第5章 TMS320C54x汇编语言程序设计

C54x提供两种编程语言，即汇编语言和C/C++语言。对于完成一般功能的代码，这两种语言都可使用，目前使用较多的是C/C++语言，但对于一些运算量很大的关键代码，最好采用汇编语言来完成，以提高程序的运算效率。对于初学者，最好先以汇编语言作为入门编程工具来学习，这样易于深刻理解DSP的工作原理。因此，汇编语言程序设计是应用软件设计的基础。下面结合例子简要介绍TMS320C54x汇编语言源程序设计的基本方法。

5.1 汇编语言概述

TMS320C54x汇编语言源程序由源语句组成。这些语句可以包含汇编语言指令、汇编伪指令和注释。程序的编写必须符合一定的格式，以便汇编器将源文件转换成机器语言的目标文件。汇编语言程序以.asm为扩展名，可以用任意的编辑器编写源文件。一条语句占源程序的一行，长度可以是源文件编辑器格式允许的长度，但汇编器每行最多读200个字符。因此，语句的执行部分必须限制在200个字符以内。

C54x的指令系统包含助记符指令和代数指令两种形式。助记符指令是一种采用助记符号表示的类似于汇编语言的指令；代数指令是一种比汇编语言更高级，类似于高级语言的代数形式指令。两种指令具有相同的功能，本章着重介绍助记符指令。

C54x的助记符指令由操作码和操作数两部分组成。在进行汇编以前，操作码和操作数都用助记符表示。助记符指令源语句的每一行通常包含4个部分：标号区、操作码区、操作数区和注释区。

源代码的书写有一定的格式，初学者往往容易忽视。每一行代码分为三个区：标号区、指令区和注释区。助记符指令语法格式为：

```
[标号][:]  操作码  [操作数]  [;注释]
```

如：

```
LD #0FFh,  A  ;将立即数0FF传送至A
```

语句的书写的基本规则如下：

(1) 所有语句必须以标号、空格、星号或分号(*或;)开始，语句各部分之间必须用空格或Tab分开。

(2) 一般区分大小写，除非加编译参数忽略大小写。

(3) 所有包含汇编伪指令的语句必须在一行完成指定,如果源程序很长,需要书写若干行,可以在前一行用反斜杠字符(\)结束,余下部分接着在下一行继续书写。下面对指令中各部分的作用及书写规则进行介绍。

1. 标号

所有汇编指令和大多数汇编伪指令都可以选用标号,主要用于定义变量、常量、程序标识时的名称,以供本程序或其他程序调用。注意:

(1) 标号为可选项,若使用标号,必须顶格写,其后的冒号":"可任选;若不使用标号,则语句的第一列必须是空格、星号或分号。

(2) 标号由字母、数字以及下画线或美元符号等组成,最多可达 32 个字符。

(3) 标号分大小写,且第一个字符不能是数字。例如

```
Start: .word 0Ah,3,7
```

Start 即为顶格写的标号,用于定义常量,供程序调用。

2. 指令区

指令区紧跟在标号区的后面,以空格或 Tab 格开。如果没有标号,也必须在指令前面加上空格或 Tab,不能顶格。指令区由操作码和操作数组成。操作码是指令代码,操作数是指令中参与操作的数值或汇编伪指令定义的内容。

如:

```
Start: LD #0FFh,  A  ;将立即数 0FF 传送至 A
```

其中,Start 为标号,LD 为操作码,0FFh 为源操作数,A 为目的操作数,此条指令完成的功能是将立即数 0FF 传送至累加器 A。

注意:

(1) 操作数之间必须用逗号",")分隔;

(2) 操作数可以是常数、符号或表达式;

(3) 操作数中的常数、符号或表达式可用来作为地址、立即数或间接地址;

(4) 操作数可以有前缀,可以使用#、* 和@符号作为操作数的前缀。

① 用#作为前缀

使用#作为前缀,汇编器将操作数作为立即数处理。即使操作数是寄存器或地址,也将作为立即数。如果操作数是地址,汇编器将把地址处理为一个数值,而不使用地址的内容。

例如:

```
Label:  ADD  #99, B
```

操作数# 99 是一个立即数。

② 用 * 作为前缀

使用 * 作为前缀,汇编器将操作数作为间接地址,即把操作数的内容作为地址。例如:

```
Label:  LD  * AR3, B
```

操作数 * AR3 指定一个间接地址。该指令将引导汇编器找到寄存器 AR3 的内容作为地址，然后将该地址中的内容装入指定的累加器 B 中。

③ 用@作为前缀

使用@作为前缀，汇编器将操作数作为直接地址，即操作数由直接地址码赋值。例如：

```
Label:  LD  @x, A
```

将直接地址 x 中的内容装入指定的累加器 A 中(绝对寻址)；或者 DP、SP 的值加上 x 作为地址，将其内容放入 A(直接寻址)。

3. 注释区

注释区用来说明指令功能的文字，便于用户阅读。注释区是可选项，在标号区、程序区之后，可单独一行或数行。如果注释在第一列开始时，前面必须标上 * 或“;”，在其他列开始的注释前面必须以分号开头，另外还有专门的注释行，以 * 开头，必须顶格开始。

5.2 寻 址 方 式

寻址方式是指寻找指令所指定的参与运算的操作数的方法。根据程序的要求采用不同的寻址方式，可以有效地缩短程序的运行时间和提高代码执行效率。C54x 芯片的寻址方式可以分为两类：数据寻址和程序寻址，其中数据寻址有 7 种方式，分别介绍如下。

1. 立即寻址

指令中含有执行指令所需的操作数，常用于初始化。操作数紧随操作码存放在程序存储器中。立即寻址的特点是指令中含有一个固定的立即数，运行速度较快，但需占用程序存储空间，且数值不能改变。

立即寻址一般用于表示常数或对寄存器初始化。需要注意的是，在立即寻址的指令中，应在数值或符号前面加一个 #，表示是一个立即数，以区别于地址。例如：

```
LD  #FF80h,A
```

图 5.1 程序存储器

如图 5.1 所示，立即数 FF80 是与操作码一起存在程序存储器中的，指令执行后，将立即数 FF80 加载到累加器 A 中。

2. 绝对寻址

由指令提供一个 16 位的操作数地址，利用 16 位地址来寻址操作数的存储单元。16 位地址的表示形式可以是 16 位符号常量，如 89AB、1234；也可以是地址标号，如 TABLE。绝对寻址的特点是指令中包含一个固定的 16 位地址，能寻址所有数据存储空间，但运行速度慢、需要较大的存储空间，因此，常用于对速度要求较低的场合。

绝对寻址有以下 4 种类型。

(1) 数据存储器地址(dmad)寻址，用于确定操作数存于数据存储单元的地址。

例如：

```
MVKD  EXAM1, * AR5
```

EXAM1 为数据存储器的 16 位地址 dmad 值，指令将数据存储器 EXAM1 地址单元中的数据传送到 AR5 寄存器所指向的数据存储单元中。

(2) 程序存储器地址(pmad)寻址，用于确定程序存储器中的一个地址。

例如：

```
MVPD  TABLE, * AR2
```

TABLE 是程序存储器的 16 位地址 pmad 值，指令将程序存储器 TABLE 地址单元中的内容传送到 AR2 寄存器所指向的数据存储单元中。

(3) 端口(PA)寻址，用一个符号或一个数字来确定外部 I/O 端口的地址。

例如：

```
PORTR  FIFO, * AR5
```

FIFO 为 I/O 端口地址 PA，指令将一个数从端口为 FIFO 的 I/O 口传送到 AR5 寄存器所指向的数据存储单元中。

(4) *(1k)寻址，使用一个指定数据空间的地址来确定数据存储器中的一个地址。

例如：

```
LD  * (PN), A
```

把地址为 PN 的数据单元中的数据装到累加器 A 中，这种寻址可用于支持单数据存储器操作数的指令。

注意：*(1k)寻址的指令不能与循环指令(RPT，RPTZ)一起使用。

3. 累加器寻址

以累加器的内容作为地址去访问程序存储单元，即将累加器中的内容作为地址，用来对存放数据的程序存储器寻址。该寻址方式主要用于完成程序存储空间与数据存储空间之间的数据传输。

注意，只能使用累加器 A 寻址程序空间，Smem 用来寻址数据空间。

例如：

```
READA  x  ;将累加器 A 的内容作为地址读程序存储器，并存入数据存储单元 x 中
```

4. 直接寻址

寻址地址为 DP 或 SP 的值加上指令提供的偏移量进行寻址，所要寻址的数据存储器 16 位地址由基地址和偏移地址构成。数据存储器的低 7 位地址为偏移地址，数据页指针 DP 和堆栈指针 SP 提供基地址，使用 DP 还是 SP 由 CPL 值来决定。当 CPL=0 时，数据存储器 16 位地址由 DP 和偏移地址 dmad 构成；当 CPL=1 时，数据存储器 16 位地址由 SP 加偏移地址 dmad 构成。

直接寻址的标识为：在变量前加“@”，如@x；或者在偏移量前加“@”，如@5。

指令格式如图 5.2 所示。

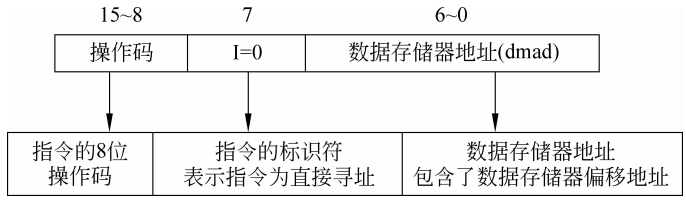

图 5.2 指令格式

直接寻址的优点是每条指令只需要一个字。因此,主要用于要求运算速度较快的场合。

例如:

```
(1)     RSBX   CPL      ;CPL=0
        LD   @x,A       ;DP+x 的低 7 位地址的内容给累加器 A
(2)     SSBX   CPL      ;CPL=1
        LD   @x,A       ;SP+x 的低 7 位地址的内容给累加器 A
```

5. 间接寻址

间接寻址是根据辅助寄存器(AR0~AR7)给出的 16 位地址进行寻址的,每一个辅助寄存器都可以用来寻址 64K 字数据存储空间中的任何一个单元。两个辅助寄存器算术运算单元(ARAU0 和 ARAU1)可以根据辅助寄存器的内容进行操作,完成 16 位无符号数算术运算。

间接寻址的特点是通过辅助寄存器和辅助寄存器指针来寻址数据空间存储单元,并自动实现增量、减量、变址寻址、循环寻址,共有 16 种修正地址的方式,主要用于需要按固定步长寻址的场合。

例如:

```
LD   *AR1,A   ;以辅助寄存器 AR1 所指的 16 位地址为地址,取该地址
              ;的内容,传送给累加器 A
```

6. 存储器映像寄存器寻址

存储器映像寄存器寻址是一种不考虑 DP 和 SP 为何值、以 0 为基地址来访问 MMR 的寻址方式,主要用于修改存储器映像寄存器的内容,但不影响 DP 或 SP 的值。

7. 堆栈寻址

堆栈是指当发生中断或子程序调用时,用来自动保存 PC 内容以及保护现场或传送参数。堆栈寻址是利用 SP 指针,按照先进后出的原则进行寻址,主要用来管理系统堆栈中的操作。

5.3 指 令 系 统

C54x 的助记符指令由操作码和操作数两部分组成,在进行汇编以前,操作码和操作数都是用助记符表示的。常用的符号以及运算符如表 5.1 和表 5.2 所示。

表 5.1　指令系统中的符号及其含义

序号	符　　号	含　　义
1	A	累加器 A
2	ALU	算术逻辑运算单元
3	AR	泛指通用辅助寄存器
4	ARx	指定某一辅助寄存器 AR0～AR7
5	ARP	ST0 中的 3 位辅助寄存器指针
6	ASM	ST1 中的 5 位累加器移位方式位－16～15
7	B	累加器 B
8	BRAF	ST1 中的块重复操作标志
9	BRC	块重复操作寄存器
10	BIT 或 bit_code	用于测试指令，指定数据存储器单元中的哪一位被测试，取指范围为 0～15
11	C16	ST1 中的双 16 位/双精度算术运算方式位
12	C	ST0 中的进位位
13	CC	2 位条件码(0≤CC≤3)
14	CMPT	ST1 中的 ARP 修正方式位
15	CPL	ST1 中的直接寻址编辑标志位
16	cond	表示一种条件的操作数，用于条件执行指令
17	[d],[D]	延时选项
18	DAB	D 地址总线
19	DAR	DAB 地址寄存器
20	dmad	16 位立即数数据存储器地址(0～65535)
21	Dmem	数据存储器操作数
22	DP	ST0 中的数据存储器页指针(0≤DP≤511)
23	dst	目的累加器(A 和 B)
24	dst_	与 dst 相反的目的累加器
25	EAB	E 地址总线
26	EAR	EAB 地址总线
27	extpmad	23 位立即程序存储器地址
28	FRCT	ST1 中的小数方式位
29	hi(A)	累加器的高阶位(AH 或 BH)
30	HM	ST1 中的保持方式位
31	IFR	中断标志寄存器
32	INTM	ST1 中的中断屏蔽位
33	K	少于 9 位的短立即数
34	K3	3 位立即数(0≤K3≤7)

续表

序号	符　　号	含　　义
35	K5	5位立即数(－16≤K5≤15)
36	K9	9位立即数(0≤K9≤511)
37	1K	16位长立即数
38	Lmem	利用长字寻址的32位单数据存储器操作数
39	Mmr,MMR	存储器映像寄存器
40	MMRx,MMRy	存储器映像寄存器,AR0～AR7或SP
41	n	XC指令后面的字数,取1或2
42	N	指定状态寄存器,N＝0为ST0,N＝1为ST1
43	OVA	ST0中的累加器A溢出标志
44	OVB	ST0中的累加器B溢出标志
45	OVdst	指定目的累加器(A或B)的溢出标志
46	OVdst_	指定与Ovdst相反的目的累加器的溢出标志
47	OVsrc	指定源累加器(A或B)的溢出标志
48	OVM	ST1中的溢出方式位
49	PA	16位立即端口地址(0≤PA≤65535)
50	PAR	程序存储器地址寄存器
51	PC	程序计数器
52	Pmad	16位立即程序存储器地址(0≤Pmad≤65535)
53	Pmem	程序存储器操作数
54	PMST	处理器工作方式状态寄存器
55	prog	程序存储器操作数
56	[R]	舍入选项
57	rnd	循环寻址
58	RC	重复计数器
59	RTN	快速返回寄存器
60	REA	块重复结束地址寄存器
61	RSA	块重复起始地址寄存器
62	SBIT	用于指定状态寄存器位的4位地址(0～15)
63	SHFT	4位移位值(0～15)
64	SHIFT	5位移位值(－16～15)
65	Sind	间接寻址的单数据存储器操作数
66	Smem	16位单数据存储器操作数
67	SP	堆栈指针寄存器
68	src	源累加器(A或B)

续表

序号	符　　号	含　　义
69	ST0,ST1	状态寄存器 0,状态寄存器 1
70	SXM	ST1 中的符号扩展方式位
71	T	暂存器
72	TC	ST0 中的测试/控制标志
73	TOS	堆栈顶部
74	TRN	状态转移寄存器
75	TS	由 T 寄存器的 5～0 位所规定的移位数(－16～31)
76	uns	无符号数
77	XF	ST1 中的外部标志状态位
78	XPC	程序计数器扩展寄存器
79	Xmem	16 位双数据存储器操作数,用于双数据操作数指令
80	Ymem	16 位双数据存储器操作数,用于双数据操作数指令和单数据操作指令

表 5.2　指令系统的运算符号

序号	符　　号	运 算 功 能	求值顺序
1	＋　－　～　！	取正、取负、按位求补、逻辑负	从右至左
2	＊　／　％	乘法、除法、求模	从左至右
3	＋　－	加法、减法	从左至右
4	^	指数	从左到右
5	＜＜　＞＞	左移、右移	从左到右
6	＜　≤	小于、小于等于	从左到右
7	＞　≥	大于、大于等于	从左到右
8	≠　！＝	不等于	从左到右
9	&	按位与运算	从左到右
10	∧	按位异或运算	从左到右
11	|	按位或运算	从左到右

C54x 的指令系统共有 129 条基本指令,由于操作数的寻址方式不同,由它们可以派生多至 205 条指令。

按指令的功能可分成 6 大类：数据传送指令、程序控制指令、算术运算指令、并行操作指令、逻辑运算指令和重复操作指令。

5.3.1 数据传送指令

数据传送指令是从存储器中将源操作数传送到目的操作数所指定的存储器中,包括装载指令、存储指令、条件存储指令、混合装载和存储指令。

1. 装载指令

即取数或赋值指令，用于将存储器内容或立即数赋给目的寄存器，共计 21 条。

指令格式：操作码 源操作数 [,移位数], 目的操作数

其中操作码包括 DLD、LD、LDM、LDR、LDU 和 LTD，最常用的是 LD 指令。各指令的格式和功能如表 5.3 所示。

表 5.3 装载指令的格式与功能

指令	指令格式	指令功能	说明
DLD	DLD Lmem,dst	dst=Lmem	将 Lmem 所指定的单数据存储器中的 32 位数据送入累加器 A 或 B 中
LD	LD Smem,dst	dst=Smem	将 Smem 所指定的单数据存储器中的 16 位数据送入累加器 A 或 B 中
	LD Smem,TS,dst	dst=Smem<<TS	将 Smem 所指定的单数据存储器中的数据，按 TS 所给定的移位数(−16≤TS≤31)移位，然后送入 A 或 B
	LD Smem,16,dst	dst=Smem<<16	将 Smem 所指定的单数据存储器的数据左移 16 位后送入 A 或 B
	LD Smem [,SHIFT],dst	dst=Smem<<SHIFT	将 Smem 所指定的单数据存储器的数据，按 SHIFT 所给定的移位数移位，然后送入 A 或 B
	LD Xmem,SHFT,dst	dst=Xmem<<SHFT	将 Xmem 所指定的双数据存储器的数据，按 SHFT 所给定的移位数移位，然后送入 A 或 B
	LD #K,dst	dst=#K	将短立即数 K 送入累加器 A 或 B
	LD #lk [,SHFT],dst	dst=#lk<<SHFT	将长立即数 lk 移位后，送入累加器 A 或 B
	LD #lk,16,dst	dst=#lk<<16	将长立即数 lk 左移 16 位后，送入累加器 A 或 B
	LD src,ASM [,dst]	dst=src<<ASM	将源累加器 src 中的数据，按 ASM(−16≤SAM≤15)所给定的移位数移位后，送入目的累加器 dst
	LD src [,SHIFT] [,dst]	dst=src<<SHIFT	将源累加器 src 中的数据，按 Shift 所给定的移位数移位后，送入目的累加器 dst
	LD Smem,T	T=Smem	将 Smem 所指定的单数据器的数据送入暂存器 T
	LD Smem,DP	DP=Smem(8−0)	将 Smem 所指定的单数据存储器的低 9 位数据送入数据存储器页指针 DP
	LD # k9,DP	DP=#k9	将 9 位立即数送入 DP
	LD #k5,ASM	ASM=#k5	将 5 位立即数送入累加器移位方式位 ASM
	LD #k3,ARP	ARP=#k3	将 3 位立即数送入 ARP(3 位辅助寄存器指针位)
	LDSmem,ASM	ASM=Smem(4−0)	将 Smem 所指定的单数据存储器的低 5 位数据送入 ASM

续表

指令	指令格式	指令功能	说明
LDM	LDM MMR,dst	dst=MMR	将 MMR 寄存器中的数据送入累加器 dst
LDR	LDRSmem,dst	dst(31－16)=rnd(Smem)	将 Smem 所指定的单数据存储器的数据舍入后送入累加器
LDU	LDUSmem,dst	dst=uns(Smem)	将 Smem 所指定的单数据存储器的无符号数据送入累加器
LTD	LTDSmem	T=Smem,(Smem+1)=Smem	将单数据存储器 Smem 的数据送入寄存器 T,并延时

2. 存储指令

存储指令是将源操作数或立即数存入指定存储器或寄存器,共计 14 条。

指令格式：操作码 源操作数 [,移位数], 目的操作数

其中操作码包括 DST、ST、STH、STL、STLM 和 STM,各指令的格式和功能如表 5.4 所示。

表 5.4 存储指令的格式与功能

指令	指令格式	指令功能	说明
DST	DSTsrc, Lmem	Lmem=src	累加器值存入长字存储单元
ST	STT, Smem	Smem=T	暂存器值存入存储单元
	STTRN, Smem	Smem=TRN	状态寄存器值存入存储单元
	ST #lk, Smem	Smem=#lk	长立即数存入存储单元
STH	STHsrc, Smem	Smem=src(31－16)	累加器高阶位存入存储单元
	STHsrc,ASM, Smem	Smem=src(31－16)<<ASM	累加器高阶位移位后存入存储单元
	STHsrc,SHFT, Xmem	Xmem=src(31－16)<<SHFT	累加器高阶移位后存入存储单元
	STHsrc [,SHIFT], Smem	Smem=src(31－16)<<SHIFT	累加器高阶位移位后存入存储单元
STL	STLsrc, Smem	Smem=src(15－0)	累加器低阶位存入存储单元
	STLsrc, ASM, Smem	Smem=src(15－0)<<ASM	累加器低阶位移位后存入存储单元
	STLsrc, SHFT, Xmem	Xmem=src(15－0)<<SHFT	累加器低阶位移位后存入存储单元
	STLsrc[,SHIFT], Smem	Smem=src(15－0)<<SHIFT	累加器低阶位移位后存入存储单元
STLM	STLMsrc, MMR	MMR=src(15－0)	累加器低阶位存入 MMR
STM	STM #lk, MMR	MMR=#lk	长立即数存入 MMR