CHALMERS

# Digital- och datorteknik

Föreläsning #11

Biträdande professor Jan Jonsson

Institutionen för data- och informationsteknik
Chalmers tekniska högskola

CHALMERS

## Datavägen

### Datavägens olika delar:

Vår dataväg kommer att innehålla följande delar:

- Databuss:
  En gemensam kommunikationskanal för utbyte av data.
- Register:
  En liten uppsättning minneselement för korttidslagring av data.
- ALU:
  En beräkningsenhet som utför aritmetiska och logiska operationer
- Primärminne:
  En större uppsättning minneselement för lagring av data
- Styrsignaler:
  Signaler som aktiverar dataöverföring till och från register och primärminne samt väljer operationer och operander för ALU

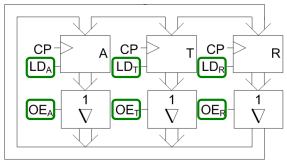
CHALMERS

## Datavägen

### Dataregister:

Vår första version av datavägen innehåller tre dataregister: A, T och R.
I denna version kan vi bara utföra överföring av data mellan register.
De styrsignaler som används här är:

- LD (välj register vars innehåll skall uppdateras från databuss)
- OE (välj register vars innehåll skall läggas ut på databuss)

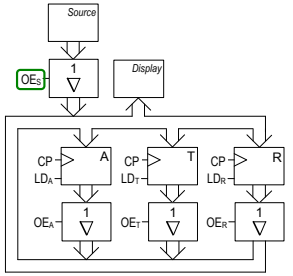



CHALMERS

## Datavägen

### Dataregister och källregister:

I vår andra version av datavägen lägger vi till ett källregister 'Source' för att kunna ladda våra register med konstanta datavärden.
En ny styrsignal, $OE_S$, behövs för att lägga ut innehållet i källregistret på databussen.




Källregistret kommer så småningom att försvinna, i samband med att vi får tillgång till instruktioner i vårt assemblerspråk som kan ladda konstanta datavärden i register.

CHALMERS

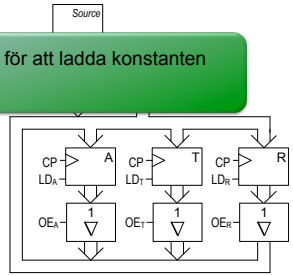
# Datavägen

## Dataregister och källregister:

I vår andra version av datavägen lägger vi till ett källregister 'Source' för att kunna ladda våra register med konstanta datavärden.

En ny styrsignal, $OE_S$, behövs för att lägga ut innehållet i källregistret på databussen.

Ange de styrsignaler som krävs för att ladda konstanten $2C_{16}$ i register A.




Källregistret kommer så småningom att försvinna, i samband med att vi får tillgång till instruktioner i vårt assemblerspråk som kan ladda konstanta datavärden i register.

CHALMERS

# Datavägen

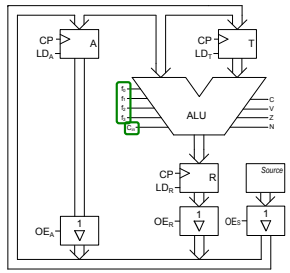
## Dataregister, källregister och ALU:

I vår tredje version av datavägen lägger vi till en ALU för att kunna utföra aritmetiska och logiska operationer på våra data. Utgången från ALU ansluts till register R, och ena ingången på ALU ansluts till register T.

Nya styrsignaler som läggs till:

- $f_3,f_2,f_1,f_0$ för att välja operation i ALU
- $C_{in}$ för att sätta "carry-in" till ALU

Observera att transmissionsgrindarna för register T nu har försvunnit, då registrets innehåll inte längre behöver läggas ut på databussen.




CHALMERS

# ALU

## FLISP ALU:

Den ALU som vi kommer att använda framöver har stöd för följande operationer. Se sidan 44 i "Instruktionslista för FLISP".

Observera att detta skiljer sig från den ALU som ni kommer att använda för Laboration 2.

| funktion | | | | operation | utsignaler | | | | | | | | N | Z | V | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $f_3$ | $f_2$ | $f_1$ | $f_0$ | RTN | $u_7$ | $u_6$ | $u_5$ | $u_4$ | $u_3$ | $u_2$ | $u_1$ | $u_0$ | | | | |
| 0 | 0 | 0 | 0 | U= 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | U= $FD_{16}$ | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | U= $FE_{16}$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | U= $FF_{16}$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | U= E | $e_7$ | $e_6$ | $e_5$ | $e_4$ | $e_3$ | $e_2$ | $e_1$ | $e_0$ | $u_7$ | (7) | (6) | (5) |
| 0 | 1 | 0 | 1 | U= $D_{15..0}+C_{in}$ | | | | | | | | | $u_7$ | (7) | (6) | (5) |
| 0 | 1 | 1 | 0 | U= D∨E | $d_7 \vee e_7$ | $d_6 \vee e_6$ | $d_5 \vee e_5$ | $d_4 \vee e_4$ | $d_3 \vee e_3$ | $d_2 \vee e_2$ | $d_1 \vee e_1$ | $d_0 \vee e_0$ | $u_7$ | (7) | 0 | 0 |
| 0 | 1 | 1 | 1 | U= D∧E | $d_7 \wedge e_7$ | $d_6 \wedge e_6$ | $d_5 \wedge e_5$ | $d_4 \wedge e_4$ | $d_3 \wedge e_3$ | $d_2 \wedge e_2$ | $d_1 \wedge e_1$ | $d_0 \wedge e_0$ | $u_7$ | (7) | 0 | 0 |
| 1 | 0 | 0 | 0 | U= D⊕E | $d_7 \oplus e_7$ | $d_6 \oplus e_6$ | $d_5 \oplus e_5$ | $d_4 \oplus e_4$ | $d_3 \oplus e_3$ | $d_2 \oplus e_2$ | $d_1 \oplus e_1$ | $d_0 \oplus e_0$ | $u_7$ | (7) | 0 | 0 |
| 1 | 0 | 0 | 1 | U= D+$C_{in}$ | | | | | | | | | $u_7$ | (7) | (6) | (5) |
| 1 | 0 | 1 | 0 | U= D+$FF_{16}$ | | | | | | | | | $u_7$ | (7) | (6) | (5) |
| 1 | 0 | 1 | 1 | U= D+E+$C_{in}$ | | | | | | | | | $u_7$ | (7) | (6) | (5) |
| 1 | 1 | 0 | 0 | U= D+$E_{15..0}+C_{in}$ | | | | | | | | | $u_7$ | (7) | (6) | (5) |
| 1 | 1 | 0 | 1 | U= D<<1 ($C_{in}$) | $d_6$ | $d_5$ | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $d_0$ | $C_{in}$ | $u_7$ | (7) | (6) | (5) |
| 1 | 1 | 1 | 0 | U= ($C_{in}$) 1>>D | $C_{in}$ | $d_7$ | $d_6$ | $d_5$ | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $u_7$ | (7) | (6) | (5) |
| 1 | 1 | 1 | 1 | U= ($d_7$)1>>D | $d_7$ | $d_7$ | $d_6$ | $d_5$ | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $u_7$ | (7) | 0 | (5) |

För kommentarer se sidan 44 i "Instruktionslista för FLISP"

CHALMERS

# ALU

## FLISP ALU:

Den ALU som vi kommer att använda framöver har stöd för följande operationer. Se sidan 44 i "Instruktionslista för FLISP".

Observera att detta skiljer sig från den ALU som ni kommer att använda för laboration 2.

Ange den styrsignalsekvens som krävs för att öka innehållet i register A med 1.

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | U= $FF_{16}$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | U= E | $e_7$ | $e_6$ | $e_5$ | $e_4$ | $e_3$ | $e_2$ | $e_1$ | $e_0$ | $u_7$ | (7) | 0 | 0 |
| 0 | 1 | 0 | 1 | U= $D_{15..0}+C_{in}$ | | | | | | | | | $u_7$ | (7) | (6) | (5) |
| 0 | 1 | 1 | 0 | U= D∨E | $d_7 \vee e_7$ | $d_6 \vee e_6$ | $d_5 \vee e_5$ | $d_4 \vee e_4$ | $d_3 \vee e_3$ | $d_2 \vee e_2$ | $d_1 \vee e_1$ | $d_0 \vee e_0$ | $u_7$ | (7) | 0 | 0 |
| 0 | 1 | 1 | 1 | U= D∧E | $d_7 \wedge e_7$ | $d_6 \wedge e_6$ | $d_5 \wedge e_5$ | $d_4 \wedge e_4$ | $d_3 \wedge e_3$ | $d_2 \wedge e_2$ | $d_1 \wedge e_1$ | $d_0 \wedge e_0$ | $u_7$ | (7) | 0 | 0 |
| 1 | 0 | 0 | 0 | U= D⊕E | $d_7 \oplus e_7$ | $d_6 \oplus e_6$ | $d_5 \oplus e_5$ | $d_4 \oplus e_4$ | $d_3 \oplus e_3$ | $d_2 \oplus e_2$ | $d_1 \oplus e_1$ | $d_0 \oplus e_0$ | $u_7$ | (7) | 0 | 0 |
| 1 | 0 | 0 | 1 | U= D+$C_{in}$ | | | | | | | | | $u_7$ | (7) | (6) | (5) |
| 1 | 0 | 1 | 0 | U= D+$FF_{16}$ | | | | | | | | | $u_7$ | (7) | (6) | (5) |
| 1 | 0 | 1 | 1 | U= D+E+$C_{in}$ | | | | | | | | | $u_7$ | (7) | (6) | (5) |
| 1 | 1 | 0 | 0 | U= D+$E_{15..0}+C_{in}$ | | | | | | | | | $u_7$ | (7) | (6) | (5) |
| 1 | 1 | 0 | 1 | U= D<<1 ($C_{in}$) | $d_6$ | $d_5$ | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $d_0$ | $C_{in}$ | $u_7$ | (7) | (6) | (5) |
| 1 | 1 | 1 | 0 | U= ($C_{in}$) 1>>D | $C_{in}$ | $d_7$ | $d_6$ | $d_5$ | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $u_7$ | (7) | (6) | (5) |
| 1 | 1 | 1 | 1 | U= ($d_7$)1>>D | $d_7$ | $d_7$ | $d_6$ | $d_5$ | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $u_7$ | (7) | 0 | (5) |

CHALMERS

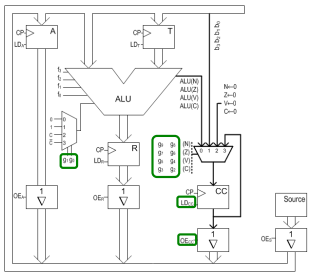
# Datavägen

## Dataregister, källregister, ALU och flaggregister:

I vår fjärde version av datavägen lägger vi till ett flaggregister, CC, med främsta syfte att spara de flaggbitar som vi får efter en ALU-operation.

Nya styrsignaler som läggs till:

- $g_9$,$g_8$ väljer N-flaggans nya värde
- $g_7$,$g_6$ väljer Z-flaggans nya värde
- $g_5$,$g_4$ väljer V-flaggans nya värde
- $g_3$,$g_2$ väljer C-flaggans nya värde
- $g_1$,$g_0$ väljer ”carry-in” till ALU
- $LD_{CC}$ för att välja om innehållet i CC skall uppdateras eller ej
- $OE_{CC}$ för att lägga ut innehållet i CC på databussen




CHALMERS

# Flaggregister

## Val av ”carry-in” och flaggsättning:

Styrsignalerna till de två väljarna för ALU ”carry-in” respektive flaggsättning i CC har följande betydelser.

Se sidan 48 i ”Instruktionslista för FLISP”.

Val av Carry in till ALU

| $g_1$ | $g_0$ | Signal till $C_{in}$ | RTN |
|---|---|---|---|
| 0 | 0 | konstant värde 0 | 0→ $C_{in}$ |
| 0 | 1 | konstant värde 1 | 1→ $C_{in}$ |
| 1 | 0 | C-flagga från CC | C→ $C_{in}$ |
| 1 | 1 | C-flagga invers från CC | C'→ $C_{in}$ |

Val av flaggsättning i CC

| $g_3$ | $g_2$ | **C** väljs enligt: | RTN |
|---|---|---|---|
| 0 | 0 | C tas från ALU:n | ALU(C)→C |
| 0 | 1 | C tas från bit 0 på bussen | $b_0$→ C |
| 1 | 0 | Återställning (nollställning) av C | 0→ C |
| 1 | 1 | C återförs (ändras ej) | |

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

| $g_7$ | $g_6$ | **Z** väljs enligt: | RTN |
|---|---|---|---|
| 0 | 0 | Z tas från ALU:n | ALU(Z)→ Z |
| 0 | 1 | Z tas från bit 2 på bussen | $b_2$→ Z |
| 1 | 0 | Återställning (nollställning) av C | 0→ Z |
| 1 | 1 | Z återförs (ändras ej) | |

| $g_9$ | $g_8$ | **N** väljs enligt: | RTN |
|---|---|---|---|
| 0 | 0 | V tas från ALU:n | ALU(N)→N |
| 0 | 1 | N tas från bit 3 på bussen | $b_3$→ N |
| 1 | 0 | Återställning (nollställning) av N | 0→ N |
| 1 | 1 | N återförs (ändras ej) | |

CHALMERS

# Flaggregister

## Val av ”carry-in” och flaggsättning:

Styrsignalerna till de två väljarna för ALU ”carry-in” respektive flaggsättning i CC har följande betydelser.

Se sidan 48 i ”Instruktionslista för FLISP”.

Ange den styrsignalsekvens som krävs för att 1-ställa C-flaggan.

Val av Carry in till ALU

| $g_1$ | $g_0$ | Signal till $C_{in}$ | RTN |
|---|---|---|---|
| 0 | 0 | konstant värde 0 | 0→ $C_{in}$ |
| 0 | 1 | konstant värde 1 | 1→ $C_{in}$ |
| 1 | 0 | C-flagga från CC | C→ $C_{in}$ |
| 1 | 1 | C-flagga invers från CC | C'→ $C_{in}$ |

Val av flaggsättning i CC

| $g_3$ | $g_2$ | **C** väljs enligt: | RTN |
|---|---|---|---|
| 0 | 0 | C tas från ALU:n | ALU(C)→C |
| 0 | 1 | C tas från bit 0 på bussen | $b_0$→ C |
| 1 | 0 | Återställning (nollställning) av C | 0→ C |
| 1 | 1 | C återförs (ändras ej) | |

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

| $g_7$ | $g_6$ | **Z** väljs enligt: | RTN |
|---|---|---|---|
| 0 | 0 | Z tas från ALU:n | ALU(Z)→ Z |
| 0 | 1 | Z tas från bit 2 på bussen | $b_2$→ Z |
| 1 | 0 | Återställning (nollställning) av C | 0→ Z |
| 1 | 1 | Z återförs (ändras ej) | |

| $g_9$ | $g_8$ | **N** väljs enligt: | RTN |
|---|---|---|---|
| 0 | 0 | V tas från ALU:n | ALU(N)→N |
| 0 | 1 | N tas från bit 3 på bussen | $b_3$→ N |
| 1 | 0 | Återställning (nollställning) av N | 0→ N |
| 1 | 1 | N återförs (ändras ej) | |

CHALMERS

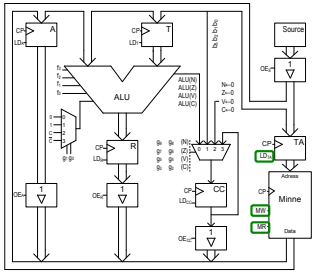
# Datavägen

## Dataregister, källregister, ALU, flaggregister och minne:

I vår femte version av datavägen lägger vi till ett primärminne för att kunna lagra större mängder data under en längre tid. För att peka ut exakt ett utav minnets register inför vi ett adressregister, TA.

Nya styrsignaler som läggs till:

- $LD_{TA}$ för att välja om innehållet i TA skall uppdateras eller ej.
- MR för att lägga ut innehållet i det valda minnesregistret på databussen
- MW för att uppdatera det valda minnesregistret med innehållet på databussen

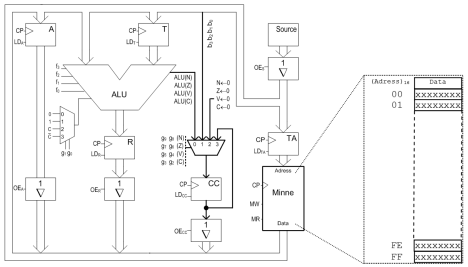


CHALMERS

# Primärminne

## Adressutrymme:

I vår dator består TA av n = 8 bitar, vilket innebär att vi kan adressera $2^8$ = 256 olika minnesregister.




CHALMERS

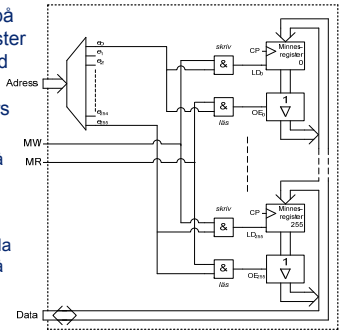
# Primärminne

## Minnesregister:

Varje minnesregister fungerar på samma sätt som våra dataregister A, T och R, d v s de laddas med en LD-signal, och avläses via en transmissionsgrind som styrs av en OE-signal.

Minnesregistren väljs ut baserat på minnesadressen med hjälp av en binäravkodare som genererar $2^8$ = 256 olika styrsignaler.

LD- och OE-signalerna för det valda registret tas sedan fram baserat på om det är en läsning (MR) eller skrivning (MW).




CHALMERS

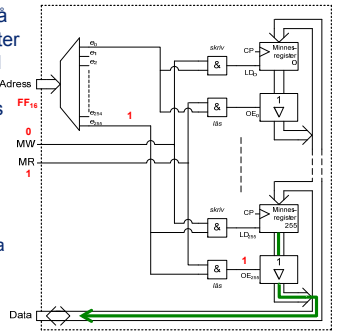
# Primärminne

## Minnesregister:

Varje minnesregister fungerar på samma sätt som våra dataregister A, T och R, d v s de laddas med en LD-signal, och avläses via en transmissionsgrind som styrs av en OE-signal.

Minnesregistren väljs ut baserat på minnesadressen med hjälp av en binäravkodare som genererar $2^8$ = 256 olika styrsignaler.

LD- och OE-signalerna för det valda registret tas sedan fram baserat på om det är en läsning (MR) eller skrivning (MW).




CHALMERS

# Datavägen

## Styrsignaler:

Vi har sett att man kan beskriva operationerna på datavägen i form av en tabell med styrsignaler. Varje rad i tabellen representerar då en operation som triggas med en klockpuls. Sekvenser av operationer som tar flera klockpulser i anspråk kan sålunda beskrivas med flera rader i tabellen.

Vad vi har fått nu är alltså ett program för att styra datavägen.

Vi har sett att man kan ange styrsignalerna i tabellen på två sätt:

- Kompakt form: enbart aktiva styrsignaler, d v s de med värdet '1'.
- Utförlig form: alla styrsignaler oavsett värde.

| RTN | steg | *Source* | $OE_S$ | $OE_A$ | $OE_B$ | $OE_{CC}$ | $LD_A$ | $LD_T$ | $LD_{16}$ | $LD_B$ | $LD_{CC}$ | $f_2$ | $f_1$ | $f_0$ | $g_8$ | $g_7$ | $g_6$ | $g_5$ | $g_4$ | $g_3$ | $g_2$ | $g_1$ | $g_0$ | MR | MW |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | | | | |