

AB

Architektura systemów komputerowych

Procesor – część II

Rejestry i stos

Rejestry procesora to komórki pamięci o niewielkich rozmiarach umieszczone wewnątrz procesora i służące do przechowywania tymczasowych wyników obliczeń, adresów lokacji w pamięci operacyjnej itd.



Rejestry procesora dostępne programowo

AX	
BX	CX
DX	EX
HX	EX
PC	
SP	
IR	
F	

AX – Akumulator

Zawiera jeden z operandów działania i do niego przekazywany jest wynik

BX,CX,DX,EX,HX,LX – rejesty robocze (uniwersalne) komórki pamięci o niewielkich rozmiarach (najczęściej 4/8/16/32/64/128/256 bitów) umieszczone wewnątrz procesora i służące do przechowywania tymczasowych wyników obliczeń, adresów lokacji w pamięci operacyjnej itd.

IR - rejestr rozkazów
PC - licznik rozkazów

Sterują
pracą
programu

F - rejestr stanu (rejestr flag)

SP - wskaźnik stosu

Rejestry procesora dostępne programowo

DEFINICJE:

Licznik rozkazów (PC) - rejestr mikroprocesora zawierający adres komórki pamięci, w której przechowywany jest rozkaz przeznaczony do wykonywania jako następny

Rejestr rozkazów (IR) - jest częścią jednostki kontrolnej procesora, w której przechowywana jest aktualnie wykonywana instrukcja maszynowa.

Rejestr stanu (flag) (F) - rejestr procesora opisujący i kontrolujący jego stan. Jest to rejestr bitowy – każdy z jego bitów ma pewne znaczenie

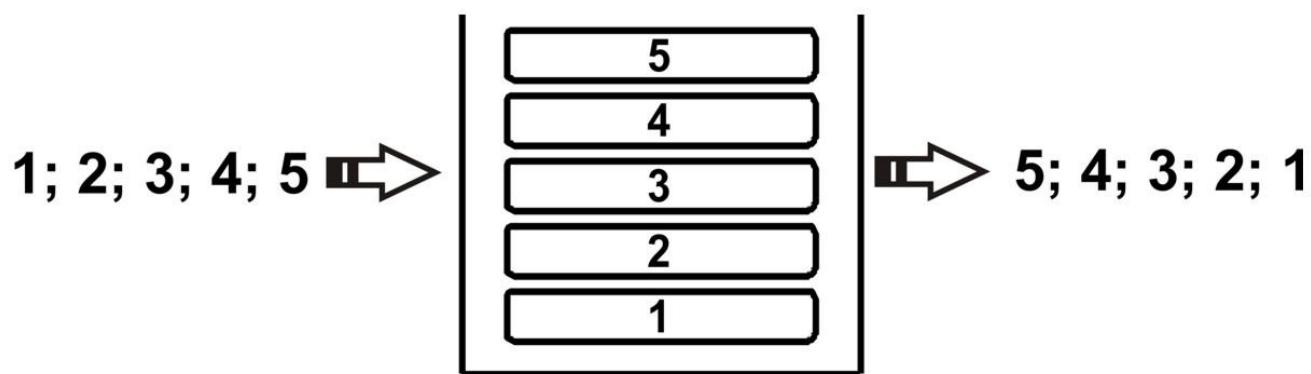


Wskaźnik stosu (SP) – służy do wskazywania szczytu stosu adresu ostatniego elementu na stosie)

Stos

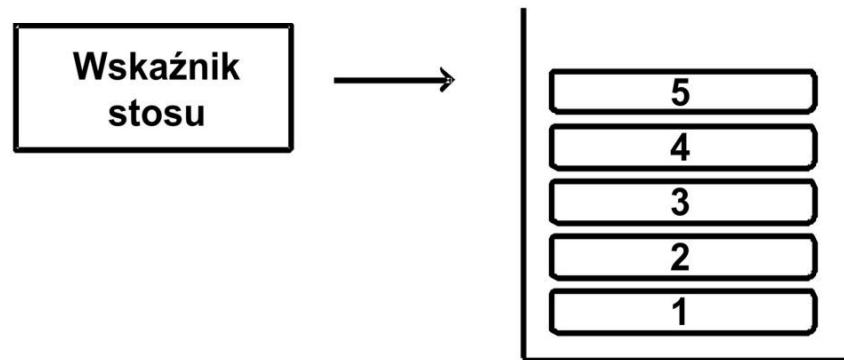
Stosem nazywany wyróżniony obszar pamięci używany według następujących reguł:

1. Informacja zapisywana jest do kolejnych komórek, żadnego adresu nie wolno pominąć,
2. Odczyt informacji następuje w kolejności odwrotnej do zapisu,
3. Informuje odczytujemy z ostatnio zapisanej komórki, natomiast zapisujemy do pierwszej wolnej.



Stos

- ✓ Stos jest rodzajem pamięci (buforem) typu LIFO (ang. *Last In First Out*).
- ✓ Stos ma stały wymiar i położenie w pamięci. Konieczna jest więc tylko znajomość adresu pierwszej wolnej komórki stosu (wierzchołka stosu). Do tego służy rejestr SP - wskaźnik stosu.





Rejestry procesora dostępne programowo

Na stosie zapamiętywane są stany rejestrów w trakcie przełączania się procesora pomiędzy różnymi wątkami (zadaniami).

Gdy procesor wykonuje program, w jego rejestrach znajdują się istotne informacje – wyniki częściowe wykonywanych operacji, adres następnego rozkazu czekającego na wykonanie i wiele innych.

Przełączenie się na obsługę innego programu, czy też przejście do jednej z procedur obsługi przerwań (umożliwiających między innymi obsługę urządzeń wejścia-wyjścia), wymaga usunięcia tych danych i zastąpienia ich nowymi.

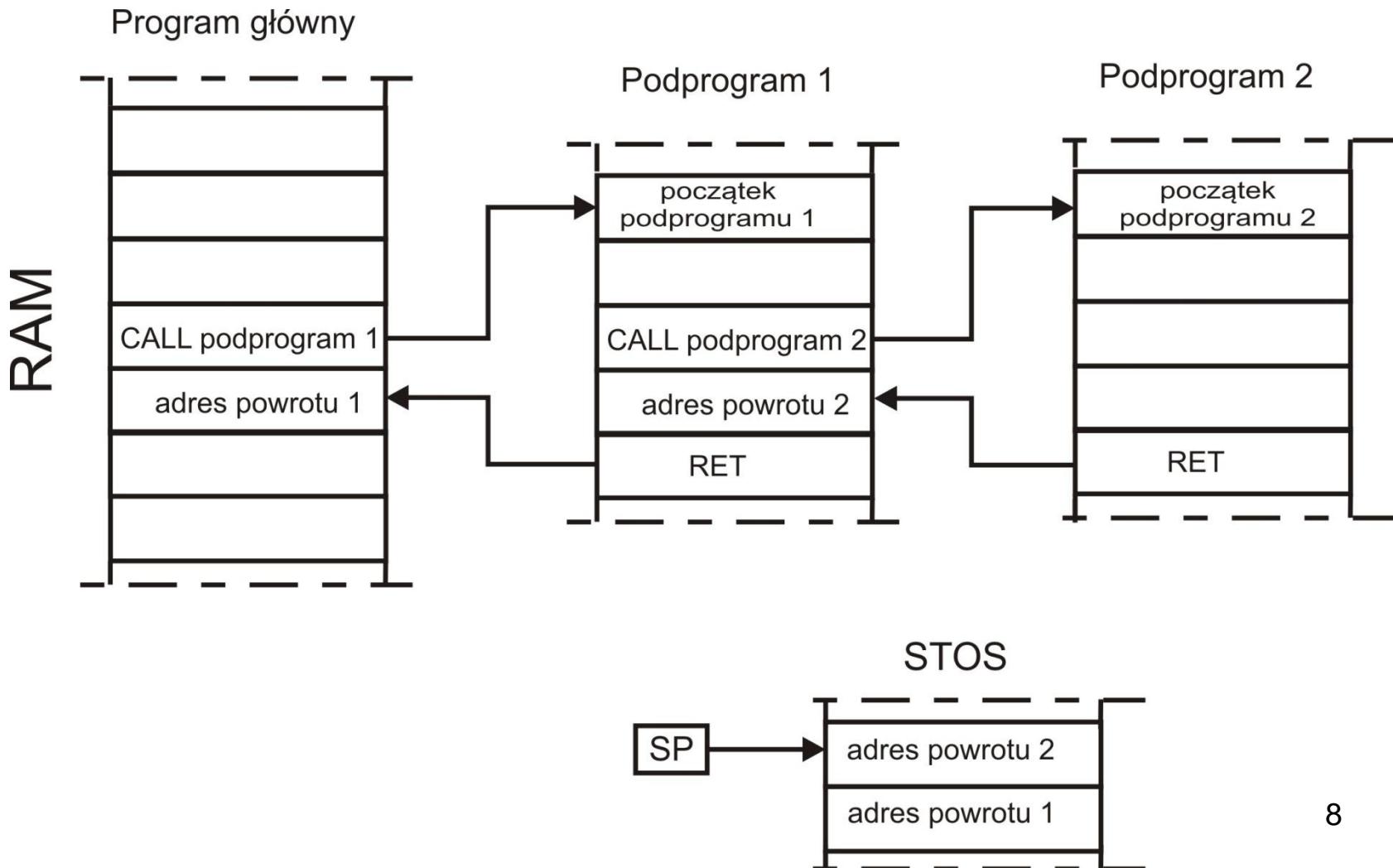
Poprzednia zawartość rejestrów zapamiętywana jest na stosie i może być później przywrócona.

Procesor odkłada na stos zawartość rejestrów w określonej kolejności, a odczytuje w kolejności odwrotnej, dzięki temu nie ma potrzeby zapisywania z którego rejestru pochodziła każda wartość.



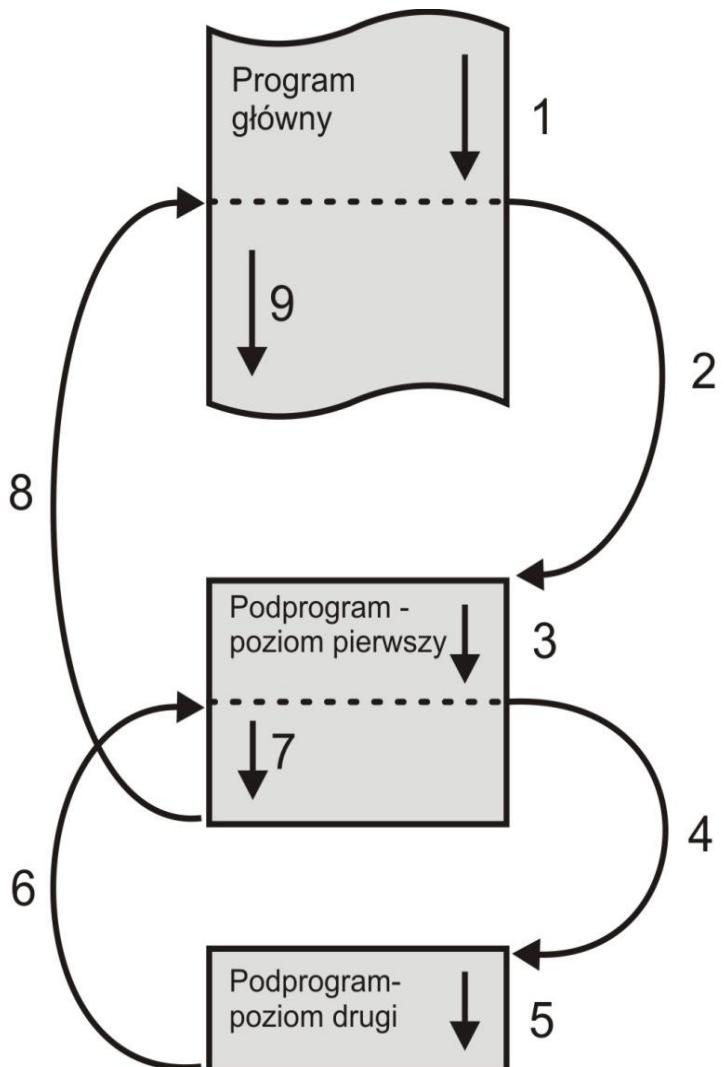
Rejestry procesora dostępne programowo

Stos przechowuje także adresy powrotów przy wywołaniu podprocedur. Dzięki budowie stosu (kolejki LIFO) zapamiętywane są adresy oraz, niejako automatycznie, ich kolejność.





Wykorzystanie stosu do wywołania podprogramów



1. Procesor wykonuje program.
2. Napotyka rozkaz skoku, odkłada na stos adres kolejnego rozkazu i przechodzi do podprogramu.
3. Procesor wykonuje podprogram.
4. Napotyka rozkaz skoku, odkłada na stos adres kolejnego rozkazu i przechodzi do podprogramu następnego poziomu.
5. Wykonuje podprogram drugiego poziomu.
6. Procesor pobiera z wierzchołka stosu adres i powraca do ostatnio wykonywanej procedury.
8. Procesor pobiera z wierzchołka stosu adres i powraca do programu głównego.
9. Procesor kontynuuje program główny.

Przetwarzanie potokowe

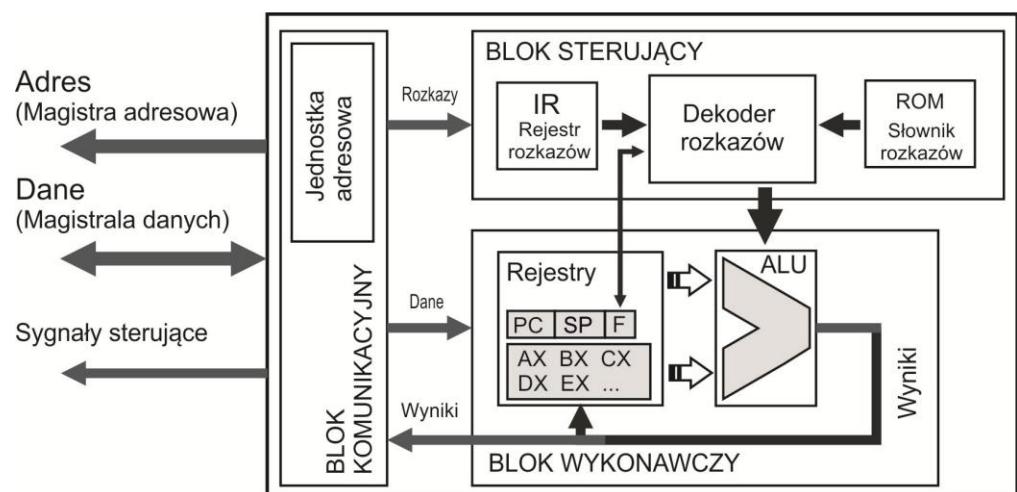
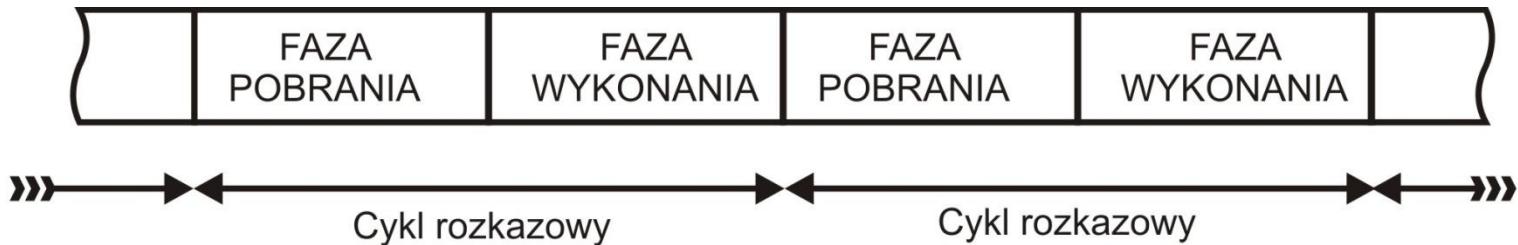
Przetwarzanie potokowe – jeden ze sposobów sekwencyjnego (szeregowego) przetwarzania danych.



Zasada przetwarzania potokowego

Przypomnienie: W najprostszej wersji cykl rozkazowy procesora składa się z dwóch faz:

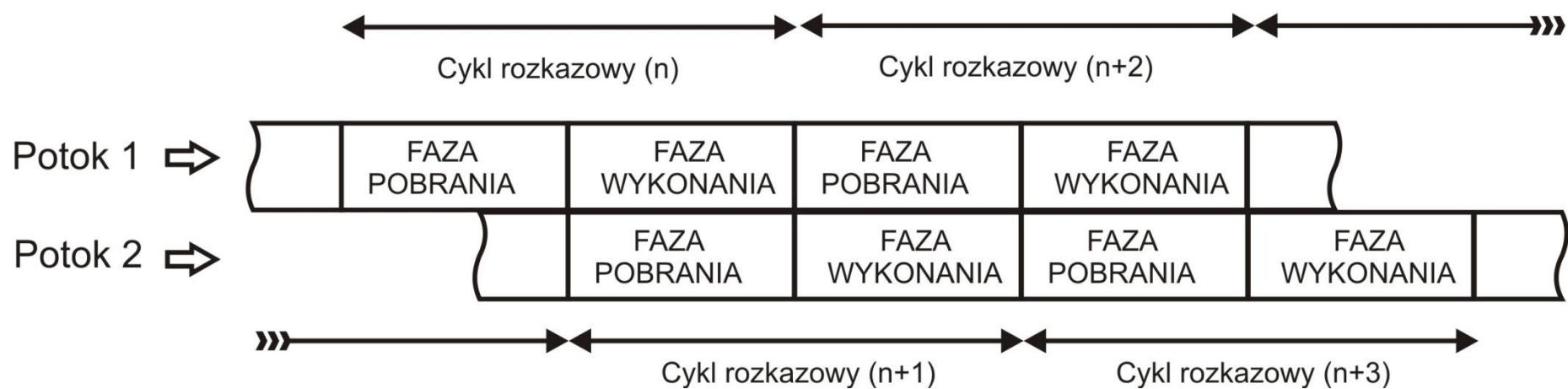
- fazy pobrania
- wykonania rozkazu





Dwustopniowy potok rozkazów

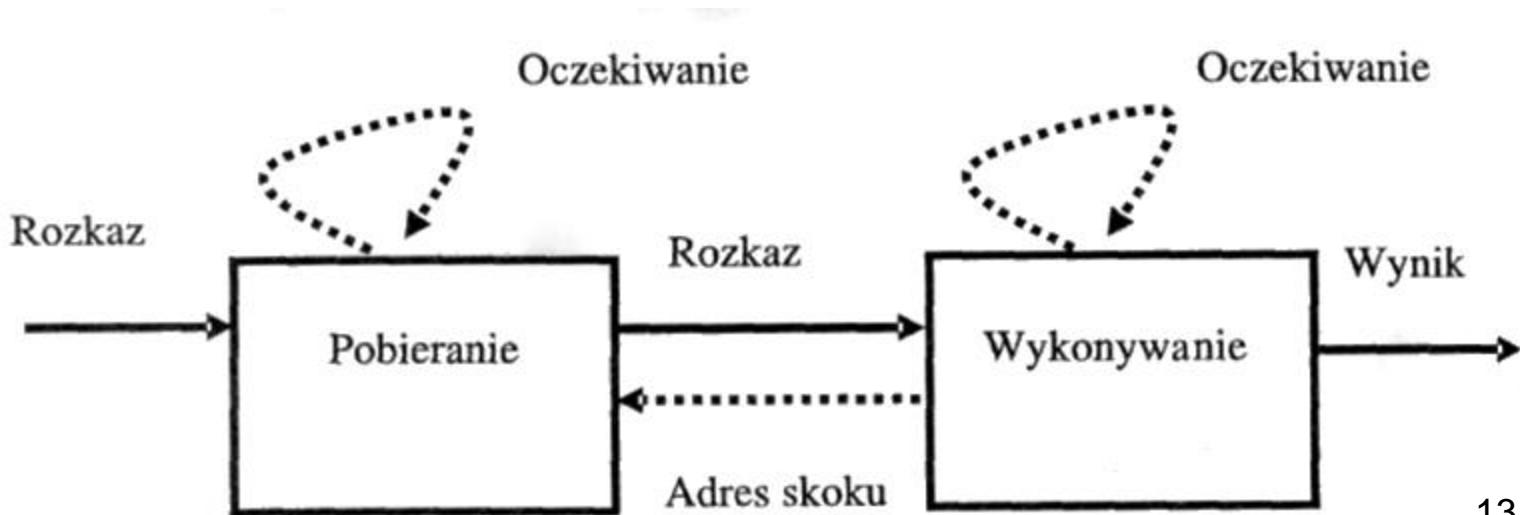
Jeżeli w procesorze wydzielić dwa „stanowiska obsługi” - jedno dla pobierania rozkazów, a drugie dla ich wykonywania, wówczas można równocześnie realizować obie fazy cyklu rozkazowego: pobieranie następnego rozkazu odbywa się w czasie, gdy jest wykonywany rozkaz poprzedni .
Takie działanie, analogiczne do obróbki na taśmie produkcyjnej, nazywa się przetwarzaniem potokowym (pipeline).



Zakładając, że czas trwania każdej fazy cyklu jest taki sam, uzyskuje się dwukrotne skrócenie czasu kończenia kolejnych rozkazów (uzyskiwania wyników), mimo że sumaryczny czas cyklu pozostaje niezmieniony.

Powody przestojów w przetwarzaniu potokowym

- Wystąpienie w programie rozgałęzień zależnych od efektów poprzedzającej operacji (skoków warunkowych) - w takim przypadku kolejny pobrany rozkaz może okazać się nieprzydatny i trzeba pobierać inny, wskazany przez wykonywany rozkaz skoku. Następuje wówczas opróżnienie potoku (*flush*) powodujące chwilowy przestój jednostki wykonawczej,
- Z kolei przestoje jednostki pobierającej rozkazy pojawiają się wtedy, gdy samo wykonanie rozkazu wymaga kontaktu z pamięcią dla odczytania lub zapisania argumentów.





Potoki wielostopniowe

Współczesne procesory stosują wielostopniowe przetwarzanie potokowe (liczba stopni wynosi od kilku do kilkunastu), a dla zredukowania strat wydajności wprowadza się rozwiązania sprzętowo-programowe, takie jak równoczesny dostęp do pamięci rozkazów i pamięci danych czy dynamiczne przewidywanie skoków.



Potok pięciostopniowy

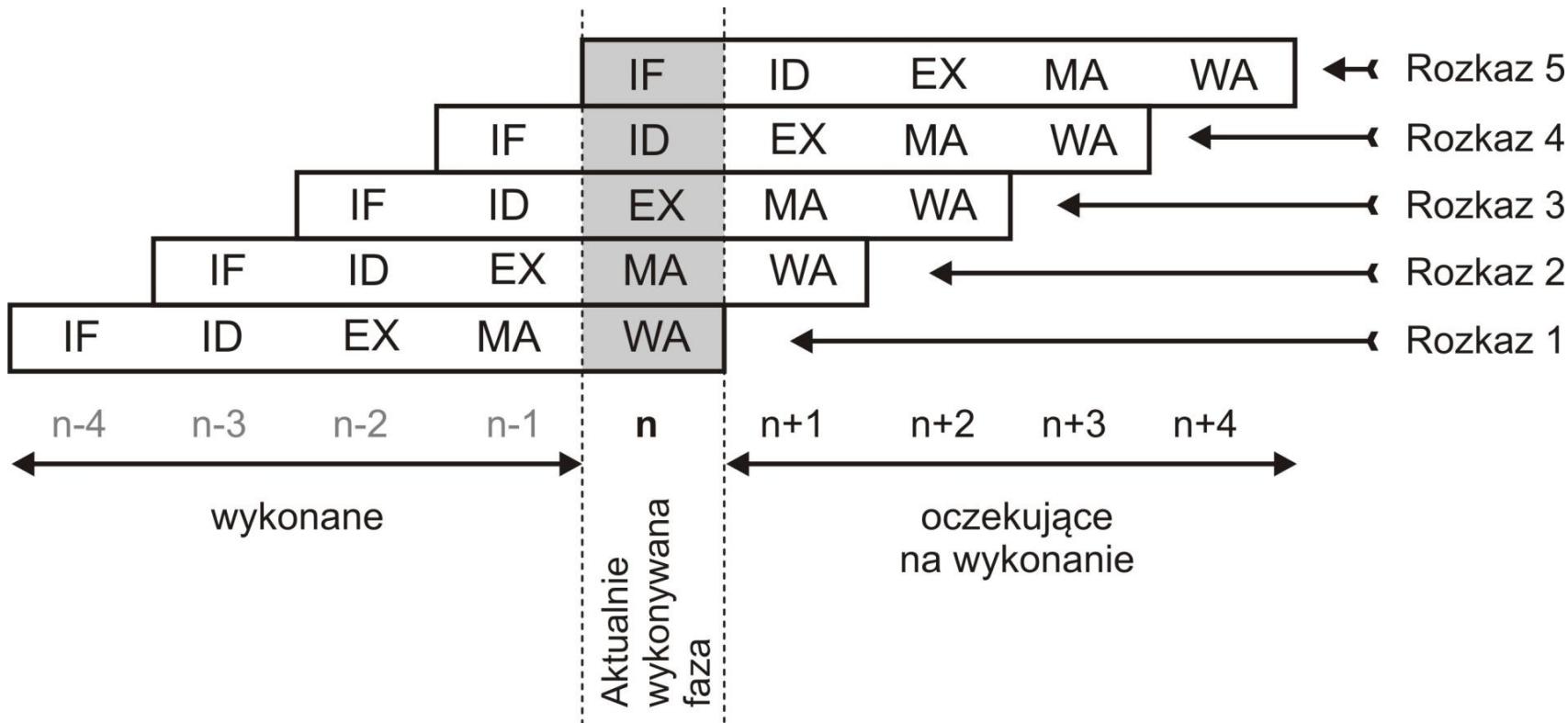
Dla potoku pięciostopniowego (stosowanego między innymi w procesorze Intel Pentium) wyróżniamy następujące fazy:

1. **IF** (ang. instruction fetch) – **faza pobierania** – procesor pobiera rozkaz z I-Cache (ang. Instruction Cache), czyli bloku pamięci podręcznej zawierającego instrukcję.
2. **ID** (ang. instruction decode) – **faza dekodowania** – blok dekodowania rozpoznaje rozkaz, na jego podstawie generuje instrukcje sterujące do bloku wykonawczego oraz inicjuje pobranie argumentów operacji,
3. **EX** (ang. execute) – **faza wykonania** – w tej fazie jednostka arytmetyczno-logiczna wykonuje odpowiednie działania na dostarczonych do niej danych,
4. **MA** (ang. memory access) – **operacje na pamięci** – procesor pobiera dane z pamięci D-Cache (ang. Data Cache), czyli bloku pamięci podręcznej zawierającej dane.
5. **WB** (ang. write back) – **zapis do rejestrów** – wyniki zapisane są do akumulatora lub wskazanego rejestru.

Każda z wymienionych faz wykonywana jest w osobnym bloku procesora.



Potok pięciostopniowy



W każdym cyklu jest przetwarzanych jednocześnie 5 rozkazów znajdujących się na różnych etapach realizacji (a kolejne rozkazy są kończone w każdym takcie zegara).



Konflikty w przetwarzaniu potokowym

Idea przetwarzania potokowego opiera się na założeniu, że **instrukcje programu zapisane są w pamięci operacyjnej pod kolejnymi adresami** i mogą być wykonywane równolegle, niezależnie od siebie. Założenie to sprawdza się dla ponad 90% dowolnego kodu.

W każdym jednak programie zdarzają się sytuacje, w których mechanizm przetwarzania potokowo nie działa w pełni wydajnie. Sytuacje takie nazywamy **konfliktami przetwarzania potokowego**.

Wyróżniamy trzy rodzaje konfliktów:

1. **konflikty sterowania,**
2. **konflikty danych,**
3. **konflikty zasobów.**

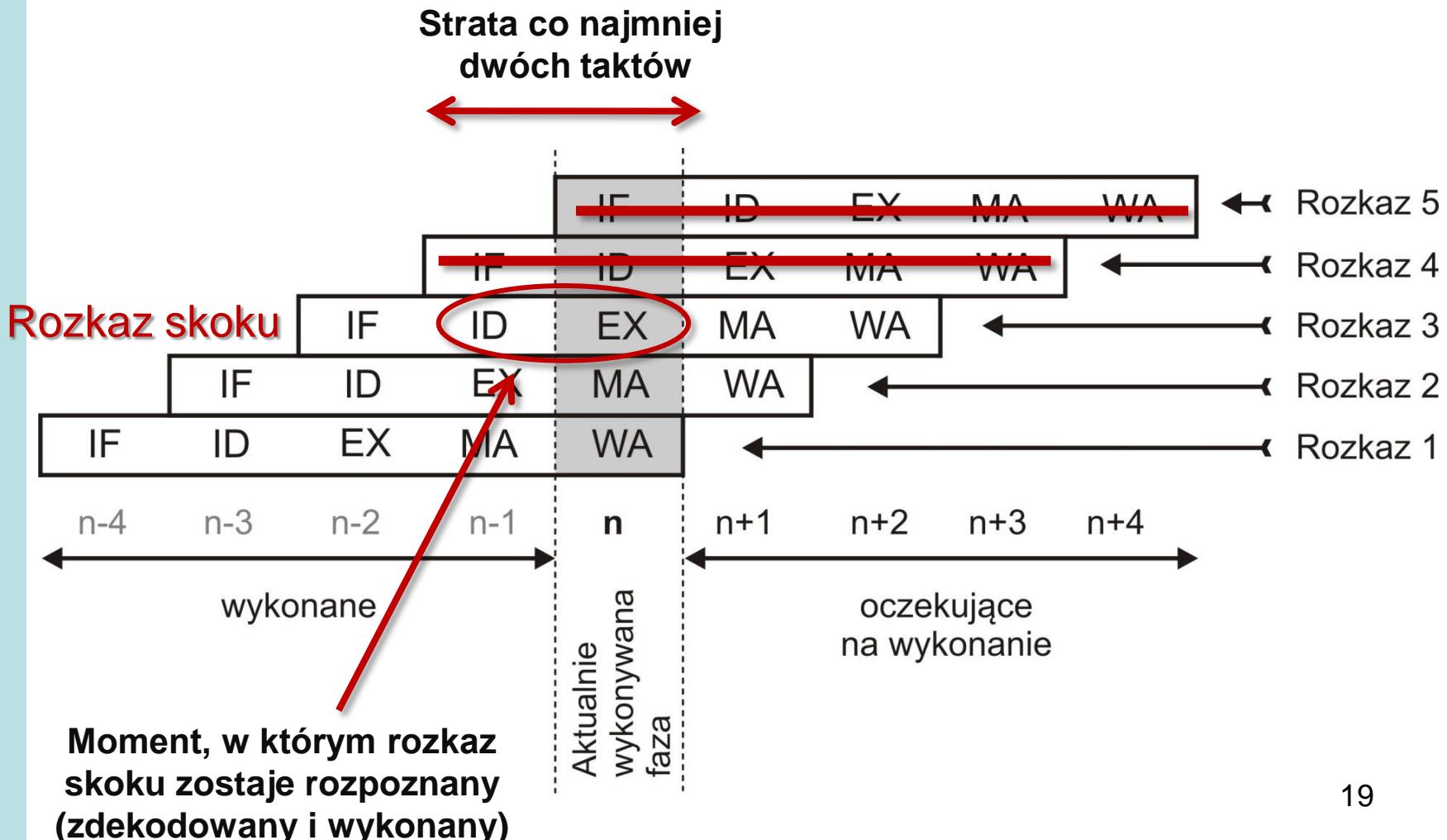


Konflikty w przetwarzaniu potokowym

Konflikty sterowania (ang. control hazard) to zaburzenia sekwencji wykonywanego programu wywołane przez rozgałęzienia programu.

- ✓ Instrukcje programu zapisane są zwykle w kolejno po sobie następujących komórkach pamięci. Dzięki temu procesor może, jeszcze w trakcie wykonywania poprzedniej instrukcji, pobrać następną, znajdująca się pod kolejnym adresem. Pojawienie się instrukcji skoku oznacza jednak, że następne polecenie powinno być pobrane z całkiem innego miejsca w pamięci.

Konflikty sterowania powoduje to konieczność zapełnienia kolejki rozkazów od nowa i skutkuje stratą kilku taktów.





Konflikty w przetwarzaniu potokowym

Jednym z rozwiązań sprzętowych optymalizujących pracę procesora jest **bufor adresu docelowego BTB** (ang. branch target buffer), służący do przewidywania skoków.

Mechanizm BTB próbuje przewidzieć, czy program wykona skok, a następnie pobiera odpowiednie instrukcje.

Wykorzystywane są dwie metody prognozowania skoków –

- prognozowanie statyczne, oparte na analizie kodu rozkazu i rozpoznawaniu skoków bezwarunkowych
- dynamiczne, oparte na historii przetwarzania i analizie szansy wykonania rozgałęzienia.

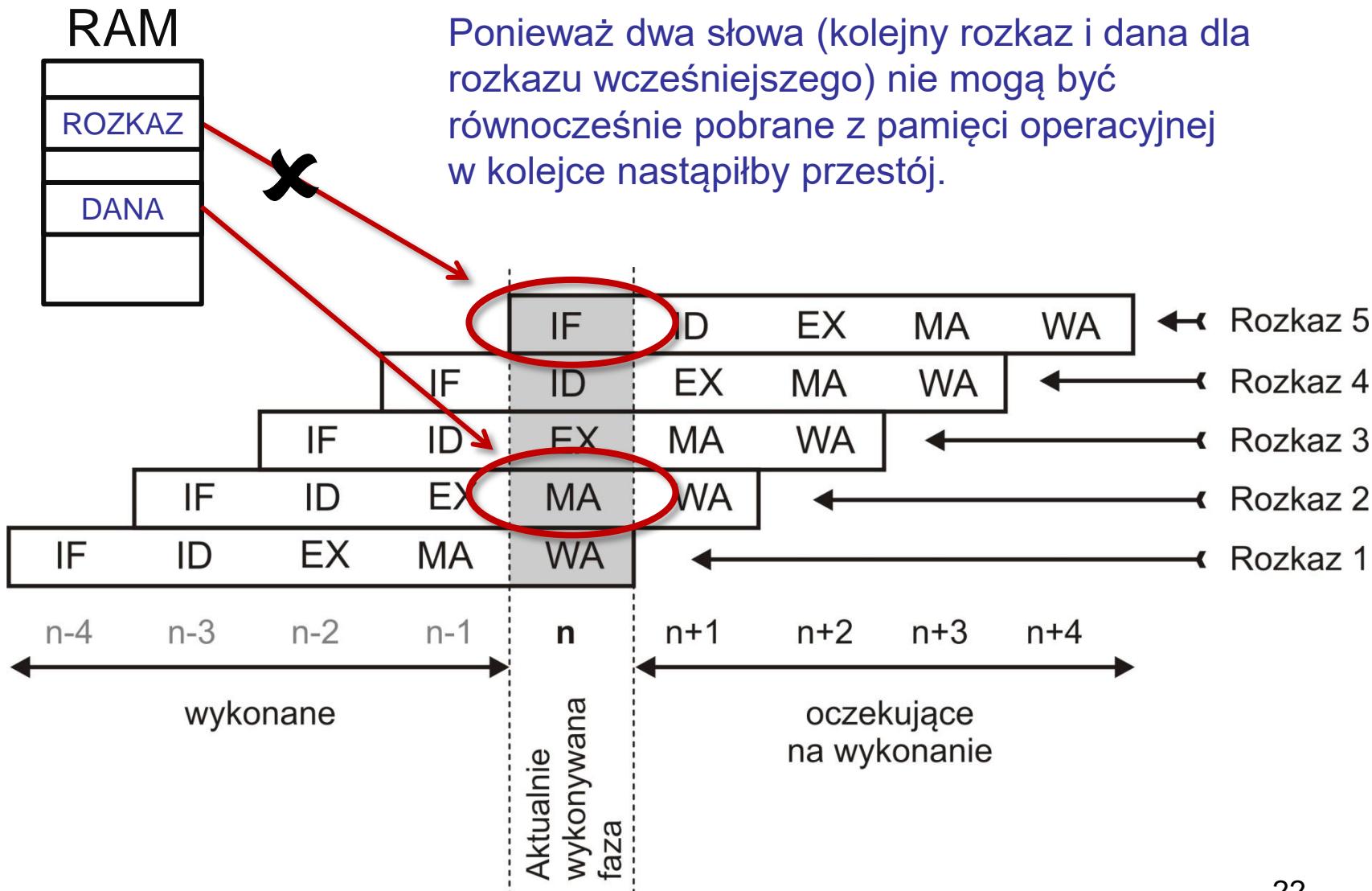


Konflikty w przetwarzaniu potokowym

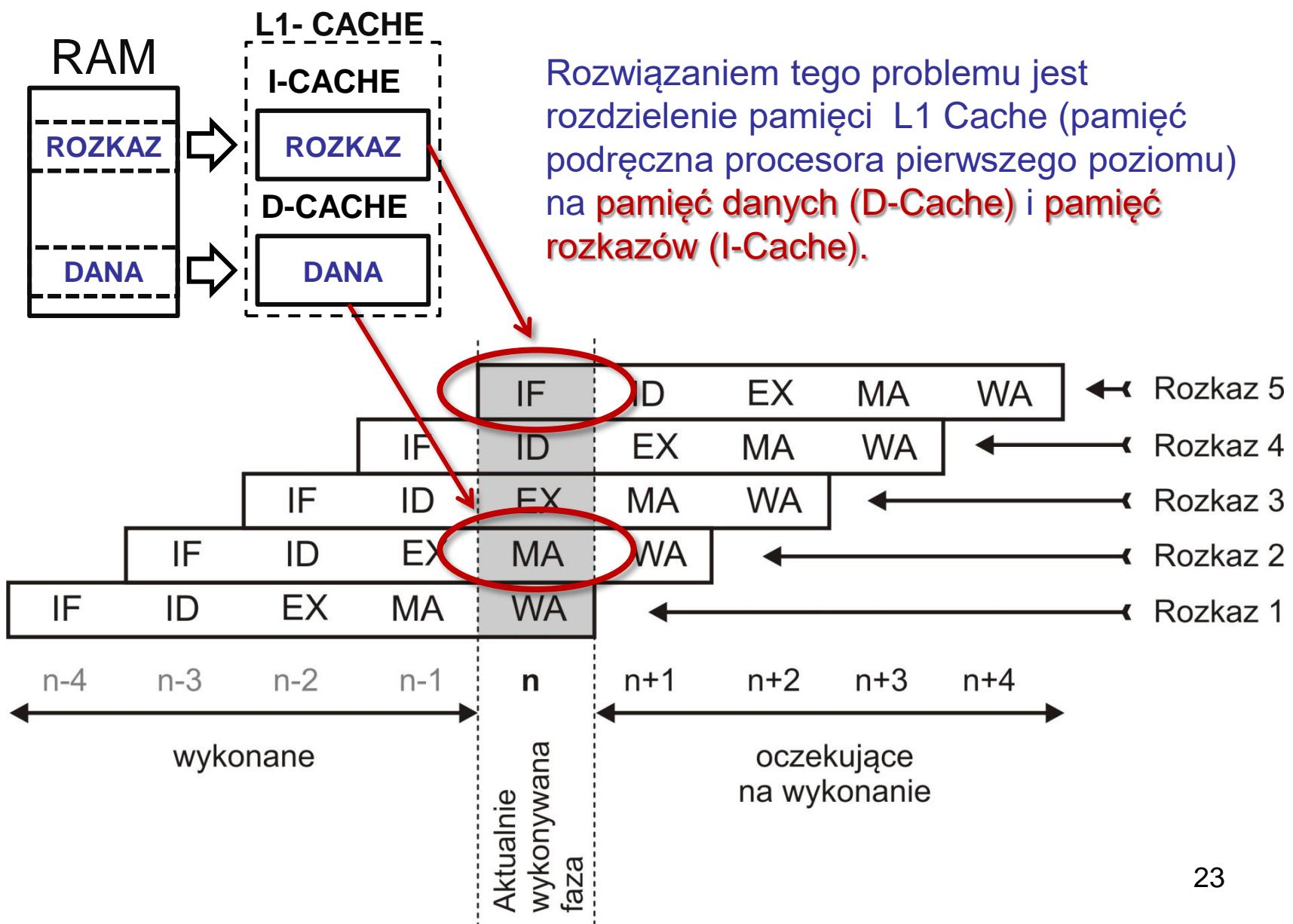
Konflikt zasobów (ang. structural hazard). Powstaje on wówczas, gdy rozkazy będące przetwarzane w różnych etapach kolejki jednocześnie sięgają do tych samych zasobów komputera (pamięci operacyjnej, czy też układów wejścia-wyjścia).

- ✓ Przykładem może być tu sytuacja, gdy rozkaz będący w trakcie przetwarzania pobiera dane z pamięci operacyjnej (stopień MA), a kolejny rozkaz ma być właśnie w tym momencie pobierany (stopień IF).

Konflikty w przetwarzaniu potokowym



Konflikty w przetwarzaniu potokowym





Konflikty w przetwarzaniu potokowym

Konflikt danych (*data hazard*) Pojawiają się one gdy dwa, przetwarzane w różnych stopniach kolejki, rozkazy korzystają z tej samej danej .

Na przykład po wykonaniu pary rozkazów:

$$a \leftarrow 1 + 2;$$

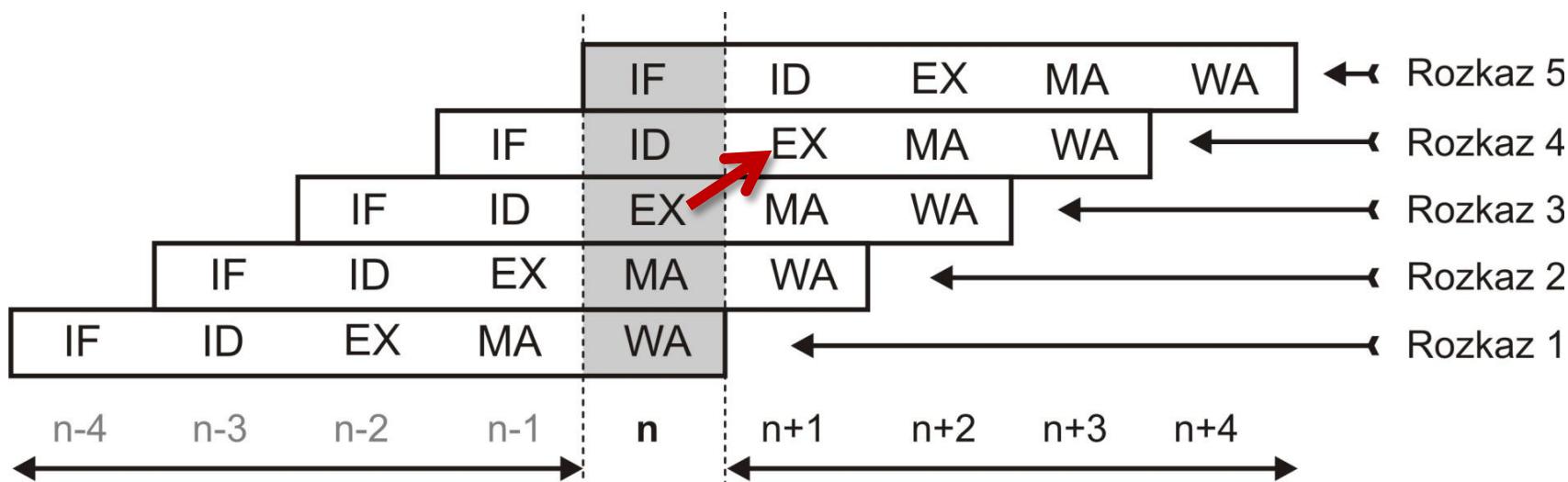
$$b \leftarrow a + 1;$$

w zmiennej (lub rejestrze) „b” powinna znajdować się wartość 4.

Jeżeli jednak rozkazy te wykonywane są równocześnie, z niewielkim tylko przesunięciem, może się zdarzyć, że rozkaz drugi ($b \leftarrow a + 1;$) pobierze wartość zmiennej (rejestru) „a” zanim rozkaz poprzedni ($a \leftarrow 1 + 2;$) zdąży zapisać w nim wynik obliczenia.

Konflikty w przetwarzaniu potokowym

Zapobieganie przestojom wynikającym z konfliktów danych jest tworzenie „skrótów” pomiędzy potokami. Skrót taki jest bezpośrednim połączeniem pomiędzy stanowiskami etapów wykonania EX. Na tym etapie wynik operacji, chociaż nie został jeszcze zapisany, jest już dostępny i może być wykorzystany jako argument następnego rozkazu. Procesory wyposażone są w układy sprzętowe sprawdzające, czy sąsiednie instrukcje współdzielą zasób.





Ograniczenia przetwarzania potokowego

- ✓ Zwiększanie liczby stopni potoku wydaje się dobrą metodą zwiększenia wydajności procesowa.
- ✓ Cyklu przetwarzania rozkazu nie można jednak w nieskończoność dzielić na coraz mniejsze fragmenty.
- ✓ **Przetwarzanie każdego stopnia musi zajmować tyle samo czasu** (taśma musi przesuwać się ze stałą prędkością), dalszy podział może więc przynieść straty zamiast zysków.

Przetwarzanie równoległe (superskalarne)

Superskalarność - możliwość jednoczesnego wykonywania kilku rozkazów maszynowych, realizowana poprzez zwiększenie jednostek wykonawczych wewnętrz procesora.



Przetwarzanie superskalarne

Następny krokiem w rozwoju procesorów było **przetwarzanie superskalarne**, dzięki któremu możliwe jest ukończenie kilku rozkazów w pojedynczym takcie zegara.

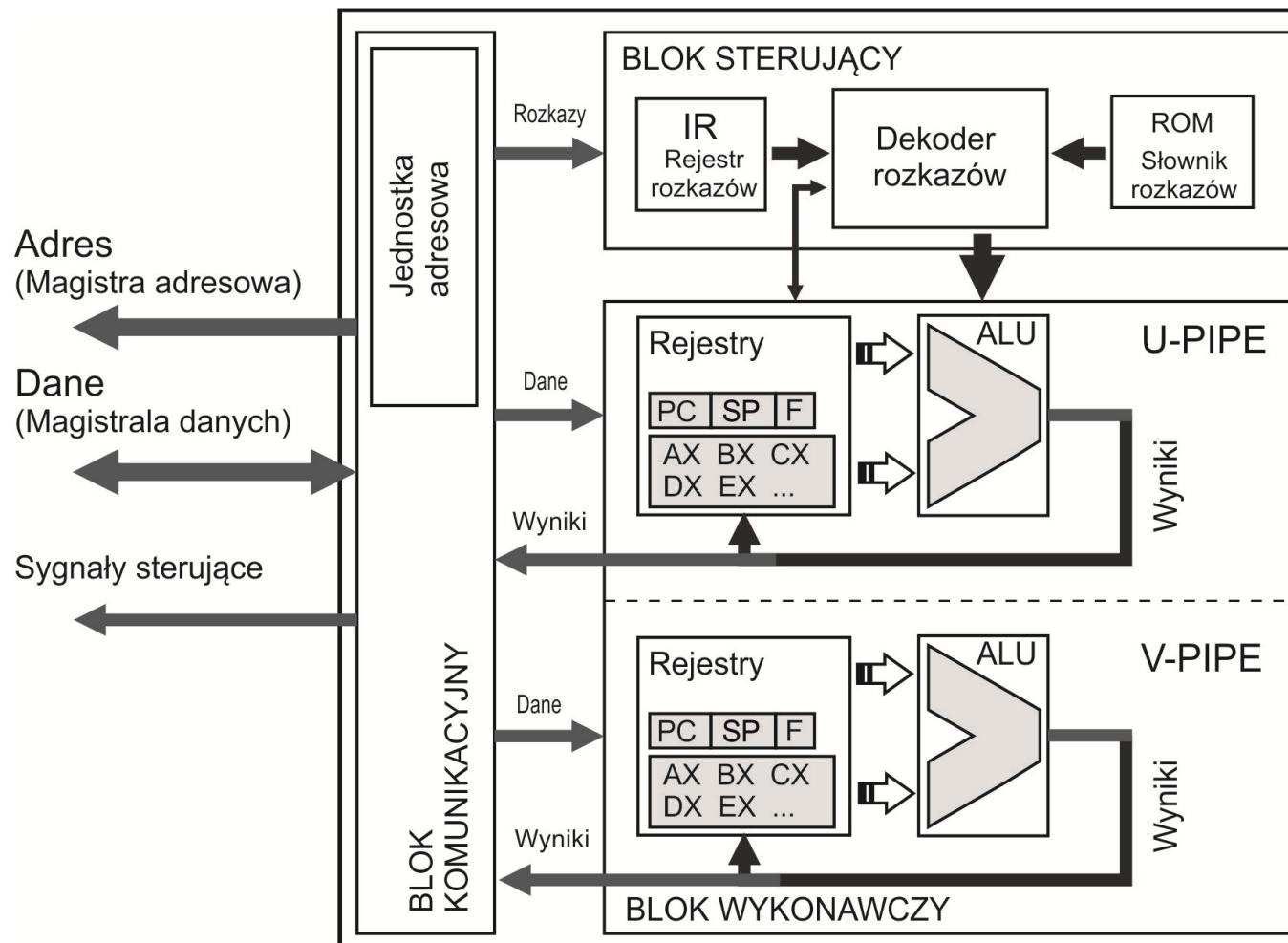
Efekt ten osiągany jest poprzez **umieszczenie w procesorze kilku jednostek wykonawczych** (jednostka arytmetyczno-logiczna, jednostka zmiennoprzecinkowa itp.).

Pierwszym procesorem dla komputerów klasy PC wykorzystującym przetwarzanie superskalarne był, wprowadzony na rynek w 1993 r. procesor Pentium. Posiadał on dwa potoki (określane jako **u-pipe** i **v-pipe**).

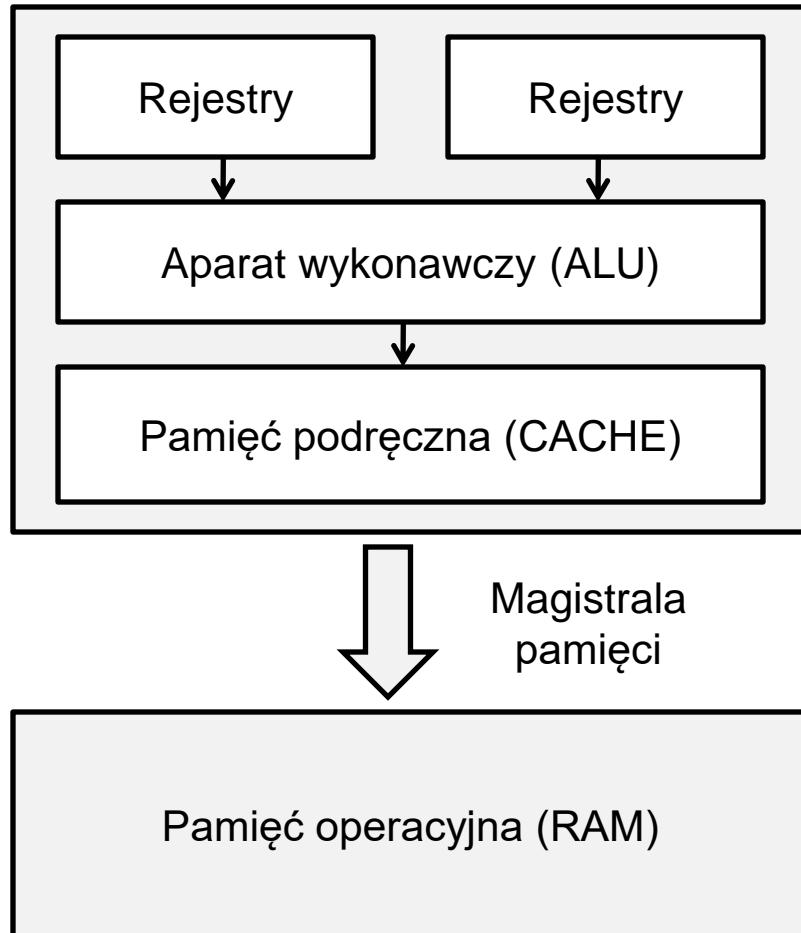
- U-pipe - jest potokiem głównym i mógł wykonywać wszystkie instrukcje.
- V-pipe - jest potokiem dodatkowym, wykonującym tylko niektóre instrukcje. W przypadku, gdy dwie kolejne instrukcje nie mogły być sparowane (wykonane równolegle) pracował tylko potok u-pipe.



Przetwarzanie superskalarne



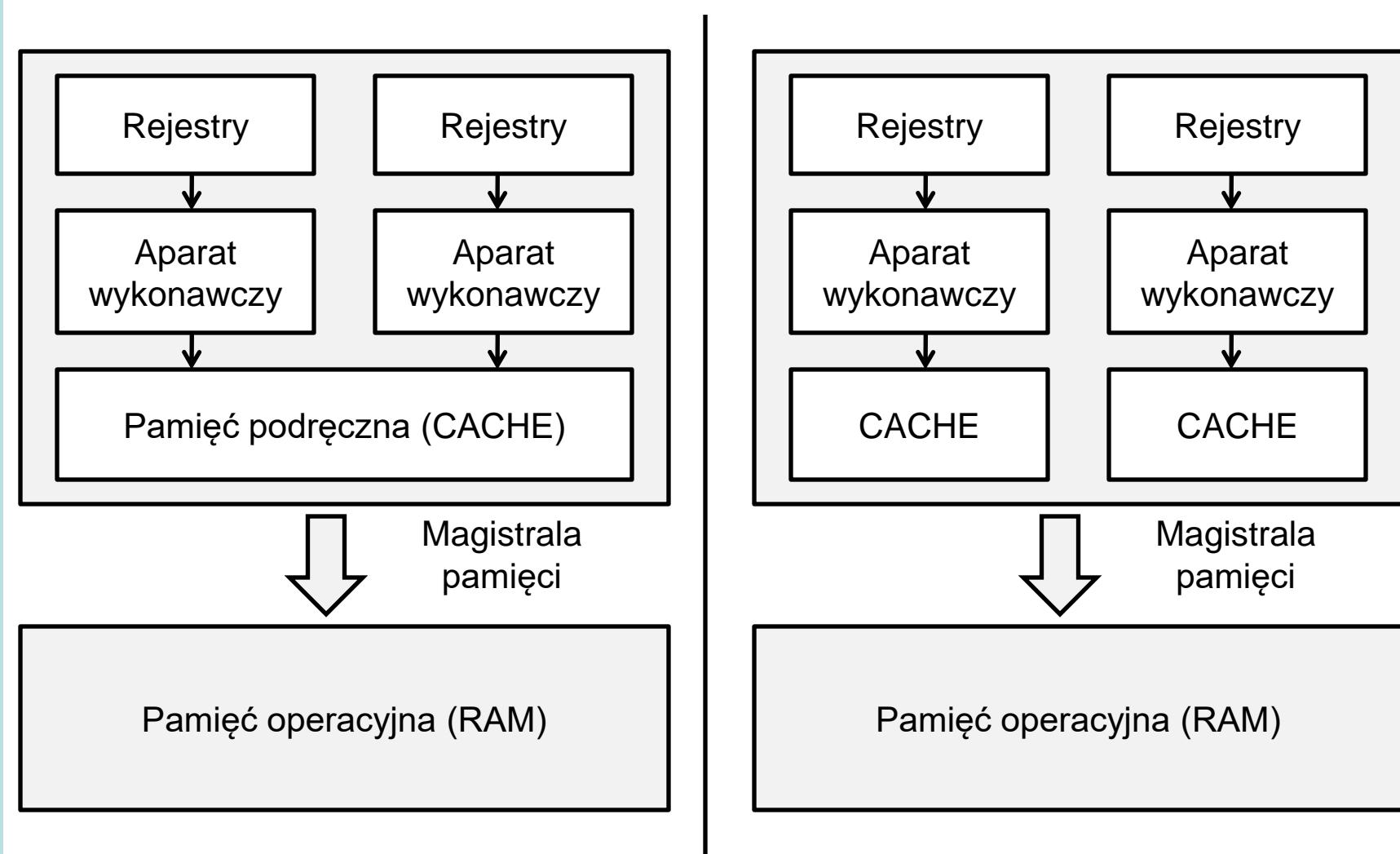
Procesory logiczne – technologia HT



Hyper-Threading (HT) – procesor posiada dwa zestawy rejestrów dzięki czemu emuluje obecność dwóch układów nazywanych **procesorami logicznymi**

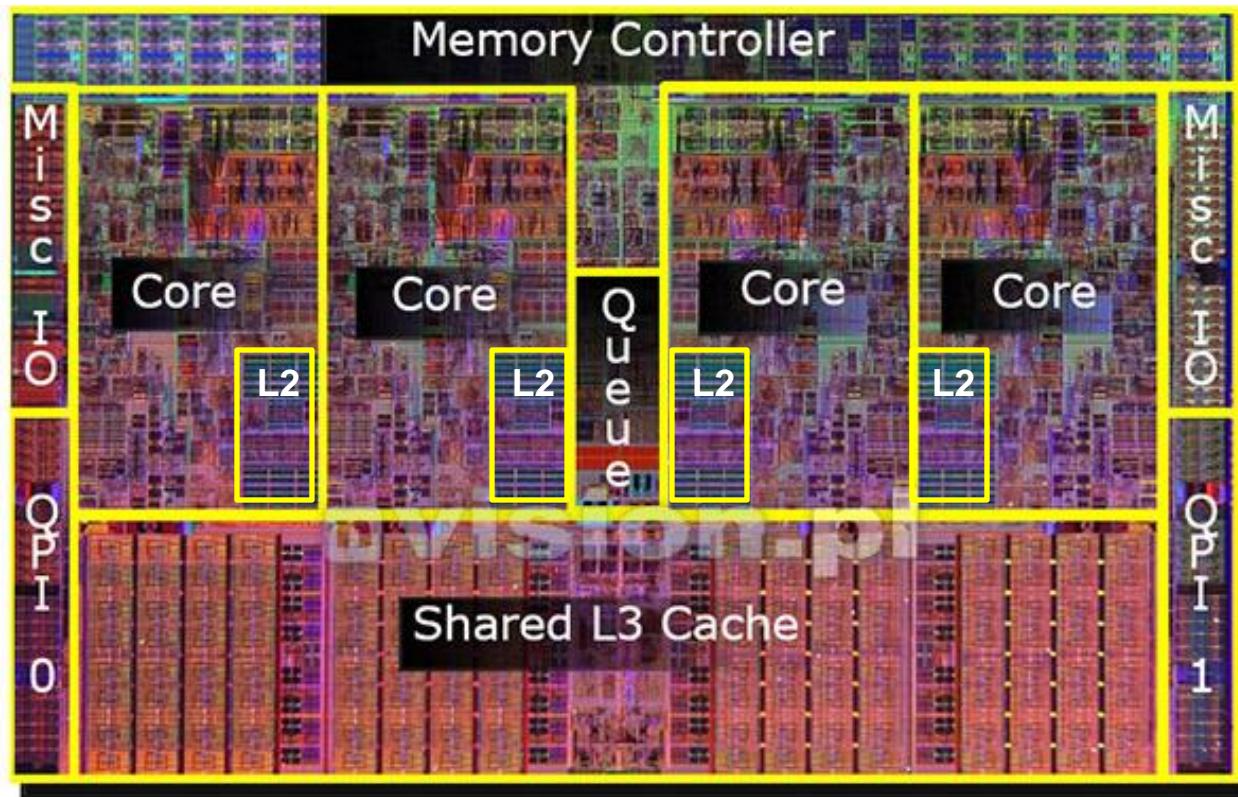
Pamiętajmy: program „widzi” procesor jako zestaw rejestrów.

Procesory fizyczne - wielordzeniowość



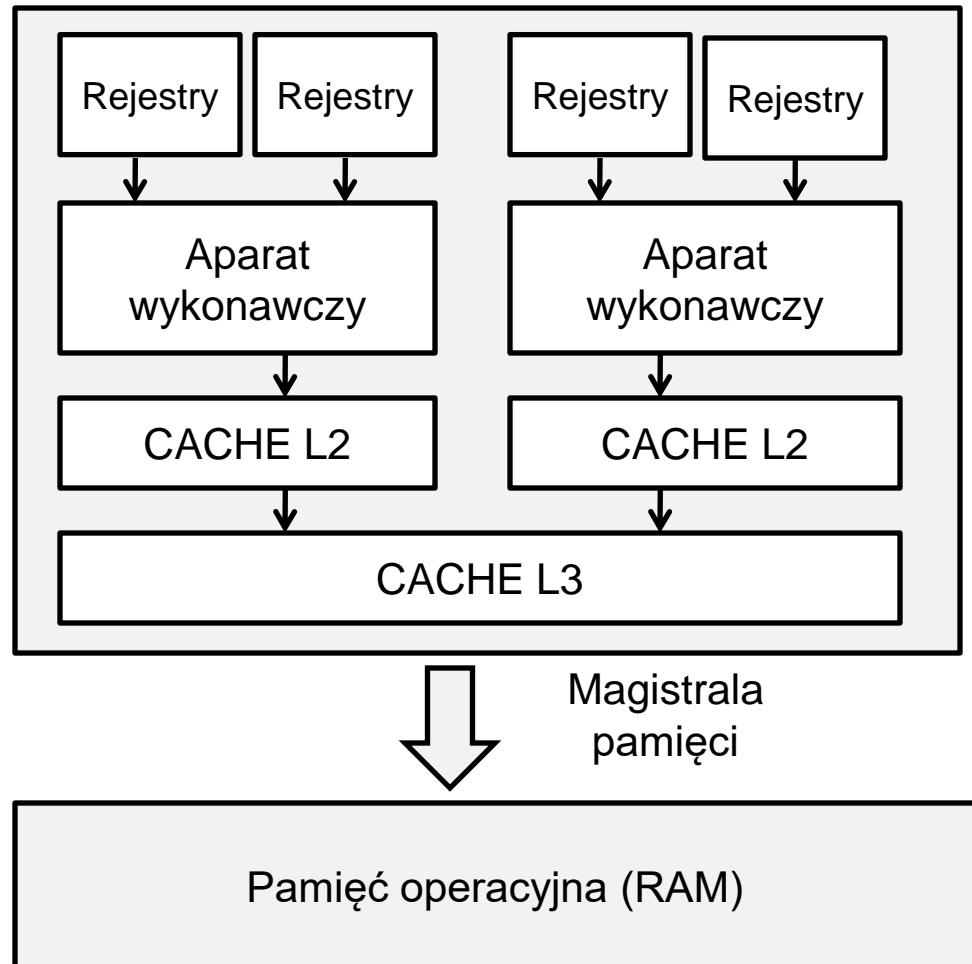
Procesory fizyczne - rdzenie

W praktyce spotykam też hybrydę dwóch powyższych schematów.
W procesorze Intel Core i7 każdy z rdzeni posiada własną pamięć podręczną poziomów L1 i L2, natomiast pamięć poziomu L3 jest wspólna.



Źródło: http://nvision.pl/img/art/procesory/intel_core_i7/intel_core_i7-2.jpg

Procesory fizyczne – rdzenie + HT



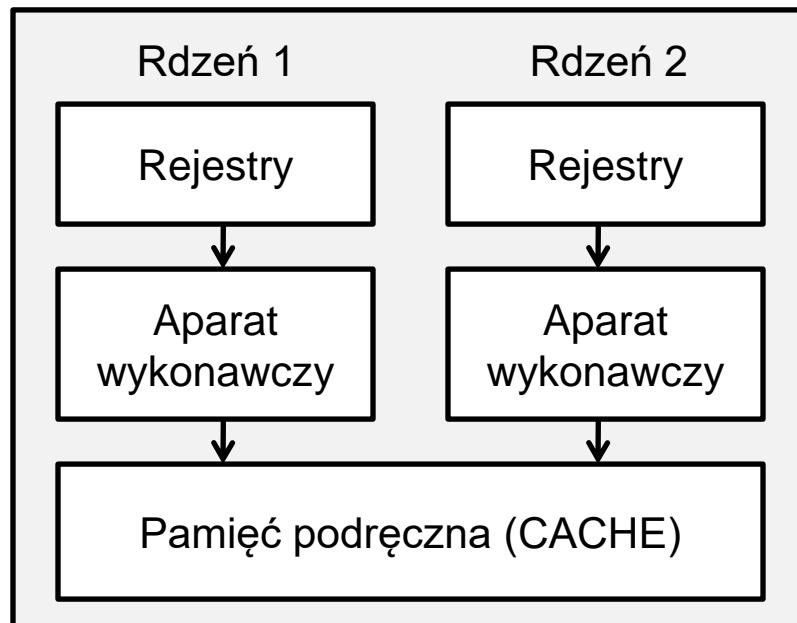
W niektórych procesorach dla każdego z fizycznych rdzeni zastosowano technologię Hyper-Threading.

Oznacza to, że każdy rdzeń widziany jest jako dwa procesory logiczne i może obsługiwać dwa wątki jednocześnie.

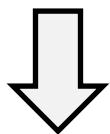
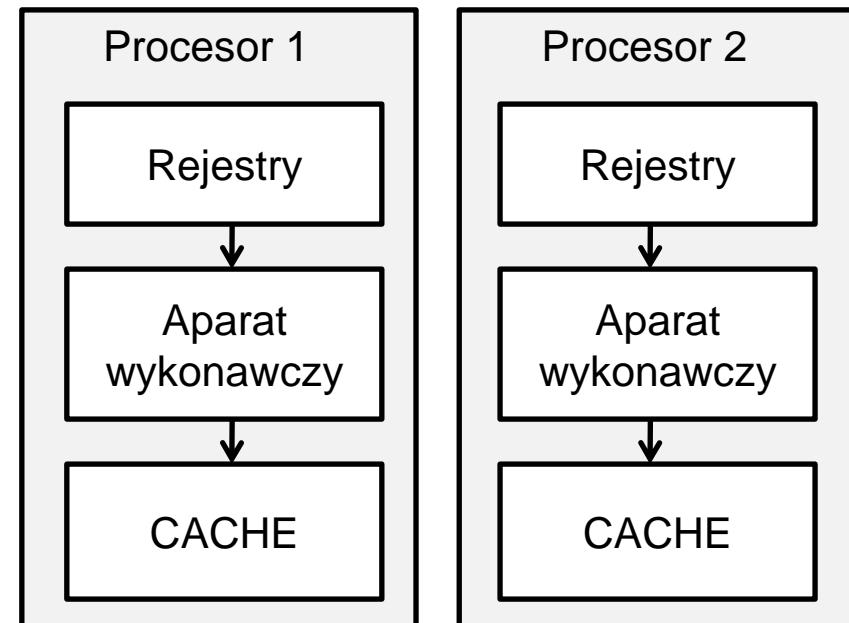
Np. czterordzeniowy procesor Core i7 może wykonywać 8 wątków jednocześnie.

Wielordzeniowość VS. wieloprocesorowość

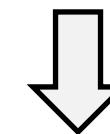
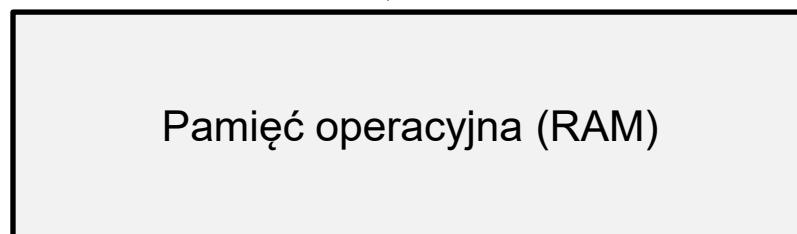
Procesor dwurdzeniowy



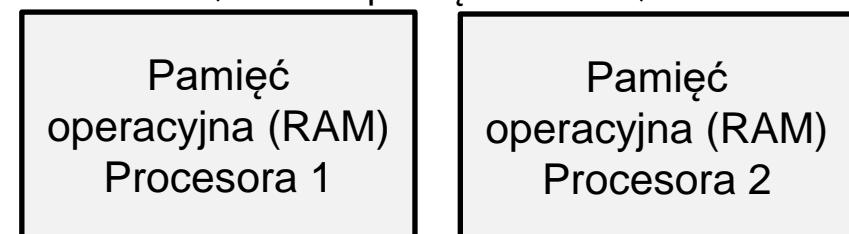
System (płyta) dwuprocesorowa



Magistrala
pamięci

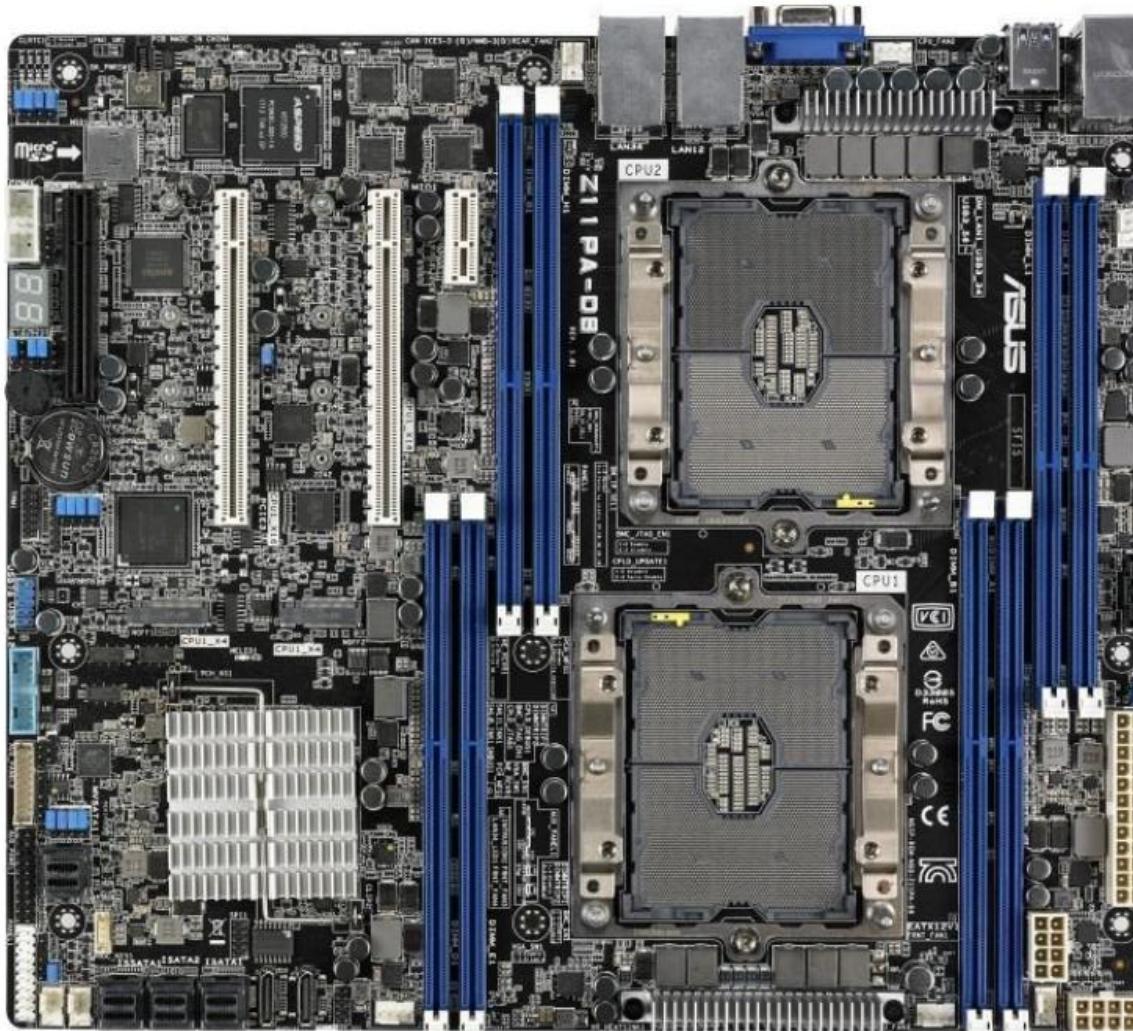


Dwie odrębne
magistrale
pamięci



Wielordzeniowość VS. wieloprocesorowość

System (płyta) dwuprocesorowa



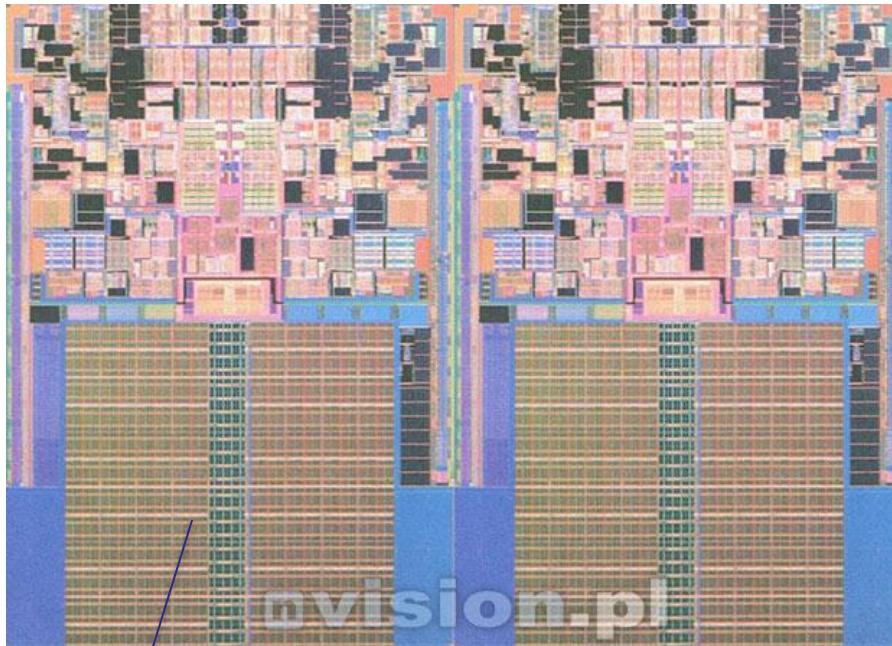
Przykład płyty głównej w standardzie CEB

Pamięć CACHE

Pamięć podręczna CACHE – mechanizm, w którym część spośród danych zgromadzonych w pamięci o dłuższym czasie dostępu lub niższej przepustowości jest dodatkowo przechowywana w szybszej pamięci.

Pamięć CACHE stosuje się w celu poprawy szybkości dostępu do tych informacji, które będą potrzebne w najbliższej przyszłości.

Pamięć CACHE



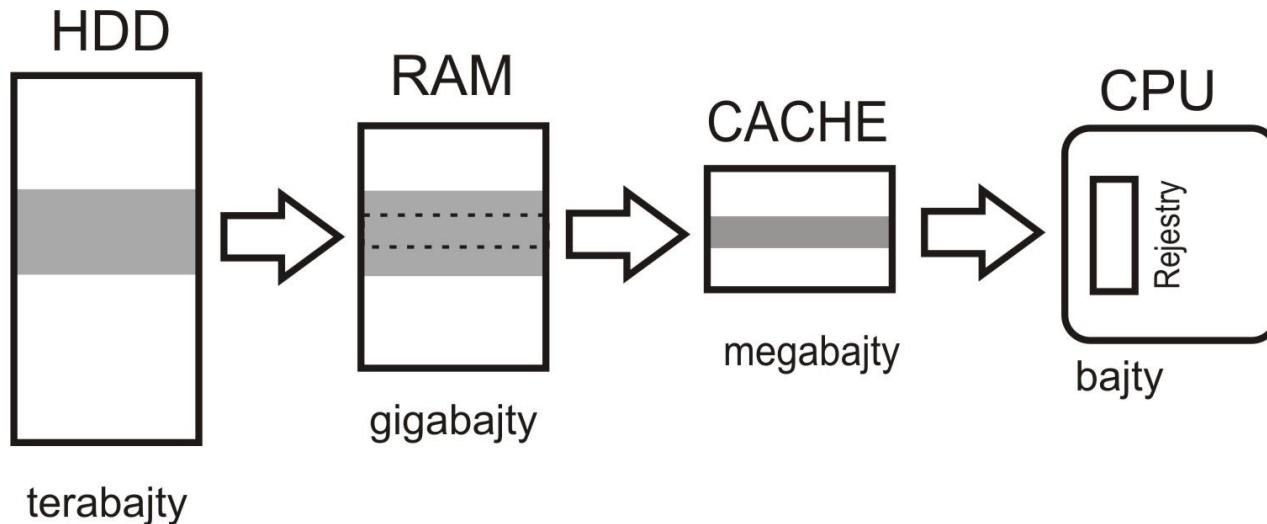
CACHE
wbudowany



CACHE
zewnętrzny



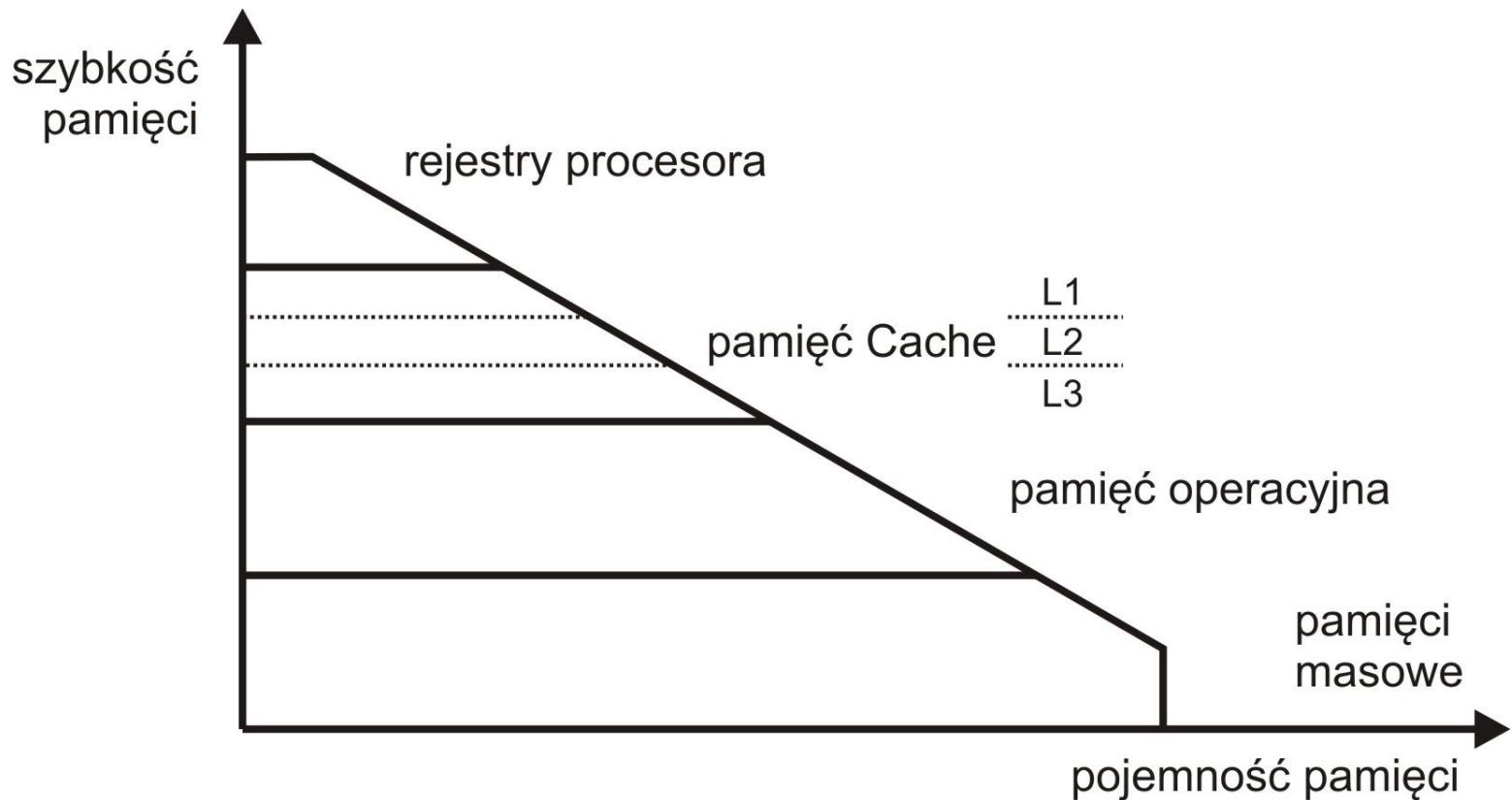
Zasoby pamięciowe komputera



- System zarządzania pamięcią zapewnia dostęp procesora do tych różnorodnych zasobów w sposób niewidoczny dla programu użytkowego.
- Zadanie to jest realizowane sprzętowo na poziomie pamięci cache i programowo - przez system operacyjny - na poziomie pamięci wirtualnej.
- W obu przypadkach obowiązuje ta sama koncepcja: niewielki fragment pamięci wyższego poziomu jest w miarę potrzeby kopowany do pamięci niższego poziomu, bliższego procesora.



Zasoby pamięciowe komputera



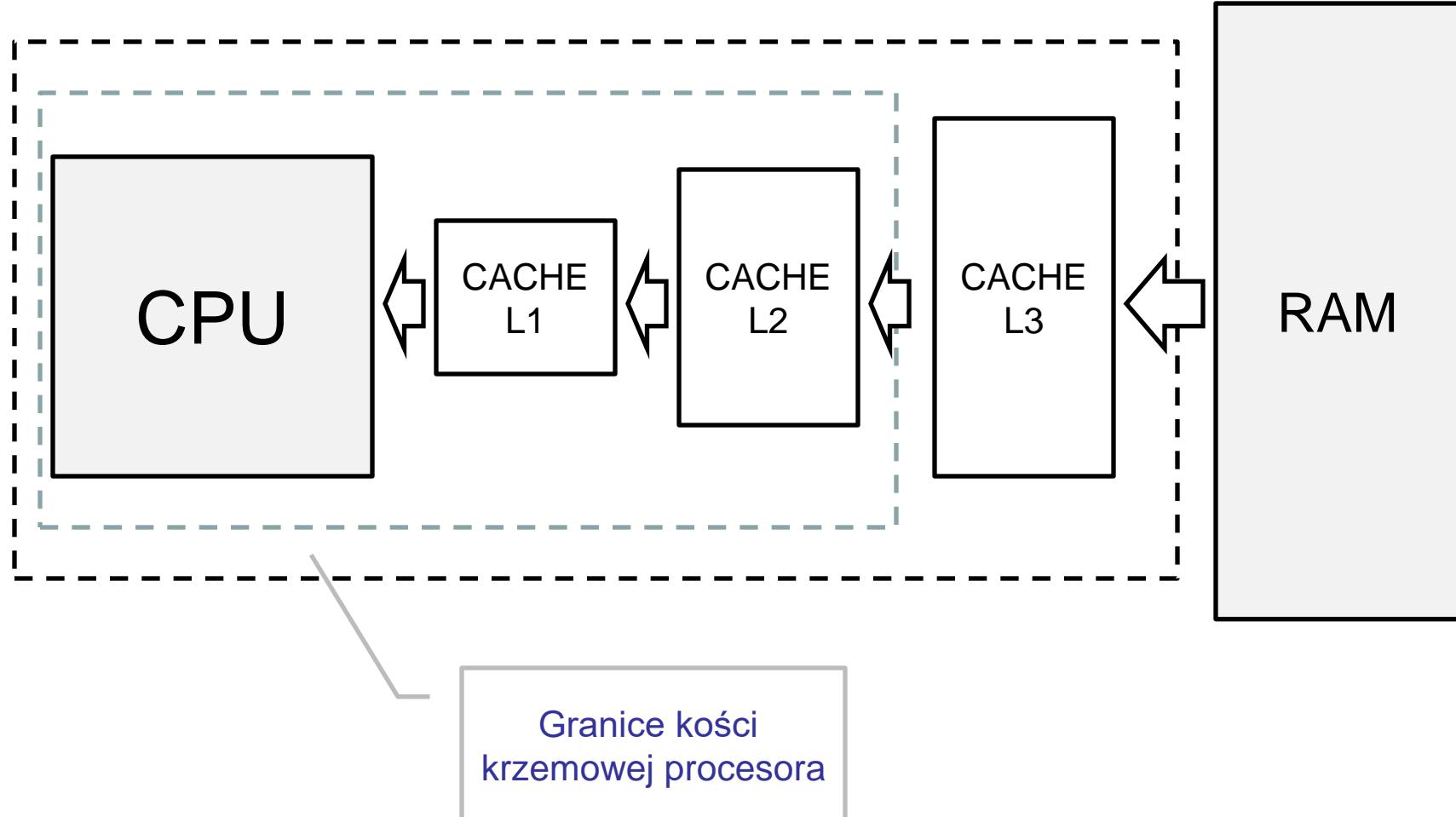


Poziomy pamięci cache

Pamięci wielodrożne są często stosowane w różnych konfiguracjach i na różnych poziomach hierarchii - obecnie w procesorach spotyka się pamięci *cache* wielopoziomowe:

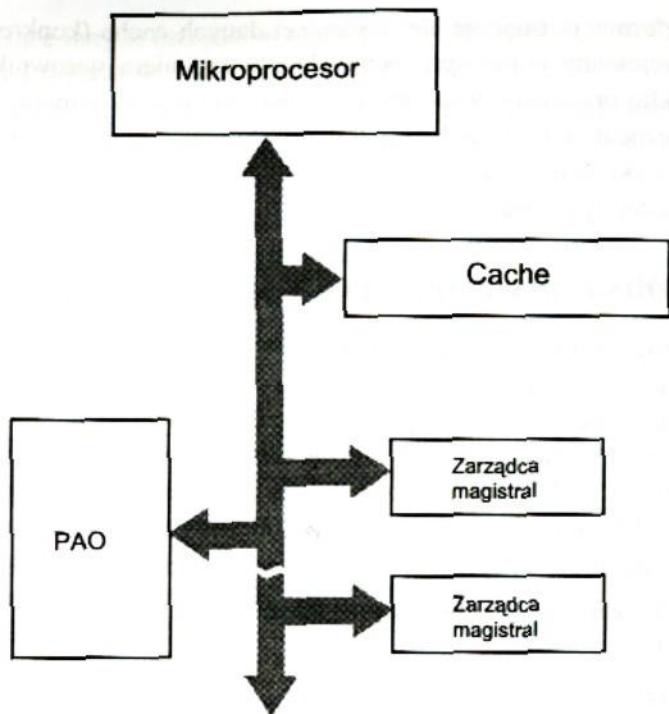
- poziom pierwszy (**L1**), zintegrowany z procesorem, stanowią osobne pamięci rozkazów (*I-cache*) i danych (*D-cache*),
- poziom drugi (**L2**) zajmuje większa i wolniejsza wspólna pamięć również umieszczona w module procesora,
- trzeci poziom *cache* (**L3**) można dołączyć w osobnym module.

Organizacja pamięci CHAHE





Architektura Look-aside („patrz w bok”)



W układzie konwencjonalnym (często określany nazwą *Look-Aside*) którym mamy do czynienia w procesorach x86 i Pentium, pamięć podręczna dołączona jest równolegle do magistrali pamięciowej.

- ✓ procesor odwołuje do pamięci cache wykorzystując magistralę pamięciową.
- ✓ częstotliwość pracy cache jest więc taka sama jak pamięci głównej, jedynie czas dostępu może ulec skróceniu.



Pamięć buforowa cache

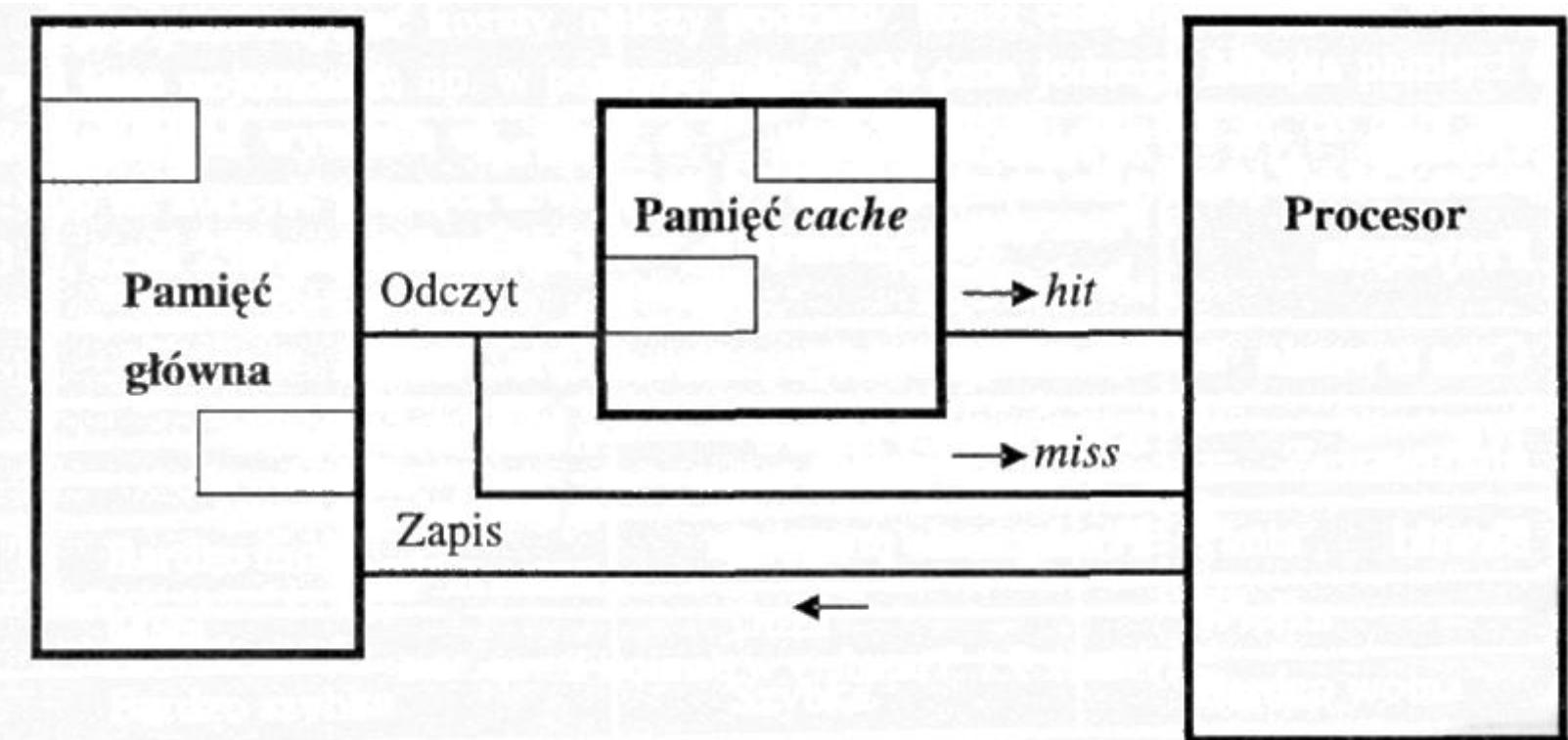
Działanie pamięci buforowej *cache*, (zwanej pamięcią podręczną):

- prawie zawsze, kiedy procesor poszukuje jakiegoś miejsca w pamięci, jego kopia jest w *cache* - przypadek taki nazywa się **trafiением (hit)**;
- jeżeli odczytywany adres nie jest odwzorowany w *cache*, czyli próba odczytu jest **chybiona (miss)**;
- wówczas następuje odczyt z pamięci głównej, ale równocześnie do *cache* jest przepisywany blok (zwany linią) kilku lub kilkunastu sąsiednich bajtów wraz z ich adresem. W ten sposób chybienia powodują uzupełnianie pamięci buforowej aż do momentu, kiedy nie będzie w niej miejsca i wtedy usuwa się jedną linię - np. tę, która była najdawniej używana.



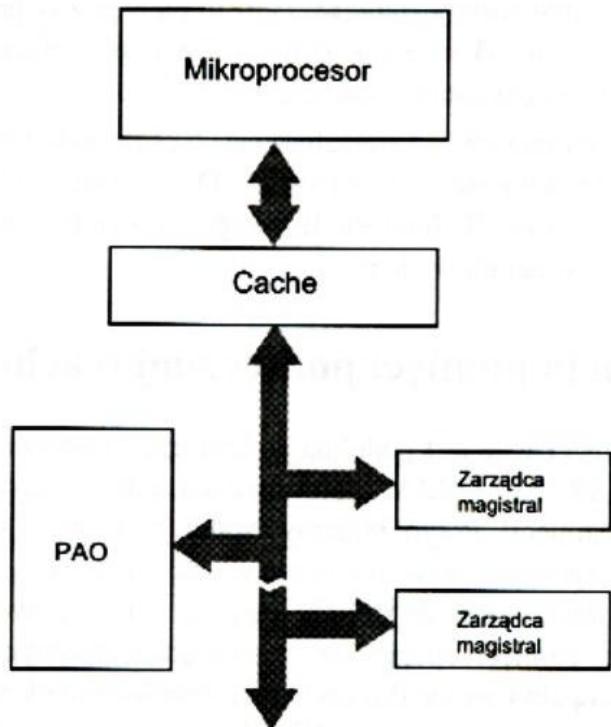
Pamięć buforowa cache

Źródło: Komorowski W.: Krótki kurs architektury i organizacji komputerów, MIKOM



- Efektywność pamięci buforowej określa tzw. współczynnik trafień, czyli procent trafionych prób odczytu pamięci.
- W rzeczywistych systemach współczynnik trafień wynosi zależnie od programu 97% - 99%.

Architektura Look-throught („patrz w bezpośrednio”)



Wojtuszkiewicz K.: Urządzenia techniki komputerowej,
część I: Jak działa komputer, MIKOM

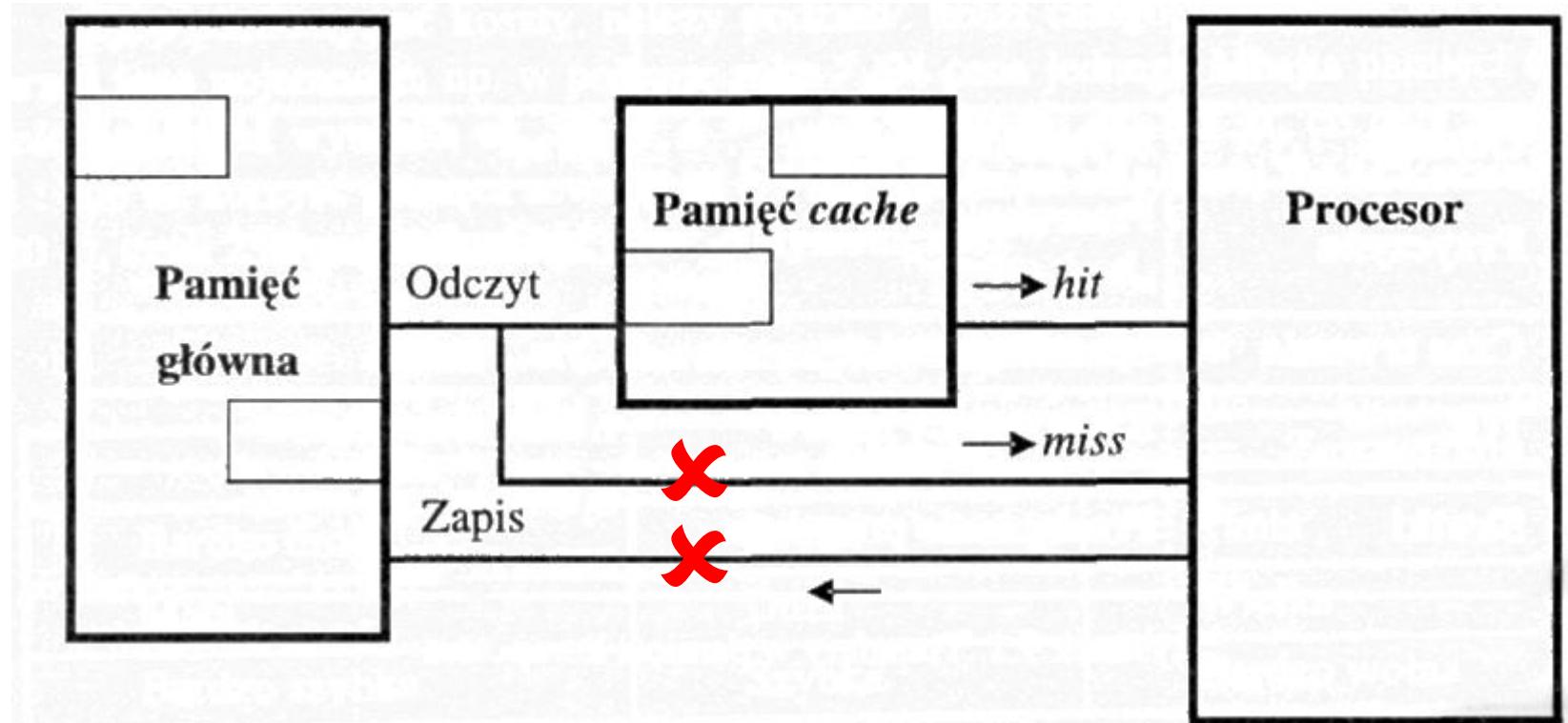
Drugi sposób podłączenia pamięci CACHE określany jest mianem *Look-Through* lub *Inline Cache*.

- ✓ Procesor, zanim sięgnie do pamięci głównej, napotyka układ pamięci podręcznej.
- ✓ Ta z kolei, sprzężona jest z pamięcią główną poprzez właściwą magistralę pamięciową.
- ✓ Cache może więc być taktowany z prędkością większą niż sama magistrala pamięciowa, na przykład dwa razy szybciej - takie rozwiązanie zastosowano w procesorze Pentium II.



Pamięć buforowa cache

Źródło: Komorowski W.: Krótki kurs architektury i organizacji komputerów, MIKOM

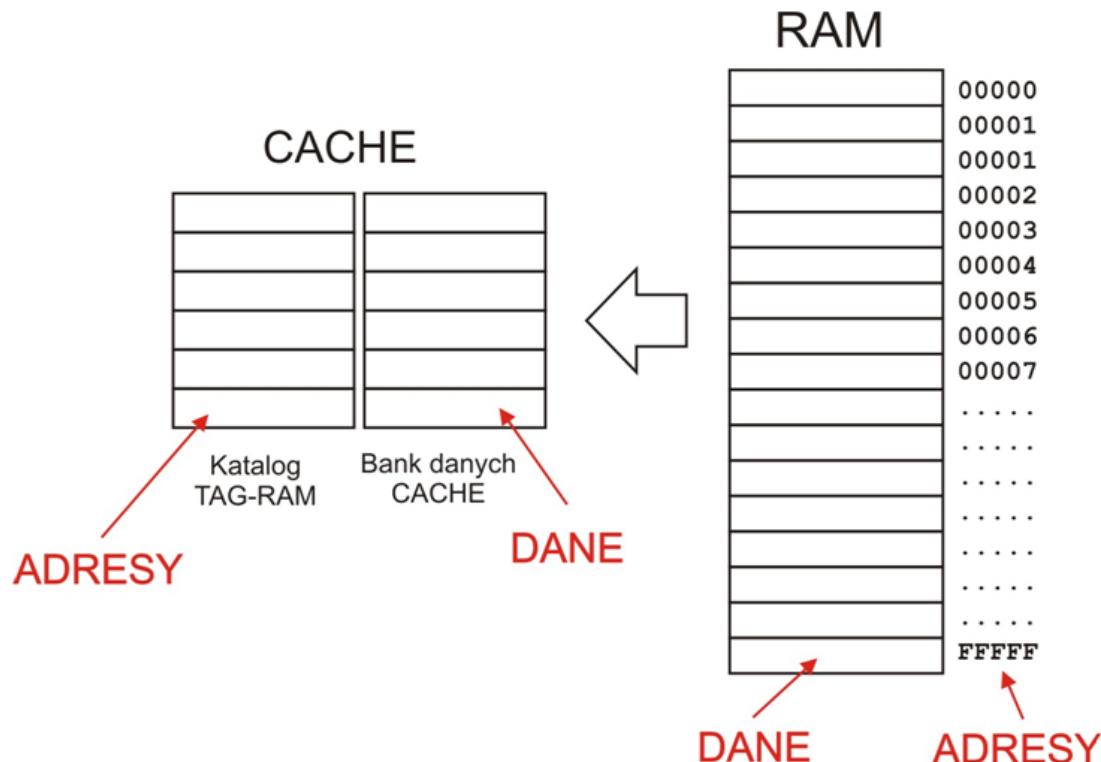


- ✓ W e współczesnej architekturze procesorów nie jest już możliwe odwołanie się procesora do pamięci RAM bez pośrednictwa pamięci CAHE

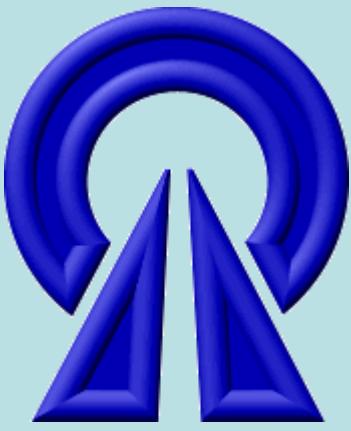


Pamięć buforowa cache

Ponieważ w pamięci buforowej znajdują się fragmenty pochodzące z różnych, niespójnych, obszarów przestrzeni adresowej, muszą w niej być skopowane **nie tylko dane, ale i ich adresy**.



W katalogu adresów mogą być zapisane również inne informacje dotyczące np. aktywności poszczególnych linii, które są używane przez algorytmy usuwania niepotrzebnych bloków.



Dziękuję za uwagę

*dr Artur Bartoszewski
UTH Radom*





Literatura

1. Metzger Piotr - *Anatomia PC*, wydanie XI, Helion 2007
2. Wojtuszkiewicz Krzysztof - *Urządzenia techniki komputerowej, część I: Jak działa komputer*, MIKOM, Warszawa 2000
3. Wojtuszkiewicz Krzysztof - *Urządzenia techniki komputerowej, część II: Urządzenia peryferyjne i interfejsy*, MIKOM, Warszawa 2000
4. Komorowski Witold - *Krótki kurs architektury i organizacji komputerów*, MIKOM Warszawa 2004
5. Gook Michael - *Interfejsy sprzętowe komputerów PC*, Helion, 2005