Zasada lokalności – analiza wykonania programu (IA-32/Linux)

```
        movl  tablica[, %ebx, 4], % eax     ; szukanie najmniejszego w tablicy
label:  decl  %ebx                          ; liczb naturalnych
        cmpl tablica[, %ebx, 4], % eax
        jbe  hop                            ; omiń gdy (%eax) ≤ (tablica(,%ebx,4))
        movl  tablica[, %ebx, 4], % eax
        mov  %ebx, adres
hop:    cmp  0, %ebx
        jnz  label
```

- rozkazy pętli są wielokrotnie powtarzane
- instrukcje programu tworzą sekwencje (kolejne komórki)

- tablica jest z natury (sposób deklaracji) spójnym blokiem pamięci
- odwołania do tablicy są powtarzane i systematyczne

WNIOSEK:

W *komputerze przechowującym program w pamięci (PSC)* programy i dane
mają tendencję do skupiania w wymiarze przestrzennym i czasowym.

Zasada lokalności - przejawy

lokalność przestrzenna (ang. *spatial locality*)

Jest prawdopodobne użycie informacji z sąsiednich lokacji pamięci.

- kody rozkazów – zależność *lokacyjna sekwencyjna* (licznik rozkazów)
- struktury danych – skupione (zmienne robocze) lub regularne (tablice)

lokalność tymczasowa (ang. *temporal locality*)

Kod użyty będzie zapewne ponownie użyty w nieodległym czasie.

- kody rozkazów – pętle programowe
- struktury danych – zmienne robocze: ciągłe używanie
  – struktury regularne: wielokrotne użycie elementów

WNIOSEK:

Ma sens utworzenie *w pobliżu procesora* **bufora** zawierającego **kopie informacji** (danych) z pamięci operacyjnej aktualnie używanych lub których użycie jest bardzo prawdopodobne i korzystanie z tych kopii.

Zasady użycia buforów

→ użycie bufora musi być *przeźroczyste dla programu*
– niewidoczne na poziomie ISA (listy rozkazów),
– realizowane na poziomie HSA (organizacji procesora/komputera)

!! *sposób wykonania* rozkazu **nie** może być **zależny** od obecności **bufora**, ale
czas wykonania rozkazu w obecności bufora powinien być krótszy

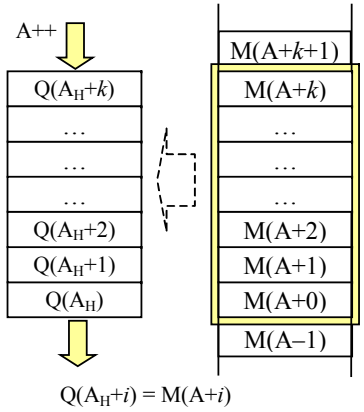
!! użycie bufora pamięci **nie otwiera** nowej przestrzeni adresowej:
→ *nie ma* rozkazów dostępu do danych w buforze (sprzeczne z koncepcją PSC)
→ *potrzebne* polecenia sterowania zawartością i kontroli stanu bufora

bufor pamięci
- powinien być wykonany w szybkiej technologii (statycznej)
- powinien być umieszczony w bezpośrednim sąsiedztwie procesora

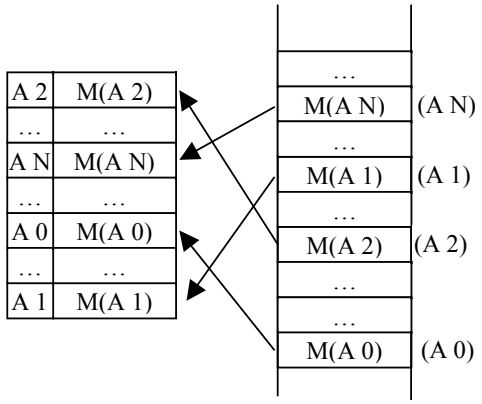
Pamięć główna wspomagana buforem pamięci umożliwia:
- szybki dostęp do danych (pamięć statyczna)
- tanią realizację rozległej przestrzeni adresowej (pamięć dynamiczna)

Bufory pamięci

bufor zawiera **kopie** aktualnie przetwarzanych danych



bufor sekwencyjny – kolejka



bufor blokowo-skojarzeniowy

organizacja sekwencyjna – kolejka

- bufor rozkazów (ang. *instruction queue*), bufor zapisów (ang. *write buffer*)

organizacja blokowo-skojarzeniowa

- pamięć podręczna (ang. *cache memory*) – zawiera **kopie** używanych danych

Skuteczność buforów cache

Zasada lokalności *nie gwarantuje obecności kopii* danych w buforze *cache*.

Skuteczność użycia pamięci podręcznej zależy od:

- organizacji (struktury) bufora
- strategii wypełniania i aktualizacji bufora

Ilościowa ocena skuteczności:

- *współczynnik trafień* (ang. *hit rate*) h
- *współczynnik chybień* (ang. *miss rate*) $m=1-h$
- t_{mp} – średnia strata czasu w razie chybienia (ang. *miss penalty*)

Średni czasu dostępu do pamięci w obecności jednopoziomowego bufora cache

$$t_a = (1-m)\,t_{ca} + m\,(t_{ram} + t_{mp})$$

Średni czasu dostępu do pamięci w obecności wielopoziomowego bufora cache zależy od organizacji bufora.

Wyszukiwanie danych w pamięci podręcznej

Aby wykorzystać efekt lokalności przestrzennej, dane powinny być kopiowane do bufora wraz z danymi z sąsiednich lokacji – niezależne tworzenie kopii pojedynczych bajtów/słów jest ignorowaniem lokalności.

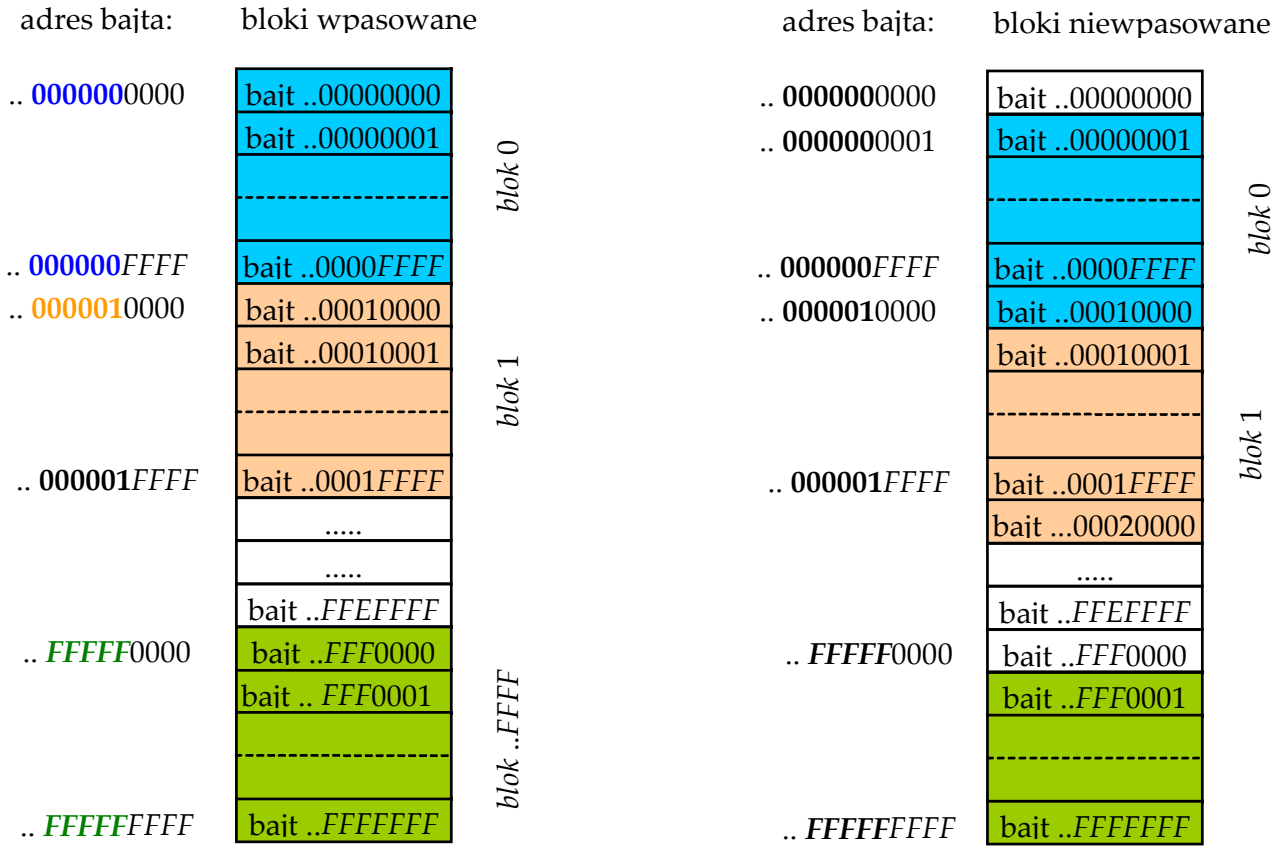
Linia (blok danych) – uporządkowany zbiór słów z kolejnych lokacji w pamięci

SPOSTRZEŻENIE:
Bloki danych (**linie**) w buforze tworzą **zbiór nieuporządkowany** (lok. czasowa),
dane bloku tworzą **zbiór uporządkowany** (lokalność przestrzenna: sąsiedztwo)

Jedynym ***identyfikatorem*** słowa (bajtu) jest jego ***adres w pamięci operacyjnej***

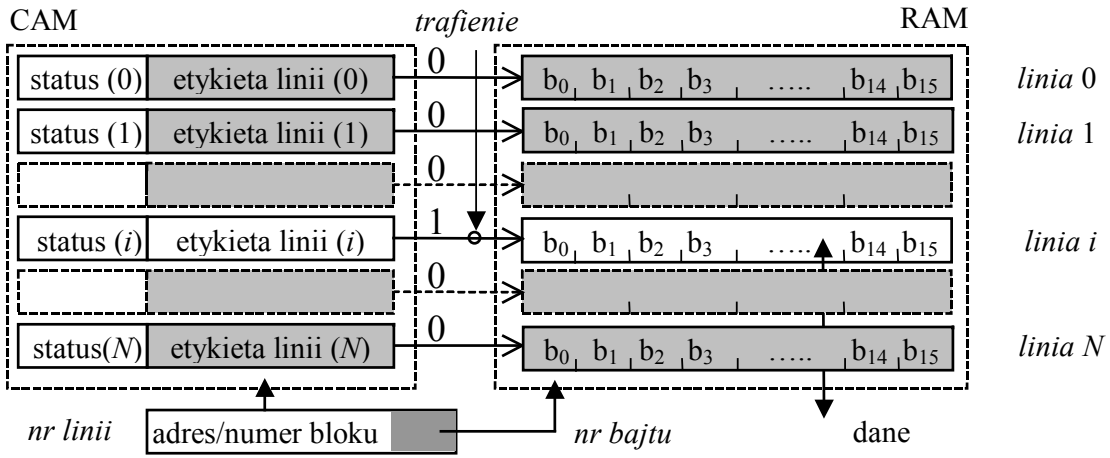
Z uwagi na szybkość wyszukiwania danych w buforze
→ **identyfikator linii** (bloku danych) powinien być **krótki**
→ **struktura linii** powinna być **jednorodna**:
ustalony rozmiar linii ułatwia identyfikację → upraszcza i przyśpiesza dostęp

Wpasowanie bloków w przestrzeni adresowej pamięci operacyjnej



Organizacja pamięci podręcznej

→ każdy blok zawiera 2^k bajtów/słów,
wpasowanych na granicy paragrafu o rozmiarze 2^k bajtów/słów, dzięki czemu:
→ wskaźnik/numer bloku (etykieta adresu, ang. *address tag*) jest taki jak wyższe bity adresu każdego bajtu/słowa w bloku wpasowanym
→ kopia bloku danych jest zawartością pojedynczej linii bufora cache



Schemat organizacji bufora pamięci podręcznej

Organizacja odwzorowania danych w pamięci podręcznej

→ linia – jednostka wymiany danych między buforem a pamięcią główną
→ rozmiar lokacji $s=2^k$ bajtów/słów (linia wymiany)
→ identyfikator lokacji w buforze – skrócony $(n–k)$-bitowy adres a (z pominięciem k niższych bitów)
→ przestrzeń adresowa procesora = suma rozłącznych linii

→ czas wyszukiwania w buforze (zbiór nieuporządkowany) – $O(\log_2 N)$
(N – liczba linii) → optymalny rozmiar bufora $N=2^n$ linii

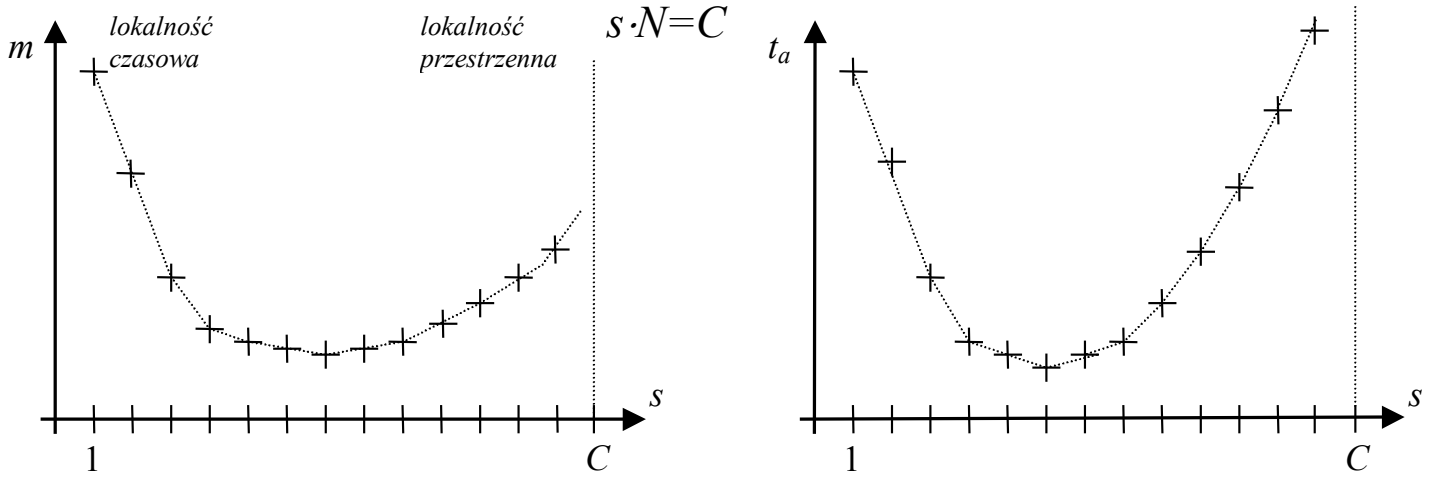
$M(a)$ – zawartość linii o adresie (skróconym) a w pamięci głównej
$C(i)$ – zawartość linii o numerze i w buforze *cache*

Odwzorowanie różnowartościowe

$$\{M(a)\} \rightarrow^{w} \{C(i)\},$$
$$C(i) = M(a) \Rightarrow \forall j \neq i\colon C(j) \neq M(a)$$

Charakterystyki skuteczności

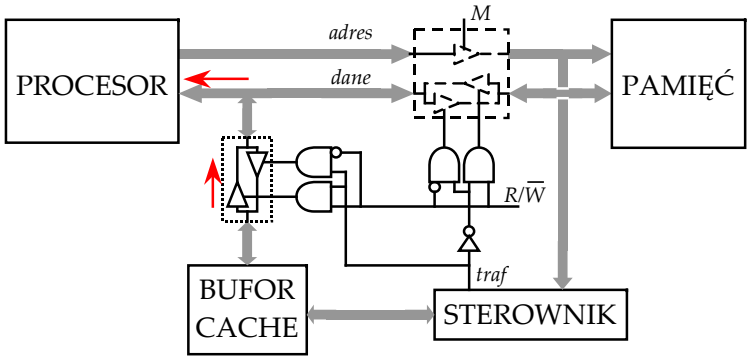
- większa liczba linii (N) – lepsza lokalność czasowa
- większy rozmiar linii (s) – lepsza lokalność przestrzenna
- zwiększanie rozmiaru linii dla ustalonej pojemności bufora *cache* ($C=s \cdot N$):
  – dominujące zanikanie lokalności czasowej
  – wzrost strat czasu w razie chybienia (czas wymiany ~ liczby transferów)



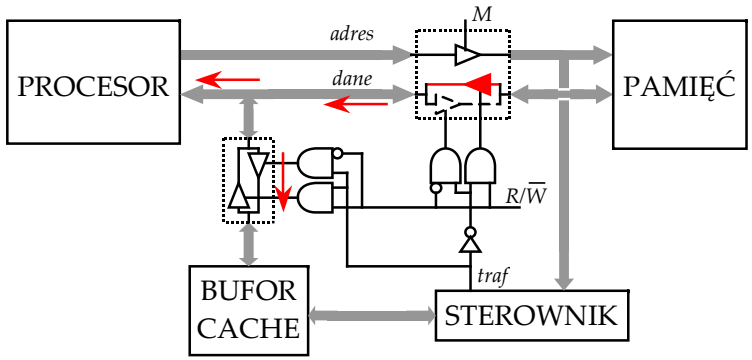
Zależność współczynnika chybień m i średniego czasu dostępu do pamięci t_a od rozmiaru linii s dla ustalonej pojemności bufora *cache* ($C=s \cdot N$)

Współdziałanie pamięci podręcznej z procesorem – odczyt

trafienie:



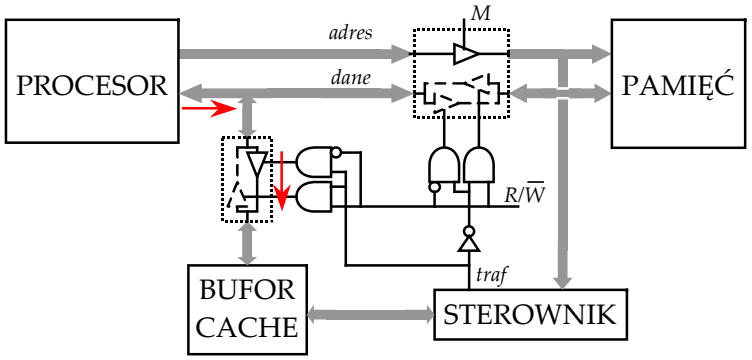
chybienie:



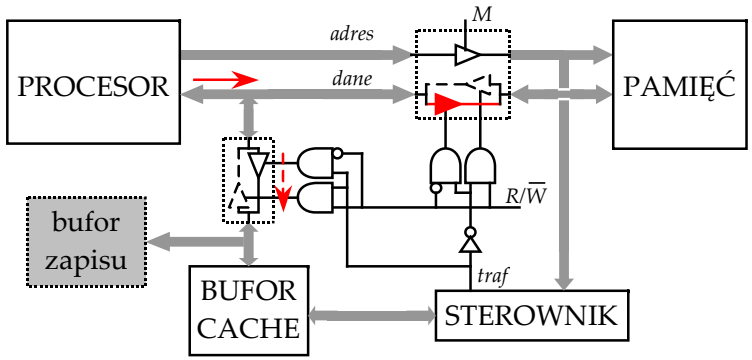
chybienie: etap ① – odczyt z pamięci (blokowy) (→ usunięcie linii → wypełnienie),
etap ② – kopiowanie odczytanego bloku

Współdziałanie pamięci podręcznej z procesorem – zapis

trafienie:

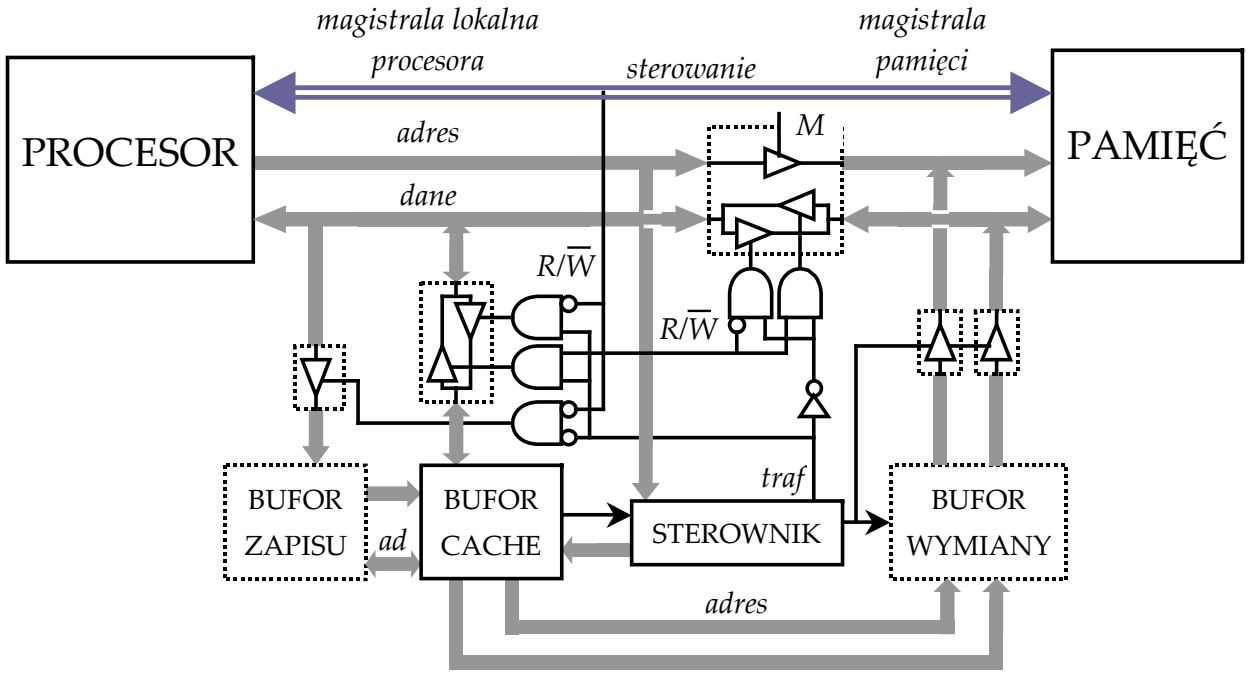


chybienie:



chybienie: etap ① – (*no allocate on write*) zapis do pamięci/ bufora zapisu,
etap ② – ... /aktualizacja odczytu zawartością bufora zapisu

Współpraca bufora *cache* z procesorem i pamięcią



Współpraca pamięci podręcznej i pamięci głównej (*traf* – sygnał trafienia w buforze *cache*, *R*/*W* – odczyt/zapis (1/0), *M* – transfer/blokada adresu (1/0))

Odwzorowanie linii w buforze pamięci podręcznej (1)

! Niektóre linie pamięci głównej ***nie mogą być kopiowane*** do bufora *cache*

- *odwzorowanie całkowicie skojarzeniowe* (ang. *fully associative*)
  - *linia* pamięci głównej może być skopiowana *w dowolnej lokacji* bufora pamięci podręcznej

  (+) wymiana linii konieczna wtedy, gdy wszystkie linie są użyte
  (+) największy współczynnik trafień, brak migotania (ang. *thrashing*)
  (–) najdłuższy czas kojarzenia (sprawdzenie każdej linii w buforze)

- *odwzorowanie bezpośrednie* (ang. *direct mapped*)
  - *rozłącznym podzbiorom linii* pamięci głównej przypisane są *unikatowe lokacje* w buforze pamięci podręcznej

  (+) najkrótszy czas kojarzenia (sprawdzenie tylko jednej linii w buforze wskazanej przez rekord indeksujący adresu (ang. *cache index*)
  (–) najmniejszy współczynnik trafień
  (–) chybienia wskutek konfliktu odwzorowania (ang. *conflict miss*) – migotanie

Odwzorowanie linii w buforze pamięci podręcznej (2)

Kompromis:

- *odwzorowanie grupowo-skojarzeniowe* / *wielodrożne* (ang. *set-associative/multi-way*)
  - – *rozłącznym podzbiorom linii* pamięci głównej przypisano *rozłączne podzbiory lokacji linii* w buforze pamięci podręcznej (bezpośrednie odwzorowanie bloków, pełne skojarzenie w podzbiorze)

  (+) czas dostępu dłuższy niż dla pamięci z odwzorowaniem bezpośrednim
  (+) niewielkie migotanie, duży współczynnik trafień
  (+) konflikt odwzorowania maleje ze wzrostem liczby lokacji w bloku
  (–) skomplikowana obsługa – algorytmy wymiany i wypełniania

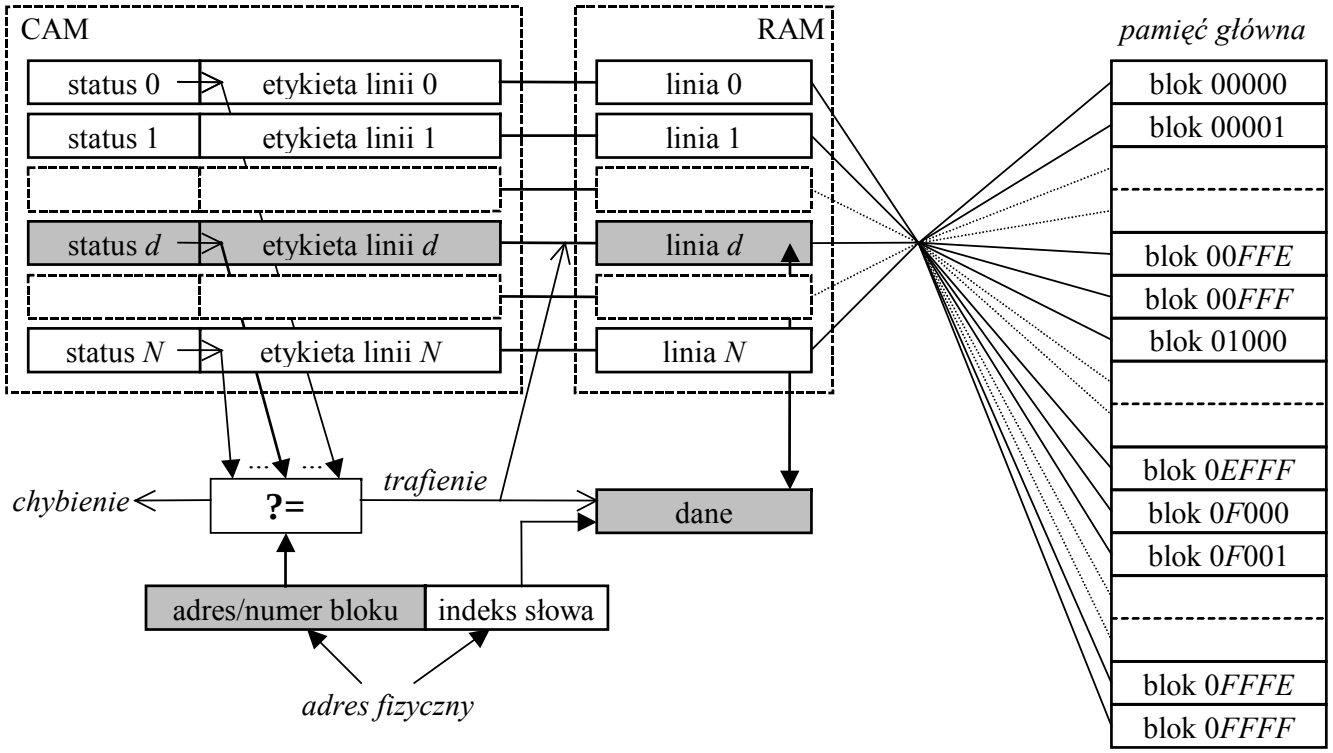
odwzorowanie:

z przeplotem – podzbiór tworzą linie, których adresy (etykiety) różnią się o 2^K (K – liczba podzbiorów linii w buforze)

bez przeplotu – podzbiór tworzy 2^K linii pamięci głównej o kolejnych adresach

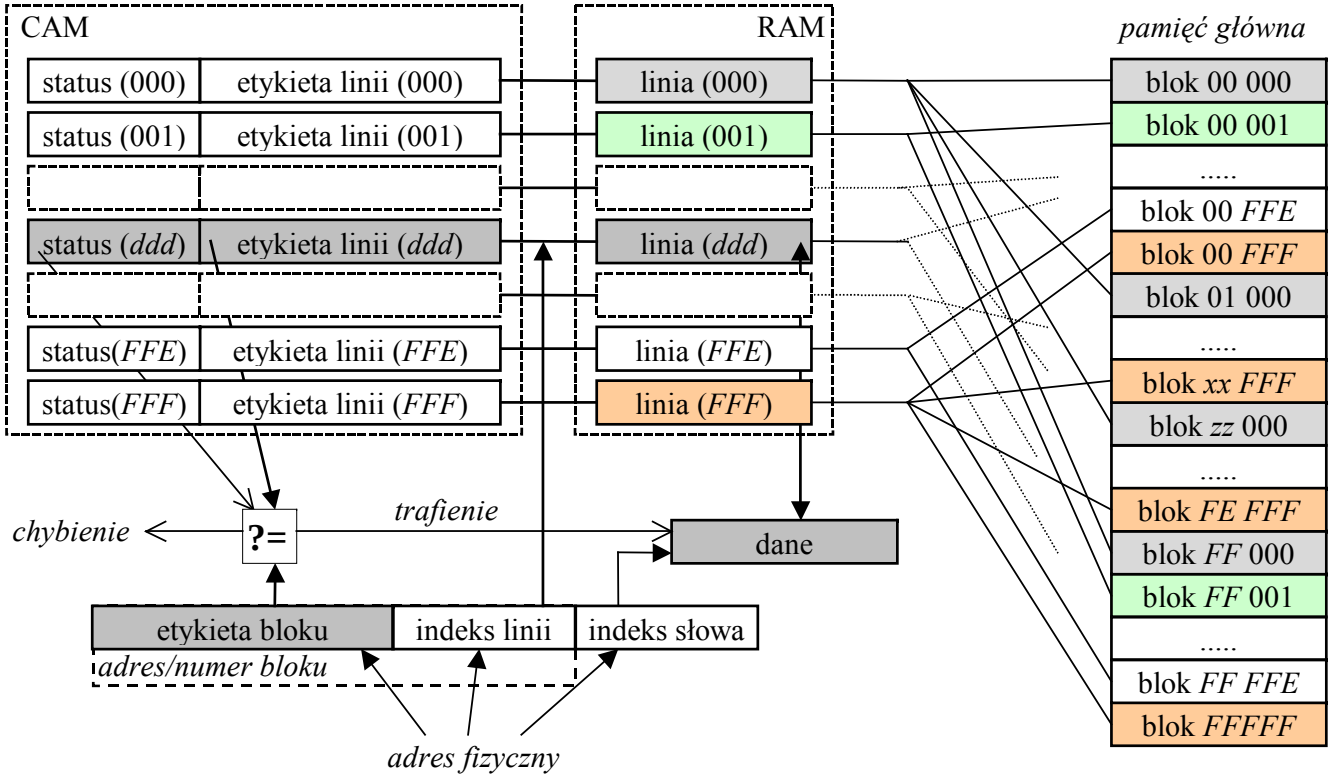
drożność bufora – liczba możliwych jednoczesnych odwzorowań w buforze linii z jednego podzbioru (ang. *number of ways*), nie musi być równa 2^t

Odwzorowanie całkowicie skojarzeniowe



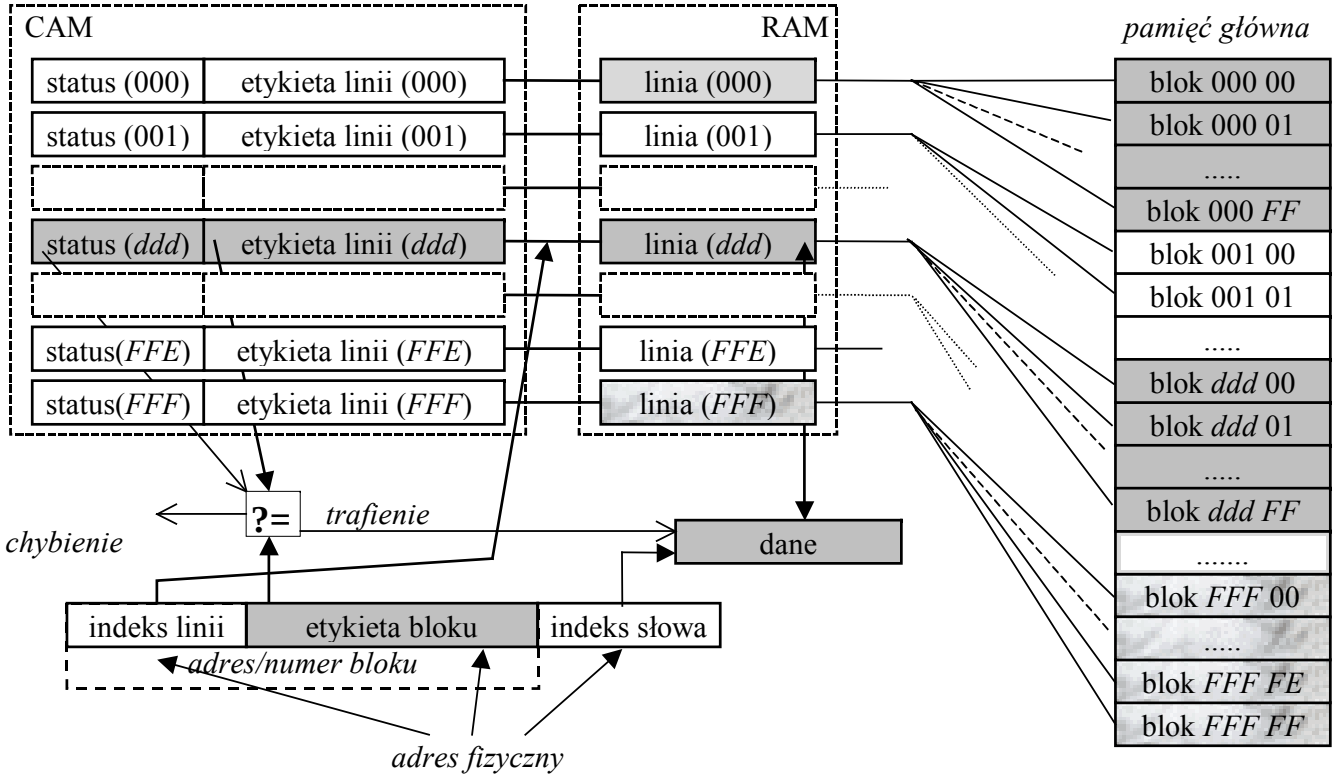
Bufor całkowicie asocjacyjny (ang. *fully associative*)

Odwzorowanie bezpośrednie – z przeplotem



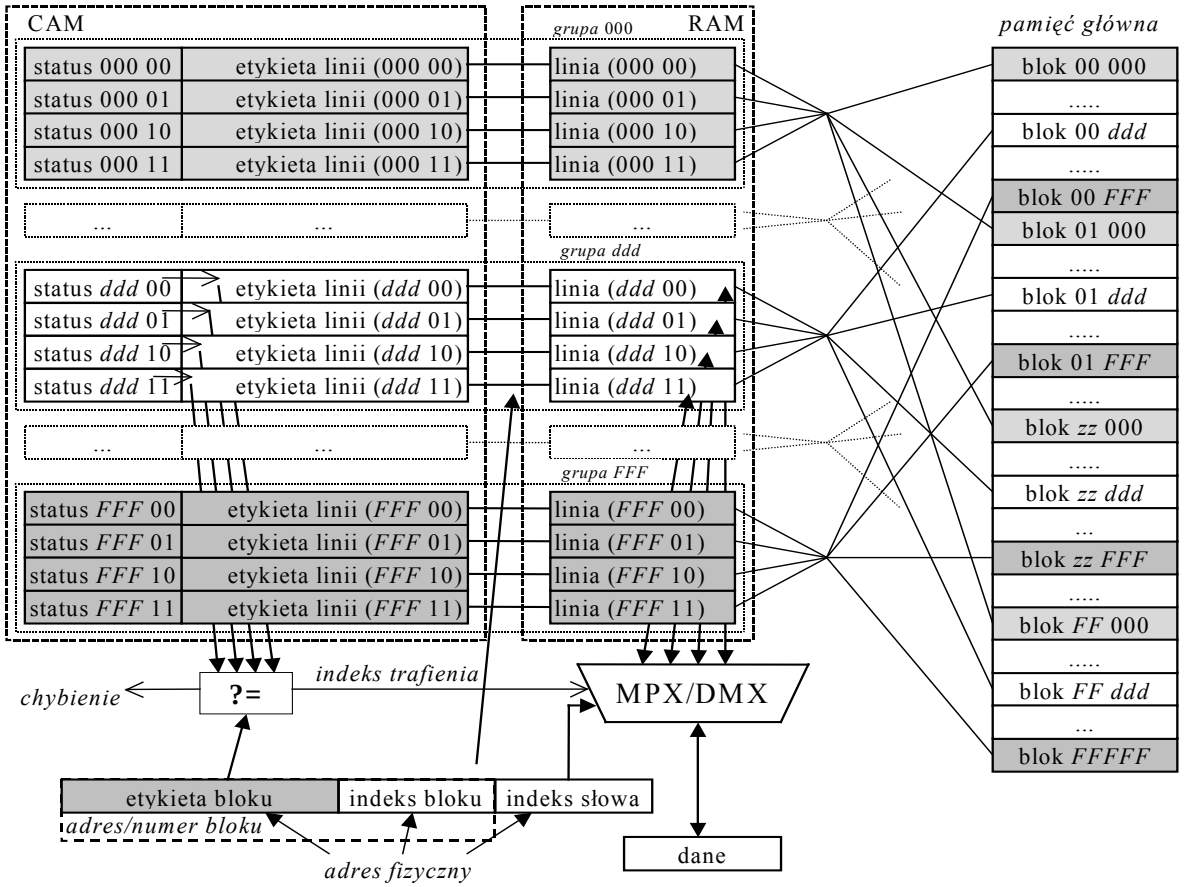
Odwzorowanie bezpośrednie (ang. *direct mapped*) z przeplotem (ang. *interlace*) bloków

Odwzorowanie bezpośrednie – bez przeplotu



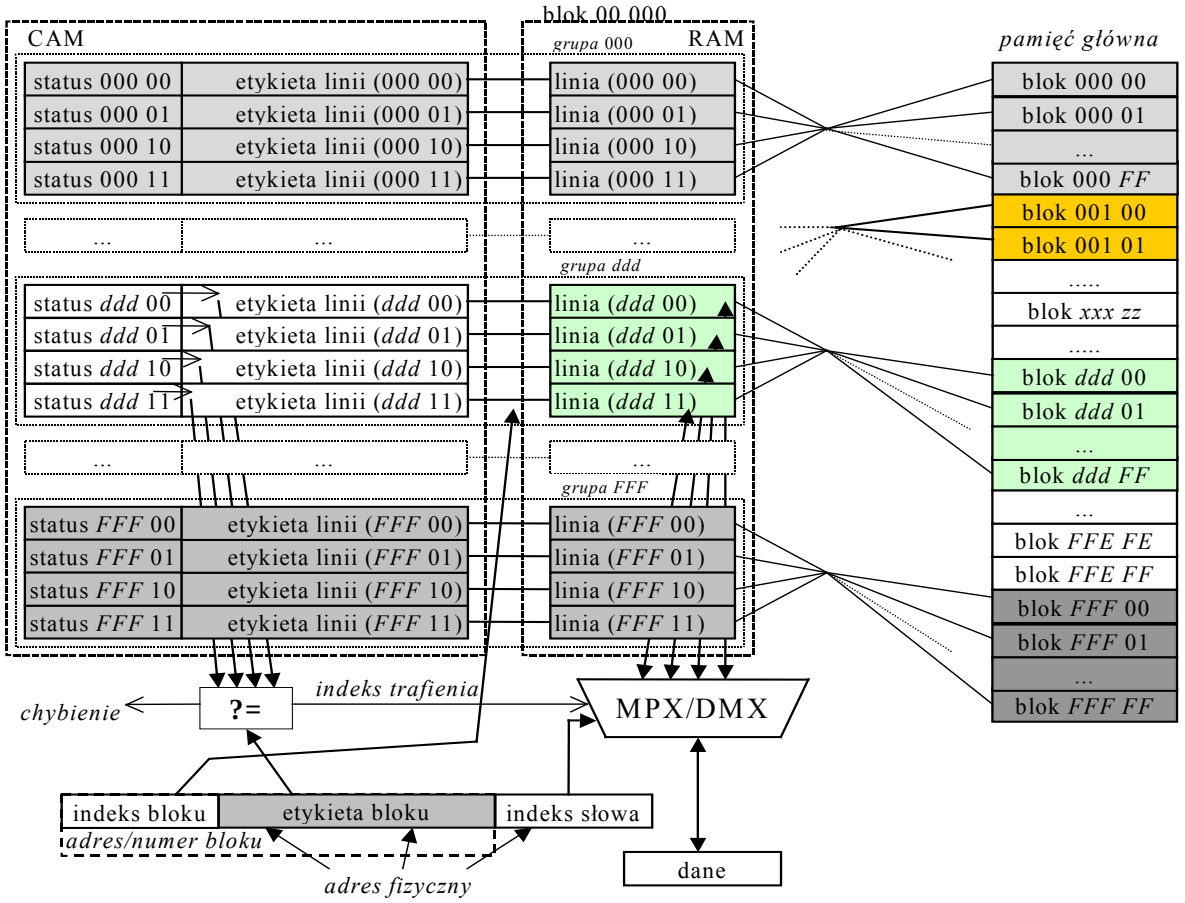
Odwzorowanie bezpośrednie bez przeplotu – bardzo częsty konflikt odwzorowania

Odwzorowanie grupowo-skojarzeniowe (wielodrożne) – z przeplotem



Bufor grupowo-skojarzeniowy 4-drożny (4-*way set-associative*)
(uwaga: liczba linii w grupie nie musi być potęgą dwójki)

Odwzorowanie grupowo-skojarzeniowe (wielodrożne) – bez przeplotu



Odwzorowanie blokowo-skojarzeniowe bez przeplotu – duże ryzyko konfliktu

Kategorie chybień

Przyczyny braku trafienia w buforze *cache* można zakwalifikować do 3 kategorii (ang. *three C's model: compulsory-capacity-conflict*):

– nieuniknione (ang. *compulsory*):
pierwsza próba dostępu do bloku musi skutkować chybieniem, bo blok nie może być odwzorowany w buforze – nie było wcześniej zapotrzebowania;

– (ograniczona) pojemność (ang. *capacity*):
jeśli bufor jest zbyt mały, aby pomieścić wszystkie bloki potrzebne podczas wykonania programu, niektóre bloki muszą być usuwane z bufora i później ewentualnie ponownie kopiowane;

– konflikt (ang. *conflict*):
jeśli odwzorowanie bloków nie jest w pełni skojarzeniowe (ang. *fully associative*), odwzorowanie bloku w grupie może wymagać usunięcia innego bloku, mimo wolnych miejsc w innych grupach;

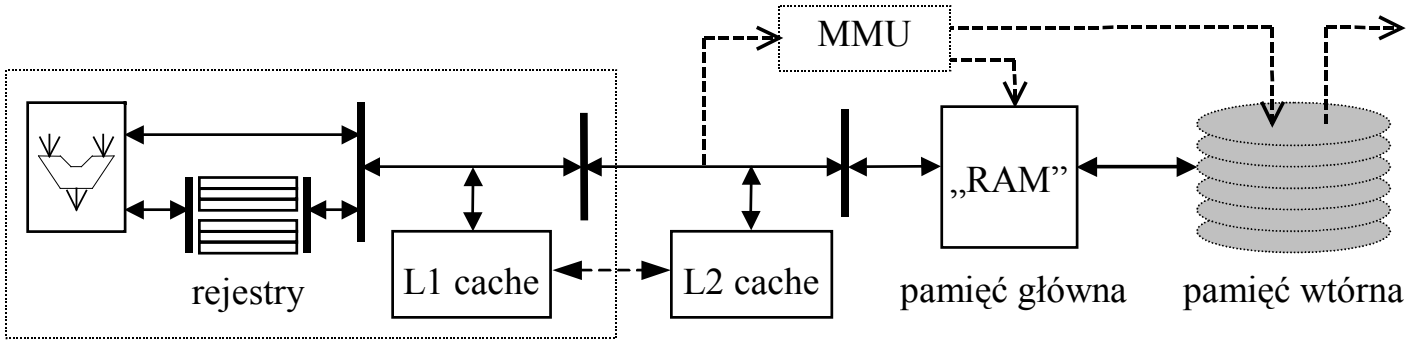
w systemie wieloprocesorowym także:

– spójność (ang. *coherency*) – brak bloku w buforze może być spowodowany działaniem innego procesora, używającego tej samej pamięci operacyjnej.

Redukcja chybień

1. Zwiększenie rozmiaru bloku/linii
   – czynnik/efekt/zjawisko: wzrost lokalności przestrzennej
   * wady: wzrost strat czasu w razie chybienia (dłuższy czas kopiowania bloku)
2. Zwiększenie pojemności bufora cache
   – czynnik/efekt/zjawisko: słabszy efekt ograniczonej pojemności
   * wady: dłuższy czas dostępu, większy pobór energii, wyższy koszt
3. Lepszy efekt skojarzeniowości
   – czynnik/efekt/zjawisko: rzadsze chybienia wskutek konfliktu
   * wady: dłuższy czas wyszukiwania bloku i w efekcie czas dostępu
4. Bufor wielopoziomowy
   – czynnik/efekt/zjawisko: mały szybki bufor 1. poziomu, duży, wolniejszy bufor 2.poz.
   * wady: trudniejsza obsługa w porównaniu z buforem jednopoziomowym
5. Priorytet chybień odczytu nad zapisem
   – czynnik/efekt/zjawisko: - dodatkowy bufor zapisu
   * wady: możliwy hazard odczyt po zapisie (ang. *read after write*)
6. Unikanie translacji adresu wirtualnego
   – czynnik/efekt/zjawisko: identyfikatorem w cache jest adres rzeczywisty
   * wady: rozmiar bufora 1.pozimu równy rozmiarowi strony

Pamięć wielopoziomowa



t_a	0,25–1 ns	0,5–2 ns	1–5 ns	10–50 ns	1–10 ms
rozmiar	1–8 kb	32–128 kB	¼–1MB	256 MB–4GB	20–120 GB
	register	*cache*	*cache*	*memory*	*storage*

Hierarchia pamięci wielopoziomowej

- *niektóre przestrzenie* adresowe lub ich części *nie mogą być buforowane*
- *procesor adresuje pamięć główną* – sterownik bufora przechwytuje transfery
- *specjalne rozkazy* – sposób działania wewnętrznego bufora cache

Wielopoziomowy bufor pamięci podręcznej

zasada lokalności → ma sens buforowanie bufora lub jego rozdzielenie
→ pamięć podręczna wielopoziomowa

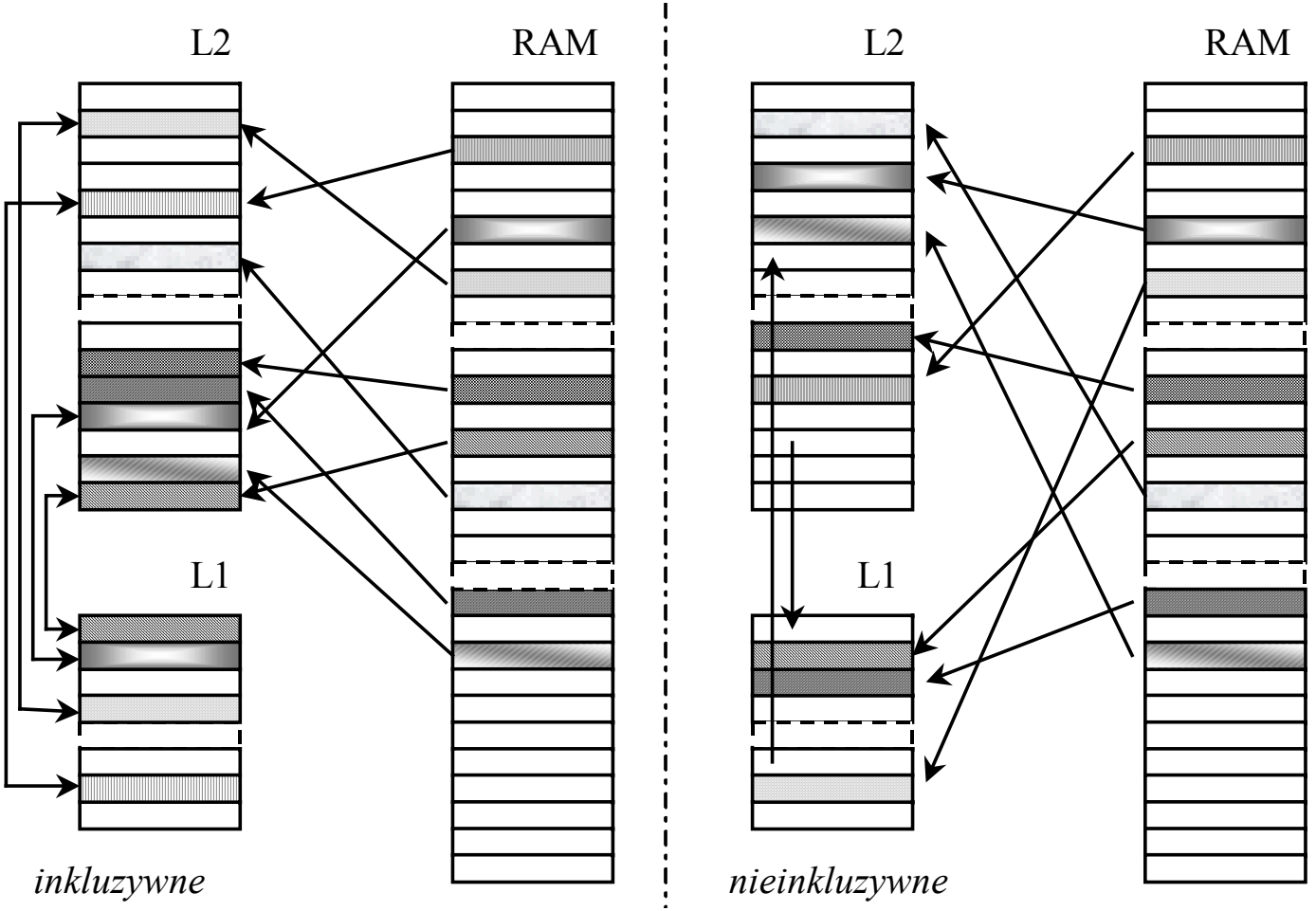
- *organizacja hierarchiczna* – pamięć *inkluzywna*
  → *lokalizacja* danej umieszczonej w pamięci L(i) dostępna w L(i+1)
  → bufor L(i) poziomu i zawiera kopie niektórych danych z bufora L(i+1) poziomu i+1, niektóre linie w L(i) mogą być zaktualizowane
  → *dana aktualna* dostępna *zawsze na najniższym poziomie*

- *organizacja równoległa* – pamięć *nieinkluzywna* (ATHLON™ – *victim cache*)
  → żadna lokalizacja danej w buforze L(i) nie jest dostępna w L(i+1)
  → dostęp do bufora L(i) znacznie krótszy niż do L(i+1)
  → w buforze L(i+1) są dane usunięte z L(i)

problem spójności

- zgodność wszystkich kopii informacji, albo
- znajomość (świadomość) lokalizacji informacji aktualnej

Odwzorowanie danych w buforze wielopoziomowym



Obsługa wielopoziomowego bufora cache

Inkluzywny (hierarchiczny) bufor cache

Każda linia bufora poziomu niższego (L1) ma swój oryginał
w buforze cache poziomu wyższego (L2)

Usunięcie zmodyfikowanej linii z bufora L1
wymusza aktualizację oryginału w buforze L2
Usunięcie niezmodyfikowanej linii z bufora L1
nie powoduje żadnej akcji w buforze L2

Nieinkluzywny (równoległy) bufor cache

Żadna linia bufora poziomu niższego (L1) nie ma kopii
w buforze cache poziomu wyższego (L2)

Usunięcie linii z bufora L1 wymusza wymianę linii z buforem L2

Usunięcie linii z bufora L2 wymusza aktualizację linii w pamięci (buforze L3)

Skuteczność użycia bufora wielopoziomowego

(t_{mp} – kara za chybienie (*miss penalty*))

Średni czasu dostępu do pamięci

- w buforze jednopoziomowym

$$t_a = (1-m)\,t_{ca} + m\,(t_{ram} + t_{mp})$$

- w buforze dwupoziomowym o strukturze inkluzywnej – L1⊂L2)

$$t_a = (1-m_1)\,t_{ca1} + (m_1 - m_2)(t_{ca2} + t_{mp1}) + m_2(t_{ram} + t_{mp1} + t_{mp2})$$

($m_1 > m_2$, bo każde trafienie w L1 jest trafieniem w L2,
a każde chybienie w L2 jest chybieniem w L1)

- w buforze dwupoziomowym o strukturze nieinkluzywnej – L1∩L2=∅)

$$t_a = (1-m_1)\,t_{ca1} + (m_1 - m_2)\,t_{ca2} + (m_1 + m_2)(t_{ram} + t_{mp})$$

(trafienie w L1 jest chybieniem w L2, trafienie w L2 jest chybieniem w L1,
skutkiem podwójnego chybienia jest wypełnienie L1)

Parametry bufora cache

bufor poziomu L1

- rozmiar linii – dostosowany do rozmiaru magistrali danych procesora (systemu) i możliwości transferów blokowych
- rozmiar pamięci – wystarczający do pomieszczenia kilku (2^N) stron lub innych jednostek przydziału (bloków) pamięci

bufor poziomu L2

- rozmiar linii – identyczny jak w L1
- rozmiar bufora – wystarczający do pomieszczenia zbioru roboczego procesu

Typowe parametry

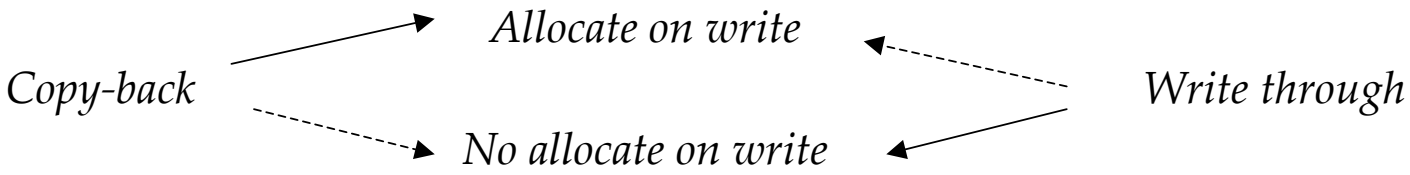
- szerokość magistrali danych – 2^k bajtów,
- transfer blokowy – nałożenie 2^m transferów całą szerokością magistrali
  $\rightarrow$ linia powinna zawierać $2^{k+m} = 2^B$ bajtów
- rozmiar strony – 2^P bajtów
  $\rightarrow$ rozmiar bufora L1 – 2^N stron = 2^{P+N} bajtów
  $\rightarrow$ łączna liczba linii w buforze L1 = 2^{P+N-B}
  $\rightarrow$ rozmiar bufora L2 – 2^W 2^N stron

Aktualizacja zawartości bufora cache

Warunkiem utrzymania spójności jest aktualizacja linii, która obejmuje

- unieważnienie linii zawierającej dane przypadkowe lub nieaktualne
- wypełnienie linii (bloku) nową zawartością
- wymiana linii w razie braku wolnego miejsca w buforze
- zapis w obszarze skopiowanej linii
  - *zapis skrośny* (ang. *write through*) – zgodność wszystkich kopii informacji
  - *zapis lokalny* (ang. *copy-back*) – znajomość lokalizacji informacji aktualnej

Zapis może być poprzedzony kopiowaniem nieobecnej linii (ang. *allocate on write*)



Obciążenie magistrali pamięci transferami powinno być minimalne

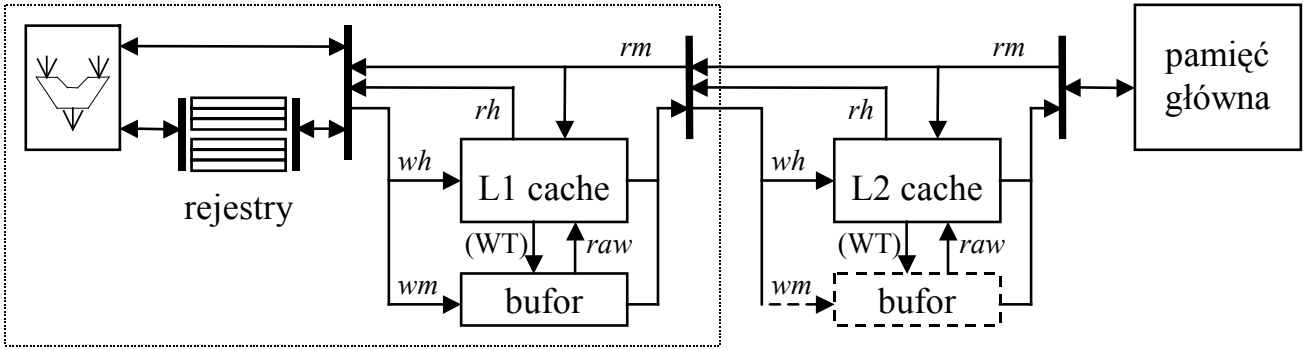
$$memory\ \ traffic\ \ ratio = \frac{transfery(memory + cache)}{transfery(cache)}$$

Bufory zapisu (*write buffer*)

Chybienie podczas próby zapisu

- w trybie WT (*no allocate-on-write*) wymaga transferu do poziomu wyższego
- w trybie CB uprzedzające kopiowanie (*allocate-on-write*) wymaga blokady zapisu oraz powoduje niezgodność kopii (zapis lokalny) po odblokowaniu
- w trybie CB kopiowanie odłożone (*no allocate-on-write*) do chybienia w razie odczytu *lub kolejnego zapisu* nie powoduje strat jeśli zapis jest buforowany

Bufor zapisu: pamięć FIFO (kolejka – musi być zachowana kolejność zapisów)
→ nasycenie bufora (ang. *write buffer saturation*) → blokada kolejnego zapisu



Bufory zapisu i ścieżki przepływu danych w systemie pamięci
(*rm/rh – read miss/hit, wm/wh – write miss/hit, raw – read after write*)

Obsługa pamięci podręcznej

- unieważnianie linii (ang. *line invalidation*)
  – przed pierwszym wypełnieniem
  – wskutek zewnętrznej zmiany oryginału danych w pamięci głównej
  – przełączanie procesów – unieważnienie wszystkich linii (*line flush*)
- wypełnianie linii (ang. *line fill*) oraz wymiana linii (*line exchange*)
  – chybienie odczytu (ang. *miss on read*) lub (w trybie AOW) zapisu
- odczyt danej (ang. *read*)
  – trafienie odczytu (ang. *hit on read*)
- zapis danej (ang. *write*) → rozbieżność kopii z oryginałem
  – trafienie zapisu (ang. *hit on write*)
    * zapis skrośny (jednoczesny) (ang. *write through*, WT) – modyfikuje kopię w buforze wyższego poziomu
    * zapis lokalny (zwrotny) (ang. *write/copy back*, WB/CB) – w kopii lokalnej, opóźniony zapis do bufora wyższego poziomu podczas usuwania linii
    * zapis bezpośredni do pamięci głównej (ang. *write aside*) i unieważnienie linii
  – chybienie zapisu (ang. *miss on write*) – zapis do pamięci głównej lub bufora poziomu wyższego (NAOW) lub wypełnienie i zapis (AOW)

Zarządzanie pamięcią podręczną

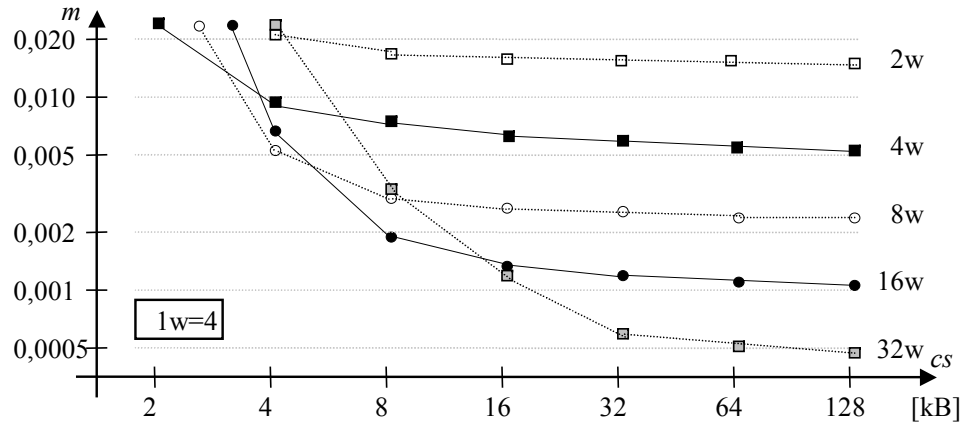
Sposoby umieszczania i aktualizacji danych w buforze:

- schematy (ang. *strategies*) wypełniania pamięci podręcznej
  – odwzorowanie pamięci głównej w liniach pamięci podręcznej
  – losowe – zbiór linii w zbiór lokacji linii (dowolnie)
  – bezpośrednie – podzbiór linii w linię lokacji
  – blokowo-skojarzeniowe – podzbiór linii w podzbiór lokacji linii

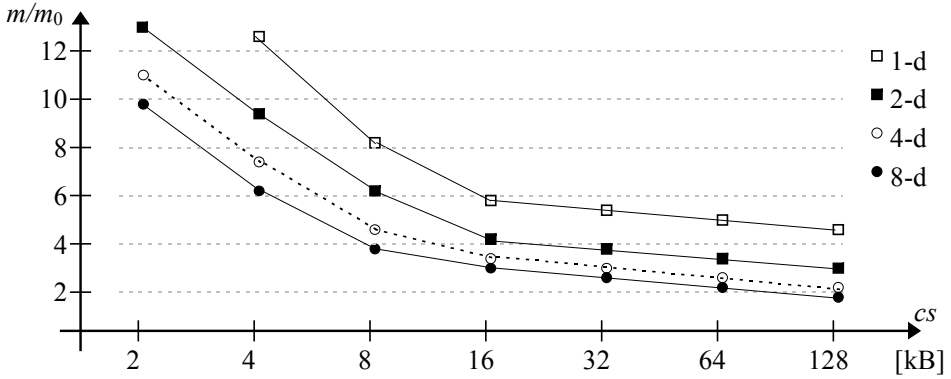
- schematy *wymiany* kopii w pamięci podręcznej w celu *aktualizacji* bufora (*bezzasadne w odwzorowaniu bezpośrednim*!)
  – losowa (ang. *random*) – dowolnie (skuteczna przy odwzorowaniu losowym)
  – kolejkowa (FIFO) – kolejność wymiany linii jest zgodna z kolejnością ich wypełniania (usuwana jest linia najdawniej alokowana)
  – LRU (ang. *last recently used*) – wymieniana jest linia najdawniej użyta (skuteczna przy odwzorowaniu blokowo-skojarzeniowym).

- schematy *pobierania* linii z pamięci głównej
  – pobranie wymuszone (ang. *fetching on demand*) – uaktywniane chybieniem
  – pobranie uprzedzające (ang. *prefetching*) – na podstawie prognozy dostępu.

Organizacja pamięci podręcznej a charakterystyki skuteczności



Zależność współczynnika chybień od rozmiaru linii (pamięć dwudrożna)



Wpływ organizacji pamięci na częstość konfliktów odwzorowania

Schematy wymiany linii

Aktualizacja bufora podczas przełączania zadań

- bufor „ciepły” (ang. *warm cache*) – część bufora nie jest wymieniana
- bufor „zimny” (ang. *cold cache*) – unieważnianie całego bufora

Algorytmy wymiany

- losowy – tylko w buforze całkowicie asocjacyjnym
  - o – ryzyko usunięcia potrzebnej linii – 2^{-K} (K – liczba lokacji w buforze)
- kolejkowy (FIFO)
  - o – liczba bitów historii – $\log_2 S$ (S – liczba lokacji w podzbiorze)
  - o – ryzyko usunięcia potrzebnej linii – $p\ 2^{-S}$
- wg używalności (LRU)
  - o – liczba bitów historii – $(S–1)\log_2 S$ (S – liczba lokacji w podzbiorze)
  - o – ryzyko usunięcia potrzebnej linii – $p\ 2^{-S}$

Markowanie używalności w buforze blokowo-skojarzeniowym:

- 2 linie w grupie: 0/1
- 4 linie w grupie: 2 podgrupy, markowanie podgrupy 0/1 i linii 0/1
- 8 linii w grupie: hierarchiczne markowanie podgrup binarnych

Schematy pobierania linii

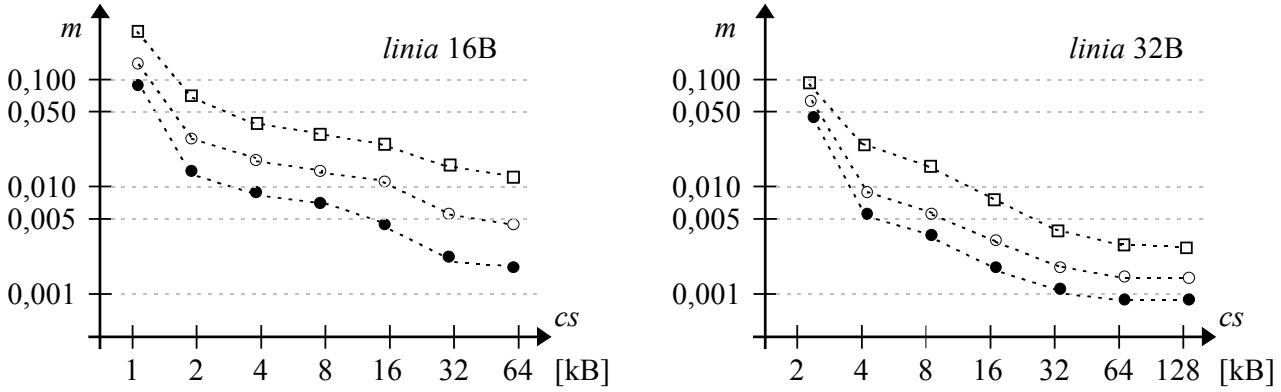
- pobranie wymuszone (ang. *demanded fetching*) – uaktywniane chybieniem
- pobranie uprzedzające (ang. *prefetching*) – na podstawie prognozy dostępu, nie powinno powodować opóźniania pobrań wymuszonych (rozmiar linii wpływa na szybkość wypełniania)

Antycypacja jednostopniowa (ang. *one block lookahead*, OBL)
– w chwili pobrania linii *i*-tej z pamięci głównej należy też pobrać linię *i*+1

Pobranie uprzedzające (antycypowane)
- *automatyczne* (ang. *prefetch always*), inicjowane podczas każdej próby dostępu
  → przy okazji zwrotu do linii *i* jest zawsze pobierana linia *i*+1
- *implikowane* w razie chybienia (ang. *prefetch on a miss*)
  → wraz z wymuszonym wskutek chybienia pobraniem linii *i* zawsze jest pobierana linia *i*+1
- *markowane* (ang. *tagged prefetch*)
  → podczas pierwszego odwołania do linii *i*-tej, wraz z nią jest pobierana linia *i*+1, która zostaje oznaczona (ang. *tagged*)

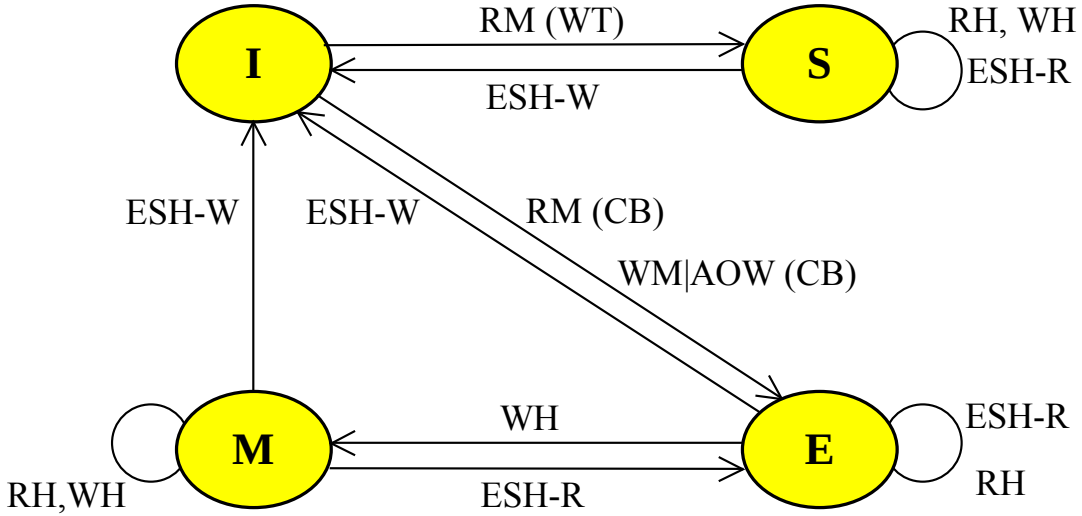
Intensywność pobrań uprzedzających

- *automatyczne* (ang. *prefetch always*)
  → obciążenie magistral o 20–80% większe niż w pobraniach wymuszonych
- *implikowane* w razie chybienia (ang. *prefetch on a miss*)
  → obciążenie magistral nieznacznie większe niż w pobraniach wymuszonych
- *markowane* (ang. *tagged prefetch*)
  → obciążenie magistral podobne jak w pobraniach wymuszonych



Współczynnik chybień *m* w funkcji rozmiaru pamięci podręcznej *cs* dla strategii pobrań (□ – wymuszone, o – implikowane, • – markowane, automat.)

Zintegrowany model spójności pamięci podręcznej



Stany linii pamięci podręcznej obsługiwanej w trybie WT lub CB

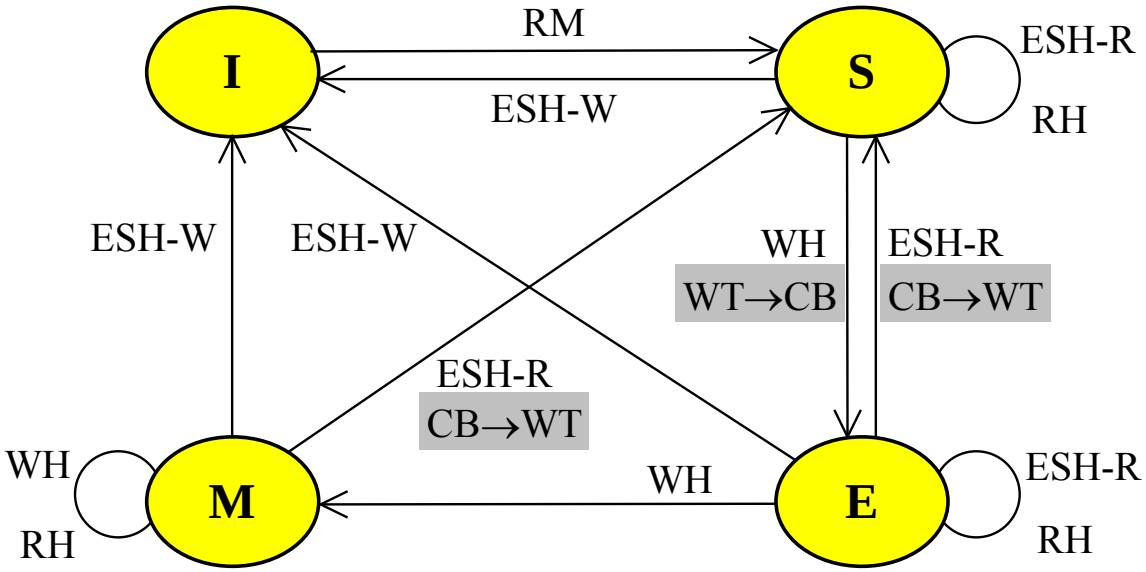
(I – nieważny (ang. *invalid*), E – zgodny (ang. *exclusive*), M – zmieniony (ang. *modified*), S – współdzielony (ang. *shared*))

M – chybienie (ang. *miss*), H – trafienie (ang. *hit*), R – podczas odczytu, W – podczas zapisu, ESH – podglądnięcie trafienia zewnętrznego (ang. *external snoop hit*)

Strategia zapisu jednorazowego (*write-once*)

(Pentium) niewiele rejestrów → dużo zmiennych roboczych w pamięci

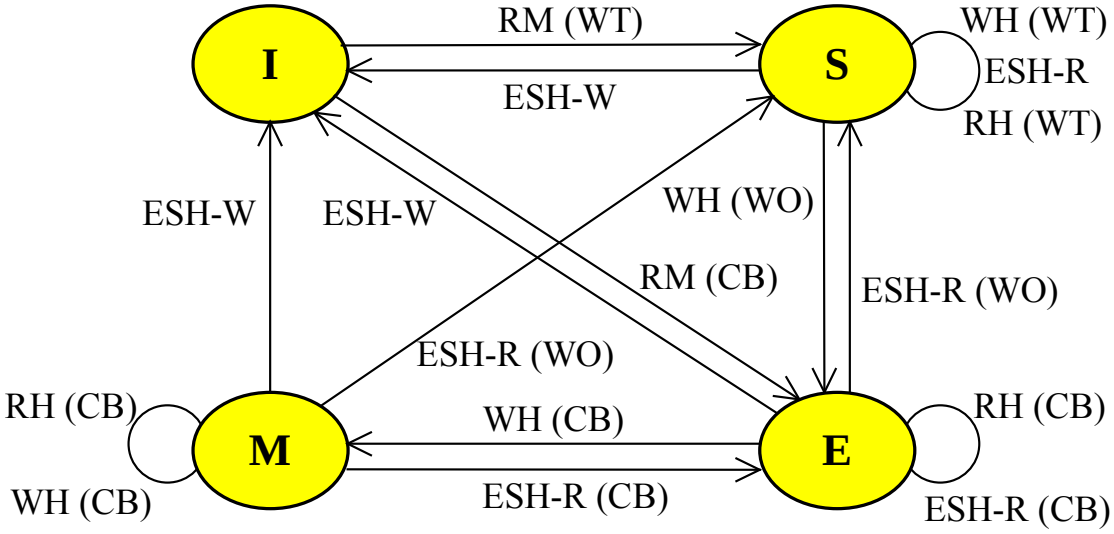
→ pierwszy zapis skrośny (WT), kolejne zapisy lokalne (CB)



Stany linii pamięci podręcznej obsługiwanej w trybie WO

L1: WT→CB, L2: CB – krótki czas zapisu, częściowa zgodność danych

Pełny model spójności (MESI)



Model spójności wielopoziomowej pamięci podręcznej (protokół MESI)

Obsługa pamięci dwupoziomowej:

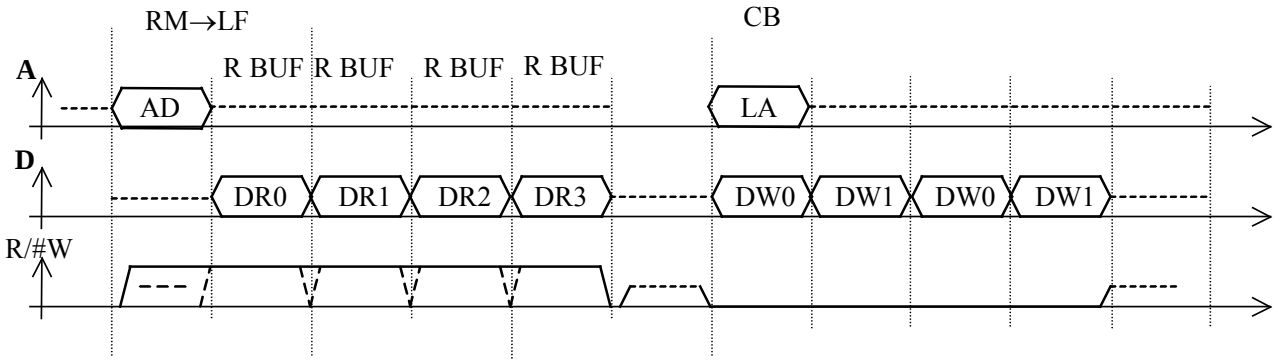
L1: WT, L2: WT – długi czas zapisu, całkowita zgodność danych

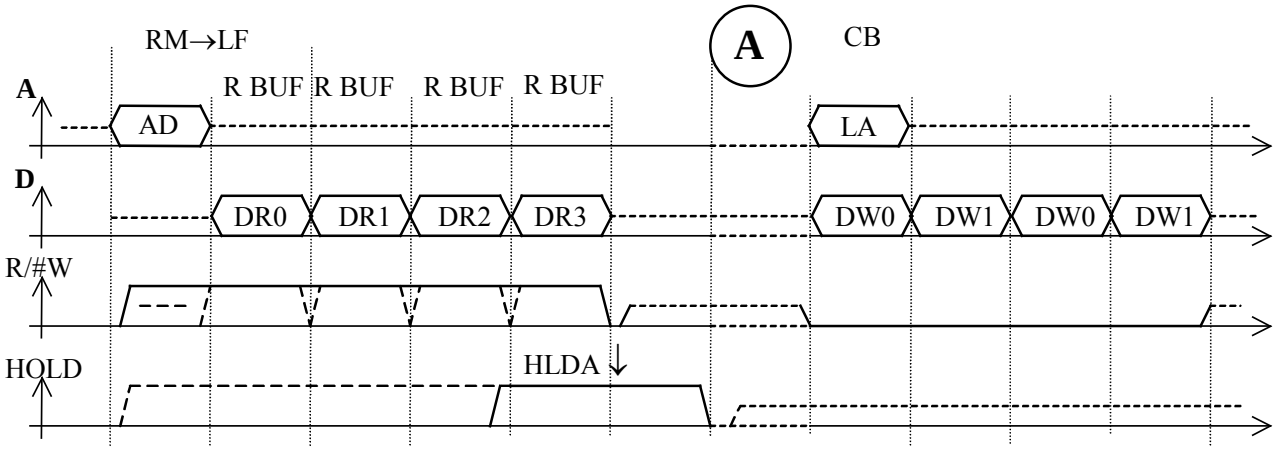
L1: CB, L2: CB – najkrótszy czas zapisu, utrata zgodności danych po zapisie

L1: WT, L2: CB – krótszy czas zapisu, zgodność danych L1-L2

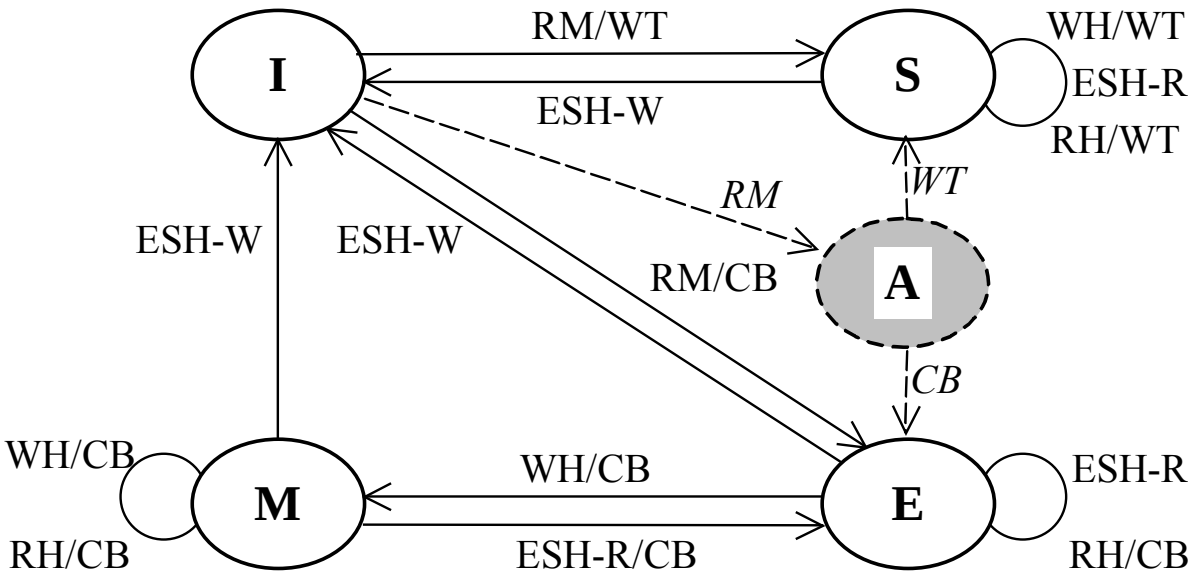
L1: CB, L2: WT – utrata zgodności danych po zapisie, długi czas wymiany

Wymiana linii – aspekty czasowe





Rozszerzony model spójności MESI



(I – nieważny (*invalid*), E – zgodny (*exclusive*), M – zmieniony (*modified*), A – przydzielony (ang. *allocated*), S – współdzielony (*shared*))

RH / WH – trafienie podczas odczytu (R) / zapisu (W), RM – chybienie podczas odczytu (*read miss*), ESH – podglądnięcie trafienia zewnętrznego (*external snoop hit*),

Spójność pamięci w systemie wieloprocesorowym

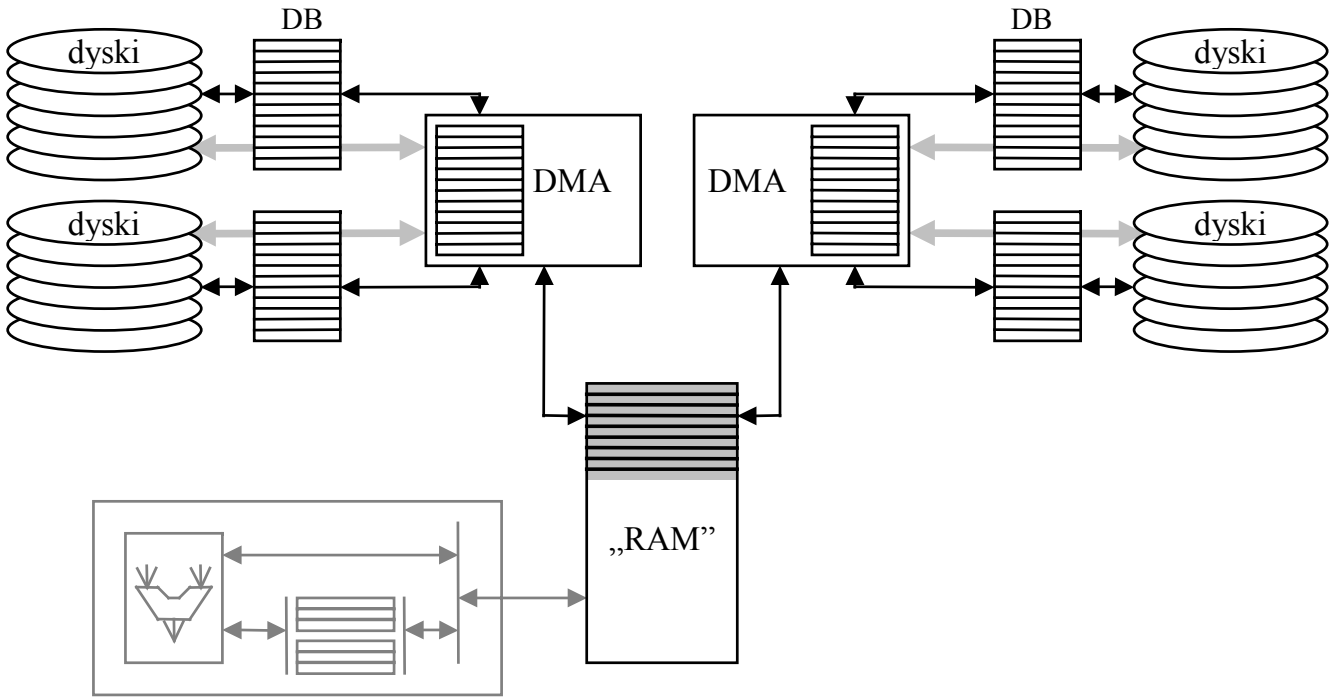
Spójność
- zgodność wszystkich kopii informacji, albo
- znajomość (świadomość) lokalizacji informacji aktualnej

Protokół uzgadniania
- działania procesora (ang. *CPU action*)
- działania na magistrali (ang. *bus action*) – podglądanie magistrali (ang. *snooping*)

Modele spójności systemu wieloprocesorowego ze wspólną pamięcią
- zapisz–unieważnij (ang. *write–invalidate*)
  - Synapse– aktualizacja lokalna, inne kopie unieważniane
  - Illinois – aktualizacja lokalna, kopia markowana (wyłączna – dzielona), podobny protokół jak w modelu MESI
  - Berkeley – aktualizacja lokalna, inne kopie tylko do odczytu
- zapisz–aktualizuj (ang. *write–update*)
  - Firefly – aktualizacja globalna, kopia markowana (wyłączna – dzielona)
  - Dragon – aktualizacja ograniczona (z wyłączeniem pamięci głównej), kopia markowana (wyłączna – dzielona)

Pamięć podręczna dysku (*disk cache*)

– bufor wirtualnej przestrzeni adresowej
– obsługa : *copy back* (*write through* wymusza niepotrzebne czasochłonne zapisy)
– rozmiar: wielokrotność rozmiaru sektora, ścieżki, cylindra itp.



Możliwe usytuowania bufora pamięci podręcznej dysku

Pamięć podręczna tablicy stron (TLB)

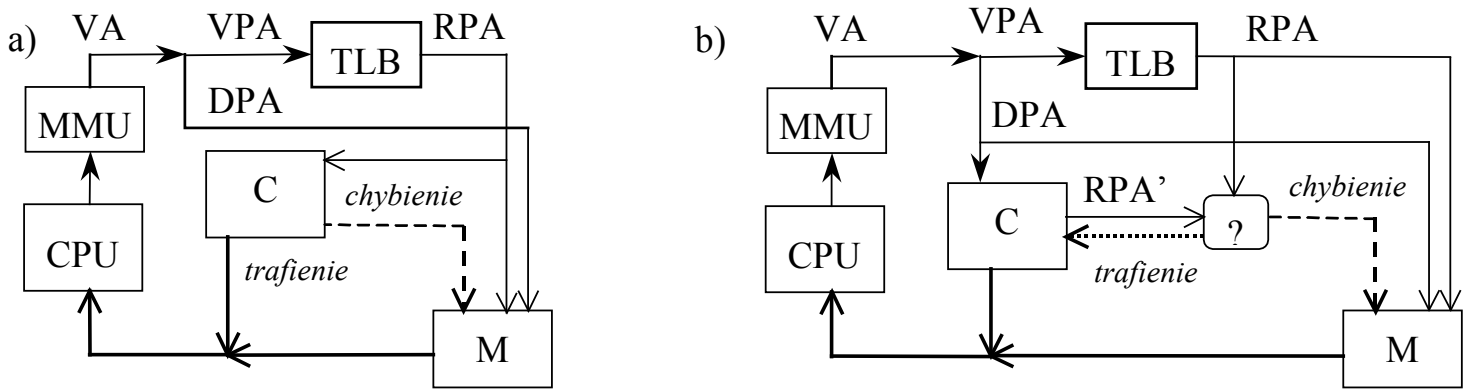
Odwzorowanie wirtualnej przestrzeni adresowej w realną opisuje **tablica stron**

Liczba stron wirtualnych jest ogromna, więc pełna tablica zajmowałaby bardzo duży obszar rzeczywistej przestrzeni adresowej.

Zamiast tego, na podstawie zasady lokalności w skali makro,
- tworzony jest ograniczonego rozmiaru bufor translacji bieżących, stanowiący w istocie pamięć podręczną tablicy stron oraz
- określony jest wynik każdego odwzorowania opisanego w buforze (pełna tablica stron fizycznie nie musi istnieć)

Przyśpieszanie translacji adresu podczas dostępu do pamięci podręcznej*)

W środowisku wielozadaniowym unikatowe adresy wirtualne są przekształcane w adresy rzeczywiste, np. przez tablicę stron. Bufor cache wspomaga pamięć operacyjną, więc powinien zawierać wskaźniki rzeczywiste bloków. Źródłowym wskaźnikiem danej jest jednak w tym środowisku adres wirtualny



Adres wirtualny jako wskaźnik bloku podczas dostępu do pamięci podręcznej: CPU – procesor, M – pamięć główna, C – pamięć podręczna, MMU – jednostka zarządzania pamięcią, TLB – bufor translacji, VA – adres wirtualny, VPA – adres strony wirtualnej, RPA – adres strony rzeczywistej, DPA – bezpośredni adres na stronie