实验三 单周期 CPU

noname

2020 年 5 月 21 日

1 实验目的

1. 理解计算机硬件的基本组成、结构和工作原理
2. 掌握数字系统的设计和调试方法
3. 熟练掌握数据通路和控制器的设计和描述方法

2 逻辑设计

2.1 单周期 CPU(Single CPU)

待设计的单周期 CPU 可以执行如下 6 条指令:

add: $rd \leftarrow rs + rt$; $op = 000000, funct = 100000$

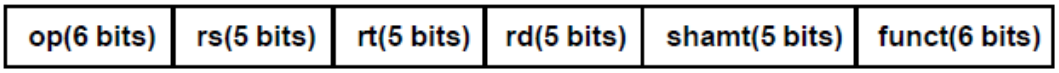

图 1: R-type 指令

addi: $rt \leftarrow rs + imm$; $op = 001000$
lw: $rt \leftarrow M(rs + addr)$; $op = 100011$
sw: $M(rs + addr) \leftarrow rt$; $op = 101011$
beq: $if(rs = rt)\ then\ pc \leftarrow pc + 4 + addr << 2$
$else\ pc \leftarrow pc + 4$; $op = 000100$

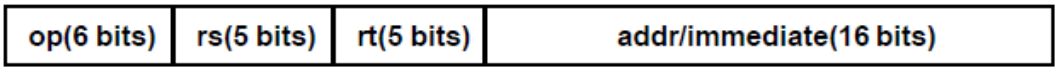

图 2: I-type 指令

j: $pc \leftarrow (pc + 4)[31 : 28]|(add << 2)[27 : 0]$; $op = 000010$

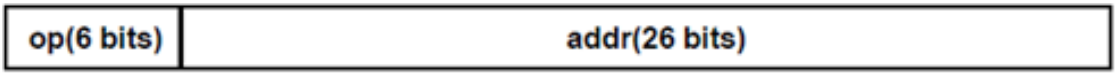

图 3: J-type 指令

分析以上待实现指令的功能，设计 CPU 的数据通路和控制单元 (橙色部分) 如图 4 所示
其中 ALU 和寄存器堆利用实验 1 和实验 2 设计的模块来实现
指令存储器 ROM 和数据存储器 RAM 均采用 IP 例化实现, 容量为 256 x 32 位的分布式存储器
实现时稍微改动了一下数据通路:

i. 添加了指令分割模块用来分割指令，便于表示

ii. 去掉了 ALUcontrol 模块，因为要求实现的 R 类型指令仅有 add 一条

iii. 同时去掉了 MemRead 控制信号：因为使用的是分布式 RAM, 不需要读使能

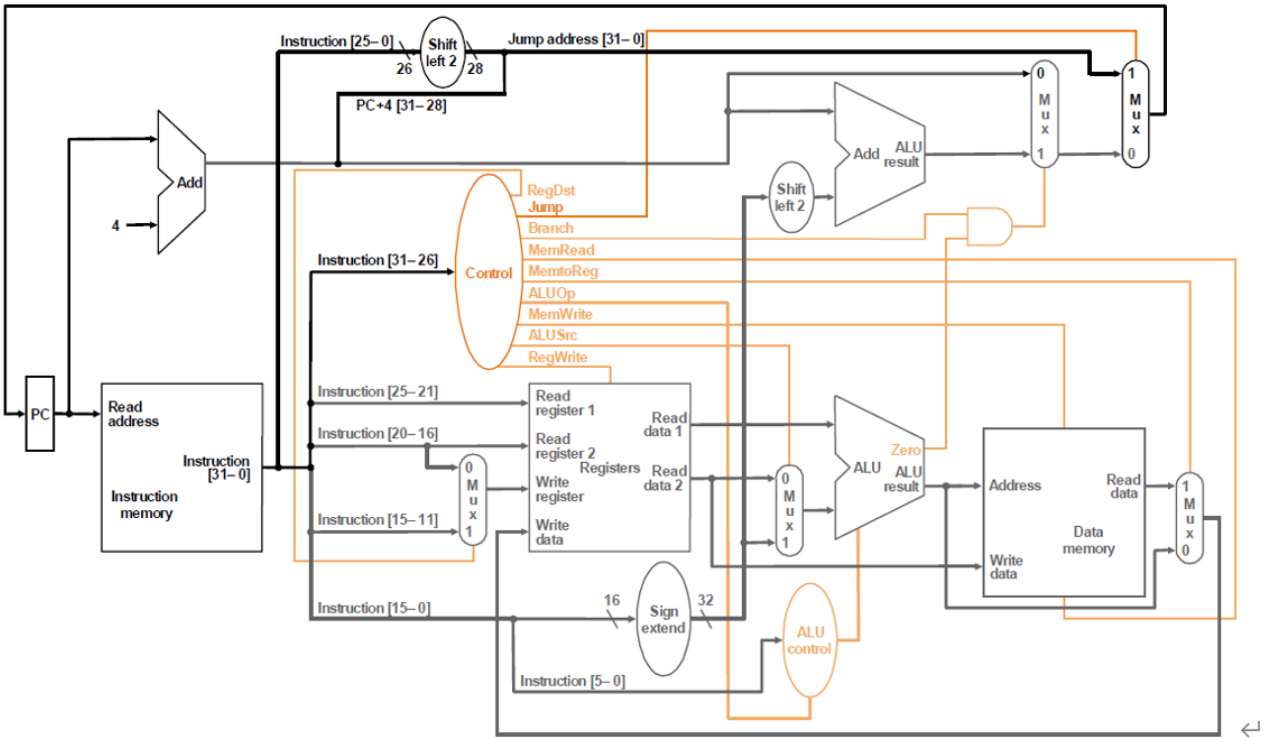

图 4: 逻辑框图

在此仅介绍各个模块功能，代码在附件内展示:

1.SingleCPU: 主要负责连接各个模块成为一个完整的 CPU 以及接收 CLK 和 RST 信号任务
由于没有 Mux 模块，所以 SingleCPU 内部也实现了多选器的功能用来选择各模块的输入信号

2.InsCut: 指令分割，将 32 位的指令分割成 op,rs,rt,rd,imm,addr,func 等，便于使用

3.ControlUnit: 控制单元 (译码模块), 输入 opcode，输出各个控制信号控制各个 CPU 单元来执行不同的指令

4.PCAdd: 求 NextPC

5.PC: 将求得的 NextPC 每个时钟周期更新为当前 PC

6.RegFile: Lab2 例化的寄存器堆 (两异步读，一同步写)，用来存取数据，后半个周期写，保证数据的正确性
且增加限制条件使得 0 号寄存器值恒为 0，满足仿真波形的要求 **7.ALU:**Lab1 例化的 ALU，没什么好说的

8.InsMem: 分布式 ROM，用来读取指令

9.DataMem: 分布式 RAM，用来读写数据 (异步读且无需读使能，无需 MemRead 信号)

2.1.1 单周期 CPU 仿真结果

具体仿真结果的正确性在录制视频内已经讲过了，可以看出是按照汇编文件逐条执行的，且执行结果均正确
这里只放出仿真波形并进行简要分析

根据助教给出的 asm 文件可知，addi、add 与 lw 均正确运行，PC 和 NextPC 的值也说明正常执行

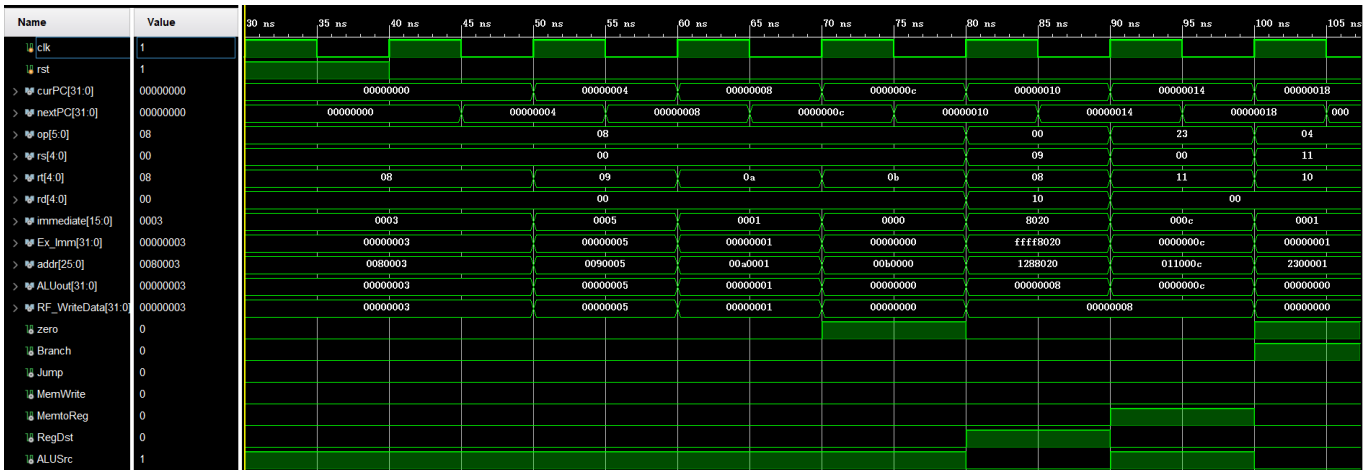

图 5: 单周期 CPU1

之后是 beq 指令 (curPC = 18), 由 zero 和 branch 的值可知判定成功，NextPC 也说明跳转成功，beq 正常执行

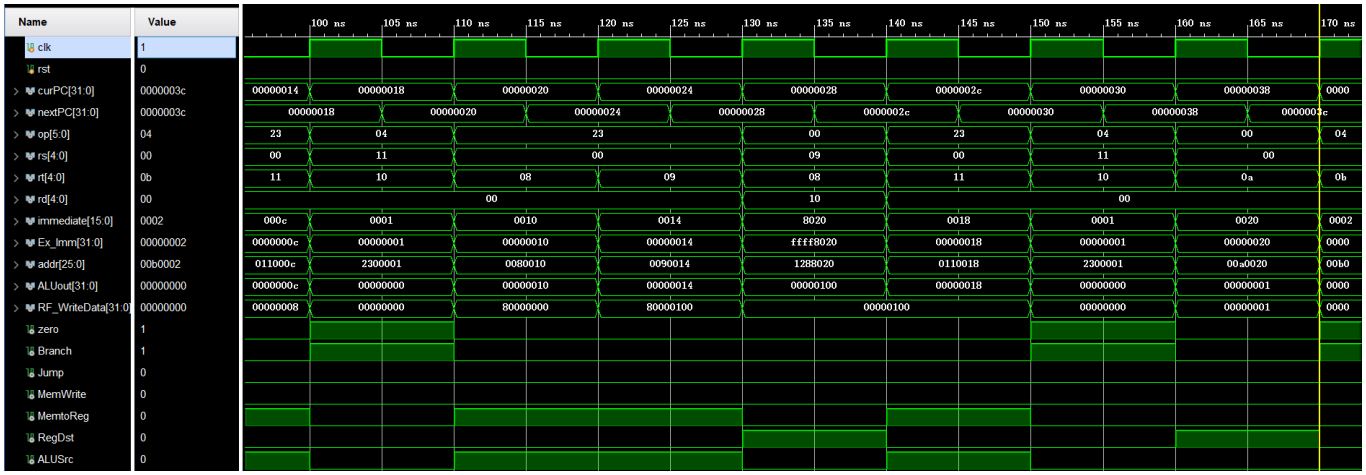

图 6: 单周期 CPU2

按照 asm 文件继续执行，最后 curPC 在 48 与 4c 间循环 Jump 执行，且 0x08 处值为 1，说明 CPU 逻辑正确，Jump 也正常执行

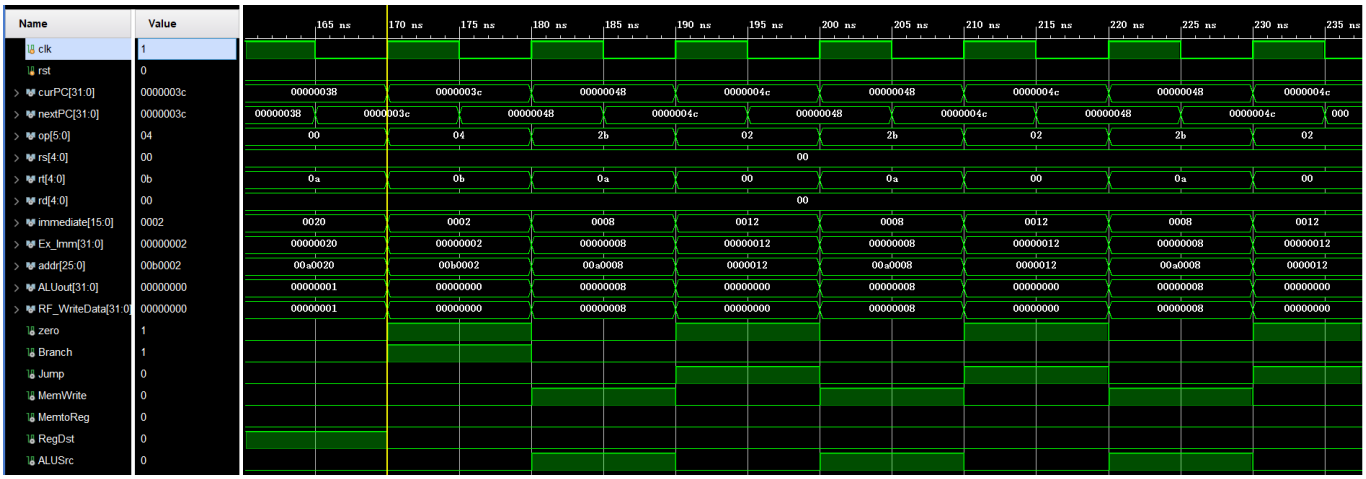

图 7: 单周期 CPU3

2.2 调试单元 (Debug Unit)

为了方便下载调试，设计一个调试单元 DBU，该单元可以用于控制 CPU 的运行方式，显示运行过程的中间状态和最终运行结果。DBU 的端口与 CPU 以及 FPGA 开发板外设（拨动/按钮开关、LED 指示灯、7-段数码管）的连接如图 -3 所示。为了 DBU 在不影响 CPU 运行的情况下，随时监视 CPU 运行过程中寄存器堆和数据存储器的内容，可以为寄存器堆和数据存储器增加 1 个用于调试的读端口

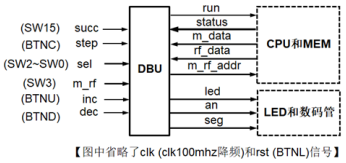

图 8: DBU 逻辑图

控制 CPU 运行方式

succ = 1: 控制 CPU 连续执行指令,run = 1(一直维持)

succ = 0: 控制 CPU 执行一条指令, 每按动 step 一次,run 输出维持一个时钟周期的脉冲

sel = 0: 查看 CPU 运行结果 (存储器或者寄存器堆内容)

m_rf: 1, 查看存储器 (MEM)　0, 查看寄存器堆 (RF)

m_rf_addr:MEM/RF 的调试读口地址 (字地址), 复位时为零

inc/dec:m_rf_addr 加 1 或减 1

rf_data/m_data: 从 RF/MEM 读取的数据字

16 个 LED 指示灯显示 m_rf_addr

8 个数码管显示 rf_data/m_data

sel = 1-7: 查看 CPU 运行状态 (status)

12 个 LED 指示灯 (SW11 SW0) 依次显示控制器的控制信号 (Jump, Branch, RegDst,

RegWrite, MemRead, MemtoReg, MemWrite, ALUOp, ALUSrc) 和 ALUZero, 其中 ALUOp 为 3 位
8 个数码管显示由 sel 选择的一个 32 位数据

sel = 1：pc_in, PC 的输入数据

sel = 2：pc_out, PC 的输出数据

sel = 3：instr, 指令存储器的输出数据

sel = 4：rf_rd1, 寄存器堆读口 1 的输出数据

sel = 5：rf_rd2, 寄存器堆读口 2 的输出数据

sel = 6：alu_y, ALU 的运算结果

sel = 7：m_rd, 数据存储器的输出数据

首先对单周期 CPU 的模块进行修改以供 DBU 模块使用

1.SingleCPU: 增加了 DBU 内的控制信号输入用来控制 CPU 的运行

2.RegFile: 增加了一个异步读端口，用来读出 m_rf_addr 的内容

3.DataMem: 同样的，例化一个双端口 RAM，用来读出 m_rf_addr 的内容

4.PC:PC 模块增加控制信号 PCWre(即 run 信号)，用来判断是否更新 PC

5.DBU: 接收输入信号并传递给 SingleCPU, 输出为 LED 和 SW，用来查看运行状态

6.EDG: 增加取边沿电路，使得一次输入仅带来一次改变 (去抖动烧写板子时用，这里不表)

2.2.1 DBU 仿真结果

放出仿真波形并进行简要分析

首先 succ=1, 连续执行指令, 并令 m_rf=1, 读出 DataMem 对应地址数据
与 Data.coe 文件中对应地址数据相同 (未涉及 sw 指令的地址)
之后令 m_rf=0, 读出 Reg File 内容，由单周期 CPU 的运行结果可知，读出数据正确

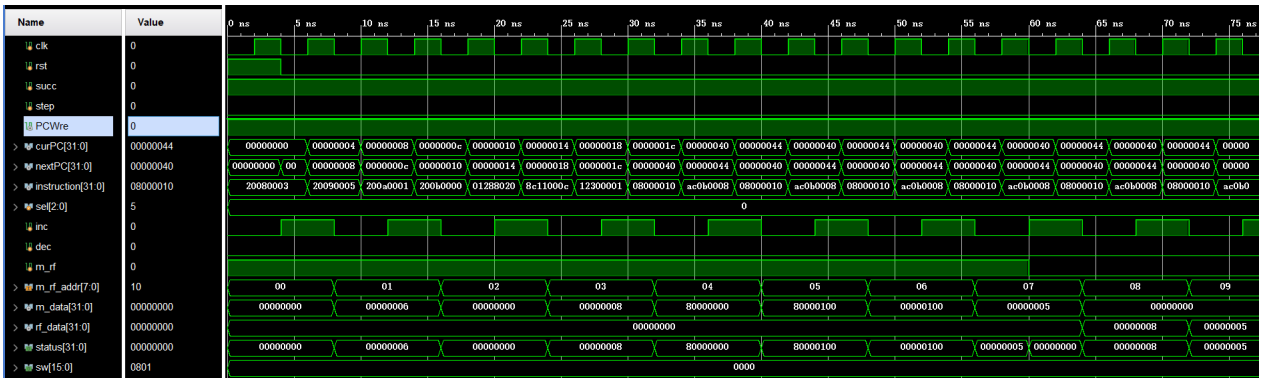

图 9: 调试单元 DBU1

之后令 succ = 0，可以看出，PC 不再更新且读出数据也正确

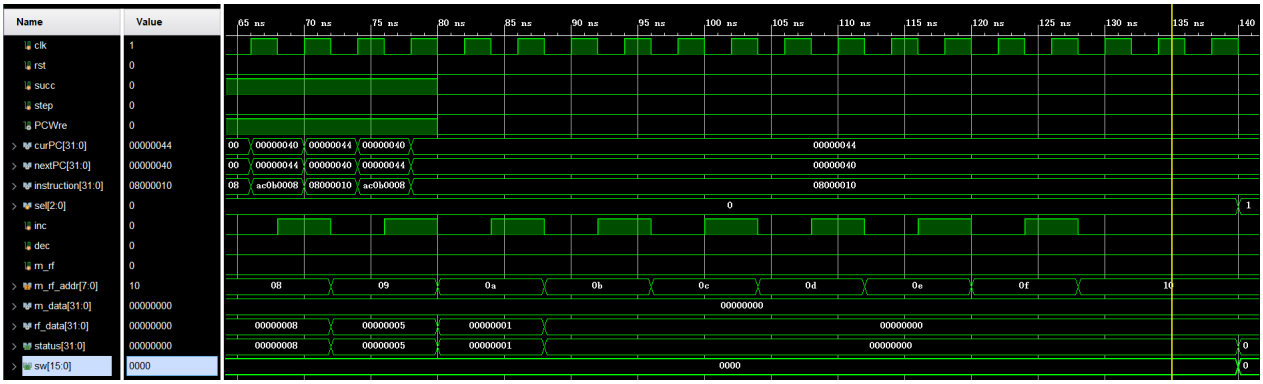

图 10: 调试单元 DBU2

继续执行, 给出 2 个 step 信号，可看出时钟更新。并且令 sel = 1,2,3,4,5 检查 status 和 sw 值，均正确。此时 m_rf_addr 不更新，可以说明 DBU 设计成功且检验结果均正确

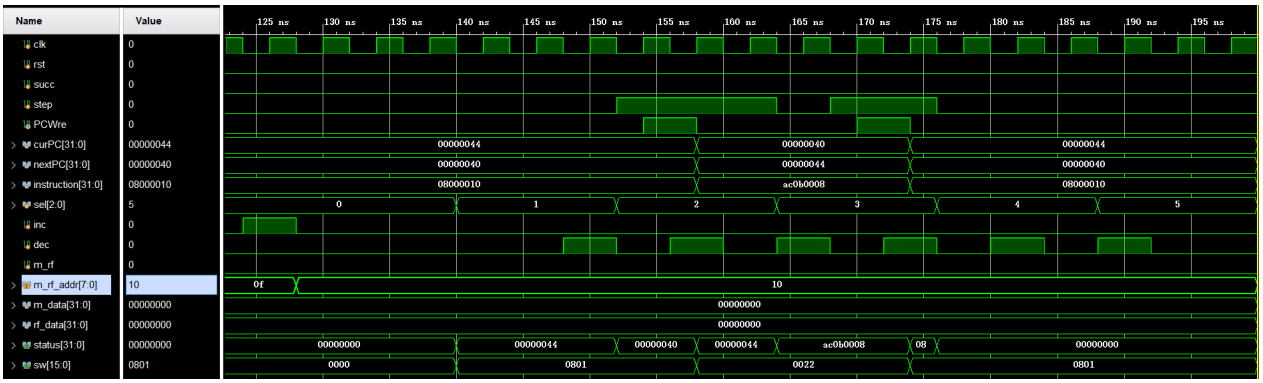

图 11: 调试单元 DBU3

3 实验结果

经仿真检验，单周期 CPU 及其 DebugUnit 均正确实现

4 思考题

修改数据通路和控制器，增加支持如下指令

$accm : rd \leftarrow M(rs) + rt;$ $op = 000000, funct = 101000$

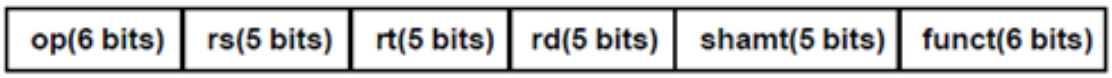

图 12: accm

增加一个 ALUSrcA 前的 Mux 选择器和 ReadData 到 ALUSrcA 多选器的连线
并将原来的 ALUSrc 改名为 ALUSrcB ControlUnit 译码时增加选项，若为 R 类型指令需按照 funct 来译出控制信号
原来的指令保持其他控制信号不变,ALUSrcA = 0 选择寄存器堆读出数据,ALUSrcB 为之前的 ALUSrc

accm 其他信号与 add 的信号相同赋值, 但 ALUSrcA = 1 选择数据存储器读出的 rs 地址数据
accm 指令实现

5 意见与建议

本次实验难度一般，但坑比较多，比如 PC 和 DataMem 读写数据时传入地址的位数
数据通路和指令执行较为复杂，Debug 困难较大

6 附件

本次实验所需提交代码过多, 作为附件上传到 BB 系统上 (仅提交.v 文件和 coe 文件)