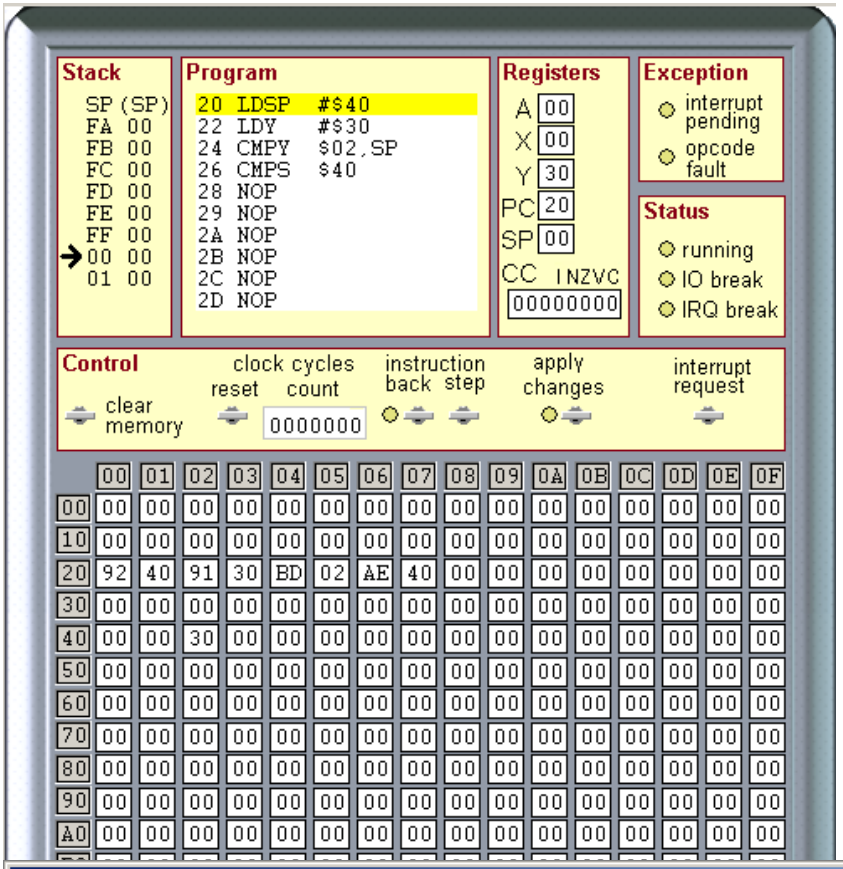
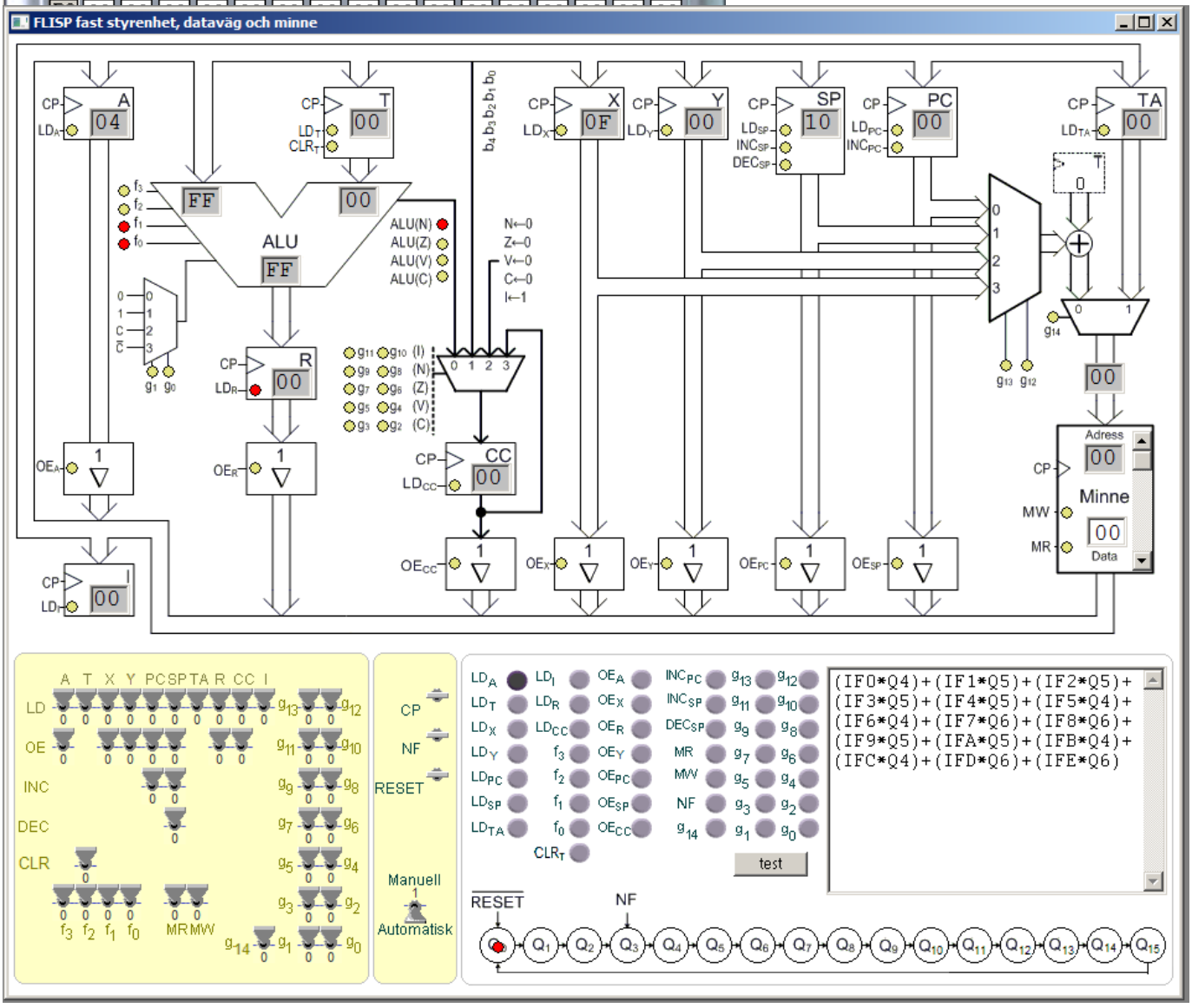
2014-08-19

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)

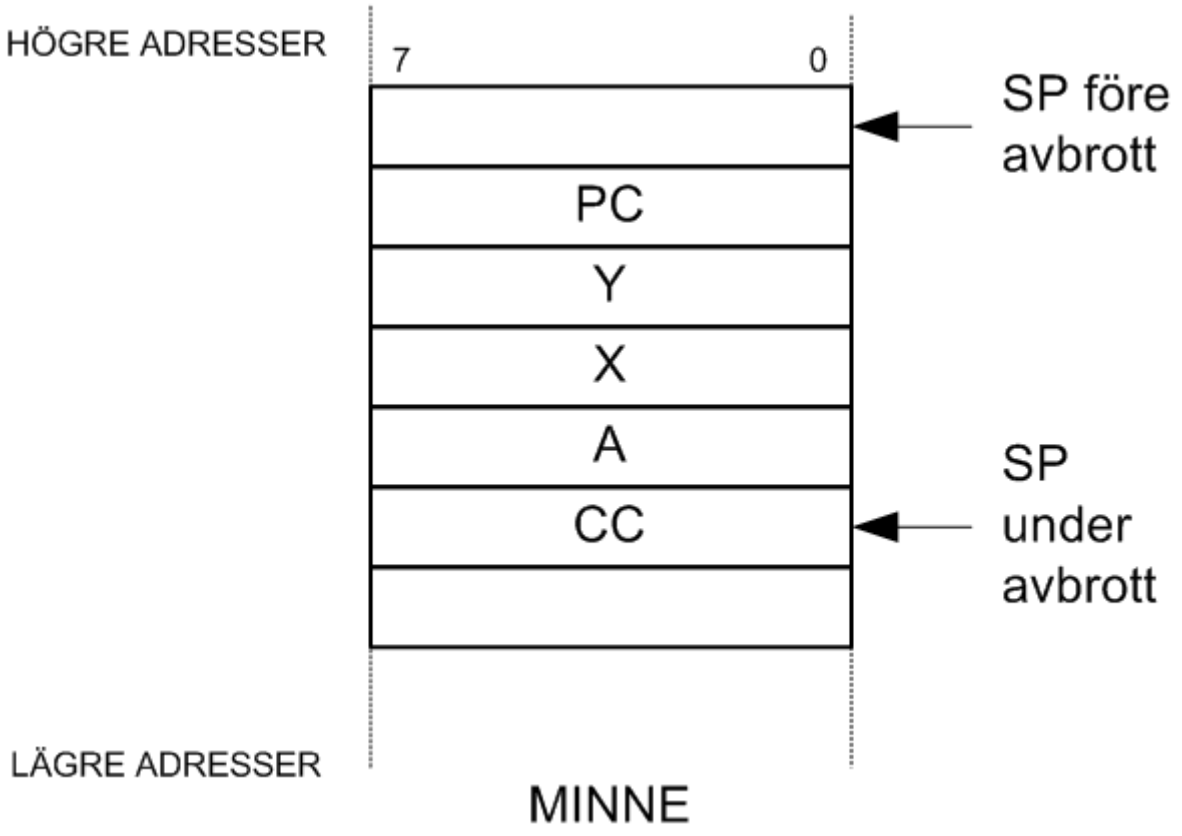
Del II: Mikroarkitekturen

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"

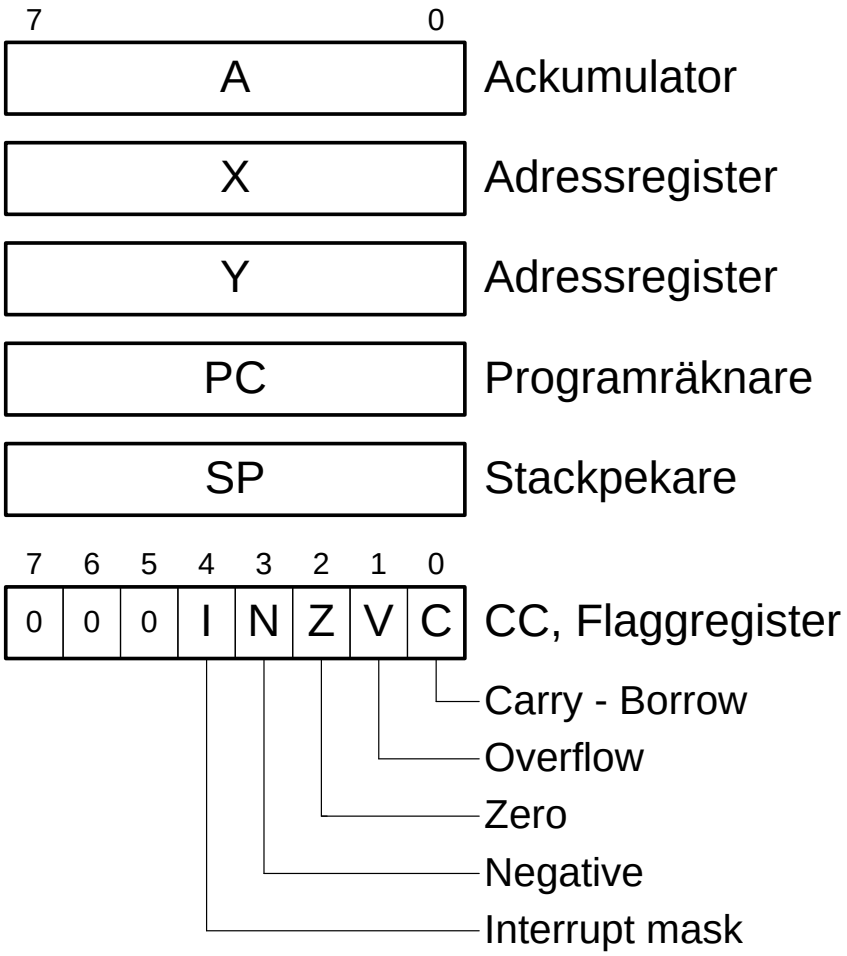


Vid "IRQ" pekar PC på den instruktion som skulle ha utförts om inget avbrott signalerats.

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

Registeruppsättning

Det finns 6 register som är direkt åtkomliga för assemblerprogrammeraren.



Ackumulator: 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

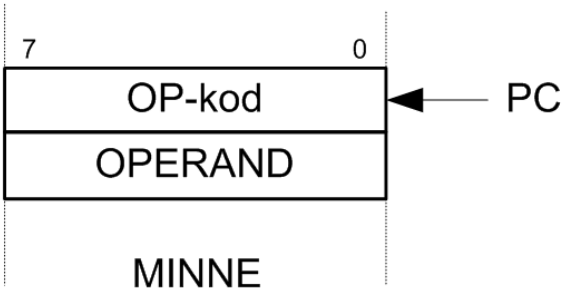
Totalt används 8 distinkta adresseringssätt.

Inherent [ih]

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

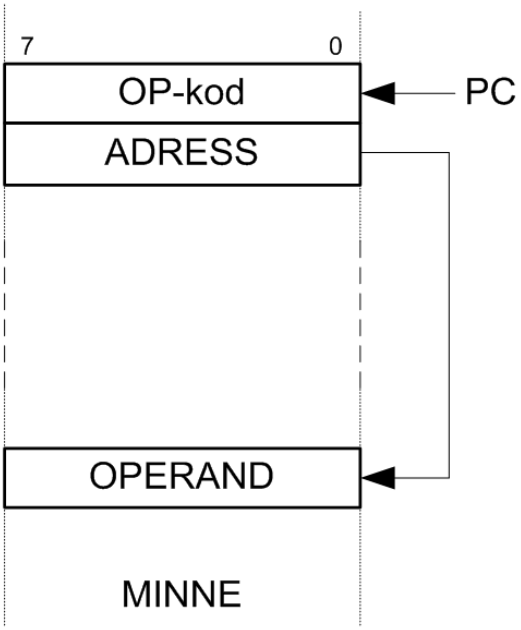
Omedelbar (Immediate) [im]

Operanden är kodad tillsammans med operationen.
RTN: M(PC+1)
Assemblersyntax: #<DATA>



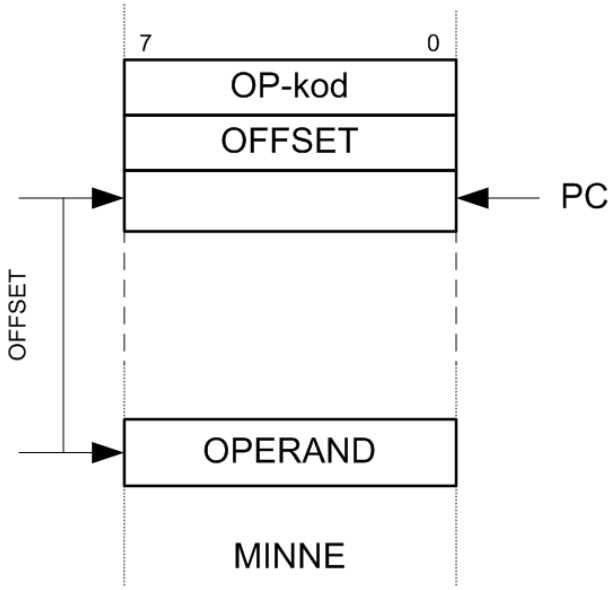
Absolut (Absolute) [ab]

Operanden finns i minnet. Minnesadressen är kodad tillsammans med operationen.
RTN: [M(PC+1)]
Assemblersyntax: <ADRESS>



Relativ [pc]

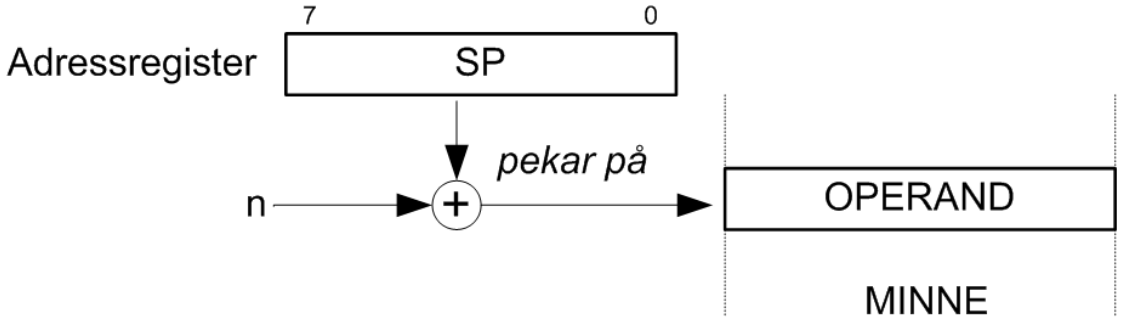
PC-relativ adressering används av programflödesinstruktioner (”branch on condition”). Adressen till operanden bestäms av en offset till aktuell PC, därav namnet ”Relativ”.



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

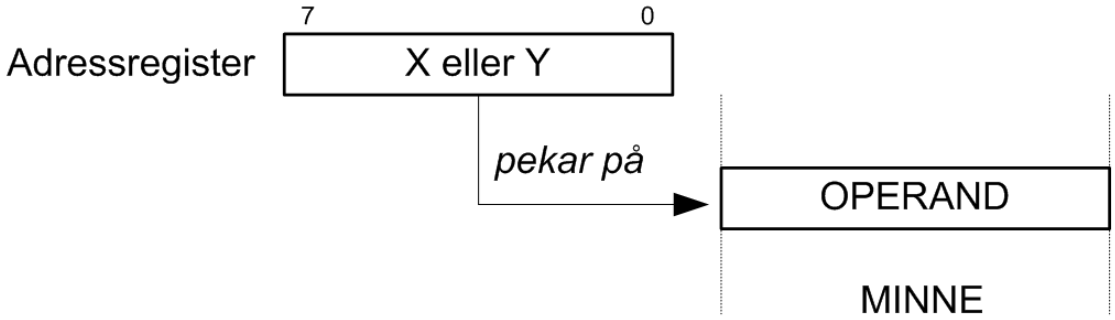


Konstanten **n** (8 bitar) adderas modulo 256 och kan anges med eller utan tecken.

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

Indirekt register med konstant offset (Indexed) [nR]

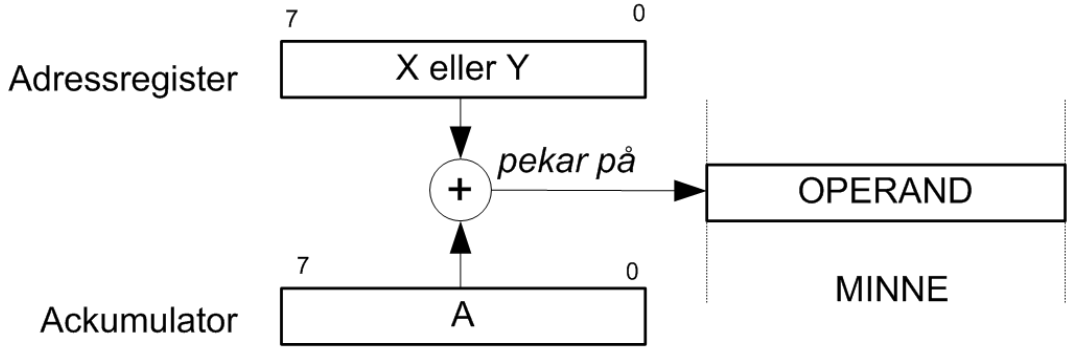
Operanden finns i minnet. En konstant n (”offset”) adderas till innehållet i register R (X eller Y) och bildar adressen till operandens plats i minnet.
RTN: M(R+n);
Assemblersyntax: `n,R`



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

Indirekt register med ackumulator 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`

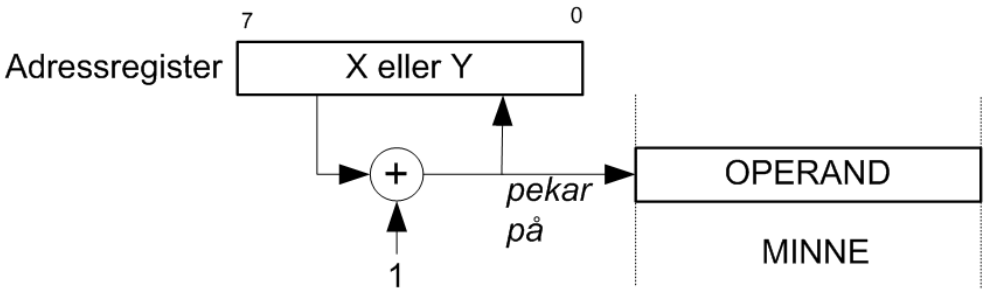


Indirekt register med pre/post decrement/increment (Indexed)

Pre auto increment [+R]

RTN: R+1→R; M(R)

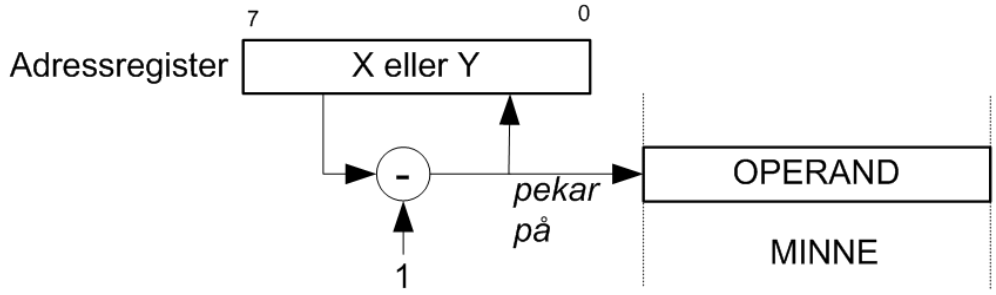
Assemblersyntax: ,+R



Pre auto decrement [–R]

RTN: R-1→R; M(R)

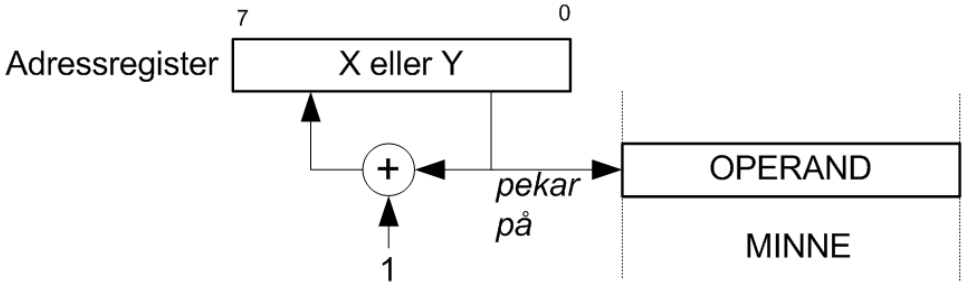
Assemblersyntax: ,-R



Post auto increment [R+]

RTN: M(R); R+1→R

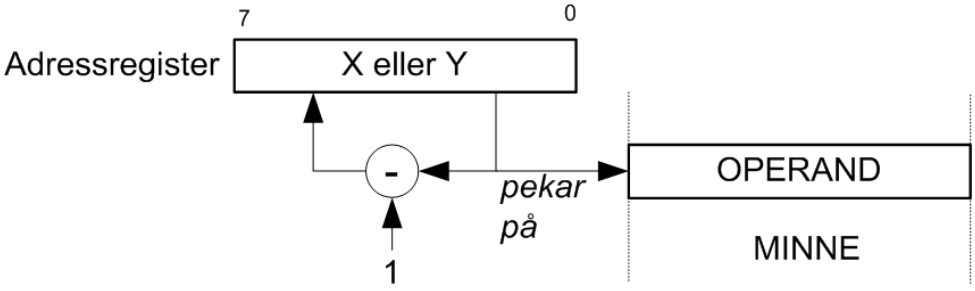
Assemblersyntax: ,R+



Post auto decrement [R-]

RTN: M(R); R-1→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
```

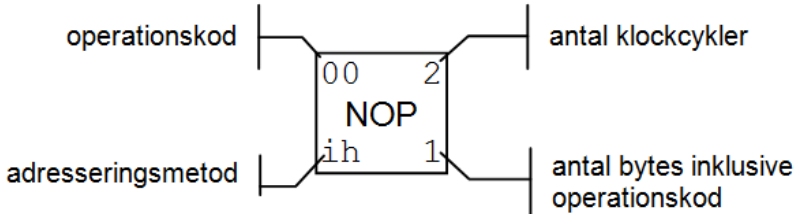
”Stack operations”

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

”Misc.”

```
NOP
```

Karta över operationskoder



Anm:
“Blanka” fält utgör ”otillåten operationskod” och föranleder undantagshantering

00 2 NOP ih 1	10 3 PSHA ih 1	20 5 BSR pc 2	30 3 STX ab 2	40 3 STX ns 2	50 3 STX nx 2	60 3 STX ax 1	70 3 STX ny 2	80 3 STX ay 1	90 2 LDX im 2	A0 3 LDX ab 2	B0 3 LDX ns 2	C0 3 LDX nx 2	D0 3 LDX ny 2	E0	F0 2 LDA im 2
01 4 ANDCC im 2	11 3 PSHX ih 1	21 4 BRA pc 2	31 3 STY ab 2	41 3 STY ns 2	51 3 STY nx 2	61 3 STY ax 1	71 3 STY ny 2	81 3 STY ay 1	91 2 LDY im 2	A1 3 LDY ab 2	B1 3 LDY ns 2	C1 3 LDY nx 2	D1 3 LDY ny 2	E1 3 STA ab 2	F1 3 LDA ab 2
02 4 ORCC im 2	12 3 PSHY ih 1	22 4 BMI pc 2	32 3 STSP ab 2	42 3 STSP ns 2	52 3 STSP nx 2	62 3 STSP ax 1	72 3 STSP ny 2	82 3 STSP ay 1	92 2 LDSP im 2	A2 3 LDSP ab 2	B2 3 LDSP ns 2	C2 3 LDSP nx 2	D2 3 LDSP ny 2	E2 3 STA ns 2	F2 3 LDA ns 2
03	13 3 PSHCC ih 1	23 4 BPL pc 2	33 2 JMP ab 2	43 2 RTS ih 1	53 4 JMP nx 2	63 4 JMP ax 1	73 4 JMP ny 2	83 4 JMP ay 1	93 4 SBCA im 2	A3 5 SBCA ab 2	B3 5 SBCA ns 2	C3 5 SBCA nx 2	D3 5 SBCA ny 2	E3 3 STA nx 2	F3 3 LDA nx 2
04	14 3 PULA ih 1	24 4 BEQ pc 2	34 4 JSR ab 2	44 6 RTI ih 1	54 5 JSR nx 2	64 5 JSR ax 1	74 5 JSR ny 2	84 5 JSR ay 1	94 4 SUBA im 2	A4 5 SUBA ab 2	B4 5 SUBA ns 2	C4 5 SUBA nx 2	D4 5 SUBA ny 2	E4 3 STA ax 1	F4 3 LDA ax 1
05 3 CLRA ih 1	15 3 PULX ih 1	25 4 BNE pc 2	35 3 CLR ab 2	45 3 CLR ns 2	55 3 CLR nx 2	65 3 CLR ax 1	75 3 CLR ny 2	85 3 CLR ay 1	95 4 ADCA im 2	A5 5 ADCA ab 2	B5 5 ADCA ns 2	C5 x ADCA nx 2	D5 5 ADCA ny 2	E5 4 STA x+ 1	F5 4 LDA x+ 1
06 3 NEGA ih 1	16 3 PULY ih 1	26 4 BVS pc 2	36 4 NEG ab 2	46 4 NEG ns 2	56 4 NEG nx 2	66 4 NEG ax 1	76 4 NEG ny 2	86 4 NEG ay 1	96 4 ADDA im 2	A6 5 ADDA ab 2	B6 5 ADDA ns 2	C6 5 ADDA nx 2	D6 5 ADDA ny 2	E6 4 STA x- 1	F6 4 LDA x- 1
07 3 INCA ih 1	17 3 PULCC ih 1	27 4 BVC pc 2	37 4 INC ab 2	47 4 INC ns 2	57 4 INC nx 2	67 4 INC ax 1	77 4 INC ny 2	87 4 INC ay 1	97 3 CMPA im 2	A7 4 CMPA ab 2	B7 4 CMPA ns 2	C7 4 CMPA nx 2	D7 4 CMPA ny 2	E7 4 STA +x 1	F7 4 LDA +x 1
08 3 DECA ih 1	18 2 TFR A,C ih 1	28 4 BCS pc 2	38 4 DEC ab 2	48 4 DEC ns 2	58 4 DEC nx 2	68 4 DEC ax 1	78 4 DEC ny 2	88 4 DEC ay 1	98 3 BITA im 2	A8 4 BITA ab 2	B8 4 BITA ns 2	C8 4 BITA nx 2	D8 4 BITA ny 2	E8 4 STA -x 1	F8 4 LDA -x 1
09 2 TSTA ih 1	19 2 TFR C,A ih 1	29 4 BCC pc 2	39 3 TST ab 2	49 3 TST ns 2	59 3 TST nx 2	69 3 TST ax 1	79 3 TST ny 2	89 3 TST ay 1	99 4 ANDA im 2	A9 5 ANDA ab 2	B9 5 ANDA ns 2	C9 5 ANDA nx 2	D9 5 ANDA ny 2	E9 3 STA ny 2	F9 3 LDA ny 2
0A 3 COMA ih 1	1A 2 TFR X,Y ih 1	2A 4 BHI pc 2	3A 4 COM ab 2	4A 4 COM ns 2	5A 4 COM nx 2	6A 4 COM ax 1	7A 4 COM ny 2	8A 4 COM ay 1	9A 4 ORA im 2	AA 5 ORA ab 2	BA 5 ORA ns 2	CA 5 ORA nx 2	DA 5 ORA ny 2	EA 3 STA ay 1	FA 3 LDA ay 1
0B 3 LSLA ih 1	1B 2 TFR Y,X ih 1	2B 4 BLS pc 2	3B 4 LSL ab 2	4B 4 LSL ns 2	5B 4 LSL nx 2	6B 4 LSL ax 1	7B 4 LSL ny 2	8B 4 LSL ay 1	9B 4 EORA im 2	AB 5 EORA ab 2	BB 5 EORA ns 2	CB 5 EORA nx 2	DB 5 EORA ny 2	EB 4 STA y+ 1	FB 4 LDA y+ 1
0C 3 LSRA ih 1	1C 2 TFR X,S ih 1	2C 4 BGT pc 2	3C 4 LSR ab 2	4C 4 LSR ns 2	5C 4 LSR nx 2	6C 4 LSR ax 1	7C 4 LSR ny 2	8C 4 LSR ay 1	9C 3 CMPX im 2	AC 4 CMPX ab 2	BC 4 CMPX ns 2	CC 4 LEAX nx 2	DC 4 LEAX ns 2	EC 4 STA y- 1	FC 4 LDA y- 1
0D 3 ROLA ih 1	1D 2 TFR S,X ih 1	2D 4 BGE pc 2	3D 4 ROL ab 2	4D 4 ROL ns 2	5D 4 ROL nx 2	6D 4 ROL ax 1	7D 4 ROL ny 2	8D 4 ROL ay 1	9D 3 CMPY im 2	AD 4 CMPY ab 2	BD 4 CMPY ns 2	CD 4 LEAY ny 2	DD 4 LEAY ns 2	ED 4 STA +y 1	FD 4 LDA +y 1
0E 3 RORA ih 1	1E 2 TFR Y,S ih 1	2E 4 BLE pc 2	3E 4 ROR ab 2	4E 4 ROR ns 2	5E 4 ROR nx 2	6E 4 ROR ax 1	7E 4 ROR ny 2	8E 4 ROR ay 1	9E 3 CMPSP im 2	AE 4 CMPSP ab 2	BE 4 LEASP ns 2	CE 4 LEASP nx 2	DE 4 LEASP ny 2	EE 4 STA -y 1	FE 4 LDA -y 1
0F 3 ASRA ih 1	1F 2 TFR S,Y ih 1	2F 4 BLT pc 2	3F 4 ASR ab 2	4F 4 ASR ns 2	5F 4 ASR nx 2	6F 4 ASR ax 1	7F 4 ASR ny 2	8F 4 ASR ay 1	9F 4 EXG A,C ih 1	AF 4 EXG X,Y ih 1	BF 4 EXG X,S ih 1	CF 4 EXG Y,S ih 1	DF	EF	FF

Detaljerad beskrivning av FLIS-processorns instruktioner

Instruktion **Add with Carry** Variant	Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
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	Ackumulator 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 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<<*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 + Opr + C → 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 metod	OP	#	~	Operation	Flaggor N	Z	V	C
ADCA	#Data	Immediate	95	2	4	A + Data+C → A	Δ	Δ	Δ	Δ
ADCA	Adr	Absolute	A5	2	5	A + M(Adr)+C → A				
ADCA	n,SP	Indexed	B5	2	5	A + M(n+SP) + C → A				
ADCA	n,X	Indexed	C5	2	5	A + M(n+X) + C → A				
ADCA	n,Y	Indexed	D5	2	5	A + M(n+Y) + C → A				

ADDA Add data into register A

RTN	A + Opr → 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 metod	OP	#	~	Operation	Flaggor N	Z	V	C
ADDA	#Data	Immediate	96	2	4	A + Data → A	Δ	Δ	Δ	Δ
ADDA	Adr	Absolute	A6	2	5	A + M(Adr) → A				
ADDA	n,SP	Indexed	B6	2	5	A + M(n+SP) → A				
ADDA	n,X	Indexed	C6	2	5	A + M(n+X) → A				
ADCA	n,Y	Indexed	D5	2	5	A + M(n+Y) + C → A				

ANDA Logical AND data into register A

RTN	A $\wedge$ 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 metod	OP	#	~	Operation	Flaggor N	Z	V	C
ANDA #Data	Immediate	99	2	4	A $\wedge$ Data $\rightarrow$ A	Δ	Δ	0	-
ANDA Adr	Absolute	A9	2	5	A $\wedge$ M(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				

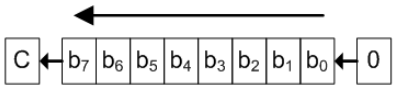
ANDCC Logical AND data into register CC

RTN	CC $\wedge$ 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 metod	OP	#	~	Operation	Flaggor I	N	Z	V	C
ANDCC #Data	Immediate	01	2	4	CC $\wedge$ Data $\rightarrow$ CC	Δ	Δ	Δ	Δ	Δ

ASL Arithmetic shift left

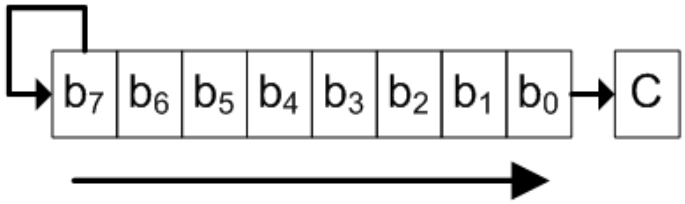
RTN	A <<1 → A eller M(EA) <<1 → M(EA)
Flaggor	N: Kopia av bit 7 efter skiftet. Z: Ettställs om samtliga åtta bitar i resultatet blir noll. V: Ettställs om C och bit 7 är olika efter operationen, dvs overflow vid 2-komplements-representation inträffar. C: bit 7 före skiftet blir ny carrybit efter skiftet.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

Detaljer:

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

ASR Arithmetic shift right

RTN	A >>1 → A 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: 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** Variant	Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
ASRA	Inherent	0F	1	3	A>>1 → A	Δ	Δ	0	Δ
ASR Adr	Absolute	3F	2		M(Adr)>>1 → M(Adr)				
ASR n,SP	Indexed	4F	2		M(n+SP)>>1→M(n+SP)				
ASR n,X	Indexed	5F	2		M(n+X)>>1 → M(n+X)				
ASR A,X	Indexed	6F	1		M(A+X)>>1 → M(A+X)				
ASR n,Y	Indexed	7F	2		M(n+Y)>>1 → M(n+Y)				
ASR A,Y	Indexed	8F	1		M(A+Y)>>1 → M(A+Y)				

BCC Branch on carry clear (= BHS)

RTN	If C = 0: PC+Offset → 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 metod	OP	#	~	Operation	Flaggor N	Z	V	C
BCC Adr	Relativ	29	2	4	If C=0: PC+Offset → 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⊕V = 0: PC+Offset → PC
Flaggor	Påverkas ej
Beskrivning	Testar värdet hos Booleska uttrycket N⊕V. Om N⊕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⊕V =1 utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.

Detaljer:

Instruktion **BGE**	Adressering metod	OP	#	~	Operation	Flaggor 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⊕V)+Z = 0: PC+Offset → PC
Flaggor	Påverkas ej
Beskrivning	Testar värdet hos Booleska uttrycket (N⊕V)+Z. Om (N⊕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⊕V)+Z =1 utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.

Detaljer:

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

BHI Branch if higher than zero

RTN	If C+Z = 0: PC+Offset → 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 → PC	-	-	-	-

BITA Bit test register A

RTN:	A ∧ 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 metod	OP	#	~	Operation	Flaggor N	Z	V	C
`BITA #Data`	Immediate	98	2	3	A ∧ Data	Δ	Δ	0	-
`BITA Adr`	Absolute	A8	2	4	A ∧ M(Adr)				
`BITA n,SP`	Indexed	B8	2	4	A ∧ M(n+SP)				
`BITA n,X`	Indexed	C8	2	4	A ∧ M(n+X)				
`BITA n,Y`	Indexed	D8	2	4	A ∧ M(n+Y)				

BLE Branch on less than or equal to zero

RTN	If (N⊕V)+Z = 1: PC+Offset → PC
Flaggor	Påverkas ej
Beskrivning	Testar värdet hos Booleska uttrycket (N⊕V)+Z. Om (N⊕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⊕V)+Z = 0 utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.

Detaljer:

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

BLS Branch on lower or same

RTN	If C+Z = 1: PC+Offset → 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 metod	OP	#	~	Operation	Flaggor N	Z	V	C
BLS Adr	Relativ	2B	2	4	If C+Z = 1: PC+Offset → PC	-	-	-	-

BLT Branch on less than zero

RTN	If N⊕V = 1: PC+Offset → PC
Flaggor	Påverkas ej
Beskrivning	Testar värdet hos Booleska uttrycket N⊕V. Om N⊕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 minnetOm N⊕V = 0 utförs inget hopp. Nästa instruktion blir i så fall den direkt efter branchinstruktionen i minnet.

Detaljer:

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

BMI Branch on minus

RTN	If N = 1: PC+Offset → 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 metod	OP	#	~	Operation	Flaggor N	Z	V	C
BMI Adr	Relativ	22	2	4	If N = 1: PC+Offset → PC	-	-	-	-

BNE Branch if not equal to zero

RTN	If Z = 0: PC+Offset → 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 metod	OP	#	~	Operation	Flaggor N	Z	V	C
BNE Adr	Relativ	25	2	4	If (Z = 0) PC+Offset → PC	-	-	-	-

BPL Branch on plus

RTN	If N = 0: PC+Offset → 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 metod	OP	#	~	Operation	Flaggor N	Z	V	C
BPL Adr	Relativ	23	2	4	If N = 0: PC+Offset → PC	-	-	-	-

BRA Branch always

RTN	PC+Offset → PC
Flaggor	Påverkas ej
Beskrivning	Ett hopp utförs till adressen ADRESS = PC+Offset. Offset räknas från adressen efter branchinstruktionen, dvs vid uträkning av destinationsadressen 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 (återhoppsadressen) skrivs först på stacken. Ett hopp utförs sedan till adressen ADRESS = PC+Offset. Offset räknas från adressen efter BSR-instruktionen, dvs vid uträkningen av hopp-adressen pekar PC på operationskoden direkt efter BSR-instruktionen (= återhoppsadressen)

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

CLR Clear register or memory

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

Detaljer:

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

CMP Compare register and data

RTN	R – 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** Variant	Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
CMPA #Data	Immediate	97	2	3	A – Data	Δ	Δ	Δ	Δ
CMPA Adr	Absolute	A7	2	4	A – M(Adr)				
CMPA n,SP	Indexed	B7	2	4	A – M(n+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 – Data				
CMPX Adr	Absolute	AC	2	4	X – M(Adr)				
CMPX n,SP	Indexed	BC	2	4	X – M(n+SP)				
CMPY #Data	Immediate	9D	2	3	Y – Data				
CMPY Adr	Absolute	AD	2	4	Y – M(Adr)				
CMPY n,SP	Indexed	BD	2	4	Y – M(n+SP)				
CMPSP #Data	Immediate	9E	2	3	SP – Data				
CMPSP Adr	Absolute	AE	2	4	SP – M(Adr)				

COM Complement register or memory

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: Påverkas ej
Beskrivning	

Detaljer:

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

DEC Decrement register or memory

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: Påverkas ej
Beskrivning	Subtraherar 1 från operanden

Detaljer:

Instruktion **DEC** Variant	Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
DECA	Inherent	08	1	3	A–1→A	Δ	Δ	Δ	-
DEC Adr	Absolute	38	2	4	M(Adr) –1 → M(Adr)				
DEC n,SP	Indexed	48	2	4	M(n+SP) –1 → M(n+SP)				
DEC n,X	Indexed	58	2	4	M(n+X) –1 → M(n+X)				
DEC A,X	Indexed	68	1	4	M(A+X) –1 → M(A+X)				
DEC n,Y	Indexed	78	2	4	M(n+Y) –1 → M(n+Y)				
DEC A,Y	Indexed	88	1	4	M(A+Y) –1 → 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** Variant	Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
EORA #Data	Immediate	9B	2	4	A⊕Data → A	Δ	Δ	0	-
EORA Adr	Absolute	AB	2	5	A⊕M(Adr) → A				
EORA n,SP	Indexed	BB	2	5	A⊕M(n+SP) → A				
EORA n,X	Indexed	CB	2	5	A⊕M(n+X) → A				
EORA n,Y	Indexed	DB	2	5	A⊕M(n+Y) → 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** Variant	Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
EXG A,CC	Inherent	9F	1	4	A ↔ CC	Δ	Δ	Δ	Δ
EXG X,Y	Inherent	AF	1	4	X ↔ Y	-	-	-	-
EXG X,SP	Inherent	BF	1	4	X ↔ SP	-	-	-	-
EXG Y,SP	Inherent	CF	1	4	Y ↔ SP	-	-	-	-

INC Increment register or memory

RTN	A + 1 → A 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: Påverkas ej
Beskrivning	

Detaljer:

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

JMP Jump

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

Detaljer:

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

JSR Jump to subroutine

RTN	SP–1 → SP; PC → M(SP); EA → 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 **JSR** Variant	Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
JSR Adr	Absolute	34	2	4	SP–1 → SP PC → M(SP) Adr → PC	-	-	-	-
JSR n,X	Indexed	54	2	5	SP–1 → SP PC → M(SP) n+X → PC				
JSR A,X	Indexed	64	1	5	SP–1 → SP PC → M(SP) A+X → PC				
JSR n,Y	Indexed	74	2	5	SP–1 → SP PC → M(SP) n+Y → PC				
JSR A,Y	Indexed	84	1	5	SP–1 → SP PC → M(SP) A+Y → 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 **LD** Variant		Adressering metod	OP	#	~	Operation	Flaggor 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

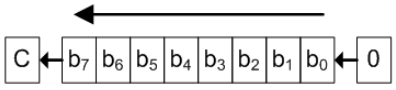
Varianter	
RTN	EA → R ; R kan vara X,Y eller SP
Flaggor	Påverkas ej
Beskrivning	Laddar effektiva adressen i R.

Detaljer:

Instruktion **LEA** Variant		Adressering metod	OP	#	~	Operation	Flaggor 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.

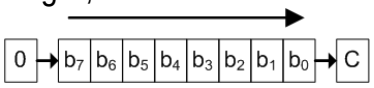


Detaljer:

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

NEG Negate register or memory

RTN	0 – R → R eller 0 – M → 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 ≠ 0.
Beskrivning	2-komplementerar innehållet i angivet register eller minnesinnehåll.

Detaljer:

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

NOP No operation

RTN	
Flaggor	Påverkas ej
Beskrivning	Instruktionen utför ingenting

Detaljer:

Instruktion **NOP**	Adressering metod	OP	#	~	Operation	Flaggor 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 metod	OP	#	~	Operation	Flaggor N	Z	V	C
ORA	#Data	Immediate	9A	2	4	$A \vee Data \rightarrow A$	Δ	Δ	0	-
ORA	Adr	Absolute	AA	2	5	$A \vee M(Adr) \rightarrow A$				
ORA	n,SP	Indexed	BA	2	5	$A \vee M(n+SP) \rightarrow A$				
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	$CC \vee Data \rightarrow CC$
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 metod	OP	#	~	Operation	Flaggor I	N	Z	V	C
ORCC	#Data	Inherent	02	2	4	$CC \vee Data \rightarrow CC$	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 metod	OP	#	~	Operation	Flaggor 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

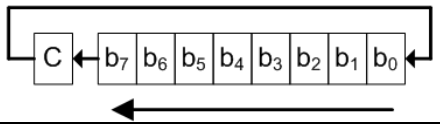
Varianter	
RTN	M(SP) → R SP+1 → SP
Flaggor	Flaggorna påverkas endast vid PULC, då flaggorna får värden från stacken
Beskrivning	Översta dataordet på stacken läses och placeras i angivet register. Stackpekaren uppdateras sedan

Detaljer:

Instruktion **PUL** Variant	Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
PULA	Inherent	14	1	3	M(SP) → A, 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

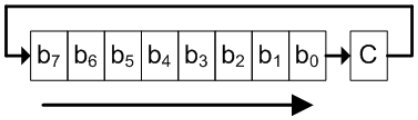


Detaljer:

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

RTS Return from Subroutine

RTN	M(SP) → PC; SP+1 → SP
Flaggor	Påverkas ej
Beskrivning	Återhopp från en subrutin utförs genom att översta dataordet på stacken, dvs återhoppsadressen, läses och placeras i PC. Stackpekaren uppdateras sedan

Detaljer:

Instruktion **RTS**	Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
RTS	Inherent	43	1	2	M(SP) → PC SP+1 → SP	-	-	-	-

RTI Return from Interrupt

Varianter	
RTN	M(SP) → CC; SP+1 → SP, M(SP) → A; SP+1 → SP, M(SP) → X; SP+1 → SP, M(SP) → Y; SP+1 → SP, M(SP) → PC; SP+1 → SP
Flaggor	Påverkas ej
Beskrivning	Återhopp från undantagshantering utförs genom att samtliga register återställes från stacken och stackpekaren uppdateras.

Detaljer:

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

SBCA Subtract with borrow Data from register A

RTN	A – M – C → A, 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 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** Variant	Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
SBCA #Data	Immediate	93	2	4	A–Data-C → A	Δ	Δ	Δ	Δ
SBCA Adr	Absolute	A3	2	5	A–M(Adr)-C → A				
SBCA n,SP	Indexed	B3	2	5	A – M(n+SP) – C → A				
SBCA n,X	Indexed	C3	2	5	A – M(n+X) – C → A				
SBCA n,Y	Indexed	D3	2	5	A – M(n+Y) – C → A				

ST Store register into memory

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

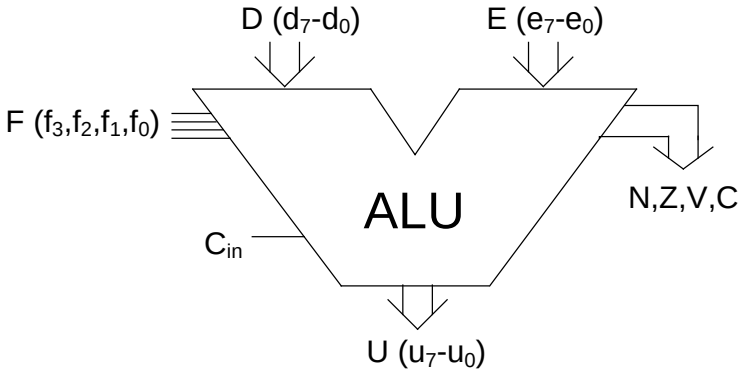
RTN	A – 0 eller M – 0 M = data från minnesadress
Flaggor	N: Ettställs om resultatets teckenbit (bit 7) får värdet 1. Z: Ettställs om samtliga åtta bitar i resultatet blir noll. V: Nollställs. C: Nollställs
Beskrivning	Låter datavärdet i A eller M passera ALU:n och sätter flaggvipporna N och Z så att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

Detaljer:

Instruktion **TST** Variant	Adressering metod	OP	#	~	Operation	Flaggor 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 (Aritmetic 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	$U=D \vee E$	u_7	(1)	0	0
0	1	1	1	$U=D \wedge E$	u_7	(1)	0	0
1	0	0	0	$U=D \oplus E$	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 = c_8, dvs. carry ut från additionen av de mest signifikanta siffrorna.

(4) C = utskiftad bit, dvs. bit d_7 före vänsterskiftet.

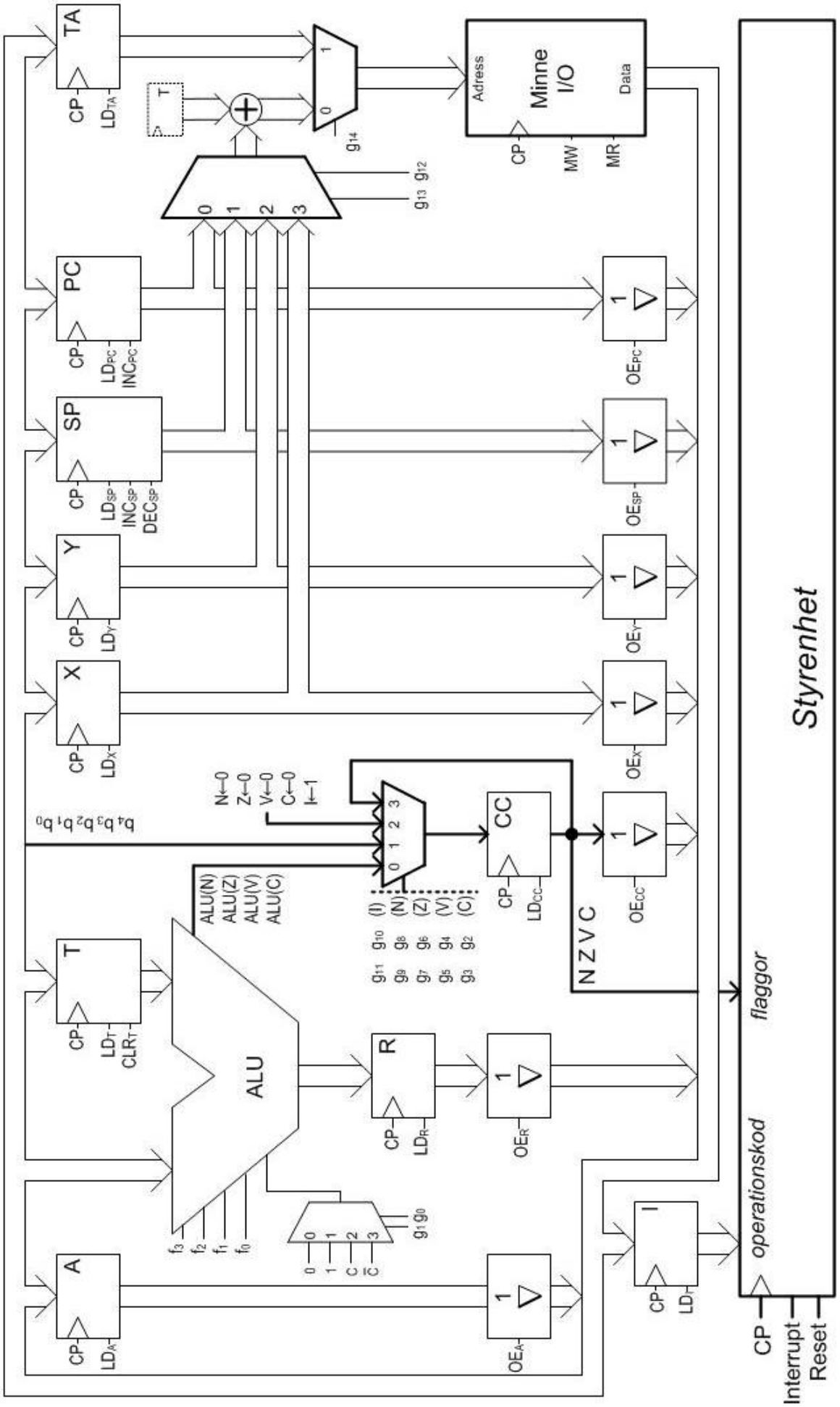
(5) C = utskiftad bit, dvs. bit d_0 före högerskiftet.

(6) V = $C_{in} \oplus d_7$ vid högerskift, $d_7 \oplus d_6$ vid vänsterskift, 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
Insignaler till styrenheten	
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
Utsignaler från styrenheten	
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_5	g_4	V väljs enligt:	RTN
0	0	V tas från ALU:n	ALU(V)→V
0	1	V tas från bit 1 på bussen	b_1→ V
1	0	Återställning (nollställning) av V	0→ V
1	1	V 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 C	0→ Z
1	1	Z återförs (ändras ej)	

g_9	g_8	N väljs enligt:	RTN
0	0	V tas från ALU:n	ALU(N)→N
0	1	N tas från bit 3 på bussen	b_3→ N
1	0	Återställning (nollställning) av N	0→ N
1	1	N 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)