

计算机组成原理课程设计报告

题目: 简单模型机设计 计算机科学与技术学院 xxxx 级 x 班 xxxxxxxxx XXX 2021 年 11 月 30 日

目录

总述	4
课程设计目的	4
课程设计内容与要求	4
课程设计工具	4
课程设计方法	4
指令系统	5
微程序控制下的指令系统	5
指令格式	5
基本字长	5
指令类型	5
寻址方式	5
实现指令组(22条)	6
硬布线控制下的指令系统	9
指令格式	9
基本字长	9
指令类型	9
寻址方式	10
实现指令组(10条)	10
总体结构与数据通路	12
总体结构框图	12
器件设计以及原理图	12
微程序-Control Unit	20
CU 的实现及其部件原理图	20
指令的执行流程(微程序控制)	24
微指令格式	25
CROM 内的微程序以及地址	25
RAM 内的应用程序	26
硬布线-Control Unit	29
CU 框图以及设计原理	29
指令执行流程	32

XXXXXXXXX 张 XX 计组课设报告

指	令流	程数据通路	.33
RA	MИ	9的应用程序	.35
课程	设计	总结	.36
课	设中	遇到的问题以及解决	.36
收	获与	体会	.37
附录.			.38
附:	录 1	数据通路图	.38
附:	录 2	微程序流程图	.39
附:	录 3	微程序	.40
附:	录 4	控制信号列表	.41
附:	录 5	控制信号逻辑表达式	.42

总述

课程设计目的

通过计算机组成原理的课程设计,进一步深入理解计算机组成的原理,学会运用计算机底层基础知识,充分提高动手能力,巩固对于 CPU 的理解,详细了解微程序和硬布线的设计思路以及其他基础知识,从而提高学生各方面认知、实践能力。

课程设计内容与要求

单独或两人组队合作完成设计任务, 充分利用所学知识, 实现具有一定特色、鲁棒性的模型计算机。

课程设计工具

软件: Quartus II 18.0、Windows 10 OS 硬件: 康芯 KX-CDS 计算机组成实验箱

课程设计方法

课程设计分为两个阶段:第一阶段为微程序实现的模型机内核。通过对于μPC,μIR,CROM 等微程序部件的设计,来实现微程序控制整个 CPU 配合主存RAM 进行各种指令的执行。第二个阶段为硬布线实现的模型机内核。将多个部件通过硬件连线的方式来控制指令流水进行,以达到整个计算机能够正常进行计算、程序执行。

指令系统

微程序控制下的指令系统

指令格式

指令格式有单字长指令和双字长指令,在双字长指令之中,第二字节通常是操作数的值或者是操作数的地址。

单字长指令格式如下:

7	4	3	2	1		0
操作码 OP	寻址方式	寄存器号	寻址方	式	寄存器号	
	源 操	作数	目白	勺 操	作数	_

双字长指令格式如下:

15	1	.2 1	11	10	9	8
操作码	ら OP	寻址方式	寄存器号	寻址方式	寄存器号	
7						0
		操作	三数 / 操作数	文地址		

基本字长

主存 RAM 的基本字长为 1 byte 即 8 位,地址线 8 条 微程序存储器 CROM 的基本微指令字长为 16 位,地址线 10 条

指令类型

- 1) 模型机有单操作数指令、双操作数指令、无操作数指令
- 2) 操作码有 4 位, 最多可以定义 16 条不同的指令
- 3) 无操作数指令有: Fetch 取指指令, HALT 停机指令
- 4) 单操作数指令有: lw sw 取数存数指令, DEC 自减指令,SDL SDR 算数左移右移指令、JMP 无条件跳转等
- 5) 双操作数指令有:ADD加法指令,SUB减法指令,XOR异或指令,MUL乘法指令,DIV除法指令, JNE有条件跳转指令等

寻址方式

- 1) 寻址方式位为00为立即数寻址, 操作数在指令的下一个单元
- 2) 寻址方式位为01为寄存器寻址,操作数在寄存器中,寄存器号为0

代表操作数在RO寄存器中,寄存器号为1代表操作数在R1寄存器中

- 3) 寻址方式位为10为直接寻址、操作数的地址在指令的下一个单元
- 4) 寻址方式位为11为寄存器间接寻址, 操作数的地址在寄存器里面

实现指令组(22条)

(1) Fetch

功能:实现取指,进行取指周期操作,从内存单元取出这次要执行的指令,存入 IR,更新 MAR 以及 PC 指令长度: 8 位

(2) LW \$05 R0

功能:从存储器中读取一个 word 的数据,存到寄存器 RO 之中。例如:将立即数 05 送到寄存器 RO 之中寻址方式:立即数寻址|寄存器寻址指令长度:可以认为是双字长指令 16 位,立即数存在后一个内存单元。

(3) LW \$06 R1

功能:从存储器中读取一个 word 的数据,存到寄存器 R1 例如:将立即数 06 送到寄存器 R1 之中寻址方式:立即数寻址|寄存器寻址指令长度:可以认为是双字长指令 16 位,立即数存在后一个内存单元。

(4) ADD R0 R1

功能:把RO和R1的数据相加,存入R1中RO+R1->R1寻址方式:寄存器寻址 |寄存器寻址 |寄存器寻址 指令长度:8位

(5) SW R1 (Address*)

功能:把R1中的数据存到 后面立即数对应的地址中,立即数是从下一个内存单元取出的。并且更新PC。指令长度:可以认为是双字长指令16位,立即数作为地址,存在后一个内存单元

(6) HALT

功能:停机指令,使节拍发生器暂停,从而能够让模型机暂停运行。 指令长度:8位

(7) SUB R0 R1

功能: 即把 RO 中的数据作为被减数, R1 作为减数, 得到的差存

在 R1 中。

寻址方式:寄存器寻址 | 寄存器寻址

指令长度:8位

(8) XOR R0 R1

功能: R0 中的数据与 R1 中的数据进行异或操作, 然后存入 R1 之中。

指令长度:8位

(9) DEC R0

功能: --R0 -> R0 将 R0 中的数字自减, 然后存入 R0 中, 设计 DEC 指令能够便于后面实现复杂程序的实现, 比如利用循环实现求和程序

指令长度:8位

(10) SW R0 (Address*)

功能:把RO中的数据存到 后面立即数对应的地址中,立即数是从下一个内存单元取出的。并且更新PC。

指令长度:可以认为是双字长指令16位,立即数作为地址,存在后一个内存单元

(11) MUL R0 R1 -> R3

功能:把 RO 和 R1 中的八位数据相乘,低位存入 R3 寄存器之中,即自己手动实现了八位 X 八位的乘法。

指令长度:8位

(12) DIV R0 R1 -> R1

功能:把 RO 和 R1 的八位数据相除, 商存入 R1 之中, 余数对其, 即自己动手实现了八位除八位的除法。

指令长度:8位

(13) JMP (Address*)

功能: 无条件跳转,即根据下一内存单元的值来进行修改 PC 和MAR,实现跳转。之所以设置成双字长指令是因为 RAM 内存空间为 256,需要八位才能寻完所有空间,所以需要单独一个内存单元的字长来存跳转的地址。

指令长度: 可以认为是双字长指令 16 位, 立即数作为地址, 存在后一个内存单元, 然后 jump 到对应的地址

(14) JNE R0 R1

功能: 即条件跳转指令。 判断 RO 和 R1 是否相等,如果不相等,则跳转到下一内存单元所存值的地址去,如果相等,则顺序执行。其中用到了一个判断寄存器 JUDGE 来实现判断

指令长度:可以认为是双字长指令16位,立即数作为地址,存在

于后一个内存单元,然后 jump 到对应的地址

(15) ADD R1 R3

功能: 把R1和R3的数据相加, 存入R3中R1+R3->R3, 从而能够把R3作为一个存前缀和的寄存器,配合DECR1指令, 就能实现求前缀和的函数。

寻址方式:寄存器寻址 | 寄存器寻址

指令长度:8位

(16) MOV R3 R0

功能:实现寄存器之间的数据流动与复制,即把寄存器 R3 中的数据,复制到 R0 之中,配合 SW R0 的指令,就能实现把前缀和存入内存 RAM。

指令长度:8位

(17) SAL R0

功能: 算术左移, 单目运算, 把 RO 算数左移后存入 RO 寄存器之中。

指令长度:8位

(18) SAR RO

功能: 算术右移, 单目运算, 把 RO 算数右移后存入 RO 寄存器之中。

指令长度:8位

(19) PUSH p

功能:入栈指令,认为内存单元最高地址是一个栈结构,而 PUSH 就是根据当前的 SP(栈指针)值进行数据 p 的入栈操作,入栈之后更新 SP 栈指针的值。

指令长度: 16位

(20) POP

功能: 出栈指令,将栈顶的数据弹出,并且修改 SP 指令长度: 8 位

(21) 间接寻址(可多重)

功能:多重间接寻址,间接寻址可能许多进行多次,为了能够正常返回最初间址的地址,每次在间址之前需要 PUSH 把当前 PC 存入主存 RAM 的栈结构中,然后进行间址指令(双字长指令)根据下一单元的地址作为指令的地址,去取指,然后如果说取到的还是间址指令,则继续间址。当间址结束,则 POP 出原来的 PC值,修改对应的 PC和 MAR 寄存器的值,实现依次返回,回到最初的起点,继续执行程序。

指令长度: 16位

(22) Lw SP

功能:专门用一个 SP 寄存器来维护栈指针 Sp 的值, Lw SP 就是把下一内存单元的值赋给 SP 寄存器。

指令长度: 16位

硬布线控制下的指令系统

指令格式

指令格式有单字长指令和双字长指令,在双字长指令之中,第二字节通常是操作数的值或者是操作数的地址。

单字长指令格式如下:

7	4	3	2	1		0
操作码 OP	寻址方式	寄存器号	寻址方式	弋	寄存器号	
_	源 操	作数	目的	力操	作数	

双字长指令格式如下:

15	12	11	10	9	8
操作码 0	P 寻址方式	寄存器号	寻址方	式 寄存	器号
7					0
	操	作数 / 操作	数地址		

基本字长

主存RAM的基本字长为1 byte 即8位,地址线8条容量为8 x 256 bit 无微程序存储器 CROM

指令类型

- 1) 模型机有单操作数指令、双操作数指令、无操作数指令
- 2) 操作码有 4 位, 最多可以定义 16 条不同的指令
- 3) 无操作数指令有: Fetch 取指指令, HALT 停机指令
- 4) 单操作数指令有: lw sw 取数存数指令, SDL SDR 算数左移右移 指令等
- 5) 双操作数指令有: ADD 加法指令,SUB 减法指令 ,MUL 乘法指令,DIV 除法指令等

寻址方式

- 1) 寻址方式位为00为立即数寻址, 操作数在指令的下一个单元
- 2) 寻址方式位为01为寄存器寻址,操作数在寄存器中,寄存器号为0 代表操作数在R0寄存器中,寄存器号为1代表操作数在R1寄存器中
- 3) 寻址方式位为10为直接寻址、操作数的地址在指令的下一个单元
- 4) 寻址方式位为11为寄存器间接寻址, 操作数的地址在寄存器里面

实现指令组(10条)

(1) lw \$05 R0

功能:从存储器中读取一个 word 的数据, 存到寄存器 RO 之中。

例如:将立即数 05 送到寄存器 R0 之中

寻址方式:立即数寻址|寄存器寻址

指令长度:可以认为是双字长指令 16 位,立即数存在后一个内存单元。

(2) lw \$06 R1

功能:从存储器中读取一个 word 的数据, 存到寄存器 R1

例如:将立即数 06 送到寄存器 R1 之中

寻址方式: 立即数寻址|寄存器寻址

指令长度:可以认为是双字长指令 16 位,立即数存在后一个内

存单元。

(3) ADD R0 R1

功能: 把 RO 和 R1 的数据相加, 存入 R1 中 R0 + R1 -> R1

寻址方式:寄存器寻址 | 寄存器寻址

指令长度:8位

(4) sw R1 (Address#)

功能:把 R1 中的数据存到 后面立即数对应的地址中,立即数是从下一个内存单元取出的。并且更新 PC。

指令长度:可以认为是双字长指令16位,立即数作为地址,存在后一个内存单元

(5) HALT

功能:停机指令,使节拍发生器暂停,从而能够让模型机暂停运行。

指令长度:8位

(6) SUB RO R1

功能:即把RO中的数据作为被减数,R1作为减数,得到的差存在R1中。

寻址方式:寄存器寻址 | 寄存器寻址

指令长度:8位

(7) MUL R0 R1 -> R2

功能:把 RO 和 R1 中的八位数据相乘,低位存入 R2 寄存器之中,即自己手动实现了八位 X 八位的乘法。

指令长度:8位

(8) DIV R0 R1 -> R2

功能:把 RO 和 R1 的八位数据相除,商存入 R2 之中,余数对其,即自己动手实现了八位除八位的除法。

指令长度:8位

(9) SAL R0

功能: 算术左移, 单目运算, 把RO 算数左移后存入RO 寄存器之中。

指令长度:8位

(10) SAR R0

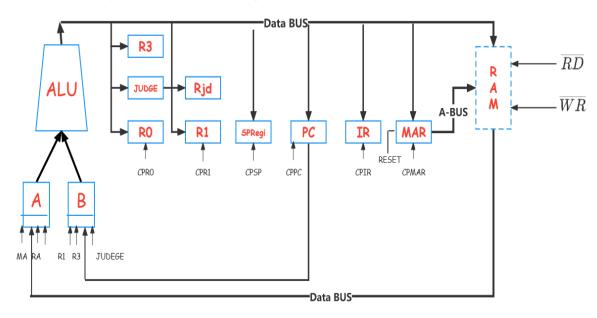
功能: 算术右移, 单目运算, 把RO 算数右移后存入RO 寄存器之中。

指令长度: 8位

总体结构与数据通路

总体结构框图

微程序和硬布线整体架构框图一样 如下图所示:

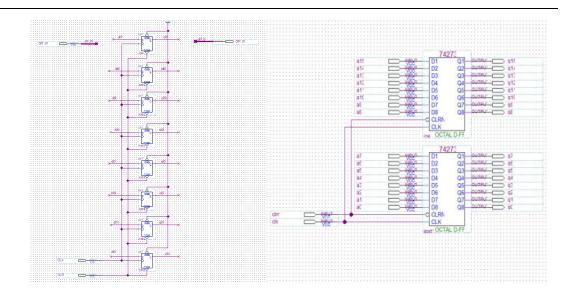


器件设计以及原理图

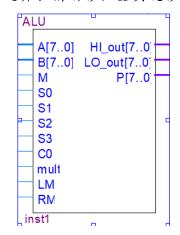
叙述微程序整体架构框图中的每个部件的功能和作用, 并且给出设计原理图。

- 1. RO 寄存器 R1 寄存器 R3 寄存器 SPRegi 寄存器 为 16 位通用寄存器 功能:存储数据、指令。
- 2. PC 寄存器为程序计数器, 16 位功能:存储当前程序所在的地址。
- 3. IR 寄存器为指令寄存器, 16 位功能:存储指令。
- 4. MAR 寄存器 为地址寄存器, 8 位 功能:存储数据或者指令地址,从而访问对应的主存 RAM 下的数据。
- 5. Rjd 为判断寄存器, 16 位 功能: 存判断结果。

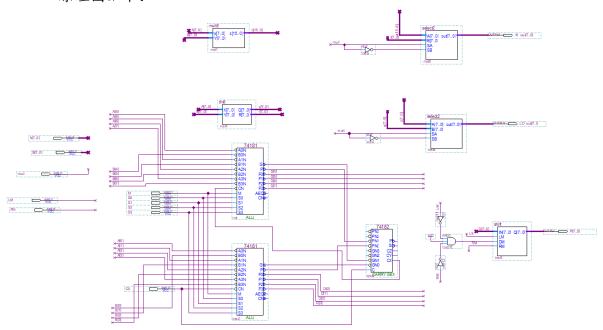
以上的寄存器都采用 2 片 74273 组成带复位的寄存器或者用 D 触发器设计,如下图所示:



6. ALU 为 8 位运算器,功能:可以进行逻辑运算、算术运算、借助乘法器和除法器也能实现乘除运算、借助移位器实现移位运算。框图如下:



原理图如下:

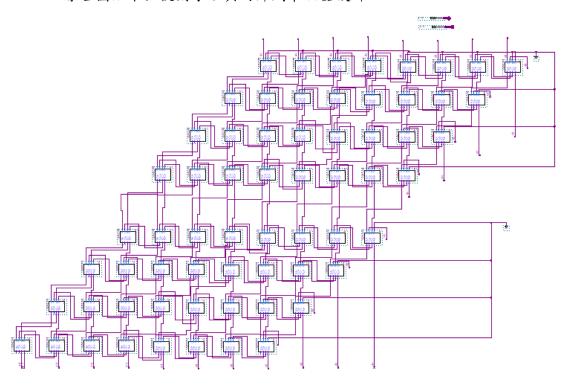


7. 八位阵列乘法器:

功能实现了8*8的乘法器,输出15位数字,框图如下:

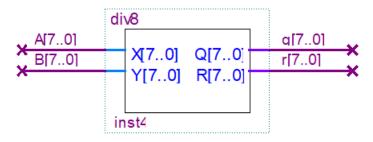


原理图如下: 使用了经典的阵列乘法器设计

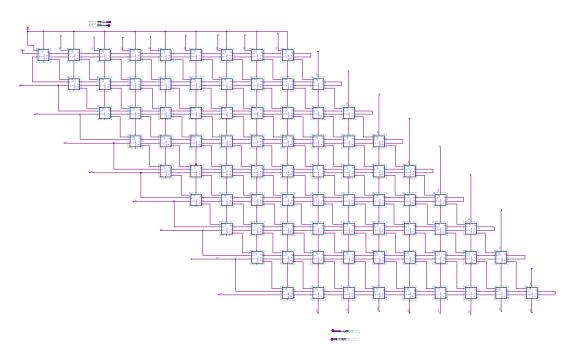


8. 除法器设计:

功能:实现了8/8的阵列除法器,输出8位商q以及8位余数r:

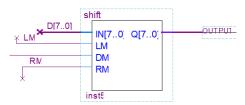


原理图如下:

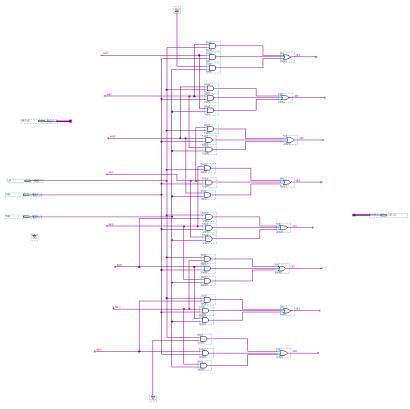


9. 移位器:

功能:能够实现八位数据的左移右移以及直传。使用最基本的与或非实现

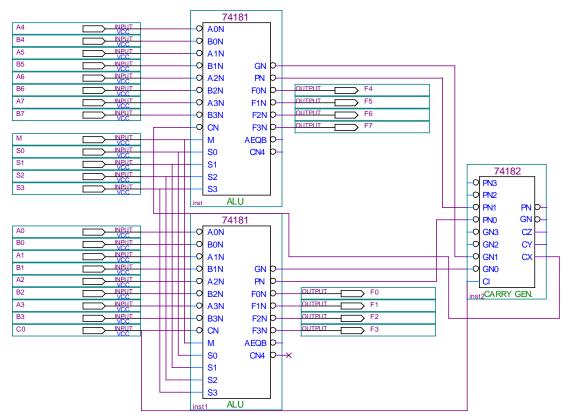


原理图:



15 / **43**

10.ALU 中算数与逻辑运算的部件,即八位超前进位并行加法器:使用两片 74181 和 一片 74182。功能:进行算数运算以及逻辑运算,例如加减、逻辑加减等等。



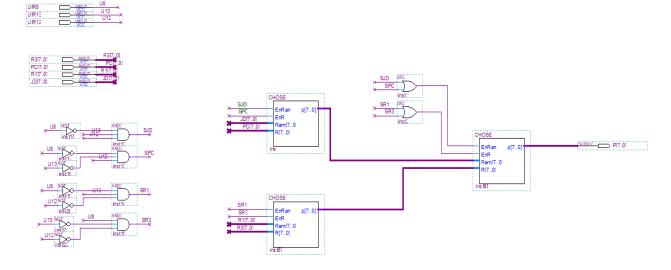
11.A、B为 八位四选一数据选择器

功能:

A选择器 可以将来自于 RAM RO 以及 SPRegi 的数据根据 CU 的控制信号进行选择

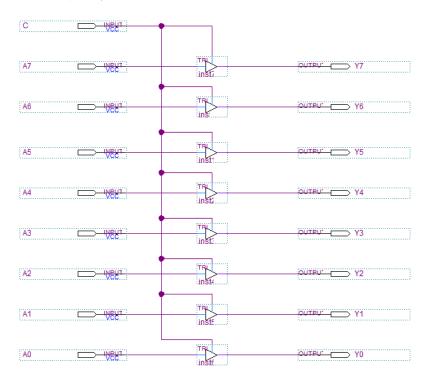
B选择器 可以将来自于 PC R1 R3 以及 Rjd 的数据根据 CU 的控制信号进行选择

原理图: (借助于八位2选1数据选择器)



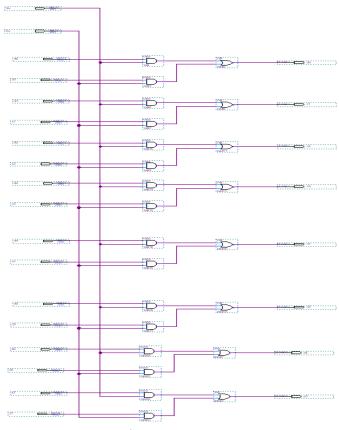
12. 八位三态门 TRI-8 的设计:

功能:实现数据的三态传输 实现:使用了八个TRI实现:



13. 二选一数据选择器的设计:

功能:实现八位数据的片选。实现: 先和片选信号 and, 然后 or。

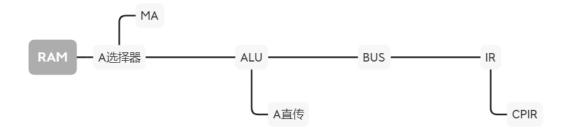


17 / 43

数据流动及通路

部件之间的连接是采用了以 CPU 为中心的总线连接方式。

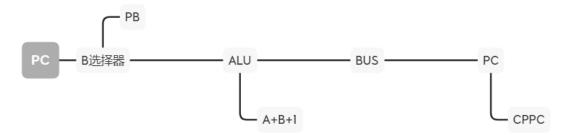
1. 取指令时的数据通路: 从 RAM 取出指令, 通过选择器选择, 然后经过 ALU 直传之后, 输入到 D-BUS 数据总线, 然后在 CPIR 控制下, 打入 IR。



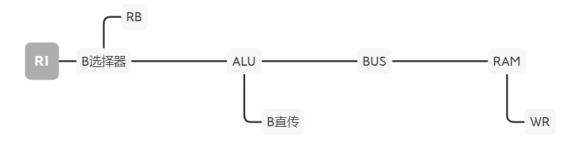
2. 送指令数据通路: 从 PC 经 B 选择器和 ALU 直传后送至 MAR



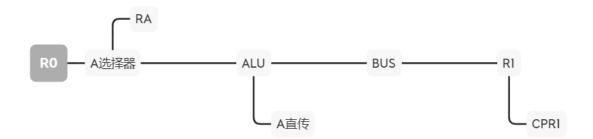
3. PC+1 数据通路:



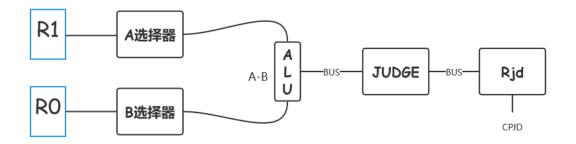
4. R1 到 RAM 的数据通路: 将 R1 存入 RAM 之中



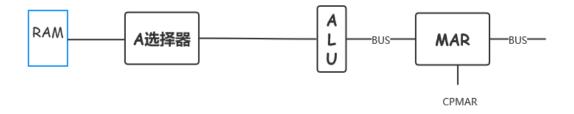
5. R0 -> R1



6. 判断 RO R1 是否相等的数据通路



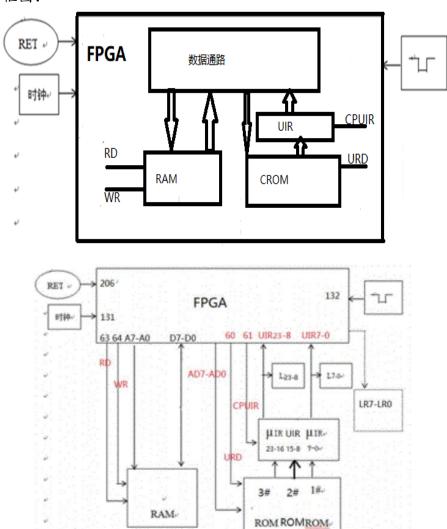
7. JP 跳转的时候的数据通路



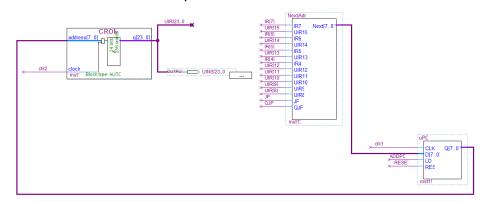
微程序-Control Unit

CU的实现及其部件原理图

1. CU 框图:



对应于原理图: 主要由 CROM μPC 下地址形成部件 NextAdr 以及 RAM 组成



2. 部件设计及功能

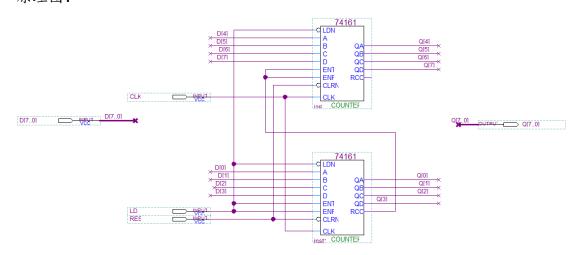
1) μPC

功能:实现微指令的地址控制。 μPC 使用两个 74161 器件组成,实现 八位同步二进制计数器,其中 74161 功能表如下:

清0	预置	控	制	时钟	ř	置数	据输)	\		输	出	
\overline{R}_D	\overline{LD}	EP	ET	CP	A₃	A ₂	A ₁	A _o	Q ₃	Q_2	Q_1	Q_0
0	×	×	×	×	X	X	X	X	0	0	0	0
1	0	×	×	↑	d₃	d₂	d,	d₀	d₃	d₂	d,	d₀
1	1	0	×	×	×	×	×	×		保	持	
1	1	×	0	×	×	×	×	×		保	持	
1	1	1	1	↑	×	×	×	×		ì†	数	

LD=0 且 CP 为上升沿的时候,进行置数,LD =1 且 CP 为上升沿进行计数,从而实现 μ PC 的自加 1。

作用: 微地址形成电路中的核心部件,能够使 CU 顺序执行微命令,并且能够结合 QJP,实现不同的下地址形成方式。 原理图:

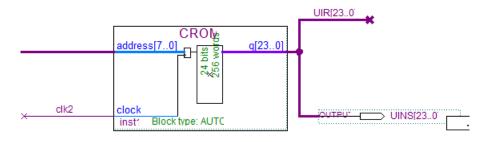


2) 控制存储器 CROM

功能: 存储微程序、微指令,在时钟脉冲以及 IR 的作用下,执行对应节拍和指令的微指令,发出微命令。

作用: 存储微指令

原理图:

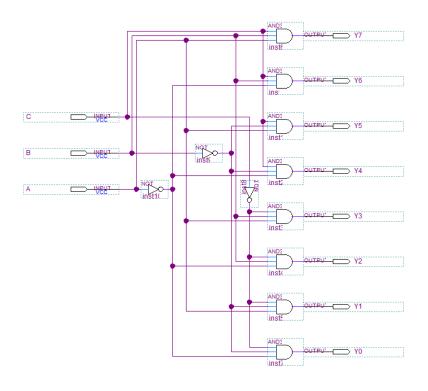


3) 译码器

功能: 38 译码器来根据输入的微指令发出对应的信号,例如片选信号、打入信号、下地址形成方式信号等等

作用:形成微地址形成电路的控制信号

原理图:进行3-8译码

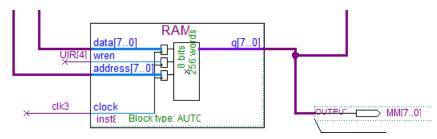


4) RAM

功能:随机存取存储器,能够根据不同的地址以及读写信号进行读写操作,能够查看程序是否成功运行

作用:存储指令和数据、模拟主存 Main Memory

原理图:

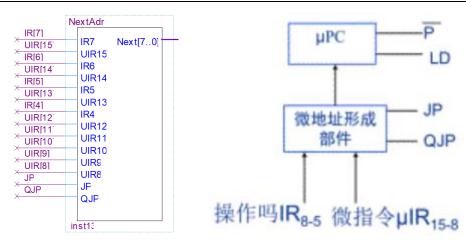


5) 下地址形成部件 NextAdr

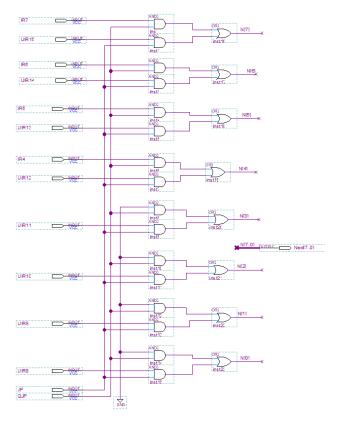
功能:产生后继微地址,即配合 μ PC来实现不同的后继微地址生成方式,包括:增量方式(μ PC+1)、无条件转移方式(JP)、按操作码转移方式(QJP) 还有 YJP 给定高四位低四位按源寻址方式转移、MJP 给定高四位低四位、按目寻址方式转移。

作用:产生后继微地址。

框图:



原理图:根据输入的 IR UIR 以及 JP QJP 来形成下地址

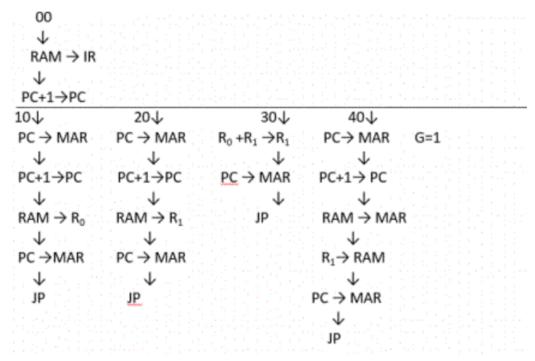


指令的执行流程(微程序控制)

列出了取指周期以及执行周期的指令执行流程,包括每条指令(基本指令)的取指周期以及执行周期的微操作和节拍安排。

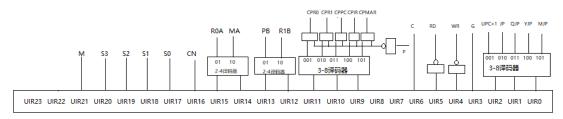
这里取指周期有三个节拍, 执行周期有四个节拍

工作周期	节拍	微操作	MOV1	MOV2	MOV3	ADD1	ADD2	SUB	HALT
	T0	RAM->MAR	1	1	1	1	1	1	1
取指周期	T1	PC+1->PC	1	1	1	1	1	1	1
	T2	QJP	1	1	1	1	1	1	1
		PC->MAR	1	1	1				
		R0+R1->R1				1			
	T0	R0-R1->R1						1	
		G=1							1
		R0->MAR					1		
		PC+1->PC	1	1	1				
	T1	PC->MAR				1		1	
执行周期		RAM->R0					1		
17(1] /可粉		RAM->R0	1						
	T2	RAM->R1		1					
	12	PC->MAR					1		
		RAM->MAR			1				
		PC->MAR	1	1					
	T3	R0+R1->R1					1		
		R1->RAM			1				
	T4	PC->MAR			1				

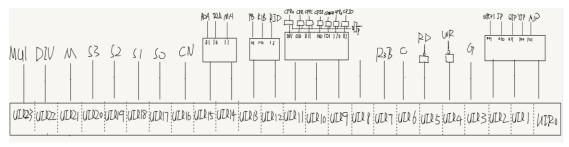


微指令格式

基础的形式:



拓展之后的微指令格式:



微指令字长: 24位

微指令格式: 单字长指令

微地址下地址形成方式: QJP JP uPC+1...

作用: μ IR23 控制乘法、 μ IR22 控制除法、 μ IR21- μ IR16 控制 ALU 算数以及逻辑运算, μ IR15 14 实现 A 选择器的片选, μ IR13 12 7 控制 B 选择器的片选, μ IR11- μ IR9 片选打入寄存器的信号、 μ IR5 4 控制 RAM 的读写, μ IR3 控制停机, μ IR2- μ IR0 片选下地址形成方式的控制信号。

CROM 内的微程序以及地址

CROM 微程序以及对应的地址,更详细的见附录 3。 注:为了实现更多(>16)指令而不破坏指令结构,编写了两个 CROM,即复用了部分微指令空间 微程序入口

指令	入口地址
Fetch	00H
LW R0	10H
LW R1	20H
ADD R0 R1	30H
SW R1	40H
HALT	50H
SUB R0 R1	60H
XOR R0 R1	70H
DEC R0	80H
SW R0	90H
MUL R0 R1	A0H

JMP	вон	
JNE	C0H	
DIV	D0H	
ADD R0 R3	E0H	
SW R3	F0H	
SAL R0	D0H	
SAR	A0H	
POP	E0H	
PUSH	60H	
EA(间址)	E0H	
LW SP	70H	

RAM 内的应用程序

简述:

利用了函数、存储程序的思想

首先进行了加、减、异或、左移、右移、间接寻址的测试,并存储或输出结果。

测试完成之后,通过跳转指令,跳转到 Function1 的首地址

Function1: 地址 A0H 首先初始化 R0R1

存储 R0*R1 和 R0 / R1 的结果

跳转到 Funtion2 的首地址

Function2: 地址 B0H

进行从 RO/R1 到 RO * R1 之间所有数字的求和, 利用循环程序实现首先初始化 R3=0, 作为存和的寄存器 然后进行循环, 循环结构:

```
LOOP
```

R3 = R0 + R3;R0 - -

判断 RO 是否等于 R1 进行跳转;

END

汇编代码:

LW 05# R0; 给 R0 赋值 05 LW 06# R1; 给 R1 赋值 05 ADD R0 R1; R0 和 R1 相加

SW R1 (E0); 存加法结果到地址 E0

SUB RO R1; RO 和 R1 相减

SW R1 (E1); 存减法结果到地址 E1

JMP AO#; 无条件跳转到函数 1 的入口地址 AO

Function 1:

MOV R3 R0; 将 R3 中的数据复制到 R0

SW R0 (F0); 存乘法结果到 F0 SW R1 (F1); 存除法结果到 F1

JMP B0#; 无条件跳转到函数 2 的入口地址 B0

Function 2:

LW 00# R3; 给 R3 赋值 00

ADD R0 R3; R0 与 R3 求和, 存到 R3, 相当于累和

DEC RO; RO 自减

JNP RO R1; 判断 RO 和 R1 是否相等,不相等则跳转 add,

否则顺序执行

MOV R3 R0; 将 R3 中的数据复制到 R0, 存最终结果

SW R0 (F2); 存乘法结果到 F2 主存地址空间

HALT; 停机指令

(循环部分已标注)

LW SP FF; SP 寄存器赋值 FF LW R0 FF; R0 赋值 FF

LW R1 01; R1 赋值 01

SAR R0; R0 右移存入 R0 SW R0 (E0); 右移结果存入 E0 SAL R0; R0 左移存入 R0 SW R0 (E1); 左移结果存入 E1

PUSH; PC 入栈

EA (A0); 到地址 A0 间接寻址

POP; PC 出栈返回 HALT; 停机指令

机器语言:

RAM1:

RAM 地址	机器语言
000H	0001 1000
001H	0000 0101
002H	0010 1001
003H	0000 0001
004H	0110 0001
005H	0100 0111
006H	1110 0000
007H	0111 0001
008H	0100 0111

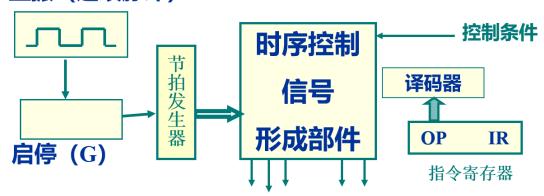
009H	1110 0001
00AH	1011 1000
0A0H	0001 1000
0A1H	0000 0110
0A2H	0010 1001
0A3H	0000 0011
0A4H	1010 0000
0A5H	1101 0000
0A6H	1111 0000
0A7H	0100 0111
0A8H	1111 0000
0A9H	1001 0011
0AAH	1111 0001
0ABH	0011 0000
0B0H	1110 0000
0B1H	1000 0000
0B2H	1100 0001
0B3H	1111 0000
0B4H	1001 0011
0B5H	1111 0010
0B6H	0101 0000
RAM2:	
RAM 地址	机器语言
000H	0111 0000
001H	1111 1111
002H	0001 1000
003H	1111 1111
004H	0010 1001
005H	0000 0001
006H	
006H	1010 0000
005H 007H	1010 0000 1001 0011
007H	1001 0011
007H 008H	1001 0011 1110 0000
007H 008H 009H	1001 0011 1110 0000 1101 0000
007H 008H 009H 00AH	1001 0011 1110 0000 1101 0000 1001 0011
007H 008H 009H 00AH 00BH	1001 0011 1110 0000 1101 0000 1001 0011 1110 0001
007H 008H 009H 00AH 00BH 00CH	1001 0011 1110 0000 1101 0000 1001 0011 1110 0001 0110 0000
007H 008H 009H 00AH 00BH 00CH	1001 0011 1110 0000 1101 0000 1001 0011 1110 0000 1110 0000
007H 008H 009H 00AH 00BH 00CH 00DH	1001 0011 1110 0000 1101 0000 1001 0011 1110 0001 0110 0000 1110 0000
007H 008H 009H 00AH 00BH 00CH 00DH 00EH	1001 0011 1110 0000 1101 0000 1001 0011 1110 0000 1110 0000 1010 0000 1101 0000

硬布线-Control Unit

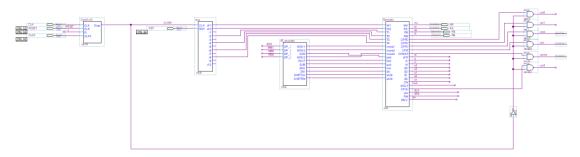
CU框图以及设计原理

• CU 框图

主振 (连续脉冲)



对应于原理图:

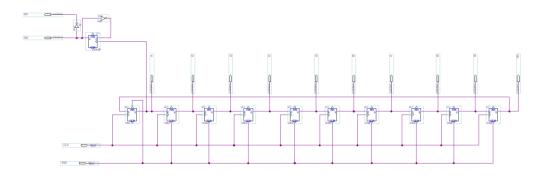


• 节拍发生器

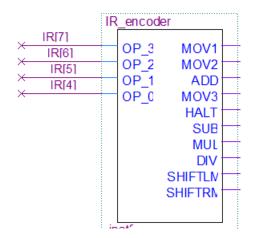
功能:将单个连续脉冲,转化为多个节拍的连续脉冲,从而实现节拍控制的指令流水。

作用:指令执行步骤的接续是通过变换节拍发生器的状态组合完成的 相当于 微程序设计里面的下地址形成部件,控制下一条指令的执行。

原理图:这里将一个周期的连续脉冲变成了 10 个节拍,之所以这样是可能更灵活的选择某节拍周期的相对长短,便于控制。



• 译码器

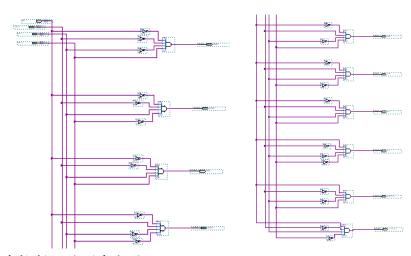


框图:

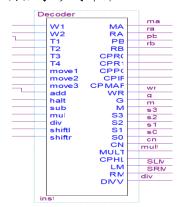
功能:替代微程序中的微地址映射部件,变成操作码译码器,根据取出的指令来产生对应的节拍中的对应的控制信号

作用: 根据操作码来产生控制信号

原理图:



• 时序控制信号形成部件

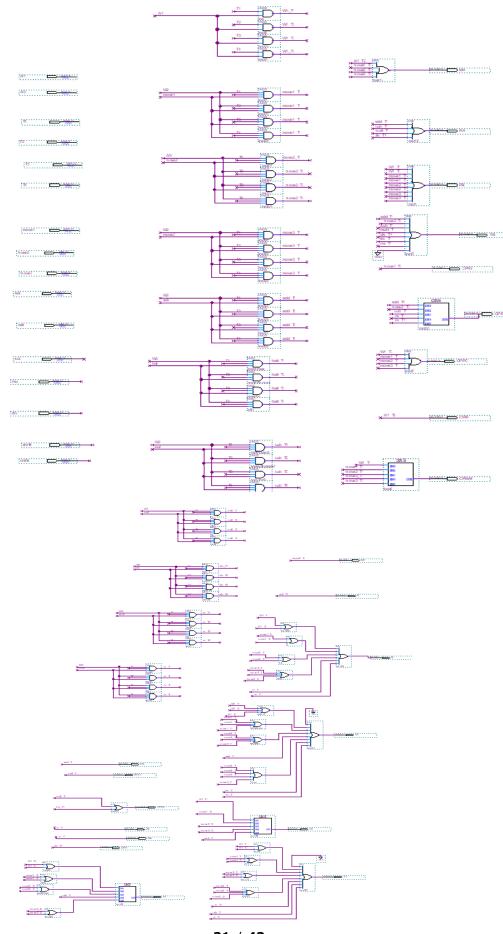


框图:

功能:通过当前的周期以及节拍和执行的指令信号,来产生控制信号,取代了微程序中的微指令寄存器,产生的控制信号使得模型机可以顺序执行程序,执行微操作。

作用:产生当前节拍下的时序控制信号

原理图:



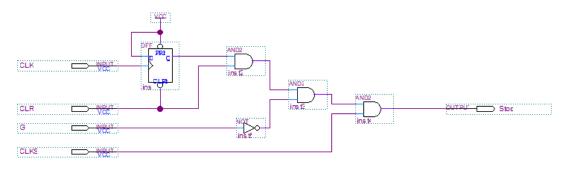
31 / 43

• 启停电路

功能:将一个单脉冲转化为连续脉冲,并且通过控制信号来实现停机或者暂停

作用:控制整个模型机的启动、暂停与停止

原理图:



其他部件与微程序设计的模型机完全一致~

指令执行流程

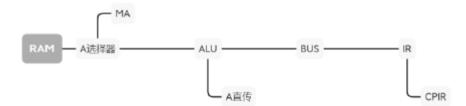
表格: 列出每条指令的取指周期以及执行周期的微操作和节拍安排

周期	节	微操作	LWR0	LWR1	SWR1	ADD	SUB	HALT
	拍							
取指	T0	PC->MAR	1	1	1	1	1	1
周期	Т1	DAM > ID	1	1	1	1	1	1
	T1	RAM->IR	1	1	1	1	1	1
	T2	PC+1->PC	1	1	1	1	1	1
11. /-					_			
执行	T0	PC->MAR	1	1	1			
周期		R0+R1->R1				1		
		R0-R1->R1					1	
		G=1						1
	T1	PC+1->PC	1	1	1			
		PC->MAR				1	1	
	T2	RAM->R0	1					
		RAM->R1		1				
		PC->MAR						
		RAM->MAR			1			
	T3	PC->MAR	1	1				
		R1->RAM			1			

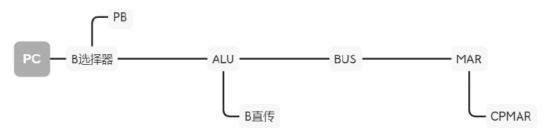
周期	节拍	微操作	MUL	DIV	SAL	SAR
取指周期	ТО	PC->MAR	1	1	1	1
7.4.294	T1	RAM->IR	1	1	1	1
	T2	PC+1->PC	1	1	1	1
执行	T0	PC->MAR				
周期		R0*R1->R3	1			
		R0/R1->R3		1		
		R1<< ->R1			1	
		R1>> ->R1				1
	T1	PC+1->PC				
		PC->MAR	1	1	1	1
	T2	RAM->R0				
		RAM->R1				
		PC->MAR				
		RAM->MAR				
	T3	PC->MAR				
		R1->RAM				

指令流程数据通路

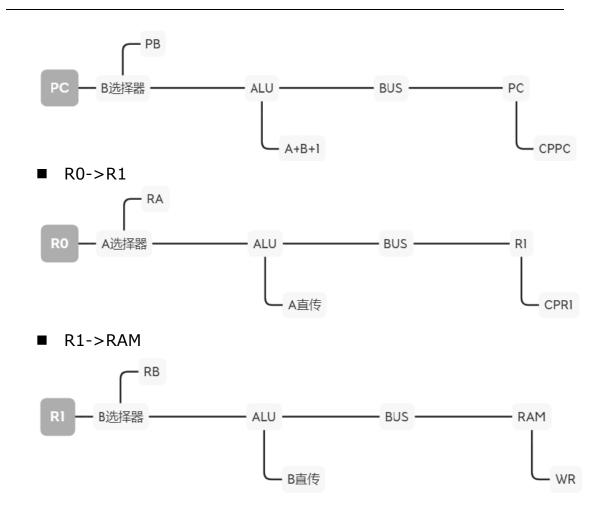
- 指令流程和数据通路 根据以下的数据通路就能实现指令的执行
 - 取指令



■ 送指令地址



■ PC+1



• 各控制信号的列表 详细请见附录 4

1	1: MOV1 2:MOV2	3:ADD 4:MOV3 5H	ALT 6SUB 7MUTI 8D	IV 9LEFT A R	IGHT A:RAM RO B:PC R1							
状态	W1 (取指周期)		₩2		₩2(执行周期)							
指令	所有指令	MOV1	MOV2	ADD (0011)	NOV3(0100)	HALT0101	减法(0110)	乘(0111)	除(1000)	左移(1001)	右移1010	
控制信号												
MA(RAM)	T2	T3	T3		T3							
7 RA(RO)				T1			T1	T1	T1			
PB (PC)	T1、T3	T1、T2	T1、T2		T1、T2							
RB(R1)				T1	T4		T1	T1	T1	T1	T1	
o CPRO		T3										
1 CPR1			T3	T1			T1			T1	T1	
2 CPPC	T3	T2	T2		T2							
3 CPIR	T2											
4 CPMAR	T1	T1	T1		T1、T3							
5 RD	T2	T3	T3		T3							
6 VR					T4							
7 驱动C					T4							
8 G						T1						
9 M	1 (T1) 、1 (T2) 、0 (T3)	1 (T1) 、0 (T2) 、1 (T3)	1 (T1) 、0 (T2) 、1 (T3)	0 (T1)	1 (T1) 、0 (T2) 、1 (T3) 、1 (T4)		0 (T1)			T1	T1	
	1 (T1) 、1 (T2) 、1 (T3)	1 (T1) 、1 (T2) 、1 (T3)	1 (T1) 、1 (T2) 、1 (T3)	1 (T1)	1 (T1) 、1 (T2) 、1 (T3) 、1 (T4)		0 (T1)			T1	T1	
1 S2	0 (T1) 、1 (T2) 、0 (T3)	(T1) 、0 (T2) 、1 (T3)	0 (T1) 、0 (T2) 、1 (T3)	0 (T1)	0 (T1) 、0 (T2) 、1 (T3) 、0 (T4)		1 (T1)					
2 S1	1 (T1) 、1 (T2) 、0 (T3)	1 (T1) 、0 (T2) 、1 (T3)	1 (T1) 、0 (T2) 、1 (T3)	0 (T1)	1 (T1) 、0 (T2) 、1 (T3) 、1 (T4)		1 (T1)			T1	T1	
3 S0	D (T1) 、1 (T2) 、1 (T3)	(T1) 、1 (T2) 、1 (T3)	0 (T1) 、1 (T2) 、1 (T3)	1 (T1)	0 (T1) 、1 (T2) 、1 (T3) 、0 (T4)		0 (T1)					
4 CN	0 (T3)	0 (T2)	0 (T2)	1 (T1)	0 (T2)		0 (T1)					
5 MULT								1(T1)				
6 CPR2								1(T1)	1(T1)			
7 LM										1(T1)		
8 RM											1(T1)	
9	T1:PC-> MAR T2:RAM->IR T3: PC+1->PC	T1: PC->MAR T2: PC+1->PC T3:RAM->R0	T1: PC->MAR T2: PC+1->PC T3:RAM->R1	T1: R0+ R1 -> R1	T1: PC->MAR T2: PC+1->PC T3:RAM->MAR T4:R1->RAM		T1: R0- R1 -> R1	T1: R0* R1 -> HILC	T1: R0/R1 -> HILO	T1: R1 << -> R1	1: R1 >> ->	

• 各控制信号的逻辑表达式 见附录 5

RAM 内的应用程序

• 简述:

进行了八位的加、减、乘、除、左移、右移的测试,并且存到了 RAM 之中

• 汇编代码:

LW 06# R0; 给 R0 赋值 06 LW 01# R1; 给 R1 赋值 01

ADD RO R1; RO 和 R1 相加, 结果存到 R1

SW R1 (E0); 存加法结果到地址 E0

SUB R0 R1; R0 和 R1 相减

SW R1 (E1); 存减法结果到地址 E1 LW 02\$ R1; 存减法结果到地址 E1

MUL R0 R1; 进行 R0*R1 的八位乘法, 存入 R3 之中 DIV R0 R1; 进行 R0/R1 的八位除法, 存入 R3 之中

SAL R1; R1 左移存入 R1

 SRL R1;
 R1 在之前左移的基础上,右移存入 R1

 SW R1(E2);
 存入移位结果到内存 RAM 的 E2 地址之中

• 机器语言

000H 0001 1000 001H 0000 0110 002H 0010 1001 003H 0000 0001 004H 0011 0001 005H 0100 0111 006H 1110 0000 007H 0110 0001 008H 0100 0111 009H 1110 0001 00BH 0000 0010 00CH 0111 0001 00DH 1000 0001 00EH 1001 0001 01H 0100 0111 01H 1110 0001 012H 1010 0001 013H 0100 0111	D 4 4 4 1 1 1 1	1 m - 2
001H 0000 0110 002H 0010 1001 003H 0000 0001 004H 0011 0001 005H 0100 0111 006H 1110 0000 007H 0110 0000 008H 0100 0111 009H 1110 0001 00AH 0010 1001 00CH 0111 0001 00DH 1000 0001 00EH 1001 0001 01H 0100 0111 01H 1110 0010 012H 1010 0001 013H 0100 0111	RAM 地址	机器语言
002H 0010 1001 003H 0000 0001 004H 0011 0001 005H 0100 0111 006H 1110 0000 007H 0110 0000 008H 0100 0111 009H 1110 0001 00AH 0010 1001 00BH 0000 0010 00CH 0111 0001 00DH 1000 0001 00EH 1001 0001 01H 0100 0111 01H 110 0010 012H 1010 0001 013H 0100 0111	000H	0001 1000
003H 0000 0001 004H 0011 0001 005H 0100 0111 006H 1110 0000 007H 0110 0000 008H 0100 0111 009H 1110 0001 00AH 0010 1001 00BH 0000 0010 00CH 0111 0001 00DH 1000 0001 00EH 1001 0001 01H 0100 0111 01H 1100 0001 012H 1010 0001 013H 0100 0111	001H	0000 0110
004H 0011 0001 005H 0100 0111 006H 1110 0000 007H 0110 0000 008H 0100 0111 009H 1110 0001 00AH 0010 1001 00BH 0000 0010 00CH 0111 0001 00DH 1000 0001 00EH 1001 0001 01H 1110 0010 012H 1010 0001 013H 0100 0111	002H	0010 1001
005H 0100 0111 006H 1110 0000 007H 0110 0000 008H 0100 0111 009H 1110 0001 00AH 0010 1001 00BH 0000 0010 00CH 0111 0001 00DH 1000 0001 010H 0100 0111 011H 1110 0010 012H 1010 0001 013H 0100 0111	003H	0000 0001
006H 1110 0000 007H 0110 0000 008H 0100 0111 009H 1110 0001 00AH 0010 1001 00BH 0000 0010 00CH 0111 0001 00DH 1000 0001 010H 0100 0111 011H 1110 0010 012H 1010 0001 013H 0100 0111	004H	0011 0001
007H 0110 0000 008H 0100 0111 009H 1110 0001 00AH 0010 1001 00BH 0000 0010 00CH 0111 0001 00DH 1000 0001 00EH 1001 0001 010H 0100 0111 011H 1110 0010 012H 1010 0001 013H 0100 0111	005H	0100 0111
008H 0100 0111 009H 1110 0001 00AH 0010 1001 00BH 0000 0010 00CH 0111 0001 00DH 1000 0001 00EH 1001 0001 010H 0100 0111 011H 1110 0010 012H 1010 0001 013H 0100 0111	006H	1110 0000
009H 1110 0001 00AH 0010 1001 00BH 0000 0010 00CH 0111 0001 00DH 1000 0001 00EH 1001 0001 010H 0100 0111 012H 1010 0001 013H 0100 0111	007H	0110 0000
00AH 0010 1001 00BH 0000 0010 00CH 0111 0001 00DH 1000 0001 00EH 1001 0001 010H 0100 0111 011H 1110 0001 012H 1010 0001 013H 0100 0111	008H	0100 0111
00BH 0000 0010 00CH 0111 0001 00DH 1000 0001 00EH 1001 0001 010H 0100 0111 011H 1110 0010 012H 1010 0001 013H 0100 0111	009H	1110 0001
00CH 0111 0001 00DH 1000 0001 00EH 1001 0001 010H 0100 0111 011H 1110 0010 012H 1010 0001 013H 0100 0111	00AH	0010 1001
00DH 1000 0001 00EH 1001 0001 010H 0100 0111 011H 1110 0010 012H 1010 0001 013H 0100 0111	00BH	0000 0010
00EH 1001 0001 010H 0100 0111 011H 1110 0010 012H 1010 0001 013H 0100 0111	00CH	0111 0001
010H 0100 0111 011H 1110 0010 012H 1010 0001 013H 0100 0111	00DH	1000 0001
011H 1110 0010 012H 1010 0001 013H 0100 0111	00EH	1001 0001
012H 1010 0001 013H 0100 0111	010H	0100 0111
013H 0100 0111	011H	1110 0010
	012H	1010 0001
014H 1110 0011	013H	0100 0111
	014H	1110 0011
015H 0101 0000	015H	0101 0000

课程设计总结

课设中遇到的问题以及解决

• 问题:在微程序控制的模型机中,存在如果想实现复杂功能,例如循环求和,寄存器数据覆盖现象。

解决: 通过增加寄存器的数目,相应的拓展片选信号以及打入信号,即可增加更多自己所必须的寄存器用来存储更多的堆栈数据,从而实现复杂功能。

- 问题:节拍时序问题:安排多少个节拍以及每个节拍干什么事?解决:不管哪一种 CU 的设计方案都需要仔细考虑时序信号问题。因为一条指令的执行会分成许多个微操作,而这些微操作是有明显的先后顺序的。因此使用节拍发生器来使得单脉冲变成连续脉冲,避免多次点击脉冲不自动化。其次,尝试不同的节拍数目以及每个节拍的时钟控制谁。例如:在微程序控制中,分为四个节拍,第一节拍 clk1 给到μPC 使得能够在下地址形成部件的作用下产生 CROM 中要执行的微指令的指令地址;第二个节拍 clk2 给到 CROM 读出对应地址的首条微指令;第三个节拍 clk3,则给到 RAM,来读出数据进行处理,同时更新了 DataBus,然后第四个节拍 clk4,则是进行打入信号的控制。可以理解为,先把要处理的数据算出来放在总线上,然后再输出控制信号,把他们打入对应的寄存器/元件中。
- 问题:硬布线控制的 CU 之中,增加一条指令就会导致几乎大部分控制部件的线路发生改变。

解决: 采用分层控制的思想。例如, 把周期和节拍的 and 放在一层, 最后每个信号再一层, 这样添加指令只要做到有序更改, 不会出错。或提前把所有指令写完, 写出每个信号的表达式, 然后一次性完成。

- 问题:设计微程序时,发现部分微地址空间的指令读不出来解决:后来逐一排查,发现 DataBus 上指令正确,但是一到 IR 就会莫名一位变成 0,然后排查发现是 IR 寄存器里面有一根线没有连接好,导致都置为 0。
- 问题:微程序设计时 16 条指令满了怎么办? 解决: 当想要设计更多功能或者实现更复杂程序的时候,发现指令条数已经满了,为什么?因为操作码只有 4 位,或者说需要依靠四位操作码来寻址微指令的空间,故最多就 16 条。因此解决方案两种:
 - ① 增加操作码的位数,可以将操作码增加到 5 位或者 6 位,然后在对应的地址空间写微指令即可。缺点:单个指令对应的最多微操作变少,并且揭示了这次微程序课设的缺点,不能很好的反应指令中寻址方式、寄存器的作用。即:一条指令的实现,仅仅与你写的微程序有关,就是和操作码对应,没法灵活指定寻址方式或者寄存器。例如: LW RO 如果还想给 R1 load word,必须重写一条指令,因为没法做到普适性,因为没有使用指令的后几位。

② 复用、单独测试。即把想实现的指令都写出来,然后可以多写几个 CROM 以及 RAM,分别测试不同的指令,即在同一 CROM 空间,重复覆盖微指令。因为这种方法不用大改整体架构,属于无奈之举~

收获与体会

经过这次《计算机组成原理课程设计》的大实验锻炼,也算是自己从电路原理角度,实现了或者说模拟了一个计算机的原始模型机,包括控制器、运算器、存储器、IO 设备等。并且是两种控制器的实现方式。也通过这两个实验,让我从指令的相关设计到模型机的整体结构,再到控制器的不同实现有了一个完整的概念框图。

明白了两种实现方式的区别:

- ❖ 两种控制器实质性的差别,表现在处理指令各执行步骤的接续关系的方案和给出时序控制信号的方法完全不同,从而造成控制器的具体组成和运行原理、运行性能上的一些差异。
- ◆ 两种控制器组成的主要差异:微程序控制器中的控存变成这里的时序信号产生部件,还取消了微指令寄存器;原来的下地址形成部件变成了这里的节拍发生器;原来的微地址映射部件变成这里的操作码译码器;一些信号连接关系也有某些变化。

对于硬布线方式,由于控制信号众多,逻辑表达式比较多,导致设计硬件线路庞大复杂,调试困难,一旦完成设计,添加新的控制功能很困难,但是硬件线路很多,速度是其最大的优点。对于微程序方式,它具有设计规整、调试、维修以及更改、扩充指令方便的优点,易于实现自动化设计,已成为现代控制器的主流,但由于增加了一级控制存储器,指令的执行速度比组合逻辑控制器慢。

整个课程设计下来我的感受一句话概括就是:

知其然知其所以然

当这个课设从底层一条条线搭建起来的时候,必须每一层都很清晰。你要明白整体流程架构,才能不出方向错误。明白每一条指令的执行流程,才能编好微程序。明白每一个器件的工作原理以及设计方式,才能一层层搭建起这个模型机。

硬件的学习,对于计科人来说,计组绝对是重中之重。除此之外,计算机体系结构以及计算机系统原理都是相辅相成的。近几年硬件行业的势头也很猛,大有赶超软件之势。从最初的学习计算机系统,到底层硬件的融会贯通,硬件的发展很大程度上能影响上层算法、软件的发展,比如机器学习、深度学习就是得益于算力的提升。目前,国家和社会企业对于硬件的重视程度不断加大,学好硬件,不仅是一个好的就业、科研方向,也是为国家、社会贡献的一条途径。

反思不足:

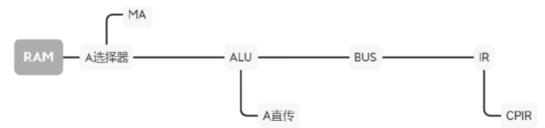
非常遗憾的一点就是在微程序中,指令冗余问题没有解决。还是像 MOV 一样,都是存数操作,按理说不管多少寄存器,一个 Mov 指令就行,但是没用上指令后面几位的有效信息,导致相似指令的编写有些重复。你有多个寄存器你就得编写多个 Load 和 Store 指令,再比如 add 你想要不同组合的寄存器想加就得编写多个 add 相似的指令。 如果有机会,应该把寻址方式、源目操作数都

作为控制信号考虑进去,可能大大减少指令冗余,会更像真实的计算机~

附录

附录1 数据通路图

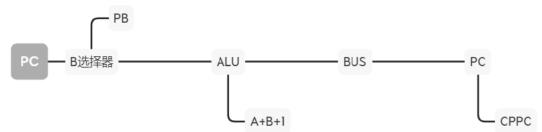
• 取指令



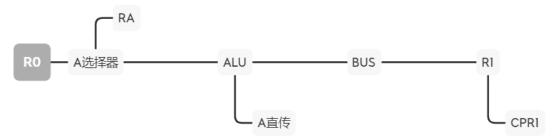
• 送指令地址



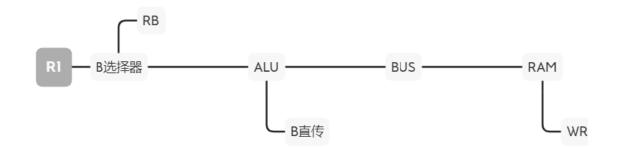
• 指令计数器加一



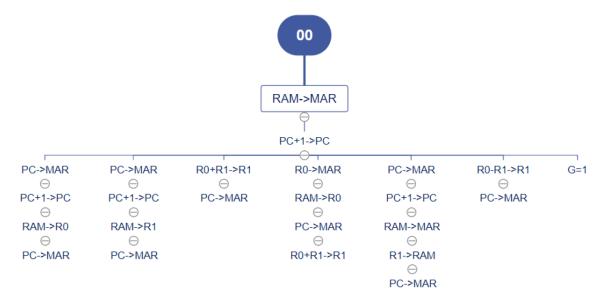
• R0->R1



• R1->RAM



附录2 微程序流程图



注: 每一条指令执行结束都会有一个 JP 操作

附录3 微程序

Lanca and an	Lacorer														
	微地址	μIR23μIR	μΙR21μΙR20μΙR19μΙR18μΙR17μ			μΙR11μΙR10μΙΙ	R9	μIR7	μIRe		ιIR3μΙ	R2µIR1µIR		进制化	
取指周期	_		M S3 S2 S1 S0 C0	A选择	B选择	输出分配			C	RD WR		转移方式		ROM#2	_
RAM-> IR	_	00	1 1111 0	10	00	100	0	0	0	10	0	001	3E	88	21
PC+1->PC	_	00	0 1001 0	00	01	011	0	0	0	00	0	001	12	16	01
QJP	02	00	0 0000 0	00	00	000	0	0	0	00	0	011	00	00	03
MOV1 05	_	00	1 1010 0		^-		-	_	_	22	_	007			
PC->MAR	10	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
PC+1->PC	_	00	0 1001 0	00	01	011	0	0	0	00	0	001	12	16	01
RAM->RO	12	00	1 1111 0	10	00	001	0	0	0	10	0	001	3E	82	21
PC->MAR	13	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
JP	14	00	0 0000 0	00	00	000	0	0	0	00	0	010	00	00	02
MOV2 01	R1						_								
PC->MAR	20	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
PC+1->PC	21	00	0 1001 0	00	01	011	0	0	0	00	0	001	12	16	01
RAM->R1	22	00	1 1111 0	10	00	010	0	0	0	10	0	001	3E	84	21
PC->MAR	23	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
JP	24	00	0 0000 0	00	00	000	0	0	0	00	0	010	00	00	02
ADD RO	R1 -> R1														
R0+R1->R	30	00	0 1001 1	01	10	010	0	0	0	00	0	001	13	64	01
PC->MAR	31	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
JP	32	00	0 0000 0	00	00	000	0	0	0	00	0	010	00	00	02
MOV3 R1															
PC->MAR	40	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
PC+1->PC		00	0 1001 0	00	01	011	0	0	0	00	0	001	12	16	01
RAM->MAR		00	1 1111 0	10	00	101	0	0	0	10	0	001	3E	8A	21
R1=>RAM	43	00	1 1010 0	00	10	000	0	0	1	01	0	001	34	20	51
PC->MAR	44	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
JP	45	00	0 0000 0	00	00	000	0	0	0	00	0	010	00	00	02
Jr	40	UU	0 0000 0	00	00	1 000	1 0	1 0	U	00	U	010	1 00	1 00	02
Late to the	Lorenza														D. Tell
微操作	微地址	μIR23μIR	μΙR21μΙR20μΙR19μΙR18μΙR17μ	5μIR14 μI	R13µIR12	μΙR11μΙR10μΙF	R9	μIR7	μIR6	µIR5µIR4	ιIR3μΙ	R2µIR1µIR0	十プ	、进制作	弋码
HALT															
G=1	50	00	0 0000 0	00	00	000	0	0	0	00	1	000	00	00	08
SUB RO(0		->R1													
R0-R1->R	_	0 0	0 0110 0	0 1	10	0 10	0	0	0	0 0	0	0 0 1	0C		01
PC->MAR	_	0 0											_	_	01
JP	62	0 0											00	00	02
MOV R1 A	lA_														
PC->MAR	40	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
PC+1->PC	41	00	0 1001 0	00	01	011	0	0	0	00	0	001	12	16	01
RAM->MAR	42	00	1 1111 0	10	00	101	0	0	0	10	0	001	3E	8A	21
R1->RAM	43	00	1 1010 0	00	10	000	0	0	1	01	0	001	34	20	51
PC->MAR	44	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
JP	45	00	0 0000 0	00	00	000	0	0	0	00	0	010	00	00	02
异或															
R0^R1->R	R170	00	1 0110 1	0 1	10	0 10	0	0	0	0 0	0	0 0 1	2D	64	01
PC->MAR		00										_			01
JP	72	00													02
MOV R1 B															
PC->MAR	40	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
PC+1->PC	_	00	0 1001 0	00	01	011	0	0	0	00	0	001	12	16	01
RAM->MAR		00	1 1111 0	10	00	101	0	0	0	10	0	001	3E	8A	21
R1->RAM	43	00	1 1010 0	00	10	000	0	0	1	01	0	001	34	20	51
PC->MAR	44	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
JP	45	00	0 0000 0	00	00	000	0	0	0	00	0	010	00	00	02
131	40	- 50	0 0000 0	00	- 00	000	V		U	00	J	010	00	00	02
MI I = C	Lancere														D. T.
微操作		μIR23μIR	μΙR21μΙR20μΙR19μΙR18μΙR17μ	5μIR14 μI	R13µIR12	μΙR11μΙR10μΙF	R9	μIR7	μIR6	μIR5μIR4	IR3μΙ	R2µIR1µIR0	十方	、进制作	弋码
JUMP	ВОН												_		
PC->MAR													34	1A	01
PC+1->PC				_	_	L							12	16	01
RAM (MAR)	_	00	1 1111 0	10	00	011	0	0	0	0	0	001	3E	86	01
RAM->RO	-	00		_	_	L							3E	82	21
RO->PC	ВЗН	00	1 1111 0	01	00	011	0	0	0	00	0	001	3E	46	01
													34	1A	01
PC->MAR														r	r]
	B3H B4H												00	00	02
	B4H COH												00		
JP	B4H COH	00				111			0		_	001	0C	6E	01
JP BEQ	B4H C0H UDGE	00			10 11	111 110	0	0	0		_	001		6E	
JP BEQ RO-R1->JI	B4H COH UDGE		1 1010 0	00			0	0		00	0	001	0C	6E 3C	01
JP BEQ RO-R1->JI JUDGE->R	B4H COH UDGE	00	1 1010 0	00	11	110	0	0	0	00	0	001 001	0C 34	6E 3C D6	01 01
JP BEQ RO-R1-> JI JUDGE-> R: R2 + PC	B4H COH UDGE	00	1 1010 0	00	11	110	0	0	0	00	0	001 001	0C 34 13	6E 3C D6	01 01 01
JP BEQ RO-R1->JI JUDGE->R R2 + PC- PC->MAR JP	B4H COH UDGE 12 ->PC	00	1 1010 0	00	11	110	0	0	0	00	0	001 001	0C 34 13	6E 3C D6	01 01 01 01
JP BEQ RO-R1->JI JUDGE->R R2 + PC PC->MAR JP	B4H C0H TUDGE 12 ->PC C3H	00	1 1010 0 0 1001 1	00 11	11 01	110 011	0	0	0	00	0	001 001 001	0C 34 13 34	6E 3C D6 1A	01 01 01 01 02
JP BEQ R0-R1->J1 JUDGE->R: R2 + PC - PC->MAR JP 自減 R0-1->R0	B4H C0H UDGE 22 ->PC C3H	00	1 1010 0 0 1001 1	00 11	11	110	0	0	0	00	0	001 001	0C 34 13 34	6E 3C D6 1A	01 01 01 01 02
JP	B4H C0H UDGE 12 ->PC C3H 80H 81H	00 00 00 00	1 1010 0 0 1001 1	00 11	11 01	110 011	0	0	0	00	0	001 001 001	0C 34 13 34	6E 3C D6 1A	01 01 01 01 02 01 01
JP BEQ R0-R1->JI JUDGE->R: R2 + PC->MAR JP ☐ ₩ R0-1->R0 PC->MAR JP	B4H C0H UDGE 12 ->PC C3H 80H 81H 82H	00	1 1010 0 0 1001 1	00 11	11 01	110 011	0	0	0	00	0	001 001 001	0C 34 13 34	6E 3C D6 1A	01 01 01 01 02
JP BEQ RO-R1-> JI JUDGE->R: R2 + PC - PC->MAR JP 自滅 RO-1->RO PC->MAR JP	B4H C0H UDGE 22 ->PC C3H 80H 81H 82H	00 00 00 00 00 00	1 1010 0 0 1001 1	00 11 01	11 01 00	110 011 001	0	0	0	00 00 00	0	001 001 001 001	0C 34 13 34 1F 34	6E 3C D6 1A 42	01 01 01 01 02 01 01 02
JP BEQ R0-R1->JI JUDGE->R: R2 + PC- PC->MAR JP R0-1->R0 PC->MAR JP MOV4 R0 PC->MAR	B4H C0H UDGE 12 ->PC C3H B0H 81H 82H 地址	00 00 00 00 00 00	1 1010 0 0 1001 1 011111 1 1010 0	00 11 01	11 01 00 00	110 011 001 001	0	0	0	00 00 00	0 0	001 001 001 001	0C 34 13 34 1F 34	6E 3C D6 1A 42 1A	01 01 01 01 02 01 01 02
JP BEQ R0-R1->JI JUDGE->R: R2 + PC - PC->MAR JP F1->R0 F1	B4H C0H UDGE 12 → >PC C3H 80H 81H 82H H±h: 90H	00 00 00 00 00 00	1 1010 0 0 1001 1 011111 1 1010 0 0 1001 0	00 11 01 00 00	11 01 00 00 01 01	110 011 001 001 101 011	0 0	0 0	0 0	00 00 00 00	0 0	001 001 001 001 001	0C 34 13 34 1F 34 34	6E 3C D6 1A 42 1A	01 01 01 01 02 01 01 02 01 01
JP BEQ RO-R1->JJ JUDGE->R: R2 + PC - PC->MAR JP MOV4 RO PC->MAR PC->MAR PC->MAR	B4H C0H UDGE 122 ->PC C3H 80H 81H 82H +地址 90H 91H 192H	00 00 00 00 00 00 00 00	1 1010 0 0 1001 1 011111 1 1010 0 0 1001 0 1 1111 0	00 11 01 00 00 10	11 01 00 00 01 01 00	110 011 001 001 101 011 101	0 0 0	0 0 0 0	0 0 0 0	00 00 00 00 00 00 10	0 0 0 0 0	001 001 001 001 001 001 001	0C 34 13 34 1F 34 12 3E	6E 3C D6 1A 42 1A 1A 16 8A	01 01 01 01 02 01 01 02 01 01 01 01
JP BEQ RO-R1->JJ JUGE=>R: R2 + PC - PC->MAR JP 自該 RO-1->RO PC->MAR JP MOV4 RO PC->MAR JP PC+1->PC RAM->MAR RO->RAM	B4H C0H UDGE 22 ->PC C3H 80H 81H 82H 144 h 90H 91H 92H 93H	00 00 00 00 00 00 00 00	1 1010 0 0 1001 1 011111 1 1010 0 0 1001 0 1 1111 0 1 1111 0	00 11 01 00 00 00 10 01	11 01 00 00 00 01 01 00 00	110 011 001 001 001 101 101 000	0 0 0 0 0 0	0 0 0 0 0	0 0 0 0 0 1	00 00 00 00 00 00 10 01	0 0 0 0 0 0 0	001 001 001 001 001 001 001 001	0C 34 13 34 1F 34 12 3E 3E 3E	6E 3C D6 1A 42 1A 1A 16 8A 40	01 01 01 02 01 01 02 01 01 02 01 01 01 01 51
JP BEQ RO-R1->JJ JUDGE->R: R2 + PC - PC->MAR JP MOV4 RO PC->MAR PC->MAR PC->MAR	B4H C0H UDGE 122 ->PC C3H 80H 81H 82H +地址 90H 91H 192H	00 00 00 00 00 00 00 00	1 1010 0 0 1001 1 011111 1 1010 0 0 1001 0 1 1111 0 1 1111 0 1 1010 0	00 11 01 00 00 10	11 01 00 00 01 01 00	110 011 001 001 101 011 101	0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 1 0 0	00 00 00 00 00 00 10	0 0 0 0 0 0	001 001 001 001 001 001 001	0C 34 13 34 1F 34 12 3E	6E 3C D6 1A 42 1A 1A 16 8A	01 01 01 01 02 01 01 02 01 01 01 01
JP BEQ RO-R1->JJ JUGE=>R: R2 + PC - PC->MAR JP 自該 RO-1->RO PC->MAR JP MOV4 RO PC->MAR JP PC+1->PC RAM->MAR RO->RAM	B4H C0H UDGE 22 ->PC C3H 80H 81H 82H 144 h 90H 91H 92H 93H	00 00 00 00 00 00 00 00	1 1010 0 0 1001 1 011111 1 1010 0 0 1001 0 1 1111 0 1 1111 0	00 11 01 00 00 00 10 01	11 01 00 00 00 01 01 00 00	110 011 001 001 001 101 101 000	0 0 0 0 0 0	0 0 0 0 0	0 0 0 0 0 1	00 00 00 00 00 00 10 01	0 0 0 0 0 0 0	001 001 001 001 001 001 001 001	0C 34 13 34 1F 34 12 3E 3E 3E	6E 3C D6 1A 42 1A 1A 16 8A 40	01 01 01 02 01 01 02 01 01 02 01 01 01 01 51

微操作	微地址	μIR23μIR	μΙR21μΙR20μΙR19μΙR18μΙR17μ	5μIR14 μ	IR13µIR12	μΙR11μΙR1ΟμΙR	19	μIR7	μIR6	μIR5μIR4	ιIR3μΙ	R2µIR1µIR0	十プ	\进制作	弋码
MOV XX R	3														
PC->MAR	30H	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
PC+1->PC	31H	00	0 1001 0	00	01	011	0	0	0	00	0	001	12	16	01
RAM->R3	32H	00	1 1111 0	10	00	000	1	0	0	10	0	001	3E	81	21
PC->MAR	33H	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
JP	34H	00	0 0000 0	00	00	000	0	0	0	00	0	010	00	00	02
MUTI									_						
R0*R1->R3	AOH	10	0 0000 0	01	10	000	1	0	0	00	0	001	80	61	01
PC->MAR	A1H	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
JP	A2H	00	0 0000 0	00	00	000	0	0	0	00	0	010	00	00	02
DIV	D1					010						201			
RO/R1->R1	DOH	01	0 0000 0	01	10	010	0	0	0	00	0	001	40	64	01
PC->MAR	D1H	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
JP	D2H	00	0 0000 0	00	00	000	U	U	U	00	0	010	00	00	02
ADD RO R		00	0 1001 1	01	00	000	,	1	0	00	0	001	10	41	01
RO+R3->R3 PC->MAR	E0 E1	00	0 1001 1 1 1010 0	01	00	000 101	0	0	0	00	0	001 001	13 34	41 1A	81 01
JP	E2	00	0 0000 0	00	00	000	0	0	0	00	0	010	00	00	02
MOV R3 R0		00	0 0000 0	00	00	000	U	U	U	00	U	010	00	00	02
R3 - >R0	F0	00	1 1010 0	00	00	001	0	1	0	00	0	001	34	02	81
PC->MAR	F1	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
JP	F2	00	0 0000 0	00	00	000	0	0	0	00	0	010	00	00	02
SAL	12		0 0000 0	- 00	- 00	000	V	V	•	- 00	•	010	00	00	02
RO< <ro< td=""><td>DOH</td><td>01</td><td>0 0000 0</td><td>01</td><td>00</td><td>001</td><td>0</td><td>0</td><td>0</td><td>00</td><td>0</td><td>001</td><td>40</td><td>42</td><td>01</td></ro<>	DOH	01	0 0000 0	01	00	001	0	0	0	00	0	001	40	42	01
PC->MAR	D1H	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
JP	D2H	00	0 0000 0	00	00	000	0	0	0	00	0	010	00	00	02
J-							-	_							
微操作	微地址	птрозптр	UTDO1UTDOOUTD1OUTD1OUTD17	EUTD14 U	TD19HTD19	TD11TD10TF			TDC	UTDEUTD4	TDOUT	DOUTDANTE	1.2	느 되는 쓰리 7	D 771
1/2/A JAN 1 F	[[[[]]]]	PIRZSPIR.	μΙR21μΙR20μΙR19μΙR18μΙR17μ	ομικι4 μ	TV126TV17	μIR11μIR10μIF	(9	μIK7	hTKG	4714671H	11K2h1		一丁ノ	与进制在	T 作马
			世址 PC+1是顺序下一条指令	5μ1K14 μ	IKI3#IKI2	HIKIIHIKIOHIN	(9	μIR7	μIKC	41КЭНІК4	11K2h1	.RZ#IRI#IRO	T7	\进刊1	【吗
				00 00	01	101	0	μ <u>ι</u> κ/	0	00 00	11 R3μ1 0	001	34	1A	01
间接寻址	此时PC己	经间接的地	也址 PC+1是顺序下一条指令												
间接寻址 PC->MAR	此时PC已 EO	经间接的出 00	也址 PC+1是顺序下一条指令 1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
间接寻址 PC->MAR RAM->PC	此时PC己 EO E1	经间接的地 00 00	地 PC+1是顺序下一条指令 1 1010 0 1 1111 0	00	01	101 011	0	0	0	00 10	0	001 001	34 3E	1A 86	01 21
回接寻址 PC->MAR RAM->PC PC->MAR	此时PC已 E0 E1 E2 E3	经间接的出 00 00 00 00	世 PC+1是順序下一条指令 1 1010 0 1 1111 0 1 1010 0 0 0000 0	00 10 00 00	01 00 01 00	101 011 101 000	0 0 0	0 0 0 0	0 0 0	00 10 00 00	0 0 0	001 001 001 010	34 3E 34 00	1A 86 1A 00	01 21 01 02
回接寻址 PC->MAR RAM->PC PC->MAR JP	此时PC已 E0 E1 E2 E3	经间接的出 00 00 00	世 PC+1是順序下一条指令 1 1010 0 1 1111 0 1 1010 0 0 0000 0	00 10 00 00	01 00 01 00	101 011 101 000	0 0 0 0	0 0 0 0	0 0 0 0	00 10 00 00	0 0 0 0	001 001 001 010	34 3E 34	1A 86 1A 00	01 21 01 02
间接寻址 PC->MAR RAM->PC PC->MAR JP POP SP+1->SP	此时PC已经 E0 E1 E2 E3 80H 81H	经间接的均 00 00 00 00 00	世 PC+1是順序下一条指令 1 1010 0 1 1111 0 1 1010 0 0 0000 0 0 0000 0 1 1111 0	00 10 00 00 11	01 00 01 00	101 011 101 000	0 0 0	0 0 0 0	0 0 0 0	00 10 00 00 00	0 0 0 0	001 001 001 010 001 001	34 3E 34 00 00 3E	1A 86 1A 00	01 21 01 02 01 01
同接寻址 PC->MAR RAM->PC PC->MAR JP POP SP+1->SP SP->MAR RAM->PC	此时PC已 E0 E1 E2 E3 80H 81H 82H	经间接的均 00 00 00 00 00 00	世中 PC+1是順序下一条指令 1 1010 0 1 1111 0 1 1010 0 0 0000 0 0 0000 0 1 1111 0 1 1111 0	00 10 00 00 11 11 11	01 00 01 00 00 00 00	101 011 101 000 110 101	0 0 0 0	0 0 0 0	0 0 0 0	00 10 00 00 00 00	0 0 0 0	001 001 001 010 001 001 001	34 3E 34 00 00 3E 3E	1A 86 1A 00 CC CA 86	01 21 01 02 01 01 01 21
同接寻址 PC->MAR RAM->PC PC->MAR JP POP SP+1->SP SP->MAR RAM->PC PC+1->PC	E0 E1 E2 E3 80H 81H 82H 83H	经间接的均 00 00 00 00 00 00 00	世 PC+1是順序下一条指令	00 10 00 00 11 11 11 10	01 00 01 00 00 00 00	101 011 101 000 110 101 011	0 0 0 0	0 0 0 0	0 0 0 0	00 10 00 00 00 00 10	0 0 0 0	001 001 001 010 001 001 001 001	34 3E 34 00 00 3E 3E 12	1A 86 1A 00 CC CA 86 16	01 21 01 02 01 01 01 21 01
PC->MAR RAM->PC PC->MAR JP POP SP+1->SP SP->MAR RAM->PC PC+1->PC PC+1->PC PC->MAR	E0 E1 E2 E3 80H 81H 82H 83H 84H	经间接的均 00 00 00 00 00 00 00 00	世中 PC+1是順序下一条指令	00 10 00 00 11 11 11 10 00	01 00 01 00 00 00 00 01	101 011 101 000 110 101 011 011 101	0 0 0 0	0 0 0 0 0	0 0 0 0 0 0 0	00 10 00 00 00 00 10	0 0 0 0 0 0 0	001 001 001 010 001 001 001 001 001	34 3E 34 00 00 3E 3E 12 34	1A 86 1A 00 CC CA 86 16 1A	01 21 01 02 01 01 01 21 01 01
PC->MAR RAM->PC PC->MAR JP POP SP+1->SP SP->MAR RAM->PC PC+1->PC PC+1->PC PC->MAR	此时PC已: E0 E1 E2 E3 80H 81H 82H 83H 84H 85H	经间接的均 00 00 00 00 00 00 00 00 00	世 PC+1是順序下一条指令	00 10 00 00 11 11 11 10 00 00	01 00 01 00 00 00 00 01 01	101 011 101 000 110 101 011 011 101 000	0 0 0 0	0 0 0 0	0 0 0 0	00 10 00 00 00 00 10	0 0 0 0	001 001 001 010 001 001 001 001	34 3E 34 00 00 3E 3E 12	1A 86 1A 00 CC CA 86 16	01 21 01 02 01 01 01 21 01
PC->MAR RAM->PC PC->MAR JP POP SP+1->SP SP->MAR RAM->PC PC->MAR RAM->PC PC->MAR JP	此时PC已: E0 E1 E2 E3 80H 81H 82H 83H 84H 85H 此时PC是	空间接的地 の0 00 00 00 00 00 00 00 00 00	## PC+1是顺序下一条指令	00 10 00 00 11 11 10 00 00 00	01 00 01 00 00 00 00 01 01 01 SP存的是指	101 011 101 000 110 101 011 011 101 000 令的前一条	0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0	00 10 00 00 00 00 10 00 00 00	0 0 0 0 0 0 0 0	001 001 001 010 001 001 001 001 001 001	34 3E 34 00 00 3E 3E 12 34 00	1A 86 1A 00 CC CA 86 16 1A	01 21 01 02 01 01 01 01 01 01
阿接寻址 PC->MAR RAM->PC PC->MAR JP POP SP+1->SP SP->MAR RAM->PC PC+1->PC PC->MAR JP	此时PC已: E0 E1 E2 E3 80H 81H 82H 83H 84H 85H 此时PC是	空间接的山 00 00 00 00 00 00 00 00 00 0	世址 PC+1是顺序下一条指令	00 10 00 00 11 11 11 10 00 00 00 	01 00 01 00 00 00 00 01 01 01 00 SP存的是指	101 011 101 000 110 101 011 011 101 000 令的前一条	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	00 10 00 00 00 00 10 00 00 00	0 0 0 0 0 0 0 0	001 001 001 010 001 001 001 001 001 010	34 3E 34 00 00 3E 3E 12 34 00	1A 86 1A 00 CC CA 86 16 1A 00	01 21 01 02 01 01 01 01 01 02
阿接	此时PC已: E0 E1 E2 E3 80H 81H 82H 83H 84H 85H #85H #85H #86H	空间接的址 00 00 00 00 00 00 00 00 00 00 00 00 00	世址 PC+1是顺序下一条指令	00 10 00 00 11 11 11 10 00 00 00 00 -——————————	01 00 01 00 00 00 01 01 01 00 SP存的是指	101 011 101 000 110 101 101 011 011 101 000 令的前一条 101		0 0 0 0 0 0 0	0 0 0 0 0 0 0	00 10 00 00 00 10 00 00 00 00	0 0 0 0 0 0 0 0	001 001 001 010 001 001 001 001 001 001	34 3E 34 00 00 3E 3E 12 34 00 3E 17	1A 86 1A 00 CC CA 86 16 1A 00 CA	01 21 01 02 01 01 01 01 01 02
何接 子址 PC→MAR RAM→PC PC→MAR JP POP SP+1→SP SP→MAR RAM→PC PC+1→PC PC→MAR JP PUSH SP→MAR SP−1→SP PC+1→SP	此时PC已: E0 E1 E2 E3 80H 81H 82H 83H 84H 85H 86H 86H 86H 86H 86H 86H 86H 86H 86H 86	全间接的址 00 00 00 00 00 00 00 00 00 00 00 00 00	世 PC+1是順序下一条指令	00 10 00 00 11 11 11 10 00 00 00 - 条指令 11	01 00 01 00 00 00 00 01 01 01 00 SP存的是指	101 011 101 000 110 101 101 011 101 000 今的前一条 110 101			0 0 0 0 0 0 0 0 0	00 10 00 00 00 10 00 00 00 00 00	0 0 0 0 0 0 0 0	001 001 001 010 001 001 001 001 001 001	34 3E 34 00 00 3E 3E 12 34 00 3E 17 17 18	1A 86 1A 00 CC CA 86 16 1A 00 CA CC	01 21 01 02 01 01 21 01 01 02 01 01 51
何接	此时PC已: E0 E1 E2 E3 S0H S1H S2H S3H S4H S5H 此时PC是 60H 60H 62H 63H	空间接的址 00 00 00 00 00 00 00 00 00 00 00 00 00	世址 PC+1是顺序下一条指令	00 10 00 00 11 11 11 10 00 00 00 00 -——————————	01 00 01 00 00 00 01 01 01 00 SP存的是指	101 011 101 000 110 101 101 011 011 101 000 令的前一条 101		0 0 0 0 0 0 0	0 0 0 0 0 0 0	00 10 00 00 00 10 00 00 00 00	0 0 0 0 0 0 0 0	001 001 001 010 001 001 001 001 001 001	34 3E 34 00 00 3E 3E 12 34 00 3E 17	1A 86 1A 00 CC CA 86 16 1A 00 CA	01 21 01 02 01 01 01 01 01 02
回接	此时PC已: E0 E1 E2 E3 80H 81H 82H 83H 84H 85H 86H 86H 86H 86H 86H 86H 86H 86H 86H 86	全间接的址 00 00 00 00 00 00 00 00 00 00 00 00 00	世 PC+1是順序下一条指令	00 10 00 00 11 11 11 10 00 00 00 - 条指令 11	01 00 01 00 00 00 00 01 01 01 00 SP存的是指	101 011 101 000 110 101 101 011 101 000 今的前一条 110 101			0 0 0 0 0 0 0 0 0	00 10 00 00 00 10 00 00 00 00 00	0 0 0 0 0 0 0 0	001 001 001 010 001 001 001 001 001 001	34 3E 34 00 00 3E 3E 12 34 00 3E 17 17 18	1A 86 1A 00 CC CA 86 16 1A 00 CA CC	01 21 01 02 01 01 21 01 01 02 01 01 51
回接 手址 PC->MAR RAM->PC PC->MAR JP POP SP+1->SP SP->MAR RAM->PC PC+1->PC PC->MAR JP PUSH SP->MAR SP-1->SP PC+1->RA PC->MAR JP SP-1->RA SP-1RA SP-1RA SP-1RA SP-1RA SP-1RA SP-1-RA	此时PC已: E0 E1 E2 E3 S0H S1H S2H S3H S4H S4H S6H S1H S6H S6H S6H S6H S6H S6H S6H S6H S6H S6	学同接的出 の の の の の の の の の の の の の	世址 PC+1是順序下一条指令	00 10 00 00 11 11 11 10 00 00 00 00 00 11 11	01 00 00 00 00 00 00 01 01 01 00 00 00 0	101 011 101 000 110 101 101 011 101 000 今的前一条 101 110			0 0 0 0 0 0 0 0 0	00 10 00 00 00 00 10 00 00 00 00 00 00	0 0 0 0 0 0 0 0	001 001 001 001 001 001 001 001 001 001	34 3E 34 00 3E 3E 12 34 00 3E 12 34 34 34 34 34	1A 86 1A 00 CC CA 86 16 1A 00 CA CC 10	01 21 01 02 01 01 01 01 01 02 01 01 01 01
回接	此时PC已: E0 E1 E2 E3 80H 81H 82H 83H 84H 85H 此时PC是 60H 610H 62H 63H	学问接的出 の の の の の の の の の の の の の	世址 PC+1是順序下一条指令	00 10 00 00 00 11 11 11 10 00 00 00 - 条指令 11 11 00 00	01 00 01 00 00 00 00 00 01 01 01 00 00 0	101 011 101 000 110 101 101 011 101 000 令的前一条 101 110 000			0 0 0 0 0 0 0 0 0	00 10 00 00 00 00 10 00 00 00 00 00 00 0	0 0 0 0 0 0 0 0	001 001 001 001 001 001 001 001 001 001	34 3E 34 00 00 3E 3E 12 34 00 3E 12 34 00 3E 34 80	1A 86 1A 00 CC CA 86 1A 00 CC CA 1A CC CA 10 1A CC CC 10 1A	01 21 01 02 01 01 01 01 01 01 51
回接 手加 PC->MAR RAM->PC PC->MAR JP POP SP+1->SP SP->MAR RAM->PC PC+1->PC PC-MAR JP PUSH SP->MAR SP-1->SP PC+1->PC PC+NAR JP PC+1->PC SP-MAR SP-1-SP PC+1->RO RO SP-MAR PC-NAR PC-NAR PC-NAR PC-NAR PC-NAR	此时PC已: E0 E1 E2 E3 80H 81H 82H 83H 84H 85H 44H 60H 610H 642H 62H 64H	经间接的址 00 00 00 00 00 00 00 00 00 00 00 00 00	世址 PC+1是顺序下一条指令	00 10 00 00 11 11 10 00 00 00 - 条指令 11 11 00 00	01 00 00 00 00 00 01 01 01 00 00 01 01 0	101 011 101 000 110 101 011 011 011 000 令的前一条 101 110 000 000			0 0 0 0 0 0 0 0 0 0	00 10 00 00 00 700 10 00 00 00 00 00 00		001 001 001 010 001 001 001 001 001 001	34 3E 34 00 00 3E 3E 12 34 00 3E 17 12 34 80 34	1A 86 1A 00 CC CA 86 16 1A 00 CA CC 110 1A	01 21 01 02 01 01 01 01 01 02 01 01 01 01 01
回接	此时PC已: E0 E1 E2 E3 S0H S3H S2H S3H S5H S5H S6H S6H S6H S6H S6H S6H S6H S6H S6H S6	学问接的出 の の の の の の の の の の の の の	世址 PC+1是順序下一条指令	00 10 00 00 00 11 11 11 10 00 00 00 - 条指令 11 11 00 00	01 00 01 00 00 00 00 00 01 01 01 00 00 0	101 011 101 000 110 101 101 011 101 000 令的前一条 101 110 000			0 0 0 0 0 0 0 0 0	00 10 00 00 00 00 10 00 00 00 00 00 00 0	0 0 0 0 0 0 0 0	001 001 001 001 001 001 001 001 001 001	34 3E 34 00 00 3E 3E 12 34 00 3E 12 34 00 3E 34 80	1A 86 1A 00 CC CA 86 1A 00 CC CA 1A CC CA 10 1A CC CC 10 1A	01 21 01 02 01 01 01 01 01 01 51
回接 手址 PC->MAR RAM->PC PC->MAR JP POP SP+1->SP SP->MAR JP POP PC->MAR JP POP PC->MAR JP PUSH SP->MAR SP-1->SP PC+1->RA JP PC+1->RA JP PC+1->RA PC->MAR JP SAR RO>>>RO SAR RO>->RO PC->MAR JP Load SP(I Load	此时PC已: E0 E1 E2 E3 S0H S3H S2H S3H S4H S5H S6H S6H S6H S6H S6H S6H S6H S6H S6H S6	学问接的出 の の の の の の の の の の の の の	世址 PC+1是順序下一条指令	00 10 00 00 11 11 11 10 00 00 分 11 11 00 00 00	01 00 00 00 00 00 00 01 01 00 00 SP存的是指 00 01 01	101 011 101 000 110 101 101 011 101 000 令的前一条 101 110 000 101			0 0 0 0 0 0 0 0 0 0 0	00 10 00 00 00 00 10 00 00 00 00 00 00 0		001 001 001 001 001 001 001 001	34 3E 34 00 00 3E 3E 12 34 00 3E 12 34 00 3E 12 34 00 00 00 00 00 00 00 00 00 0	1A 86 1A 00 CC CA 86 16 1A 00 CC CA CC 10 1A 1A 00	01 21 01 02 01 01 01 01 02 01 01 01 01 01 01 01 01 01 01 01
回接 手址 PC->MAR RAM->PC PC->MAR PC->	此时PC已: E0 E1 E2 E3 80H 81H 83H 84H 85H 此时PC是 60H 610H 62H 63H 64H A0H A1H A2H	学问接的出 の0 00 00 00 00 00 00 00 00 00	世址 PC+1是順序下一条指令	00 10 00 00 11 11 11 10 00 00 00 00 11 11	01 00 00 00 00 00 00 01 01 01 00 00 00 0	101 011 101 000 110 101 101 011 101 000 000 101 101 000 101				00 10 00 00 00 00 10 00 00 00 00 00 00 0		001 001 001 001 001 001 001 001	34 3E 34 00 00 3E 3E 12 34 00 3E 17 12 34 00 3E 38 19 10 10 10 10 10 10 10 10 10 10	1A 86 1A 00 CC CA 86 16 1A 00 CC CA 10 1A 1A 00 1A	01 21 01 02 01 01 01 01 01 02 01 01 01 01 01 01 01 01 01
回接 手加 PC->MAR RAM->PC PC->MAR JP POP SP+1->SP SP->MAR RAM->PC PC+1->PC PC+1->PC PC+1->PC PC+1->PC PC+1->PC PC->MAR JP PC+1->RA PC->MAR JP DC->MAR JP DC->MAR JP DC->MAR JP DC->MAR JP DC->MAR DC->MA	此时PC已: E0 E1 E2 E3 S0H 81H 82H S3H 84H 85H 此时PC是 60H 610H 64H A0H A1H A2H R2D 70 71	学问接的サ 00 00 00 00 00 00 00 00 00 0	世址 PC+1是順序下一条指令	00 10 00 00 11 11 10 00 00 00 - 条指令 11 11 00 00 00	01 00 01 00 00 00 01 01 00 SP存的是指 00 01 01 01 01	101 011 101 000 110 101 011 011				00 10 00 00 00 10 00 00 00 00 00 00 00 0		001 001 001 001 001 001 001 001 001 001	34 3E 34 00 00 3E 3E 12 34 00 3E 12 34 00 3E 15 16 17 18 19 19 19 19 19 19 19 19 19 19	1A 86 1A 00 CC CA 86 16 1A 00 CC CA 10 1A 10 1A 1A 16	01 21 01 02 01 01 01 02 01 01 01 02 01 01 01 01 01 01 01 01 01 01 01 01 01
回接	此时PC已: E0 E1 E2 E3 S0H S3H S2H S3H S5H S5H B6H 60H 610H 62H 64H A1H A2H R2D 70 71	学问接的サ の0 00 00 00 00 00 00 00 00 00	世址 PC+1是顺序下一条指令	00 10 00 00 11 11 10 00 00 	01 00 00 00 00 00 00 00 00 SP存的是指 00 00 01 01 01	101 011 101 000 110 101 011 011				00 10 00 00 00 00 10 00 00 00 00 00 00 0		001 001 001 001 001 001 001 001 001 001	34 3E 32 34 00 3E 12 34 00 3E 12 34 00 3E 17 12 34 00 3E 17 18 19 19 19 19 19 19 19 19 19 19	1A 86 1A 1A 00 CC CA 86 16 1A 00 CC CA 1A 1A 1A 1A 00 1A 1A 16 8C	01 21 01 02 01 01 01 01 02 01 01 01 01 01 01 01 01 01 01 01
回接 手加 PC->MAR RAM->PC PC->MAR JP POP SP+1->SP SP->MAR RAM->PC PC+1->PC PC+1->PC PC+1->PC PC+1->PC PC+1->PC PC->MAR JP PC+1->RA PC->MAR JP DC->MAR JP DC->MAR JP DC->MAR JP DC->MAR JP DC->MAR DC->MA	此时PC已: E0 E1 E2 E3 S0H 81H 82H S3H 84H 85H 此时PC是 60H 610H 64H A0H A1H A2H R2D 70 71	学问接的サ 00 00 00 00 00 00 00 00 00 0	世址 PC+1是順序下一条指令	00 10 00 00 11 11 10 00 00 00 - 条指令 11 11 00 00 00	01 00 01 00 00 00 01 01 00 SP存的是指 00 01 01 01 01	101 011 101 000 110 101 011 011				00 10 00 00 00 10 00 00 00 00 00 00 00 0		001 001 001 001 001 001 001 001 001 001	34 3E 34 00 00 3E 3E 12 34 00 3E 12 34 00 3E 15 16 17 18 19 19 19 19 19 19 19 19 19 19	1A 86 1A 00 CC CA 86 16 1A 00 CC CA 10 1A 10 1A 1A 16	01 21 01 02 01 01 01 02 01 01 01 02 01 01 01 01 01 01 01 01 01 01 01 01 01

附录 4 控制信号列表

Position .	en CRAK Disc				¥2 (执:	- Dien					
た	WI (取指周期) びまれる	WOV1	W 0V2	100/0011			减法(0110)	乘(0111)	除(1000)	左移(1001)	+40,000
ife 動信号	所有指令	BUVI	NUV2	ADD (0011)	NOV3(0100)	HALT (0101)	保佐(0110)	36(0111)	Bk (1000)	左移 (1001)	右移1010
a (Ram)	T2	T3	T3		T3						+
(R0)	12	15	13	T1	15		T1	T1	T1		+
(PC)	T1、T3	T1, T2	T1, T2	11	T1, T2		11	11	11		+
(R1)	11/ 10	115 12	115 12	T1	T4		T1	T1	T1	T1	T1
RO		T3			12			1.2	- 11	- 11	11
R1			T3	T1			T1			T1	T1
PC	T3	T2	T2		T2		1				
IR	T2										
MAR	T1	T1	T1		T1、T3						
	T2	T3	T3		T3						
3					T4						
ithC					T4						
						T1					
	1 (T1) 、1 (T2) 、0 (T3)			0 (T1)	1 (T1) 、0 (T2) 、1 (T3) 、1 (T4)		0 (T1)			T1	T1
	1 (T1) 、1 (T2) 、1 (T3)				1 (T1) 、1 (T2) 、1 (T3) 、1 (T4)		0 (T1)			T1	T1
	D (T1) 、1 (T2) 、0 (T3)				0 (T1) 、0 (T2) 、1 (T3) 、0 (T4)		1 (T1)				
	1 (T1) 、1 (T2) 、0 (T3)				1 (T1) 、0 (T2) 、1 (T3) 、1 (T4)		1 (T1)			T1	T1
	D (T1) 、1 (T2) 、1 (T3)				0 (T1) 、1 (T2) 、1 (T3) 、0 (T4)		0 (T1)				
CN	0 (T3)	0 (T2)	0 (T2)	1 (T1)	0 (T2)		0 (T1)	. (5.1)			
MULT								1(T1) 1(T1)	1(T1)		-
CPR2								1(11)	1(11)	1(T1)	
LM RM										1(11)	1(T1)
R.M.											1(11
	T1:PC-> MAR	T1: PC->MAR	T1: PC->MAR		T1: PC->MAR						
	T2:RAM->IR	T2: PC+1->PC	T2: PC+1->PC	T1: R0+ R1 -> R1	T2: PC+1->PC		T1: R0- R1 -> R1	T1: R0* R1 -> HILO	T1: RO/R1 -> HILO	T1: R1 << → R1	1: R1 >>
	T3: PC+1->PC	T3:RAM->R0	T3:RAM->R1		T3:RAN->NAR T4:R1->RAN						

附录 5 控制信号逻辑表达式

$$\begin{aligned} &\text{MA} = W_1 T_2 + W_2 T_3 (\text{MOV1} + \text{MOV2} + \text{MOV3}) \\ &\text{RA} = W_2 T_1 \cdot (\text{ADD} + \text{SUB} + \text{MUL} + \text{DIV}) \\ &\text{PB} = W_1 (T_1 + T_3) + W_2 (T_1 + T_2) (\text{MOV1} + \text{MOV2} + \text{MOV3}) \\ &\text{RB} = W_2 (\text{ADD} \cdot T_1 + \text{MOV3} \cdot T_4) + W^2 \cdot T_1 \cdot (\text{SUB} + \text{MUL} + \text{DIV}) \\ &\text{CPR}_0 = W_2 \cdot \text{MOV1} \cdot T_3 \cdot P \\ &\text{CPR}_1 = W_2 (\text{ADD} \cdot T_1 + \text{MOV2} \cdot T_3 + (SUB + SAR + SAL) \cdot T_1) P \\ &\text{CPPC} = \left(W_1 T_3 + W_2 T_2 (\text{MOV1} + \text{MOV2} + \text{MOV3}) \right) P \\ &\text{CPIR} = W_1 T_2 P \\ &\text{CPMAR} = \left(W_1 T_1 + W_2 (\text{MOV1} \cdot T_1 + \text{MOV2} \cdot T_1 + \text{MOV3} \cdot (T_1 + T_3)) \right) P \\ &\overline{RD} = \overline{W_1 T_2} + \overline{W_2 T_3} (MOV1 + MOV2 + MOV3) \\ &\overline{WR} = \overline{W_2 MOV3} \bullet T_4 \\ &C = W_2 MOV3 \bullet T_4 \\ &G = W_2 HALT \bullet T_1 \\ &M = W_1 (T_1 + T_2 + \overline{T_3}) + W_2 \bullet MOV1 \bullet (T_1 + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (T_1 + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (T_1 + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (T_1 + \overline{T_2} + T_3) + W_2 \bullet MOV1 \bullet (\overline{T_1} + \overline{T_2} + W_2 \bullet ADD \bullet \overline{T_1} + W_2 \bullet MOV3 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet (\overline{T_1} + \overline{T_2} + T_3) + W_2 \bullet MOV2 \bullet ($$

$$= W_{1}\overline{T_{3}} + W_{2} \bullet MOV1 \bullet \overline{T_{2}} + W_{2} \bullet MOV2 \bullet \overline{T_{2}} + W_{2} \bullet ADD \bullet \overline{T_{1}} + W_{2} \bullet MOV3$$

$$\bullet \overline{T_{2}}$$

$$S_{0} = W_{1}(\overline{T_{1}} + T_{2} + T_{3}) + W_{2} \bullet MOV1 \bullet (\overline{T_{1}} + T_{2} + T_{3}) + W_{2} \bullet MOV2 \bullet (\overline{T_{1}} + T_{2} + T_{3})$$

$$+ W_{2} \bullet ADD \bullet T_{1} + W_{2} \bullet MOV3 \bullet (\overline{T_{1}} + \overline{T_{4}} + T_{2} + T_{3})$$

$$= W_{1}\overline{T_{1}} + W_{2} \bullet MOV1 \bullet \overline{T_{1}} + W_{2} \bullet MOV2 \bullet \overline{T_{1}} + W_{2} \bullet ADD \bullet T_{1} + W_{2} \bullet MOV3$$

$$\bullet \overline{T_{1}} \bullet \overline{T_{4}}$$

$$CN = W_{1}\overline{T_{3}} + W_{2} \bullet MOV1 \bullet \overline{T_{2}} + W_{2} \bullet MOV2 \bullet \overline{T_{2}} + W_{2} \bullet ADD \bullet T_{1} + W_{2} \bullet MOV3$$

$$\bullet \overline{T_{2}}$$