9. Assembler: Der Prozessor Motorola 68000

9.1 Architektur des Prozessors M 68000

9.2 Adressierungsarten des M 68000

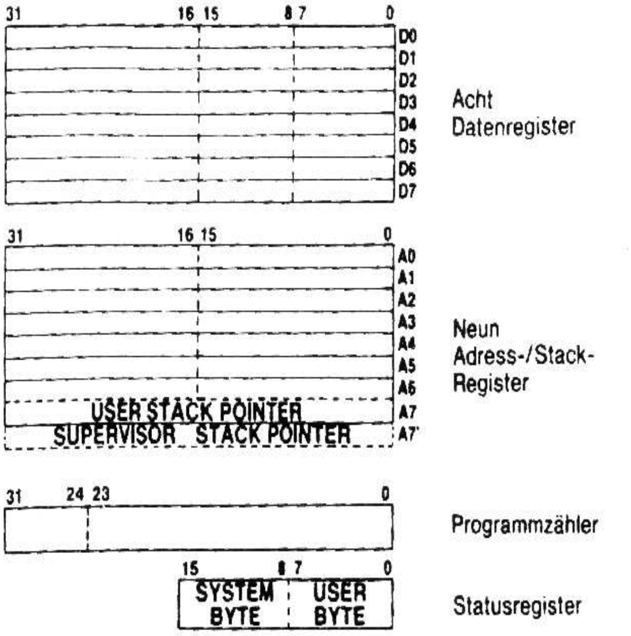
9.1 Beschreibung des Prozessors M 68000

Charakteristische Daten des Motorola 68000

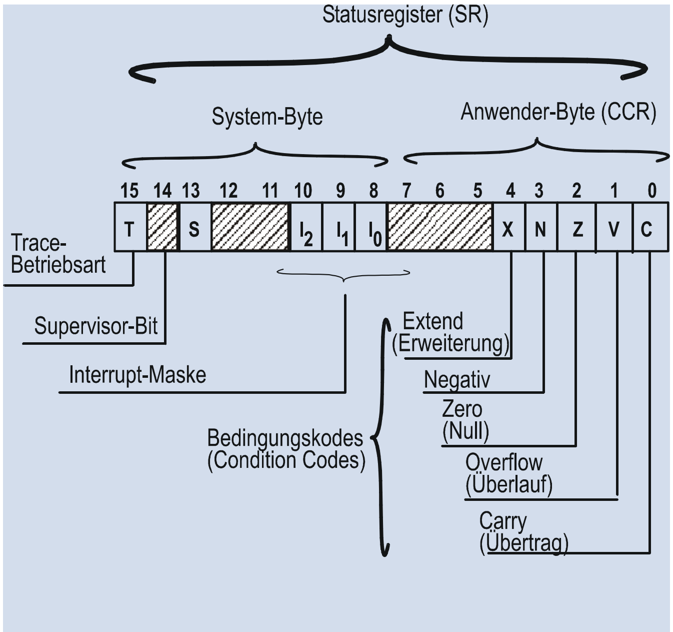
- 56 Maschinenbefehle
- 14 Adressierungsarten
- Zweiadressmaschine mit
  - acht Datenregistern
  - neun Adress-/ Stackregistern
  - einem Befehlszähler (PC)
  - einem Statusregister
- CPU-intern 32- Bit-Rechnerorganisation
- Busbreite 16 Bit
- Taktfrequenz 4-12,5 MHz (je nach Modell)

“Urvater“ einer sehr erfolgreichen Prozessor-Familie (68010, 68020, 68030, 68040)

Registermodell des M 68000


31
16 15
8 7
0
D0
D1
D2
D3
D4
D5
D6
D7
Acht
Datenregister
31
16 15
0
A0
A1
A2
A3
A4
A5
A6
USER STACK POINTER
SUPERVISOR STACK POINTER
A7
A7'
Neun
Adress-/Stack-
Register
31
24 23
0
Programmzähler
15
8 7
0
SYSTEM BYTE
USER BYTE
Statusregister

Statusregister


Statusregister (SR)
System-Byte
Anwender-Byte (CCR)
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
T
S
I2
I1
I0
X
N
Z
V
C
Trace-
Betriebsart
Supervisor-Bit
Interrupt-Maske
Extend
(Erweiterung)
Negativ
Zero
(Null)
Overflow
(Überlauf)
Carry
(Übertrag)
Bedingungskodes
(Condition Codes)

System - und User - (Anwender -) byte im Statusregister

Die Flags im Status-Register SR (1)

Das **Carry-Flag C** wird immer dann auf Eins gesetzt, wenn bei einer arithmetischen Oparation im M68000 (z. B. Addieren, Schieben) aus dem vordersten Bit des Zieloperanden (Bit 31 bei Langwort, Bit 15 bei Wort und Bit 7 bei Byte) eine Eins als Übertrag entsteht oder bei Subtraktion ein "Borgen" durchgeführt wird. Ein Beispiel für die Anwendung des Carry-Flags wäre die Addition von zwei Zahlen, die jeweils größer als 32 Bit sind.

Das **Overflow-Flag V** zeigt an, ob während der Durchführung einer arithmetischen Operation ein Überschreiten eines Zahlenbereichs aufgetreten ist, d. h. wenn z. B. bei Addition von zwei positiven Zahlen das Ergebnis eine negative Zahl im Zweierkomplement darstellt. Außerdem wird das Overflow-Flag benutzt, um bei einer Divisionsoperation anzuzeigen, dass der Dividend zu groß ist bzw. der Quotient größer als 16 Bit werden würde. Ist dieser Fall eingetreten, so wird das Overflow-Flag auf Eins gesetzt, ansonsten auf Null.

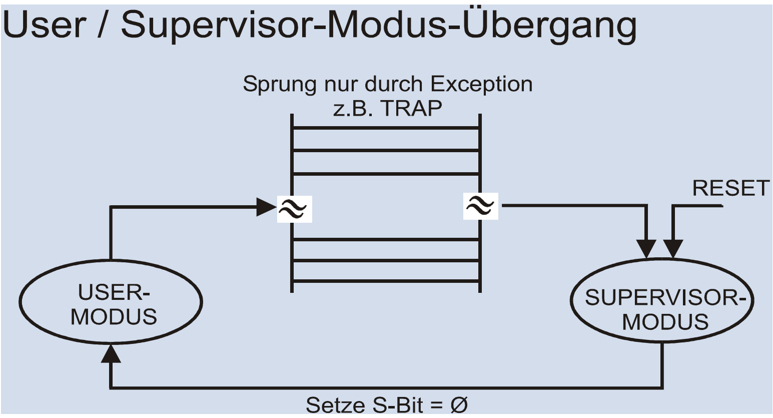
Das **Zero-Flag Z** wird auf Eins gesetzt, wenn bei einer Operation das Ergebnis gleich Null ist bzw. wenn bei einer Vergleichsoperation (CMP) beide Operanden gleich sind, ansonsten enthält es eine Null.

Die Flags im Status-Register SR (2)

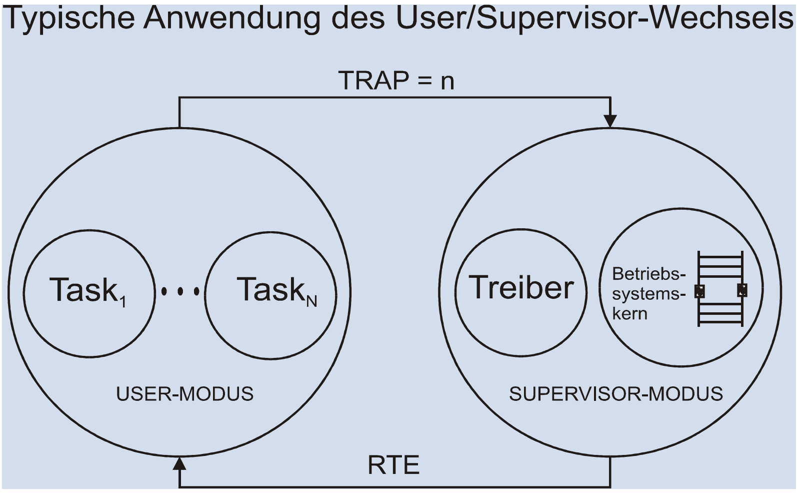
Das **Negativ-Flag N** steht nach der Ausführung einer Oparation auf Eins, wenn das Ergebnis eine negative Zahl im Zweierkomplement darstellt, d.h. das höchst-wertigste Bit im Ergebnis eine Eins ist. Ist dies nicht der Fall, so enthält dieses Flag eine Null.

Das **Extend-Flag X** ist eine Besonderheit des M68000. Generell lässt sich über dieses Flag sagen, dass es sich, wenn es gesetzt wird, genauso verhält wie das Carry-Flag. Bei einigen Befehlen jedoch wird das Carry-Flag verändert, das Extend-Flag nicht. Man kann das Extend-Flag dazu verwenden, den Inhalt des Carry-Flags über mehrere Operationen weg aufzubewahren. So wird das Extend-Flag z. B. bei Addition und Subtraktion genauso gesetzt wie das Carry-Flag, bei Vergleichsoperationen (CMP) jedoch nicht verändert. Für exakte Aussagen, wann das Extend-Flag verwendet wird und wann nicht, ist es am Besten, die einzelnen Befehlsbeschreibungen nachzulesen. Ein Beispiel für eine Anwendung des Extend-Flags im M68000 ist die Addition von zwei Zahlen, die jeweils größer als 32 Bit sind.

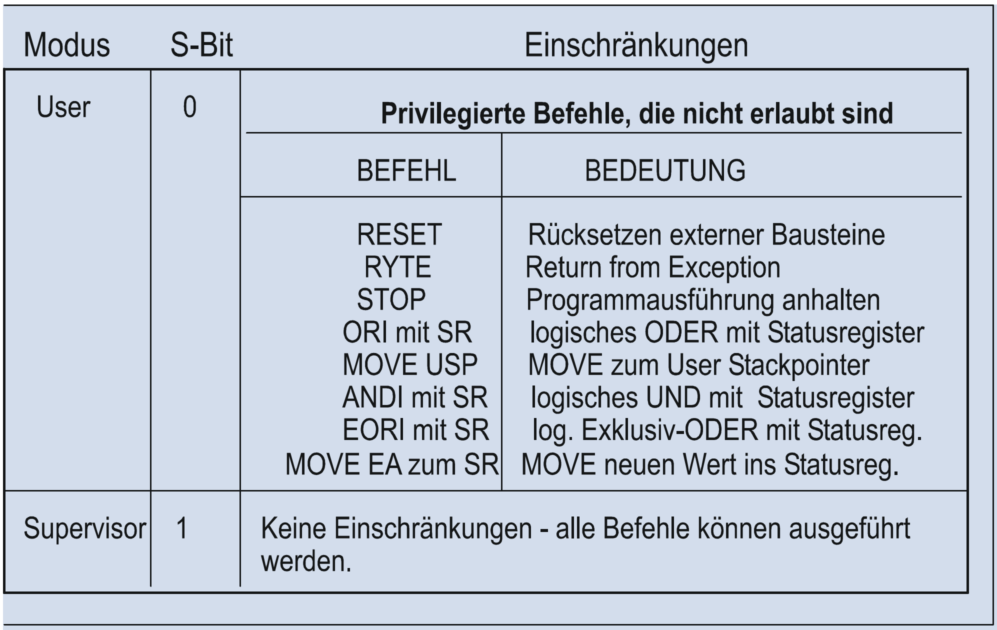
User Mode und Supervisor Mode (1)


User / Supervisor-Modus-Übergang
Sprung nur durch Exception
z.B. TRAP
RESET
USER-MODUS
SUPERVISOR-MODUS
Setze S-Bit = 0

User Mode und Supervisor Mode (2)


Typische Anwendung des User/Supervisor-Wechsels
TRAP = n
Task$_1$
Task$_N$
USER-MODUS
Treiber
Betriebs-systems-kern
SUPERVISOR-MODUS
RTE

Supervisor Mode und privilegierte Befehle



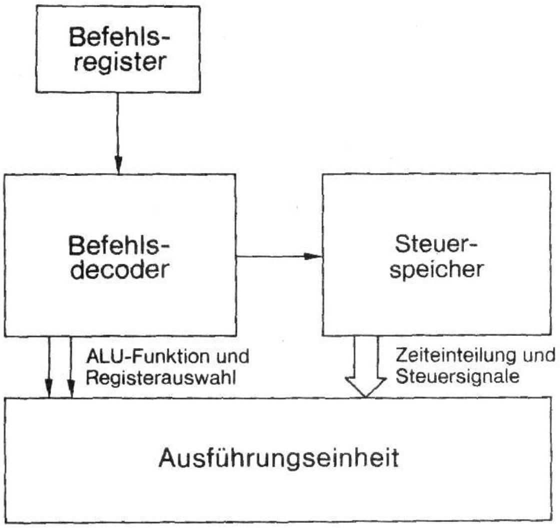
Modus	S-Bit	Einschränkungen	
User	0	**Privilegierte Befehle, die nicht erlaubt sind**	
		BEFEHL	BEDEUTUNG
		RESET	Rücksetzen externer Bausteine
		RYTE	Return from Exception
		STOP	Programmausführung anhalten
		ORI mit SR	logisches ODER mit Statusregister
		MOVE USP	MOVE zum User Stackpointer
		ANDI mit SR	logisches UND mit Statusregister
		EORI mit SR	log. Exklusiv-ODER mit Statusreg.
		MOVE EA zum SR	MOVE neuen Wert ins Statusreg.
Supervisor	1	Keine Einschränkungen - alle Befehle können ausgeführt werden.	

Privilegierte Befehle

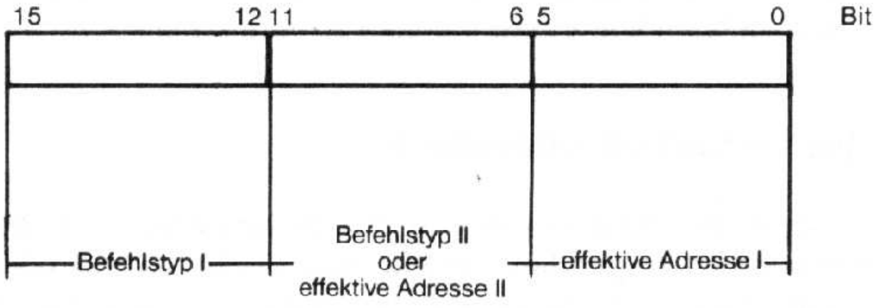
Mikroprogramm-Steuerung

Ca. 180 Steuersignale in der Ausführungseinheit (= Rechenwerk = ALU).

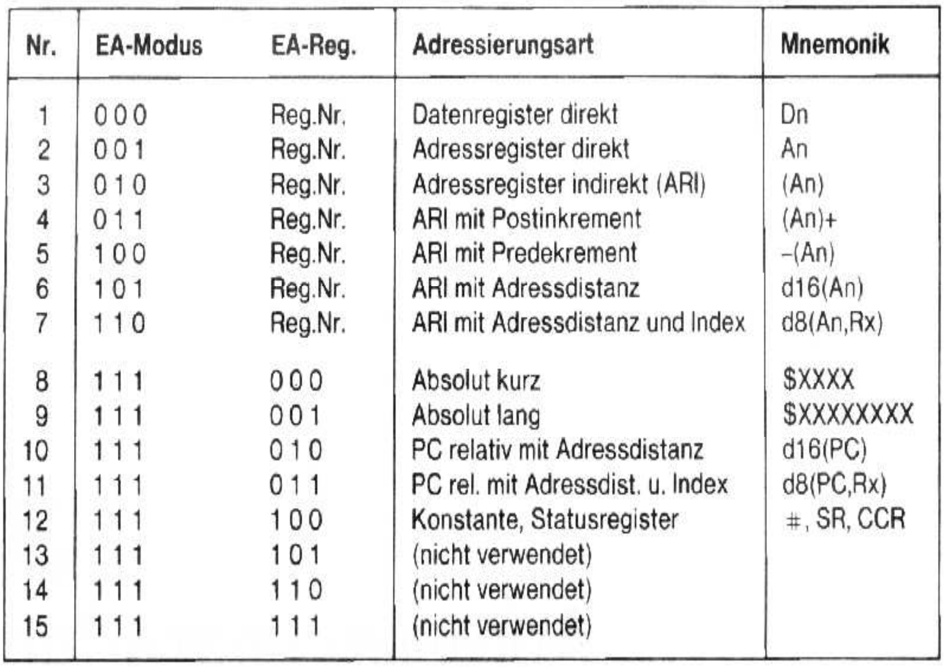
Steuerspeicher ist als mikroprogrammiertes ROM ausgeführt.


Befehls-register
Befehls-decoder
Steuer-speicher
ALU-Funktion und Registerauswahl
Zeiteinteilung und Steuersignale
Ausführungseinheit

9.2 Adressierungsarten des M 68000



Adressierungsarten des M 68000 (2)



Nr.	EA-Modus	EA-Reg.	Adressierungsart	Mnemonik
1	0 0 0	Reg.Nr.	Datenregister direkt	Dn
2	0 0 1	Reg.Nr.	Adressregister direkt	An
3	0 1 0	Reg.Nr.	Adressregister indirekt (ARI)	(An)
4	0 1 1	Reg.Nr.	ARI mit Postinkrement	(An)+
5	1 0 0	Reg.Nr.	ARI mit Predekrement	–(An)
6	1 0 1	Reg.Nr.	ARI mit Adressdistanz	d16(An)
7	1 1 0	Reg.Nr.	ARI mit Adressdistanz und Index	d8(An,Rx)
8	1 1 1	0 0 0	Absolut kurz	$XXXX
9	1 1 1	0 0 1	Absolut lang	$XXXXXXXX
10	1 1 1	0 1 0	PC relativ mit Adressdistanz	d16(PC)
11	1 1 1	0 1 1	PC rel. mit Adressdist. u. Index	d8(PC,Rx)
12	1 1 1	1 0 0	Konstante, Statusregister	#, SR, CCR
13	1 1 1	1 0 1	(nicht verwendet)	
14	1 1 1	1 1 0	(nicht verwendet)	
15	1 1 1	1 1 1	(nicht verwendet)	

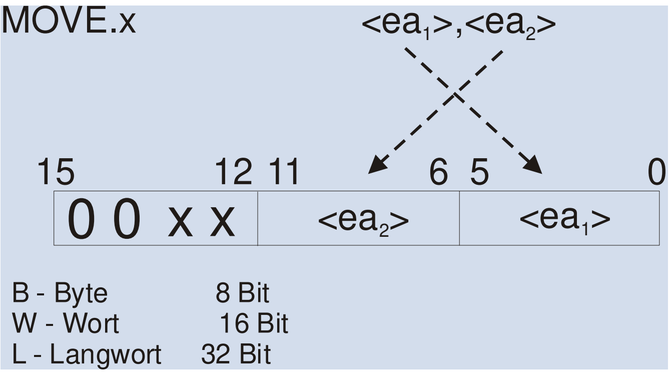
Datenformate (1)

Der M68000 verarbeitet folgende Datenlängen:

Name	**Größe**	**Verarbeitung**
Bit	1 Bit	Bitmanipulationsbefehle
BCD	4 Bit	BCD-Befehle (gepackt, d.h. zwei BCD-Zahlen in einem Byte)
Byte	8 Bit	versch. Befehle, externe Daten
Wort*	16 Bit	versch. Befehle, externe Daten
Langwort	32 Bit	versch. Befehle, externe Daten

* Ein Wort ist die Standardverarbeitung, da der M68000 einen 16bit-Datenbus hat. Die Datenorganisation im Speicher zeigt Bild 2-10a.

Datenformate (2)



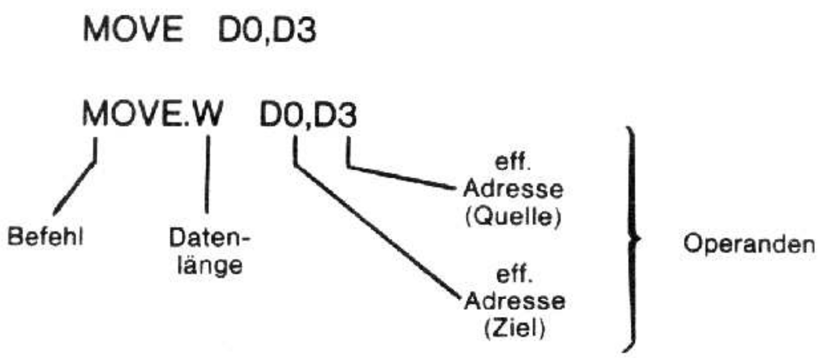
B - Byte	8 Bit
W - Wort	16 Bit
L - Langwort	32 Bit

Adressierungsart "Register direkt“ (1)

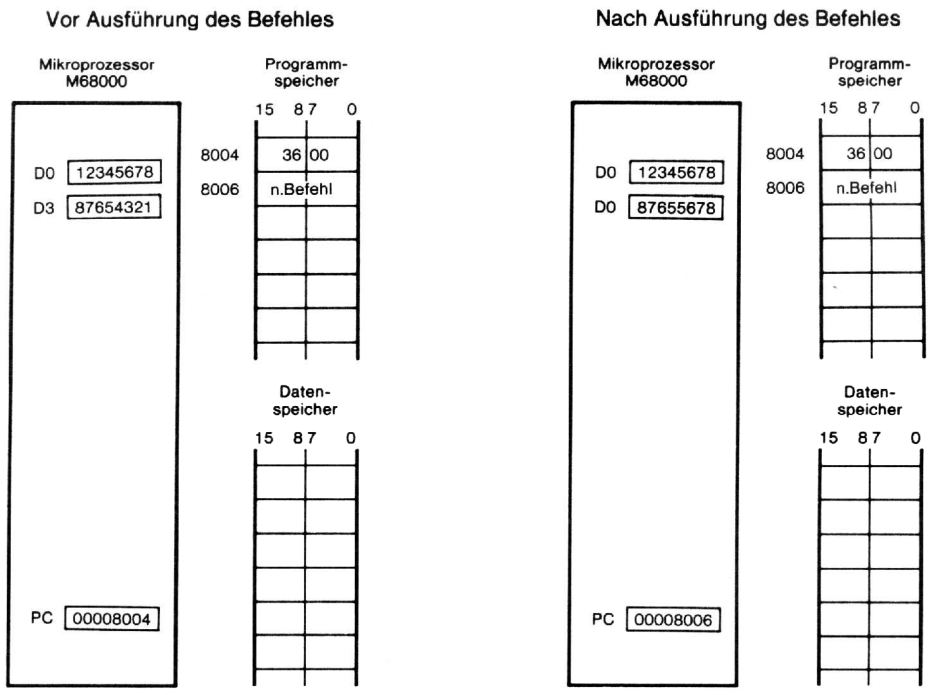
Die Operandenadresse ist ein Register (Datenregister, Adressregister, Statusregister).

Beispiel: MOVE D0,D3

Gleichbedeutend mit:


MOVE D0,D3
MOVE.W D0,D3
Befehl
Daten-länge
eff. Adresse (Quelle)
eff. Adresse (Ziel)
Operanden

Adressierungsart "Register direkt“ (2)


Vor Ausführung des Befehles
Mikroprozessor M68000
D0 12345678
D3 87654321
PC 00008004
Programm-speicher
15 8 7 0
8004 36 00
8006 n.Befehl
Daten-speicher
15 8 7 0
Nach Ausführung des Befehles
Mikroprozessor M68000
D0 12345678
D0 87655678
PC 00008006
Programm-speicher
15 8 7 0
8004 36 00
8006 n.Befehl
Daten-speicher
15 8 7 0

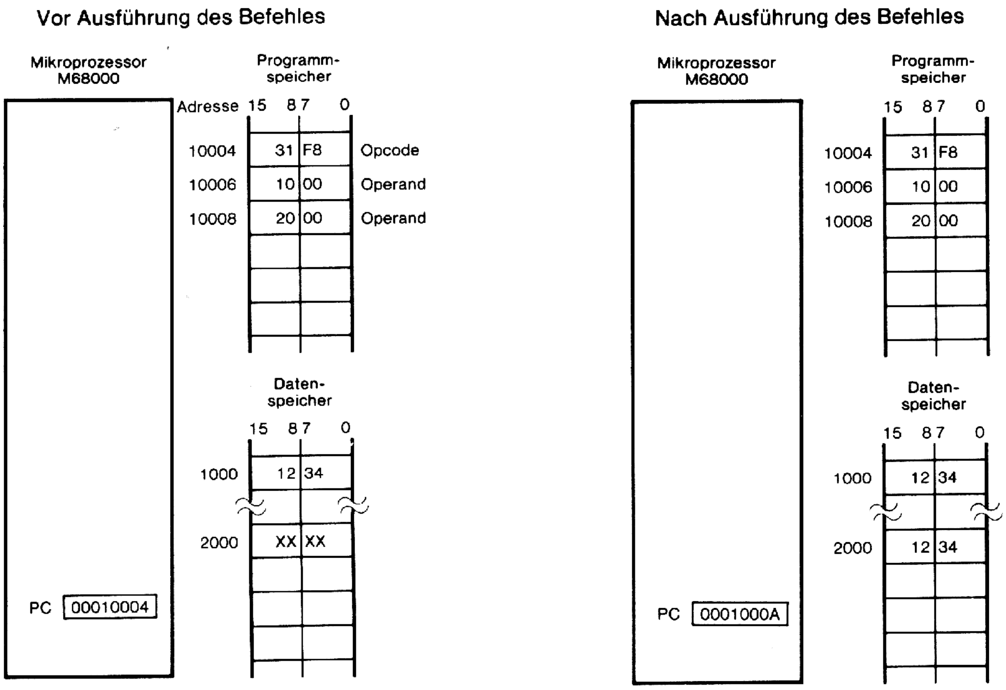
Absolute Adressierung (1)

Die Operandenadresse ist eine absolute Adresse im Speicher.

Beispiel 1: MOVE $1000,$2000

Gleichbedeutend mit: MOVE.W $1000,$2000

Absolute Adressierung (2)


Vor Ausführung des Befehles
Mikroprozessor M68000
PC 00010004
Programm-speicher
Adresse
15 8 7 0
10004 31 F8 Opcode
10006 10 00 Operand
10008 20 00 Operand
Daten-speicher
15 8 7 0
1000 12 34
2000 XX XX
Nach Ausführung des Befehles
Mikroprozessor M68000
PC 0001000A
Programm-speicher
15 8 7 0
10004 31 F8
10006 10 00
10008 20 00
Daten-speicher
15 8 7 0
1000 12 34
2000 12 34

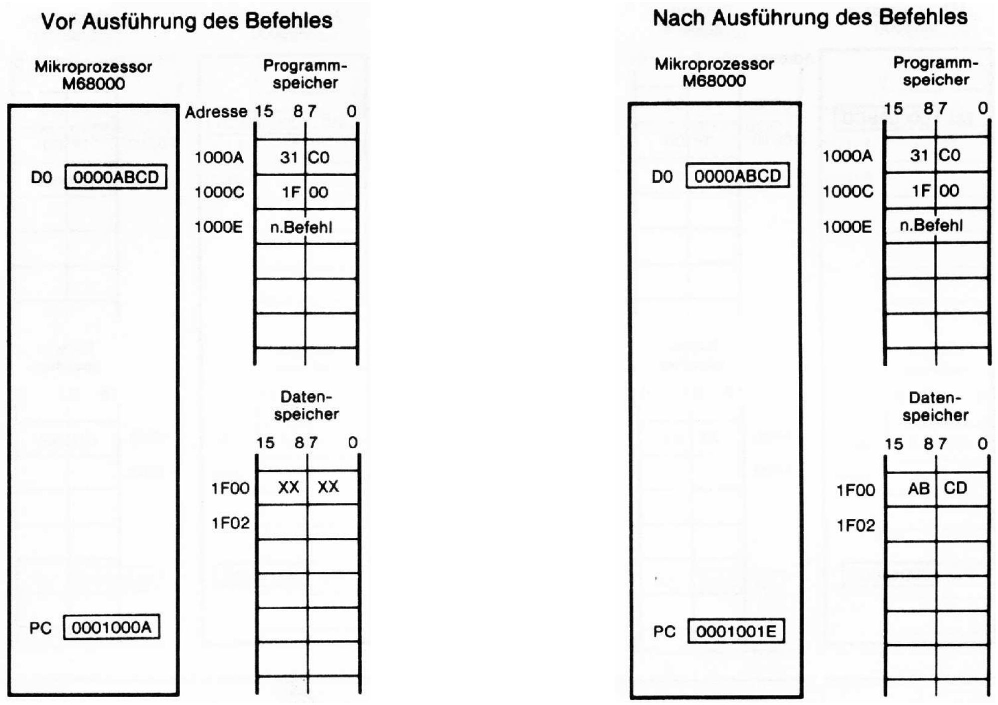
XXXX = undefinierter Wert

Absolute Adressierung (3)

Beispiel 2: MOVE D0,$1F00

Gleichbedeutend mit MOVE.W D0,$1F00

Absolute Adressierung (4)

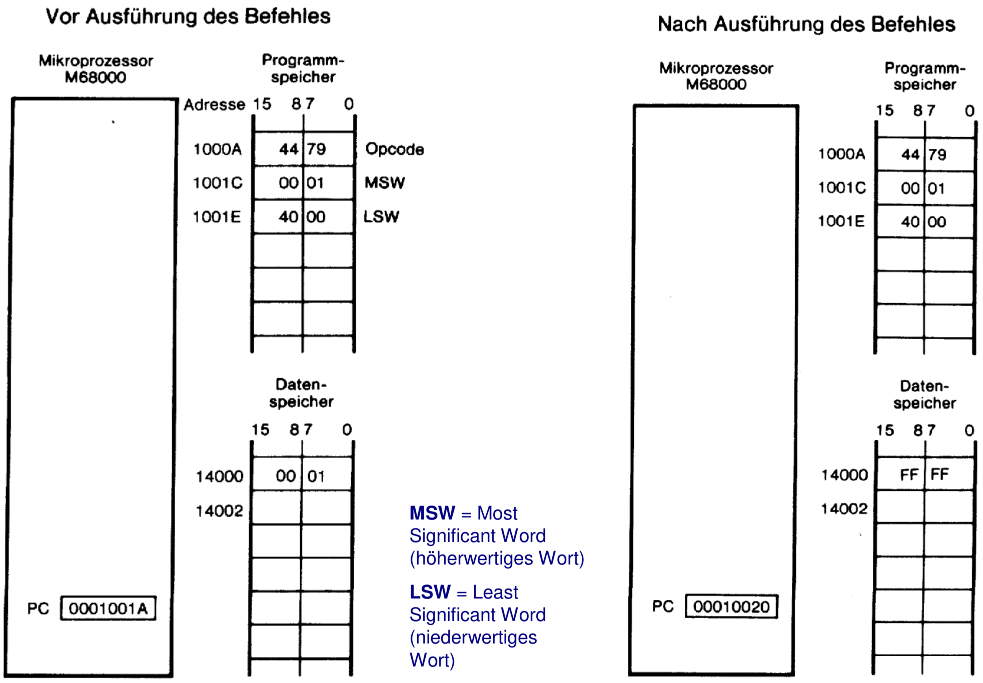

Vor Ausführung des Befehles
Mikroprozessor M68000
Programm-speicher
Adresse 15 8 7 0
D0 0000ABCD
1000A 31 C0
1000C 1F 00
1000E n.Befehl
Daten-speicher
15 8 7 0
1F00 XX XX
1F02
PC 0001000A
Nach Ausführung des Befehles
Mikroprozessor M68000
Programm-speicher
15 8 7 0
D0 0000ABCD
1000A 31 C0
1000C 1F 00
1000E n.Befehl
Daten-speicher
15 8 7 0
1F00 AB CD
1F02
PC 0001001E

Absolute Adressierung (5)

Beispiel 3: NEG $14000

Gleichbedeutend mit: NEG.W $14000

Absolute Adressierung (6)


Vor Ausführung des Befehles
Mikroprozessor M68000
Programm-speicher
Adresse 15 8 7 0
1000A 44 79 Opcode
1001C 00 01 MSW
1001E 40 00 LSW
Daten-speicher
15 8 7 0
14000 00 01
14002
PC 0001001A
Nach Ausführung des Befehles
Mikroprozessor M68000
Programm-speicher
15 8 7 0
1000A 44 79
1001C 00 01
1001E 40 00
Daten-speicher
15 8 7 0
14000 FF FF
14002
PC 00010020

MSW = Most Significant Word (höherwertiges Wort)

LSW = Least Significant Word (niederwertiges Wort)

Konstanten-Adressierung (1)

Der **Operand steht im Befehl** oder folgt unmittelbar dem Befehl im nächsten Wort oder den nächsten beiden Worten des Speichers.

Die Verarbeitung ist wesentlich schneller als bei absoluter Adressierung von Konstanten, die im Speicher abgelegt sind (erfordert nur eine "Fetch"-Operation).

Häufige Verwendung für kleine Ganzzahl-Konstanten (0, 1, -1).

Konstanten-Adressierung (2)

Beispiel 1:

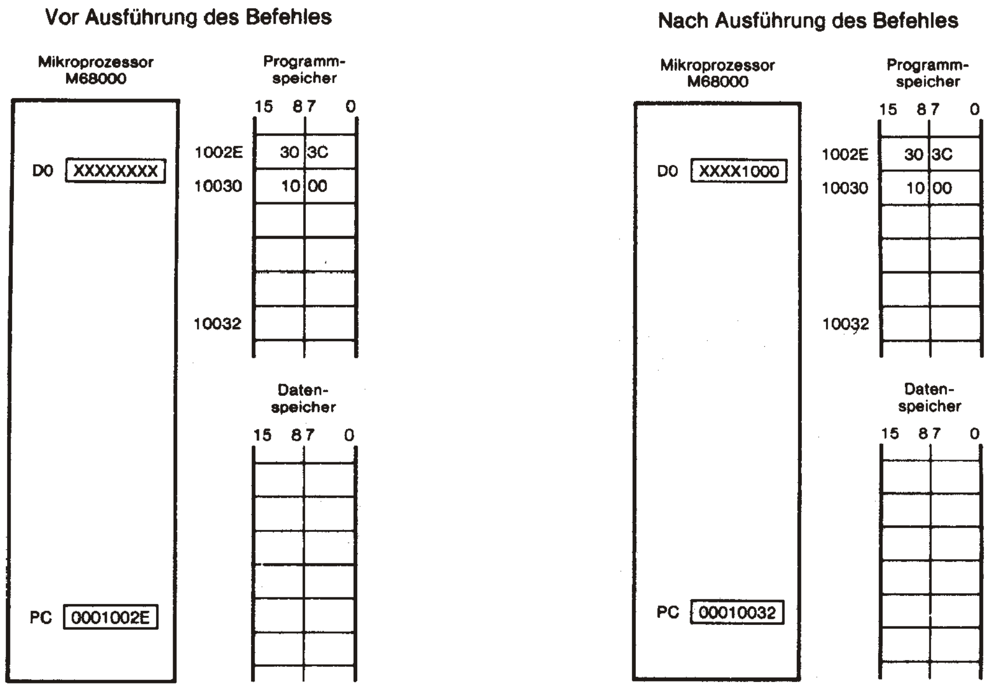
MOVEQ #$5A,D3 (MOVEQ = MOVE QUICK)

Von der Funktion her ist dieser Befehl identisch mit dem Befehl

MOVE.L #$5A,D3 ,

jedoch nicht in Bezug auf Länge und Ausführungszeit!

Konstanten-Adressierung (3)


Vor Ausführung des Befehles
Mikroprozessor M68000
Programm-speicher
15 8 7 0
D0 XXXXXXXX
1002E
30 3C
10030
10 00
10032
Daten-speicher
15 8 7 0
PC 0001002E
Nach Ausführung des Befehles
Mikroprozessor M68000
Programm-speicher
15 8 7 0
D0 XXXX1000
1002E
30 3C
10030
10 00
10032
Daten-speicher
15 8 7 0
PC 00010032

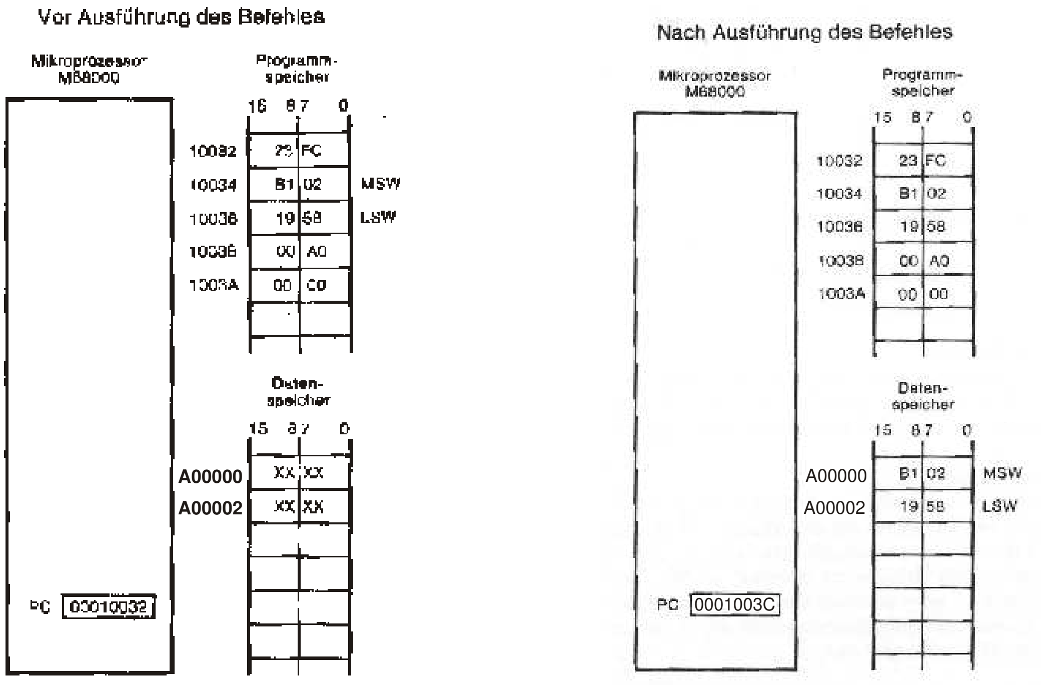
Konstanten-Adressierung (4)

Beispiel 2:

Langwort- Konstante

MOVE.L #$B1021958, $A00000

Konstanten-Adressierung (5)


Vor Ausführung des Befehles
Mikroprozessor M68000
Programm-speicher
15 8 7 0
10032 23 FC
10034 B1 02 MSW
10036 19 58 LSW
10038 00 A0
1003A 00 C0
Daten-speicher
15 8 7 0
A00000 XX XX
A00002 XX XX
PC 00010032
Nach Ausführung des Befehles
Mikroprozessor M68000
Programm-speicher
15 8 7 0
10032 23 FC
10034 B1 02
10036 19 58
10038 00 A0
1003A 00 00
Daten-speicher
15 8 7 0
A00000 B1 02 MSW
A00002 19 58 LSW
PC 0001003C

Register-indirekte Adressierung (1)

Ein **Adressregister enthält die Adresse** des Operanden.

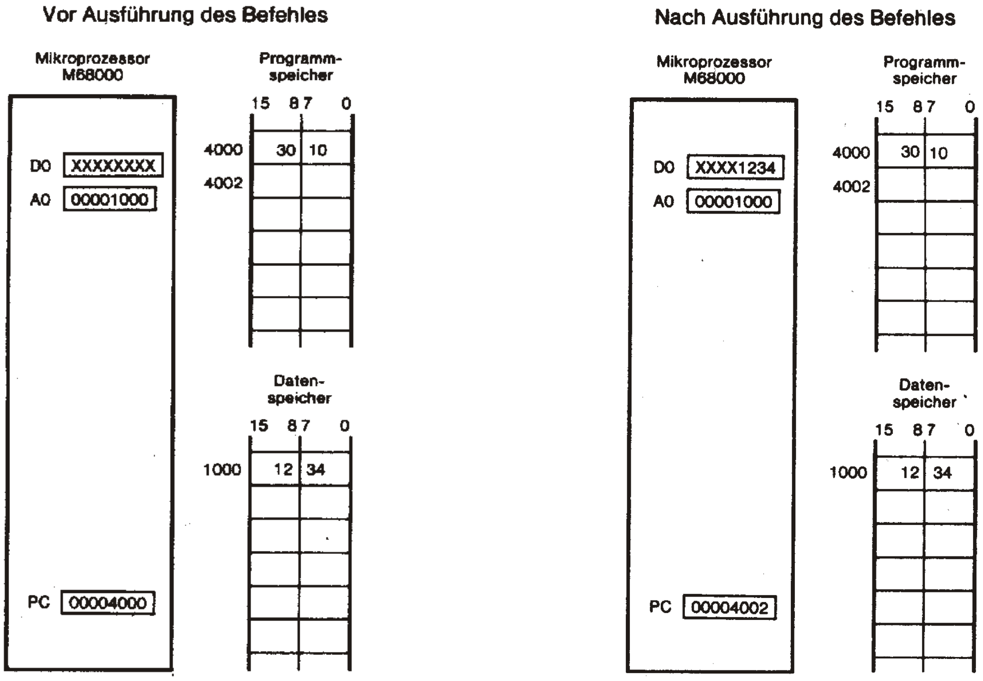
Beispiel 1:

```
MOVE   (A0),D0
```

Gleichbedeutend mit

```
MOVE.W (A0),D0
```

Register-indirekte Adressierung (2)


Vor Ausführung des Befehles
Mikroprozessor M68000
D0 XXXXXXXX
A0 00001000
PC 00004000
Programm-speicher
15 8 7 0
4000 30 10
4002
Daten-speicher
15 8 7 0
1000 12 34
Nach Ausführung des Befehles
Mikroprozessor M68000
D0 XXXX1234
A0 00001000
PC 00004002
Programm-speicher
15 8 7 0
4000 30 10
4002
Daten-speicher
15 8 7 0
1000 12 34

Register-indirekte Adressierung (3)

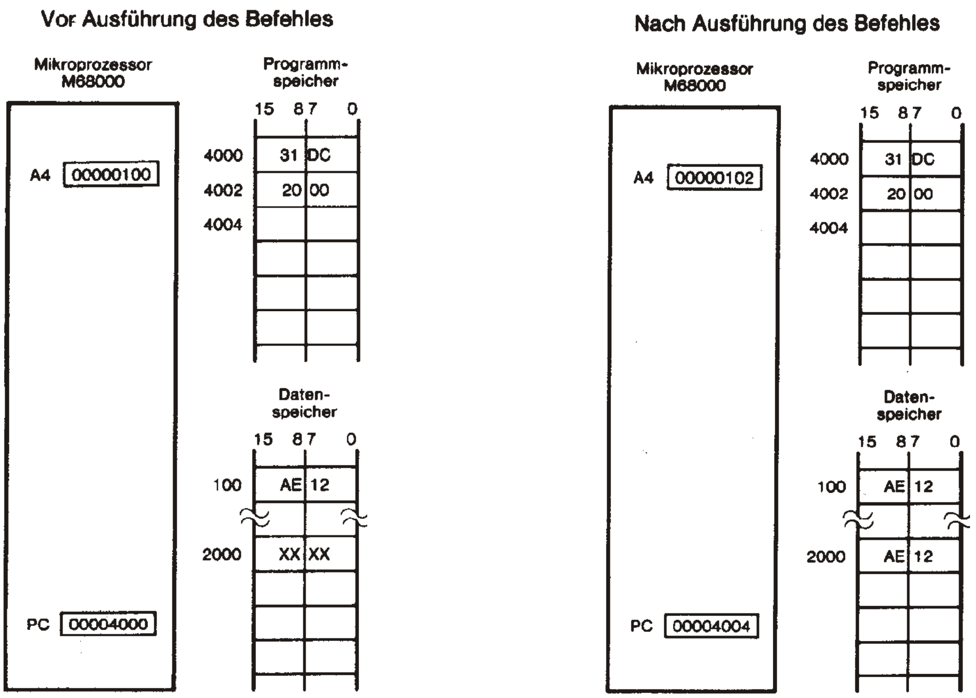
Beispiel 2: mit Postinkrement

MOVE (A4)+,$2000

Gleichbedeutend mit:

MOVE.W (A4)+,$2000

Register-indirekte Adressierung (4)


Vor Ausführung des Befehles
Mikroprozessor M68000
A4 00000100
PC 00004000
Programm-speicher
15 8 7 0
4000 31 DC
4002 20 00
4004
Daten-speicher
15 8 7 0
100 AE 12
2000 XX XX
Nach Ausführung des Befehles
Mikroprozessor M68000
A4 00000102
PC 00004004
Programm-speicher
15 8 7 0
4000 31 DC
4002 20 00
4004
Daten-speicher
15 8 7 0
100 AE 12
2000 AE 12

Register-indirekte Adressierung (5)

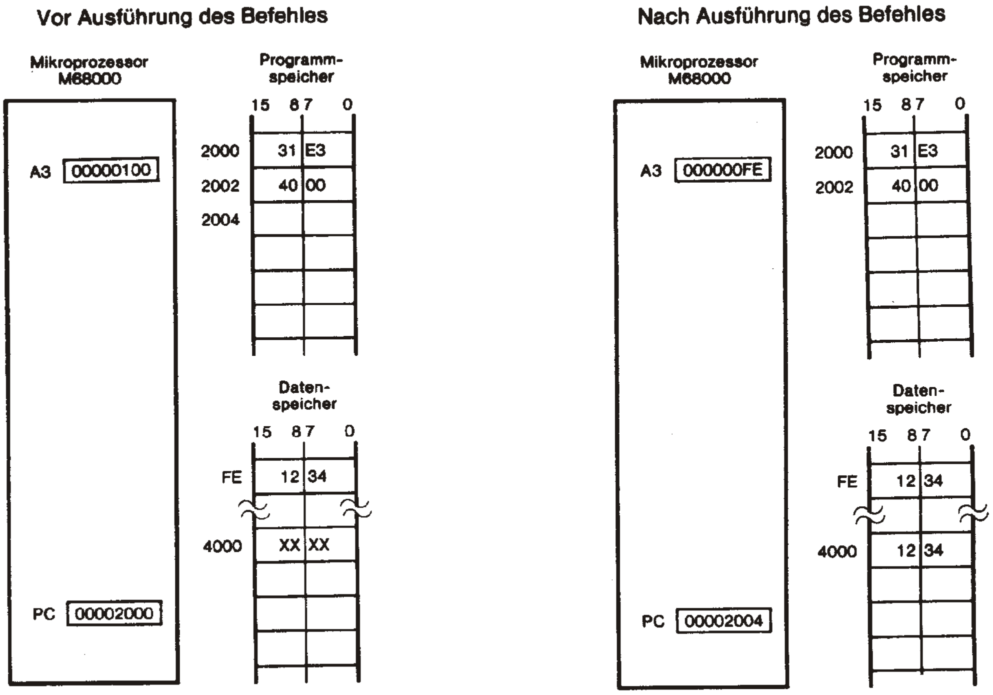
Beispiel 3: mit Prädekrement

```
MOVE      -(A3),$4000
```

Gleichbedeutend mit

```
MOVE.W    -(A3),$4000
```

Register-indirekte Adressierung (6)


Vor Ausführung des Befehles
Mikroprozessor M68000
A3 00000100
PC 00002000
Programm-speicher
15 8 7 0
2000 31 E3
2002 40 00
2004
Daten-speicher
15 8 7 0
FE 12 34
4000 XX XX
Nach Ausführung des Befehles
Mikroprozessor M68000
A3 000000FE
PC 00002004
Programm-speicher
15 8 7 0
2000 31 E3
2002 40 00
Daten-speicher
15 8 7 0
FE 12 34
4000 12 34

Register-indirekte Adressierung (7)

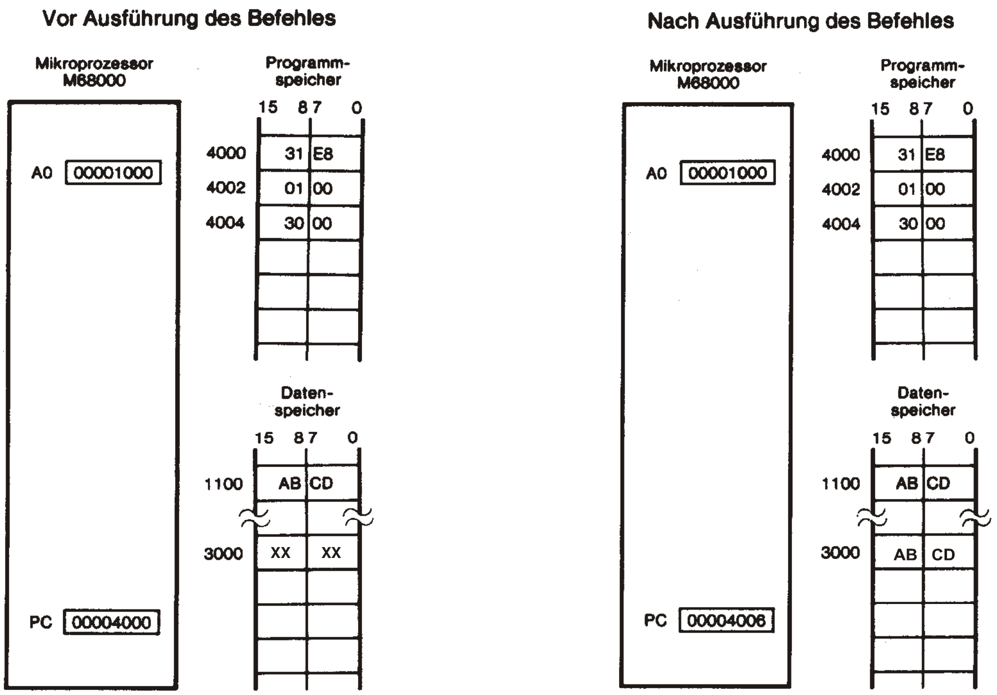
Beispiel 4: mit Adressdistanz

```
MOVE     $100(A0),$3000
```

Gleichlautend mit

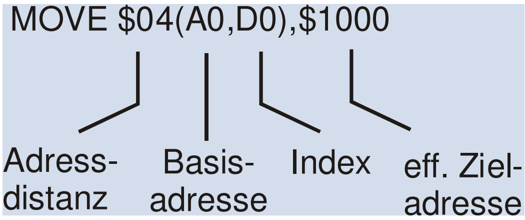
```
MOVE.W   256(A0),$3000
```

Register-indirekte Adressierung (8)


Vor Ausführung des Befehles
Mikroprozessor M68000
A0 00001000
PC 00004000
Programm-speicher
15 8 7 0
4000 31 E8
4002 01 00
4004 30 00
Daten-speicher
15 8 7 0
1100 AB CD
3000 XX XX
Nach Ausführung des Befehles
Mikroprozessor M68000
A0 00001000
PC 00004006
Programm-speicher
15 8 7 0
4000 31 E8
4002 01 00
4004 30 00
Daten-speicher
15 8 7 0
1100 AB CD
3000 AB CD

Register-indirekte Adressierung (9)

Beispiel 5: mit Index, mit Adressdistanz



MOVE $04(A0,D0),$1000

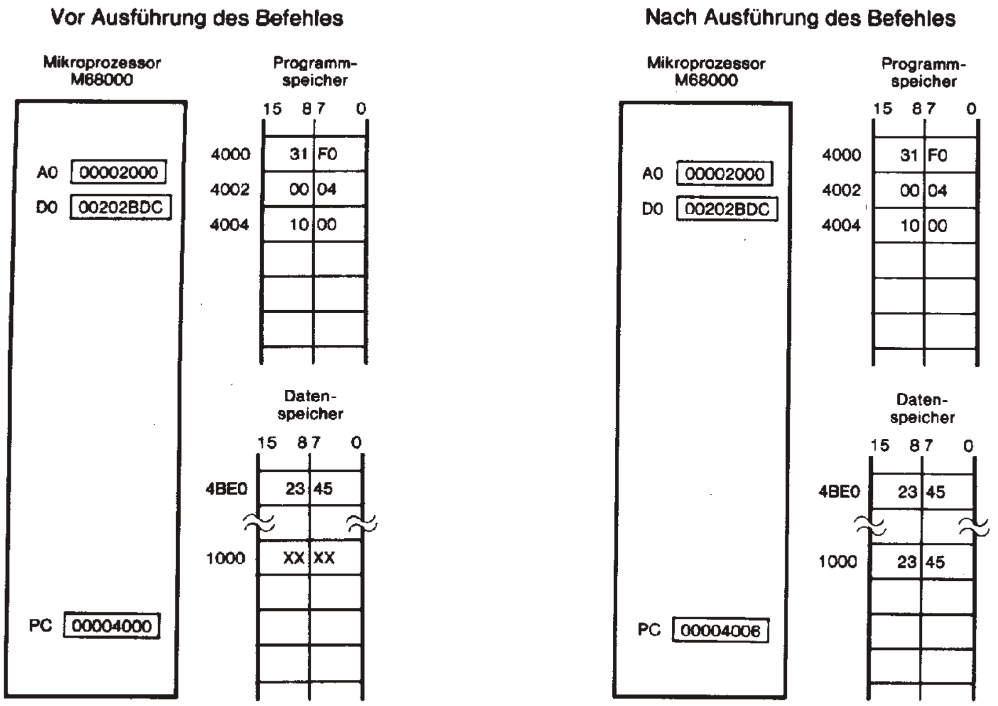
Adress-distanz

Basis-adresse

Index

eff. Ziel-adresse

Register-indirekte Adressierung (10)


Vor Ausführung des Befehles
Mikroprozessor M68000
A0 00002000
D0 00202BDC
PC 00004000
Programm-speicher
15 8 7 0
4000 31 F0
4002 00 04
4004 10 00
Daten-speicher
15 8 7 0
4BE0 23 45
1000 XX XX
Nach Ausführung des Befehles
Mikroprozessor M68000
A0 00002000
D0 00202BDC
PC 00004006
Programm-speicher
15 8 7 0
4000 31 F0
4002 00 04
4004 10 00
Daten-speicher
15 8 7 0
4BE0 23 45
1000 23 45

Adressierung relativ zum Programmzähler (1)

Der **Programmzähler (PC) dient als Basisregister**. Dazu kann eine positive oder negative Distanz angegeben werden sowie ein Indexregister. Dabei wird die Tatsache ausgenutzt, dass die zu verarbeitenden Daten häufig in der Nähe der Maschinenbefehle im Speicher stehen (modulare Programmierung!).

Vorteil gegenüber absoluten Adressen

Im Mehrbenutzerbetrieb können Programme “reentrant“ geschrieben werden: Derselbe Programmcode läuft korrekt an verschiedenen Speicheradressen, mit jeweils anderen Benutzerdaten.

Vorteile gegenüber expliziter Nennung des Basisregisters

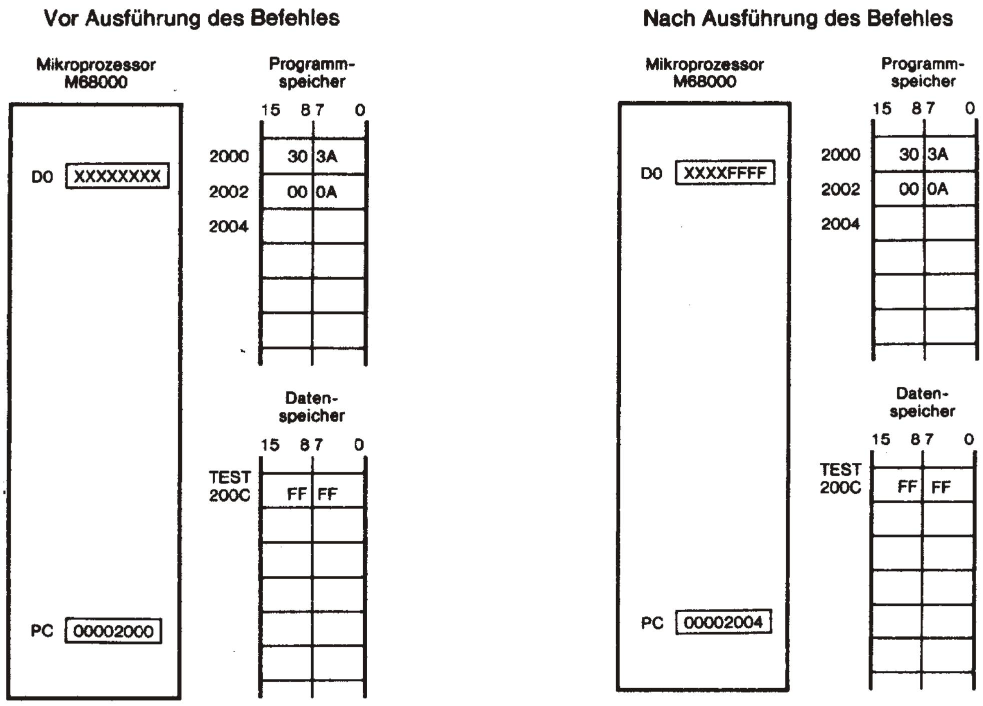
Die Verwaltung des Basisregisters bleibt dem Programmierer erspart. Die Befehle sind kürzer, da die Bitcodierung für das Basisregister entfällt.

Adressierung relativ zum Programmzähler (2)

Beispiel

	MOVE	TEST(PC),D0
Gleichbedeutend mit:	MOVE.W	TEST(PC),D0
Oder (für diesen Fall)	MOVE.W	$A(PC),D0
Oder (für diesen Fall)	MOVE.W	10(PC),D0

Adressierung relativ zum Programmzähler (3)


Vor Ausführung des Befehles
Mikroprozessor M68000
D0 XXXXXXXX
PC 00002000
Programm-speicher
15 8 7 0
2000 30 3A
2002 00 0A
2004
Daten-speicher
15 8 7 0
TEST 200C FF FF
Nach Ausführung des Befehles
Mikroprozessor M68000
D0 XXXXFFFF
PC 00002004
Programm-speicher
15 8 7 0
2000 30 3A
2002 00 0A
2004
Daten-speicher
15 8 7 0
TEST 200C FF FF

Programmbeispiel mit PC als Basisregister

Programmbeispiel 1: Absolutes Programm

```
10            00000000           ORG.S  $2000
20  002000    3038200C           MOVE TEST, D0
21  002004    4E71               NOP
22  002006    4E71               NOP
23  002008    4E71               NOP
24  00200A    4E71               NOP
25  00200C    FFFF       TEST    DC    $FFFF
30                               END
```

Programmbeispiel 2: Verschiebbares Programm

```
10                               ORG   $2000
20  002000    303A000A           MOVE TEST(PC), D0
21  002004    4E71               NOP
22  002006    4E71               NOP
23  002008    4E71               NOP
24  00200A    4E71               NOP
25  00200C    FFFF       TEST    DC   $FFFF
30                               END
```

Zusammenfassung der Adressierungsarten

Adressierungsart	Mnemonik
Datenregister direkt	Dn
Adressregister direkt	An
Adressregister indirekt (ARI)	(An)
ARI mit Postinkrement	(An)+
ARI mit Prädekrement	-(An)
ARÌ mit Adressdistanz	d16(An)
ARI mit Adressdistanz und Index	d8(An,Rx)
Absolut kurz	$XXXX
Absolut lang	$XXXXXXXX
PC relativ mit Adressdistanz	d16(PC)
PC rel. Mit Adressdist. & Index	d8(PC,Rx)
Konstante, Statusregister	#, SR, CCR