

# Arch-2025-Lab1 实验报告

## 基础流水线构建与运算符指令

23307130064 周弈成

## 内容简介

根据五级流水线的CPU框架，构建能够实现基本运算指令的CPU。实现的运算指令有：

- 算术运算与逻辑运算：addi, xori, ori, andi, add, sub, and, or, xor
- 扩展指令：addiw, addw, subw

## 实验结果

能够通过测试，显示“HIT GOOD TRAP”。

```
juliki@DESKTOP-3R69GOQ: ~ × + v
TEST= ./build/emu --diff /home/juliki/arch-2025/ready-to-run/riscv64-nemu-interpretor-so -i ./ready-to-run/lab1/lab1-test.t.bin --dump-wave || true
Emu compiled at Feb 27 2025, 00:44:37
The image is ./ready-to-run/lab1/lab1-test.bin
Using simulated 256MB RAM
Using /home/juliki/arch-2025/ready-to-run/riscv64-nemu-interpretor-so for difftest
[src/device/io/mmio.c:19,add_mmio_map] Add mmio map 'clint' at [0x38000000, 0x3800ffff]
[src/device/io/mmio.c:19,add_mmio_map] Add mmio map 'uartlite' at [0x40600000, 0x4060000c]
[src/device/io/mmio.c:19,add_mmio_map] Add mmio map 'uartlite1' at [0x23333000, 0x2333300f]
dump wave 0-999999 to /home/juliki/arch-2025/build/1740588278.fst...
The first instruction of core 0 has committed. Difftest enabled.
[src/cpu/cpu-exec.c:393,cpu_exec] nemu: HIT GOOD TRAP at pc = 0x00000000080010004
[src/cpu/cpu-exec.c:394,cpu_exec] trap code:0
[src/cpu/cpu-exec.c:74,monitor_statistic] host time spent = 4365 us
[src/cpu/cpu-exec.c:76,monitor_statistic] total guest instructions = 16385
[src/cpu/cpu-exec.c:77,monitor_statistic] simulation frequency = 3753722 instr/s
Program execution has ended. To restart the program, exit NEMU and run again.
Program execution has ended. To restart the program, exit NEMU and run again.
sh: 1: spike-dasm: not found

===== Commit Group Trace (Core 0) =====
commit group [0]: pc 0080010000 cmtcnt 1
commit group [1]: pc 0080010004 cmtcnt 1
commit group [2]: pc 0080010008 cmtcnt 1
commit group [3]: pc 008001000c cmtcnt 1 <--
commit group [4]: pc 008000fffd0 cmtcnt 1
commit group [5]: pc 008000fffd4 cmtcnt 1
commit group [6]: pc 008000fffd8 cmtcnt 1
commit group [7]: pc 008000fffdc cmtcnt 1
commit group [8]: pc 008000fffe0 cmtcnt 1
```

## 文件结构

在/vsrc下，有/include、/pipeline、/src三个文件夹，其中：

- /include：代表头文件，其中新建的pipes.sv包含必要的数据类型与结构体定义；
- /pipeline：代表流水线执行的各个阶段模块；
- /src：放置调用流水线执行模块的CPU核心流程模块core.sv。

## 阶段介绍

分为取指、译码、执行、访存、写回五阶段。

取指阶段：仅把指令传入dataF，并使程序计数器pc改为原值+4。

译码阶段：将指令译码为控制信号，立即数和寄存器地址，需要把立即数符号扩展，并访问寄存器取出读入寄存器的值。

- 根据dataF中指令，按照RISC-V指令格式拆分，先判断op部分确定I/R指令以及是否为64位扩展的指令，再根据F3和F7判断运算符号。
- 所有指令均需要写回，相关控制信号设为真并记录地址；
- I指令ALU一端为立即数，R指令则为寄存器值，控制信号相应区分；
- w型运算指令需要截取并符号扩展ALU结果，需要相应控制信号；
- 运算符号也作为控制信号解析；
- I指令需要符号扩展12位立即数值，一并编写扩展模块。

执行阶段：将译码阶段获取的寄存器值放入ALU，根据控制信号确定的运算符号计算。

- ALU即根据传入控制信号运算；
- w型指令需要符号扩展32位立即数值，一并编写扩展模块。

访存阶段：目前无效果，传递执行阶段的值。

写回阶段：根据访存阶段的运算结果和控制信号传来的写回地址，写入寄存器模块。

寄存器文件需要单独编写，同时具有读取和写入功能，根据控制信号开放使能，读取功能为立即读取的组合逻辑，而写入需要等待一周期，为时序逻辑。

## 流程与接口设计

此次实验的流水线由于无非运算指令，不会冲突，原则简单：上一阶段的数据处理完毕后下一阶段开始工作。在数据结构体中，通过valid标记，收到上阶段数据valid信号后，下阶段工作。

取指阶段的开始需要等待外部接口传入数据，且内部指令处理完毕。

接口需要输出写回阶段的状态和写回操作。

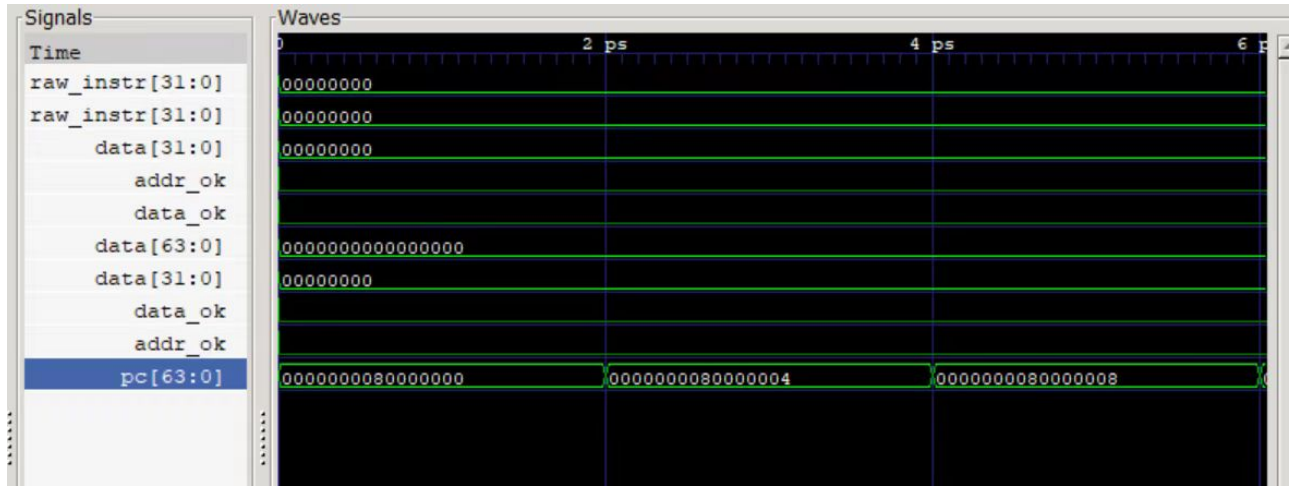
需要把实时更新的寄存器文件（下一周期）输出，因而设计组合逻辑写入临时寄存器next\_reg并作为decode阶段输出，到下阶段时序逻辑同步。

## 实验过程

先设计模块，再设计流程和接口。

- 模块设计参照给出的MIPS32位流水线CPU模块，改建为RISC-V64位，接口几乎相同（指令格式有变化）。
- 设计后，起初将各个模块的刷新条件统一设置为外部接口传入数据，导致各模块无法工作。

- 调试时，原因排查许久，最后索性设置无提交，观察完整的pc与指令执行波形图，发现了问题。



无法正常显示指令的问题图

- 修复后，可以成功运行，但是出现新的问题：寄存器x0不能写入非0值。

```

juliki@DESKTOP-3R69GOQ: ~
commit group [7]: pc 008000011c cmtcnt 1
commit group [8]: pc 0080000120 cmtcnt 1
commit group [9]: pc 0080000124 cmtcnt 1
commit group [a]: pc 0080000128 cmtcnt 1
commit group [b]: pc 008000012c cmtcnt 1
commit group [c]: pc 0080000130 cmtcnt 1
commit group [d]: pc 0080000134 cmtcnt 1
commit group [e]: pc 0080000138 cmtcnt 1
commit group [f]: pc 008000013c cmtcnt 1

===== Commit Instr Trace =====
commit inst [0]: pc 0080000140 inst bf42cb13 wen 1 dst 00000016 data ffffffffbbbf4
commit inst [1]: pc 0080000144 inst 019e6e33 wen 1 dst 0000001c data ffffffffbbcd9
commit inst [2]: pc 0080000148 inst 0186e033 wen 1 dst 00000000 data 000000000000f79 <--
commit inst [3]: pc 008000010c inst d3ecf793 wen 1 dst 0000000f data 0000000000000000
commit inst [4]: pc 0080000110 inst af1a8993 wen 1 dst 00000013 data ffffffffbbaf1
commit inst [5]: pc 0080000114 inst 0009ef33 wen 1 dst 0000001e data ffffffffbbaf1
commit inst [6]: pc 0080000118 inst 012d8b33 wen 1 dst 00000016 data 0000000000000453
commit inst [7]: pc 008000011c inst 00dcc933 wen 1 dst 00000012 data 0000000000000458
commit inst [8]: pc 0080000120 inst 7784089b wen 1 dst 00000011 data 000000000000002d
commit inst [9]: pc 0080000124 inst 004b043b wen 1 dst 00000008 data 0000000000000598
commit inst [a]: pc 0080000128 inst 01e8f333 wen 1 dst 00000006 data 0000000000000021
commit inst [b]: pc 008000012c inst 01dfe733 wen 1 dst 0000000e data ffffffffbbfff
commit inst [c]: pc 0080000130 inst 403a80b3 wen 1 dst 00000001 data 000000000000074b
commit inst [d]: pc 0080000134 inst d4eae993 wen 1 dst 00000013 data ffffffffbbfd4e
commit inst [e]: pc 0080000138 inst 00eac233 wen 1 dst 00000004 data ffffffffbbfff
commit inst [f]: pc 008000013c inst 41c003b3 wen 1 dst 00000007 data 0000000000000327

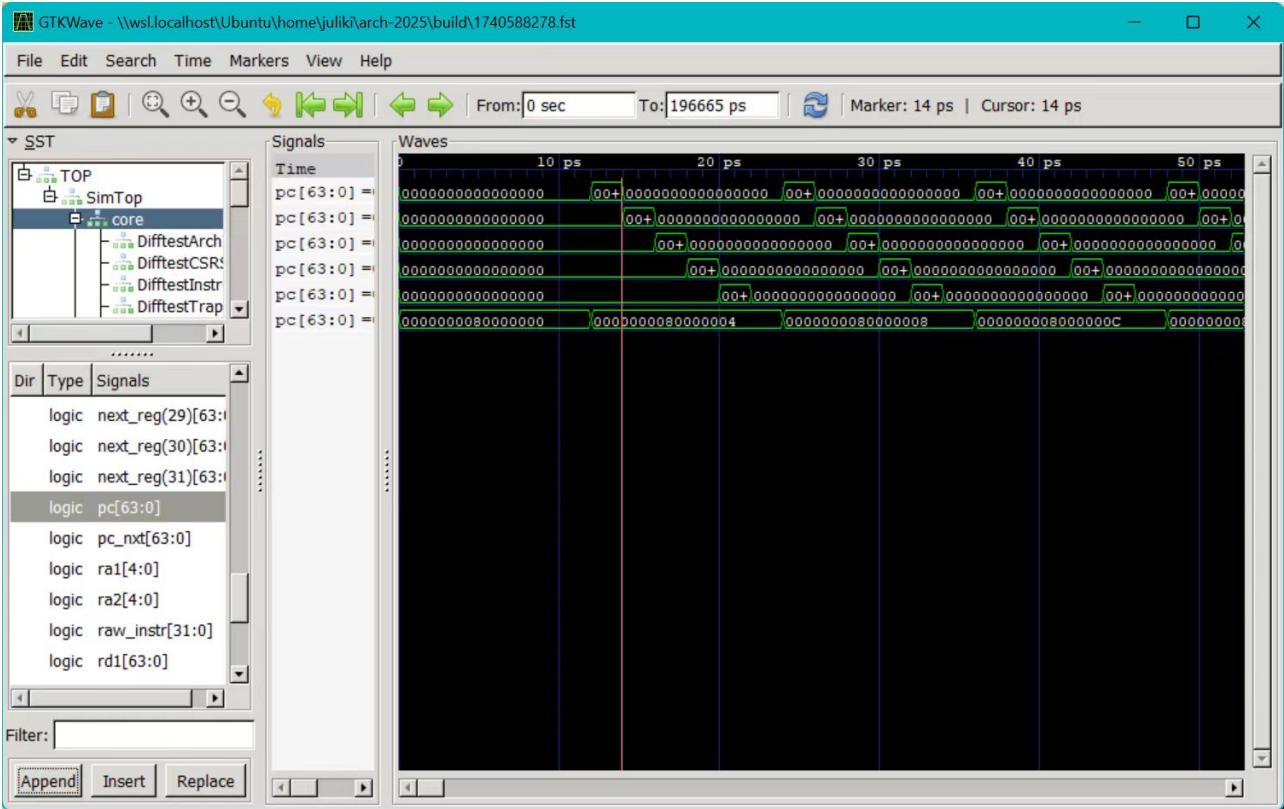
===== REF Regs =====
$0: 0x0000000000000000 ra: 0x000000000000074b sp: 0x0000000000000000 gp: 0xffffffffffff8b5

```

调试写入x0的错误

- 修改写回阶段控制信号逻辑：地址为0时关闭写使能，后能正常运行。

- 波形图呈周期六段。



最终的指令波形图