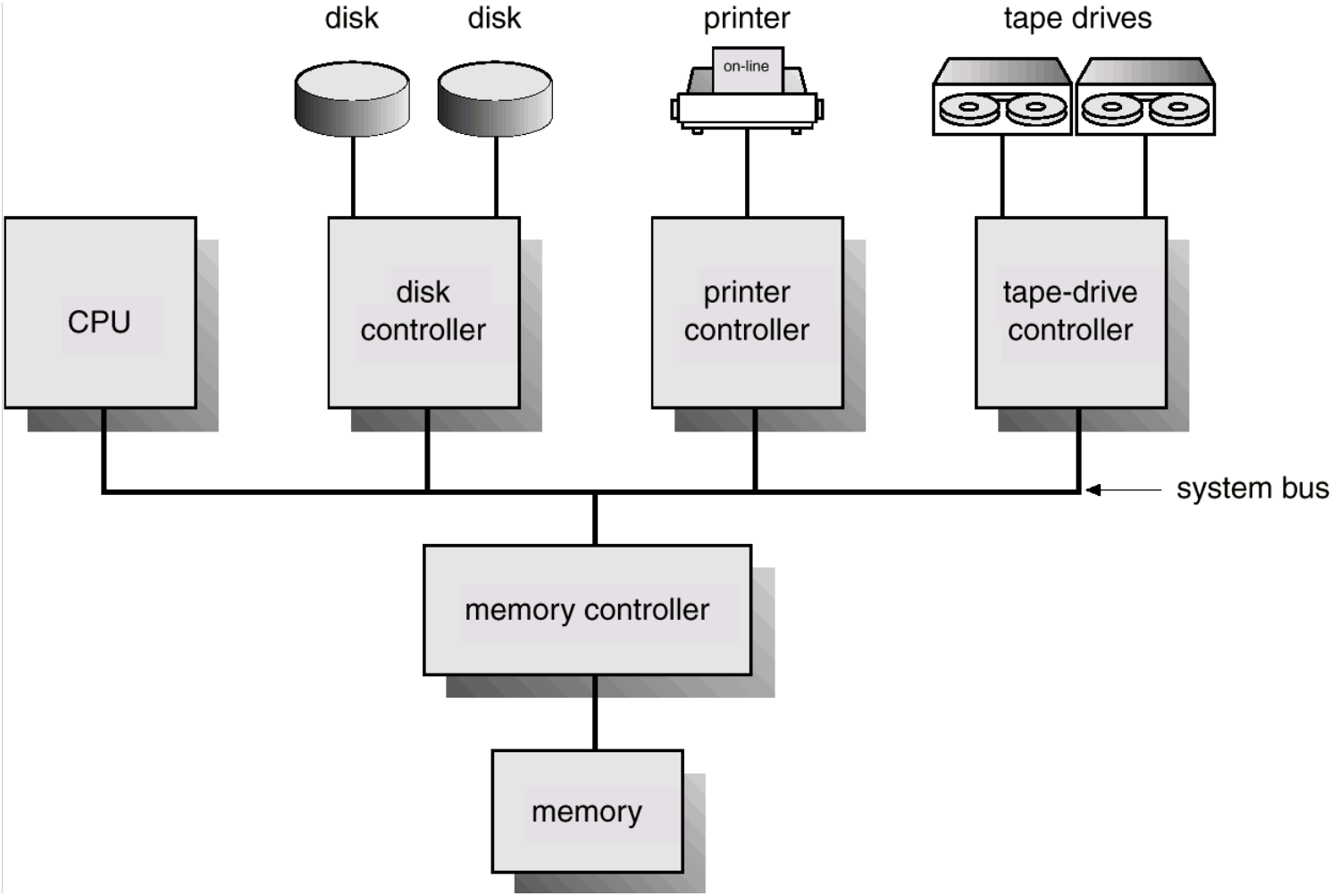
Wykład 2

Struktury systemów komputerowych istotne z punktu widzenia systemu operacyjnego

Uproszczony schemat architektury komputera



- Procesor, pamięć i urządzenia we-wy podłączone są do wspólnej **magistrali.**
- W rzeczywistych systemach mamy do czynienia z kilkoma fizycznymi magistrralami (PCI, ISA, USB, ...)

Praca systemu komputerowego

- Procesor i urządzenia wejścia-wyjścia mogą pracować współbieżnie
- Każdy kontroler we-wy obsługuje jeden typ urządzeń
- Każdy kontroler-posiada lokalny bufor
- Procesor przesyła dane do/z pamięci oraz do/z lokalnych buforów
- Wejście-wyjście przeprowadzane jest pomiędzy lokalnym buforem kontrolera a urządzeniem.
- Kontroler informuje o zakończeniu operacji zgłaszając **przerwanie.**

- **SYSTEM OPERACYJNY OPIERA SIĘ NA PRZERWANIACH !!!**

Przerwania

- Przerwania programowe (ang. trap)
  - Wywołanie systemu operacyjnego (np. specjalny rozkaz **syscall** w procesorach MIPS)
  - Rozkaz pułapki (brk w x86)
- Sprzętowe zewnętrzne (asynchroniczne względem programu)
  - Kontroler we-wy informuje procesor o zajściu zdarzenia, na przykład
    - Zakończenie transmisji danych.
    - Nadejście pakietu z sieci.
    - Przerwanie zegara.
    - Błąd parzystości pamięci.
- Sprzętowe wewnętrzne, głównie niepowodzenia (ang. fault)
  - Dzielenie przez zero
  - Przepełnienie stosu
  - Brak strony w pamięci (w przypadku implementacji stronicowania)
  - Naruszenie mechanizmów ochrony.

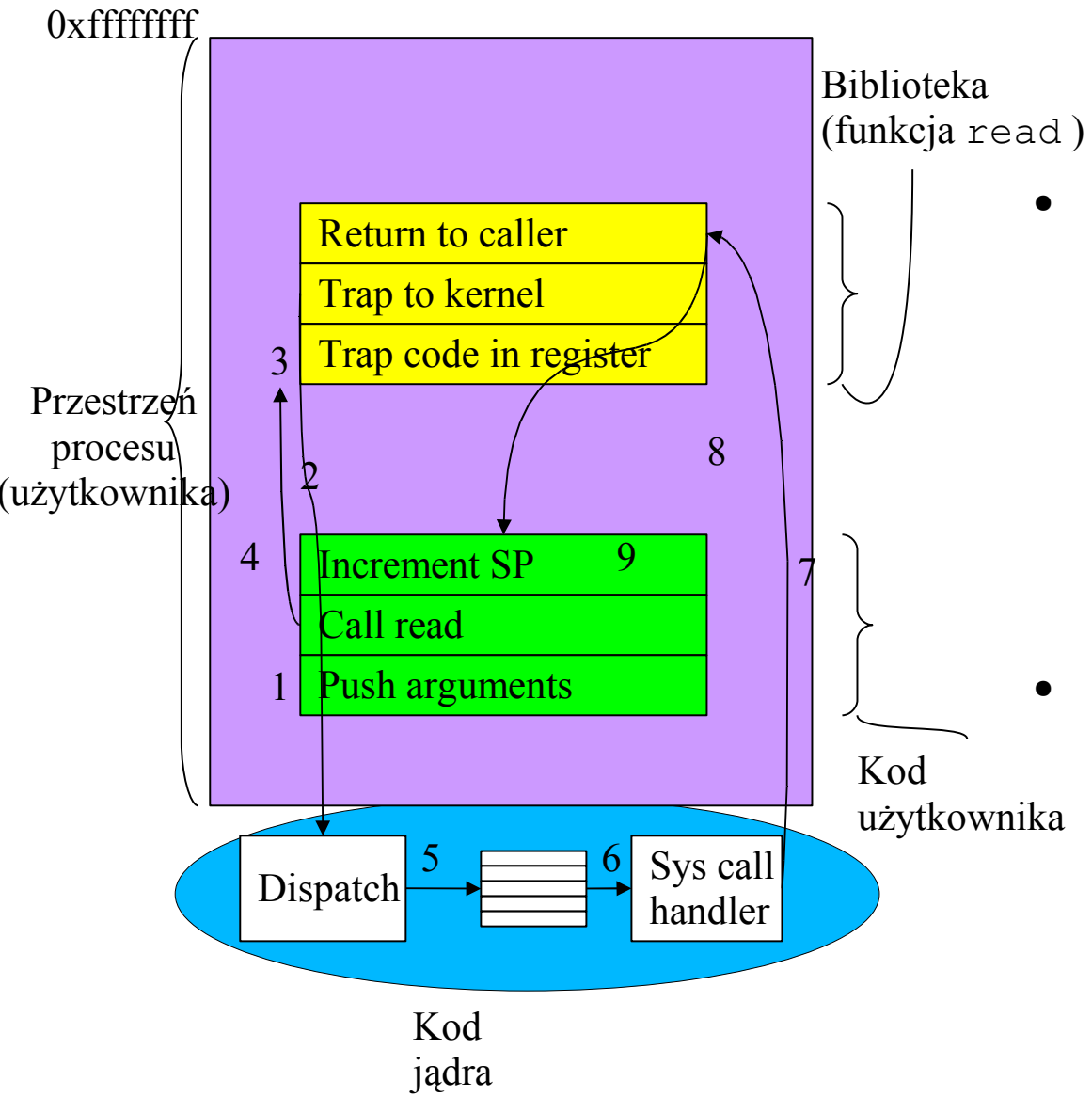
Obsługa przerwania

- Wykonywana przez system operacyjny
  - Zapamiętanie stanu procesora (rejestrów i licznika rozkazów)
  - Określenie rodzaju przerwania
    - przepytywanie (ang. Polling)
    - wektor przerwań (tablica adresów indeksowana numerem przerwania)
  - Przejście do właściwej procedury obsługi
  - Odtworzenie stanu procesora i powrót z przerwania
- Uwaga: Odtworzenie stanu procesora może dotyczyć innego procesu niż zapamiętanie. (**przełączenie kontekstu** – ang. context switch). Przykład:
  - Wykonuje się proces A
  - Przerwanie zegara=>zapamiętanie stanu procesu A
  - System operacyjny stwierdza że A zużył cały przydzielony kwant czasu procesora. System postanawia przekazać sterowanie procesowi B.
  - Odtworzenie stanu procesu B(przełączenie kontekstu) => powrót z przerwania
  - Wykonuje się proces B

Krótko o procesach

- Najprościej możemy określić proces jako ***"Wykonujący się program"***
- Ta definicja ma pewne niuanse. Rozpatrzmy przypadek procesu powstałego w wyniku uruchomienia programu użytkownika.
  - Proces może wywoływać kod tego programu.
  - Mówimy, że proces ***"proces wykonuje się w trybie użytkownika".***
  - Może także wywoływać funkcję bibliotek współdzielonych (.so w Linuksie, biblioteki dynamiczne .dll w Windows), które formalnie nie są częścią programu.
- Po pewnym czasie proces (tzn. kod programu lub biblioteki) decyduje się wykonać funkcję systemową, np. otworzyć plik
- Mówimy wtedy, że ***"proces wykonuje się w trybie jądra".***
  - W tej sytuacji proces wykonuje kod systemu operacyjnego, a nie kod programu z którego został wczytany.
  - Widzimy, że nie ma relacji 1<=>1 pomiędzy procesami i programami, ponieważ proces wykonuje kod dwóch programów: swojego i jądra systemu

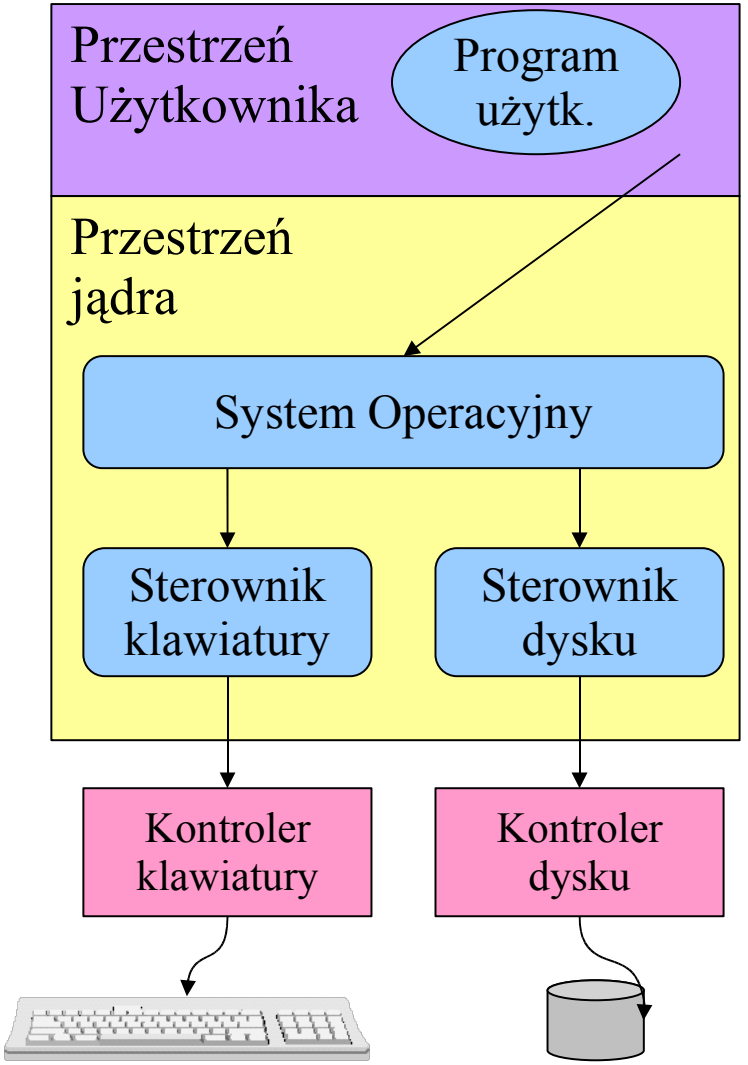
Anatomia wywołania systemowego w Uniksie na przykładzie ***read(plik,bufor,ile)***



- Program użytkownika w języku C
  - (1) Przekazanie argumentów na stosie
  - (2) Wywołanie funkcji bibliotecznej *read*
- Funkcja biblioteczna *read*
  - (3) Załadowanie numeru wywołania do rejestrów
  - Załadowanie argumentów wywołania do rejestrów
  - (4) Przerwanie programowe – przekazanie sterowania do systemu
- System operacyjny
  - (5) Procedura obsługi przerwania (na podstawie numeru wywołania i tablicy adresów (6)) wykonuje skok do odpowiedniego handlera
  - Handler powraca do funkcji *read* (7), a ta do programu użytkownika (8).

Sterowniki urządzeń we-wy (ang. device drivers)

- Fragment kodu jądra usytuowany pomiędzy kontrolerem we-wy a resztą jądra.
  - Zapewnia to standaryzacje interfejsu do różnego rodzaju urządzeń.
  - Jądro wykorzystuje interfejs (zunifikowany) do komunikacji ze sterownikami urządzeń.
- Sterownik komunikuje się z kontrolerem przy pomocy magistrali systemowej.
- Kontroler komunikuje się z urządzeniem we-wy.

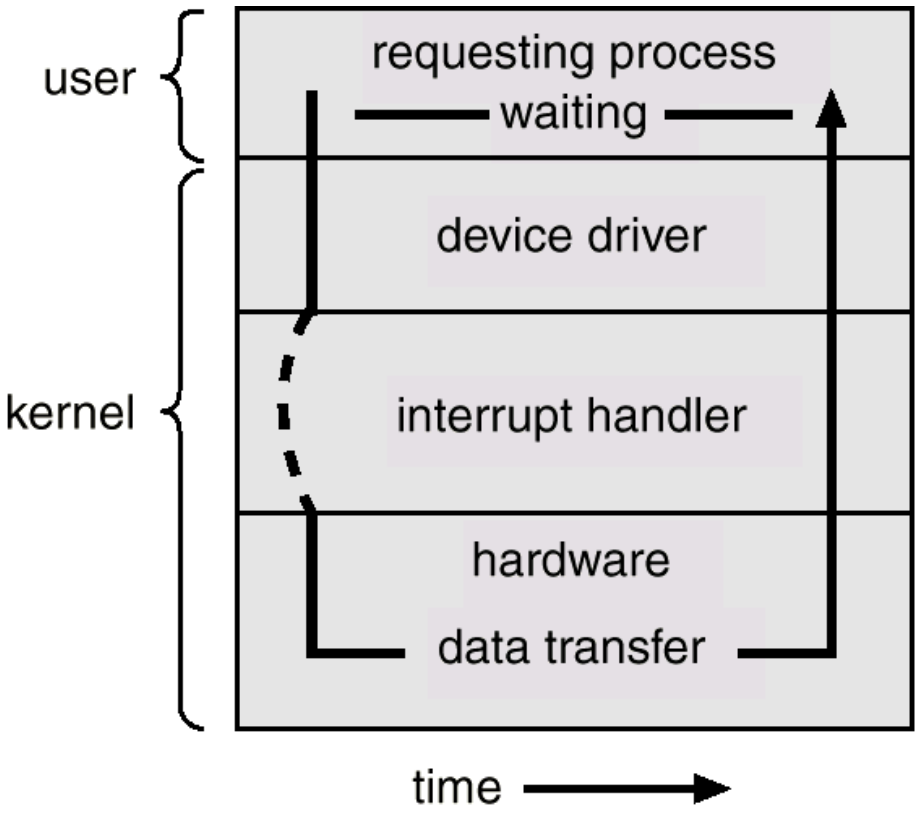


Przykład interfejsu jądro <=> sterowniki: Linux 2.0.x

```
struct file_operations {
 int (*lseek) (struct inode *, struct file *, off_t, int);
 int (*read) (struct inode *, struct file *, char *, int);
 int (*write) (struct inode *, struct file *, const char *, int);
 int (*readdir) (struct inode *, struct file *, void *, filldir_t);
 int (*select) (struct inode *, struct file *, int, select_table *);
 int(*ioctl)(struct inode*,struct file*,unsigned int,unsigned long);
 int(*mmap)(struct inode *, struct file *, struct vm_area_struct *);
 int (*open) (struct inode *, struct file *);
 void (*release) (struct inode *, struct file *);
 int (*fsync) (struct inode *, struct file *);
 int (*fasync) (struct inode *, struct file *, int);
 int (*check_media_change) (kdev_t dev);
 int (*revalidate) (kdev_t dev);
};
```

- Struktura file_operations, której każde pole jest adresem funkcji.
- Sterownik musi (A) utworzyć egzemplarz struktury (B) wypełnić pola (nie wszystkie są obowiązkowe) adresami funkcji wykonujących odpowiednie czynności (B) zarejestrować się w systemie, podając adres stworzonej struktury.

Synchroniczne wejście-wyjście

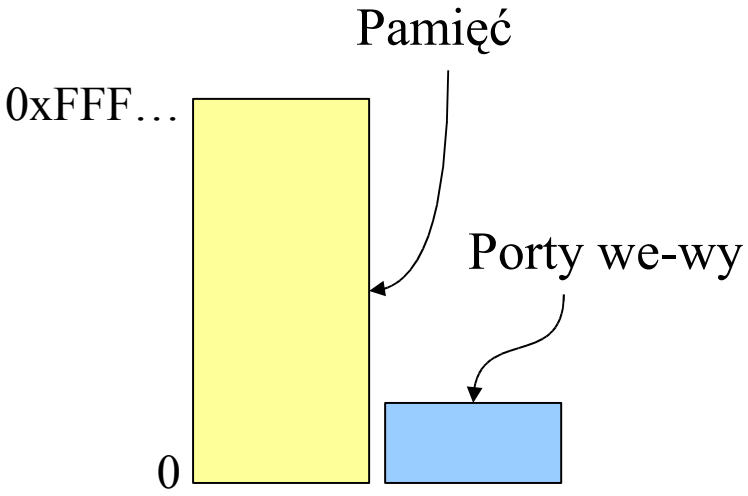


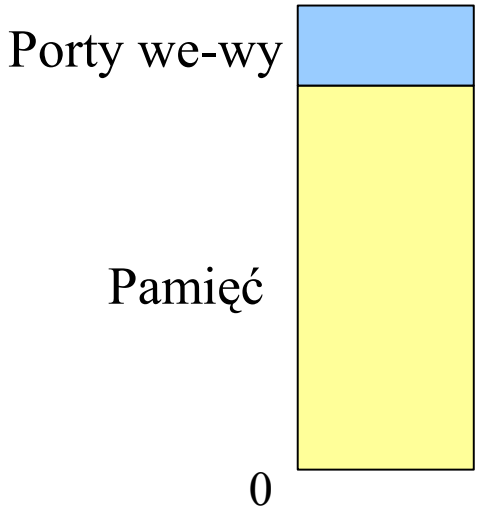
- Powrót z systemu operacyjnego po przeprowadzeniu operacji we-wy
- Proces żądający operacji we-wy jest wstrzymywany na jej czas trwania
  - W tym czasie procesor może zostać przydzielony innemu procesowi.

Różnice w prędkości transmisji urządzeń zewnętrznych

Urządzenie	Prędkość transmisji
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Printer / scanner	200 KB/sec
USB	1.5 MB/sec
Digital camcorder	4 MB/sec
Fast Ethernet	12.5 MB/sec
Hard drive	20 MB/sec
FireWire (IEEE 1394)	50 MB/sec
XGA monitor	60 MB/sec
PCI bus	500 MB/sec

Przestrzeń adresowa urządzeń we-wy

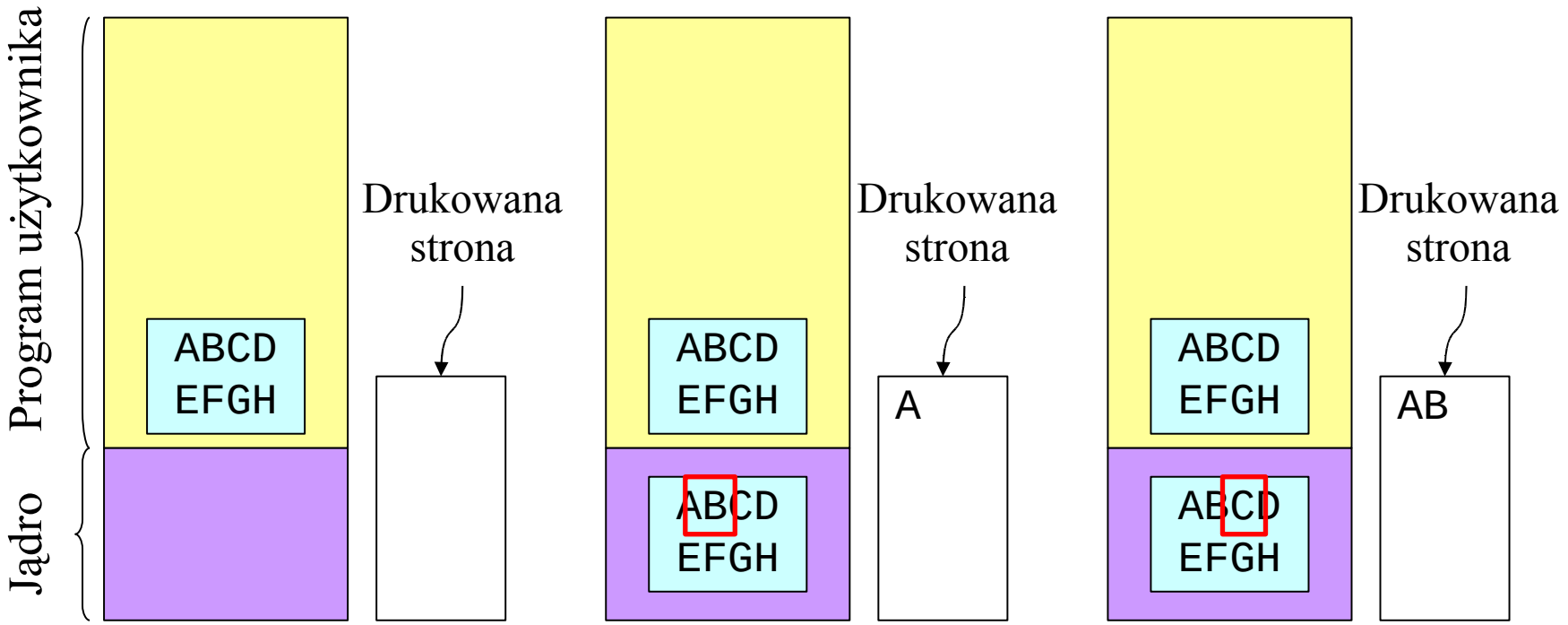




- Odrębna przestrzeń adresowa dla portów we-wy.
- Specjalne rozkazy we-wy odwołujące się do portów

- Porty we-wy w tej samej przestrzeni adresów, co pamięć.
- Dostęp do portów we-wy za pomocą tych samych rozkazów, co dostęp do pamięci

Metoda 1: Programowane wejście-wyjście (ang. Programmed Intput-Output - PIO)



- Drukowanie wiersza tekstu na drukarce.
- Odrębny bufor we-wy w pamięci jądra.

Programowe wejście-wyjście (ciąg dalszy)

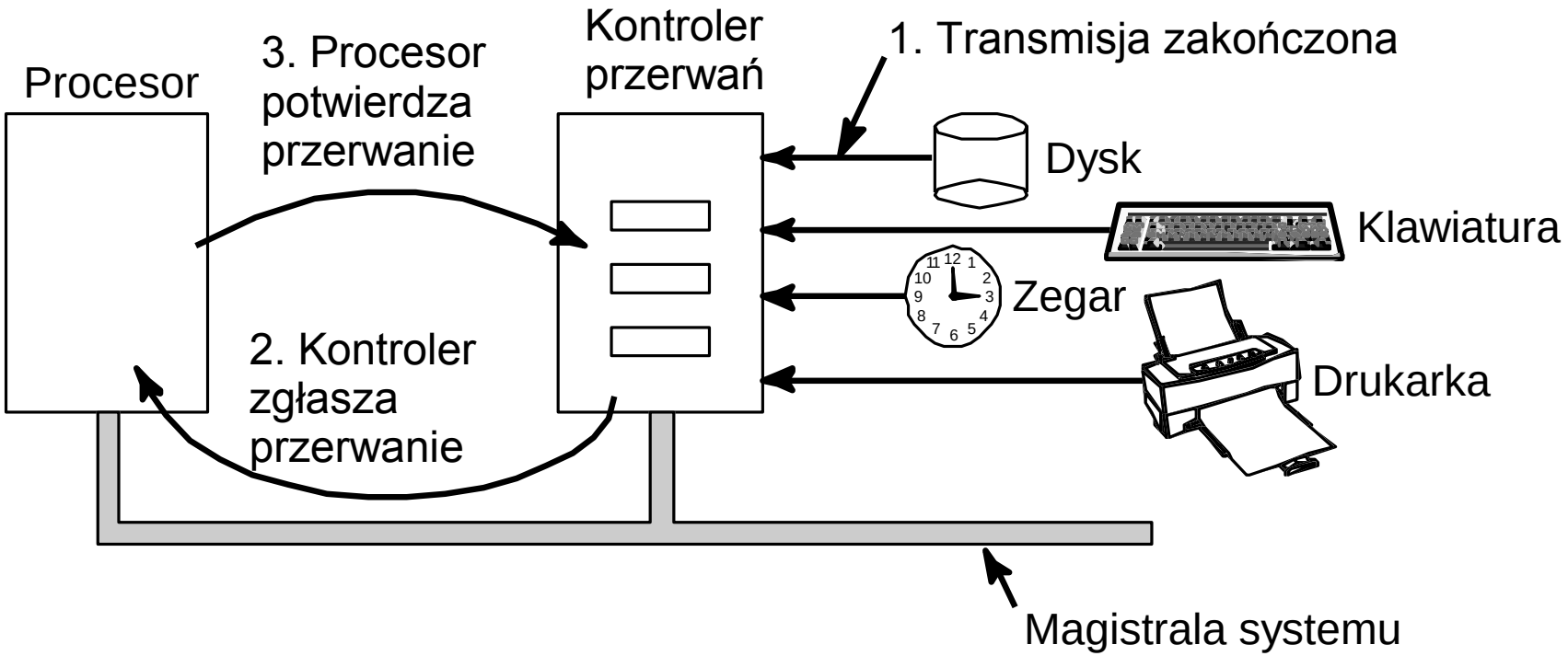
Kod wykonywany przez system operacyjny

```
copy_from_user (buffer, p, count);  // kopiuj dane do bufora jądra
for (j = 0; j < count; j++) {       // przesyłaj odrębnie każdy znak
  while (*printer_status_reg != READY)
    ;                               // czekaj aż drukarka stanie się wolna
  *printer_data_reg = p[j];         // wyślij pojedynczy znak do drukarki
}
return_to_user();                 // powrót do programu użytkownika
```

- Dwa rejestry we-wy:
  - `printer_status_reg`: Aktualny stan drukarki (czy może odebrać następny znak)
  - `printer_data_reg`: Bajt danych wysyłany do drukarki.

- Problem: bezczynne oczekiwanie procesora w pętli while
  - Drukarka jest ***znacznie*** wolniejsza od procesora

Metoda 2: Wejście-wyjście sterowane przerwaniami (ang. interrupt driven)



- Zakończenie transmisji każdego znaku (niekoniecznie pojedynczego znaku, ale o tym później) potwierdzane jest przerwaniem.

Metoda 2: Wejście-wyjście sterowane przerwaniami (ang. interrupt driven)

Kod wykonywany przez wywołanie systemowe

```
copy_from_user (buffer, p, count);
j = 0;
enable_interrupts();
while (*printer_status_reg != READY)
  ;
*printer_data_reg = p[0];
scheduler();
```

Wstrzymaj (zablokuj) bieżący proces i uruchom inny gotowy do wykonania

```
if (count == 0) {
  unblock_user();
} else {
  *printer_data_reg = p[j];
  count--;
  j++;
}
acknowledge_interrupt();
return_from_interrupt();
```

Odblokuj zablokowany proces (zablokowany proces może powrócić do kodu użytkownika)

Procedura obsługi przerwania

- Urządzenie zgłasza przerwanie po odebraniu każdego bajtu.

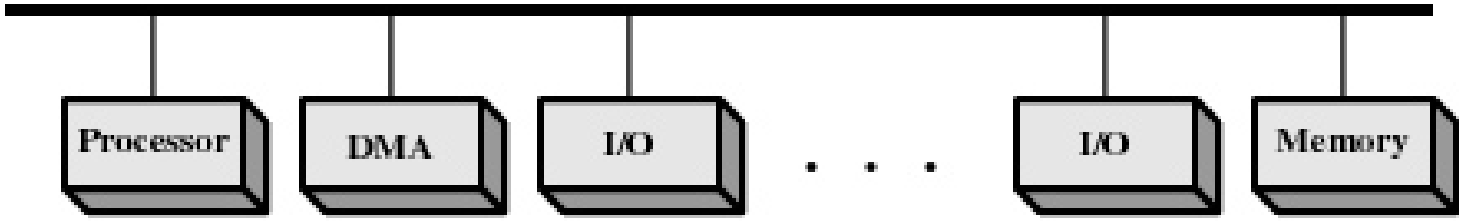
Wejście-wyjście sterowane przerwaniami

- **Zaleta:** Możliwość wykorzystania procesora w czasie przeprowadzania wejścia-wyjścia
- **Wada:** Większy narzut związany z przesyłaniem jednego bajtu (przejście do procedury, obsługa, potwierdzenie, powrót przerwania)
- Gdy szybkość procesora jest znacznie większa od szybkości urządzenia, nie stanowi to problemu. Niech obsługa przerwania zajmuje minimum 2 us.
  - Przyjmijmy, że wolna drukarka zgłasza przerwania co 1 ms (prędkość transmisji 1000 bajtów/s), w takim przypadku obsługa przerwań obciąża procesor w **0.2%.**
  - Jeżeli szybki dysk zgłasza przerwania co 4us (prędkość transmisji 250 000 bajtów/s), to obsługa przerwań obciąża procesor w **50%.**
  - Gdy prędkość transmisji przekracza 500 000 bajtów na sekundę wykorzystanie tej metody (przerwanie po każdym bajcie) nie jest możliwe.
  - W takiej sytuacji tryb PIO może pozwolić na szybszą transmisję danych.
  - Możemy próbować łączyć tryb PIO z buforowaniem po stronie kontrolera, i ze zgłaszaniem przerwania po przesłaniu całego sektora.
- **Pytanie:** Czy musimy angażować procesor do przesyłania danych pomiędzy urządzeniem wewnętrznym i pamięcią ?

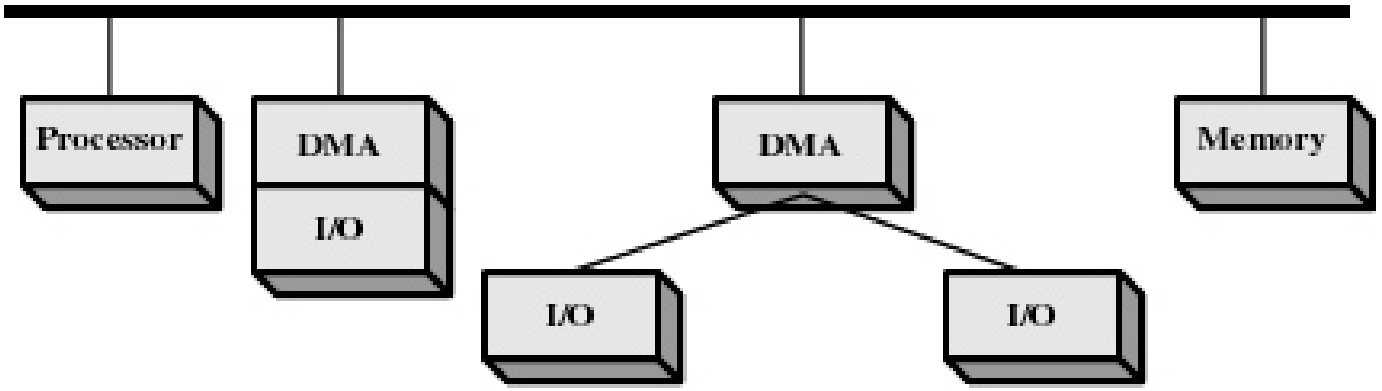
Metoda 3: Bezpośredni dostęp do pamięci (Direct Memory Access - DMA)

- Przesyłanie **bloku** danych pomiędzy urządzeniem a pamięcią odbywa się bez angażowania procesora.
  - Przesłanie bloku poprzez DMA jest współbieżne z pracą procesora.
  - Gdy nadejdzie czas przesłania kolejnego bajtu urządzenie DMA przejmuje (na chwilę) od procesora kontrole nad magistralą – ang. **cycle stealing.**
  - Następuje transmisja bajtu pomiędzy urządzeniem i pamięcią. W tym czasie procesor ma zablokowany dostęp do magistrali. (Nie musi to oznaczać zatrzymania pracy procesora – jeżeli posiada on pamięć podręczną)
  - Po przesłaniu bajtu urządzenie DMA zwalnia magistralę.
- Przerwanie generowane jest po przesłaniu kompletnego bloku danych.
- Bardzo mały narzut związany z przesłaniem jednego bajtu.
- Metoda wykorzystywana w sytuacji, w której szybkość urządzenia zewnętrznego jest zbliżona do szybkości pamięci.

Organizacja DMA

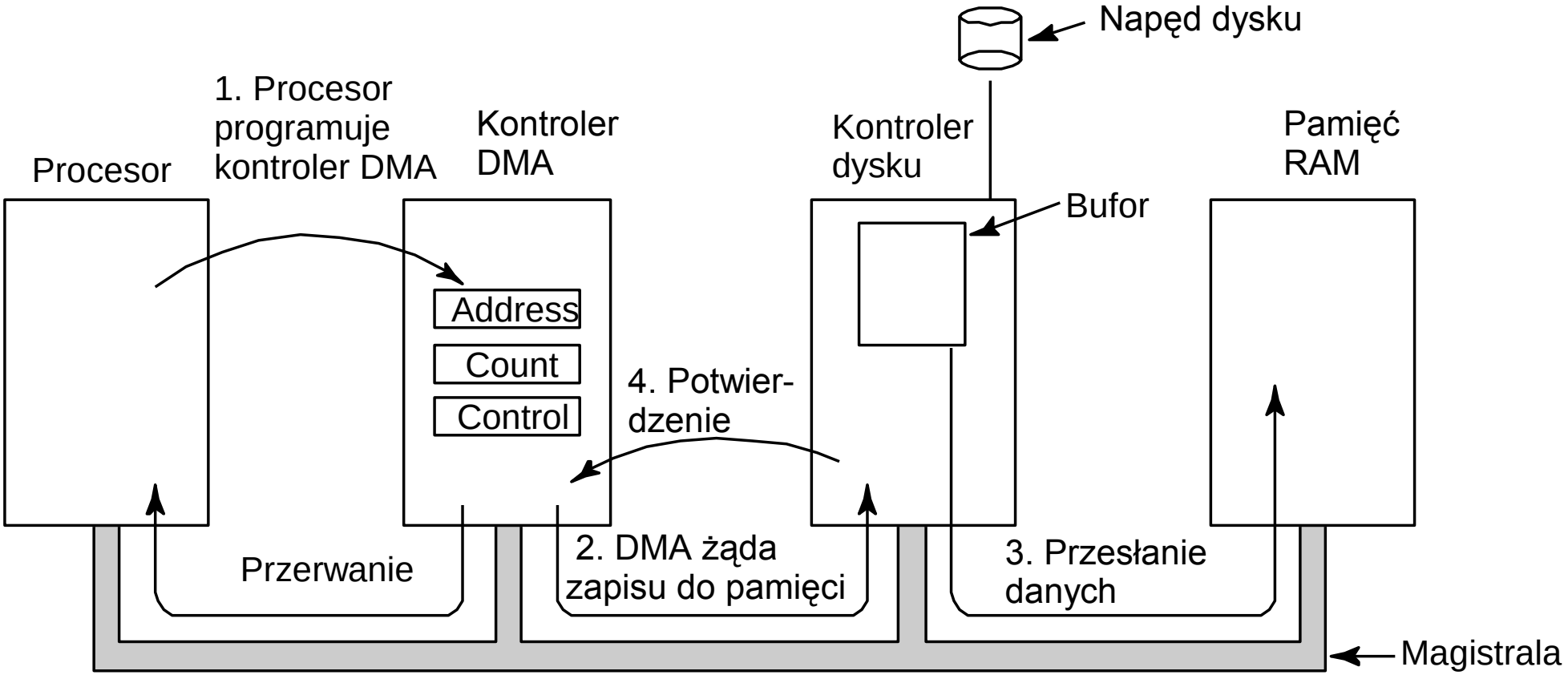


- (a) DMA jest kolejnym modułem podłączonym magistrali
  - Transfer może wymagać dwóch cykli magistrali
  - Starsze systemy (Z80, IBM PC AT)



- (b) DMA zintegrowane z kontrolerem urządzenia
  - Nowsze systemy (magistrala PCI).

Operacja z wykorzystaniem DMA



- Address – gdzie przesłać dane, Count – ile przesłać, Control – rodzaj operacji (np. odczyt zapis).
- Kontoler DMA zajmuje się zliczaniem bajtów, podawaniem adresu i sygnałów sterujących na magistralę (punkty 2,3,4)

Operacja we-wy z wykorzystaniem DMA

Kod wykonywany przez wywołanie systemowe

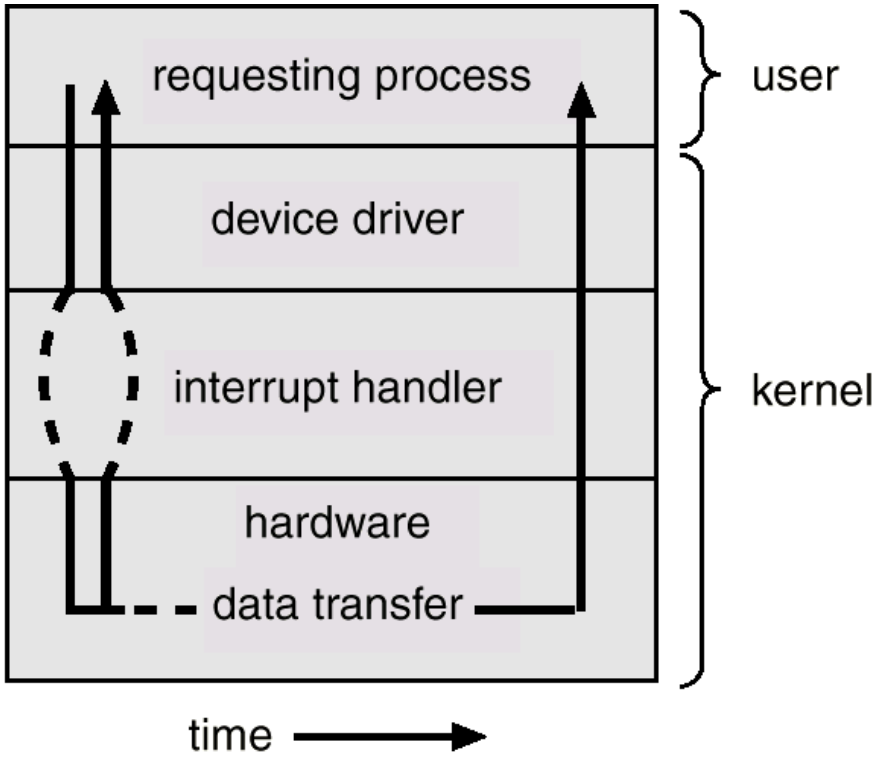
```
copy_from_user (buffer, p, count);
set_up_DMA_controller();
scheduler(); // wstrzymaj aktualny proces
             // i przekaż procesor innemu
```

Kod wykonywany przez procedurę obsługi przerwania

```
acknowledge_interrupt();
unblock_user(); // odblokuj czekający proces
return_from_interrupt();
```

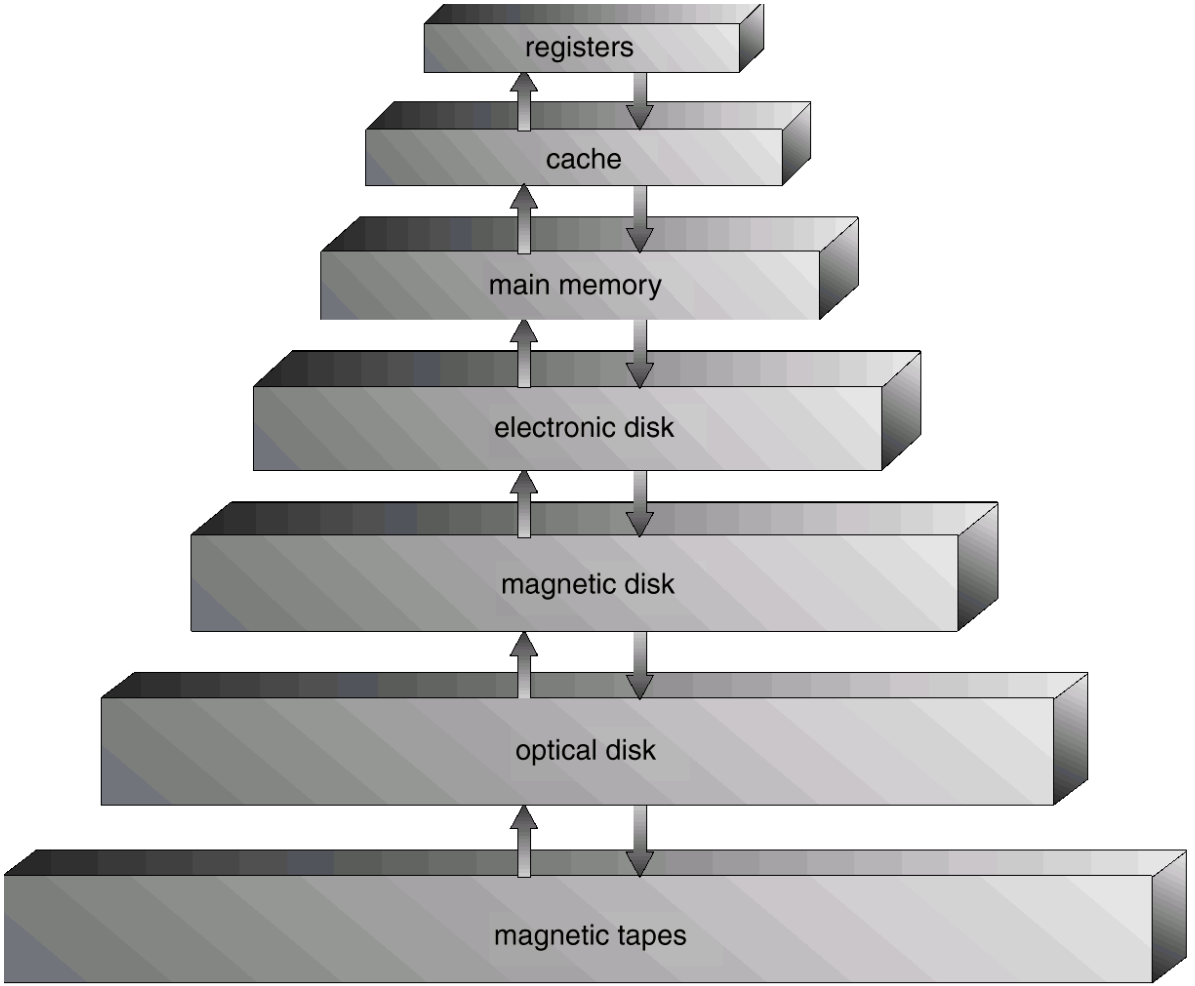
- `set_up_DMA_controller()` - zaprogramuj DMA (numer urządzania, adres bufora w pamięci, liczba bajtów)
- DMA zajmuje się zliczaniem bajtów i oczekiwaniem na gotowość urządzenia

Asynchroniczne wejście-wyjście



- Powrót z systemu operacyjnego po **zainicjalizowaniu** operacji we-wy
- Proces co jakiś czas sprawdza czy operacja się zakończyła
  - W tym rozwiązaniu w czasie trwania operacji we-wy możliwe jest wykorzystanie procesora przez ten sam proces.

Hierarchia pamięci



- Przemieszczając się w dół hierarchii
  - Zwiększamy czas dostępu
  - Zmniejszamy koszt jednego bajtu

Wykorzystanie pamięci podręcznych (ang. caching)

- Wykorzystanie szybkiej pamięci do przechowywania najczęściej używanych danych.
  - Pamięć podręczna procesora.
  - Pamięc podręczna dysku.
- Wymaga wprowadzenie polityki zarządzania pamięcią podręczną.
- Problem spójności pamięci podręcznej: (ang. cache coherency) Informacja przechowywana w pamięci podręcznej niezgodna z informacją przechowywaną w pamięci głównej
  - Przykład 1. System dwuprocesorowy. Każdy procesor ma własną pamięć podręczną. Zawartość jednej komórki pamięci przechowywana w obydwu pamięciach podręcznych. Procesor A zapisuje tę komórkę, Procesor B próbuje odczytu .....
  - Przykład 2. Pamięć podręczna dysku. Zmodyfikowana zawartość pewnych sektorów dysku jest przechowywana przez pewien czas w pamięci operacyjnej zanim zostanie zapisana fizycznie na dysk. Jeżeli w tym czasie nastąpi załamanie systemu .....

Mechanizmy ochrony (ang. protection)

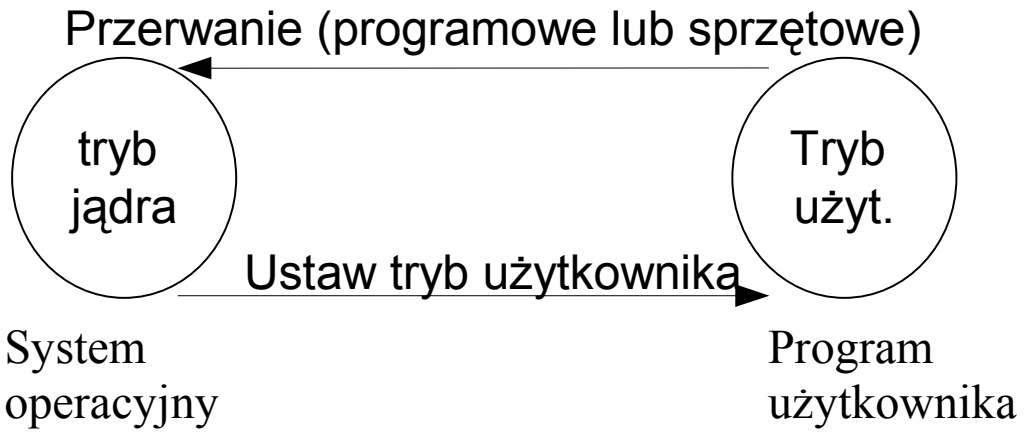
- Potrzeba zapewnienia, że "źle sprawujący się program" nie zakłóci pracy innych programów i samego systemu operacyjnego. Program użytkownika nie może być w stanie wykonać pewnych operacji.

- Przykłady "złego zachowania się programu"
  - Bezpośrednia komunikacja z urządzeniami wejścia-wyjścia => **ochrona we-wy**
  - Dostęp do pamięci należącej do innych procesów lub do systemu => **ochrona pamięci**
  - Zablokowanie przerwań, zmiana wektora przerwań => **ochrona systemu przerwań**
  - Nieskończona pętla => **ochrona procesora**

- Program użytkownika nie ma prawa wykonać żadnej z powyższych operacji !!!

Podwójny tryb pracy

- Procesor może wykonywać instrukcje w jednym z dwóch trybów
  - Tryb jądra (instrukcje wykonywane przez system operacyjny)
  - Tryb użytkownika (instrukcje wykonywane przez program użytkownika)



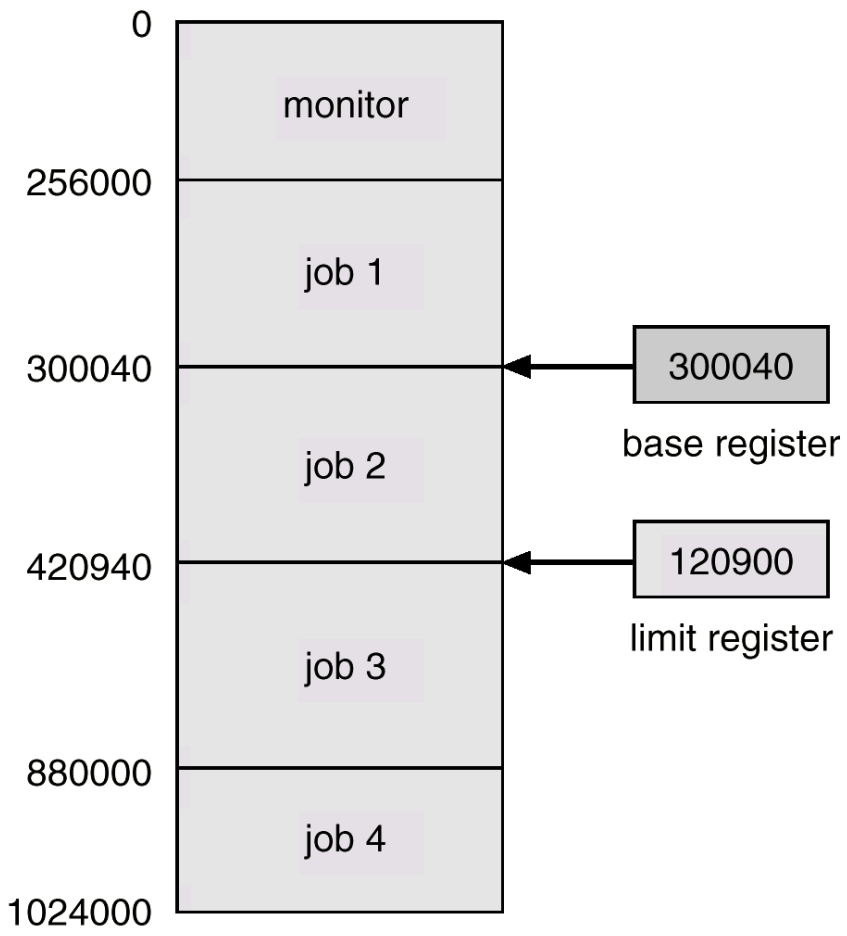
- **Instrukcje uprzywilejowane** – mogą być wywoływane wyłącznie w trybie jądra
  - Próba ich wykonania w trybie użytkownika powoduje przerwanie i przejście do systemu operacyjnego

Mechanizmy ochrony

- Instrukcje uprzywilejowane
  - Instrukcje do komunikacji z urządzeniami we-wy
  - Blokowanie/odblokowanie przerwań
  - Zmiana wektora przerwań
    - Aby zapewnić, że program użytkownika nigdy nie wykona się w trybie jądra

- Przerwanie zegara:
  - gwarantuje ochronę procesora przed programem użytkownika z nieskończoną pętlą

Przykład realizacji ochrony pamięci: rejestry bazowy i limitu



- Rejestry te określają zakres dopuszczalnych adresów procesu.
  - B – rejestr bazowy (ang. base)
  - L – rejestr limitu
  - A – adres pamięci, do którego odwołuje się program
- Jeżeli $B \leq A < B+L$ => w porządku
- W przeciwnym wypadku generuj przerwanie (obsługiwane przez system operacyjny)
- Rozkazy zmieniające wartość rejestrów bazowego i limitu są rozkazami uprzywilejowanymi.