CHALMERS
UNIVERSITY OF TECHNOLOGY

# Digital- och datorteknik

Föreläsning #15

Biträdande professor Jan Jonsson

Institutionen för data- och informationsteknik
Chalmers tekniska högskola

CHALMERS
UNIVERSITY OF TECHNOLOGY

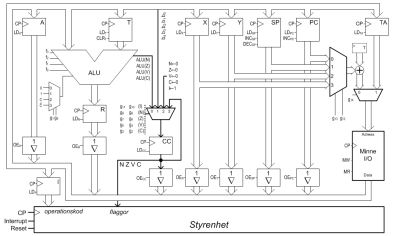
## FLIS-processorn

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.

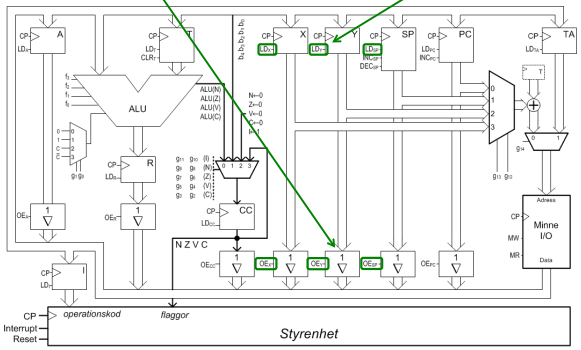



CHALMERS
UNIVERSITY OF TECHNOLOGY

## 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.

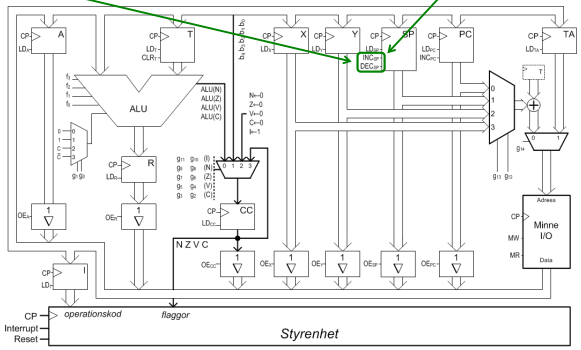



CHALMERS
UNIVERSITY OF TECHNOLOGY

## 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.

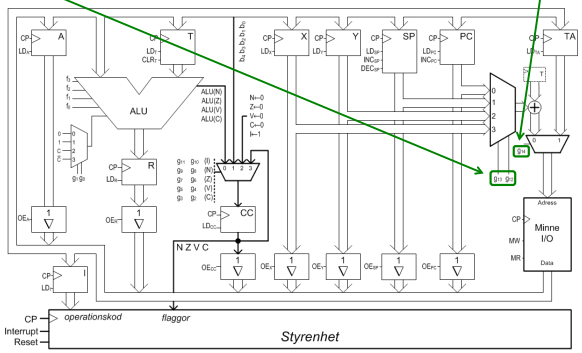


CHALMERS

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

## FLIS-processorn

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

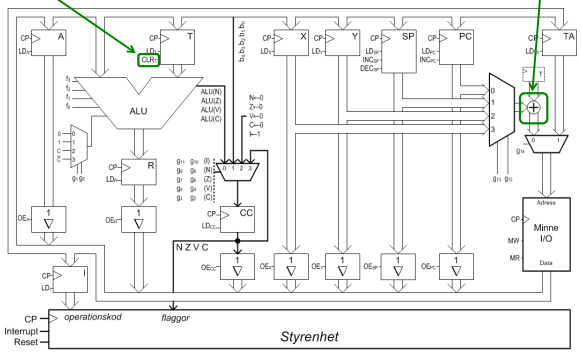



CHALMERS

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

## FLIS-processorn

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



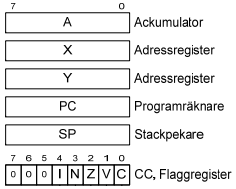

CHALMERS

## 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.




CHALMERS

## 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.

CHALMERS

# 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

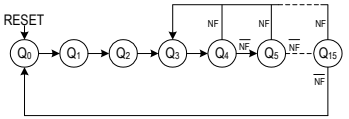
1. adressen till första maskininstruktion i minnescellen på adress $FF_{16}$
2. en maskininstruktion i varje minnescell vars adress kan komma att lagras i PC

CHALMERS

# FLIS-processorn

## Den automatiska styrenhetens tre faser:

Sekvensnätet som realiserar styrenhetens tre faser har följande tillståndsgraf:



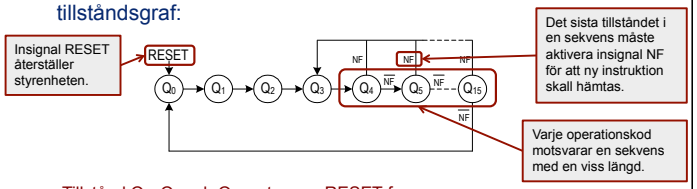

Tillstånd $Q_0$, $Q_1$ och $Q_2$ motsvarar RESET-fasen
Tillstånd $Q_3$ motsvarar FETCH-fasen
Tillstånd $Q_4$ … $Q_{15}$ reserveras för de sekvenser av RTN-operationer som utförs i EXECUTE-fasen för aktuell maskininstruktion

CHALMERS

# 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

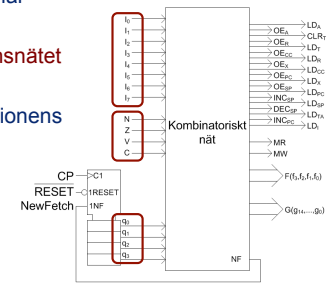
CHALMERS

# 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

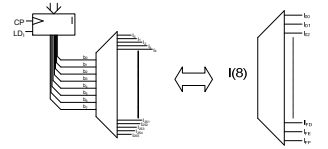


CHALMERS

# FLIS-processorn

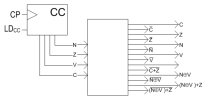
## Hur genereras styrsignalerna (RTN-operationerna)?

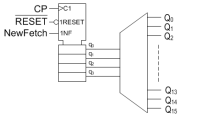
Insignalerna till styrenhetens kombinatoriskt nät genererar i sin tur, via olika avkodare, ett antal interna generatorsignaler.

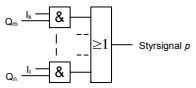



Generatorsignalerna kombineras via AND/OR-nät så att en viss styrsignal aktiveras för en viss instruktion $I_x$ vid ett visst tillstånd $Q_y$.

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




CHALMERS

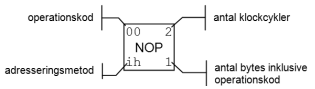
# FLIS-processorn

## 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




CHALMERS

Se sidan 11 i "Instruktionslista för FLISP"

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

CHALMERS

# FLIS-processorn

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

### Inherent

Information om operandens nya värde anges av själva operationskoden.

T ex: CLA 0 → A

INCA A + 1 → A

### Immediate

Operandens värde anges av instruktionens operandinfo.

T ex: LDA #$FF $FF_{16}$ → A

CHALMERS

## 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.

CHALMERS

## 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.

CHALMERS

## 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.

CHALMERS

## 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 |

CHALMERS
UNIVERSITY OF TECHNOLOGY

# 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 } |