

Dr inż. Andrzej Czerepicki

Języki i paradygmaty programowania

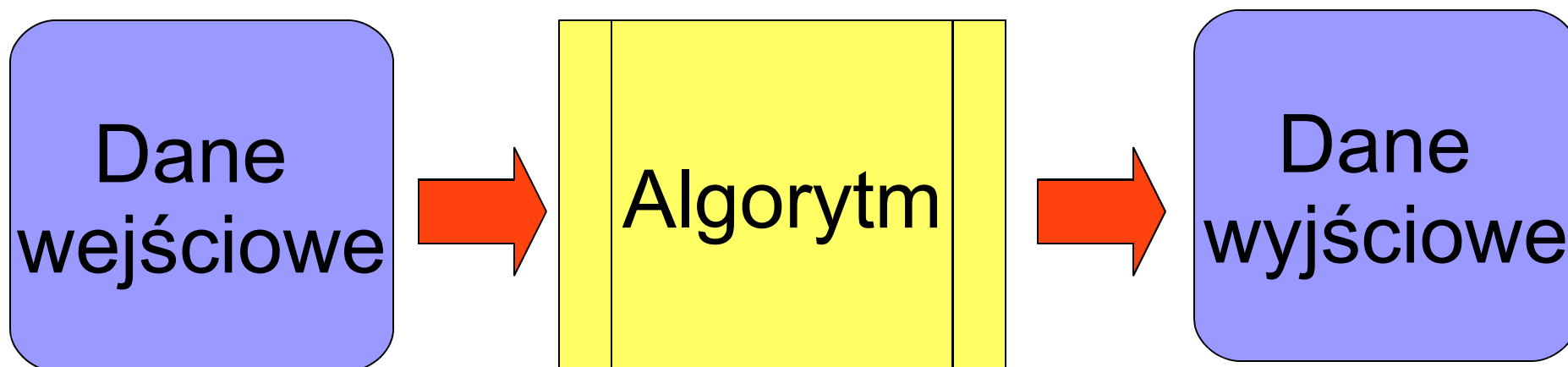
Studia podyplomowe

Wykład 2.

Reprezentacja danych i algorytmów w komputerze

1. Dane wejściowe a wyjściowe

- Głównym celem algorytmu jest przekształcenie **danych wejściowych** na **dane wyjściowe** zgodnie z instrukcjami stanowiącymi treść algorytmu



1. Dane wejściowe a wyjściowe (c.d.)

Przykład: algorytm „wyplata gotówki z bankomatu”

- Danymi wejściowymi są:
 - Numer karty bankomatowej
 - Kod PIN wprowadzony przez klienta
 - Żądana przez klienta kwota do wypłaty
- Danymi wyjściowymi są:
 - Wynik autoryzacji karty
 - Kwota do wypłacenia
 - Paragon

1. Dane wejściowe a wyjściowe (c.d.)

- Dane w systemach komputerowych reprezentują pewną **abstrakcję** przedmiotów lub pojęć świata rzeczywistego
 - W rzeczywistości karta bankomatowa posiada wiele cech (rozmiar, kolor, tworzywo, nazwa banku, numer karty itp.)
 - W kontekście konkretnego algorytmu istotnymi są tylko nieliczne z cech karty (numer i kod PIN)
- » Zbiór istotnych z punktu widzenia algorytmu cech obiektu nazywamy jego abstrakcją

2. Typy danych

- W zależności od rodzaju informacji, którą przechowują obiekty, rozróżniają następujące **typy danych**:
 - Proste (*syn.* podstawowe, wbudowane) typy danych
 - Cyfry
 - Liczby
 - Znaki
 - *Każdy niepodzielny (atomowy) z punku widzenia danego algorytmu bądź wykonującej o maszyny typ danych*
 - Złożone typy danych
 - Typy danych zbudowane na podstawie grupowania prostych typów danych

2. Typy danych (c.d.)

- Przykład (algorytm „Pobranie gotówki”):
 - Proste typy danych:
 - Numer karty (liczba naturalna)
 - Data ważności karty (data)
 - Kod PIN (liczba naturalna)
 - Kwota do wypłaty (liczba całkowita)
 - Złożone typy danych
 - Nazwisko właściciela (ciąg znaków)
 - Kod PIN (jeśli jest traktowany jako ciąg cyfr)
 - Karta bankomatowa jako złożona struktura danych składająca się z pól { numer, data, nazwisko }

2. Typy danych (c.d.)

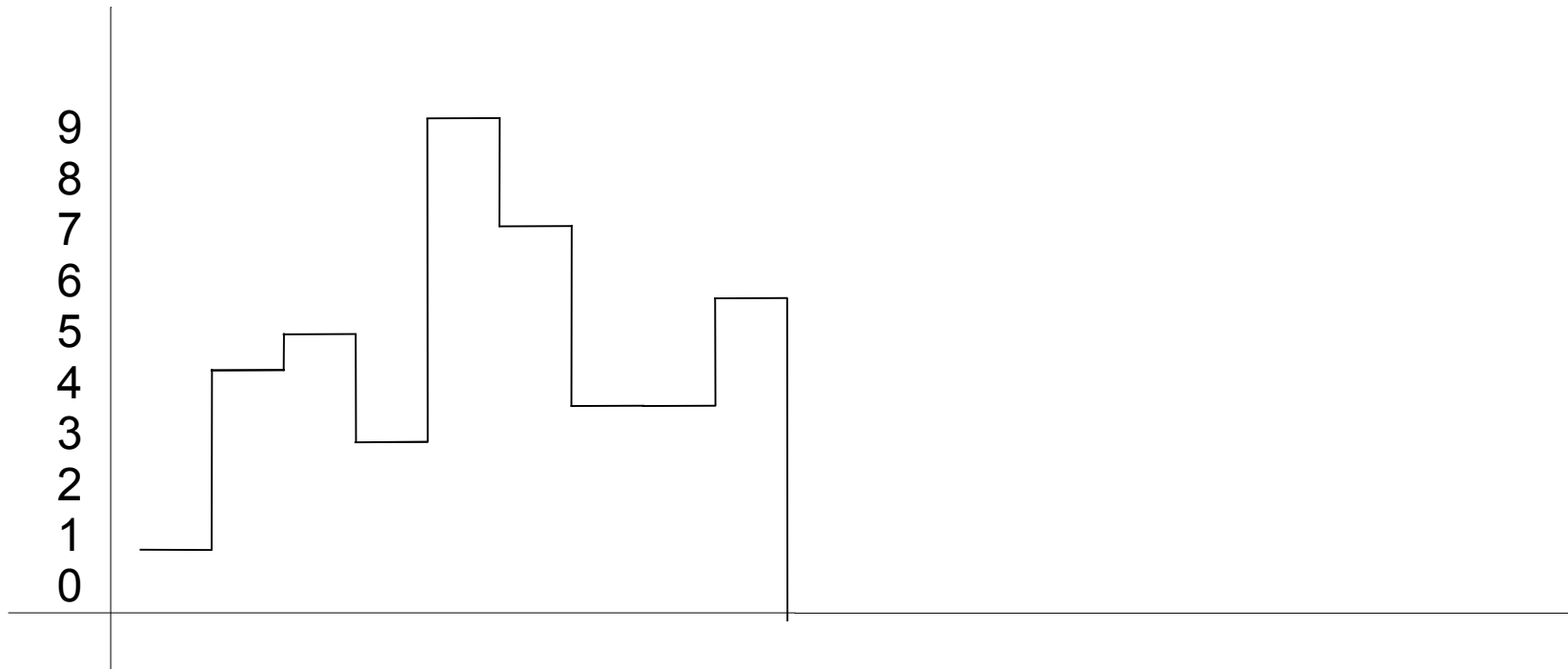
- Granica pomiędzy **prostymi** a **złożonymi typami danych** zależy od konkretnego systemu (dokładnie – języka programowania, ale o tym później)
- Wszystkie **proste typy danych** bazują na zapisie liczbowym:
 - Np. symbol może być reprezentowany przez jego kod ASCII w postaci liczby z zakresu [32..128]
 - Wartość „prawda” czy „fałsz” - przez 1 lub 0 odpowiednio
 - itp.

3. Reprezentacja liczb w komputerze

- Naturalną dla człowieka formą zapisu liczb jest **system dziesiętny**
- Skonstruowanie komputera przechowującego liczby zapisane w systemie dziesiętnym okazało się jednak zadaniem bardzo skomplikowanym i nie odniosło sukcesu
 - Mimo to w latach 50. powstały pewne działające prototypy
 - Ciekawostką jest fakt iż mechaniczne maszyny liczące projektowane w XIX wieku również miały działać w dziesiętnym systemie

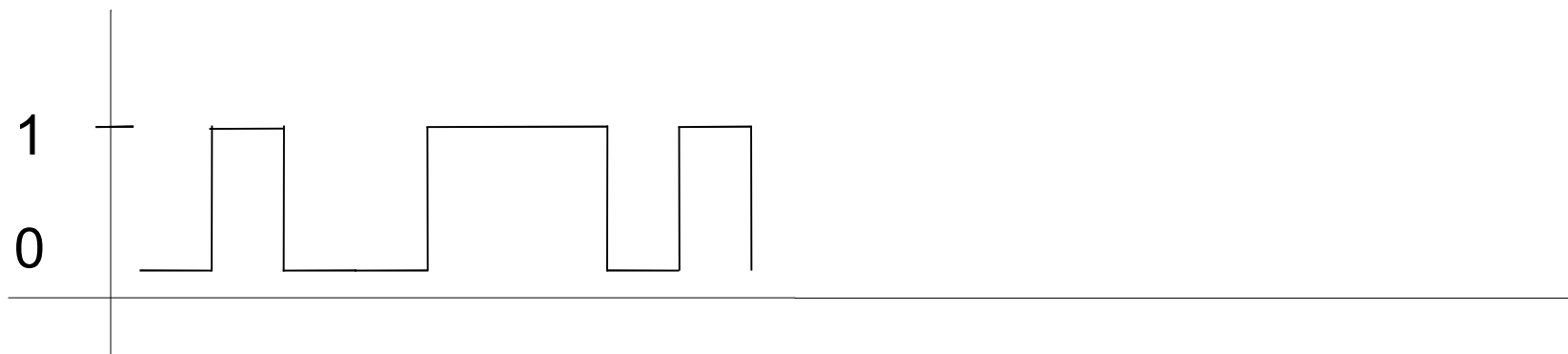
3. Reprezentacja liczb w komputerze (c.d.)

- Główną przyczyną problemów jest konieczność **ilościowej analizy** sygnału elektrycznego, co prowadzi do powstania błędów odczytu oraz wymaga bardzo precyzyjnego sprzętu



3. Reprezentacja liczb w komputerze (c.d.)

- Dopiero zbudowanie komputera działającego na bazie **dwójkowego systemu liczbowego** pozwoliło uzyskać wysoką stabilność systemu:
 - Analiza ilościowa sygnału elektrycznego została zastąpiona **analizą jakościową**:
 - 1 = jest sygnał (napięcie > 0 V)
 - 0 = brak sygnału (napięcie ~ 0 V)



4. Dwójkowy system liczbowy

- Dwójkowy (*syn.* binarny) system liczbowy został opisany po raz pierwszy w pracach niemieckiego matematyka W. Leibniza w XVII wieku
- **Alfabet** dwójkowego systemu liczbowego składa się z dwóch liczb:
 - $\{ 0, 1 \}$
- **Bazą** dwójkowego systemu jest liczba 2

4. Dwójkowy system liczbowy (c.d.)

- Każda liczba w systemie dwójkowym może być zapisana w postaci wyrażenia

$$\sum_{k=0}^n a_k \times 2^k, \quad a_k \in \{0,1\}$$

4.1. Zapis liczb całkowitych w systemie dwójkowym

- Przykład:

1011 (w systemie dwójkowym) =

$$1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 =$$

$$1 * 8 + 0 * 4 + 1 * 2 + 1 * 1 =$$

$$8 + 0 + 2 + 1 =$$

11 (w systemie dziesiętnym)

Wyraz a^b oznacza operację podniesienia a do potęgi b

4.2. Operacje arytmetyczne w systemie dwójkowym

- Podstawową operacją arytmetyczną jest **operacja dodawania** (realizowana wg zasad identycznych z operacją dodawania liczb dziesiętnych)

5	0	1	0	1
+				
7	0	1	1	1
=	<hr/>			
12	1	1	0	0

4.2. Operacje arytmetyczne w systemie dwójkowym (c.d.)

- Pozostałe operacje arytmetyczne (odejmowanie, mnożenie, dzielenie) również są realizowane za zasadach podobnych do systemu dziesiętnego, przy tym należy zauważyć że:
 - Mnożenie przez 2 można zastąpić uzupełnieniem zapisu liczby o dodatkowe 0 w skrajnej prawej pozycji (przesunięcie o 1 znak w lewą stronę)
 - Dzielenie przez 2 można zastąpić przesunięciem zapisu liczby o 1 pozycję w prawą stronę

4.3. Reprezentacja liczb rzeczywistych w systemie dwójkowym

- Liczby rzeczywiste mogą być reprezentowane na dwa sposoby:
 - Zapis stało-pozycyjny
 - Pozycja przecinka jest stała
 - Na wartość liczby składa się **część całkowita** oraz **ułamkowa**
 - Zapis zmienne-pozycyjny
 - Brak określonej pozycji przecinka
 - Na wartość liczby składa się **mantysa** pomnożona przez 2 podniesione do potęgi reprezentowanej przez **cechę**

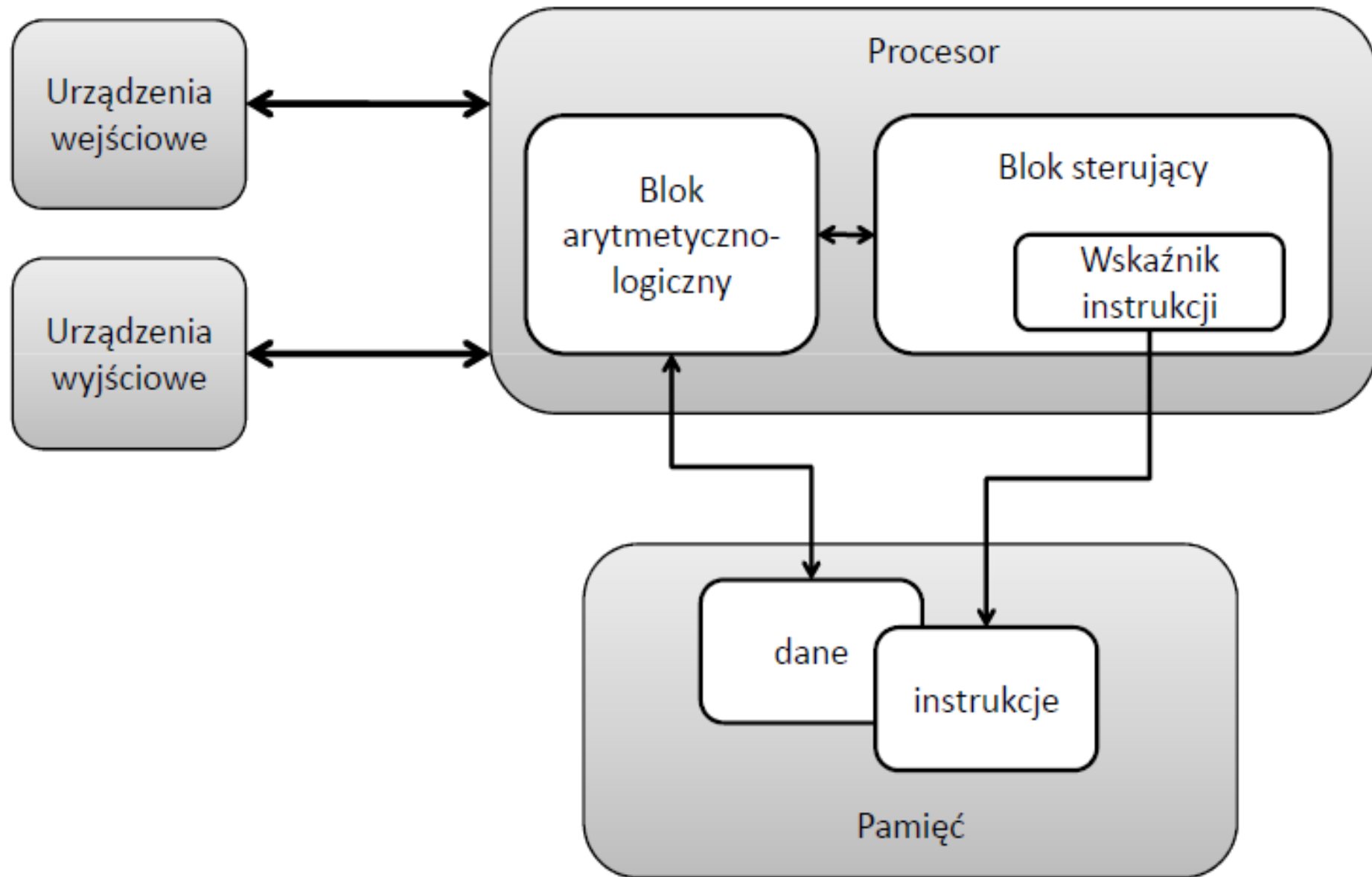
Reprezentacja danych w komputerze (podsumowanie)

- Wewnętrzna reprezentacja **danych** we współczesnych komputerach odbywa się w oparciu o system liczbowy dwójkowy $\{ 0, 1 \}$
 - W sytuacjach wymagających interakcji z operatorem komputera, dane z systemu dwójkowego mogą być konwertowane do systemu dziesiętnego (np. wyprowadzanie informacji na ekran) oraz odwrotnie (np. wprowadzanie danych z klawiatury)
- W jakiej formie należy przedstawić algorytm, który ma być wykonany na komputerze?

5. Reprezentacja algorytmu w komputerze

- Algorytm powinien być zapisany w formie, zrozumiałej dla komputera, na którym ma być wykonany
 - np. pierwszy algorytm „programowania” krosna tkackiego był zapisywany na kartach perforowanych
- Formy zapisu algorytmu (p. wykład 1) są różne, wybór konkretnej na ogół zależy od **architektury komputera**
- Współczesne komputery są zbudowane w oparciu o **architekturę von Neumanna**

5.1. Architektura von Neumanna



5.1. Architektura von Neumanna (c.d.)

- Podstawowymi elementami komputera są:
 - Procesor
 - Odpowiada za wykonanie **operacji arytmetycznych** oraz steruje **kolejnością wykonania instrukcji**
 - Pamięć
 - Przechowuje **dane** niezbędne do działania algorytmu oraz sam **algorytm** w postaci zbioru instrukcji programu
 - Urządzenia wejściowe i wyjściowe
 - Odpowiadają za komunikację z otoczeniem, **wprowadzanie danych** wejściowych do systemu oraz **wyprowadzanie wyników** działania algorytmu

5.1. Architektura von Neumanna (c.d.)

- Architektura von Neumanna wprowadza wyraźny podział części składowych komputera na **sprzęt** (ang. *hardware*) oraz **oprogramowanie** (ang. *software*)
 - Sprzęt jest niezmienny
 - Oprogramowanie może się zmieniać niezależnie od sprzętu

5.1. Architektura von Neumanna (c.d.)

- Postulat programowego sterowania:
 - **Komputerem steruje program** (nie odwrotnie!)
 - Program realizuje pewien algorytm i jest ciągiem jednoznacznych instrukcji
 - Instrukcje są przechowywane w pamięci komputera
 - Kolejność wybierania instrukcji zależy od programu
 - Algorytm liniowy za każdym razem wybiera następną instrukcję, zaś złożone algorytmy potrafią modyfikować adres następnej instrukcji w zależności od kontekstu wykonania

5.1. Architektura von Neumanna (c.d.)

- Postulat jednorodnej pamięci:
 - Dane i programy (algorytmy) są przechowywane wspólnie w pamięci komputera
 - Pamięć składa się z wielu niezależnych ponumerowanych komórek, każda z których może być dostępna do zapisu lub odczytu
 - Program może być modyfikowany w trakcie wykonania
 - **Program może być wynikiem działania innego programu**
 - Na tej zasadzie działają m. innymi kompilatory

5.2. Inne architektury komputerowe

- Architektura von Neumanna odegrała ogromną rolę w rozwoju komputerów, faktycznie wypierając inne architektury
- Na uwagę zasługuje również tzw. architektura Harvardzka, polegająca na **osobnym przechowywaniu danych oraz programów**, lecz ze względu na większe nakłady i złożoność, została ona zaimplementowana w kilku komputerach eksperymentalnych
- Mieszane architektury są wykorzystywane w niektórych rodzajach mikrokontrolerów

Podsumowanie

- Architektura von Neumanna wymaga jednolitego formatu przechowania **algorytmu oraz danych**
- Dane są przechowywane w **systemie dwójkowym (binarnym)**
- **Algorytm w celu wykonania na komputerze również musi być zapisany w formacie binarnym (!)**
 - Format ten często jest nazywany **kodem maszynowym** (*syn. kod wykonywalny, ang. executable*)

Podsumowanie (c.d.)

- Zapis algorytmu w kodzie maszynowym jest idealny do wykonania na komputerze, lecz zupełnie nieczytelny dla człowieka
 - W latach 50. właśnie tak projektowano większość algorytmów
 - Koszt zrozumienia, znalezienia błędu oraz poprawienia algorytmu tak zapisanego był bardzo wysoki
- Poszukiwania formy zapisu algorytmu, która by pozwoliła na łatwiejsze programowanie algorytmów przez człowieka, doprowadziły do stworzenia **języków programowania**