**单周期型32位CPU（核）设计报告**



团队成员： 杨沁儒 、何荣燊、李永浩

学号： 320220940961、320220940271、320220940

班级： 数据科学四班

**目录**

[一、 实验题目 1](#_Toc166410514)

[二、 实验原理 1](#_Toc166410515)

[三、 指令集 2](#_Toc166410516)

[四、 数据通路 3](#_Toc166410517)

[五、 模块设计 3](#_Toc166410518)

# 实验题目

请参考课程讲述的CPU核结构，设计一个兼容32位MIPS或RISC-V指令集的单周期CPU（核）。

• 基础要求（0~80分）：

• 使用Verilog或其他设计语言（其他设计语言不提供技术支持）• 使用FPGA内部静态存储器（block RAM）作为指令缓存（iCache）和数据缓存（dCache），将机器指令和数据写入coe文件，将coe文件导入iCache（和dCache）（请参考附件所提供的设计模版）• 整体微体系结构采用冯诺伊曼微结构或哈佛微结构均可

• 前仿（功能仿真）成功，并给出性能指标，包括：频率（或者等效频率）、已实现指令的条数和CPI等

• 指令条数不限制，但要求：

• 若采用MIPS指令集，所实现的指令需覆盖R型、I型计算类、I型取数类、I型存数类、I型条件判断类、J型。

• 若采用RISC-V指令集，需覆盖（I型）基本指令集中与MIPS要求对应的指令。

• 若采用自定义指令集和其他指令集，请联系代课老师或研究生助教。

• 可不实现关于中断、异常、控制状态等相关的指令。

• 加分项（共0～20分）

• 使用C语言编写简单程序，覆盖到CPU所支持的分指令，使用MIPS或RISCV交叉编译（mips-linuxgcc或gcc-riscv）为汇编源码，使用MIPS或RISCV模拟器翻译为机器指令并执行，加0~20分。（需课间或其他时间现场演示）

• 提交材料：

• 详细设计文档

• 软硬件设计工程（包括代码）

• 开发时间：

• 三周

# 实验原理

单周期CPU指的是一条指令的执行在一个时钟周期内完成，然后开始下一条指令的执行，即一条指令用一个时钟周期完成。

在cpu处理指令时，需要经过如下五个阶段：

1. 取指令(IF)：根据程序计数器PC中的指令地址，从存储器中取出一条指令，同时，PC根据指令字长度自动递增产生下一条指令所需要的指令地址，但遇到“地址转移”指令时，则控制器把“转移地址”送入PC，当然得到的“地址”需要做些变换才送入PC。
2. 指令译码(ID)：对取指令操作中得到的指令进行分析并译码，确定这条指令需要完成的操作，从而产生相应的操作控制信号，用于驱动执行状态中的各种操作。
3. 指令执行(EXE)：根据指令译码得到的操作控制信号，具体地执行指令动作，然后转移到结果写回状态。
4. 存储器访问(MEM)：所有需要访问存储器的操作都将在这个步骤中执行，该步骤给出存储器的数据地址，把数据写入到存储器中数据地址所指定的存储单元或者从存储器中得到数据地址单元中的数据。
5. 结果写回(WB)：指令执行的结果或者访问存储器中得到的数据写回相应的目的寄存器中。

# 指令集

采用RISC-V指令集，实现了R型、I型、S型、B型、U型、J型指令。RISC-V指令集的所有的指令都是32位长度。

R类型指令：用于寄存器 - 寄存器操作；

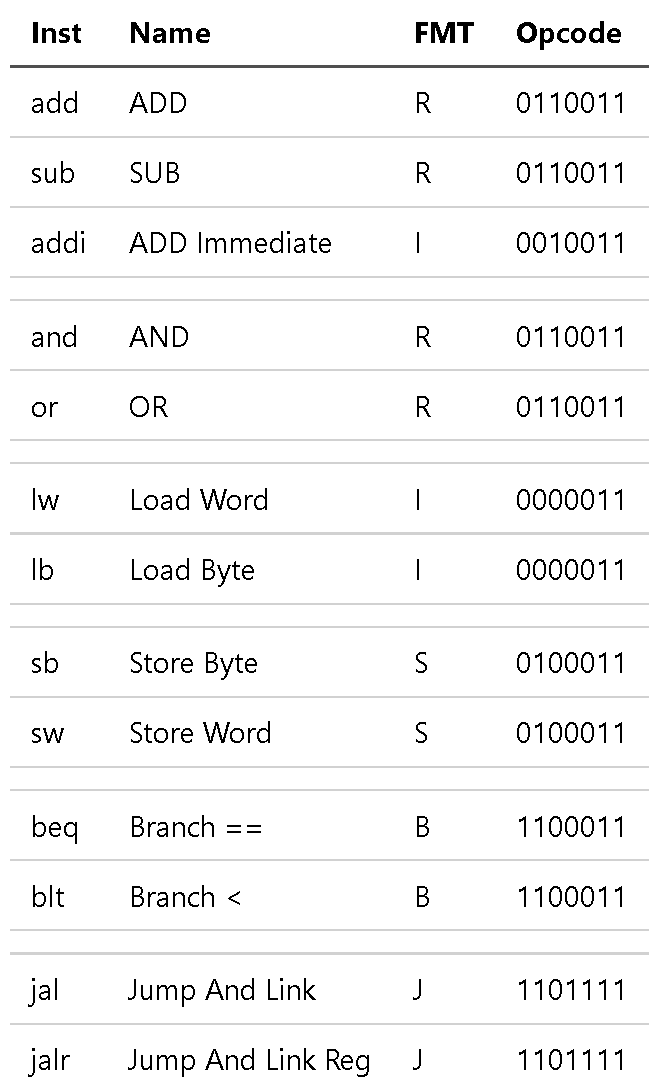
I 类型指令：用于短立即数和访存 load 操作；

S 类型指令：用于访存 store 操作；

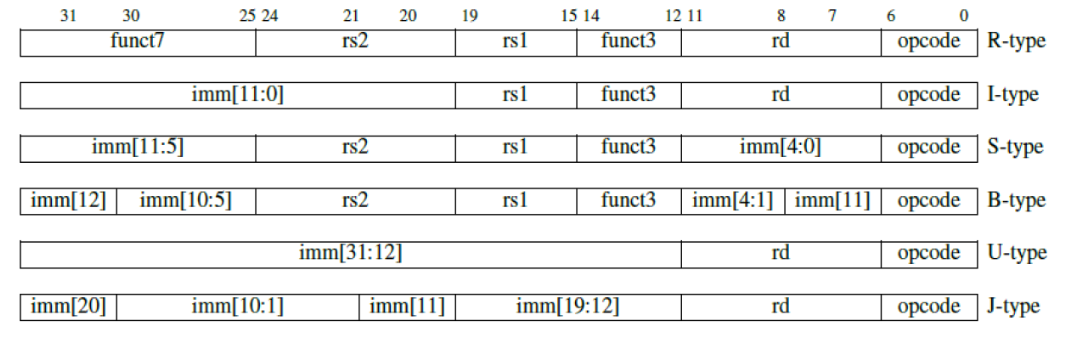
B 类型指令：用于条件跳转操作；

U 类型指令：用于长立即数操作；

J 类型指令：用于无条件操作；

下图将给出实现的指令的类型和操作码：

图表 1

如图为RISC-V六种基本指令格式，RISC-V 指令格式是一个典型的三操作数，7 位操作码的指令格式。

图表 2

其中 opcode 表示 7 位指令操作码，其作用是区分不同的指令；funct3 表示 3 位 的功能码，funct7 表示 7 位的功能码，它们可以辅助区分不同种类的指令；rs1 和 rs2表示两个 5 位的源寄存器；rd 是 5 位的目的寄存器，指令运算的结果就存储 rd 中；imm 代表不同长度的立即数，可直接作为操作数使用。

# 数据通路

# 模块设计

采用哈佛结构设计cpu，包括模块：RegFile、InstrMem、DataMem、ID、PC、ALU、ControlUnit、defines。

* RegFile：寄存器模块，用于存储、读写指令执行过程中的数据和中间结果；
* InstrMem：指令内存模块，它接收指令读取地址作为输入，并输出对应地址处的指令数据。指令存储器用于存储程序的指令序列；
* DataMem：数据内存模块；
* ID：指令解码器（Instruction Decoder）模块，用于从指令中解析出各种控制信号和操作数地址；
* PC：程序计数器模块，实现程序计数器的更新逻辑，并从指令存储器中读取指令；
* ALU：算术逻辑单元；
* ControlUnit：控制单元模块，协调CPU中的各个部件，控制指令的执行和数据流动；
* defines：预处理定义了一系列常量和指令类型，用于在设计硬件逻辑时方便地引用和识别这些常量。

下图给出各个模块设计的端口、端口类型、端口位宽和功能：



图表 3