单周期MIPS CPU设计

没有使用mooc提供的工具，而是直接根据educoder要求的八条指令进行设计

首先根据educoder提供的数据通路图将CPU中控制器之外的的硬布线进行连接

其次，根据硬布线控制器的要求，需要对用到的八条指令的指令译码逻辑、ALU控制信号逻辑、指令译码对应控制信号的逻辑进行设置，以下有两种方法获取具体的八条指令的func、op字段

一种方法是直接将sort.hex文件中的十六进制数据拷贝到指令寄存器中，通过指令寄存器的反汇编引脚获得八条指令与指令中func、op字段的关系

另一种方法是根据mooc中给定的MIPS指令解说，直接得到八条指令的func、op字段，并自己构造合法的MIPS指令语句，并通过反汇编引脚进行印证

通过上面两种方法，就可以完成控制器中指令译码逻辑，

然后，可以通过依次在指令寄存器中访问这八条指令，根据educoder和mooc中MIPS指令手册中解说的指令的具体含义，在CPU硬布线的基础上给定控制信号，完成指令译码对应控制信号的逻辑的设计，从而完成在不同指令下数据通路的构建

与此同时，在访问不同指令时，也可以根据指令的含义完成ALU控制信号逻辑的设计

最终，控制器得以实现，而硬布线电路已经设计完毕，从而完成了单周期MIPS CPU的设计

在搭建完成后，自己遇到了一些问题（详见电路上的备注），并通过逐条执行MIPS机器指令的方法，理解sort.hex的机器语言逻辑，并根据每条机器指令下数据通路的正确与否修改控制器设置时出现的错误

多周期微程序MIPS CPU设计

首先一个任务是完成MIPS微程序地址转移逻辑的设计，即通过机器指令的译码信号来自动生成执行该指令的微程序的入口地址，该入口地址虽然可以自己设定，但自己选择参考educoer的测试算例，毕竟写出来还是要通过这些算例的评测的，也可以防止设定地址出错而影响后面的实验

仿照mooc中MIPS微程序CPU的控制器之外的硬布线进行连接

将sort.hex数据镜像加载到存储器中，通过微指令自动生成表格，在伪指令存储器的0号单元处存放已经设定好的取指令微指令，观察一个时钟周期后，指令被成功取到IR中，借用单周期MIPS CPU的指令译码逻辑，将指令成功译码，转移地址生成逻辑生成了正确的addi指令地址，因此将指令译码微指令的P置为1，并输入表格自动生成的微指令到微指令存储器中，观察到一个时钟周期后，微指令存储器的访问地址指向正确，由于上一条微指令的其他使能均为0，因此硬布线环节中各寄存器不变

因此，可以以一个时钟周期，也就是一条微指令运行的时间为单位，参照mooc的讲义上设定的微程序控制器的状态图，调整不同指令下，为了构建正确数据通路需要的使能信号，包括ALU控制信号，并将表格中生成的16进制微指令输入到微指令寄存器中，从而逐步完成微指令寄存器的构建

如果程序运行出了问题，可以通过架构中可以设置断点的时钟信号产生器，在educoder上出现问题的地方设置断点，并在断点时刻分析数据通路中数据错误发生在哪里，并修改相应的微指令的使能信号，形成正确的时钟通路

比较特殊的是syscall指令的微程序，它只有一条微指令，并且循环执行，下址字段就是微指令自己的位置，P为0，此时其他使能信号已经无关紧要，因为微程序控制器只会循环执行这一条微指令，甚至连取机器指令微指令都不会再执行，因此微调AluOp让它的使能信号不悬空即可（原始的设定中，Alu\_Control只有三种可能的值，第四种值会令AluOp悬空，需要加以修正）

此外，由于在实验架构的规定中，syscall虽然func为0，但不属于R\_TYPE；同时，如果单纯凭借func是否为0来判断是否为R\_TYPE，就会在syscall机器指令下，同时点亮R\_TYPE和func的指令译码信号，造成微程序入口地址可能发生错误，因此对R\_TYPE的判断逻辑中要排除syscall的情况

同时，还发现，在educoder上面，有的引脚范例输出和实际输出都是大量的0，和本地的测试结果相违背，怀疑是这些位置并没有真正的获取信号，而是用了简单的下拉

多周期硬布线MIPS CPU设计

本次任务中，由于仍旧为多周期和MIPS指令集架构的CPU，控制器之外CPU布线的部分与多周期微程序MIPS CPU一致，因此替换控制器复制电路即可

同时，由于仍旧和多周期微程序MIPS CPU中，采用一样的MIPS指令集架构及相同的译码信号标签，对于指令译码和ALU控制信号的逻辑，也可复用多周期微程序MIPS CPU中，已经完成的实现。

经过mooc的提示，通过多周期硬布线CPU状态机的现态经组合逻辑生成的控制信号，也可以转化为微程序CPU中的微指令产生的控制信号，从而简化工作量

以下部分为本次实验新的内容

状态机的状态转移逻辑可以通过实验包提供的表格，自动生成表达式，从而放到logisim中自动生成电路来完成模块的完善

然后其实实验就做完了