单周期 CPU 设计

目标

设计一个单周期 **32 位** MIPS CPU，依据给定过的指令集，设计核心的控制信号。依据给定的数据通路和控制单元信号进行设计。

指令集

实现 7 条指令子集：**ori，lui，addu，sub，bne，lw，sw**，假设不会溢出

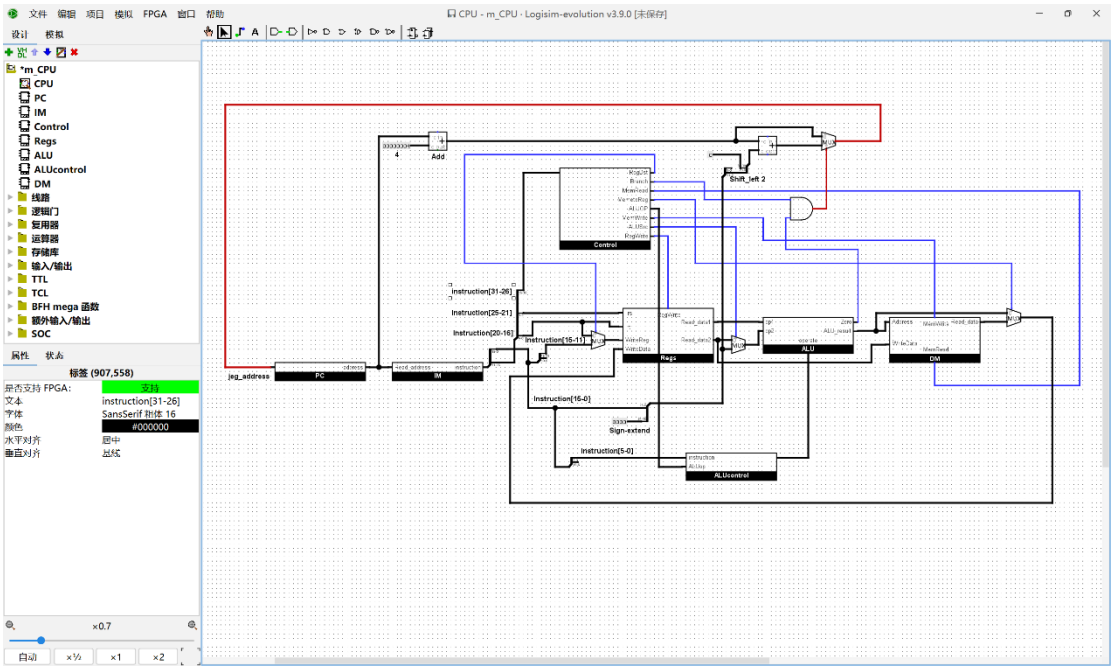
指令的类型

R 型： addu, sub（opcode 为 0）

I 型：ori,lui,lw,sw, bne

设计数据通路

思路

- 依据流水线一般，分为五个阶段分别设计
- 对于一个 MIPS 指令包含如下 5 个处理步骤：
  - **IF**：从指令存储器中读取指令
  - **ID**：指令译码的同时读取寄存器。 MIPS 的指令格式允许同时进行指令译码和读寄存器
  - **EX**：执行操作或计算地址
  - **MEM**：从数据存储器中读取操作数
  - **WB**：将结果写回寄存器
- logisim 实现，但内部未完善
- 

对于控制信号

Control

1. RegDst:写寄存器组的**地址来自 rt 还是 rd**
   1. 当写地址来自 rt,即 **ori, lui,lw** 指令的 rt 时，DegDst 为状态 **0**
   2. 当写地址来自 rd,即 **addu, sub** 指令的 rd 时，DegDst 为状态 **1**

2. RegWrite:是否**往寄存器里写数据**
    1. 往寄存器里写数据时，即 **ori, lui, addu, sub, lw** 为状态 **1**
    2. 不写时，即 **bne, sw** 为 **0**
3. ALUSrc：第二个**操作数来自寄存器还是立即数扩展**
    1. 第二个**操作数来自寄存器**时 ,即 **addu, sub, bne**，为 **0**
    2. 第二个**操作数来自立即数扩展**时，即 **ori, lui, lw, sw**,为 **1**
4. Branch(在表里为 **PCSrc**):是否为**分支指令**
    1. 不是分支指令，为 **0**
    2. 是分支指令 ，即 **bne** 为 **1**
5. MemRead： 是否**读存储器**
    1. 不读，为 **0**
    2. 读寄存器时，即 **lw** ,为 **1**
6. MemWrite :是否**写存储器**
    1. 不写存储器 ，为 **0**
    2. 写存储器时， 即 **sw**，为 **1**
7. MemtoReg:写回寄存器的值**来自 ALU 输出结果还是存储器输出**
    1. **ALU 输出作为结果寄存器输入**，即 **ori, lui, addu ,sub**，为 **0**
    2. **存储器输出作为结果寄存器输入**， 即 **lw, sw** ,为 **1**
8. ALUOP:**控制 ALU 的操作**
    1. 将操作码（opcode)传给 ALUcontrol

所以对于上述指令集

指令	操作码	类型	RegDst	RegWrite	ALUSrc	PCSrc	MemRead	MemWrite	MemtoReg	ALUop
addu	000000	R	1	1	0	0	0	0	1	000000
sub	000000	R	1	1	0	0	0	0	1	000000
ori	001101	I	0	1	1	0	0	0	1	001101
lui	001111	I	0	1	1	0	0	0	1	001111
bne	000101	I	0	0	0	1	0	0	0	000101
lw	100011	I	0	1	1	0	1	0	0	100011
sw	101011	I	0	0	1	0	0	1	0	101011

ALUcontrol

指令	操作码	类型	ALUop	Function 字段	ALU 操作	ALU 控制码
addu	000000	R	000000	100001	加运算	00
sub	000000	R	000000	100010	减运算	01
ori	001101	I	001101	******	或运算	10
lui	001111	I	001111	******	左移运算	11
bne	000101	I	000101	******	减运算	01
lw	100011	I	100011	******	加运算	00
sw	101011	I	101011	******	加运算	00