五级流水线RISC-V CPU 实验报告

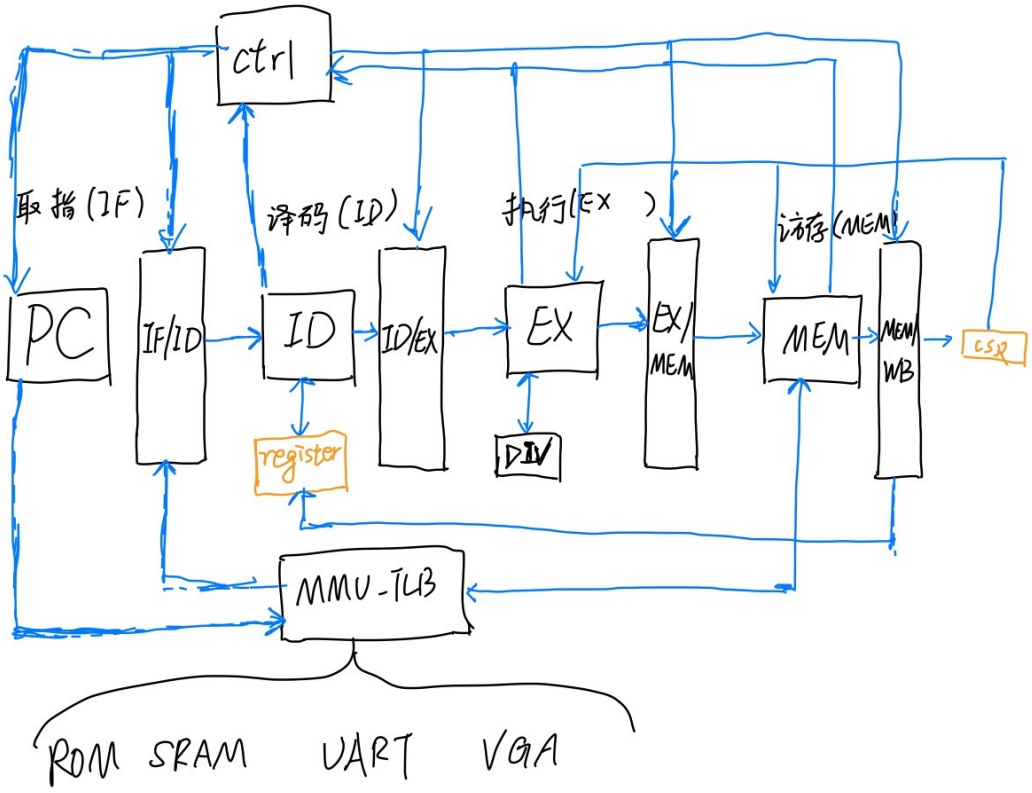
张行健 郭峥岩 王高源

实验目标

1. 深入理解流水线结构计算机指令的执行方式，掌握流水处理器的基本设计方法。
2. 深入理解计算机的各部件组成及内部工作原理。
3. 加深对于 RISCV32I 指令集的理解。
4. 掌握计算机外部输入输出的设计。
5. 提高硬件设计和调试的能力。

数据通路：五级流水线

数据通路采用五级流水设计，包括取指（IF）、译码（ID）、执行（EX）、访存（MEM）、写回（WB）五个模块。模块内部使用组合逻辑设计，模块之间有寄存器保存运算结果。下面是我们的数据通路设计图。



模块设计

流水线结构设计

IF & pc_reg

pc_reg存储当前访问的指令的地址，一般是一个时钟周期+4更新，会受到跳转指令、流水线暂停的影响。

IF阶段没有单独的模块，只有IF_ID这样的中间模块，是时序逻辑，每个时钟上升沿将读取出的指令保存下来供下个阶段使用。

ID

ID阶段实现指令译码，识别同时解析从IF传递过来的指令，产生相应的流水线控制信号传递给EX模块，是一个组合逻辑电路。

ID阶段同时还接入了从EX和MEM模块回接的数据旁路，用于处理解决数据冲突。同时输出相应的信号到control控制模块，用于解决控制冲突。

对于中断异常指令，解析出相应的指令之后把信息传递给EX和MEM模块，为了实现精确异常，处理异常相关的工作在后续MEM阶段进行。

EX

EX阶段实现算术逻辑运算，接收ID阶段传来的指令类型、运算数等信息并做相应的运算。运算类型分为逻辑运算、移位运算、算术运算、U型指令运算。运算结果、使能信号等传给MEM模块，最后在回写阶段保存至寄存器。

EX部分同时还需与CSR寄存器相连，读取所需的寄存器值。

对于两条异常指令，EX阶段设置了相应的mepc、mcause的值。

MEM

MEM阶段实现访存，与MMU_TLB模块，顶层模块中的存储相关信号配合完成对sram与uart的读写。

MEM阶段主要将储存与访问指令解析出来，把读写信号、数据信号、虚拟地址信号准备好，连接到mmu_tlb，同时把外部输入的数据信号接入。

WB

wb与if一样没有单独的模块，功能直接实现在mem_wb中，与regfile模块连接写入寄存器的值。

冲突处理

数据冲突

数据冲突的解决和课件中采取的方法基本一致，由于RISCV中产生的寄存器相关的冲突只有写后读冲突。

内存无关的写后读冲突，添加数据旁路，将结果尽快前传到需要使用它的位置。

内存相关的写后读冲突，采用暂停流水线（插气泡）的方法，等待相应数据写入之后再使用相应的信息。

结构冲突

当MEM阶段有数据需要读写时会与取指阶段产生结构冲突，此时需要将取指阶段暂停来读写内存。

对内存读写的使能信号可以直接当成对取指阶段暂停的信号，取指阶段收到暂停信号后将传入后面模块的数据改为Zero，实现插入气泡。

控制冲突

ID阶段解析指令，计算出转移地址，产生相应的转移信号给控制模块。控制模块在接收到相应的信号之后会暂停流水线，同时将PC值置为转移指令的目标地址，之后流水线正常运行即可。

总体而言暂停了一个时钟周期，实现了跳转指令。

中断异常

为了实现精确异常，我们在之前流水线的各个阶段收集异常信息，之后在MEM阶段进行统一的异常处理。从MEM模块将异常指令的标志传入control模块，同时将用到的csr寄存器的相关信息接入control模块。

control模块会产生flush信号清空流水线，同时将接下来的PC设置成相应的异常处理程序的入口地址或者是返回的PC地址。

同时设定一个寄存器表示CPU当前所处的模式，异常指令只有在相对应的模式下才能处理与识别。

中断异常同时要实现的还有模式之间的转换，分别为M,S,U三种模式。本次实验过程中运行kernel的时候处于M模式，通过mret等指令，跳转PC地址的同时也会更改模式。ecall,ebreak等异常也会更改模式。有些指令（比如mret）只有在M模式下才能运行（具有更高的权限）。执行这些指令的同时要设定csr寄存器中的某些字段的相应值，以便能够正确的维护权限管理。

存储模块：MMU+TLB

本模块通过访问TLB和内存中的页表完成虚拟地址到物理地址的转换，同时根据物理地址的MMIO将相应区间的地址映射到sram与uart上，对不同的外设产生读写信号。虚拟地址与物理地址的访问同样配合了用户态与机器态的权限管理。

TLB采用直接相联的设计，支持刷新，更新。

IF阶段与MEM阶段各设置一个mmu_tlb模块，分别负责指令与数据，防止冲突。

具体工作流程是：

1. 判断CPU模式，若为用户态，则进入2开始虚实地址转换；若为机器态，则访问的地址应是物理地址，跳转到7
2. 判断虚拟地址是否在指定的区间内，如果不在则触发访问地址错误的异常
3. 从TLB中读取表项，如果存在表项则直接读取表项进行转换，跳转到7；如果不存在表项则进入读取二级页表的状态机
4. 访问内存读取一级页表，此时暂停流水线
5. 访问内存读取二级页表，此时暂停流水线
6. 更新TLB
7. 此时得到了物理地址，通过物理地址访问内存得到数据
8. 跳转到1，进行下一个数据的读取

SRAM

SRAM通过组合逻辑实现了一个时钟周期内读写，代价就是时钟频率不能太高。读写信号的给出方式与之前的实验相似，只不过没有了在读写信号拉低前的数据保持时间，实验表明这样也是可以工作的。

UART

UART读写不用硬件来判断状态，而是采用软件轮询的方式来进入读写，因此硬件负责在相应的存取指令处将读写信号拉低即可。

成果展示

成果概述

- 五级流水线32位RISC-V CPU，主频35M，支持RISC-V基础指令与3条拓展指令

- 支持中断异常
- 支持页表与TLB

成果展示

性能测试

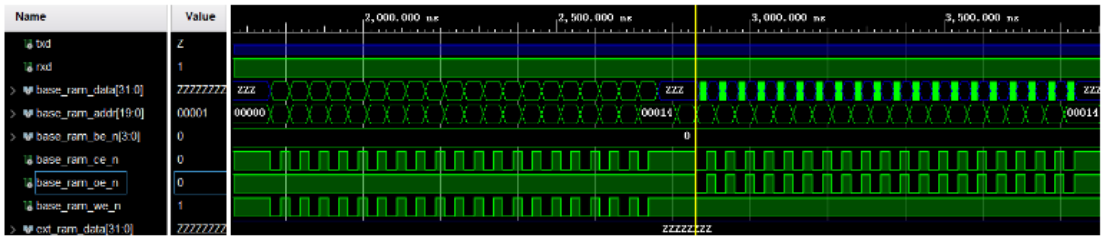
主频：35M

名称	1PTB	2DCT	3CCT	4MDCT	CRYPTONIGHT
时间(s)	11.409	5.704	13.310	11.303	0.565

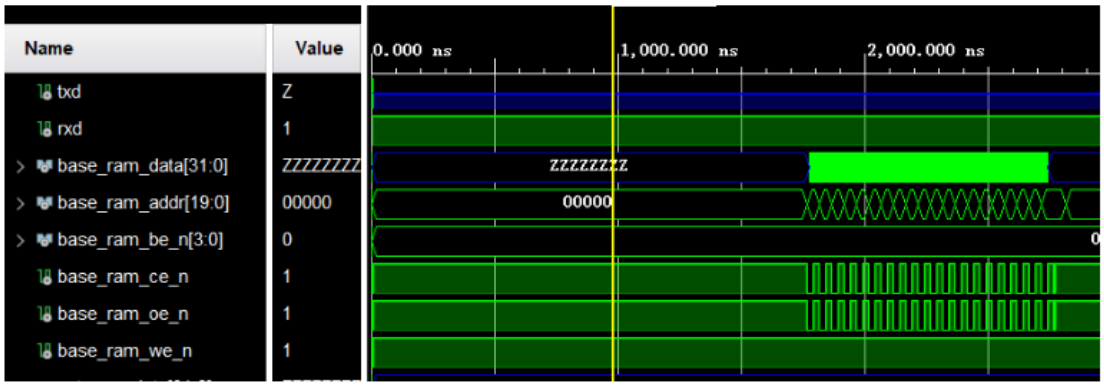
仿真

实验过程伴随着详细的前仿真、后仿真，功能仿真、时序仿真等。

functional simulation



timing simulation



真机测试

ecall演示：

```
>> a
addr: 0x80100000
one instruction per line, empty line to end.
[0x80100000] .section .text
[0x80100000] .globl _start
[0x80100000] _start:
[0x80100000]  addi s0, x0, 30
[0x80100004]  addi t0, x0, 0x21
[0x80100008]  addi t1, x0, 0x80
[0x8010000c] loop:
[0x8010000c]  add a0, x0, t0
[0x80100010]  ecall
[0x80100014]  addi t0, t0, 0x1
[0x80100018]  bne t0, t1, loop
[0x8010001c]  jr ra
[0x80100020]

>>
Invalid command
Usage:  R: print registers
        D: display memory
        A: put assembly at specified address
        U: read data and disassemble
        G: run user code
        T: print page table
>> g
addr: 0x80100000
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
elapsed time: 0.131s
>>
```

虚拟地址映射演示：斐波那契数列

```
>> a
addr: 0x80100000
one instruction per line, empty line to end.
[0x80100000]    li t0,0x7FC10000
[0x80100004]    addi t1,x0,0x0
[0x80100008]    addi t2,x0,0x1
[0x8010000c]    addi t4,x0,0xa
[0x80100010] loop:
[0x80100010]    add t3,t1,t2
[0x80100014]    #将斐波那契数存入其相对应的位置
[0x80100014]    sw t2,(t0)
[0x80100018]    addi t0,t0,0x4
[0x8010001c]    add t1,x0,t2
[0x80100020]    add t2,x0,t3
[0x80100024]    add t4,t4,-1
[0x80100028]    bne t4,x0,loop
[0x8010002c]    jr ra
[0x80100030]
>> g
addr: 0x80100000

elapsed time: 0.000s
>> d
addr: 0x80400000
num: 100
0x80400000: 0x00000001
0x80400004: 0x00000001
0x80400008: 0x00000002
0x8040000c: 0x00000003
0x80400010: 0x00000005
0x80400014: 0x00000008
0x80400018: 0x0000000d
0x8040001c: 0x00000015
0x80400020: 0x00000022
0x80400024: 0x00000037
0x80400028: 0xf7dbb438
0x8040002c: 0x3efb7687
0x80400030: 0xc7df6ed0
0x80400034: 0x78fbedda
0x80400038: 0x0f1f7dbb
```

组内分工：

张行健：基本框架的搭建，MEM模块，TLB，SRAM，环境搭建与测试软件编写

郭峥岩：ID模块、冲突处理（control模块）、中断异常、MMU_TLB

王高源：EX模块，CSR读写，中断异常

心得体会

张行健：

1. 前仿真可以检查出一些逻辑上的错误，后仿真可以检查出硬件时序设计上的一些错误。设计仿真流程同样需要花费很大的精力
2. 实验可以促进基础知识的学习，基础知识不牢固做实验会遇到很大的困难

郭峥岩：

1. 实现软件思维到硬件思维的转换很关键，理解时序的变化的同时流水线中发生的变化对于各个模块的设计至关重要。
2. 有时自己觉得效果相同的两个时序逻辑与组合逻辑电路，可能会由于延迟等原因而产生微妙的区别。
3. 仿真在硬件实验中非常重要。

王高源：

1. 软硬件思维转换很重要，硬件往往会牵扯到时序的很多问题，若不认真设计就会和想象中所实现的功能大相径庭。
2. 完成这次实验也让我对计算机较低层的实现有了更深的理解，收获还是很多的。
3. 团队合作很重要，队友很强也能教会自己学更多东西。

附录

代码仓库：https://git.tsinghua.edu.cn/cod-ta/cod20-grp27

部分实验过程记录在在线文档中：https://shimo.im/docs/6d8yTTj3yq8yYP9C

指令流程表

指令	ID	EX	MEM
ADD	获取两个源寄存器的值，获取目的寄存器的地址	计算结果	-
AND	获取两个源寄存器的值，获取目的寄存器的地址	计算结果	-
OR	获取两个源寄存器的值，获取目的寄存器的地址	计算结果	-
XOR	获取两个源寄存器的值，获取目的寄存器的地址	计算结果	-
MIN	获取两个源寄存器的值，获取目的寄存器的地址	计算结果	-
MINU	获取两个源寄存器的值，获取目的寄存器的地址	计算结果	-
ADDI	获取一个源寄存器的值，一个立即数的值，获取目的寄存器的地址	计算结果	-
ANDI	获取一个源寄存器的值，一个立即数的值，获取目的寄存器的地址	计算结果	-
ORI	获取一个源寄存器的值，一个立即数的值，获取目的寄存器的地址	计算结果	-
SLLI	获取一个源寄存器的值，一个立即数的值，获取目的寄存器的地址	计算结果	-
SRLI	获取一个源寄存器的值，一个立即数的值，获取目的寄存器的地址	计算结果	-

指令	ID	EX	MEM
CTZ	获取一个源寄存器的值，获取目的寄存器的地址	计算结果	-
SB	获取两个源寄存器的值，获取目的寄存器的地址	获取offset，计算地址	内存存储
SW	获取两个源寄存器的值，获取目的寄存器的地址	获取offset，计算地址	内存存储
LB	获取一个源寄存器的值，获取目的寄存器的地址	获取offset，计算地址	内存读取
LW	获取一个源寄存器的值，获取目的寄存器的地址	获取offset，计算地址	内存读取
LUI	获取目的寄存器的地址	-	-
JAL	获取目的寄存器的地址，计算跳转地址，开始暂停流水线进行跳转	-	-
JALR	获取目的寄存器的地址，计算跳转地址，开始暂停流水线进行跳转	-	-
AUIPC	获取一个立即数的值，获取目的寄存器地址	计算PC与立即数的和	-
CSRRW	获取一个源寄存器的值	读CSR原值并计算	将新值写入CSR
CSRRC	获取一个源寄存器的值	读CSR原值并计算	将新值写入CSR

指令	ID	EX	MEM
CSRRS	获取一个源寄存器的值	读CSR原值并计算	将新值写入CSR
ECALL	解析指令	生成mepc和mcause的值	输出信号给control，之后由control控制跳转到异常处理程序地址处，同时连接输出信号给csr寄存器模块，修改相应值并按照指令功能更改mode，并将当前PC保存
EBREAK	解析指令	生成mepc和mcause的值	输出信号给control，之后由control控制跳转到异常处理程序地址处，同时连接输出信号给csr寄存器模块，修改相应值并按照指令功能更改mode，并将当前PC保存
MRET	解析指令	-	输出信号给control，之后由control控制跳转到用户程序地址处，同时连接输出信号给csr寄存器模块，修改相应值并按照指令功能更改mode
sfence.vma	解析指令，输出信号清空tlb	-	-