Jak to się dzieje, że programy działają,
czyli opowieść symbiozie sprzętu i oprogramowania

Roman Simiński
roman.siminski@us.edu.pl

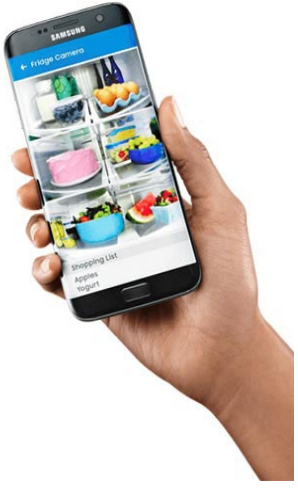
UNIWERSYTET ŚLĄSKI
W KATOWICACH

Wydział Nauk Ścisłych i Technicznych

Komputery „oczywiste”

dreamstime.com

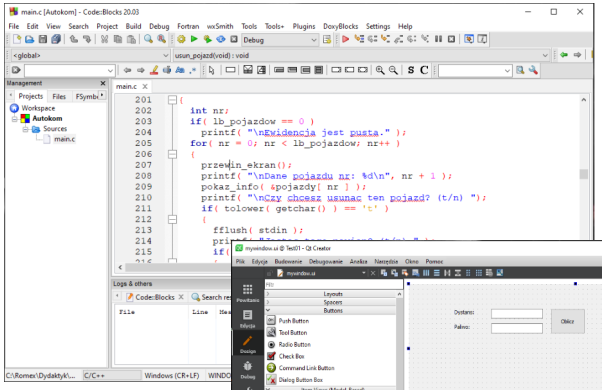
Komputery „nieoczywiste”

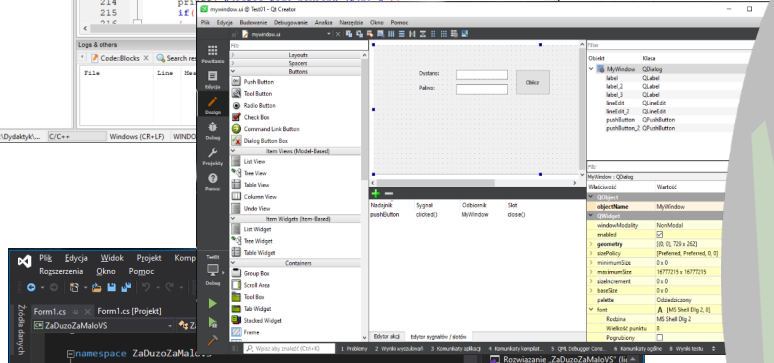


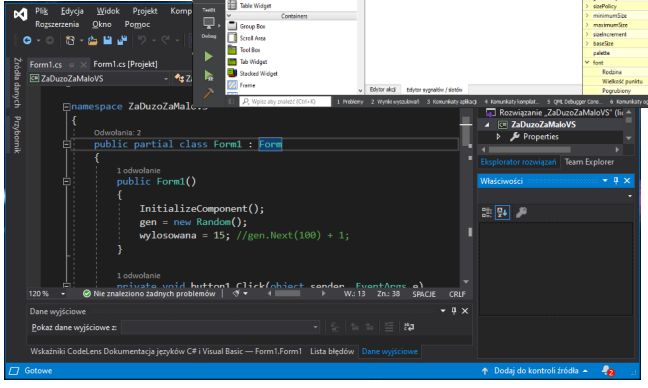
Źródło: www.geniusgadget.com

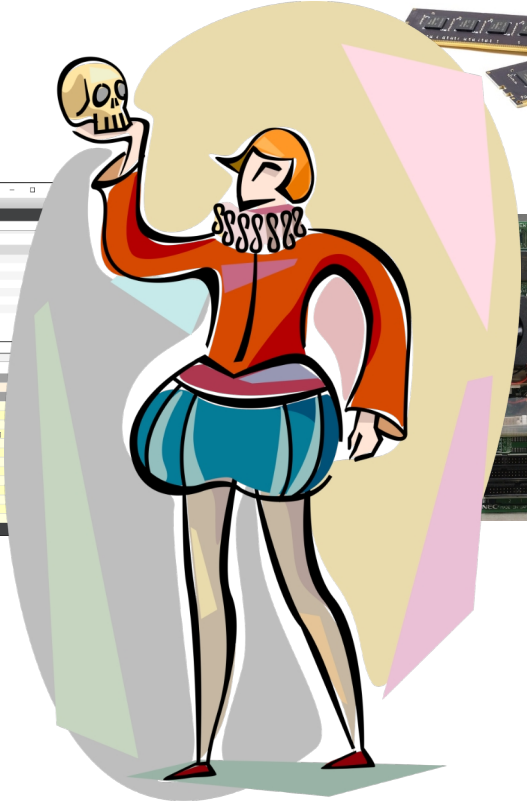
Komputery „wszędzie” - , IoT czyli Internet Rzeczy i Cloud Computing

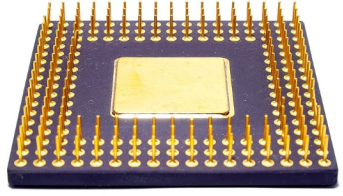
Sprzęt a oprogramowanie, oto jest pytanie...

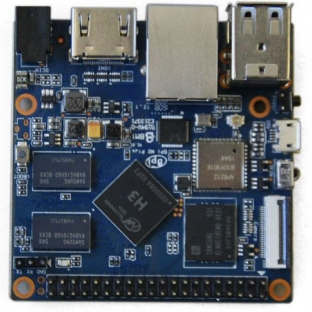










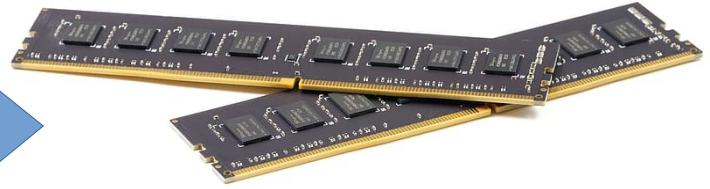


Ze świata analogowego do świata cyfrowego...




Digitalizacja





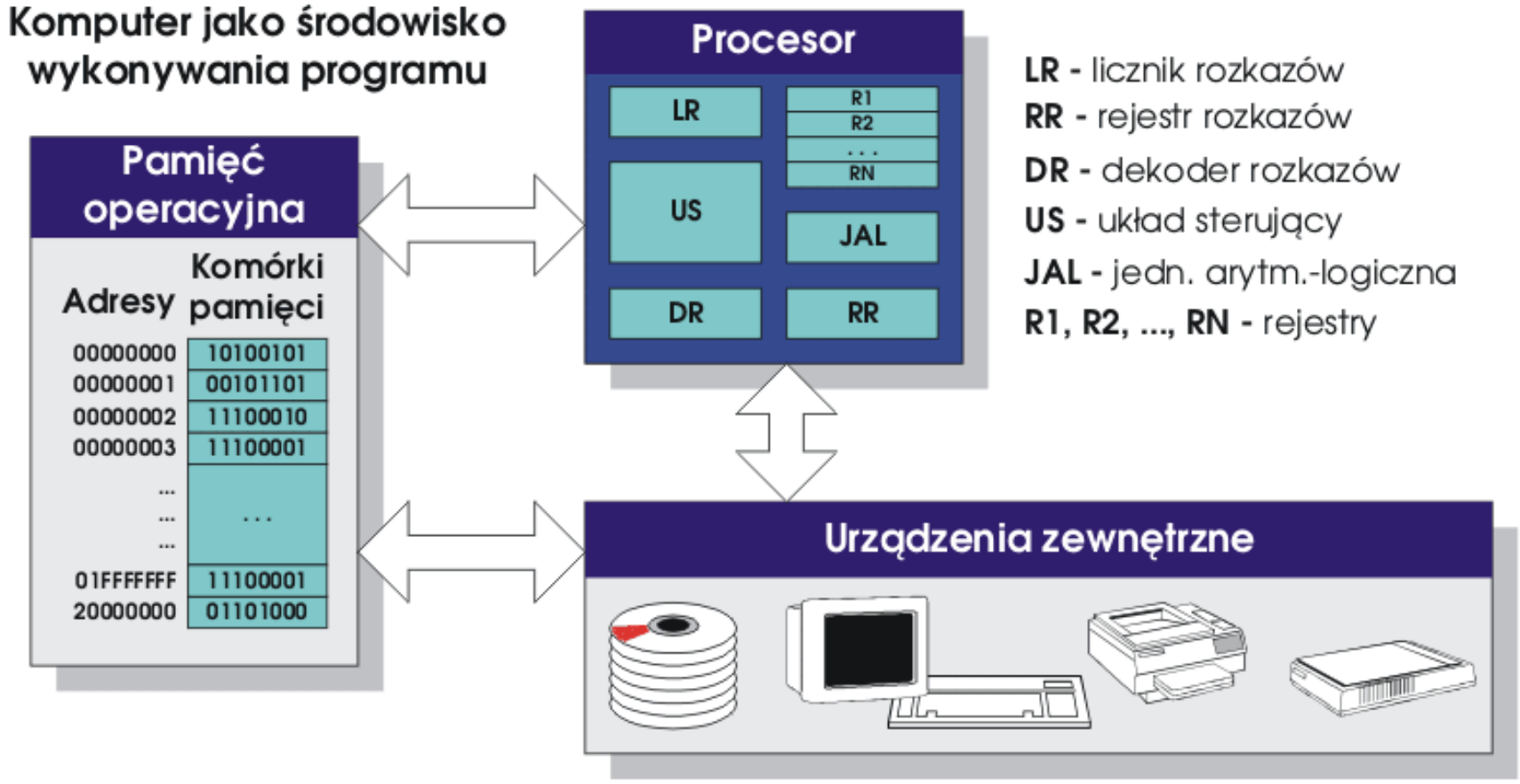
Jak cyfrowo reprezentować rozkazy?

Rozkazy też mogą zostać „zdigitalizowane” poprzez ich ponumerowanie...

Padnij, powstań, padnij powstań

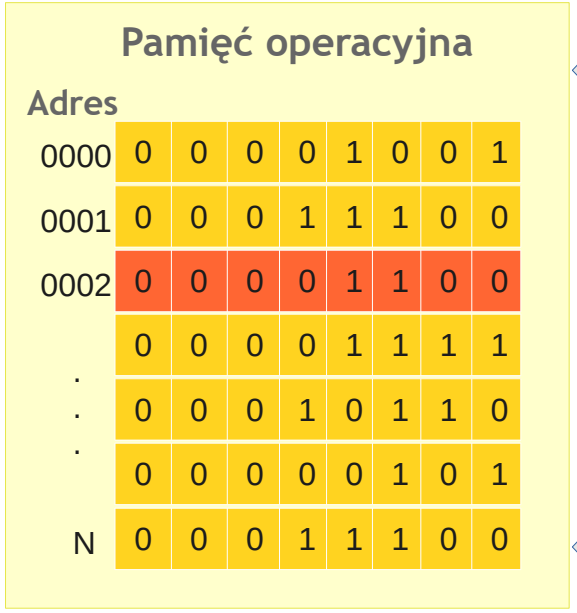
01010 11010 01010 11010

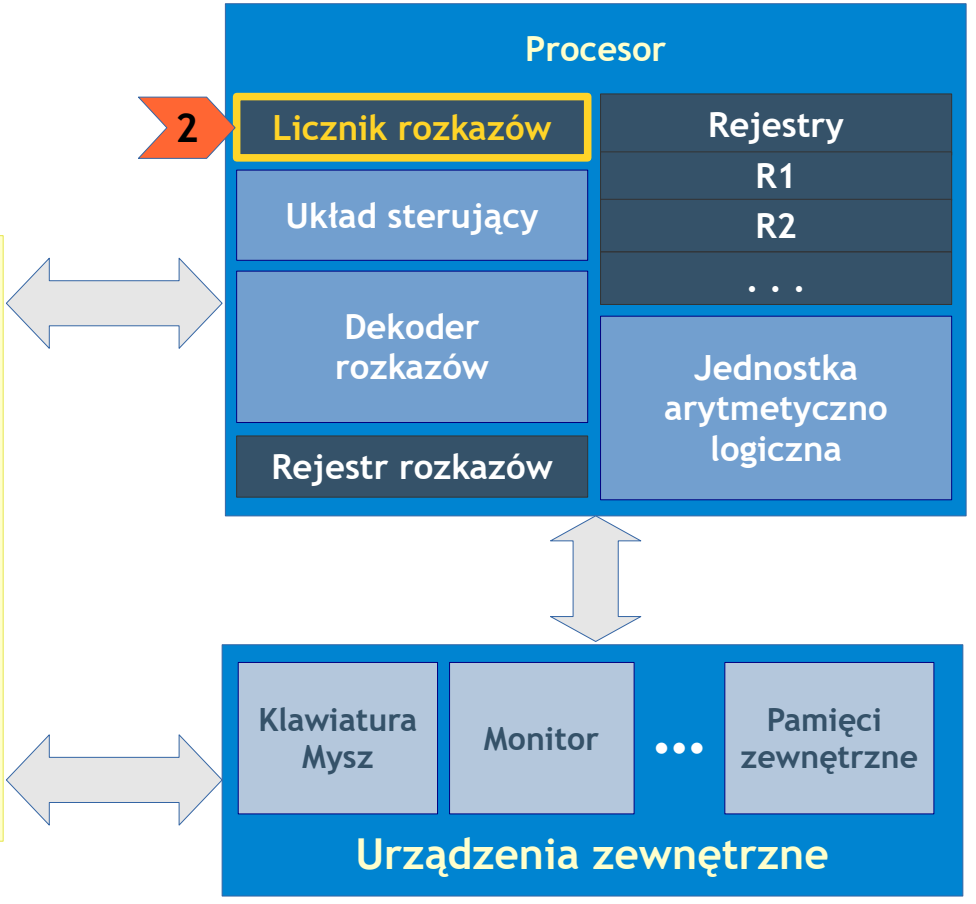
Architektura typowego komputera



Cykl rozkazowy: przygotowanie do wykonania rozkazu

Licznik rozkazów zawiera adres rozkazu do pobrania

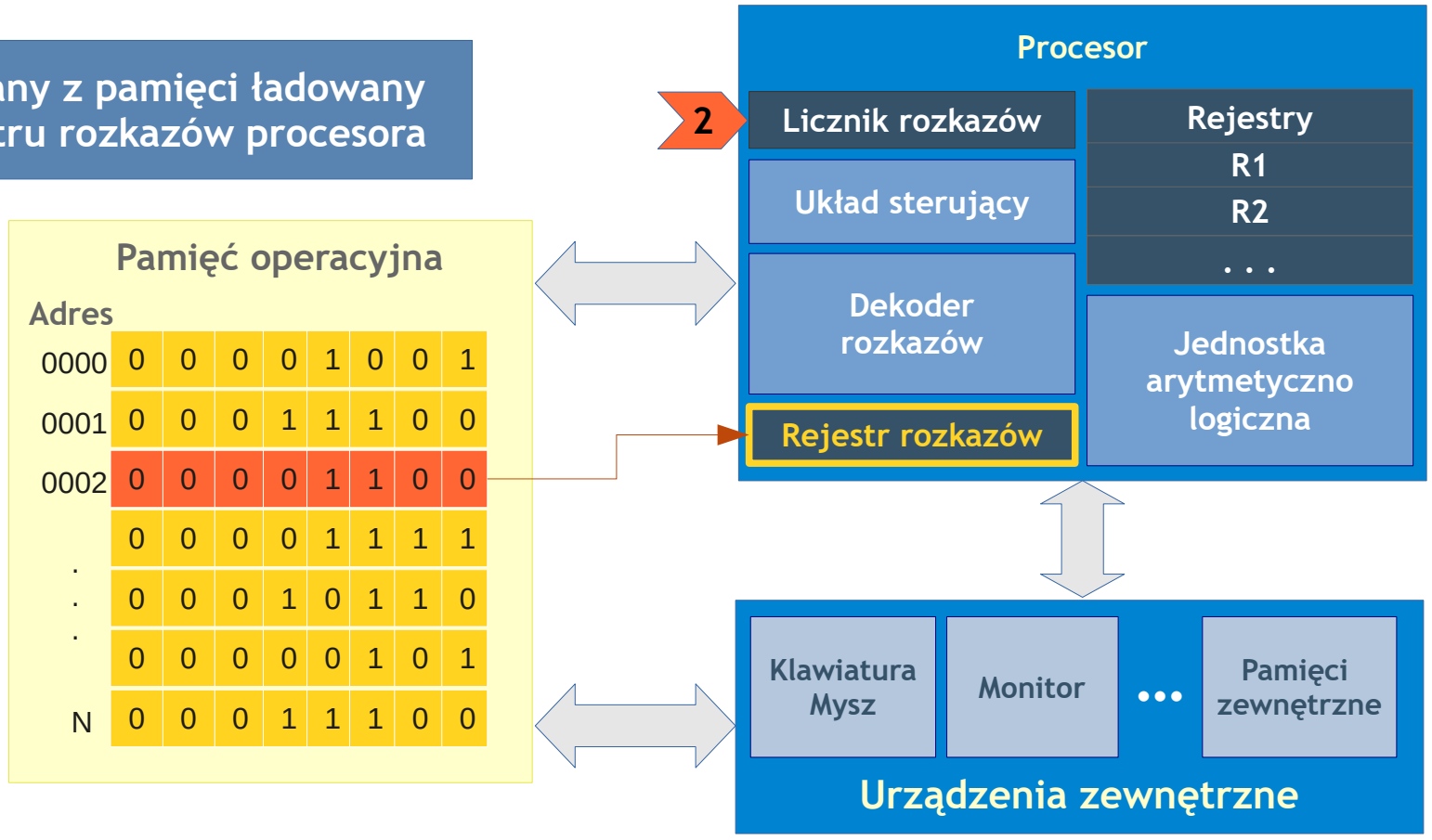




UNIWERSYTET ŚLĄSKI W KATOWICACH

Cykl rozkazowy: pobranie rozkazu z pamięci operacyjnej

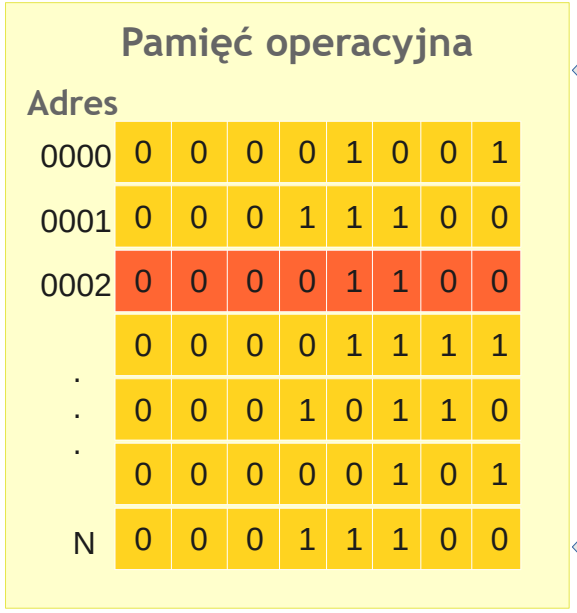
Rozkaz pobrany z pamięci ładowany jest do rejestru rozkazów procesora

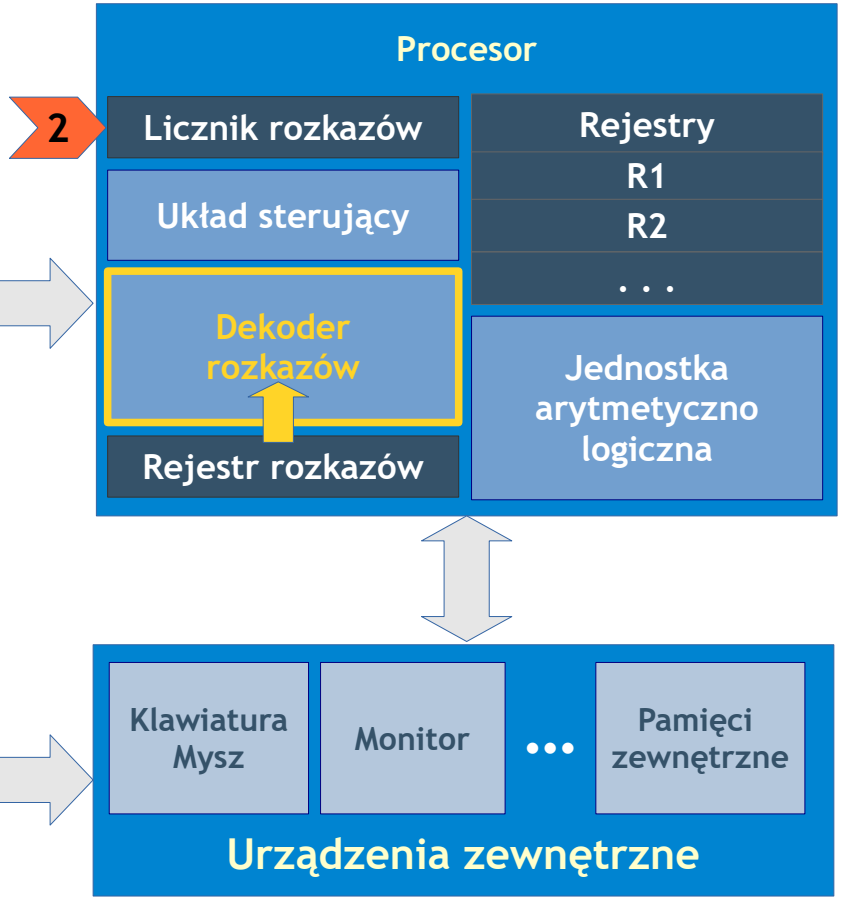


UNIWERSYTET ŚLĄSKI
W KATOWICACH

Cykl rozkazowy: zdekodowanie rozkazu

Dekoder rozkazów zna każdy rozkaz i potrafi go rozpoznać

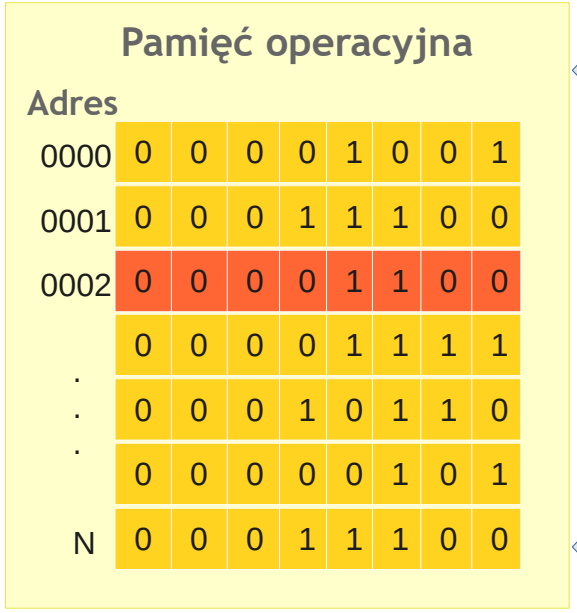


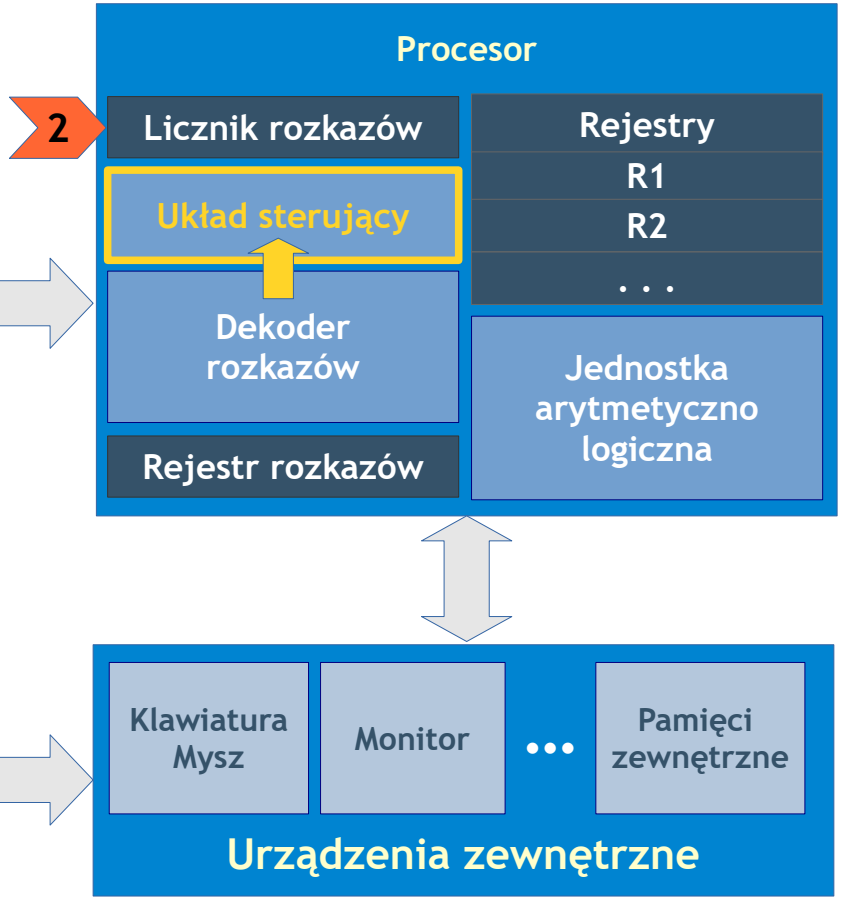


UNIWERSYTET ŚLĄSKI
W KATOWICACH

Cykl rozkazowy: wykonanie rozkazu

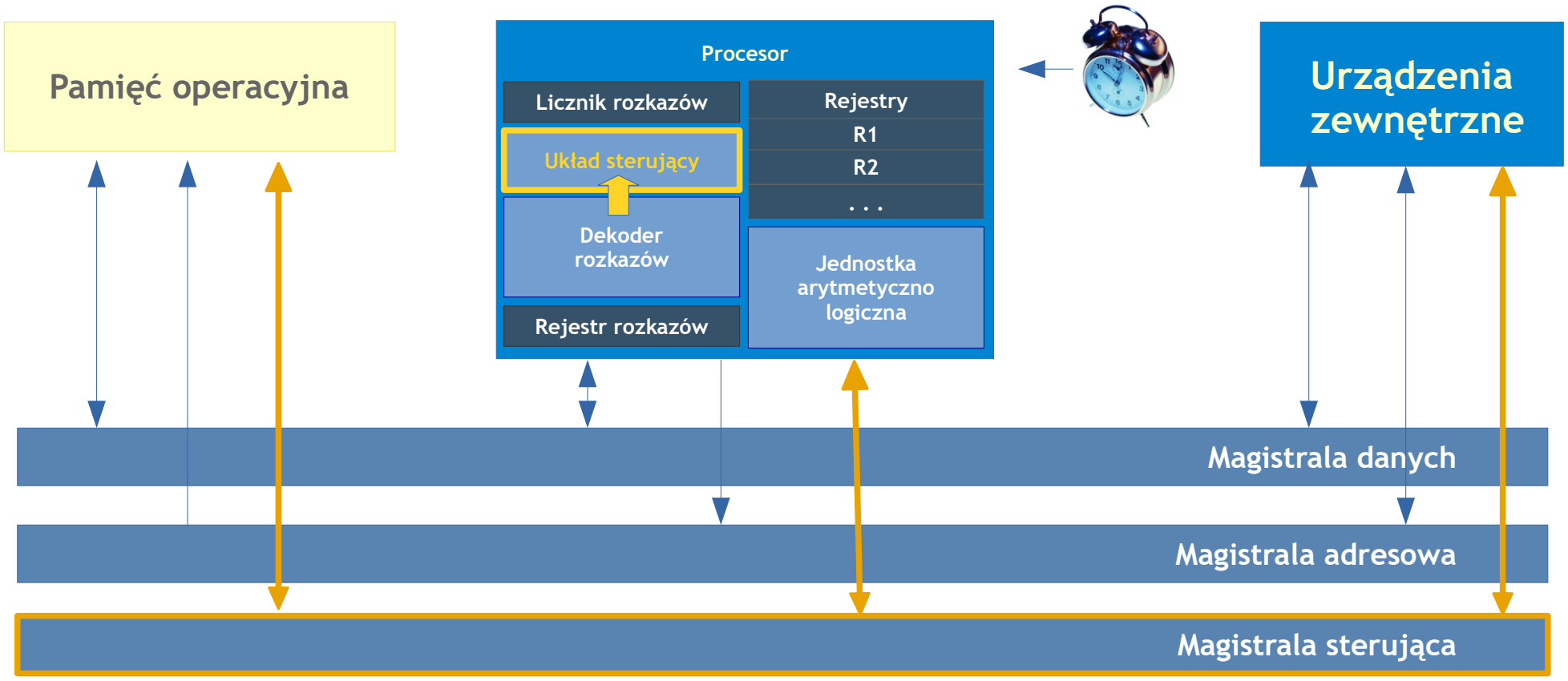
Układ sterujący za przepis na wykonanie każdego rozkazu





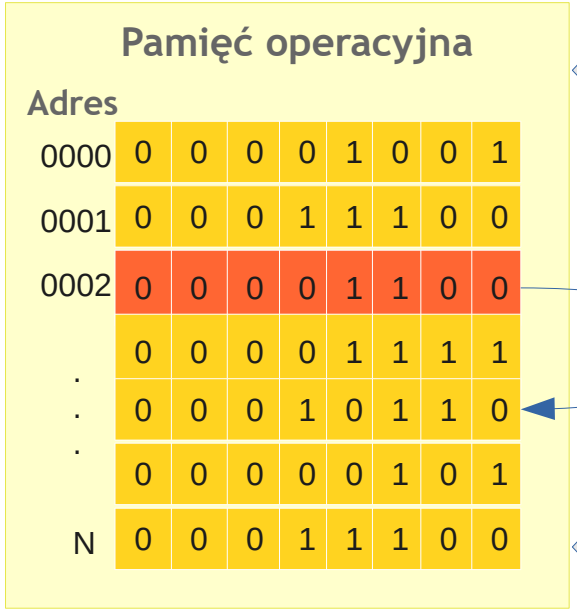
UNIWERSYTET ŚLĄSKI
W KATOWICACH

Wykonanie rozkazu: sekwencja mikrooperacji

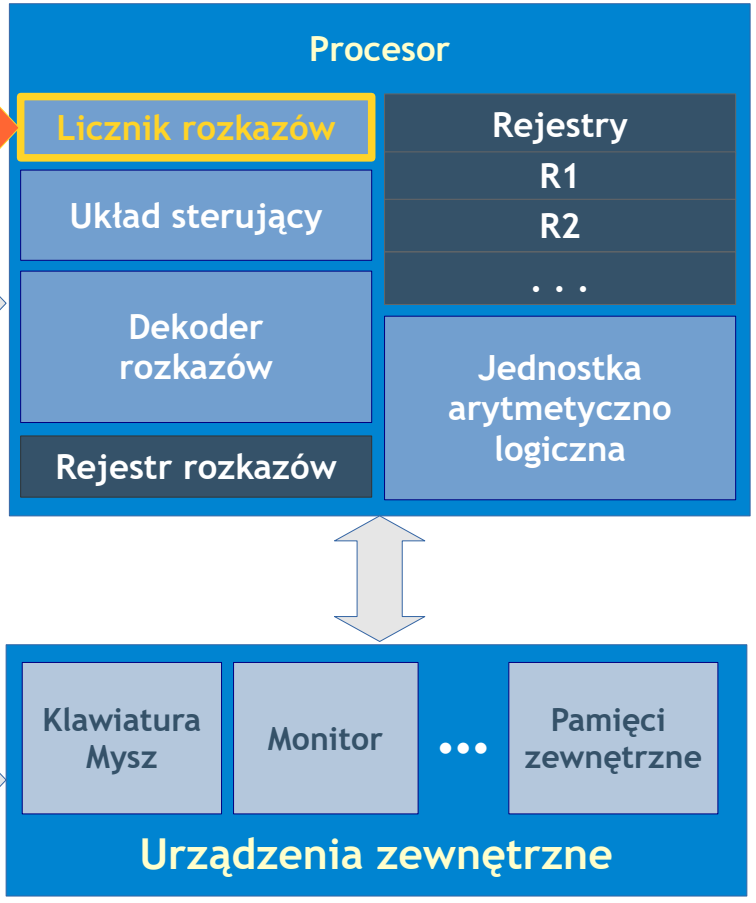


Cykl rozkazowy: ustalenie adresu rozkazu następnego

Trzeba ustalić, skąd w następnym cyklu zostanie pobrany rozkaz



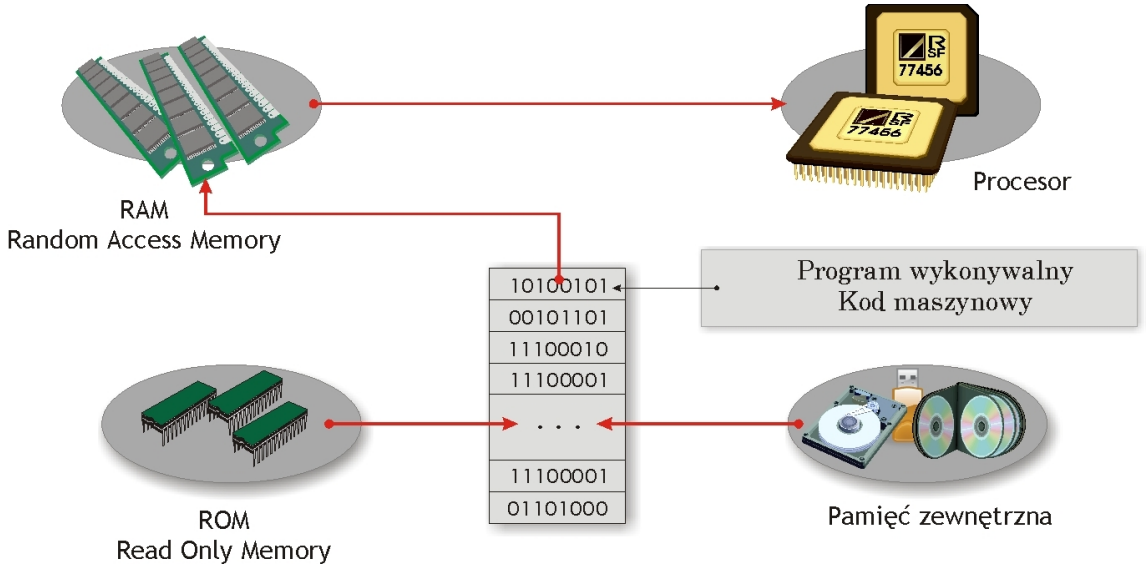
?



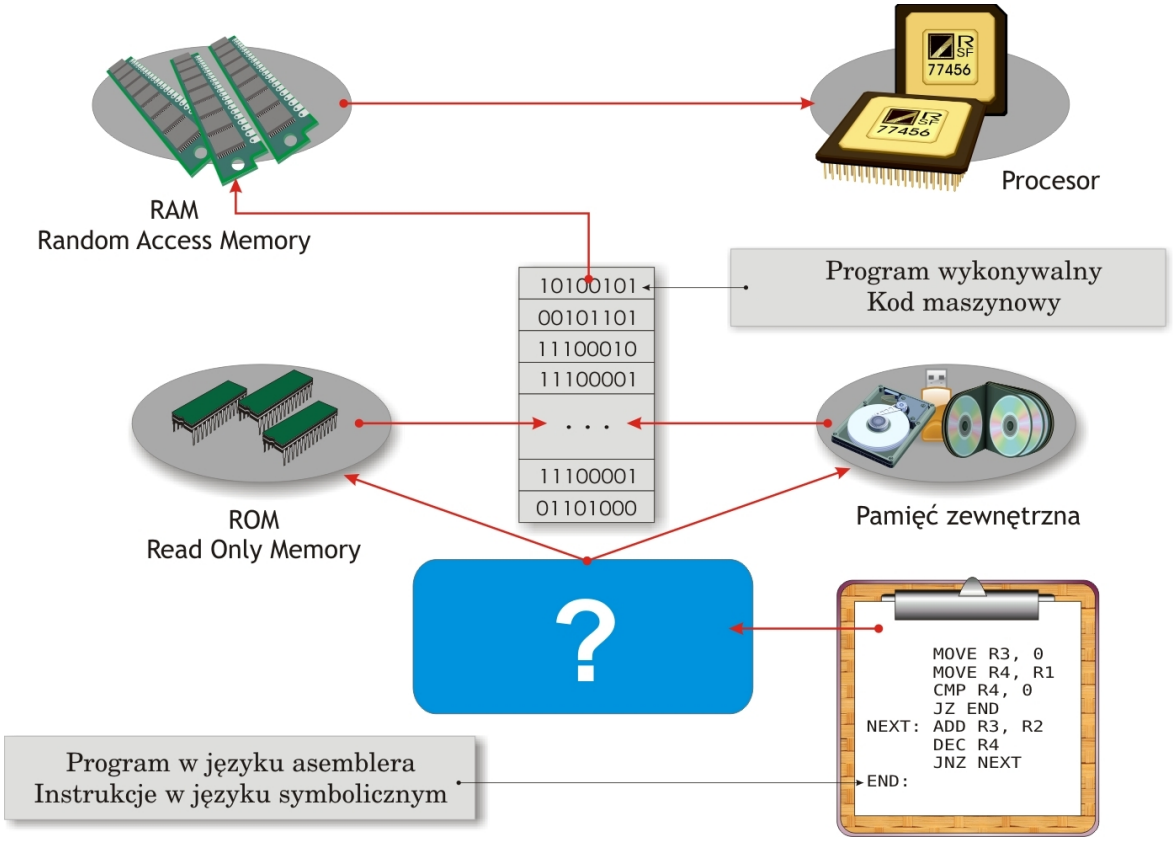
UNIWERSYTET ŚLĄSKI W KATOWICACH

Jak powstaje program?

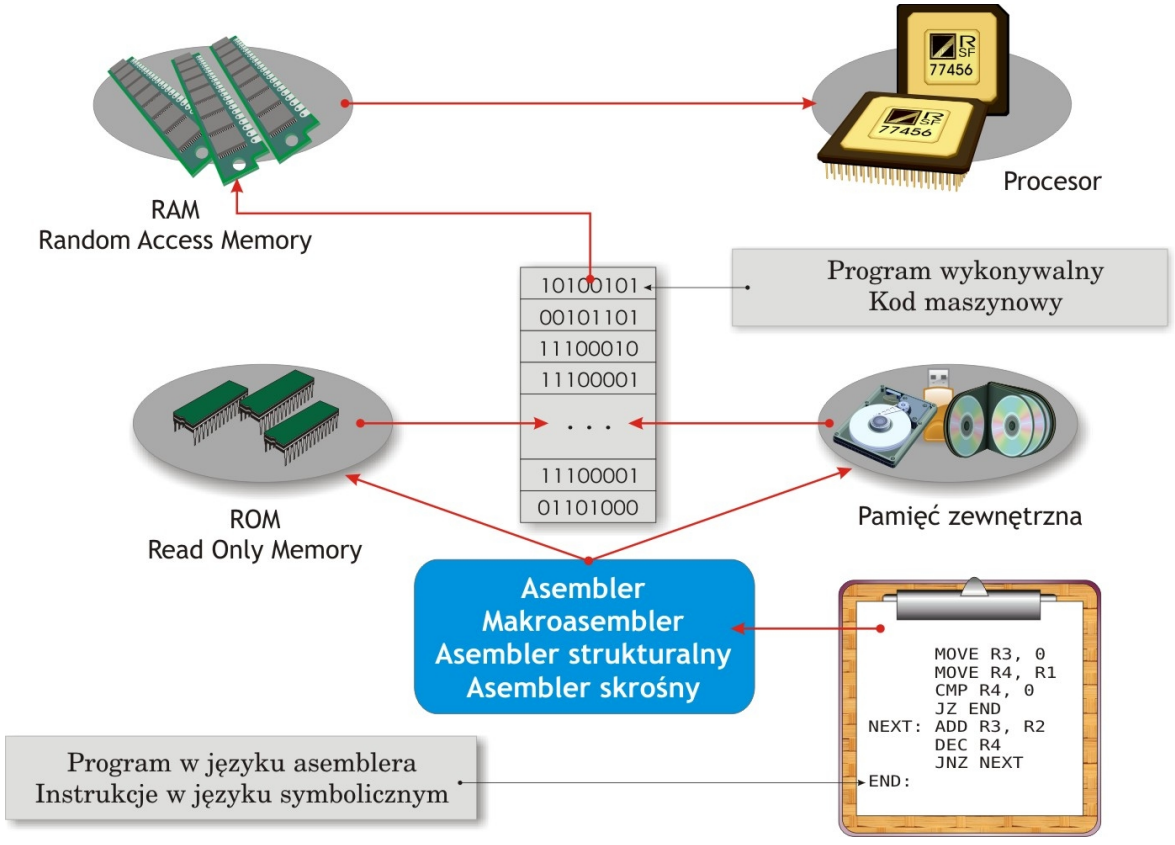
Procesor wykonuje rozkazy maszynowe



Programowanie w języku symbolicznym

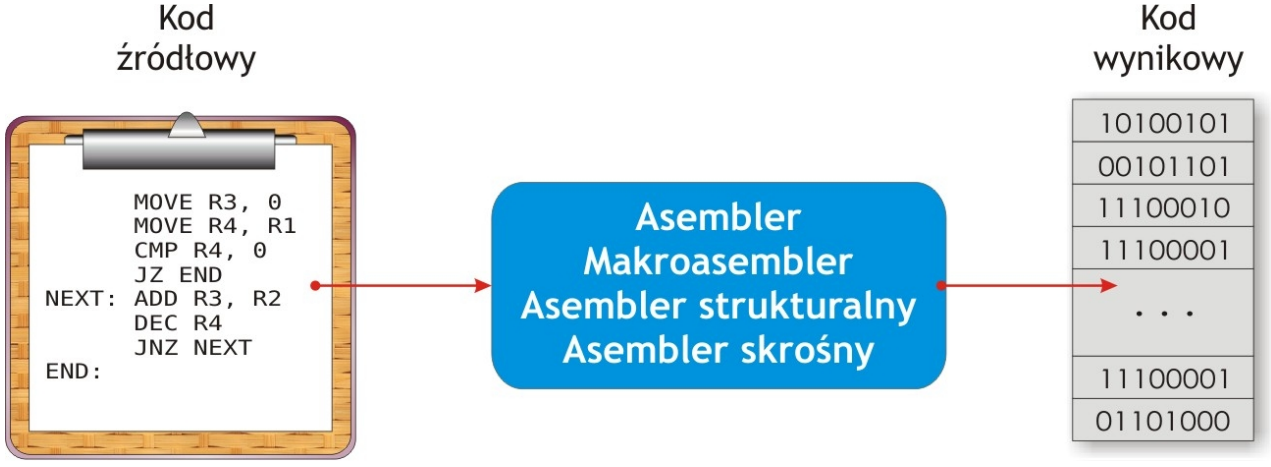


Programowanie w języku symbolicznym



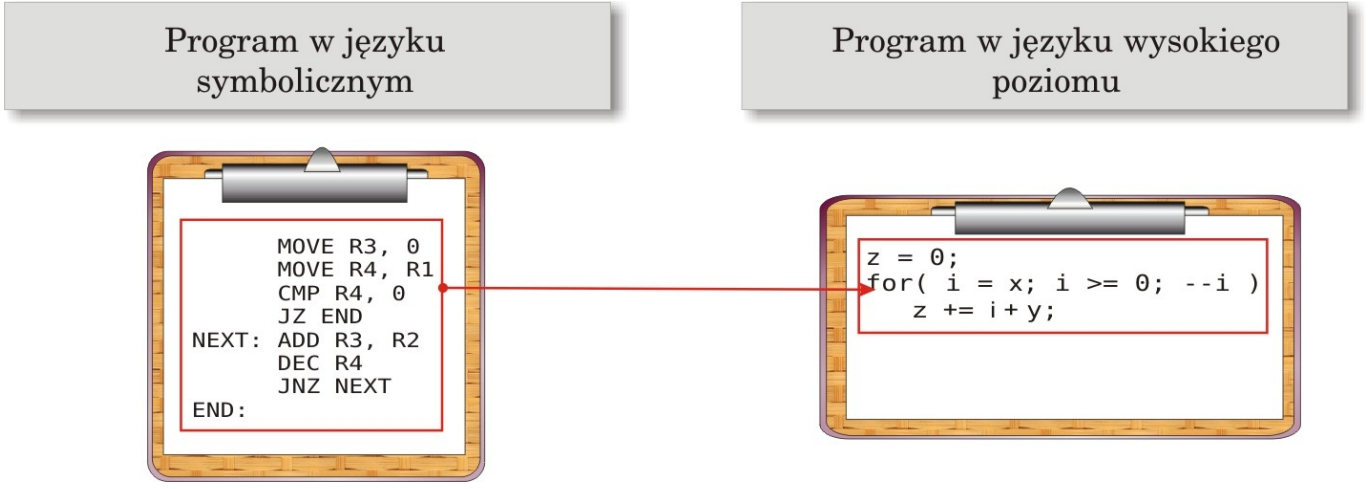
Programowanie w języku symbolicznym

- **Język symboliczny** (język asemblera) umożliwia pisanie programów z wykorzystaniem symboli (mnemonik, mnemoników) przypisanych poszczególnym rozkazom procesora.
- **Mnemoniki** to symboliczne oznaczenia rozkazów i innych elementów, np. rejestrów — *mov*, *move*, *ld*, *add*, *sub*, *out*, *ax*, *bc*, *si*, *di*, *ss*, … .
- **Asembler** to program dokonujący *tłumaczenia kodu źródłowego* programu w języku symbolicznym na *kod maszynowy* — tzw. *kod wynikowy*.

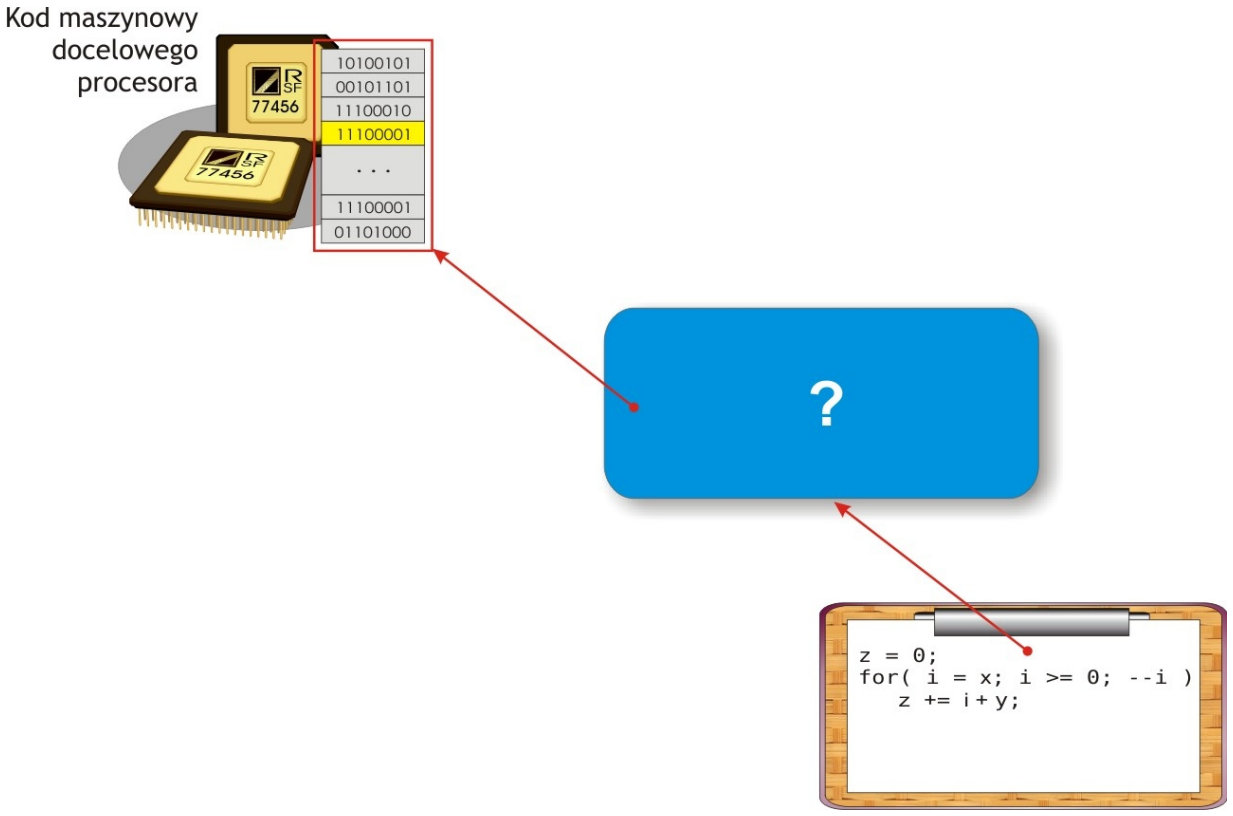


Języki wysokiego poziomu

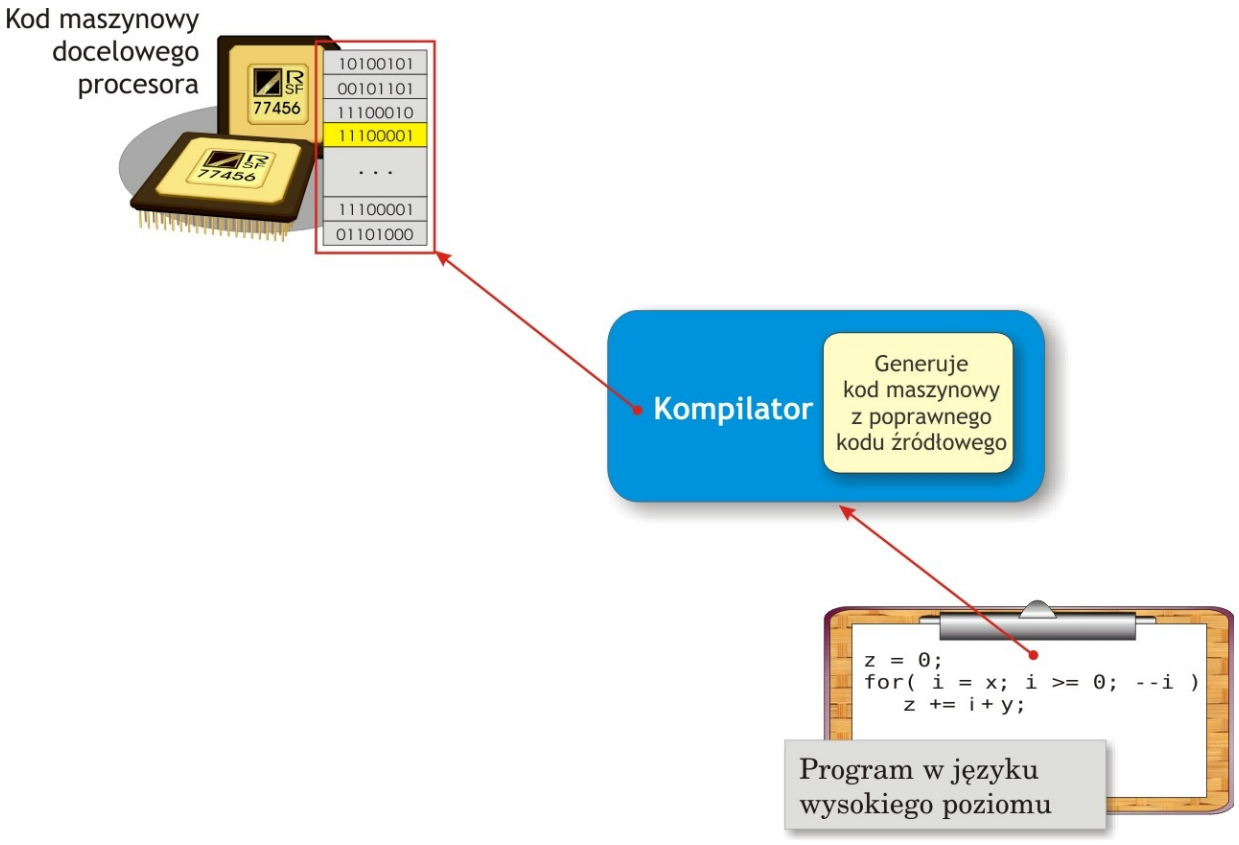
- *Język wysokiego poziomu* — niezależny od platformy sprzętowej i systemowej język programowania, pozwalający programiście skoncentrować się na logice rozwiązywanego problem.
- Notacja typowych języków wysokiego poziomu to połączenie elementów języka angielskiego z notacją wywodzącą się z matematyki.
- Jedna instrukcja programu w języku wysokiego poziomu odpowiada zwykle wielu rozkazom maszynowym.



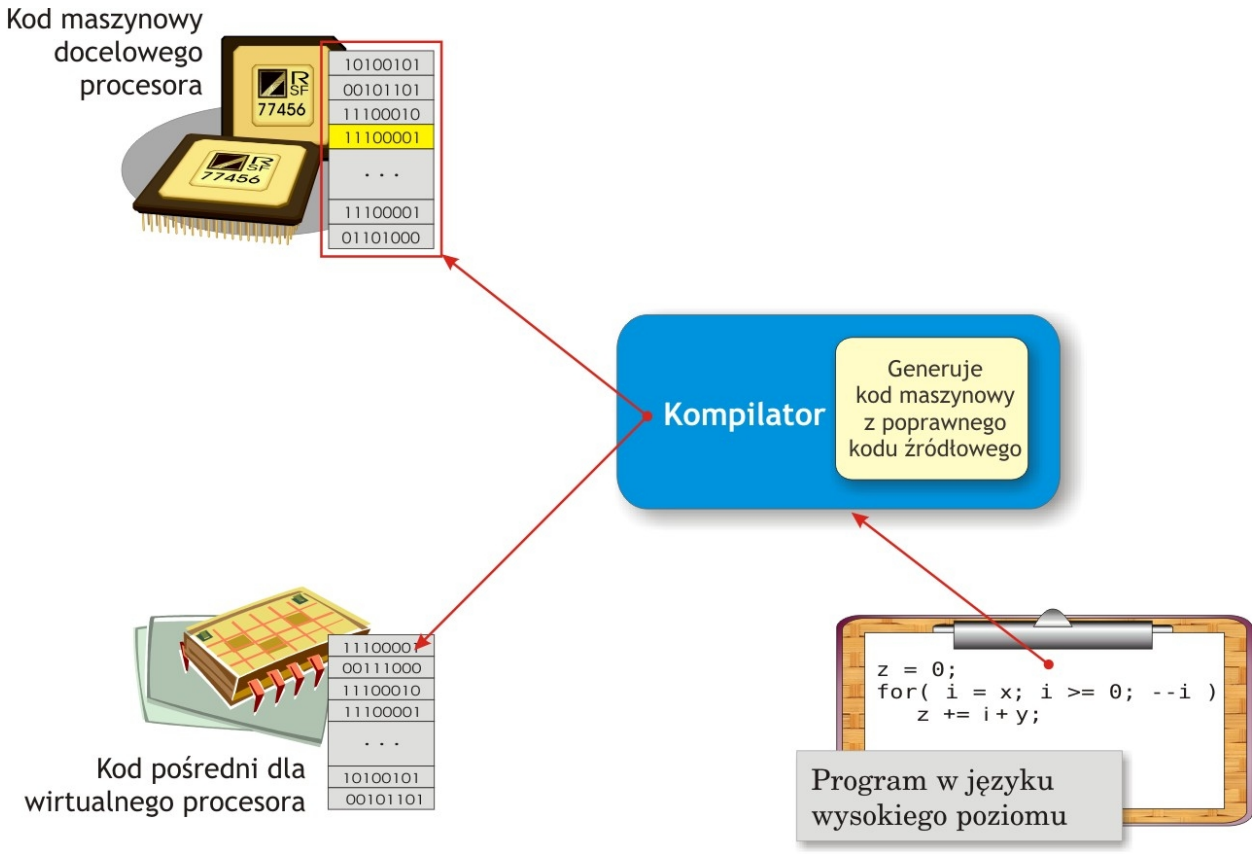
Translacja programu w języku wysokiego poziomu



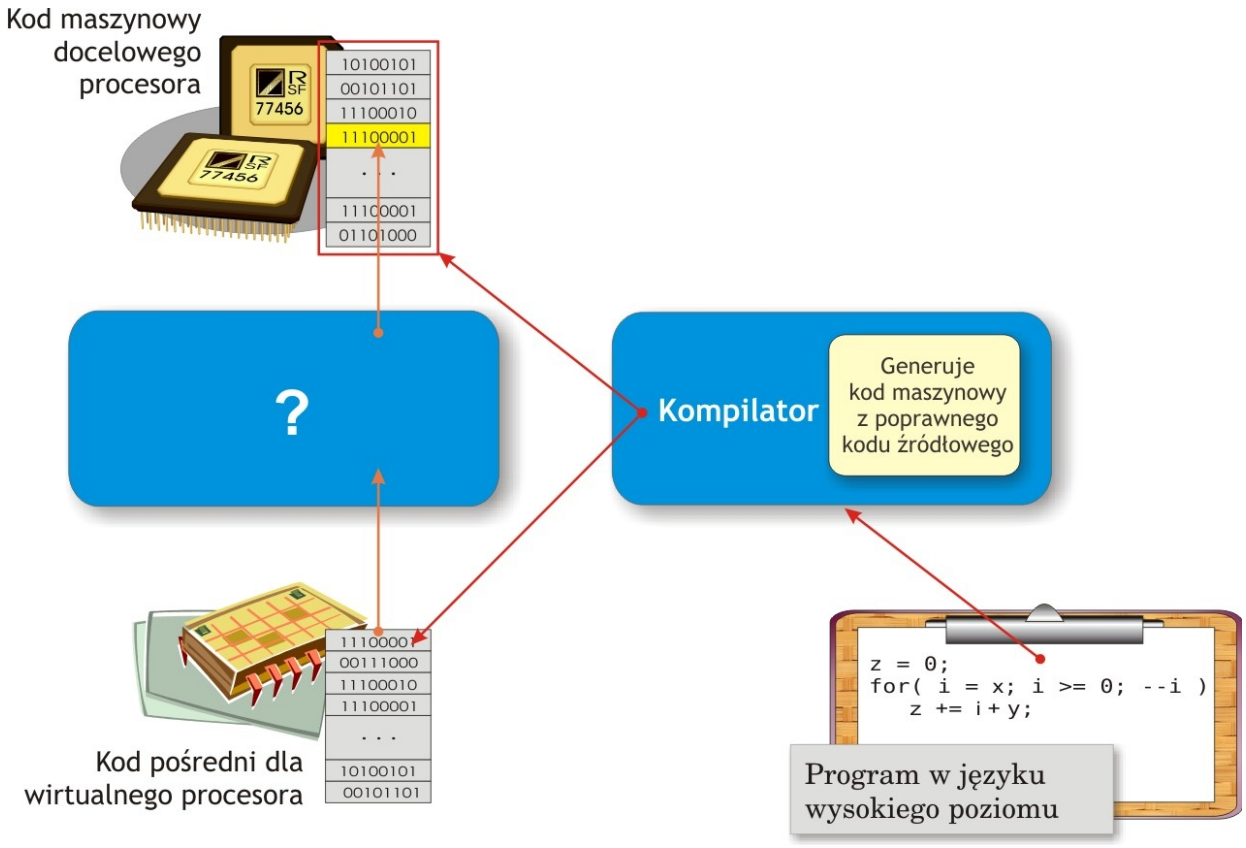
Translacja programu w języku wysokiego poziomu: kompilator



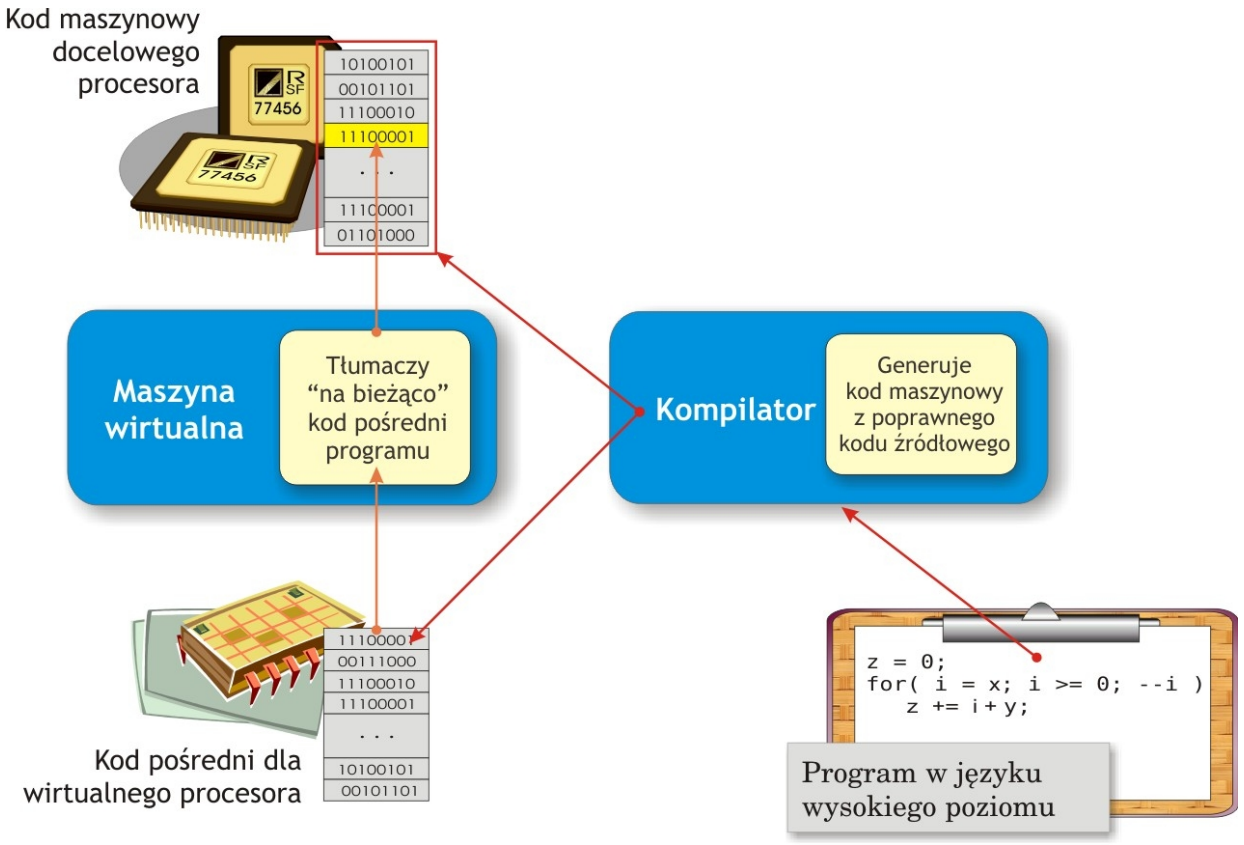
Hybrydowe podejście do kompilacji - kod pośredni



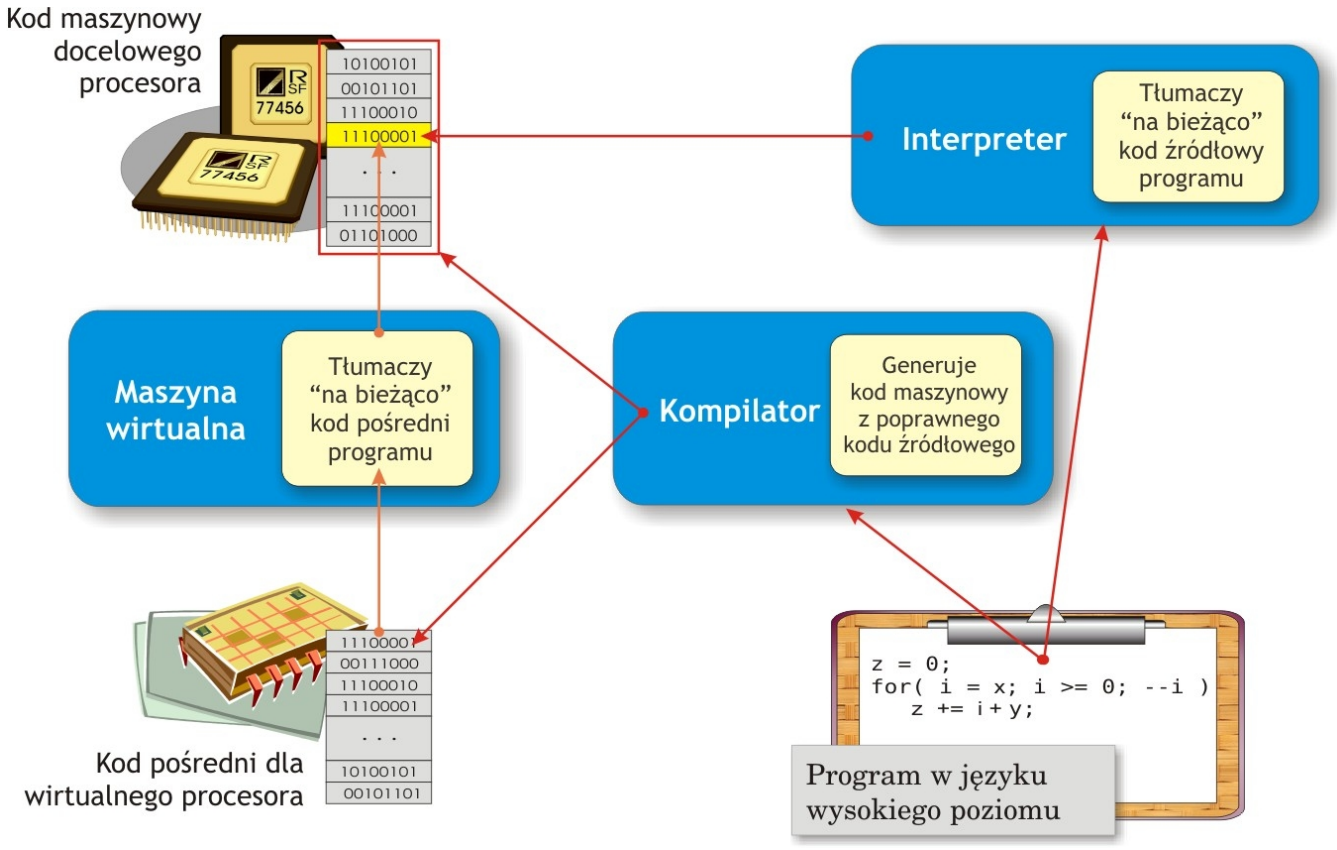
Kod pośredni wymaga kolejnej translacji



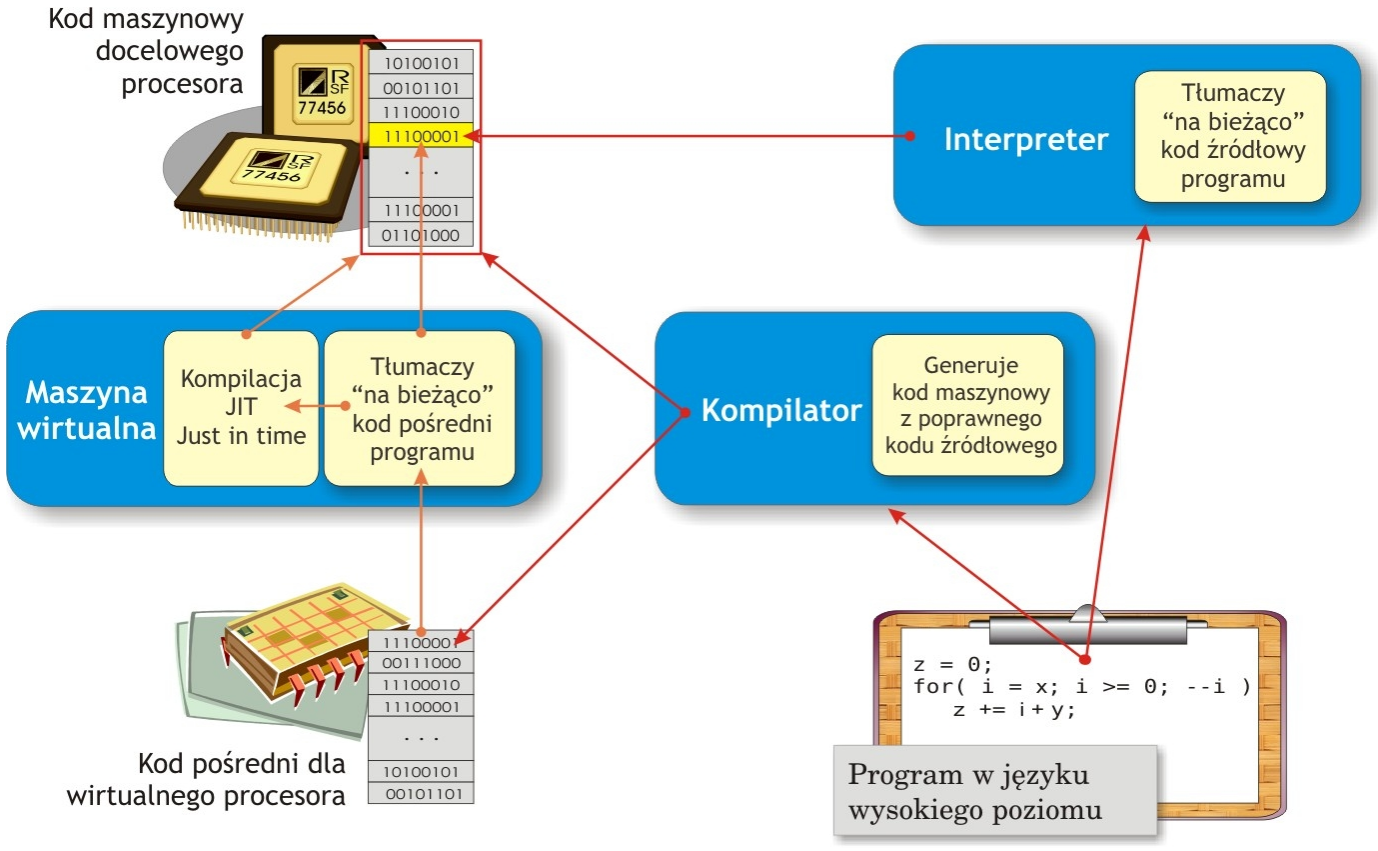
Maszyna wirtualna jako translator kodu pośredniego



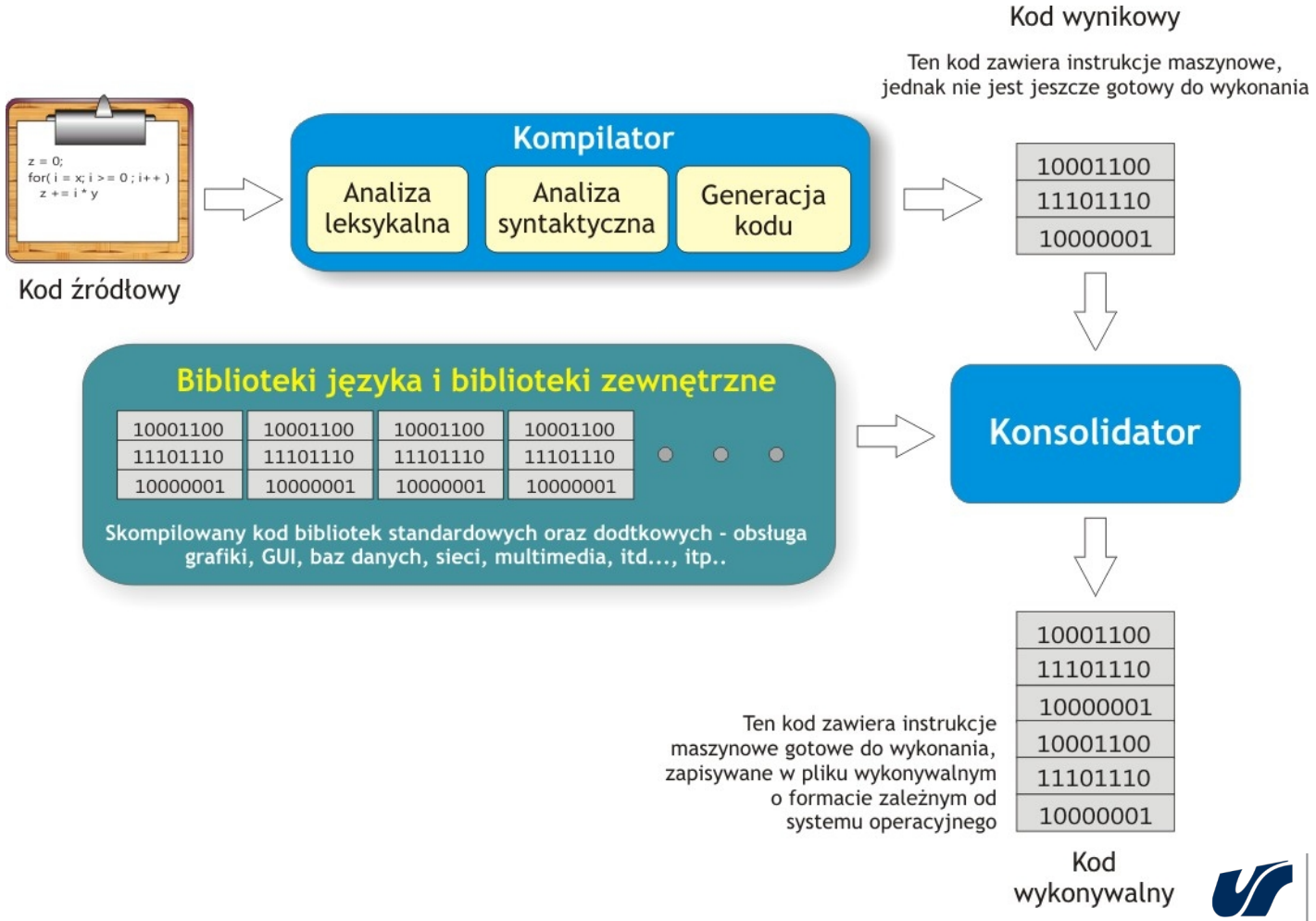
Interpreter - tłumaczenie „symultaniczne”



Kompilacja: Just In Time



Typowy przebieg generowania kodu kompilowanego



Rodzaje aplikacji i języki ich programowania

Aplikacje klasy „desktop”

C/C++	Qt, VCL, MFC, WinForms, WxWidgets, GTK/GTK+
C#	.Net, .Net Core, WinForms, WPF, UWP
Java	JVM, AWT, Swing, FX
Python	QPyt,PyGTK, WxPython, Tkinter
Swift	SwiftUI

Systemy operacyjne, oprogramowanie narzędziowe, programy instalowane na stacja roboczych

Koncepcja aplikacji klient-serwer

Żądanie usługi
Wizualizacja

Warstwa kliencka
Front-end

Żądanie

Odpowiedź

Nasłuchiwanie
Działanie na żądanie

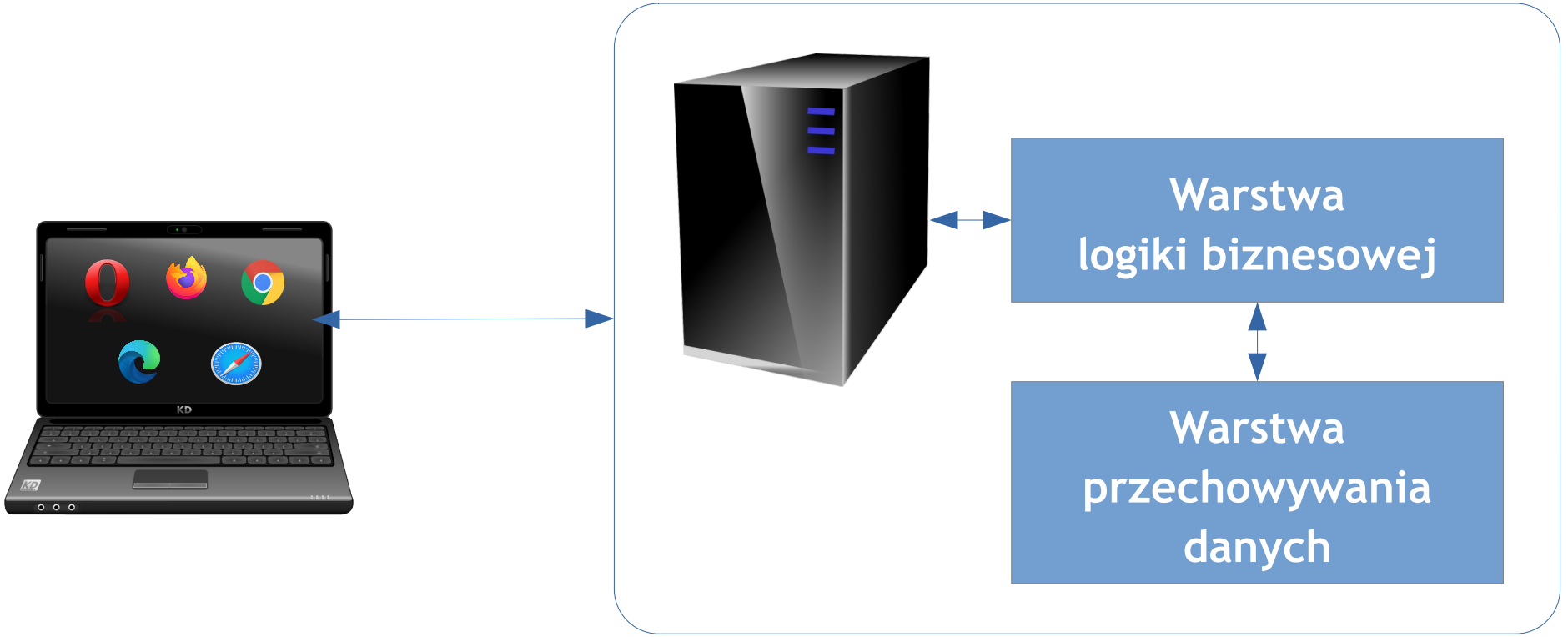
Warstwa serwerowa
Back-end

Warstwa kliencka, front-end

JavaScript TypeScript	Angular, React, Vue.js, Backbone.js, Ember, Electron
HTML CSS	Bootstrap, Foundation, YAML LESS, Skeleton

Front-end to różne aktywności, ale głównie programowanie

Warstwa serwerowa, back-end



Warstwa serwerowa, back-end

C/C++ | Java | C# | Node.js

PHP | Ruby | Python | Golang

Warstwa logiki biznesowej

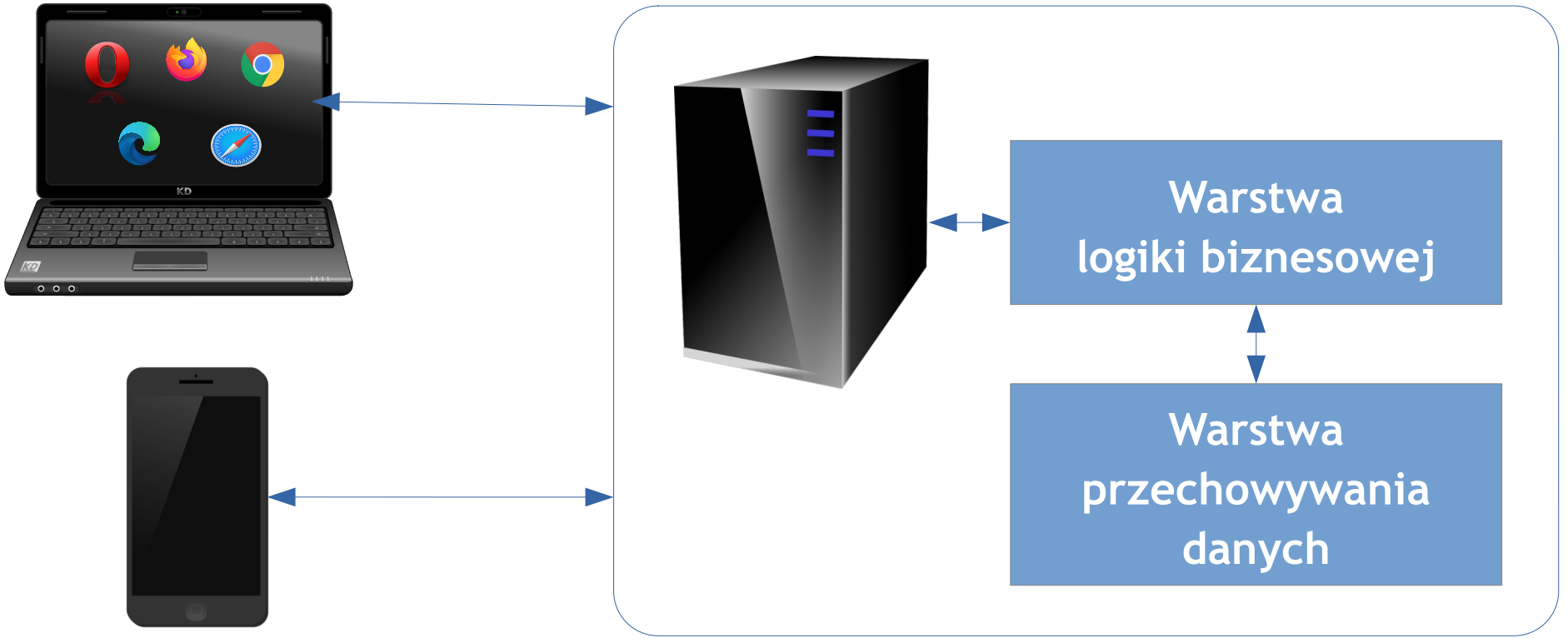
Bazy SQL | Bazy NoSQL

Warstwa przechowywania danych

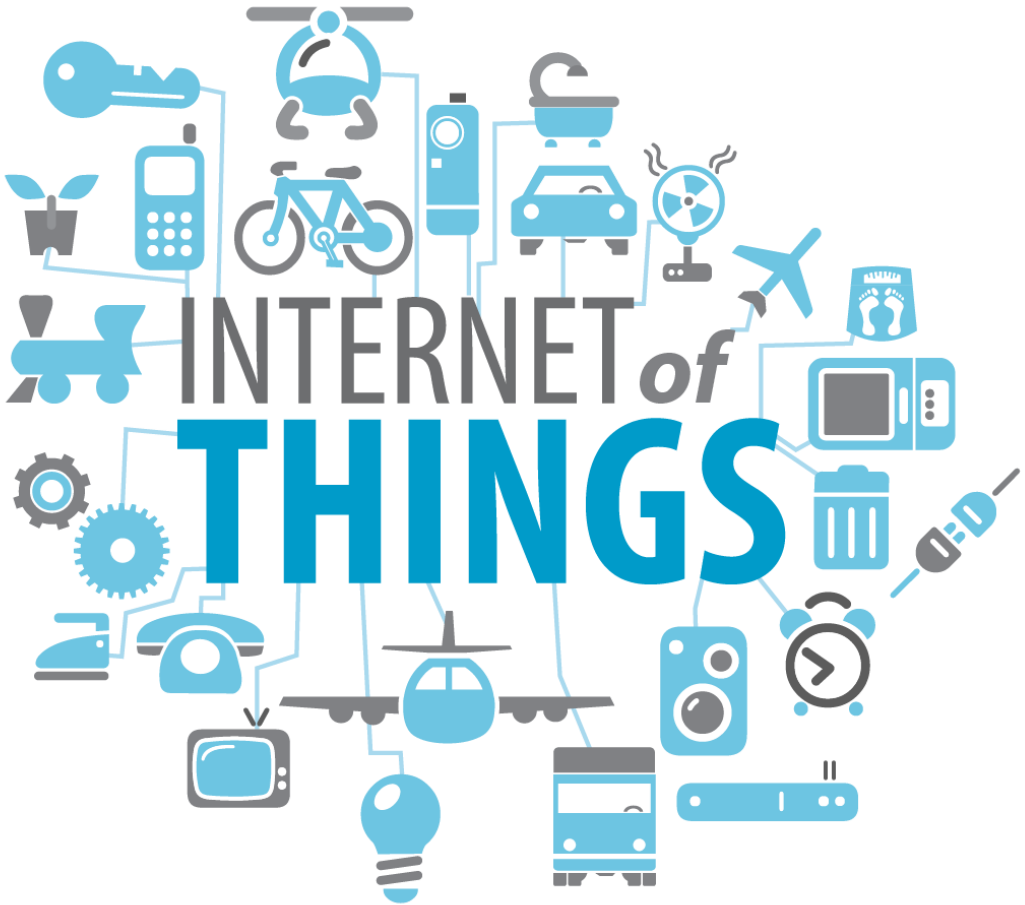
Programowanie urządzeń mobilnych

C/C++ Java/Kotlin Swift	Podejście „natywne” osadzone w kontekście systemu operacyjnego
Progressive Web Apps	Wykorzystanie technologi internetowych: JavaScrip, HTML, CSS + dodatkowe pakiety

Wszystko teraz związane jest z siecią Internet



Internet rzeczy



Źródło: http://bwcio.businessworld.in

Dziękuję za uwagę

roman.siminski@us.edu.pl

UNIWERSYTET ŚLĄSKI
W KATOWICACH