北京邮电大学课程设计报告

<table>
<tr><td>课程设计名称</td><td colspan="2">计算机组成原理课程设计</td><td>学 院</td><td>计算机学院</td><td>指导教师</td><td>张杰老师、
靳秀国老师、</td></tr>
<tr><td>班 级</td><td>班内序号</td><td colspan="2">学 号</td><td>学生姓名</td><td colspan="2">成绩</td></tr>
<tr><td>2018211318</td><td>4</td><td colspan="2">2018210255</td><td>史嘉程</td><td colspan="2"></td></tr>
<tr><td>2018211318</td><td>23</td><td colspan="2">2018212130</td><td>刘俊</td><td colspan="2"></td></tr>
<tr><td>2018211318</td><td>8</td><td colspan="2">2018210789</td><td>范珈诚</td><td colspan="2"></td></tr>
<tr><td>课
程
设
计
内
容</td><td colspan="6">① 基本内容：基于 TEC-8 的数据通路设计一个硬布线控制器，构成一个完整的 CPU，该控制器能完成控制台的操作，并能执行规定的基本指令。
② 实验方法：本实验依赖于实验室 TEC-8 实验系统的资源，采用 VHDL 进行开发。
③ 团队分工：控制器的设计（各模块的功能分析、整体设计）和实现（程序调试、文档编写工作）由三名组员共同完成。其中刘俊主要负责最终框架的设计；范珈诚主要负责代码实现并进行测试；史嘉程主要负责最终的复盘以及报告编写工作。
④ 实验成果：
实现了基本顺序硬布线控制器和流水型硬布线控制器；
实现了指定程序运行的起始位置的功能（修改 PC 指针）和扩展了指令集（DEC、XOR、NOP）；
运用数字逻辑的知识（卡诺图）优化控制器速度。</td></tr>
<tr><td>学生
课程设计
报告
（附页）</td><td colspan="6">详情请见小组实验报告和源代码</td></tr>
<tr><td>课
程
设
计
成
绩
评
定</td><td colspan="6">遵照实践教学大纲并根据以下四方面综合评定成绩：
1、课程设计目的任务明确，选题符合教学要求，份量及难易程度
2、团队分工是否恰当与合理
3、综合运用所学知识，提高分析问题、解决问题及实践动手能力的效果
4、是否认真、独立完成属于自己的课程设计内容，课程设计报告是否思路清晰、文字通顺、书写规范
评语:

成绩:

指导教师签名：

年 月 日</td></tr>
</table>

注：评语要体现每个学生的工作情况，可以加页。

北京郵電大學

Beijing University of Posts and Telecommunications

本 科 生 课 程 设 计

题目：基于 TEC-8 实验系统的模型机控制器

课 程 名 称 ：计算机组成原理课程设计

姓 名（学号）：史嘉程 (2018210255)

刘 俊 (2018212130)

范珈诚 (2018210789)

学 院 ：计算机学院

专 业 ：计算机大类

指 导 教 师 ：张 杰老师、靳秀国老师

二〇二〇年 九月

目录

1. 总体概述

1.1. 教学目的

1) 融会贯通计算机组成原理课程各章节的内容

- 通过知识的综合运用加深对计算机系统各模块的工作原理及相互联系的认识，特别是对硬布线控制器的认识
- 建立清晰的整机概念

2) 掌握硬布线控制器的设计方法

3) 学习运用可编程逻辑技术进行逻辑设计和调试的基本步骤和方法

- 熟悉集成开发软件中设计、模拟调试工具的使用
- 体会可编程逻辑技术相对于传统开发技术的优点

1) 培养科学研究的独立工作能力，取得工程设计与组装调试的实践经验

1.2 设计任务

- **设计**

  按照给定的数据格式、指令系统和数据通路，在所提供的器件范围内(TEC-8)，设计一个基于硬布线控制器的顺序（流水）模型处理器

- **实现**

  根据设计方案，在通用实验台上进行组装并调试成功

1.3 实验环境

在本次课程设计的过程中，我们小组使用了 Altera 公司的综合性 CPLD/FPGA 开发软件 Quartus Ⅱ。这款 EDA 带有的功能仿真的功能，为我们控制器的设计提供了极大的便利，加快了整体开发速

度。

本次课程设计过程中，我们小组所依赖的实验环境如下：

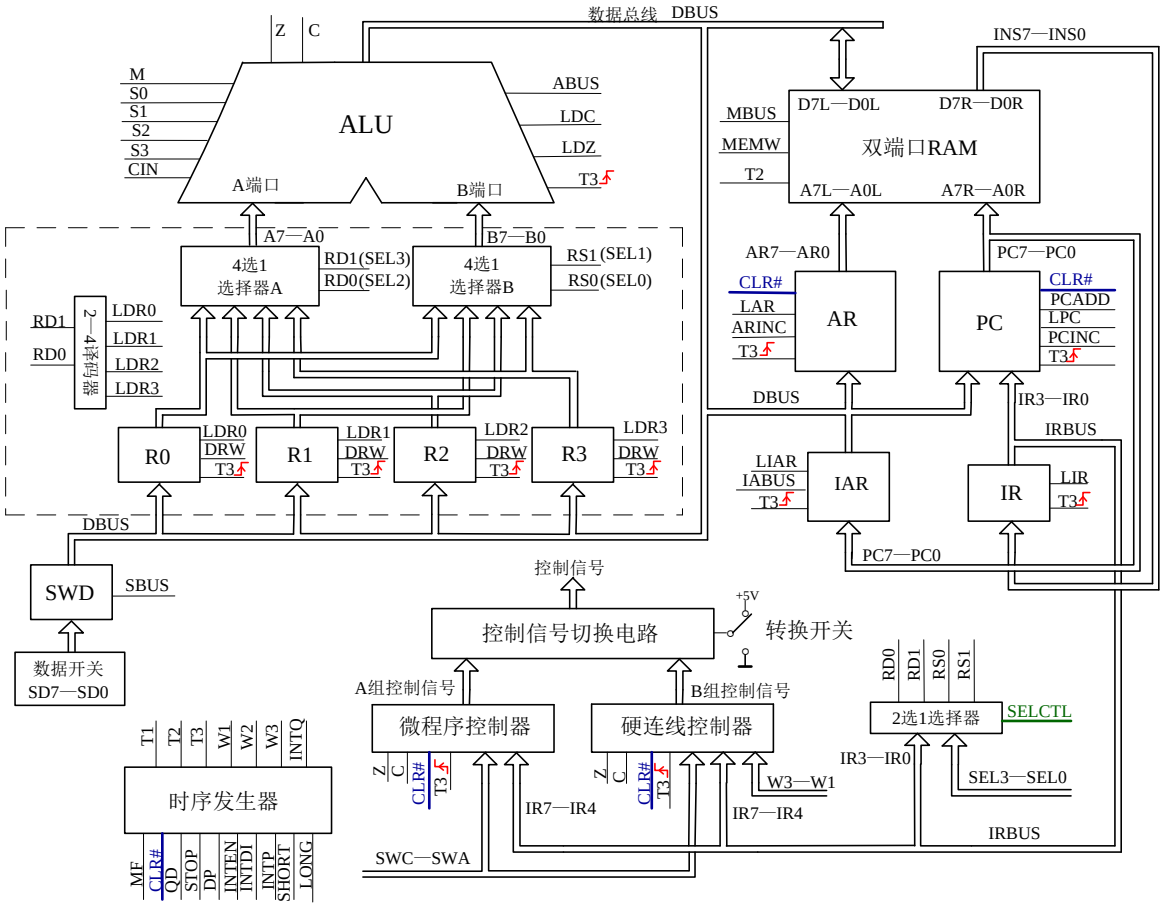
① 开发环境：Windows 系统、Quartus Ⅱ 9.1(64-Bit)的 PC 机

② 调试工具：数字存储示波器一台、直流万用表一只、逻辑测试笔一支

③ 实验平台：TEC-8 计算机硬件综合实验系统

1.4 数据通路

本次课程设计是依赖于学校实验室提供的 TEC-8 计算机硬件综合实验系统而进行的，而该实验系统的数据通路如下图所示：



TEC-8 计算机硬件综合实验系统数据通路

2 设计思路

2.1 控制器的设计思路

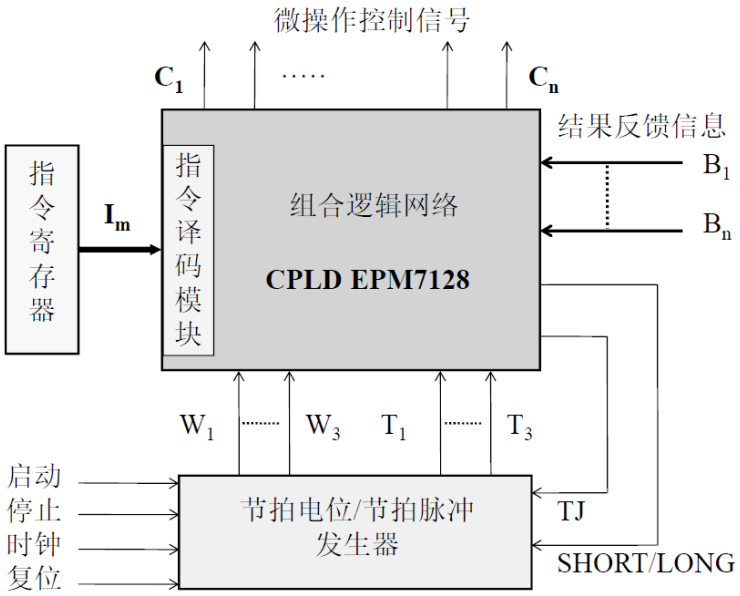
硬布线控制器是将控制部件做成产生专门固定时序控制信号的逻辑电路，产生各种控制信

号，因而又称为组合逻辑控制器，这种逻辑电路是由一种门电路和触发器构成的复杂树形逻辑网络。当执行不同的机器指令时，通过激活一系列彼此很不相同的控制信号来实现对指令的解释。

根据硬布线控制器的基本原理，针对每个控制信号C_x，可以列出它的函数表达式：

$$C_x = f(I_m, M_i, T_k, B_j)$$

其中：I_m代表机器指令操作码译码器 d 输出信号，M_i代表节拍信号发生器输出的节拍信号，T_k代表时序信号发生器输出的时序信号，B_j则是状态条件判断信号。

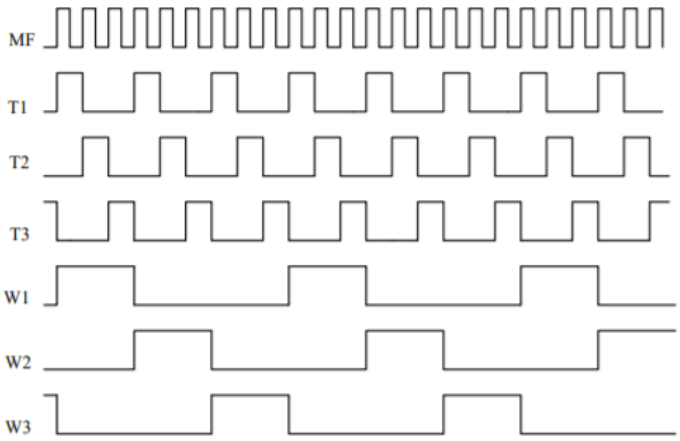


硬布线控制器逻辑模块图

2.2 控制器的时序系统

本实验只要求设计硬布线控制器，而时序系统和数据通路等通过实验室提供的 TEC-8 实验模拟系统完成。因此我们设计的硬布线控制器需要对接 TEC-8 实验系统的时序系统。

TEC-8 模型计算机主时钟 MF 的频率为 1MHz，执行一条微指令需要 3 个节拍脉冲 T1、T2、T3。TEC-8 模型计算机时序采用不定长机器周期，绝大多数指令采用 2 个机器周期 W1、W2，少数指令采用 3 个机器周期 W1、W2、W3。在我们设计的指令集中，不包含采用一个机器周期的指令，所有指令均为二个机器周期或者三个机器周期。



TEC-8 模型计算机时序图

2.3 TEC-8 顺序控制器的设计思路

在 TEC-8 实验系统中，时序信号$T_k(T_1 \sim T_3)$已经直接输送至数据通路，我们设计的组合逻辑电路不需要将T_k作为控制器输入；因为 TEC-8 实验系统的设计较为简单，以及字长的限制，机器指令系统的设计也相应的简化，可不使用专门的操作码译码器，将操作码$IR_4 \sim IR_7$直接当作I_m；控制台操作看作特殊指令，控制台开关信号 SWC、SWB、SWA 也看做I_m；时序模块的节拍信号则当作组合逻辑函数表达式中的M_i；B_j信号可以包括数据通路中的进位标志 C，零标志 Z 以及其他状态条件判断信号。

2.4 机器指令的节拍数

TEC-8 模型计算机时序采用不定长机器周期，绝大多数指令，节拍发生器产生节拍电位信号期 W1、W2。

但是我们设计的控制器也考虑到了一个节拍电位的指令和三个节拍电位的指令，并通过在时序电路中加入了新的控制信号 SHORT 和 LONG。对于一个节拍电位的指令，控制信号 SHORT 会通知节拍发生器在 W1 节拍后不再产生节拍 W2；对于三个节拍电位的指令，控制信号 LONG 会通知节拍发生器在 W2 之后继续产生节拍 W3。

3 详细设计

3.1 基本顺序硬布线控制器

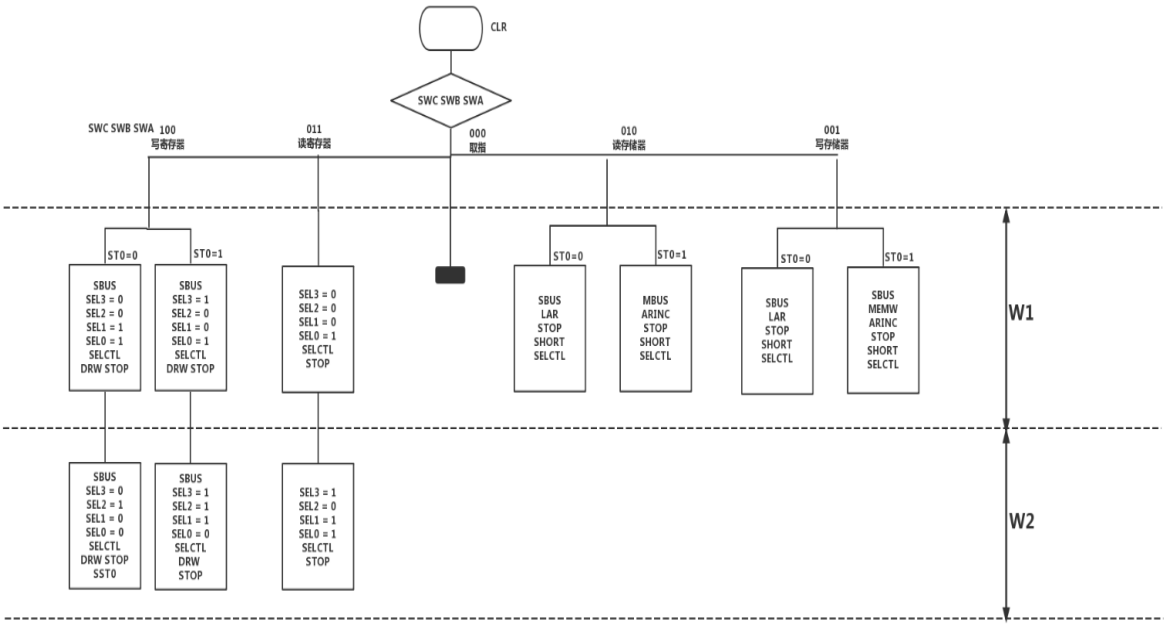
3.1.1 指令集

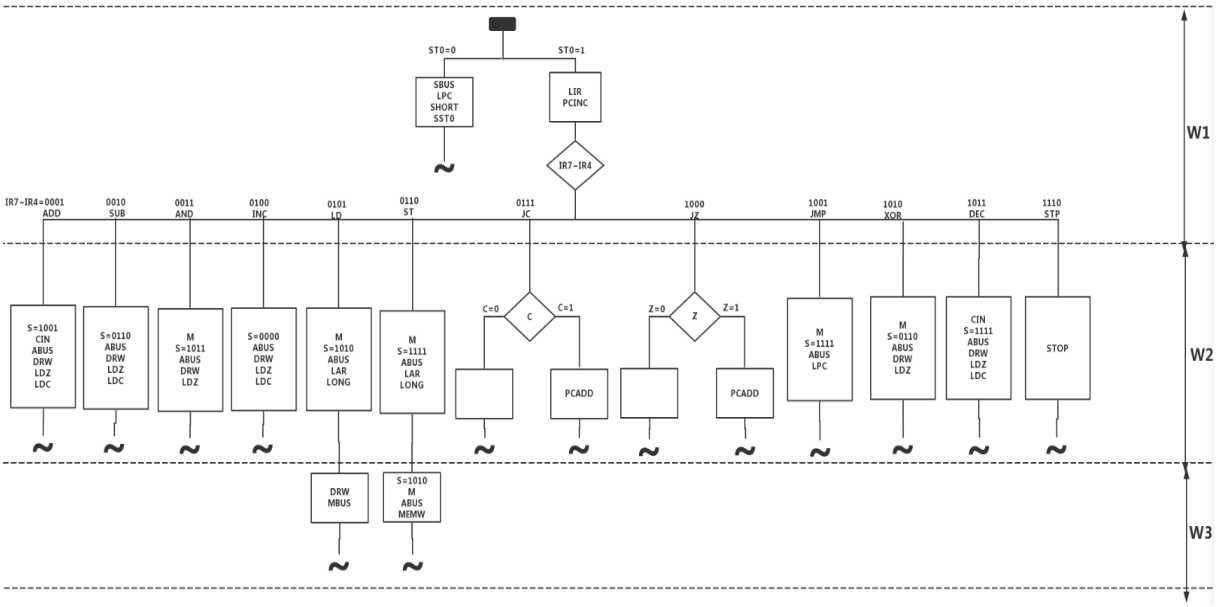
本次设计的基本顺序硬布线控制器总共包含了 12 条指令，在 TEC-8 原有的指令集上

添加了新的 DEC、XOR 两条指令，以取代原有的中断指令。

名称	汇编语言	功能	指令格式		
			IR7 IR6 IR5 IR4	IR3 IR2	IR1 IR0
加法	ADD Rd, Rs	Rd←Rd+Rs	0001	Rd	Rs
减法	SUB Rd, Rs	Rd←Rd-Rs	0010	Rd	Rs
逻辑与	AND Rd, Rs	Rd←Rd and Rs	0011	Rd	Rs
加 1	INC Rd	Rd←Rd+1	0100	Rd	XX
取数	LD Rd, [Rs]	Rd←[Rs]	0101	Rd	Rs
存数	ST Rs, [Rd]	Rs→[Rd]	0110	Rd	Rs
C 条件转移	JC addr	如果 C=1，则 PC←@+offset	0111	offset	
Z 条件转移	JZ addr	如果 Z=1，则 PC←@+offset	1000	offset	
无条件转移	JMP [Rd]	PC←Rd	1001	Rd	XX
异或指令	XOR Rd, Rs	Rd←Rd xor Rs	1010	Rd	Rs
减 1	DEC Rd	Rd←Rd-1	1011	Rd	XX
停机	STOP	暂停运行	1110	XX	XX

根据上述的指令集，我们设计出了所有指令指令周期的流程图，以节拍电位 W1~W3 为时间单位。流程图如下所示：





3.1.2 指令集组合逻辑译码表

根据上述设计的指令集和流程图，我们初步设计的硬布线控制器的组合逻辑译码表如下：

指令 IR	ADD	SUB	AND	INC	LD	ST	JC	JZ	JMP	XOR	DEC	STP
LIR	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
PCINC	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
S3	W2		W2		W2	W2+W3			W2		W2	
S2		W2				W2			W2	W2	W2	
S1		W2	W2		W2	W2+W3			W2	W2	W2	
S0	W2		W2			W2			W2		W2	
CIN	W2										W2	
ABUS	W2	W2	W2	W2	W2	W2+W3			W2	W2	W2	
DRW	W2	W2	W2	W2	W3					W2	W2	
LDZ	W2	W2	W2	W2						W2	W2	
LDC	W2	W2		W2							W2	
M			W2		W2	W2+W3			W2	W2		
LAR					W2	W2						
LONG					W2	W2						
MBUS					W3							

MEMW						W3						
PCADD							C*W2	Z*W2				
LPC									W2			
STOP												W2

3.1.3 指令组合逻辑表达式及其优化

根据组合逻辑译码表，我们可以得到信号逻辑表达式如下：

$$LIR = PCINC = W1$$

$$S3 = W2 * (ADD + AND + LD + ST + JMP + DEC) + W3 * ST$$

$$S2 = W2 * (SUB + JMP + ST + XOR + DEC)$$

$$S1 = W2 * (SUB + AND + LD + ST + JMP + XOR + DEC) + W3 * ST$$

$$S0 = W2 * (ADD + AND + ST + JMP + DEC)$$

$$CIN = W2 * (ADD + DEC)$$

$$ABUS = W2 * (ADD + SUB + AND + INC + LD + ST + JMP + XOR + DEC) + W3 * ST$$

$$DRW = W2 * (ADD + SUB + AND + INC + XOR + DEC) + W3 * LD$$

$$LDZ = W2 * (ADD + SUB + AND + INC + XOR + DEC)$$

$$LDC = W2 * (ADD + SUB + INC + DEC)$$

$$M = W2 * (AND + LD + ST + JMP + XOR) + W3 * ST$$

$$LAR = LONG = W2 * (LD + ST)$$

$$MBUS = W3 * LD$$

$$MEMW = W3 * ST$$

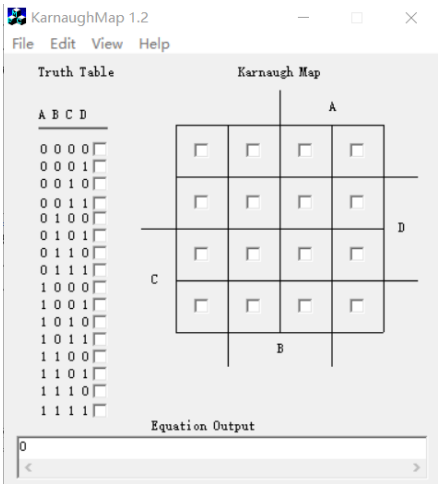
$$PCADD = W2 * (C * JC + Z * JZ)$$

$$LPC = W2 * JMP$$

$$STOP = W2 * STP$$

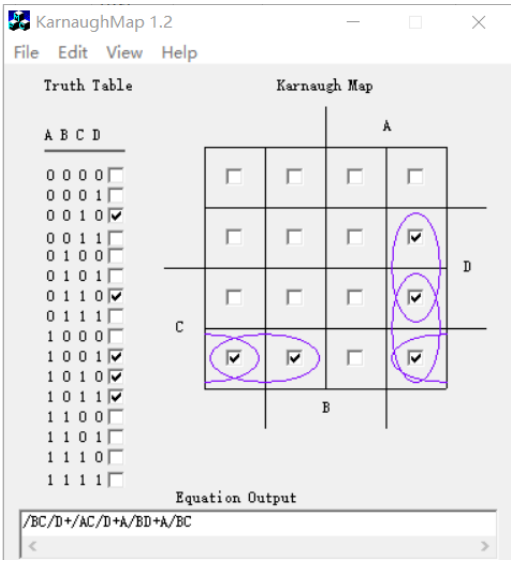
但是在实际的代码实现中，我们发现，虽然直接用上述的逻辑表达式搭配 CASE 分支语句简洁直观，代码清晰，可读性强，且便于后续的修改调试、管理维护等工作，但在运行时我们发现，CASE 分支语句会时指令执行速度大幅度变慢。因此，我们考虑不采用 CASE 分支语句，而是将微指令的组合逻辑电路利用卡诺图进行化简，得到各信号对应的 IR 信号与节拍信号的与或逻辑表达式。

我们采用了软件 KarnaughMap 1.2 进行卡诺图的化简工作。



以指令 LDC 为例，介绍利用卡诺图化简的思路：

$S2 = W2 * (SUB + JMP + ST + XOR + DEC)$，而 SUB，JMP，ST，XOR，DEC 对应的 IR7~IR4 分别为 0010，1001，0110，1010，1011。因此，以 A、B、C、D 分别为 IR7、IR6、IR5、IR4，则可以得到卡诺图如下：



节拍 W2 下的卡诺图

因为 S2 只包含节拍 W2，不包含节拍 W3，因此通过一个卡诺图即可得出最终结果。因此化简可得，$S2 = W2 * (\sim IR6 * IR5 * \sim IR4 + \sim IR7 * IR5 * \sim IR4 + IR7 * \sim IR6 * IR4 + IR7 * \sim IR6 * IR5)$。

综上，通过卡诺图化简后的组合逻辑表达式为：

$LIR = PCINC = W1 * ST0$

$$S3 = W2 * (\sim IR6 * IR4 + \sim IR7 * \sim IR5 * IR4 + \sim IR7 * IR6 * IR5 * `IR4) + W3 * (\sim IR7 * IR6 * IR5 * \sim IR4)$$

$$S2 = W2 * (\sim IR6 * IR5 * \sim IR4 + \sim IR7 * IR5 * \sim IR4 + IR7 * \sim IR6 * IR4 + IR7 * \sim IR6 * IR5)$$

$$S1 = W2 * (\sim IR6 * IR5 + \sim IR7 * IR5 * \sim IR4 + IR7 * \sim IR6 * IR4 + \sim IR7 * IR6 * \sim IR5 * IR4) + W3 * (\sim IR7 * IR6 * IR5 * \sim IR4)$$

$$S0 = W2 * (\sim IR6 * IR4 + \sim IR7 * IR6 * IR5 * \sim IR4)$$

$$CIN = W2 * (\sim IR7 * \sim IR6 * \sim IR5 * IR4 + IR7 * \sim IR6 * IR5 * IR4)$$

$$ABUS = W2 * (\sim IR6 * IR4 + \sim IR6 * IR5 + \sim IR7 * \sim IR5 * \sim IR4 + \sim IR7 * IR6 * \sim IR4 + \sim IR7 * IR6 * \sim IR5) + W3 * (\sim IR7 * IR6 * IR5 * \sim IR4)$$

$$DRW = W2 * (A\sim IR6 * IR5 + \sim IR7 * \sim IR6 * IR4 + \sim IR7 * IR6 * \sim IR5 * \sim IR4) + W3 * (\sim IR7 * IR6 * \sim IR5 * IR4)$$

$$LDZ = W2 * (\sim IR6 * IR5 + \sim IR7 * \sim IR6 * IR4 + \sim IR7 * IR6 * \sim IR5 * \sim IR4)$$

$$LDC = W2 * (\sim IR7 * \sim IR6 * \sim IR5 * IR4 + \sim IR7 * IR6 * \sim IR5 * \sim IR4 + IR7 * \sim IR6 * IR5 * IR4)$$

$$M = W2 * (\sim IR7 * \sim IR6 * \sim IR5 * IR4 + \sim IR7 * IR6 * \sim IR5 * \sim IR4 + \sim IR7 * IR6 * IR5 * \sim IR4 + IR7 * \sim IR6 * \sim IR5 * IR4 + IR7 * \sim IR6 * IR5 * \sim IR4) + W3 * (\sim IR7 * IR6 * IR5 * \sim IR4)$$

$$LAR = LONG = W2 * (\sim IR7 * IR6 * \sim IR5 * IR4 + \sim IR7 * IR6 * IR5 * \sim IR4)$$

$$MBUS = W3 * (\sim IR7 * IR6 * \sim IR5 * IR4)$$

$$MEMW = W3 * (\sim IR7 * IR6 * IR5 * \sim IR4)$$

$$PCADD = W2 * (C * \sim IR7 * IR6 * IR5 * IR4 + Z * IR7 * \sim IR6 * \sim IR5 * \sim IR4)$$

$$LPC = W2 * (IR7 * \sim IR6 * \sim IR5 * IR4) + W1 * (\sim ST0)$$

$$STOP = W2 * (IR7 * IR6 * IR5 * \sim IR4) + W1 * \sim ST0$$

3.1.4 控制台操作组合逻辑表达式

- **SWCBA=100 写寄存器**

$SBUS = 1$

$SEL3 = ST0$

$SEL2 = W2$

$SEL1 = \sim ST0 * W1 + ST0 * W2$

$SEL0 = 0$

$SELCTL = DRW = STOP = 1$

$SSTP = \sim ST0 * W2$

- **SWCBA=011 读寄存器**

$SEL3 = W2$

$SEL2 = 0$

$SEL1 = W2$

$SEL0 = 1$

$SELCTL = STOP = 1$

$SSTP = \sim ST0 * W2$

- **SWCBA=010 读存储器**

$SBUS = LAR = \sim ST0 * W1$

$SELCTL = SHORT = STOP = W1$

$ARINC = MBUS = ST0 * W1$

- **SWCBA=001 写存储器**

$SBUS = W1$

$SELCTL = SHORT = STOP = W1$

$ARINC = MEMW = ST0 * W1$

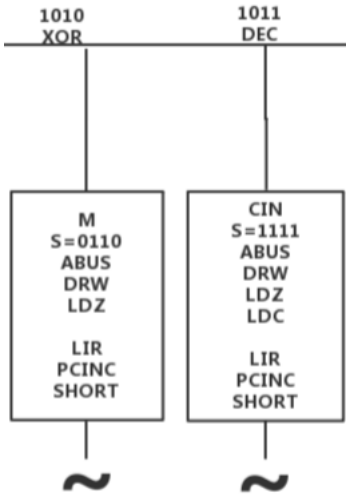
3.1.5 扩展功能的实现

在基本顺序硬布线控制器中，我们小组除了实现了控制器的基本功能（基本指令集），还完成了课程设计任务中的两个扩展功能。下面我们将介绍这两个额外功能：

① 在原指令基础上进行扩展

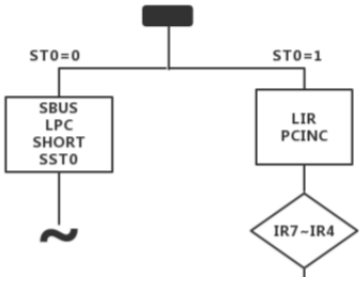
除了 TEC-8 实验系统的基本指令集，我们额外实现了两个指令：XOR 指令和 DEC

指令。DEC 指令与原指令集中的 INC 指令具体实现，只是在 ALU 中由加法运算变为减法运算，在此不再赘述；XOR 指令的实现是为了便于后续数据的交换操作，同样是依赖于算术逻辑部件 ALU，具体实现与 AND 指令类似，而在 ALU 的操作由逻辑与改为了异或运算。



② 修改 PC 指针功能（任意指针）

在实现该功能时，我们小组引入了三个个新的信号 ST0、SST0 和 CLR。其中 CLR 信号的作用是将 ST0、SST0 置为 0。ST0、SST0 作为中间信号使用，原理与读写存储器一致。ST0 用来标志现在执行的是这条指令是什么阶段，利用 ST0 可以使得部分代码只在特定时期阶段执行，SST0 则是用来在指令或程序需要执行下一个阶段时用于更改 ST0 的值。SST0 信号置为 0 时，下一阶段 ST0 置为 1。



当 SWCBA=000（启动程序）时，ST0 信号初始为 0，通过 SBUS 和 LPC 手动修改 PC 指针，接着通过触发 SST0 信号，在进入程序入口之后将 ST0 修改为 1，便可进入正常取指执行的工作流程中。

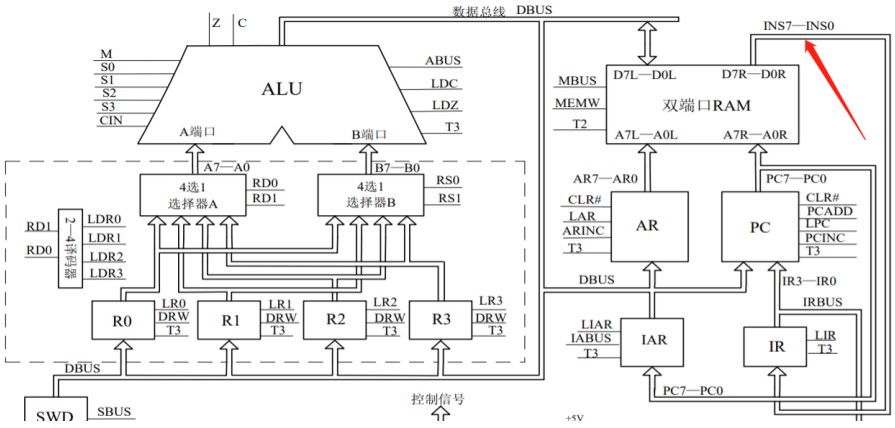
3.2 实现指令流水线的硬布线控制器

3.2.1 实现流水线的原理

流水 CPU 以机器运行程序的时间并行性为原理，通过实现同一节拍内进行多个操作，大幅度提高系统性能。流水线中要求尽量多的并行操作，以便充分利用硬件资源，减少闲置。流水线的设计分为横向设计和纵向设计：横向设计中把不冲突的、可以同时实现的控制放在同一节拍中；纵向设计中，顺序控制要连同并发控制一起考虑，尤其是出现冲突时，要前后错开。

非流水模型机的运行逻辑：每一条指令都在执行完执行周期之后，下一条指令再执行取指周期。而经过我们小组的讨论，我们认为可以将指令的取指周期与上一条指令的执行周期放入同一个节拍，这样实现的流水线控制器下，所有指令将没有一个专门用于取值的节拍，当前一个指令的执行周期结束之后自动进入下一个指令的执行周期。

实现原理：TEC-8 中将双端口存储器中的指令打入 IR 有单独的数据通路，不需要应对流水线各段争用总线的问题。如下图所示，指令通过双端口 RAM 中的 IRBUS（红箭头标注）直接打入 IR，不需要经过 DBUS，而 IRBUS 在执行周期中不会被任何指令占用，因此非常好的规避了竞争总线的问题。



IRBUS 的位置

实现方法：在每条指令的执行周期中都加入 LIR 和 PCINC 信号，使得控制器取出下一条指令。

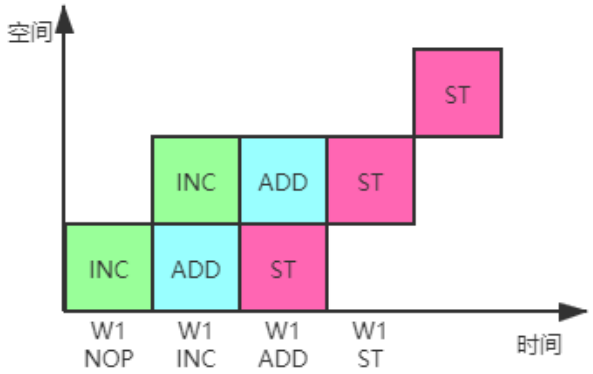
因为通过上述思想设计的流水线控制器中的每一个指令都没有节拍用来取值，所以第一条指令如何取值是一个比较关键的问题。对此，我们引入了一个新的指令 NOP，指令代

码是 00H。如此设计是因为按下复位按钮 CLR 后，IR 被复位为 00H，于此正好对应 NOP 指令。这样在 NOP 指令的执行周期就会自动完成下一条指令的取指。

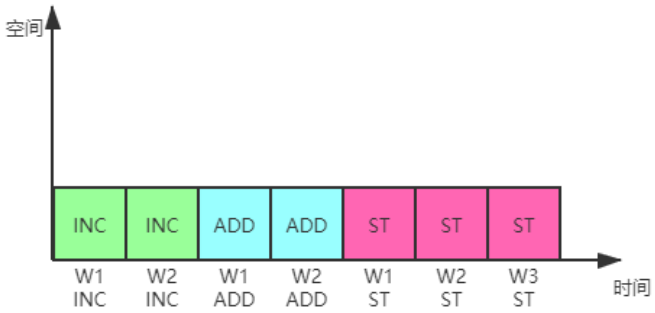
以下是我们设计的流水线控制器的实例分析：

地址	机器指令
00H	INC R0
01H	ADD R0, R1
02H	ST R1, [R2]

其中：INC 和 ADD 是二周期指令，ST 指令是三周期指令。流水线时空图如下所示：



流水线时空图



非流水线时空图

解析：系统刚开始运行时，IR 中被 CLR 置为 00H，因此第一个节拍执行 NOP 指令的执行周期以及 INC 指令的取指周期；第二个节拍，执行 INC 指令的操作，同时读取指令 ADD；第三个节拍同上。第四个节拍，因为 ST 是三周期指令，因此不执行取指操作，到第五个节拍，即 ST 指令的最后一个节拍再进行下一条指令的取指。通过与非流水线的时空图进行比较，执行完这三条指令，流水线控制器所花费的节拍数为 5，非流水线需要的节拍数

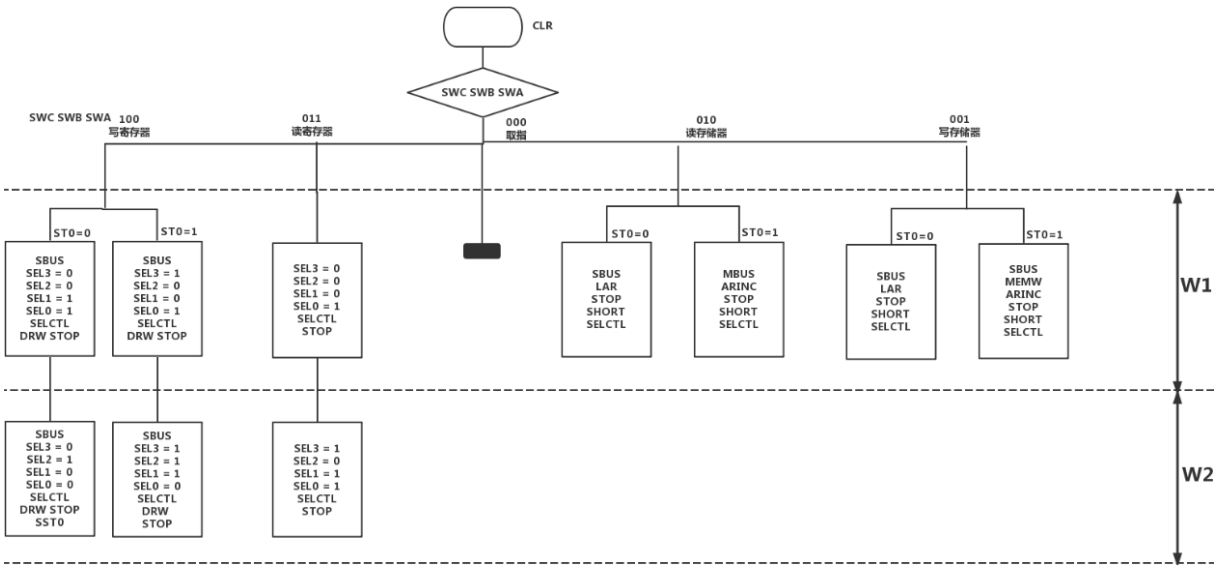
为 7。加速比达到了$S=\frac{7}{5}=1.4$。理想状况下，指令条数趋于无穷大时，三周期指令为 2 条，二周期指令为 11 条，则加速比约为$S=\frac{\frac{n*(2*3+11*2)}{13}}{\frac{n*(2*2+11)}{13}}=\frac{28}{15}=1.8$。

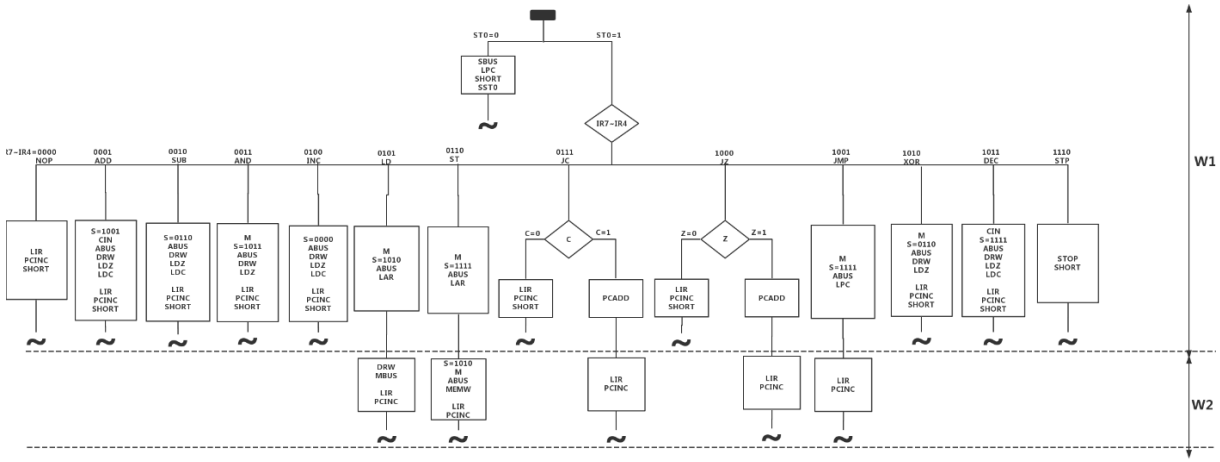
3.2.2 指令集

在原有的指令集基础上，添加了 NOP 指令。

名称	汇编语言	功能	指令格式		
			IR7 IR6 IR5 IR4	IR3 IR2	IR1 IR0
空指令	NOP	无	0000	XX	XX
加法	ADD Rd, Rs	Rd←Rd+Rs	0001	Rd	Rs
减法	SUB Rd, Rs	Rd←Rd-Rs	0010	Rd	Rs
逻辑与	AND Rd, Rs	Rd←Rd and Rs	0011	Rd	Rs
加 1	INC Rd	Rd←Rd+1	0100	Rd	XX
取数	LD Rd, [Rs]	Rd←[Rs]	0101	Rd	Rs
存数	ST Rs, [Rd]	Rs→[Rd]	0110	Rd	Rs
C 条件转移	JC addr	如果 C=1，则 PC←@+offset	0111	offset	
Z 条件转移	JZ addr	如果 Z=1，则 PC←@+offset	1000	offset	
无条件转移	JMP [Rd]	PC←Rd	1001	Rd	XX
异或指令	XOR Rd, Rs	Rd←Rd xor Rs	1010	Rd	Rs
减 1	DEC Rd	Rd←Rd-1	1011	Rd	XX
停机	STOP	暂停运行	1110	XX	XX

根据上述的指令集，我们画出的指令流程图如下：





3.2.3 指令集组合译码表

根据上述设计的指令集和流程图，我们初步设计的硬布线控制器的组合逻辑译码表如下：

指令 IR	NOP	ADD	SUB	AND	INC	LD	ST	JC	JZ	JMP	XOR	DEC	STP
LIR	W1	W1	W1	W1	W1	W2	W2	~C*W1 +C*W2	~Z*W1 +Z*W2	W2	W1	W1	
PCINC	W1	W1	W1	W1	W1	W2	W2	~C*W1 +C*W2	~Z*W1 +Z*W2	W2	W1	W1	
S3		W1		W1		W1	W1+W2			W1		W1	
S2			W1				W1			W1	W1	W1	
S1			W1	W1		W1	W1+W2			W1	W1	W1	
S0		W1		W1			W1			W1		W1	
CIN		W1										W1	
ABUS		W1	W1	W1	W1	W1	W1+W2			W1	W1	W1	
DRW		W1	W1	W1	W1	W2					W1	W1	
LDZ		W1	W1	W1	W1						W1	W1	
LDC		W1	W1		W1							W1	
M				W1		W1	W1+W2			W1	W1		
LAR						W1	W1						
SHORT	W1	W1	W1	W1	W1			~C*W1	~Z*W1		W1	W1	W1
MBUS						W2							
MEMW							W2						
PCADD								C*W1	Z*W1				

LPC										W1			
STOP													W1

3.2.4 指令组合逻辑表达式及其优化

根据组合逻辑译码表，我们可以得到信号逻辑表达式如下：

$$LIR = PCINC = W1 * (NOP + ADD + SUB + AND + INC + \sim C * JC + \sim Z * JZ + XOR + DEC) + W2 * (LD + ST + C * JC + Z * JZ + JMP)$$

$$S3 = W1 * (ADD + AND + LD + ST + JMP + DEC) + W2 * ST$$

$$S2 = W1 * (SUB + JMP + ST + XOR + DEC)$$

$$S1 = W1 * (SUB + AND + LD + ST + JMP + XOR + DEC) + W2 * ST$$

$$S0 = W1 * (ADD + AND + ST + JMP + DEC)$$

$$CIN = W1 * (ADD + DEC)$$

$$ABUS = W1 * (ADD + SUB + AND + INC + LD + ST + JMP + XOR + DEC) + W2 * ST$$

$$DRW = W1 * (ADD + SUB + AND + INC + XOR + DEC) + W2 * LD$$

$$LDZ = W1 * (ADD + SUB + AND + INC + XOR + DEC)$$

$$LDC = W1 * (ADD + SUB + INC + DEC)$$

$$M = W1 * (AND + LD + ST + JMP + XOR) + W2 * ST$$

$$LAR = LONG = W1 * (LD + ST)$$

$$SHORT = W1 * (NOP + ADD + SUB + AND + INC + \sim C * JC + \sim Z * JZ + XOR + DEC + STP)$$

$$MBUS = W2 * LD$$

$$MEMW = W2 * ST$$

$$PCADD = W1 * (C * JC + Z * JZ)$$

$$LPC = W1 * JMP$$

$$STOP = W1 * STP$$

通过卡诺图化简结果如下：

$$LIR = PCINC = W1 * (\sim IR6 * IR5 + \sim IR7 * \sim IR6 + \sim IR7 * \sim IR5 * \sim IR4 + \sim C * \sim IR7 * IR6 * IR5 * IR4 + \sim Z * IR7 * \sim IR6 * \sim IR5 * \sim IR4) + W2 * (\sim IR7 * IR6 * \sim IR5 * IR4 + \sim IR7 * IR6 * IR5 * \sim IR4 + IR7 * \sim IR6 * \sim IR5 * IR4 + C * \sim IR7 * * IR6 * IR5 * IR4 + Z * IR7 * \sim IR6 * \sim IR5 * \sim IR4)$$

$$S3 = W1 * (\sim IR6 * IR4 + \sim IR7 * \sim IR5 * IR4 + \sim IR7 * IR6 * IR5 * `IR4) + W2 * (\sim IR7 * IR6 * IR5 * \sim IR4)$$

$$S2 = W1 * (\sim IR6 * IR5 * \sim IR4 + \sim IR7 * IR5 * \sim IR4 + IR7 * \sim IR6 * IR4 + IR7 * \sim IR6 * IR5)$$

$$S1 = W1 * (\sim IR6 * IR5 + \sim IR7 * IR5 * \sim IR4 + IR7 * \sim IR6 * IR4 + \sim IR7 * IR6 * \sim IR5 * IR4) + W3 * (\sim IR7 * IR6 * IR5 * \sim IR4)$$

$$S0 = W1 * (\sim IR6 * IR4 + \sim IR7 * IR6 * IR5 * \sim IR4)$$

$$CIN = W1 * (\sim IR7 * \sim IR6 * \sim IR5 * IR4 + IR7 * \sim IR6 * IR5 * IR4)$$

$$ABUS = W1 * (\sim IR6 * IR4 + \sim IR6 * IR5 + \sim IR7 * \sim IR5 * \sim IR4 + \sim IR7 * IR6 * \sim IR4 + \sim IR7 * IR6 * \sim IR5) + W2 * (\sim IR7 * IR6 * IR5 * \sim IR4)$$

$$DRW = W1 * (A\sim IR6 * IR5 + \sim IR7 * \sim IR6 * IR4 + \sim IR7 * IR6 * \sim IR5 * \sim IR4) + W2 * (\sim IR7 * IR6 * \sim IR5 * IR4)$$

$$LDZ = W1 * (\sim IR6 * IR5 + \sim IR7 * \sim IR6 * IR4 + \sim IR7 * IR6 * \sim IR5 * \sim IR4)$$

$$LDC = W1 * (\sim IR7 * \sim IR6 * \sim IR5 * IR4 + \sim IR7 * IR6 * \sim IR5 * \sim IR4 + IR7 * \sim IR6 * IR5 * IR4)$$

$$M = W1 * (\sim IR7 * \sim IR6 * \sim IR5 * IR4 + \sim IR7 * IR6 * \sim IR5 * \sim IR4 + \sim IR7 * IR6 * IR5 * \sim IR4 + IR7 * \sim IR6 * \sim IR5 * IR4 + IR7 * \sim IR6 * IR5 * \sim IR4) + W2 * (\sim IR7 * IR6 * IR5 * \sim IR4)$$

$$LAR = LONG = W1 * (\sim IR7 * IR6 * \sim IR5 * IR4 + \sim IR7 * IR6 * IR5 * \sim IR4)$$

$$SHORT = W1 * (\sim IR6 * \sim IR5 + \sim IR7 * \sim IR6 + \sim IR7 * \sim IR5 * \sim IR4 + IR7 * IR5 * \sim IR4 + \sim C * \sim IR7 * IR6 * IR5 * IR4 + \sim Z(IR7 * \sim IR6 * \sim IR5 * \sim IR4)$$

$$MBUS = W2 * (\sim IR7 * IR6 * \sim IR5 * IR4)$$

$$MEMW = W2 * (\sim IR7 * IR6 * IR5 * \sim IR4)$$

$$PCADD = W1 * (C * \sim IR7 * IR6 * IR5 * IR4 + Z * IR7 * \sim IR6 * \sim IR5 * \sim IR4)$$

$$LPC = W1 * (IR7 * \sim IR6 * \sim IR5 * IR4)$$

$$STOP = W1 * (IR7 * IR6 * IR5 * \sim IR4)$$

3.3 模型机中断功能的探究

3.3.1 模型机中断功能说明

通过查阅资料，TEC-8 模型计算机中有一个简单的单机中断系统，只支持单级中断、单个中断请求，有中断屏蔽功能。

但是此次实验并未提供 IABUS，IAR、和 INTEN 等信号，并不能写出带中断功能的控制器，只能提供一个解决方案。

为了实现中断功能，需要新增三条用于中断的指令：IRET、DI、EI。

名称	汇编语言	功能	指令格式		
			IR7 IR6 IR5 IR4	IR3 IR2	IR1 IR0
中断返回	IRET	返回断点	1100	XX	XX
关中断	DI	禁止中断	1101	XX	XX
开中断	EI	允许中断	1111	X	XX

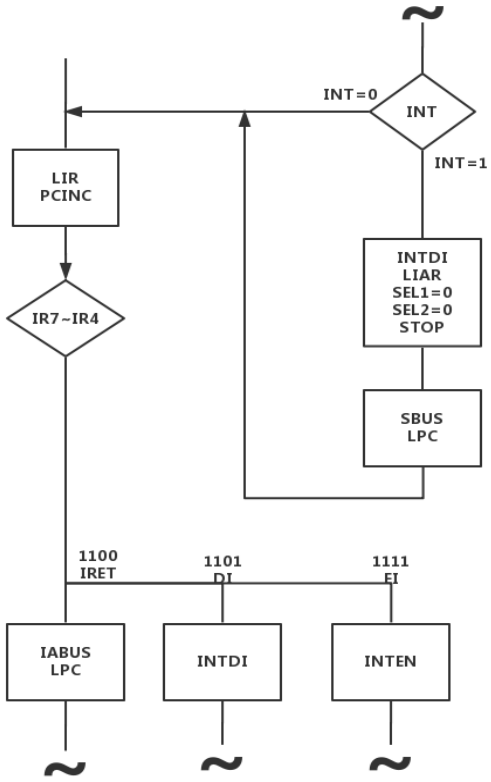
- DI 指令称作关中断指令。此条指令执行后，即使发生中断请求，TEC-8 也不响应中断请求。
- EI 指令称作开中断指令，此条指令执行后，TEC-8 响应中断。
- IRET 是中断返回指令，作为中断服务程序的最后一条指令执行。

3.3.2 模型机中断功能流程图

信号说明：

- IABUS：当它为 1 时，将中断地址寄存器中的地址送数据总线 DBUS。
- LIAR ：当它为 1 时，在 T3 的上升沿，将当前 PC 值写入中断地址寄存器 IAR。
- INTDI ：当它为 1 时，允许中断触发器（在时序发生器中）为 0，禁止 TEC-8 模型计算机响应中断请求。
- INTEN：当它为 1 时，允许中断触发器（在时序发生器中）为 1，允许 TEC-8 模型计算机响应中断请求。
- INT：判断是否执行中断的信号
- PULSE：按下 PULSE 按钮产生的高电平有效的中断请求脉冲信号。

流程图见下页：



3.2.3 模型机中断功能代码实现

使用 VHDL 对中断控制器进行实现：其中 EN_INT 为允许中断触发器，当它为 1 时，允许中断，当它为 0 时，禁止中断发生。复位脉冲 CLR#使 EN_INT 复位为 0。

```
INT_EN_P:   process(CLR#, MF, INTEN, INTDI, PULSE, EN_INT)
            begin
              if   CLR# = '0' then
                       EN_INT<= '0';
              elsif   MF'event and MF = '1' then
                       EN_INT <= INTEN or (EN_INT and (not INTDI) );
              end if;
              INT <= EN_INT and PULSE;
            end process;
```

3.2.4 模型机中断功能代码实现

检测：中断信号 INT 由 EI、DI 和 PULSE 信号决定

在每一条指令执行之后，会根据中断信号 INT 是否为 1 决定程序分支。如果 INT 为 1，则转到中断服务程序中进行中断处理，如果 INT 为 0，则继续执行下一条指令

执行：检测到中断信号 INT 后，转到中断服务程序前会产生 INTDI 信号，禁止新的中断发生，产生 LIAR 信号，将程序计数器 PC 的当前值保存在中断地址寄存器中，产生 STOP 信号，等待手动设置中断向量。在数据开关 SD7~SD0 上设置好中断地址后，机器将中断向量读到 PC 后，转到中断服务程序继续执行。

返回：执行指令 IRET，从中断地址返回。这条指令产生 IABUS 信号，将断点地址送到数据总线 DBUS,产生信号 LPC，将断点从数据总线装入 PC，恢复被中断程序。

4 设计与调试过程

4.1 实验日志

日期	主要工作
2020.8.10	① 回顾 VHDL 的语法、Quartus Ⅱ的基本操作； ② 进行课程设计相关资料的收集和汇总； ③ 小组内根据每个人擅长的部分进行任务划分，并讨论了控制器的设计思路； ④ 对小组希望最终产出的成果进行了确定，以及对接下来几天的工作安排进行规划。
2020.8.11	① 设计并整合了基本顺序控制器的指令架构，包括指令集、流程图、译码表和组合逻辑表达式等； ② 设计并新增了两条扩展功能指令：DEC、XOR，并将扩展指令的相关内容添加到基本顺序控制器的指令架构。
2020.8.12	① 对基本顺序控制器进行了代码实现； ② 采用卡诺图对代码进行了化简和优化，提高了运行速度；

	③ 在基本控制器的基础上设计了流水型硬连线控制器，包括指令集、流程图、译码表和组合逻辑表达式等； ④ 实现了任意指定程序入口，可以通过数据开关给定用户指定的程序入口地址（即修改 PC 指针的地址）。
2020.8.13	① 对流水型硬布线控制器进行了代码实现； ② 采用卡诺图对代码进行了化简和优化，提高了运行速度； ③ 对两个版本的代码进行了初步的调试，验证程序功能的正确性和鲁棒性。
2020.8.14	① 对小组在开发过程中积累的文档、日志进行汇总，开始撰写课程设计报告； ② 初步设计了若干个小测试程序，针对两个版本的控制器在功能的完整性、正确性、可靠性与健壮性上进行了较为全面的测试； ③ 对控制器的源代码进行了变量、格式、排版上的统一；
2020.8.15	① 小组讨论了针对中断问题的解决方法，提出了一个确实可行的解决方案，并设计相应流程图和代码； ② 设计了更加复杂的测试程序，针对两个版本的控制器在功能的完整性、正确性、可靠性与健壮性上进行了较为全面的测试，并修复了一些逻辑上的错误。 ③ 对控制器中所有潜在的问题进行了逐一修复和排查，迭代出最终版本的两个控制器； ④ 小组对整个课程设计的过程进行总结，每个小组成员抒发了自己的心得体会，并对整个开发过程和设计思路进行复盘。
2020.8.16	① 课程设计报告撰写工作完成。

4.2 调试及测试控制器功能

我们在老师提供的测试程序的基础上，添加了对自己设计的指令的一小段测试程序，合并成一个程序对控制器的功能进行测试。测试指令如下：

初值：R2=0FDH，R3=61H，【60H】=67H，【61H】=80H，【62H】=0FDH，【80H】=60H，【0FEH】=03H, 【0FFH】=03H

地址	程序指令	机器码
00H	XOR R2,R3	10101011
01H	XOR R2,R3	10101011
02H	XOR R2,R3	10101011
03H	DEC R2	10111000

04H	STP	11100000
05H	LD R0,[R2]	01010010
06H	INC R2	01001000
07H	LD R1,[R2]	01010110
08H	ADD R0,R1	00010001
09H	JC 01H	01110001
0AH	AND R1,R0	00110100
0BH	SUB R0,R2	00100010
0CH	INC R1	01000100
0DH	STA R0,[R1]	01100100
0EH	INC R3	01001100
0FH	JZ 02H	10000010
10H	LD R2,[R3]	01011011
11H	JMP [R2]	10011000
12H	INC R3	01001100
13H	INC R3	01001100
14H	SUB R0,R2	00100010
15H	LD R2,[R0]	01011000
16H	ADD R3,R2	00011110
17H	LD R3,[R3]	01011111
18H	STP	11100000

最终结果：R0=80H，R1=83H，R2=60H，R3=0FDH，【81H】=86H，【82H】=04H，【83H】=83H

其中：前三条指令是为了展示我们所设计的 XOR 指令的功能，通过三次 XOR 指令的异或操作，可以交换 R2 和 R3 寄存器中的值，之后我们测试了 DEC 指令，将 R2 中的内容减一。这样得到的结果与老师提供的测试程序的初值一致，即可紧接着进行后续的测试程序。

因为老师提供的测试程序较为复杂，在实验室进行结果正确性的探究较为困难，因此我们使用 c 语言实现了一个该测试程序的测试程序，如下所示：

```
#include<cstdio>
int M[1005];
int R0,R1,R2,R3;
void print(int x)
{
    printf("==============%X==============\n",x);
    printf("R0:%X  ",R0);
    printf("R1:%X  ",R1);
    printf("R2:%X  ",R2);
    printf("R3:%X\n",R3);
    for(int i=0x60;i<=0x62;i++) printf("[%X]=%X  ",i,M[i]);puts("");
```

```
12.     for(int i=0x80;i<=0x83;i++) printf("[%X]=%X  ",i,M[i]);puts("");
13.     for(int i=0xFD;i<=0xFF;i++) printf("[%X]=%X  ",i,M[i]);puts("");
14.     puts("");
15.     puts("");
16. }
17. int main()
18. {
19.     R0=0,R1=0;
20.     R2=0x60,R3=0xFD;
21.     M[0x60]=0x67;
22.     M[0x61]=0x80;
23.     M[0x62]=0xFD;
24.     M[0x80]=0x60;
25.     M[0xFE]=0x03;
26.     M[0xFF]=0x03;
27.     bool C=0,Z=0;
28.     L00:
29.         R0=M[R2];
30.         print(0x00);
31.     L01:
32.         R2++;
33.         if(R2>255) R2-=256,C=1,Z=1;
34.         else C=0,Z=0;
35.         print(0x01);
36.     L02:
37.         R1=M[R2];
38.         print(0x02);
39.     L03:
40.         R0+=R1;
41.         if(R0>255)
42.         {
43.             R0-=256;
44.             C=1;
45.         }
46.         else C=0;
47.         print(0x03);
48.     L04:
49.         if(C) goto L06;
50.     L05:
51.         R1&=R0;
52.         print(0x05);
53.     L06:
54.         R0-=R2;
```

```
55.         if(R0<0) R0+=256,C=1;
56.         else C=0;
57.         print(0x06);
58.     L07:
59.         R1++;
60.         if(R1>255) R1-=256,C=1,Z=1;
61.         else C=0,Z=0;
62.         print(0x07);
63.     L08:
64.         M[R1]=R0;
65.         print(0x08);
66.     L09:
67.         R3++;
68.         if(R3>255) R3-=256,C=1,Z=1;
69.         else Z=0,C=0;
70.         print(0x09);
71.     L0A:
72.         if(Z) goto L0D;
73.     L0B:
74.         R2=M[R3];
75.         print(0x0B);
76.     L0C:
77.         if(R2==1) goto L01;
78.         if(R2==2) goto L02;
79.         if(R2==3) goto L03;
80.         if(R2==4) goto L04;
81.         if(R2==5) goto L05;
82.         if(R2==6) goto L06;
83.         if(R2==7) goto L07;
84.         if(R2==8) goto L08;
85.         if(R2==9) goto L09;
86.         if(R2==10) goto L0A;
87.         if(R2==11) goto L0B;
88.     L0D:
89.         R3++;
90.         print(0x0D);
91.     L0E:
92.         R3++;
93.         print(0x0E);
94.     L0F:
95.         R0-=R2;
96.         print(0x0F);
97.     L10:
```

```
98.         R2=M[R0];
99.         print(0x10);
100.    L11:
101.        R3+=R2;
102.        print(0x11);
103.    L12:
104.        R3=M[R3];
105.        print(0x12);
106.    }
```

我们通过 int 型变量来模拟寄存器，数组来模拟内存，布尔型变量 C,Z 表示进位、相等等结果。其中，条件跳转类指令通过 goto 语句进行实现。该测试程序相比手工计算高效地计算出结果，通过将结果与我们的控制器得出的结果进行比对，二者一致。综上可知，本次小组设计的控制器在整机功能的完备性与正确性方面是非常成功的。

5 实验总结与心得

5.1 实验中遇到的问题

① 由于习惯了软件开发语言 C++，python 等，对 VHDL 代码内部的执行顺序不理解，不明白并行是如何实现的，对于代码编写方面有误解。

【解决方案】：构造体中的语句与子模块之间是并行处理的。子模块内部（如 process）是顺序处理的。顺序处理的语句，前者会对后者产生影响，但处理机器指令时，各信号都是严格按照时钟节拍改变的，因此不会产生冲突。

② 在编写代码的过程中，如何尽可能避免由于看错/写错而带来的不应该的错误？

【解决方案】：对于卡诺图优化后得到的新表达式，每次选中一行高亮来保证不会看错行，对于 VHDL 的代码，每行只对应原公式的一项，这样也有利于之后的比对检查过程，尽可能避免了低级错误。

③ 流水硬连线控制器中跳转指令以及存取数据指令能否在一个节拍中完成？

【解决方案】：不能。在跳转指令中，PCADD 和 PCINC 指令会冲突，需在不同节拍中运行。二 LD 和 ST 指令 LAR 和 DRW、MEMW 写入不同的数据，不能在同一周期。

④ 流水硬连线控制器是怎么让程序进入流水开始工作的？

【解决方案】：在执行 CLR 且 SWCBA=000 时，IR 寄存器的值为全 0，执行 NOP 指令，NOP 指

令执行完开始执行程序中的第一条指令进入流水模式开始工作。

⑤ 在实验室调试的过程中，我们发现无论怎么修改和编译源程序，烧录进 TEC-8 中的程序始终没有改变。

【解决方案】：最终我们发现，我们烧录进 TEC-8 的 pof 文件的修改日期始终是 8 月 17 日，并没有随我们的编译而改变。查阅资料发现，只有完整版的 quartus II 才能下载编译后的 pof 文件，而未破解的版本不能，因此我们使用了完整版的 quartus II 进行开发，终于解决了问题。

⑥ 在测试流水型控制器时，我们发现在执行 SUB 指令之后，PIR 会自动跳过下一条指令，而将下下条指令打入 IR（并非偶发现象，我们重复了多次实验，都遇到了类似问题，并且同时间做实验的其他小组也遇到了相同的问题）

【解决方案】：最终我们更换了一台 TEC-8 机器，则不再出现类似情况。我们分析该 BUG 可能与 TEC-8 不同型号的内部实现有关。

5.2 控制器的优点与缺点

5.2.1 优点：

① 指令系统完备

我们设计的控制器不仅包含了所有常见的指令，还在原有的指令集上扩展了一部分指令集，如 DEC、XOR、NOP。这些指令的实现使得我们的控制器能够满足一些简单的程序的运行要求。

② 扩展功能的实现

除了课程设计要求的基础功能，我们还在设计的控制器的基础上对功能进行了扩展，如实现了设定 PC 的值，即可以从任意位置开始执行程序。因为在基本设计中，程序只能从 00H 开始执行，这样的控制器不具备灵活性，且为后续的调试带来非常大的困难。该功能的实现，为调试工作带来了极大便利，也使控制器更具灵活性。

③ 更良好的性能

我们设计的控制器在两个方面提升了整体的性能：在基本顺序控制器的前提上，实现了流水型的控制器，且加速比接近 2；利用卡诺图对信号组合逻辑表达式进行化简，优化为不同信号构成的与或非逻辑表达式。这样一来，执行大量指令时程序的运行速度会有大幅提升，仅仅受限于与或非门的时延和主时钟频率，而不受其他因素的干扰。

5.2.2 缺点：

① 指令集规模不够庞大

虽然我们实现了一些基本的指令集，但是受限于 IR 的位数，指令集的上限也只能有 16 条指令，这对指令集的规模具有非常大的限制。因此只能实现一些基本的运算操作，对于复杂的程序来说实现有一定的难度。

② 流水功能不完善

在我们实现的流水型硬布线控制器中，实际上只实现了取指周期和执行周期的流水，对于流水问题中的其他子问题并没有讨论，比如：各段之间互通信息、互相等待的问题；各段工作时序协调一致的问题；写后读、读后写等问题。如果有更多时间，我们希望将这些遗漏进行修复与完善。

③ 缺少中断功能

经过我们小组的讨论，我们认为本此课程设计提供的资源很难实现中断功能，因此我们根据我们认为还需要补充的条件对中断功能进行了设计，但并没有在程序中去实现它，此外我们还将原先指令集中的三条与中断相关的指令替换为其他指令。

5.3 小组成员实验心得

5.3.1 史嘉程：

本次课程设计的过程中，我们小组三名同学共同完成了两个版本的硬布线控制器，包括基本顺序控制器和流水型控制器。本次设计的理论基础都来自于大二上学期所学的《数字逻辑》与本学期所学的《计算机组成原理》，而包括 TEC-8 的数据通路、VHDL 语言的学习也在以往的实验课上有所涉猎，因此在前期的理论准备上并没有遇到特别大的问题。然而，最让我们小组头疼的，是将所掌握的理论知识落实到实际应用之上，以及一些细节的实现。但是经过整个小组的团结一致和数天的艰苦努力，我们还是做出了令自己满意的成果。接下来我想主要谈一谈我在本次课程设计中的一些心得体会。

首先，我认为本次课程设计是对我项目开发能力的极大的提升。一个项目的设计与开发，与平常我们做练习题，根据一个题目得到一个结果的“面向题目编程”完全不一样。项目的设计与开发需要设计者的大局观，框架的设计更是如此。如果在项目的开发初期不具备大局观，则到开发中期，可能会遇到很多新的问题，而需要重新回到一开始从头进行修改。

项目开发离不开团队内部的配合与协调。在实际的公司或者实验室中，大型的项目都会由多人协作完成，更离不开小组内部的分工和调度。我们小组在本次课程设计的最开始就合理的划分了每个人的任务模块，虽然因为疫情大家不在一起，每个人的空闲时间并不重合，但我们的分工合理的利用了大家的时间，提高了项目的开发效率。在项目开发过程中，我们还根据当前遇到的一些情况进行临时的调整，以保持项目开发保持在正确的轨道上。

其次，习惯于软件开发的我，深刻地体会到软件开发与硬件开发的区别。平常习惯于 C++、Python、Java 这类语言的我，在学习 VHDL 以及使用 Quartus II 的过程中都遇到了相当大的阻碍。比如软件开发中，除非设置多线程，其他时候代码的实现都是从上至下的，而硬件语言如 VHDL，在编程时需要考虑 Process 代码块带来的并行的影响，从而产生一些自己意想不到的 BUG。

硬件开发的调试也给我带来了不小的难题。在软件开发中，我习惯于“COUT 大法”或者在 IDE 中设置断点来进行调试，而硬件开发则很难做到这些点。对于硬件的调试需要超乎平常的耐心，因为相关的经验太少，很难根据经验一下判断出问题在哪。

此外，在许多软件开发语言中，我们有宏定义来进行全局的操作，使得程序的修改容易很多，但是硬件开发过程中则并非如此，修改一个小地方需要在程序中改动非常多地方，也为开发带来了很多麻烦。

总的来说，本次的课程设计过程中我收获颇丰。并且我深刻的感受到实际的编程、开发对于知识的理解和渗透有多么巨大的帮助：他锻炼了我的动手能力、项目管理能力、团队协作能力、将想法落地的能力，还对我理解 CPU 的构造、指令的实现有非常大的推动作用。我相信本次课程设计能为我今后的专业学习以及实际工程开发道路带来启发。

5.3.2 刘俊：

通过此次实验，我们完成了 CPU 控制器的编写。经过一学期对计算机组成原理的理论学习，我对计算机组成部分有了较全面的了解，而这次控制器编写的实验，让我更深入的了解了控制器的工作过程和硬件编程的相关知识。

由于疫情，我们不得不在网上进行合作，为实验增加了额外的难度。而且由于不能返校，控制器只能通过仿真波形来检测正确性，调试难度增加。但是经过小组成员的不懈努力，最终解决了这些问题。

通过这次课设，我收获颇丰。首先，这次实验让我更加了解硬件编程。做硬件要对硬件足够熟悉，不然实践中会被各种信号卡壳。开始实验时，便重点关注了每个信号的具体含义，它们的作用

和触发条件。由于上计算机组成原理课程时使用过 TEC8，对数据通路和信号灯比较了解，大部分信号都能在指导书中找到含义，但在实验指导书提供的流程图中，出现了 SST0,ST0 信号，经过研究，这是在读写寄存器时用到的中间信号，简化了流程。了解了信号后，便开始设计我们自己的流程图，增加了 XOR 和 DEC 指令，进行到这一步还比较轻松。为了加快控制器的运行速度提高运行效率，我们决定用卡诺图对产生的逻辑表达式进行化简，开始时是手动进行化简，发现 4 阶卡诺图化简工作量太大，于是通过不断寻找方法，找到了一个自动化简卡诺图的工具软件，不仅减少了工作量，还使得化简结果更加可靠。这也告诉我们要善于信息的收集，使用更多方便的软件提高效率。

同时这次课程设计让我对理论知识有了更加深刻的理解。这门课设的基础课程是“数字逻辑”、“计算机组成原理”，在理论学习过程中，有很多类似“这样设计，这样做有什么用”的问题。在这次实验里，我对课本知识有了更深入的理解，一定程度上解决了这些问题，比如数字逻辑里学的卡诺图化简，能减少程序的选择分支，加快运行速度。以及学习了流水 CPU 原理后，也疑惑流水到底是怎么具体实现的，这次实验中，我们将取指指令放到每条指令的最后，使大部分指令都变为单周期指令，能实现 2 级流水，我也更加深刻的了解了流水的工作过程。

总而言之，这次课程设计带给我的收获是巨大的，不仅让我补全了平时学习上的知识漏洞，加深了我对计算机组成原理的理解，更让我学会了如何进行团队合作，同时也提升了自身合作能力和解决问题的能力，是一次很有意义的实践。

5.3.3 范珈诚：

高中时期以及大学前两年的学习，接触到的大多数都是软件相关的知识与开发工作，先前对于“数字逻辑”，“计算机组成原理”两门课程的了解微乎其微，对计算机硬件也不甚了解。通过对这两门课程的学习，让我逐渐对计算机硬件的有关知识有了初步的了解，对支撑整机的控制，软件运行的底层架构有了初步的认识。而这次课程设计，也算是对于这两门课程相关知识的一个综合性大考察。通过本次课程设计，我收获了包括知识本身与知识之外的很多。

对于知识的本身，本次课程设计让我重温了数字逻辑与计算机组成原理两门课程的核心知识点，例如数字逻辑表达式，卡诺图，VHDL 语言，还有指令系统，计算机整机结构，流水线原理等，这些知识点自始至终贯彻在我们的课程设计过程当中，是我们重要的知识依托。

对于程序设计方面，在此之前，我接触过的语言大多是 C/C++，python 语言等高级语言，对于 VHDL 这类硬件语言，以及其中的并行串行相关的说法几乎一窍不通，仅仅停留在数字逻辑课程所

接触的那些知识。在此次课程设计过程中，接手代码编写工作对于我而言是一个挑战，但也是一次历练的机会。考虑到大量的 CASE 语句会对效率产生影响，我们想到用卡诺图来消除对 CASE 语句的依赖，但与此同时，我们不得不面对卡诺图化简后繁杂的公式，在编写代码的过程中，为了尽可能避免低级错误的产生，我们想出了各种办法，代码编写完成后，又经过了多轮的比对检查，来确保转化的准确无误。每一个小错误都可能导致整个系统的崩溃，因此这个过程非常需要耐心与细心。

对于团队协作方面，由于疫情的影响，在课程设计的初期，小组内三名成员无法聚集在一起，而是在三个不同的城市进行线上合作，尽管合作效果会比三个人在一起稍差，但这次课程设计还是让我感受到多人协作的力量远比单人要强得多，同时提高了我的协作能力，对于今后要面临的更多的协作任务也会有很大帮助。

总的来说，这次课程设计给了我巨大的收获。它补足了我知识的不足，加深了我对课程内容的理解，提高了我的实践能力与团队协作能力，对我今后的学习有很大的指导意义。

6 控制器源代码

因为控制器的完整源代码较长，因此放在报告的最后进行展示。

6.1 基本顺序硬布线控制器代码

```
1.  LIBRARY IEEE;
2.  USE IEEE.STD_LOGIC_1164.all;
3.
4.  ENTITY cpu IS
5.  PORT (
6.        SWC : IN STD_LOGIC; --控制台模式
7.        SWB : IN STD_LOGIC;
8.        SWA : IN STD_LOGIC;
9.        CLR : IN STD_LOGIC; -- 复位
10.       T3 : IN STD_LOGIC;
11.       W3 : IN STD_LOGIC; --W3 节拍
12.       W2 : IN STD_LOGIC; --W2 节拍
13.       W1 : IN STD_LOGIC; --W1 节拍
14.       IR : IN STD_LOGIC_VECTOR(7 DOWNTO 4);
15.       C : IN STD_LOGIC;
16.       Z : IN STD_LOGIC;
17.       S : OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --运算类型
18.       M : OUT STD_LOGIC;
19.       CIN : OUT STD_LOGIC; --进位
20.       SEL3 : OUT STD_LOGIC;
```

```
21.       SEL2 : OUT STD_LOGIC; -- ALU A
22.       SEL1 : OUT STD_LOGIC;
23.       SEL0 : OUT STD_LOGIC; -- ALU B
24.       SELCTL : OUT STD_LOGIC;
25.       LIR : OUT STD_LOGIC; --DBUS 打入 IR
26.       LDC : OUT STD_LOGIC; --在 T3 上升沿,保存进位
27.       LDZ : OUT STD_LOGIC; --在 T3 上升沿,保存结果为 0 标志
28.       LAR : OUT STD_LOGIC; --在 T3 上升沿,将 DBUS 打入 AR
29.       LPC : OUT STD_LOGIC; --在 T3 上升沿,将 DBUS 打入 PC
30.       PCINC : OUT STD_LOGIC; --在 T3 上升沿,将 PC 自增
31.       PCADD : OUT STD_LOGIC; --PC 加偏移量
32.       ARINC : OUT STD_LOGIC; --在 T3 上升沿,将 IR 自增
33.       LONG : OUT STD_LOGIC; --长指令(W1 W2 W3)
34.       SHORT : OUT STD_LOGIC; --短指令(W1)
35.       ABUS : OUT STD_LOGIC; --运算器结果打入总线
36.       MBUS : OUT STD_LOGIC; --双端口 RAM 左端打入总线
37.       SBUS : OUT STD_LOGIC; --S 打入总线
38.       DRW : OUT STD_LOGIC; --在 T3 上升沿,将总线数据写入 SEL3SEL2 选中的寄存器
39.       MEMW : OUT STD_LOGIC; --为 1 时,将总线数据写入 AR 指向的存储单元,为 0 时读存储器
40.       STOP : OUT STD_LOGIC --暂停信号
41.       );
42. END cpu ;
43.
44. ARCHITECTURE logic of cpu is
45.    SIGNAL ST0 : STD_LOGIC := '0';
46.    SIGNAL SST0 : STD_LOGIC := '0';
47.    SIGNAL SW : STD_LOGIC_VECTOR (2 DOWNTO 0);
48. BEGIN
49.     SW <= SWC & SWB & SWA;
50.     PROCESS(SST0, T3, SW, CLR, IR, ST0, W1, W2, W3, C, Z)
51.     BEGIN
52.         IF (CLR = '0') THEN
53.             ST0 <= '0';
54.         ELSIF (T3'EVENT AND T3 = '0') THEN
55.             IF (ST0 = '1' AND W2 = '1' AND SW = "100") THEN
56.                 ST0 <= '0';
57.             END IF;
58.             IF (SST0 = '1') THEN
59.                 ST0 <= '1';
60.             END IF;
61.         END IF;
62.         S <= "0000";
63.         M <= '0';
```

```
64.         CIN <= '0';
65.         SEL3 <= '0';
66.         SEL2 <= '0';
67.         SEL1 <= '0';
68.         SEL0 <= '0';
69.         SELCTL <= '0';
70.         LIR <= '0';
71.         LDC <= '0';
72.         LDZ <= '0';
73.         LPC <= '0';
74.         LAR <= '0';
75.         PCINC <= '0';
76.         PCADD <= '0';
77.         ARINC <= '0';
78.         LONG <= '0';
79.         SHORT <= '0';
80.         ABUS <= '0';
81.         MBUS <= '0';
82.         SBUS <= '0';
83.         DRW <= '0';
84.         MEMW <= '0';
85.         STOP <= '0';
86.         SST0 <= '0';
87.
88.         CASE SW IS
89.             WHEN "100" =>
90.                 SBUS <= '1';
91.                 SEL3 <= ST0;
92.                 SEL2 <= W2;
93.                 SEL1 <= (NOT ST0 AND W1) OR (ST0 AND W2);
94.                 SEL0 <= W1;
95.                 SELCTL <= '1';
96.                 DRW <= '1';
97.                 STOP <= '1';
98.                 SST0 <= NOT ST0 AND W2;
99.             WHEN "011" =>
100.                 SEL3 <= W2;
101.                 SEL2 <= '0';
102.                 SEL1 <= W2;
103.                 SEL0 <= '1';
104.                 SELCTL <= '1';
105.                 STOP <= '1';
106.             WHEN "010" =>
```

```
107.                 SBUS <= NOT ST0 AND W1;
108.                 LAR <= NOT ST0 AND W1;
109.                 STOP <= W1;
110.                 SHORT <= W1;
111.                 SELCTL <= W1;
112.                 MBUS <= ST0 AND W1;
113.                 ARINC <= ST0 AND W1;
114.                 SST0 <= NOT ST0 AND W1;
115.             WHEN "001" =>
116.                 SBUS <= W1;
117.                 LAR <= NOT ST0 AND W1;
118.                 STOP <= W1;
119.                 SHORT <= W1;
120.                 SELCTL <= W1;
121.                 MEMW <= ST0 AND W1;
122.                 ARINC <= ST0 AND W1;
123.                 SST0 <= NOT ST0 AND W1;
124.             WHEN "000" =>
125.                 IF (ST0 = '0') THEN
126.                     SBUS <= W1;
127.                     SHORT <= W1;
128.                     SST0 <= W1;
129.                     LPC <= W1;
130.                     STOP <= W1;
131.                 ELSIF (ST0 = '1') THEN
132.                     LIR <= W1;
133.                     PCINC <= W1;
134.                     S(3) <= (W2 AND ((NOT IR(6) AND IR(4)) OR
135.                                      (NOT IR(7) AND NOT IR(5) AND IR(4)) OR
136.                                      (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4))))
    OR
137.                              (W3 AND (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4)));
138.
139.                     S(2) <= W2 AND ((NOT IR(6) AND IR(5) AND NOT IR(4)) OR
140.                                     (NOT IR(7) AND IR(5) AND NOT IR(4)) OR
141.                                     (IR(7) AND NOT IR(6) AND IR(4)) OR
142.                                     (IR(7) AND NOT IR(6) AND IR(5)));
143.
144.                     S(1) <= (W2 AND ((NOT IR(6) AND IR(5)) OR
145.                                     (NOT IR(7) AND IR(5) AND NOT IR(4)) OR
146.                                     (IR(7) AND NOT IR(6) AND IR(4)) OR
```

```
147.                                    (NOT IR(7) AND IR(6) AND NOT IR(5) AND IR(4))))
    OR
148.                               (W3 AND (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4)));
149.
150.                     S(0) <= W2 AND ((NOT IR(6) AND IR(4)) OR (NOT IR(7) AND IR(6) A
    ND IR(5) AND NOT IR(4)));
151.
152.                     CIN <= W2 AND ((NOT IR(7) AND NOT IR(6) AND NOT IR(5) AND IR(4)
    ) OR
153.                                    (IR(7) AND NOT IR(6) AND IR(5) AND IR(4)));
154.
155.                     ABUS <= (W2 AND ((NOT IR(6) AND IR(4)) OR
156.                                    (NOT IR(6) AND IR(5)) OR
157.                                    (NOT IR(7) AND NOT IR(5) AND IR(4)) OR
158.                                    (NOT IR(7) AND IR(5) AND NOT IR(4)) OR
159.                                    (NOT IR(7) AND IR(6) AND NOT IR(4)) OR
160.                                    (NOT IR(7) AND IR(6) AND NOT IR(5)))) OR
161.                               (W3 AND (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4)));
162.
163.                     DRW <= (W2 AND ((NOT IR(6) AND IR(5)) OR
164.                                    (NOT IR(7) AND NOT IR(6) AND IR(4)) OR
165.                                    (NOT IR(7) AND IR(6) AND NOT IR(5) AND NOT IR(4
    )))) OR
166.                               (W3 AND (NOT IR(7) AND IR(6) AND NOT IR(5) AND IR(4)));
167.
168.                     LDZ <= W2 AND ((NOT IR(6) AND IR(5)) OR
169.                                    (NOT IR(7) AND NOT IR(6) AND IR(4)) OR
170.                                    (NOT IR(7) AND IR(6) AND NOT IR(5) AND NOT IR(4
    )));
171.
172.                     LDC <= W2 AND ((NOT IR(7) AND NOT IR(6) AND NOT IR(5) AND IR(4)
    ) OR
173.                                    (NOT IR(7) AND NOT IR(6) AND IR(5) AND NOT IR(4
    )) OR
174.                                    (NOT IR(7) AND IR(6) AND NOT IR(5) AND NOT IR(4
    )) OR
175.                                    (IR(7) AND NOT IR(6) AND IR(5) AND IR(4)));
176.
177.                     M <= (W2 AND ((NOT IR(7) AND NOT IR(6) AND IR(5) AND IR(4)) OR
```

```
                                    (NOT IR(7) AND IR(6) AND NOT IR(5) AND IR(4)) O
R
                                    (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4)) O
R
                                    (IR(7) AND NOT IR(6) AND NOT IR(5) AND IR(4)) O
R
                                    (IR(7) AND NOT IR(6) AND IR(5) AND NOT IR(4))))
 OR
                              (W3 AND (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4)));

                    LAR <= W2 AND ((NOT IR(7) AND IR(6) AND NOT IR(5) AND IR(4)) OR

                                    (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4)));

                    LONG <= W2 AND ((NOT IR(7) AND IR(6) AND NOT IR(5) AND IR(4)) O
R
                                    (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4)));

                    MBUS <= W3 AND (NOT IR(7) AND IR(6) AND NOT IR(5) AND IR(4));

                    MEMW <= W3 AND (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4));

                    PCADD <= W2 AND ((C AND NOT IR(7) AND IR(6) AND IR(5) AND IR(4)
) OR
                                    (Z AND IR(7) AND NOT IR(6) AND NOT IR(5) AND NO
T IR(4)));

                    LPC <= W2 AND (IR(7) AND NOT IR(6) AND NOT IR(5) AND IR(4));

                    STOP <= W2 AND (IR(7) AND IR(6) AND IR(5) AND NOT IR(4));
                END IF;
            WHEN OTHERS => NULL;
        END CASE;
    END PROCESS;
  END logic;

```

6.2 实现流水线的硬布线控制器

```
1.  LIBRARY IEEE;
2.  USE IEEE.STD_LOGIC_1164.all;
3.
4.  ENTITY op_cpu IS
5.  PORT (
6.        SWC : IN STD_LOGIC; --控制台模式
7.        SWB : IN STD_LOGIC;
8.        SWA : IN STD_LOGIC;
9.        CLR : IN STD_LOGIC; -- 复位
10.       T3 : IN STD_LOGIC;
11.       W3 : IN STD_LOGIC; --W3 节拍
12.       W2 : IN STD_LOGIC; --W2 节拍
13.       W1 : IN STD_LOGIC; --W1 节拍
14.       IR : IN STD_LOGIC_VECTOR(7 DOWNTO 4);
15.       C : IN STD_LOGIC;
16.       Z : IN STD_LOGIC;
17.       S : OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --运算类型
18.       M : OUT STD_LOGIC;
19.       CIN : OUT STD_LOGIC; --进位
20.       SEL3 : OUT STD_LOGIC;
21.       SEL2 : OUT STD_LOGIC; -- ALU A
22.       SEL1 : OUT STD_LOGIC;
23.       SEL0 : OUT STD_LOGIC; -- ALU B
24.       SELCTL : OUT STD_LOGIC;
25.       LIR : OUT STD_LOGIC; --DBUS 打入 IR
26.       LDC : OUT STD_LOGIC; --在 T3 上升沿,保存进位
27.       LDZ : OUT STD_LOGIC; --在 T3 上升沿,保存结果为 0 标志
28.       LAR : OUT STD_LOGIC; --在 T3 上升沿,将 DBUS 打入 AR
29.       LPC : OUT STD_LOGIC; --在 T3 上升沿,将 DBUS 打入 PC
30.       PCINC : OUT STD_LOGIC; --在 T3 上升沿,将 PC 自增
31.       PCADD : OUT STD_LOGIC; --PC 加偏移量
32.       ARINC : OUT STD_LOGIC; --在 T3 上升沿,将 IR 自增
33.       LONG : OUT STD_LOGIC; --长指令(W1 W2 W3)
34.       SHORT : OUT STD_LOGIC; --短指令(W1)
35.       ABUS : OUT STD_LOGIC; --运算器结果打入总线
36.       MBUS : OUT STD_LOGIC; --双端口 RAM 左端打入总线
37.       SBUS : OUT STD_LOGIC; --S 打入总线
38.       DRW : OUT STD_LOGIC; --在 T3 上升沿,将总线数据写入 SEL3SEL2 选中的寄存器
39.       MEMW : OUT STD_LOGIC; --为 1 时,将总线数据写入 AR 指向的存储单元,为 0 时读存储器
40.       STOP : OUT STD_LOGIC --暂停信号
```

```
41.       );
42. END op_cpu ;
43.
44. ARCHITECTURE logic of op_cpu is
45.    SIGNAL ST0 : STD_LOGIC := '0';
46.    SIGNAL SST0 : STD_LOGIC := '0';
47.    SIGNAL SW : STD_LOGIC_VECTOR (2 DOWNTO 0);
48. BEGIN
49.     SW <= SWC & SWB & SWA;
50.     PROCESS(SST0, T3, SW, CLR)
51.     BEGIN
52.         IF (CLR = '0') THEN
53.             ST0 <= '0';
54.         ELSIF (T3'EVENT AND T3 = '0') THEN
55.             IF (SST0 = '1') THEN
56.                 ST0 <= '1';
57.             END IF;
58.             IF (ST0 = '1' AND W2 = '1' AND SW = "100") THEN
59.                 ST0 <= '0';
60.             END IF;
61.         END IF;
62.     END PROCESS;
63.     PROCESS(IR, ST0, W1, W2, W3, SW, C, Z)
64.     BEGIN
65.         S <= "0000";
66.         M <= '0';
67.         CIN <= '0';
68.         SEL3 <= '0';
69.         SEL2 <= '0';
70.         SEL1 <= '0';
71.         SEL0 <= '0';
72.         SELCTL <= '0';
73.         LIR <= '0';
74.         LDC <= '0';
75.         LDZ <= '0';
76.         LPC <= '0';
77.         LAR <= '0';
78.         PCINC <= '0';
79.         PCADD <= '0';
80.         ARINC <= '0';
81.         LONG <= '0';
82.         SHORT <= '0';
83.         ABUS <= '0';
```

```
84.         MBUS <= '0';
85.         SBUS <= '0';
86.         DRW <= '0';
87.         MEMW <= '0';
88.         STOP <= '0';
89.         SST0 <= '0';
90.
91.         CASE SW IS
92.             WHEN "100" =>
93.                 SBUS <= '1';
94.                 SEL3 <= ST0;
95.                 SEL2 <= W2;
96.                 SEL1 <= (NOT ST0 AND W1) OR (ST0 AND W2);
97.                 SEL0 <= W1;
98.                 SELCTL <= '1';
99.                 DRW <= '1';
100.                 STOP <= '1';
101.                 SST0 <= NOT ST0 AND W2;
102.             WHEN "011" =>
103.                 SEL3 <= W2;
104.                 SEL2 <= '0';
105.                 SEL1 <= W2;
106.                 SEL0 <= '1';
107.                 SELCTL <= '1';
108.                 STOP <= '1';
109.             WHEN "010" =>
110.                 SBUS <= NOT ST0 AND W1;
111.                 LAR <= NOT ST0 AND W1;
112.                 SST0 <= NOT ST0 AND W1;
113.                 STOP <= W1;
114.                 SHORT <= W1;
115.                 SELCTL <= W1;
116.                 MBUS <= ST0 AND W1;
117.                 ARINC <= ST0 AND W1;
118.             WHEN "001" =>
119.                 SBUS <= W1;
120.                 LAR <= NOT ST0 AND W1;
121.                 SST0 <= NOT ST0 AND W1;
122.                 STOP <= W1;
123.                 SHORT <= W1;
124.                 SELCTL <= W1;
125.                 MEMW <= ST0 AND W1;
126.                 ARINC <= ST0 AND W1;
```

```
127.             WHEN "000" =>
128.                 IF (ST0 = '0') THEN
129.                     SBUS <= W1;
130.                     LPC <= W1;
131.                     SHORT <= W1;
132.                     SST0 <= W1;
133.                 ELSIF (ST0 = '1') THEN
134.                     LIR <= (W1 AND ((NOT IR(6) AND IR(5)) OR
135.                                     (NOT IR(7) AND NOT IR(6)) OR
136.                                     (NOT IR(7) AND NOT IR(5) AND NOT IR(4)) OR
137.                                     (NOT C AND NOT IR(7) AND IR(6) AND IR(5) AND IR
    (4)) OR
138.                                     (NOT Z AND IR(7) AND NOT IR(6) AND NOT IR(5) AN
    D NOT IR(4)))) OR
139.                                 (W2 AND ((NOT IR(7) AND IR(6) AND NOT IR(5) AND IR(4))
    OR
140.                                     (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4)) O
    R
141.                                     (IR(7) AND NOT IR(6) AND NOT IR(5) AND IR(4)) O
    R
142.                                     (C AND NOT IR(7) AND IR(6) AND IR(5) AND IR(4))
     OR
143.                                     (Z AND IR(7) AND NOT IR(6) AND NOT IR(5) AND NO
    T IR(4))));
144.
145.                     PCINC <= (W1 AND ((NOT IR(6) AND IR(5)) OR
146.                                     (NOT IR(7) AND NOT IR(6)) OR
147.                                     (NOT IR(7) AND NOT IR(5) AND NOT IR(4)) OR
148.                                     (NOT C AND NOT IR(7) AND IR(6) AND IR(5) AND IR
    (4)) OR
149.                                     (NOT Z AND IR(7) AND NOT IR(6) AND NOT IR(5) AN
    D NOT IR(4)))) OR
150.                                 (W2 AND ((NOT IR(7) AND IR(6) AND NOT IR(5) AND IR(4))
    OR
151.                                     (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4)) O
    R
152.                                     (IR(7) AND NOT IR(6) AND NOT IR(5) AND IR(4)) O
    R
153.                                     (C AND NOT IR(7) AND IR(6) AND IR(5) AND IR(4))
     OR
154.                                     (Z AND IR(7) AND NOT IR(6) AND NOT IR(5) AND NO
    T IR(4))));
155.
```

```
156.                    S(3) <= (W1 AND ((NOT IR(6) AND IR(4)) OR
157.                                    (NOT IR(7) AND NOT IR(5) AND IR(4)) OR
158.                                    (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4))))
    OR
159.                            (W2 AND (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4)));

160.
161.                    S(2) <= W1 AND ((NOT IR(6) AND IR(5) AND NOT IR(4)) OR
162.                                    (NOT IR(7) AND IR(5) AND NOT IR(4)) OR
163.                                    (IR(7) AND NOT IR(6) AND IR(4)) OR
164.                                    (IR(7) AND NOT IR(6) AND IR(5)));
165.
166.                    S(1) <= (W1 AND ((NOT IR(6) AND IR(5)) OR
167.                                    (NOT IR(7) AND IR(5) AND NOT IR(4)) OR
168.                                    (IR(7) AND NOT IR(6) AND IR(4)) OR
169.                                    (NOT IR(7) AND IR(6) AND NOT IR(5) AND IR(4))))
    OR
170.                            (W2 AND (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4)));

171.
172.                    S(0) <= W1 AND ((NOT IR(6) AND IR(4)) OR
173.                                    (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4)));

174.
175.                    CIN <= W1 AND ((NOT IR(7) AND NOT IR(6) AND NOT IR(5) AND IR(4)
    ) OR
176.                                    (IR(7) AND NOT IR(6) AND IR(5) AND IR(4)));
177.
178.                    ABUS <= (W1 AND ((NOT IR(6) AND IR(4)) OR
179.                                    (NOT IR(6) AND IR(5)) OR
180.                                    (NOT IR(7) AND NOT IR(5) AND IR(4)) OR
181.                                    (NOT IR(7) AND IR(5) AND NOT IR(4)) OR
182.                                    (NOT IR(7) AND IR(6) AND NOT IR(4)) OR
183.                                    (NOT IR(7) AND IR(6) AND NOT IR(5)))) OR
184.                            (W2 AND (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4)));

185.
186.                    DRW <= (W1 AND ((NOT IR(6) AND IR(5)) OR
187.                                    (NOT IR(7) AND NOT IR(6) AND IR(4)) OR
188.                                    (NOT IR(7) AND IR(6) AND NOT IR(5) AND NOT IR(4
    )))) OR
189.                            (W2 AND (NOT IR(7) AND IR(6) AND NOT IR(5) AND IR(4)));
```

```
190.
191.                     LDZ <= W1 AND ((NOT IR(6) AND IR(5)) OR
192.                                    (NOT IR(7) AND NOT IR(6) AND IR(4)) OR
193.                                    (NOT IR(7) AND IR(6) AND NOT IR(5) AND NOT IR(4
    )));
194.
195.                     LDC <= W1 AND ((NOT IR(7) AND NOT IR(6) AND NOT IR(5) AND IR(4)
    ) OR
196.                                    (NOT IR(7) AND NOT IR(6) AND IR(5) AND NOT IR(4
    )) OR
197.                                    (NOT IR(7) AND IR(6) AND NOT IR(5) AND NOT IR(4
    )) OR
198.                                    (IR(7) AND NOT IR(6) AND IR(5) AND IR(4)));
199.
200.                     M <= (W1 AND ((NOT IR(7) AND NOT IR(6) AND IR(5) AND IR(4)) OR

201.                                    (NOT IR(7) AND IR(6) AND NOT IR(5) AND IR(4)) O
    R
202.                                    (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4)) O
    R
203.                                    (IR(7) AND NOT IR(6) AND NOT IR(5) AND IR(4)) O
    R
204.                                    (IR(7) AND NOT IR(6) AND IR(5) AND NOT IR(4))))
     OR
205.                              (W2 AND (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4)));

206.
207.                     LAR <= W1 AND ((NOT IR(7) AND IR(6) AND NOT IR(5) AND IR(4)) OR

208.                                    (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4)));

209.
210.                     SHORT <= W1 AND ((NOT IR(6) AND IR(5)) OR
211.                                    (NOT IR(7) AND NOT IR(6)) OR
212.                                    (NOT IR(7) AND NOT IR(5) AND NOT IR(4)) OR
213.                                    (IR(7) AND IR(5) AND NOT IR(4)) OR
214.                                    (NOT C AND NOT IR(7) AND IR(6) AND IR(5) AND IR
    (4)) OR
215.                                    (NOT Z AND IR(7) AND NOT IR(6) AND NOT IR(5) AN
    D NOT IR(4)));
216.
217.                     MBUS <= W2 AND (NOT IR(7) AND IR(6) AND NOT IR(5) AND IR(4));
218.
```

```
219.                     MEMW <= W2 AND (NOT IR(7) AND IR(6) AND IR(5) AND NOT IR(4));
220.
221.                     PCADD <= W1 AND ((C AND NOT IR(7) AND IR(6) AND IR(5) AND IR(4)
    ) OR
222.                                      (Z AND IR(7) AND NOT IR(6) AND NOT IR(5) AND NO
    T IR(4)));
223.
224.                     LPC <= W1 AND (IR(7) AND NOT IR(6) AND NOT IR(5) AND IR(4));
225.
226.                     STOP <= W1 AND (IR(7) AND IR(6) AND IR(5) AND NOT IR(4));
227.                 END IF;
228.             WHEN OTHERS => NULL;
229.         END CASE;
230.     END PROCESS;
231.  END logic;
```