LC3

PB15111679 庄涛

平台

EDA工具及版本号：ISE 14.7

开发板信号：Spartan6 XC6LX16-CS324

外设

Nexys3上的5个按键（上下左右中），8个开关，4个7段数码管

技术点

之前所有的实验的技术点都有用到，其中ip核的功能是自己实现的，因为需要一些额外的信号。以下列举本次实验在之前几个实验中没有涉及到的技术点：

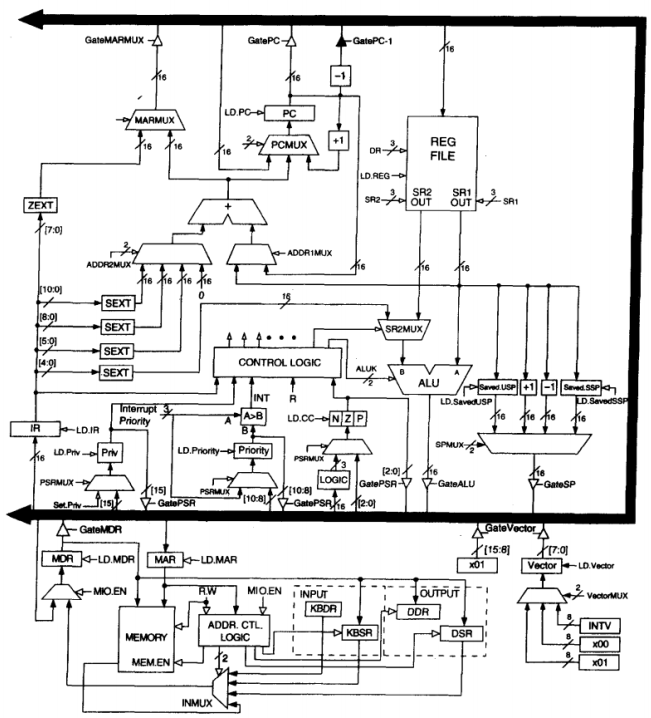
* + 冯诺依曼模型
  + LC3结构
  + 汇编指令
  + 内存映射
  + 中断机制
  + 栈
  + 微指令结构的状态机

原理

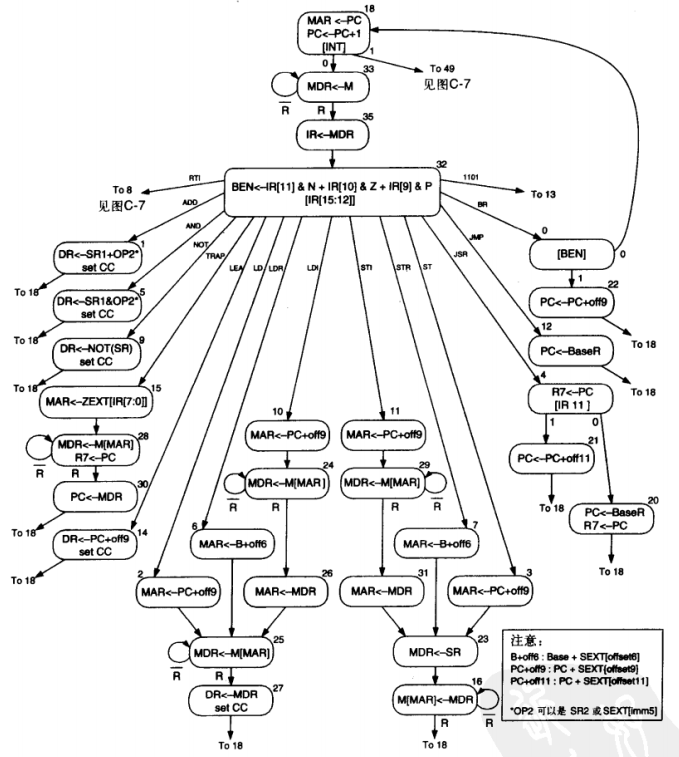
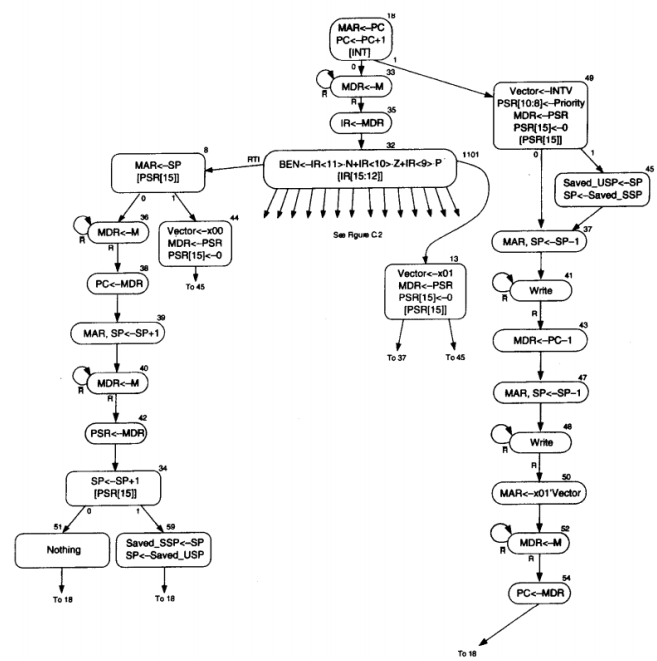
1. 基本原理

由于内容过多（内存映射，指令集结构，汇编指令集，中断机制，异常处理，状态机结构，数据通路），因此这里不再赘述，想了解基本原理的话请参考《Introduction to computing systems》的附录。

1. 数据通路



1. 状态图

1. 键盘

键盘实现采用的是nexys3 上的5个按键，使用了整流和优先编码方式，保证一次只有一个按键写入。KBDR = { 11·`b0, up, down, left, right, middle } 映射到x7E00，KBSR映射到x7E02。另外由于读写内存需要给 CPU提供反馈信号R(ready)，当需要读写键盘时，需要提供一个键盘的相应的R信号。

1. 显示器

显示器的实现采用的是nexy3上的7段数码管，下面是26个字母和0-9数字在7段数码管上的解决方案。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  |  |  |  |  |  |  |  |  |  |
| a/A | b/B | c/C | d/D | e/E | f/F | g/G | h/H | i/I | j/J |
|  |  |  |  |  |  |  |  |  |  |
| k/K | l/L | m/M | n/N | o/O | p/P | q/Q | r/R | s/S | t/T |
|  |  |  |  |  |  |  |  |  |  |
| u/U | v/V | w/W | x/X | y/Y | z/Z |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

由于7段数码管的表达能力有限，因此有些字母并不太像其原貌。如“k, m, s, t, v, w, x, z”，但以尽量做到相似或含义明显。比如“M”，在n上加一横杠说明是两个n，相应的还有“w”。“S”与“5”相似，因此为了对称，把“s”的中间去掉了，相应的还有“Z”；“x”难以表达，因此只显示了一个斜杠来代表。

DDR = { position, 6`b0, ASCII } 映射到x7E04。DSR映射到x7E06。ASCII大于或等于128的那些字符，与7段数码管的128种显示情况相对应，以方便任意显示。

同键盘一样，显示器也要提供相应的反馈信号R。

1. 开关

开关实现采用的是nexys3上的8个开关。{ 8`b, sw } 映射到x7E08，只可读不可写，一般用于字符或8位二进制数字输入，搭配按键一起使用。

1. 内存

由于nexys3上的存储资源有限，因此只用了大小为x8000 \* 16 bit的内存。

由地址控制逻辑来控制内存与外设内存映射的读写

1. 有限状态机的控制存储器

这里的实现比较繁琐。主要是《Introduction to computing systems》并没有提供微指令的具体内容，只能自己根据数据通路和状态图来自己一个一个地填写，然后在仿真中不断找出错误并修改，花费时间极大。下面给出微指令的详细内容。

001001001000000000000000000000xxxxxxxxxxxxxxxxx0xx

000001001000001100000000100000xx0001xxxxxxxx0000xx

000001100110000000000000010000xxxxxx010xx1xxxxx0xx

000001011110000000000000010000xxxxxx010xx1xxxxx0xx

001101010000001000000010000000xx0100xxxxxxxxxxx0xx

000001001000001100000000100000xx0001xxxxxxxx0010xx

000001100110000000000000010000xxxx01101xx1xxxxx0xx

000001011110000000000000010000xxxx01101xx1xxxxx0xx

010010010010000000000000100000xxxx10xxxxxxxxx110xx

000001001000001100000000100000xx0001xxxxxxxx0100xx

000001100010000000000000010000xxxxxx010xx1xxxxx0xx

000101000000000000000000000000xxxxxxxxxxxxxxxxx11x

00000100100000001000000000000010xx01100xxxxxxxx0xx

010010010101000000100100000010xxxxxxxxxxxx100xx0x0

000001001000001100000000010000xx00xx010xx1xx0xx0xx

000001110010000000000000010000xxxxxxxxxxx0xxxxx0xx

000101000000000000000000000000xxxxxxxxxxxxxxxxx11x

000000000000000000000000000000xxxxxxxxxxxxxxxxx0xx

01011000011000001000001000000000xxxxxxxxxxxxxxx0xx

000000000000000000000000000000xxxxxxxxxxxxxxxxx0xx

000001001000001010000010000000100101100xxxxxxxx0xx

000001001000001010000010000000100101100xxxxxxxx0xx

00000100100000001000000000000010xxxx010xxxxxxxx0xx

000001000001000000000000100000xxxx00x11xxxxxx110xx

000101100001000000000000000000xxxxxxxxxxxxxxxxx10x

000101100101000000000000000000xxxxxxxxxxxxxxxxx10x

000001100110000000000001000000xxxxxxxxxxxxxxxxx0xx

000101001000001100000001000000xx00xxxxxxxxxx0xx0xx

000101110001001000000010000000xx01xxxxxxxxxxxxx10x

000101110101000000000000000000xxxxxxxxxxxxxxxxx10x

00000100100000001000000100000001xxxxxxxxxxxxxxx0xx

000001011110000000000001000000xxxxxxxxxxxxxxxxx0xx

1xxxxxxxxx00010000000000000000xxxxxxxxxxxxxxxxx0xx

000110000101000000000000000000xxxxxxxxxxxxxxxxx10x

010011001100001000000000000001xx1010xxx10xxxxxx0xx

000010000000100000000001000000xxxxxxxxxxxxxxxxx0xx

000110010001000000000000000000xxxxxxxxxxxxxxxxx10x

000010100110001000000000000001xx1010xxx01xxxxxx0xx

00001001110000001000000100000001xxxxxxxxxxxxxxx0xx

000010100010001000000000000001xx1010xxx00xxxxxx0xx

000110100001000000000000000000xxxxxxxxxxxxxxxxx10x

000110100100000000000000000000xxxxxxxxxxxxxxxxx11x

000010001000000101100001000000xxxxxxxxxxxxxx1xx0xx

000010111101000000000000000100xxxxxxxxxxxxxxxxx0xx

000010110101000000000000000010xxxxxxxxxxxx01xxx0xx

000010010100001000001000000001xx1010xxx10xxxxxx0xx

000000000000000000000000000000xxxxxxxxxxxxxxxxx0xx

000011000010001000000000000001xx1010xxx01xxxxxx0xx

000111000000000000000000000000xxxxxxxxxxxxxxxxx11x

010010010101000001100100000010xxxxxxxxxxxx000xx0x0

000011010010000000000000001000xxxxxxxxxxxxxxxxx0xx

000001001000000000000000000000xxxxxxxxxxxxxxxxx0xx

000111010001000000000000000000xxxxxxxxxxxxxxxxx10x

000000000000000000000000000000xxxxxxxxxxxxxxxxx0xx

00000100100000001000000100000001xxxxxxxxxxxxxxx0xx

000000000000000000000000000000xxxxxxxxxxxxxxxxx0xx

000000000000000000000000000000xxxxxxxxxxxxxxxxx0xx

000000000000000000000000000000xxxxxxxxxxxxxxxxx0xx

000000000000000000000000000000xxxxxxxxxxxxxxxxx0xx

000001001000001000010000000001xx1010xxx11xxxxxx0xx

000000000000000000000000000000xxxxxxxxxxxxxxxxx0xx

000000000000000000000000000000xxxxxxxxxxxxxxxxx0xx

000000000000000000000000000000xxxxxxxxxxxxxxxxx0xx

000000000000000000000000000000xxxxxxxxxxxxxxxxx0xx

1. 总线

总线也就是BUS，在Verilog中将其设为inout 类型，并且在输出时，若没有输出数据，就输出高阻态。

1. 中断机制的补充

LC3的优先级是0，显示器的优先级是1，键盘的优先级是2。书中没有讲到如何使用中断机制，这里说明一下。从状态图中可看出，发生中断后，pc = mem[ {15`b0000\_0001\_00000,优先级} ]，因此我们只需先写好中断后想要执行的代码，然后把上述代码的首地址存储在 { 8`h01, 5`b0, 优先级 } 中即可。

1. 栈

总共有两个栈：用户栈US，权限模式栈SS

用户栈栈顶是x7000（x6999是第一个位置），权限栈栈顶是x3000（x2999是第一个位置），向地址较低处伸展。注意合理控制好存储数量，以防覆盖掉重要内容。

1. 系统函数的改写

由于nexys3的特殊性，需要专门改写适用于nexys3上LC3的系统函数

以下是相应内容

|  |  |  |
| --- | --- | --- |
| 陷入矢量 | 功能 | 详细描述 |
| X20 | 读键盘 | 循环检测KBSR[15]，当它为1时，读取KBDR到R0 |
| X21 | 读开关 | 循环检测KBSR[15]，当它为1时，读取KBDR，若KBDR==1（按了中键），那么读取SWDR到R0；否则重新开始。 |
| X22 | 显示字符 | 将R0中的ASCII对应的字符显示出来，位置由R1决定 |
| X23 | 输出四个字符 | 将mem[R0], mem[R0+1], mem[R0+2], mem[R0+3]依次显示出来 |
| X24 | 提示输入开关 | 首先在屏幕上提示“in=”并将输入的开关作为ASCII，将相应的字符回显出来 |
| X25 | 停机 | 在屏幕上显示“HALT”，程序终止运行，按下中键后，程序继续运行 |
| X26 | 输出字符串 | 在屏幕上显示首地址为R0的字符串，到16`h0000为止。其中，每显示4个字符，就等待用户按下中键，然后再显示接下来的字符串。 |
| X27 | 等待 | 程序将等待R0ms |

具体的汇编代码实现请查看program/sys.asm

1. PSR

图C-8中把PSR分成了三个部分：Priv, Priority, CC，实现时我把他们放在了一个16位寄存器里。特别要注意对PSR的更新，容易出错。

1. 勘误

* 微指令总共有50个控制信号，而书中说49个。
* 图C-8中CONTROL LOGIC并没有输出到SP2MUX的控制信号，实际上这个控制信号是IR[5]
* 图C-8中控制信号SR2实际上是IR[2:0]
* 系统函数中不建议使用固定位置来存储寄存器的值以空出位置给临时变量。因为当发生中断时，中断时调用的程序有可能去调用这些系统函数，这样就会导致该固定位置的内容被覆盖，使得中断返还时内容丢失，因此应该使用栈来存储那些寄存器的值。

电路功能

电路功能简单的说就是一个支持了一套汇编指令因而图灵完备的计算机。

以下是四个演示程序的功能描述，相应的代码在文件夹program中，相应的视频在文件夹vedio中，相应的烧写文件在文件夹bit中

1. 猜数字 (guess)

猜一个事先确定好的一个0-255的数字

把8个开关作为输入的二进制数字，设定好后按中键。

若输入小了，就显示MORE，并接受下一个输入

若输入大了，就显示LESS，并接受下一个输入

若输入正确，就显示DONE

1. 排序 ( sort )

依次输入四个0-9的数字，把8个开关作为输入（注意输入范围是0-9），设定好后按中键。输入四个数字后按中键，数码管上将显示从大到小的数字序列

1. 交互（hi）

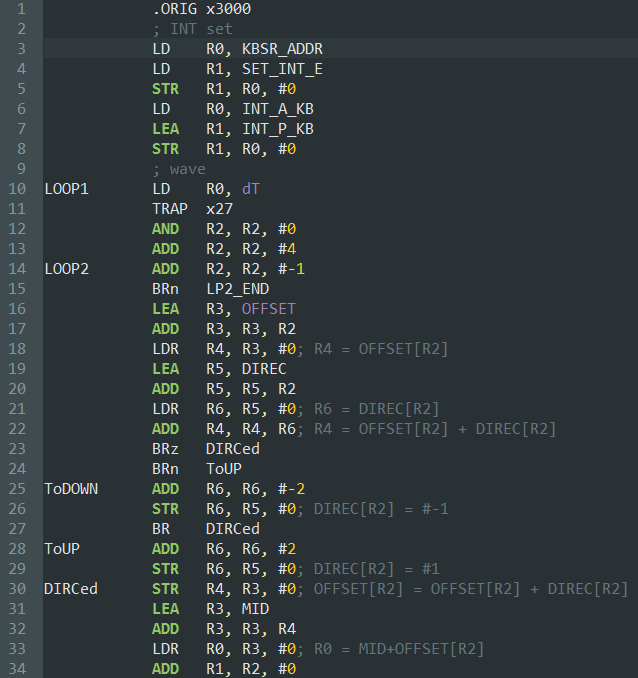
输入四位字符（最好由数字和字母组成）作为你的名字，然后不断按中键可查看输出的字符串，直到结束。输出的字符串为“Hi, xxxx. My name is LC3.”其中xxxx为输入的字符。

1. 动态波形图像+中断机制 (wave)

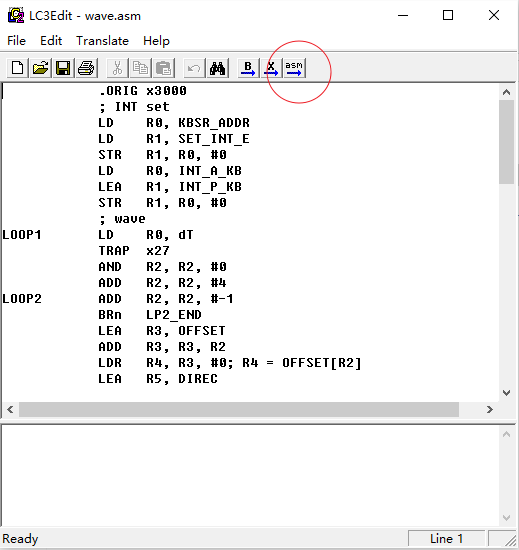
显示器将显示一个动态的波形图像，当按下中键后发生中断，中断时的执行大致就是显示HALT并暂停，当中键再次按下时，程序返回之前的程序即继续显示动态波形图像。这里要注意的是实现方式是中断而非轮询。

使用方法

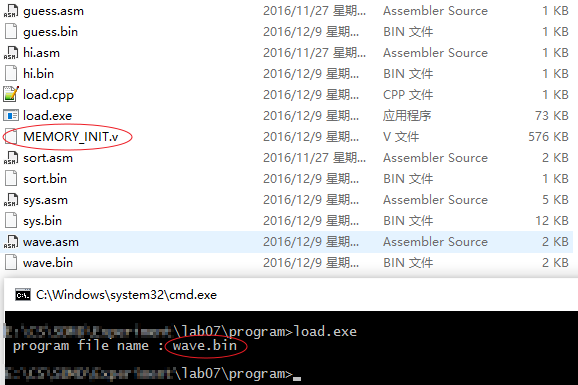
1. 编写一个汇编程序



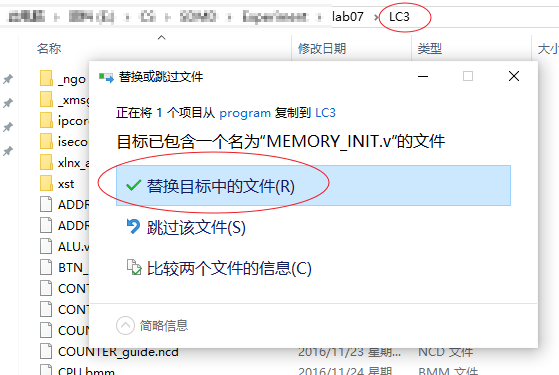
1. 使用program/LC3/LC3Edit.exe将汇编程序转换为LC3机器码xx.bin



1. 运行load.exe输入xx.bin，生成相应的内存初始化文件MEMORY\_INIT.v



1. 将生成的MEMORY\_INIT.v复制到文件夹LC3中，覆盖源文件



1. 在ISE14.7中编译得到lc3.bit
2. 将lc3.bit烧写入nexys3

总结

优点

图灵完备

缺点

没用使用标准外设（显示器，键盘、音响和鼠标）。

有大量的warning，然后发现有时会出现随机烧写错误，重新烧写问题就没了，估计问题出在warning，但只是烧写时随机出现，而在运行中不出现，而且时间有限，因此没有去修复这个不太关键的bug。

附录

所有相关的文件可以在[home.ustc.edu.cn/~ustczt/LC3](http://home.ustc.edu.cn/~ustczt/LC3)中获取