第3章

运算方法与运算器

内容要点：

重点介绍运算方法，包括定点加减法运算、定点乘法运算、定点除法运算、逻辑运算以及浮点加减法运算。

介绍运算器的结构，包括算术逻辑部件(ALU)，通用寄存器和标志寄存器，以及各部件相互连接的数据通路。

定点运算器以位片电路 Am2901 为例作了详细介绍。

3.1 定点加减法运算例题分析

定点加法运算是计算机内最重要的运算，减法可通过补码加法实现，乘法和除法可以通过一系列加减法和移位实现。

并行加法器是算术逻辑部件的核心和主体，不管计算机运算采用什么码制，加法器结构本质上都是执行补码操作完成的。

已知两个定点数 x 和 y，将 x 和 y 用补码表示成$[x]_{补}$、$[y]_{补}$，则有：

$$[x+y]_{补}=[x]_{补}+[y]_{补}$$

$$[x-y]_{补}=[x]_{补}+[-y]_{补}$$

因此，不管 x、y 的真值是正数还是负数，求和时只要把 x、y 用补码表示，就可直接做加法，得到的结果就是 x 与 y 和的补码，而不需考虑谁大谁小、谁负谁正，结果的符号就是和的补码的正确符号。

做减法时，只要把减数符号改号，求得$[-y]_{补}$直接按加法运算，最后得到的结果就是(x－y)的补码。

做补码加减法，其补码符号位和数值位一起参加运算，最后得到的结果符号位就是二数和或差的正确符号位。

需要注意，当补码加减法溢出时，结果的符号位将出现错误，为了避免错误，可采用双符号位法来解决，即每一个数的补码符号位用两个相同的数来表示：正数的符号用 00 来表示，负数的符号用 11 来表示。

当结果的符号出现 01 或 10 时，说明补码加减法时出现溢出，01 表示正向溢出，或叫上溢，10 表时负向溢出，或叫下溢。

【例 3-1】 已知两个定点二进制小数，$[x]_补=0.1001$，$[y]_补=0.0101$，求$[x+y]_补$的值。

解：执行加减法运算，为防止溢出时出现错误的符号，两个操作数都用双符号位表示，不管 x、y 是正数或是负数。

$$\begin{array}{rr} [x]_补 & 00.1001 \\ +[y]_补 & +\ 00.0101 \\ \hline [x]_补+[y]_补 & 00.1110 \end{array}$$

$[x]_补+[y]_补=[x+y]_补=0.1110$

两个正数相加，结果的符号为正，和也正确，说明运算正常。

【例 3-2】 已知两个定点二进制数：$[x]_补=0.1101$，$[y]_补=0.1010$，求$[x+y]_补$的值。

解：两个操作数若不采用双符号位表示。

$[x]_补=0.1101$，$[y]_补=0.1010$

$[x+y]_补=[x]_补+[y]_补=0.1101+0.1010=1.0111$

两个正数相加，结果为负数，显然是错误的，原因是两个数相加，最高位数值位的和大于 1，产生进位，把符号位变成 1，实际结果为正，但结果大于 1，溢出了，属于上溢。

为避免溢出造成的错误，用双符号位表示操作数的数符，即$[x]_补=00.1101$，$[y]_补=00.1010$，则

$[x+y]_补=[x]_补+[y]_补=00.1101+00.1010=01.0111$

结果的符号为 01，说明出现问题，即结果为正，但溢出了，称为上溢。

今后执行补码加减法运算，不管结果如何，操作数必须用双符号位表示，才能给出正确的运算结果。

【例 3-3】 已知两个定点二进制小数：$[x]_补=0.1101$，$[y]_补=0.1010$，求$[x-y]_补$的值。

解：$[x-y]_补=[x+(-y)]_补=[x]_补+[-y]_补$

做减法时，应先求出减数$[-y]_补$，然后再作$[x]_补+[-y]_补$，即可得到$[x-y]_补$的结果。

$[y]_补=0.1010$，求$[-y]_补$时，可将$[y]_补$各位变反末位+1 得到，

$$\begin{aligned}[-y]_补&=1.0101+0.0001=1.0110\\ [x-y]_补&=[x]_补+[-y]_补\\ &=00.1101+11.0110\\ &=00.0011\end{aligned}$$

注意：最高符号位的进位可以作为补码的模数扔掉，对结果没有影响。

【例 3-4】 已知两个定点二进制小数：$[x]_补=1.1001$，$[y]_补=0.1011$，求$[x-y]_补$的值。

解：$[x-y]_补=[x]_补+[-y]_补$

先求$[-y]_补=1.0101$ 是由$[y]_补$各位变反末位+1 得到。

$$[x]_{补}+[-y]_{补}=11.1001+11.0101$$
$$=10.1110$$

扔掉最高符号位之进位，结果为10.1110表示其差为负数，但溢出了，称为下溢。

3.2 定点乘除法运算例题分析

原码乘法是计算机中的基本运算。

原码又称符号—绝对值表示法，最高位为符号位，后面是数的绝对值。求两个数原码的乘积时，可分别求出乘积的符号和乘积的绝对值。乘积的符号按照同号两数相乘乘积为正，异号两数相乘乘积为负，因此乘积的符号可用参加运算的两数符号的“异或”求得。

两数的乘积就是两数绝对值的乘积，绝对值看作正数。两个正数做乘法时，需根据乘数每位的值（“0”或“1”），决定将被乘数加到部分积中，或是不加，根据每位乘数的位权，将部分积进行移位，以便错位进行相加，根据乘数的位数，决定进行几次加被乘数的操作。

【例 3-5】 已知$[A]_{原}=0.1011$，$[B]_{原}=0.0101$为定点二进制小数。求$[A]_{原}\times[B]_{原}$的值。

解：求二数乘积的关键是求二数绝对值的积。

(1) 做乘法前，需将被乘数放入被乘数寄存器A中，乘数放入乘数寄存器B中，将部分积寄存器Z清“0”，将乘数位数放入乘法次数计数器C_d中。

(2) 乘法步骤：

① 根据乘数寄存器末位之值B_n，决定是否把被乘数加到部分积寄存器Z中。

若$B_n=1$，通过加法器做$Z+B\rightarrow Z$。

若$B_n=0$，通过加法器做$Z+0\rightarrow Z$。

加法操作均采用双符号位表示运算数据。

② 将乘数寄存器B右移1位，因为判断乘数之值的电路设在乘寄存器的末位；另外空出乘数寄存器的高位可用来存放双倍乘积的低位。

同时将部分积寄存器的内容右移1位，低位移入B之高位。主要是因为乘数各位数之位权不同，必须根据乘数的位数，实现错位相加。

③ 乘法次数计数器C_d-1，判断$C_d=0$?

若$C_d\neq0$，继续执行乘法下步操作，判断$B_n=1$?

若$C_d=0$，乘法结果得二倍字长乘积。

(3) 最后求乘积符号，将二数符号位异或，$A_0\oplus B_0\rightarrow Z_0$

乘法结束时，得到二倍字长乘积，乘积高位在Z中，乘积低位在乘数寄存器B中，乘数消失。

具体乘法步骤：

部分积寄存器	乘数寄存器	说明
00.0000	0101	起始，$C_d=4$
+ 00.1011		$B_4=1$, +A
00.1011		
00.0101	1010	Z、B 右移 1 位，$C_d-1=3$
+ 00.0000		$B_3=0$, +0
00.0101		
00.0010	1101	Z、B 右移 1 位，$C_d-1=2$
+ 00.1011		$B_2=1$, +A
00.1101		
00.0110	1111	Z、B 右移 1 位，$C_d-1=1$
+ 00.0000		$B_1=0$, +0
00.0110		
00.0011	0111	Z、B 右移 1 位，$C_d-1=0$
		乘法结束

乘积符号 $Z_0=A_0\oplus B_0=0\oplus 0=0$，乘积为正。

因此，$[A]_原\times[B]_原=0.00110111$，高位积在 Z 中，低位积在 B 中。

【例 3-6】 已知两个定点二进制数，$A=0.101011$，$B=0.011010$，用原码两位乘法求 $[A]_原\times[B]_原$ 的值。

解：乘法步骤与原码一位乘法类似，区别在于每次根据乘数末 2 位的值决定把几倍被乘数加到部分积中，另外在乘数寄存器末尾又增加一个附加位 B_{n+1}，以便每次根据乘数末 2 位的值及附加位的值决定具体的乘法操作。每次乘法后，乘数寄存器右移 2 位，以便判断新的乘数位，部分积寄存器右移 2 位，以便实现按乘数之位权实现错位加法。

乘数末 2 位 B_{n-1}	B_n	附加位 B_{n+1}	乘法规则	说明
0	0	0	$Z_{i+1}=\frac{1}{4}(Z_i+0)$	加 0，右移 2 位
0	1	0	$Z_{i+1}=\frac{1}{4}(Z_i+A)$	加 A，右移 2 位
1	0	0	$Z_{i+1}=\frac{1}{4}(Z_i-2A)$	减 2A，右移 2 位
1	1	0	$Z_{i+1}=\frac{1}{4}(Z_i-A)$	减 A，右移 2 位
0	0	1	$Z_{i+1}=\frac{1}{4}(Z_i+A)$	加 A，右移 2 位
0	1	1	$Z_{i+1}=\frac{1}{4}(Z_i+2A)$	加 2A，右移 2 位
1	0	1	$Z_{i+1}=\frac{1}{4}(Z_i-A)$	减 A，右移 2 位
1	1	1	$Z_{i+1}=\frac{1}{4}(Z_i+0)$	加 0，右移 2 位

设立乘法附加位是为了简化乘法规则和乘法控制电路。乘法运算时要加 2 倍被乘

数，可能溢出，为了得到正确结果，有关操作数都需用 3 位符号位，000 表示正数，111 表示负数，3 个符号位不同，表示溢出，但运算过程中并不担心会溢出，因为加法之后，马上要右移 2 位。

加法运算中＋2A，实际上是把被乘数 A 左移 1 位送到加法器中实现。

运算过程中用到有关＋A，－A，＋2A，－2A 各数需要在运算前求出。

$+A=000.101011$ $\qquad$ $+2A=001.010110$

$[-A]_{补}=111.010101$ $\qquad$ $[-2A]_{补}=110.101010$

两位原码乘法运算过程如下：

部分积寄存器	乘数寄存器附加位	说明
000.000000	011010 $\boxed{0}$	起始
+ 110.101010		乘数末 3 位 100，－2A
110.101010		
111.101010	100010 $\boxed{1}$	右移 2 位
+ 111.010101		乘数末 3 位 101，－A
① 110.111111		
111.101111	111001 $\boxed{1}$	右移 2 位
+ 001.010110		乘数末 3 位 011，＋2A
① 001.000101		
000.010001	011110 $\boxed{0}$	右移 2 位

6 位乘数的两位乘法共做 3 次加法，3 次右移，得到结果，比一位乘法运算速度快一倍，但运算器结构要复杂一些，除了执行＋A，－A 运算外，还要做＋2A，－2A 运算。部分积寄存器和乘数寄存器还需设置右移 2 位线路。

乘积符号 $Z_0=A_0 \oplus B_0=0 \oplus 0=0$，为正数。

最后得双倍字长乘积：Z＝0.010001011110，高位乘积在部分积寄存器中，低位乘积在乘数寄存器中。

【例 3-7】 已知两个定点二进制小数：A＝0.1011，B＝0.1101。用恢复余数除法求 $[A]_{原} \div [B]_{原}$ 的值。

解：原码除法求商，将商的符号与商的数值分开处理。

商的符号：根据同号两数相除商为正，异号两数相除商为负得到。可用异或电路求得 $A_0 \oplus B_0$。

求商的数值：

(1) 判断除法溢出，比较余数决定商什么。

在定点小数除法中，被除数之值必须小于除数，否则商的数值要大于 1，造成溢出。

计算机中判断溢出是用被除数减除数来实现的，如果够减，表示商的数值大于 1，此时商寄存器末位商“1”，如果不够减商“0”。第 1 次除法商“1”，表示被除数大于除数，商大于 1，在定点小数运算中表示结果溢出，不再继续做除法，转溢出处理。

一般情况，定点小数运算结果应小于 1，第 1 位商“0”，表示不够减，不溢出。继续作除法时，还应加上除数，恢复成原来的被除数，才能继续做下一步除法。

(2) 余数左移 1 位。第 1 次不够减，商“0”，商在整数位上，恢复余数后，将余数左移 1 位，使余数乘 2，以便再与除数比较大小。

(3) 根据比较结果上商，上商线路设在商寄存器末位，每次运算商寄存器也左移 1 位，以便下次上商。

根据除数位数，决定除法次数，如果被除数、除数数值位都是 4 位，加上第 1 次判溢出共需做 5 次减法，4 次左移即可得 4 位商，结束除法。

除法操作开始前需把被除数 A 放在被除数寄存器，除法开始后，被除数寄存器用来存放部分余数。除数放在除数寄存器 B 中，还需设置一个商寄存器 C，每次商在商寄存器末位。

除法过程中用到 +B，−B 操作，可先求出 $[-B]_{补}=[-0.1101]_{补}=1.0011$，$[+B]_{补}=0.1101$。

恢复余数除法：当每次余数为负，商“0”时，需要加上除数，以恢复原来余数。

原码恢复余数除法过程：

被除数寄存器 A	商寄存器 C	说明
00.1011	0.0000	起始
+ 11.0011		−B, 判溢出
11.1110	0.000$\boxed{0}$	$R_0<0$, 商 0, 不溢出
+ 00.1101		+B, 恢复余数
① 00.1011		
01.0110	0.000□	A 、 C 左移 1 位
+ 11.0011		−B
① 00.1001	0.000$\boxed{1}$	$R_1>0$, 商 “ 1 ”
01.0010	0.001□	A 、 C 左移 1 位
+ 11.0011		−B
① 00.0101	0.001$\boxed{1}$	$R_2>0$, 商 “ 1 ”
00.1010	0.011□	A 、 C 左移 1 位
+ 11.0011		−B
11.1101	0.011$\boxed{0}$	$R_3<0$, 商 “ 0 ”
+ 00.1101		+B, 恢复余数
① 00.1010		
01.0100	0.110□	A 、 C 左移 1 位
+ 11.0011		−B
100.0111	0.110$\boxed{1}$	$R_4>0$, 商 “ 1 ”

除法过程做 5 次减法，4 次移位，求得 5 位商。第 1 次求商实际上是从整数位上判溢出，当整数位为“1”，表示结果大于“1”，溢出，应该结束除法。多数情况第 1 次求得的整数商为“0”。

两数同号相除，商的符号为正。

$[A]_{原} \div [B]_{原} = 0.1101$

余数为 0.0111，实际上这是左移 4 位后的余数，真正余数 $R = 0.0111 \times 2^{-4}$。

【例 3-8】 已知两个定点二进制小数，$A = 0.1001$，$B = -0.1011$。

用加减交替法求$[A]_{原} \div [B]_{原}$的值。

解： 加减交替法除法又叫不恢复余数除法，因为恢复余数时需多做若干次加法，使除法过程变慢，速度降低。

原码除法，商的符号位单独处理，商的数值看作两个正数相除。

加减交替法运算规则与恢复余数除法之区别在于：

恢复余数除法每次减除数后，根据余数决定商及是否恢复余数。当余数为正时，上商"1"，不恢复余数，下次做减法；余数为负时，上商"0"，且需恢复余数，下次做减法。

加减交替法每次根据余数决定上商和下次做加法还是做减法，但都不恢复余数。当余数为正时，上商"1"，不恢复余数，下次做减法；当余数为负时，上商"0"，不恢复余数，但下次做加法。

因为不需恢复余数，所以除法操作中，加减法的次数固定，速度比恢复余数除法快。

本例中，$[A]_{原} = 0.1001$，$[B]_{原} = 1.1011$，做原码除法时，不考虑数的符号，都作为正数进行运算。因此 B 虽为负数，但原码除法求商的绝对值时，可把 B 当作正数看，商的符号单独处理。

$+B$ 操作：$+0.1011$　　　　　$-B$ 操作：$+[-0.1011]_{补} = 1.0101$

加减交替原码除法过程如下：

被除数放在被除数寄存器 A 中，除数放在除数寄存器 B 中，商从商寄存器末位开始上商。

被除数寄存器 A	商寄存器 C	说明
00.1001	0.000 0	起始
+ 11.0101		−B, 判溢出
11.1110	0.000 $\boxed{0}$	$R_0<0$, 商"0", 不溢出
11.1100	0.000 $\boxed{\ }$	A 、C 左移 1 位
+ 00.1011		+B
① 00.0111	0.000 $\boxed{1}$	$R_1>0$, 商"1"
00.1110	0.001 $\boxed{\ }$	A 、C 左移 1 位
+ 11.0101		−B
① 00.0011	0.001 $\boxed{1}$	$R_2>0$, 商"1"
00.0110	0.011 $\boxed{\ }$	A 、C 左移 1 位
+ 11.0101		−B
11.1011	0.011 $\boxed{0}$	$R_3<0$, 商"0"
11.0110	0.110 $\boxed{\ }$	A 、C 左移 1 位
+ 00.1011		+B
① 00.0001	0.110 $\boxed{1}$	$R_4>0$, 商"1"

5 位数的除法采用加减交替法共做 5 次加减法，4 次移位，求得 5 位商，结束运算，第 1 位商实际上是判结果溢出否。

二数异号为商的符号 $A_0 \oplus B_0 = 0 \oplus 1 = 1$，商为负，$[A]_{原} \div [B]_{原} = 0.1001 \div 1.1011 = 1.1101$，余数左移 4 次后，$R_4 = 0.0001$，真正余数 $R = 0.0001 \times 2^{-4}$。

3.3 逻辑运算例题分析

逻辑运算处理的逻辑变量是“真与假”两个相反的逻辑概念，用二进制数 0,1 表示。

常用的逻辑运算有逻辑加、逻辑乘、异或、求反等运算，可直接使用或门、与门、异或门、非门实现。

所有的逻辑运算都是按位运算，相邻各位没有进位关系。

【例 3-9】 如何将 8 位寄存器中的数据最高位置“1”或置“0”？如何将 ASCII 码中低 4 位分离出来？

解：利用逻辑运算可以解决这些问题。

任何一位二进制数 S，经过下列运算可得下列结果：

$S \vee 1 = 1$　　　　$S \wedge 0 = 0$　　　　$S \oplus 1 = \overline{S}$

因此 8 位寄存器中的数据 A，可通过 $A \vee 80H$ 将最高位置“1”；通过 $A \wedge 7FH$ 可将最高位置“0”；通过 $A \oplus 80H$ 可将最高位变反。

如果需要将 ASCII 码低 4 位分离出来，可进行下列逻辑操作：

$$(\text{ASCII}) \wedge 0FH$$

【例 3-10】 化简逻辑函数 $F = AB + \overline{A}C + BCD$。

解：利用基本逻辑代数公式化简：

$$\begin{aligned} F &= AB + \overline{A}C + BCD \\ &= AB + \overline{A}C + BC + BCD \\ &= (AB + \overline{A}C) + (BC + BCD) \\ &= AB + \overline{A}C + BC \\ &= AB + \overline{A}C \end{aligned}$$

其中用到：

① 互补律：$A + \overline{A} = 1$。

② 0－1 律：$A + 1 = 1$。

③ 吸收律：$A + AB = A$，即 $A + AB = A(1 + B) = A$。

④ 第二吸收律：$A + \overline{A}B = A + B$，即

$$\begin{aligned} A + \overline{A}B &= A + AB + \overline{A}B \\ &= A + B(A + \overline{A}) \\ &= A + B \end{aligned}$$

$AB + \overline{A}C + BC = AB + \overline{A}C$，即

$$\begin{aligned} AB + \overline{A}C + BC(A + \overline{A}) &= AB + \overline{A}C + ABC + \overline{A}BC \\ &= AB + ABC + \overline{A}C + \overline{A}CB \\ &= AB + \overline{A}C \end{aligned}$$

3.4 位片结构定点运算器例题分析

运算器是计算机数据处理中心，主要功能是对数据进行算术运算和逻辑运算。运算器的核心是一个并行加法器，运算结果的特征，如溢出、结果为“0”等需保存在专门的标志寄存器中。

运算器中设置一组寄存器，用来存放参加运算的数据和中间结果，区别于单累加器结构，通用寄存器中每个寄存器都可以存放运算结果，程序员通过指令使用这组寄存器。

为了完成乘除法运算，运算器中提供左右移位功能，为了支持乘除法运算设置乘商寄存器，用来存放乘数或商数。

【例 3-11】 运算器中各寄存器间如何交换数据？运算器与存储器和 I/O 如何交换数据？

解：运算器的核心是算术逻辑部件(ALU)，是整机的数据处理中心，计算机中各部件的数据都需要送到 ALU 中来处理。ALU 有两个输入端，送入两个被运算的数据，一个输出端送出运算结果。运算器中有多个数据寄存器，通过两个多路数据选择器选择需要的寄存器中的数据，分别送给 ALU 的两个数据输入端，经过算术逻辑部件处理，由 ALU 输出。外部数据，例如存储器和 I/O 等需要送往通用寄存器的数据，也需要经过 ALU 输入多路选择器，经过 ALU 再送入指定的寄存器中。ALU 输出数据线即可作为与各寄存器连接的共用数据线，这样可以节省寄存器间的连接线，也比较整齐，我们把运算器中各寄存器间交换数据的公共通路叫内总线，或 CPU 总线，这种结构连线较少，扩充容易。

计算机中各部件包括存储器与 I/O 设备都需要与 ALU 联系，都需要建立与 ALU 的连接通路，因此 ALU 是整个计算机的数据传输中心。冯·诺依曼在总结现代计算机的特征时，专门指出整个计算机运算器为中心。

【例 3-12】 什么叫位片？位片结构有什么特点？

解：随着集成电路技术的发展，可以把整个中央处理器(CPU)做在一个芯片上。其缺点是计算机的字长、指令系统都已固定，控制逻辑也不能修改，这给不同用途的用户带来很大不便。

位片结构将运算器纵向划分成标准模块，每个模块都是一个字长较短的标准运算器。如 Am2901 就是一个四位运算器，包括全加器、通用寄存器、多路选择器、乘商寄存器及移位电路等。研制不同字长的运算器，可以通过选用不同数目的位片来实现，为设计、制造、使用部门提供很大方便。

【例 3-13】 Am2901 位片运算器的 9 位控制码有什么功能？

解：Am2901 是一个 4 位的运算器芯片，其 ALU 输入数据选择、运算性质、运算结果去向，都是由其控制码决定的。

9 位控制码中的 $I_0I_1I_2$ 用来决定把什么数据送入 ALU 的哪个输入端，数据来源包括通用寄存器、乘商寄存器、零或外部数据。当然决定从通用寄存器取数时，还必须根据指令中指定的通用寄存器号来选取。2901 有 16 个通用寄存器，其地址编号必须用 4 位二进制数表示。

$I_3I_4I_5$ 用来决定运算种类，包括＋、－、∧、∨、⊕等。

$I_6I_7I_8$ 用来决定运算结果去向及移位功能，去向包括：通用寄存器、Q 寄存器及片外部件。送给 16 个通用寄存器的具体哪个寄存器是由指令给定的 B 口地址决定的。

Am2901 是一种功能很强的典型的运算器，弄清其工作原理，对了解计算机是如何工作的很有意义。

【例 3-14】 运算器中设置标志寄存器有什么用处？

解：运算器执行算术运算或逻辑运算，得到运算结果，这个结果往往成为后续指令程序分支的条件，因此需要保存运算结果的特征。如结果为零、为负、溢出、有进位等，保存运算结果特征的寄存器，称为标志寄存器，有时也叫程序状态字 PSW。每种特征在寄存器中用一位来表示。

如结果为"0"时，其对应特征位 Z＝1，若 Z＝0，表示结果不为"0"；如结果溢出时，其对应特征位 V＝1，若 V＝0，表示结果不溢出；同理 C＝1 表示结果有进位，F＝1，表示结果为负等。

【例 3-15】 双端口存储器有什么特点和用处？

解：一般存储器都是给一个地址，读出一个数，属于单端口存储器，不能给两个地址读出两个数。为了提高访存的速度，希望能够给存储器两个地址，同时读出两个数，这里说的存储器当然是对一个存储器而言，不是两个存储器。如果这样，访存速度可以提高一倍，这种存储器称为双端口存储器，在这种存储器内部需要设置两套独立的读出控制电路。

注意：为了防止两个端口同时向一个存储单元写入不同数，规定写入双端口存储器时，只允许一个端口作为写入端口，但两个端口可以同时读出同一个存储单元，不会发生矛盾。

运算器中，ALU 操作，多数情况需要两个操作数，如果两个操作数都放在一个类似存储器的通用寄存器组中，需要两步才能读出两个操作数。为了提高运算速度，希望 ALU 运算时，一拍能够读出两个操作数，这就要求通用寄存器组具有双端口读出的功能，需要设置两套独立的读出控制电路，分别同时给出两个地址，从两个端口同时读出两个操作数。

这种双端口通用寄存器，给两个地址，可以同时读出两个操作数，供给 ALU 两个输入端，在一拍时间内 ALU 可给出运算结果。当两个操作数都来自同一个寄存器，也是允许的，但写入通用寄存器时，不允许同时写入两个操作数，防止同时写入同一个寄存器时造成冲突。所以只设置一套写入控制电路。在运算器中，这种设置不影响写入速度，因为 ALU 只能产生一个运算结果，保存在任何指定的一个寄存器中就可以了，不需要同时写入两个数据。

在 Am2901 中，A 口、B 口都是读出端口。A 口地址、B 口地址都是 4 位，分别控制从 16 个地址中读出有关单元内容送给 A 口和 B 口输出。但写入时，只能由 B 口地址指定，将运算结果写入 B 口地址中。

这种操作类似于二地址指令，A 作为源地址，B 作为目的地址，操作码是 OP，执行下述运算：

$$(B)OP(A)\rightarrow B$$

【例 3-16】 运算器中，在 ALU 输入与通用寄存器输出之间设置数据锁存器，有什么用途？