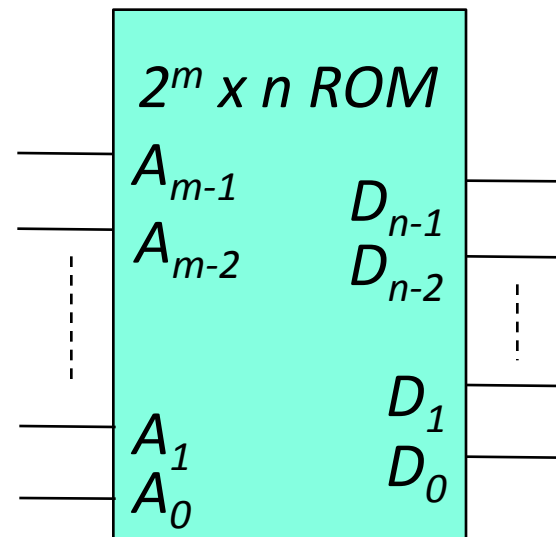


Läsminne Read Only Memory ROM

- Ett läsminne har addressingångar och datautgångar
- Med m addresslinjer kan man accessa 2^m olika minnesadresser
- På varje address finns det ett dataord på n bitar
- Oftast har ROM minnet också en Output Enable (OE) ingång



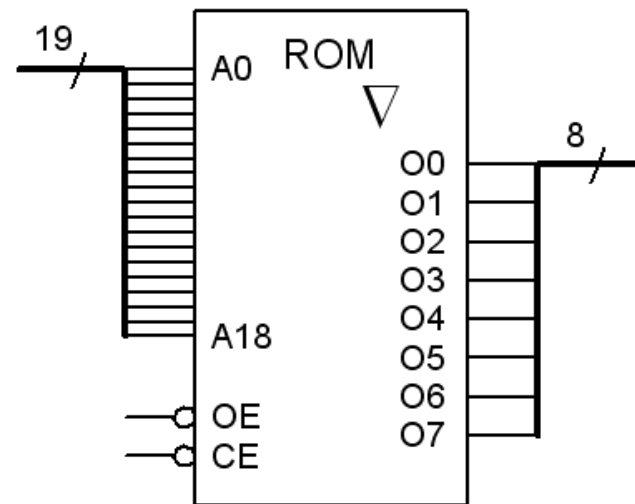
Läsminne Read Only Memory ROM

Exempel på ett ROM

Läsminne:



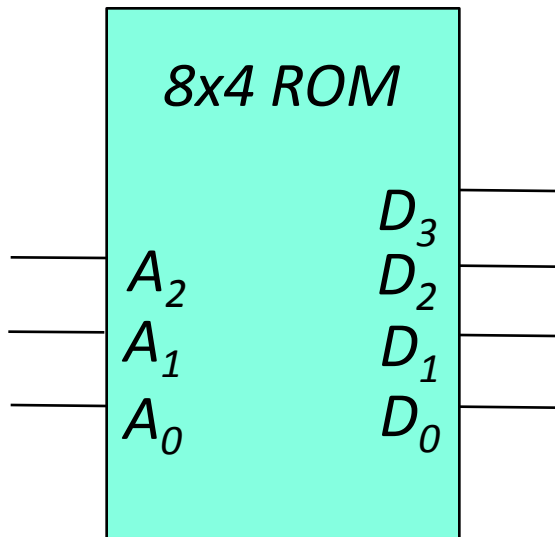
ROM 4M 512k $\times$ 8 bit



$\overline{CE}$ **Chip Enable** aktiverar chippet
 $\overline{OE}$ **Output Enable** kopplar in datautgångarna
(annars är dom i Three-state läge)

Litet ROM

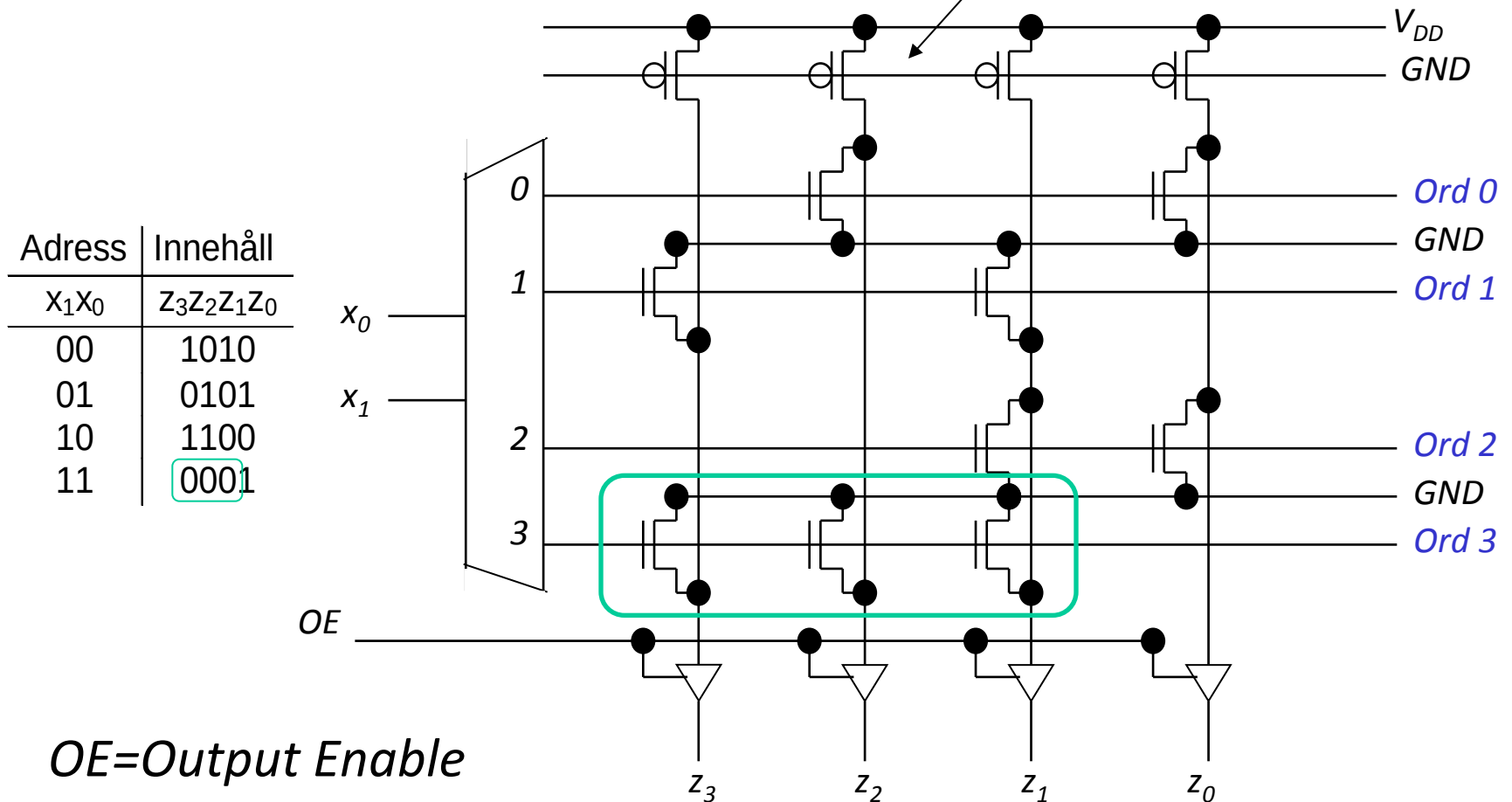
Möjlig minnesinnehåll



A_2	A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1	0
0	0	1	0	1	1	0
0	1	0	1	1	1	1
0	1	1	1	1	0	1
1	0	0	0	0	1	1
1	0	1	0	0	0	0
1	1	0	1	0	0	1
1	1	1	0	0	1	1

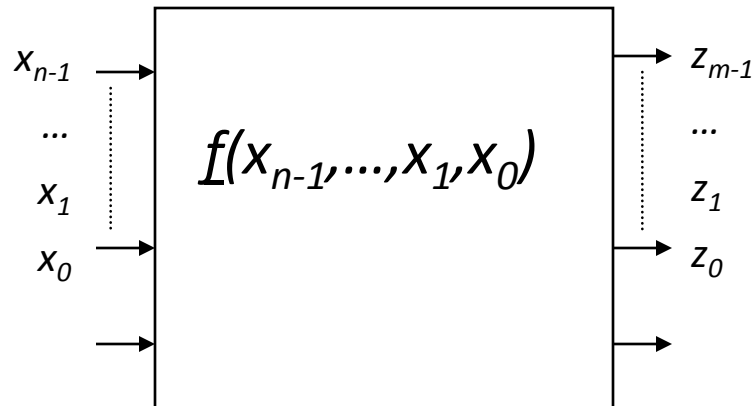
ROM

Pullup "1"



OE=Output Enable

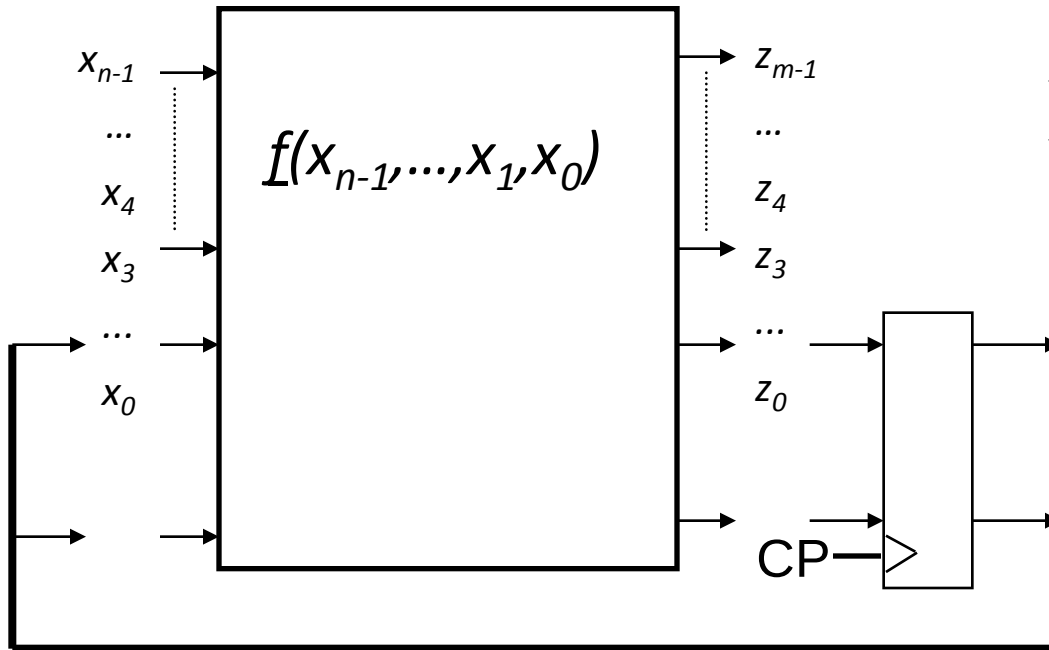
ROM implementering av kombinatoriska funktioner



Adress	Innehåll
$x_{n-1} \dots x_1 x_0$	$z_{m-1} \dots z_1 z_0$
0...00	1...10
0...01	0...01
...	...
1...11	0...01

Ett ROM med n ingångar med m utgångar kan användas för att implementera en kombinatorisk funktion med m utgångar och 2^n min-termer

ROM implementering av sekvenskrets



Adress	Innehåll
$x_{n-1} \dots x_3 x_2 x_1 x_0$	$z_{m-1} \dots z_3 z_2 z_1 z_0$
0...0000	1...0001
0...0001	0...0010
...	...
1...1111	0...0000

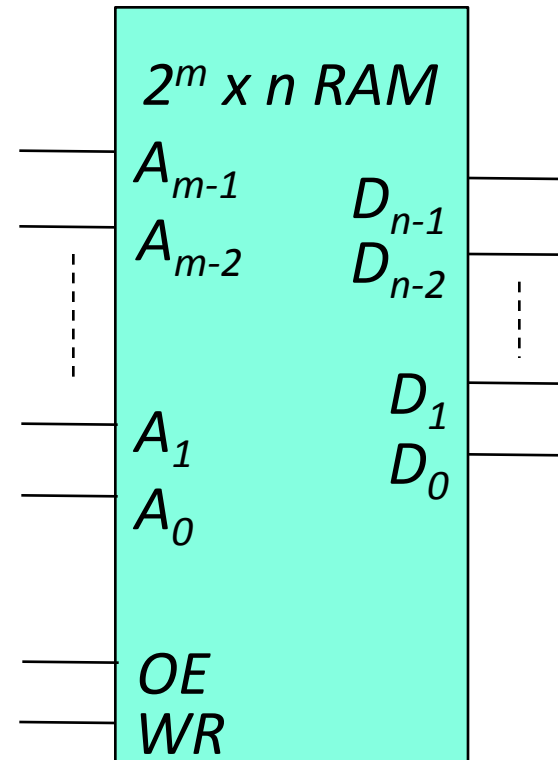
*En Moore-automat =
ett ROM och ett
register med D-vippor*

Mha återkopplingar kan ROMet användas till att generera sekvenser och implementera tillståndsmaskiner

Läs-Skriv-Minne

Random Access Memory RAM

- RAM-minnet har även en Write (WR) ingång som möjliggör att skriva in ett dataord på en given address
- $D_{n-1} \dots D_0$ är alltså *både* in- och utgångar

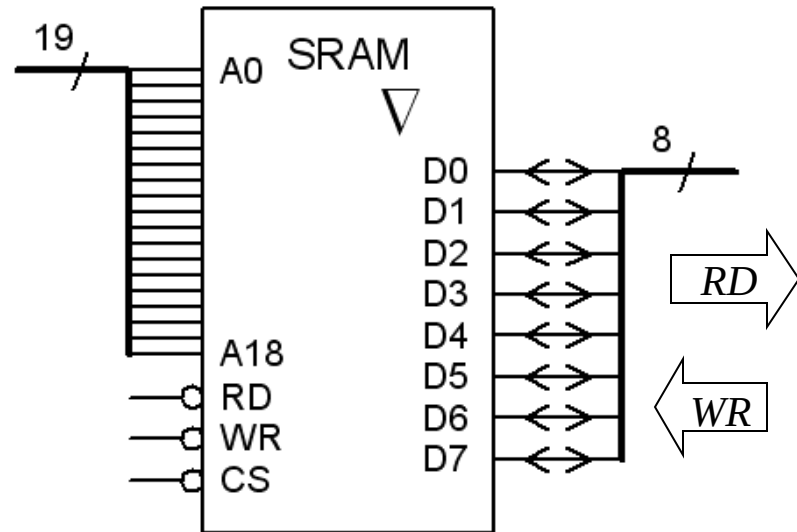


Läs-Skriv-Minne Random Access Memory RAM

Läs/Skrivminne:



SRAM 4M 512k $\times$ 8 bit



$\overline{CS}$

Chip Select aktiverar chippet

$\overline{RD}$

RD läsning från minnet, datautgångarna är aktiva

$\overline{WR}$

WR skrivning i minnet

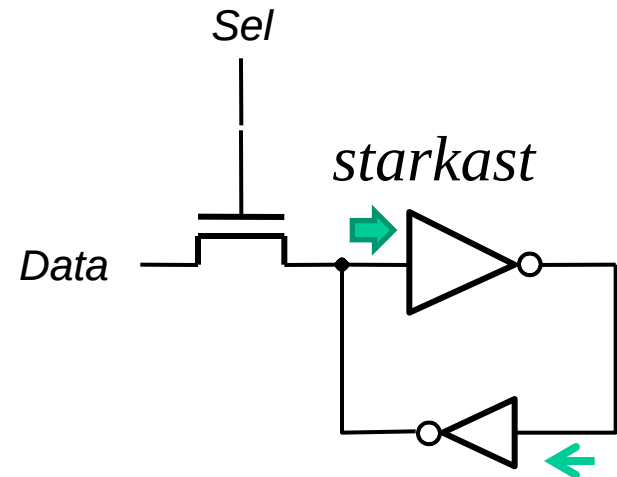
$\overline{WR}$

(vid write är datautgångarna i three-state läge)

SRAM

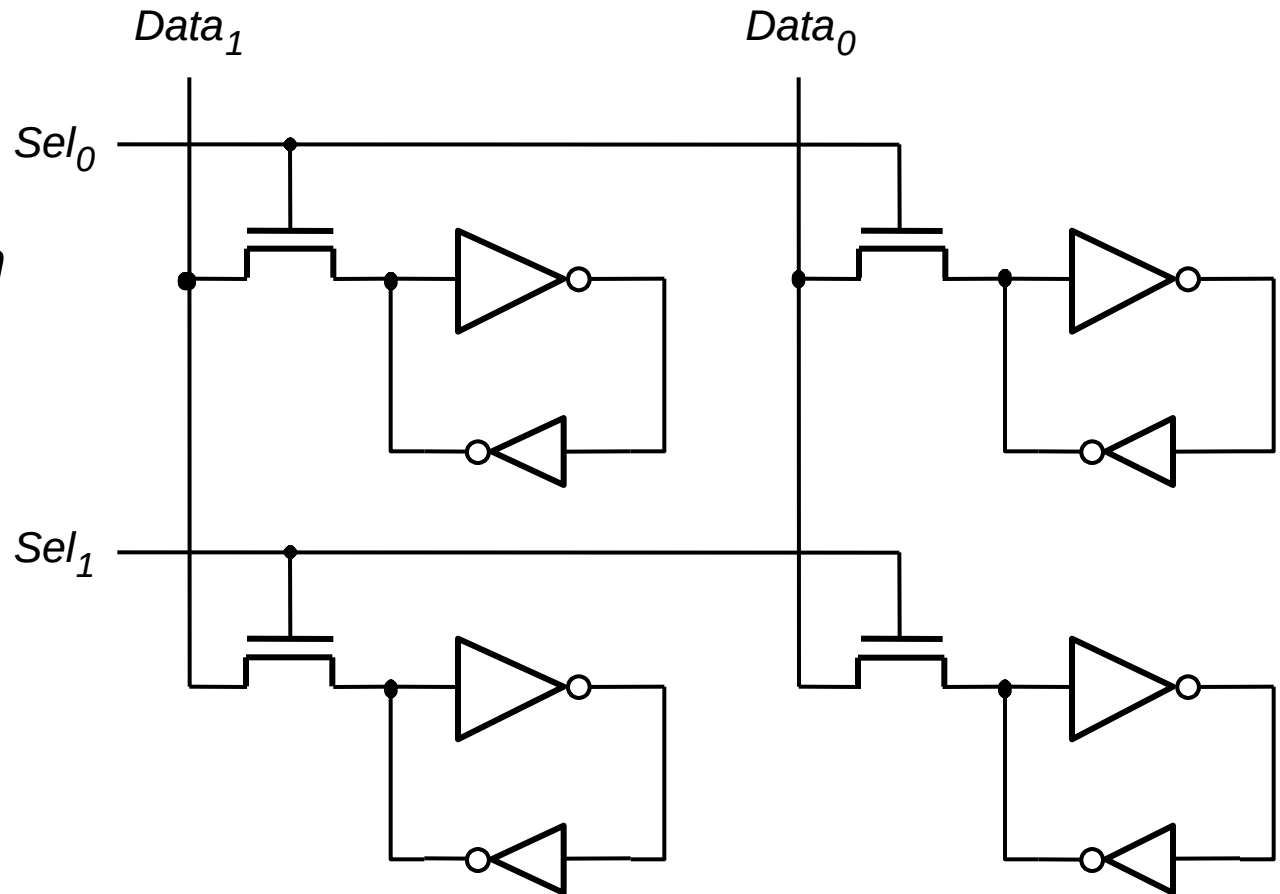
Static Random Access Memory

- Ett SRAM-minne innehåller en matris av SRAM-celler
- För att skriva används 'Data' som ingång!
 - 'Sel' sätts till 1 och det värde som läggs ut på 'Data' sparas i cellen
- För att läsa används 'Data' som utgång!
 - 'Sel' sätts till 1, och värdet i cellen hamnar på utgången



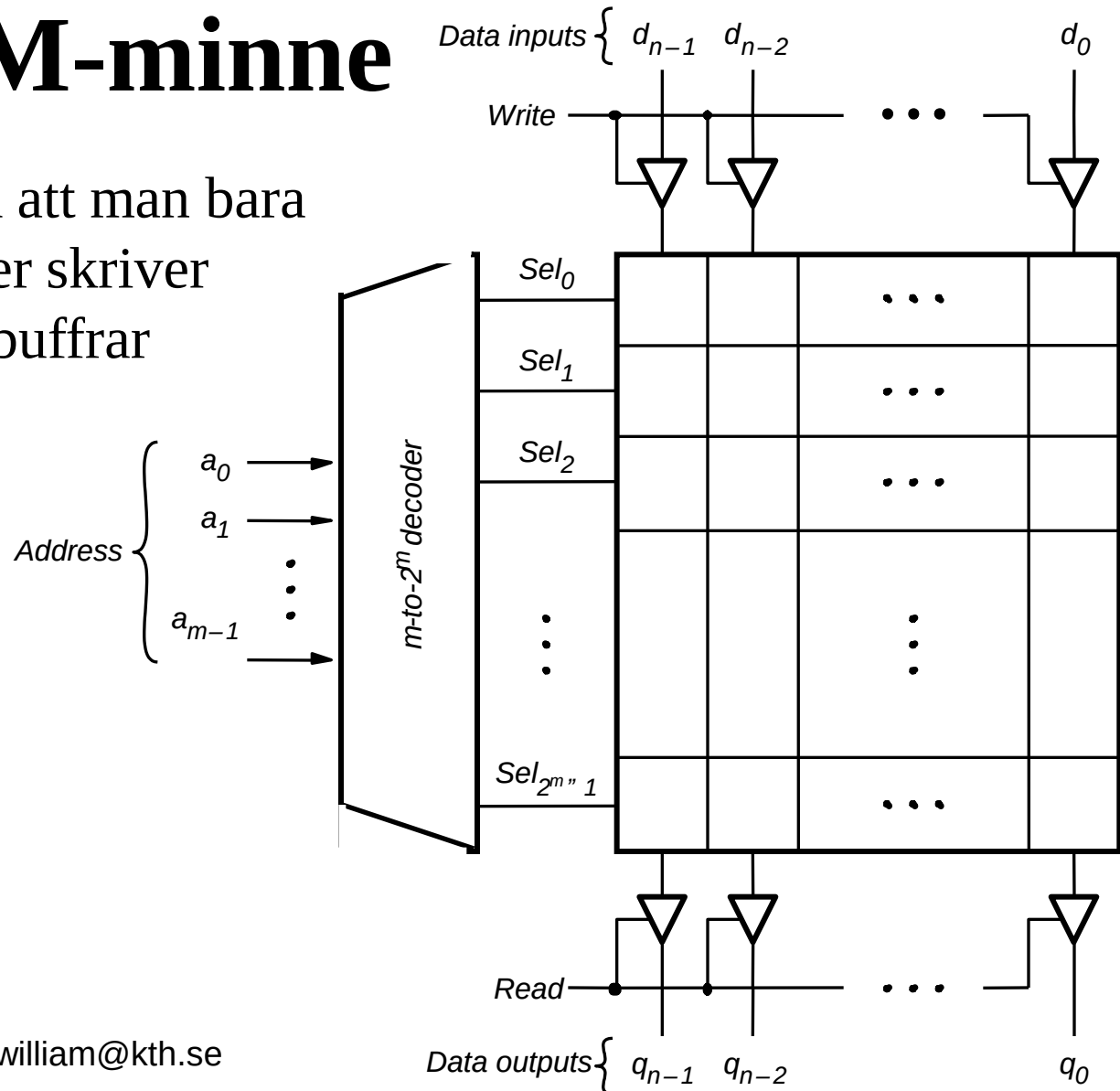
SRAM

*Man bildar nu en
matris av $2^m \times n$
SRAM-celler*



SRAM-minne

För att säkerställa att man bara antingen läser eller skriver används Tristate-buffrar

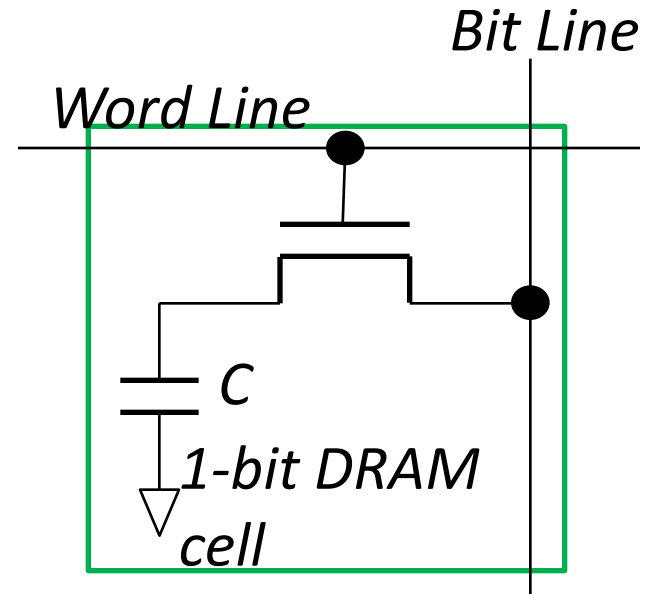


DRAM Dynamiskt RAM

- **SRAM** minnescellen behöver 4 transistorer och det blir för kostsamt att implementera ett stort minne
- **DRAM** minnescellerna använder bara en transistor och en kondensator

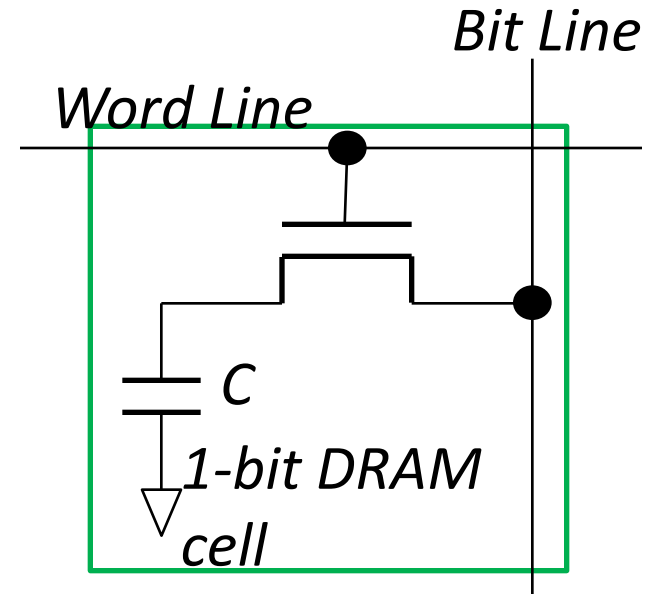
DRAM Minnescell

- DRAM-cellen består bara av en transistor och en kondensator
- Skrivning
 - För att ladda cellen ska ordlinjen sättas till '1'
 - Cellen får nu bitlinjens värde



DRAM Minnescell

- Läsning är mer komplex
 - Man vill inte tappa informationen vid läsning!
 - Bit-linjen sätts på en spänning mellan High and Low
 - För att läsa cellen ska ordlinjen sättas till '1'
 - Bitlinjens justerar nu sin spänning antingen uppåt eller nedåt
 - En extra krets (sense amplifier) detekterar ändringens riktning och skapa en riktig 0:a eller 1:a
 - Även laddningen i kondensatorn C måste återställas!



DRAM Minne



Minnesmodul med 8st kapslar



*Kapsel
256Mbit (32M×8)*

SRAM vs DRAM

- SRAM tar mer plats än DRAM men kräver en enklare accesslogik och är därför snabbare (men också dyrare)
- DRAM används för RAM-minnen i våra vanliga datorer
- När man tar bort strömmen försvinner innehållet av SRAM eller DRAM-minnet!

Minnestyper

- Flyktiga minnen
 - Minnen tappar sin information om man kopplar bort strömförsörjningen
 - static RAM (SRAM)
 - dynamic RAM (DRAM)
- Icke-flyktiga minnen
 - Minnen behåller sin information om man kopplar bort strömförsörjningen
 - **Flash (blockvis skrivning)**
 - **EPROM, EEPROM (bytevis skrivning)**

Det behövs en kombination av olika minnen i en elektroteknisk konstruktion!

Flash-minne

- icke-flyktigt minne
- låg kostnad och låg effektförbrukning
- kan suddas och uppdateras, men det tar mycket mer tid än i ett RAM-minne

EPROM Erasable Programmable ROM



Programmerbart ROM-
minne (kan programmeras
med en kretsprogrammerare)
Erasable – kan raderas med
hjälp av UV-ljus för att
därefter programmeras om.
Därav ”fönstret” på chippets
ovansida.

I moderna elektronikutrustningar slipper Du träffa på EPROM.

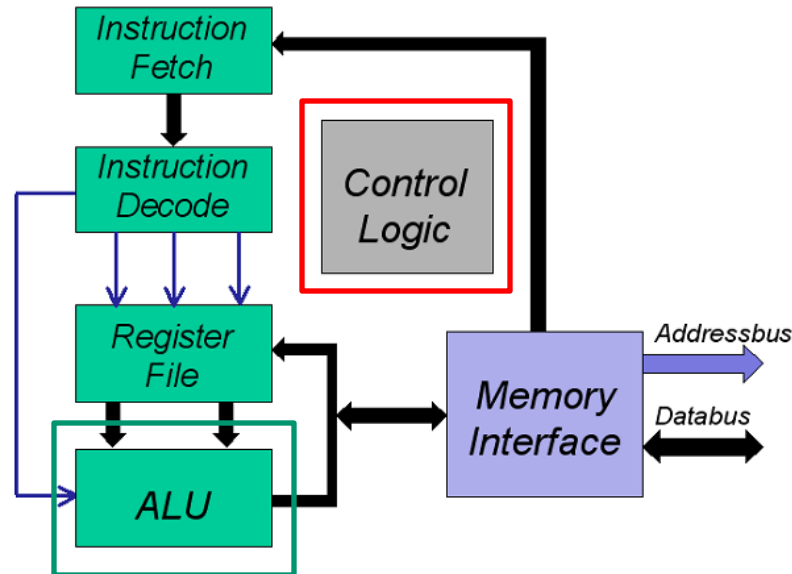
Minnesteknologier

Teknologi	Accesstid	Kostnad \$/GB
SRAM	1 ns	1000
DRAM	50 ns	100
HDD	10 ms	1

Snabba minnen är dyra och billiga minnen är slöa!

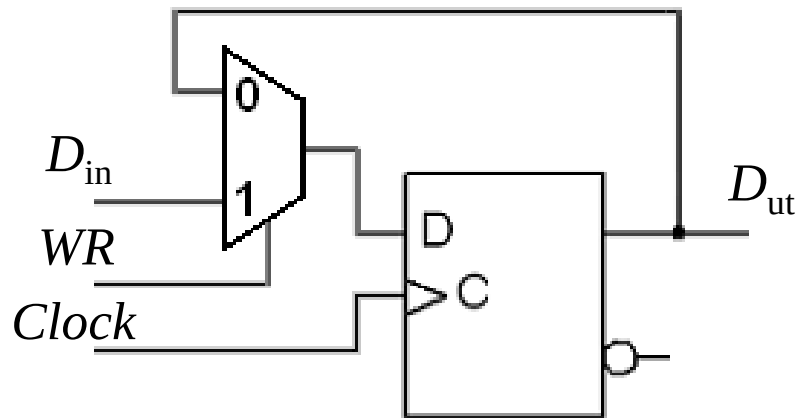
Principiella ”mellan tummen och pekfingret” siffror.

Logik i mikroprocessorn

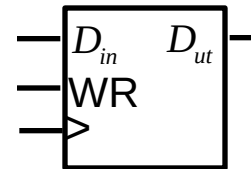


Det finns både kombinatorisk och sekventiell logik i en processor. **Kontrolllogiken** är en tillståndsmaskin medan **ALU:n** är mest kombinatorik.

Registerelement



Symbol



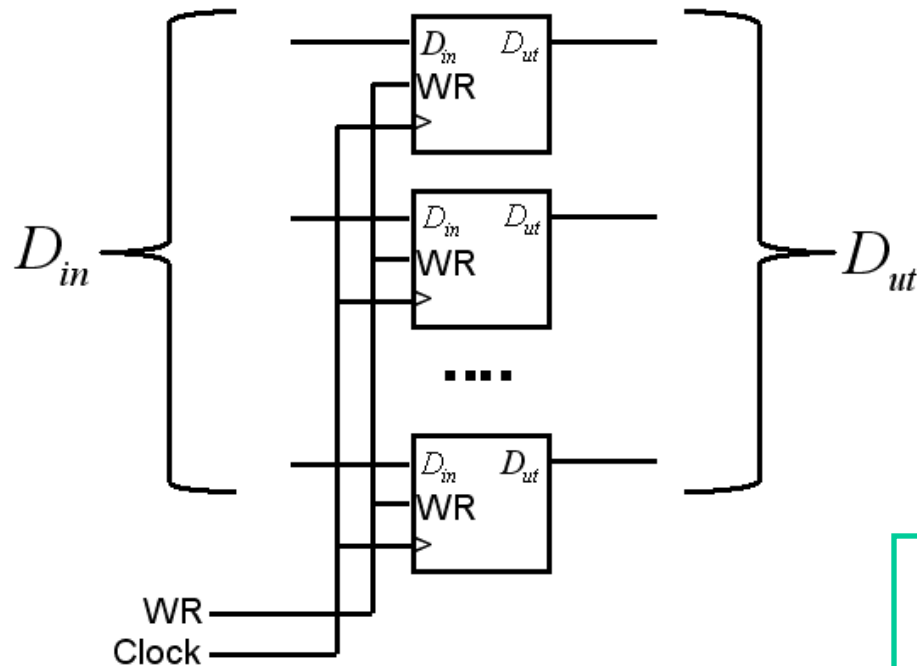
$WR / \overline{hold}$

$WR = 1$ synkron skrivning

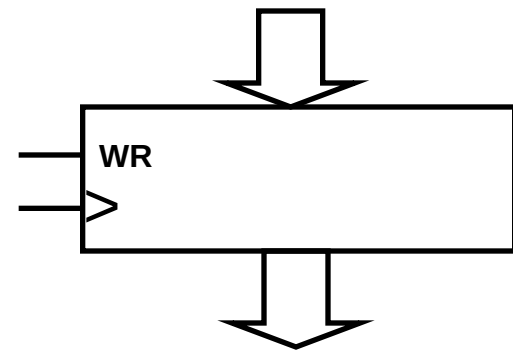
$WR = 0$ hold

*1 logikelement i
en FPGA*

Register

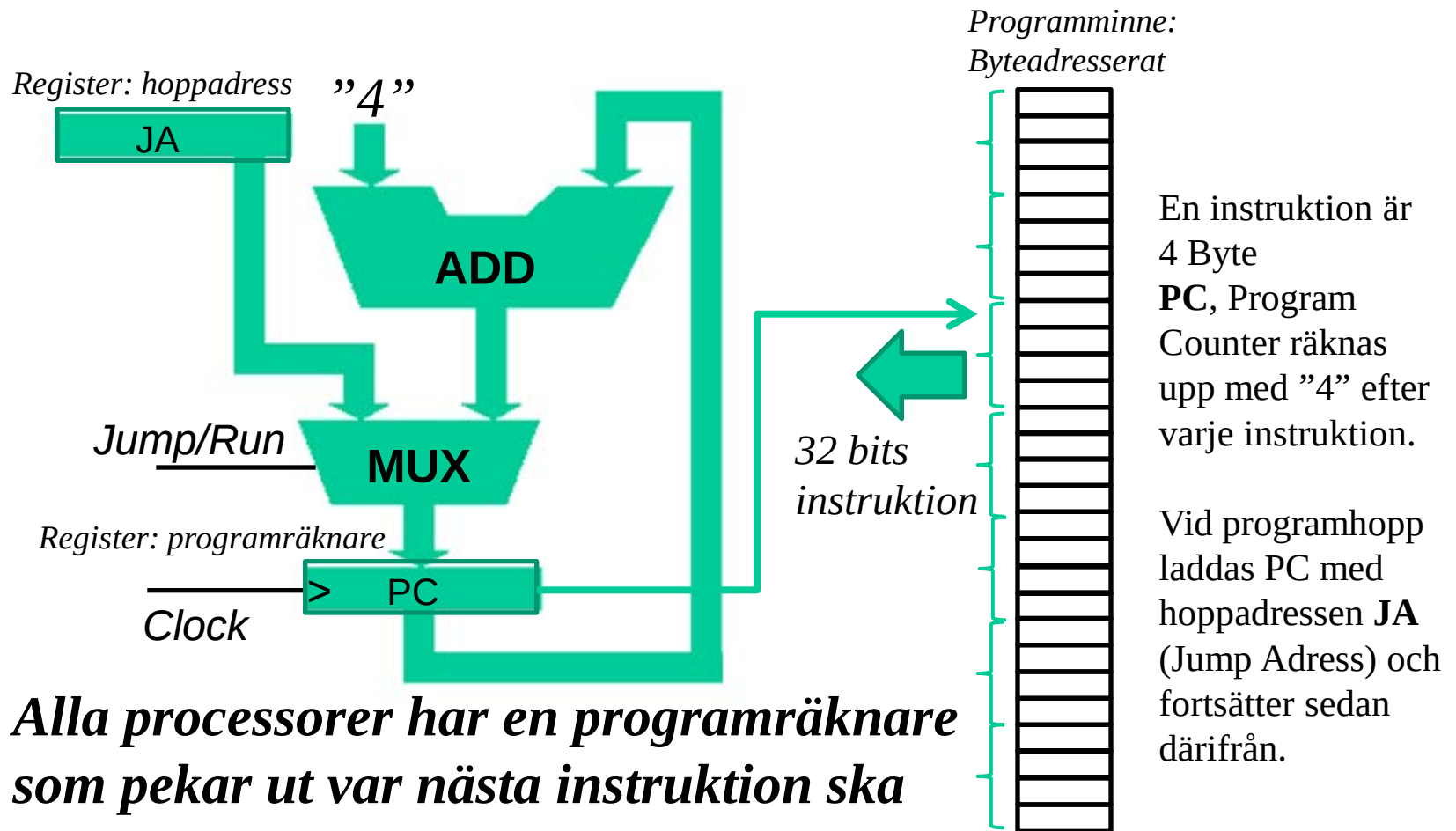


Symbol



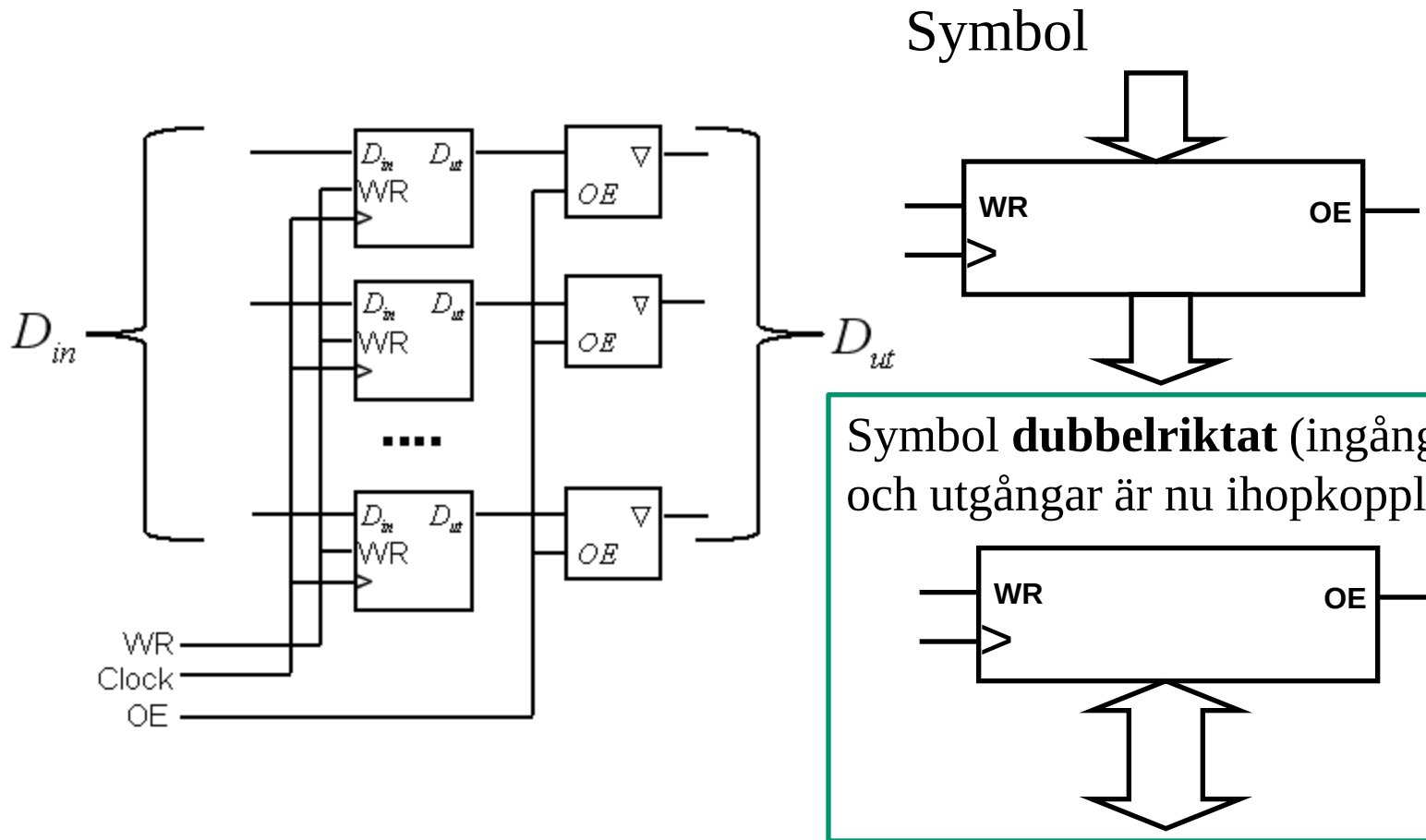
*32 bitars register är 32
logikelement i en FPGA*

Programräknar-register

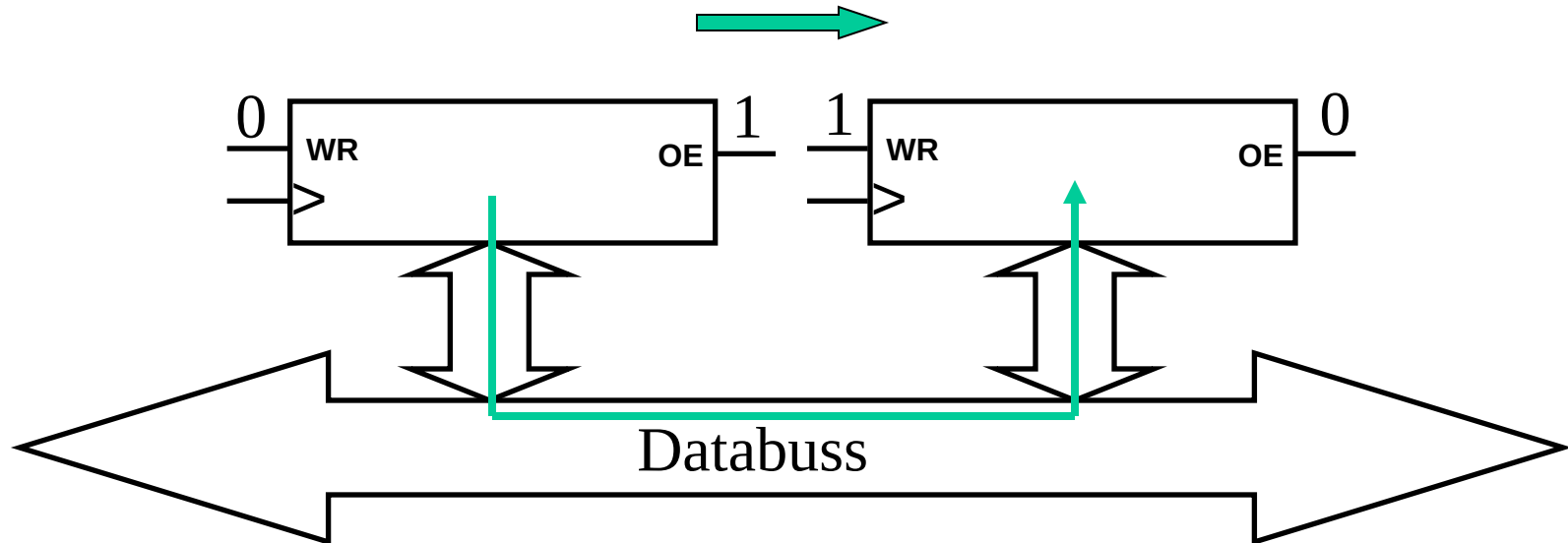


Alla processorer har en programräknare som pekar ut var nästa instruktion ska hämtas i minnet.

Register med threestateutgång

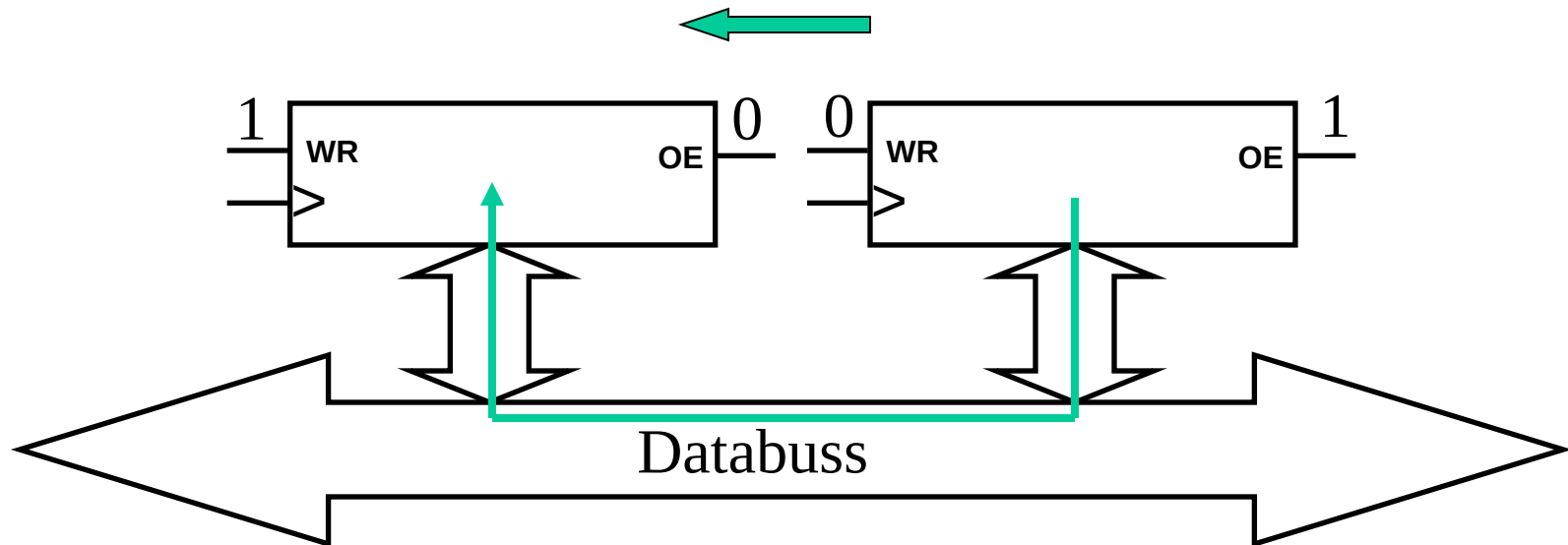


Register och Databuss



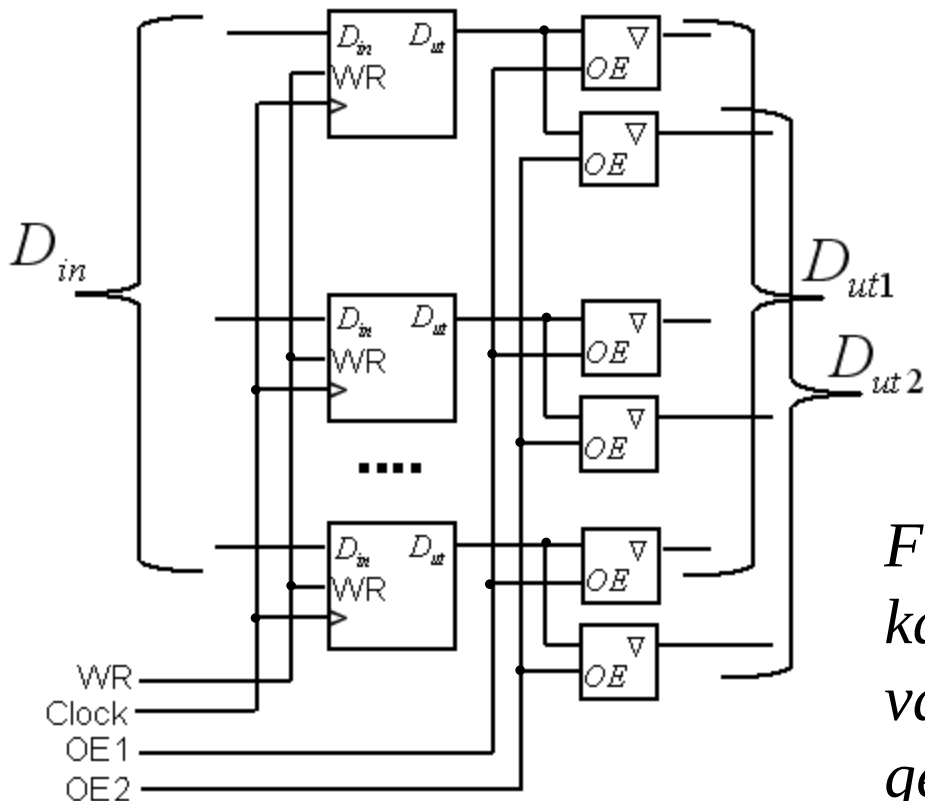
Flera dubbelriktade register med threestateutgångar kan kopplas ihop med varandra för att bilda en gemensam databuss.

Register och Databuss

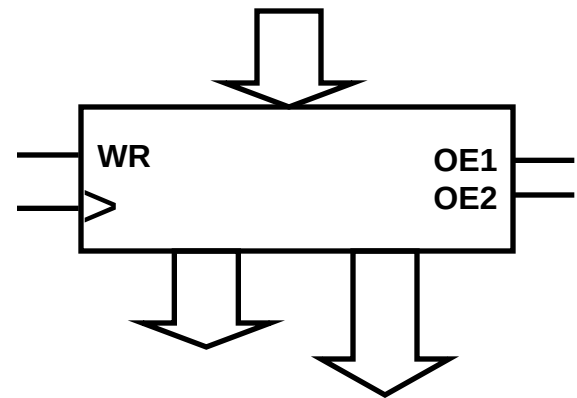


Data kan nu styras att kopieras mellan ***alla*** register på databussen.

Dubbelport register

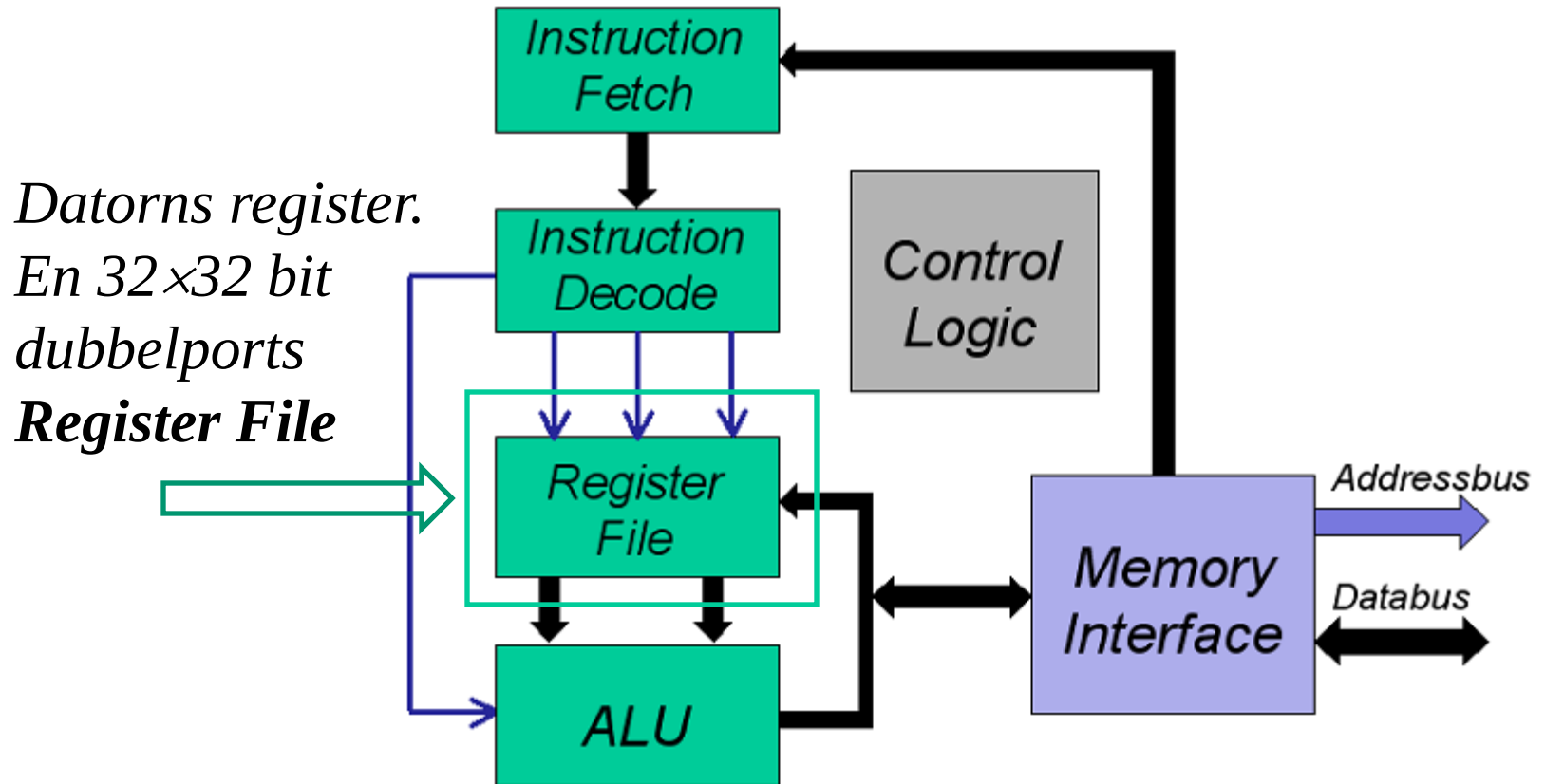


Symbol



*Flera **Dubbelportregister**
kan kopplas ihop med
varandra till **två**
gemensamma utgångsbussar*

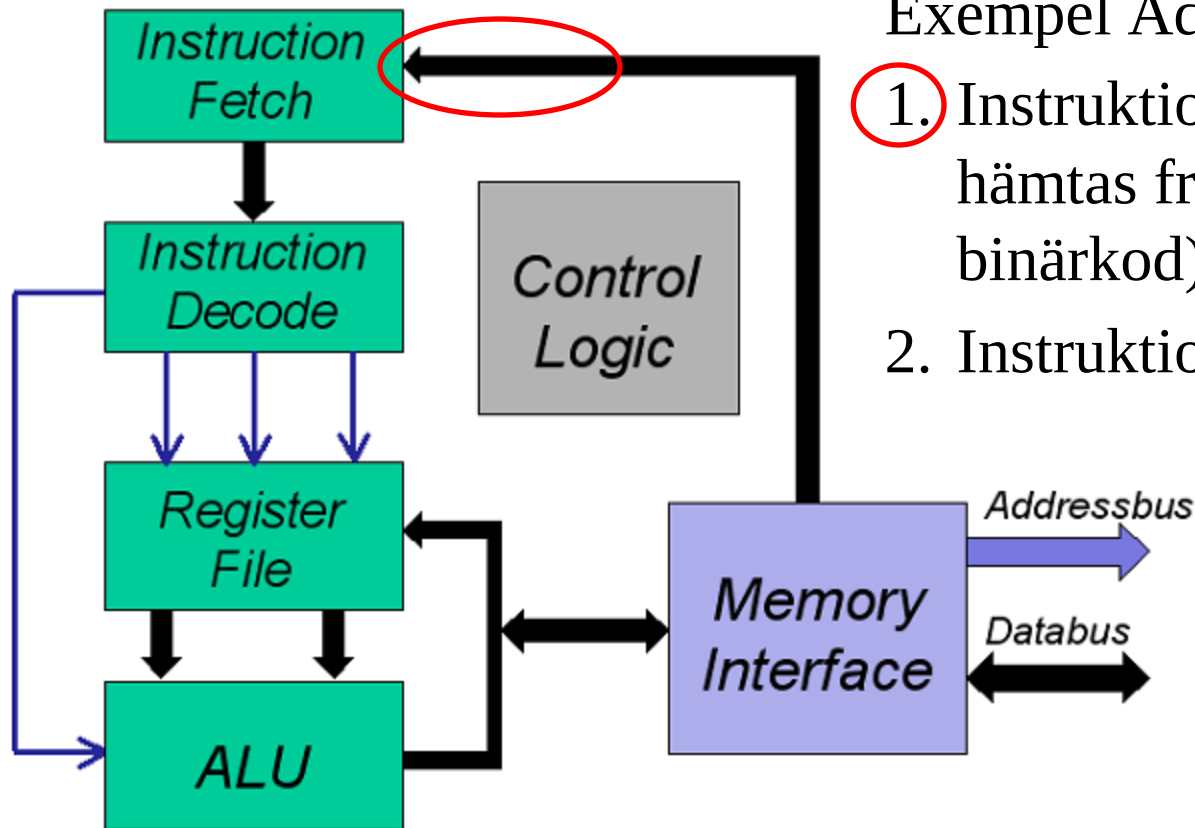
Mikrodatorn - arkitektur



Mikrodatoren - **add**

Exempel Add instruktion

1. Instruktion **add R1, R2, R3** hämtas från minnet (som binärkod)
2. Instruktionen avkodas

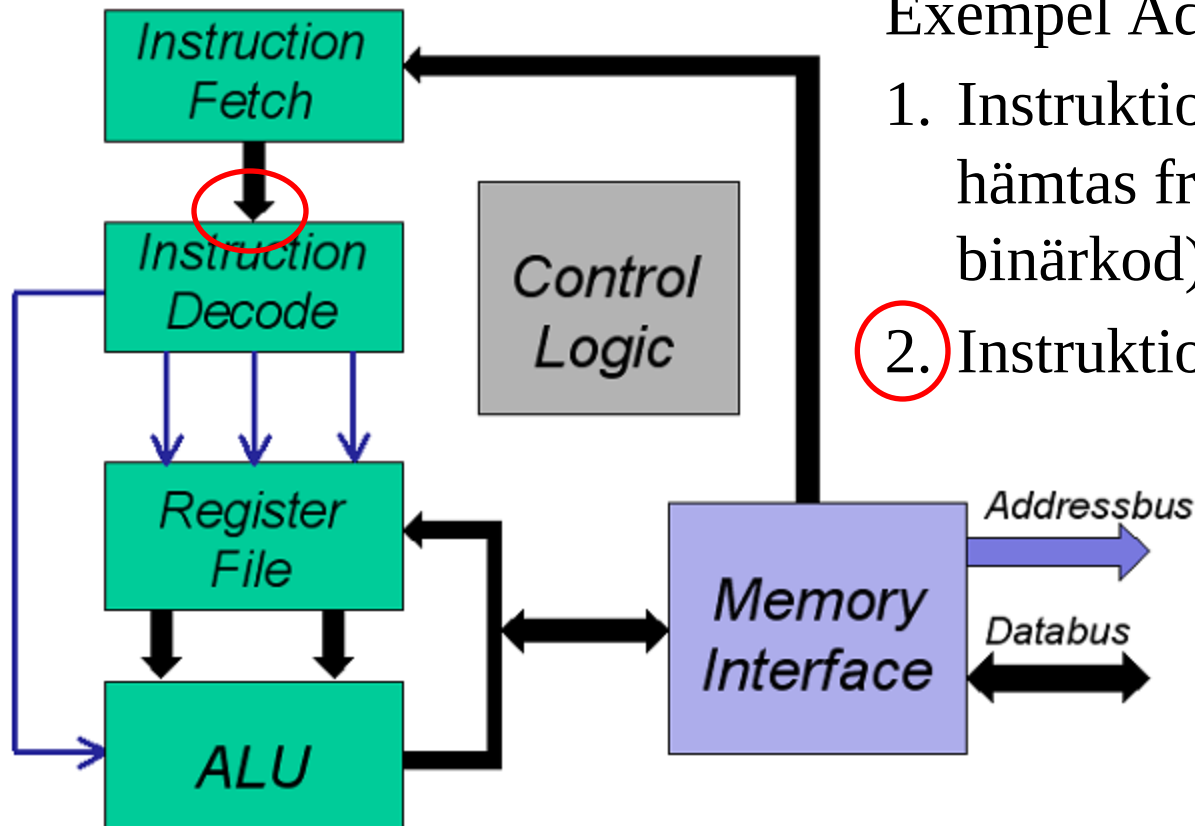


Mikrodatoren - **add**

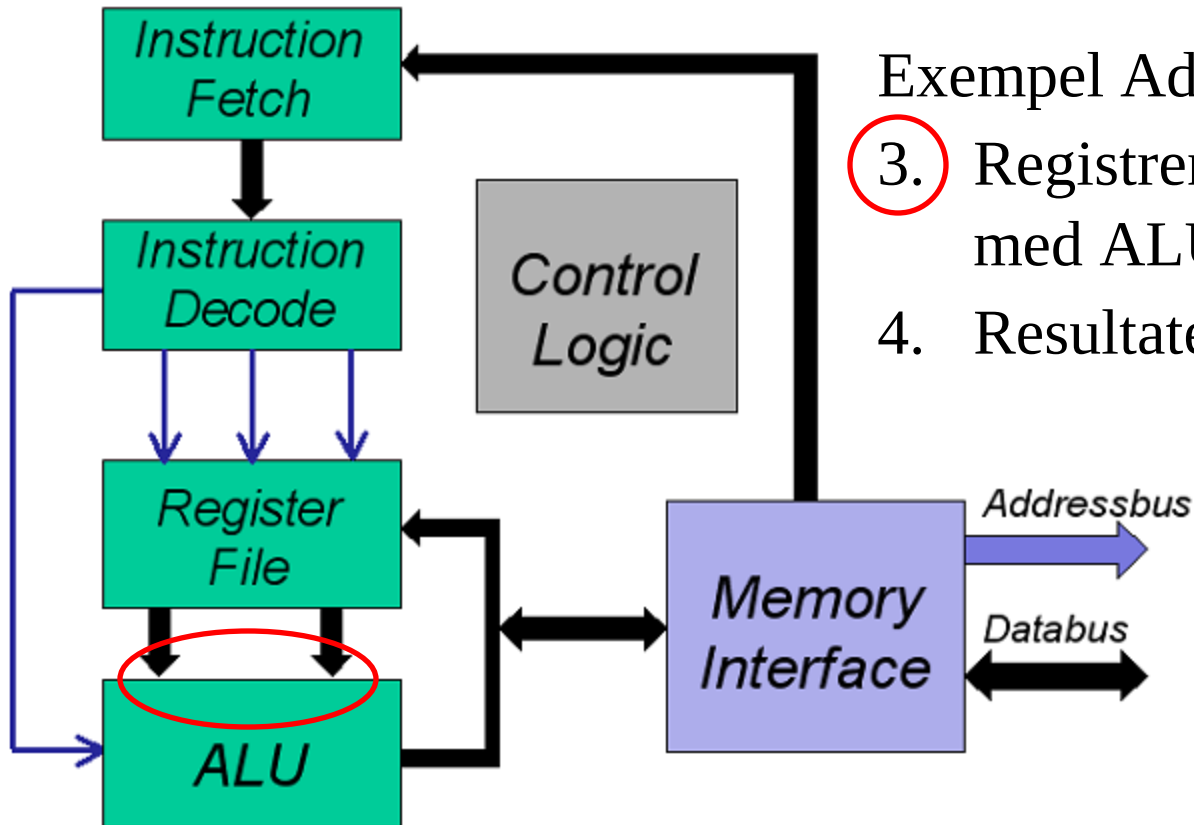
Exempel Add instruktion

1. Instruktion **add R1, R2, R3**
hämtas från minnet (som
binärkod)

2. Instruktionen avkodas



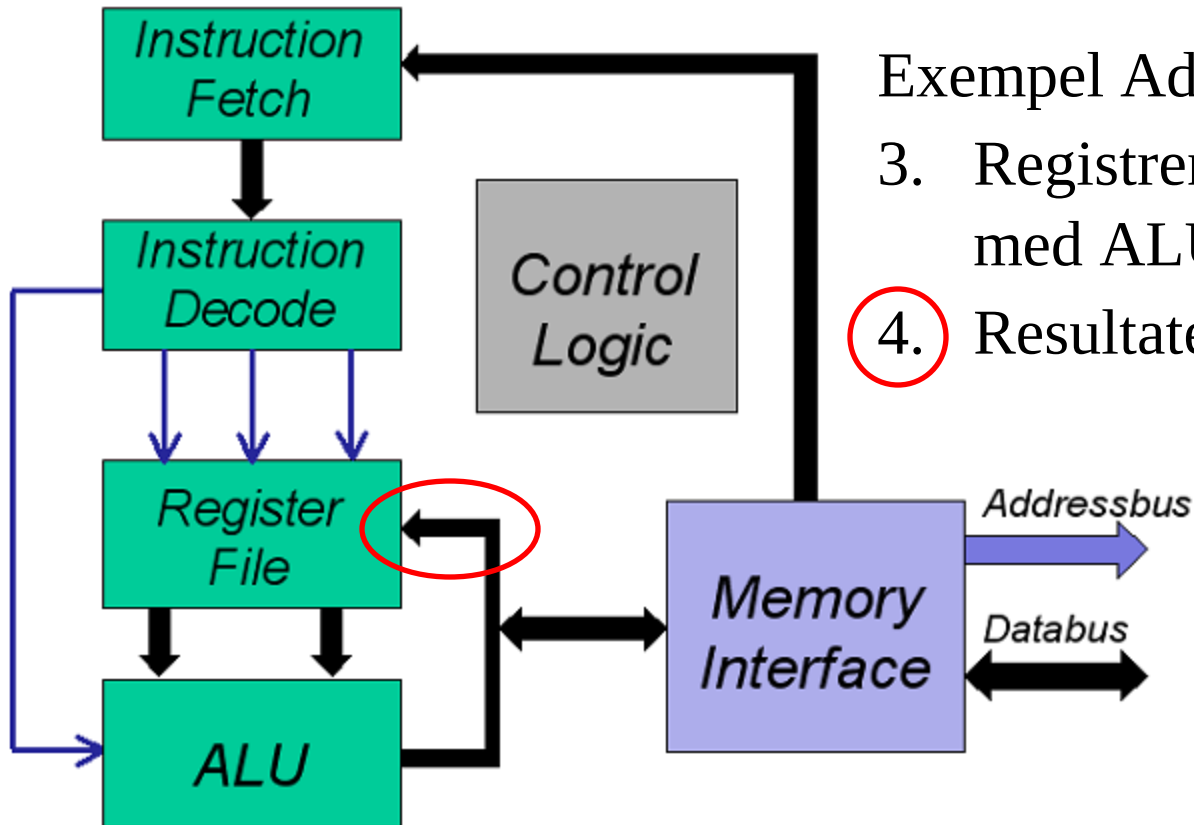
Mikrodatorn - **add**



Exempel Add instruktion

3. Registren **R2** och **R3** adderas med ALU:n
4. Resultatet skrivs till register **R1**

Mikrodatorn - **add**



Exempel Add instruktion

3. Registren **R2** och **R3** adderas med ALU:n

4. Resultatet skrivs till register **R1**

Det krävs således flera klockpulsperioder för att genomföra en instruktion.

(man kan kanske ordna med en Pipeline?)

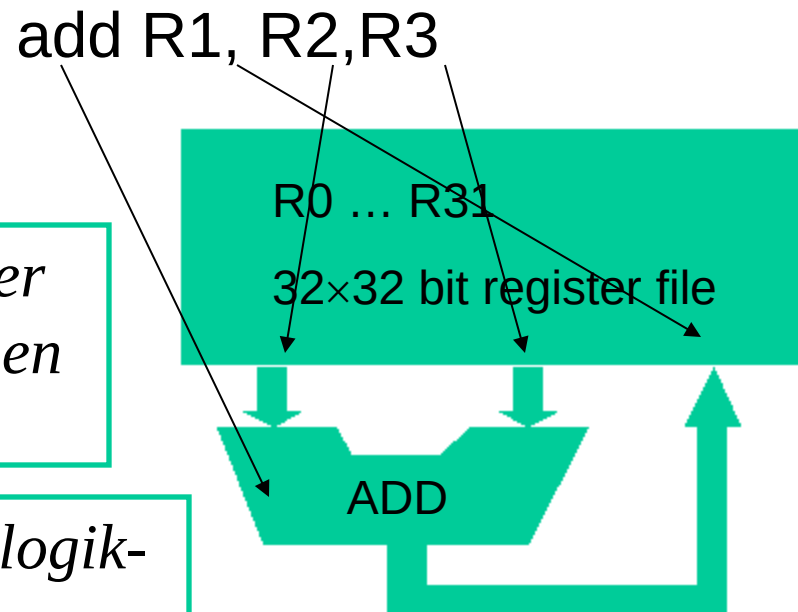
Register file

Processorn har en 32×32 bitars register file (med dubbelports-register). Man kan därför samtidigt *läsa* från *två* valfria register eller skriva till *ett* valfritt register per klockpuls.

Datorinstruktionen add,
innebär att summan
 $R2+R3$ läggs i R1

*En register file med 32 register
är $32^2 = 1024$ logikelement i en
FPGA*

*En 32 bitars adderare är 32 logik-
element i en FPGA*



Möjligt instruktionsformat

