****

课程设计报告书

**计算机组成与体系结构CPU 2OP#1指令设计**

**学 院 计算机科学与工程学院**

**专 业 　计算机科学技术**

**学生姓名 陈伟航　朱思晓　赵鸿燕　张炜媚**

**学生学号 201230600095**

**201230471046**

**201230471046**

**201230471046**

**指导教师 赖晓铮**

**课程编号 145243**

**课程学分 2**

**起始日期 2014,9,1 - 2014,10,21**

|  |  |
| --- | --- |
| 教  师  评  语 | 教师签名：  日期： |
| 成  绩  评  定 |  |
| 备  注 |  |

**计算机组成与体系结构CPU 2OP#1指令设计**

一、选题背景

主要问题：在已有的FS-I-SYSTEM硬件仿真系统中完成微指令的编写。

应达到的技术要求：在已有的FS-I-SYSTEM硬件仿真系统中通过指令调用让微指令完成一定的操作。

设计的指导思想：Harvard architecture、Von Neumann architecture。FS-I-SYSTEM的CPU类似于早期的哈佛架构，其程序存放在E2ROM里，数据存放在位于CPU核心单元的RAM中，这种指令与数据分开存放的架构正是哈佛架构的显著特征。FS-I-SYSTEM的是16位指令的CPU，很多机器指令需要若干条微指令完成，具有RISC的特征。

二、方案论证(设计理念)

1.设计理念：FS-I-SYSTEM的CPU类似于早期的哈佛架构，其程序存放在E2ROM里，数据存放在位于CPU核心单元的RAM中，这种指令与数据分开存放的架构正是哈佛架构的显著特征，而这与冯·诺依曼体系结构不同。FS-I-SYSTEM的是16位指令的CPU，很多机器指令需要若干条微指令完成，具有RISC的特征。

2.小组完成2OP#1指令，汇编代码为OP R1 R2，作用是将R1和R2进行OP运算（OP可以是ADD、SUB、XOR等），将结果存回R1。在具体的运行过程中，R1和R2中的数据分别在两条指令中被加载到ALU的两个操作数上，R1与R2由机器码提供的参数指定。ALU进行运算的种类，也由机器指令中特定位置的参数提供。ALU经过运算后，将结果写回R1，寄存器输入的过程与原理与上述输出过程类似。

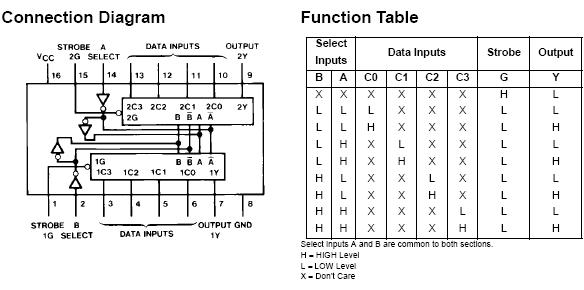
3.指令涉及到的CPU内核部件作用及接口说明：

**ALU部分：**

**74LS153**

使用的器件：输入输出缓冲器

管脚图与真值表：

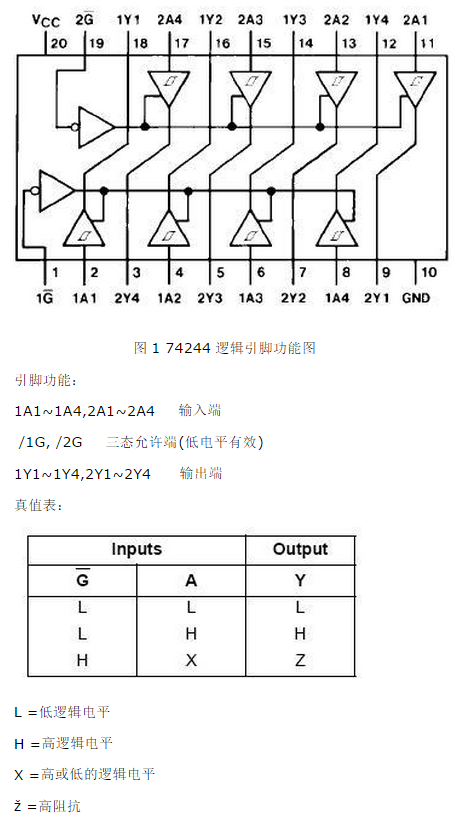


功能：74153里面有两个地址码共用的4选1数据选择器。通过输入不同的地址码A1，A0，可以控制输出Y选择4个输入数据D0～D3中的一个

**74LS244**

使用的器件：ALU\_SELECTOR\_1，ALU\_SELECTOR\_0

管脚图与真值表：

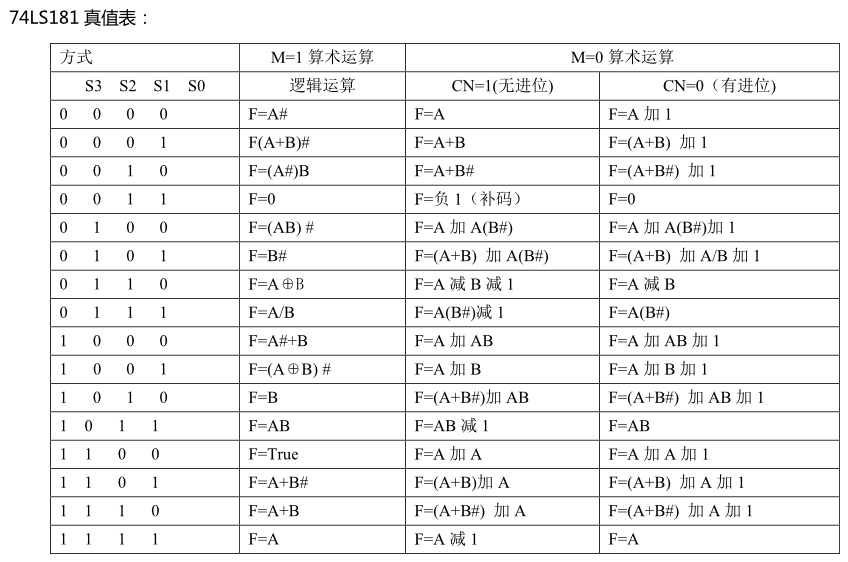


功能：

74ls244是ttl 八同相三态缓冲器/线驱动器，其coms器件对应为74hc244,常用在单片机mcu系统中，作为单片机的输入输出数据缓冲器，在选通时输入数据送到总线上，在非选通时对总线呈高阻态。

**74LS181**

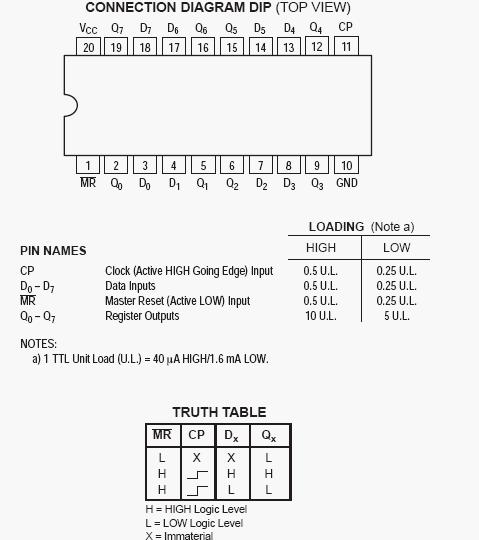
使用的器件：ALU\_0~3



功能：4位运算器

**74LS273**

功能：8位三态D锁存器（时序锁存）



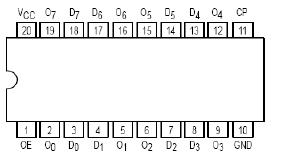
**Register部分**

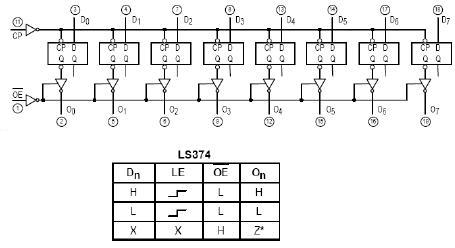
**74LS374**

使用的器件：AH,AL,BH,BL,DH,DL

功能：八上升沿 D 触发器(3S,时钟输入有回环特性)

引脚图与真值表：

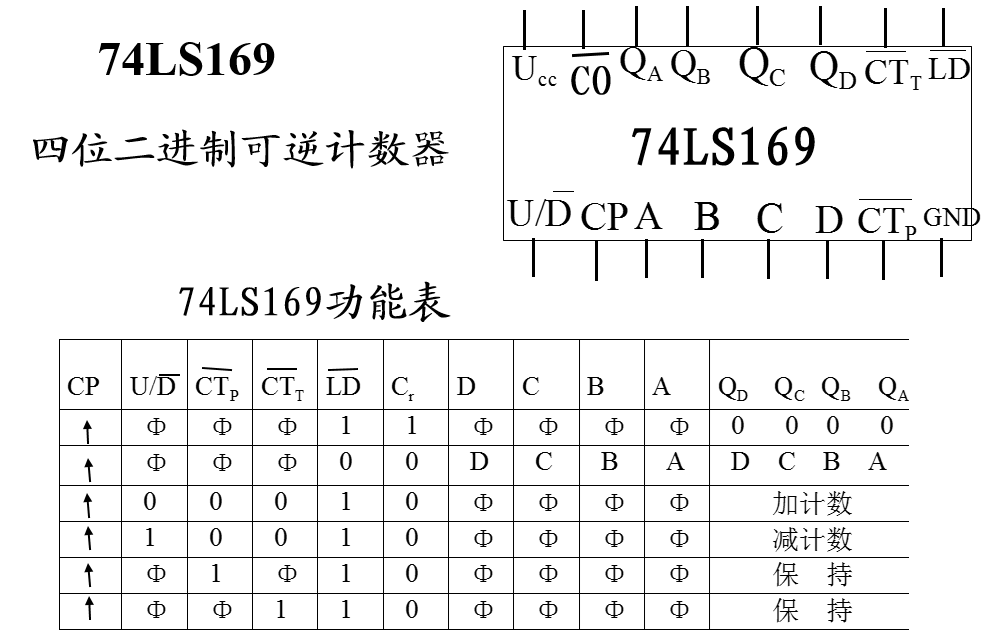




**74LS169**

使用的器件：CH\_H,CH\_L,CL\_H,CL\_L

功能、引脚图：



4.完成步骤及分工：

1）分析需求（2OP#1指令格式与机器码定义）。（陈伟航、朱思晓）

2）编写测试程序，实现将两个寄存器的值进行运算，再存回寄存器。（陈伟航、朱思晓）

3）分析指令数据通路，画出微程序方框图，写出周期表。（张炜媚、赵鸿燕）

4）编写微指令，填写微程序表。（朱思晓、陈伟航）

5）编译测试程序、微程序。（张炜媚、赵鸿燕）

6）导入硬件仿真平台、调试。（张炜媚、赵鸿燕）

三、过程论述

**1.完成的指令格式、机器码定义。**

2OP #1

OP码：0010100

适用指令：

ADD RX, RX

SUB RX, RX

AND RX, RX

OR RX, RX

XOR RX, RX

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | X | X | X | F0 | E0 | X | X | X | X |
| X | X | X | X | X | X | X | X | X | X | F1 | E1 | 0 | 0 | B1 | A1 |
| X | X | X | X | X | X | X | X | X | X | F2 | E2 | D2 | C2 | B2 | A2 |

ADD RL, RL

SUB RL, RL

AND RL, RL

OR RL, RL

XOR RL, RL

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | X | X | X | F0 | E0 | X | X | X | X |
| X | X | X | X | X | X | X | X | X | X | F1 | E1 | 1 | 0 | B1 | A1 |
| X | X | X | X | X | X | X | X | X | X | F2 | E2 | D2 | C2 | B2 | A2 |

ADD RH, RH

SUB RH, RH

AND RH, RH

OR RH, RH

XOR RH, RH

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | X | X | X | F0 | E0 | X | X | X | X |
| X | X | X | X | X | X | X | X | X | X | F1 | E1 | 0 | 1 | B1 | A1 |
| X | X | X | X | X | X | X | X | X | X | F2 | E2 | D2 | C2 | B2 | A2 |

E0： REGISTERS\_OE\_0

F0： REGISTERS\_OE\_1

A1：REGISTERS\_LDRX\_0

B1：REGISTERS\_LDRX\_1

E1： REGISTERS\_OE\_0

F1： REGISTERS\_OE\_1

A2：ALU\_CI

B2：ALU\_S0

C2：ALU\_S1

D2：ALU\_S2

E2：ALU\_S3

F2： ALU\_M

X： 无效位

备注：

1）

F0E0位译码（默认00）：

00：AX/AL/AH 01：BX/BL/BH 10：CX/CL/CH 11：DX/DL/DH

B1A1位译码（默认00）：

00：AX/AL/AH 01：BX/BL/BH 10：CX/CL/CH 11：DX/DL/DH

F1E1位译码（默认00）：

00：AX/AL/AH 01：BX/BL/BH 10：CX/CL/CH 11：DX/DL/DH

F2E2D2C2B2A2位译码：

010011： ADD

001100： SUB

11011X： AND

11110X： OR

10110X： XOR

2）

F0E0位要与B1A1位相同。

**2.测试程序**

;格式：

;(地址) (机器语言高8位) (机器语言低8位)

;示例程序

;验证 2OP #5 指令

;代码段部分

CODE

;MOV CL, 0x0001

0000 00011100 00001010

0001 00000000 00000001

;MOV BL, 0x0011

0002 00011100 00001001

0003 00000000 00000011

;ADD CL, BL

0004 00101000 00100000

0005 00000000 00011010

0006 00000000 00010011

;HALT

0007 01101010 00000000

测试程序功能：

将立即数0x0001和0x0003分别用MOV14指令存到RC和RB中，2OP#5指令将RB的值加到RC，RC的值在程序结束后应该为0x004。

RC和RB中的数据分别在两条指令中被加载到ALU的两个操作数上，寄存器的选择由机器码提供的参数实现（具体实现是：根据机器码的参数，IR使能register区内被选择寄存器的OE(第8位模式)端，相应的寄存器上的数据输出到register区bus(T1时刻)，当微程序控制器使能register区总的OE端（T3时刻）时，register区bus上的数据输出至CPU总的bus）。

ALU进行运算的种类，也由机器指令中特定位置的参数提供，并从program区直接通过总线bus送至ALU控制端。

ALU经过运算后，将结果写回RC，寄存器输入的过程与原理与上述输出过程类似。

**3.2OP#1数据通路**

1）第一操作数传到总线，将总线上的第一操作数锁存到ALU第一操作数寄存器DRA中

2）指令寄存器锁存第二参数指令

3）第二操作数传到总线，将总线上的第二操作数锁存到ALU第二操作数寄存器DRB中

4）PROGRAM数据输出到总线，ALU控制寄存器锁存总线数据

5）ALU结果输出到总线，结果锁存到寄存器中标志寄存器锁存标志位

**4.微程序方框图**



**5.指令周期表**

周期一：

|  |  |
| --- | --- |
| T1 | 锁存微指令(10010100),第一操作数传到总线 |
| T2 | 锁存下地址(00001001) |
| T3 | 第一操作数锁存到ALU第一操作数寄存器DRA中 |
| T4 | —— |

周期二：

|  |  |
| --- | --- |
| T1 | 锁存微指令(00001001) |
| T2 | 锁存下地址(00001010) |
| T3 | IR锁存第二参数指令 |
| T4 | PC+1 |

周期三：

|  |  |
| --- | --- |
| T1 | 锁存微指令(00001010) ,第二操作数传到总线 |
| T2 | 锁存下地址(00001011) |
| T3 | 第二操作数锁存到ALU第二操作数寄存器DRB中 |
| T4 | —— |

周期四：

|  |  |
| --- | --- |
| T1 | 锁存微指令(00001011),PROGRAM数据输出到总线 |
| T2 | 锁存下地址(00001100) |
| T3 | ALU控制寄存器锁存总线数据 |
| T4 | PC+1 |

周期五：

|  |  |
| --- | --- |
| T1 | 锁存微指令(00001100),ALU结果输出到总线 |
| T2 | 锁存下地址(00000000) |
| T3 | 寄存器锁存结果，标志寄存器锁存标志位 |
| T4 | P2中断标志检查 |

**6.微指令代码**

(148 10010100)

10000000 00000000 00000000 00010100 00000000 00001001

; (00000000) 2OP #1: REGISTERS->CPU\_DB,ALU\_LD\_DRA

(9 00001001)

00000000 01010000 00000000 00000000 00000000 00001010

;(10010100)2OP #1: PROGRAM->IR, PC+1

(10 00001010)

01000000 00000000 00000000 00010100 00000000 00001011

;(10001001)2OP #1: REGISTERS->CPU\_DB,ALU\_LD\_DRB

(11 00001011)

00100000 01000000 00000000 00100100 00000000 00001100

;(10001010)2OP #1: PROGRAM->CPU\_DB,ALU\_LD\_CTL,PC+1

(12 00001100)

00001000 00000000 00000100 00011101 00000000 00000000

;(10001011)2OP#1: ALU->CPU\_DB,REGISTERS\_LD\_EN,FLAGS\_LD\_EN,P2

**7.编译程序、微程序、导入到硬件仿真平台**

写好微程序和测试程序后，分别用各自的编译器编译（实际过程是对ASCII的译码），在Proteus8中将微程序编译结果导入微程序储存器中，将测试程序编译结果导入PROGRAM区中，点击开始仿真，点击CPU控制区的RESET按钮。此时，如果要自动运行，点击AUTOSTART，如果要手动调试，点击MANUAL\_MODE，然后点击MANUAL\_PAUSE来产生时钟脉冲。

四、结果分析

在完成2OP#1的过程中，在各个环节均未发现异常。

在第一条MOV语句结束后，RC呈现0x0001，在执行第二条MOV语句的过程中，也可以看到0x0003被存入RB，ALU计算结束，并将结果写回RC后，RC低八位如预期地呈现结果00000100。

五、课程设计总结

**个人课程设计总结**

陈伟航：纸上得来终觉浅，绝知此事要躬行。大二下学期学了计算机组成原理之后，对计算机组成有一些理解，但很多细节的地方仍然不够明朗。刚接触这次课程设计的内容的时候，仅有上学期实验课剩下的一点印象，为了写好一条看似简单的2OP指令，翻课本、查资料，问老师、问助教，在各种的电路里徘徊，柳暗花明，总算对一个CPU的运行有了更加具体的认识，相信这对以后写程序也会有许多增益。

朱思晓：

此次计组课设让我收获颇丰，从一开始担心无法完成课设而跑去向师兄整晚请教，到略通门路，帮助其他组的同学解决困难，再到最为另一个相对简单的CPU仿真平台添加了中断系统，并在赖老师的指导下设计了其终端处理机制，我对计算机底层尤其是CPU的实现也算有了一些浅薄的认识。

赵鸿燕：

张炜媚：

**发现的电路问题（朱思晓）**

我在帮助2OP#6组的同学进行微程序调试时，发现了一个仿真平台的BUG：由于ALU内部译码分配的不合理，ALU输出结果可能不会被正确地初始化（应该是0x0000，实际为0x0F0F），如果ALU计算过程不涉及高8位，高8位的错误输出将不会被刷新，而直接输出。这个输出结果如果只有低8位被存至RC，错误将不会在RC处显现（0x0004），但如果全16位都存入RC，将得到错误的运算结果（0x0F04）。

经过研究后，师兄提出了一个机智的解决方法：用每次开始仿真时必做的RESET动作模拟上电时的初始化信号，对ALU进行一次额外的初始化。这个修复方案实际运行效果良好，于是有了修复后的版本。

参考文献

1. 微型计算机原理与接口技术(第4版)