

# Programowanie funkcyjne

## matematyka vs elektronika

# Historia bardzo krótka

- XIX wiek: pierwsze programowalne maszyny
- Lata 30 XX wieku: podstawy teoretyczne wyrażania algorytmów:
  - Rachunek  $\lambda$
  - Maszyna Turinga
- Lata 40 XX wieku:
  - pierwsze komputery zasilane elektrycznie
  - architektura Johna von Neumanna
- Lata 50 XX wieku - paradygmat imperatywny:
  - języki maszynowe (1GL),
  - asemblery (2GL)
  - języki wysokiego poziomu (3GL) (Fortran, ALGOL, COBOL)



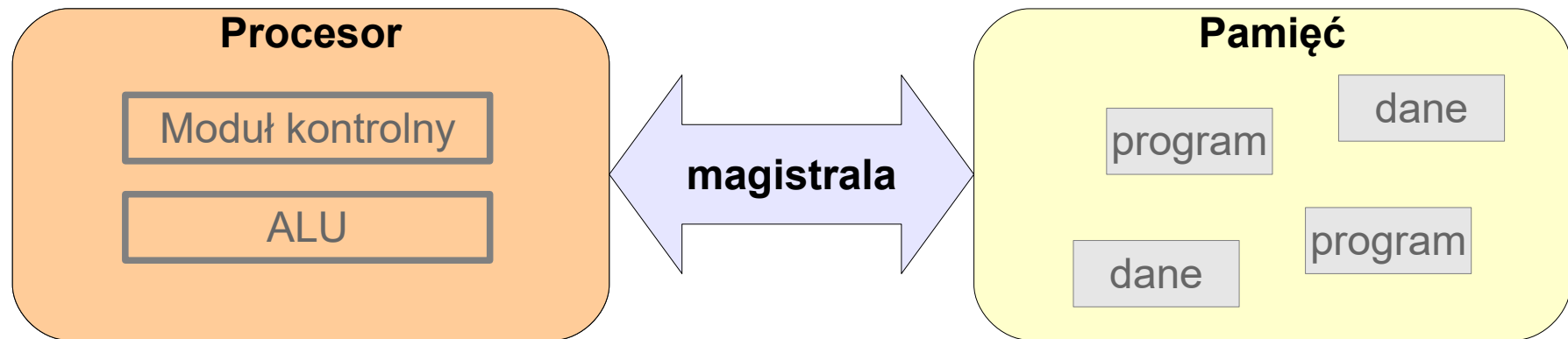
# Historia bardzo krótka

- Lata 60-70 XX wieku: rozwój języków 3GL
  - Programowanie systemowe: C
  - Podejście obiektywne: Smalltalk
  - Programowanie logiczne: Prolog
  - Podejście funkcyjne: Lisp
- Lata 80-90 XX wieku - **paradigm shift**
  - Rozwój imperatywnych języków obiektywnych: C++, Java, ...
  - Języki specjalizowane (4GL) (SQL, PostScript)
- XXI wiek - kolejny paradigm shift ?
  - Integracja koncepcji i paradygmatów
  - Programowanie współbieżne - nowe wyzwania



- Podstawowy paradygmat w programowaniu
- Języki: Fortran, Algol, C, C++, Pascal, Java, C#, Perl, PHP, ...
- Program imperatywny to sekwencja rozkazów do wykonania na określonych danych:
  - Definiowanie zmiennych
  - Modyfikowanie wartości zmiennych
  - Stosowanie struktur kontrolnych: warunków, pętli, ...

# Architektura von Neumanna



- Wspólna pamięć danych i instrukcji
- Magistrala określonej szerokości - słowo maszynowe
- Skończony zestaw rozkazów:
  - Dostęp do komórek pamięci
  - Operacje arytmetyczno-logiczne
  - Skoki do innych instrukcji programu

# Paradygmat imperatywny a architektura von Neumanna

Języki imperatywne wywodzą się bezpośrednio z architektury von Neumanna:

- Typy danych są uwarunkowane długością słowa maszynowego
- Operatory i warunki odzwierciedlają możliwości ALU
- Struktury kontrolne to skoki do określonych instrukcji
- Zmienne to nazwane adresy w pamięci
- Odczyt wartości zmiennej to dostęp do komórki pamięci
- Zmiana wartości zmiennej to modyfikacja zawartości komórki pamięci

- Wartości zmiennych w imperatywnych językach programowania można zmieniać

```
int x;
```

```
x = 2;
```

```
x = 3;
```

- Zmienne są zmienne - mutable variables

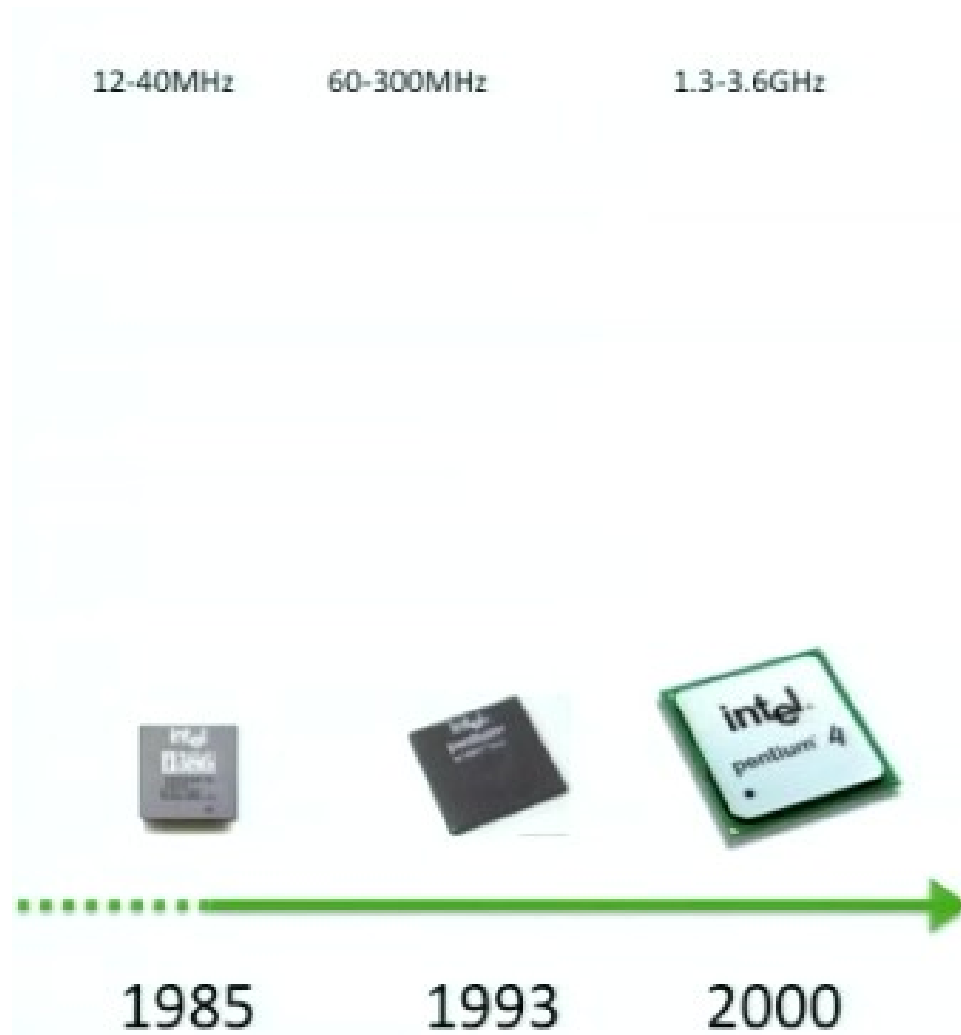
- W twierdzeniach (teoriach) matematycznych:

|                     |   |
|---------------------|---|
| <code>int x;</code> | → definicja niewiadomej $x$ typu <code>int</code> |
| <code>x = 2;</code> | → tu już wiemy, że niewiadoma $x$ to 2            |
| <code>x = 3;</code> | → $2 = 3$   |

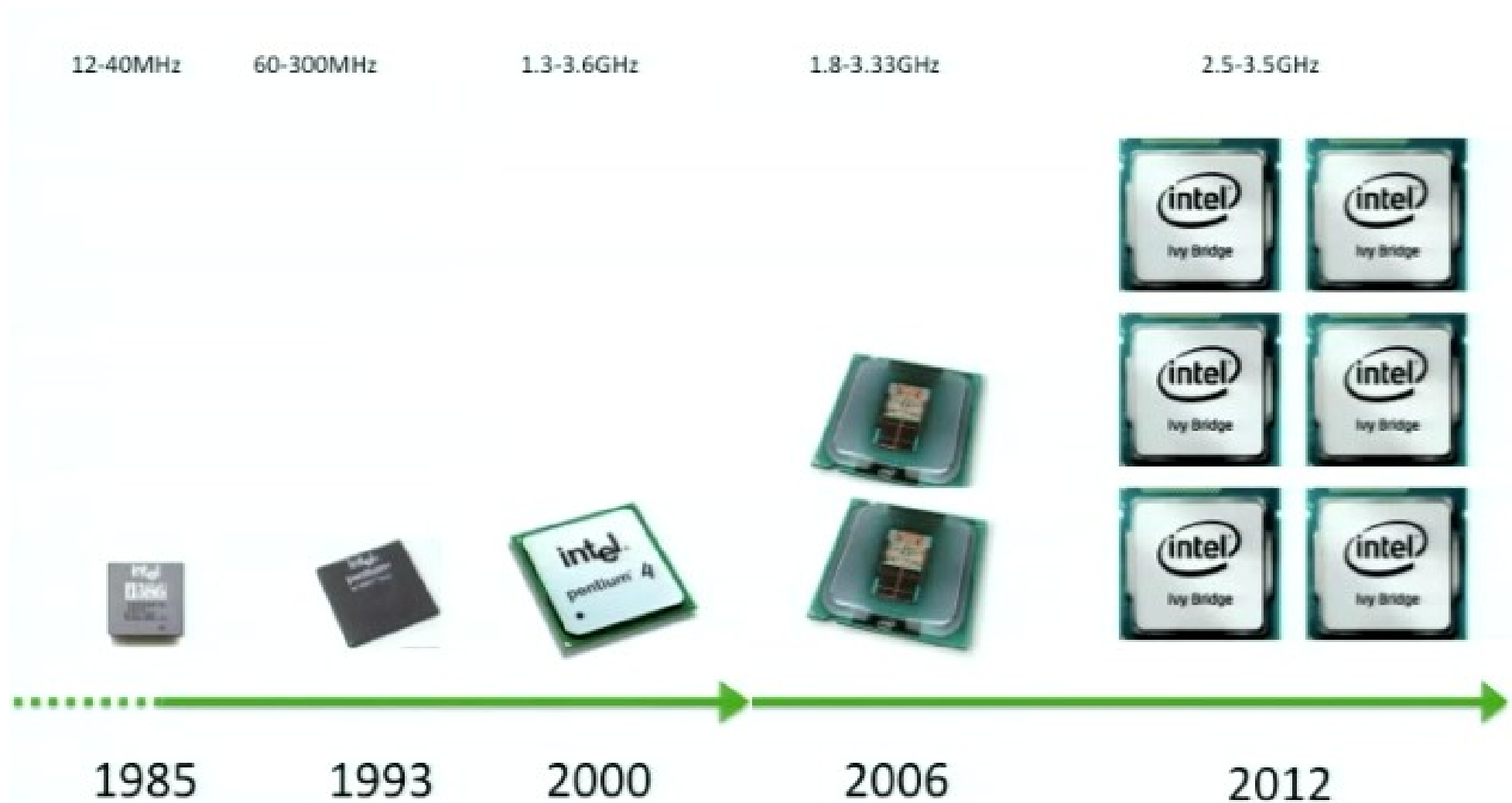
- mutable variables nie pozwalają na stosowanie wielu przydatnych twierdzeń
- W praktyce nie jest to wielki problem



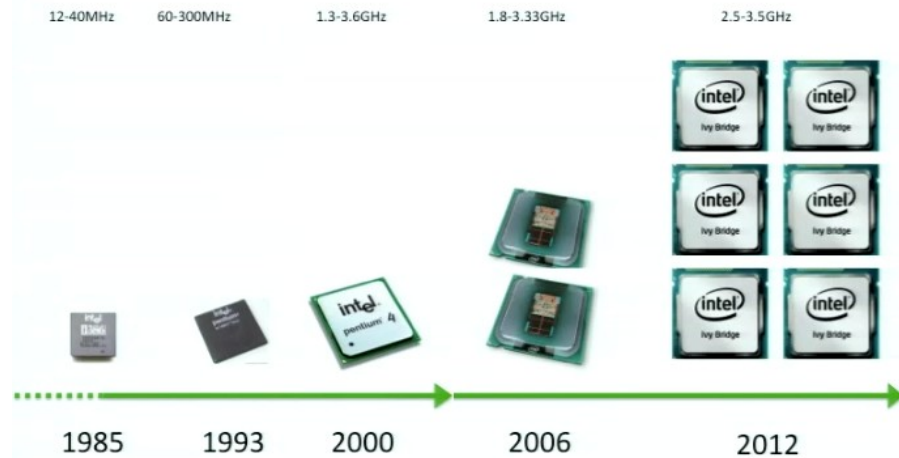
# Architektura von Neumanna i współczesne komputery



# Architektura von Neumanna i współczesne komputery



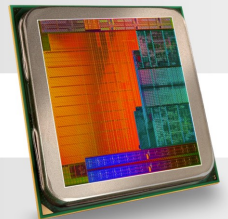
# Architektura von Neumanna i współczesne komputery



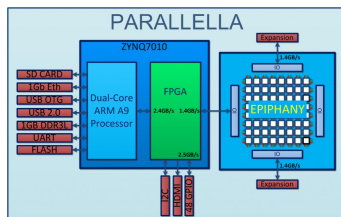
GPGPU

Accelerated  
Processing  
Unit with  
heterogeneous  
Uniform  
Memory  
Access

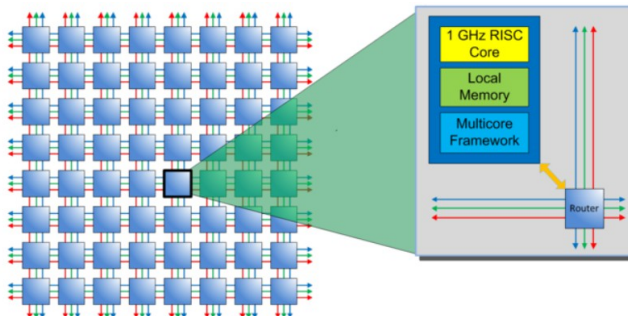
“Kaveri”



AMD's **MOST ADVANCED**  
APU EVER



Epiphany  
accelerator

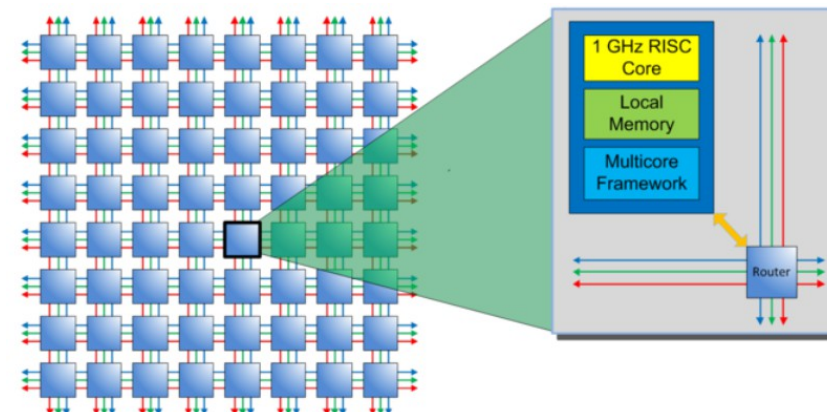


Manycore  
architecture

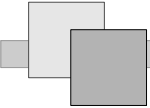
# Mutable variables a współbieżność

- Jeśli zmienne mogą zmieniać wartość to współbieżne wątki współdzielące zmienne muszą synchronizować dostęp, powodując:
  - Wykorzystanie złożonych mechanizmów
  - Wąskie gardła
  - Zakleszczenia
  - Ogromne trudności z testowaniem i wykrywaniem błędów
- W praktyce: będzie to przyczyną kolejnego „paradigm shift”

i dlatego o tym mówimy



# Praca! Programista Java!!!

Two overlapping squares, one light gray and one dark gray, are positioned on the left side of the slide, partially overlapping the title bar.

Podsumowując: najpopularniejsze języki programowania czerpią podstawowe założenia i ograniczenia z architektury sprzętowej opracowanej 70 lat temu!

Nowe systemy ciągle tworzone są głównie w Javie...

# Praca! Programista Java!!!

Podsumowując: najpopularniejsze języki programowania czerpią podstawowe założenia i ograniczenia z architektury sprzętowej opracowanej 70 lat temu!

Nowe systemy ciągle tworzone są głównie w Javie...

„Teraz prędko, zanim dotrze do nas, że to bez sensu”



Król Julian

- Bezwładność
  - Opracowane biblioteki, narzędzia, systemy, standardy
  - Miliony programistów
  - Rozpędzony system edukacji
  - ...
- Ale warto wiedzieć, że można inaczej



# Zadanie domowe

- <http://www.youtube.com/watch?v=TchnPmCGjA>
- <https://www.erlang-factory.com/upload/presentations/834/snakebitten-1.pdf>
- Seven Languages in Seven Weeks:  
A Pragmatic Guide to Learning Programming Languages  
Bruce A. Tate

