《计算机组成原理实验》

(第六次)

厦门大学信息学院软件工程系　曾文华

2023年6月1日

目录

1、理想流水线MIPS处理器

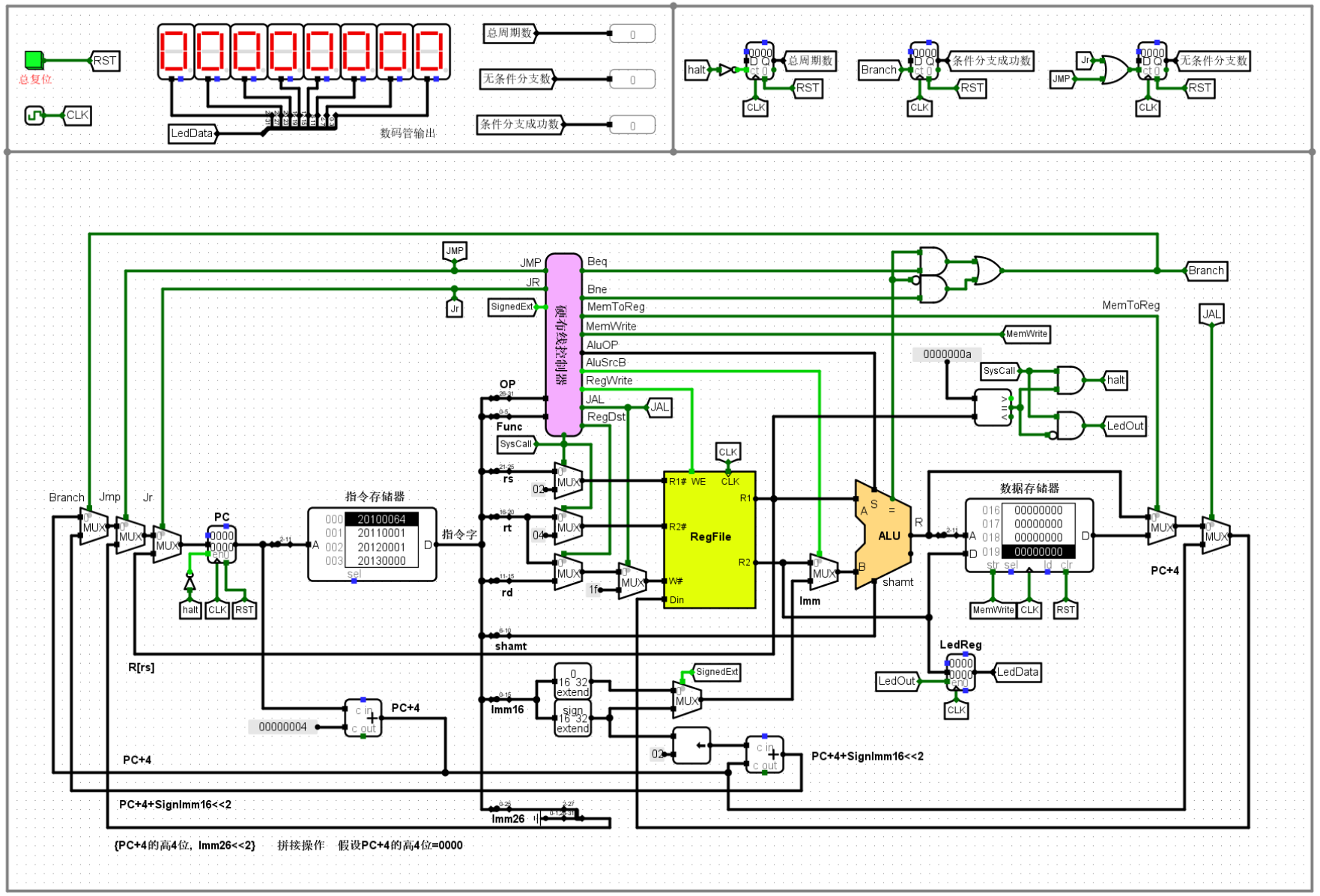
- 理想流水线MIPS处理器分为5个阶段：
  ① IF（取指）
  ② ID（译码）
  ③ EX（执行）
  ④ MEM（访存）
  ⑤ WB（写回）

- 通过在单周期MIPS处理器的数据通路上增加4个流水线寄存器，即可得到理想流水线MIPS处理器：
  ① IF/ID
  ② ID/EX
  ③ EX/MEM
  ④ MEM/WB

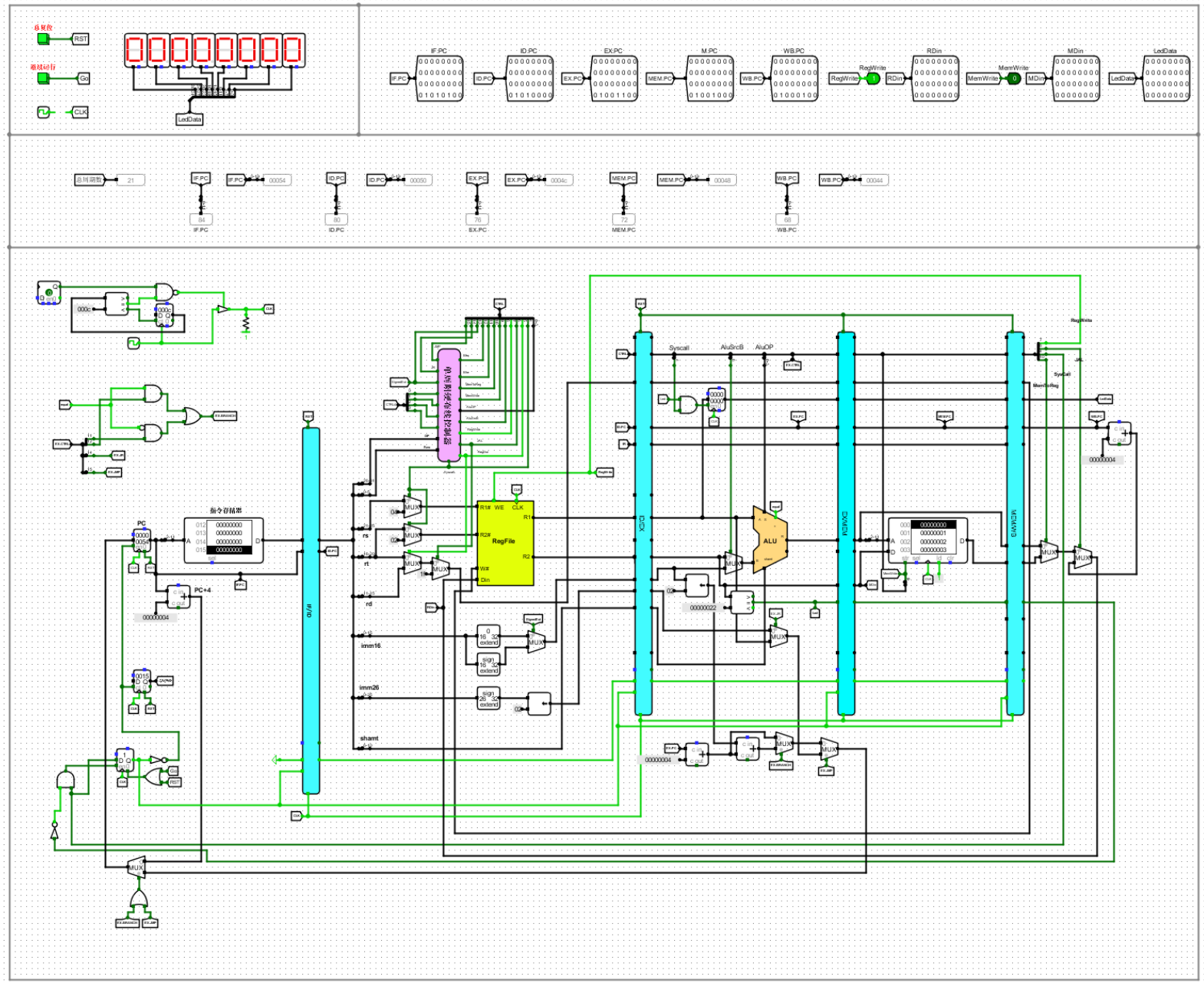
- 流水寄存器用于锁存前一段功能部件加工处理完成的数据和控制信号，为下一段的功能部件提供数据输入和控制信号。

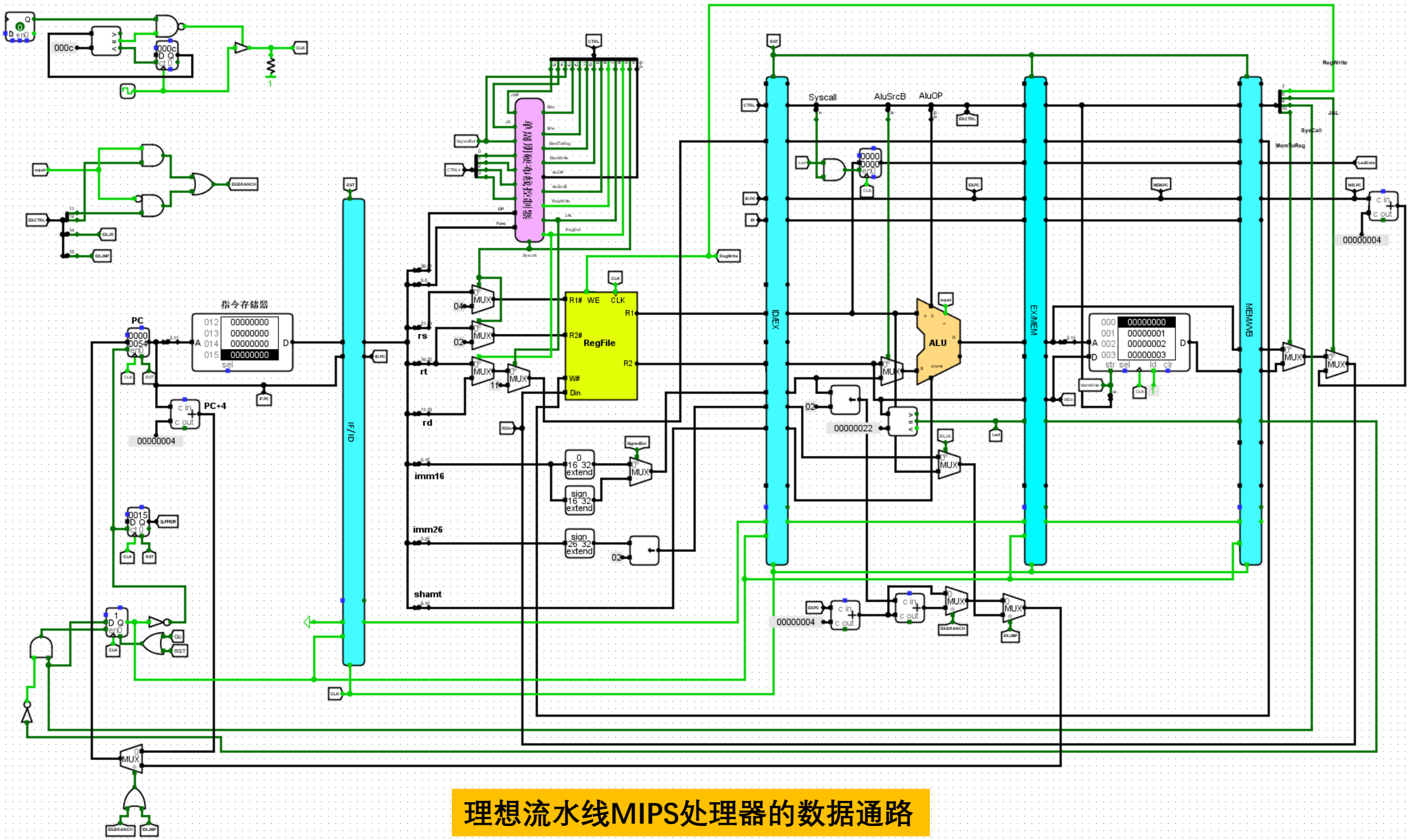
单周期 MIPS 处理器（硬布线控制器）数据通路

支持24条指令



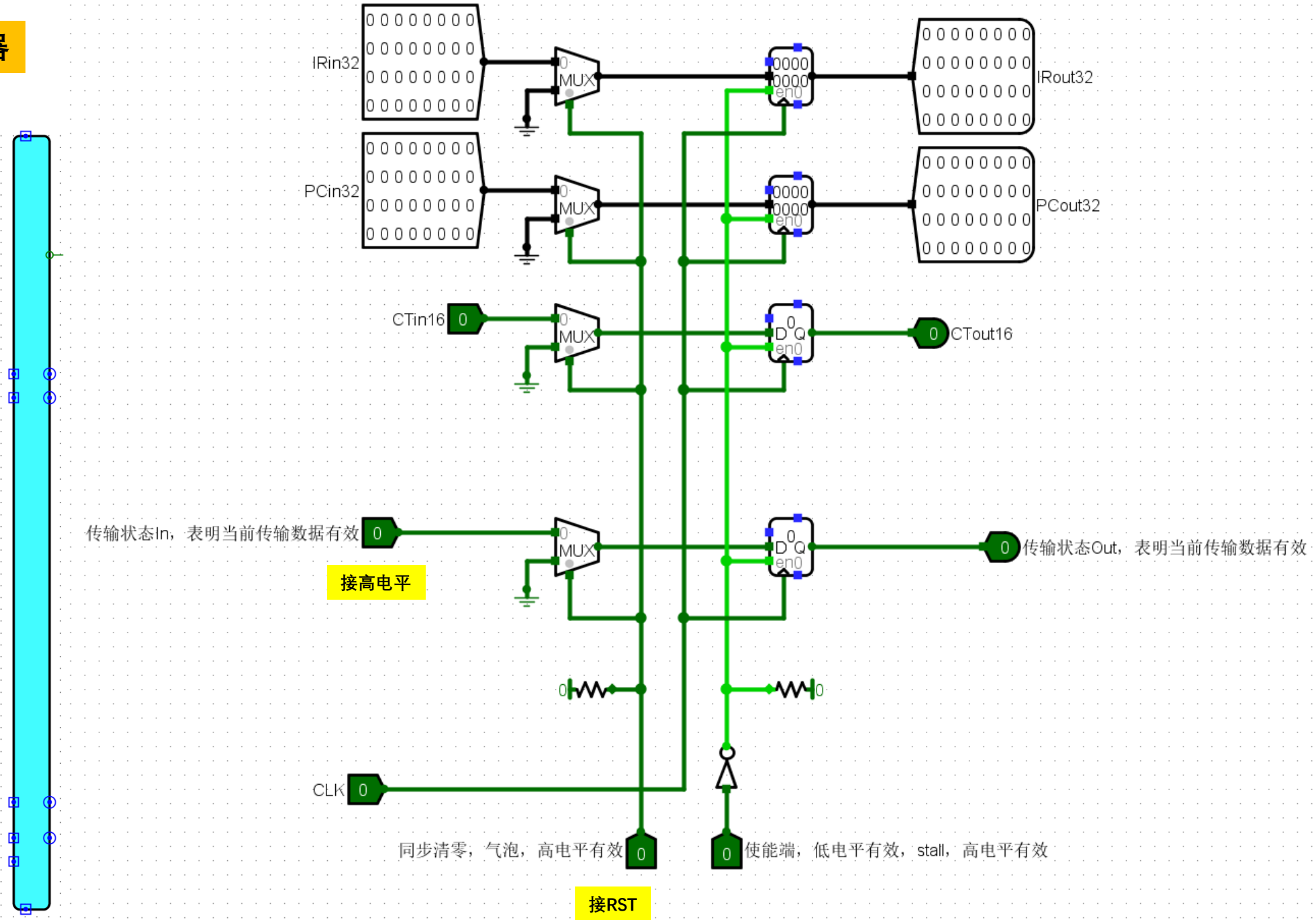
理想流水线MIPS处理器



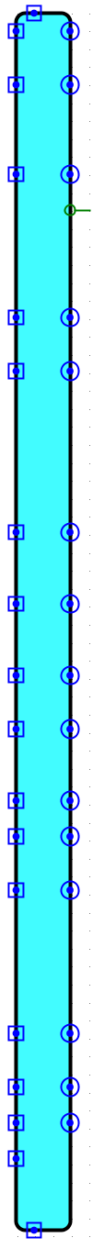


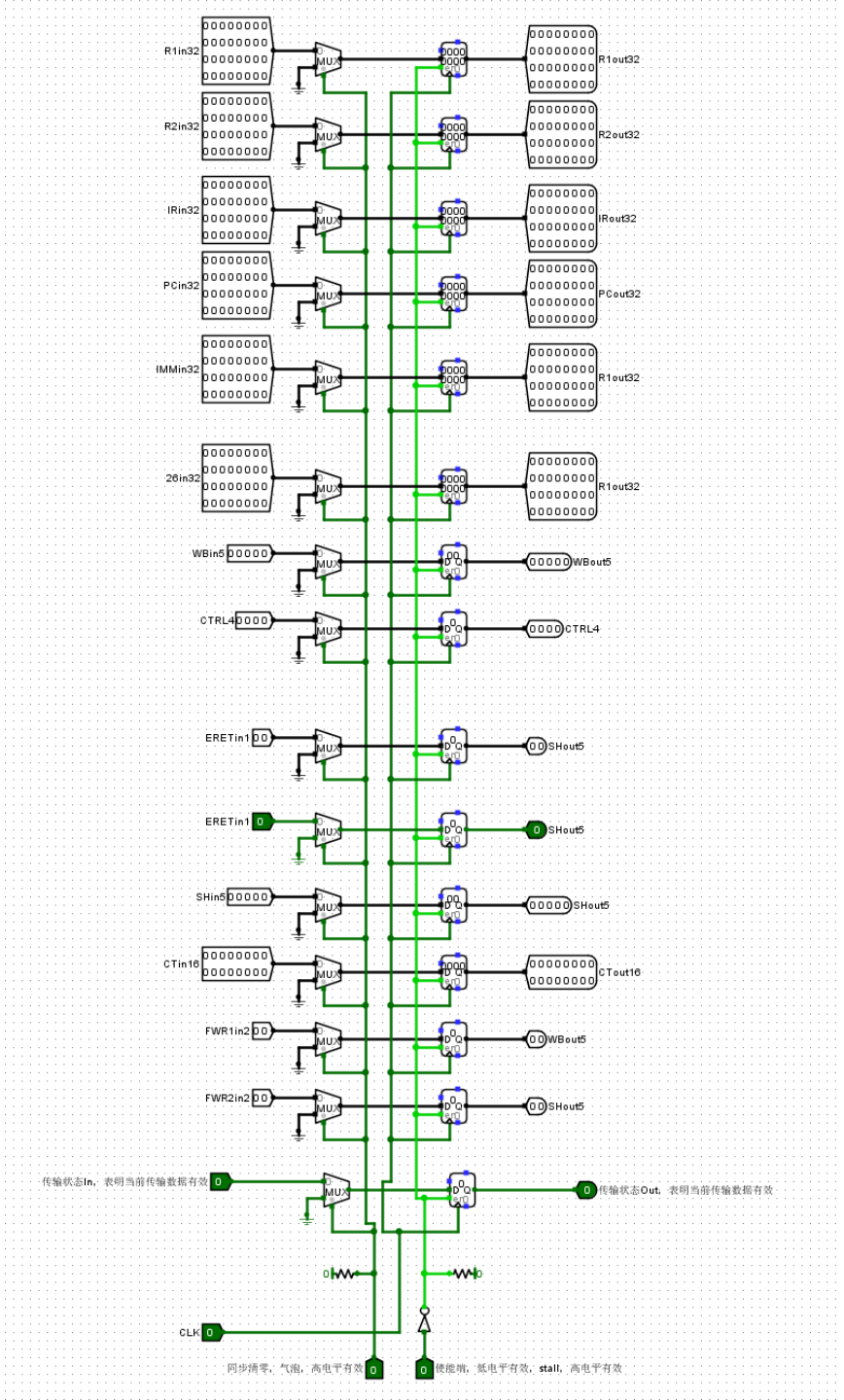
理想流水线MIPS处理器的数据通路

① IF/ID流水寄存器

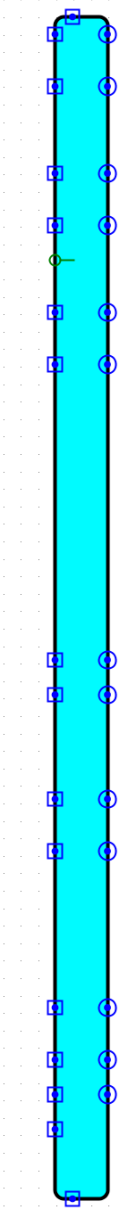

IRin32
PCin32
CTin16
传输状态In，表明当前传输数据有效
接高电平
CLK
同步清零，气泡，高电平有效
接RST
使能端，低电平有效，stall，高电平有效
IRout32
PCout32
CTout16
传输状态Out，表明当前传输数据有效
MUX
D
Q
en

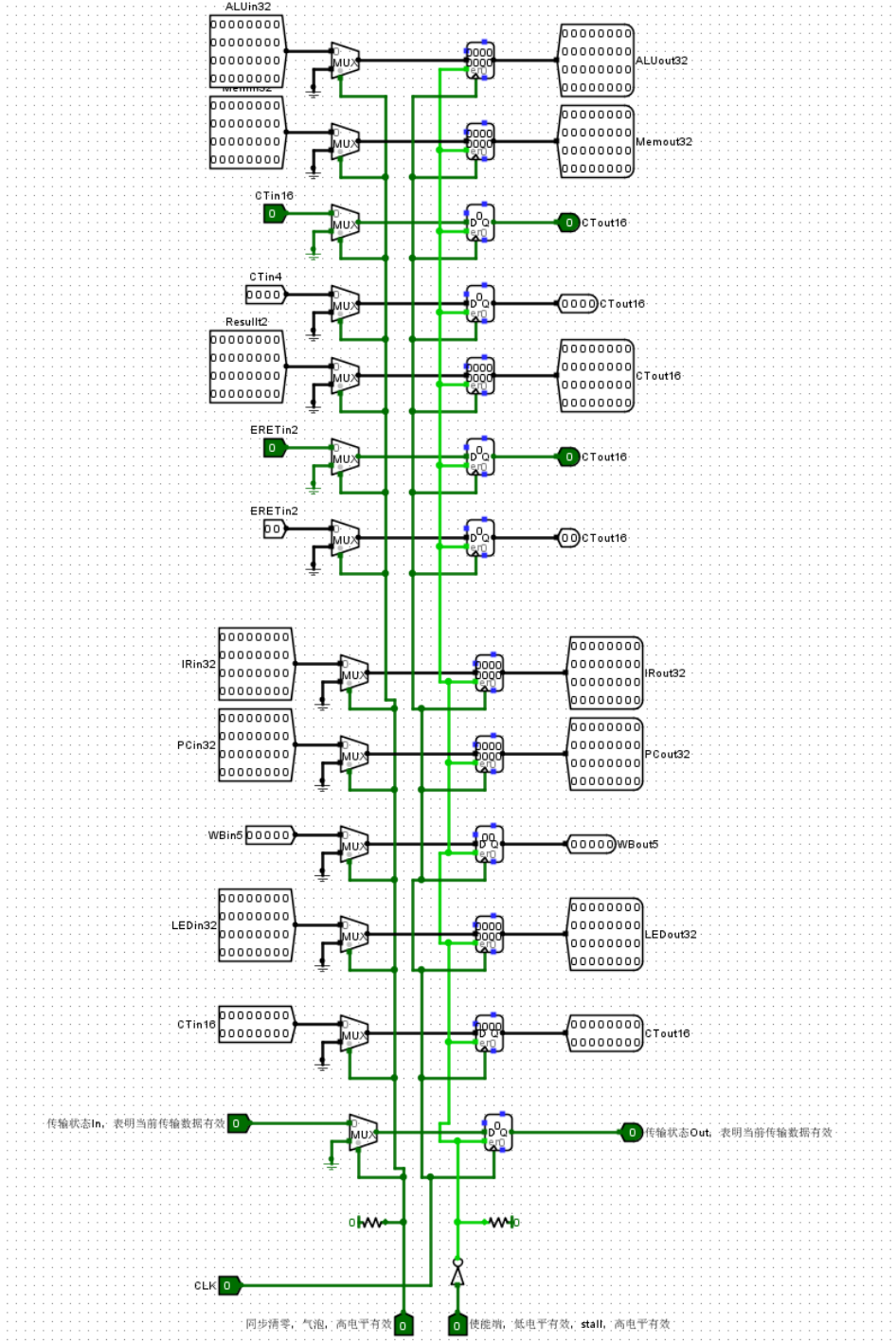
② ID/EX流水寄存器



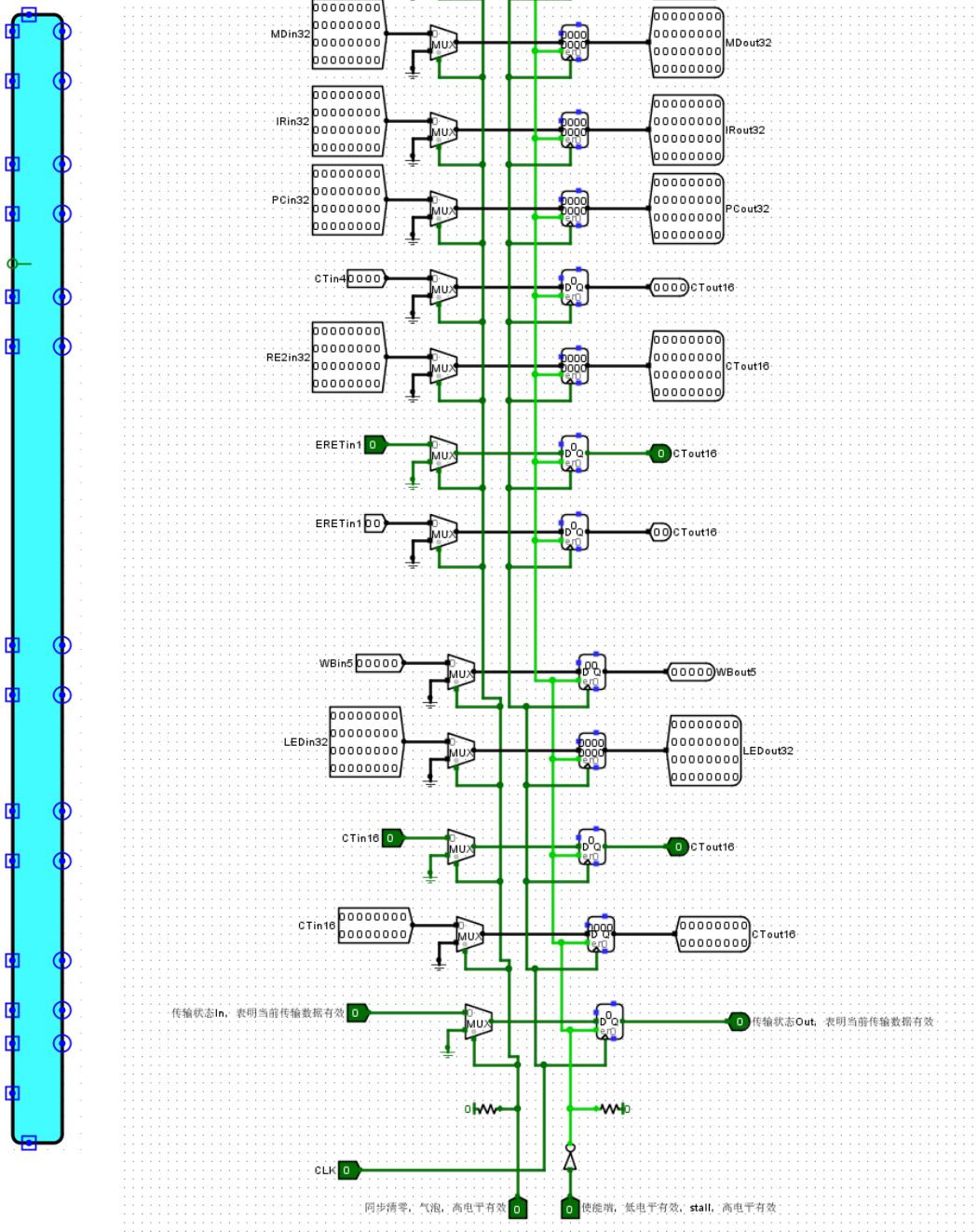

R1in32
R1out32
R2in32
R2out32
IRin32
IRout32
PCin32
PCout32
IMMin32
R1out32
26in32
R1out32
WBin5
WBout5
CTRL4
CTRL4
ERETin1
SHout5
ERETin1
SHout5
SHin5
SHout5
CTin16
CTout16
FWR1in2
WBout5
FWR2in2
SHout5
传输状态In，表明当前传输数据有效
传输状态Out，表明当前传输数据有效
CLK
同步清零，气泡，高电平有效
使能端，低电平有效，stall，高电平有效

③ EX/MEM流水寄存器



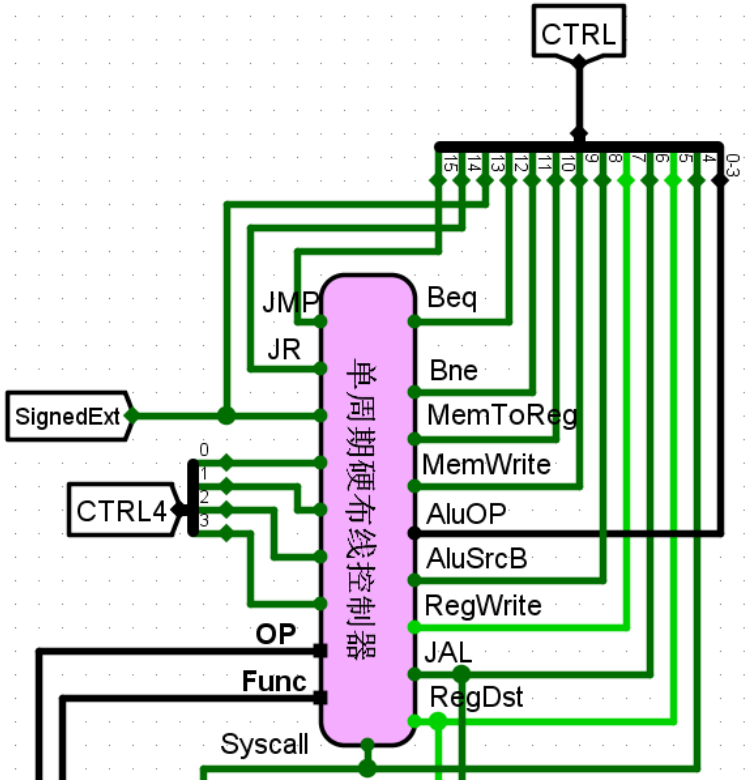


④ MEM/WB流水寄存器



- CTRL（EX.CTRL）共16位：
  - 0-3：AluOP（运算器操作控制信号）
  - 4：Syscall（Syscall指令译码信号）
  - 5：RegDst（目的寄存器选择信号）
  - 6：JAL（JAL指令译码信号）
  - 7：RegWrite（寄存器写控制信号）
  - 8：AluSrcB（ALU的B口选择控制信号）
  - 9：MemWrite（存储器写控制信号）
  - 10：MemToReg（存储器输出送寄存器控制信号）
  - 11：Bne（不等则转的条件转移指令译码信号）
  - 12：Beq（相等则转的条件转移指令译码信号）
  - 13：SignedExt（符号扩展控制信号）
  - 14：JR（JR指令译码信号）
  - 15：JMP（JMP指令译码信号）

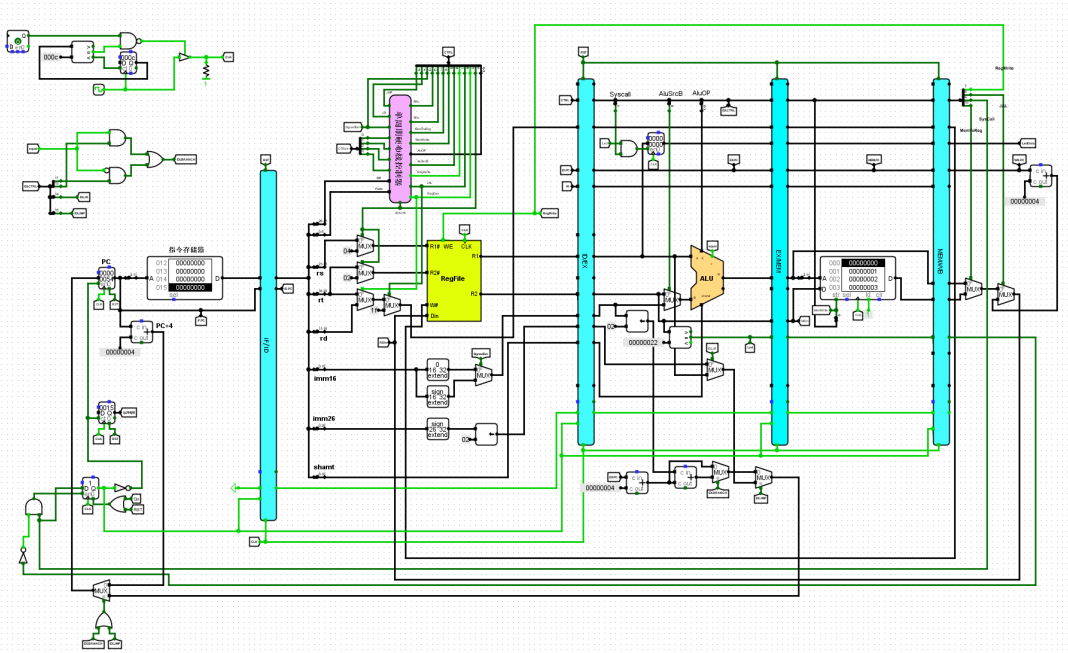
- CTRL4共4位（这里没有使用）：
  - 0：XORI（XORI指令译码信号）
  - 1：SRAV（SRAV指令译码信号）
  - 2：LH（LH指令译码信号）
  - 3：BLTZ（BLTZ指令译码信号）



- **数据通路中的其他信号：**
  - IF.PC：IF段的PC值
  - IF.IR：IF段的IR值
  - ID.PC ： ID段的PC值
  - ID.IR：ID段的IR值
  - EX.PC ： EX段的PC值
  - EX.IR：EX段的IR值
  - MEM.PC ： MEM段的PC值
  - MEM.IR：MEM段的IR值
  - WB.PC ： WB段的PC值
  - WB.IR：WB段的IR值

  - EX.BRANCH： EX段的条件分支控制信号（包括Beq和Bne指令）
  - EX.JMP：EX段的JMP控制信号（对应JMP指令）
  - EX.JR：EX段的JR控制信号（对应JR指令）
  - MemWrite：存储器写控制信号
  - MDin：数据存储器写入数据（32位）
  - RDin：寄存器堆写入数据（32位）
  - euqal：两个操作数相等时，equal=1
  - Led：当Syscall指令使用34号功能（34=22H）时，Led=1
  - LedData：当Led=1时，LedData为a0寄存器的值（32位），送8个数码管显示

  - CLK：时钟
  - RST：复位信号
  - Go：继续运行控制信号
  - 总周期数：程序执行的总时钟周期数

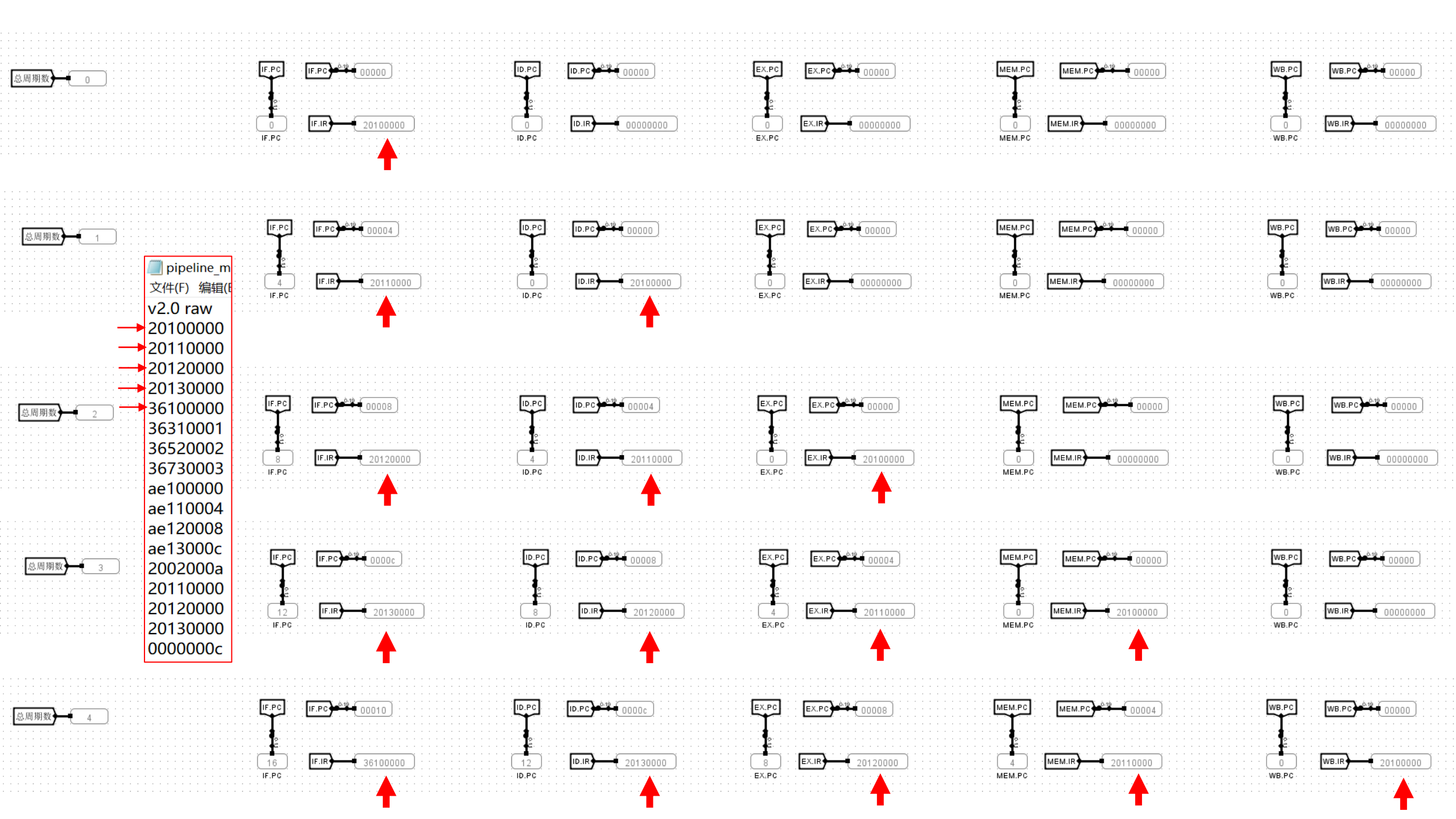


测试程序
test.asm

test.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#理想流水线测试程序    test.asm
#该程序将0、1、2、3分别存放到0、1、2、3号存储器中（地址为0、4、8、12）
#该程序中的所有指令（17条）均无相关性
#5段流水线执行周期数=5+(17-1)=21

addi $s0,$zero,0        # 将4个寄存器s0至s3全部赋值为0
addi $s1,$zero,0
addi $s2,$zero,0
addi $s3,$zero,0

ori $s0,$s0,0           # 将4个寄存器s0至s3分别赋值为0、1、2、3
ori $s1,$s1,1
ori $s2,$s2,2
ori $s3,$s3,3

sw $s0,0($s0)        # 将4个寄存器s0至s3分别存储到地址为0、4、8、12的存储单元中
sw $s1,4($s0)
sw $s2,8($s0)
sw $s3,12($s0)

addi $v0,$zero,10       #v0=10    设置syscall指令的编号（10号表示停机）

addi $s1,$zero,0        #3条无用指令，消除syscall指令与“addi $v0,$zero,10”指令的相关性
addi $s2,$zero,0
addi $s3,$zero,0

syscall                 #停机
```


总周期数 0
IF.PC 00000
IF.IR 20100000
ID.PC 00000
ID.IR 00000000
EX.PC 00000
EX.IR 00000000
MEM.PC 00000
MEM.IR 00000000
WB.PC 00000
WB.IR 00000000
总周期数 1
IF.PC 00004
IF.IR 20110000
ID.PC 00000
ID.IR 20100000
EX.PC 00000
EX.IR 00000000
MEM.PC 00000
MEM.IR 00000000
WB.PC 00000
WB.IR 00000000
pipeline_m
文件(F) 编辑(E
v2.0 raw
20100000
20110000
20120000
20130000
36100000
36310001
36520002
36730003
ae100000
ae110004
ae120008
ae13000c
2002000a
20110000
20120000
20130000
0000000c
总周期数 2
IF.PC 00008
IF.IR 20120000
ID.PC 00004
ID.IR 20110000
EX.PC 00000
EX.IR 20100000
MEM.PC 00000
MEM.IR 00000000
WB.PC 00000
WB.IR 00000000
总周期数 3
IF.PC 0000c
IF.IR 20130000
ID.PC 00008
ID.IR 20120000
EX.PC 00004
EX.IR 20110000
MEM.PC 00000
MEM.IR 20100000
WB.PC 00000
WB.IR 00000000
总周期数 4
IF.PC 00010
IF.IR 36100000
ID.PC 0000c
ID.IR 20130000
EX.PC 00008
EX.IR 20120000
MEM.PC 00004
MEM.IR 20110000
WB.PC 00000
WB.IR 20100000

- 要求：
  - （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。
  - （2）分析理想流水线MIPS处理器的数据通路电路。
  - （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）。

2、气泡流水线MIPS处理器

- 前面的理想流水线MIPS处理器存在：
  ① 控制冲突（分支冲突）
  ② 数据冲突

- 通过在前面的理想流水线MIPS处理器上，增加数据相关检测电路以及相应的控制信号，就可以解决数据冲突和控制冲突（分支冲突）。

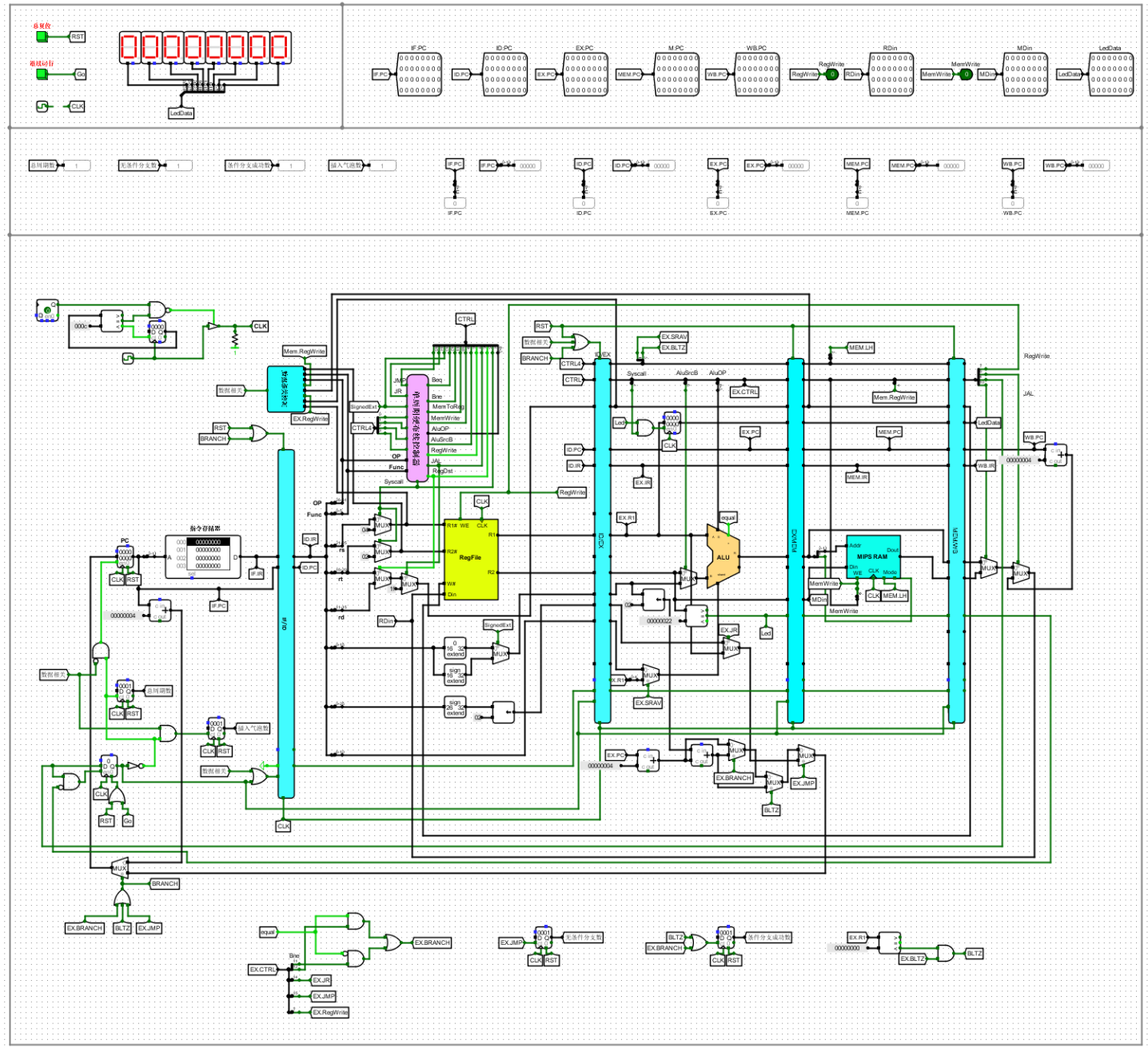
气泡流水线的控制信号

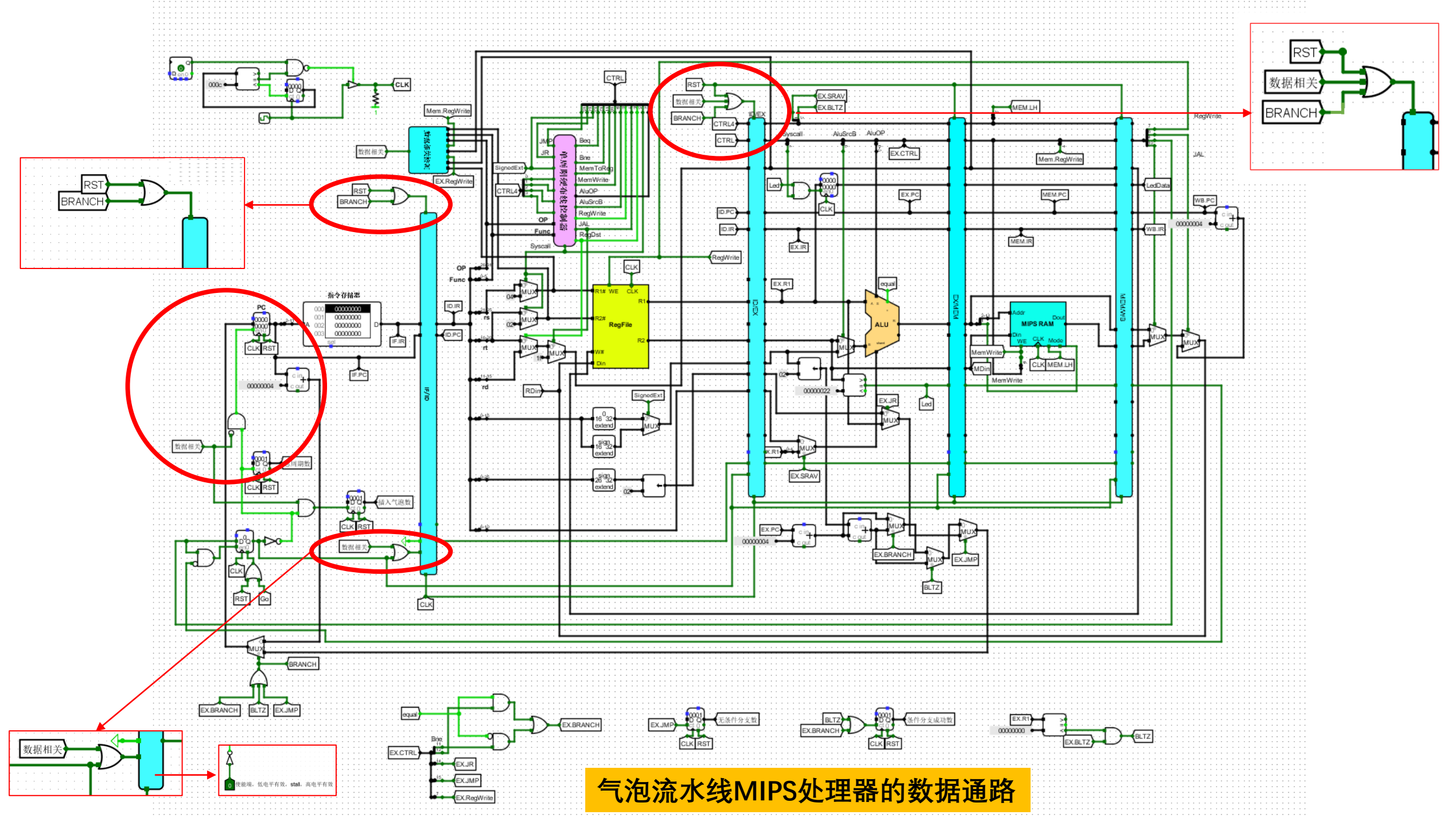
```
Stall = DataHazzard                          #数据相关时，要阻塞（暂停）IF、ID段指令的执行
PC.EN = ~Stall                               #阻塞IF段指令的执行（程序计数器PC使能端输入=0）
IF/ID.EN = ~Stall                            #阻塞ID段指令的执行（IF/ID流水寄存器使能端输入=0）

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
```

数据相关检测信号

```
DataHazzard = RsUsed & (rs≠0) & EX.RegWrite & (rs==EX.WriteReg#)
            + RtUsed & (rt≠0) & EX.RegWrite & (rt==EX.WriteReg#)
            + RsUsed & (rs≠0) & MEM.RegWrite & (rs==MEM.WriteReg#)
            + RtUsed & (rt≠0) & MEM.RegWrite & (rt==MEM.WriteReg#)
```

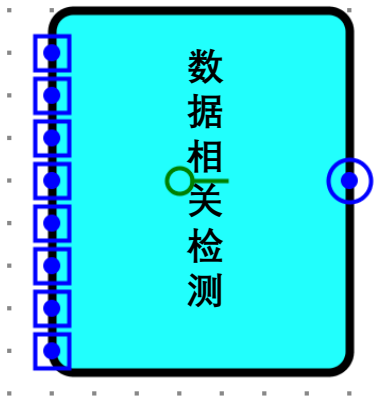
气泡流水线MIPS处理器

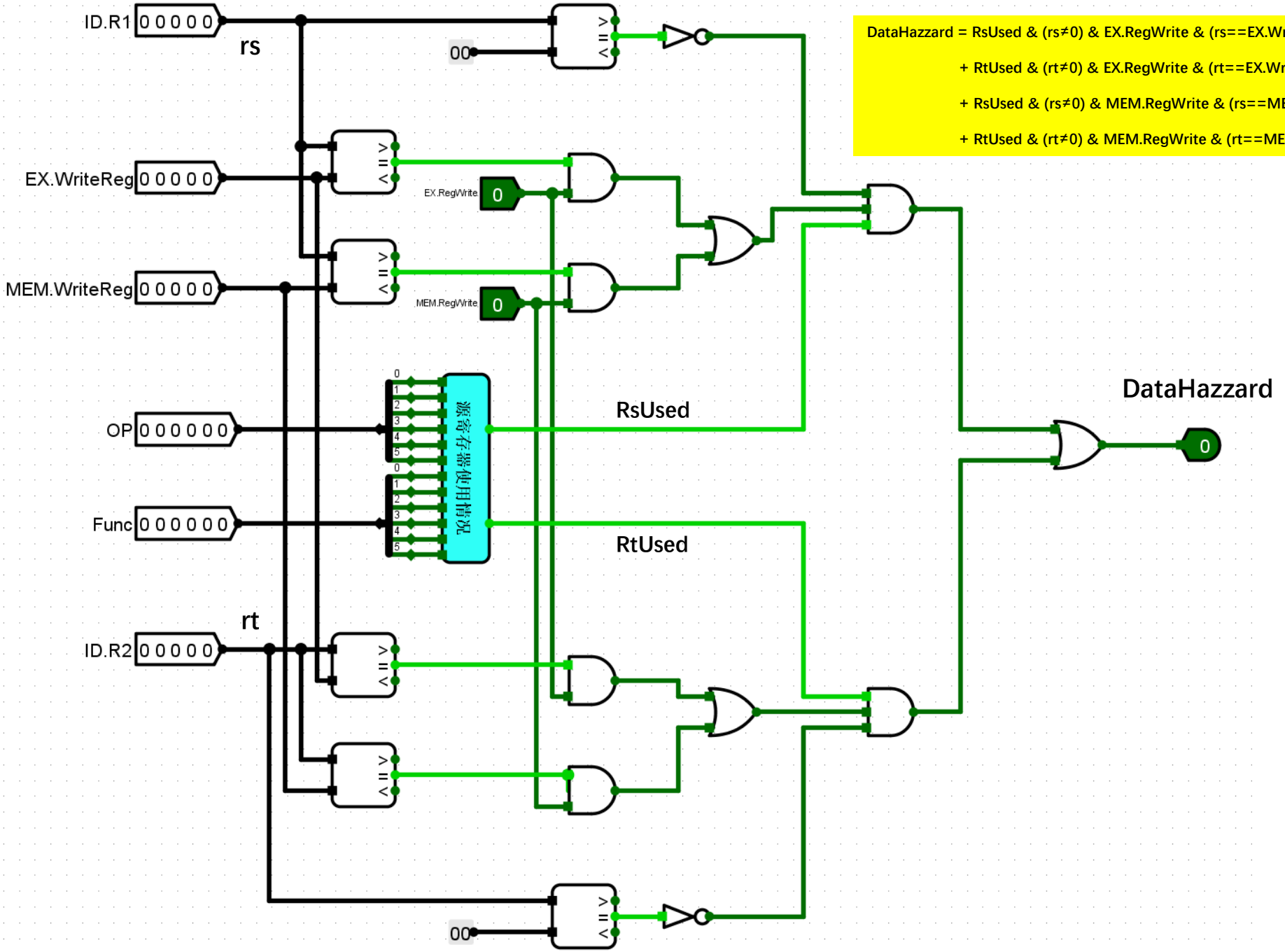




气泡流水线MIPS处理器的数据通路

数据相关检测电路





DataHazzard = RsUsed & (rs≠0) & EX.RegWrite & (rs==EX.WriteReg#)

+ RtUsed & (rt≠0) & EX.RegWrite & (rt==EX.WriteReg#)

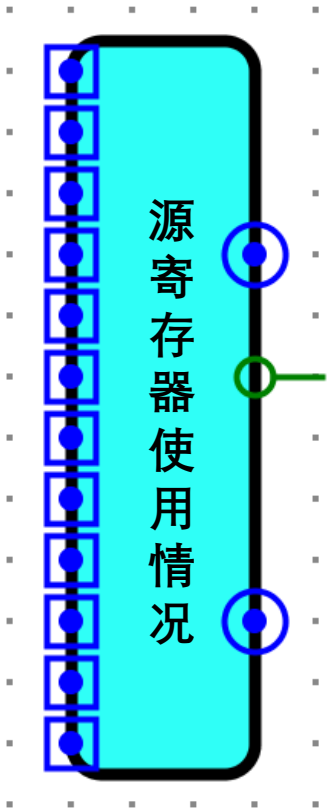
+ RsUsed & (rs≠0) & MEM.RegWrite & (rs==MEM.WriteReg#)

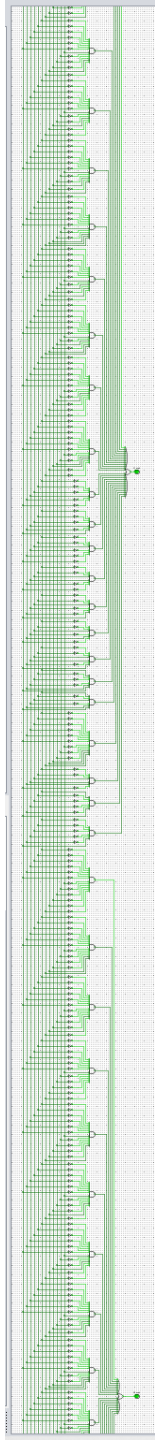
+ RtUsed & (rt≠0) & MEM.RegWrite & (rt==MEM.WriteReg#)

源寄存器使用情况

输入：
OP
Func

输出：
RsUsed
RtUsed





测试程序1
test1.asm

测试5条分支指令（j、jal、jr、beq、bne）的控制相关

test1.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#气泡流水线分支相关测试程序  test1.asm
#该程序测试j、jal、jr、beq、bne等分支指令的相关性，采用气泡流水线后，该程序无分支相关
#程序功能：s1的初值为32，调用子程序后，s1减1，变为31（1FH），然后在数码管上显示，之后再减1，变为30（1EH），在数码管上显示，直到变为0
#该程序执行后，过一段时间，数码管显示1F，之后减1、显示1E，一直到0
        addi $s1,$zero,32
        addi $v0,$zero,34               #34=22H     syscall的34号功能（8个数码管显示8个十六进制数，即32位二进制数）
        j jmp_next1                     #跳转到jmp_next1

        addi $s1,$zero,4                #接下去3条指令不会执行
        addi $s2,$zero,5
        addi $s3,$zero,6

jmp_next1:
        beq $zero,$zero, jmp_next2      #相当于j指令    跳转到jmp_next2

        addi $s1,$zero,7                #接下去3条指令不会执行
        addi $s2,$zero,8
        addi $s3,$zero,9

jmp_next2:
        bne $zero,$s1, jmp_next3        #因为s1=32  不等于0  因此跳转到jmp_next3

        addi $s1,$zero,10               #接下去3条指令不会执行
        addi $s2,$zero,11
        addi $s3,$zero,12

jmp_next3:
        jal jmp_func                    #子程序调用指令

        addi $v0,$zero,10               #syscall的10号功能      子程序返回到这里
        nop
        nop
        nop
        syscall                         #系统退出

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
```

测试程序2
test2.asm

测试数据相关

test2.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#气泡流水线测试程序——数据相关测试    test2.asm
#该程序用于测试数据相关，使用气泡流水线后，该程序无数据相关
#该程序计算：0+1+2+3+4+5+6+7=28=1CH
#该程序执行结束后，数码管显示1C
 addi $s1,$0,4          #s1=4
 sw $s1,0($s1)          # 4 -> (4)
 lw $s2,0($s1)          # (4) -> s2=4

 addi $s2,$s2,-4        # s2-4=0 -> s2=0
 addi $s0,$0,0          # s0=0

 addi $s1,$s0,1         #s1=1
 add $s0,$s0,$s1        #s0=1
 addi $s2,$s2,4         #s2=4
 sw $s0,0($s2)          #1 -> (4)   04H

 addi $s1,$s1,1         #s1=2
 add $s0,$s0,$s1        #s0=1+2=3
 addi $s2,$s2,4         #s2=8
 sw $s0,0($s2)          #3 -> (8)    08H

 addi $s1,$s1,1         #s1=3
 add $s0,$s0,$s1        #s0=1+2+3=6
 addi $s2,$s2,4         #s2=12
 sw $s0,0($s2)          #6 -> (12)   0CH

 addi $s1,$s1,1         #s1=4
 add $s0,$s0,$s1        #s0=1+2+3+4=10
 addi $s2,$s2,4         #s2=16
 sw $s0,0($s2)          #10 -> (16)   10H

 addi $s1,$s1,1         #s1=5
 add $s0,$s0,$s1        #s0=1+2+3+4+5=15
 addi $s2,$s2,4         #s2=20
 sw $s0,0($s2)          #15 -> (20)   14H

 addi $s1,$s1,1         #s1=6
 add $s0,$s0,$s1        #s0=1+2+3+4+5+6=21
 addi $s2,$s2,4         #s2=24
 sw $s0,0($s2)          #21 -> (24)   18H

 addi $s1,$s1,1         #s1=7
 add $s0,$s0,$s1        #s0=1+2+3+4+5+6+7=28
 addi $s2,$s2,4         #s2=28
 sw $s0,0($s2)          #28 -> (28)    1CH

 add $a0,$0,$s0                 #a0=s0     a0送数码管显示
 addi $v0,$zero,34              #syscall的34号功能（在数码管上显示a0的值）
 syscall

 addi $v0,$zero,10              #syscall的10号功能（系统退出）
 addi $s0,$zero,0
 addi $s0,$zero,0
 addi $s0,$zero,0
 syscall
```

测试程序3
test3.asm

测试教材P267上的5条指令的数据相关

test3.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#该测试测试通过气泡流水线解决数据相关性      test3.asm    （见教材的P267）
#程序运行后，在数码管上依次显示：1、1、2、5、1

and $s1,$s1,$s2              #s1 = s1 ^ s2
sub $s2,$s1,$zero            #s2 = s1 - 0
add $s3,$s1,$s1              #s3=s1 + s1
or $s4,$s5,$s1               #s4=s5 | s1
and $s5,$s6,$s1              #s5=s6 ^ s1

addi $s1,$zero,1             #给s1、s2、s5、s6赋值
addi $s2,$zero,1
addi $s5,$zero,5
addi $s6,$zero,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 ^ 101            =1

add $a0,$0,$s1               #要显示的值放在a0中
addi $v0,$zero,34            #syscall的34号功能（在数码管上显示s1的值）
syscall

add $a0,$0,$s2               #要显示的值放在a0中
addi $v0,$zero,34            #syscall的34号功能（在数码管上显示s2的值）
syscall

add $a0,$0,$s3               #要显示的值放在a0中
addi $v0,$zero,34            #syscall的34号功能（在数码管上显示s3的值）
syscall

add $a0,$0,$s4               #要显示的值放在a0中
addi $v0,$zero,34            #syscall的34号功能（在数码管上显示s4的值）
syscall

add $a0,$0,$s5               #要显示的值放在a0中
addi $v0,$zero,34            #syscall的34号功能（在数码管上显示s5的值）
syscall

addi $v0,$zero,10            #v0=10   设置syscall指令的编号（10号表示停机）
syscall                      #停机
```

求累加和程序

sum_mips.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#MIPS程序 求累加和                    sum_mips.asm
#求累加和：1+2+……+n，n的值为10（可以改变），累加和的结果存放到地址为0的数据存储器中
#该程序执行结束后，数码管显示37H（55）

main:
        addi $s0,$zero,10          # n=10 -> s0
        addi $s1,$zero,1           #      1 -> s1
        addi $s2,$zero,1           #      1 -> s2
        addi $s3,$zero,0           #      0 -> s3
loop:
        add $s3,$s3,$s1            # s3+s1 -> s3
        beq $s1,$s0,finish         # 如果s1=s0，则转finish
        add $s1,$s1,$s2            # s1+s2 -> s1
        j loop
finish:
        sw $s3,0($zero)            # s3 存到地址为0的存储单元中

        add $a0,$0,$s3
        addi $v0,$zero,34          #syscall的34号功能（在数码管上显示a0的值）
        syscall

        addi   $v0,$zero,10        # syscall的10号系统调用
        syscall                    # 程序退出
```

求Fib数程序

fib_mips.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#MIPS程序   计算费波那契数列       fib_mips.asm
# n=10（可以改变），计算好的n+1个数（0、1、1、2、3、5、8、13、21、34、55……）保存在地址为0、4、8、12、...的数据存储器中
#该程序执行后，数码管依次显示：0、1、1、2、3、5、8、D、15、22、37
main:
        addi $s2,$zero,10               # $s2=n=10

        addi $s1,$zero,0                # $s1=0
        sw $s1,0($zero)                 # 0 -> (0)

        addi $s1,$zero,1                # $s1=1
        sw $s1,4($zero)                 # 1 -> (4)

        sw $s1,8($zero)                 # 1 -> (8)

        addi $s3,$zero,2                # $s3 = 2
        addi $s4,$zero,1                # $s4 = 1
        addi $s5,$zero,1                # $s5 = 1
loop:
        add $s6,$s4,$s5                 # $s4+$s5 -> $s6
        addi $s4,$s5,0                  # $s5 -> $s4
        addi $s5,$s6,0                  # $s6 -> $s5
        addi $s3,$s3,1                  # $s3+1 -> $s3
        add $s7,$s3,$s3                 # $s3+$s3 -> $s7
        add $s0,$s7,$s7                 # $s7+$s7 -> $s0
        sw $s6,0($s0)                   # $s6 -> 0($s0)
        beq $s2,$s3,end                 # if $s2 = $s3 goto end
        j loop                          # goto loop
end:


        lw $a0,0($zero)
        addi $v0,$zero,34       #syscall的34号功能（在数码管上显示a0的值      0号单元）
        syscall
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop

        lw $a0,4($zero)
        addi $v0,$zero,34       #syscall的34号功能（在数码管上显示a0的值      4号单元）
        syscall

        lw $a0,40($zero)
        addi $v0,$zero,34       #syscall的34号功能（在数码管上显示a0的值      40号单元）      55=37H
        syscall
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop

        addi   $v0,$zero,10     # syscall的10号系统调用
        syscall                 # 程序退出
```

排序程序

降序

sort1_mips.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#MIPS程序	冒泡法排序（降序排列，从大到小）	sort1_mips.asm
#先将5个数（3、5、2、1、4）存放到地址为0开始的数据存储器中，然后对这5个数进行排序。
#该程序执行后，数码管依次显示5、4、3、2、1

main:
	addi $s0,$zero,3		#第1个数=3（可以修改）保存到(0)
	sw $s0,0($zero)
	addi $s0,$zero,5		#第2个数=5（可以修改）保存到(4)
	sw $s0,4($zero)
	addi $s0,$zero,2		#第3个数=2（可以修改）保存到(8)
	sw $s0,8($zero)
	addi $s0,$zero,1		#第4个数=1（可以修改）保存到(12)
	sw $s0,12($zero)
	addi $s0,$zero,4		#第5个数=4（可以修改）保存到(16)
	sw $s0,16($zero)

	addi $s0,$zero,0		#$s0=0		排序区间开始地址
	addi $s1,$zero,16		#$s1=16=5*4-4	排序区间结束地址
sort_loop:
	lw $s3,0($s0)			#$s3=($s0)
	lw $s4,0($s1)			#$s4=($s1)
	slt $t0,$s3,$s4			#如果$s3<$s4，则置$t0=1；否则，置$t0=0	降序排序	从大到小
	beq $t0,$zero,sort_next		#如果$t0=0，则转sort_nent
	sw $s3,0($s1)			#交换($s0)和($s1)
	sw $s4,0($s0)			#交换($s0)和($s1)
sort_next:
	addi $s1,$s1,-4			# $s1-4 -> $s1
	bne $s0,$s1,sort_loop		#如果$s0不等于$s1，则转sort_loop
	addi $s0,$s0,4			#$s0+4 -> $s0
	addi $s1,$zero,16		#$s1=16=5*4-4	排序区间结束地址
	bne $s0,$s1,sort_loop		#如果$s0不等于$s1，则转sort_loop

	lw $a0,0($zero)
	addi $v0,$zero,34		#syscall的34号功能（在数码管上显示a0的值	0号单元）
	syscall
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
```

```
lw $a0,16($zero)
addi $v0,$zero,34		#syscall的34号功能（在数码管上显示a0的值	16号单元）
syscall
nop
nop
nop
nop
nop
nop
nop
nop
nop

addi	$v0,$zero,10		# syscall的10号系统调用
syscall				# 程序退出
```

排序程序

升序

sort2_mips.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#MIPS程序   冒泡法排序（升序排列，从小到大）        sort2_mips.asm
#先将5个数（3、5、2、1、4）存放到地址为0开始的数据存储器中，然后对这5个数进行排序。
#该程序执行后，数码管依次显示1、2、3、4、5

main:
        addi $s0,$zero,3            #第1个数=3（可以修改）保存到(0)
        sw $s0,0($zero)
        addi $s0,$zero,5            #第2个数=5（可以修改）保存到(4)
        sw $s0,4($zero)
        addi $s0,$zero,2            #第3个数=2（可以修改）保存到(8)
        sw $s0,8($zero)
        addi $s0,$zero,1            #第4个数=1（可以修改）保存到(12)
        sw $s0,12($zero)
        addi $s0,$zero,4            #第5个数=4（可以修改）保存到(16)
        sw $s0,16($zero)

        addi $s0,$zero,0            #$s0=0              排序区间开始地址
        addi $s1,$zero,16           #$s1=16=5*4-4     排序区间结束地址
 sort_loop:
        lw $s3,0($s0)               #$s3=($s0)
        lw $s4,0($s1)               #$s4=($s1)
        slt $t0,$s4,$s3             #如果$s4<$s3，则置$t0=1；否则，置$t0=0     升序排序        从小到大
        beq $t0,$zero,sort_next     #如果$t0=0，则转sort_nent
        sw $s3,0($s1)               #交换($s0)和($s1)
        sw $s4,0($s0)               #交换($s0)和($s1)
sort_next:
        addi $s1,$s1,-4             # $s1-4 -> $s1
        bne $s0,$s1,sort_loop       #如果$s0不等于$s1，则转sort_loop
        addi $s0,$s0,4              #$s0+4 -> $s0
        addi $s1,$zero,16           #$s1=16=5*4-4   排序区间结束地址
        bne $s0,$s1,sort_loop       #如果$s0不等于$s1，则转sort_loop

        lw $a0,0($zero)
        addi $v0,$zero,34           #syscall的34号功能（在数码管上显示a0的值       0号单元）
        syscall
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
```

```
lw $a0,16($zero)
addi $v0,$zero,34           #syscall的34号功能（在数码管上显示a0的值       16号单元）
syscall
nop
nop
nop
nop
nop
nop
nop
nop
nop

addi   $v0,$zero,10         # syscall的10号系统调用
syscall                     # 程序退出
```

- 要求：
  - （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。
  - （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？
  - （3）分析气泡流水线MIPS处理器的数据通路电路。
  - （4）分析气泡流水线MIPS处理器的数据相关检测电路。

3、重定向流水线MIPS处理器

- 气泡流水线通过延缓ID段取操作数动作的方式解决数据冲突问题，但是大量气泡的插入会严重影响指令流水线性能。
- 重定向流水线先不考虑ID段所取的寄存器操作数是否正确，而是等到指令实际使用这些操作数时再考虑正确性问题。
- 通过改造气泡流水线MIPS处理器，增加重定向机制，增加Load-Use数据相关检测机制，使得流水线能在不插入气泡的情况下处理大部分数据相关问题。

重定向流水线的控制信号

```
Stall = LoadUse                          #发生LoadUse相关时，要阻塞（暂停）IF、ID段指令的执行
PC.EN = ~Stall                           #阻塞（暂停）IF段指令的执行（程序计数器PC使能端输入=0）
IF/ID.EN = ~Stall                        #阻塞（暂停）ID段指令的执行（IF/ID流水寄存器使能端输入=0）

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 + LoadUse  #出现分支跳转或LoadUse相关时，清空ID/EX段流水寄存器
```

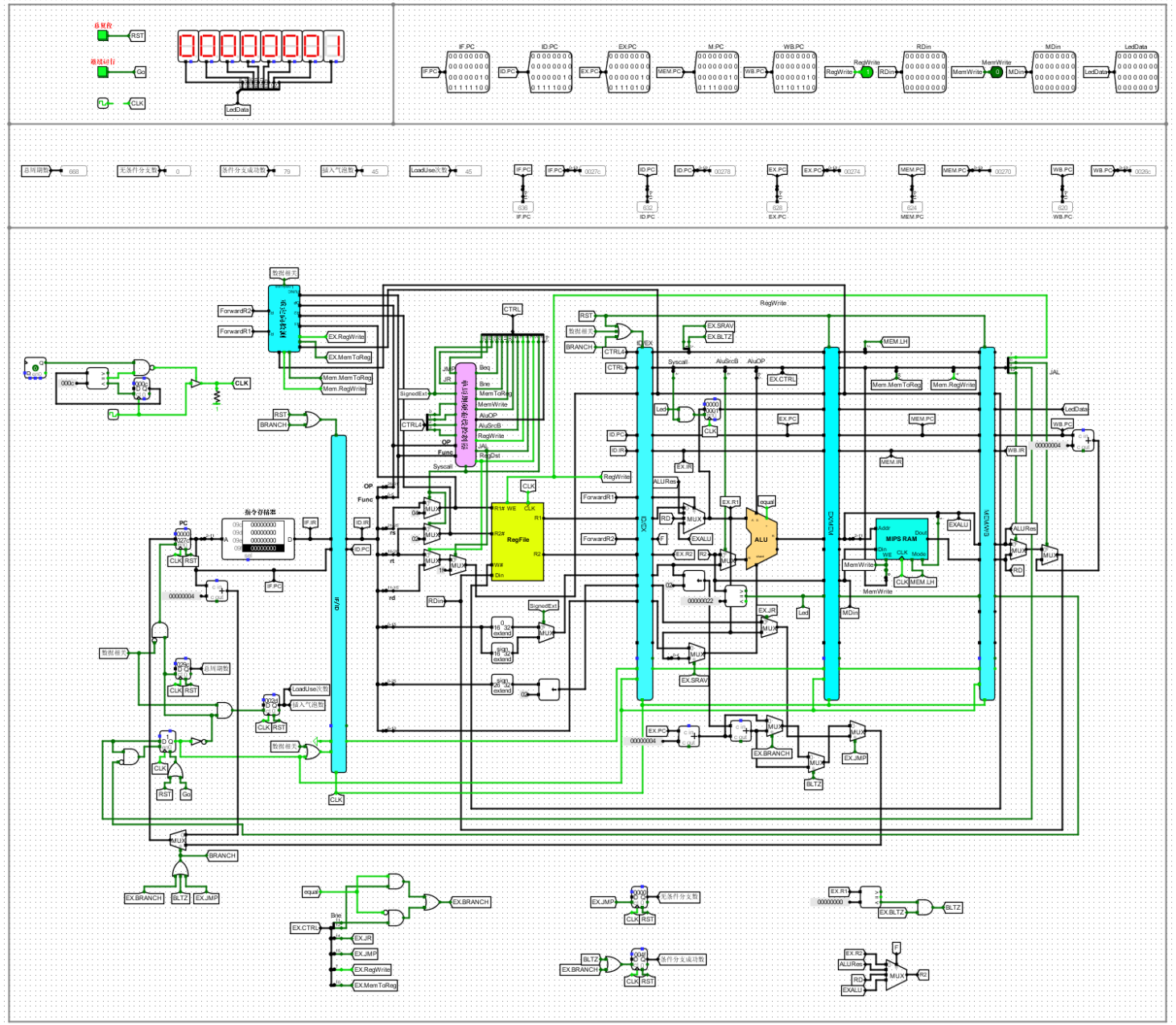
Load-Use数据相关检测与重定向选择信号

```
LoadUse = RsUsed & (rs≠0) & EX.MemRead & (rs==EX.WriteReg#)
        + RtUsed & (rt≠0) & EX.MemRead & (rt==EX.WriteReg#)
```

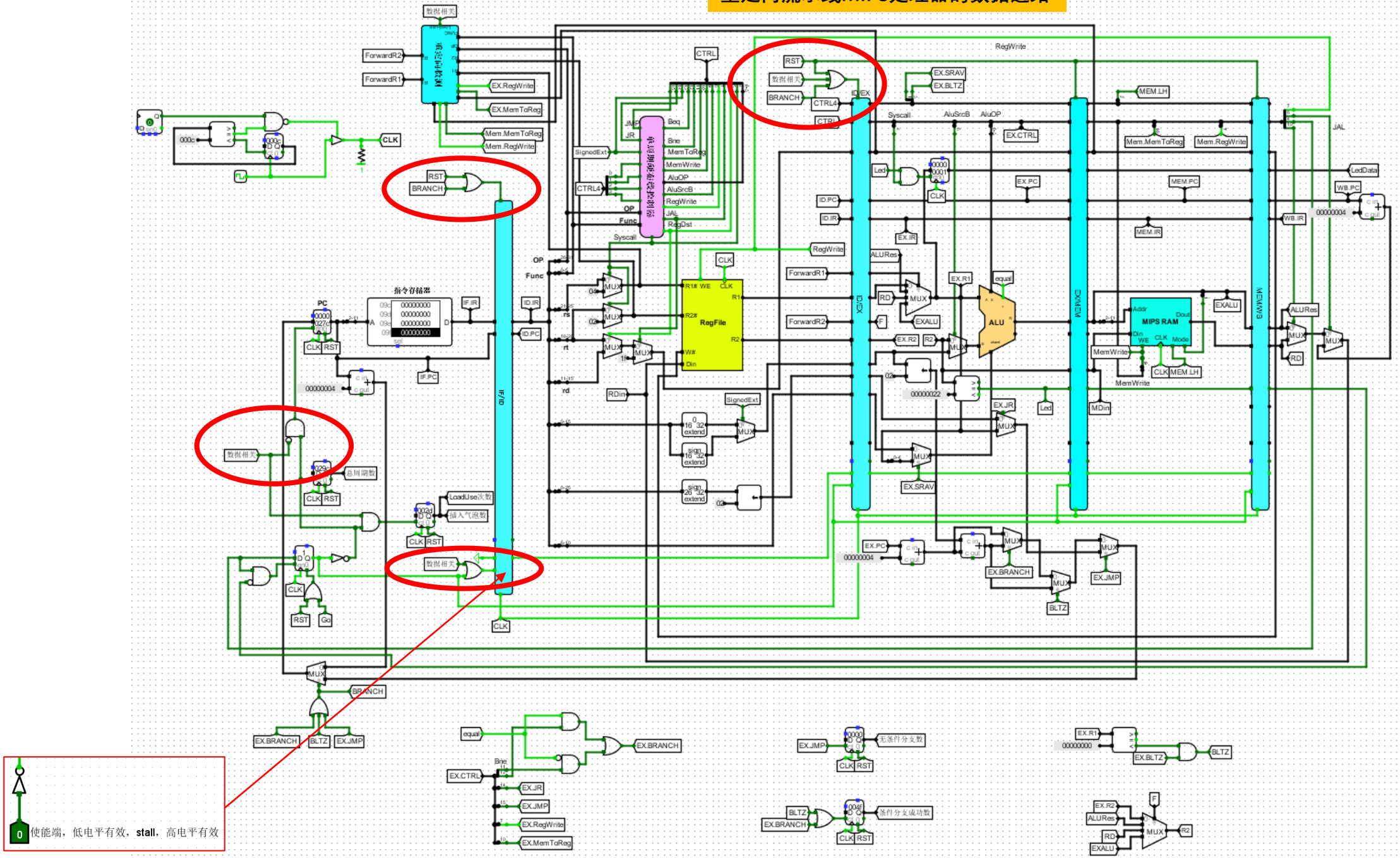
```
if (RsUsed & (rs≠0) & EX.RegWrite & (rs==EX.WriteReg#) )   RsForware=2
else if ( RsUsed & (rs≠0) & MEM. RegWrite & (rs==MEM.WriteReg#)   RsForware=1
else   RsForware=0
```

```
if (RtUsed & (rt≠0) & EX.RegWrite & (rt==EX.WriteReg#) )   RtForware=2
else if ( RtUsed & (rt≠0) & MEM. RegWrite & (rt==MEM.WriteReg#)   RtForware=1
else   RtForware=0
```

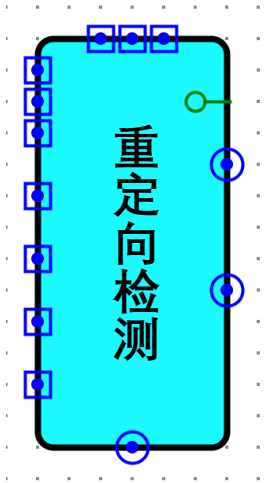
重定向流水线MIPS处理器

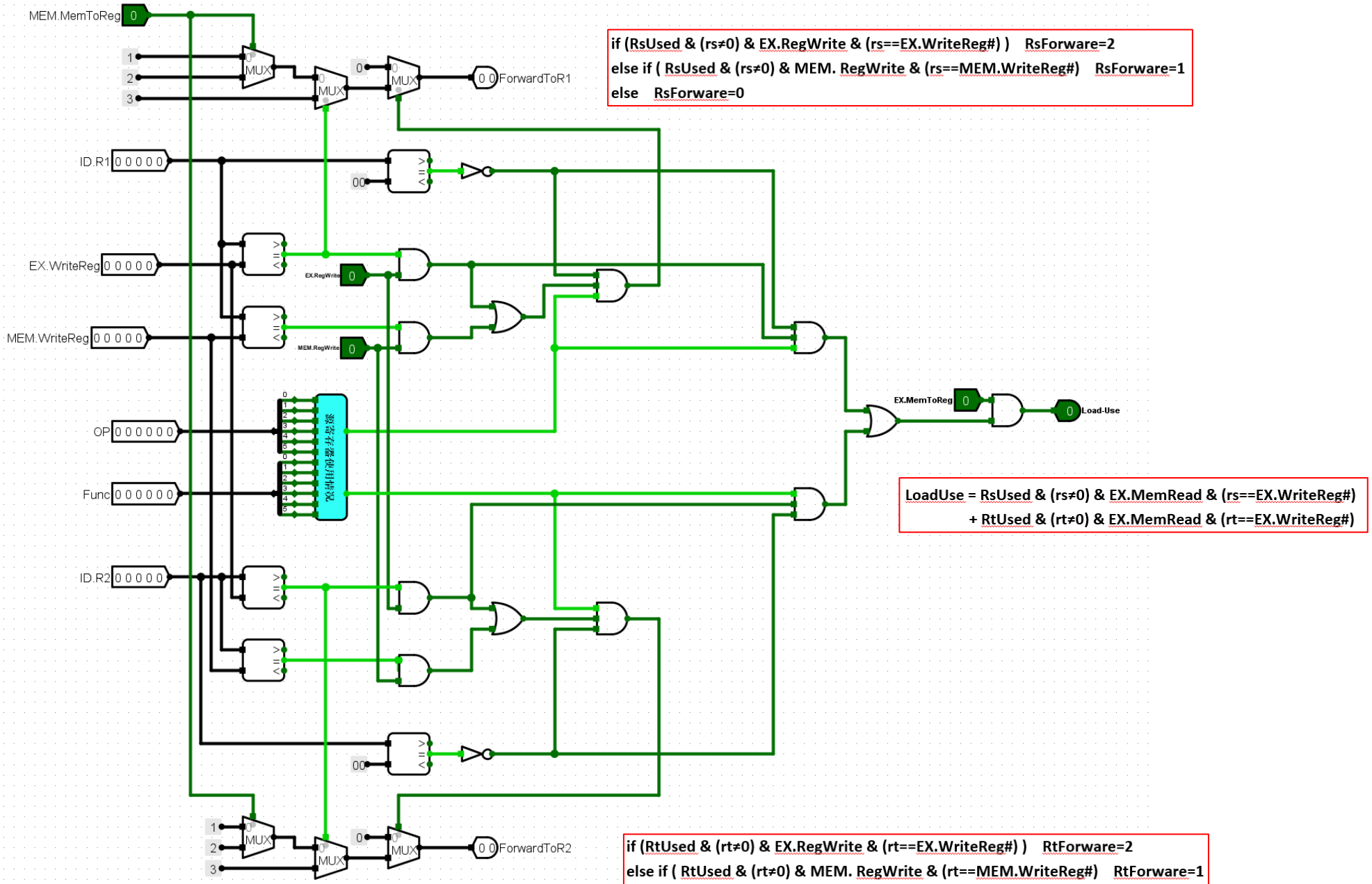


重定向流水线MIPS处理器的数据通路



重定向检测电路


重定向检测


MEM.MemToReg
ID.R1
EX.WriteReg
MEM.WriteReg
OP
Func
ID.R2
MUX
ForwardToR1
ForwardToR2
EX.RegWrite
MEM.RegWrite
寄存器堆内部转发
EX.MemToReg
Load-Use
if (RsUsed & (rs≠0) & EX.RegWrite & (rs==EX.WriteReg#)) RsForware=2
else if (RsUsed & (rs≠0) & MEM. RegWrite & (rs==MEM.WriteReg#) RsForware=1
else RsForware=0
LoadUse = RsUsed & (rs≠0) & EX.MemRead & (rs==EX.WriteReg#)
+ RtUsed & (rt≠0) & EX.MemRead & (rt==EX.WriteReg#)
if (RtUsed & (rt≠0) & EX.RegWrite & (rt==EX.WriteReg#)) RtForware=2
else if (RtUsed & (rt≠0) & MEM. RegWrite & (rt==MEM.WriteReg#) RtForware=1
else RtForware=0

测试程序1
test1.asm

测试5条分支指令的控制相关

test1.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#气泡流水线分支相关测试程序  test1.asm
#该程序测试j、jal、jr、beq、bne等分支指令的相关性，采用气泡流水线后，该程序无分支相关
#程序功能：s1的初值为32，调用子程序后，s1减1，变为31（1FH），然后在数码管上显示，之后再减1，变为30（1EH），在数码管上显示，直到变为0
#该程序执行后，过一段时间，数码管显示1F，之后减1、显示1E，一直到0
        addi $s1,$zero,32
        addi $v0,$zero,34               #34=22H     syscall的34号功能（8个数码管显示8个十六进制数，即32位二进制数）
        j jmp_next1                     #跳转到jmp_next1

        addi $s1,$zero,4                #接下去3条指令不会执行
        addi $s2,$zero,5
        addi $s3,$zero,6

jmp_next1:
        beq $zero,$zero, jmp_next2      #相当于j指令    跳转到jmp_next2

        addi $s1,$zero,7                #接下去3条指令不会执行
        addi $s2,$zero,8
        addi $s3,$zero,9

jmp_next2:
        bne $zero,$s1, jmp_next3        #因为s1=32  不等于0  因此跳转到jmp_next3

        addi $s1,$zero,10               #接下去3条指令不会执行
        addi $s2,$zero,11
        addi $s3,$zero,12

jmp_next3:
        jal jmp_func                    #子程序调用指令

        addi $v0,$zero,10               #syscall的10号功能      子程序返回到这里
        nop
        nop
        nop
        syscall                         #系统退出

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
```

测试程序2
test2.asm

测试数据相关

test2.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#气泡流水线测试程序——数据相关测试     test2.asm
#该程序用于测试数据相关，使用气泡流水线后，该程序无数据相关
#该程序计算：0+1+2+3+4+5+6+7=28=1CH
#该程序执行结束后，数码管显示1C
 addi $s1,$0,4          #s1=4
 sw $s1,0($s1)          # 4 -> (4)
 lw $s2,0($s1)          # (4) -> s2=4

 addi $s2,$s2,-4        # s2-4=0 -> s2=0
 addi $s0,$0,0          # s0=0

 addi $s1,$s0,1         #s1=1
 add $s0,$s0,$s1        #s0=1
 addi $s2,$s2,4         #s2=4
 sw $s0,0($s2)          #1 -> (4)   04H

 addi $s1,$s1,1         #s1=2
 add $s0,$s0,$s1        #s0=1+2=3
 addi $s2,$s2,4         #s2=8
 sw $s0,0($s2)          #3 -> (8)    08H

 addi $s1,$s1,1         #s1=3
 add $s0,$s0,$s1        #s0=1+2+3=6
 addi $s2,$s2,4         #s2=12
 sw $s0,0($s2)          #6 -> (12)   0CH

 addi $s1,$s1,1         #s1=4
 add $s0,$s0,$s1        #s0=1+2+3+4=10
 addi $s2,$s2,4         #s2=16
 sw $s0,0($s2)          #10 -> (16)   10H

 addi $s1,$s1,1         #s1=5
 add $s0,$s0,$s1        #s0=1+2+3+4+5=15
 addi $s2,$s2,4         #s2=20
 sw $s0,0($s2)          #15 -> (20)   14H

 addi $s1,$s1,1         #s1=6
 add $s0,$s0,$s1        #s0=1+2+3+4+5+6=21
 addi $s2,$s2,4         #s2=24
 sw $s0,0($s2)          #21 -> (24)   18H

 addi $s1,$s1,1         #s1=7
 add $s0,$s0,$s1        #s0=1+2+3+4+5+6+7=28
 addi $s2,$s2,4         #s2=28
 sw $s0,0($s2)          #28 -> (28)    1CH

 add $a0,$0,$s0                 #a0=s0     a0送数码管显示
 addi $v0,$zero,34              #syscall的34号功能（在数码管上显示a0的值）
 syscall

 addi $v0,$zero,10              #syscall的10号功能（系统退出）
 addi $s0,$zero,0
 addi $s0,$zero,0
 addi $s0,$zero,0
 syscall
```

测试程序3
test3.asm

测试教材P267上的5条指令的数据相关

test3.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#该测试测试通过气泡流水线解决数据相关性      test3.asm    （见教材的P267）
#程序运行后，在数码管上依次显示：1、1、2、5、1

and $s1,$s1,$s2              #s1 = s1 ^ s2
sub $s2,$s1,$zero            #s2 = s1 - 0
add $s3,$s1,$s1              #s3=s1 + s1
or $s4,$s5,$s1               #s4=s5 | s1
and $s5,$s6,$s1              #s5=s6 ^ s1

addi $s1,$zero,1             #给s1、s2、s5、s6赋值
addi $s2,$zero,1
addi $s5,$zero,5
addi $s6,$zero,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 ^ 101          =1

add $a0,$0,$s1               #要显示的值放在a0中
addi $v0,$zero,34            #syscall的34号功能（在数码管上显示s1的值）
syscall

add $a0,$0,$s2               #要显示的值放在a0中
addi $v0,$zero,34            #syscall的34号功能（在数码管上显示s2的值）
syscall

add $a0,$0,$s3               #要显示的值放在a0中
addi $v0,$zero,34            #syscall的34号功能（在数码管上显示s3的值）
syscall

add $a0,$0,$s4               #要显示的值放在a0中
addi $v0,$zero,34            #syscall的34号功能（在数码管上显示s4的值）
syscall

add $a0,$0,$s5               #要显示的值放在a0中
addi $v0,$zero,34            #syscall的34号功能（在数码管上显示s5的值）
syscall

addi $v0,$zero,10            #v0=10   设置syscall指令的编号（10号表示停机）
syscall                      #停机
```

求累加和程序

sum_mips.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#MIPS程序 求累加和                    sum_mips.asm
#求累加和：1+2+……+n，n的值为10（可以改变），累加和的结果存放到地址为0的数据存储器中
#该程序执行结束后，数码管显示37H（55）

main:
        addi $s0,$zero,10          # n=10 -> s0
        addi $s1,$zero,1           #      1 -> s1
        addi $s2,$zero,1           #      1 -> s2
        addi $s3,$zero,0           #      0 -> s3
loop:
        add $s3,$s3,$s1            # s3+s1 -> s3
        beq $s1,$s0,finish         # 如果s1=s0，则转finish
        add $s1,$s1,$s2            # s1+s2 -> s1
        j loop
finish:
        sw $s3,0($zero)            # s3 存到地址为0的存储单元中

        add $a0,$0,$s3
        addi $v0,$zero,34          #syscall的34号功能（在数码管上显示a0的值）
        syscall

        addi   $v0,$zero,10        # syscall的10号系统调用
        syscall                    # 程序退出
```

求Fib数程序

fib_mips.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#MIPS程序   计算费波那契数列       fib_mips.asm
# n=10（可以改变），计算好的n+1个数（0、1、1、2、3、5、8、13、21、34、55……）保存在地址为0、4、8、12、...的数据存储器中
#该程序执行后，数码管依次显示：0、1、1、2、3、5、8、D、15、22、37
main:
        addi $s2,$zero,10               # $s2=n=10

        addi $s1,$zero,0                # $s1=0
        sw $s1,0($zero)                 # 0 -> (0)

        addi $s1,$zero,1                # $s1=1
        sw $s1,4($zero)                 # 1 -> (4)

        sw $s1,8($zero)                 # 1 -> (8)

        addi $s3,$zero,2                # $s3 = 2
        addi $s4,$zero,1                # $s4 = 1
        addi $s5,$zero,1                # $s5 = 1
loop:
        add $s6,$s4,$s5                 # $s4+$s5 -> $s6
        addi $s4,$s5,0                  # $s5 -> $s4
        addi $s5,$s6,0                  # $s6 -> $s5
        addi $s3,$s3,1                  # $s3+1 -> $s3
        add $s7,$s3,$s3                 # $s3+$s3 -> $s7
        add $s0,$s7,$s7                 # $s7+$s7 -> $s0
        sw $s6,0($s0)                   # $s6 -> 0($s0)
        beq $s2,$s3,end                 # if $s2 = $s3 goto end
        j loop                          # goto loop
end:


        lw $a0,0($zero)
        addi $v0,$zero,34       #syscall的34号功能（在数码管上显示a0的值      0号单元）
        syscall
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
                                                        lw $a0,40($zero)
                                                        addi $v0,$zero,34       #syscall的34号功能（在数码管上显示a0的值      40号单元）      55=37H
                                                        syscall
                                                        nop
                                                        nop
                                                        nop
                                                        nop
                                                        nop
                                                        nop
                                                        nop
                                                        nop
                                                        nop
                                                        addi    $v0,$zero,10    # syscall的10号系统调用
                                                        syscall                 # 程序退出

        lw $a0,4($zero)
        addi $v0,$zero,34       #syscall的34号功能（在数码管上显示a0的值      4号单元）
        syscall
```

排序程序

降序

sort1_mips.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#MIPS程序   冒泡法排序（降序排列，从大到小）        sort1_mips.asm
#先将5个数（3、5、2、1、4）存放到地址为0开始的数据存储器中，然后对这5个数进行排序。
#该程序执行后，数码管依次显示5、4、3、2、1

main:
        addi $s0,$zero,3            #第1个数=3（可以修改）保存到(0)
        sw $s0,0($zero)
        addi $s0,$zero,5            #第2个数=5（可以修改）保存到(4)
        sw $s0,4($zero)
        addi $s0,$zero,2            #第3个数=2（可以修改）保存到(8)
        sw $s0,8($zero)
        addi $s0,$zero,1            #第4个数=1（可以修改）保存到(12)
        sw $s0,12($zero)
        addi $s0,$zero,4            #第5个数=4（可以修改）保存到(16)
        sw $s0,16($zero)

        addi $s0,$zero,0            #$s0=0              排序区间开始地址
        addi $s1,$zero,16           #$s1=16=5*4-4       排序区间结束地址
sort_loop:
        lw $s3,0($s0)               #$s3=($s0)
        lw $s4,0($s1)               #$s4=($s1)
        slt $t0,$s3,$s4             #如果$s3<$s4，则置$t0=1；否则，置$t0=0      降序排序      从大到小
        beq $t0,$zero,sort_next     #如果$t0=0，则转sort_nent
        sw $s3,0($s1)               #交换($s0)和($s1)
        sw $s4,0($s0)               #交换($s0)和($s1)
sort_next:
        addi $s1,$s1,-4             # $s1-4 -> $s1
        bne $s0,$s1,sort_loop       #如果$s0不等于$s1，则转sort_loop
        addi $s0,$s0,4              #$s0+4 -> $s0
        addi $s1,$zero,16           #$s1=16=5*4-4  排序区间结束地址
        bne $s0,$s1,sort_loop       #如果$s0不等于$s1，则转sort_loop

        lw $a0,0($zero)
        addi $v0,$zero,34           #syscall的34号功能（在数码管上显示a0的值      0号单元）
        syscall
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        lw $a0,16($zero)
        addi $v0,$zero,34           #syscall的34号功能（在数码管上显示a0的值      16号单元）
        syscall
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop

        addi   $v0,$zero,10         # syscall的10号系统调用
        syscall                     # 程序退出
```

排序程序

升序

sort2_mips.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#MIPS程序　冒泡法排序（升序排列，从小到大）　　sort2_mips.asm
#先将5个数（3、5、2、1、4）存放到地址为0开始的数据存储器中，然后对这5个数进行排序。
#该程序执行后，数码管依次显示1、2、3、4、5

main:
        addi $s0,$zero,3           #第1个数=3（可以修改）保存到(0)
        sw $s0,0($zero)
        addi $s0,$zero,5           #第2个数=5（可以修改）保存到(4)
        sw $s0,4($zero)
        addi $s0,$zero,2           #第3个数=2（可以修改）保存到(8)
        sw $s0,8($zero)
        addi $s0,$zero,1           #第4个数=1（可以修改）保存到(12)
        sw $s0,12($zero)
        addi $s0,$zero,4           #第5个数=4（可以修改）保存到(16)
        sw $s0,16($zero)

        addi $s0,$zero,0           #$s0=0           排序区间开始地址
        addi $s1,$zero,16          #$s1=16=5*4-4    排序区间结束地址
 sort_loop:
        lw $s3,0($s0)              #$s3=($s0)
        lw $s4,0($s1)              #$s4=($s1)
        slt $t0,$s4,$s3            #如果$s4<$s3，则置$t0=1；否则，置$t0=0    升序排序    从小到大
        beq $t0,$zero,sort_next    #如果$t0=0，则转sort_nent
        sw $s3,0($s1)              #交换($s0)和($s1)
        sw $s4,0($s0)              #交换($s0)和($s1)
sort_next:
        addi $s1,$s1,-4            # $s1-4 -> $s1
        bne $s0,$s1,sort_loop      #如果$s0不等于$s1，则转sort_loop
        addi $s0,$s0,4             #$s0+4 -> $s0
        addi $s1,$zero,16          #$s1=16=5*4-4  排序区间结束地址
        bne $s0,$s1,sort_loop      #如果$s0不等于$s1，则转sort_loop

        lw $a0,0($zero)
        addi $v0,$zero,34          #syscall的34号功能（在数码管上显示a0的值    0号单元）
        syscall
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
```

```
lw $a0,16($zero)
addi $v0,$zero,34          #syscall的34号功能（在数码管上显示a0的值    16号单元）
syscall
nop
nop
nop
nop
nop
nop
nop
nop
nop

addi   $v0,$zero,10        # syscall的10号系统调用
syscall                    # 程序退出
```

- 要求：
  - （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？
  - （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？
  - （3）分析重定向流水线MIPS处理器的数据通路电路。
  - （4）分析重定向流水线MIPS处理器的重定向检测电路。

4、动态分支预测流水线MIPS处理器

- 采用重定向机制后，只有少数Load-Use相关才需要插入气泡，此时流水线中的控制冲突对性能影响最大。
- 动态分支预测技术依据分支指令的分支跳转历史，不断地动态调整预测策略，提升预测准确率，在ID段提前预取正确的指令，以避免分支指令后续指令被清空，提升流水线性能。
- 通过改造重定向流水线MIPS处理器，增加动态分支预测逻辑，从而优化流水线性能。

动态分支预测流水线的控制信号

```
Stall = LoadUse                          #发生LoadUse相关时，要阻塞（暂停）IF、ID段指令的执行
PC.EN = ~Stall                           #阻塞（暂停）IF段指令的执行（程序计数器PC使能端输入=0）
IF/ID.EN = ~Stall                        #阻塞（暂停）ID段指令的执行（IF/ID流水寄存器使能端输入=0）

IF/ID.CLR = PredictErr                   #预测失败时，清空IF/ID段流水寄存器（ID段）
ID/EX.CLR = Flush = PredictErr + LoadUse #预测失败或LoadUse相关时，清空ID/EX段流水寄存器（EX段）
```

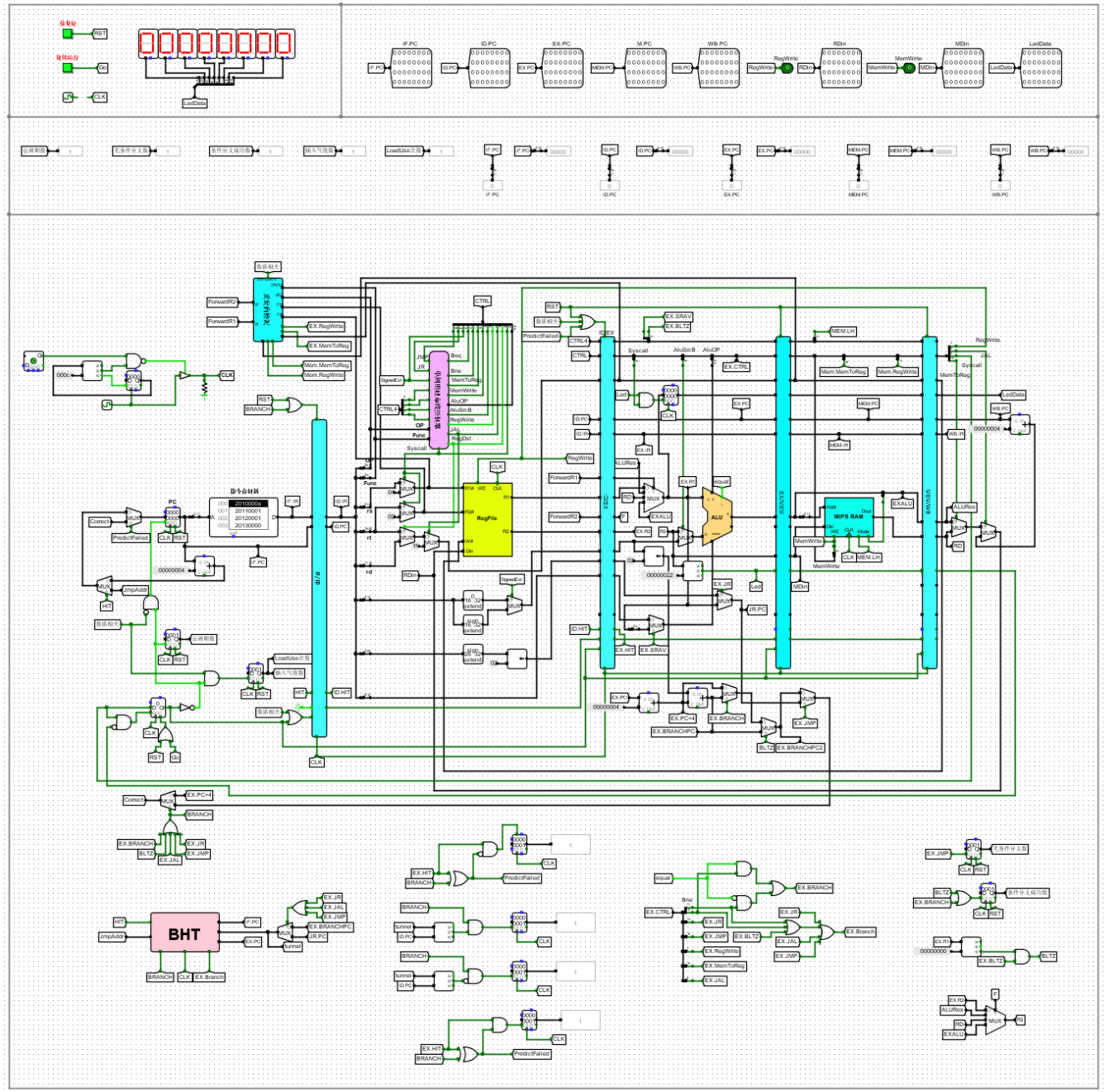
Load-Use数据相关检测与重定向选择信号

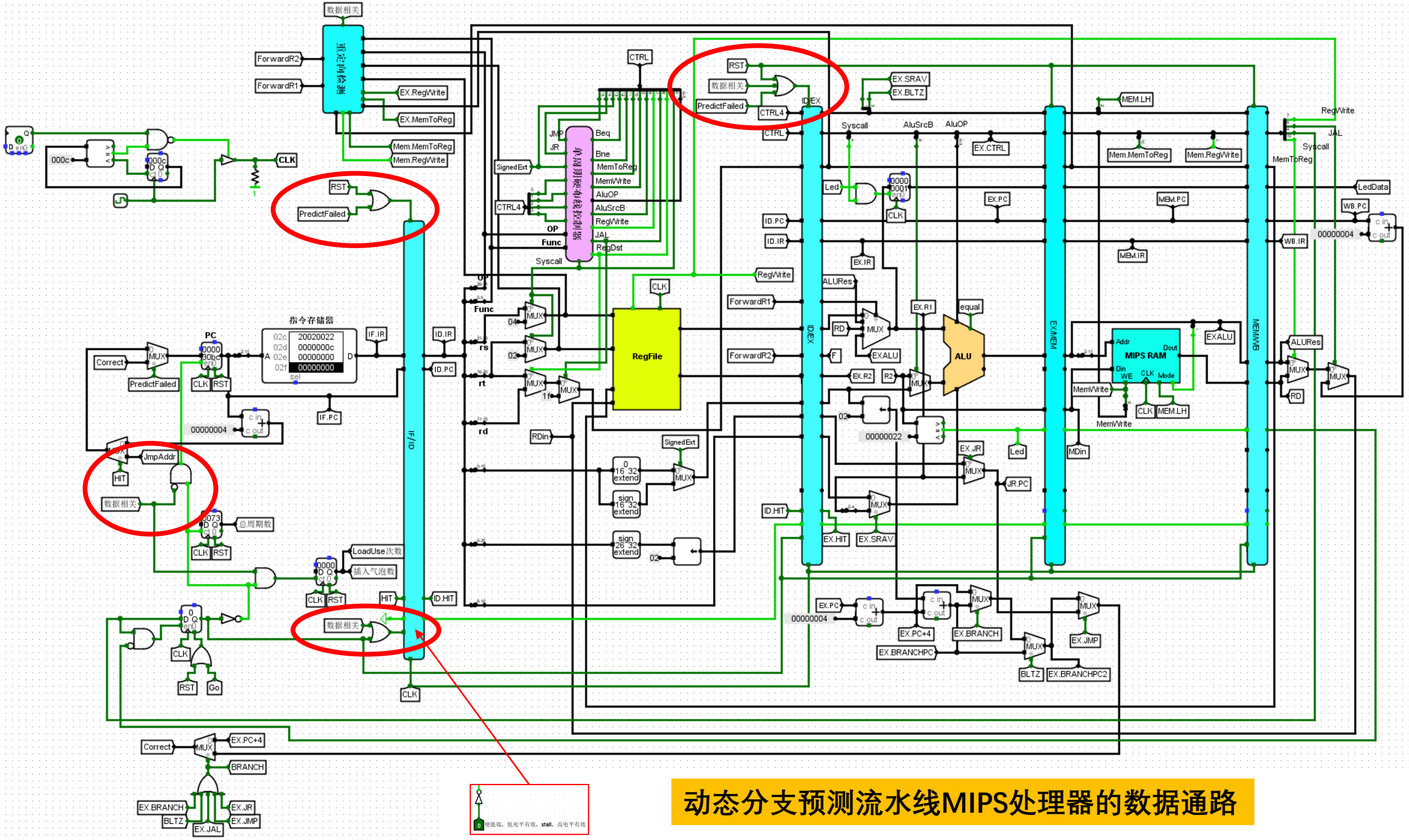
```
LoadUse = RsUsed & (rs≠0) & EX.MemRead & (rs==EX.WriteReg#)
        + RtUsed & (rt≠0) & EX.MemRead & (rt==EX.WriteReg#)
```

```
if (RsUsed & (rs≠0) & EX.RegWrite & (rs==EX.WriteReg#) )   RsForware=2
else if ( RsUsed & (rs≠0) & MEM. RegWrite & (rs==MEM.WriteReg#)   RsForware=1
else   RsForware=0
```

```
if (RtUsed & (rt≠0) & EX.RegWrite & (rt==EX.WriteReg#) )   RtForware=2
else if ( RtUsed & (rt≠0) & MEM. RegWrite & (rt==MEM.WriteReg#)   RtForware=1
else   RtForware=0
```

动态分支预测流水线MIPS处理器

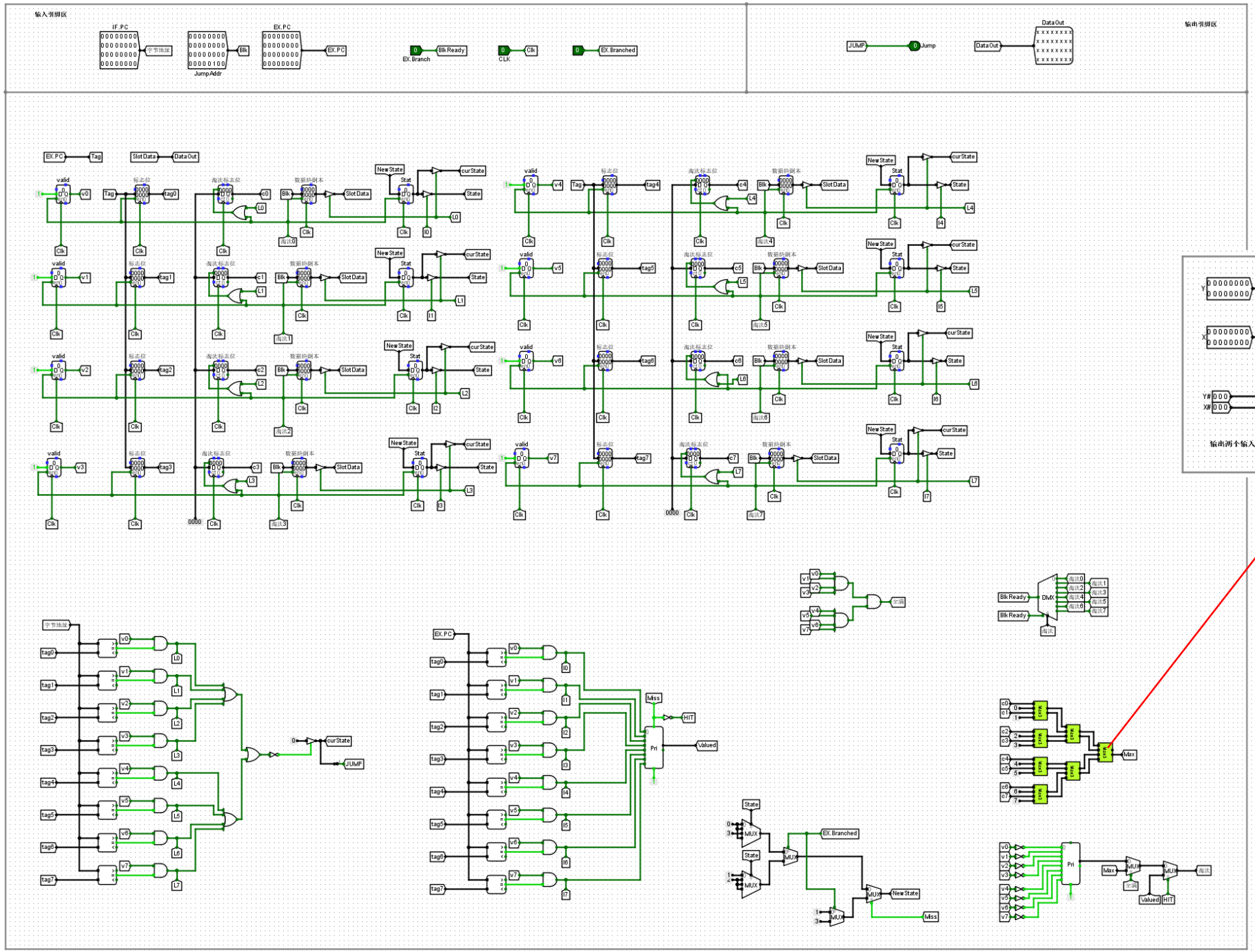




动态分支预测流水线MIPS处理器的数据通路

BHT

分支历史表


输入引脚区
IF.PC
字节地址
Blk
EX.PC
Jump.Addr
EX.Branch
Blk Ready
CLK
Clk
EX.Branched
输出引脚区
JUMP
Jump
Data Out
valid
Tag
标志位
海沃标志位
数据块副本
Slot Data
New State
Stat
curState
State
JUMP
Miss
HIT
Valued
Pri
DMX
MUX
Max
Valued
输出两个输入中的较大值，并输出较大值的3位编号

测试程序1
test1.asm

测试5条分支指令的控制相关

test1.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#气泡流水线分支相关测试程序  test1.asm
#该程序测试j、jal、jr、beq、bne等分支指令的相关性，采用气泡流水线后，该程序无分支相关
#程序功能：s1的初值为32，调用子程序后，s1减1，变为31（1FH），然后在数码管上显示，之后再减1，变为30（1EH），在数码管上显示，直到变为0
#该程序执行后，过一段时间，数码管显示1F，之后减1、显示1E，一直到0

        addi $s1,$zero,32
        addi $v0,$zero,34               #34=22H     syscall的34号功能（8个数码管显示8个十六进制数，即32位二进制数）
        j jmp_next1                     #跳转到jmp_next1

        addi $s1,$zero,4                #接下去3条指令不会执行
        addi $s2,$zero,5
        addi $s3,$zero,6

jmp_next1:
        beq $zero,$zero, jmp_next2      #相当于j指令    跳转到jmp_next2

        addi $s1,$zero,7                #接下去3条指令不会执行
        addi $s2,$zero,8
        addi $s3,$zero,9

jmp_next2:
        bne $zero,$s1, jmp_next3        #因为s1=32  不等于0  因此跳转到jmp_next3

        addi $s1,$zero,10               #接下去3条指令不会执行
        addi $s2,$zero,11
        addi $s3,$zero,12

jmp_next3:
        jal jmp_func                    #子程序调用指令

        addi $v0,$zero,10               #syscall的10号功能      子程序返回到这里
        nop
        nop
        nop
        syscall                         #系统退出

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
```

测试程序2
test2.asm

测试数据相关

test2.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#气泡流水线测试程序——数据相关测试    test2.asm
#该程序用于测试数据相关，使用气泡流水线后，该程序无数据相关
#该程序计算：0+1+2+3+4+5+6+7=28=1CH
#该程序执行结束后，数码管显示1C
  addi $s1,$0,4          #s1=4
  sw $s1,0($s1)          # 4 -> (4)
  lw $s2,0($s1)          # (4) -> s2=4

  addi $s2,$s2,-4        # s2-4=0 -> s2=0
  addi $s0,$0,0          # s0=0

  addi $s1,$s0,1         #s1=1
  add $s0,$s0,$s1        #s0=1
  addi $s2,$s2,4         #s2=4
  sw $s0,0($s2)          #1 -> (4)   04H

  addi $s1,$s1,1         #s1=2
  add $s0,$s0,$s1        #s0=1+2=3
  addi $s2,$s2,4         #s2=8
  sw $s0,0($s2)          #3 -> (8)    08H

  addi $s1,$s1,1         #s1=3
  add $s0,$s0,$s1        #s0=1+2+3=6
  addi $s2,$s2,4         #s2=12
  sw $s0,0($s2)          #6 -> (12)   0CH

  addi $s1,$s1,1         #s1=4
  add $s0,$s0,$s1        #s0=1+2+3+4=10
  addi $s2,$s2,4         #s2=16
  sw $s0,0($s2)          #10 -> (16)   10H

  addi $s1,$s1,1         #s1=5
  add $s0,$s0,$s1        #s0=1+2+3+4+5=15
  addi $s2,$s2,4         #s2=20
  sw $s0,0($s2)          #15 -> (20)   14H

  addi $s1,$s1,1         #s1=6
  add $s0,$s0,$s1        #s0=1+2+3+4+5+6=21
  addi $s2,$s2,4         #s2=24
  sw $s0,0($s2)          #21 -> (24)   18H

  addi $s1,$s1,1         #s1=7
  add $s0,$s0,$s1        #s0=1+2+3+4+5+6+7=28
  addi $s2,$s2,4         #s2=28
  sw $s0,0($s2)          #28 -> (28)    1CH

  add $a0,$0,$s0                #a0=s0     a0送数码管显示
  addi $v0,$zero,34             #syscall的34号功能（在数码管上显示a0的值）
  syscall

  addi $v0,$zero,10             #syscall的10号功能（系统退出）
  addi $s0,$zero,0
  addi $s0,$zero,0
  addi $s0,$zero,0
  syscall
```

测试程序3
test3.asm

测试教材P267上的5条指令的数据相关

test3.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#该测试测试通过气泡流水线解决数据相关性      test3.asm    （见教材的P267）
#程序运行后，在数码管上依次显示：1、1、2、5、1

and $s1,$s1,$s2              #s1 = s1 ^ s2
sub $s2,$s1,$zero            #s2 = s1 - 0
add $s3,$s1,$s1              #s3=s1 + s1
or $s4,$s5,$s1               #s4=s5 | s1
and $s5,$s6,$s1              #s5=s6 ^ s1

addi $s1,$zero,1             #给s1、s2、s5、s6赋值
addi $s2,$zero,1
addi $s5,$zero,5
addi $s6,$zero,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 ^ 101          =1

add $a0,$0,$s1               #要显示的值放在a0中
addi $v0,$zero,34            #syscall的34号功能（在数码管上显示s1的值）
syscall

add $a0,$0,$s2               #要显示的值放在a0中
addi $v0,$zero,34            #syscall的34号功能（在数码管上显示s2的值）
syscall

add $a0,$0,$s3               #要显示的值放在a0中
addi $v0,$zero,34            #syscall的34号功能（在数码管上显示s3的值）
syscall

add $a0,$0,$s4               #要显示的值放在a0中
addi $v0,$zero,34            #syscall的34号功能（在数码管上显示s4的值）
syscall

add $a0,$0,$s5               #要显示的值放在a0中
addi $v0,$zero,34            #syscall的34号功能（在数码管上显示s5的值）
syscall

addi $v0,$zero,10            #v0=10   设置syscall指令的编号（10号表示停机）
syscall                      #停机
```

测试程序4
test4.asm

test4.asm - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#测试程序    test4.asm
#程序执行完后，数码管显示 1

.text
        addi $s1,$zero,5            #设置循环次数=5

        j jmp_next1                 #载入BPB（分支预测表），BPB有1个表项

        addi $s1,$zero,1            #以下2条指令不执行
        addi $s2,$zero,2

jmp_next1:
        j jmp_next2                 #载入BPB（分支预测表），BPB有2个表项

        addi $s1,$zero,1            #以下2条指令不执行
        addi $s2,$zero,2

jmp_next2:
        j jmp_next3                 #载入BPB（分支预测表），BPB有3个表项

        addi $s1,$zero,1            #以下2条指令不执行
        addi $s2,$zero,2

jmp_next3:
        j jmp_next4                 #载入BPB（分支预测表），BPB有4个表项

        addi $s1,$zero,1            #以下2条指令不执行
        addi $s2,$zero,2

jmp_next4:
        j jmp_next5                 #载入BPB（分支预测表），BPB有5个表项

        addi $s1,$zero,1            #以下2条指令不执行
        addi $s2,$zero,2

jmp_next5:
        j jmp_next6                 #载入BPB（分支预测表），BPB有6个表项

        addi $s1,$zero,1            #以下2条指令不执行
        addi $s2,$zero,2

jmp_next6:
        j jmp_next7                 #载入BPB（分支预测表），BPB有7个表项

        addi $s1,$zero,1            #以下2条指令不执行
        addi $s2,$zero,2

jmp_next7:
        j jmp_next8                 #载入BPB（分支预测表），BPB有8个表项

        addi $s1,$zero,1            #以下2条指令不执行
        addi $s2,$zero,2

jmp_next8:
        addi $s1,$s1,-1
        bne $s1,$zero,jmp_next2     #s1的初值为5，因此需要跳转5次

        addi $s0,$zero,1            #赋值  s0=1    s1=1    s2=255    s3=3
        addi $s2,$zero,255
        addi $s1,$zero,1
        addi $s3,$zero,3

        beq $s0,$s2,jmp_next9       #s0 不等于 s2，不转，执行下一条指令
        beq $s0,$s0,jmp_next9       #条件永远成立，跳转

        addi $s1,$zero,1            #以下2条指令不执行
        addi $s2,$zero,2

jmp_next9:
        bne $s1,$s1,jmp_next10      #条件永远不成立，执行下一条指令
        bne $s1,$s2,jmp_next10      #s1 不等于 s2 ，跳转

        addi $s1,$zero,1            #以下2条指令不执行
        addi $s2,$zero,2

jmp_next10:
        jal func                    #子程序调用指令

        addi $v0,$zero,10           #子程序返回到这里
        syscall

func:
        addi $s0,$zero,0            #s0=0
        addi $s0,$s0,1              #s0=1
        add $a0,$zero,$s0           #a0=s0=1

        addi $v0,$zero,34           #显示
        syscall

        jr $ra
```

求累加和程序

sum_mips.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#MIPS程序 求累加和                sum_mips.asm
#求累加和：1+2+……+n，n的值为10（可以改变），累加和的结果存放到地址为0的数据存储器中
#该程序执行结束后，数码管显示37H（55）

main:
        addi $s0,$zero,10       # n=10 -> s0
        addi $s1,$zero,1        #      1 -> s1
        addi $s2,$zero,1        #      1 -> s2
        addi $s3,$zero,0        #      0 -> s3
loop:
        add $s3,$s3,$s1         # s3+s1 -> s3
        beq $s1,$s0,finish      # 如果s1=s0，则转finish
        add $s1,$s1,$s2         # s1+s2 -> s1
        j loop
finish:
        sw $s3,0($zero)         # s3 存到地址为0的存储单元中

        add $a0,$0,$s3
        addi $v0,$zero,34       #syscall的34号功能（在数码管上显示a0的值）
        syscall

        addi   $v0,$zero,10     # syscall的10号系统调用
        syscall                 # 程序退出
```

求Fib数程序

fib_mips.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#MIPS程序   计算费波那契数列       fib_mips.asm
# n=10（可以改变），计算好的n+1个数（0、1、1、2、3、5、8、13、21、34、55……）保存在地址为0、4、8、12、...的数据存储器中
#该程序执行后，数码管依次显示：0、1、1、2、3、5、8、D、15、22、37
main:
        addi $s2,$zero,10               # $s2=n=10

        addi $s1,$zero,0                # $s1=0
        sw $s1,0($zero)                 # 0 -> (0)

        addi $s1,$zero,1                # $s1=1
        sw $s1,4($zero)                 # 1 -> (4)

        sw $s1,8($zero)                 # 1 -> (8)

        addi $s3,$zero,2                # $s3 = 2
        addi $s4,$zero,1                # $s4 = 1
        addi $s5,$zero,1                # $s5 = 1
loop:
        add $s6,$s4,$s5                 # $s4+$s5 -> $s6
        addi $s4,$s5,0                  # $s5 -> $s4
        addi $s5,$s6,0                  # $s6 -> $s5
        addi $s3,$s3,1                  # $s3+1 -> $s3
        add $s7,$s3,$s3                 # $s3+$s3 -> $s7
        add $s0,$s7,$s7                 # $s7+$s7 -> $s0
        sw $s6,0($s0)                   # $s6 -> 0($s0)
        beq $s2,$s3,end                 # if $s2 = $s3 goto end
        j loop                          # goto loop
end:


        lw $a0,0($zero)
        addi $v0,$zero,34       #syscall的34号功能（在数码管上显示a0的值      0号单元）
        syscall
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop


        lw $a0,4($zero)
        addi $v0,$zero,34       #syscall的34号功能（在数码管上显示a0的值      4号单元）
        syscall
                                                lw $a0,40($zero)
                                                addi $v0,$zero,34       #syscall的34号功能（在数码管上显示a0的值      40号单元）      55=37H
                                                syscall
                                                nop
                                                nop
                                                nop
                                                nop
                                                nop
                                                nop
                                                nop
                                                nop


                                                addi   $v0,$zero,10     # syscall的10号系统调用
                                                syscall                 # 程序退出
```

排序程序

降序

sort1_mips.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#MIPS程序   冒泡法排序（降序排列，从大到小）      sort1_mips.asm
#先将5个数（3、5、2、1、4）存放到地址为0开始的数据存储器中，然后对这5个数进行排序。
#该程序执行后，数码管依次显示5、4、3、2、1

main:
        addi $s0,$zero,3            #第1个数=3（可以修改）保存到(0)
        sw $s0,0($zero)
        addi $s0,$zero,5            #第2个数=5（可以修改）保存到(4)
        sw $s0,4($zero)
        addi $s0,$zero,2            #第3个数=2（可以修改）保存到(8)
        sw $s0,8($zero)
        addi $s0,$zero,1            #第4个数=1（可以修改）保存到(12)
        sw $s0,12($zero)
        addi $s0,$zero,4            #第5个数=4（可以修改）保存到(16)
        sw $s0,16($zero)

        addi $s0,$zero,0            #$s0=0           排序区间开始地址
        addi $s1,$zero,16           #$s1=16=5*4-4    排序区间结束地址
sort_loop:
        lw $s3,0($s0)               #$s3=($s0)
        lw $s4,0($s1)               #$s4=($s1)
        slt $t0,$s3,$s4             #如果$s3<$s4，则置$t0=1；否则，置$t0=0      降序排序      从大到小
        beq $t0,$zero,sort_next     #如果$t0=0，则转sort_nent
        sw $s3,0($s1)               #交换($s0)和($s1)
        sw $s4,0($s0)               #交换($s0)和($s1)
sort_next:
        addi $s1,$s1,-4             # $s1-4 -> $s1
        bne $s0,$s1,sort_loop       #如果$s0不等于$s1，则转sort_loop
        addi $s0,$s0,4              #$s0+4 -> $s0
        addi $s1,$zero,16           #$s1=16=5*4-4  排序区间结束地址
        bne $s0,$s1,sort_loop       #如果$s0不等于$s1，则转sort_loop

        lw $a0,0($zero)
        addi $v0,$zero,34           #syscall的34号功能（在数码管上显示a0的值      0号单元）
        syscall
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
```

```
lw $a0,16($zero)
addi $v0,$zero,34          #syscall的34号功能（在数码管上显示a0的值      16号单元）
syscall
nop
nop
nop
nop
nop
nop
nop
nop
nop

addi   $v0,$zero,10        # syscall的10号系统调用
syscall                    # 程序退出
```

排序程序

升序

sort2_mips.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#MIPS程序   冒泡法排序（升序排列，从小到大）        sort2_mips.asm
#先将5个数（3、5、2、1、4）存放到地址为0开始的数据存储器中，然后对这5个数进行排序。
#该程序执行后，数码管依次显示1、2、3、4、5

main:
        addi $s0,$zero,3            #第1个数=3（可以修改）保存到(0)
        sw $s0,0($zero)
        addi $s0,$zero,5            #第2个数=5（可以修改）保存到(4)
        sw $s0,4($zero)
        addi $s0,$zero,2            #第3个数=2（可以修改）保存到(8)
        sw $s0,8($zero)
        addi $s0,$zero,1            #第4个数=1（可以修改）保存到(12)
        sw $s0,12($zero)
        addi $s0,$zero,4            #第5个数=4（可以修改）保存到(16)
        sw $s0,16($zero)

        addi $s0,$zero,0            #$s0=0             排序区间开始地址
        addi $s1,$zero,16           #$s1=16=5*4-4      排序区间结束地址
 sort_loop:
        lw $s3,0($s0)               #$s3=($s0)
        lw $s4,0($s1)               #$s4=($s1)
        slt $t0,$s4,$s3             #如果$s4<$s3，则置$t0=1；否则，置$t0=0     升序排序     从小到大
        beq $t0,$zero,sort_next     #如果$t0=0，则转sort_nent
        sw $s3,0($s1)               #交换($s0)和($s1)
        sw $s4,0($s0)               #交换($s0)和($s1)
sort_next:
        addi $s1,$s1,-4             # $s1-4 -> $s1
        bne $s0,$s1,sort_loop       #如果$s0不等于$s1，则转sort_loop
        addi $s0,$s0,4              #$s0+4 -> $s0
        addi $s1,$zero,16           #$s1=16=5*4-4   排序区间结束地址
        bne $s0,$s1,sort_loop       #如果$s0不等于$s1，则转sort_loop

        lw $a0,0($zero)
        addi $v0,$zero,34           #syscall的34号功能（在数码管上显示a0的值     0号单元）
        syscall
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
```

```
lw $a0,16($zero)
addi $v0,$zero,34           #syscall的34号功能（在数码管上显示a0的值     16号单元）
syscall
nop
nop
nop
nop
nop
nop
nop
nop
nop

addi   $v0,$zero,10         # syscall的10号系统调用
syscall                     # 程序退出
```

- 要求：
  - （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？

  - （2）分析动态分支预测流水线MIPS处理器的数据通路电路。

  - （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。

实验报告文件提交要求

- 1、请按照FTP上的实验报告模板（厦门大学计算机组成原理实验报告样本（第6次实验）.docx），撰写实验报告（Word版本），实验报告命名为：学号+姓名+第6次实验报告。

- 2、将实验报告（Word版本）上传到FTP上，第6次实验报告提交截止时间（2周内）：2023年6月14日晚上24点。

Thanks