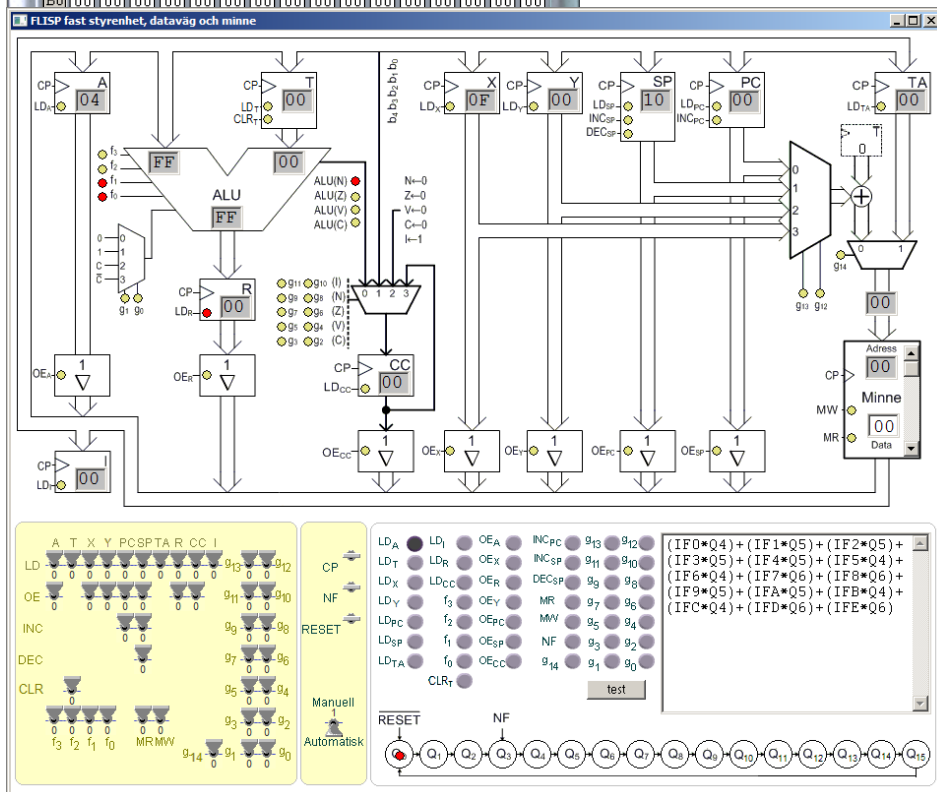
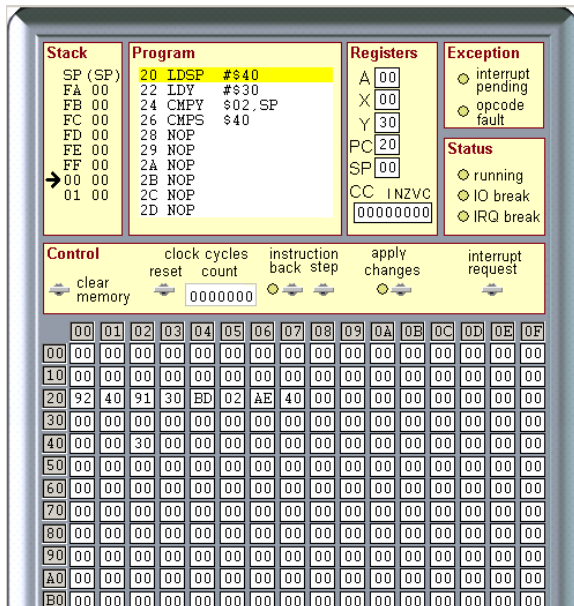


2013-08-14

FLEXIBLE INSTRUCTION SET PROCESSOR

FLISP



Detta häfte utgör den sammanfattande beskrivningen av FLIS-processorn. Häftet är indelat i två delar. Del 1 behandlar assemblerprogrammerarens bild av FLISP, del 2 beskriver detaljer i dataväg och den fasta styrenheten.

Del I: Programmerarens bild

Vektorer och minne
Registeruppsättning
Adresseringssätt
Assemblerspråk
Assemblerdirektiv
Instruktionsgrupper
Karta över operationskoder
Instruktionslista (detaljerad beskrivning)

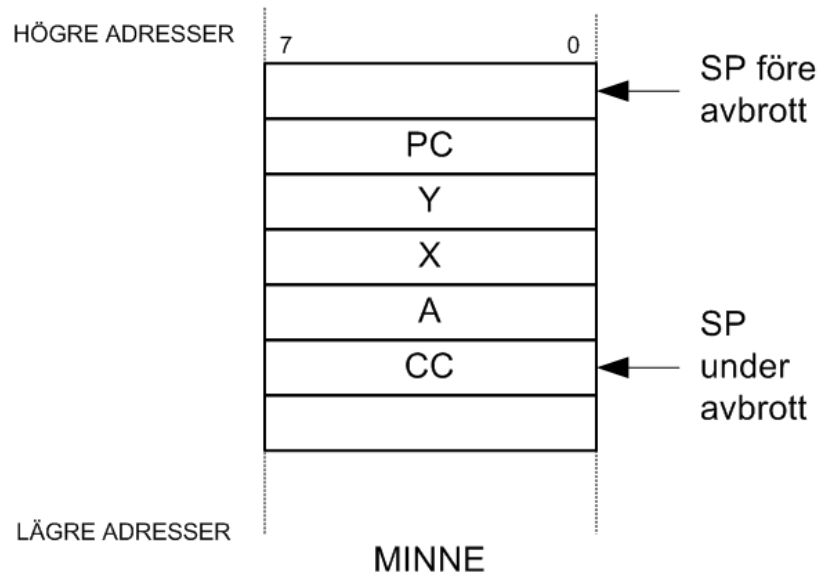
Del II: Mikroarkitekturen

RTN-notation
ALU-funktioner
Datavägen
Fast styrenhet

Vektorer och I/O

\$FF "Reset", återstart
 \$FE "Trap", undantag för ej implementerad operationskod
 \$FD "IRQ", avbrott
 \$FC "I/O" Port för anslutning av perifer enhet
 \$FB "I/O" Port för anslutning av perifer enhet

Stackens utseende vid "IRQ" eller "Trap"

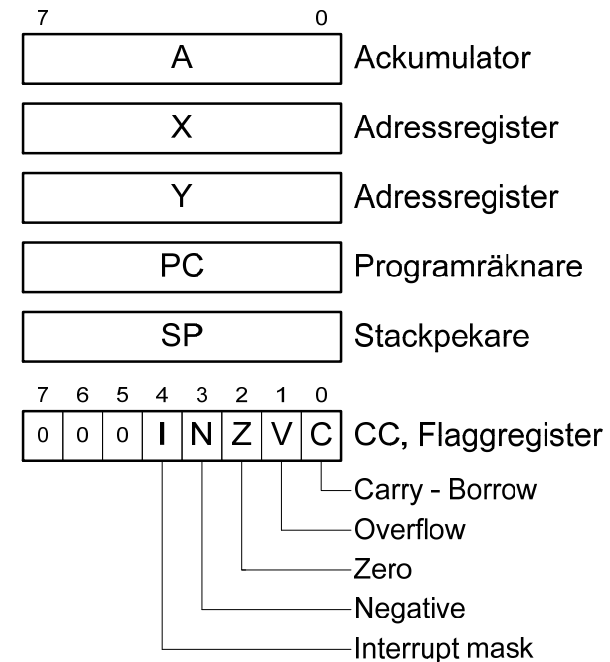


Vid "IRQ" pekar PC på den instruktion som skulle ha utförts om inget avbrott signalerats.

Vid "Exception" pekar PC på instruktionen omedelbart efter den otillåtna operationskoden.

Registeruppsättning

Det finns 6 register som är direkt åtkomliga för assemblerprogrammeraren.



Akkumulator: Generellt register för aritmetisk-/logik- operationer där resultatet alltid placeras i registret ("ackumuleras").

Adressregister: Används för adressberäkning.

Programräknare: Innehåller adressen till aktuell instruktion.

Stackpekare: Speciellt register, används implicit av vissa instruktioner.

CC, Flaggregister: Innehåller resultatindikatorer ("flaggor") från ALU'n och en bit för maskering av avbrott.

Adresseringssätt

Totalt används 8 distinkta adresseringssätt.

Inherent [ih]

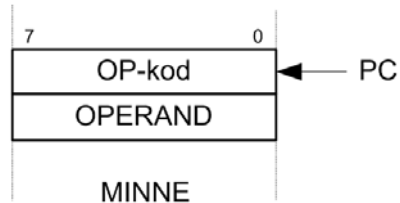
Operanden eller operanderna ges direkt av instruktionen. Ingen extra operandinformation (utöver operationskod) krävs. Det finns ingen generell RTN-beskrivning för inherent adressering.

Omedelbar (Immediate) [im]

Operanden är kodad tillsammans med operationen.

RTN: $M(PC+1)$

Assemblersyntax: #<DATA>

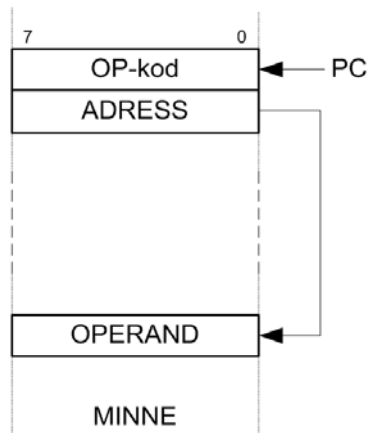


Absolut (Absolute) [ab]

Operanden finns i minnet. Minnesadressen är kodad tillsammans med operationen.

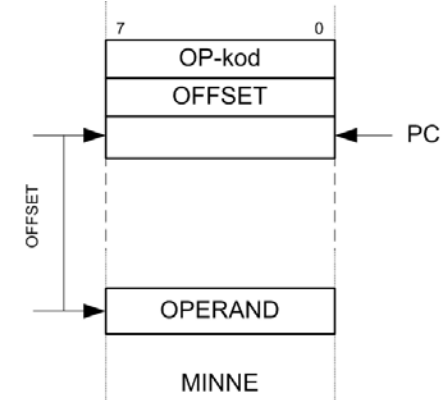
RTN: $[M(PC+1)]$

Assemblersyntax: <ADRESS>



Relativ [pc]

PC-relativ adressering används av programflödesinstruktioner ("branch on condition"). Adressen till operanden bestäms av en offset till aktuell PC, därav namnet "Relativ".



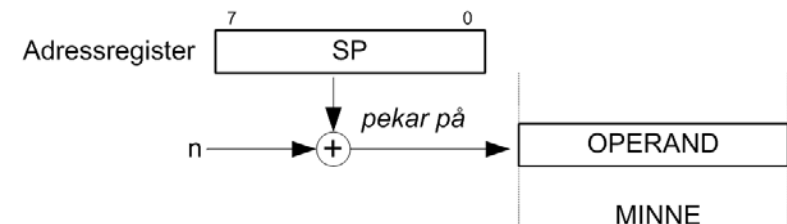
Observera att adressberäkningar som inbegriper PC förutsätter att PC pekar på nästa instruktion

Indirekt stackregister med konstant offset (Indexed) [ns]

Operanden finns i minnet. En konstant n ("offset") adderas till innehållet i register SP och bildar adressen till operandens plats i minnet.

RTN: $M(SP+n)$

Assemblersyntax: n, SP



Konstanten n (8 bitar) adderas modulo 256 och kan anges med eller utan tecken.

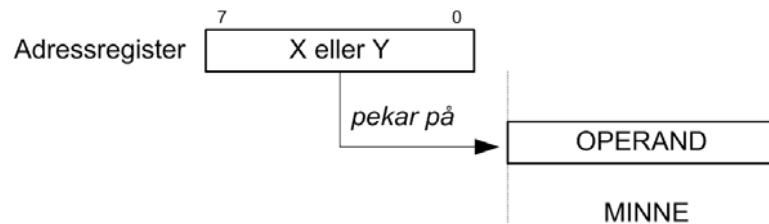
De indexerade adresseringssätten kan också användas tillsammans med något av registren X eller Y. I den följande framställningen betecknas något X eller Y med bokstaven R.

Indirekt register med konstant offset (Indexed) [nR]

Operanden finns i minnet. En konstant n ("offset") adderas till innehållet i register R (X eller Y) och bildar adressen till operandens plats i minnet.

RTN: $M(R+n)$;

Assemblersyntax: n, R



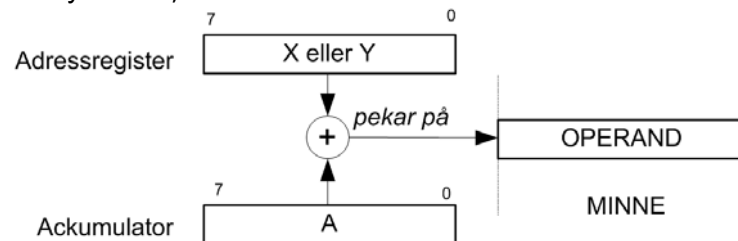
Konstanten **n** (8 bitar) adderas modulo 256 och kan anges med eller utan tecken.

Indirekt register med akkumulator offset (Indexed)[aR]

Operanden finns i minnet. Innehållet i register A adderas till innehållet i register R (X eller Y) och bildar adressen till operandens plats i minnet.

RTN: $M(R+A)$

Assemblersyntax: A, R

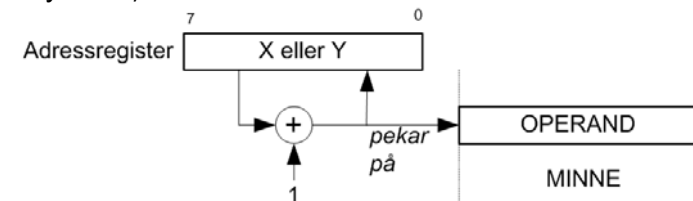


Indirekt register med pre/post decrement/increment (Indexed)

Pre auto increment [+R]

RTN: $R+1 \rightarrow R; M(R)$

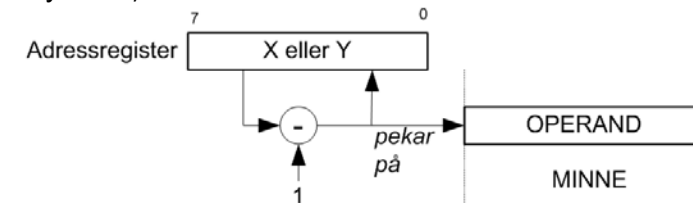
Assemblersyntax: , +R



Pre auto decrement [-R]

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

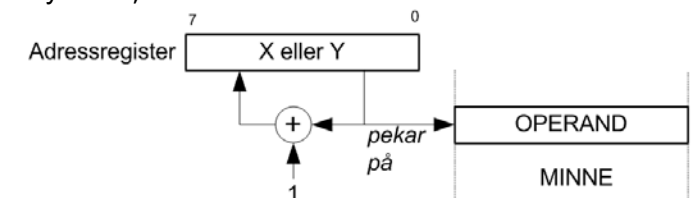
Assemblersyntax: , -R



Post auto increment [R+]

RTN: $M(R); R+1 \rightarrow R$

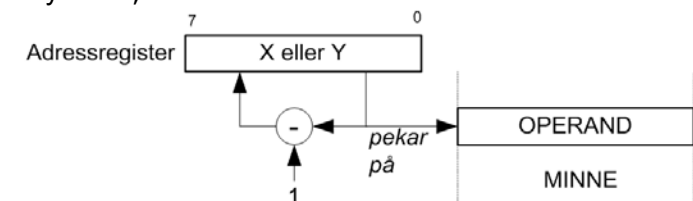
Assemblersyntax: , R+



Post auto decrement [R-]

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

Assemblersyntax: , R-



Assemblerspråk

Assemblerspråket består dels av mnemoniska beteckningar för Flisp's instruktioner och adresseringssätt dels av en uppsättning "pseudoinstruktioner", även kallade "assemblerdirektiv".

Adresseringssätt

Syntax för att ange specifikt adresseringssätt finns detaljerat beskrivet i föregående avsnitt.

Mnemoniska beteckningar för instruktioner

Se "Instruktionsuppsättning" eller "Instruktionslista" nedan.

Prefix för talbaser

Tal givna på decimal form saknar prefix

För att ange tal på binär form används prefixet '%'

För att ange tal på hexadecimal form används prefixet '\$'

Assemblerdirektiv

Följande tabell listar de vanligaste assemblerdirektiven tillsammans med korta förklaringar:

Följande förkortningar används:

<Val>	Värde (adress eller data) , kan anges som en konstant med godtycklig talbas (godtyckligt prefix). Kan också anges i form av symbol.
Sym	'Sym' står här för ett godtyckligt (tillåtet) symbolnamn.
[Sym]	'Sym' kan, men behöver inte anges. Då 'Sym' anges kommer symbolen att tilldelas värdet hos adressen till den första byten av det reserverade minnet.

Assemblerdirektiv:

ORG <Val>	"ORIGIN": Anger startadress för påföljande kod/data. Om en symbol används för att ange startadressen måste symbolen vara definierad, dvs inga framåtreferenser är tillåtna här.
Sym EQU <Val>	"EQUATE": Symbolen 'Sym' representerar värdet <Val>.
[Sym] FCB <Val>, <Val>...	"FORM CONSTANT BYTE": Skapar en sträng med initierade data i minnet
[Sym] FCS "<ASCII tecken>"	"FORM CONSTANT STRING": Skapar en sträng med ASCII-tecken i minnet.
[Sym] RMB <Val>	"RESERVE MEMORY BYTES": Reservera <Val> bytes i minnet. Minnesinnehållet på dessa adresser är odefinierat.

Instruktionsgrupper

"Load/Store"

LDA, LDX, LDY, LDSP, LEAX, LEAY, LEASP
STA, STX, STY, STSP

"Data movement"

TFR, EXG

"Program (Flow) control"

JMP, JSR, BRA, BSR, B(condition), RTS, RTI

"Integer arithmetic"

ADDA, ADCA, SUBA, SBCA,
CLRA, CLR, NEGA, NEG,
DECA, DEC, INCA, INC

"Integer test"

CMPA, CMPX, CMPY, CMPSP, BITA, TSTA, TST

"Logical operations"

ANDA, ORA, ANDCC, ORCC,
EORA, COMA, COM

"Shift/rotate"

ASRA, ASR,
LSLA, LSL, LSRA, LSR,
ROLA, ROL, RORA, ROR

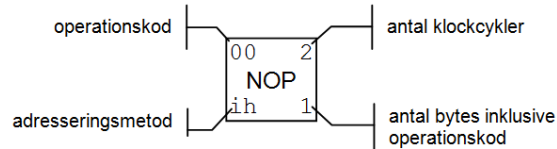
"Stack operations"

PSHA, PSHCC, PSHX, PSHY, PULA, PULCC, PULX, PULY

"Misc."

NOP

Karta över operationskoder



Anm:

”Blanka” fält utgör ”otillåten operationskod” och föranleder undantagshantering

00	210	320	530	340	350	360	370	380	390	2A0	3B0	3C0	3D0	3E0	F0	2
ih	NOP	PSHA	BSR	STX	STX	STX	STX	STX	STX	LDX	LDX	LDX	LDX	LDX	LDX	LDX
01	411	321	431	341	351	361	371	381	391	2A1	3B1	3C1	3D1	3E1	3F1	3
ANDCC	PSHX	BRA	STY	STY	STY	STY	STY	STY	LDY	LDY	LDY	LDY	LDY	STA	LDA	
im	2	1	2	2	2	2	1	2	1	2	2	2	2	2	2	2
02	412	322	432	342	352	362	372	382	392	2A2	3B2	3C2	3D2	3E2	3F2	3
ORCC	PSHY	BMI	STSP	STSP	STSP	STSP	STSP	STSP	LDSP	LDSP	LDSP	LDSP	LDSP	STA	LDA	
im	2	1	2	2	2	2	1	2	1	2	2	2	2	2	2	2
03	13	223	433	243	253	263	273	283	293	2A3	3B3	3C3	3D3	3E3	3F3	3
	PSHCC	BPL	JMP	RTS	JMP	JMP	JMP	JMP	JMP	SBCA	SBCA	SBCA	SBCA	STA	LDA	
ih	1	2	2	1	1	2	2	2	2	2	2	2	2	2	2	2
04	14	324	434	244	254	264	274	284	294	2A4	3B4	3C4	3D4	3E4	3F4	3
	PULA	BEQ	JSR	RTI	JSR	JSR	JSR	JSR	SUBA	SUBA	SUBA	SUBA	SUBA	STA	LDA	
ih	1	2	2	1	1	2	2	2	2	2	2	2	2	2	2	2
05	315	325	435	345	355	365	375	385	395	2A5	3B5	3C5	3D5	3E5	3F5	4
CLRA	PULX	BNE	CLR	CLR	CLR	CLR	CLR	CLR	ADCA	ADCA	ADCA	ADCA	ADCA	STA	LDA	
ih	1	1	2	2	2	2	2	2	1	2	2	2	2	2+	1+	1
06	316	326	436	346	356	366	376	386	396	2A6	3B6	3C6	3D6	3E6	3F6	4
NEGA	PULY	BVS	NEG	NEG	NEG	NEG	NEG	NEG	ADDA	ADDA	ADDA	ADDA	ADDA	STA	LDA	
ih	1	1	2	2	2	2	2	2	1	2	2	2	2	2-	1-	1
07	317	327	437	347	357	367	377	387	397	2A7	3B7	3C7	3D7	3E7	3F7	4
INCA	PULCC	BVC	INC	INC	INC	INC	INC	INC	CMPA	CMPA	CMPA	CMPA	CMPA	STA	LDA	
ih	1	1	2	2	2	2	2	2	1	2	2	2	2	2+	1+	1
08	318	328	438	348	358	368	378	388	398	2A8	3B8	3C8	3D8	3E8	3F8	4
DECA	TFR A,C	BCS	DEC	DEC	DEC	DEC	DEC	DEC	BITA	BITA	BITA	BITA	BITA	STA	LDA	
ih	1	1	2	2	2	2	2	2	1	2	2	2	2	2-	1-	1
09	219	229	439	349	359	369	379	389	399	2A9	3B9	3C9	3D9	3E9	3F9	3
TSTA	TFR C,A	BCC	TST	TST	TST	TST	TST	TST	ANDA	ANDA	ANDA	ANDA	ANDA	STA	LDA	
ih	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	2
0A	31A	22A	43A	34A	35A	36A	37A	38A	39A	2AA	3BA	3CA	3DA	3EA	3FA	3
COMA	TFR X,Y	BHI	COM	COM	COM	COM	COM	COM	ORA	ORA	ORA	ORA	ORA	STA	LDA	
ih	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	2
0B	31B	22B	43B	34B	35B	36B	37B	38B	39B	2AB	3BB	3CB	3DB	3EB	3FB	4
LSLA	TFR Y,X	BLS	LSL	LSL	LSL	LSL	LSL	LSL	EORA	EORA	EORA	EORA	EORA	STA	LDA	
ih	1	1	2	2	2	2	2	2	1	2	2	2	2	2+	1+	1
0C	31C	22C	43C	34C	35C	36C	37C	38C	39C	2AC	3BC	3CC	3DC	3EC	3FC	4
LSRA	TFR X,S	BGT	LSR	LSR	LSR	LSR	LSR	LSR	CMPX	CMPX	CMPX	LEAX	LEAX	STA	LDA	
ih	1	1	2	2	2	2	2	2	1	2	2	2	2	2-	1-	1
0D	31D	22D	43D	34D	35D	36D	37D	38D	39D	2AD	3BD	3CD	3DD	3ED	3FD	4
ROLA	TFR S,X	BGE	ROL	ROL	ROL	ROL	ROL	ROL	CMPY	CMPY	CMPY	LEAY	LEAY	STA	LDA	
ih	1	1	2	2	2	2	2	2	1	2	2	2	2	2+	1+	1
0E	31E	22E	43E	34E	35E	36E	37E	38E	39E	2AE	3BE	3CE	3DE	3EE	3FE	4
RORA	TFR Y,S	BLE	ROR	ROR	ROR	ROR	ROR	ROR	CMPSP	CMPSP	LEASP	LEASP	LEASP	STA	LDA	
ih	1	1	2	2	2	2	2	2	1	2	2	2	2	2-	1-	1
0F	31F	22F	43F	34F	35F	36F	37F	38F	39F	2AF	3BF	3CF	3DF	3EF	3FF	4
ASRA	TFR S,Y	BLT	ASR	ASR	ASR	ASR	ASR	ASR	EXG A,C	EXG X,Y	EXG X,S	EXG Y,S				
ih	1	1	2	2	2	2	2	2	1	1	1	1	1			

Detaljerad beskrivning av FLIS-processorns instruktioner

Instruktion	Adressering	Operation	Flaggor
Add with Carry	metod	OP # ~	N Z V C
Variant			
ADCA #Data	Immediate	95 2 A + Data + C → A	Δ Δ Δ Δ
ADCA Adr	Absolute	A5 2 A + M(Adr) + C → A	
ADCA n, SP	Indexed	B5 2 A + M(n + SP) + C → A	
ADCA n, X	Indexed	C5 2 A + M(n + X) + C → A	

Instruktion/Variant

Här anges instruktionens mnemonics med assemblersyntax för de tillgängliga adresseringssätten.

Adressering

OP	Operationskod för instruktion, hexadecimal form
#	Antal bytes i instruktionen
~	Antal klockcykler som krävs för att utföra en instruktion

Operationsbeskrivningar (RTN, register transfer notation)

n	Konstant uttryckt i talbas 10
N _r	Konstanten N uttryckt i talbasen r.
EA	Effektiv adress
Opr	Operand, data på effektiv adress
M(Adr)	Minnesinnehåll på adressen "Adr"
→	Kopiering

Följande symboler är beteckningar som reserverats för register

A	Akkumulator A
X	Adressregister X
Y	Adressregister Y
CC	Flaggregister (Condition Codes Register)
PC	Programräknare (Program Counter)
SP	Stackpekare (Stack Pointer)

Följande symboler är beteckningar som reserverats för villkorsindikatorer (flaggor).

N	Teckenflaggan ("Negative")
Z	Nollflaggan ("Zero")
V	Overflowflaggan
C	Carryflaggan

Operatörer

+	Addition
-	Subtraktion
^	Logiskt "OCH" (AND)
∨	Logiskt "ELLER" (OR)
⊕	Logiskt "EXKLUSIVT ELLER" (XOR)
Opr<<d	"Opr" skiftas vänster. Biten d skiftas in i den minst signifikanta positionen.
d>>Opr	"Opr" skiftas höger. Biten d skiftas in i den mest signifikanta positionen.
Opr'	Bitvis komplementering av operanden "Opr"
Flaggor	
-	Statusbiten påverkas ej vid operationen
0	Statusbiten nollställs vid operationen
1	Statusbiten ettställs vid operationen
Δ	Statusbiten nollställs/ettställs av operationens resultat
?	Odefinierat värde. Anger att en flagga får ett slumpartat värde efter operationen
!	Statusbiten används för speciellt syfte

ADCA Add data with carry into register A

RTN:	$A + \text{Opr} + C \rightarrow A$
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 2-komplements-overflow inträffar vid additionen C: Ettställs om summan vid additionen ej ryms i åtta bitar
Beskrivning:	Utför åttabitars addition av dataordet i minnet och innehållet i register A. Resultatets åtta minst signifikanta bitar placeras i register A. 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

Detaljer:

Instruktion ADD with Carry Variant	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
ADCA #Data	Immediate	95	2	4	$A + \text{Data} + C \rightarrow A$	Δ	Δ	Δ	Δ
ADCA Adr	Absolute	A5	2	5	$A + M(\text{Adr}) + C \rightarrow A$				
ADCA n, SP	Indexed	B5	2	5	$A + M(n + SP) + C \rightarrow A$				
ADCA n, X	Indexed	C5	2	5	$A + M(n + X) + C \rightarrow A$				
ADCA n, Y	Indexed	D5	2	5	$A + M(n + Y) + C \rightarrow A$				

ADDA Add data into register A

RTN	$A + \text{Opr} \rightarrow A$
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 register A. Resultatets åtta minst signifikanta bitar placeras i register A. Den nionde biten (mest signifikant) placeras i C- biten (C-flaggan) i CC-registret

Detaljer:

Instruktion ADD Variant	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
ADDA #Data	Immediate	96	2	4	$A + \text{Data} \rightarrow A$	Δ	Δ	Δ	Δ
ADDA Adr	Absolute	A6	2	5	$A + M(\text{Adr}) \rightarrow A$				
ADDA n, SP	Indexed	B6	2	5	$A + M(n + SP) \rightarrow A$				
ADDA n, X	Indexed	C6	2	5	$A + M(n + X) \rightarrow A$				
ADCA n, Y	Indexed	D5	2	5	$A + M(n + Y) + C \rightarrow A$				

ANDA Logical AND data into register A

RTN	$A \wedge \text{Opr} \rightarrow A$
Flaggor	N: Ettställs om resultatets teckenbit (bit 7) får värdet 1. Z: Ettställs om samtliga åtta bitar i resultatet blir noll. V: Nollställs. C: Påverkas ej
Beskrivning	Utför bitvis AND-operation mellan dataordet i minnet och innehållet i register A. Resultatet placeras i register A.

Detaljer:

Instruktion ANDA Variant	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
ANDA #Data	Immediate	99	2	4	$A \wedge \text{Data} \rightarrow A$	Δ	Δ	0	-
ANDA Adr	Absolute	A9	2	5	$A \wedge M(\text{Adr}) \rightarrow A$				
ANDA n, SP	Indexed	B9	2	5	$A \wedge M(n + SP) \rightarrow A$				
ANDA n, X	Indexed	C9	2	5	$A \wedge M(n + X) \rightarrow A$				
ANDA n, Y	Indexed	D9	2	5	$A \wedge M(n + Y) \rightarrow A$				

ANDCC Logical AND data into register CC

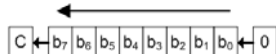
RTN	$CC \wedge \text{Data} \rightarrow CC$
Flaggor	Flaggorna nollställs i de positioner där CC eller Data innehåller någon nolla.
Beskrivning	Utför bitvis AND-operation mellan innehållet i flaggregistret (CC) och dataordet. Resultatet placeras i flaggregistret

Detaljer:

Instruktion ANDCC	Adressering				Operation	Flaggor			
	metod	OP	#	~		I	N	Z	V
ANDCC #Data	Immediate	01	2	4	$CC \wedge \text{Data} \rightarrow CC$	Δ	Δ	Δ	Δ

ASL Arithmetic shift left

RTN	$A \ll 1 \rightarrow A$ eller $M(EA) \ll 1 \rightarrow M(EA)$
Flaggor	N: Kopias av bit 7 efter skiftet. Z: Ettställs om samtliga åtta bitar i resultatet blir noll. V: Ettställs om C och bit 7 är olika efter operationen, dvs overflow vid 2-komplements-representation inträffar. C: bit 7 före skiftet blir ny carrybit efter skiftet.
Beskrivning	Skiftar operanden ett steg till vänster, dvs. multiplicerar ett tal med eller utan inbyggt tecken med 2. Instruktionen är identisk med LSL.

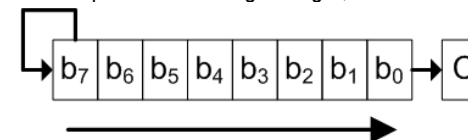


Detaljer:

Instruktion ASL	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
ASLA	Inherent	0B	1	3	$A \ll 1 \rightarrow A$	Δ	Δ	Δ	Δ
ASL Adr	Absolute	3B	2	4	$M(Adr) \ll 1 \rightarrow M(Adr)$				
ASL n, SP	Indexed	4B	2	4	$M(n+SP) \ll 1 \rightarrow M(n+SP)$				
ASL n, X	Indexed	5B	2	4	$M(n+X) \ll 1 \rightarrow M(n+X)$				
ASL A, X	Indexed	6B	1	4	$M(A+X) \ll 1 \rightarrow M(A+X)$				
ASL n, Y	Indexed	7B	2	4	$M(n+Y) \ll 1 \rightarrow M(n+Y)$				
ASL A, Y	Indexed	8B	1	4	$M(A+Y) \ll 1 \rightarrow M(A+Y)$				

ASR Arithmetic shift right

RTN	$A \gg 1 \rightarrow A$ eller $M \gg 1 \rightarrow M$
Flaggor	N: Ettställs om resultatets teckenbit (bit 7) får värdet 1. Z: Ettställs om samtliga åtta bitar i resultatet blir noll. V: Nollställs. C: bit 0 före skiftet blir ny carrybit efter skiftet.
Beskrivning	Skiftar operanden ett steg till höger, dvs. dividerar tal med inbyggt tecken med 2



Detaljer:

Instruktion ASR	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
ASRA	Inherent	0F	1	3	$A \gg 1 \rightarrow A$	Δ	Δ	0	Δ
ASR Adr	Absolute	3F	2		$M(Adr) \gg 1 \rightarrow M(Adr)$				
ASR n, SP	Indexed	4F	2		$M(n+SP) \gg 1 \rightarrow M(n+SP)$				
ASR n, X	Indexed	5F	2		$M(n+X) \gg 1 \rightarrow M(n+X)$				
ASR A, X	Indexed	6F	1		$M(A+X) \gg 1 \rightarrow M(A+X)$				
ASR n, Y	Indexed	7F	2		$M(n+Y) \gg 1 \rightarrow M(n+Y)$				
ASR A, Y	Indexed	8F	1		$M(A+Y) \gg 1 \rightarrow M(A+Y)$				

BCC Branch on carry clear (= BHS)

RTN	If C = 0: PC+Offset $\rightarrow$ PC
Flaggor	Påverkas ej
Beskrivning	Testar C-flaggans värde. Om C=0 utförs ett hopp till adressen ADRESS = 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.

Detaljer:

Instruktion BCC	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
BCC Adr	Relativ	29	2	4	If C=0: PC+Offset $\rightarrow$ PC	-	-	-	-

BCS Branch on carry set (= BLO)

RTN	If C = 1: PC+Offset → PC
Flaggor	Påverkas ej
Beskrivning	Testar C-flaggans värde. Om C=1 utförs ett hopp till adressen ADRESS = 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.

Detaljer:

Instruktion BCS	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
BCS Adr	Relativ	28	2	4	If (C = 1) PC+Offset → PC	-	-	-	-

BEQ Branch on equal to zero

RTN	If Z = 1: PC+Offset → PC
Flaggor	Påverkas ej
Beskrivning	Testar Z-flaggans värde. Om Z=1 utförs ett hopp till adressen ADRESS = 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.

Detaljer:

Instruktion BEQ	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
BEQ Adr	Relativ	24	2	4	If (Z = 1) PC+Offset → PC	-	-	-	-

BGE Branch on greater than or equal to zero

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 ADRESS = 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.

Detaljer:

Instruktion BGE		Adressering				Operation	Flaggor			
		metod	OP	#	~		N	Z	V	C
BGE	Adr	Relativ	2D	2	4	If ((N⊕V) = 0) PC+Offset → PC	-	-	-	-

BGT Branch on greater than zero

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

Detaljer:

Instruktion BGT		Adressering				Operation	Flaggor			
		metod	OP	#	~		N	Z	V	C
BGT	Adr	Relativ	2C	2	4	If $((N \oplus V) + Z = 0)$ PC+Offset → PC	-	-	-	-

BHI Branch if higher than zero

RTN	If $C+Z = 0$: $PC+Offset \rightarrow PC$
Flaggor	Påverkas ej
Beskrivning	Beskrivning: Testar värdet hos Booleska uttrycket $C+Z$. Om $C+Z = 0$ utförs ett hopp till adressen $ADRESS = PC+Offset$. 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.

Detaljer:

Instruktion BHI	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
BHI Adr	Relativ	2A	2	4	If $(C+Z = 0)$: $PC+Offset \rightarrow PC$	-	-	-	-

BITA Bit test register A

RTN:	$A \wedge Opr$
Flaggor:	N: Ettställs om resultatets teckenbit (bit 7) får värdet 1. Z: Ettställs om samtliga åtta bitar i resultatet blir noll. V: Nollställs. C: Påverkas ej
Beskrivning:	Utför bitvis AND-operation mellan dataordet i minnet och innehållet i register A. Resultatet lagras ej, utan påverkar endast flaggorna

Detaljer:

Instruktion BITA Variant	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
BITA #Data	Immediate	98	2	3	$A \wedge Data$	Δ	Δ	0	-
BITA Adr	Absolute	A8	2	4	$A \wedge M(Adr)$				
BITA n, SP	Indexed	B8	2	4	$A \wedge M(n+SP)$				
BITA n, X	Indexed	C8	2	4	$A \wedge M(n+X)$				
BITA n, Y	Indexed	D8	2	4	$A \wedge M(n+Y)$				

BLE Branch on less than or equal to zero

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 $ADRESS = PC+Offset$. 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.

Detaljer:

Instruktion BLE	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
BLE Adr	Relativ	2E	2	4	If $(N \oplus V) + Z = 1$: $PC+Offset \rightarrow PC$	-	-	-	-

BLS Branch on lower or same

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 $ADRESS = PC+Offset$. 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.

Detaljer:

Instruktion BLS	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
BLS Adr	Relativ	2B	2	4	If $C+Z = 1$: $PC+Offset \rightarrow PC$	-	-	-	-

BLT Branch on less than zero

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 ADRESS = 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.

Detaljer:

Instruktion BLT	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
BLT Adr	Relativ	2F	2	4	If $(N \oplus V) = 1$: PC+Offset $\rightarrow$ PC	-	-	-	-

BMI Branch on minus

RTN	If N = 1: PC+Offset $\rightarrow$ PC
Flaggor	Påverkas ej
Beskrivning	Testar N-flaggans värde. Om N=1 utförs ett hopp till adressen ADRESS = 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.

Detaljer:

Instruktion BMI	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
BMI Adr	Relativ	22	2	4	If N = 1: PC+Offset $\rightarrow$ PC	-	-	-	-

BNE Branch if not equal to zero

RTN	If Z = 0: PC+Offset $\rightarrow$ PC
Flaggor	Påverkas ej
Beskrivning	Testar Z-flaggans värde. Om Z=0 utförs ett hopp till adressen ADRESS = 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.

Detaljer:

Instruktion BNE	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
BNE Adr	Relativ	25	2	4	If (Z = 0) PC+Offset $\rightarrow$ PC	-	-	-	-

BPL Branch on plus

RTN	If N = 0: PC+Offset $\rightarrow$ PC
Flaggor	Påverkas ej
Beskrivning	Testar N-flaggans värde. Om N=0 utförs ett hopp till adressen ADRESS = 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.

Detaljer:

Instruktion BPL	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
BPL Adr	Relativ	23	2	4	If N = 0: PC+Offset $\rightarrow$ PC	-	-	-	-

BRA Branch always

RTN	PC+Offset → PC
Flaggor	Påverkas ej
Beskrivning	Ett hopp utförs till adressen ADDRESS = 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

Detaljer:

Instruktion BRA	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
BRA Adr	Relativ	21	2	4	PC+Offset → PC	-	-	-	-

BSR Branch to subroutine

RTN	SP-1 → SP PC → M(SP) PC+Offset → PC
Flaggor	Påverkas ej
Beskrivning	PC-värdet (återhoppadressen) skrivs först på stacken. Ett hopp utförs sedan till adressen ADDRESS = PC+Offset. Offset räknas från adressen efter BSR-instruktionen, dvs vid uträkningen av hopp-adressen pekar PC på operationskoden direkt efter BSR-instruktionen (= återhoppadressen)

Detaljer:

Instruktion BSR	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
BSR Adr	Relativ	20	2	5	SP-1 → SP PC → M(SP) PC+Offset → PC	-	-	-	-

BVC Branch if no overflow

RTN	If V = 0: PC+Offset → PC
Flaggor	Påverkas ej
Beskrivning	Testar V-flaggans värde. Om V=0 utförs ett hopp till adressen ADDRESS = 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

Detaljer:

Instruktion BVC	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
BVC Adr	Relativ	27	2	4	If V = 0: PC+Offset → PC	-	-	-	-

BVS Branch if overflow

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

Detaljer:

Instruktion BVS	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
BVS Adr	Relativ	26	2	4	If V = 1: PC+Offset → PC	-	-	-	-

CLR Clear register or memory

RTN	$0 \rightarrow A$, eller $0 \rightarrow M$
Flaggor	N: Nollställs. Z: Ettställs. V: Nollställs C: Nollställs
Beskrivning	Beskrivning: Register A eller innehållet på minnesadress nollställs.

Detaljer:

Instruktion CLR	Adressering				Operation	Flaggor			
Variant	metod	OP	#	~		N	Z	V	C
CLRA	Inherent	05	1	3	$0 \rightarrow A$	0	1	0	0
CLR Adr	Absolute	35	2	3	$0 \rightarrow M(\text{Adr})$				
CLR n, SP	Indexed	45	2	3	$0 \rightarrow M(n+\text{SP})$				
CLR n, X	Indexed	55	2	3	$0 \rightarrow M(n+X)$				
CLR A, X	Indexed	65	1	3	$0 \rightarrow M(A+X)$				
CLR n, Y	Indexed	75	2	3	$0 \rightarrow M(n+Y)$				
CLR A, Y	Indexed	85	1	3	$0 \rightarrow M(A+Y)$				

CMP Compare register and data

RTN	$R - \text{Opr}$ där $R = A, X, Y$ 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	Operanden subtraheras från innehållet i det angivna registret. Skillnaden lagras ej, utan påverkar endast flaggorna.

Detaljer:

Instruktion CMP	Adressering				Operation	Flaggor			
Variant	metod	OP	#	~		N	Z	V	C
CMPA #Data	Immediate	97	2	3	$A - \text{Data}$	Δ	Δ	Δ	Δ
CMPA Adr	Absolute	A7	2	4	$A - M(\text{Adr})$				
CMPA n, SP	Indexed	B7	2	4	$A - M(n+\text{SP})$				
CMPA n, X	Indexed	C7	2	4	$A - M(n+X)$				
CMPA n, Y	Indexed	D7	2	4	$A - M(n+Y)$				
CMPX #Data	Immediate	9C	2	3	$X - \text{Data}$				
CMPX Adr	Absolute	AC	2	4	$X - M(\text{Adr})$				
CMPX n, SP	Indexed	BC	2	4	$X - M(n+\text{SP})$				
CMPLY #Data	Immediate	9D	2	3	$Y - \text{Data}$				
CMPLY Adr	Absolute	AD	2	4	$Y - M(\text{Adr})$				
CMPLY n, SP	Indexed	BD	2	4	$Y - M(n+\text{SP})$				
CMPSP #Data	Immediate	9E	2	3	$\text{SP} - \text{Data}$				
CMPSP Adr	Absolute	AE	2	4	$\text{SP} - M(\text{Adr})$				

COM Complement register or memory

RTN	$R' \rightarrow R$ eller $M' \rightarrow M$
Flaggor	N: Ettställs om resultatets teckenbit (bit 7) får värdet 1. Z: Ettställs om samtliga åtta bitar i resultatet blir noll. V: Nollställs. C: Påverkas ej

Beskrivning

Detaljer:

Instruktion COM	Adressering				Operation	Flaggor			
Variant	metod	OP	#	~		N	Z	V	C
COMA	Inherent	0A	1	3	$A' \rightarrow A$	Δ	Δ	0	-
COM Adr	Absolute	3A	2	4	$M(\text{Adr})' \rightarrow M(\text{Adr})$				
COM n, SP	Indexed	4A	2	4	$M(n+\text{SP})' \rightarrow M(n+\text{SP})$				
COM n, X	Indexed	5A	2	4	$M(n+X)' \rightarrow M(n+X)$				
COM A, X	Indexed	6A	1	4	$M(A+X)' \rightarrow M(A+X)$				
COM n, Y	Indexed	7A	2	4	$M(n+Y)' \rightarrow M(n+Y)$				
COM A, Y	Indexed	8A	1	4	$M(A+Y)' \rightarrow M(A+Y)$				

DEC Decrement register or memory

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: Påverkas ej

Beskrivning Subtraherar 1 från operanden

Detaljer:

Instruktion DEC	Adressering				Operation	Flaggor			
Variant	metod	OP	#	~		N	Z	V	C
DECA	Inherent	08	1	3	$A-1 \rightarrow A$	Δ	Δ	Δ	-
DEC Adr	Absolute	38	2	4	$M(\text{Adr})-1 \rightarrow M(\text{Adr})$				
DEC n, SP	Indexed	48	2	4	$M(n+\text{SP})-1 \rightarrow M(n+\text{SP})$				
DEC n, X	Indexed	58	2	4	$M(n+X)-1 \rightarrow M(n+X)$				
DEC A, X	Indexed	68	1	4	$M(A+X)-1 \rightarrow M(A+X)$				
DEC n, Y	Indexed	78	2	4	$M(n+Y)-1 \rightarrow M(n+Y)$				
DEC A, Y	Indexed	88	1	4	$M(A+Y)-1 \rightarrow M(A+Y)$				

EOR Exclusive-OR register A

RTN	$A \oplus M \rightarrow A$
Flaggor	N: Ettställs om resultatets teckenbit (bit 7) får värdet 1. Z: Ettställs om samtliga åtta bitar i resultatet blir noll. V: Nollställs. C: Påverkas ej.

Beskrivning Utför bitvis XOR-operation mellan dataordet i minnet och innehållet i register A. Resultatet placeras i register A.

Detaljer:

Instruktion DEC	Adressering				Operation	Flaggor			
Variant	metod	OP	#	~		N	Z	V	C
EORA #Data	Immediate	9B	2	4	$A \oplus \text{Data} \rightarrow A$	Δ	Δ	0	-
EORA Adr	Absolute	AB	2	5	$A \oplus M(\text{Adr}) \rightarrow A$				
EORA n, SP	Indexed	BB	2	5	$A \oplus M(n+\text{SP}) \rightarrow A$				
EORA n, X	Indexed	CB	2	5	$A \oplus M(n+X) \rightarrow A$				
EORA n, Y	Indexed	DB	2	5	$A \oplus M(n+Y) \rightarrow A$				

EXG Exchange register contents

RTN	$R1 \leftrightarrow R2$
Flaggor	Påverkas endast om CC-registret är det ena registret som används
Beskrivning	Data skiftas mellan angivna register

Detaljer:

Instruktion EXG	Adressering				Operation	Flaggor			
Variant	metod	OP	#	~		N	Z	V	C
EXG A, CC	Inherent	9F	1	4	$A \leftrightarrow \text{CC}$	Δ	Δ	Δ	Δ
EXG X, Y	Inherent	AF	1	4	$X \leftrightarrow Y$	-	-	-	-
EXG X, SP	Inherent	BF	1	4	$X \leftrightarrow \text{SP}$	-	-	-	-
EXG Y, SP	Inherent	CF	1	4	$Y \leftrightarrow \text{SP}$	-	-	-	-

INC Increment register or memory

RTN	$A + 1 \rightarrow A$ 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: Påverkas ej
Beskrivning	

Detaljer:

Instruktion	Adressering				Operation	Flaggor			
INC									
Variant	metod	OP	#	~		N	Z	V	C
INCA	Inherent	07	1	3	$A+1 \rightarrow A$	Δ	Δ	Δ	-
INC Adr	Absolute	37	2	4	$M(\text{Adr}) + 1 \rightarrow M(\text{Adr})$				
INC n, SP	Indexed	47	2	4	$M(n+\text{SP}) + 1 \rightarrow M(n+\text{SP})$				
INC n, X	Indexed	57	2	4	$M(n+X) + 1 \rightarrow M(n+X)$				
INC A, X	Indexed	67	1	4	$M(A+X) + 1 \rightarrow M(A+X)$				
INC n, Y	Indexed	77	2	4	$M(n+Y) + 1 \rightarrow M(n+Y)$				
INC A, Y	Indexed	87	1	4	$M(A+Y) + 1 \rightarrow M(A+Y)$				

JMP Jump

RTN	$EA \rightarrow PC$
Flaggor	Påverkas ej
Beskrivning	Ovillkorlig programflödesändring, nästa instruktion hämtas från effektiva adressen EA.

Detaljer:

Instruktion	Adressering				Operation	Flaggor			
JMP									
Variant	metod	OP	#	~		N	Z	V	C
JMP Adr	Absolute	33	2	2	$\text{Adr} \rightarrow PC$	-	-	-	-
JMP n, X	Indexed	53	2	4	$n+X \rightarrow PC$				
JMP A, X	Indexed	63	1	4	$A+X \rightarrow PC$				
JMP n, Y	Indexed	73	2	4	$n+Y \rightarrow PC$				
JMP A, Y	Indexed	83	1	4	$A+Y \rightarrow PC$				

JSR Jump to subroutine

RTN	$SP-1 \rightarrow SP$; $PC \rightarrow M(SP)$; $EA \rightarrow PC$
Flaggor	Påverkas ej
Beskrivning	PC-värdet, som är adressen till instruktionen efter JSR-instruktionen, dvs återhoppsadressen, skrivs först på stacken. Ett hopp utförs sedan till adressen EA.

Detaljer:

Instruktion	Adressering				Operation	Flaggor			
JSR									
Variant	metod	OP	#	~		N	Z	V	C
JSR Adr	Absolute	34	2	4	$SP-1 \rightarrow SP$ $PC \rightarrow M(SP)$ $\text{Adr} \rightarrow PC$	-	-	-	-
JSR n, X	Indexed	54	2	5	$SP-1 \rightarrow SP$ $PC \rightarrow M(SP)$ $n+X \rightarrow PC$				
JSR A, X	Indexed	64	1	5	$SP-1 \rightarrow SP$ $PC \rightarrow M(SP)$ $A+X \rightarrow PC$				
JSR n, Y	Indexed	74	2	5	$SP-1 \rightarrow SP$ $PC \rightarrow M(SP)$ $n+X \rightarrow PC$				
JSR A, Y	Indexed	84	1	5	$SP-1 \rightarrow SP$ $PC \rightarrow M(SP)$ $A+X \rightarrow PC$				

LD Load register

RTN	M(EA) → R
Flaggor	N: Ettställs om resultatets teckenbit (bit 7) får värdet 1. Z: Ettställs om samtliga åtta bitar i resultatet blir noll. V: Nollställs. C: Påverkas ej
Beskrivning	Laddar dataord från minnet till angivet register R (A,X,Y eller SP)

Detaljer:

Instruktion	Adressering					Operation				Flaggor			
LD Variant	metod	OP	#	~						N	Z	V	C
LDA #Data	Immediate	F0	2	2	Data → A	Δ	Δ	0	-				
LDA Adr	Absolute	F1	2	3	M(Adr) → A								
LDA n, SP	Indexed	F2	2	3	M(n+SP) → A								
LDA n, X	Indexed	F3	1	3	M(n+X) → A								
LDA A, X	Indexed	F4	1	3	M(A+X) → A								
LDA , X+	Indexed	F5	1	4	M(X) → A, X+1 → X								
LDA , X-	Indexed	F6	1	4	M(X) → A, X-1 → X								
LDA , +X	Indexed	F7	1	4	X+1 → X, M(X) → A								
LDA , -X	Indexed	F8	1	4	X-1 → X, M(X) → A								
LDA n, Y	Indexed	F9	1	3	M(n+Y) → A								
LDA A, Y	Indexed	FA	1	3	M(A+Y) → A								
LDA , Y+	Indexed	FB	1	4	M(Y) → A, Y+1 → Y								
LDA , Y-	Indexed	FC	1	4	M(Y) → A, Y-1 → Y								
LDA , +Y	Indexed	FD	1	4	Y+1 → Y, M(Y) → A								
LDA , -Y	Indexed	FE	1	4	Y-1 → Y, M(Y) → A								
LDX #Data	Immediate	90	2	2	Data → X								
LDX Adr	Absolute	A0	2	3	M(Adr) → X								
LDX n, SP	Indexed	B0	2	3	M(n+SP) → X								
LDX n, X	Indexed	C0	2	3	M(n+X) → X								
LDX n, Y	Indexed	D0	2	3	M(n+Y) → X								
LDY #Data	Immediate	91	2	2	Data → Y								
LDY Adr	Absolute	A1	2	3	M(Adr) → Y								
LDY n, SP	Indexed	B1	2	3	M(n+SP) → Y								
LDY n, X	Indexed	C1	2	3	M(n+X) → Y								
LDY n, Y	Indexed	D1	2	3	M(n+Y) → Y								
LDSP #Data	Immediate	92	2	2	Data → SP								
LDSP Adr	Absolute	A2	2	3	M(Adr) → SP								
LDSP n, SP	Indexed	B2	2	3	M(n+SP) → SP								
LDSP n, X	Indexed	C2	2	3	M(n+X) → SP								
LDSP n, Y	Indexed	D2	2	3	M(n+Y) → SP								

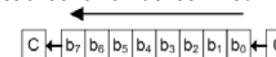
LEA Load effective address

Varianter	
RTN	EA → R ; R kan vara X,Y eller SP
Flaggor	Påverkas ej
Beskrivning	Laddar effektiva adressen i R.
Detaljer:	

Instruktion	Adressering					Operation				Flaggor			
LEA Variant	metod	OP	#	~						N	Z	V	C
LEAX n, X	Indexed	CC	2	4	X + n → X	-	-	-	-				
LEAX n, SP	Indexed	DC	2	4	SP + n → X								
LEAY n, Y	Indexed	CD	2	4	Y + n → Y								
LEAY n, SP	Indexed	DD	2	4	SP + n → Y								
LEASP n, SP	Indexed	BE	2	4	SP + n → SP								
LEASP n, X	Indexed	CE	2	4	X + n → SP								
LEASP n, Y	Indexed	DE	2	4	Y + n → SP								

LSL Logical shift left, ASL Arithmetic shift left

RTN	A <<1 → A eller M(EA) <<1 → M(EA)
Flaggor	N: Kopia av bit 7 efter skiftet. Z: Ettställs om samtliga åtta bitar i resultatet blir noll. V: Ettställs om C och bit 7 är olika efter operationen, dvs overflow vid 2-komplements-representation inträffar. C: bit 7 före skiftet blir ny carrybit efter skiftet.
Beskrivning	Skiftar operanden ett steg till vänster, dvs. multiplicerar ett tal med eller utan inbyggt tecken med 2. Instruktionen är identisk med ASL.

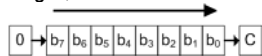


Detaljer:

Instruktion	Adressering					Operation				Flaggor			
LSL Variant	metod	OP	#	~						N	Z	V	C
LSLA	Inherent	0B	1	3	A <<1 → A	Δ	Δ	Δ	Δ				
LSL Adr	Absolute	3B	2	4	M(Adr) <<1 → M(Adr)								
LSL n, SP	Indexed	4B	2	4	M(n+SP) <<1 → M(n+SP)								
LSL n, X	Indexed	5B	2	4	M(n+X) <<1 → M(n+X)								
LSL A, X	Indexed	6B	1	4	M(A+X) <<1 → M(A+X)								
LSL n, Y	Indexed	7B	2	4	M(n+Y) <<1 → M(n+Y)								
LSL A, Y	Indexed	8B	1	4	M(A+Y) <<1 → M(A+Y)								

LSR Logical shift right

RTN	$A \gg 1 \rightarrow A$ eller $M \gg 1 \rightarrow M$
Flaggor	N: Nollställs. Z: Ettställs om samtliga åtta bitar i resultatet blir noll. V: Ettställs om overflow vid 2-komplements-representation inträffar. C: bit 0 före skiftet blir ny carrybit efter skiftet.
Beskrivning	Skiftar operanden ett steg till höger, dvs. dividerar ett tal utan inbyggt tecken med 2



Detaljer:

Instruktion LSR Variant	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
LSRA	Inherent	0C	1	3	$A \gg 1 \rightarrow A$	0	Δ	Δ	Δ
LSR Adr	Absolute	3C	2	4	$M(\text{Adr}) \gg 1 \rightarrow M(\text{Adr})$				
LSR n, SP	Indexed	4C	1	4	$M(n+\text{SP}) \gg 1 \rightarrow M(n+\text{SP})$				
LSR n, X	Indexed	5C	2	4	$M(n+X) \gg 1 \rightarrow M(n+X)$				
LSR A, X	Indexed	6C	1	4	$M(A+X) \gg 1 \rightarrow M(A+X)$				
LSR n, Y	Indexed	7C	2	4	$M(n+Y) \gg 1 \rightarrow M(n+Y)$				
LSR A, Y	Indexed	8C	1	4	$M(A+Y) \gg 1 \rightarrow M(A+Y)$				

NEG Negate register or memory

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.

Detaljer:

Instruktion NEG Variant	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
NEGA	Inherent	06	1	3	$-A \rightarrow A$	Δ	Δ	Δ	Δ
NEG Adr	Absolute	36	2	4	$-M(\text{Adr}) \rightarrow M(\text{Adr})$				
NEG n, SP	Indexed	46	2	4	$-M(n+\text{SP}) \rightarrow M(n+\text{SP})$				
NEG n, X	Indexed	56	2	4	$-M(n+X) \rightarrow M(n+X)$				
NEG A, X	Indexed	66	1	4	$-M(A+X) \rightarrow M(A+X)$				
NEG n, Y	Indexed	76	2	4	$-M(n+Y) \rightarrow M(n+Y)$				
NEG A, Y	Indexed	86	1	4	$-M(A+Y) \rightarrow M(A+Y)$				

NOP No operation

RTN									
Flaggor		Påverkas ej							
Beskrivning		Instruktionen utför ingenting							
Detaljer:									
Instruktion NOP	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
NOP	Inherent	00	1	2	No operation	-	-	-	-

ORA Logical OR data into register A

RTN	$A \vee M \rightarrow A$
Flaggor	N: Ettställs om resultatets teckenbit (bit 7) får värdet 1. Z: Ettställs om samtliga åtta bitar i resultatet blir noll. V: Nollställs. C: Påverkas ej
Beskrivning	Utför bitvis OR-operation mellan dataordet i minnet och innehållet i register A. Resultatet placeras i register A.

Detaljer:

Instruktion ORA Variant	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
ORA #Data	Immediate	9A	2	4	$A \vee \text{Data} \rightarrow A$	Δ	Δ	0	-
ORA Adr	Absolute	AA	2	5	$A \vee M(\text{Adr}) \rightarrow A$				
ORA n, SP	Indexed	BA	2	5	$A \vee M(n+\text{SP}) \rightarrow A$				
ORA n, X	Indexed	CA	2	5	$A \vee M(n+X) \rightarrow A$				
ORA n, Y	Indexed	DA	2	5	$A \vee M(n+Y) \rightarrow A$				

ORCC Logical OR Data into register CC

RTN	$CC \vee \text{Data} \rightarrow CC$
Flaggor	Flaggorna ettställs i de positioner där CC eller Data innehåller någon etta
Beskrivning	Utför bitvis OR-operation mellan innehållet i flaggregistret (CC) och dataordet. Resultatet placeras i flaggregistret

Detaljer:

Instruktion ORCC	Adressering				Operation	Flaggor				
	metod	OP	#	~		I	N	Z	V	C
ORCC #Data	Inherent	02	2	4	$CC \vee \text{Data} \rightarrow CC$	Δ	Δ	Δ	Δ	Δ

PSH Push register on Stack

RTN	SP-1 → SP R → M(SP)		
Flaggor	Påverkas ej		
Beskrivning	Stackpekaren uppdateras först. Angivet registerinnehåll skrivs sedan på stacken		

Detaljer:

Instruktion PSH Variant	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
PSHA	Inherent	10	1	3	SP-1 → SP A → M(SP)	-	-	-	-
PSHX	Inherent	11	1	3	SP-1 → SP X → M(SP)				
PSHY	Inherent	12	1	3	SP-1 → SP Y → M(SP)				
PSHCC	Inherent	13	1	3	SP-1 → SP CC → M(SP)				

PUL Pull register from stack

Varianter			
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		

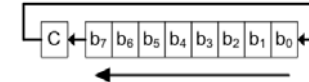
Detaljer:

Instruktion PUL Variant	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
PULA	Inherent	14	1	3	M(SP) → PC, SP+1 → SP	-	-	-	-
PULX	Inherent	15	1	3	M(SP) → X SP+1 → SP	-	-	-	-
PULY	Inherent	16	1	3	M(SP) → Y SP+1 → SP	-	-	-	-
PULCC	Inherent	17	1	3	M(SP) → CC SP+1 → SP	Δ	Δ	Δ	Δ

ROL Rotate left

RTN	A << A → A eller M << 1 → M		
Flaggor	N: Kopia av bit 7 efter skiftet. Z: Ettställs om samtliga åtta bitar i resultatet blir noll. V: Ettställs om C och bit 7 är olika efter operationen, dvs overflow vid 2-komplements-representation inträffar. C: bit 7 före skiftet blir ny carrybit efter skiftet.		

Beskrivning Skiftar operanden ett steg till vänster, dvs. multiplicerar ett tal med eller utan inbyggt tecken med 2

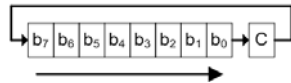


Detaljer:

Instruktion ROL Variant	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
ROLA	Inherent	0D	1	3	A << 1 → A	Δ	Δ	Δ	Δ
ROL Adr	Absolute	3D	2	4	M(Adr) << 1 → M(Adr)				
ROL n, SP	Indexed	4D	2	4	M(n+SP) << 1 → M(n+SP)				
ROL n, X	Indexed	5D	2	4	M(n+X) << 1 → M(n+X)				
ROL A, X	Indexed	6D	1	4	M(A+X) << 1 → M(A+X)				
ROL n, Y	Indexed	7D	2	4	M(n+Y) << 1 → M(n+Y)				
ROL A, Y	Indexed	8D	1	4	M(A+Y) << 1 → M(A+Y)				

ROR Rotate right

RTN	$A \gg 1 \rightarrow A$ eller $M \gg 1 \rightarrow M$
Flaggor	N: C före skiftoperationen. Z: Ettställs om samtliga åtta bitar i resultatet blir noll. V: Ettställs om overflow vid 2-komplements-representation inträffar. C: bit 0 före skiftet blir ny carrybit efter skiftet.
Beskrivning	Skiftar operanden ett steg till höger, dvs. dividerar ett tal utan inbyggt tecken med 2



Detaljer:

Instruktion ROR Variant	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
RORA	Inherent	0E	1	3	$A \gg 1 \rightarrow A$	Δ	Δ	Δ	Δ
ROR Adr	Absolute	3E	2	4	$M(\text{Adr}) \gg 1 \rightarrow M(\text{Adr})$				
ROR n, SP	Indexed	4E	2	4	$M(n+\text{SP}) \gg 1 \rightarrow M(n+\text{SP})$				
ROR n, X	Indexed	5E	2	4	$M(n+X) \gg 1 \rightarrow M(n+X)$				
ROR A, X	Indexed	6E	1	4	$M(A+X) \gg 1 \rightarrow M(A+X)$				
ROR n, Y	Indexed	7E	2	4	$M(n+Y) \gg 1 \rightarrow M(n+Y)$				
ROR A, Y	Indexed	8E	1	4	$M(A+Y) \gg 1 \rightarrow M(A+Y)$				

RTS Return from Subroutine

RTN	$M(\text{SP}) \rightarrow \text{PC}; \text{SP}+1 \rightarrow \text{SP}$
Flaggor	Påverkas ej
Beskrivning	Återhopp från en subrutin utförs genom att översta dataordet på stacken, dvs återhoppadressen, läses och placeras i PC. Stackpekaren uppdateras sedan

Detaljer:

Instruktion RTS	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
RTS	Inherent	43	1	2	$M(\text{SP}) \rightarrow \text{PC}$ $\text{SP}+1 \rightarrow \text{SP}$	-	-	-	-

RTI Return from Interrupt

Varianter	
RTN	$M(\text{SP}) \rightarrow \text{CC}; \text{SP}+1 \rightarrow \text{SP},$ $M(\text{SP}) \rightarrow \text{A}; \text{SP}+1 \rightarrow \text{SP},$ $M(\text{SP}) \rightarrow \text{X}; \text{SP}+1 \rightarrow \text{SP},$ $M(\text{SP}) \rightarrow \text{Y}; \text{SP}+1 \rightarrow \text{SP},$ $M(\text{SP}) \rightarrow \text{PC}; \text{SP}+1 \rightarrow \text{SP}$
Flaggor	Påverkas ej
Beskrivning	Återhopp från undantagshantering utförs genom att samtliga register återställs från stacken och stackpekaren uppdateras.

Detaljer:

Instruktion RTI Variant	Adressering				Operation	Flaggor				
	metod	OP	#	~		I	N	Z	V	C
RTI	Inherent	44	1	6	$M(\text{SP}) \rightarrow \text{CC}; \text{SP}+1 \rightarrow \text{SP},$ $M(\text{SP}) \rightarrow \text{A}; \text{SP}+1 \rightarrow \text{SP},$ $M(\text{SP}) \rightarrow \text{X}; \text{SP}+1 \rightarrow \text{SP},$ $M(\text{SP}) \rightarrow \text{Y}; \text{SP}+1 \rightarrow \text{SP},$ $M(\text{SP}) \rightarrow \text{PC}; \text{SP}+1 \rightarrow \text{SP}$	Δ	Δ	Δ	Δ	Δ

SBCA Subtract with borrow Data from register A

RTN	$A - M - C \rightarrow A$, M = data från minneadress 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 åttabits subtraktion av dataordet i minnet från innehållet i register A, resultatet placeras i register A. 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.

Detaljer:

Instruktion SBC	Adressering				Operation	Flaggor			
Variant	metod	OP	#	~		N	Z	V	C
SBCA #Data	Immediate	93	2	4	$A - \text{Data} - C \rightarrow A$	Δ	Δ	Δ	Δ
SBCA Adr	Absolute	A3	2	5	$A - M(\text{Adr}) - C \rightarrow A$				
SBCA n, SP	Indexed	B3	2	5	$A - M(n + SP) - C \rightarrow A$				
SBCA n, X	Indexed	C3	2	5	$A - M(n + X) - C \rightarrow A$				
SBCA n, Y	Indexed	D3	2	5	$A - M(n + Y) - C \rightarrow A$				

ST Store register into memory

RTN	$R \rightarrow M$
Flaggor	N: Påverkas ej. Z: Påverkas ej. V: Påverkas ej. C: Påverkas ej
Beskrivning	Lagrar angivet registerinnehåll i minnet på den effektiva adressen

Detaljer:

Instruktion ST	Adressering				Operation	Flaggor			
Variant	metod	OP	#	~		N	Z	V	C
STA Adr	Absolute	E1	2	3	$A \rightarrow M(\text{Adr})$	-	-	-	-
STA n, SP	Indexed	E2	2	3	$A \rightarrow M(n + SP)$				
STA n, X	Indexed	E3	2	3	$A \rightarrow M(n + X)$				
STA A, X	Indexed	E4	1	3	$A \rightarrow M(A + X)$				
STA , X+	Indexed	E5	1	4	$A \rightarrow M(X)$, $X + 1 \rightarrow X$				
STA , X-	Indexed	E6	1	4	$A \rightarrow M(X)$, $X - 1 \rightarrow X$				
STA , +X	Indexed	E7	1	4	$X + 1 \rightarrow X$, $A \rightarrow M(X)$				
STA , -X	Indexed	E8	1	4	$X - 1 \rightarrow X$, $A \rightarrow M(X)$				
STA n, Y	Indexed	E9	2	3	$A \rightarrow M(n + Y)$				
STA A, Y	Indexed	EA	1	3	$A \rightarrow M(A + Y)$				
STA , Y+	Indexed	EB	1	4	$A \rightarrow M(Y)$, $Y + 1 \rightarrow Y$				
STA , Y-	Indexed	EC	1	4	$A \rightarrow M(Y)$, $Y - 1 \rightarrow Y$				
STA , +Y	Indexed	ED	1	4	$Y + 1 \rightarrow Y$, $A \rightarrow M(Y)$				
STA , -Y	Indexed	EE	1	4	$X - 1 \rightarrow X$, $A \rightarrow M(X)$				
STX Adr	Absolute	30	2	3	$X \rightarrow M(\text{Adr})$				
STX n, SP	Indexed	40	2	3	$X \rightarrow M(n + SP)$				
STX n, X	Indexed	50	2	3	$X \rightarrow M(n + X)$				
STX A, X	Indexed	60	2	3	$X \rightarrow M(A + X)$				
STX n, Y	Indexed	70	2	3	$X \rightarrow M(n + Y)$				
STX A, Y	Indexed	80	2	3	$X \rightarrow M(A + Y)$				
STY Adr	Absolute	31	2	3	$Y \rightarrow M(\text{Adr})$				
STY n, SP	Indexed	41	2	3	$Y \rightarrow M(n + SP)$				
STY n, X	Indexed	51	2	3	$Y \rightarrow M(n + X)$				
STY A, X	Indexed	60	2	3	$Y \rightarrow M(A + X)$				
STY n, Y	Indexed	71	2	3	$Y \rightarrow M(n + Y)$				
STY A, Y	Indexed	81	2	3	$Y \rightarrow M(A + Y)$				
STSP Adr	Absolute	32	2	3	$SP \rightarrow M(\text{Adr})$				
STSP n, SP	Indexed	42	2	3	$SP \rightarrow M(n + SP)$				
STSP n, X	Indexed	52	2	3	$SP \rightarrow M(n + X)$				
STSP A, X	Indexed	62	2	3	$SP \rightarrow M(A + X)$				
STSP n, Y	Indexed	72	2	3	$SP \rightarrow M(n + Y)$				
STSP A, Y	Indexed	82	2	3	$SP \rightarrow M(A + Y)$				

SUBA Subtract data from register A

RTN	A – Opr → A
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 operanden från innehållet i register A. Resultatet placeras i register A. 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 A – M utförs på det traditionella sättet A + M' + 1

Detaljer:

Instruktion SUBA	Adressering				Operation	Flaggor			
Variant	metod	OP	#	~		N	Z	V	C
SUBA #Data	Immediate	94	2	4	A–Data → A	Δ	Δ	Δ	Δ
SUBA Adr	Absolute	A4	2	5	A–M(Adr) → A				
SUBA n, SP	Indexed	B4	1	5	A – M(n+SP) → A				
SUBA n, X	Indexed	C4	2	5	A – M(n+X) → A				
SUBA n, Y	Indexed	D4	2	5	A – M(n+Y) → A				

TFR Transfer register to register

Varianter	
RTN	R1 → R2
Flaggor	Påverkas ej såvida man inte flyttar ett registerinnehåll till CC-registret
Beskrivning	Data kopieras mellan angivna register

Detaljer:

Instruktion TFR	Adressering				Operation	Flaggor			
Variant	metod	OP	#	~		N	Z	V	C
TFR A, CC	Inherent	18	1	2	A → CC	Δ	Δ	Δ	Δ
TFR CC, A	Inherent	19	1	2	CC → A	-	-	-	-
TFR X, Y	Inherent	1A	1	2	X → Y	-	-	-	-
TFR Y, X	Inherent	1B	1	2	Y → X	-	-	-	-
TFR X, SP	Inherent	1C	1	2	X → SP	-	-	-	-
TFR SP, X	Inherent	1D	1	2	SP → X	-	-	-	-
TFR Y, SP	Inherent	1E	1	2	Y → SP	-	-	-	-
TFR SP, Y	Inherent	1F	1	2	SP → Y	-	-	-	-

TST Test

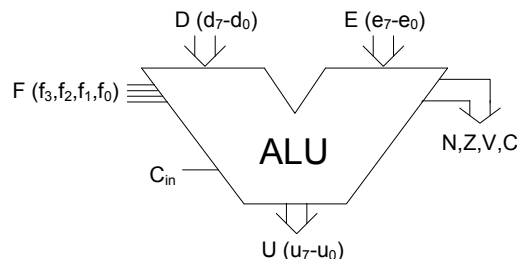
RTN	A – 0 eller M – 0 M = data från minnesadress
Flaggor	N: Ettställs om resultatets teckenbit (bit 7) får värdet 1. Z: Ettställs om samtliga åtta bitar i resultatet blir noll. V: Nollställs. C: Nollställs
Beskrivning	Låter datavärdet i A eller M passera ALU:n och sätter flaggvipporna N och Z så att man kan avgöra datavärdets tecken eller om det är noll. Endast flaggvipporna påverkas.

Detaljer:

Instruktion TST	Adressering				Operation	Flaggor			
Variant	metod	OP	#	~		N	Z	V	C
TSTA	Inherent	09	1	2	A–0	Δ	Δ	0	0
TST Adr	Absolute	39	2	3	M(Adr)–0				
TST n, SP	Indexed	49	1	3	M(n+SP) – 0				
TST n, X	Indexed	59	1	3	M(n+X) – 0				
TST A, X	Indexed	69	1	3	M(A+X) – 0				
TST n, Y	Indexed	79	1	3	M(n+Y) – 0				
TST A, Y	Indexed	89	1	3	M(A+Y) – 0				

ALU-funktioner

Flisp ALU (Arithmetic and Logic Unit) arbetar med 8-bitars operander (D och E) samt 1-bits operand C_{in} . Resultatet består av 8-bitar (U) och fyra "flaggbitar", N,Z,V och C.



U(u ₇ -u ₀)	Utsignal, 8 bitar, resultatet av ALU:ns operation
D(d ₇ -d ₀)	Insignal, 8 bitar, operand 1
E(e ₇ -e ₀)	Insignal, 8 bitar, operand 2
C _{in}	Insignal, 1 bit, operand 3
F = (f ₃ , f ₂ , f ₁ , f ₀).	Bestämmer den operation som utförs av ALU:n
N	Teckenflaggan är identisk med den mest signifikanta biten (teckenbiten) av utsignalen U från ALU:n
Z	Nollflaggan, visar om en ALU-operation ger värdet noll som resultat på U-utgången
V	Spillflaggan visar om en aritmetisk operation ger "overflow" enligt reglerna för 2-komplementaritmetik. V-flaggans värde är 0 vid andra operationer än aritmetiska
C	Carryflaggan innehåller minnessiffran ut (carry-out) från den mest signifikanta bitpositionen (längst till vänster) då en aritmetisk operation utförs av ALU:n. Vid subtraktion gäller för denna ALU att C = 1 om lånesiffra (borrow) uppstår och C = 0 om lånesiffra inte uppstår. Carryflaggans värde är 0 vid andra operationer än aritmetiska

Funktionstabell för ALU

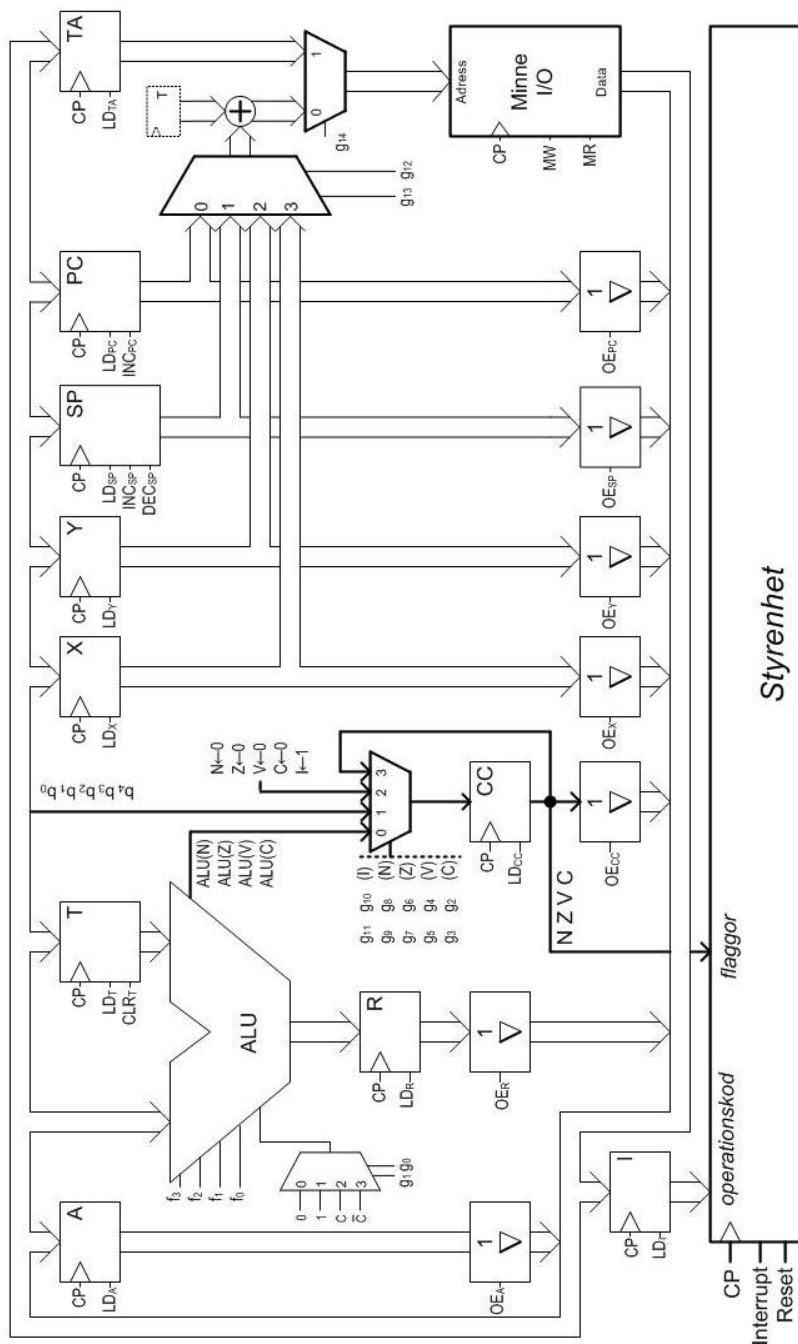
funktion				operation				flaggor			
f ₃	f ₂	f ₁	f ₀	RTN				N	Z	V	C
0	0	0	0	U=0				0	1	0	0
0	0	0	1	U=FD ₁₆				1	0	0	0
0	0	1	0	U=FE ₁₆				1	0	0	0
0	0	1	1	U=FF ₁₆				1	0	0	0
0	1	0	0	U=E				u ₇	⁽¹⁾	0	0
0	1	0	1	U=D _{1k} + C _{in}				u ₇	⁽¹⁾	⁽⁸⁾	⁽⁷⁾
0	1	1	0	U=D∨E				u ₇	⁽¹⁾	0	0
0	1	1	1	U=D∧E				u ₇	⁽¹⁾	0	0
1	0	0	0	U=D⊕E				u ₇	⁽¹⁾	0	0
1	0	0	1	U=D + C _{in}				u ₇	⁽¹⁾	⁽²⁾	⁽³⁾
1	0	1	0	U=D + FF ₁₆				u ₇	⁽¹⁾	⁽²⁾	⁽³⁾
1	0	1	1	U=D + E + C _{in}				u ₇	⁽¹⁾	⁽²⁾	⁽³⁾
1	1	0	0	U=D + E _{1k} + C _{in}				u ₇	⁽¹⁾	⁽²⁾	⁽³⁾
1	1	0	1	U=D<<1 (C _{in})				u ₇	⁽¹⁾	⁽⁶⁾	⁽⁴⁾
1	1	1	0	U=(C _{in}) D >> 1				u ₇	⁽¹⁾	⁽⁶⁾	⁽⁵⁾
1	1	1	1	U=(d ₇) D >> 1				u ₇	⁽¹⁾	0	⁽⁵⁾

Anm:

D_{1k} = 1-komplement D = D'

- (1) $Z = \overline{u_7} \wedge \overline{u_6} \wedge \overline{u_5} \wedge \overline{u_4} \wedge \overline{u_3} \wedge \overline{u_2} \wedge \overline{u_1} \wedge \overline{u_0}$,
dvs. Z=1 då samtliga bitar i register U är 0, Z=0 annars.
- (2) $V = (\overline{u_7} \wedge d_7 \wedge e_7) \vee (u_7 \wedge \overline{d_7} \wedge \overline{e_7})$,
dvs. V-flaggan sätts enligt reglerna för tvåkomplementsaritmetik.
- (3) C = c₈, dvs. carry ut från additionen av de mest signifikanta siffrorna.
- (4) C = utskiftad bit, dvs. bit d₇ före vänsterskiftet.
- (5) C = utskiftad bit, dvs. bit d₀ före högerskiftet.
- (6) $V = C_{in} \oplus d_7$ vid vänsterskift, $d_7 \oplus d_6$ vid högerskift skift, dvs. sätts till 1 om skiftet föranleder teckenbyte.
- (7) C nollställs om D=0, C ettställs annars.
- (8) V ettställs om D=(10000000)₂, V nollställs annars.

Dataväg, styrenhet och minne



RTN (register transfer notation)

Operationsbeskrivningar

n	Konstant uttryckt i talbas 10
N _r	Konstanten N uttryckt i talbasen r.
EA	Effektiv adress
Opr	Operand, data på effektiv adress
M(Adr)	Minnesinnehåll på adressen "Adr"
→	Kopiering

Följande symboler är beteckningar som reserverats för FLISP register

A	Ackumulator A
X	Adressregister X
Y	Adressregister Y
CC	Flaggregister (Condition Codes)
PC	Programräknare (Program Counter)
SP	Stackpekare (Stack Pointer)
T	Temporärregister
R	Resultatregister
TA	Temporärt adressregister

Följande symboler är beteckningar som reserverats för villkorsindikatorer (flaggor).

N	Teckenflaggan ("Negative")
Z	Nollflaggan ("Zero")
V	Overflowflaggan
C	Carryflaggan

Operatorer

+	Addition
-	Subtraktion
^	Logiskt "OCH" (AND)
∨	Logiskt "ELLER" (OR)
⊕	Logiskt "EXKLUSIVT ELLER" (XOR)
Opr<<1 (d)	"Opr" skiftas vänster ett steg. Biten d skiftas in i den minst signifikanta positionen.
(d) 1>>Opr	"Opr" skiftas höger. Biten d skiftas in i den mest signifikanta positionen.
Opr'	Bitvis komplementering av operanden "Opr"

Relationsoperatorer

=	likhet
!=	olikhet
>	större än
<	mindre än
>=	större än eller likhet
<=	mindre än eller likhet

Signaler i fast styrenhet

Signal	Funktion
<i>Insignaler till styrenheten</i>	
CP	"Clock Pulse", systemklocka
Reset	Asynkron återställningssignal
Interrupt	Extern avbrottsignal, samplas vid CP (positiv flank), kontrolleras vid NF
Q _{state}	Tillstånd, "state" anges på hexadecimal form med siffrorna 0 t.o.m F, dvs totalt 16 olika tillstånd
I _{opcode}	Operationskod, anger innehållet i instruktionsregistret avkodat av en "en av 256-väljare". "opcode" anges på hexadecimal form, dvs 0 t.o.m FF.
N-flagga	N-flaggan, dvs. bit 3 i CC
Z-flagga	Z-flaggan, dvs. bit 2 i CC
V-flagga	V-flaggan, dvs. bit 1 i CC
C-flagga	C-flaggan, dvs. bit 0 i CC
<i>Utsignaler från styrenheten</i>	
LD _A	"Load register A enable", Ackumulator
LD _I	"Load register I enable", instruktionsregister
OE _A	"Output enable register A", Ackumulator
LD _T	"Load register T enable", Temporärregister
CLR _T	"Clear register T", Temporärregister
LD _R	"Load register R enable", Resultatregister
OE _R	"Output enable register R", Resultatregister
LD _{CC}	"Load register CC enable", Flaggregister (Condition Codes)
OE _{CC}	"Output enable register CC", Flaggregister (Condition Codes)
LD _X	"Load register X enable", Adressregister
OE _X	"Output enable register X", Flaggregister
LD _Y	"Load register Y enable", Adressregister
OE _Y	"Output enable register Y", Flaggregister
LD _{PC}	"Load register PC enable", Programräknare
OE _{PC}	"Output enable register PC", Programräknare
INC _{PC}	"Increment register PC", Addera 1 till programräknare
LD _{SP}	"Load register SP enable", Stackpekare
OE _{SP}	"Output enable register SP", Stackpekare
INC _{SP}	"Increment register PC", Addera 1 till stackpekare
DEC _{SP}	"Decrement register SP", Subtrahera 1 från stackpekare
LD _{TA}	"Load register TA enable", Adressberäkningsregister
MR	"Memory Read", Avkoda adressbuss och klocka ut data från minne/"Input"
MW	"Memory Write", Avkoda adressbuss och klocka in data till minne/"Output"
f ₃ ..f ₀	Funktionsväljare för ALU (se sidan 44)
g ₁₄ ..g ₀	Styrsignaler för väljarfunktioner (se sidan 48)

Dessutom genereras internt signalen NF (New Fetch) för att initiera en ny hämtfas.

Styrsignaler för väljarfunktioner

Val av flaggsättning i CC

g ₃ g ₂	C väljs enligt:	RTN
0 0	C tas från ALU:n	ALU(C) → C
0 1	C tas från bit 0 på bussen	b ₀ → C
1 0	Återställning (nollställning) av C	0 → C
1 1	C återförs (ändras ej)	

g ₇ g ₆	Z väljs enligt:	RTN
0 0	Z tas från ALU:n	ALU(Z) → Z
0 1	Z tas från bit 2 på bussen	b ₂ → Z
1 0	Återställning (nollställning) av Z	0 → Z
1 1	Z återförs (ändras ej)	

g ₁₁ g ₁₀	I väljs enligt:	RTN
0 0	I återförs (ändras ej)	
0 1	I tas från bit 4 på bussen	b ₄ → I
1 0	I återställs (ettställs)	1 → I
1 1	I återförs (ändras ej)	

Val av Carry in till ALU

g ₁ g ₀	Signal till C _{in}	RTN
0 0	konstant värde 0	0 → C _{in}
0 1	konstant värde 1	1 → C _{in}
1 0	C-flagga från CC	C → C _{in}
1 1	C-flagga invers från CC	C' → C _{in}

Val av register för adressering av minne/IO

g ₁₄ g ₁₃ g ₁₂	Register till adressbuss:	RTN
0 0 0	Register PC	M(PC)
0 0 1	Register SP ("bas") och register T ("offset")	M(T+SP)
0 1 0	Register Y ("bas") och register T ("offset")	M(T+Y)
0 1 1	Register X ("bas") och register T ("offset")	M(T+X)
1 0 0	Adressberäkningsregister, ingen offset	M(TA)
1 0 1	Adressberäkningsregister, ingen offset	M(TA)
1 1 0	Adressberäkningsregister, ingen offset	M(TA)
1 1 1	Adressberäkningsregister, ingen offset	M(TA)