DSP Programmierung

Datenadressierung

Datenadressierung

Immediate Adressierung

```
start:  R0=123;       // Integer als Dezimalzahl
        R1=0x123;     // Integer als Hexadezimalzahl
        R2=-1;        // negativ Integer
        R3=0.75r;     // Festkommazahl "fractional"
        R4=-0.75r;    // negative Festkommazahl "fractional"
        R5=0.01e2;    // Gleitkommazahl "floating point"
        R6=10.0e-1;   // Gleitkommazahl "floating point"
        R7=-1.0;      // Gleitkommazahl "floating point"
```

Datenadressierung

Immediate Adressierung
/* Adresse einer Variablen */

```
B0=input1; //Adresse der Variablen
M0=1;
L0=3;
```

/* Adressieren des Vektors ivek */

```
B0=ivek;     // Vektoradresse
L0=@ivek; // Vektorlänge
```

Adressgenerator DAG

DAG1	*DAG2*
registers (32-bit)	*registers (24-bit)*
B0-B7	B8-B15
I0-I7	I8-I15
M0-M7	M8-M15
L0-L7	L8-L15

Adressgenerator DAG

DAG1	*DAG2*
registers (32-bit)	*registers (24-bit)*
B0-B7	B8-B15
I0-I7	I8-I15
M0-M7	M8-M15
L0-L7	L8-L15

Adressgenerator DAG

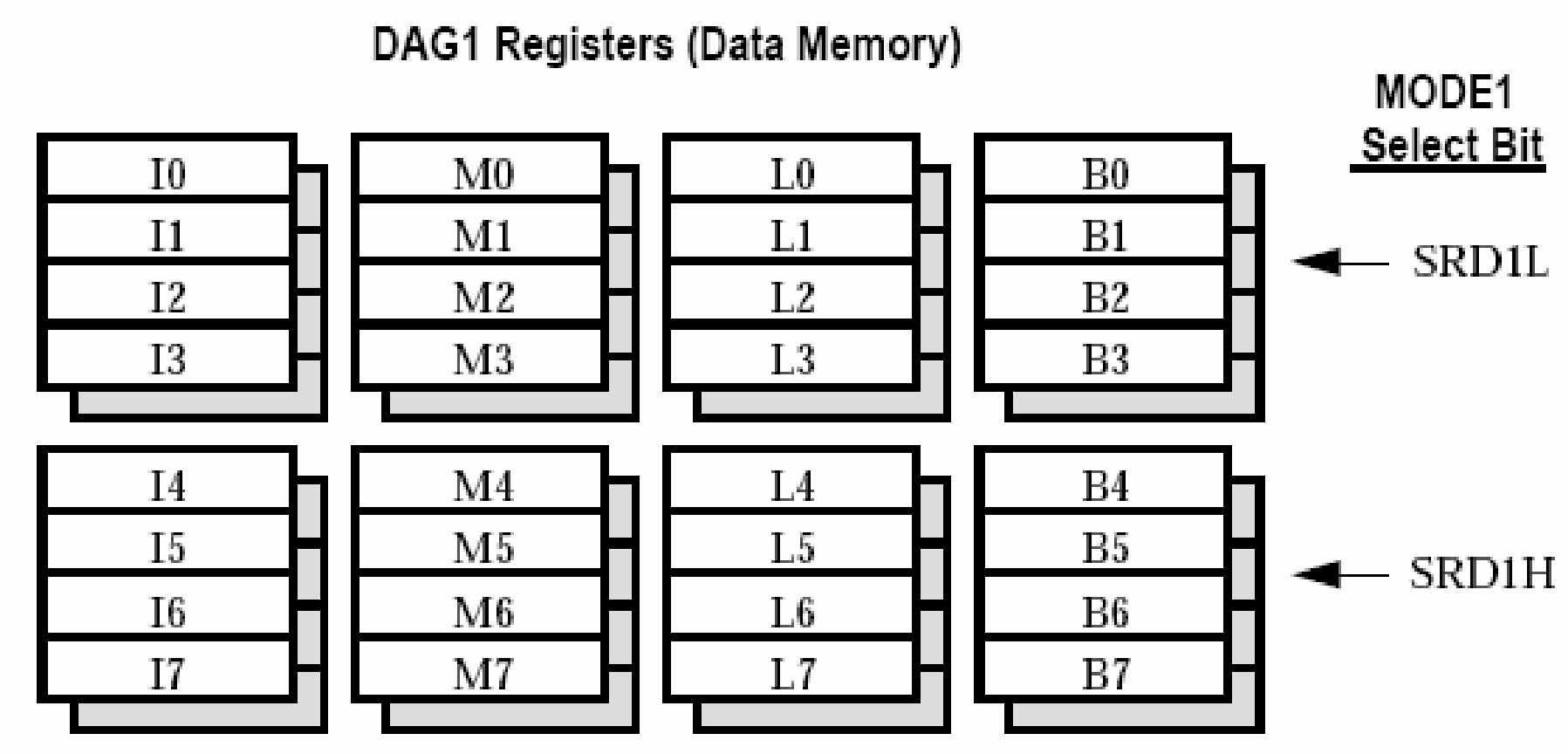
Register

Ix	*Index Register*	*Pointer zu Speicherzelle*
Mx	*Modify Register*	*modifiziert IX Register*
Bx	*Base Register*	*Vektoranfangsadresse*
Lx	*Length Register*	*Vektorlänge*

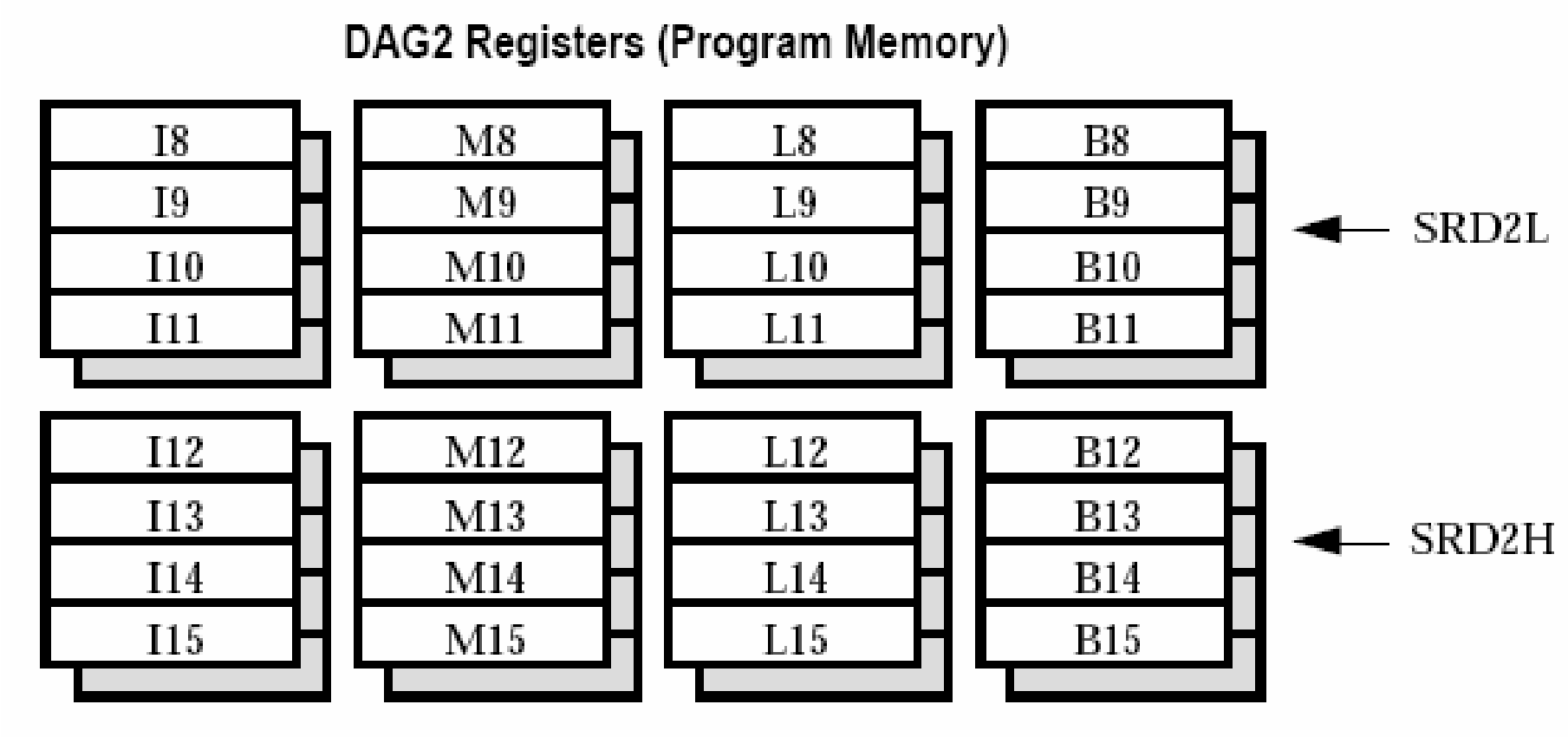
B und L Register werden nur bei Datenvektoren benötigt

Bei Initialisierung von Bx wird Ix ebenfalls initialisiert

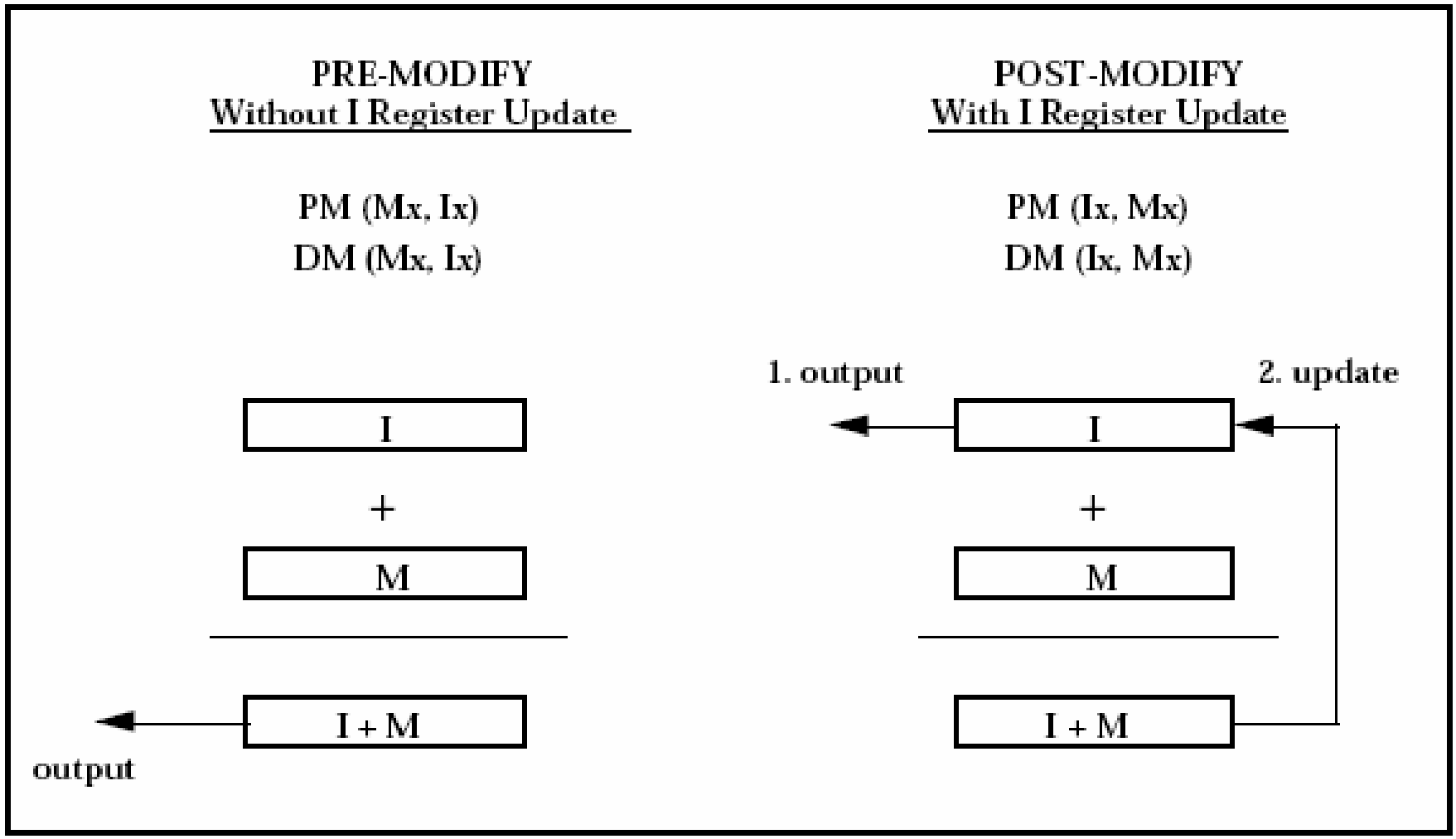
Adressgeneratoren



Adressgeneratoren



Adressmodifikation



Datenadressierung

Indirekte Adressierung mit Adressgenerator

Speicher lesen

R6 = PM(I15,M12); *Indirect addressing with post-modify*

R6 = PM(M12,I15); *Indirect addressing with pre-modify*

Datenadressierung

Indirekte Adressierung mit Adressgenerator

Speicher schreiben

DM(M0,I2) = TPERIOD; *Indirect addressing with pre-modify*

DM(I14,M12) = TPERIOD; *Indirect addressing with post-modify*

Datenadressierung

Indirekte Adressierung mit
Immediate Modifier

32-bit modifier:

R1=DM(0x40000000,I1);

DM address = I1 + 0x4000 0000

6-bit modifier:

F6=F1+F2,PM(I8,0x0B)=ASTAT;

PM address = I8, I8 = I8 + 0x0B

Datenadressierung

Programmbeispiel

Im Daten- und Programmspeicher werden initialisierte und nicht initialisierte Variablen angelegt. Die Werte der initialisierten Variablen werden in die nicht initialisierten Variablen kopiert

Datenadressierung

```
.SECTION /dm seg_dmda;
.VAR dmSkalar=0x55;
.VAR dmVectori[4];
.VAR dmVectorf[6]= 1.0,2e2,-0.3e-2,123.456,0.0,777.8;
.VAR dminput[]="Werte.dat";

.SECTION /pm seg_pmda;
.VAR pmSkalar;
.VAR pmVectori[4]=100,234,12345,100000000;
.VAR pmVectorf[6];
.VAR pmkopie[9];
.VAR pmkopiek[3];
```

Datenadressierung

```
.SECTION /dm seg_dmda;
.VAR dmSkalar=0x55;
.VAR dmVectori[4];
.VAR dmVectorf[6]= 1.0,2e2,-0.3e-2,123.456,0.0,777.8;
.VAR dminput[]="Werte.dat";

.SECTION /pm seg_pmda;
.VAR pmSkalar;
.VAR pmVectori[4]=100,234,12345,100000000;
.VAR pmVectorf[6];
.VAR pmkopie[9];
.VAR pmkopiek[3];
```

Datenadressierung

```
.SECTION /pm seg_rth;
        nop;
.align 0x4;
        nop;
        jump start;
```

Datenadressierung

```
.SECTION /pm seg_pmco;
/*      1.Teil
   Kopieren eines Wertes von dmSkalar nach
   pmSkalar Wert durch direkte Adressierung
   lesen und schreiben
   direkte Adressierung */

start:      R8=DM(dmSkalar);
            PM(pmSkalar)=R8;
```

Datenadressierung

```
/*  2.Teil
    Kopieren der Werte von pmVectori nach dmVectori
    1. Adressgeneratoren einstellen
    2. Werte in einer Schleife Kopieren
*/
            B0=dmVectori;
            M0=1;
            L0=@dmVectori;
            B8=pmVectori;
            M8=1;
            L8=@pmVectori;
loop1a:     LCNTR=L8, DO loop1e UNTIL LCE;
            R0=PM(I8,M8);
loop1e:     DM(I0,M0)=R0;
```

Datenadressierung

```
/*    3.Teil lesen der Werte einesVektors und schreiben der gelesenen Werte in zwei Vektoren
      Kopieren der Werte von dminput nach pmkopie und pmkopiek
      1. Adressgeneratoren einstellen
      2. Werte in einer Schleife Kopieren
      Die Dimension des Ziel-Vektors pmkopiek ist kleiner als die Dimension des Quell-Vektors dminput
                                                    */
            B2=dminput;
            M2=1;
            L2=@dminput;
            B10=pmkopie;
            M10=1;
            L10=@pmkopie;
            B11=pmkopiek;
            L11=@pmkopiek;
loop2a:     LCNTR=L2, DO loop2e UNTIL LCE;
            R0=DM(I2,M2);
            PM(I10,M10)=R0;
loop2e:     PM(I11,M10)=R0;
```

Datenadressierung

/* 4.Teil parallel lesen und schreiben
Kopieren der Werte von dmVectorf nach pmVectorf
1. Adressgeneratoren einstellen
2. Werte in einer Schleife Kopieren
    1. Wert vor der Schleife lesen
    in der Schleife n-1-ten Wert schreiben
    und parallel dazu n-ten Wert lesen
    nach der Schleife n-ten Wert schreiben
*/

Datenadressierung

```
                B1=dmVectorf;
                M0=1;
                L1=@dmVectorf;
                B9=pmVectorf;
                M9=1;
                L9=@pmVectorf;
                R1=@pmVectorf-1;
                R0=DM(I1,M0);
loop3a:         LCNTR=R1, DO loop3e UNTIL LCE;
loop3e:         R0=DM(I1,M0), PM(I9,M9)=R0;
                PM(I9,M9)=R0;
                idle;
```