Geheugenbeheer

ICT Infrastructuren

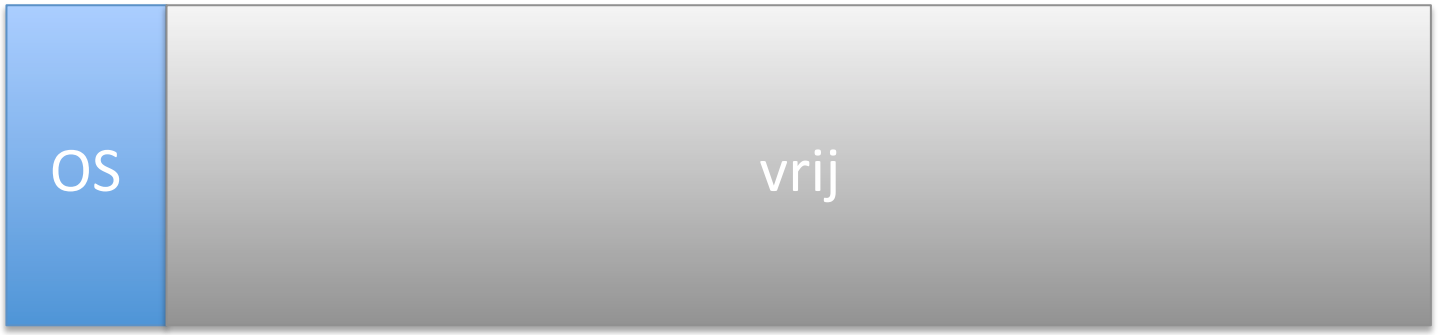
2 december 2013

Doelen van geheugenbeheer

- Relocatie (flexibel gebruik van geheugen)
- Bescherming
- Gedeeld/gemeenschappelijk geheugen
- Logische indeling van procesonderdelen
- Fysieke plaatsing in RAM

Eenvoudig geheugenbeheer

- OS gebruikt een klein stukje geheugen
- rest is voor gewone processen



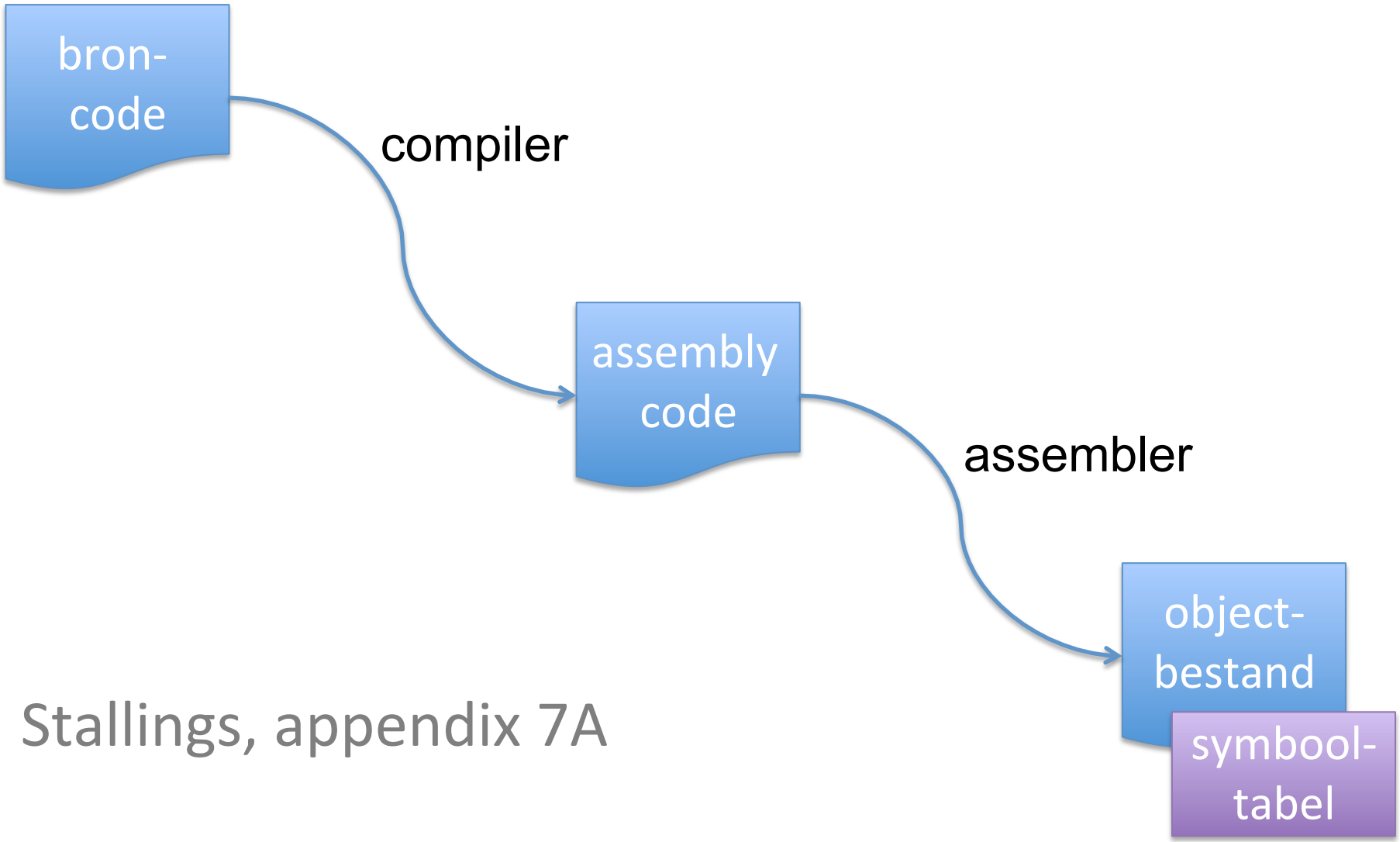
Eenvoudig geheugenbeheer

- OS gebruikt een klein stukje geheugen
- rest is voor gewone processen
  - processen beginnen op adres 1000
  - waar plaats je een tweede proces?

Relocatie

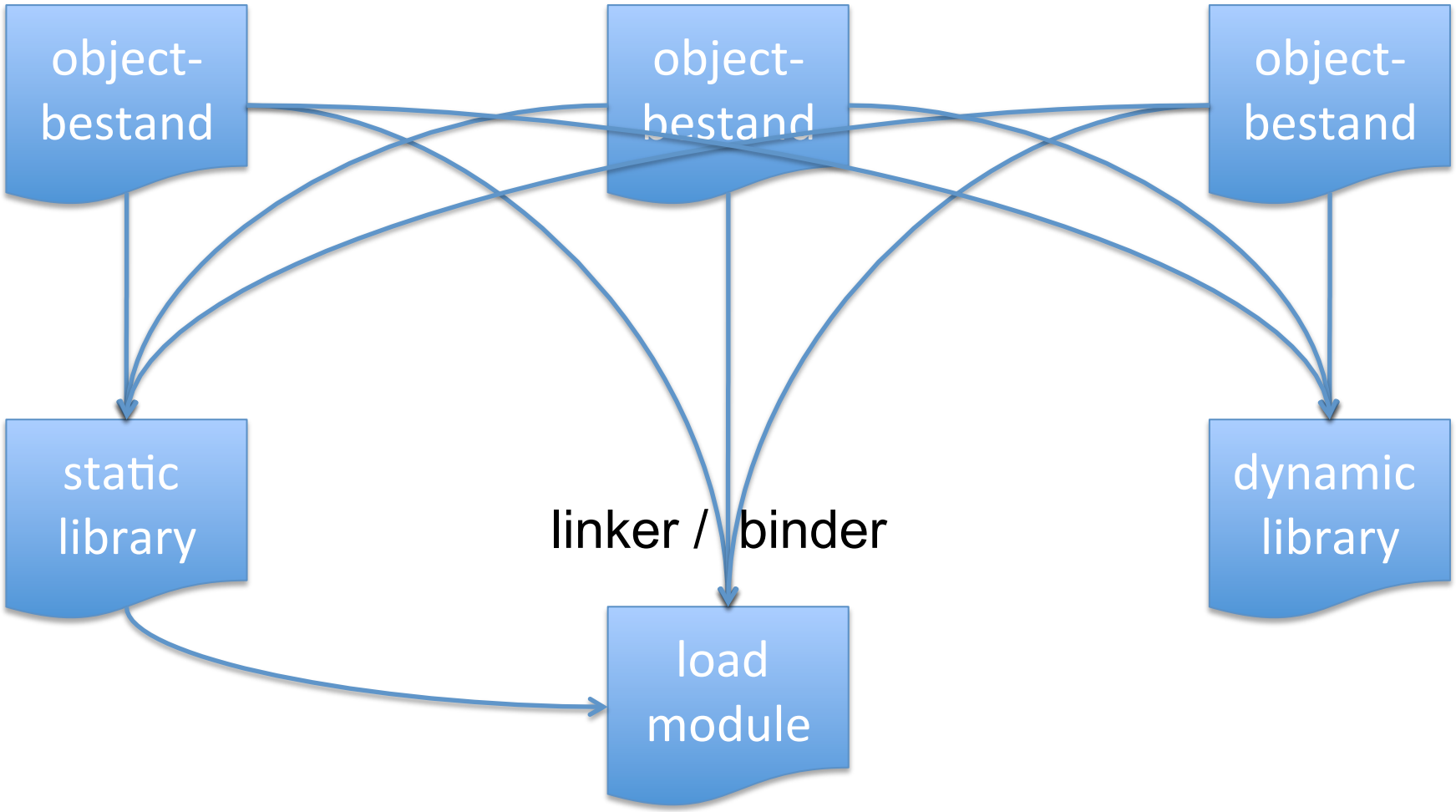
- programma aanpassen aan beginadres
- OS moet adressen vertalen
- bestand bevat nodige informatie

Van broncode naar proces

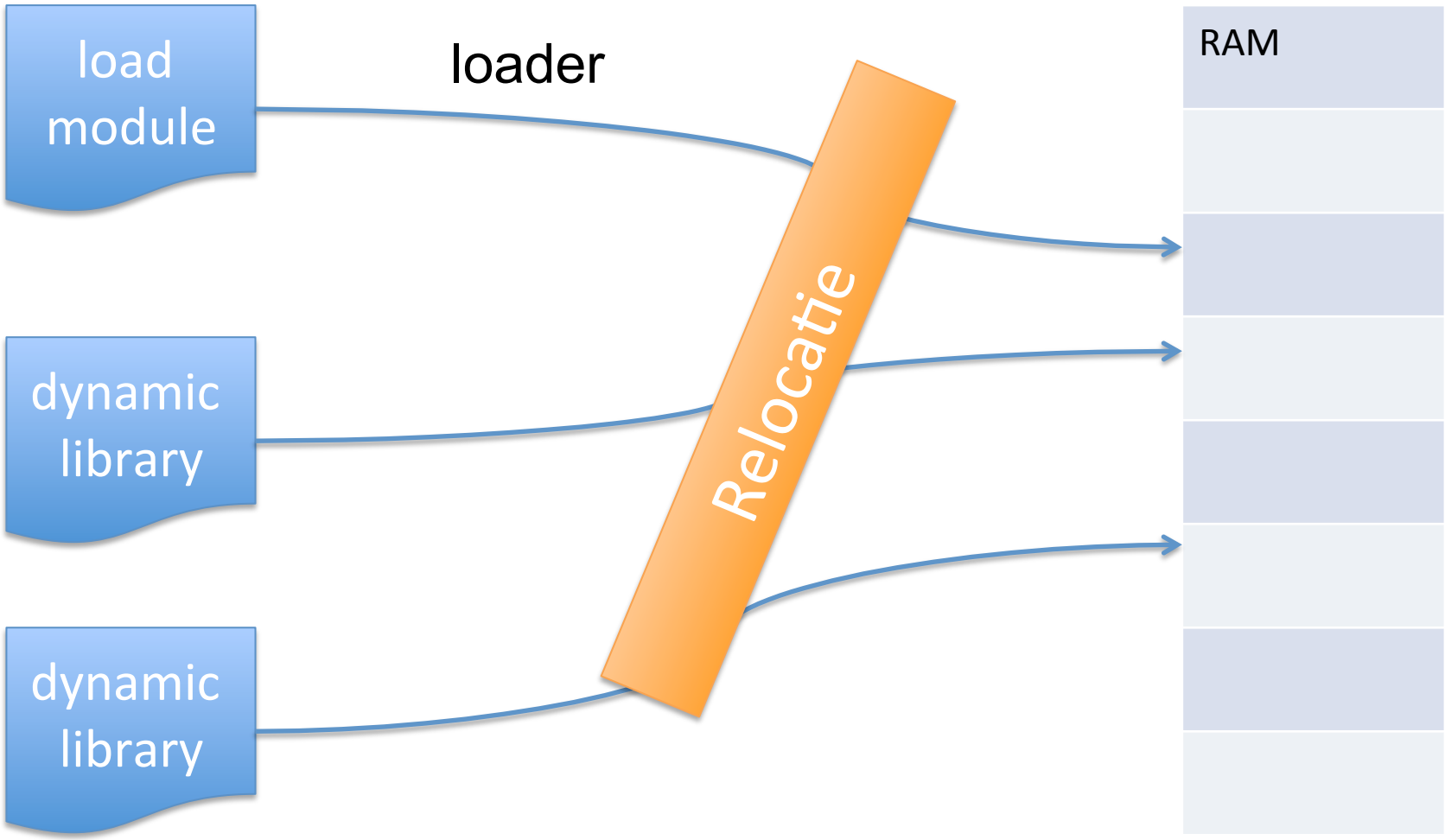


Stallings, appendix 7A

Van broncode naar proces



Van broncode naar proces



Geheugenbeheer voor meer dan één proces

- Processen afschermen:
  geen vreemde gegevens lezen of verknoeien

- Gedeeld geheugen:
  samenwerkende processen kunnen gemeenschappelijke gegevens hebben
  - b.v. programmacode
  - gemeenschappelijke variabelen (op verzoek)

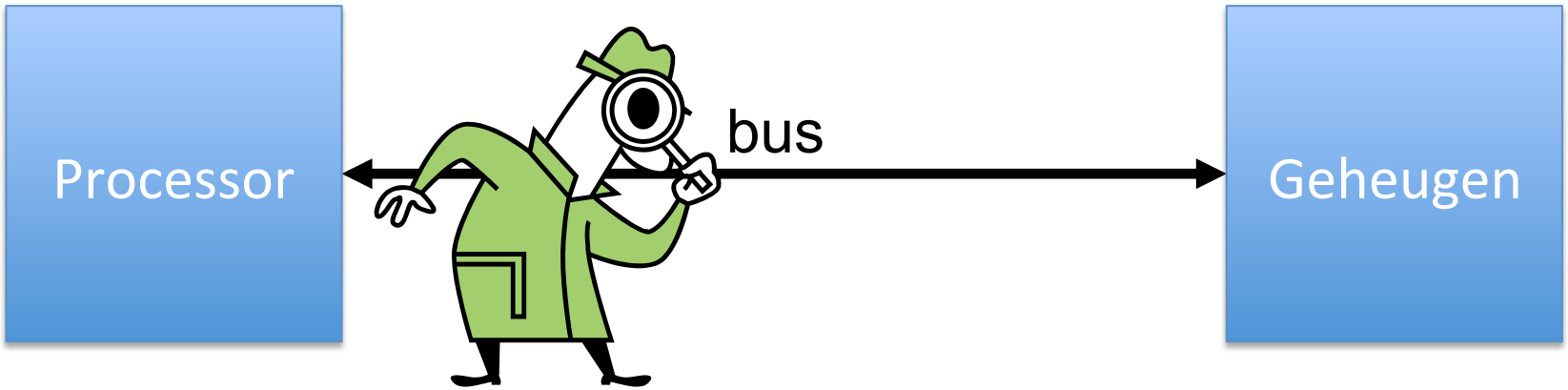
Logische organisatie van RAM

- verschillende soorten gegevens in RAM:
  - programmacode
  - constanten
  - variabelen
  - stapel
- OS kan specifieker beschermen

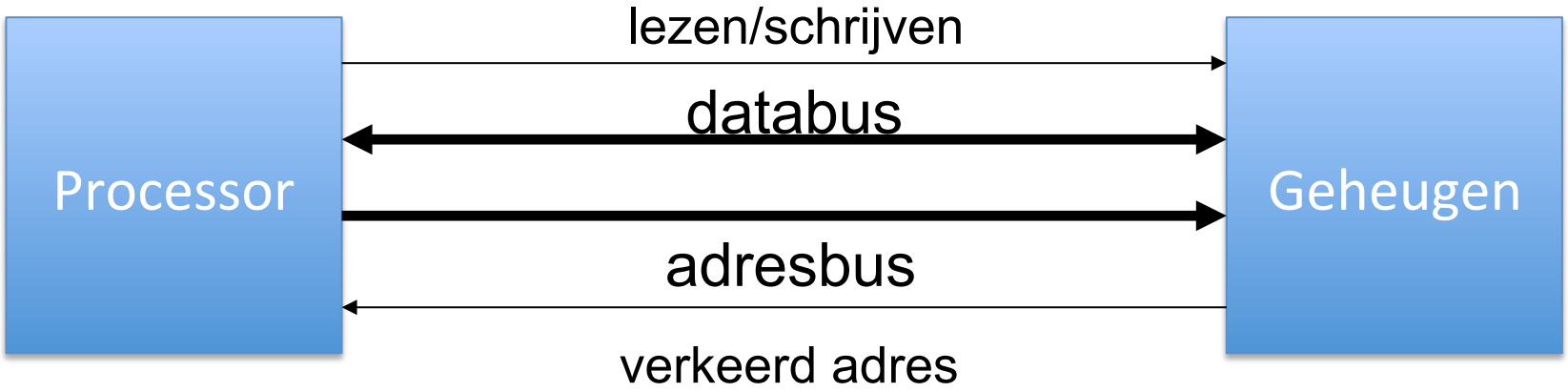
Fysieke plaatsing in RAM

- programma en data vaak te groot voor RAM
- trucs om RAM groter te laten lijken:
  - logisch adres ≠ fysisch adres
  - logisch = wat de programmeur/CPU ziet
  - fysiek = wat de RAM-chip ziet
- Memory Management Unit vertaalt logisch → fysiek

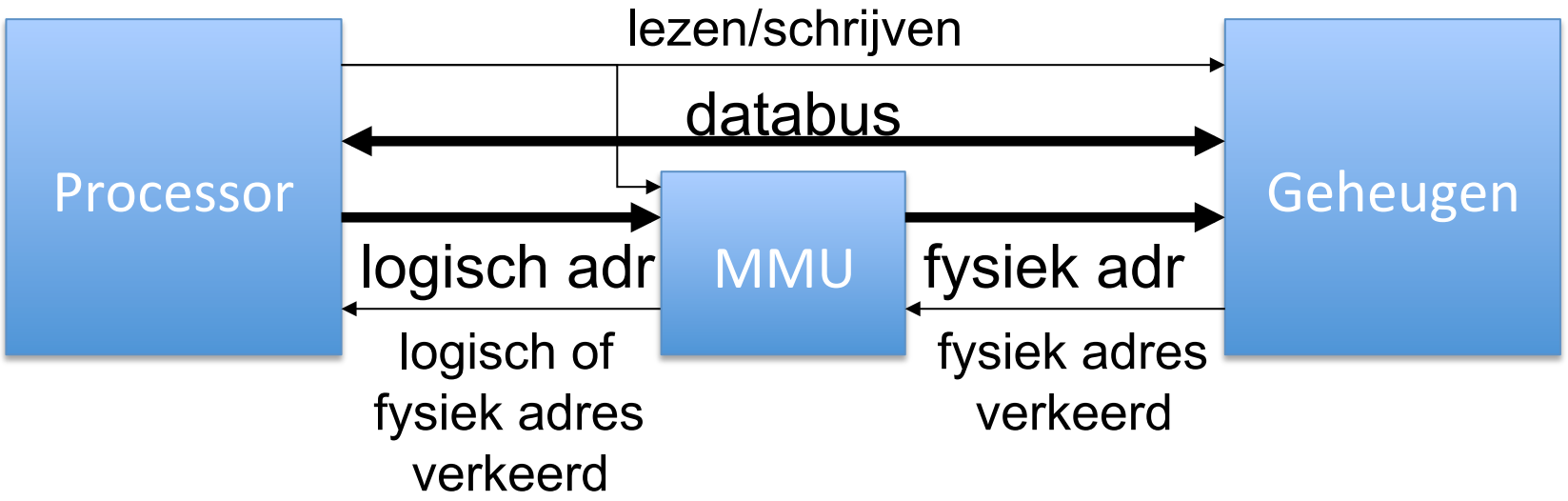
Memory Management Unit



Memory Management Unit



Memory Management Unit



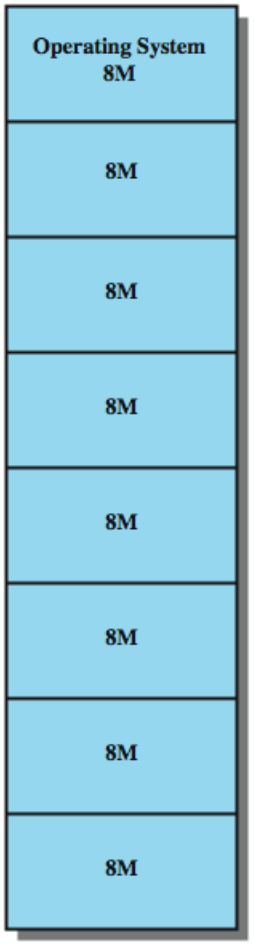
Technieken voor logische adressen

- Partitionering
- Paging
- Segmentatie
- Virtueel geheugen

Partitionering

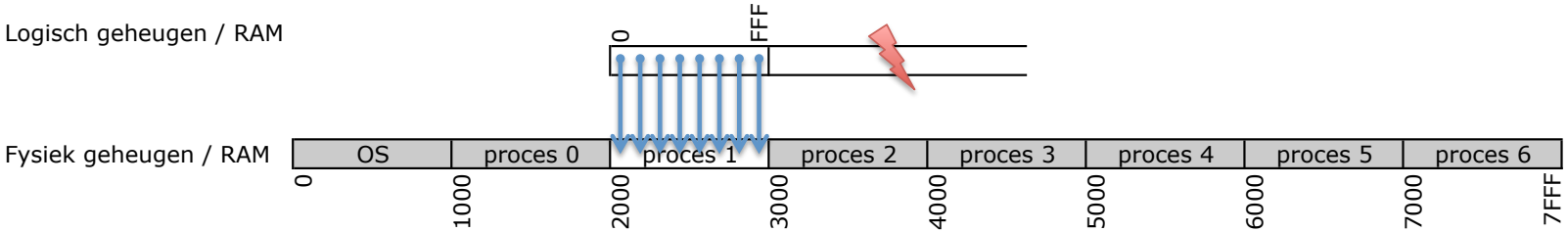
- verouderd,
  basis van andere technieken
- geheugen verdeeld in vaste blokken
- elk proces krijgt één blok

- logisch adres = adres binnen blok
- fysiek adres = logisch adres +
  beginadres van blok

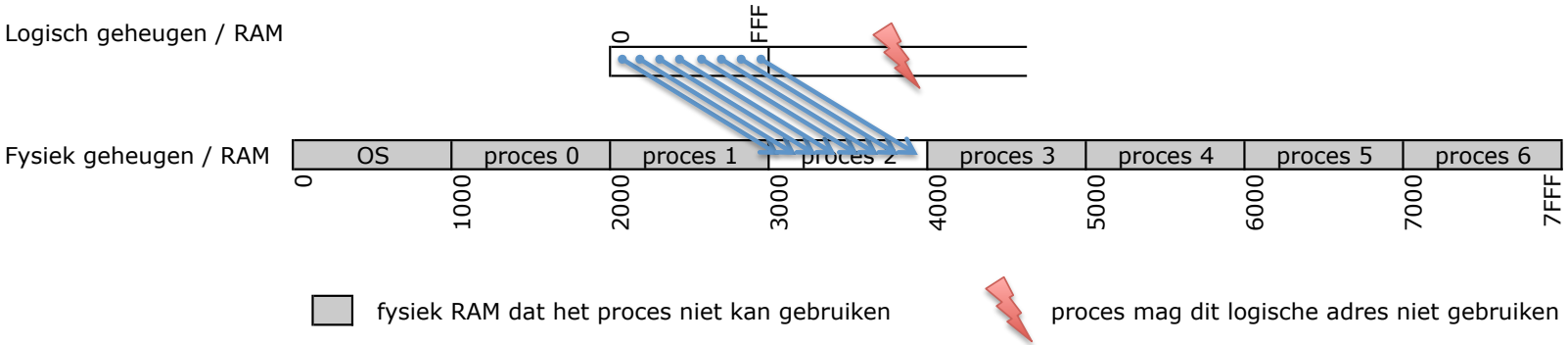


Partitionering

Als proces 1 actief is:



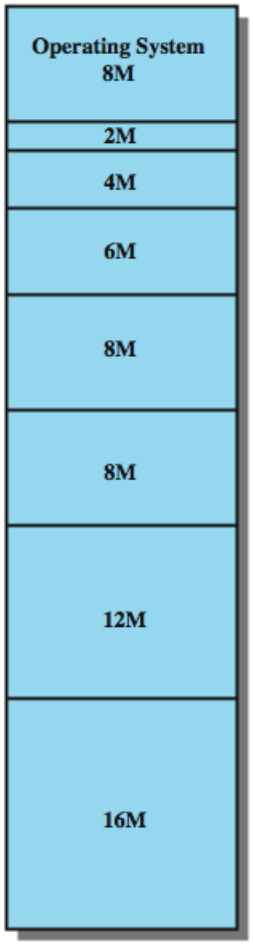
Na een process switch naar proces 2:



Partitionering

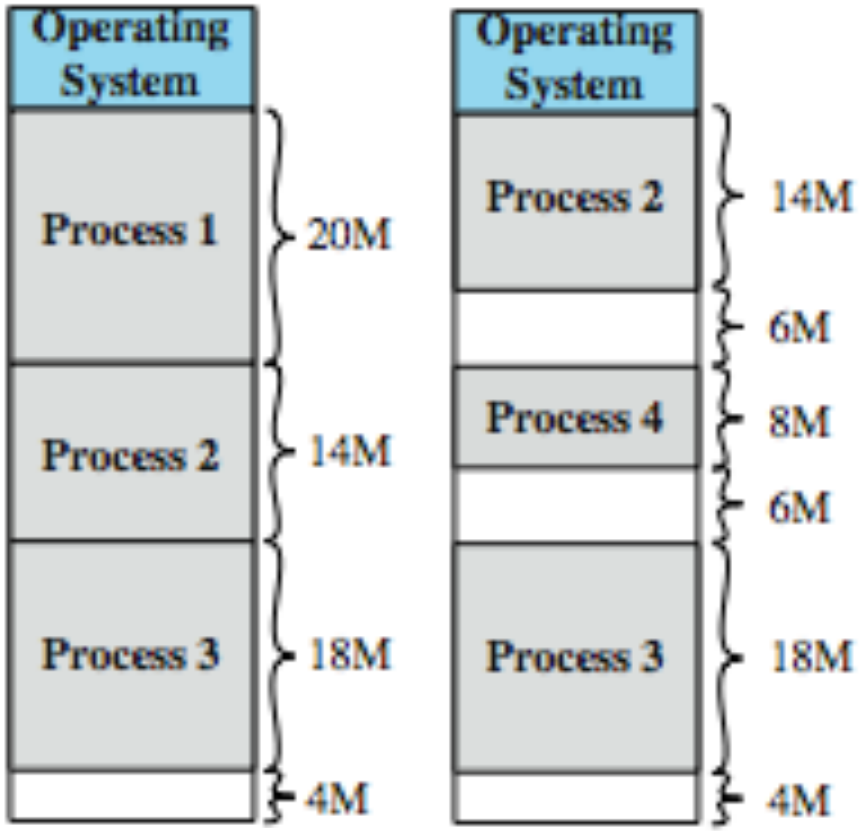
- nadeel: kleine en grote processen krijgen evenveel geheugen (interne fragmentatie)

- variant: verschillend grote blokken
- nieuw probleem:
  veel kleine processen → inefficiënt
  twee grote processen → wachten



Dynamische partitionering

- OS maakt blok aan als proces start
- bij einde ontstaat gat (externe fragmentatie)
- verschillende groottes
  - welk gat gebruiken voor nieuw proces?
  - “first / next / best fit”



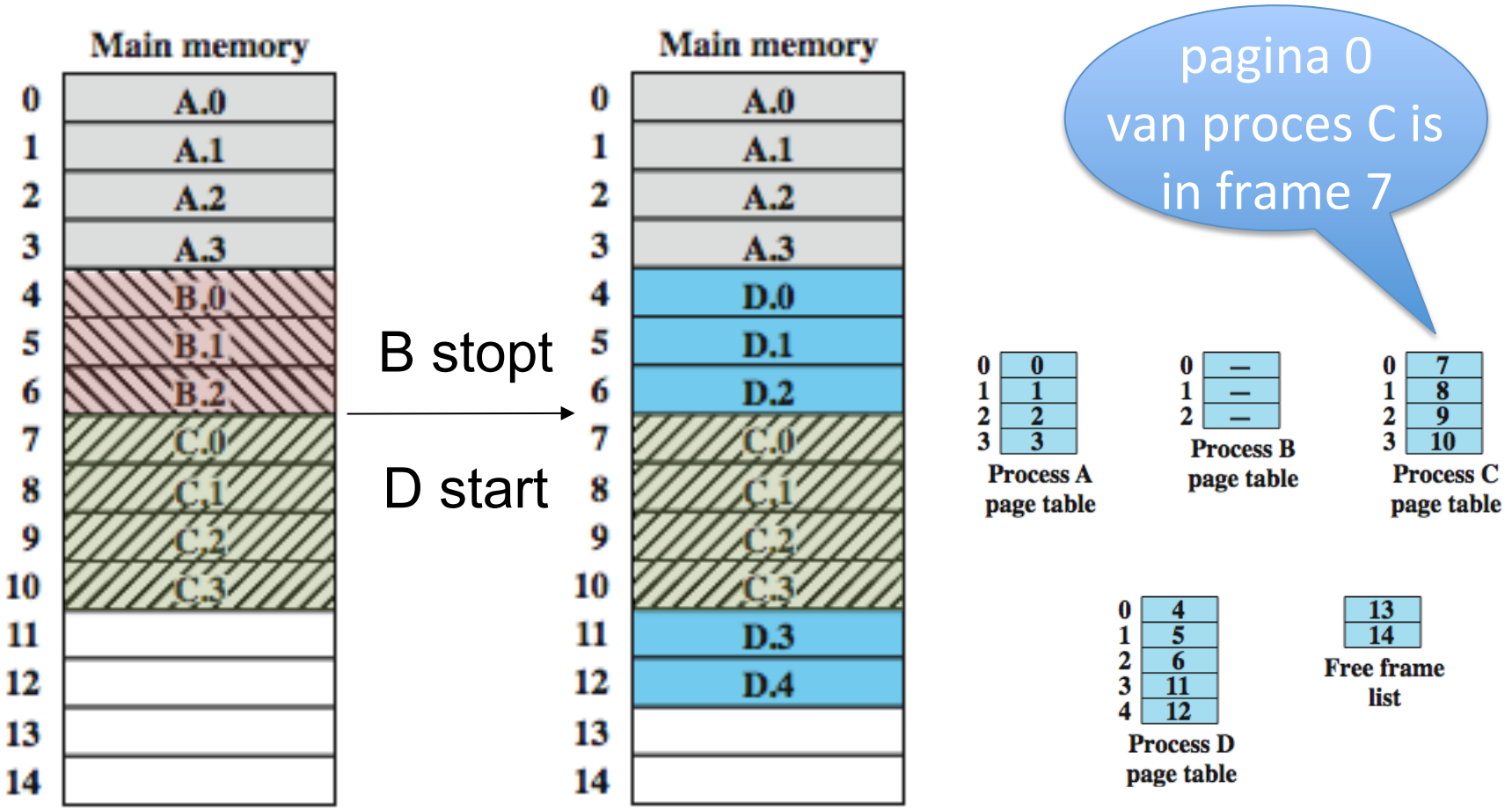
drie opdrachten uit Stallings

- Paging: hoofdstuk 7.3, figuur 7.9 en 7.12a
- Segmentatie: hoofdstuk 7.4, figuur 7.12b
- Virtueel geheugen: hoofdstuk 8.1:
  pagina 361–364, 367, figuur 8.4

Paging

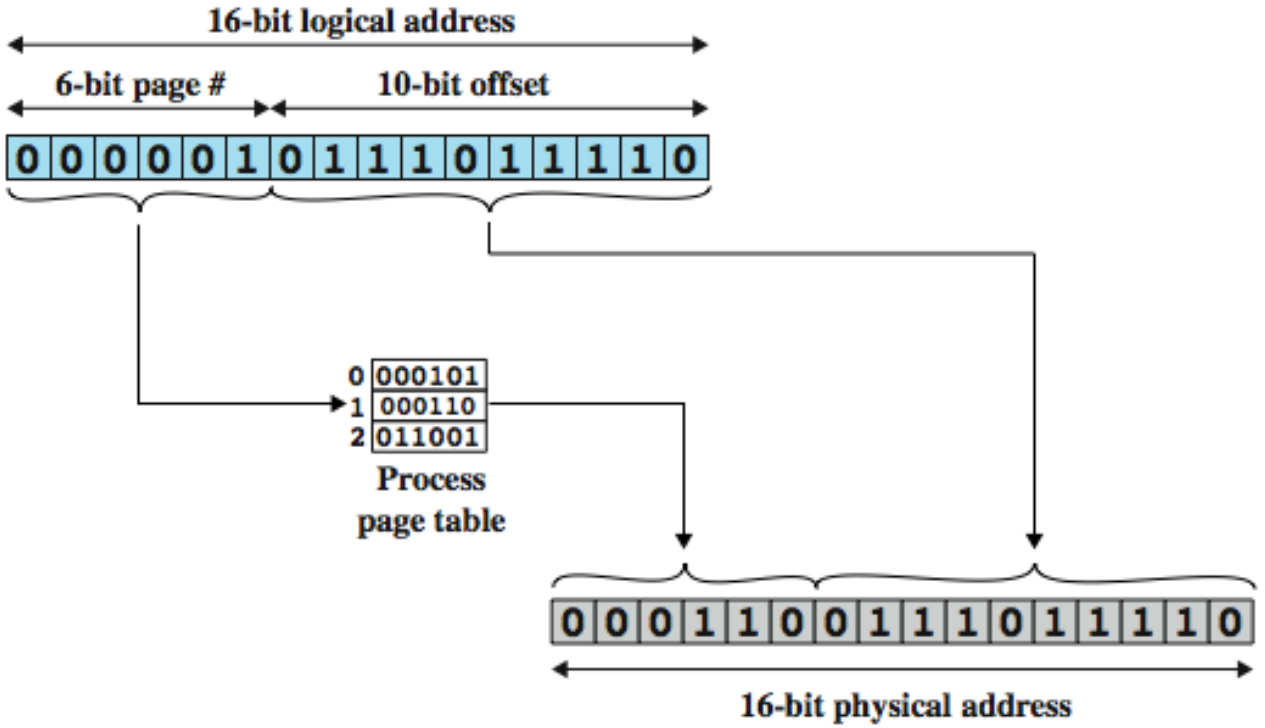
- geheugen verdelen in kleine blokken van vaste grootte
  - frame = fysiek geheugenblok
  - pagina = logisch geheugenblok
- proces gebruikt 1 of meer pagina's
- fysieke adressen van pagina's hoeven niet op elkaar te volgen

Voorbeeld: Paging

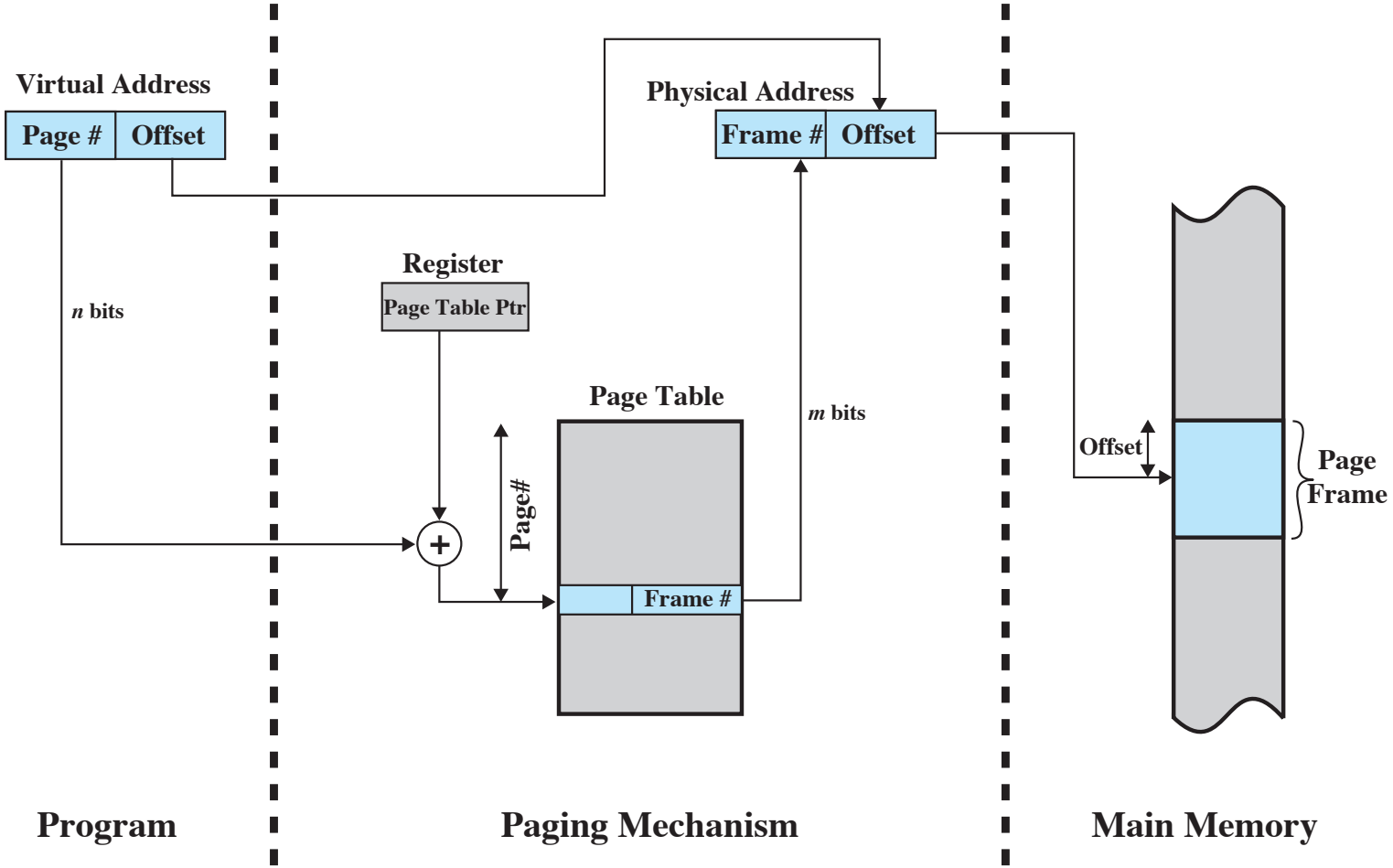


Paging: Berekening van fysiek adres

- hogere bits van logisch adres: paginanummer
- lagere bits: offset binnen de pagina



Paging: Berekening van fysiek adres



Segmentatie

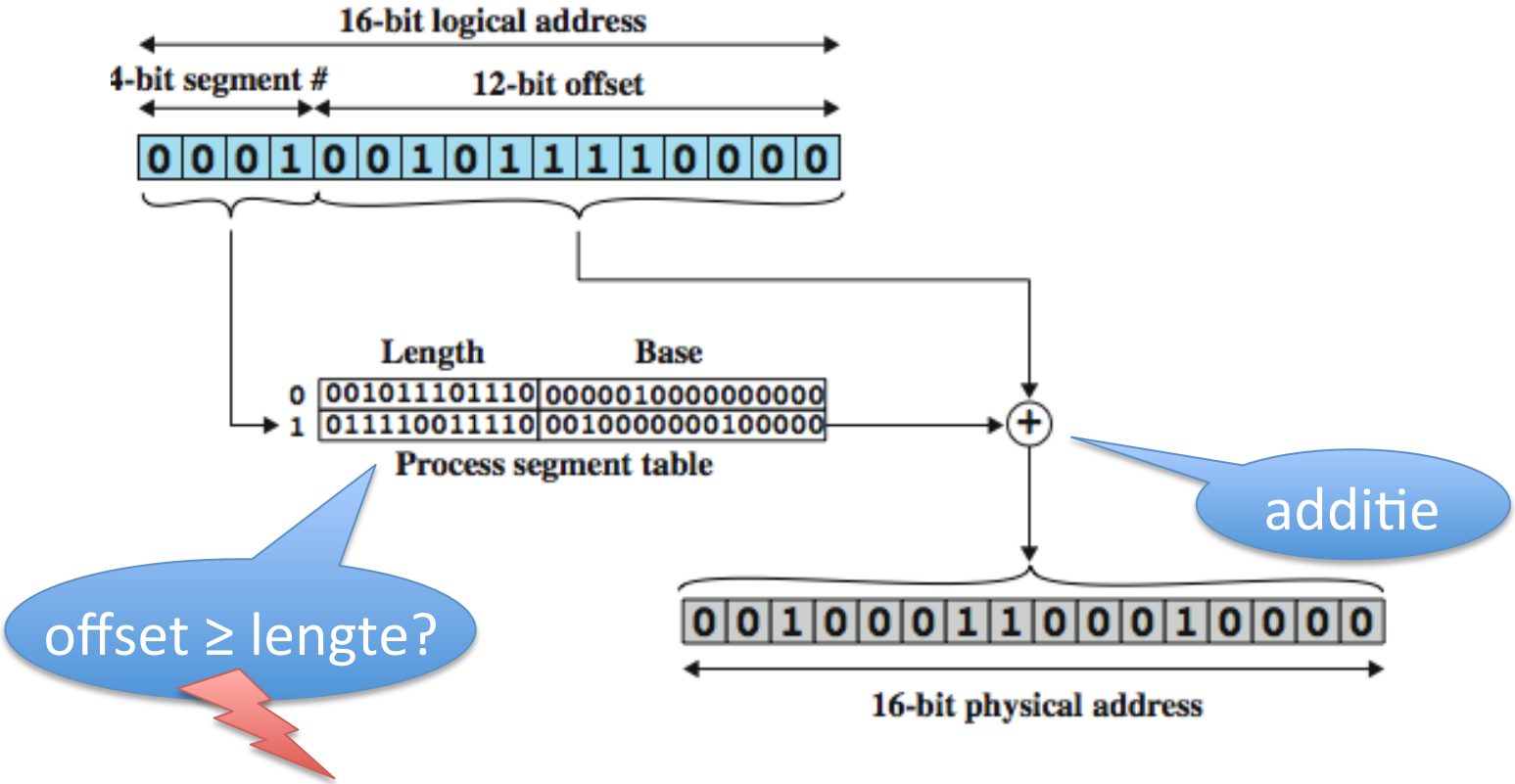
- geheugen verdelen in blokken van variabele grootte
  - segment bevat een stuk programmacode of gegevens voor één proces
- lijkt op dynamisch partitioneren
  - maar proces kan meerdere segmenten hebben
- segmenttabel ≈ paginatabel +

Segmentatie

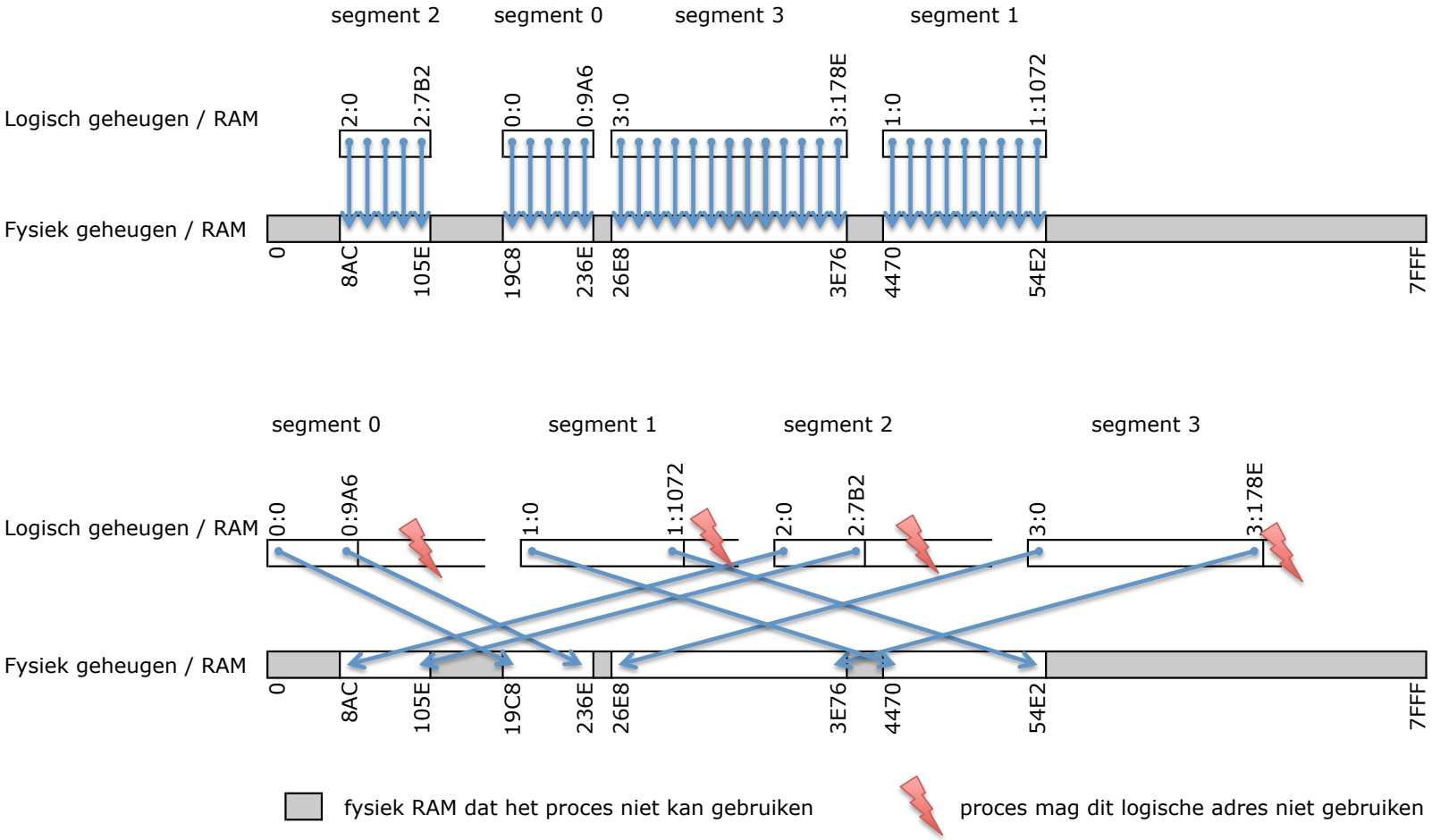
- programmeur ziet segmenten
- betere bescherming
  - read-only segmenten
  - execute-only segmenten
  - strengere controle op verkeerd adres
- geen interne fragmentatie
- wel externe fragmentatie

Segmentatie: Berekening van fysiek adres

- hogere bits van logisch adres: segmentnr
- lagere bits: offset binnen het segment



Segmentatie

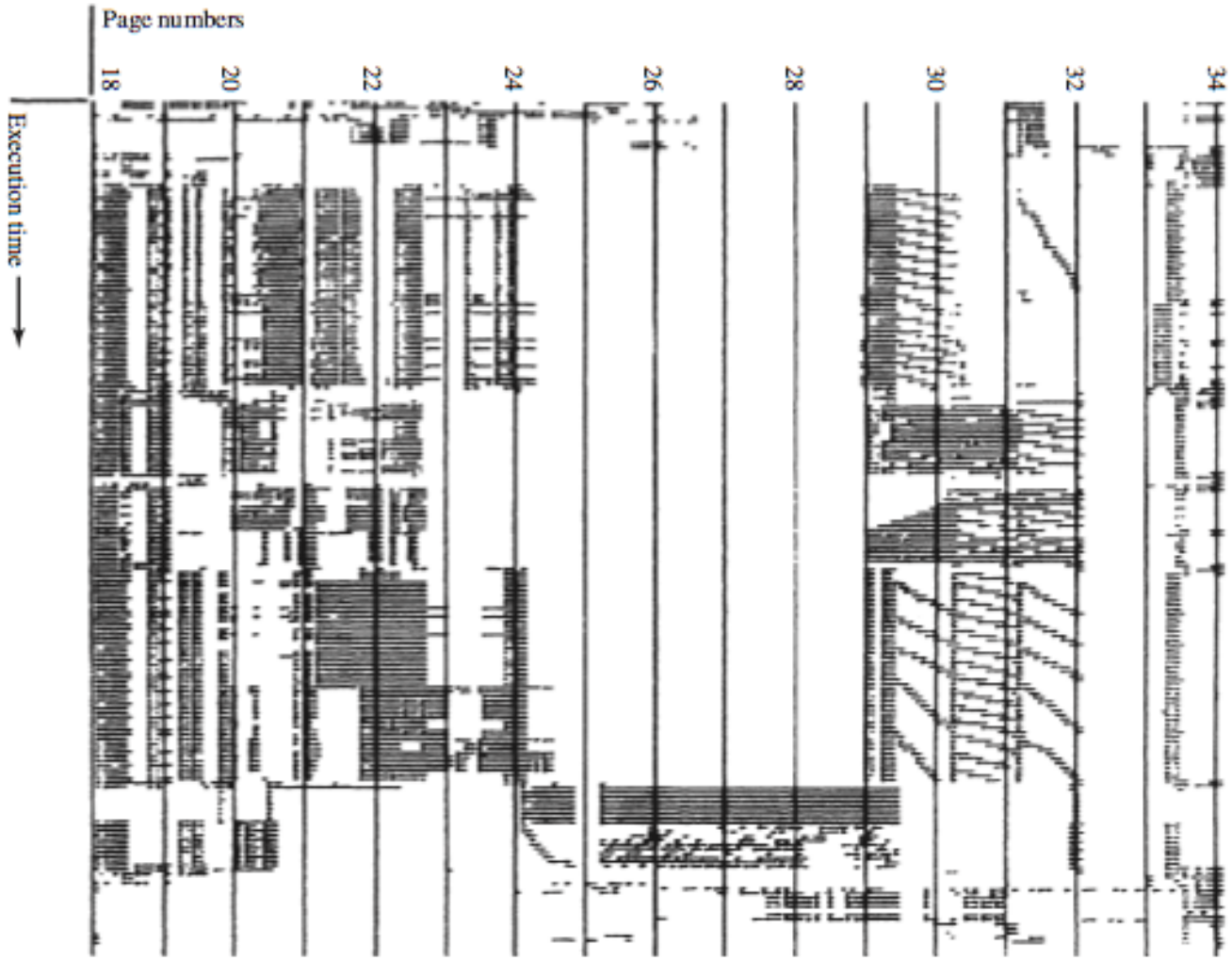


Virtueel geheugen: Idee

- Localiteit: proces gebruikt
  niet alle pagina's/segmenten tegelijk

- $\Rightarrow$ ongebruikte delen hoeven niet in hoofdgeheugen te staan!

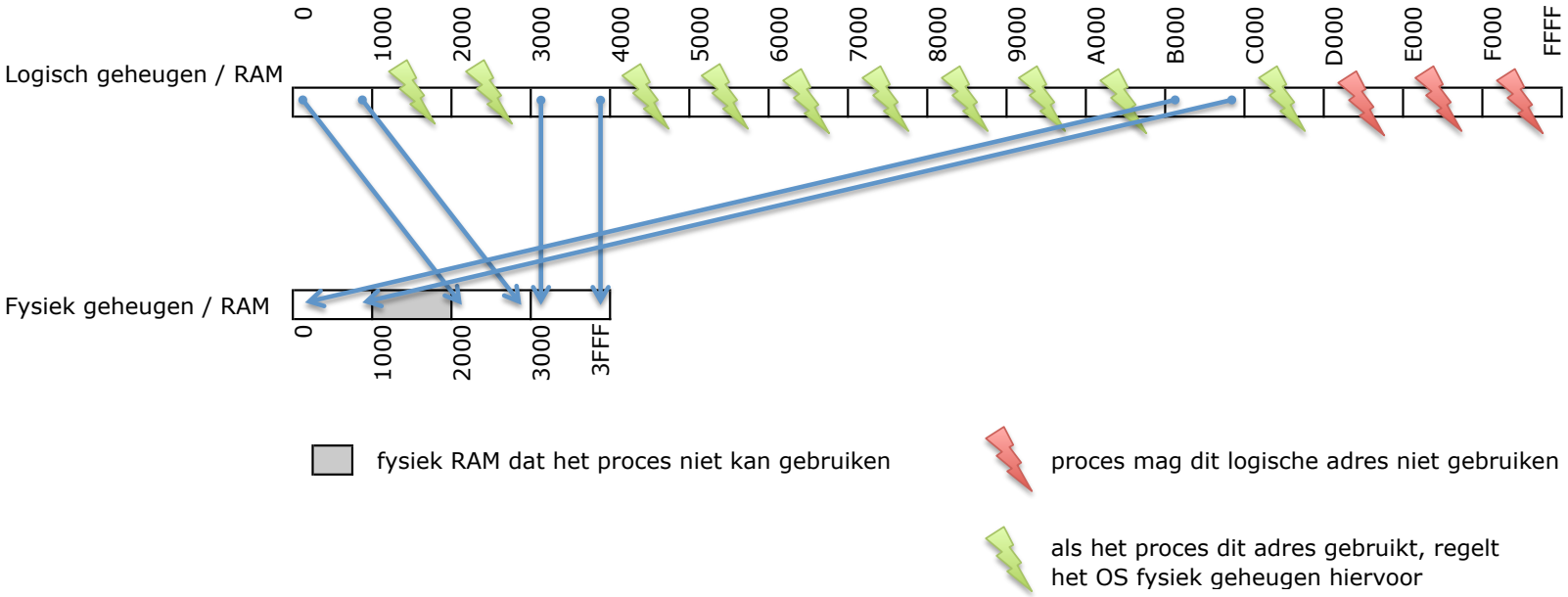
Localiteit



Virtueel geheugen: Uitwerking

- stukjes van proces:
  - sommige in hoofdgeheugen
  - andere op harde schijf
- als proces stukje op harde schijf gebruikt:
  - MMU genereert interrupt
  - OS reserveert nieuw frame/segment
  - OS laadt stukje daarheen
- extra informatie in pagina- of segmenttabel

Virtueel RAM met paginering

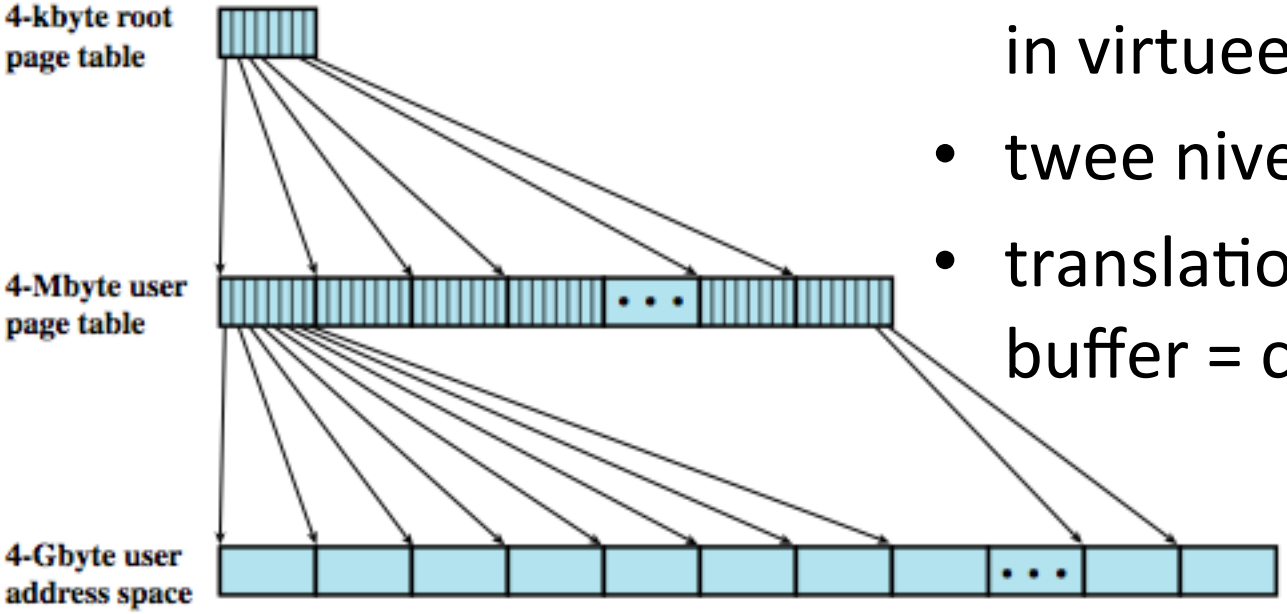


Virtueel geheugen: Voordelen

- meer processen
- grotere processen
- snellere start van processen

- efficiënt! (meestal...)

Meer details over...
Paginatabellen



- paginatabel kan erg groot worden
- idee: paginatabel in virtueel geheugen
- twee niveaus
- translation lookaside buffer = cache

Meer details over... omgekeerde paginatabel

- andere oplossing voor grote paginatabel
- klassiek: logisch → fysiek adres
- omgekeerd: fysiek → logisch adres
  - één entry per fysiek frame
  - grootte van paginatabel
    in verhouding met fysiek geheugen
  - hash-functie voor snelle vertaling logisch → fysiek

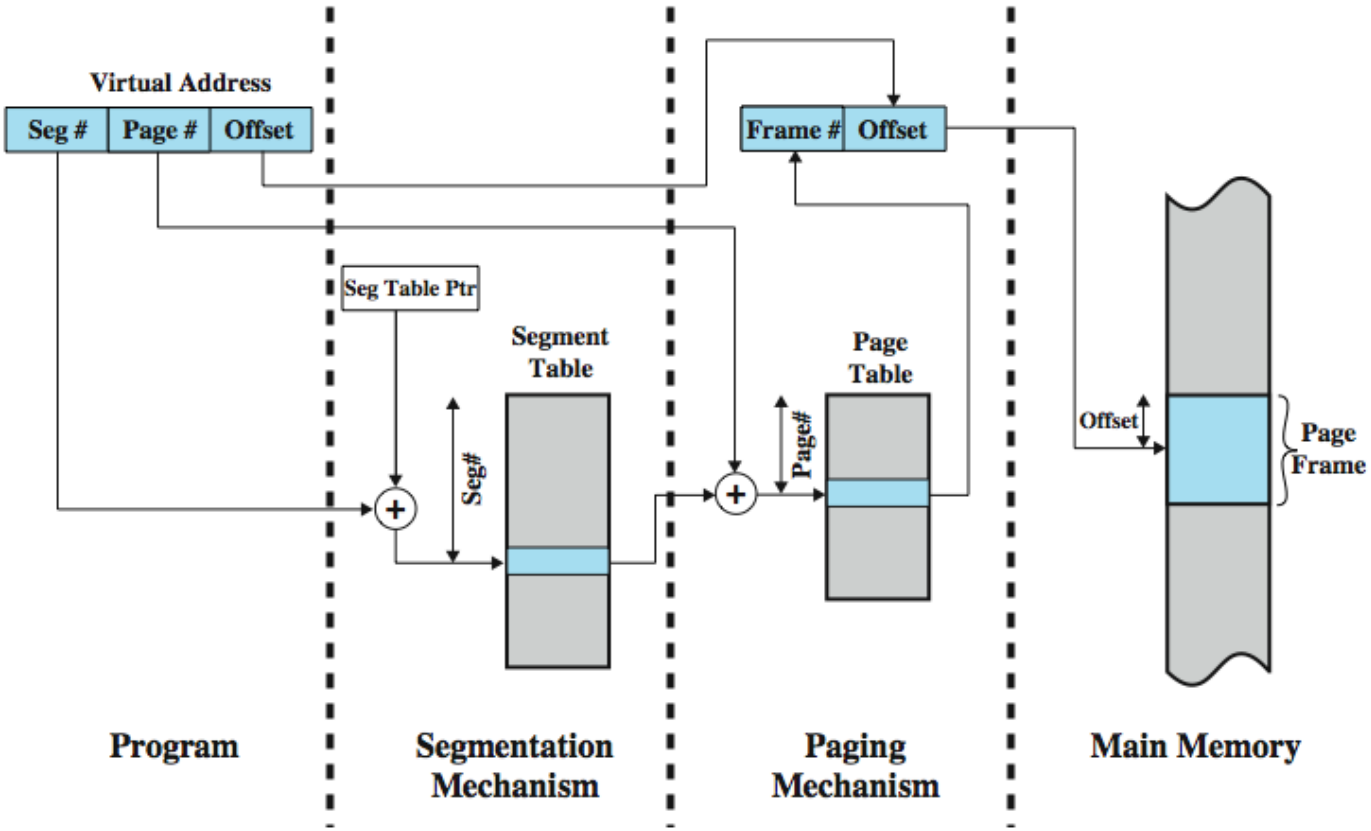
Virtueel geheugen met segmenten

- minder gebruikelijk, ook mogelijk
- zelfde principes als bij paging, behalve segmentgrootte

Alles-in-één-pakket

- segmentatie+paging+virtueel geheugen
- combineert voordelen van beide kanten
- nadeel: ingewikkelde berekening van fysiek adres

Berekening van fysiek adres



Samenvatting

- Doelen van geheugenbeheer:
  Wie weet ze nog?

- meer flexibiliteit door logisch/fysiek geheugen
- geheugen groter laten lijken: virtueel RAM
  - werkt vanwege localiteit