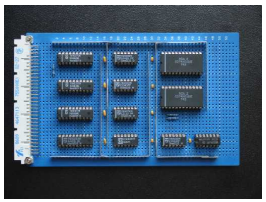
Digital- och datorteknik



Föreläsning #15

Biträdande professor Jan Jonsson

Institutionen för data- och informationsteknik
Chalmers tekniska högskola

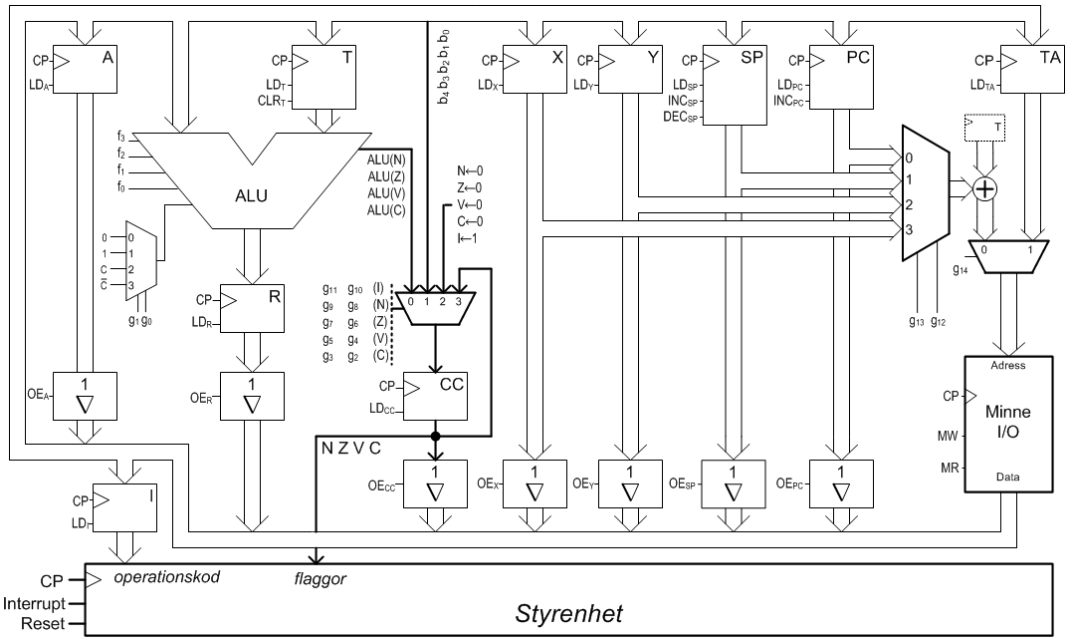
FLIS-processorn

Dataväg med pekarregister och stackpekare:

I vår sjunde, och slutliga, version av datavägen har vi fått ytterligare tre pekarregister: X, Y och SP.

Register X och Y används generellt till att referera till datastrukturer i primärminnet, t ex listor, tabeller eller köer.

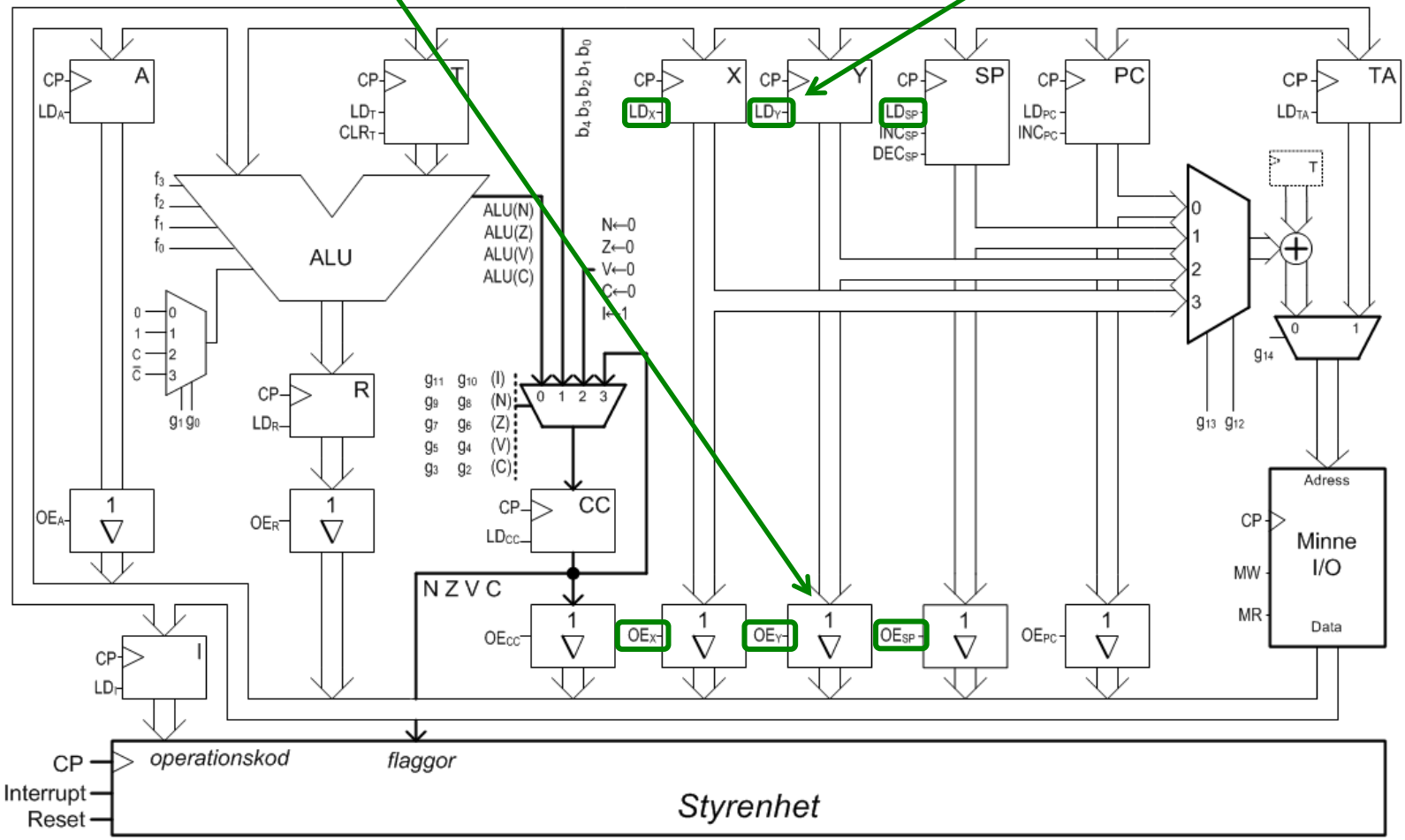
Stackpekaren, SP, används som referens till en datastruktur, kallad ”stacken”, där temporära variabler och programflödesadresser lagras.



FLIS-processorn

OE_X, OE_Y och OE_{SP} väljer om innehållet i respektive register skall läggas ut på bussen.

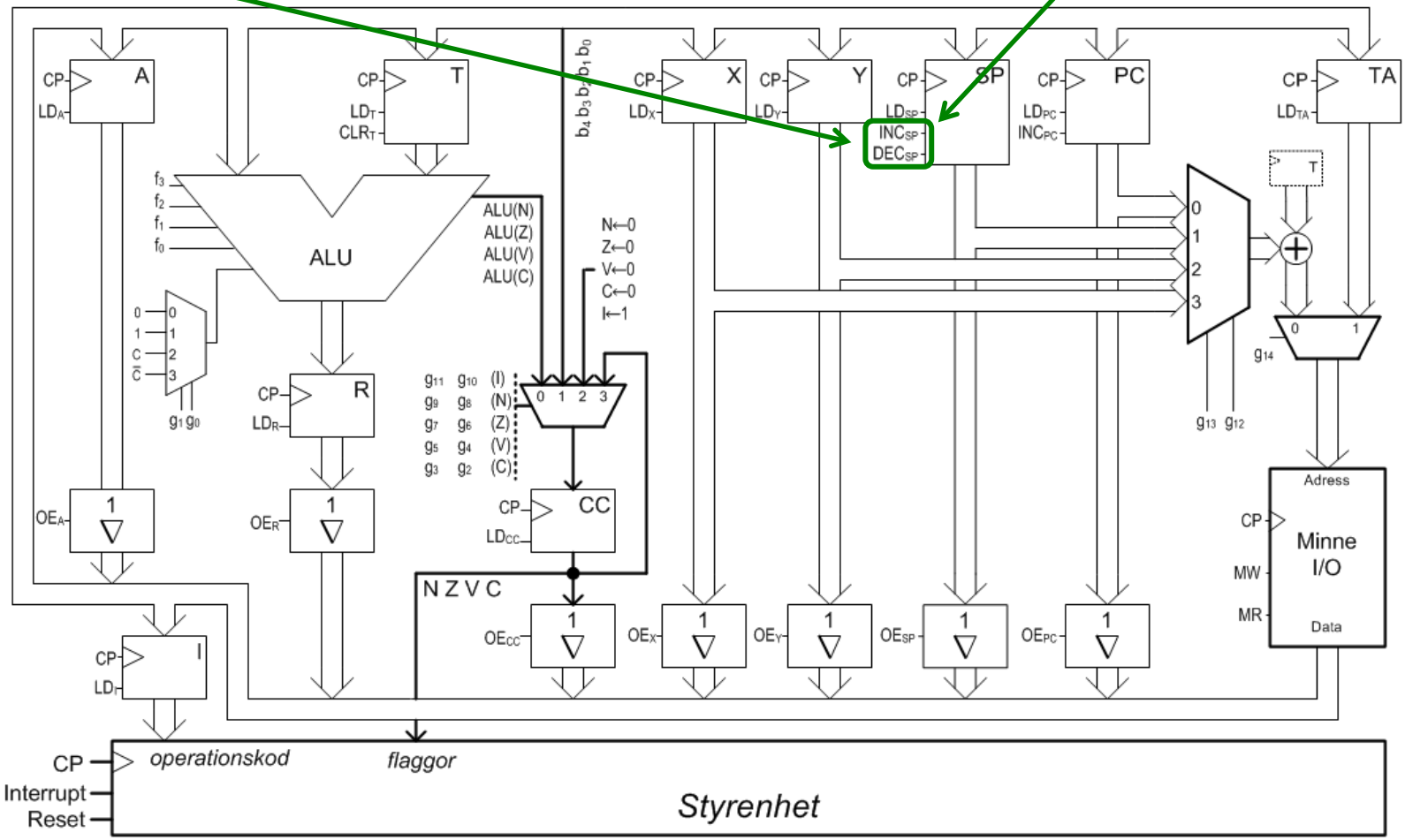
LD_X, LD_Y och LD_{SP} väljer om respektive register skall uppdateras eller ej.



FLIS-processorn

Operationer på ”stacken” sker så ofta att man har speciella styrsignaler för att ändra värdet på SP.

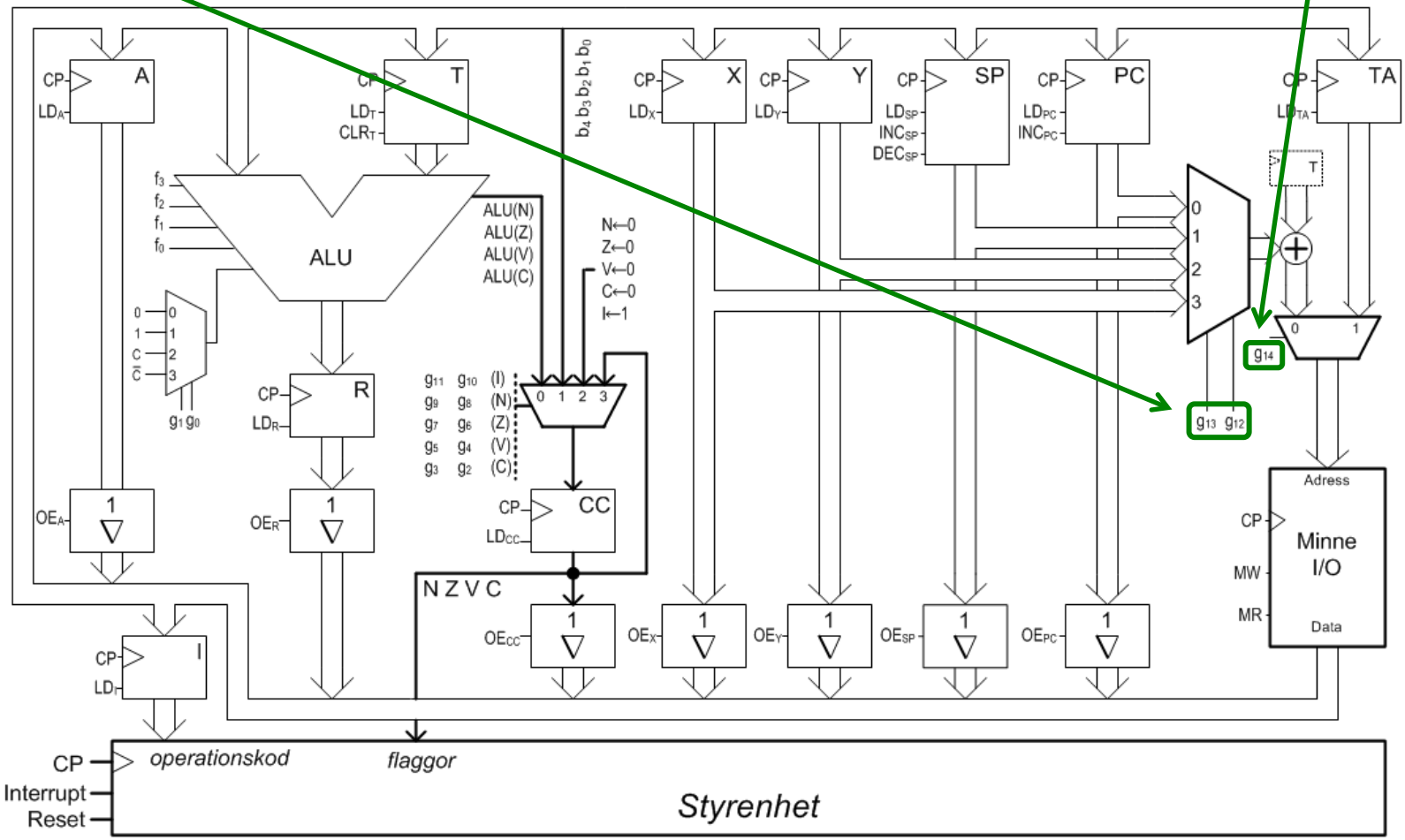
INC_{SP} och DEC_{SP} anger om värdet i SP skall ökas resp. minskas med 1.



FLIS-processorn

g_{13} och g_{12} anger vilket av pekarregistren X, Y, SP, eller PC som adresserar minnet när g_{14} = 0.

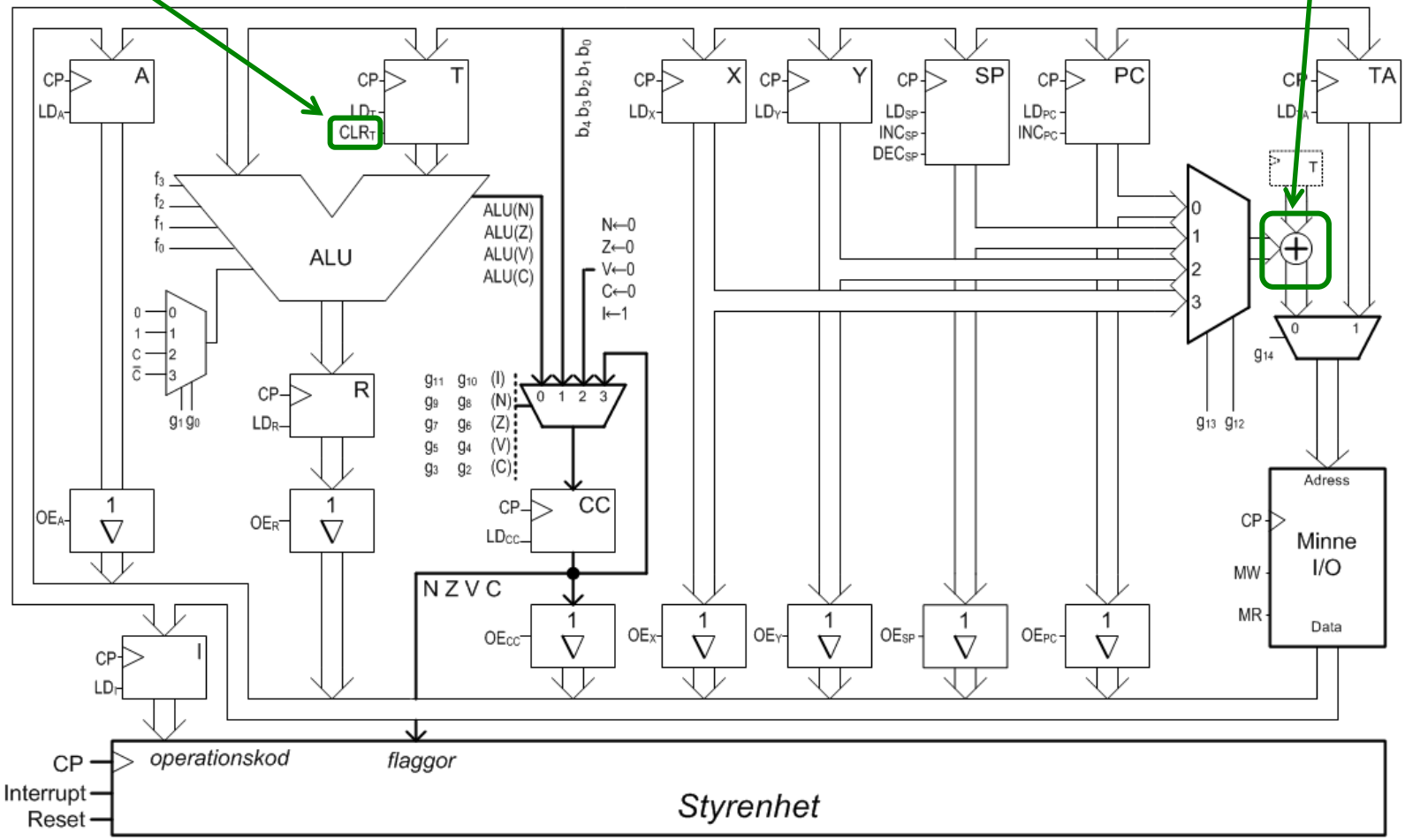
g_{14} anger om minnet skall adresseras via TA (g_{14}=1) eller via något av de andra pekarregistren (g_{14} = 0) .



FLIS-processorn

Värdet 0 behövs så ofta i register T att man har en speciell styrsignal CLR_T för att nollställa registret.

Värdet i T adderas alltid till värdet i det pekarregister (X, Y, SP eller PC) som valts med g_{13} och g_{12}.

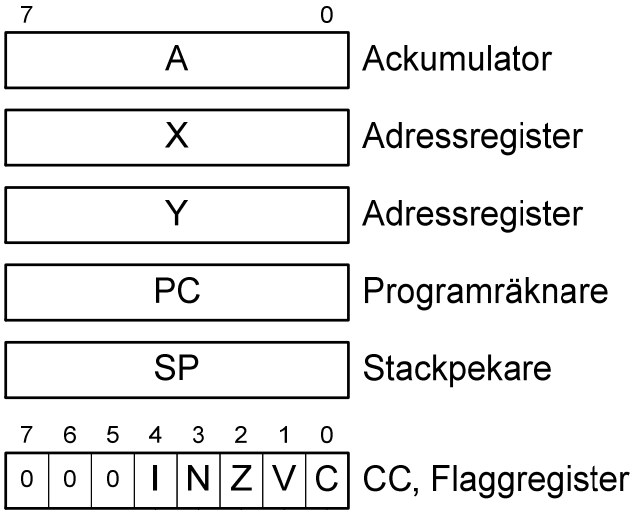


FLIS-processorn

Programmerarens modell av datorn:

Operationer på datavägen ges med hjälp av ett maskinspråk, vilket är en uppsättning maskininstruktioner (binära kodord) som är avsedda att avkodas av datorn. Den, för människan, läsbara representationen av ett maskinspråk kallas assemblerspråk.

För att reducera mängden detaljer, och därmed också reducera risken för att införa fel, kan den som programmerar datorn med hjälp av maskinspråk bara direkt referera till en delmängd av datavägens register.



FLIS-processorn

Var lagrar vi instruktionerna?

Sett från programmerarens perspektiv:

Ett datorprogram består av sekvenser av maskininstruktioner.

Maskininstruktionerna lagras i primärminnet, i enlighet med Turings/von Neumanns ”det lagrade programmets princip”.

Sett från datavägens perspektiv:

Till varje operationskod i en maskininstruktion hör en sekvens av RTN-operationer (styrsignaler).

RTN-operationerna lagras i den automatiska styrenheten, och utgör för programmeraren en icke-synlig och icke-modifierbar del av datorns hårdvara.

FLIS-processorn

Den automatiska styrenhetens tre faser:

- Återställningsfas (RESET): Adressen till programmets första maskininstruktion hämtas från datorns resetvektor (minnescellen på adress FF_{16}) och lagras i PC.
- Hämtfas (FETCH): En operationskod hämtas (från den minnescell vars adress ligger i PC) och lagras i register I.
- Utförandefas (EXECUTE): Innehållet i register I (en operationskod) avkodas och den till operationskoden tillhörande sekvensen av RTN-operationer genomlöps.

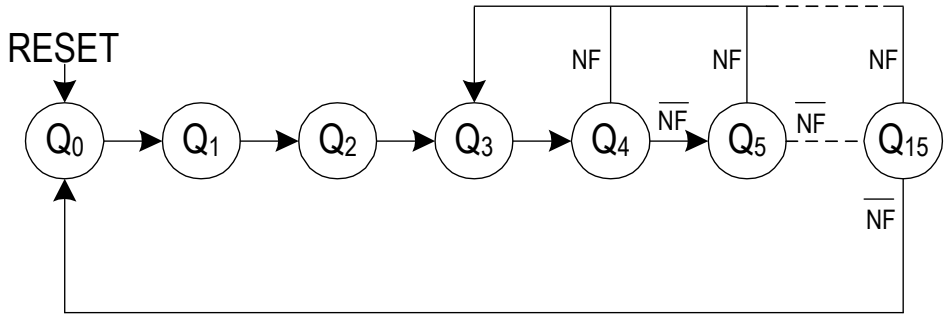
Om styrenheten skall fungera som tänkt måste minnet alltså innehålla

1. adressen till första maskininstruktion i minnescellen på adress FF_{16}
2. en maskininstruktion i varje minnescell vars adress kan komma att lagras i PC

FLIS-processorn

Den automatiska styrenhetens tre faser:

Sekvensnätet som realiserar styrenhetens tre faser har följande tillståndsgraf:



Tillstånd Q_0, Q_1 och Q_2 motsvarar RESET-fasen

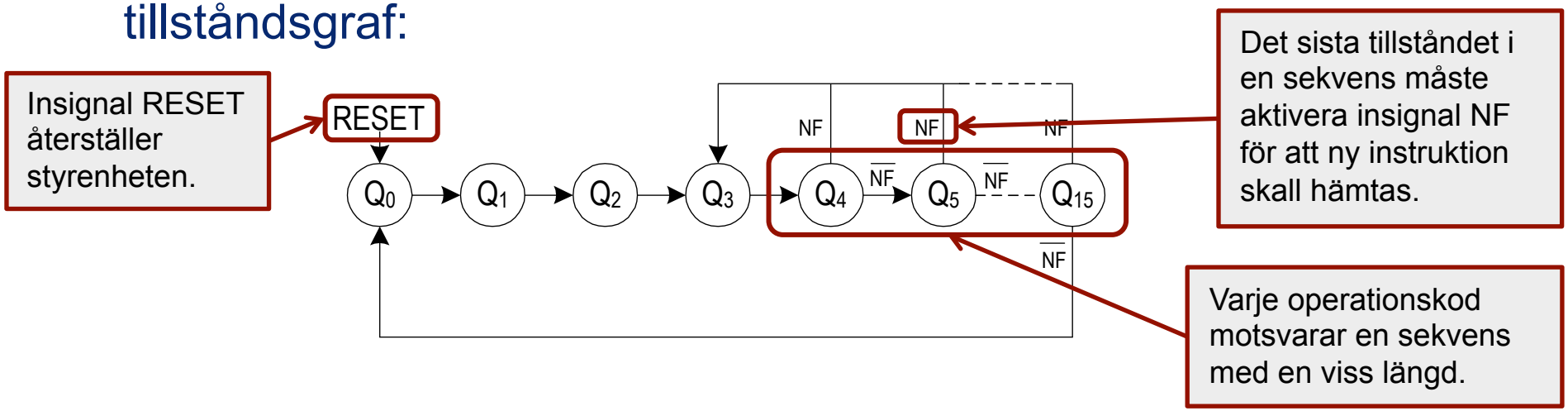
Tillstånd Q_3 motsvarar FETCH-fasen

Tillstånd Q_4 … Q_{15} reserveras för de sekvenser av RTN-operationer som utförs i EXECUTE-fasen för aktuell maskininstruktion

FLIS-processorn

Den automatiska styrenhetens tre faser:

Sekvensnätet som realiserar styrenhetens tre faser har följande tillståndsgraf:



Tillstånd Q_0, Q_1 och Q_2 motsvarar RESET-fasen

Tillstånd Q_3 motsvarar FETCH-fasen

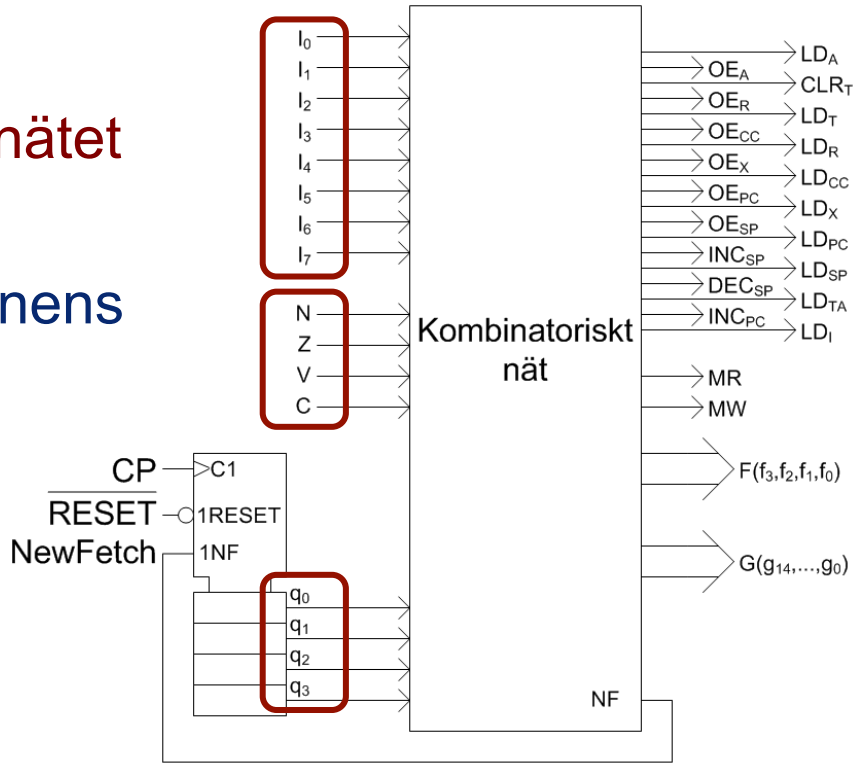
Tillstånd Q_4 … Q_{15} reserveras för de sekvenser av RTN-operationer som utförs i EXECUTE-fasen för aktuell maskininstruktion

FLIS-processorn

Hur genereras styrsignalerna (RTN-operationerna)?

Styrsignalerna till datavägen genereras av ett kombinatoriskt nät som har följande insignaler:

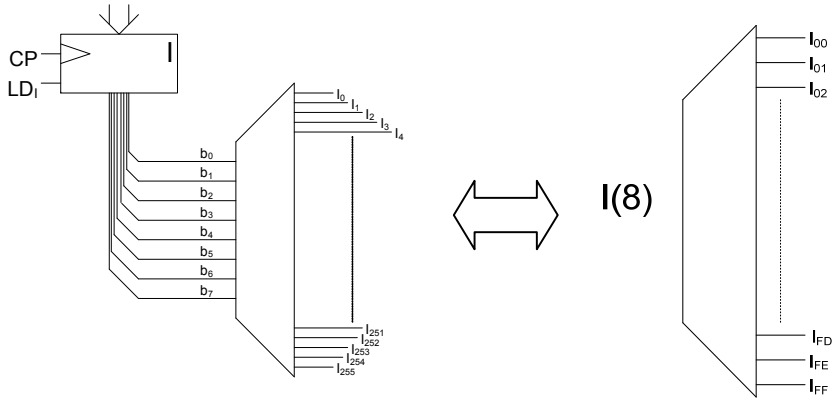
- Tillståndssignalerna i sekvensnätet
- Innehållet i register I
  (= nuvarande maskininstruktionens operationskod)
- Datavägens flaggbitar



FLIS-processorn

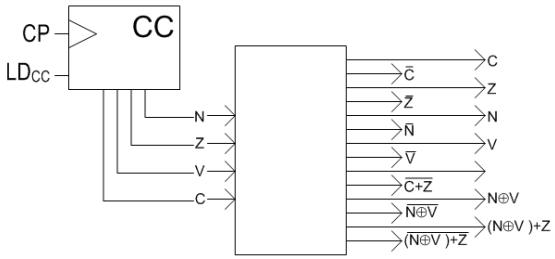
Hur genereras styrsignalerna (RTN-operationerna)?

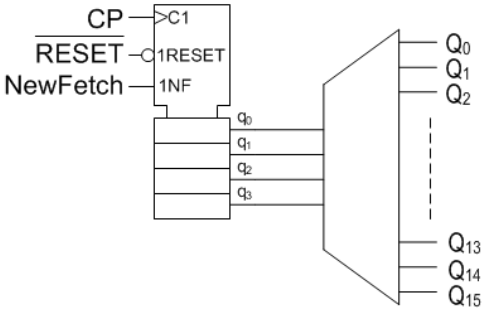
Insignalerna till styrenhetens kombinatoriskt nät genererar i sin tur, via olika avkodare, ett antal interna generatorsignaler.

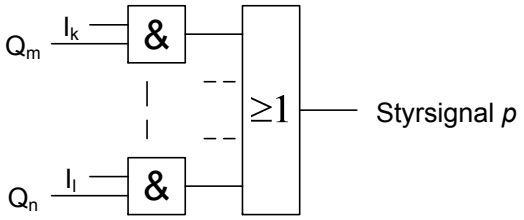


Generatorsignalerna kombineras via AND/OR-nät så att en viss styrsignal aktiveras för en viss instruktion I_x vid ett visst tillstånd Q_y.

Styrsignaler kan även aktiveras enbart för en viss kombination av flaggbitar (vid villkorliga hoppinstruktioner).







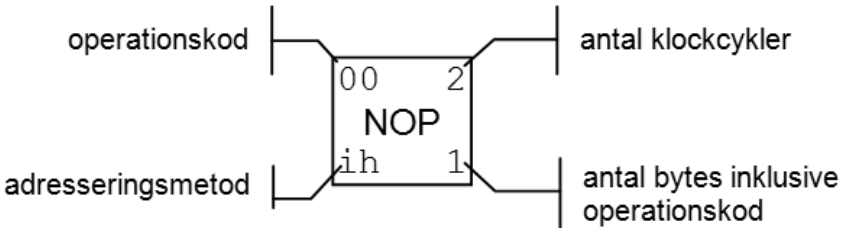
FLIS-processorn

Adresseringsmetoder

En maskininstruktions adresseringmetod ges av instruktionens operationskod, och uttrycker läge (”source” & ”destination”) eller värde för instruktionens operander.

Adresseringmetoden påverkar därför

- # bytes som behövs för att lagra maskininstruktionen i minnet
- längden (# klockcykler) på den sekvens av tillstånd som maskininstruktionen utnyttjar i styrenhetens EXECUTE-fas



Se sidan 11 i
”Instruktionslista för FLISP”

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

FLIS-processorn

Adresseringsmetoder (se Instruktionslista för FLISP sid. 5-8)

Inherent

Information om operandens nya värde anges av själva operationskoden.

T ex: CLA $0 \rightarrow A$

INCA $A + 1 \rightarrow A$

Immediate

Operandens värde anges av instruktionens operandinfo.

T ex: LDA #$FF $FF_{16} \rightarrow A$

FLIS-processorn

Adresseringsmetoder (se Instruktionslista för FLISP sid. 5-8)

Absolute (kallas ibland ”direct”)

Operandens, eller hoppdestinationens, läge i primärminnet (den s k effektivadressen, EA) anges av instruktionens operandinfo.

T ex: LDA $FF $M(FF_{16}) \rightarrow A$

JMP $58 $58_{16} \rightarrow PC$

Indirect

I denna adresseringmod (som dock ej finns på FLISP) anger operandinfo en adress i primärminnet där effektivadressen kan hämtas.

FLIS-processorn

Adresseringsmetoder (se Instruktionslista för FLISP sid. 5-8)

PC relative

EA utgörs av PC-registrets nuvarande värde plus en offset (i 2-komplementform). Värdet på offset anges av operandinfo.

T ex: BRA L1

Egentligen lyder instruktionen:

'BRA offset' där offset = { EA för L1 } – { värde i register PC }

Genom att ange hoppdestinationen relativt PC-registret kan programkoden placeras på godtycklig plats i minnet, s k positionsoberoende kod.

FLIS-processorn

Adresseringsmetoder (se Instruktionslista för FLISP sid. 5-8)

Register indirect

EA utgörs av en basadress plus en eventuell offset (i 2-komplementform). Varifrån basadress och offset hämtas anges av instruktionens operationskod och operandinfo.

Ett av registren X, Y och SP måste ingå i beräkningen av EA. Dessutom kan en konstant (för X, Y och SP) eller innehållet i register A (för X och Y) ingå.

FLIS-processorn

Adresseringsmetoder (se Instruktionslista för FLISP sid. 5-8)

Register indirect (forts.)

Ett av registren X, Y och SP måste ingå i beräkningen av EA. Dessutom kan en konstant (för X, Y och SP) eller innehållet i register A (för X och Y) ingå.

Exempel:

STA 2,X	EA = { värde i register X } + 2 ; A → M(EA)
LDSP A,Y	EA = { värde i register Y } + { värde i register A } ; M(EA) → SP
LEASP -8,SP	EA = { värde i register SP } – 8 ; EA → SP
JMP 0,X	EA = { värde i register X } + 0 ; EA → PC

FLIS-processorn

Adresseringsmetoder (se Instruktionslista för FLISP sid. 5-8)

Register indirect (forts.)

Observera att såväl ett register som en konstant kan utgöra basadress.

Exempel:

LDA $80,X	basadress = { konstanten 80_{16} } ; offset = { värde i register X }
LEAX A,Y	basadress = { värde i register A } ; offset = { värde i register Y }
LEASP -8,SP	basadress = { värde i register SP } ; offset = { konstanten -8 }