P5 课上测试 1&2

2021-12-02

H1 P5 课上测试

> 通过阅读本文，您可以大致了解 2021 年秋季北航计算机组成课程 P5 课上测试的题目内容、难度和解题思路
>
> P5 课上测试的主要内容是对课下用 Verilog 搭建的流水线 CPU 进行强测，同时添加一些新指令
>
> 题目每年都会发生变化，题意描述可能与原题有一定差异

P5 过了两次啊...

总结点经验教训吧

写在前面

关于 Mars

Mars 只是一个模拟器！

Mars 不是编译器，它不会自动改变你的汇编代码，给它加上延迟槽！！！如果你想要处理延迟槽，请自行加入无关指令或者 `nop`

同理，自己搭建的 CPU 会保持与 Mars 一样的行为，即执行 b/j 类指令的后面一条指令，这才是与 Mars 做对拍的原理

P5 课上测试 1&2

2021-12-02

在课上测试的时候，如果你看不明白指令的 RTL 语言到底说了些啥玩意，Mars 和提供的.class 文件是你唯一可以信任的工具，即使是 RTL 语言描述也不一定是完全正确的，你的 CPU 的行为务必与 Mars 保持一致！

新加指令务必在 Mars 上运行模拟一下！

新加指令务必在 Mars 上运行模拟一下！

新加指令务必在 Mars 上运行模拟一下！

为 CPU 添加计算指令

只需要改 control 的控制信号即可，这个没啥难度，如果要求更复杂的指令，可以写一个函数

为 CPU 添加 bltzal 指令

> Mars 中的 `bltzal` 指令行为有误，应当是无条件链接
> 这样的话，删除线里面的东西都不需要了...

`bltzal` 指令是 Mars 支持的唯一一个 branch and link 型指令，其添加步骤为

首先增加 `bltzal` 的定义，在 control 中新添 `wire bltzal` ，并且添加指令类型 `branch_link` ，其中包含 `bltzal` ，把 `bltzal` 同时添加进 `branch` 类型中

修改 `CMP` ，使其支持小于 0 的判断，这里注意符号运算 `$signed(rt) < 0`

修改 control，解码得到 `bltzal` 时 `NPCOp` 返回 `NPC_b` ， `CMPOp` 返回 `CMP_bltzal`

~~修改 DE, EM, MW 流水线寄存器，使其新增 b_jump 参与流水~~

~~修改 control，增加新的输入 b_jump，根据是否跳转以及指令是否属于 branch_link 类判断~~ ~~`GRFA3`~~ ~~时~~ ~~`$ra`~~ ~~还是~~ ~~`5'b0`~~

同时对于 `j_link` 和 `b_link` 指令，我们都有 `GRFWDSel` 为 `WDSel_pc8`

不需要考虑新的阻塞，因为 `bltzal` 属于 b 类指令，需要$Tuse_{rs}$ 为 0，因此原本的 `branch` 类的阻塞条件仍然适用；如果超时，或许可以考虑$Tuse_{rt}$ 为 3，这时可以不阻塞

2021-12-02

为 CPU 添加 bltzall 指令

多了一个 l 是 likely 的意思，反正就是指如果不跳转，清空延迟槽

这个时候在上面指令的基础上，在 `CMP` 里面加上一个新的输出 `flush` ，如果不跳就 `flushFD` 级流水线寄存器就行了

其他的都跟上面的 link 是一样的，而且也不用特别考虑阻塞，因为不跳 `GRFA3` 默认设成 0，就没有阻塞问题，已经被排除了

注意跳的时候的判断条件包括 `bltzall` 不能在被阻塞，否则下一条指令没法拿到正确的值

第一次课上测试

题目没记太全，大致说一下

T1 `swc`

是一条运算类指令，大意是如果 `rt` 寄存器值为奇数，则把 `rs` 寄存器值**循环左移** `rt[4:0]` 位，如果 `rt` 寄存器值为偶数，则把 `rs` 寄存器值**循环右移** `rt[4:0]` 位，结果存入 `rd` 寄存器中

解法

直接改动 `ALU` 即可，注意不能用位拼接，必须用 for 循环

关键代码如下（以左移为例）

```Verilog
C = A;
for(i = 0; i < B[4:0]; i = i + 1) C = {C[0], C[31:1]};
```

课上可以利用这种简单的运算类指令测试课下的正确性

H3 T2 bonall

#

印象深刻，下面是同学回忆的题面

2021-12-02

指令描述

opcode	rs	rt	offest
操作数没记	rs 寄存器地址	rt 寄存器地址	16 位偏移量

RTL 语言描述

```
I:   target_offest <- sign(offest||0^2)
     condition <- GPR[rs] + GPR[rt] = 0
     GPR[31] <- PC + 8

I+1: if condition:
                PC <- PC + starget_offest
      else:
                NullilyCurrentInstruction()
      endif
```

重要提示：`32'h8000_0000` 和 `32'h8000_0000` 不互为相反数

一句话描述：读 `$rs` 和 `$rt` 寄存器，如果两者值互为相反数，则**执行延迟槽后跳转并链接 PC+8**，否则**不执行延迟槽，同时仍然要链接 PC+8**

解法

首先明确是无条件链接，所以前面说的 `b_jump` 也不用加了，无脑写 `PC+8` 即可

修改 `CMP` ，使其支持相反数判断， `jump` 在是相反数时置为 1

关键其实在如何清空延迟槽， `CMP` 如果时 `bonall` 指令并且不跳转，则生成一个 `flush` 信号，如果 `flush` 为 1 并且没有处于阻塞状态，则 `flush` 一下 FD 级寄存器即可（*想一想：为什么可以直接清空 FD 级寄存器，**为什么不能处于阻塞状态***）

```Verilog
1    assign flush = (CMPOp == `CMP_bltzall && !jump);
```

显然 `bonall` 属于 `branch` 类型，添加后转发阻塞自动支持

我挂在了没看到无条件链接上面

T3 `lhogez`

是一条访存类，大致题目意思是：从 `DM` 中取一个半字，然后判断这个半字中是 1 多还是 0 多，如果 1 多就 `lh` 这个半字到 `rt` ，0 多就执行往 31 号寄存器里面写 `PC + 4`

解法

直接在数据通路中的 W 级判断读出的 `W_DMRD` 中半字的 0 多还是 1 多，生成一个 `check` 信号，然后输入到 control 里面去判断写什么，往哪里写

这里会存在 DEM 级没有 check 信号，输入的是浮空值，会导致判断出现问题

新知识（对我而言的）：Verilog 语法中有 `===` 三个等于号可以让浮空值 Z 和错误值 X 参与比较判断

阻塞可以遇到这条指令全阻到 D 级，也可以判断如果是读 `rt` 或者 `ra` 才阻塞在 D 级，后者可能会出现 `Run less cycles than expected` 错误

总结

总的来看，一定是计算 + 跳转 + 访存，计算最容易，一般只要改动 ALU，可以测一测看看课下有没有问题，我 T1 调了好一会，就是因为课下存在几个 bug

跳转有其固定的套路，只需要改一改 CMP，然后如果条件链接就把 b_jump 信号跟着流水线传递，然后在 control 里面把 GRFA3 的控制改成 `(branch_link && b_jump) ? 5'd31 : 5'd0` ，如果无条件链接就直接写 `5'd31` ，然后就是如果清空延迟槽的话，CMP 需要生成一个 flush_check 信号，大概是 `assign flush_check = (CMPOp == bonall? && !b_jump)` 就是如果是 bonall 并且不跳转则置为 1，然后改一下 `FD_REG_Flush = (flush_check && !Stall)` ，在这时清空 FD 寄存器，也就是清空了延迟槽指令

访存可以在 datapath 里面增加判断 DMRD 的逻辑，生成一个 check 信号连到 control 里面根据 check 判断写不写，写哪里，写什么这些问题，注意 check 信号为浮空值的问题，关键是考虑阻塞，暴力的方法是遇到访存指令，后面全阻塞在 D 级，稍微聪明点的方法是如果后面有这条指令，前面的寄存器需要读**后面这条指令可能需要写**的寄存器，在这里 lhogez 就是指 rt 和 31 号寄存器，则阻塞在 D 级

代码示例如下

```Verilog
wire E_stall_rs = ((E_lhogez ? (D_rs_addr == 5'd31 || D_rs_addr == E_rt_
wire E_stall_rt = ((E_lhogez ? (D_rt_addr == 5'd31 || D_rt_addr == E_rt_

wire M_stall_rs = ((M_lhogez ? (D_rs_addr == 5'd31 || D_rs_addr == M_rt_
wire M_stall_rt = ((M_lhogez ? (D_rt_addr == 5'd31 || D_rt_addr == M_rt_

assign Stall = E_stall_rs | E_stall_rt | M_stall_rs | M_stall_rt;
```

第二次课上测试

T1 `bltzal`

用法 `bltzal $rs, $rt, offset`

当 rs 的值小于 0 时，跳转到 `offset+2^rt[1:0]` 条指令后的位置

不论是否跳转延迟槽均执行，无条件链接 `GPR[31]=PC+8`

解法

相较于上一次的跳转，这一次修改了跳转的位置，因此需要相应的改变 NPC

无条件链接跟上一次做法相同

T2 addei

不支持溢出的有符号加法

用法 `addei $rs, $rt, imm16`

先把 `imm16` 做高位 1 扩展，然后与 `rs` 相加，如果溢出则向 `rt` 中写入高位 1 扩展的立即数，如果不溢出则向 `rt` 中写入加法的结果

解法

这次的计算指令既要改 `ALU` ，也要改 `EXT` ，添加 `EXTOp` 信号，支持高位 1 扩展溢出的判断方法根据 RTL 语言描述来就行

T3 lbget

大于等于 0 时存储字节

用法 `lbget $rs, $rt, offset`

指令描述

opcode	ba'se	rt	offest
110101	base 寄存器地址	rt 寄存器地址	16 位偏移量

RTL 语言描述

```
vaddr <- sign_extend(offest) + GPR[base]
pAddr <- vaddr31...2||0^2
memword <- memory[pAddr]
byte <- vAddr1...0
condition <- memword[7 + 8*byte] = 0
if condition:
        GPR[rt] <- sign_extend(memword7+8*byte...8*byte)
```

```
else:
        GPR[base] <- sign_extend(memword7+8*byte...8*byte)
endif
```

总之就是如果字节读出来 < 0，就写到 `base` 里面，否则就写到 rt 里面

解法

在 `mips.v` 的数据通路中添加 `DMRD` 的判断逻辑，生成一个 `lbget_check` 信号，代码如下

Verilog

```
reg lbget_check;
always @(*) begin
    if($signed(W_DMRD[7:0]) < 0) lbget_check = 1'b1;
    else lbget_check = 1'b0;
end
```

然后再 control 里面的 `GRFA3` 里面添加下面的判断

Verilog

```
assign GRFA3 = (lbget && lbget_check === 1'bz) ? 5'd0 :
               (calc_r | jalr) ? rd :
               (calc_i | (load && !lhwo) | (lbget && lbget_check == 1'b1
                  (lbget && lbget_check == 1'b0) ? rs :
               (jal | branch_link) ? 5'd31 :
               5'd0;
```

阻塞如上文所写的那样就行

如果你是按照上文中比较聪明的方法阻塞，可能课上测试的时候会 `Run less cycle`，这时候改成暴力阻塞就行了，出现这个错误难道是 CPU 跑的比标程快?

总结

其实第二次课上测试的题目比第一次还要灵活一点，但是总结完第一次的套路之后，本次测试 1.3h 就解决了，所以想要过课上还是要**非常熟悉自己的 CPU 流水线架构**，不管是自己设计的还是借鉴别人的架构，**自己一定要想一想常见的跳转和访存怎么添加**

北航计算机组成原理 Verilog MIPS 流水线CPU P5课上

更新于 2021-12-02 阅读次数 774 次

赞赏

请我喝[茶]~(￣▽￣)~*

P5 课上测试 1&2

上一篇
北航计算机组成原理
P5课下学习—流水线CPU设计(1)

下一篇
北航计算机组成原理
P6课下&课上总结

P5 课上测试 1&2

最新评论

1 花露水 @ 62 days ago
大佬保佑我今晚pre不挂

2 do @ 2022-11-20
十分感谢，这篇帮我省下了大量琢磨P7的时间。（话说这篇的实现方法和roife学长的好像）

3 Anonymous @ 2022-06-15
哈哈，现在再看，我现在连OS的Extra都懒得做，当年还兴致勃勃的去做不加分的P8 人的变化可真大...

4 陈俊一 @ 2021-12-13
p6一遍过前来还愿

5 陈俊一 @ 2021-12-07
p5一遍过前来还愿

6 Edge @ 2021-11-23
但愿P4一遍过

7 Edge @ 2021-11-23
一神！！！

8 陈俊一 @ 2021-11-23
太强了

9 陈俊一 @ 2021-11-23
太强了

10 Administrator @ 2021-11-18
不要随便输名字啊

随机文章

1 北航操作系统
BUAA-OS Lab4实验总结

2 北航计算机组成原理
P1课下学习

3 北航计算机组成原理
P5课上测试1&2

4 北航计算机组成原理
Verilog 学习笔记（2）

5 北航计算机组成原理
P5课下学习—流水线CPU设计(1)

6 北航操作系统
BUAA-OS Lab5实验总结

7 北航计算机组成原理
Logisim斐波那契电路

8 北航计算机组成原理
P1课上测试游记

9 北航计算机组成原理
P8课下&课上总结

10 北航操作系统
BUAA-OS Lab1实验总结