单周期 CPU 设计实验报告

设计者学号：14241217
设计者姓名：张聿恩

一. 实验要求

1. 应用 logisim 实现 MIPS 简化指令集单周期 CPU，包括如下指令：实现 9 条指令子集：ori，lui，addu，sub，slt，bne，lw，sw，stp。指令的定义、操作和操作码请参考学习通资料中的 MIPS 指令集手册。假设不会产生溢出。注意：stp 指令不是 MIPS 指令，功能是停机，即时钟停止输入，所有控制信号清零。

2. 应用已经完成的 CPU 部件模块。

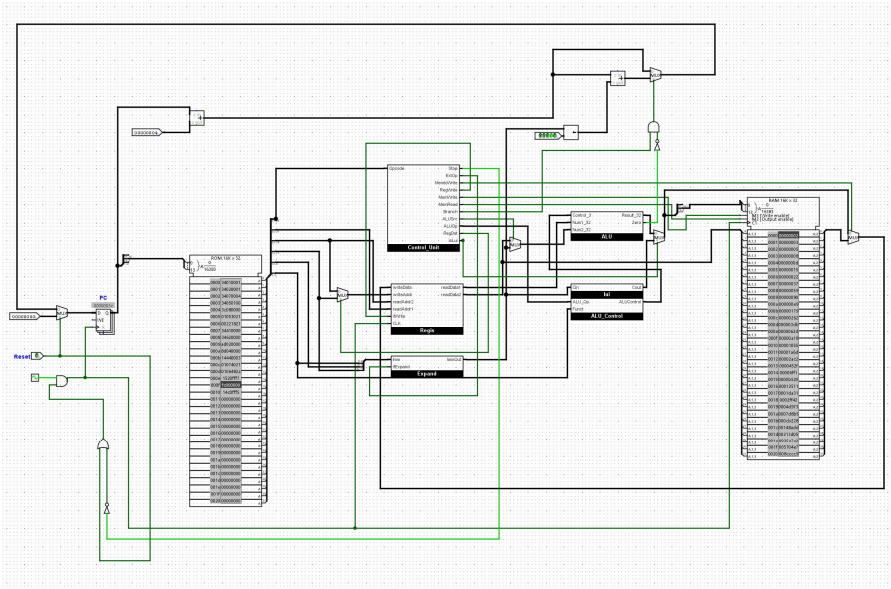
3. 指令存储器使用 ROM，数据存储器使用 RAM。数据存储器使用异步模式。RAM 和 ROM 的大小均为 64KB，按照字节编地址，按字访问，要求地址边界对齐。即使用 16K*32 位的 ROM 和 RAM。程序加载从 0x0000 开始。全都使用大尾端模式。

4. 汇编语言程序中，使用$x 表示寄存器编号为 x。规定$0 中的值只能为 0，编程中只读不写。其他寄存器均可在程序中使用。

5. 设置一个计数器和一组十六进制显示器，显示当前周期数。当程序停止时（即停机时），该显示器显示运行的总周期数。

二. 实验设计

请给出 main 的截图，并回答如下问题。



（此处插入 main 截图）

问题 1：你是如何实现 lui 指令的。

我在 CU 里面加了一个接口，叫做 isLui，这个接口告诉数据通路，该命令是否为 lui，如果是，则在输入 RAM 的时候，进行判断，isLui 是高电平则输入经过移位的数，否则输入指令操作后的数

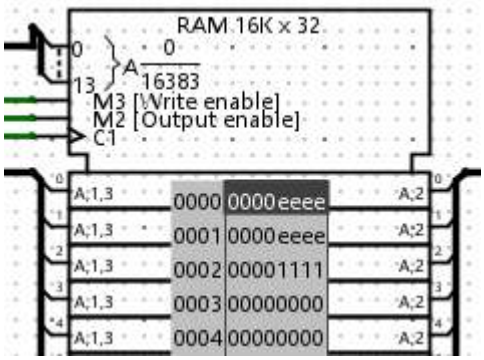
问题 2：你是如何处理按符号位和按 0 扩展两种扩展方式的。

我在 CU 里面加了一个接口，叫做 ExtOp，旨在判断是否需要扩展位。这个 ExtOp 在对命令进行编码的时候，已经编码到命令（CU）的 ROM 里面。在 ExControl 里面，有两个输入，分别是低 16 位的数，以及一个 ExtOp 的判断输入。ExControl 内部有一个判断，如果需要符号扩展（0），则输出经过位扩展（符号）的 32 位数据；如果需要零位扩展（1），则输出补充前置 0 的 32 位数据。

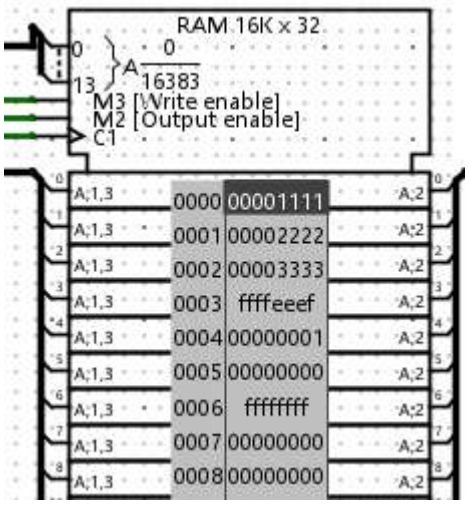
三． 实验测试

请完成单周期第一组测试程序，并将结果依次截图插入。

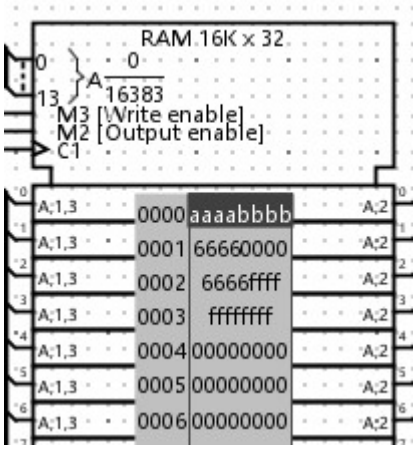
（1）程序 1 测试结果截图。



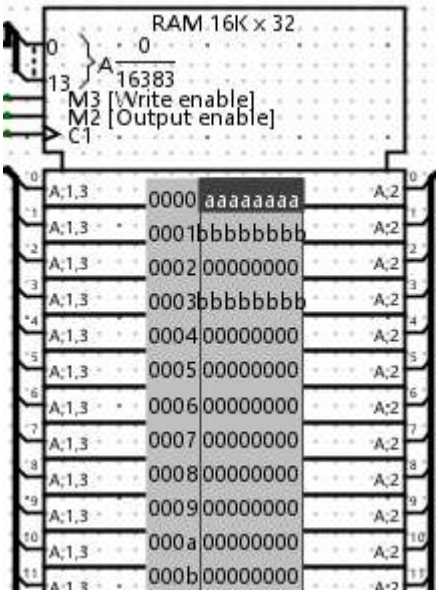
（2）程序 2 测试结果截图。



（3）程序 3 测试结果截图。



（4）程序 4 测试结果截图。



（5）程序 5 测试结果截图。

FPGA 窗口 帮助 HEX 编辑

0000	00000002	00000003	00000005	00000008	0000000d	00000015	00000022	00000037
0008	00000059	00000090	000000e9	00000179	00000262	000003db	0000063d	00000a18
0010	00001055	00001a6d	00002ac2	0000452f	00006ff1	0000b520	00012511	0001da31
0018	0002ff42	0004d973	0007d8b5	000cb228	00148add	00213d05	0035c7e2	005704e7
0020	008cccc9	00e3d1b0	01709e79	02547029	03c50ea2	06197ecb	09de8d6d	0ff80c38
0028	19d699a5	29cea5dd	43a53f82	6d73e55f	b11924e1	1e8d0a40	cfa62f21	ee333961
0030	bdd96882	ac0ca1e3	69e60a65	15f2ac48	7fd8b6ad	95cb62f5	15a419a2	ab6f7c97
0038	c1139639	6c8312d0	2d96a909	9a19bbd9	c7b064e2	61ca20bb	297a859d	8b44a658
0040	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
0048	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
0050	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

四． 实验体会

（说说你在实验过程中，遇到的难点是什么，你是如何解决的。基础知识不够不算）

1、 lui 的问题

我一开始对 lui 理解错了，以为是在 ALU 里面，于是在一开始设置命令编码（CU）的时

候没有充分考虑 ExtOp 的位数就开始编码，导致已经编码完毕之后发现 lui 出现问题。这时候再去修改命令编码已经不太现实（工作量太大），于是经过思考，我发现 lui 命令只需要把 isLui 输出(判断是否为 lui 命令)，然后在 ALU 输出的时候进行判断即可，于是临时更改了 CU 的接口，并且在 ALU 输出部分加了一个 lui 判断，完美解决问题。

2、 bne 的问题

在测试实验四的时候，我发现出现了很离奇的事情：RAM 中出现了 55555554 这样的数字。没有输入输出是这个数，于是我对每个时钟周期依次暂停，用左上角的小手图标工具查看线路内部的传输数值，最终发现是 ALU 的问题：ALU 会输出 sub（bne 的 ALU 命令），导致出现 55…54 这样的数字。再经过排查，发现是 bne 编码的时候，我把 MemRead 设为 1 了，于是调整了 CU 的编码，然后顺利解决。

这个实验给我最大的体会就是 ROM（命令编码）的重要性，几乎每次调试失败都是因为 ROM 编码写错/忽略条件导致的。还有就是在编码命令的时候一定要考虑周全，否则一旦编码就不好再改动了。