Deadlock
Kriterier for at deadlock kan oppstå
Deadlock
Dining Philosophers Problem
Mulige løsninger for deadlock
Internminne

Kriterier for at deadlock kan oppstå

1. Mutex: ressurser som ikke kan deles
2. En prosess kan beholde sine ressurser mens den venter på andre.
3. En prosess kan ikke tvinges til å gi opp sin ressurs (felles minne, disk, etc.)

Med 1, 2 og 3 oppfylt, kan deadlock oppstå ved sirkulær venting!

Deadlock
Kriterier for at deadlock kan oppstå
Deadlock
Dining Philosophers Problem
Mulige løsninger for deadlock
Internminne

Deadlock

To eller fler prosesser venter på hverandre, ingen kommer videre.
Eks. 1: P1 venter på P2, P2 venter på P3 og P3 venter på P1 (sirkulær venting)
Eks. 2: Deadlock med to semaforer S1 og S2, initialisert til 1:

PA	PB-kode
Wait(S1)	
	Wait(S2)
Wait(S2)	
	Wait(S1)
.	.
.	.
.	.
Signal(S1)	Signal(S2)
Signal(S2)	Signal(S1)

Deadlock
Kriterier for at deadlock kan oppstå
Deadlock
Dining Philosophers Problem
Mulige løsninger for deadlock
Internminne

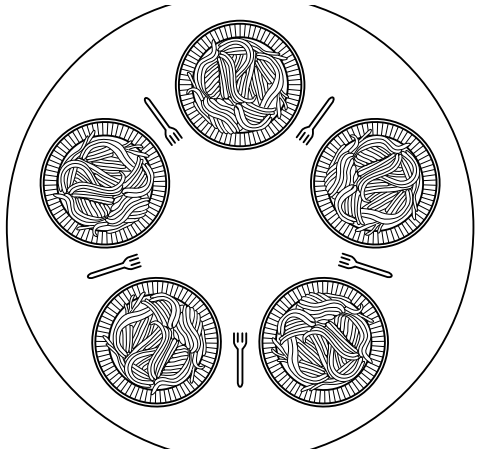
Dining Philosophers Problem

Filosoftilstander:

1. tenker (uten gafler)
2. Spiser spaghetti med 2 gafler

Tar opp en gaffel av gangen.

Problem Programmer en filosofprosess slik at 5 prosesser kan spise og tenke i tidsrom av varierende lengde og dele ressursene (gaflene) **uten** at deadlock (vranglås) kan oppstå.



Deadlock
Kriterier for at deadlock kan oppstå
Deadlock
Dining Philosophers Problem
Mulige løsninger for deadlock
Internminne

Mulige løsninger for deadlock

1. Forhindre. Internt i OS-kjernen må deadlock forhindres. Umulig å forhindre bruker-deadlock.
2. Løse opp deadlock. Generelt vanskelig.
3. Ignorere problemet (mest vanlig metode for et OS).

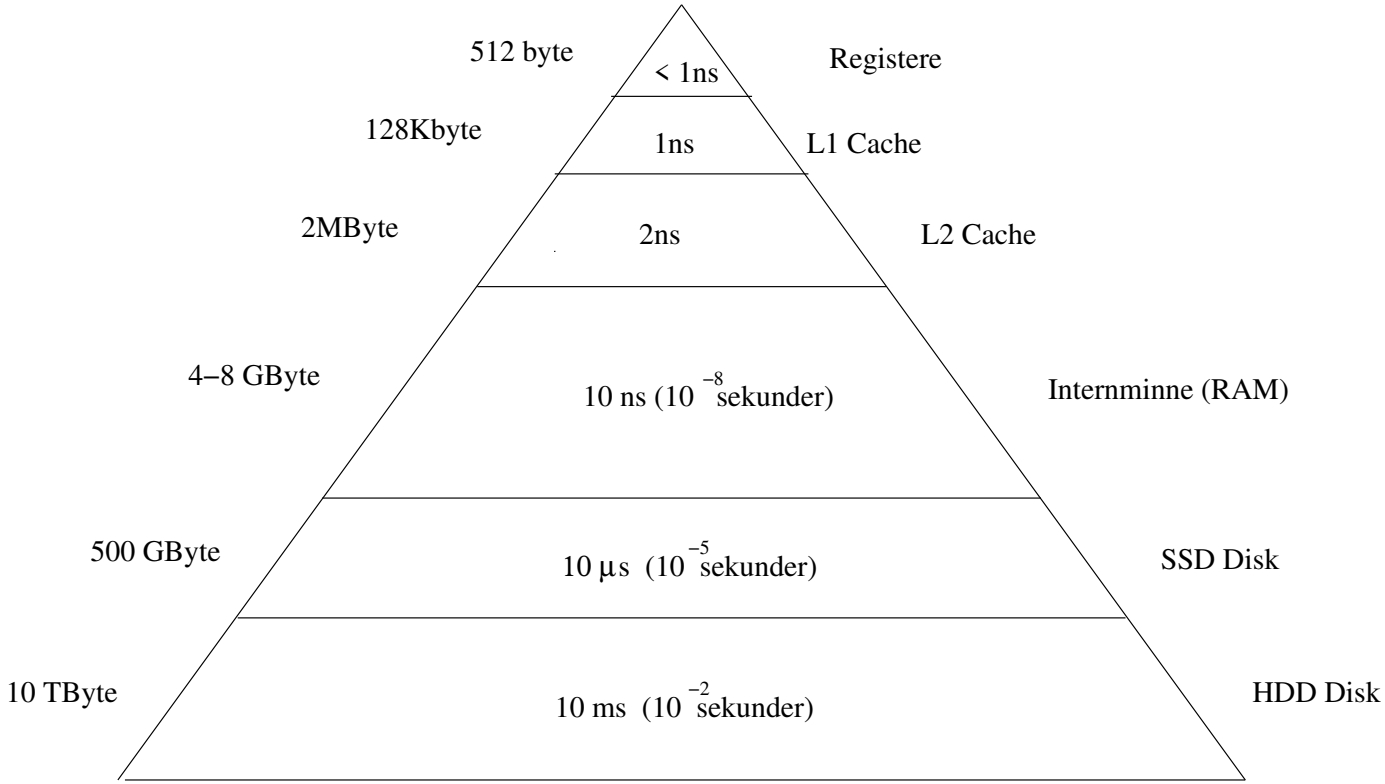
Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

Internminne og Cache

- RAM = Random Access Memory
- CPU-registere og cache er laget av SRAM (Static RAM)
- SRAM består av 6 transistorer, er meget hurtig og statisk (trenger ikke å oppfriskes)
- Internminnet er laget av DRAM (Dynamic RAM), består av en transistor og en kondensator, må oppfriskes 10 ganger pr sekund
- Synchronous DRAM (SDRAM), er DRAM hvor lesing og skriving er synkronisert med en ekstern klokke.
- DDR4 (Double Data Rate) SDRAM (2133 - 4000 Mhz)
- Nyeste: DDR5 kom i juni 2020 (4800 - 5600 Mhz)

Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

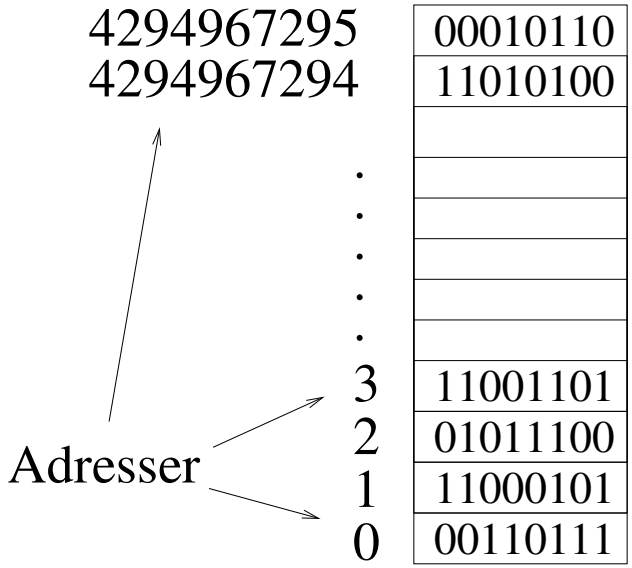
Minne-pyramiden



Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

Internminnet

Internminnet/RAM (Random Access Memory)/arbeidsminnet er et stort array av bytes med hver sin adresse:



Deadlock
Internminne
Internminne og Cache
Minne-pyramiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

Adresserommet

- Adressene må kunne lagres i registere
- For eksempel inneholder %rsp adressen til toppen av stack'en
- Adresserommet: Alle mulige adresser (f. eks. IPv4 adresserommet 0.0.0.0 - 255.255.255.255)
- Adresserom for internminnet: 0 - Maks
- Antall bit man bruker bestemmer hvor stort adresserommet blir

Registerstørrelse (i bit)	antall adresser
16	$2^{16} = 64$ K (Kilo, 10^3)
32	4 G (Giga, 10^9)
48	256 T (Tera, 10^{12})
64	20 E (Exa, 10^{18})

Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

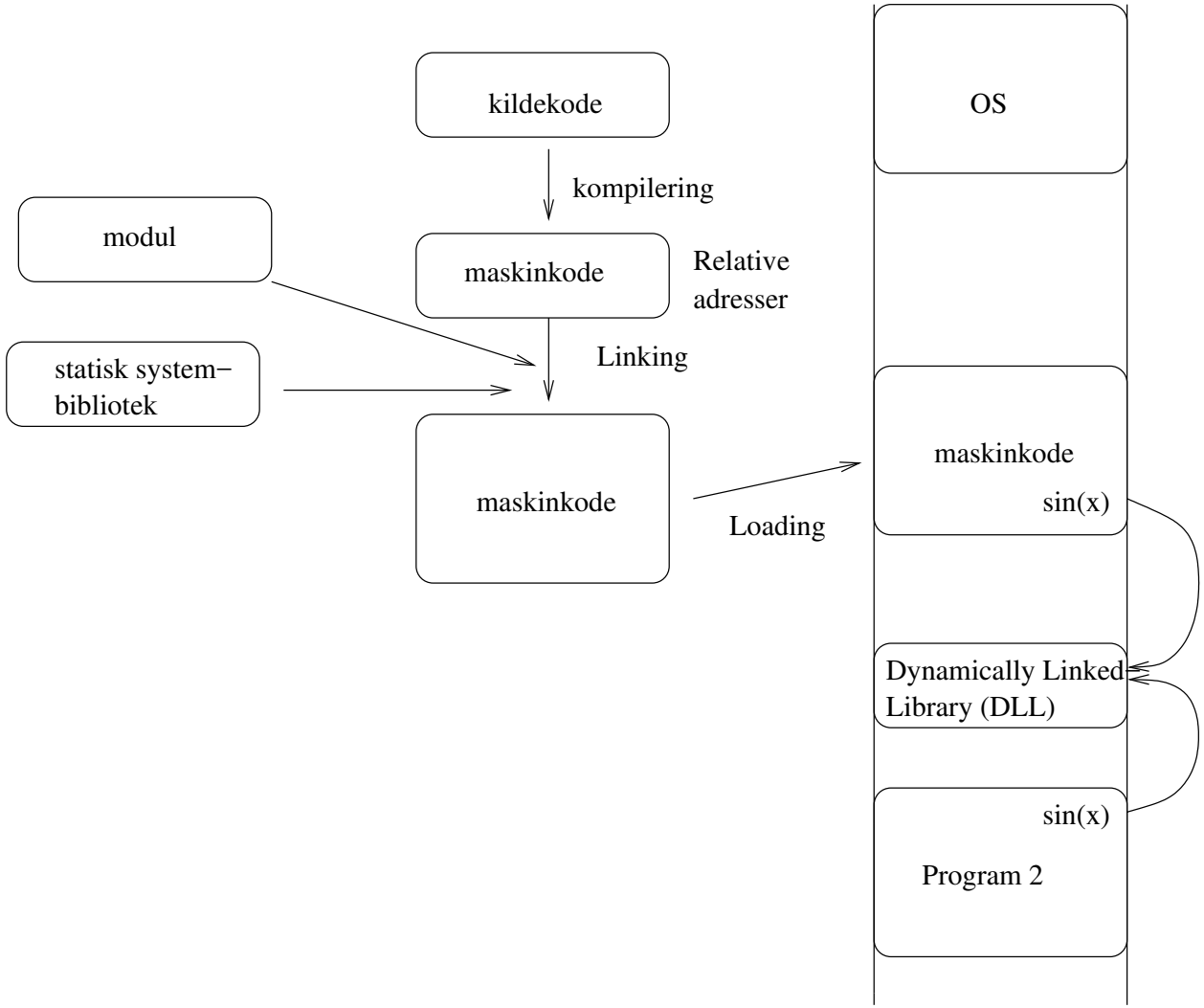
Virtuelt adresserom

- Ikke plass til alle programmer i internminnet på en gang
- Hvert enkelt program får sitt eget virtuelle adresserom fra 0 til f.eks. 4G (32 bit adresser)
- Prosessen tror den har tilgang til alt dette minnet
- I virkeligheten ligger noe i RAM og andre deler på harddisken
- Disse virtuelle eller logiske adressene brukes overalt hvor programmet refererer til seg selv
- I instruksjonen `mov (1023), %al` er 1023 den virtuelle adressen
- Når et programmet lastes inn i internminnet og kjøres vil det variere hvor i det fysiske minnet programmet legges
- CPU må oversette ekstremt hurtig mellom virtuelle og fysiske adresser
- Gjøres av MMU (Memory Management Unit)

Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

Internminnet

Et kjørbart program ligger i utgangspunktet på harddisken, men må lastes inn i internminnet før det kan kjøres.



Deadlock
Internminne
Internminne og Cache
Minne-pyramiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

C++ library

Et C++ prosjekt kan ha metoder man ofte bruker i en egen library-fil som man kobler sammen med hovedprogrammet når man skal bruke det.

```
g++ -c calcTools.cpp                          # Lager maskinkode calcTools.o
g++ -c randTools.cpp                          # Lager maskinkode randTools.o
ar rcv libTools.a  calcTools.o   randTools.o # Lager lib-filen libTools.a
```

Senere kan dette biblioteket brukes i et prosjekt:

```
g++ -c -I../Tools mainsim.cpp       # Lager maskinkode mainsim.o
g++ -c -I../Tools simulation.cpp    # Lager maskinkode simulation.o
g++ -c  -I../Tools user.cpp         # Lager maskinkode user.o
g++ -o sim mainsim.o simulation.o user.o  -lm  -L../Tools -lTools
sim  # Kjører programmet
```

Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

C++ dynamic library

Man kan også lage et dynamisk (shared) library med filendelse so.

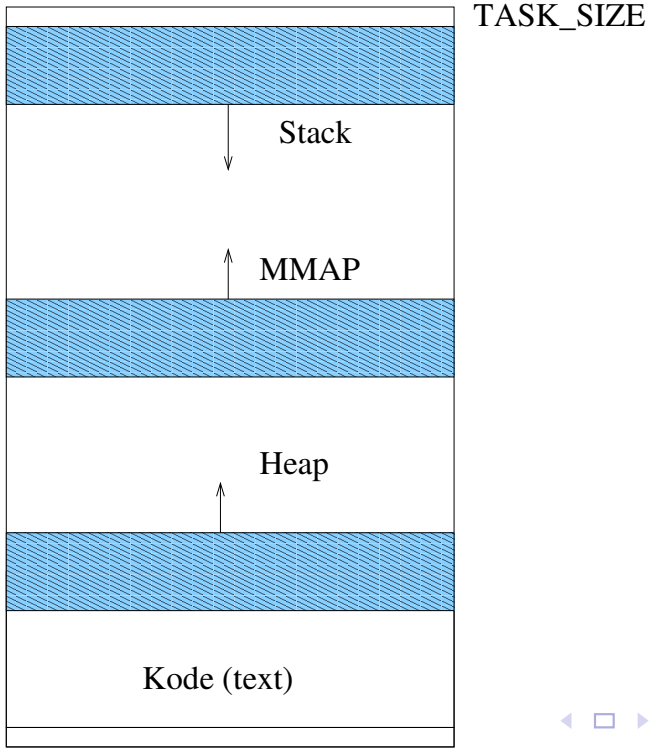
```
g++ -fpic -c randTools.cpp
g++ -fpic -c calcTools.cpp
g++ -shared -o libsTools.so randTools.o calcTools.o

g++ -o sim mainsim.o simulation.o user.o  -lm  -L../sTools -lsTools
export LD_LIBRARY_PATH="../sTools"
sim  # Kjører programmet
```

Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

Linux-prosess segmentation

- Den statiske binære koden som prosessen kjører, text segmentet, ofte bare kalt text
- Heap'en hvor globale variabler og data som dynamisk generers lagres
- Stack'en hvor de lokale variablene lagres, brukes også til funksjonskall
- MMAP, minneavbildninger av filer(og devicer) på disk direkte i det virtuelle minnet



Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

Minneadressering og MMU

- Et programs virtuelle adressering til variabler, subrutiner, bibliotek, data og så videre må knyttes til fysiske adresser
- Kunne skjedd ved loading, men tidkrevende og tungvint
- I moderne OS gjøres dette dynamisk mens programmene kjører
- Muligjør at programmer og biblioteker kan flyttes til og fra harddisk og bare loades når det er behov for dem.
- OS oversetter fysiske adresser til logiske/virtuelle? Altfor sakte og for mye belasting av CPU
- Må skje på brøkdelen av et nanosekund, ikke tid til en add-instruksjon
- Gjøres av egen enhet inne i CPU: MMU (Memory Managment Unit)

Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

MMU (Memory Mangagement Unit)

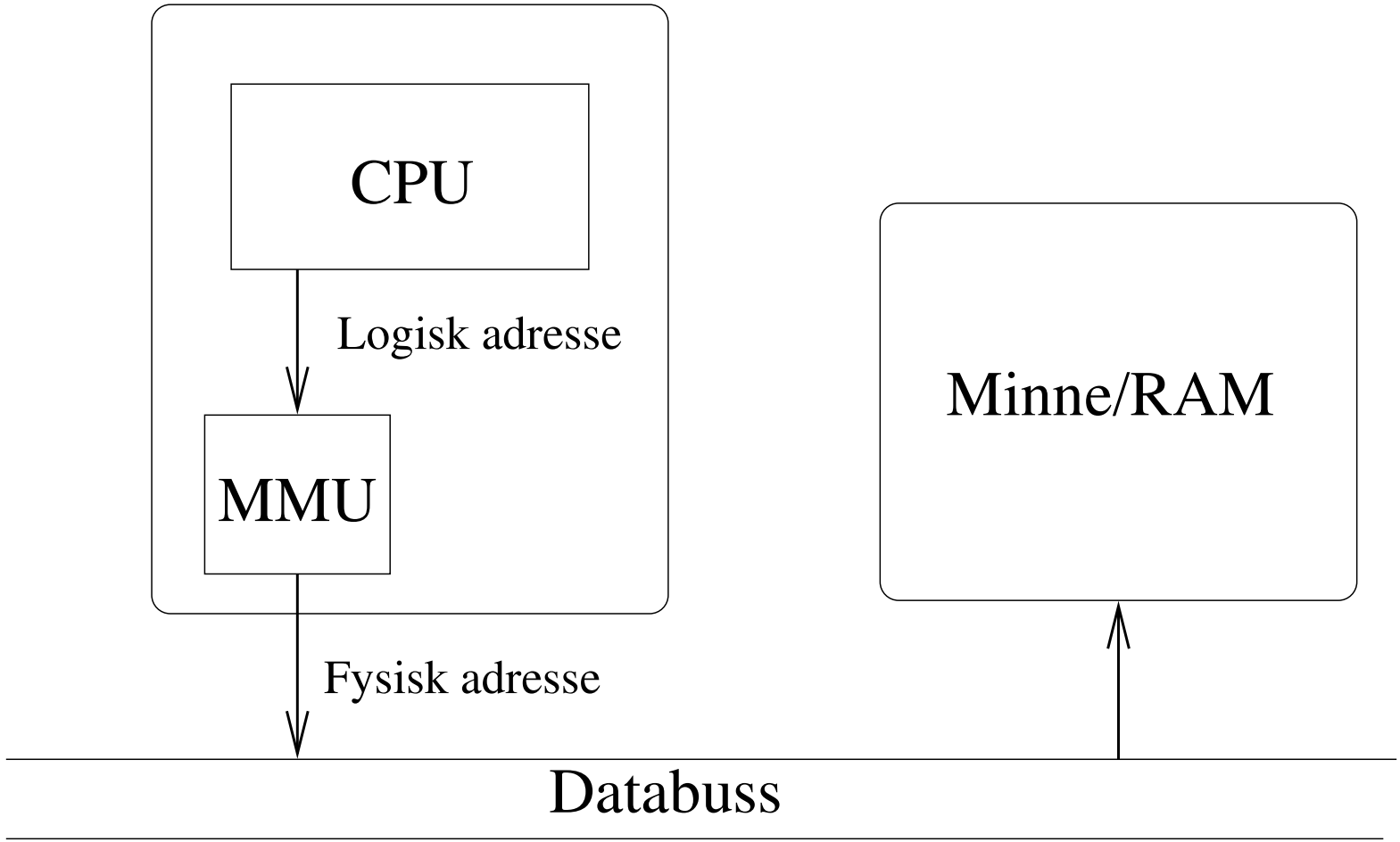


Figure: MMU oversetter logiske adresser fra CPU til fysiske RAM-adresser i realtime

Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

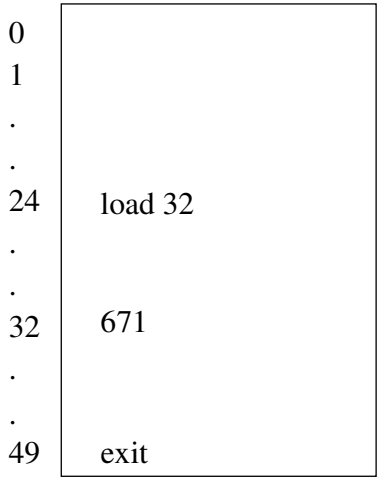
Prog1 og Prog2

- Anta at de to programmene Prog1 og Prog2 skal kjøres
- Adressene i de kompilerte programmene må være logiske og ikke til en fastlagt adresse i minne
- MMU oversetter til fysiske adresser i real-time
- Trenger å oppdatere MMU-tabellene når programmene plasseres eller omplasseres i RAM

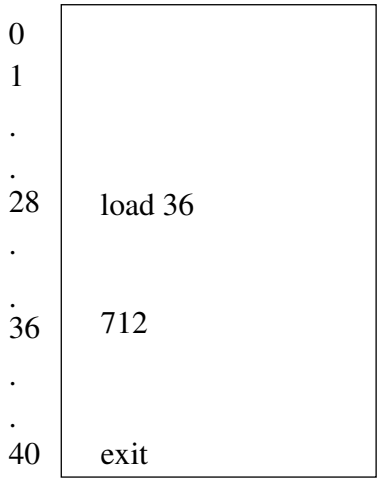
Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

Prog1 og Prog2

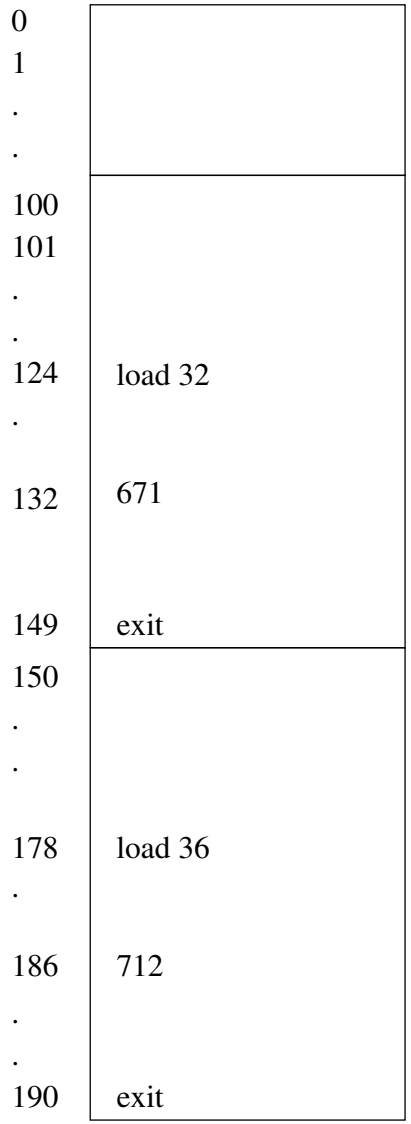
Prog 1



Prog 2



Minne



Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

Eksempel på MMU-tabell

- CPU utfører instruksjonen 'load 32' for Prog1
- Den logiske adressen 32 sendes til MMU som bruker sin tabell for Prog1-adresser til å oversette til fysisk adresser 132
- MMU trenger følgende tabell:

Prog1		Prog2	
0	100	0	150
.	.	.	.
.	.	.	.
24	124	28	178
.	.	.	.
.	.	.	.
32	132	36	186
.	.	.	.
40	140	40	190

- Alt for minnekrevende å ha en linje i tabellen for hver adresse!
- Det logiske minnet deles derfor opp i pages
- I eksempelet, la en page ha 50 adresser
- Prog1 starter på adresse 100 og Prog2 startet på adresse 150

Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

Paging

- Deler inn en prosess sitt virtuelle minnerom inn i like store biter (pages/sider)
- OS kan da effektivt laste disse sidene inn og ut av minnet og samtidig holde oversikt over hvor hver side er i en page-tabell
- Fast sidestørrelser hindrer huller når sider lastes inn og ut; fragmentering
- Dynamisk flytting av deler av prosesser til og fra disk blir mulig
- Gir full kontroll for OS over prosessers minnebruk
- Mulliggjør å bruke diskplass til å utvide minnet, virtuelt minne

Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

Pages

- En page har en størrelse 2^n bytes, typisk er n = 12 eller 13
- Page-størrelsen er dermed 4 eller 8 Kbytes
- 4Kbytes er vanlig for X86-prosessorer
- Context switch; tabellen for den gamle prosessen lagres i PCB
- Deler av tabellen til den nye prosessen lastes inn i MMU

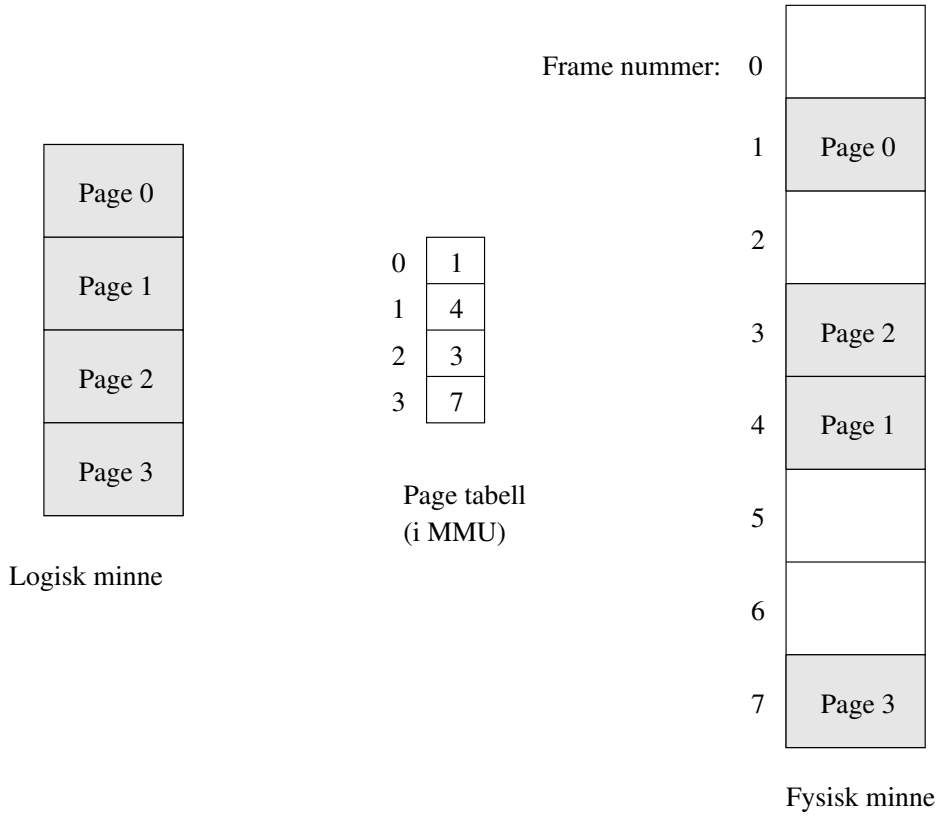
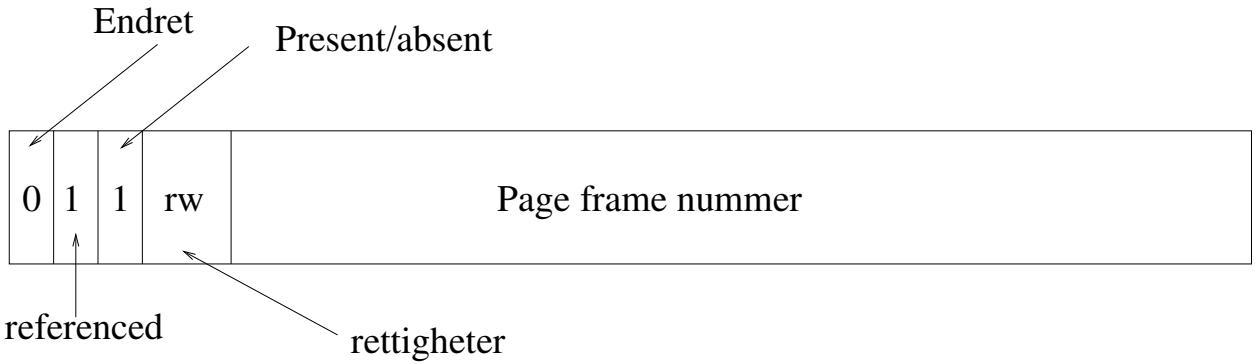


Figure: Logisk minne og paging

Deadlock
Internminne
Internminne og Cache
Minne-pyramiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

Page Table Entry



- Page Frame nummer: Fysisk frame-nummer i RAM
- Present: Hvis 0 blir det en page-fault
- Endret: Hvis 1 er siden dirty og må skrives til disk om den fjernes
- Rettigheter: lese, skrive, kjøre
- Referenced: settes hvis brukt, brukes av paging-algoritmer

Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

TLB - Translation Lookaside Buffer

- En prosess som bruker 100 Mbyte minne vil med 4 Kbyte page størrelse bestå av omtrent 25.000 sider
- En 4 GByte prosess gir en million sider i MMU
- Ikke plass til å lagre adressen til alle disse sidene i MMU
- Den fullstendige tabellen ligger selv i internminnet
- MMU bruker en Translation Lookaside Buffer (TLB) som er hurtig cache minne
- Inneholder en liten del av page-tabellen,
- Ved oppslag på adresser til sider som ikke ligger i TLB, hentes de fra RAM
- Dette kalles TLB-miss eller soft-miss. Tar vesentlig lenger tid enn om de er i TLB

Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

Typisk TLB ytelse

- størrelse: 16 - 4096 linjer (1 - 256 kBytes)
- En cache linje er vanligvis 64 bytes
- oppslagstid: 0.5 - 1 klokke-sykel
- ekstra tid ved TLB-miss: 10-100 klokke-sykler
- TLB-miss frekvens: 0.01 - 1%

Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

Internminnet og Cache

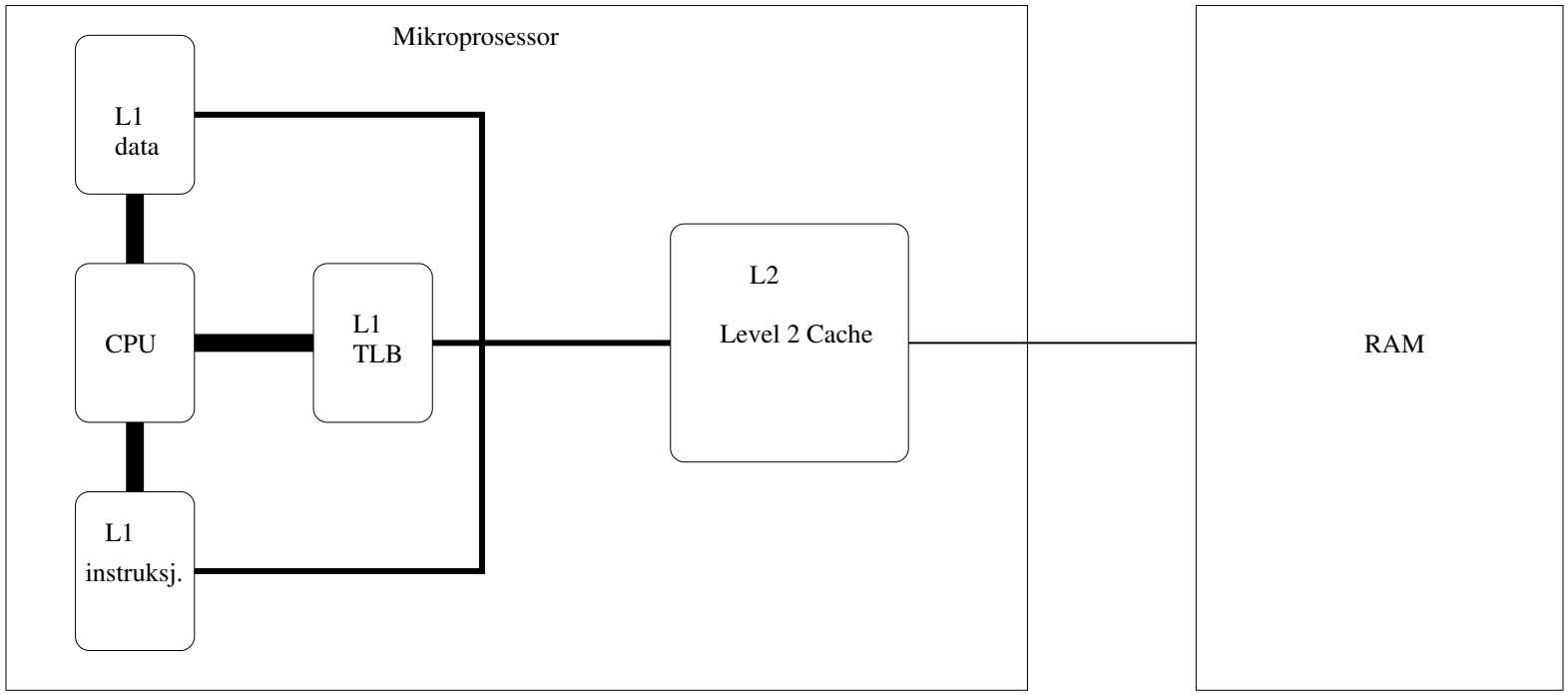


Figure: Level 1 cache (L1) bestående av tre deler. I AMD Athlon 64 er TLB i tillegg delt i to deler, en for adresser til instruksjoner og en for adresser til data.

For Intel Core i7 og AMD Opteron K10 har også L3 cache fått plass på prosessor-chip'en.

Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

64K virtuell og 32K fysisk minne

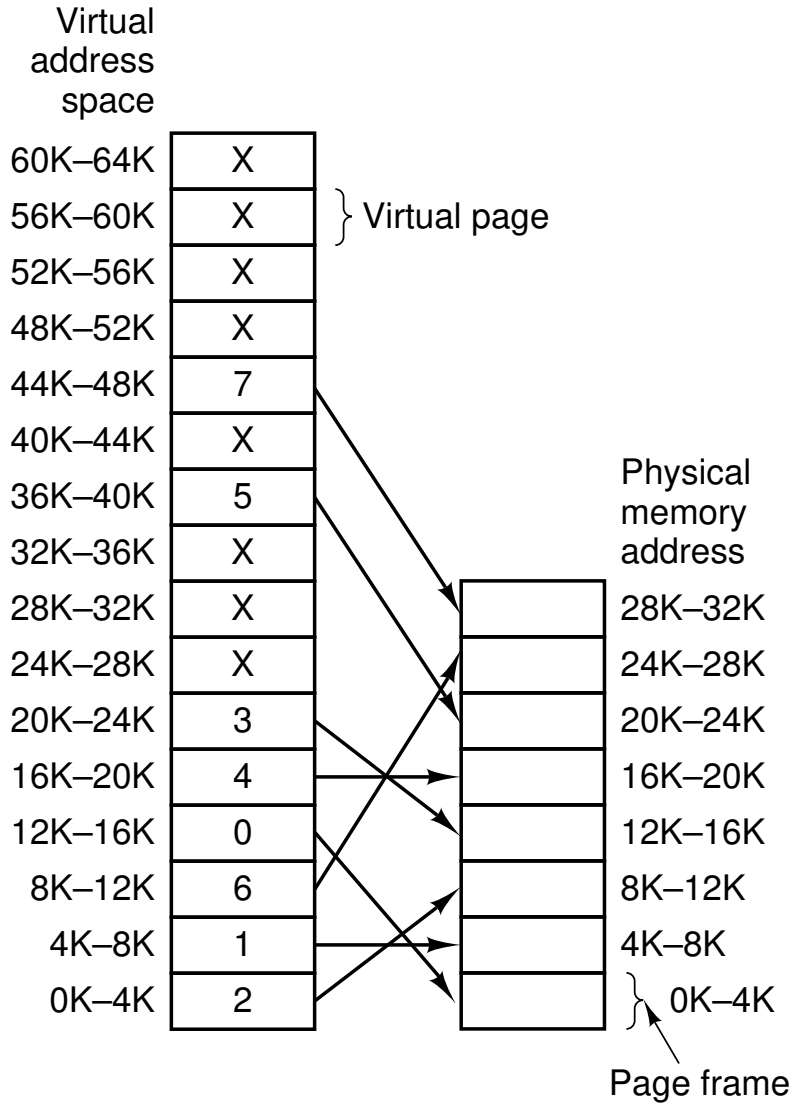


Figure: Figure 3-9 i Tanenbaum. Forholdet mellom virtuelle og fysiske adresser

Deadlock
Internminne
Internminne og Cache
Minne-pyramiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

MMU tabell

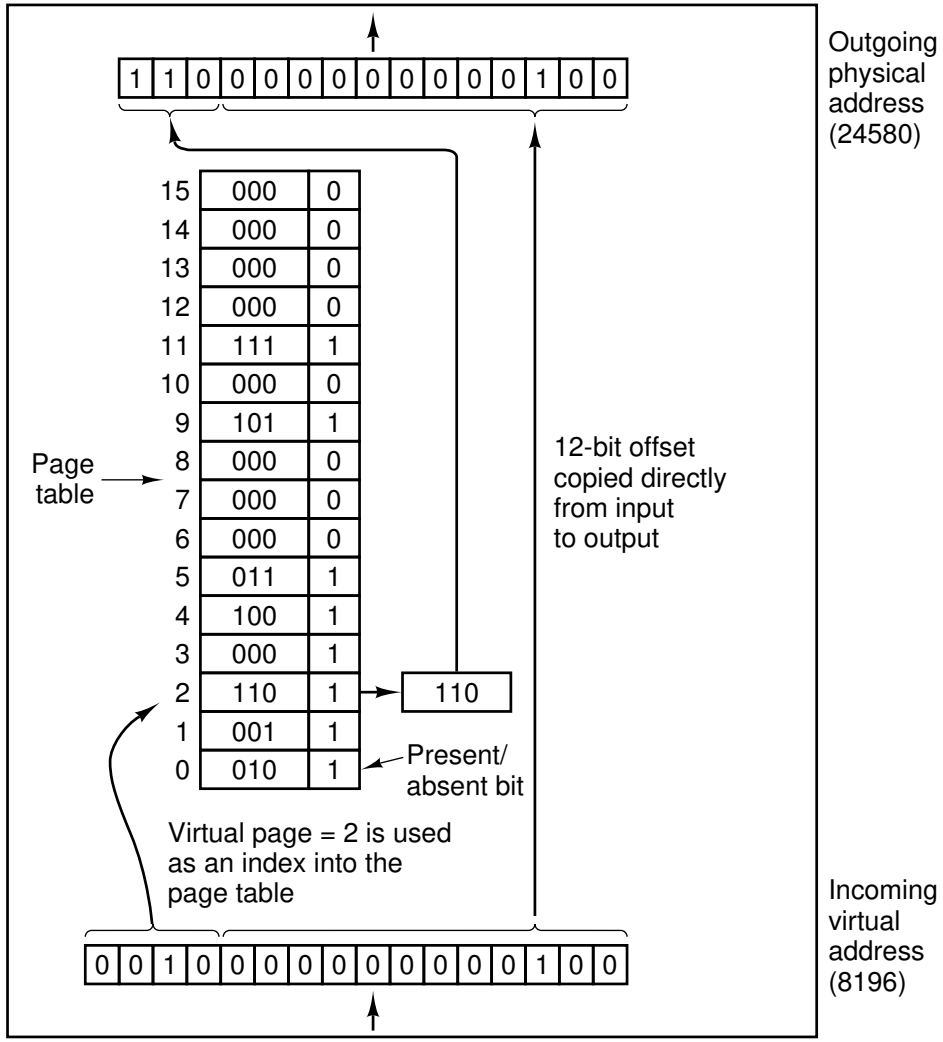


Figure: Figure 3-10 i Tanenbaum. Slik oversetter MMU virtuelle(logiske) adresser til fysiske.

Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

Paging og swapping

- Pages gjør det mulig å dynamisk laste inn og ut deler av en prosess
- Minnet til det samlede antall prosesser på en maskin være større enn det fysiske minnet
- Resten lagres page for page på harddisk på swap-området

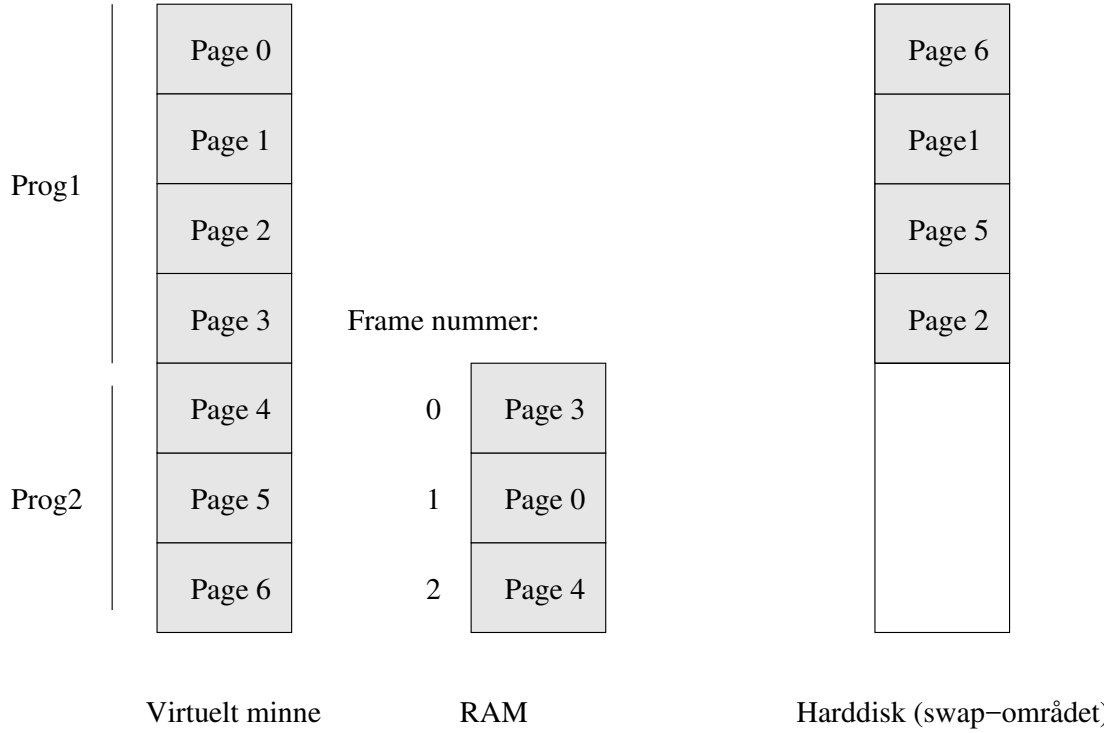


Figure: Virtuelt minne. Bare en page av Prog2 ligger i minnet.

Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

Paging-algoritmer

1. Page-fault (En page mangler i minnet; ligger på swap på disk)
2. Ingen ledig frame i minnet
3. OS må velge hvilken page som skal legges på disk
4. Gjøres av paging-algoritme

Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

Dynamisk allokering

Et program kan be om at det settes av RAM til sine variabler før det starter, men det kan også be om at minne allokeres dynamisk. Følgende Java-statement

```
PCB = new process;
```

gjør at denne plassen settes av i minnet først når programmet utfører det.

- Programmet tildeles page for page med minne
- I C og C++ må man eksplisitt delete objekter som ikke er i bruk lenger for å frigjøre minne
- JVM utfører dette automatisk (garbage collection)
- Den delen av et programs minne som inneholder variabler og data og som dynamisk kan øke og minke i størrelse, kalles ofte heap.

Deadlock
Internminne
Internminne og Cache
Minne-pryamiden
Internminnet
Adresserommet
Virtuelt adresserom
Internminnet
C++ library
C++ dynamic library
Linux-prosess segmentation
Minneadressering og MMU
MMU (Memory Mangagement Unit)
Prog1 og Prog2
Prog1 og Prog2
Eksempel på MMU-tabell
Paging
Pages
Page Table Entry
TLB - Translation Lookaside Buffer
Typisk TLB ytelse
Internminnet og Cache
64K virtuell og 32K fysisk minne
MMU tabell

Noen minne-begreper

- **Soft miss** page-referanse er ikke i TLB; må hentes fra internminnet. Også kalt TLB-miss
- **Hard miss** = page fault. En page mangler i minnet(og i TLB); må hentes fra disk
- **Major fault** = page fault. En page mangler i minnet(og i TLB); må hentes fra disk
- **Minor fault** = En page mangler i page-tabellen i RAM og må lages. Må IKKE hentes fra disk
- **Dirty page** En side som har blitt endret slik at den må skrives til disk om den må ut av minnet
- **Working set** (Windows) Det sett av sider som en prosess har brukt nylig
- **Segment** En logisk del av et programs minne, data, programtekst, stack-segmenter
- **Buffer cache** Del av minnet som brukes som filsystem-cache