实验二：取指、译码模块实现

1 实验目的

i. 熟悉 RISC-V 指令集；

ii. 掌握单周期 CPU 各个控制信号的作用和生成过程；

iii. 掌握单周期 CPU 控制器的工作原理及其设计方法；

iv. 掌握取指、译码阶段数据通路、控制器的执行过程；

2 实验环境（推荐）

与实验一相同：

i. IDE: vscode

ii. verilog compiler: iverilog

iii. waveform viewer: GTKwave

3 实验任务

本实验要求大家编写一个取指（IF）模块和一个译码（ID）模块。其中，IF 模块包含 PC、ADD、Instr Memory 三个子模块；ID 模块包含 Register File、Control Unit、Imm Generator 三个子模块。

3.1 实验描述

图1给出了取指、译码模块的设计图。

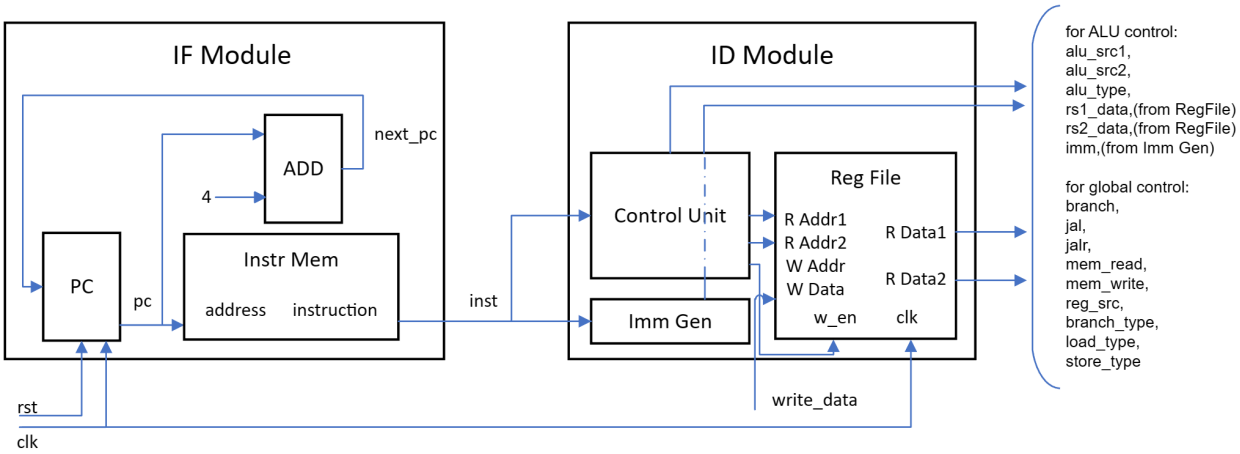

图 1: IF-ID 模块设计图

3.2 实验要求

1. PC：D 触发器结构的 PC 寄存器。这个模块需要实现 3 个输入，分别为 *clk*, *rst*, *new_pc*。其中，*clk* 和 *rst* 分别连接时钟信号与复位信号；而 *new_pc* 信号直接连接下一个周期的 PC 值（暂时不考虑跳转）。此外，这个模块需要实现一个输出，为 *pc*。*pc* 信号即当前周期的 PC 寄存器值，需要连接到加法器的一个输入端口以及指令存储器的 *address* 端口。PC 寄存器为 32 位。

2. ADD：加法器，用于计算下一条指令地址，输入值分别为 *pc* 和立即数 *32'h4*。

3. Instr Memory：复用 Lab1 中的 RAM 模块。*address* 端口接收当前周期的 PC 值作为指令数据的地址，从 *read_data* 端口读指令。指令利用 Verilog 的 `inital` 语句提前烧录。由于指令内存对于取指模块来说是只读的，因此所有的与写操作相关的端口都不连接到其他任何模块上。

4. Register File：寄存器堆，其中包含由 32 个 32 位寄存器构成的寄存器数组。这个模块需要实现 6 个输入和 2 个输出。其中，输入包含两个读地址端口、一个写地址端口、一个写数据端口、时钟信号 *clk*、写使能信号 *w_en*；输出则包含两个读数据端口。寄存器堆的读写受 *w_en* 控制。寄存器的读由组合逻辑实现（即当前周期内即可读取数据），写由时序逻辑实现（根据时钟沿的选取会有一定的延迟）。

   Tips：寄存器堆的写数据由后续阶段生成，本次实验中不做连接。

5. Imm Generator：立即数生成器，需要实现 1 个输入和 1 个输出。该模块读入指令，根据当前指令的格式生成立即数。

6. Control Unit：控制模块，需要生成三类信号：

   i. ALU 相关信号：包括 alu_src1/2、alu_type，直接输出。

   ii. 寄存器读写地址 (5bit)：rs1_read_addr、rs2_read_addr、reg_write_addr，传递给 Reg File。

   iii. 其他控制信号：branch、branch_type 等，详见表 2，直接输出。

   Tips：*Control Unit* 内涉及的信号很多，大家可以根据自己的思路将其拆分成几个子模块。

3.3 模块组织

|–top.v	设计顶层文件，参照图1将各模块连接。
|—ram.v	指令内存，提前写入指令
|——template/adder.v	模板文件，加法器（加法操作数长度是可变参数）
|——dp_components/pc.v	PC 寄存器
|——dp_components/reg_file.v	register file
|——cp_components/control_unit.v	控制逻辑生成
|——cp_components/alu_control.v	alu 运算符选择信号生成
|——cp_components/imm_gen.v	立即数生成

表 1: 模块间的包含关系及推荐的组织格式

各个模块的引用关系如表1所示。为了工程的美观性和可读性，我们建议大家不要把所有的 Verilog 文件都放到项目文件夹下，最好按模块的包含层次把文件组织到子文件夹中。表1提供了一种文件夹组织方案，供大家参考。（我们不强制大家这样组织，大家可以按照自己的习惯组织自己的工程）

3.4 输出信号设计

signal	meaning	signal	meaning
write_data	32bit，寄存器写入数据（模块输入）	inst	32bit，指令内容（模块输入）
rs1/2_data	32bit，rs1/2 数据 (Reg File 输出)	imm	32bit，生成立即数 (imm_gen 输出)
branch	是否为 branch 指令	alu_src1/2	1bit，alu 操作数来源
jal	是否为 jal 指令	alu_type	4bit，运算符选择
jalr	是否为 jalr 指令	reg_src	2bit，写寄存器的数据来源
mem_read	是否进行内存读	branch_type	3bit，branch 判断方式
store_type	3bit，store 的方式	load_type	3bit，load 的方式
mem_write	是否进行内存写		

表 2: ID 模块的输出信号含义

ID 模块涉及的信号及含义如表2所示。其中，上半部分为需要输出的数据信号，下半部分为 Control Unit 需要输出的控制信号。表中标注“是否”字样的代表布尔类型，1bit。

Tips: alu_src1/2 用于控制 alu 操作数来源，可以思考一下：除了 rs_data，还有什么可能会被作为 alu 的操作数？对于每一类指令，alu 的操作数分别对应什么？

此外，本次 Lab 不对 branch_type、store_type、load_type 这三个信号进行测试。在代码实现上，这三个信号由于实现方式的不同，输出的方式也可能有所差异。我们在这里列举出来是考虑到后续 CPU 的完整实现中会用到这些信号。

4 实验原理

4.1 取指阶段原理

PC 为 32bit(1 word) 的寄存器，其存放当前周期需要执行的指令的地址。默认情况下（不产生分支或跳转），当前指令执行完毕后，PC 会通过累加指向下一条指令的地址。RISCV 架构的地址以字节为单位，每条指令的长度为 4 字节，故使用 PC+4 来获取下一条指令。PC 寄存器输出的指令地址传入指令存储器，即可取出相应的指令。

4.2 译码阶段原理

如图2所示，32 位 RISCV 指令在不同类型指令中分别有不同结构。但 [6:0] 表示的 opcode，以及 [14:12] 表示的 funct3，为译码阶段明确指令控制信号的主要字段。译码模块就是根据这些字段区分指令类型，并解析指令内容，生成控制逻辑、立即数、寄存器地址等。详细的指令解析请参考课件和RISCV 官方手册的”RV32I Base Integer Instruction Set, Version 2.1” 部分。

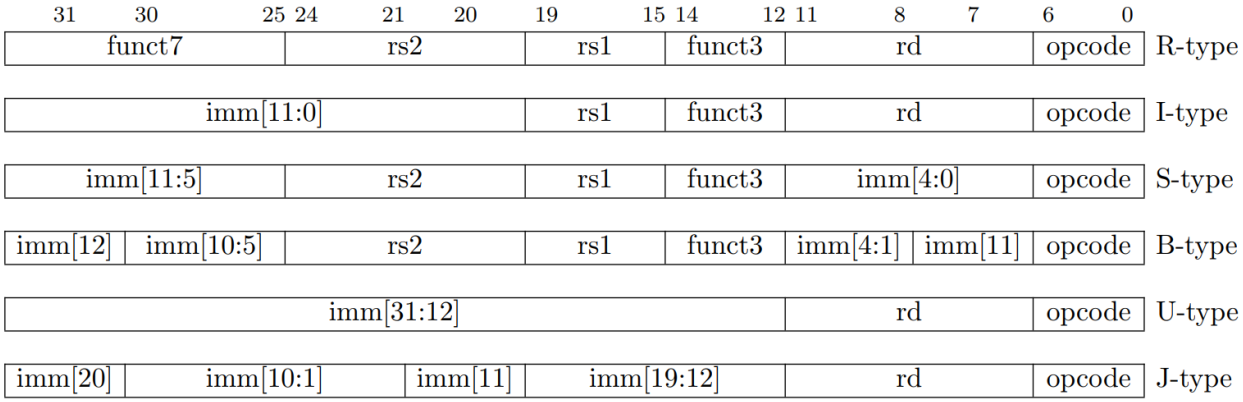

图 2: RISC-V base instruction formats

5 测试与评分

请在本地调试完毕后，再将代码提交到测试平台，并查看测试平台中显示的分数。

5.1 tb 验证思路

由于本次 lab 以取指译码为目标，并不涉及完整的指令执行，因此 tb 文件主要考察以下几个方面（详见 tb 文件）。

1. 能正确从 RAM 中读出指令内容并分解。
2. 如果涉及立即数，能正确生成立即数——检查模块 IMM Generator。
3. 如果指令涉及 ALU，能明确 ALU 需要进行哪一种计算——检查模块 ALU Control。
4. 能正确生成各种控制信号——检查模块 Control Unit。
5. alu_src1/2 的值是由大家自行决定的，为了验证这两个信号功能上的正确性，额外增加了两个输出（这两个输出仅用作本次 lab 的验证，不属于该模块真正的输出 port。），即根据这两个信号选择生成的 alu_op1/2，这两个信号会在后续的执行阶段被送往 ALU 进行运算。

5.2 平台使用

1. 课程平台地址：微处理器设计与智能芯片课程实践
2. 请按规定字段进行注册，并牢记自己的密码。
3. 从测试平台上下载的代码包包括模板文件和简化的测试文件，请先仔细阅读 README，其中提到了你至少应该上传哪些文件，可以自行定义其他辅助模块一并压缩上传。
4. 测试平台会用自己的测试用例覆盖提供给同学们的用例，可以不再上传与测试相关的文件。
5. 请珍惜实验室服务器资源，切勿恶意提交代码。
6. 与平台、实验有关的任何问题可以与助教联系。

7. 欢迎大家提出有关平台、实验的各种建议，让本课程和配套实习越来越好。

5.3 评分规则

1. 能通过编译不报错，获得一个绿色的PASS和 1 分。这主要是为了帮大家明确自己正确提交了文件并通过了编译，所以请不要获得一个 PASS 之后就结束，务必根据 performance 分数确认自己的完成度。

2. 平台会进行一系列的测试输入与输出，每一段输出对应 Lab 的一个功能。

注：平台评分主要作为对你完成度的提示，并不等于你最终课程实习部分的分数。

5.4 实验报告

在实验的同时和实验之后撰写实验报告是非常良好的习惯。这不仅有助于提高实验效率，理清实验思路，帮助 debug，也有助于记录实验过程和结果，还可以和其他同学交流和分享。因此，我们需要大家在 lab 完成过后为我们提交一个实验报告，实验报告有以下几点注意事项：

1. 包含你的设计思路，可以包括相应的模块组织层次、一些不同于助教提示的地方等。

2. 你 debug 的过程，或崩溃或大起大落或柳暗花明的心路历程等。

3. 最终用于 Lab 评分的那次提交：包括代码包、分数截图、提交时平台为本次提交生成的 ID。

4. 实验报告不做字数要求，体现思考和实验过程即可（反卷第一名）。即使最后有 bug 没解决，没有以满分通过平台测试，讲一讲你的 debug 过程与思考也会获得一些分数的补偿：）

6 提示与帮助

根据大家的反馈后续更新，更新后我会在群里 at 大家。

1. 好多同学在 iverilog 的使用上有一些疑问，尤其是头文件的重复引用情况比较严重，因此我们提供了两个脚本文件：run_tb.bat（Windows 系统下跑这个）和 run_tb.sh（Linux 系统下跑这个）。

2. 好多同学在使用 define 文件时有一些疑问，考虑到代码的规范性、降低大家的难度，我们给出了我们认为会用到的宏定义，大家也可以自行删改。