
Uniwersytet
w Siedlcach
Architektura
Systemów
Komputerowych
dr Marcin
Stępniak

Pamięć

Pamięć: pisanie & czytanie

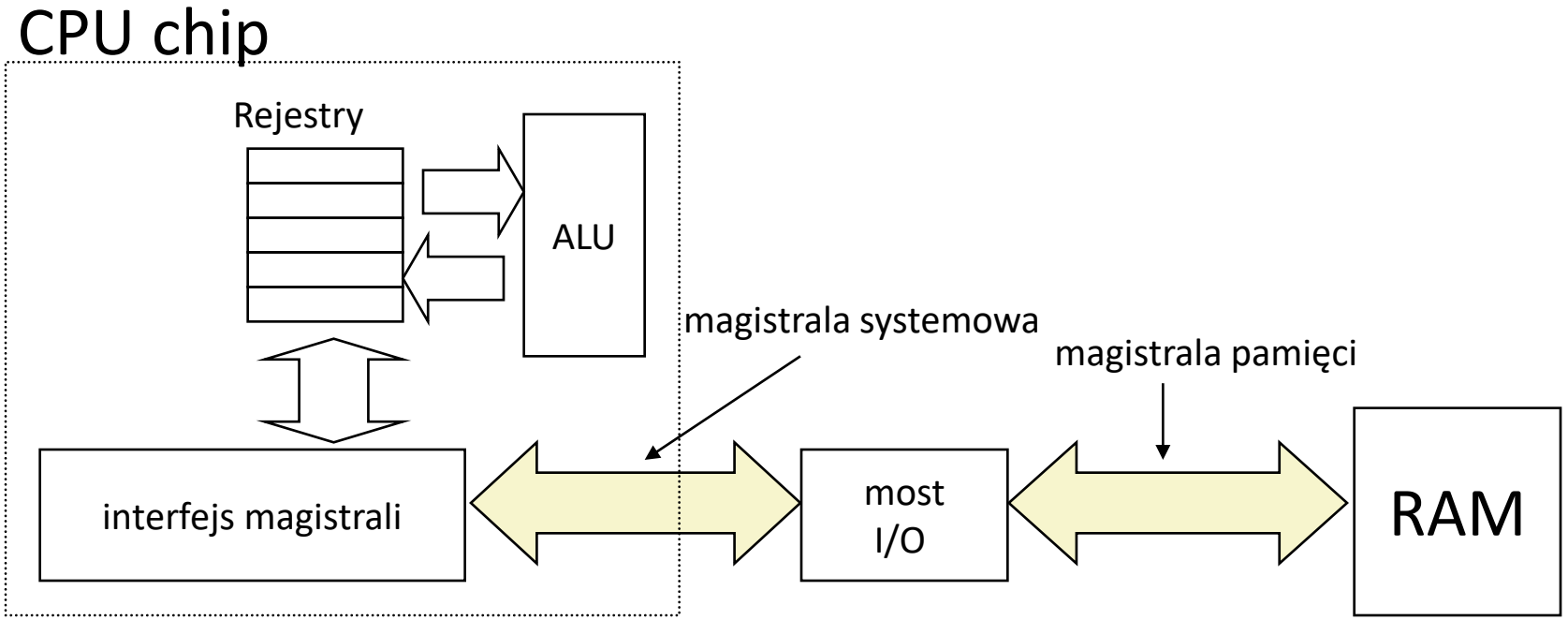
- Zapisz
  - Transfer danych z CPU do pamięci RAM
    `mov [CL],AL`
  - Operacja zapisz to co jest w rejestrze **AL,** w komórce pamięci RAM o adresie, który jest w rejestrze **CL**

- Czytaj
  - Transfer danych z pamięci RAM do CPU
    `mov AL, [CL]`
  - Operacja pobierz zawartość z komórki pamieci RAM o adresie umieszczonym w rejestrze **CL**, i umieść ją (tę zawartość) w rejestrze **AL**

From 5th lecture

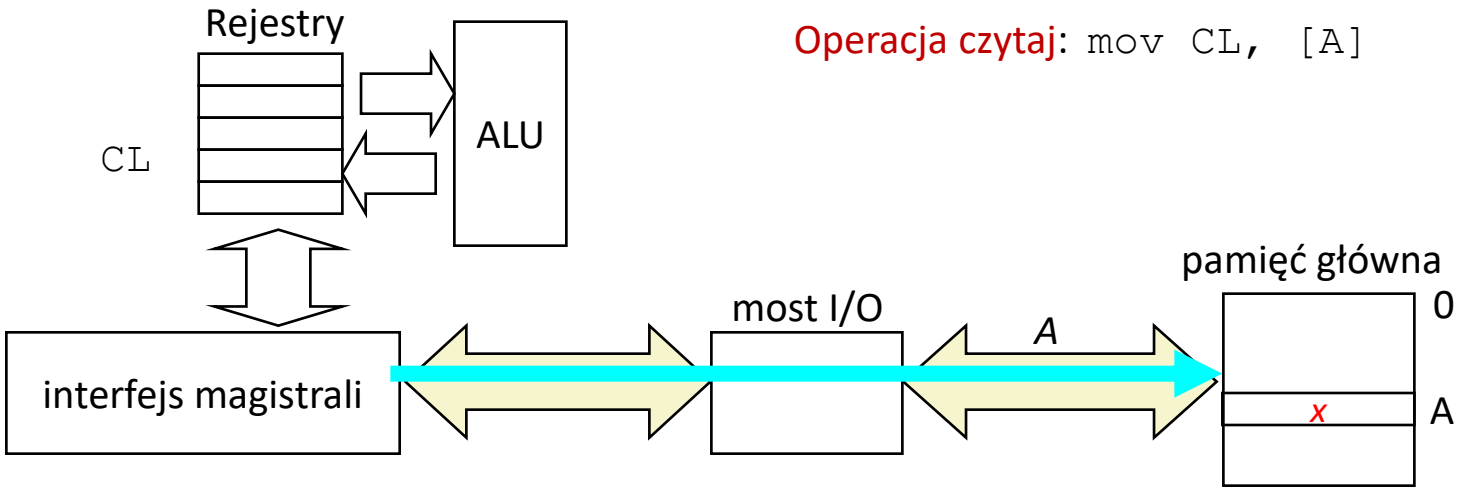
Klasyczna struktura magistrali (ang. bus) łącząca procesor i pamięć

- Magistrala to zestaw równoległych przewodów, które przenoszą adresy, dane i sygnały sterujące.
- Magistrale są zwykle wspólne dla wielu urządzeń.



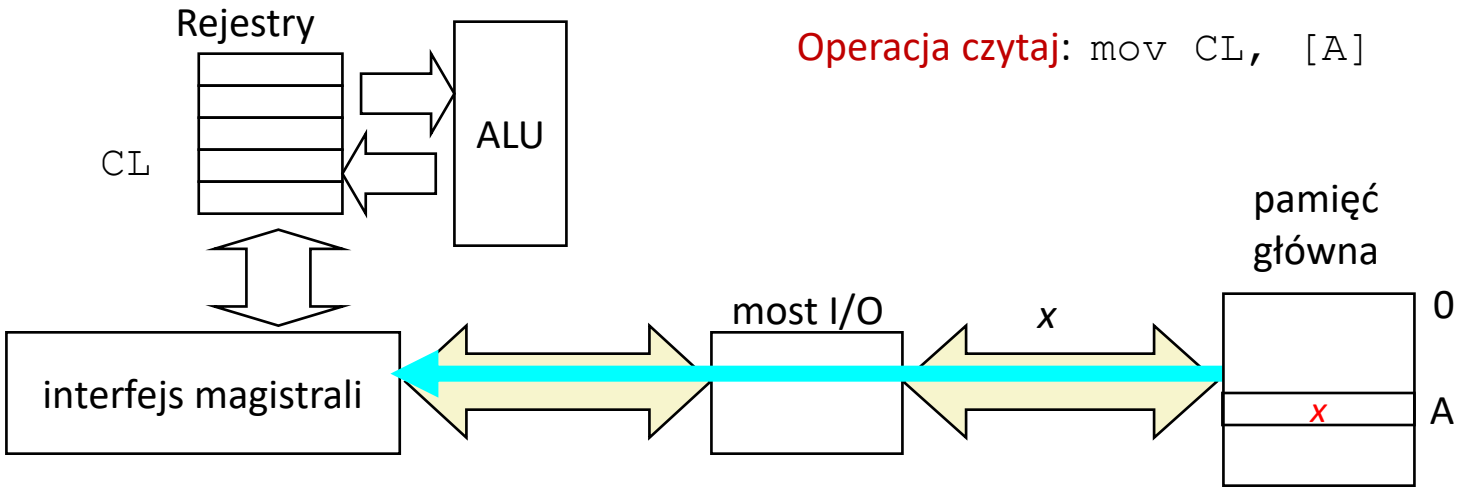
Czytanie z RAM (1)

- CPU wstawia adres A do magistrali.



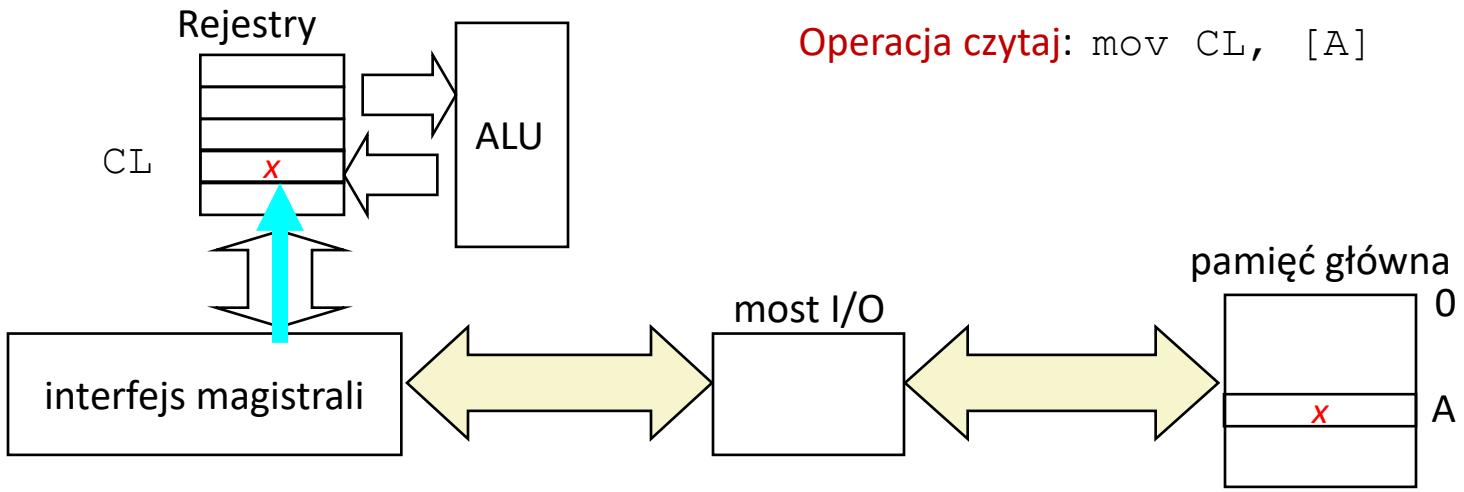
Czytanie z RAM (2)

- Pamięć główna odczytuje adres A z magistrali pamięci, sięga do komórki pod tym adresem, czyta x i umieszcza je na magistrali danych



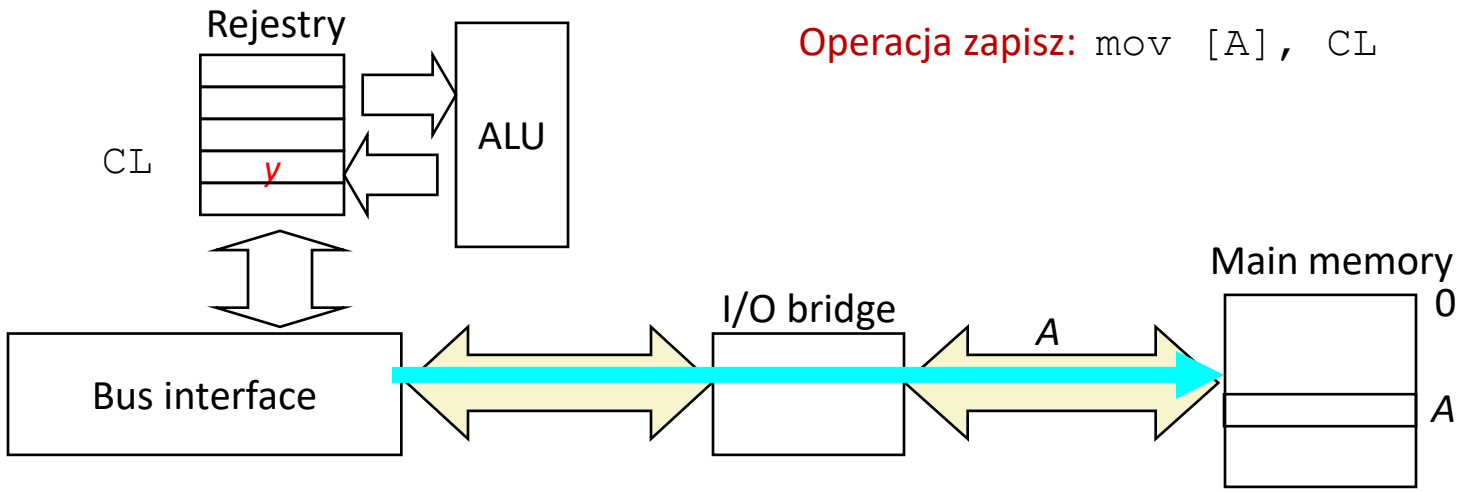
Czytanie z RAM (3)

- CPU czyta x z magistrali danych i kopiuje do rejestru CL



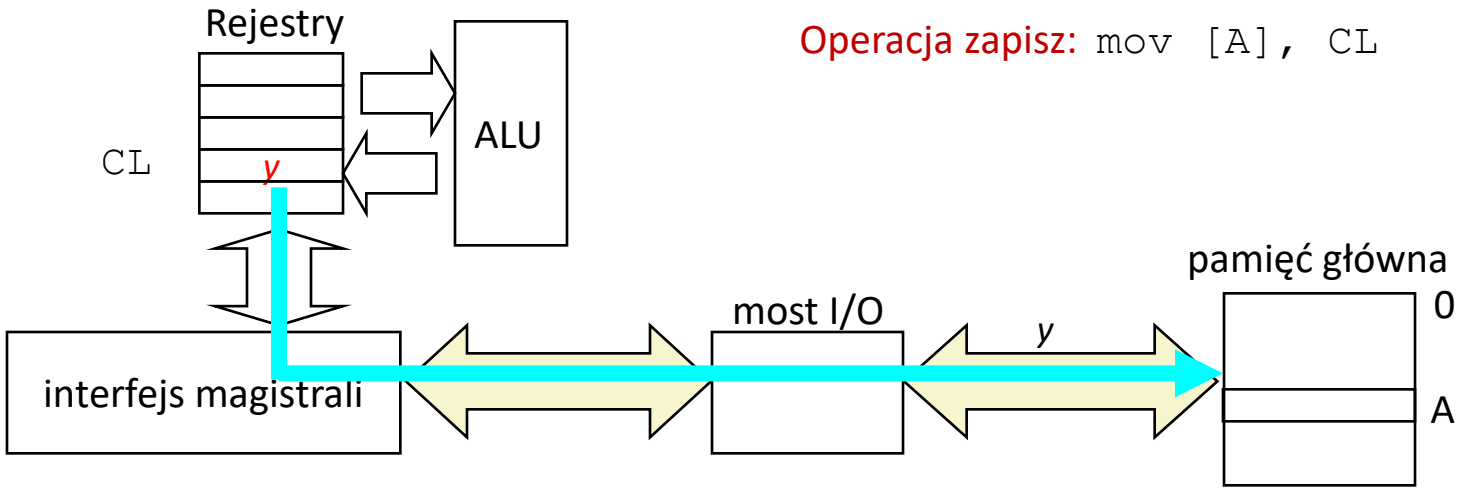
Wpisywanie do RAM (1)

- CPU wstawia adres A na magistralę. RAM czyta adres i czeka na dane do zapisania pod tym adresem



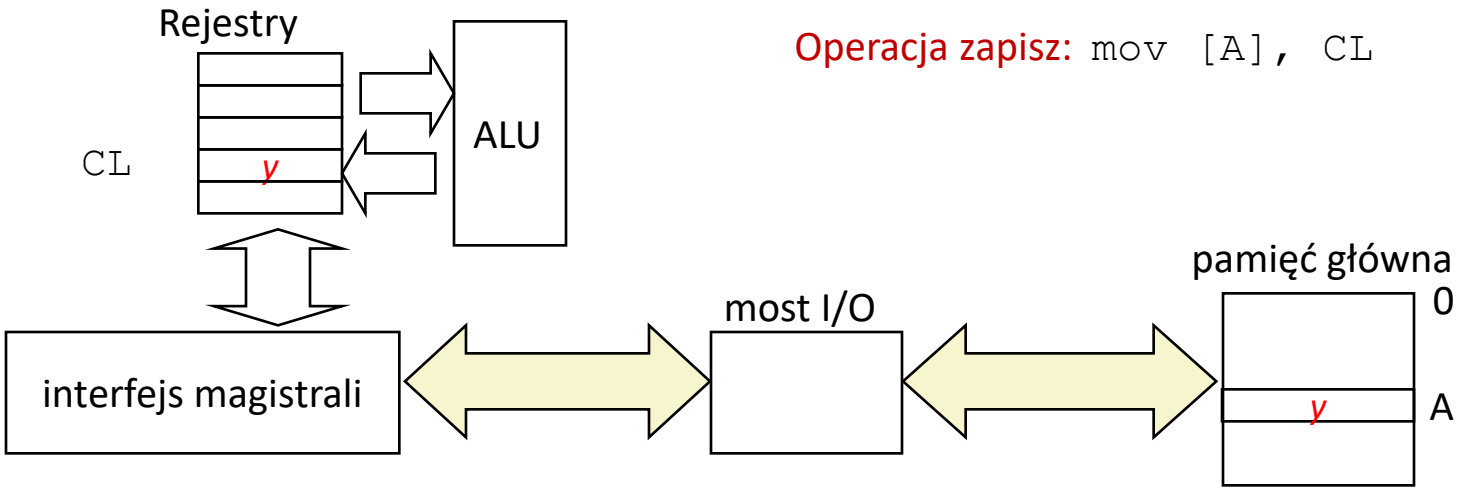
Wpisywanie do RAM(2)

- CPU wstawia dane z rejestru CL na magistralę



Wpisywanie do RAM (3)

- RAM czyta y z magistrali i zapisuje do komórki o adresie A

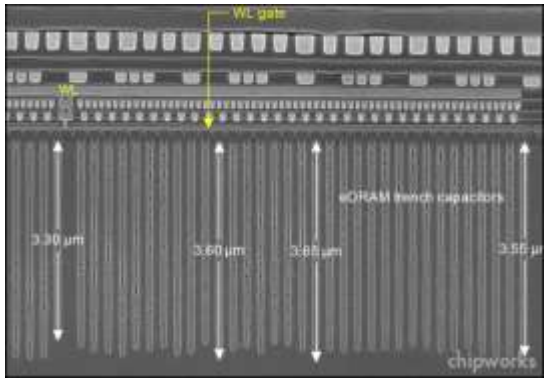


Random-Access Memory (RAM)

- Pamięć RAM jest realizowana jako osobny układ scalony lub jako część procesora
- Podstawową jednostką pamięci jest zwykle komórka (jeden bit na komórkę). Razem tworzą pamięć RAM
- RAM występuje w dwóch odmianach:
  - SRAM (statyczna pamięć RAM)
  - DRAM (dynamiczna pamięć RAM)

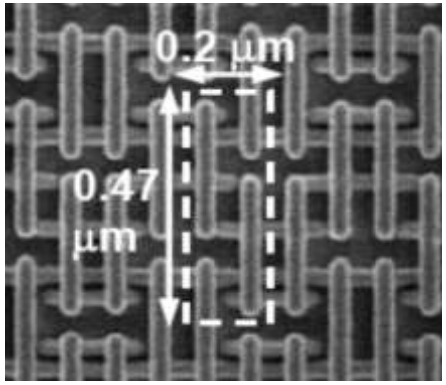
RAM

- DRAM



- 1 tranzystor + 1 kondensator / bit
- Stan musi być cykliczne odświeżany

- SRAM



- 6 tranzystorów / bit
- Utrzymuje stale stan

SRAM vs DRAM podsumowanie

	Tranzystory na bit	czas dostępu	odświeżanie	EDC	Koszt	Zastosowanie
SRAM	6 lub 8	1x	Nie	może	100x	pamięć Cache
DRAM	1	10x	Tak	Tak	1x	pamięć główna, bufory

EDC: Error detection and correction (wykrywanie i korekta błędu)

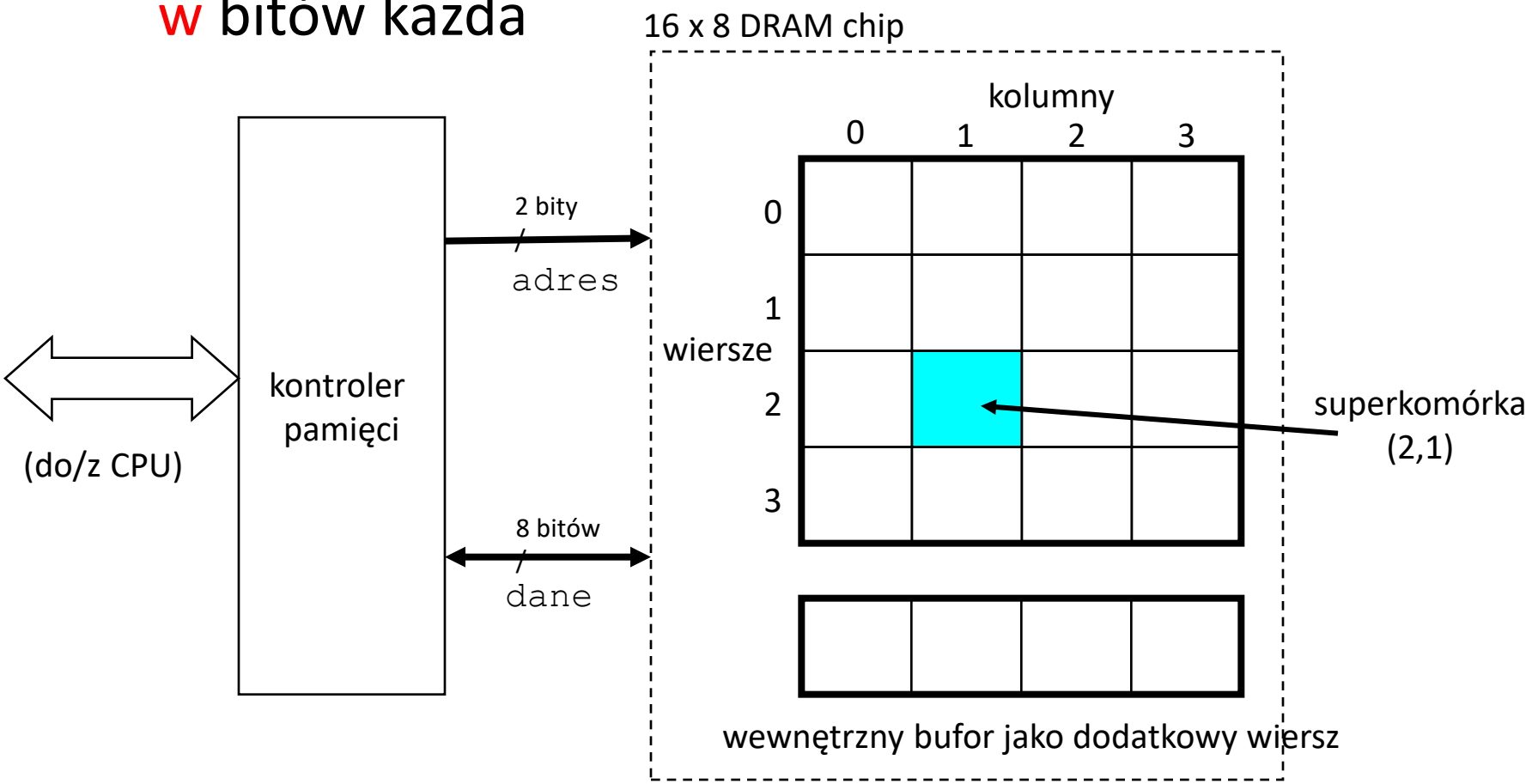
- Trendy rozwoju:
- SRAM ewoluje wraz z technologią półprzewodników
- Skalowanie pamięci DRAM jest ograniczone potrzebą minimalnej pojemności kondensatorów

ulepszone DRAM

- Działanie komórki DRAM nie zmieniło się od czasu jej wynalezienia i komercjalizacji przez Intel w 1970 roku.
- Rdzenie DRAM z lepszą logiką interfejsu i szybszym We / Wy:
  - Synchroniczna pamięć DRAM (SDRAM) wykorzystuje konwencjonalny sygnał zegarowy zamiast sterowania asynchronicznego
  - Synchroniczna pamięć DRAM o podwójnej szybkości przesyłania danych (DDR SDRAM)
    - Zegary wysyłają dwa bity na cykl . Różne typy rozróżniane według rozmiaru małego bufora pobierania wstępnego: DDR (2 bity), DDR2 (4 bity), DDR3 (8 bitów), DDR4 (16 bitów)
    - Do 2010 roku standard dla większości systemów serwerowych i stacjonarnych
    - Intel Core i7 obsługuje DDR3 i DDR4 SDRAM

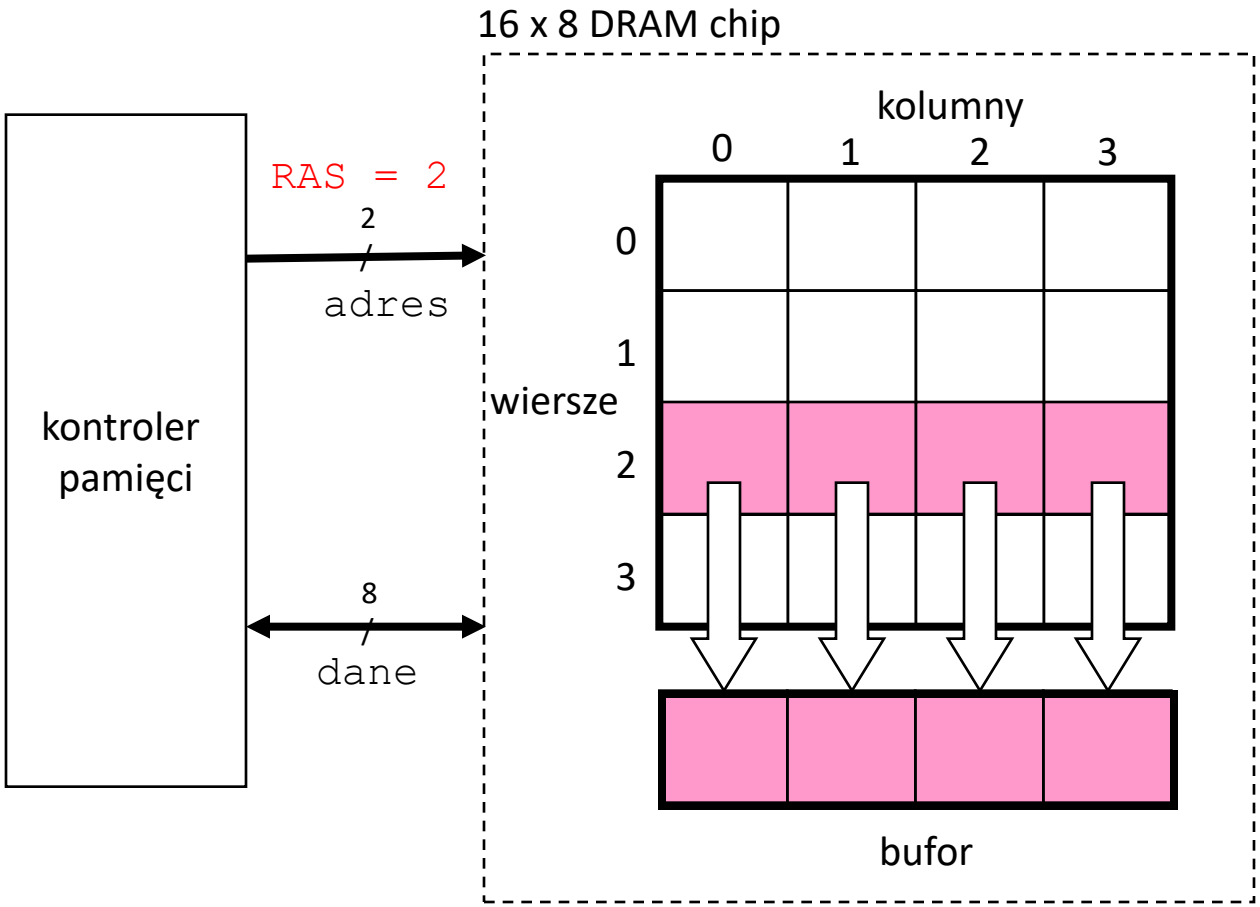
Klasyczna struktura DRAM

- *d* x *w* DRAM:
  - *d* · *w* bitów. W *d* superkomórkach (supercells) po w bitów każda



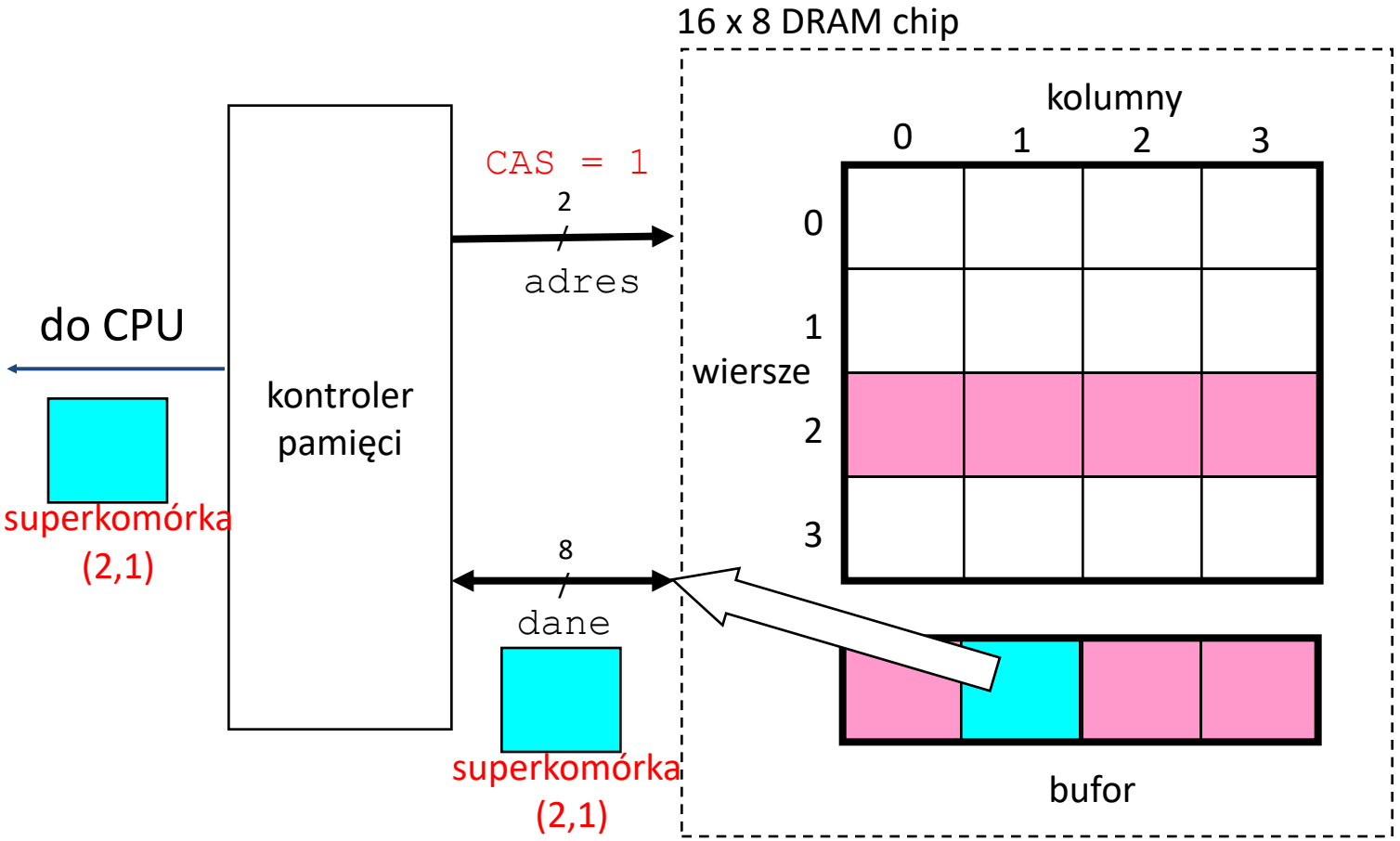
Czytanie: DRAM Supercell (2,1)

Krok 1: Row access strobe (RAS) wybiera wiersz numer 2, który jest nastepnie kopiowany do bufora

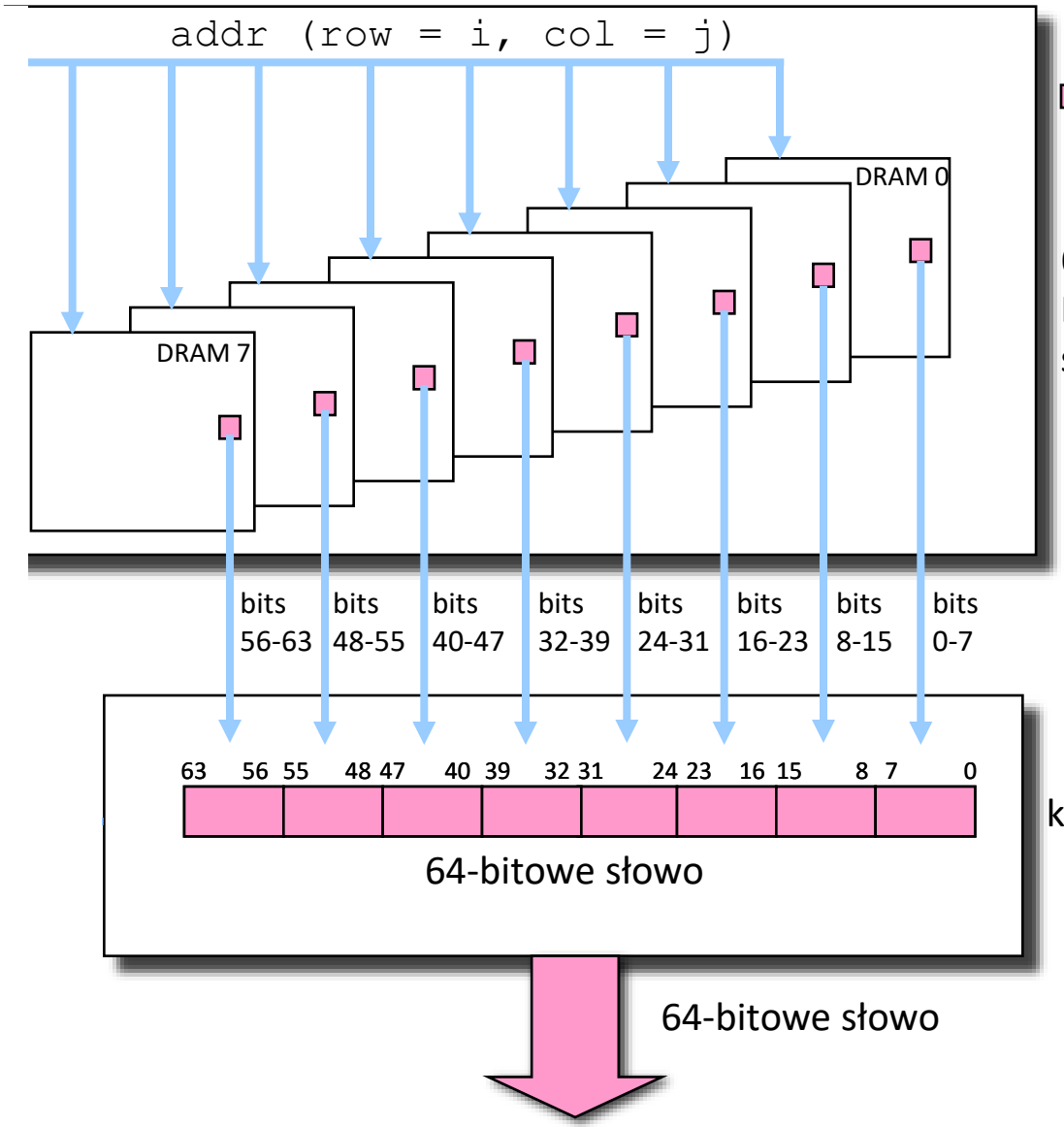


Czytanie: DRAM Supercell (2,1)

- Krok 2: Column access strobe (CAS) wybiera kolumnę numer 1 z bufora, która nastepnie jest kopiowana i przesyłana do kontrolera pamięci (Memory controller) a potem do CPU
- Krok 3: Dane z bufora odświerzają zawartość wiersza numer 2



Moduły pamięci



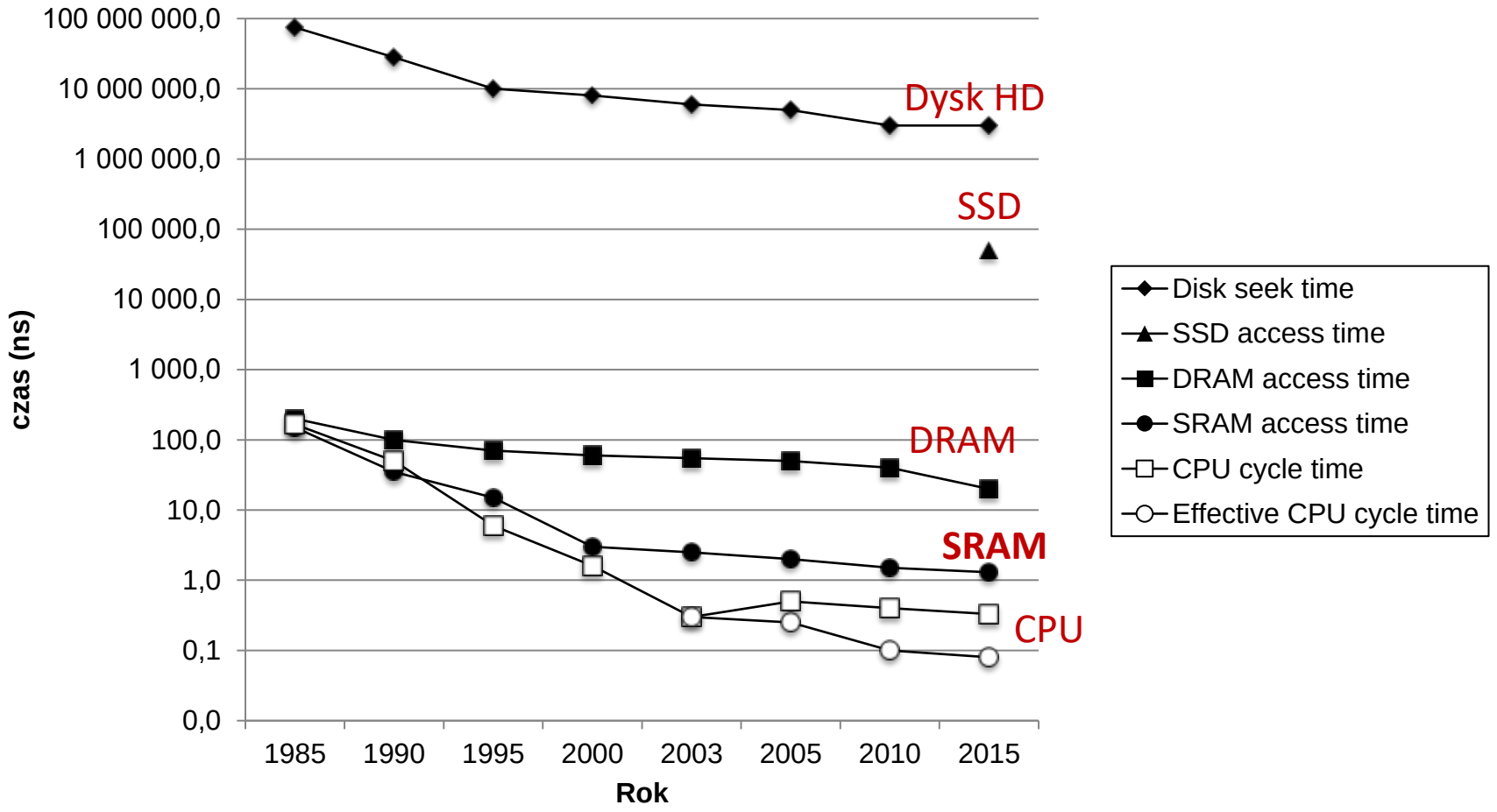
: superkomórka(i,j)

64 MB
Moduł pamięci składający się z ośmiu 8Mx8 DRAMs

kontroler pamięci

Luka szybkości pomiędzy CPU a Pamięcią

DRAM, SRAM, SSD, dysk HD oraz CPU

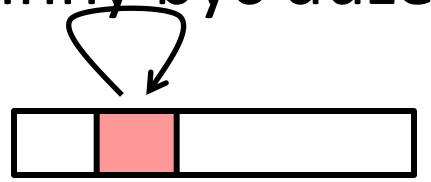


Jak tę lukę minimalizować!

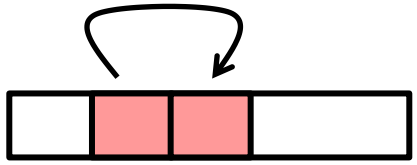
- Można próbować na poziomie programowania
- Kod jest zapisywany w pamięci
- Jeśli ten kod ma własność lokalności

Lokalność

- Zasada Lokalności: Aplikacja (w trakcie wykonywania instrukcji) używa adresów danych i innych instrukcji, które są blisko zapisane w pamięci. Jeśli są skoki, to nie powinny byc duże



- Czasowa (temporalna) lokalność:
  - Ostatnio używane odnośniki są często używane w kontynuacji wykonania



- Przestrzenna lokalność:
  - Adresy danych i instrukcje zapisane (w pamięci) blisko siebie są często używane w bliskich chwilach czasu

Przykłady lokalności w kodzie


```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

- Referencje do danych — Lokalnie, przestrzennie czy temporalnie?
  - do elementów tablicy jeden po drugim — przestrzennie
  - do zmiennej **`sum`** po każdej iteracji — temporalnie
- Referencje do instrukcji
  - w kolejności w kodzie — przestrzennie
  - cykle w pętli — temporalnie

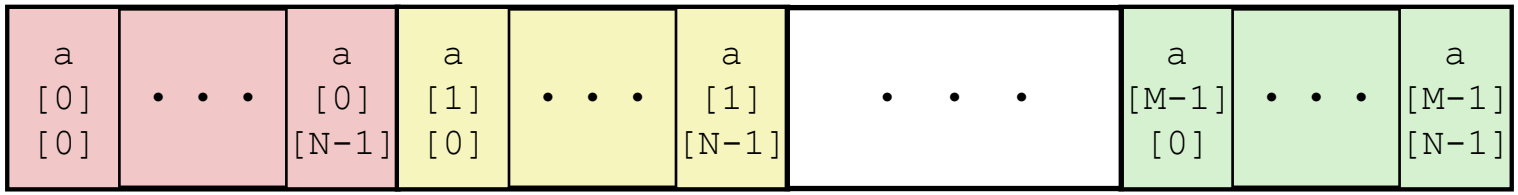
Jakościowa ocena lokalności kodu

- Fakt: profesjonalny programista potrafi ocenić jakościową lokalność patrząc na kod
- Pytanie: czy funkcja niżej na dobrą lokalność z względu na tablicę `a`?

Odpowiedź: TAK

```
int sum_array_rows(int a[M][N])
{
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```



Przykład

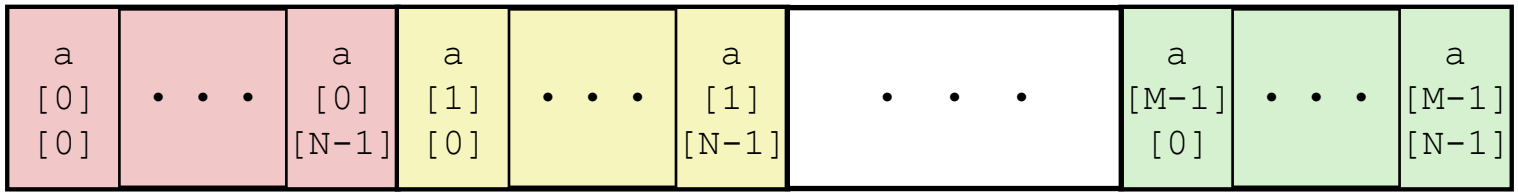
- Pytanie: a czy poniższa funkcja też posiada dobrą lokalność ze względu na tablicę?

```
int sum_array_cols(int a[M][N])
{
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];
    return sum;
}
```

Odpowiedź: Nie, chyba że…

M jest małe

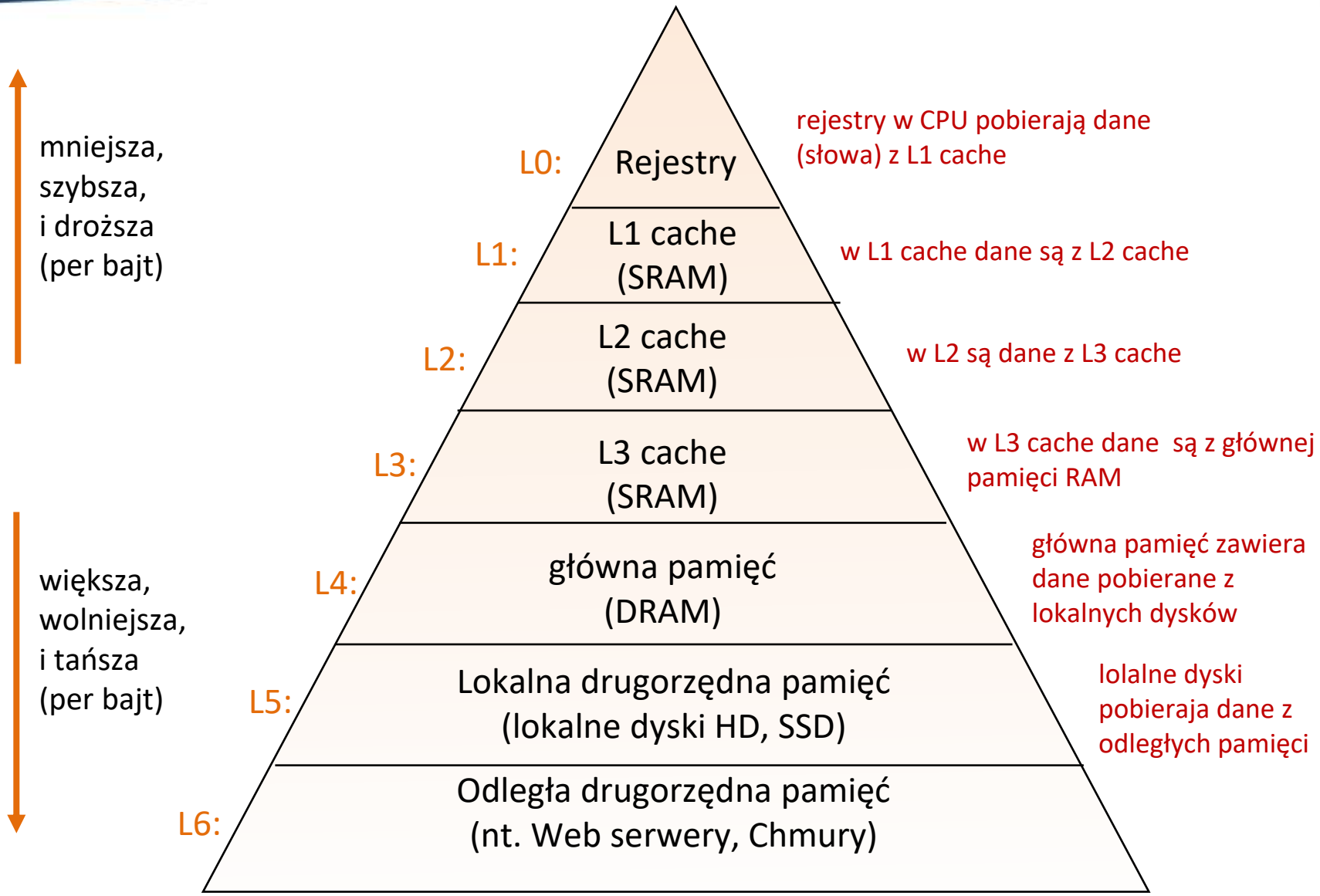


Hierarchia pamięci

- Kluczowe czynniki w rozwoju hardweru (sprzętu) i softweru (oprogramowania):
  - Technologie szybkich pamięci są:
    - drogie
    - dotyczą mniejszych pojemności
    - wymagają więcej energii -> wydzielanie więcej ciepła i grzanie
  - Luka pomiędzy CPU (wiele rdzeni) a główną pamięcią poszerza sie (wąskie gardło von Neumanna)
  - Dobry kod (z lokalnością) pozwała zmniejszyć nieco tę lukę

- Konieczna jest hierarchiczna organizacja pamięci

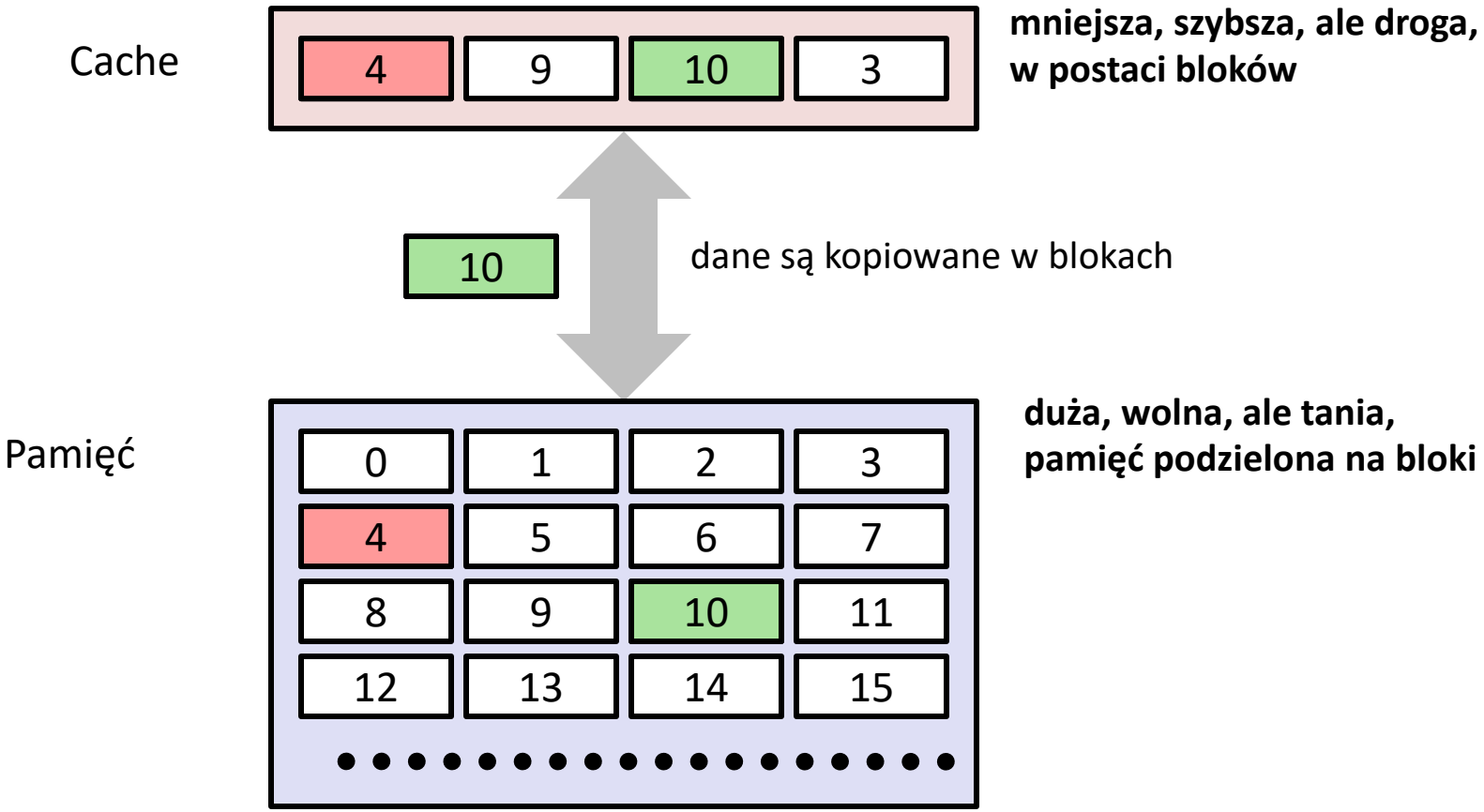
Hierarchia


mniejsza,
szybsza,
i droższa
(per bajt)
większa,
wolniejsza,
i tańsza
(per bajt)
L0:
Rejestry
rejestry w CPU pobierają dane (słowa) z L1 cache
L1:
L1 cache
(SRAM)
w L1 cache dane są z L2 cache
L2:
L2 cache
(SRAM)
w L2 są dane z L3 cache
L3:
L3 cache
(SRAM)
w L3 cache dane są z głównej pamięci RAM
L4:
główna pamięć
(DRAM)
główna pamięć zawiera dane pobierane z lokalnych dysków
L5:
Lokalna drugorzędna pamięć
(lokalne dyski HD, SSD)
lolalne dyski pobieraja dane z odległych pamięci
L6:
Odległa drugorzędna pamięć
(nt. Web serwery, Chmury)

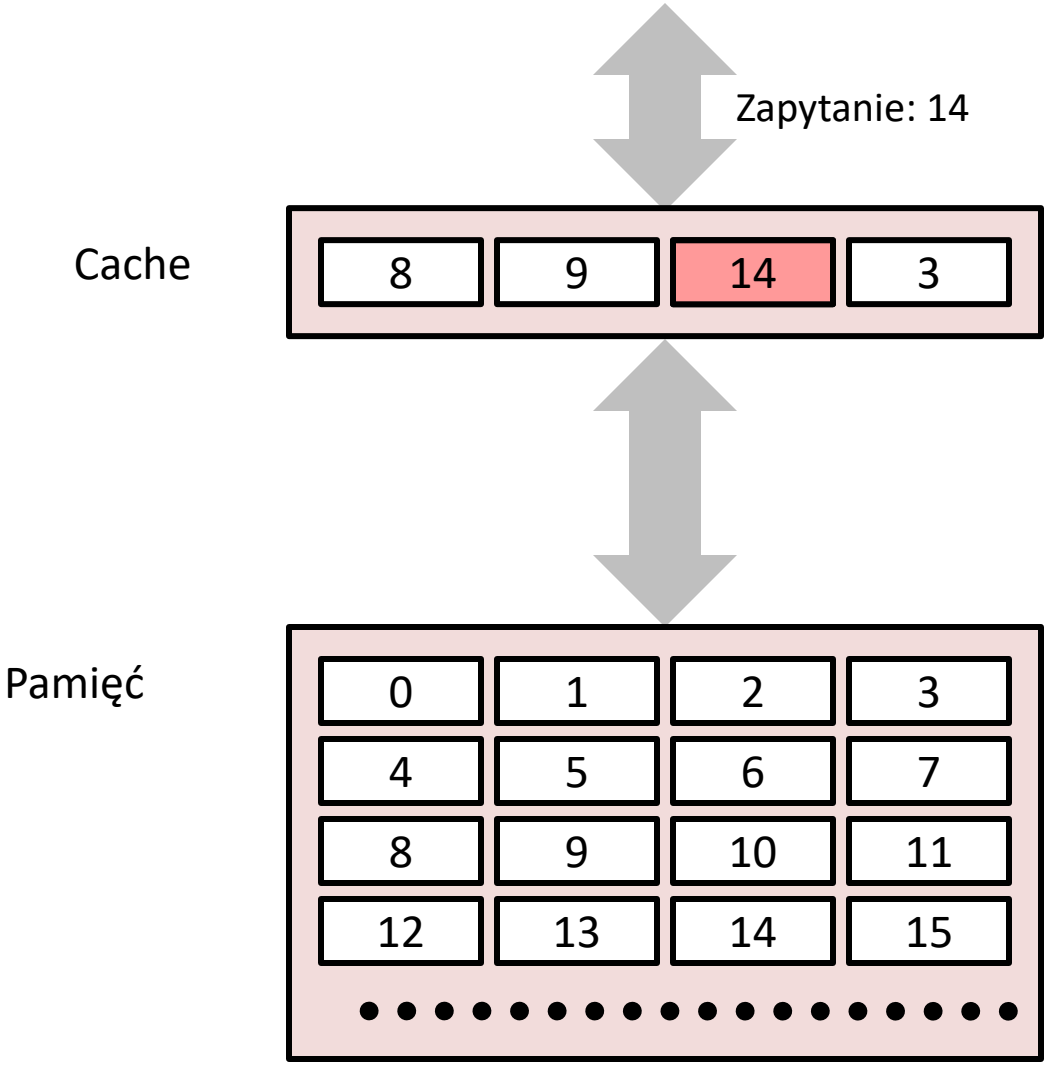
Cache

- *Cache:* mniejsza, ale szybsza pamięć
- Fundamentalna idea:
  - Dla każdego k, szybsza i mniejsza pamięć na poziomie k, służy jako cache dla większej ale wolniejszej pamieci na poziomie k+1
- Jaki jest cel?
  - Lokalność kodu powoduje, że instrukcje mogą sięgać po **dane** w cache na poziomie k , chociaż, oryginalnie, mogą być na znacznie głębszym poziomie
- *Idealnie:* Te dane powinny być umieszczone jak najwyżej, tj. jak najbliżej CPU

Cache: idea


Cache
4
9
10
3
mniejsza, szybsza, ale droga,
w postaci bloków
10
dane są kopiowane w blokach
Pamięć
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
duża, wolna, ale tania,
pamięć podzielona na bloki

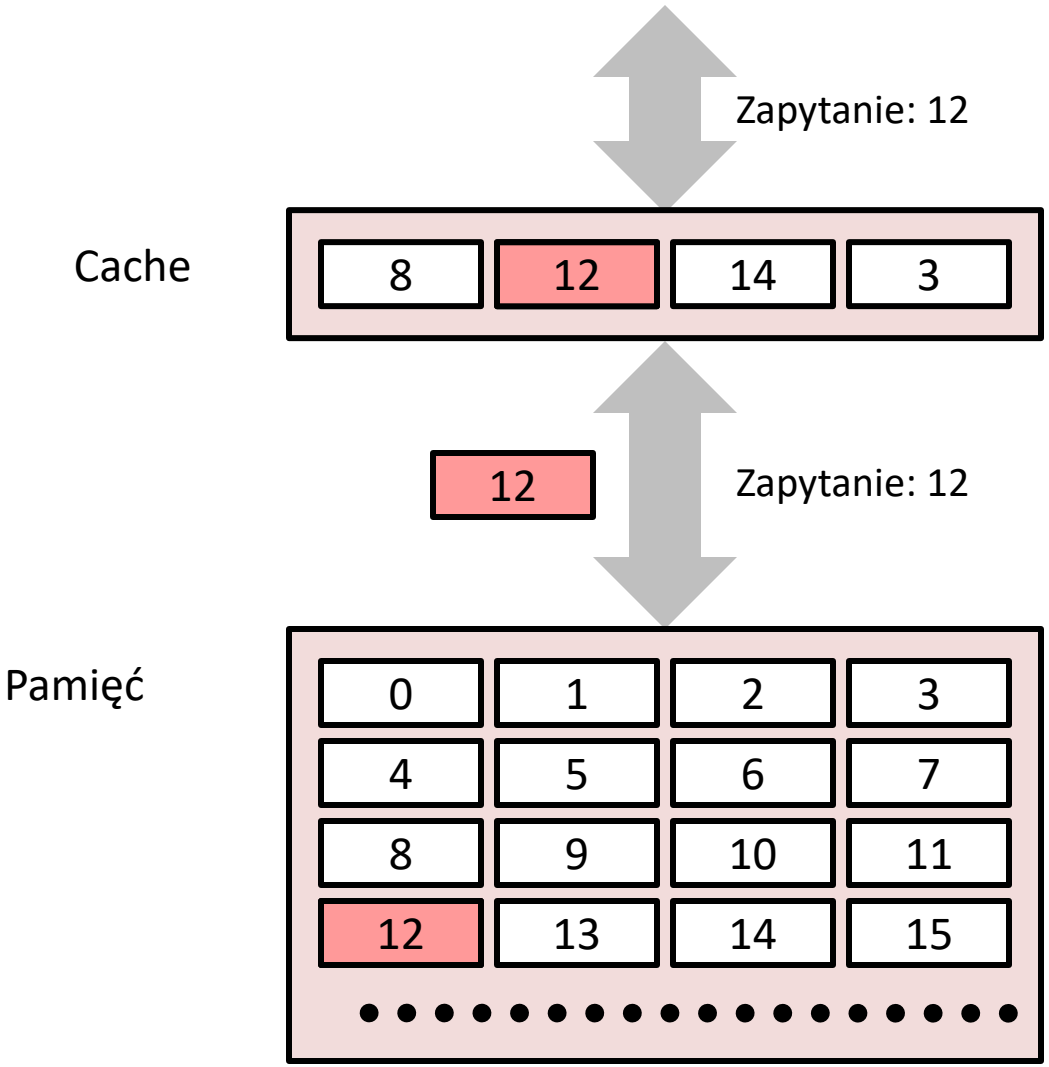
Cache : trafienie



potrzebne są dane z bloku b

blok b jest w cache:
trafienie!

Cache: pudło



potrzebne są dane z bloku b

bloku b nie ma w cache:
pudło!

blok b jest pobierany z Pamięci

blok b jest zapisany w cache

- potrzebny mechanizm do
- zarządzania:
  gdzie ma być *b* zapisany i
- zamiast jakiego bloku?

Cache: 3 typy niepowodzeń

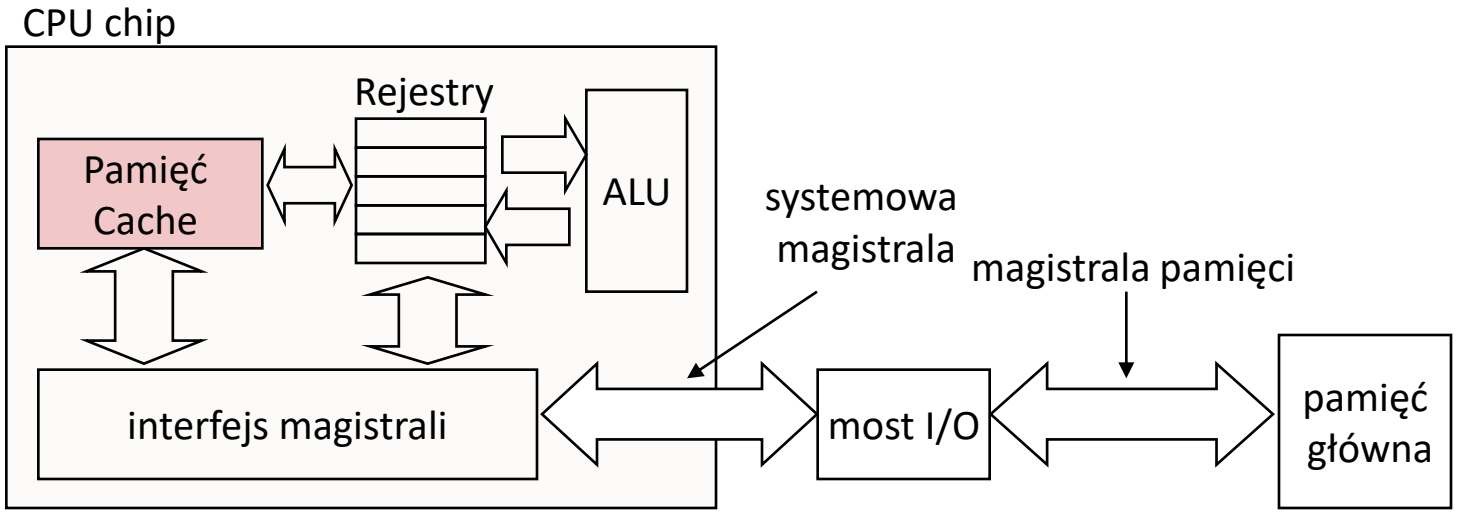
- przy starcie
  - na początku cache jest pusty, więc pierwsze zapytania będą nietrafione
- za mały cache
  - potrzebnych bloków jest więcej niż wielkość cache
- konflikt kolejnych zapytań
  - zazwyczaj bloki na poziomie głębszym są większe niż te na wyższym
  - konflikt może być, jeśli zapytania zmieniają się kolejno np.
  - 0 (nietrafione, zamiana 8 na 0),
  - 8 (nietrafione, zamiana 0 na 8),
  - 0 (nietrafione, zamiana 8 na 0),
  - 8 (nietrafione, zamiana 0 na 8),
  - ...

Cache

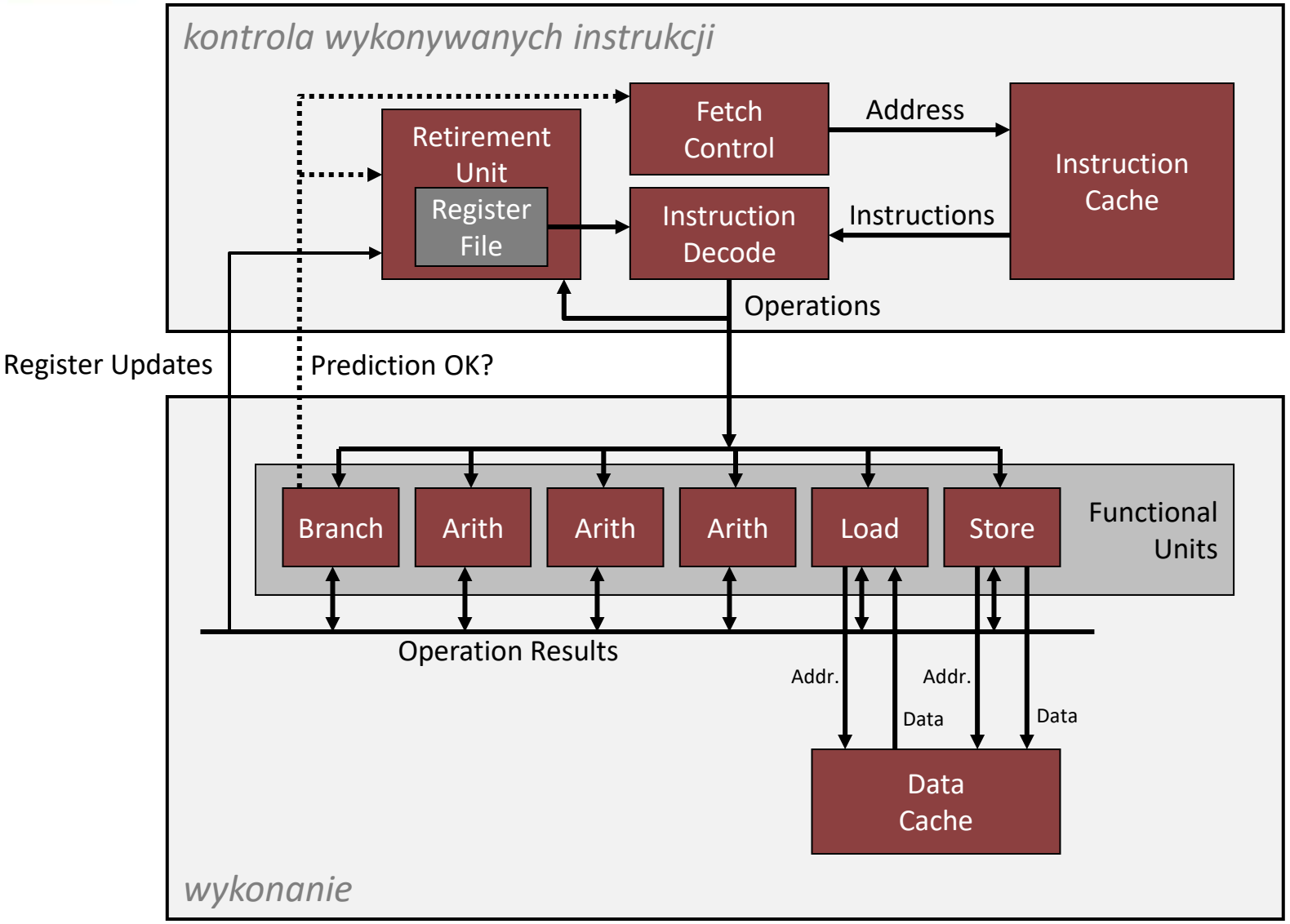
typ Cache	rozmiar bloków	zapisane	opóźnienie (w cyklach)	zarządzane przez
Registers	słowa 4-8 bajtowe	CPU core	0	kompilator
TLB	translacja adresów	On-Chip TLB	0	Hardware MMU
L1 cache	bloki 64-bajtowe	On-Chip L1	4	Hardware
L2 cache	bloki 64-bajtowe	On-Chip L2	10	Hardware
Virtual Memory	strony 4-KB	pamięć główna	100	Hardware + OS
Buffer cache	kawałki pliku	pamięć główna	100	OS
Disk cache	sektory dysku	kontroler dysku	100,000	Disk firmware
Network buffer cache	kawałki pliku	Lokalny dysk	10,000,000	NFS client
Browser cache	strony WWW	Lokalny dysk	10,000,000	Web browser
Web cache	strony WWW	odległe serwery dyskowe	1,000,000,000	Web proxy server

Cache

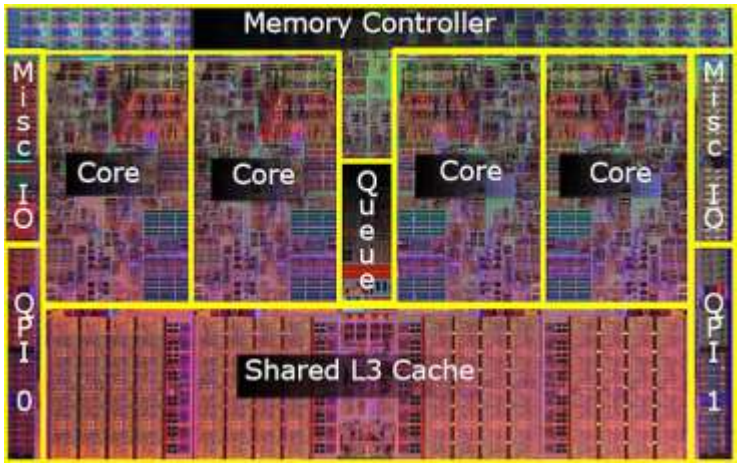
- Pamięć Cache to mała szybka pamięć oparta na SRAM zarządzana automatycznie w hardware
  - zawiera bloki z pamięci głównej (RAM) częst używane przez aktualnie wykonywany proces (procesy)
- CPU zwraca się najpierw do cache
- typowa architektura:



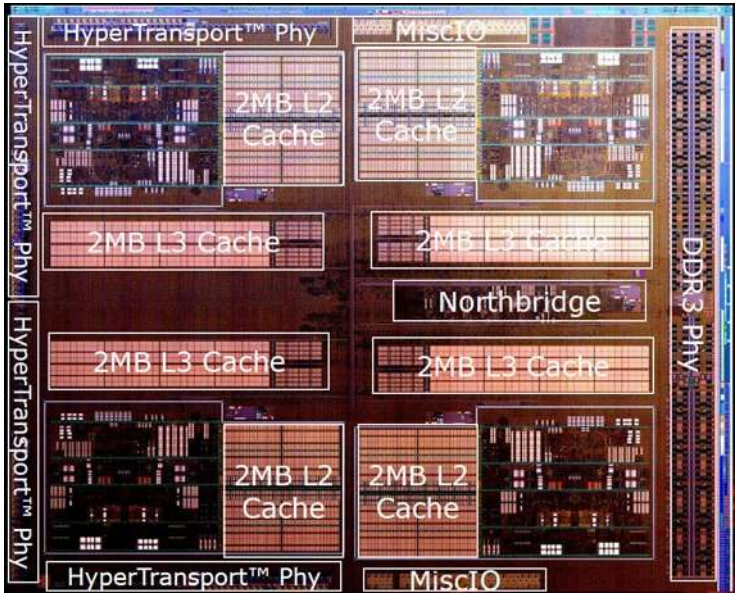
Współczesna architektura CPU



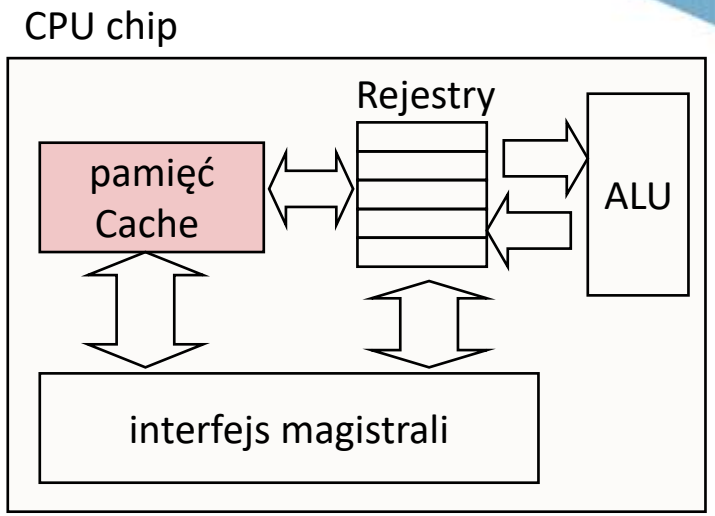
A jak to jest w czipach?



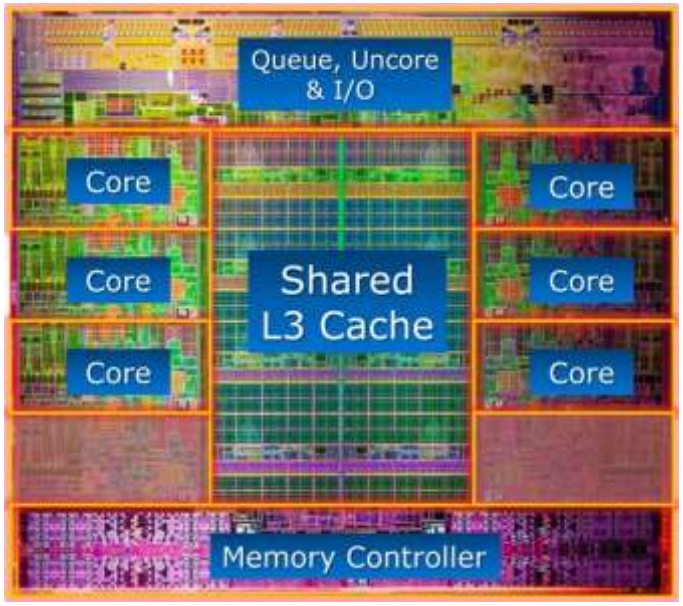
Nehalem



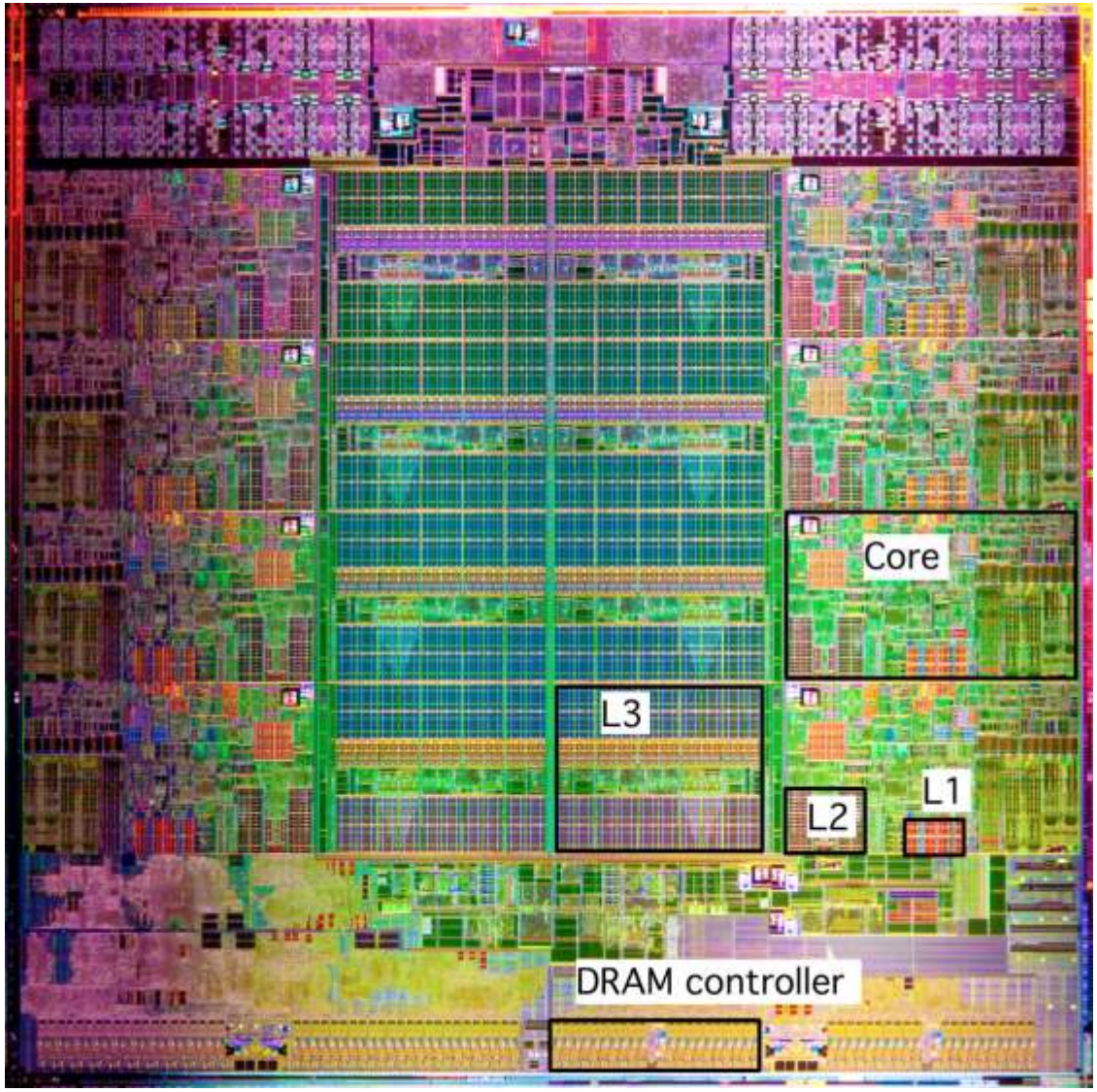
AMD FX 8150



Core i7-3960X



A jak to jest w czipach?(cd)



Intel Sandy Bridge
Processor Die

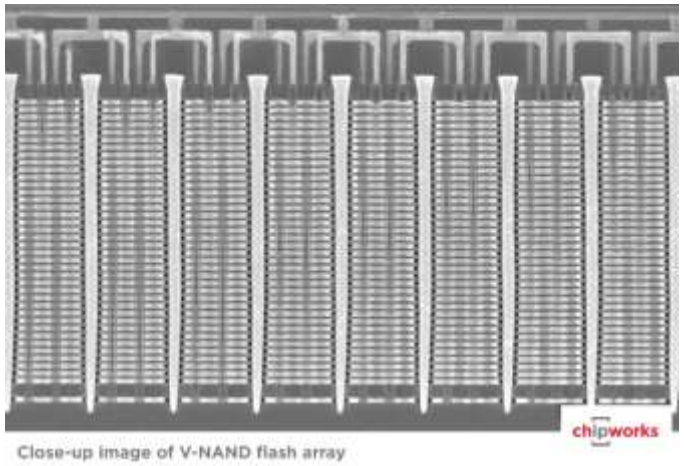
L1: 32KB Instruction + 32KB Data
L2: 256KB
L3: 3–20MB

Technologie

- magnetyczne dyski

- dostęp elektro-magnetyczny

- trwała (Flash) pamięć



Close-up image of V-NAND flash array

- implementowana jako struktura 3-D
  - 100+ poziomów komórek
  - 3 bity danych na komórkę

Co jest w środku?

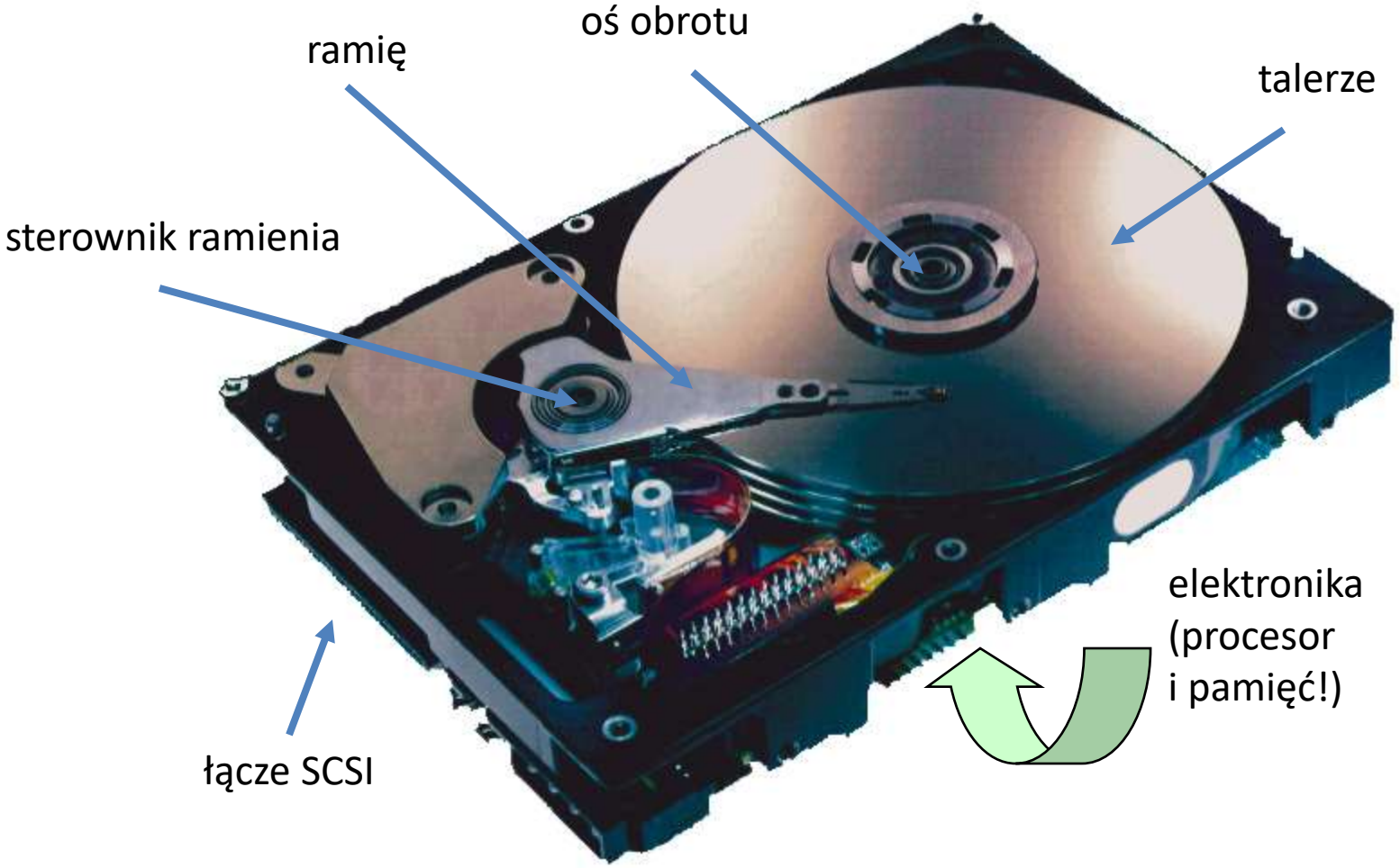
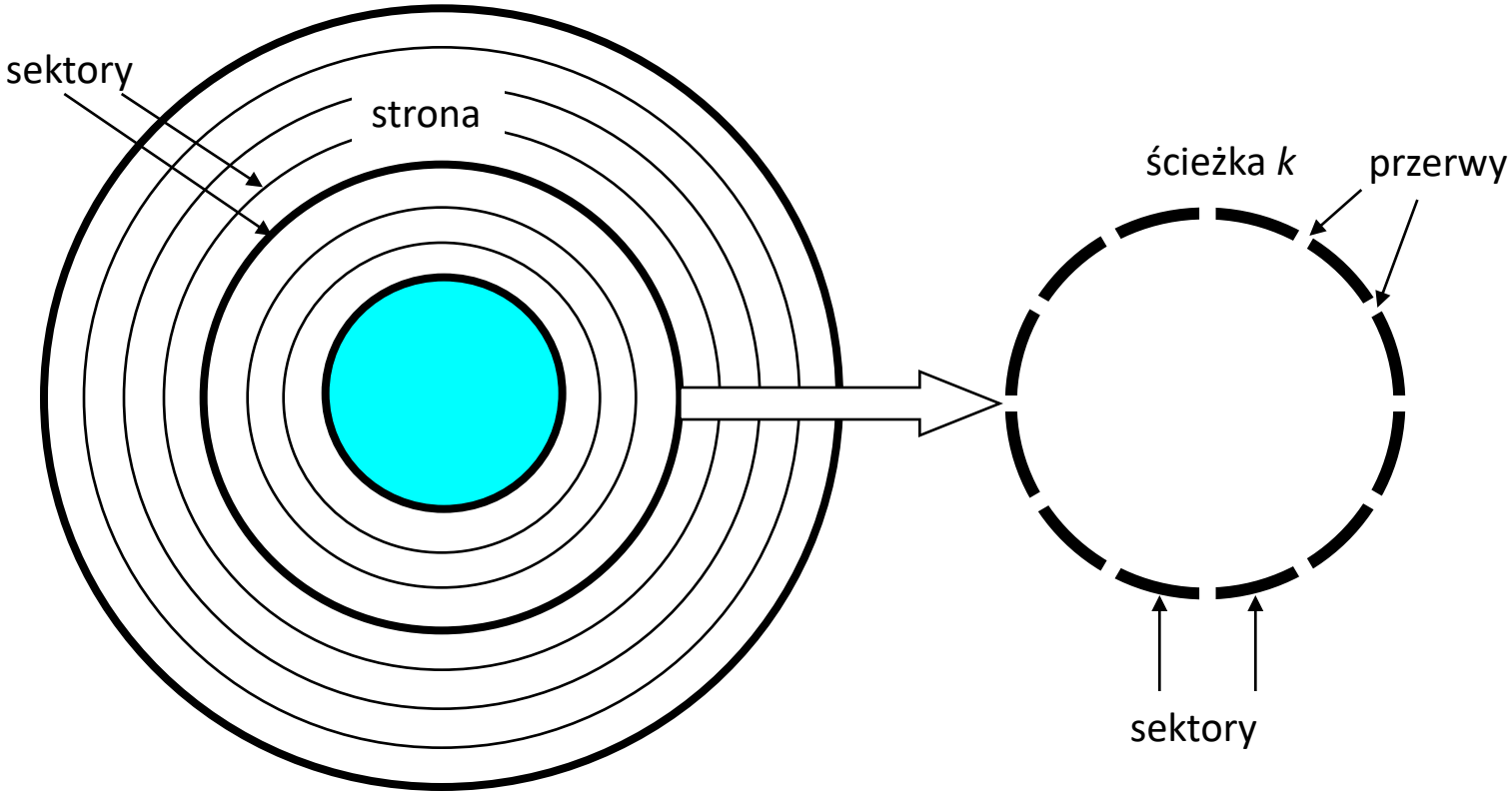


Image courtesy of Seagate Technology

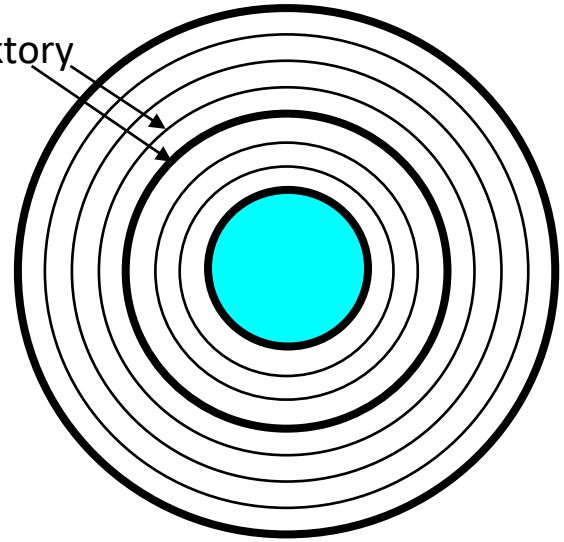
Struktura dysku

- Dysk składa się z talerzy, każdy ma dwie strony
- Strona składa się z koncentrycznych pierścieni zwanych ścieżkami (ang. tracks)
- Scieżka składa sie z sektorów odzielonych przerwami



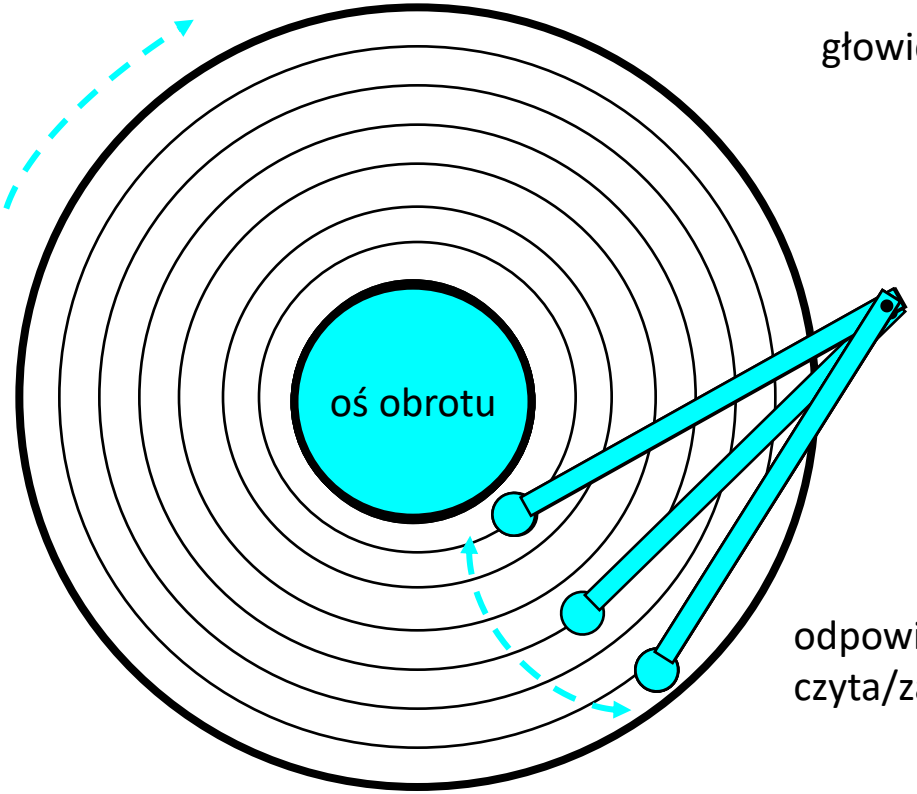
Pojemność dysku

- Pojemność: maksymalna ilość bajtów jak może być zapisana
  - gigabajty (GB), terabajty(TB), przy czym 1 GB = 10^9 Bajtów oraz 1 TB = 10^{12} Bajtów
- zależy od:
  - gęstości zapisu (bity/cal)
  - gęstość upakowania sektorów (sektory/cal)
  - gęstosc na powierzchni (bity/cal^2)



zapisywanie/czytanie

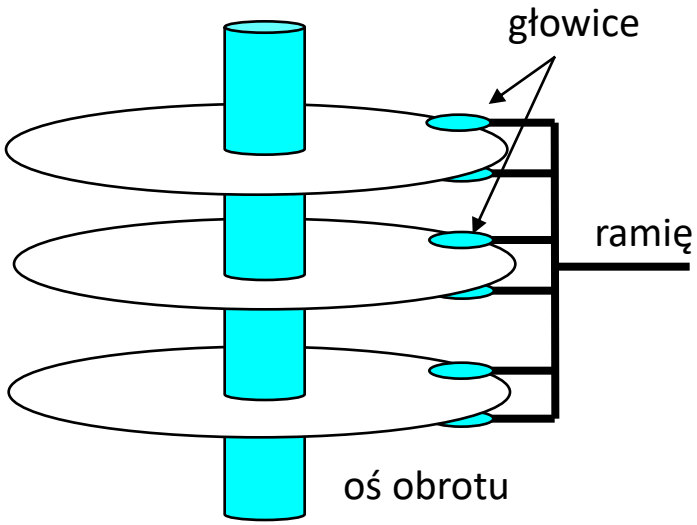
dysk wiruje ze stałą prędkością kątową



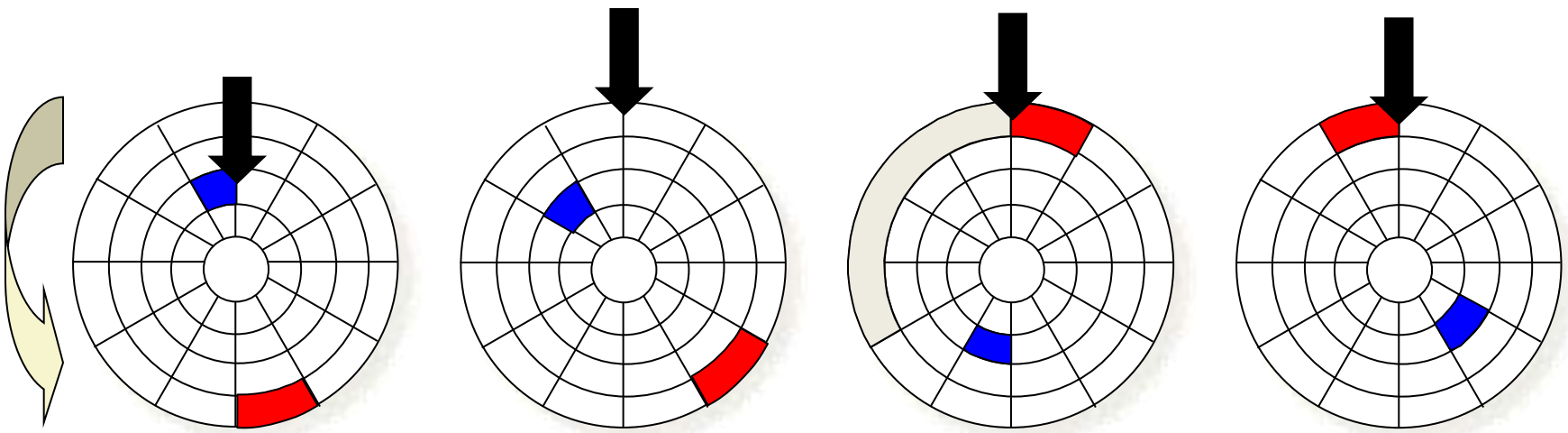
głowica (na końcu ramienia)

odpowiednio ustawiona czyta/zapisuje z/do sektorów

talerzy może być więcej, a każdy ma dwie strony



Dostęp do dysku – czasy poszczególnych operacji



Po przeczytaniu NIEBIESKIEGO sektora

transfer danych

poszukiwanie CZERWONEGO sektora

znajdź

Rotacja głowicy

opóźnienie rotacyjne

Po przeczytaniu CZERWONEGO sektora

transfer danych

Czas dostępu do dysku

- Średni czas dostępu to sektora docelowego jest wyznaczany przez :
  - $T_{dostęp} = T_{znajdź} + T_{rotacja} + T_{transfer}$
- Czas poszukiwania ($T_{znajdź}$)
  - Czas potrzebny do ustawienie głowicy nad docelowym sektorem
  - zwykle $T_{znajdź}$ to około 3—9 ms
- Opóźnienie rotacyjne ($T_{rotacja}$)
  - Czas oczekiwania na pierwszy skopiowany bit z sektora docelowego dostarczony do głowicy
  - $T_{rotacja}$ = 1/2 x 1/RPMs x 60 sec/1 min
  - typowa prędkość obrotowa = 7,200 RPMs
- Czas transferu ($T_{transfer}$)
  - czas potrzebny do przeczytania bitów z docelowego sektora T
  - $T_{transfer}$ = 1/RPM x 1/(avg # sectors/track) x 60 secs/1 min

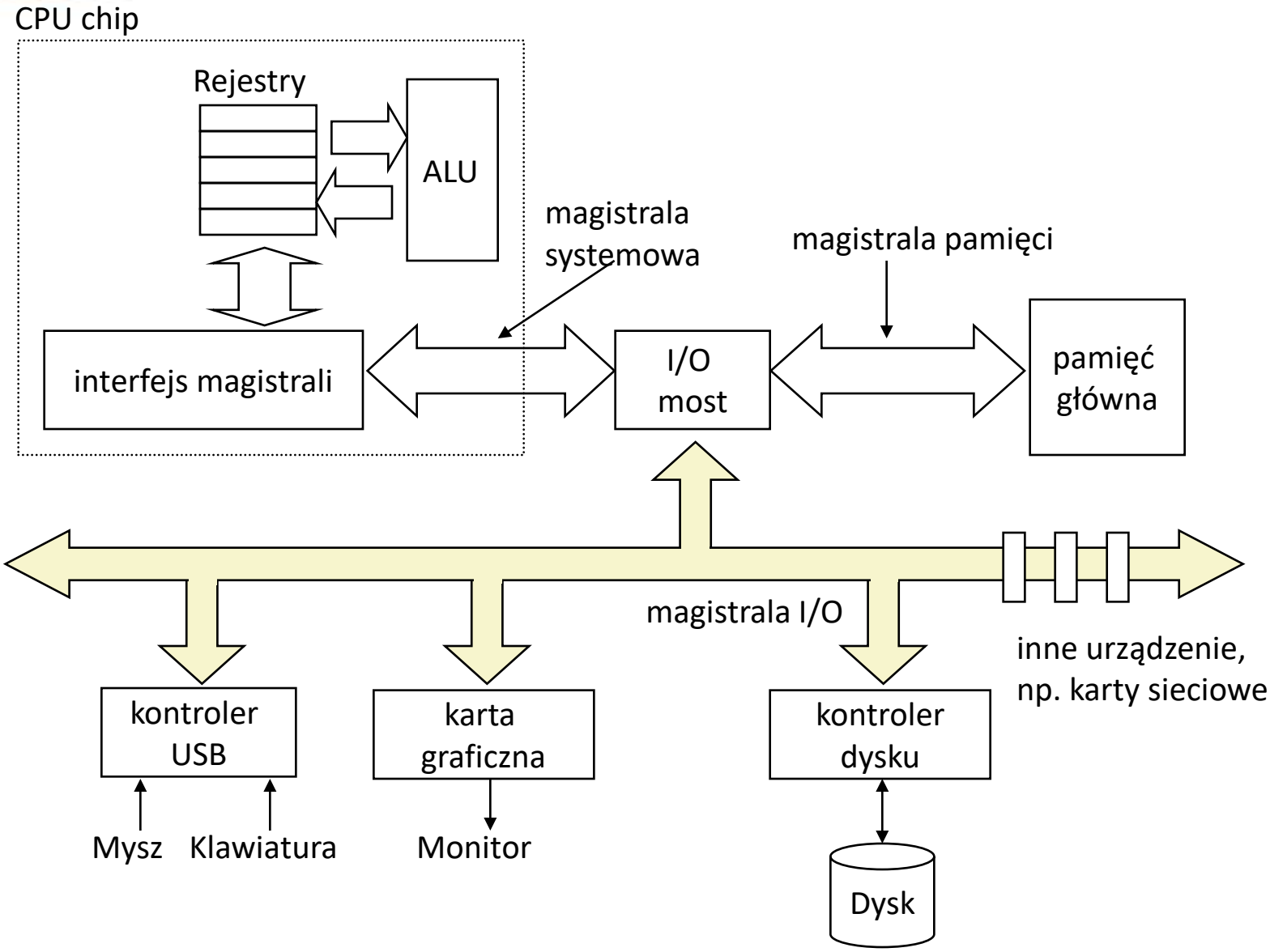
czas jednej rotacji-obrotu (w minutach)

ułamek obrotu do przeczytania

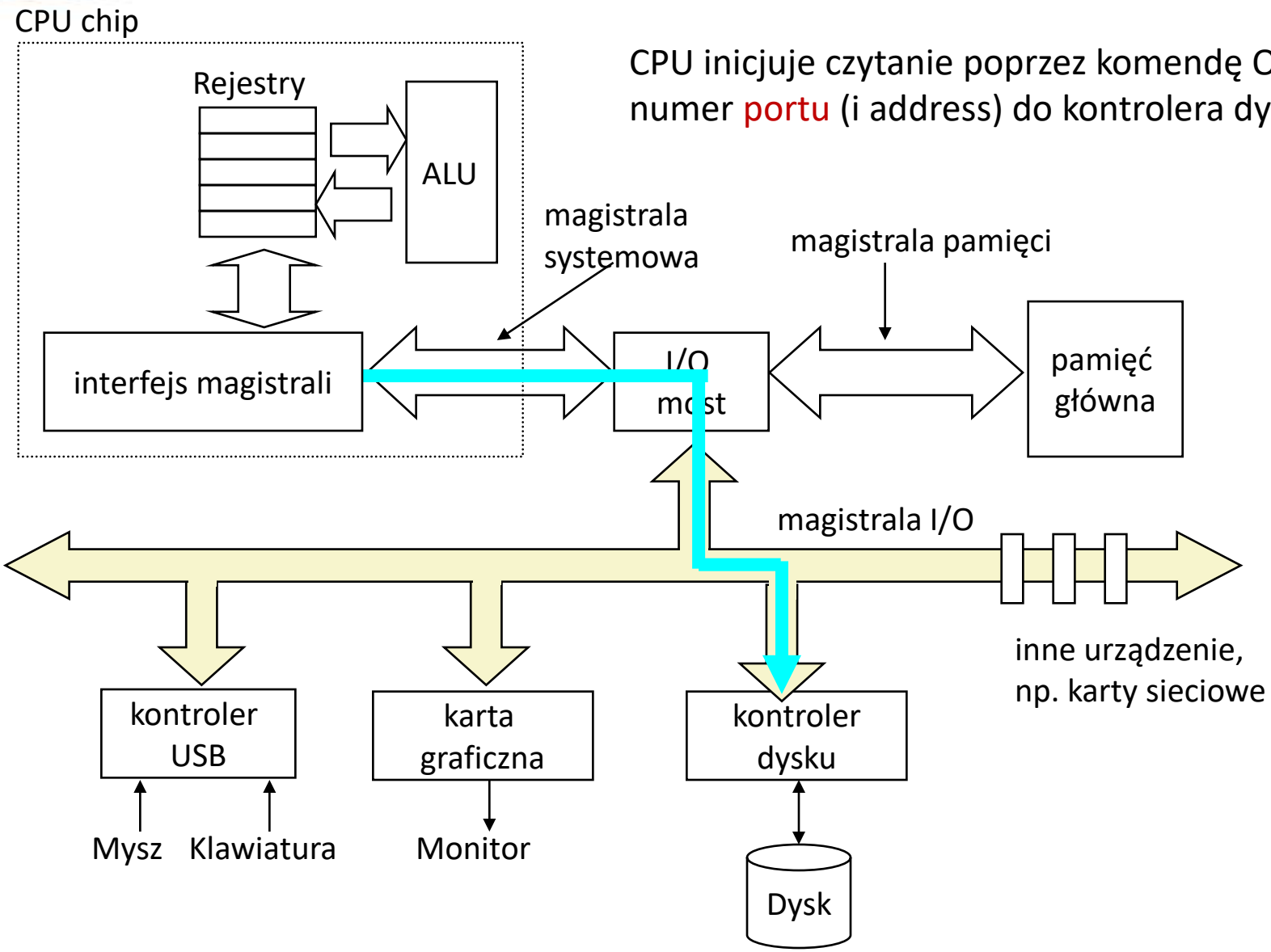
Czas dostępu do dysku - przykład

- Dane są:
  - prędkość obrotowa = 7,200 RPM
  - średni czas wyszukiwania = 9 ms
  - średnia liczba sektorów na ścieżkę (# sectors/track) = 400
- Liczymy:
  - $T_{rotacja}$ = 1/2 x (60 secs/7200 RPM) x 1000 ms/sec = 4 ms
  - $T_{transfer}$ = 60/7200 x 1/400 x 1000 ms/sec = 0.02 ms
  - $T_{dostęp}$ = 9 ms + 4 ms + 0.02 ms
- Ważne uwagi:
  - Czas dostępu zdominowany przez czas wyszukiwania opóźnienie rotacyjne
  - pierwszy bit w sektorze jest czasowo najdroższy, reszta prawie za darmo
  - ***czas dostepu dla SRAM to ok. 4 ns/double-word, dla DRAM to ok. 60 ns***
    - ***Dyski HD sa ok. 40 000 razy wolniejsze od SRAM,***
    - ***2 500 razy wolniejsze od DRAM.***

magistrala I/O



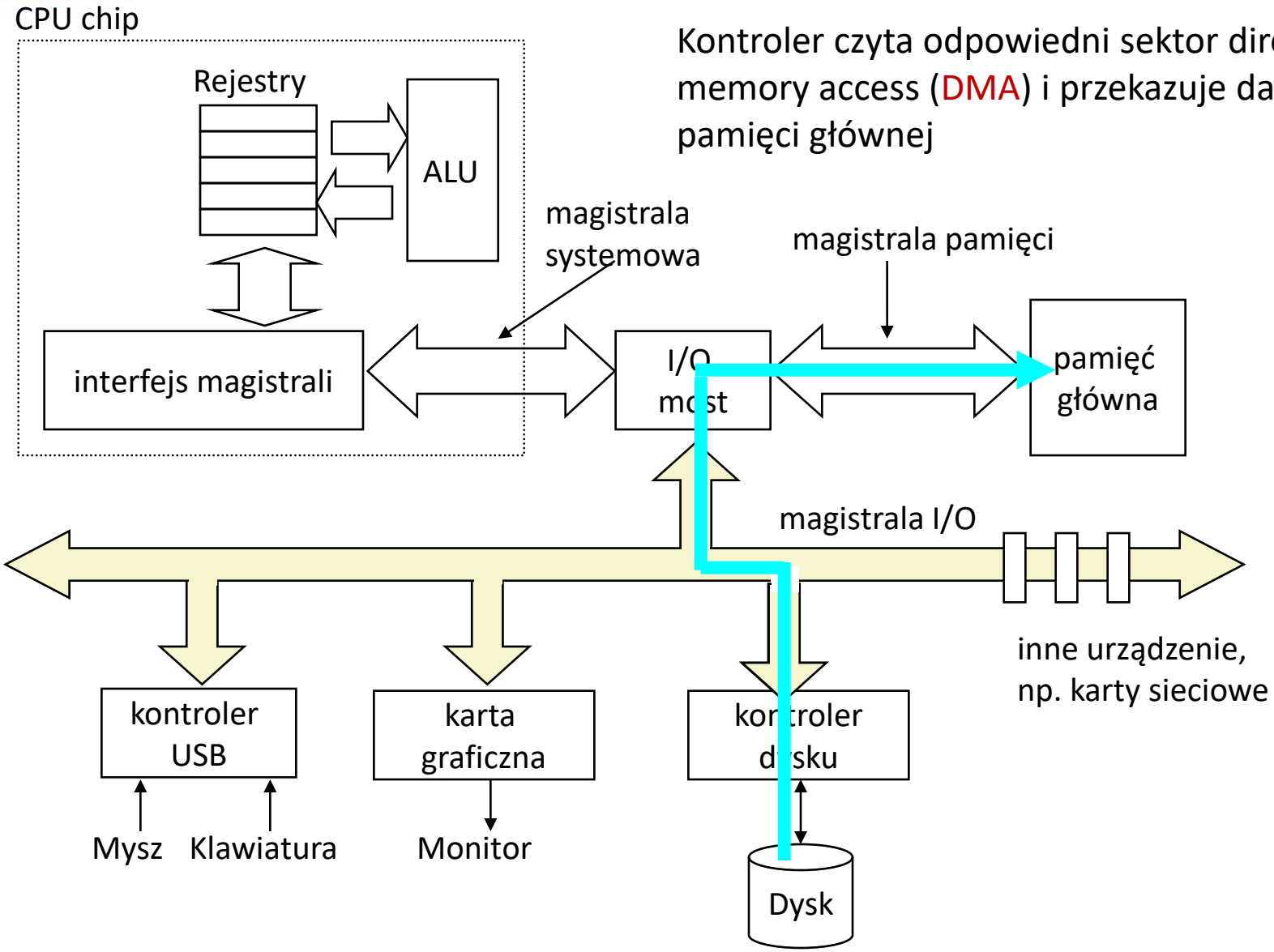
czytanie z dysku (1)



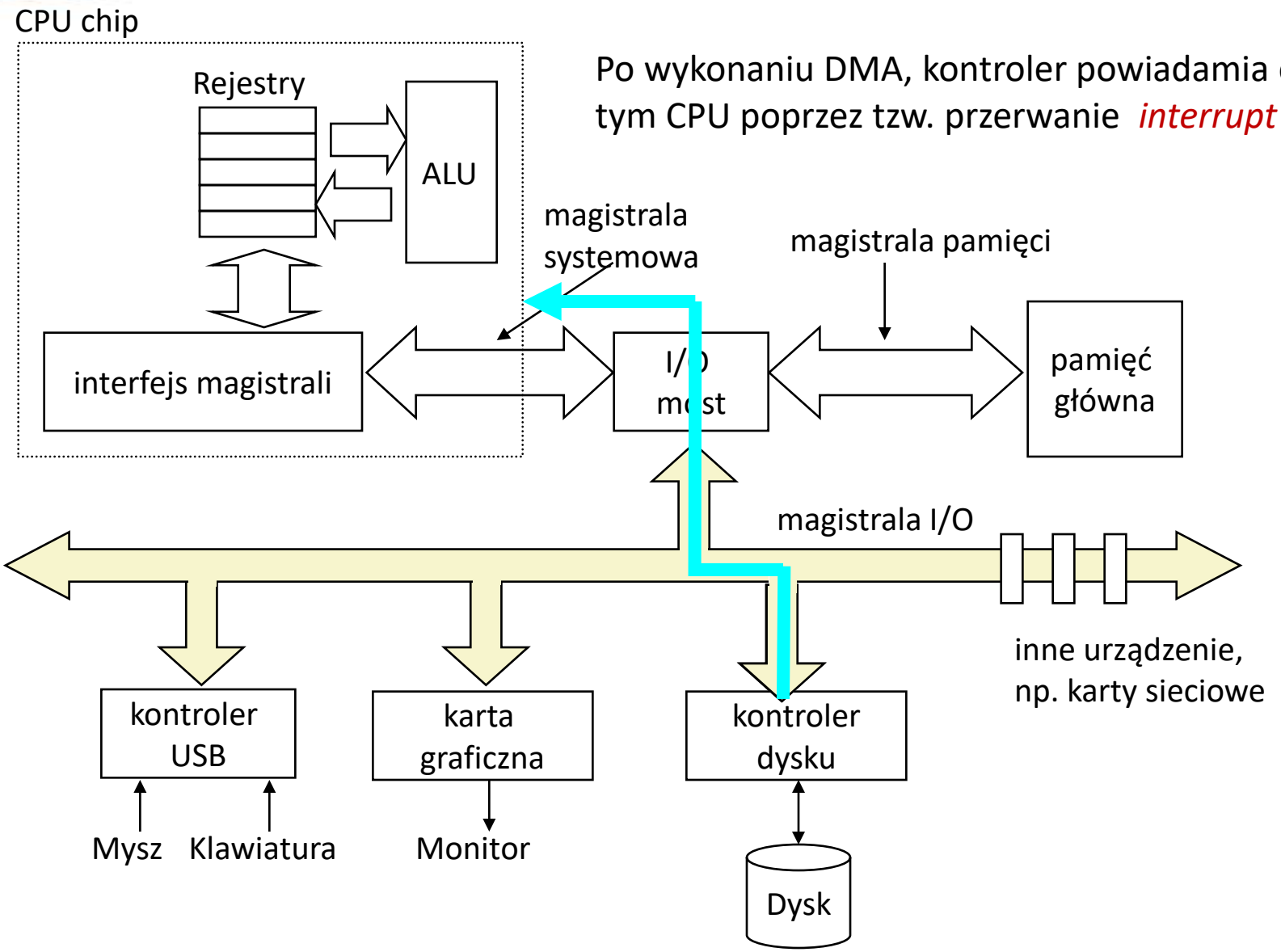
CPU inicjuje czytanie poprzez komendę Out numer portu (i address) do kontrolera dysku

czytanie z dysku (2)

Kontroler czyta odpowiedni sektor direct memory access (DMA) i przekazuje dane do pamięci głównej



czytanie z dysku (3)



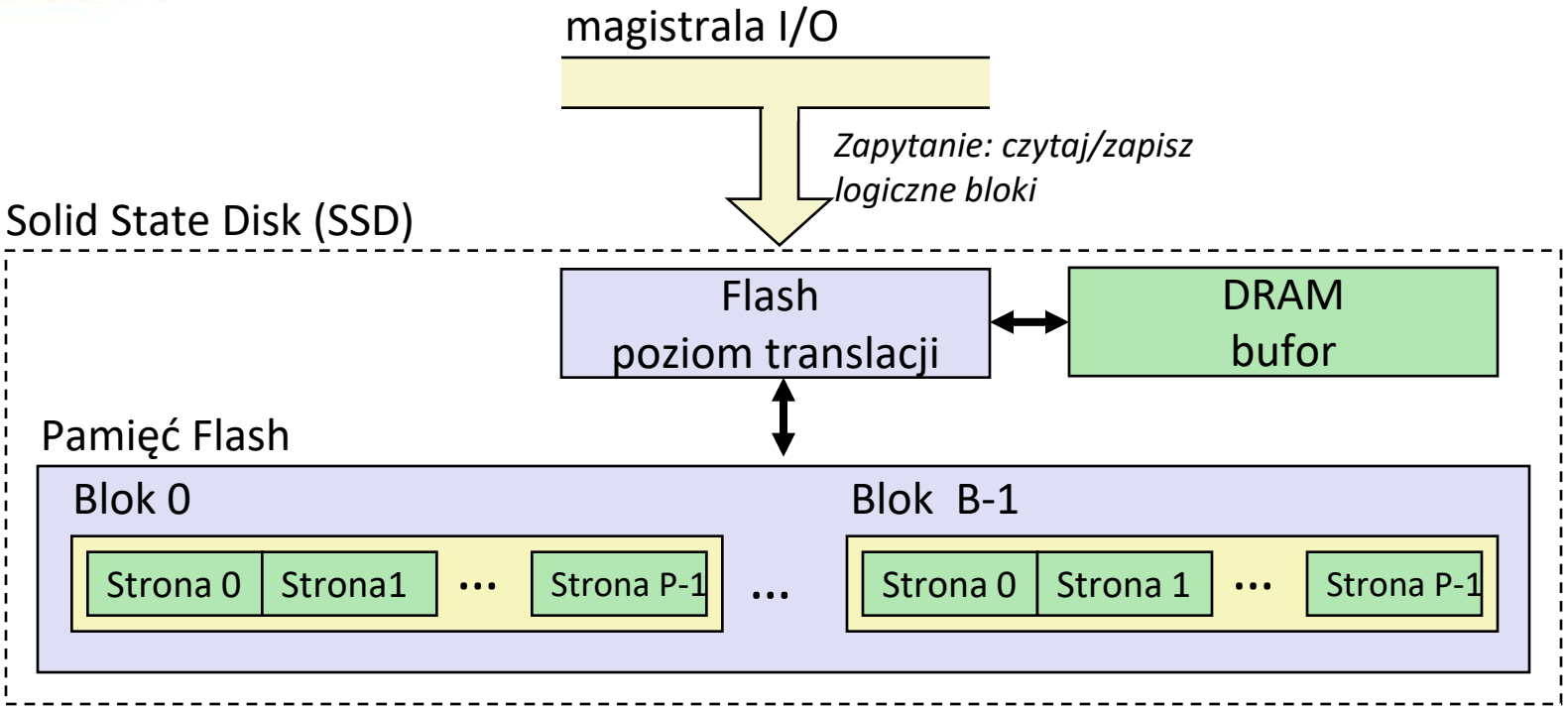
Po wykonaniu DMA, kontroler powiadamia o tym CPU poprzez tzw. przerwanie *interrupt*

Nietrwałe pamięci

- DRAM oraz SRAM są pamięciami nietrwałymi
  - dane są tracone po wyłączeniu zasilania
- Trwałe pamięci zachowują dane po wyłączeniu zasilania
  - Read-only memory (ROM): na stałe podczas ich wytwarzania
  - Electrically eraseable PROM (EEPROM): można zapisywać i nadpisywać
  - Pamięć Flash: EEPROM, można zmieniać zapis do ok. 100 000 razy
  - 3D XPoint (Intel Optane) & emerging NVMs

- Trwałe pamięci
  - oprogramowanie sprzętowe ROM
    - BIOS,
    - kontrolery dysków, karty sieciowe, karty graficzne, bezpieczeństwa

Solid State Disks (SSDs)



- Strony: od 512KB do 4KB, Bloki: od 32 do 128 stron
- Czytanie/wpisywanie: jednostką jest strona.
- Wpisywanie tylko jeśli uprzednio jej blok został wymazany z poprzedniego wpisu
- Blok jest przeznaczony na ok 100 000 zmian-wpisów

SSD kontra dyski HD

- Zalety
  - nie ma ruchomych części→ szybszy, zużywa mniej energii, bardziej odporny na wstrząsy
- Wady
  - szybciej się zużywa, jest mniej trwały
  - w 2019 r., 4 razy droższy, ale ceny spadają
- Zastosowania
  - MP3 players, smart phones, laptops
  - coraz częściej w PC-tach i serwerach

Podsumowanie

- Luka pomiedzy CPU a pamięcią (zwłaszcza masową) zwiększa się coraz bardziej
- Dobrze napisany kod *z lokalnością* może nieco tę lukę zmniejszyć
- Hierarchiczna organizacja pamięci i keszowanie pomaga wykorzystując *lokalność*
- Pamięć Flash i dalszy jej postęp technologiczny mogą zastąpić DRAM, SRAM, magnetyczne dyski ...

Storage Trends

SRAM

Metric	1985	1990	1995	2000	2005	2010	2015	*2015:1985*
$/MB	2,900	320	256	100	75	60	*320*	*116*
access (ns)	150	35	15	3	2	1.5	*200*	*115*

DRAM

Metric	1985	1990	1995	2000	2005	2010	2015	*2015:1985*
$/MB	880	100	30	1	0.1	0.06	0.02	*44,000*
access (ns)	200	100	70	60	50	40	20	*10*
typical size (MB)	0.256	4	16	64	2,000	8,000	16.000	*62,500*

Disk

Metric	1985	1990	1995	2000	2005	2010	2015	*2015:1985*
$/GB	100,000	8,000	300	10	5	0.3	0.03	*3,333,333*
access (ms)	75	28	10	8	*5*	*3*	*3*	*25*
typical size (GB)	0.01	0.16	1	20	160	1,500	3,000	*300,000*

CPU Clock Rates

Inflection point in computer history when designers hit the "Power Wall"

	1985	1990	1995	2003	2005	2010	2015	*2015:1985*
CPU	80286	80386	Pentium	P-4	Core 2	Core i7(n)	Core i7(h)	
Clock rate (MHz)	6	20	150	3,300	2,000	2,500	3,000	500
Cycle time (ns)	166	50	6	0.30	0.50	0.4	0.33	500
Cores	1	1	1	1	2	4	4	4
Effective cycle time (ns)	166	50	6	0.30	0.25	0.10	0.08	2,075

(n) Nehalem processor
(h) Haswell processor

Dziękuję za uwagę!

Slajdy na podstawie wykładów prof. Stanisława Ambroszkewicza