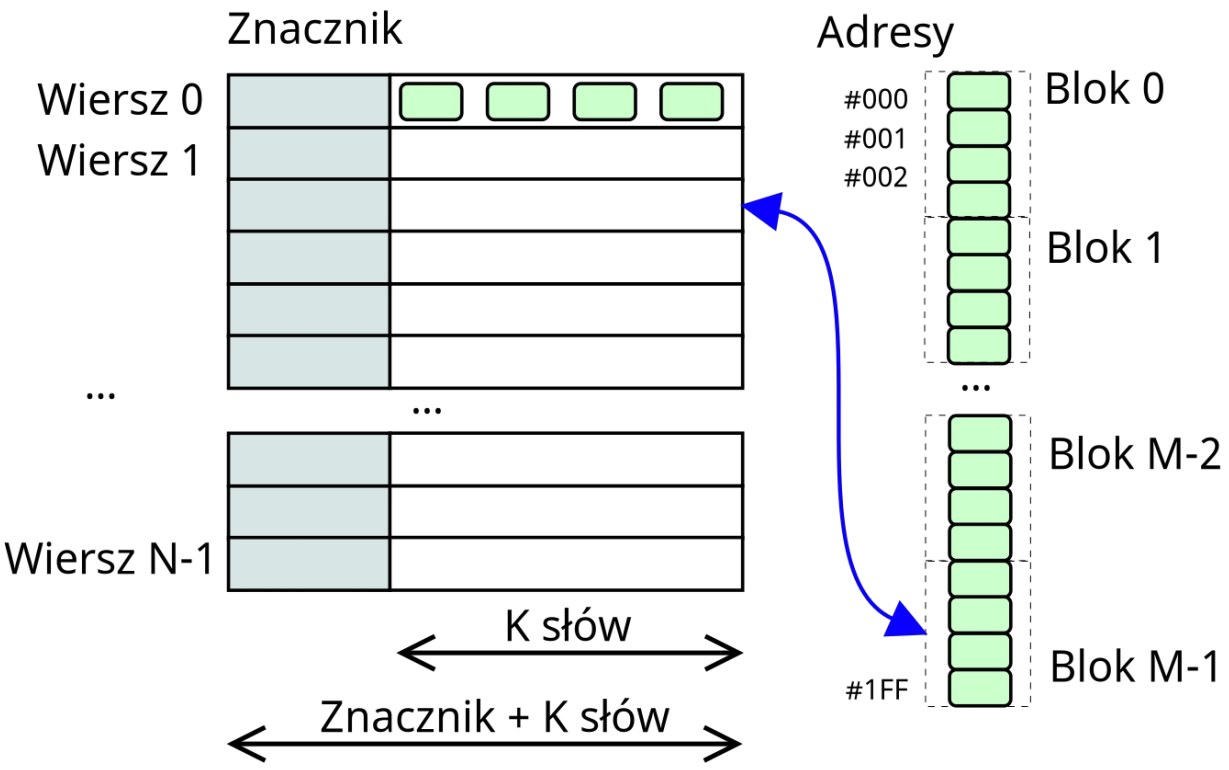
Realizacja pamięci cache

- Łączymy szybką, drogą (więc: małą) pamięć z wolniejszą, tańszą (zatem: większą)
- Szukamy najpierw blisko procesora, potem schodzimy w dół
  - Po odnalezieniu aktualizujemy wyższe warstwy
- Przy okazji ściągamy „okolice”, żeby były pod ręką (*zasada lokalności*)

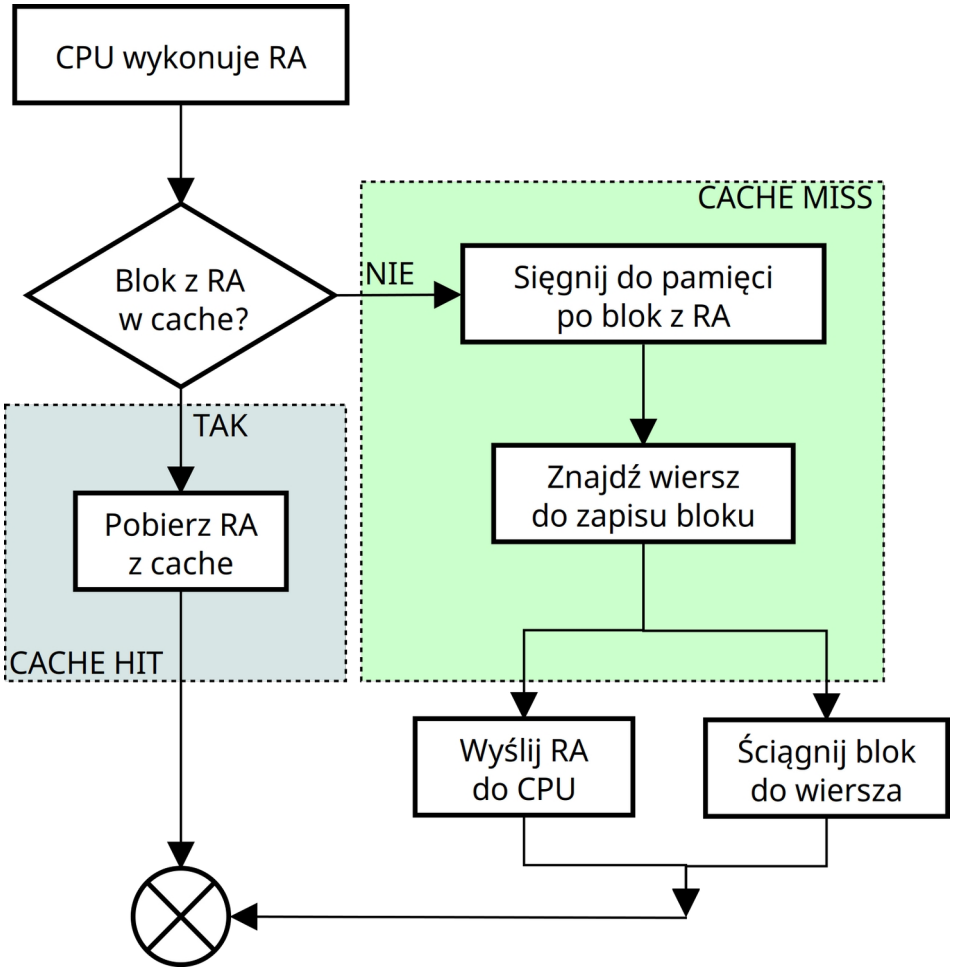
Pamięć cache vs. pamięć główna



- **Słowo** – 1, 2, 4 bajty
- **Blok** – jedn. transferu (tu: 4 słowa)
- **Wiersz** – zawiera 1 blok
- **Znacznik** – adresuje dane w cache
- $N \ll M$
- M = 2^n / K (tu n=9)

- Pewien podzbiór pamięci (bloków) zawsze jest w wierszach cache
  - sięgnięcie do bajtu/słowa z bloku B1 powoduje załadowanie B1 do któregoś wiersza cache
- Ta lokacja nie jest jednoznaczna (M >> N), więc potrzeba unikalnej identyfikacji bloków w cache

Read Address (RA) - procedura

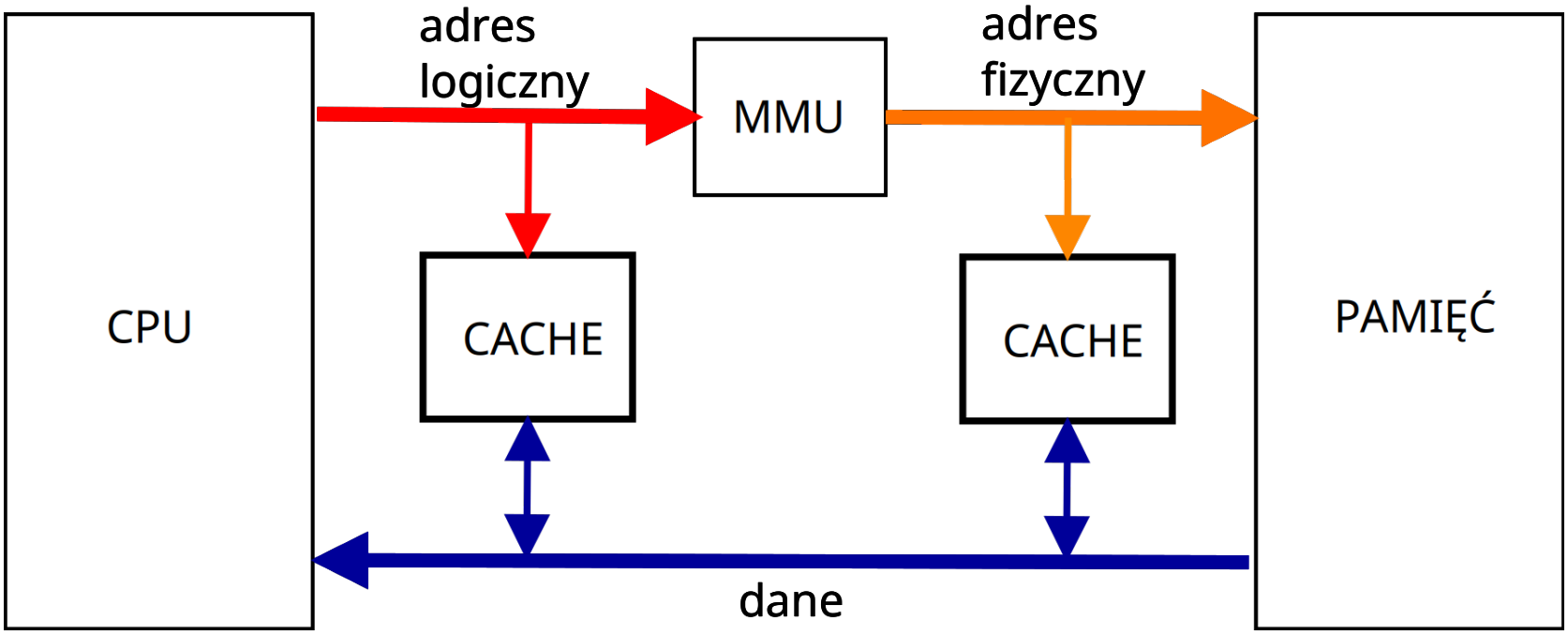


zob.
A.J. Smith, *Cache memories*

Do rozważenia

- Rodzaj przechowywanej pamięci
- Sposób odwzorowania pamięci głównej
- Mechanizm zastępowania wierszy
- Sposób zapisu/aktualizowania pam. głównej
- Rozmiar
- Długość wiersza (liczba słów)
- Wielopoziomowość

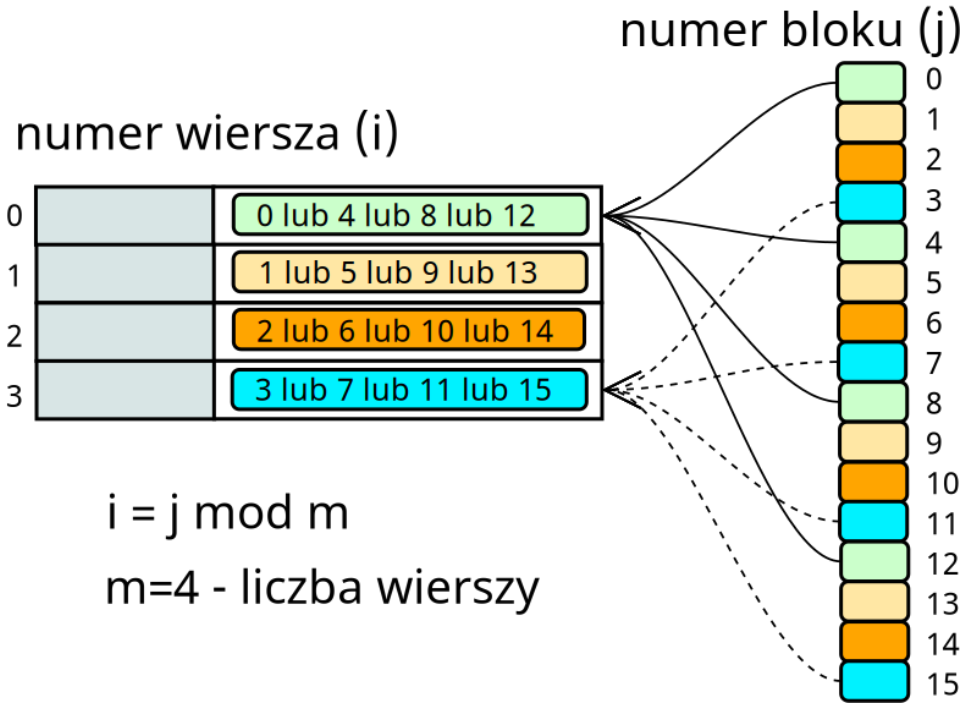
Pamięć przestrzeni fizycznej i logicznej



Przykład organizacji

- Rozmiar cache: 64kB
- Przesyłane bloki – 4B
  - Ile jest wierszy?
    - 64kB / 4B = 16 k = 2^14
- Rozmiar pam. głównej: 16 MB, adresowalna bajtowo
  - Ile bitów adresu?
    - 16 MB = 2^**24**
  - Ile jest bloków?
    - 16 MB / 4B = 4 M

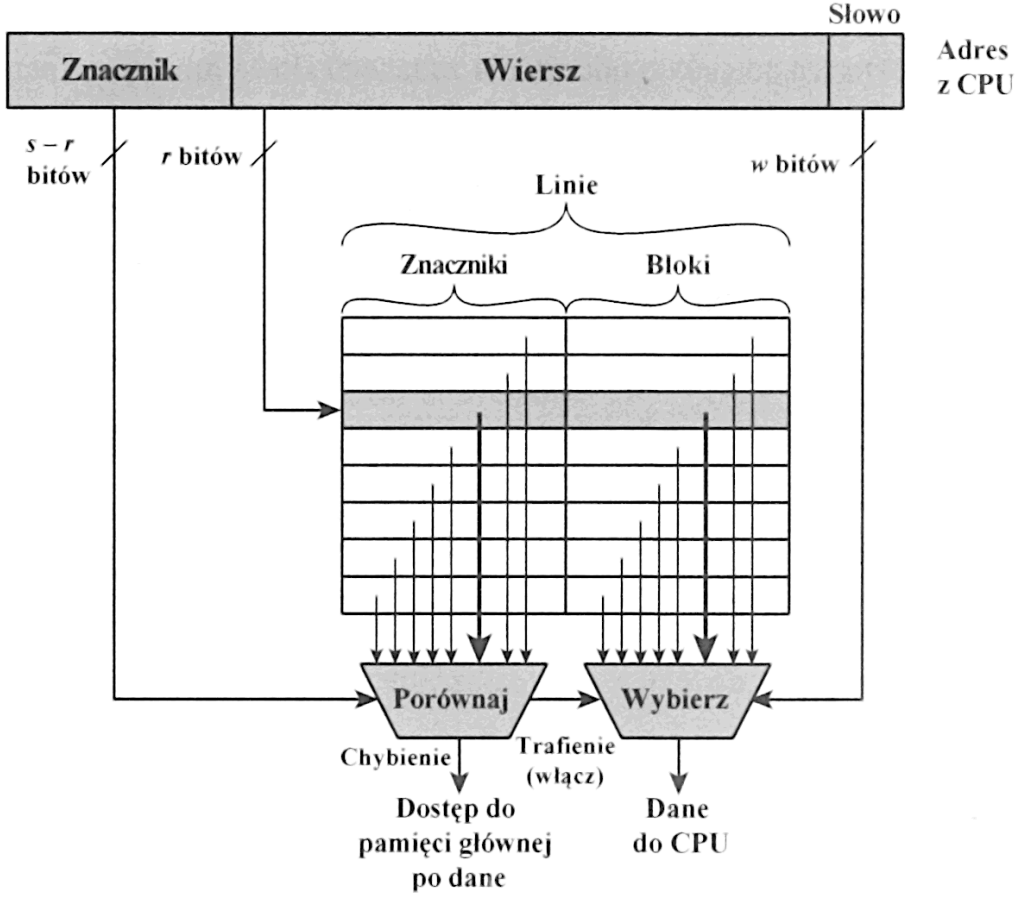
Odwzorowanie bezpośrednie



i = j mod m

m=4 - liczba wierszy

i				
0	00 **00**	01 **00**	10 **00**	11 **00**
1	00 **01**	01 **01**	10 **01**	11 **01**
2	00 **10**	01 **10**	10 **10**	11 **10**
3	00 **11**	01 **11**	10 **11**	11 **11**

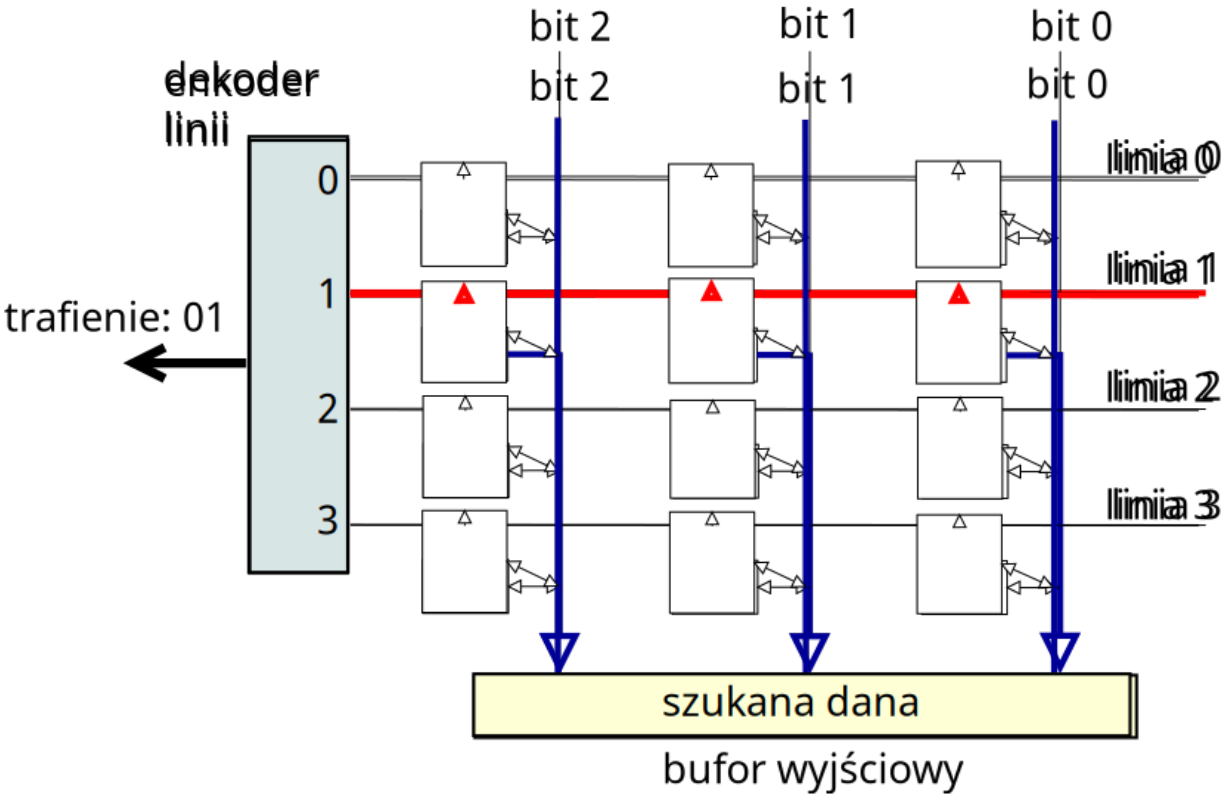


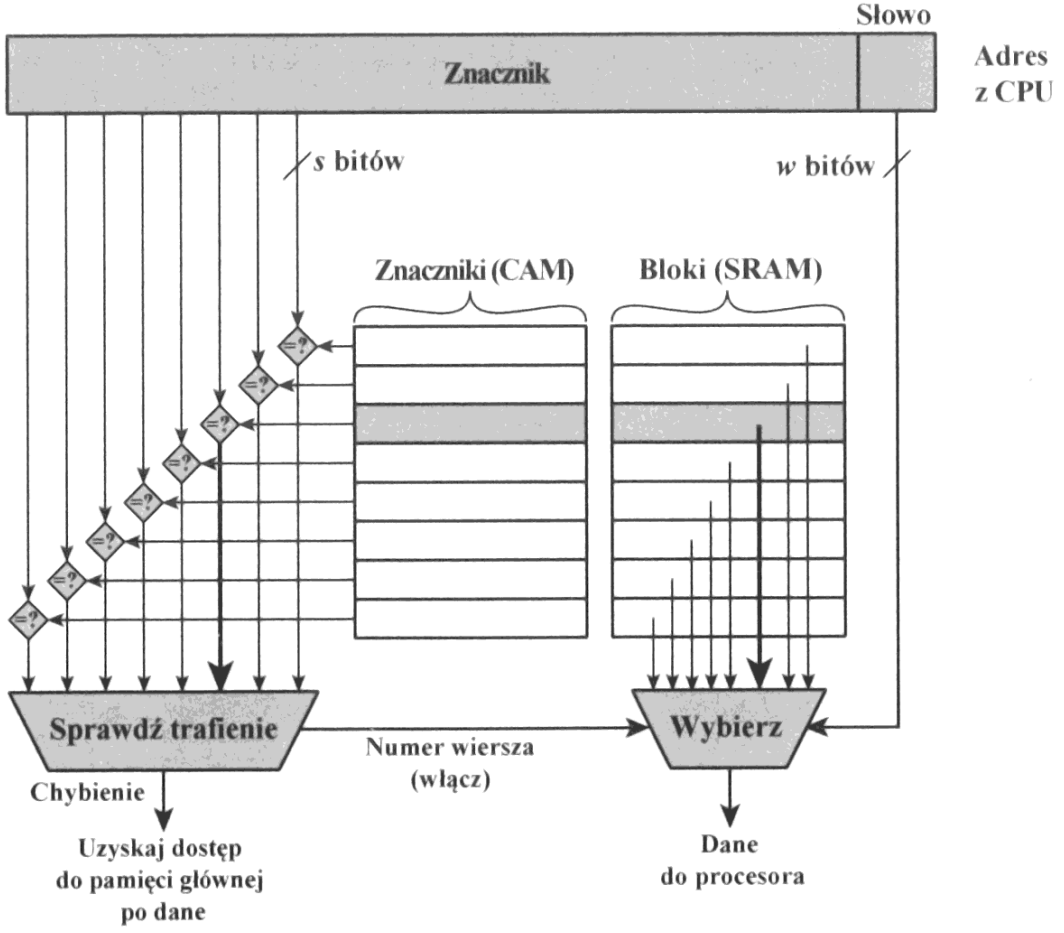
Rysunek 5.7. Organizacja pamięci podręcznej z odwzorowaniem bezpośrednim

Odwzorowanie bezpośrednie

- Każdy blok ma stałą lokację (wiersz)
- Proste i tanie w realizacji
- Co jeśli program odwołuje się do By i Bx, gdzie
  $y \bmod m = x \bmod m$?
- Pamięć podręczna ofiar
  - pamiętanie usuwanych bloków
  - mniejsza pamięć 4-16 wierszy

Pamięć adresowana zawartością (CAM)



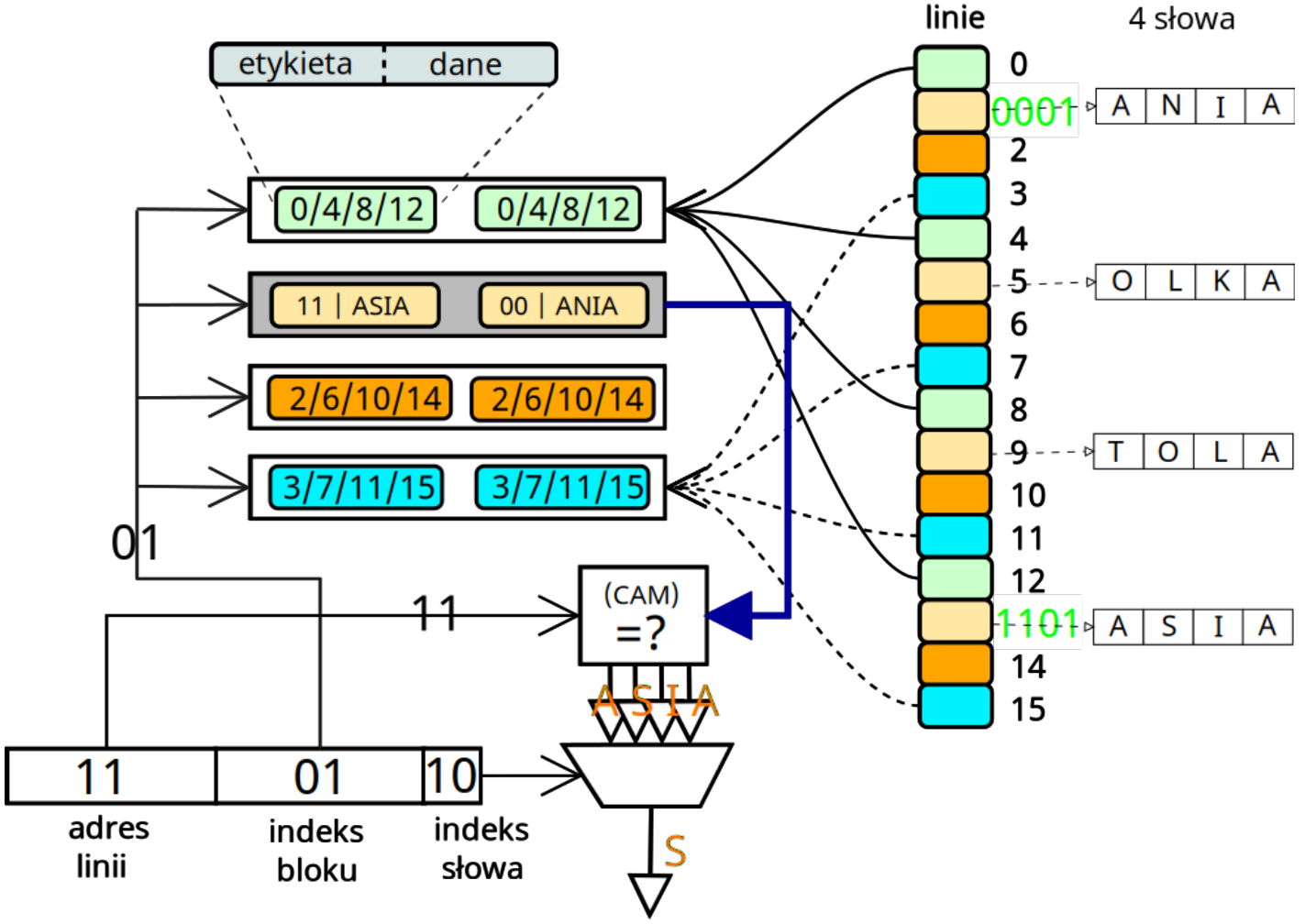


Rysunek 5.10. Organizacja w pełni skojarzeniowej pamięci podręcznej

Pamięć w pełni skojarzeniowa

- Adres trzeba porównać ze wszystkimi etykietami (złożone, trwa długo)
- Przechowywany znacznik jest duży (cały adres)
- Nie ma konfliktu odwzorowania (każda linia może trafić w każde miejsce)

Pamięć wielodrożna (sekcyjno-skojarzeniowa)



Pamięć wielodrożna

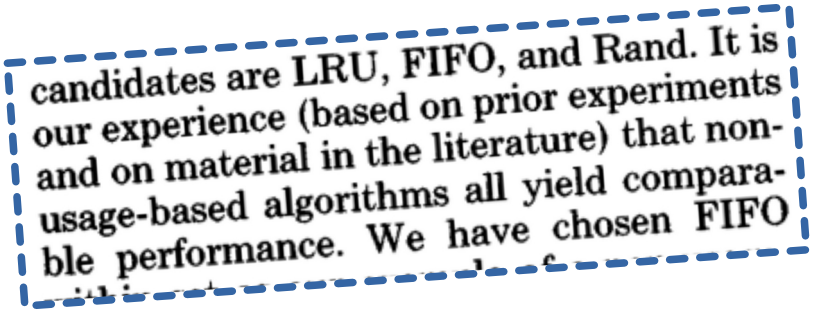
- Krótszy znacznik (część adresu)
- Mniej znaczników do porównania
- Typowo k=2, 4 (liczba wierszy na sekcję)

Algorytmy zastępowania linii

- Pamięć pełna, procesor żąda pamięci nie przechowywanej w cache: trzeba zrobić miejsce
- Dla odwzorowania bezpośredniego – jedno rozwiązanie
- Dla skojarzeniowych potrzebna jest strategia wymiany

Zastępowanie linii – pam. skojarzeniowa

- LRU – *least recently used*
  - bit USE dla każdego wiersza w sekcji (dla pam. dwudrożnej)
    - użyta linia: USE=1, druga linia: USE=0; do skasowania: z USE=0
  - lista indeksów wierszy (dla pam. w pełni skojarzeniowej)
    - ostatnio użyty wiersz idzie na początek; usuwamy ten z końca listy
- FIFO – *first in, first out*
  - bufor cykliczny indeksów wprowadzanych wierszy (np. w sekcji)
- LFU – *least frequently used*
  - licznik do każdego wiersza
- losowo


candidates are LRU, FIFO, and Rand. It is our experience (based on prior experiments and on material in the literature) that non-usage-based algorithms all yield comparable performance. We have chosen FIFO

Zapis linii

- Jest wiele miejsc przechowywania danej
- W przypadku zmiany danej – tylko jedno miejsce zawiera poprawną informację
- Jeśli zmodyfikowana linia w cache ma zostać zastąpiona – problem
  - trzeba o tym pomyśleć w momencie zapisu linii

Mechanizmy zapisu

- Zapis jednoczesny (*write through*)
  - każdy zapis propaguje do pam. głównej
  - p.g. zawsze aktualna
  - monitorowanie magistrali daje inf. o potrzebie aktualizacji
  - duży ruch, czasami niepotrzebny
- Zapis zwrotny (*write back*)
  - dla danego wiersza ustawiany *dirty bit*
  - przejściowa nieaktualność pam. gł.
  - przy zastępowaniu wiersza zapis tylko jeśli „*dirty*”
  - mniejszy ruch

Zapis przy chybieniu

- Alokacja z zapisem (write allocate)
  - najpierw pobierz blok
  - potem wykonaj zapis (jak na poprz. slajdzie)
- Alokacja bez zapisu (no-write allocate)
  - zmodyfikuj blok bezpośrednio w pam. gł.
  - nie ładuj do cache
- Czasami przy *write through* to nie ma alokacji, a przy *write back* jest – dlaczego?

Zachowanie spójności pamięci

- Co jeśli jest wiele procesorów (albo np. DMA)?
- Kilka podejść
  - monitorowanie magistrali i *write through*
  - dodatkowy sprzęt (centralny kontroler pamięci)
  - wyłącznie regionów pamięci ze współpracy z cache
  - organizacja na poziomie kompilatora

Zarządca pamięci

Utrzymywany jest katalog wszystkich danych przechowywanych w cache procesorów

Problem: wąskie gardło

1) P1 zgłasza chęć zapisu do „A”
2) Kontroler rozgłasza: *unieważnić swoje kopie* „A”
3) P1 dostaje wyłączny dostęp do „A”
4) P2 zgłasza chęć odczytu do „A”
5) Kontroler do P1: *zapisz zwrotnie swoje* „A”
6) P1 i P2 mają współdzielony dostęp do odczytu „A”

Podglądanie magistrali

- Każdy kontroler odpowiedzialny za spójność pamięci
- Realizowany jest pewien protokół
- Zapis z unieważnieniem (jeden pisze, wiele czyta)
  1) P1 rozgłasza swój zapis, P2, P… unieważniają swoje kopie, P1 pracuje na lokalnej kopii
  2) P3 zgłasza chęć dostępu, P1 zapisuje zwrotnie swoją kopię
- Zapis z aktualizacją (wiele pisze, wiele czyta)
  - W momencie aktualizacji wiersza jest on propagowany do innych

Protokół MESI

- Zmodyfikowany (M)
- Wyłączny (E)
- Wspólny (S)
- Nieważny (I)

	M	E	S	I
wiersz jest ważny?	tak	tak	tak	nie
kopia jest ważna?	nie	nie	tak	n/d
są inne kopie?	nie	nie	możliwe	możliwe
przy zapisie	zapis lokalny	zapis lokalny	na magistralę + cache	na magistralę

Diagram przejść MESI

READ MISS

1) jest czysta kopia na wyłączność
2) jest czysta kopia współdzieloną
3) jest zmodyfikowana kopia
4) nie ma kopii

READ HIT

5) w cache jest aktualnie

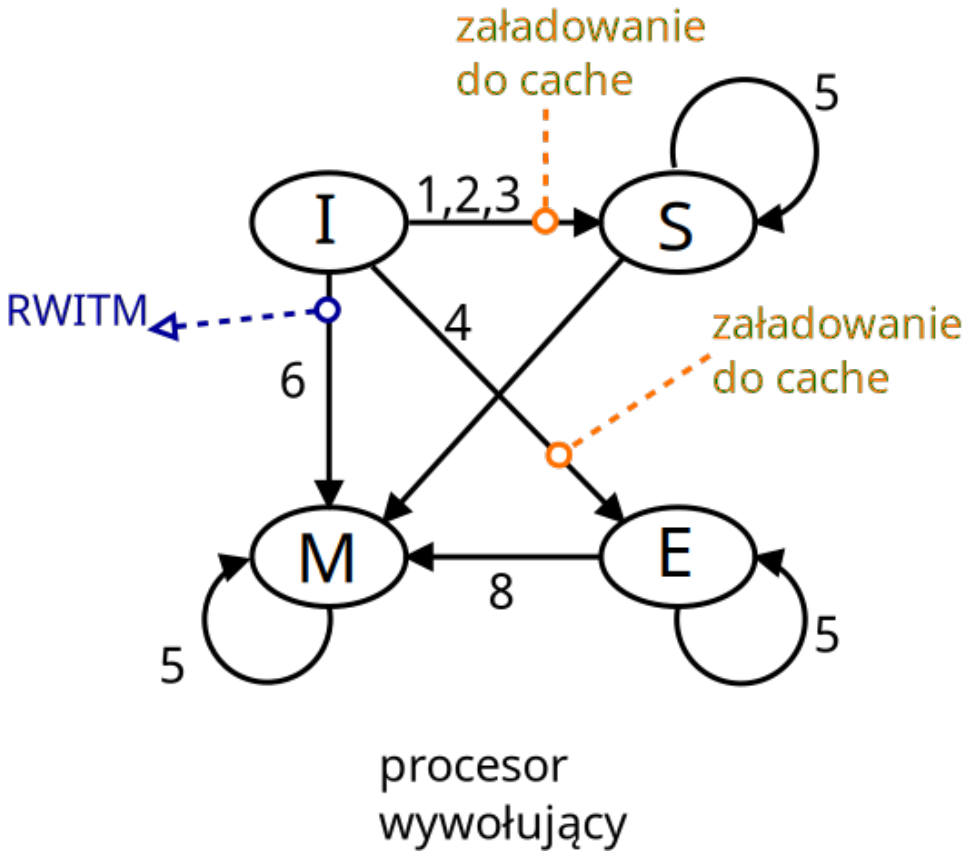
WRITE MISS

6) RWITM, aktualizacja cache
    1) jest zmodyfikowana kopia
    2) nie ma zmodyfikowanej kopii

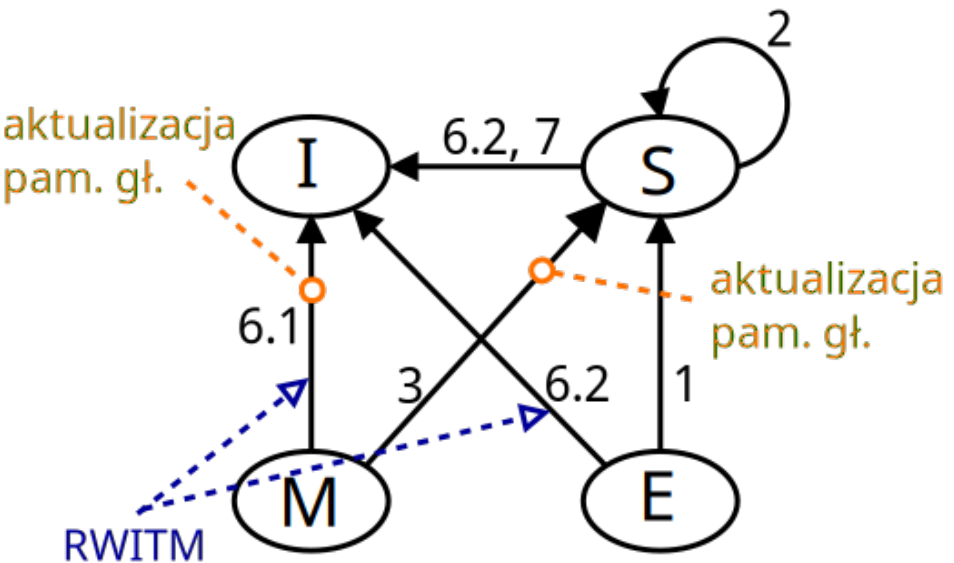
WRITE HIT

7) dana jest wspólna
8) dana jest wyłączna
9) dana jest zmodyfikowana

Przejścia stanów MESI



procesor
wywołujący



pozostałe
procesory

RWITM – Read With Intent to Write – *zamierzam czytać, żeby zapisać*