IN1805 I – Operating System Concepten

Hoofdstuk 8: Main memory

Geheugen en Adressering

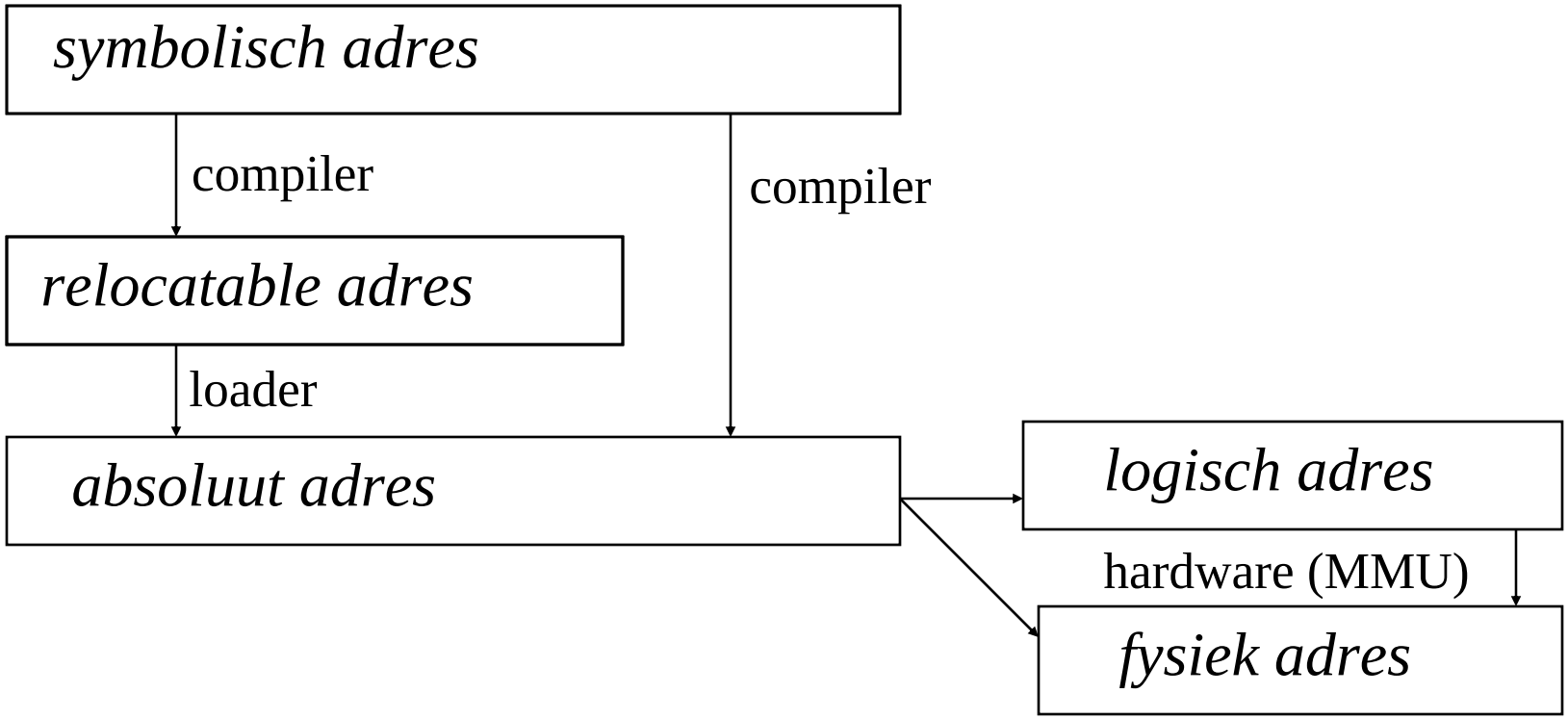
- Geheugen (main memory, primary storage) is noodzakelijk voor de uitvoering van programma's.
- te beschouwen als array van adresseerbare bytes (of woorden).
- verschillende processen moeten het geheugen kunnen delen (sharing)
- adressering op verschillende niveaus verschillend, b.v. :
  - sourcecode van een programma bevat symbolische adressen
  - loadmodule bevat reloceerbare adressen
  - programma in executie bevat absolute adressen
  - fysieke geheugen wordt geadresseerd d.m.v. fysieke adressen

Binding

- Binding is het vertalen ('mapping') van een hoger niveau adres naar een lager niveau adres
- Binding aan het absolute adres kan b.v.
  - tijdens compilatie : symbolische adressen worden omgezet naar absolute adressen. (voorbeeld MS_DOS bij .COM files)
  - tijdens het laden: reloceerbare adressen worden omgezet naar absolute adressen
- Binding aan fysieke adressen wordt gedaan tijdens executie. (speciale hardware nodig).

Binding (2)

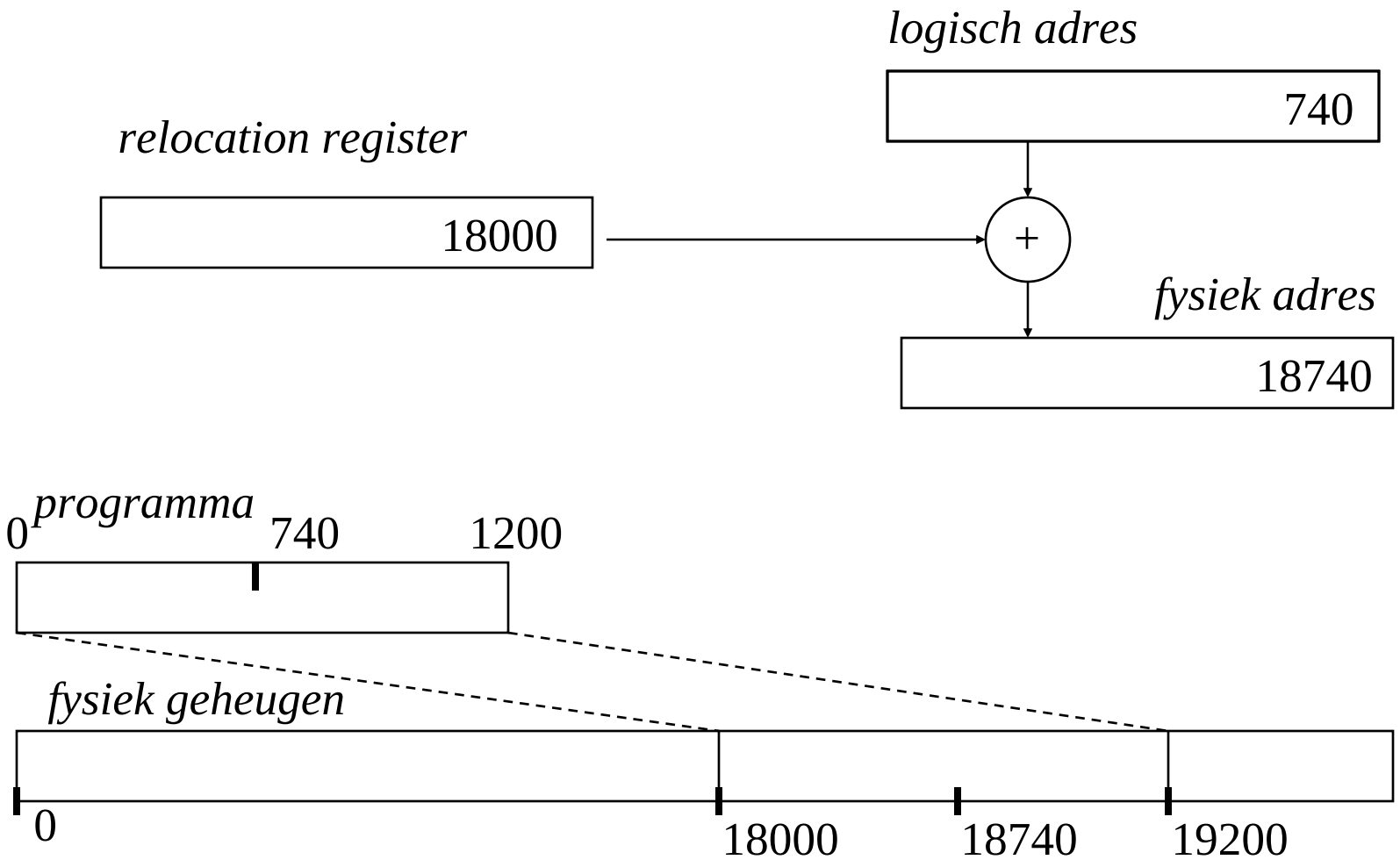
Overzicht verschillende adres typen:



Logische en Fysieke adresruimten

- Bij execution-time binding is het absolute adres niet altijd het echte fysieke adres: er wordt dan onderscheiden *logisch adres* en *fysiek adres.*
- De processor en het proces zien het logische adres (*virtueel adres)*
- Het main memory gebruikt het fysieke adres
- Afbeelden van logische adressen naar fysieke adressen gebeurt door de MMU.
- Eenvoudigste hardware implementatie: d.m.v. relocation register

relocatie register



Dynamic loading

- doel: beter gebruik van main memory
- routines in relocatable vorm op disk
- aanroepende routine
  - kijkt eerst of de aan te roepen routine al aanwezig is
    - zo ja, routine wordt aangeroepen
    - zo nee, routine moet eerst worden geladen
- geen extra voorzieningen in OS nodig
- lastig voor de programmeur

Dynamic linking

- doel: beter gebruik van main memory en shared libraries
- Bij *static linking* zijn alle routines in het loadmodule meegelinkt
- Bij *dynamic linking* wordt voor iedere routine een *stub* , meegelinkt, de routine zelf blijft op schijf en wordt door de stub geladen indien nodig (één copy in het systeem.)
- eenvoudiger voor de programmeur,
- bij aanpassing van de library hoeven niet alle loadmodules opnieuw gelinkt te worden
- extra support van OS nodig.

Swapping (1)

Swapping kan worden gebruikt bij een proces switch (b.v. bij Round Robin scheduling)

- *Swap-out* is het verplaatsen van een proces naar backing store
- *Swap-in* is het binnenhalen van een proces van backing store
- *Swap* is een swap-out gevolgd door een swap-in,
- ook genoemd: *Roll-out, Roll-in*

- Na een swap-in moet een proces op zijn oude plek terugkomen, tenzij execution-time binding wordt toegepast

Swapping (2)

Problemen

- Swap-in en swap-out tijden zijn aanzienlijk, circa 250 ms voor 10 MB (in één richting), dus omvang van het in gebruik zijnde geheugen beperkt zien te houden.
- proces moet idle zijn bij swap-out
- proces mag geen pending I/O hebben , tenzij de I/O buffers geleverd zijn door het OS

PAUZE

Verdeling logische geheugenruimte over fysieke geheugenruimte

- Contiguous:

  logische geheugenruimte wordt afgebeeld op een aaneengesloten blok fysiek geheugen

- Non-contiguous

  logische ruimte wordt gespreid over de fysieke ruimte afgebeeld . b.v. via *paging* of via *segmentering*

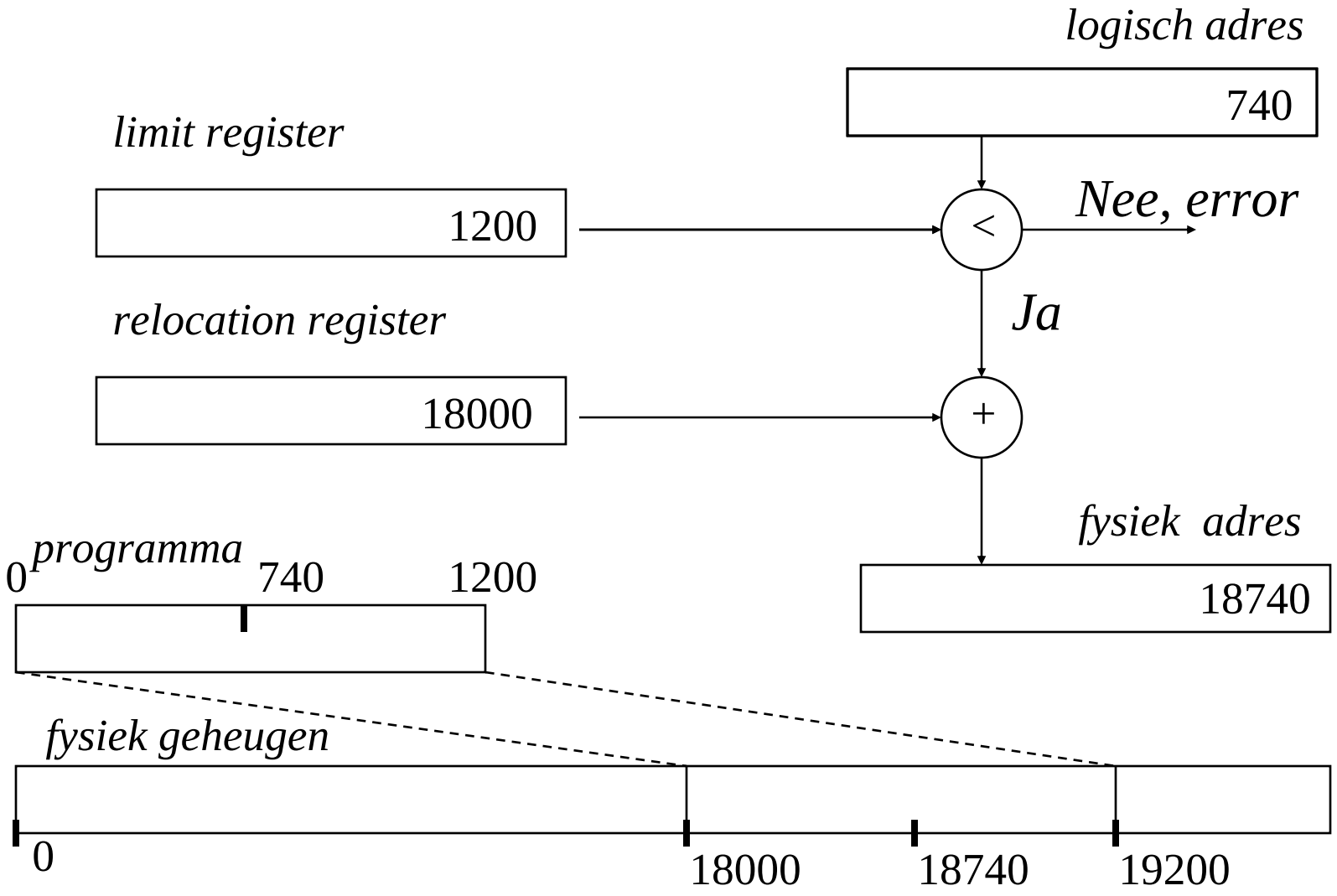
Contiguous allocation (1)

- Single partition

  (OS en 1 userproces)
  - hardware support: relocation register + limit register
- Multiple partitions

  (OS + N user processes)
  - Vaste grootte van partitions (b.v. IBM: MFT, fixed)
  - Variabele grootte van partitions (b.v. IBM: MVT, variable)
    - systeem bevat verzameling processen van verschillende grootte, en verzameling vrije ruimten van verschillende grootte.
    - Probleem van Dynamic Storage Allocation

relocation en limit register



Contiguous allocation (2)
Dynamic Storage Allocation probleem

Gegeven: een aantal vrije ruimten van verschillende omvang, en een programma van een bepaalde grootte.

Gevraagd: In welke vrije ruimte kan dit programma het best geplaatst worden?

Mogelijkheden:

- First fit (eventueel cyclisch)
- Best Fit
- Worst Fit

First fit en Best fit equivalent qua geheugenbezetting,

First fit meestal sneller

Contiguous Allocation (3)
Interne en Externe fragmentatie

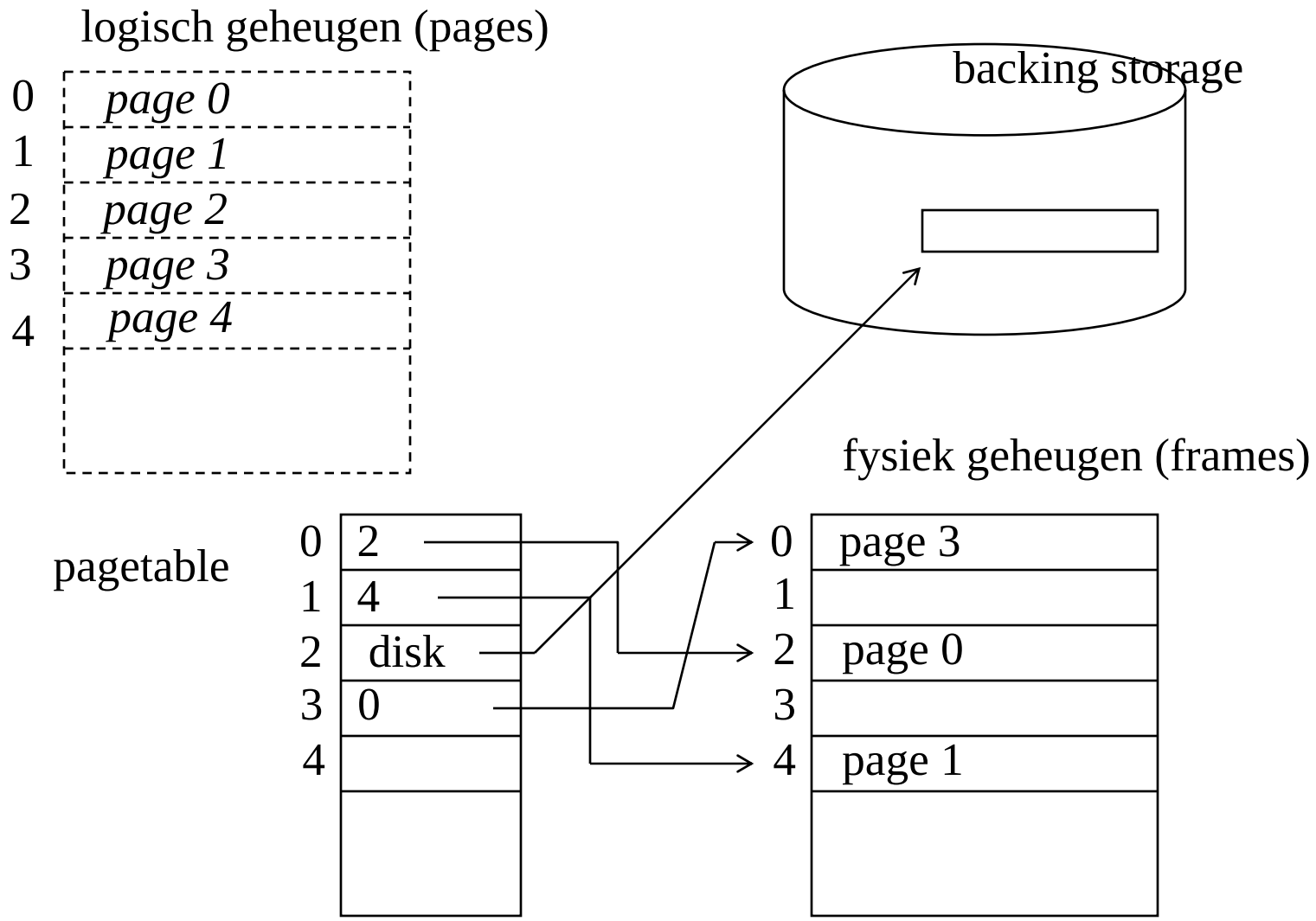
- Externe fragmentatie: Totale vrije ruimte is voldoende, maar er is niet voldoende aaneengesloten vrije ruimte om aan een bepaalde vraag te kunnen voldoen.
- Interne fragmentatie: verlies als gevolg van het verschil tussen benodigde grootte en verkregen ruimte als gevolg van de omvang van de eenheden waarin ruimte beschikbaar wordt gesteld.
- Algemeen: Bij N gealloceerde blokken gaan gemiddeld

  0.5 N blokken verloren door externe fragmentatie (50% regel, gemeten bij first-fit)

Non-contiguous allocation (1)
paging

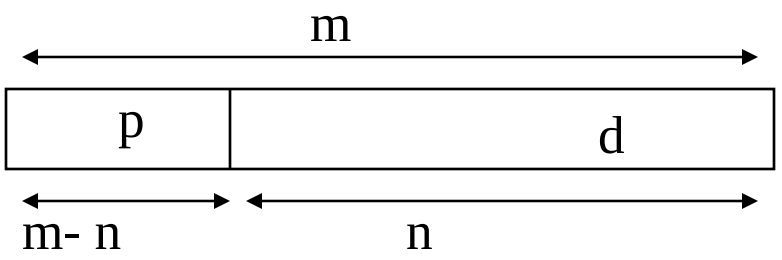
- Fysiek geheugen opdelen in *frames* van grootte S
- logisch geheugen opdelen in *pages* van grootte S
- pages worden geplaatst in frames
- op backing storage blokken reserveren van grootte S
- logisch adres bestaat uit *pagenummer* (p) en *pageoffset* (d)
- vertaling van logisch naar fysiek adres met behulp van een *pagetable*
  - voor iedere page 1 entry in de pagetable
  - pagenummer is index in de tabel
  - entry bevat framenummer plus enkele statusbits
  - er is één pagetable per proces

Non-contiguous allocation (1A)
paging voorbeeld

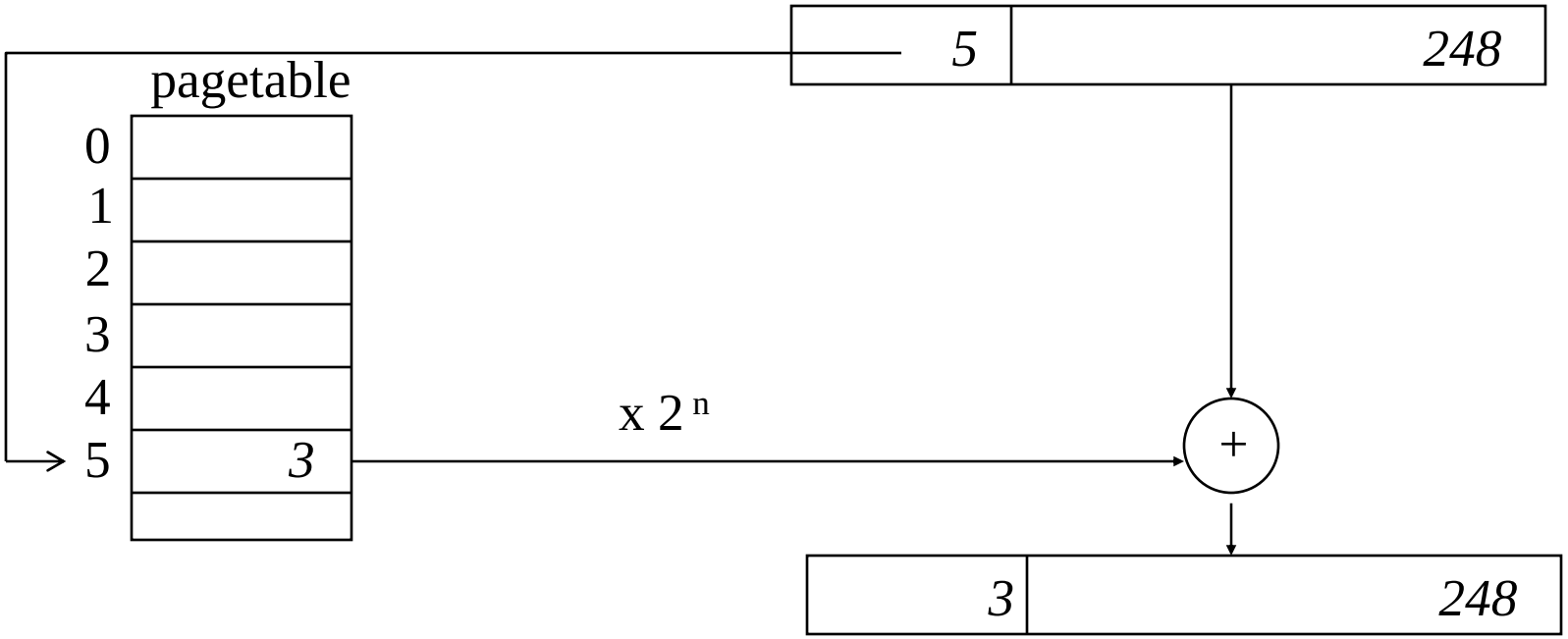


Non_contiguous allocation (1B)
paging, voorbeeld: adres vertaling

Logisch adres =
pagenummer + displacement:



voorbeeld: m=32, n=12



Non-contiguous allocation (2)
pagesize

- De pagina-grootte ligt vast in de architectuur van de machine
- voordelen van grote pagesize:
  - kleine pagetable
  - efficienter disk I/O
- nadeel van grote pagesize:
  - interne fragmentatie
- pagesizes typisch in de range van 512 tot 8192 bytes (altijd een macht van 2)

Non-contiguous allocation (3)
Hardware support voor paging

Hardware support:

- Eenvoudigst:
  - aantal speciale registers (kan alleen bij kleine pagetable)
- meestal:
  - pagetable in main memory,
  - 1 speciaal register wijst naar de pagetable , wordt geladen bij de context switch

  probleem: voor vertalen van een adres is een geheugenaccess nodig

  oplossing: *Translation Lookaside Buffer (TLB)*

Non-contiguous allocation (4)
TLB

Translation Lookaside Buffer (TLB),

- bestaat uit een aantal *associatieve registers*
- bewaart de N meest recent vertaalde pagenummer,framenummer paren
- vertalen begint met kijken in TLB.
- Als pagenummer niet in TLB, dan wordt vertaald met behulp van de pagetable. (resultaat naar TLB)
- *hitratio* moet hoog zijn (b.v. 90%)
- TLB wordt geleegd bij een contextswitch

Non-contiguous allocation (5)
protectie en paging

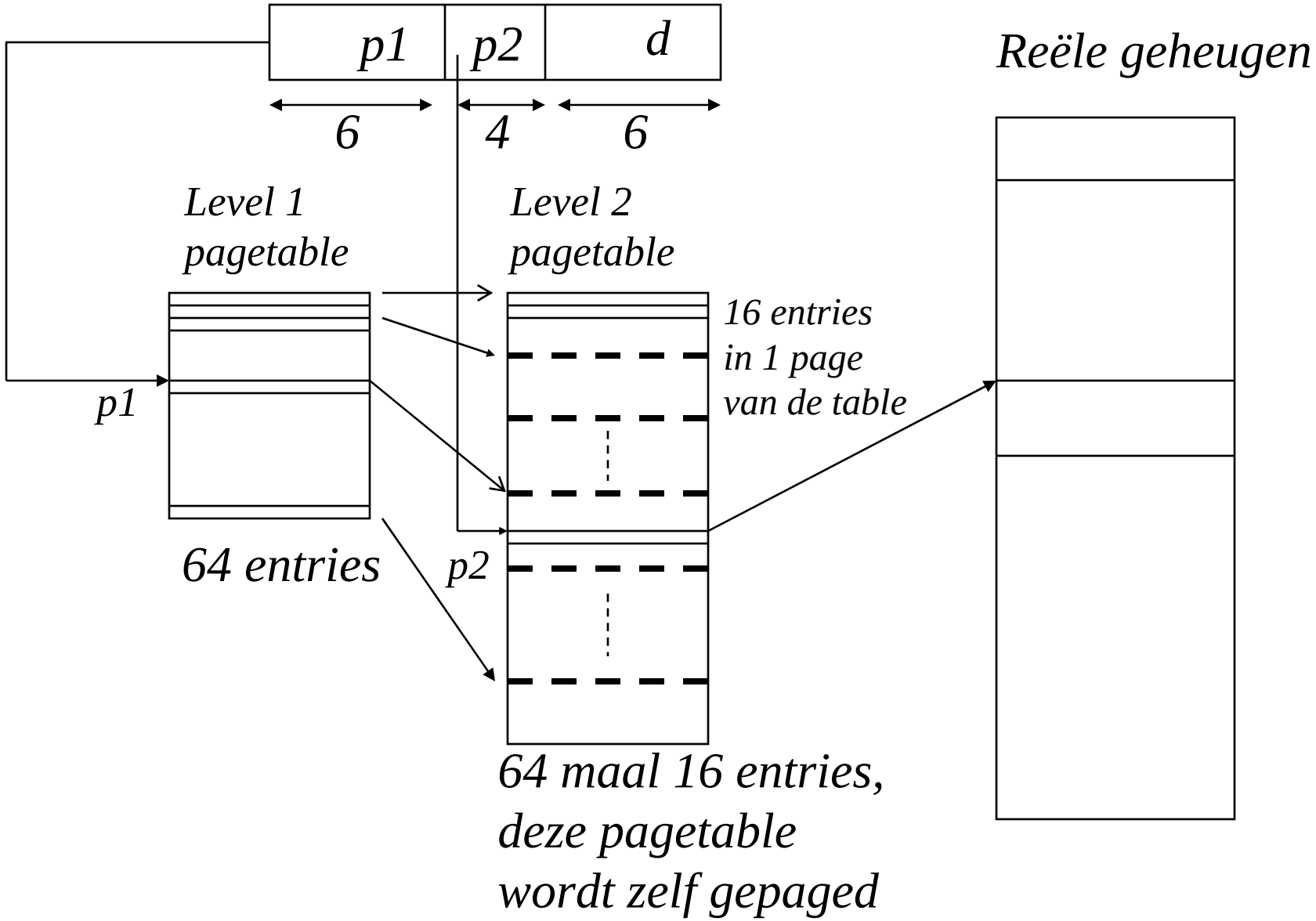
- per page protectiebits in de pagetable op te nemen
- een page kan b.v. read-only zijn
  - Bij adresvertaling wordt dan geverifieerd of schrijven is toegestaan
  - Een poging tot schrijven naar read-only page geeft een trap
- Per page ook een valid/invalid bit op te nemen
  - valid/invalid bit geeft aan of de page in de logische adresruimte zit
  - adressseren van een invalid page veroorzaakt ook een trap
- soms wordt het geldige deel van de pagetable aangegeven door een page-table length register

Non-contiguous allocation(6)
multilevel paging (hierarchical paging)

- Bij grote logische adresruimte: grote pagetable nodig, past niet in het geheugen
- oplossing: multilevel paging, b.v.
  - 2-level paging bij 32-bit adressen
  - 3- of 4-level paging bij 64-bit adressen
- bij n-level paging : n+1 geheugenaccesses nodig voor adresvertaling: caching vereist.

Alternatief voor hierarchical paging: *hashed pagetables*

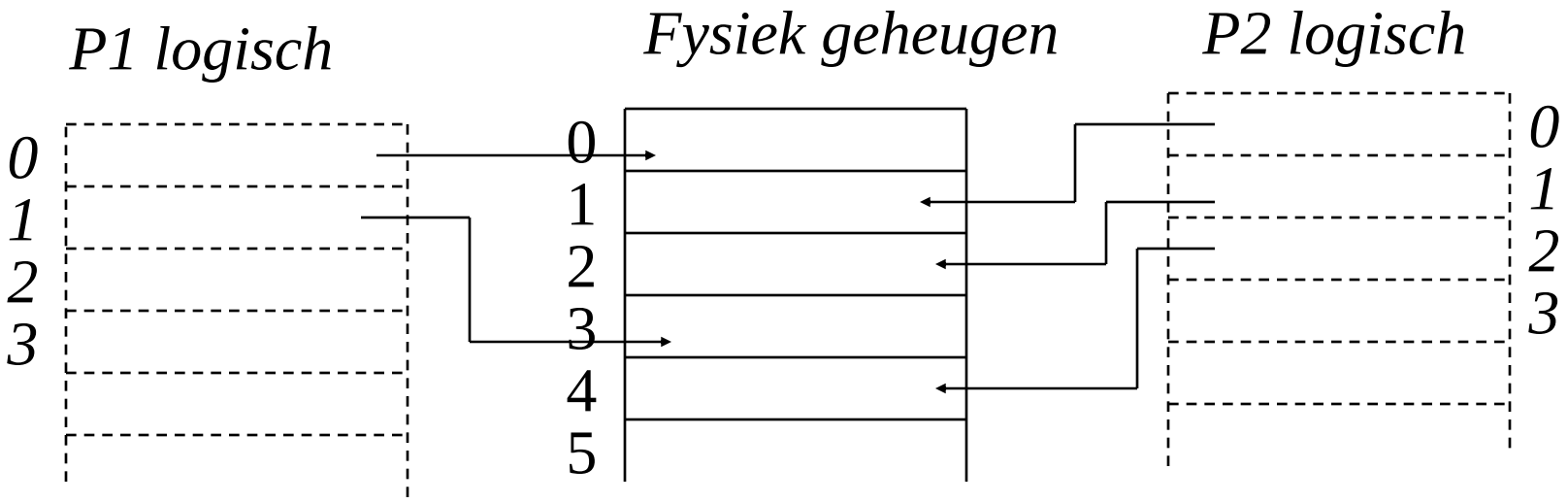
Multilevel paging voorbeeld met 16 bits adressen



Non-contiguous allocation(7)
Inverted pagetables

- Eén *inverted pagetable* voor het hele systeem, i.p.v. een pagetable per proces.
- Inverted pagetable bevat per frame:
  - procesid (pid) van eigenaar van de page in het frame
  - logisch adres van deze page
- Inverted pagetable bevat geen informatie over de niet aanwezige pages. (Hiervoor *externe pagetable* nodig)
- Bij pagefault wordt externe pagetable gebruikt (deze kan zijn uitgeswapt)
- zoeken in inverted pagetable te versnellen door hash-table.

Inverted pagetable: voorbeeld



Inverted pagetable

0	*1,0*
1	*2,0*
2	*2,1*
3	*1,1*
4	*2,2*

beschrijft de inhoud van het fysieke geheugen

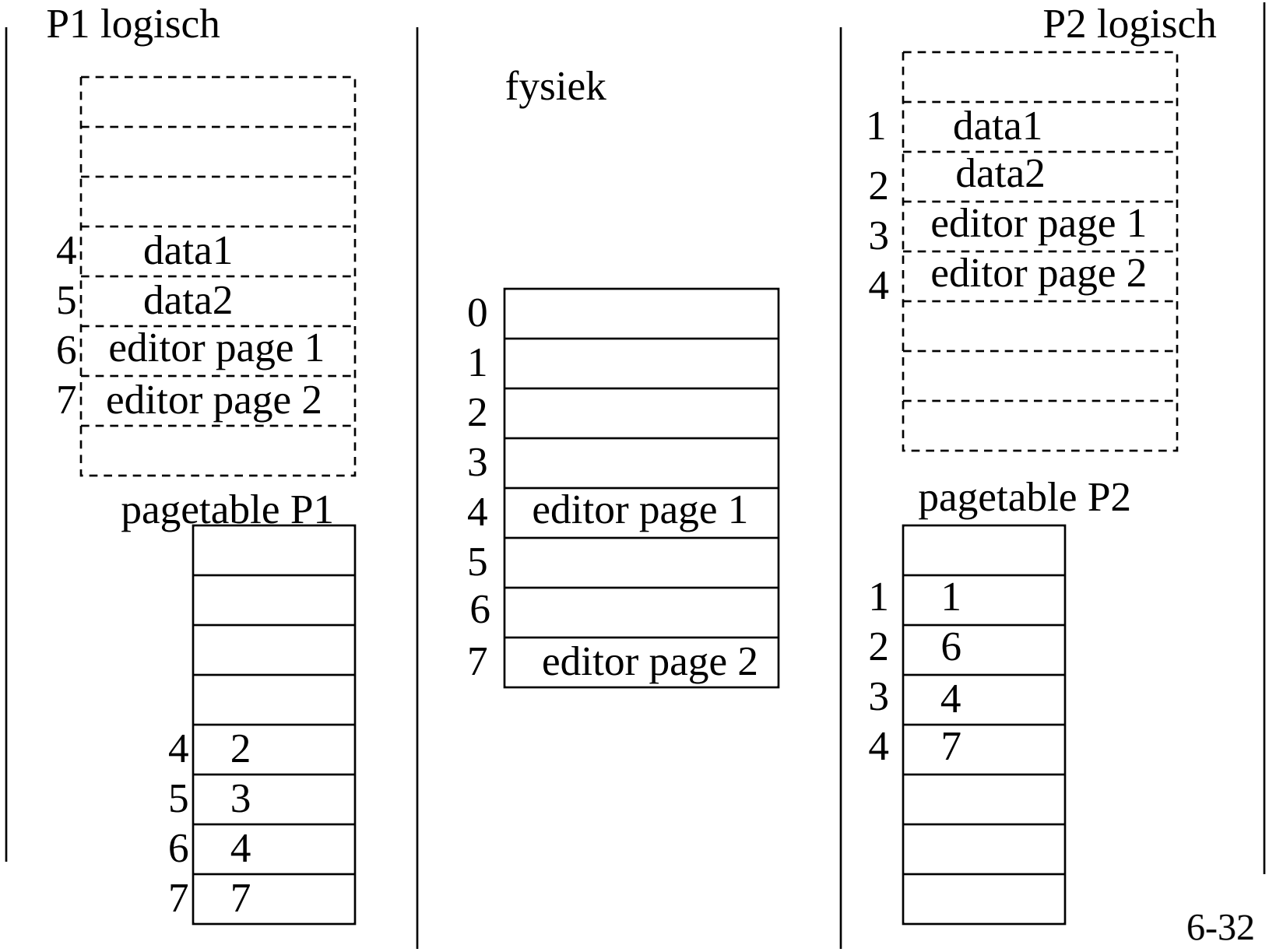
adresvertaling :
voor b.v proces 1, page 1
zoek in inverted pagetable naar 1,1
(resultaat frame 3)

Non-contiguous allocation (8)
shared pages

- Sharing van pages mogelijk bij *reentrant* code
- Bij sharing in verschillende pagetables verwijzingen naar dezelfde frame(s)
- moeilijk te realiseren bij inverted pagetables

Non-contiguous allocation(8A)

shared pages. voorbeeld



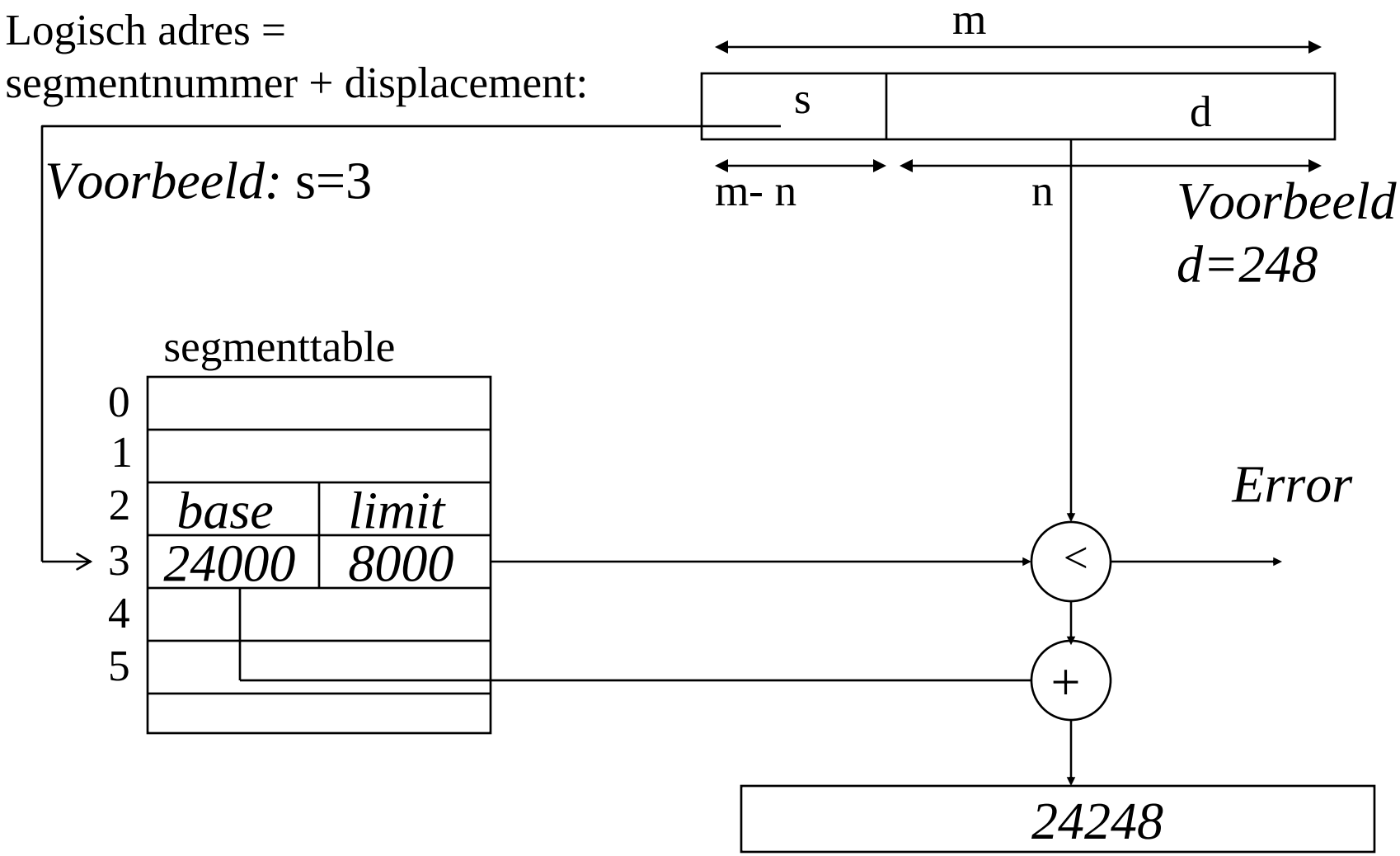
Non-contiguous allocation (9) segmentering

- Segmentering sluit aan bij het model dat een gebruiker in zijn programma van het geheugen heeft.
- een programma wordt ingedeeld in een aantal zinvolle segmenten, b.v.
  - verschillende code segmenten voor verschillende routines
  - verschillende data segmenten voor verschillende arrays
- Ieder segment heeft een nummer en een lengte
- te vergelijken met paging met variabele pagelengte

Non-contiguous allocation (10)
segmentering (vervolg)

- adres bestaat uit een segmentnummer en een segment offset
- adresvertaling met behulp van een segmenttabel
- per proces een segmenttabel
  - index is segment nummer
  - entry bevat beginadres , **lengte,** en statusbits

Non_contiguous allocation (10B)
segmentering, voorbeeld: adres vertaling



Non-contiguous allocation (11) segmentering, sharing en protectie

- protectie op basis van segmenten,
  - maakt het mogelijk binnen een proces het codedeel te beschermen tegen overschrijven, en het datadeel wel lees- en schrijfbaar te maken
  - realisatie door protectiebits op te nemen in segmenttable-entries
- segmenten te sharen door processen
  - een shared datasegment mag in verschillende processen verschillende segmentnummers hebben

Non-contiguous allocation (12)
fragmentatie bij segmentering

- Externe fragmentatie treedt op als gevolg van segmenten van verschillende lengte. (vergelijk contiguous allocation met variabele lengte)
- Opheffen van externe fragmentatie door compaction
- In het algemeen fragmentatie te verminderen door:
  - kleinere segmenten
  - segmenten van vaste lengte, *maar dat heet paging* ....

Segmentering en Paging

- Vaak wordt een combinatie van segmentering en paging gebruikt.
- Eerst segmenteren, daarna per segment een pagetable

voorbeeld: Intel 386 en hoger

- segmenten van max 4 GB (32 bits offset)
- pages van 4 KB
- per segment 2 -level pagetable (20 bits pagenummer gesplitst in 10 bits pagetablenummer en 10 bits pagenummer)