Technika Cyfrowa oraz Technika Mikroprocesorowa

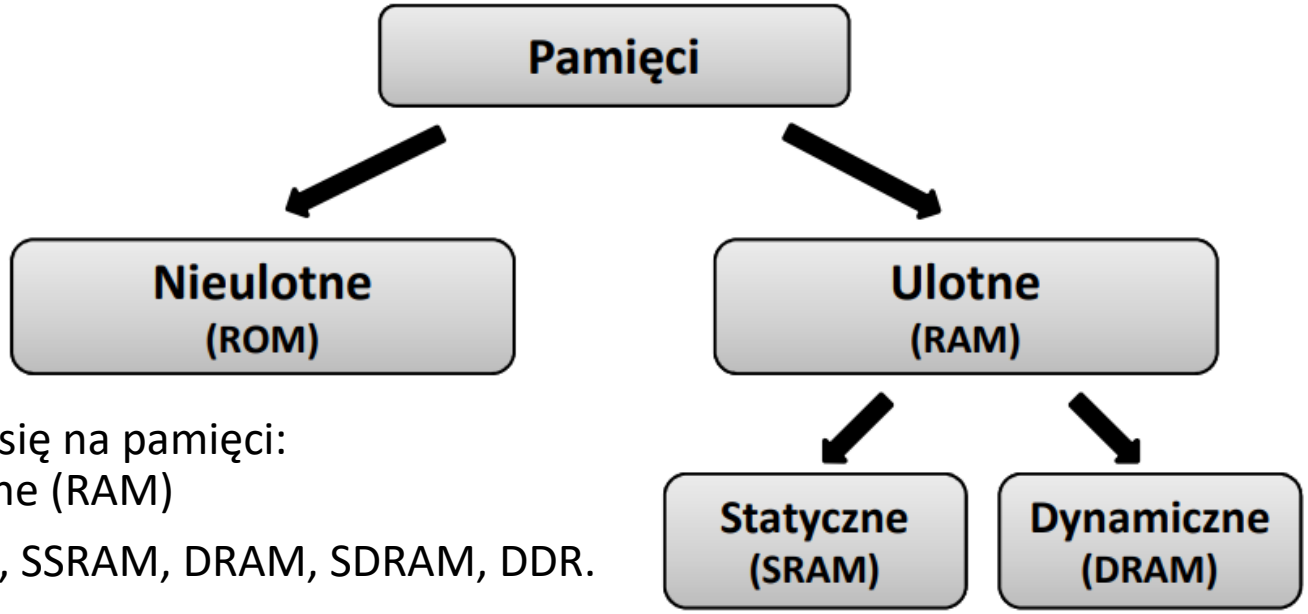
Przygotował: Piotr Pater

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
AGH University of Krakow

06.11.2025

5. Pamięci komputerowe

Pamięć komputerowa to kluczowy element systemu cyfrowego umożliwiający przechowywanie danych, instrukcji oraz stanów pośrednich.



Dzieli się na pamięci:
- Ulotne (RAM)

SRAM, SSRAM, DRAM, SDRAM, DDR.

- Nieulotne (ROM)
MPROM, PROM, EPROM, EEPROM, FLASH.

5.1 Technologie pamięci

• **SRAM** - szybka pamięć oparta na przerzutnikach bistabilnych, statyczna. Nie wymaga odświeżania. Dane zapisane w niej będą przechowywane do momentu zaniku zasilania.

• **DRAM** - pamięć dynamiczna o dostępie swobodnym wykorzystująca kondensator i wymagająca odświeżania. W przeciwnym razie dane w niej ulegają degradacji.

• **HBM** - pamięć wysokoprzepustowa zintegrowana warstwowo (stacked memory) w GPU.

• **EEPROM -** pamięci do trwałego zapisu danych i konfiguracji układów kasowana na drodze czysto elektrycznej.

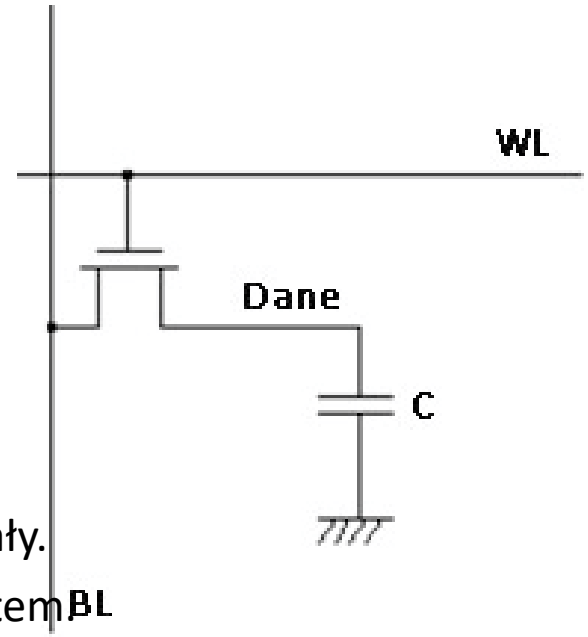
• **Flash** - nieulotna pamięć półprzewodnikowa z interferjsem szeregowym umożliwiająca szybki zapis blokowy. Jest to specyficzny rodzaj pamięci EEPROM. Różnica występuje podczas kasowania, zamiast kasowania pojedynczego słowa kasowany jest cały blok komórek, zazwyczaj 1024 bajty. Dzięki temu możliwy jest dużo szybszy zapis nowych treści.

• **ROM -** swobodny dostęp, dawniej tylko do odczytu. Informacja nie jest tracona po utracie zasilania.

5.2 Architektura pamięci

Struktura hierarchiczna:

- Dane organizowane są od komórek → bloków/stron → banków pamięci.
- Umożliwia równoległy dostęp i dużą przepustowość.



Komórka pamięci:

- Podstawowa jednostka – 1 bit informacji (0/1).
- SRAM: 6 tranzystorów – szybka, bez odświeżania.
- DRAM: 1 tranzystor + kondensator – wymaga odświeżania.
- Flash/EEPROM: tranzystor z bramką pływającą – zapis trwały.
- Linie bitowe (BL) i linie słowa (WL) sterują zapisem i odczytem.

Bloki i strony:

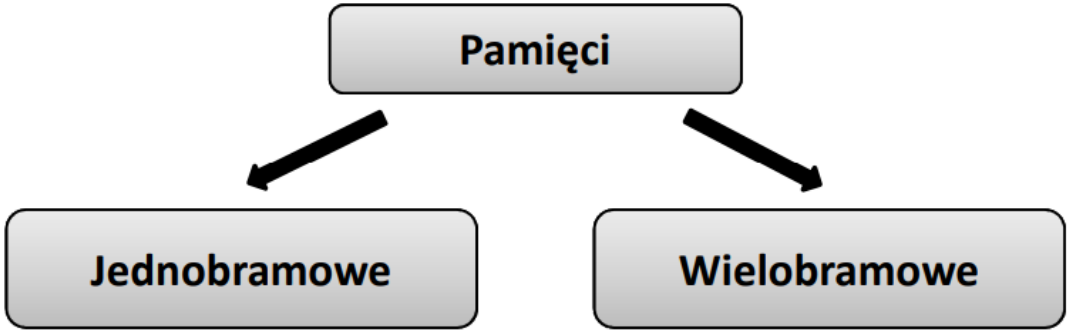
- Blok – jednostka zapisu/kasowania.
- Strona – najmniejsza jednostka odczytu/zapisu (2–16 kB).

Banki pamięci:

- Niezależne segmenty umożliwiają równoległy dostęp.
- DDR4/DDR5 – 16+ banków w grupach, interleaving.

Hierarchia systemowa:

Rejestry → Cache (SRAM) → RAM (DRAM) → SSD/HDD (Flash)



- **Pamięci jednobramowe (jednoportowe) -** pamięć umożliwiająca dostęp do danych tylko jednemu procesowi.
- **Pamięci wielobramowe (wieloportowe)** - pamięć umożliwiająca jednoczesny dostęp do danych wielu procesom. Najczęściej są to pamięci dwubramowe. Umożliwiają dwóm niezależnym procesom dostęp do wspólnych danych. Dzięki temu możliwa jest np. szybka wymiana danych między dwoma procesami. W tym celu pamięć wielobramowa musi mieć oddzielne interfejsy czyli osobne zestawy linii adresowych, danych i sterujących.

5.3 Zasada działania odczytu i zapisu

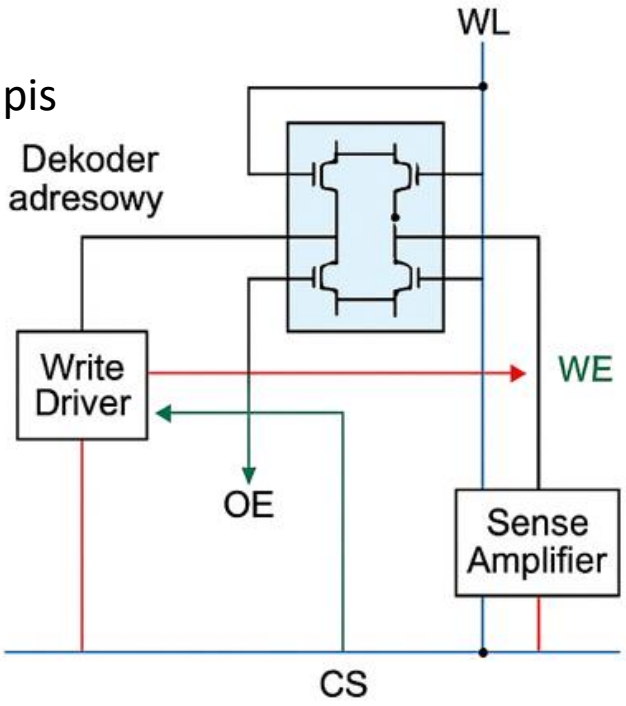
Proces zapisu (Write Operation):

• Adresowanie komórki - dekoder adresowy wybiera odpowiednią linię słowa (WL).

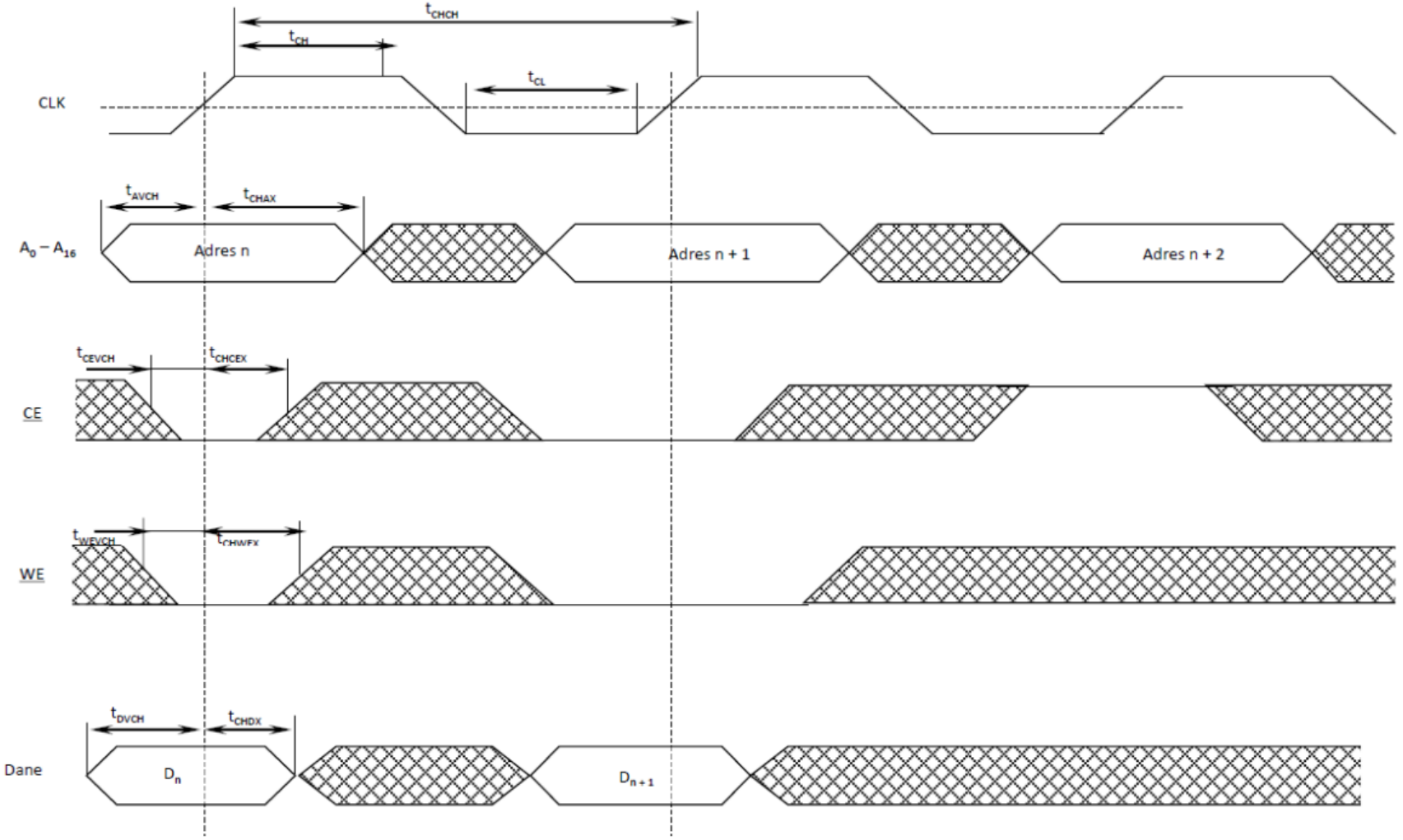
• Aktywacja sygnału WE (Write Enable) - umożliwia zapis do wybranego wiersza.

• Wprowadzenie danych - dane z magistrali danych (Data Bus) są przesyłane na linie bitowe (BL).

• Zapis w komórce - tranzystory komórki ustawiają stan logiczny 0 lub 1 w zależności od napięcia na liniach BL.



Proces Zapisu (SRAM)



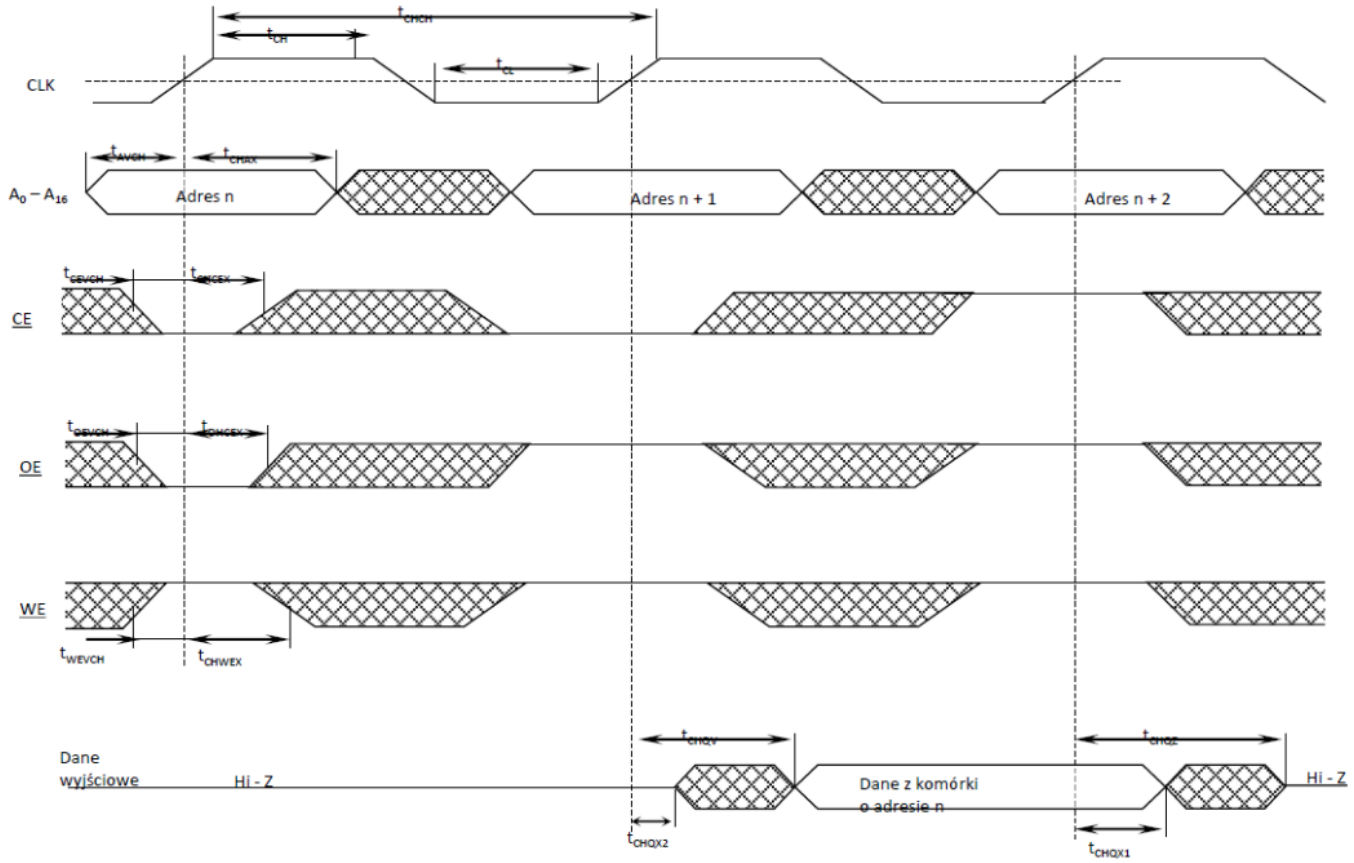
- **Proces odczytu (Read Operation):**

1. Wybór adresu - dekoder adresowy wybiera odpowiedni wiersz i kolumnę komórki.

2. Aktywacja sygnału OE (Output Enable) - włącza bufory wyjściowe.

3. Odczyt danych - stan logiczny komórki jest przenoszony na linię BL i wykrywany przez wzmacniacz odczytu (sense amplifier).

Proces Odczytu (SRAM)



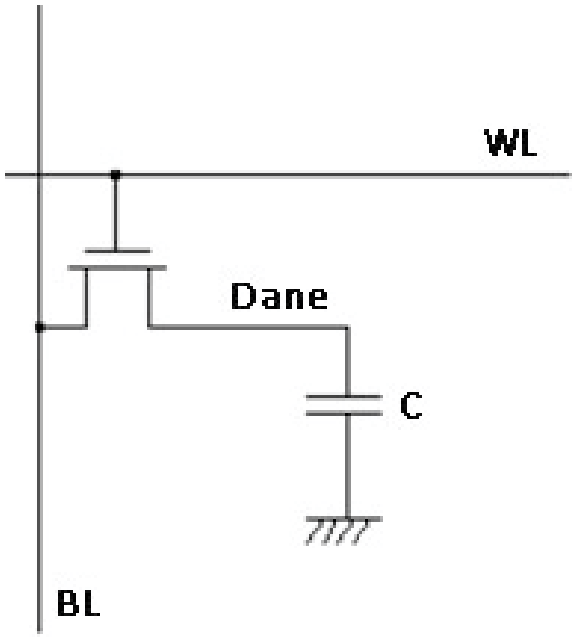
- **Pamięć DRAM (ang. Dynamic RAM) -** ulotna dynamiczna pamięć półprzewodnikowa. Dynamiczna oznacza, że aby dane w niej zgromadzone nie uległy degradacji musi być czasowo odświeżana. Cykl odświeżania powinien następować co ok. 8-20 ms. Informacja przechowywana jest w postaci ładunku elektrycznego.

- **Operacja zapisu**

-podajemy napięcie na linie bitowa - wysokie dla 1 i niskie dla 0;

- podajemy sygnał na linie adresową;

- następuje przepływ prądu, który ładuje kondensator.



- **Operacja odczytu**

- podajemy sygnał na linie adresową - następuje otwarcie kanału tranzystora;

- kondensator rozładowuje sie poprzez linie bitowa do czujnika;

- następuje porównanie odczytanej wielkości ze wzorcem, aby określić wartość bitu (0 lub 1);

- stan naładowania kondensatora zostaje odtworzony

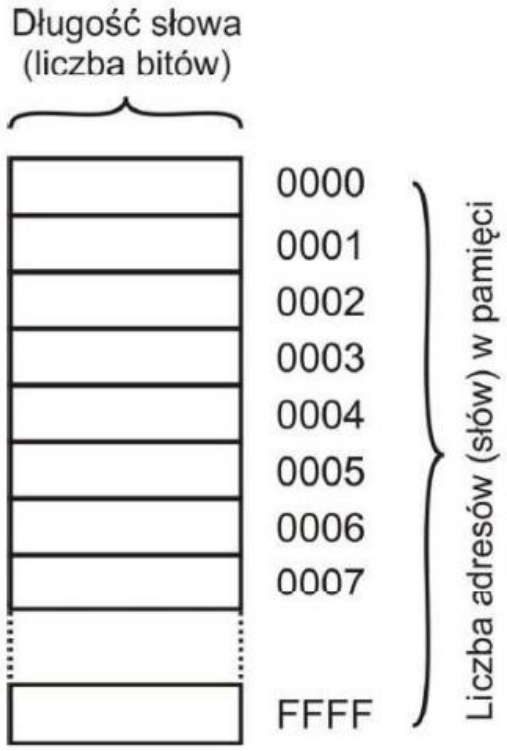
5.4 Parametry pamięci

Czas dostępu (t_ACC) - opóźnienie między adresem a pojawieniem się danych. Czas, jaki musi upłynąć od momentu podania poprawnego adresu słowa w pamięci do czasu ustalenia się poprawnej wartości tego słowa na wyjściu pamięci w przypadku operacji odczytu lub w przypadku operacji zapisu to czas jaki upłynie od momentu zapisania wartości do tego słowa z wejścia pamięci. Mierzony w nanosekundach, im krótszy, tym szybsza pamięć.

- Typowe wartości:
- SRAM: 1–10 ns
- DRAM: 20–60 ns
- Flash: 50–150 ns

Wpływają na niego:

- pojemność
-architektura banków i organizacja magistrali.



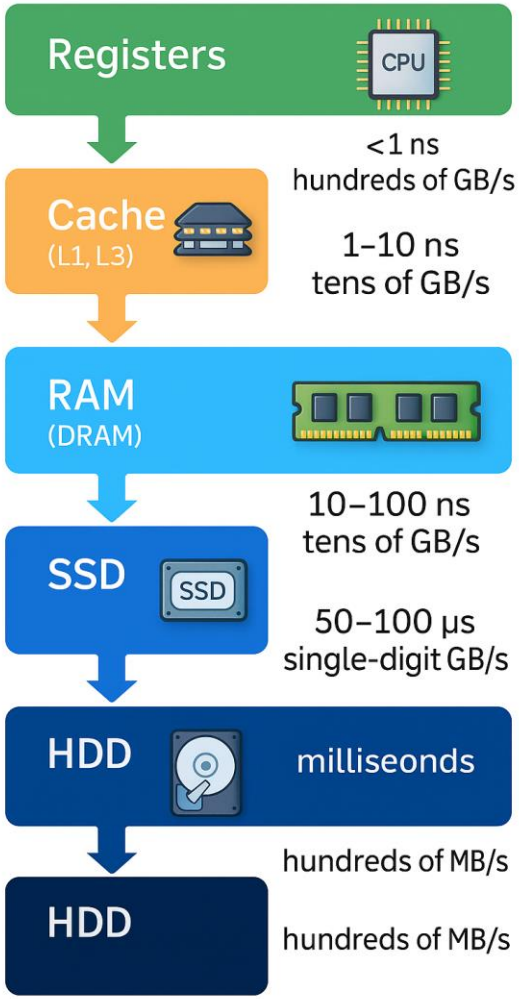
• **Przepustowość** - ilość danych, które mogą zostać przesłane w jednostce czasu. Zależna od szerokości magistrali danych (bus width) oraz częstotliwości zegara, określana w bajtach na sekundę.
B = szerokość magistrali * częstotliwość / 8
Przykład: DDR4-3200 → 25,6 GB/s przy 64-bitowej magistrali.

• **Pojemność** - ilość bitów/bajtów, które pamięć może przechować.

Typowe poziomy:
Cache (SRAM): kilkadziesiąt kB do kilku MB

RAM (DRAM): kilka – kilkadziesiąt GB

Flash / SSD: setki GB – kilka TB



5.5 Pamięci specjalizowane

Typ pamięci	Zasada działania	Typowe zastosowanie	Organizacja danych
FIFO	Pierwszy zapisany = pierwszy odczytany	Bufor transmisji, interfejsy I/O	Kolejka pierścieniowa
LIFO	Ostatni zapisany = pierwszy odczytany	Stos procesora, rekurencja	Stos (stack)
LUT	Tablica odwzorowań wejść na wyjścia	Logika FPGA, obliczenia tablicowe	Pamięć ROM adresowana wejściem

FIFO - First In, First Out

- Pamięć typu kolejka (ang. *queue*), w której dane są odczytywane w tej samej kolejności, w jakiej zostały zapisane.
- Typowa implementacja opiera się na dwóch wskaźnikach (pointerach):

  write pointer - wskazuje miejsce, w którym zostanie zapisany nowy element,

  read pointer - wskazuje element do odczytu.
- Po każdym zapisie i odczycie wskaźniki są inkrementowane, a po osiągnięciu końca pamięci - zawijane (ang. *wrap-around*).
- Zastosowania:

  - Buforowanie danych między modułami o różnych prędkościach (np. UART, DMA, karty sieciowe).

  - Przesyłanie strumieni danych w systemach DSP i komunikacyjnych.

  - Realizacja kolejek w sprzęcie FPGA lub mikrokontrolerach.

LIFO - Last In, First Out

- Pamięć typu stos (stack) - ostatni zapisany element jest pierwszym odczytanym.
  Operacje podstawowe:
  - PUSH - zapis, umieszczenie elementu na szczycie stosu,
  - POP - odczyt, pobranie elementu z wierzchołka.
- Stos wykorzystywany jest bezpośrednio w architekturze procesorów do:
  - obsługi podprogramów (CALL/RET),
  - przechowywania adresów powrotu i rejestrów kontekstu,
  - implementacji struktur danych (rekurencja, odwracanie kolejności).
- W systemach sprzętowych stos realizowany jest często jako pamięć o adresowaniu z autoinkrementacją lub autodekrementacją.

LUT - Look Up Table

- Niewielka pamięć służąca do mapowania wejść na wyjścia w sposób zdefiniowany przez użytkownika.
- W architekturach FPGA LUT realizuje funkcję logiczną o n wejściach (np. 4-LUT, 6-LUT), gdzie każda kombinacja wejść wskazuje odpowiedni bit w tablicy.
- LUT pozwala zastąpić układ logiczny (np. bramki AND/OR/XOR) jednym blokiem pamięci ROM.
- W zastosowaniach DSP i grafiki LUT jest wykorzystywana do:
  - przyspieszania obliczeń nieliniowych (np. sin, log, gamma correction),
  - szybkiej translacji danych (np. mapowanie kolorów, korekcja tonów).
- Typowa organizacja LUT:
  adres wejściowy → zawartość tablicy → wynik logiczny lub wartość numeryczna.

5.6 Przykłady zastosowań pamięci

Mikrokontrolery

W systemach wbudowanych i mikrokontrolerach stosuje się różne rodzaje pamięci, zróżnicowane pod względem szybkości, trwałości i funkcji:

- **SRAM** - używana jako pamięć operacyjna (RAM) do przechowywania zmiennych, stosu (stack) oraz buforów komunikacyjnych w interfejsach UART, SPI, I^2C czy DMA. Charakteryzuje się bardzo szybkim dostępem i małą pojemnością.
- **Flash** - pełni rolę pamięci programu, w której zapisany jest kod wykonywalny mikrokontrolera oraz stałe tablice danych (np. *Look-Up Tables* dla funkcji sinus, CRC).
- **EEPROM / Flash emulowana** - wykorzystywana do przechowywania konfiguracji użytkownika oraz parametrów kalibracyjnych; zapewnia trwałość danych nawet po wyłączeniu zasilania.
- **FIFO** - stosowane w peryferiach komunikacyjnych (np. USART, CAN, USB) do kolejkowania ramek danych, synchronizacji transmisji i buforowania informacji między różnymi zegarami.
- **Cache / Flash Accelerator** - w mikrokontrolerach z rdzeniami ARM Cortex-M7/M33/MR usprawnia dostęp do pamięci Flash poprzez lokalne buforowanie instrukcji i danych (AXI-bus).

Komputery

W systemach komputerowych pamięć stanowi wielopoziomową hierarchię, zapewniającą równowagę między pojemnością a szybkością:

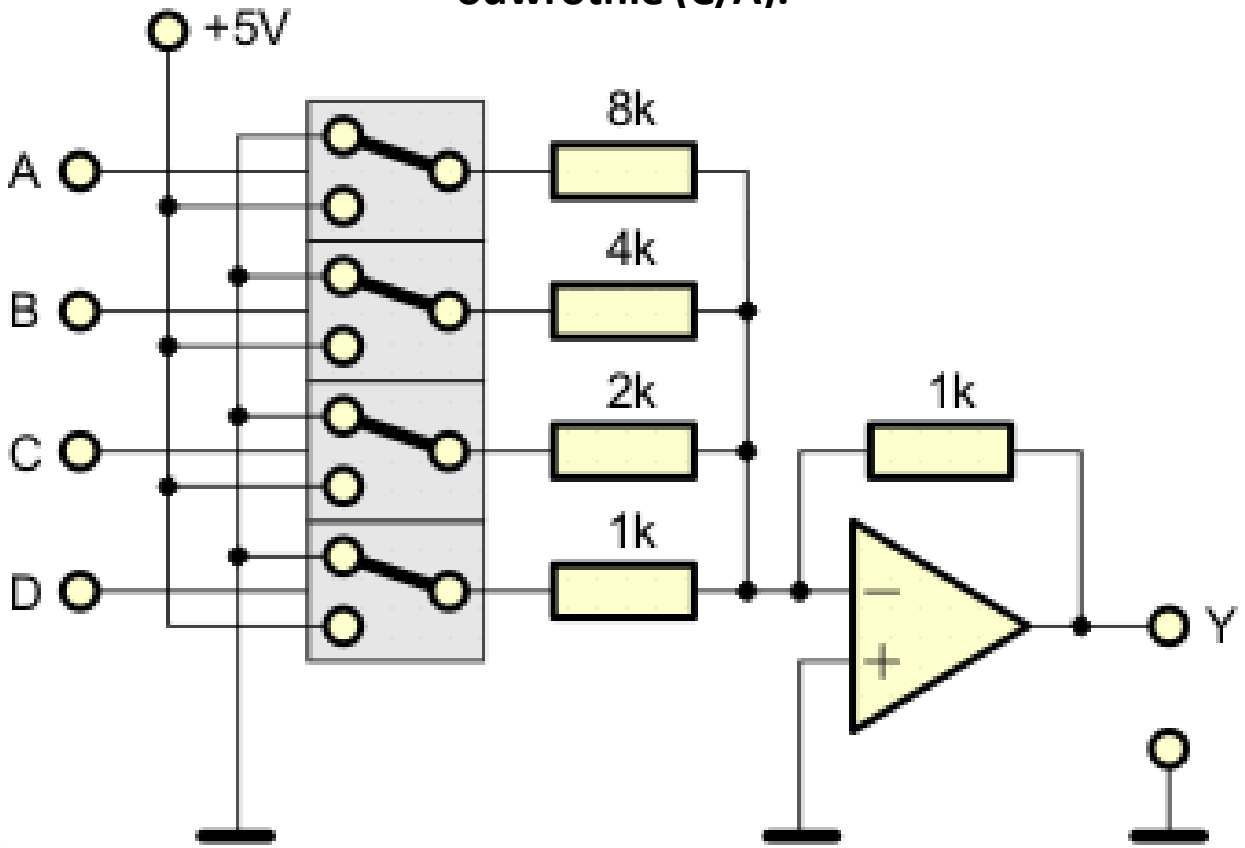
- **DRAM (DDR, DDR5, LPDDR)** - podstawowa pamięć operacyjna procesora, odpowiadająca za przechowywanie danych i kodu w czasie wykonywania programów.
- **SRAM (Cache L1, L2, L3)** - szybka pamięć podręczna wbudowana w procesor, minimalizująca opóźnienia dostępu do danych często używanych (TLB, instrukcje, rejestry).
- **GDDR / HBM (High Bandwidth Memory)** - wykorzystywana w GPU do obsługi grafiki, uczenia maszynowego i obliczeń równoległych; oferuje ekstremalną przepustowość przy dużym zużyciu energii.
- **SSD (Flash NAND, 3D XPoint)** - trwała pamięć masowa, służąca jako szybki nośnik danych, często z buforowaniem w DRAM.
- **FIFO / LIFO w systemach I/O** - stosowane w sterownikach urządzeń i stosach sieciowych do buforowania ramek, pakietów i kolejkowania operacji dyskowych.

FPGA

- Układy FPGA wykorzystują różne rodzaje pamięci do implementacji zarówno logiki kombinacyjnej, jak i pamięci tymczasowej dla danych i konfiguracji:
- **LUT (SRAM-based Look-Up Table)** - podstawowy element logiki programowalnej; realizuje funkcje logiczne poprzez odwzorowanie sygnałów wejściowych na wyjścia.
- **BRAM / URAM (Block / Ultra RAM)** - wewnętrzne bloki pamięci służące do przechowywania ramek wideo, buforów pakietów i danych strumieniowych.
- **Distributed RAM** - pamięć rozproszona w logice FPGA, wykorzystywana do przechowywania współczynników filtrów, rejestrów konfiguracji lub lokalnych zmiennych.
- **Asynchroniczne FIFO** - umożliwiają komunikację między różnymi domenami zegarowymi, zapewniając synchronizację przepływu danych bez utraty informacji.
- **CAM (Content Addressable Memory)** - pamięć adresowana zawartością; umożliwia błyskawiczne wyszukiwanie danych (np. tablice tras w routerach, ARP cache).
- **LIFO / stosy sprzętowe** - implementowane w akceleratorach FPGA do obsługi algorytmów rekursywnych, parsowania struktur danych lub backtrackingu.

6. Przetworniki A/C i C/A

Przetworniki służą do zamiany sygnałów analogowych na cyfrowe (A/C) oraz odwrotnie (C/A).



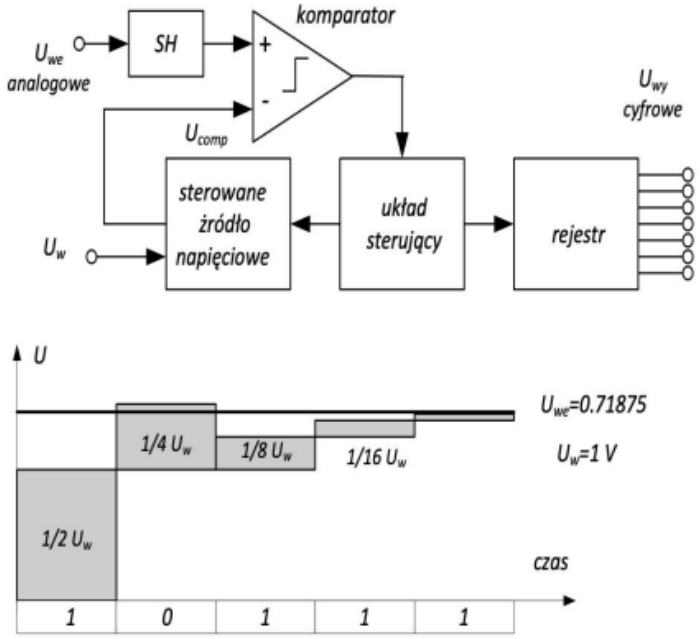
6.1 Przetworniki A/C

Przetworniki kompensacyjne realizują konwersję analogowo-cyfrową poprzez porównywanie napięcia wejściowego z napięciem wzorcowym, generowanym przez przetwornik C/A w pętli sprzężenia zwrotnego.
Wynik kolejnych porównań określa kolejne bity reprezentacji cyfrowej.

- **Równomierne (linearyzujące):**
  - Wartość wejściowa porównywana jest ze stopniowo zwiększanym napięciem odniesienia.
  - Czas konwersji jest stały, ale rośnie z rozdzielczością.
  - Stosowane w prostych multimetrów i systemach o niskiej częstotliwości próbkowania.
- **Nadążne (tracking ADC):**
  - Układ śledzi zmiany napięcia wejściowego, inkrementując lub dekrementując kod wyjściowy.
  - Szybko reaguje na zmiany, lecz ma ograniczoną precyzję.
  - Używany w układach kontroli procesów i regulatorach analogowych.

Wagowe – SAR (Successive Approximation Register):

- W każdej iteracji wybierany jest kolejny bit od MSB do LSB.
- Wykorzystuje przetwornik C/A oraz komparator do określenia, czy napięcie odniesienia jest większe czy mniejsze od napięcia wejściowego.
- Po n krokach uzyskuje się pełną konwersję n-bitową.
- Łączy wysoką szybkość działania i dokładność – najczęściej stosowany typ ADC w mikrokontrolerach i systemach pomiarowych.



6.2 Przetworniki C/A

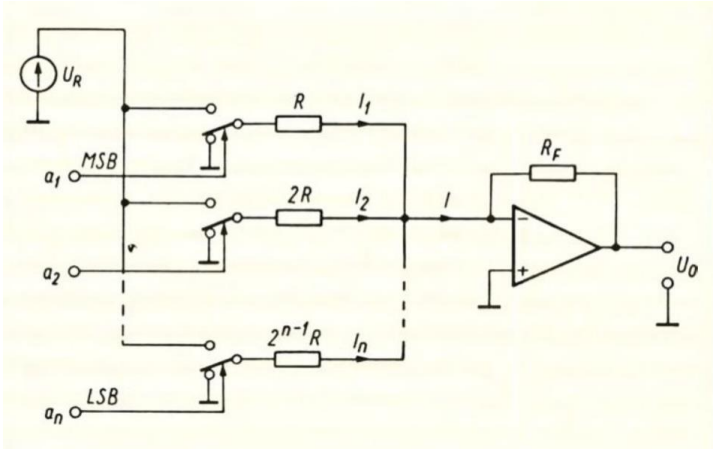
Przetwornik cyfrowo–analogowy (DAC) zamienia sygnał cyfrowy (ciąg bitów) na odpowiadające mu napięcie lub prąd analogowy.
Jest odwrotnością przetwornika A/C – realizuje operację *rekonstrukcji sygnału*.

Sieć rezystorowa (weighted resistor DAC)

- Każdy bit wejściowy steruje rezystorem o wartości odwrotnie proporcjonalnej do jego wagi binarnej (R, 2R, 4R, 8R...).

- Prądy z poszczególnych gałęzi są sumowane, tworząc napięcie wyjściowe:

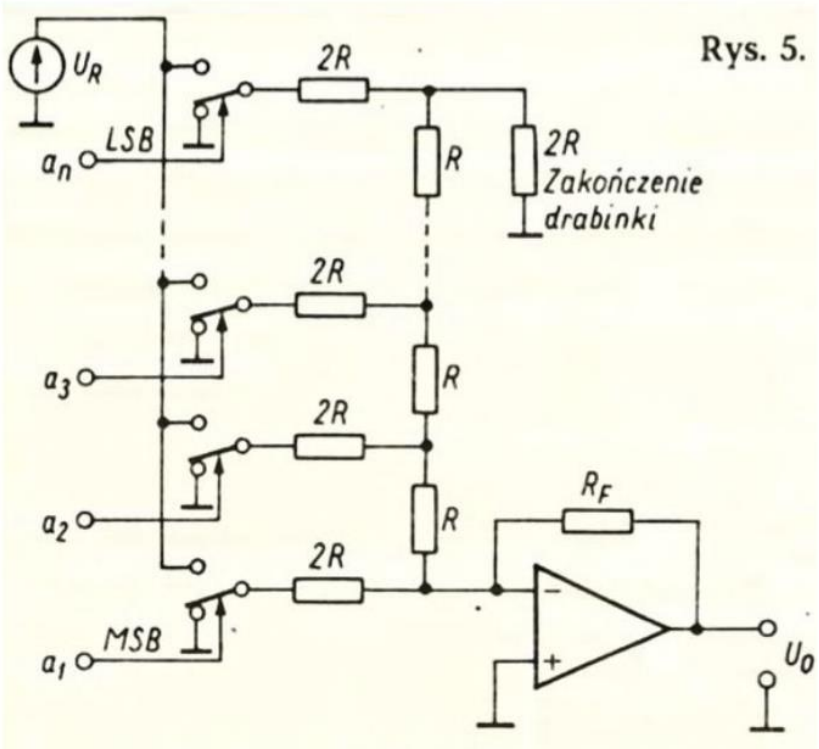
- $U_{OUT} = U_{REF} \times \frac{b_{n-1}}{2} + \frac{b_{n-2}}{4} + \ldots + \frac{b_0}{2^n}$



- Wadą jest konieczność użycia precyzyjnych rezystorów, co ogranicza liczbę bitów (typ. do 8–10).

Drabinka R–2R

- Składa się wyłącznie z dwóch wartości rezystancji: **R i 2R**.
- Struktura dzielników napięcia i sumatorów prądowych zapewnia dokładność i łatwość skalowania dla dowolnej liczby bitów.
- Każdy bit wejściowy steruje przełącznikiem, który dołącza odpowiedni poziom logiczny (0 lub Uref).
- Powszechnie stosowana w układach DAC w mikrokontrolerach, kartach dźwiękowych i sterownikach precyzyjnych.

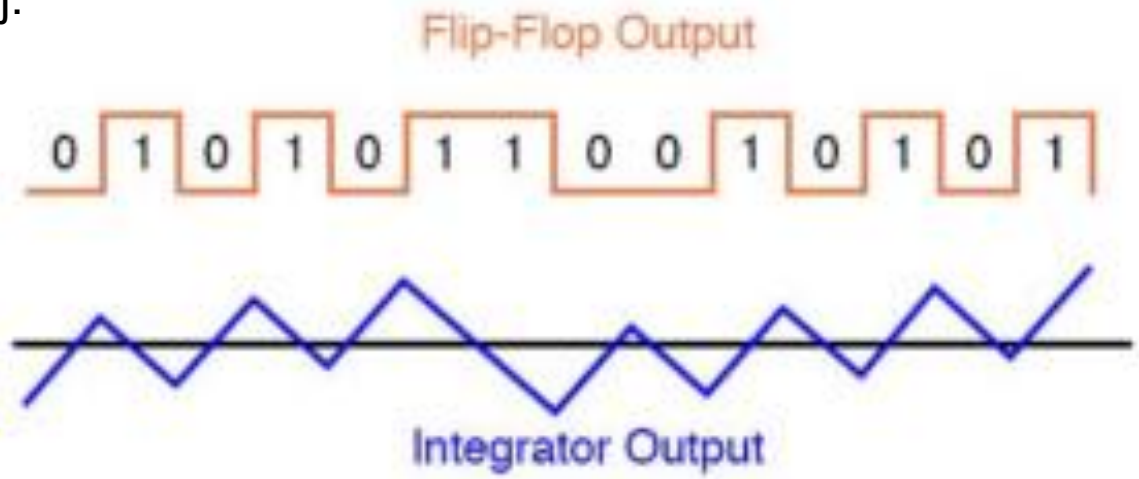


6.3 Przetworniki jednobitowy Sigma - Delta

Zaawansowany typ przetwornika A/C, charakteryzujący się bardzo wysoką rozdzielczością i precyzją pomiaru.
Jego działanie opiera się na zasadzie nadpróbkowania. W układzie Sigma–Delta sygnał wejściowy jest porównywany z jego wersją odtworzoną przez 1-bitowy przetwornik C/A w pętli sprzężenia zwrotnego.
W każdej chwili układ decyduje, czy wartość odtworzona jest większa czy mniejsza od wartości wejściowej — zapisując jedynie bit 0 lub 1.
W efekcie powstaje strumień jednobitowy, w którym gęstość bitów „1” odpowiada amplitudzie sygnału analogowego.
Taki strumień zostaje następnie przetworzony przez filtr cyfrowy o niskim paśmie przepustowym, który eliminuje szumy wysokoczęstotliwościowe, a następnie spowalniany do niższej częstotliwości próbkowania.
Ostatecznie uzyskana zostaje dokładna wartość cyfrowa, odpowiadająca amplitudzie sygnału wejściowego, przy bardzo wysokiej rozdzielczości (nawet 20–24 bity).

Zalety przetworników Sigma–Delta:
- Bardzo wysoka dokładność i liniowość,
- Niski poziom szumów kwantyzacji dzięki modulacji,
- Uproszczona struktura sprzętowa (1-bitowy modulator).

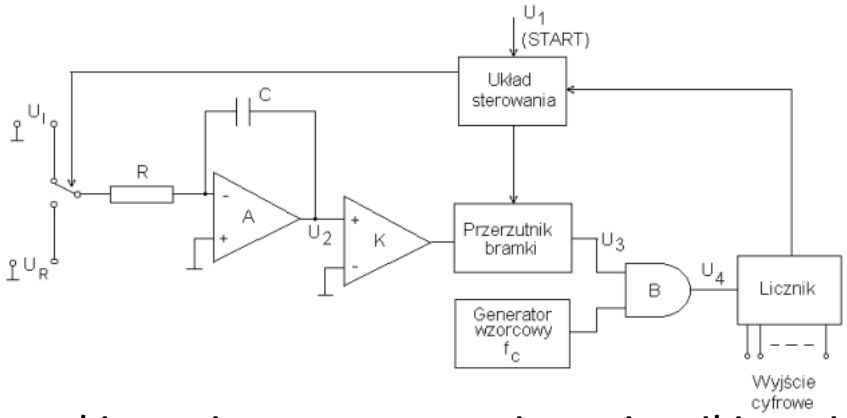
Zastosowania:
- Przetwarzanie dźwięku o wysokiej jakości (Hi-Fi, audio-DAC),
- Pomiar precyzyjny w czujnikach, wagach elektronicznych i systemach automatyki,
- Wysokorozdzielcze systemy pomiarowe w aparaturze medycznej i laboratoryjnej.



6.4 Przetwornik z podwójnym całkowaniem

Cechy charakterystyczne

- Wysoka odporność na zakłócenia i tętnienia sieciowe (np. 50/60 Hz).
- Bardzo dobra liniowość i dokładność konwersji.
- Niska prędkość działania – nie nadaje się do systemów czasu rzeczywistego.



Zastosowania

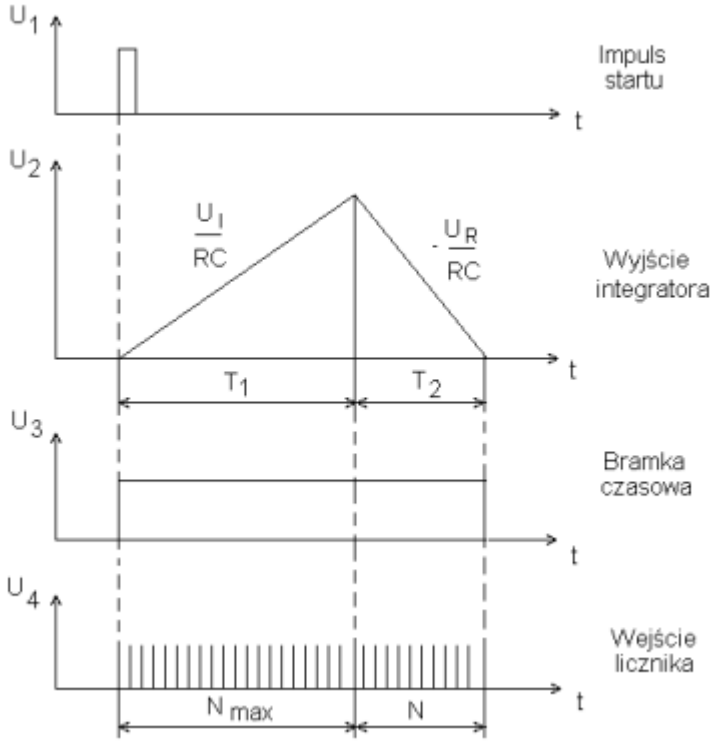
- Multimetry cyfrowe, mostki pomiarowe, precyzyjne mierniki napięcia.
- Układy pomiarowe w aparaturze przemysłowej i medycznej.
- Przetworniki w systemach o wysokiej dokładności i stabilności temperaturowej.

Proces konwersji przebiega w dwóch etapach:

- **Faza całkowania napięcia wejściowego (Integrate Phase)**
  - Przez określony czas T_1 kondensator jest ładowany prądem proporcjonalnym do napięcia wejściowego U_I.
  - Wzmacniacz operacyjny pracuje jako integrator, a napięcie na kondensatorze rośnie liniowo.
- **Faza całkowania napięcia odniesienia (Deintegrate Phase)**
  - Po czasie T_1 wejście przetwornika zostaje przełączone z sygnału U_I na stałe napięcie odniesienia U_R o przeciwnym znaku.
  - Kondensator rozładowuje się liniowo, aż napięcie osiągnie zero.
  - Czas rozładowania T_2 jest proporcjonalny do wartości napięcia wejściowego.

$$\boldsymbol{U_{IN} = U_{REF} \times \frac{T_2}{T_1}}$$

Pomiar czasu T_2) przy znanym U_{REF} i T_1(pozwala określić wartość sygnału wejściowego w postaci cyfrowej.



6.5 Parametry i błędy statyczne

Parametr	Oznaczenie	Jednostka	Znaczenie
Rozdzielczość	N	Bit/Bajt	Liczba poziomów kwantyzacji
Błąd kwantyzacji	±½ LSB	V	Dokładność odwzorowania sygnału
Błąd przesunięcia	EO	V lub LSB	Przesunięcie osi Y charakterystyki
Błąd wzmocnienia	EG	% lub LSB	Zmiana nachylenia charakterystyki
Nieliniowość całkowa	±INL [LSB]	LSB	Odchylenie od idealnej linii
Nieliniowość różniczkowa	±DNL [LSB]	LSB	Różnica między kolejnymi krokami

Rozdzielczość (Resolution)

Określa liczbę bitów wyjściowych przetwornika (np. 8, 12, 16 bitów).

- Im większa rozdzielczość, tym więcej poziomów kwantyzacji: $L = 2^N$
- Dla pełnego zakresu napięcia U_{FS} ,najmniejszy krok napięcia (LSB – *Least Significant Bit*) wynosi: $U_{LSB} = \frac{U_{FS}}{2^N}$

Błąd kwantyzacji (Quantization Error)

Powstaje w wyniku zaokrąglenia sygnału wejściowego do najbliższego poziomu dyskretnego.

- Maksymalny błąd kwantyzacji wynosi: $\pm\frac{1}{2}LSB$
- Stosunek sygnału do szumu kwantyzacji (SNR) można oszacować:
- $SNR = 6{,}02 \cdot N + 1{,}76\ [dB]$
- Oznacza to, że każdy dodatkowy bit rozdzielczości zwiększa SNR o ~6 dB.

Błąd przesunięcia (Offset Error)

- Różnica pomiędzy rzeczywistym napięciem wyjściowym przy kodzie „zero” a wartością idealną.
- Powoduje równoległe przesunięcie charakterystyki przetwarzania.
- Wynika z błędów napięcia odniesienia i niezerowego napięcia wejściowego komparatora.

Błąd wzmocnienia (Gain Error)

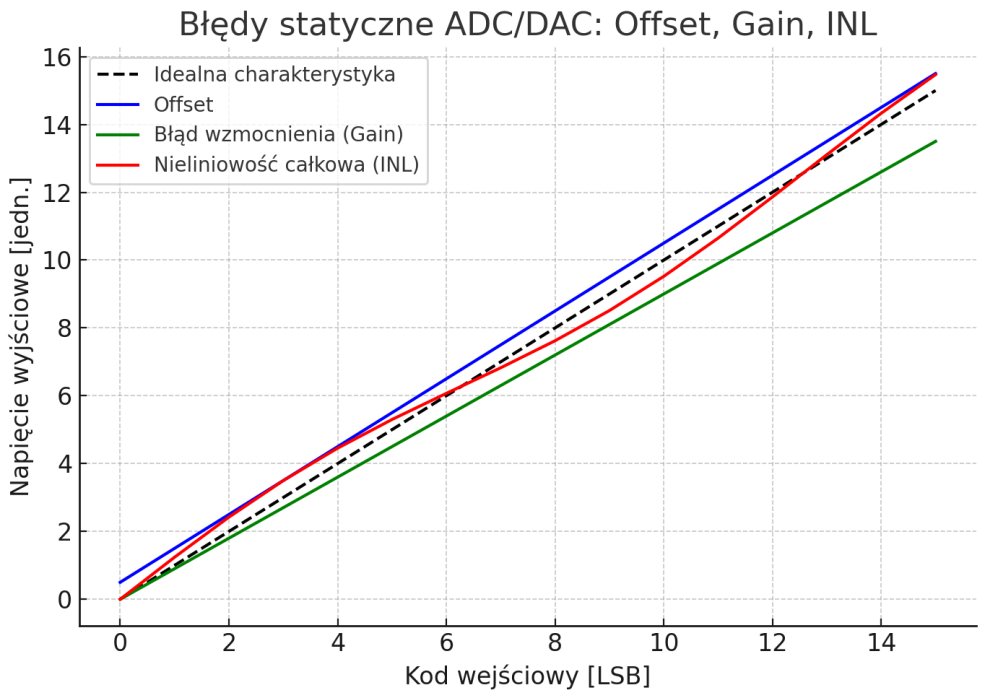
- Różnica między rzeczywistym nachyleniem charakterystyki a idealnym wzmocnieniem.
- Powoduje rozszerzenie lub skurczenie zakresu napięć wyjściowych.
- Może być kompensowany programowo lub przez kalibrację układu odniesienia.

Nieliniowość całkowa (INL – Integral Nonlinearity)

- Maksymalne odchylenie rzeczywistej charakterystyki przetwornika od idealnej linii prostej.
- Ocenia **ogólną dokładność przetwarzania** w całym zakresie kodów.
- Wyrażana w jednostkach LSB lub procentach zakresu pełnego.

Nieliniowość różniczkowa (DNL – Differential Nonlinearity)

- Mierzy różnice między rzeczywistą a idealną szerokością kroku kwantyzacji.
- DNL > 1 LSB może powodować brak monotoniczności (spadek napięcia mimo wzrostu kodu).
- Dla idealnego przetwornika DNL = 0.



6.6 Parametry dynamiczne

A/C – parametry czasowe

- **Czas konwersji** t_{conv}: od rozpoczęcia próbkowania do gotowego kodu.
  Przepustowość:

$$f_{\text{throughput}} = \frac{1}{t_{\text{acq}} + t_{\text{conv}} + t_{\text{latch}}}$$

- **Opóźnienie (latencja)**: w ADC potokowych $t_{lat} = \frac{L}{f_s}$ (L – liczba etapów).
- **Czas akwizycji** t_{acq} i **apertury** t_{ap}: czas potrzebny układowi S/H na uchwycenie próbki.
- **Jitter apertury** t_j: ogranicza SNR przy wysokich częstotliwościach wejścia:

$$SNR_j \approx -20log_{10}(2\pi f_{\text{in}} t_j) \text{ [dB]}$$

A/C – pasmo

- **Nyquist**: $f_s \geq 2f_{\text{max}}$.
- **Pasmo wejściowe** układu S/H:
  - **Small-signal BW** (−3 dB dla małych sygnałów),
  - **Full-power BW** (pasmo dla pełnej amplitudy bez nadmiernych zniekształceń).
- **Metryki dynamiczne**:

$$ENOB = \frac{SINAD - 1.76}{6.02}$$

(pośrednio zależne od pasma, jitteru i zniekształceń).

C/A – parametry czasowe

- Czas ustalania $t_{settling}$: do np. 0.5 LSB po skoku kodu.
- Slew rate: maksymalna szybkość narastania wyjścia.
- Glitch impulse: krótkie zaburzenie przy zmianie kodu (zwłaszcza przy przejściu przez połowę skali).

C/A – pasmo

- Pasmo wyjściowe (−3 dB) oraz SFDR (czystość widma).
- Filtr rekonstrukcyjny ogranicza aliasing; efektywne pasmo $\lesssim f_s/2$)dla ZOH).

Powiązania praktyczne

- Krótsze t_{acq} ,t_{conv} ⇒większa przepustowość, ale zwykle mniejsze ENOB.
- Duże f_{in} +jitter ⇒ gwałtowny spadek SNR.
- Wybór ADC/DAC dobiera się do: pasma sygnału, wymaganego ENOB/SFDR i dopuszczalnej latencji.

SINAD	sygnał / (szum + zniekształcenia)	rzeczywisty wskaźnik jakości
ENOB	przeliczenie SINAD na bity	ocenia użyteczną rozdzielczość

6.7 Zastosowania

Zastosowanie	Zakres częstotliwości próbkowania (f_s)	Rozdzielczość (bit)	Typ przetwornika	Przykłady zastosowań
Pomiar i czujniki precyzyjne	1 Hz – 10 kHz	16–24	Sigma–Delta (ΣΔ), Dual-Slope	Wagi elektroniczne, multimetry, czujniki temperatury i ciśnienia
Audio i akustyka	44,1 – 192 kHz	16–24	Sigma–Delta, SAR	Karty dźwiękowe, miksery, przetwarzanie mowy, mikrofony MEMS
Sterowanie i automatyka	1 – 500 kHz	8–12	SAR, Flash	Mikrokontrolery, napędy silników, systemy PLC, robotyka
Systemy wideo i obrazowania	10 – 100 MHz	8–12	Pipeline, Flash	Kamery cyfrowe, skanery, przetworniki obrazów
Łączność i RF (radio frequency)	100 MHz – 10 GHz	6–10	Pipeline, Time-Interleaved, Flash	SDR (Software Defined Radio), radar, Wi-Fi, LTE, 5G
Szybkie DAC	10 MHz – 5 GHz	6–14	Current-Steering, R-2R, Segmented	Generatory sygnałów, nadajniki, przetwarzanie w.cz.

Parametry częstotliwości próbkowania (f_s (i rozdzielczości bitowej (N) bezpośrednio wpływają na pasmo, dokładność i zastosowanie przetwornika analogowo–cyfrowego (ADC) lub cyfrowo–analogowego (DAC).
Dobór odpowiednich wartości stanowi kompromis między szybkością przetwarzania, dokładnością pomiaru oraz kosztem i poborem mocy.

Zależność między rozdzielczością a szybkością

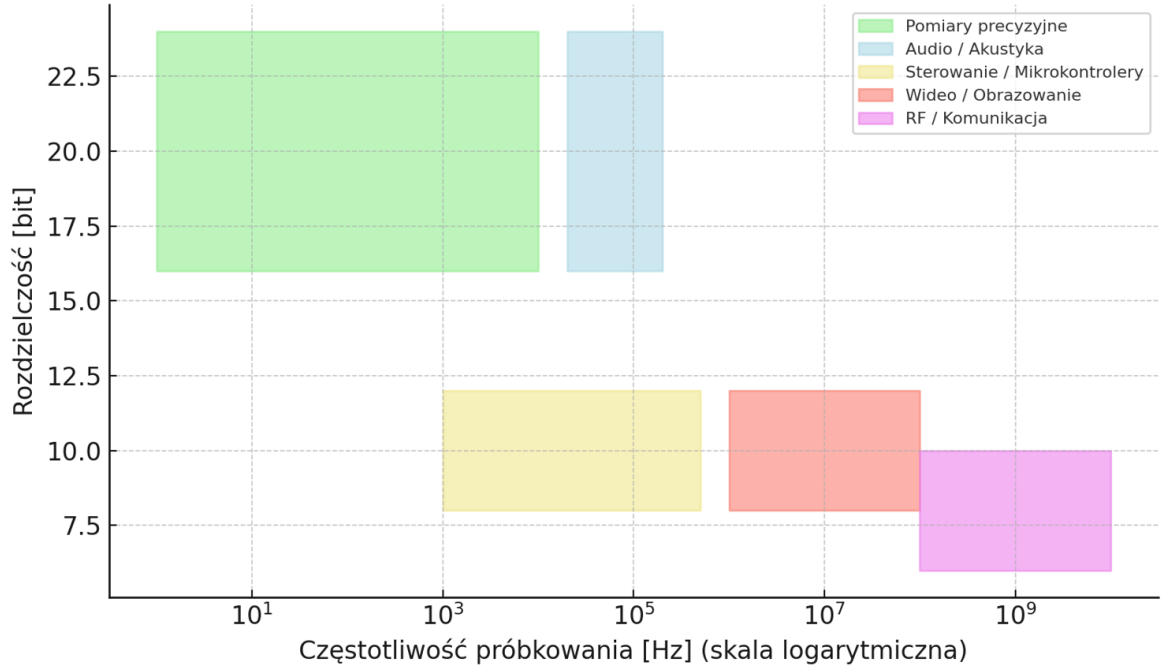
- Im większa rozdzielczość (N), tym większy czas konwersji i mniejsza częstotliwość próbkowania f_s.
- Wysoka rozdzielczość wymaga większej precyzji toru analogowego, niższego szumu i mniejszych błędów nieliniowości.
- Z kolei wysokie częstotliwości próbkowania (GHz) ograniczają liczbę użytecznych bitów (typowo 6–10 bitów w bardzo szybkich ADC).

Trendy technologiczne

- Sigma–Delta dominuje w aplikacjach niskoczęstotliwościowych, gdzie kluczowa jest dokładność i liniowość.
- SAR (Successive Approximation) to kompromis między dokładnością a szybkością – najczęściej spotykany w mikrokontrolerach.
- Pipeline i Flash ADC stosuje się tam, gdzie liczy się minimalna latencja i szybka konwersja (RF, wideo).
- Time-Interleaved ADC osiągają pasma setek MHz dzięki równoległemu próbkowaniu wieloma kanałami.

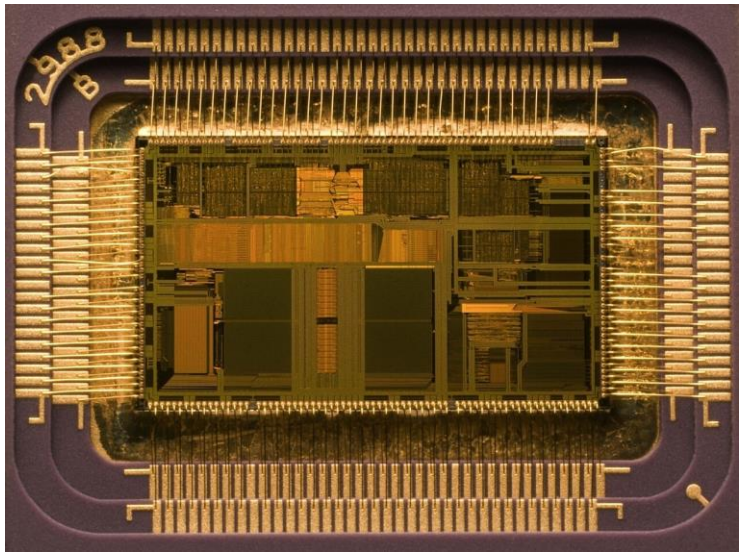
Wnioski

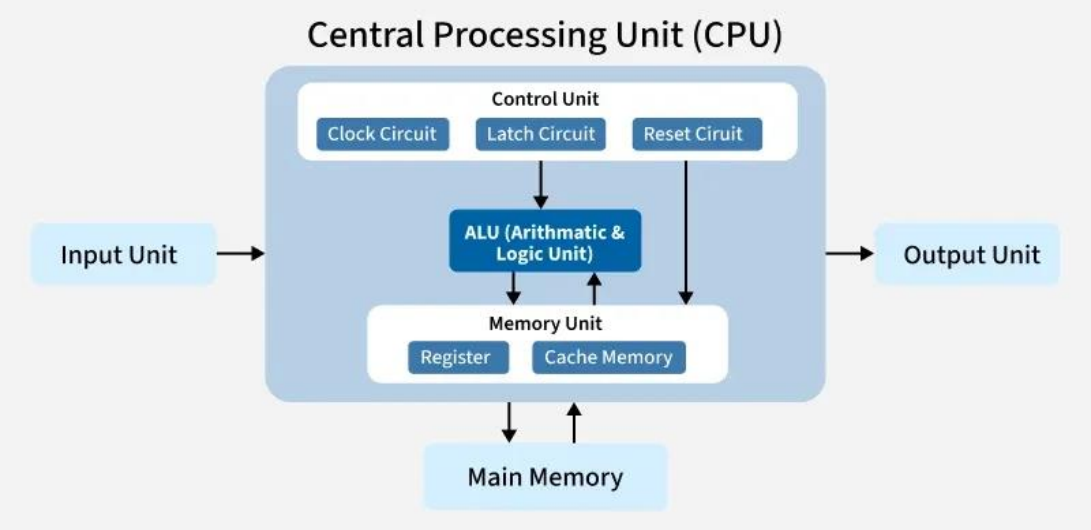
- Wysoka rozdzielczość = dokładność kosztem szybkości.
- Wysoka częstotliwość próbkowania = szybkość kosztem bitów efektywnych (ENOB).
- Ostateczny wybór ADC/DAC zależy od:
  - charakterystyki sygnału (DC, AC, impulsowy, szumowy),
  - dopuszczalnego błędu,
  - dostępnych zasobów sprzętowych i mocy obliczeniowej.



7. Architektura procesora i CPU

Jednostka centralna (CPU – Central Processing Unit) stanowi podstawowy element komputera, odpowiedzialny za realizację instrukcji programowych.
Procesor pobiera dane z pamięci operacyjnej, przetwarza je zgodnie z kodem programu, a następnie zapisuje wyniki z powrotem do pamięci lub urządzeń peryferyjnych, jednocześnie koordynując działanie wszystkich pozostałych podzespołów systemu.





- **Jednostka arytmetyczno-logiczna (ALU**) realizuje podstawowe operacje matematyczne i logiczne na danych, takie jak dodawanie, porównywanie czy przesunięcia bitowe.
- **Jednostka sterująca (CU)** odpowiada za dekodowanie instrukcji programu oraz zarządzanie przepływem danych pomiędzy poszczególnymi blokami procesora.
- **Rejestry** stanowią ultraszybką pamięć wewnętrzną, wykorzystywaną do przechowywania tymczasowych wyników obliczeń i adresów operacji.
- **Pamięć podręczna (cache)** służy do buforowania najczęściej używanych danych i instrukcji, znacząco redukując opóźnienia dostępu do pamięci operacyjnej (RAM).

Architektura von Neumanna i Harvard

- W architekturze von Neumanna procesor korzysta ze wspólnej pamięci oraz jednej magistrali do obsługi zarówno instrukcji, jak i danych. Takie rozwiązanie upraszcza konstrukcję systemu i jego implementację, jednak powoduje tzw. „wąskie gardło von Neumanna”, sytuację w której procesor musi kolejkować operacje odczytu instrukcji i danych, ponieważ nie może pobierać ich jednocześnie.
  Mimo tej ograniczonej równoległości, model ten pozostaje podstawowym i najpowszechniej stosowanym rozwiązaniem w klasycznych komputerach ogólnego przeznaczenia.
- Z kolei architektura Harvard rozdziela pamięć oraz magistrale dla instrukcji i danych, umożliwiając równoczesny dostęp do obu typów informacji. Dzięki temu procesor może wykonywać operacje szybciej i efektywniej, co znacząco poprawia wydajność. W nowoczesnych mikroprocesorach i mikrokontrolerach stosuje się często modyfikowaną wersję architektury Harvard, łączącą prostotę modelu von Neumanna z równoległością i przepustowością Harvard.

ISA (Instruction Set Architecture) określa zestaw instrukcji, które procesor potrafi rozpoznać i wykonać, a także sposób adresowania danych i organizację rejestrów. ISA stanowi interfejs między sprzętem a oprogramowaniem, definiując język, w którym komunikują się programy i procesor.

Wyróżnia się dwa główne podejścia:

- **CISC (Complex Instruction Set Computer)** – architektura o złożonych instrukcjach, zdolnych wykonywać wiele operacji w ramach jednej komendy (np. obliczenia + dostęp do pamięci).
  Przykład: rodzina procesorów x86.
  Zalety: mniejszy rozmiar kodu, łatwiejsze tłumaczenie z języków wysokiego poziomu, kompatybilność wsteczna.
  Wady: większa złożoność dekodowania i niższa efektywność potokowania.

- **RISC (Reduced Instruction Set Computer)** – architektura oparta na prostych, jednocyklicznych instrukcjach, które można szybko dekodować i przetwarzać w potoku.
  Przykład: ARM, MIPS, RISC-V.
  Zalety: większa efektywność energetyczna, lepsza współpraca z pipeline i superskalarnością.
  Wady: większy rozmiar kodu wynikowego (więcej prostych instrukcji potrzebnych do tego samego zadania).

Współczesne procesory często wykorzystują elementy obu koncepcji – np. architektura x86 (CISC) wewnętrznie przekształca instrukcje w mikrooperacje RISC, łącząc kompatybilność z wydajnością nowoczesnych potoków.

7.1 ALU – jednostka arytmetyczno-logiczna

Podstawowy blok funkcjonalny procesora odpowiedzialny za wykonywanie operacji matematycznych i logicznych na danych binarnych.
Wykonuje między innymi:

- operacje arytmetyczne: dodawanie, odejmowanie, mnożenie, dzielenie, inkrementację i dekrementację,
- operacje logiczne: AND, OR, XOR, NOT, porównania, przesunięcia bitowe.
- ALU współpracuje z rejestrami procesora, które dostarczają dane wejściowe i przechowują wyniki obliczeń, oraz z jednostką sterującą (CU), która przekazuje jej sygnały kontrolne i kody operacji.
  W bardziej złożonych architekturach ALU może być podzielona na kilka równoległych jednostek, umożliwiających równoczesne przetwarzanie wielu danych (pipeline, SIMD, superskalarność).

Pipeline to technika organizacji pracy procesora, w której różne etapy wykonania instrukcji (np. pobranie, dekodowanie, wykonanie, zapis wyniku) są realizowane równolegle dla różnych instrukcji.
Zamiast czekać, aż jedna instrukcja całkowicie się zakończy, procesor nakłada etapy wielu instrukcji na siebie, co zwiększa przepustowość. Każdy etap potoku działa w jednym takcie zegara – po n cyklach można mieć wyniki dla n instrukcji.

- Zalety: znaczny wzrost wydajności przy zachowaniu prostoty architektury.
- Wady: opóźnienia potoku (*pipeline stalls*) przy skokach warunkowych lub zależnościach danych.

SIMD to technika równoległego przetwarzania danych, w której jedna instrukcja wykonuje tę samą operację na wielu elementach danych jednocześnie.
Wykorzystywana jest głównie w obliczeniach numerycznych, grafice, przetwarzaniu sygnałów i multimediach. Instrukcja „dodaj” może jednocześnie dodać:
(a1 + b1, a2 + b2, a3 + b3, a4 + b4), zamiast czterech oddzielnych operacji.

Implementacje: Intel SSE / AVX, ARM NEON, RISC-V Vector Extension.

Zalety: ogromny wzrost wydajności przy przetwarzaniu dużych bloków danych (np. pikseli, próbek audio).
Wady: wymaga danych jednorodnych (ten sam typ i rozmiar).

Architektura **superskalarna** umożliwia równoczesne wykonanie wielu instrukcji w jednym cyklu zegara. Procesor posiada wiele jednostek wykonawczych (ALU, FPU, LSU), a jednostka sterująca dynamicznie rozdziela instrukcje między nimi.

- Dynamiczne planowanie – CPU analizuje zależności między instrukcjami, by wykonać kilka jednocześnie.
- Out-of-Order Execution – wykonywanie instrukcji w innej kolejności niż w kodzie, jeśli pozwala to uniknąć przestojów.
- Register Renaming – zapobiega konfliktom na tych samych rejestrach.
- Zalety: maksymalne wykorzystanie zasobów procesora, zwiększenie liczby operacji na takt (IPC).
  Wady: złożoność układu sterowania, większy pobór mocy.

W skrócie*:*

- **Pipeline** - zwiększa wydajność przez podział instrukcji na etapy,
- **SIMD** - przetwarza wiele danych jedną instrukcją,
- **Superskalarność** - pozwala na jednoczesne wykonywanie wielu instrukcji.

7.2 Rejestry CPU

Rejestry to najszybsze elementy pamięci w strukturze procesora.
Służą do przechowywania danych tymczasowych, adresów pamięci, liczników oraz wyników obliczeń wykonywanych przez jednostkę arytmetyczno–logiczną (ALU).
Dzięki bezpośredniemu połączeniu z ALU rejestry umożliwiają wykonywanie operacji w jednym cyklu zegara, co znacznie przyspiesza przetwarzanie danych.

Rejestry ogólnego przeznaczenia (GPR – General Purpose Registers)

- Używane przez programistę lub kompilator do przechowywania dowolnych danych i adresów.
- Mogą pełnić różne funkcje w zależności od architektury procesora (np. x86, ARM, RISC-V).
- Przykłady:
  - AX, BX, CX, DX – klasyczne rejestry ogólnego przeznaczenia w architekturze x86.
  - R0–R15 – rejestry w architekturze ARM, z których część ma zastosowanie uniwersalne, a część specjalne (np. R13 – stos, R15 – licznik rozkazów).
  - W architekturach RISC liczba rejestrów jest zwykle większa (np. 32–64), aby zminimalizować dostęp do pamięci.

Zaleta: operacje na rejestrach są kilkaset razy szybsze niż na pamięci RAM.

Rejestry specjalnego przeznaczenia (SPR – Special Purpose Registers)

Rejestry tej grupy służą do kontroli pracy procesora, zarządzania przerwaniami, obsługi stosu oraz synchronizacji potoku instrukcji.

Typowe rejestry specjalne:

- PC (Program Counter) – licznik rozkazów; wskazuje adres następnej instrukcji do wykonania.
- SP (Stack Pointer) – wskaźnik stosu; przechowuje adres szczytu stosu.
- IR (Instruction Register) – przechowuje aktualnie wykonywaną instrukcję.
- SR / PSR (Status Register / Program Status Register) – zawiera flagi stanu procesora, takie jak:
  - Z (Zero) – wynik operacji = 0,
  - C (Carry) – przeniesienie w operacji arytmetycznej,
  - N (Negative) – wynik ujemny,
  - V (Overflow) – przepełnienie arytmetyczne,
  - I (Interrupt enable) – flaga zezwolenia na przerwania.
- MAR (Memory Address Register) – przechowuje adres komórki pamięci, do której CPU ma uzyskać dostęp.
- MDR (Memory Data Register) – przechowuje dane pobrane lub wysyłane do pamięci.

- **Organizacja rejestrów w nowoczesnych procesorach**
- W procesorach **RISC** liczba rejestrów jest większa, aby zmniejszyć zależność od pamięci (np. 32–64 GPR).
- W **CISC (x86)** część rejestrów ma **sztywno przypisane funkcje**, ale w nowoczesnych wersjach (x86-64) wiele z nich pełni funkcje ogólne.
- W **architekturach superskalarnych** istnieją również **rejestry renamowane** (*Register Renaming*) – wirtualne, dynamicznie przydzielane, zapobiegające konfliktom przy równoległym wykonywaniu instrukcji.

7.3 Stos i podprogramy

- **Stos (Stack) – definicja i zasada działania**
- **Stos** to specjalna struktura danych wykorzystywana przez procesor do przechowywania tymczasowych informacji, takich jak adresy powrotu z podprogramów, zmienne lokalne czy rejestry robocze.
  Działa według zasady **LIFO (Last In, First Out)** — ostatni element, który został zapisany, jest odczytywany jako pierwszy.
- **Operacje podstawowe:**
- **PUSH** – zapis danych na stos,
- **POP** – odczyt i usunięcie danych ze stosu.
- Stos jest zarządzany przez specjalny rejestr – **SP (Stack Pointer)**, który wskazuje bieżący wierzchołek stosu w pamięci.

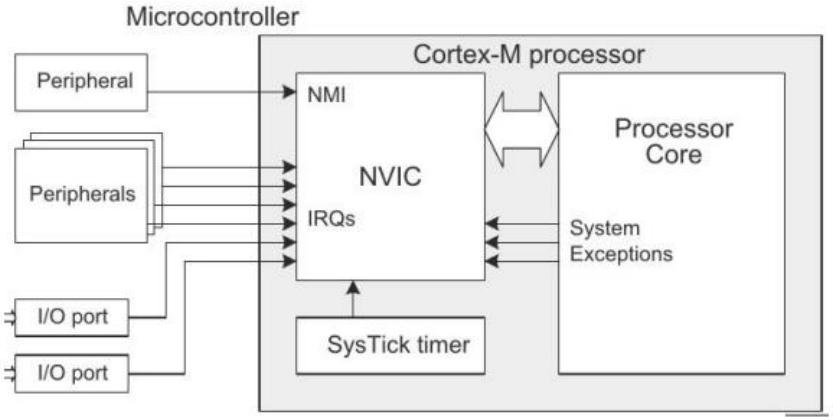
Podprogram (procedura, funkcja) to fragment kodu wykonywalnego, który realizuje określone zadanie i może być wielokrotnie wywoływany z różnych miejsc programu. Pozwala to na modularność, oszczędność pamięci i czytelność kodu.

Etapy wywołania podprogramu:

- Zapis adresu powrotu (PC) na stos – aby wiedzieć, dokąd wrócić po wykonaniu funkcji.
- Przekazanie parametrów – przez rejestry lub stos.
- Wykonanie kodu funkcji (ciała podprogramu).
- Zapis wyników – do rejestrów lub pamięci.
- Instrukcja **RET / RETURN** – pobranie adresu powrotu ze stosu i wznowienie programu głównego.

7.4 Kontroler przerwań i DMA

Przerwanie (Interrupt) to sygnał powodujący natychmiastową zmianę bieżącego przepływu programu w celu obsługi zdarzenia wymagającego reakcji procesora. Dzięki temu CPU może reagować asynchronicznie na sygnały z urządzeń peryferyjnych lub zdarzenia wewnętrzne, bez potrzeby ciągłego sprawdzania ich stanu (tzw. *pollingu*).



Rodzaje przerwań:

- Zewnętrzne (hardware) – wywoływane przez urządzenia zewnętrzne, np. klawiaturę, timer, interfejs komunikacyjny.
- Wewnętrzne (software) – generowane przez instrukcje programu, np. dzielenie przez zero, pułapki (*traps*).
- Maskowalne (IRQ) – można je tymczasowo zablokować.
- Niemaskowalne (NMI) – mają najwyższy priorytet, np. błędy sprzętowe.

Kontroler przerwań to układ zarządzający zgłoszeniami przerwań pochodzącymi z różnych źródeł i decydujący, które z nich mają zostać obsłużone w danym momencie.

Główne funkcje kontrolera:

- **Arbitraż priorytetów** – określa kolejność obsługi wielu przerwań.
- **Maskowanie** – możliwość zablokowania wybranych przerwań.
- **Wektorowanie** – automatyczne przekazanie numeru (adresu) funkcji obsługi (ISR – *Interrupt Service Routine*).
- **Potwierdzenie przerwania (acknowledge)** – sygnał informujący urządzenie, że jego żądanie zostało przyjęte.

Przykłady:

- **Intel 8259 PIC (Programmable Interrupt Controller)** – klasyczny układ dla systemów x86.
- **NVIC (Nested Vectored Interrupt Controller)** – zaawansowany kontroler zagnieżdżonych przerwań w rdzeniach ARM Cortex-M.

DMA (Direct Memory Access) to mechanizm umożliwiający bezpośrednią wymianę danych między urządzeniami peryferyjnymi a pamięcią, bez udziału CPU przy każdej operacji.

- CPU inicjuje transfer, ustawiając parametry w kontrolerze DMA (adres źródła, celu, długość).
- Kontroler DMA przejmuje kontrolę nad magistralą danych.
- Przesyła blok danych z pamięci do urządzenia (lub odwrotnie).
- Po zakończeniu transferu zgłasza przerwanie do CPU, informując o zakończeniu operacji.
- **Zalety DMA:**
- Zmniejsza obciążenie procesora – CPU może w tym czasie wykonywać inne zadania.
- Umożliwia transfery blokowe z dużą przepustowością.
- Minimalizuje opóźnienia w transmisjach (np. audio, wideo, sieć).

Burst mode	Cały blok danych przesyłany jednorazowo – CPU zatrzymane na czas transferu.
Cycle stealing	DMA przesyła pojedyncze bajty między cyklami CPU, współdzieląc magistralę.
Transparent mode	DMA działa w tle, gdy CPU nie korzysta z magistrali – pełna współpraca bez zakłóceń.

DMA i przerwania często współpracują:

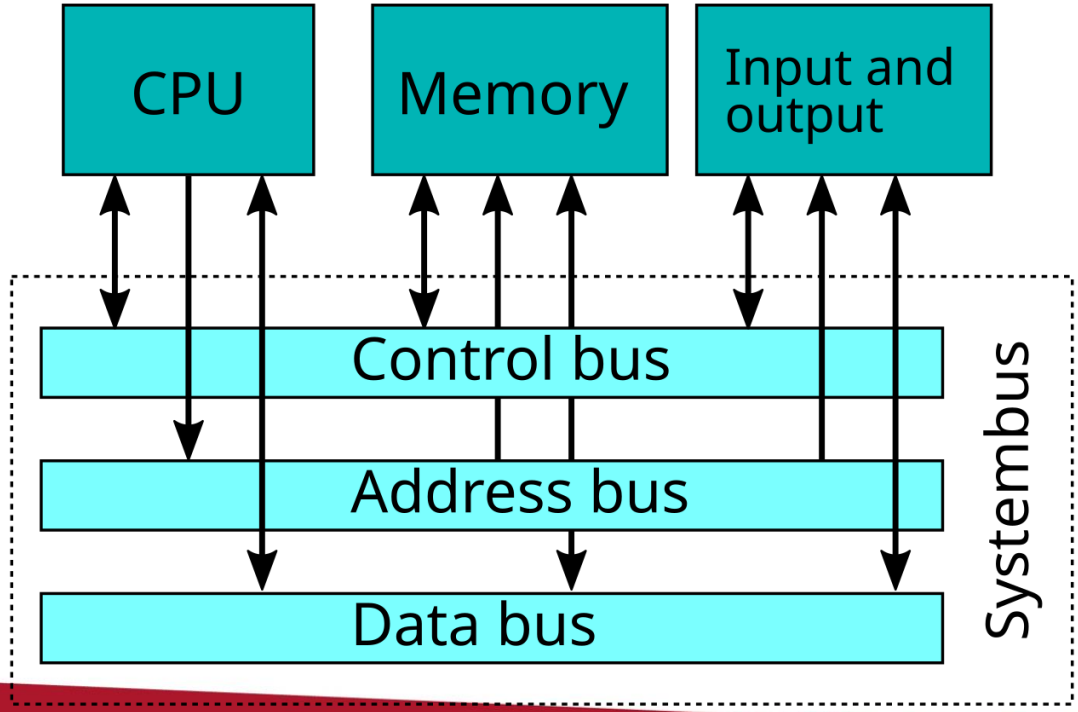
- Po zakończeniu transferu DMA generuje przerwanie, sygnalizując gotowość danych.
- CPU obsługuje to przerwanie i wykonuje dalsze operacje (np. analiza, przetwarzanie, zapis).
- W systemach real-time DMA pozwala zachować ciągłość transmisji danych bez strat i przeciążeń CPU.

W skrócie:

- Kontroler przerwań koordynuje obsługę zdarzeń z wielu źródeł,
- DMA pozwala na szybkie, autonomiczne przesyłanie danych między pamięcią a urządzeniami,
- Wspólnie umożliwiają wydajne i wielozadaniowe działanie systemu mikroprocesorowego.

7.5 Magistrale systemowe i peryferyjne

- Magistrala komunikacyjna, magistrala (ang. bus) – zespół linii przenoszących sygnały oraz układów wejścia-wyjścia służących do przesyłania sygnałów między połączonymi urządzeniami w systemach mikroprocesorowych. Łączą główne komponenty komputerea, CPU, pamięć RAM/ROM oraz mostki komunikacyjne. Zapewniają wysoką przepustowość i niezawodność transmisji.



Przykłady magistral systemowych:

- **Front-Side Bus (FSB)** – klasyczna magistrala łącząca CPU z kontrolerem pamięci (chipsetem).
- **QuickPath Interconnect (QPI)** – nowoczesna magistrala w procesorach Intel, umożliwiająca wielordzeniową komunikację.
- **HyperTransport** – odpowiednik magistrali systemowej w architekturze AMD.
- **AXI / AHB / APB (AMBA)** – standard magistral wbudowanych w mikrokontrolery ARM, łączący rdzeń CPU, pamięć i peryferia.

Przykłady magistral peryferyjnych:

Magistrala	Charakterystyka	Przepustowość / Zastosowanie
I^2C (Inter-Integrated Circuit)	Szeregowa, 2 linie (SDA, SCL), wiele urządzeń	Kilkaset kbit/s – sensory, RTC
SPI (Serial Peripheral Interface)	Szybka, pełnodupleksowa, 4 linie (MOSI, MISO, SCK, SS)	Do kilkudziesięciu Mbit/s – pamięci Flash, ADC
UART (Universal Asynchronous Receiver-Transmitter)	Transmisja asynchroniczna, 2 linie (TX, RX)	Do kilku Mbit/s – RS-232, Bluetooth
USB (Universal Serial Bus)	Standard komputerowy, hierarchiczny	Do 40 Gbit/s (USB4) – urządzenia zewnętrzne
PCI / PCIe (Peripheral Component Interconnect)	Szybka magistrala dla kart rozszerzeń	PCIe 5.0: do 64 GB/s (x16)
CAN (Controller Area Network)	Magistrala odporna na zakłócenia	Systemy motoryzacyjne, przemysłowe
Ethernet	Sieciowa magistrala komunikacyjna	10 Mbit/s – 400 Gbit/s

- W systemach z wieloma urządzeniami dostęp do magistrali wymaga **mechanizmu arbitrażu**, który przydziela prawo do transmisji jednemu z uczestników.
- **Scentralizowany** – jeden *bus arbiter* zarządza dostępem (np. AHB).
- **Rozproszony** – urządzenia samodzielnie uzgadniają priorytety (np. CAN, Ethernet CSMA/CD).
- **Hierarchiczny** – wiele poziomów magistral połączonych mostkami (*bridges*).

Topologia	Charakterystyka	Przykład zastosowania
Szyna (Bus)	Wspólne linie danych i adresów	AMBA, PCI
Gwiazda (Star)	Centralny kontroler (hub/switch)	USB, Ethernet
Pierścień (Ring)	Sygnał przesyłany w pętli	Token Ring, niektóre systemy SoC
Siatka (Mesh)	Połączenia punkt–punkt	Nowoczesne CPU wielordzeniowe

Wnioski

•Magistrale **systemowe** łączą CPU, pamięć i główne komponenty,
•Magistrale **peryferyjne** obsługują komunikację z urządzeniami zewnętrznymi,
•Współczesne procesory korzystają z **hierarchicznego układu magistral**, łączącego różne standardy w jednym SoC.

7.6 Kontroler pamięci i zarządzanie dostępem

Rola kontrolera pamięci

- Steruje przepływem danych między CPU a RAM.
- Tłumaczy żądania procesora na sygnały odczytu i zapisu.
- Realizuje adresowanie, buforowanie, synchronizację i arbitraż.
- Może być zintegrowany w CPU (IMC) lub w chipsecie (starsze rozwiązania).

Funkcja	
Adresowanie	Tłumaczenie adresów logicznych CPU na fizyczne w RAM.
Arbitraż	Kolejkowanie żądań CPU, DMA, GPU.
Synchronizacja	Generowanie sygnałów RAS/CAS, kontrola czasu odczytu i zapisu.
Buforowanie	Tymczasowe przechowywanie danych w cache.
ECC	Wykrywanie i korekcja błędów w pamięci.
Wielokanałowość	Równoległy dostęp do modułów pamięci (Dual/Quad Channel).

Zarządzanie dostępem – MMU (Memory Management Unit)

Zadanie MMU	
Translacja adresów	Mapowanie adresów logicznych → fizycznych.
Ochrona pamięci	Oddzielenie przestrzeni adresowych procesów.
Stronicowanie (Paging)	Dynamiczne przydzielanie bloków pamięci.
TLB Cache	Bufor szybkiego tłumaczenia adresów.

Poziom	**Typ pamięci**	**Czas dostępu**	**Rozmiar**
L1	Cache instrukcji/danych	1–2 ns	32–128 kB
L2	Cache pośrednia	3–10 ns	256 kB – 2 MB
L3	Cache współdzielona	10–30 ns	4–64 MB
RAM	Pamięć operacyjna	50–100 ns	GB
Dysk/SSD	Pamięć masowa	100 µs – 1 ms	TB

Priorytety dostępu do pamięci.

Źródło żądania	Priorytet	Charakterystyka
CPU	Wysoki	Operacje krytyczne i natychmiastowe
DMA / GPU	Średni	Transfery blokowe, „burst mode”
Peryferia	Niski	Okresowe dostępy (I/O, czujniki)

Wnioski:

- Kontroler pamięci - zarządzanie przepływem danych i synchronizacją RAM.
- MMU - translacja, ochrona i izolacja przestrzeni pamięci.
- Hierarchia cache i arbitraż zapewniają wydajność i bezpieczeństwo w dostępie do pamięci.

7.7 Instrukcje maszynowe i tryby adresowania

Instrukcje maszynowe – ogólna charakterystyka

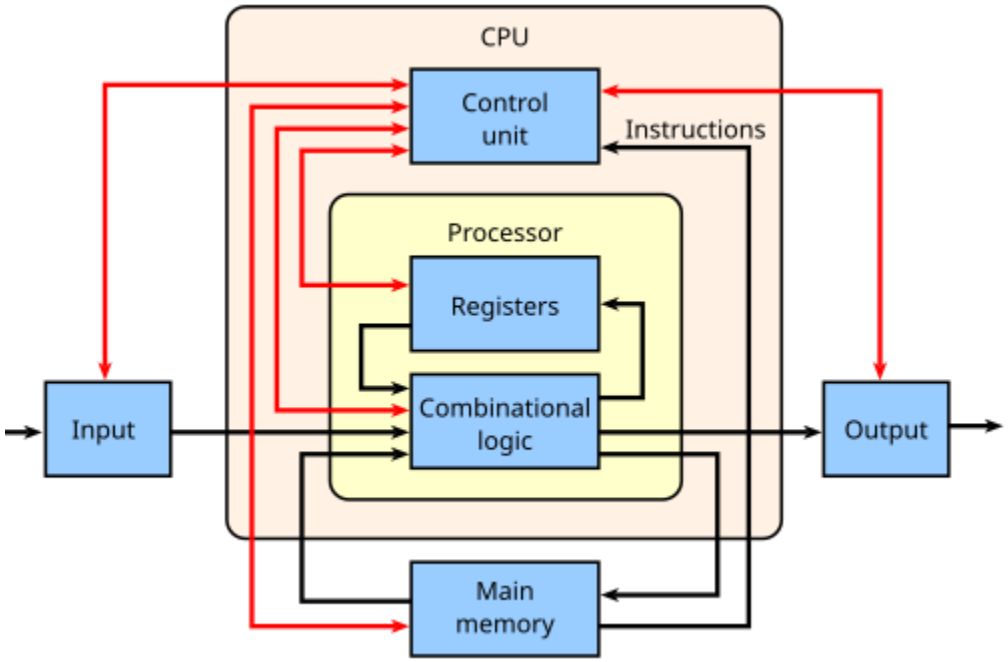
- Instrukcje maszynowe to najniższy poziom programu, który procesor może bezpośrednio wykonać.
  Każda instrukcja zawiera:
- kod operacji (opcode) – określa, co ma być wykonane,
- argumenty (operandy) – dane lub adresy danych,
- tryb adresowania – sposób, w jaki procesor interpretuje operand.

Rodzaj instrukcji		Przykłady
Arytmetyczne	Operacje matematyczne na danych	ADD, SUB, INC, DEC, MUL, DIV
Logiczne	Operacje bitowe i logiczne	AND, OR, XOR, NOT, SHL, SHR
Porównania i testowania	Ustalanie relacji między danymi	CMP, TEST
Sterujące (skoki)	Zmiana przebiegu programu	JMP, CALL, RET, JZ, JNZ
Wejścia/wyjścia (I/O)	Komunikacja z urządzeniami peryferyjnymi	IN, OUT, READ, WRITE
Transferu danych	Przenoszenie danych między rejestrami i pamięcią	MOV, LOAD, STORE, PUSH, POP
Systemowe	Obsługa przerwań, statusów, trybów CPU	INT, HLT, NOP, WAIT

Struktura instrukcji maszynowej

Każda instrukcja składa się z pól:

- Opcode - kod operacji (np. ADD, MOV, JMP)
- Adres(y) - wskazanie źródła i celu danych
- Tryb adresowania - określa sposób dostępu do danych
- Bity sterujące - flagi i opcje (np. warunek skoku, rozmiar danych)



Tryb adresowania	Opis	Przykład (symboliczny)
Natychmiastowy (Immediate)	Wartość zapisana bezpośrednio w instrukcji	`MOV A, #25`
Rejestrowy (Register)	Operand znajduje się w rejestrze	`ADD A, R1`
Bezpośredni (Direct)	Adres pamięci jest jawnie podany	`MOV A, [1000h]`
Pośredni (Indirect)	Adres danych znajduje się w rejestrze	`MOV A, [R2]`
Indeksowy (Indexed)	Adres = rejestr bazowy + przesunięcie	`MOV A, [BX+05h]`
Z bazą i indeksem (Base + Index)	Adres obliczany z dwóch rejestrów	`MOV A, [BX+SI]`
Relatywny (Relative)	Adres docelowy = PC + przesunięcie	`JMP +5`

Zależność między trybami a wydajnością

- Tryby natychmiastowy i rejestrowy – najszybsze (bez dostępu do pamięci).
- Tryby pośrednie i indeksowe – wolniejsze, ale bardziej elastyczne.
- Tryb relatywny – używany w instrukcjach skoku i sterowania przepływem programu.

Bibliografia:

https://eduinf.waw.pl/inf/alg/002_struct/0045.php
https://esezam.okno.pw.edu.pl/mod/book/view.php?id=50&chapterid=1058
https://esezam.okno.pw.edu.pl/mod/book/view.php?id=52&chapterid=1095
https://www.mbaskool.com/business-concepts/operations-logistics-supply-chain-terms/1666-lifo-last-in-first-out-inventory.html
https://home.agh.edu.pl/~ggora/lectures/MSWSiS_Pamieci.pdf
https://eduinf.waw.pl/inf/prg/009_kurs_avr/2017.php
https://elektronikjk.pl/pdf/Przetworniki_cyfrowo-analogowe.pdf
https://www.allaboutcircuits.com/textbook/digital/chpt-13/delta-sigma-adc/
https://home.agh.edu.pl/~zbrudnic/Automatyka/Cw1/Przetw_AC.html
https://zm.put.poznan.pl/opisy/nowe/cw%20misp%20przetw%20a%20c%20z%20podw%20calkow%20new.pdf
https://esezam.okno.pw.edu.pl/mod/book/view.php?id=63&chapterid=1355
https://www.geeksforgeeks.org/computer-science-fundamentals/central-processing-unit-cpu/
https://www.learncomputerscienceonline.com/instruction-cycle/
https://en.wikipedia.org/wiki/Central_processing_unit
https://en.wikipedia.org/wiki/Bus_(computing)
https://www.geeksforgeeks.org/computer-science-fundamentals/central-processing-unit-cpu/
https://www.ibm.com/think/topics/central-processing-unit

Bibliografia została ułożona w sposób chronologiczny, jeżeli jakiś aspekt jest nie jasny bądź temat nie został w pełni wyczerpany w bardzo prosty sposób można zaczerpnąć wiedzy u źródeł.

Piotr Pater