

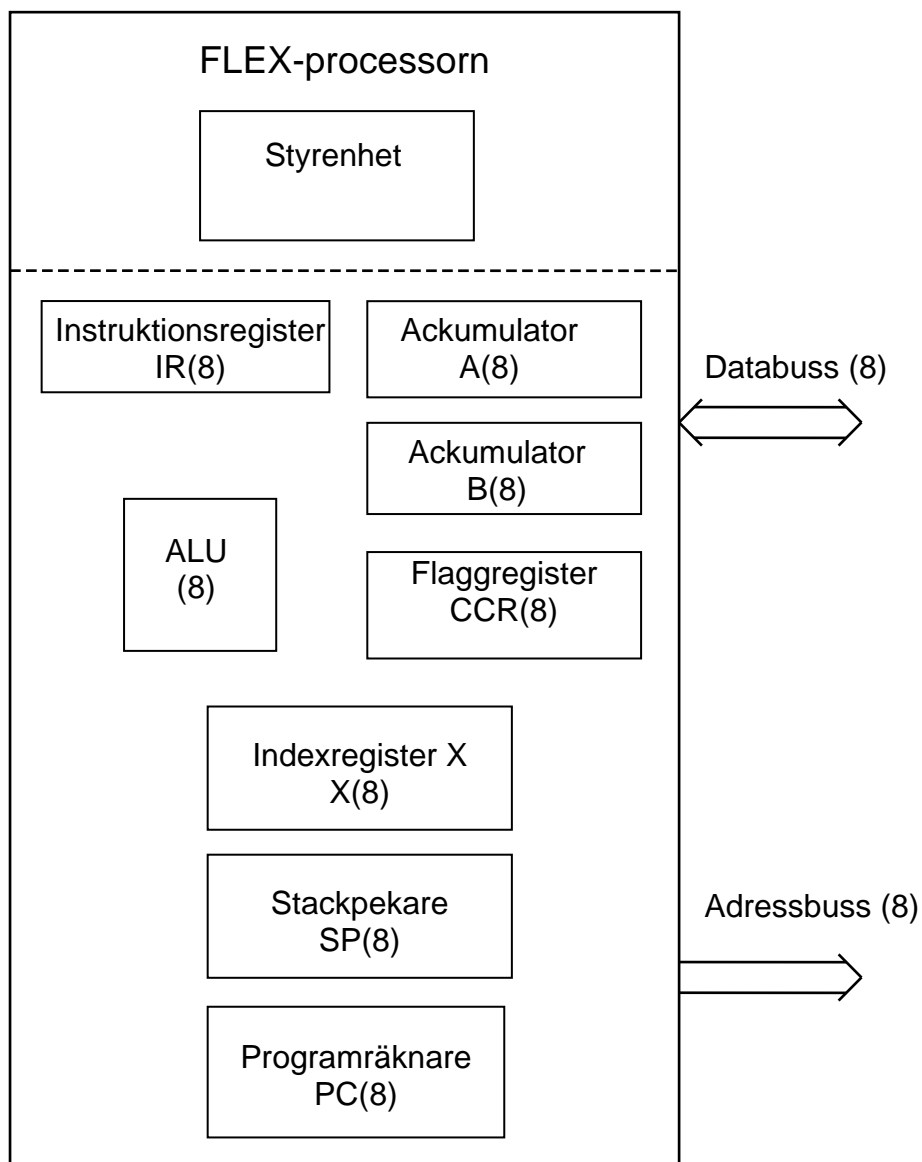
**INSTRUKTIONSLISTA  
för  
FLEX-processorn**

**2014-01-30**

## Innehåll

<b>Sidan</b>	
<b>3</b>	<b>Programmerarens bild av FLEX-processorn</b>
<b>4</b>	<b>Förklaring av beteckningar i instruktionslistan</b>
<b>5</b>	<b>Enkel dataflyttning</b>
<b>5</b>	<b>Logik</b>
<b>6</b>	<b>Aritmetik</b>
<b>6</b>	<b>No operation</b>
<b>7</b>	<b>Test</b>
<b>7</b>	<b>Hopp</b>
<b>7</b>	<b>Hopp eller Branch (Förgrening) med PC-relativ adressering</b>
<b>8</b>	<b>Manipulering av X-registrets innehåll</b>
<b>8</b>	<b>Dataflyttning med adressering via SP- och X-registret</b>
<b>9</b>	<b>Logik, Aritmetik och Test med adressering via SP- och X-registret</b>
<b>9</b>	<b>Hopp med adressering via X-registret</b>
<b>10</b>	<b>Hopp till subrutin och återhopp från subrutin</b>
<b>10</b>	<b>Branch till subrutin</b>
<b>10</b>	<b>Lagring av data på stack och hämtning av data från stack</b>
<b>10</b>	<b><i>Dataflyttning med PC-relativ adressering</i></b>
<b>11</b>	<b>Hopp till subrutin med adressering via X-registret</b>
<b>11</b>	<b><i>Dataflyttning med olika typer av indirekt adressering</i></b>
<b>12</b>	<b><i>Hopp med olika typer av indirekt adressering</i></b>
<b>12</b>	<b>Tillägg till operationsbeskrivning</b>
<b>13</b>	<b>Detaljerad beskrivning av FLEX-processorns instruktioner</b>
<b>23</b>	<b>Tabell med samtliga instruktioner i alfabetisk ordning</b>
<b>24</b>	<b>Tabell med samtliga instruktioner ordnade efter operationskod</b>

## Programmerarens bild av FLEX-processorn



Flaggregister CCR

7	6	5	4	3	2	1	0
-	-	-	-	N	Z	V	C

På de följande sidorna beskrivs kortfattat maskininstruktionerna för FLEX-processorn.

För varje instruktion anges den mnemoniska beteckningen, operationskod (OP), antal bytes (#), antal klockcykler (~), operationsbeskrivning och flaggpåverkan.

Tal i form av data, adress eller avstånd (offset) kan uttryckas antingen hexadecimalt, binärt eller decimalt på följande sätt:

\$tal = hexadecimalt  
 %tal = binärt  
 tal = decimalt

Avstånd (offset) är alltid ett tal med inbyggt tecken.

### Förklaring av beteckningar i instruktionslistan:

OP	Hexadecimal operationskod för instruktion.
#	Antal bytes i instruktion ( Används också för att beteckna adresseringsmoden "Immediate").
~	Antal klockcykler som krävs för att utföra en instruktion.
A	Innehåll i register A.
A <sub>1k</sub>	Innehåll i register A komplementeras (inverteras) bitvis.
M(Adr)	Minnesinnehåll på adressen Adr.
M <sub>1k</sub> (Adr)	Minnesinnehåll komplementeras (inverteras) bitvis.
N	Teckenflaggan ("Negative").
Z	Nollflaggan ("Zero")
V	Overflowflaggan.
C	Carryflaggan.
Δ	Används för att visa att en flagga påverkas av en operation.
-	Används för att visa att en flagga ej påverkas av en operation).
0	Nollställs. (Används oftast för att visa att en flagga nollställs av en operation).
1	Ettställs. (Används oftast för att visa att en flagga ettställs av en operation).
?	Odefinierat värde. (Används för att visa att en flagga får ett slumpartat värde 0 eller 1 efter en operation)
CC	"Condition Code" (Innehållet i flaggregistret, dvs samtliga flaggor).
n	Avstånd (offset). (Används i samband med adressering via X-registret och vid PC-relativ adressering. n är ett tal med inbyggt tecken.)
EA	Effektiv adress. För hopp- och branchinstruktioner avses adressen dit hoppet skall ske. För övriga instruktioner avses adressen till data.

**Observera att instruktioner skrivna *kursivt* i instruktionslistan ej kan assembleras eller simuleras med simulatorn Eterm6 för FLEX-processorn.**

## Enkel dataflytning

Instruktion		Adressering						Operationsbeskrivning*		Flaggor				
Operation	Beteckning	Inherent		Immediate		Absolute				3	2	1	0	
		OP	# ~	OP	# ~	OP	# ~			N	Z	V	C	
Transfer	TFR A,B	01	1 3							A → B	-	-	-	-
	TFR B,A	02	1 3							B → A	-	-	-	-
	TFR A,CCR	03	1 3							A → CCR	Δ	Δ	Δ	Δ
	TFR CCR,A	04	1 3							CCR → A	-	-	-	-
	TFR X,SP	05	1 3							X → SP	-	-	-	-
	TFR SP,X	06	1 3							SP → X	-	-	-	-
Exchange	EXG A,B	07	1 5							A ↔ B	-	-	-	-
	EXG A,CCR	08	1 5							A ↔ CCR	Δ	Δ	Δ	Δ
	EXG B,CCR	09	1 5							B ↔ CCR	Δ	Δ	Δ	Δ
	EXG X,SP	0A	1 5							X ↔ SP	-	-	-	-
Load	LDAA Adr					0B	2 5			M(Adr) → A	-	-	-	-
	LDAB Adr					0C	2 5			M(Adr) → B	-	-	-	-
	LDX Adr					0D	2 5			M(Adr) → X	-	-	-	-
	LDS Adr					0E	2 5			M(Adr) → SP	-	-	-	-
	LDAA #Data			0F	2 4					Data → A	-	-	-	-
	LDAB #Data			10	2 4					Data → B	-	-	-	-
	LDX #Data			11	2 4					Data → X	-	-	-	-
	LDS #Data			12	2 4					Data → SP	-	-	-	-
Store	STAA Adr					13	2 5			A → M(Adr)	-	-	-	-
	STAB Adr					14	2 5			B → M(Adr)	-	-	-	-
	STX Adr					15	2 5			X → M(Adr)	-	-	-	-
	STS Adr					16	2 5			SP → M(Adr)	-	-	-	-

## Logik

Instruktion		Adressering						Operationsbeskrivning*		Flaggor			
Operation	Beteckning	Inherent		Immediate		Absolute				3	2	1	0
		OP	# ~	OP	# ~	OP	# ~			N	Z	V	C
AND	ANDA Adr					17	2 7			A AND M(Adr) → A	Δ	Δ	0 0
	ANDB Adr					18	2 7			B AND M(Adr) → B	Δ	Δ	0 0
	ANDA #Data			19	2 6					A AND Data → A	Δ	Δ	0 0
	ANDB #Data			1A	2 6					B AND Data → B	Δ	Δ	0 0
OR	ORAA Adr					1B	2 7			A OR M(Adr) → A	Δ	Δ	0 0
	ORAB Adr					1C	2 7			B OR M(Adr) → B	Δ	Δ	0 0
	ORAA #Data			1D	2 6					A OR Data → A	Δ	Δ	0 0
	ORAB #Data			1E	2 6					B OR Data → B	Δ	Δ	0 0
Exclusive-OR	EORA Adr					1F	2 7			A XOR M(Adr) → A	Δ	Δ	0 0
	EORB Adr					20	2 7			B XOR M(Adr) → B	Δ	Δ	0 0
	EORA #Data			21	2 6					A XOR Data → A	Δ	Δ	0 0
	EORB #Data			22	2 6					B XOR Data → B	Δ	Δ	0 0
Complement	COMA	23	1 4							A <sub>1k</sub> → A	Δ	Δ	0 ?
	COMB	24	1 4							B <sub>1k</sub> → B	Δ	Δ	0 ?
	COM Adr					25	2 6			M <sub>1k</sub> (Adr) → M(Adr)	Δ	Δ	0 ?
Flag manipulation	ANDCC #Data			26	2 6					CCR AND Data → CCR	Δ	Δ	Δ Δ
	ORCC #Data			27	2 6					CCR OR Data → CCR	Δ	Δ	Δ Δ

## Aritmetik

Instruktion		Adressering						Operations- beskrivning*	Flaggor				
Operation	Beteckning	Inherent		Immediate		Absolute				3	2	1	0
		OP	# ~	OP	# ~	OP	# ~			N	Z	V	C
Add	ADDA Adr					28	2	7	$A+M(\text{Adr}) \rightarrow A$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	ADDB Adr					29	2	7	$B+M(\text{Adr}) \rightarrow B$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	ADDA #Data			2A	2	6			$A+\text{Data} \rightarrow A$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	ADDB #Data			2B	2	6			$B+\text{Data} \rightarrow B$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Add with carry	ADCA Adr					2C	2	7	$A+M(\text{Adr})+C \rightarrow A$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	ADCB Adr					2D	2	7	$B+M(\text{Adr})+C \rightarrow B$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	ADCA #Data			2E	2	6			$A+\text{Data}+C \rightarrow A$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	ADCB #Data			2F	2	6			$B+\text{Data}+C \rightarrow B$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Subtract	SUBA Adr					30	2	7	$A-M(\text{Adr}) \rightarrow A$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	SUBB Adr					31	2	7	$B-M(\text{Adr}) \rightarrow B$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	SUBA #Data			32	2	6			$A-\text{Data} \rightarrow A$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	SUBB #Data			33	2	6			$B-\text{Data} \rightarrow B$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Subtract with borrow	SBCA Adr					34	2	7	$A-M(\text{Adr})-C \rightarrow A$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	SBCB Adr					35	2	7	$B-M(\text{Adr})-C \rightarrow B$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	SBCA #Data			36	2	6			$A-\text{Data}-C \rightarrow A$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	SBCB #Data			37	2	6			$B-\text{Data}-C \rightarrow B$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Negate (2's-compl)	NEGA	38	1	5					$A_{1k+1} \rightarrow A$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	NEGB	39	1	5					$B_{1k+1} \rightarrow B$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	NEG Adr					3A	2	7	$M_{1k}(\text{Adr})+1 \rightarrow M(\text{Adr})$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Arithmetic shift left	ASLA	3B	1	4					$2A \rightarrow A$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	ASLB	3C	1	4					$2B \rightarrow B$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	ASL Adr					3D	2	6	$2M(\text{Adr}) \rightarrow M(\text{Adr})$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Rotate left	ROLA	3E	1	4					$2A+C \rightarrow A$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	ROLB	3F	1	4					$2B+C \rightarrow B$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	ROL Adr					40	2	6	$2M(\text{Adr}) + C \rightarrow M(\text{Adr})$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Increment	INCA	41	1	4					$A+1 \rightarrow A$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	INCB	42	1	4					$B+1 \rightarrow B$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	INX	E1	1	4					$X+1 \rightarrow X$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	INC Adr					43	2	6	$M(\text{Adr})+1 \rightarrow M(\text{Adr})$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Decrement	DECA	44	1	4					$A-1 \rightarrow A$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	DECB	45	1	4					$B-1 \rightarrow B$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	DEX	E2	1	4					$X-1 \rightarrow X$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	DEC Adr					46	2	6	$M(\text{Adr}) - 1 \rightarrow M(\text{Adr})$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Clear	CLRA	47	1	4					$0 \rightarrow A$	0	1	?	0
	CLRB	48	1	4					$0 \rightarrow B$	0	1	?	0
	CLR Adr					49	2	5	$0 \rightarrow M(\text{Adr})$	0	1	?	0

## No operation

Instruktion		Adressering						Operations- beskrivning*	Flaggor				
Operation	Beteckning	Inherent		Immediate		Absolute				3	2	1	0
		OP	# ~	OP	# ~	OP	# ~			N	Z	V	C
No operation	NOP	00	1	3					No operation	-	-	-	-

## Test

Instruktion		Adressering						Operationsbeskrivning*		Flaggor			
Operation	Beteckning	Inherent		Immediate		Absolut				3	2	1	0
		OP	# ~	OP	# ~	OP	# ~			N	Z	V	C
Compare	CMPA Adr					4A	2 6	A-M(Adr)		Δ	Δ	Δ	Δ
	CMPB Adr					4B	2 6	B-M(Adr)		Δ	Δ	Δ	Δ
	CPX Adr					4C	2 6	X-M(Adr)		Δ	Δ	Δ	Δ
	CPS Adr					4D	2 6	SP-M(Adr)		Δ	Δ	Δ	Δ
	CMPA #Data			4E	2 5			A-Data		Δ	Δ	Δ	Δ
	CMPB #Data			4F	2 5			B-Data		Δ	Δ	Δ	Δ
	CPX #Data			50	2 5			X-Data		Δ	Δ	Δ	Δ
	CPS #Data			51	2 5			SP-Data		Δ	Δ	Δ	Δ
Test, zero or minus	TSTA	52	1 3					A-0		Δ	Δ	0	0
	TSTB	53	1 3					B-0		Δ	Δ	0	0
	TST Adr					54	2 5	M(Adr)-0		Δ	Δ	0	0
Bit test	BITA Adr					55	2 6	A AND M(Adr)		Δ	Δ	0	0
	BITB Adr					56	2 6	B AND M(Adr)		Δ	Δ	0	0
	BITA #Data			57	2 5			A AND Data		Δ	Δ	0	0
	BITB #Data			58	2 5			B AND Data		Δ	Δ	0	0

## Hopp

Instruktion		Adressering						Operationsbeskrivning*		Flaggor			
Operation	Beteckning	Inherent		Immediate		Absolut				3	2	1	0
		OP	# ~	OP	# ~	OP	# ~			N	Z	V	C
Unconditional jump	JMP Adr					59	2 4	Adr → PC		-	-	-	-

## Hopp eller Branch (Förgrening) med PC-relativ adressering

Instruktion		Adressering			Operationsbeskrivning*		Flaggor			
Operation	Beteckning	Relative					3	2	1	0
		OP	#	~			N	Z	V	C
Unconditional branch	BRA Adr	5A	2	5	PC+Offs → PC		-	-	-	-
Conditional branch										
Simple conditions	BMI Adr	5B	2	5	If N = 1: PC+Offs → PC		-	-	-	-
	BPL Adr	5C	2	5	If N = 0: PC+Offs → PC		-	-	-	-
	BEQ Adr	5D	2	5	If Z = 1: PC+Offs → PC		-	-	-	-
	BNE Adr	5E	2	5	If Z = 0: PC+Offs → PC		-	-	-	-
	BVS Adr	5F	2	5	If V = 1: PC+Offs → PC		-	-	-	-
	BVC Adr	60	2	5	If V = 0: PC+Offs → PC		-	-	-	-
	BCS Adr	61	2	5	If C = 1: PC+Offs → PC		-	-	-	-
	BCC Adr	62	2	5	If C = 0: PC+Offs → PC		-	-	-	-
Unsigned numbers	BHI Adr	63	2	5	If C'Z' = 1: PC+Offs → PC		-	-	-	-
	BHS Adr	62	2	5	If C = 0: PC+Offs → PC		-	-	-	-
	BEQ Adr	5D	2	5	If Z = 1: PC+Offs → PC		-	-	-	-
	BNE Adr	5E	2	5	If Z = 0: PC+Offs → PC		-	-	-	-
	BLS Adr	64	2	5	If C+Z = 1: PC+Offs → PC		-	-	-	-
	BLO Adr	61	2	5	If C = 1: PC+Offs → PC		-	-	-	-
Signed numbers	BGT Adr	65	2	5	If (N⊕V) + Z = 0: PC+Offs → PC		-	-	-	-
	BGE Adr	66	2	5	If (N⊕V) = 0: PC+Offs → PC		-	-	-	-
	BEQ Adr	5D	2	5	If Z = 1: PC+Offs → PC		-	-	-	-
	BNE Adr	5E	2	5	If Z = 0: PC+Offs → PC		-	-	-	-
	BLE Adr	67	2	5	If (N⊕V) + Z = 1: PC+Offs → PC		-	-	-	-
	BLT Adr	68	2	5	If (N⊕V) = 1: PC+Offs → PC		-	-	-	-

## Manipulering av X- och SP-registrets innehåll

Instruktion		Adressering			Operations- beskrivning*	Flaggor			
Operation	Beteckning	Via X					3	2	1
		OP	#	~	N		Z	V	C
Load effective address	LEAX 1,-X	74	1	4	$X - 1 \rightarrow X$	-	-	-	-
	LEAX 1,X+	75	1	4	$X + 1 \rightarrow X$	-	-	-	-
	LEAX n,X	76	2	6	$X + n \rightarrow X$	-	-	-	-
	LEAX A,X	77	1	5	$X + A \rightarrow X$	-	-	-	-
	LEAX B,X	78	1	5	$X + B \rightarrow X$	-	-	-	-
	LEAS n,SP	E3	2	6	$SP + n \rightarrow SP$	-	-	-	-

## Dataflyttning med adressering via X-registret

Instruktion		Adressering			Operations- beskrivning*	Flaggor			
Operation	Beteckning	Via X					3	2	1
		OP	#	~	N		Z	V	C
Load	LDAA ,X	79	1	4	$M(X) \rightarrow A$	-	-	-	-
	LDAB ,X	7A	1	4	$M(X) \rightarrow B$	-	-	-	-
	LDAA 1,X+	7B	1	5	$M(X) \rightarrow A$ $X+1 \rightarrow X$	-	-	-	-
	LDAB 1,X+	7C	1	5	$M(X) \rightarrow B$ $X+1 \rightarrow X$	-	-	-	-
	LDAA 1,-X	7D	1	5	$X-1 \rightarrow X$ $M(X) \rightarrow A$	-	-	-	-
	LDAB 1,-X	80	1	5	$X-1 \rightarrow X$ $M(X) \rightarrow B$	-	-	-	-
	LDAA n,X	81	2	7	$M(n+X) \rightarrow A$	-	-	-	-
	LDAB n,X	82	2	7	$M(n+X) \rightarrow B$	-	-	-	-
	LDAA A,X	83	1	6	$M(A+X) \rightarrow A$	-	-	-	-
	LDAB A,X	84	1	6	$M(A+X) \rightarrow B$	-	-	-	-
	LDAA B,X	85	1	6	$M(B+X) \rightarrow A$	-	-	-	-
	LDAB B,X	86	1	6	$M(B+X) \rightarrow B$	-	-	-	-
	LDX A,X	87	1	6	$M(A+X) \rightarrow X$	-	-	-	-
	LDX B,X	88	1	6	$M(B+X) \rightarrow X$	-	-	-	-
Store	STAA ,X	89	1	4	$A \rightarrow M(X)$	-	-	-	-
	STAB ,X	8A	1	4	$B \rightarrow M(X)$	-	-	-	-
	STAA 1,X+	8B	1	5	$A \rightarrow M(X)$ $X+1 \rightarrow X$	-	-	-	-
	STAB 1,X+	8C	1	5	$B \rightarrow M(X)$ $X+1 \rightarrow X$	-	-	-	-
	STAA 1,-X	8D	1	5	$X-1 \rightarrow X$ $A \rightarrow M(X)$	-	-	-	-
	STAB 1,-X	8E	1	5	$X-1 \rightarrow X$ $B \rightarrow M(X)$	-	-	-	-
	STAA n,X	8F	2	7	$A \rightarrow M(n+X)$	-	-	-	-
	STAB n,X	90	2	7	$B \rightarrow M(n+X)$	-	-	-	-
	STAA A,X	91	1	6	$A \rightarrow M(A+X)$	-	-	-	-
	STAB A,X	92	1	6	$B \rightarrow M(A+X)$	-	-	-	-
STAA B,X	93	1	6	$A \rightarrow M(B+X)$	-	-	-	-	
STAB B,X	94	1	6	$B \rightarrow M(B+X)$	-	-	-	-	



## Logik, Aritmetik och Test med adressering via X- och SP-registret

Instruktion		Adressering			Operations- beskrivning*	Flaggor			
Operation	Beteckning	Via X				3 N	2 Z	1 V	0 C
		OP	#	~					
AND	ANDA ,X	C7	1	6	$A \text{ AND } M(X) \rightarrow A$	$\Delta$	$\Delta$	0	0
	ANDB ,X	C8	1	6	$B \text{ AND } M(X) \rightarrow B$	$\Delta$	$\Delta$	0	0
OR	ORAA ,X	C9	1	6	$A \text{ OR } M(X) \rightarrow A$	$\Delta$	$\Delta$	0	0
	ORAB ,X	CA	1	6	$B \text{ OR } M(X) \rightarrow B$	$\Delta$	$\Delta$	0	0
Exclusive-OR	EORA ,X	CB	1	6	$A \text{ XOR } M(X) \rightarrow A$	$\Delta$	$\Delta$	0	0
	EORB ,X	CC	1	6	$B \text{ XOR } M(X) \rightarrow B$	$\Delta$	$\Delta$	0	0
Complement	COM ,X	CD	1	5	$M_{1k}(X) \rightarrow M(X)$	$\Delta$	$\Delta$	0	?
Add	ADDA ,X	CE	1	6	$A + M(X) \rightarrow A$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	ADDB ,X	CF	1	6	$B + M(X) \rightarrow B$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Add with carry	ADCA ,X	D0	1	6	$A + M(X) + C \rightarrow A$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	ADCB ,X	D1	1	6	$B + M(X) + C \rightarrow B$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Subtract	SUBA ,X	D2	1	6	$A - M(X) \rightarrow A$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	SUBB ,X	D3	1	6	$B - M(X) \rightarrow B$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Subtract with borrow	SBCA ,X	D4	1	6	$A - M(X) - C \rightarrow A$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	SBCB ,X	D5	1	6	$B - M(X) - C \rightarrow B$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Negate (2's-compl)	NEG ,X	D6	1	6	$-M(X) \rightarrow M(X)$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Aritmetic shift left	ASL ,X	D7	1	5	$2M(X) \rightarrow M(X)$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Rotate left	ROL ,X	D8	1	5	$2M(X) + C \rightarrow M(X)$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Increment	INC ,X	D9	1	5	$M(X) + 1 \rightarrow M(X)$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	INC ,SP	E4	1	5	$M(SP) + 1 \rightarrow M(SP)$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Decrement	DEC ,X	DA	1	5	$M(X) - 1 \rightarrow M(X)$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	DEC ,SP	E5	1	5	$M(SP) - 1 \rightarrow M(SP)$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Clear	CLR ,X	DB	1	4	$0 \rightarrow M(X)$	0	1	?	0
Compare	CMPA ,X	DC	1	5	$A - M(X)$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
	CMPB ,X	DD	1	5	$B - M(X)$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
Test, zero or minus	TST ,X	DE	1	4	$M(X) - 0$	$\Delta$	$\Delta$	0	0
Bit test	BITA ,X	DF	1	5	$A \text{ AND } M(X)$	$\Delta$	$\Delta$	0	0
	BITB ,X	E0	1	5	$B \text{ AND } M(X)$	$\Delta$	$\Delta$	0	0

## Hopp med adressering via X-registret

Instruktion		Adressering			Operations- beskrivning*	Flaggor			
Operation	Beteckning	Via X				3 N	2 Z	1 V	0 C
		OP	#	~					
Unconditional jump	JMP ,X	95	1	3	$X \rightarrow PC$	-	-	-	-
	JMP n,X	96	2	6	$n+X \rightarrow PC$	-	-	-	-
	JMP A,X	97	1	5	$A+X \rightarrow PC$	-	-	-	-
	JMP B,X	98	1	5	$B+X \rightarrow PC$	-	-	-	-



## Hopp till subrutin med adressering via X-registret eller PC

Instruktion		Adressering			Operations- beskrivning*	Flaggor			
Operation	Beteckning	Via X					3	2	1
		OP	#	~	N		Z	V	C
Jump to subroutine	JSR ,X	9A	1	6	SP-1 → SP PC → M(SP) X → PC	-	-	-	-
	JSR n,X	9B	2	8	SP-1 → SP PC → M(SP) n+X → PC	-	-	-	-
	JSR A,X	9C	1	7	SP-1 → SP PC → M(SP) A+X → PC	-	-	-	-
	JSR B,X	9D	1	7	SP-1 → SP PC → M(SP) B+X → PC	-	-	-	-

## Dataflyttning med olika typer av indirekt adressering

(Ej assemblering el simulering)

Instruktion		Adressering						Operations- beskrivning*	Flaggor					
Operation	Beteckning	Indirekt		Indirekt via X		Indirekt via PC				3	2	1	0	
		OP	#	~	OP	#	~	OP		#	~	N	Z	V
Load	LDAA [Adr]	A3	2	6					M(M(Adr)) → A	-	-	-	-	
	LDAB [Adr]	A4	2	6					M(M(Adr)) → B	-	-	-	-	
	LDAA [,X]				A5	1	5		M(M(X)) → A	-	-	-	-	
	LDAB [,X]				A6	1	5		M(M(X)) → B	-	-	-	-	
	LDAA [n,X]				A7	2	8		M(M(n+X)) → A	-	-	-	-	
	LDAB [n,X]				A8	2	8		M(M(n+X)) → B	-	-	-	-	
	LDAA [A,X]				A9	1	7		M(M(A+X)) → A	-	-	-	-	
	LDAB [A,X]				AA	1	7		M(M(A+X)) → B	-	-	-	-	
	LDAA [B,X]				AB	1	7		M(M(B+X)) → A	-	-	-	-	
	LDAB [B,X]				AC	1	7		M(M(B+X)) → B	-	-	-	-	
	LDAA [n,PCR]							AD	2	7	M(M(n+PC)) → A	-	-	-
	LDAB [n,PCR]							AE	2	7	M(M(n+PC)) → B	-	-	-
Store	STAA [Adr]	AF	2	6					A → M(M(Adr))	-	-	-	-	
	STAB [Adr]	B0	2	6					B → M(M(Adr))	-	-	-	-	
	STAA [,X]				B1	1	5		A → M(M(X))	-	-	-	-	
	STAB [,X]				B2	1	5		B → M(M(X))	-	-	-	-	
	STAA [n,X]				B3	2	8		A → M(M(n+X))	-	-	-	-	
	STAB [n,X]				B4	2	8		B → M(M(n+X))	-	-	-	-	
	STAA [A,X]				B5	1	7		A → M(M(A+X))	-	-	-	-	
	STAB [A,X]				B6	1	7		B → M(M(A+X))	-	-	-	-	
	STAA [B,X]				B7	1	7		A → M(M(B+X))	-	-	-	-	
	STAB [B,X]				B8	1	7		B → M(M(B+X))	-	-	-	-	
	STAA [n,PCR]							B9	2	7	A → M(M(n+PC))	-	-	-
	STAB [n,PCR]							BA	2	7	B → M(M(n+PC))	-	-	-

## Hopp med olika typer av indirekt adressering

(Ej assemblering el simulering)

Instruktion		Adressering						Operationsbeskrivning*	Flaggor					
Operation	Beteckning	Indirekt		Indirekt via X		Indirekt via PC			3	2	1	0		
		OP	# ~	OP	# ~	OP	# ~		N	Z	V	C		
Unconditional jump	JMP [Adr]	BB	2	5					M(Adr) → PC	-	-	-	-	
	JMP [,X]				BC	1	4		M(X) → PC	-	-	-	-	
	JMP [n,X]				BD	2	7		M(n+X) → PC	-	-	-	-	
	JMP [A,X]				BE	1	6		M(A+X) → PC	-	-	-	-	
	JMP [B,X]				BF	1	6		M(B+X) → PC	-	-	-	-	
	JMP [n,PCR]							C0	2	6	M(n+PC) → PC	-	-	-
Jump to subroutine	JSR [Adr]	C1	2	8					SP-1 → SP PC → M(SP) M(Adr) → PC	-	-	-	-	
	JSR [,X]				C2	1	7		SP-1 → SP PC → M(SP) M(X) → PC	-	-	-	-	
	JSR [n,X]				C3	2	9		SP-1 → SP PC → M(SP) M(n+X) → PC	-	-	-	-	
	JSR [A,X]				C4	1	8		SP-1 → SP PC → M(SP) M(A+X) → PC	-	-	-	-	
	JSR [B,X]				C5	1	8		SP-1 → SP PC → M(SP) M(B+X) → PC	-	-	-	-	
	JSR [n,PCR]							C6	2	8	SP-1 → SP PC → M(SP) M(n+PC) → PC	-	-	-

### \*Tillägg till operationsbeskrivning:

Lägg märke till att varje instruktion dessutom ökar innehållet i PC.

Alla instruktioner ökar PC med ett under FETCH-sekvensen.

Instruktioner som består av två ord ökar sedan PC med ytterligare ett i samband med att ord nummer två hämtas under EXECUTE-sekvensen. Ökningarna av PC görs innan offset adderas för branchinstruktioner och innan PC-värdet lagras som återhopsadress på stacken vid subrutinanrop.

**Observera att instruktioner skrivna *kursivt* i instruktionslistan ej kan assembleras eller simuleras med simulatorn Eterm6 för FLEX-processorn.**

## Detaljerad beskrivning av FLEX-processorns instruktioner

### ADC Add Memory Data with Carry into Register

Varianter: ADCA Adr; ADCB Adr; ADCA #Data; ADCB #Data; ADCA ,X; ADCB ,X

RTN:  $R + M + C \rightarrow R$ , där R = A eller B, M = data från minneasadress eller instruktionen själv och C = carryflaggan i flaggregistret (CCR).

Flaggor: N: Ettställs om resultatets teckenbit (bit 7) får värdet 1 vid additionen.  
Z: Ettställs om samtliga åtta bitar i resultatet blir noll vid additionen.  
V: Ettställs om overflow vid 2-komplementsrepresentation inträffar vid additionen.  
C: Ettställs om summan vid additionen ej ryms i åtta bitar, dvs blir större än eller lika med 256.

Beskrivning: Utför åttabitars addition av dataordet i minnet och innehållet i reg A eller reg B. Resultatets åtta minst signifikanta bitar placeras i reg A eller reg B. Den nionde biten (mest signifikant) placeras i C-biten (C-flaggan) i CC-registret. Det gamla värdet på C-biten i flaggregistret används som minnessiffra i minst signifikant position (minnessiffra in) vid additionen.

### ADD Add Memory Data into Register

Varianter: ADDA Adr; ADDB Adr; ADDA #Data; ADDB #Data; ADDA ,X; ADDB ,X

RTN:  $R + M \rightarrow R$ , där R = A eller B, M = data från minneasadress eller instruktionen själv.

Flaggor: N: Ettställs om resultatets teckenbit (bit 7) får värdet 1 vid additionen.  
Z: Ettställs om samtliga åtta bitar i resultatet blir noll vid additionen.  
V: Ettställs om overflow vid 2-komplementsrepresentation inträffar vid additionen.  
C: Ettställs om summan vid additionen ej ryms i åtta bitar, dvs blir större än eller lika med 256.

Beskrivning: Utför åttabitars addition av dataordet i minnet och innehållet i reg A eller reg B. Resultatets åtta minst signifikanta bitar placeras i reg A eller reg B. Den nionde biten (mest signifikant) placeras i C-biten (C-flaggan) i CC-registret.

### AND Logical AND Memory Data into Register

Varianter: ANDA Adr; ANDB Adr; ANDA #Data; ANDB #Data; ANDA ,X; ANDB ,X

RTN:  $R \text{ AND } M \rightarrow R$ , där R = A eller B, M = data från minneasadress eller instruktionen själv.

Flaggor: N: Ettställs om resultatets teckenbit (bit 7) får värdet 1.  
Z: Ettställs om samtliga åtta bitar i resultatet blir noll.  
V: Nollställs.  
C: Nollställs.

Beskrivning: Utför bitvis AND-operation mellan dataordet i minnet och innehållet i reg A eller reg B. Resultatet placeras i reg A eller reg B.

### ANDCC Logical AND Data into Condition Code Register

Instruktion: ANDCC #Data

RTN: CCR AND Data  $\rightarrow$  CCR

Flaggor: Flaggorna nollställs i de positioner där CCR eller Data innehåller någon nolla.

Beskrivning: Utför bitvis AND-operation mellan innehållet i flaggregistret (CCR) och dataordet. Resultatet placeras i flaggregistret.

**ASL Arithmetic Shift Left**

Varianter: ASLA; ASLB; ASL Adr; ASL ,X

RTN: 2R → R eller 2M → M

Flaggor: N: Ettställs om resultatets teckenbit (bit 7) får värdet 1.  
 Z: Ettställs om samtliga åtta bitar i resultatet blir noll.  
 V: Ettställs om overflow vid 2-komplementsrepresentation inträffar.  
 C: Teckenbiten (bit 7) före skiftet blir ny carrybit efter skiftet.

Beskrivning: Multiplicerar ett tal med inbyggt tecken i reg A, reg B eller minnet med 2.

**BCC Branch on Carry Clear (= BHS)**

Instruktion: BCC Adr

RTN: If C = 0: PC+Offset → PC

Flaggor: Påverkas ej.

Beskrivning: Testar C-flaggans värde. Om C=0 utförs ett hopp till adressen Adr = PC+Offset. Offset räknas från adressen efter branchinstruktionen, dvs vid uträkningen av hoppadressen pekar PC på operationskoden direkt efter branchinstruktionen i minnet. Om C=1 utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.

**BCS Branch on Carry Set (= BLO)**

Instruktion: BCS Adr

RTN: If C = 1: PC+Offset → PC

Flaggor: Påverkas ej.

Beskrivning: Testar C-flaggans värde. Om C=1 utförs ett hopp till adressen Adr = PC+Offset. Offset räknas från adressen efter branchinstruktionen, dvs vid uträkningen av hoppadressen pekar PC på operationskoden direkt efter branchinstruktionen i minnet. Om C=0 utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.

**BEQ Branch on Equal**

Instruktion: BEQ Adr

RTN: If Z = 1: PC+Offset → PC

Flaggor: Påverkas ej.

Beskrivning: Testar Z-flaggans värde. Om Z=1 utförs ett hopp till adressen Adr = PC+Offset. Offset räknas från adressen efter branchinstruktionen, dvs vid uträkningen av hoppadressen pekar PC på operationskoden direkt efter branchinstruktionen i minnet. Om Z=0 utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.

**BGE Branch on Greater than or Equal to Zero**

Instruktion: BGE Adr

RTN: If  $N \oplus V = 0$ : PC+Offset → PC

Flaggor: Påverkas ej.

Beskrivning: Testar värdet hos Booleska uttrycket  $N \oplus V$ . Om  $N \oplus V = 0$  utförs ett hopp till adressen Adr = PC+Offset. Offset räknas från adressen efter branchinstruktionen, dvs vid uträkningen av hoppadressen pekar PC på operationskoden direkt efter branchinstruktionen i minnet. Om  $N \oplus V = 1$  utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.

**BGT Branch on Greater than**

Instruktion: BGT Adr

RTN: If  $(N \oplus V) + Z = 0$ : PC+Offset  $\rightarrow$  PC

Flaggor: Påverkas ej.

Beskrivning: Testar värdet hos Booleska uttrycket  $(N \oplus V) + Z$ . Om  $(N \oplus V) + Z = 0$  utförs ett hopp till adressen Adr = PC+Offset. Offset räknas från adressen efter branchinstruktionen, dvs vid uträkningen av hoppadressen pekar PC på operationskoden direkt efter branchinstruktionen i minnet. Om  $(N \oplus V) + Z = 1$  utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.

**BHI Branch if Higher**

Instruktion: BHI Adr

RTN: If  $C + Z = 0$ : PC+Offset  $\rightarrow$  PC

Flaggor: Påverkas ej.

Beskrivning: Testar värdet hos Booleska uttrycket  $C + Z$ . Om  $C + Z = 0$  utförs ett hopp till adressen Adr = PC+Offset. Offset räknas från adressen efter branchinstruktionen, dvs vid uträkningen av hoppadressen pekar PC på operationskoden direkt efter branchinstruktionen i minnet. Om  $C + Z = 1$  utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.

**BHS Branch if Higher or Same (= BCC)****BIT Bit test**

Varianter: BITA Adr; BITB Adr; BITA #Data; BITB #Data; BITA ,X; BITB ,X

RTN: R AND M, där R = A eller B, M = data från minneadress eller instruktionen själv.

Flaggor: N: Ettställs om resultatets teckenbit (bit 7) får värdet 1.  
Z: Ettställs om samtliga åtta bitar i resultatet blir noll.  
V: Nollställs.  
C: Nollställs.

Beskrivning: Utför bitvis AND-operation mellan dataordet i minnet och innehållet i reg A eller reg B. Resultatet lagras ej, utan påverkar endast flaggorna.

**BLE Branch on Less than or Equal to Zero**

Instruktion: BLE Adr

RTN: If  $(N \oplus V) + Z = 1$ : PC+Offset  $\rightarrow$  PC

Flaggor: Påverkas ej.

Beskrivning: Testar värdet hos Booleska uttrycket  $(N \oplus V) + Z$ . Om  $(N \oplus V) + Z = 1$  utförs ett hopp till adressen Adr = PC+Offset. Offset räknas från adressen efter branchinstruktionen, dvs vid uträkningen av hoppadressen pekar PC på operationskoden direkt efter branchinstruktionen i minnet. Om  $(N \oplus V) + Z = 0$  utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.

**BLO Branch if Lower (= BCS)**

**BLS Branch on Lower or Same**

Instruktion: BLS Adr

RTN: If  $C+Z = 1$ :  $PC+Offset \rightarrow PC$ 

Flaggor: Påverkas ej.

Beskrivning: Testar värdet hos Booleska uttrycket  $C+Z$ . Om  $C+Z = 1$  utförs ett hopp till adressen  $Adr = PC+Offset$ . Offset räknas från adressen efter branchinstruktionen, dvs vid uträkningen av hoppadressen pekar PC på operationskoden direkt efter branchinstruktionen i minnet. Om  $C+Z = 0$  utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.

**BLT Branch on Less than Zero**

Instruktion: BLT Adr

RTN: If  $N \oplus V = 1$ :  $PC+Offset \rightarrow PC$ 

Flaggor: Påverkas ej.

Beskrivning: Testar värdet hos Booleska uttrycket  $N \oplus V$ . Om  $N \oplus V = 1$  utförs ett hopp till adressen  $Adr = PC+Offset$ . Offset räknas från adressen efter branchinstruktionen, dvs vid uträkningen av hoppadressen pekar PC på operationskoden direkt efter branchinstruktionen i minnet. Om  $N \oplus V = 0$  utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.

**BMI Branch on Minus**

Instruktion: BMI Adr

RTN: If  $N = 1$ :  $PC+Offset \rightarrow PC$ 

Flaggor: Påverkas ej.

Beskrivning: Testar N-flaggans värde. Om  $N=1$  utförs ett hopp till adressen  $Adr = PC+Offset$ . Offset räknas från adressen efter branchinstruktionen, dvs vid uträkningen av hoppadressen pekar PC på operationskoden direkt efter branchinstruktionen i minnet. Om  $N=0$  utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.

**BNE Branch Not Equal**

Instruktion: BNE Adr

RTN: If  $Z = 0$ :  $PC+Offset \rightarrow PC$ 

Flaggor: Påverkas ej.

Beskrivning: Testar Z-flaggans värde. Om  $Z=0$  utförs ett hopp till adressen  $Adr = PC+Offset$ . Offset räknas från adressen efter branchinstruktionen, dvs vid uträkningen av hoppadressen pekar PC på operationskoden direkt efter branchinstruktionen i minnet. Om  $Z=1$  utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.

**BPL Branch on Plus**

Instruktion: BPL Adr

RTN: If  $N = 0$ :  $PC+Offset \rightarrow PC$ 

Flaggor: Påverkas ej.

Beskrivning: Testar N-flaggans värde. Om  $N=0$  utförs ett hopp till adressen  $Adr = PC+Offset$ . Offset räknas från adressen efter branchinstruktionen, dvs vid uträkningen av hoppadressen pekar PC på operationskoden direkt efter branchinstruktionen i minnet. Om  $N=1$  utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.



**BRA Branch Always**

Instruktion: BRA Adr

RTN: PC+Offset → PC

Flaggor: Påverkas ej.

Beskrivning: Ett hopp utförs till adressen Adr = PC+Offset. Offset räknas från adressen efter branchinstruktionen, dvs vid uträkningen av hoppadressen pekar PC på operationskoden som (eventuellt) finns direkt efter branchinstruktionen i minnet.

**BSR Branch to Subroutine**

Instruktion: BSR Adr

RTN: SP-1 → SP  
PC → M(SP)  
PC+Offset → PC

Flaggor: Påverkas ej.

Beskrivning: PC-värdet (återhopsadressen) skrivs först på stacken. Ett hopp utförs sedan till adressen Adr = PC+Offset. Offset räknas från adressen efter BSR-instruktionen, dvs vid uträkningen av hoppadressen pekar PC på operationskoden direkt efter BSR-instruktionen (= återhopsadressen).

**BVC Branch on Overflow Clear**

Instruktion: BVC Adr

RTN: If V = 0: PC+Offset → PC

Flaggor: Påverkas ej.

Beskrivning: Testar V-flaggans värde. Om V=0 utförs ett hopp till adressen Adr = PC+Offset. Offset räknas från adressen efter branchinstruktionen, dvs vid uträkningen av hoppadressen pekar PC på operationskoden direkt efter branchinstruktionen i minnet. Om V=1 utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.

**BVS Branch on Overflow Set**

Instruktion: BVS Adr

RTN: If V = 1: PC+Offset → PC

Flaggor: Påverkas ej.

Beskrivning: Testar V-flaggans värde. Om V=1 utförs ett hopp till adressen Adr = PC+Offset. Offset räknas från adressen efter branchinstruktionen, dvs vid uträkningen av hoppadressen pekar PC på operationskoden direkt efter branchinstruktionen i minnet. Om V=0 utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.

**CLR Clear**

Varianter: CLRA; CLRB; CLR Adr; CLR ,X

RTN: 00H → R, eller 00H → M där R = A eller B.

Flaggor: N: Nollställs.  
Z: Ettställs.  
V: ?  
C: Nollställs.

Beskrivning: Reg A, Reg B eller innehållet på aktuell minnesadress nollställs.

**CMP Compare Memory and Register**

Varianter: CMPA Adr; CMPB Adr; CPX Adr; CPS Adr  
 CMPA #Data; CMPB #Data; CPX #Data; CPS #Data; CMPA ,X; CMPB ,X

RTN:  $R - M$  eller  $R - \text{Data}$  där  $R = A, B, X$  eller SP.

Flaggor: N: Får värdet hos skillnadens teckenbit (bit 7).  
 Z: Ettställs om skillnaden blir noll.  
 V: Ettställs om 2-komplementoverflow uppstår vid subtraktionen  
 C: Ettställs om borrow uppstår vid subtraktionen.

Beskrivning: Dataordet från minnet eller från instruktionen subtraheras från innehållet i det angivna registret. Skillnaden lagras ej, utan påverkar endast flaggorna.

**COM Complement**

Varianter: COMA; COMB; COM Adr; COM ,X

RTN:  $R_{1k} \rightarrow R$  eller  $M_{1k} \rightarrow M$

Flaggor: N: Ettställs om resultatets teckenbit (bit 7) får värdet 1.  
 Z: Ettställs om samtliga åtta bitar i resultatet blir noll.  
 V: Nollställs.  
 C: ?

**DEC Decrement**

Varianter: DECA; DECB; DEX; DEC Adr; DEC ,X; DEC ,SP

RTN:  $R - 1 \rightarrow R$  eller  $M - 1 \rightarrow M$

Flaggor: N: Ettställs om resultatets teckenbit (bit 7) får värdet 1.  
 Z: Ettställs om samtliga åtta bitar i resultatet blir noll.  
 V: Ettställs om 2-komplementoverflow uppstår.  
 C: Ettställs om borrow uppstår.

**EOR Exclusive-OR**

Varianter: EORA Adr; EORB Adr; EORA #Data; EORB #Data; EORA ,X; EORB ,X

RTN:  $R \text{ XOR } M \rightarrow R$ , där  $R = A$  eller  $B$ ,  $M = \text{data från minneasadress eller instruktionen själv}$ .

Flaggor: N: Ettställs om resultatets teckenbit (bit 7) får värdet 1.  
 Z: Ettställs om samtliga åtta bitar i resultatet blir noll.  
 V: Nollställs.  
 C: Nollställs.

Beskrivning: Utför bitvis XOR-operation mellan dataordet i minnet och innehållet i reg A eller reg B. Resultatet placeras i reg A eller reg B.

**EXG Exchange Contents between Registers**

Varianter: EXG A,B; EXG A,CCR; EXG B,CCR; EXG X,SP

RTN:  $R1 \leftrightarrow R2$

Flaggor: Påverkas endast om CC-registret är det ena registret som används.

Beskrivning: Data växlas mellan angivna register.

**INC Increment**

Varianter: INCA; INCB; INX; INC Adr; INC ,X; INC ,SP

RTN:  $R + 1 \rightarrow R$  eller  $M + 1 \rightarrow M$

Flaggor: N: Ettställs om resultatets teckenbit (bit 7) får värdet 1.  
 Z: Ettställs om samtliga åtta bitar i resultatet blir noll.  
 V: Ettställs om 2-komplementoverflow uppstår.  
 C: Ettställs om summan ej ryms i åtta bitar, dvs blir lika med 256. I detta fall ettställs även Z.

**JMP Jump**

Varianter: JMP Adr; JMP ,X; JMP [Adr]; JMP [X]; JMP [n,X]; JMP [A,X]; JMP [B,X]; JMP [n,PCR]

RTN:  $EA \rightarrow PC$  (EA= Effektiva adressen. För hoppinstruktioner är EA adressen dit hoppet skall ske.)

Flaggor: Påverkas ej.

Beskrivning: Ett hopp utförs till adressen EA, dvs EA laddas i PC.

**JSR Jump to Subroutine**

Varianter: JSR Adr; JSR ,X; JSR [Adr]; JSR [X]; JSR [n,X]; JSR [A,X]; JSR [B,X]; JSR [n,PCR]

RTN:  $SP-1 \rightarrow SP$   
 $PC \rightarrow M(SP)$   
 $EA \rightarrow PC$  (EA= Effektiva adressen. För hoppinstruktioner är EA adressen dit hoppet skall ske.)

Flaggor: Påverkas ej.

Beskrivning: PC-värdet, som är adressen till instruktionen efter JSR-instruktionen, dvs återhoppadressen, skrivs först på stacken. Ett hopp utförs sedan till adressen EA, dvs EA laddas i PC.

**LD Load**

Varianter: LDAA Adr; LDAB Adr; LDX Adr; LDS Adr;  
 LDAA #Data; LDAB #Data; LDX #Data; LDS #Data;  
 LDAA ,X; LDAA 1,X+; LDAA 1,-X; LDAA n,X; LDAA A,X; LDAA B,X;  
 LDAB ,X; LDAB 1,X+; LDAB 1,-X; LDAB n,X; LDAB A,X; LDAB B,X;  
 LDX A,X; LDX B,X;  
 LDAA n,PCR; LDAB n,PCR;  
 LDAA [Adr]; LDAA [X]; LDAA [n,X]; LDAA [A,X]; LDAA [B,X]; LDAA [n,PCR];  
 LDAB [Adr]; LDAB [X]; LDAB [n,X]; LDAB [A,X]; LDAB [B,X]; LDAB [n,PCR]

RTN:  $M \rightarrow R$  eller  $Data \rightarrow R$

Flaggor: Påverkas ej.

Beskrivning: Laddar dataord från minnet eller instruktionen till angivet register.

**LEA Load Effective Address**

Varianter: LEAX 1,-X; LEAX 1,X+; LEAX n,X; LEAX A,X; LEAS n,SP

RTN:  $EA \rightarrow X$  eller  $EA \rightarrow SP$

Flaggor: Påverkas ej.

Beskrivning: Laddar effektiva adressen i reg X eller reg SP. Används för att manipulera X- eller SP-registrets innehåll.

**NEG Negate**

Varianter: NEGA; NEGB; NEG Adr; NEG ,X

RTN:  $0 - R \rightarrow R$  eller  $0 - M \rightarrow M$

Flaggor: N: Ettställs om resultatets teckenbit (bit 7) får värdet 1.  
 Z: Ettställs om samtliga åtta bitar i resultatet blir noll, dvs om det gamla värdet är noll.  
 V: Ettställs om 2-komplementoverflow uppstår.  
 C: Ettställs om det gamla värdet  $\neq 0$ .

Beskrivning: 2-komplementerar innehållet i angivet register eller minnesinnehåll.

**NOP No operation**

Instruktion: NOP

RTN: Inget händer.

Flaggor: Påverkas ej.

Beskrivning: Instruktionen utför ingenting.

**OR Logical OR Memory Data into Register**

Varianter: ORAA Adr; ORAB Adr; ORAA #Data; ORAB #Data; ORAA ,X; ORAB ,X

RTN:  $R \text{ OR } M \rightarrow R$ , där  $R = A$  eller  $B$ ,  $M =$  data från minnesadress eller instruktionen själv.

Flaggor: N: Ettställs om resultatets teckenbit (bit 7) får värdet 1.  
 Z: Ettställs om samtliga åtta bitar i resultatet blir noll.  
 V: Nollställs.  
 C: Nollställs.

Beskrivning: Utför bitvis OR-operation mellan dataordet i minnet och innehållet i reg A eller reg B. Resultatet placeras i reg A eller reg B.

**ORCC Logical OR Data into Condition Code Register**

Instruktion: ORCC #Data

RTN:  $CCR \text{ OR } \text{Data} \rightarrow CCR$

Flaggor: Flaggorna ettställs i de positioner där CCR eller Data innehåller någon etta.

Beskrivning: Utför bitvis OR-operation mellan innehållet i flaggregistret (CCR) och dataordet. Resultatet placeras i flaggregistret.

**PSH Push Register on the Stack**

Varianter: PSHA; PS HB; PSHC; PSHX

RTN:  $SP-1 \rightarrow SP$   
 $R \rightarrow M(SP)$

Flaggor: Påverkas ej.

Beskrivning: Stackpekaren uppdateras först. Angivet registerinnehåll skrivs sedan på stacken.

**PUL Pull Register from the Stack**

Varianter: PULA; PULB; PULC; PULX

RTN: M(SP) → R  
SP+1 → SP

Flaggor: Flaggorna påverkas endast vid PULC, då flaggorna får värden från stacken.

Beskrivning: Översta dataordet på stacken läses och placeras i angivet register. Stackpekaren uppdateras sedan.

**ROL Rotate Left**

Varianter: ROLA; ROLB; ROL Adr; ROL ,X

RTN:  $2R + C \rightarrow R$  eller  $2M + C \rightarrow M$

Flaggor: N: Ettställs om resultatets teckenbit (bit 7) får värdet 1.  
Z: Ettställs om samtliga åtta bitar i resultatet blir noll.  
V: Ettställs om overflow vid 2-komplementsrepresentation inträffar.  
C: Teckenbiten (bit 7) före skiftet blir ny carrybit efter skiftet.

Beskrivning: Multipliserar ett tal med inbyggt tecken i reg A, reg B eller minnet med 2 och adderar dessutom det gamla värdet på C-flaggan i den minst signifikanta positionen.

**RTS Return from Subroutine**

Instruktion: RTS

RTN: M(SP) → PC  
SP+1 → SP

Flaggor: Påverkas ej.

Beskrivning: Återhopp från en subrutin utförs genom att översta dataordet på stacken, dvs återhopsadressen, läses och placeras i PC. Stackpekaren uppdateras sedan.

**SBC Subtract with Borrow Memory Data from Register**

Varianter: SBCA Adr; SBCB Adr; SBCA #Data; SBCB #Data; SBCA ,X; SBCB ,X

RTN:  $R - M - C \rightarrow R$ , där R = A eller B, M = data från minneasadress eller instruktionen själv och C = carryflaggan (här borrow) i flaggregistret (CCR).

Flaggor: N: Ettställs om resultatets teckenbit (bit 7) får värdet 1 vid subtraktionen.  
Z: Ettställs om samtliga åtta bitar i resultatet blir noll vid subtraktionen.  
V: Ettställs om overflow vid 2-komplementsrepresentation inträffar vid subtraktionen.  
C: Ettställs om en lånesiffra = 1 uppstår från bitpositionen längst till vänster vid subtraktionen.

Beskrivning: Utför åttabitars subtraktion av dataordet i minnet från innehållet i reg A eller reg B. Resultatet placeras i reg A eller reg B. En eventuell lånebit (borrow) som uppstår vid subtraktionen placeras i C-biten (C-flaggan) i CC-registret. Det gamla värdet på C-biten i flaggregistret används som lånesiffra i minst signifikant position (lånesiffra in) vid subtraktionen.

C-flaggan representerar i detta fall en lånesiffra vid subtraktion och sätts till inversen av det värde som kommer ut från ALU:n när subtraktionen  $R - M$  utförs på det traditionella sättet  $R + M_{1k} + 1$ .

**ST Store**

Varianter: STAA Adr; STAB Adr; STX Adr; STS Adr;  
 STAA ,X; STAA 1,X+; STAA 1,-X; STAA n,X; STAA A,X; STAA B,X;  
 STAB ,X; STAB 1,X+; STAB 1,-X; STAB n,X; STAB A,X; STAB B,X;  
 STAA n,PCR; STAB n,PCR;  
 STAA [Adr]; STAA [,X]; STAA [n,X]; STAA [A,X]; STAA [B,X]; STAA [n,PCR];  
 STAB [Adr]; STAB [,X]; STAB [n,X]; STAB [A,X]; STAB [B,X]; STAB [n,PCR]

RTN: R → M

Flaggor: Påverkas ej.

Beskrivning: Lagrar angivet registerinnehåll i minnet på den effektiva adressen.

**SUB Subtract Memory Data from Register**

Varianter: SUBA Adr; SUBB Adr; SUBA #Data; SUBB #Data; SUBA ,X; SUBB ,X

RTN:  $R - M \rightarrow R$ , där R = A eller B, M = data från minneasadress eller instruktionen själv.

Flaggor: N: Ettställs om resultatets teckenbit (bit 7) får värdet 1 vid subtraktionen.  
 Z: Ettställs om samtliga åtta bitar i resultatet blir noll vid subtraktionen.  
 V: Ettställs om overflow vid 2-komplementsrepresentation inträffar vid subtraktionen.  
 C: Ettställs om en lånesiffra = 1 uppstår från bitpositionen längst till vänster vid subtraktionen.

Beskrivning: Utför åttabitars subtraktion av dataordet i minnet från innehållet i reg A eller reg B. Resultatet placeras i reg A eller reg B. En eventuell lånebit (borrow) som uppstår vid subtraktionen placeras i C-biten (C-flaggan) i CC-registret.

C-flaggan representerar i detta fall en lånesiffra vid subtraktion och sätts till inversen av det värde som kommer ut från ALU:n när subtraktionen  $R - M$  utförs på det traditionella sättet  $R + M_{1k} + 1$ .

**TFR Transfer Register to Register**

Varianter: TFR A,B; TFR B,A; TFR A,CCR; TFR CCR,A; TFR X,SP; TFR SP,X

RTN: R1 → R2

Flaggor: Påverkas ej såvida man inte flyttar ett registerinnehåll till CC-registret.

Beskrivning: Data kopieras mellan angivna register.

**TST Test**

Varianter: TSTA; TSTB; TST Adr; TST ,X

RTN:  $R - 0$  eller  $M - 0$  där R = A eller B. M = data från minnesadress.

Flaggor: N: Ettställs om resultatets teckenbit (bit 7) får värdet 1.  
 Z: Ettställs om samtliga åtta bitar i resultatet blir noll.  
 V: Nollställs.  
 C: Nollställs.

Beskrivning: Låter datavärdet i R eller M passera ALU:n och sätter flaggvipporna N och Z så att man kan avgöra datavärdets tecken eller om det är noll. Endast flaggvipporna påverkas.

### Tabell med samtliga instruktioner i alfabetisk ordning

OP	#	Beteckning	OP	#	Beteckning	OP	#	Beteckning	OP	#	Beteckning	OP	#	Beteckning
2C	2	ADCA Adr	DB	1	CLR ,X	69	2	JSR Adr	77	1	LEAX A,X	B3	2	STAA [n,X]
2E	2	ADCA #Data	4A	2	CMPA Adr	9C	1	JSR A,X	78	1	LEAX B,X	B1	1	STAA [,X]
D0	1	ADCA ,X	4E	2	CMPA #Data	9D	1	JSR B,X	76	2	LEAX n,X	14	2	STAB Adr
2D	2	ADCB Adr	DC	1	CMPA ,X	9B	2	JSR n,X	75	1	LEAX 1,X+	92	1	STAB A,X
2F	2	ADCB #Data	4B	2	CMPB Adr	9A	1	JSR ,X	74	1	LEAX 1,-X	94	1	STAB B,X
D1	1	ADCB ,X	4F	2	CMPB #Data	C1	2	JSR [Adr]	38	1	NEGA	A2	2	STAB n,PCR
28	2	ADDA Adr	DD	1	CMPB ,X	C4	1	JSR [A,X]	39	1	NEGB	90	2	STAB n,X
2A	2	ADDA #Data	4D	2	CPS Adr	C5	1	JSR [B,X]	3A	2	NEG Adr	8A	1	STAB ,X
CE	1	ADDA ,X	51	2	CPS #Data	C6	2	JSR [n,PCR]	D6	1	NEG ,X	8C	1	STAB 1,X+
29	2	ADDB Adr	4C	2	CPX Adr	C3	2	JSR [n,X]	00	1	NOP	8E	1	STAB 1,-X
2B	2	ADDB #Data	50	2	CPX #Data	C2	1	JSR [,X]	1B	2	ORAA Adr	B0	2	STAB [Adr]
CF	1	ADDB ,X	23	1	COMA	0B	2	LDAA Adr	1D	2	ORAA #Data	B6	1	STAB [A,X]
17	2	ANDA Adr	24	1	COMB	83	1	LDAA A,X	C9	1	ORAA ,X	B8	1	STAB [B,X]
19	2	ANDA #Data	25	2	COM Adr	85	1	LDAA B,X	1C	2	ORAB Adr	BA	2	STAB [n,PCR]
C7	1	ANDA ,X	CD	1	COM ,X	9F	2	LDAA n,PCR	1E	2	ORAB #Data	B4	2	STAB [n,X]
18	2	ANDB Adr	44	1	DECA	81	2	LDAA n,X	CA	1	ORAB ,X	B2	1	STAB [,X]
1A	2	ANDB #Data	45	1	DECB	0F	2	LDAA #Data	27	2	ORCC #Data	16	2	STS Adr
C8	1	ANDB ,X	46	2	DEC Adr	79	1	LDAA ,X	6C	1	PSHA	15	2	STX Adr
26	2	ANDCC #Data	E5	1	DEC ,SP	7B	1	LDAA 1,X+	6D	1	PSHB	30	2	SUBA Adr
3B	1	ASLA	DA	1	DEC ,X	7D	1	LDAA 1,-X	6E	1	PSHC	32	2	SUBA #Data
3C	1	ASLB	E2	1	DEX	A3	2	LDAA [Adr]	6F	1	PSHX	D2	1	SUBA ,X
3D	2	ASL Adr	1F	2	EORA Adr	A9	1	LDAA [A,X]	70	1	PULA	31	2	SUBB Adr
D7	1	ASL ,X	21	2	EORA #Data	AB	1	LDAA [B,X]	71	1	PULB	33	2	SUBB #Data
62	2	BCC Adr	CB	1	EORA ,X	AD	2	LDAA [n,PCR]	72	1	PULC	D3	1	SUBB ,X
61	2	BCS Adr	20	2	EORB Adr	A7	2	LDAA [n,X]	73	1	PULX	01	1	TFR A,B
5D	2	BEQ Adr	22	2	EORB #Data	A5	1	LDAA [,X]	3E	1	ROLA	03	1	TFR A,CCR
66	2	BGE Adr	CC	1	EORB ,X	0C	2	LDAB Adr	3F	1	ROLB	02	1	TFR B,A
65	2	BGT Adr	07	1	EXG A,B	84	1	LDAB A,X	40	2	ROL Adr	04	1	TFR CCR,A
63	2	BHI Adr	08	1	EXG A,CCR	86	1	LDAB B,X	D8	1	ROL ,X	06	1	TFR SP,X
55	2	BITA Adr	09	1	EXG B,CCR	A0	2	LDAB n,PCR	6A	1	RTS	05	1	TFR X,SP
57	2	BITA #Data	0A	1	EXG X,SP	82	2	LDAB n,X	34	2	SBCA Adr	52	1	TSTA
DF	1	BITA ,X	41	1	INCA	10	2	LDAB #Data	36	2	SBCA #Data	53	1	TSTB
56	2	BITB Adr	42	1	INCB	7A	1	LDAB ,X	D4	1	SBCA ,X	54	2	TST Adr
58	2	BITB #Data	43	2	INC Adr	7C	1	LDAB 1,X+	35	2	SBCB Adr	DE	1	TST ,X
E0	1	BITB ,X	E4	1	INC ,SP	80	1	LDAB 1,-X	37	2	SBCB #Data			-
67	2	BLE Adr	D9	1	INC ,X	A4	2	LDAB [Adr]	D5	1	SBCB ,X			-
64	2	BLS Adr	E1	1	INX	AA	1	LDAB [A,X]	13	2	STAA Adr			-
68	2	BLT Adr	59	2	JMP Adr	AC	1	LDAB [B,X]	91	1	STAA A,X			-
5B	2	BMI Adr	97	1	JMP A,X	AE	2	LDAB [n,PCR]	93	1	STAA B,X			-
5E	2	BNE Adr	98	1	JMP B,X	A8	2	LDAB [n,X]	A1	2	STAA n,PCR			-
5C	2	BPL Adr	96	2	JMP n,X	A6	1	LDAB [,X]	8F	2	STAA n,X			-
5A	2	BRA Adr	95	1	JMP ,X	0E	2	LDS Adr	89	1	STAA ,X			-
6B	2	BSR Adr	BB	2	JMP [Adr]	12	2	LDS #Data	8B	1	STAA 1,X+			-
60	2	BVC Adr	BE	1	JMP [A,X]	0D	2	LDX Adr	8D	1	STAA 1,-X			-
5F	2	BVS Adr	BF	1	JMP [B,X]	87	1	LDX A,X	AF	2	STAA [Adr]			-
47	1	CLRA	C0	2	JMP [n,PCR]	88	1	LDX B,X	B5	1	STAA [A,X]			-
48	1	CLRB	BD	2	JMP [n,X]	11	2	LDX #Data	B7	1	STAA [B,X]			-
49	2	CLR Adr	BC	1	JMP [,X]	E3	1	LEAS n,SP	B9	2	STAA [n,PCR]			-

### Tabell med samtliga instruktioner ordnade efter operationskod

OP	#	Beteckning	OP	#	Beteckning	OP	#	Beteckning	OP	#	Beteckning	OP	#	Beteckning
00	1	NOP	30	2	SUBA Adr	60	2	BVC Adr	90	2	STAB n,X	C0	2	JMP [n,PCR]
01	1	TFR A,B	31	2	SUBB Adr	61	2	BCS Adr	91	1	STAA A,X	C1	2	JSR [Adr]
02	1	TFR B,A	32	2	SUBA #Data	62	2	BCC Adr	92	1	STAB A,X	C2	1	JSR [,X]
03	1	TFR A,CCR	33	2	SUBB #Data	63	2	BHI Adr	93	1	STAA B,X	C3	2	JSR [n,X]
04	1	TFR CCR,A	34	2	SBCA Adr	64	2	BLS Adr	94	1	STAB B,X	C4	1	JSR [A,X]
05	1	TFR X,SP	35	2	SBCB Adr	65	2	BGT Adr	95	1	JMP ,X	C5	1	JSR [B,X]
06	1	TFR SP,X	36	2	SBCA #Data	66	2	BGE Adr	96	2	JMP n,X	C6	2	JSR [n,PCR]
07	1	EXG A,B	37	2	SBCB #Data	67	2	BLE Adr	97	1	JMP A,X	C7	1	ANDA ,X
08	1	EXG A,CCR	38	1	NEGA	68	2	BLT Adr	98	1	JMP B,X	C8	1	ANDB ,X
09	1	EXG B,CCR	39	1	NEGB	69	2	JSR Adr	99		-	C9	1	ORAA ,X
0A	1	EXG X,SP	3A	2	NEG Adr	6A	1	RTS	9A	1	JSR ,X	CA	1	ORAB ,X
0B	2	LDAA Adr	3B	1	ASLA	6B	2	BSR Adr	9B	2	JSR n,X	CB	1	EORA ,X
0C	2	LDAB Adr	3C	1	ASLB	6C	1	PSHA	9C	1	JSR A,X	CC	1	EORB ,X
0D	2	LDX Adr	3D	2	ASL Adr	6D	1	PSHB	9D	1	JSR B,X	CD	1	COM ,X
0E	2	LDS Adr	3E	1	ROLA	6E	1	PSHC	9E		-	CE	1	ADDA ,X
0F	2	LDAA #Data	3F	1	ROLB	6F	1	PSHX	9F	2	LDAA n,PCR	CF	1	ADDB ,X
10	2	LDAB #Data	40	2	ROL Adr	70	1	PULA	A0	2	LDAB n,PCR	D0	1	ADCA ,X
11	2	LDX #Data	41	1	INCA	71	1	PULB	A1	2	STAA n,PCR	D1	1	ADCB ,X
12	2	LDS #Data	42	1	INCB	72	1	PULC	A2	2	STAB n,PCR	D2	1	SUBA ,X
13	2	STAA Adr	43	2	INC Adr	73	1	PULX	A3	2	LDAA [Adr]	D3	1	SUBB ,X
14	2	STAB Adr	44	1	DECA	74	1	LEAX 1,-X	A4	2	LDAB [Adr]	D4	1	SBCA ,X
15	2	STX Adr	45	1	DECB	75	1	LEAX 1,X+	A5	1	LDAA [,X]	D5	1	SBCB ,X
16	2	STS Adr	46	2	DEC Adr	76	2	LEAX n,X	A6	1	LDAB [,X]	D6	1	NEG ,X
17	2	ANDA Adr	47	1	CLRA	77	1	LEAX A,X	A7	2	LDAA [n,X]	D7	1	ASL ,X
18	2	ANDB Adr	48	1	CLRB	78	1	LEAX B,X	A8	2	LDAB [n,X]	D8	1	ROL ,X
19	2	ANDA #Data	49	2	CLR Adr	79	1	LDAA ,X	A9	1	LDAA [A,X]	D9	1	INC ,X
1A	2	ANDB #Data	4A	2	CMPA Adr	7A	1	LDAB ,X	AA	1	LDAB [A,X]	DA	1	DEC ,X
1B	2	ORAA Adr	4B	2	CMPB Adr	7B	1	LDAA 1,X+	AB	1	LDAA [B,X]	DB	1	CLR ,X
1C	2	ORAB Adr	4C	2	CPX Adr	7C	1	LDAB 1,X+	AC	1	LDAB [B,X]	DC	1	CMPA ,X
1D	2	ORAA #Data	4D	2	CPS Adr	7D	1	LDAA 1,-X	AD	2	LDAA [n,PCR]	DD	1	CMPB ,X
1E	2	ORAB #Data	4E	2	CMPA #Data	7E		-	AE	2	LDAB [n,PCR]	DE	1	TST ,X
1F	2	EORA Adr	4F	2	CMPB #Data	7F		-	AF	2	STAA [Adr]	DF	1	BITA ,X
20	2	EORB Adr	50	2	CPX #Data	80	1	LDAB 1,-X	B0	2	STAB [Adr]	E0	1	BITB ,X
21	2	EORA #Data	51	2	CPS #Data	81	2	LDAA n,X	B1	1	STAA [,X]	E1	1	INX
22	2	EORB #Data	52	1	TSTA	82	2	LDAB n,X	B2	1	STAB [,X]	E2	1	DEX
23	1	COMA	53	1	TSTB	83	1	LDAA A,X	B3	2	STAA [n,X]	E3	2	LEAS n,SP
24	1	COMB	54	2	TST Adr	84	1	LDAB A,X	B4	2	STAB [n,X]	E4	1	INC ,SP
25	2	COM Adr	55	2	BITA Adr	85	1	LDAA B,X	B5	1	STAA [A,X]	E5	1	DEC ,SP
26	2	ANDCC #Data	56	2	BITB Adr	86	1	LDAB B,X	B6	1	STAB [A,X]	E6		-
27	2	ORCC #Data	57	2	BITA #Data	87	1	LDX A,X	B7	1	STAA [B,X]	E7		-
28	2	ADDA Adr	58	2	BITB #Data	88	1	LDX B,X	B8	1	STAB [B,X]	E8		-
29	2	ADDB Adr	59	2	JMP Adr	89	1	STAA ,X	B9	2	STAA [n,PCR]	E9		-
2A	2	ADDA #Data	5A	2	BRA Adr	8A	1	STAB ,X	BA	2	STAB [n,PCR]	EA		-
2B	2	ADDB #Data	5B	2	BMI Adr	8B	1	STAA 1,X+	BB	2	JMP [Adr]	EB		-
2C	2	ADCA Adr	5C	2	BPL Adr	8C	1	STAB 1,X+	BC	1	JMP [,X]	EC		-
2D	2	ADCB Adr	5D	2	BEQ Adr	8D	1	STAA 1,-X	BD	2	JMP [n,X]	ED		-
2E	2	ADCA #Data	5E	2	BNE Adr	8E	1	STAB 1,-X	BE	1	JMP [A,X]	EE		-
2F	2	ADCB #Data	5F	2	BVS Adr	8F	2	STAA n,X	BF	1	JMP [B,X]	EF		-