Architektura komputerów

dr inż. Sławomir Samolej
D108 A, tel: 865 1486,
email: ssamolej@prz-rzeszow.pl
WWW: ssamolej.prz-rzeszow.pl

Podział komputerów

Honeywell-Bull DPS 7
mainframe

- **Komputery główne – mainframe** – przeznaczone do realizacji złożonych obliczeń, w tym jednoczesnej obsługi kilkuset użytkowników. Istotą tych komputerów jest wysoka niezawodność, stosowane są do obsługi tzw. aplikacji krytycznych, w których oczekuj się ciągłości działania – banki, instytucje rządowe itp.

Podział komputerów

Superkomputer
Cray - 2

- **Superkomputery** – przeznaczone do obliczeń naukowo-technicznych. Wysoką moc obliczeniową uzyskuje się przez wprowadzenie przetwarzania równoległego. Do budowy stosuje się np. tysiące standardowych procesorów. Zastosowania: modelowanie molekuł, przewidywanie pogody, modelowanie syntezy jądrowej, genetyka itp.

Podział komputerów

Desktop

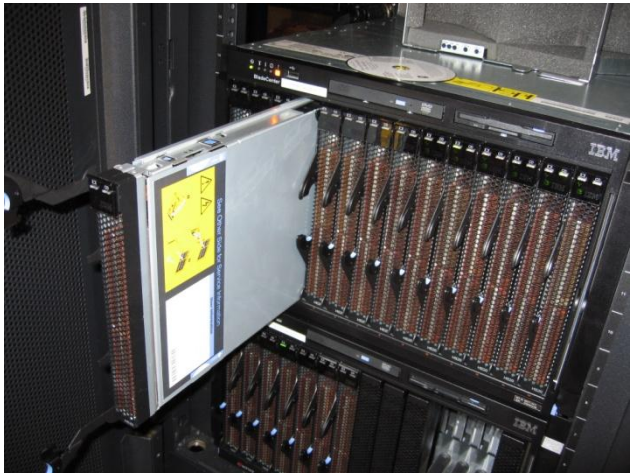
Laptop/Netbook

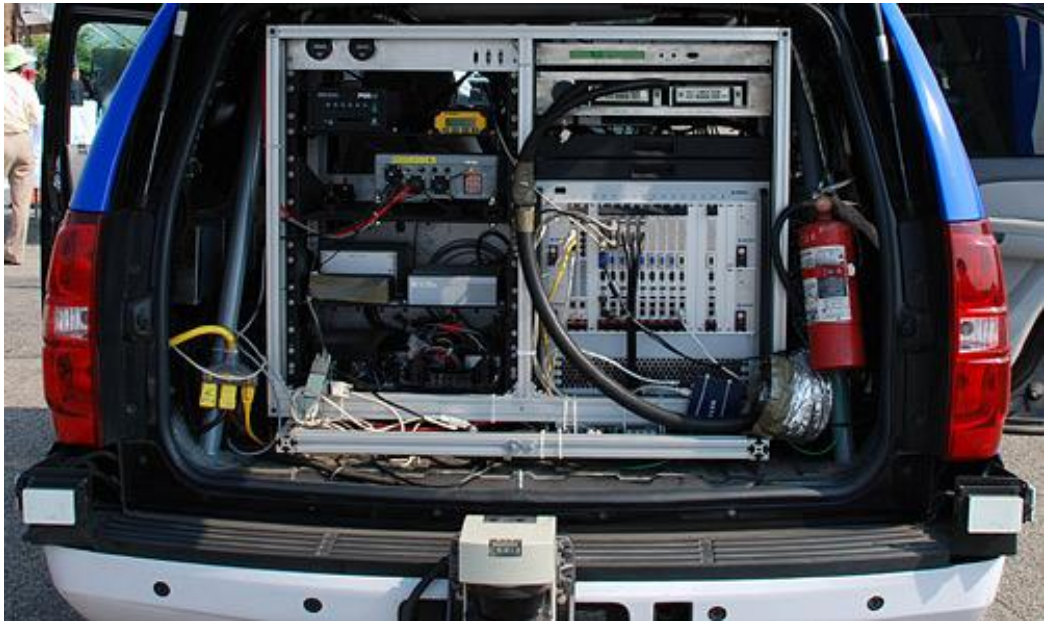
Tablet

Palmtop/Smartfon

- **Mikrokomputery** – przeznaczone do edycji tekstów, dostępu do zasobów zdalnych, obróbki niewielkich zborów danych, rozrywki, komunikacji. Przeznaczone głownie do interakcji z użytkownikiem – przyjazny (obecnie wyłącznie graficzny) interfejs użytkownika.

Podział komputerów





- **Serwery kasetowe** – nowy standard modułowej budowy komputerów. W jednej obudowie instalowane są gotowe moduły obliczeniowe, macierze dyskowe i interfejsy we/wy wspólnie zasilane i centralnie zarządzane.

Podział komputerów

- Komputery wbudowane:

Telefony komórkowe

Konsole do gier

Sterowniki mikroprocesorowe podzespołów samochodów,

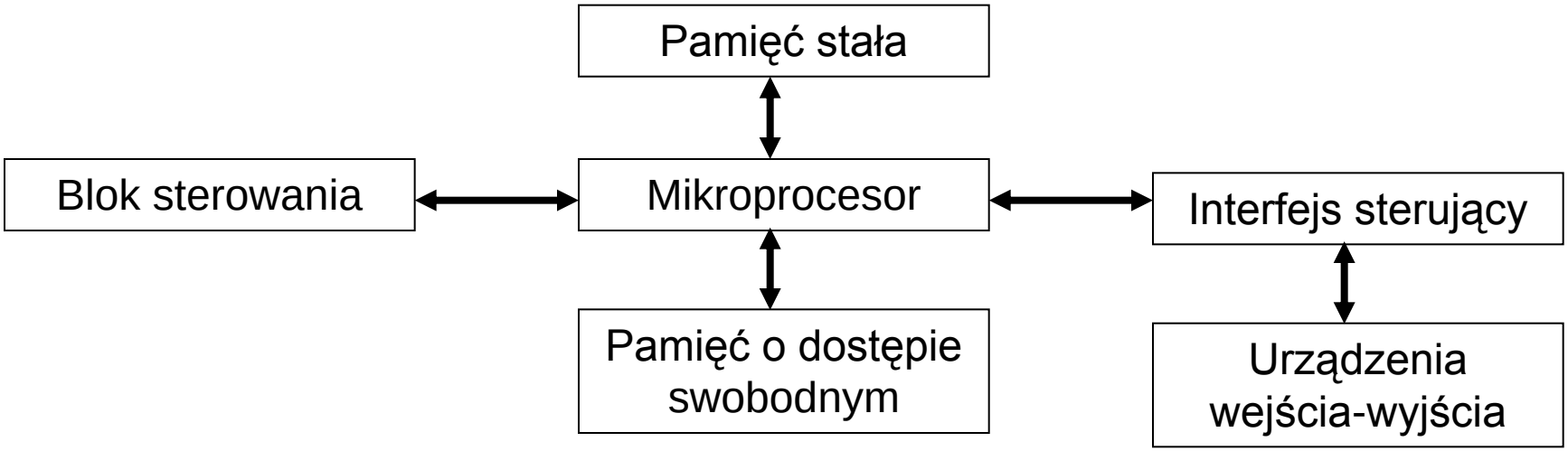
Sterowniki urządzeń AGD

Urządzenia automatyki przemysłowej

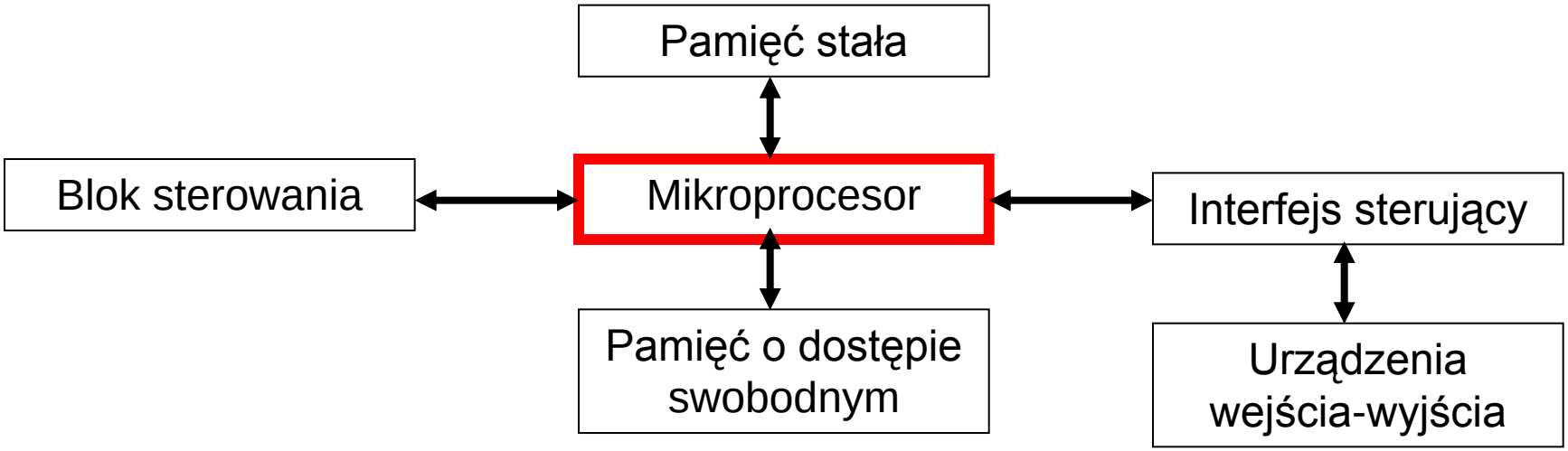
Tendencje w rozwoju sprzętu komputerowego

- Koszt sprzętu drastycznie spadają przy jednoczesnym wzroście wydajności.
- Koszty zakupu lub tworzenia oprogramowania stale rosną i przekroczyły już w większości przypadków koszty zakupu samego komputera
  - W mniejszym stopniu dotyczy to systemów operacyjnych i programów narzędziowych, w większym - aplikacji,
  - W ostatnich latach silnie rozwija się grupa programów darmowych jako przeciwwaga programów komercyjnych.
- Koszt uzyskania i przetworzenia statystycznej jednostki informacji spada, stąd poszerza się obszar zastosowania informatyki.
- Koszt przetworzenia jednostki informacji w większych i wydajniejszych komputerach jest niższy niż w małych niskowydajnych.

Bazowa architektura komputera

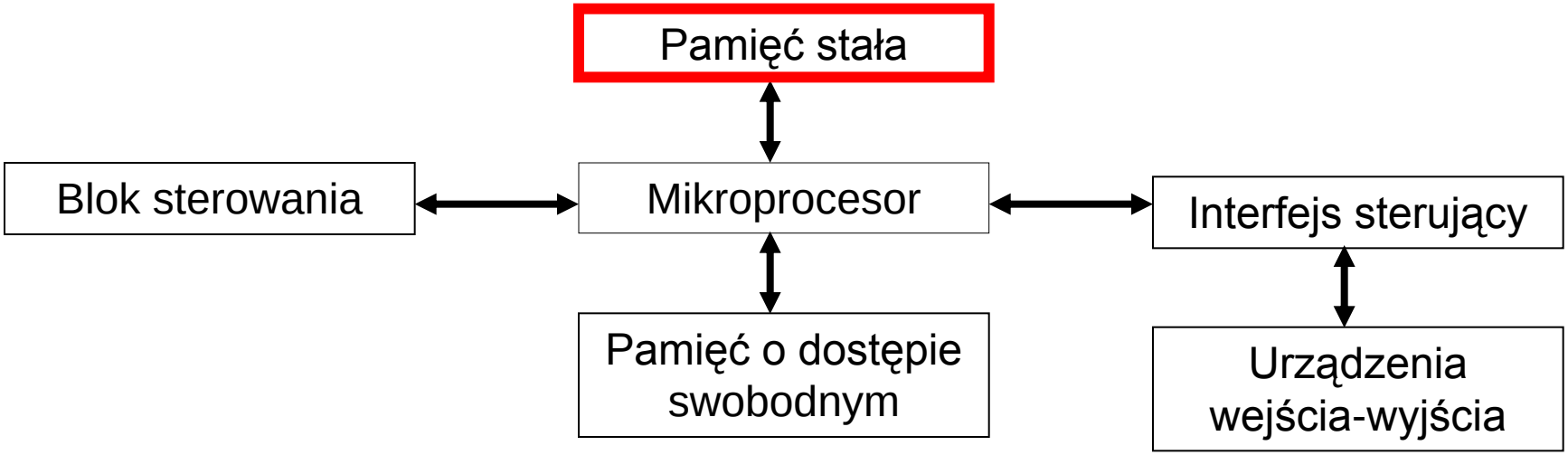


Bazowa architektura komputera



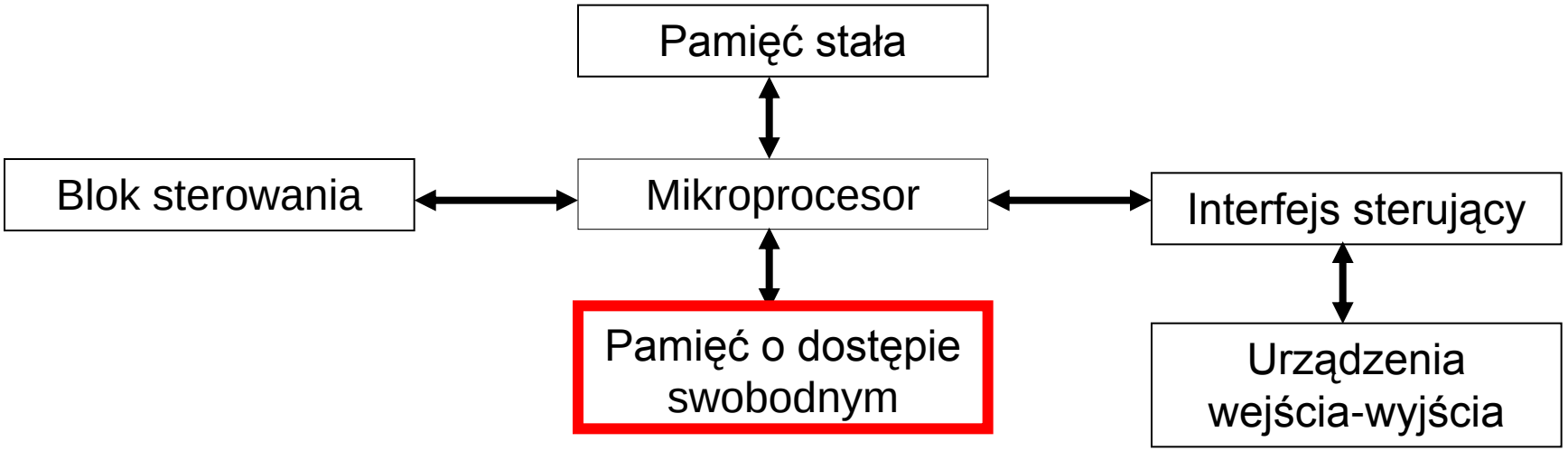
- **Mikroprocesor** to układ scalony zawierający do kilkudziesięciu milionów elementów półprzewodnikowych. Jest podstawowym urządzeniem przetwarzającym i sterującym komputera. Wykonuje większość arytmetycznych i logicznych operacji przetwarzania danych.

Bazowa architektura komputera



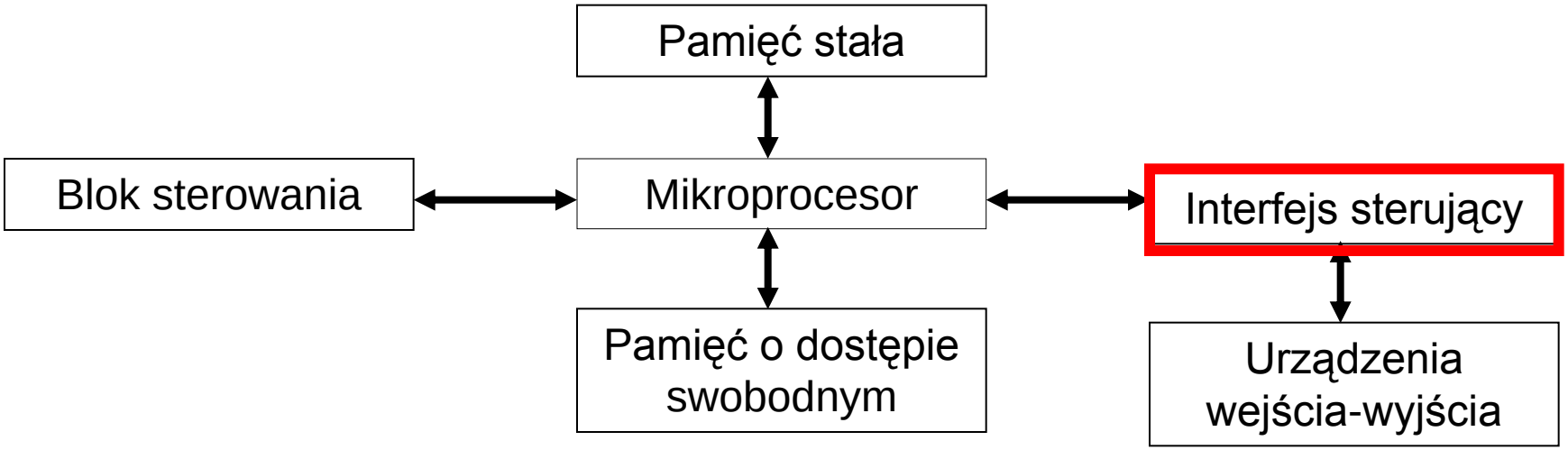
- **Pamięć stała** przechowuje niezmienne komponenty oprogramowania systemu informatycznego pomiędzy seansami pracy komputera. Pamięć ta ma zwykle mały rozmiar i jest przeznaczona do przechowywania wybranych fragmentów systemu operacyjnego. Jej stan nie jest zmieniany przez cały czas eksploatacji, chyba, że zostanie przeprogramowana. Zwyczajowa nazwa tej pamięci to ROM (ang. Read Only Memory).

Bazowa architektura komputera



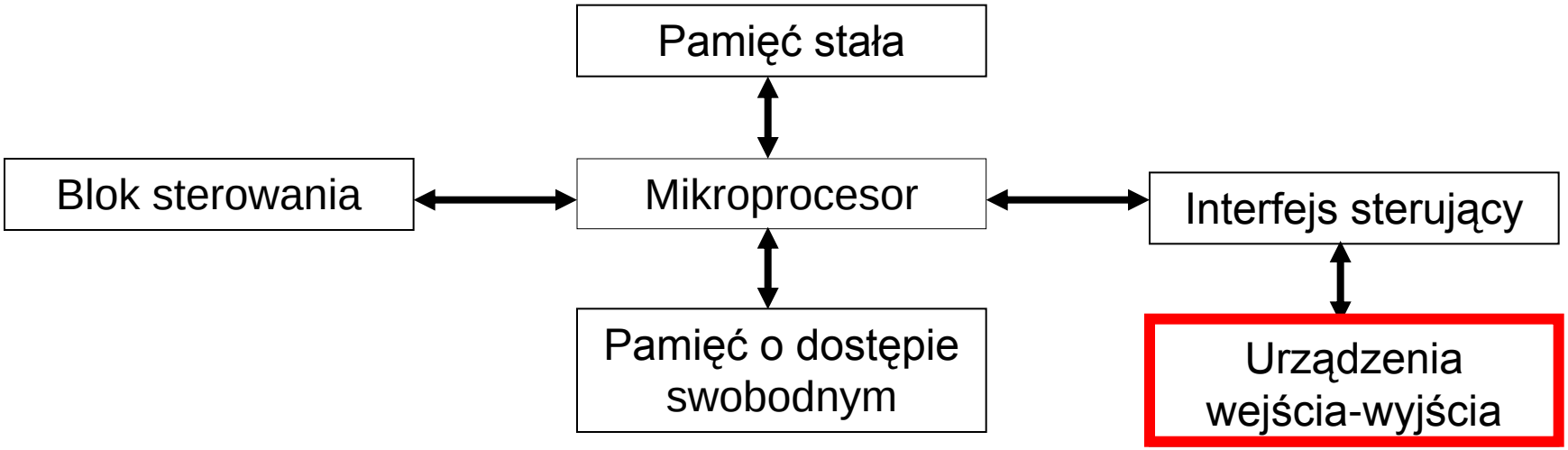
- **Pamięć o dostępie swobodnym** przeznaczona jest do przechowywania danych i programów sterujących wykonywanych w chwili obecnej przez komputer. Pamięć pozwala praktycznie na nieograniczoną ilość zapisów i odczytów informacji. Wyłączenie zasilania pamięci jest równoznaczne z utratą informacji. Duża ilość pamięci zwiększa możliwości przetwarzania przez komputer. Rozmiar pamięci o dostępie swobodnym jest zawsze wielokrotnie większy od rozmiaru pamięci stałej. Praktycznie pamięć jest jednym z najszybciej dostępnych zasobów dla procesora. Nazywana jest pamięcią RAM (ang. Random Access Memory).

Bazowa architektura komputera



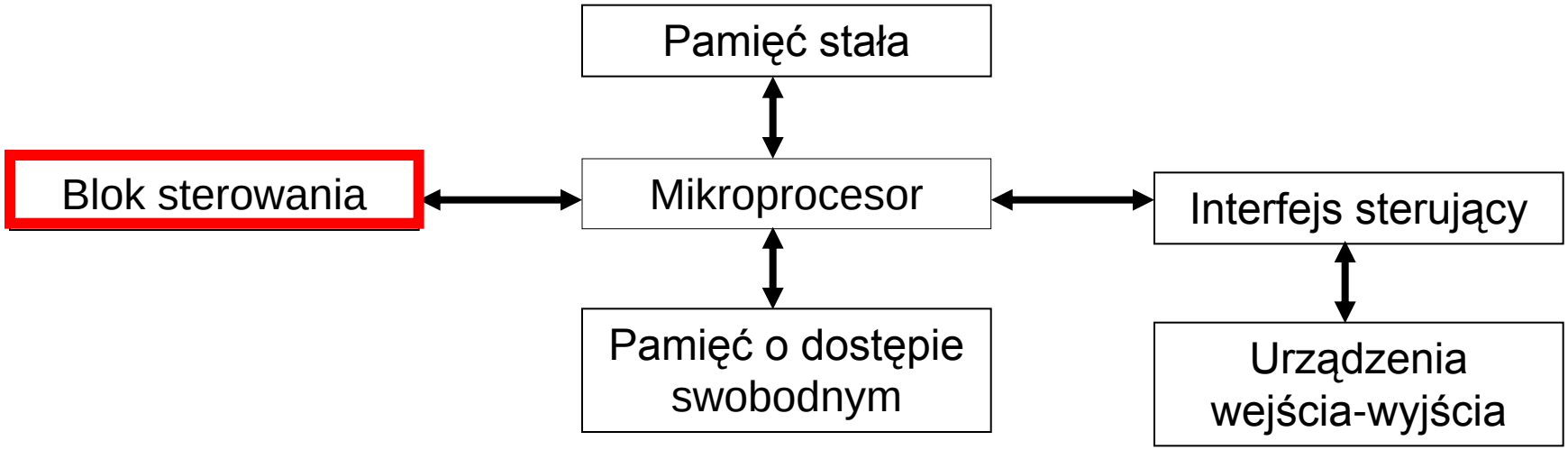
- **Interfejs sterujący** to specjalizowane procesory przeznaczone do zarządzania urządzeniami wejścia/wyjścia. Konstrukcje współczesnych komputerów cechuje wyznaczenie dla procesora przede wszystkim roli wykonywania obliczeń. Otrzymane wyniki obliczeń przekazywane są interfejsowi sterującemu, który przetwarza je na wymaganą postać: obraz, dźwięk, plik dyskowy, pakiet danych przesłany w sieci itp..

Bazowa architektura komputera



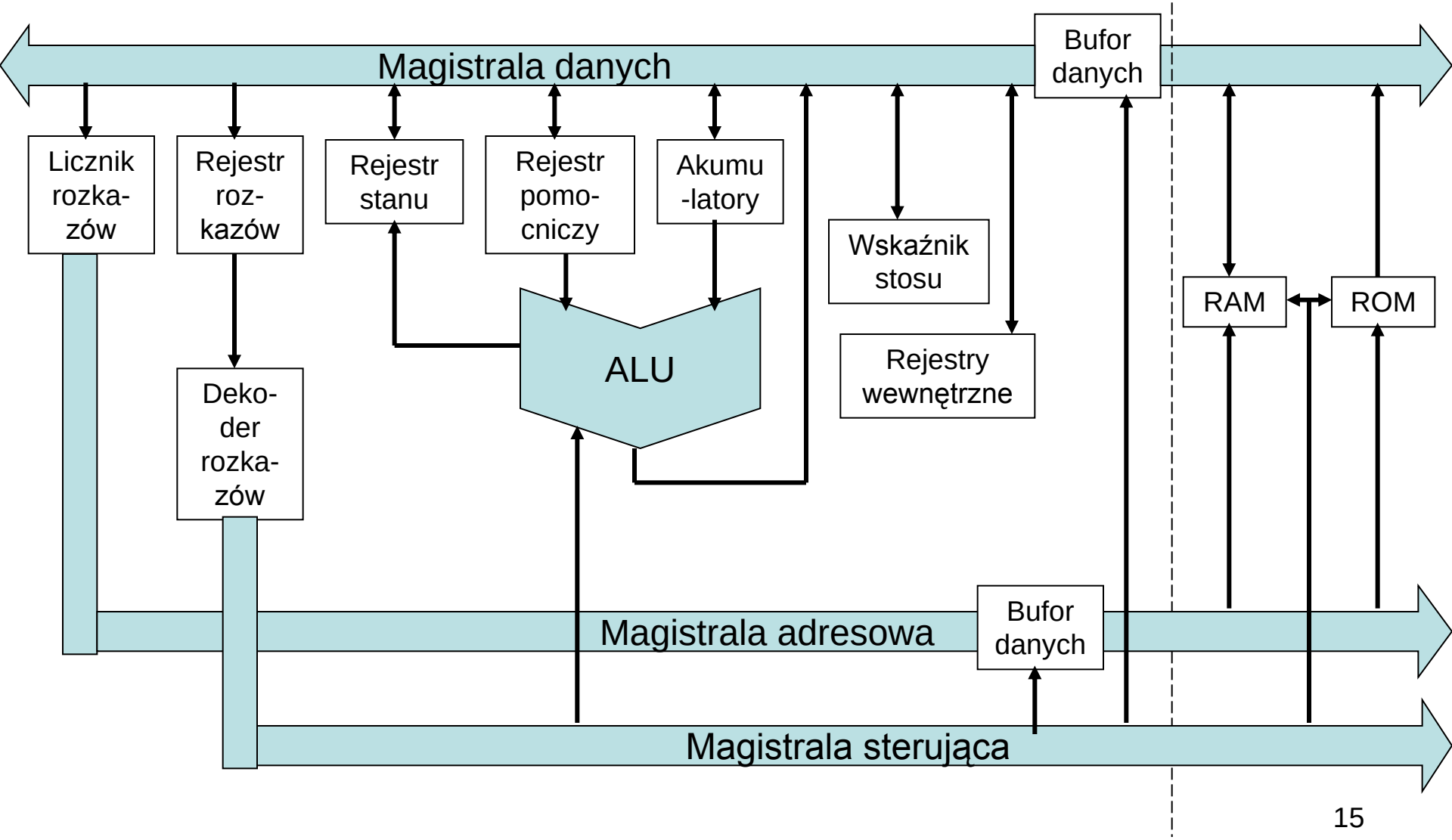
- **Urządzenia wejścia/wyjścia** to konstrukcje elektromechaniczne, wyposażone w sterowanie mikroprocesorowe, których zadaniem jest przedstawienie użytkownikowi informacji przetworzonej w komputerze oraz przyjmowanie informacji od użytkownika i wprowadzenie jej do komputera (np. klawiatura, mysz, mikrofon, czytnik kart chip, kamera, karta sieciowa, czytnik DVD, dysk twardy itp.).

Bazowa architektura komputera



- **Blok sterowania** ma za zadanie wytwarzanie impulsów synchronizujących pracę całego komputera. Dzięki niemu rozpoczynanie i zakończenie przetwarzania w poszczególnych blokach komputera będzie zsynchronizowane. Blok ten określa również sekwencję elementarnych działań wykonywanych przez pozostałe bloki komputera.

Architektura mikroprocesora



Architektura mikroprocesora

- **Magistrala danych**
  - Służy do przesyłania informacji pomiędzy dołączonymi do niej elementami architektury,
  - Łączy wewnętrzne bloków procesora oraz integruje procesor z otoczeniem,
  - Do połączeń z elementami zewnętrznymi wymagany jest **bufor danych** umożliwiający dostosowanie parametrów elektrycznych i czasowych sygnału z procesora do otoczenia (zwykle wolniejszego),
  - Szerokość magistrali jest jednym z podstawowych parametrów mikroprocesora (8,16,32,64).

Architektura mikroprocesora

- **Rejestry**
  - Szybkie adresowalne pamięci o niewielkiej pojemnści,
  - Ich liczba w komputerze jest różna – od kilku do kilkudziesięciu,
  - Mogą mieć przypisane określone funkcje (rej. Dedykowane),
  - Mogą być uniwersalne, zwykle stosuje się kombinację uniwersalnych z dedykowanymi,
  - Są najszybszymi elementami pamięci w komputerze.

Architektura mikroprocesora

- **Akumulator**
  - Rejestr o szczególnym przeznaczeniu,
  - Służy do przesyłania danych z i do pamięci,
  - Bierze udział w wykonywaniu operacji arytmetycznych i logicznych,

Architektura mikroprocesora

- **Jednostka arytmetyczno logiczna**
  - Przeznaczona do wykonania wszystkich operacji arytmetycznych i logicznych realizowanych przez procesor,
  - Zwykle operacja procesora sprowadza się na załadowaniu do akumulatora i rejestrów pomocniczych stanów pewnych komórek pamięci, przetworzeniu tych stanów w ALU, przekazaniu wyniku do akumulatora,
  - Wykonanie operacji arytmetycznej powoduje wygenerowanie **znacznika wyniku** przekazywanego do **rejestru stanu** (umożliwia to podjęcie przez programistę działań, szczególnie w przypadku otrzymania nieprawidłowych wyników – dzielenie przez 0, przekroczenie pojemności danych itp.).

Architektura mikroprocesora

- **Licznik rozkazów**
  - Mikroprocesor sterowany jest sekwencją rozkazów rozmieszczonych w adresowalnej pamięci – konieczne jest zatem wskazywanie w trakcie wykonywania programu adresu kolejnego rozkazu
  - Licznik rozkazów jest dedykowanym rejestrem o rozmiarze identycznym jak magistrala adresowa i zawiera wspomniany adres następnego rozkazu.
  - Zwykle rozkazy znajdują się w kolejnych komórkach pamięci i wtedy licznik rozkazów jest zwiększany o 1.
  - Program może również zawierać informację, że następny rozkaz znajduje się gdzie indziej, wtedy licznik rozkazów jest ustawiany na zadaną wartość i wykonywana jest tzw. instrukcja skoku.

Architektura mikroprocesora

- **Wskaźnik stosu**
  - Jedną z podstawowych funkcji programów jest możliwość wywołania podprogramów.
  - Podprogramy przejmują na czas zasoby mikroprocesora.
  - Stan rejestrów procesora sprzed wywołania podprogramu jest gromadzony w pamięci w strukturze danych typu **stos**, a po wywołaniu podprogramu jest odzyskiwany.
  - Stos jest standardowo przechowywany w pamięci zewnętrznej.
  - W mikroprocesorze znajduje się wskaźnik stosu pokazujący początkowy adres stosu w pamięci.

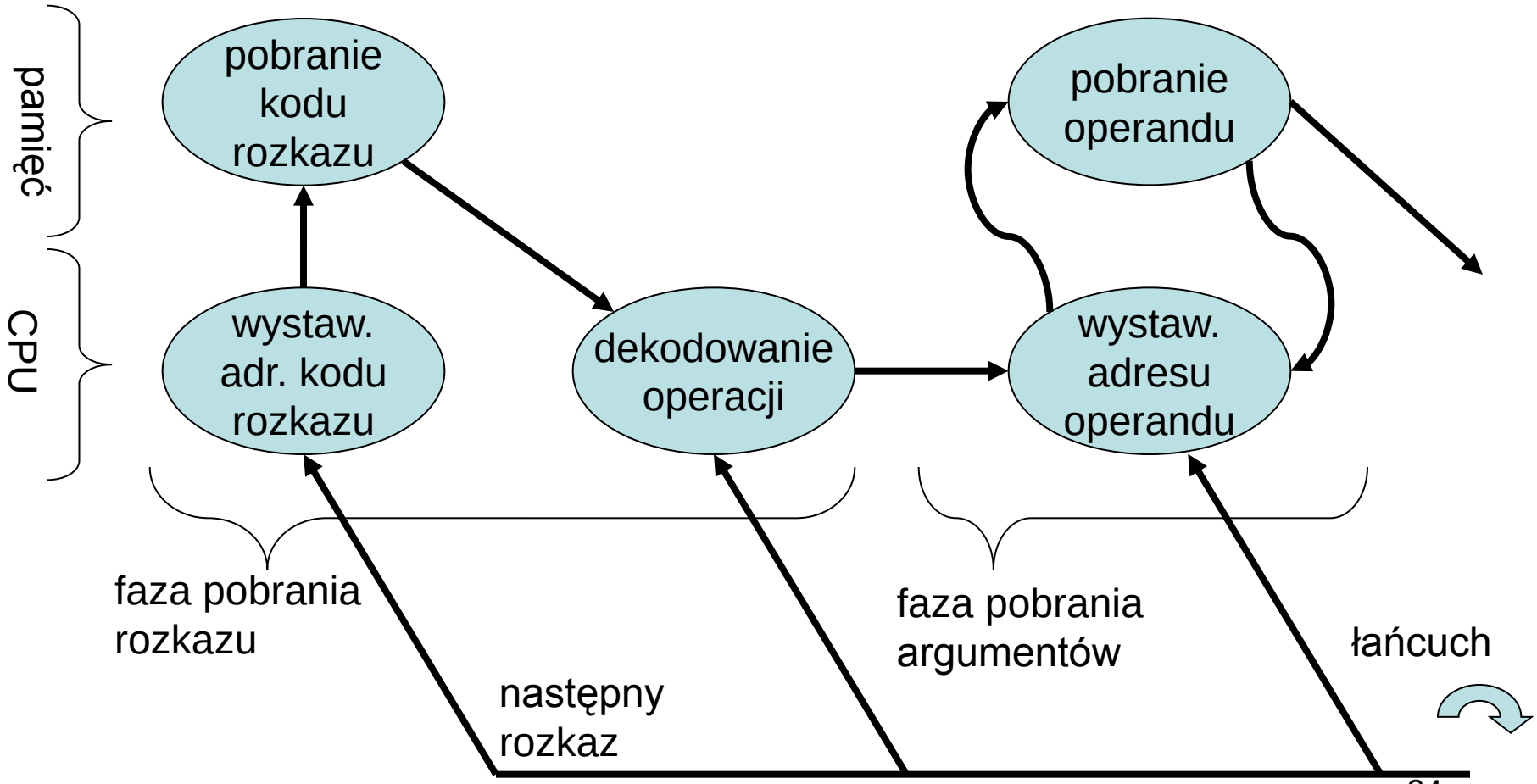
Architektura mikroprocesora

- **Rejestr rozkazów**
  - Każde polecenie programu jest przechowywane w pamięci i przed wykonaniem musi zostać do procesora przeniesione, gdzie po wstępnej obróbce (określenie jakie sygnały sterujące zostaną zastosowane do wykonania) zostanie wykonane,
  - Rejestr rozkazów przechowuje rozkaz aktualnie wykonywany,
  - Pobranie polecenia do rejestru polega na:
    - Wystawieniu na magistrali adresowej adresu komórki określonego w liczniku rozkazów,
    - Pamięć dostarcza na magistralę danych zawartość komórki, która przenoszona jest do rejestru rozkazów.

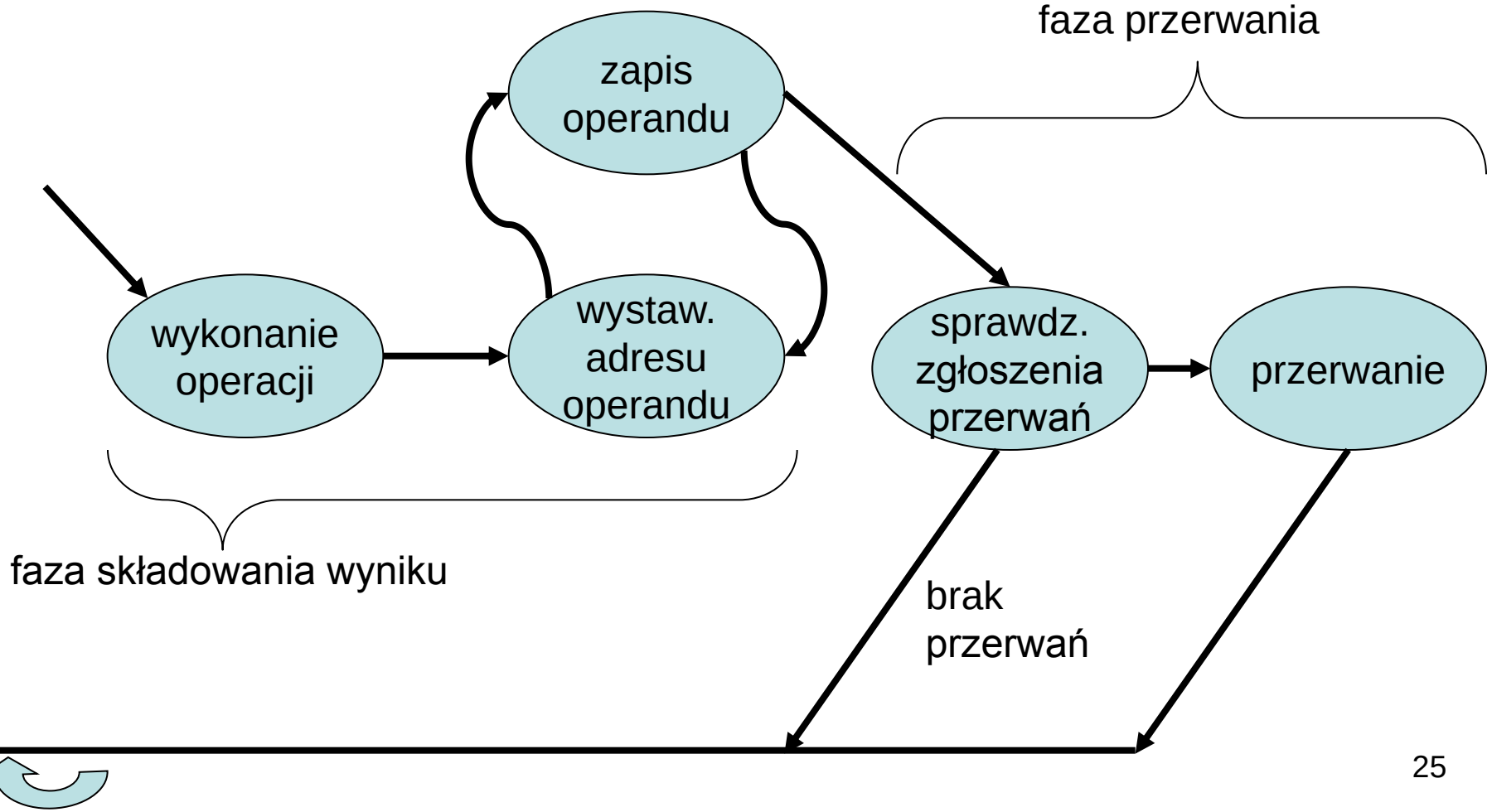
Architektura mikroprocesora

- **Dekoder rozkazów**
  - Sterowanie poszczególnymi blokami mikroprocesora odbywa się za pośrednictwem sygnałów sterujących, wytwarzanych na podstawie konkretnego polecenia w dekoderze rozkazów.
  - Sygnały wystawiane przez dekoder rozkazów stosowane są również do sterowania pracą pamięci i urządzeń wejścia/wyjścia.

Cykl rozkazowy (1)



Cykl rozkazowy (2)



Mikrokomputer vs. Mikroprocesor

Mikrokomputer

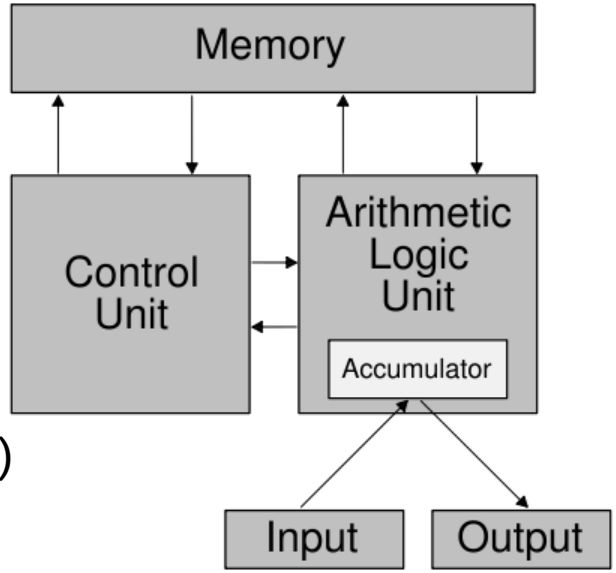
- Wyposażony jest we wbudowane układy wejścia/wyjścia
- Może zawierać w obudowie pamięć RAM/ROM oraz podsystemy wejścia/wyjścia – binarne, analogowe, szeregowe itp..
- W pewnych rozwiązaniach pojedynczy chip spełnia rolę całego komputera.
- Producenci:
  Siemens - 80C166, Motorola, Texas Instruments TMS320Cxx, National Semiconductors

Mikroprocesor

- Zwykle posiada większą moc obliczeniową i z założenia oczekuje istnienia pewnej puli peryferiów z nim współpracujących,
- Producenci:
  Intel - Pentium, AMD - Athlon, Apple-IBM-Motorola - PowerPC

Architektury systemów mikroprocesorowych

- **Architektura von Neumanna** – przedstawiina w 1945 roku przez Johna von Neumanna stworzonej wspólnie z Johnem W. Mauchly'ym i Johnem Presper Eckertem.
- Polega na ścisłym podziale komputera na trzy podstawowe części:
  - procesor (w ramach którego wydzielona bywa część sterująca oraz część arytmetyczno-logiczna)
  - pamięć komputera (zawierająca dane i sam program)
  - urządzenia wejścia/wyjścia
- System komputerowy von Neumanna nie posiada oddzielnych pamięci do przechowywania danych i instrukcji. Wykonywany program może się sam modyfikować traktując obszar instrukcji jako dane, a po przetworzeniu tych instrukcji - danych - zacząć je wykonywać.



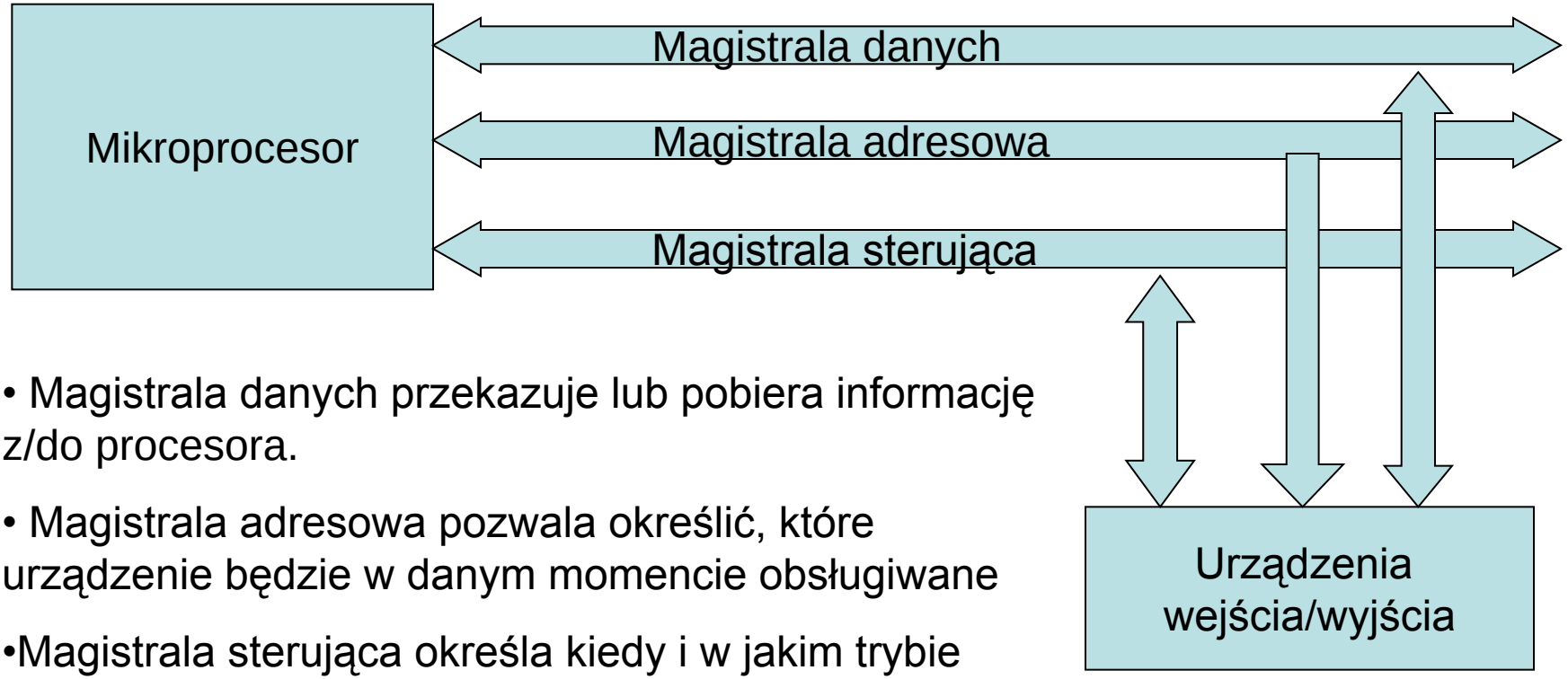
Architektury systemów mikroprocesorowych

- **Architektura harwardzka** - w odróżnieniu od architektury von Neumanna, pamięć danych programu jest oddzielona od pamięci rozkazów.
- Prostsza (w stosunku do architektury von Neumanna) budowa przekłada się na większą szybkość działania - dlatego ten typ architektury jest często wykorzystywany w procesorach sygnałowych oraz przy dostępie procesora do pamięci cache.
- Separacja pamięci danych od pamięci rozkazów sprawia, że architektura harwardzka jest obecnie powszechnie stosowana w mikrokomputerach jednoukładowych, w których dane programu są najczęściej zapisane w nieulotnej pamięci ROM (EPROM/EEPROM), natomiast dla danych tymczasowych wykorzystana jest pamięć RAM (wewnętrzna lub zewnętrzna).

Architektury systemów mikroprocesorowych

- **Zmodyfikowana architektura harwardzka** - znana również jako **architektura mieszana**, łączy w sobie cechy architektury harwardzkiej i architektury von Neumanna. Oddzielone zostały pamięci danych i rozkazów, lecz wykorzystują one wspólne magistrale danych i adresową. Architektura niniejsza umożliwia łatwe przesyłanie danych pomiędzy rozdzielonymi pamięciami.
- Przykładem wykorzystania zmodyfikowanej architektury harwardzkiej jest rodzina mikrokontrolerów 8051.

Dołączanie urządzeń do komputera



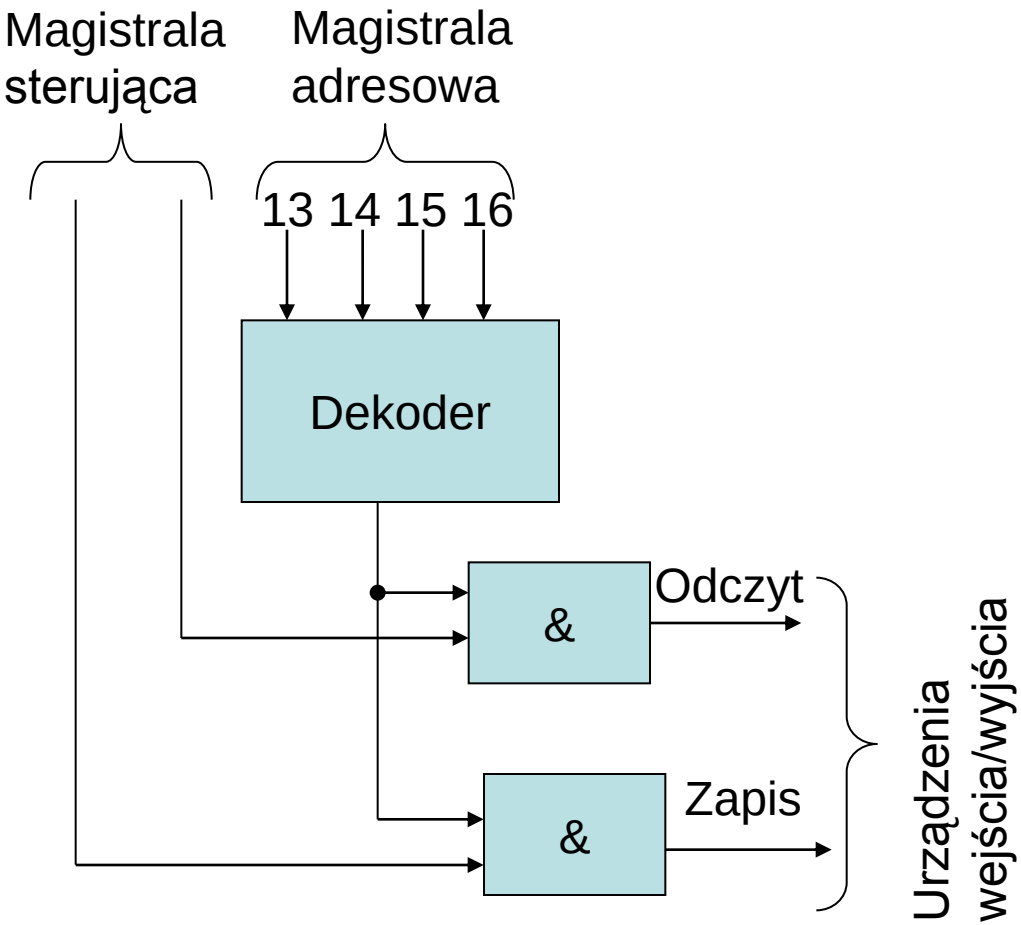
• Magistrala danych przekazuje lub pobiera informację z/do procesora.

• Magistrala adresowa pozwala określić, które urządzenie będzie w danym momencie obsługiwane

•Magistrala sterująca określa kiedy i w jakim trybie urządzenie będzie z magistrali danych , np. zapis danych, odczyt danych, gotowość urz. Wej/wyj itp..

Przykład – dostęp do urz. Wej./Wyj. jako komórki pamięci

- Dekoder analizuje stan 4 linii szyny adresowej,
- Jeśli na szynie znajduje się odpowiednia kombinacja – dekoder wytwarza sygnał informujący urządzenie o gotowości mikroprocesora do skomunikowania się z nim,
- Z magistrali sterującej interfejs pobiera i analizuje sygnały zezwolenia odczytu z pamięci i na ich podstawie generuje końcowy sygnał sterujący urządzeniami wejścia/wyjścia.



Metody sprzęgania mikroprocesora z urządzeniami wejścia/wyjścia

- **Sprzętowe wykonanie operacji wej/wyj** (por. przykład z poprzedniego slajdu).
  - Duża szybkość,
  - Wymaga konstrukcji dekodera,
  - Przyczynia się do nieefektywnego korzystania z przestrzeni adresowej.
- **Wykorzystanie specjalnych poleceń procesora**.
  - Procesor może posiadać specjalne polecenia, które zamiast odwoływać się do pamięci, wystawiają odpowiednie sygnały na magistralę sterującą.

Metody sterowania urządzeniami wejścia/wyjścia

- **Sterowanie programowe**
  - Wyboru urządzenia dokonuje zestaw sygnałów sterujących generowanych przez program zarządzający transmisją.
  - Program musi ciągle śledzić stan urządzeń.
- **PRZERWANIA** (szerzej omówione dalej)
  - Przerwanie to zdarzenie zewnętrzne względem wykonywanego aktualnie programu, zaburzając jego normalny ciąg,
  - W rezultacie przerwania może nastąpić zawieszenie aktualnie wykonanego programu i obsługi tzw. Procedury obsługi przerwania,
  - Wiele urządzeń wej./wyj. Kontaktuje się z mikroprocesorem zgłaszając przerwanie. Jest to o tyle korzystne, że tym razem procesor nie jest zaangażowany w inicjalizację i monitorowanie komunikacji.

Techniki zwiększające wydajność mikroprocesorów - przerwania

- **Przerwanie** (ang. *interrupt*) lub **żądanie przerwania** (**IRQ** - **I**nterrupt **R**e**Q**uest) – sygnał powodujący zmianę przepływu sterowania, niezależnie od aktualnie wykonywanego programu. Pojawienie się przerwania powoduje wstrzymanie aktualnie wykonywanego programu i wykonanie przez procesor kodu procedury obsługi przerwania (ang. *interrupt handler*).
- Przerwania dzielą się na dwie grupy:
  - Sprzętowe:
    - Zewnętrzne – sygnał przerwania pochodzi z zewnętrznego układu obsługującego przerwania sprzętowe;
    - Wewnętrzne, nazywane wyjątkami (ang. *exceptions*) – zgłaszane przez procesor dla sygnalizowania sytuacji wyjątkowych (np. dzielenie przez zero); dzielą się na trzy grupy:
  - Programowe – z kodu programu wywoływana jest procedura obsługi przerwania; najczęściej wykorzystywane do komunikacji z systemem operacyjnym.

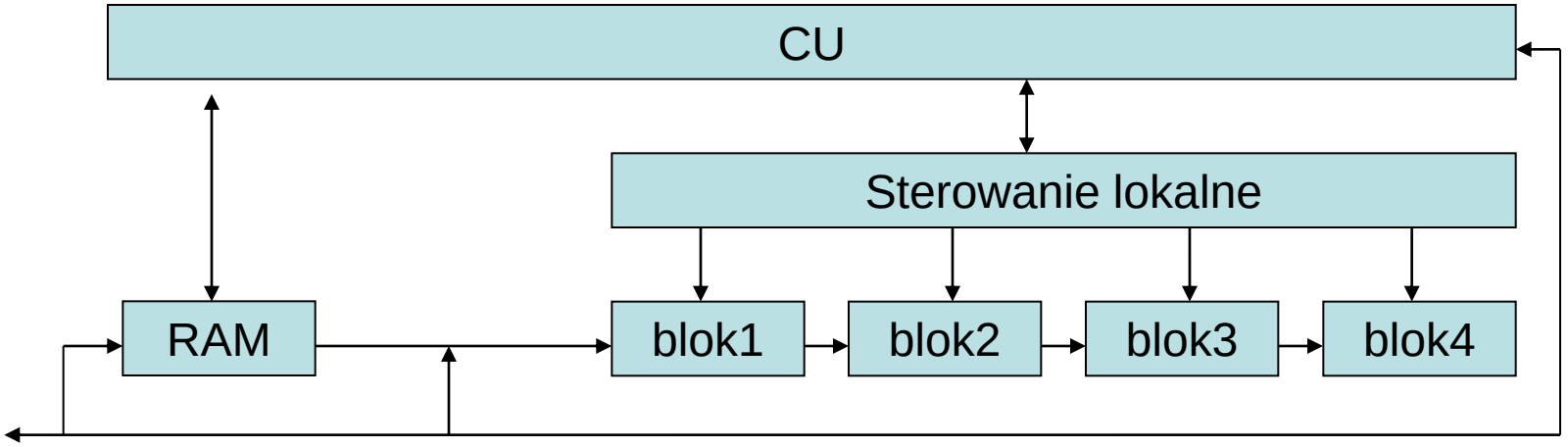
Techniki zwiększające wydajność mikroprocesorów - DMA

- **DMA** (ang. Direct Memory Area - bezpośredni dostęp do pamięci)
  - Technika, w której inne układy (np. kontroler dysku twardego, karta dźwiękowa, itd.) mogą korzystać z pamięci operacyjnej RAM lub (czasami) portów we-wy pomijając przy tym procesor główny - CPU.
  - Wymaga to współpracy ze strony procesora, który musi zaprogramować kontroler DMA do wykonania odpowiedniego transferu, a następnie na czas przesyłania danych zwolnić magistralę systemową. Sam transfer jest już zadaniem wyłącznie kontrolera DMA. Realizacja cykli DMA może przez urządzenie być zrzucona na specjalny układ (np. w komputerach PC) lub być realizowana samodzielnie przez urządzenie.
  - DMA ma za zadanie odciążyć procesor główny od samego przesyłania danych z miejsca na miejsce (np. z urządzenia wejściowego do pamięci), procesor może w tym czasie zająć się 'produktywnym' działaniem, wykonując kod programu pobrany uprzednio z pamięci RAM do pamięci cache operujący na danych w tejże pamięci zgromadzonych. Specjalizowane układy wspomagające DMA (np. te spotykane w PC) potrafią też kopiować obszary pamięci dużo szybciej niż uczyniłby to programowo procesor główny.

Techniki zwiększające wydajność mikroprocesorów – pamięć CACHE

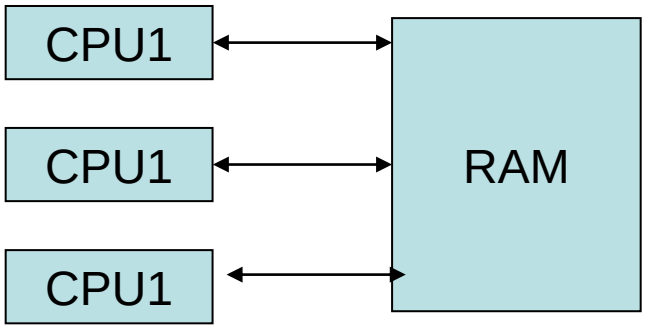
- **pamięć podręczna** -CACHE to mechanizm, w którym ostatnio pobierane dane dostępne ze źródła o dłuższym czasie dostępu i niższej przepustowości są przechowywane w pamięci o lepszych parametrach
  - Pamięć cache przyspiesza dostęp do relatywnie wolnej pamięci RAM. Charakteryzuje się bardzo krótkim czasem dostępu. Jest używana do przechowywania danych, które będą w niedługim czasie przetwarzane. Na współczesnych procesorach są 2 lub 3 poziomy pamięci cache: L1 (zintegrowana z procesorem), a także L2 i L3 (umieszczone w jednym chipie razem z procesorem, lub na płycie głównej).

Techniki zwiększające wydajność mikroprocesorów – linia potokowa



- Operacja obliczeniowa jest rozkładana na zbiór prostszych operacji elementarnych wykonywanych w kolejnych blokach,
- Bezpośrednio po zakończeniu pierwszego kroku, pierwsza porcja przetworzonych danych przekazana zostanie do drugiej itd,
- W tym czasie na wejście pierwszego bliku mogą zostać podane nowe argumenty,
- W rezultacie procesor może wykonywać n instrukcji równocześnie.

Techniki zwiększające wydajność mikroprocesorów – zrównoleglenie



- Zwiększanie szybkości działania pojedynczego procesora wcześniej, czy później napotka na barierę technologiczną,
- Choć nie wszystkie algorytmy sekwencyjne da się zastąpić równoległymi, istotnym podejściem do zwiększenia ogólnej wydajności jest wprowadzenie większej ilości procesorów do systemu,
- Ma to znaczenie w sytuacji, gdy od komputera oczekuje się między innymi interakcji z wieloma użytkownikami (mainframe) lub wykonywania współbieżnie wielu aplikacji (PC),
- Superkomputery buduje się z kilku tysięcy procesorów, dla których opracowuje się efektywne algorytmy współbieżne,
- Wprowadzenie współbieżności wprowadza dodatkowe utrudnienia w projektowaniu oprogramowania oraz niedeterminizm.

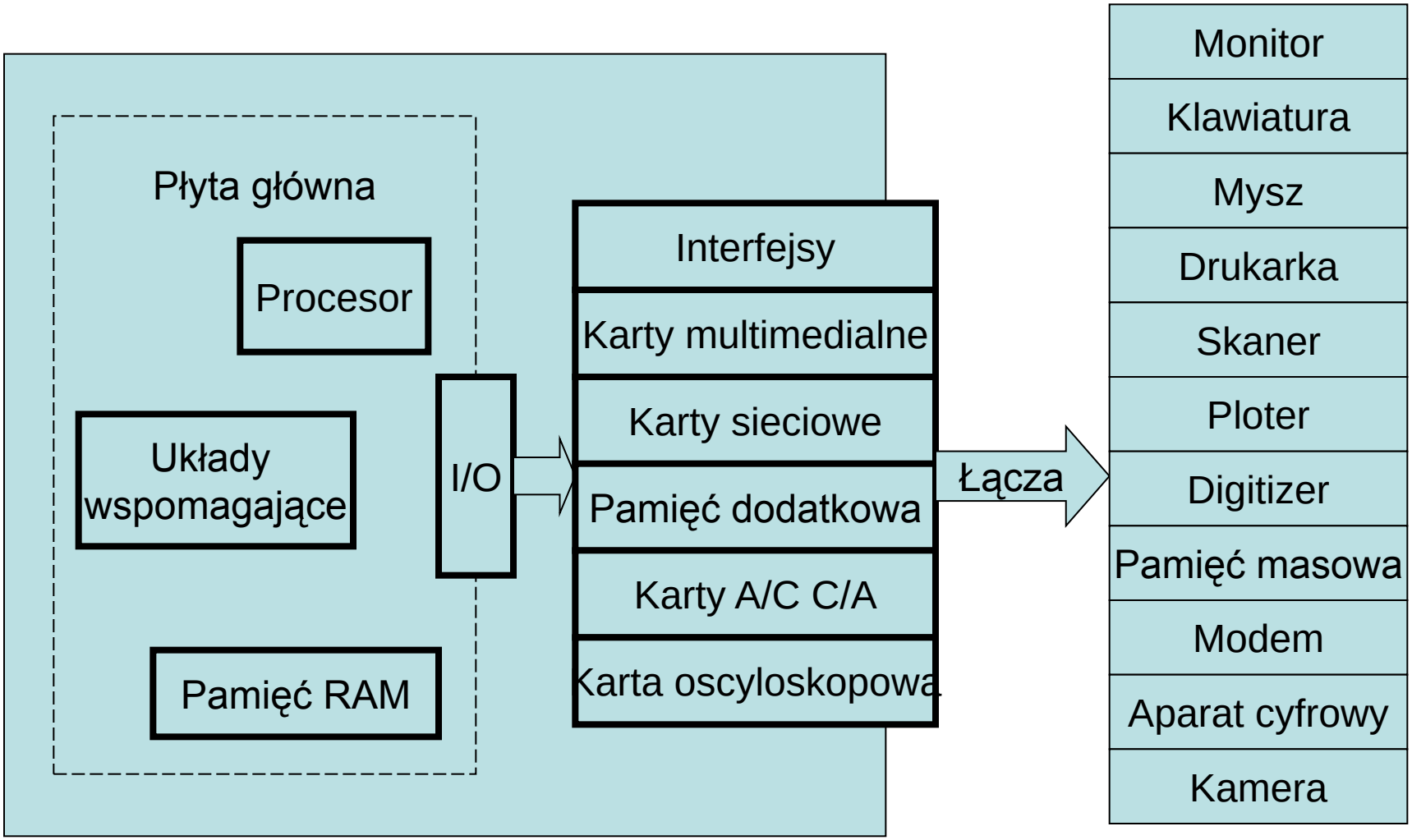
Architektury procesora RISC/CISC

- **CISC** (ang. *Complex Instruction Set Computers*)
  - duża liczba rozkazów (instrukcji)
  - mała optymalizacja – niektóre rozkazy potrzebują dużej liczby cykli procesora do wykonania
  - występowanie złożonych, specjalistycznych rozkazów
  - duża liczba trybów adresowania
  - do pamięci może się odwoływać bezpośrednio duża liczba rozkazów
  - mniejsza od RISC-ów częstotliwość taktowania procesora
  - powolne działanie dekodera rozkazów
  - każda instrukcja może wykonać kilka operacji niskiego poziomu, jak na przykład pobranie z pamięci, operację arytmetyczną, albo zapisanie do pamięci a to wszystko w jednej instrukcji.
  - Przed powstaniem procesorów RISC, wielu komputerowych architektów próbowało zmostkować lukę semantyczną – aby zaprojektować zestawy instrukcji, które wspierałyby języki programowania wysokiego poziomu przez dostarczenie instrukcji wysokiego poziomu np. call i return, instrukcje pętli i kompleksowe tryby adresowania aby pozwolić strukturom danych i szeregom dostępu być połączonym w jedną instrukcję. Rezultatem tego były programy o mniejszym rozmiarze i z mniejszą ilością odwołań do pamięci, co w tamtym czasie bardzo ograniczyło koszty pojedynczego komputera.

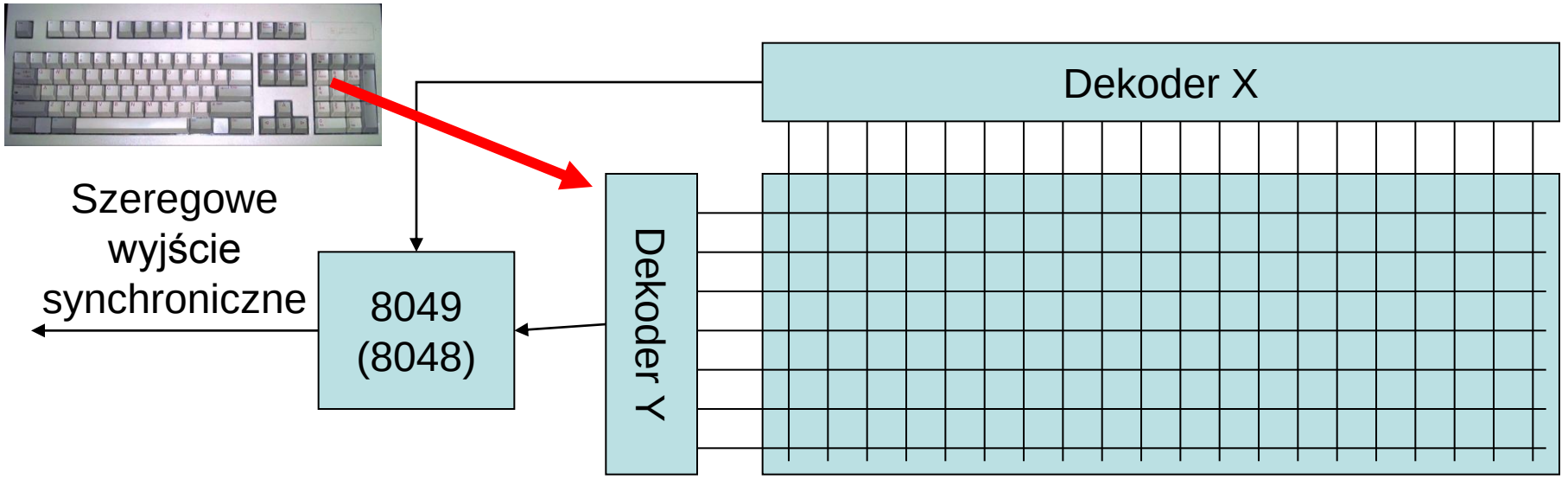
Architektury procesora RISC/CISC

- **RISC** (*Reduced Instruction Set Computers*) - przedstawiona pod koniec lat 70.
  Jej podstawowe cechy to:
  - Zredukowana liczba rozkazów do niezbędnego minimum. Ich liczba wynosi kilkadziesiąt, podczas gdy w procesorach CISC sięga setek.
  - Redukcja trybów adresowania, dzięki czemu kody rozkazów są prostsze, bardziej zunifikowane.
  - Ograniczenie komunikacji pomiędzy pamięcią, a procesorem. Przede wszystkim do przesyłania danych pomiędzy pamięcią, a rejestrami służą dedykowane instrukcje, które zwykle nazywają się **load** (załaduj z pamięci), oraz **store** (zapisz do pamięci); pozostałe instrukcje mogą operować wyłącznie na rejestrach. Schemat działania na liczbach znajdujących się w pamięci jest następujący: załaduj daną z pamięci do rejestru, na zawartości rejestru wykonaj działanie, przepisz wynik z rejestru do pamięci.
  - Zwiększenie liczby rejestrów (np. 32, 192, 256, podczas gdy np. w architekturze x86 jest zaledwie 8 rejestrów), co również ma wpływ na zmniejszenie liczby odwołań do pamięci.
  - Dzięki przetwarzaniu potokowemu (ang. *pipelining*) wszystkie rozkazy wykonują się w jednym cyklu maszynowym, co pozwala na znaczne uproszczenie bloku wykonawczego, a zastosowanie większj ilości ALU także na umożliwienie równoległego wykonywania rozkazów. Dodatkowo czas reakcji na przerwania jest krótszy.
- Przykłady:
  - RCA1802.
  - Obecnie popularne procesory Intela z punktu widzenia programisty są widziane jako CISC, ale ich rdzeń jest RISC-owy.

Budowa IBM PC

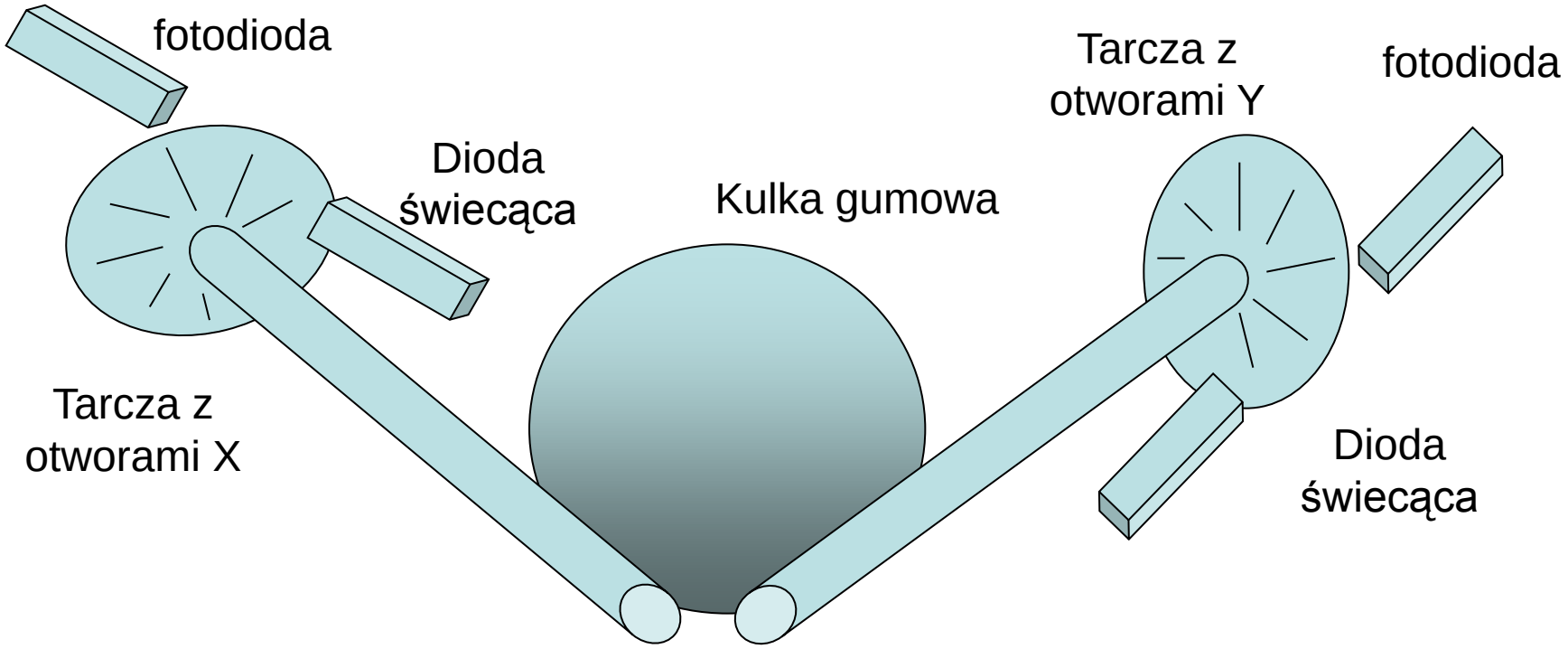


Urządzenia wej - klawiatura



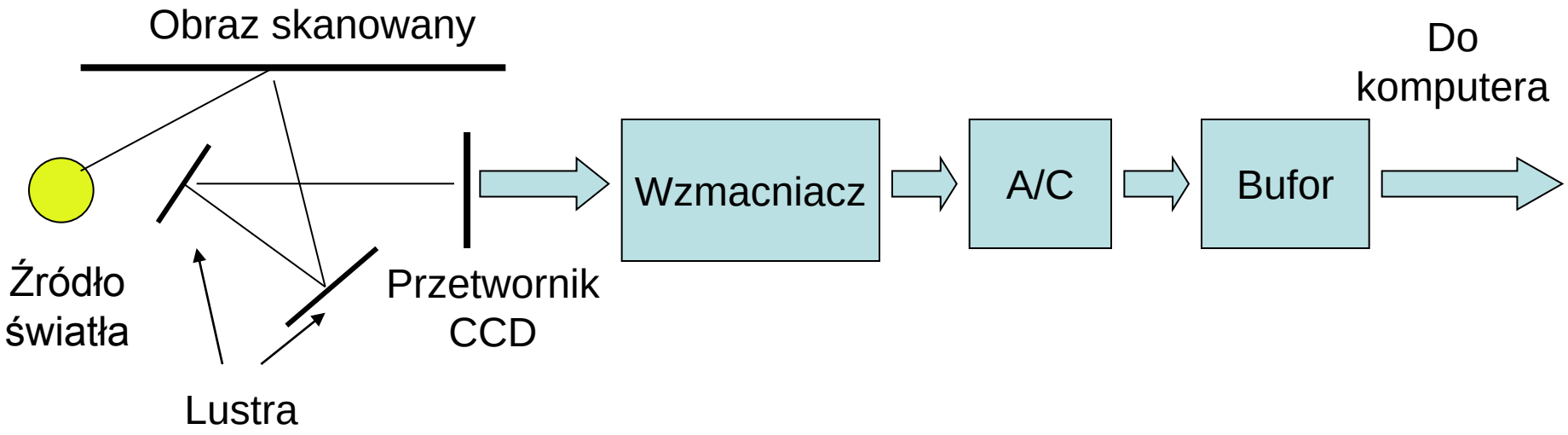
- Na każdą z linii X wysyłane są kolejno sygnały napięciowe, których pojawienie sprawdzane jest na każdej z linii Y,
- Jeśli klawisz został wciśnięty, sygnał napięciowy jest przenoszony z linii X na odpowiednią linię Y,
- Informacja o wykrytym klawiszu jest przenoszona na układ kodujący wysyłający do komputera kod wciśniętego klawisza

Urządzenia wej - mysz



- Obrót kulki powoduje obroty tarcz,
- Otwory w tarczach powodują przerywanie obwodu elektrycznego fotodiody
- Wygenerowane impulsy są przekazywane do komputera i interpretowane jako odpowiedni przyrost położenia wskaźnika myszy.

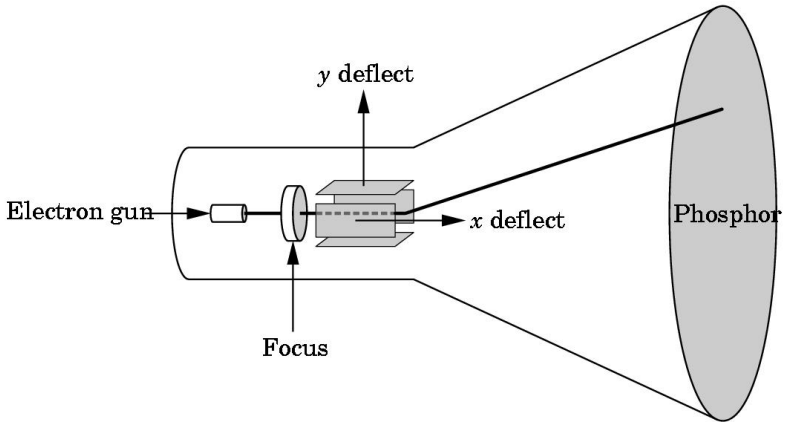
Urządzenia wej - skaner



- Analizowana jest wartość natężenia światła odbitego od powierzchni skanowanej (jasna powierzchnia odbija lepiej od ciemnej),
- Linijka przetworników CCD (ze sprzężeniem ładunkowym) przekształca obraz na serię impulsów elektrycznych ostatecznie przekształcanych na postać cyfrową.

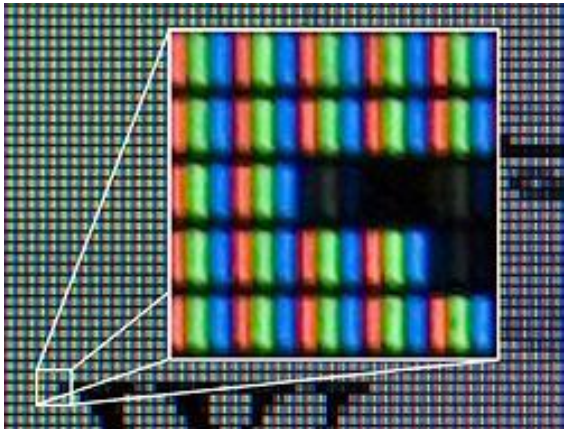
Urządzenia wyj - monitor

CRT



- Działo elektronowe produkuje 3 wiązki elektronów odchylane przez pole elektryczne powodujące świecenie luminowoforu

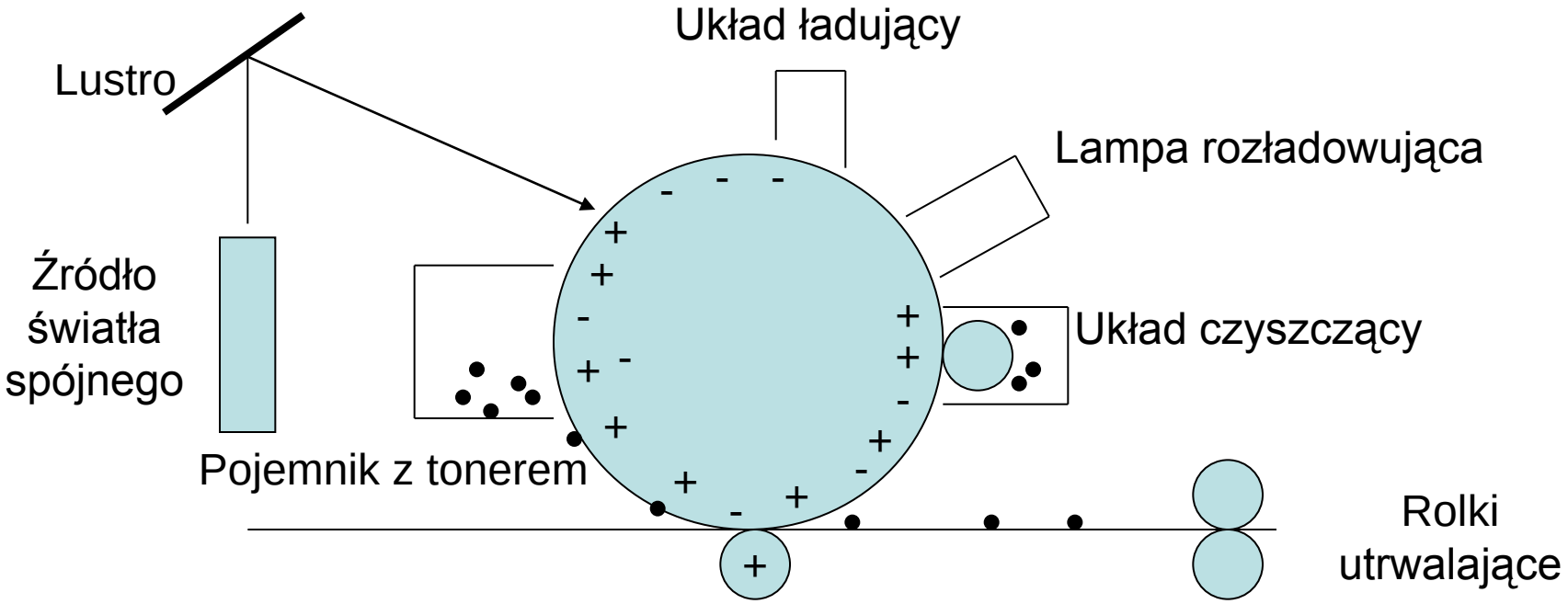
LCD



- Obraz powstaje przez zminę polaryzacji światła na skutek zmian orientacji uporządkowania cząsteczek chemicznych,

  pozostających w fazie ciekłokrystalicznej, pod wpływem przyłożonego pola elektrycznego.

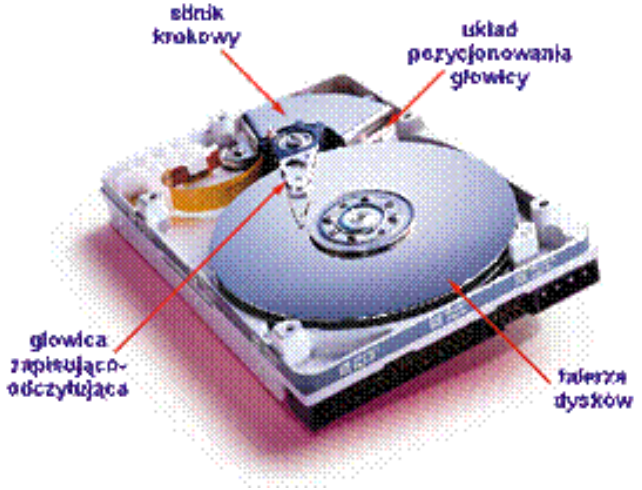
Urządzenia wyj - drukarka

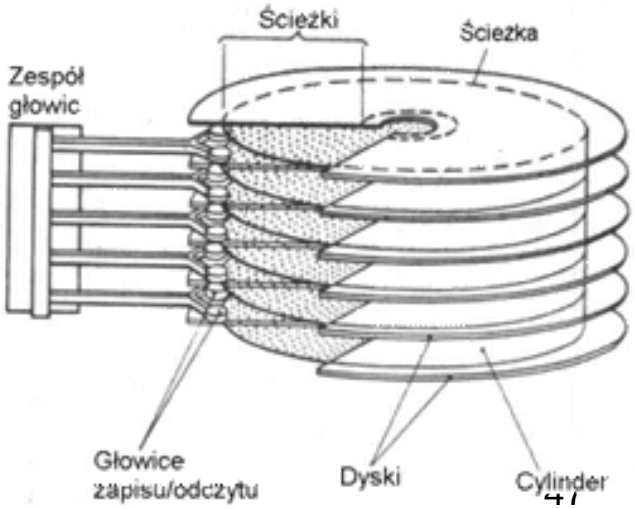


- Bęben selenowy przechodzi pod układem ładującym uzyskując ujemny ładunek elektrostatyczny,
- Następnie bęben zostaje naświetlony przez wiązkę światła spójnego uzyskując w miejscach naświetlenia ładunek dodatni,
- W miejscach naładowanych dodatnio do bębna przykleja się ujemnie naładowany toner, który następnie jest stykowo nakładany na papier i utrwalany.
- Przed ponownym wykorzystaniem bęben jest czyszczony i rozładowywany.

Urządzenia wej/wyj – dyski magnetyczne

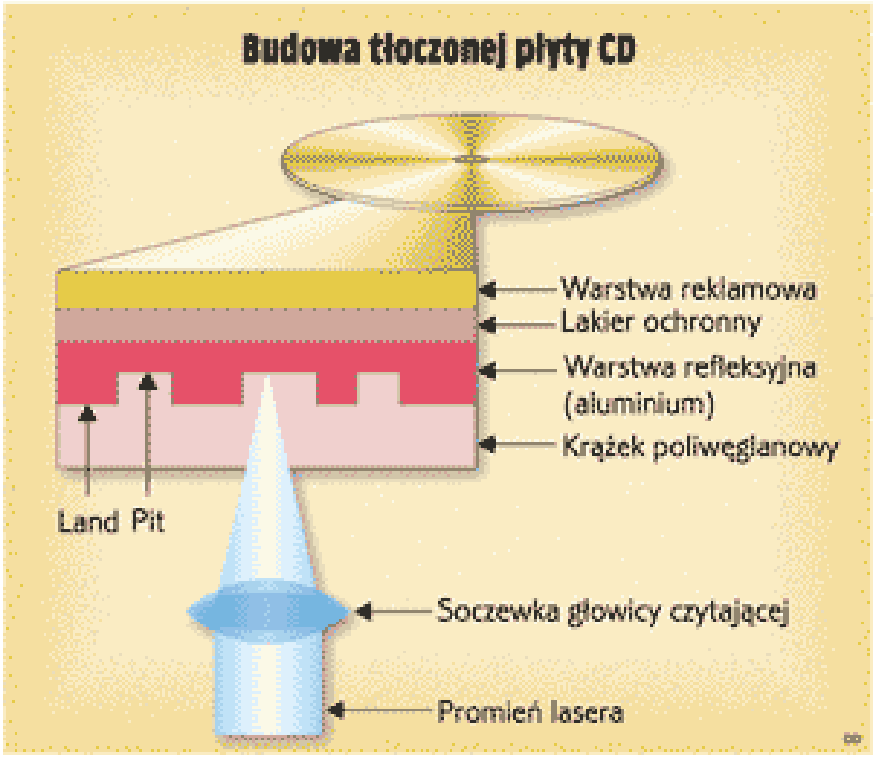
- Dane zapisane są w postaci zmian pola magnetycznego zawartych w okrągłej ścieżce,
- Dysk składa się z zestawu wirujących talerzy magnetycznych,
- Zapis/odczyt danych odbywa się dzięki głowicom sterowanym silnikami krokowymi,





Urządzenia wej/wyj – płyty CD

- Ścieżka na płycie CD na kształt spirali i zaczyna się wewnątrz płyty,
- W ścieżce znajdują się zagłębienia – pity i obszary bez zagłębień – landy.
- Podczas odczytu ścieżka jest oświetlana przez promień lasera.
- Gdy promień pada na pit zostaje rozproszony, promień odbity od landu nie rozprasza się.
- Dane na płycie zapisywane są w postaci ramek o długości 585 bitów: 27b – sekwencja synchronizująca, 1B informacyjno-sterujący, 12B – dane, 4B – kontrola, 12B - dane, 4B – kontrola.



Urządzenia wej/wyj - przegląd

- Karty sieciowe,
- Karty dźwiękowe,
- Karty telewizyjne,
- Porty szeregowe: RS232, USB,
- Karty modemowe,
- Czytniki kart chip,
- Czytniki kart pamięci,
- Karty FireWire
- Karty Bluetooth
- Karty sieci bezprzewodowej,
- Itd…

Systemy zapisu liczb

- **Kodowanie liczb naturalnych**
- **Kodowanie liczb całkowitych ze znakiem**
- **Kodowanie liczb ułamkowych**
- **Kodowanie liczb zmiennopozycyjnych**

Systemy zapisu znaków

- **Kod ASCII**

ASCII [ei-es-si-aj-aj] (ang. *American Standard Code for Information Interchange*) to 7-bitowy kod przyporządkowujący liczby z zakresu 0-127 literom (alfabetu angielskiego), cyfrom, znakom przestankowym i innym symbolom oraz poleceniom sterującym. Przykładowo litera "a" jest kodowana liczbą 97, a polecenie "powrót karetki", czyli [Enter] – liczbą 13. Litery, cyfry oraz inne znaki stosowane w kodzie ASCII tworzą zbiór znaków ASCII (95 znaków).

Pierwsze 32 kody (0-31) oraz ostatni kod (127) to tzw. Znaki sterujące, które oryginalnie nie służyły do przenoszenia informacji, tylko do sterowania urządzeniem odbierającym komunikat (informacje), np. drukarką.

Ponieważ kod ASCII jest 7-bitowy, a większość komputerów operuje na 8-bitowych bajtach, możliwe się stało powiększenie zbioru kodowanych znaków, bowiem ów ósmy bit podwoił liczbę dostępnych kodów (z 128 do 256). Powstało wiele różnych rozszerzeń ASCII wykorzystujących ten ósmy bit (np. norma ISO 8859, rozszerzenia firm IBM lub Microsoft). Rozszerzenia te nazywane są stronami kodowymi

Elementy tablicy znaków ASCII (1)

Bin	Dec	Hex	Znak	Skrót
0000 0000	0	00	Null	NUL
0000 0001	1	01	Start Of Heading	SOH
0000 0010	2	02	Start of Text	STX
0000 0011	3	03	End of Text	ETX
0000 0100	4	04	End of Transmission	EOT
0000 0101	5	05	Enquiry	ENQ
0000 0110	6	06	Acknowledge	ACK
0000 0111	7	07	Bell	BEL
0000 1000	8	08	Backspace	BS

Bin	Dec	Hex	Znak
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E

Elementy tablicy znaków ASCII (2)

0110 0001	97	61	a
0110 0010	98	62	b
0110 0011	99	63	c
0110 0100	100	64	d
0110 0101	101	65	e
0110 0110	102	66	f
0110 0111	103	67	g
0110 1000	104	68	h
0110 1001	105	69	i
0110 1010	106	6A	j
0110 1011	107	6B	k
0110 1100	108	6C	l

Systemy zapisu znaków

- **UNICODE**

ASCII i ANSI nie wystarczają, gdy trzeba zapisać w rozszerzonym zakresie np. ponad 3000 chińskich idiomów lub np. stworzyć jedną stronę kodową dla całej Europy. Unikod (ang. Unicode) jest nowoczesnym sposobem kodowania obejmującym znaki używane na całym świecie w tym wielu, jeżeli wręcz nie wszystkich, krajów (np. polskie, hieroglify czy cyrylicę), symbole muzyczne, techniczne, wymowy i inne często spotykane. W odróżnieniu od dotychczas używanych sposobów, kod numeryczny jednoznacznie identyfikuje symbol. Nie ma sytuacji, że dany kod może oznaczać różne symbole w zależności od numeru strony czy innego znacznika. Wynika z tego możliwość swobodnego mieszania znaków różnych krajów bez obawy o niejednoznaczność.

Istotę Unikodu zgrabnie odzwierciedla określenie alfabet uniwersalny.

Systemy zapisu znaków

- **UNICODE (2)**

Pełny Unikod jest standardem 32-bitowym (UCS-4). Bagatela: 4 294 967 295 znaków.
Aktualnie używane jest jednak tylko 16 bitów (można przypisać liczby 65 535 znakom).

Ze względu na to, iż nie wszystkie systemy komputerowe i programy zdolne są do używania Unikodu w pełnym zakresie oraz dla zapewnienia bezproblemowego transferu przesyłania danych przy użyciu takich systemów komputerowych określono kilka sposobów kodowania:

- **UTF-7 - format 7-bitowy;**
- **UTF-8 - format 8-bitowy;**
- **UTF-16 - format 16-bitowy;**

Specyfiką kodowań UTF-7 i UTF-8 jest przesyłanie kodów ASCII praktycznie bez zmian. Tylko kody większe niż 127 podlegają modyfikacji. Dzięki temu polskie teksty powiększają swoją objętość tylko o niewielki procent (kilka..kilkanaście) zamiast dwukrotnie lub czterokrotnie.

Kodowanie BER/DER – zapis dokumentów podpisywanych

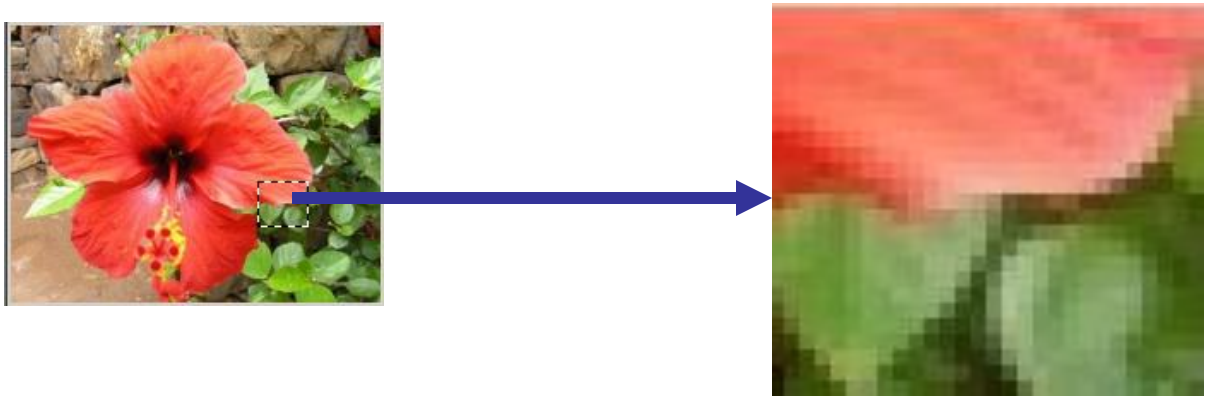
Typ danych:
GENERALIZEDTIME = 20051218133000
jest zakodowany:
18 0e
  32 30 30 35 31 32 31 38 31 33 33 30 30 30.

Systemy zapisu innych typów danych – obrazy, dźwięki ...

- **Pliki graficzne**
- **Pliki dźwiękowe, filmowe**
- **Kompresja**
- **Szyfrowanie**

Formaty plików graficznych - wprowadzenie

Obraz graficzny jest dwuwymiarową tablicą pikseli, zwana czasem rastrem.



Kolor piksela może być reprezentowany w następujący sposób:

- Dla obrazów monochromatycznych – 1 bit (czarny/biały)
- Dla obrazów „true color” dla każdego piksela określane są 3 składowe: czerwona (R), zielona (G), i niebieska (B). Gdy wartość każdej składowej reprezentowana jest przez 1 bajt, to każda barwa może być reprezentowana w 256 odcieniach jasności, co daje 16777216 (ponad 16 mln)
- W przypadku obrazu opartego na palecie, wartości poszczególnych pikseli są indeksami tablicy RGB, zwanej paletą kolorów. Liczba przypadających na jeden piksel bitów zależy od liczby kolorów w palecie. Typowa paleta może zawierać 16 (4 bity na piksel) lub 256 kolorów (8 bitów na piksel).

Formaty plików graficznych - wprowadzenie

Minimum informacji konieczne do wyświetlenia obrazu na ekranie:

1. Rozmiar rysunku (szerokość i wysokość)
2. Liczba bitów na piksel (głębokość)
3. Rodzaj grafiki (monochromatyczna / „true color” / paleta kolorów)
4. Paletę kolorów, jeśli obraz jest opisany na palecie
5. Dane graficzne (tablica pikseli)

Formaty plików graficznych - wprowadzenie

Typowa struktura pliku graficznego:

Formaty plików graficznych - wprowadzenie

Różnice w formatach plików graficznych:

- Kolejność umieszczonych w nagłówku informacji może być różna dla różnych formatów plików.
- Niektóre formaty, przystosowane do konkretnej karty graficznej, nie muszą zawierać palety, a jedynie tablice pikseli.
- Linie obrazu mogą być ułożone w kolejności od góry do dołu lub od dołu do góry.
- Jeżeli wartości pikseli są składowymi RGB, to kolejność zapisu tych składowych może być różna.
- Wartości pikseli mogą być zapisane w postaci pakietów lub planów obrazu. W przypadku pakietów bity, odpowiadające kolejnym pikselom, są zapisywane jeden za drugim. Jeżeli obraz składa się z planów, bity poszczególnych pikseli są zapisywane na tej samej pozycji w kolejnych planach. Najmniej znaczące bity pikseli obrazu znajdują się w jednym planie, a następnym planie są bity bardziej znaczące, itd.
- Dane graficzne mogą być skompresowane.

Formaty plików graficznych – mapy bitowe

Dostępne w systemie Windows mapy bitowe:

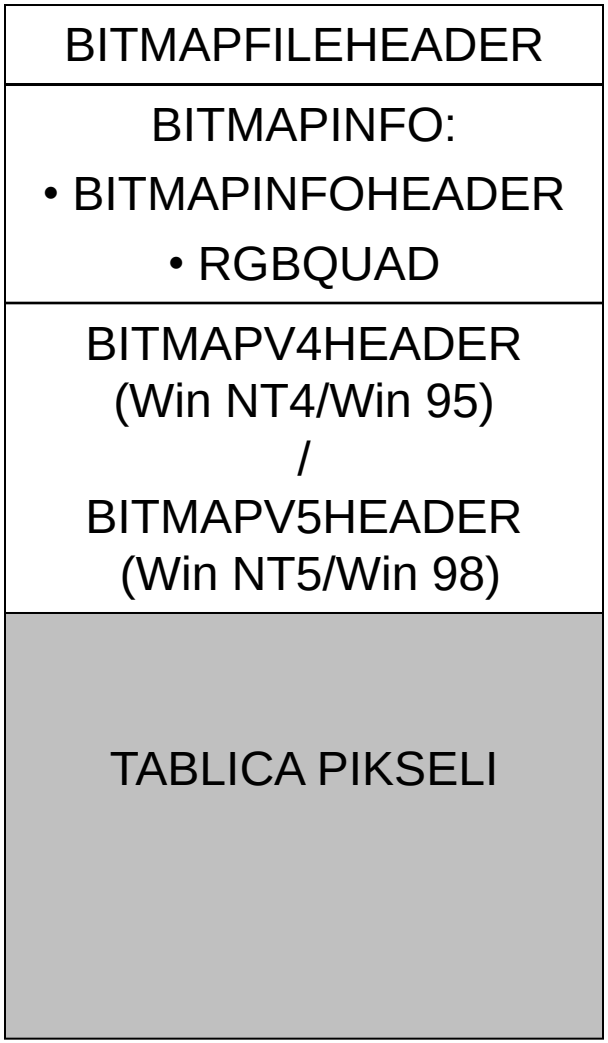
- Mapy bitowe niezależne od sprzętu (DIB – Device-Independent Bitmap – od wersji 3.0), dostosowane do wyświetlania na dowolnym urządzeniu rastrowym.
- Mapy bitowe zależne od sprzętu (DDB – Device-Dependent Bitmap), dostosowane do wyświetlenia na konkretnym urządzeniu wyjściowym.

Metoda wyświetlania plików graficznych w Windows:

- Konwersja dowolnego typu graficznego na mapę bitową DIB
- Przekształcenie mapy DIB w mapę DDB
- Przesłanie mapy DDB na konkretne urządzenie wyświetlające

Formaty plików graficznych – mapy bitowe

Struktura pliku typu BMP:



Formaty plików graficznych – mapy bitowe

Struktura nagłówka BITMAPFILEHEADER:

```
typedef struct tagBITMAPFILEHEADER { // bmfh
        WORD    bfType;
        DWORD   bfSize;
        WORD    bfReserved1;
        WORD    bfReserved2;
        DWORD   bfOffBits;
} BITMAPFILEHEADER;
```

Pole	Wielkość	Opis
bfType	WORD	Dwa bajty "BM" (od słów: mapa bitowa)
bfSize	DWORD	Całkowita wielkość pliku
bfReserved1	WORD	Wartość 0
bfReserved2	WORD	Wartość 0
bfOffBits	DWORD	Odległość od początku pliku do miejsca, w którym zaczynają się bity mapy bitowej.

Formaty plików graficznych – mapy bitowe

Struktura nagłówka BITMAPINFO:

```
typedef struct tagBITMAPINFO { // bmi
   BITMAPINFOHEADER bmiHeader;
   RGBQUAD          bmiColors[1];
} BITMAPINFO;
```

Struktura nagłówka BITMAPINFOHEADER:

```
typedef struct tagBITMAPINFOHEADER{ // bmih
   DWORD  biSize;
   LONG   biWidth;
   LONG   biHeight;
   WORD   biPlanes;
   WORD   biBitCount;
   DWORD  biCompression;
   DWORD  biSizeImage;
   LONG   biXPelsPerMeter;
   LONG   biYPelsPerMeter;
   DWORD  biClrUsed;
   DWORD  biClrImportant;
} BITMAPINFOHEADER;
```

Formaty plików graficznych – mapy bitowe

Struktura nagłówka BITMAPINFOHEADER (interpretacja pól):

Pole	**Wielkość**	**Opis**
biSize	DWORD	Wielkość tej struktury w bajtach
biWidth	LONG	Szerokość mapy bitowej w pikselach
biHeight	LONG	Wysokość mapy bitowej w pikselach
biPlanes	WORD	Wartość 1
biBitCount	WORD	Liczba bitów koloru przypadająca na jeden piksel (1,4,8 lub 24)
biCompression	DWORD	Metoda kompresji (0, gdy nie ma kompresji)
biSizeImage	DWORD	Wielkość mapy bitowej w bajtach (potrzebne jedynie w przypadku map skompresowanych)
biXPelsPerMeter	LONG	Rozdzielczość pozioma w pikselach na metr
biYPelsPerMeter	LONG	Rozdzielczość pionowa w pikselach na metr
biClrUsed	WORD	Liczba kolorów użyta w obrazie
biClrImportant	DWORD	Liczba istotnych kolorów użyta w obrazie

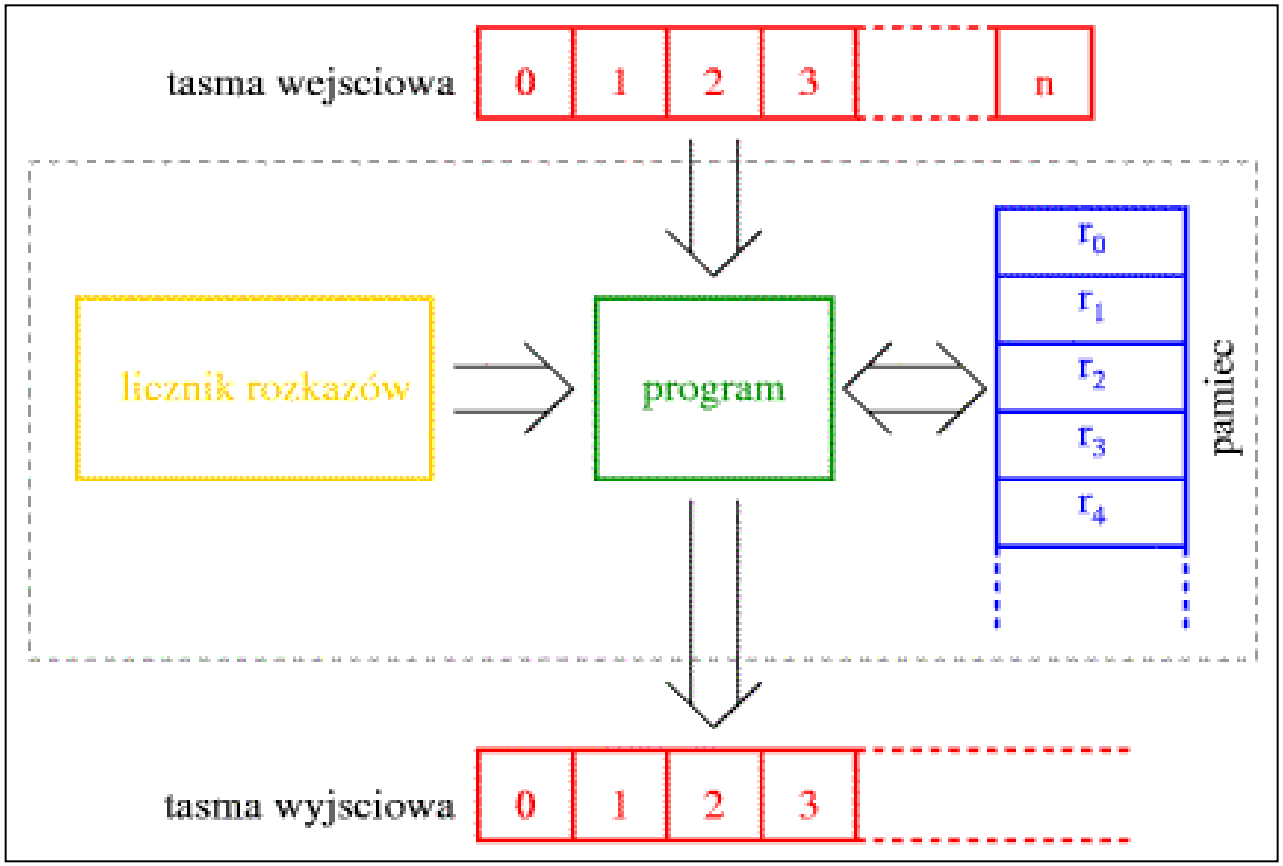
Formaty plików graficznych – mapy bitowe

Struktura nagłówka RGBQUAD:

```
typedef struct tagRGBQUAD { // rgbq
    BYTE    rgbBlue;
    BYTE    rgbGreen;
    BYTE    rgbRed;
    BYTE    rgbReserved;
} RGBQUAD;
```

Pole	Wielkość	Opis
rgbBlue	**BYTE**	**Intensywność barwy niebieskiej**
rgbGreen	**BYTE**	**Intensywność barwy zielonej**
rgbRed	**BYTE**	**Intensywność barwy czerwonej**
rgbReserved	**BYTE**	**Wartość 0**

Matematyczny model procesora – maszyna RAM



Elementy maszyny RAM

- **Program.** Jest to blok reprezentujący kod programu realizowanego przez maszynę, który stanowi zbiór rozkazów do wykonania. Maszyna RAM nie może modyfikować, ani w sposób bezpośrednio odczytywać jego zawartości.
- **Licznik rozkazów.** Wskazuje instrukcję, która ma zostać wykonana w następnym cyklu maszynowym.
- **Pamięć (zespół rejestrów).** Uporządkowany zbiór nieskończonej liczby rejestrów, mogących przechowywać dowolnie duże liczby całkowite. Pierwszy z rejestrów (**r0**), zwany **akumulatorem** lub **sumatorem** jest wyróżniony pod względem funkcjonalnym i stanowi domyślny argument dla większości instrukcji.
- **Taśma wejściowa.** Jest źródłem danych wejściowych. Posiada nieskończoną długość. Może być tylko odczytywana. Pozwala na dostęp sekwencyjny, to znaczy dane są odczytywane kolejno według porządku umieszczenia ich na taśmie, bez możliwości przewijania.
- **Taśma wyjściowa.** Jest miejscem zapisu danych wyjściowych. Posiada nieskończoną długość. Może być tylko zapisywana. Pozwala na dostęp sekwencyjny, to znaczy dane są zapisywane na taśmie kolejno, bez możliwości przewijania.

Instrukcje Maszyny RAM (1)

Nr	mnem. [op.]	Działanie	Opis
1	load =i	r0 := i	Wpisuje liczbę i do sumatora
	load p	r0 := rp	Wpisuje wartość z rejestru rp do sumatora
	load *p	r0 := r[rp]	Wpisuje wartość z rejestru o numerze zawartym w rejestrze rp do sumatora
2	store p	rp := r0	Wpisuje wartość z sumatora do rejestru rp
	store *p	r[rp] := r0	Wpisuje wartość z sumatora do rejestru o numerze zawartym w rejestrze rp
3	add =i	r0 := r0 + i	Dodaje do sumatora wartość i
	add p	r0 := r0 + rp	Dodaje do sumatora wartość z rejestru rp
	add *p	r0 := r0 + r[rp]	Dodaje do sumatora wartość z rejestru o numerze zawartym w rejestrze rp
4	sub =i	r0 := r0 – i	Odejmuje od sumatora wartość i
	sub p	r0 := r0 – rp	Odejmuje od sumatora wartość z rejestru rp
	sub *p	r0 := r0 – r[rp]	Odejmuje od sumatora wartość z rejestru o numerze zawartym w rejestrze rp
5	mult =i	r0 := r0 * i	Mnoży zawartość sumatora przez wartość i
	mult p	r0 := r0 * rp	Mnoży zawartość sumatora przez wartość z rejestru rp
	mult *p	r0 := r0 * r[rp]	Mnoży zawartość sumatora przez wartość z rejestru o numerze zawartym w rejestrze rp

Instrukcje Maszyny RAM (2)

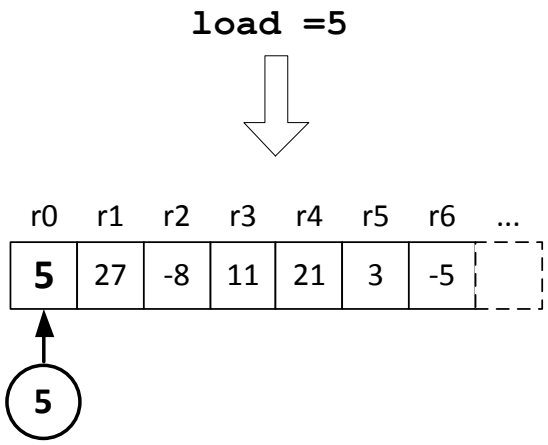
6	**div =i**	**r0 := floor(r0 / i)**	**Dzieli całkowitoliczbowo zawartość sumatora przez wartość i**
	div p	r0 := floor(r0 / rp)	Dzieli całkowitoliczbowo zawartość sumatora przez wartość z rejestru rp
	div *p	r0 := floor(r0 / r[rp])	Dzieli całkowitoliczbowo zawartość sumatora przez wartość z rejestru o numerze zawartym w rejestrze rp
7	read p	rp := TWE	Wczytuje kolejną wartość z taśmy wejściowej do rejestru rp
	read *p	r[rp] := TWE	Wczytuje kolejną wartość z taśmy wejściowej do rejestru o numerze zawartym w rejestrze rp

Instrukcje Maszyny RAM (3)

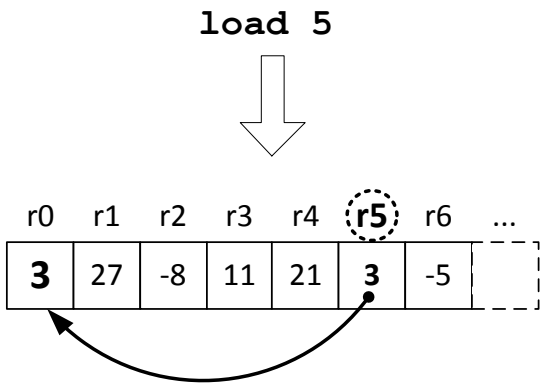
8	write =i	TWY ← i	Zapisuje na taśmie wyjściowej wartość i
	write p	TWY ← rp	Zapisuje na taśmie wyjściowej wartość z rejestru rp
	write *p	TWY ← r[rp]	Zapisuje na taśmie wyjściowej wartość z rejestru o numerze zawartym w rejestrze rp
9	jump etykieta	skocz do etykiety	Skacze bezwarunkowo do miejsca w programie wskazanego etykietą
10	jzero etykieta	skocz do etykiety gdy r0 = 0	Skacze warunkowo do miejsca w programie wskazanego etykietą, gdy wartość sumatora jest zerowa
11	jgtz etykieta	skocz do etykiety gdy r0 > 0	Skacze warunkowo do miejsca w programie wskazanego etykietą, gdy wartość sumatora jest dodatnia
12	halt	zatrzymaj program	

Tryby adresowania

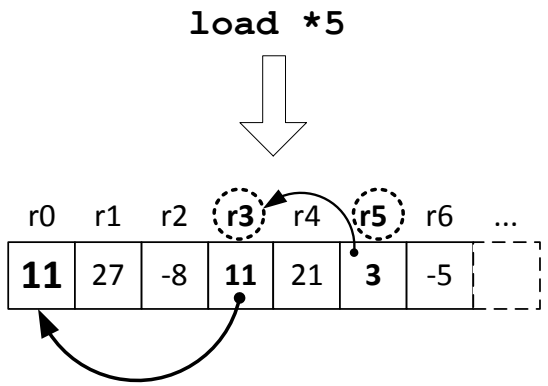
a) adresowanie **natychmiastowe**



b) adresowanie **bezpośrednie**



c) adresowanie **pośrednie**



Przykładowy program

```
1       read 0    # r0 <- TWE(a)
2       read 1    # r1 <- TWE(b)
3       add 1     # r0 <- r0 + r1
4       write 0   # TWY <- r0
5       halt
```

Rejestry: r0 = 23
r1 = 10
r0 = 33

Taśma wejściowa: 23 10

Taśma wyjściowa: 33