第4章 运算方法与运算器

4.1 ~4.4 定点数加减乘除运算

4.5 浮点数的四则运算

4.6 运算器的组织

4.1 定点数的加减运算

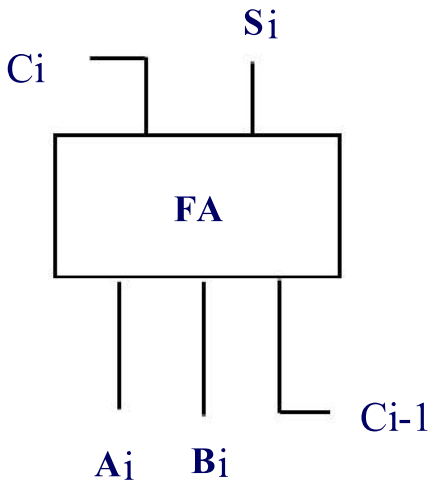
本节内容

- 并行加法器及先行进位链
- 补码的加减运算及溢出判断

二－十进制加法运算及加法器

3.3.1 加法器

一位全加器



A_i	B_i	C_{i-1}	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$Si = \overline{A_i} \cdot \overline{B_i} \cdot C_{i-1} + \overline{A_i} \cdot B_i \cdot \overline{C_{i-1}} + A_i \cdot \overline{B_i} \cdot \overline{C_{i-1}} + A_i \cdot B_i \cdot C_{i-1} = A_i \oplus B_i \oplus C_{i-1}$$

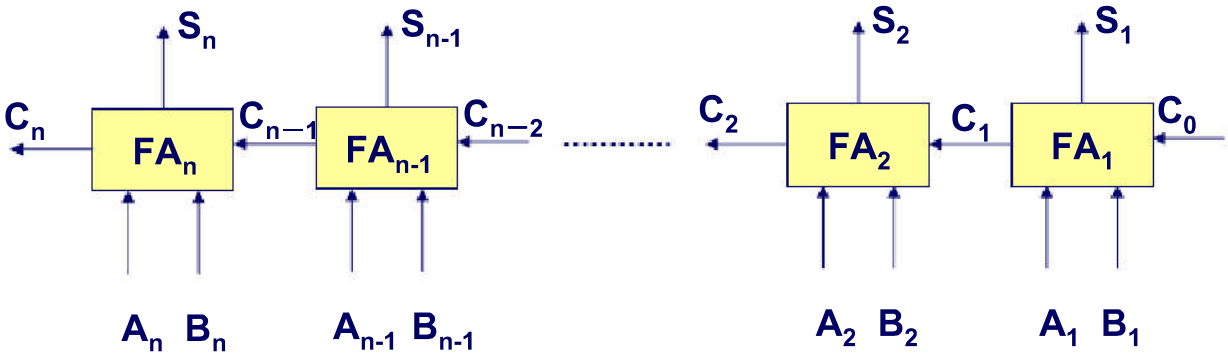
$$C_i = \overline{A_i} \cdot B_i \cdot C_{i-1} + A_i \cdot \overline{B_i} \cdot C_{i-1} + A_i \cdot B_i \cdot \overline{C_{i-1}} + A_i \cdot B_i \cdot C_{i-1} = A_i \cdot B_i + (A_i \oplus B_i) C_{i-1}$$

加法器的实现方案

❖串行加法器：只用1个一位全加器，每次加一位，在移位寄存器的配合下，实现n位加法。速度慢。

❖行波进位的并行加法器

- 用n个一位全加器串行连接。

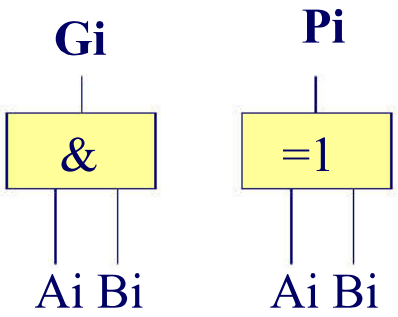


❖特点：高位运算等待低位进位，逐位等待。进位连接方式称为“串行进位”或“行波进位”(Carry Ripple)。

先行进位CLA（Carry Look Ahead）

❖先行进位的并行加法器

- 加快进位的产生和传递。



❖令 $G_i=A_iB_i$

$P_i=A_i \oplus B_i$

❖则 $C_i = A_iB_i + (A_i \oplus B_i) C_{i-1} = G_i + P_i C_{i-1}$

❖若 $G_i=1$, 则 $C_i=1$, 故称G_i为进位生成函数;

$P_i=1$, 则$C_i = C_{i-1}$, 故称P_i为进位传递函数。

❖先行进位加法器的基本思想:

不用全加器，重新设计并行加法电路，使高位运算不等待低位进位，而是从低位的运算数直接运算得到。

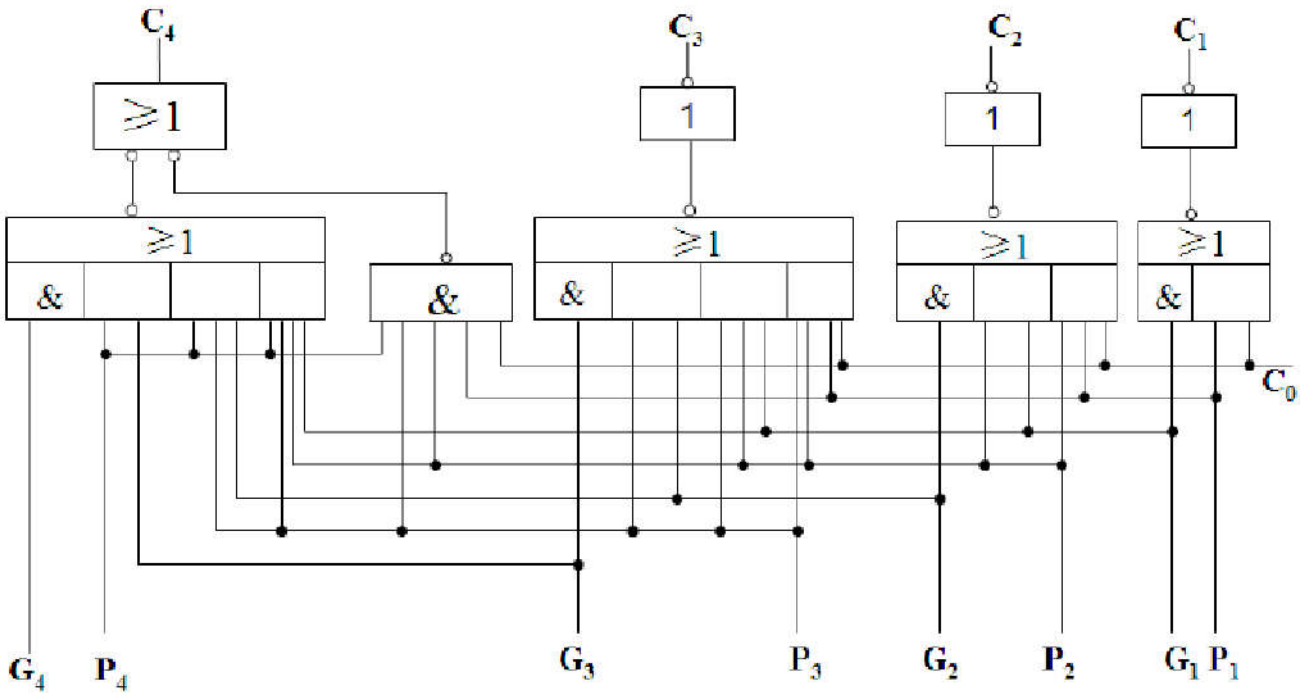
$C_i = G_i + P_i C_{i-1}$

- C1 = G1 + P1 C0
- C2 = G2 + P2 C1
  = G2 + P2 (G1 + P1 C0) = G2 + P2 G1 +P2 P1 C0
- C3 = G3 + P3 C2
  = G3 + P3 (G2 + P2 G1 +P2 P1 C0)
  = G3 + P3 G2 + P3 P2 G1 + P3 P2 P1 C0
- C4 = G4 + P4C3 =G4+P4 (G3+P3G2+P3P2G1+P3P2P1C0)
  = G4 + P4 G3+ P4 P3 G2 +P4 P3 P2 G1 + P4 P3 P2 P1 C0

- 四位进位输出仅由本位进位产生函数、进位传递函数、最低进位C0决定，当A，B各位同时到来，经过三级门的时延，各位进位同时产生。

  产生进位方式称为“并行进位”或“超前进位”或“先行进位”。

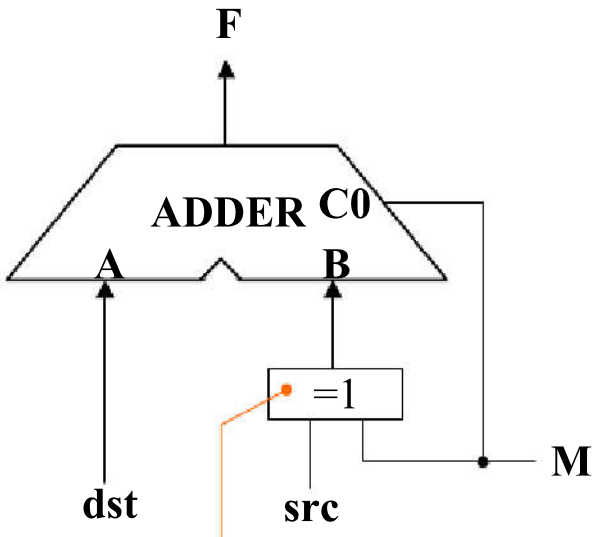
先行进位链



4.1.2补码的加减运算

- 补码加减法的依据是
  - $[X+Y]_{补}=[X]_{补}+[Y]_{补}$
    - 包括符号位在内一起运算
    - 求两个数的和，可以先将这两个数的补码相加，所得到的结果就是这两个数和的补码。
    - 意义：可以利用二进制加法器进行补码加运算。
  - $[-X]_{补}=-[X]_{补}$
  - $[X-Y]_{补}=[X]_{补}-[Y]_{补}=[X]_{补}+[-Y]_{补}$
    - 求两个数的差，可以转换为加法进行计算。将第一个数的补码加上第二个数相反数的补码，所得到的结果就是这两个数差的补码。
    - 意义：用加法器进行减法运算

4.1.3 补码加减运算的逻辑实现



用异或门实现可控取反：
b ⊕ 0 = b
b ⊕ 1 = ~b

- ❖加法器 Σ
  - $F = A+B+C_0$
- ❖M=0
  - B = src
  - C0 = 0
  - F= dst + src
- ❖M=1
  - B= ~src
  - C0 = 1
  - F=dst + ~src + 1
    =dst - src

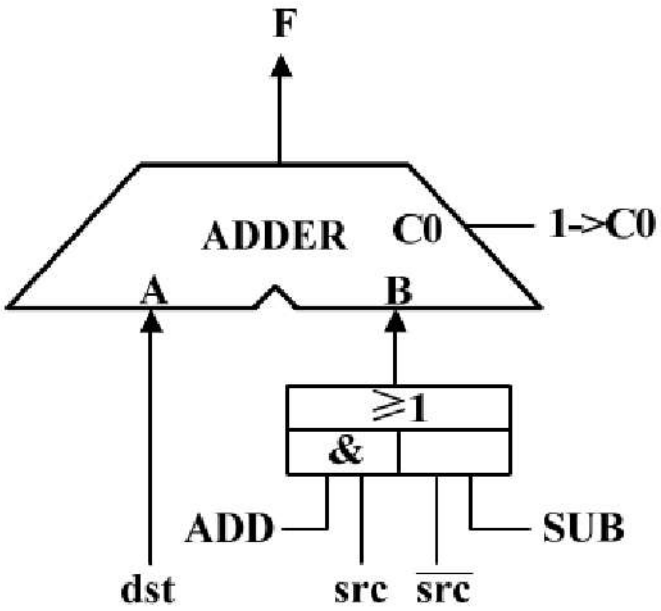
另一种输入选择逻辑

- dst + src:
  - ADD=1，SUB=0
    - 即src—>B
  - (1→C0) = 0
- dst − src:
  - SUB=1， ADD=0，
    - 即src的反—>B端
  - (1→C0) = 1
  - 取反加一
- 特别地，若

  ADD=0,SUB=0,(1→C0)=0

  则 B=0, C0=0

  ∴ F=A



程序状态字PSW（标志寄存器）

❖PSW (Program Status Word)

PSW

SF	ZF	OF	CF

- 一个专用寄存器，存放运算结果的特征标志。

- SF（Sign Flag）符号标志。
  - SF = 1表示结果为负数
  - SF = 0表示结果为正数
  - 逻辑方程：$\mathrm{SF} = S_n$

- ZF（Zero Flag）零标志。
  - ZF = 1表示结果为零
  - ZF = 0表示结果非零
  - 逻辑方程：$\mathrm{ZF} = \mathrm{S}_n + S_{n-1} + \cdots + S_0$

程序状态字PSW（标志寄存器）

❖运算结果的特征

- CF（Carry Flag）进位标志。
  - CF = 1表示有进位，CF = 0表示没有进位;
  - 逻辑方程: $\mathrm{CF} = S_{n+1}$
  - 无符号数加法运算溢出

- OF（Overflow Flag）溢出标志。
  - OF = 1表示补码运算结果溢出
  - OF = 0表示补码运算结果不溢出

补码加法溢出的判断

❖溢出：运算结果超出机器数的表示范围。

+7　0111

+)-4　1100

10011

有进位，无溢出

01101

无进位，无溢出

溢出判断方法

❖判别方法一

- 当两加数同号，但和的符号不同时，则溢出。

$$OF = X_n \bullet Y_n \bullet \overline{S}_n + \overline{X}_n \bullet \overline{Y}_n \bullet S_n$$

- Xn、Yn、Zn分别表示被加数、加数、和的符号

溢出判断方法（续）

❖判别方法二——变形补码（双符号位）判别法

+7　00111

正溢出：结果大于最大正数

无溢出

$$OF = S_{n+1} \oplus S_n$$

负溢出：结果小于最小负数

011___

双符号位相同，无溢出

负溢出

程序状态字PSW（标志寄存器）

❖运算结果的特征

- CF（Carry Flag）进位标志。
  - 加法运算时，CF = 1表示有进位，CF = 0表示没有进位；
  - 减法运算时，CF = 0表示有借位，CF = 1表示没有借位。

无符号数减法运算的借位

- 1 – 2 = -1，有借位
- 转化成加法运算：
  $1 + (-2)_{补}$

- 2 – 1 = 1，无借位
- 转化成加法运算：
  $2 + (-1)_{补}$

```
  +2   0010
+)-1   1111
```

对于无符号数的减法运算，转换为加法后，其借位是运算产生的进位的反，即：

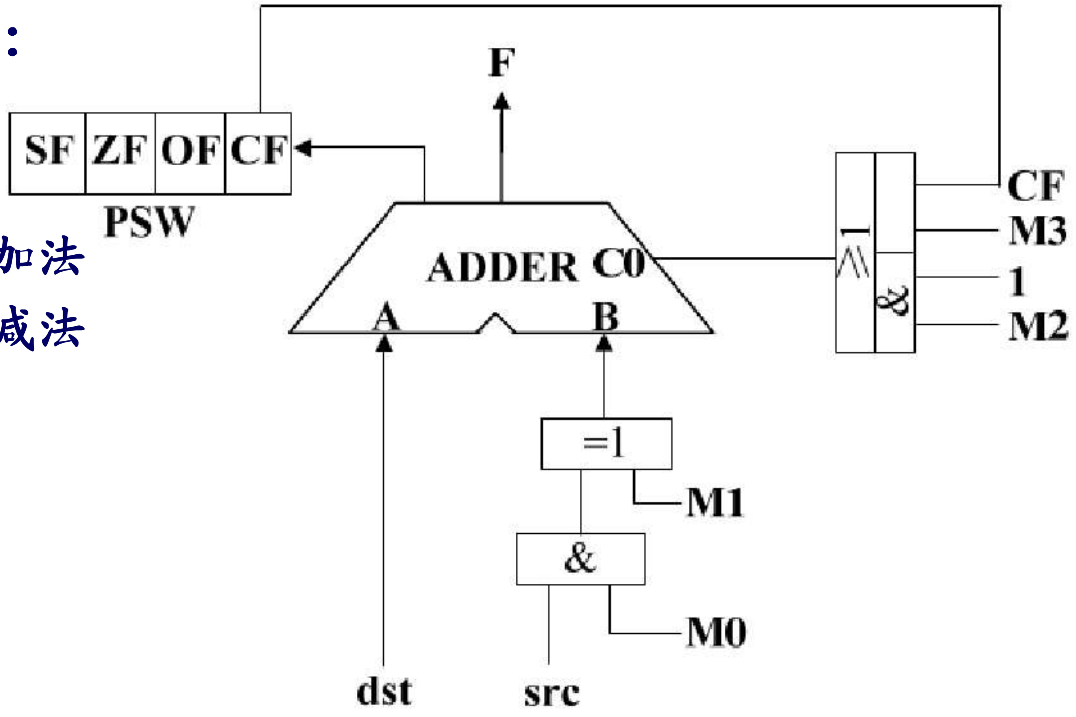
CF = 0表示有借位，CF = 1表示没有借位

4.6.2 定点运算器实例

1. 多功能加减运算电路

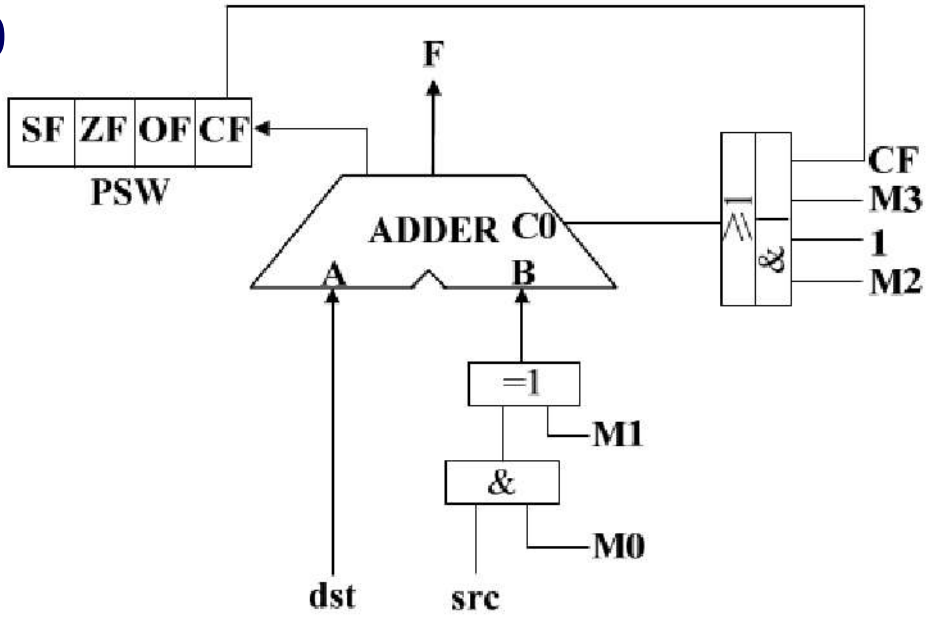
运算功能:

- 加法
- 减法
- 带进位的加法
- 带借位的减法
- 加1
- 减1
- 传送



多功能加减运算电路

- A = dst
- B = (src • M0) ⊕ M1
- C0 = M2 + (M3 • CF)
- F = A + B + C0



多功能加减运算电路

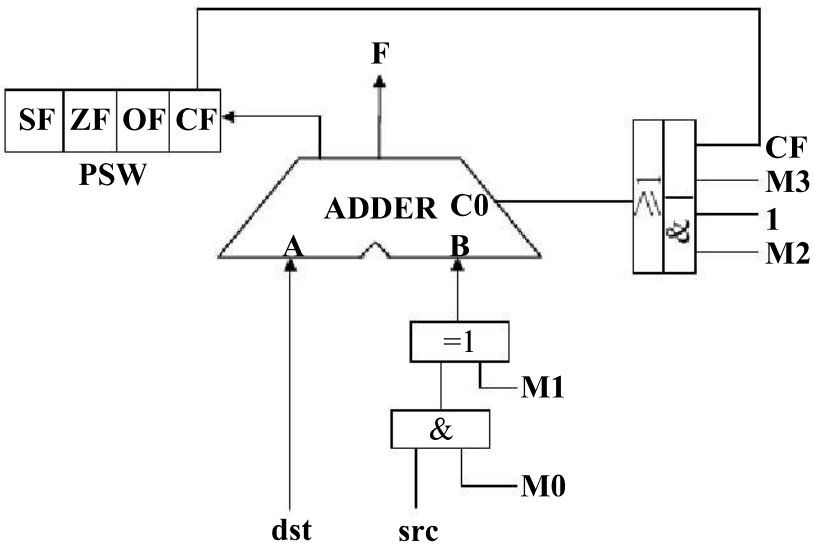
❖加法

- M3=0, M2=0, M1=0, M0=1

则

- B = src
- C0 = 0

∴F = dst + src



❖B = (src • M0) ⊕ M1

❖C0 = M2 + (M3 • CF)

❖F = A + B + C0

多功能加减运算电路

❖如何用n位加法器做2n位的

- 先对低n位加，进位保存在C
- 再对高n位加，同时加上CF

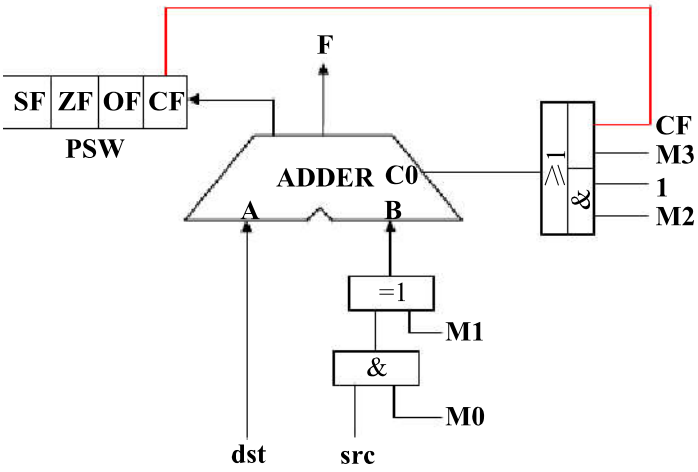
❖带进位的加法ADDC

- M3=1, M2=0, M1=0, M0=1

则

- B = src
- C0 = CF

❖F = dst + src + CF



多功能加减运算电路

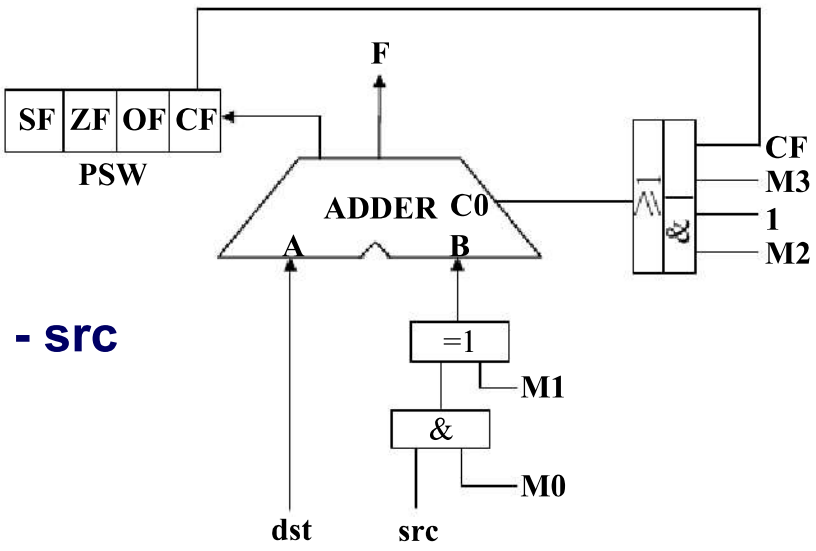
❖减法

- M3=0, M2=1, M1=1, M0=1

则

- B = (~src)
- C0 = 1

∴ F = dst + ~src + 1 = dst - src



❖B = (src • M0) ⊕ M1

❖C0 = M2 + (M3 • CF)

❖F = A + B + C0

多功能加减运算电路

- 用n位加法器做2n位的减法？
  - 先对低n位减，借位的反保存在CF
    - 由于减法转换为补码加法，CF是借位的反
  - 再对高n位减，同时减去借位
- 带借位的减法：

  ∵ dst – src – borrow = dst + (~src) + 1 – borrow

  ∵ 1 – borrow = (~borrow)

  ∴ dst – src – borrow = dst + (~src) + (~borrow)
- 而减法运算时，CF是借位的反
  - 即CF = (~borrow)

  ∴ dst – src – borrow = dst + (~src) + CF

多功能加减运算电路

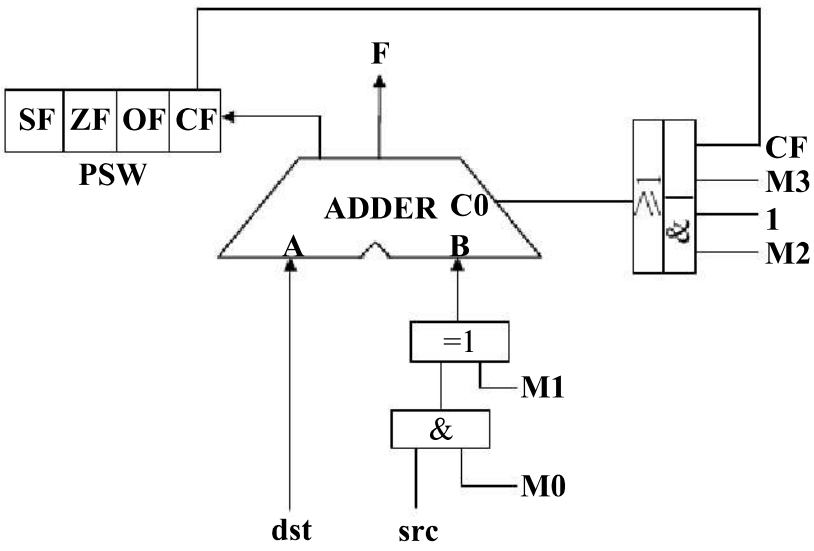
❖带借位的减法

- M3=1, M2=0, M1=1, M0=1

则

- B = ~src
- C0 = CF

∴ F = dst + ~src + CF



❖B = (src • M0) ⊕ M1

❖C0 = M2 + (M3 • CF)

❖F = A + B + C0

多功能加减运算电路

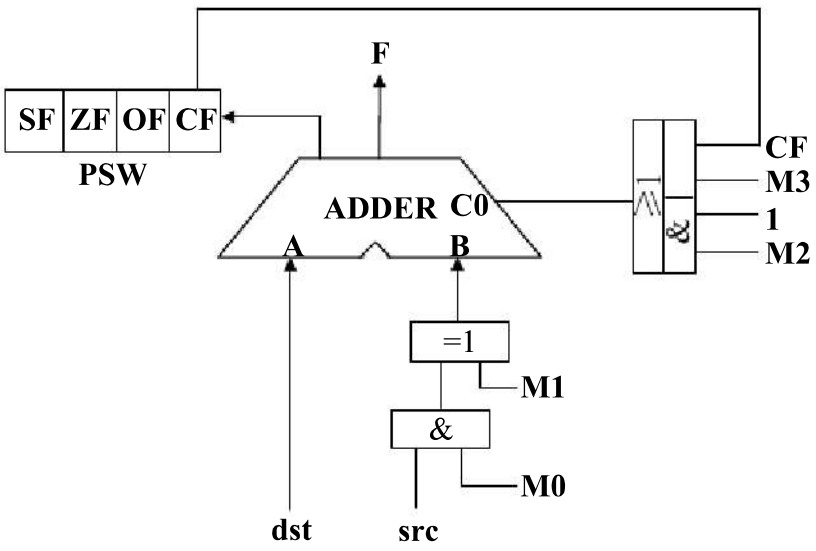
❖加1

- M3=0, M2=1, M1=0, M0=0

则

- B = 0
- C0 = 1

∴ F = dst + 1



❖B = (src • M0) ⊕ M1

❖C0 = M2 + (M3 • CF)

❖F = A + B + C0

多功能加减运算电路

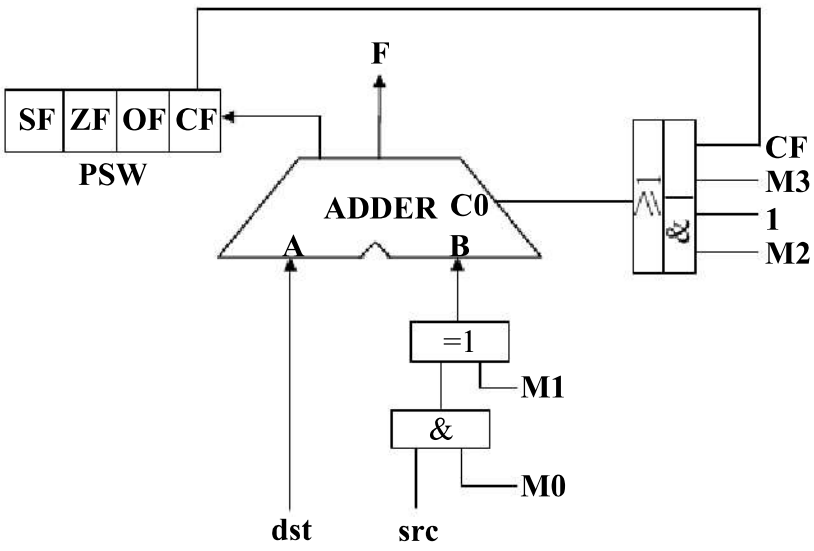
❖减1

- M3=0, M2=0, M1=1, M0=0

则

- B = -1（即111…1）
- C0 = 0

∴ F = dst - 1



❖B = (src • M0) ⊕ M1

❖C0 = M2 + (M3 • CF)

❖F = A + B + C0

多功能加减运算电路

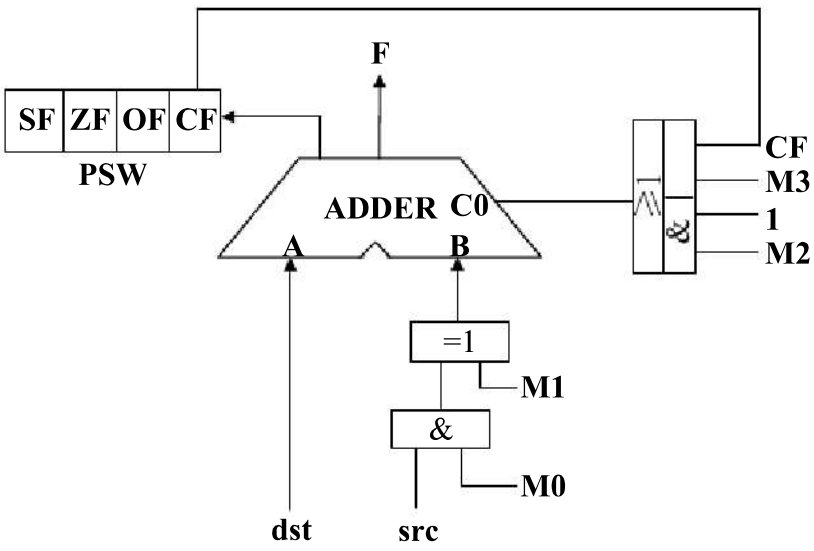
❖数据传送

- M3=0, M2=0, M1=0, M0=0

则

- B = 0
- C0 = 0

∴ F = dst



❖B = (src • M0) ⊕ M1

❖C0 = M2 + (M3 • CF)

❖F = A + B + C0

多功能加减运算电路

- B = (src • M0) ⊕ M1
- C0 = M2 + (M3 • CF)
- F = A + B + C0

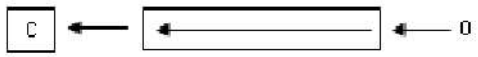
M3	M2	M1	M0	B	C0	F	
0	0	0	0	0	0	dst	数据传送
0	0	0	1	src	0	dst+src	加法
0	0	1	0	-1	0	dst-1	减一
0	1	1	1	~src	1	dst-src	减法
0	1	0	0	0	1	dst+1	加一
1	0	0	1	src	CF	dst+src+CF	带进位加法
1	0	1	1	~src	CF	dst+~src+CF	带借位减法

4.2 定点数移位运算

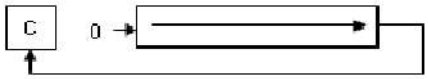
- 是计算机中基本运算之一
- 和加减运算相结合可实现乘除运算
- 包括逻辑移位、算术移位和循环移位

逻辑移位

- 逻辑左移
  - 将寄存器的每一位数据向左移动一个位置，最低位补0，最高位移至进位位；
- 逻辑右移
  - 将寄存器的每一位数据向右移动一个位置，最低位移至进位位，最高位补0。



(c)逻辑左移



(d)逻辑右移

例：11000101逻辑左移一位后得10001010。最高位移至进位位。

11000101逻辑右移一位后得01100010。最低位移至进位位

补码的算术移位

- 补码右移：各位右移，符号位复制。
  - 例：0.1101　右移　0.0110
  - 1.0110　右移　1.1011
  - 每右移一位，数值是原来的1/2。
- 补码左移：

  各位左移，最低位补0，符号位不变或移出。
  - 每左移一位，数值是原来的2倍（不溢出的情况下）。
  - 无论符号位如何处理，均可能造成溢出。
  - 如果采用双符号位，最高符号位表示数符，第二符号位可暂时用来表示数值位。

(a)算术左移　　(b)算术右移

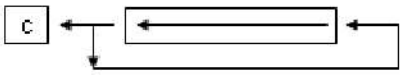
循环移位

循环左移：

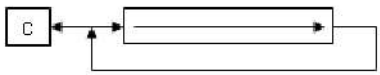
- 最高位移至进位位，同时移至最低位。

❖ 循环右移：

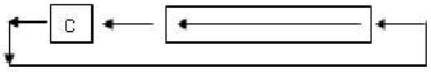
- 最低位移至进位位，同时移至最高位。



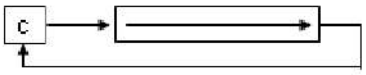
(e)循环左移



(f)循环右移



(g)带进位的循环左移



(h)带进位的循环右移

❖ 带进位的循环左移：

- 最高位移至进位位，进位位移至最低位。

❖ 带进位的循环右移：

- 最低位移至进位位，进位位移至最高位。

小结

- 用加法器实现补码加减法运算的原理
- 补码加法运算结果的溢出判断方法
- 程序状态字中各个标志位的含义
- 逻辑移位、算术移位

作业

3.5 分析超前进位并行加法器能提高运算速度的原因。

4.1

4.3

4.5