podstawy i języki programowania

pjp

Podstawy programowania

Część pierwsza

Od języka symbolicznego do języka wysokiego poziomu

Autor

Roman Simiński

Kontakt

roman.siminski@us.edu.pl

www.programowanie.siminskionline.pl

Niniejsze opracowanie zawiera skrót treści wykładu, lektura tych materiałów nie zastąpi uważnego w nim uczestnictwa.
Opracowanie to jest chronione prawem autorskim. Wykorzystywanie jakiegokolwiek fragmentu w celach innych niż nauka własna jest nielegalne.
Dystrybuowanie tego opracowania lub jakiejkolwiek jego części oraz wykorzystywanie zarobkowe bez zgody autora jest zabronione.

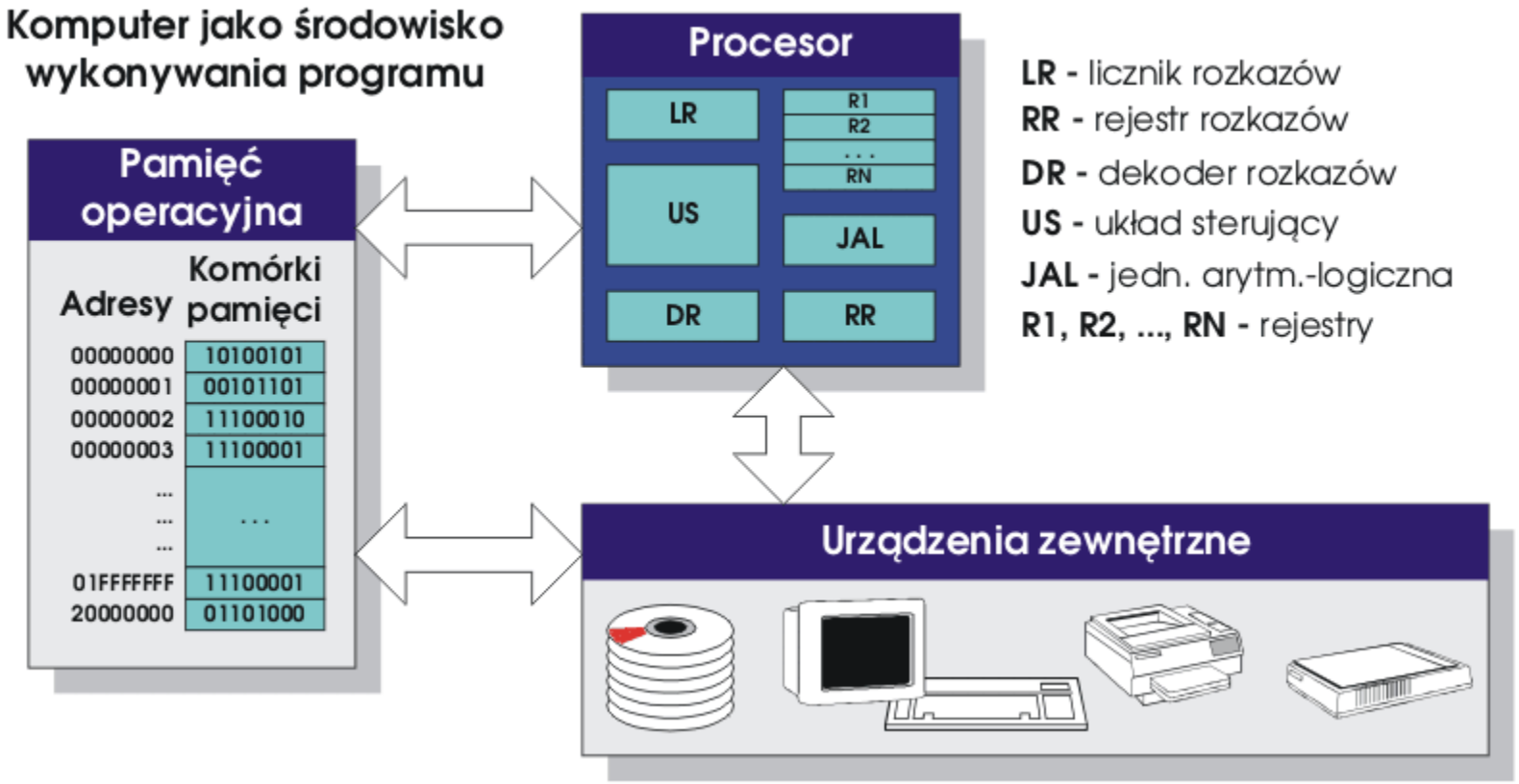
Plan wykładu

- Program wykonywalny, kod maszynowy.
- Kod maszynowy, język symboliczny.
- Język symboliczny, asembler.
- Języki wysokiego poziomu a kod maszynowy.
- Różne spojrzenia na podział i rozwój języków programowania.
- Rozwój metod i technik programowania.

Copyright © Roman Simiński

Sprzęt a oprogramowanie — powtórka z architektury komputerów

Kod maszynowy to jedyna postać programu komputerowego, która jest zrozumiała dla komputera i może być przez niego wykonana.

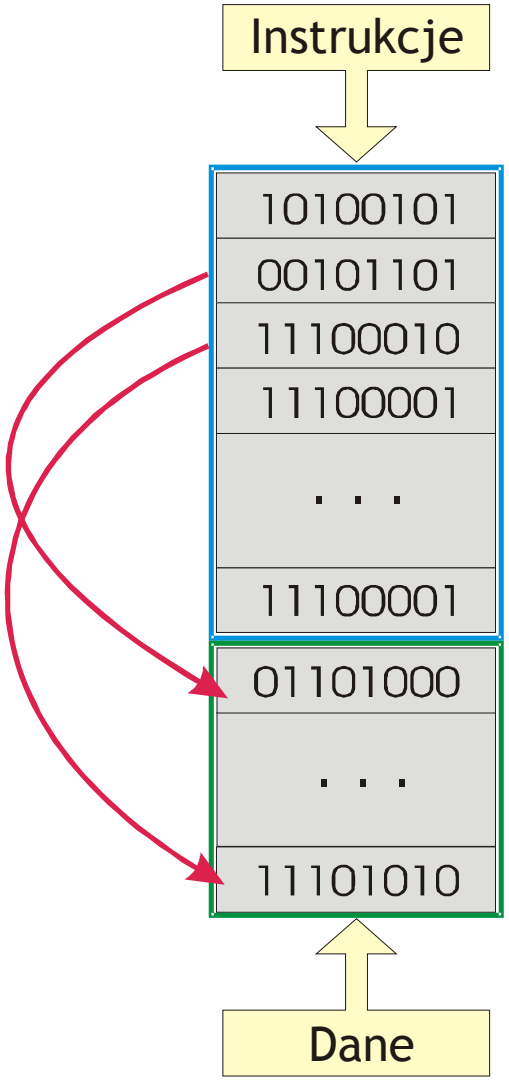


Kod maszynowy to ciąg rozkazów zrozumiałych dla procesora. Procesor pobiera, dekoduje i wykonuje te rozkazy. Rozkazy pobierane są z komórek pamięci operacyjnej, gdzie są zapisane w postaci binarnej.

Copyright © Roman Simiński

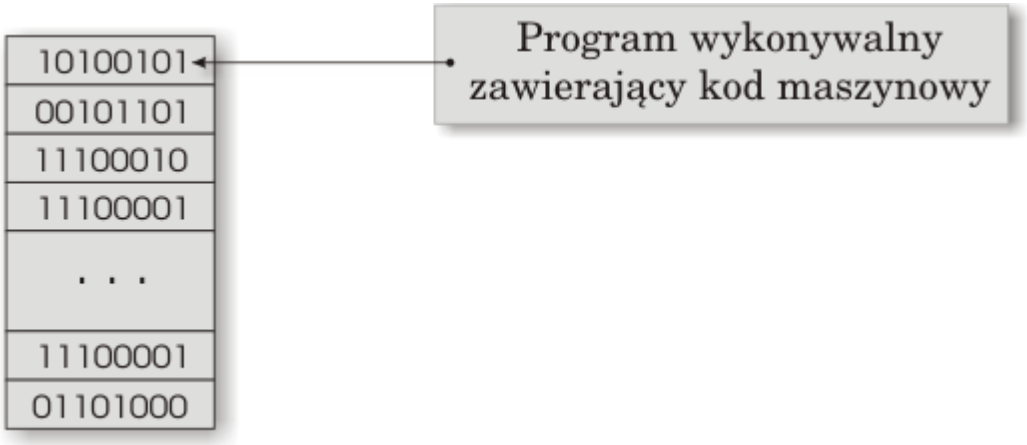
Sprzęt a oprogramowanie — powtórka z architektury komputerów

- Kod maszynowy to ciąg rozkazów procesora, inaczej instrukcji maszynowych oraz danych, na których te rozkazy operują.
- Rozkazy i dane zapisywane są w postaci liczb binarnych.
- Każdy procesor ma swoją własną listę rozkazów, rozkazy te zwykle różnią się od rozkazów innych procesorów.
- Kod maszynowy jest dedykowany dla procesora danego typu.
- Kod maszynowy konkretnego procesora zwykle jest niezrozumiały dla innego procesora.
- Procesor wykonuje tylko kod maszynowy załadowany do pamięci operacyjnej komputera (ROM lub RAM).



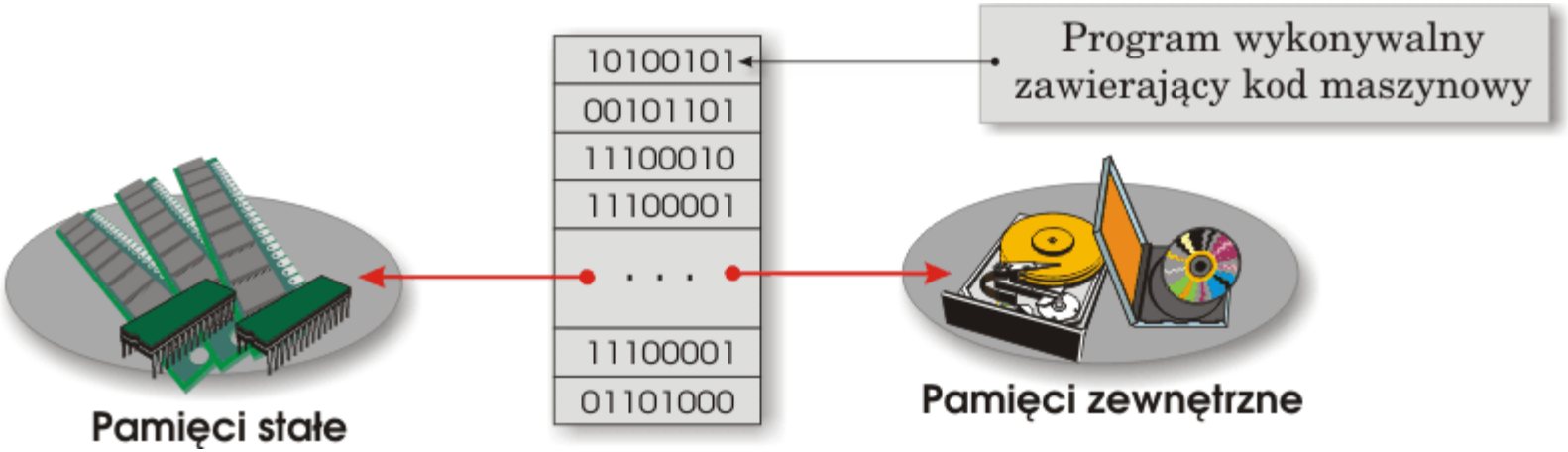
Copyright © Roman Simiński

Od kodu maszynowego do języka symbolicznego



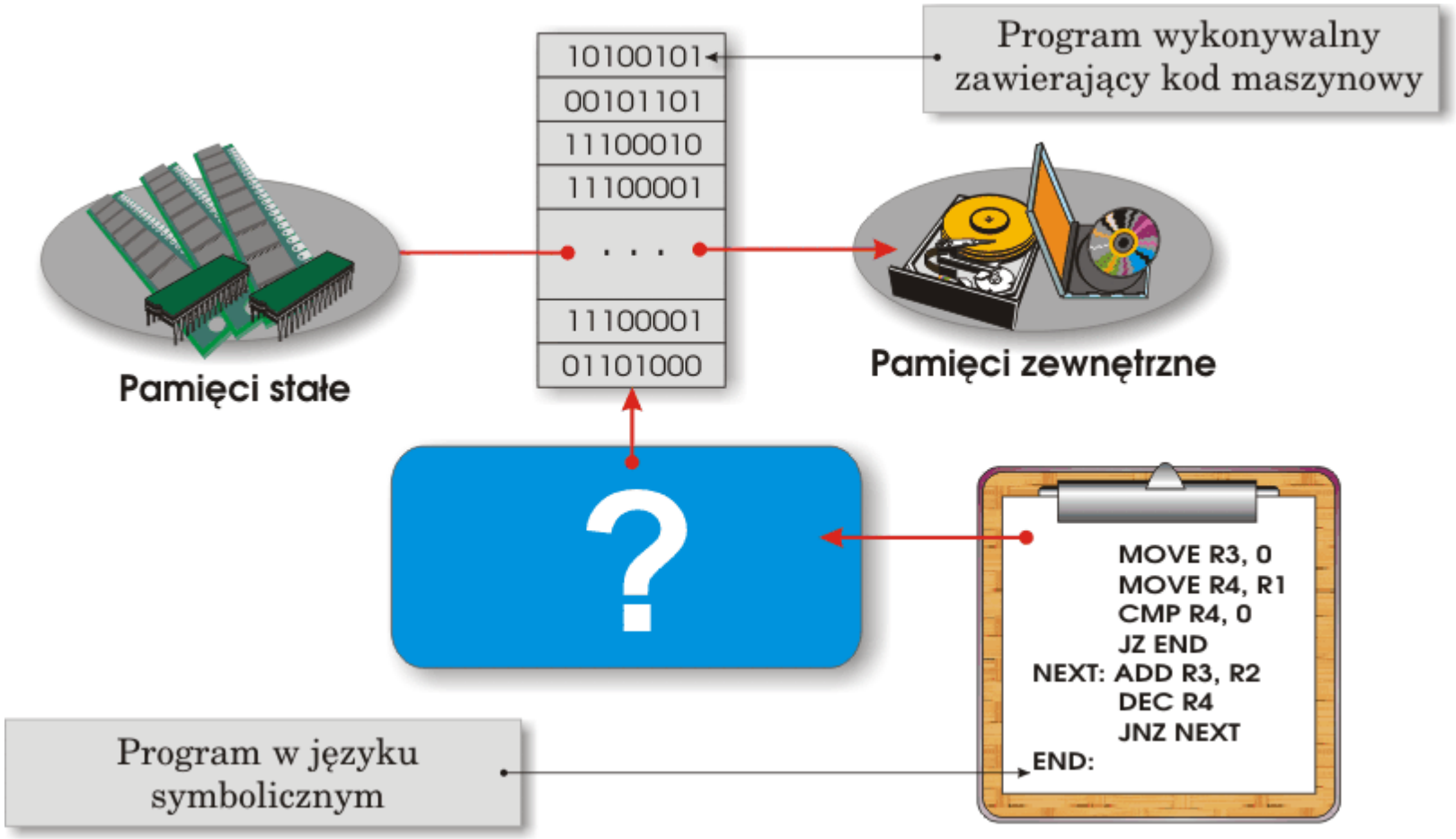
Copyright © Roman Simiński

Od kodu maszynowego do języka symbolicznego

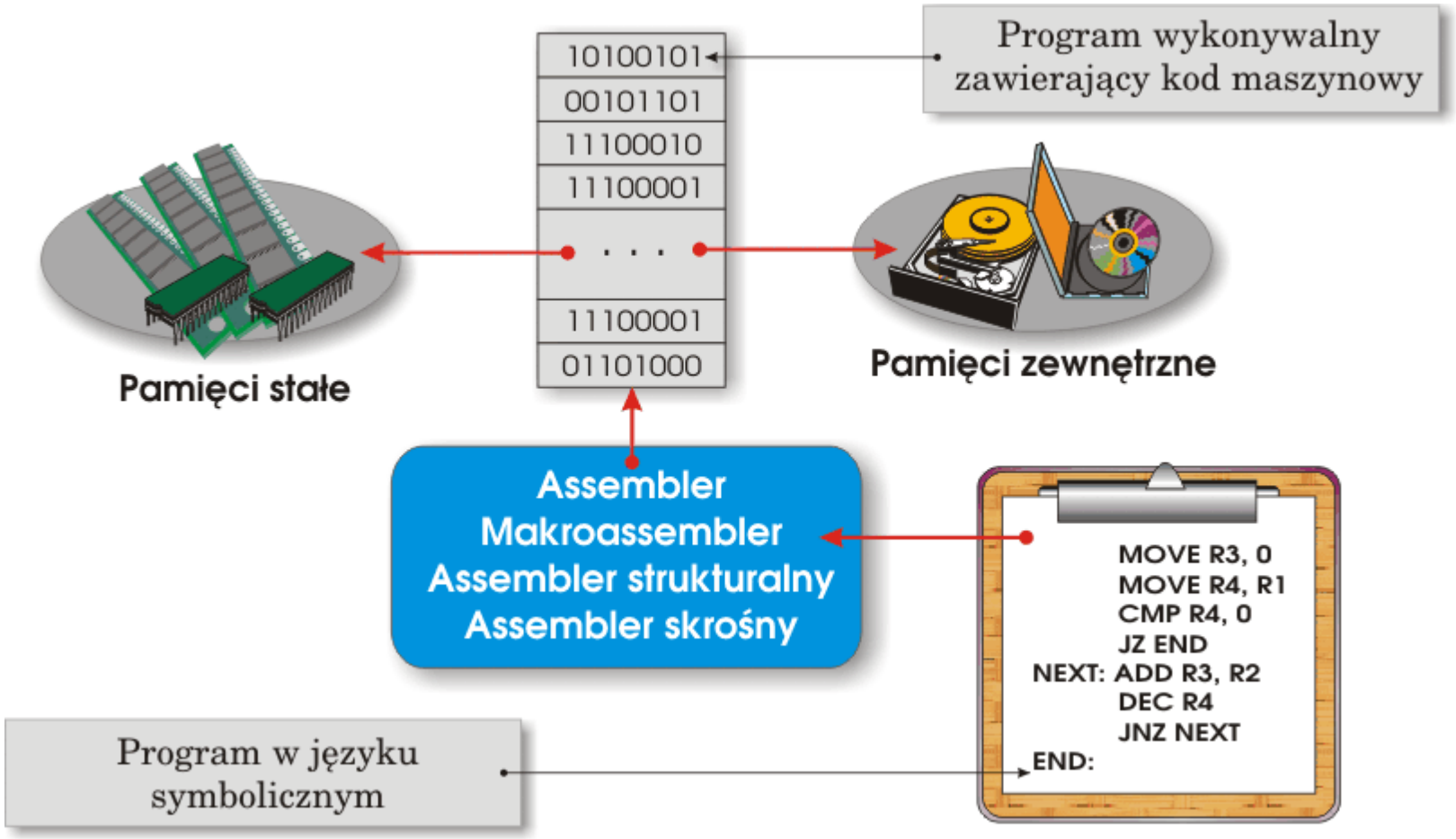


Copyright © Roman Simiński

Od kodu maszynowego do języka symbolicznego



Od kodu maszynowego do języka symbolicznego

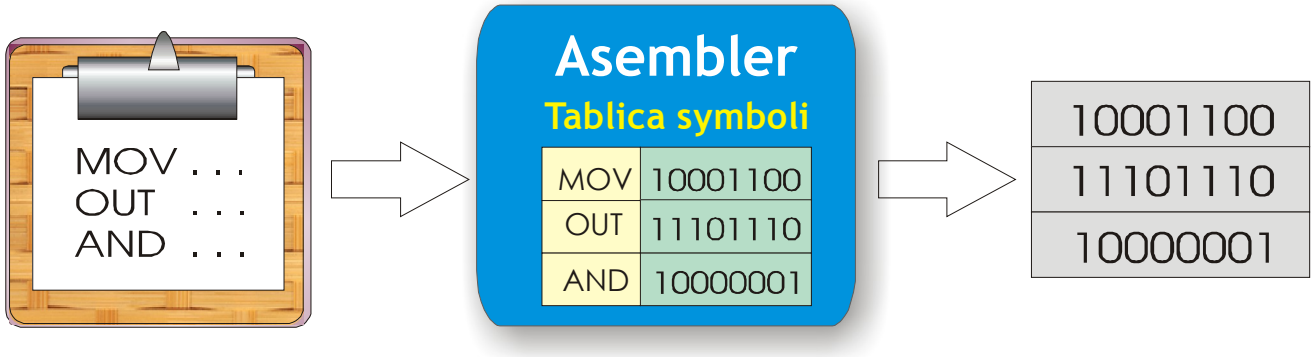


Język symboliczny i asembler — kilka podstawowych informacji

Język symboliczny (język asemblera) umożliwia pisanie programów z wykorzystaniem symboli (mnemoników) przypisanych poszczególnym rozkazom procesora. Program w języku symbolicznym jest zrozumiały dla programisty, lecz nie jest zrozumiały dla procesora. *Mnemonik* to symboliczne oznaczenia rozkazu i innych elementów, np. rejestrów.

Znaczenie rozkazu	Kod binarny	Mnemonik
Przesłanie bajtu pamięc—rejestr	10001100	MOV
Wyprowadzenie bajtu do układu wyjściowego	11101110	OUT
Mnożenie logiczne	10000001	AND
. . .	. . .	. . .

Asembler to program dokonujący tłumaczenia kodu źródłowego programu w języku symbolicznym na kod wynikowy (kod maszynowy).



Copyright © Roman Simiński

Języki wysokiego poziomu a kod maszynowy

Język wysokiego poziomu — niezależny od platformy sprzętowej i systemowej język programowania, pozwalający programiście skoncentrować się na logice rozwiązywanego problemu.

Notacja typowych języków wysokiego poziomu to połączenie elementów języka angielskiego z notacją wywodzącą się z matematyki.

```
        MOVE R3, 0
        MOVE R4, R1
        CMP R4, 0
        JZ END
NEXT:   ADD R3, R2
        DEC R4
        JNZ NEXT
END:
```

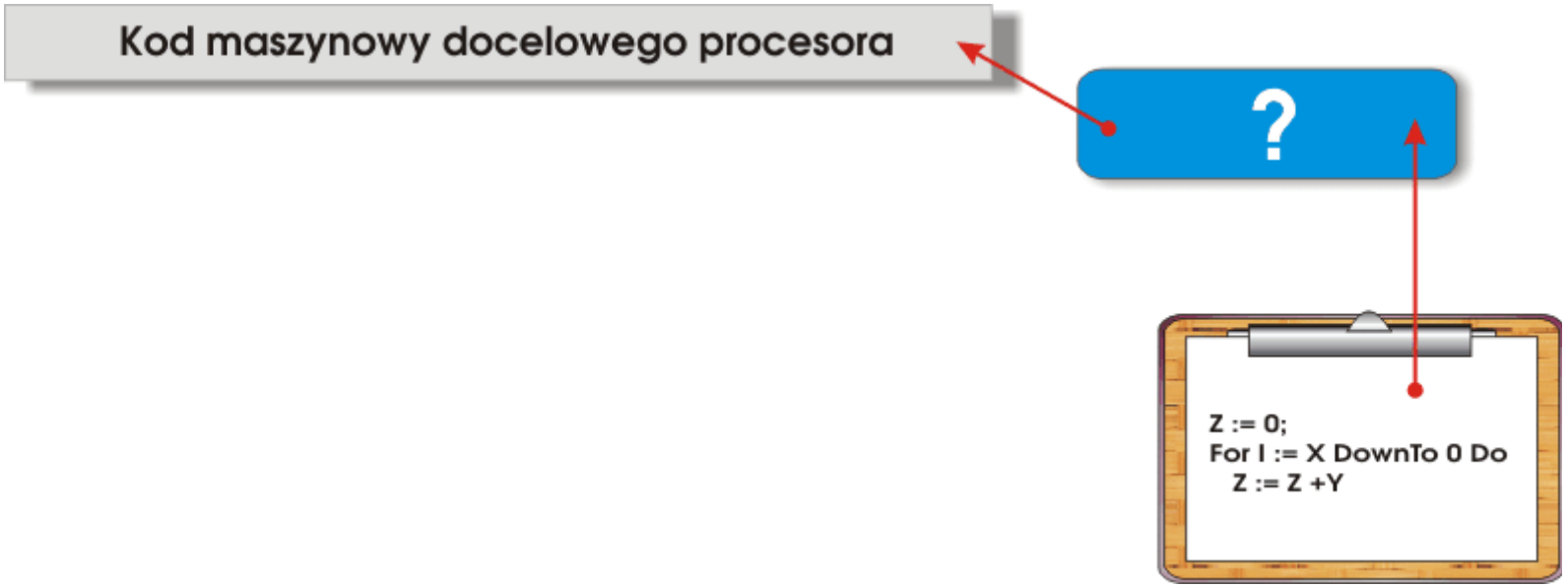
Program w języku symbolicznym

```
Z := 0;
For I := X DownTo 0 Do
  Z := Z +Y
```

Program w języku wysokiego poziomu

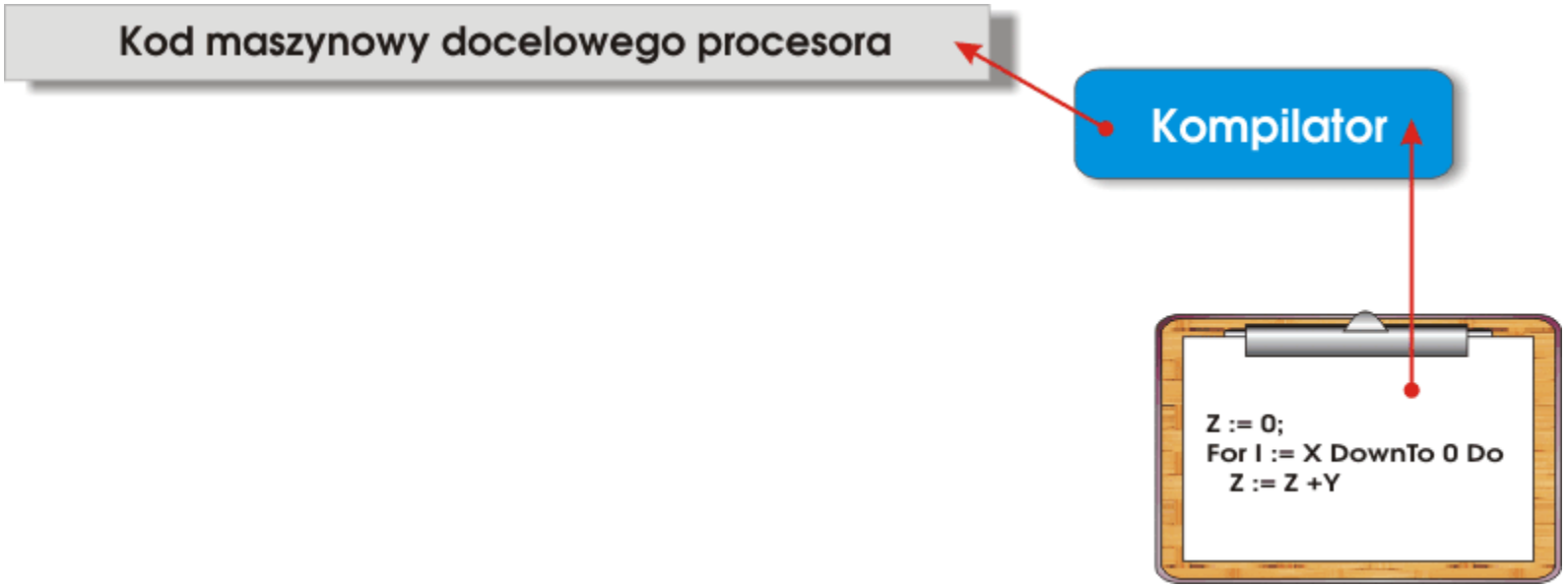
Copyright © Roman Simiński

Języki wysokiego poziomu a kod maszynowy



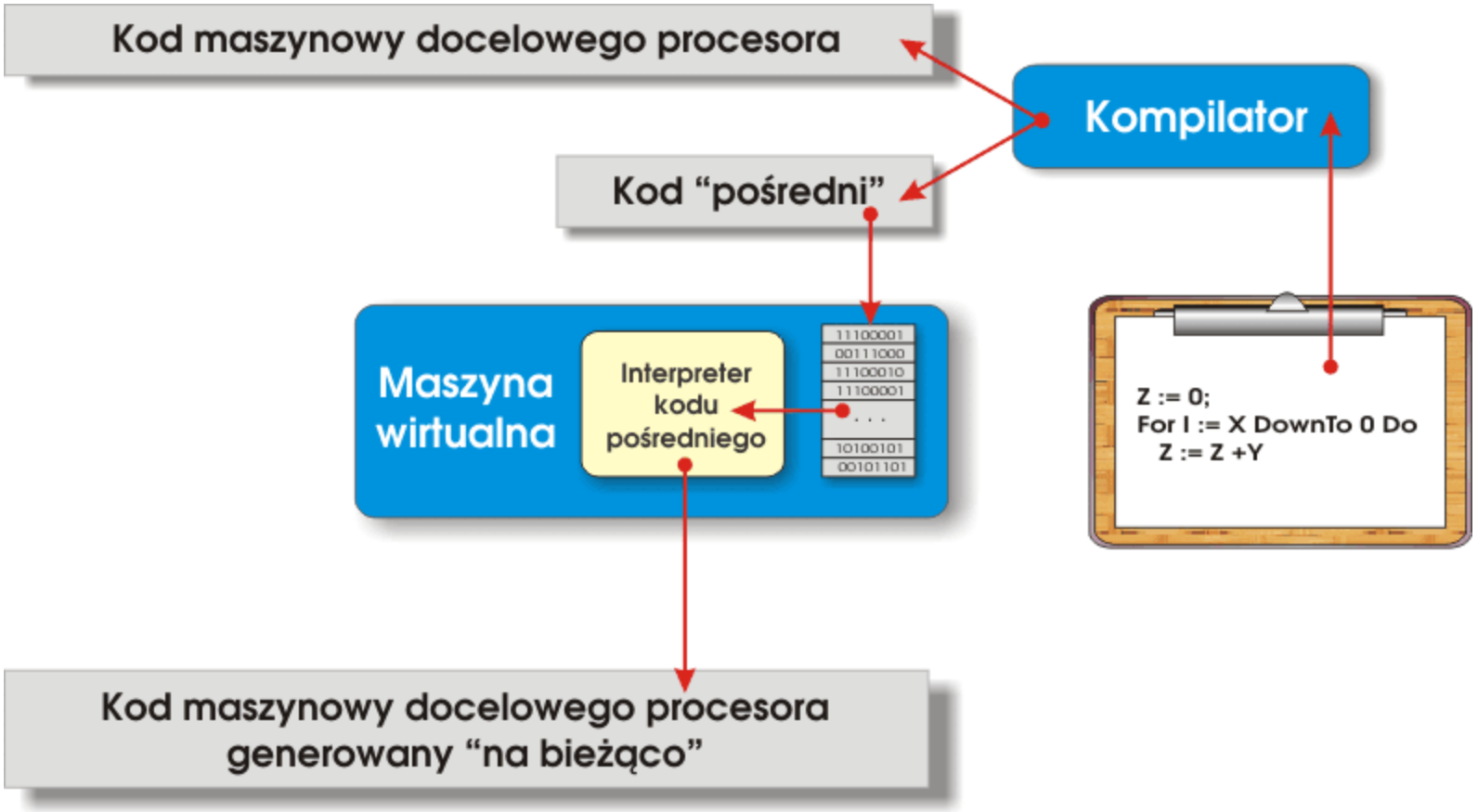
Copyright © Roman Simiński

Języki wysokiego poziomu a kod maszynowy

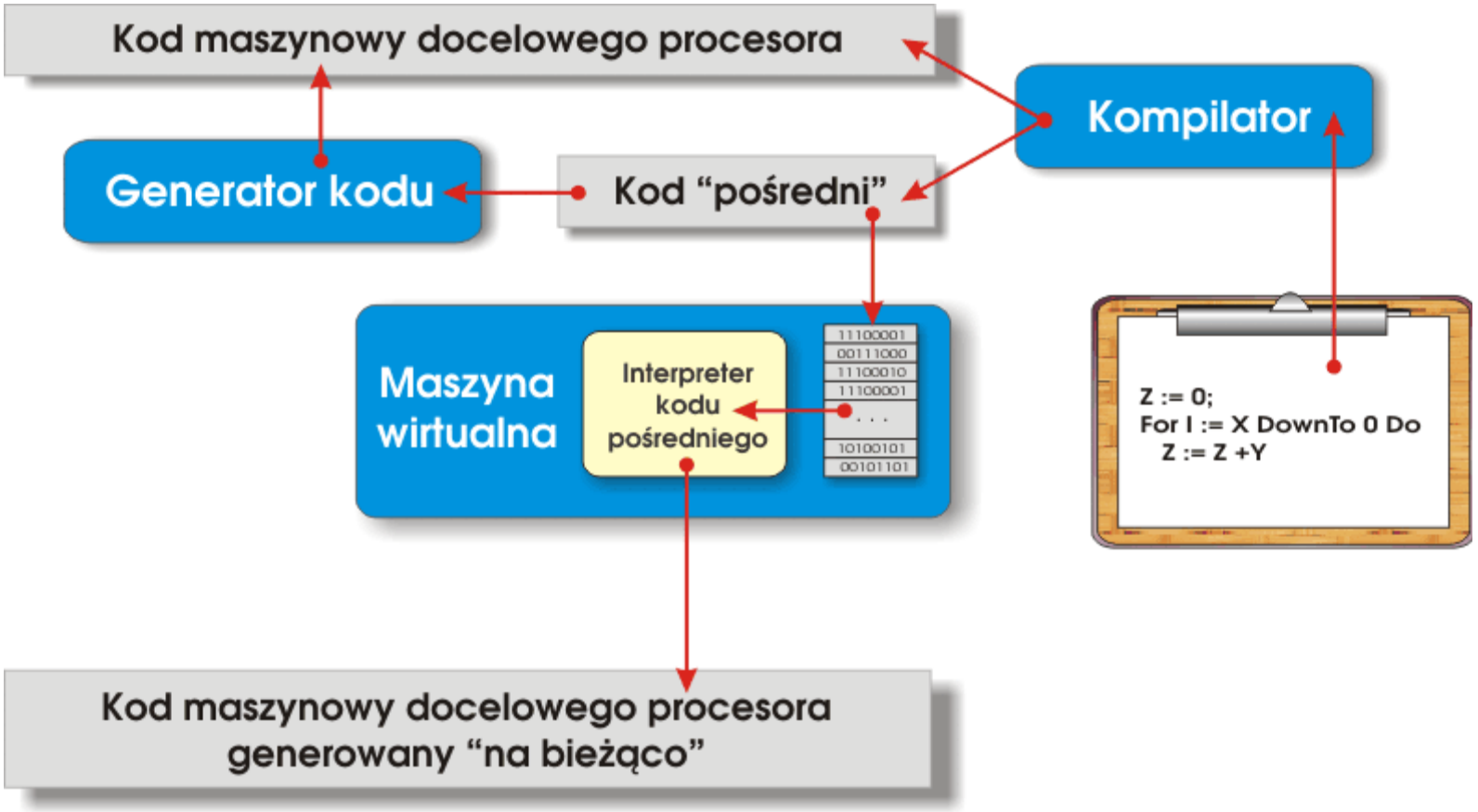


Copyright © Roman Simiński

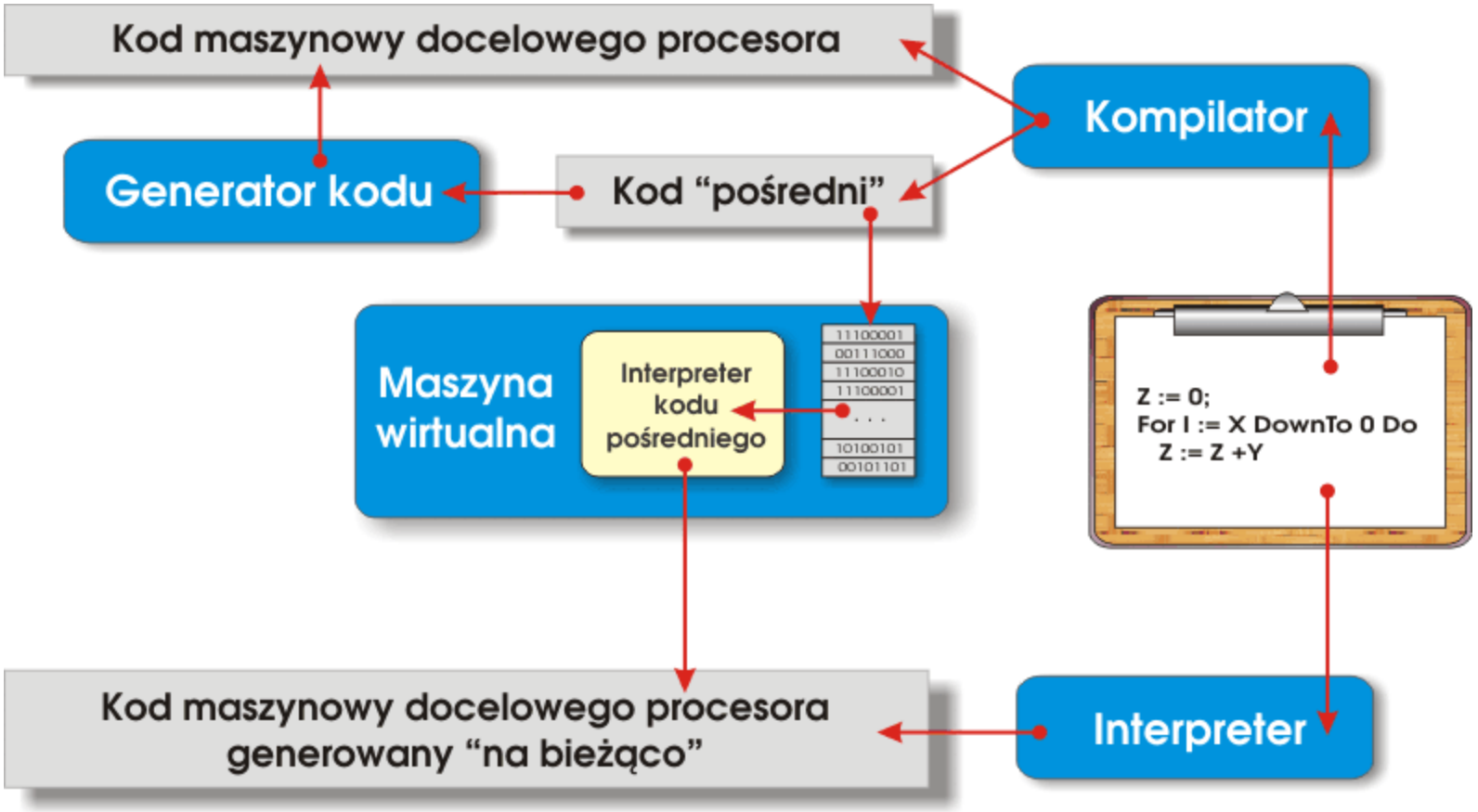
Języki wysokiego poziomu a kod maszynowy



Języki wysokiego poziomu a kod maszynowy

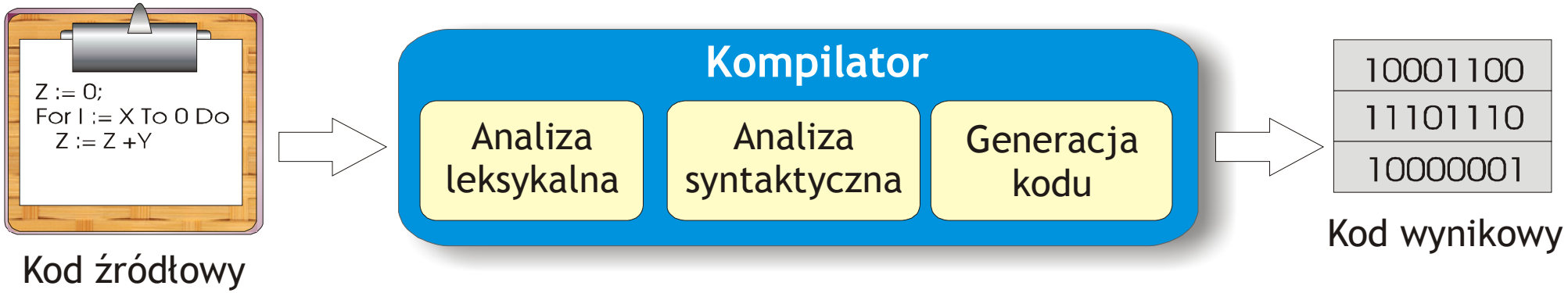


Języki wysokiego poziomu a kod maszynowy



Języki wysokiego poziomu a kod maszynowy

Kompilator to program dokonujący tłumaczenia programu napisanego w języku wysokiego poziomu na kod maszynowy konkretnego, rzeczywistego lub wirtualnego procesora. Zatem kompilator zamienia kod źródłowy na kod maszynowy.



Copyright © Roman Simiński

Kilka ważnych pojęć

Kod źródłowy zawiera program zapisany w danym języku programowania. Kod źródłowy jest najczęściej plikiem tekstowym.

Kod wynikowy zawiera program w postaci kodu maszynowego. Kod wynikowy jest najczęściej plikiem o ustalonej strukturze wewnętrznej i rozszerzeniu nazwy.

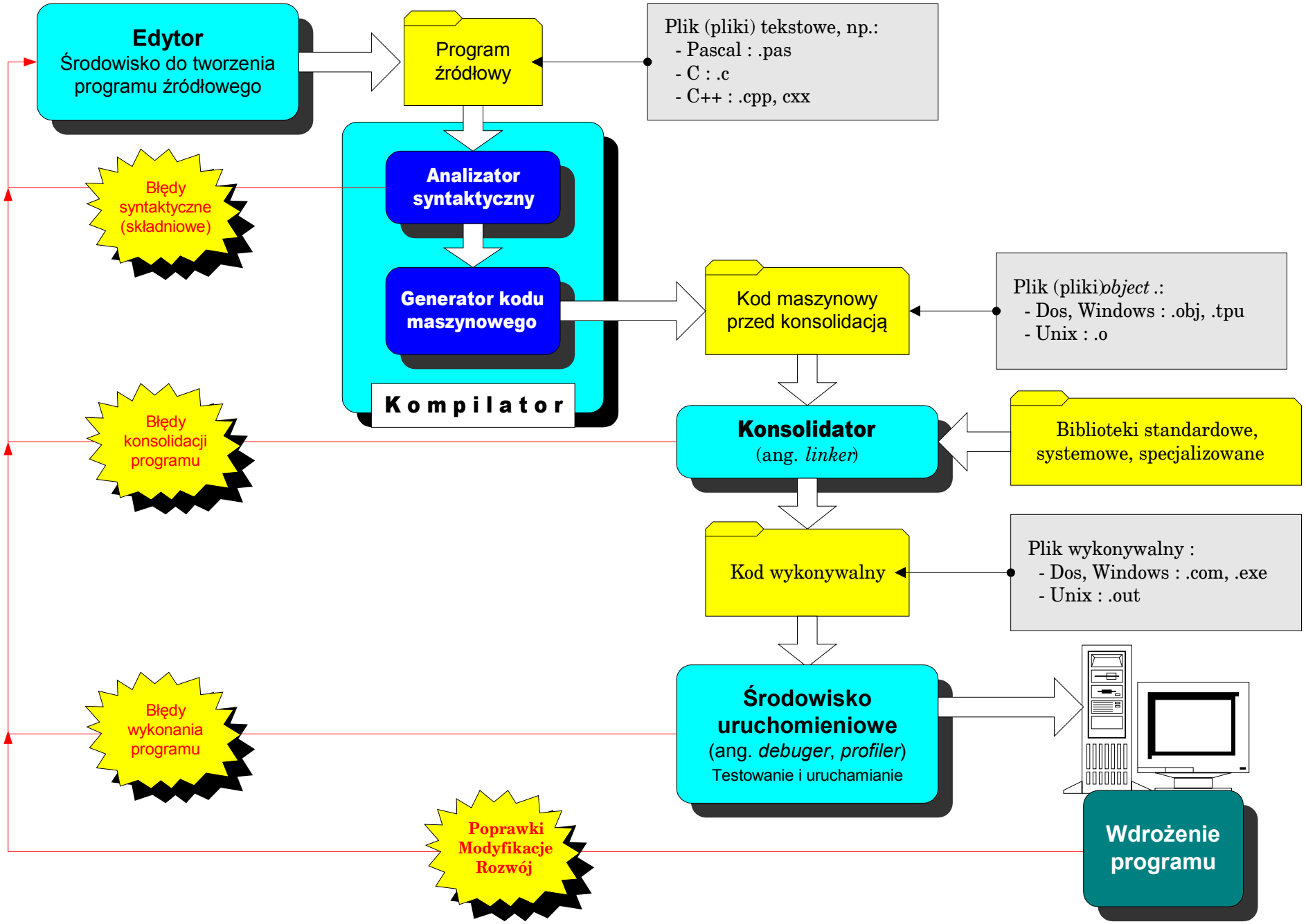
Maszyna wirtualna to program emulujący działanie osobnego komputera (procesora). Jest to samodzielne środowisko wykonawcze, które zachowuje się jak osobny komputer, pozwala na wykonanie kodu dedykowanego dla wirtualnego procesora na procesorze rzeczywistym.

Konsolidator (ang. *linker*) to program łączący w jedną całość wszystkie, uprzednio skompilowane, elementy programu tworząc odpowiedni kod wynikowy.

Debuger (odpluskwiacz) program umożliwiający kontrolowane uruchamianie programu, w trakcie którego można śledzić jego przebieg, kontrolować i zmieniać zawartość zmiennych itp. Podstawowe narzędzie w walce z błędami wykonania, zwanymi pluskwami (ang. *bug*).

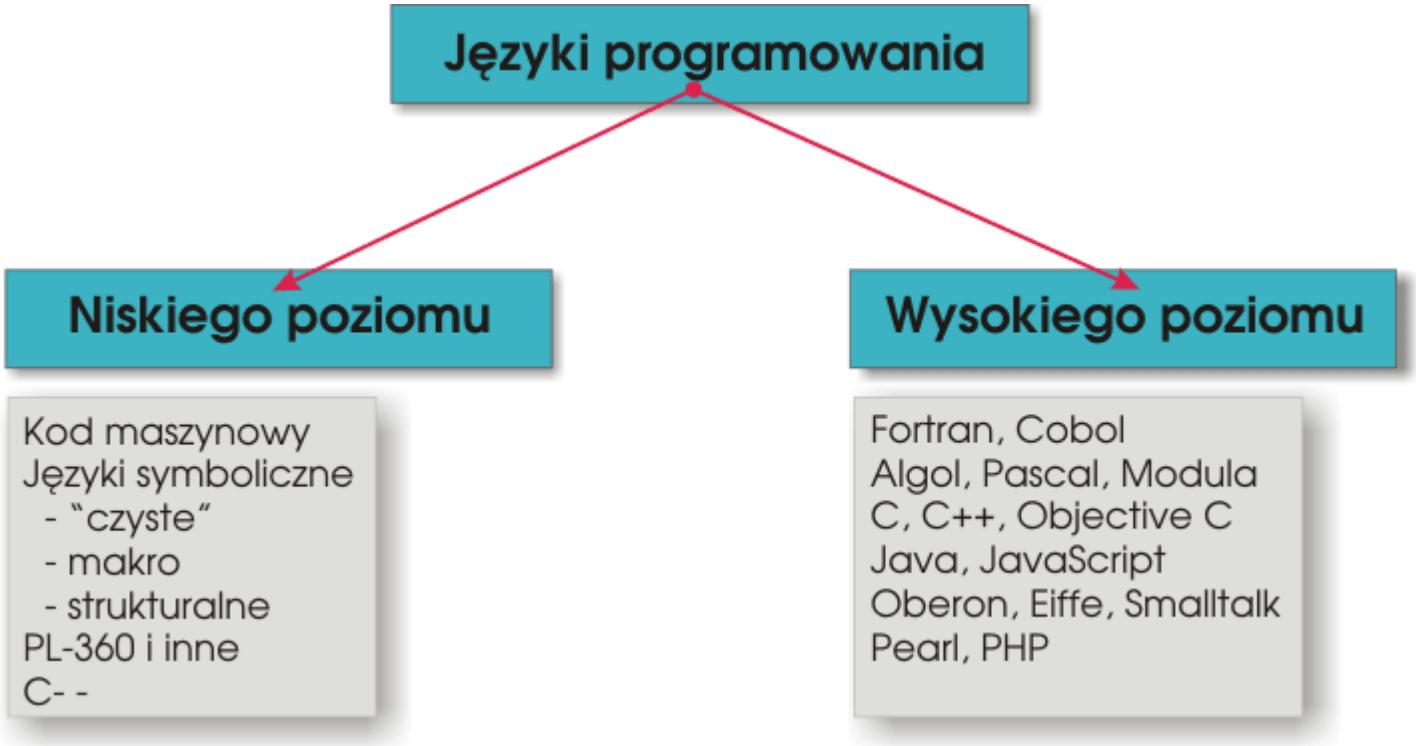
Copyright © Roman Simiński

Przebieg prac programistycznych i wykorzystywane narzędzia



Copyright © Roman Simiński

Podział języków programowania



Copyright © Roman Simiński

Języki programowania wysokiego poziomu

Zalety:

- duża niezależność od platformy sprzętowo-sytemowej,
- elastyczność i łatwość programowania,
- wielość technik, metod i paradygmatów programowania,
- wsparcie dla budowania dużych i złożonych programów.

Wady:

- czasami wolny i duży kod wynikowy,
- kłopotliwe programowanie fragmentów “bliskich sprzętu”,
- zależność od kompilatora, bibliotek i narzędzi wspomagających.

Język wysokiego poziomu pozwala programiście skupić się na rozwiązywaniu potencjalnie złożonego problemu, a nie na kodowaniu.

Copyright © Roman Simiński

Języki programowania niskiego poziomu

Zalety:

- szybki i relatywnie nieduży kod wynikowy,
- dobre i efektywne wykorzystanie sprzętu,
- implementacja programów dedykowanych i specyficznych

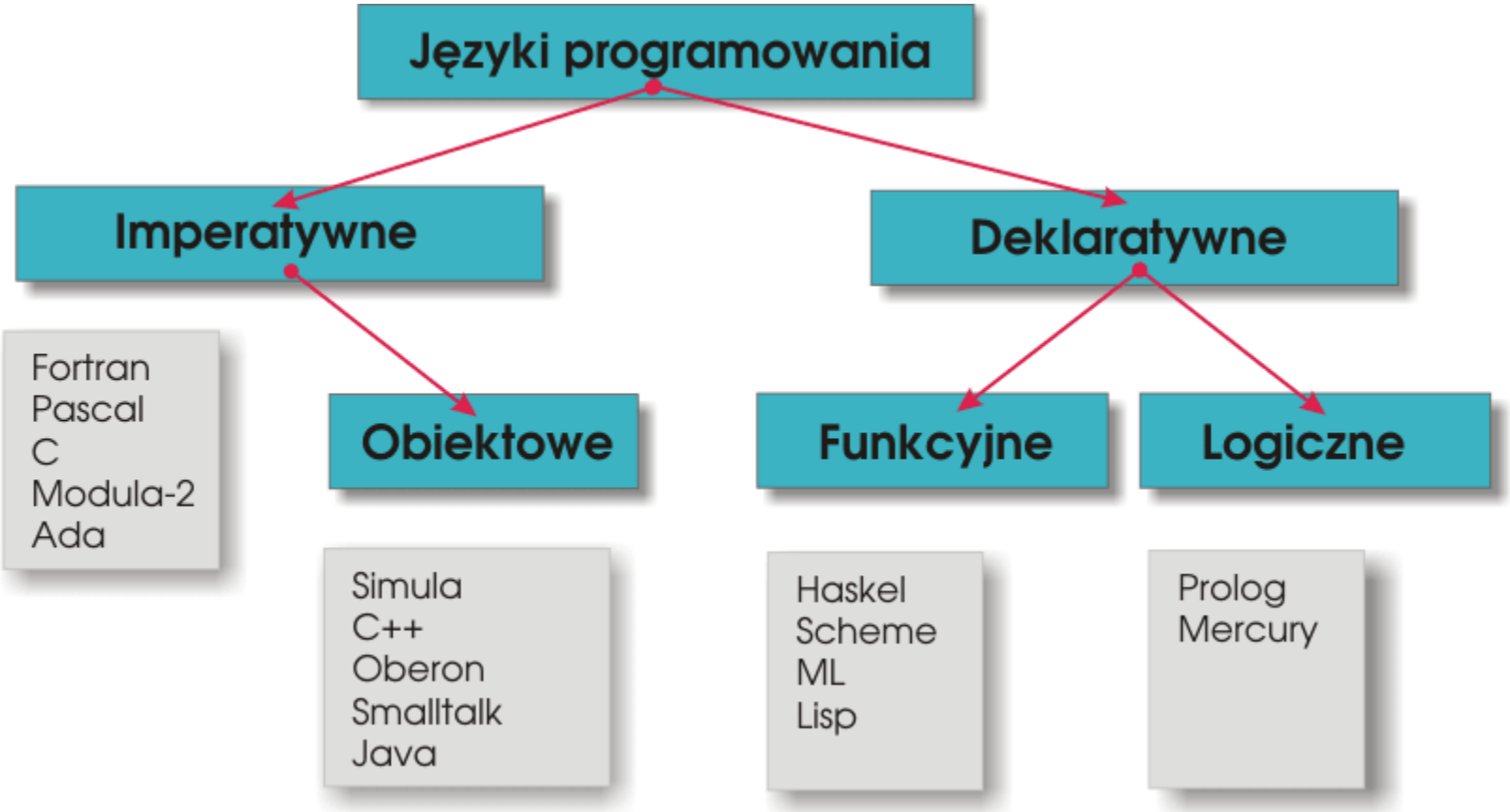
Wady:

- ograniczenie do listy rozkazów danego procesora,
- wymagana znajomość struktury i specyfiki działania procesora i jego otoczenia,
- ręczne zarządzanie danymi, stosem, uciążliwe kodowanie,
- trudne kodowanie rozwiązań problemów złożonych.

Programista myśli kategoriami sprzętu a nie dziedziny problemu, dużo uwagi poświęca na kodowanie, często kosztem myślenia o rozwiązaniu problemu.

Copyright © Roman Simiński

Podział języków wg. metod programowania



Copyright © Roman Simiński

Dwa spojrzenia na historyczny podział języków programowania

Rośnie poziom abstrakcji

1-sza generacja
kodowanie maszynowe

2-ga generacja
języki symboliczne

3-cia generacja
języki wysokiego poziomu
(strukturalne lub nie)

4-ta generacja
jak 3-cia + ukierunkowanie
na dziedzinę problemu

5-ta generacja
języki deklaratywne
języki sztucznej inteligencji

Rozwój technik programowania

1-sza generacja
podprogramy, dane globalne

2-ga generacja
jak 1-sza generacja plus:
- wielopoziomowe podprogramy
- dane lokalne, parametry

3-cia generacja
jak 2-ga generacja plus:
- modularyzacja
- rozbudowana typizacja, obiekty

4-ta generacja
jak 3-cia + ukierunkowanie
na dziedzinę problemu

2007

Copyright © Roman Simiński

Rozwój metod i języków programowania

Metody i języki programowania ciągle się rozwijają.

Rozwój dotyczy różnych aspektów:

- technicznych,
- praktycznych,
- metodycznych,
- a nawet ... filozoficznych.

Programista i projektant systemów informatycznych “skazany” jest na ciągły rozwój i samokształcenie.

Copyright © Roman Simiński

Rozwój metod i języków programowania

Programowanie niestrukturalne

Programowanie strukturalne

Programowanie obiektowe

Programowanie sterowane przepływem

Programowanie sterowane zdarzeniami

Programowanie na poziomie kodu

Programowanie "wizualne" RAD

Generowanie kodu, systemy CASE

Programowanie genetyczne

Programowanie komponentowe

Programowanie w logice

Copyright © Roman Simiński