鲸落_falling
码龄3年 暂无认证

12	34	46	14	2万+
原创	粉丝	获赞	评论	访问

304	140	16万+	23万+	
积分	收藏	周排名	总排名	等级

TA的主页 私信 关注

搜博主文章

最新文章

Android笔试准备

LPC2124定时器实现跑马灯

LPC2124定时器实现跑马灯（2）

关于xml的约束文件DTD与Schema

Spring学习笔记1-下载、创建第一个工程

分类专栏

 嵌入式 3篇

 markdown编辑技巧 1篇

 复习

 Flask 1篇

 配置 1篇

 Spring 1篇

归档

2019

10月	7月	6月
1篇	2篇	9篇

热门文章

使用logisim设计简易CPU 11763

RFID的复习博客 3662

基于Flask框架的简单网页开发_2、加入html文件以及css、js文件 2349

LPC2124定时器实现跑马灯 1082

嵌入式复习提纲 816

最新评论

RFID的复习博客
王认真： 这学期江大物联网要考主观题，占60分，同志们觉得能考哪里

使用logisim设计简易CPU
北堪部落： 能不能把circ文件发出来?
嵌入式复习提纲
王认真： 物联网17的马上要考嵌入式了过来请个愿，祝全体同学顺利通过
LPC2124定时器实现跑马灯
m0_46915385： 我进不去中断是什么情况

用logisim设计简易CPU

鲸落_falling 2019-06-21 09:06:20 11779 收藏 83

分类专栏： 计组

声明：
设计图纸来自以下链接的博客，这篇文章是按照自己的想法重构了一下设计思路，写了一些自己的理解。
https://www.cnblogs.com/kingduan/p/4054484.html#_Toc402178283

目录

言

或许我们一直有一种疑惑，计算机是怎样运行起来，他是怎样通过处理不同的任务与命令完
函不轻易可为的复杂运算，这貌似是一件很神奇的事情。
其实很早之前就想进行类似的探究，这次的硬件课设正好是一个契机，从最底层的逻辑单元
对CPU的运行原理进行深度探究。

CPU精简结构

我们在学习计算机组成原理的时候了解过CPU是由**控制单元**和**运算单元**这两个主要的功能模
成，或许细分还有clk时钟、存储器和寄存器组，这些在后文会详细解释，在这里我们只需要
CPU的主要功能就是**运算**，于是我们将CPU的功能精简为只剩下加减运算，这样有助于我们
个过程的演示与理解。
的CPU主要包括以下部分：

控制器（Controller）
协调指挥计算机各部件工作

- 1、 指令控制器
  指令控制器是控制器中相当重要的部分，它要完成取指令、分析指令等操作，然后交给执行单元（ALU或FPU）来执行，同时还要形成下一条指令的地址。
- 2、 时序控制器
  时序控制器的作用是为每条指令按时间顺序提供控制信号。时序控制器包括时钟发生器和倍频定义单元，其中时钟发生器由石英晶体振荡器发出非常稳定的脉冲信号，就是CPU的主频；而倍频定义单元则定义了CPU主频是存储器频率（总线频率）的几倍。

运算器（算术逻辑单元，ALU）
完成算术运算和逻辑运算
ALU主要完成对二进制数据的定点算术运算（加减乘除）、逻辑运算（与或非异或）以及移位操作。在某些CPU中还有专门用于处理移位操作的移位器。
通常ALU由两个输入端和一个输出端。

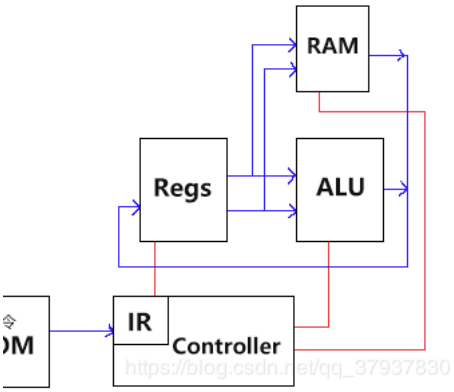
寄存器组（Regs）
通用寄存器组是一组最快的存储器，用来保存参加运算的操作数和中间结果。

存储器（内存，RAM）
存储程序和数据，实现记忆的功能，内存储器(Memory)也被称为内存，其作用是用于暂时存放CPU中的运算数据，以及与硬盘等外部存储器交换的数据。

指令寄存器（IR）
这一部分属于控制器，CPU中程序起点是这儿，所以着重强调以下，有助于以下内容理解
指令寄存器（IR，Instruction Register），是临时放置从内存里面取得的程序指令的寄存器，用于存放当前从主存储器读出的正在执行的一条指令。

大体框图

框图描述了CPU的基本组成以及之间的关系



的为控制线路，蓝色部分为数据传输的通路

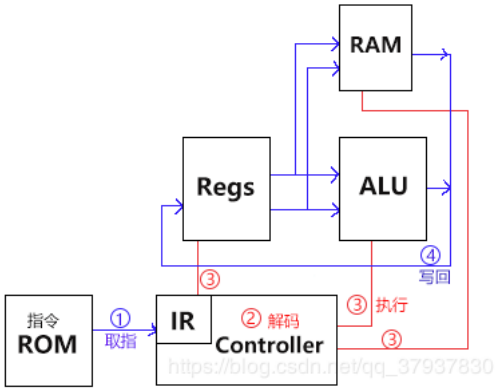
CPU工作过程

以下我们先举一个CPU执行加法操作的例子，了解一下CPU的主要工作过程：

总体过程

首先，CPU从存储器或高速缓冲存储器中取出预存的操作指令，放入指令寄存器，并对指令译码。

把指令分解成一系列的微操作，然后发出各种控制命令，执行微操作系列，从而完成一条指令的执行。



以上是大体的操作流程，乍一看似懂非懂，没事，接下来我们逐字解析-

汇编语言

在介绍指令之前先介绍汇编语言。

是人类所能直接识别的最低层次的语言，也可以说是计算机所能识别的最高层次的语言，人类可以利用汇编语言和计算机交流，让计算机执行各种命令。

比如在此例加法操作中的汇编代码是：ADD R2，R0，R1

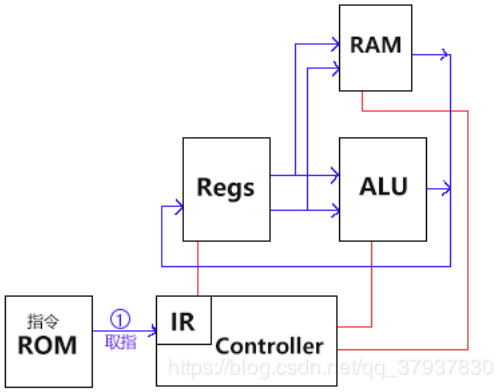
- 汇编代码 组成：
  - 进行什么操作？（操作类型，功能选择）
    这里的操作类型为执行加法操作ADD
  - 对什么进行操作？（操作结果，目标寄存器）
    这里是对寄存器R2进行操作，或者说目标寄存器是R2
    加之后的结果存放到目标寄存器R2中
  - 操作什么？（操作数）
    这里进行的操作是将寄存器R0和寄存器R1中的值相加，
    或者说源寄存器为R0和R1

这条汇编代码的执行结果为：$R2=R1+R1，(相加

二编程序
接下来在汇编语言的基础上介绍汇编程序
二编程序（Assembler）一将汇编语言书写的程序翻译成机器语言程序的软件。
几器其实不能直接识别汇编语言，还需要汇编程序将汇编语言翻译成对应的机器语言，也就是接下要介绍的机器指令

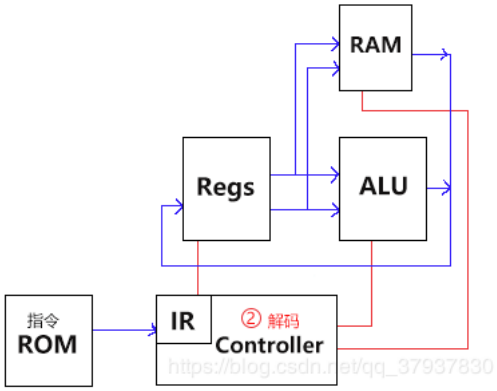
指令
指令是计算机规定执行操作的类型和操作数的基本命令。
一条机器指令实际上就是一串二进制序列，
比如本例中的汇编代码：ADD R2，R0，R1 对应的机器指令就是：
0001 0000 1000 0100
这串指令是16位的，每一位都有其特定的含义或者说都代表对应的操作，
比如说从左往右的3位000 代表的操作就是选择两个寄存器运算的功能
后面五位1 0000 代表的意思是选择加法操作，后面的功能会在后文详细介绍。

第一阶段：提取
这些16位的二进制指令实际是存储在RAM或者Cache中的，执行CPU想要执行这个指令，需要将他从RAM中提取出来。由程序计数器（Program Counter）指定存储器的位置。(程序计数器保存供识别程序位置的数值。换言之，程序计数器记录了CPU在程序里的踪迹。)



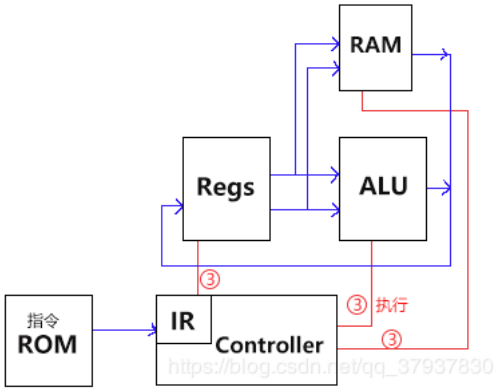
第二阶段：解码
CPU根据存储器提取到的指令来决定其执行行为。
在解码阶段，指令被拆解为有意义的片段。根据CPU的指令集架构定义将数值解译为指令。一部分的指令数值为运算码（Opcode），其指示要进行哪些运算。
本例中的指令为：
0001 0000 1000 0100
可以按照不同字段的作用解析为以下格式：

汇编代码	ADD R2， R0， R1															
十六进制指令	1				0				8				4			
二进制指令	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
具体含义	两个寄存器运算		X		选择加法运算				R2		R0		R1		X	
位含义	功能选择				选择运算				目标寄存器		源寄存器		源寄存器		空位	



第三阶段：执行

经过解码，一条指令可以被解码成很多不同的微指令，这些微指令片段被控制器输出到不同的寄存器，从而控制不同的寄存器操作，完成加法功能。比如在例子中，控制器选择了R0和R1寄存器，并将他们的值送到ALU算术逻辑单元进行加法运算。

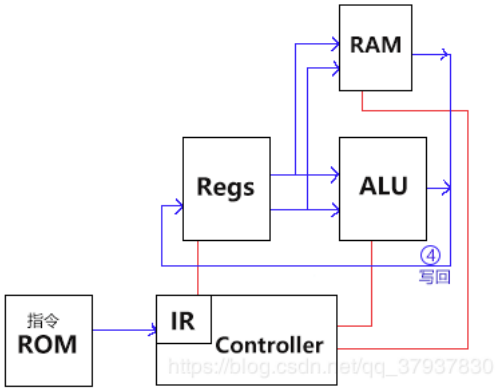


第四阶段：写回

完成加法功能之后，运算结果被写入到R2寄存器中。

运算结果经常被写进CPU内部的暂存器，以供随后指令快速存取。在其它案例中，运算结果可能写进速度较慢，但容量较大且较便宜的主记忆体中。某些类型的指令会操作程序计数器，而不直接产生结果。这些一般称作“跳转”（Jumps），并在程式中带来循环行为、条件性执行（透过条件跳转）和函式。许多指令会改变标志暂存器的状态位元。这些标志可用来影响程式行为，缘由于它们时常显出各种运算结果。例如，以一个“比较”指令判断两个值大小，根据比较结果在标志暂存器上设置一个数值。这个标志可藉由随后跳转指令来决定程式动向。在执行指令并写回结果之后，程序计数器值会递增，反覆整个过程，下一个指令周期正常的提取下

一个顺序指令。

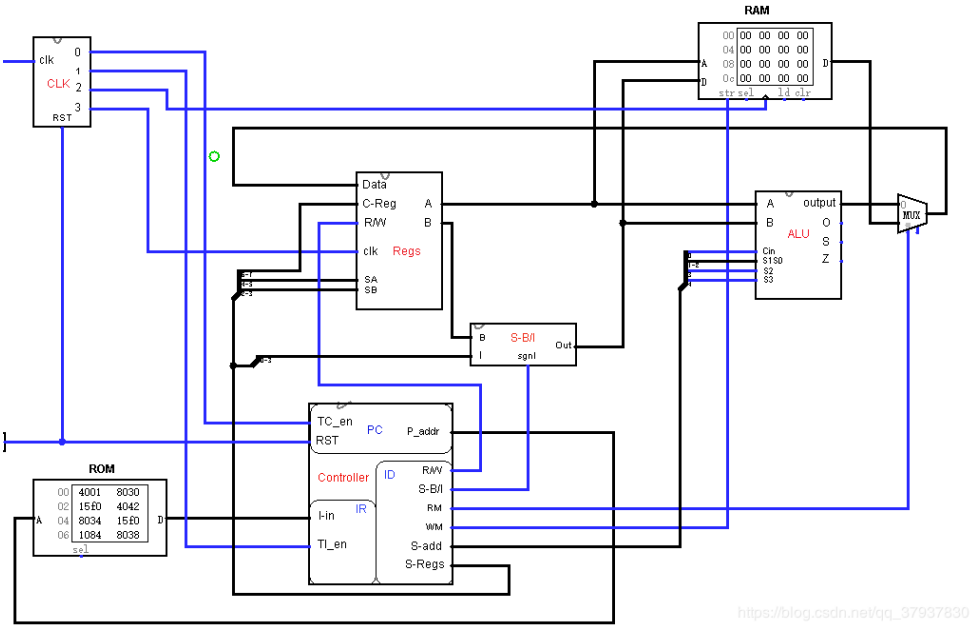


宏观实现预览

里我们简单预览一下已完成Demo整体的运行步骤，结合着以上的理论简要分析Demo的大体

是经过示例了解CPU各个模块的功能以及系统数据流图。
文第4部分往后将会详细讲解各个部分的具体实现，第一遍看下留个整体映像，与后文结合，
时照将会加深理解，

系统总览



logisim软件的仿真功能)

解析：

上模块的基础上加上了CLK时钟模块，RST复位模块和S-B/I模块；
模块上文简要介绍过，在后文也将具体介绍每个模块的引脚功能，这边只需要大体了解。

CLK：时钟模块，这里一个指令周期被细分为四个机器周期，以分步执行微指令

RST：复位模块，使系统回到初始状态

ROM：存放指令的地方，这里每一条指令以四位16进制序列的形式存放在ROM中

Controller：控制模块，包含三部分

- 程序计数器（PC）
- 指令寄存器（IR）
- 指令译码器（ID）

egs：寄存器组，里面包含

- R0~2：三个用来存放数据的数据寄存器；
- R3：一个用来存放地址的地址寄存器
  注：寄存器组中的寄存器只是暂时存放数据。

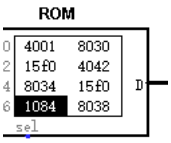
-B/I：通路选择器，可选择B通道导通或者I立即数通道导通

ALU：算术逻辑单元，主要计算A、B通道的输入，并将他们从output输出到Regs的data端口。

RAM：内存，CPU内部存放数据的地方，寄存器组中的寄存器只是暂时存放数据。

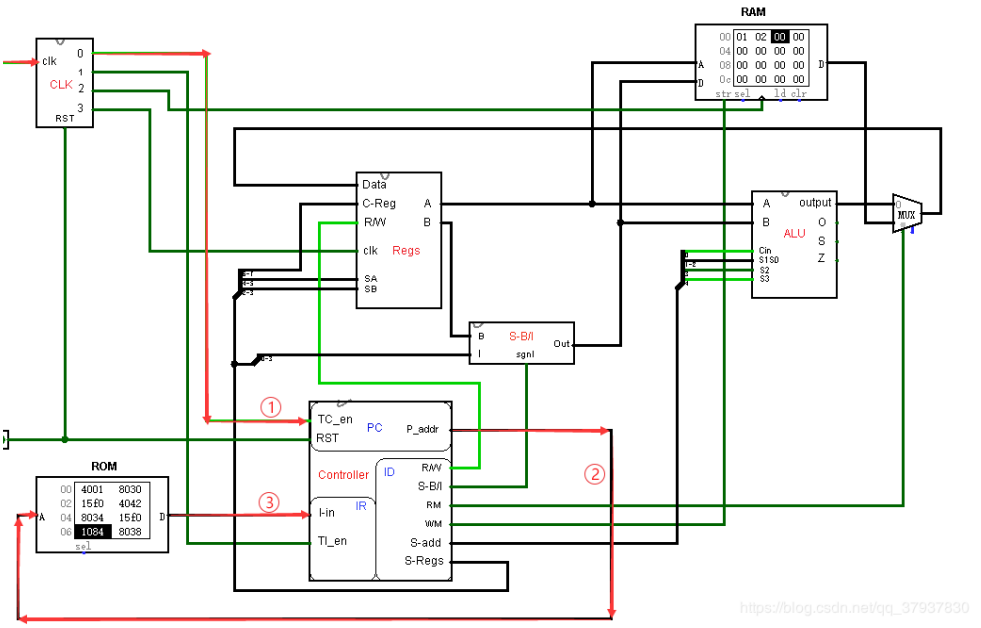
举例演示流程

以上的例子：加法操作，ADD R2，R0，R1
代码在ROM的位置在

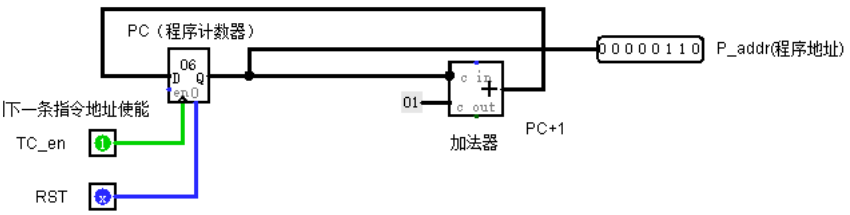


我们从这里开始执行：点击clk产生时钟信号：

0 机器周期0：

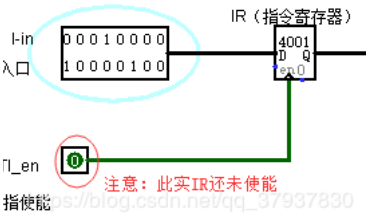


器周期0中，顺着红线的思路往下走：
TC_en为PC功能的使能，使能之后，+1后的PC指向现在将要取指的指令的地址，
dr=0000 0110=06



P_addr输出到ROM的A口，在ROM中选中地址为

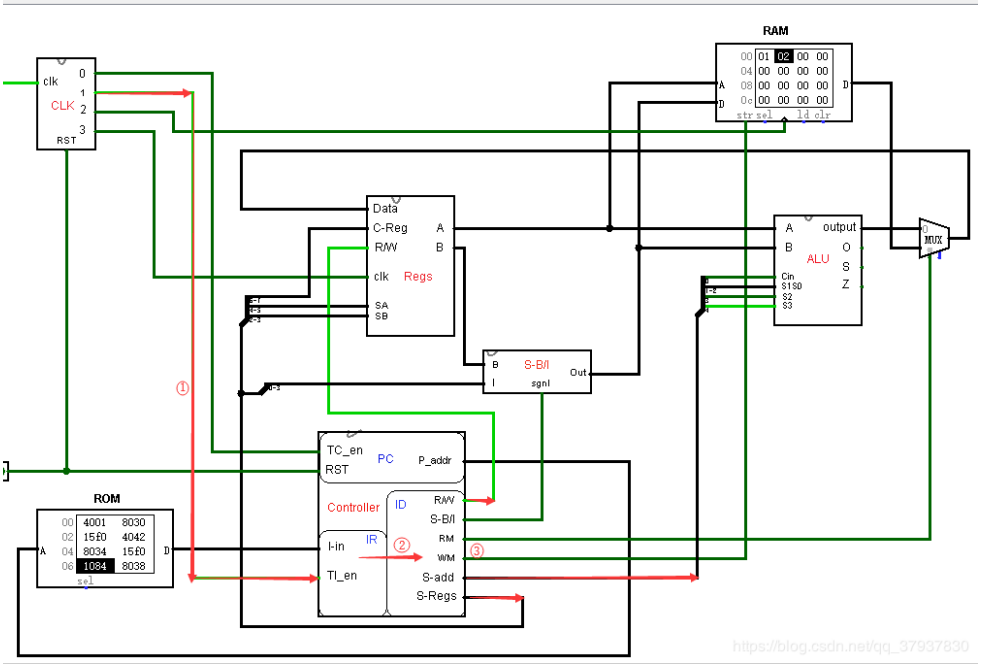
指令从ROM的D口传给指令寄存器（IR）中的指令入口



此时TI_en取指使能口还为0，未使能，所以指令还没有进入IR

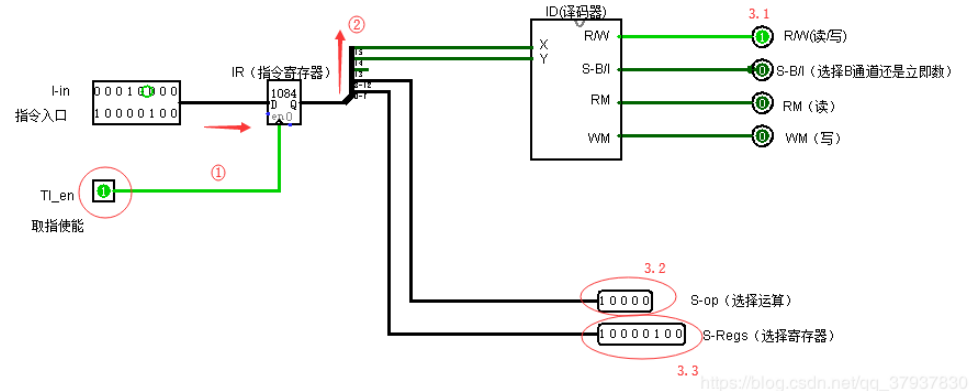
1 机器周期1：

点击clk，进入第二个机器周期：



、系统给了TI_en高电平，使能了**取值**功能，数据进入指令寄存器IR中；

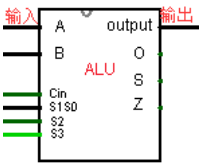
、 指令进入IR后立即被译码器（ID）**译码**，译码后变成不同的控制电平从各个控制位输出；



、不同的控制位控制不同的功能，

- 3.1 **R/W**控制位为1，意味着实现的是写功能：
  这里的写的意思是将计算结果写入寄存器的意思，W控制位控制寄存器组（Regs）中的R/W位

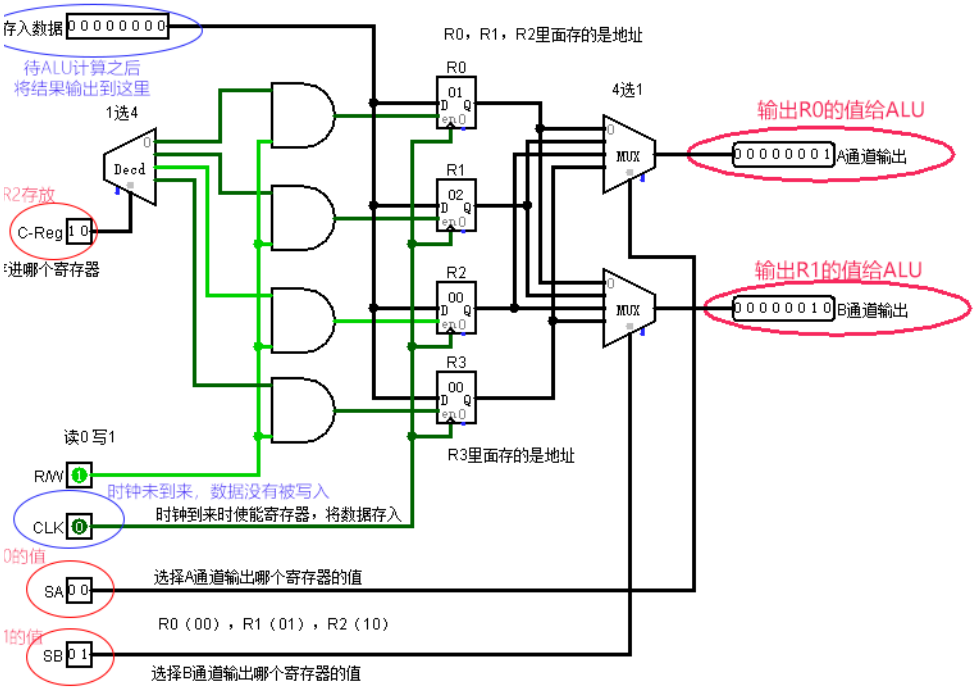
- 3.2 **S-op**：选择运算方法（Select-operation）：（5位）
  10000这五位控制算术逻辑单元(ALU)，表示选择加法操作。
  ALU选择了这个功能会将A，B通道输入的值相加，并从output口输出，默认输出到寄存器组（Regs）的data口



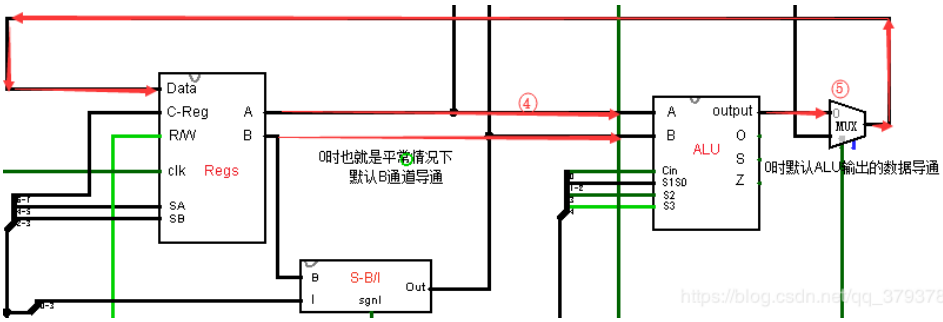
- 3.3 **S-Regs**：选择寄存器（Select-Register）（8位）
  1000 0100这八位控制寄存器组，这8位选择的是加操作时，哪几个寄存器是加数（源寄存器），哪个寄存器是被存放结果的目的寄存器：

1	0	0	0	0	1	0	0
R2		R0		R1		X	
目标寄存器		源寄存器		源寄存器		空位	

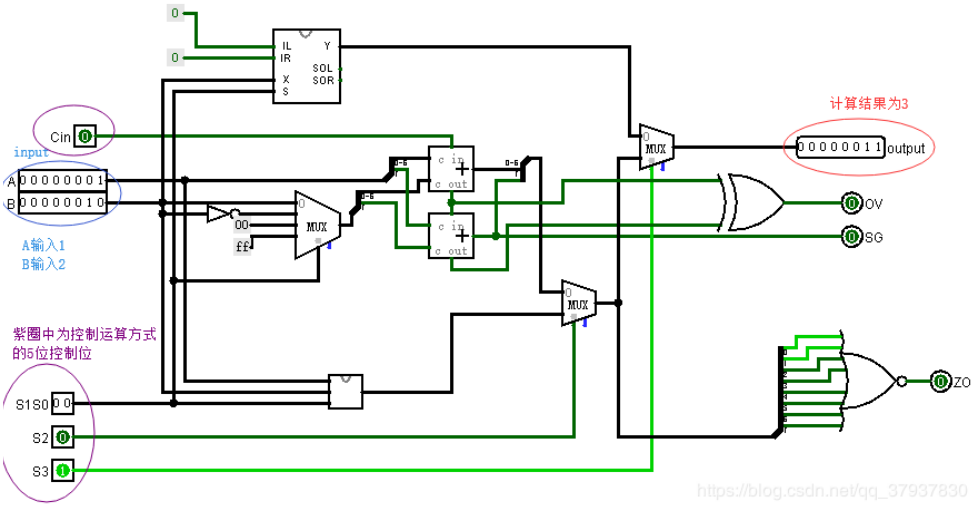
在Regs模块内部中具体的控制效果如下图中红色小圈：
右边的红色大圈为Regs模块中输出的值，分别输出给3.2中ALU的A、B通道，给ALU进行加法运算。
在这里的进度中，ALU还未将计算结果返回给Regs寄存器组，于是，寄存器组中存入数据的蓝色框框为原来的数据。



路接在上面的过程之后，是ALU进行加运算的过程。数据流图如下：



、Regs中输出的数值传给ALU的A、B通道。在ALU内部模块中进行加运算

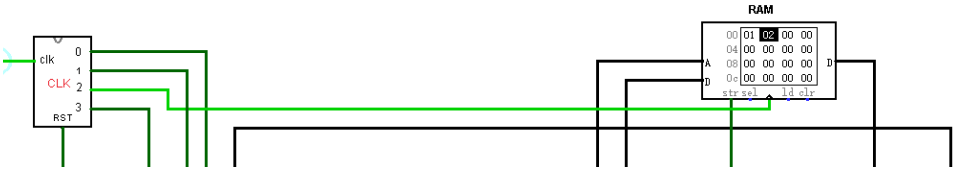


、加运算后，ALU将结果输出给Regs的data口，于是又回到寄存器组Regs中了。

这个数据在Regs中等待被存进R2中，需要等到clk时钟到来时候，才能被使能写入R2中。

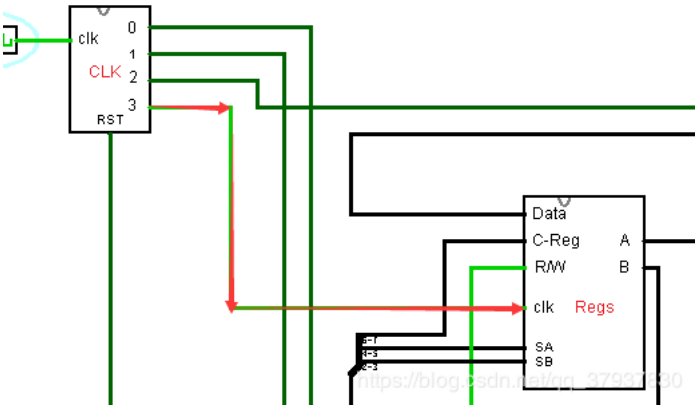
2 机器周期2：

个机器周期中，clk使能RAM，使数据能够存进去，由于本add操作中不需要把数据存进RAM所以此周期没啥改变：

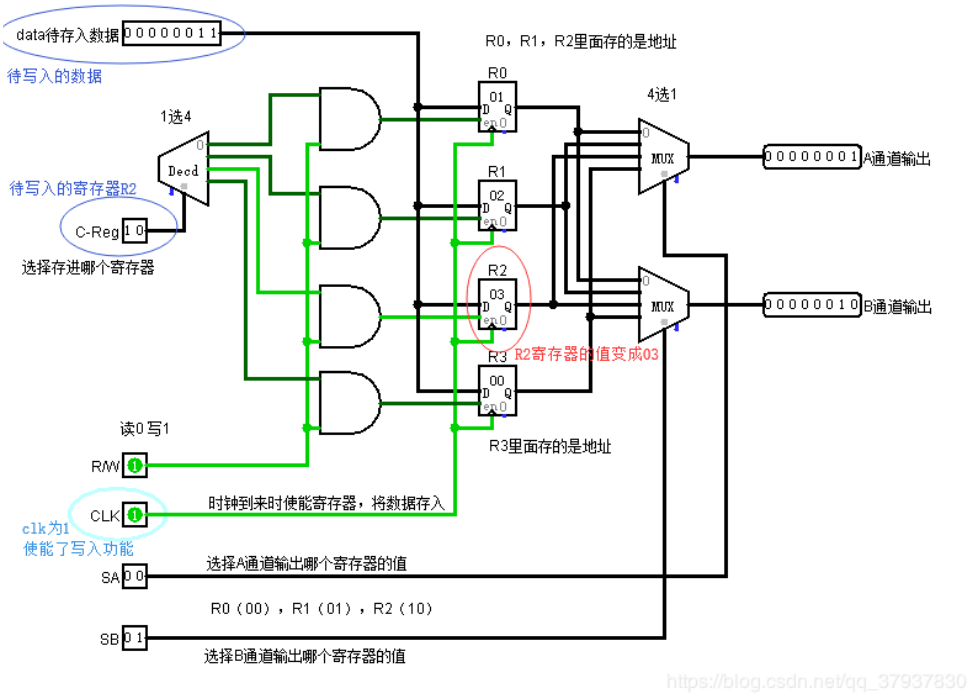


3 机器周期3：

期中，clk给了Regs中的clk端口高电平，使能了Regs中数据写入功能：



字器组模块内部中，clk得到了高电平，使能了数据写入功能，待写入的数据为data，待写入
字器为R2（10）
值变成了03。



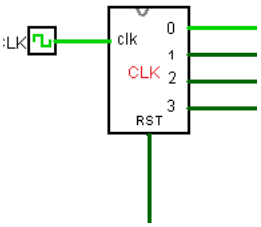
整个加法操作完成。接下待进入各个模块的详细设计与说明。

系统设计

CLK模块

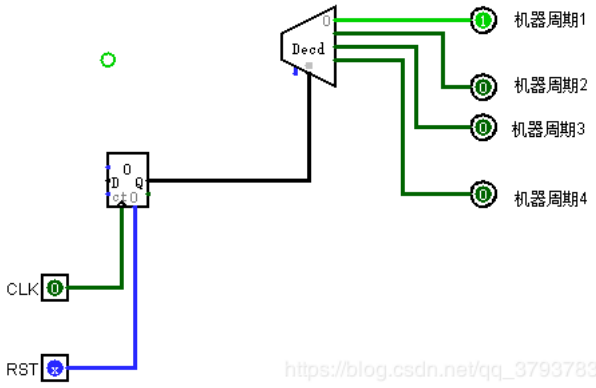
整个CPU中的作用就是将一路CLK信号变成4个循环的机器周期，4个机器周期组成一条指
执行周期，也就是一个指令周期。

览

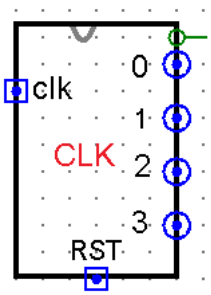


LK模块的内部组成：

- CLK基础模块连接一个初始为0，最大计数为3，的一个计数器，这样可以构成一个周期为4的循环
- 计数器的输出接到一个Decoder解码器上，将计数输出转变成四个不同的CLK输出，构成四个机器周期



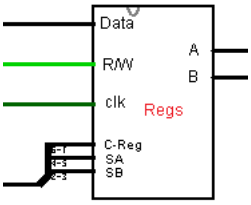
成之后的CLK模块：



Regs模块

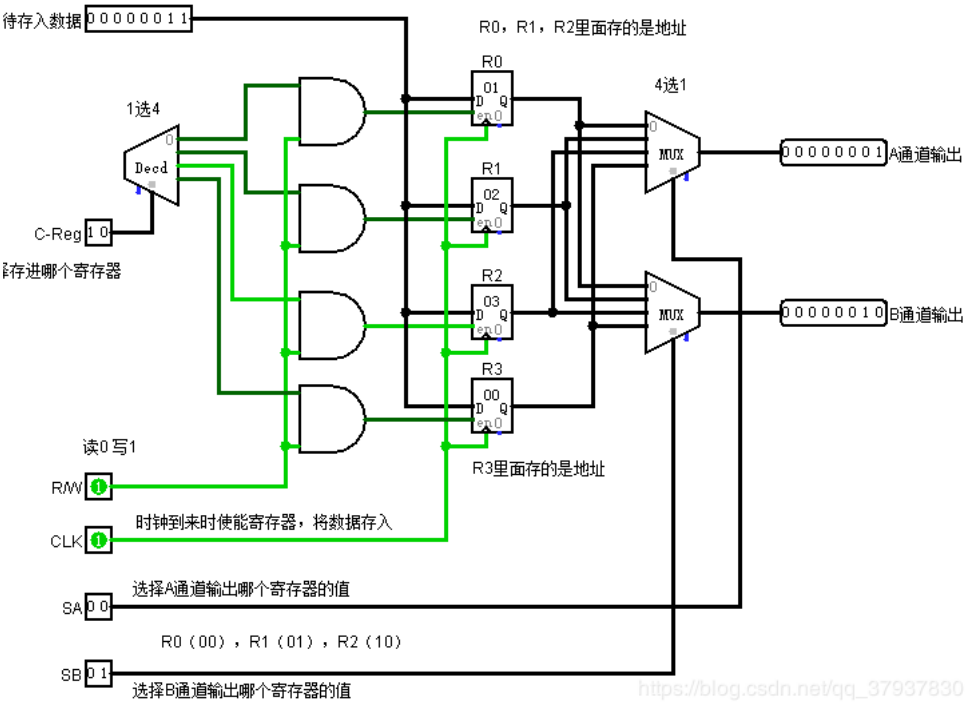
存器组模块的作用：
择待存数据的寄存器和将要输出数据的寄存器，data
择寄存器数据输出通道A/B
择存储地址的地址寄存器，addr
寄存器的作用是在运算过程中短暂存储数据或者地址，不能像ROM那样长期存储

览：



内部组成

构图示：



输入：

- **data**
  待存入的8位2进制数据，是ALU模块的output输出的数值

控制位：
egs模块共8位控制位

- **R/W （1位）**
  选择读取寄存器的数值还是将数值写入寄存器，0读1写
- **CLK （1位）**
  读写功能的使能，当时钟到来时才能进行读写功能
- **C-Reg (2位)**：(Choose-Register)
  当选择写功能的时候，选择将要写进数据的寄存器，00代表选择R0，01-R1，10-R2，11-R3
- **SA （2位）**
  当选择读寄存器功能的时候，选择A通道输出哪个寄存器的值
- **SB （2位）**
  当选择读寄存器功能的时候，选择B通道输出哪个寄存器的值

存器组：

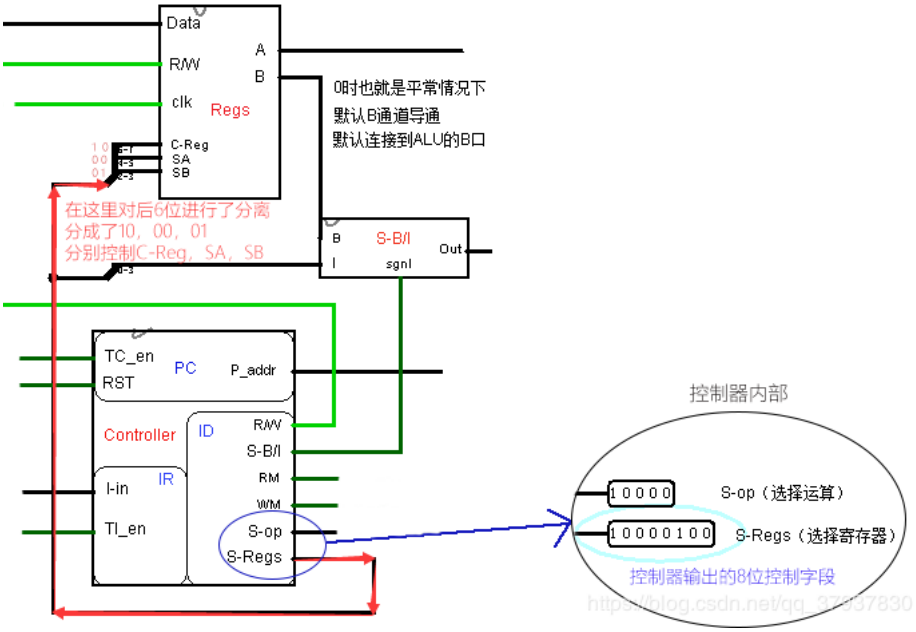
- 三个数据寄存器，R0，R1，R2
  用来短暂存放用于计算的数据的寄存器
- 一个地址寄存器，R3
  专用于将数据写进内存的操作中（WM），用于标识写进RAM中的地址，有自加的指令，相当于PC

输出
寄存器组共一个数值输入和两个数据输出

- A通道有两个功能：
  - 输出数值给ALU的A口运算
  - 输出地址给RAM的A（address）口，选择存进RAM的位置
- B通道也有两个功能
  - 输出数值给ALU的B口运算
  - 输出数据给RAM的D（Data）口，选择存进RAM的数值

控制位详解：

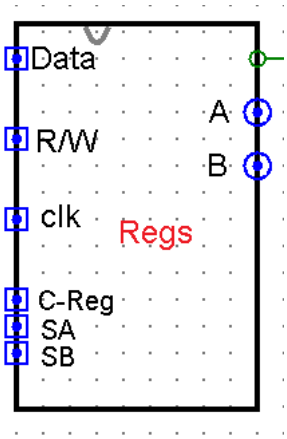
由上可知，共8个控制位；
其中 C-Reg(2位)、SA（2位）、SB（2位）这6位控制位的控制字段来自控制器模块传递过来的8位二进制字段S-Regs，也就是下图中红线标注的这一部分：
而R/W（1位）控制位也是由控制器传出的控制字段R/W决定
clk（1位）控制位则是由CLK模块的第四个机器周期给的clk信号



以下是指令详解：

汇编代码	ADD R2，R0，R1															
十六进制指令	1				0				8				4			
二进制指令	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
位含义	功能选择				选择运算				目标寄存器		源寄存器		源寄存器		空位	
具体含义	两个寄存器运算			X	选择加法运算				R2		R0		R1		X	
对应控制位	000代表R/W位为1				S3	S2	S0 S1		Cin	C-Reg		A		B		无
控制模块	此条中主要控制Regs				ALU				Regs							

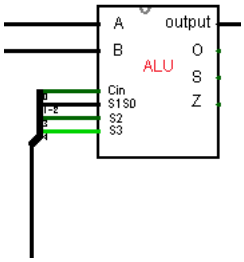
集成之后的模块：



ALU模块

作用
ALU顾名思义算术逻辑单元，CPU的本质就是处理运算，因此，ALU与Controller可以算是CPU的核心部分，
可以简单的将这部分理解成一个黑匣子；两个运算输入和一个运算输出，还有一个功能选择。
(其实主要是由于这部分是课设的队友做的，我也没太大研究，主要研究指令集以及控制器，寄存器部分，如果以后有兴趣了应该会具体补充)

预览



组成：

有两个数据输入端

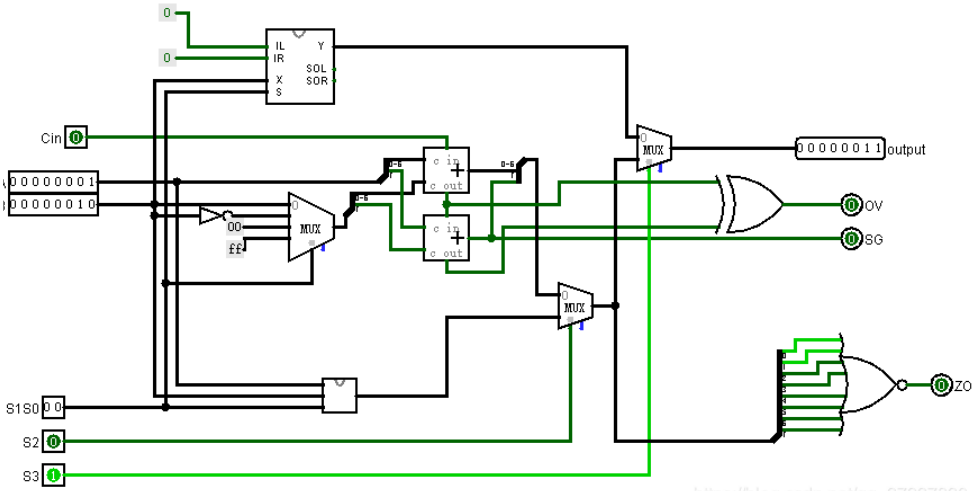
- A通道数据输入：
  是Regs模块A通道输出的数据
- B通道数据输入
  是Regs模块B通道输出的数据；
  或者是 立即数通道的立即数输入；
  由S-B/I模块决定，一般情况下默认为B通道数据，下文讲解

一个数据输出端Output:
计算结果输出，直接输出到Regs模块的data（待写入数据）端口

五个控制位，四个控制字段：

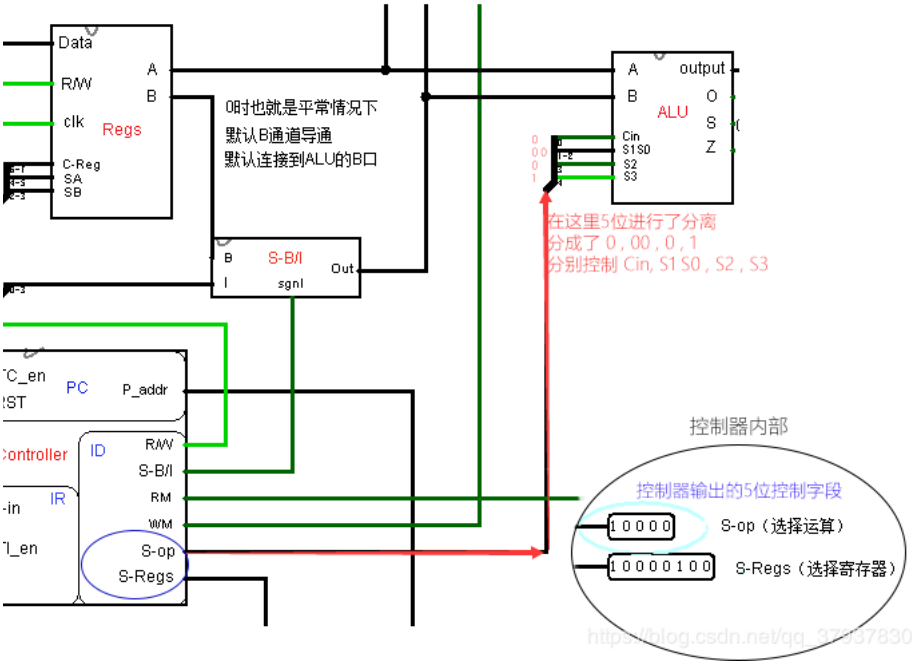
- Cin（1位）
- S1 S0 (2位)
- S2（1位）
- S3（1位）

这五位数从Cin到S3（从指令序列低位到高位）组成一个5位的控制字段，代表不同的运算功能，比如1 0000代表选择ALU为加法运算。



控制位详解：

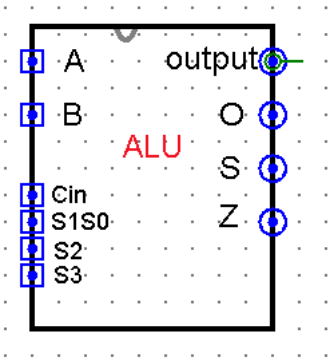
同上，这五位控制位，来自控制器传出的五位控制字段：



指令详解：

汇编代码	ADD R2， R0， R1															
十六进制指令	1				0				8				4			
二进制指令	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
位含义	功能选择				选择运算				目标寄存器				源寄存器			空位
具体含义	两个寄存器运算			X	选择加法运算				R2				R0			R1				X
对应控制位	000代表R/W位为1				S3	S2	S0 S1		Cin	C-Reg			A			B				无
控制模块	此条中主要控制Regs				ALU					Regs										

块集成：

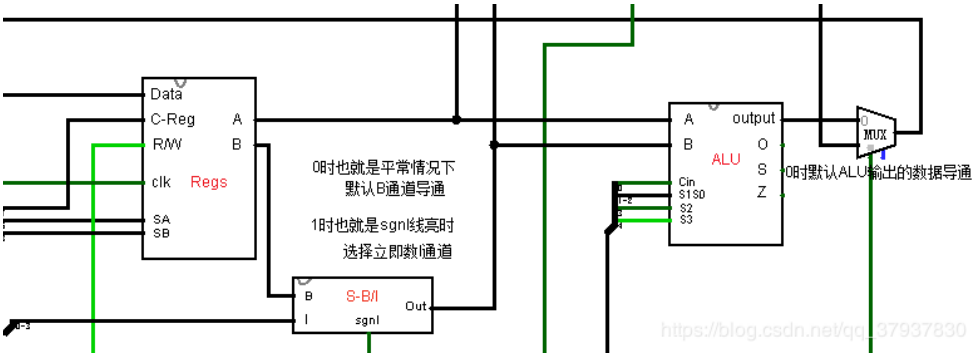


S-B/I模块

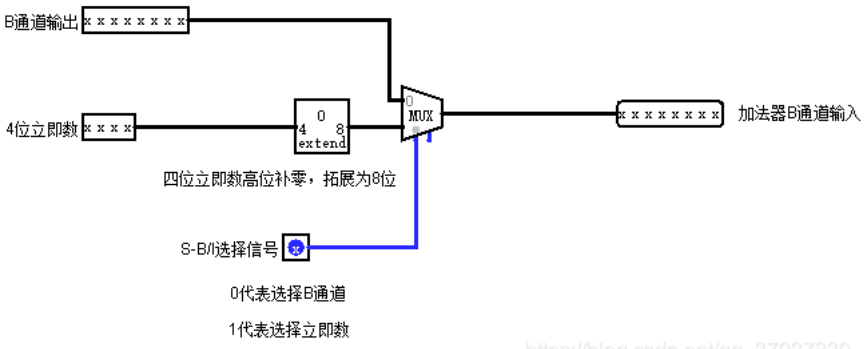
己定义的一个模块名，意为Select-B_{tunnel}/Immediate$_{tunnel}$选择B通道还是立即数通道

用
们知道，我们在进行加法运算的时候需要Regs模块的A通道输出R0寄存器的值给ALU的A通
，需要Regs模块的B通道输出R1寄存器的值给ALU的B通道，来使ALU完成加运算。
是在别的指令中，比如说赋值MOV指令：MOV R0，1
们需要将立即数1的值写进R0寄存器，而已知的写寄存器的数据是从ALU的Output传递给
egs的data口的；
此我们需要考虑怎么将立即数传给ALU，于是我们可以将要写入的立即数写入指令中，在解
之后，传给立即数通道，通过立即数通道传给ALU，将B通道与立即数通道复用，通过一位
断位判断什么时候用哪个通道输入给ALU的B输入口。

览如下：

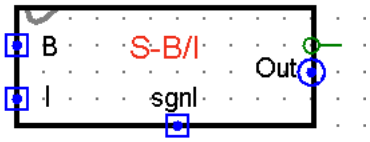


块组成：



- B通道数据输入：
  一个8位的Regs模块的B通道数据输入
- I 立即数通道数据输入：
  - 一个解码后的4位的立即数输入
  - 因为输出为8位，因此还需要一个4位extend8位的拓展器
- S-B/I的选择信号：
  0代表默认选择B通道数据，1代表选择立即数通道
- 一个2选1的选择器，多路选择器
- 一个8位数据输出
  输出选择的数据到ALU的B数据输入口

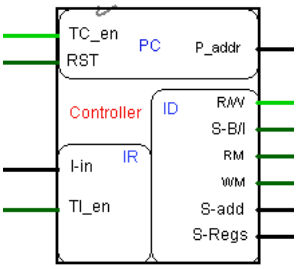
块集成



控制器模块

用
们知道，CPU是计算机的核心，而控制器是一个CPU的核心。控制器的功能主要有取指、解码、控制各寄存器进行执行微指令。

览：



成:
制器包括：

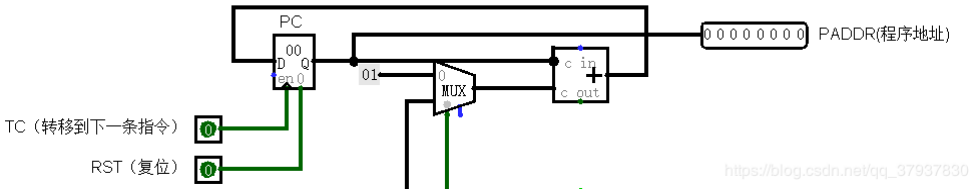
- PC（程序计数器）
  负责保存程序执行到的位置，
- IR （指令寄存器）
  当计算机的某一计算循环开始时，先根据地址寄存器的地址，从内存储器（RAM）中读出一条指令，存入指令寄存器中。
- ID （指令译码器）
  指令寄存器的相应位送入指令译码器(操作码译码、变址译码等)。根据译码结果产生相应的控制信号，完成指令规定的运算、传送数据等动作。

1 程序计数器（PC）的设计

用：
C的作用在于知道指令在哪里；
一般来说，程序包括三种结构：**顺序、分支**和**循环**；其中**分支**和**循环**实际上可以由跳转（Jump）实现，
以程序的执行方式可以归纳为**顺序**和**跳转**：

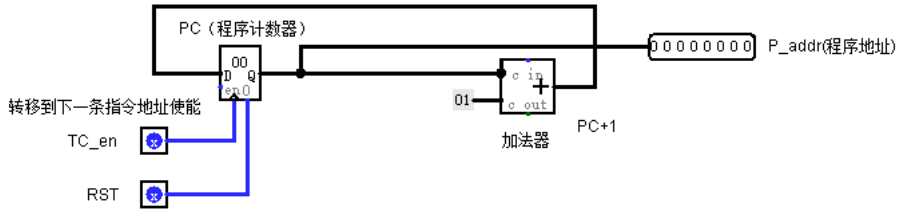
- 对于顺序执行，只需要对程序计数器每次加一个固定的数即可实现；
- 对于跳转，有两种方式实现，一种是绝对跳转，一种是相对跳转，
  - 绝对跳转直接给出下一条指令的地址，对程序计数器直接置数，
  - 对于相对跳转，则通过程序计数器加上或者减去某个值实现，**一般来说大多数情况下用的是相对跳转。**

理论实现：
程序计数器可以用寄存器、加法器和数据选择器实现。
以下是用logisim设计的PC电路：
(电路设计部分参考文章开始的链接)



- 如果程序顺序执行，则PC加上MUX选择的常数01
- 如果程序要跳转，MUX选择下方的通路，数据为地址偏移量，PC=PC+地址偏移量（MUX为二选一的选择器，或者可以理解为闸门，一个时刻只有左边的一路被选择输出）

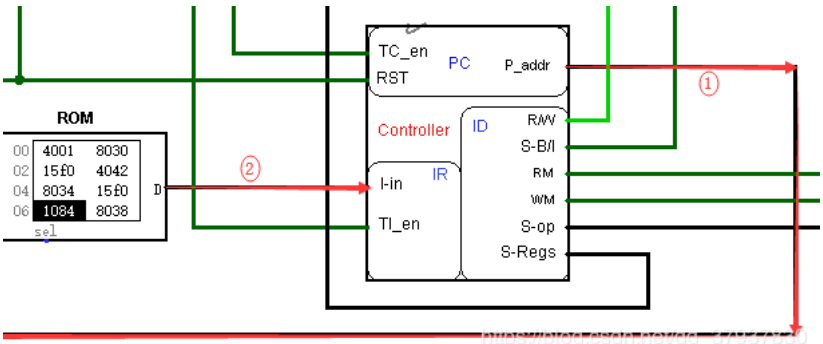
实际设计：
由于本次课设要求简单实现，CPU功能，因此极尽简化之后的系统只需要简单的顺序执行程序，因此，选择分支部分就可以省去，最终的模型如下：



- Tc_en：PC+1的使能，实际作用是将加1后的PC输出到P_addr
- RST：复位

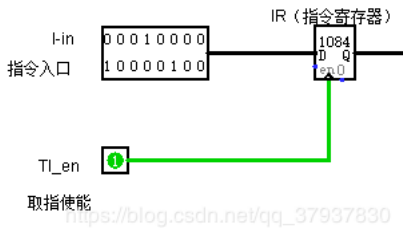
2 指令寄存器（IR）的设计

地址传给P_addr之后经历如下过程
程序计数器给定了地址输出到存储器（ROM）中；
ROM会将对应地址的16进制指令给IR寄存器的I-in指令入口；



指令逻辑上是暂时存在I_in中，还没有进入IR，需要TI_en取指使能口变成高电平，
才会静茹IR指令寄存器
寄存器与一般的寄存器功能一样，暂时存放指令）

际实现：

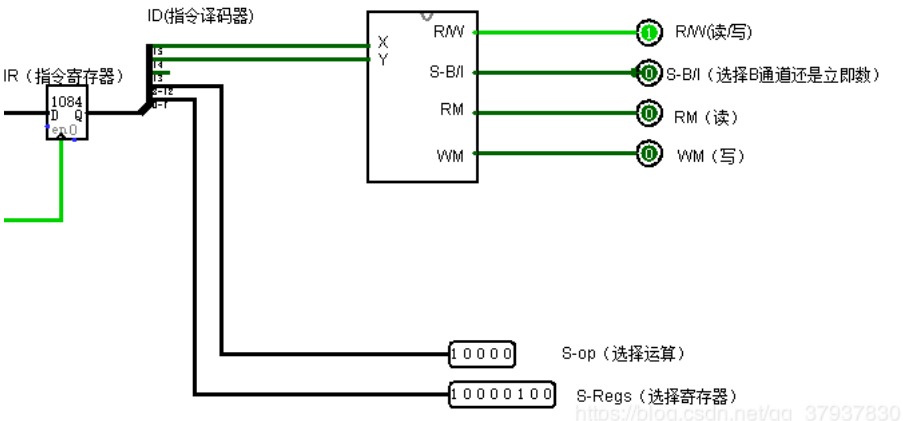


3 指令译码器（ID）的设计

计指令译码器之前，我们首先要明确指令格式，这里使用得指令集设计标准类似MIPS，只不
PS指令集为32位，这里为16位。
的16位指令是根据算术逻辑单元（ALU）与寄存器组（Regs）的控制字段设计的，指令部分
文会有详细解释，这边先说以下指令译码器的设计部分。

用:
指令译码器的作用在于翻译指令，我们知道指令在CPU中实际是一个二进制序列，
有操作码，操作数组成，分别代表要进行进行什么操作，以及操作的具体数值和寄存器，
将一个二进制序列逐位拆解为不同的控制位，以控制不同的模块单元。

际实现：
：ID不止是图中矩形部分，还包括下面的S-op与S-Regs，或者准确说，左边的分离分支
plitter是译码器

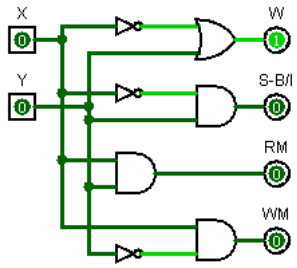


6位的指令由一个splitter 分离逐位分离之后，变成不同的控制字段输出

- 0-7位：S-Regs
  这8位主要控制寄存器或者立即数的选择，在Regs模块的设计中有详细的介绍，在后文的指令格式详解中也会有详细介绍
- 8-12位：S-op
  这5位主要控制ALU，功能是运算方式的选择，在上面的例子中1 0000 代表选择加法运算
- 13位：空位
- 14、15位：
  这两个位组合控制功能的选择
  具体功能如下：

选择功能	14-15bit
两个寄存器运算	00
寄存器+立即数计算	01
写主存	10
读主存	11

于是我们只需要将这2位的字段解析成每个控制字段的对应输出就行
在实际电路中以一个译码器的方式实现：将2位二进制译成4个输出，实际是由真值表推出来：



指令结构详解：

为容对应上文中ID指令译码器设计部分。
计指令译码器之前，我们首先要明确指令格式，这里使用得指令集设计标准类似MIPS，只不PS指令集为32位，这里为16位。这里的16位指令是根据算术逻辑单元与寄存器组的输入口制字段设计的。
个操作对应的控制字段如下：

功能分析								
	S3~Cin(5bit)	W	CI	RM	WM	SF	JP	JW(3bit)	IM
存器运算	V	1	0	0	0	1	0	X	X
器+常量运算	V	1	1	0	0	1	0	X	V
存	X	0	0	X	1	0	0	X	X
存	X	1	X	1	0	0	0	X	X
	X	0	X	X	0	0	1	V	V

/的意思是有效，X的意思是无效（无关），IM指的是立即数。

图中，可以对W/CI/RM/WM/SF/JP/JW进行编码，编码成000,001,010,011,100，这样只需个信号就能确定六个控制信号了，也就是这样：

选择功能	14-15bit
两个寄存器运算	00
寄存器+立即数计算	01
写主存	10
读主存	11

为写入ROM中的16进制代码：

001：

汇编代码	MOV R0 ，1															
含义	将立即数1存进寄存器R0里面															
十六进制指令	4				0				0				1			
二进制指令	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
位含义	功能选择		X		X				目标寄存器		X		立即数			
具体含义	寄存器+立即数		X						R0		X		立即数			
对应控制位	R/W+S-B/I			S3	S2	S0 S1		Cin	C-Reg		SA		I通道			
控制模块	Regs+I通道		ALU						Regs				I 立即数通道			

030:

汇编代码	STORE R3 ，R0															
含义	将R0寄存器里的数值存进RAM，地址为R3寄存器里存的地址															
十六进制指令	8				0				3				0			
二进制指令	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
位含义	功能选择		X		X				X		地址寄存器		数据寄存器		X	
具体含义	写操作		X						X		R3		R0			
对应控制位	WM			S3	S2	S0 S1		Cin	C-Reg		SA		SB			
控制模块	RAM		ALU						Regs							

084：

汇编代码	ADD R2， R0， R1															
含义	R2=R0+R1，将寄存器中的值相加后存进R2寄存器里															
十六进制指令	1				0				8				4			
二进制指令	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
位含义	功能选择				选择运算				目标寄存器		源寄存器		源寄存器		空位	
具体含义	两个寄存器运算		X		选择加法运算				R2		R0		R1		X	
对应控制位	R/W位为1			S3	S2	S0 S1		Cin	C-Reg		A		B		无	
控制模块	此条中主要控制	Regs		ALU					Regs							

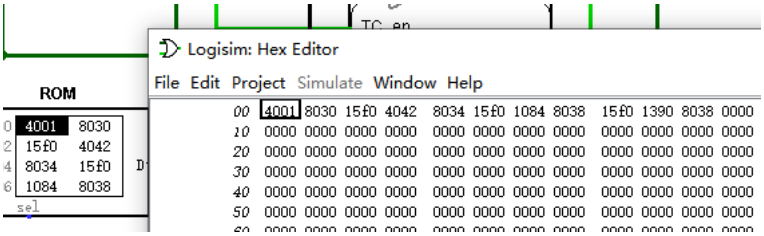
390

汇编代码	SUB R2， R1， R0															
含义	R2=R1-R0，将寄存器中的值相加后存进R2寄存器里															
十六进制指令	1				3				9				0			
二进制指令	0	0	0	1	0	0	1	1	1	0	0	1	0	0	0	0
位含义	功能选择				选择运算				目标寄存器		源寄存器		源寄存器		空位	
具体含义	两个寄存器运算		X		选择减法运算				R2		R1		R0		X	
对应控制位	R/W位为1			S3	S2	S0 S1		Cin	C-Reg		A		B		无	
控制模块	此条中主要控制	Regs		ALU					Regs							

5f0

汇编代码	INC R3 ，R3															
含义	地址寄存器R3里面存的地址自加1，以便下次写入RAM下一个地址															
十六进制指令	1				5				f				0			
二进制指令	0	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0
含义	两个寄存器运算				自加+1				R3		R3		X		X	

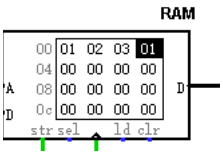
次设计用的仿真指令，存在ROM中：



译义：

指令	汇编代码
4001	MOV R0，1
8030	STORE R3, R0
15f0	INC R3,R3
4042	MOV R1，2
8034	STORE R3, R1
15f0	INC R3,R3
1084	ADD R2,R0,R1
8038	STORE R3, R2
15f0	INC R3,R3
1390	SUB R2,R1,R0
8038	STORE R3, R2

仿真效果：



完结，over

PU的设计和实现 阡飞陌 1万+

J的简介： 此CPU为冯·诺伊曼架构的，顾名思义程序存储器和数据存储器共用了一组数据和地址总线，当然有兴趣...

建之logisim篇_junlinqu的博客-CSDN博客_cpu logisim 7-31

n 调试技巧保证后加的指令不影响之前指令的正确性。这样一来,就算 CPU 出现 bug,只需调试新加的指令即可。_cpu...

手画一个CPU——Logisim,下_qq_43561302的博客-CSDN博客 7-20

试结果 实验十:MIPS微程序CPU设计 实验十一:硬布线控制器状态机设计 实验十二:多周期MIPS硬布线控制器CPU设...

isim画简易CPU一】8位可控加减法电路 qq_43588553的博客 6807

自定义目录标题欢迎使用Markdown编辑器新的改变功能快捷键合理的创建标题，有助于目录的生成如何改变文本的...

器设计（Logisim实现） Willy__QI的博客 1万+

台介绍1、Logisim软件是一种用于设计和模拟数字逻辑电路的工具。其简单的工具栏界面和构建它们时的电路仿真，...

simcpu,logisim单周期cpu已通过仿真实验-Mail服务器代码类... 7-16

m单周期CPULogisim单周期CPU已通过仿真测试可以运行小规模程序logisimcpu更多下载资源、学习资料请访问CS...

m单周期CPU_logisim单周期cpuj型指令,单周期cpu设计logisim... 7-14

gisim软件描述的单周期CPU,支持MIPS指令,可扩展性较好logisim单周期cpuj型指令更多下载资源、学习资料请访问...

ogisim构建小型数字系统（运动码表） 班克o 5131

要介绍的是中国大学mooc上“计算机硬件系统设计”课程中的一个实验。首先，我对本课程的课程组深表感谢，开...

科技大学计算机组成原理实验报告-CPU设计实验.docx 05-06

技大学《计算机组成原理》实验报告（总），报告目录： 1 CPU设计实验 2 1.1 设计要求 2 1.2 方案设计 3 1.3 实验...

_booksyhay的专栏-CSDN博客_logisim cpu如何运行小规模程序 8-12

CPU底层的运行机制很感兴趣,查了一些计算机组成原理方面的资料,发现可以使用LogiSim软件用最基础的门电路搭...

一步学做一个CPU——2,Logisim的简单入门_ 一个程..._CSDN博客 8-12

栏: cpu设计 Logisim的简单入门 先来以一个与门电路的实验,来看一下基本的操作,在Logisim中一个与门电路生成过...

im设计模型机——编制并执行程序 蛇皮团团怪的博客 4083

算机组成原理课程设第一部分使用logisim软件设计一个模型机器，包括但不限于CPU，主存，实现至少4条指令，...

的CPU及流水线CPU设计（1）---logisim部件设计 TONNE_YANG的专栏 1万+

n的设计是设计CPU的基础，在往后的CPU的代码书写的过程中必然时刻伴随着设计图纸的需求。（如有转载，请注...

理实验 Logisim CPU实验 Hust_qq_43434682的博客..._CSDN博客 8-11

理实验 Logisim CPU实验 HustITTTJH 2020-06-12 19:28:21 1227 收藏 21 分类专栏: 计算机组成原理 版权 这次C...

PU设计实践_使用LogiSim设计CPU-硬件开发文档类资源 8-15

U设计实践开始前的话本人大学党一个,计算机科学与技术专业,平时对电子技术、计算机科学和软件技术有所研究,在...