System pamięci

Pamięć wirtualna

Pamięć wirtualna

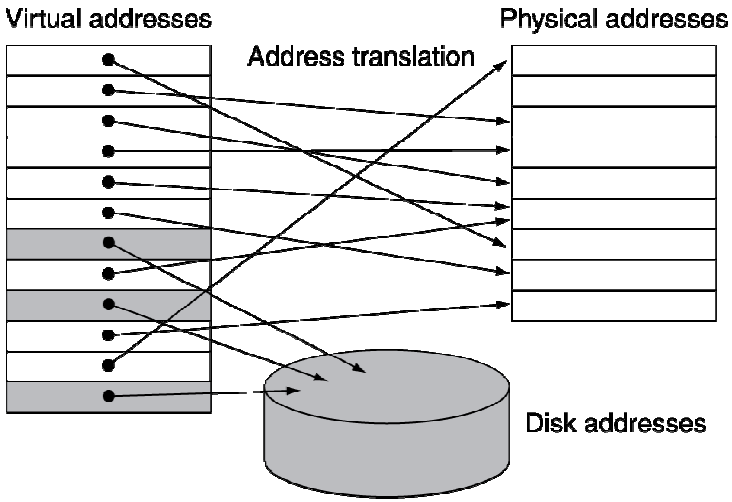
- Model pamięci cache+RAM nie jest jeszcze realistyczny
- W rzeczywistych systemach działa wiele programów jednocześnie
  - Każdy może używać tej samej przestrzeni adresowej
  - Programy mogą używać większej przestrzeni adresowej niż rzeczywisty RAM
- Rozwiązanie – pamięć wirtualna

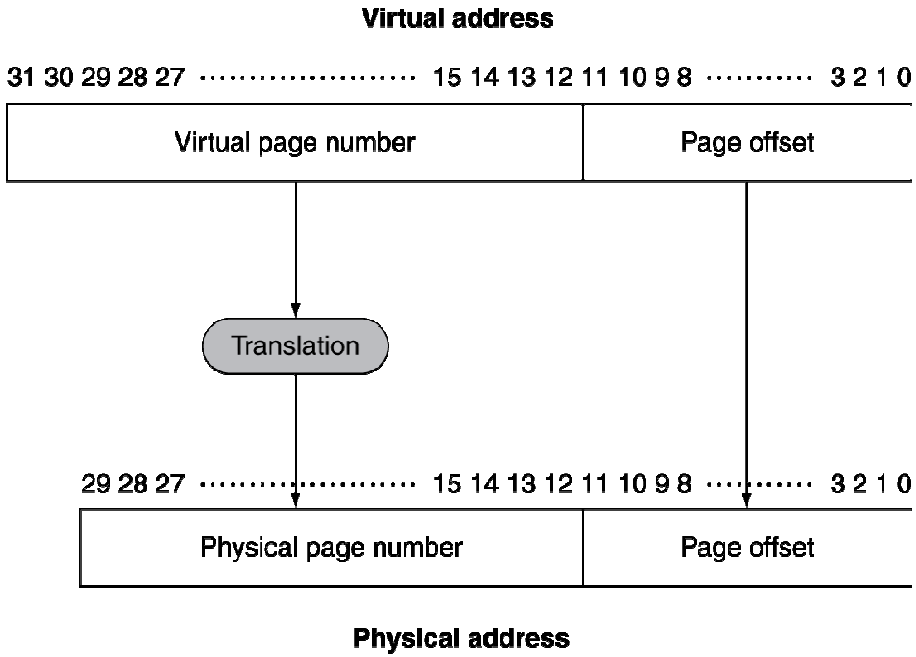
Pamięć wirtualna

- RAM jest używany jako ,,cache'' dla dysku
- Obsługa pamięci wirtualnej rozłożona pomiędzy sprzęt oraz system operacyjny
- Adresy występujące w skompilowanych programach to tzw. *adresy wirtualne*
- CPU i OS tłumaczą adesy wirtualne na fizyczne
  - ,,Blokiem'' jest twz. *strona*
  - Chybienie nazywamy *błędem strony*

Tłumaczenie adresu

- Stały rozmiar strony (np. 4K)





Kara za błąd strony

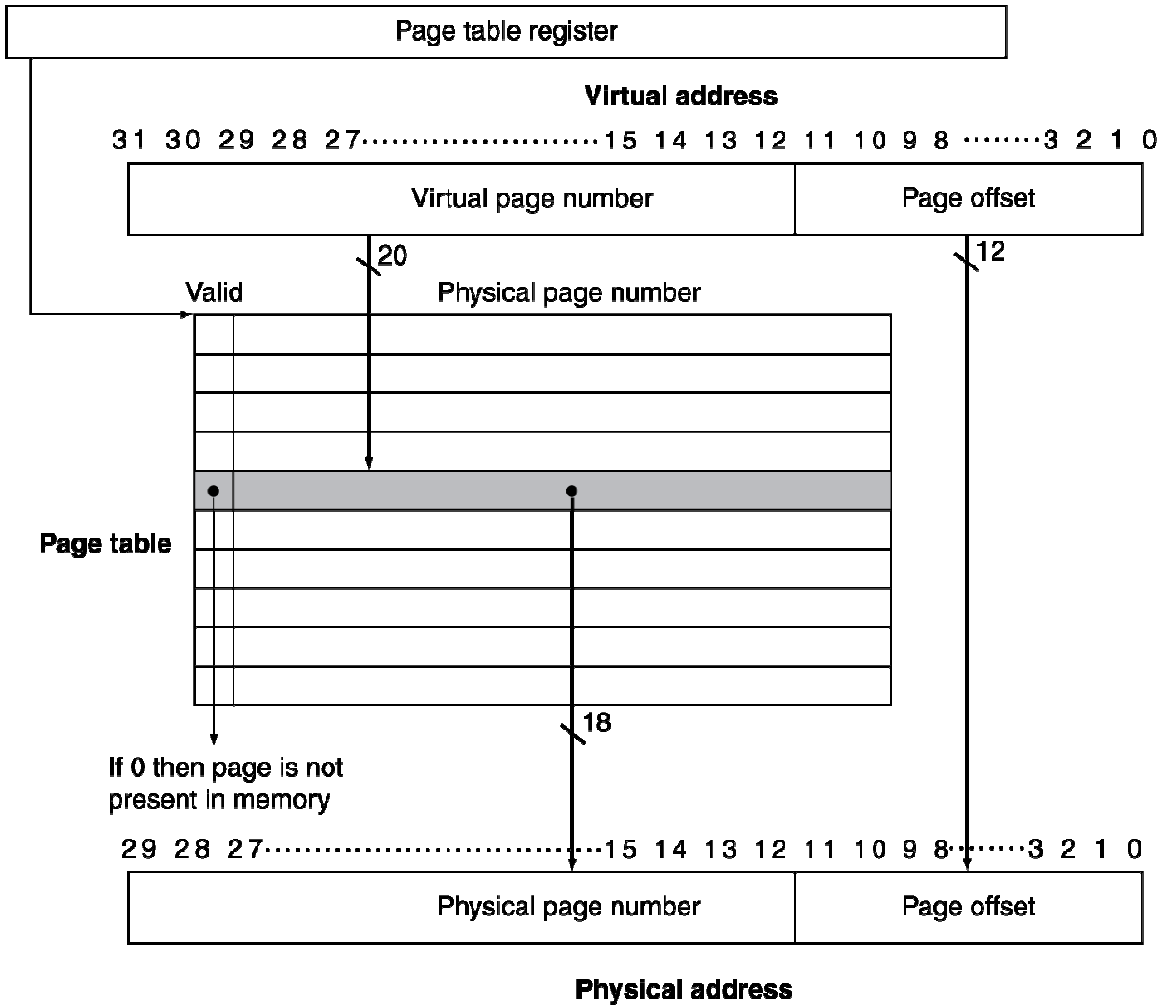
- W przypadku błędu strony, strona musi być pobrana z dysku
  - Zajmuje to miliony cykli
  - Odpowiedzialny za pobranie strony jest OS
- Nacisk na minimalizację liczby błędów stron
  - Pełna ,,skojarzeniowość''
  - Sprytne algorytmy wymiany stron

Tablice stron

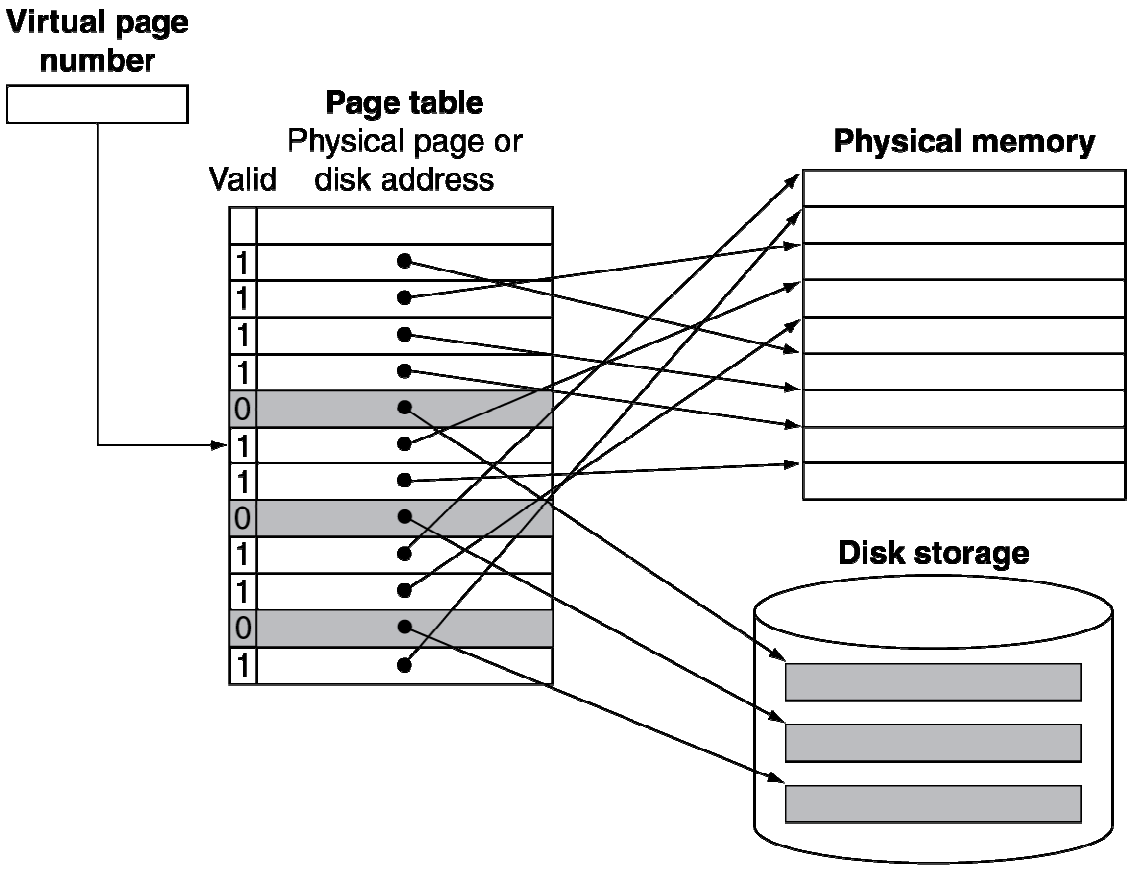
- Przechowują informacje o położeniu strony
  - Wpisy indeksowane wirtualnym numerem strony
  - CPU ma rejestr przechowujący adres tablicy stron w pamięci RAM
  - Każdy proces ma swoją tablicę stron
- Jeśli strona jest w pamięci
  - Tablica stron przechowuje fizyczny adres w RAM
  - Plus dodatkowe informacje (o odwołaniach do strony)
- Jeśli strony nie ma w pamięci
  - Tablica stron przechowuje adres fizyczny na dysku (w tzw. swapie).
- Tablice stron mogą być duże.

Proces tłumaczenia

- Jeszcze uproszczony!



Strony w pamięci i na dysku



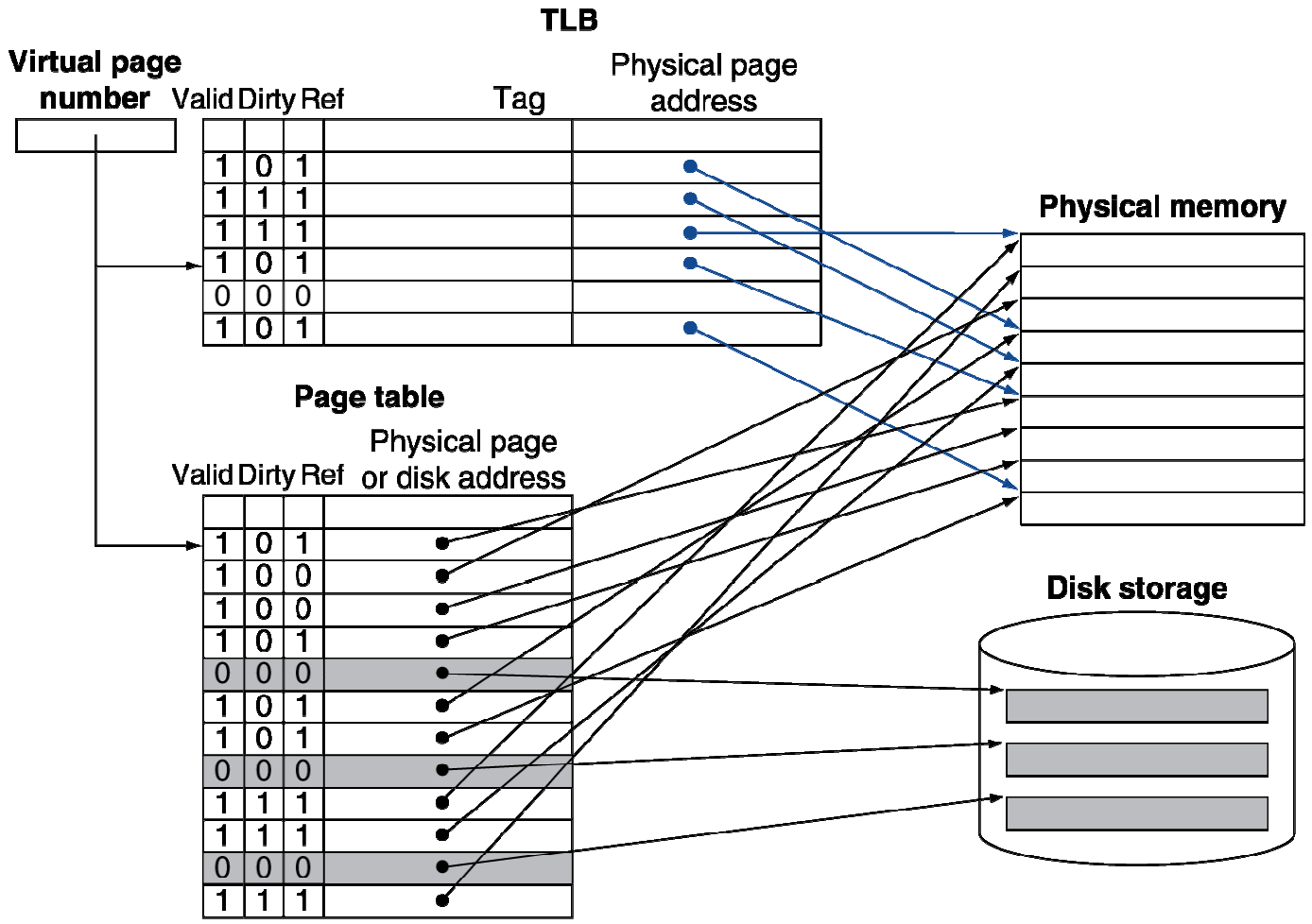
Strategia wymiany i zapisu

- Za wymianę odpowiedzialny OS; najczęściej używa (przybliżenia) LRU
- Zapisy na dysk zajmują miliony cykli
  - Stosowany zapis-po (zapis-przez niepraktyczny)
  - W tablic stron jest ,,bit modyfikacji'' (dirty bit)

Szybkie tłumaczenie: TLB

- Dotychczas zakładaliśmy, że cała tablica stron jest w RAM
  - Każdy odczyt wymaga sięgnięcia do RAM!
  - Dopiero wtedy mamy rzeczywisty adres.
- Przy dostępie do stron używamy tzw. szybkiego bufora translacji (TLB, table lookaside buffer)
  - TLB to rodzaj cachu na płytce CPU
  - Typowo mieści 16–512 wpisów tabeli stron,
  - 0.5–1 cykla na trafienie, 10–100 cykli na chybienie, współczynnik chybień bardzo mały: 0.01%–1%

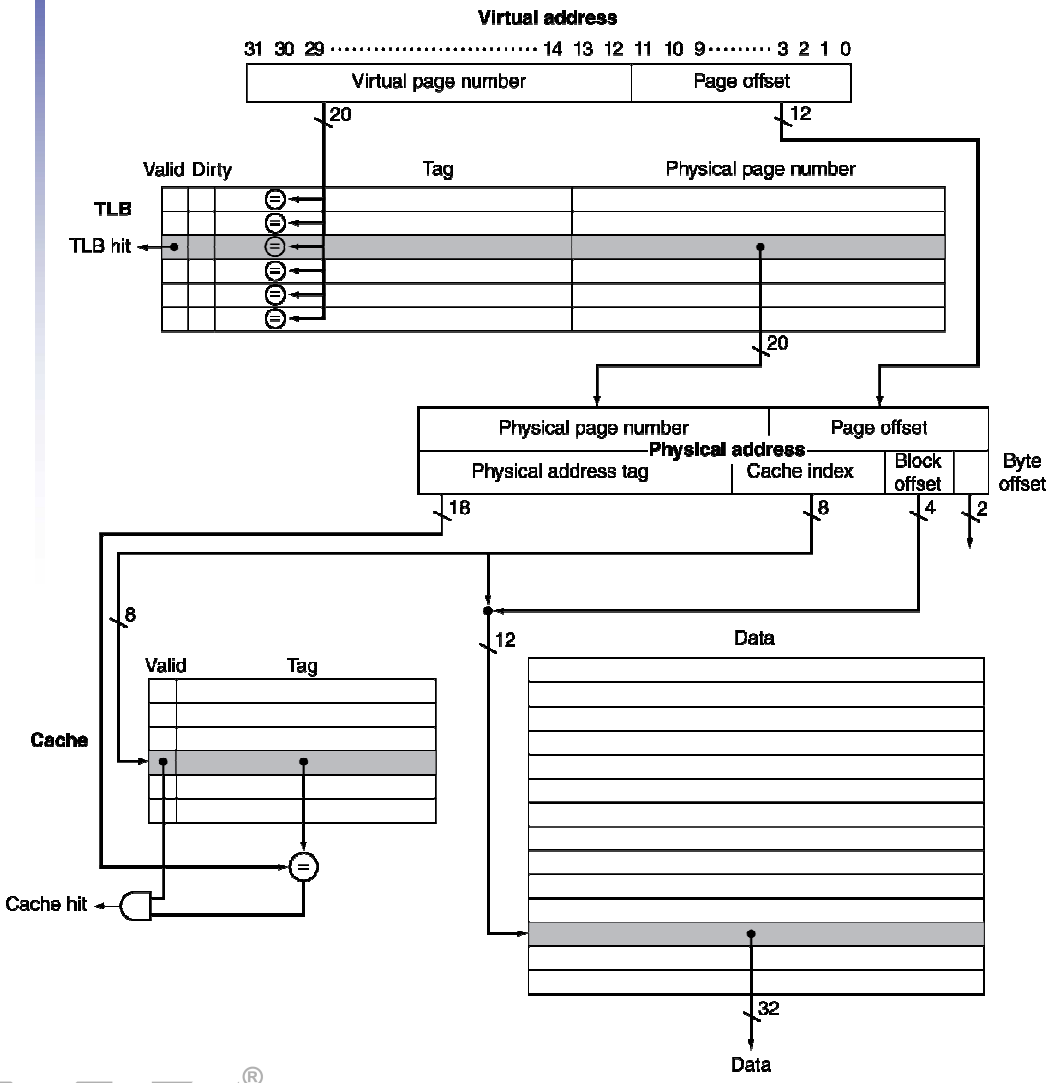
Fast Translation Using a TLB



Chybienia w TLB

- Jeśli strona jest w pamięci
  - Ładujemy wpis z tablicy stron i ponawiamy
  - Obsługiwane często przez hardware
    - Złożone w przypadku skomplikowanych struktur tablicy stron
  - Może być też obsługiwane przez software
    - Zgłaszany jest tzw. wyjątek, który powoduje załadowanie odpowiedniego programu (handler)
- Jeśli wystąpił błąd strony (strona na dysku)
  - OS ściaga stronę z dysku i modyfikuje tab. str.
  - W tym czasie CPU wykonuje inny program

TLB i Cache



- Odwołanie do adresu
  - CPU podaje adres wirtualny do TLB
  - Trafienie: mamy adres fizyczny i dalej postępujemy tak jak tydzień temu: sięgamy do cache i jeśli trzeba to do RAM

Hierarchia pamięci (zasady)

- Podobne zasady obowiązują na różnych poziomach hierarchii
  - Idea ,,cachowania''
- Zagadnienia
  - Gdzie umieścić blok?
  - Jak znaleźć blok?
  - Który blok wymienic?
  - Jaką strategię zapisu przyjąć?

Porównanie parametrów

Feature	Typical values for L1 caches	Typical values for L2 caches	Typical values for paged memory	Typical values for a TLB
Total size in blocks	250–2000	15,000–50,000	16,000–250,000	40–1024
Total size in kilobytes	16–64	500–4000	1,000,000–1,000,000,000	0.25–16
Block size in bytes	16–64	64–128	4000–64,000	4–32
Miss penalty in clocks	10–25	100–1000	10,000,000–100,000,000	10–1000
Miss rates (global for L2)	2%–5%	0.1%–2%	0.00001%–0.0001%	0.01%–2%

Źródła chybień

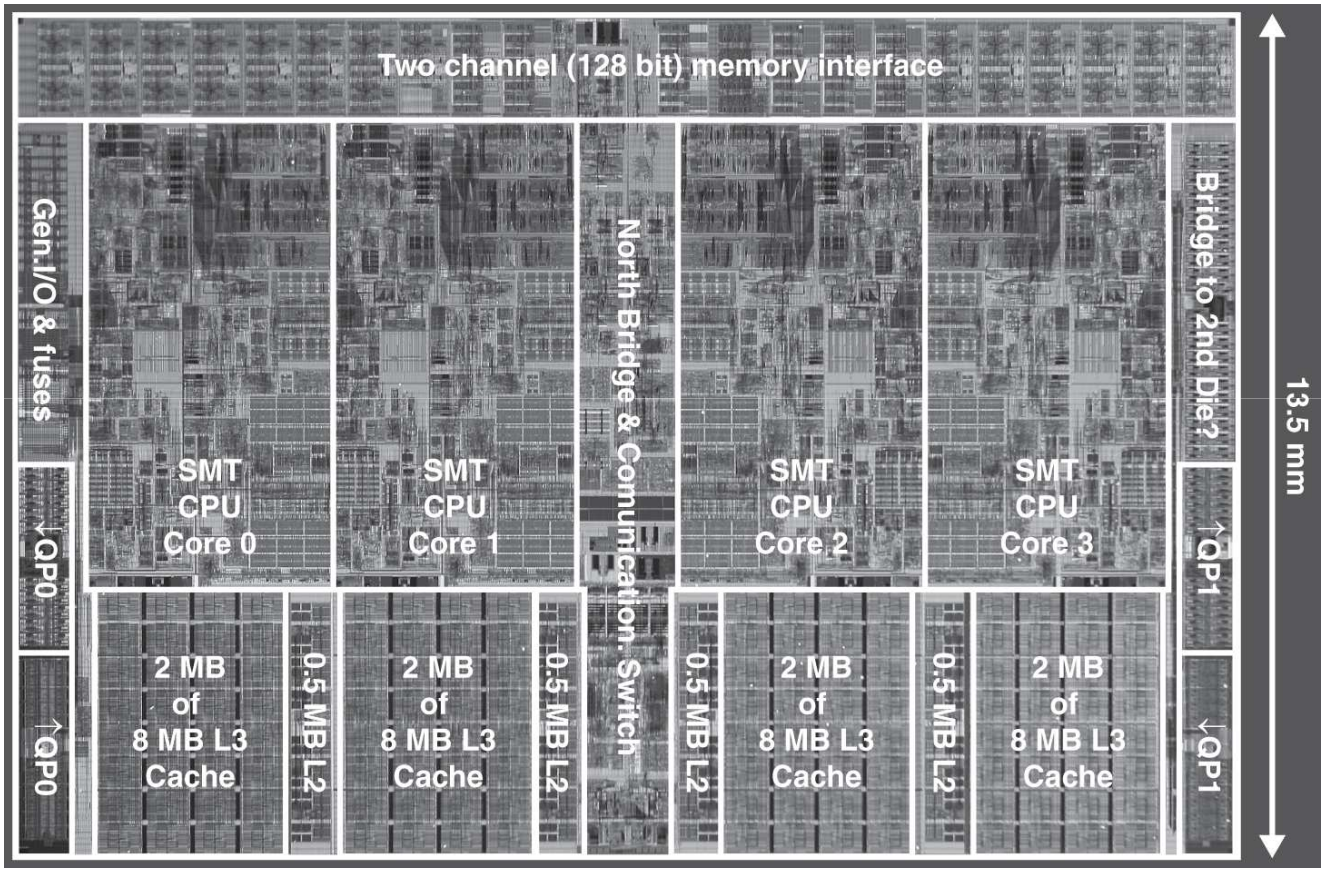
- Chybienia nieuniknione (compulsory misses)
  - Pierwsze odwołania do bloku
- Chybienia z braku miejsca (capacity misses)
  - Cache ma skończony, niewielki rozmiar
- Konflikt miejsca (conflict misses)
  - W cache bez pełnej skojarzeniowości
  - Rywalizacja bloków o mijsce w zbiorze

Kompromisy

Zmiana	Działanie na współczynnik chybień	Efekt negatywny
Zwiększenie cache	Maleje liczba chybień typu capacity	Wzrasta czas dostępu
Wzrot skojarzeniowośći	Malej liczba chybień typu conflict	Wzrasta czas dostępu
Zwiększenie rozmiaru bloku	Maleje liczba chybień typu compulsory	Wzrasta kara za chybienie. Dla dużych bloków może wzrosnąć całkowita liczba chybień.

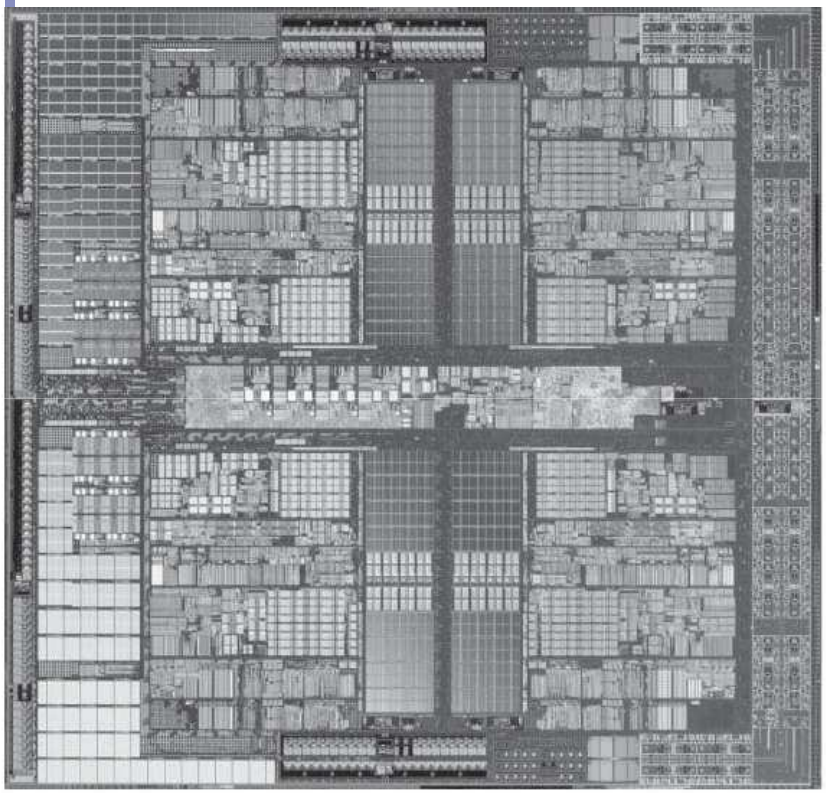
Przykłady – Intel Nehalem

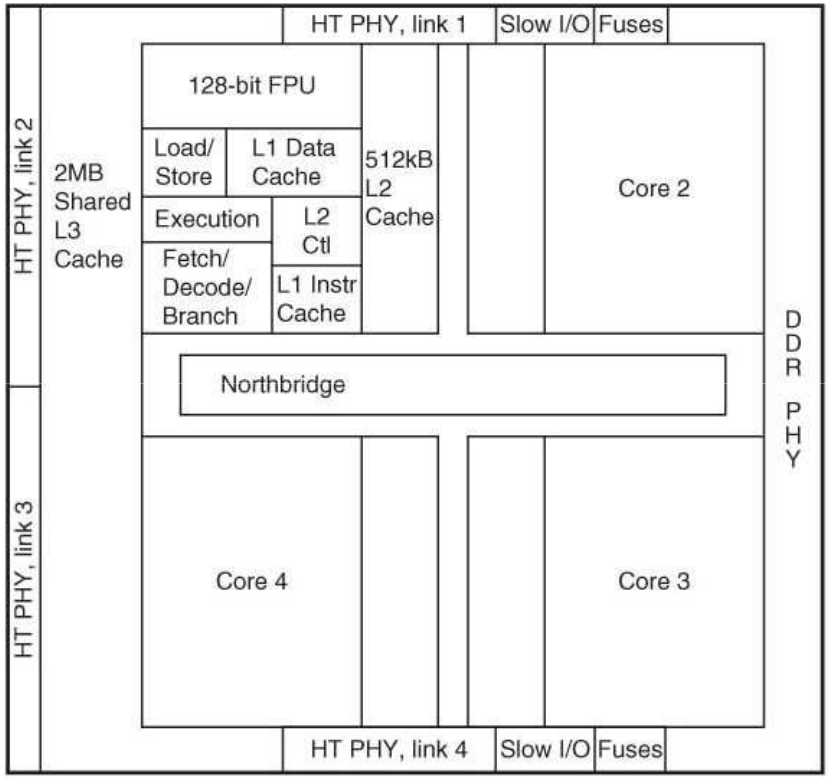
Procesor o 4 rdzeniach



Każdy rdzeń: 32KB L1 I-cache, 32KB L1 D-cache, 512KB L2 cache

AMD Opteron X4 (Barcelona)





Każdy rdzeń: 64KB L1 I-cache, 64KB L1 D-cache, 512KB L2 cache

2-poziomowy bufor TLB

	Intel Nehalem	AMD Opteron X4
Adres wirt.	48 bitów	48 bitów
Adres fiz.	44 bitów	48 bitów
Rozm. strony	4KB, 2/4MB	4KB, 2/4MB
L1 TLB (każdy rdzeń)	L1 I-TLB: 128 wpisów L1 D-TLB: 64 wpisy 4-drożna, LRU	L1 I-TLB: 48 wpisów L1 D-TLB: 48 wpisów Pełna skojarzeniowość, LRU
L2 TLB (każdy rdzeń)	L2 TLB: 512 wpisów 4-drożna, LRU	L2 I-TLB: 512 wpisów L2 D-TLB: 512 wpisów obie 4-drożne, round-robin LRU
Chyb. TLB	Obsłga sprzętowa	Obsługa sprzętowa

3-poziomowy cache

	Intel Nehalem	AMD Opteron X4
L1 cache (każdy rdzeń)	L1 I-cache: 32KB, 64-bajtowe bloki, 4-drożna, approx LRU, hit time n/a; L1 D-cache: 32KB, 64-bajtowe bloki, 8-drożna, approx LRU, zapis ,,po'', hit time n/a	L1 I-cache: 32KB, 64-bajtowe bloki, 2-drożna, LRU, hit time 3 cykle; L1 D-cache: 32KB, 64-bajtowe bloki, 2-drożna, LRU, zapis ,,po'', hit time 9 cykli
L2 cache (każdy rdzeń)	256KB, 64-bajtowe bloki, 8-drożna, approx LRU, zapis ,,po'', hit time n/a	512KB, 64-bajtowe bloki, 16-drożna, approx LRU, zapis ,,po'', hit time n/a
L3 cache (dzielona pomiędzy rdzeniami)	8MB, 64-bajtowe bloki, 16-drożna, wymiana n/a, zapis ,,po'', hit time n/a	2MB, 64-bajtowe bloki, 32-drożna, wymieniany blok używany przez najmniej rdzeni; zapis ,,po'', hit time: 32 cykle

AMD OPTERON X4

Tabela przedstawia liczbę chybień w cache oraz współczynnik CPI na benchmarku SPEC2006 dla procesora AMD OPTERON X4 2356 (Barcelona). Następny slajd rozszyfrowuje skróty z kolumny ,,Name'' (grupy programów SPEC2006)

Name	CPI	L1 D cache misses/1000 instr	L2 D cache misses/1000 instr	DRAM accesses/1000 instr
perl	0.75	3.5	1.1	1.3
bzip2	0.85	11.0	5.8	2.5
gcc	1.72	24.3	13.4	14.8
mcf	10.00	106.8	88.0	88.5
go	1.09	4.5	1.4	1.7
hmmer	0.80	4.4	2.5	0.6
sjeng	0.96	1.9	0.6	0.8
libquantum	1.61	33.0	33.1	47.7
h264avc	0.80	8.8	1.6	0.2
omnetpp	2.94	30.9	27.7	29.8
astar	1.79	16.3	9.2	8.2
xalancbmk	2.70	38.0	15.8	11.4
Median	1.35	13.6	7.5	5.4

AMD OPTERON X4

Description	Name	Instruction Count $\times 10^9$	CPI	Clock cycle time (seconds $\times 10^9$)	Execution Time (seconds)	Reference Time (seconds)	SPEC Ratio
Interpreted string processing	perl	2,118	0.75	0.4	637	9,770	15.3
Block-sorting compression	bzip2	2,389	0.85	0.4	817	9,650	11.8
GNU C compiler	gcc	1,050	1.72	0.4	724	8,050	11.1
Combinatorial optimization	mcf	336	10.00	0.4	1,345	9,120	6.8
Go game (AI)	go	1,658	1.09	0.4	721	10,490	14.6
Search gene sequence	hmmer	2,783	0.80	0.4	890	9,330	10.5
Chess game (AI)	sjeng	2,176	0.96	0.4	837	12,100	14.5
Quantum computer simulation	libquantum	1,623	1.61	0.4	1,047	20,720	19.8
Video compression	h264avc	3,102	0.80	0.4	993	22,130	22.3
Discrete event simulation library	omnetpp	587	2.94	0.4	690	6,250	9.1
Games/path finding	astar	1,082	1.79	0.4	773	7,020	9.1
XML parsing	xalancbmk	1,058	2.70	0.4	1,143	6,900	6.0
Geometric Mean							11.7