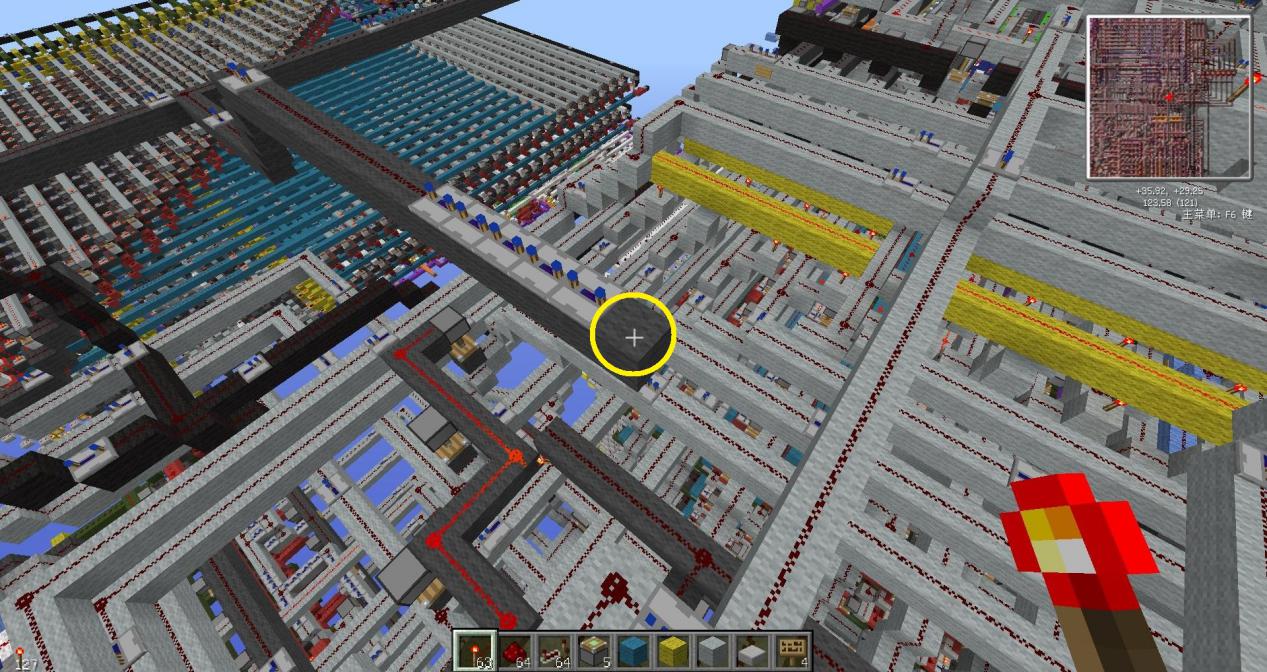
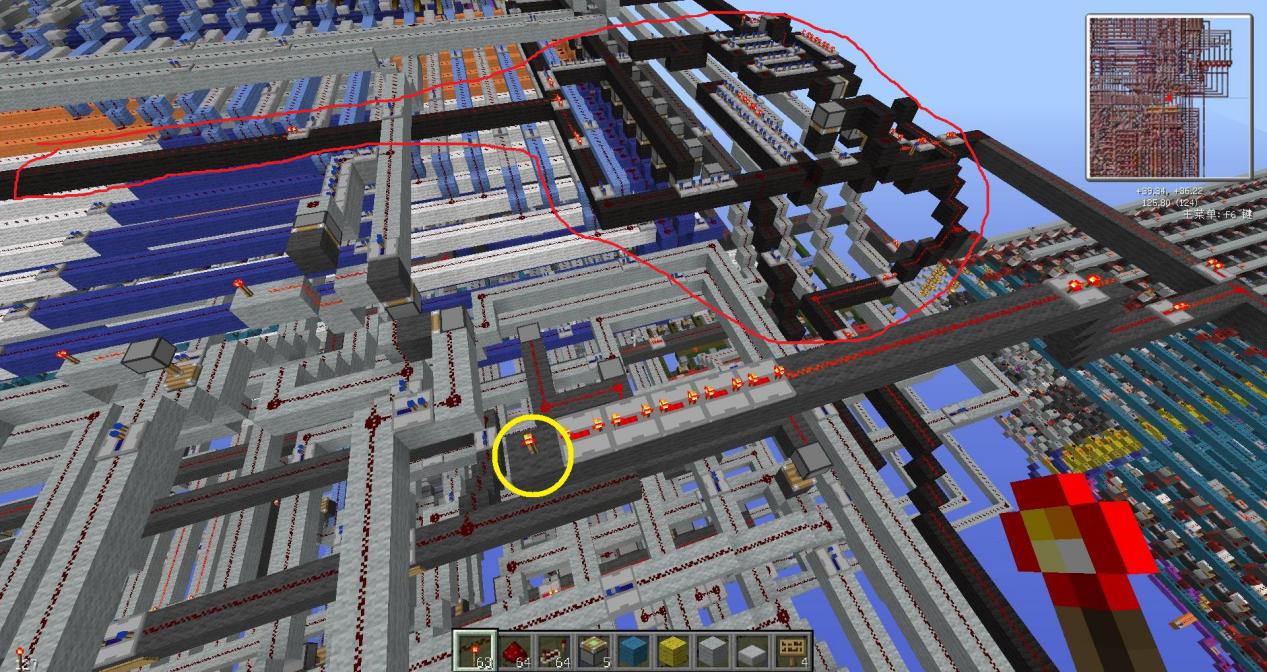
**关于CPU功能的介绍**

Alpha21016项目里原本规划了30条指令，但由于时间和精力原因只完成了不到10种指令。我在”Alpha21016最终版演示存档”存档的程序储存器里预先写好了一段代码，一共5条指令，功能是从数据储存器里取出一个字的数据，然后做了2次寄存器转移复制，最后将数据存回数据储存器的另一个地址。

在下图中画黄圈的位置放置信号源就可以启动CPU，敲掉信号源就可以停止CPU。



启动后注意下图红圈内黑色的电路，这个黑色的结构贯穿了整个总线，是用于指令周期结束阶段对总线进行清存的，每个周期的时间是7秒。



启动CPU后就可以看到预先写好的5条指令执行的过程，但是CPU没有对应的GUI输出显示，所以除了电路信号流转之外没有其他辅助的东西。

指令流水线的过程如下面几张图所示：

启动CPU后首先被激活的部分是下图黄圈中的时钟发生器，时钟发生器会发出各种同步信号到各相关结构。



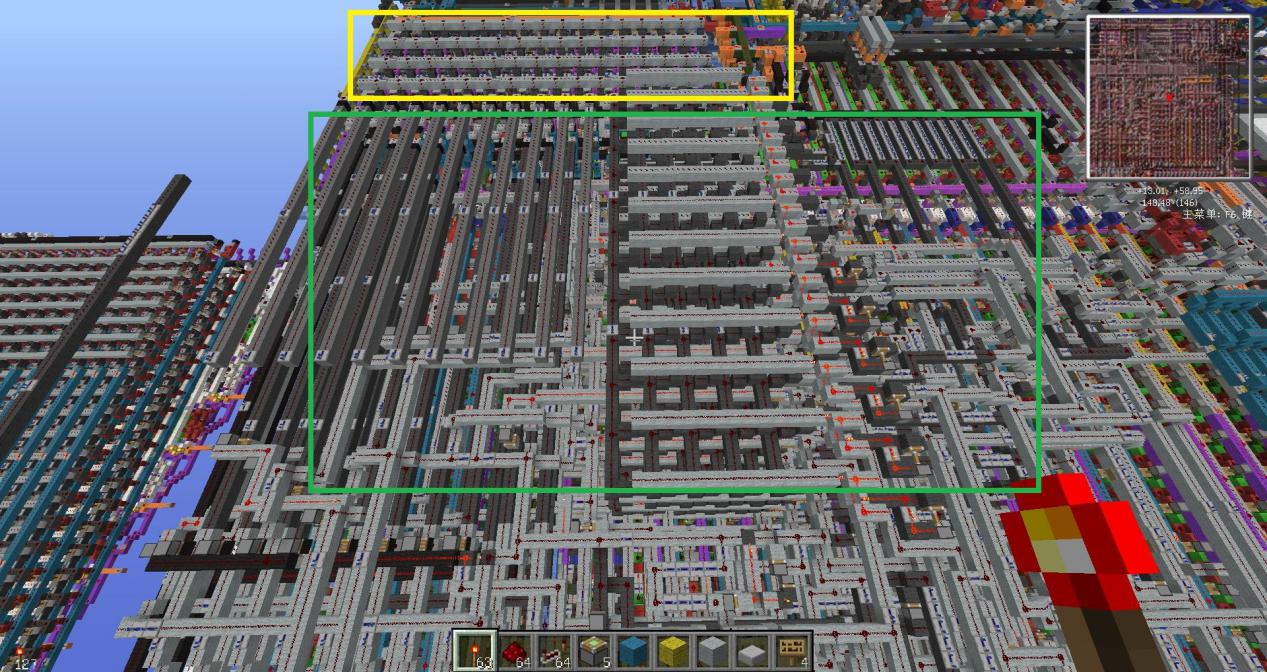
指令流水线分为三级。第一级是取指。第一个开始运转的功能电路是下图黄圈中的程序计数器，这个电路的完整版可以在教学存档中找到。



程序计数器从下图黄圈中的程序储存器中依次取指令来执行，黄圈最左边的是000000000号指令，但是程序计数器目前设定默认是从二进制编号第000000001号指令开始执行的，也就是左数第二条。所以物理意义上的第一条指令(000000000)并不执行，执行的是第二条指令(000000001)。



当指令被取出后就开始了指令流水线的第二级。首先指令会送到指令缓冲队列，是下图的黄圈部分。当指令缓冲队列中未处理的指令达到4条时，就给程序计数器发出阻塞的信号，停止下一条指令的取指。然后每当指令时钟周期一个循环结束，会判断上一条指令是否执行完，如果执行完，指令缓冲队列会将队列最前端的一条指令POP到译码器。译码器分为预译码和正式译码。预译码主要是为分支预测和跳转指令准备的，目前没有完工。正式译码是为所有指令工作的，是下图的绿圈部分。在上一条指令执行的同时，当前指令的译码就已经在进行了，这就是流水线的意义。



当译码工作完成后，各种控制端的信号就将传输到下图黄圈中的指令发射端，等待当前指令周期/寄存器周期的结束，然后下一个周期开始，指令发射端就将控制各种储存器和ALU进行工作。这就开始了指令流水线的第三级。



数据储存器的低地址是下图绿圈代表的最外侧的一层，编写指令如果想要方便调试的话，就会利用外侧的这些低地址，因为玩家飞到这里很快就能找到对应的储存器进行观察。如果选更高一些的地址的话，就要往内侧去找，里面的空间较狭窄，很难快速定位。



最后一张图就是ALU和部分寄存器。蓝圈的部分是X寄存器，黄圈的部分是Y寄存器，单操作数的计算指令很多都是用X寄存器存操作数和结果，双操作数的计算指令用X和Y两个寄存器存操作数，X寄存器存结果。X寄存器同时是主存取数据后默认传送的地址，也就是说假如一条指令从数据储存器的地址000000001取了一个字节的数据，这个数据会存到X寄存器的低8位。而Y寄存器则是作为数据存入数据储存器的中转。



另外键盘，显示器和时钟的电路功能是差不多全部完成的，启动按钮也放在了较为显眼的位置，可以摸索一下如何使用。不过这些部分和CPU是没有交互的，CPU还需要做一个和外部设备进行通信的桥接装置，比较直接的想法就是做一个buffer的队列结构，用于处理CPU和外部设备的信号。