学号：19200135221　　姓名：胡恒宇　　实验日期：5.31 成绩评定：　　　　　。

|  |  |
| --- | --- |
| 实验名称： | 实验四 微程序控制器 |
| 实验内容：  1. 建立工程cpu，复制代码cpu.vhd、cpu\_def.vhd，编译，  2. 建立仿真文件，并对仿真结果加以说明和分析。 | |
| 实验目的：  1．理解微程序控制器的控制原理  2．进一步掌握指令流程和功能  3．了解掌握微程序控制器的设计思路和方法 | |
| 1. **数据通路**     **一个简单的微处理器，分成控制器和数据通路，执行简单的直接寻址指令。为了简化微处理器的设计，我们假定只有一条总线，且总线和所有数据通路组件的宽度都是8位。由于单总线可能会被许多不同的组件驱动，每个组件需要使用三态缓冲器以确保在任一时刻仅有一个组件能将有效数据送至总线上。我们用一个时钟驱动所有的时序块以确保设计完全同步。**  **由微处理器执行的程序与数据一起存储于存储器中，存储器地址寄存器（MAR）和数据寄存器（MDR）作为地址和数据信号在存储器与总线间的缓冲器。**  **算术逻辑单元（ALU）执行算术操作（ADD、SUB）。算术逻辑单元是一个组合模块，算术操作的结果保存于一个称为累加器（ACC）的寄存器中。ALU的输入是总线和ACC。ALU可以有更多的输出或者标志来表示ACC中结果的性质，比如为负，这些标志作为控制器的输入。**  **程序的不同指令在存储器中是顺序存储的。因此需要保存下一条将要执行的指令的地址。这是使用程序计数器（PC）来完成的，如果执行一个分支，程序就要跳出顺序执行，所以必须加载一个新的地址到PC中。**  **最后，从存储器中读取的指令需要保存与执行。指令寄存器（IR）保存当前指令。操作码的相应位输入至控制器，以产生相应的控制信号。**  **控制器产生许多控制信号，这些信号决定写总线的组件、读总线的寄存器以及执行的ALU操作。**   1. **指令系统？**      1. **指令格式怎样？写出五条指令的机器码（用二进制表示）。**   **微程序的编码采用直接编码方法，有15个控制信号，外加5位地址，所以一共20位。微处理器的所有数据通路宽度是8位，操作码是3位，只有直接寻址方式，所以说地址是5位。**  **Load 4 －> 01000 －> 00000100**  **Add 5 －> 01010 －> 01000101**  **Store 6 －> 01001 －> 00100110**  **Sub －> 01011**  **Bne 7 －> 01100 －>10000111**   1. **模拟时执行的程序。**        1. **微操作控制信号有哪些？**  | 控制信号 | 描述 | | --- | --- | | ACC\_bus | 用ACC的内容驱动总线 | | load\_ACC | 将总线上的数据载入ACC | | PC\_bus | 用PC的内容驱动总线 | | load\_IR | 将总线上的数据装载至IR | | load\_MAR | 将总线上的数据装载至MAR | | MDR\_bus | 用MDR的内容驱动总线 | | load\_MDR | 将总线上的数据装载至MDR | | ALU\_ACC | 用ALU的结果装载ACC | | INC\_PC | PC+1并将结果存至PC中 | | Addr\_bus | 用IR中指令的操作码部分驱动总线 | | CS | 片选。用MAR的内容设置存储器地址 | | R\_NW | 读取，不可写。当R\_NW无效且CS有效时，MDR的内容存储于存储器中 | | ALU\_add | 在ALU中执行加法操作 | | ALU\_sub | 在ALU中执行减法操作 |  1. **控制存储器中存储的微指令格式，指出操作控制字段采用什么格式？多少位？下址字段多少位？**   **控制器：微程序控制器**  **微指令编码：水平型微指令**  **采用长格式，一条微指令能控制数据通路中多个功能部件并行操作。**  **共20位，下地址5位**   1. **如何形成后继微地址？微码中有两处分支，一是根据OP多路分支，二是根据Z\_flag二路分支，微码设计没有采用P1和P2等测试位的方法，分析代码，回答该微码是如何实现分支的？**   **直接由微指令的下地址字段指出。微指令格式中设置一个下地址字段，由微指令的下地址字段直接指出后继微指令的地址**  **根据微地址取出微代码，生成下地址如果下址是01111则根据op得到后继地址，如果下址是10000则根据Z\_flag得到后继地址，如Z\_flag为“1”，则后继地址为01110，如Z\_flag为“0”，则后继地址为01101。**  **8． 解释某一条指令的执行过程，地址转移逻辑**  **（为避免抄袭，需解释第n条指令，n为学号除4的余数）**  **学号21，余数为1**  **第一条指令：add5：从mem(5)取数到总线上，与ACC中的数相加**  **9． 解释某一条微指令的组成。10000110011100001**  **（为避免抄袭，需解释第n条指令，n为学号除15的余数）**  **学号21，余数为6**    **第六条：000000101000010 00000**  **高15位表示微命令，低5位表示地址转移逻辑**  **后5位下一条微指令地址为00000，即回到第0号指令**  **第六位ALU\_add为1，在ALU中执行加法操作**  **第十一位ALU\_ACC为1用ALU的结果装载ACC**  **第十三位MDR\_bus为1，用MDR的内容驱动总线**    **10． 执行完bne指令后程序转到什么地方执行？**  **bne执行完毕后，由于Z\_flag不为0，跳转至内存7地址所指的地址执行指令，内存地址7的内容为0，所以跳转至第0条指令load4重复执行。**  **仿真结果与分析：**  **C:\Users\Microsoft\AppData\Roaming\DingTalk\2146097823_v2\ImageFiles\65\lQLPJxZYzLT-523NAbzNBJ6wu5P5qjhPCigCko4-D0DTAA_1182_444.png**  **add\_r\_out：微地址**  **data\_r\_out：微指令内容**  **outpc：程序里的count利用outpc跟踪的，执行01234地址后，然后执行0地址是因为bne执行完毕后，由于Z\_flag不为0，跳转至内存7地址所指的地址执行指令，内存地址7的内容为0，所以跳转至第0条指令load4重复执行。**  **outir：跟outpc有关系，是从outpc读过来的指令。**  **op\_out：这是指令outir对应的操作码前三位。**  **outmar：指令outir读过来进行拆分。**  **outmdr：有的时候是指令，有的时候是读出来的数据。**  **outalu：alu的输出，相当于acc的内容**  **outbus：总线上出现的内容，它会不停的变化**  **mem\_addr：看哪个地址都可以**  **程序的执行结果：6号地址单元的内容变为5**  **最开始系统resset为0，add\_r也全部为0，在下一个时钟上升沿，根据add\_r的0号地址读出0号地址里的微指令，在读取完3号微指令以后跳转至8号，接着跳转至4，最后又跳转至0号地址进行循环。**  **心得体会：**  **通过本次实验，理解微程序控制器的控制原理，进一步掌握指令流程和功能，了解掌握微程序控制器的设计思路和方法。** | |