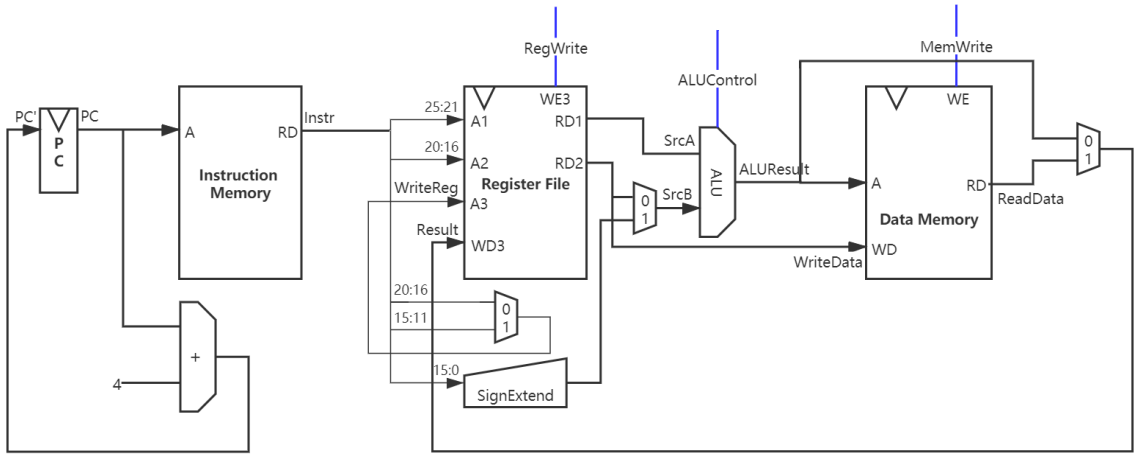
17300180070 马逸君

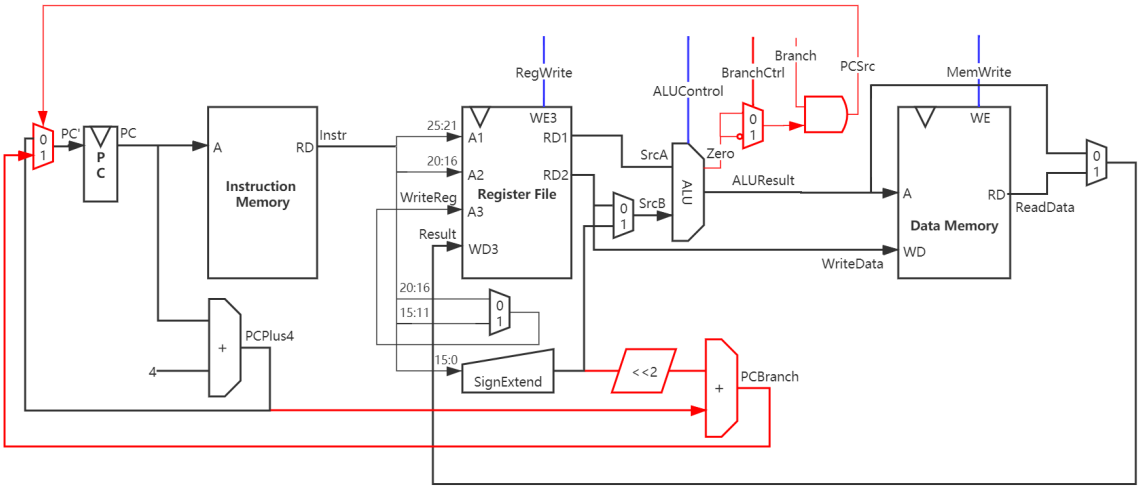
MIPS单周期处理器 实验报告

一、设计原理

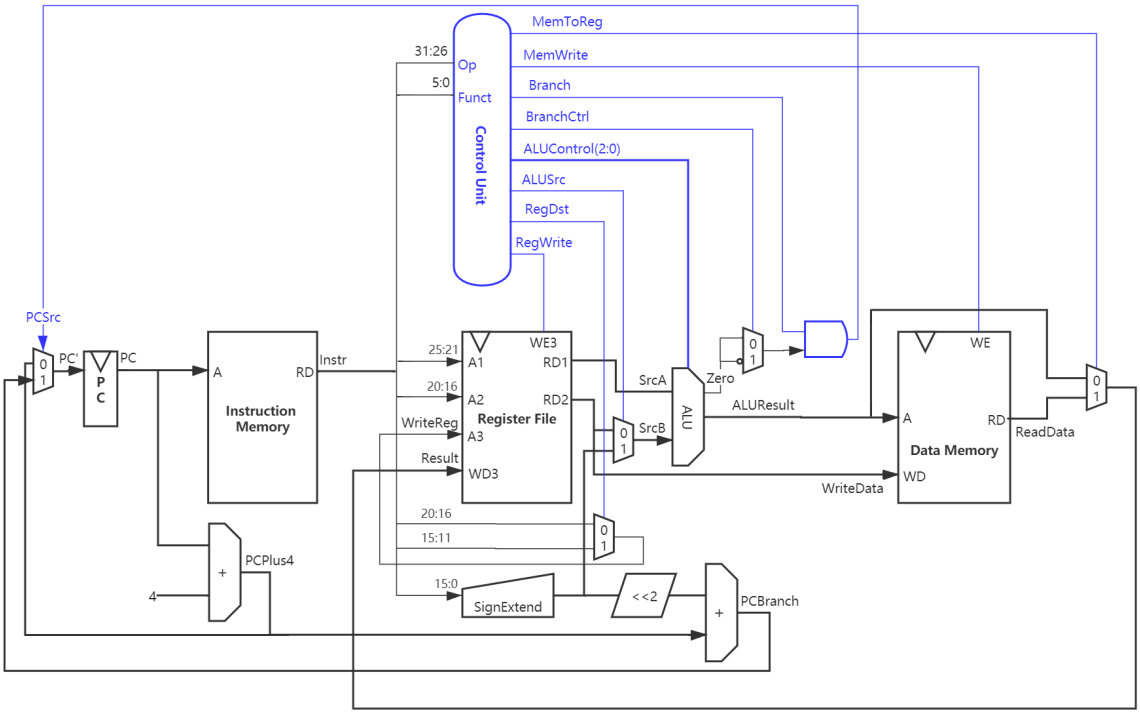
处理器的“状态元件”为程序计数器PC、指令内存、寄存器组、数据内存。我们以这些元件为基础，添加包括ALU在内的其他部件以处理各种类型指令，就可得到MIPS单周期处理器。



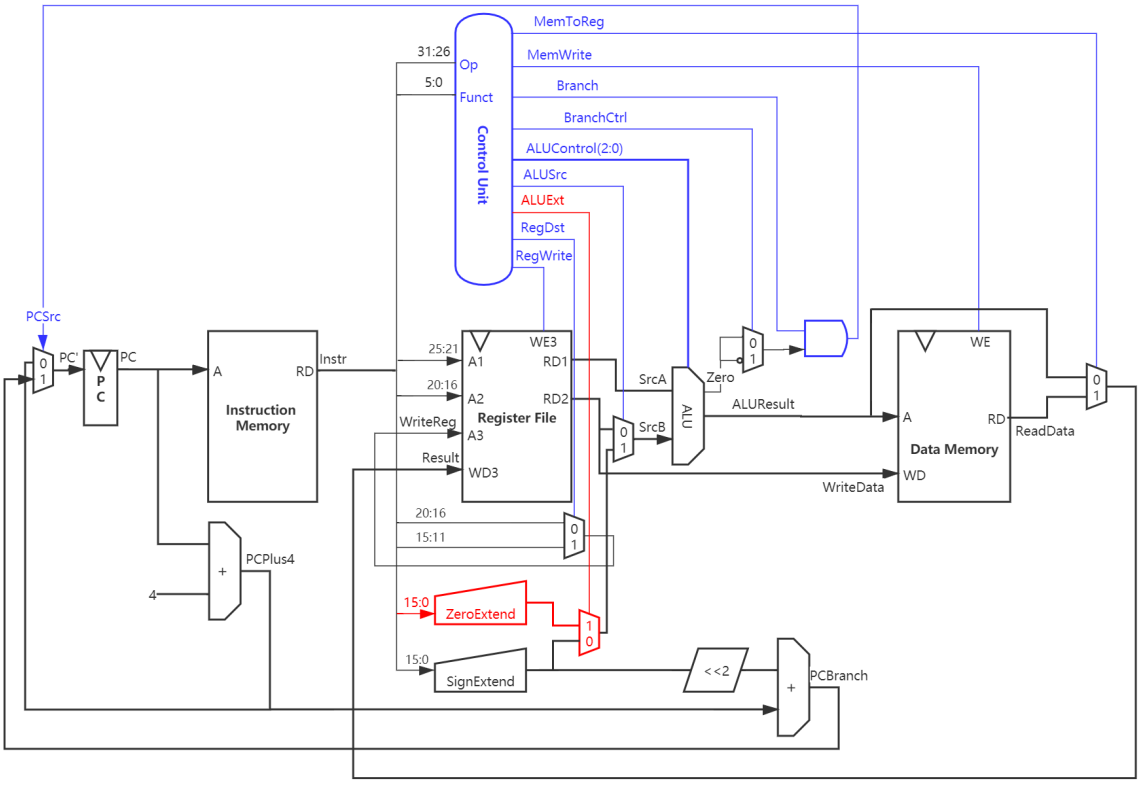
如上图所示，首先需要处理的是最基础的R类指令（寄存器-寄存器指令，包括add, sub, and, or, slt）、lw、sw指令，我们按照微处理器基本模型将元件相互连接，就可以实现这些指令了。



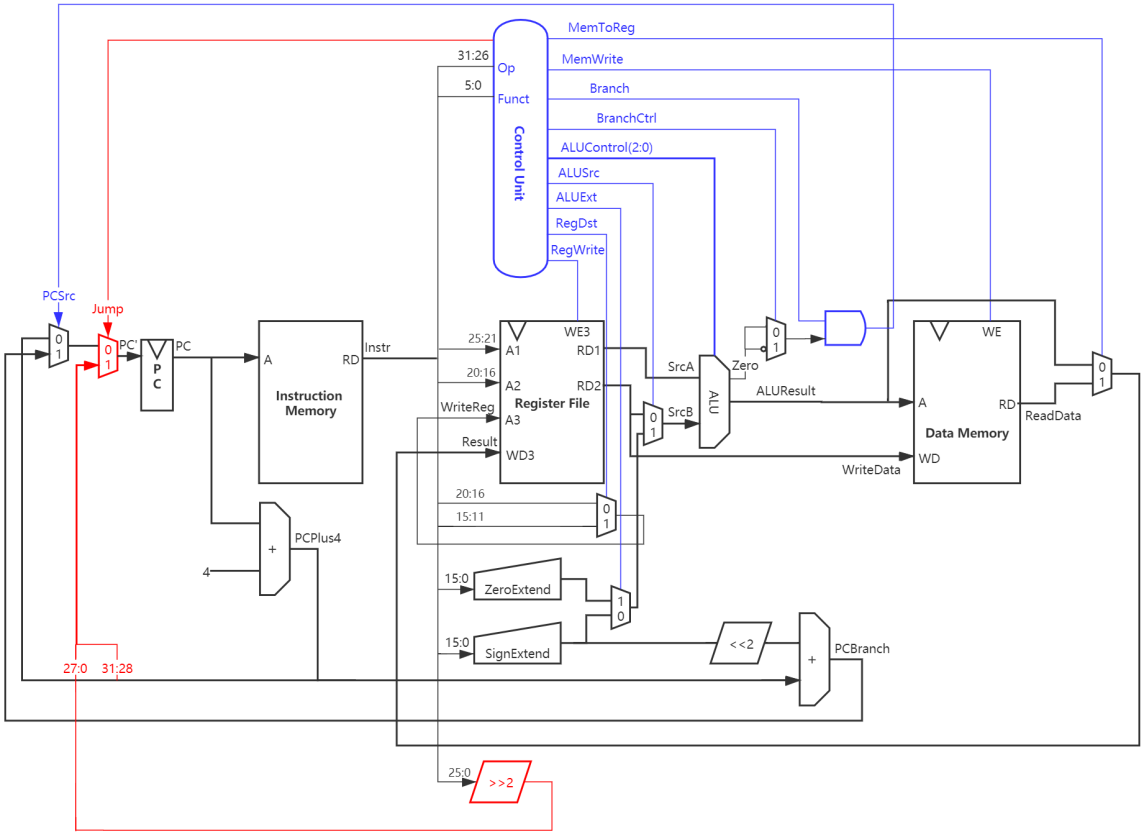
之后我们扩展数据路径以处理beq、bne两个分支指令。



与此同时我们实现控制单元，并与数据路径中的相应端口连接。

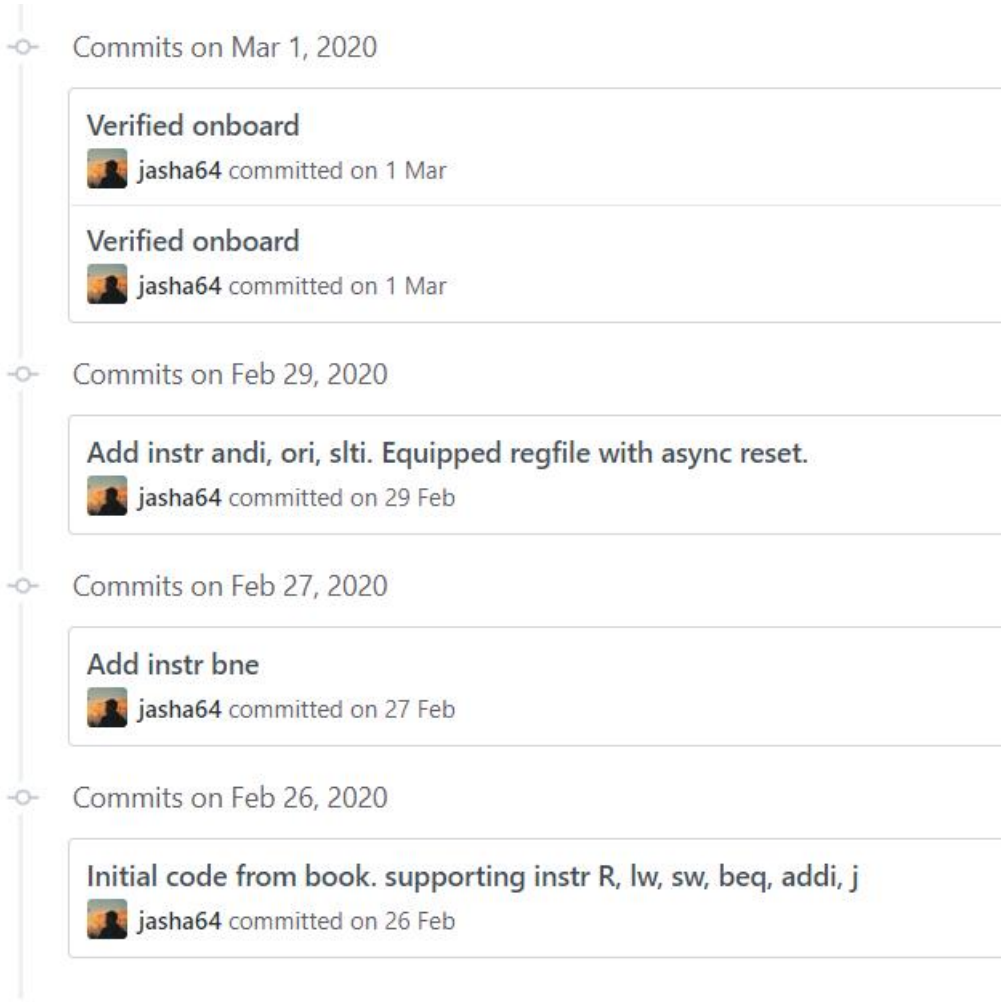


下一步是添加I类指令（立即数类型）andi, ori, slti（另一个I类指令addi用第一步中的电路就可以实现）。



最后，我们再添加元件以处理无条件跳转指令j。至此，我们就得到了一个功能较为完善的MIPS单周期处理器，支持的指令有：add，sub，and，or，slt，addi，andi，ori，slti，sw，lw，j，nop，beq，bne。

二、开发步骤



https://github.com/jasha64/MIPS_Pipeline_Cache_BPU

可于该github项目的提交记录中查看所有中间步骤的代码（包括最终成品）。（注：为防止抄袭，该项目目前为未公开状态，将于学期结束后公开）

三、测试

我们采用自动化的测试代码（代码见项目中的cpu_tb.sv）和拔尖课程提供的4笔测资进行测试。这4笔测资分别是：

ad hoc：课本7.6.3节提供的测资，覆盖书上实现过的所有指令（R类指令、addi、sw、lw、j）。

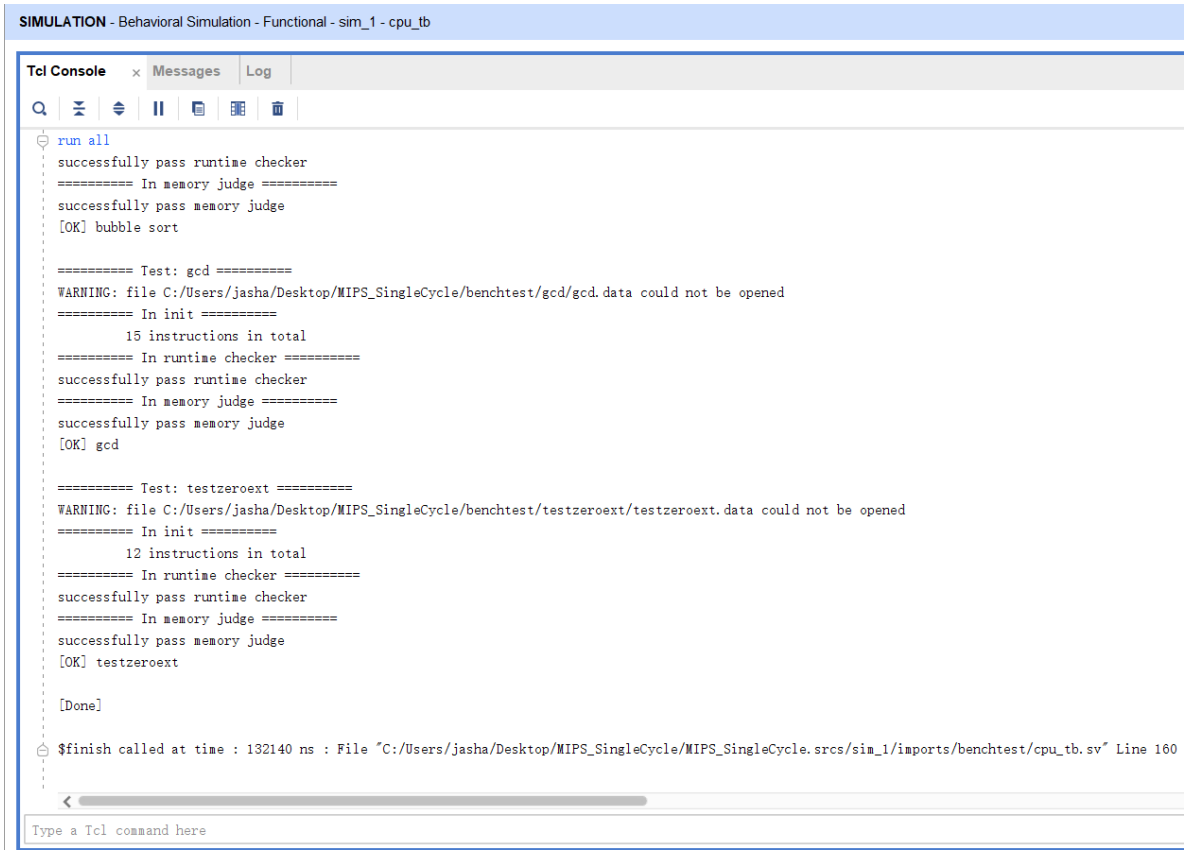
bubble sort：冒泡排序的汇编程序，主要考察循环过程（用bne指令）。

gcd：求最大公约数的汇编程序，主要考察循环过程（用j、beq指令）。

testzeroext：考察对操作数进行零扩展的I类指令（andi, ori）。

（具体测资文件，请见提交的项目目录/benchtest文件夹下的内容）

我的MIPS单周期处理器能通过上面所有的测资，测试结果如下：




```
========== Test: ad hoc ==========
WARNING: file C:/Users/jasha/Desktop/MIPS_SingleCycle/benchtest/ad hoc/ad hoc.data could not be opened
========== In init ==========
         19 instructions in total
========== In runtime checker ==========
successfully pass runtime checker
========== In memory judge ==========
successfully pass memory judge
[OK] ad hoc

========== Test: bubble sort ==========
========== In init ==========
         18 instructions in total
========== In runtime checker ==========
successfully pass runtime checker
========== In memory judge ==========
successfully pass memory judge
[OK] bubble sort

========== Test: gcd ==========
WARNING: file C:/Users/jasha/Desktop/MIPS_SingleCycle/benchtest/gcd/gcd.data could not be opened
========== In init ==========
         15 instructions in total
========== In runtime checker ==========
successfully pass runtime checker
========== In memory judge ==========
successfully pass memory judge
[OK] gcd

========== Test: testzeroext ==========
```

```
WARNING: file
C:/Users/jasha/Desktop/MIPS_SingleCycle/benchtest/testzeroext/testzeroext.d
ata could not be opened
========== In init ==========
         12 instructions in total
========== In runtime checker ==========
successfully pass runtime checker
========== In memory judge ==========
successfully pass memory judge
[OK] testzeroext

[Done]

$finish called at time : 132140 ns : File
"C:/Users/jasha/Desktop/MIPS_SingleCycle/MIPS_SingleCycle.srcs/sim_1/import
s/benchtest/cpu_tb.sv" Line 160
```

四、上板验证

我们选择两笔测资进行上板验证。

首先，我们将CPU的result信号直接连接到七段数码管，并将.\MIPS_SingleCycle\MIPS_SingleCycle.runs\Obj\all.dat中的内容烧录到指令内存中，观察CPU在每个周期的运算结果。测资内容与预期输出如下：（由本人自主设计）

```
addi $s0, $0, 12  # $s0 = 12 = 0x0000000c
andi $s2, $s0, -8  # $s2 = $s0 & 0x0000fff8 = 0x00000008 = 8
ori $s3, $s1, 10  # $s3 = $s1 | 0x0000000a = 0x0000000a = 10
slti $s4, $s2, 5  # $s4 = ($s2 < 5) = 0
nop  # 0
add $t0, $s0, $s1  # $t0 = $s0 + $s1 = 12
sub $t0, $s2, $s3  # $t0 = $s2 - $s3 = -2
and $t0, $s3, $s1  # $t0 = $s3 & $s1 = 0
or $t0, $s1, $s2  # $t0 = $s1 | $s2 = 8
slt $t0, $s0, $s2  # $t0 = ($s0 < $s2) = 0
sw $s0, 0($0)  # Mem[0] = $s0 = 12
lw $t0, 0($0)  # $t0 = Mem[0] = 12
nop  # 0
nop  # 0
```

由实验录像（.\测资1.mp4），CPU在开发板上运行的每条指令实际输出都符合预期输出。

然后，我们参照老师提供的IO接口课件，实现一个IO接口，从开发板上的开关读入数据（并显示到七段数码管左半部分）、将计算结果输出到LED灯（并显示到七段数码管右半部分）。
将.\MIPS_SingleCycle\MIPS_SingleCycle.runs\Obj\add.dat中的内容烧录到指令内存中，这笔测资的内容：（由老师课件提供）

```
main:
    addi $s0, $0, 0
    sw   $s0, 0x80($0)
chkSwitch:
    lw   $s1, 0x80($0)
    andi $s2, $s1, 0x2
    beq  $s2, $0, chkSwitch
    lw   $s3, 0x88($0)
    lw   $s4, 0x8C($0)
    add  $s5, $s4, $s3
chkLED:
    lw   $s1, 0x80($0)
    andi $s2, $s1, 0x1
    beq  $s2, $0, chkLED
    sw   $s5, 0x84($0)
    j    chkSwitch
```

我们将开关拨到"12"和"34"，单击右侧按钮(开关输入)，再单击左按钮(LED输出)，即可看到相加结果。实验照片如下图所示，也可查看实验录像（.\两位数相加.mp4）。