南京大学

数电实验十一：

单周期 CPU

课程名称：　数字逻辑与计算机组成实验

姓名：　孙文博

学号：　201830210

班级：　数电一班

邮箱：　201830210@smail.nju.edu.cn

实验时间：　2022.5.20 - 2022.5.25

一、 实验目的

本实验的目标是利用 FPGA 实现 RV32I 指令集中除系统控制指令之外的其余指令。利用单周期方式实现 RV32I 的控制通路及数据通路，并能够顺利通过功能仿真。

二、 实验环境

设计\编译环境：Quartus (Quartus Prime 17.1) Lite Edition

FPGA 芯片：Cyclone II 5CSXFC6D6

三、 实验过程

1. RISC-V 指令集

RISC-V 是由 UC Berkeley 推出的一套开源指令集。该指令集包含一系列的基础指令集和可选扩展指令集。在本实验中我们主要关注其中的 32 位基础指令集 RV32I。RV32I 指令集中包含了 40 条基础指令，涵盖了整数运算、存储器访问、控制转移和系统控制几个大类。RV32I 中的程序计数器 PC 及 32 个通用寄存器均是 32 位长度，访存地址线宽度也是 32 位，RV32I 的指令长度也统一为 32 位，在实现过程中无需支持 16 位的压缩指令格式。

RV32I 的指令编码非常规整，分为六种类型，其中四种类型为基础编码类型，其余两种是变种：

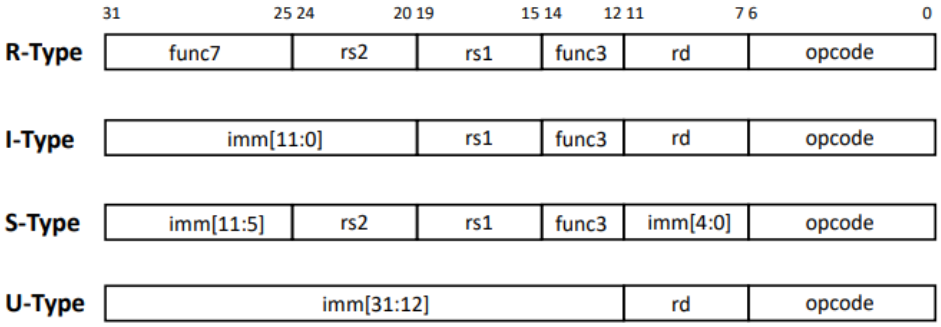


图 11-1: RV32I 指令的四种基本格式

本实验中需要实现的 RV32I 指令含包含以下三类：

- **整数运算指令：**可以是对两个源寄存器操作数，或一个寄存器一个立即数操作数进行计算后，结果送入目的寄存器。运算操作包括带符号数和无符号数的算术运算、移位、逻辑操作和比较后置位等。
- **控制转移指令：**条件分支包括 beq，bne 等等，根据寄存器内容选择是否跳转。无条件跳转指令会将本指令下一条指令地址 PC+4 存入 rd 中供函数返回时使用。
- **存储器访问指令：** 对内存操作是首先寄存器加立即数偏移量，以计算结果为地址读取/写入内存。读写时可以是按 32 位字，16 位半字或 8 位字节来进行读写。读写时区分无符号数和带符号数。

它们的指令码表均在手册中给出，这里不再重复。

2. 单周期 CPU 设计思路

在了解了 RV32I 指令集的指令体系结构（Instruction Set Architecture，ISA）之后，我们将着手设计 CPU 的微架构（micro

architecture）。同样的一套指令体系结构可以用完全不同的微架构来实现。不同的微架构在实现的时候只要保证程序员可见的状态，即 PC、通用寄存器和内存等，在指令执行过程中遵守 ISA 中的规定即可，具体微架构的实现可以自由发挥。

在本实验中，我们首先来实现单周期 CPU 的微架构。所谓单周期 CPU 是指 CPU 在每一个时钟周期中需要完成一条指令的所有操作，即每个时钟周期完成一条指令。每条指令的执行过程一般需要以下几个步骤：

1. **取指令：**使用本周期新的 PC 从指令存储器中取出指令，并将其放入指令寄存器（IR）中；

2. **指令译码：**对取出的指令进行分析，生成本周期执行指令所需的控制信号，并计算下一条指令的地址；

3. **读取操作数：**从寄存器堆中读取寄存器操作数，并完成立即数的生成；

4. **运算：**利用 ALU 对操作数进行必要的运算；

5. **访问内存：**包括读取或写入内存对应地址的内容；

6. **寄存器写回：**将最终结果写回到目的寄存器中；

每条指令执行过程中的以上几个步骤需要 CPU 的控制通路和数据通路配合来完成。其中控制通路主要负责控制信号的生成，通过控制信号来指示数据通路完成具体的数据操作。数据通路是具体完成数据存取、运算的部件。

下图是 RV32I 单周期 CPU 的模块设计示意图，接着我们针对

该 CPU 的控制通路和数据通路分别进行分析。在实现了每个部分的具体内容之后可以将多个模块合并到一起，在头歌平台上进行检测。

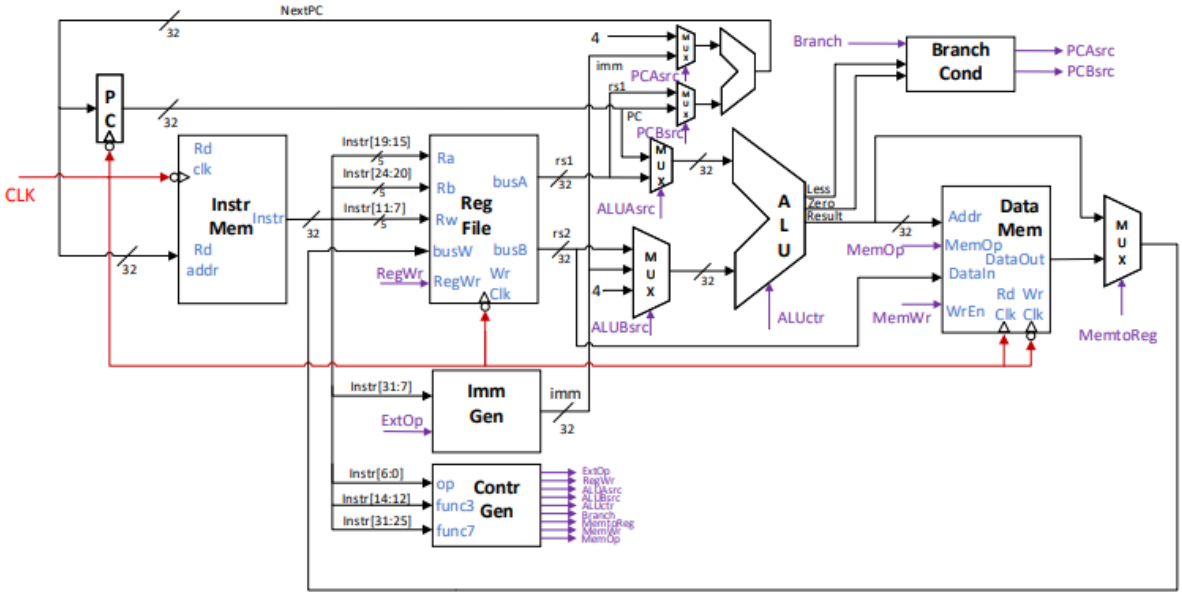


图 11-2: 单周期 CPU 的电路图

3. 控制通路——指令存储器和 PC 生成部分

程序计数器 PC 控制了整个 CPU 指令执行的顺序。在顺序执行的条件下，下一周期的 PC 为本周期 PC+4。如果发生跳转，PC 将会变成跳转目标地址。本设计中每个时钟周期是以时钟信号 CLK 的下降沿为起点的。在上一周期结束前，利用组合逻辑电路生成本周期将要执行的指令的地址 NextPC。在时钟下降沿到达时，将 NEXT PC 同时加载到 PC 寄存器和指令存储器的地址缓冲中去，完成本周期指令执行的第一步。而指令存储器部分，由于头歌测试直接给出当前指令编码，因此无需实现指令存储器。

指令部分的实现代码如下：

```
always @(negedge clock)
begin
if(reset)
    PC=0;
else
begin
    PC=((PCAsrc==0?4:imm)+(PCBsrc==0?PC:rs1));
end
end
assign dbgdata=PC;
assign imemclk=~clock;
assign dmemrdclk=clock;
assign dmemwrclk=~clock;
assign dmemwe=MemWr;
assign dmemop=MemOP;
assign dmemdatain=rs2;
assign dmemaddr=aluresult;
assign imemaddr=PC;
endmodule
```

4. 控制通路——控制信号生成

在确定指令类型后，需要生成每个指令对应的控制信号，来控制数据通路部件进行对应的动作。控制信号生产部件 (Control Signal Generator) 是根据 instr 中的操作码 opcode，及 func3 和 func7 来生成对应的控制信号的。其中操作码实际只有高 5 位有用，func7 实际只有次高位有用。控制信号包括：

- **ExtOP**: 宽度为 3bit，选择立即数产生器的输出类型，具体含义参见表 11-9。
- **RegWr**：宽度为 1bit，控制是否对寄存器 rd 进行写回，为 1 时写回寄存器。
- **ALUAsrc**：宽度为 1bit，选择 ALU 输入端 A 的来源。为 0 时选择 rs1，为 1 时选择 PC。
- **ALUBsrc**：宽度为 2bit，选择 ALU 输入端 B 的来源。为 00 时选择 rs2，为 01 时选择 imm(当是立即数移位指令时，只有低 5 位有效)，为 10 时选择常数 4（用于跳转时计算返回地址 PC+4）。
- **ALUctr**：宽度为 4bit，选择 ALU 执行的操作，具体含义参见表 **??**。
- **Branch**：宽度为 3bit，说明分支和跳转的种类，用于生成最终的分支控制信号，含义参见表 11-12。
- **MemtoReg**：宽度为 1bit，选择寄存器 rd 写回数据来源，为 0 时选择 ALU 输出，为 1 时选择数据存储器输出。
- **MemWr**：宽度为 1bit，控制是否对数据存储器进行写入，为 1 时写回存储器。
- **MemOP**：宽度为 3bit，控制数据存储器读写格式，为 010 时为 4 字节读写，为 001 时为 2 字节读写带符号扩展，为 000 时为 1 字节读写带符号扩展，为 101 时为 2 字节读写无符号扩展，为 100 时为 1 字节读写无符号

其中每个信号的含义已经在手册中解释清楚了，这里直接上代码实现：

```
//跳转控制模块(alu标志未输入，跳转控制信号输入,pc选择信号输出)
module branchctr(
    input [2:0] Branch,
    input Less,
    input Zero,
    output reg PCAsrc,
    output reg PCBsrc
);
always @(*)
begin
case(Branch)
3'b000:
begin
    PCAsrc=0;PCBsrc=0;
end
3'b001:
begin
    PCAsrc=1;PCBsrc=0;
end
3'b010:
begin
    PCAsrc=1;PCBsrc=1;
```

```
//控制信号生成模块
module ctr(
    input [6:0] op,
    input [2:0] func3,
    input [6:0] func7,
    output reg [2:0] ExtOP,
    output reg RegWr,
    output reg ALUAsrc,
    output reg [1:0] ALUBsrc,
    output reg [3:0] ALUctr,
    output reg [2:0] Branch,
    output reg MemtoReg,
    output reg MemWr,
    output reg [2:0] MemOP
    );
always @(*)
begin
case(op[6:2])
5'b01101:
begin
    ExtOP=3'b001;RegWr=1;Branch=3'b000;MemtoReg=0;MemWr=0;ALUBsrc=2'b01;
5'b01100:
begin
    RegWr=1;Branch=3'b000;MemtoReg=0;MemWr=0;ALUAsrc=0;ALUBsrc=2'b00;
case(func3)
3'b000:ALUctr=func7[5]==0?4'b0000:4'b1000;
3'b001:ALUctr=4'b0001;
3'b010:ALUctr=4'b0010;
3'b011:ALUctr=4'b1010;
3'b100:ALUctr=4'b0100;
3'b101:ALUctr=func7[5]==0?4'b0101:4'b1101;
3'b110:ALUctr=4'b0110;
3'b111:ALUctr=4'b0111;
endcase
end
5'b11011:
begin
```

5. 数据通路——寄存器堆及 ALU

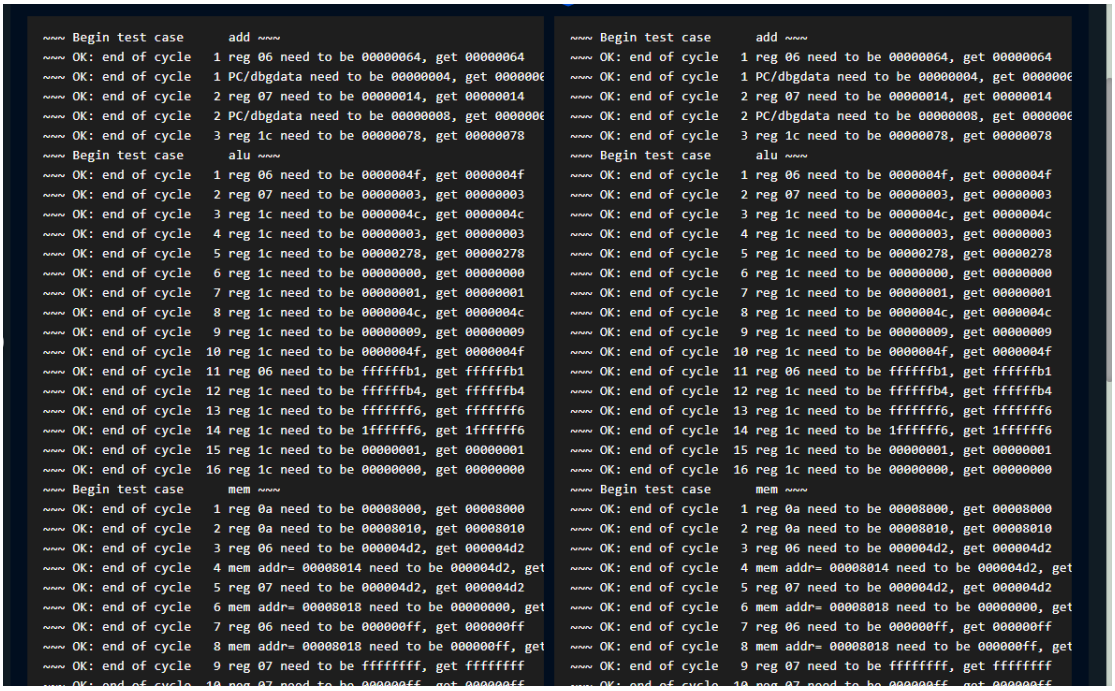
这部分内容在实验十中已经实现（所以说实验十是实验十一的铺

垫），直接搬运过来即可：

```
//寄存器堆模块
module regfile(
    input [4:0] Ra,
    input [4:0] Rb,
    input [4:0] Rw,
    input [31:0] wrdata,
    input RegWr,
    input clk,
    output [31:0] rs1,
    output [31:0] rs2
    );
    reg [31:0] regs[31:0];
module alu(
    input [31:0] dataa,
    input [31:0] datab,
    input [3:0] ALUctr,
    output less,
    output zero,
    output reg [31:0] aluresult
    );
    wire [31:0] F1;
    wire [31:0] F2;
    wire [31:0] F3;
    wire [31:0] F4;
    wire [31:0] F5;
    wire zero1;
    wire zero2;
    wire cout1;
    wire cout2;
    reg less_cpy;
    reg zero_cpy;
```

6. 代码测试部分

这里我们直接在头歌平台上进行检测，没有在本地的 Quartus 软件上进行 testbench 仿真文件检测。



此外还有官方测试集版本：

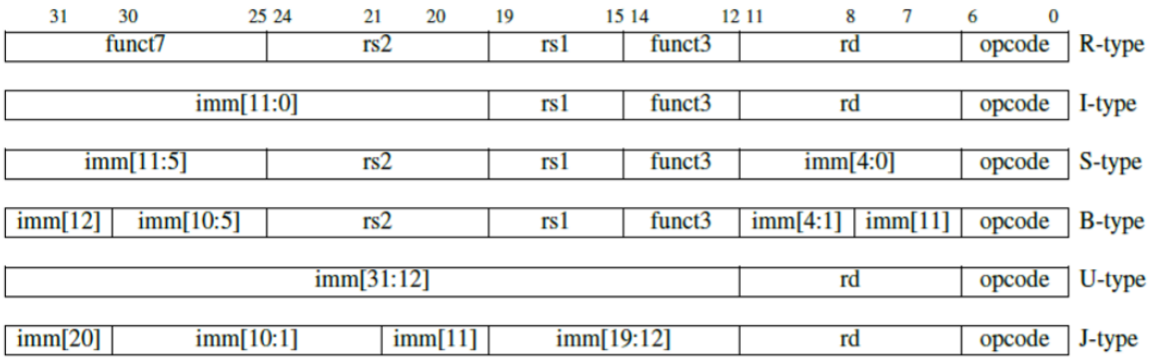
至此，一个完整的单周期 CPU 已经实现！

四、 实验结果

1. 思考题

☞ **思考**: 为什么会有 S-Type/B-Type，U-Type/J-Type 这些不同的立即数编码方案？指令相关的立即数为何在编码时采用这样“奇怪”的 bit 顺序?

RV321 的六种指令格式如下：



其中 B-Type 型和 J-Type 型指令的出现是为了方便获取操作数的符号位，从而进行与之相关的符号拓展/零拓展操作。

☞ **思考:** 为什么我们将起始为 0x80000000 的代码段和数据段地址只取低 18 位来生成代码和数据存储器的初始化文件，我们的 CPU 仍然正确地执行并找到对应的数据?

在第二步操作的分析中可知，对地址取低 18 位，并将地址除以 4（从 byte 编址改成我们存储器中 4 字节编址），对于正常的数据行，awk 会将 token 分成四个一组重新打印，从而得到正确的地址和代码段。

☞ **思考:** 如果数据存储器是用 4 片 8bit 存储器来实现的，如何生成 4 片存储器对应的初始化文件?

不确定是否可以通过 IP 核生成.mif 文件的方法实现。

2. 在线测试

完成单周期 CPU 的实现，并顺利通过了课程在头歌系统上的两个测试文件。

五、 总结与反思

本次实验是实验十的后续，这两个实验最终要实现一个完整的单周期CPU。由于较为抽象且不易验收，因此重点在于头歌的测试部分。在实验手册事无巨细的指导下，我们顺利完成了 CPU 各个部分的代码实现，并综合在一起通过了头歌上的两个测试。不得不再次称赞手册编写者的认真与细致，让我们可以找到自己的 bug 所在，也不得不再次感慨人类智慧的发达，可以发明出 CPU 这种智慧的结晶。

> “陛下，我们把这台计算机命名为‘秦一号’。请看，那里，中心部分，是 *CPU*，是计算机的核心计算元件。由您最精锐的五个军团构成，对照这张图您可以看到里面的加法器、寄存器、堆栈存贮器；外围整齐的部分是内存，构建这部分时我们发现人手不够，好在这部分每个单元的动作最简单，就训练每个士兵拿多种颜色的旗帜，组合起来后，一个人就能同时完成最初二十个人的操作，这就使内存容量达到了运行‘秦 *1.0*’操作系统的最低要求；”
>
> —《三体》，刘慈欣