

Toegepaste Informatica

MBI07a

2022-2023



**UCLL**  
HOGESCHOOL



Computer  
Systems

## Memory & Memory Management

Jeroen Jean, Rudi Swennen, Tiebe Van  
Nieuwenhove, Frédéric Vogels

# Memory

Nut van memory

Memory architecture

Memory management OS

# Nut van memory

- Uitvoerbare code van elk proces zit in memory (of toch een deel)
- Variabelen worden in memory geplaatst.
- Programma starten duurt even
  - Deel van programma wordt gekopieerd naar RAM
  - Dan pas kan CPU het gebruiken (cfr Week 7: CPU)

loading...



Why?

# Nut van memory

- Snelheid:
  - SSD: +/- 50 microseconden om te lezen/schrijven
  - RAM: +/- 17 nanoseconden om te lezen of te schrijven
- RAM = 3000 keer sneller dan SSD

Met RAM



Zonder RAM



# Memory architecture: HW

- SDRAM (**S**ynchronous **D**ynamic **R**andom **A**ccess **M**emory)
- DIMM (**D**ual **I**ndline **M**emory **M**odule):
  - 288 'pins' (verdeeld over beide zijden = dual)
    - Data
    - Control
    - Address
  - 4 of 8 IC's (Integrated **C**ircuits)

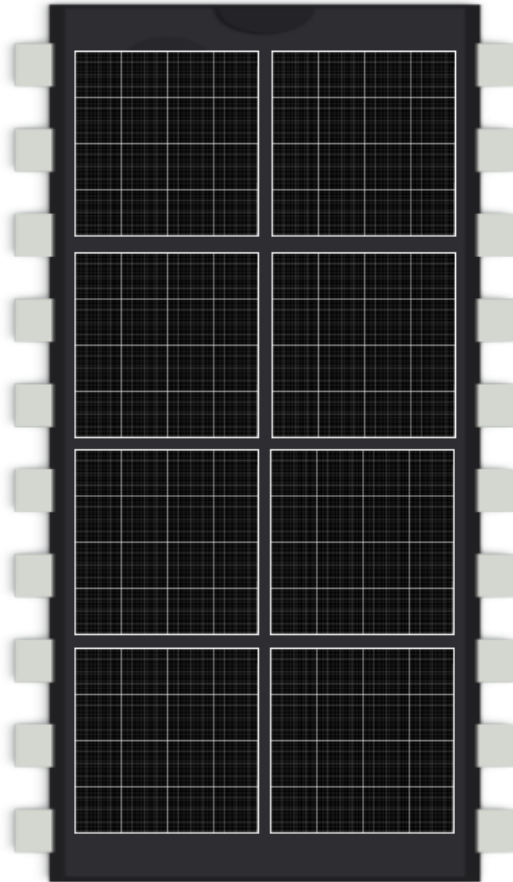


# Memory architecture: HW

- SODIMM (Small-Outline Dual Inline Memory Module):
  - 260 'pins'
- Vooral voor laptops en notebooks



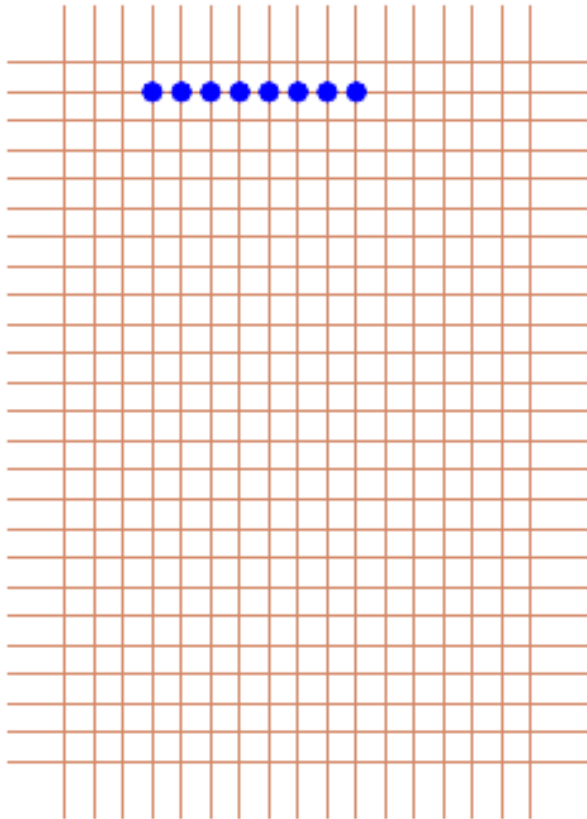
# Memory architecture: HW



- Elke IC is opgebouwd uit verschillende **bankgroepen** met hetzelfde aantal **banks**
- **Bank = Matrix** (rijen en kolommen)
  - Aantal kolommen blijft steeds hetzelfde
  - Aantal rijen kan veranderen:
    - Afhankelijk van de totale opslagruimte van IC

# Memory architecture: HW

- Waar wordt de eigenlijke data bewaard?

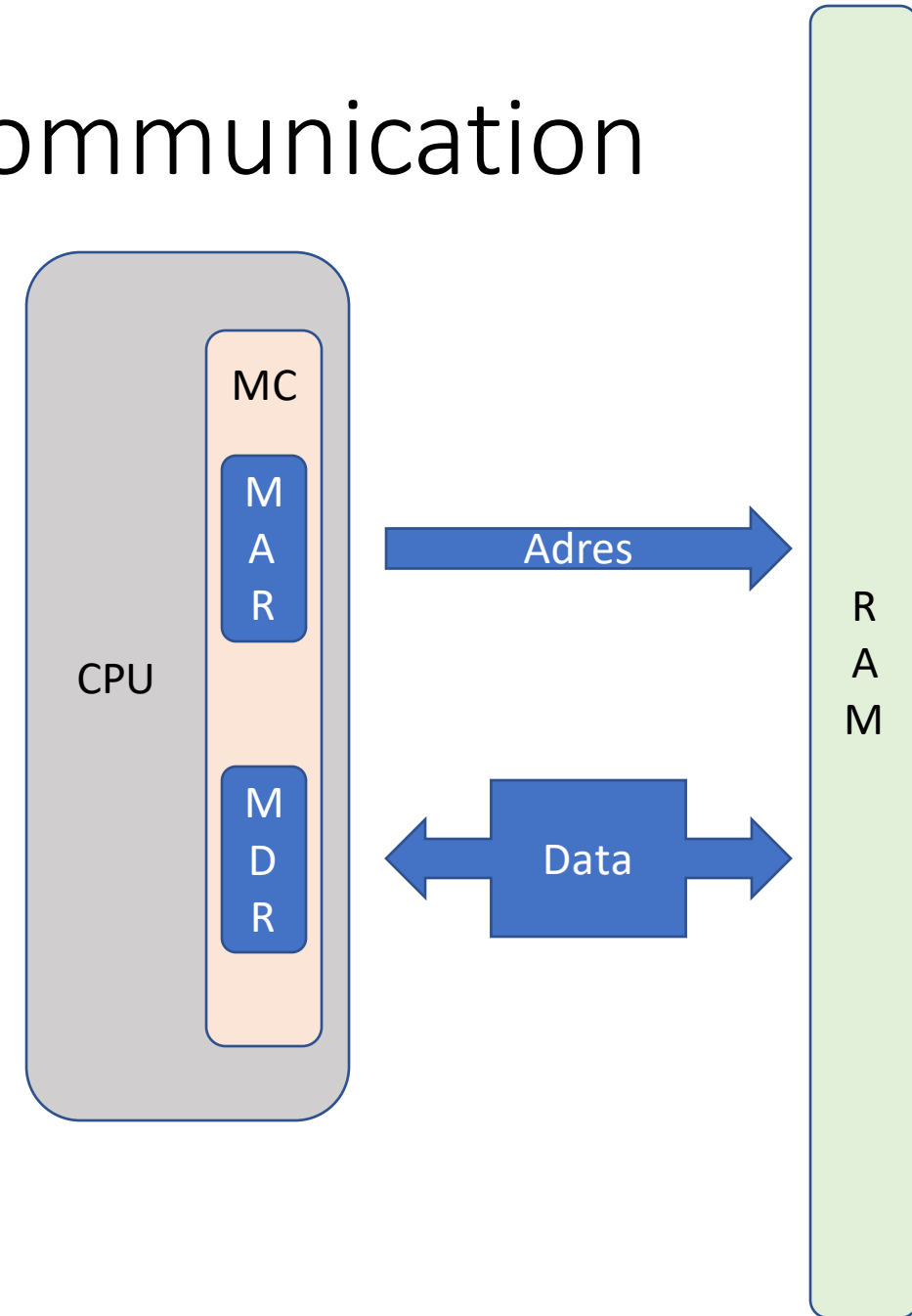


- **Kruising** lijn en kolom = 1 bit
  - dmv transistor:
    - 0 V = binaire 0
    - 1,2 V = binaire 1



# Memory architecture: communication

- MC (**M**emory **C**ontroler) gebruikt 2 registers:
  - MAR: **M**emory **A**ddress **R**egister
    - Voor fysiek adres van geheugen
  - MDR: **M**emory **D**ata **R**egister
    - Lees of schrijf data



# Memory Architecture: HW

- Verloop lees/schrijf opdracht:
  - CPU vraag logisch adres aan MC
  - MC berekent correct fysisch adres
  - Stuur adres naar DIMM(s)
  - Bits worden teruggestuurd naar MC (lezen)  
*of*
  - Bits worden bewaard in transistors (schrijven)

# Memory controller: HW

- Adres bestaat steeds uit aantal delen:

- Bankgroep
- Banknummer
- Rijnummer
- Kolomnummer

- Hoe wordt correcte DIMM gekozen?

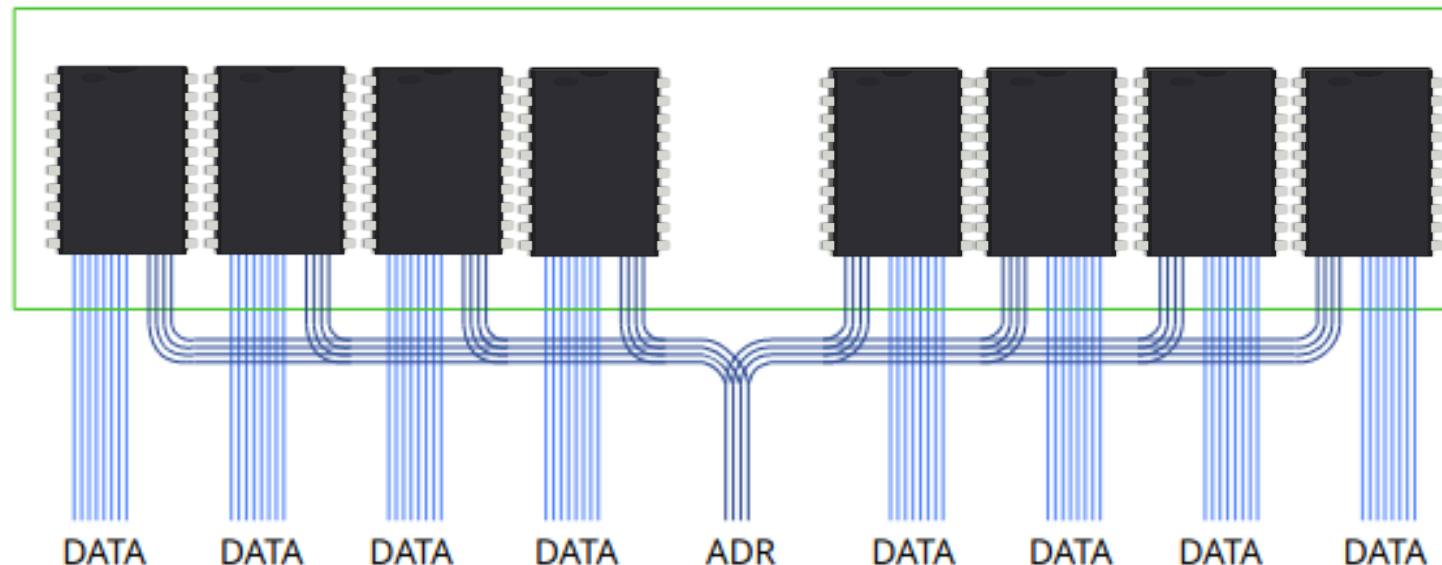
- Aparte controle lijn, om correct DIMM te activeren



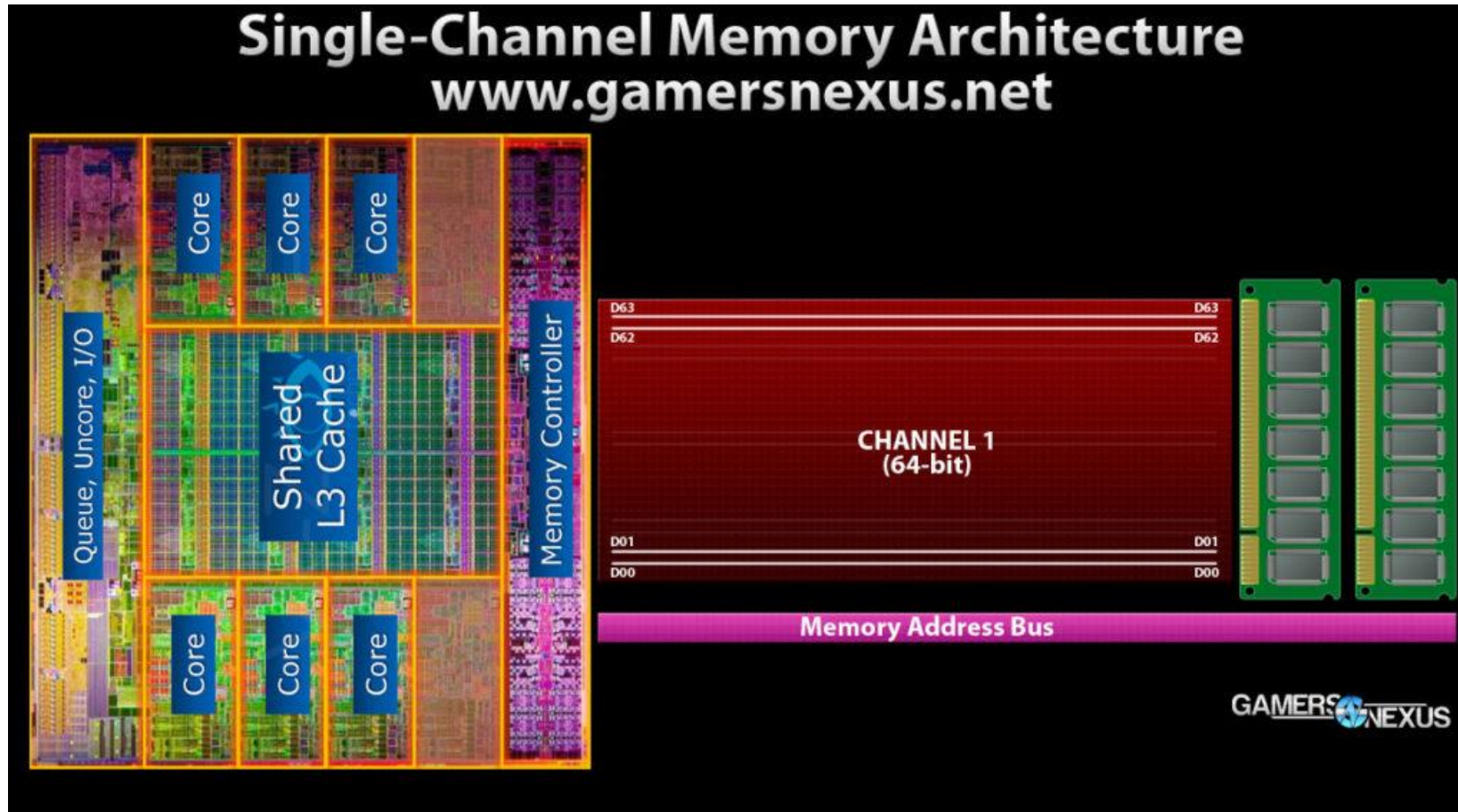
# Memory architecture: HW

- Elke IC, heeft zijn eigen data lijnen (8 bit/IC).
- Zelfde adres wordt naar elk IC gestuurd
- Vanaf kolomnr x worden 8 rijen verbonden met data uitgang IC

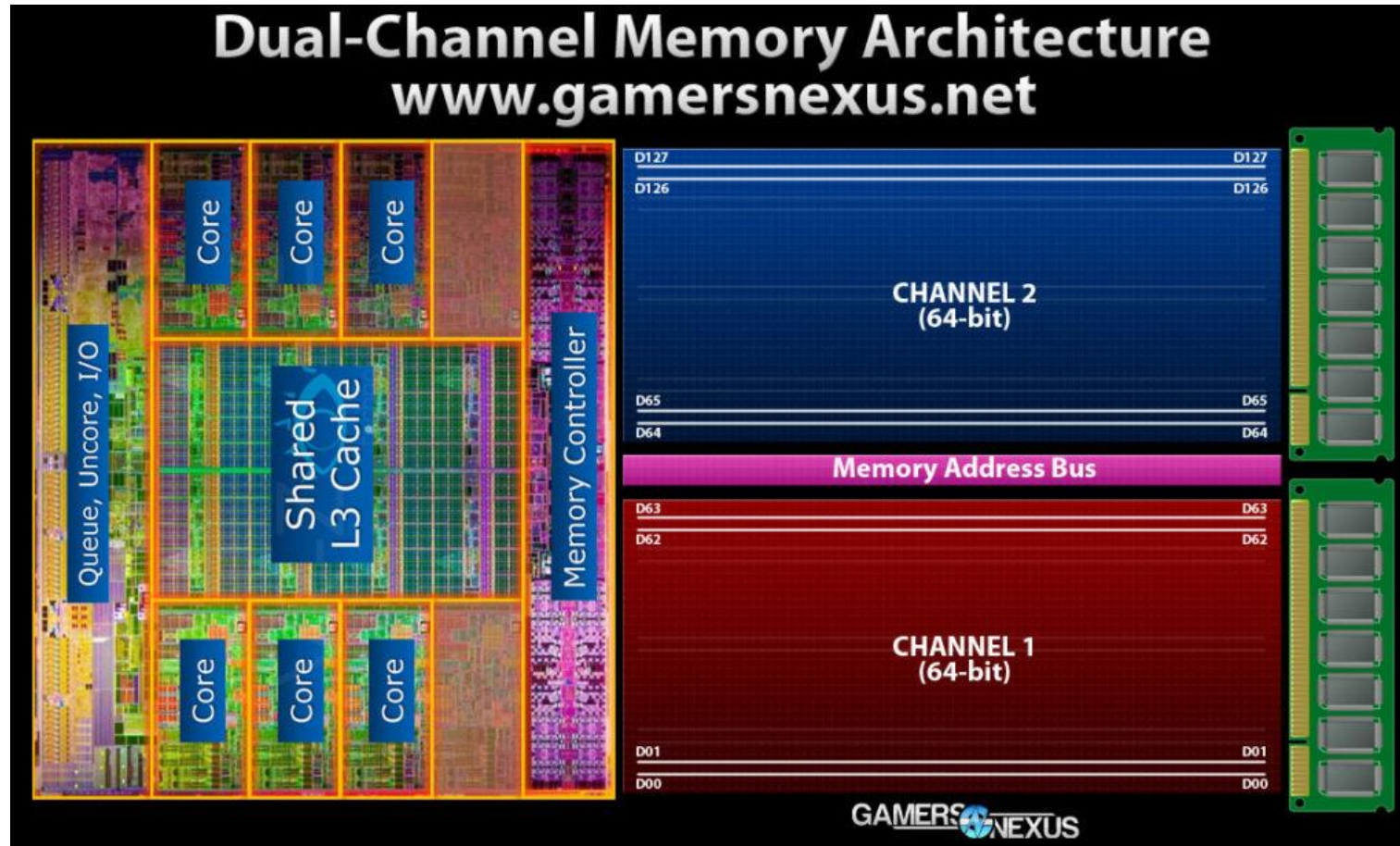
$$8 \text{ (rijen)} * 8 \text{ (IC's)} = 64 \text{ bit}$$



# Memory architecture: single channel

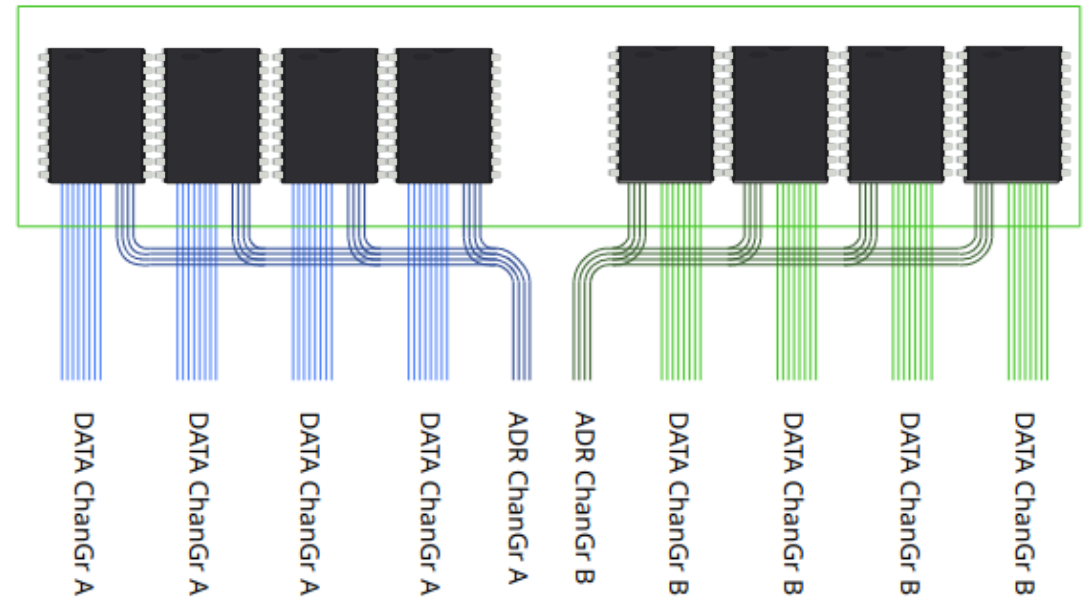


# Memory architecture: dual channel



# Memory architecture: HW

- **DDR5:**
  - Per Channel, 2 channelgroups
    - Apart aanspreekbaar
    - 32 bit per channelgroup
  - Veel hogere bandbreedte
- = Sneller op alle vlakken





# Memory architecture: Perf Gain

- DDR5 biedt heel wat performance upgrades.

Features	DDR4	DDR5	DDR5 Advantages
Speed	1.6 to 3.2 GT/s 0.8 to 1.6 GHz clock	4.8 to 8.4 GT/s 1.6 to 4.2 GHz clock	Higher bandwidth
IO Voltage	1.2 V	1.1 V	Lower power
Power Management	On motherboard	On DIMM PMIC	Better power efficiency Better scalability
Channel Architecture	72-bit data channel (64 data + 8 ECC) 1 channel per DIMM	40-bit data channel (32 data + 8 ECC) 2 channels per DIMM	Higher memory efficiency Lower latency
Burst Length	BC4, BL8	BC8, BL16	Higher memory efficiency
Max. Die Density	16Gb	64Gb	Higher capacity DIMMs
More Intelligence	SPD (I <sup>2</sup> C)	SPD Hub & Temperature Sensors (I <sup>3</sup> C)	Enhanced system management Greater telemetry for thermal management



# Refresh

- Transistor verliest beetje stroom.

==> Zelfs onder spanning, lekt er stroom



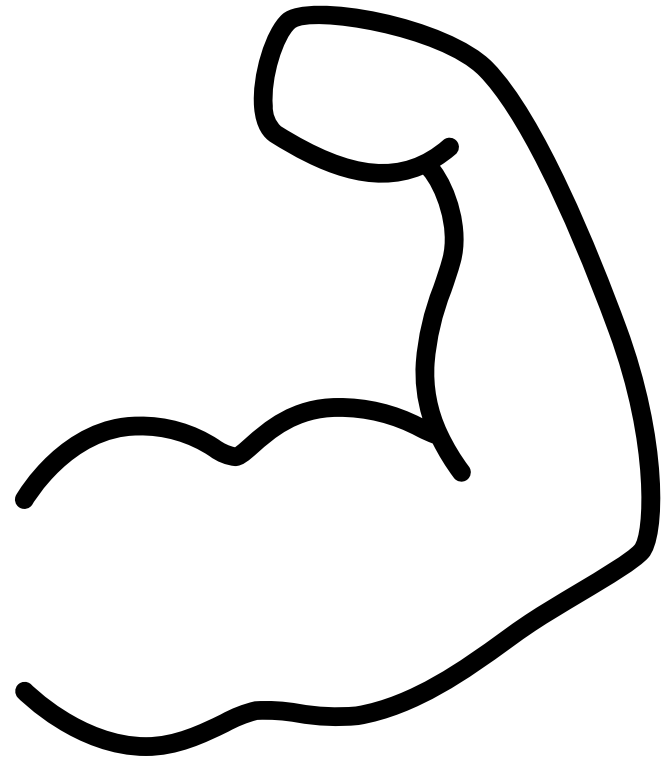
- Oplossing:

- Elke 64ms alle transistor refreshen
- $3\text{ms}/\text{refresh} = 5 \text{ nanoseconden per rij}$

# Memory architecture

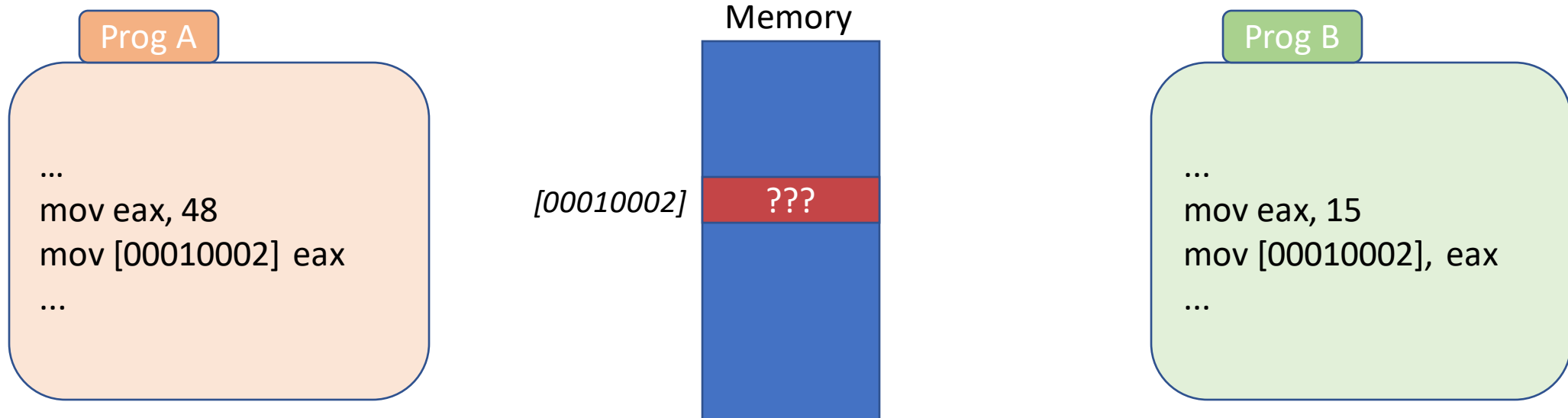
- 16 refresh / seconden
- 4000 – 4800 miljoen R/W opdrachten / seconden

***" DRAM is very powerfull "***



# Memory management: Segmentatie

- OS gebruikt enkel logische adressen.
  - Van byte 0 tot byte ? (cfr LBA)
- Programma's mogen voorbepaalde logische adressen niet gebruiken.

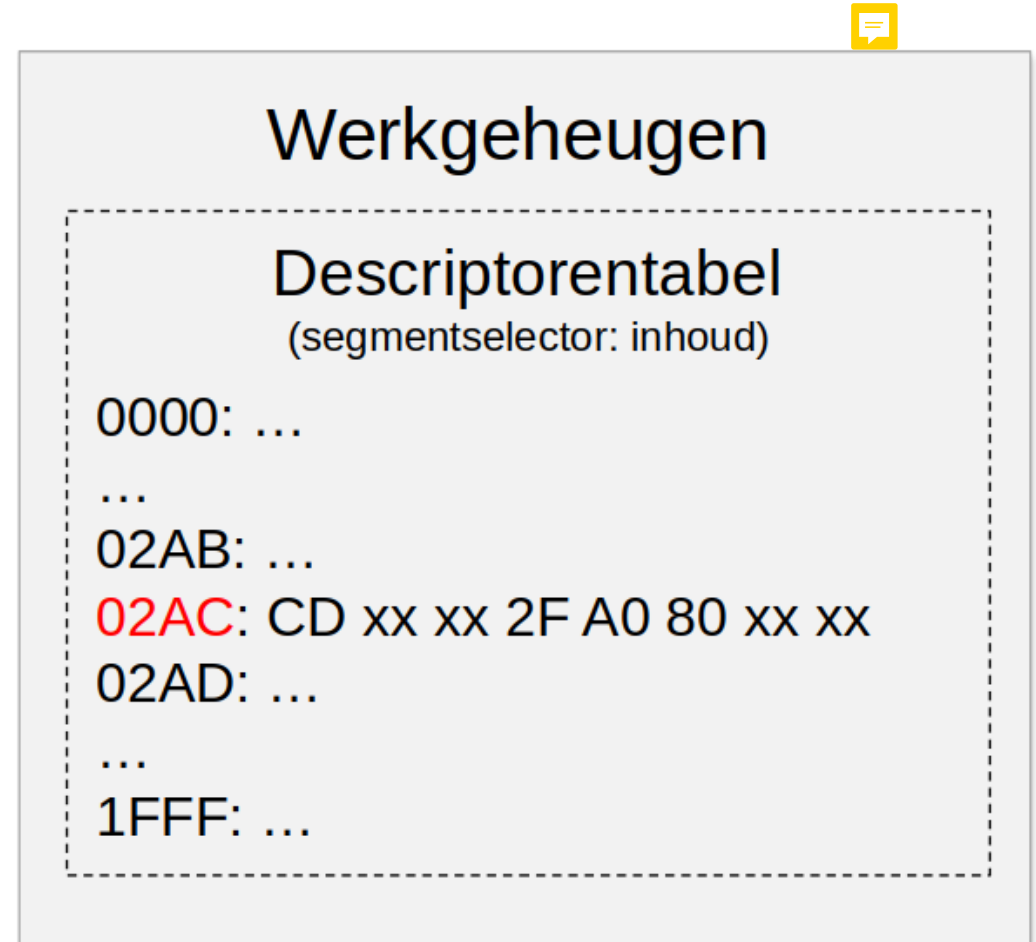


# Memory management: Segmentatie

- Voor elk programma wordt één byte in het geheugen gekozen als beginpunt.
  - Absoluut adres van beginpunt = basis.
- Bytes voorbij de basis: aanduiden met hun verplaatsing t.o.v. de basis:
  - $\text{adres} = \text{basis} + \text{verplaatsing}$


# Memory management: Segmentatie

- Elk programma in uitvoering krijgt een segment van het geheugen:
  - Code segment
  - Data segment
  - ...
- OS houdt tabel bij (descriptortabel):
  - Basisadres van elk segment van programma



# Memory management: Segmentatie

- Nadeel:

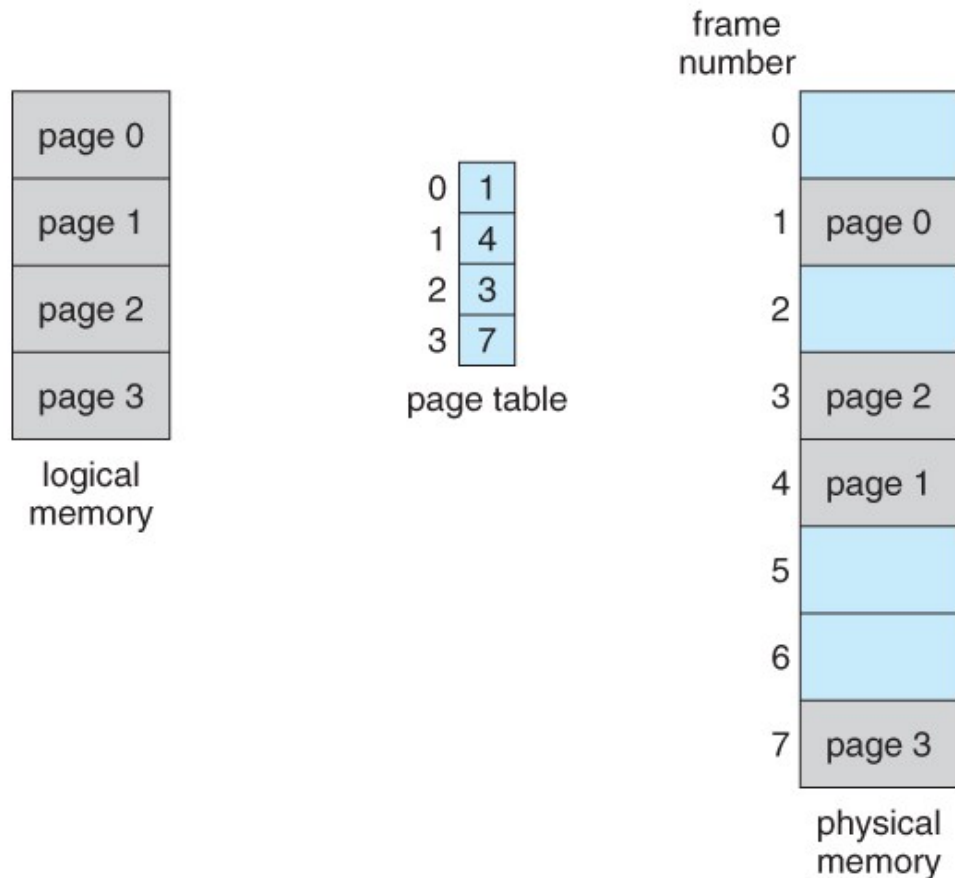
1. Fragmentatie 
2. Volledig segment in geheugen, enkel maar deeltje nodig voor uitvoering.



# Memory management: Paging

- Paging: oplossing voor nadeel 2
- Verdeel programma in gelijke "*pagina's*":
  - Pagina = Deel van programma met vaste grootte vb 4KB
- Verdeel geheugen in gelijke "*frames*":
  - Frame = Deel van geheugen met vaste grootte (= grootte pagina)

# Memory management: Paging

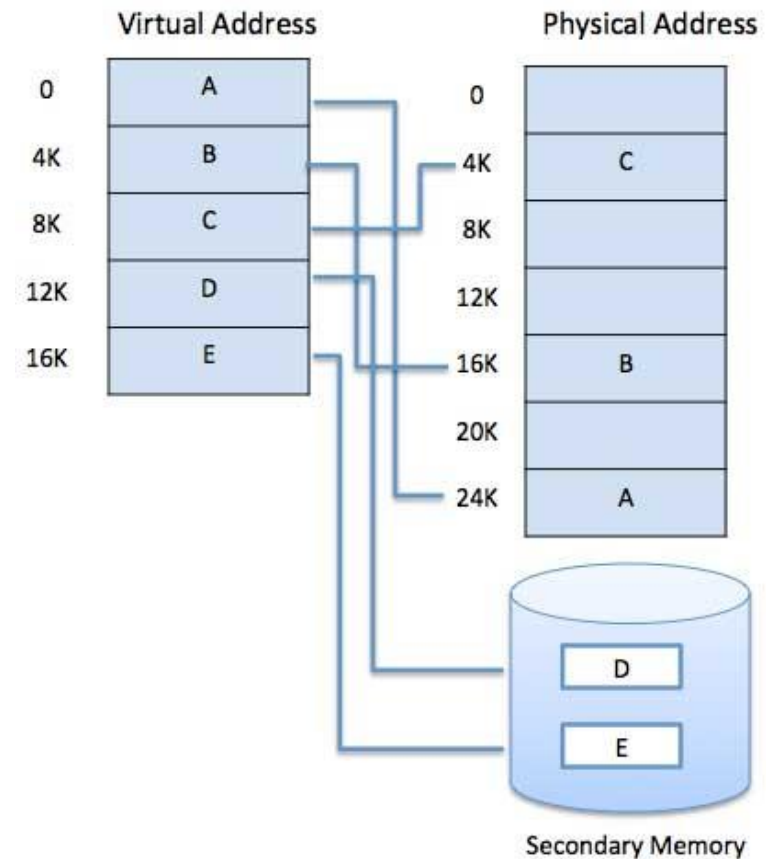


- Per proces een paginatablel:
  - Mapping pagina --> frame
- Verwijzing naar correcte paginatablel tijdens uitvoering:
  - Specifiek register
  - Veranderd steeds bij context switch

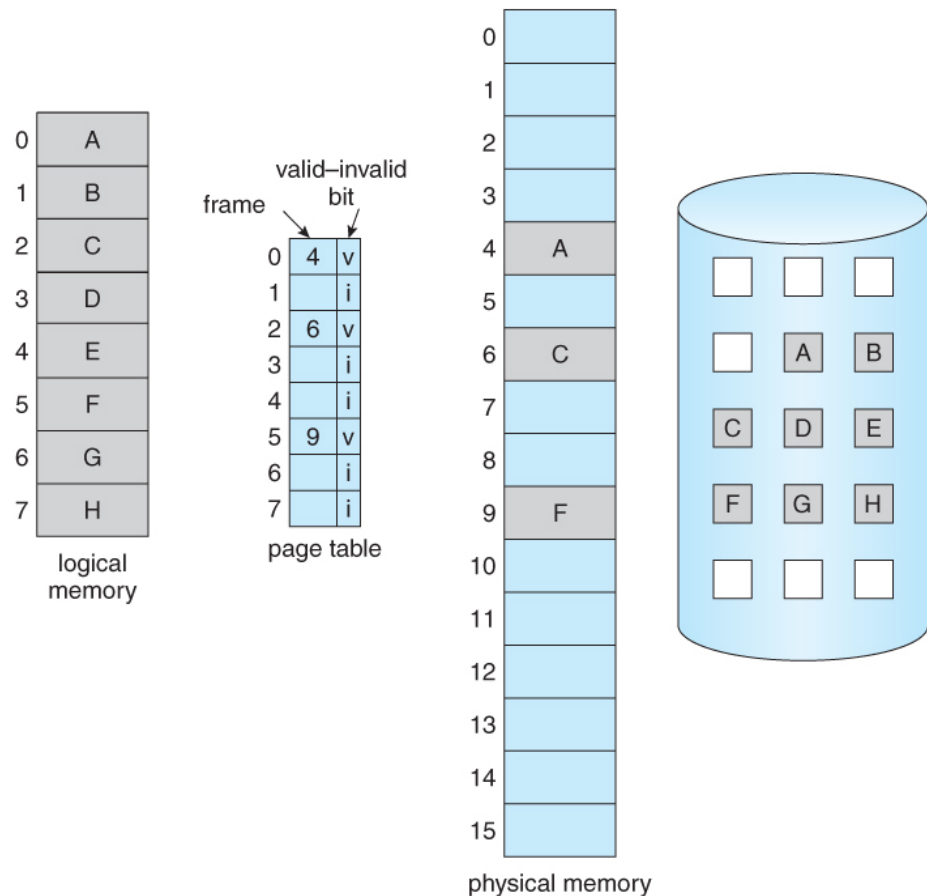


# Memory management: Virtual memory

- Deel van SSD/HDD dat aangeboden wordt als Memory
- OS krijgt totale ruimte (RAM + Virtual memory) ter zijner beschikking.
- MC doet vertaling van virtueel adres naar correct fysiek adres
  - Ofwel naar RAM adres
  - Ofwel naar SSD/HDD adres



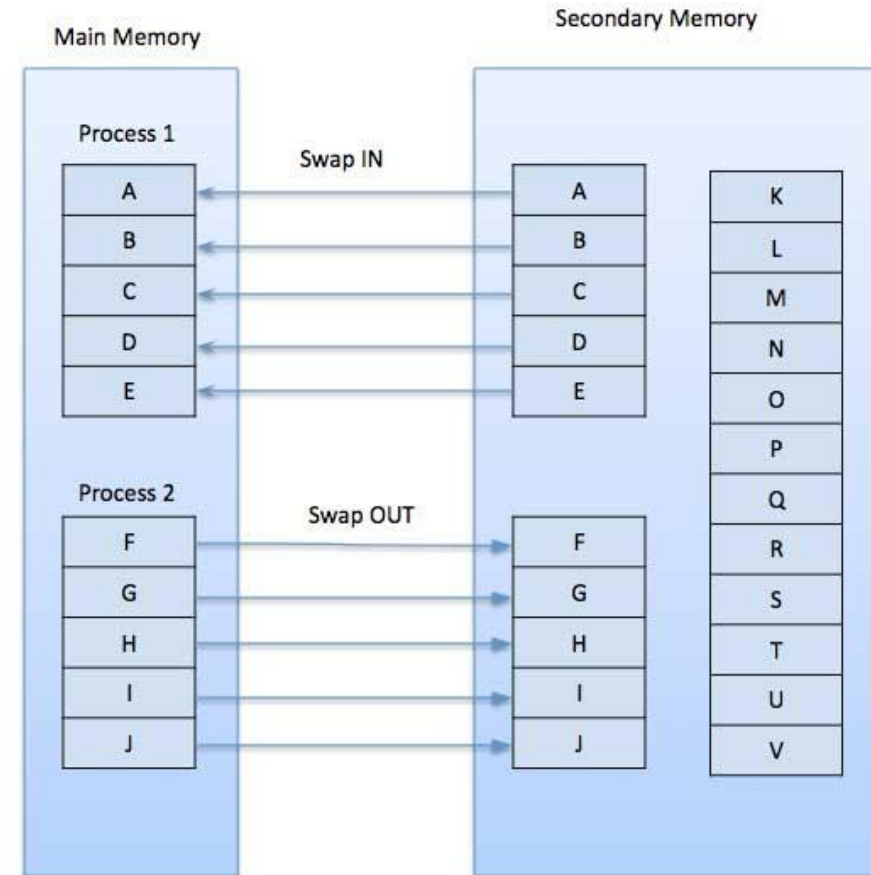
# Memory management: On demand paging



- Niet alle pages zijn nodig tijdens uitvoering
- Optimalisatie: Enkel pages inladen die effectief gebruikt worden.
- Page table krijgt extra bit
  - Aanwezig in geheugen = True or False

# Memory management: Swapping

- Nieuwe page inladen, maar geheugen is vol?  
==> Swapping
- Verwijder page uit geheugen:
  - **NIET** deleten  
*maar*
  - **Verplaats** naar specifiek deel van SDD/HDD (later terug nodig):
    - Swap file (Windows)
    - Swap partitie (Linux)



# Memory Management: Swapping algoritmes

- Welke page(s) moet even weg?
- Verschillende algoritmes:
  - FIFO (First in First out)
    - Oudste pagina eerst
  - Optimal Page Replacement
    - Vervang de pagina die in de toekomst het minste zal gebruikt worden.
  - LRU (Least Recently Used)
    - Pagina die het langste niet meer gebruikt is.

# Memory management: Swapping

- Swapping vertraagd computer
- OS zal swapping enkel gebruiken indien echt nodig.

