

Hoofdstuk 2

Werkgeheugen, centrale verwerkseenheid, programma's



H2: Werkgeheugen, centrale verwerkseenheid, programma's

- Werkgeheugen
- Centrale verwerkseenheid: welke bevelen?
- Programma's schrijven
- De uitvoering van programma's
- Bevelenwachtrij
- Rekeneenheid
- Klok

Werkgeheugen

Welkom bij ALDI
Hoge kwaliteit – Lage prijs

Overzicht **Processor** **Windows 8.1**

Nu verkrijgbaar

Multimedia pc-systeem
MEDION® AKOYA® E4015 E (MD 8321)

- AMD A10-7300 processor
- Windows 8.1 Update
- Geïntegreerde Radeon™ R7-graphics
- 4 GB Dual Channel werkgeheugen
- 1.000 GB harde schijf
- Wi-Fi®

Per stuk
399.00

LiveSafe



acer
XC-105 A4412
desktop computer

- klaar voor elke taak door de krachtige AMD A4 Quad-core processor en 8 GB intern geheugen
- kom geen opslagruimte te kort door de grote 1 TB harde schijf
- veelgebruikte aansluitingen zoals USB en hoofdtelefoon zitten aan de voorkant

398,-
incl. 6,05 thuiskopieheffing

Een
Kot
upgr.
Win

AMD A10
PROCESOR



INTEL PENTIUM DUAL CORE PROCESSOR E5400 (2.7GHZ, 2MB, L2 CACHE)
ALL-IN-ONE: ALLE COMPONENTEN IN HET SCHERM, DUS GEEN PC KAST

500GB HARDE SCHIJF

hp **b&w WiFi CERTIFIED** **Windows 7 Home Premium** **intel Pentium**

ALL IN ONE ALLROUND COMPUTER

HP Type 200-5120nl.

- 21,5" Brightview LCD breedbeeldscherm
- Videokaart: Intel Graphics Media Accelerator X4500HD
- Intern geheugen: 4GB
- Aansluitingen: o.a. 7XUSB 2.0
- Netwerk: o.a. 100/1000 lan
- Kaartlezer: 6-in-1
- Webcam: VCA met ingebouwde microfoon
- Inclusief: toetsenbord en optische muis
- 2 jaar garantie

WINST PAKKER!

~~699,-~~ (831,- incl.) **499,-** (593,- incl.)

Werkgeheugen

- Miljarden bytes (= 8 bits)
- Elke byte heeft een **adres**
 - Uniek nummer, van 00000000 tot FFFFFFFF op een 32-bit CPU
 - Altijd positief
- Maximale grootte van werkgeheugen wordt bepaald door de grootte van het adres
 - Wat is de maximale hoeveelheid werkgeheugen?

Werkgeheugen

- Soms is het handiger om bytes te groeperen
 - 1 byte = 256 verschillende getallen
 - 2 bytes = 65.536 verschillende getallen = **woord**
 - 4 bytes = 4.294.967.296 verschillende getallen = **dubbelwoord**
- Vb.: **int** in Java is een dubbelwoord
 - 4 bytes, dus 32 bits
 - -2.147.483.648 tot 2.147.483.647

Merk op: *het adres van een (dubbel)woord is het adres van de eerste byte van dat (dubbel)woord*

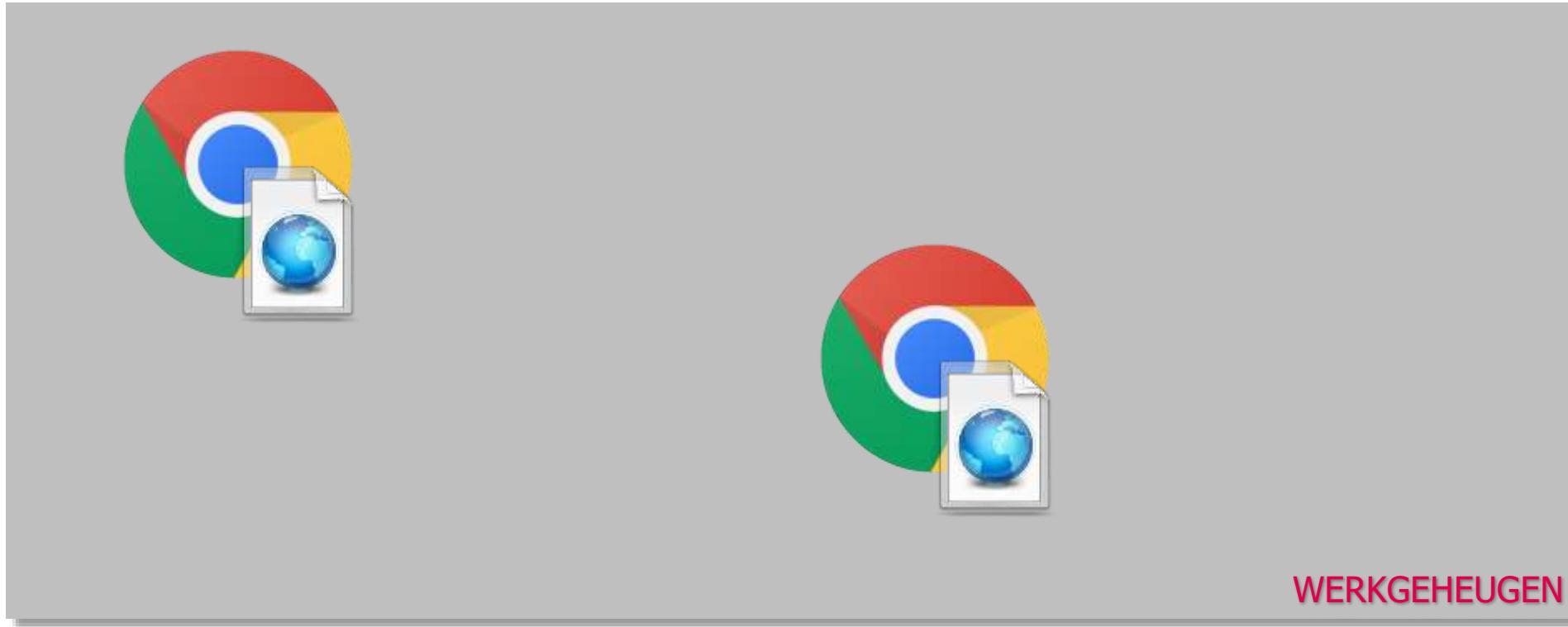
Absolute Adressering

0A4	00	2A	DD	C3
0A8	E1	00	12	34
0AC	B0	23	89	0A

*“In het dubbelwoord met adres **0A8** staat de waarde **(E1001234)_h**”*

Terminologie: *het adres van een byte/(dubbel)woord noemen we ook het **absolute adres***

Relatieve Adressering



Basis en Verplaatsing

- 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 :
 $\text{adres} = \text{basis} + \text{verplaatsing}$

Basis en Verplaatsing


$$\text{adres}_{\text{html}1} = \text{basis}_{\text{browser}1} + 1234$$

$$\text{adres}_{\text{html}2} = \text{basis}_{\text{browser}2} + 1234$$

WERKGEHEUGEN

Basis en Verplaatsing

- Notatie: (basis:verplaatsing).
- Vb.: byte met absoluut adres **012A3C4D** kan aangeduid worden met :
 - (012A3C00:0000004D), of ook:
 - (01000000:002A3C4D), of ook:
 - (01234567:**?????????**) (oefening)

Opgelet: wanneer we "adres" zeggen,
bedoelen we meestal "verplaatsing"!

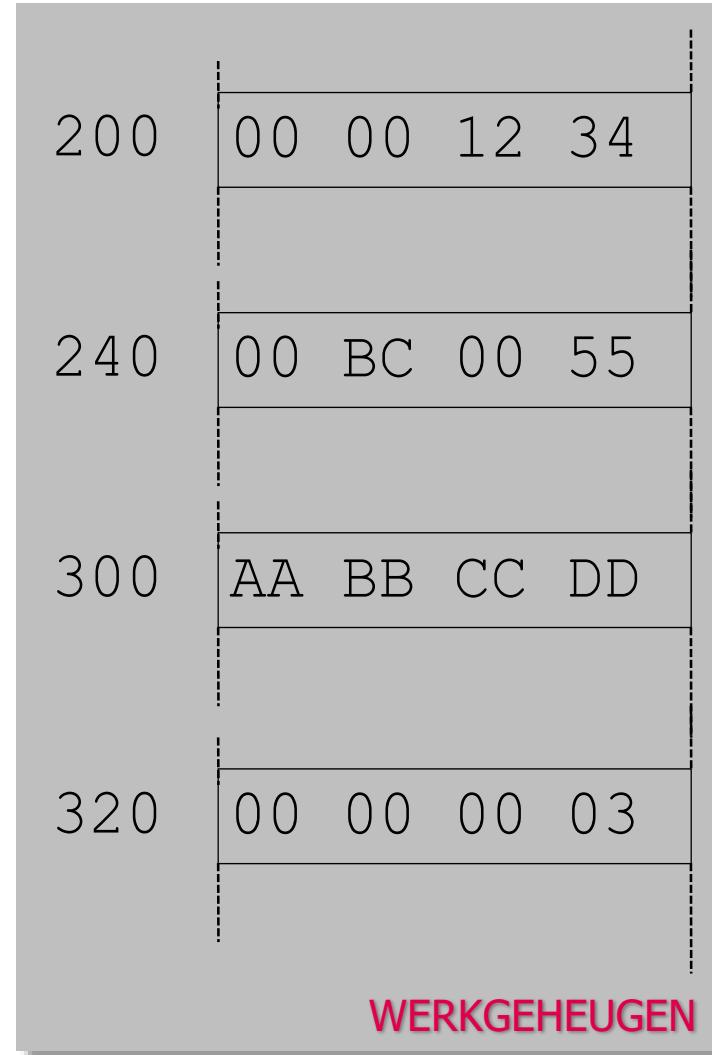


H2: Werkgeheugen, centrale verwerkseenheid, programma's

- Werkgeheugen
- Centrale verwerkseenheid: welke bevelen?
- Programma's schrijven
- De uitvoering van programma's
- Bevelenwachtrij
- Rekeneenheid
- Klok

Centrale Verwerkingsseenheid

- Kan **dingen doen...**
 - Bevelen die een voor een uitgevoerd worden
 - Staan achter elkaar in het geheugen, bvb. **A1 00 02 00 00 03 05 40 02 00 00 A3 00 03 00 00 2B 05 00 02 00 00 03 05 20 03 00 00**
- ... met **data**
 - In registers (EAX, EBX, ECX, EDX, ...) of het werkgeheugen
- Door de uitvoering van bevelen verandert de data in de registers en/of het werkgeheugen



Instructie Vb. 1

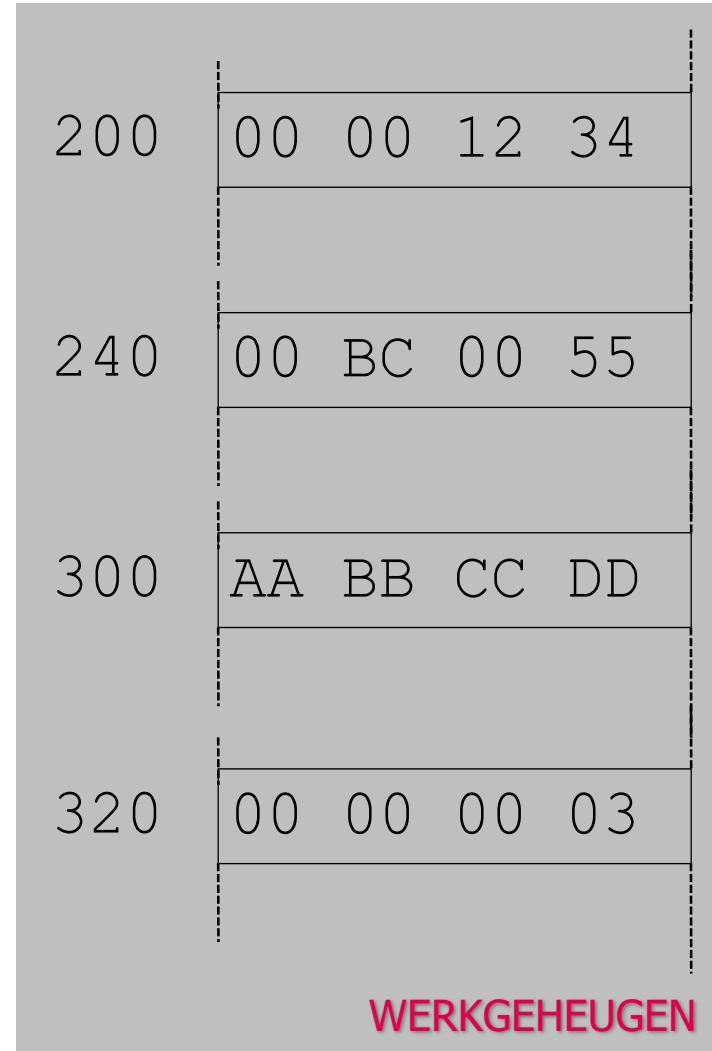
Bevel: A1 00 02 00 00

- kopieer
- de inhoud v.h. dubbelw.
met adres **00000200**
- naar EAX

Resultaat:

EAX:

00	00	12	34
----	----	----	----



Instructie Vb. 2

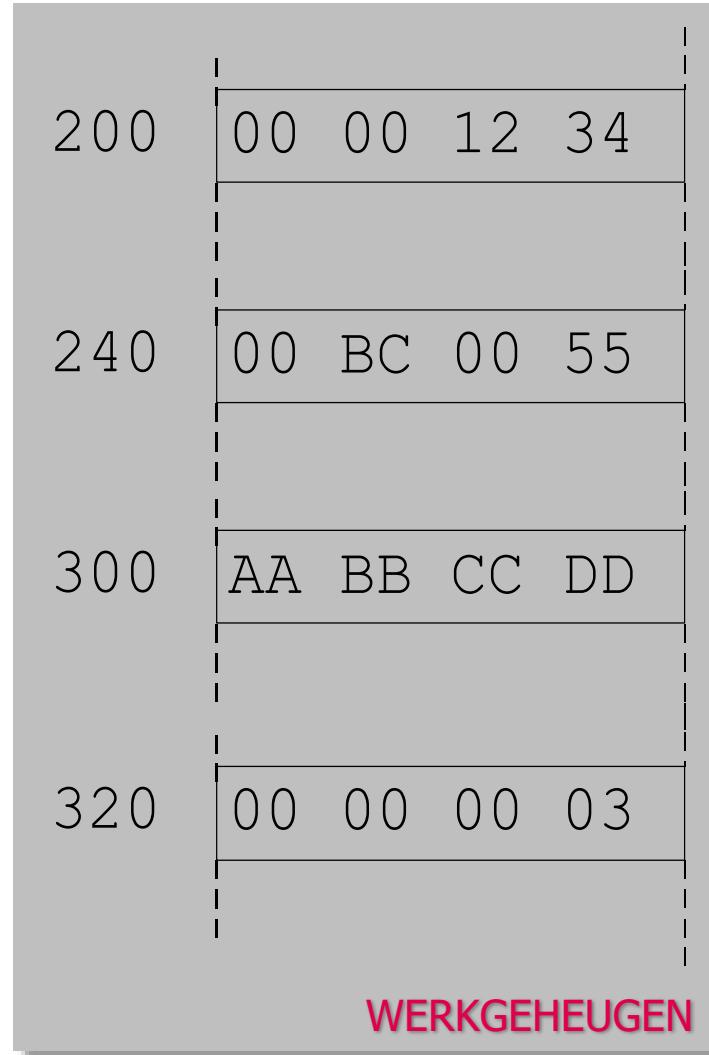
Bevel: 03 05 40 02 00 00

- tel
- de inhoud v.h. dubbelw. met adres **00000240** op
- bij de inhoud van EAX

Resultaat: **00001234**
+00BC0055
00BC1289

EAX :

00	BC	12	89
----	----	----	----



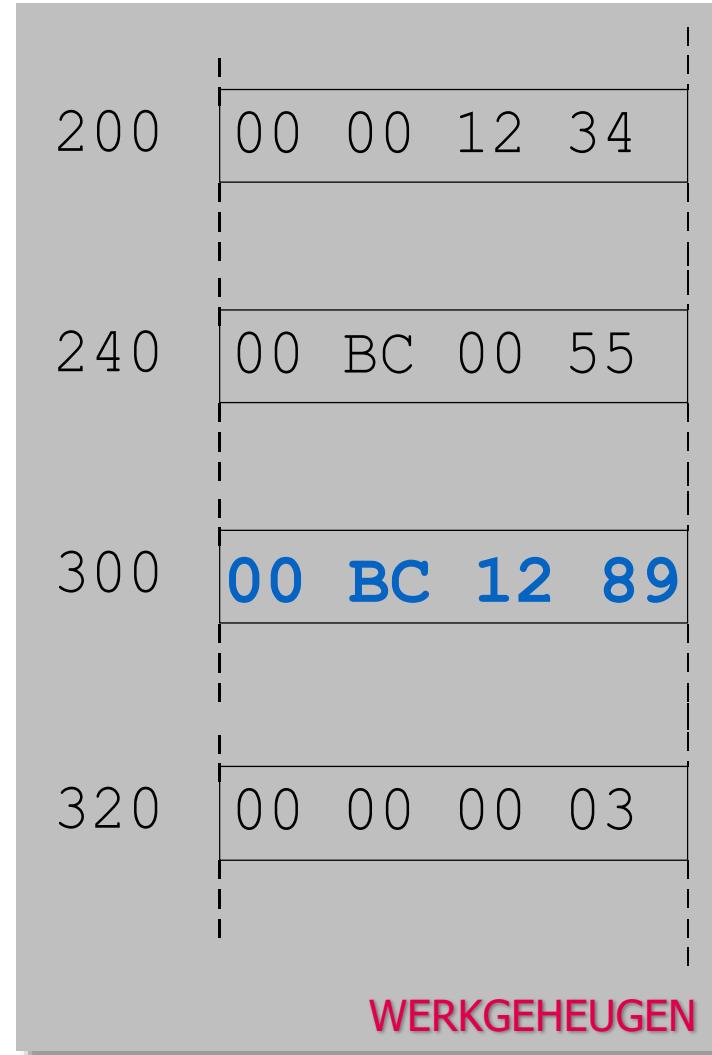
Instructie Vb. 3

Bevel: **A3 00 03 00 00**

- kopieer
- de inhoud van EAX
- naar het dubbelwoord met adres **00000300**

EAX:

00 BC 12 89



Instructie Vb. 3

Bevel: **A3 00 03 00 00**

- kopieer
- de inhoud van EAX
- naar het dubbelwoord met adres **00000300**

Resultaat

EAX:

00 BC 12 89

Een bevel bestaat uit twee delen

- **Wat** moet er gebeuren?
 - Kopieer naar EAX, tel op bij EAX, kopieer inhoud EAX naar, ...
 - Dit is de **functiecode**
- **Waarmee** moet het gebeuren?
 - De inhoud van een dubbelwoord in het werkgeheugen of een register
 - Dit is de (optionele) **operand**

C.V.E. begrijpt :

A1 00 02 00 00
03 05 40 02 00 00
A3 00 03 00 00
2B 05 00 02 00 00
03 05 20 03 00 00

Moeten wij die bevelen zo schrijven? **Neen!**

Bevelen

A1|00 02 00 00

03 05|40 02 00 00

A3|00 03 00 00

2B 05|00 02 00 00

03 05|20 03 00 00

Adres =

00000200

00000240

00000300

00000200

00000320

- In een bevel staan de bytes van het adres omgekeerd.
 - Vervelend; maar zo wil de c.v.e. het...
- Moeten wij die adressen zo schrijven? **Neen!**

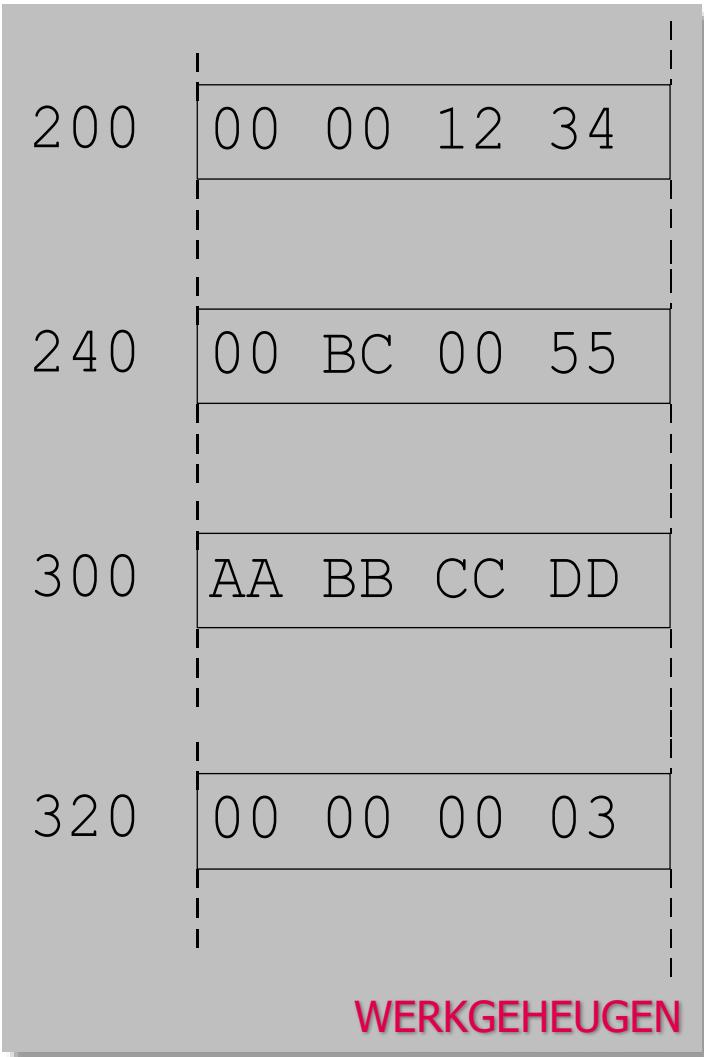
Mnemotechnische functiecodes

- Bvb.: i.p.v. A1: **mov eax**

- **mov eax, [200h]**
- **add eax, [240h]**
- **mov [300h], eax**
- **sub eax, [200h]**
- **add eax, [320h]**

1. h = hexadecimal
2. eax = EAX
3. Adres tussen [en]
4. Linkse parameter krijgt resultaat

Vb.: mov eax, [200h]

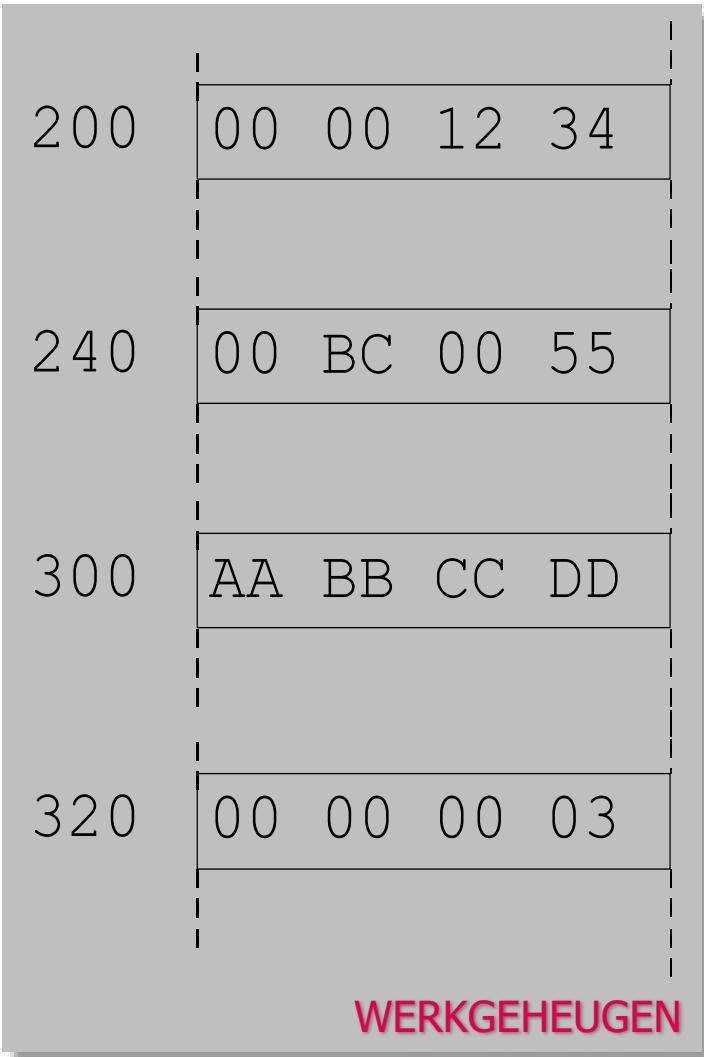


Resultaat:

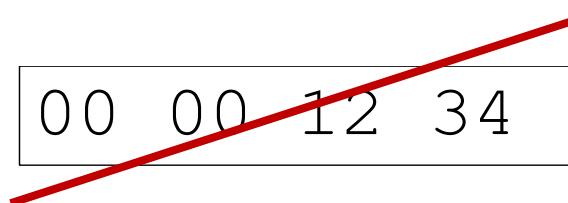
EAX:

00	00	12	34
----	----	----	----

Vb.: add eax, [240h]



EAX:



Resultaat:

EAX:



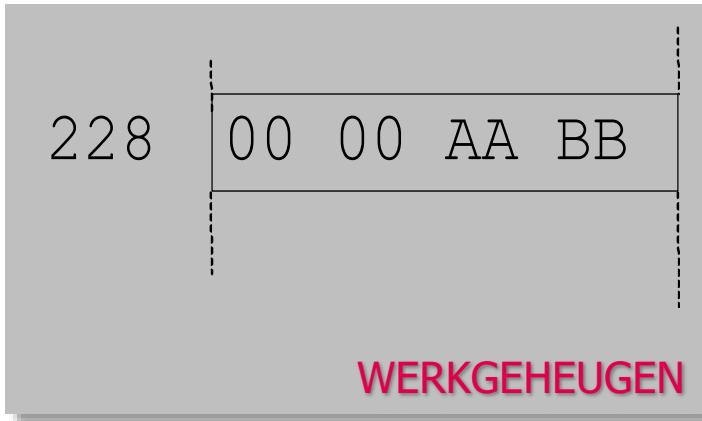
Vb.: inv [500h]



Effect:

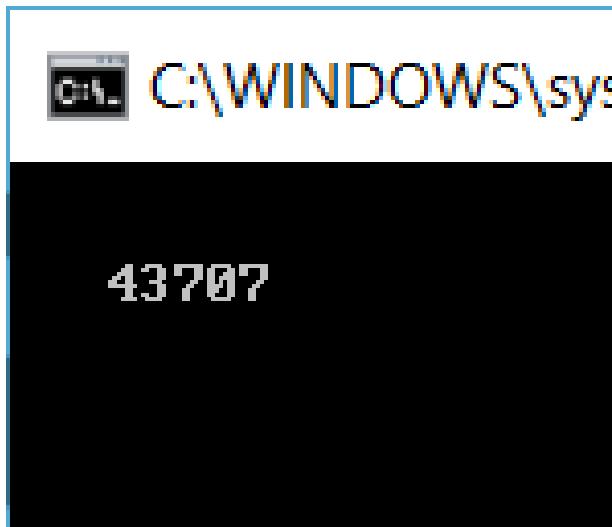
1. computer wacht
2. gebruiker typt getal in, bvb. 748 (+ Enter)
3. computer rekent getal om naar binair: **00 00 02 EC**
4. computer plaatst getal in het dubbelwoord met adres **500h**

Vb.: uit [228h]



Effect:

1. computer rekent getal om naar decimaal: 43707
2. computer toont dit getal op het scherm



- **inv, uit**: GEEN echte machine-bevelen, maar UCLL-bevelen.
- De programmeur **schrijft** bvb. mov eax,[100h];
- De CPU **wil** bvb. A1 00010000
- Programma zoals wij het schrijven, moet vertaald worden
 - mov eax,... → A1 ...

Assembler vs. Compiler

- Programma in machinetaal wordt **geassembleerd** door het assembleerprogramma (E.: assembler)
 - 1 bevel voor de programmeur is 1 bevel na vertaling
- Programma in hogere programmeertaal wordt **gecompileerd** door een compilator (E.: compiler)
 - 1 bevel voor de programmeur is 1, 2, ... 100 bevelen na vertaling

Programmavoorbeeld

- **Schrijf een programma dat:**
 - aan de gebruiker een getal vraagt
 - het dubbel van dit getal toont

- **Bevelen:**
 - mov, add, sub, inv, uit

Programmavoorbeeld

```
%include "gt.asm"  
  
covar  
  
inleiding  
  
inv [100h]  
  
mov eax,[100h]  
  
add eax,[100h]  
  
mov [104h],eax  
  
uit [104h]  
  
slot
```

inleidende bevelen
zijn vereist

afsluitend bevel

De kunst van Assembly

- Je weet precies wat je van de computer wilt
- Je weet precies welke bevelen je mag geven:
 - wat het effect is (semantiek)
 - hoe je deze bevelen moet geven (syntax)
- Je werkt gestructureerd



H2: Werkgeheugen, centrale verwerkseenheid, programma's

- Werkgeheugen
- Centrale verwerkseenheid: welke bevelen?
- **Programma's schrijven**
- De uitvoering van programma's
- Bevelenwachtrij
- Rekeneenheid
- Klok

Voorbeeld: bereken $2x+4y+8z$

Bevelen:

- mov
- add
- sub
- inv
- uit

```
%include "gt.asm"
covar
inleiding
inv [100h] ;x op adres 100
inv [200h] ;y op adres 200
inv [300h] ;z op adres 300
mov eax,[300h] ;z
add eax,[300h] ;2xz
add eax,[200h] ;2xz+y
mov [104h],eax
add eax,[104h] ;2xz+y + 2xz+y
```

Voorbeeld: bereken $2x+4y+8z$

Bevelen:

- mov
- add
- sub
- inv
- uit

```
add eax,[100h] ;4z+2y+x
mov [104h],eax
add eax,[104h] ;4z+2y+x +
                 4z+2y+x
mov [104h],eax
uit [104h]
slot
```

```
%include "gt.asm"
covar
inleiding
inv [100h]      ;x op adres 100
inv [200h]      ;y op adres 200
inv [300h]      ;z op adres 300
mov eax,[300h]  ;z
add eax,[300h]  ;2xz
add eax,[200h]  ;2xz+y
mov [104h],eax
add eax,[104h]  ;2xz+y + 2xz+y
```

- Zelf adressen kiezen? **Neen!**
 - Op de plaats van het adres een naam schrijven = **symbolisch adres**
 - De computer vervangt die namen dan door echte adressen

Eis: *alle symbolische adressen
moeten gedefinieerd zijn!*

Voorbeeld: bereken $2x+4y+8z$

inv [x]

inv [y]

inv [z]

mov eax,[z] ;z

add eax,[z] ;2xz

add eax,[y] ;2xz+y

mov [hulpd],eax

add eax,[hulpd] ;2xz+y + 2xz+y

...

Voorbeeld: bereken $2x+4y+8z$

```
%include "gt.asm"
covar
x: resd 1
y: resd 1
z: resd 1
hulpd: resd 1
inleiding
inv [x]
...
```

- **resd 1 = reserveer 1 dubbelwoord**
- **(resw 1 = ... woord)**
- **(resb 1 = ... byte)**

- mnemotechnische codes vervangen :
 - `mov eax,... → A1`
- symbolische adressen vervangen :
 - `x → 000`
 - `y → 004 (waarom 4 meer?)`
 - `z → 008`
 - `hulpd → 00C`

Opgelet: *tijdens oefeningen alleen
symbolische adressen gebruiken!*

Constanten

- Vb.: *schrijf een programma dat een getal, x , vraagt en de waarde van $x+1$ toont*

%include "gt.asm"	inv [x]
covar	mov eax,[x]
x: resd 1	add eax,[een]
een: dd 1	mov [hulpd],eax
hulpd: resd 1	uit [hulpd]
inleiding	slot

Constanten

- een: dd 1
 - vertaalprogramma kent het symbolisch adres *een*
 - *een* is het adres van een dubbelwoord
 - de inhoud van dit dubbelwoord is 1

een
00 00 00 01

Constanten en Veranderlijken

- **Veranderlijke:** plaats in het geheugen voor een tussenresultaat
 - bvb. dubbelwoord met adres x, hulpd, ... ;
 - geen zinvolle inhoud na de vertaling, vandaar:
hulpd: resd 1
- **Constante:** plaats in het geheugen met een voorgedefinieerde waarde
 - tijdens de vertaling komt er al iets in: iets dat we in ons programma nodig hebben, bvb. : 1
een: dd 1



https://compsys.gt.khleuven.be/Exercise/Solution/1067

Bereken de som

Computersystemen Oefeningen Ranking Theorie Mijn studenten Admin Ingelogd als u0055108 [Uitloggen]

Hoofdstuk 1

Hoofdstuk 2

- Instructies
- Eerste programma's
- Symbolische adressen
- Sprongbevelen
- Sprongbevelen (werking)
- Extra oefeningen
- Theorie

Hoofdstuk 3

Hoofdstuk 4

Hoofdstuk 5

Bereken de som

Jouw antwoord voor deze vraag is correct!

Schrijf een programma dat aan de gebruiker 4 getallen vraagt en de som ervan toont op het scherm.

Broncode

```
1 %include "gt.asm"
2 covar
3 hulpd:      resd 1
4 * inleiding
5     inv [hulpd]
6     mov eax, [hulpd]
7     inv [hulpd]
8     add eax, [hulpd]
9     inv [hulpd]
10    add eax, [hulpd]
11    inv [hulpd]
12    add [hulpd], eax
13    uit [hulpd]
14 slot
```

Console

Voer uit

Vermenigvuldigen

$$\begin{array}{r} 82 \\ \times 45 \\ \hline 3690 \end{array}$$

32-bit
× 32-bit
64-bit

Voorbeeld:

...

getal1: dd 7

getal2: dd 5

...

mov eax,[getal1]

imul dword [getal2]

...

Past niet in een register



Effect

Na vertaling :

getal1: dd 7

getal2: dd 5

getal1

00 00 00 07

getal2

00 00 00 05

Na uitvoering van:

mov eax, [getal1]

imul dword [getal2]

EAX :

00 00 00 07

EDX :

00 00 00 00

EAX :

00 00 00 23

imul dword [adres]

- **adres:** adres van een dubbelwoord
- inhoud van **EAX** wordt vermenigvuldigd met de inhoud van het dubbelwoord
 - Alleen de inhoud van EAX kan vermenigvuldigd worden
- het product (resultaat) komt in het REGISTERPAAR (**EDX,EAX**)

Opmerking: *er zijn varianten
maar die doen we niet!*

imul dword [adres]

- Resultaat is altijd **64 bits** (=16 hex. Cijfers of 8 bytes)
 - De eerste helft hebben we meestal niet nodig; toch is hij er.
- Dword-aanduiding geeft aan dat [adres] een 32-bit getal bevat
 - Lang geleden: $(8 \text{ bits}) * (8 \text{ bits}) \rightarrow \text{imul byte } [\text{adres}]$
 - Minder lang geleden: $(16 \text{ bits}) * (16 \text{ bits}) \rightarrow \text{imul word } [\text{adres}]$
 - Kan nog. Daarom: **dword**



Nog een voorbeeld

neg: dd -3

pos: dd 5

neg

FF	FF	FF	FD
----	----	----	----

pos

00	00	00	05
----	----	----	----

mov eax,[neg]

imul dword [pos]

EAX :

FF	FF	FF	FD
----	----	----	----

EDX :

FF	FF	FF	FF
----	----	----	----

EAX :

FF	FF	FF	F1
----	----	----	----

Vermenigvuldiging

- Eerst één van de factoren in EAX
- Dan: **imul dword [adres]**
(adres van de 2^o factor)
- $$\begin{array}{r} \text{EAX} \\ \times \text{dub.w.} \\ \hline (\text{EDX}, \text{EAX}) \end{array}$$

Merk op: als het resultaat ligt tussen -2,1...miljard en +2,14... miljard, dan kunnen we verder werken met de inhoud van EAX.

- Waarom zeggen we: 10 gedeeld door 2 is 5? **Omdat $5 \times 2 = 10$.**

$$\begin{array}{r} \text{EAX} \\ \times \text{dub.w.} \\ \hline (\text{EDX}, \text{EAX}) \end{array}$$

$$\begin{array}{r} (\text{EDX}, \text{EAX}) \mid \underline{\text{dub.w.}} \\ \hline \mid \text{EAX} \\ \hline \text{EDX} \end{array}$$

Deling

nul: dd 0

deeltal: dd 85

deler: dd 3

nul

00 00 00 00

deeltal

00 00 00 55

deler

00 00 00 03

mov edx, [nul]

mov eax, [deeltal]

idiv dword [deler]

EDX :

00 00 00 00

EAX :

00 00 00 55

EDX :

00 00 00 01

EAX :

00 00 00 1C

idiv dword [adres]

- inhoud van REGISTERPAAR (**EDX,EAX**) wordt als één getal, gedeeld
- **adres:** adres van de deler (een dubbelwoord)
- het quotiënt komt in **EAX**
- de rest komt in **EDX**

Opgelet: *wanneer het resultaat niet in EAX geraakt, crasht het programma!*

Deling Negatieve getallen

nul: dd 0

deeltal: dd -20

deler: dd -5

mov edx, [deeler]

mov eax, [deeltal]

idiv dword ptr [deler]

nul

00 00 00 00

deeltal

FF FF FF EC

deler

00 00 00 03

EAX :

00 00 00 00	FF FF FF EC
-------------	-------------

EDX :

00 00 02 55	55 55 55 4E
-------------	-------------

Merk op: 5555554E

is een positief getal!

Deling Negatief Getal

- In (EDX,EAX) moet de correcte voorstelling van het deeltal staan. **Welk?**
 - 00 00 00 00 als het deeltal niet negatief is
 - FF FF FF FF als het deeltal wel negatief is
- Hoe doen we dit? **Vermenigvuldig met 1.**

Deling Negatief Getal

een: dd 1

deeltal: dd -20

deler: dd 3

een

00	00	00	01
----	----	----	----

deeltal

FF	FF	FF	EC
----	----	----	----

deler

00	00	00	03
----	----	----	----

`mov eax, [deeltal]`

`imul dword [een]`

`idiv dword [deler]`



EDX :

FF	FF	FF	FF
----	----	----	----

EAX :

FF	FF	FF	EC
----	----	----	----

EDX :

FF	FF	FF	FE
----	----	----	----

EAX :

FF	FF	FF	FA
----	----	----	----

- Deling van positief of negatief getal
 - deeltal in **EAX**
 - **imul dword [adres_getal1]**
 - **idiv dword [adres_deler]**
- Ben je zeker dat het deeltal ≥ 0 is?
 - 0 naar EDX is ook goed

Sprongbevelen

- Bevelen tot nog toe: add, sub, imul, idiv, ...
- Kracht van een computer: **herhaling**
 - een reeks bevelen meerdere malen uitvoeren zolang een bepaalde voorwaarde geldt...

Onvoorwaardelijke Sprong

```
inv [x]  
haha: mov eax,[x]  
       add eax,[een]  
       ...  
       mov [hulpd],eax  
jmp haha
```

- **haha** is een **symbolisch adres** van een bevel
 - Een bevel is een aantal opeenvolgende bytes in het geheugen, bvb.: A1 64 00 00 00
 - Het adres van een bevel is het adres van de eerste byte van dat bevel

Voorwaardelijke Sprong

In Java:

```
if (x > y) {  
    ... bevelen indien x>y;  
} else {  
    ... bevelen indien x≤y;  
}
```

In Assembly:

- eerst vergelijken, bvb. is $x > y$?
- dan: spring als "het" groter is
- geen else-stuk

Wat gebeurt er bij het vergelijken?
Instellen van de vlaggen.

- Een **vlag** is een bit in de CPU
 - nul-vlag (ZF), teken-vlag (SF), de overloop-vlag (OF), ...
 - Stelt toestand na een bepaalde instructies voor
- **add, sub:** resultaat wordt vergeleken met 0
 - ($\text{resultaat} = 0$) $\Leftrightarrow \text{ZF}=1$
 - ($\text{resultaat} < 0$) $\Leftrightarrow \text{SF}=1$
 - (overloop) $\Leftrightarrow \text{OF}=1$

- **add, sub:** resultaat wordt vergeleken met 0
 - (resultaat = 0) \Leftrightarrow ZF=1
 - (resultaat < 0) \Leftrightarrow SF=1
 - (overloop) \Leftrightarrow OF=1

beta: dd -5

gamma: dd 2147483647

...

ZF = 0, SF = 0, OF = 0

sub eax, eax

add eax, [beta]

sub eax, [gamma]

- Twee getallen met elkaar vergelijken? **cmp**

```
getal1: resd 1
```

```
getal2: resd 1
```

```
...
```

```
mov eax, [getal1]
```

```
cmp eax, [getal2]
```

- Achter de schermen wordt getal2 van getal1 afgetrokken. **Het resultaat bepaalt de vlaggen.**

- Achter de schermen wordt getal2 van getal1 afgetrokken. **Het resultaat bepaalt de vlaggen.**
 - 1° getal = 2° getal: ZF=1
 - 1° getal < 2° getal: ZF=0 en SF≠OF
 - 1° getal > 2° getal: ZF=0 en SF=OF
- Moeten wij zit zo programmeren? **Neen!**

Voorwaardelijke Sprong

- Nieuwe instructies, bvb. **Jg** (jump if greater)
 - = spring indien 1° getal > 2° getal
 - = spring indien ($ZF = 0$ en $SF=OF$)
- Als wat groter is dan wat? **Het 1° t.o.v. het 2° .**

Voorwaardelijke Sprong

inv [a]

inv [b]

mov eax, [a]

cmp eax, [b] ;cmp = compare

jl verder ;jl = jump indien less

...

verder: ...

less = 1° getal < 2° getal (dus [a] < [b])

Varianten

- je ... jne (equal ... not equal)
- jl ... jnl (less ... not less)
- jg ... jng (greater ... not greater)
- jnl = jge (greater or equal)
- jng = jle (less or equal)
- jo ... jno (overflow ... not overflow)

Voorwaardelijke Sprong

- Eerst **vlaggen** instellen
 - add, sub, cmp
- Dan **voorwaardelijke sprong** naar **symbolisch adres**
 - jg, jng, jl, jnl, je, jne



Oefening

- Vb. : Een programma dat de getallen 100, 99, 98, ..., 0 toont.

- Schema :

getal = 100;

repeat

{toon getal;

 getal = getal - 1;}

until (getal < 0)

- Keuze : getal in EAX,...

honderd: dd 100

hulpd: resd 1

een: dd 1

mov eax,[honderd]

mov [hulpd],eax

uit [hulpd]

sub eax,[een]



Oefening

```
%include "gt.asm"
```

```
covar
```

```
honderd: dd 100
```

```
hulpd:    resd 1
```

```
een:      dd 1
```

```
inleiding
```

```
        mov eax, [honderd]
```

```
hoger:   mov [hulpd],eax
```

```
        uit [hulpd]
```

```
        sub eax, [een]
```

```
        jnl hoger
```

```
slot
```

Wanneer springt deze
instructie en wanneer niet?



H2: Werkgeheugen, centrale verwerkseenheid, programma's

- Werkgeheugen
- Centrale verwerkseenheid: welke bevelen?
- Programma's schrijven
- De uitvoering van programma's
- Bevelenwachtrij
- Rekeneenheid
- Klok

Programma's Uitvoeren

- Een programma staat in het werkgeheugen
 - Waar?
 - In welke gedaante??
 - Hoe komt het in het werkgeheugen???
 - Hoe gebeurt de uitvoering????

Programma's Uitvoeren

Vb.:

...	inleiding
alfa: dd 17	mov eax, [alfa]
beta: dd -5	imul dword [mu]
gamma: dd 123	add eax, [beta]
mu: dd 7	imul dword [mu]
	add eax, [gamma]
...	
	slot

In welke gedaante?

A1 9C 01 00 00	mov eax, [alfa]
F7 2D A8 01 00 00	imul dword [mu]
03 05 A0 01 00 00	add eax, [beta]
F7 2D A8 01 00 00	imul dword [mu]
03 05 A4 01 00 00	add eax, [gamma]

Deze omzetting wordt gedaan door het
vertaalprogramma.

In welke gedaante?

```
...
alfa: dd 17
beta: dd -5
gamma: dd 123
mu: dd 7
inleiding
mov eax, [alfa]
imul dword [mu]
add eax, [beta]
imul dword [mu]
add eax, [gamma]
...
slot
```

IDE

```
...
61 6C 66 61 3A 20 64 64 20 31 37
62 65 74 61 3A 20 64 64 20 2D 35
67 61 6D 6D 61 3A 20 64 64 20 31 32 33
6D 75 3A 20 64 64 20 37
69 6E 6C 65 69 64 69 6E 67
6D 6F 76 20 65 61 78 2C 20 5B 61 6C 66 61 5D
69 6D 75 6C 20 64 77 6F 72 64 20 5B 6D 75 5D
61 64 64 20 65 61 78 2C 20 5B 62 65 74 61 5D
69 6D 75 6C 20 64 77 6F 75 64 20 5B 6D 75 5D
61 64 64 20 65 61 78 2C 20 5B 67 61 6D 6D 61 5D
...
73 6C 6F 74
```

ASCII

```
...
00 00 00 11
FF FF FF FB
00 00 00 7B
00 00 00 07
...
A1 9C 01 00 00
F7 2D A8 01 00 00
03 05 A0 01 00 00
F7 2D A8 01 00 00
03 05 A4 01 00 00
...

```

EXE

De Assembler

- Vervangt symbolische adressen
 - [alfa] → 00 00 01 9C
 - [beta] → 00 00 01 A0
 - [gamma] → 00 00 01 A4
 - [mu] → 00 00 01 A8
- Vertaalt (ASCII-voorstelling van) bevelen
 - 6D 6F 76 20 65 61 78 2C 5B 61 6C 66 61 5D 0D 0A
(d.i. mov eax,[alfa]) wordt A1 9C 01 00 00
- Vertaalt constanten en veranderlijken
 - 62 65 74 61 3A 20 64 64 20 2D 35
(d.i. beta: dd -5) wordt FF FF FF FB
- Schrijft een object (.obj)-bestand weg

Programma naar Geheugen

- Programma (.exe) wordt gemaakt door de **linker**
 - Combinatie van verschillende object-bestanden
- Een programma wordt int het werkgeheugen geladen door het **besturingssysteem**
 - Constanten en veranderlijken op een locatie, programmacode op een andere locatie
 - Besturingssysteem kiest deze locaties



c.v.e.

EDX

EAX

EBX

ECX

Rekeneenheid

bewerkingen (+, -, *, /)
uitvoeren

Besturingseenheid

welk bevel is er aan
de beurt?

werkgeheugen

De Besturingseenheid

- Bevat twee registers
 - Het **bevelregister** bevat de bytes van het bevel dat uitgevoerd wordt
 - De **bevelenteller** (instruction pointer, EIP) bevat het adres van het volgende bevel

Terminologie: *eigenlijk bevat de bevelenteller de verplaatsing van het volgende bevel (t.o.v. de basis van het programma)*

Haal- en Uitvoercyclus

- **Haalcyclus**
 - Volgend bevel ophalen (waar?)
 - Beveleenteller aanpassen (lengte van opgehaald bevel erbij optellen)
- **Uitvoercyclus**
 - Het bevel analyseren (wat moet er gebeuren?)
 - Ervoor zorgen dat het gebeurt

Vraagje: *hoe lang is een bevel?*



BEVELENTELLER

00 00 00 00

BEVELREGISTER

A1 9C 01 00 00

- **Haal:** bevel met adres 00 00 00 00
- **Haal:** pas bevelenteller aan: +5

programma

| A1 | 9C | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03 |

| 05 | A0 | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03 |

| 05 | A4 | 01 | 00 | 00 | | | | | | | | | |



BEVELENTELLER

00 00 00 05

BEVELREGISTER

A1 9C 01 00 00

- **Uitvoering:** wat betekent A1 9C 01 00 00?
- **Uitvoering:** doe mov eax, [0000019C]

programma

| A1 | 9C | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03 |

| 05 | A0 | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03 |

| 05 | A4 | 01 | 00 | 00 | | | | | | | | | |



BEVELENTELLER

00 00 00 05

BEVELREGISTER

A1 9C 01 00 00

- Haal: bevel met adres 00 00 00 05

programma

| A1 | 9C | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03 |

| 05 | A0 | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03 |

| 05 | A4 | 01 | 00 | 00 | | | | | | | | | |



BEVELENTELLER

00 00 00 05

BEVELREGISTER

F7 2D A8 01 00 00

- **Haal:** bevel met adres 00 00 00 05
- **Haal:** pas bevelenteller aan: +6

programma

| A1 | 9C | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03 |

| 05 | A0 | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03 |

| 05 | A4 | 01 | 00 | 00 | | | | | | | | | |



BEVELENTELLER

00 00 00 0B

BEVELREGISTER

F7 2D A8 01 00 00

- **Haal:** bevel met adres 00 00 00 05
- **Haal:** pas bevelenteller aan: +6

programma

A1 | 9C | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03

05 | A0 | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03

05 | A4 | 01 | 00 | 00 | | | | | | | |



BEVELENTELLER

00 00 00 0B

BEVELREGISTER

F7 2D A8 01 00 00

- **Uitvoering:** wat betekent F7 2D A8 01 00 00?
- **Uitvoering:** doe `imul dword [000001A8]`

programma

| A1 | 9C | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03 |

| 05 | A0 | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03 |

| 05 | A4 | 01 | 00 | 00 | | | | | | | | | |



BEVELENTELLER

00 00 00 0B

BEVELREGISTER

F7 2D A8 01 00 00

- Haal: bevel met adres 00 00 00 0B

programma

A1 | 9C | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03

05 | A0 | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03

05 | A4 | 01 | 00 | 00 | | | | | | | |



BEVELENTELLER

00 00 00 0B

BEVELREGISTER

03 05 A0 01 00 00

- **Haal:** bevel met adres 00 00 00 0B
- **Haal:** pas bevelenteller aan: +6

programma

| A1 | 9C | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03 |

| 05 | A0 | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03 |

| 05 | A4 | 01 | 00 | 00 | | | | | | | | | |



BEVELENTELLER

00 00 00 11

BEVELREGISTER

03 05 A0 01 00 00

- **Haal:** bevel met adres 00 00 00 0B
- **Haal:** pas bevelenteller aan: +6

programma

| A1 | 9C | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03 |

| 05 | A0 | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03 |

| 05 | A4 | 01 | 00 | 00 | | | | | | | | | |



BEVELENTELLER

00 00 00 11

BEVELREGISTER

03 05 A0 01 00 00

- Uitvoering: wat betekent 03 05 A0 01 00 00?
- Welke zijn de volgende waarden in b.t. en b.r.?

programma

A1 | 9C | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03

05 | A0 | 01 | 00 | 00 | F7 | 2D | A8 | 01 | 00 | 00 | 03

05 | A4 | 01 | 00 | 00 | | | | | | | |

Sprongbevelen

```
...  
    mov  eax, [a1]  
    sub  eax, [a2]  
    cmp  edx, 4  
    je   aja  
    mov  eax, [a1]  
    mov  ebx, [a2]  
    mov  ecx, [a3]  
    mov  edx, [a4]  
aja: add  [a1], eax  
     add  [a2], ebx  
...  
IDE
```



...	...
01D	A1 9C 01 00 00
022	2B 05 A0 01 00 00
028	81 FA 04 00 00 00
02E	74 17
030	A1 9C 01 00 00
035	8B 1D A0 01 00 00
03B	8B 0D A4 01 00 00
041	8B 15 A8 01 00 00
047	01 05 9C 01 00 00
04D	01 1D A0 01 00 00
...	...

EXE



BEVELENTELLER

00 00 00 28

BEVELREGISTER

2B 05 A0 01 00 00

- Haal: bevel met adres 00 00 00 28

...
028	81 FA 04 00 00 00	cmp edx, 4
02E	74 17	je aja
030	A1 9C 01 00 00	mov eax, [a1]
...
047	01 05 9C 01 00 00	aja: add [a1], eax



BEVELENTELLER

00 00 00 28

BEVELREGISTER

81 FA 04 00 00 00

- **Haal:** bevel met adres 00 00 00 28
- **Haal:** pas bevelenteller aan: +6

...
028	81 FA 04 00 00 00	cmp edx, 4
02E	74 17	je aja
030	A1 9C 01 00 00	mov eax, [a1]
...
047	01 05 9C 01 00 00	aja: add [a1], eax



BEVELENTELLER

00 00 00 2E

BEVELREGISTER

81 FA 04 00 00 00

- **Uitvoering:** wat betekent 81 FA 04 00 00 00?
- **Uitvoering:** doe **cmp edx, 4**

Vlaggen!

...
028	81 FA 04 00 00 00	cmp edx, 4
02E	74 17	je aja
030	A1 9C 01 00 00	mov eax, [a1]
...
047	01 05 9C 01 00 00	aja: add [a1], eax



BEVELENTELLER

00 00 00 2E

BEVELREGISTER

81 FA 04 00 00 00

- Haal: bevel met adres 00 00 00 2E

...
028	81 FA 04 00 00 00	cmp edx, 4
02E	74 17	je aja
030	A1 9C 01 00 00	mov eax, [a1]
...
047	01 05 9C 01 00 00	aja: add [a1], eax



BEVELENTELLER

00 00 00 2E

BEVELREGISTER

74 17

- Haal: bevel met adres 00 00 00 2E
- Haal: pas bevelenteller aan: +2

...
028	81 FA 04 00 00 00	cmp edx, 4
02E	74 17	je aja
030	A1 9C 01 00 00	mov eax, [a1]
...
047	01 05 9C 01 00 00	aja: add [a1], eax



BEVELENTELLER

00 00 00 30

BEVELREGISTER

74 17

- **Haal:** bevel met adres 00 00 00 2E
- **Haal:** pas bevelenteller aan: +2

...
028	81 FA 04 00 00 00	cmp edx, 4
02E	74 17	je aja
030	A1 9C 01 00 00	mov eax, [a1]
...
047	01 05 9C 01 00 00	aja: add [a1], eax



BEVELENTELLER

00 00 00 30

BEVELREGISTER

74 17

- **Uitvoering:** wat betekent 74 17?
- **Uitvoering:** doe je aja
 - Twee mogelijkheden: wel of niet springen

...
028	81 FA 04 00 00 00	cmp edx, 4
02E	74 17	je aja
030	A1 9C 01 00 00	mov eax, [a1]
...
047	01 05 9C 01 00 00	aja: add [a1], eax



Geen sprong

BEVELENTELLER

00 00 00 30

BEVELREGISTER

74 17

- Haal: bevel met adres 00 00 00 30

...
028	81 FA 04 00 00 00	cmp edx, 4
02E	74 17	je aja
030	A1 9C 01 00 00	mov eax, [a1]
...
047	01 05 9C 01 00 00	aja: add [a1], eax



Geen sprong

BEVELENTELLER

00 00 00 30

BEVELREGISTER

A1 9C 01 00 00

- Haal: bevel met adres 00 00 00 30
- Haal: pas bevelenteller aan: +5

...
028	81 FA 04 00 00 00	cmp edx, 4
02E	74 17	je aja
030	A1 9C 01 00 00	mov eax, [a1]
...
047	01 05 9C 01 00 00	aja: add [a1], eax



Geen sprong

BEVELENTELLER

00 00 00 35

BEVELREGISTER

A1 9C 01 00 00

- **Uitvoering:** doe mov eax, [a1]
- **Haal:** bevel met adres 00 00 00 35
- ...

...
028	81 FA 04 00 00 00	cmp edx, 4
02E	74 17	je aja
030	A1 9C 01 00 00	mov eax, [a1]
...
047	01 05 9C 01 00 00	aja: add [a1], eax



Wel sprong

BEVELENTELLER

00 00 00 30

BEVELREGISTER

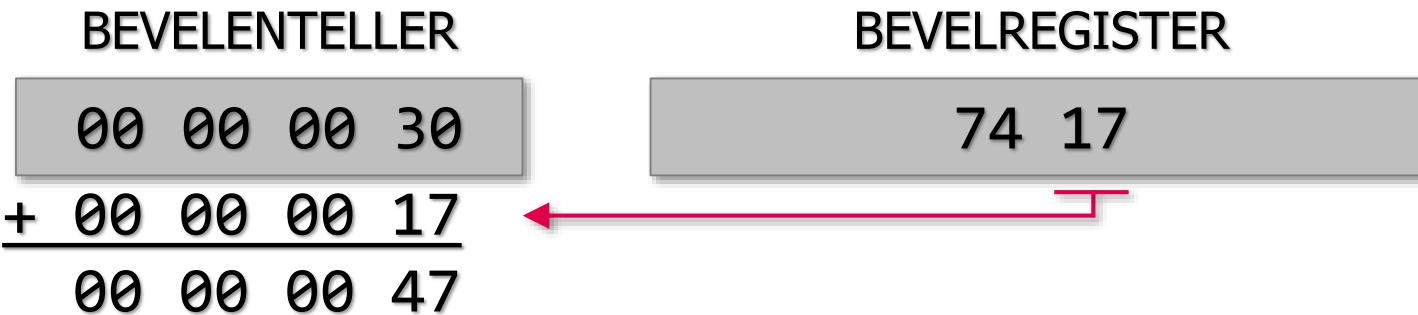
74 17

- **Uitvoering:** wat betekent 74 17?
- **Uitvoering:** doe je aja

...
028	81 FA 04 00 00 00	cmp edx, 4
02E	74 17	je aja
030	A1 9C 01 00 00	mov eax, [a1]
...
047	01 05 9C 01 00 00	aja: add [a1], eax



Wel sprong



Bevelenteller wordt aangepast!

...
028	81 FA 04 00 00 00	cmp edx, 4
02E	74 17	je aja
030	A1 9C 01 00 00	mov eax, [a1]
...
047	01 05 9C 01 00 00	aja: add [a1], eax



Wel sprong

BEVELENTELLER

00 00 00 47

BEVELREGISTER

74 17

- Haal: bevel met adres 00 00 00 47

...
028	81 FA 04 00 00 00	cmp edx, 4
02E	74 17	je aja
030	A1 9C 01 00 00	mov eax, [a1]
...
047	01 05 9C 01 00 00	aja: add [a1], eax



Wel sprong

BEVELENTELLER

00 00 00 47

BEVELREGISTER

01 05 9C 01 00 00

- **Haal:** bevel met adres 00 00 00 47
- **Haal:** pas bevelenteller aan: +6

...
028	81 FA 04 00 00 00	cmp edx, 4
02E	74 17	je aja
030	A1 9C 01 00 00	mov eax, [a1]
...
047	01 05 9C 01 00 00	aja: add [a1], eax



Wel sprong

BEVELENTELLER

00 00 00 4D

BEVELREGISTER

01 05 9C 01 00 00

- **Haal:** bevel met adres 00 00 00 47
- **Haal:** pas bevelenteller aan: +6

...
028	81 FA 04 00 00 00	cmp edx, 4
02E	74 17	je aja
030	A1 9C 01 00 00	mov eax, [a1]
...
047	01 05 9C 01 00 00	aja: add [a1], eax



Wel sprong

BEVELENTELLER

00 00 00 4D

BEVELREGISTER

01 05 9C 01 00 00

- **Uitvoering:** doe add [a1], eax
- ...

...
028	81 FA 04 00 00 00	cmp edx, 4
02E	74 17	je aja
030	A1 9C 01 00 00	mov eax, [a1]
...
047	01 05 9C 01 00 00	aja: add [a1], eax

Oefening

022	A1 9C 01 00 00	haha: <code>mov eax, [a1]</code>
...
055	2B 05 A0 01 00 00	<code>sub eax, [a2]</code>
05B	75 C5	<code>jnz haha</code>
05D	F7 2D A0 01 00 00	<code>imul dword [a2]</code>

BEVELENTELLER

00 00 00 5B

BEVELREGISTER

2B 05 A0 01 00 00

Hoe wordt de bevelenteller aangepast als de sprong uitgevoerd wordt?

Oefening

022	A1 9C 01 00 00	haha: mov eax, [a1]
...
055	2B 05 A0 01 00 00	sub eax, [a2]
05B	75 C5	jnz haha
05D	F7 2D A0 01 00 00	imul dword [a2]

BEVELENTELLER

00 00 00 5B

00 00 00 5D

BEVELREGISTER

2B 05 A0 01 00 00

75 C5

Oefening

022	A1 9C 01 00 00	haha: <code>mov eax, [a1]</code>
...
055	2B 05 A0 01 00 00	<code>sub eax, [a2]</code>
05B	75 C5	<code>jnz haha</code>
05D	F7 2D A0 01 00 00	<code>imul dword [a2]</code>

BEVELENTELLER

00	00	00	5D
+	??	??	??
<hr/>			

BEVELREGISTER

75	C5
----	----



Oefening

022	A1 9C 01 00 00	haha: <code>mov eax, [a1]</code>
...
055	2B 05 A0 01 00 00	<code>sub eax, [a2]</code>
05B	75 C5	<code>jnz haha</code>
05D	F7 2D A0 01 00 00	<code>imul dword [a2]</code>

BEVELENTELLER

00	00	00	5D
+ FF	FF	FF	C5
<hr/>			
00	00	00	22

BEVELREGISTER

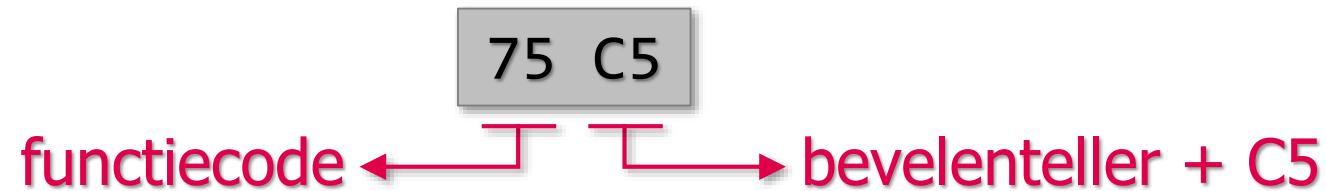
75	C5
----	----



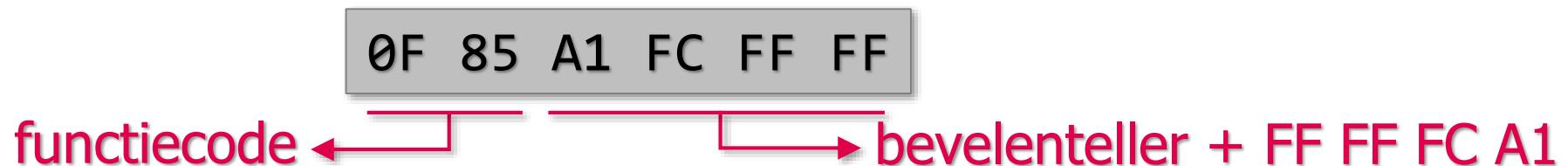
Sprongbevel

- Geen sprong: **niets doen**
 - Het adres van het bevel dat na de sprong komt, staat al in de bevelenteller
- Wel sprong: **bevelenteller aanpassen**
 - Een bedrag optellen dat staat in de bytes van het sprongbevel
 - Dit bedrag werd berekend door het **vertaalprogramma**

Hoe ver kan men springen?



- Slechts 1 byte, dus van -128 tot +127
- Soms niet genoeg, in dat geval andere functiecode



Merk op: beide sprongen zijn een 'jnz'

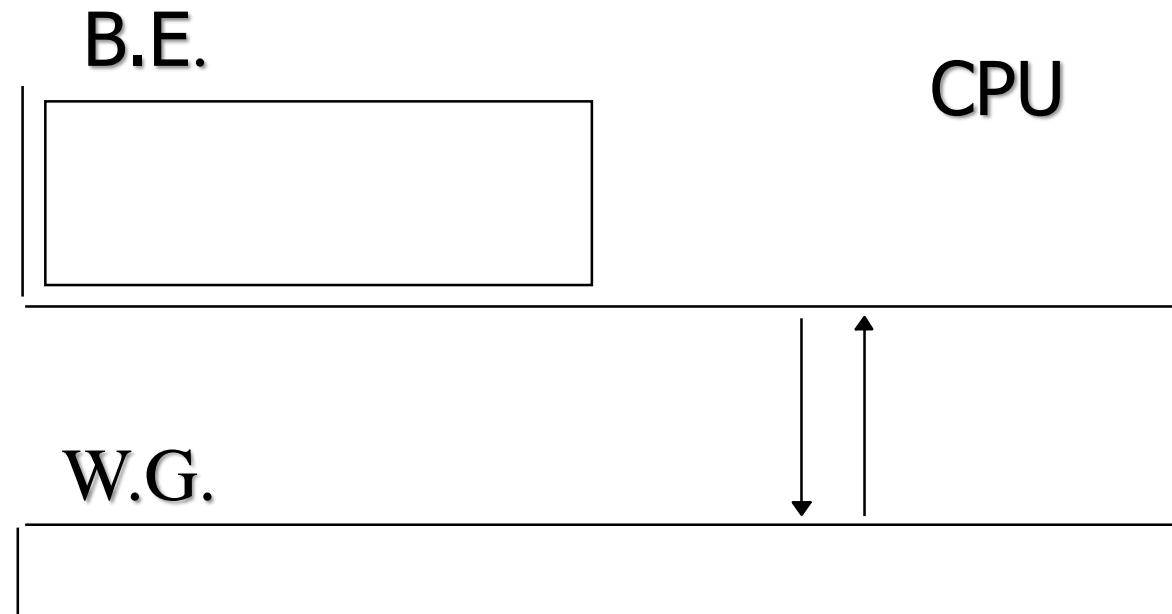


H2: Werkgeheugen, centrale verwerkseenheid, programma's

- Werkgeheugen
- Centrale verwerkseenheid: welke bevelen?
- Programma's schrijven
- De uitvoering van programma's
- **Bevelenwachtrij**
- Rekeneenheid
- Klok

Bevelenwachtrij

- Tussen werkgeheugen en CPU is er een verbinding
 - Wordt gebruikt om data te kopiëren
 - Ze is bvb. 32 bits breed (kan ook meer of minder zijn)



Bevelenwachtrij

- Wat wordt er gekopieerd?
 - van register naar werkgeheugen
`mov [uitkomst], eax`
 - van werkgeheugen naar register
`mov eax, [uitkomst]`
 - één van de operanden bij een berekening
`imul dword [factor]`
 - de bevelen zelf
- De verbinding is relatief **traag**; er kunnen nooit twee dingen tezelfdertijd gekopieerd worden.

Bevelenwachtrij

Vb.:

`add eax, [term]`

`mov [uitkomst], eax`

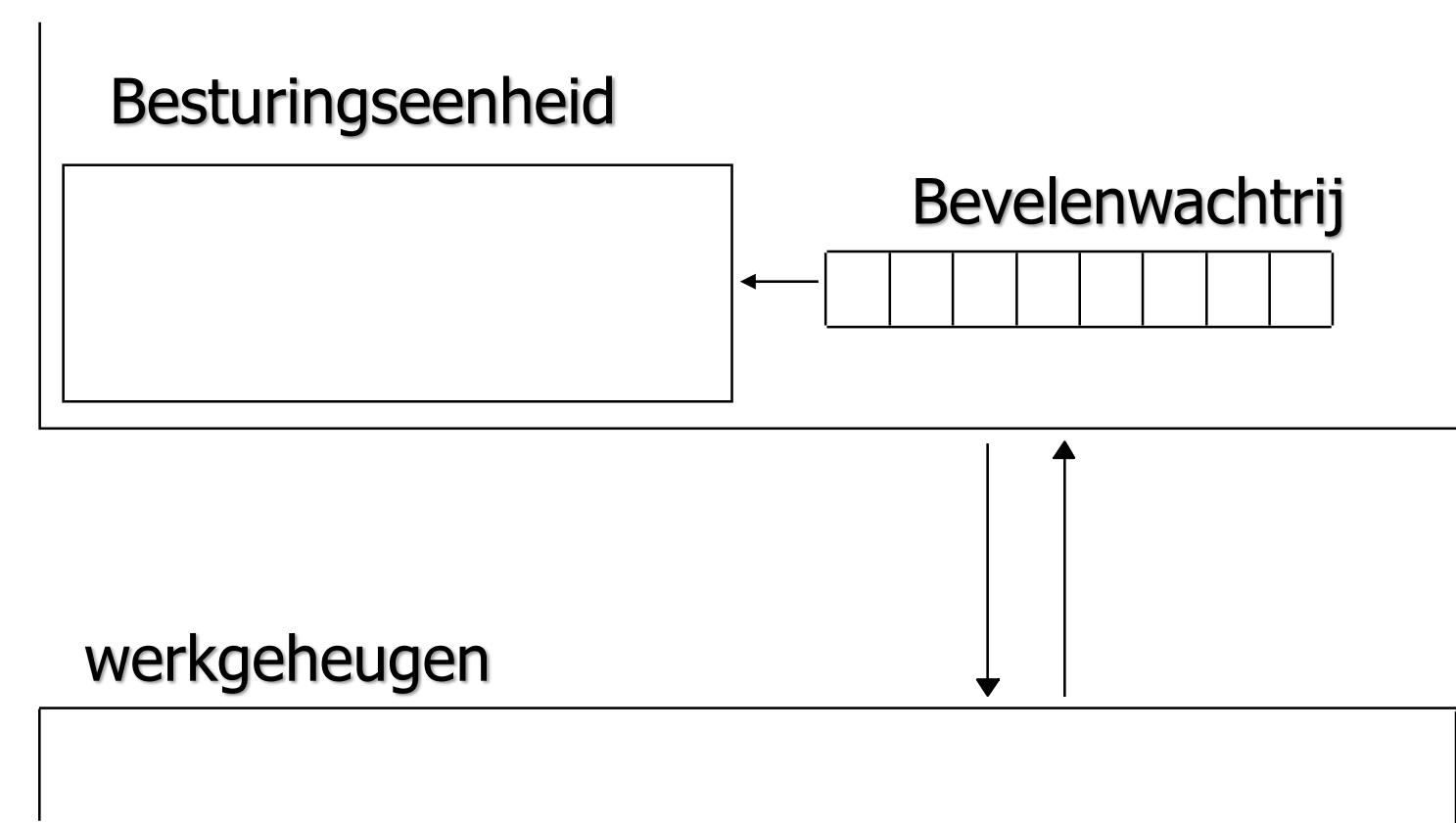
1. bevel `add eax, [term]` ophalen
(gebruik van de verbinding)
2. bevel `add eax, [term]` uitvoeren
 1. dubbelwoord met adres term ophalen
(gebruik van de verbinding)
 2. optelling uitvoeren (de verbinding is vrij)
3. bevel `mov [uitkomst], eax` ophalen
(gebruik van de verbinding)
4. ...



Bevelenwachtrij

- Slimmigheidje: het mov-bevel al ophalen tijdens de uitvoering van de optelling!

Altijd bij een moderne computer: **bevelenwachtrij** (prefetch queue)



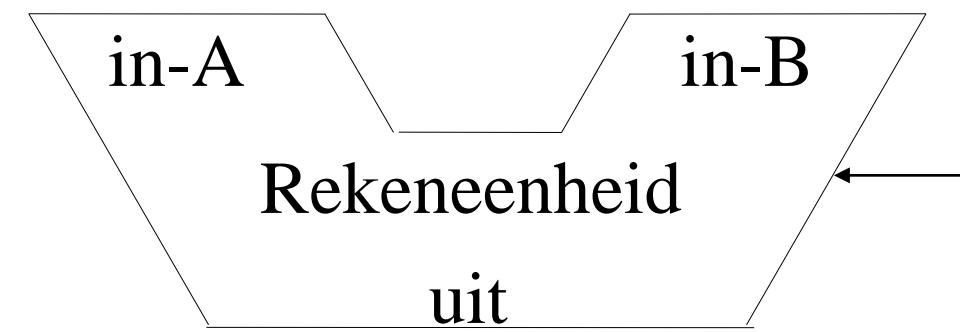


H2: Werkgeheugen, centrale verwerkseenheid, programma's

- Werkgeheugen
- Centrale verwerkseenheid: welke bevelen?
- Programma's schrijven
- De uitvoering van programma's
- Bevelenwachtrij
- **Rekeneenheid**
- **Klok**

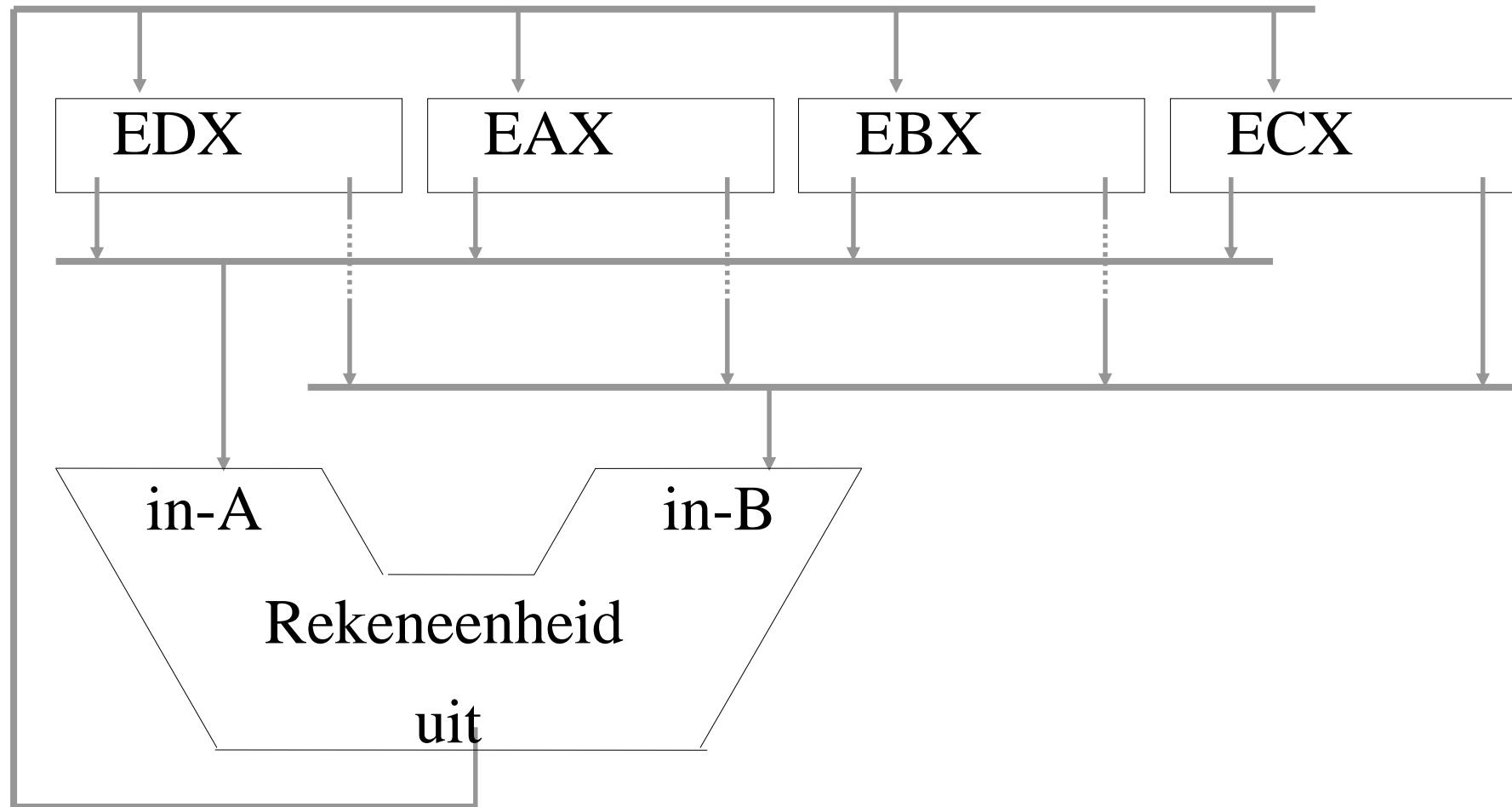
Rekeneenheid

- Arithmetic and Logical Unit (ALU)
- Uitvoering van bewerkingen: $+$, $-$, $*$, $/$, AND, OR, ...
- 2 ingangen, 1 uitgang
- Krijgt bevel ($+$, $-$, $*$, ...) van besturingseenheid



Besturingseenheid

Verbindingen in de CPU





H2: Werkgeheugen, centrale verwerkseenheid, programma's

- Werkgeheugen
- Centrale verwerkseenheid: welke bevelen?
- Programma's schrijven
- De uitvoering van programma's
- Bevelenwachtrij
- Rekeneenheid
- Klok

register A :

0	0	1	1	1	0	1	0	1	1	0	0	0	1	1	1	0	1	0	0
1	1	0	0	0	0	0	0	1	1	1	1	1	0	1	0	1	0	0	1

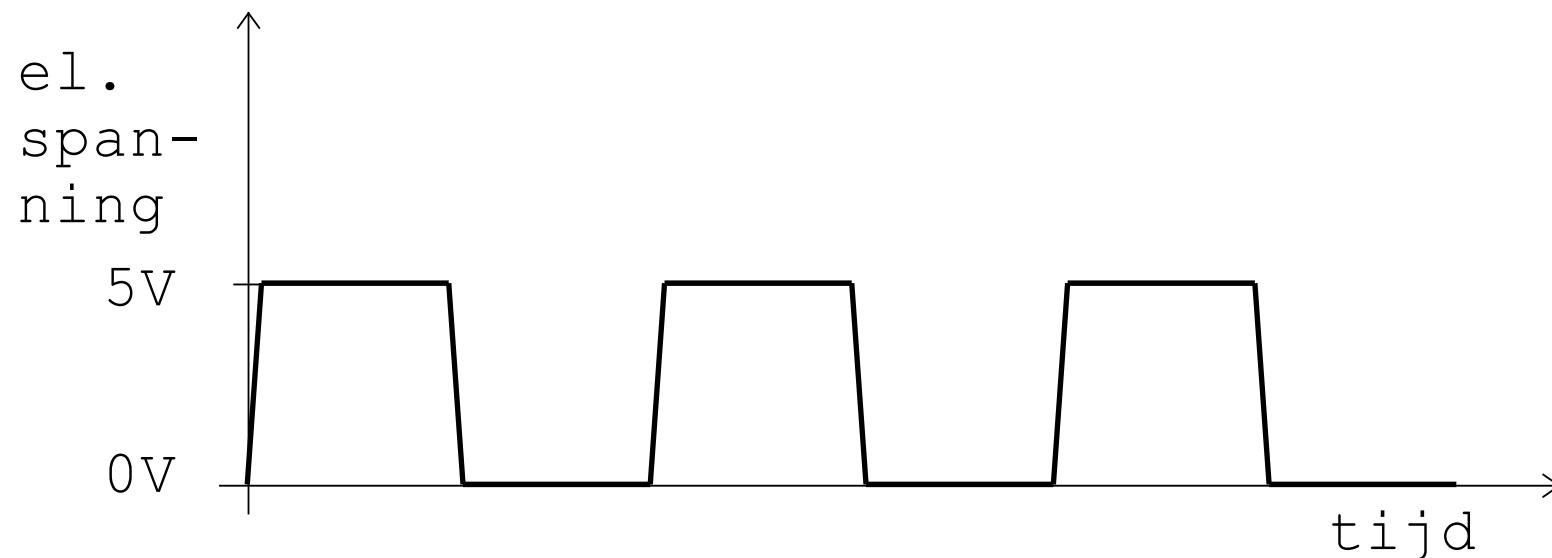
register B :

Vraag: wanneer is de berekening van een som klaar?

Vraag: hoe lang moet een verbinding gesloten zijn om te worden gesloten?

Vraag: wanneer is het ophalen van data uit het werkgeheugen klaar?

- De **klok** produceert met perfecte regelmaat spanningspulsen (een puls = een tik)
 - Bvb. 3.700.000.000 cycli per seconde, d.i. een klok van 3.7GHz



- Uitvoering van een bevel: in **stappen**, volgens het "tikken" van de klok, bvb.: `add eax, edx`
 1. `eax, edx` kopiëren op (verschillende) bus
 2. Getal van elke bus kopiëren naar één ingang van A.L.U.
 3. + bevel naar A.L.U.
 4. uitgang A.L.U. kopiëren naar bus
 5. bus kopiëren naar `eax`
- Klok als **dirigent**

- Voor elk bevel: bepaald aantal klokcycli nodig
 - sprong-bevel: 1 cyclus
 - optelling/aftrekking: 3 cycli
 - vermenigvuldiging: 18 cycli
- Als de klok sneller tikt (en als de apparatuur meekeert):
snellere uitvoering van de bevelen.
 - Als in een fabriek de lopende band sneller loopt, wordt er dan meer geproduceerd?