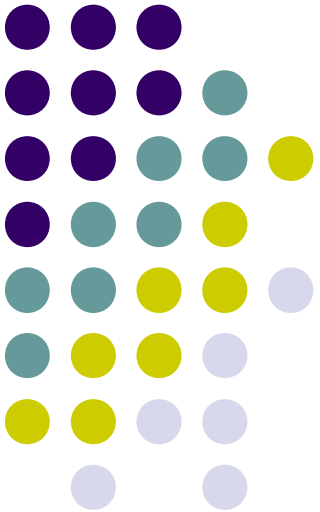
Architektura systemów informatycznych

Architektura i organizacja pamięci



Literatura: Hyde R. 2005, Zrozumieć komputer, Profesjonalne programowanie Część 1, Helion, Gliwice

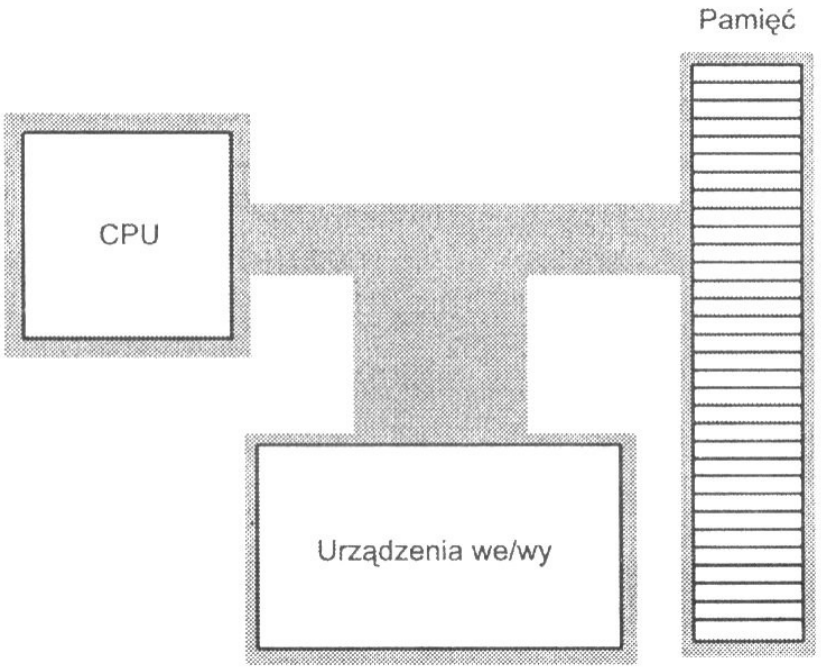
Podstawowe elementy systemu komputerowego

Podstawową obowiązującą obecnie architekturą komputera jest architektura von Neumanna (ang. *von Neumann architekture* -VNA)

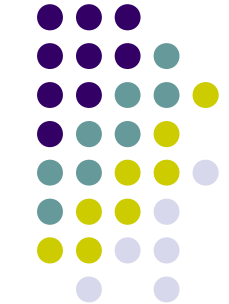
Architektura VNA obowiązuje w rodzinie komputerów 80x86

Podstawowe elementy architektury VNA:

- jednostka centralna (procesor, CPU)
- pamięć
- urządzenia wejścia-wyjścia



Magistrala systemowa (szyna systemowa)



Magistrala systemowa łączy poszczególne składniki maszyny VNA.

Magistrale główne CPU:

Magistrala danych – przeznaczona do wymiany danych między składnikami systemu (magistrale obecnie stosowane: 32- i 64-bitowe)

Magistrala adresowa – przeznaczona do wymiany adresów danych umieszczonych w pamięci, 1 linia – 2 adresy: 0 i 1.
(pierwsze procesory obsługiwały 20 linii adresowych co dawało obsługę 1 MB pamięci 2^{20}, obecnie procesory obsługują ponad 36 linii (2^{36}) co pozwala obsługiwać obszary pamięci ponad 64 GB)

Magistrala sterująca – przeznaczona do przesyłania sygnałów czy dane mają być zapisywane czy odczytywane z pamięci

Fizyczna organizacja pamięci

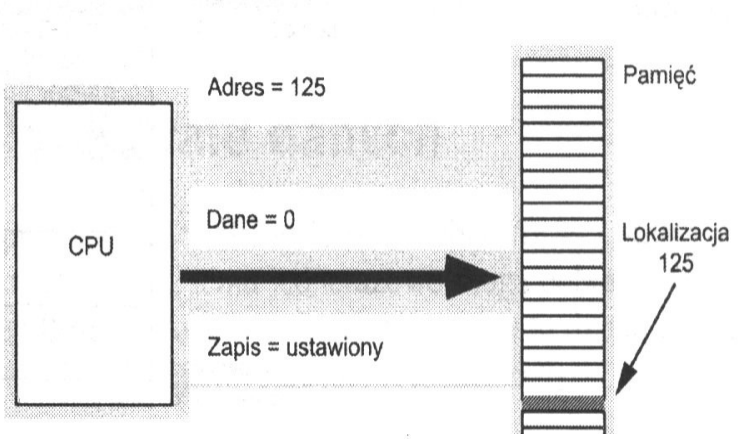
Zapis i odczyt danej jednobajtowej

Deklaracja tablicy bajtów:

```
Pamięc: array[0..1048575] of byte;
```

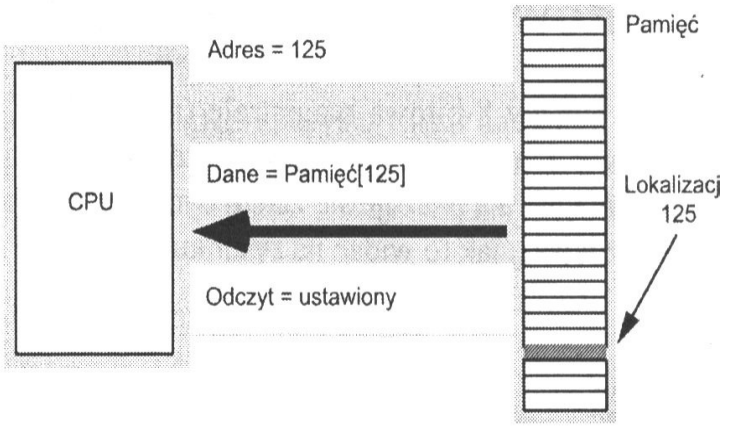
- **Zapis do pamięci**

```
Pamięc[125] := 0;
```



- **Odczyt z pamięci**

```
CPU := Pamięć[125];
```

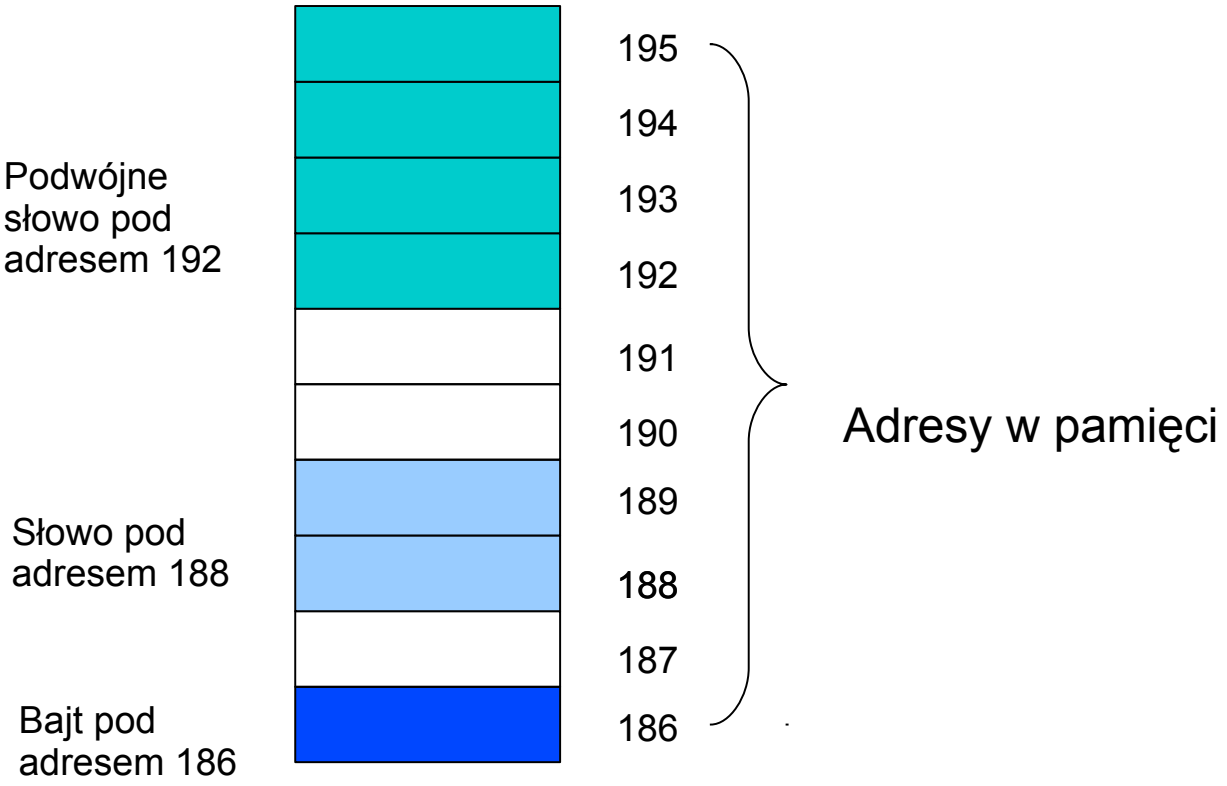


Fizyczna organizacja pamięci

Pamięć jest tablicą bajtów

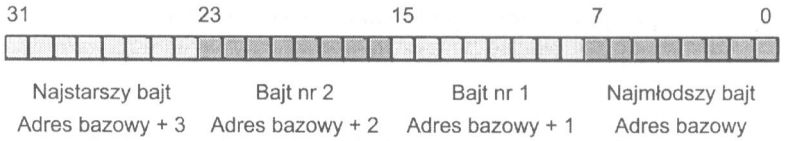
Problem: w jaki sposób zapisać większe ilości danych?

Przykład zapisu danych w pamięci (procesory 80x86)

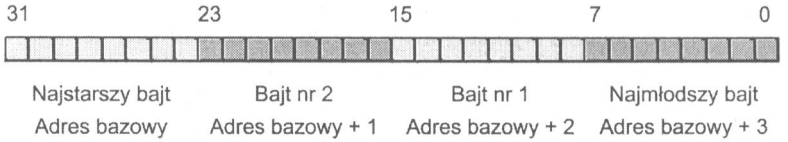


Kolejność zapisu bajtów dla danych 4-bajtowych (podwójne słowo - DWORD)

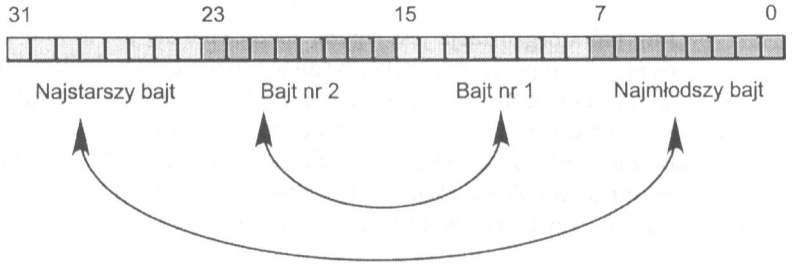
- Układ bajtów w podwójnym słowie (32 - bity) w procesorze Intel 80x86 (little endian)



- Alternatywny układ bajtów w podwójnym słowie Apple Macintosh i zamknięte Unix (big endian)

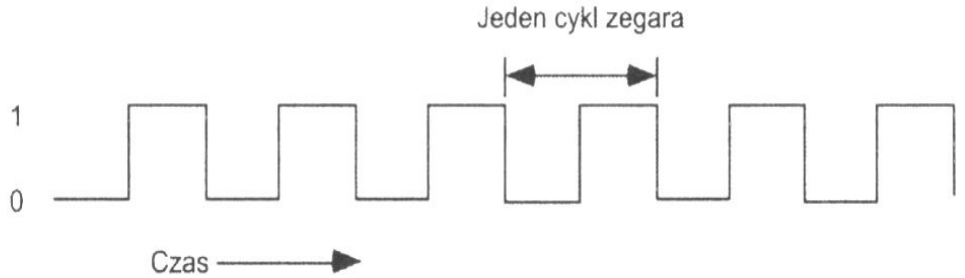


- Konwersja między little ednian a big endian – lustrzane odbicie bajtów obiektu

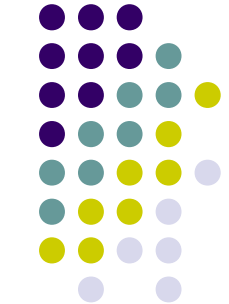


Zegar systemowy

- **Zegar systemowy** to sygnał elektryczny pojawiający się na magistrali sterującej, który regularnie zmienia swój stan między zerem a jedynką. Wszelkie działania procesora są zsynchronizowane z przełączeniami sygnału zegara.
- Częstotliwość, z jaką zegar systemowy zmienia swoje stany, to częstotliwość zegara systemowego, zaś czas przełączenia się z zera na jedynkę i z powrotem na zero to **okres zegara** lub cykl zegara.
- Obecne procesory działają z szybkością 3 – 4 miliardów herców (3 – 4GHz) czyli cykl wynosi około 0,3 ns.

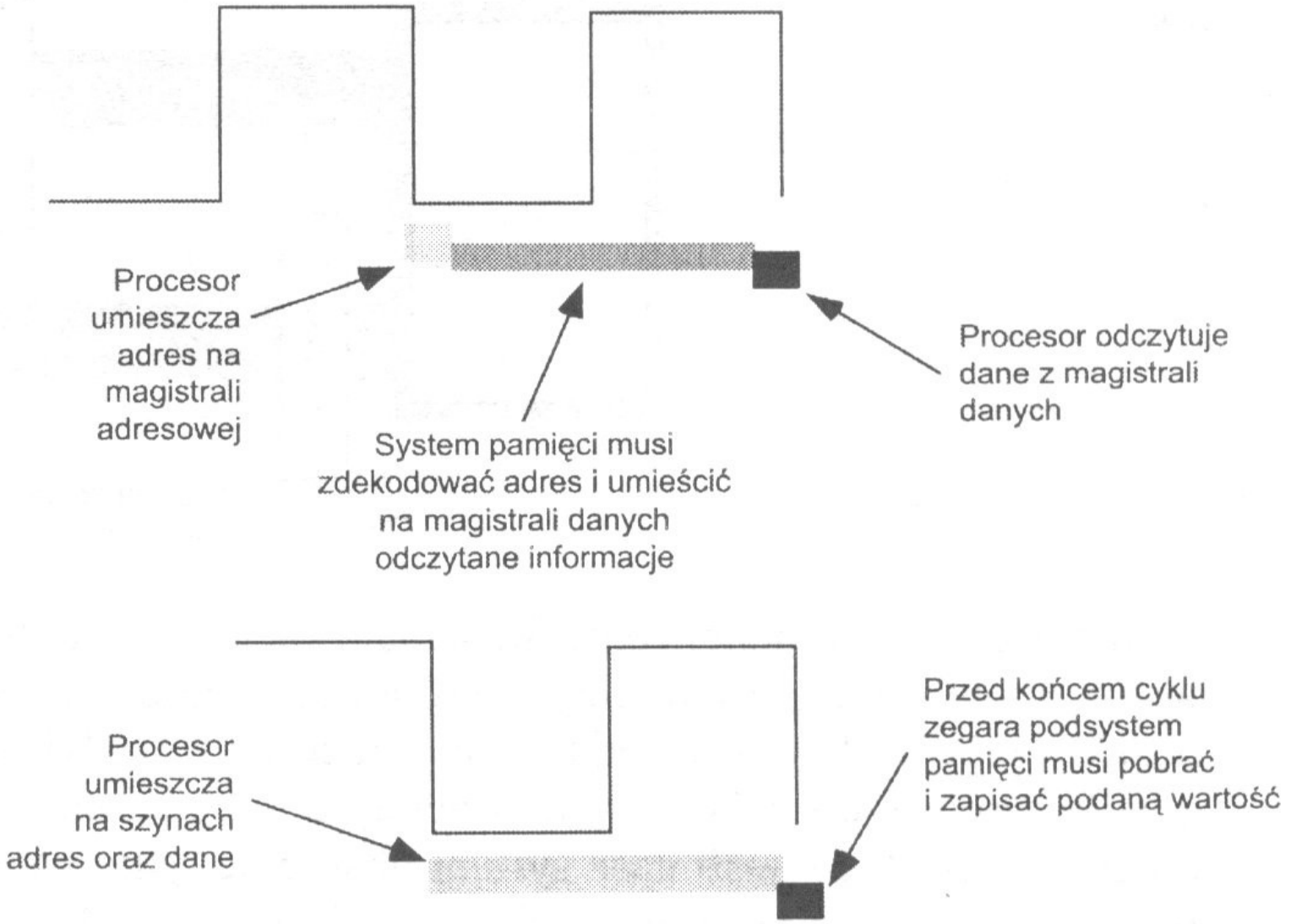


Dostęp do pamięci a zegar systemowy



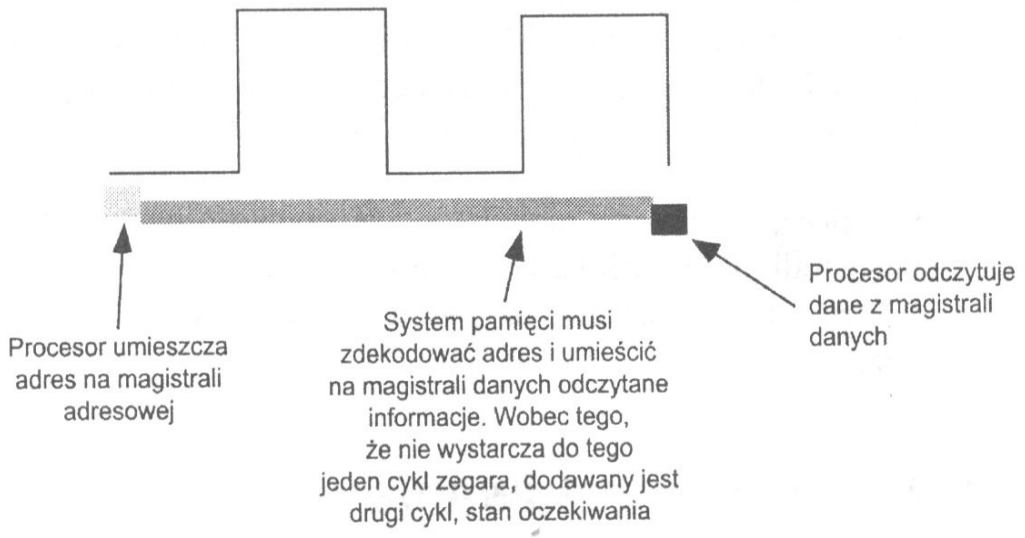
- **Czas dostępu do pamięci** to liczba cykli upływających między żądaniem odczytu lub zapisu pamięci a zakończeniem tej operacji.
- Procesory o szybkości od 100MHz do 4 Ghz mogą używać zegarów magistrali taktujących z częstotliwością 800 Mhz, 500 Mhz, 400 Mhz, 133 Mhz, 100 Mhz i 66 Mhz, są więc o wiele szybsze od pamięci
- Typowy system z procesorem Pentium 3GHz (0,33 ns) korzysta z pamięci RAM taktowanej zegarem 500 MHz (czas dostępu to 2,0 ns). **Musi więc czekać na odpowiedź pamięci.**

Zapis i odczyt z pamięci wykonywany w jednym cyklu zegara procesora



Stany oczekiwania

- Stan oczekiwania to dodatkowy cykl zegara, który pozwala urządzeniu odpowiedzieć na żądanie procesora.
- Wszystkie typowe procesory mają dodatkowe sygnały wyprowadzenie (dołączane do magistrali sterującej), które pozwala wprowadzać stan oczekiwania. W razie potrzeby obwody dekodujące ustawiają ten sygnał, aby dać pamięci potrzebny jej czas.



Pamięć podręczna

Pamięć podręczna inaczej bufor znajduje się pomiędzy procesorem a pamięcią główną.

W przeciwieństwie do zwykłej pamięci poszczególne bajty bufora nie mają stałych adresów, lecz można przypisywać im różne adresy. Dzięki temu procesor może przechowywać dane ostatnio wykorzystywane.

Adresy, z których procesor w ogóle nie korzystał lub z których korzystał stosunkowo dawno, pozostają w powolnej pamięci głównej.

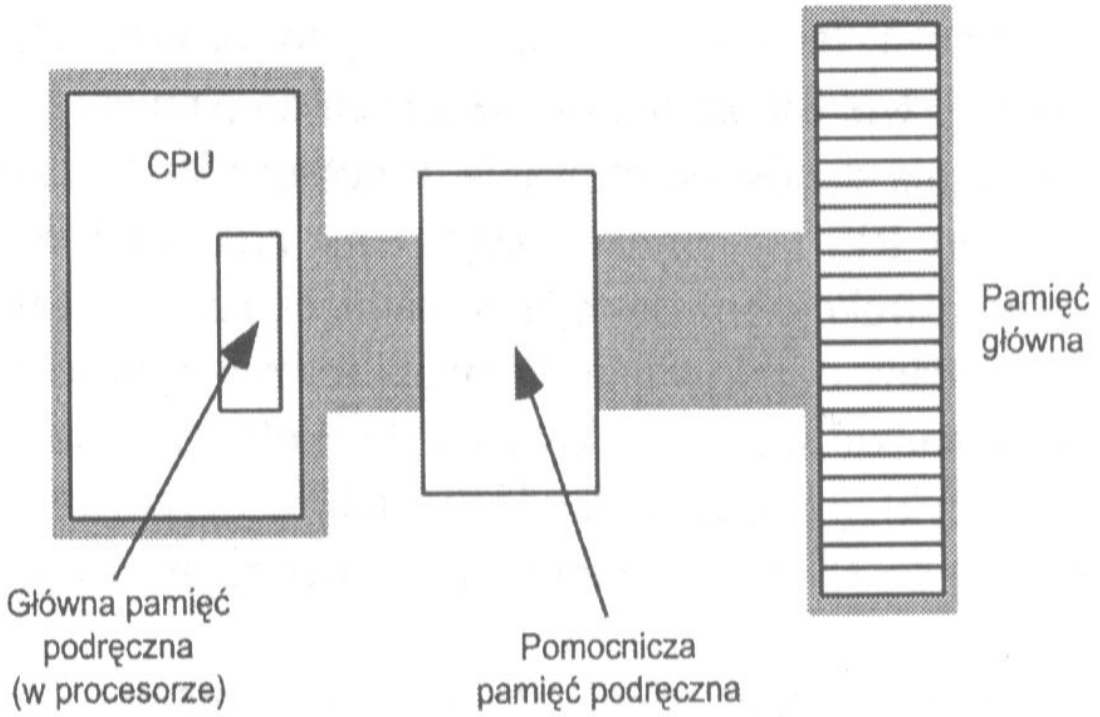
Przykład: pętla

```
for i:=0 to 10 do
        A[i]:=0;
```

Wykorzystuje wielokrotnie te same zmienne: i , tablica A[]

Pamięć podręczna ogranicza problem stanów oczekiwania zwalniających szybkość procesora

Dwupoziomowy system pamięci podręcznej L1 i L2



Dostęp procesora do pamięci

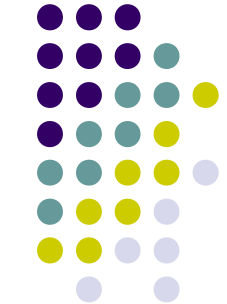
- **Bezpośredni tryb adresowania** – polega na kodowaniu adresu zmiennej w pamięci jako części instrukcji maszynowej sięgającej do tej zmiennej
- **Pośredni tryb adresowania** – używa rejestru do zapamiętania adresu w pamięci. Metoda stosowana przy zmiennych wskaźnikowych.
- **Tryb adresowania indeksowanego** – instrukcje maszynowe kodują zarówno offset (adresowanie bezpośrednie) jak i identyfikator rejestru. Procesor wylicza sumę tych dwóch składników adresu tworząc adres efektywny. Metoda stosowana przy tablicach i strukturach
- **Skalowane indeksowane tryby adresowania** – do adresowania indeksowanego dodana możliwość mnożenia rejestru adresu przez stałą wielkość. Metoda stosowany przy sięganiu do elementów tablic.

Hierarchia pamięci

- Rejestry procesora
- Pamięć podręczna pierwszego poziomu L1
- Pamięć podręczna drugiego poziomu L2
- Pamięć główna RAM
- NUMA
- Pamięć wirtualna
- Pliki dyskowe
- Zasoby sieciowe
- Nośniki zewnętrzne
- Wydruki

Hierarchia pamięci

Jak działa hierarchia pamięci?



- Celem istnienia hierarchii pamięci jest umożliwienie szybkiego dostępu do dużych ilości danych
- Hierarchia ma umożliwić korzystanie z zasad **przestrzennej lokalności odwołań** oraz **czasowej lokalności odwołań** czyli przenoszenie często używanych danych do szybszych podsystemów pamięci, a danych używanych rzadko do podsystemów wolniejszych.
- Procesor najczęściej korzysta z pamięci podręcznej L1 , jeżeli nie znajdzie tam szukanych danych, przekazuje żądanie do pamięci L2. Jeżeli danych nie ma w pamięci L2, to są szukane w pamięci głównej a na końcu w wirtualnej. Za obsługę pamięci wirtualnej odpowiada system operacyjny.

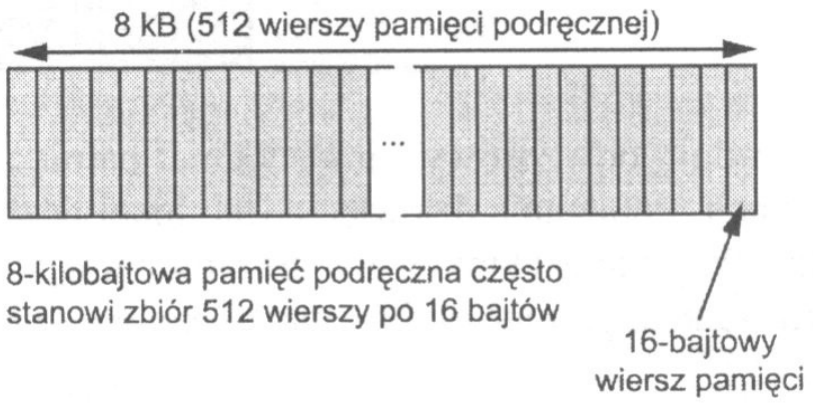
Budowa pamięci podręcznej

Pamięć podręczna jest podzielona na bloki wierszy pamięci podręcznej, przy czym każdy wiersz zawiera ustaloną liczbę bajtów (np. 16, 32, 64).

Wiersz pamięci podręcznej o wielkości np. 16 bajtów zawiera w sobie 16 bajtów z 16 - bajtowego zakresu pamięci głównej.

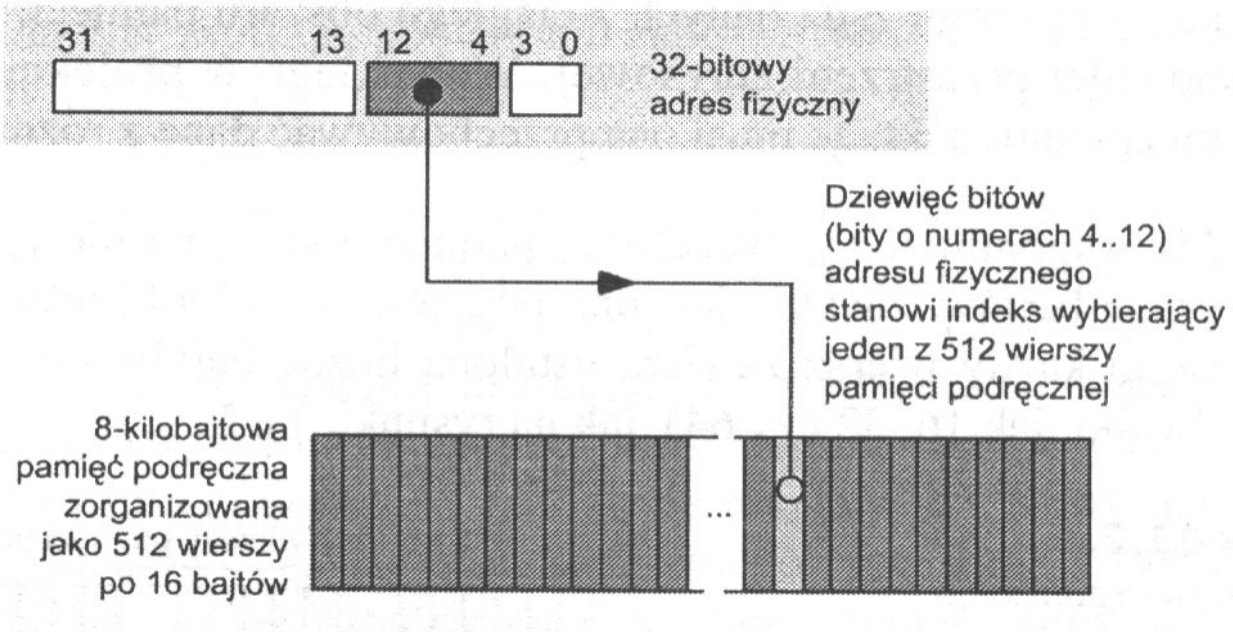
Systemy odczytywania wiersza z pamięci głównej do pamięci podręcznej:

- Pamięć odwzorowana bezpośrednio
- Pamięć w pełni powiązana
- Pamięć powiązana n-krotnie

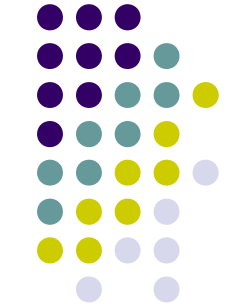


Pamięć odwzorowana bezpośrednio

- Pamięć odwzorowana bezpośrednio charakteryzuje się tym, że blok pamięci głównej jest zawsze ładowany dokładnie do tego samego wiersza pamięci podręcznej
- Liczba 9-bitowa w adresie z pamięci głównej pozwala zaadresować jeden z 512 wierszy (od 0 do 511) pamięci podręcznej



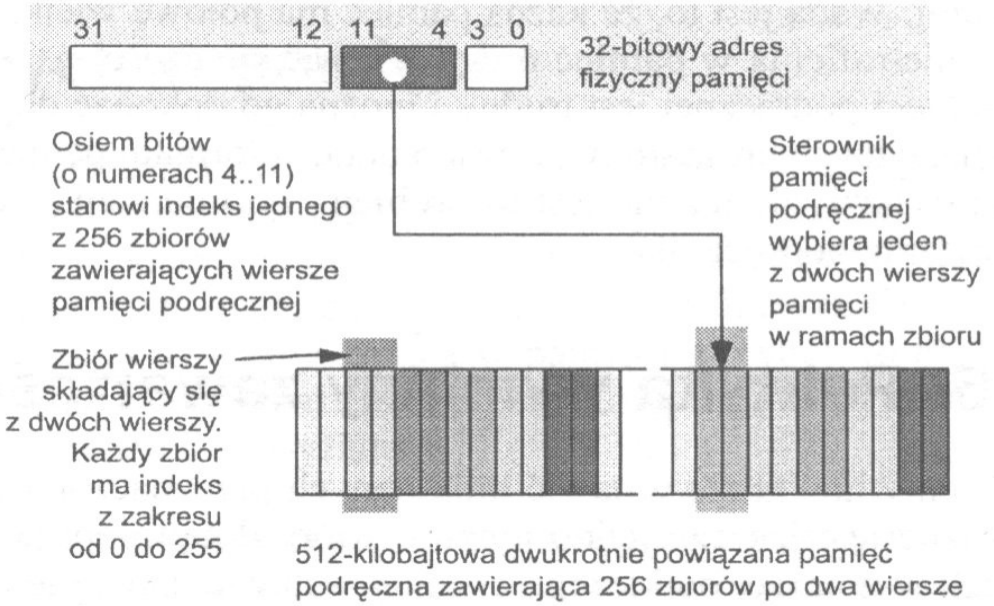
Pamięć w pełni powiązana



- Sterownik pamięci podręcznej może umieścić blok bajtów w dowolnym jej wierszu
- Dodatkowe obwody pozwalające zrealizować takie powiązanie są kosztowne i mogą spowalniać działanie całości dlatego w pamięci L1 i L2 nie stosuje się tej metody

Pamięć powiązana n-krotnie

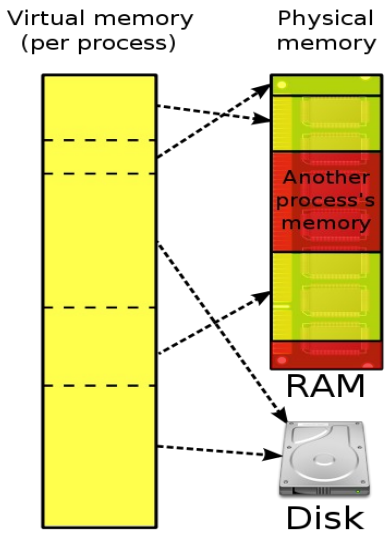
- Pamięć jest podzielona na zbiory wierszy. Procesor wybiera używany zbiór, opierając się na wybranych bitach adresu, jak w przypadku odwzorowania bezpośredniego.
- W każdym zbiorze wierszy znajduje się n wierszy pamięci. Sterownik pamięci podręcznej wykorzystuje algorytm w pełni powiązanego odwzorowania, aby wybrać jeden z n wierszy
- Liczba 8-bitowa pozwala zaadresować 256 zborów wierszy (od 0 do 255)



512-kilobajtowa dwukrotnie powiązana pamięć podręczna zawierająca 256 zbiorów po dwa wiersze

Pamięć wirtualna

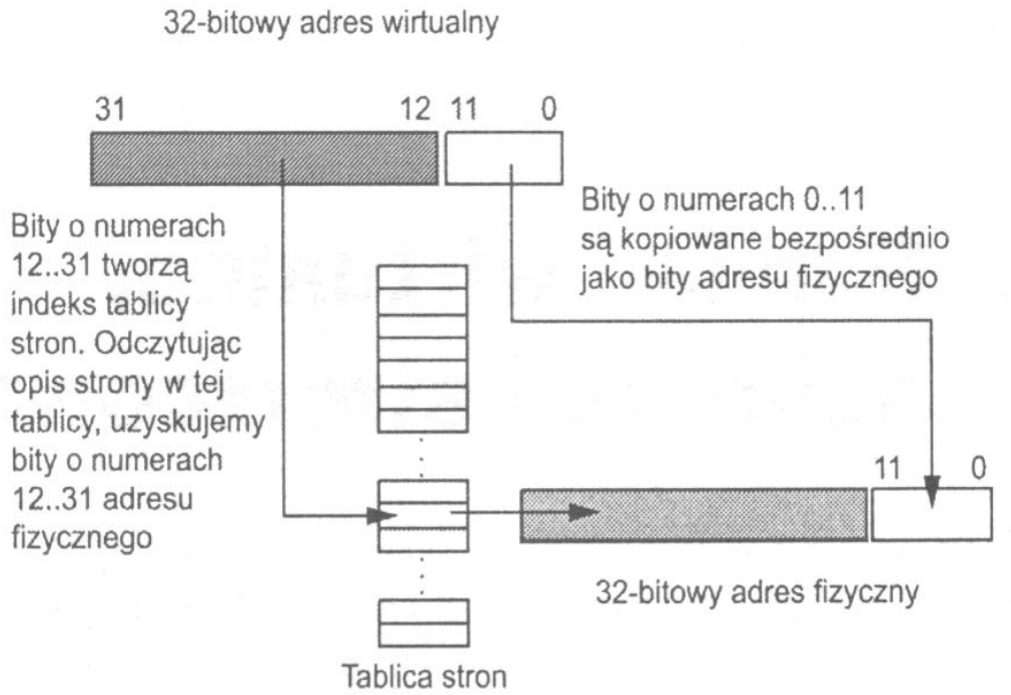
- Pamięć wirtualna procesorów 80x86 udostępnia każdemu procesowi osobną 32-bitową przestrzeń adresową
- Każda aplikacja posiada własną niezależną przestrzeń adresową
- Procesor odwzorowuje adresy wirtualne widziane przez programy na różne adresy fizyczne, proces ten nazywa się **stronicowaniem**
- Pamięć główna jest podzielona na bloki bajtów nazywane **stronami** o wielkości 4096 bajtów.
- Za pomocą tablicy asocjacyjnej odwzorowuje się najmłodsze bity adresu wirtualnego na najmłodsze bity adresu fizycznego, a najstarszych bitów adresu wirtualnego stosuje się jako indeksu do strony



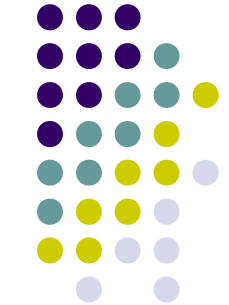
Pamięć wirtualna

Przekształcanie 32 bitowego adresu wirtualnego na adres fizyczny

- 20 bitów starszych adresu wirtualnego indeksuje strony pamięci głównej
- 12 bitów młodszych pozwala adresować przesunięcie wewnątrz strony o wielkości 4096 B

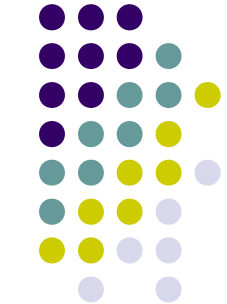


NUMA



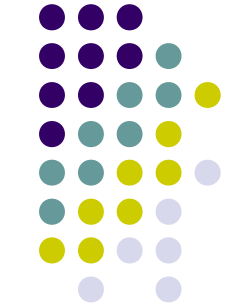
- **NUMA *Non-Uniform Memory Access*** (niejednolity dostęp do pamięci) – różne pamięci o różnych czasach dostępu
- Przykładem takich pamięci jest pamięć karty graficznej lub pamięci flash (pen-drive)
- Typowa karta graficzna łączy się z procesorem magistralą AGP i PCI z szybkością 33 MHz, jest więc w stanie przekazać 132 MB na sekundę.
- Procesor z 64 bitową magistralą taktowaną zegarem 800 MHz może przekazać 6,4 GB na sekundę, jest więc 48 razy szybszy niż w przypadku magistrali PCI. Dane do obsługi grafiki są często buforowane przez pamięć główną zanim zostaną przeniesione do pamięci karty graficznej

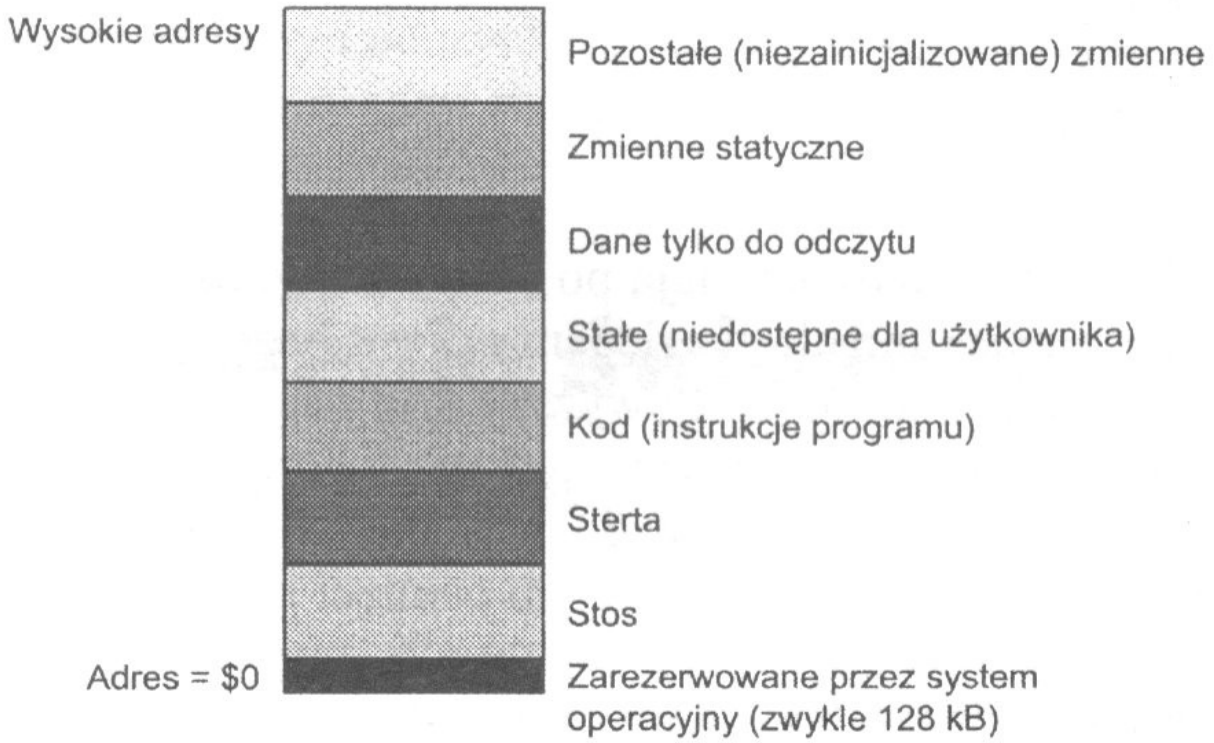
Organizacja pamięci w trakcie działania programu



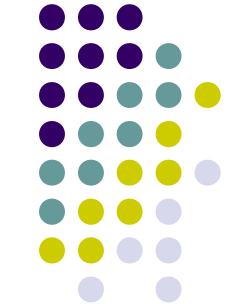
- **Segment kodu**, zawierający instrukcje maszynowe programu
- **Segment stałych**, zawierający wygenerowane przez kompilator dane tylko do odczytu
- **Segment danych tylko do odczytu**, zawierający dane użytkownika przeznaczone tylko do wglądu
- **Segment statyczny**, zawierający inicjalizowane zmienne statyczne zdefiniowane przez użytkownika
- **Segment BSS**, zawierający niezainicjalizowane zmienne zdefiniowane przez użytkownika
- **Segment stosu**, w którym program przechowuje zmienne lokalne i inne tymczasowe dane
- **Segment sterty**, gdzie program umieszcza zmienne dynamiczne

Organizacja pamięci w systemie Windows



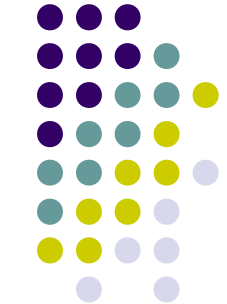


Cechy obiektów w pamięci



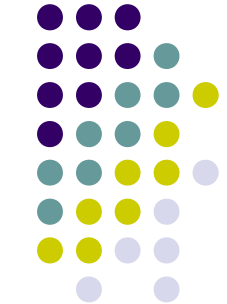
- **Wiązanie** – proces łączenia atrybutu z obiektem, np. przypisywanie wartości do zmiennej
- **Czas życia atrybutu** – czas od pierwszego powiązania atrybutu z obiektem do czasu zerwania tego powiązania lub powiązanie tego atrybutu z innym obiektem
- **Obiekty statyczne** – obiekty, których atrybuty są powiązane z nimi przed wykonaniem aplikacji, jeszcze podczas kompilacji programu
- **Obiekty dynamiczne** – obiekty, które w chwili działania programu mają przypisywane pewne atrybuty

Segment kodu



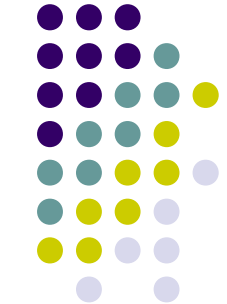
- Segment kodu zawiera instrukcje maszynowe programu.
- Kompilator przekłada te instrukcje zapisane przez programistę na ciąg jedno- lub wielobajtowych wartości.
- Podczas wykonywania programu procesor interpretuje te wartości jako instrukcje maszynowe

Segment stosu



- **Stos** to struktura danych rozszerzająca się i zmniejszająca między innymi w trakcie wywołania procedur i kończenia ich działania
- W czasie działania pracy system umieszcza na stosie wszystkie zmienne automatyczne (niestatyczne zmienne lokalne), parametry funkcji, wartości tymczasowe i inne obiekty
- Większość procesorów realizuje swoje działania na stosie, korzystając z rejestru zwanego wskaźnikiem stosu np. CPU 80x86 – jest to **sprzętowa obsługa stosu**
- Niektóre procesory (np. MIPS Rx000) używają do obsługi stosu tylko rejestr ogólnego przeznaczenia - jest to wtedy **programowa obsługa stosu**

Segment sterty



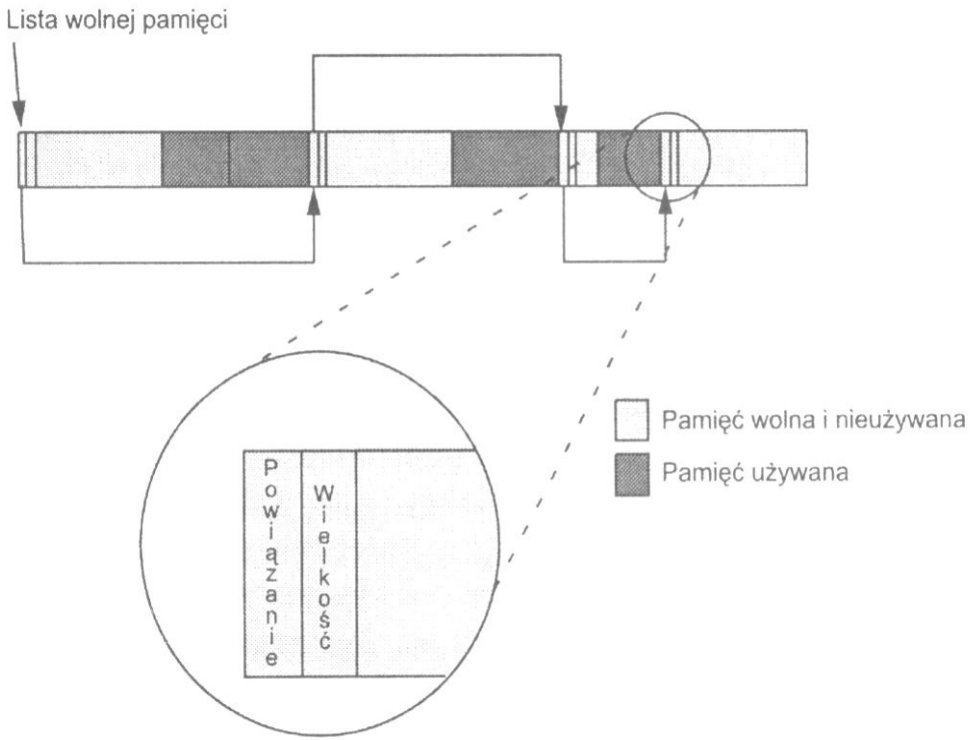
- Obszar pamięci przeznaczony do dynamicznej alokacji zmiennych
- Aplikacja do danych ze sterty odwołuje się przez zmienne wskaźnikowe pozwalające zarezerwować odpowiednią ilość bajtów i przechowujące adres tej pamięci
- Języki programowania używają do tego specjalnych funkcji C (malloc i free), C++(new i delete), Pascal (new i dispose)

Alokacja pamięci

Zarządzanie stertą za pomocą listy bloków wolnej pamięci wykonuje menadżer sterty.

Każdy wolny blok posiada dwie informacje:

- Wielkość wolnego bloku
- Wskaźnik (powiązanie) do następnego wolnego bloku



Fragmentacja pamięci

- Niewłaściwie wykonana alokacja pamięci przez menadżera sterty może doprowadzić do szybkiej fragmentacji i zablokować dostęp do pamięci dla następnych programów