LUNDS UNIVERSITET

Digitala System: Datorteknik

ERIK LARSSON

Digitala system: Översikt

- Antal högskolepoäng: 4,5. Prestationsbedömning: Skriftlig tentamen. **Tid: 120 timmar**
- Antal högskolepoäng: 3. Prestationsbedömning: Godkända laborationsuppgifter. Delmomentet omfattar: Laborationer del 1. **Tid: 80 timmar**
- Antal högskolepoäng: 7,5. Prestationsbedömning: Godkända laborationsuppgifter. Delmomentet omfattar: Laborationer del 2. **Tid: 200 timmar**

LUNDS UNIVERSITET

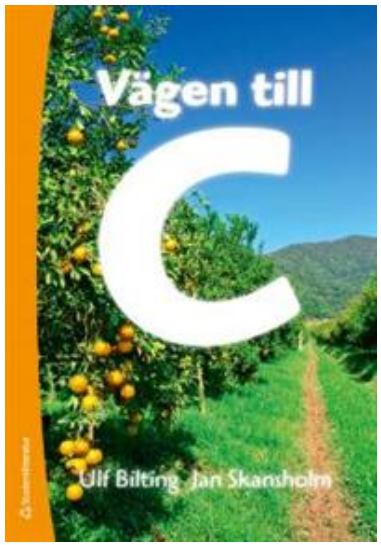
Digitala System: Datorteknik

- Datormodellen: Datorns delar och funktion. CPU:n på registernivå
- Assemblyprogrammering: Data- och instruktionsformat. Adresseringsmetoder. Instruktionsrepertoar. ”Timing” och exekveringstid. Stack och subrutiner
- Programutveckling i C: Editering. Kompilering. Länkning. Testning med hjälp av högnivådebugger.

LUNDS UNIVERSITET

Litteratur

- Föreläsningsanteckningar
- Vägen till C, Jan Skansholm, Ulf Bilting
  Förlag: Studentlitteratur
  ISBN10: 9144076061
  ISBN13: 9789144076065



LUNDS UNIVERSITET

Föreläsningar

- FÖ1: Processorn
- FÖ2: C och assembly
- FÖ3: Minnen
- FÖ4: Pipeline
- FÖ5: OS

LUNDS UNIVERSITET

Blooms taxonomi - lärande

- Blooms taxonomi, lanserades av Benjamin Bloom, beskriver olika steg i lärandet:

1. Kunskap - Att kunna ta till sig fakta - memorera.
2. Förståelse - Att kunna förstå fakta som memorerats.
3. Tillämpning - Att kunna utföra något med de fakta och med hjälp av den kunskap som man fått.
4. Analys - Att kunna se samband mellan olika fakta.
5. Syntes - Att kunna dra egna slutsatser.
6. Utvärdering - Att kunna producera, planera och generera något nytt.

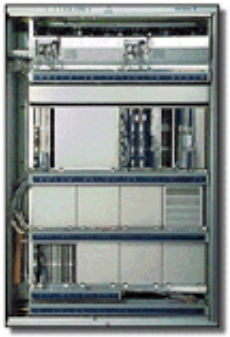
LUNDS UNIVERSITET

LUNDS UNIVERSITET

Digitala System: Datorteknik

ERIK LARSSON

Elektronik



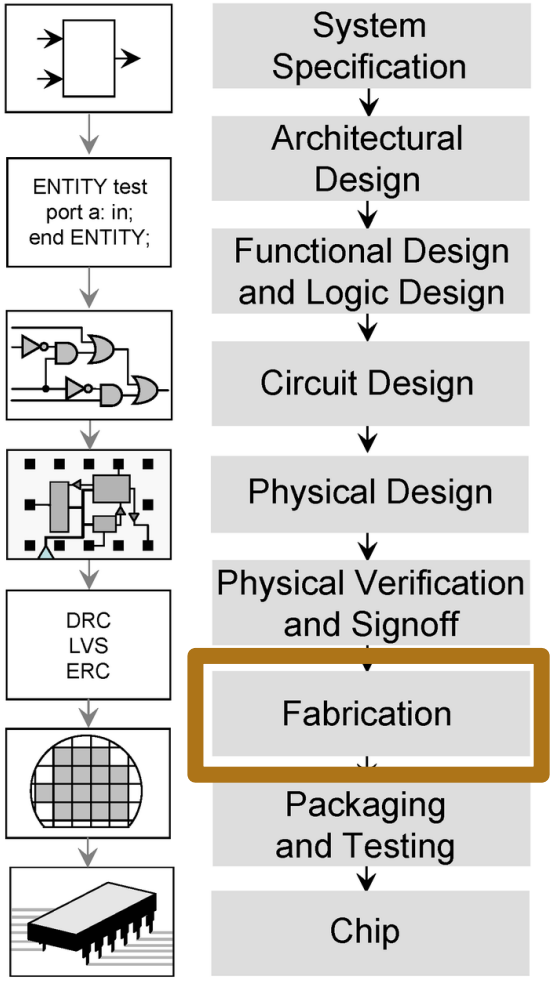
LUNDS UNIVERSITET

Elektronik

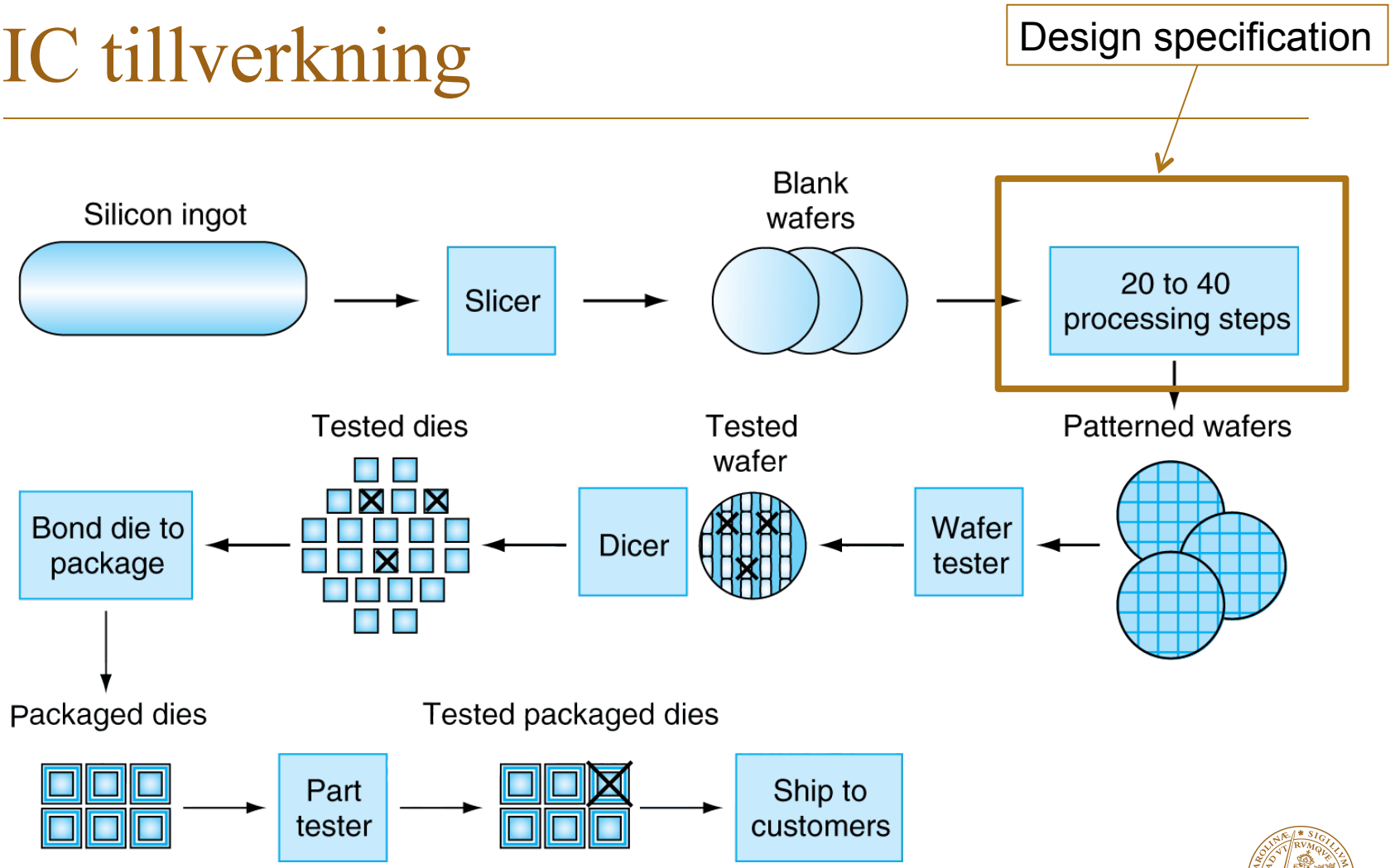
LUNDS UNIVERSITET

Design steg för IC

- Feasibility study and die size estimate
- Function analysis
- System Level Design
- Analogue Design, Simulation & Layout
- Digital Design, Simulation & Synthesis
- System Simulation & Verification
- Design For Test and Automatic test pattern generation
- Design for manufacturability (IC)
- Tape-in
- Mask data preparation
- Tape-out
- Wafer fabrication
- Die test
- Packaging
- Post silicon validation and integration
- Device characterization
- Tweak (if necessary)
- Datasheet generation
- Ramp up
- Production
- Yield Analysis / Warranty Analysis Reliability (semiconductor)
- Failure analysis on any returns
- Plan for next generation chip using production information if possible

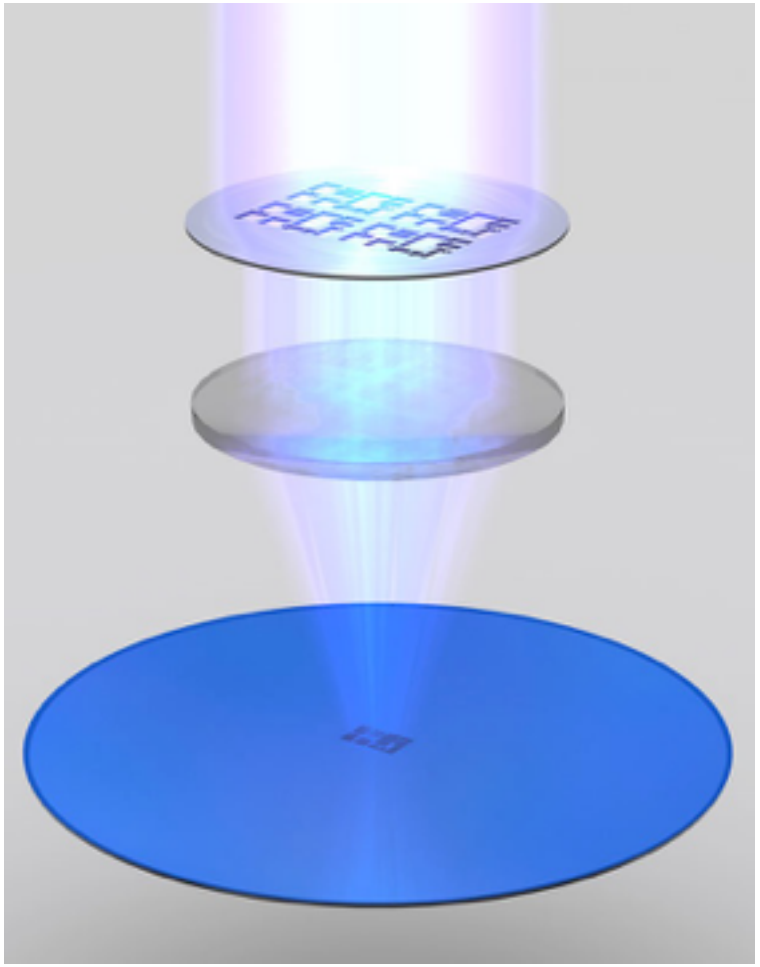


IC tillverkning



LUNDS UNIVERSITET

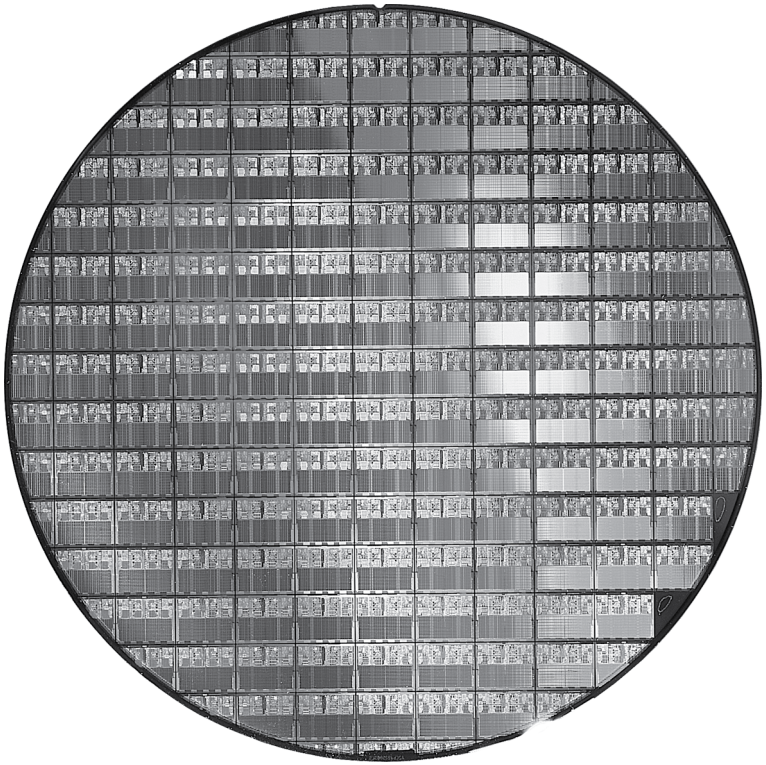
Kostnad

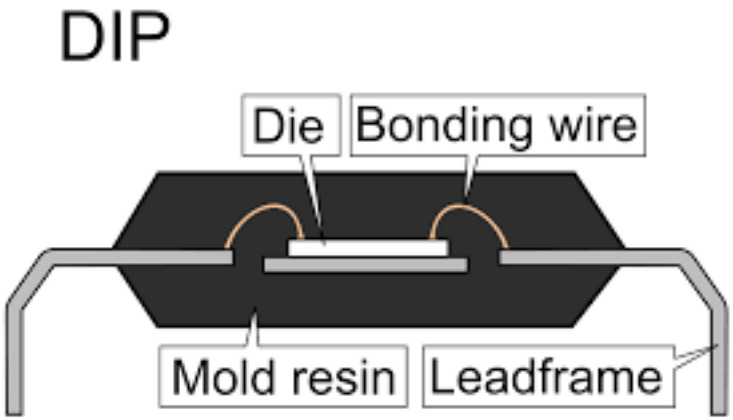


- Random defects and systematic defects
- A photomask can range from **$1,000 to $100,000** for a single mask.
- As many as **30 masks** may be required to form a complete mask set.
- A few “re-spins” increase cost and delay time-to-market

LUNDS UNIVERSITET

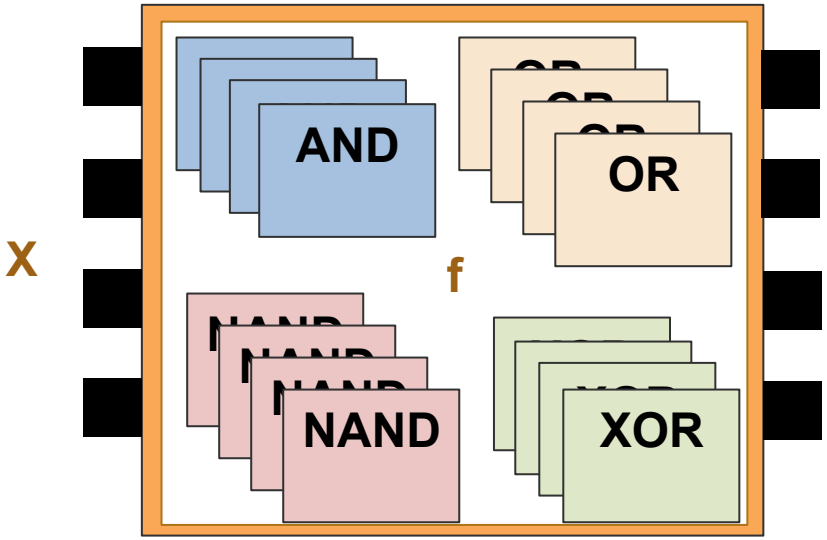
IC, die och wafer

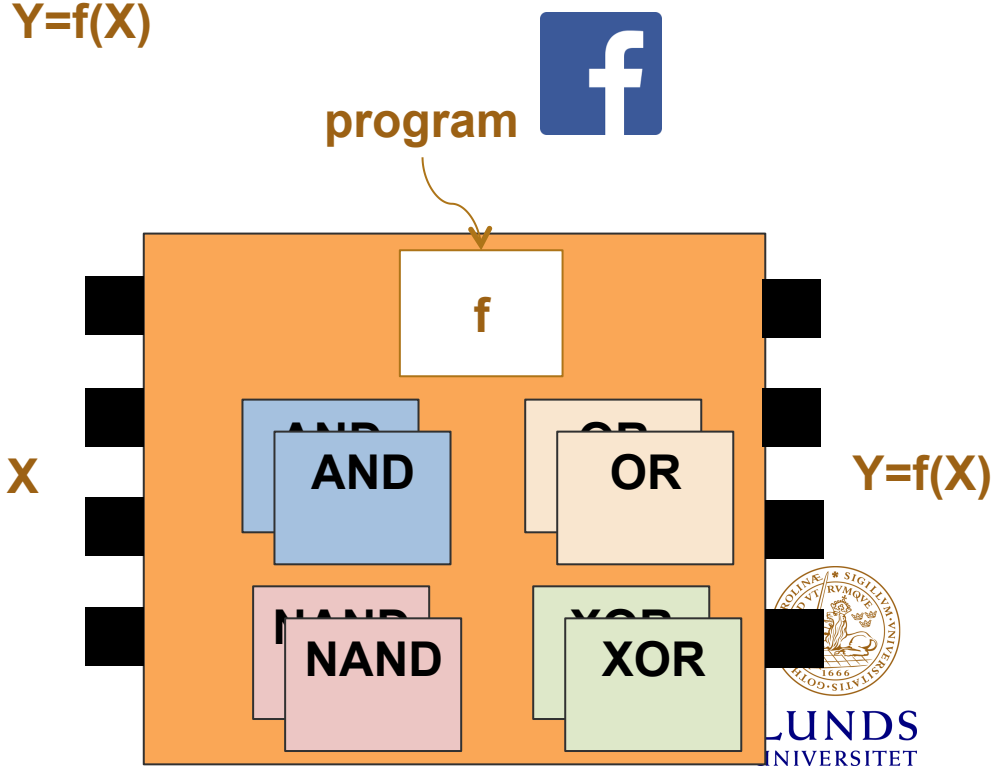




LUNDS UNIVERSITET

IC



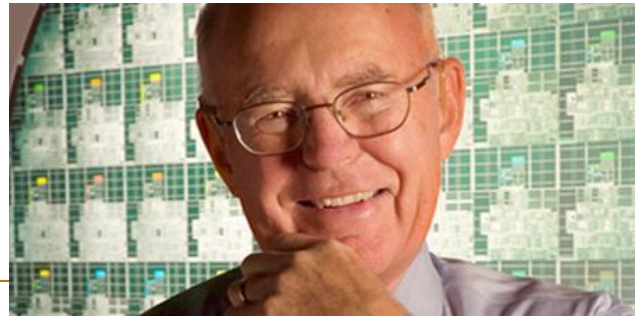


Datorer

- "I think there is a world market for maybe five computers."
  -- Thomas Watson, chairman of IBM, 1943
- "Computers in the future may weigh no more than 1.5 tons."
  -- Popular Mechanics, forecasting the relentless march of science, 1949
- "There is no reason anyone would want a computer in their home."
  -- Ken Olson, president, chairman and founder of Digital Equipment Corp., 1977

LUNDS UNIVERSITET

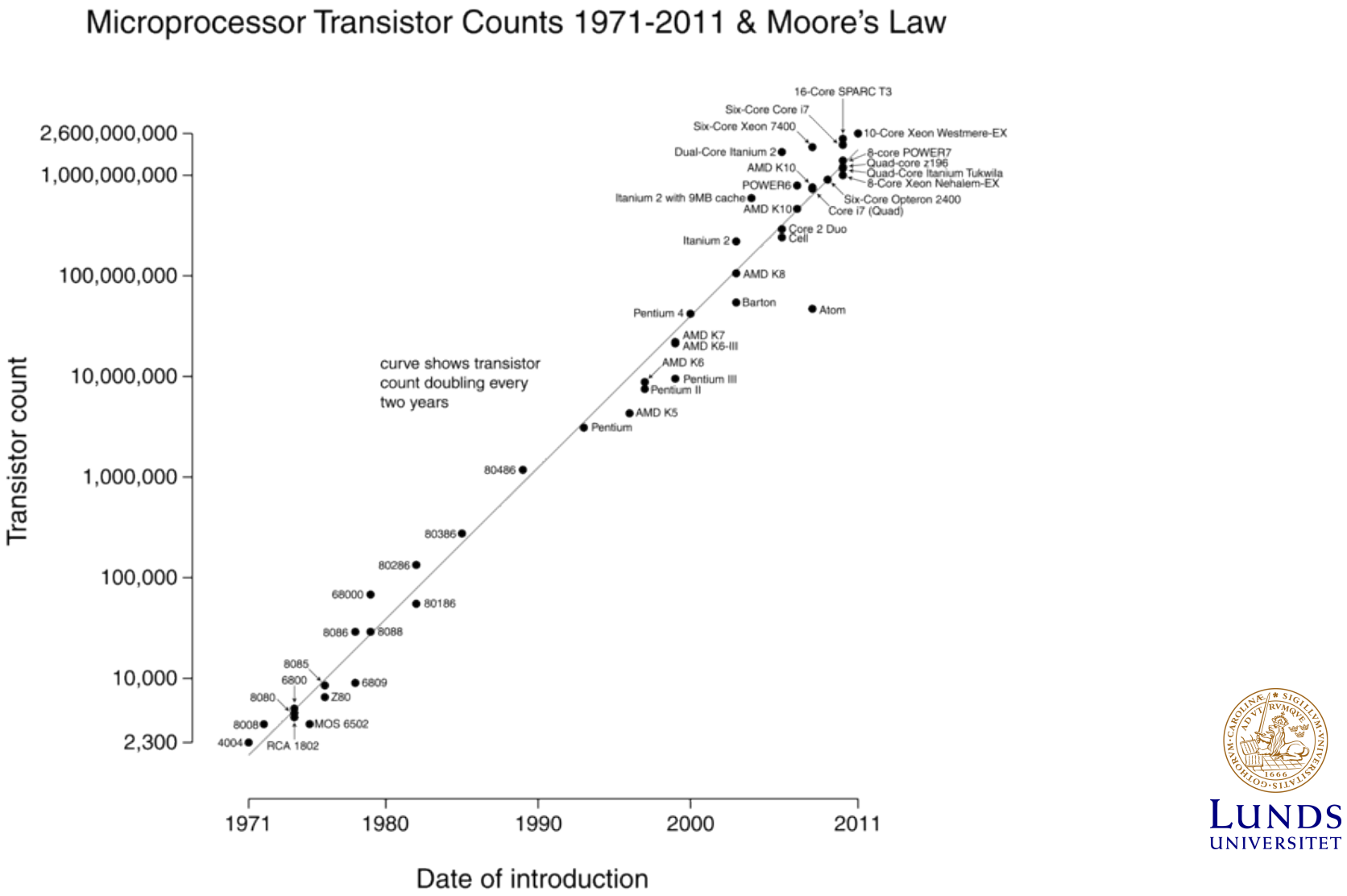
Intel



- Gordon E. Moore, co-founder of Intel (Integrated Electronics) (tillsammans med Robert Noyce)
- Första tanken på namn var: Moore & Noyce Electronics
- Moores lag (1965): Antalet transistorer på ett chip växer exponentiellt; det dubblas var 24:e månad.

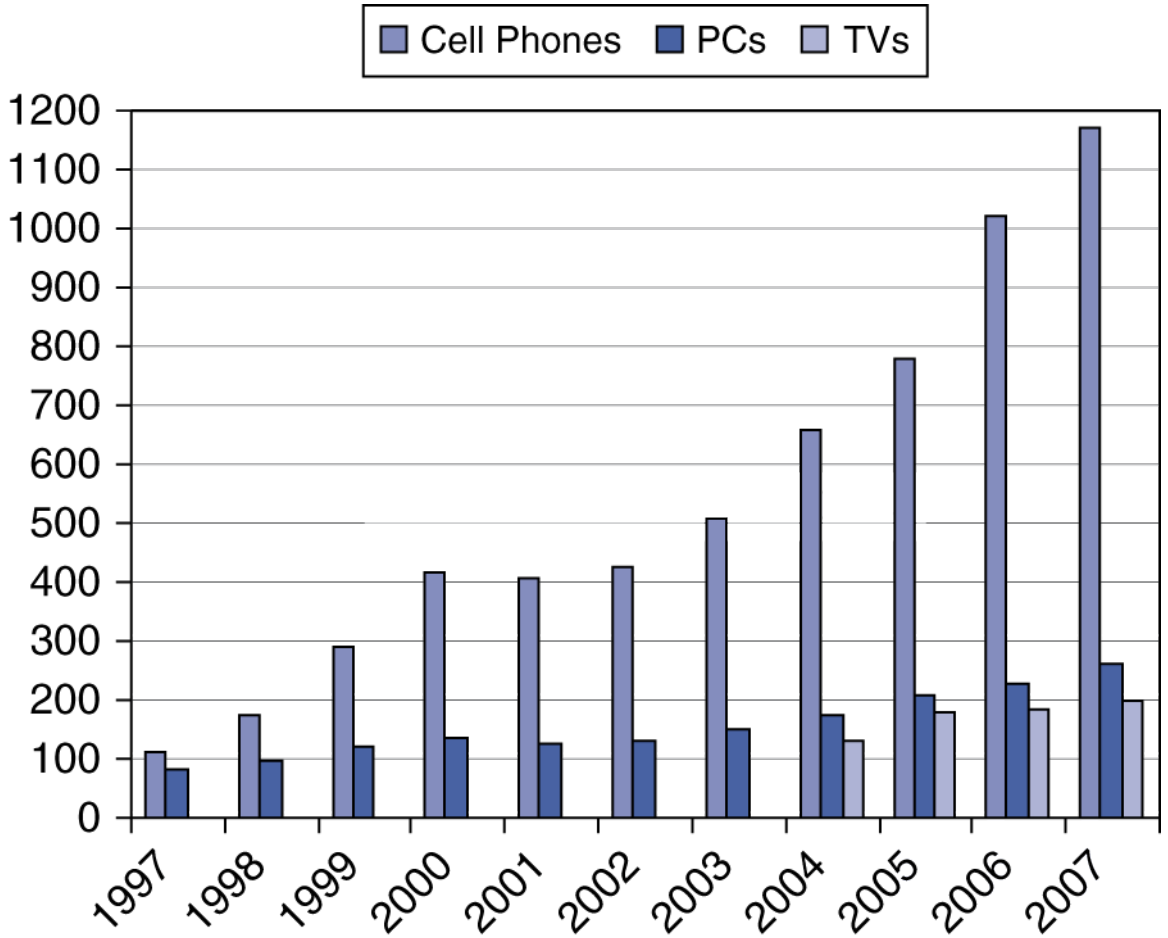
LUNDS UNIVERSITET

Trender


Microprocessor Transistor Counts 1971-2011 & Moore's Law
Transistor count
2,600,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
2,300
curve shows transistor count doubling every two years
16-Core SPARC T3
Six-Core Core i7
Six-Core Xeon 7400
10-Core Xeon Westmere-EX
Dual-Core Itanium 2
8-core POWER7
Quad-core z196
AMD K10
Quad-Core Itanium Tukwila
POWER6
8-Core Xeon Nehalem-EX
Itanium 2 with 9MB cache
Six-Core Opteron 2400
AMD K10
Core i7 (Quad)
Core 2 Duo
Itanium 2
Cell
AMD K8
Barton
Atom
Pentium 4
AMD K7
AMD K6-III
AMD K6
Pentium III
Pentium II
AMD K5
Pentium
80486
80386
80286
68000
80186
8086
8088
8085
6800
6809
8080
Z80
8008
MOS 6502
4004
RCA 1802
1971
1980
1990
2000
2011
Date of introduction

LUNDS
UNIVERSITET

Marknad för processorer



LUNDS UNIVERSITET

Datorer

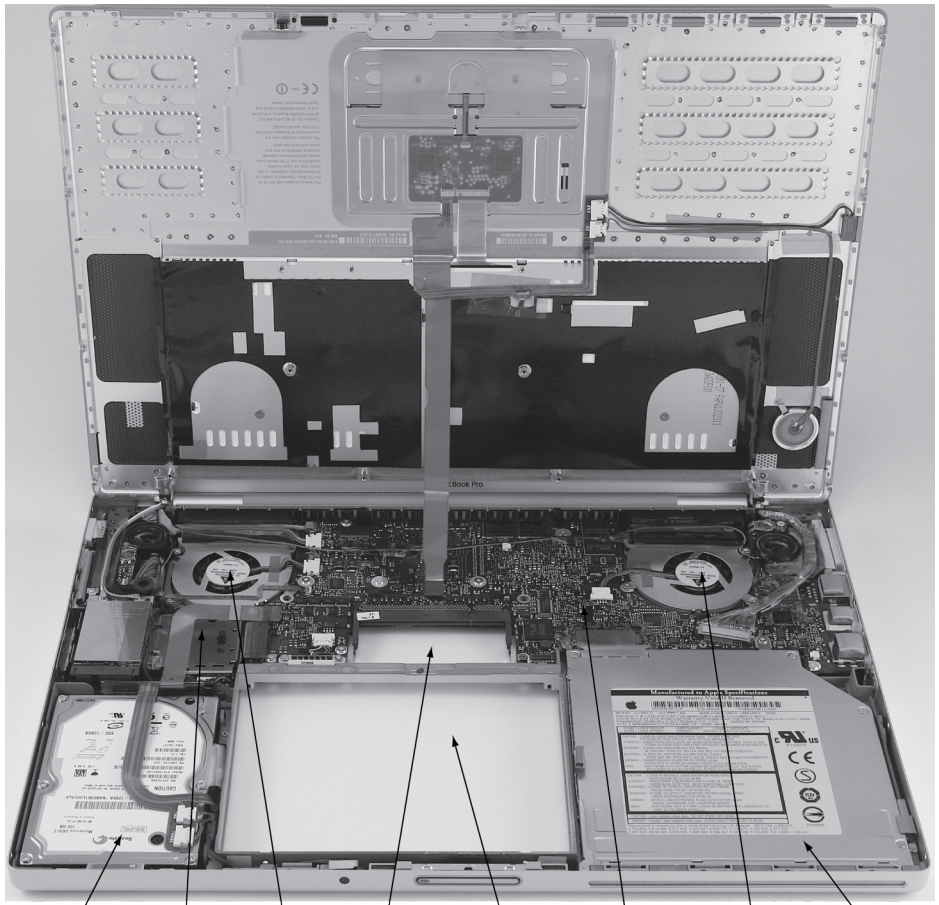
- Desktop datorer
  - Generella, olika program beroende på användare
  - Viktigt med kostnad/prestanda avvägning
- Server computers
  - Nätverksbaserade
  - Hög prestanda, hög pålitlighet
  - Små servers till mycket stora system
- Inbyggda system (Embedded computers)
  - Datorn en del av systemet
  - Ofta hårda krav på effektförbrukning, prestanda/ kostnad

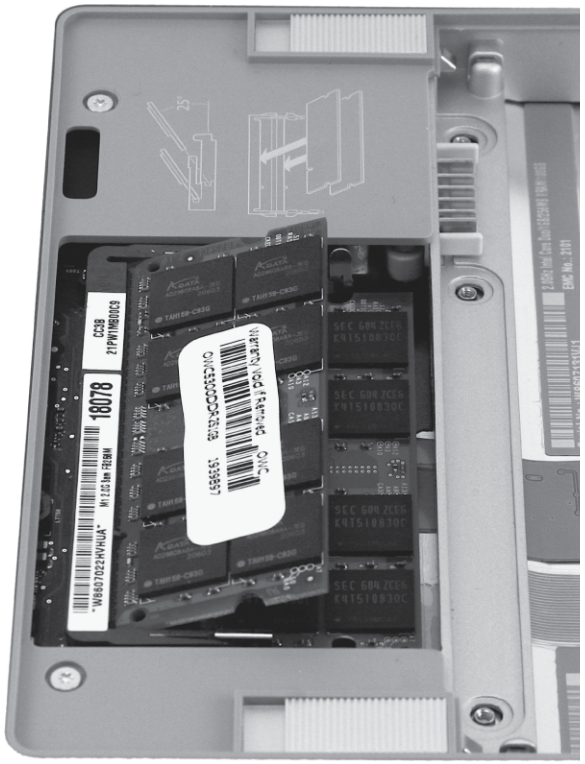
LUNDS UNIVERSITET

Datorns delar

LUNDS UNIVERSITET

Datorns insida

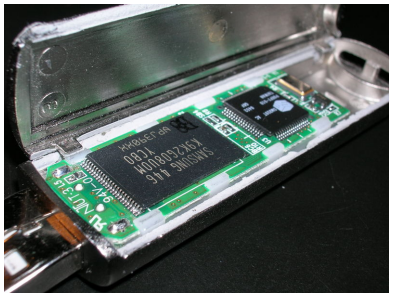




LUNDS UNIVERSITET

Lagring av data och instruktioner

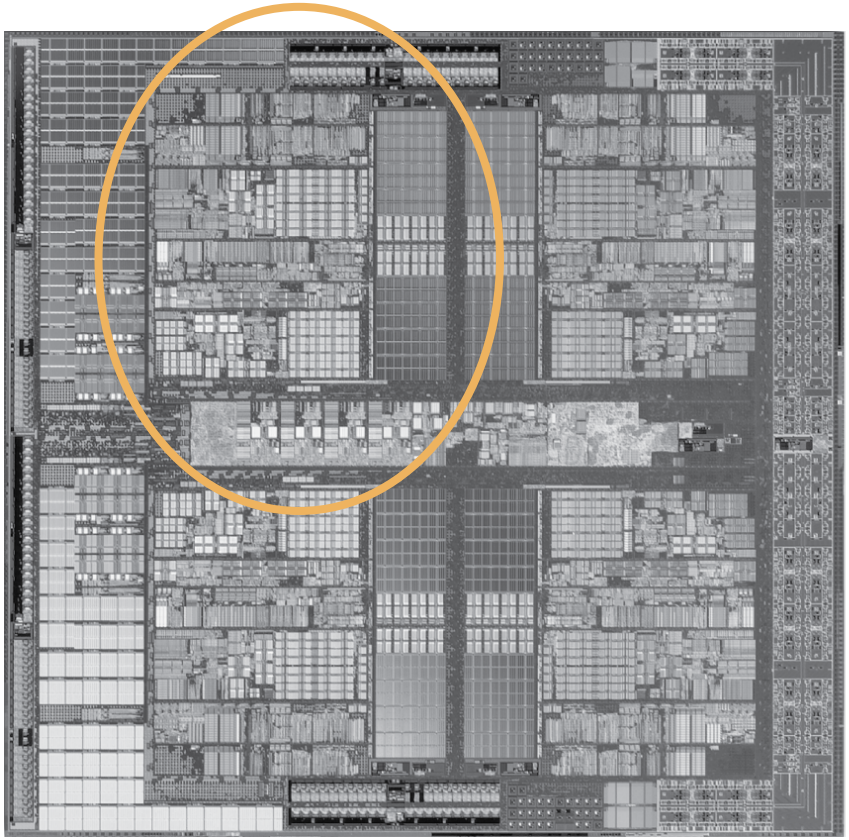
- Volatile main memory
  - Förlorar information (instruktioner och data) när strömmen stängs av
- Non-volatile secondary memory
  - Magnetic disk (hårddisk)
  - Flash memory
  - Optical disk (CDROM, DVD)

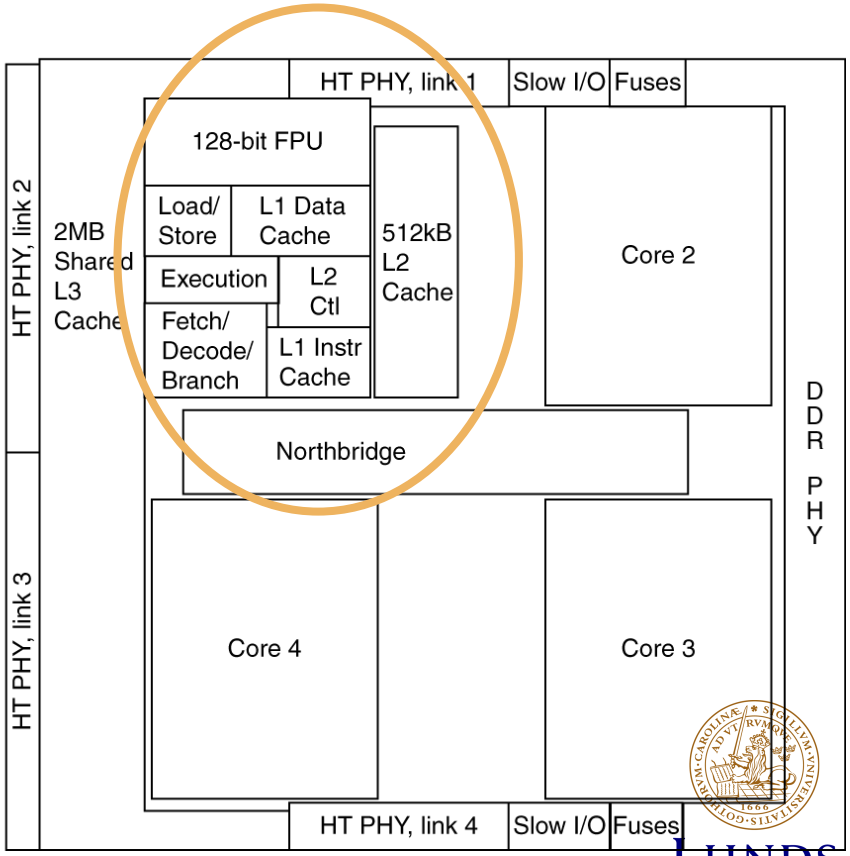


LUNDS UNIVERSITET

Processorn

- AMD Barcelona: 4 processor cores





LUNDS UNIVERSITET

Dator

- John von Neumann arkitektur
  - Gemensamt minne för instruktioner och data (vad som är vad bestäms av kontext)
  - Aritmetisk enhet för logiska och aritmetiska operationer
  - Kontrollenhet
- Instruktioner exekveras sekventiellt

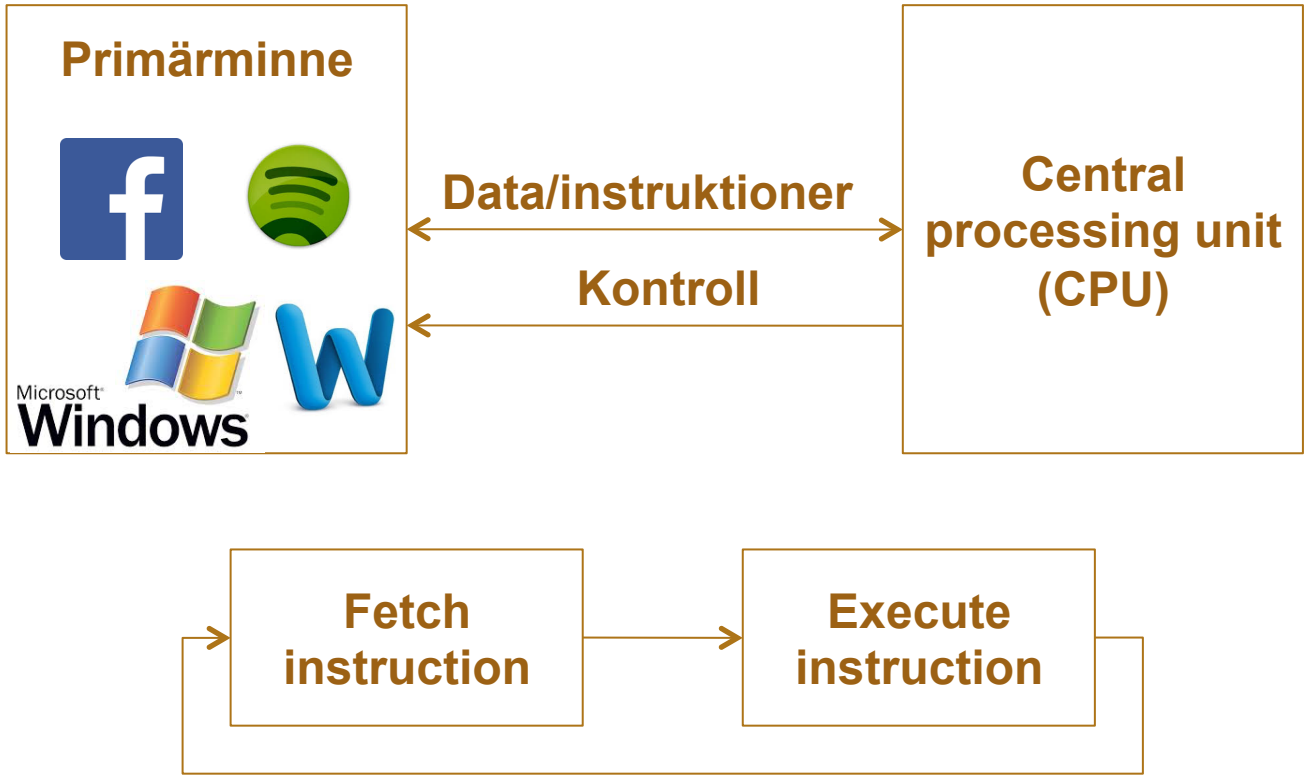
- John von Neumann (1903-1957)

Dator

- En dator är en maskin som behandlar data automatiskt under kotroll av en lista av instruktioner, så kallat program.
- Minnet är adresserbart (oberoende av vad som lagras)
- Instruktioner exekveras i ordning, så vida ordningen explicit ändras
- Programmet bestämmer vilket problem som ska lösas

LUNDS UNIVERSITET

Dator



LUNDS UNIVERSITET

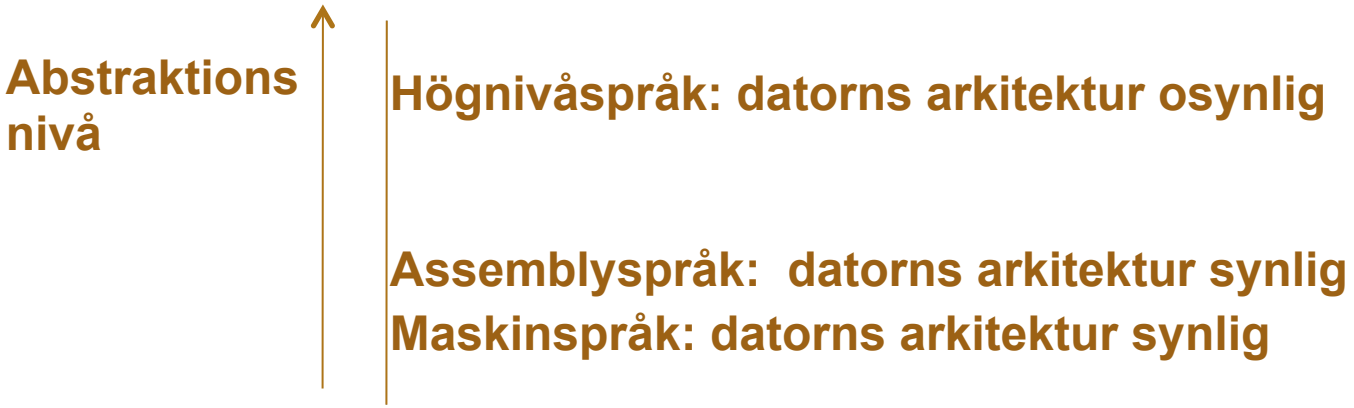
Användarprogram, OS och hårdvara

- 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


LUNDS UNIVERSITET

Program

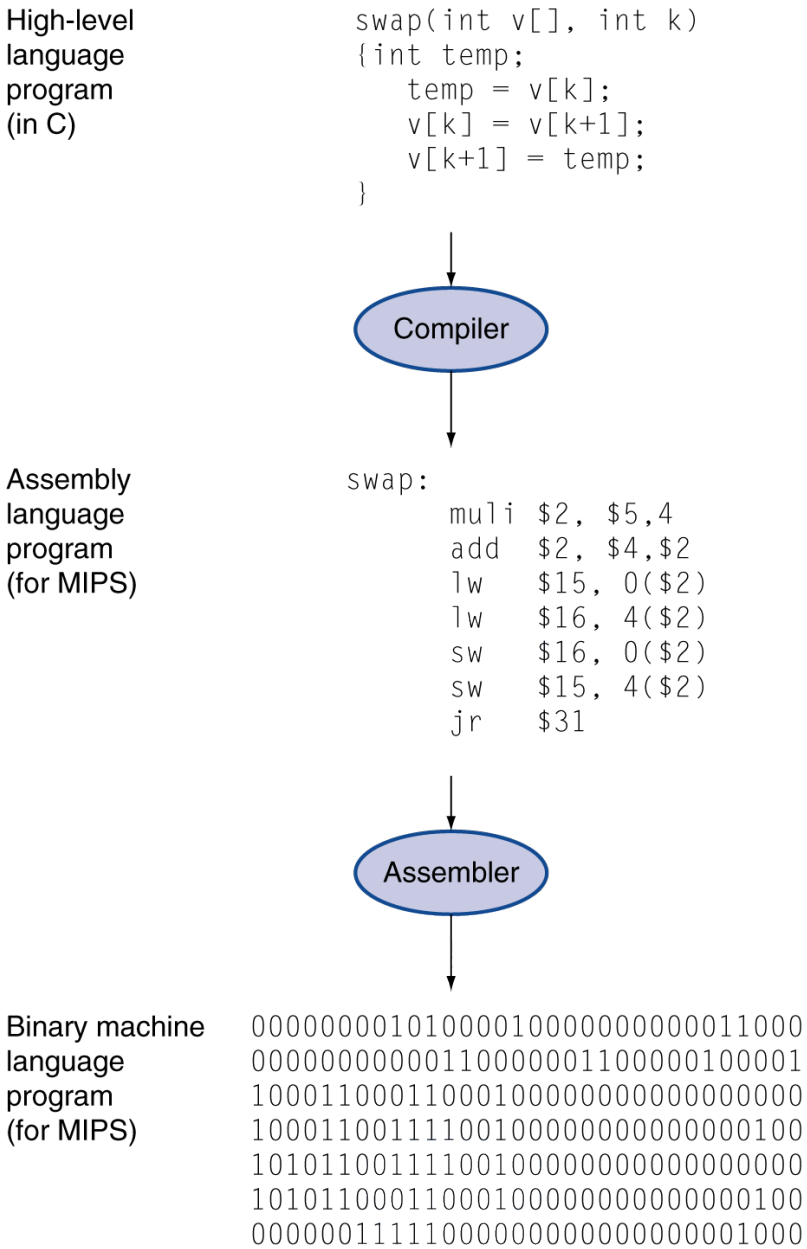
- För att bestämma vad som görs – (f(X)) – behövs program, programmeringsspråk.
- Programmeringsspråk:
  - Högnivåspråk (C, C++)
  - Assemblyspråk (t ex ADD R1, R2)
  - Maskinspråk (t ex 001101….101)



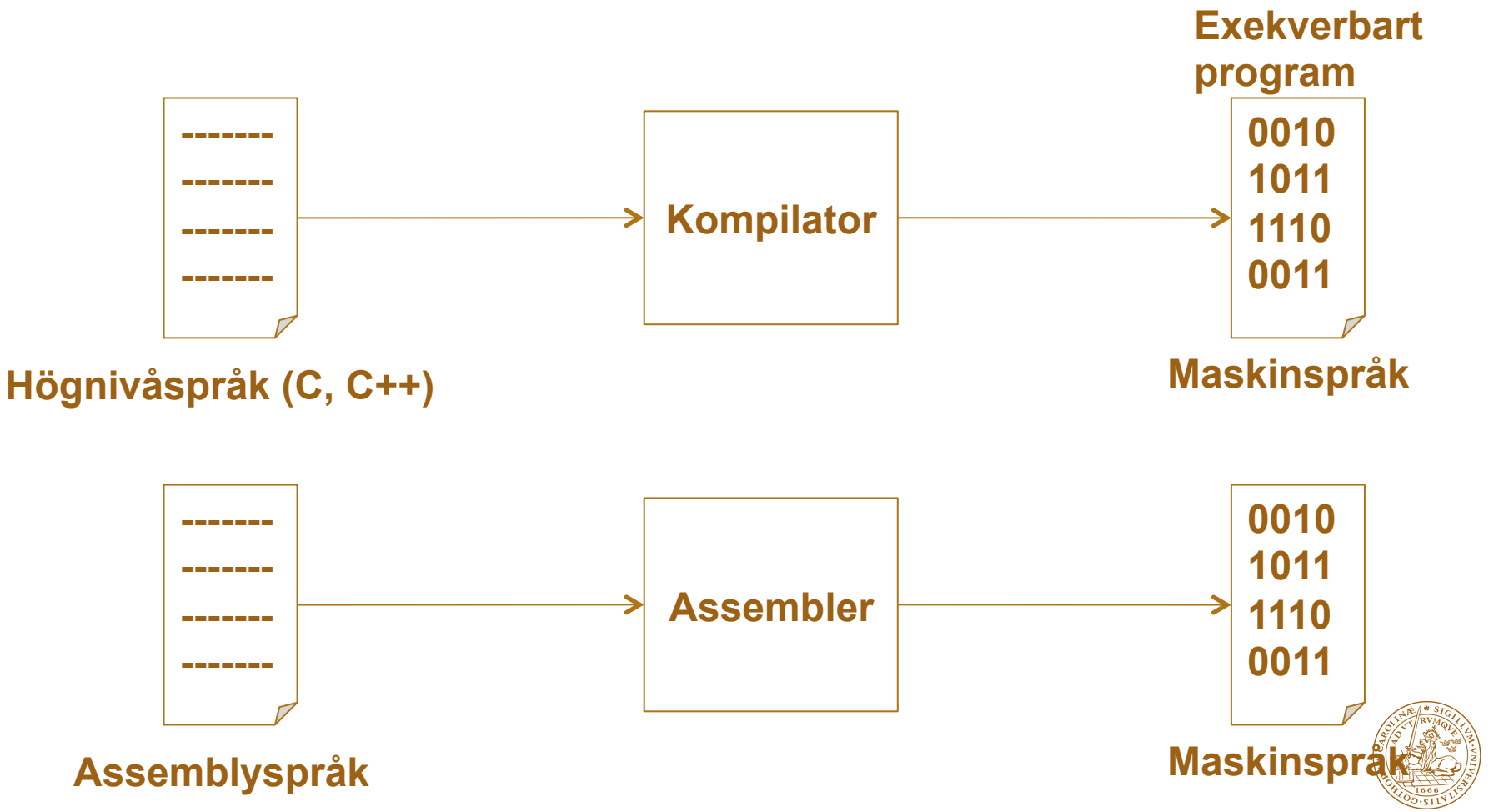
LUNDS UNIVERSITET

Program

- Abstraktionsnivå:
  - Högnivåspråk
  - Assemblyspråk
  - Maskinspråk

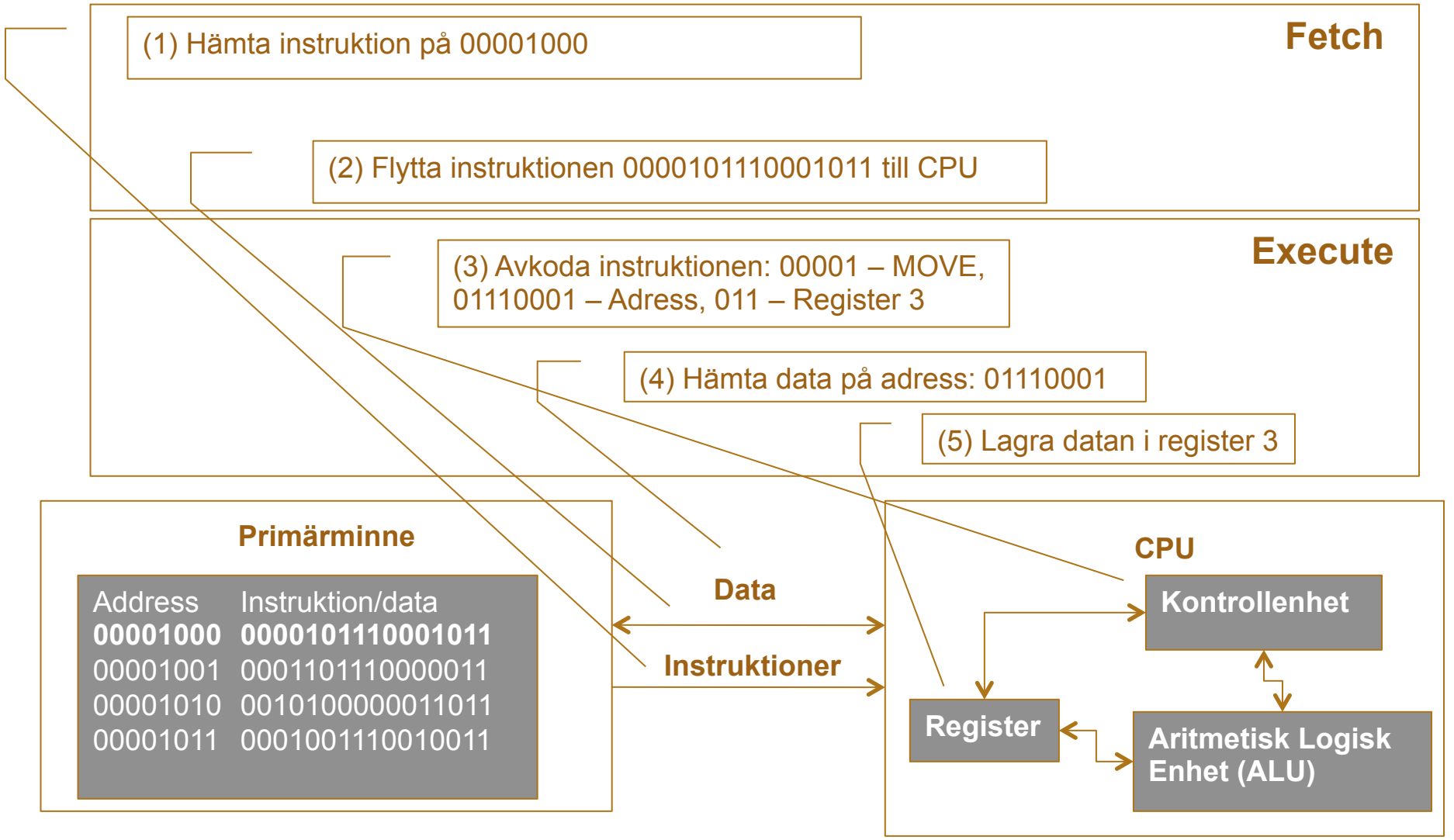


Program

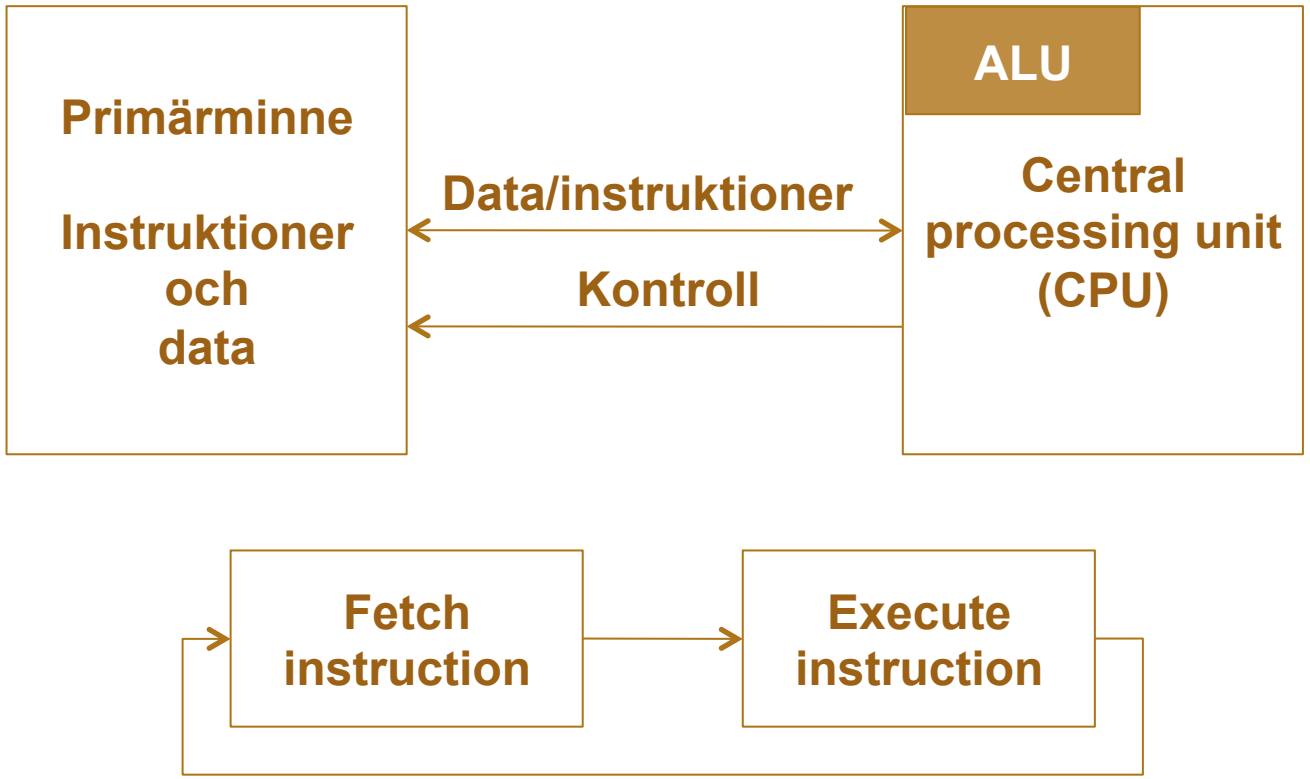


LUNDS UNIVERSITET

Exekvera en instruktion



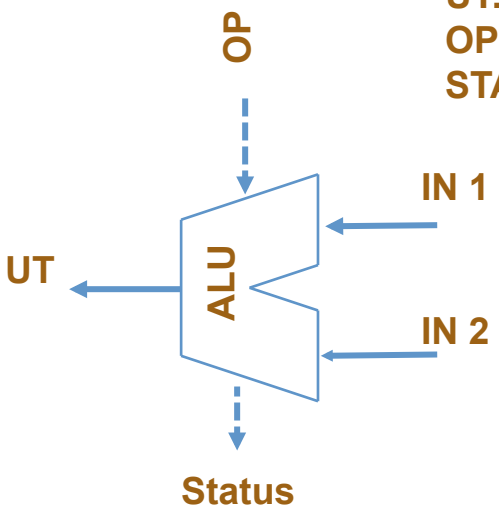
ALU – arithmetic logic unit



LUNDS UNIVERSITET

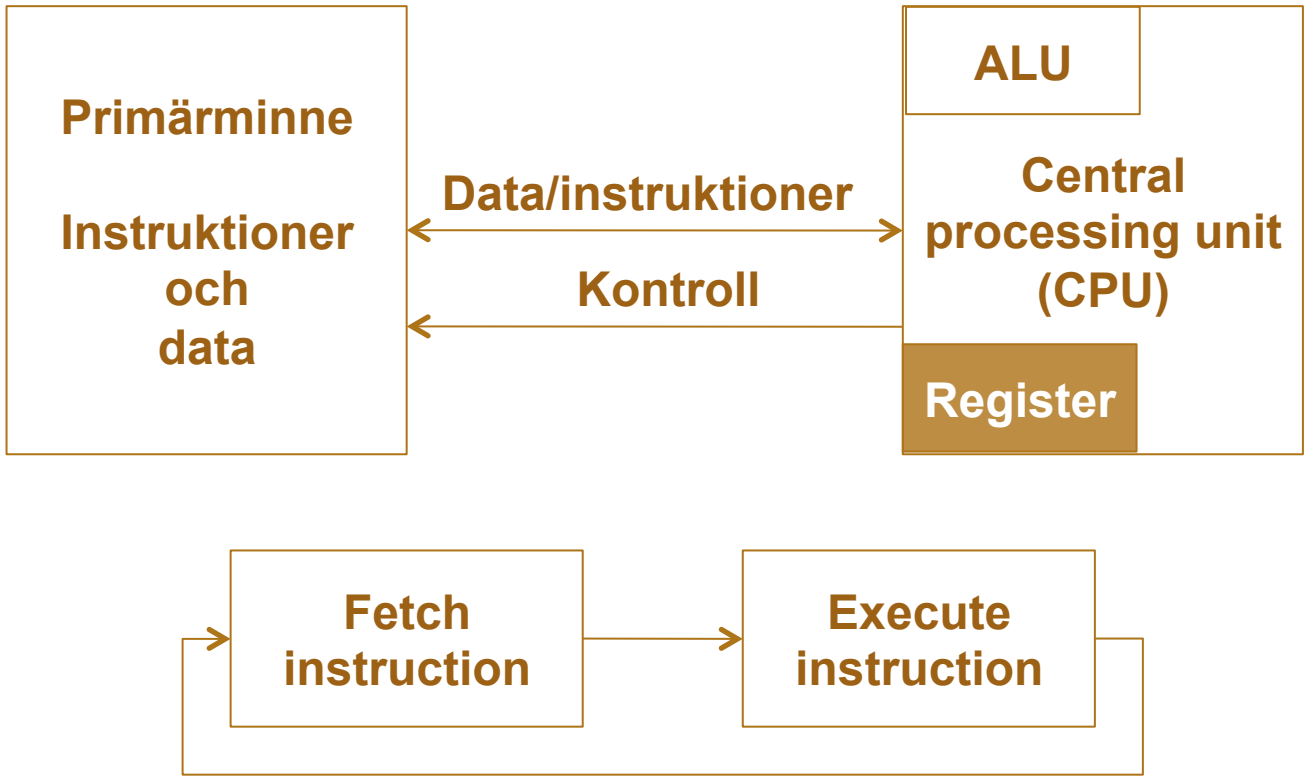
ALU – arithmetic logic unit

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



LUNDS UNIVERSITET

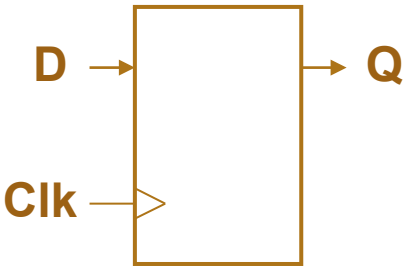
Register

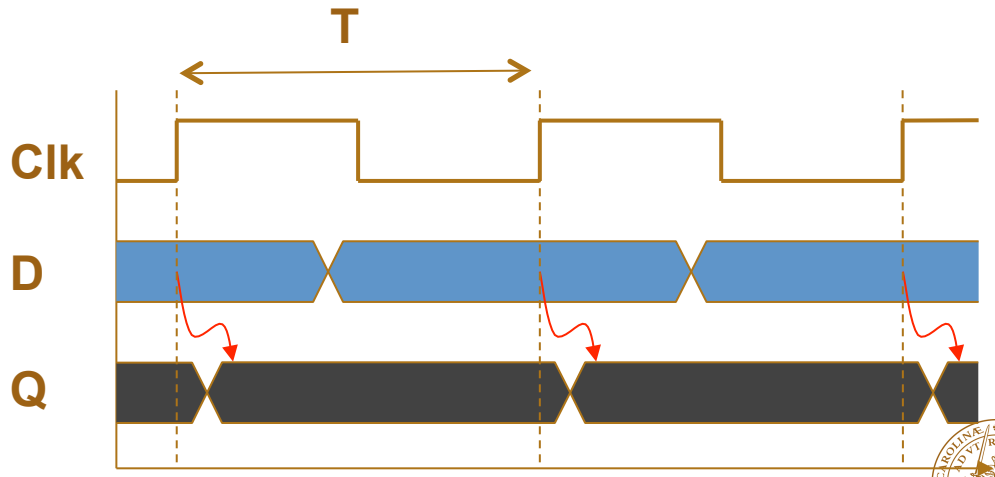


LUNDS UNIVERSITET

Sekventiell logik

- Register: lagrar data
- Använder klocka för att bestämma när data ska uppdateras
- Flank-triggat (Edge-triggered): uppdaterar när Clk går från 0 till 1





LUNDS UNIVERSITET

Register

- 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)

LUNDS UNIVERSITET

Primärminne

LOAD – läs data i minne

- data läses från en given minnesplats (adress) och hamnar i processorn

STORE – skriv data i minne

- data skrivs in i minnet på en given adress

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

- 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	?

LUNDS UNIVERSITET

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

LUNDS UNIVERSITET

Minne

- Exempel:

  C deklaration x char;
  // char behöver 1 byte

LOAD r1, 3

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

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

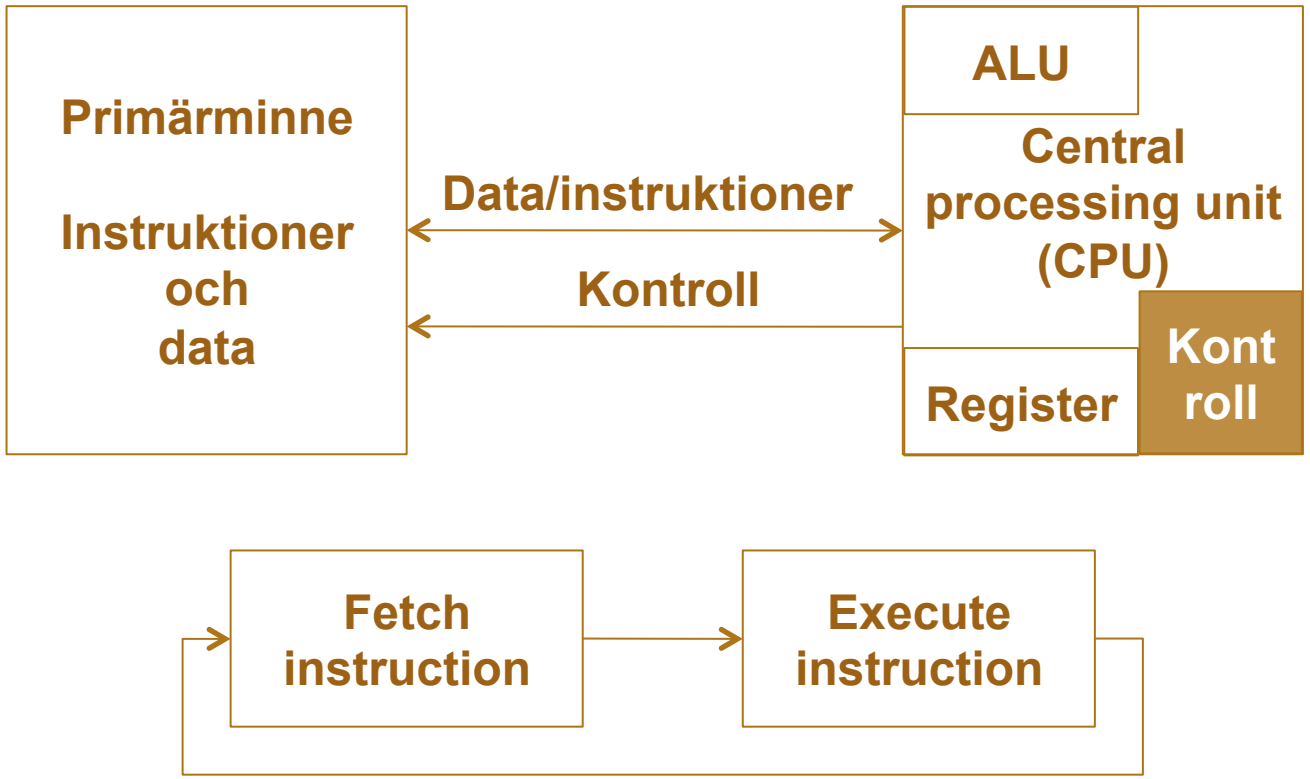
Givet: **1111 1111** **0000 1111** **1111 0000** **1010 1010**

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

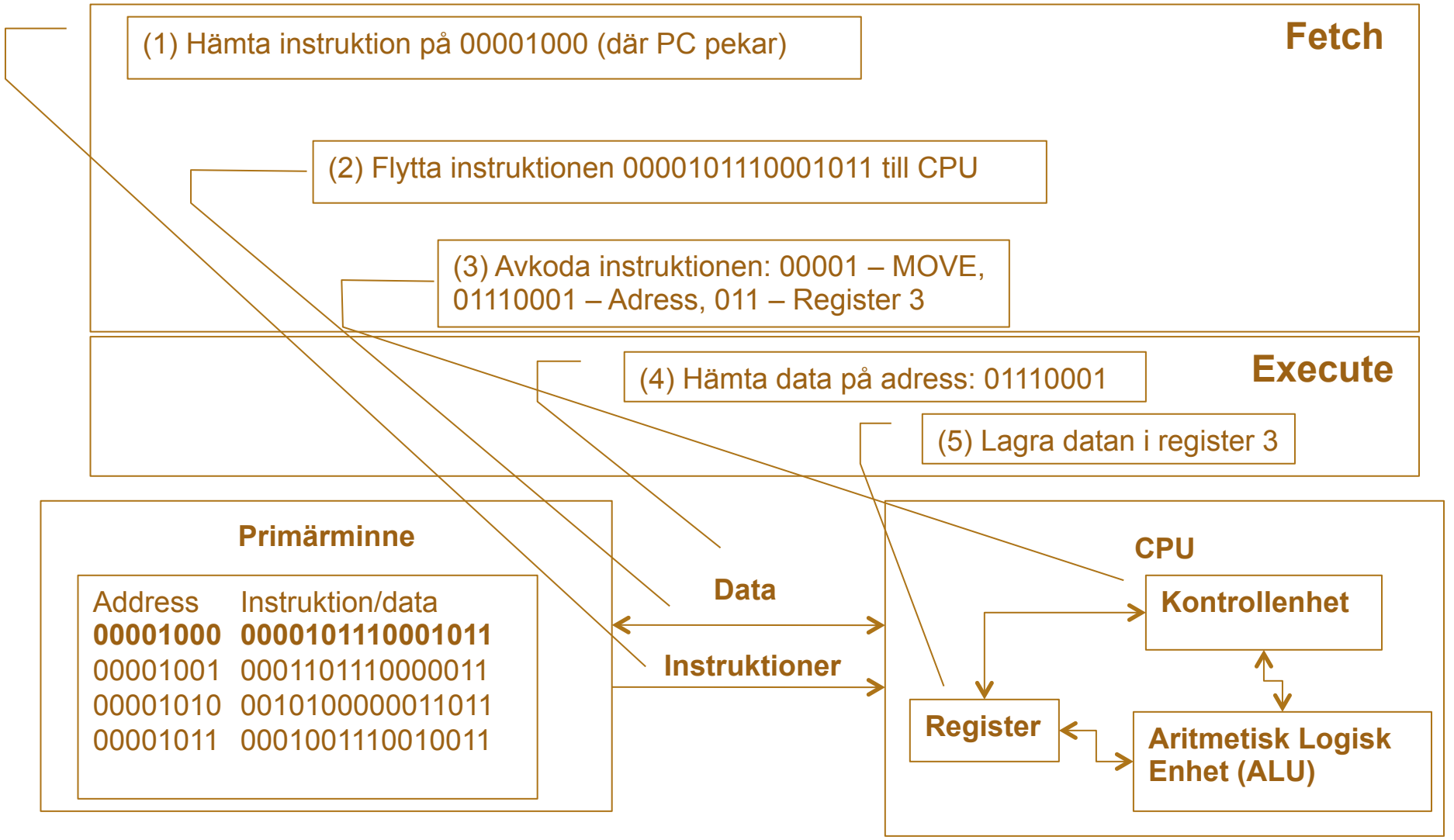
Little-endian	
Adress	Data
….	….
n	1010 1010
n+1	1111 0000
n+2	0000 1111
n+3	1111 1111
….	….

LUNDS UNIVERSITET

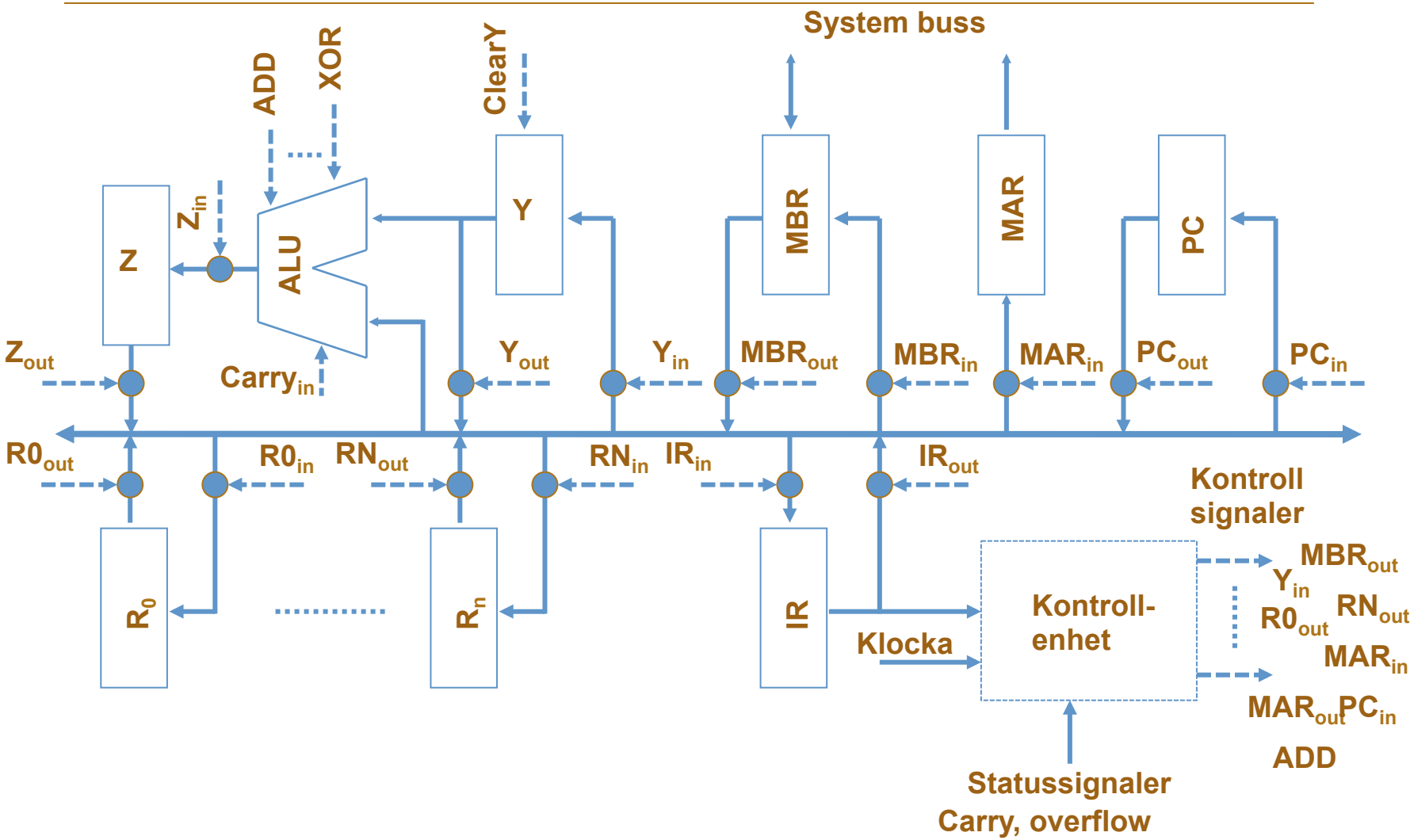
Kontrollenhet



Exekvering av en instruktion



Kontrollenhet

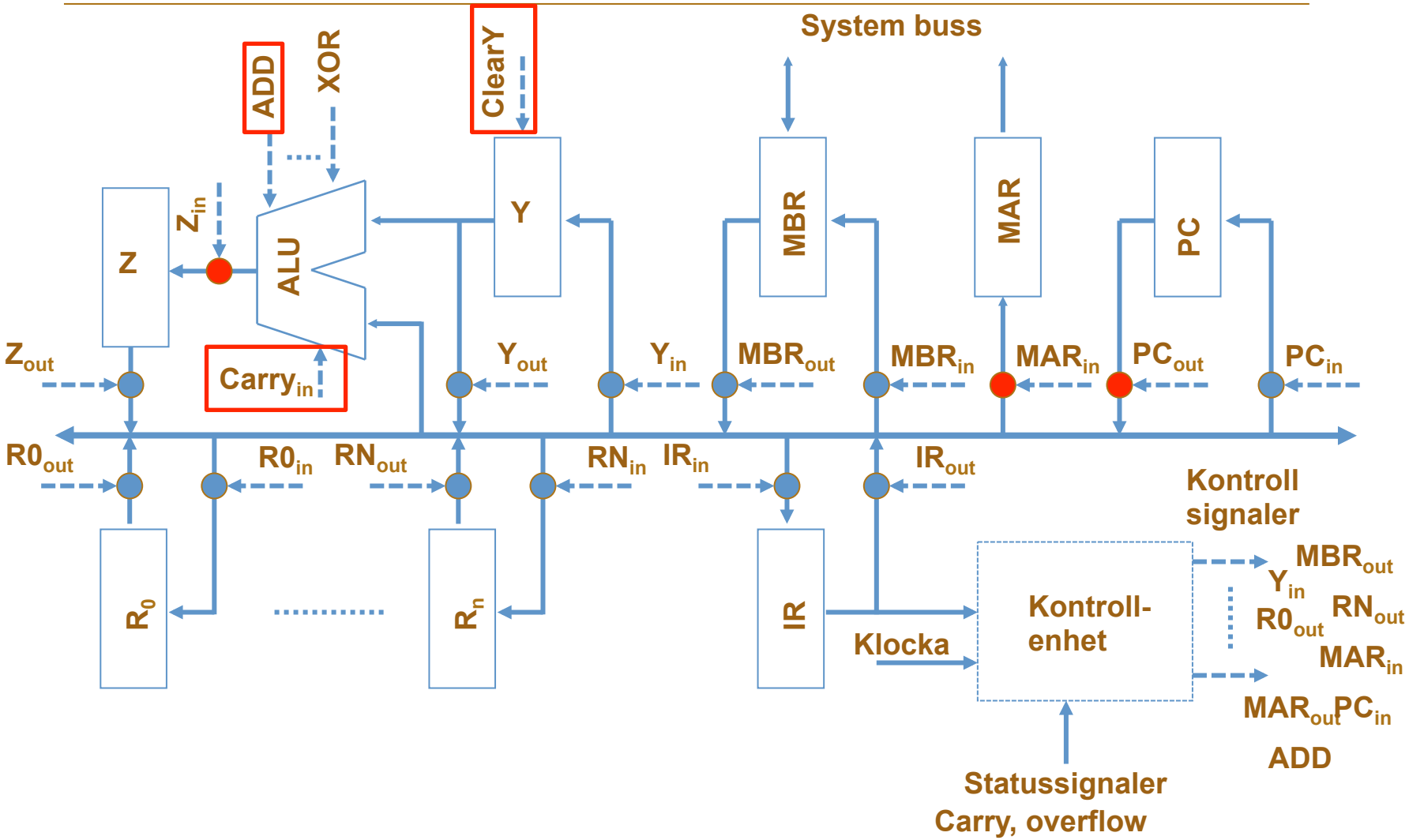


Kontrollenhet

- 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

LUNDS UNIVERSITET

Kontrollenhet

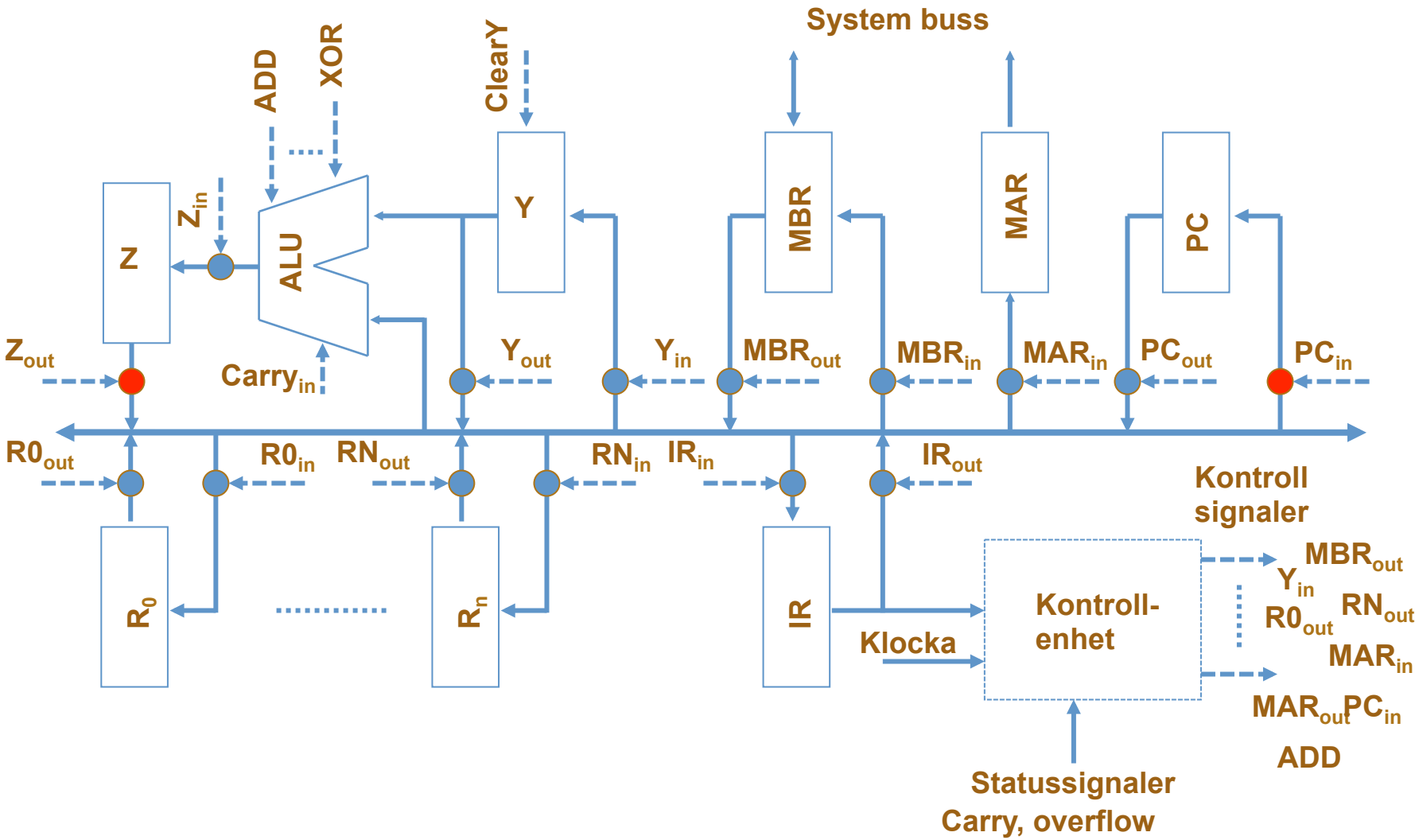


Kontrollenhet

- 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

LUNDS UNIVERSITET

Kontrollenhet

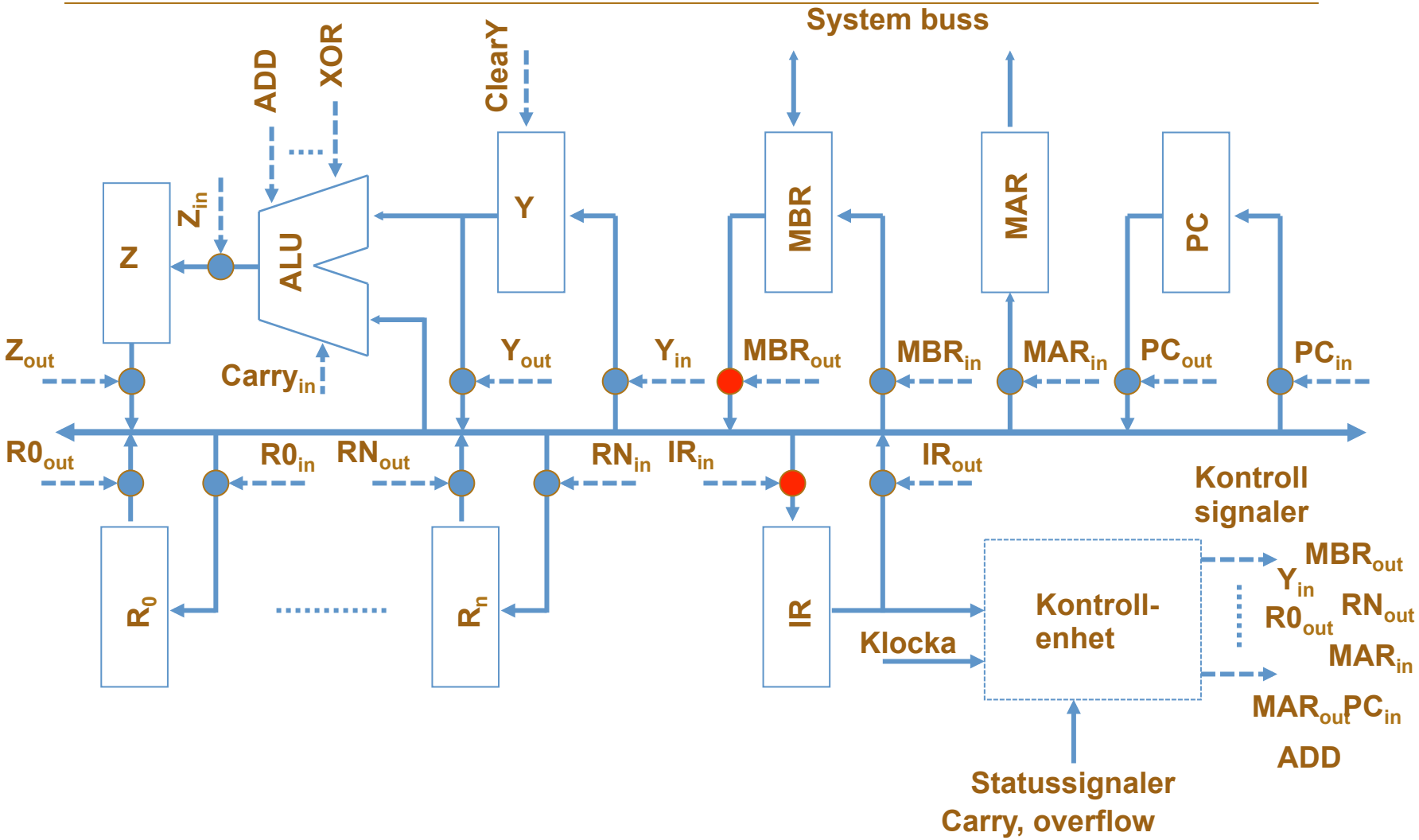


Kontrollenhet

- 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

LUNDS UNIVERSITET

Kontrollenhet

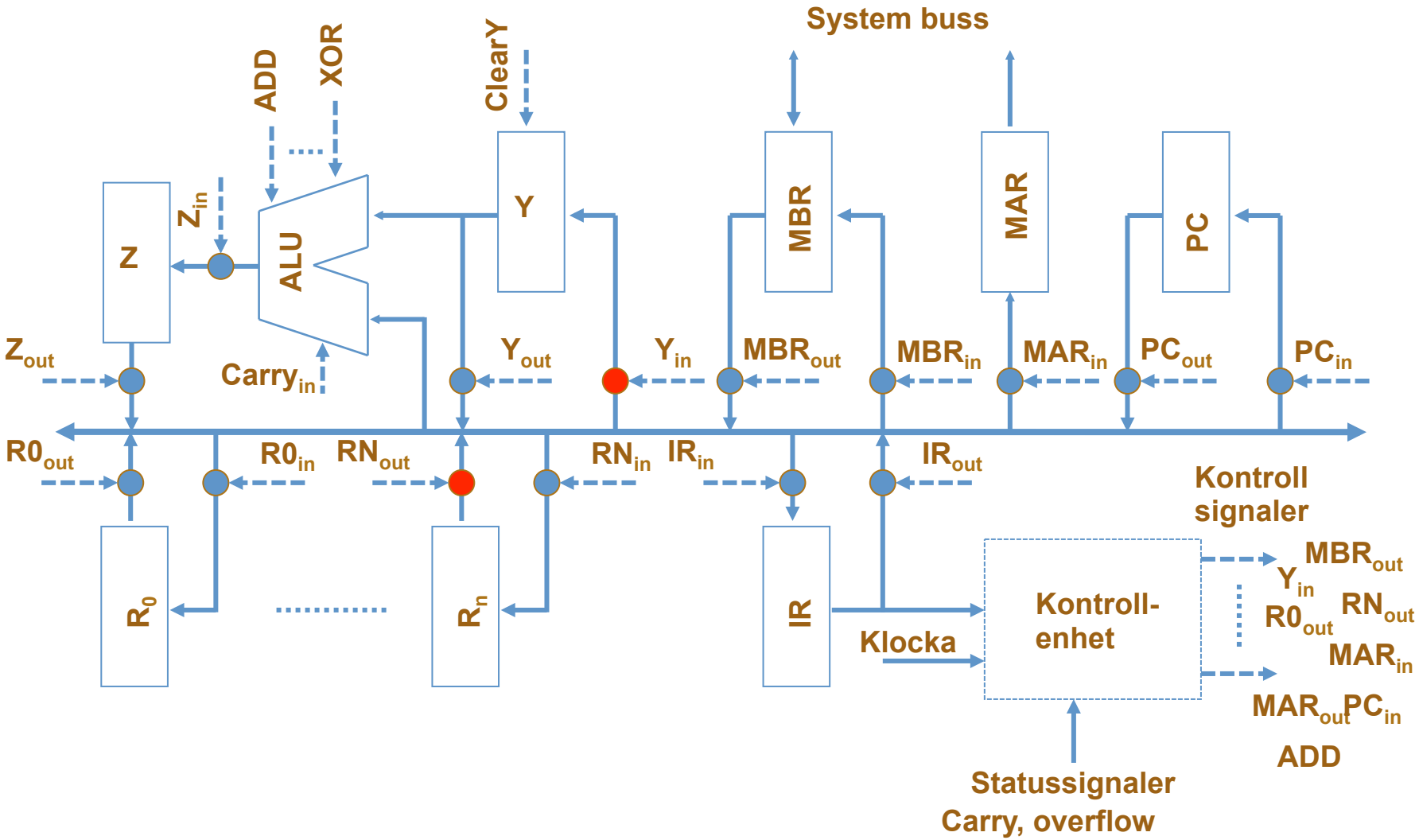


Kontrollenhet

- 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

LUNDS UNIVERSITET

Kontrollenhet

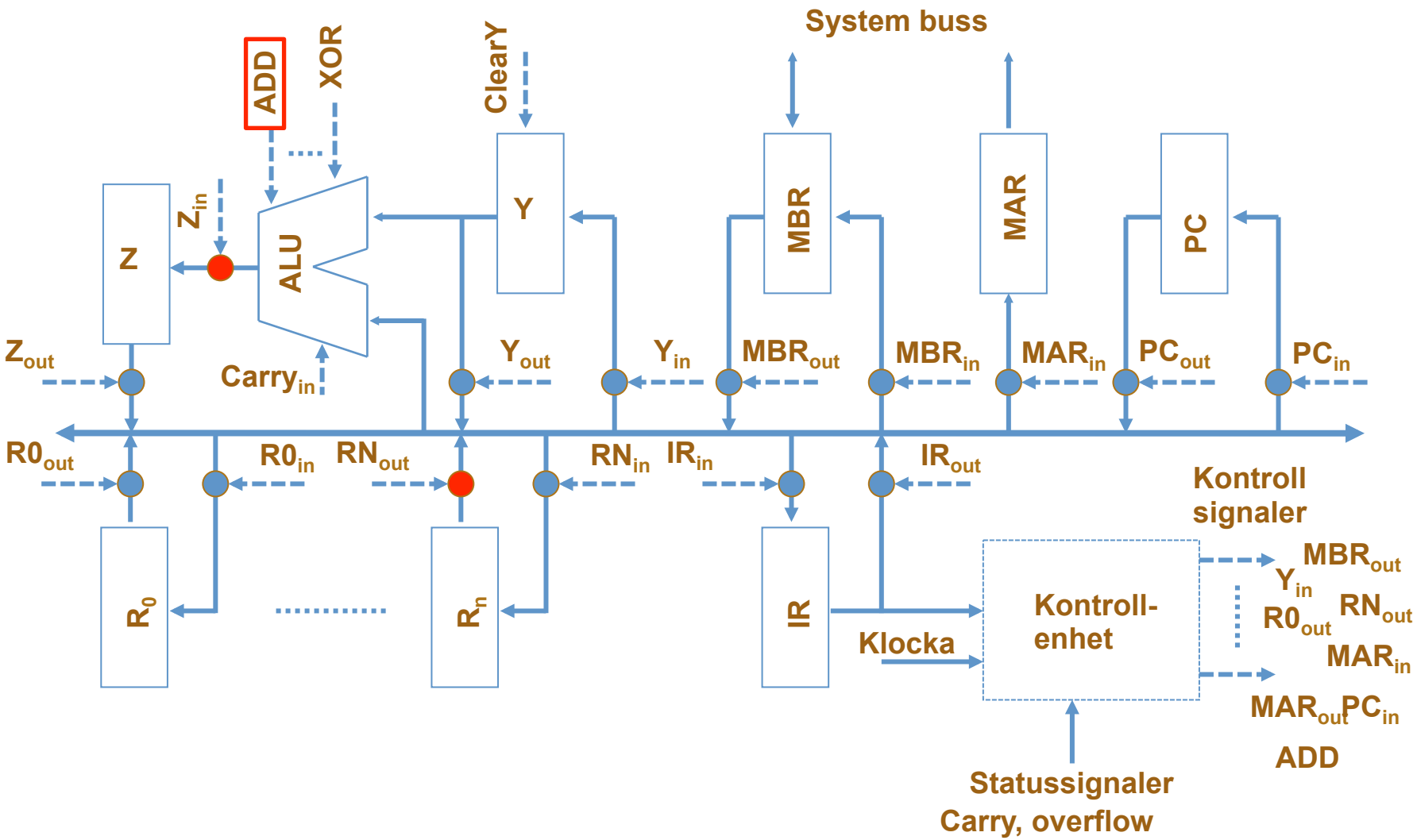


Kontrollenhet

- 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

LUNDS UNIVERSITET

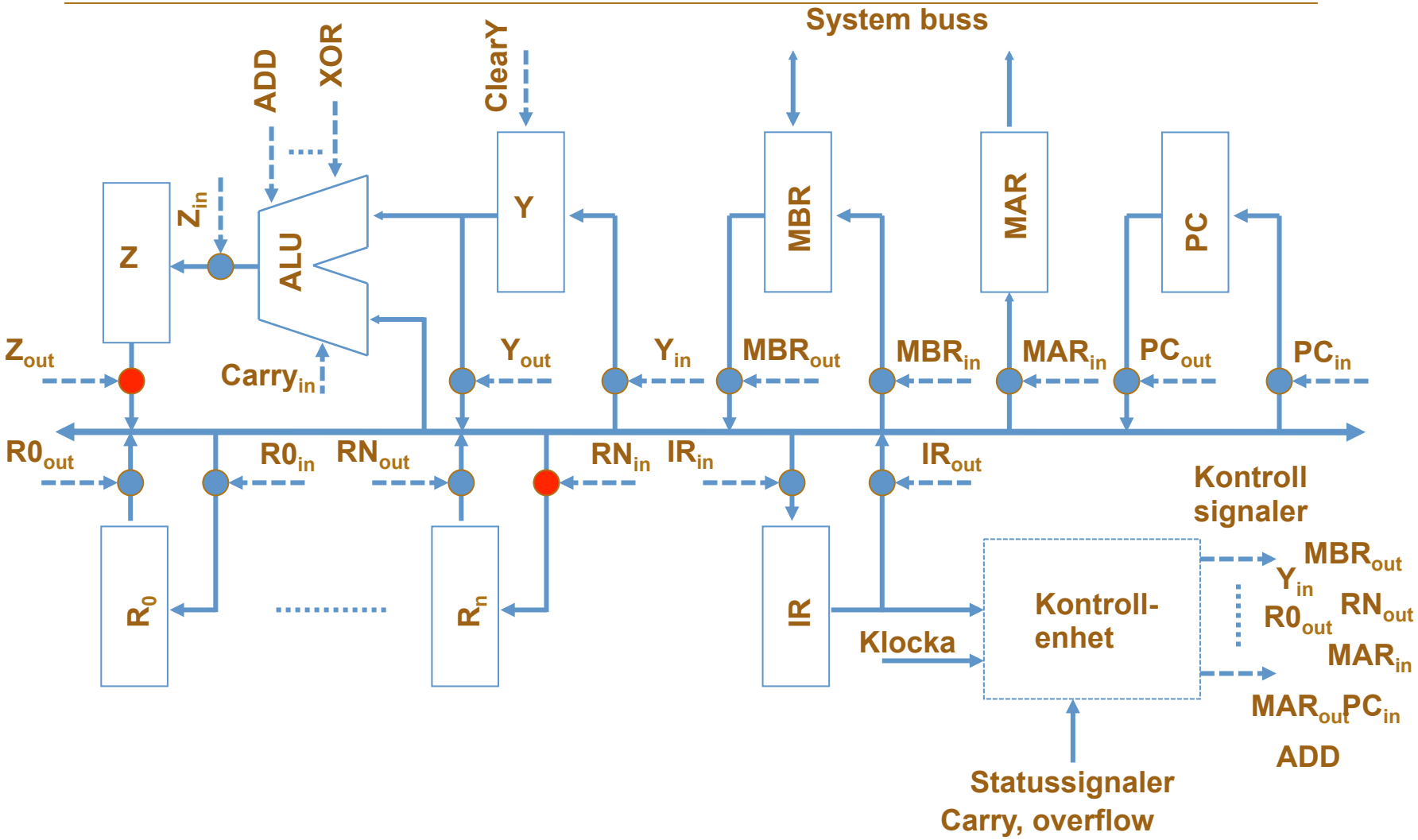
Kontrollenhet



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



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

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

LUNDS UNIVERSITET

LUNDS
UNIVERSITET