### 计算机系统结构实验二

### 指令译码器实验

### 一、实验目的

1学习了解指令系统的概念

2.理解指令译码器的作用和重要性。

3.学习设计指令译码器。

### 二、实验原理

指令系统

为了指令的规整性和便于译码，主要采用定长操作码方案，操作码长度只有8位和16位两种，绝大部分指令为8位操作码。寻址方式包括了寄存器寻址、立即数寻址、直接地址和相对寻址。

根据指令的执行步骤不同，可以把全部指令分为A、B两组。其中A组指令完成的是通用寄存器之间的数据运算或传送，或其他一些特殊操作，在取指之后可一步完成，包括：ADD，SUB，AND，CMP，XOR，TEST，OR，MVRR，DEC，INC，SHL，SHR，ADC，SBB，JR，JRC，JRNC，JRZ，JRNZ，JRS，JRNS，CLC，STC；B组指令完成的是一次内存读写操作，在取指之后可两步完成，包括：JMPA，LDRR，STRR，MVRD。在指令编码时以指令操作码的最高位来区分 A、B组指令，0为A组指令，1为B组指令。见表1。

表1 指令格式表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指令格式 | 汇编语句 | CZVS | 指令类型 | 功能说明 |
| 00000000DRSR | ADD DR,SR | \*\*\*\* | A组指令 | DR←DR+SR |
| 00000001DRSR | SUB DR,SR | \*\*\*\* | A组指令 | DR←DR-SR |
| 00000010DRSR | AND DR,SR | \*\*\*\* | A组指令 | DR←DR and SR |
| 00000011DRSR | CMP DR,SR | \*\*\*\* | A组指令 | DR-SR |
| 00000100DRSR | XOR DR,SR | \*\*\*\* | A组指令 | DR←DR xor SR |
| 00000101DRSR | TEST DR,SR | \*\*\*\* | A组指令 | DR and SR |
| 00000110DRSR | OR DR,SR | \*\*\*\* | A组指令 | DR←DR or SR |
| 00000111DRSR | MVRR | …. | A组指令 | DR←SR |
| 00001000DR0000 | DEC DR | \*\*\*\* | A组指令 | DR←DR-1 |
| 00001001DR0000 | INC DR | \*\*\*\* | A组指令 | DR←DR+1 |
| 00001010DR0000 | SHL DR | \*… | A组指令 | DR,C←DR\*2 |
| 00001011DR0000 | SHR DR | \*… | A组指令 | DR,C←DR/2 |
| 00001100DRSR | ADC DR,SR | \*\*\*\* | A组指令 | DR←DR+SR+C |
| 00001101DRSR | SBB DR,SR | \*\*\*\* | A组指令 | DR←DR-SR-C |
| 01000000OFFSET | JR ADR | …. | A组指令 | 无条件跳转 |
| 01000100OFFSET | JRC ADR | …. | A组指令 | C=1时跳转 |
| 01000101OFFSET | JRNC ADR | …. | A组指令 | C=0时跳转 |
| 01000110OFFSET | JRZ ADR | …. | A组指令 | Z=1时跳转 |
| 01000111OFFSET | JRNZ ADR | …. | A组指令 | Z=0时跳转 |
| 01000001OFFSET | JRS ADR | …. | A组指令 | S=1时跳转 |
| 01000011OFFSET | JRNS ADR | …. | A组指令 | S=0时跳转 |
| 011110000000000 | CLC | 0… | A组指令 | C=0 |
| 011110100000000 | STC | 1… | A组指令 | C=1 |
| 1000000000000000ADR | JMPA ADR | …. | B组指令 | 无条件跳转到ADR |
| 10000010DRSR | LDRR DR,[SR] | …. | B组指令 | DR←[SR] |
| 10000011DRSR | STRR [DR],SR | …. | B组指令 | [DR]←SR |
| 10000001DR0000 | MVRD DR,DATA | …. | B组指令 | DR←DATA |

指令译码器是计算机控制器中最重要的部分。所谓组合逻辑控制器就是指指令译码电路是由组合逻辑实现的。组合逻辑控制器又称硬连线控制器，是早期设计计算机的一种方法。这种控制器中的控制信号直接由各种类型的逻辑门和触发器等组合产生。这样，一旦控制部件构成后，除非重新设计和物理上对它重新连线，否则要想增加新的功能是不可能的。结构上的这种缺陷使得硬连线控制器的设计和调试变得非常复杂而且代价很大。所以，硬连线控制器曾一度被微程序控制器所取代。但是随着新一代机器及VLSI技术的发展，这种控制器又得到了重视，如RISC机广泛使用这种控制器。

1. 组合逻辑控制器原理

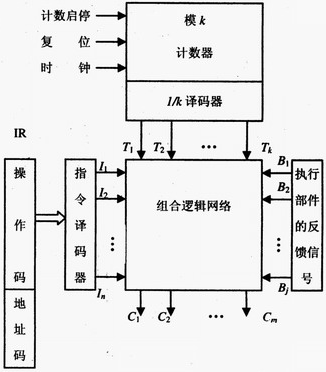


图1 组合逻辑控制器的结构方框图

图1是组合逻辑控制器的结构方框图。逻辑网络的输入信号来源有三个：（1）指令操作码译码器的输出In；（2）来自时序发生器的节拍信号Tk；（3）来自执行部件的反馈信

号Bj。逻辑网络的输出信号就是微操作控制信号，用来对执行部件进行控制。

显然，组合逻辑控制器的基本原理，可描述为：某一微操作控制信号Cm是指令操作码译码器的输出In、时序信号（节拍电位信号Tk）和状态条件信号Bj的逻辑函数。即

Cm＝f(In，Tk，Bj)

用这种方法设计控制器，需要根据每条指令的要求，让节拍电位和时序脉冲有步骤地去控制机器的各有关部分，一步一步地执行指令所规定的微操作，从而在一个指令周期内完成一条指令所规定的全部操作。

一般来说，组合逻辑控制器的设计步骤如下。

（1）绘制指令流程图

为了确定指令执行过程所需的基本步骤，通常是以指令为线索，按指令类型分类，将每条指令归纳成若干微操作，然后根据操作的先后次序画出流程图。

（2）安排指令操作时间表

指令流程图的进一步具体化，把每一条指令的微操作序列分配到各个机器周期的各个时序节拍信号上。要求尽量多的安排公共操作，避免出现互斥。

（3）安排微命令表

以指令流程图为依据，表示出在哪个机器周期的哪个节拍有哪些指令要求哪些微命令。

（4）进行微操作逻辑综合

根据微操作时间表，将执行某一微操作的所有条件（哪条指令、哪个机器周期、哪个节拍和脉冲等）都考虑在内，加以分类组合，列出各微操作产生的逻辑表达式，并加以简化。

（5）实现电路

根据上面所得逻辑表达式，用逻辑门电路的组合或大规模集成电路来实现。

2．实验设计目标

本实验指令译码器的设计相对比较简单，节拍（t1、t2和t3）因素只在存储器读写时需要对存储器地址分时使用时需要考虑，这里暂不考虑节拍脉冲t，也就是说微操作控制信号只是指令操作码In和Bj的函数：

Cm = f（In，Bj）

式中的In主要代表指令操作码IR[15..12]，还有辅助操作码（如算术逻辑指令时的IR[2..0]，我们这里要求只考虑指令操作码IR[15..12]。Bj代表进位标志C和结果为0标志Z。

要求产生的微操作控制信号如下：

op\_code 控制ALU进行8种运算操作的3位编码。

c\_z\_j\_flag 为1表示需要条件转移。

lj\_instruct 为1表示本条指令是条“JMP ADR”指令。

DRWr 为1表示在t3的下降沿将本条指令的执行结果写入目的寄存器。

Mem\_Write 为1表示本条指令有存储器写操作，存储器的地址是目的寄存器的内容。

DW\_intruct 为1表示本条指令是双字指令。

change\_z 为1表示本条指令可能改变z（结果为0）标志。

change\_c 为1表示本条指令可能改变c（进位）标志。

sel\_memdata 为1表示本条指令写入目的寄存器的值来自读存储器。

3．顶层设计实体的引脚要求

引脚要求的对应关系如下：

（1）指令IR[15..12]对应实验台开关S3—S0

进位C对应实验台开关S4

结果为0标志Z对应实验台开关S5

（2）控制信号对应如下：

DRWr 指示灯L0

DW\_intruct 指示灯L1

change\_z 指示灯L2

change\_c 指示灯L3

c\_z\_j\_flag 指示灯L4

op\_code[2..0] 指示灯L10-L8

sel\_memdata 指示灯L11

Mem\_Write 指示灯L12

lj\_instruct 指示灯L13

#### 三、实验步骤

1 将子板上的JTAG端口和PC机的USB口用下载电缆连接。打开实验台电源。

2 打开模块

双击instruction\_decoder.qpf

3 执行Tools→Programmer命令，将instruction\_decoder.sof下载到FPGA中。注意在执行Programmer命令中，应在program/configure下的方框中打勾，然后下载。

（1）实验台上模式开关设置成10000，即组成原理部件实验单步调试模式。

（2）将设计在Quartus Ⅱ下输入，编译后下载到CSR-FLY上的FPGA中。

（3）拨动实验台上的开关S5—S0，改变IR[15..12]、进位标志C和结果为0标志Z，观察指示灯L13-L8, L4-L0显示的控制信号，并填写表。

计算机结构实验报告

2018 **级** 软件工程 **班** **姓名** 荆薇 **学号** 201710513088

表1 指令译码器实验(C=0且Z=0)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 指令 | IR[15..12] | OP[2..0] | c\_z\_j\_flag | lj\_instruct | DRWr | Mem\_Write | DW\_intruct | change\_z | change\_c | sel\_memdata |
| ADD DR,SR | 0000 | 000 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| INC DR | 0001 | 001 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| SUB DR,SR | 0010 | 010 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| DEC DR | 0011 | 011 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| AND DR,SR | 0100 | 100 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| OR DR,SR | 0101 | 101 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| NOT DR | 0110 | 110 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| MOV DR,SR | 0111 | 111 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| JMP ADR | 1000 | 111 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| JNC ADR | 1001 | 111 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| JNZ ADR | 1010 | 111 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MVRD DR,DATA | 1100 | 111 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| LDR DR,SR | 1101 | 111 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| STR SR,DR | 1110 | 111 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

表2指令译码器实验(C=0且Z=1)

L4 L13 L0 L12 L1 L2 L3 L11

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 指令 | IR[15..12] | OP[2..0] | c\_z\_j\_flag | lj\_instruct | DRWr | Mem\_Write | DW\_intruct | change\_z | change\_c | sel\_memdata |
| ADD DR,SR | 0000 | 000 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| INC DR | 0001 | 001 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| SUB DR,SR | 0010 | 010 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| DEC DR | 0011 | 011 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| AND DR,SR | 0100 | 100 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| OR DR,SR | 0101 | 101 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| NOT DR | 0110 | 110 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| MOV DR,SR | 0111 | 111 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| JMP ADR | 1000 | 111 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| JNC ADR | 1001 | 111 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| JNZ ADR | 1010 | 111 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MVRD DR,DATA | 1100 | 111 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| LDR DR,SR | 1101 | 111 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| STR SR,DR | 1110 | 111 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

表3指令译码器实验(C=1且Z=0)

L10-l8 L4 L13 L0 L12 L1 L2 L3 L11

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 指令 | IR[15..12] | OP[2..0] | c\_z\_j\_flag | lj\_instruct | DRWr | Mem\_Write | DW\_intruct | change\_z | change\_c | sel\_memdata |
| ADD DR,SR | 0000 | 000 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| INC DR | 0001 | 001 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| SUB DR,SR | 0010 | 010 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| DEC DR | 0011 | 011 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| AND DR,SR | 0100 | 100 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| OR DR,SR | 0101 | 101 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| NOT DR | 0110 | 110 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| MOV DR,SR | 0111 | 111 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| JMP ADR | 1000 | 111 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| JNC ADR | 1001 | 111 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| JNZ ADR | 1010 | 111 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MVRD DR,DATA | 1100 | 111 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| LDR DR,SR | 1101 | 111 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| STR SR,DR | 1110 | 111 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

表4指令译码器实验(C=1且Z=1)

L10-l8 L4 L13 L0 L12 L1 L2 L3 L11

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 指令 | IR[15..12] | OP[2..0] | c\_z\_j\_flag | lj\_instruct | DRWr | Mem\_Write | DW\_intruct | change\_z | change\_c | sel\_memdata |
| ADD DR,SR | 0000 | 000 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| INC DR | 0001 | 001 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| SUB DR,SR | 0010 | 010 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| DEC DR | 0011 | 011 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| AND DR,SR | 0100 | 100 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| OR DR,SR | 0101 | 101 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| NOT DR | 0110 | 110 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| MOV DR,SR | 0111 | 111 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| JMP ADR | 1000 | 111 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| JNC ADR | 1001 | 111 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| JNZ ADR | 1010 | 111 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MVRD DR,DATA | 1100 | 111 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| LDR DR,SR | 1101 | 111 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| STR SR,DR | 1110 | 111 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

回答问题：

1. **说明本实验实现的部分是什么，在CPU实现中起的作用是？**

**答：**本实验的部分是指令译码器将指令操作码进行译码，产生一系列微操作控制信号（op\_code、c\_z\_j\_flag、lj\_instruct、DRWr、Mem\_Write、DW\_intruct、change\_z、change\_c、sel\_memdata）

在CPU中的作用是对现行指令进行分析后，确定指令类型和指令所要完成的操作以及寻址方式，按译码得到的微操作信号控制计算机中的所有部件按指令操作。

1. **结合计算机组成与系统结构理论知识，说明一条指令转化为CPU内部控制信号的过程？**

答：

在取指令阶段，指令从主存中被取到指令寄存器中。用程序计数器PC中的数值表示当前指令在主存中的位置，指令取出后PC的数值自增。取出指令后进入指令译码阶段，指令译码器按照预定的指令格式，对取回的指令进行拆分和解释，识别区分出不同的指令类别以及各种获取操作数的方法，产生不同的微操作序列。

1. **结合实验理解一下MOV DR,SR 和JMP ADR的OP码为何为“111”而不是其他?**

答：8种运算操作用3位2进制编码来表示，111是其中的最后一个，表示转移类指令。MOV和JMP它们都会进行地址的转移。