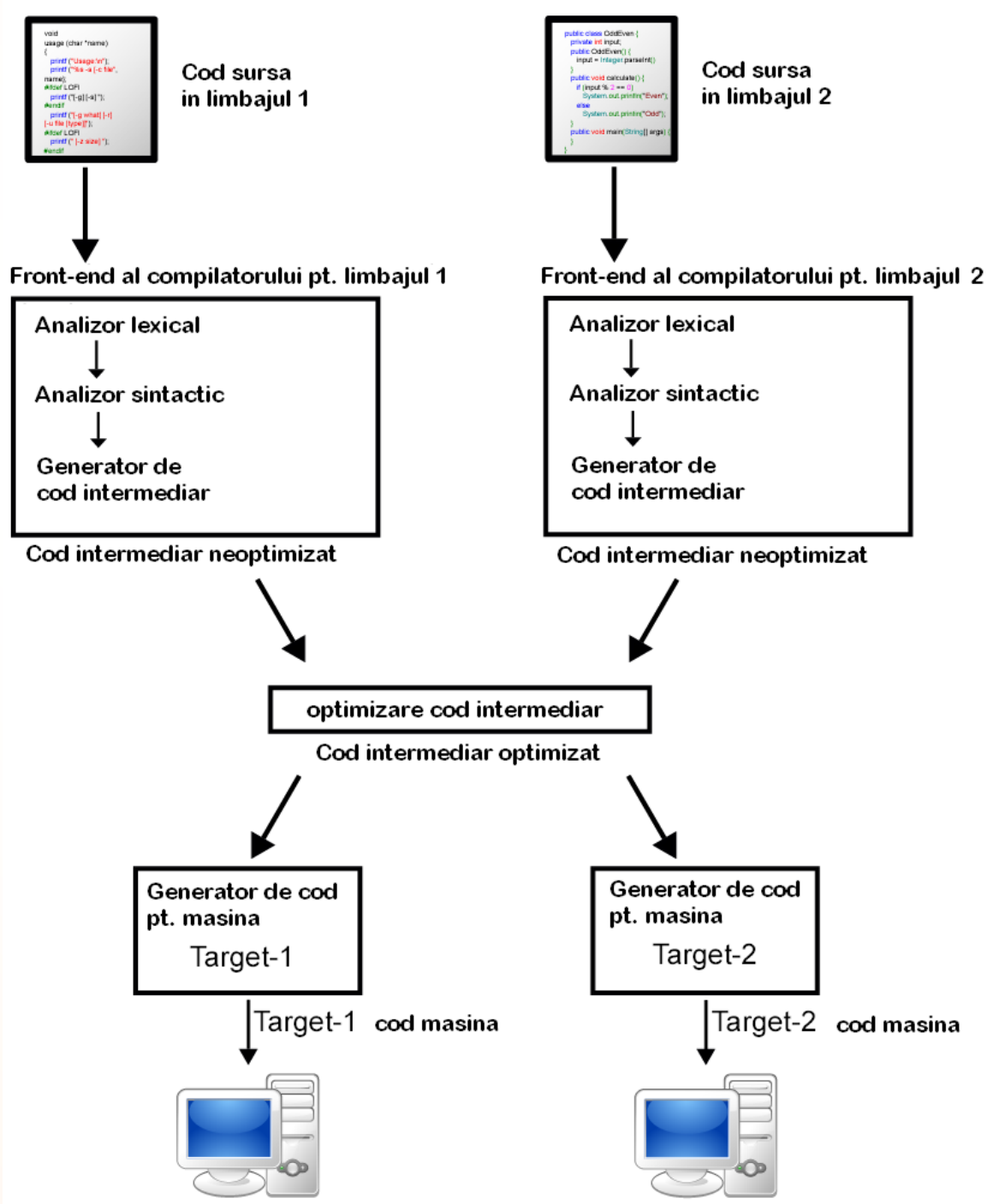
Compilator: multi-limbaj, multi-target (*masina*)

in stransa legatura cu separarea fazelor



Exemple:

CIL **- Common Intermediate Language**

– Microsoft .Net

anterior cunoscut sub numele MSIL

(Microsoft Intermediate Language)

RTL - register transfer language

- GNU Compiler Collection

- multe alte compilatoare

Codul intermediar

- limbaj intermediar:
  - usor de transcris din arborele sintactic
  - usor de translatat in cod masina

  - proiectat inclusiv pt. a fi inteles/utilizat de oameni
  - mai apropiat de limbajul procesorului decat limbajul sursa

Codul intermediar

- limbaj intermediar:
  - usor de transcris din arborele sintactic
  - usor de translatat in cod masina

Reprezentari intermediare:

"*intre*" arborele de analiza sintactica si ASM

- high-level: mentine structura limbajului
- mid-level: independent de limbaj si masina ***(tinde sa fie)***
- low-level: dependent de masina

Codul intermediar

- limbaj intermediar:
  - mai apropiat de limbajul *procesorului* decat limbajul sursa

limbaj de asamblare

In mod uzual, o masina de tipul x86 are:

- Registri
- Memorie, incluzand: stiva , programul propriu-zis
- Instructiunile limbajului masina, incluzand instr. de
  - mutare a datelor intre registri si memorie
  - calcul: aritmetice ,…

Codul intermediar

- limbaj intermediar:
  - usor de transcris din arborele sintactic
  - usor de translatat in cod masina

Reprezentari :

- expresii bazate pe arbori

- cod intermediar cu 3 adrese
  - cvadruple, triplete, …

Arbori sintactici

"***Abstract syntax trees***"

Alte denumiri **(intalnite in literatura de specialitate, in lb/ romana)**

- arbori de sintaxa abstracta
- arbori sintactici abstracti

Arbori sintactici

- reprezentare apropiata de structura sintactica a programelor
- nu este arborele de derivare, ci o varianta simplificata

Proprietati:

- nodurile interioare sunt operatori
- descendentii unui nod sint *operanzii* lui

  ➔ fiecare subarbore formeaza o "*unitate logica*"

Exemplu: **phc**

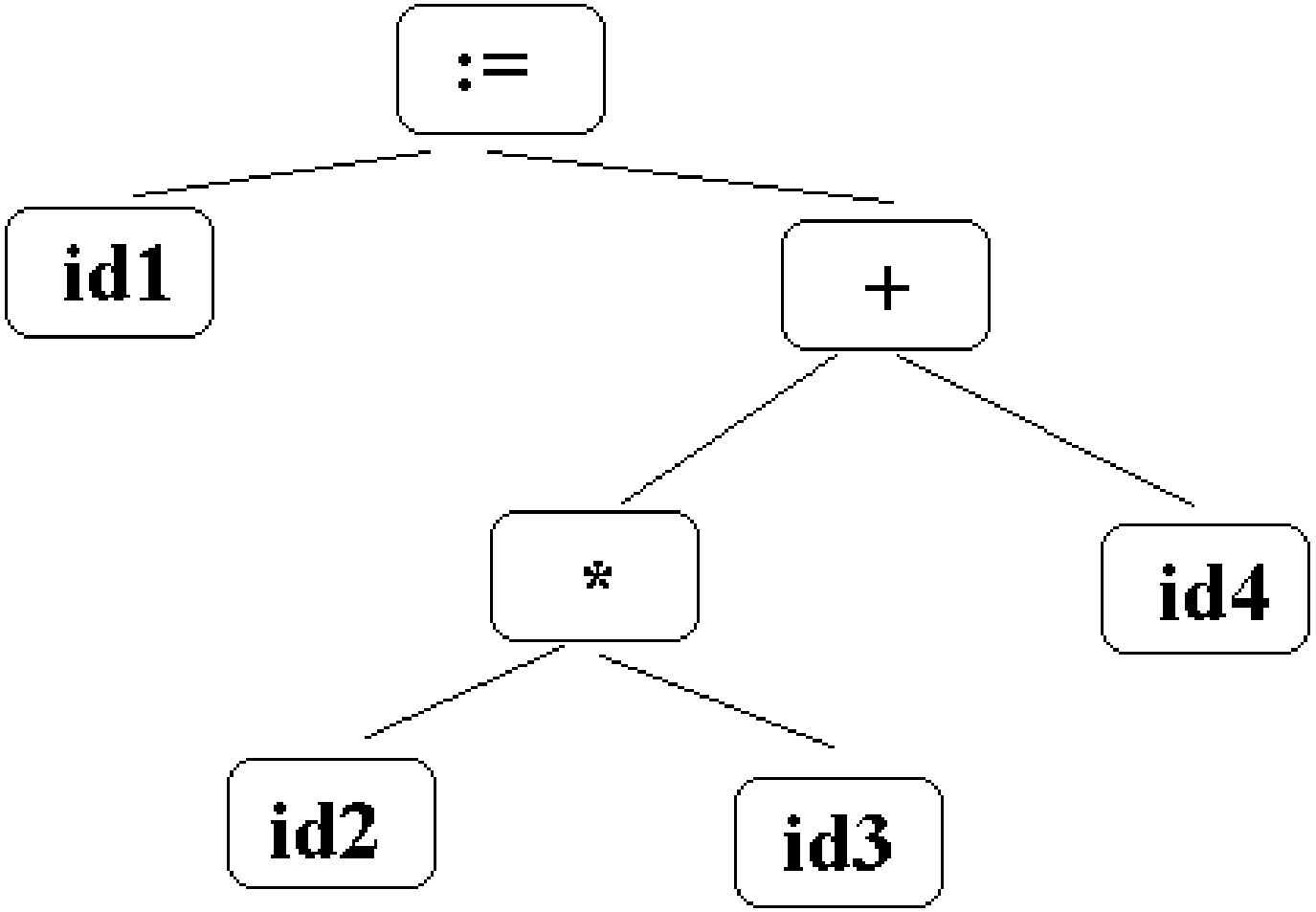
compilator open source pentru PHP

scriptul PHP este reprezentat intern ca

un arbore sintactic abstract

Arbori sintactici

- Exemplu: id1:=id2*id3+id4



Arbori sintactici

Exercitii/exemple:

Dati arb. sint. abstr. pentru urmatoarea instr. if:

- if id1>id2 then id3:=id2
  else id3:=id1

Dati arb. sint. abstr. pentru urmatoarea instr. while:

- while id1>id2 do
  id1:=id2-id1

Forma poloneza

- aceeasi idee ca si la:

  forma poloneza postfixata pentru expr. aritmetice
- <u>exemplu</u>:

  id1 := id2 * id3 + id4 => id1 id2 id3 * id4 + :=

- operatorii apar in ordinea in care se executa operatiile

  <u>Avantaj</u>:

  evaluarea: parcurgand o singura data expresia si executand operatiile tinand cont de aritatea lor

Cod intermediar cu 3 adrese

- secventa de instructiuni cu forma generala:
  <rezultat> := <arg1> <operator> <arg2>
- operatii:
  - binare
  - unare – se reprezinta doar un operand
- reprezentare
  - cvadruple
  - triplete
  - triplete indirecte

ex.
expresii aritmetice

C.i. cu 3 adrese – repr. cvadruple

- structura tip inregistrare ce contine 4 campuri:

operator	arg1	arg2	rez

Exemplu:

A := B*(C+D)

operator	arg1	arg2	rez
…	…	…	…
+	C	D	T1
*	B	T1	T2
:=	T2		A

C.i. cu 3 adrese – repr. triplete

- structura tip inregistrare ce contine 3 campuri:

operator	arg1	arg2

- se renunta la introducerea numelor temporare ce stocheza rezultate intermediare
- se considera ca instructiunea care calculeaza o valoare temporara retine acea valoare

C.i. cu 3 adrese – repr. triplete

Exemplu:

A := B*(C+D)

	operator	arg1	arg2
…	…	…	…
(51)	+	C	D
(52)	*	B	(51)
(53)	:=	A	(52)

C.i. cu 3 adrese – r. triplete indirecte

- codul contine instructiunile intr-o ordine oarecare
- pentru a obtine ordinea in care se executa operatiile, se foloseste un tabel suplimentar cu 2 campuri:

nr. de ordine a operatiei	nr. operatiei propriu-zise

C.i. cu 3 adrese – triplete indirecte

nr. de ordine a operatiei	nr. operatiei propriu-zise
51	131
52	132
53	133

	operator	arg1	arg2
…	…	…	…
(131)	+	C	D
(132)	*	B	(131)
(133)	:=	(132)	

Observatii:

- cvadruplele necesita mai mult spatiu, datorita numelor temporare
- spatiu:

  cvadruple > triplete indirecte > triplete
- triplete indirecte:

  orice rearanjari ale codului (faza de optimizare) implica modificari doar in ordinea adreselor, si nu in ordinea listei de triplete

Cvadruple (conventii pt. ex. cu care vom lucra noi)

- operanzi: constanta numerica
  valoarea unei variabile
- **operanzi speciali**
  - **@ adresa variabilei**
  - **^ variabila de la adresa indicata de valoarea variabilei**
- operatii
  - aritmetice binare: +, * , …
  - aritmetice unare: -
  - de atribuire (copiere): :=
  - salt neconditionat goto et
  - salt conditionat g<operlogic> exp1 exp2 et

Exercitii: traduceti in cod intermediar cu 3 adrese

- a:=b+(c*d)
- a:=7
- PP: *a,b: byte*

  while a>b do a:=a-b;
- PP: *a,i: byte*

  for i:=1 to a do …
- PP: *t:array[..5] of byte*

  t[3] :=7
- PP *int t[5][10];*

  t[3][7] = 20;
- PP *a : record*

  x:byte;

  y:real;

  z:integer

  end

  a.y :=2

Optimizare cod intermediar

rearanjarea codului intermediar in vederea obtinerii unui program mai eficient

optimizari locale

- realizarea unor calcule in mom. compilarii
- eliminarea operatiilor redundante si a expresiilor comune
- eliminarea codului inaccesibil *(secvente moarte)*
- scurtcircuitarea expresiilor logice

optimizarea ciclurilor

- factorizarea invariantilor de cicluri
- …