Registermaschine	Motorola 68000
Zentrale Recheneinheit	**Zentrale Recheneinheit**
Befehlszähler (b)	Befehlszähler (`PC`)
Akkumulator (c_0)	8 Datenregister (`D0` bis `D7`)
Wertebereich des Akkumulators: $\mathbb{N}$	Wertebereich eines Registers: $\{0, \ldots, 2^{32}-1\}$ oder: $\{-2^{31}, \ldots, 2^{31}-1\}$
	8 Adreßregister (`A0` bis `A7`)
	Wertebereich eines Registers: $\{0, \ldots, 2^{32}-1\}$
	Spezialregister (`CCR`, `SR`, etc.)
Speicherregister	**Hauptspeicher**
c_1, c_2, c_3, ... (beliebig viele Register)	`$000000` bis `$ffffff`, also 2^{24} Speicherzellen
Wertebereich eines Speicherregisters: $\mathbb{N}$	Wertebereich einer Speicherzelle: $\{0, \ldots, 2^{8}-1\}$ oder: $\{-2^{7}, \ldots, 2^{7}-1\}$
	Zwei oder vier Speicherzellen können zu einer größeren Zelle zusammengefaßt werden.

Allgemeiner Aufbau eines 68000-Befehls

Befehlsname. *Verarbeitungsbreite* *Quelloperand*, *Zieloperand* oder
Befehlsname. *Verarbeitungsbreite* *Zieloperand*

Befehlsnamen können z.B. sein `ADD` (addiere), `MOVE` (bewege), `ASR` (arithmetisches Schieben nach rechts), `CMP` (compare/vergleiche), `BGE` (branch if greater or equal/verzweige wenn größer oder gleich) etc.

Die Verarbeitungsbreite gibt an, wie viele der Speicherzellen des Hauptspeichers zusammengefaßt werden sollen bzw. welcher Teil eines Registers verwendet werden soll. Es sind zulässig: `B` (byte, 8 bit), `W` (word, 2 byte, 16 bit) und `L` (long word, 4 byte, 32 bit). Allerdings sind nicht für jeden Befehl alle Verarbeitungsbreiten erlaubt.

Beispiel

```
ADD.W    D1,D2
```

Nimm die unteren 16 Bit des Inhaltes des Datenregisters `D1`, addiere sie zu den unteren 16 Bit des Datenregisters `D2` und speichere das Ergebnis in den unteren 16 Bit des Datenregisters `D2`.

Indirekte Adressierung

Der Motorola 68000 verfügt über sehr mächtige Möglichkeiten zur indirekten Adressierung. Anders als bei der Registermaschine kann jedoch nur über die Adreßregister der zentralen Recheneinheit indirekt adressiert werden. Folgende Adressierungsarten sind zulässig:

Schreibweise	Benennung	angespr. Adresse	neuer Inhalt von `An`
`(An)`	einfach indirekt	`[An]`	
`-(An)`	indirekt mit Predecrement	`[An]−vb`	`[An]−vb`
`(An)+`	indirekt mit Postinkrement	`[An]`	`[An]+vb`
`d(An)`	indirekt mit Offset	`[An]+d`	
`d(An,Dn)`	indirekt mit Offset und Index	`[An]+d+[Dn]`	

(`[An]` bezeichnet den Inhalt des Adreßregisters `An`, `vb` die Verarbeitungsbreite des Befehls in Byte.)

C — Assembler — Maschinencode

C-Quelltext	Pseudo-Zwischencode	Assemblercode (Motorola 68000)	Maschinencode (hexadezimal)	Takte (68000)
`z = 0;` `i = 0;`	`D0 = 0` `D1 = 0`	`01: CLR.W D0` `02: CLR.W D1`	`42 40` `42 41`	4 4
	`A0 = &a` `A1 = &b`	`03: LEA.L a,A0` `04: LEA.L b,A1`	`41 f9 xx xx xx xx` `43 f9 xx xx xx xx`	12 12
`while (i < n) {`	verzweige	`05: BRA.B 13`	`60 12`	10

`b[i] = (i+1) *a[i+1];`	`D2 = 1`
	`D2 += D1`
	`D3 = D1`
	`D3 += D3`
	`D2 *= A0[D3+2]`
	`A1[D3] = D2`
		`06: MOVEQ.L #1,D2`
		`07: ADD.W D1,D2`
		`08: MOVE.W D1,D3`
		`09: ADD.W D3,D3`
		`10: MULS.W 2(A0,D3.W),D2`
		`11: MOVE.W D2,(A1,D3.W)`
			`74 01`
			`d4 41`
			`36 01`
			`d6 43`
			`c5 f0 30 02`
			`33 82 30 00`
				4
				4
				4
				4
				$\leq$ 80
				14

`i = i+1;`	`D1 += 1`	`12: ADDQ.W #1,D1`	`52 41`	4
`} /* while (i < n) */`	`D1 < n ?` wenn ja, verzweige	`13: CMP.W n,D1` `14: BLT.B 06`	`b2 79 xx xx xx xx` `6d e6`	16 10/8

Bedeutung einiger Assemblerbefehle:

`CLR`	clear	lösche (setze auf Null)	`CMP`	compare	vergleiche
`LEA`	load effective address	lade Speicheradresse	`BLT`	branch if less than	verzweige wenn kleiner
`MOVE`	move	bewege	`BNE`	branch if not equal	verzweige wenn ungleich
`MULS`	multiply signed	multipliziere mit Vorzeichen	`BRA`	branch always	verzweige immer

C-Quelltext	Pseudo-Zwischencode	Assemblercode (Motorola 68000)	Maschinencode (hexadezimal)	Takte (68000)
`i = n-1;`	`D1 = -1` `D1 += n`	`15: MOVEQ.L #-1,D1` `16: ADD.W n,D1`	`72 ff` `d2 79 xx xx xx xx`	4 16

`y = b[n-1]`	`A2 = &y`
	`D2 = n`
	`D2 += D2`
	`*A2 = A1[D2-2]`
		`17: LEA.L y,A2`
		`18: MOVE.W n,D2`
		`19: ADD.W D2,D2`
		`20: MOVE.W -2(A1,D2.W),(A2)`
			`45 f9 xx xx xx xx`
			`34 39 xx xx xx xx`
			`d4 42`
			`34 b1 20 fe`
				12
				16
				4
				18

`while (i >= 1) {`	verzweige	`21: BRA.B 35`	`60 26`	10

`y = y *x +b[i-1];`	`D2 = *A2`
	`D2 *= x`
	`D3 = D1`
	`D3 += D3`
	`D2 += A1[D3-2]`
	`*A2 = D2`
		`22: MOVE.W (A2),D2`
		`23: MULS.W x,D2`
		`24: MOVE.W D1,D3`
		`25: ADD.W D3,D3`
		`26: ADD.W -2(A1,D3.W),D2`
		`27: MOVE.W D2,(A2)`
			`34 12`
			`c5 f9 xx xx xx xx`
			`36 01`
			`d6 43`
			`d4 71 30 fe`
			`34 82`
				8
				$\leq$ 82
				4
				4
				14
				8

`if (y == 0) {`	`D2 != 0 ?` wenn ja, verzweige	`28: TST.W D2` `29: BNE.W 33`	`4a 42` `66 08`	4 10/8

`z = z+1;`
`a[i] = 0;`
`}`
	`D0 += 1`
	`A0[D3] = 0`
	verzweige
		`30: ADDQ.W #1,D0`
		`31: CLR.W (A0,D3.W)`
		`32: BRA.B 34`
			`52 40`
			`42 70 30 00`
			`60 06`
				4
				18
				10

`else a[i] = 1;`	`A0[D3] = 1`	`33: MOVE.W #1,(A0,D3.W)`	`31 bc 00 01 30 00`	18
`i = i-1;`	`D1 -= 1`	`34: SUBQ.W #1,D1`	`53 41`	4
`} /* while (i >= 1) */`	`D1 >= 1 ?` wenn ja, verzweige	`35: CMP.W #1,D1` `36: BGE.B 22`	`b2 7c 00 01` `6c d4`	8 10/8