计算机组成与系统结构实验-基于微程序控制的CPU设计

原创

Fushicho_XF 于 2018-06-11 21:48:34 发布 11412 收藏 75
分类专栏： 计算机组成实验 文章标签： 计算机组成实验 汇编

本文章主要是为了通俗的解释计算机组成实验的微程序汇编的实现方法：

实验目的

在掌握部件单元电路实验的基础上，初步了解如何基于微程序控制进行CPU设计。

实验软件

Dais-CMStudio

实验要求

1、学习教材3.1，3.2节，熟悉联机方式下的开发环境。

2、完成以下内容：

任务1：用汇编语言编写一个源程序，完成以下功能：

从I/O输入56h，并存入内存的18号单元；

从I/O输入78h，存入寄存器R0；

将两者相加，存入19号单元；

将19号单元的内容输出到I/O显示。

任务2：增加一条减法指令，指令格式如下：

指令格式：SUB R0, [addr]

机器指令码：11000000 XXXXXXXX XXXXXXXX

说明：R0 - [addr]→R0

任务3：增加一条寄存器加法指令，指令格式如下：

指令格式：ADR Rd, Rs

机器指令码：1110RdRs

说明：Rd + Rs→Rd

任务实现

任务1：因为软件内已经含有了5条基本指令助记符等的内容了，我们直接使用进行汇编即可（打开CMX16文件的MXJ6.ASM）

具体实现的汇编代码如下：

IN R0,IOL　　（将IO低位的数据存入R0通用寄存器）

STA 0018h,R0　（将R0中的数据存入内存的18号单元）

IN R0,IOL

ADD R0,0018h　（将0018h号单元的内存数据与R0相加存入R0）

STA 0019h,R0　（将R0中的数据存入内存的19号单元）

OUT IOH ,0019　（在IO高位地址输出0019h号单元的内存数据）

JMP START

任务2：我们需要在指令系统中先模仿ADD助记符添加一条SUB的助记符，由机器指令码11000000转化为16进制得到C0，因此的到SUB的助记符为：SUB　　R0,*　　C0　　3　　;R0 - [addr]→R0

接下来就是计算该机器指令码对应的微程序的微地址：

每条指令的微程序入口地址怎样确定？

位号：10 9 8 7 6 5 4 3 2 1 0

内容：1 1 IR7 IR6 IR5 0 0 0 0 0 0　（红色即为上面的机器指令码，将其带入即可）

因此计算的SUB给出的这个机器指令码要求的入口地址为0780（这是上面内容从后开始4位4位的转换为16进制的结果。入口地址就是当程序运行到这条指令时会根据你的助记符中的机器指令码结合这个计算方法自动跳到微地址为780的地方继续运行）

那么接下来就容易了，在程序本身存在ADD指令的情况下，我们只需要照抄就行，将ADD微地址入口为0680开始的微指令照抄到SUB的微地址入口0780，然后将最后一条微指令的CPU加法运算改为减法运算就行。(即M=0,S2=1,S1=0,S0=1）

微指令	数据读出	数据写入	总线规则	ALU 运算器	微变址	EM 地址	PC	uPC
BC F8 3F	PC 程序计数器	AR 地址寄存器	字传递	A 输出	禁止	PC 程序	增量	+1
7A FB DF	RAM 数据存储器	B 运算暂存器	奇→偶	A 输出	禁止	AR 数据		+1
BC F8 1F	PC 程序计数器	AR 地址寄存器	字传递	A 输出	禁止	PC 程序	增量	+1
7B FB DF	RAM 数据存储器	B 运算暂存器	奇→奇	A 输出	禁止	AR 数据		+1
BC C6 1F	ALU 运算器	AR 地址寄存器	字传递	B 输出	禁止	PC 程序	增量	+1
7A FB DF	RAM 数据存储器	B 运算暂存器	奇→偶	A 输出	禁止	AR 数据		+1
F8 F9 DF	通用寄存器	A 运算暂存器	奇→偶	A 输出	禁止	PC 程序	保持	+1
F1 6E 4D	ALU 运算器	通用寄存器	偶→奇	A 减 B	结尾变址	PC 程序	保持	0001

（第一条微指令的微地址为0780，这样应该明白了吧~）

然后只需要把任务1的ADD改成SUB进行操作就可以验证你写的对不对了~

任务3：

思考：首先要将Rd作为源寄存器将数据放入AXL中，再将Rs作为源寄存器源寄存器将数据放入BXL中，将（AX+BX）L的结果存入Rd中即可。现在在问题是如何处理将Rd和Rs分别作为源寄存器，然而对于Rd和Rs寄存器，我们只要写好指令系统里的助记符就行。（如果用C++来解释就类似一个函数入口吧）

这里需要补充知识点：通用寄存器就是实验箱上面的DX（R3,R2）和CX（R1，R0）

因此查看R0通用寄存器的数据就是看CX的后2位就行，其他同理。

然后根据题目要求的机械指令码为1110RdRs

Rs或Rd	选定的寄存器
00 01 10 11	R0 R1 R2 R3

具体实现见下：（如果上面看不懂请先继续看下去，在回来看估计会恍然大悟，我也是摸了挺久的~）

首先是添加指令系统的助记符：

助记符 操作数 指令码 指令长度

通过编码规则可得到 ADR R0 ,R0 E0 1 ;

ADR R0 ,R1 E1 1 ;

ADR R0 ,R2 E2 1 ;

ADR R0 ,R3 E3 1 ;

…（以下省略12条，类比上面写下去即可）

至于IN的助记符也需要修改为 IN R0,IOL 20 1；

IN R1,IOL 21 1；

IN R2,IOL 22 1；

IN R3,IOL 23 1；

（上面的助记符很多，但是他们单纯只是2条指令而已，不要误会~）

他们的指令码如何计算呢，下面举个李子就知道了：

下面详细解说一下指令码如何计算还有通过指令码如何得到该“函数”的微指令入口地址：

机器是通过计算机器指令码去到相应的微指令入口地址的，具体实现就是

位号：10 9　8　　7　6　　5 4 3 2 1 0

内容：1　1　IR7　IR6 IR5　 0 0 0 0 0 0 =>　07C0

(标彩色的部分就是机器指令码，而4，3组成Rs，2，1组成Rd，可以参考上面通用寄存器的表)

这个就是ADR　R0 ,R0　E0　　1　；这条助记符的微指令入口位置。

那么我们只需要从7C0这个位置开始修改就行。

至于ADR的其他助记符的微指令入口地址就不用计算了，机器会自动根据你的助记符R0~R3翻译得到相应的通用寄存器进行操作。（具体我也不清楚，看到他微指令是那么跑的，期初我修改了微指令4处想要实现IN的操作，可是写了发现它并不运行我写的后面三条，数据却是正确的存入了对应的通用寄存器）

至于微指令怎么写，这里先要提醒一点就是我们是通过XP来确定你的Rs和Rd的，因此在读通用寄存器数据时根据XP的0和1来改变你使用的是哪个通用寄存器（Rs或Rd）

注意：下面这个任务三的微指令表中目录行的第二个S2应该为S1（即M12下面那个是S1）

微址	M23	M22	M21	M20	M19	M18	M17	M16	代码	M15	M14	M13	M12	M11	M10	M9	M8	代码	M7	M6	M5	M4	M3	M2	M1	M0	代码	后续微址	说明
	E/M	IP	MWR	R/M	o2	o1	o0	OP		M	CN	S2	S2	S0	X2	X1	X0		XP	W	ALU	Iu	IE	IR	Icz	Ids			
07C0	1	1	1	1	1	0	1	0		1	1	1	1	1	0	0	1		0	1	1	1	1	1	1	1		+1	RS→BL
07C1	1	1	1	1	1	0	0	0		1	1	1	1	1	0	0	1		1	1	1	1	1	1	1	1		+1	RD→AL
07C2	1	1	1	1	0	0	0	1		0	1	1	0	0	1	1	0		0	1	1	0	1	1	0	1		0001	AL+BL→RD

大致是上面这样吧，反正我实验成功了，若是Rs或者Rd反了可以修改的是它们的Xp的0，1值（至于溢出测试也没有测，实验时间有限，心有余而力不足）

具体实现我懒的在写一次，增加一条指令无非就是修改2处，1：指令系统的助记符；2：计算出对应微地址然后写微指令即可！

如果有错误请指出，谢谢~