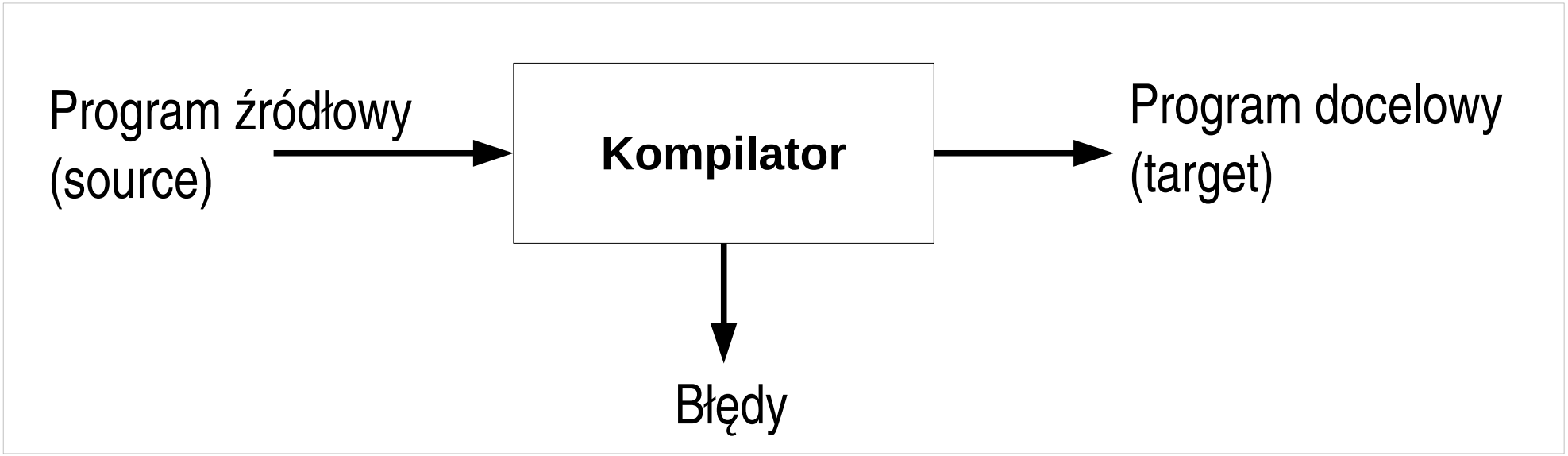
Techniki kompilacji

A.Aho,R.Sethi,J.Ullman "*Compilers:Principles,Techniques and Tools*"
J.Elder - "*Compiler Construction: A Recursive Descent Model*"
Materiały dostępne w Internecie np. www.compilerconstruction.org

Kompilator jako narzędzie tłumaczące



Standardowo: język źródłowy = język wysokiego poziomu
język docelowy = kod maszynowy procesora

Różne koncepcje kompilatorów (translatorów):

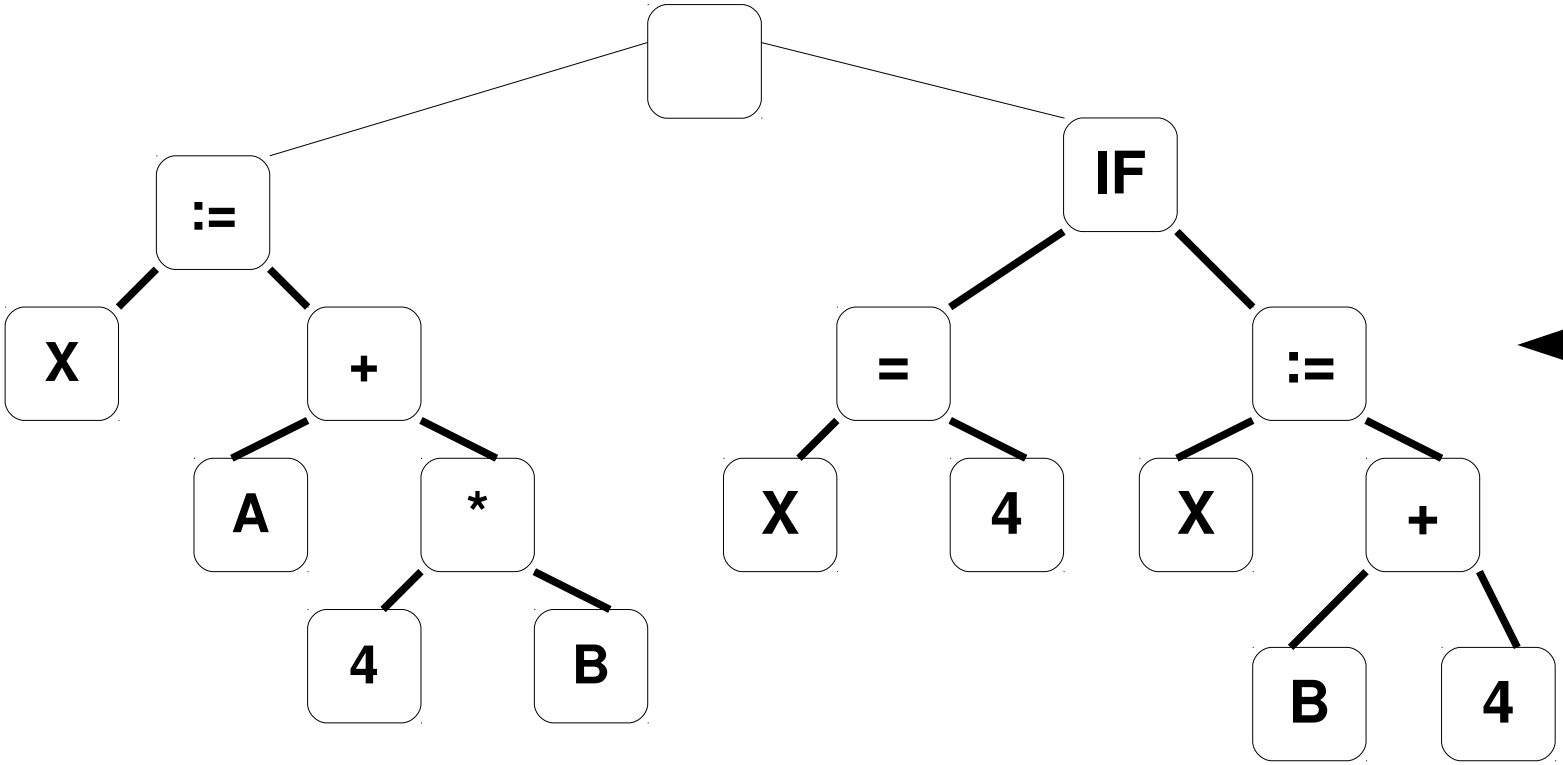
- single-pass / multi-pass
- optimizing
- debugging
- load-and-go
- just-in-time
- silicon compilers
- kompilacja “skrośna” (cross-compiling)
- dekompilacja

Języki kompilowane a interpretowane

Dwie fazy kompilacji

- Analiza - rozpoznajemy instrukcje programu i zapisujemy w postaci pośredniej (drzewo, graf, język)
- Synteza - z postaci pośredniej generujemy program w języku docelowym

X := A + 4 * B; IF X=4 THEN X := B+4;



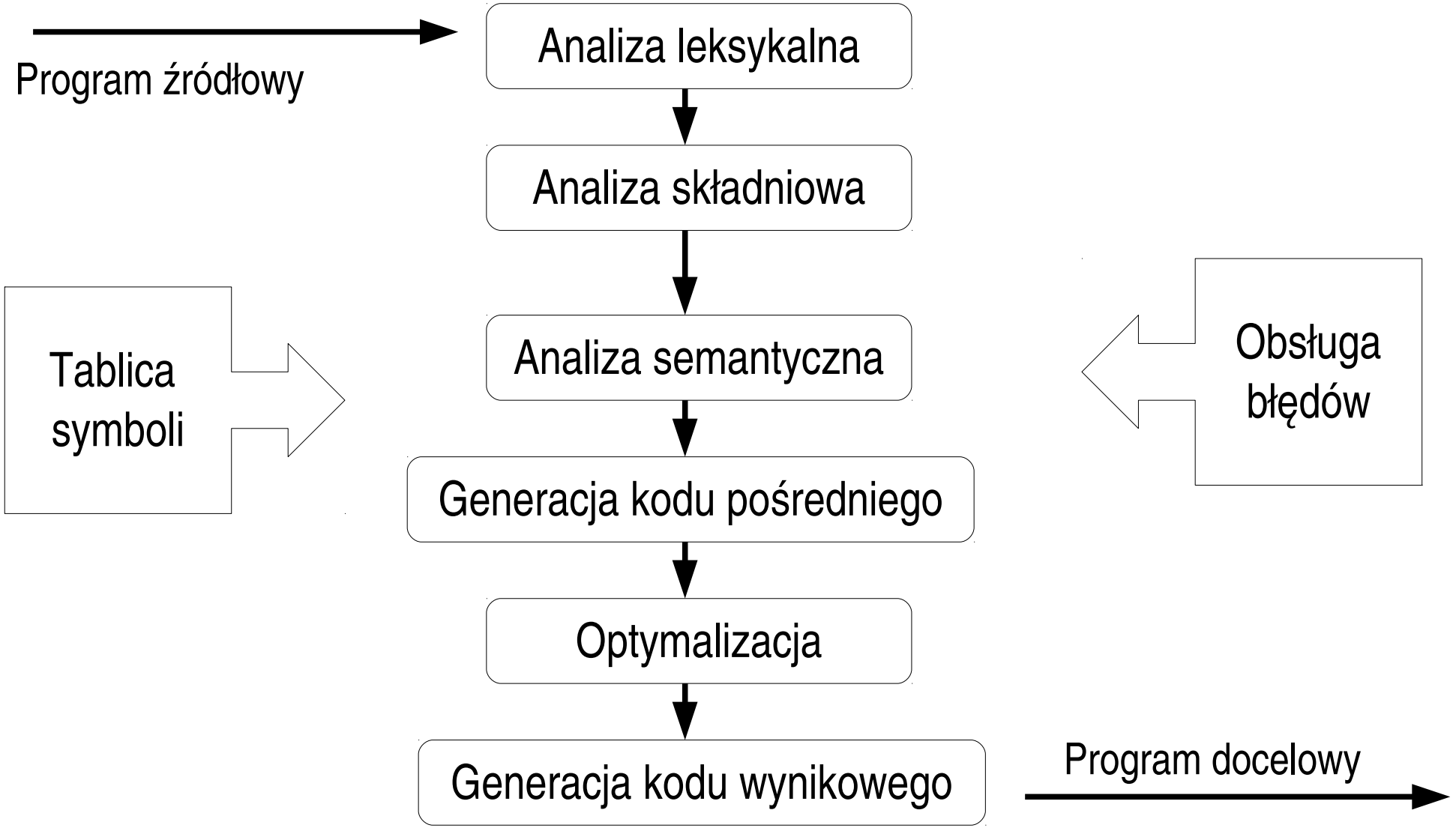
ANALIZA

SYNTEZA

```
MULT    B, 4, R1
ADD     A, R1, R2
STORE   X, R2
```

```
STORE   4, R3
SUB     R2, R3, R4
JMPN0   R4, 3
ADD     B, 4, R1
STORE   X, R1
...
```

Fazy kompilatora - model szczegółowy



Omówienie kolejnych faz

Front-end:

Analiza leksykalna
- atomy leksykalne
- błędy rozpoznawane

Analiza składniowa
- budowa drzewa wywodu
- błędy

Analiza semantyczna
- pojęcie semantyki języka
- dekorowanie i przebudowa drzewa
- błędy

Tablica symboli
- rodzaje elementów

Back-end:

Generacja kodu pośredniego
- formy kodu

Optymalizacja
- rodzaje

Generacja kodu wynikowego

Zagadnienia dodatkowe

Problematyka wydobywania się z błędów
Wykorzystanie podziału na fazy do konstrukcji kompilatorów “wielojęzykowych”
Narzędzia wykorzystujące fazę analizy
Programy współpracujące z kompilatorem:
- Preprocesor (makra, rozszerzenia języka, dołączanie plików)
- Assembler (relokowalność kodu)
- Program konsolidujący (linker)
- Program ładujący (loader)
- Odśmiecacz (garbage collector)

Narzędzia do budowy kompilatorów