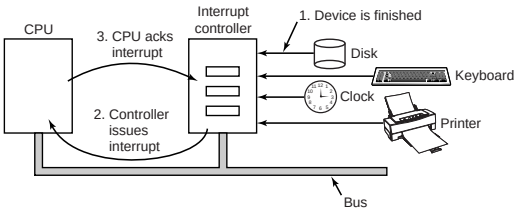
Systemy operacyjne
Obsługa wejścia/wyjścia

[2] Obsługa przerwań - powtórka



- po zakończeniu zlecenia urządzenie zewnętrzne generuje przerwanie – ustawienie sygnału na odpowiedniej linii,
- sygnał wykrywany przez kontroler przerwań, kontroler w zależności od aktualnego stanu i priorytetu urządzenia wstrzymuje bądź natychmiast generuje sygnał przerwania do procesora wystawiając na magistrali adresowej numer przerwania,
- procesor wykonuje program ustalony odpowiednim adresem z przypisanego danemu przerwaniu wektora przerwań,
- zazwyczaj procedura obsługi wysyła do kontrolera kod zwolnienia zajętości kontrolera przerwań dopuszczając obsługę kolejnych przerwań.

[3] Przerwania w nowoczesnych architekturach

- problem przetwarzania potokowego i superskalarnego,

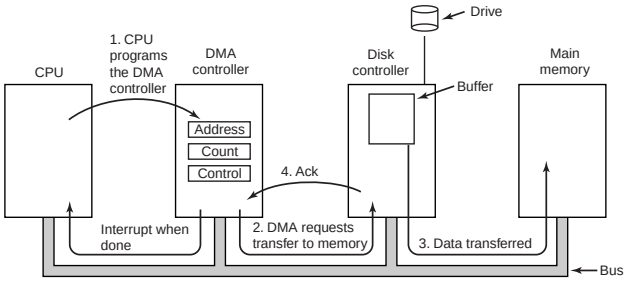
Przerwaniem precyzyjnym (ang. *precise interrupt*) nazywa się przerwanie, które pozostawia system w dobrze zdefiniowanym stanie. Implikuje następujące własności:

1. ustalona jednoznacznie wartość licznika rozkazów (LR),
2. wszystkie instrukcje przed LR w pełni wykonane,
3. żadna instrukcja po LR nie została wykonana,
4. znany stan wykonania instrukcji wskazywanej aktualnie przez LR.

- pozostałe przerwania określa się mianem **przerwań nieprecyzyjnych** (ang. *imprecise interrupts*).
- przerwania nieprecyzyjne wymagają większego wykorzystania stosu, są źródłem wolniejszej obsługi.

- przerwania precyzyjne wymagają dużo bardziej złożonej logiki procesora.

[4] **Bezpośredni dostęp do pamięci (DMA)**



- moduły DMA kontrolują wymianę danych pomiędzy pamięcią główną a urządzeniami zewnętrznymi,
- wykorzystanie wolnego czasu magistrali bądź (zazwyczaj) wstrzymywanie procesora, na jeden cykl co jakiś czas, w związku z przesyłaniem danych po przez magistralę (po jednym słowie),
- praca procesora przerywana jednorazowo po przesłaniu całego bloku danych, nie ma potrzeby przełączania kontekstu w trakcie transferu.

[5] **Obługa wejścia/wyjścia**

Podział typów urządzeń zewnętrznych:

- **urządzenia blokowe**, możliwy odczyt/zapis każdego bloku niezależnie,
- **urządzenia znakowe**, łańcuch znaków bez podziału na bloki, nie ma adresowalności ani ustawianego wskaźnika bieżącej pozycji,
- czasem, ze względu na specyfikę, wyróżnia się jako osobną klasę **urządzenia sieciowe/komunikacyjne**,
- niektóre urządzenia nie pasują do powyższej klasyfikacji, na przykład *czasomierze* (ang. *timers*),

[6] **Różnice w zarządzaniu we/wy**

Różnice w zarządzaniu urządzeniami zewnętrznymi:

- złożoność obsługi,

- wymóg dodatkowego wsparcia sprzętowego,
- rozróżnianie priorytetów,
- jednostka przepływu,
- reprezentacja danych,
- reakcja urządzeń i obsługa błędów,
- metoda komunikacji,
- metoda oprogramowywania.

[7] **Cele oprogramowania we/wy**

- niezależność obsługi ogólnej od specyfiki urządzenia,
- ujednolicenie nazewnictwa,
- obsługa błędów – w im niższej warstwie tym lepiej,
- metoda przesyłania - blokująca/nieblokująca, synchroniczna/asynchroniczna,
- buforowanie.

[8] **Poziomy obsługi urządzeń zewnętrznych**

W systemie komputerowym wyróżnia się następujące poziomy obsługi:

1. **poziom fizyczny**, bezpośrednia manipulacja rejestrami urządzeń zewnętrznych, obsługa przerwań, inicjowanie transmisji,
2. **poziom wywołań systemowych** - komunikacja z programi obsługi urządzeń zewnętrznych,
3. **poziom usług** - oferuje jednolitą metodę dostępu.

[9] **Komunikacja z urządzeniami zewnętrznymi (I)**

Jak procesor komunikuje się z rejestrami kontrolnymi i jak odwołuje się do buforów danych urządzenia zewnętrznego. Dwie techniki komunikacji:

1. **porty we/wy**, z każdym rejestrem kontrolnym skojarzony port o ustalonym numerze. Komunikacja przez specjalne instrukcje:

```
IN   REG, PORT
OUT  PORT, REG
```

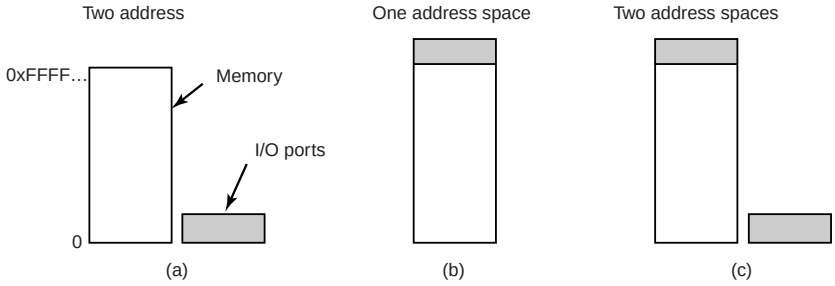
2. **we/wy odwzorowywane w pamięci**

   - sterownik może być w całości napisany w C, nie wymaga wstawek asemblerowych, bo nie ma specjalnych instrukcji,
   - nie wymaga dedykowanego mechanizmu ochrony,
   - szybsze testowanie zawartości rejestrów kontrolnych,

   ale

   - wymaga wyłączenia cache dla regionu odwzorowania,
   - komplikuje architekturę rozwiązań z szynami wielu typów.

[10] **Komunikacja z urządzeniami zewnętrznymi (II)**



a. przestrzenie rozłączne, porty wejścia/ wyjścia,

b. odzworowanie w adresy pamięci,

c. podejścia mieszane, np. w Pentium: adresy 640kB - 1MB zarezerwowane dla urządzeń zewnętrznych a ponadto porty we/wy 0 – 64K.

[11] **Programowanie urządzeń zewnętrznych**

Metody programowania urządzeń zewnętrznych

1. programowalne we/wy (ang. *polling, busy waiting*),

2. programowanie z wykorzystaniem przerwań (ang. *interrupt-driven*),

3. z wykorzystaniem DMA.

[12] **Programowalne we/wy**

```
copy_from_user(buffer, p, count);                /* p is the kernel bufer */
for (i = 0; i < count; i++) {                    /* loop on every character */
      while (*printer_status_reg != READY) ;     /* loop until ready */
      *printer_data_register = p[i];             /* output one character */
}
return_to_user( );
```

Przykład: pisanie łańcucha znaków na drukarkę

[13] **Programowanie przerwaniowe we/wy**

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

(a)

```
if (count == 0) {
     unblock_user( );
} else {
     *printer_data_register = p[i];
     count = count – 1;
      i = i + 1;
}
acknowledge_interrupt( );
return_from_interrupt( );
```

(b)

Przykład: pisanie łańcucha znaków na drukarkę

a. kod obsługi funkcji systemowej,

b. właściwa obsługa przerwania.

[14] **Programowanie z wykorzystaniem DMA**

```
copy_from_user(buffer, p, count);
set_up_DMA_controller( );
scheduler( );
```

(a)

```
acknowledge_interrupt( );
unblock_user( );
return_from_interrupt( );
```

(b)

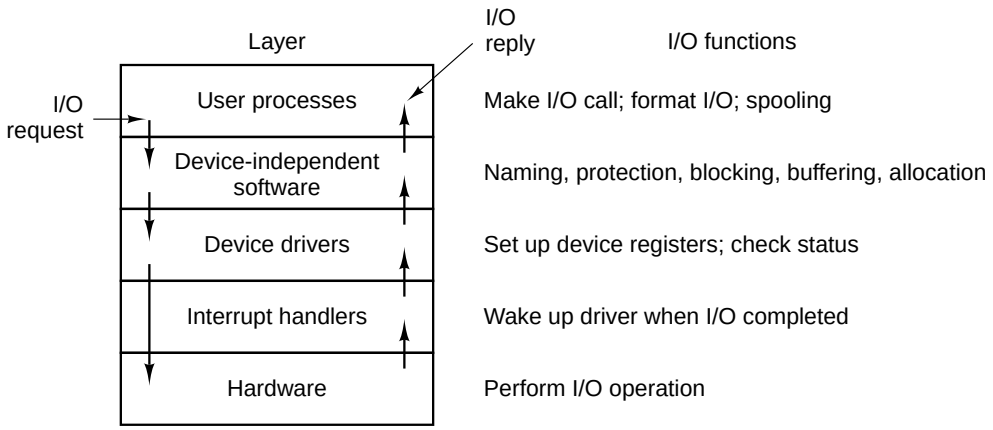
Przykład: pisanie łańcucha znaków na drukarkę

a. kod obsługi funkcji systemowej,

b. właściwa obsługa przerwania,

- redukcja liczby przerwań z jednego na drukowany znak do jednego na wydruk bufora,
- nie zawsze najlepsza metoda – kwestie rozmiaru zakresu i względnej szybkości procesora i kontrolera DMA.

[15] **Struktura warstwowa systemu we/wy**



[16] **Warstwa niezależna od urządzeń**

- ukrycie specyfiki poszczególnych urządzeń zbliżonego typu,
- nazywanie urządzeń z wykorzystaniem numerów **major** i **minor**,
- ochrona urządzeń przed nieautoryzowanym dostępem,
- obsługa różnych rozmiarów bloków różnych urządzeń,
- udostępnianie mechanizmów buforowania (tzw. programowy cache),
- zarządzanie dostępnością urządzeń blokowych,
- zarządzanie przydziałem urządzeń użytkownikom,

- część systemu obsługi błędów.

[17] **Wydajność dostępu do dysku**

- **czas opóźnienia** (ang. *seek time*) czas ustawienia głowicy na docelowej ścieżce,
- **opóźnienie rotacyjne** (ang. *rotational delay/latency*) czas ustawienia głowicy na początku docelowego sektora,
- **czas dostępu** (ang. *access time*) czas wyszukiwania + opóźnienie rotatcyjne
- czas wyszukiwania decyduje o wydajności,
- duża rola dyskowej pamięci podręcznej (algorytmy wymiany LRU, LFU).

[18] **Algorytmy szeregowania dostępu do dysku**

Ze względu na zlecającego:

RSS (ang. *random scheduling*) losowy,

FIFO najbardziej sprawiedliwy,

PRI priorytetowy, z szeregowaniem realizowanym zewnętrznie,

LIFO (ang. *Last In First Out*) maksymalizacja lokalności i wykorzystania zasobów.

Ze względu na zlecenie:

SSTF (ang. *shortest service time first*) z najmniejszym ruchem ramienia,

SCAN algorytm windy, ramię przesuwa się w dół i w górę obsługując zlecenia,

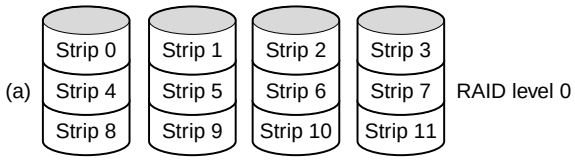
C-SCAN cykliczny SCAN, ramię przesuwa się w jednym kierunku z szybkim nawrotem,

[19] **Redundancja w zarządzaniu dyskami**

RAID (ang. *Redundant Array of Independent Disks*) – nazwa i klasyfikacja podana przez pracowników Uniwersytetu w Berkeley.

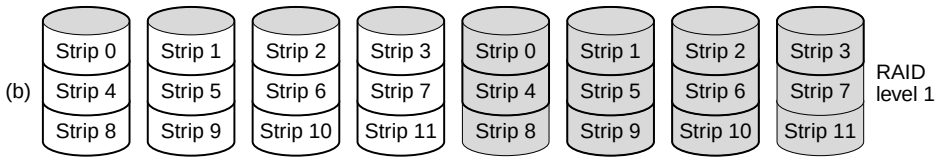
- technika tworzenia dysku wirtualnego o ustalonych własnościach związanych z niezawodnością, wydajnością i administrowalnością z grupy niezależnych dysków fizycznych,
- dane dystrybuowane na dyski macierzy,
- wykorzystanie redundancji w celu zwiększenia odporności na uszkodzenia, w szczególności odporności na uszkodzenia poszczególnych dysków.

[20] **Rozwiązania RAID (RAID 0)**



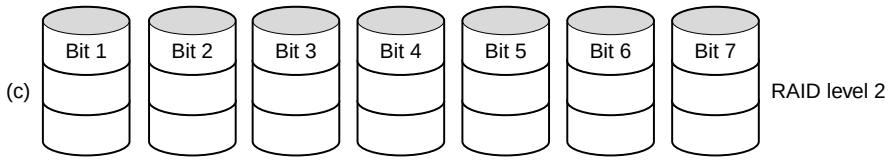
- nie ma redundancji danych,
- podział na **konkatenację** (ang. *concatenation*) i **paskowanie** (ang. *stiping*),
- zwiększenie wydajności i elastyczności zarządzania, rozwiązanie oszczędne, brak odporności na uszkodzenia.

[21] **Rozwiązania RAID (RAID 1)**



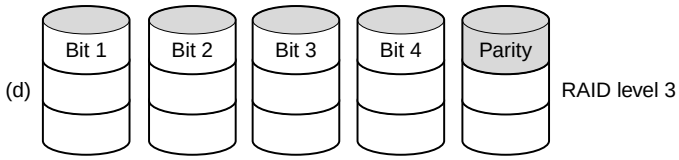
- **odbicie lustrzane** (ang. *mirroring*), pełna redundancja danych,
- z punktu widzenia niezawodności najlepsze rozwiązanie,
- rozwiązanie kosztowne.

[22] **Rozwiązania RAID (RAID 2)**



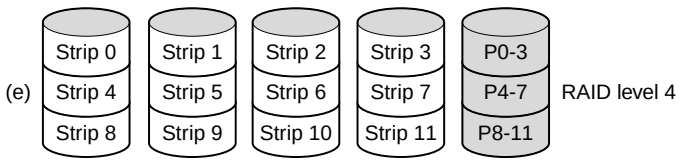
- kod korekcyjny wyznaczany w analizie poszczególnych bitów,
- wykorzystanie kodów korekcyjno-detekcyjnych (kod Hamminga),
- rozwiązanie kosztowne wydajnościowo i pamięciowo.

[23] **Rozwiązania RAID (RAID 3)**



- rozwiązanie analogiczne do RAID 2, z bitem parzystości zamiast kodu korekcyjno-detekcyjnego,
- duża wydajność w kontekście transferu, mała w kontekście przepustowości obsługi zapytań.

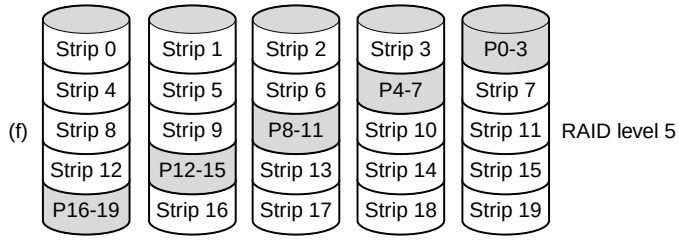
[24] **Rozwiązania RAID (RAID 4)**



- RAID 4 – RAID 6, niezależny dostęp do poszczególnych dysków, zlecenia mogą być obsługiwane niezależnie równolegle, lepsza wydajność w kontekście przepustowości obsługi zapytań,
- paskowanie z dużymi paskami,

- parzystość wyliczana na poziomie bitu, ale wymaga odczytu bloków.

[25] **Rozwiązania RAID (RAID 5)**



- rozwiązanie analogiczne do paskowania z dodanym bitem parzystości,
- rozwiązanie ekonomiczne - redundancja kosztuje jeden dysk,
- dobra wydajność odczytu, istotna degradacja wydajności zapisu,
- jakość rozwiązania determinowana odpowiednimi wartościami parametrów konfiguracyjnych wpływających na wydajność.

[26] **RAID - aspekty dodatkowe**

- RAID 6, jak RAID 5 z dwoma niezależnie rozmieszczanymi sumami kontrolnymi,
- RAID 10 = 1 + 0
- rozwiązania sprzętowe a rozwiązania programowe,
- stosowane produkcyjne poziomy RAID: 0, 1, 5, 1+0, 0+1,
- typowa konfiguracja w zastosowaniach:
  - RAID 1 dla małych danych krytycznych (np. dyski systemu operacyjnego),
  - RAID 5 dla dużych danych produkcyjnych (np. dyski podłączonej macierzy).