Assembly en Assemblers

Processoren

5 januari 2015

Doel van vandaag

- Ik heb al losse eindjes over assembly verteld en een voorbeeldprogramma doorlopen.
- vandaag: algemeen + systematisch overzicht

Programmeertalen

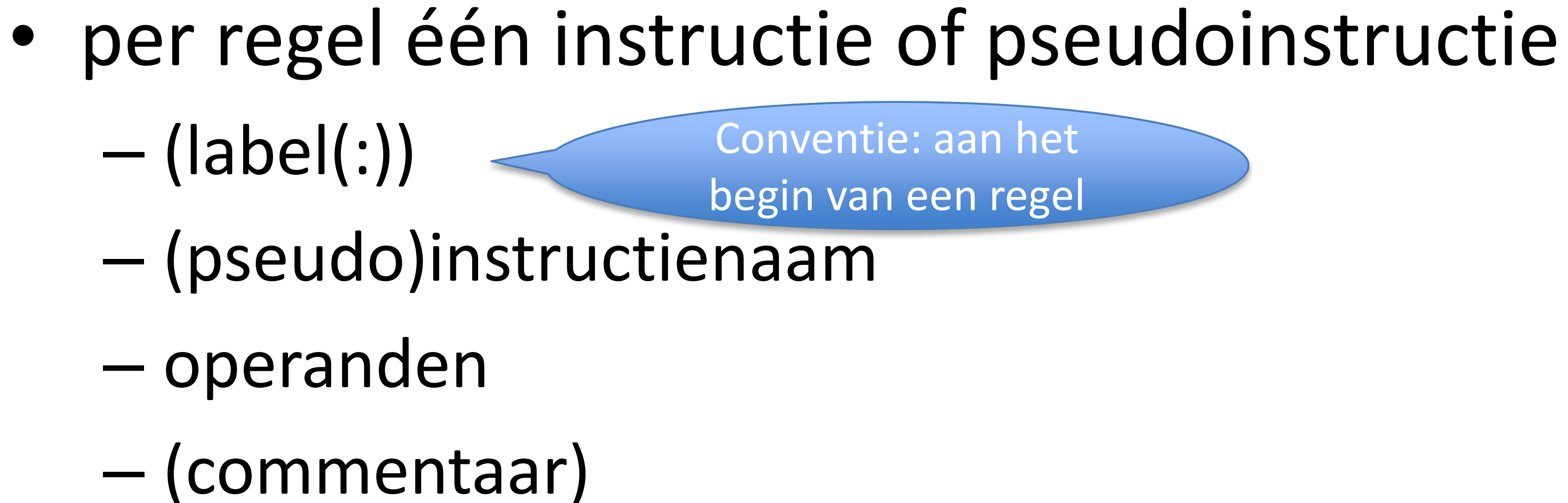
- Machinetaal
- Assembly

- Hogere programmeertaal

Voor- en nadelen van Assembly

t.o.v. machinetaal	t.o.v. hogere talen
+ makkelijker te onthouden + minder foutgevoelig	+ volledige controle van de CPU + efficiënte, kleine programma's − slechts voor één processortype − kleine instructies → foutgevoelig

Formaat van assembly-instructies



- per regel één instructie of pseudoinstructie
  - (label(:))
  - (pseudo)instructienaam
  - operanden
  - (commentaar)

Labels

- namen voor adressen in code of data
  - code: waarheen de processor kan springen

```
            JUMP.O   error
            ...
            ...
  error:    ...
```

  - data: globale variabelen

```
            READ     [length], R1
            ...
            ...
  length:   .data    177
```

Instructies

- Naam: geeft aan wat de instructie doet
  - ADD optellen
  - READ (uit RAM) lezen
  - DAA (x86) decimal adjust after addition
  - kleine verschillen tussen namen (MOV/MOVE) volgens smaak van de ontwerper
- Instructie bepaalt aantal + formaat operanden

Pseudoinstructies

- regieaanwijzingen aan de assembler
  - waar moet het programma opgeslagen worden?

```
                .org        1024
```

  - datadefinitie, b.v.

```
tabel:          .data       10,20,0x30
hellostring:    .ascii      ”Hello, world!”
```

  - macrodefinities
- lijken vaak erg op instructies

Voorbeeld: x86-assembly-programma

Label	Opcode	Operands	Comments
FORMULA:	MOV	EAX,I	; register EAX = I
	ADD	EAX,J	; register EAX = I + J
	MOV	N,EAX	; N = I + J
I	DD	3	; reserve 4 bytes initialized to 3
J	DD	4	; reserve 4 bytes initialized to 4
N	DD	0	; reserve 4 bytes initialized to 0

Tanenbaum, Structured Computer Organization, Fifth Edition, © 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Wat doet een assembler?

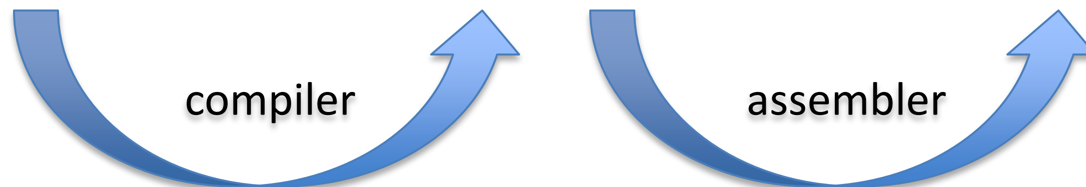
- mnemonische namen vertalen
- adressen berekenen
- (vaak) macros

- pseudoinstructies

Vertaalstappen

hogere taal	assembly	machinetaal

i := i + 3;	LOADHI 13, R2
	ADD 7, R2, R2
	READ [R2], R1
	ADD 3, R1, R1
	WRITE R1, [R2]
		800D
		06A7
		1100
		17A3
		1900



- Soms zit de assembler in de compiler ingebouwd.

Twee-pass-assembler

- Probleem: forward reference
  = label-adres is soms nog niet bekend als het gebruikt wordt

- Oplossing: programma 2× lezen
  - 1e pass: symbooltabel berekenen
  - 2e pass: instructies vertalen
    en objectbestand aanmaken

Wat rest er na assembleren?

- Grote programma's opgedeeld in modules
  - elke module apart assembleren
  - na het assembleren
    voegt een **linker** de objectbestanden samen
    tot een programmabestand

  - voordeel: bij kleine wijziging sneller

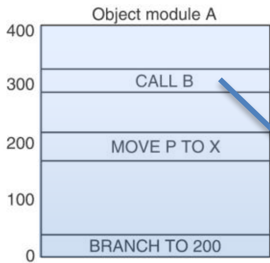
Taken van de linker

1. symbooltabellen vergelijken en samenvoegen
   - pseudoinstructies geven aan
     welke symbolen gedefinieerd of gebruikt worden

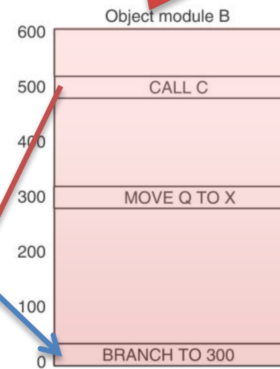
2. reloceren (eerste deel)
   - startadres van de modules kiezen
   - adressen daaraan aanpassen

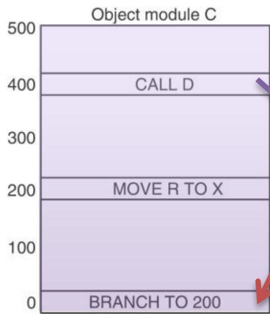
Symbooltabellen samenvoegen

gebruikt B

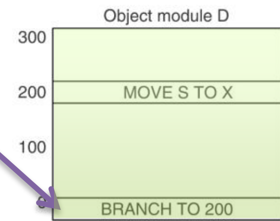


definieert B
gebruikt C

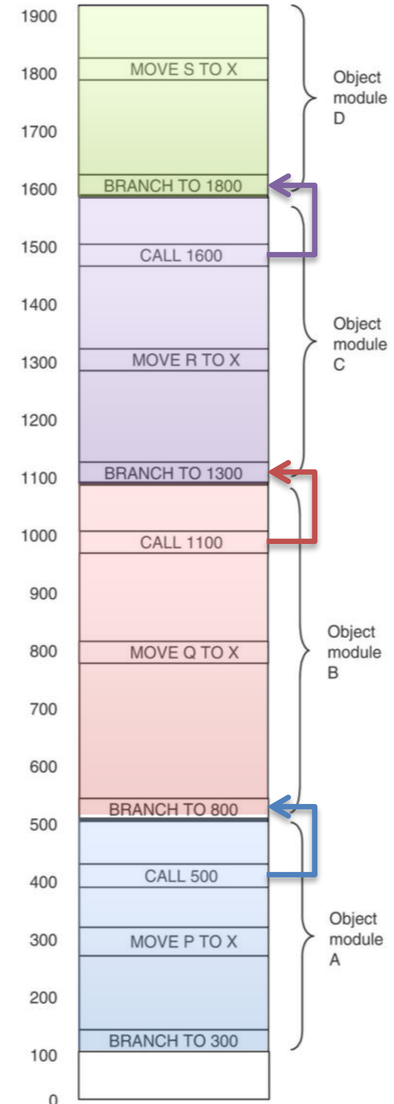




definieert C
gebruikt D

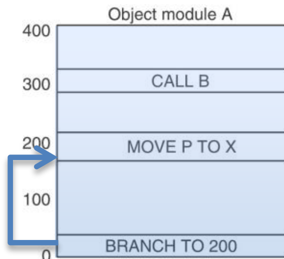


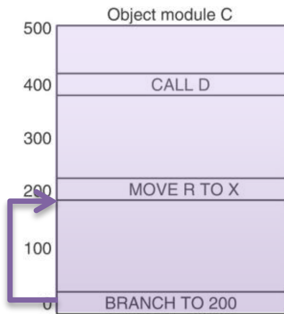
definieert D



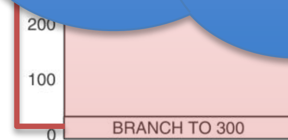
Tanenbaum, Structured Computer Organization, Fifth Edition, © 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

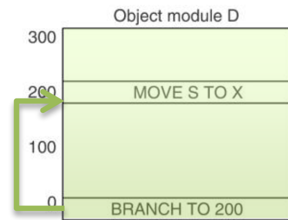
Reloceren



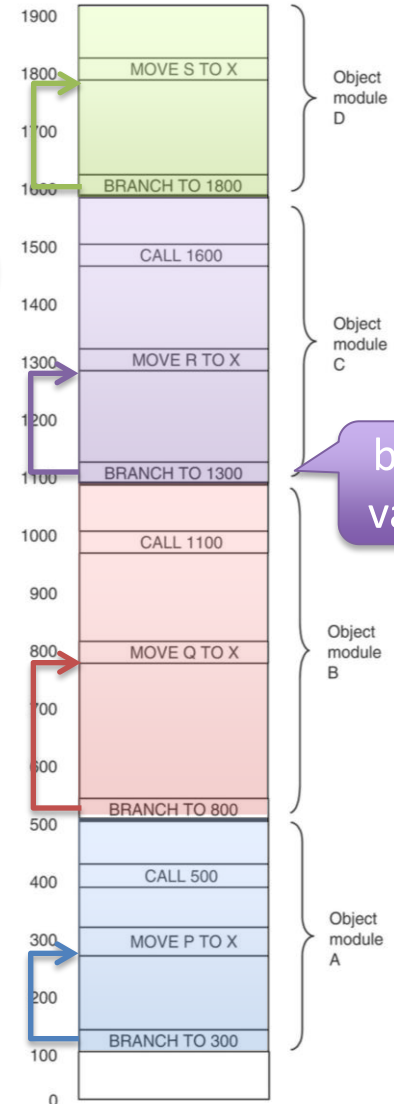


adres t.o.v. begin van C: 200





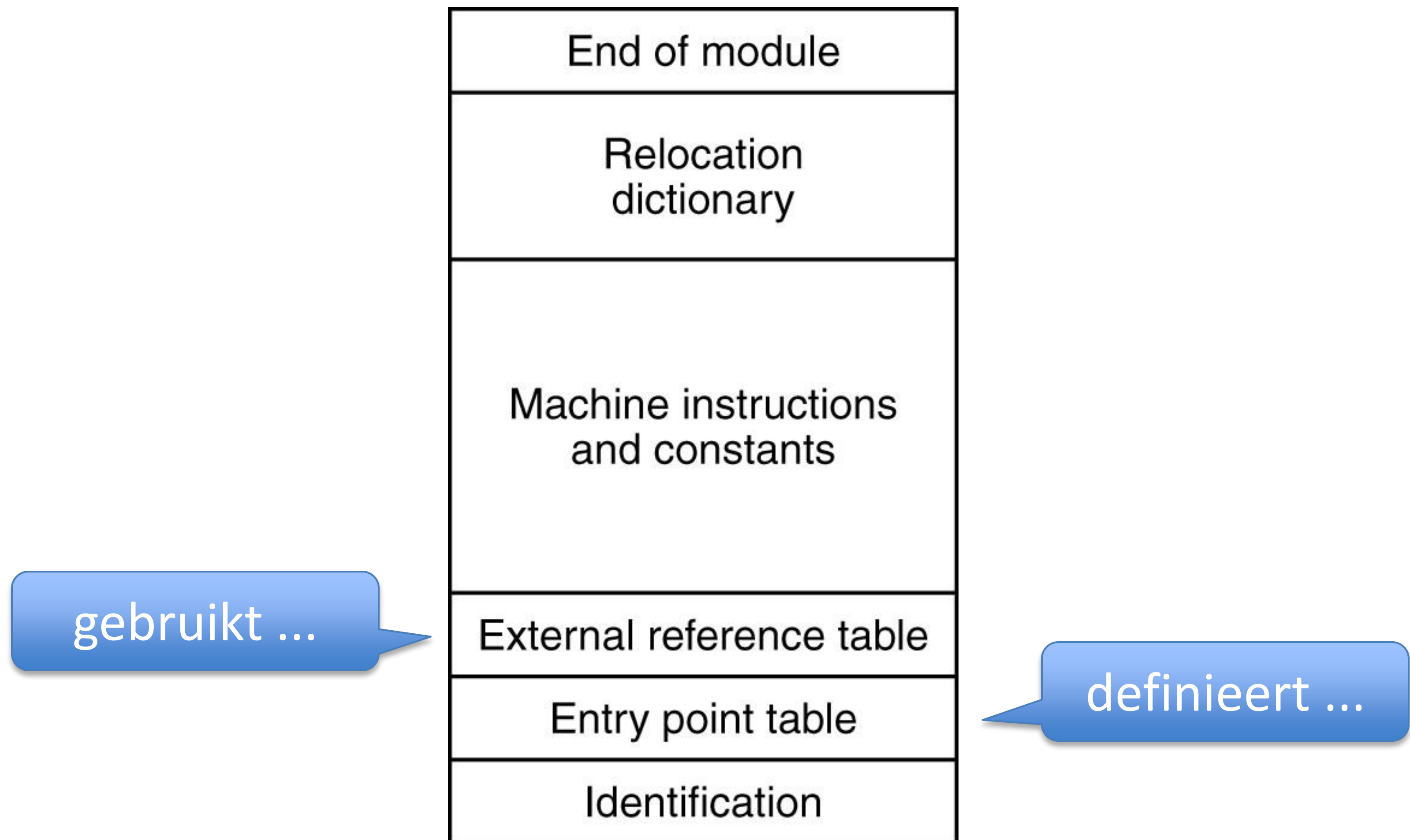
in de praktijk:
soms PC-relatieve adressen
→ relocatie niet nodig
(ADD.GE 4, PC, PC)



beginadres van C + 200

Tanenbaum, Structured Computer Organization, Fifth Edition, © 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Onderdelen van een objectbestand



Tanenbaum, Structured Computer Organization, Fifth Edition, © 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

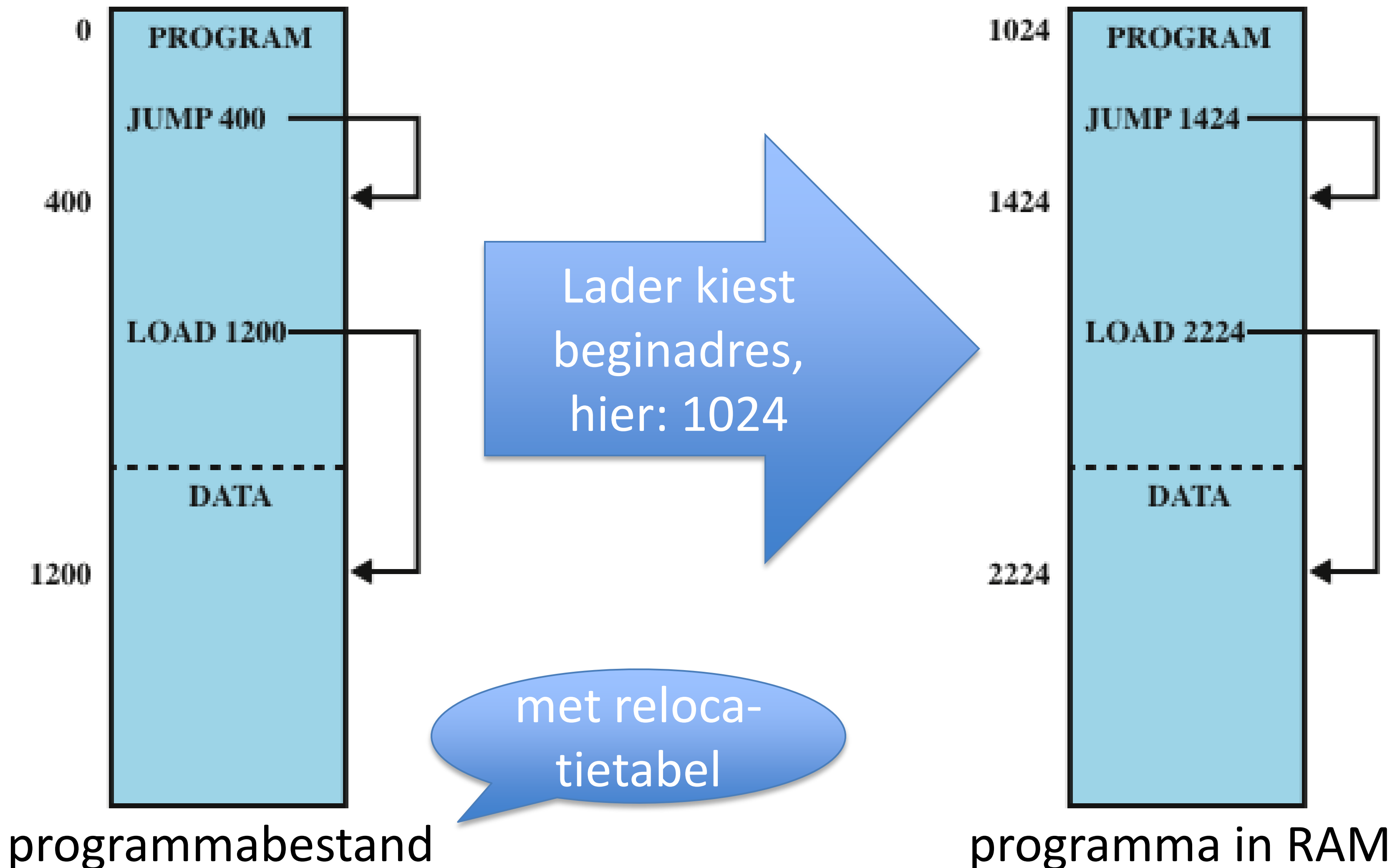
Programmabestand is nog geen programma

- **Lader** kopieert programmabestand naar RAM

- Probleem: welke plaats in het RAM?
  - vaste plaats: eenvoudig, maar inflexibel
  - variabele plaats: vereist tweede relocatiestap

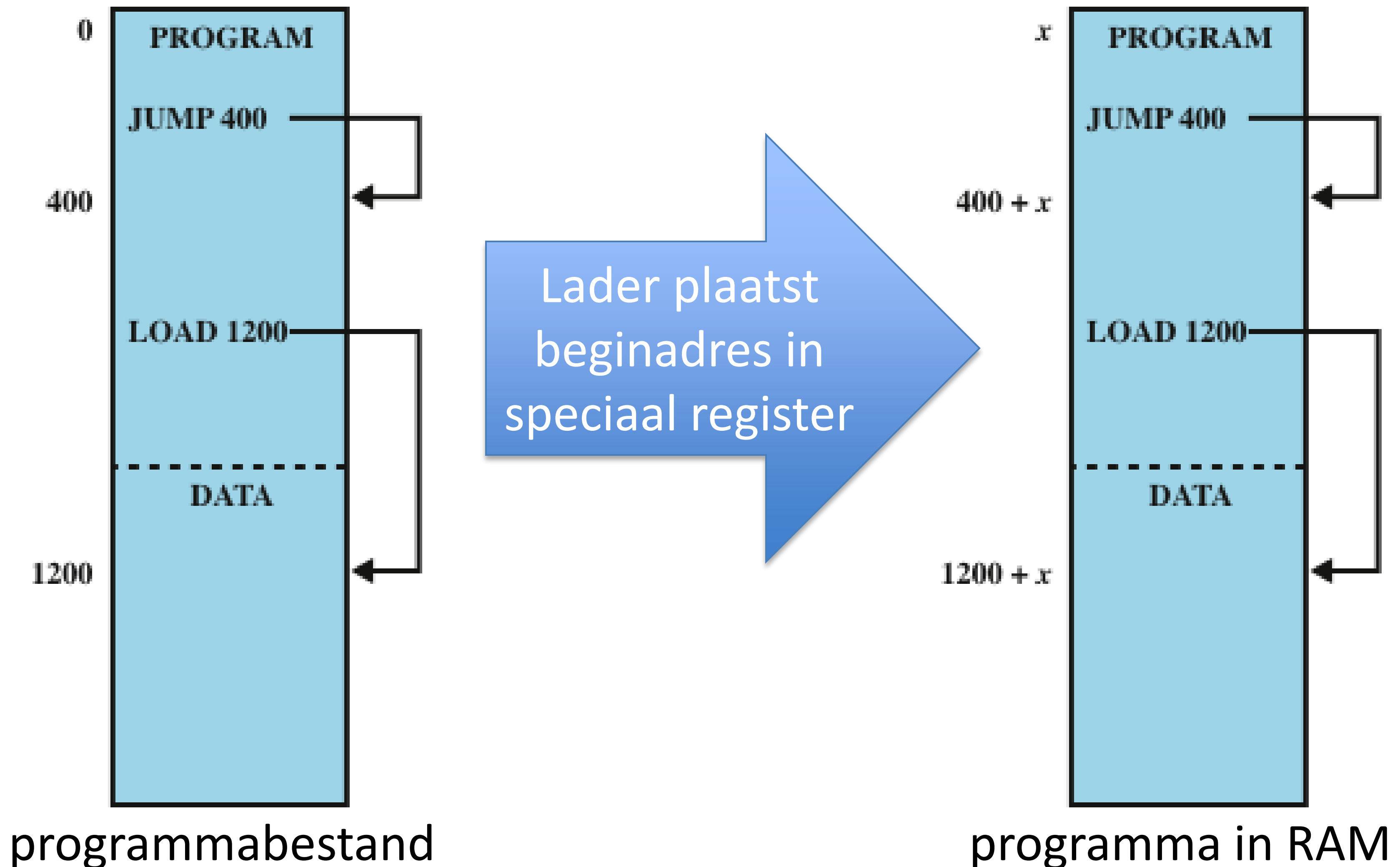
  - sommige besturingssystemen eisen dat programma alleen relatieve adressen bevat (.COM-bestand)

Voorbeeld: laden met relocatie



Stallings, Operating Systems, Sixth Edition, © 2008 Pearson Education, Inc. All rights reserved.

Voorbeeld: laden met relatieve adressen



Stallings, Operating Systems, Sixth Edition, © 2008 Pearson Education, Inc. All rights reserved.

Library

- samenvatting van meerdere objectbestanden in één groot bestand
- handzamer voor de programmeur

- linker behandelt library net als objectbestand

Standaard library

- voorgedefiniëerde functies van een compiler:
  niet echt ingebouwd,
  maar in de standaard library

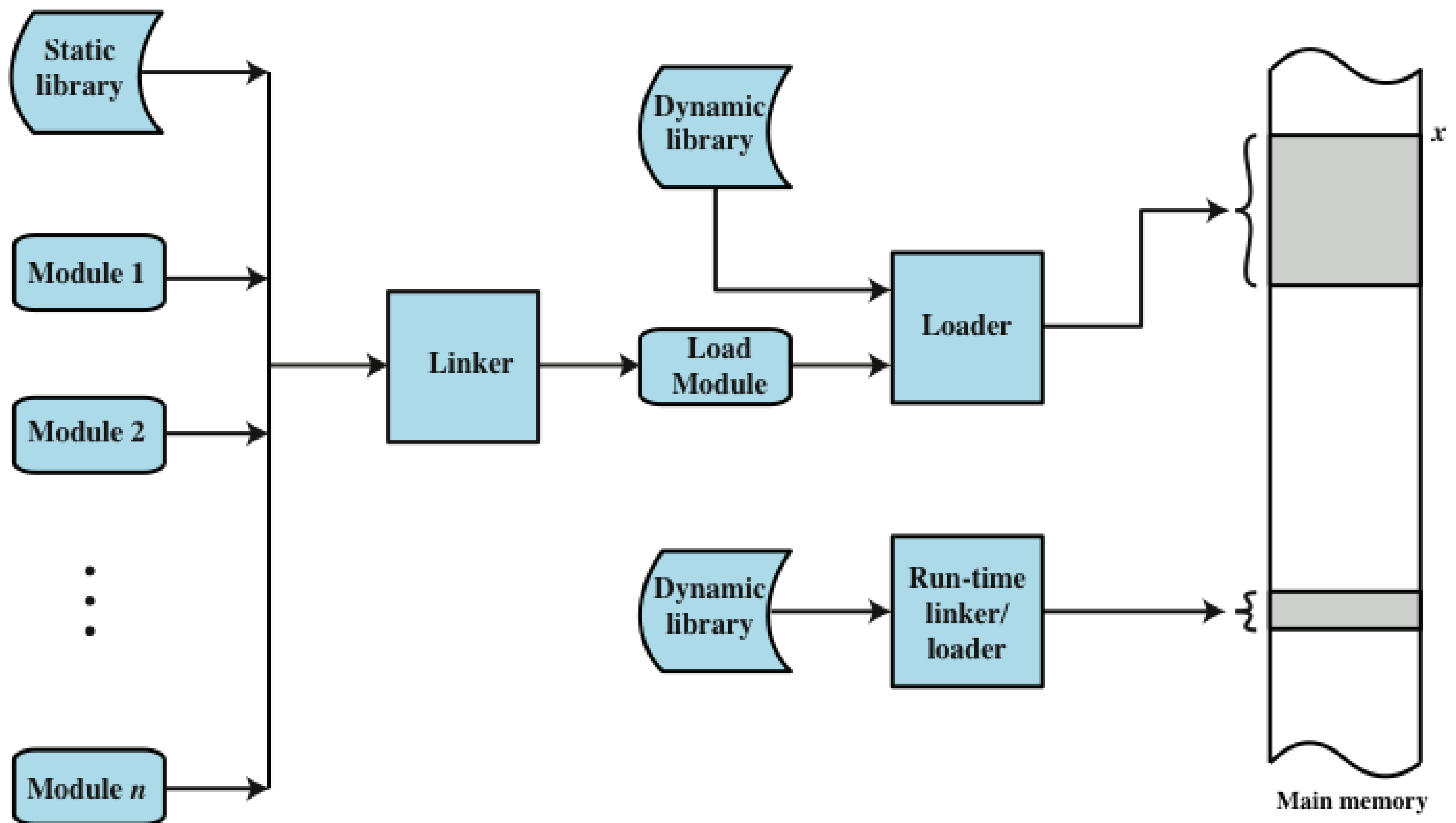
- Standaard library
  wordt bij veel programma's gebruikt
- elk programmabestand bevat dezelfde library
  – inefficiënt!

Dynamische library

- library in een apart bestand (.DLL, .so) opgeslagen i.p.v. programmabestand
- laatste link-stap gebeurt pas
  - als programma geladen wordt
  - of als programma library-functie gebruikt

- vaak gebruikte libraries worden slechts één keer opgeslagen

Voorbeeld: van objectbestand tot proces



Stallings, Operating Systems, Sixth Edition, © 2008 Pearson Education, Inc. All rights reserved.

Samenvatting

- Assembly: gemakkelijk te onthouden notatie voor machine-instructies
- Assembler vertaalt ook labels
- Linker voegt objectbestanden+libraries samen in programmabestand
- Lader kopieert programmabestand naar RAM
- Dynamische libraries: linken tijdens/na laden

Toegift:
control structures in assembly

if – then – else

volgorde van operanden verschilt: de 1e operand is het doel.

Pascal	assembly (practicum)	assembly (8086

if (i = 3) then
  j := 1
else
  k := 2;
	READ [i], R2
	SUB 3, R2, R0
	JUMP.NZ else
	MOVE 1, R2
	WRITE R2, [j]
	JUMP endif
	else: MOVE 2, R2
	WRITE R2, [k]
	endif:
		MOV AX, [i]
		CMP AX, 3
		JNZ else
		MOV [j], 1
		JMP endif
		else: MOV [k], 2
		endif:

loop

Pascal	assembly (practicum)	assembly (8086)

for i := 1 to 5 do
...;
	MOVE 1, R1
	WRITE R1, [i]
	for: ~~READ [i], R1~~
	SUB 5, R1, R0
	JUMP.L endfor
	...
	READ [i], R1
	ADD 1, R1, R1
	WRITE R1, [i]
	JUMP for
	endfor:
		MOV [i], 1
		for: MOV AX,[i]
		CMP AX, 5
		JG endfor
		...
		MOV AX, 1
		ADD [i], AX
		JMP for
		endfor:

Troostend slotwoord

Ik begrijp wel
dat niet iedereen een werkende processor krijgt,
maar ik ben ervan overtuigd
dat jullie zo meer geleerd hebben
dan wanneer ik slechts
een werkende processor had gedemonstreerd.