M68k Cheat Sheet für BSVC

Erste Schritte

Verzeichnis Programme\BSVC\Samples in Homeverzeichnis kopieren
File → open setup → simple.setup öffnen
File → Load program → example.h86 öffnen
Links: Datenregister (Dx), Adressregister (Ax), PC und Status-Register (SR)
Unten: Bedienknöpfe für Programabarbeitung
Prozessor starten: Reset → PC := [0004] (Zeigt auf ersten Befehl)
View → Program Listing:
Blaue Markierung zeigt nächsten auszuführenden Befehl
Rot markierte Zeile ist Breakpoint
View → Memory: pro Zeile Adresse u. 16 Bytes des Programmspeichers in hex

Editieren

Quelltextdatei (z.B. example.asm) mit beliebigem Editor öffnen
Einrückungen mit Leerzeichen (bei Tabs derzeit keine korrekte Übernahme in das Listing), letzte Zeile muss Leerzeile sein
Ganzzeiligen Kommentar mit * einleiten
1. Spalte: Label / symbolische Adressen
2. Spalte: Mnemonic (der eigentliche Befehl)
3. Spalte: Operanden, ggf. durch Komma getrennt und ohne Leerzeichen
4. Spalte: Kommentare. Üblich ist, diese durch ; einzuleiten

Überrsetzen

`68kasm -l example.asm` Achtung: Befehl abtippen, nicht kopieren!

Operandenangaben

Ziel := [Quelle] op [Ziel] in der Reihenfolge **Quelle,Ziel**

Der zweite Operand ist gleichzeitig der Zieloperand
Mindestens ein Operand muss ein Register sein!

Angabe der Datenbreite durch Befehls-Postfix

.b – 8 Bit (Byte), bei Registern untere 8 Bit
.w – 16 Bit (Word), bei Registern untere 16 Bit
.l – 32 Bit (Longword)

Status Register (SR): rechte 8 Bit beinhalten das CCR: `000XNZVC`

Adressierungsarten

Immediate mode	`move.b #1,D0`	
Direct mode	`move.b 2,D0`	`move.b D2,D0`
Indirect mode	`move.b (2),D0`	`move.b (A2),D0`
Indirect mode mit Post-Increment	`move.b (A2)+,D0`	
Indirect mode mit Pre-Decrement	`move.b -(A2),D0`	
Index mode	`move.b 2(A2),D0`	
Index mode mit Verschiebung	`move.b 2(D2,A2),D0`	

Bei den index modes entspricht die resultierende Adresse der Summe der Adressangaben.

Zahlenformate

dezimal	`move.b 100,D0`
hexadezimal	`move.b $FF,D0`
binär	`move.b %10010010,D0`
oktal	`move.b @100,D0`

Datenbewegungs-Befehle

`move.b D1,D2`	Inhalt Kopieren
`lea <Adresse>,A2`	lade Adresse, beispielsweise eines Labels
`ext.w D1`	8 Bit → 16 Bit (vorzeichenrichtige Erweiterung)
`ext.l D1`	16 Bit →32 Bit (vorzeichenrichtige Erweiterung)

Arithmetik

`add.b D1,D2`	D2 := D2 + D1 (setzt X-Flag bei Überlauf)
`addx.b D1,D2`	D2 := D2 + D1 + X
`sub.b D1,D2`	D2 := D2 - D1
`mulu.b D1,D2`	D2 := D2 · D1 (ohne Vorzeichen),
`muls.b D1,D2`	D2 := D2 · D1 (mit Vorzeichen)
`divu.w D1,D2`	D2 := D2 : D1 (ohne Vorzeichen), D2: Rest und Quotient
`divs.w D1,D2`	D2 := D2 : D1 (mit Vorzeichen), D2: Rest und Quotient

Logische Operationen

`and.b D1,D2`	D2 := D2 and D1 (Bitweises and)
`or.b D1,D2`	D2 := D2 or D1 (Bitweises or)
`xor.b D1,D2`	D2 := D2 xor D1 (Bitweises xor)
`not.b D1`	D2 := not D1 (Bitweises not)

Bit-Test

`btst.b #0,D2`	Test Bit-Nr. 0 in D2, wenn 0 →Z=1

Stack- und Unterprogramm-Befehle

`move.l D1,-(A7)`	Register D1 auf Stack sichern
`move.l (A7)+,D1`	Register D1 vom Stack zurückholen
`move.l 4(A7),D1`	32-Bit-Wort von Adresse (Stack+4) nach d1 kopieren
`move.l D1,4(A7)`	32-Bit-Wort von D1 nach Adresse (Stack+4) kopieren
`bsr <adresse>`	rufe UP an <adresse> auf
`rts`	beende UP und kehre zum Hauptprogramm zurück

Schiebe- und Rotations-Befehle

`lsr.l D1`	logisches Shift rechts, links (MSB) wird 0 eingefügt		
`asr.l D1`	arithmetisches Shift rechts, links (MSB) wird Vorz. eingefügt		
`lsl.l D1`	shilft links, arithmetisch und logisch sind identisch		
`ror.l D1`	Rotation rechts	`roxr.l D1`	… mit X-Bit
`rol.l D1`	Rotation links	`roxl.l D1`	… mit X-Bit

Bedingte Sprünge

Einzelbitabfrage		ohne Vorzeichen			mit Vorzeichen		
Befehl	NZVC	Befehl	Bed.	NZVC O	Befehl	Bed.	NZVC O
bpl	0xxx	bhi	>	x0x0 &	bgt	<	00xx &
bmi	1xxx	bhs	≥	xxx0	bge	≤	0xxx
bvc	xx0x	bne	≠	x0xx	bne	≠	x0xx
bvs	xx1x	beq	=	x1xx	beq	=	x1xx
bcc	xxx0	bls	≤	x1x1 |	ble	≤	11xx |
bcs	xxx1	blo	<	xxx1	blt	<	1xxx

O = & - UND-Verknüpfung der Flags; O = | - ODER-Verknüpfung der Flags
Zusatz `.s` am Befehl gibt an, dass ein kurzer, relativer Sprung erfolgt

Direktiven

label	`equ <ausdruck>`	label wird folgend durch ausdruck ersetzt
	`org $2000`	folgender Programmcode bzw. Daten stehen ab dieser Adresse

Vorbelegter Speicher und Speicherreservierung

`dc.b 50`	ein Byte mit Inhalt 50
`dc.b 01,02,03,04,05`	mehrere Bytes
`dc.b 'DC Example'`	mehrere Bytes, ASCII-codiert
`ds.b 50`	50 Bytes reservieren