5-stage-pipeline-cpu

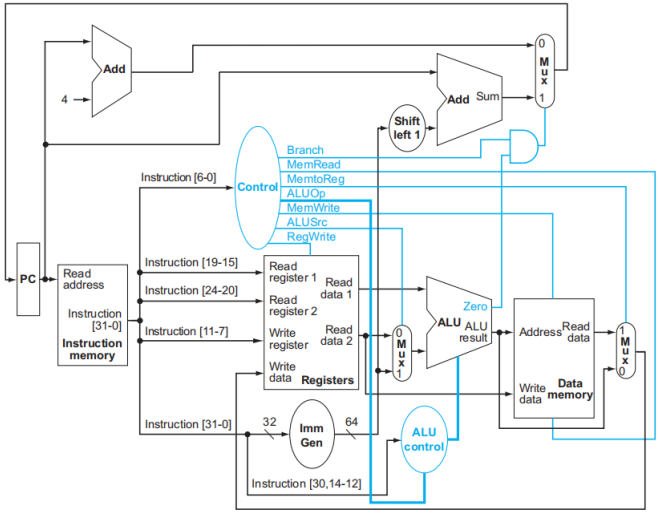
1　功能汇总

1.1　目前进度

写了一个 32 位 5 段流水的 cpu，目前已实现 register 的内部上推功能，但 memory 的内部上推和 cpu 停止还没做。

1.2　文件命名

设计文件中器件的命名按照《Computer organization and design》中的方式命名。



1.3 实现函数

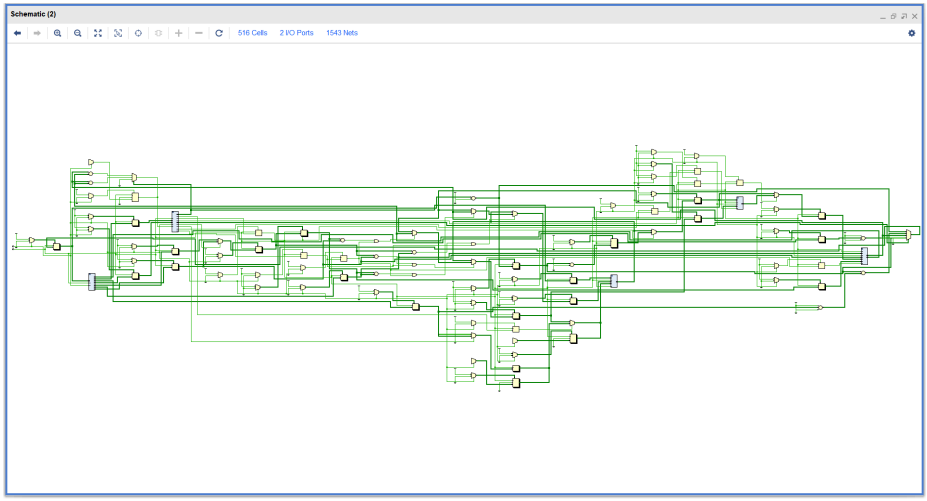
该代码实现了以下 20 个函数：

表 4.4　20 条 MIPS 整数指令

指令	[31:26]	[25:21]	[20:16]	[15:11]	[10:6]	[5:0]	意 义
add	000000	rs	rt	rd	00000	100000	寄存器加
sub	000000	rs	rt	rd	00000	100010	寄存器减
and	000000	rs	rt	rd	00000	100100	寄存器与
or	000000	rs	rt	rd	00000	100101	寄存器或
xor	000000	rs	rt	rd	00000	100110	寄存器异或
sll	000000	00000	rt	rd	sa	000000	左移
srl	000000	00000	rt	rd	sa	000010	逻辑右移
sra	000000	00000	rt	rd	sa	000011	算术右移
jr	000000	rs	00000	00000	00000	001000	寄存器跳转
addi	001000	rs	rt	immediate			立即数加
andi	001100	rs	rt	immediate			立即数与
ori	001101	rs	rt	immediate			立即数或
xori	001110	rs	rt	immediate			立即数异或
lw	100011	rs	rt	offset			取整数数据字
sw	101011	rs	rt	offset			存整数数据字
beq	000100	rs	rt	offset			相等转移
bne	000101	rs	rt	offset			不等转移
lui	001111	00000	rt	immediate			设置高位
j	000010	address					跳转
jal	000011	address					调用

1.4 结构图

该 cpu 的结构图如下：



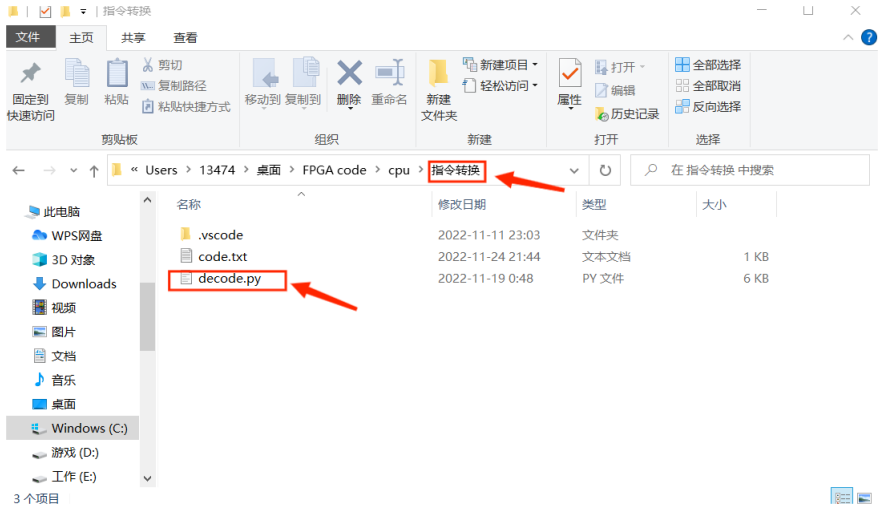
2　操作方法

2.1　代码写入

在 cpu 运行前需要向 instruction memory 中写入 cpu 的软件代码，本模型采用了 python 模拟向 instruction memory 写入代码的过程。

操作步骤如下：

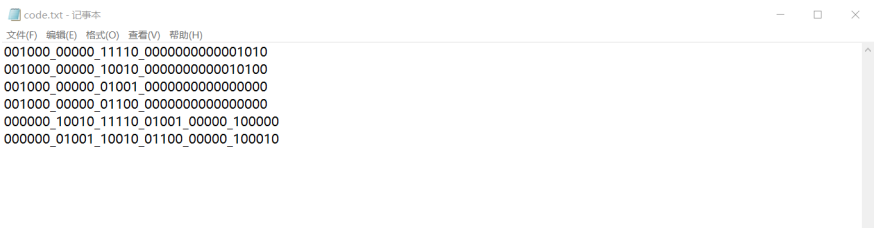
1. 打开“指令转换”文件夹中的”decode.py”



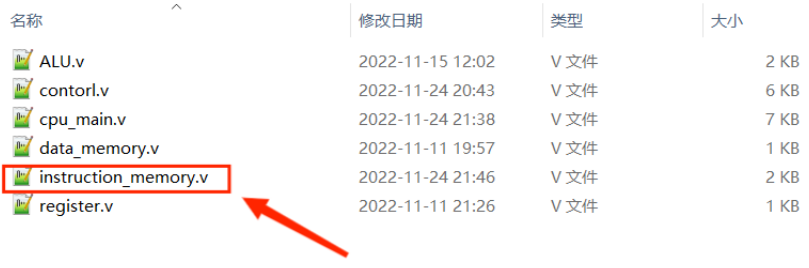
2. 打开”decode.py” 后，在 debug 模式下运行（直接运行好像会导致数据存不进文件），直接输入想要的代码即可。注意，要加入变量（如 x1,x2,x3），应使用 load 函数，所有的变量都需要用 load 函数定义。

```
13474@LAPTOP-RT1BMTBG MINGW64 /c/Users/13474/Desktop/FPGA code/cpu/指令转换 (master)
$  cd c:\\Users\\13474\\Desktop\\FPGA\ code\\cpu\\指令转换 ; /usr/bin/env C:\\Users\\13474\\AppData\\Local\\Programs\\Python\\Python311\
\python.exe c:\\Users\\13474\\.vscode\\extensions\\ms-python.python-2022.18.2\\pythonFiles\\lib\\python\\debugpy\\adapter/../..\\debugpy
\\launcher 62287 -- c:\\Users\\13474\\Desktop\\FPGA\ code\\cpu\\指令转换\\decode.py
要存入x数据，使用函数"load x 变量值"
请输入cpu指令，直接回车结束输入：load x1 10
001000_00000_11110_0000000000001010
要存入x数据，使用函数"load x 变量值"
请输入cpu指令，直接回车结束输入：load x2 20
001000_00000_10010_0000000000010100
要存入x数据，使用函数"load x 变量值"
请输入cpu指令，直接回车结束输入：load x3 0
001000_00000_01001_0000000000000000
要存入x数据，使用函数"load x 变量值"
请输入cpu指令，直接回车结束输入：load x4 0
001000_00000_01100_0000000000000000
要存入x数据，使用函数"load x 变量值"
请输入cpu指令，直接回车结束输入：add x3 x2 x1
000000_10010_11110_01001_00000_100000
要存入x数据，使用函数"load x 变量值"
请输入cpu指令，直接回车结束输入：sub x4 x3 x2
000000_01001_10010_01100_00000_100010
要存入x数据，使用函数"load x 变量值"
请输入cpu指令，直接回车结束输入：
```

检查一下数据是否载入，打开”code.txt”，即可看到刚刚生成的代码：

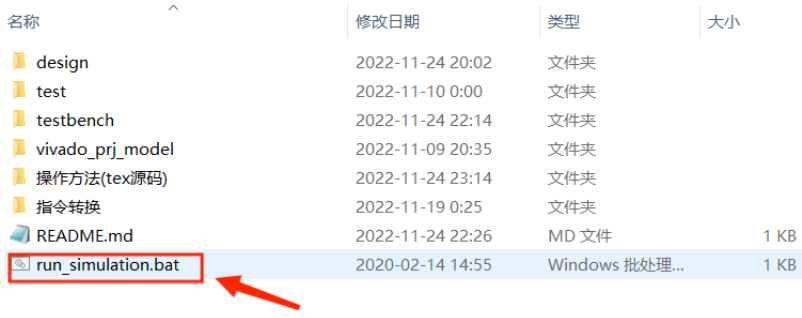


3. 打开”design” 文件夹的”instruction_memory.v”，用复制粘贴的形式将刚刚生成的代码输入对应区域即可。（尝试过 $readmemb，但好像加载不进去）

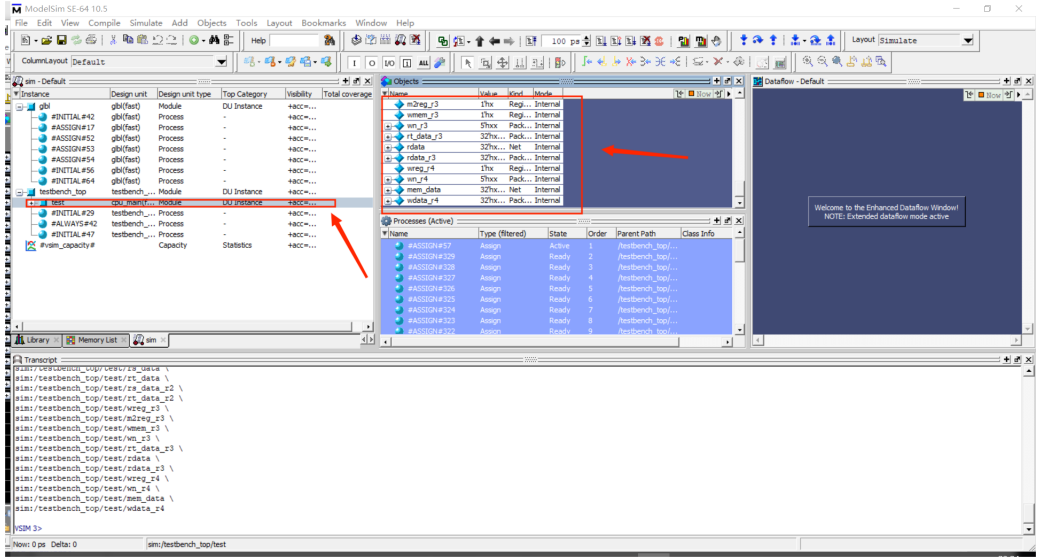


```
always @(*) begin
    if (!rst) begin
        instruction <= 'd0;
        loading <= 'b1;
    for (i=32'b0; i<order; i=i+1'b1)begin
        mem[i] <= 32'b0;
        end
    end
    //写入指令
    else if(loading) begin
        mem[0] <= 32'b001000_00000_00001_0000000000001010;
        mem[1] <= 32'b001000_00000_11110_0000000000010100;
        mem[2] <= 32'b001000_00000_00111_0000000000000000;
        mem[3] <= 32'b001000_00000_10110_0000000000000000;
        mem[4] <= 32'b000000_11110_00001_00111_00000_100000;
        mem[5] <= 32'b000000_00111_11110_10110_00000_100010;
        mem[6] <= 32'b101011_10110_10110_0000000000000000;
        mem[7] <= 32'b000000_00000_00000_00000_00000_000000;
        mem[8] <= 32'b000000_00000_00000_00000_00000_000000;
        mem[9] <= 32'b000000_00000_00000_00000_00000_000000;
        mem[10] <= 32'b000000_00000_00000_00000_00000_000000;
        mem[11] <= 32'b000000_00000_00000_00000_00000_000000;
        mem[12] <= 32'b000000_00000_00000_00000_00000_000000;
        loading <= 'b0;
    end

    else if(addr_pc[31:2] < order) begin
        instruction <= mem[addr_pc[31:2]];
    end

    else instruction <= 'd0;


end
//decode
assign op = instruction [31:26];
assign rs = instruction [25:21];
assign rt = instruction [20:16];
assign rd = instruction [15:11];
assign sa = instruction [10:6];
assign func = instruction [5:0];
assign immediate = instruction [15:0];
assign address = instruction [25:0];
```

4. 打开”run_simulation.bat”，按”1” 运行即可。



仿真开始后记得在’test’ 那里把想要的参数加进去。点击’RUN-ALL’ 开始运行就行。



3 个人感想

3.1 项目经历

这个 cpu 现在已经做了一个多月，从 10 月初开始搞的。9 月的时候我连 verilog 是啥都不知道，搞了两个多月可以搞出一个能跑的 cpu 还是很高兴的。（虽然搞的不咋滴）

本来可以搞的更快一点的，但是考试还是有一点多，然后又要学英语，又要给实验室画一些 PCB，每天弄这个的时间不是很多，每天大概有两个小时吧。

9 月份的时候学了一下啥叫 verilog，写了几个小程序就开始搞这个了，还是有一点难的，一开始找资料整了挺久。

后面有空的话会慢慢完善这个东西，不过期末考又差不多到了，不知道有没有空搞。