# 实验四 构建寄存器组和控制器



本节实验要完成的子电路包括：寄存器组 Regs、程序计数器 PC、指令寄存器 IR、指令译码器 ID、S-B/I 电路、时钟电路 CLK 等，如下图所示。

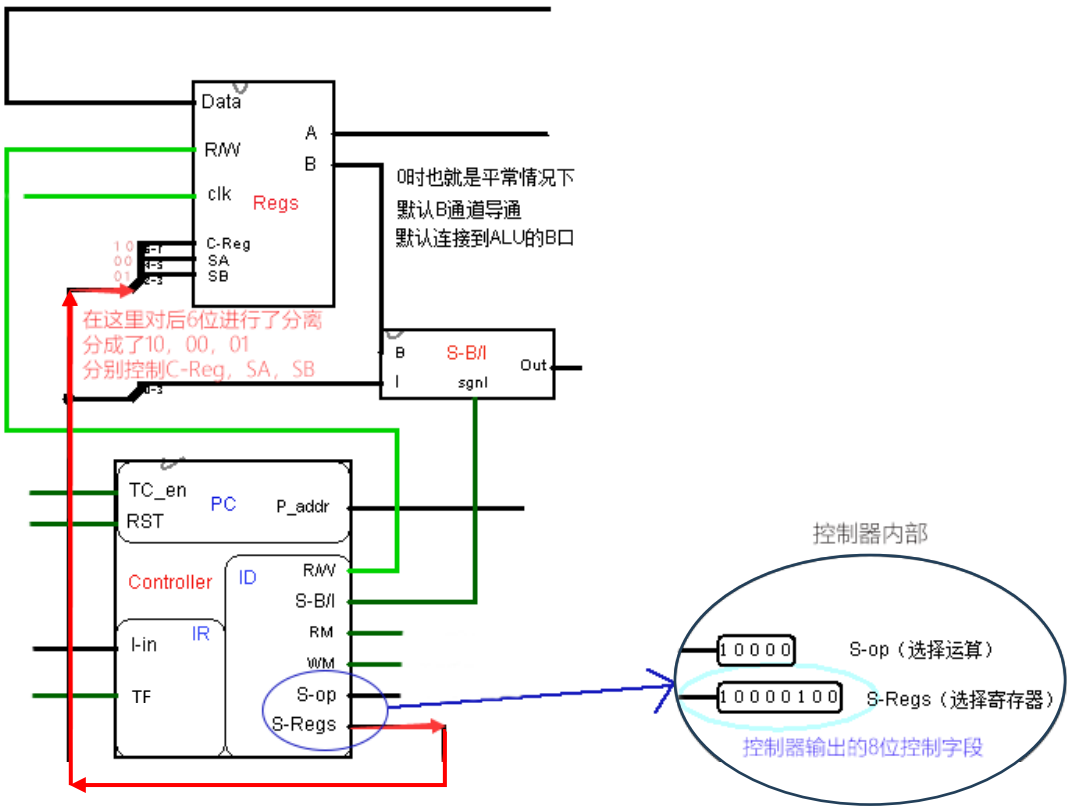



图 1 寄存器和控制器

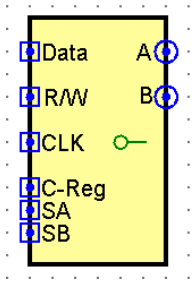
# 1、 构建 Regs 寄存器组

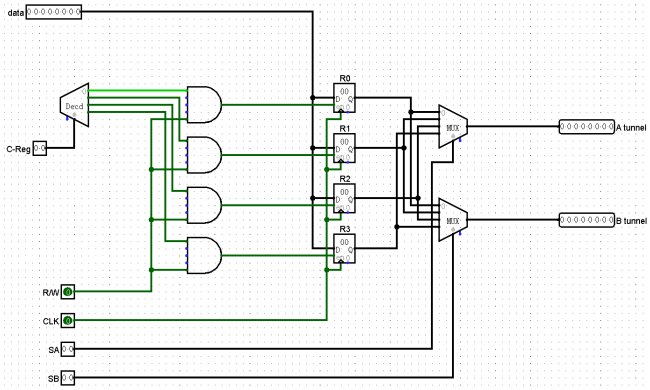
**（1）Regs 寄存器组模块的作用：**

- 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。
- 寄存器通过 C-Reg 位选择：本系统设有 4 个寄存器（R0~R3），可以利用 2 位编码（C-Reg）选定某一个寄存器。选定规则为：

| C-Reg | 所选定的寄存器 |
|---|---|
| 00 | R0 |
| 01 | R1 |
| 10 | R2 |
| 11 | R3 |

- 选择相应寄存器（R0~R3）并写入数据 Data，并将寄存器内容输出至相应通道 A/B。




**（2）对寄存器的写入：**

需要对寄存器写入，就需要数据输入端 Data，此外，还需要指定写入哪个寄存器。可以由 C-Reg 位决定，4 个寄存器需要 2 位 C-Reg，8 个寄存器需要 3 位。

思考：如果一个寄存器都不需要写入怎么办？时钟是不会变的，一旦时钟来了就会写入，这时候，使能端就发挥作用了，可以利用使能端结合地址选择一个寄存器，利用读写端（R/W）和使能端（CLK）结合实现要不要写入。

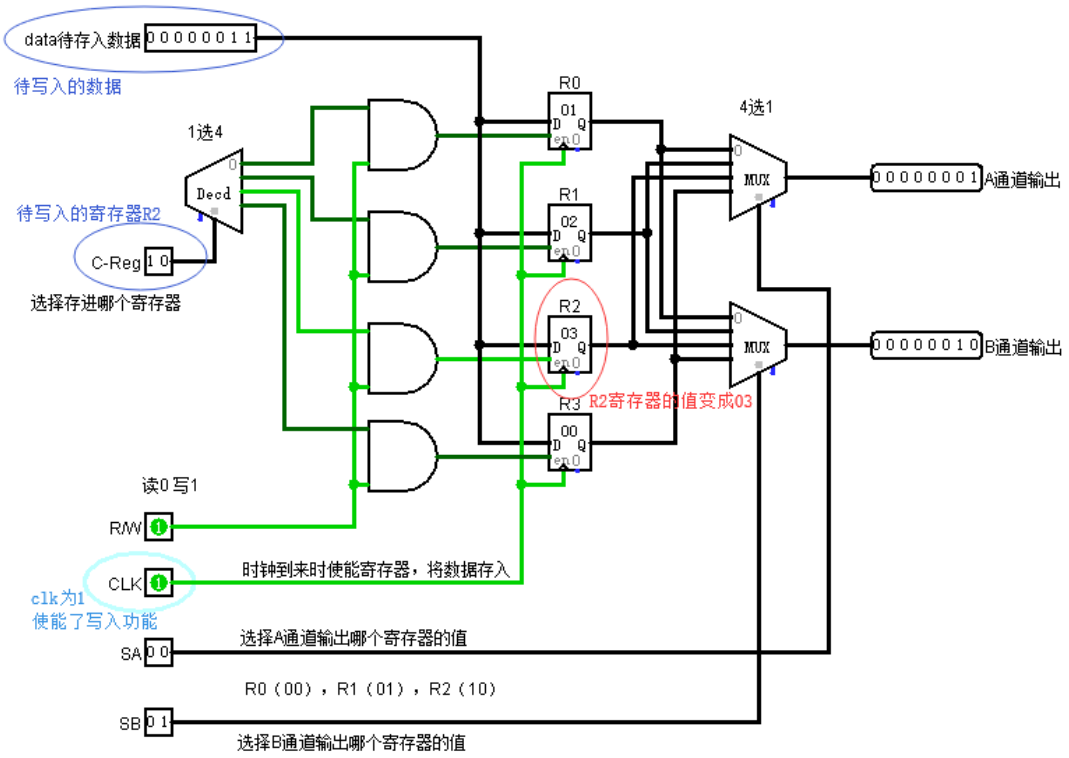
**（3）对寄存器的读取：**

时钟信号作为使能端，读写端控制门电路
参考下一页

寄存器的输出端可以持续不断的输出。而根据前面 ALU 的设计，可以从寄存器取两个操作数输入 ALU。因此，寄存器组需要有两个输出端，这两个输出要能够分别连接到 ALU 的两个输入端，也就是要实现选择两个寄存器分别输出

到 A 端和 B 端。A、B 两个输出通道可以分别设置两个数据选择器，每个选择器能够选择任意一个寄存器。

由上述描述，Regs 模块内部组成可以如下图所示：




图中各引脚信号的说明如下：

（1）输入信号：

- **data**：待存入的 8 位二进制数据，可以连接 ALU 模块的 output 输出端

（2）控制位：寄存器组模块共 8 位控制位

- **R/W（1 位）**：选择对寄存器读取还是写入，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，即 CLK=1 时，可读写寄存器；CLK=0 时，不能读写寄存器
- **C-Reg（2 位）**：（Choose-Register）当选择写入功能时，选择将要写进数据的寄存器，00 表示选择 R0，01-R1，10-R2，11-R3
- **SA（2 位）**：当选择读寄存器功能时，选择哪个寄存器的值输出到 A 通道，00 表示选择 R0，01-R1，10-R2，11-R3
- **SB（2 位）**：当选择读寄存器功能时，选择哪个寄存器的值输出到 B 通道，00 表示选择 R0，01-R1，10-R2，11-R3

（3）寄存器组：

- 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

（4）输出：寄存器组共有一个数据输入端和两个数据输出端

- A 通道有两个功能 ：
  - 输出数值给 ALU 的 A 口参与运算
  - 输出地址给内存 RAM 的 A（Adress）口，作为存进 RAM 的位置
- B 通道也有两个功能
  - 输出数值给 ALU 的 B 口参与运算
  - 输出数据给内存 RAM 的 D（Data）口，作为存进 RAM 的数值

（5）控制位详解：

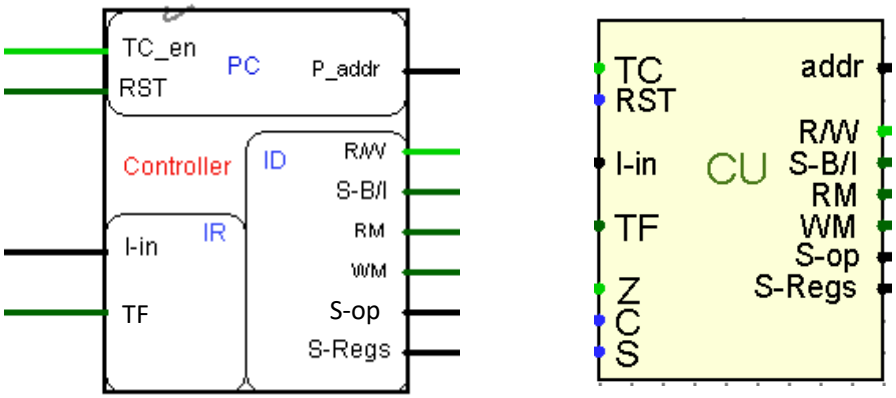
由上可知，共 8 个控制位；

- 其中 C-Reg（2 位）、SA（2 位）、SB（2 位）这 6 个控制位的控制字段来自**控制器模块（CU）**中 S-Regs 字段的 2~7 位，也就是图 1 中红线标注的部分。
- 而 R/W（1 位）控制位由控制器 CU 传出的控制字段 R/W 决定。
- CLK 控制位是由 CLK 模块的第四个时钟周期发送的 CLK 信号给出，因此寄存器的输入输出在一个机器周期的第 4 个时钟周期完成。

# 2、控制器模块

（1）作用

运算器和控制器是 CPU 的核心部件。控制器的功能主要有取指、分析指令、控制各寄存器执行微指令，分解时序等。




（2）组成：

- PC（程序计数器）：负责保存指令执行的地址。
- IR（指令寄存器）：从 PC 所指向的内存中读出一条指令，存入指令寄存器中。
- ID（指令译码器）：将指令寄存器的相应位送入指令译码器（操作码译码、地址译码等）。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。

## 2.1 程序计数器（PC）的设计

（1）作用：

PC 的作用在于保存指令的地址；

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，所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 PC 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，直接修改 PC 的值
- 对于相对跳转，则通过 PC 加上或者减去某个值实现，一般来说**大多数情况下用的都是相对跳转**。

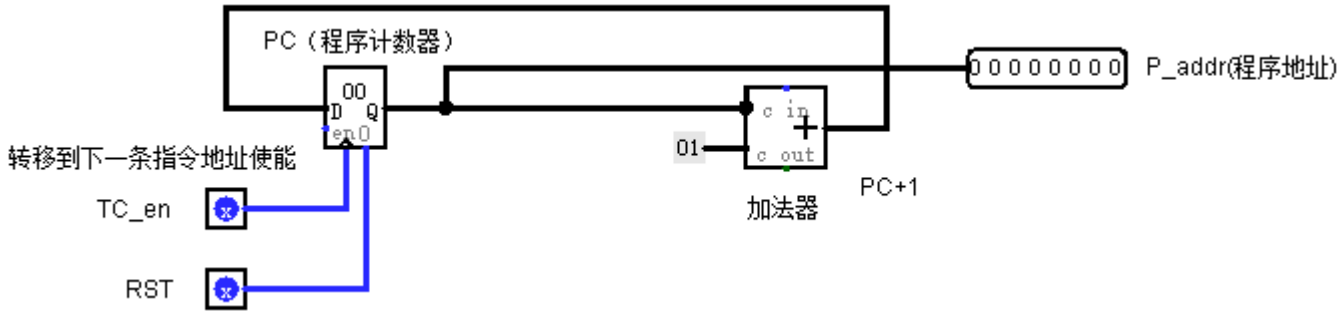
（2）设计实现：

**下面两种 PC 的设计方案，同学可任选一种：**

1）顺序结构程序的 PC 设计（后期只能实现顺序结构程序设计）：

如果系统只需要执行简单的顺序结构程序，则每次 PC 只需要加 1。因此，程序计数器 PC 可以用寄存器、加法器实现。

寄存器中为当前 PC 值，通过加法器＋常数 1，获取下一条指令的地址。

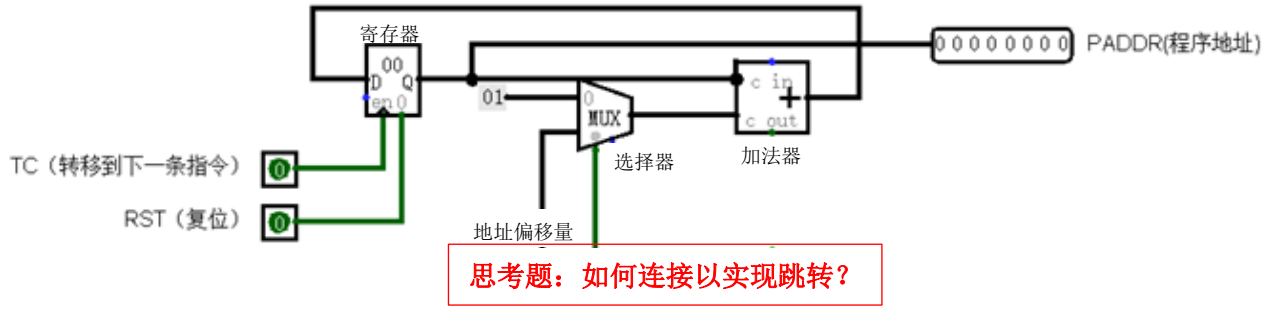



Tc_en：PC+1 的使能，实际作用是将加 1 后的 PC 输出到 P_addr

RST：复位

2）带跳转结构的 PC 设计（后期能实现分支和循环结构程序设计）：

程序计数器 PC 可以用寄存器、加法器和数据选择器实现。




① 如果程序顺序执行，则 PC=PC+1，所加值为 MUX 选择的常数 01

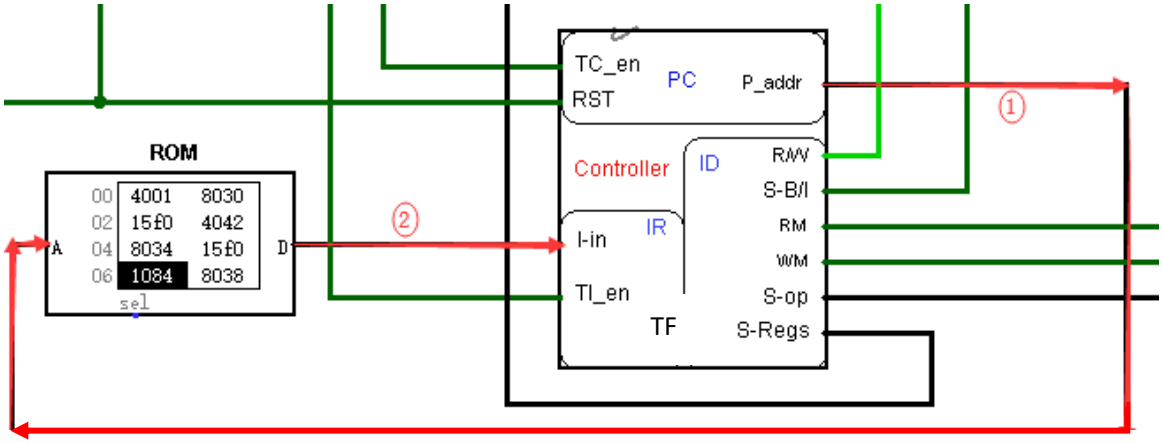
② 如果程序要跳转，则 PC=PC+地址偏移量

## 2.2 指令寄存器（IR）的设计

如下图所示，程序计数器 PC 产生的地址 P_addr 经历如下过程：

① P_addr（即待取指令的地址）传送给存储器（ROM）；

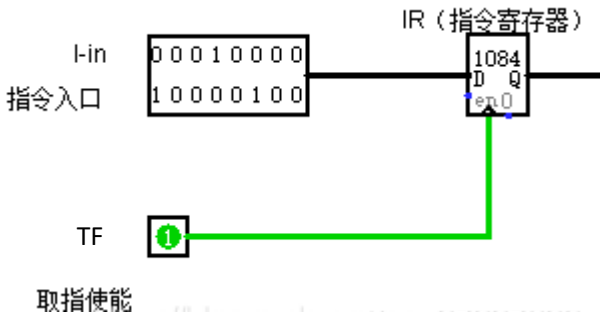
② ROM 会将对应地址中的 16 进制指令传送入 IR 寄存器的 I-in 入口；




此时指令逻辑上是暂时存储在 I_in 中，还没有进入 IR，需要 TF 取指使能口变成高电平，数据才会进入 IR 指令寄存器

（指令寄存器与一般的寄存器功能一样，用于暂时存放指令）

实现电路如下：




## 2.3 指令译码器（ID）的设计

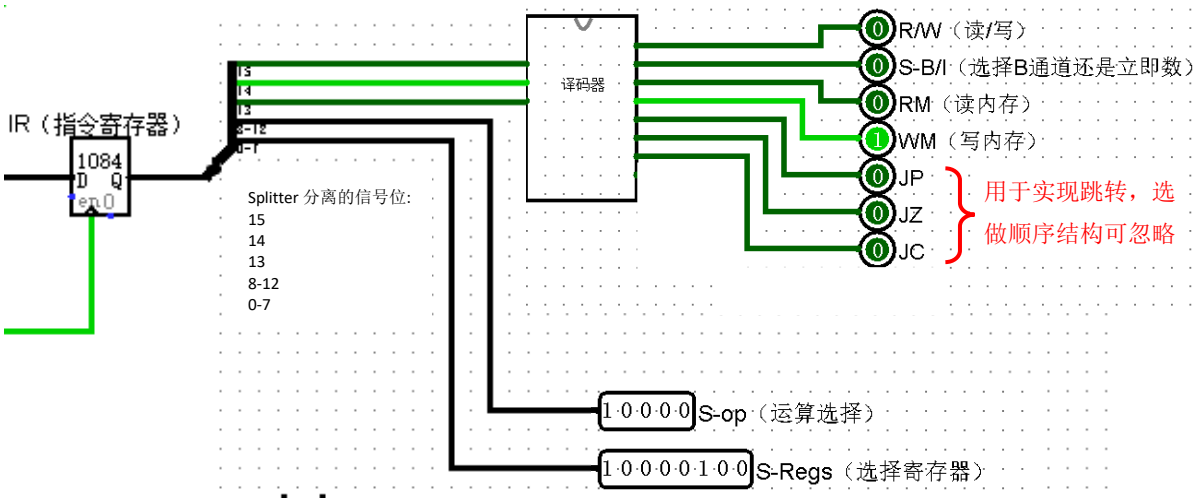
在设计指令译码器之前，首先要明确指令格式，这里所使用的指令集设计标准类似 MIPS，只不过 MIPS 指令集为 32 位，这里为 16 位。

这里的 16 位指令是根据算术逻辑单元（ALU）与寄存器组（Regs）的控制字段设计的，指令部分在下一次实验中会有详细解释，先介绍指令译码器的设计部分。

（1）作用：

指令译码器的作用在于解释翻译指令，指令在 CPU 中实际是一个二进制序列，由**操作码+寻址方式+操作数**组成，分别代表要进行什么操作，以及所操作的具体数据和寄存器，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。

（2）设计实现如下：




注：指令译码器 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，左边的分离器 splitter 起到了译码器作用。

16 位的指令由一个 splitter 逐位分离之后，变成不同的控制字段输出，其中：

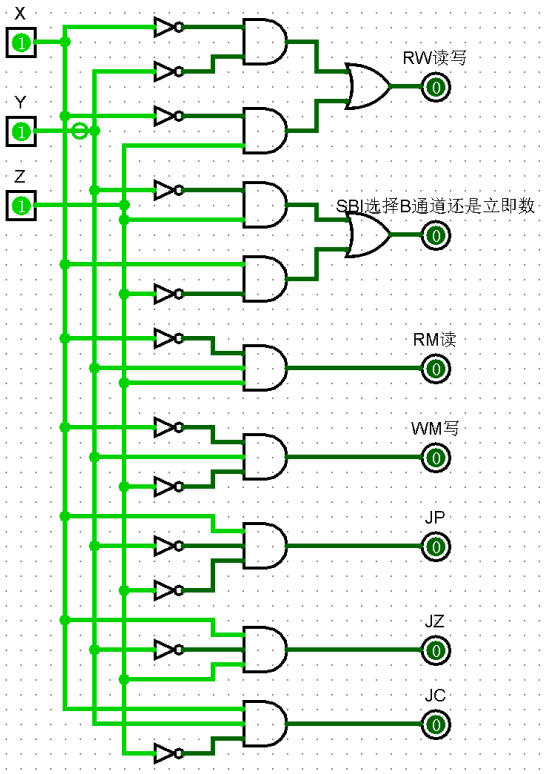
- 0-7 位：S-Regs，这 8 位主要控制寄存器或者立即数的选择，在 Regs 模块的设计中有详细的介绍，在下一章的指令格式详解中也会有详细介绍；
- 8-12 位：S-op，这 5 位主要控制 ALU，功能是运算方式的选择，对应 ALU 的 S3~S0 及 Cin，例如 1 0000 代表选择加法运算；
- 13~15 位：这 3 个位组合决定操作数的寻址方式和控制功能，选做顺序结构程序设计的同学可忽略红色三项的功能实现。

| 寻址方式<br>和控制功能 | 3 位组合 | R/W | S-B/I | RM | WM | JP | JZ | JC |
|---|---|---|---|---|---|---|---|---|
| 选择两个寄存器进行运算（RW，可读可写） | 000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 寄存器和立即数进行运算 | 001 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 写主存 | 010 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 读主存 | 011 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 无条件转移 JP | 100 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 条件转移(相等转) JZ | 101 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 条件转移(小于转) JC | 110 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 选择两个寄存器进行运算（RO，只读） | 111 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

因此需要将这 3 位的字段解析成每个控制字段的对应输出即可。

在实际电路中可以用一个**译码器**的方式实现：将 3 位二进制译成多个输出，实际电路可以由真值表得来。（操作方法可参考实验 2 中的“根据真值表构建 3-8 译码器”）




## 2.4 CLK 模块

CLK 模块在整个 CPU 中的作用是将一路 CLK 信号变成 4 个循环的时钟周期，可分别称为 T0、T1、T2、T3，可以看作是时序，不同的 T 周期启动不同的模块工作。4 个时钟周期组成一条指令的执行周期，也就是一个指令周期。

一个指令周期的 4 个时钟周期具体操作为：

① T0 时钟周期从 ROM 中取指令，在时钟周期的上升沿把指令送入指令寄存器。

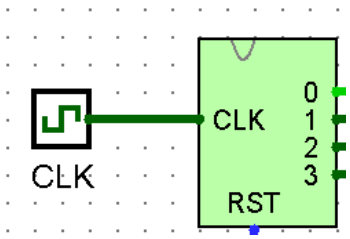
② T1 时钟周期进行指令译码。

③ T2 时钟周期执行指令，并把运算器的结果或者 RAM 读出的数据送入寄存器组，所以寄存器组需要第三个时钟上升沿。

④ T3 时钟周期需要修改程序计数器。

将一个指令周期分解成四个时钟周期，即将一个时钟分解成四个时钟，可以使用**时钟发生器、一个四进制计数器和一个译码器**实现。

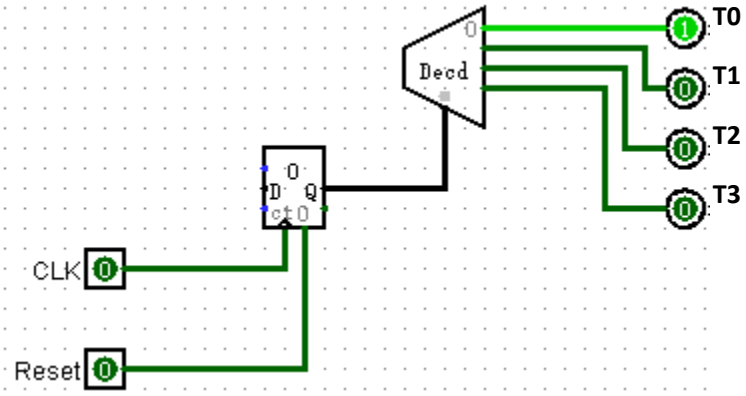
电路预览如下：




CLK 模块的设计实现如下：

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。

计数器的输出连接到一个 Decoder 解码器上，将计数输出转变成四个不同的 CLK 输出，构成 4 个时钟周期，用于完成一个基本指令周期。




## 2.5 S-B/I 模块

S-B/I 模块意为 Select-$B_{tunnel}$/$Immediate_{tunnel}$，意为选择寄存器 B 通道还是立即数。

（1）作用：

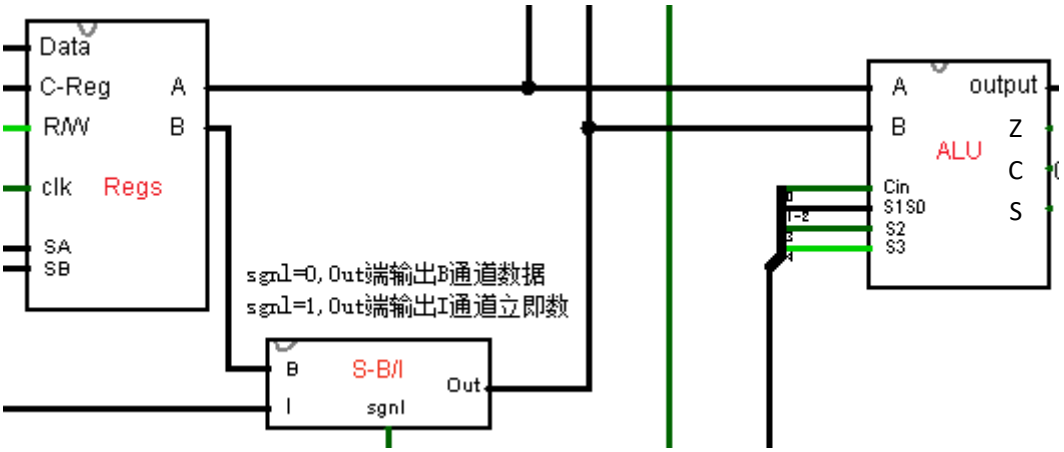
在实际的指令中有操作数都为寄存器的指令，也有操作数是立即数的指令，例如，赋值指令：

MOV R0, 2

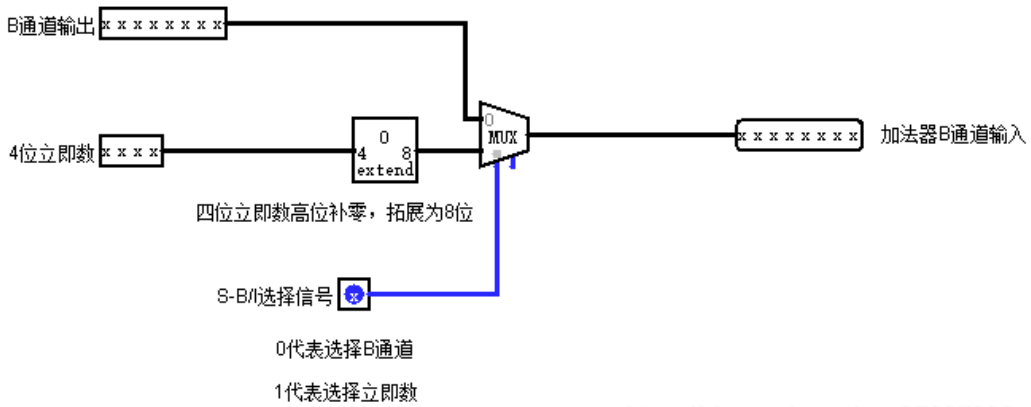
需要将立即数 2 写进 R0 寄存器，要考虑怎么将立即数传给 ALU。可以将要写入的立即数放在指令中作为指令的一部分，在解码之后，传送给立即数通道（B/I 中的 I 通道），再通过立即数通道传给 ALU。

因此，将寄存器的 B 通道输出与立即数通道复用，通过一个选择位选择输入给 ALU 中 B 口的数据来自寄存器还是来自指令中的立即数。

（2）模块连接如下：




（3）S-B/I 模块的实现如下：



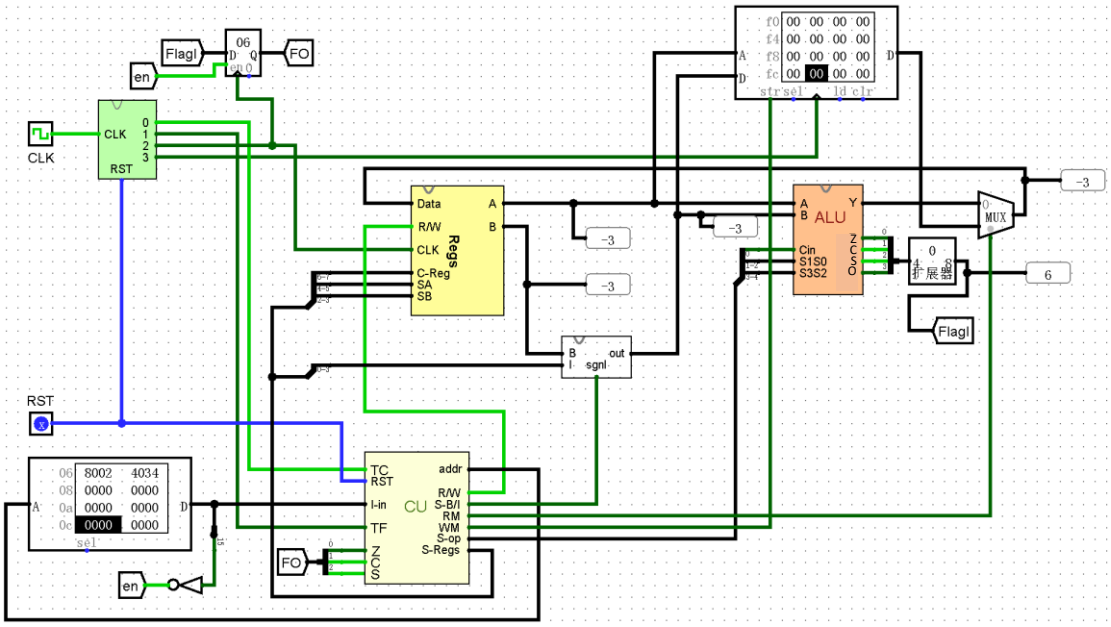

- B 通道数据输入：一个 8 位的 Regs 模块的 B 通道数据输入
- I 立即数通道数据输入：
  - 指令解码后 4 位的立即数输入
  - 因为输出为 8 位，因此还需要一个 4 位扩展成 8 位的位扩展器
- S-B/I 的选择信号：
  - 0 代表默认选择 B 通道数据，1 代表选择立即数通道

- 一个 2 选 1 的选择器，多路选择器
- 一个 8 位数据输出
- 输出的数据连接到 ALU 的 B 数据输入口

# 3、系统连接

模块解析：

- **CLK**：时钟模块，这里一个指令周期被细分为四个时钟周期，以按时序分步执行微指令
- **RST**：复位模块，使系统回到初始状态
- **ROM**：存放指令，这里每一条指令以四位 16 进制序列的形式存放在 ROM 中
- **CU**：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）
- **Regs**：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。
- **S-B/I**：通路选择器，可选择 B 通道输出或者 I 立即数通道输出
- **ALU**：算术逻辑单元，主要计算 A、B 通道的输入，并将计算结果从 output 输出到 Regs 的 data 端口。
- **RAM**：内存，计算机系统中存放数据的存储器，寄存器组中的寄存器只是暂时存放数据。
- **标志位**：Z、C、S、O 等标志位应保存在标志寄存器 Flag 中，运算类指令可对 Flag 写，转移类指令只能读取标志位。（顺序结构可不考虑标志位的保存和连接）


CLK
RST
Flagl
FO
en
Data
R/W
CLK
C-Reg
SA
SB
A
B
Regs
ALU
Cin
S1S0
S3S2
Y
Z
C
O
MUX
扩展器
-3
6
out
sgnl
TC
RST
I-in
TF
CU
addr
R/W
S-B/I
RM
WM
S-op
S-Regs
Z
C
S
sel
clr