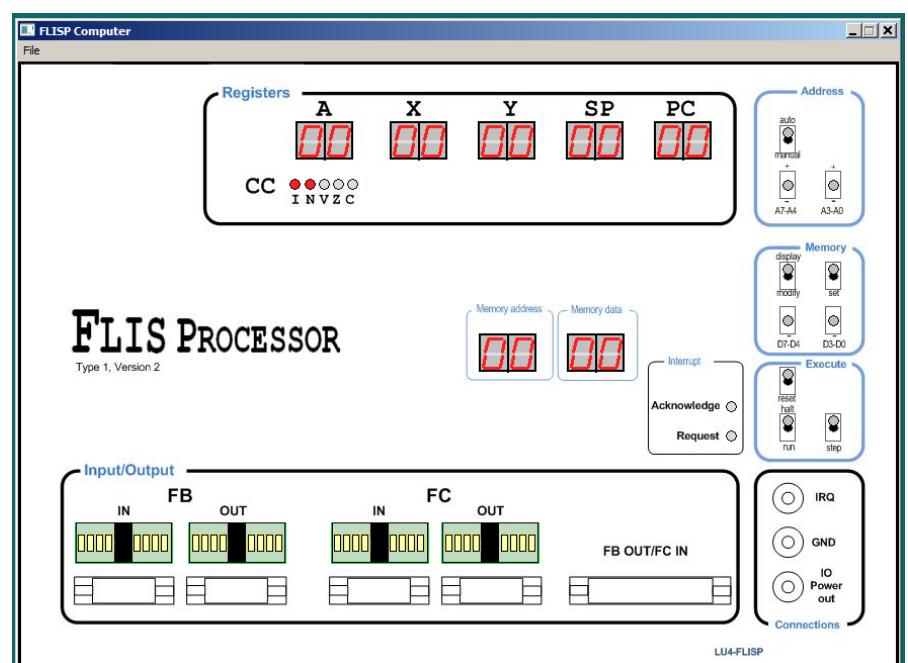
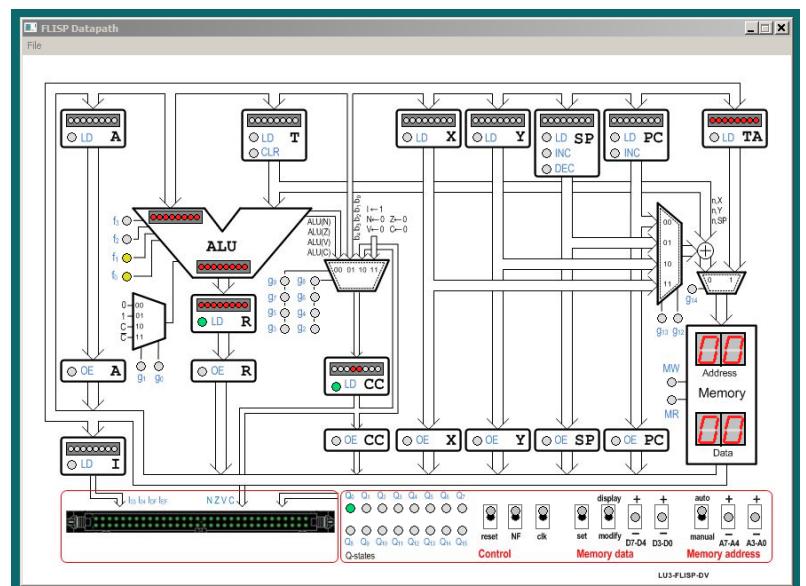


Version: 2025-12-09

FLEXIBLE INSTRUCTION SET PROCESSOR

FLISP HANDBOK



Innehåll

Vektorer och I/O	3
Registeruppsättning	4
Adresseringssätt.....	5
Inherent [ih]	5
Omedelbar (Immediate) [im]	5
Absolut (Absolute) [ab]	5
Relativ [pc].....	6
Indirekt stackregister med konstant offset (Indexed) [ns]	6
Indirekt register med konstant offset (Indexed) [nR]	7
Indirekt register med ackumulator offset (Indexed)[aR].....	7
Indirekt register med pre/post decrement/increment (Indexed)	8
Assemblerspråk.....	9
Instruktionsgrupper	10
Karta över operationskoder	11
ADCA Add data with carry into register A	13
ADDA Add data into register A	13
ANDA Logical AND data into register A.....	14
ANDCC Logical AND data into register CC	14
ASL Arithmetic shift left.....	15
ASR Arithmetic shift right.....	16
BCC Branch on carry clear (= BHS).....	16
BCS Branch on carry set (= BLO).....	17
BEQ Branch on equal to zero	17
BGE Branch on greater than or equal to zero	18
BGT Branch on greater than zero	18
BHS Branch if higher or same (=BCC).....	18
BHI Branch if higher than zero	19
BLO Branch on lower (=BCS).....	19
BITA Bit test register A	19
BLE Branch on less than or equal to zero	20
BLS Branch on lower or same	20
BLT Branch on less than zero.....	21
BMI Branch on minus	21
BNE Branch if not equal to zero.....	22
BPL Branch on plus	22
BRA Branch always	23
BSR Branch to subroutine	23
BVC Branch if no overflow	24
BVS Branch if overflow	24
CLR Clear register or memory	25
CMP Compare register and data	26
COM Complement register or memory	27
DEC Decrement register or memory	27
EOR Exclusive-OR register A	28
EXG Exchange register contents	28
INC Increment register or memory	29

JMP	Jump	29
JSR	Jump to subroutine.....	30
LD	Load register	31
LEA	Load effective address	32
LSL	Logical shift left, ASL Arithmetic shift left.....	32
LSR	Logical shift right	33
NEG	Negate register or memory	33
NOP	No operation.....	34
ORA	Logical OR data into register A.....	34
ORCC	Logical OR Data into register CC.....	34
PSH	Push register on Stack	35
PUL	Pull register from stack.....	35
ROL	Rotate left.....	36
ROR	Rotate right.....	37
RTS	Return from Subroutine	37
RTI	Return from Interrupt.....	38
SBCA	Subtract with borrow Data from register A	39
ST	Store register into memory	40
SUBA	Subtract data from register A.....	41
TFR	Transfer register to register	42
TST	Test.....	42
ALU-funktioner		43
Funktionstabell för ALU.....		44
Dataväg, styrenhet och minne		45
Signaler i fast styrenhet		47
Styrsignaler för väljarfunktioner.....		48
Villkorstest		49
ASCII-tabellen		50

Detta häfte utgör den sammanfattande beskrivningen av Flis-processorn. Häftet är indelat i tre delar. Del 1 behandlar assemblerprogrammerarens bild av Flisp, del 2 beskriver detaljer i mikroarkitekturen och del 3 innehåller diverse tabeller.

Del I: Programmerarens bild

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

Del II: Mikroarkitekturen

ALU-funktioner
Funktionstabell för ALU
Dataväg, styrenhet och minne
RTN (register transfer notation)
Signaler i fast styrenhet
Signaler för väljarfunktioner

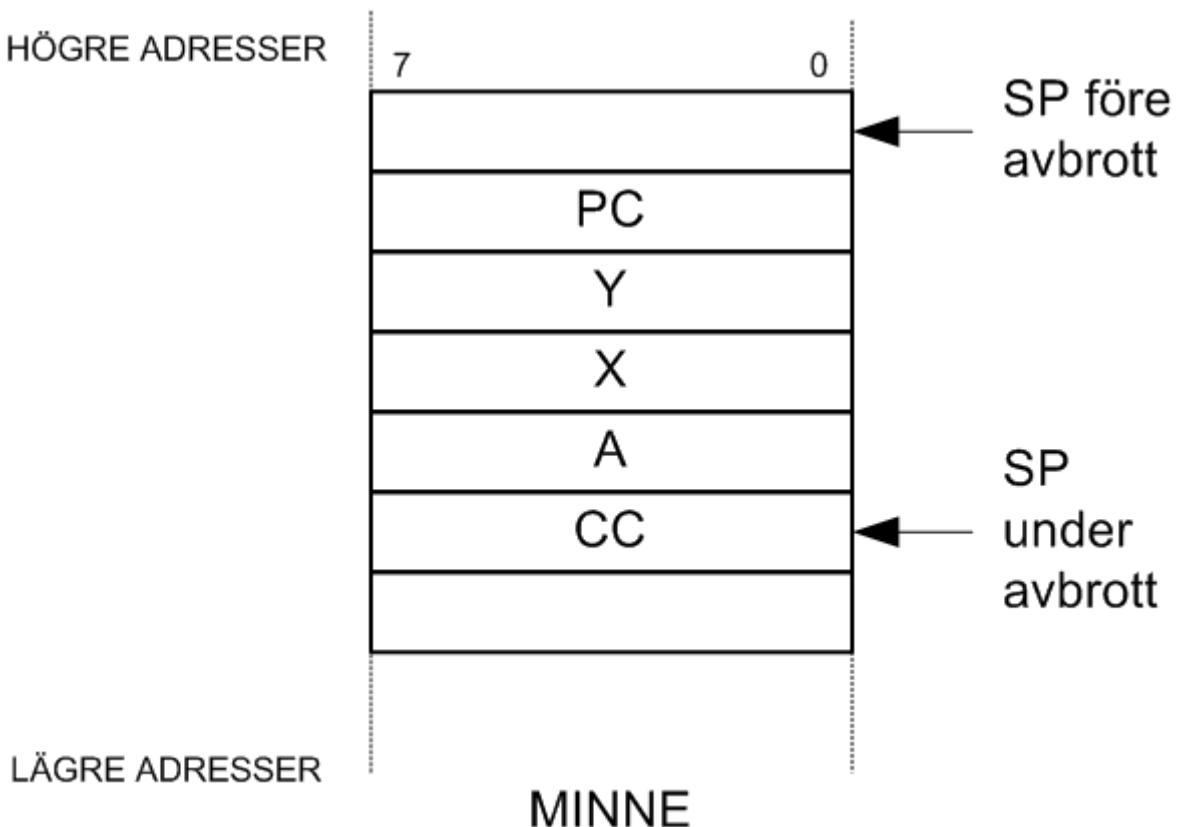
Del III: Övrigt

Villkorstest
ASCII-tabellen

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"

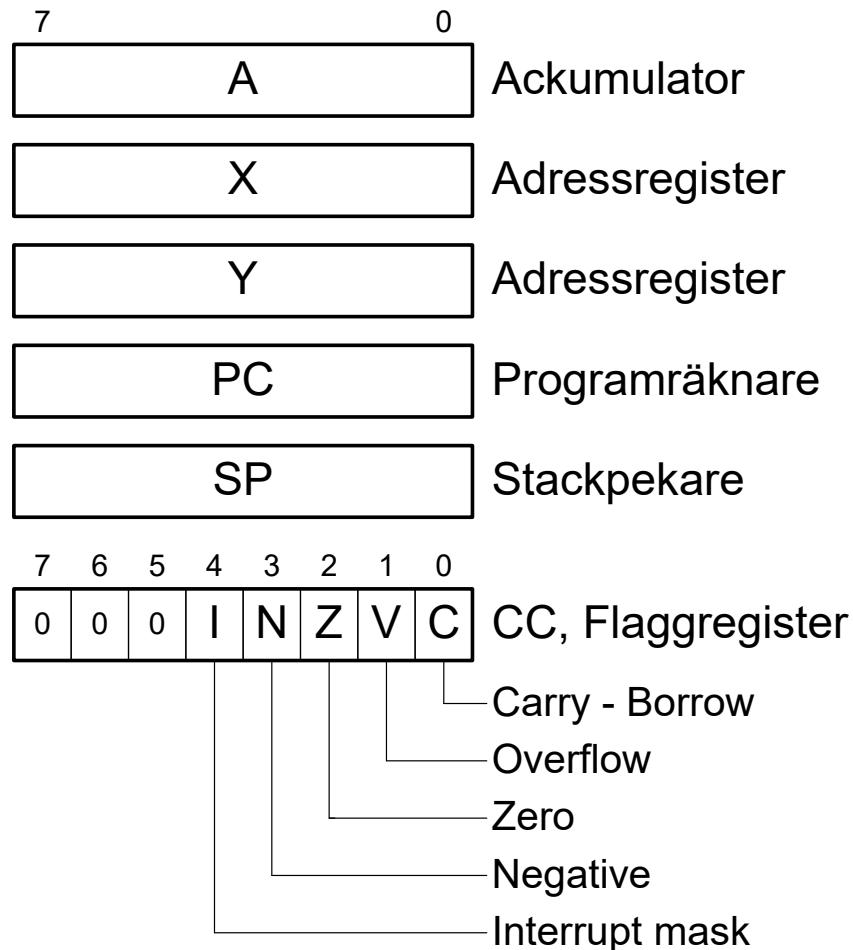


Vid "IRQ" pekar PC på den instruktion som skulle ha utförts om inget avbrott signalerats.

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]

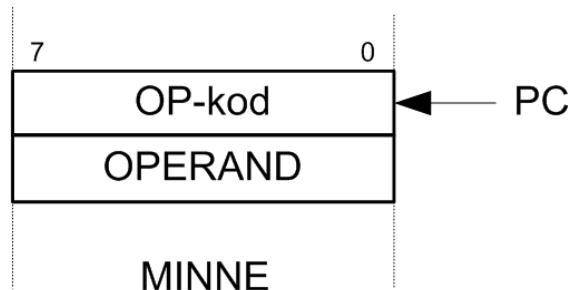
Operanden eller operanderna ges direkt av instruktionen. Ingen extra operandinformation (utöver operationskod) krävs. Det finns ingen generell RTN-beskrivning för inherent adressering.

Omedelbar (Immediate) [im]

Operanden är kodad tillsammans med operationen.

RTN: $M(PC+1)$

Assemblersyntax: #<DATA>

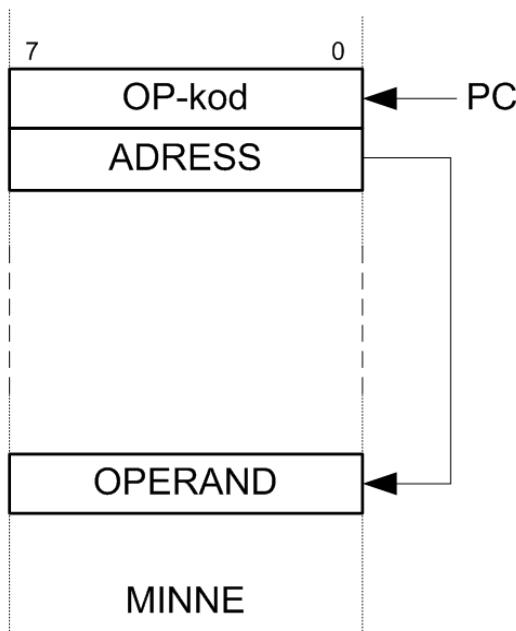


Absolut (Absolute) [ab]

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

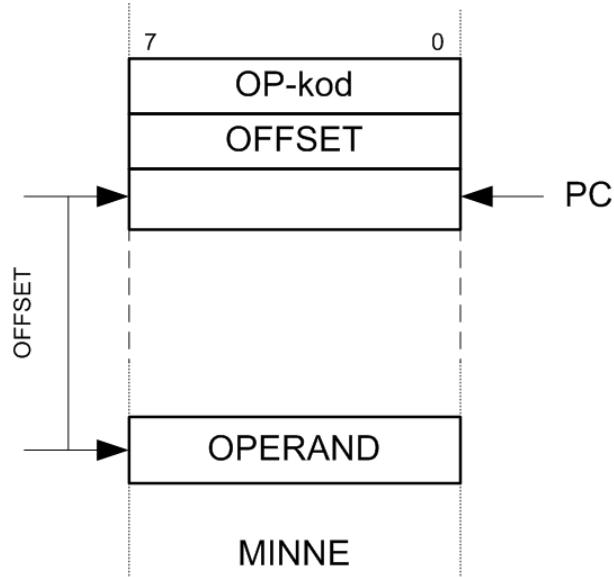
RTN: $[M(PC+1)]$

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".



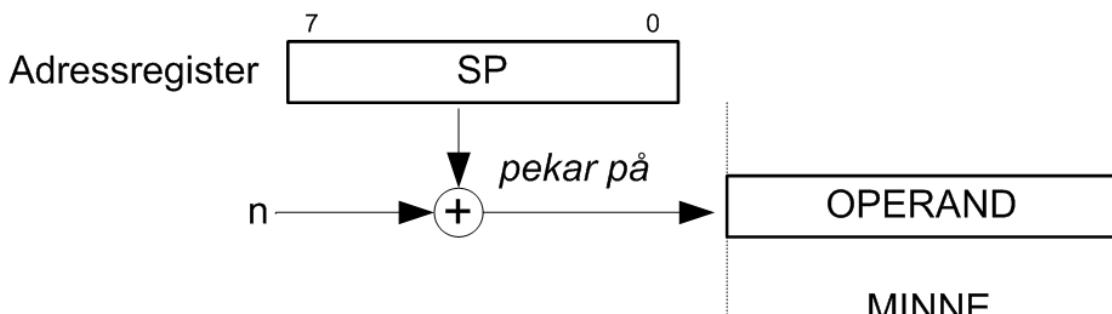
Observera att adressberäkningar som inbegriper PC förutsätter att PC pekar på nästa instruktion

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.

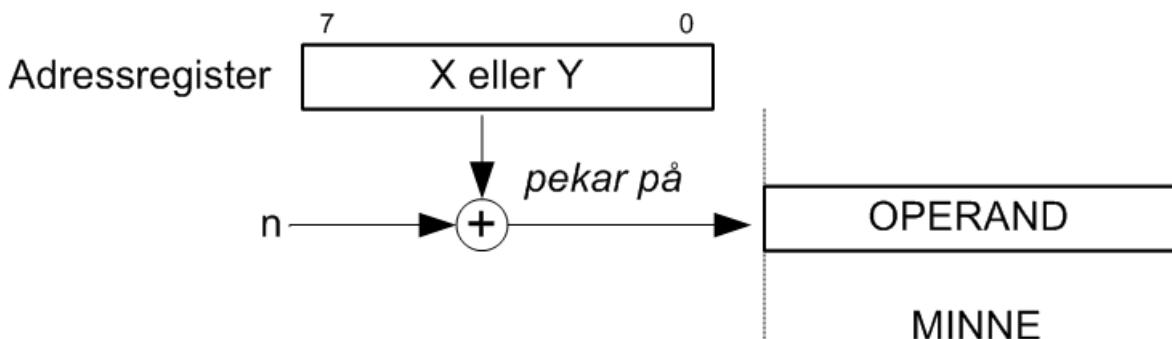
De indexerade adresseringssätten kan också användas tillsammans med något av registren X eller Y. I den följande framställninigen betecknas något X eller Y med bokstaven R.

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



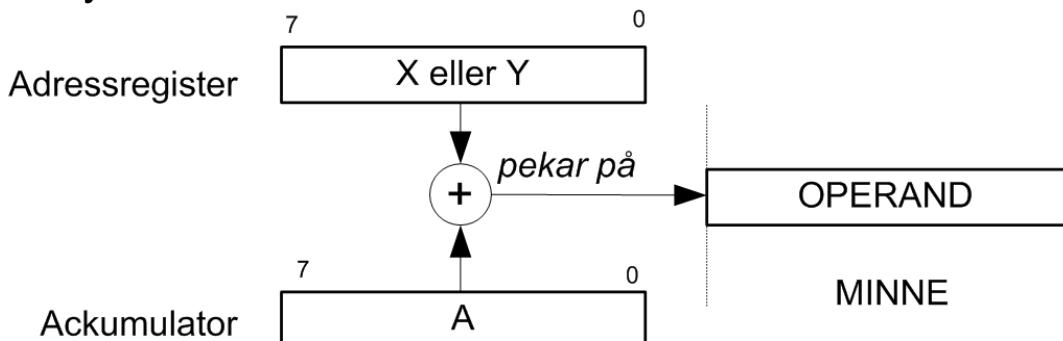
Konstanten n (8 bitar) adderas modulo 256 och kan anges med eller utan tecken.

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

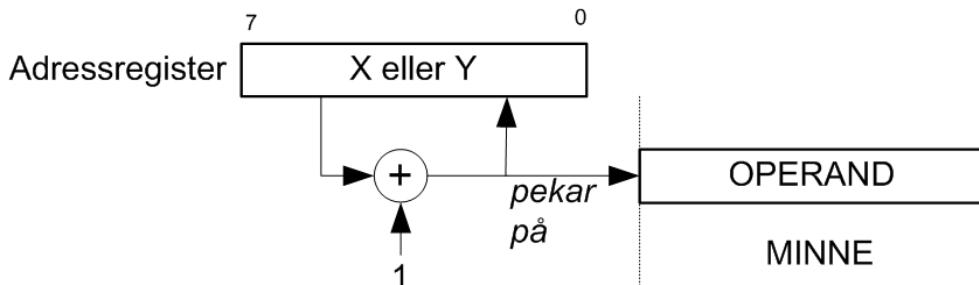


Indirekt register med pre/post decrement/increment (Indexed)

Pre auto increment [+R]

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

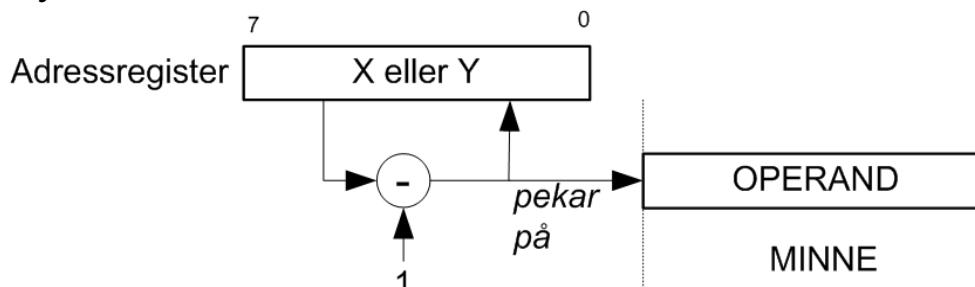
Assemblersyntax: , +R



Pre auto decrement [-R]

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

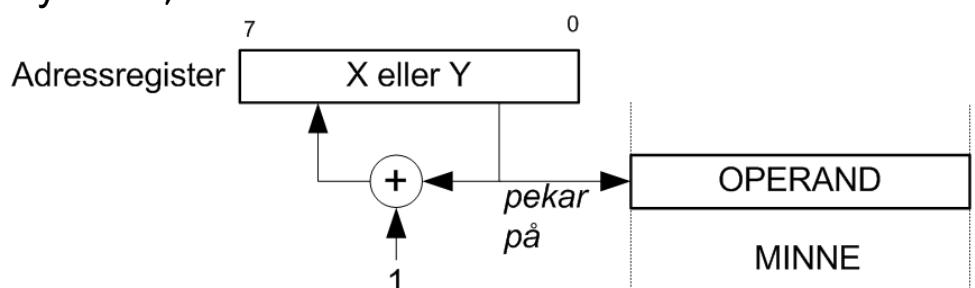
Assemblersyntax: , -R



Post auto increment [R+]

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

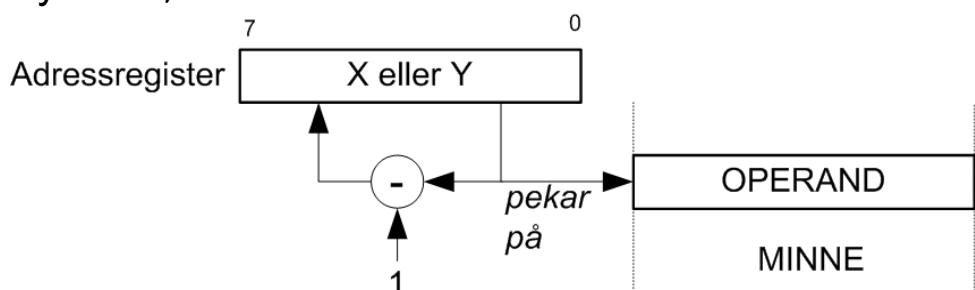
Assemblersyntax: , R+



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 bytes 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

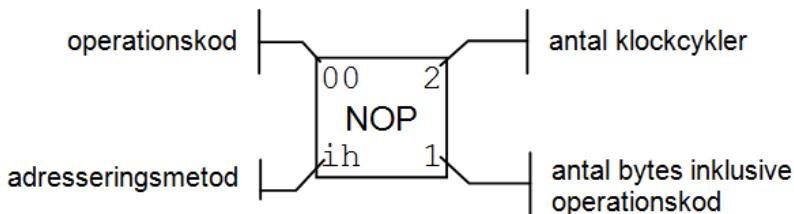
"Stack operations"

PSHA, PSHC, PSHX, PSHY, PULA, PULC, PULX, PULY

"Misc."

NOP

Karta över operationskoder



Anm:

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

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

Detaljerad beskrivning av FLIS-processorns instruktioner

Instruktion Add with Carry Variant	Adressering metod	Operation			Flaggor
		OP	#	~	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 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 ARTN: $A + \text{Opr} + C \rightarrow A$

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 metod	OP	#	~	Operation	Flaggor 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 ARTN $A + \text{Opr} \rightarrow A$

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 metod	OP	#	~	Operation	Flaggor 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	$A + M(n+X) \rightarrow A$	
ADDA n, Y	Indexed	D6	2	5	$A + M(n+Y) \rightarrow A$	

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

ASL**Arithmetic shift left**

RTN	$A \ll 1 \rightarrow A$ eller $M(EA) \ll 1 \rightarrow M(EA)$
Flaggor	<p>N: Kopia av bit 7 efter skiftet.</p> <p>Z: Ettställs om samtliga åtta bitar i resultatet blir noll.</p> <p>V: Ettställs om C och bit 7 är olika efter operationen, dvs overflow vid 2-komplements-representation inträffar.</p> <p>C: bit 7 före skiftet blir ny carrybit efter skiftet.</p>
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
Variant	metod	OP	#	\sim		N Z V C
ASLA	Inherent	0B	1	3	$A \ll 1 \rightarrow A$	$\Delta \quad \Delta \quad \Delta \quad \Delta$
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	$A >> 1 \rightarrow A$ 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: 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			
Variant	metod	OP	#	~					N	Z	V	C
ASRA	Inherent	0F	1	3	$A >> 1 \rightarrow A$				Δ	Δ	0	Δ
ASR Adr	Absolute	3F	2	4	$M(Adr) >> 1 \rightarrow M(Adr)$							
ASR n, SP	Indexed	4F	2	4	$M(n+SP) >> 1 \rightarrow M(n+SP)$							
ASR n, X	Indexed	5F	2	4	$M(n+X) >> 1 \rightarrow M(n+X)$							
ASR A, X	Indexed	6F	1	4	$M(A+X) >> 1 \rightarrow M(A+X)$							
ASR n, Y	Indexed	7F	2	4	$M(n+Y) >> 1 \rightarrow M(n+Y)$							
ASR A, Y	Indexed	8F	1	4	$M(A+Y) >> 1 \rightarrow M(A+Y)$							

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

BGE Branch on greater than or equal to zero

RTN	If $N \oplus V = 0$: $PC + Offset \rightarrow PC$
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 metod	OP	#	~	Operation	Flaggor N Z V C
BGE Adr	Relativ	2D	2	4	If $((N \oplus V) = 0)$ $PC + Offset \rightarrow PC$	- - - -

BGT Branch on greater than zero

RTN	If $(N \oplus V) + Z = 0$: $PC + Offset \rightarrow PC$
Flaggor	Påverkas ej
Beskrivning	Testar värdet hos Booleska uttrycket $(N \oplus V) + Z$. Om $(N \oplus V) + Z = 0$ utförs ett hopp till adressen $ADRESS = PC + Offset$. 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 metod	OP	#	~	Operation	Flaggor N Z V C
BGT Adr	Relativ	2C	2	4	If $((N \oplus V) + Z = 0)$ $PC + Offset \rightarrow PC$	- - - -

BHS Branch if higher or same (=BCC)

Se instruktionen "BCC"

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

BLO Branch on lower (=BCS)

Se instruktionen "BCS"

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 metod	OP	#	~	Operation	Flaggor
BITA #Data	Immediate	98	2	3	A \wedge Data	N Z V C
BITA Adr	Absolute	A8	2	4	A \wedge M(Adr)	Δ Δ 0 -
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 → 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 metod	Operation			Flaggor			
		OP	#	~	N	Z	V	C
BLE Adr	Relativ	2E	2	4	If $(N \oplus V) + Z = 1$: 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	Operation			Flaggor			
		OP	#	~	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 \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 metod	OP	#	\sim	Operation	Flaggor 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 metod	OP	#	\sim	Operation	Flaggor 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 → 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 metod	OP	#	~	Operation	Flaggor 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 metod	OP	#	~	Operation	Flaggor 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	$0 \rightarrow A$, 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 Variant	Adressering metod	Operation			Flaggor			
		OP	#	~	N	Z	V	C
CLRA	Inherent	05	1	3	0 → A			
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 en lånesiffra = 1 ("borrow") uppstår från bitpositionen längst till vänster 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 – Data
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	$R' \rightarrow R$ 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	Operandens bitar inverteras

Detaljer:

Instruktion COM	Adressering	Operation				Flaggor
Variant	metod	OP	#	\sim		N Z V C
COMA	Inherent	0A	1	3	$A' \rightarrow A$	Δ Δ 0 -
COM Adr	Absolute	3A	2	4	$M(Adr)' \rightarrow M(Adr)$	
COM n, SP	Indexed	4A	2	4	$M(n+SP)' \rightarrow M(n+SP)$	
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	#	\sim		N Z V C
DECA	Inherent	08	1	3	$A-1 \rightarrow A$	Δ Δ Δ -
DEC Adr	Absolute	38	2	4	$M(Adr)-1 \rightarrow M(Adr)$	
DEC n, SP	Indexed	48	2	4	$M(n+SP)-1 \rightarrow M(n+SP)$	
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 EOR	Adressering			Operation			Flaggor			
Variant	metod	OP	#	~			N	Z	V	C
EORA #Data	Immediate	9B	2	4	$A \oplus Data \rightarrow A$		Δ	Δ	0	-
EORA Adr	Absolute	AB	2	5	$A \oplus M(Adr) \rightarrow A$					
EORA n, SP	Indexed	BB	2	5	$A \oplus M(n+SP) \rightarrow A$					
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	$A \leftrightarrow CC$		Δ	Δ	Δ	Δ
EXG X, Y	Inherent	AF	1	4	$X \leftrightarrow Y$		-	-	-	-
EXG X, SP	Inherent	BF	1	4	$X \leftrightarrow SP$		-	-	-	-
EXG Y, SP	Inherent	CF	1	4	$Y \leftrightarrow SP$		-	-	-	-

INC Increment register or memoryRTN $A + 1 \rightarrow A$ 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 INC Variant	Adressering metod	OP	#	~	Operation	Flaggor N Z V C
INCA	Inherent	07	1	3	$A+1 \rightarrow A$	Δ Δ Δ -
INC Adr	Absolute	37	2	4	$M(Adr) + 1 \rightarrow M(Adr)$	
INC n, SP	Indexed	47	2	4	$M(n+SP) + 1 \rightarrow M(n+SP)$	
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 JumpRTN $EA \rightarrow PC$

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	$Adr \rightarrow PC$	- - - -
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 → 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	Operation			Flaggor			
		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	2	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	2	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	Adressering				Operation		Flaggor			
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)

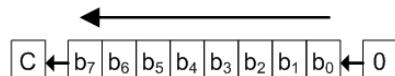
Flaggor N: Kopia av bit 7 efter skiften.

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 skiften blir ny carrybit efter skiften.

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	Adressering				Operation		Flaggor			
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 $A >> 1 \rightarrow A$ eller $M >> 1 \rightarrow M$

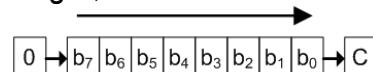
Flaggor N: Nollställs.

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	Adressering				Operation				Flaggor			
Variant	metod	OP	#	~					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	2	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 $0 - R \rightarrow R$ 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	Adressering				Operation				Flaggor			
Variant	metod	OP	#	~					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	Operation			Flaggor
	metod	OP	#	~	N Z V C
NOP	Inherent	00	1	2	No operation

ORA Logical OR data into register ARTN $A \vee M \rightarrow A$

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	Adressering	Operation			Flaggor
Variant	metod	OP	#	~	N Z V C
ORA #Data	Immediate	9A	2	4	$A \vee Data \rightarrow A$
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 CCRTN $CC \vee Data \rightarrow CC$

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	Operation			Flaggor
	metod	OP	#	~	I N Z V C
ORCC #Data	Inherent	02	2	4	$CC \vee Data \rightarrow CC$

PSH Push register on Stack

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

Detaljer:

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

PUL Pull register from stack

Varianter	
RTN	$M(SP) \rightarrow R$ $SP+1 \rightarrow SP$
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:

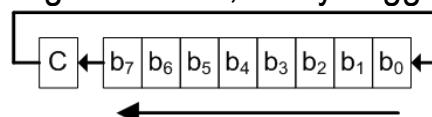
Instruktion PUL	Adressering	Operation				Flaggor
Variant	metod	OP	#	~		N Z V C
PULA	Inherent	14	1	3	$M(SP) \rightarrow A$, $SP+1 \rightarrow SP$	- - - -
PULX	Inherent	15	1	3	$M(SP) \rightarrow X$ $SP+1 \rightarrow SP$	- - - -
PULY	Inherent	16	1	3	$M(SP) \rightarrow Y$ $SP+1 \rightarrow SP$	- - - -
PULC	Inherent	17	1	3	$M(SP) \rightarrow CC$ $SP+1 \rightarrow SP$	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, Carry-flaggan ingår i skiftet.



Detaljer:

Instruktion ROL	Adressering				Operation				Flaggor			
Variant	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 $>>1 \rightarrow A$ eller M $>>1 \rightarrow M$
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, Carry-flaggan ingår i skiftet.

Detaljer:

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

RTS Return from Subroutine

RTN	M(SP) \rightarrow PC; SP+1 \rightarrow 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	Operation				Flaggor
	metod	OP	#	\sim		N Z V C
RTS	Inherent	43	1	2	M(SP) \rightarrow PC SP+1 \rightarrow SP	- - - -

RTI Return from Interrupt

Varianter

RTN $M(SP) \rightarrow CC; SP+1 \rightarrow SP,$
 $M(SP) \rightarrow A; SP+1 \rightarrow SP,$
 $M(SP) \rightarrow X; SP+1 \rightarrow SP,$
 $M(SP) \rightarrow Y; SP+1 \rightarrow SP,$
 $M(SP) \rightarrow PC; SP+1 \rightarrow SP$

Flaggor Påverkas ej

Beskrivning Återhopp från undantagshantering utförs genom att samtliga register återställes från stacken och stackpekaren uppdateras.

Detaljer:

Instruktion RTI Variant	Adressering metod	OP # ~			Operation	Flaggor				
		OP	#	~		I	N	Z	V	C
RTI	Inherent	44	1	6	$M(SP) \rightarrow CC; SP+1 \rightarrow SP,$ $M(SP) \rightarrow A; SP+1 \rightarrow SP,$ $M(SP) \rightarrow X; SP+1 \rightarrow SP,$ $M(SP) \rightarrow Y; SP+1 \rightarrow SP,$ $M(SP) \rightarrow PC; SP+1 \rightarrow SP$	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 ("borrow") 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	Operation				Flaggor			
		OP	#	~		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	$R \rightarrow M$
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 Variant	Adressering metod	Operation			Flaggor			
		OP	#	~	N	Z	V	C
STA Adr	Absolute	E1	2	3	A \rightarrow M(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(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	1	3	X \rightarrow M(A+X)			
STX n, Y	Indexed	70	2	3	X \rightarrow M(n+Y)			
STX A, Y	Indexed	80	1	3	X \rightarrow M(A+Y)			
STY Adr	Absolute	31	2	3	Y \rightarrow M(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	61	1	3	Y \rightarrow M(A+X)			
STY n, Y	Indexed	71	2	3	Y \rightarrow M(n+Y)			
STY A, Y	Indexed	81	1	3	Y \rightarrow M(A+Y)			
STSP Adr	Absolute	32	2	3	SP \rightarrow M(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	1	3	SP \rightarrow M(A+X)			
STSP n, Y	Indexed	72	2	3	SP \rightarrow M(n+Y)			
STSP A, Y	Indexed	82	1	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 Variant	Adressering metod	Operation			Flaggor			
		OP	#	~	N	Z	V	C
SUBA #Data	Immediate	94	2	4	A–Data → A	Δ	Δ	Δ
SUBA Adr	Absolute	A4	2	5	A–M(Adr) → A			
SUBA n, SP	Indexed	B4	2	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 $R1 \rightarrow R2$

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	#	\sim	Operation	Flaggor N Z V C
TFR A, CC	Inherent	18	1	2	A \rightarrow CC	$\Delta \Delta \Delta \Delta$
TFR CC, A	Inherent	19	1	2	CC \rightarrow A	- - - -
TFR X, Y	Inherent	1A	1	2	X \rightarrow Y	- - - -
TFR Y, X	Inherent	1B	1	2	Y \rightarrow X	- - - -
TFR X, SP	Inherent	1C	1	2	X \rightarrow SP	- - - -
TFR SP, X	Inherent	1D	1	2	SP \rightarrow X	- - - -
TFR Y, SP	Inherent	1E	1	2	Y \rightarrow SP	- - - -
TFR SP, Y	Inherent	1F	1	2	SP \rightarrow 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

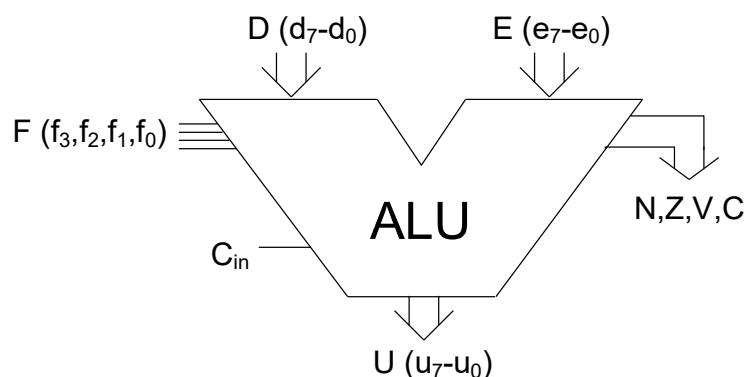
Beskrivning Låter datavärdet i A eller M passera ALU:n och sätter flaggvipporna N och Z så att man kan avgöra datavärdets tecken eller om det är noll. Endast flaggvipporna påverkas.

Detaljer:

Instruktion TST Variant	Adressering metod	OP	#	\sim	Operation	Flaggor N Z V C
TSTA	Inherent	09	1	2	A-0	$\Delta \Delta 0 0$
TST Adr	Absolute	39	2	3	M(adr)-0	
TST n, SP	Indexed	49	2	3	M(n+SP)-0	
TST n, X	Indexed	59	2	3	M(n+X)-0	
TST A, X	Indexed	69	1	3	M(A+X)-0	
TST n, Y	Indexed	79	2	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 f_1, f_0 .
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	$U=D \ll 1 (C_{in})$	u_7	(1)	(6)	(4)
1	1	1	0	$U=(C_{in}) D \gg 1$	u_7	(1)	(6)	(5)
1	1	1	1	$U=(d_7) D \gg 1$	u_7	(1)	0	(5)

Anm:

$$D_{1k} = 1\text{-komplement } D = D'$$

$$(1) Z = \overline{u_7} \wedge \overline{u_6} \wedge \overline{u_5} \wedge \overline{u_4} \wedge \overline{u_3} \wedge \overline{u_2} \wedge \overline{u_1} \wedge \overline{u_0},$$

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. Observera att vid addition med 2-komplement (subtraktion) gäller att C-flaggan är inversen av c_8 (alltså borrow vid subtraktion).

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

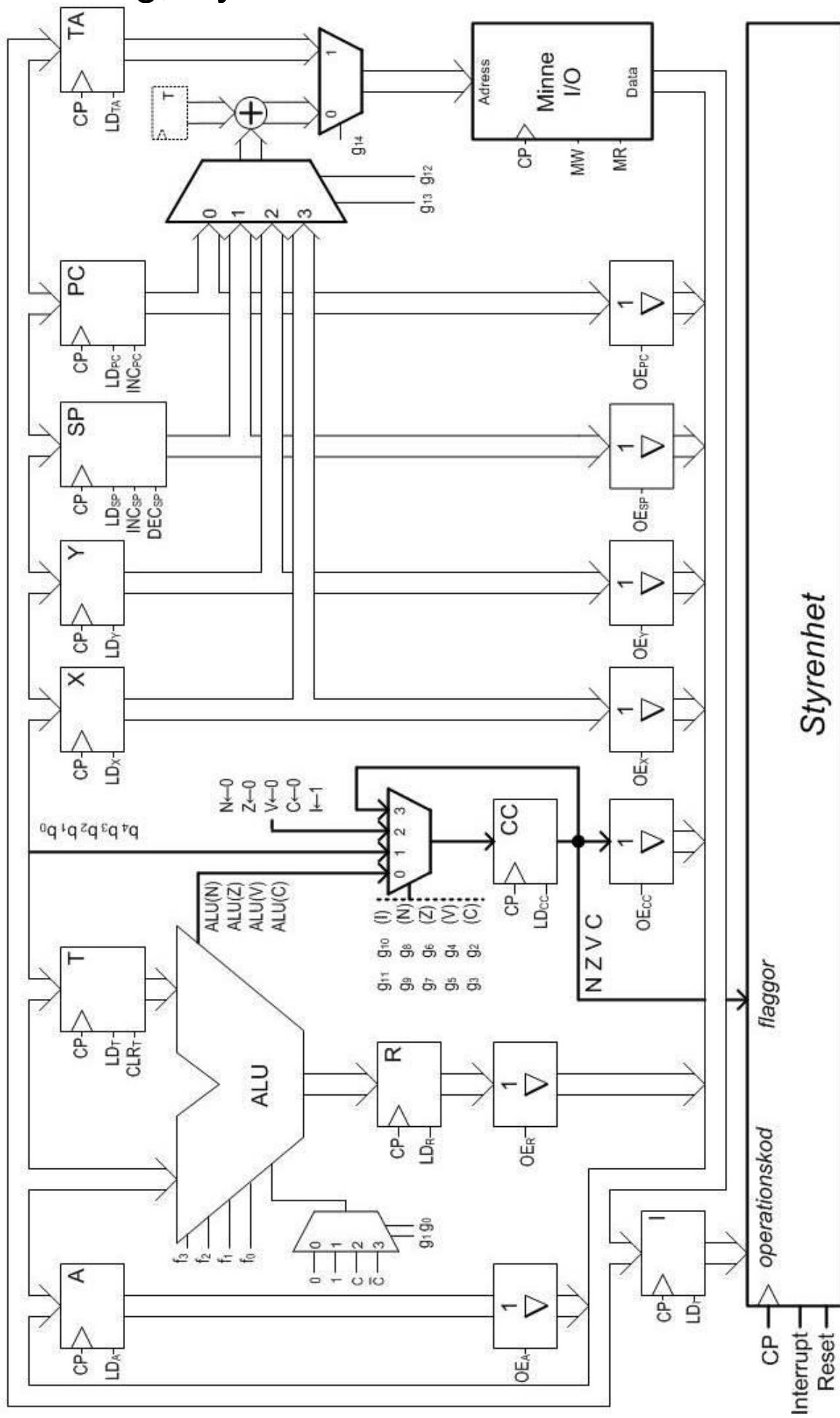
(5) C = utskiftad bit, dvs. bit d_0 före högerskiftet.

(6) V = $C_{in} \oplus d_7$ vid högerskift, $d_7 \oplus d_6$ vid vänsterskift, dvs. sätts till 1 om skiften 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, sampelas vid CP (positiv flank), kontrolleras vid NF
Q_{state}	Tillstånd, "state" angerne 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" angerne 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 SP", 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	$ALU(C) \rightarrow C$
0 1	C tas från bit 0 på bussen	$b_0 \rightarrow C$
1 0	Återställning (nollställning) av C	$0 \rightarrow C$
1 1	Z återförs (ändras ej)	

$g_5 g_4$	V väljs enligt:	RTN
0 0	V tas från ALU:n	$ALU(V) \rightarrow V$
0 1	V tas från bit 1 på bussen	$b_1 \rightarrow V$
1 0	Återställning (nollställning) av V	$0 \rightarrow V$
1 1	V återförs (ändras ej)	

$g_7 g_6$	Z väljs enligt:	RTN
0 0	Z tas från ALU:n	$ALU(Z) \rightarrow Z$
0 1	Z tas från bit 2 på bussen	$b_2 \rightarrow Z$
1 0	Återställning (nollställning) av Z	$0 \rightarrow Z$
1 1	Z återförs (ändras ej)	

$g_9 g_8$	N väljs enligt:	RTN
0 0	N tas från ALU:n	$ALU(N) \rightarrow N$
0 1	N tas från bit 3 på bussen	$b_3 \rightarrow N$
1 0	Återställning (nollställning) av N	$0 \rightarrow N$
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	$b_4 \rightarrow I$
1 0	I återställs (ettställs)	$1 \rightarrow I$
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	$0 \rightarrow C_{in}$
0 1	konstant värde 1	$1 \rightarrow C_{in}$
1 0	C-flagga från CC	$C \rightarrow C_{in}$
1 1	C-flagga invers från CC	$C' \rightarrow C_{in}$

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	$M(TA)$

Villkorstest

Relation	Betydelse	Datatyper	Flaggvillkor	Instruktion
<i>Enkla flaggtest</i>				
$==$	Lika	<code>signed/unsigned</code>	$Z=1$	BEQ
$!=$	Skild från	<code>signed/unsigned</code>	$Z=0$	BNE
<0	Mindre än 0, (Negativt)	<code>signed</code>	$N=1$	BMI
≥ 0	Större än, eller lika med 0	<code>signed</code>	$N=0$	BPL
	Tvåkomplementspill	<code>signed</code>	$V=1$	BVS
	Inget tvåkomplementspill	<code>signed</code>	$V=0$	BVC
<i>Jämförelse av tal utan tecken</i>				
$>$	Större än	<code>unsigned</code>	$C + Z = 0$	BHI
\geq	Större än eller lika	<code>unsigned</code>	$C=0$	BCC
$<$	Mindre än	<code>unsigned</code>	$C=1$	BCS
\leq	Mindre än eller lika	<code>unsigned</code>	$C + Z = 1$	BLS
<i>Jämförelse av tal med tecken</i>				
$>$	Större än	<code>signed</code>	$Z + (N \oplus V) = 0$	BGT
\geq	Större än eller lika	<code>signed</code>	$N \oplus V = 0$	BGE
$<$	Mindre än	<code>signed</code>	$N \oplus V = 1$	BLT
\leq	Mindre än eller lika	<code>signed</code>	$Z + (N \oplus V) = 1$	BLE

ASCII-tabellen

Hex	ASCII														
00	NUL	10	DLE	20		30	0	40	©	50	P	60	`	70	p
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	"	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	\$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u
06	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v
07	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w
08	BS	18	CAN	28	(38	8	48	H	58	X	68	h	78	x
09	HT	19	EM	29)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[6B	k	7B	{
0C	FF	1C	FS	2C	,	3C	<	4C	L	5C	\	6C	l	7C	
0D	CR	1D	GS	2D	-	3D	=	4D	M	5D]	6D	m	7D	}
0E	SO	1E	RS	2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
0F	S1	1F	US	2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL