LUNDS UNIVERSITET

Datorteknik

ERIK LARSSON

Personal

- Kursansvar: Erik Larsson: Erik.Larsson@eit.lth.se
- Lärare:

  Gautham Rangasamy, Heera Menon, Siyu Tan, Navya Sri Garigapati, Steffen Malkowsky, Qiuyan Liang, Zhongyunshen Zhu, and Sidra Muneer
- Forskningsingenjör: Christoffer Cederberg
- Kurssekreterare: Erik Göthe: erik.gothe@eit.lth.se

LUNDS UNIVERSITET

Nyheter

- Förra året ny laborationsplattform
- Förra året nya laborationsuppgifter
- Nytt lektionsmaterial

LUNDS UNIVERSITET

Kursbeskrivning

Syfte

- Ge en introduktion till hur ett datorsystem fungerar på maskinspråksnivå och hårdvarunivå.

Mål

- Förstå ett datorsystems funktion och dess ingående delar
- Förstå samspelet mellan hårdvara, maskinspråk och högnivåspråk
- Förstå grunder för operativsystem och kommunikation med omvärlden

LUNDS UNIVERSITET

Kursbeskrivning

Mål (forts)

- Ha viss färdighet och erfarenhet i programmering på maskinspråksnivå
- Kunna göra och analysera enkla designer av system där datorsystem används som systemkomponenter
- Kunna analysera och utvärdera olika lösningar för programmering på maskinspråksnivå
- Visa prov på insikt om möjligheter och begränsningar med datorteknik

LUNDS UNIVERSITET

Kursprogram

- Kursen består av föreläsningar, lektioner och laborationer.
- Kursen svarar mot 6 hp, vilket motsvarar 160 timmar arbete.

Kursmoment

- *Tentamen* - 3 hp, dvs 80 timmar arbete
- *Laborationer* - 3hp, vilket motsvarar 80 timmar av arbete

LUNDS UNIVERSITET

Kursmoment Tentamen

Betyg

- För att få betyg 3 ska duggor (elektroniska frågor) vara godkända. OBS. För att få betyg 3 behöver du inte skriva tentamen
- För att få betyg 4 och 5, gäller att duggor (elektroniska frågor) måste vara godkända och ett högre betyg kan fås genom att göra tentamen

Övergångsregler

- Se kurshemsidan

LUNDS UNIVERSITET

Duggor

Godkända duggor ger betyg 3

- Tid för duggor:
  - Dugga 1: 3:e februari – 9:e februari
  - Dugga 2: 17:e februari – 23:e februari
  - Dugga 3: 2:e mars – 8:e mars
- Vid varje försök: 1 timme
- Nytt försök kan påbörjas tidigast 16 timmar efter senaste försök avslutats
- För godkänt: 80% rätt
- Duggorna finns på:
  - http://elearning.eit.lth.se/moodle/

LUNDS UNIVERSITET

Duggor

Dugga 1

- Denna dugga är möjlig att göra funder en begränsad tid och för varje försök har du en begränsad tid på dig.
- Du kommer få 10 slumpmässigt valda frågor och för att bli godkänd krävs att du besvarat 8 frågor rätt.
- Du kan påbörja nästa försök på duggan 16 timmar efter att du avslutat förra försök och detta gäller så länge duggan är öppen.
- Läsanvisningar:

1. Föreläsningsmaterial FÖ1-FÖ3,
2. Kapitel 1 förutom 1.9 och 1.13,
3. Kapitel 2 förutom 2.15-2.18 och 2.22,
4. Appendix A: A.1-A.6 och
5. Appendix B förutom B.4, B.6, B.12, och B.14

The quiz will not be available until Monday, 3 February 2020, 12:00 AM

This quiz will close at Sunday, 9 February 2020, 11:59 PM

Time limit: 1 hour

Grading method: Highest grade

LUNDS UNIVERSITET

Kursmoment: Tentamen

- Examination:
  - Skriftlig tentamen.
  - Första tentamen: 2020-03-19 klockan 08:00-13:00
- Maximalt 50 poäng
- Betygsgränser:
  - Betyg 3: Minst 20 poäng
  - Betyg 4: Minst 30 poäng
  - Betyg 5: Minst 40 poäng

Motivera och förklara tydligt för tankegång är viktig

LUNDS UNIVERSITET

Kursmoment: Laborationer

- Moment svarar mot 3 hp, dvs 80 timmar arbete
- Schemalagd tid: 4*2 (lektioner)+4*4(laborationer)=20 timmar
- Material: Finns på kurshemsida
- 4 laborationsuppgifter
- Godkänt: Demonstration

Kan jag bara arbeta med laborationerna under laborationstid?
Svar: Utrustning kommer stå framme så du kan laborera när som helst

Kan jag gå på laborationstillfällen dit jag inte är anmäld?
Svar: Nej

LUNDS UNIVERSITET

Kursprogram

Slutbetyg

- För att få slutbetyg 3: laboration och duggor (elektroniska frågor) godkända. OBS. För att få betyg 3 behöver du inte skriva tentamen
- För att få slutbetyg 4 och 5: laboration och duggor (elektroniska frågor) godkända och ett högre betyg fåtts på tentamen

Övergångsregler

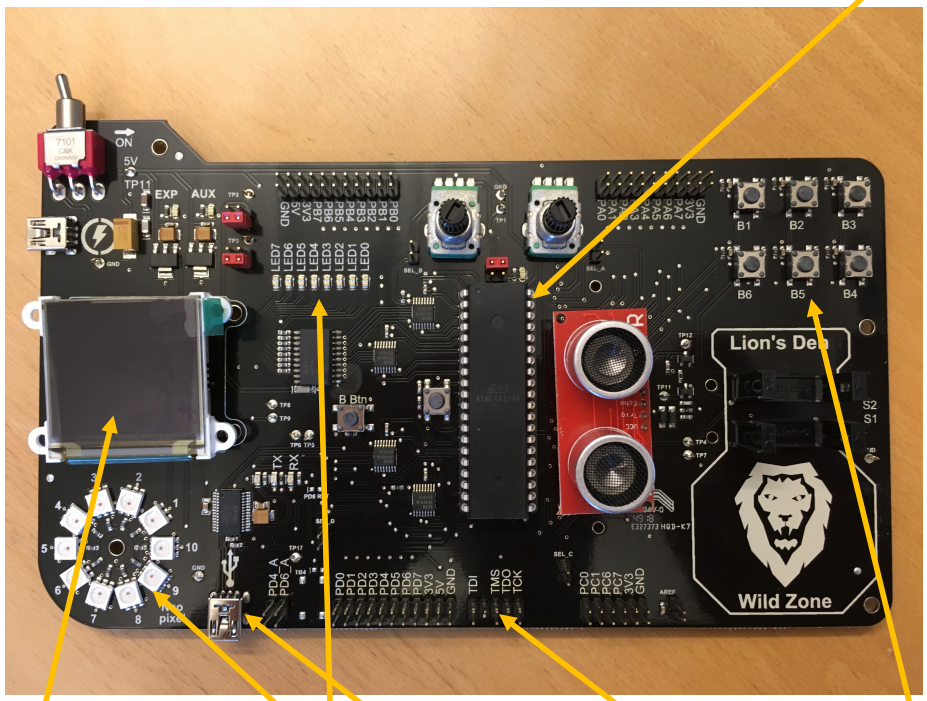
- Se kurshemsidan

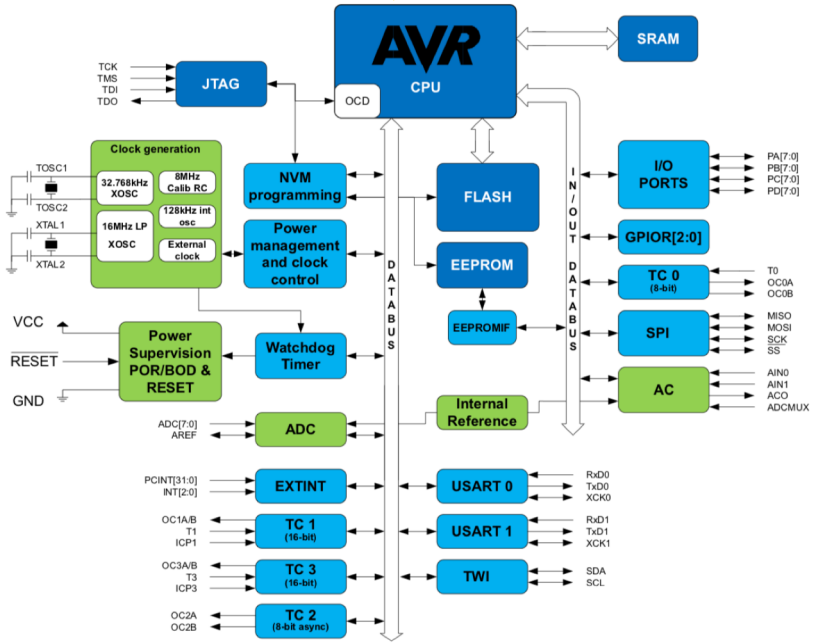
Upplägg

- **Föreläsningar:**
  - Syfte: Ge överblick. Underlätta läsning av teori.
  - Förberedelser: Inga.
- **Lektioner/övningstillfällen:**
  - Syfte: Klargöra laborationer. Koppla del av teori till praktik
  - Förberedelser: Se kurshemsidan
- **Laborationer:**
  - Syfte: Ge praktisk kunskap och förståelse
  - Förberedelser: Se kurshemsidan

LUNDS UNIVERSITET

Laborationer





ATmega1284

Display

Lysdioder

USB-UART

JTAG

Tryckknappar

LUNDS UNIVERSITET

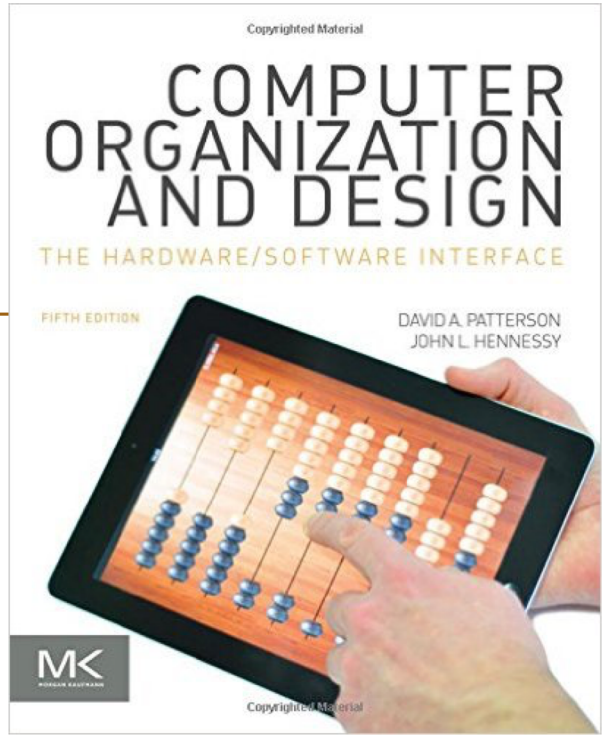
Laborationer: Regler

- Laborationsuppgifter ska lösas självständigt av varje laborationsgrupp
- Det är inte tillåtet att **ge** laborationsresultat till en annan grupp
- Det är inte tillåtet att **ta, kopiera eller på något annat sätt efterlikna** en annan grupps resultat
- Alla gruppmedlemmar måste ta aktiv del i alla delar av laborationen, det inkluderar att skriva programkod, testa och felsöka, genomföra experiment, och demonstrera (examination). **Examinationen är alltid baserad på individuella resultat.**

LUNDS UNIVERSITET

Litteratur

- Rekommenderad kursbok
  - D. A. Patterson and J. L. Hennessy, Computer Organization and Design – the Hardware/Software Interface, Morgan Kaufmann.
- Läsanvisningar
  - Se kurshemsidan
- Föreläsningsanteckningar
  - Se kurshemsidan
- Simulatorer
  - Se kurshemsidan



LUNDS UNIVERSITET

Blooms taxonomi - lärande

- **Kunskap:** Den studerande visar att han/hon kan återge faktakunskaper som han memorerat. Hjälpfrågor: Vad? Vem? Hur skulle du definiera?
- **Förståelse:** Den studerande visar att han/hon förstår relationer och samband i det kunskapsstoff han/hon lärt sig. Hjälpfrågor: Vad är det viktigaste i artikeln? På vilket sätt skiljer sig / liknar teorierna varandra?
- **Tillämpning:** Den studerande kan tillämpa sina kunskaper på nya, för honom relativt okända tillämpningsområden. Hjälpfrågor: Hur kan du tillämpa...?
- **Analys:** Den studerande kan analysera fram beståndsdelar i en helhet och kan hitta det viktiga. Hjälpfrågor: Vilka bevis hittar du som stöder…?
- **Syntes:** Den studerande kan producera och skapa något eget och relativt unikt (förena begrepp på ett nytt sätt). Hjälpfrågor: Vad händer om…? Hur kan vi förbättra / lösa problemet?
- **Värdering:** Den studerande förmår utvärdera idéer, kunskap, metoder och lösningar. Hjälpfrågor: Vad är din åsikt om..?

LUNDS UNIVERSITET

Kursombud

LUNDS UNIVERSITET

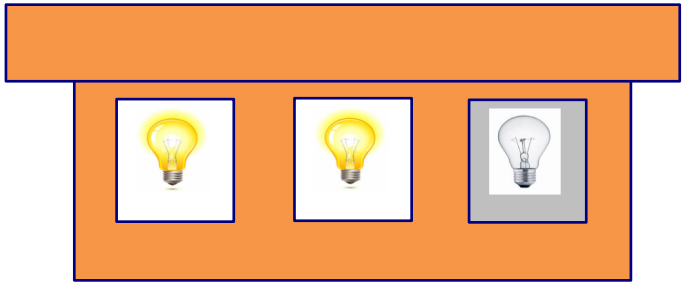
LUNDS UNIVERSITET

Datorteknik

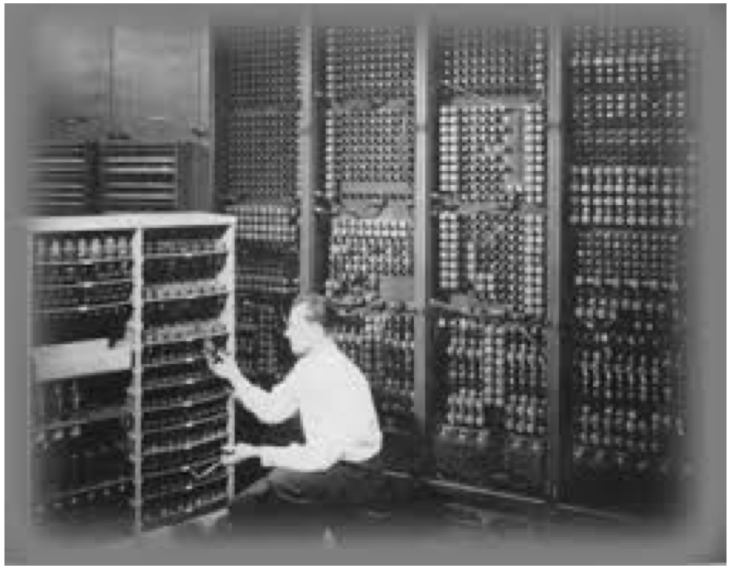
ERIK LARSSON

I begynnelsen

- Tidigt 1900: transistorn uppfinns (Nobelpriset i fysik 1956)
- Används i datorer som ”strömbrytare”

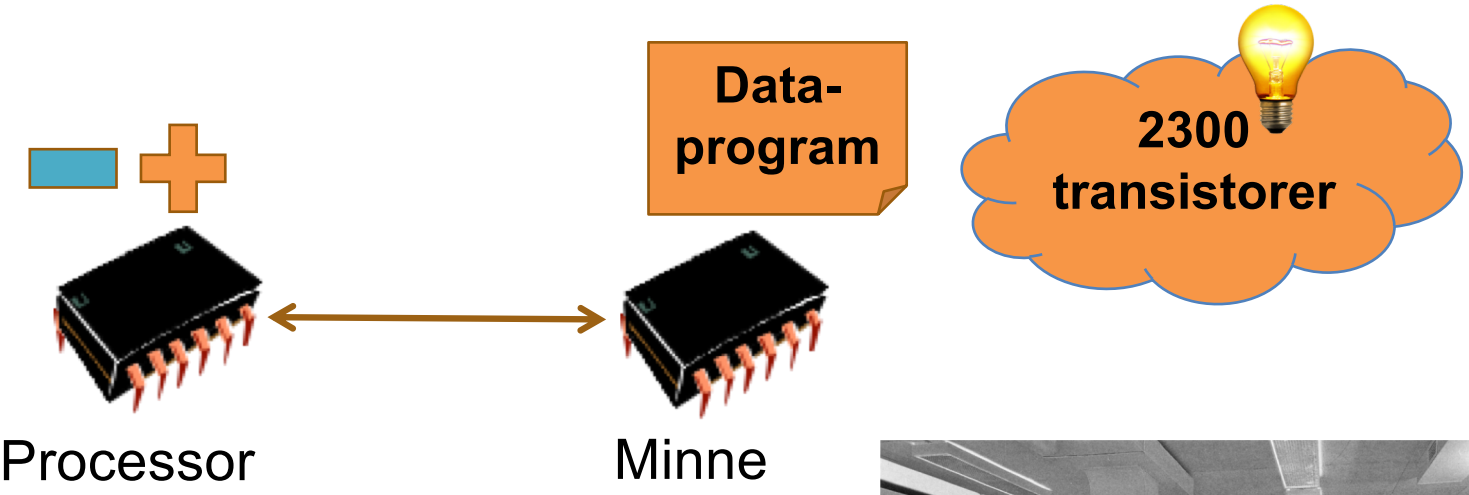


- **<u>Om framtiden:</u>** ”Jag tror att världs-marknaden är omkring fem datorer” Thomas Watson, president of IBM, 1943

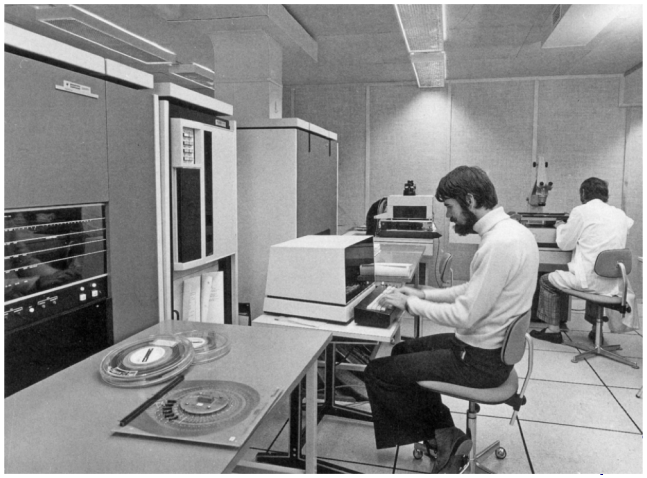


I förrgår

- 1958: första integrerade kretsen (Nobelpriset i fysik 2000)
- 1971: första processorn som integrerad krets (Intel 4004)

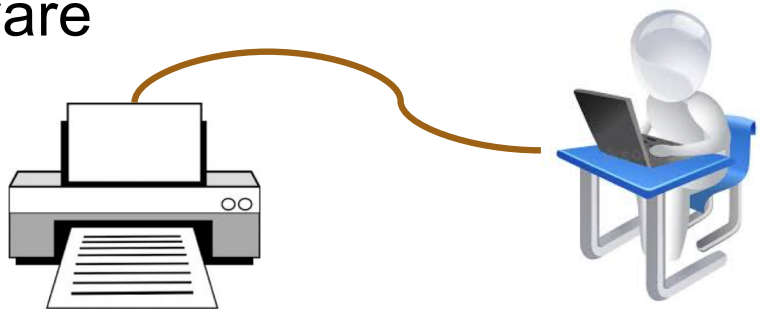


- **Om framtiden:** ”Det finns ingen anledning att någon skulle vilja ha en dator hemma.” Ken Olsen, grundare av Digital Equipment Corporation, 1977



I går

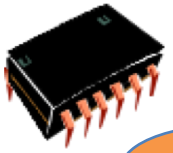
- Var och en satt med en dator, sammankopplad med skrivare



- Program och data lagrades på floppydisketter.
  - En floppydiskett: 1.44 Mbyte ≈ 1.44 sekunder musik
- **Tankar om framtiden (1)**: "Internet är en fluga", Ines Uusmann, 1996 Kommunikationsminister och ansvarig för IT
- **Tankar om framtiden (2):** Iphone är en skittelefon (Ericsson)

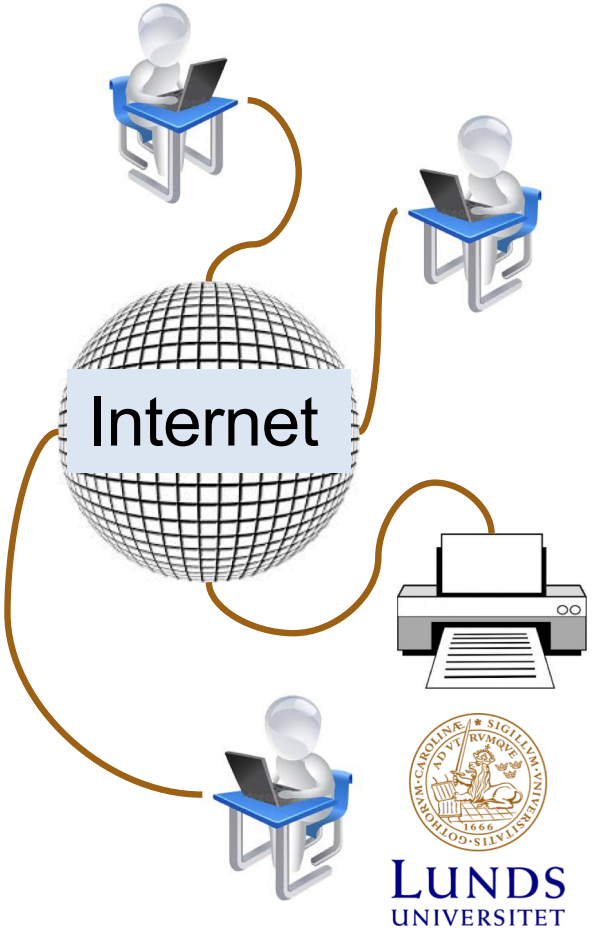
LUNDS UNIVERSITET

I dag



- **7 miljarder transistorer**
- **Ledningar 1000 gånger smalare än hårstrån**

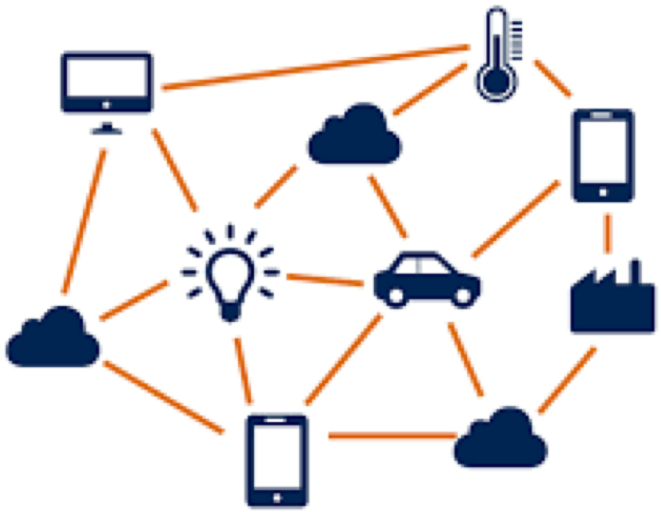
- Sammankopplade datorer
  - Epost, Skype, E-handel
- Nya ”produkter” där ”data/information” är den nya oljan
  - Facebook, Apple, Amazon, Netflix och Google (FAANG)



LUNDS UNIVERSITET

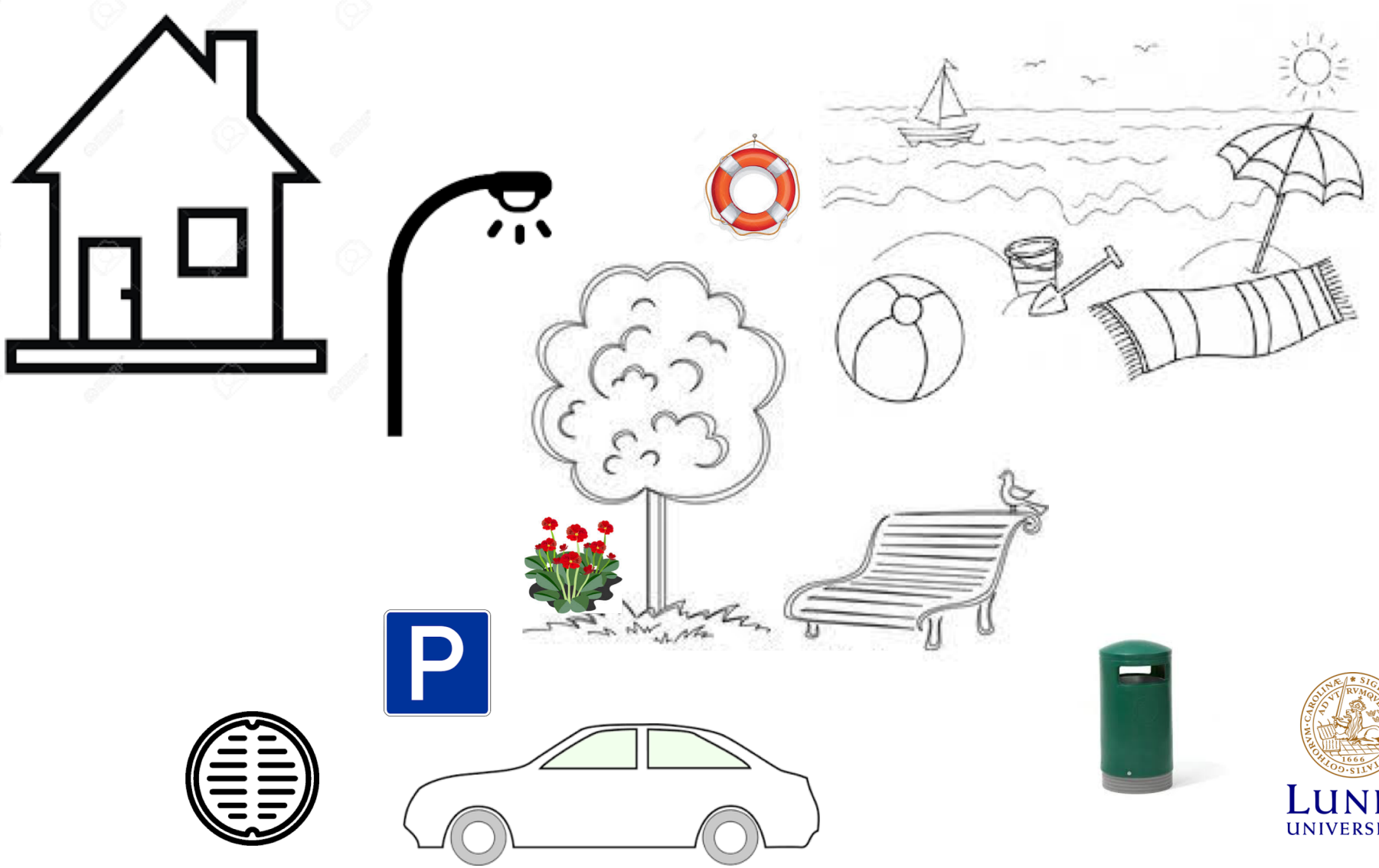
I morgon

- Sammankopplade saker – internet-of-things (IoT)
  - Varenda sak innehåller en dator
  - Varenda sak är sammankopplad
- Hur kommer detta påverka, t ex staden vi lever i?



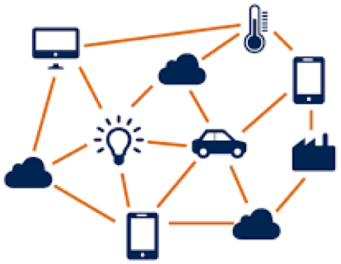
LUNDS UNIVERSITET

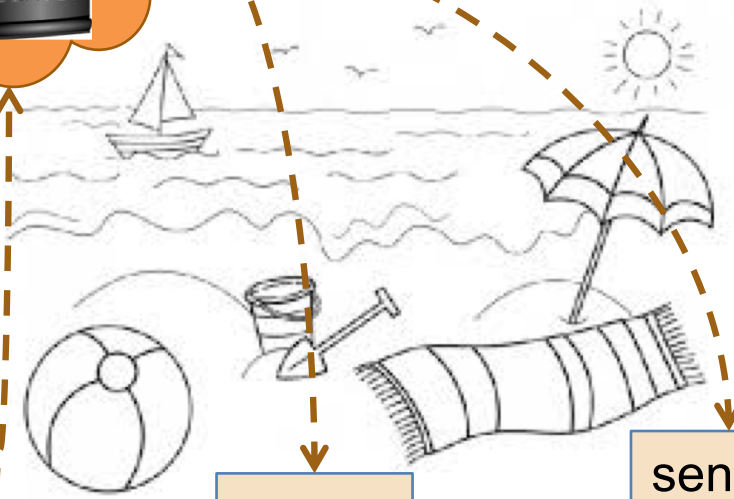
Vår stad idag

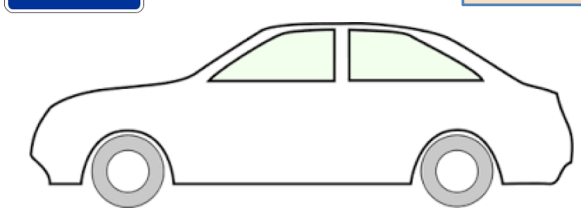


LUNDS UNIVERSITET

Vår stad imorgon







sensor

sensor

sensor

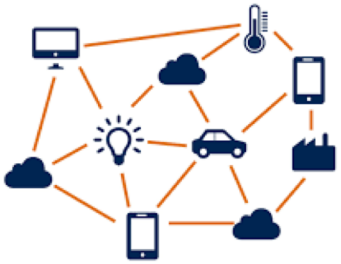
sensor

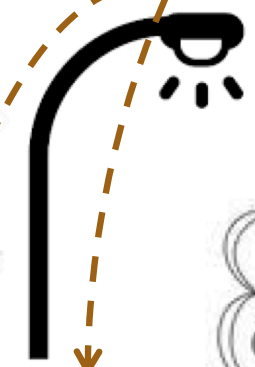
sensor

sensor

LUNDS UNIVERSITET

Vår stad imorgon





sensor

sensor

sensor

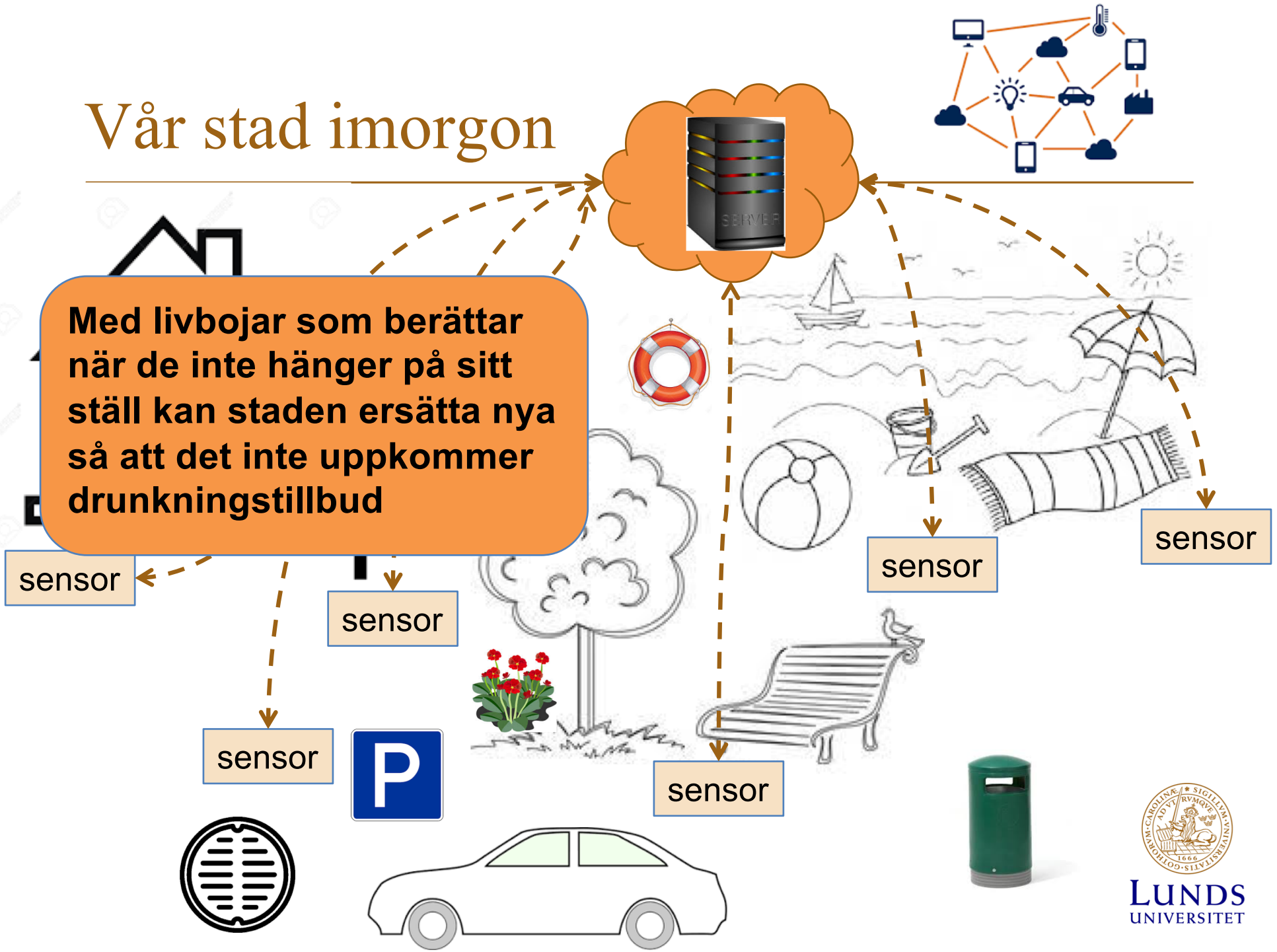
sensor

sensor

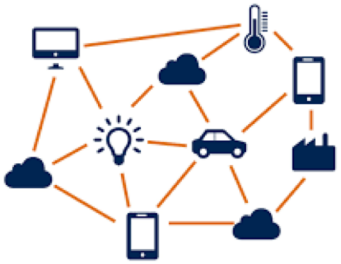
Smarta soptunnor talar om hur mycket skräp de innehåller - kan planera tömningen – bra för miljön.

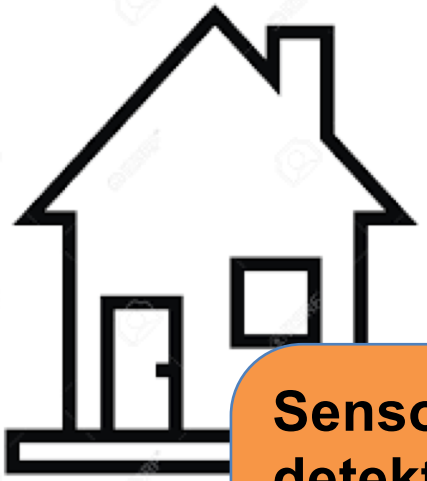
LUNDS UNIVERSITET

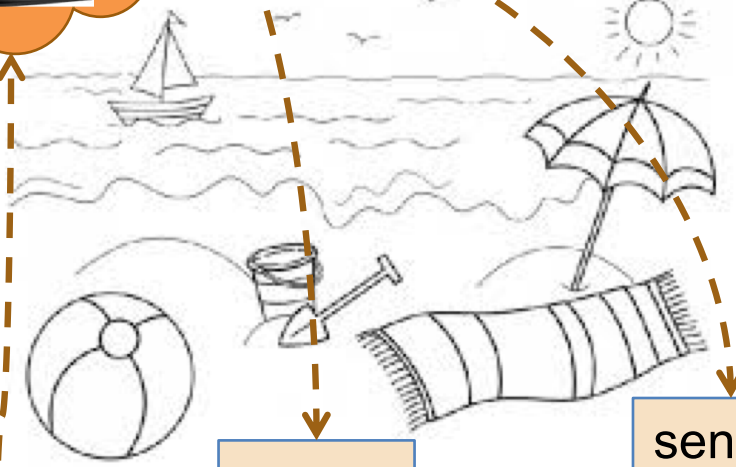
Vår stad imorgon



Vår stad imorgon







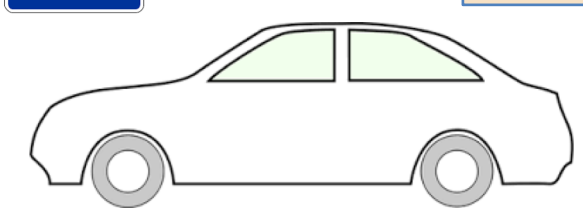
Sensorer som detekterar vattenläckor, för 15% av allt dricksvatten når inte konsument.

sensor

sensor

sensor

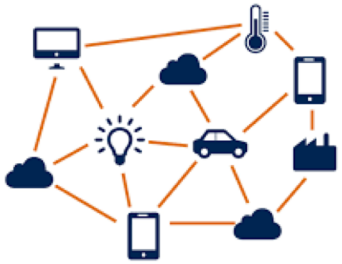
sensor

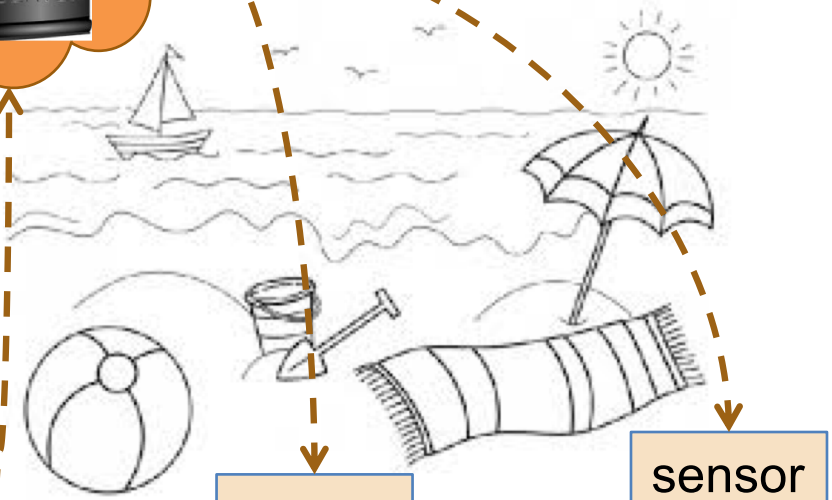


LUNDS UNIVERSITET

Vår stad imorgon

Vår stad imorgon



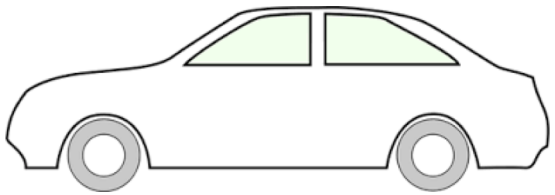


sensor

sensor

sensor

Med smarta brunnar som talar om när de är övertäckta kan de rensas innan regnet kommer så det inte blir översvämning



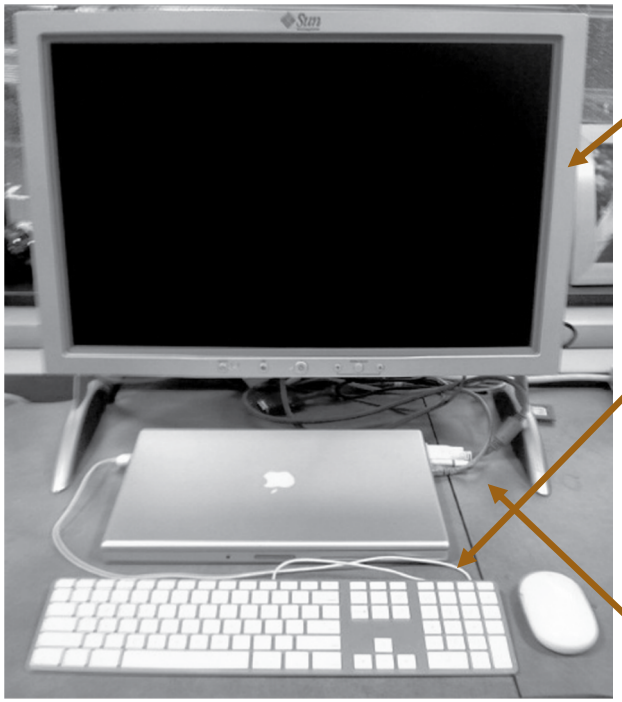
LUNDS UNIVERSITET

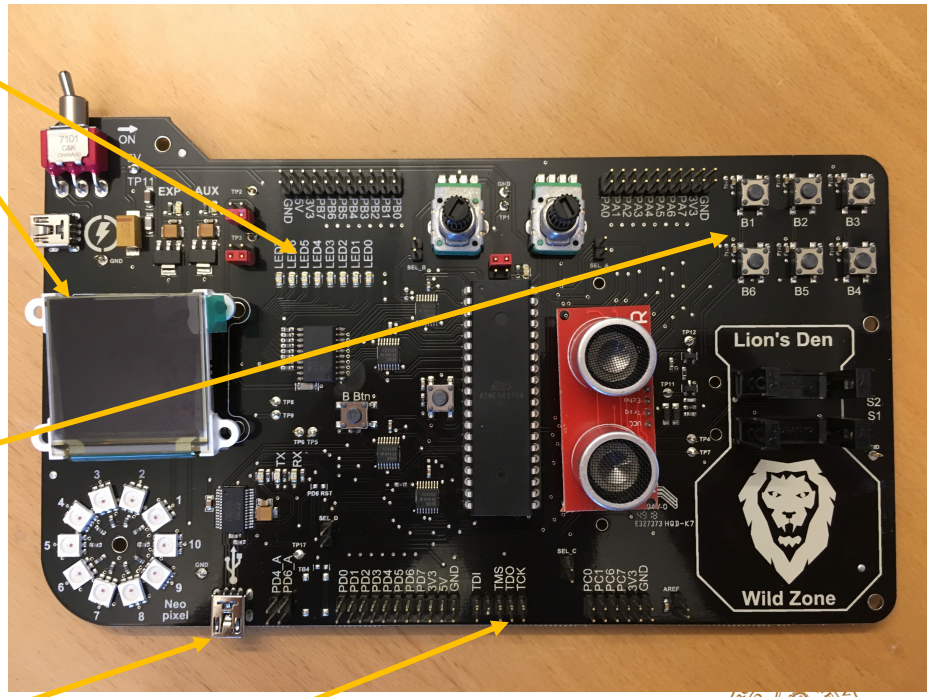
Datorns delar

Output device

Input device

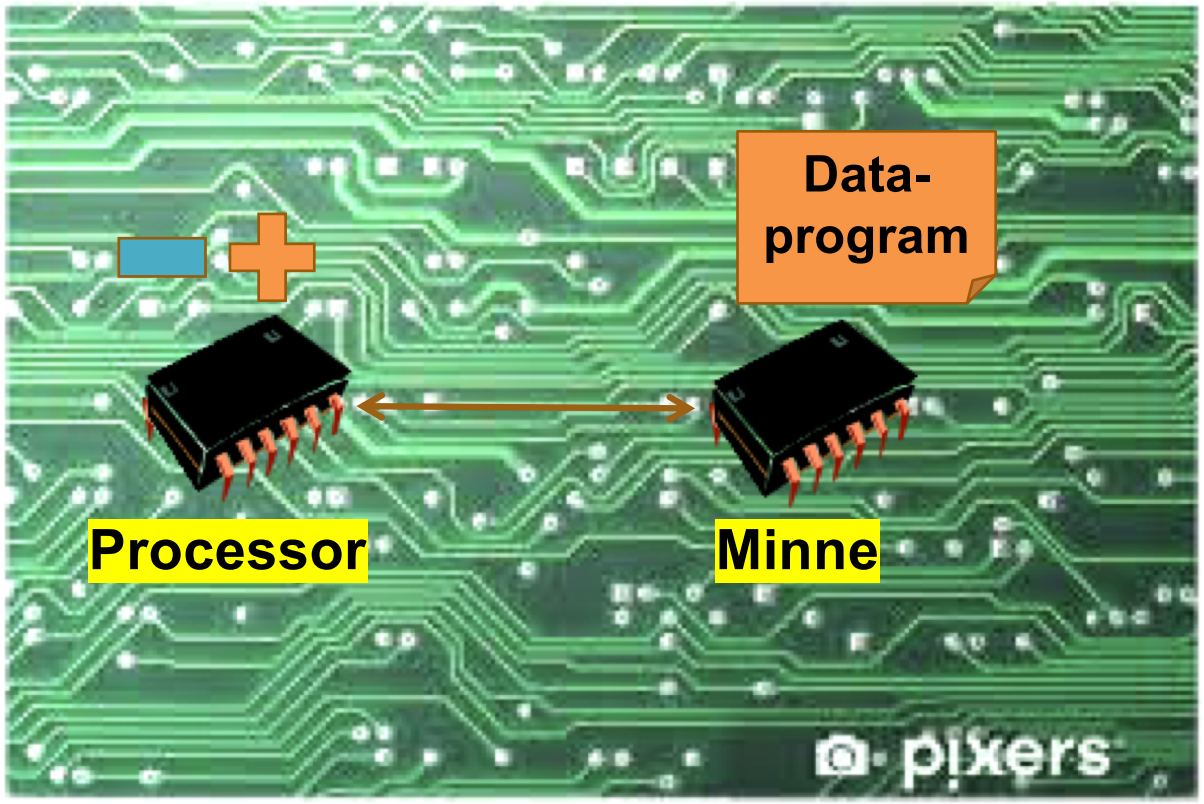
Network cable





LUNDS UNIVERSITET

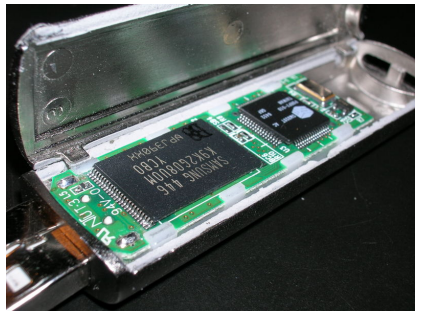
Dator



LUNDS UNIVERSITET

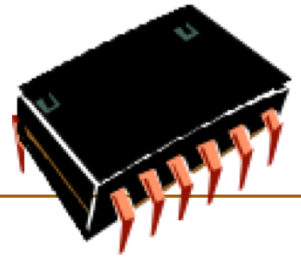
Minne

- Volatile main memory
  - Förlorar information (instruktioner och data) när strömmen stängs av
- Non-volatile secondary memory
  - Magnetic disk (hårddisk)
  - Flash memory
  - Optical disk (CDROM, DVD)

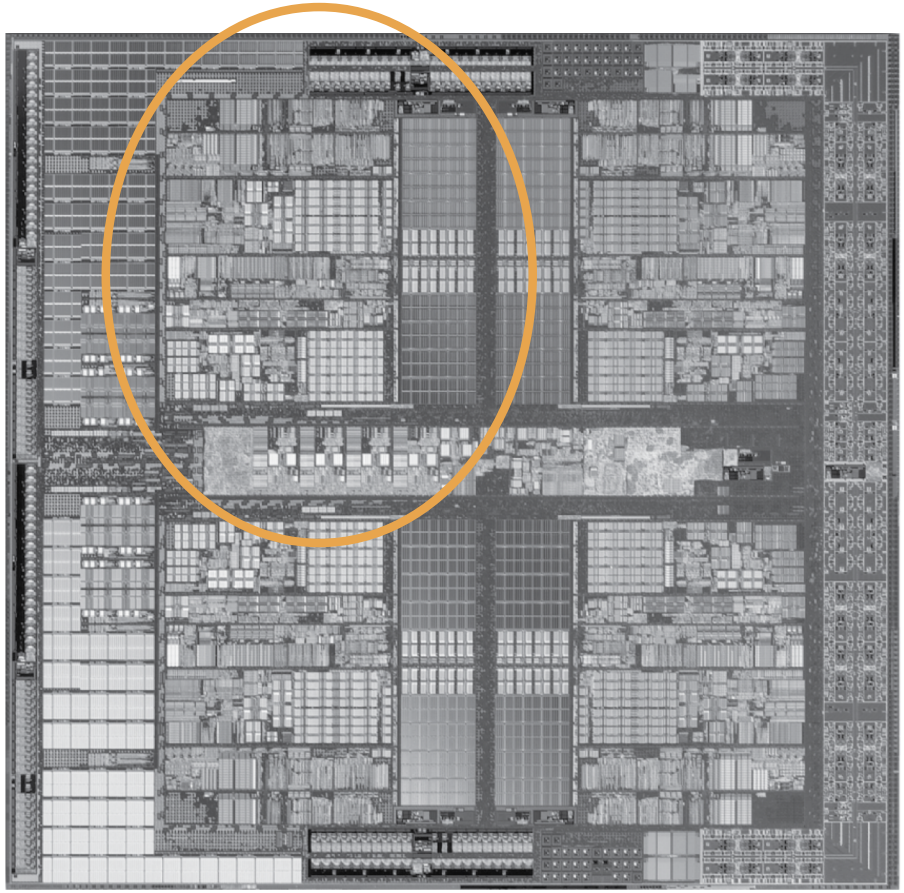


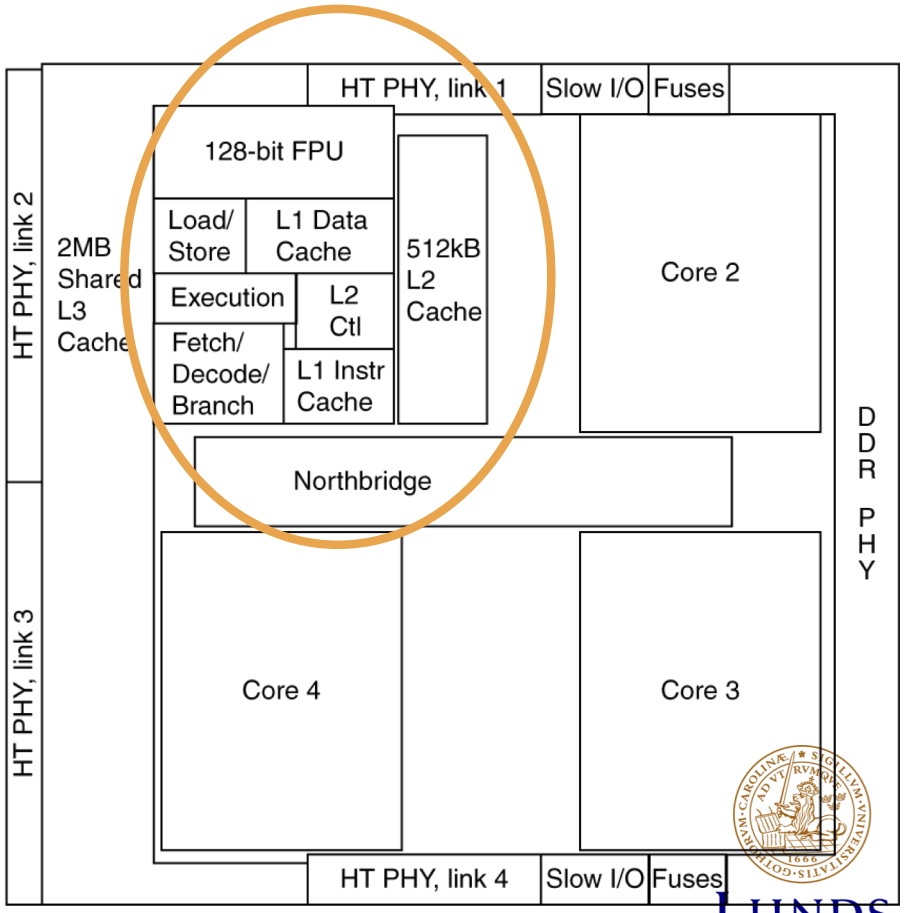
LUNDS UNIVERSITET

Processorn



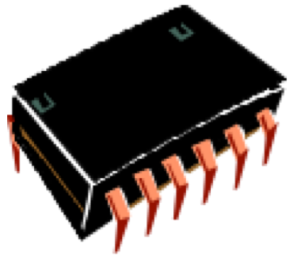
- AMD Barcelona: 4 processor cores

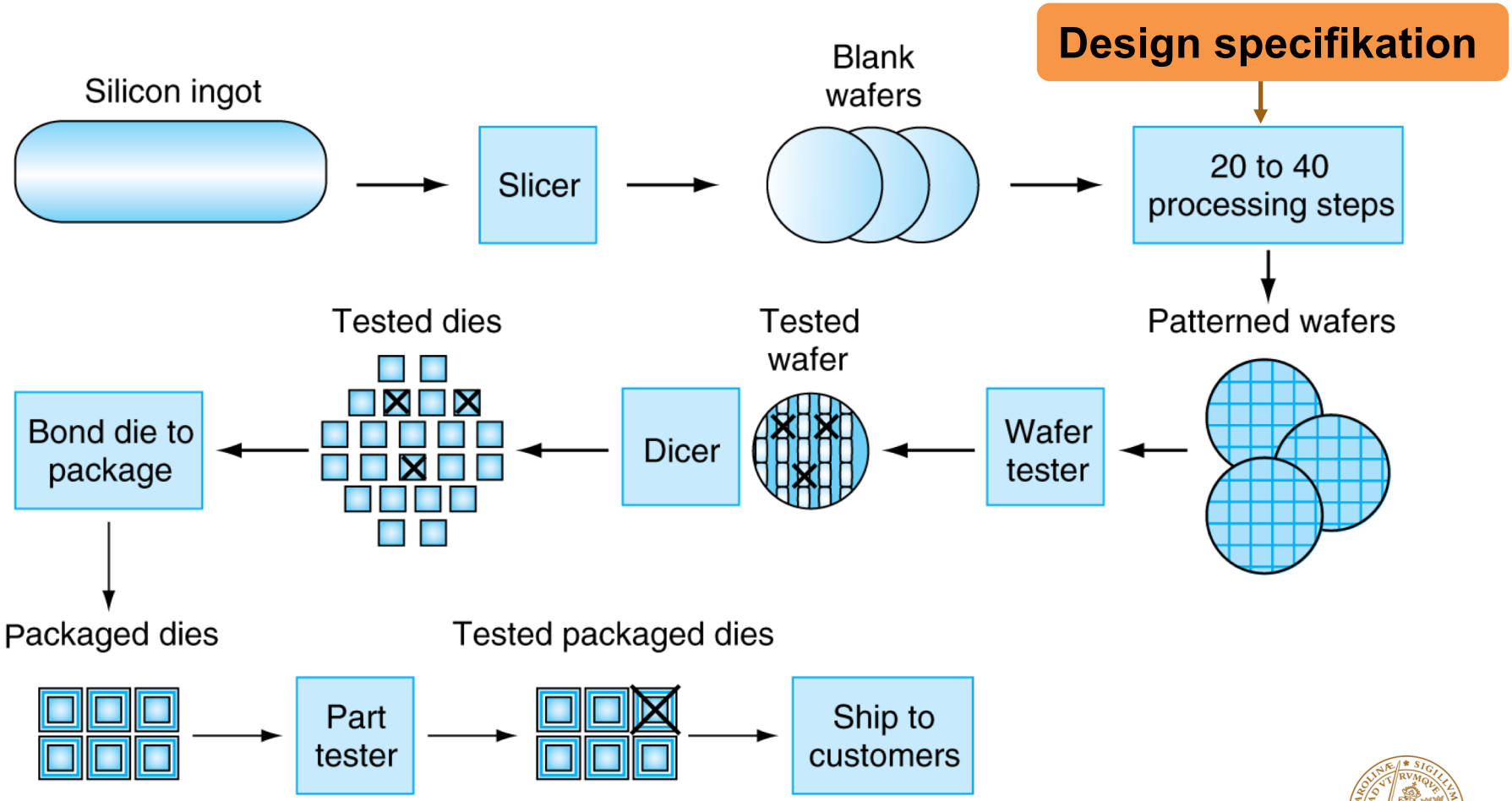




LUNDS UNIVERSITET

IC tillverkning

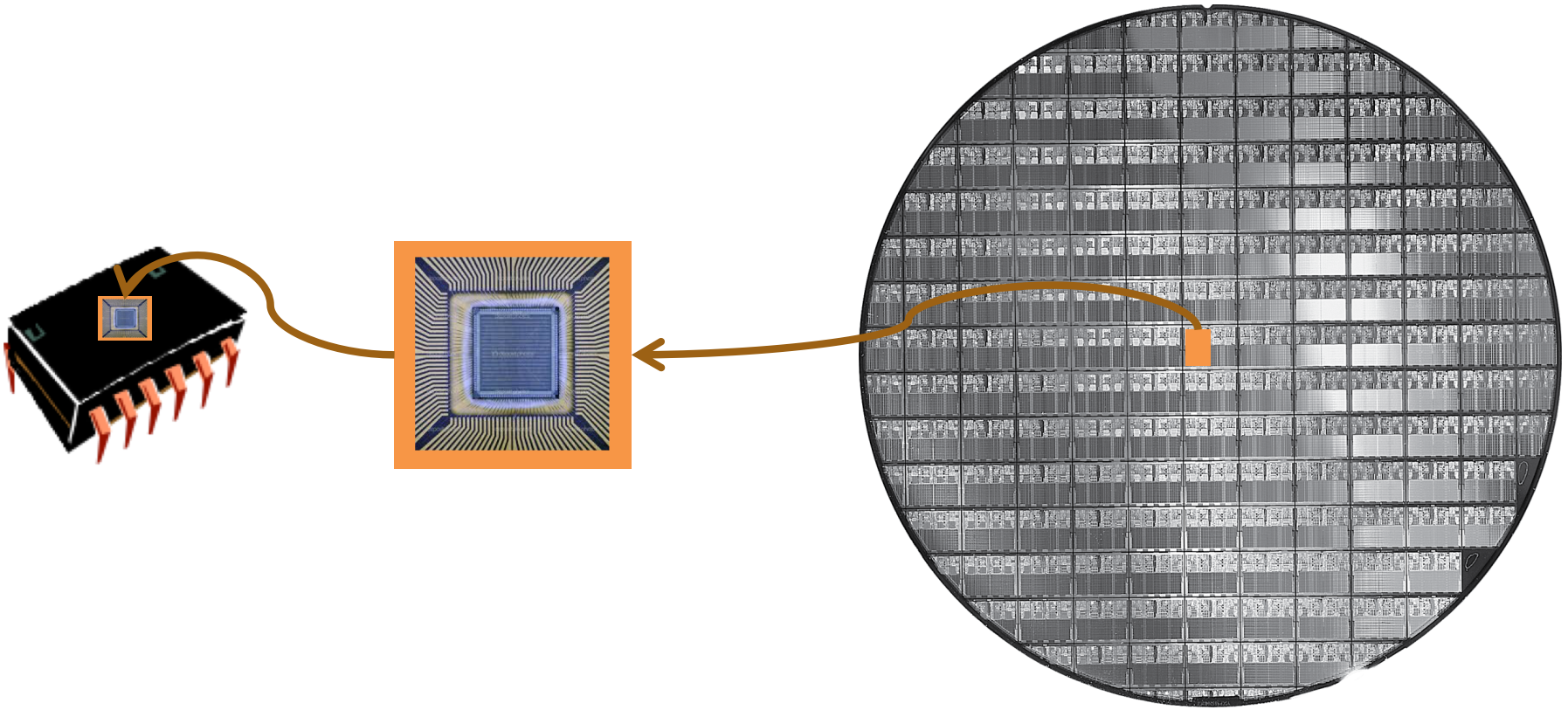




LUNDS UNIVERSITET

IC, die och wafer

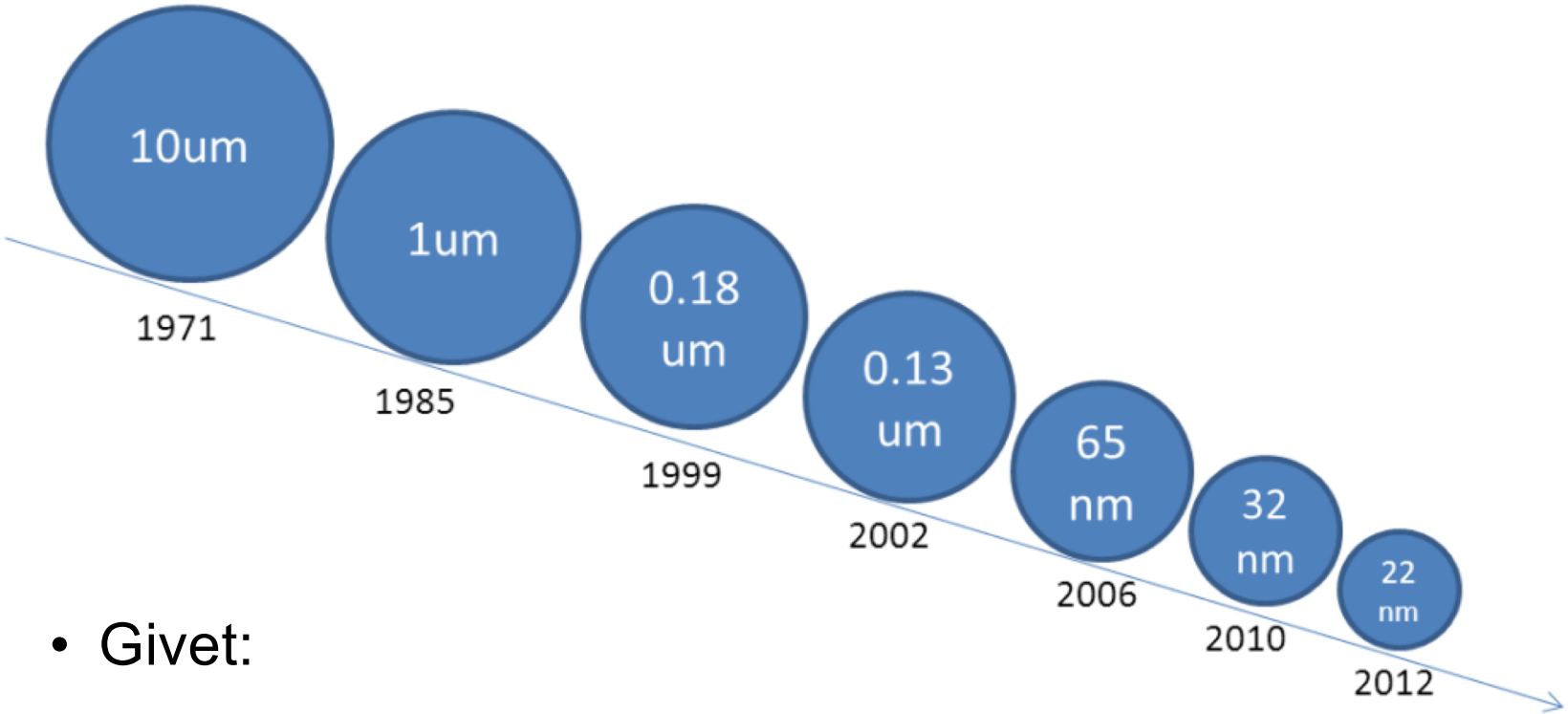
SPARC64 X+ (Athena+)
3 miljarder transistorer
600 mm² (2.5cmx2.5cm)



- AMD Opteron X2: 300mm wafer, 117 chips, 90nm technology
- X4: 45nm technology

LUNDS UNIVERSITET

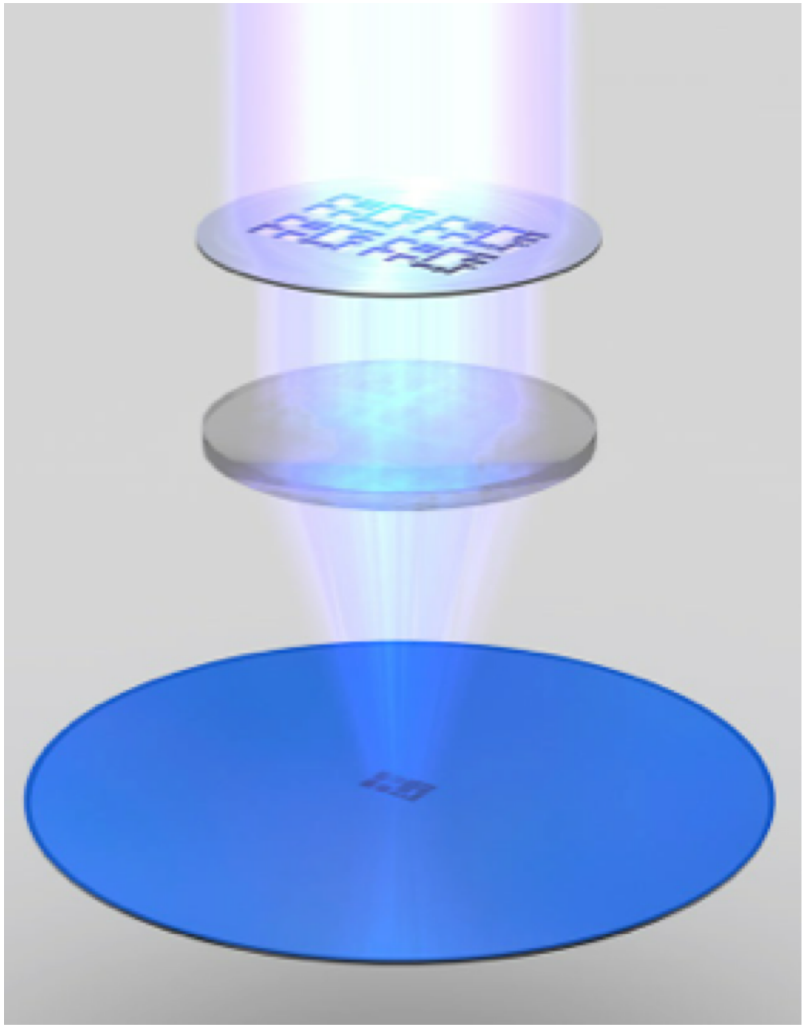
Trend: Minituarization

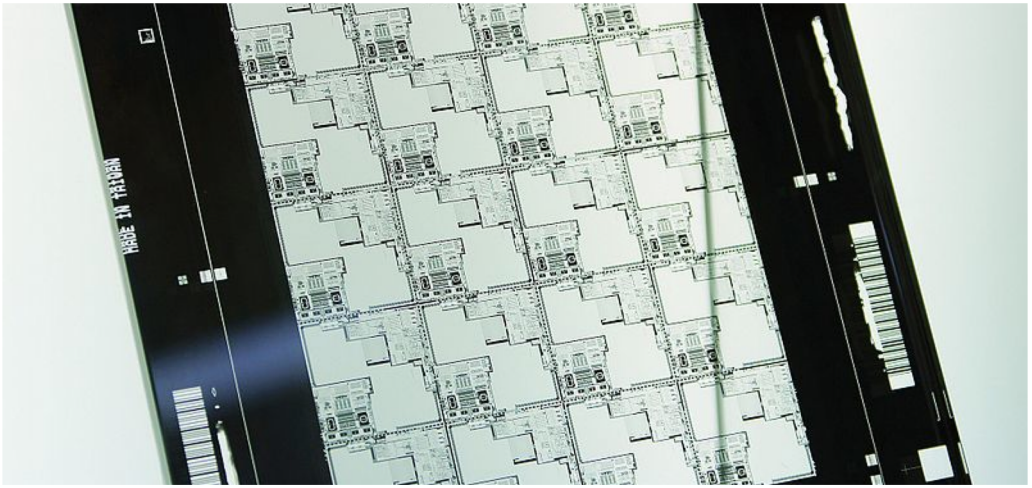


- Givet:
  - Ett hårstrå är 20mikrometer tjockt
  - I en 20nm technologi kan delar vara 20nm
- Skillnad 1000 gånger

LUNDS UNIVERSITET

IC tillverkning

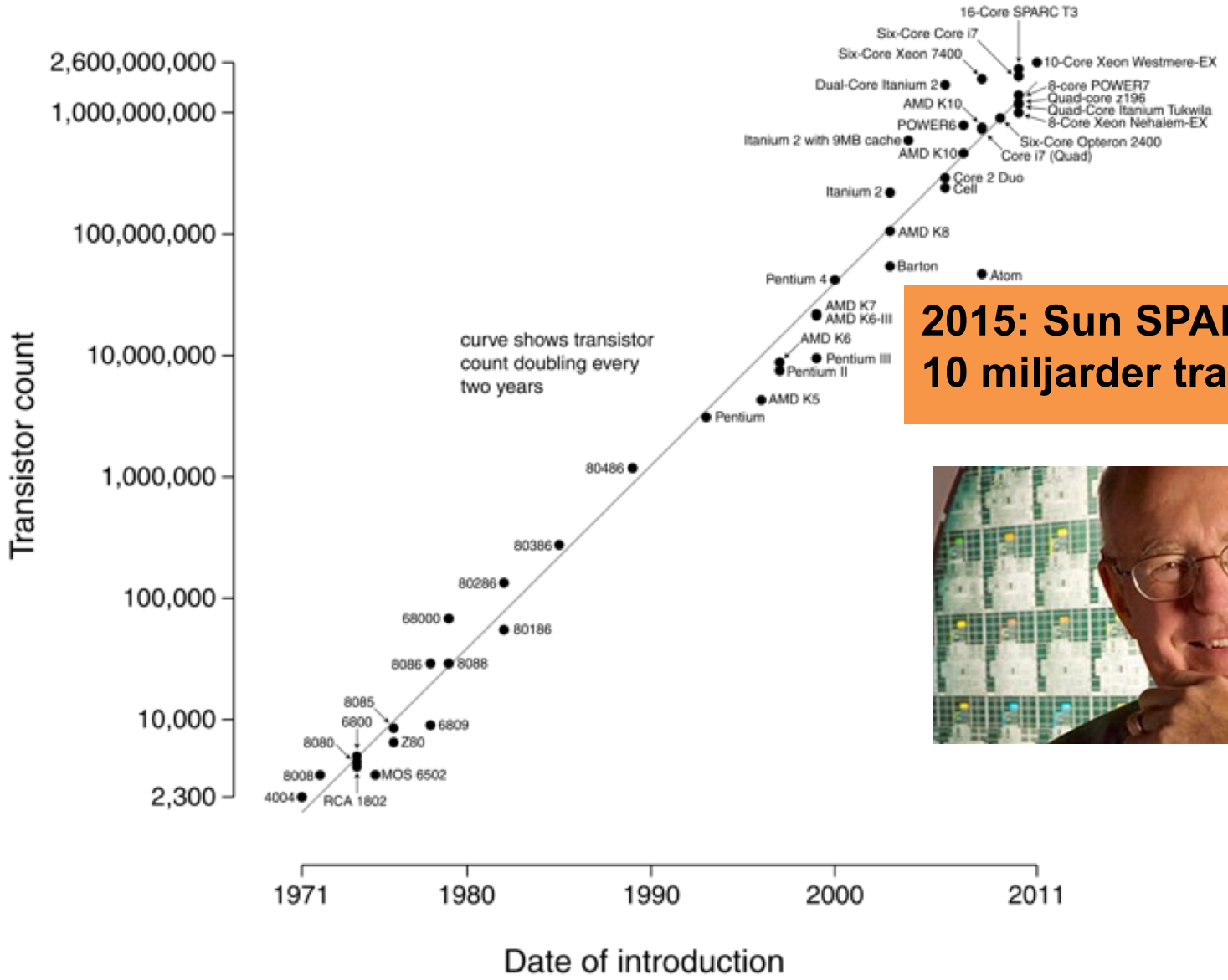




Kostnad för att iordningställa tillverkning i en 45 nm process ligger kring $200–500 miljoner. Att köpa en mask kan kosta mellan $1,000 to $100,000 och det krävs upp till 30 olika masker för en komplett tillverkning.

LUNDS UNIVERSITET

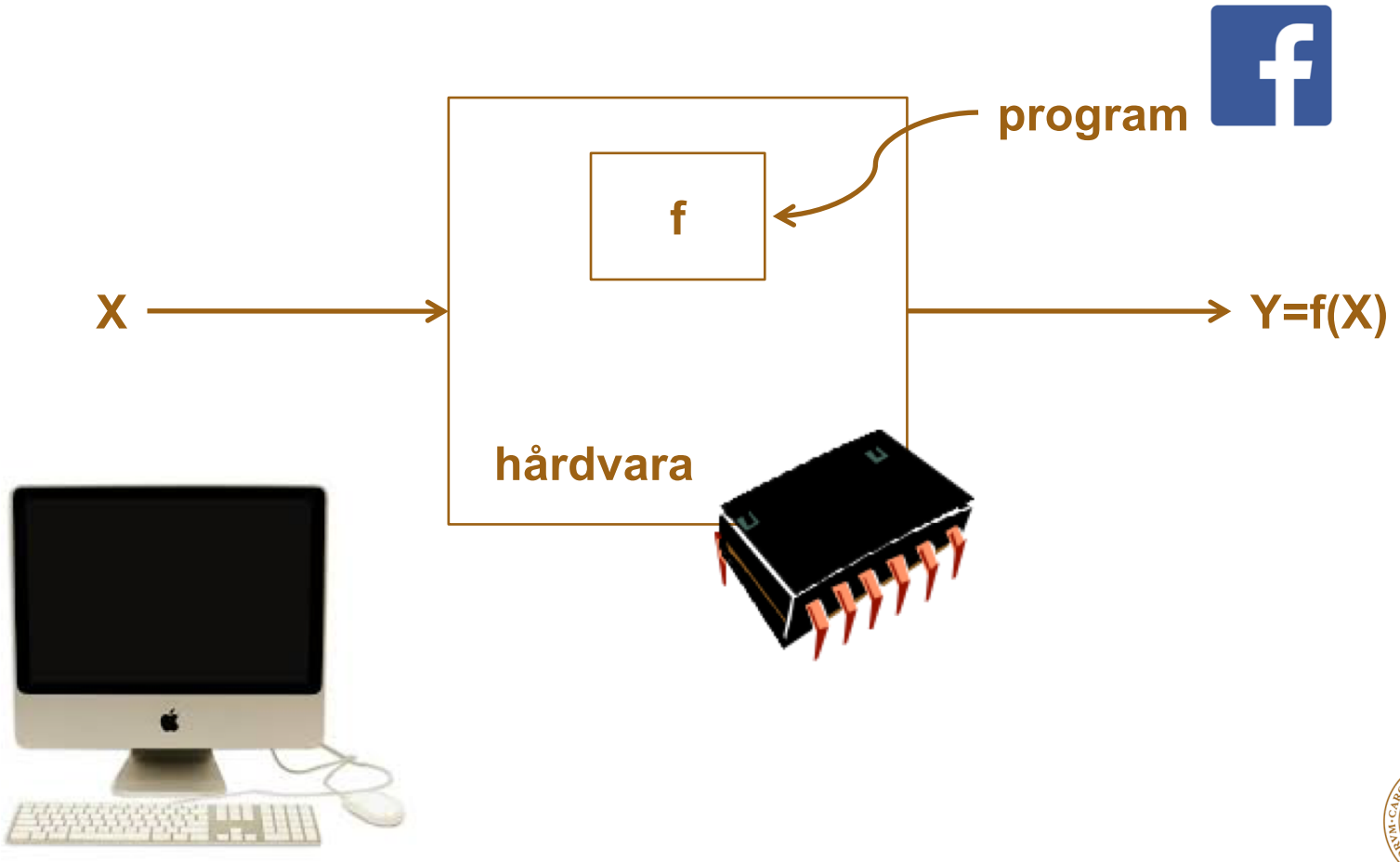
Moores lag – Trend: Transistor count



2015: Sun SPARC M7 - 10 miljarder transistorer

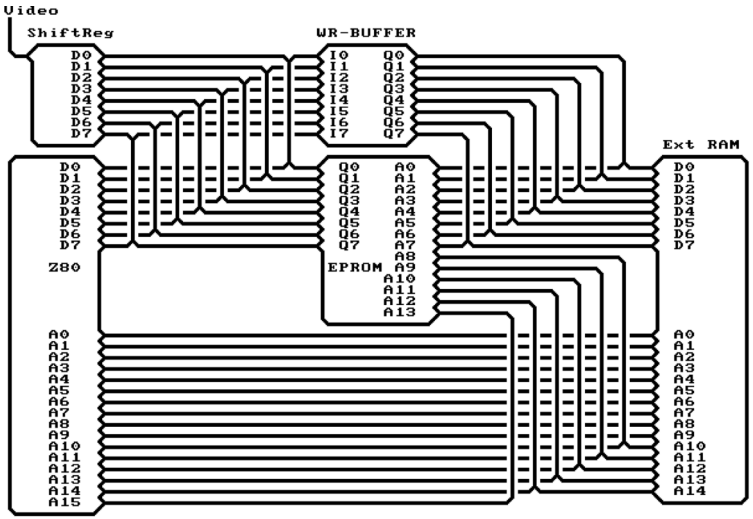
LUNDS UNIVERSITET

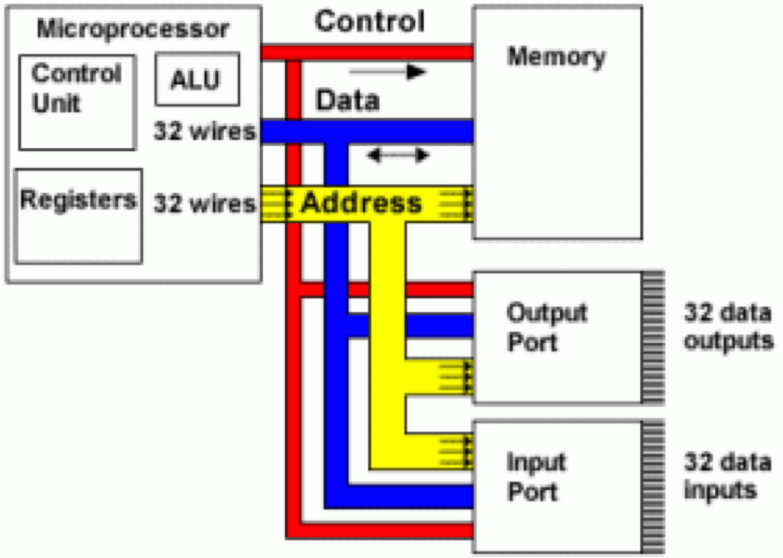
Dator=hårdvara+programvara

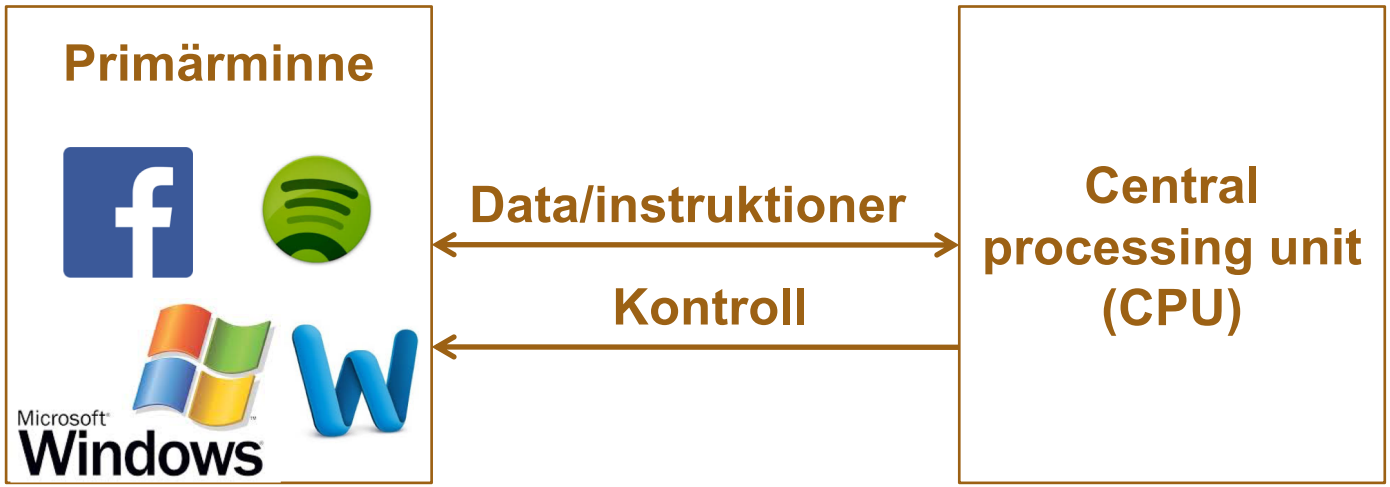


LUNDS UNIVERSITET

Dator



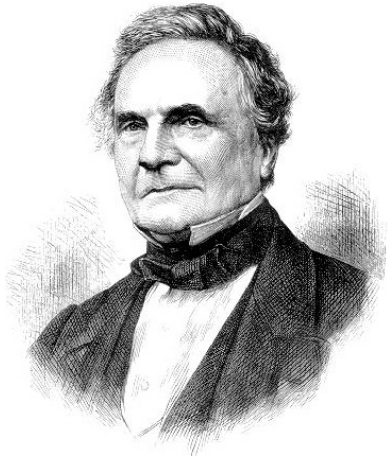




LUNDS UNIVERSITET

Dator

- Charles Babbages differensmaskin
  - Aritmetisk enhet för logiska och aritmetiska operationer
  - Kontrollenhet
  - Instruktioner exekveras sekventiellt
- John von Neumann arkitektur
  - Gemensamt minne för instruktioner och data
- Harvardarkitektur (Harvard Mark I (1944))
  - Separat minne för instruktioner och data



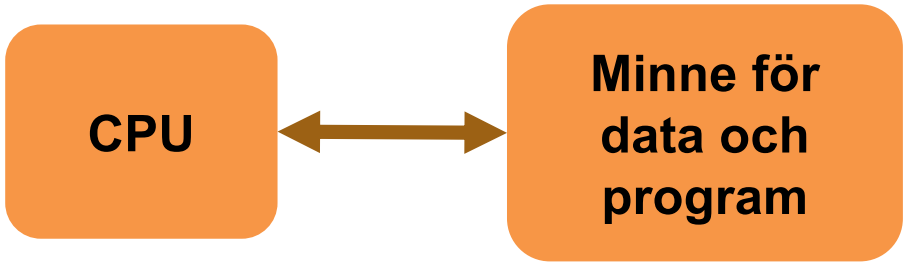
- Charles Babbages (1791-1871)

- John von Neumann (1903-1957)

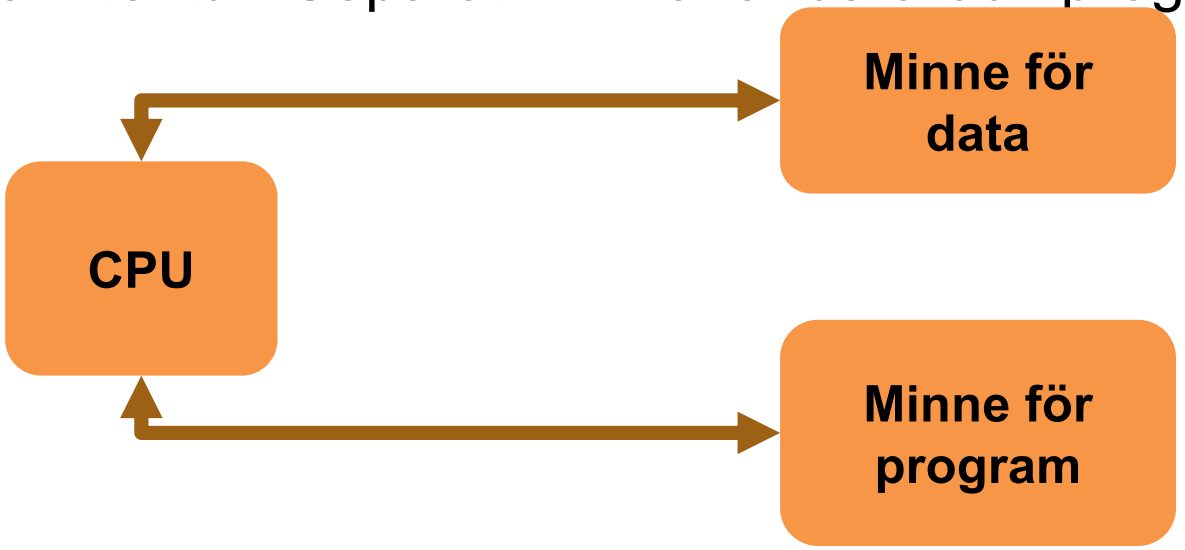
Dator

```
#include <stdio.h>
int main() {
   int x, y, z;
   printf(”Ge två tal: ");
   scanf("%d %d", &x, &y);
   z = x + y;
   printf("%d + %d = %d",x,y,z);
   return 0;
}
```

- John von Neumann arkitektur:
  Gemensamt minne för data och program

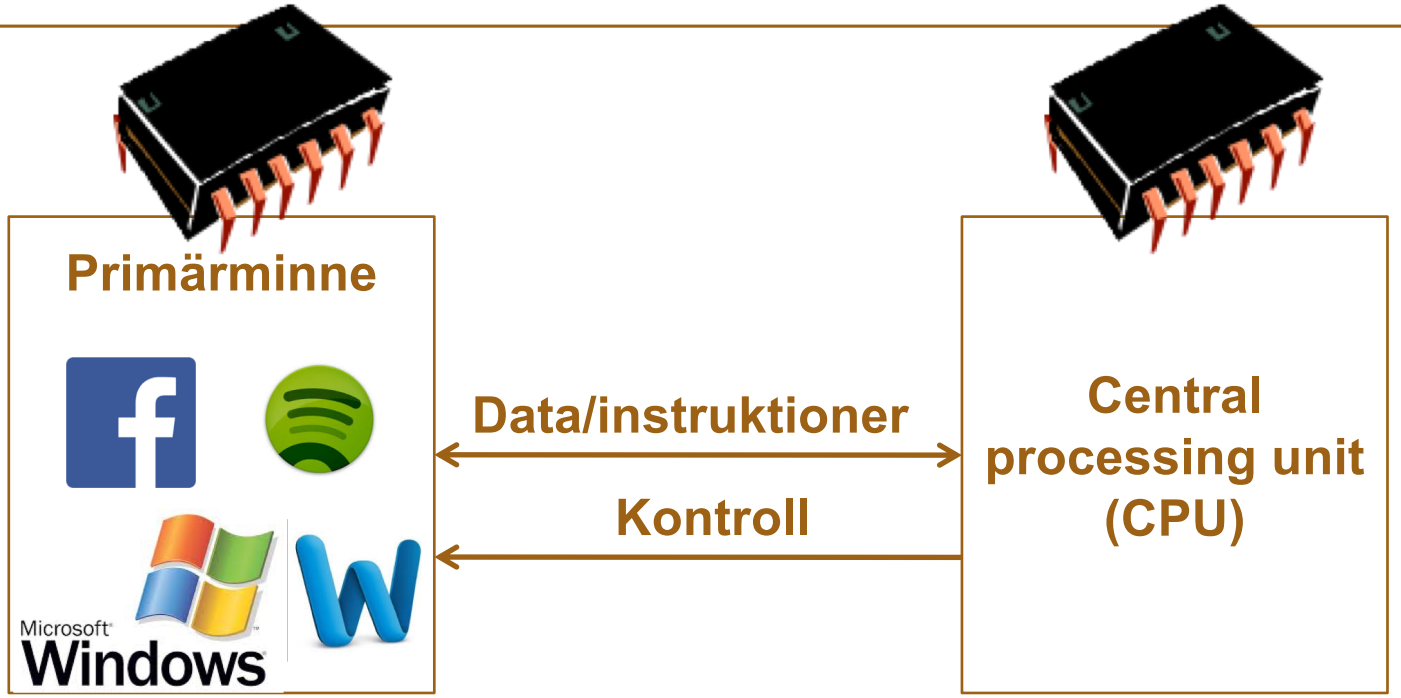


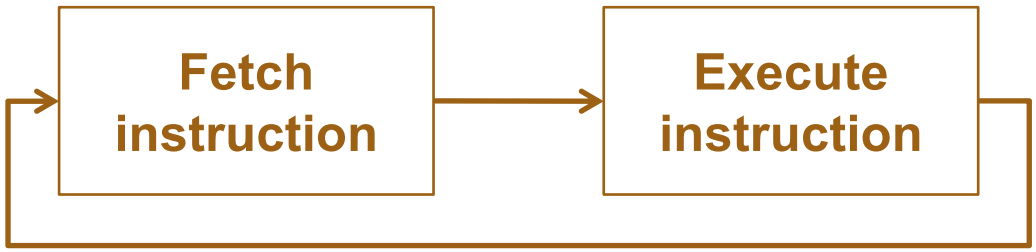
- Harvardarkitektur: Separat minne för data och program



LUNDS UNIVERSITET

Dator





LUNDS
UNIVERSITET

Användarprogram, OS och hårdvara

- Applikationsprogram, t ex Word, Facebook, är ofta skrivna i ett högnivåspråk, t ex C, C++.
- System programvara
  - Kompilatorer: Översätter program i högnivåspråk till maskinspråk
  - Operativsystem:
    - » Hantera I/O, minne och schemaläggning av jobb (tasks)

- Hårdvara
  - Processor, minne, I/O-kontrollenhet

Program

- För att bestämma vad som görs behövs program,och för att göra program behövs programmeringsspråk.
- Programmeringsspråk:
  - Högnivåspråk (C, C++)
  - Assemblyspråk (t ex ADD R1, R2)
  - Maskinspråk (t ex 001101….101)

Högnivåspråk: datorns arkitektur osynlig

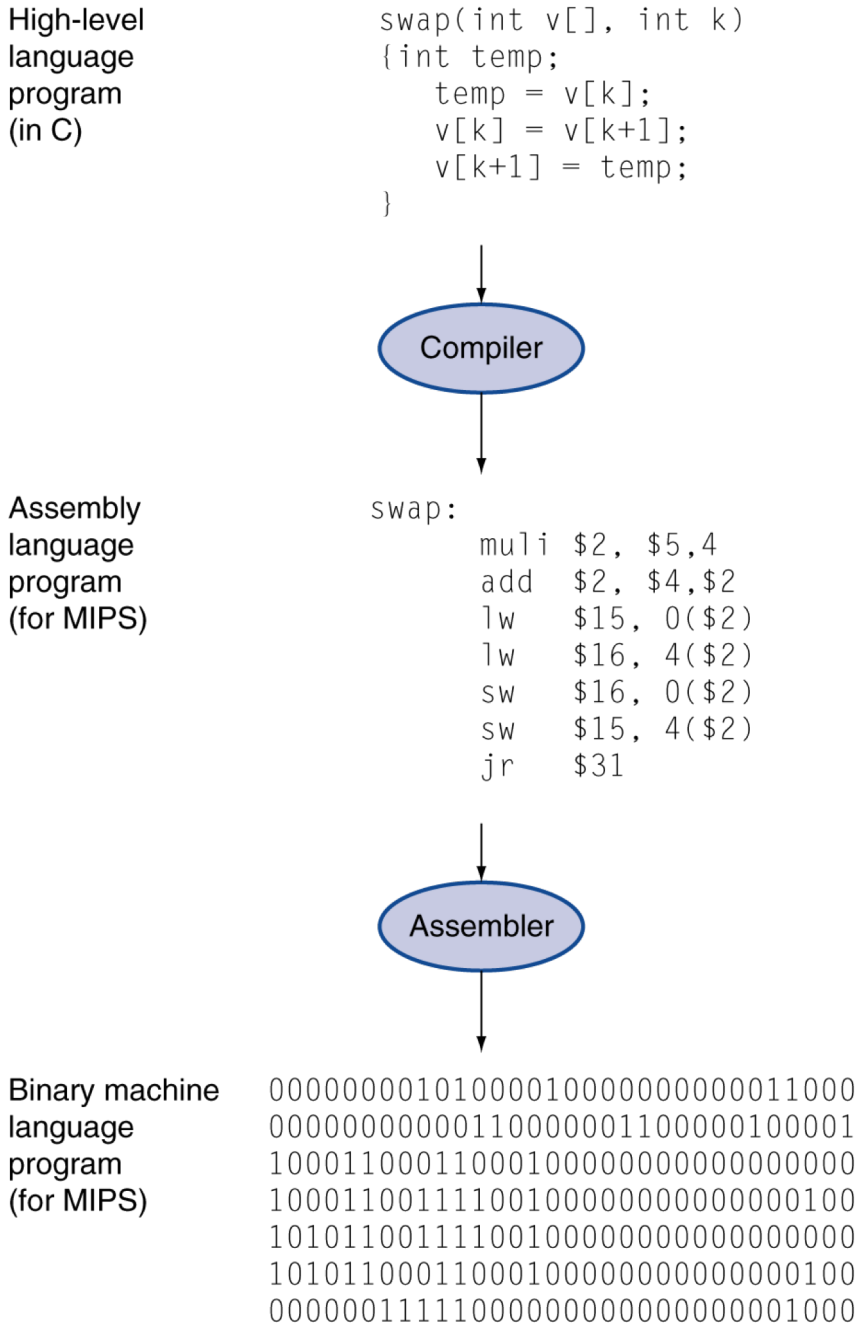
Assemblyspråk: datorns arkitektur synlig

Maskinspråk: datorns arkitektur synlig

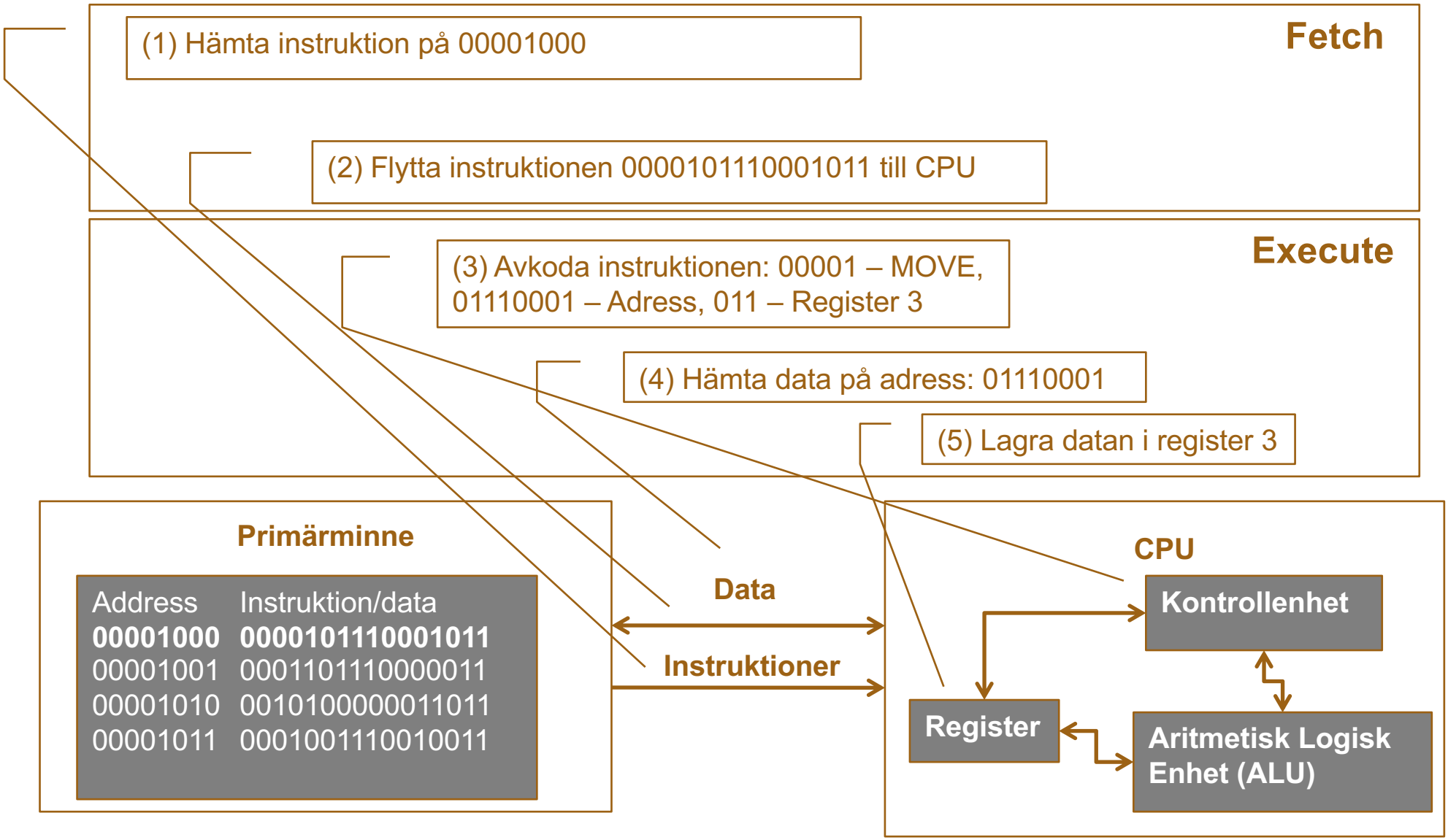
LUNDS UNIVERSITET

Program

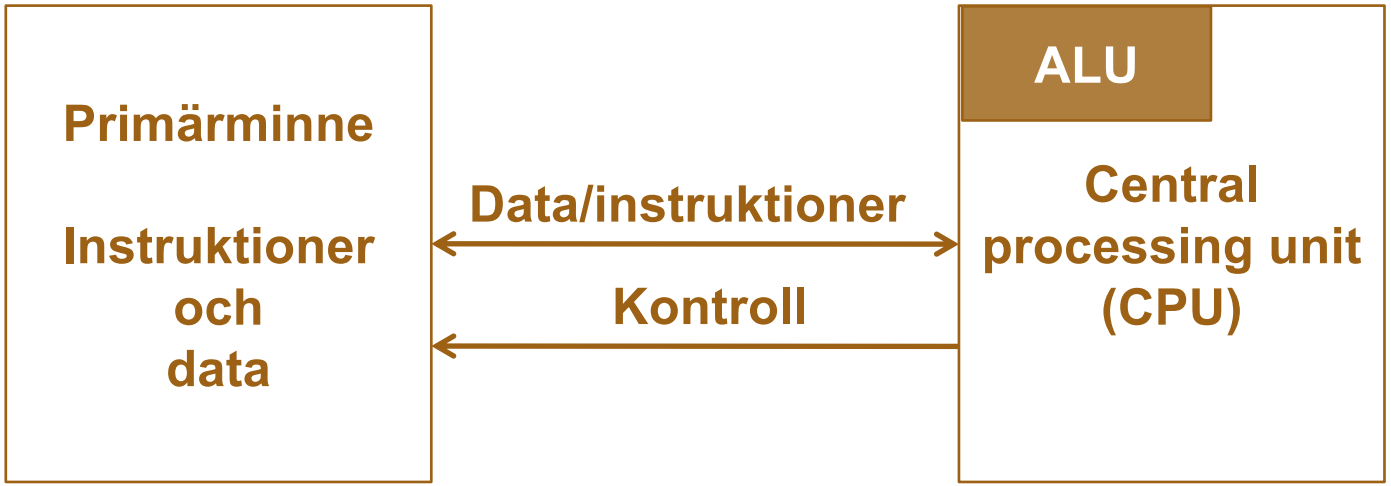
- Abstraktionsnivå:
  - Högnivåspråk
  - Assemblyspråk
  - Maskinspråk

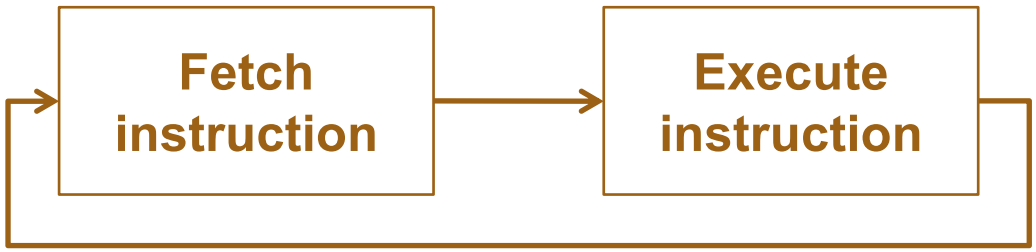


Dator


(1) Hämta instruktion på 00001000
Fetch
(2) Flytta instruktionen 0000101110001011 till CPU
(3) Avkoda instruktionen: 00001 – MOVE, 01110001 – Adress, 011 – Register 3
Execute
(4) Hämta data på adress: 01110001
(5) Lagra datan i register 3
Primärminne
Address Instruktion/data
00001000 0000101110001011
00001001 0001101110000011
00001010 0010100000011011
00001011 0001001110010011
Data
Instruktioner
CPU
Kontrollenhet
Register
Aritmetisk Logisk Enhet (ALU)

ALU – arithmetic logic unit

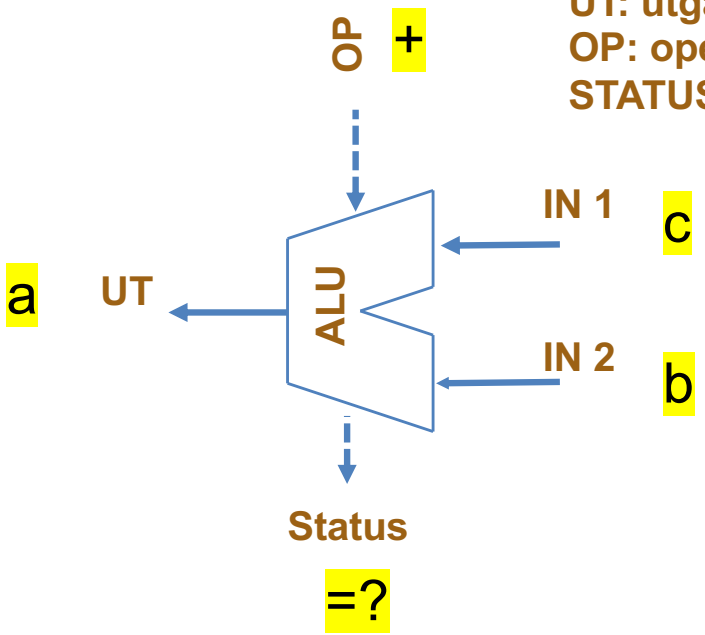




LUNDS UNIVERSITET

ALU – arithmetic logic unit

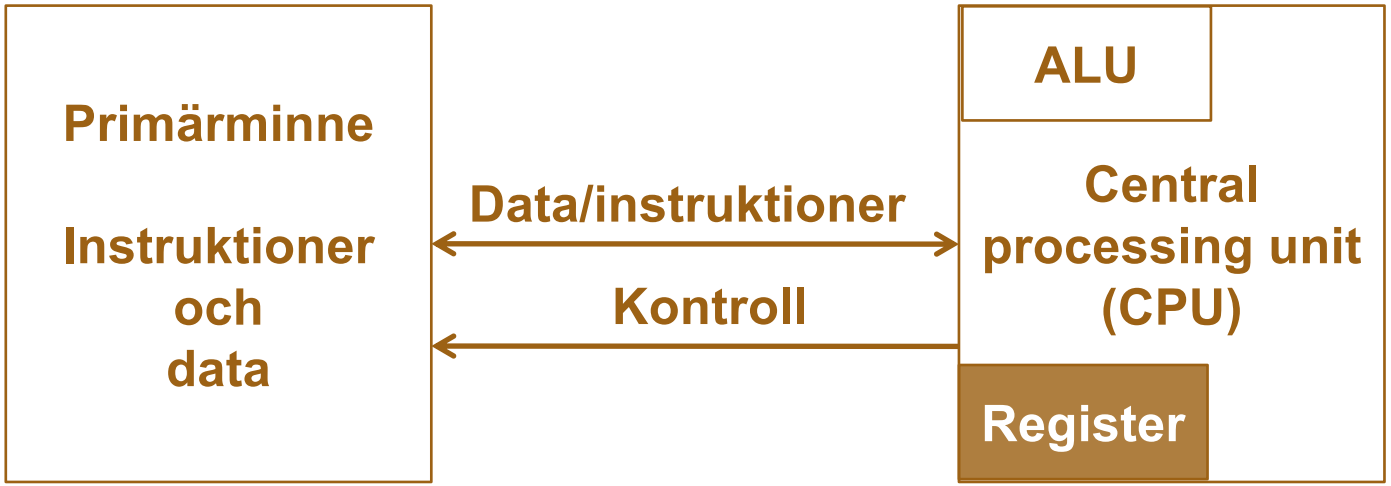
- Högnivåspråk: `a=b+c;`
- Assembly: `ADD a,b,c;`

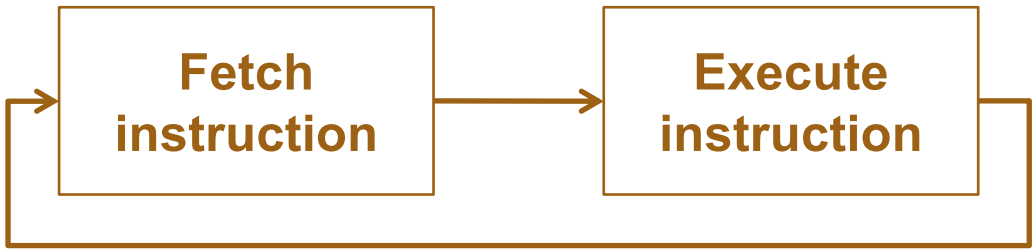


IN 1, IN 2: Ingångar
UT: utgång (resultat)
OP: operation (ADD, SUB, AND, ….
STATUS: Overflow, Zero, ….

LUNDS UNIVERSITET

Register





LUNDS
UNIVERSITET

Register

- Alla processorer har någon form av register fil för temporär lagring
- Register synliga för användare, som kan vara:
  - Helt generella, eller
  - Specifika för data och instruktioner och/eller
  - Specifika för heltal och flyttal
  - Speciella för t ex multiplikation/division för att kunna hantera resultat eller för adressberäkningar: basregister och stackpekare

LUNDS UNIVERSITET

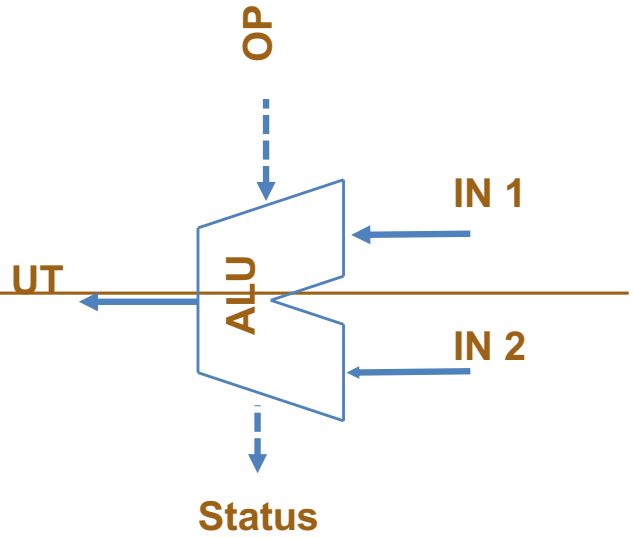
Register

- Kontroll och statusregister
  - Inte direkt kontrollerbara av programmeraren, t ex:
    - » Programräknare (program counter (PC)): adress till den instruktion som ska hämtas.
    - » Instruktions register (IR): senaste instruktionen som hämtats
    - » Memory address register (MAR): adress som ska läsas/skrivas
    - » Memory buffer register (MBR): data som ska skrivas eller data som lästs från minnet
    - » Program status word (PSW): Olika flaggor (är avbrott på eller av, supervisor mode)

Register

- Exempel:
  - Varför använda register?
  - Hur bestäms var lagring ska ske (register eller minnet)?
- Exempel:
  - Högnivåspråk: `a=b+c;`
- Lösning: `ADD a,b,c` //a <- b+ c (a, b, och c är minnesplatser)

Register



- Högnivåspråk: `a=b+c+d;`
- Alternativ 1:

```
                    //a, b, c, d ligger i minnet
ADD a,b,c           //hämta b och c, addera b och c, spara resultat a
ADD a,a,d           //hämta a (som nu är b+c) och d, adderar a och
                     d, spara resultat i a
```

- Antal minnesaccesser:

  6=3+3

- Tid för en minnesaccess: 100 ns gör att minnesläsning/skrivning tar 600 ns

LUNDS UNIVERSITET

Register

- Högnivåspråk: `a=b+c+d;`
- Alternativ 2:

```
ADD r1, b, c   //hämta b och c, addera b och c,
                 spara resultat i register 1
ADD a, r1, d   //hämta d och r1, adderar b+c
                 (som är i r1) med d, spara resultat i a
```

- Antal minnesaccesser:

  4=2+2

- Tid för en:
  - minnesaccess: 100 ns
  - registeraccess: 1 ns:
- Minnesläsning/skrivning tar 400 + 2 ns = 402 ns

30% sparad tid
(600-402)/600

LUNDS UNIVERSITET

Primärminne

LOAD – läs data i minne

- Processorn läser data från en given minnesplats (adress)

STORE – skriv data i minne

- Processorn skriver data på en given adress i minnet

Adress	Instruktion/Data
0	1111 0000
1	1010 0101
2	1100 0011
3	0011 0011
4	1111 1111
5	0000 1111
6	1111 0000
7	1010 1010

Variabler och minne

Varför deklarera variabler?

- Exempel, C deklaration:

```
x unsigned char; // värde mellan 0 och 255
```

1 byte – 8 bitar ger att dessa tal kan representeras:

Binärt	Decimalt
0000 0000	0
0000 0001	1
0000 0010	2
0000 0011	3
…….	
1111 1101	253
1111 1110	254
1111 1111	255

LUNDS UNIVERSITET

Variabler och minne

- Exempel, C deklaration

```
x signed char;    // värde mellan -128 och 127
```

1 byte – 8 bitar: Hur representera “-” talen?

Binärt	Decimalt
0000 0000	0
0000 0001	1
0000 0010	2
0000 0011	3
…….	
1111 1101	?
1111 1110	?
1111 1111	?

Variabler och minne

- Alternativ 1: MSB är teckenbit //Most Significant Bit
- Får 2 nollor (-0 och 0)

Binärt	Decimalt
0000 0000	0
0000 0001	1
0000 0010	2
…	….
1000 0000	-0
….	….
1111 1110	-126
1111 1111	-127

Variabler och minne

- Alternativ 2: Två-kompliment

Binärt	Decimalt
0000 0000	0
0000 0001	1
0000 0010	2
…	….
1000 0000	-127
….	….
1111 1110	-2
1111 1111	-1

- Exempel: -1 (1111 1111) + 1 = 0

Minne

- Exempel:

  C deklaration: `x char;`
  // char behöver 1 byte

LOAD r1, 3

// ladda register r1 med data som finns på adress 3

STORE r1, 3
/ spara det som finns i register r1 på adress 3

Adress	Instruktion/Data
0	1111 0000
1	1010 0101
2	1100 0011
3	0011 0011
4	1111 1111
5	0000 1111
6	1111 0000
7	1010 1010

Minne

- Exempel:

  C deklaration: `x int;`
  // int behöver 2 bytes

LOAD r1, 3

// ladda register r1 med data som finns på plats 3 och ?

STORE r1, 3
/ spara det som finns i register r1 på plats 3 och ?

Adress	Instruktion/Data
0	1111 0000
1	1010 0101
2	1100 0011
3	0011 0011
4	1111 1111
5	0000 1111
6	1111 0000
7	1010 1010

Byte eller wordadresserat minne

Adress	Byte	Data
0	0	1111 0000
1	1	1010 0101
2	2	1100 0011
3	3	0011 0011
4	4	1111 1111
5	5	0000 1111
6	6	1111 0000
7	7	1010 1010

Adress	Byte	Data
0	0	1111 0000
	1	1010 0101
	2	1100 0011
	3	0011 0011
1	4	1111 1111
	5	0000 1111
	6	1111 0000
	7	1010 1010

Byte eller word adresserat minne?

- Byteadresserat
  - Måste hålla koll på vad som ska anses vara ord
  - Exempel: Läs adress 4, som är en del i ett ord.
- Wordadresserat:
  - Läs word (4 byte) per gång.
  - Fragmentering om en byte ska lagras (tar 4 bytes).

Adress	Byte	Data
0	0	1111 0000
1	1	1010 0101
2	2	1100 0011
3	3	0011 0011
4	4	1111 1111
5	5	0000 1111
6	6	1111 0000
7	7	1010 1010

Hur sätts byte ihop till word?

- Byte 4, 5, 6, och 7 bildar ett ord, men hur ser ordet ut?
- Två alternativ:
  - 1111 1111 0000 1111 1111 0000 1010 1010 (ordning 4, 5, 6, 7)
  - 1010 1010 1111 0000 0000 1111 1111 1111 (ordning 7, 6, 5, 4)

Adress	Byte	Data
0	0	1111 0000
1	1	1010 0101
2	2	1100 0011
3	3	0011 0011
4	4	1111 1111
5	5	0000 1111
6	6	1111 0000
7	7	1010 1010

Hur sätts byte ihop till word?

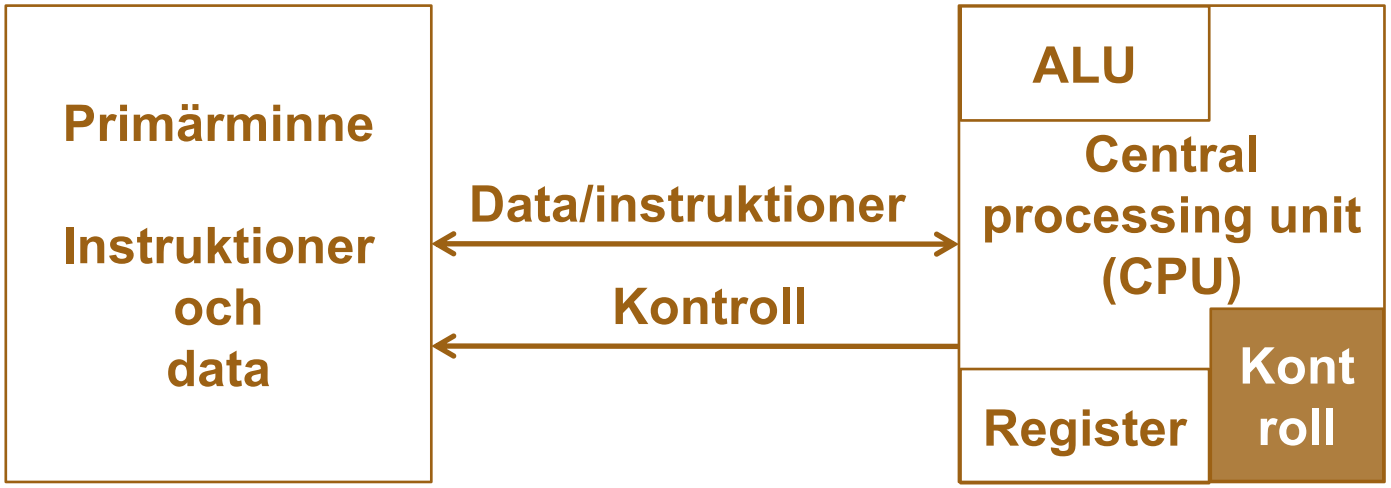
MSB LSB

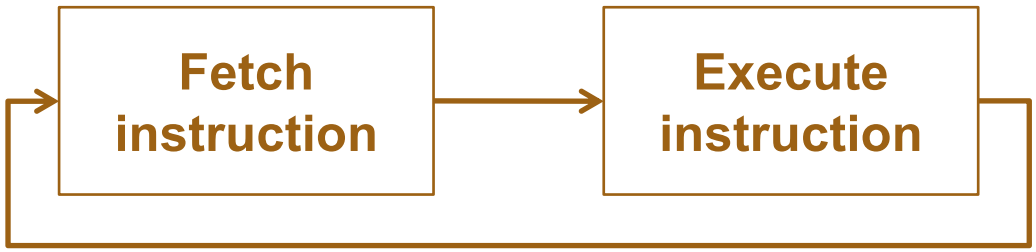
Givet: **1111 1111** **0000 1111** **1111 0000** **1010 1010**

Big-endian	
Adress	Data
….	….
n	1111 1111
n+1	0000 1111
n+2	1111 0000
n+3	1010 1010
….	….

Little-endian	
Adress	Data
….	….
n	1010 1010
n+1	1111 0000
n+2	0000 1111
n+3	1111 1111
….	….

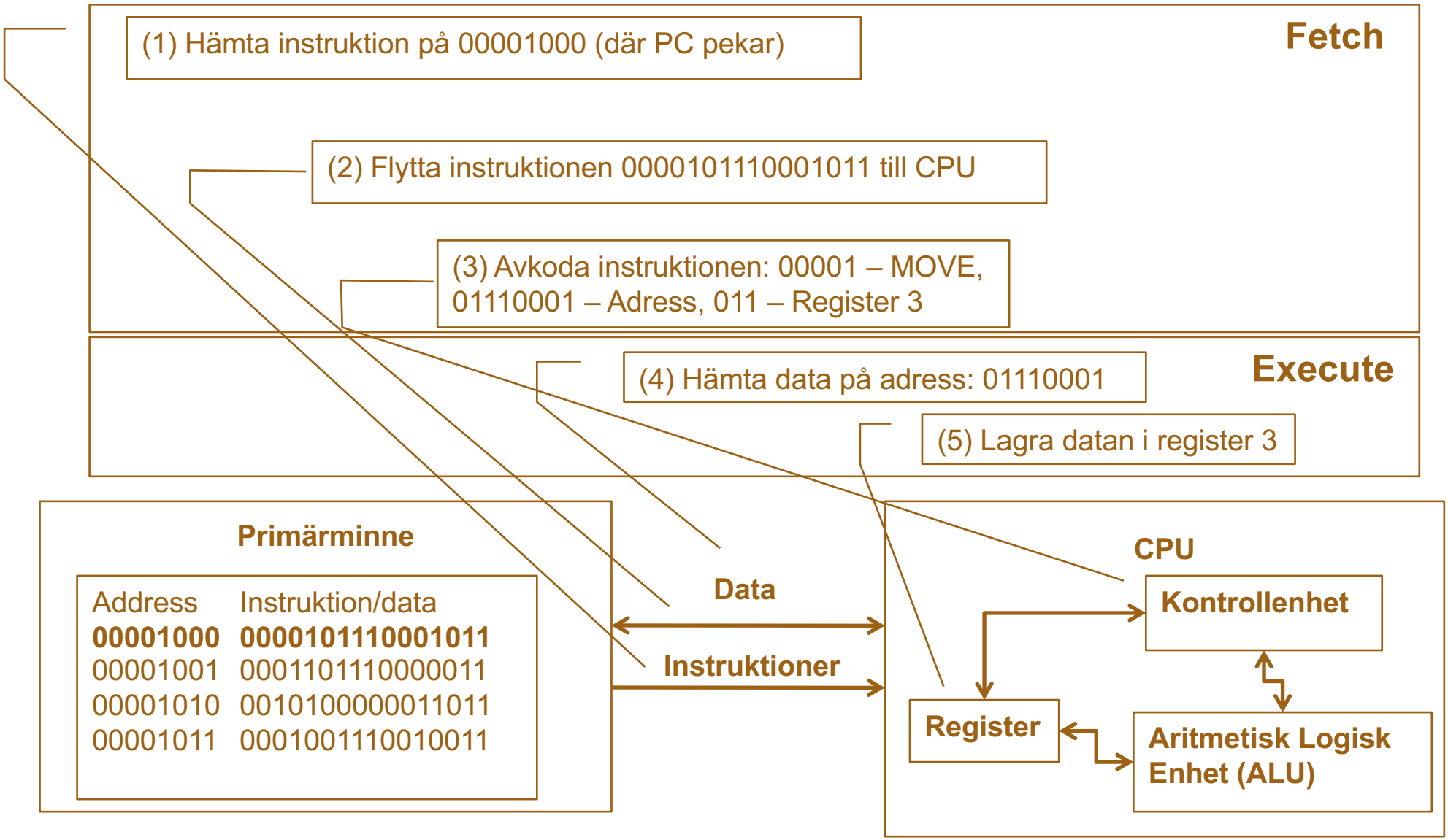
Kontrollenhet



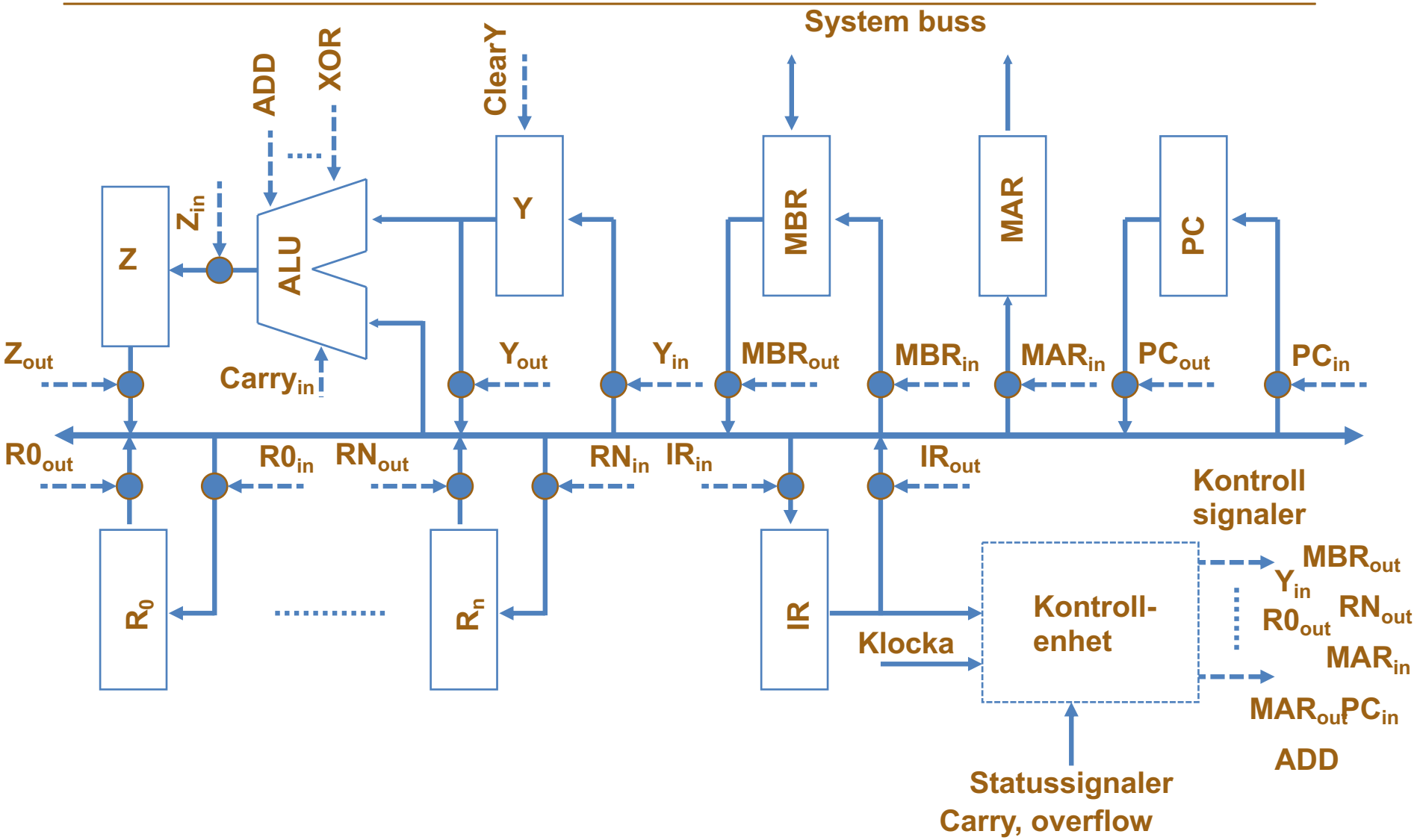


LUNDS
UNIVERSITET

Exekvering av en instruktion



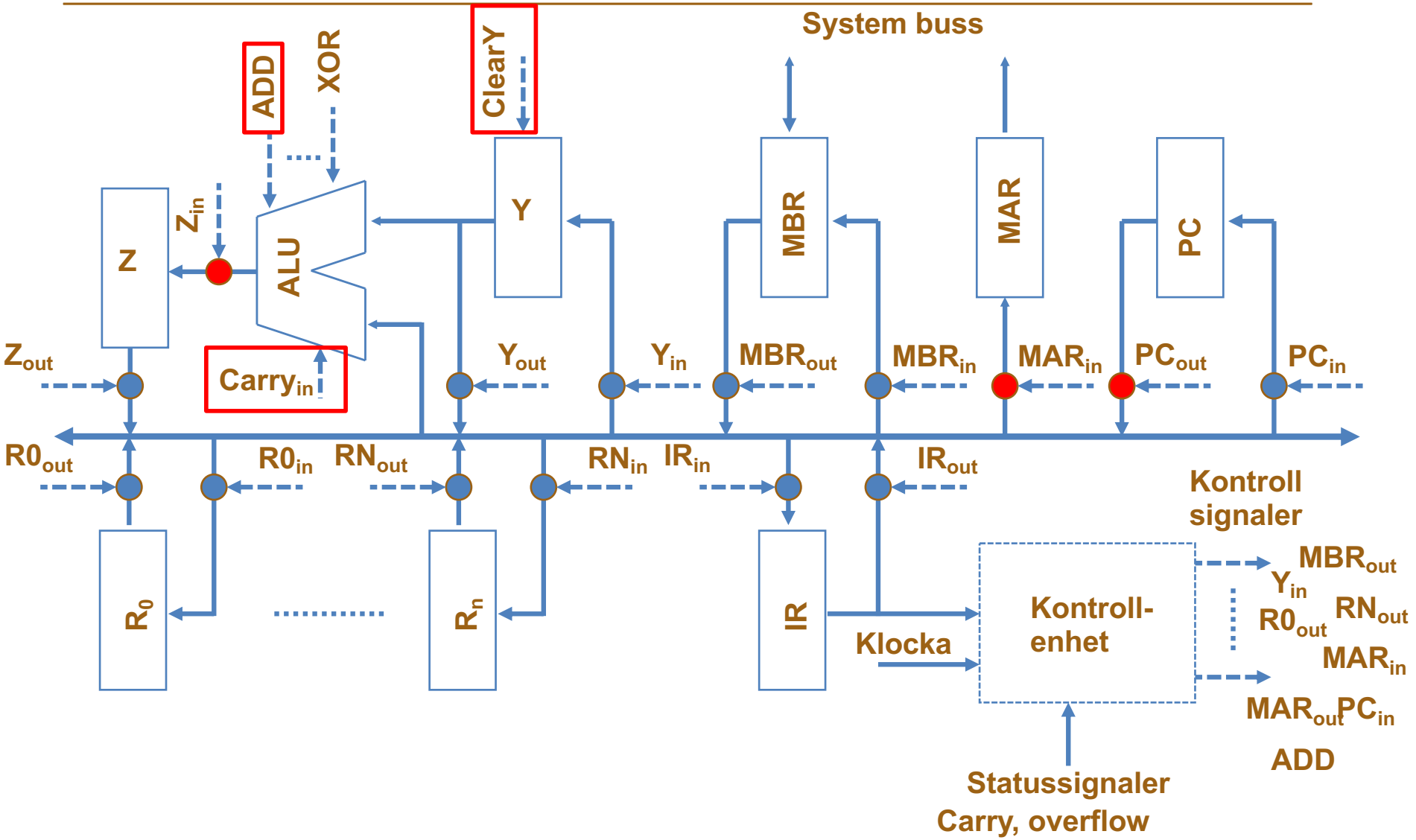
Kontrollenhet



Kontrollenhet

- Instruktion:
  - ADD R1, R3 // R1 <- R1 + R3
- Kontrollsteg:
  1. PC_{out}, MAR_{in}, Read, Clear Y, Carry-in, Add, Z_{in}
  2. Z_{out}, PC_{in}
  3. MBR_{out}, IR_{in}
  4. $R1_{out}$, Y_{in}
  5. $R3_{out}$, Add, Z_{in}
  6. Z_{out}, $R1_{in}$, End

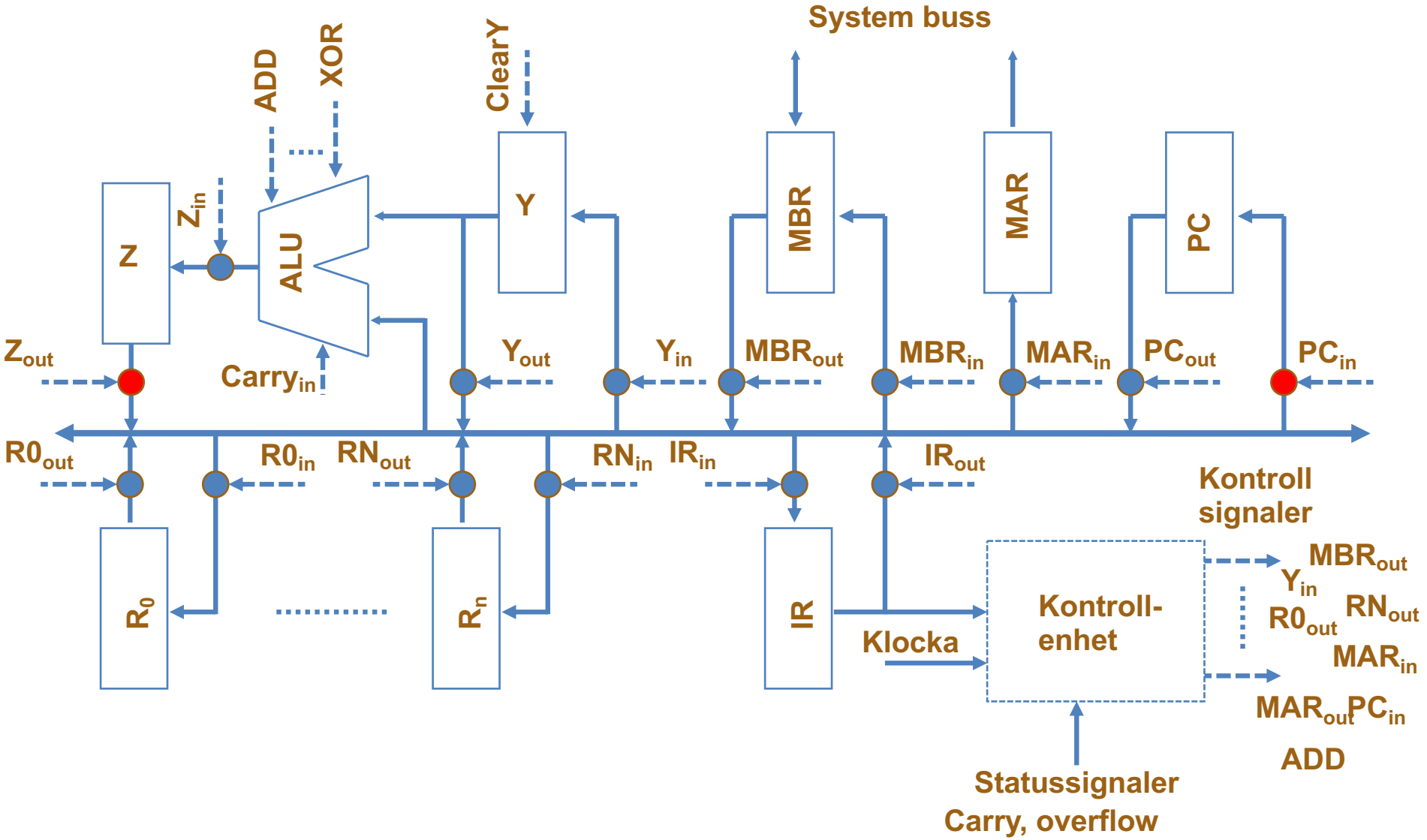
Kontrollenhet



Kontrollenhet

- Instruktion:
  - ADD R1, R3 // R1 <- R1 + R3
- Kontrollsteg:
  1. PC_{out}, MAR_{in}, Read, Clear Y, Carry-in, Add, Z_{in}
  2. Z_{out}, PC_{in}
  3. MBR_{out}, IR_{in}
  4. $R1_{out}$, Y_{in}
  5. $R3_{out}$, Add, Z_{in}
  6. Z_{out}, $R1_{in}$, End

Kontrollenhet

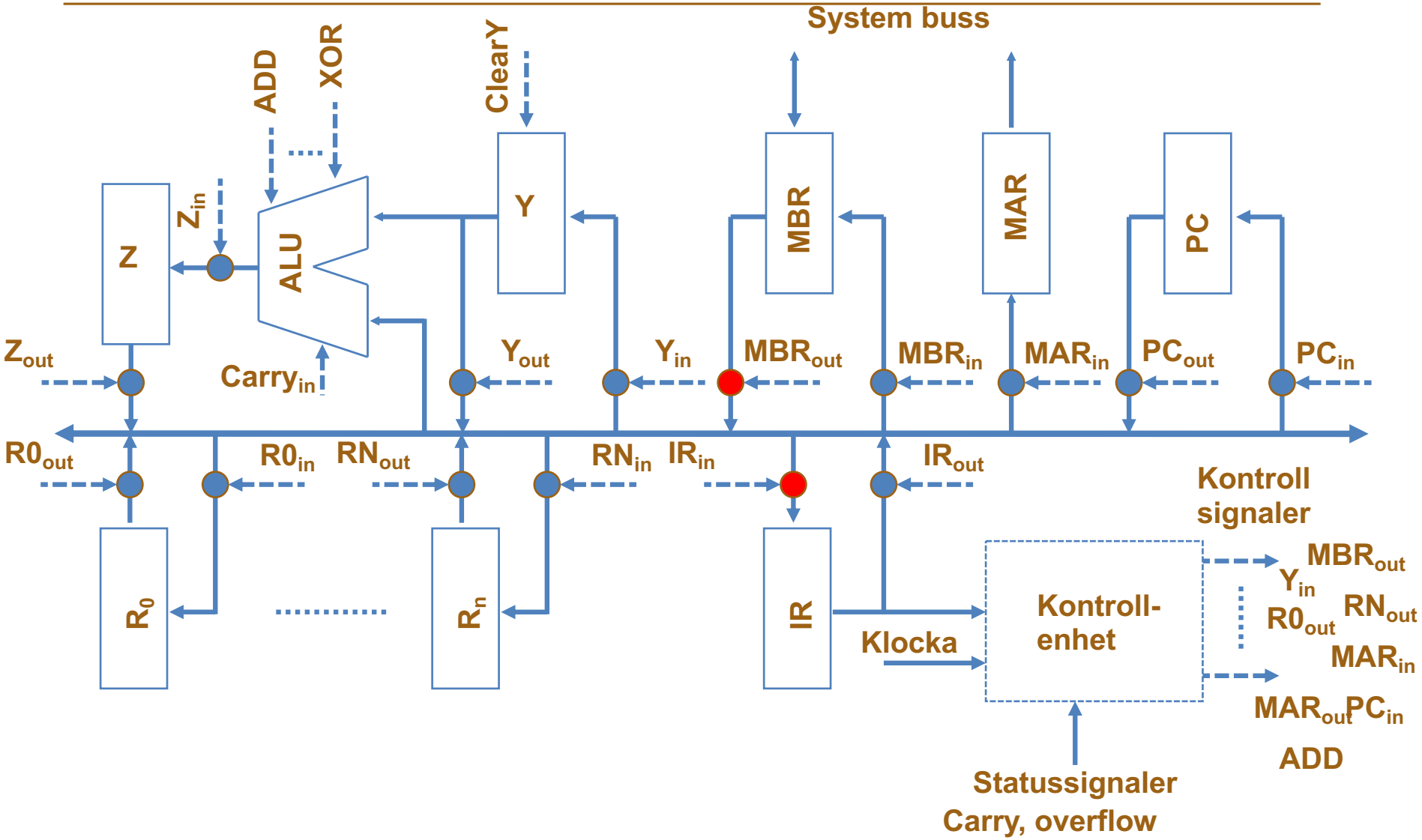


Kontrollenhet

- Instruktion:
  - ADD R1, R3 // R1 <- R1 + R3
- Kontrollsteg:
  1. PC_{out}, MAR_{in}, Read, Clear Y, Carry-in, Add, Z_{in}
  2. Z_{out}, PC_{in}
  3. $\underline{MBR_{out}, IR_{in}}$
  4. $R1_{out}$, Y_{in}
  5. $R3_{out}$, Add, Z_{in}
  6. Z_{out}, $R1_{in}$, End

LUNDS UNIVERSITET

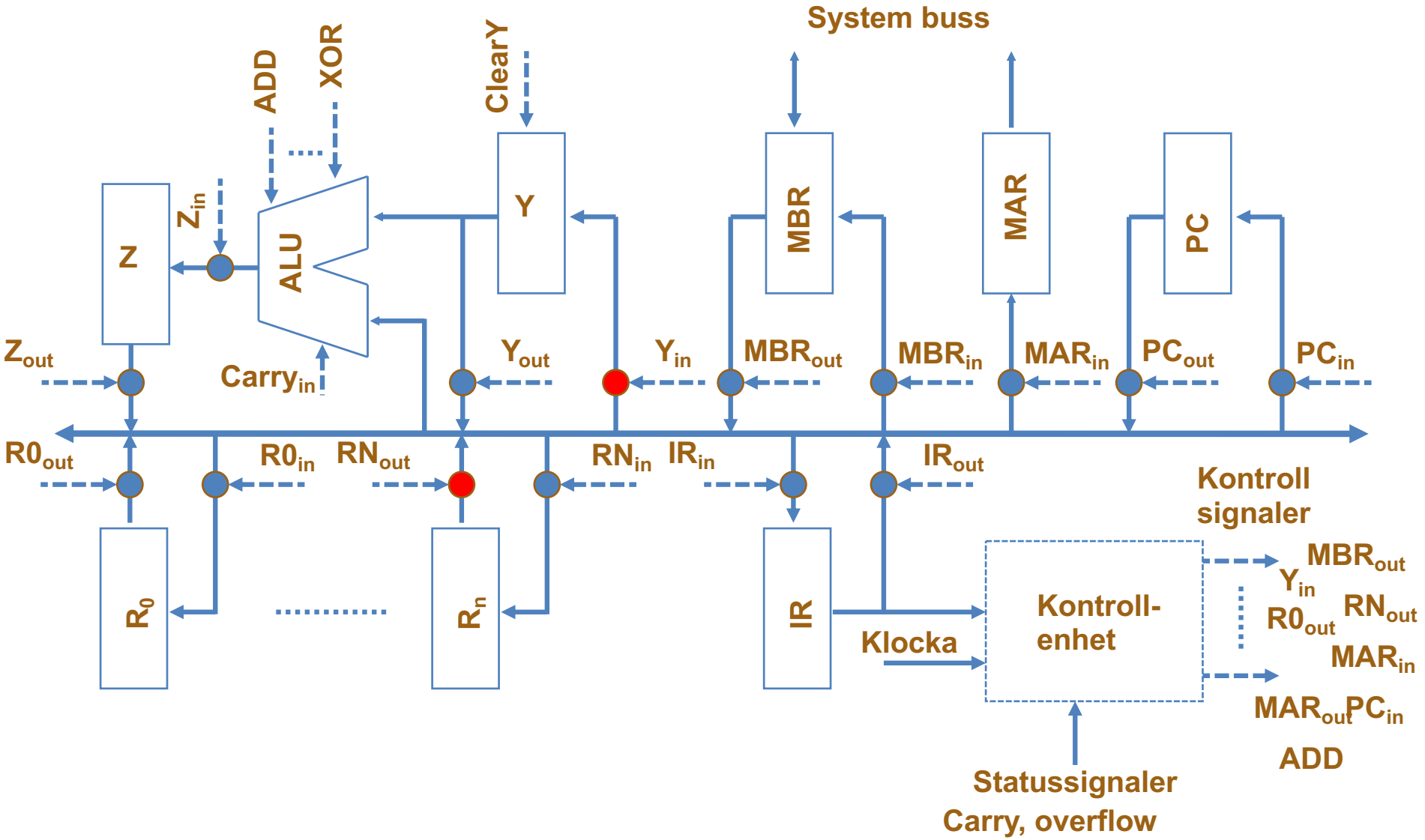
Kontrollenhet



Kontrollenhet

- Instruktion:
  - ADD R1, R3 // R1 <- R1 + R3
- Kontrollsteg:
  1. PC_{out}, MAR_{in}, Read, Clear Y, Carry-in, Add, Z_{in}
  2. Z_{out}, PC_{in}
  3. MBR_{out}, IR_{in}
  4. $R1_{out}$, Y_{in}
  5. $R3_{out}$, Add, Z_{in}
  6. Z_{out}, $R1_{in}$, End

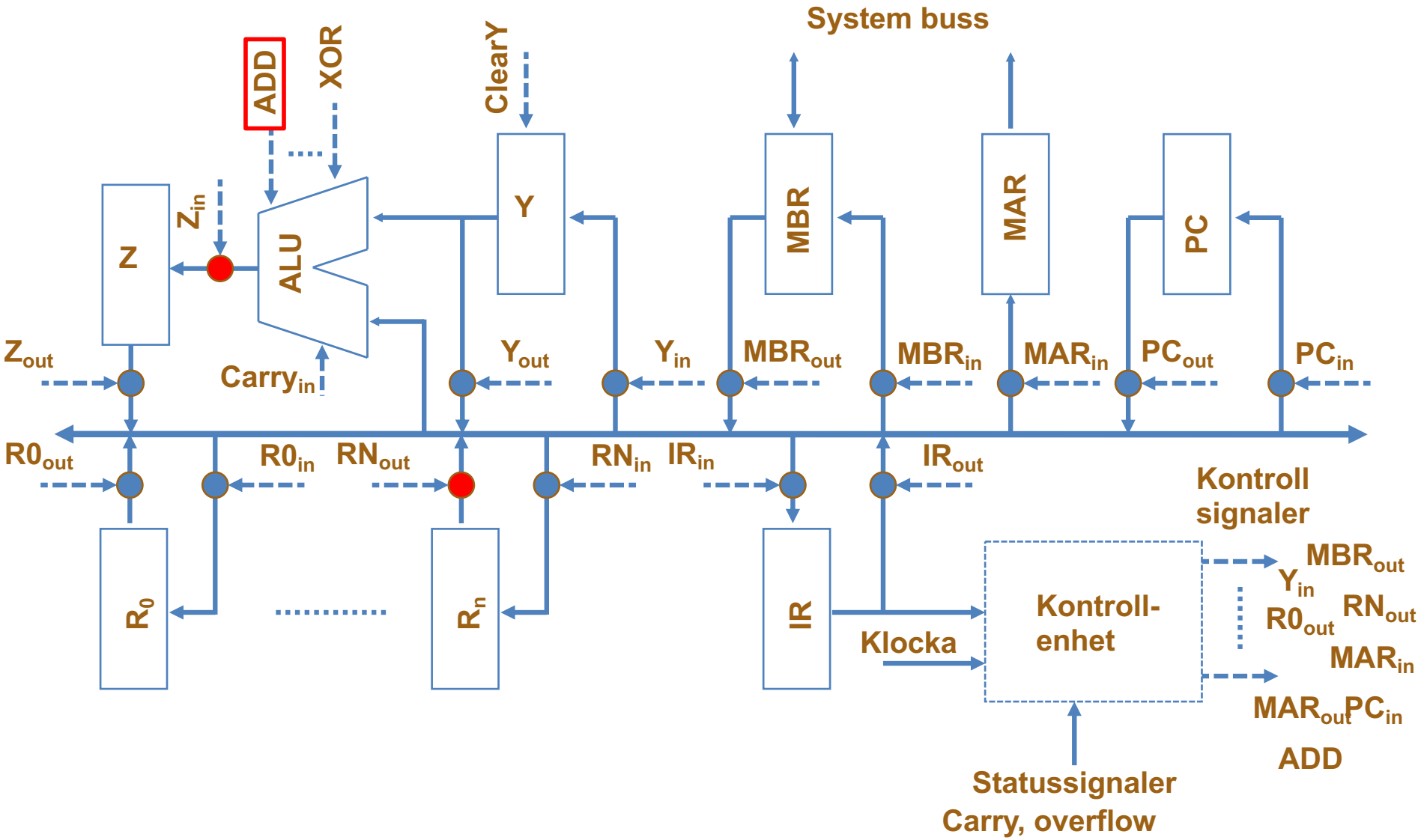
Kontrollenhet



Kontrollenhet

- Instruktion:
  - ADD R1, R3 // R1 <- R1 + R3
- Kontrollsteg:
  1. PC_{out}, MAR_{in}, Read, Clear Y, Carry-in, Add, Z_{in}
  2. Z_{out}, PC_{in}
  3. MBR_{out}, IR_{in}
  4. $R1_{out}$, Y_{in}
  5. $\underline{R3_{out}, \text{ Add}, Z_{in}}$
  6. Z_{out}, $R1_{in}$, End

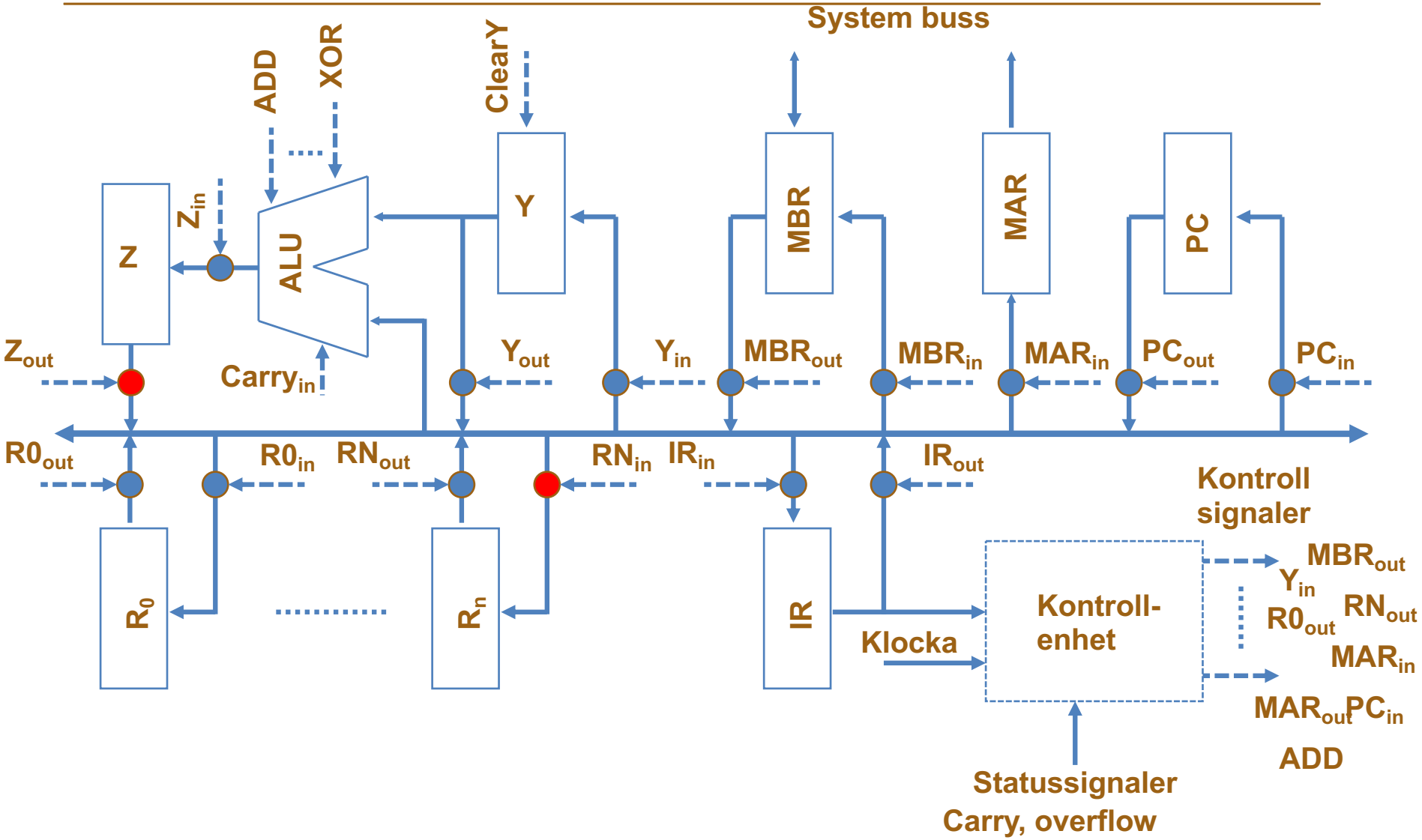
Kontrollenhet



Kontrollenhet

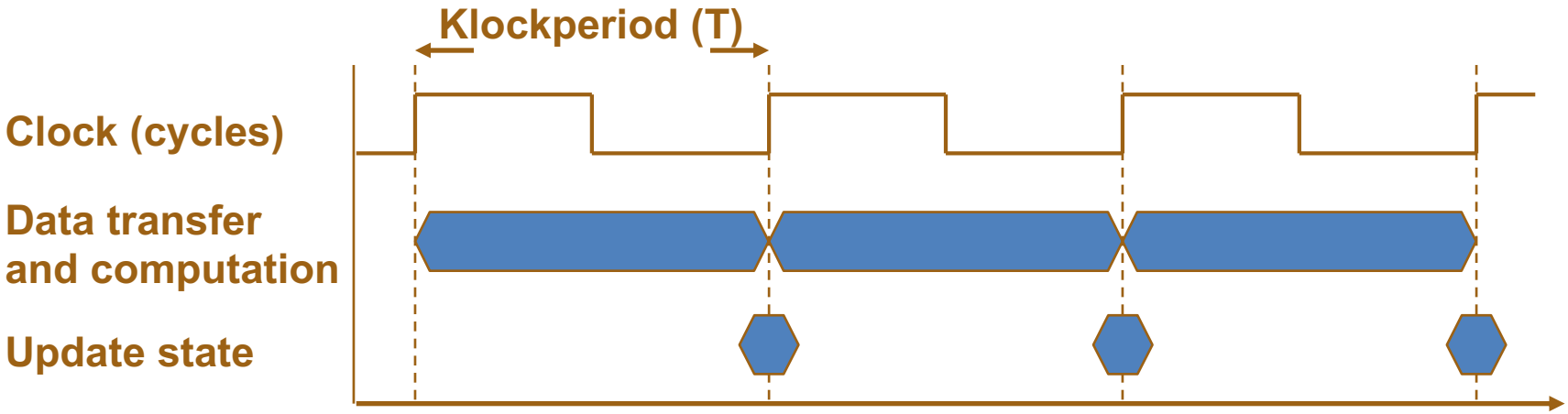
- Instruktion:
  - ADD R1, R3 // R1 <- R1 + R3
- Kontrollsteg:
  1. PC_{out}, MAR_{in}, Read, Clear Y, Carry-in, Add, Z_{in}
  2. Z_{out}, PC_{in}
  3. MBR_{out}, IR_{in}
  4. $R1_{out}$, Y_{in}
  5. $R3_{out}$, Add, Z_{in}
  6. Z_{out}, $R1_{in}$, End

Kontrollenhet



Klockning

- En klocka styr hastigheten på digital hårdvara

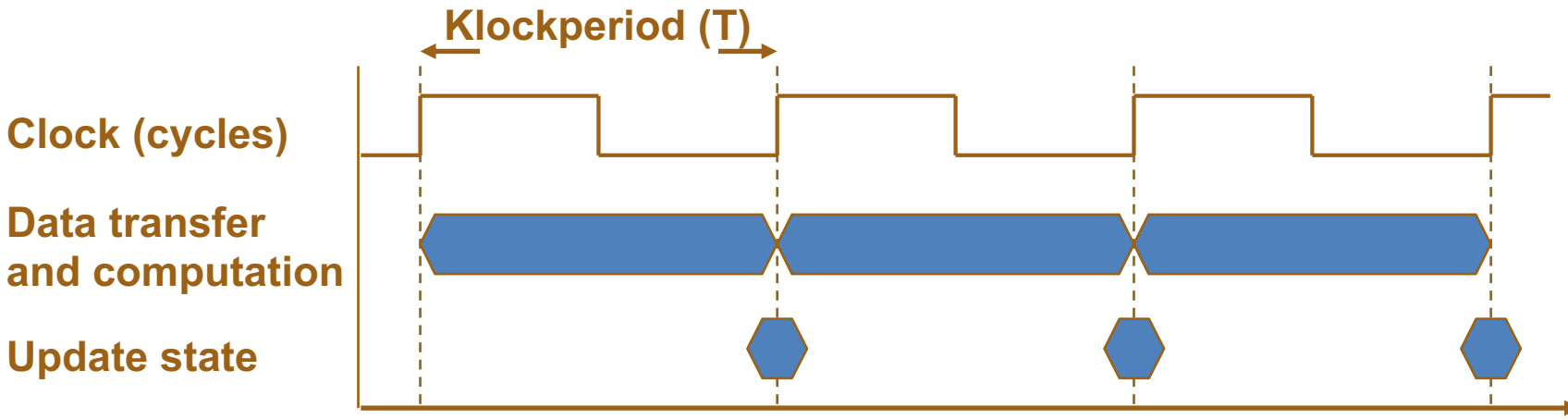


- Klockperiod: tid för en klockcykel (T)
  - e.g., 250ps = 0.25ns = 250×10^{-12}s
- Klockfrekvens (rate): cycles per second (f)
  - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz
- Samband: f=1/T

LUNDS UNIVERSITET

Klockning

År	Intel 4004	Apple A12	Skillnad
Lansering	1971	2018	47 år
Transistorer	2300	6.9 miljarder	3 miljoner
Klockfrekvens	740KHz	2.49GHz	3364
Klockperiod	1350000ps	400ps	3364
Teknologi	10µm	7 nm	700



LUNDS UNIVERSITET

Exekveringstid

- Antal klockcykler för att exekvera en maskininstruktion
  - Clocks per instruction (CPI)
- Tid för en klockcykel
  - Tid för en klockperiod (T)
  - Frekvens (f) är: f=1/T
- Antal maskininstruktioner
  - Instruction count (IC)
- Exekveringstid i klockcykler = CPI x T x IC

LUNDS UNIVERSITET

Exekveringstid

- Instruktion:
  - ADD R1, R3 // R1 <- R1 + R3
- Kontrollsteg = Klockcykler per instruktion (CPI)
  1. PC_{out}, MAR_{in}, Read, Clear Y, Carry-in, Add, Z_{in}
  2. Z_{out}, PC_{in}
  3. MBR_{out}, IR_{in}
  4. $R1_{out}$, Y_{in}
  5. $R3_{out}$, Add, Z_{in}
  6. Z_{out}, $R1_{in}$, End

Exekveringstid

- Prestanda kan ökas genom:
  - Öka klockfrekvensen (f) (minska T)
  - Minska antal instruktioner (IC)
  - Minska antal klockcykler per instruktion (CPI)

LUNDS UNIVERSITET

Exekveringstid

- MIPS (akronym av ”miljoner instruktioner per sekund” eller engelska ”million instructions per second”)

Processor	MIPS	År
Intel 8080 vid 2 MHz	0,640	1974
Motorola 68000 vid 8 MHz	1	1979
ARM 7500FE vid 40 MHz	35,9	1996
Intel 486DX vid 66 MHz	54	1992
Zilog eZ80 vid 50 MHz	80	1999
ARM10 vid 300 MHz	400	1998
IBM PowerPC 440GP vid 500 MHz	1000	2002
Athlon 64 vid 2,8 GHz	8400	2005

LUNDS UNIVERSITET

Prestanda

- Algoritm
  - Bestämmer vilka och hur många *operationer* som ska utföras
- Programmeringsspråk, kompilator, arkitektur
  - Bestämmer hur många *maskininstruktioner* som ska utföras per *operation*
- Processor och minnessystem
  - Bestämmer hur snabbt instruktioner exekveras
- I/O (Input/Output) och operativsystem
  - Bestämmer hur snabbt I/O operationer ska exekveras

LUNDS UNIVERSITET

LUNDS
UNIVERSITET