**实验报告**

**学院（系）名称：**计算机与通信工程学院

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **姓名** | 范得良 | **学号** | 20145534 | **专业** | 计算机科学与技术 |
| **班级** | 2014级4班 | **实验项目** | 实验四：基本模型机设计与实现 | | |
| **课程名称** | | 计算机组成与结构 | | **课程代码** | 0669026 |
| **实验时间** | | 2016年3月24日 13：00-16：00 | | **实验地点** | 7-109 |
| **批改意见** | | | | **成绩** |  |
| **教师签字：** | | | | | |
| 一、实验目的  1．掌握一个简单CPU 的组成原理。  2．在掌握部件单元电路的基础上，进一步将其构造一台基本模型计算机。  3. 为其定义五条机器指令，编写相应的微程序，并上机调试掌握整机概念。  二、实验要求  按照实验步骤完成实验项目，实现简单模型机控制单元微程序的模拟设计。  三、实验原理  本实验要实现一个简单的CPU，并且在此CPU的基础上，继续构建一个简单的模型计算机。  CPU由运算器（ALU）、微程序控制器（MC）、通用寄存器（R0），指令寄存器（IR）、程序计数器（PC）和地址寄存器（AR）组成，如图1所示。这个CPU 在写入相应的微指令后，就具备了执行机器指令的功能，但是机器指令一般存放在主存当中，CPU必须和主存挂接后，才有实际的意义，所以还需要在该CPU的基础上增加一个主存和基本的输入输出部件，以构成一个简单的模型计算机。    图1 基本CPU构成原理图    图2 微程序控制器组成原理框图  微程序控制器的基本任务是完成当前指令的翻译和执行，即将当前指令的功能转换成可以控制的硬件逻辑部件工作的微命令序列，完成数据传送和各种处理操作。它的执行方法就是将控制各部件动作的微命令的集合进行编码，即将微命令的集合仿照机器指令一样，用数字代码的形式表示，这种表示称为微指令。这样就可以用一个微指令序列表示一条机器指令，这种微指令序列称为微程序。微程序存储在一种专用的存储器中，称为控制存储器,微程序控制器原理框图如图2所示。  控制器是严格按照系统时序来工作的，因而时序控制对于控制器的设计是非常重要的，从前面的实验可以很清楚地了解时序电路的工作原理，本实验所用的时序由时序单元来提供，分为四拍TS1、TS2、TS3、TS4（时序单元的介绍见第三章3.3 系统实验单元电路中6．时序与操作台单元部分）。  微程序控制器的组成见见第三章3.3 系统实验单元电路中4．微程序控制器电路单元（MC 单元）中图3-9，其中控制存储器采用3片2816的E2PROM，具有掉电保护功能，微命令寄存器18 位，用两片8D 触发器（273）和一片4D触发器（175）组成。微地址寄存器6位，用三片正沿触发的双D触发器（74）组成，它们带有清“0”端和预置端。在不判别测试的情况下，T2 时刻打入微地址寄存器的内容即为下一条微指令地址。当T4 时刻进行测试判别时，转移逻辑满足条件后输出的负脉冲通过强置端将某一触发器置为“1”状态，完成地址修改。  在实验平台中设有一组编程控制开关KK3、KK4、KK5（位于时序与操作台单元），可实现对存储器（包括存储器和控制存储器）的三种操作：编程、校验、运行。（详细介绍见第三章3.3 系统实验单元电路中6．时序与操作台单元部分）。  位于实验平台 MC 单元左上角一列三个指示灯 MC2、MC1、MC0 用来指示当前操作的微程序  字段，分别对应 M23——M16、M15——M8、M7——M0。实验平台提供了比较灵活的手动操作方式，比如在上述操作中在对地址置数后将开关 KK4 拨至‘减 1’档，则每次随着开关 ST 的两次拨动操作，字节数依次从高 8 位到低 8 位递减，减至低 8 位后，再按动两次开关ST，微地址会自动减一，继续对下一个单元的操作。  微指令字长共24位，控制位顺序如表1所示。  表1 微指令格式  其中 MA5…MA0 为 6 位的后续微地址，A、B、C 为三个译码字段，分别由三个控制位译码出多位。C 字段中的 P<1>为测试字位。其功能是根据机器指令及相应微代码进行译码，使微程序转入相应的微地址入口，从而实现完成对指令的识别，并实现微程序的分支，本系统上的指令译码原理如图3所示，图中I[7]…I[2]为指令寄存器的第 7…2 位输出，SE5…SE0 为微控器单元微地址锁存器的强置端输出，指令译码逻辑在 IR 单元的 INS\_DEC（GAL20V8）中实现。  从第三章3.3 系统实验单元电路中4．微程序控制器电路单元（MC 单元）中图3-9中也可以看出，微控器产生的控制信号比表1中的要多，这是因为实验的不同，所需的控制信号也不一样，本实验只用了部分的控制信号。  本实验除了用到指令寄存器（IR）、程序计数器（PC）和通用寄存器 R0 外，还要用到 IN和 OUT 单元，从微控器出来的信号中只有 IOM、WR 和 RD 三个信号，所以对这两个单元的读写信号还应先经过译码，其译码原理如图4所示。IR 单元的原理图如图5所示，R0单元原理如图 6所示，IN 单元的原理图见第三章3.3 系统实验单元电路中7．输入设备单元（IN 单元）中图3-15所示，OUT 单元的原理图见第三章3.3 系统实验单元电路中8．输出设备单元（OUT 单元）中图3-16所示。    图3 指令译码原理图    图4 读写控制逻辑 图5 IR单元原理图  系统的程序计数器（PC）和地址寄存器（AR）集成在一片CPLD 芯片中 。CLR 连接至CON 单元的总清端CLR，按下CLR 按钮，将使PC清零，LDPC和T3 相与后作为计数器的计数时钟，当LOAD为低时，计数时钟到来后将CPU内总线上的数据打入PC，如图7所示。  本模型机共安排了五条机器指令：IN（输入）、ADD（二进制加法）、OUT（输出）、JMP（无条件转移），HLT（停机），其指令格式如下（高４位为操作码）：  助记符 机器指令码 说明  IN 0010 0000 IN → R0  ADD 0000 0000 R0 + R0 → R0  OUT 0011 0000 R0 → OUT  JMP addr 1110 0000 \*\*\*\*\*\*\*\* addr → PC  HLT 0101 0000 停机    图6 R0原理图 图7 程序计数器(PC)原理图  其中JMP 为双字节指令，其余均为单字节指令，\*\*\*\*\*\*\*\*为addr 对应的二进制地址码。要求CPU自动从存储器读取指令并执行。根据以上要求，设计数据通路图，如图8所示。  系统涉及到的微程序流程见图9所示，当拟定“取指”微指令时，该微指令的判别测试字段为 P<1>测试。指令译码原理见图3所示，由于“取指”微指令是所有微程序都使用的公用微指令，因此 P<1> 的测试结果出现多路分支。本机用指令寄存器的高6位（IR7—IR2）作为测试条件，出现５路分支，占用５个固定微地址单元，剩下的其它地方就可以一条微指令占用控存一个微地址单元随意填写，微程序流程图上的单元地址为16进制。    图8 数据通路图    图9 简单模型机微程序流程图  当全部微程序设计完毕后，应将每条微指令代码化，填入表2中，表2即为将图9的微程序流程图按微指令格式转化而成的“二进制微代码表”。  表2 二进制微代码表  地址 十六进制 高五位 S3-S0 A字段 B字段 C字段 MA5-MA0  00 00 00 01 00000 0000 000 000 000 000001  01 00 6D 43 00000 0000 110 110 101 000011  03 10 70 70 00010 0000 111 000 001 110000  04  05  1D  30  32  33 28 04 01 00101 0000 000 010 000 000001  35 00 00 35 00000 0000 000 000 000 110101  3C  设计一段机器程序，要求从 IN单元读入一个数据，存于 R0，将 R0 和自身相加，结果存于R0，再将 R0 的值送 OUT 单元显示。 根据要求可以得到如下程序，地址和内容均为二进制数。  地 址 内 容 助记符 说 明  00000000 00100000 ; START: IN R0 从 IN 单元读入数据送 R0  00000001 00000000 ; ADD R0,R0 R0 和自身相加，结果送 R0  00000010 00110000 ; OUT R0 R0 的值送 OUT 单元显示  00000011 11100000 ; JMP START 跳转至 00H 地址  00000100 00000000 ;  00000101 01010000 ; HLT 停机    四、实验步骤  1.按图10连接实验线路。  2. 写入实验程序，并进行校验，分两种方式，手动写入和联机写入。  1) 手动写入和校验  (1) 手动写入微程序  ① 将时序与操作台单元的开关 KK1 置为‘停止’档，KK3 置为‘编程’档，KK4 置为‘控存’档，KK5 置为‘置数’档。  ② 使用 CON单元的 SD05——SD00给出微地址，IN单元给出低 8 位应写入的数据，连续两次按动时序与操作台的开关 ST，将 IN单元的数据写到该单元的低 8位。  ③ 将时序与操作台单元的开关 KK5 置为‘加 1’档。  ④ IN单元给出中 8 位应写入的数据，连续两次按动时序与操作台的开关 ST，将 IN单元的数据写到该单元的中 8位。IN单元给出高 8 位应写入的数据，连续两次按动时序与操作台的开关 ST，将 IN单元的数据写到该单元的高 8 位。  ⑤ 重复①、②、③、④四步，将表2的微代码写入 2816芯片中。  (2) 手动校验微程序  ① 将时序与操作台单元的开关 KK1 置为‘停止’档，KK3 置为‘校验’档，KK4 置为‘控存’档，KK5 置为‘置数’档。  ② 使用 CON单元的 SD05——SD00给出微地址，连续两次按动时序与操作台的开关 ST，MC 单元的指数据指示灯 M7——M0 显示该单元的低 8 位。  ③ 将时序与操作台单元的开关 KK5 置为‘加 1’档。  ④ 连续两次按动时序与操作台的开关 ST，MC 单元的指数据指示灯 M15——M8 显示该单元的中 8 位，MC 单元的指数据指示灯 M23——M16 显示该单元的高 8位。  ⑤ 重复①、②、③、④四步，完成对微代码的校验。如果校验出微代码写入错误，重新写入、校验，直至确认微指令的输入无误为止。    图10 实验接线图  (3) 手动写入机器程序  ① 将时序与操作台单元的开关 KK1 置为‘停止’档，KK3 置为‘编程’档，KK4 置为‘主存’档，KK5 置为‘置数’档。  ② 使用 CON单元的 SD07——SD00给出地址，IN单元给出该单元应写入的数据，连续两次按动时序与操作台的开关 ST，将 IN单元的数据写到该存储器单元。  ③ 将时序与操作台单元的开关 KK5 置为‘加 1’档。  ④ IN单元给出下一地址（地址自动加 1）应写入的数据，连续两次按动时序与操作台的开关 ST，将 IN 单元的数据写到该单元中。然后地址会又自加 1，只需在 IN 单元输入后续地址的数据，连续两次按动时序与操作台的开关 ST，即可完成对该单元的写入。  ⑤ 亦可重复①、②两步，将所有机器指令写入主存芯片中。  (4) 手动校验机器程序  ①将时序与操作台单元的开关 KK1 置为‘停止’档，KK3 置为‘校验’档，KK4 置为‘主存’档，KK5 置为‘置数’档。  ② 使用CON单元的SD07——SD00给出地址，连续两次按动时序与操作台的开关ST，CPU内总线的指数据指示灯 D7——D0 显示该单元的数据。  ③ 将时序与操作台单元的开关 KK5 置为‘加 1’档。  ④ 连续两次按动时序与操作台的开关 ST，地址自动加 1，CPU内总线的指数据指示灯 D7——D0 显示该单元的数据。此后每两次按动时序与操作台的开关 ST，地址自动加 1，CPU 内总线的指数据指示灯 D7——D0 显示该单元的数据，继续进行该操作，直至完成校验，如发现错误，则返回写入，然后校验，直至确认输入的所有指令准确无误。  ⑤ 亦可重复①、②两步，完成对指令码的校验。如果校验出指令码写入错误，重新写入、校验，直至确认指令码的输入无误为止。  2) 联机写入和校验  联机软件提供了微程序和机器程序下载功能，以代替手动读写微程序和机器程序，但是微程序和机器程序得以指定的格式写入到以TXT为后缀的文件中，微程序和机器程序的格式如下：    本次实验程序如下，程序中分号‘；’为注释符，分号后面的内容在下载时将被忽略掉：  ; //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* //  ; // //  ; // CPU 与简单模型机实验指令文件 //  ; // //  ; //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* //  ; //\*\*\*\*\*\* Start Of Main Memory Data \*\*\*\*\*\* //  $P 00 20 ; START: IN R0 从 IN 单元读入数据送 R0  $P 01 00 ; ADD R0,R0 R0 和自身相加，结果送 R0  $P 02 30 ; OUT R0 R0 的值送 OUT 单元显示  $P 03 E0 ; JMP START 跳转至 00H 地址  $P 04 00 ;  $P 05 50 ; HLT 停机  ; //\*\*\*\*\*\*\* End Of Main Memory Data \*\*\*\*\*\*\* //    ; //\*\*\*\* Start Of MicroController Data \*\*\*\* //  $M 00 000001 ; NOP  $M 01 006D43 ; PC->AR,PC 加 1  $M 03 107070 ; MEM->IR, P<1>  $M 04 \_\_\_\_\_\_ ; R0->B  $M 05 \_\_\_\_\_\_ ; A 加 B->R0  $M 1D \_\_\_\_\_\_ ; MEM->PC  $M 30 \_\_\_\_\_\_ ; R0->A  $M 32 \_\_\_\_\_\_ ; IN->R0  $M 33 280401 ; R0->OUT  $M 35 000035 ; NOP  $M 3C \_\_\_\_\_\_ ; PC->AR,PC 加 1  ;//\*\* End Of MicroController Data \*\*//    选择联机软件的“【转储】—【装载】”功能，在打开文件对话框中选择上面所保存的文件，软件自动将机器程序和微程序写入指定单元。 选择联机软件的“【转储】—【刷新指令区】”可以读出下位机所有的机器指令和微指令，并在指令区显示，对照文件检查微程序和机器程序是否正确，如果不正确，则说明写入操作失败，应重新写入，可以通过联机软件单独修改某个单元的指令，以修改微指令为例，先用鼠标左键单击指令区的‘微存’TAB 按钮，然后再单击需修改单元的数据，此时该单元变为编辑框，输入 6 位数据并回车，编辑框消失，并以红色显示写入的数据。  3. 运行程序  方法一：本机运行  将时序与操作台单元的开关 KK1、KK3 置为‘运行’档，按动 CON单元的总清按钮 CLR，  将使程序计数器 PC、地址寄存器 AR 和微程序地址为 00H，程序可以从头开始运行，暂存器 A、B，指令寄存器 IR 和 OUT 单元也会被清零。  将时序与操作台单元的开关 KK2 置为‘单步’档，每按动一次 ST 按钮，即可单步运行一条微指令，对照微程序流程图，观察微地址显示灯是否和流程一致。每运行完一条微指令，观测一次 CPU内总线和地址总线，对照数据通路图，分析总线上的数据是否正确。  当模型机执行完 JMP 指令后，检查 OUT 单元显示的数是否为 IN单元值的 2 倍，按下CON  单元的总清按钮 CLR，改变 IN单元的值，再次执行机器程序，从 OUT 单元显示的数判别程序执行是否正确。    方法二：联机运行  将时序与操作台单元的开关 KK1 和 KK3 置为‘运行’档，进入软件界面，选择菜单命令“【实验】—【简单模型机】”，打开简单模型机数据通路图。  按动 CON单元的总清按钮 CLR，然后通过软件运行程序，选择相应的功能命令，即可联机  运行、监控、调试程序，当模型机执行完 JMP指令后，检查 OUT 单元显示的数是否为 IN单元值的 2 倍。在数据通路图和微程序流中观测指令的执行过程，并观测软件中地址总线、数据总线以及微指令显示和下位机是否一致。    实验结果截图：  【心得体会】  1.通过本次实验我掌握了一个简单的CPU 的组成原理：  CPU由运算器（ALU）、微程序控制器（MC）、通用寄存器（R0），指令寄存器（IR）、程序计数器（PC）和地址寄存器（AR）组成。  2.通过本次实验，除了掌握了部件单元电路，而且还利用他们构造了一台基本模型计算机，提高了动手能力，进一步理解了计算机的组成。  3.学会了上机编写和调试微程序，进一步了解了微程序：  微程序控制器的基本任务是完成当前指令的翻译和执行，即将当前指令的功能转换成可以控制的硬件逻辑部件工作的微命令序列，完成数据传送和各种处理操作。它的执行方法就是将控制各部件动作的微命令的集合进行编码，即将微命令的集合仿照机器指令一样，用数字代码的形式表示，这种表示称为微指令。这样就可以用一个微指令序列表示一条机器指令，这种微指令序列称为微程序。微程序存储在一种专用的存储器中，称为控制存储器。 | | | | | |