单周期CPU设计文档

一、设计的目的于说明

使用logisim搭建一个32位8指令单周期CPU；

处理器应支持的指令集为：{addu, subu, ori, lw, sw, beq, lui, nop}；

nop机器码为0x00000000， 即空指令，不进行任何有效行为（修改寄存器等）；

addu,subu不支持溢出；

采用模块化和层次化设计，顶层有效的驱动信号要求包括且仅包括：reset

(clk 使用内置时钟模块)。

二、单周期数据通路设计

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 指令 | Adder | | PC | IM.A | GRF | | | | ALU | | DM | | EXT | Nadd | |
| A | B | RA1 | RA2 | WA | WD | A | B | A | WD | A | B |
| R型指令 | PC | 4 | Adder | PC | Rs | Rt | Rd | ALU | RF.RD1 | RF.RD2 |  |  |  |  |  |
| ORI | PC | 4 | Adder | PC | Rs |  | Rt | ALU | RF.RD1 | EXT |  |  | Imm16 |  |  |
| LUI | PC | 4 | Adder | PC | $0 |  | RS | ALU | 0 | EXT |  |  | Imm16 |  |  |
| LW | PC | 4 | Adder | PC | Rs |  | Rt | DM.RD | RF.RD1 | EXT | ALU |  | Imm16 |  |  |
| SW | PC | 4 | Adder | PC | Rs | Rt |  |  | RF.RD1 | EXT | ALU | RF.RD2 | Imm16 |  |  |
| BEQ | PC | 4 | Adder|Nadd | PC | Rs | Rt |  |  | RF.RD1 | RF.RD2 |  |  | Imm16 | Adder | shift |

三、模块规格

1. IFU（取指令单元）：

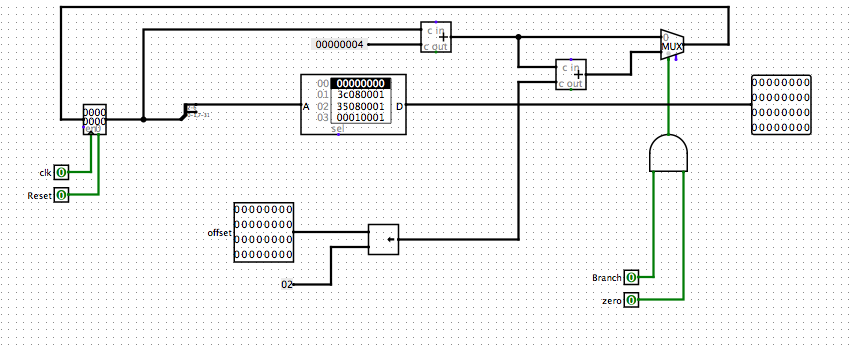
·内部包括 PC（程序计数器）、IM(指令存储器)及相关逻辑；

·PC 用寄存器实现，应具有复位功能；

·起始地址：0x00000000；

·IM用 ROM 实现，容量为 32bit \* 32；

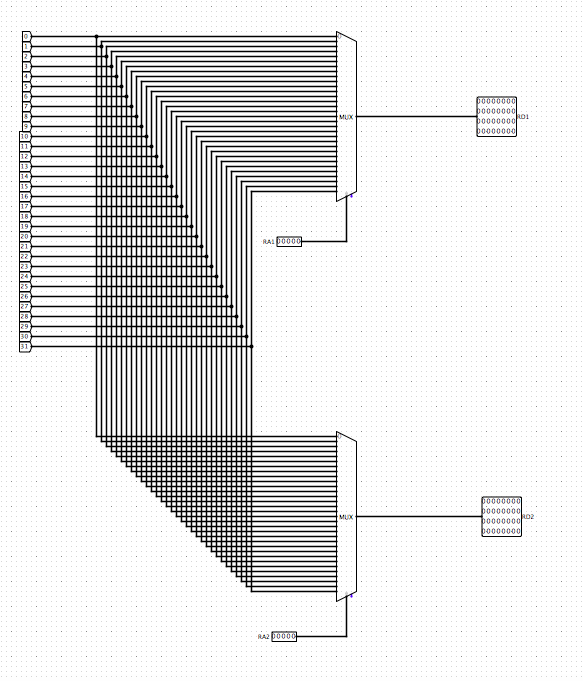
·因 IM 实际地址宽度仅为 5 位，故需要使用恰当的方法将 PC 中储存的地址同 IM 联系起来。



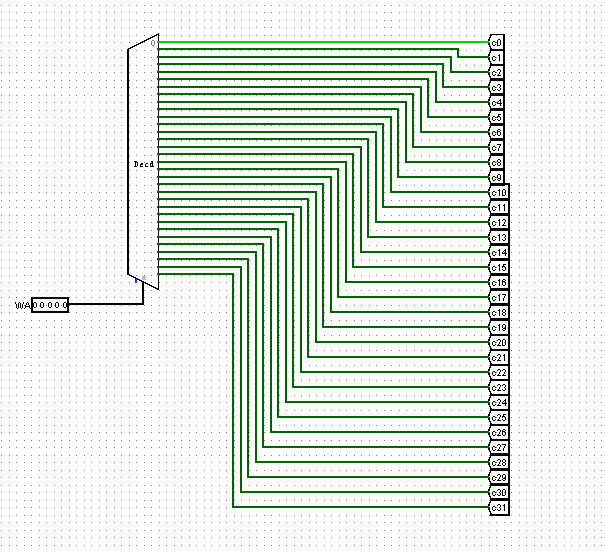
1. GRF（通用寄存器组，也称为寄存器文件、寄存器堆）：

·用具有写使能的寄存器实现，寄存器总数为 32 个；

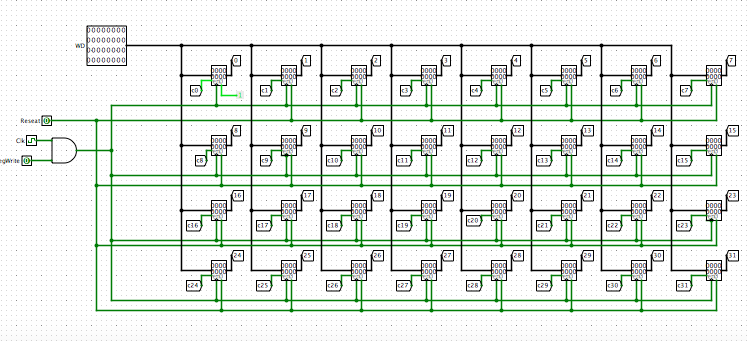
·0 号寄存器的值保持为 0。



说明：根据输入的地址决定所要输出的寄存器



说明：译码器，控制寄存器的使用

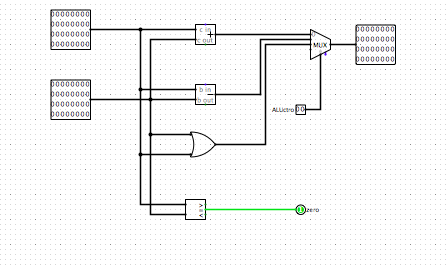


说明：寄存器由时钟信号、读写控制信号、译码器控制，输出由多路选择器控制。

3. ALU（算术逻辑单元）：

·提供 32 位加、减、或运算及大小比较功能；

·可以不支持溢出（不检测溢出）。



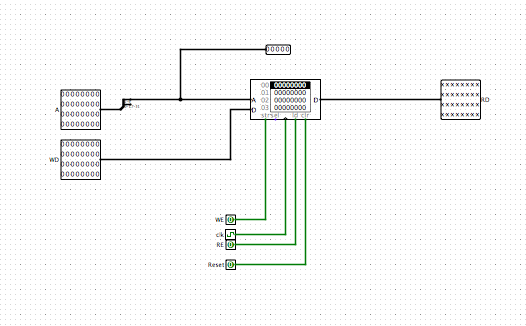
说明：支持加法、减法、32位按位或运算和比较是否相等

4. DM（数据存储器）：

·使用 RAM 实现，容量为 32bit \* 32；

·起始地址：0x00000000；

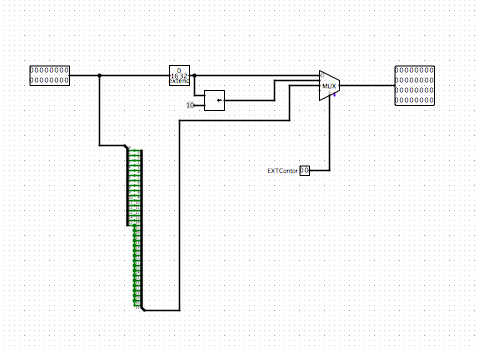
·RAM 应使用双端口模式，即设置 RAM 的 Data Interface 属性为 Separate load and store ports。



说明：有外界提供32位地址，由分线器将第3位到第七位作为RAM的地址。

1. EXT：

·使用 logisim 内置的 Bit Extender。

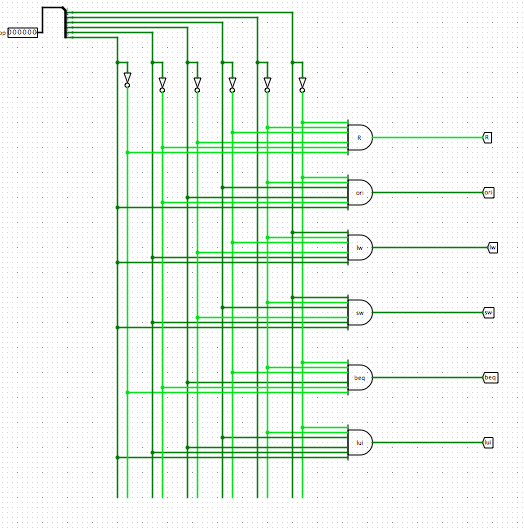


1. 控制器设计

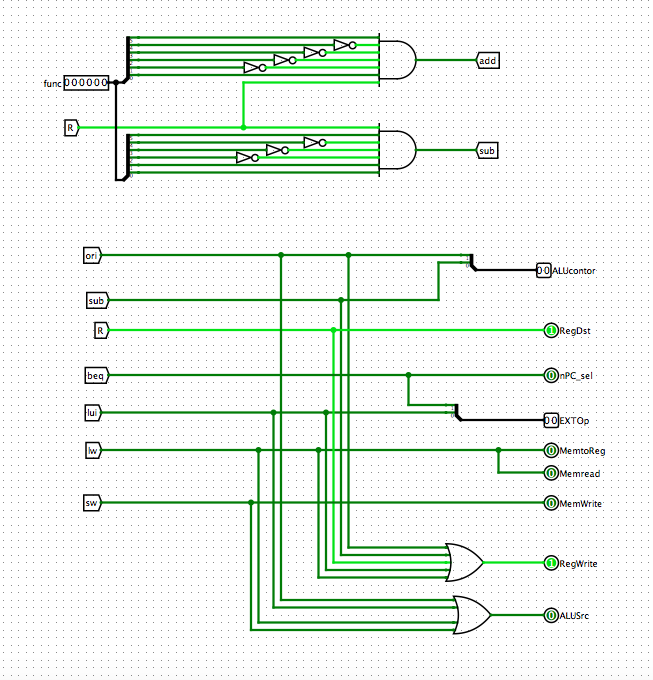
我们把解码逻辑分解为和逻辑和或逻辑两部分：和逻辑的功能是识别，将输入的机器码识别为相应的指令；或逻辑的功能是生成，根据输入的指令的不同，产生不同的控制信号。

真值表：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| func | 100001 | 100011 | n/a | | | | |
| op | 000000 | 000000 | 001101 | 100011 | 101011 | 000100 | 001111 |
|  | addu | subu | ori | lw | sw | beq | lui |
| nPC\_sel | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| MemRead | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| MemtoReg | 0 | 0 | 0 | 1 | x | x | 0 |
| EXTOp | x | x | 0 | 0 | 0 | 0 | 1 |
| RegDst | 1 | 1 | 0 | 0 | x | x | 0 |
| MemWrite | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| RegWrite | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| ALUSrc | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| ALUOP | add | subtract | or | add | add | cmp | add |



说明：将输入的机械码识别为相应的指令。



说明：由不同的指令产生相应的控制信号。