

FLEX

Instruktionslista

Innehåll

1	Förklaring av beteckningar i instruktionslistan	2
2	Detaljerad beskrivning av FLEX-processorns instruktioner	3
3	Operationskoder, maskincykler och flaggpåverkan	13
3.1	Enkel dataflyttning	13
3.2	Logik	14
3.3	Aritmetik	15
3.4	Diverse	16
3.5	Test och jämförelse	16
3.6	Ovillkorlig programflödesändring	16
3.7	Villkorlig programflödesändring	17
3.8	Manipulering av X- och S-registrets innehåll	17
3.9	Dataflyttning med adressering via X-registret	18
3.10	Logik, Aritmetik och Test med adressering via X- och S-registret	19
3.11	Hopp med adressering via X-registret	20
3.12	Hopp till subrutin och återhopp från subrutin	20
3.13	Branch till subrutin	20
3.14	Lagring av data på stack och hämtning av data från stack	21
3.15	Hopp till subrutin med adressering via X-registret eller PC	21
3.16	*Tillägg till operationsbeskrivningar	21
4	Tabell med samtliga instruktioner ordnade efter operationskod	22

1 Förklaring av beteckningar i instruktionslistan

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.

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' Innehåll i register A komplementeras (inverteras) bitvis.
M(Adr) Minnesinnehåll på adressen Adr.
M'(Adr) Minnesinnehåll komplementeras (inverteras) bitvis.
N Teckenflaggan ("Negative").
Z Nollflaggan ("Zero")
V Overflowflaggan.
C Carryflaggan.
a "Affected". (Används för att visa att en flagga påverkas av en operation).
• "Not affected". (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.

2 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 åttabits 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 åttabits 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: $2R \rightarrow R$ eller $2M \rightarrow M$

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: 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 $\rightarrow$ 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 $\rightarrow$ 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 $\rightarrow$ 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 $\rightarrow$ 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 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 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.

FLEX INSTRUKTIONSLISTA	
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 hopp-adressen 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 operations-koden 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 operations-koden 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.	

FLEX INSTRUKTIONSLISTA	
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 – 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' → R eller M' → 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 → R eller M – 1 → 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 XOR M → 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 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 ↔ 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 rymms i åtta bitar, dvs blir lika med 256. I detta fall ettställs även Z.

JMP Jump

Varianter: JMP Adr; JMP ,X

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

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;

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,S

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

Flaggor: Påverkas ej.

Beskrivning: Laddar effektiva adressen i reg X eller reg S. Används för att manipulera X- eller S-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 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 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: $CCR \text{ OR } Data \rightarrow CCR$

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; PSBH; PSHC; PSHX

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

Flaggor: Påverkas ej.

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

FLEX INSTRUKTIONSLISTA	
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 → R eller 2M +C → 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 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 → 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' + 1.

FLEX INSTRUKTIONSLISTA	
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;
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 → 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' + 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.

3 Operationskoder, maskinnycklar och flaggpåverkan

3.1 Enkel dataflyttning

Instruktion		Adressering						Operations- beskrivning*		Flaggor				
Operation	Beteckning	Inhere nt	Immediat e			Absolut e				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	a	a	a	a	
	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	a	a	a	a	
	EXG B,CCR	09	1	5					B ↔ CCR	a	a	a	a	
	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)	•	•	•	•	

3.2 Logik

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

3.3 Aritmetik

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

3.4 Diverse

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

3.5 Test och jämförelse

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

3.6 Ovillkorlig programflödesändring

Instruktion		Adressering									Operationsbeskrivning*	Flaggor						
Operation	Beteckning	Inherent			Immediate			Absolute					3	2	1	0		
		OP	#	~	OP	#	~	OP	#	~			N	Z	V	C		
Unconditional jump	JMP Adr							59	2	4	Adr → PC	.	.	.	.			
Unconditional branch	BRA Adr	5A			2			5			PC+Offs → PC				.	.	.	.

3.7 Villkorlig programflödesändring

Instruktion		Adressering			Operationsbeskrivning*	Flaggor			
Operation	Beteckning	Relative				3	2	1	0
		OP	#	~		N	Z	V	C
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	•	•	•	•

3.8 Manipulering av X- och S-registrets innehåll

Instruktion		Adressering			Operations- beskrivning*	Flaggor			
Operation	Beteckning	Via X				3	2	1	0
		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	$S + n \rightarrow S$	•	•	•	•

3.9 Dataflyttning med adressering via X-registret

Instruktion		Adressering			Operations- beskrivning*	Flaggor			
Operation	Beteckning	Via X				3 N	2 Z	1 V	0 C
		OP	#	~					
Load	LDAA ,X	79	1	4	M(X) → A	•	•	•	•
	LDAB ,X	7A	1	4	M(X) → B	•	•	•	•
	LDAA 1,X+	7B	1	5	M(X) → A X+1 → X	•	•	•	•
	LDAB 1,X+	7C	1	5	M(X) → B X+1 → X	•	•	•	•
	LDAA 1,-X	7D	1	5	X-1 → X M(X) → A	•	•	•	•
	LDAB 1,-X	80	1	5	X-1 → X M(X) → B	•	•	•	•
	LDAA n,X	81	2	7	M(n+X) → A	•	•	•	•
	LDAB n,X	82	2	7	M(n+X) → B	•	•	•	•
	LDAA A,X	83	1	6	M(A+X) → A	•	•	•	•
	LDAB A,X	84	1	6	M(A+X) → B	•	•	•	•
	LDAA B,X	85	1	6	M(B+X) → A	•	•	•	•
	LDAB B,X	86	1	6	M(B+X) → B	•	•	•	•
	LDX A,X	87	1	6	M(A+X) → X	•	•	•	•
	LDX B,X	88	1	6	M(B+X) → X	•	•	•	•
Store	STAA ,X	89	1	4	A → M(X)	•	•	•	•
	STAB ,X	8A	1	4	B → M(X)	•	•	•	•
	STAA 1,X+	8B	1	5	A → M(X) X+1 → X	•	•	•	•
	STAB 1,X+	8C	1	5	B → M(X) X+1 → X	•	•	•	•
	STAA 1,-X	8D	1	5	X-1 → X A → M(X)	•	•	•	•
	STAB 1,-X	8E	1	5	X-1 → X B → M(X)	•	•	•	•
	STAA n,X	8F	2	7	A → M(n+X)	•	•	•	•
	STAB n,X	90	2	7	B → M(n+X)	•	•	•	•
	STAA A,X	91	1	6	A → M(A+X)	•	•	•	•
	STAB A,X	92	1	6	B → M(A+X)	•	•	•	•
	STAA B,X	93	1	6	A → M(B+X)	•	•	•	•
	STAB B,X	94	1	6	B → M(B+X)	•	•	•	•

3.10 Logik, Aritmetik och Test med adressering via X- och S-registret

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

3.11 Hopp med adressering via X-registret

Instruktion		Adressering			Operations- beskrivning*	Flaggor			
Operation	Beteckning	Via X				3	2	1	0
		OP	#	~					
Unconditional jump	JMP ,X	95	1	3	X → PC	•	•	•	•
	JMP n,X	96	2	6	n+X → PC	•	•	•	•
	JMP A,X	97	1	5	A+X → PC	•	•	•	•
	JMP B,X	98	1	5	B+X → PC	•	•	•	•

3.12 Hopp till subrutin och återhopp från subrutin

Instruktion		Adressering									Operations- beskrivning*	Flaggor			
Operation	Beteckning	Inherent			Immediate			Absolute				3	2	1	0
		OP	#	~	OP	#	~	OP	#	~		N	Z	V	C
Jump to subroutine	JSR Adr							69	2	7	SP-1 → SP PC → M(SP) Adr → PC	•	•	•	•
Return from subroutine	RTS	6A	1	4							M(SP) → PC SP+1 → SP	•	•	•	•

3.13 Branch till subrutin

Instruktion		Adressering			Operations- beskrivning*	Flaggor			
Operation	Beteckning	Relative				3	2	1	0
		OP	#	~					
Branch to subroutine	BSR Adr	6B	2	7	SP-1 → SP PC → M(SP) PC+Offs → PC	•	•	•	•

3.14 Lagring av data på stack och hämtning av data från stack

Instruktion		Adressering			Operations- beskrivning*	Flaggor			
Operation	Beteckning	Inherent				3	2	1	0
		OP	#	~		N	Z	V	C
Push accumulator A	PSHA	6C	1	5	SP-1 → SP A → M(SP)	•	•	•	•
Push accumulator B	PSHB	6D	1	5	SP-1 → SP B → M(SP)	•	•	•	•
Push register CC	PSHC	6E	1	5	SP-1 → SP CCR → M(SP)	•	•	•	•
Push register X	PSHX	6F	1	5	SP-1 → SP X → M(SP)	•	•	•	•
Pull accumulator A	PULA	70	1	4	M(SP) → A SP+1 → SP	•	•	•	•
Pull accumulator B	PULB	71	1	4	M(SP) → B SP+1 → SP	•	•	•	•
Pull register CC	PULC	72	1	4	M(SP) → CCR SP+1 → SP	a	a	a	a
Pull register X	PULX	73	1	4	M(SP) → X SP+1 → SP	•	•	•	•

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

Instruktion		Adressering			Operations- beskrivning*	Flaggor			
Operation	Beteckning	Via X				3 N	2 Z	1 V	0 C
		OP	#	~					
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	•	•	•	•

3.16 *Tillägg till operationsbeskrivningar

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 återhoppadress på stacken vid subrutinanrop.

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