LUNDS UNIVERSITET

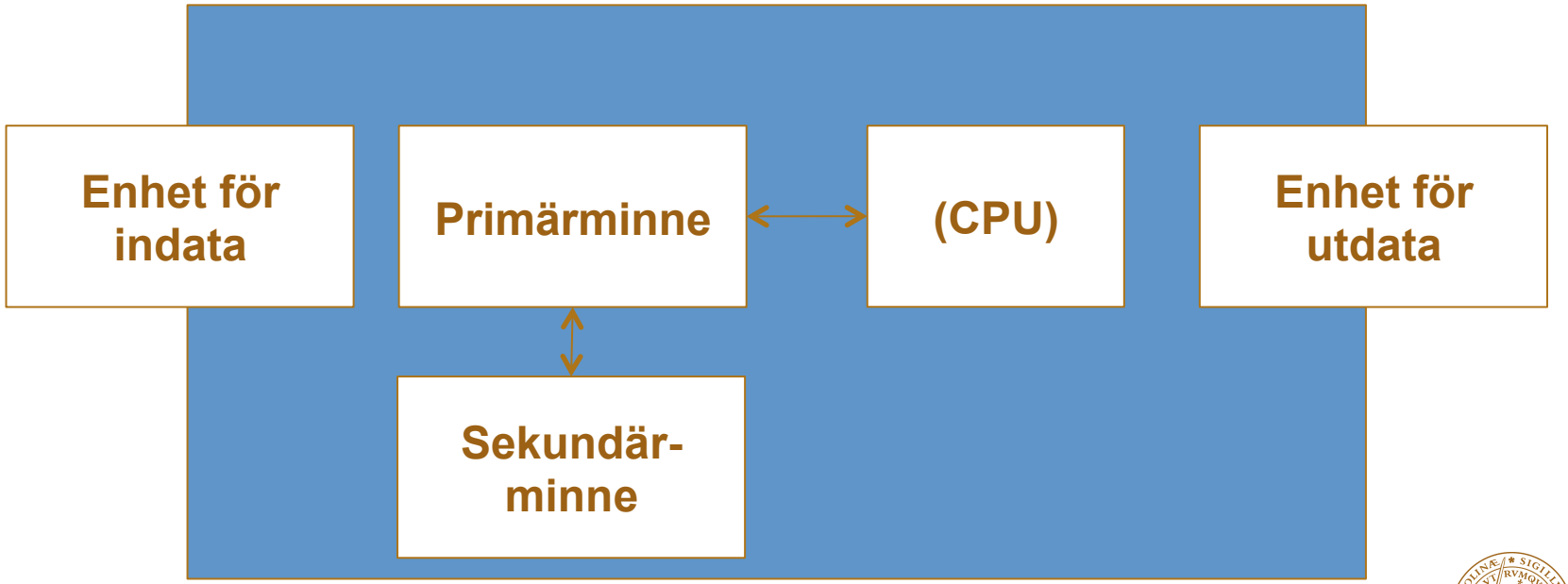
Digitala System: Datorteknik

ERIK LARSSON

Översikt

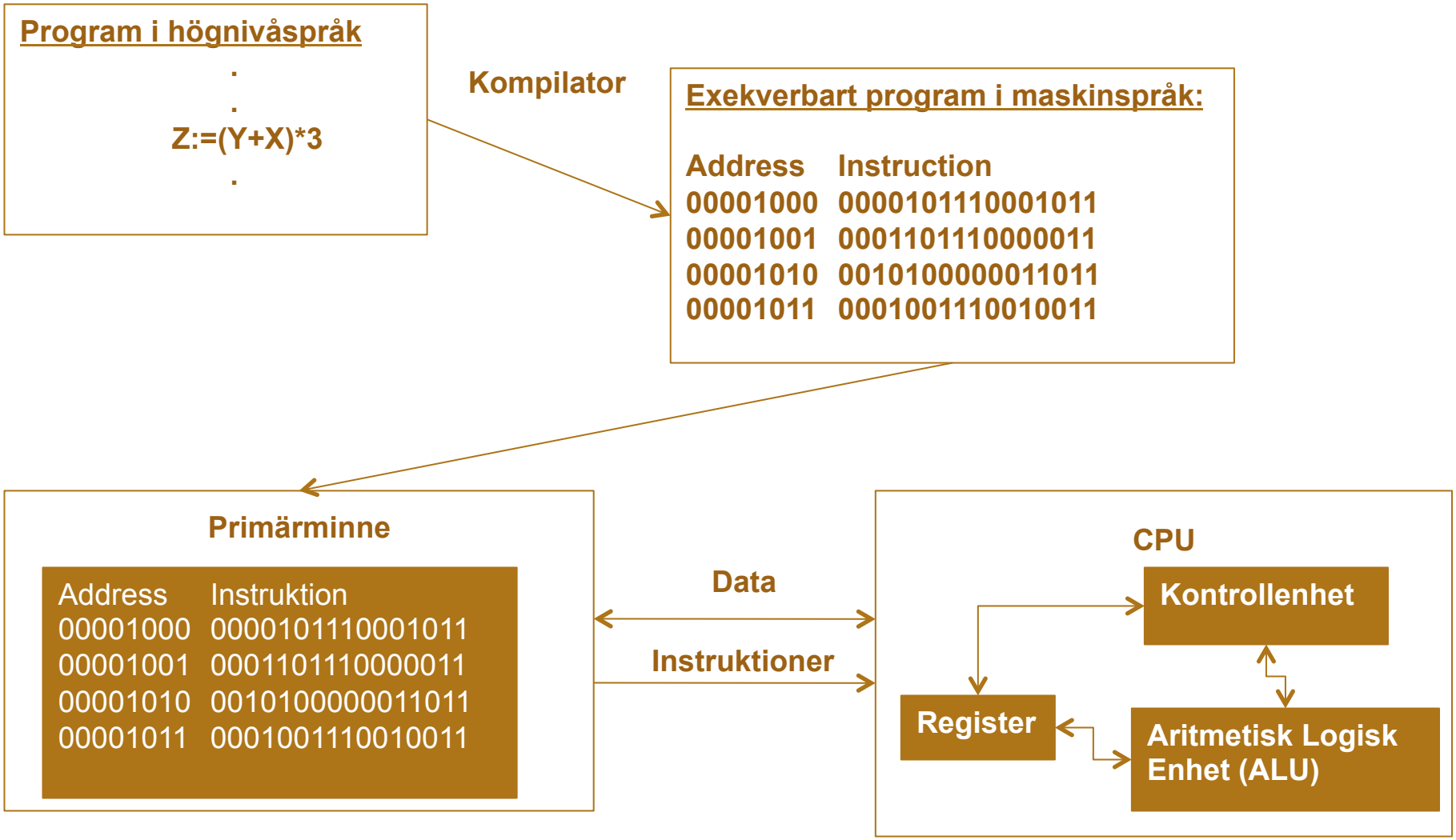
- Minnets komponenter
- Minneshierarkin
- Cacheminne
- Paging
- Virtuellt minne

Minnets komponenter

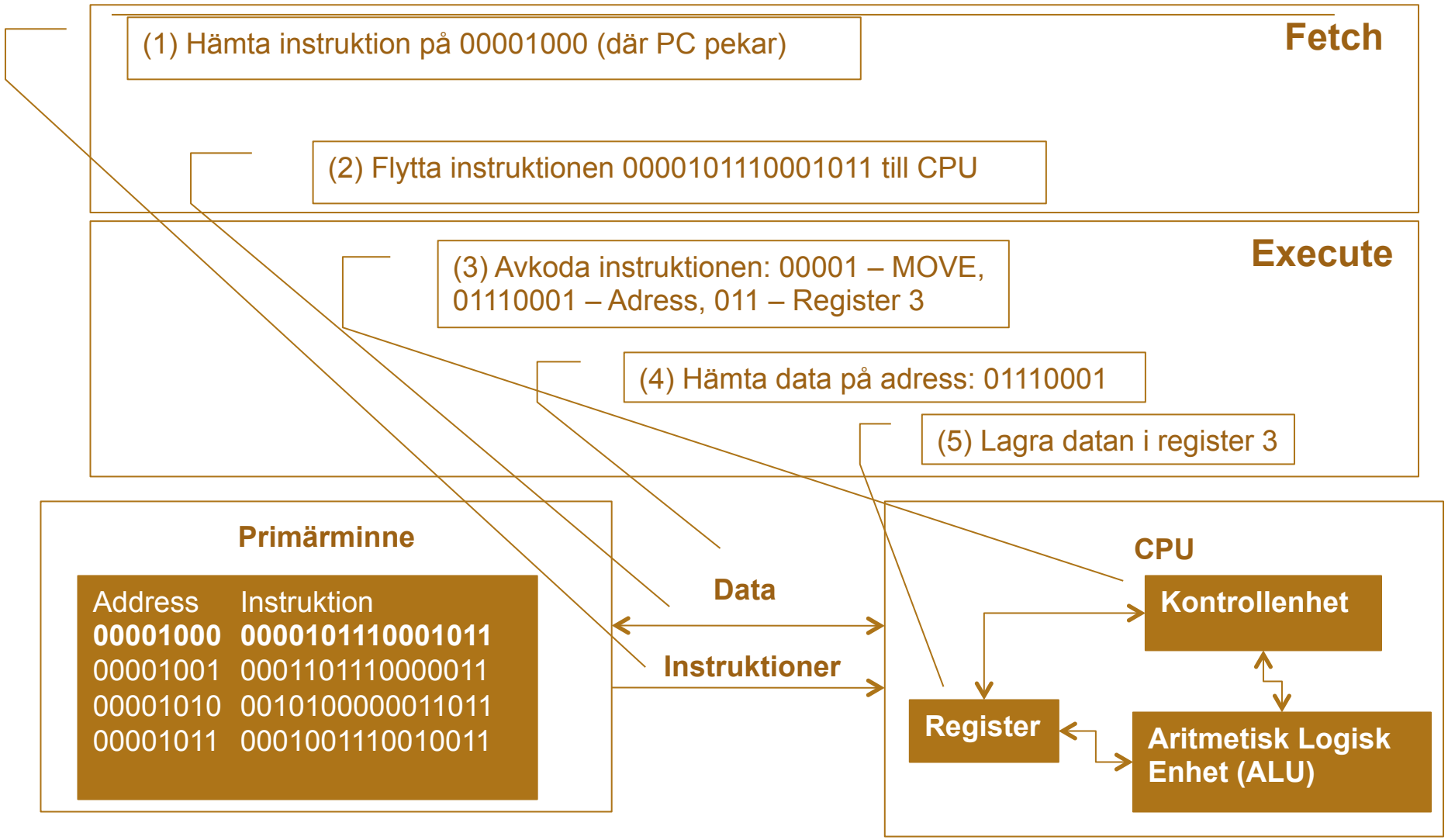


LUNDS UNIVERSITET

Programexekvering

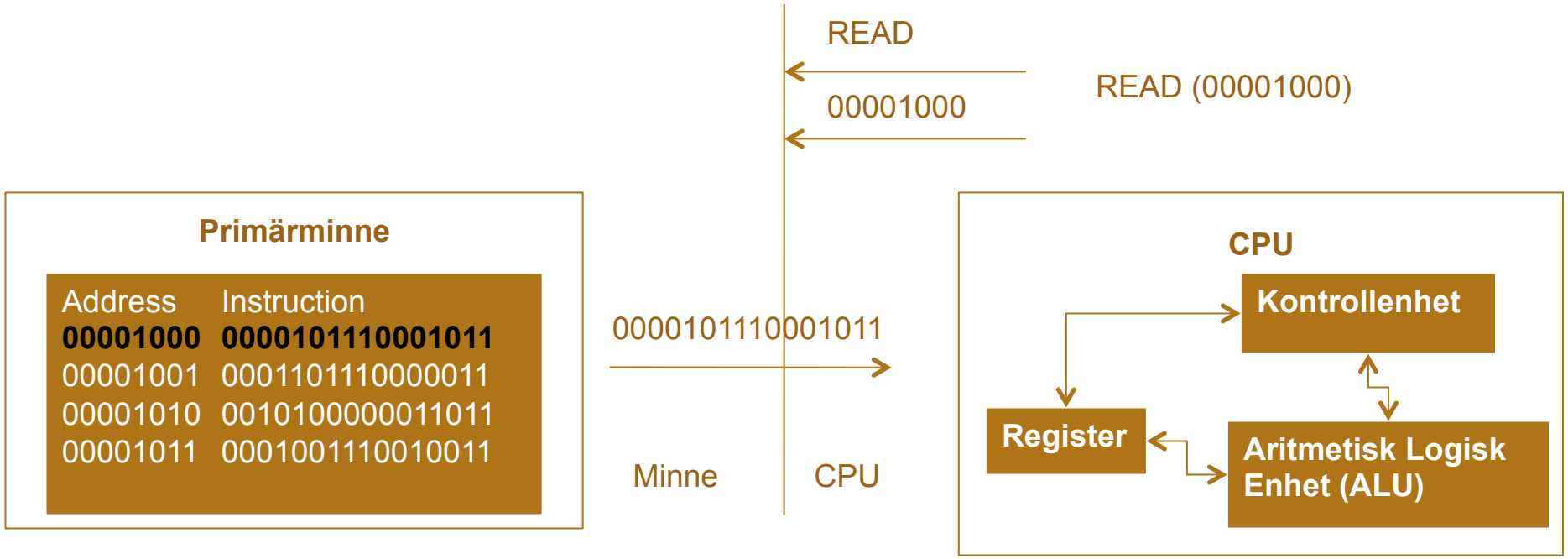


Programexekvering



Minnet från processorns sida

- Processorn ger kommandon/instruktioner med en adress och förväntar sig data.
  - Exempel: READ(ADR) -> DATA

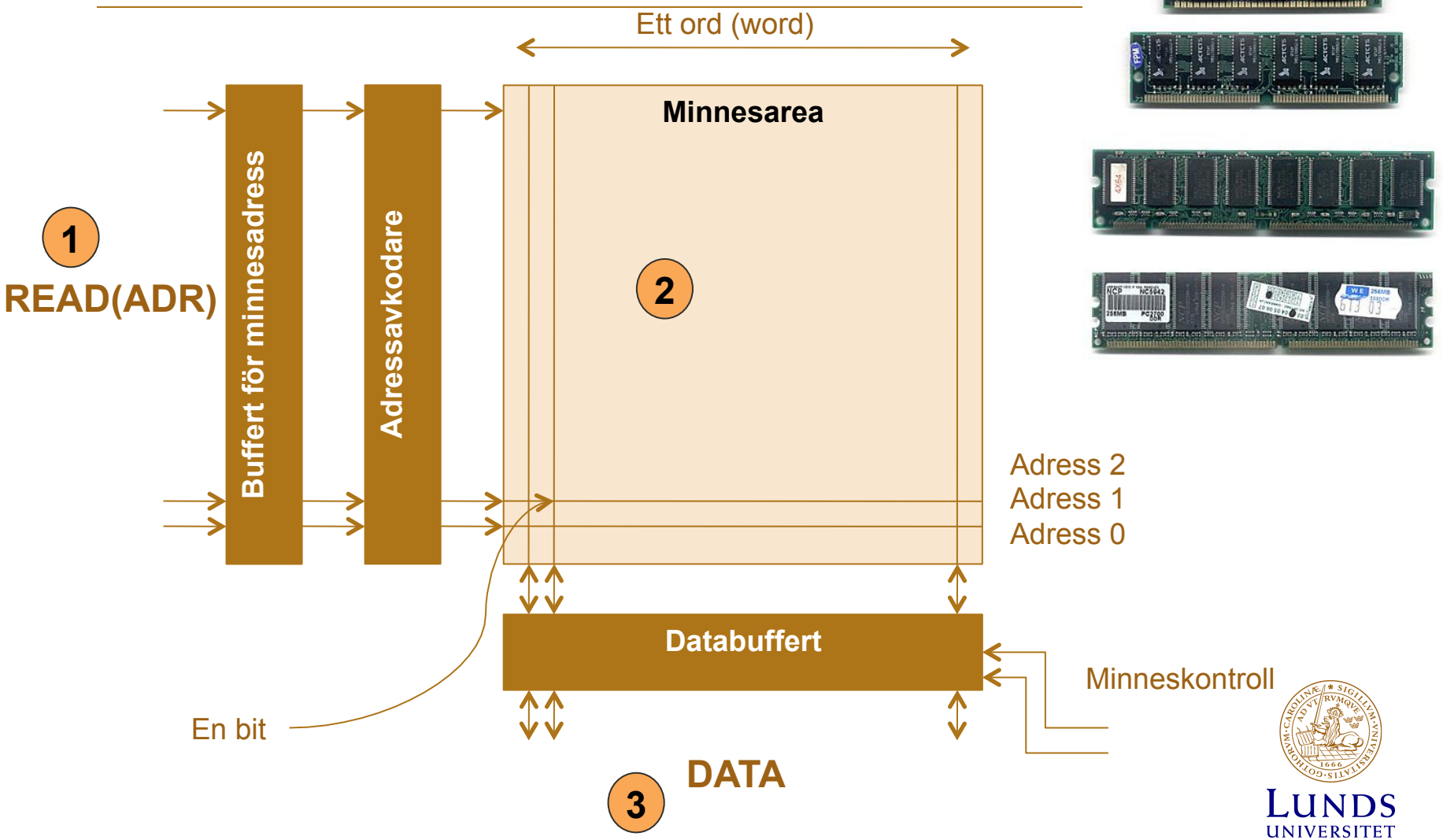


Minnet

- Minnet kan delas upp i primärminne och sekundärminne
- Primärminnet förlorar sitt innehåll när strömmen stängs av. Minnet är flyktigt (volatile)
  - Random-Access Memory (RAM)
    - » Dynamiska RAM (DRAM) och statiska RAM (SRAM)
- Sekundärminnet behåller sitt innehåll när strömmen slås av. Minnet är icke-flyktigt (non-volitile)
  - Hårddisk, flashminne, magnetband
- Andra: CD, DVD

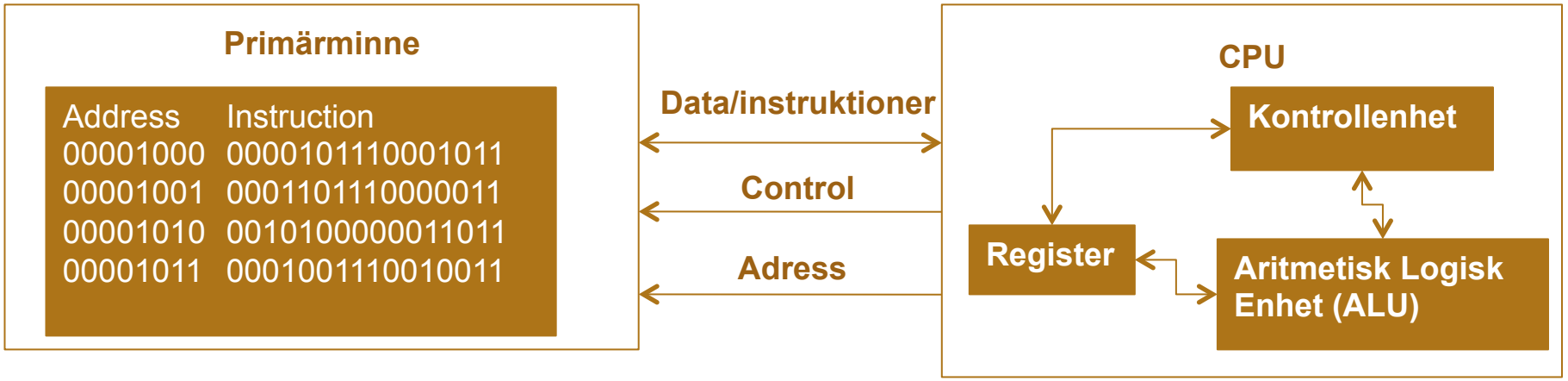
LUNDS UNIVERSITET

Primärminne



LUNDS UNIVERSITET

Primärminne

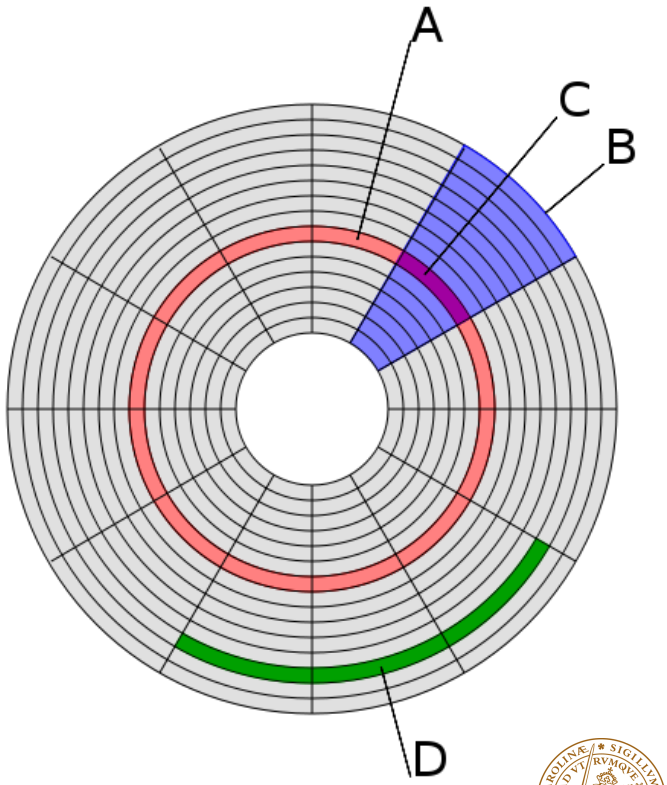


Sekundärminne

- Flera skivor – bildar cylinder.
- Läs och skrivhuvud flyttas för att läsa önskad data
- Tid för att “hitta” data: någon ms

Sekundärminne

- A – track, (B – geometrisk sektor)
  C – sektor, D – cluster
- En sektor kan vara 512-4096 bytes och består av sektor header, data area, error korrektion kod (ECC)



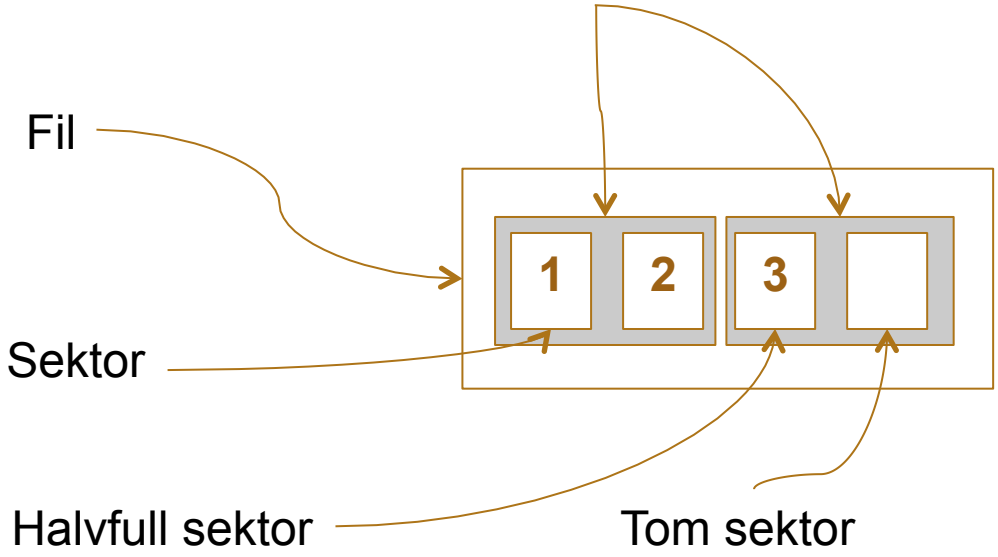
LUNDS UNIVERSITET

Sekundärminne

- Vill lagra en fil som är 1500 bytes
- Hårddisk
  - Antag sektor = 512, cluster = 2*512
- Lösning:
  - Ta 2 stycken cluster (2048 bytes)

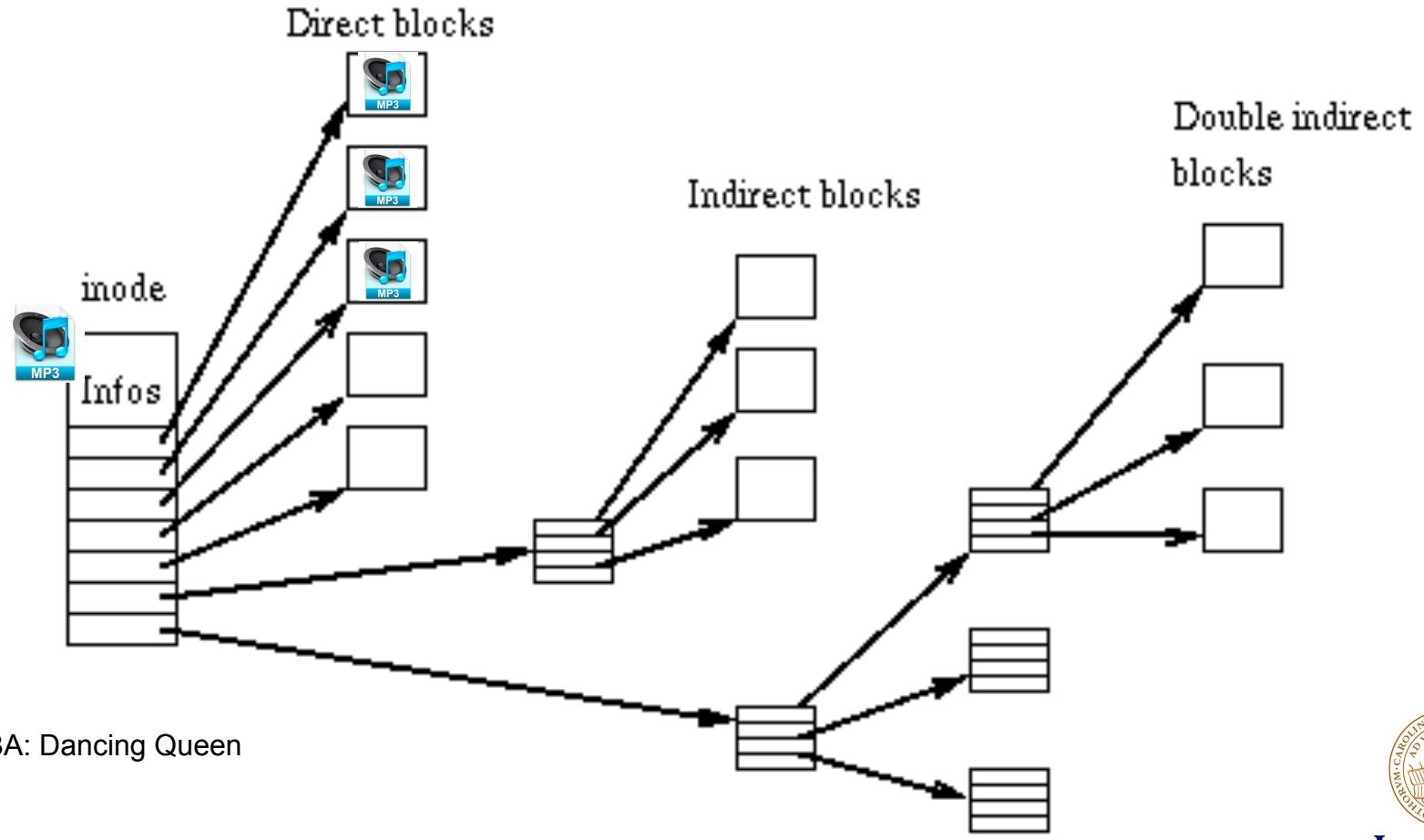
byte 0
byte 2
……
byte 1499
1500 bytes

byte 0
byte 2
……
byte 1023
2*512 bytes



LUNDS UNIVERSITET

Filsystem - Inode



ABBA: Dancing Queen

LUNDS UNIVERSITET

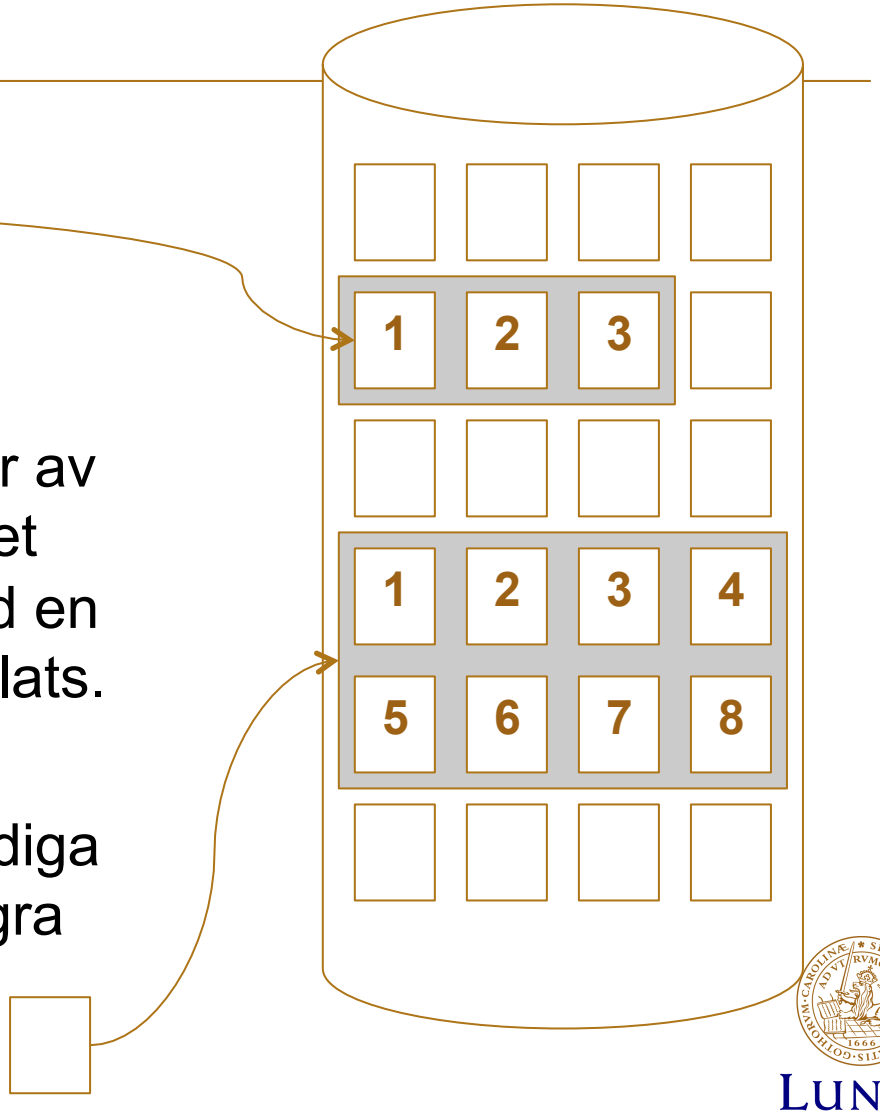
Sekundärminne

- Problem 1
  - Intern fragmentering
- Problem 2
  - Vilka cluster på hårddisk ska användas?

LUNDS UNIVERSITET

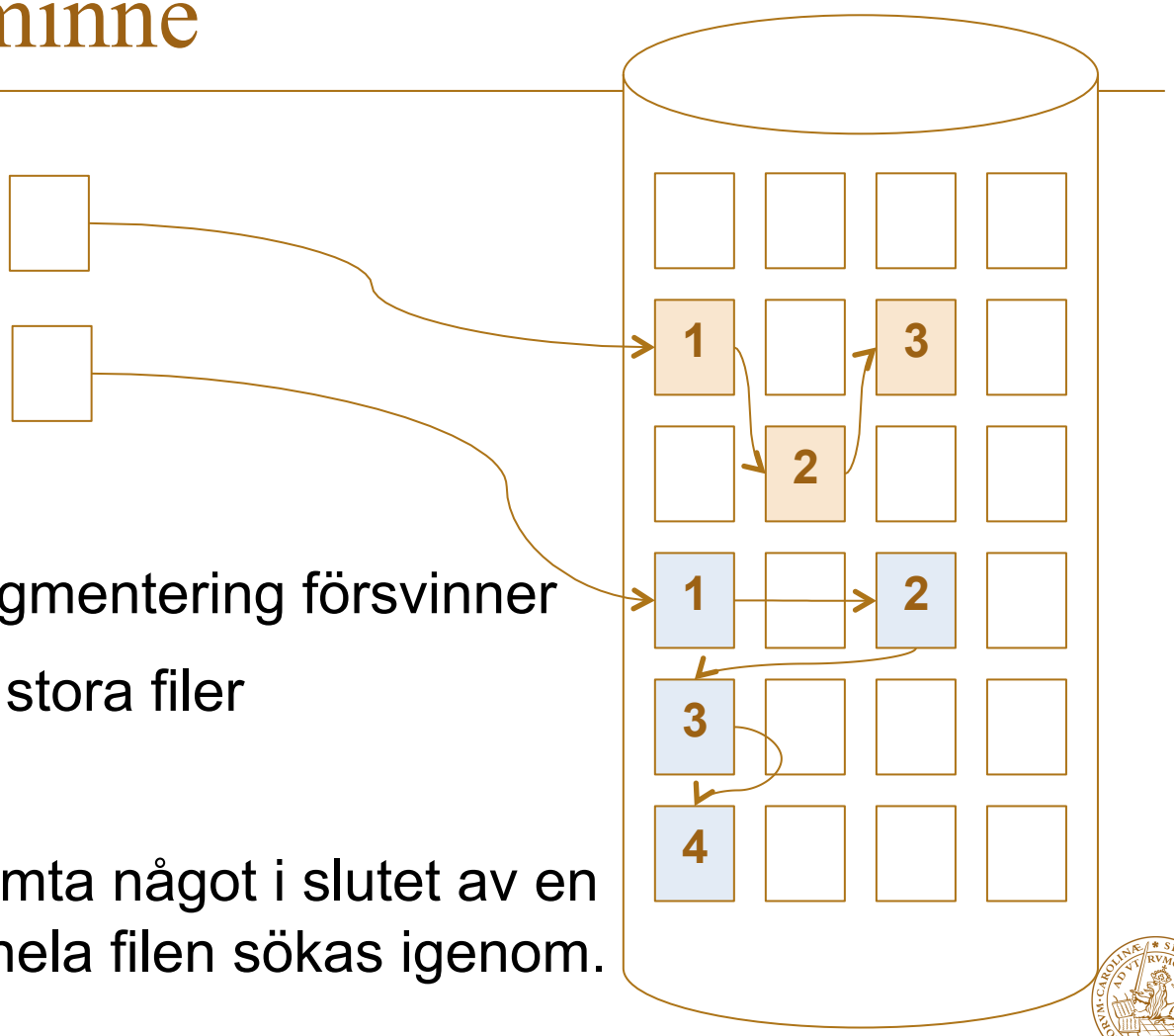
Sekundärminne

- Närliggande:
- Välj cluster som ligger bredvid varandra
- Problem – när många filer av olika storlek lagras kan det vara svårt att få plats med en stor fil trots att det finns plats.
  - Extern fragmentering: Exempel: Finns 13 lediga cluster (block). Vill lagra en fil som behöver 5 block – men hur?



Sekundärminne

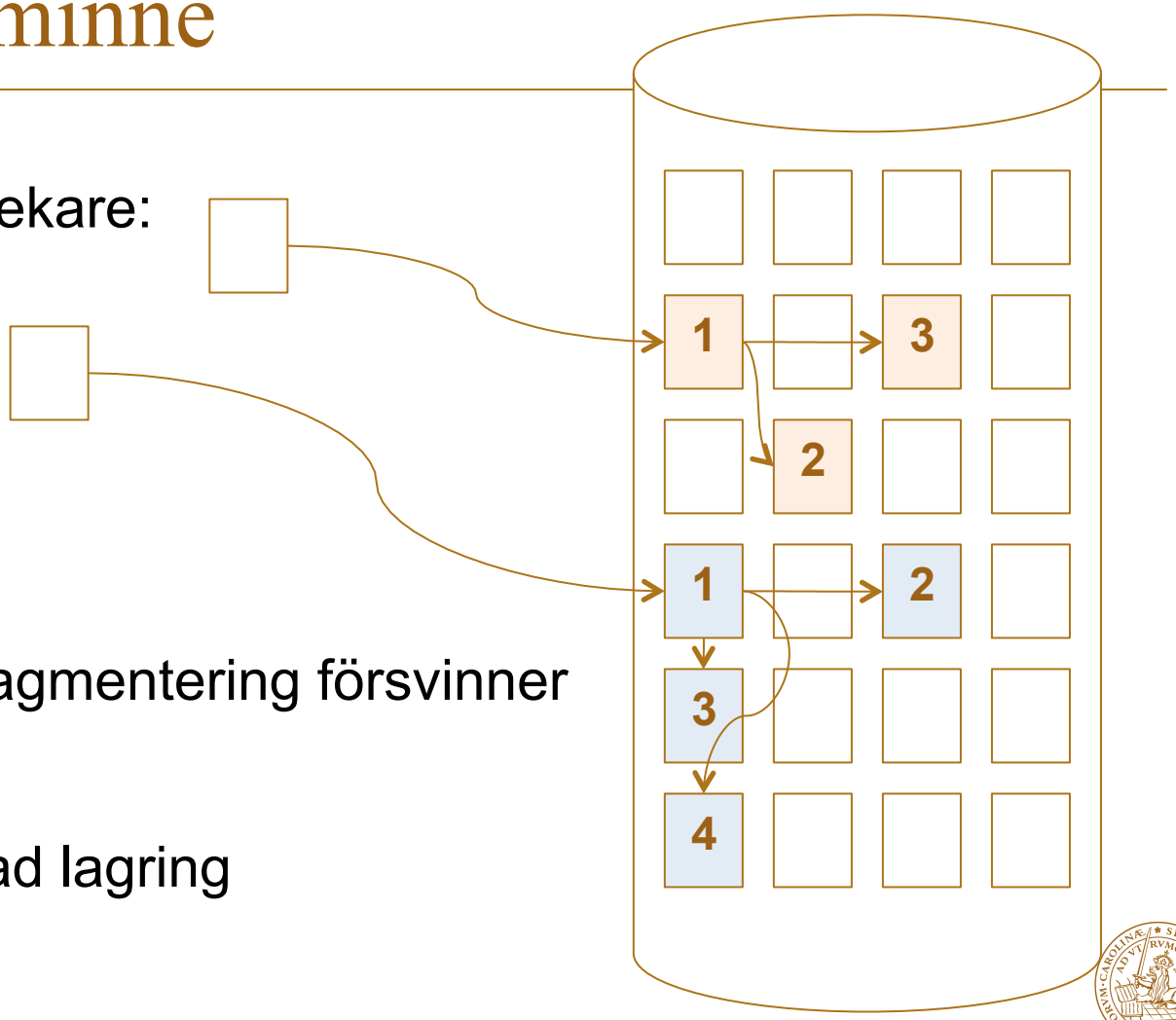
- Länkad lista:



- Fördel:
  - extern fragmentering försvinner
  - kan lagra stora filer
- Nackdel:
  - För att hämta något i slutet av en fil måste hela filen sökas igenom.

Sekundärminne

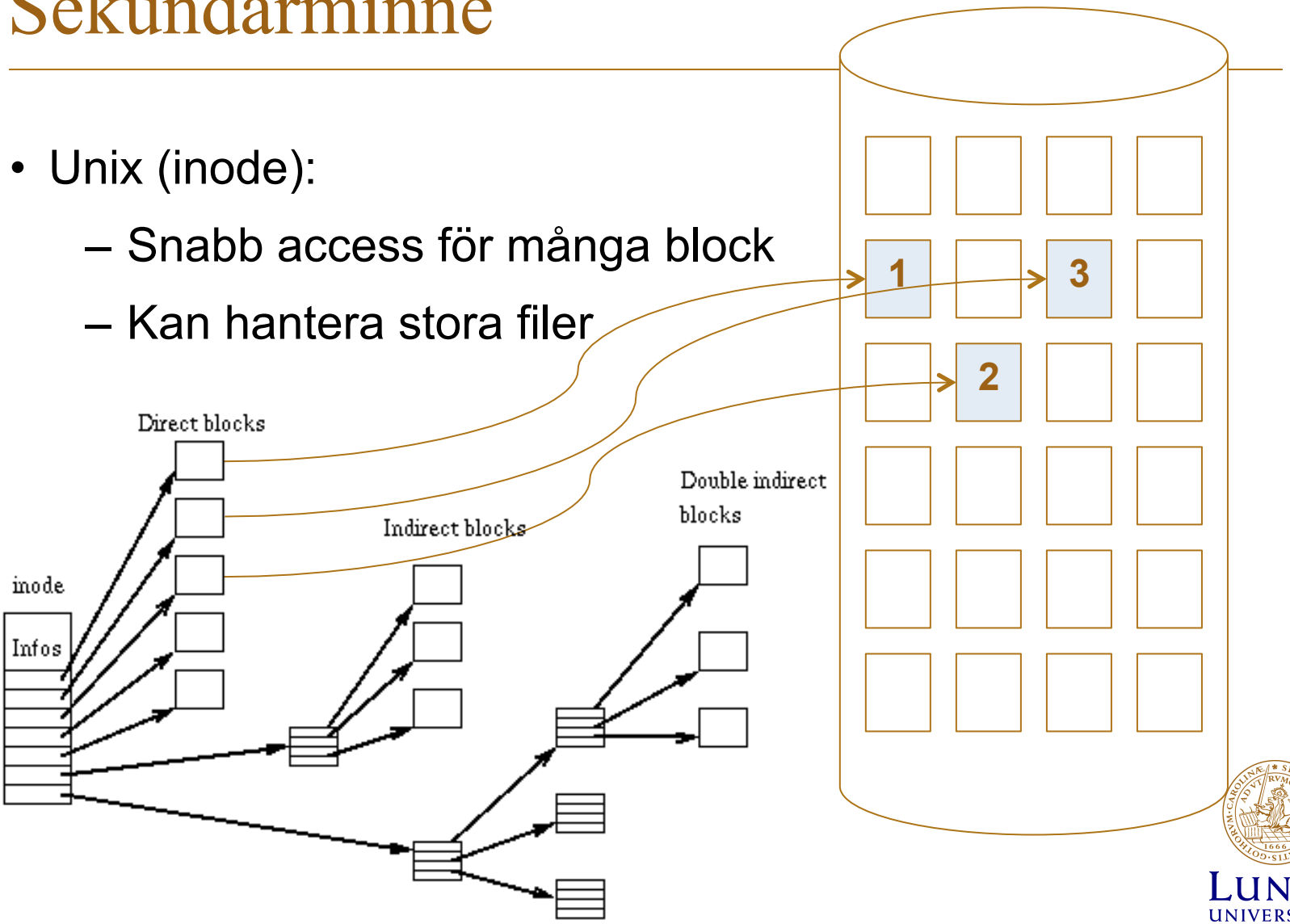
- Block med pekare:



- Fördel:
  - extern fragmentering försvinner
- Nackdel:
  - begränsad lagring

Sekundärminne

- Unix (inode):
  - Snabb access för många block
  - Kan hantera stora filer



LUNDS UNIVERSITET

Sekundärminne

- Små kluster (blocks) ger liten intern fragmentering
  - Antag sector = 512, cluster = 2*512
  - Vill lagra en fil som är 1500 bytes
  - Ta 2 stycken cluster (~2000 bytes)
- Men, kräver mer hantering (fler kluster på hårddisken)

- Vad innebär det att ta bort en fil?
  - Kan man återskapa information från en hårddisk?

LUNDS UNIVERSITET

Sekundärminne

- Schemaläggning (hårddisk)
  - Läs och skrivtid på hårddisk kritiskt
  - Var/hur filer lagras
  - Olika schemaläggare:
    - » shortest-seek time - from head
    - » elevator algorithm - move back and forth
    - » one-way elevator - move in one direction

Sekundärminne

- Flashminne
  - Utvecklat av Dr. Fujio Masuoka (Toshiba) kring 1980
- Mobiltelefoner, kameror, MP3-spelare och i datorer
  - Non-volatile och random access
- Kapacitet: mindre än en “hårddisk”
- Begränsat antal skrivningar
  - Block 0: bad blocks
  - Block 1: bootable block

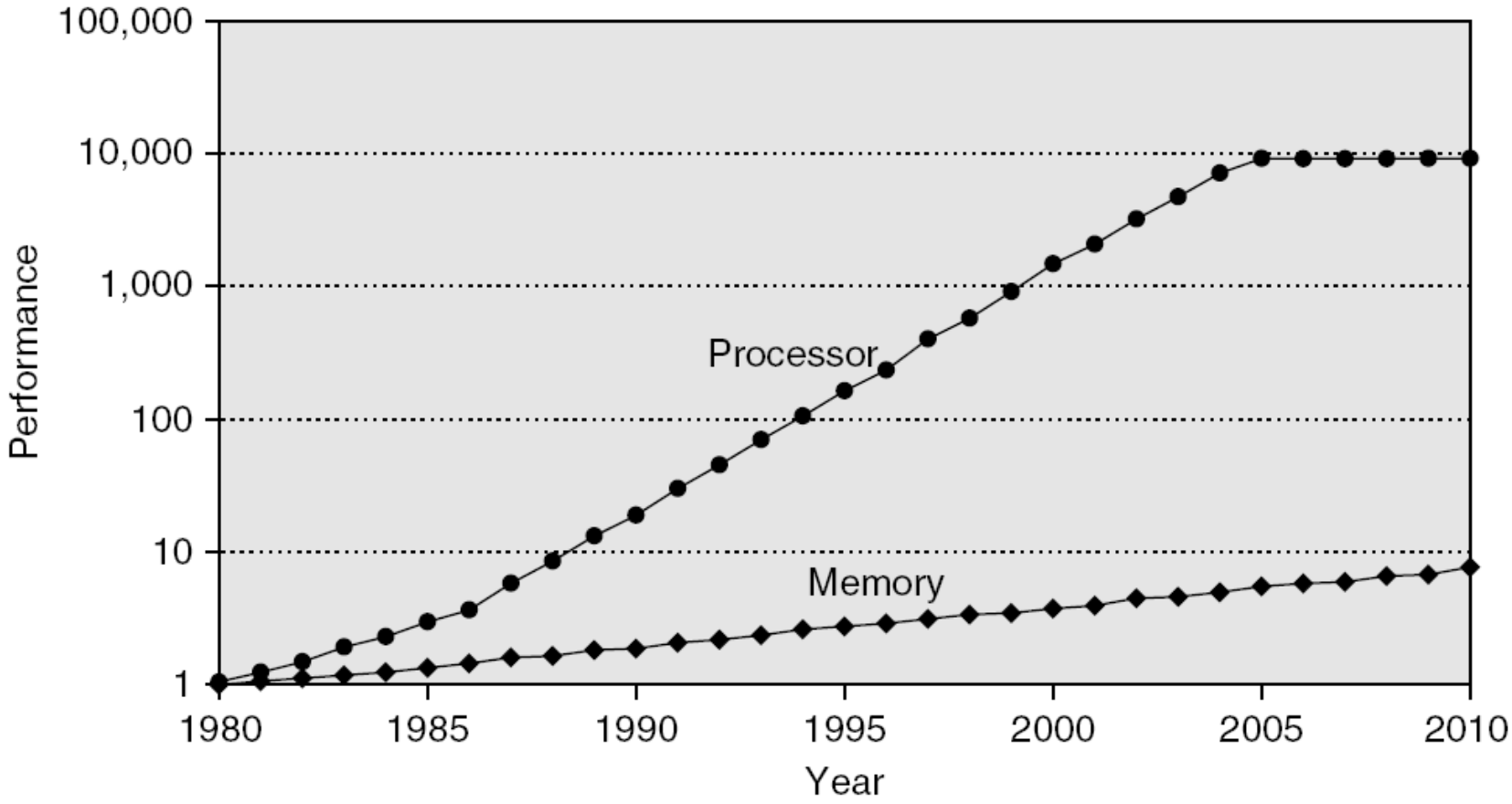
LUNDS UNIVERSITET

Sekundärminne

- Lågnivåformatering
  - Dela in hårddisk i tracks och sectors
    - » En sector är 512-4098 bytes
- Partitioning
  - Dela in en fysisk hårddisk i en eller flera logiska hårddiskar, t ex C:, D:, E:
- Högnivåformatering
  - Bestäm för vilket operativ system hårddisken ska användas

LUNDS UNIVERSITET

Minne-processor hastighet



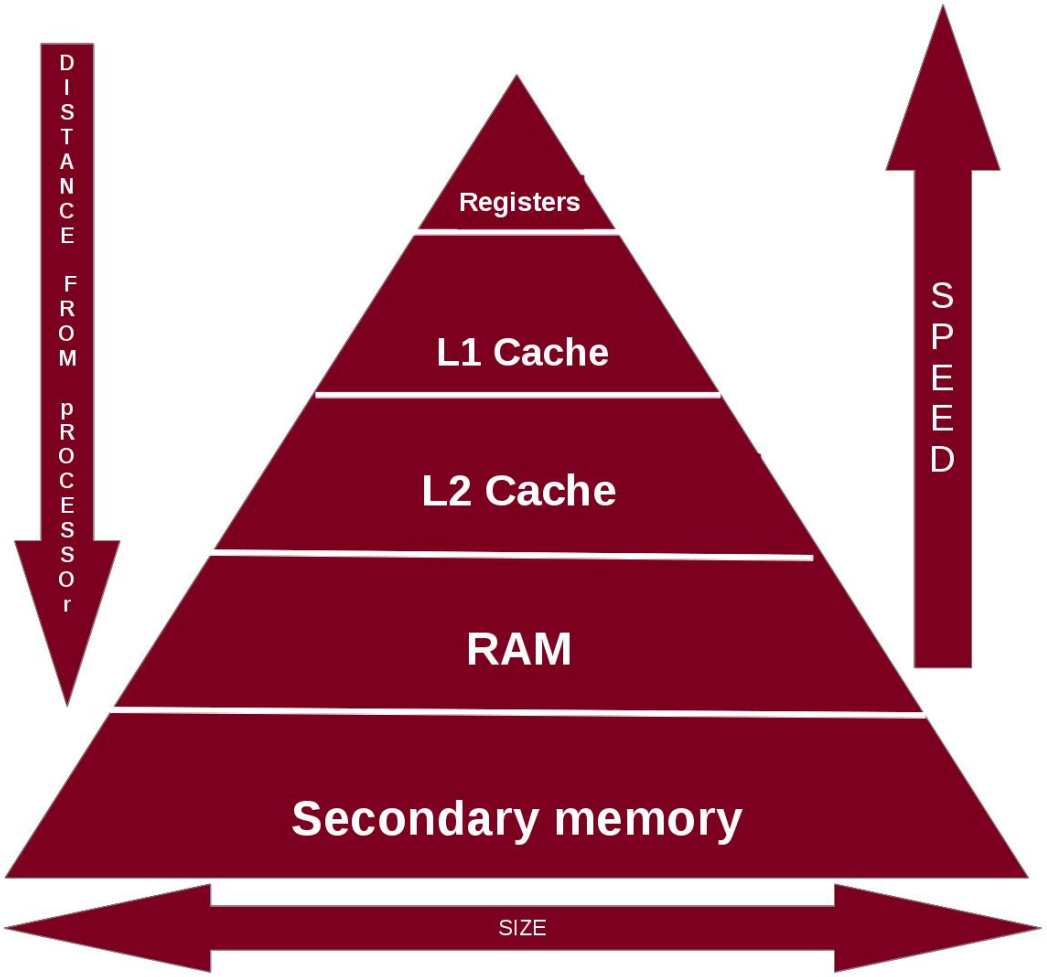
Design av minnesystem

- Vad vill vi ha?
  - Ett minne som får plats med stora program och som fungerar i hastighet som processorn
    - » Fetch – execute (MHz/GHz/Multi-core race)

- Grundproblem:
  - Processorer arbetar i hög hastighet och behöver stora minnen
  - Minnen är mycket långsammare än processorer
- Fakta:
  - Större minnen är långsammare än mindre minnen
  - Snabbare minnen kostar med per bit

LUNDS UNIVERSITET

Minneshierarki



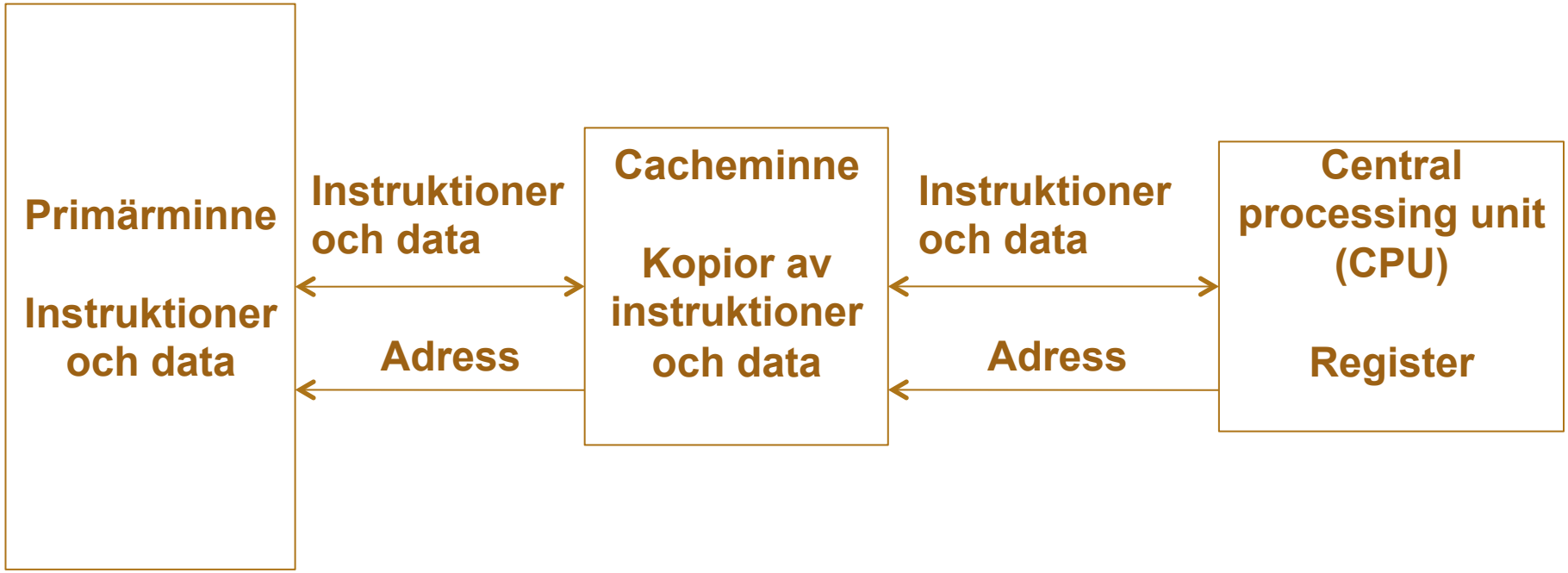
LUNDS UNIVERSITET

Minneshierarki

- Processor registers:
  - 8-32 registers (32 bitar -> 32-128 bytes)
  - accesstid: få ns, 0-1 klockcykler
- On-chip cache memory (L1):
  - 32 till 128 Kbytes
  - accesstid = ~10 ns, 3 klockcykler
- Off-chip cache memory (L2):
  - 128 Kbytes till 12 Mbytes
  - accesstid = 10-tal ns, 10 klockcykler
- Main memory:
  - 256 Mbytes till 4Gbytes
  - accesstid = ~100 ns, 100 klockcykler
- Hard disk:
  - 1Gbyte tid 1Tbyte
  - accesstid = 10-tal milliseconds, 10 000 000 klockcykler

LUNDS UNIVERSITET

Cacheminne



Accesstid: 100ns

Accesstid: 10ns

LUNDS UNIVERSITET

Cacheminne

- Ett cacheminne är mindre och snabbare än primärminnet
  - Hela program får inte plats
  - Men, data och instruktioner ska vara tillgängliga när de behövs
- Om man inte har cacheminne:
  - Accesstid för att hämta en instruktion=100ns
- Om man har cacheminne:
  - Accesstid för att hämta en instruktion=100+10=110 ns
    - » Först ska instruktionen hämtas till cacheminne och sedan hämtas instruktionen från cacheminnet till CPU

LUNDS UNIVERSITET

Cache – exempel 1

- Program: Assemblyinstruktioner
  x=x+1; Instruktion1: x=x+1;
  y=x+5; Instruktion2: y=x+5;
  z=y+x; Instruktion3: z=y+x;
- Om man inte har cacheminne:
  - Accesstid för att hämta en instruktion=100ns
    - » Tid för att hämta instruktioner: 3*100=300ns
- Om man har cacheminne:
  - Accesstid för att hämta en instruktion=100+10=110ns
    - » Tid för hämta instruktioner: 3*110=330ns

LUNDS UNIVERSITET

Cache – exempel 2

- Antag:
  - 1 maskininstruktion per rad
  - 100 ns för minnesaccess till primärminnet
  - 10 ns för minnesaccess till cacheminnet
- Programmet och dess maskininstruktioner.

Exempel program:	Assembly
while (x<1000){	Instruktion1: while1000
x=x+1;	Instruktion2: x=x+1
printf("x=%i",x);	Instruktion3: print x}
while (y<500){	Instruktion4: while500
y=y+1;	Instruktion5: y=y+1
printf("y=%i",y);	Instruktion6: print y
}	

Utan cache – exempel 2

- Antal instruktioner Instruktioner som exekveras:

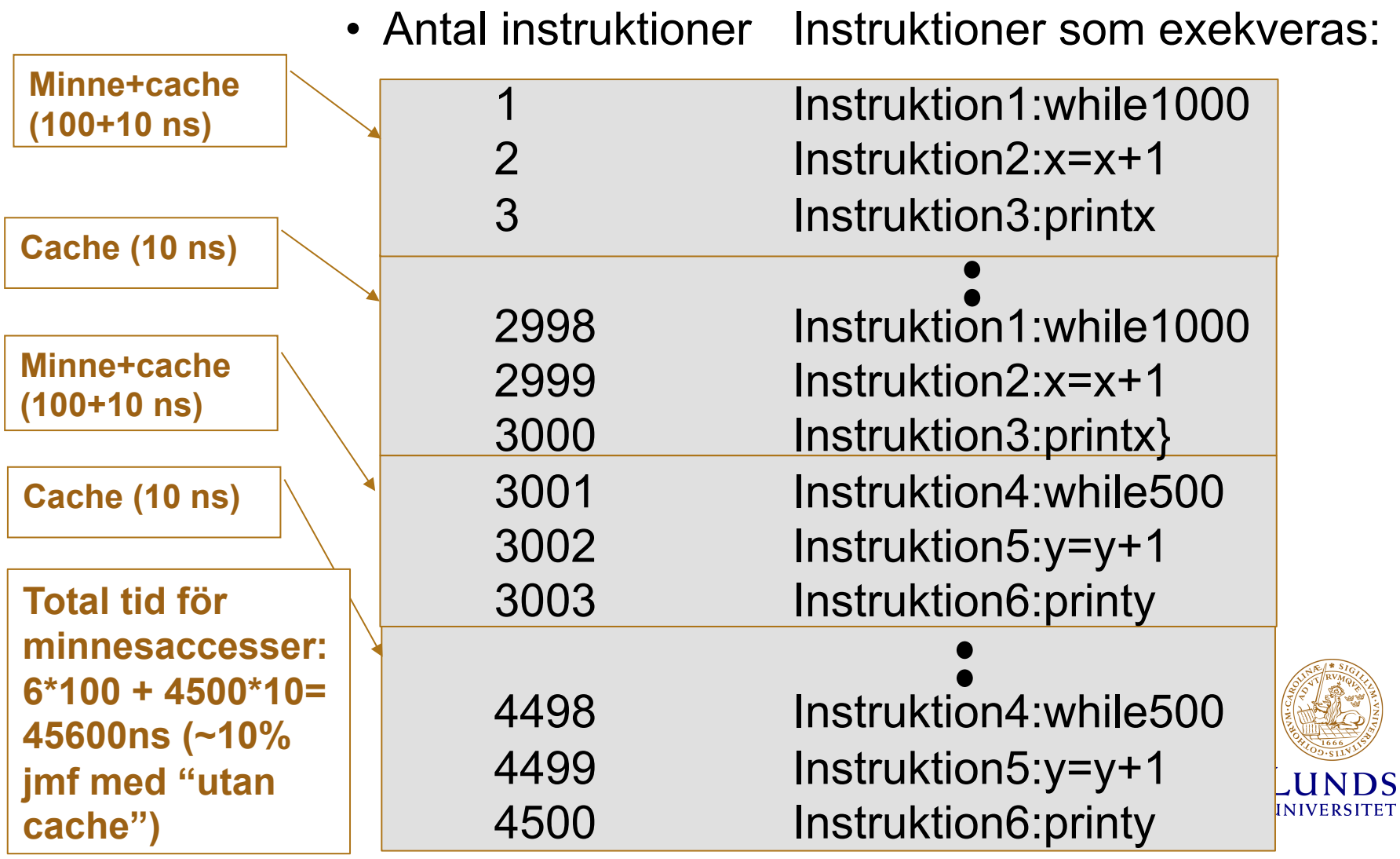
1	Instruktion1:while1000
2	Instruktion2:x=x+1
3	Instruktion3:printx
	⋮
2998	Instruktion1:while1000
2999	Instruktion2:x=x+1
3000	Instruktion3:printx}
3001	Instruktion4:while500
3002	Instruktion5:y=y+1
3003	Instruktion6:printy
	⋮
4498	Instruktion4:while500
4499	Instruktion5:y=y+1
4500	Instruktion6:printy

Minnes access för 1 instruktion: 100 ns

Totalt 4500 instruktioner.

Tid för minnesaccesser: 4500*100= 450000ns

Med cache – exempel 2



Cacheminne

- Minnesreferenser tenderar att gruppera sig under exekvering
  - både instruktioner (t ex loopar) och data (datastrukturer)
- Lokalitet av referenser (locality of references):
  - Temporal lokalitet – lokalitet i tid –
    - » om en instruktion/data blivit refererat nu, så är sannolikheten stor att samma referens görs inom kort
  - Rumslokalitet –
    - » om instruktion/data blivit refererat nu, så är sannolikheten stor att instruktioner/data vid adresser i närheten kommer användas inom kort

LUNDS UNIVERSITET

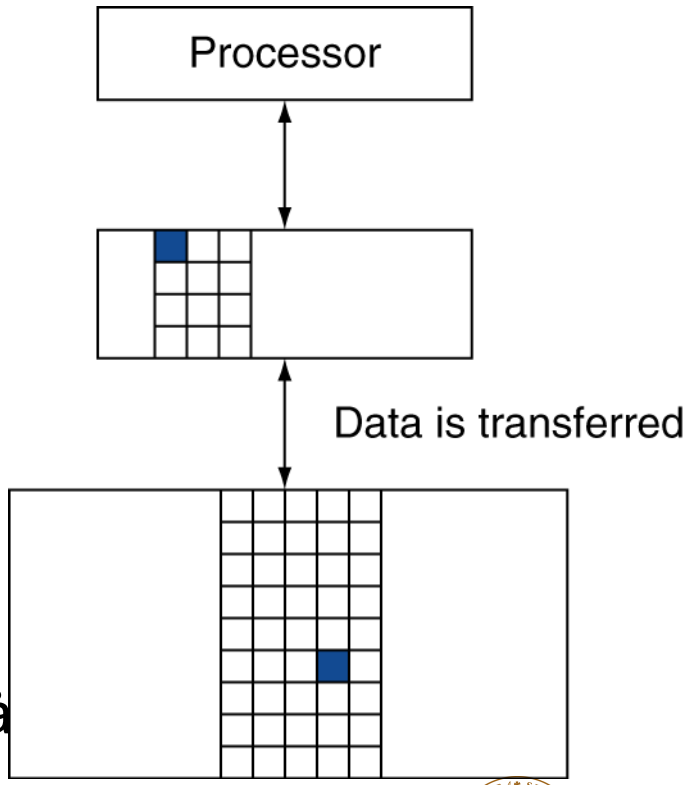
Utnyttja lokalitet

- Minneshierarki
  - Lagra allt på hårddisk
  - Kopiera ”recently accessed (and nearby) items” från disk till mindre primärminne
  - Kopiera mer ”recently accessed (and nearby) items” från primärminne till cacheminne
    - » Cacheminne kopplat till CPU

LUNDS UNIVERSITET

Minneshierarki - nivåer

- Block (line): enhet som kopieras
  - Kan vara flera ”words”
- Om ”accessed data” finns i högsta nivån (upper level)
  - Hit: access ges av högsta nivå
    - » Hit ratio: hits/accesses
- Om ”accessed data” inte finns på aktuell nivå
  - Miss: block kopieras från lägre nivå
  - Tid: miss penalty, Miss ratio: antal missar/accesses = 1 – hit ratio
  - Sedan kan data nås från högre nivå



Exempel: Cacheminne

(1) Processorn läser på adress 22

(2) Data på adress 22 finns ej i cache

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Miss	110

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	**Y**	**10**	**Mem[10110]**
111	N		

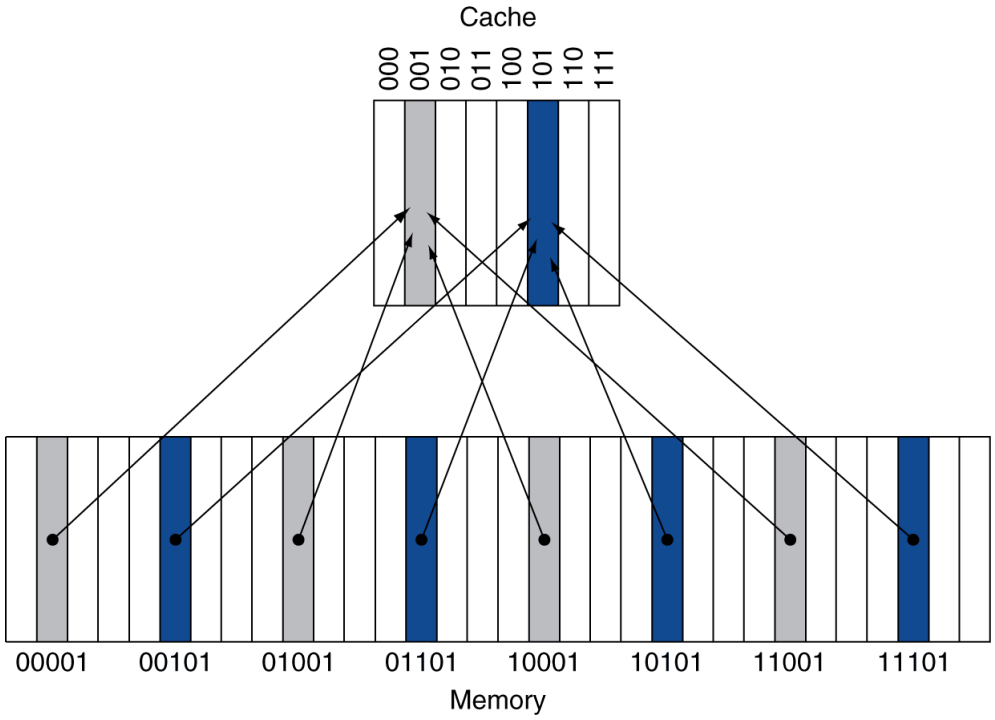
Minnesdata på plats 22

Valid data

LUNDS UNIVERSITET

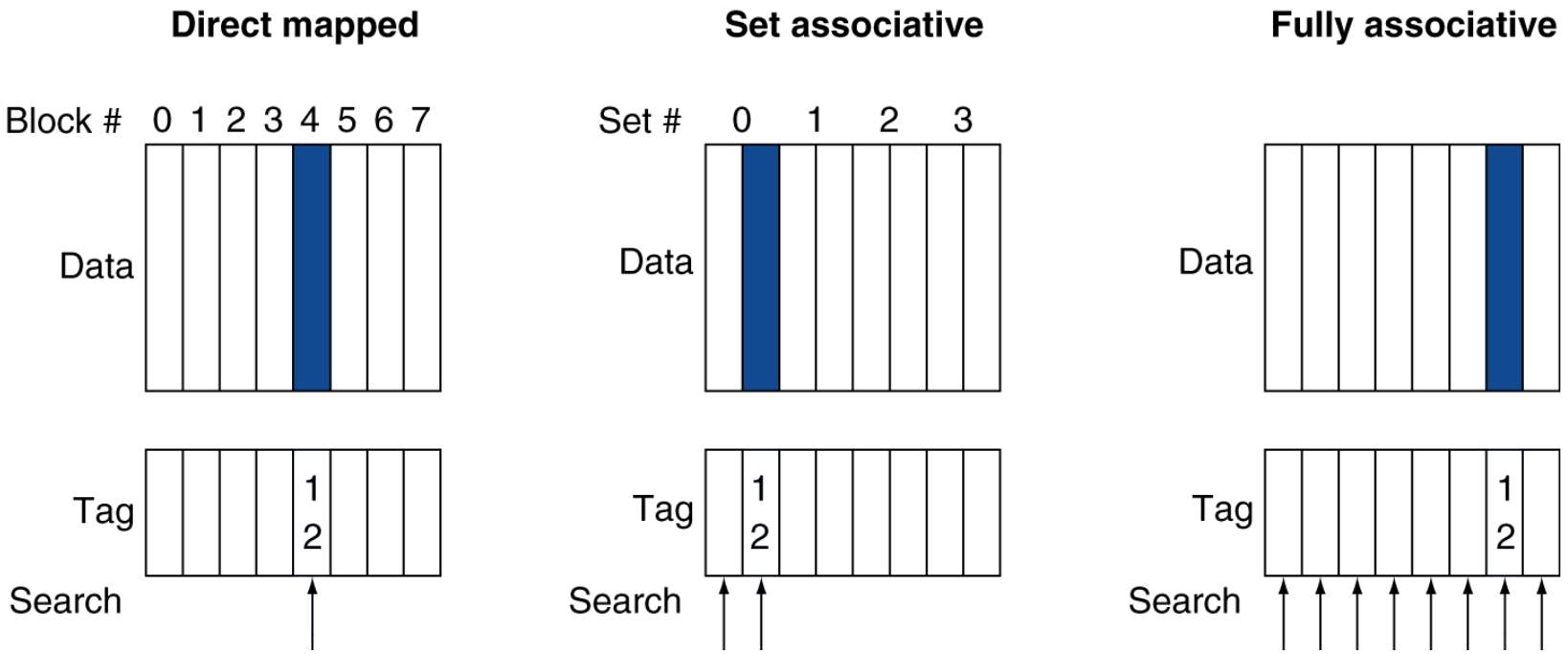
Direktmappad cache

- Primärminnet är mycket större än cacheminnet
- Direktmappning – data i cache på ett ställe



LUNDS UNIVERSITET

Cacheminne



LUNDS UNIVERSITET

Cacheminne

- För ett cacheminne med 8 block

One-way set associative
(direct mapped)

Block	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

Two-way set associative

Set	Tag	Data	Tag	Data
0				
1				
2				
3				

Four-way set associative

Set	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0								
1								

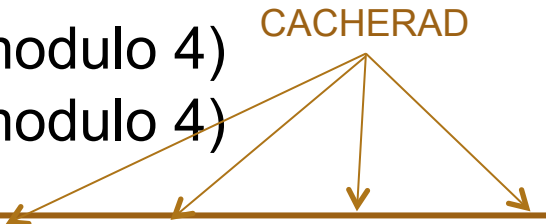
Eight-way set associative (fully associative)

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data

Jämför cacheminnen

- Direct mapped, 2-way set associative, fully associative
- Block access sequence: 0, 8, 0, 6, 8
- Direct mapped:

  Block 0 vill till cache line 0
  Block 8 vill till cache line 0 (8 modulo 4)
  Block 6 vill till cache line 2 (6 modulo 4)



TID

Block address	Cache index	Hit/miss	Cache content after access			
			0	1	2	3
0	0	miss	**Mem[0]**			
8	0	miss	**Mem[8]**			
0	0	miss	**Mem[0]**			
6	2	miss	Mem[0]		**Mem[6]**	
8	0	miss	**Mem[8]**		Mem[6]	

LUNDS UNIVERSITET

Jämför cacheminnen

- Direct mapped, 2-way set associative, fully associative
- Block access sequence: 0, 8, 0, 6, 8
- 2-way set associative:

  Block 0 vill till set 0 (0 modulo 2)
  Block 8 vill till set 0 (0 modulo 2)
  Block 6 vill till set 0 (0 modulo 2)

TID ↓

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	miss	**Mem[0]**			
8	0	miss	Mem[0]	**Mem[8]**		
0	0	hit	**Mem[0]**	Mem[8]		
6	0	miss	Mem[0]	**Mem[6]**		
8	0	miss	**Mem[8]**	Mem[6]		

LUNDS UNIVERSITET

Jämför cacheminnen

- Direct mapped, 2-way set associative, fully associative
- Block access sequence: 0, 8, 0, 6, 8
- Fully associative: Block kan placeras var som helst

TID

Block address		Hit/miss	Cache content after access			
0		miss	**Mem[0]**			
8		miss	Mem[0]	**Mem[8]**		
0		hit	**Mem[0]**	Mem[8]		
6		miss	Mem[0]	Mem[8]	**Mem[6]**	
8		hit	Mem[0]	**Mem[8]**	Mem[6]	

LUNDS UNIVERSITET

Design av cache

- Om cachemiss, hur välja cacherad som ska ersättas?
- Hur hålla minnet konsistent(skrivstrategi)?
- Hur många cacheminnen?
  - Nivåer - Levels (L1, L2, L3)
    - » större cache ger högre hit-rate men är långsammare
  - Unifierad eller separata cacheminnen för instruktioner och data

LUNDS UNIVERSITET

Ersättningsalgoritmer

- Slumpmässigt val – en av kandidaterna väljs slumpmässigt
- Least recently used (LRU) – kandidat är den cacherad vilken varit i cachen men som inte blivit refererad (läst/ skriven) på länge
- First-In First Out (FIFO) – kandidat är den som varit längst i cacheminnet
- Least frequently used (LFU) – kandidat är den cacherad som refererats mest sällan
- Ersättningsalgoritmer implementeras i hårdvara – prestanda viktigt.

LUNDS UNIVERSITET

Skrivstrategier

- Problem: håll minnet konsistent
- Exemplet:

```
x=0;
While (x<1000)
    x=x+1;
```

- Variablen x kommer “finnas” i primärminnet och i cacheminnet
- I primärminnet är x=0 medan i cacheminnet är x=0,1,2… och till sist 1000

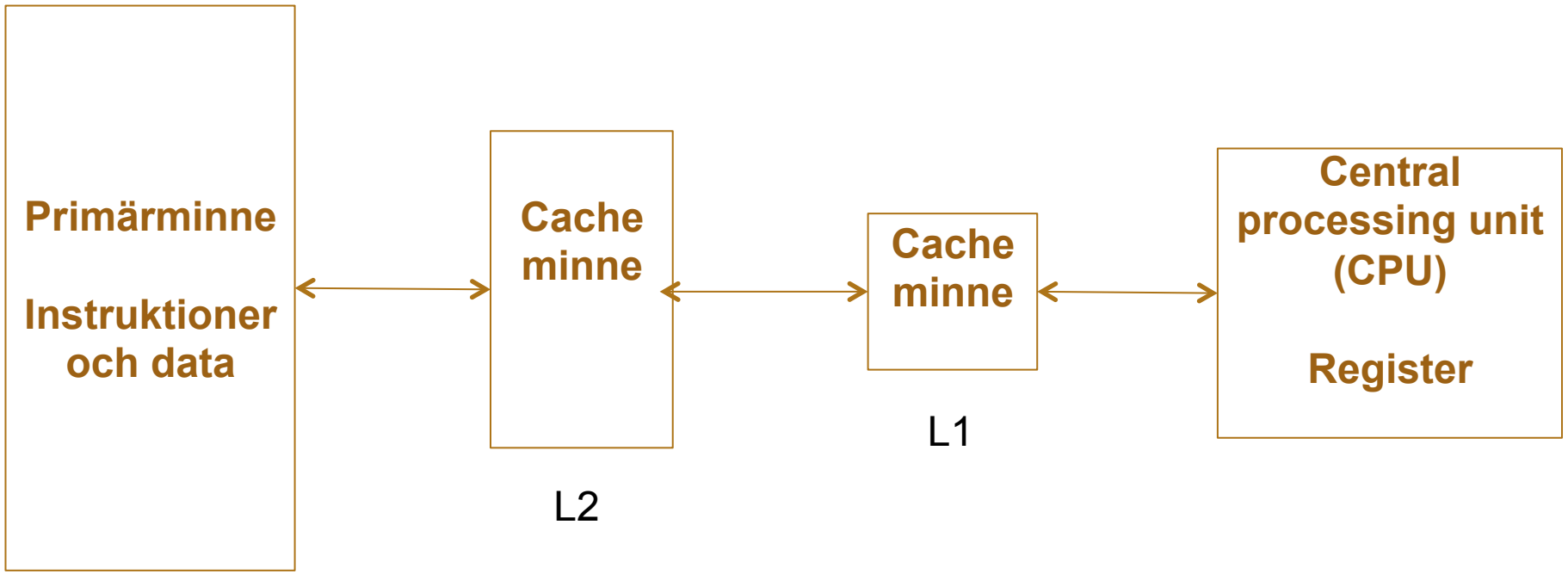
LUNDS UNIVERSITET

Skrivstrategier

- Write-through
  - skrivningar i cache görs också direkt i primärminnet
- Write-through with buffers
  - skrivningar buffras och görs periodiskt
- Write (Copy)-back
  - primärminnet uppdateras först när en cacherad byts ut (ofta används en bit som markerar om en cacherad blivit modifierad (dirty)).
- (Omodifierade cacherader behöver inte skrivas i primärminnet)

LUNDS UNIVERSITET

Antal cachenivåer (levels)



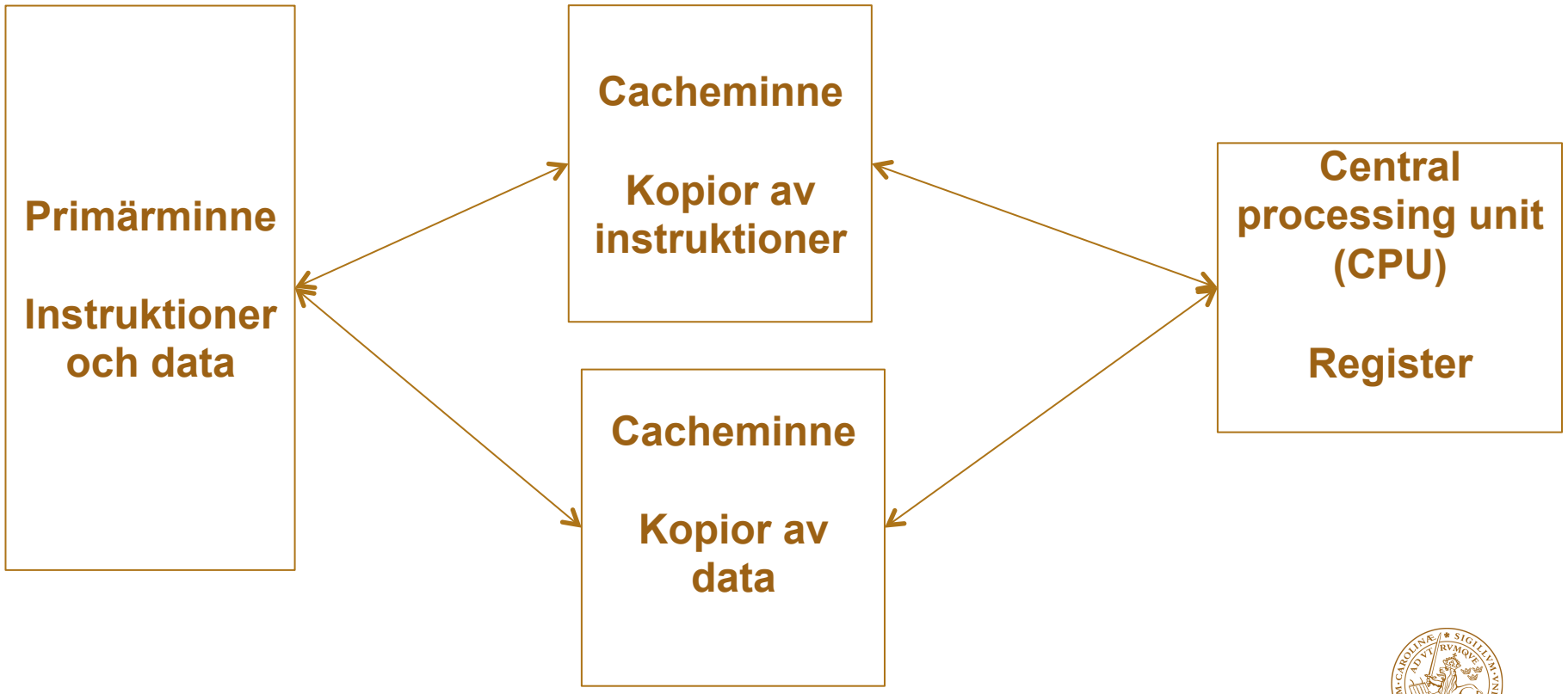
LUNDS UNIVERSITET

Antal cachenivåer (levels)

- Level 1: närmst CPU. Litet och snabbt
- Level 2: större men långsammare (jämfört med Level 1 cache). Level 2 cache “stödjer” missar i Level 1 cache.
- Level 3: större men långsammare (jämfört med Level 2 cache). Level 3 cache “stödjer” missar i Level 2 cache.
- Primärminne: Störst men långsammast.

LUNDS UNIVERSITET

Separat instruktion/data cache



Prestanda

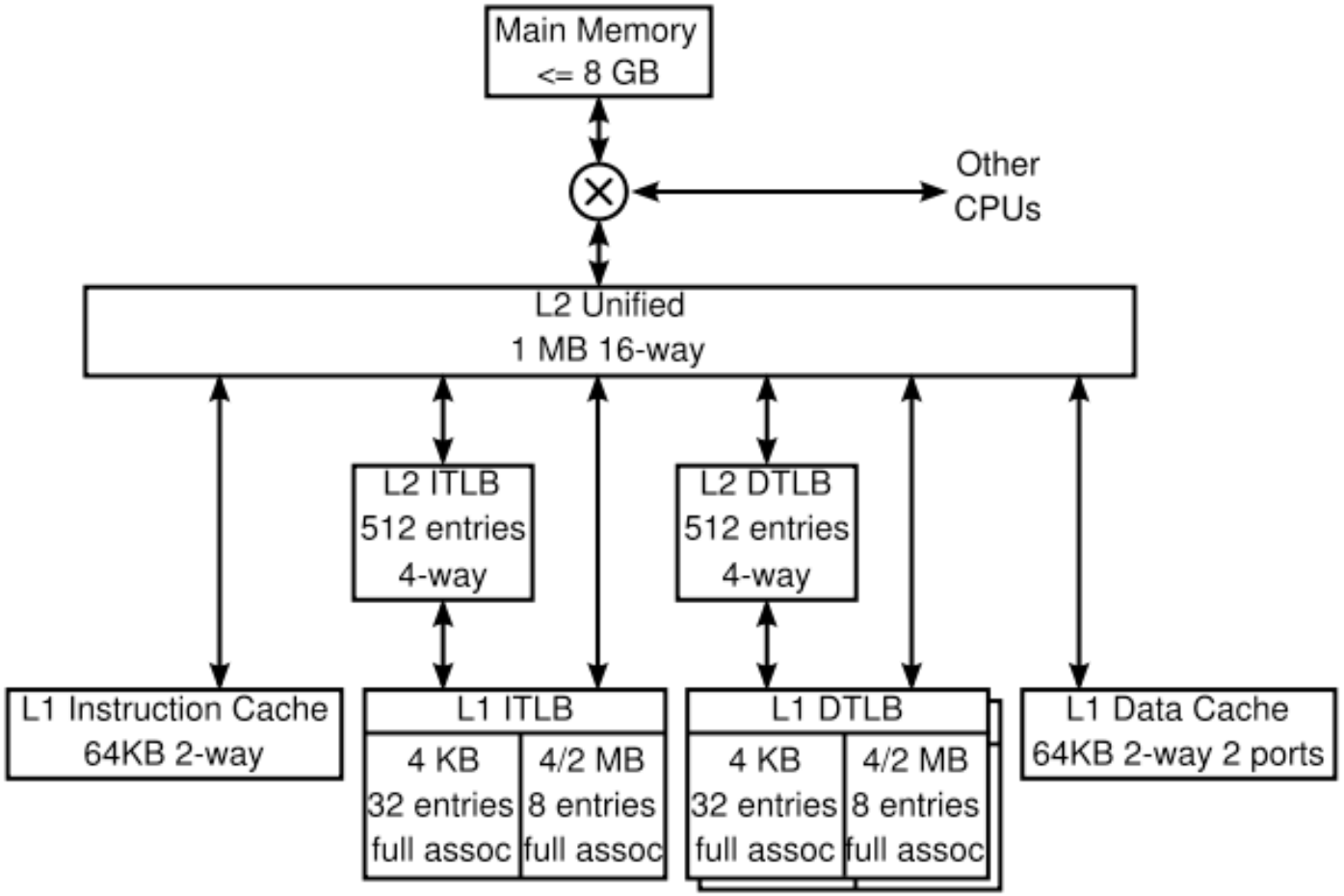
- När CPU prestanda ökar, så blir miss penalty viktig att minimera
- För att undersöka prestanda måste man ta hänsyn till cacheminne
- Cachemissar beror på algoritm(implementation) och kompilatorns optimering

LUNDS UNIVERSITET

Exempel

- PowerPC 603
  - two on-chip caches, for data and instructions, each cache: 8 Kbytes, line size: 32 bytes, 2-way set associative organization, (simpler cache organization than the 601 but stronger processor)
- PowerPC 604
  - two on-chip caches, for data and instructions, each cache: 16 Kbytes, line size: 32 bytes, 4-way set associative organization
- PowerPC 620
  - two on-chip caches, for data and instructions, each cache: 32 Kbytes, line size: 64 bytes, 8-way set associative organization

AMD Athlon 64 CPU

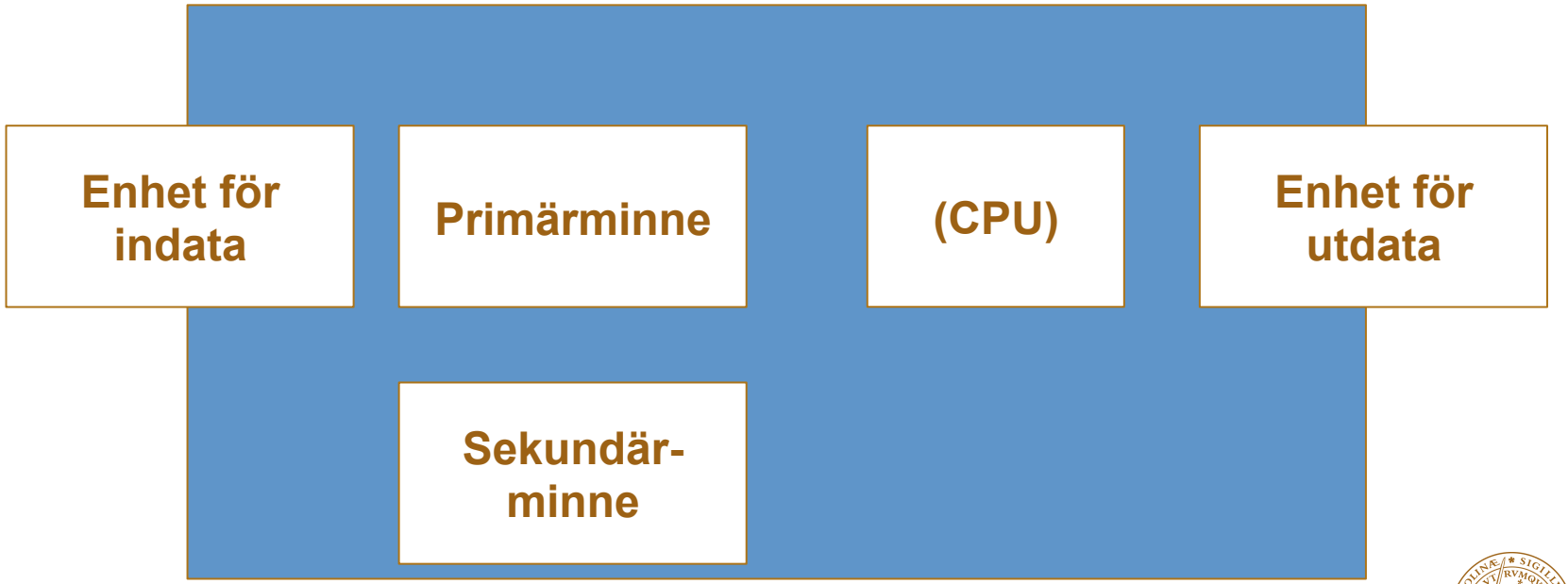


LUNDS UNIVERSITET

Översikt

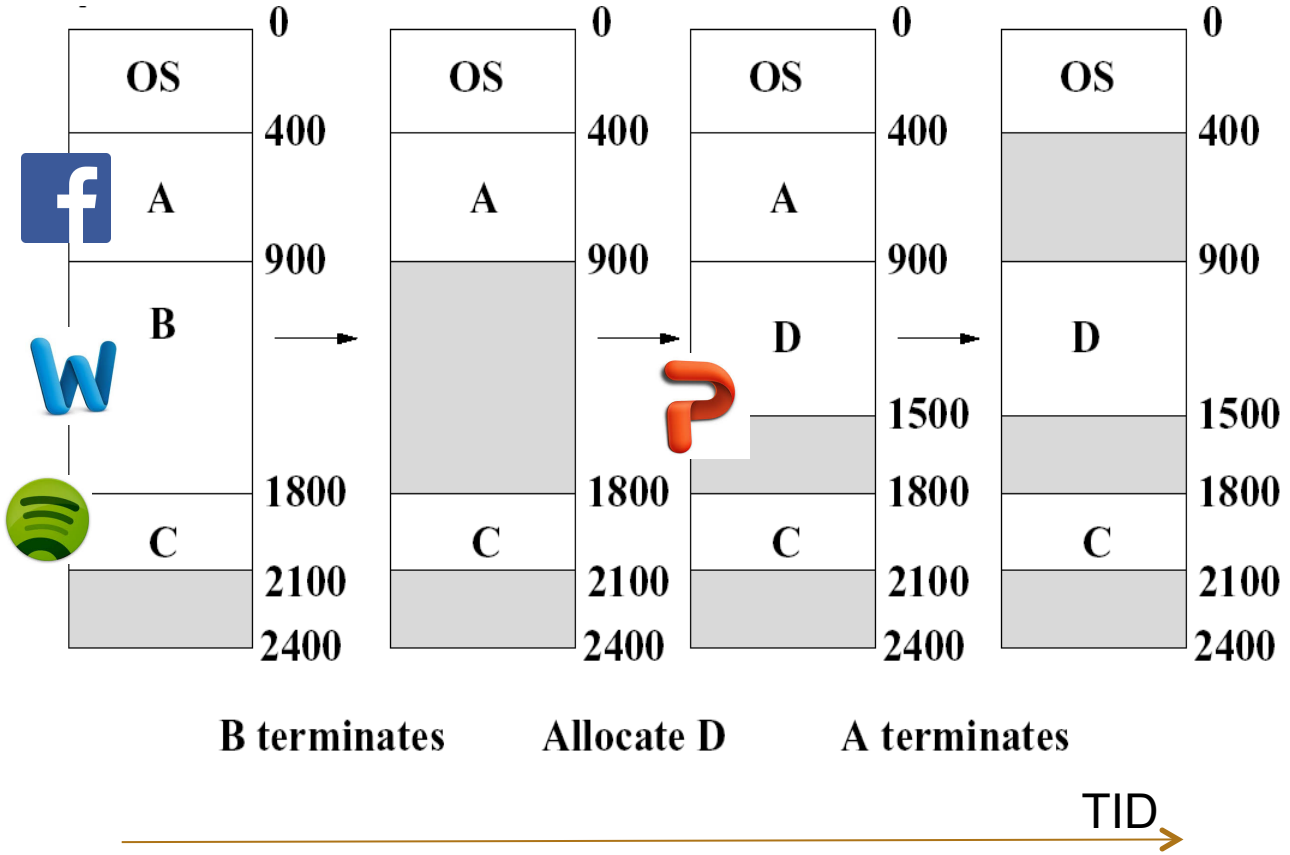
- Minnets komponenter
- Minneshierarkin
- Cacheminne
- Paging
- Virtuellt minne

Minnets komponenter



LUNDS UNIVERSITET

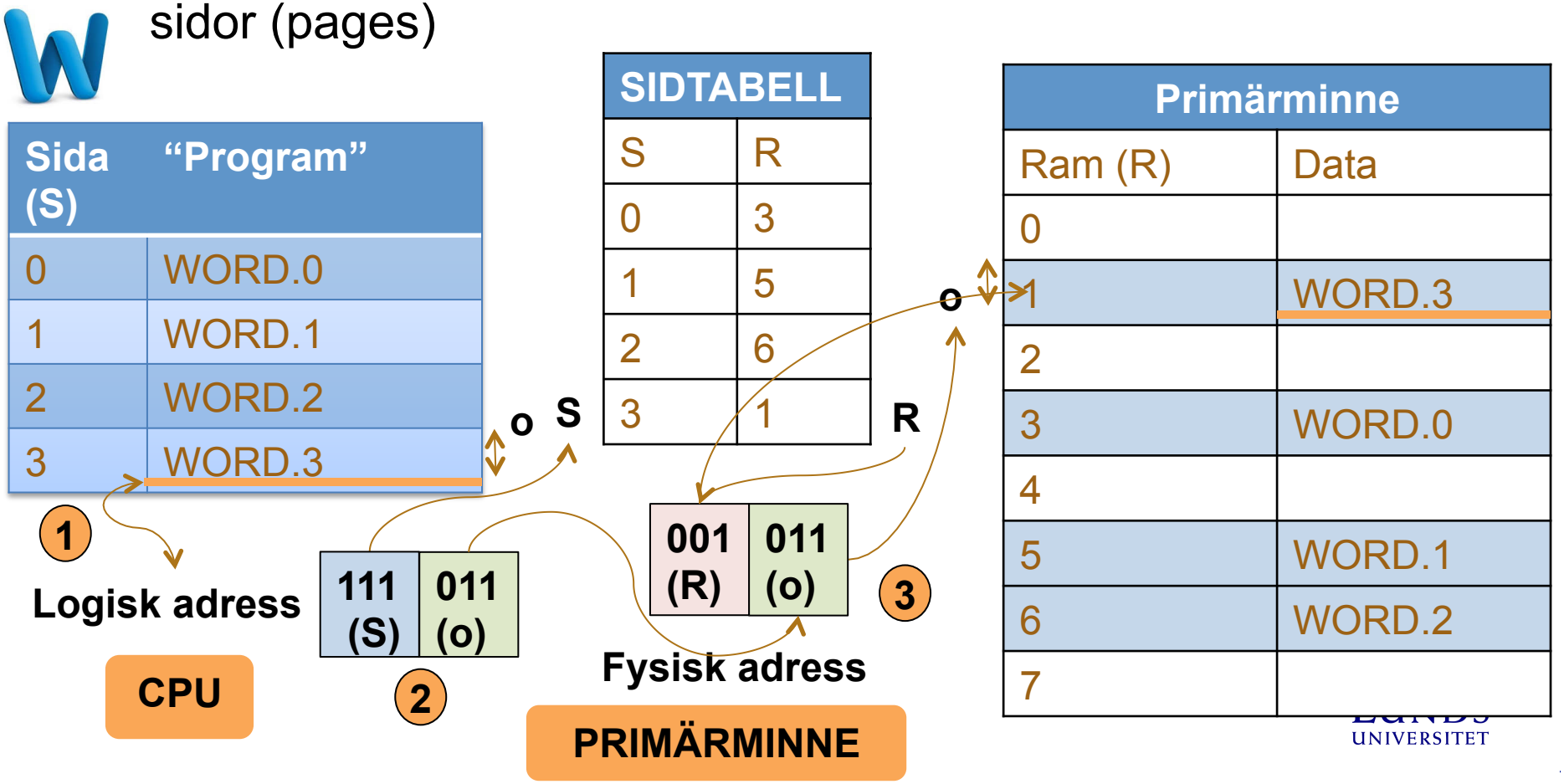
Minnets innehåll över tiden



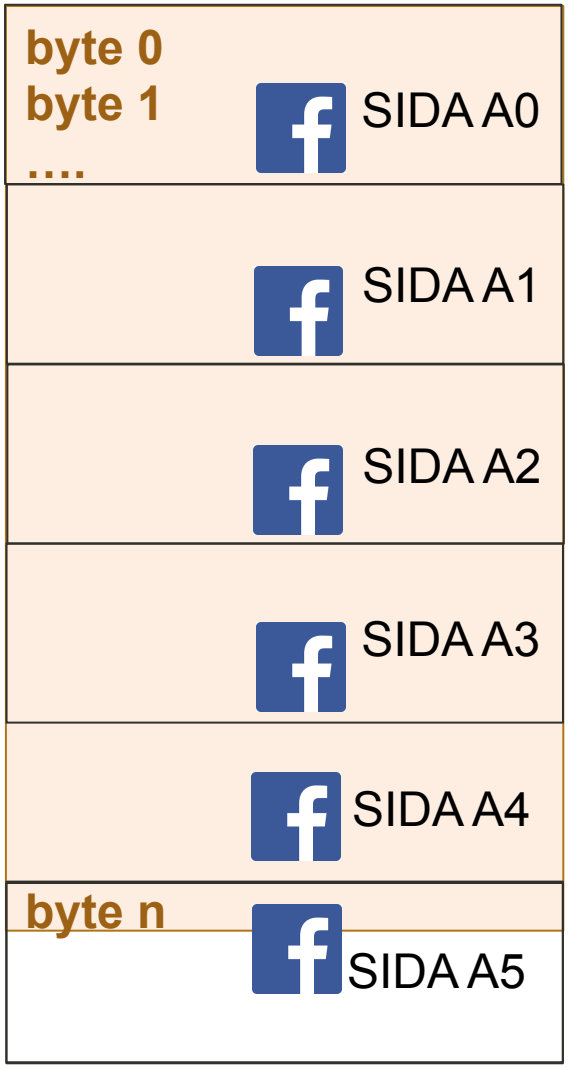
LUNDS UNIVERSITET

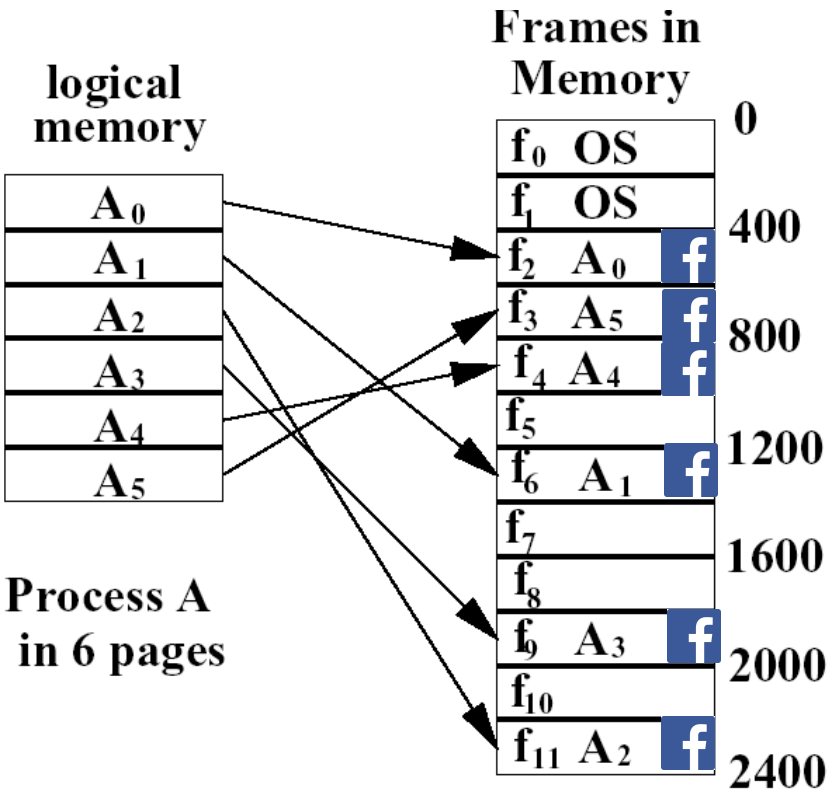
Paging

- Dela upp primärminnet i ramar (frames) och program i sidor (pages)



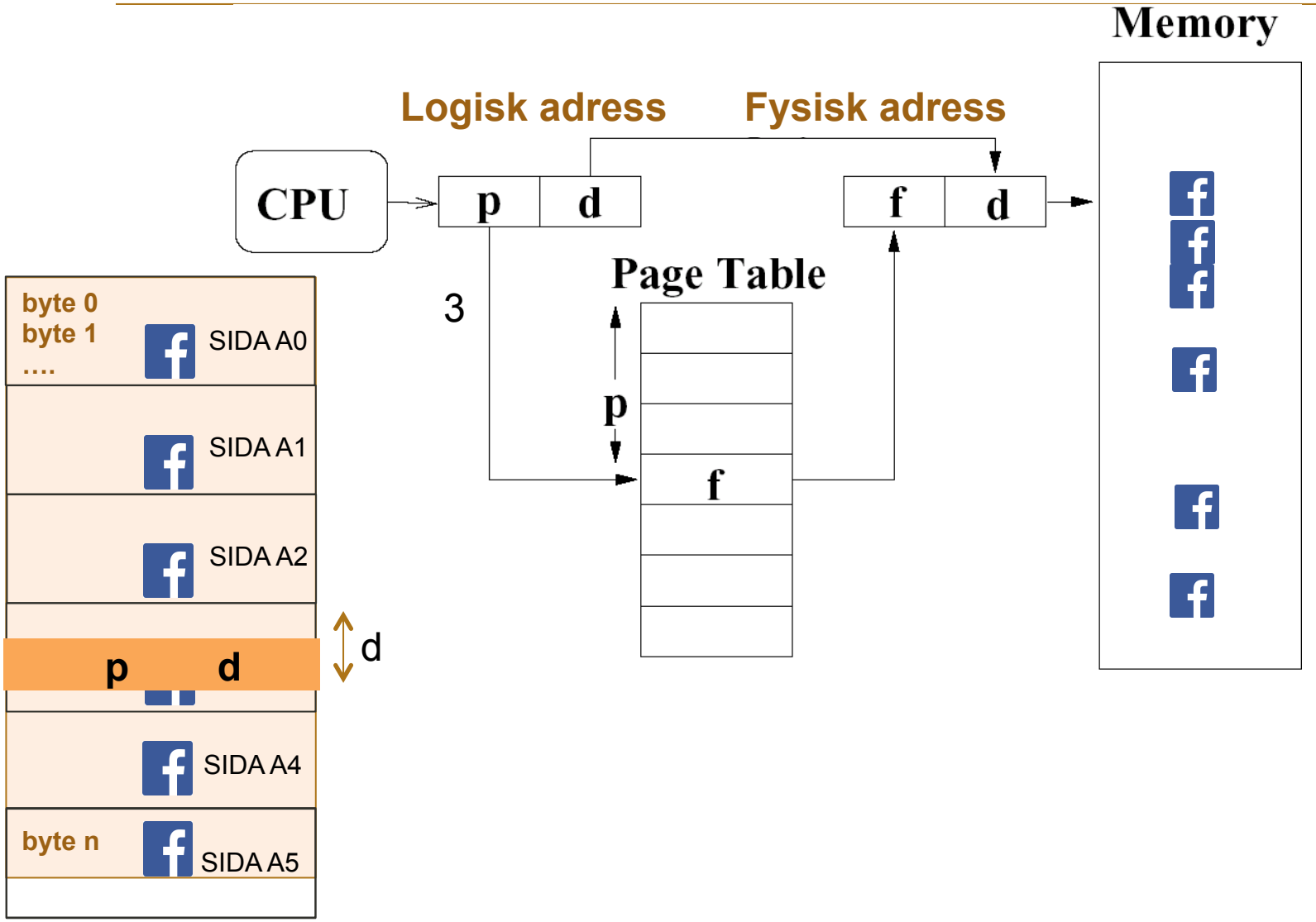
Paging





LUNDS UNIVERSITET

Paging



LUNDS UNIVERSITET

Översikt

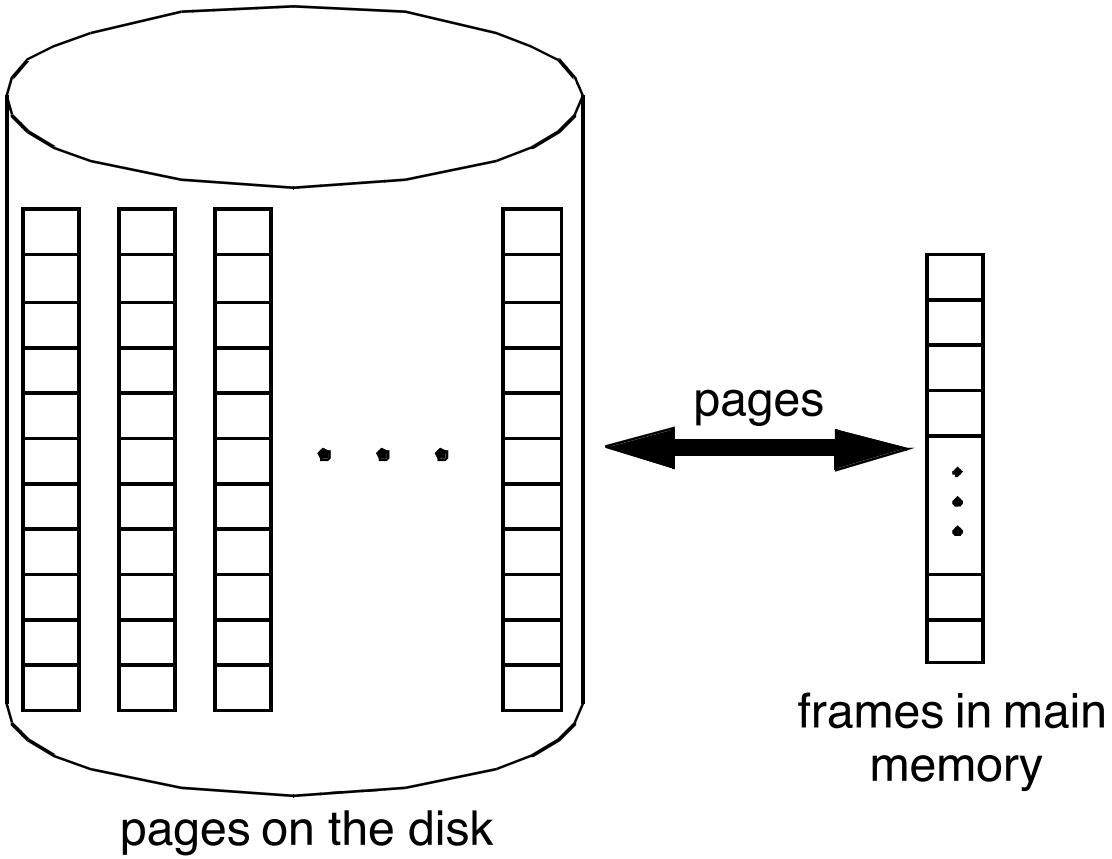
- Minnets komponenter
- Minneshierarkin
- Cacheminne
- Paging
- Virtuellt minne

Virtuellt minne

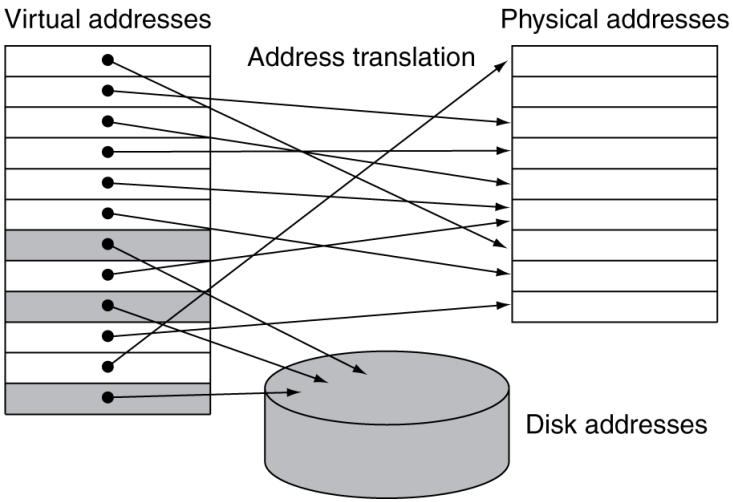
- Använd primärminne som “cache” för sekundärminne (hårddisk)
  - Hanteras med hårdvara och operativsystem
- Program delar primärminnet
  - Varje program får sin virtuella adressrymd
  - Skyddas från andra program
- CPU och OS översätter virtuella adresser till fysiska adresser
  - Ett “block” kallas för sida (page)
  - “Miss” kallas för sidfel (page fault)

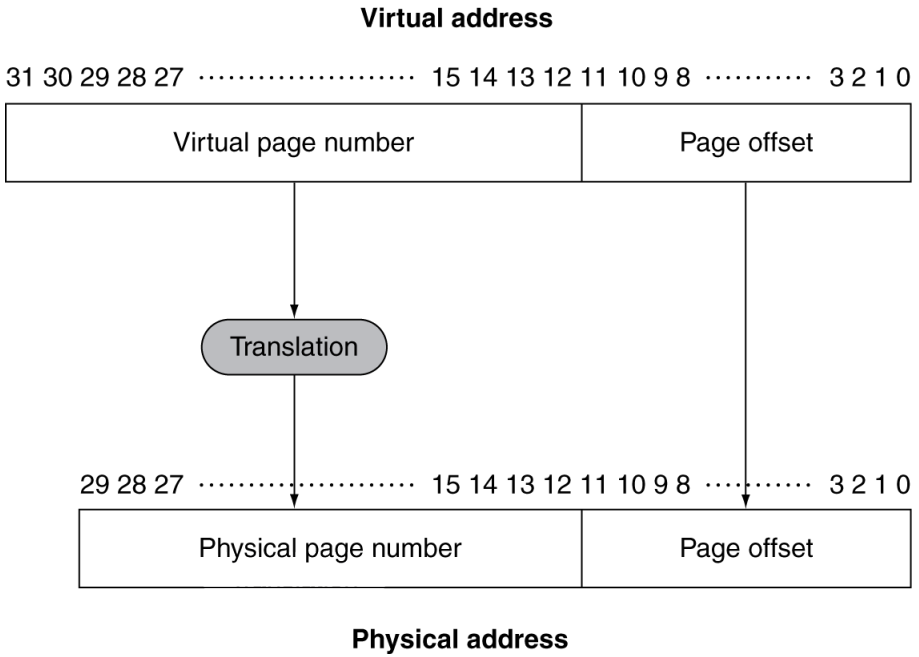
LUNDS UNIVERSITET

Virtuellt minne



Adressöversättning





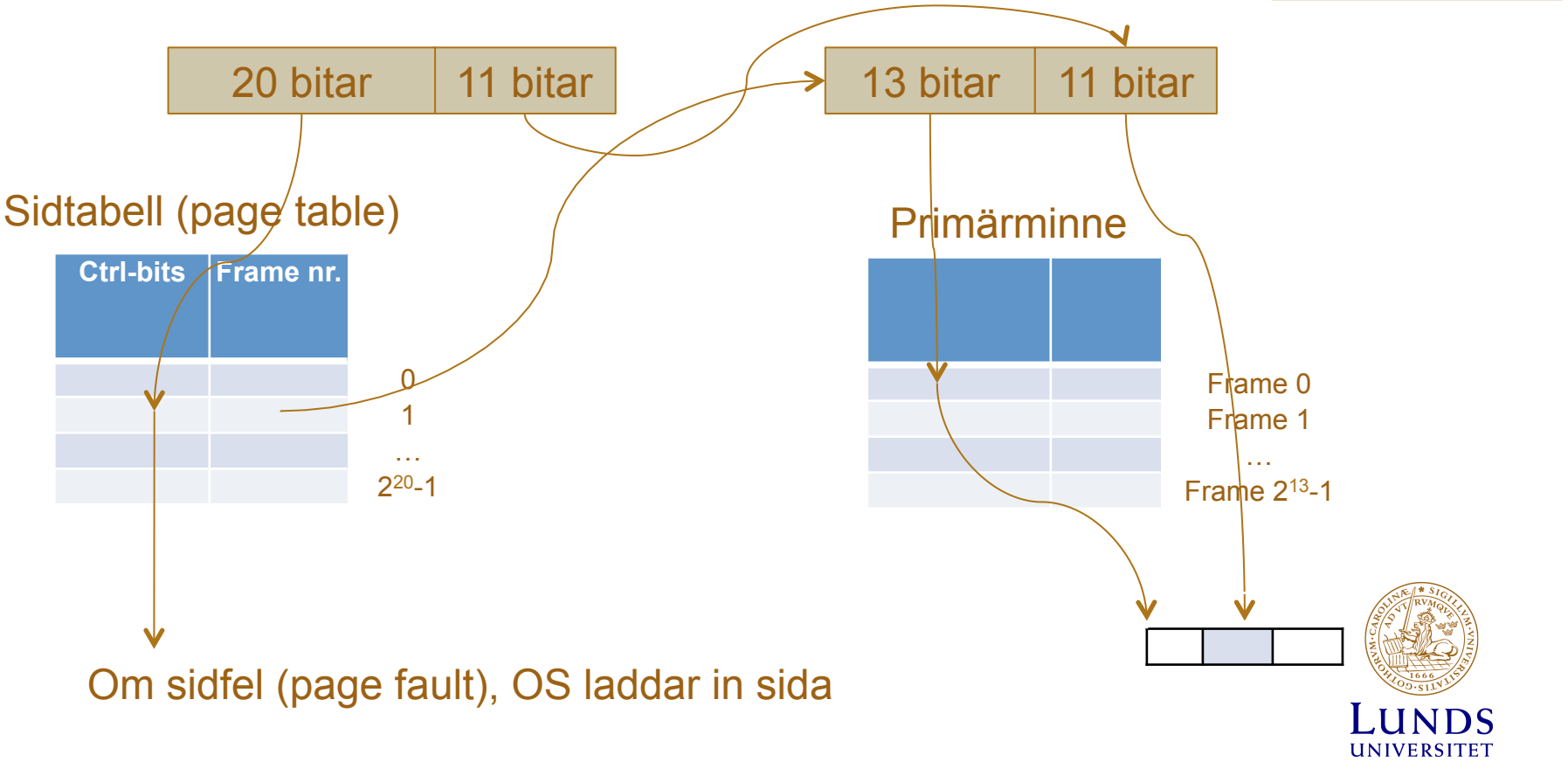
LUNDS UNIVERSITET

Virtuellt minne

Memory Management Unit (MMU)

- Virtuell adress: 31 bitar

Fysisk adress: 24 bitar



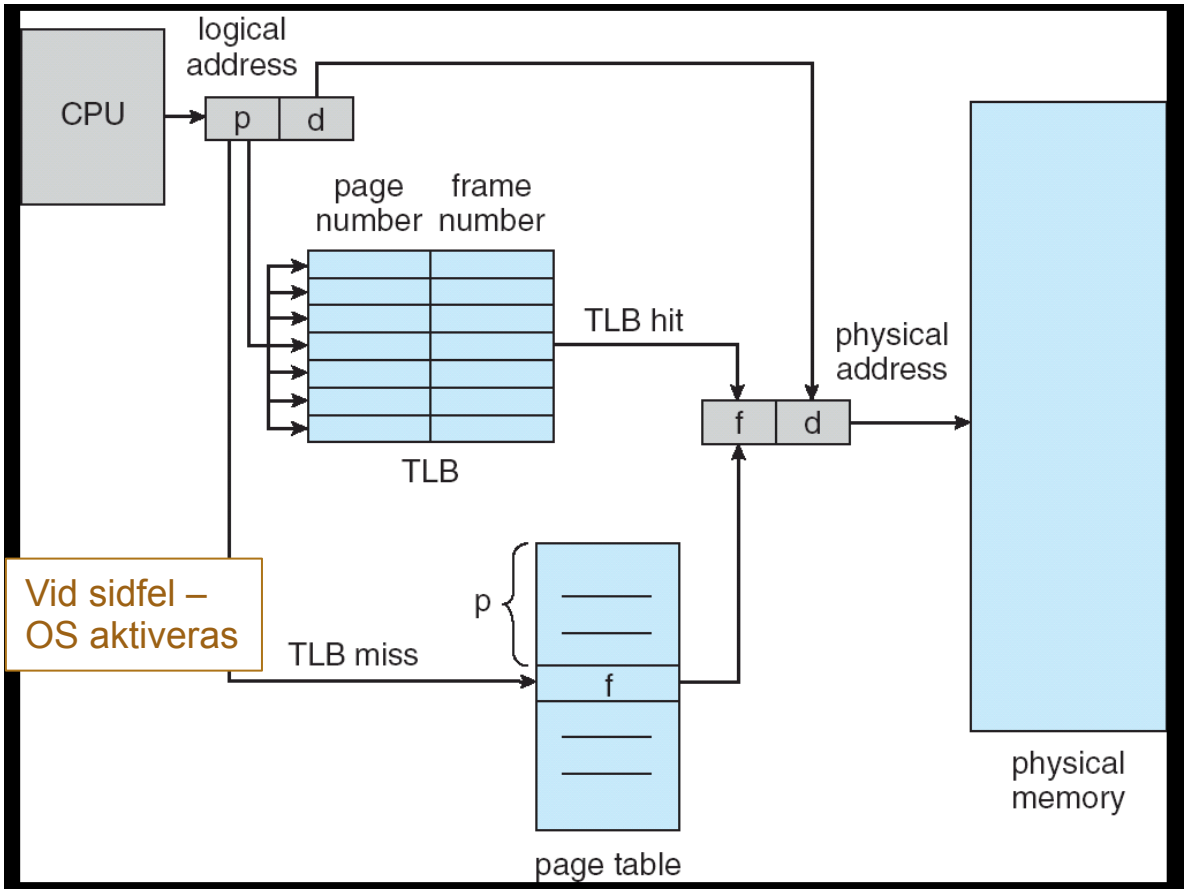
LUNDS UNIVERSITET

Virtuellt minne

- Problem med sidtabell
  - Tid vid läsning av adress:
    - » 1 läs sidtabell
    - » 2 läs data
  - Stora sidtabeller
- Använd cache - Translation Look-Aside Buffer (TLB) – för sidtabeller

LUNDS UNIVERSITET

Translation Look-Aside Buffer (TLB)



Sammanfattning

- Snabba minnen är små, stora minnen är långsamma
  - Vi vill ha snabba och stora minnen
  - Cacheminnen och virtuellt minne ger oss den illusion
- Lokalitet viktigt för att cacheminnen och virtuellt minne ska fungera
  - Program använder vid varje tidpunkt en liten del av sitt minne ofta
- Minneshierarki
  - L1 cache <->L2 cache …. Primärminne - Sekundärminne

LUNDS UNIVERSITET

LUNDS
UNIVERSITET