

TDIU11 – Föreläsning 4

Minneshantering (Memory management)

Filip Strömbäck

Planering

Vecka	Föreläsning	Seminarie
4	Processer och trådar	—
5	Filsystem och lagring	Utmaning 1: Schemaläggning
6	Minneshantering	Rapport 1: Filsystem I
7	Virtuellt minne	Utmaning 2: Filsystem
8	Säkerhet	Utmaning 3: Virtuellt minne
9	Repetition/utblickar	Rapport 2: Filsystem II
10	—	Utmaning 4: Säkerhet
11	Tentaförberedelse	Utmaning 5: Repetition (tisdag)
12	(omtenta-p)	Rapport: Uppsamling

- 1 Minneshantering
- 2 Strategier: Kontinuerlig
- 3 Strategier: Segmentering
- 4 Strategier: Paging
- 5 Alternativ lagring av page table
- 6 Nästa vecka

Mål

Process 1

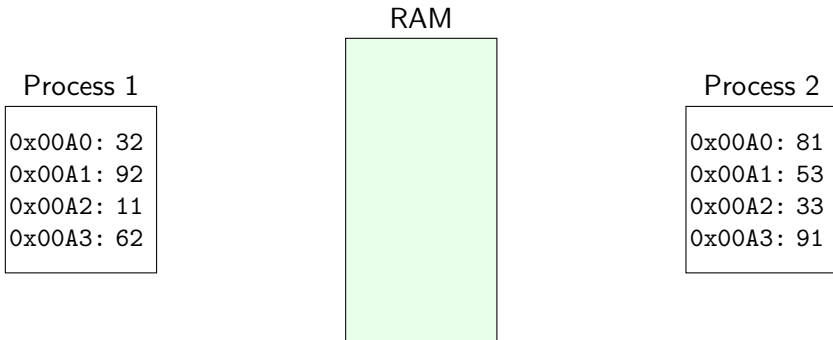
0x00A0:	32
0x00A1:	92
0x00A2:	11
0x00A3:	62

Hur går detta ihop?

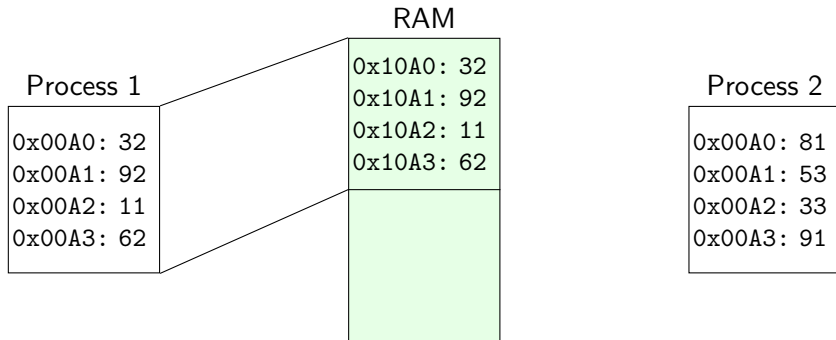
Process 2

0x00A0:	81
0x00A1:	53
0x00A2:	33
0x00A3:	91

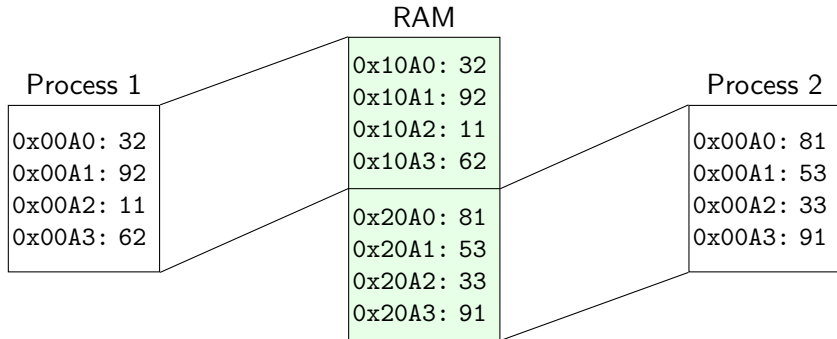
Mål



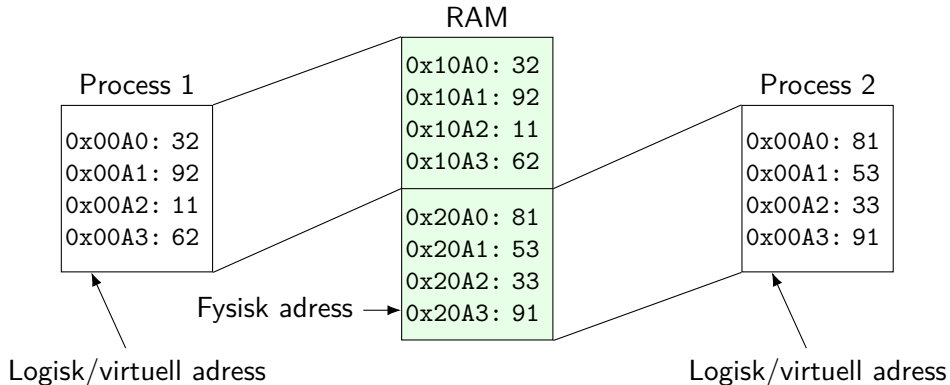
Mål



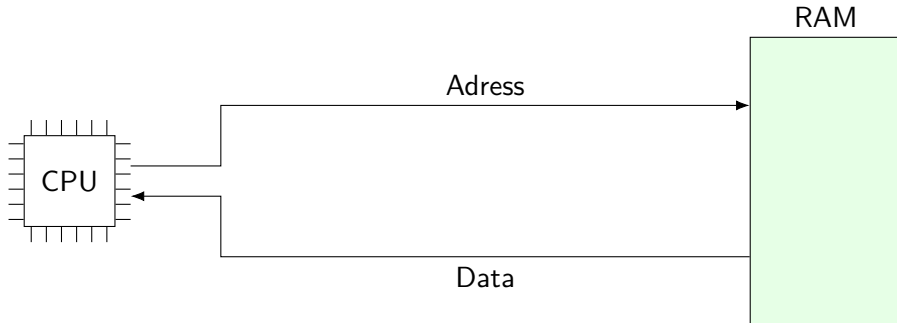
Mål



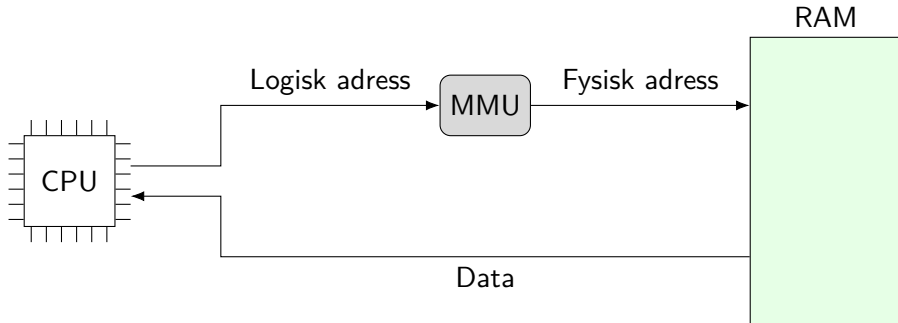
Mål



Hur löser vi detta?



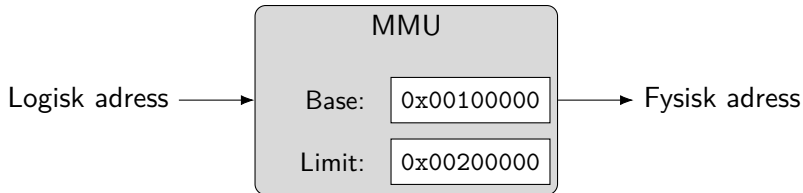
Hur löser vi detta?



Notera: logisk och fysisk adress kan vara olika stora!

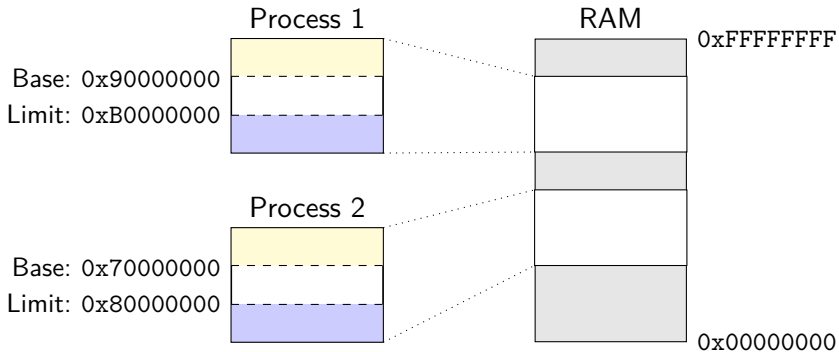
- 1 Minneshantering
- 2 **Strategier: Kontinuerlig**
- 3 Strategier: Segmentering
- 4 Strategier: Paging
- 5 Alternativ lagring av page table
- 6 Nästa vecka

Kontinuerlig (Continuous allocation)

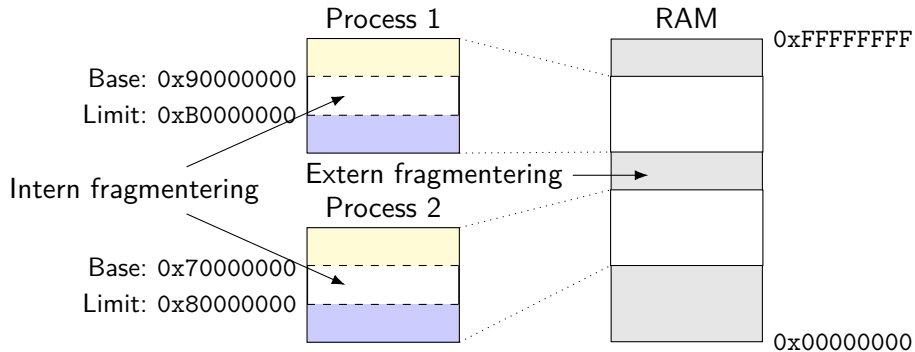


Notera: Varje process har olika MMU-inställningar!

Kontinuerlig (Continuous allocation)

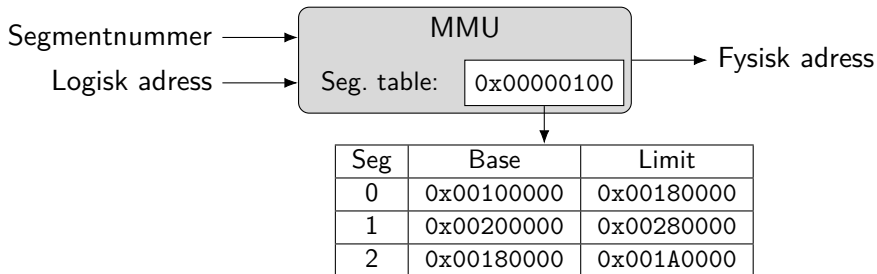


Kontinuerlig (Continuous allocation)



- 1 Minneshantering
- 2 Strategier: Kontinuerlig
- 3 **Strategier: Segmentering**
- 4 Strategier: Paging
- 5 Alternativ lagring av page table
- 6 Nästa vecka

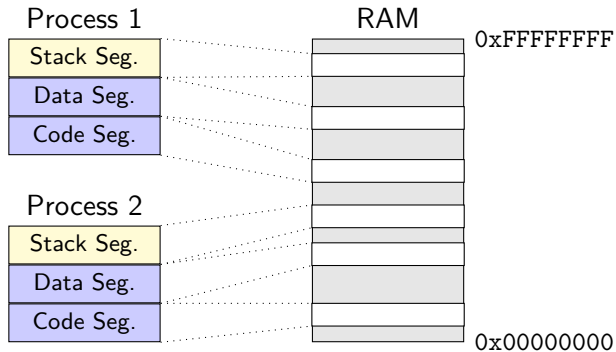
Segmentering



Mindre block $\Rightarrow$ Mindre problem med fragmentering

Notera: Varje process har olika segmenttabeller!

Segmentering

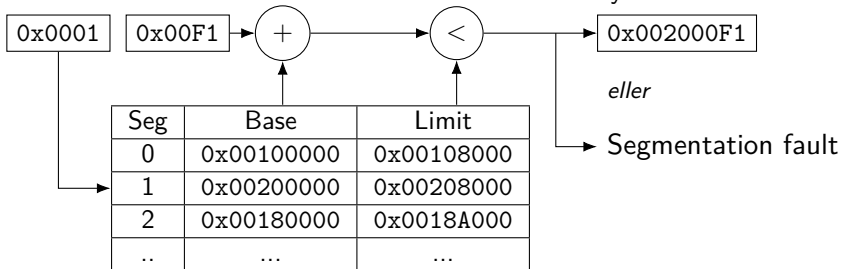


Segmentering: Hur ser logiska adresser ut?

Alternativ 1: Segmentregister (t.ex. Protected Mode i Intel x86)

`mov ds:[0x00F1], 0x4F` DS: 1

Logisk adress:



Segmentering: Hur ser logiska adresser ut?

Alternativ 2: Mest signifikanta bitar i logisk adress

```
mov [0x000100F1], 0x4F
```

Logisk adress:

0x0001 | 00F1

+

<

Fysisk adress:

0x002000F1

eller

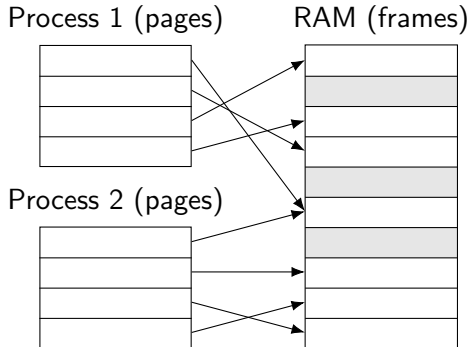
Segmentation fault

Seg	Base	Limit
0	0x00100000	0x00108000
1	0x00200000	0x00208000
2	0x00180000	0x0018A000
..	...	...

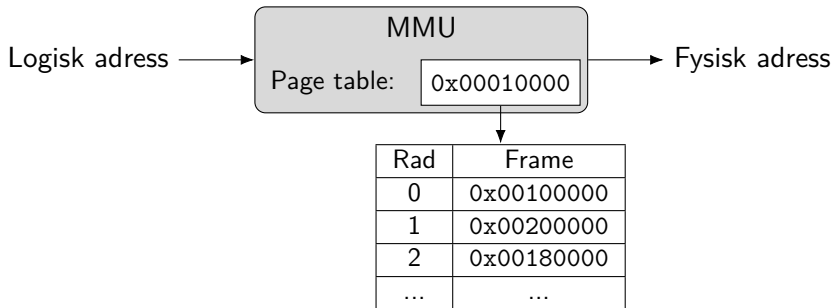
- 1 Minneshantering
- 2 Strategier: Kontinuerlig
- 3 Strategier: Segmentering
- 4 **Strategier: Paging**
- 5 Alternativ lagring av page table
- 6 Nästa vecka

Paging

- Vi vill undvika *extern fragmentering*
 - Idé: Vi delar upp den virtuella adressrymden i *pages*
 - Alla *pages* har *samma storlek*
- ⇒ Vi kan lagra en *page* var som helst i RAM (i vilken *frame* som helst)
- ⇒ Bara *intern fragmentering* kvar!
- Hur stor ska en *page* vara?



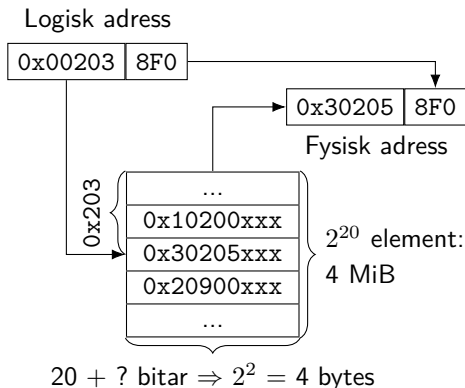
Paging



Olika page tables för olika processer!

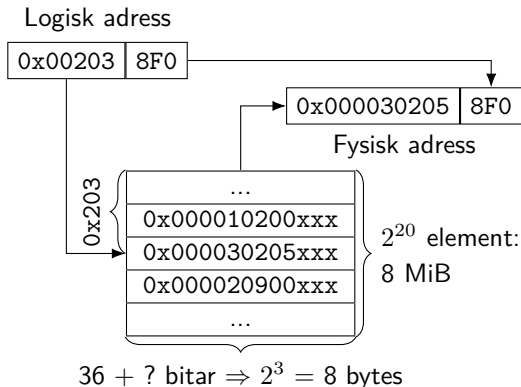
Paging: Hur stor blir page-table?

- 32-bit logiska adresser
- 32-bit fysiska adresser
- 4 KiB page och frame
- Page-table får inte plats i en page $\Rightarrow$ Fragmentering!
- Kan vi välja en bra page-size?



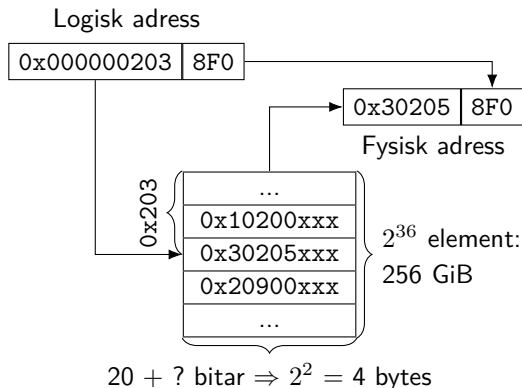
Paging: Hur stor blir page-table?

- 32-bit logiska adresser
- 48-bit fysiska adresser
- 4 KiB page och frame
- Page-table får inte plats i en page $\Rightarrow$ Fragmentering!
- Kan vi välja en bra page-size?



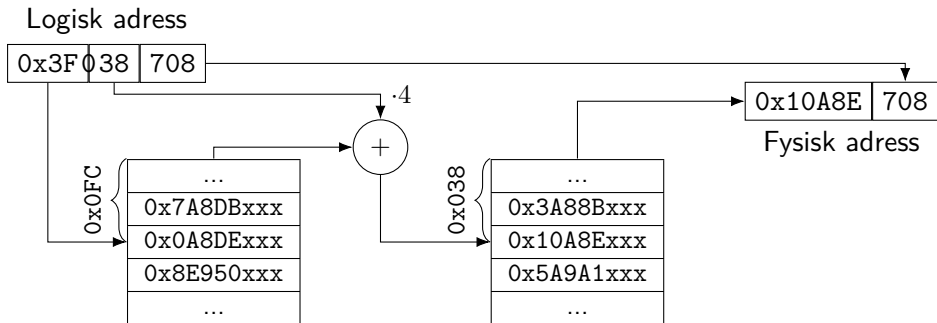
Paging: Hur stor blir page-table?

- 48-bit logiska adresser
- 32-bit fysiska adresser
- 4 KiB page och frame
- Page-table får inte plats i en page $\Rightarrow$ Fragmentering!
- Kan vi välja en bra page-size?



Hierarkisk Page-table

Idé: Vi lagrar page-table i flera nivåer!



Kostnad av paging

Med 2 nivåer paging:

- Kostar 2 läsningar att slå upp adress
- Varje minnesaccess kostar som 3
- RAM är förhållandevis långsamt...

Lösning:

- TLB (Translation Lookaside Buffer)
- Cache för MMU-beräkningar (64–1024 element)
- Undviker minnesåtkomst om elementet finns

Kostnad av paging

Effective Access Time:

Antag:

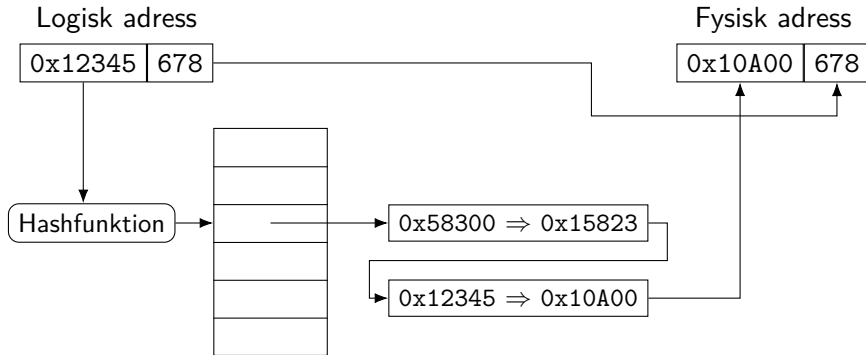
- α = hit-ratio, i %
- t = RAM-access
- ε = uppslagning i TLB
Litet i förhållande till t

$$\text{EAT} = (t + \varepsilon)\alpha + (3t + \varepsilon)(1 - \alpha)$$

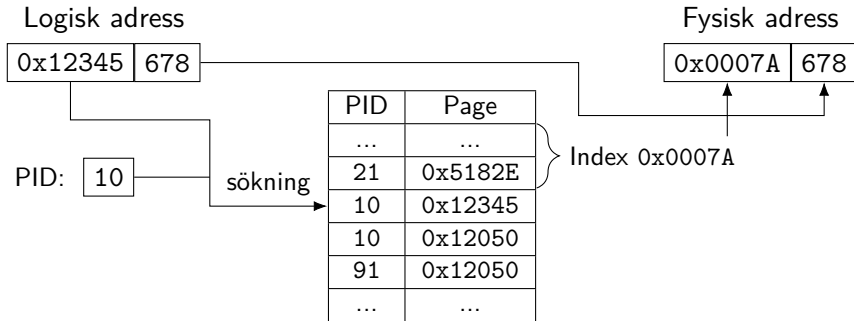
- $\alpha = 80\%$, $t = 100$ ns ger $\text{EAT} = 120$ ns
- $\alpha = 99\%$, $t = 100$ ns ger $\text{EAT} = 101$ ns

- 1 Minneshantering
- 2 Strategier: Kontinuerlig
- 3 Strategier: Segmentering
- 4 Strategier: Paging
- 5 **Alternativ lagring av page table**
- 6 Nästa vecka

Hashad Page-table

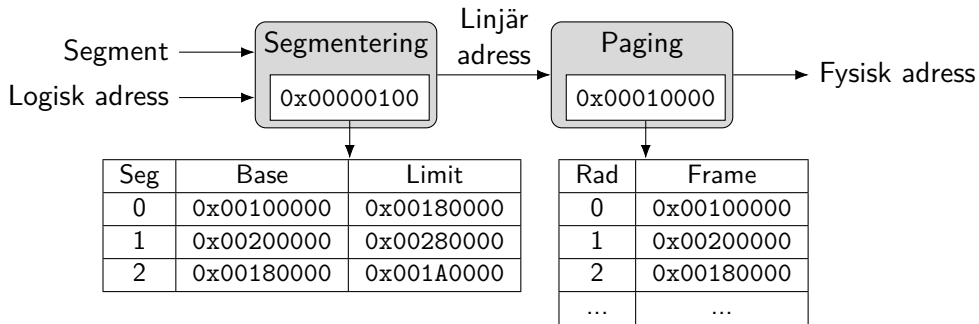


Inverterad Page-table



PT har ett element per *frame*. Svårt att dela minne mellan processer.

Segmentering + Paging (Intel x86)



- 1 Minneshantering
- 2 Strategier: Kontinuerlig
- 3 Strategier: Segmentering
- 4 Strategier: Paging
- 5 Alternativ lagring av page table
- 6 **Nästa vecka**

Nästa vecka

På 64-bitarssystem har vi 256 TiB logisk adressrymd, men mycket mindre RAM.

- Varför är det användbart?
- Hur kan vi använda mer minne än vad som finns tillgängligt?

Filip Strömbäck

www.liu.se