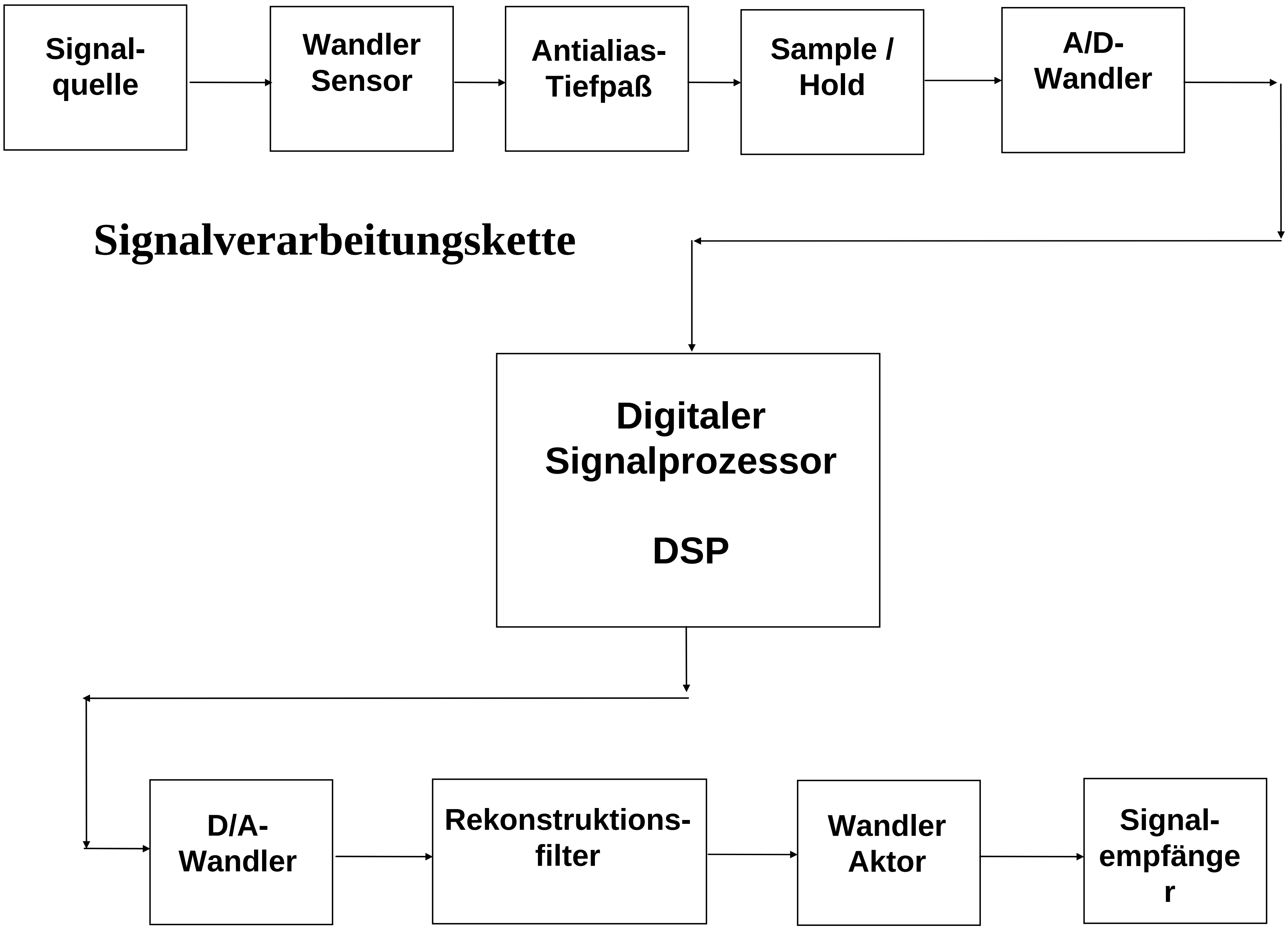
Spezielle Prozessoren

Digitale Signalprozessoren
DSP

Signalverarbeitungskette

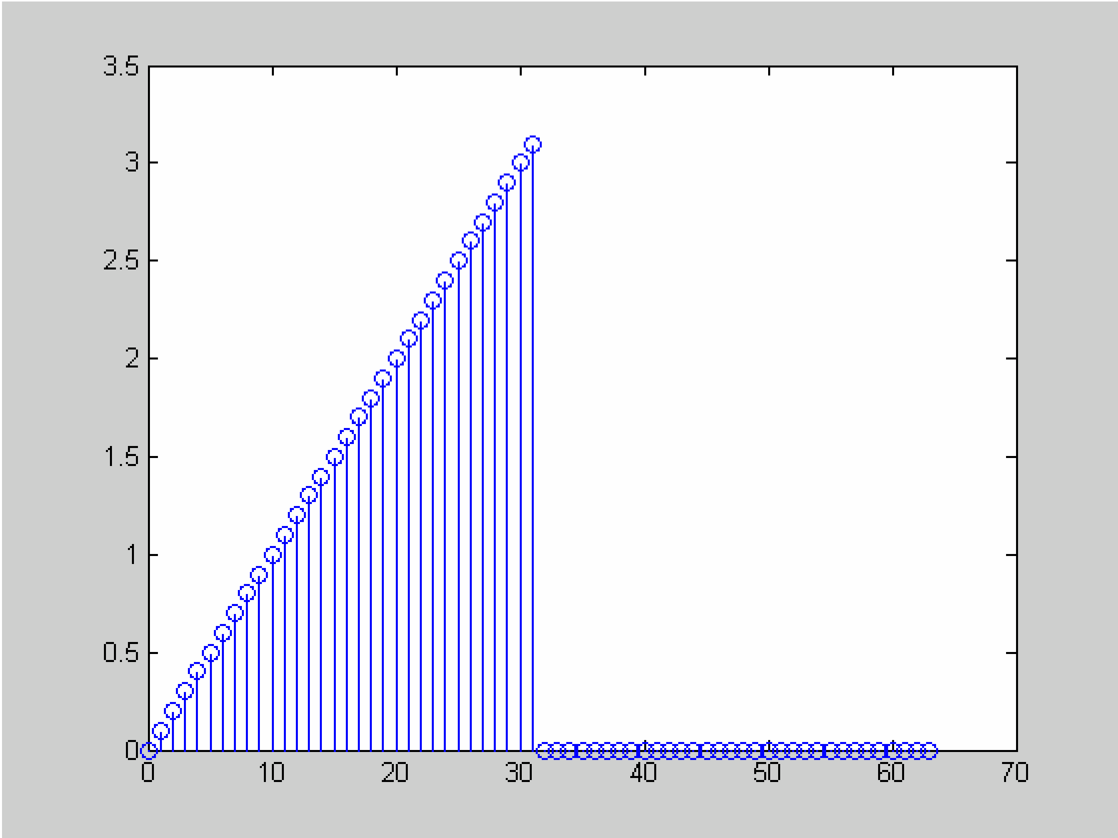


Diskrete Fouriertransformation

$$F(m) = \sum_{n=0}^{N-1} f(n) \cdot e^{-\frac{jmn2\Pi}{N}}$$

$$f(n) = \frac{1}{N} \cdot \sum_{m=0}^{N-1} F(m) \cdot e^{\frac{jmn2\Pi}{N}}$$

Bsp. Diskrete Fouriertransformation diskretes Signal

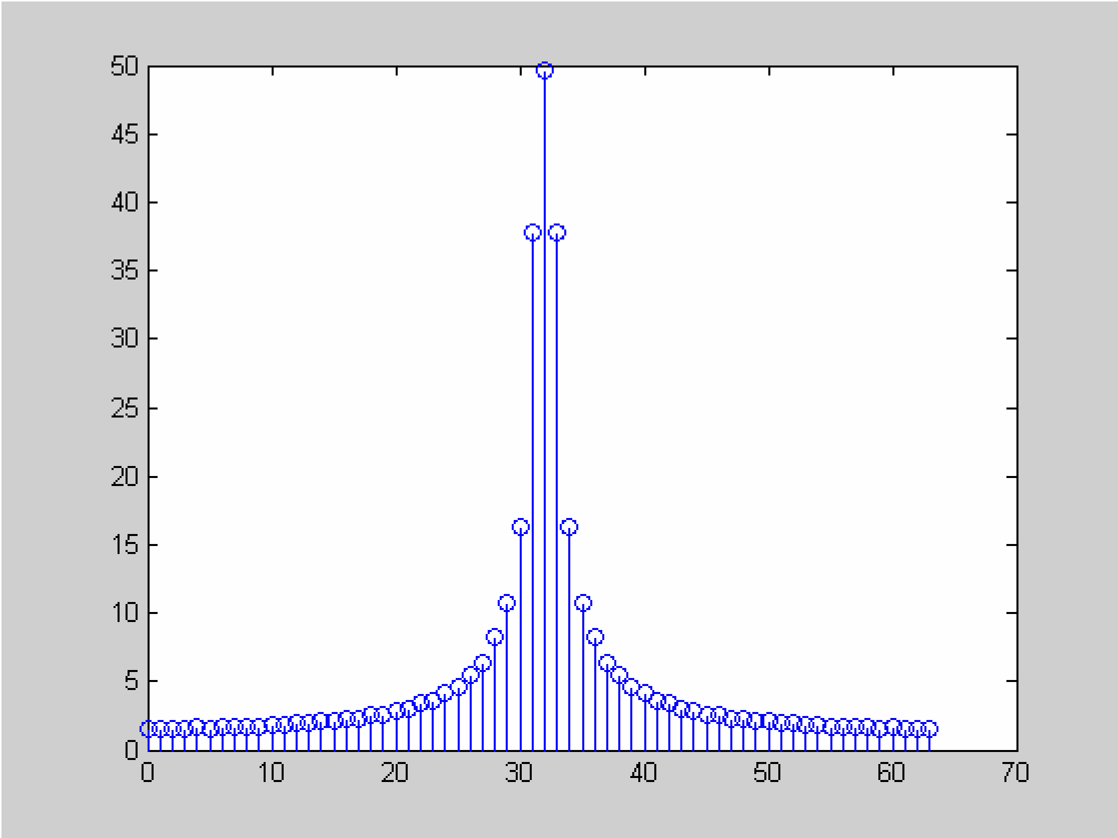


Bsp. Diskrete Fouriertransformation diskretes Signal

Diskrete Fouriertransformation in MATLAB

```
freq=fftshift(fft(sig))
stem(unabh,abs(freq))
```

Bsp. Diskrete Fouriertransformation
diskrete Frequenzen



Diskrete Faltung

$$y(n) = \sum_{k=-\infty}^{\infty} x(k) \cdot g(n-k) = x(n) * g(n) = \sum_{k=-\infty}^{\infty} g(k) \cdot x(n-k)$$

Diskrete Korrelation

$$k(n) = \sum_{k=-\infty}^{\infty} y(k) \cdot x(k+n)$$

Differenzengleichung

$$\sum_{k=0}^{N} a_k \, y(n-k) = \sum_{k=0}^{M} b_k \, x(n-k)$$

Differenzengleichung rekursiv

$$y(n) = \frac{1}{a_0} \cdot \left(\sum_{k=0}^{M} b_k x(n-k) - \sum_{k=1}^{N} a_k y(n-k) \right)$$

Bsp IIR - Tiefpass

Bestimmung der Filterkoeffizienten mit MATLAB

[bk,ak]=butter(4,0.001)

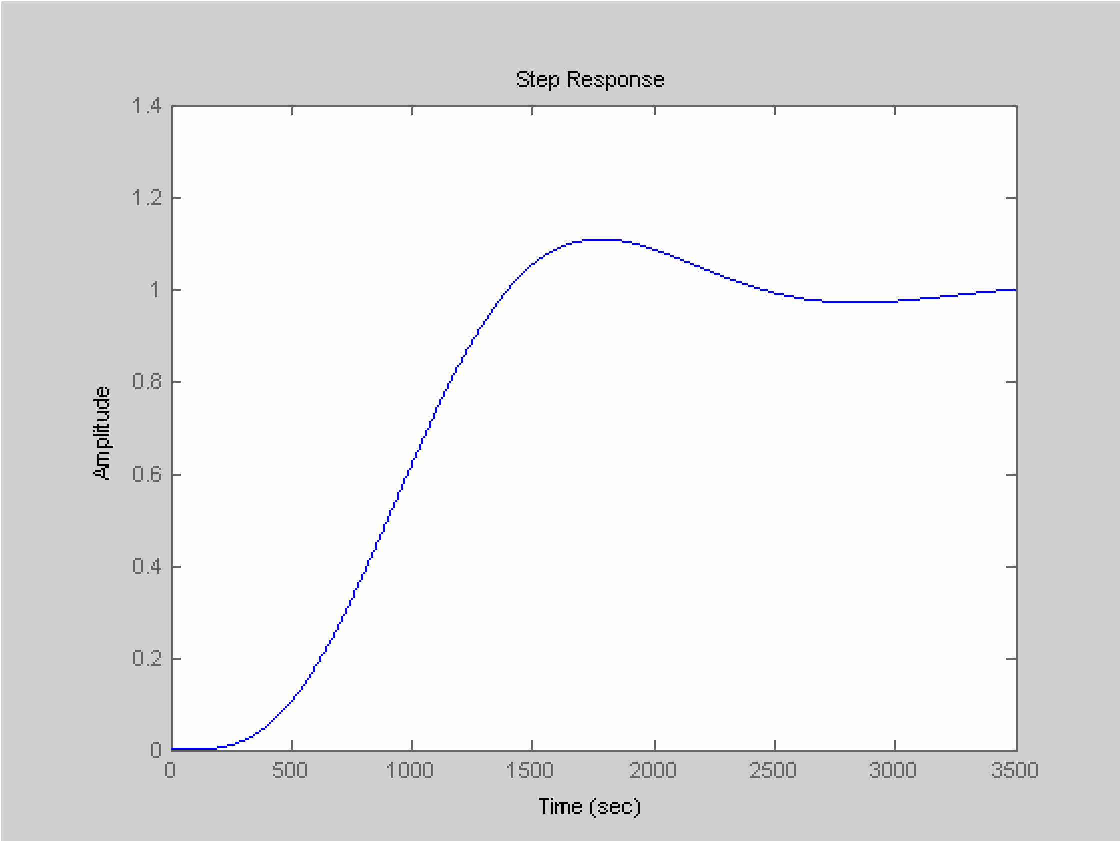
bk=1.0e-010 *

0.0606   0.2425   0.3638   0.2425   0.0606

ak=1.0000 -3.9918   5.9754 -3.9754   0.9918

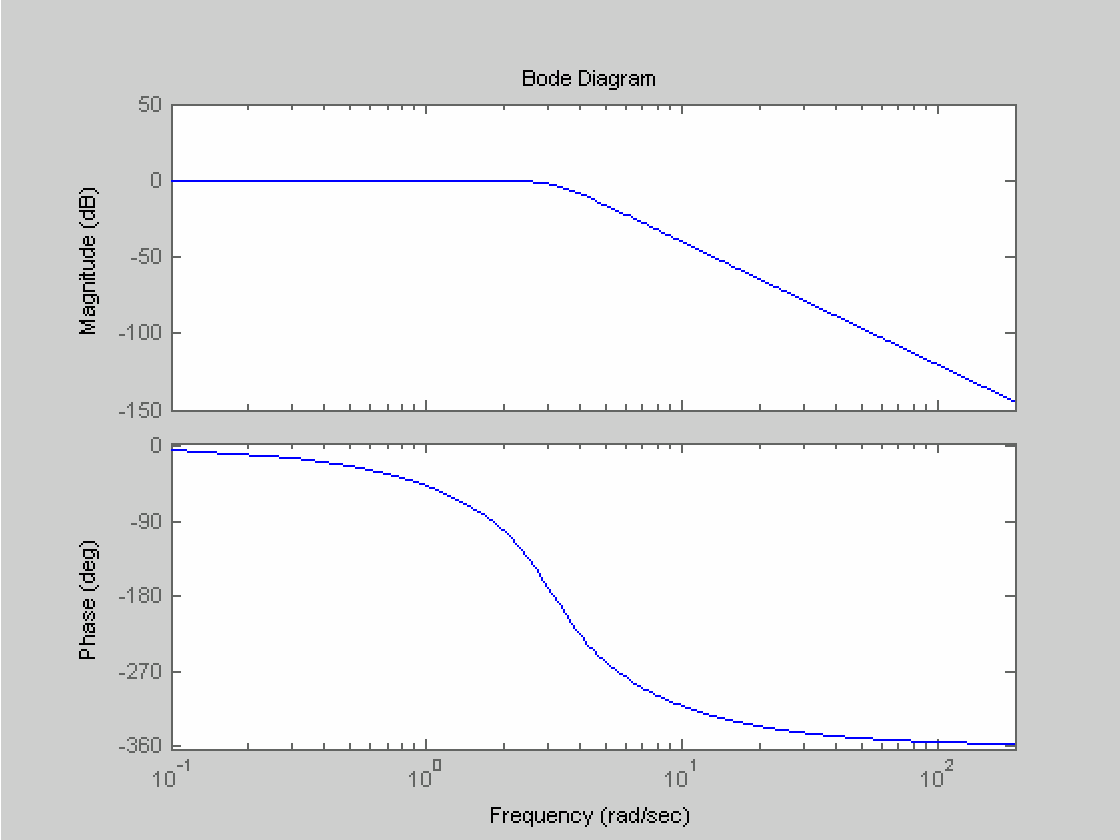
Bsp IIR - Tiefpass

dstep(bk,ak)



Bsp IIR - Tiefpass

dbode(zbut,nbut,0.001,w)



Differenzengleichung
nicht rekursiv

$$y(n) = \sum_{k=0}^{M} \frac{b_k}{a_0} \cdot x(n-k)$$

Bsp FIR-Tiefpass

Bestimmung der Filterkoeffizienten mit Hilfe von MATLAB

```
zfir=fir1(8,0.01)
```

Bsp FIR-Tiefpass

Filterkoeffizienten aus Matlab sind

Nennerkoeffizienten für G(z)

bfir =

0.0181 0.0488 0.1227 0.1967
0.2274 0.1967 0.1227 0.0488

0.0181

Bsp FIR-Tiefpass

Nennerkoeffizienten für G(z)

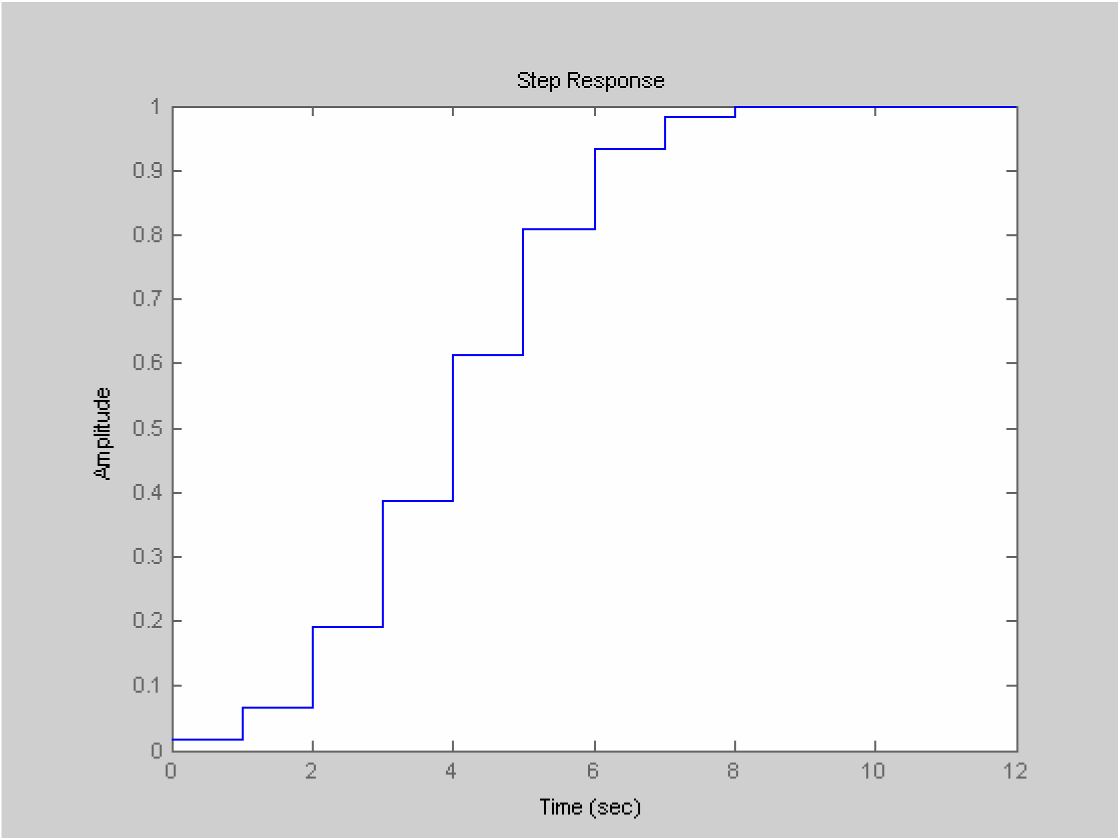
nfir =

1	0	0	0	0	0	0	0	0

Bsp FIR-Tiefpass

Diskrete Sprungantwort in MATLAB

dstep(zfir,nfir)



Matrizenmultiplikation

$$p_{ij} = \sum_{k=1}^{m} a_{ik} \cdot b_{kj}$$

Zweidimensionale Faltung

$$P_N(i,j) = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} P_A(k,l)\cdot G_2(i-k, j-l)$$

Zweidimensionale Kreuzkorrelation

$$K_{t,f}(i,j) = \sum_{k=-m}^{m} \sum_{l=-n}^{n} T(k,l) \cdot O(i-k, j-l)$$

Ergebnis

Ähnlicher Algorithmus für eine Vielzahl unterschiedlicher Aufgaben

- Multiplikation und Akkumulation

Anforderungen an die Architektur

- Harvardarchitektur
  - Gleichzeitiger Zugriff auf zwei Operanden aus dem Speicher
- Befehlsabarbeitung in einem Takt / Befehl
- Dreiadressmaschine bei Registeradressierung

  erg=erg+op1*op2

Anforderungen an die Architektur

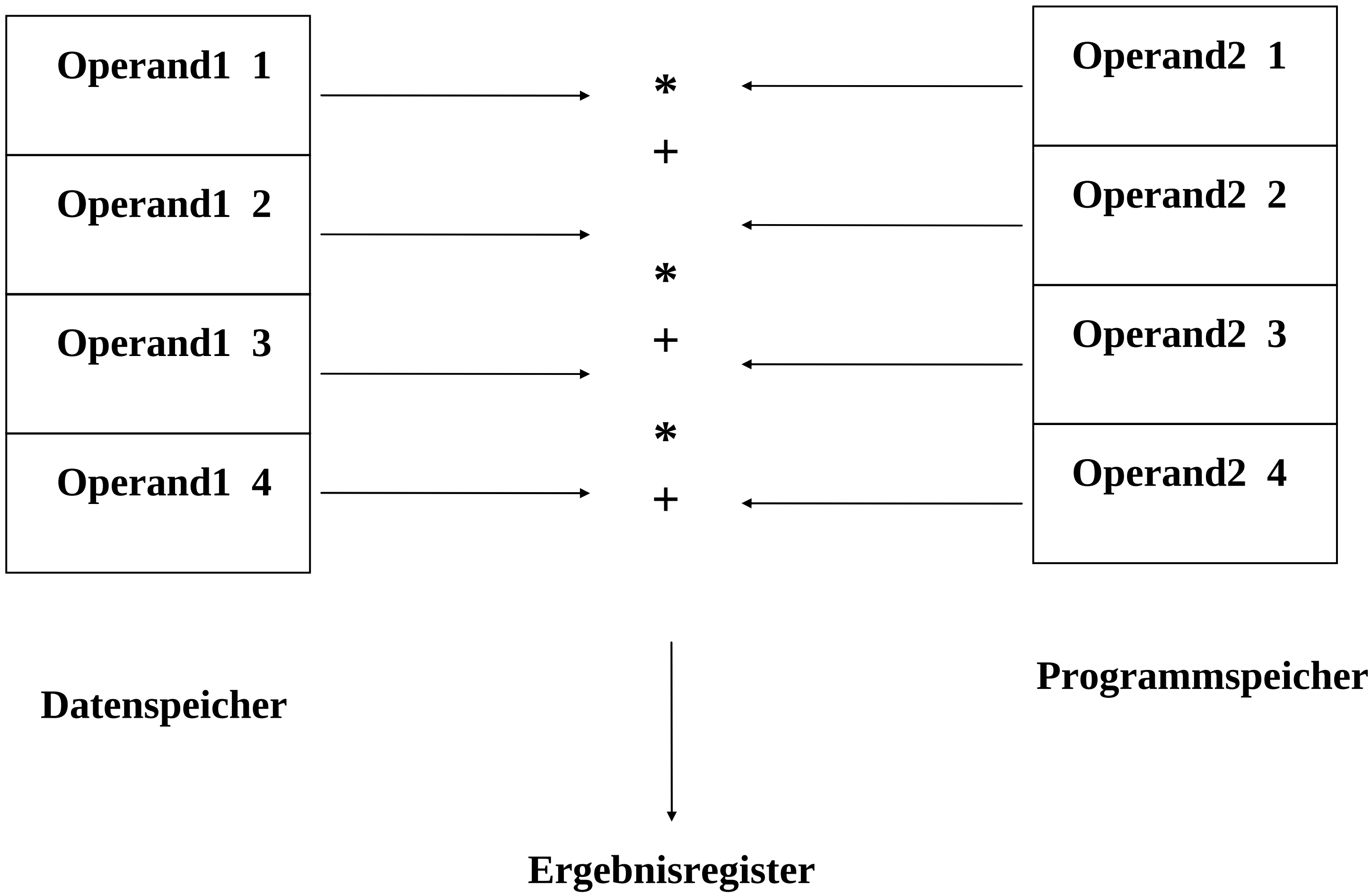
- Hardwaremultiplizierer

Histogramm

$$H(P(k,l) = H(P(k,l)+1$$

$$k = 0...M-1 \quad l = 0...N-1$$

DSP Algorithmen



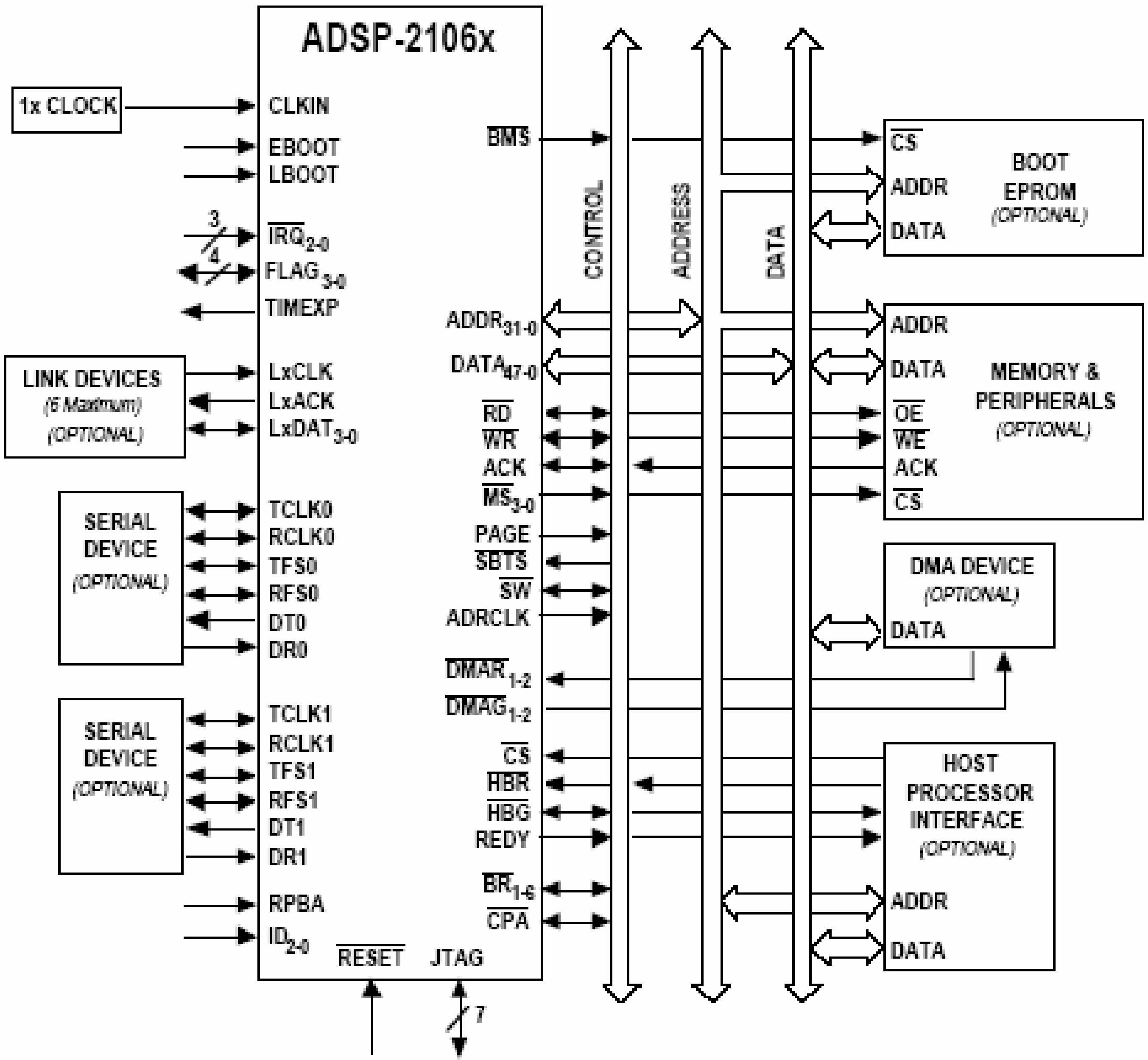
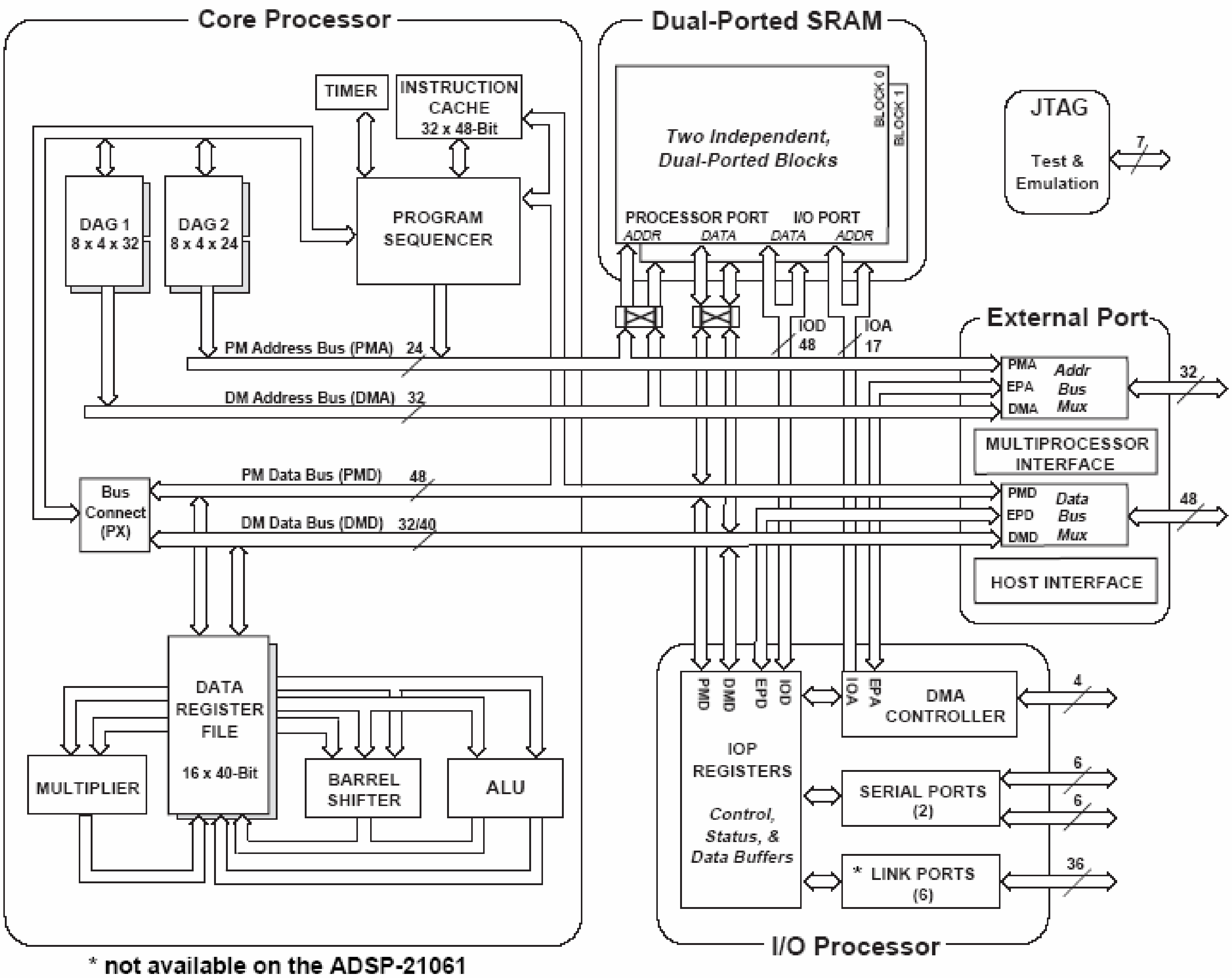


Figure 1.3 ADSP-2106x System



* not available on the ADSP-21061

Figure 1.2 ADSP-2106x SHARC Block Diagram

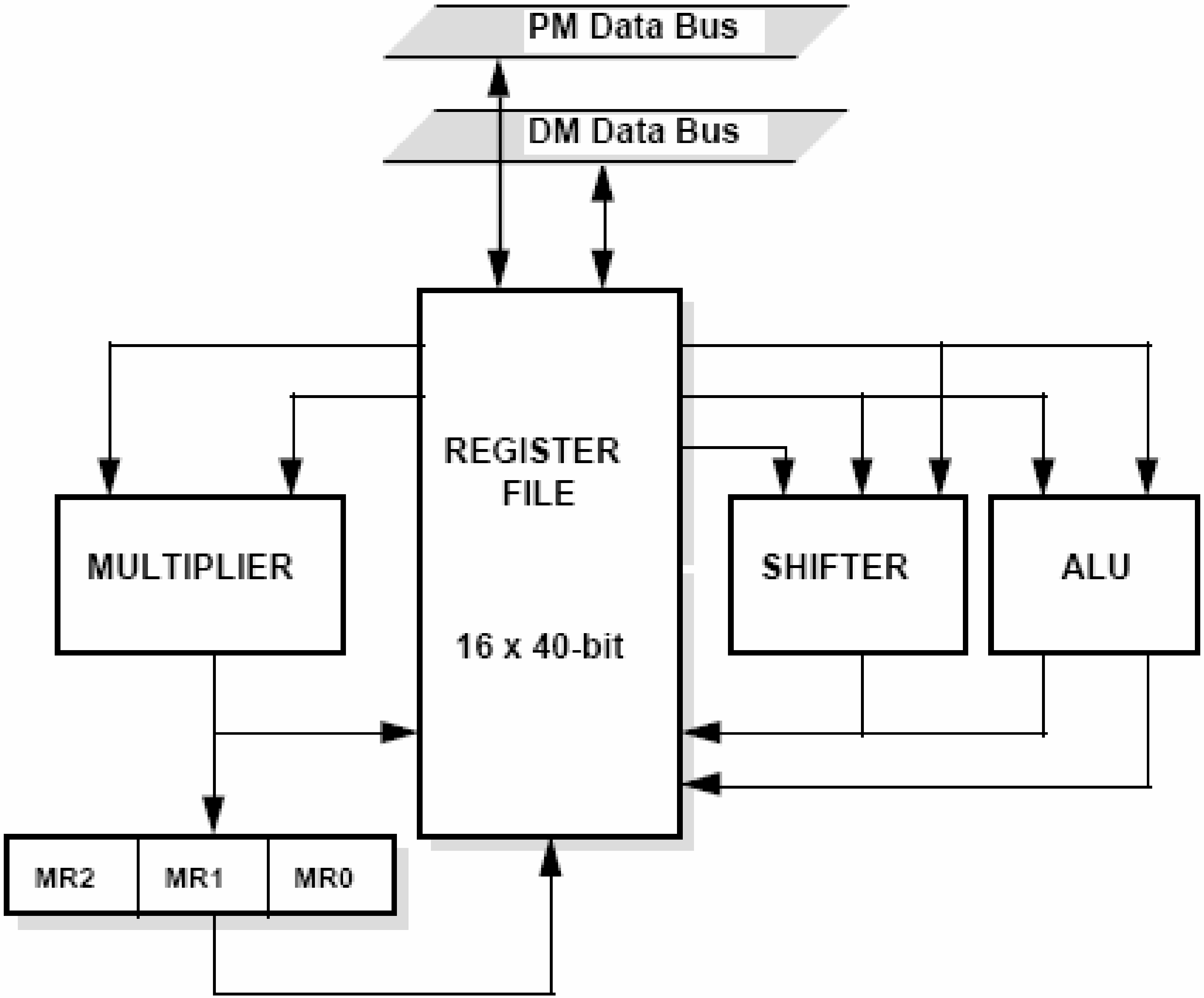
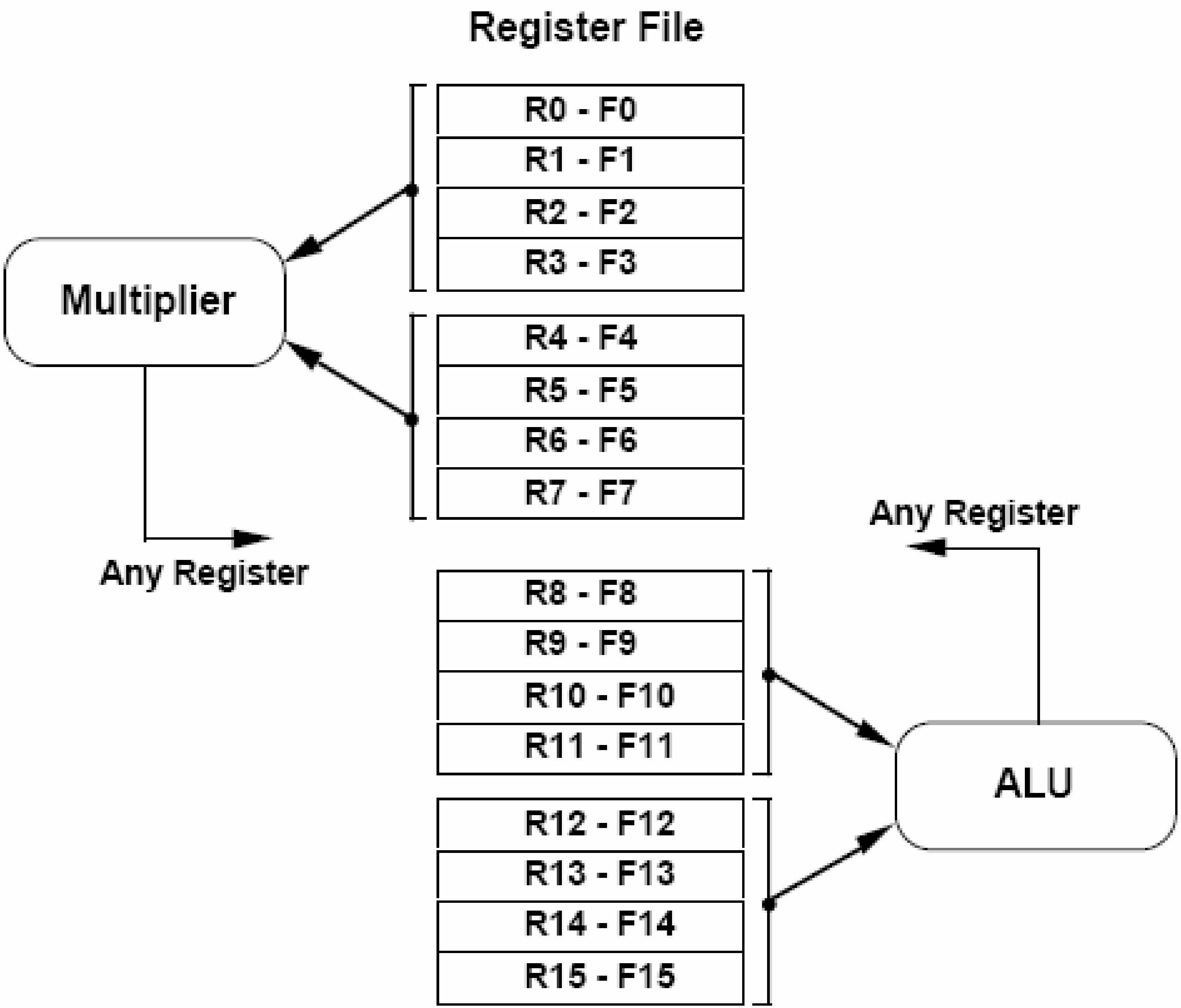
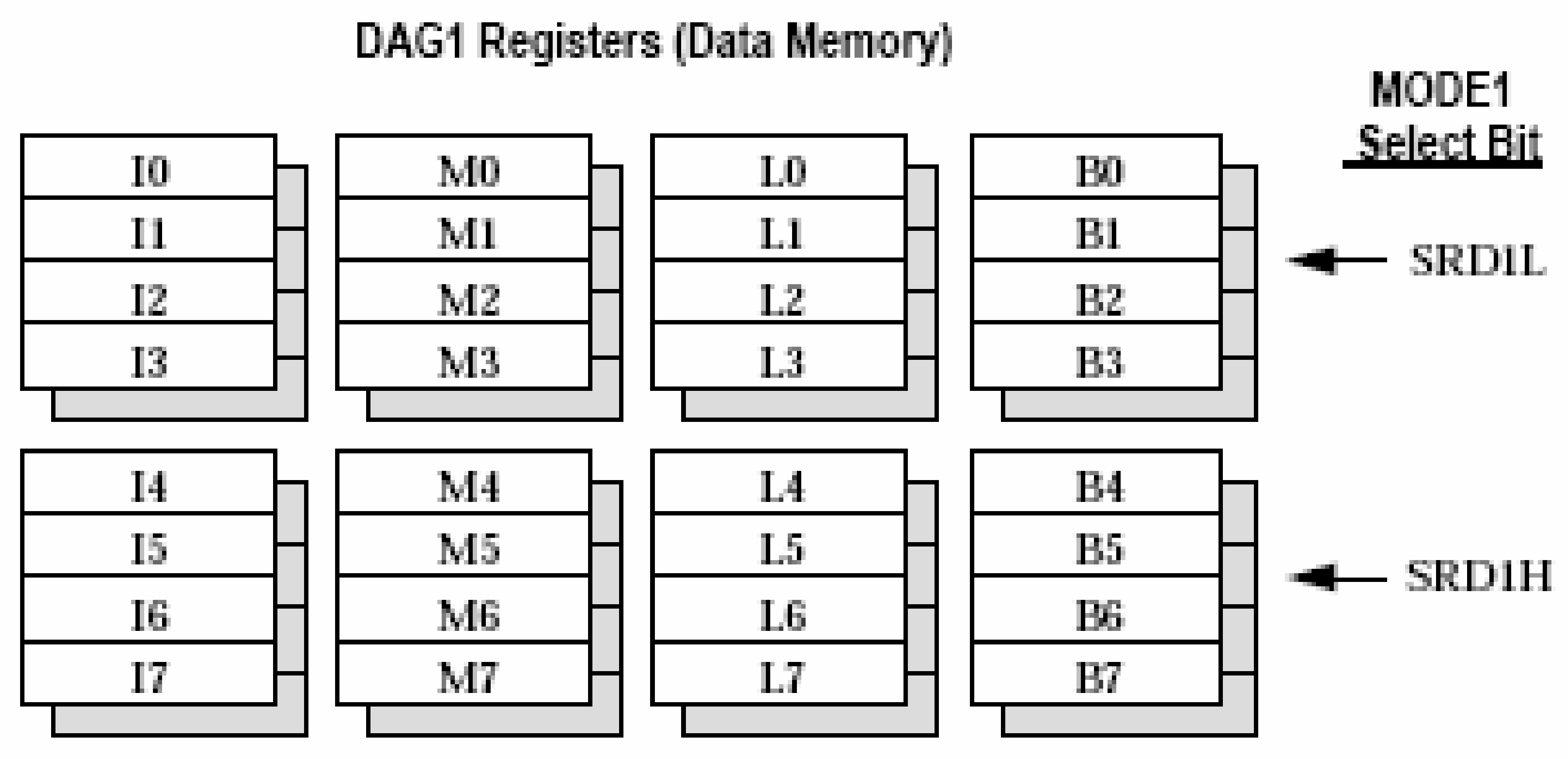
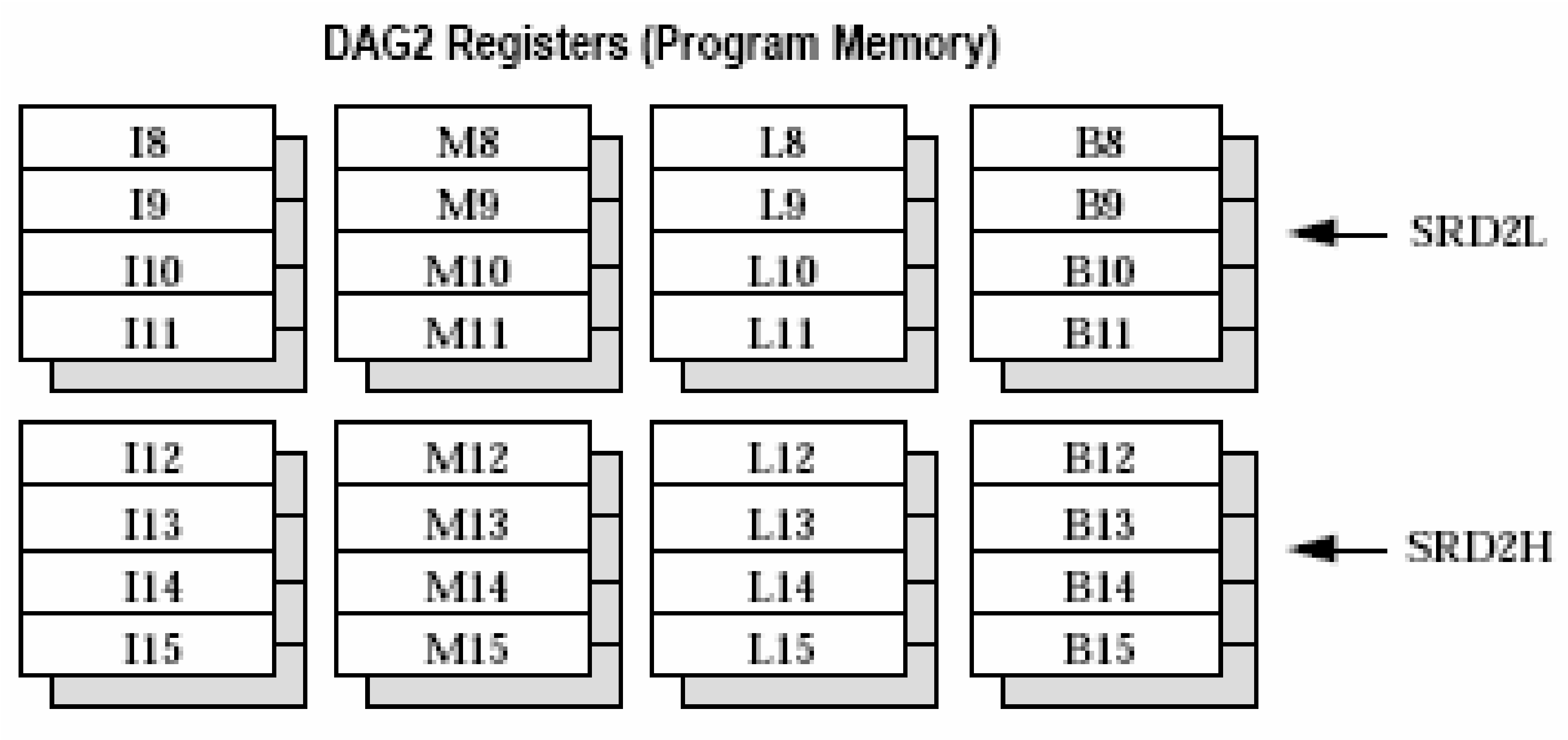


Figure 2.1 Computation Units

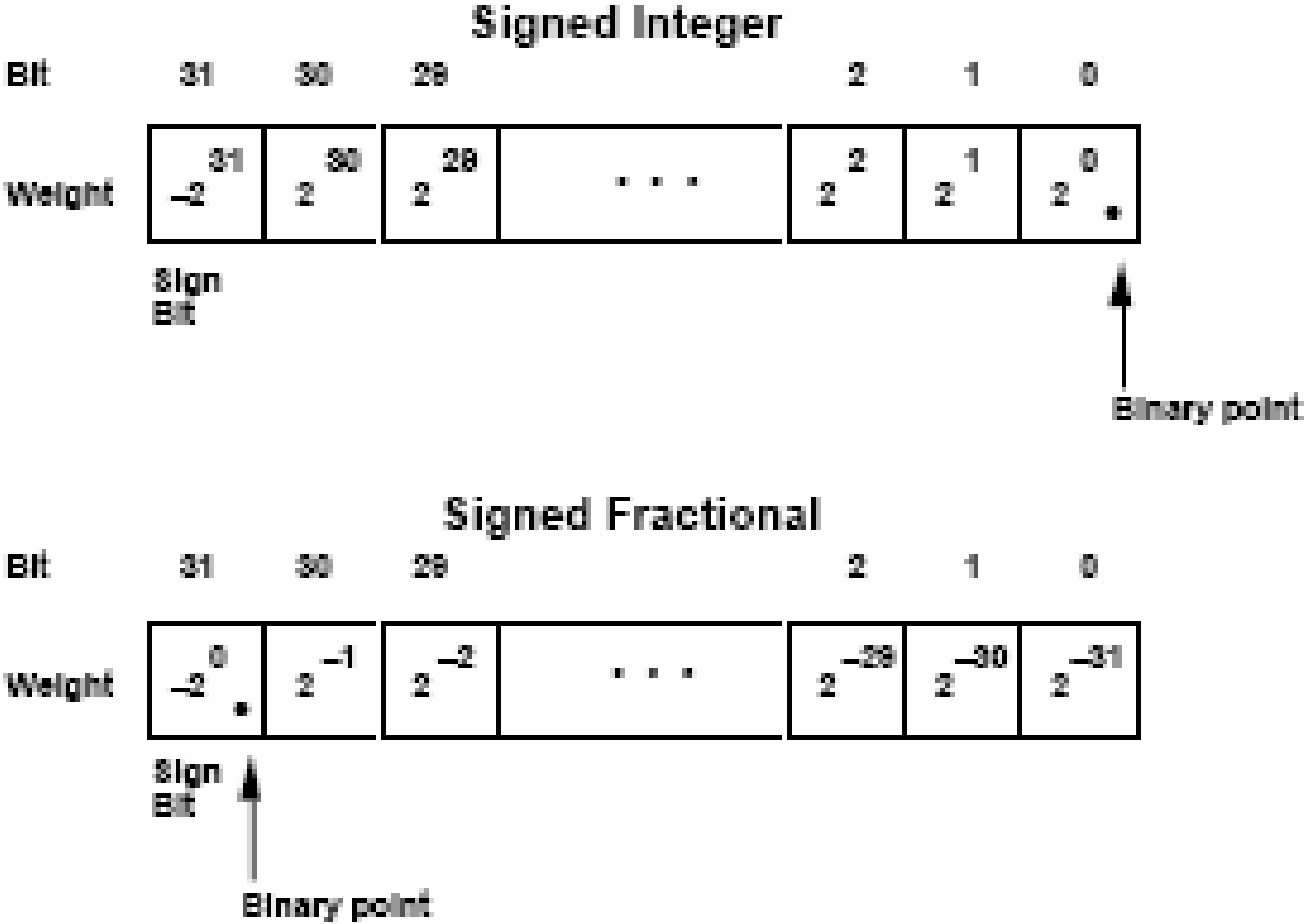


Register File
R0 - F0
R1 - F1
R2 - F2
R3 - F3
R4 - F4
R5 - F5
R6 - F6
R7 - F7
Multiplier
Any Register
R8 - F8
R9 - F9
R10 - F10
R11 - F11
R12 - F12
R13 - F13
R14 - F14
R15 - F15
Any Register
ALU

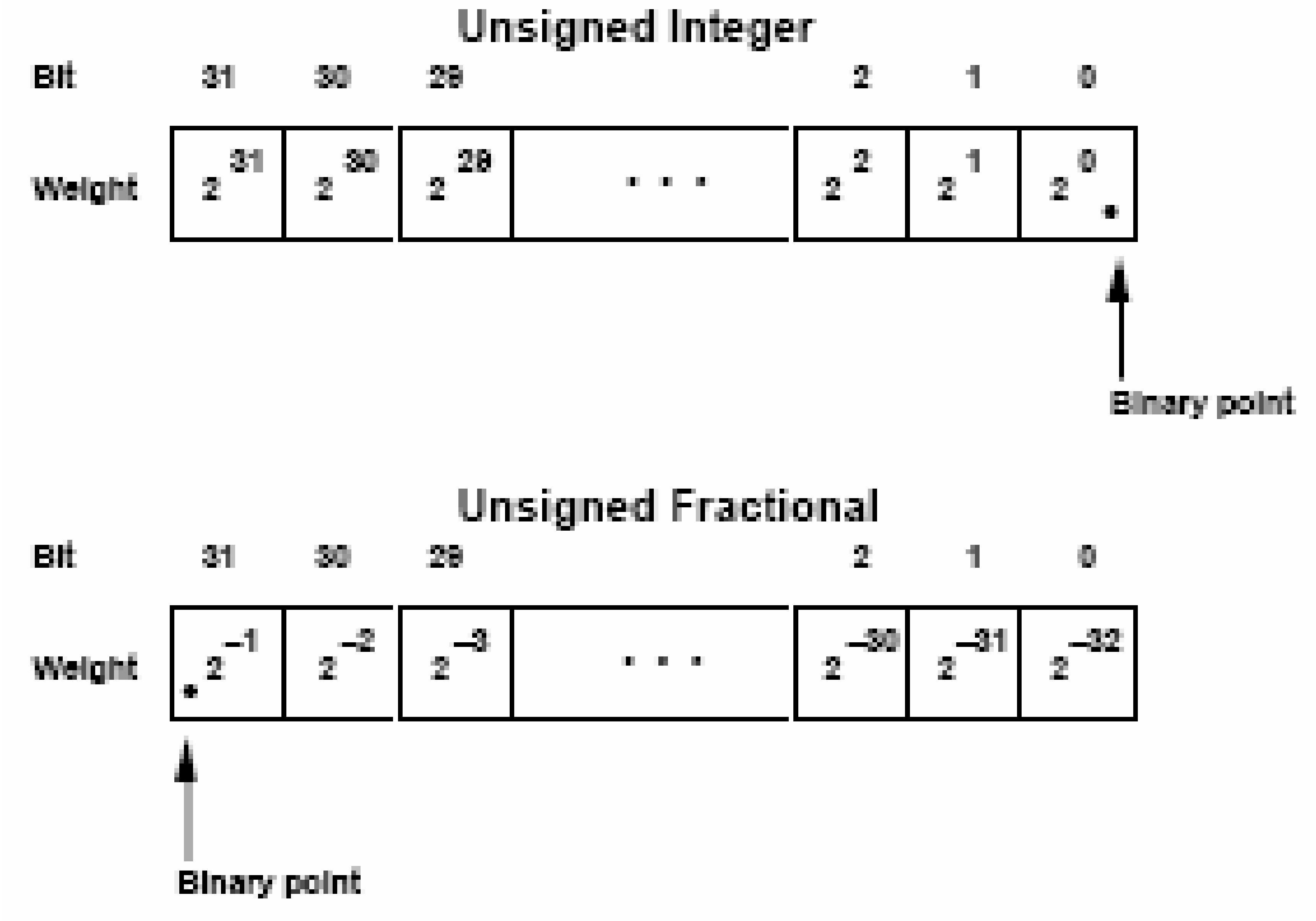

DAG1 Registers (Data Memory)
MODE1
Select Bit
I0
I1
I2
I3
M0
M1
M2
M3
L0
L1
L2
L3
B0
B1
B2
B3
SRD1L
I4
I5
I6
I7
M4
M5
M6
M7
L4
L5
L6
L7
B4
B5
B6
B7
SRD1H


DAG2 Registers (Program Memory)
I8
I9
I10
I11
M8
M9
M10
M11
L8
L9
L10
L11
B8
B9
B10
B11
SRD2L
I12
I13
I14
I15
M12
M13
M14
M15
L12
L13
L14
L15
B12
B13
B14
B15
SRD2H

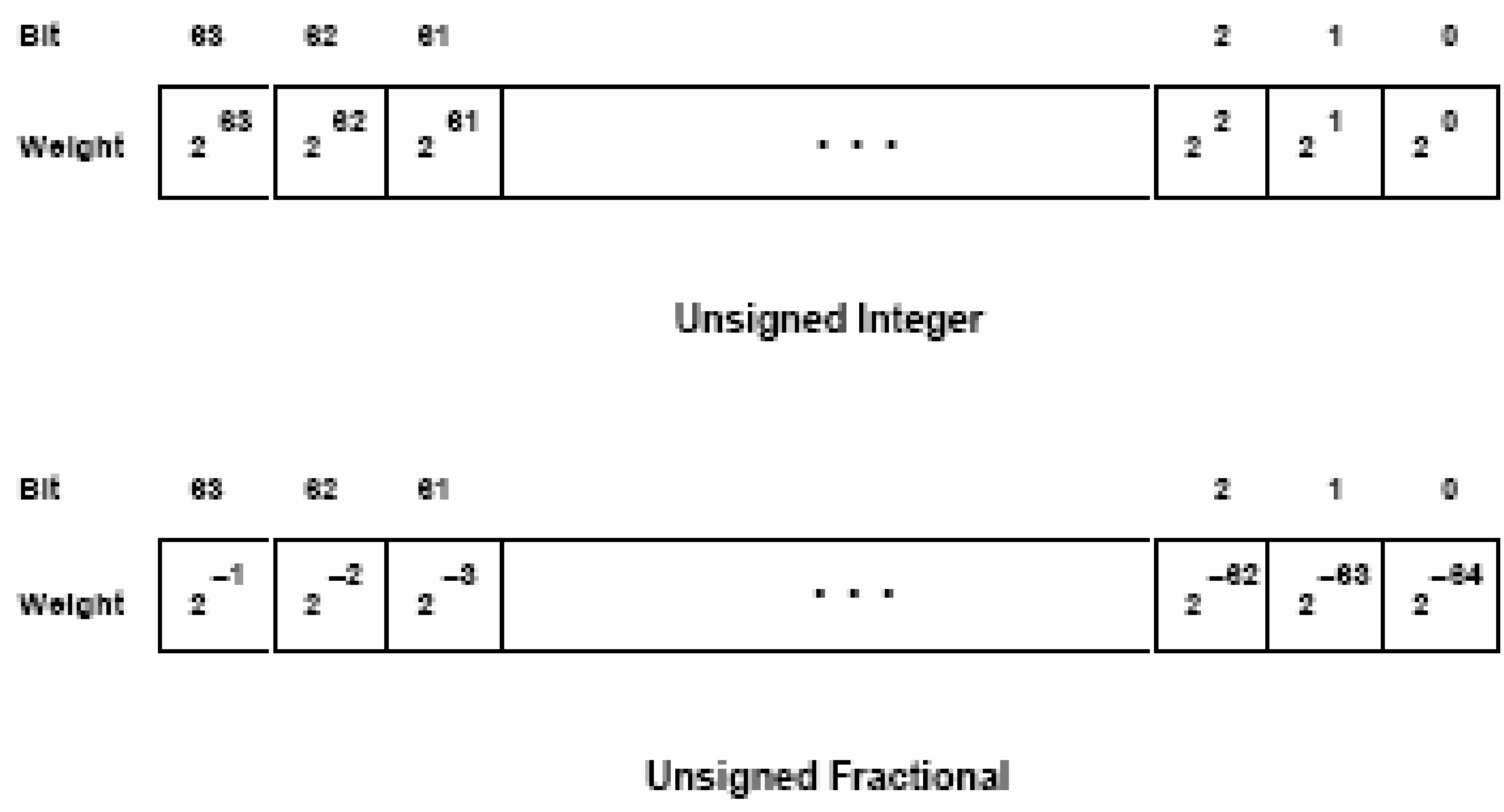
Zahlenformate Festkomma



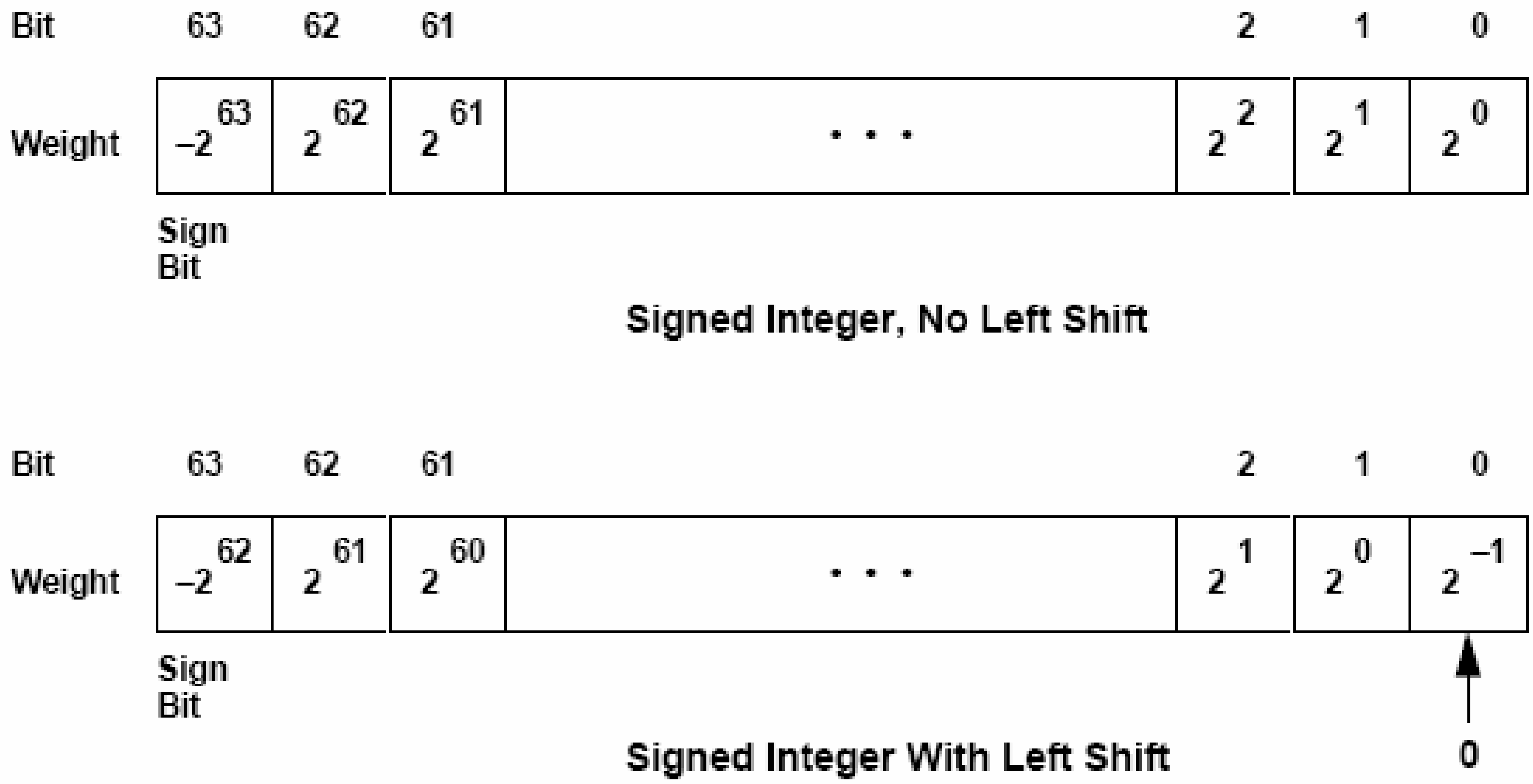
Zahlenformate Festkomma



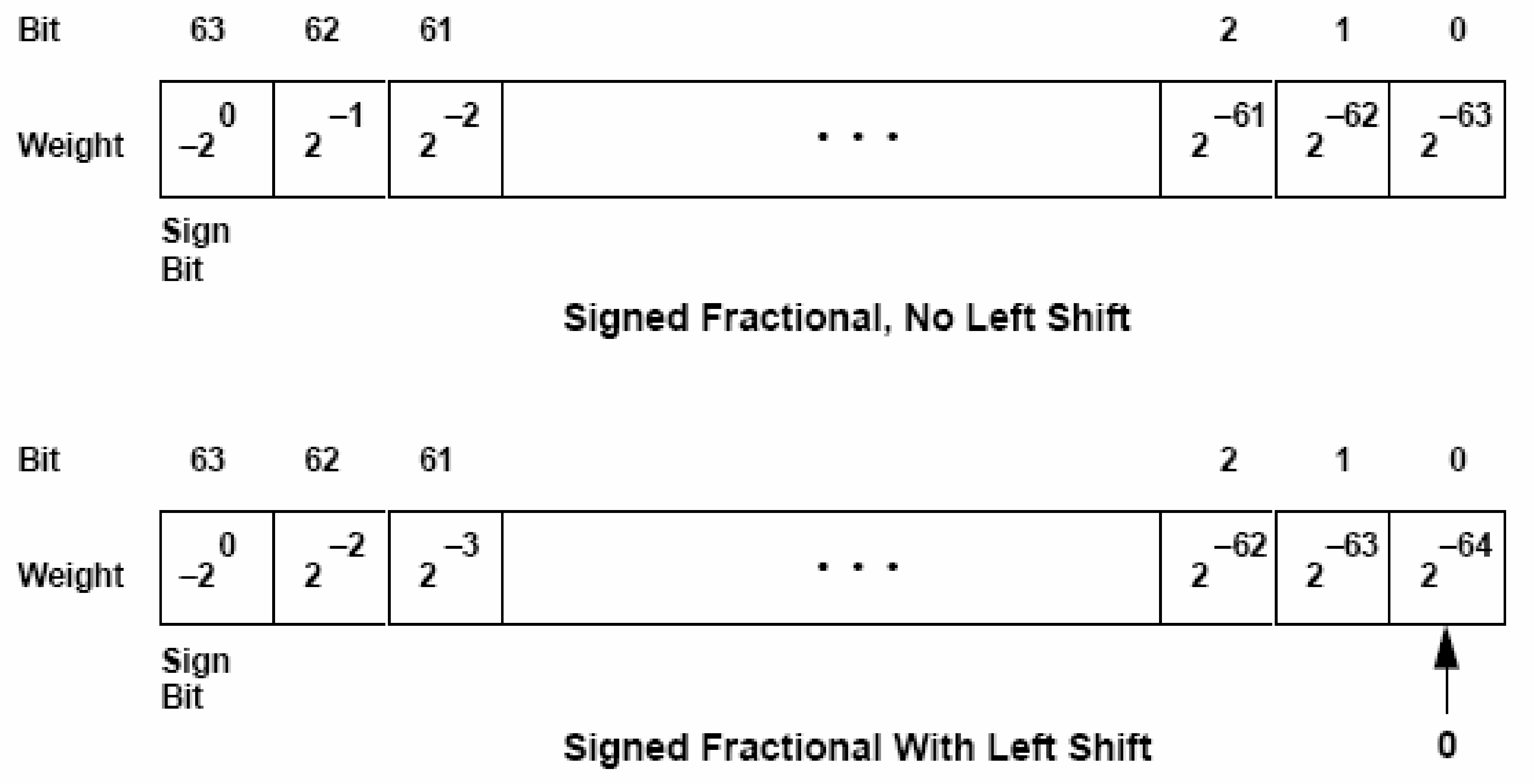
Zahlenformate Festkomma



Zahlenformate Festkomma



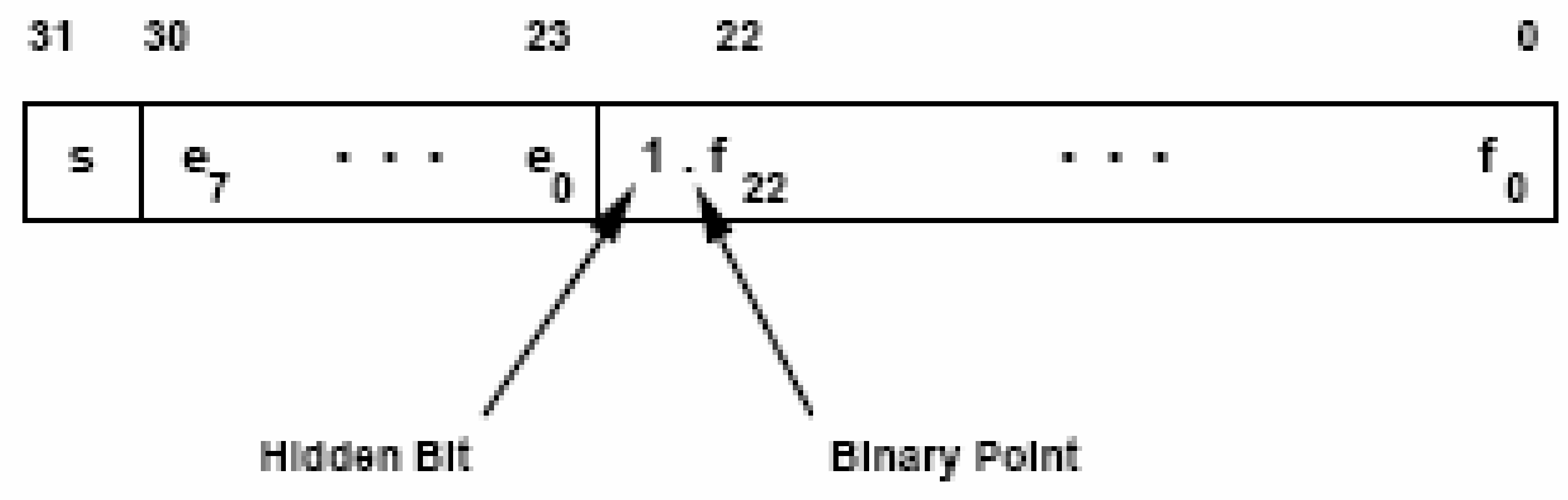
Zahlenformate Festkomma



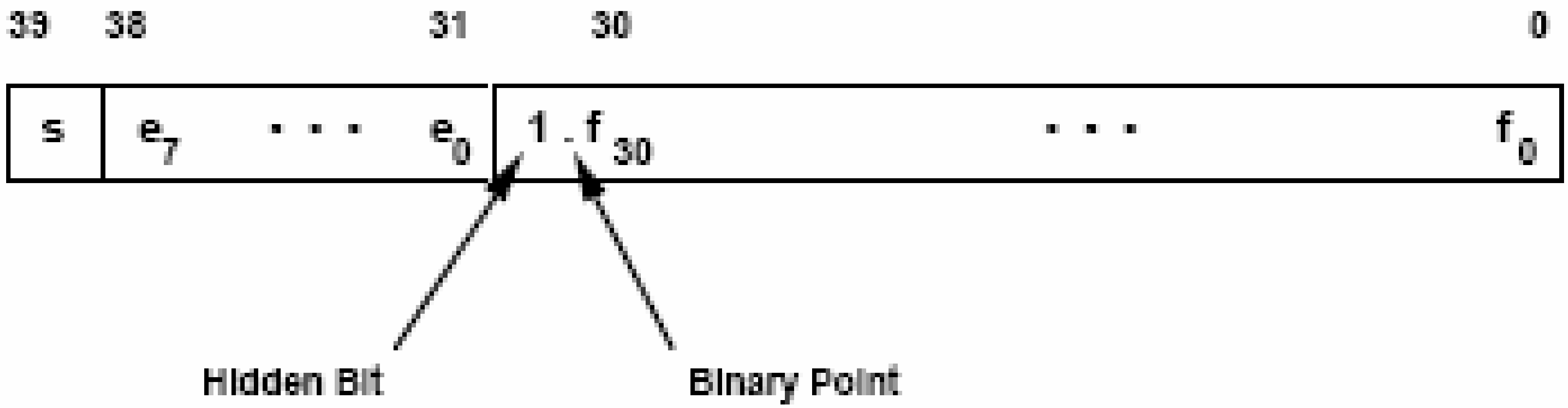
Zahlenformate Gleitkomma

Type	*Exponent*	*Fraction*	*Value*
NAN	255	Nonzero	Undefined
Infinity	255	0	$(-1)^s$ Infinity
Normal	$1 \leq e \leq 254$	Any	$(-1)^s \, (1.f_{22\text{-}0}) \, 2^{e-127}$
Zero	0	0	$(-1)^s$ Zero

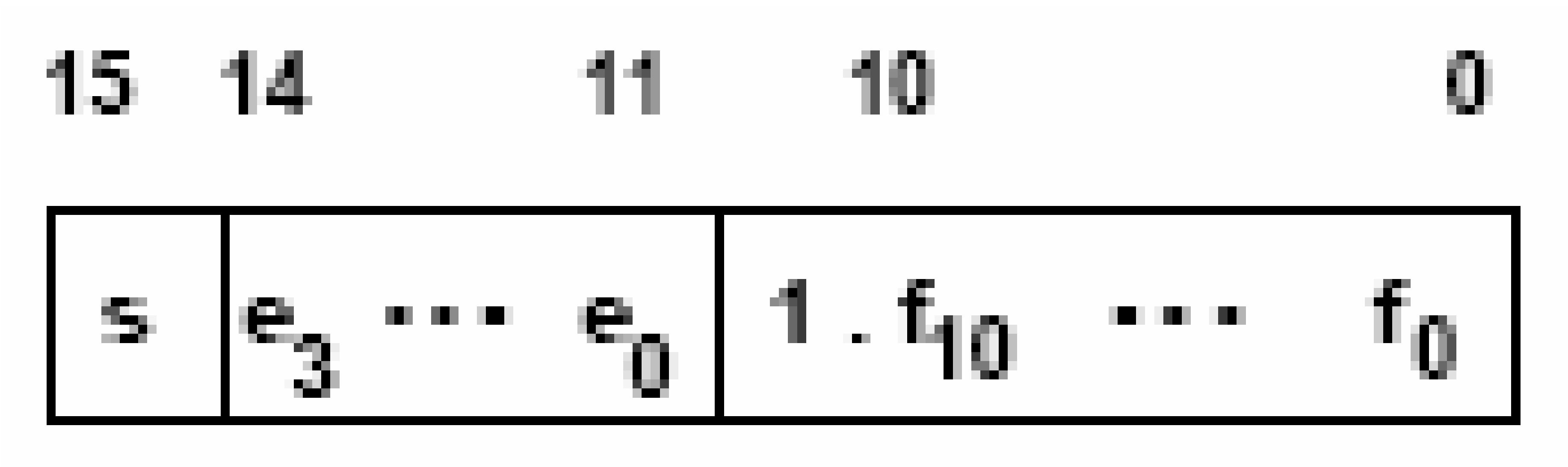
Zahlenformate Gleitkomma



Zahlenformate Gleitkomma



Zahlenformate Gleitkomma



ALU - Einstellung

MODE1

Bit	*Name*	*Function*
13	ALUSAT	1=Enable ALU saturation (full scale in fixed-point) 0=Disable ALU saturation
15	TRUNC	1=Truncation; 0=Round to nearest
16	RND32	1=Round to 32 bits; 0=Round to 40 bits

ALUSAT Bei Überlauf werden positive Festkommazahlen auf den größten Wert gesetzt
Bei Überlauf werden negative Festkommazahlen auf den kleinsten Wert gesetzt
Andernfalls bleiben die höheren 32 Bit erhalten

ALU – Flags im ASTAT Register

Bit	*Name*	*Definition*
0	AZ	ALU result zero or floating-point underflow
1	AV	ALU overflow
2	AN	ALU result negative
3	AC	ALU fixed-point carry
4	AS	ALU X input sign (ABS, MANT operations)
5	AI	ALU floating-point invalid operation
10	AF	Last ALU operation was a floating-point operation
31-24	CACC	Compare Accumulation register (results of last 8 compare operations)

ALU – Flags im STICKY Register

- ALU updates four “sticky” flags in the STKY register. Once set, a sticky flag remains high until explicitly cleared.
- *STKY*

Bit	*Name*	*Definition*
0	AUS	ALU floating-point underflow
1	AVS	ALU floating-point overflow
2	AOS	ALU fixed-point overflow
5	AIS	ALU floating-point invalid operation

ALU Festkommabefehle

Vereinbarungen

Rn, Rx, Ry = Any register file location; treated as fixed-point
Fn, Fx, Fy = Any register file location; treated as floating-point
c = ADSP-21xx-compatible instruction

* set or cleared, depending on results of instruction
** may be set (but not cleared), depending on results of instruction
– no effect

ALU Festkommabefehle

Instruction		*ASTAT Status Flags*								*STKY Status Flags*			
Fixed-point:		AZ	AV	AN	AC	AS	AI	AF	CACC	AUS	AVS	AOS	AIS
c	Rn = Rx + Ry	*	*	*	*	0	0	0	–	–	–	**	–
c	Rn = Rx – Ry	*	*	*	*	0	0	0	–	–	–	**	–
c	Rn = Rx + Ry + CI	*	*	*	*	0	0	0	–	–	–	**	–
c	Rn = Rx – Ry + CI – 1	*	*	*	*	0	0	0	–	–	–	**	–
	Rn = (Rx + Ry) / 2	*	0	*	*	0	0	0	–	–	–	–	–
	COMP(Rx, Ry)	*	0	*	0	0	0	0	*	–	–	–	–
	Rn = Rx + CI	*	*	*	*	0	0	0	–	–	–	**	–
	Rn = Rx + CI – 1	*	*	*	*	0	0	0	–	–	–	**	–
	Rn = Rx + 1	*	*	*	*	0	0	0	–	–	–	**	–
	Rn = Rx – 1	*	*	*	*	0	0	0	–	–	–	**	–
c	Rn = –Rx	*	*	*	*	0	0	0	–	–	–	**	–
c	Rn = ABS Rx	*	*	0	0	*	0	0	–	–	–	**	–
	Rn = PASS Rx	*	0	*	0	0	0	0	–	–	–	–	–
c	Rn = Rx AND Ry	*	0	*	0	0	0	0	–	–	–	–	–
c	Rn = Rx OR Ry	*	0	*	0	0	0	0	–	–	–	–	–
c	Rn = Rx XOR Ry	*	0	*	0	0	0	0	–	–	–	–	–
c	Rn = NOT Rx	*	0	*	0	0	0	0	–	–	–	–	–
	Rn = MIN(Rx, Ry)	*	0	*	0	0	0	0	–	–	–	–	–
	Rn = MAX(Rx, Ry)	*	0	*	0	0	0	0	–	–	–	–	–
	Rn = CLIP Rx BY Ry	*	0	*	0	0	0	0	–	–	–	–	–

ALU Gleitkommabefehle

Floating–point:												
Fn = Fx + Fy	*	*	*	0	0	*	1	–	**	**	–	**
Fn = Fx – Fy	*	*	*	0	0	*	1	–	**	**	–	**
Fn = ABS (Fx + Fy)	*	*	0	0	0	*	1	–	**	**	–	**
Fn = ABS (Fx – Fy)	*	*	0	0	0	*	1	–	**	**	–	**
Fn = (Fx + Fy)/2	*	0	*	0	0	*	1	–	**	–	–	**
COMP(Fx, Fy)	*	0	*	0	0	*	1	*	–	–	–	**
Fn = –Fx	*	*	*	0	0	*	1	–	–	**	–	**
Fn = ABS Fx	*	*	0	0	*	*	1	–	–	**	–	**
Fn = PASS Fx	*	0	*	0	0	*	1	–	–	–	–	**
Fn = RND Fx	*	*	*	0	0	*	1	–	–	**	–	**
Fn = SCALB Fx BY Ry	*	*	*	0	0	*	1	–	**	**	–	**
Rn = MANT Fx	*	*	0	0	*	*	1	–	–	**	–	**
Rn = LOGB Fx	*	*	*	0	0	*	1	–	–	**	–	**
Rn = FIX Fx BY Ry	*	*	*	0	0	*	1	–	**	**	–	**
Rn = FIX Fx	*	*	*	0	0	*	1	–	**	**	–	**
Fn = FLOAT Rx BY Ry	*	*	*	0	0	0	1	–	**	**	–	–
Fn = FLOAT Rx	*	0	*	0	0	0	1	–	–	–	–	–
Fn = RECIPS Fx	*	*	*	0	0	*	1	–	**	**	–	**
Fn = RSQRTS Fx	*	*	*	0	0	*	1	–	–	**	–	**
Fn = Fx COPYSIGN Fy	*	0	*	0	0	*	1	–	–	–	–	**
Fn = MIN(Fx, Fy)	*	0	*	0	0	*	1	–	–	–	–	**
Fn = MAX(Fx, Fy)	*	0	*	0	0	*	1	–	–	–	–	**
Fn = CLIP Fx BY Fy	*	0	*	0	0	*	1	–	–	–	–	**

Multiplizierer

Multiplier instructions include:

- Floating-point multiplication
- Fixed-point multiplication
- Fixed-point multiply/accumulate with addition, rounding optional
- Fixed-point multiply/accumulate with subtraction, rounding optional
- Rounding result register
- Saturating result register
- Clearing result register

Multiplizierer - Flags

ASTAT

Bit	*Name*	*Definition*
6	MN	Multiplier result negative
7	MV	Multiplier overflow
8	MU	Multiplier underflow
9	MI	Multiplier floating-point invalid operation

Multiplizierer - Flags

STKY

Bit	*Name*	*Definition*
6	MOS	Multiplier fixed-point overflow
7	MVS	Multiplier floating-point overflow
8	MUS	Multiplier underflow
9	MIS	Multiplier floating-point invalid operation

Multiplizierer - Vereinbarungen

* set or cleared,

** may be set (but not cleared– no effect)

Rn, Rx, Ry = R15-R0; register file location, treated as fixed-point

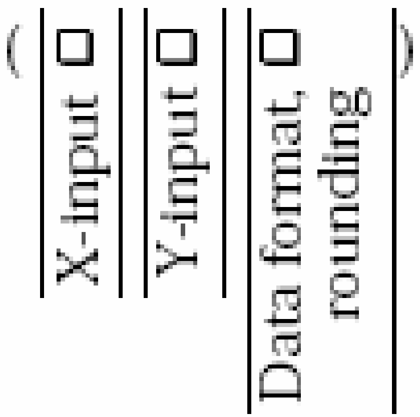
Fn, Fx, Fy = F15-F0; register file location, treated as floating-point

MRxF = MR2F, MR1F, MR0F; multiplier result accumulators, foreground

MRxB = MR2B, MR1B, MR0B; multiplier result accumulators, background

Multiplizierer
Zahlenformatangabe

Optional Modifiers for Fixed-Point:



S	Signed input
U	Unsigned input
I	Integer input(s)
F	Fractional input(s)
FR	Fractional inputs, Rounded output
(SF)	Default format for 1-input operations
(SSF)	Default format for 2-input operations

Multiplizierer Befehle

Instruction	MU	MN	MV	MI	MUS	MOS	MVS	MIS
	ASTAT Flags				*STKY Flags*			
Fixed-Point:								
Rn / MRF / MRB = Rx * Ry (*S* / *U* , *S* / *U* , *F* / *I* / *FR*)	*	*	*	0	–	**	–	–
Rn = MRF / Rn = MRB / MRF = MRF / MRB = MRB + Rx * Ry (*S* / *U* , *S* / *U* , *F* / *I* / *FR*)	*	*	*	0	–	**	–	–
Rn = MRF / Rn = MRB / MRF = MRF / MRB = MRB – Rx * Ry (*S* / *U* , *S* / *U* , *F* / *I* / *FR*)	*	*	*	0	–	**	–	–

Multiplizierer Befehle

Instruction		*ASTAT Flags*				*STKY Flags*			
		MU	MN	MV	MI	MUS	MOS	MVS	MIS
Rn = SAT MRF Rn = SAT MRB MRF = SAT MRF MRB = SAT MRB	*(SI)* *(UI)* *(SF)* *(UF)*	*	*	*	0	–	**	–	–
Rn = RND MRF Rn = RND MRB MRF = RND MRF MRB = RND MRB	*(SF)* *(UF)*	*	*	*	0	–	**	–	–
MRF / MRB = 0		0	0	0	0	–	–	–	–
MRxF / MRxB = Rn		0	0	0	0	–	–	–	–
Rn = MRxF / MRxB		0	0	0	0	–	–	–	–

Multiplizierer
Festkommaergebnis