重庆理工大學

实验报告

实验名称：基本的实验.

专业班级：

学　　号：

姓　　名：

联系电话：

指导老师：

实验时间：

重庆理工大学电气与电子工程学院

电气与控制工程实验中心

【成绩】

【教师签名】

【实验目的】

1.掌握循环操作指令的运用

2.掌握用汇编语言编写DSP程序的方法.

3.掌握TMS320C54x中的双操作数指令.

4.掌握TMS320C54x中的并行运算和长字指令。

【实验原理及内容】

1.循环操作.TMS320C54x具有丰富的程序控制与转移指令,利用这些指令可以执行分支转移,循环控制以及子程序操作。本实验要求编写一程序完成分支转移,循环控制以及子程序操作。完成 $y=\sum_{i=1}^{5} x_i$ 的计算,这个求和计算可以通过一个循环操作指令BANZ来完成。BANZ的功能是当辅助寄存器的值不为0时转移到指定标号执行。

2.双操作数乘法:TMS320C54x片内的多总线结构,允许在一个机器周期内通过两个16位数据总线(C总线和D总线)寻址两个数据和系数。双操作数指令是用间接寻址方式获得操作数的并且只能用AR2到AR5的辅助寄存器。双操作数指令是用间接寻址方式获得操作数,双操作数指令占用较少的程序空间,而获得更快的运行速度.

3.并行运算:TMS320C54x片内有1条程序总线,3条数据总线和4条地址总线这3条数据总线(CB,DB,和EB)将内部各单元连接起来,其中,CB和DB总线传送从数据存储器读出的操作数EB总线传送写到存储器中的数据。并行运算就是同时利用D总线和E总线。并行指令有并行加载和乘法指令,并行加减和存储指令,并行存储和乘法指令以及并行存储和加/减法指令4种

4.长字运算:长操作数指令中的一个重要问题是,高16位和低16位操作数在存储器中如何排列。一般情况下,高16位操作数放在存储器中的低地址单元,低16位操作数放在存储器中的高地址单元。例如 12782345H.它在存储器中的存入方式是

(0060H)=1278H(高字) (0061H)=2345H(低字)

【实验设备】

一台装有CCS软件的计算机

【实验方案及步骤】

一、循环操作：

1. 编译前 ①启动 CCS，选择模拟 CPU (C6416 Devie Simulator)

②打开工程文件，打开".asm"文件和".cmd"文件；

2. 编译和单步调试：

①Build工程，编译无错误后，选择"Load Progrom"命令装载程序，然后打开存储器数据窗口以及CPU寄存器窗口。

②单步运行程序，观察寄存器和存储器的变化，理解循环指令的三条语句

③修改重复指令实现循环操作，重新Build工程，装载程序后得到与之前相同的结果，记录原始语句，修改后的语句，并记录运行后变量 x, y 的地址和数值。

二、双操作数乘法：

1. ①打开工程文件，Build工程，装载程序，运行观察结果。

2. ②改为单操数乘加指令来实现

①先将其中一个操作数装入T寄存器中 ②去掉原双操作数MAC指令中装入T寄存器的那个操作数；③用BANZ指令重新构造循环。

3. 重新运行程序后记录观察量 x, y, z 的地址和数值

三、并行运算：

1. 打开工程文件，Build工程，装载程序运行观察结果。

2. 改用非并行指令来实现：

①去掉分隔符，将一条并行指令变成2条指令；②修改为使用累加器的高位。

3. 观察记录寄存器，存储器的变化。

四、长字运算

1. 打开工程文件，Build工程文件，装载程序，运行观察结果

2. 改为单字行指令来实现：①每一条长字指令改为2条单条指令实现 ②左移16位装入高位，③用ADDS加法指令实现低位装入。

3. 运行后记录哪些寄存器，存储器的数值会发生变化。

五 分析实验结果完成实验报告。

【实验电路图】

【实验数据处理及分析】

1. 循环操作.

程序运行后，辅助寄存器 AR2 的值随着循环次数的增加而减少，因为在循环开始前装入了循环次数 4 在 AR2 中，循环结束后又用 BANZ 指令使用 AR2 的值减一后又进行下一次循环，直到 AR2 中的值为零时到此出循环；而 AR3 则存放每一次循环的累加值，因此是递增的。

2. 并行运算

将一条并行指令 "ST A, 4AR5 || LD 4AR2+, B" 改为非并行指令，不仅需要去掉符号符 "||" 还需把使用累加器的高位的形式表示出来变成 "STH A, 4AR5"

3. 长字运算

将长字指令 "DLD 4(X), A"
"DADD 4(Y), A."
"DST A, 4(Z)"

改为单字指令.

```
LD 4(X), 16, A
ADDS 4(X+1), A
AND 4Y, 16, B
ADDS 4(Y+1), B, A
STH A, 4Z
STL A, 4(Z+1)
```

【实验结论】

1. 将循环操作指令改为重复指令后，虽然实现逻辑相似，但代码结构会更加清晰，易于阅读和维护。

2. 将双操作数乘法指令改为单操作数乘法指令可以简化某些计算场景。例如需要计算一个数的平方时，单操作数乘法可以直接完成，而无需额外的寄存器来暂存其中一个操作数。但对于一般的乘法运算，双操作数乘法指令更适合。

3. 将并行运算指令改为非并行运算指令，会增加流水线中的依赖关系，进而影响的执行效率。

4. 使用长操作数指令时，一般高16位操作数放在存储器中的低地址单元，低16位操作数放在存储器中的高地址单元。

【思考题】

1. 在§3.1中，循环控制是在循环体结束处通过指令"BANZ loop, *AR2-"检测循环变量实现的，如改为在循环体开始处检测循环变量如何修改？

答：可以在"STL A, *AR(4)"指令前加一个"end loop"标签，然后在循环体的上一条指令改为"BNZ *AR2, end loop"，当AR2不为零时就会结束循环。

2. 双操作数指令和并行指令说明了TMS320C54x的总线结构有何特点？

答：① 多总线结构，允许在一个机器周期内通过两个16位数据总线（总线和D总线）寻址两个数据和系数；

② 片内有1条程序总线，3条数据总线和4条地址总线，这3条数据总线（CB, DB和EB）将内部各单元连接在一起。其中，CB、DB总线传送从数据存储器读出的操作数，EB总线传送写到存储器中的数据。

③ 还有数据预取机制、高效的内存访问、指令与数据分离等特点。

实验一 基本指令实验（数据记录）

3.1 循环操作

	循环语句（原始语句）	用重复语句实现（修改后的语句）
语句分析	STM #4, AR2	RPT #4
	loop: ADD *AR3+, A	ADD *AR3+,A
	BANZ loop, *AR2-	

		寄存器名称	各次循环开始时寄存器数值变化（用 16 进制表示）					
			1	2	3	4	5	结束后
数据分析	循环变量	AR2	0x0004	0x0003	0x0002	0x0001	0x0000	0xFFFF
	X 寻址	AR3	0x0070	0x0071	0x0072	0x0073	0x0074	
	累加器	A	0x000A	0x001E	0x0021	0x0025	0x003E	

	变量	地址	数值
运行结果	x	0x0070	0x000A 0x0014 0x0003 0x0004 0x0019
	y	0x0075	0x003E

3.2 双操作数乘法

	双操作数乘加指令（原始语句）	用单操作数乘加指令实现（修改后的语句）
语句分析	RPTZ A, #9	LD #0, A
	MAC *AR3+, *AR4+, A	STM #9, AR2
	STH A, *(z)	LOOP: LD *AR3+,T
	STL A, *(z+1)	MAC *AR4+,A
		BANZ LOOP, *AR2-

	变量	地址	数值
运行结果	A	0x0060	0x000A 0x000B 0x000C 0x000D 0x000E 0x000F 0x0010 0x0011 0x0012 0x0013
	X	0x006A	0x000A 0x000B 0x000C 0x000D 0x000E 0x000F 0x0010 0x0011 0x0012 0x0013
	z	0x0074	0x0000 0x0889

3.3 并行运算

语句分析		说明：哪些寄存器(名称)，或存储器(地址)？数值怎么变化？	
并行语句（原始语句）	ST A, *AR5 || LD *AR2+, B	0x0072	0x114A
		AR2	0x0073->0074H
		B	0x0000000000->0x0010200000
不用并行语句实现（修改后的语句）	STH A, *AR5	0x0072	0x114A
	LD *AR2+,16, B	AR2	0x0073->0074H
		B	0x0000000000->0x0010200000

3.5 长字运算

语句分析		说明：哪些寄存器(名称)，或存储器(地址)？数值怎么变化？	
长字指令（原始语句）	DLD *(x), A	A	0x0000000000->0x0016782345
	DADD *(y), A	A	0x0016782345->0x002698268C
	DST A, *(z)	0074H	0x0000->0x2698
		0075H	0x0000->0x268C
单字指令（修改后的语句）	LD *(x),16, A	A	0x0000000000->0x0016780000
	ADDS *(x+1), A	A	0x0016780000->0x0016782345
	ADD *(y),16, A	A	0x0016782345->0x0026982345
	ADDS *(y+1), A	A	0x0026982345->0x002698268C
	STH A,*(z)	0x0074	0x0000->0x2698
	STL A,*(z+1)	0x0075	0x0000->0x268C