上海大学　计算机学院

《计算机组成原理实验》报告十三

姓名 胡才郁　　**学号** 20121034

时间 周四 9-11　　**机位** __　　**指导教师** 刘学民

实验名称: 建立指令流水系统

一、实验目的

1. 了解指令流水系统的设计方式。
2. 编制一条可以流水方式运行的指令。

二、实验原理

1. 硬部件的并行工作

在微指令编码上，如果几个子操作的微指令码中为低电平（有效）的都不相同，于是可以将这几个子操作的微指令码合并成一个微指令，微指令控制三部分硬件并行工作。

2. 指令流水执行

把 “使用不同硬件的操作可以同时工作”的概念推广到相继的两条指令之间，就形成“指令的流水线执行模式”。

这个模式下，同一时间有多条指令各自在不同的硬件中执行，而对同一条指令而言，不同时间顺序在不同的硬件中执行，很像在流水型生产线上的产品，不同时刻顺序在不同的工位上加工。这就是指令流水模式的名称来源。

要形成指令流水模式，每条指令都应该分成几个独立的子操作，当前趋指令的后几个子操作与后继指令的前几个子操作不使用同样的硬件时，系统就可设计成流水线方式。而现代计算机大都采用指令流水模式，但这个模式会使中断响应过程变得复杂，所以实时系统中多是有限地采用它。

3. 实验箱系统的指令流水硬件基础

实验箱系统中的很多操作可以在不同的硬件中同时执行，典型的是“取指令”的微操作，其微指令码为 CBFFFF，与大多数的微操作无关。在厂家给的默认指令系统中这个操作编在了每条指令的最后一个状态，即每条指令的操作完成后就取下一条指令。这是典型的“取指、执行、取指……”模式。即一条指令先被“取指”，再执行其他微操作，完成后再取下一条指令。如果一条指令的最后一个微操作与取指无关，就可以把二者合并成一个微指令，于是这个指令的最后一个微

操作与取下一条指令并行进行。对下一条指令而言，其“取指”与“其他操作”在不同硬件中顺序执行——指令二级流水。

4. 实验箱系统实现指令流水的技巧

若取指令操作（CBFFFF）与它前面的微操作码没有相同的位为 0，则这两个微操作码的“与”就是二者合并后的微指令。在程序中这条指令就会和它的后继指令形成二级流水模式。

三、实验内容

1. 实验任务一：分析流水指令集 insfile2.MIC

(1) 实验步骤

①打开下载好的 infile1 与 infile2 目录，在目录中选择并打开 infile1.mic 与 infile2.mic，对比其微指令的微程序部分。

(2) 数据记录、分析与处理

经过对两文件中不同微指令微程序部分的对比发现 ADD、SUB 以及一些逻辑运算命令类型的微指令大多都能被改造成流水线指令.

下面分别以 MOV A，R？和 MOV A，@R?两条指令的非流水方式与流水为例，进行分析。

对于 MOV A，R？指令而言，取值指令 CBFFFF 与前一条微指令 FFF7F7 微指令码中为低电平（有效）的都不相同，因此可以将其合并为 CBF7F7。所以在 insfile2 之中，只有一条微指令，并且此指令包含了两个微操作。

而对于 MOV A，@R?指令而言，取值指令 CBFFFF 与前一条微指令 D7BFF7 伪指令码中为低电平（有效）的部分相同，因此无法合并，所以在 insfile2 之中并无微指令合并的操作，其伪指令与 insfile1 之中相同。

表 1. MOV A, @R? 非流水指令系统与流水指令系统差异

insfile1				Insfile2		
MOV A, R?	T1	FFF7F7		MOV A, R?	T0	CBF7F7
	T0	CBFFFF				FFFFFF
		FFFFFF				FFFFFF
		FFFFFF				FFFFFF

表 2. MOV A, @R? 非流水指令系统与流水指令系统差异

insfile1				Insfile2		
MOV A, @R?	T2	FF77F7		MOV A, @R?	T2	FF77F7
	T1	D7BFF7			T1	D7BFF7
	T0	CBFFFF			T0	CBFFFF
		FFFFFF				FFFFFF

同理，对于 SUB A, @R? 指令而言，取值指令 CBFFFF 与前一条微指令 FFFE91 微指令码中为低电平（有效）的都不相同，因此可以将其合并为 CBFE91。如下表所示：

表 3. SUB A, @R? 非流水指令系统与流水指令系统差异

insfile1				Insfile2		
SUB A, @R?	T1	FF77EF		SUB A, @R?	T2	FF77EF
	T0	D7BFEF			T1	D7BFEF
		FFFE91			T0	**CBFE91**
		CBFFFF				FFFFFF

其他具体经过改造的命令如下表：

表 3. 改造后的流水指令

AND A, R?	ADD A, R?	ADD A, @R	ADD A, MM
ADD A, #II	ADDC A, R?	ADDC A, @R?	ADDC A, MM
ADDC A, #II	SUB A, @R?	SUB A, MM	SUB A, #II
SUBC A, R?	SUBC A, @R?	SUBC A, MM	SUBC A, #II
AND A, R?	AND A, @R?	AND A, MM	AND A, #II
OR A, R?	OR A, @R?	OR A, MM	MOV A, R?
MOV R?, A	IN	OUT	RR A
RL A	RRC A	RLC A	CPL A

(3) 实验结论

如果 1 条微指令中的相邻微程序之间二进制下 0 的位数不同，可以将这两条微程序合并，实现将其改造为流水指令的工作。

2. 实验任务二：改造实验十二中自己编制的指令集，并适当添加指令，使其中至少一条指令成流水方式。

(1) 实验步骤

①向上次实验的指令集添加新指令，并对其进行分析。

②对指令进行流水方式改造

(2) 数据记录、分析与处理

此实验任务处理方式与上一个任务处理方式类似，并将指令 A-W A, #*，OUTA、延时这 3 条改造成了流水方式。

若取值指令 CBFFFF 与前一条微指令的微指令码中为低电平（有效）的都不相同，可以进行合并，否则无法进行合并。

表 4. 上次实验中非流水方式与流水方式对比

非流水方式					流水方式			
FATCH	T0	00	CBFFFF		_FATCH_	T0	00	CBFFFF
		01	FFFFFF				01	FFFFFF
		02	FFFFFF				02	FFFFFF
		03	FFFFFF				03	FFFFFF
LD A, #*	T1	04	C7FFF7		LD A, #*	T1	04	C7FFF7
	T0	05	CBFFFF			T0	05	CBFFFF
		06	FFFFFF				06	FFFFFF
		07	FFFFFF				07	FFFFFF
A-W A, #*	T2	08	C7FFEF		**A-W A, #***	T1	08	C7FFEF
	T1	09	FFFE91			T0	09	**CBFE91**
	T0	0A	CBFFFF				0A	FFFFFF
		0B	FFFFFF				0B	FFFFFF
跳到 *	T1	0C	C6FFFF		跳到 *	T1	0C	C6FFFF
	T0	0D	CBFFFF			T0	0D	CBFFFF
		0E	FFFFFF				0E	FFFFFF
		0F	FFFFFF				0F	FFFFFF
OUTA	T1	10	FFDF9F		**OUTA**	T0	10	**CBDF9F**
	T0	11	CBFFFF				11	FFFFFF
		12	FFFFFF				12	FFFFFF
		13	FFFFFF				13	FFFFFF
延时	T3	14	FFFFFF		**延时**	T3	14	**CBFFFF**
	T2	15	FFFFFF			T2	15	FFFFFF
	T1	16	FFFFFF			T1	16	FFFFFF
	T0	17	CBFFFF			T0	17	FFFFFF

(3) 实验结论

如果 1 条微指令中的相邻微程序之间二进制下 0 的位数不同，可以将这两条微程序合并，实现将其改造为流水指令的工作。

3. 实验任务三：在自己编制的两个指令集中运行同一个程序，观测运行情况和效率。程序来源自定。

(1) 实验步骤

①使用实验任务二中编写的汇编程序，注意在编制过程中应多采用经过流水化改造后的指令。

②同时流水方式与非流水方式.DAT .MAC 的文件名一定要与.MIC 文件名相同，此时这三个文件分别组成一个指令系统，分别组成流水、非流水两套指令系统。

③在两套指令系统下运行，并比较程序运行快慢。

(2) 实验现象

使用流水指令系统与非流水指令系统，OUT 寄存器之中 22 33 55 都发生变化。而使用流水指令系统时，OUT 寄存器之中 22 33 55 的变化速度明显加快。

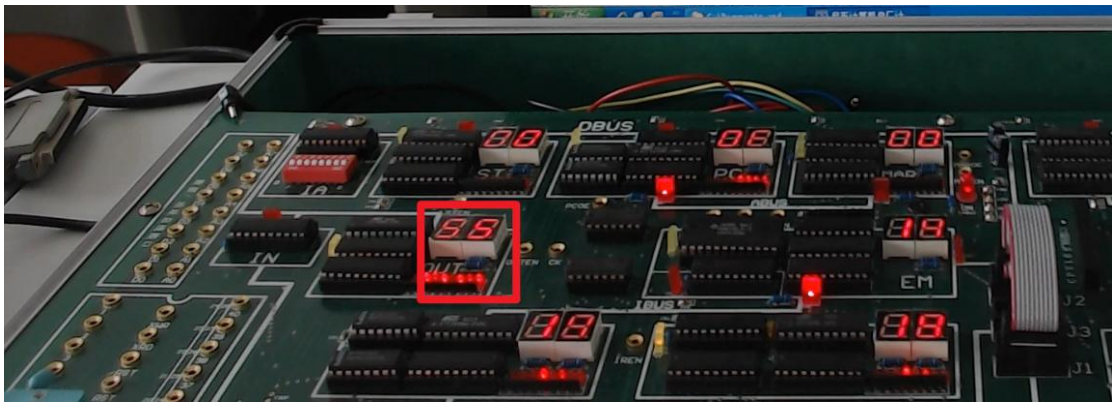

图 1 程序运行时 OUT 寄存器的示数变化

(3) 数据记录、分析与处理

此任务中非流水与流水指令系统的.MIC 文件同试验任务二。.ASM 文件中各语句及其功能如下表所示

表 5 .ASM 文件

LOOP:	LD A, #55H	设置起始地址源为 60H
	OUTA	**累加器 A 中内容输出至 OUT 寄存器**
	延时	
	延时	
	延时	

	延时	
	LD A, #22H	无条件跳转 T1 地址处
	OUTA	**累加器 A 中内容输出至 OUT 寄存器**
	延时	
	延时	
	延时	
	延时	
	LD A, #55H	
	A-W A, #22H	**累加器 A 中内容减去直接数 22H**
	OUTA	**累加器 A 中内容输出至 OUT 寄存器**
	延时	
	延时	
	延时	
	延时	
	跳到 LOOP	跳转至 LOOP 地址处
	END	

下面分析此程序如何实现本实验任务:

在主程序中，对于 55H 与 22H 而言，在 OUT 寄存器输出之前先将对应的立即数放在 A 累加器中，之后直接输出。而对于 33H 而言，在 OUT 寄存器输出之前将 55H 立即数放入 A 累加器，再使其减去立即数 22H，所得结果即为 33H，再将其输出至 OUT 寄存器。

并且在程序末尾进行跳转，实现了循环。每一轮循环之中依次输出 55H、22H、33H。

下面分析流水指令系统中 OUT 寄存器 22 33 55 的变化速度明显加快的原因:

相比较原非流水指令集，经过流水化改造后的指令集让同一个程序的运行速度加快了，其中减法 A-W A,由 3 条微程序优化为 2 条微程序，而输出 OUTA 由 2 条微指令缩短到了 1 条微程序，延时程序由 4 条微指令缩短到了 1 条微指令。因此变化速度明显加快。

(4) 实验结论

使用流水指令系统可以优化程序执行时间。

四、建议和体会

实验过程当中，如果只是观看视频，去理解每一步操作的原因，那么便很难记住对应的格式，后续的汇编程序操作将变得十分频繁。

经过本次实验，我掌握了如何将非流水指令系统改造为流水指令系统，深刻理解了程序执行的步骤。强化了我的汇编程序的编写能力及调试能力，受益匪浅。

五、思考题

问题：计组实验课临近尾声，请你对该课程的授课形式、实验内容等提出你的建议？

答：诚惶诚恐中迎来了计算机组成原理实验课的尾声，对每次课堂老师在微信群中的耐心答疑充满感激。这两个学期的学习伴随着我个人对计算机组成原理了解的步步深入，一次次的实验帮助我理解理论课堂上学习的内容，同时也是一次次对自己实践能力的验证。个人认为之前的授课形式——实验+报告已经非常合理，而这个学期的疫情导致没有办法亲自动手做实验，只能依据视频的结果反过来推理知识，的确有些可惜。

同时希望课程能提供更完整的实验指导书（带有答案），现有指导书中虽然已经详细的介绍了操作，但仍有许多问题没有提供标准答案，在学习过程中没有标准答案作为参考有一些遗憾。