指令存储器中读出当前指令，然后进行指令解析和寄存器读取操作。与重定向流水线MIPS处理器不同的是，动态分支预测流水线MIPS处理器通过分支预测器来预测分支指令的执行情况，并根据预测结果的正确性决定是否需要执行分支延迟槽，从而加快程序的执行效率和速度。

执行阶段：在执行阶段，动态分支预测流水线MIPS处理器通过ALU来实现指令的各种算术和逻辑运算。同时，动态分支预测流水线MIPS处理器还配备了数据前向单元来处理数据相关问题，从而进一步提高程序执行效率。与气泡型流水线MIPS处理器不同的是，动态分支预测流水线MIPS处理器通过分支预测器来预测分支指令的执行结果，并在预测正确时继续执行下一条指令，从而避免了由于预测错误而导致的浪费时间和资源问题。

访存阶段：在访存阶段，动态分支预测流水线MIPS处理器通过数据存储器来进行对内存数据的读写操作。与重定向流水线MIPS处理器和气泡型流水线MIPS处理器类似，动态分支预测流水线MIPS处理器在访存过程中也需要考虑数据依赖问题，以确保正确性和稳定性。

写回阶段：在写回阶段，动态分支预测流水线MIPS处理器会将处理结果写回到寄存器堆中。与重定向流水线MIPS处理器不同的是，动态分支预测流水线MIPS处理器在写回阶段还需要进行分支延迟槽的判断和处理，以确保程序的正确性和稳定性。同时，在写回阶段，动态分支预测流水线MIPS处理器还需要将预测器的状态作出相应的调整，以进一步提高程序执行效率和准确性。

**3)分析动态分支预测流水线MIPS处理器的BHT（分支历史表）电路。**

BHT电路是由两个部分构成：一是历史寄存器，用于存储分支指令的历史信息；二是BHT表格，用于存储预测器的状态和预测结果。

历史寄存器是一个N位的寄存器，其中N表示历史长度，即能够纪录的分支历史最多的条目数。在每条指令执行过程中，历史寄存器会将当前指令的分支情况（即是否为分支指令以及分支方向）记录下来，然后将其作为BHT表格的索引值，从而能够快速定位到相应的预测器状态和预测结果。

BHT表格是一个二维数组，其中第一维表示预测器状态的数量，第二维表示预测结果的数量。例如，对于一种“2-bit saturating counter”预测器，其状态数量为4（00、01、10、11），预测结果数量为2（0、1）。因此，该预测器的BHT表格就是一个4×2的数组，每个数组元素都对应着一个预测器状态和预测结果的组合。

当BHT表格被访问时，它会根据历史寄存器中的值来选择相应的预测器状态，并返回预测结果。如果返回的预测结果与实际执行结果一致，则预测器状态保持不变，否则则会根据实际结果调整预测器状态，以提高预测正确率。

请按照实验课件的要求，验证测试程序的运行结果，将运行结果黏贴到实验报告中，给出相关的文字说明，并对有关的电路原理进行分析。