System pamięci

Pamięć podręczna

Technologia

- Static RAM (SRAM)
  - Ułamki nanosekund, $2000 – $5000 za GB
- Dynamic RAM (DRAM)
  - 50ns – 70ns, $20 – $75 za GB
- Dyski magnetyczne
  - 5ms – 20ms, $0.20 – $2 za GB
- Idealny system pamięci
  - Czas dostępu porównywalny z SRAM
  - Pojemność i koszt porównywalne z dyskami

DRAM i SRAM

- SRAM: bit informacji przechowywany dzięki ,,zapętleniu'' sygnału, jak w przerzutnikach;
- DRAM: bit informacji przechowywany w miniaturowym kondensatorze; ładunek ucieka i wymaga odświeżania co kilkadziesiąt milisekund
- Bity w DRAM są gęściej upakowane niż w SRAM

Hierarchia pamięci

- Rejestry
- Cache L1 (SRAM)
- Cache L2 (SRAM); ew. Cache L3
- Pamięć główna RAM (DRAM)
- Dyski magnetyczne
- Dyski optyczne
- Taśmy magnetyczne
- … rośnie czas dostępu i pojemność, maleje cena

Zasada lokalności

- Programy odwołują się w jednej chwili do niewielkiego fragmentu przestrzeni adres.
- Lokalność czasowa:
  - Słowa, do których się odwołujemy mogą być za chwile znowu potrzebne
  - rozkazy w pętli, zmienne indeksujące, liczniki
- Lokalność miejscowa:
  - Oprócz aktualnego słowa mogą się przydać za moment słowa sąsiednie
  - Np. dostęp do rozkazów, tablic

Cache – ogólna zasada

- Przechowujemy wszystko w pamięci głównej (DRAM), a ostatnio używane słowa oraz słowa sąsiednie dodatkowo w pamięci cache (SRAM)
- Słowa szukamy zawsze najpierw w cache, a jeśli tam go nie znajdziemy - ściągamy je z pamięci głównej do cache razem z całym **blokiem** danych
- Pamięć cache znajduje się na płytce procesora

Cache – ogólna zasada, cd.

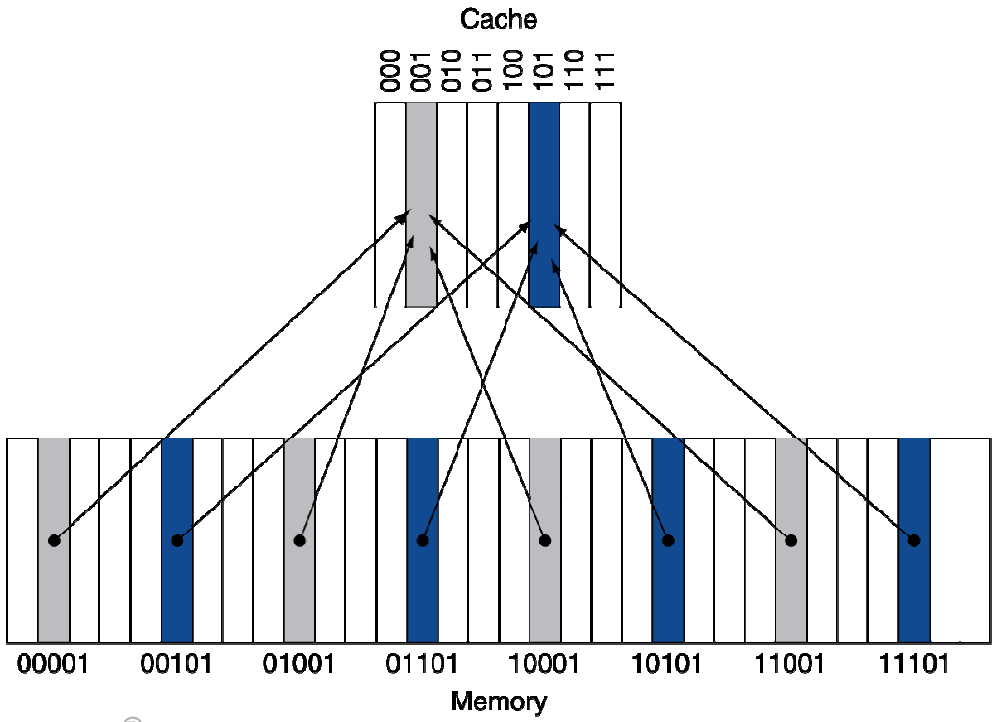
- Pamięć główna RAM podzielona na bloki
- Każdy blok to pewna, ustalona liczba słów
- Transfer RAM - Cache – całe bloki
- Transfer Cache – procesor – pojedyncze słowa
- **Trafienie** (hit): poszukiwane słowo znajduje się w cache; **chybienie** (miss): musi być ściągnięte z RAM
- Współczynnik trafień: #trafień / #odczytów

Cache - mapowanie

- Skąd wiemy, w którym miejscu w cache znajduje się poszukiwane słowo?
- Istnieje kilka możliwych rozwiązań:
  - mapowanie bezpośrednie
  - mapowanie skojarzeniowe
  - mapowanie sekcyjno-skojarzeniowe

Mapowanie bezpośrednie

- Adres (numer) bloku w RAM wyznacza **wiersz** (blok) w cache
- Nie ma wyboru:
  - (Nr bloku w RAM) modulo (#bloków cache)



- #wierszy cache - potęga 2
- Używamy mniej znaczących bitów numeru bloku w RAM

Znaczniki i bity aktualności

- Skąd wiadomo, który blok przechowywany jest w danym wierszu cache?
  - Informacja o adresie przechowywana jest razem z danymi
  - Wystarczą oczywiście bardziej znaczące bity
  - Nazywamy je **znacznikiem** albo tagiem
- A co jeśli w danym wierszu cache nie ma żadnych danych?
  - **Bit aktualności**: 1 = są, 0 = nie ma
  - Np. przy starcie komputera bity aktualności są zerowane

Przykład

- 8 wierszy, blok=słowo, mapowanie bezpośrednie
- Stan początkowy

Nr wiersza	Bit akt.	Znacznik	Dane
000	0		
001	0		
010	0		
011	0		
100	0		
101	0		
110	0		
111	0		

Przykład

Adres	Adres bin.	Trafienie?	Blok cache
22	10 110	nie	110

Nr wiersza	Bit Akt.	Znacznik	Dane
000	0		
001	0		
010	0		
011	0		
100	0		
101	0		
110	**1**	**10**	**Mem[10110]**
111	0		

Przykład

Adres	Adres bin.	Trafienie?	Blok cache
26	11 010	nie	010

Nr wiersza	Bit akt.	Znacznik	Dane
000	0		
001	0		
010	1	11	Mem[11010]
011	0		
100	0		
101	0		
110	1	10	Mem[10110]
111	0		

Przykład

Adres	Adres bin.	Trafienie	Blok cache
22	10 110	Tak	110
26	11 010	Tak	010

Nr wiersza	Bit akt.	Znacznik	Dane
000	0		
001	0		
010	1	11	Mem[11010]
011	0		
100	0		
101	0		
110	1	10	Mem[10110]
111	0		

Przykład

Adres	Adres bin.	Trafienie?	Blok cache
16	10 000	Nie	000
3	00 011	Nie	011
16	10 000	Tak	000

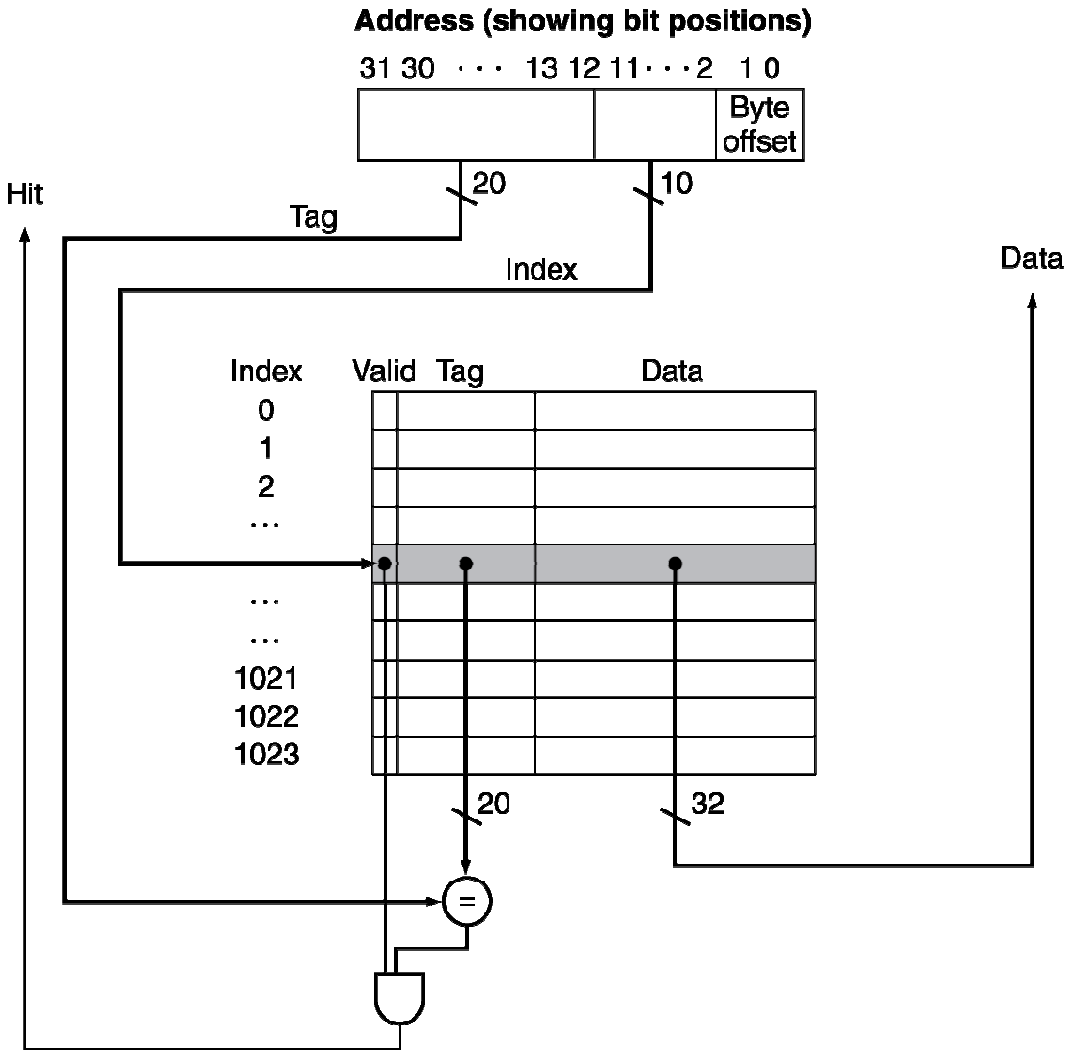
Nr wiersza	Bit Akt.	Znacznik	Dane
000	1	10	Mem[10000]
001	0		
010	1	11	Mem[11010]
011	1	00	Mem[00011]
100	0		
101	0		
110	1	10	Mem[10110]
111	0		

Przykład

Adres	Adres bin.	Trafienie?	Blok cache
18	10 010	nie	010

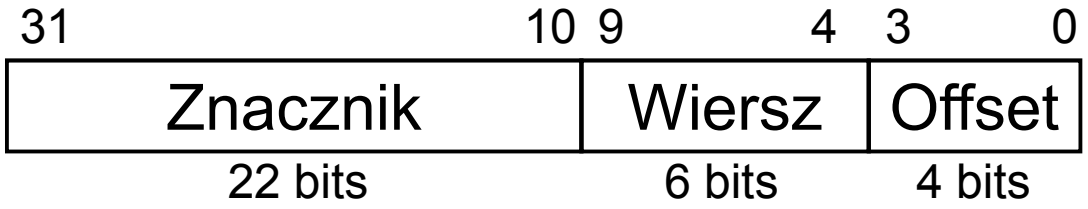
Wiersz cache	Bit akt.	Znacznik	Dane
000	1	10	Mem[10000]
001	1		
010	**1**	**10**	**Mem[10010]**
011	1	00	Mem[00011]
100	0		
101	0		
110	1	10	Mem[10110]
111	0		

Logiczny podział adresu



Przykład: wielosłowowe bloki

- 64 wiersze w RAM, 16 bajtów na blok
  - W którym wierszu cache przechowywana jest zawartość adresu 1200?
- Numer bloku w RAM = $\lfloor 1200/16 \rfloor = 75$
- Nr wiersza cache = 75 modulo 64 = 11



Rozmiar bloku

- Czy większe bloki zwiększają współczynnik trafień?
  - Niekoniecznie!
- W cache o ustalonym rozmiarze
  - Większe bloki $\Rightarrow$ mniej bloków
  - Większa rywalizacja o wiersze cache
- Większa ,,kara'' za chybienie (*miss penalty*)

Chybienia w cache

- Trafienie odczytu w cache: CPU pracuje standardowo
- Chybienie:
  - Wstrzymujemy potok
  - Przenosimy cały blok z RAM do cache (zazwyczaj wiele słów!)
  - Chybienie przy odczycie rozkazu:
    - Ponownie pobierz rozkaz (będzie trafienie!)
  - Chybienie przy odczycie danych:
    - Dokończ odczyt

Zapis ,,przez'' (write-through)

- Trafienie zapisu: wystarczyłoby zapisać tylko do cache
  - Ale cache i RAM są wtedy niespójne!
- Zapis ,,przez'': zapisujemy też do pamięci
- Zapisy trwają wtedy długo
  - np., bazowe CPI (# cykli na instrukcję) = 1, 10% rozkazów sw, zapis do pamięci 100 cykli procesora
    - efektywne CPI = 1 + 0.1×100 = 11
- Rozwiązanie: bufor zapisu
  - Przechowuje zapisywane dane
  - CPU kontynuuje pracę
    - Zatrzymujemy potok dopiero po przepełnieniu bufora

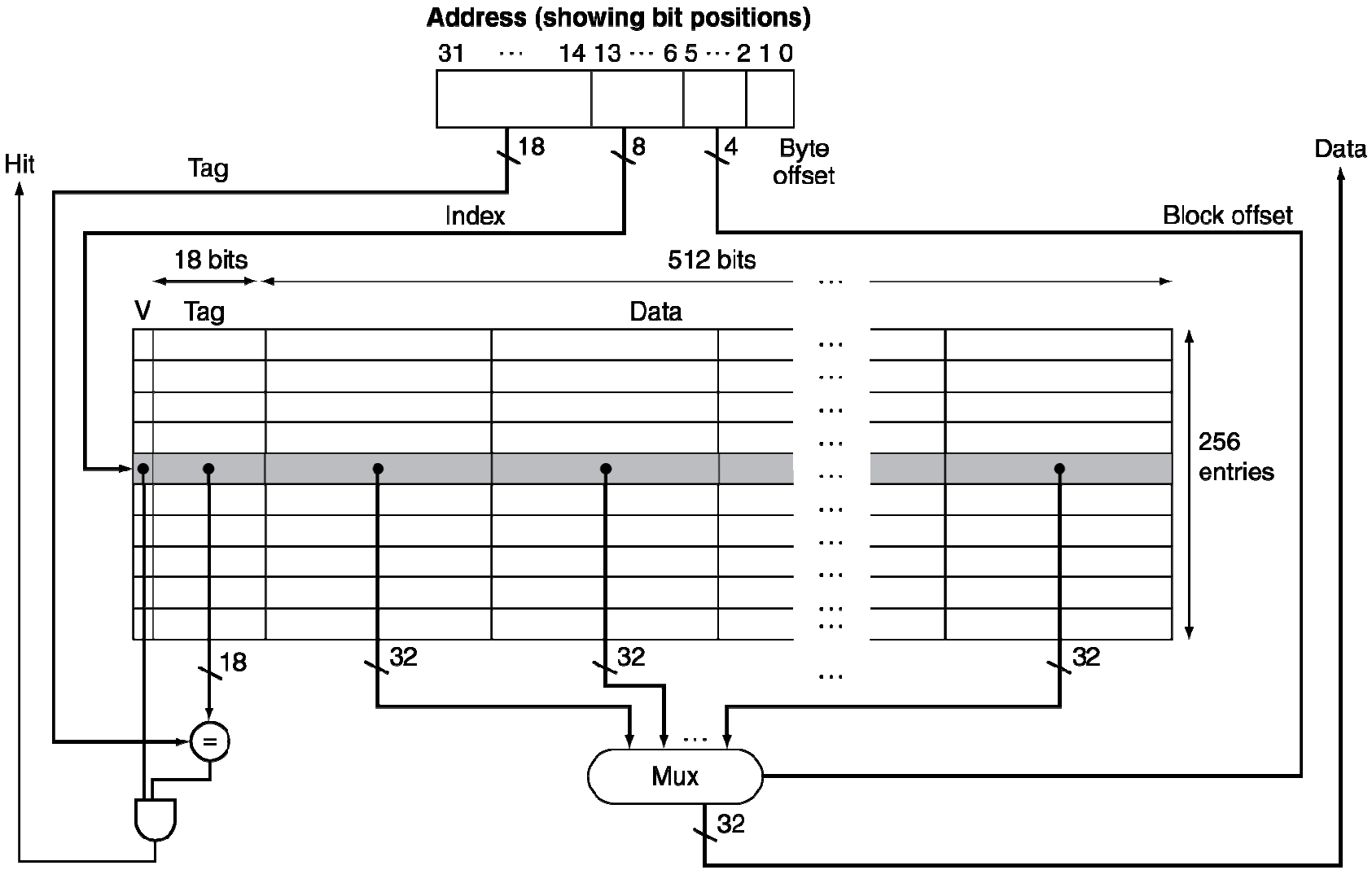
Zapis „po” (write-back)

- Zapisujemy tylko do cache
  - Dodatkowy bit (*dirty bit*) w wierszu cache wskazuje, czy blok został zmodyfikowany
- Przy wymianie zmodyfikowanego bloku
  - Zapisujemy go do RAM
  - Można również użyć bufora zapisu

Przykład: Intrinsity FastMATH

- Procesor MIPS do urządzeń wbudowanych
- Dzielony cache: I-cache (rozkazów) and D-cache (danych)
  - każdy 16KB: 256 wierszy po 16 słów
  - obsługuje write-through oraz write-back
- współczynnik chybień na SPEC2000
  - I-cache: 0.4%
  - D-cache: 11.4%
  - Średnia ważona: 3.2%

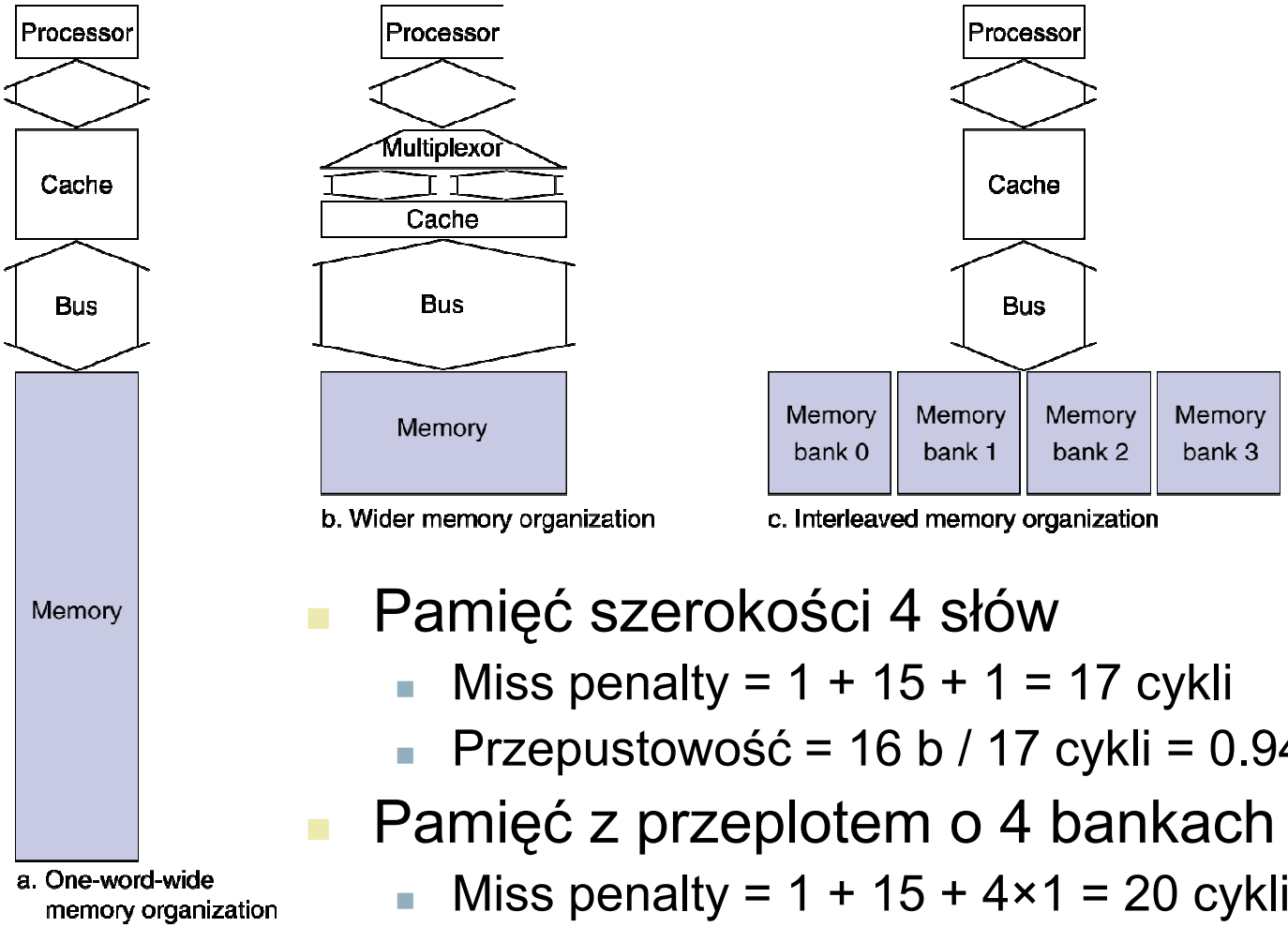
Przykład: Intrinsity FastMATH



Pamięć główna a cache

- Pamięć główna zbudowana jako DRAM
  - Ustalona szerokość (np. 1 słowo)
  - Połączona z CPU magistralą sterowaną zegarem
    - Częstotliwość zegara magistrali mniejsza niż procesora
- Przykładowy odczyt bloku
  - 1 cykl magistrali na przesłanie adresu
  - 15 cykli na dostęp do pamięci
  - 1 cykl na transfer danych
- Dla 4-słowowego bloku i pamięci szer. 1 słowa
  - Miss penalty = 1 + 4×15 + 4×1 = 65 cykli mag.
  - Przepustowość = 16 bajtów / 65 cykli = 0.25 B/cykl

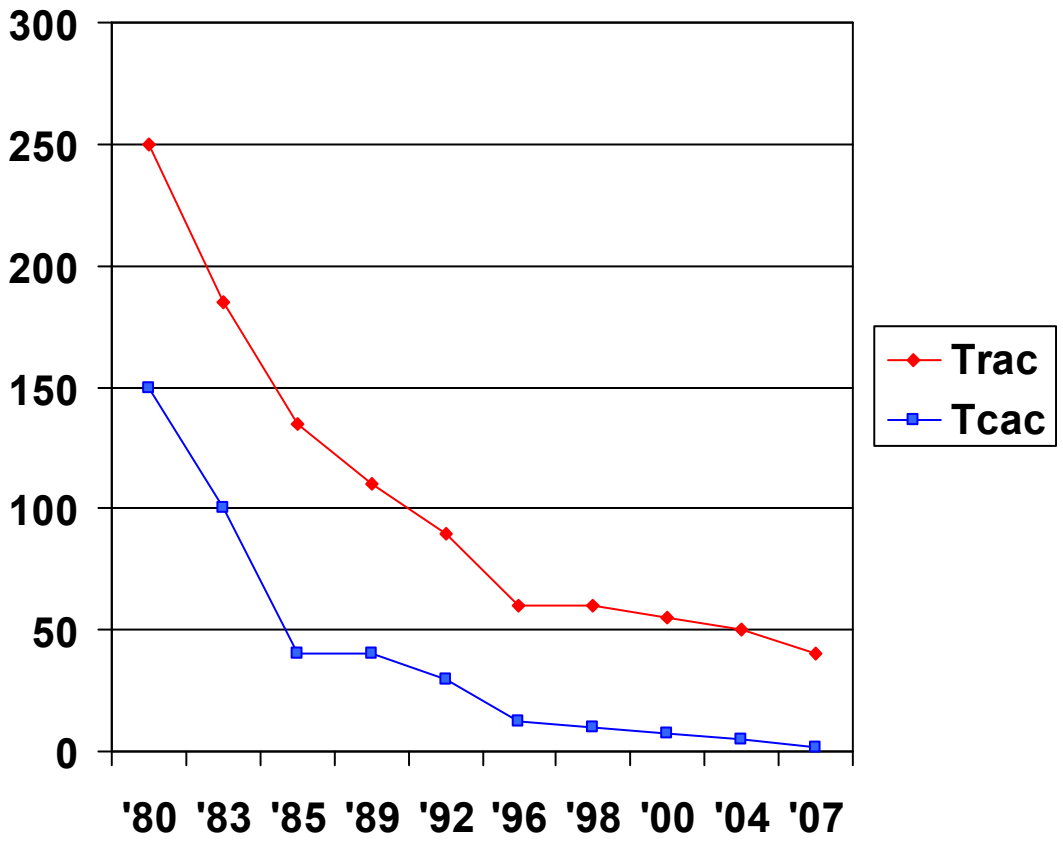
Zwiększenie przepustowości



- Pamięć szerokości 4 słów
  - Miss penalty = 1 + 15 + 1 = 17 cykli
  - Przepustowość = 16 b / 17 cykli = 0.94 B/cykl
- Pamięć z przeplotem o 4 bankach
  - Miss penalty = 1 + 15 + 4×1 = 20 cykli
  - Przpustowość = 16 b / 20 cykli = 0.8 B/cykl

Generacje DRAM

Rok	Pojemność chipa	$/GB
1980	64Kbit	$1500000
1983	256Kbit	$500000
1985	1Mbit	$200000
1989	4Mbit	$50000
1992	16Mbit	$15000
1996	64Mbit	$10000
1998	128Mbit	$4000
2000	256Mbit	$1000
2004	512Mbit	$250
2007	1Gbit	$50



Przykład obliczeń

- Dane
  - Współczynnik chybień w I-cache = 2%
  - Współczynnik chybień w D-cache = 4%
  - Miss penalty = 100 cykli procesora
  - bazowe CPI (idealny cache) = 2
  - Lw & sw stanowią 36% rozkazów
- Stracone cykle / rozkaz
  - I-cache: 0.02 × 100 = 2
  - D-cache: 0.36 × 0.04 × 100 = 1.44
- rzeczywiste CPI = 2 + 2 + 1.44 = 5.44

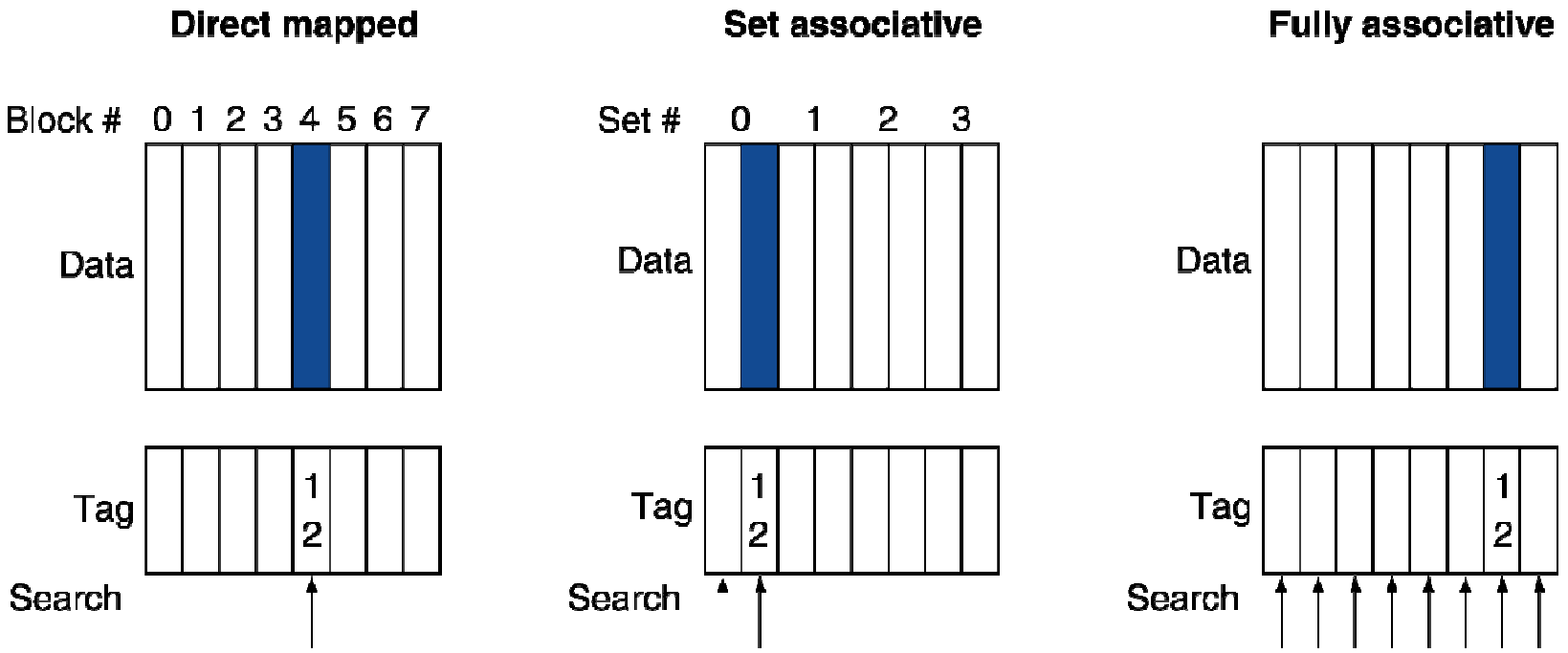
Przykład obliczeń – cd.

- Co się stanie jeśli poprawimy bazowe CPI do 1, ale nie poprawimy transferu z RAM?
- rzeczywiste CPI = 1 + 2 + 1.44 = 4.44
- Dwukrotne przyspieszenie procesora spowodowało zmniejszenie CPI tylko o około 1/5!

Mapowanie skojarzeniowe

- Pełne:
  - Blok może iść do dowolnego wiersza cache
  - Wszystkie tagi badane jednocześnie
  - Każdy wiersz wymaga komparatora
- Sekcyjno-skojarzeniowe (*n*-drożne)
  - Cache podzielony na *n*-wierszowe **sekcje**
  - Numer bloku RAM określa sekcję
    - (Nr bloku) modulo (# sekcji w cache)
  - Badamy jednocześnie tagi z jednej sekcji
  - Wystarczy *n* komparatorów

Mapowanie skojarzeniowe



Liczba wierszy w sekcji

- For a cache with 8 entries

One-way set associative
(direct mapped)

Block	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

Two-way set associative

Set	Tag	Data	Tag	Data
0				
1				
2				
3				

Four-way set associative

Set	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0								
1								

Eight-way set associative (fully associative)

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data

Przykład

- Porównamy 4-blokowe pamięci cache
  - Z mapowaniem bezpośrednim, 2-drożnym skojarzeniowym, pełnym skojarzeniowym
  - Nr żądanych bloków: 0, 8, 0, 6, 8
- Mapowanie bezpośrednie:

Block address	Cache index	Hit/miss	Cache content after access			
			0	1	2	3
0	0	miss	**Mem[0]**			
8	0	miss	**Mem[8]**			
0	0	miss	**Mem[0]**			
6	2	miss	Mem[0]		**Mem[6]**	
8	0	miss	**Mem[8]**		Mem[6]	

Przykład – cd.

- 2-drożne sekcyjno-skojarzeniowe

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	miss	**Mem[0]**			
8	0	miss	Mem[0]	**Mem[8]**		
0	0	hit	**Mem[0]**	Mem[8]		
6	0	miss	Mem[0]	**Mem[6]**		
8	0	miss	**Mem[8]**	Mem[6]		

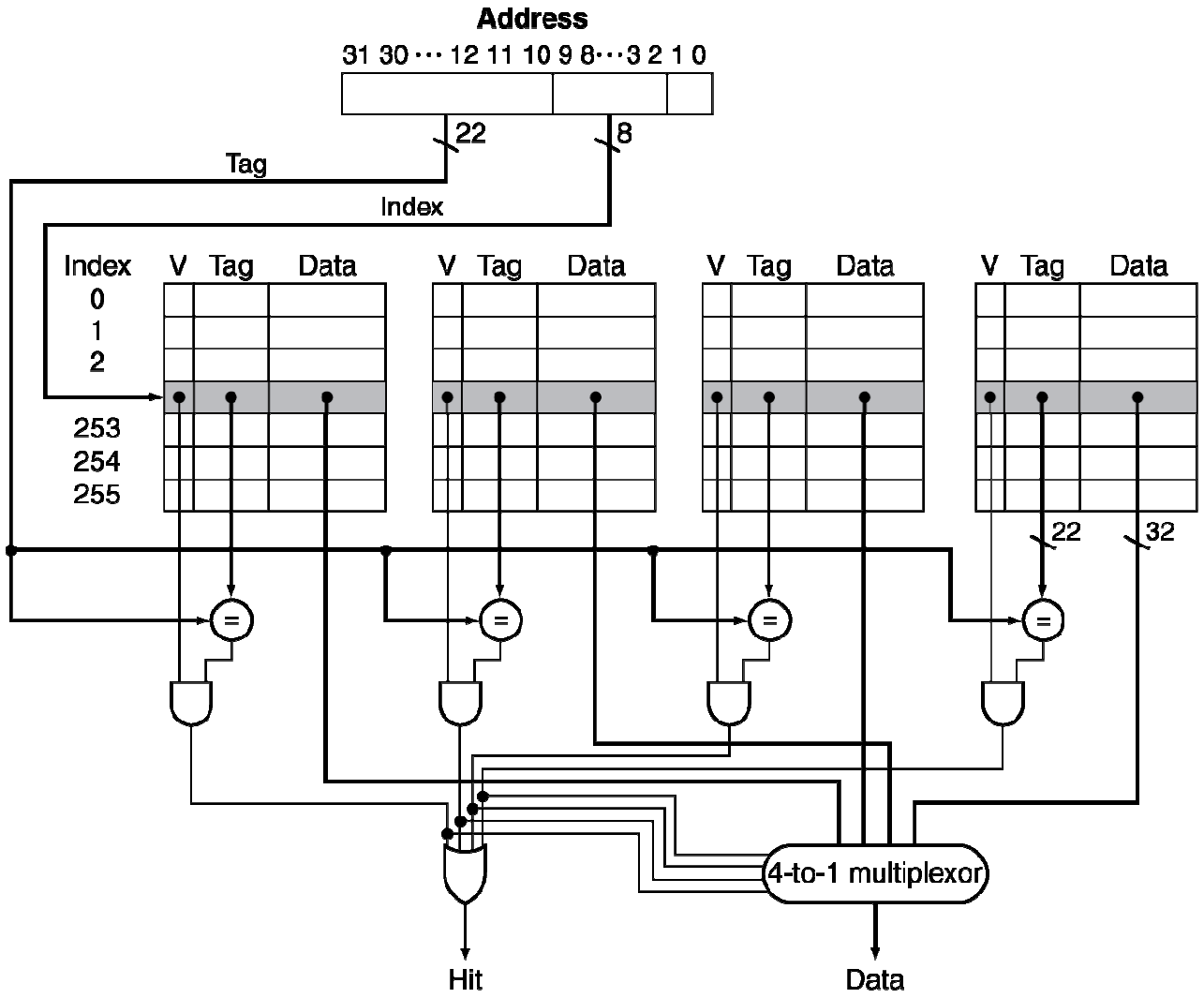
- Pełne skojarzeniowe

Block address		Hit/miss	Cache content after access			
0		miss	**Mem[0]**			
8		miss	Mem[0]	**Mem[8]**		
0		hit	**Mem[0]**	Mem[8]		
6		miss	Mem[0]	Mem[8]	**Mem[6]**	
8		hit	Mem[0]	**Mem[8]**	Mem[6]	

Ile wierszy w sekcji?

- Zwiększenie liczby wierszy w sekcji zmniejsza liczbę chybień
  - Ale niezbyt imponująco…
- Symulacja systemu z 64KB pamięcią D-cache, 16-słowowe bloki, SPEC2000
  - 1-drożna (mapowanie bezpośrednie): 10.3%
  - 2-drożna: 8.6%
  - 4-drożna: 8.3%
  - 8-drożna: 8.1%

Organizacja



Strategia wymiany bloków

- Mapowanie bezpośrednie: nie dotyczy
- Sekcyjno-skojarzeniowe
  - Zajmij wiersz z bitem akt. = 0 (jeśli taki jest)
  - W przeciwnym wypadku kilka strategii
- Najdawniej używany (LRU)
  - Proste dla 2-drożnej (jeden dodatkowy bit), do zrobienie dla 4-drożnej, bardzo kosztowne dla większych n
- FIFO, LFU, losowy (przy dużym n zaskakująco dobry), …

Cache wielopoziomowy

- Cache L1 dołączony do CPU
  - Mały, szybki
- Cache L2 obsługuje chybienia L1
  - Większy, wolniejszy, ale wciąż szybszy niż DRAM
- Niektóre systemy mają cache L3

Cache wielopoziomowy cd.

- Dane:
  - CPU CPI = 1, zegar: = 4GHz
  - Współczynnik chybień (na rozkaz) = 2%
  - Czas dostępu do DRAM = 100ns
- Cache jednopoziomowy:
  - Kara za chybienie = 100ns/0.25ns = 400 cykli
  - Efektywne CPI = 1 + 0.02 × 400 = 9

Przykład cd.

- Dodajemy cache L-2
  - Czas dostępu = 5ns
  - Współczynnik chybień = 0.5%
- Chybienie w L1, trafienie w L2
  - Kara za chybienie = 5ns/0.25ns = 20 cykli
- Chybienie w L-2
  - Dodatkowa kara = 500 cykli
- CPI = 1 + 0.02 × 20 + 0.005 × 400 = 3.4
- Poprawa wydajności = 9/3.4 = 2.6