

Systemy Operacyjne

Joachim J. Włodarz

E-mail: jjw@us.edu.pl

Wykład

(C) JJW

□ 01 – Wstęp

- a – Koncepcja systemu operacyjnego
- b – Rozwój systemów operacyjnych – szkic historyczny
- c – Rodzaje systemów operacyjnych – krótki przegląd
- d – System operacyjny a sprzęt
- e – Podstawowe struktury i funkcjonowanie systemu operacyjnego

□ 02 – Procesy i wątki

- a – Procesy
- b – Wątki
- c – Komunikacja międzyprocesowa
- d – Szeregowanie zadań

□ 03 – Dostęp do zasobów

- a – Koncepcja zasobu
- b – Problem zakleszczenia
- c – Metody postępowania

□ 04 – Zarządzanie pamięcią

- a – Podstawy
- b – Wymiana i stronicowanie
- c – Pamięć wirtualna
- d – Segmentacja
- e – Zagadnienia implementacyjne

□ 05 – Wejście/Wyjście

- a – Podstawy – aspekty sprzętowe i programowe
- b – Warstwowa struktura obsługi urządzeń wejścia/wyjścia
- c – Urządzenia blokowo–zorientowane (dyski, etc.)
- d – Urządzenia znakowo–zorientowane (terminale, etc.)
- e – Graficzne interfejsy użytkownika
- f – Sieci komunikacyjne

□ 06 – Systemy plikowe

- a – Pliki – struktura, organizacja, etc.
- b – Struktury katalogowe
- c – Implementacja systemu plikowego
- d – Przykładowe systemy plikowe

□ 07 – Multimedia i ich przetwarzanie

- a – Podstawy
- b – Pliki i dane multimedialne
- c – Kompresja/dekompresja danych multimedialnych
- d – Zagadnienia związane z systemowym szeregowaniem zadań
- e – Zagadnienia związane z obsługą pamięci podręcznej

□ 08 – Systemy wieloprocessorowe

- a – Podstawy
- b – Zagadnienia związane z obsługą sprzętu
- c – Systemowa obsługa wieloprocessorowości
- d – Multikomputery i systemy rozproszone

□ 09 – Bezpieczeństwo i ochrona danych

- a – Koncepcja bezpiecznego środowiska pracy
- b – Podstawy kryptografii
- c – Autentyfikacja i autoryzacja
- d – Zagrożenia wewnętrzne i zewnętrzne
- e – Zarządzanie dostępem i ochrona danych
- f – Zaufane systemy operacyjne

□ 10 – UNIX i systemy pokrewne

- a – Przegląd historyczny
- b – Obsługa i funkcjonowanie procesów i wątków
- c – Zarządzanie pamięcią
- d – Obsługa urządzeń wejścia/wyjścia
- e – Systemy plikowe
- f – Zagadnienia związane z bezpieczeństwem i ochroną danych

□ 11 – Windows NT i systemy pokrewne

- a – Przegląd historyczny
- b – Ogólna struktura systemu
- c – Podstawowe podsystemy i emulowane środowiska pracy
- d – Koncepcja i obsługa procesów i wątków
- e – Zarządzanie pamięcią
- f – Obsługa urządzeń wejścia/wyjścia
- g – Systemy plikowe
- h – Zagadnienia związane z bezpieczeństwem i ochroną danych

□ 12 – Systemy operacyjne klasy mainframe

- a – Zagadnienia ogólne
- b – Systemy zorientowane wsadowo: MVS, OS/390, zOS
- c – Systemy maszyn wirtualnych: VM/CMS, zVM
- d – Systemy klastrowe: OpenVMS

□ 13 – Projektowanie systemu operacyjnego

- a – Zagadnienia ogólne związane z naturą problemu
- b – Projektowanie i implementacja interfejsu systemowego
- c – Zagadnienia związane z wydajnością pracy
- d – Współczesne trendy rozwojowe systemów operacyjnych

Laboratorium

- Komplementarna część w stosunku do wykładu
- Cele:
 - uzupełnienie wykładu o elementy praktyczne
 - możliwie pełne zapoznanie z jednym, wybranym systemem
 - zapoznanie z innymi, typowymi systemami/środowiskami
 - uzyskanie odpowiednich umiejętności praktycznych

Literatura

- podstawowy podręcznik o charakterze ogólnym:

- A. Sliberschatz, P. B. Galvin: "Podstawy systemów operacyjnych", WNT (2006).

- podręczniki pomocnicze:

- A. Tanenbaum: "Modern Operating Systems", 2nd Ed, Prentice Hall (2004).
 - B. DuCharme: "The Operating Systems Handbook", McGraw–Hill (1994)
(wersja online jest aktualizowana na bieżąco)

- literatura uzupełniająca

- wg. potrzeb, odnośniki zostaną podane na zajęciach

- online: <http://www.mfc.us.edu.pl/inf>

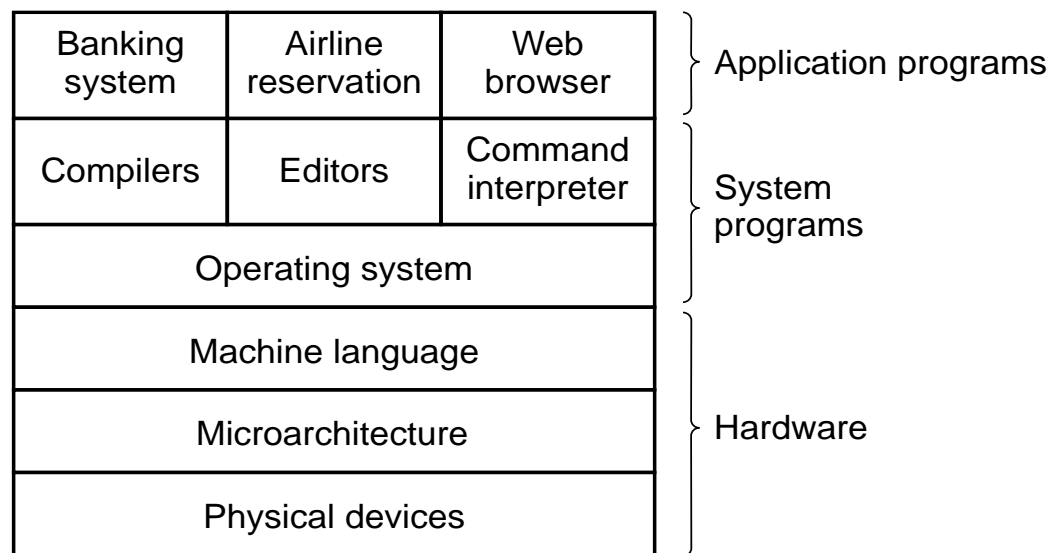
01 – Wstęp

- ☐ Co to jest system operacyjny ?
- ☐ Historia systemów operacyjnych
- ☐ Rodzaje systemów operacyjnych
- ☐ Systemy komputerowe – aspekty sprzętowe
- ☐ System operacyjny – podstawowe pojęcia
- ☐ Odwołania systemowe
- ☐ Struktura systemu operacyjnego

>>>

Struktura systemu komputerowego

- ☐ sprzęt
- ☐ oprogramowanie systemowe
- ☐ oprogramowanie aplikacyjne



>>>

Co to jest system operacyjny ?

- "rozszerzenie" sprzętu

- ukrywa przed użytkownikiem detale sprzętowe
- udostępnia użytkownikowi "maszynę wirtualną"

- zarządca zasobów

- przydział dostępu do zasobu ("czas")
- przydział odpowiedniej wielkości zasobu ("przestrzeń")

>>>

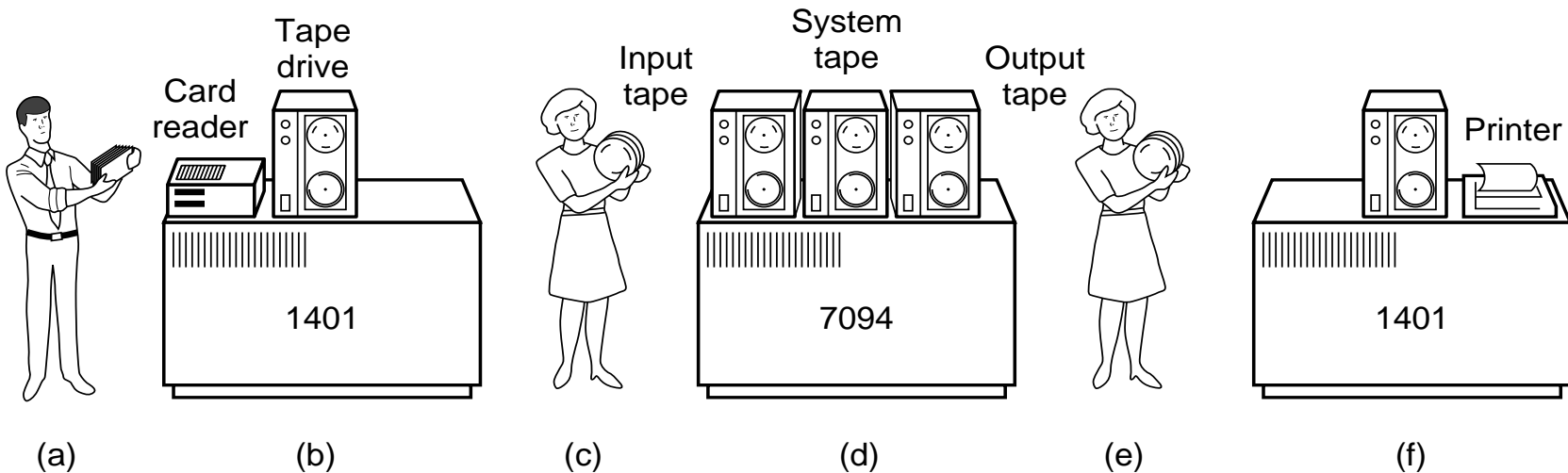
Historia systemów operacyjnych (1)

- pierwsza generacja (1945–1955)
 - lampy próżniowe, obsługa manualna
- druga generacja (1955–1965)
 - tranzystory, systemy przetwarzania wsadowego
- trzecia generacja (1965 – 1980)
 - układy scalone, systemy wieloprogramowe
- czwarta generacja (1980 – ?)
 - komputery osobiste, systemy odpowiednie do zastosowań

>>>

Historia systemów operacyjnych (2)

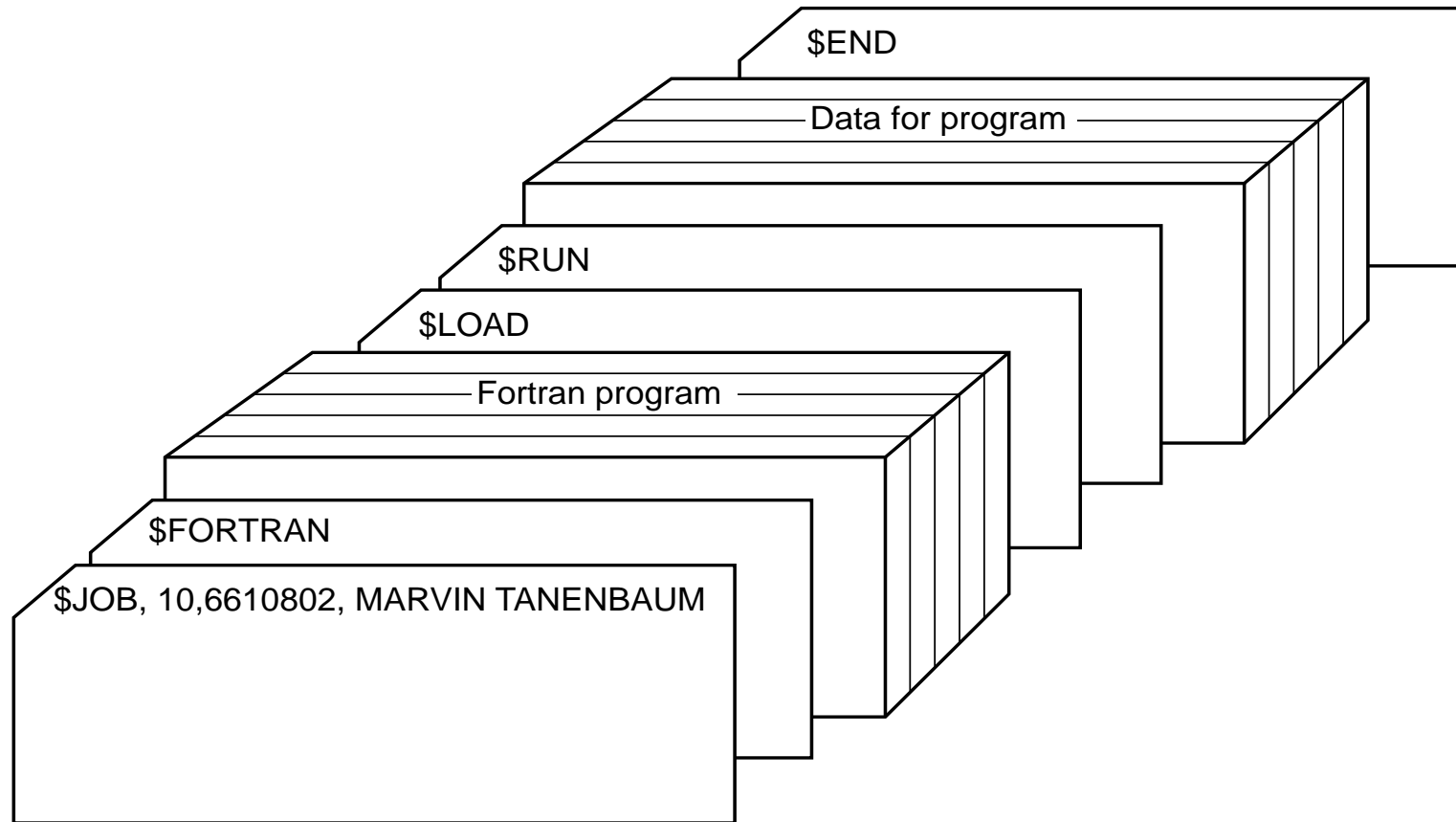
- wczesne systemy przetwarzania wsadowego
 - (a) – użytkownik
 - (b) (f) – systemy pomocnicze (satelitarne)
 - (c) (e) – operator systemu
 - (d) – system mainframe



>>>

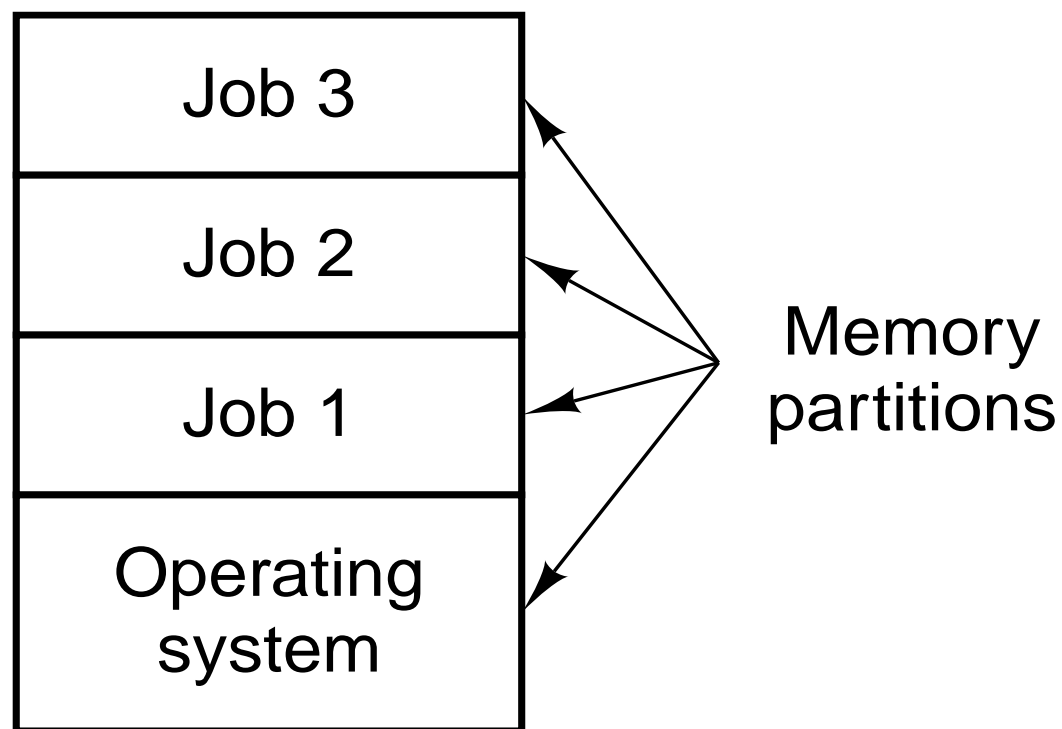
Historia systemów operacyjnych (3)

- typowe zadanie wsadowe (FMS)



Historia systemów operacyjnych (4)

- systemy wieloprogramowe/wielozadaniowe



>>>

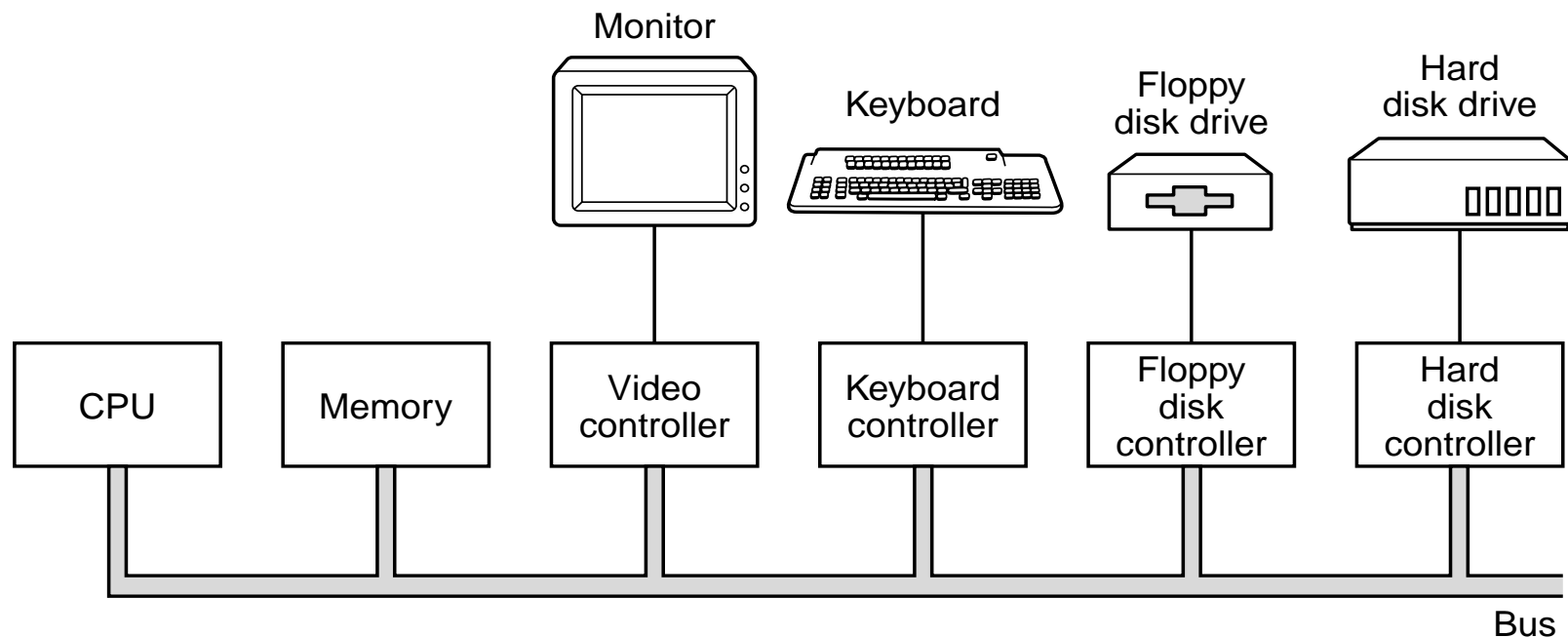
Rodzaje systemów operacyjnych

- ☐ dla systemów klasy mainframe
- ☐ dla serwerów różnego rodzaju
- ☐ dla wieloprocessorów i multikomputerów
- ☐ dla komputerów osobistych
- ☐ dla systemów czasu rzeczywistego
- ☐ dla systemów wbudowanych
- ☐ dla "inteligentnych przedmiotów"

>>>

Aspekty sprzętowe (1)

- schemat prostego komputera osobistego

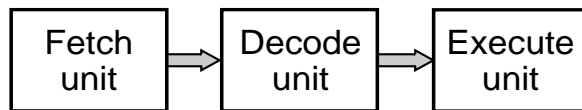


>>>

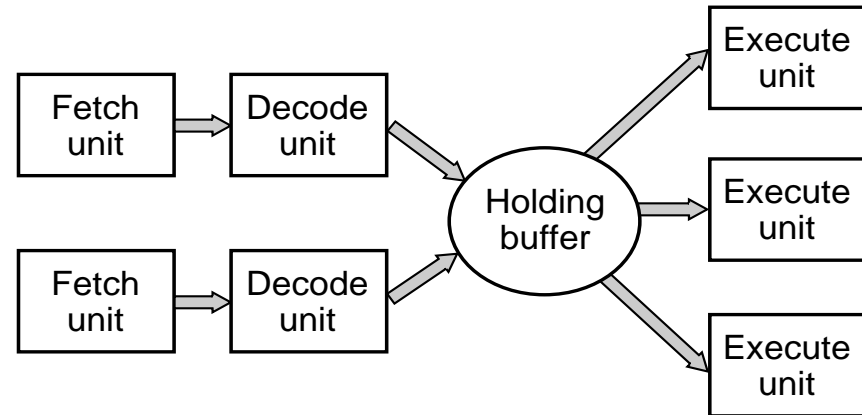
Aspekty sprzętowe (2)

□ procesor centralny

- (a) – proste przetwarzanie potokowe
- (b) – przetwarzanie superskalarne



(a)



(b)

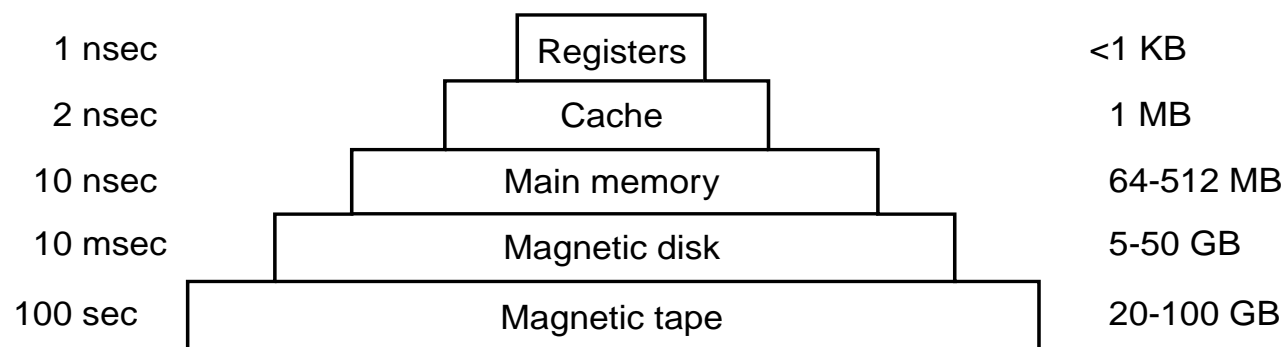
>>>

Aspekty sprzętowe (3)

- typowa "hierarchia pamięci"

Typical access time

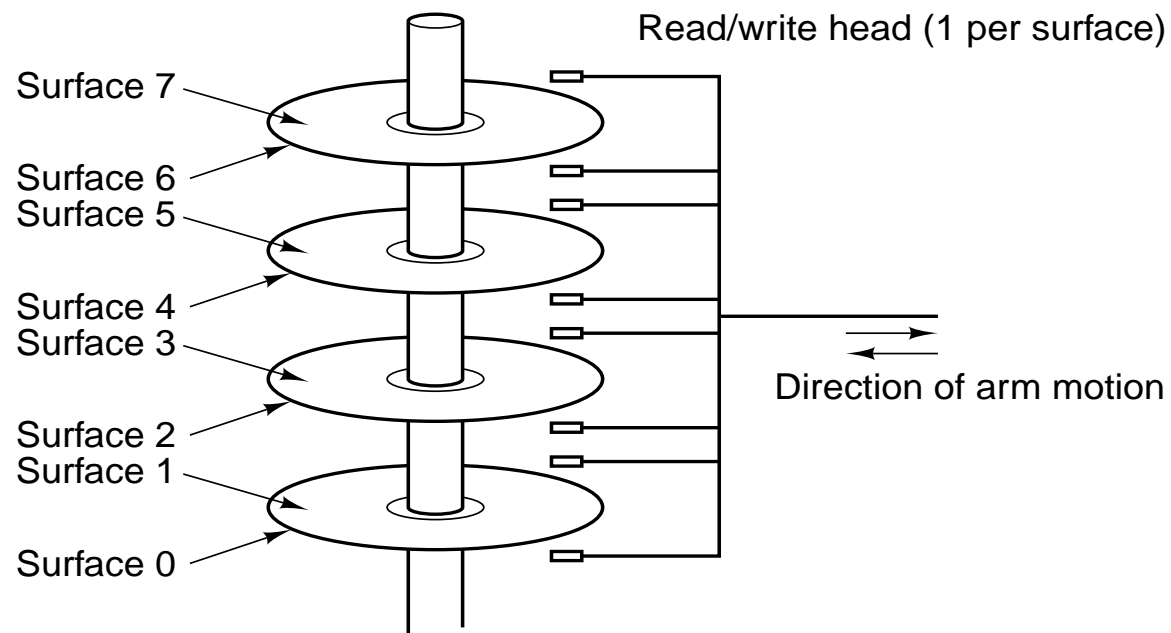
Typical capacity



>>>

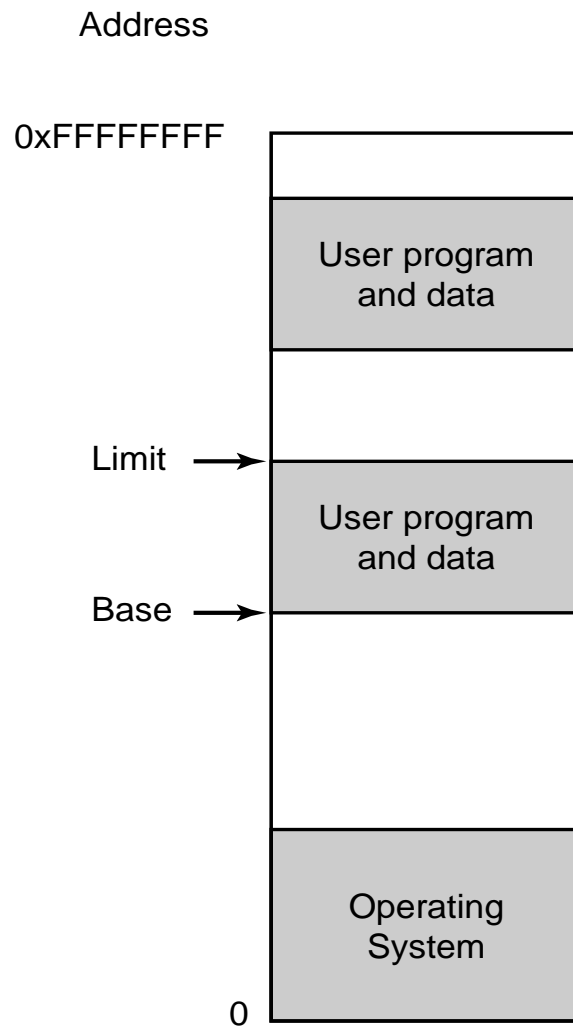
Aspekty sprzętowe (4)

- typowa pamięć dyskowa

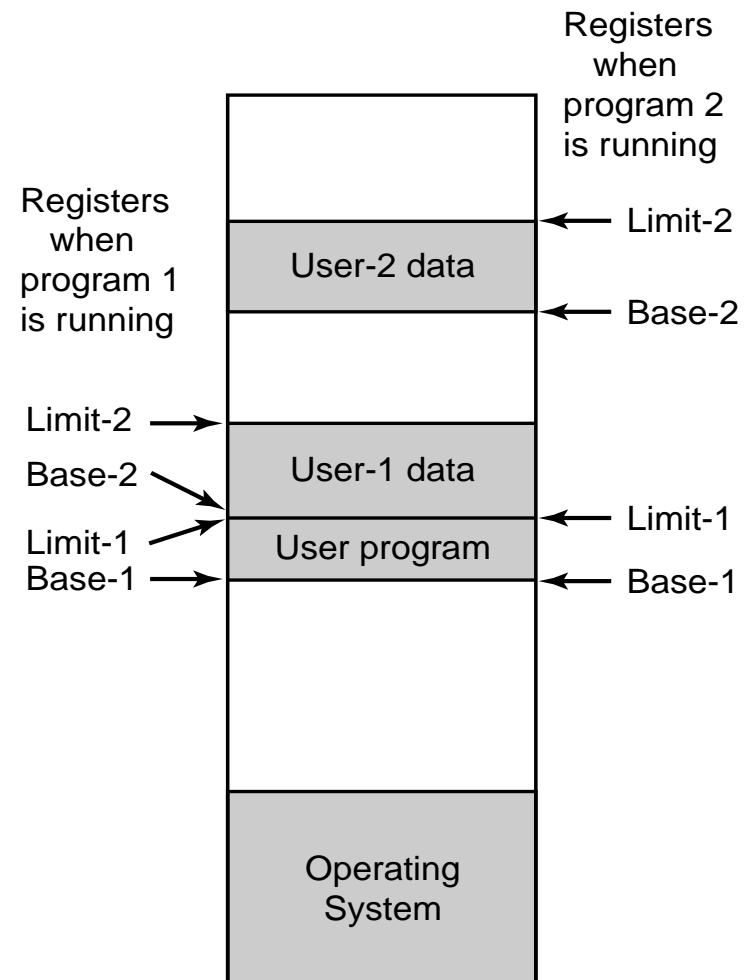


>>>

Aspekty sprzętowe (5)



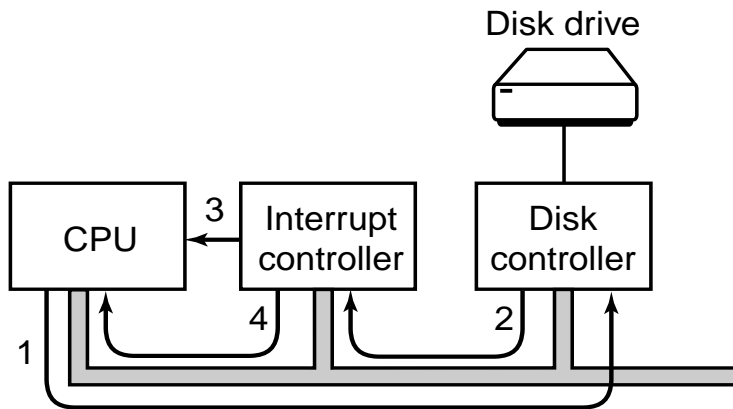
(a)



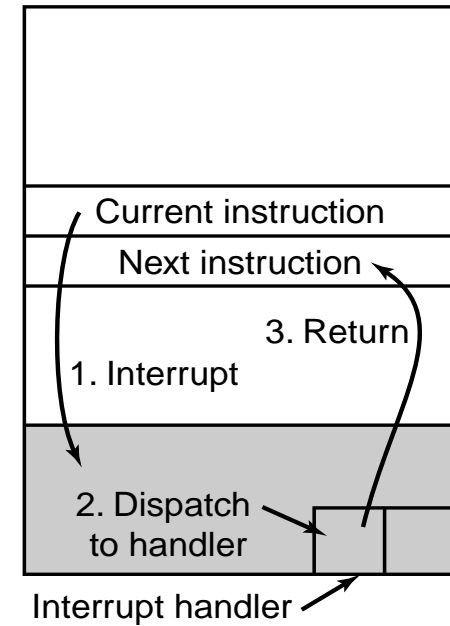
(b)

Aspekty sprzętowe (6)

- obsługa urządzeń We/Wy
 - (a) – etapy obsługi
 - (b) – obsługa przerwania przez system operacyjny



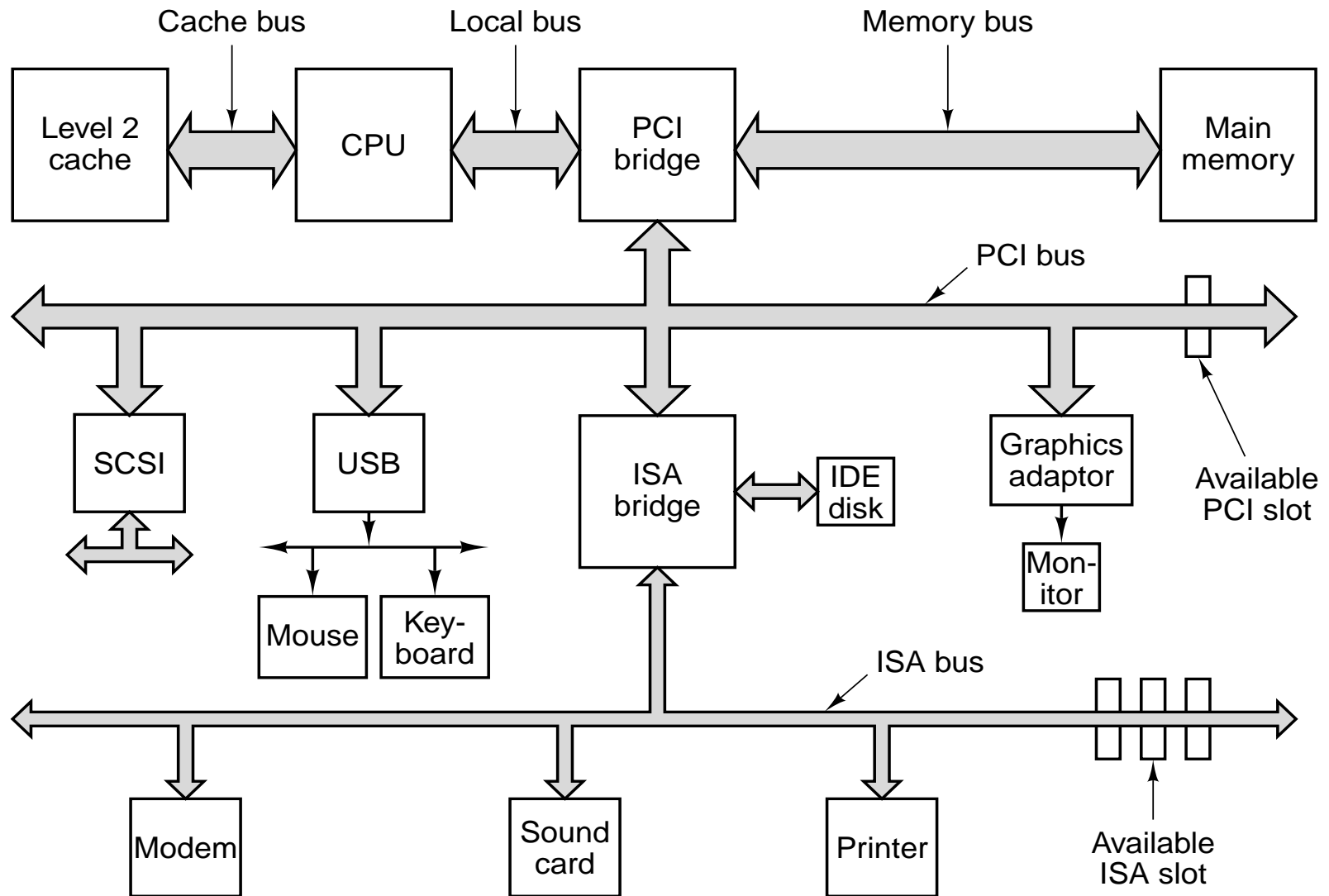
(a)



(b)

>>>

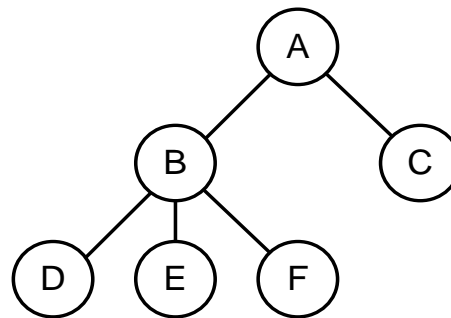
Aspekty sprzętowe (7)



Podstawowe pojęcia (1)

- proces
 - aktywna, zarządzana jednostka, której system przydziela zasoby

- "drzewo procesów"

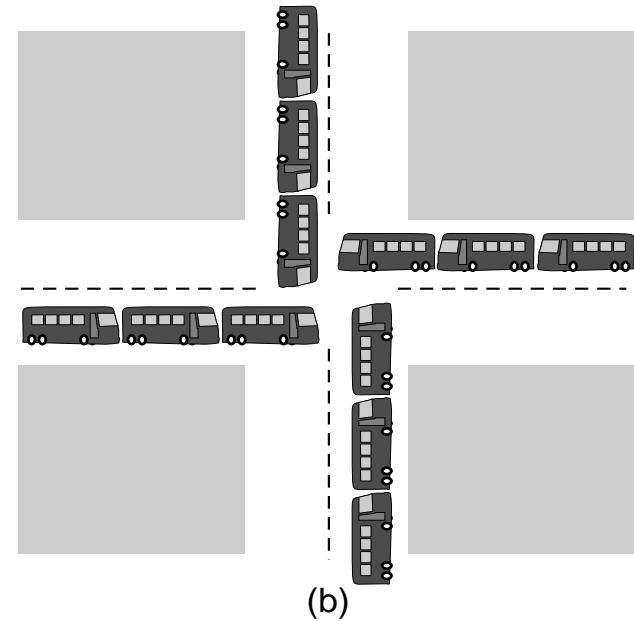
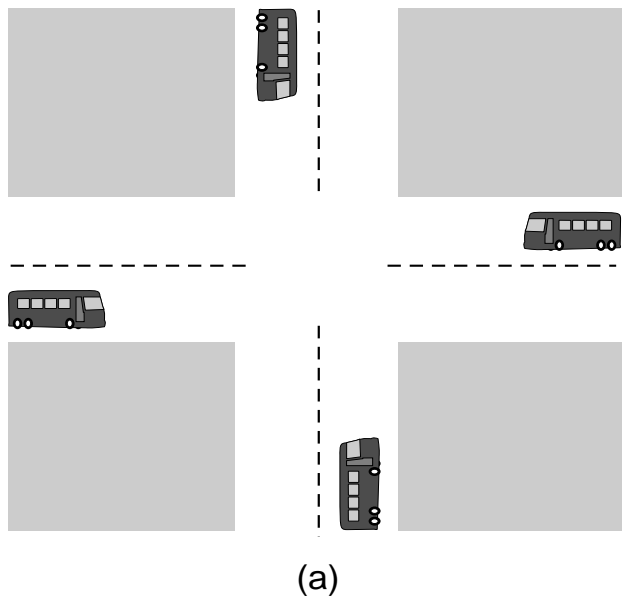


>>>

Podstawowe pojęcia (2)

□ zakleszczenie (ang.: deadlock)

- (a) – potencjalne
- (b) – aktualne

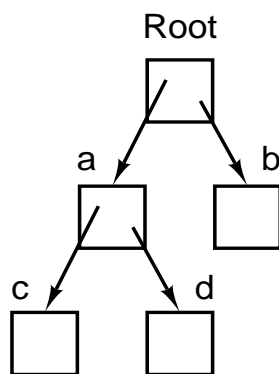


>>>

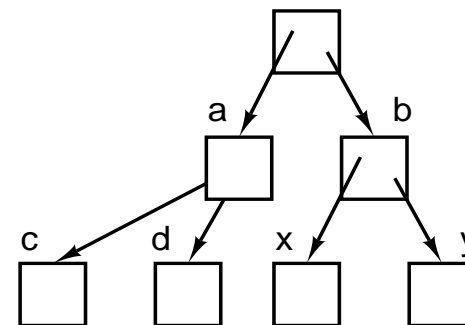
Podstawowe pojęcia (3)

□ pliki i systemy plikowe

- (a) pliki na dyskietce i w systemie plikowym
- (b) system plikowy po dołączeniu dyskietki



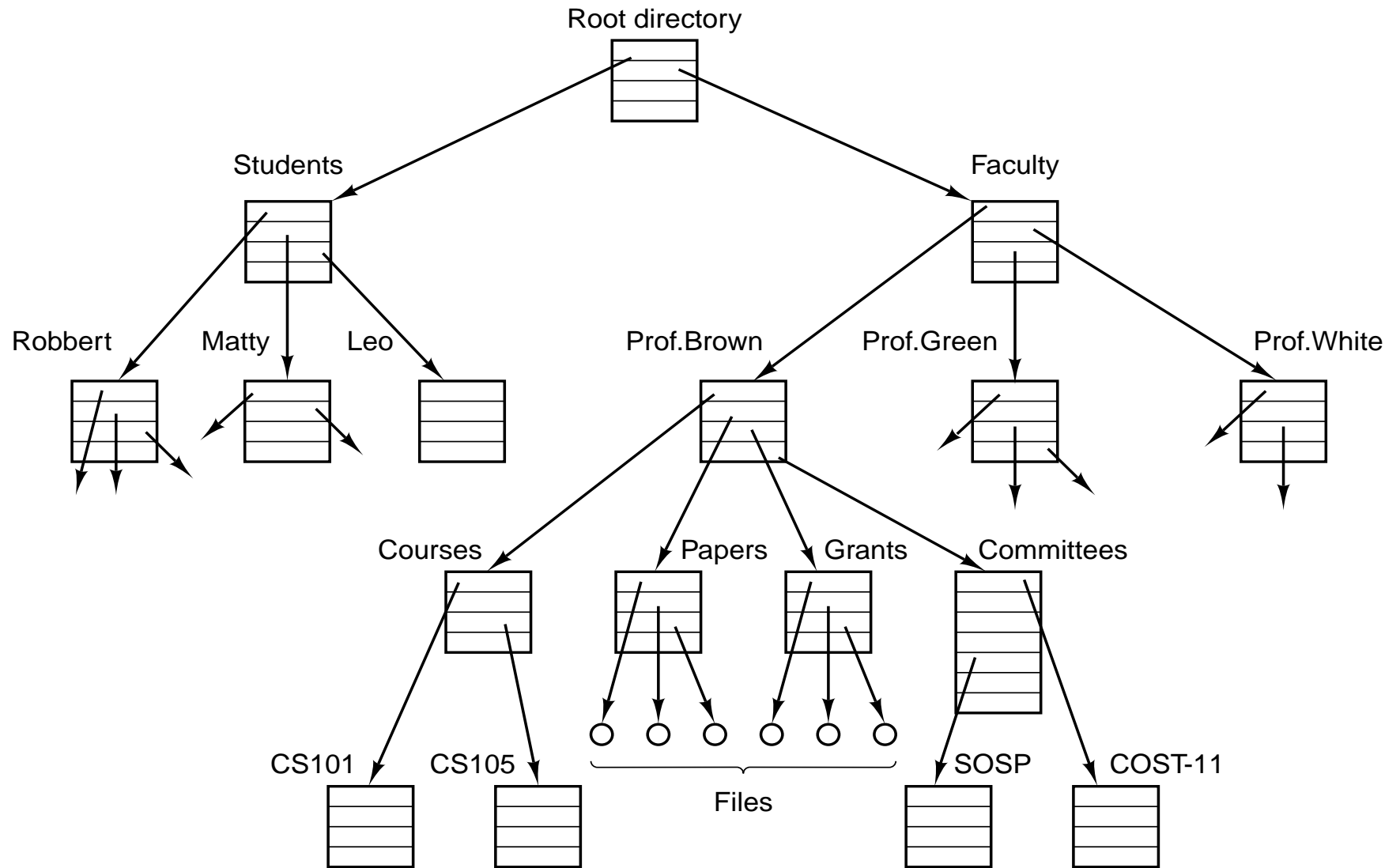
(a)



(b)

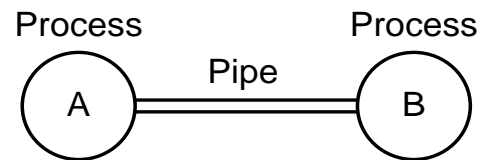
>>>

Podstawowe pojęcia (4)



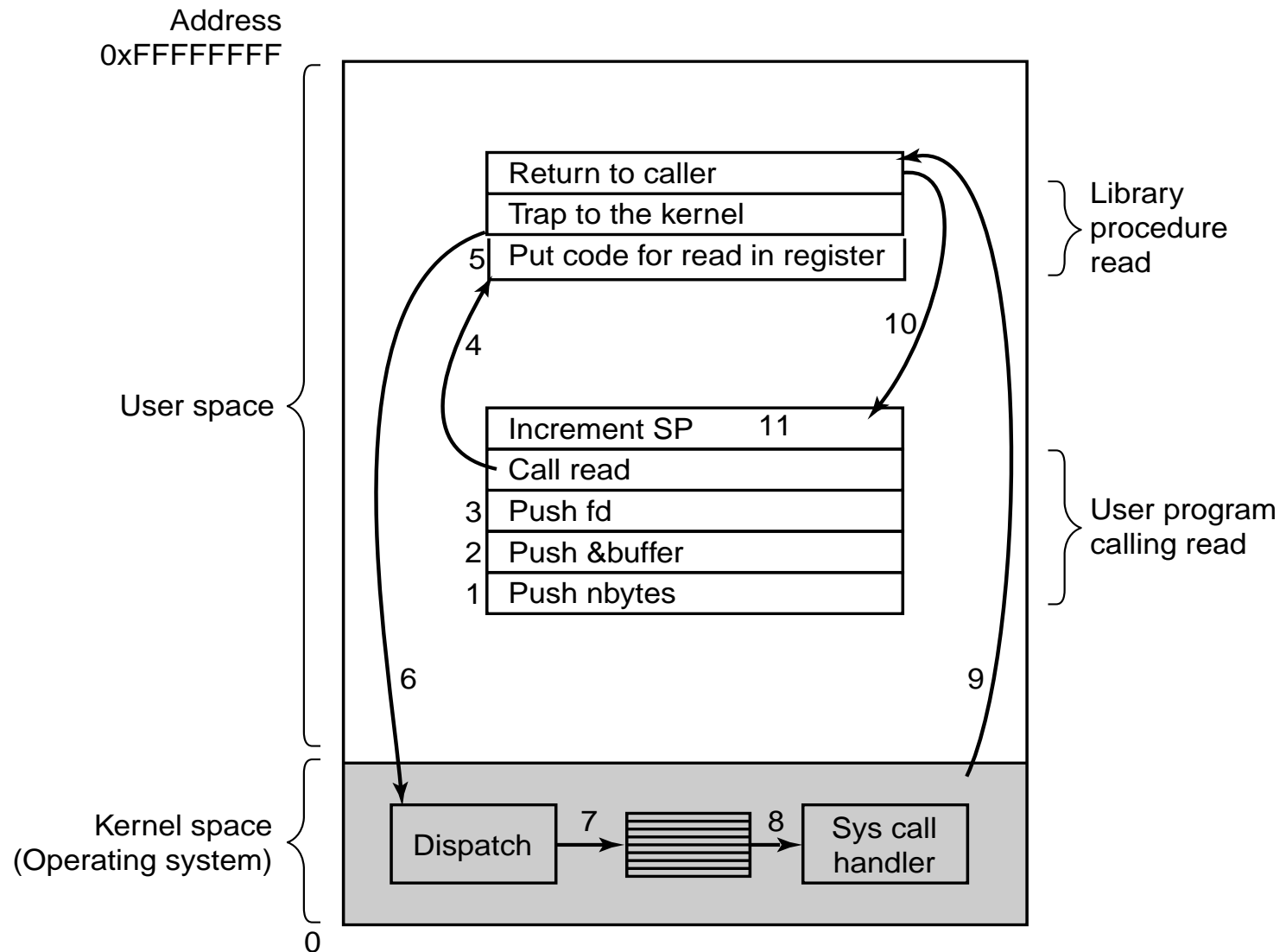
Podstawowe pojęcia (5)

- komunikacja międzyprocesowa
- łącze komunikacyjne



>>>

Odwołanie systemowe – obsługa



Typowe odwołania systemowe

Process management

Call	Description
pid = fork()	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

Directory and file system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Miscellaneous

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

Wykorzystanie odwołań systemowych

- trywialna "powłoka systemowa"

```
#define TRUE 1

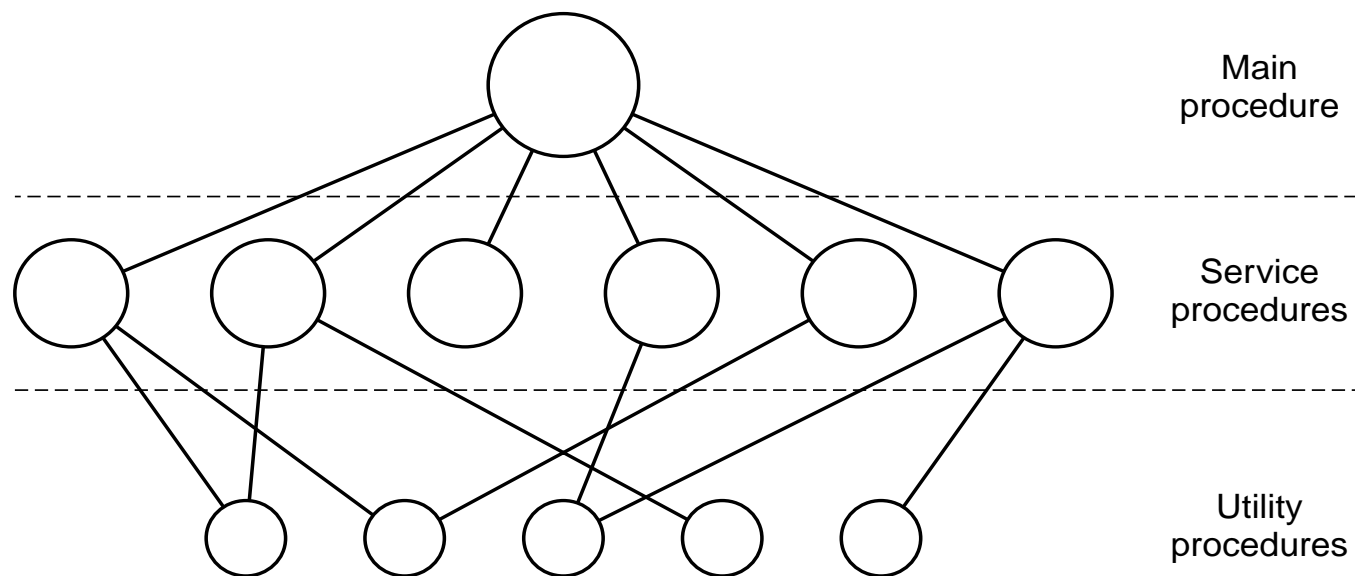
while (TRUE) {                                /* repeat forever */
    type_prompt( );                            /* display prompt on the screen */
    read_command(command, parameters);        /* read input from terminal */

    if (fork( ) != 0) {                        /* fork off child process */
        /* Parent code. */
        waitpid(-1, &status, 0);              /* wait for child to exit */
    } else {
        /* Child code. */
        execve(command, parameters, 0);      /* execute command */
    }
}
```

>>>

Struktura systemu operacyjnego (1)

- prosty system monolityczny



>>>

Struktura systemu operacyjnego (2)

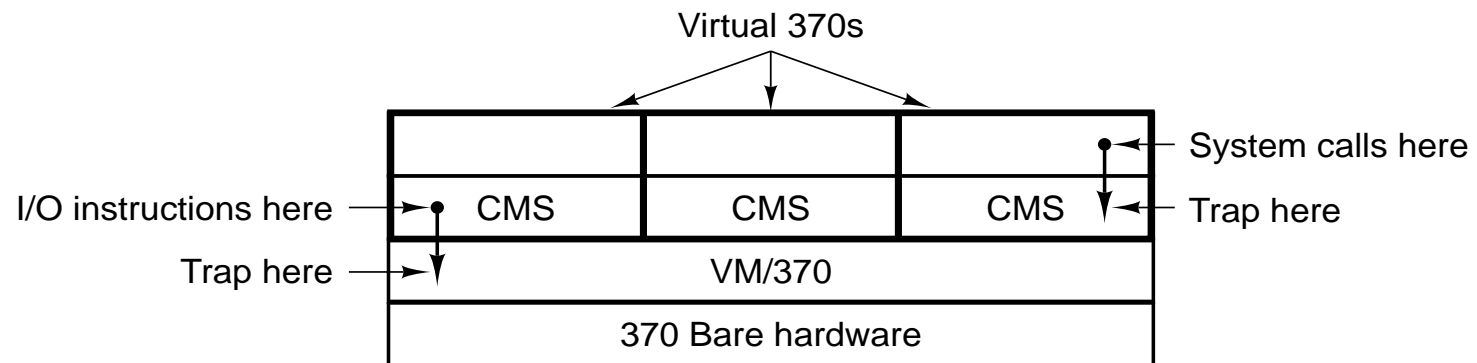
- struktura warstwowa (system THE)

Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

>>>

Struktura systemu operacyjnego (3)

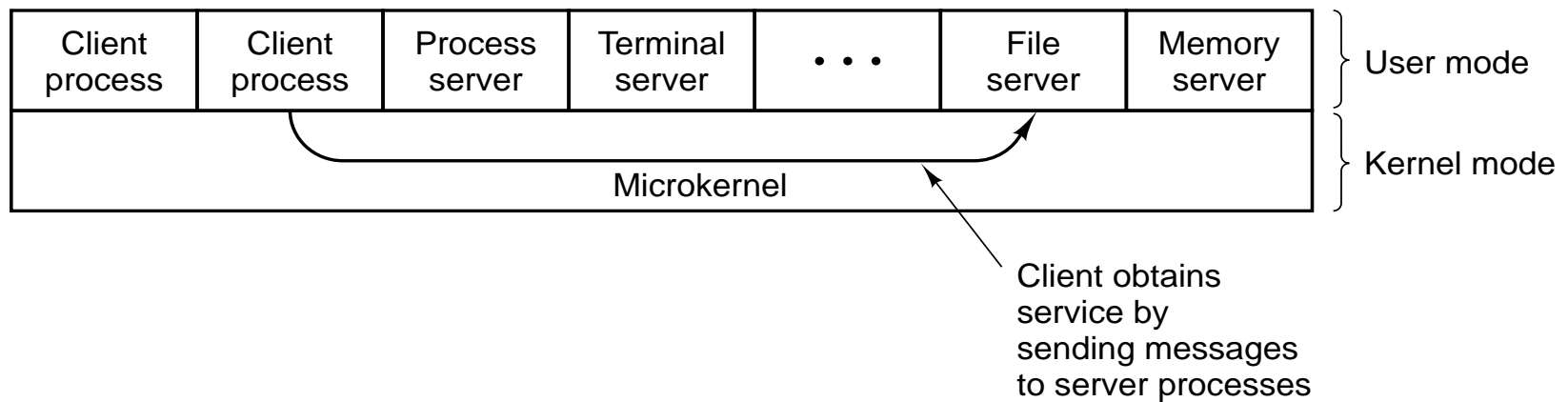
- systemy maszyn wirtualnych (VM/CMS)



>>>

Struktura systemu operacyjnego (4)

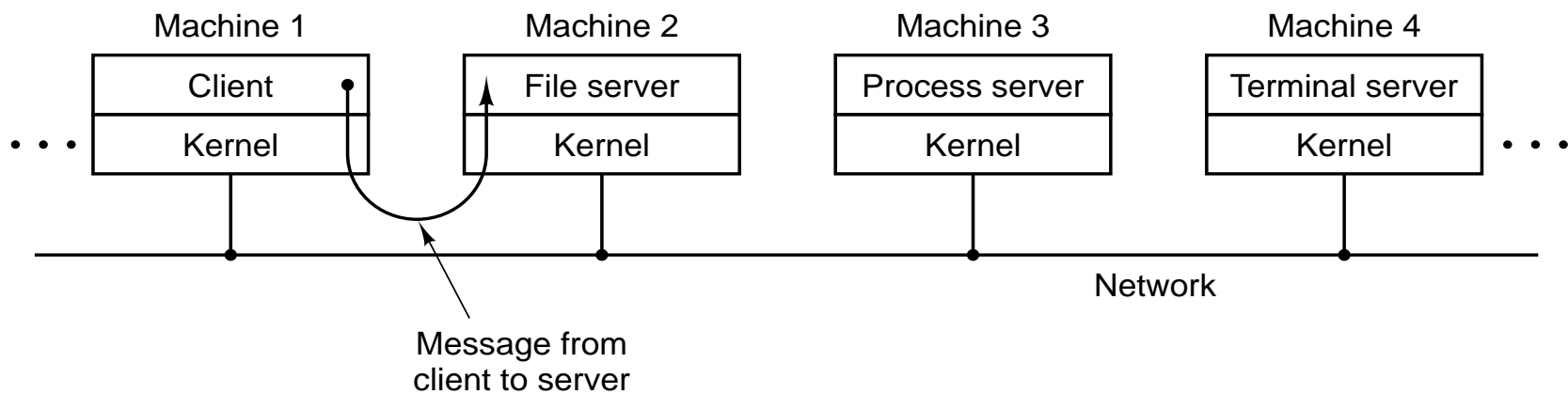
□ mikrojądro – model klient–serwer



>>>

Struktura systemu operacyjnego (5)

□ system rozproszony



===

02 – Procesy i wątki

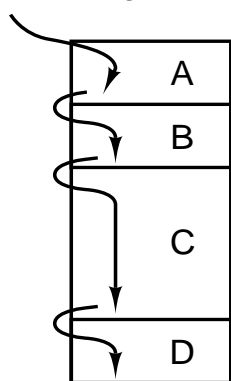
- ☐ Procesy
- ☐ Wątki
- ☐ Komunikacja międzyprocesowa
- ☐ Problemy komunikacji międzyprocesowej
- ☐ Szeregowanie zadań

>>>

Procesy a wieloprogramowość

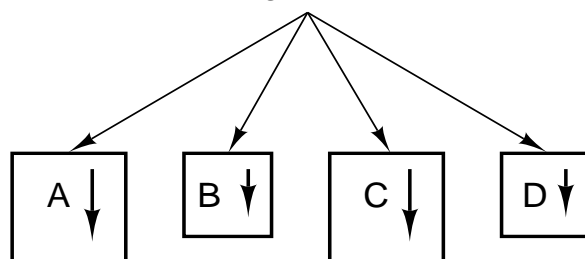
- "równoczesne" wykonywanie kilku programów
 - (a) – przełączanie aktywności wg. jednego licznika wykonania
 - (b) – niezależne liczniki wykonania dla każdego programu --> procesy
 - (c) – diagram aktywności poszczególnych procesów

One program counter

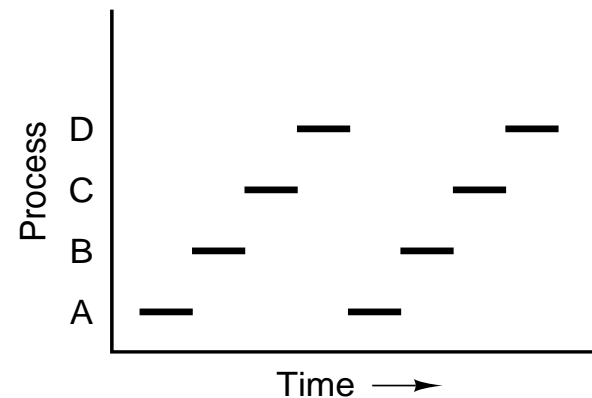


(a)

Four program counters



(b)



(c)

>>>

Procesy

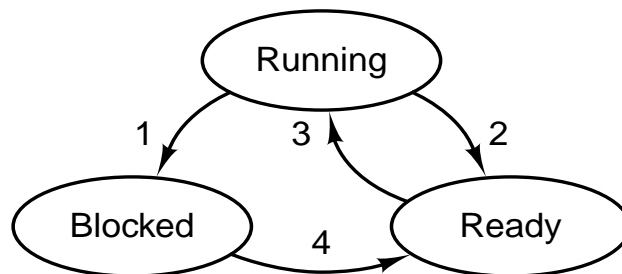
- tworzenie nowego procesu
 - inicjowanie systemu operacyjnego
 - podsystem inicjowania/kreacji procesów
 - polecenie użytkownika (interakcyjne)
 - przetwarzanie wsadowe (nieinterakcyjne)
- usunięcie procesu
 - normalne zakończenie pracy (dobrowolne)
 - wykrycie błędu aplikacji (dobrowolne)
 - wystąpienie błędu krytycznego (wymuszone)
 - wymuszenie zakończenia pracy przez system lub inny proces
- powiązania
 - wszystkie utworzone procesy są równorzędne (np. VMS, WinNT)
 - hierarchia starszeństwa, grupy procesów (np. Unix)

>>>

Stany procesu

- trzy podstawowe stany procesu
 - stan wykonywania (Running)
 - stan zatrzymania (Blocked)
 - stan gotowości (Ready)

- uproszczony diagram stanów

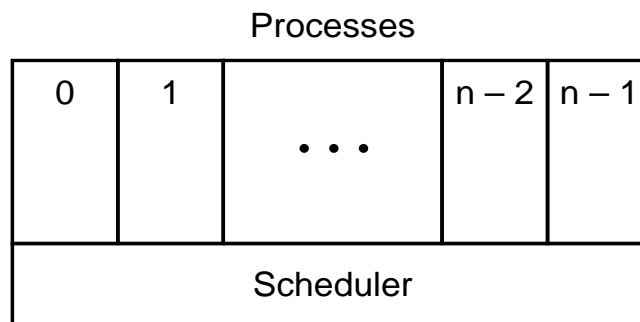


1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

>>>

Szeregowanie procesów

- podsystem szeregowania
 - "planista" (scheduler)
 - obsługa przerwań, wyjątków, etc.
 - praca na różnych poziomach wykonywania



- systemowa tabela procesów
 - zawiera informację o stanie procesów

>>>

Element typowej tabeli procesów

Process management	Memory management	File management
Registers Program counter Program status word Stack pointer Process state Priority Scheduling parameters Process ID Parent process Process group Signals Time when process started CPU time used Children's CPU time Time of next alarm	Pointer to text segment Pointer to data segment Pointer to stack segment	Root directory Working directory File descriptors User ID Group ID

Obsługa przerwań

□ przykładowy schemat postępowania (Unix)

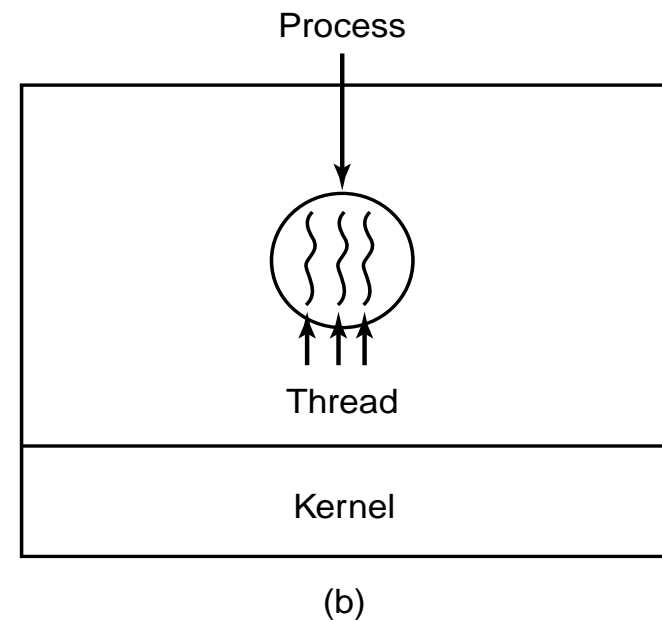
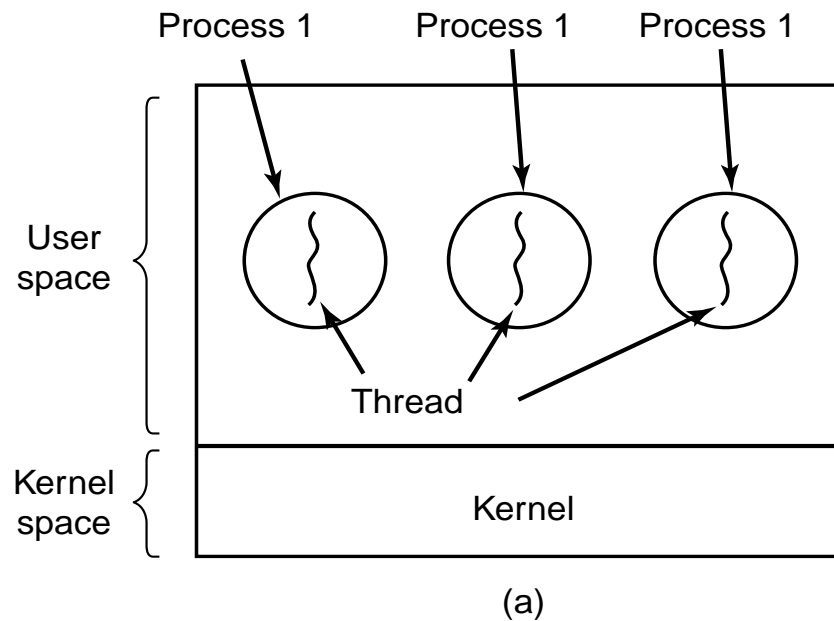
1. Hardware stacks program counter, etc.
2. Hardware loads new program counter from interrupt vector.
3. Assembly language procedure saves registers.
4. Assembly language procedure sets up new stack.
5. C interrupt service runs (typically reads and buffers input).
6. Scheduler decides which process is to run next.
7. C procedure returns to the assembly code.
8. Assembly language procedure starts up new current process.

>>>

Wątki wykonania

□ procesy a wątki wykonania

- (a) – trzy procesy, każdy posiada tylko jeden wątek wykonania
- (b) – jeden proces, posiadający trzy różne wątki wykonania



>>>

Systemowy opis wątków wykonania

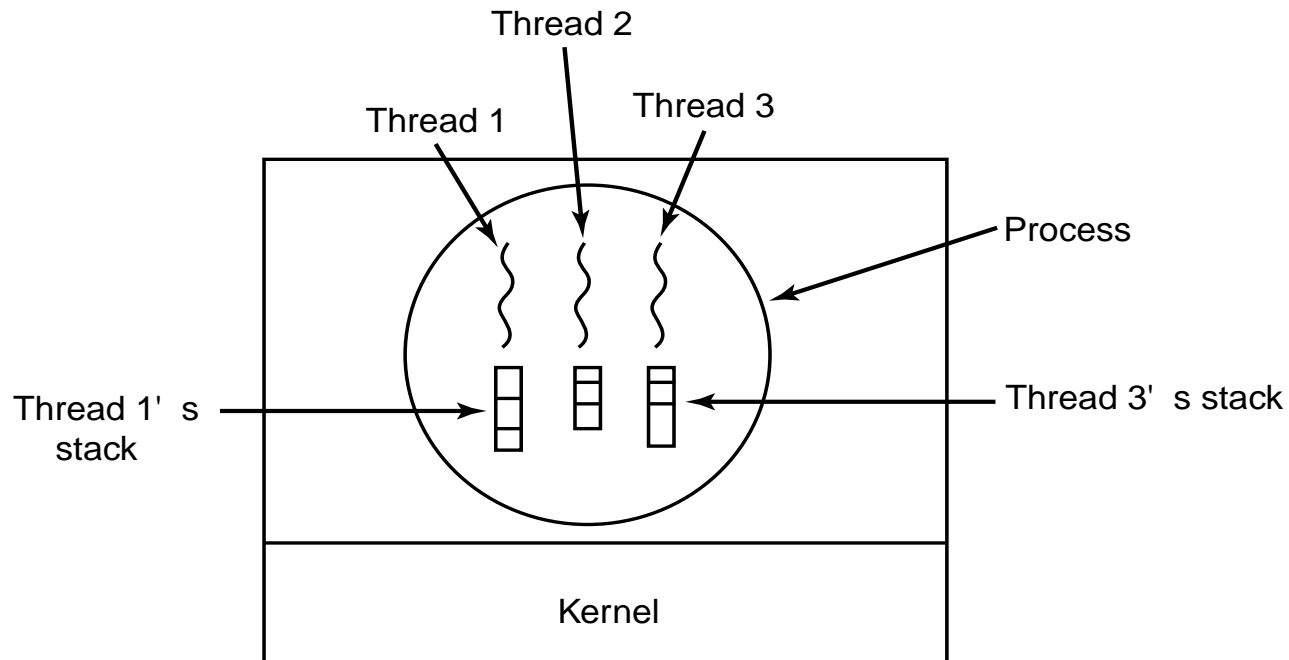
- parametry procesu
 - dotyczą wszystkich wątków danego procesu
- parametry wątku
 - zawierają dane specyficzne dla danego wątku

Per process items	Per thread items
Address space	Program counter
Global variables	Registers
Open files	Stack
Child processes	State
Pending alarms	
Signals and signal handlers	
Accounting information	

>>>

Wątki wykonania a stos

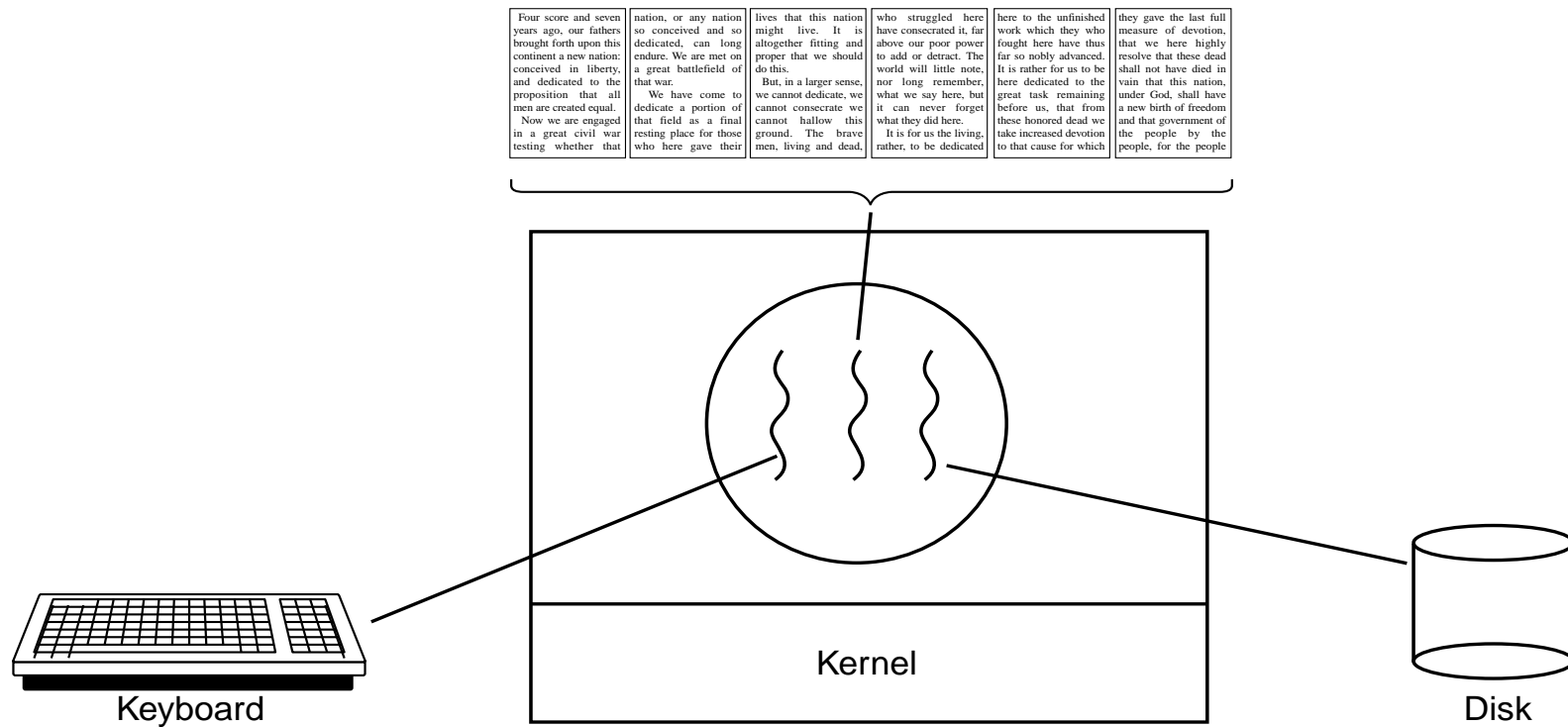
- każdy wątek musi posiadać własny stos



>>>

Wykorzystanie wielowątkowości (1)

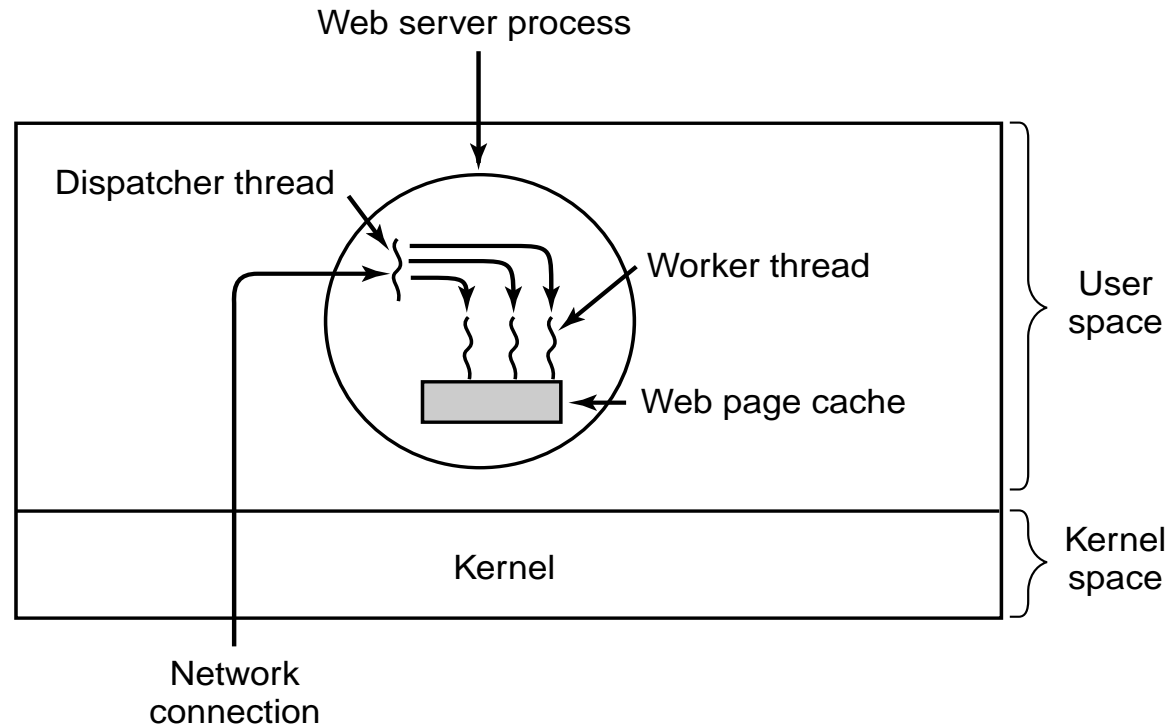
- edytor/procesor tekstu



>>>

Wykorzystanie wielowątkowości (2)

□ serwer WWW



>>>

Wykorzystanie wielowątkowości (3)

- szkic fragmentów kodu serwera WWW
 - (a) – wątek ekspedytora (dispatcher thread)
 - (b) – wątek roboczy (worker thread)

```
while (TRUE) {  
    get_next_request(&buf);  
    handoff_work(&buf);  
}
```

(a)

```
while (TRUE) {  
    wait_for_work(&buf)  
    look_for_page_in_cache(&buf, &page);  
    if (page_not_in_cache(&page))  
        read_page_from_disk(&buf, &page);  
    return_page(&page);  
}
```

(b)

>>>

Wykorzystanie wielowątkowości (4)

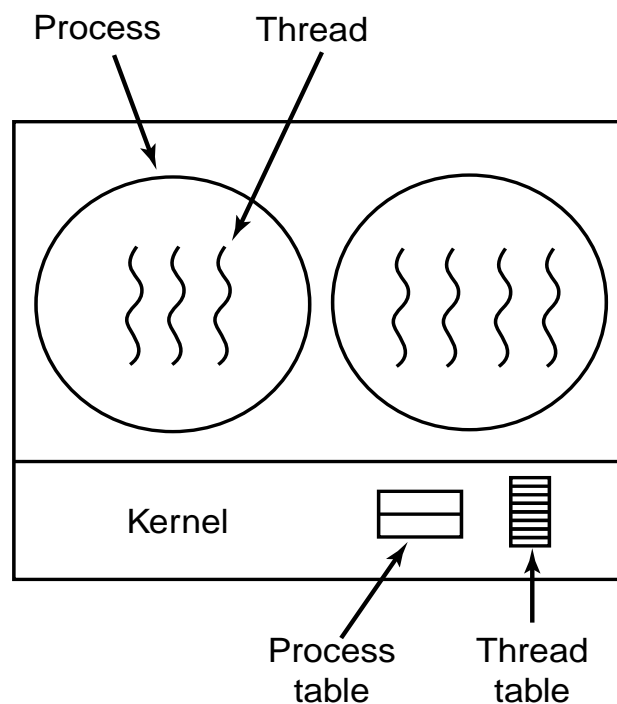
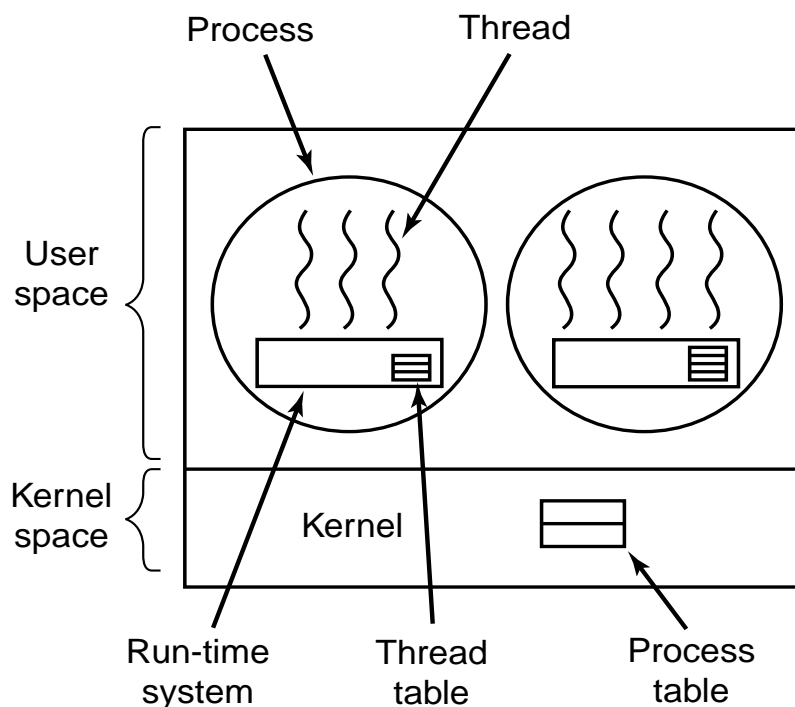
- typowe sposoby implementacji serwerów
 - porównanie trzech modeli

Model	Characteristics
Threads	Parallelism, blocking system calls
Single-threaded process	No parallelism, blocking system calls
Finite-state machine	Parallelism, nonblocking system calls, interrupts

>>>

Implementacja wielowątkowości (1)

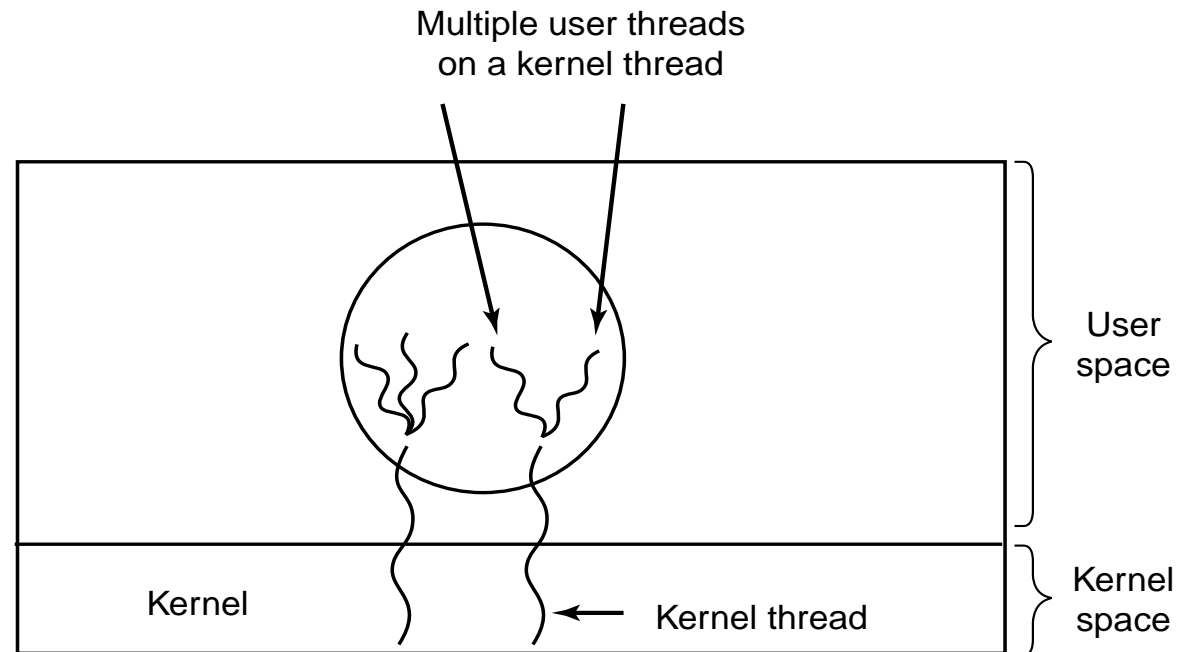
- na poziomie użytkownika (user-level)
- na poziomie jądra (kernel-level)



>>>

Implementacja wielowątkowości (2)

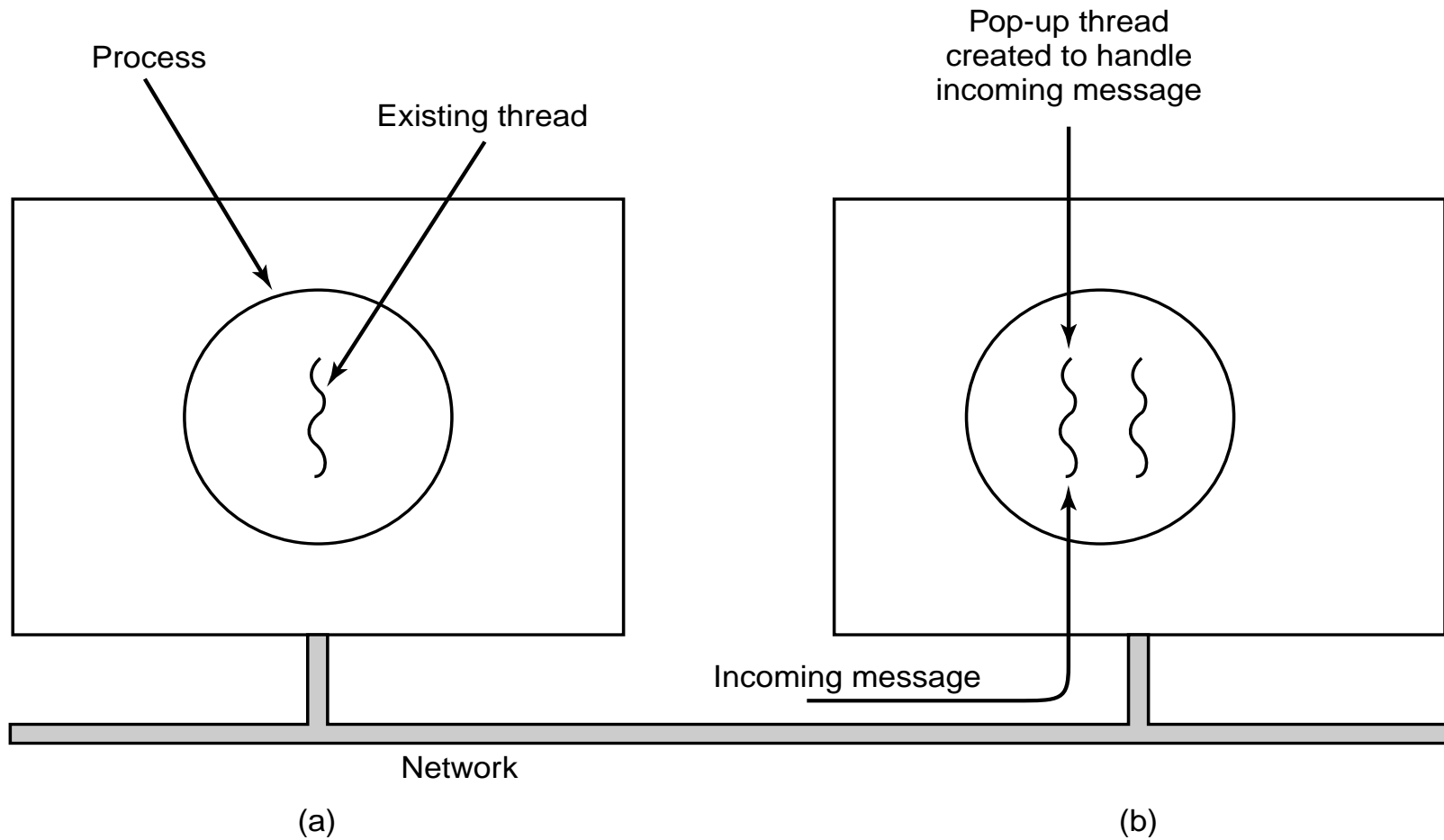
- model hybrydowy



>>>

Wątki kreowane dynamicznie (1)

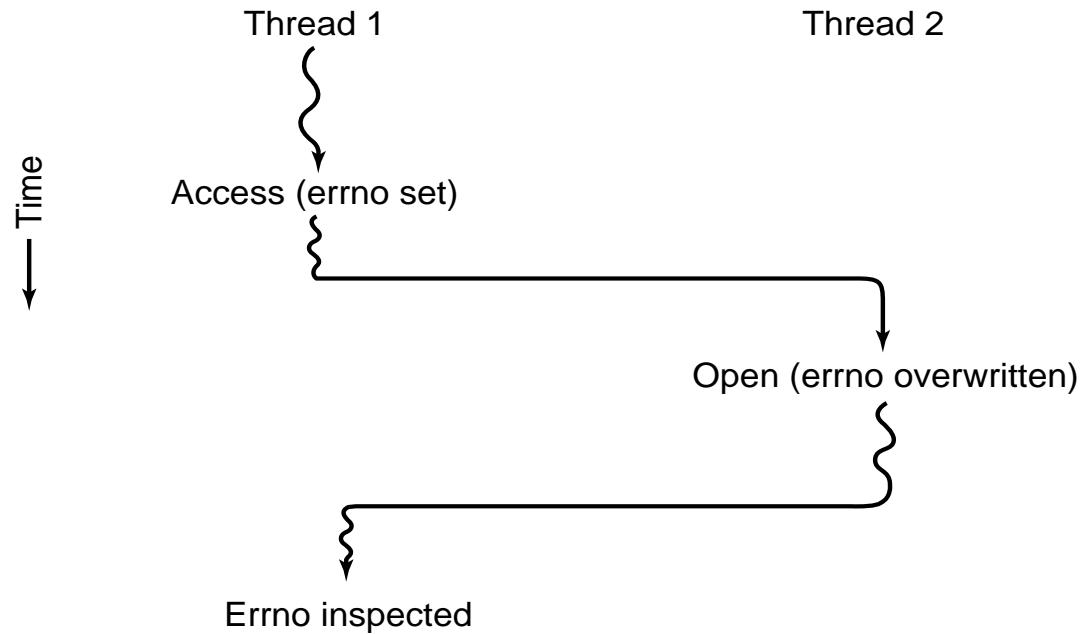
- "pop-up" threads
 - przykład: obsługa komunikatów



>>>

Wątki kreowane dynamicznie (2)

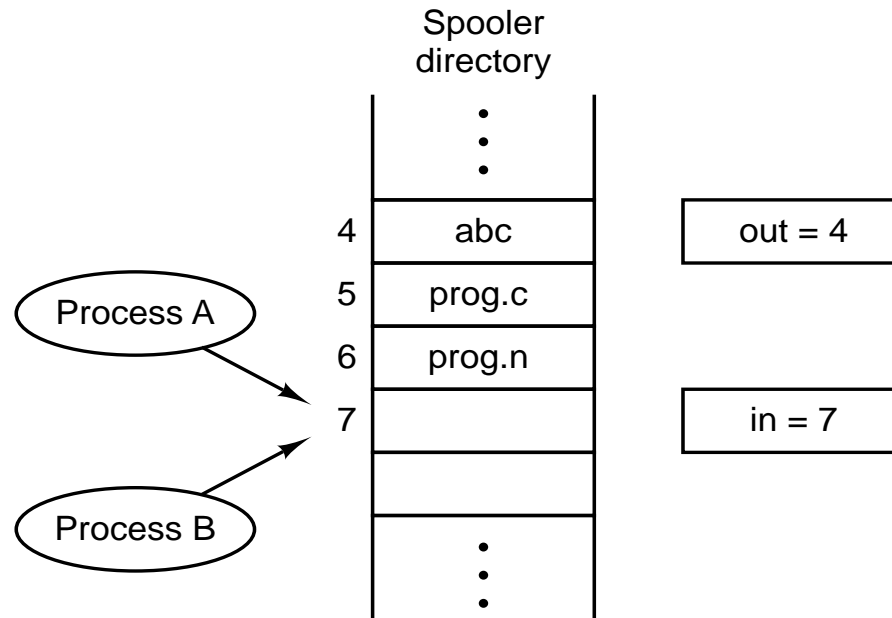
- możliwe sytuacje konfliktowe
 - przykład: wykorzystanie zmiennej globalnej 'errno'



>>>

Współzawodnictwo i współpraca

- sytuacja wyścigu ("race condition")
 - przykład: próba równoczesnego uzyskania dostępu do segmentu 7



- współpraca
 - wymaga komunikacji między procesami/wątkami
 - może wymagać zapewnienia wyłączności dostępu do zasobów

>>>

Wyłączny dostęp do zasobów (1)

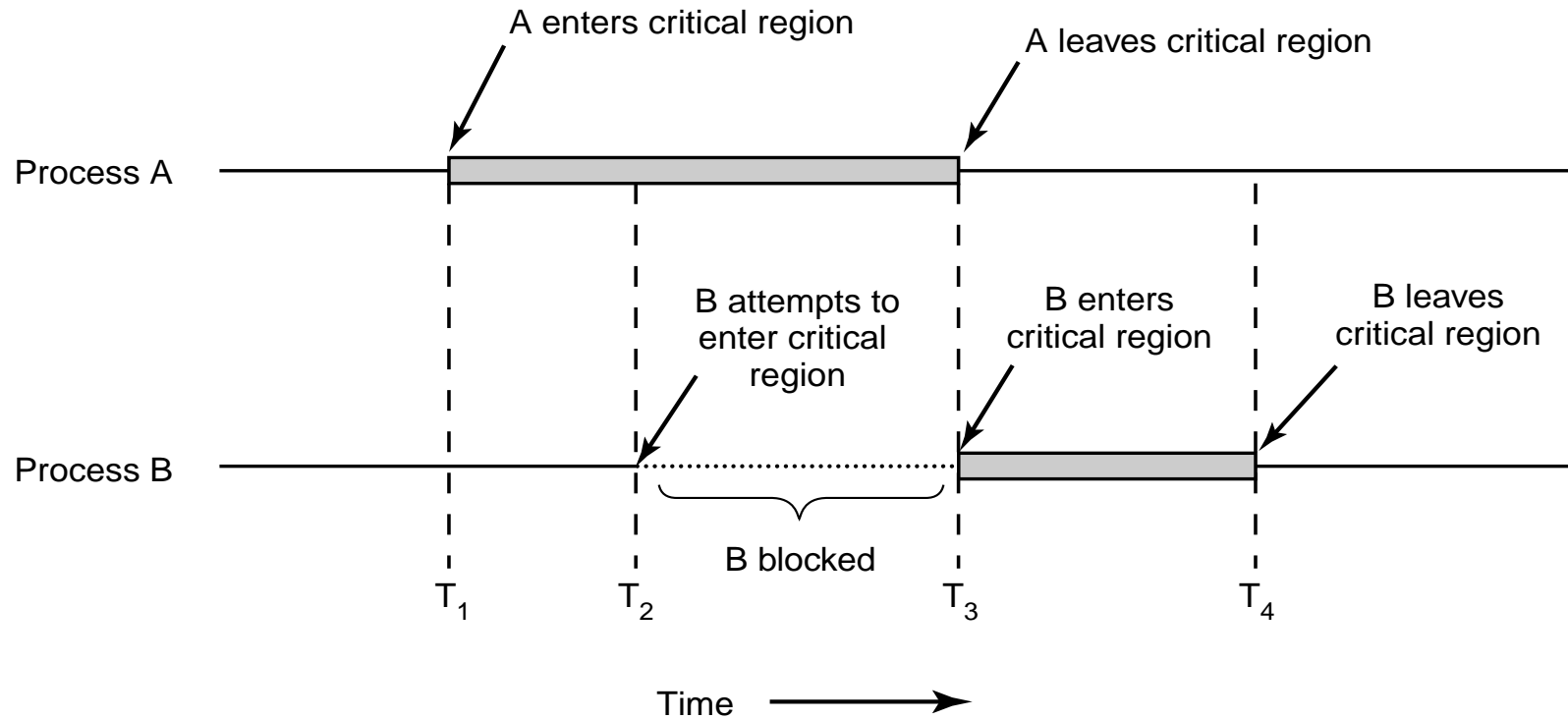
- region krytyczny (sekcja krytyczna) procesu
 - segment kodu wymagający wyłączności dostępu do zasobów
 - określany na podstawie wymaganych zasobów (pliki, obszary pamięci, etc.)

- podstawowe warunki
 - precyzyjne określenie regionu krytycznego
 - w regionie krytycznym może wykonywać się tylko jeden proces
 - niezależność od szybkości przetwarzania
 - niezależność od ilości procesorów przetwarzających dane
 - poza regionem krytycznym procesy nie mogą się wzajemnie blokować
 - czas oczekiwania na wejście do regionu krytycznego jest zawsze skończony

>>>

Wyłączny dostęp do zasobów (2)

- przykład wykorzystania regionu krytycznego



Wyłączny dostęp do zasobów (3)

- przykład implementacji dla regionu krytycznego
 - metoda "busy waiting"
 - sterowanie poprzez zmienną globalną
 - (a) – proces A (turn=0)
 - (b) – proces B (turn=1)

```
while (TRUE) {  
    while (turn != 0)      /* loop */ ;  
    critical_region( );  
    turn = 1;  
    noncritical_region( );  
}
```

(a)

```
while (TRUE) {  
    while (turn != 1)      /* loop */ ;  
    critical_region( );  
    turn = 0;  
    noncritical_region( );  
}
```

(b)

Semafor (1)

- definicja
 - zmienna całkowita, do wyłącznego użytku przez funkcje sterujące
 - może sygnalizować ilość dostępnych zasobów
- przypadek szczególny: "zamek"
 - MUTEX (ang.: MUTual EXclusion)
 - standardowo sygnalizuje dostępność/niedostępność
- standardowe funkcje sterujące
 - wait(), down() – zmniejszenie wartości semafora o jeden
 - signal(), up() – zwiększenie wartości semafora o jeden
- przykład: problem producent – konsument

>>>

Semafor (2)

```
#define N 100
typedef int semaphore;
semaphore mutex = 1;
semaphore empty = N;
semaphore full = 0;

void producer(void)
{
    int item;

    while (TRUE) {
        item = produce_item();
        down(&empty);
        down(&mutex);
        insert_item(item);
        up(&mutex);
        up(&full);
    }
}

void consumer(void)
{
    int item;

    while (TRUE) {
        down(&full);
        down(&mutex);
        item = remove_item();
        up(&mutex);
        up(&empty);
        consume_item(item);
    }
}
```

```
/* number of slots in the buffer */
/* semaphores are a special kind of int */
/* controls access to critical region */
/* counts empty buffer slots */
/* counts full buffer slots */

/* TRUE is the constant 1 */
/* generate something to put in buffer */
/* decrement empty count */
/* enter critical region */
/* put new item in buffer */
/* leave critical region */
/* increment count of full slots */

/* infinite loop */
/* decrement full count */
/* enter critical region */
/* take item from buffer */
/* leave critical region */
/* increment count of empty slots */
/* do something with the item */
```

Monitory (1)

- definicja

- zbiór procedur, reprezentujących wykonywane operacje, oraz
- współdzielone dane, przetwarzane przez te procedury

- właściwości

- procedury monitora korzystają wyłącznie ze zmiennych lokalnych i parametrów wywołania
- monitor gwarantuje bezkonfliktowe wykonanie zleconych operacji

>>>

Monitory (2)

- przykładowa struktura monitora

```
monitor example
    integer i;
    condition c;

    procedure producer( );
        .
        .
        .
    end;

    procedure consumer( );
        .
        .
        .
    end;
end monitor;
```

Komunikaty

□ problem typu producent–konsument

```
#define N 100                                /* number of slots in the buffer */

void producer(void)
{
    int item;
    message m;                                /* message buffer */

    while (TRUE) {
        item = produce_item( );               /* generate something to put in buffer */
        receive(consumer, &m);                 /* wait for an empty to arrive */
        build_message(&m, item);               /* construct a message to send */
        send(consumer, &m);                    /* send item to consumer */
    }
}

void consumer(void)
{
    int item, i;
    message m;

    for (i = 0; i < N; i++) send(producer, &m); /* send N empties */
    while (TRUE) {
        receive(producer, &m);                 /* get message containing item */
        item = extract_item(&m);               /* extract item from message */
        send(producer, &m);                    /* send back empty reply */
        consume_item(item);                    /* do something with the item */
    }
}
```


Problem pięciu jedzących filozofów (1)

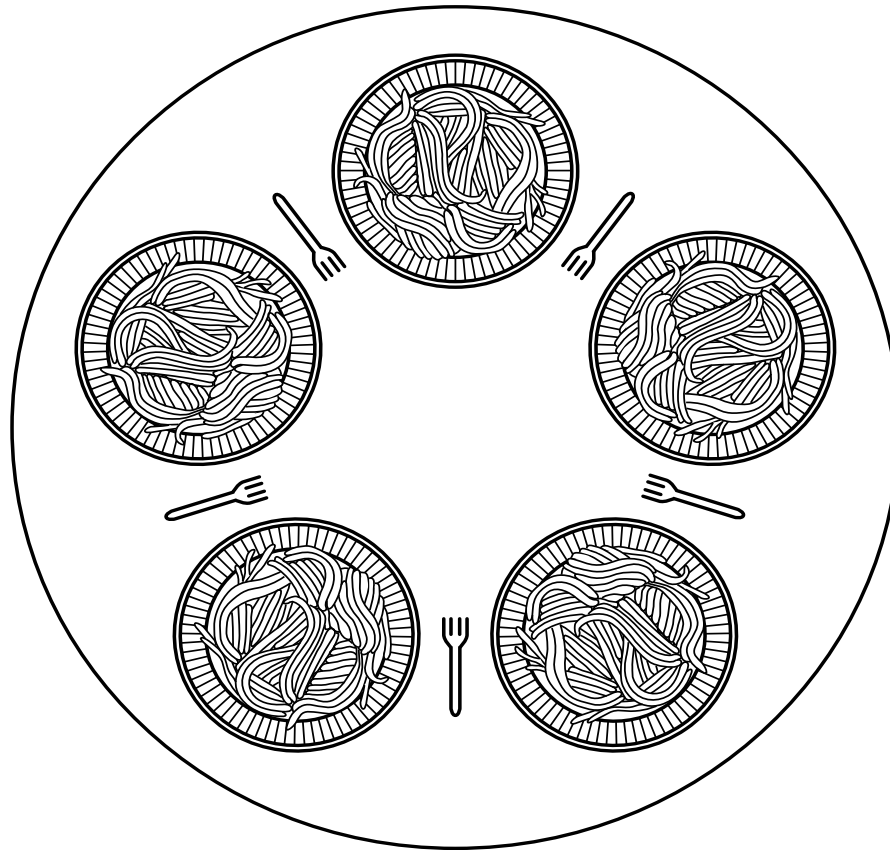
- klasyczny problem z zakleszczeniem

- opis
 - pięciu filozofów siedzi przy okrągłym stole
 - na stole znajduje się pięć talerzy ze spaghetti
 - pomiędzy talerzami leży tylko pięć widelców
 - jedzenie wymaga dwóch widelców
 - jednorazowo można pobrać tylko jeden widelec
 - każdy z filozofów albo je, albo myśli :–)

- schemat
 - następny slajd

>>>

Problem pięciu jedzących filozofów (2)



>>>

Problem pięciu jedzących filozofów (3)

- prosta implementacja – zakleszczenie (!)

```
#define N 5                                     /* number of philosophers */

void philosopher(int i)                         /* i: philosopher number, from 0 to 4 */
{
    while (TRUE) {
        think();                               /* philosopher is thinking */
        take_fork(i);                          /* take left fork */
        take_fork((i+1) % N);                  /* take right fork; % is modulo operator */
        eat();                                 /* yum-yum, spaghetti */
        put_fork(i);                          /* put left fork back on the table */
        put_fork((i+1) % N);                   /* put right fork back on the table */
    }
}
```

- możliwe rozwiązanie
 - następny slajd

Problem pięciu jedzących filozofów (4)

```
#define N          5                /* number of philosophers */
#define LEFT      (i+N-1)%N        /* number of i's left neighbor */
#define RIGHT     (i+1)%N          /* number of i's right neighbor */
#define THINKING  0                /* philosopher is thinking */
#define HUNGRY    1                /* philosopher is trying to get forks */
#define EATING    2                /* philosopher is eating */
typedef int semaphore;              /* semaphores are a special kind of int */
int state[N];                      /* array to keep track of everyone's state */
semaphore mutex = 1;               /* mutual exclusion for critical regions */
semaphore s[N];                    /* one semaphore per philosopher */

void philosopher(int i)             /* i: philosopher number, from 0 to N-1 */
{
    while (TRUE) {                  /* repeat forever */
        think();                   /* philosopher is thinking */
        take_forks(i);             /* acquire two forks or block */
        eat();                     /* yum-yum, spaghetti */
        put_forks(i);              /* put both forks back on table */
    }
}

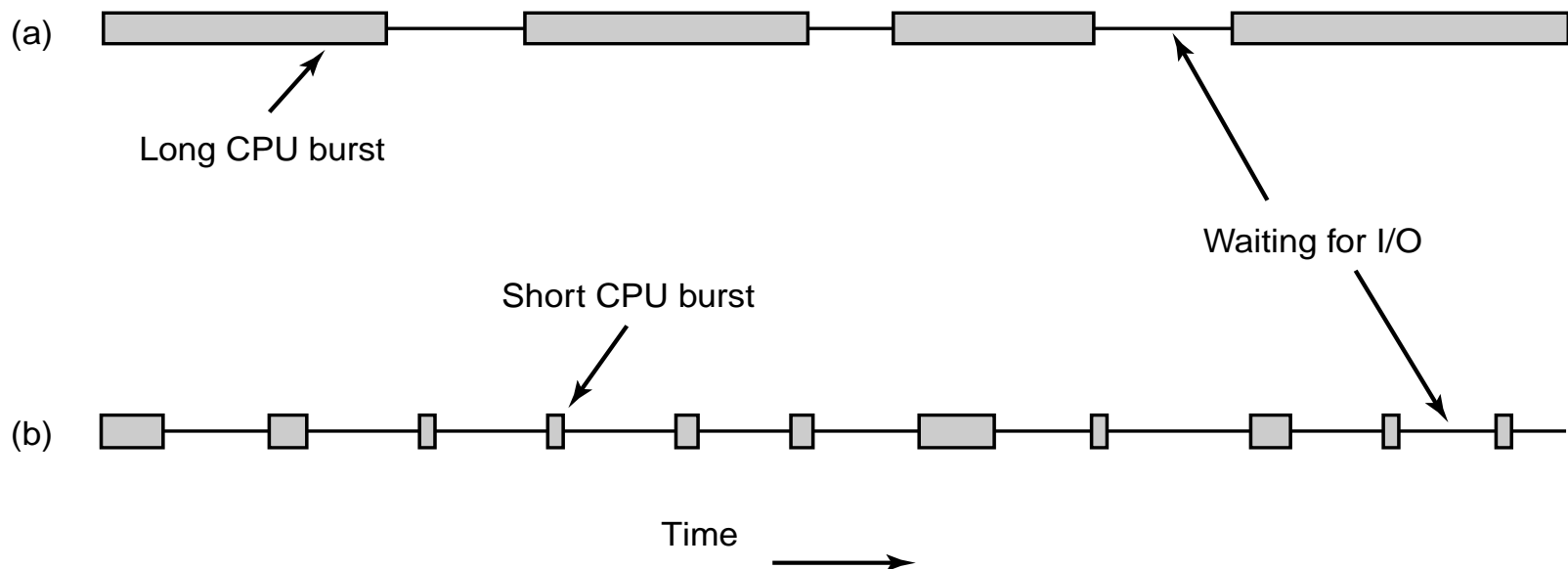
void take_forks(int i)              /* i: philosopher number, from 0 to N-1 */
{
    down(&mutex);                   /* enter critical region */
    state[i] = HUNGRY;              /* record fact that philosopher i is hungry */
    test(i);                        /* try to acquire 2 forks */
    up(&mutex);                     /* exit critical region */
    down(&s[i]);                     /* block if forks were not acquired */
}

void put_forks(i)                  /* i: philosopher number, from 0 to N-1 */
{
    down(&mutex);                   /* enter critical region */
    state[i] = THINKING;            /* philosopher has finished eating */
    test(LEFT);                     /* see if left neighbor can now eat */
    test(RIGHT);                    /* see if right neighbor can now eat */
    up(&mutex);                     /* exit critical region */
}

void test(i)                       /* i: philosopher number, from 0 to N-1 */
{
    if (state[i] == HUNGRY && state[LEFT] != EATING && state[RIGHT] != EATING) {
        state[i] = EATING;
        up(&s[i]);
    }
}
```

Szeregowanie zadań (1)

- typowe zadanie wymaga dostępu do CPU i operacji I/O
 - (a) – "CPU-bound" – przeważa zapotrzebowanie na CPU
 - (b) – "I/O-bound" – przeważa zapotrzebowanie na operacje I/O

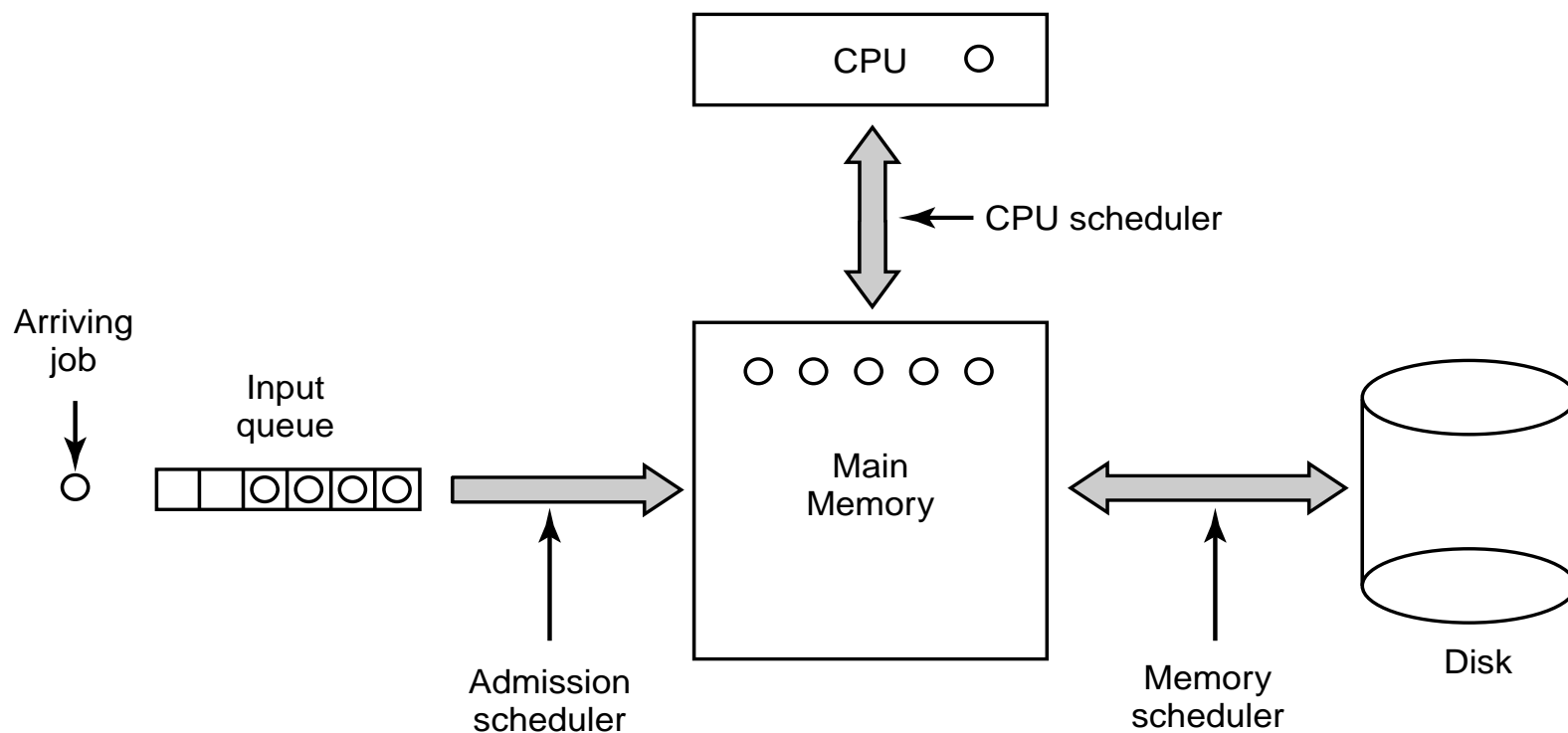


Szeregowanie zadań (2)

- główne cele algorytmów szeregowania
 - sprawiedliwy przydział zasobów (fairness)
 - wymuszanie ustalonych zasad przydziału (policy enforcement)
 - równoważenie obciążenia (balance)
- dla przetwarzania wsadowego
 - maksymalna przepustowość przetwarzania zadań (throughput)
 - minimalizacja czasu przebiegu pojedynczego zadania (turnaround)
 - maksymalne wykorzystanie CPU (CPU utilization)
- dla systemów interakcyjnych
 - minimalizacja czasu odpowiedzi (response time)
 - zaspokajanie oczekiwań użytkowników
- dla systemów czasu rzeczywistego
 - gwarantowany czas reakcji systemu
 - przetwarzanie deterministyczne

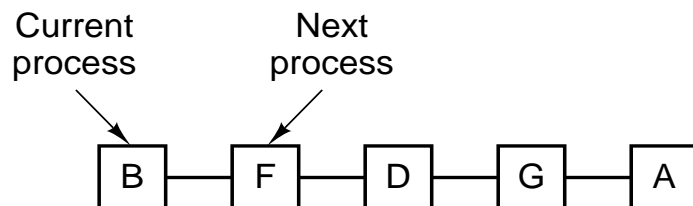
Systemy wsadowe – szeregowanie

- typowe rodzaje
 - "shortest job first"
 - wielopoziomowe, najczęściej trójpoziomowe

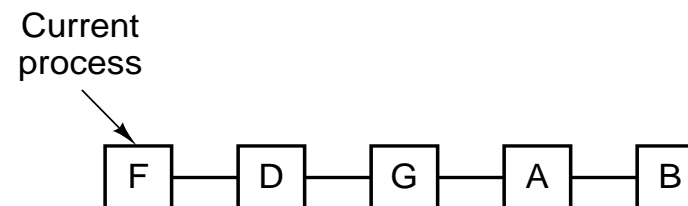


Systemy interaktywne – szeregowanie

□ szeregowanie "Round Robin"

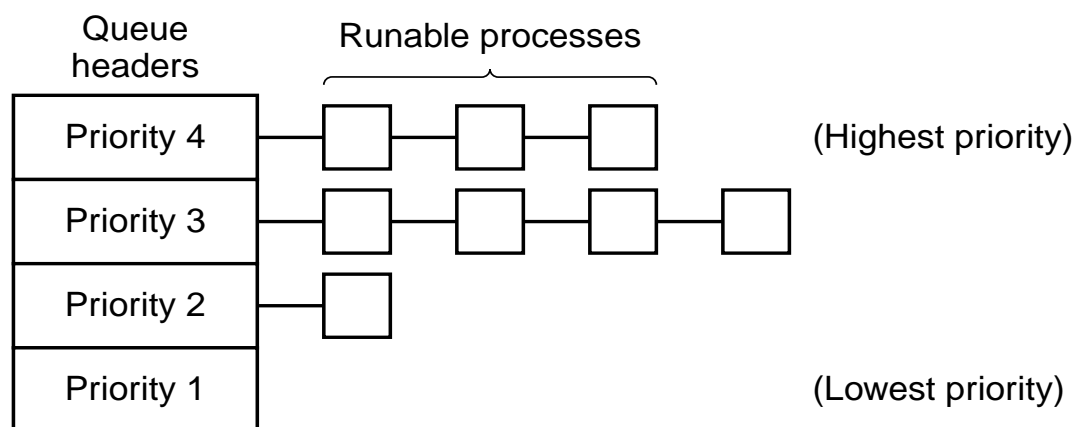


(a)



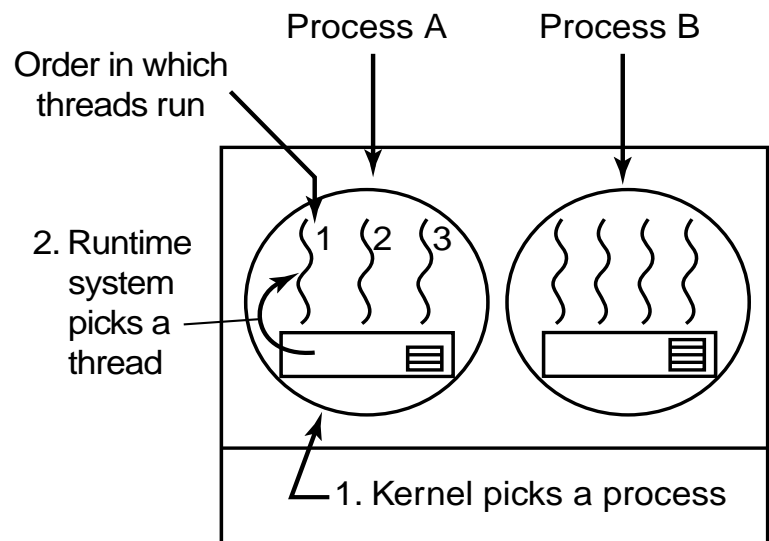
(b)

□ priorytety wykonania/dostępu



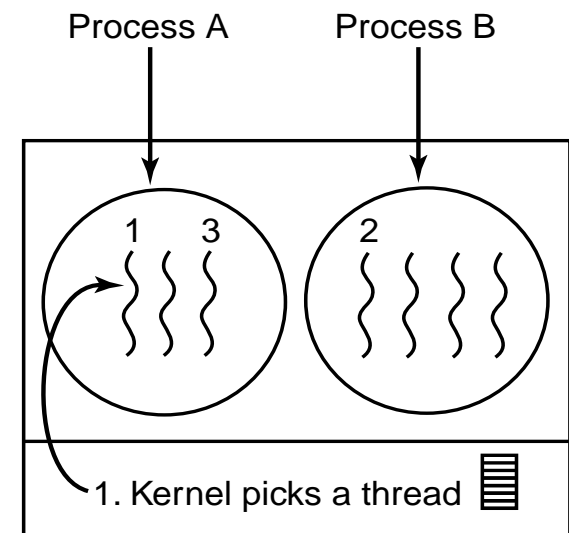
Szeregowanie wątków

- dwa podstawowe sposoby szeregowania
 - (a) – na poziomie procesu użytkownika (user-level)
 - (b) – na poziomie jądra systemu operacyjnego (kernel-level)



Possible: A1, A2, A3, A1, A2, A3
Not possible: A1, B1, A2, B2, A3, B3

(a)



Possible: A1, A2, A3, A1, A2, A3
Also possible: A1, B1, A2, B2, A3, B3

(b)

03 – Dostęp do zasobów

- ☐ Zasoby i ich udostępnianie
- ☐ Sytuacja zakleszczenia
- ☐ Modelowanie dostępu do zasobów
- ☐ Wykrywanie i likwidowanie zakleszczenia
- ☐ Unikanie i zapobieganie zakleszczeniom
- ☐ Zagadnienia pokrewne

>>>

Zasoby

- procesor centralny lub pula procesorów
 - uniprocessor – fizyczny lub emulowany (firmware, HAL, etc.)
 - multiprocessor – fizyczny (np. SMP) lub emulowany (np. HT)
 - partycja fizycznej puli procesorów (lub sprzętu)
- specjalizowane koprocесory
 - ogólnodostępne (arytmetyczny, macierzowy, etc.)
 - dedykowane (FFT, etc.)
- pamięć operacyjna
 - lokalna
 - dzielona/rozproszona
- urządzenia wejścia–wyjścia
 - niezależny kanał I/O (koprocessor I/O, lokalna pamięć operacyjna)
 - urządzenia sterowane przez procesor centralny

>>>

Udostępnianie zasobów (1)

□ procesy vs. zasoby – typowe sytuacje

- procesy wymagają dostępu do zasobów w "rozsądnej kolejności"
- procesy mogą wymagać wyłączności dostępu do określonych zasobów
- procesy mogą wymagać dostępności kilku zasobów jednocześnie
- procesy mogą zajmować/rygłować określone zasoby
- dostęp do danego zasobu może wymagać odpowiedniego reżimu czasowego
- dostęp do danego zasobu wymaga dodatkowego postępowania (np. inicjowania)
- dostęp do danego zasobu jest możliwy tylko jeden raz (!)

□ system vs. zasoby – wywłaszczanie

- zasoby wywłaszczalne – odebranie dostępu do zasobu jest możliwe bez niepożądanych skutków
- zasoby niewywłaszczalne – odebranie dostępu spowoduje załamanie wykonania

>>>

Udostępnianie zasobów (2)

- typowa sekwencja dostępu
 - żądanie dostępu do zasobu
 - użytkowanie (dedykowane lub współdzielone)
 - zwolnienie zasobu

- sytuacja odmowy dostępu
 - proces oczekujący może zostać wstrzymany
 - załamanie wykonania (zazwyczaj z kodem błędu)

- zakleszczenie
 - wstrzymanie wykonania zaangażowanych procesów
 - trwałe zajęcie zaangażowanych zasobów
 - możliwe zawieszenie pracy całego systemu (!)

>>>

Zakleszczenie

☐ definicja

- "zbiór procesów znajduje się w sytuacji zakleszczenia, jeśli każdy proces w tym zbiorze oczekuje na zdarzenie, które może zostać spowodowane wyłącznie przez inny proces z tego zbioru"
- oczekiwane zdarzenie to najczęściej zwolnienie zasobu zajmowanego przez inny proces

☐ uwarunkowania

- dostępność zasobu na zasadach wzajemnego wykluczania
- procesy zajmujące zasoby mogą żądać kolejnych zasobów
- wywłaszczenie zasobu nie jest możliwe
- sytuacja oczekiwania cyklicznego (procesy oczekują na zwolnienie zasobu zajmowanego przez następny proces)

☐ skutki

- niemożność dalszego wykonywania
- niemożność zwolnienia zasobów
- niemożność uaktywnienia procesu

>>>

Zakleszczenia – strategie postępowania

- "algorytm ostrygi", "algorytm strusia" (Ostrich Algorithm)
 - założenie: zakleszczenia nie wystąpią, a jeśli już, to
 - wystąpienie zakleszczenia jest bardzo mało prawdopodobne, a
 - systemowy koszt obsługi zakleszczenia jest bardzo duży :–)
 - szeroko stosowany: Windows (!!!), UNIX (!), etc.

- wykrywanie i likwidowanie
 - wymaga modelowania możliwych sytuacji
 - wymaga niezależnego (!) systemu obsługi zakleszczeń

- unikanie
 - odpowiedni system przydziału zasobów

- zapobieganie
 - usunięcie/redukcja uwarunkowań

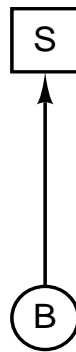
>>>

Zakleszczenia – modelowanie (1)

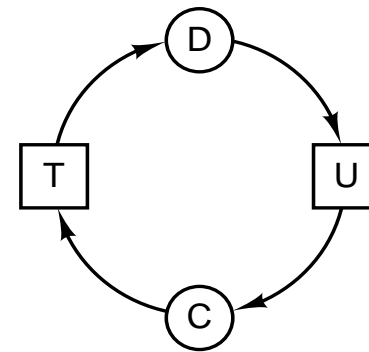
- grafy przydziału zasobów
 - (a) – zasób R przydzielony procesowi A
 - (b) – proces B oczekujący na zasób S
 - (c) – zakleszczenie: procesy C i D vs. zasoby T i U



(a)



(b)



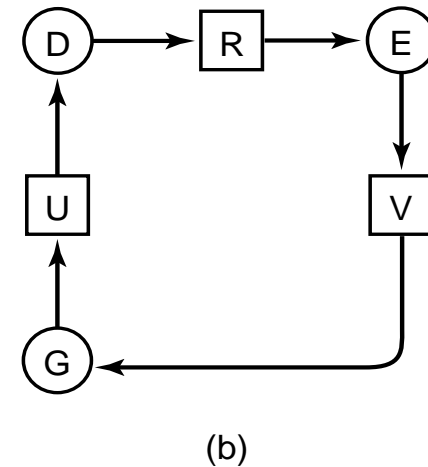
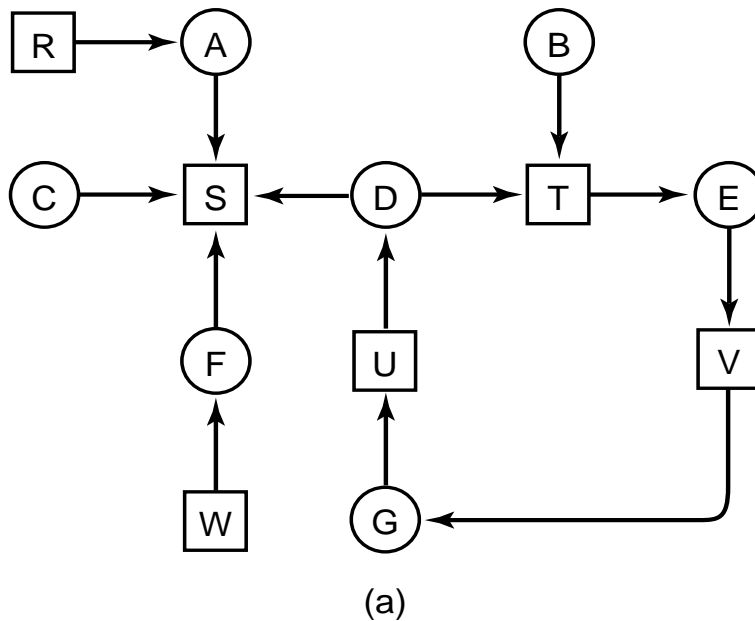
(c)

>>>

Zakleszczenia – modelowanie (2)

□ przykład

- (a) – kompletny graf przydziału zasobów
- (b) – cykl odpowiadający zakleszczeniu



Zakleszczenia – modelowanie (3)

- formalna reprezentacja
 - macierz przydziału zasobów (allocation matrix)
 - macierz zamówień (request matrix)

Resources in existence
($E_1, E_2, E_3, \dots, E_m$)

Current allocation matrix

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & \cdots & C_{1m} \\ C_{21} & C_{22} & C_{23} & \cdots & C_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ C_{n1} & C_{n2} & C_{n3} & \cdots & C_{nm} \end{bmatrix}$$

Row n is current allocation
to process n

Resources available
($A_1, A_2, A_3, \dots, A_m$)

Request matrix

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & \cdots & R_{1m} \\ R_{21} & R_{22} & R_{23} & \cdots & R_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ R_{n1} & R_{n2} & R_{n3} & \cdots & R_{nm} \end{bmatrix}$$

Row 2 is what process 2 needs

>>>

Zakleszczenia – modelowanie (4)

□ przykład

$$\begin{array}{c} \text{Tape drives} \\ \text{Plotters} \\ \text{Scanners} \\ \text{CD Roms} \end{array} \\ E = (4 \quad 2 \quad 3 \quad 1)$$

$$\begin{array}{c} \text{Tape drives} \\ \text{Plotters} \\ \text{Scanners} \\ \text{CD Roms} \end{array} \\ A = (2 \quad 1 \quad 0 \quad 0)$$

Current allocation matrix

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

Request matrix

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

>>>

Zakleszczenia – likwidowanie

☐ metoda wywłaszczania

- odebranie potrzebnego zasobu innemu procesowi
- przydział zasobu na określony czas, po upływie którego zasób podlega wywłaszczeniu, bez względu na zapotrzebowanie
- j/w, ale w zależności od zapotrzebowania i/lub priorytetu
- ograniczone zastosowanie, zależnie od natury zasobu
- wymaga (quasi)niezależnego podsystemu obsługi

☐ metoda odtwarzania

- cykliczny zapis "obrazu" procesu (checkpointing)
- odtworzenie/restart procesu jeśli wystąpiło zakleszczenie
- wymaga (quasi)niezależnego, obciążającego podsystemu obsługi

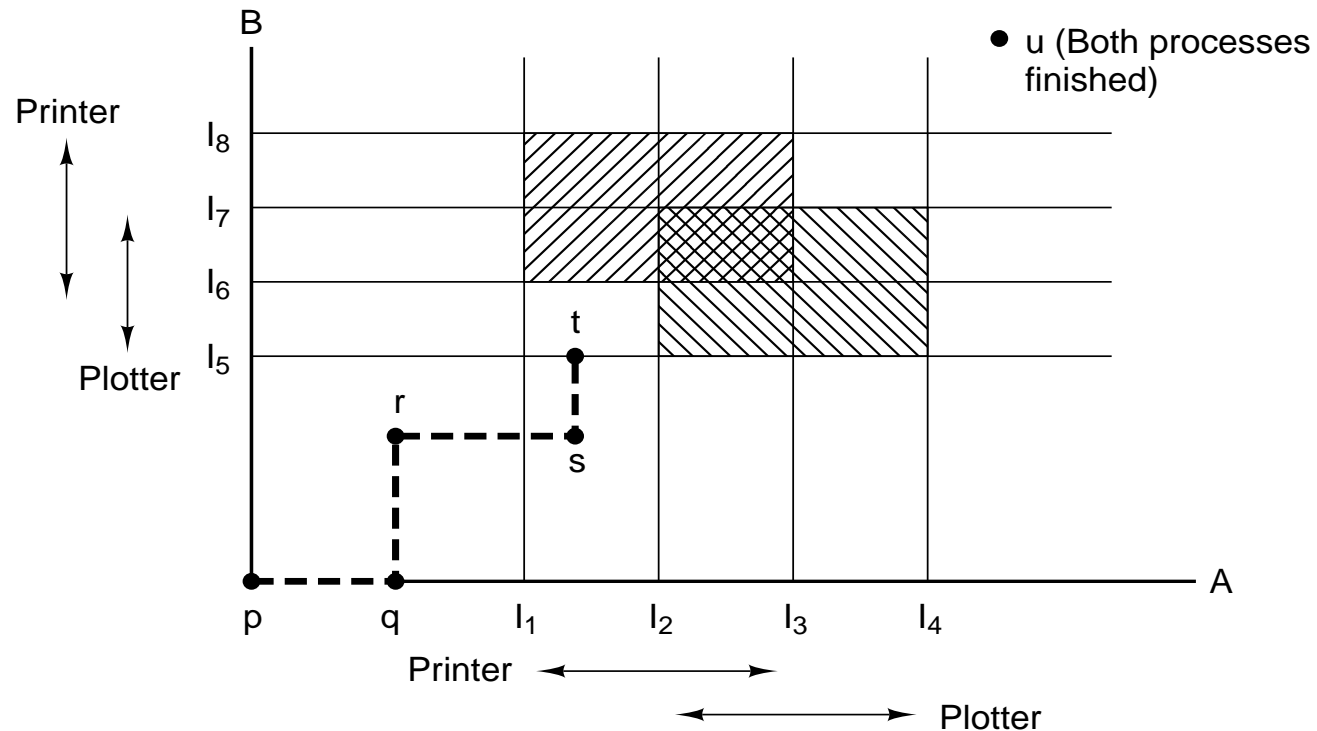
☐ metoda usuwania

- usunięcie jednego lub kilku zaangażowanych procesów
- możliwy wybór procesu do usunięcia (wg. priorytetu, stanu, etc.)
- najprostsza metoda likwidacji zakleszczeń (!)
- ograniczona skuteczność ("zombie")

>>>

Zakleszczenia – unikanie (1)

- trajektorie dostępu do zasobów
 - procesy A i B wymagają dostępu do drukarki i do plotera



Zakleszczenia – unikanie (2)

□ algorytm bankiera

- przydział zasobów nie może uniemożliwić zaspokojenia potrzeb wykonywanych procesów

	Process	Tape drives	Plotters	Scanners	CD ROMs
A	3	0	1	1	
B	0	1	0	0	
C	1	1	1	0	
D	1	1	0	1	
E	0	0	0	0	

Resources assigned

	Process	Tape drives	Plotters	Scanners	CD ROMs
A	1	1	0	0	
B	0	1	1	2	
C	3	1	0	0	
D	0	0	1	0	
E	2	1	1	0	

Resources still needed

E = (6342)
P = (5322)
A = (1020)

>>>

Zakleszczenia – zapobieganie (1)

- problem wzajemnego wykluczania
 - spooling (Simultaneous Peripheral Operations OnLine)
 - unikanie wyłącznego przydziału zasobów
- problem jednoczesnego zajmowania/zamawiania
 - zamawianie wszystkich potrzebnych zasobów przed uruchomieniem
 - zamawianie zasobów niezbędnych w danej chwili (interwale)
- problem niemożności wywłaszczenia
 - odbieranie zasobów z możliwością późniejszej kontynuacji
 - kolejkovanie zamówień, jeśli kontynuacja nie jest możliwa
- problem oczekiwania cyklicznego
 - sekwencyjne uporządkowanie udostępnianych zasobów
 - przydział zasobów w odpowiedniej kolejności

>>>

Zakleszczenia – inne zagadnienia

- dwuetapowe zajmowanie zasobów
 - I etap – zamawianie potrzebnych zasobów pojedynczo, do momentu wyczerpania listy lub stwierdzenia zajęcia przez inny proces; restart, jeśli lista zasobów jest niepełna
 - II etap – właściwe wykonanie, ze zwalnianiem niepotrzebnych zasobów na bieżąco
- inne zakleszczenia (non–resource deadlocks)
 - zakleszczenie w wyniku wzajemnego oczekiwania na wykonanie pewnego zadania
 - problem dwóch semaforów – zaangażowane procesy muszą wykonać np. operację down() na dwóch semaforach (mutex i inny) w określonej kolejności
- problem "głodzenia procesu"
 - najczęściej wynik działania algorytmu "Shortest Job First" (SJF)
 - długo wykonujące się zadanie może nigdy nie otrzymać zasobu (!)

===

04 – Zarządzanie pamięcią

- ☐ Podstawy
- ☐ Wymiana i stronicowanie
- ☐ Pamięć wirtualna
- ☐ Segmentacja
- ☐ Zagadnienia implementacyjne

>>>

Zarządzanie pamięcią – podstawy (1)

□ "pamięć idealna"

- jednorodna
- nieograniczona
- szybka
- nieulotna :–)

□ "pamięć realna"

- niejednorodna, "ekonomiczna" hierarchia pamięci, w tym:
- niewielka ilość szybkiej pamięci (rejstry, cache, etc.)
- wystarczająca ilość średnio–szybkiej pamięci operacyjnej
- nadmiarowa ilość pamięci masowej, tym więcej im wolniej pracuje :–)
- ulotna, zmienna trwałość zapisu: od kilku ns do kilkuset lat

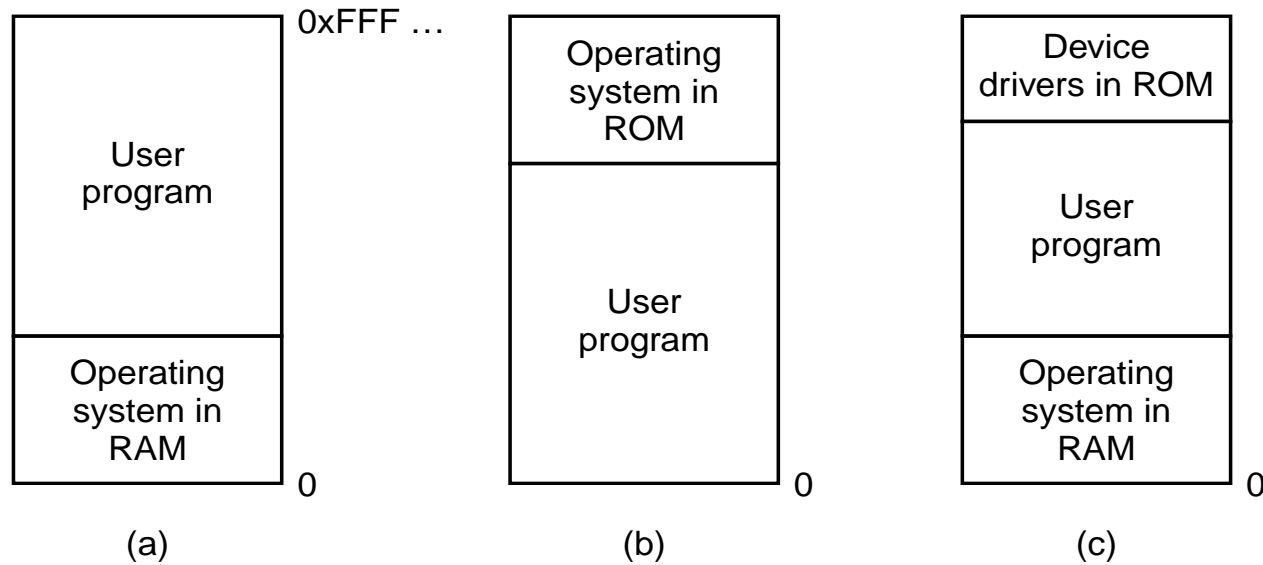
□ podstawowe zagadnienia

- obsługa hierarchii pamięci
- "przeźroczystość" dostępu

>>>

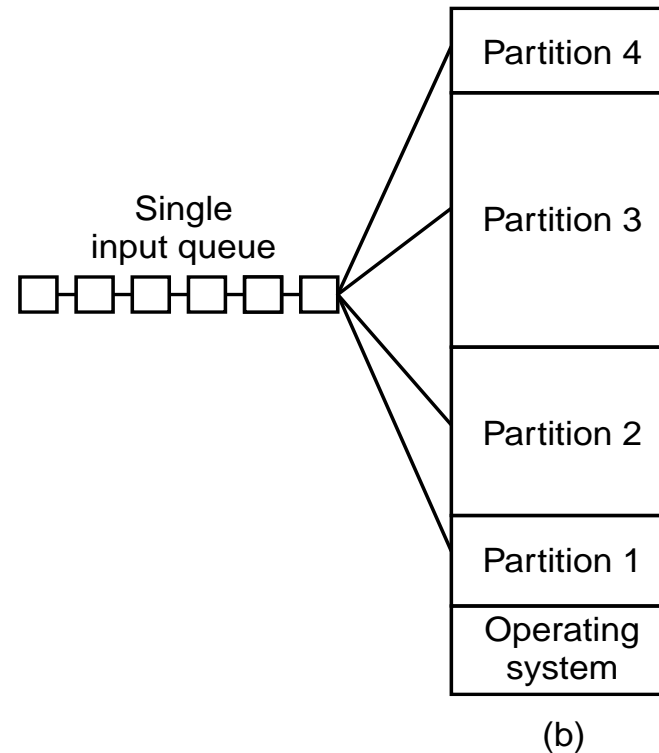
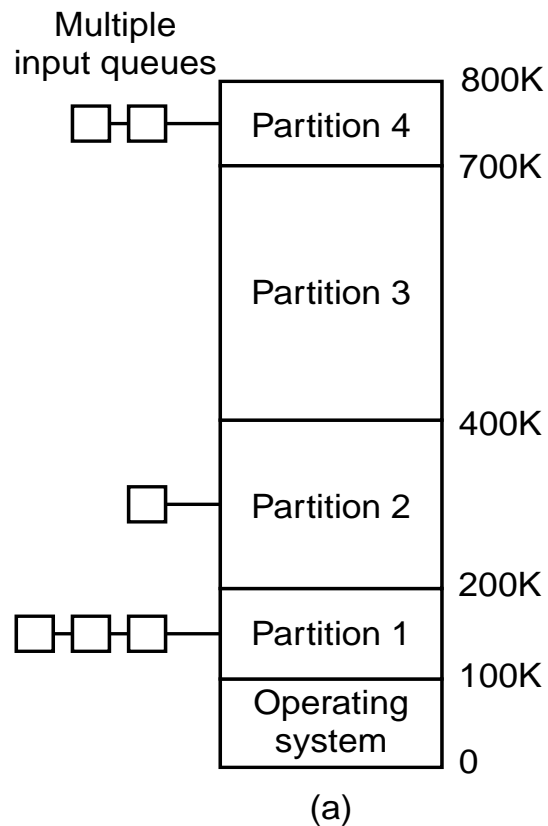
Zarządzanie pamięcią – podstawy (2)

- prosty system jednoprogramowy
 - (a) – rezydentny (RAM)
 - (b) – rezydentny (ROM)
 - (c) – modułarny (RAM+ROM)



Zarządzanie pamięcią – podstawy (3)

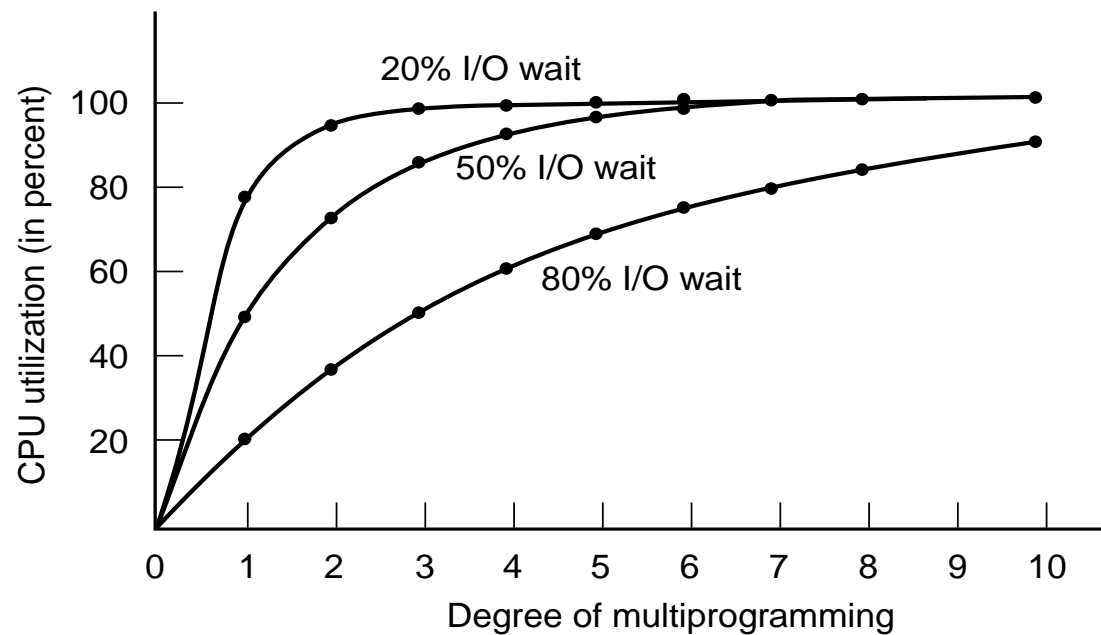
- partycje o ustalonej wielkości
 - (a) – niezależne kolejki zadań dla każdej partycji
 - (b) – pojedyncza, wspólna kolejka dla wszystkich partycji



Wieloprogramowość – modelowanie

□ wykorzystanie CPU

- rośnie ze wzrostem ilości aktywnych procesów
- maleje przy oczekiwaniu na operacje We/Wy
- wieloprogramowość istotnie zmniejsza skutki oczekiwania na We/Wy



Wieloprogramowość a wydajność

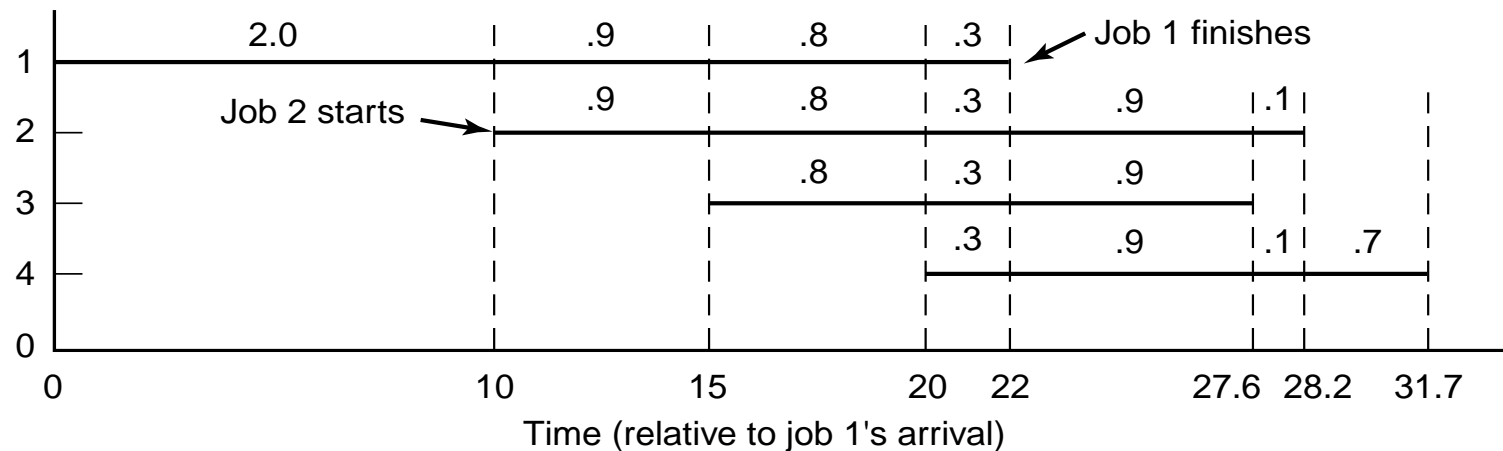
- przykład: cztery procesy, I/O wait=80%

Job	Arrival time	CPU minutes needed
1	10:00	4
2	10:10	3
3	10:15	2
4	10:20	2

(a)

	# Processes			
	1	2	3	4
CPU idle	.80	.64	.51	.41
CPU busy	.20	.36	.49	.59
CPU/process	.20	.18	.16	.15

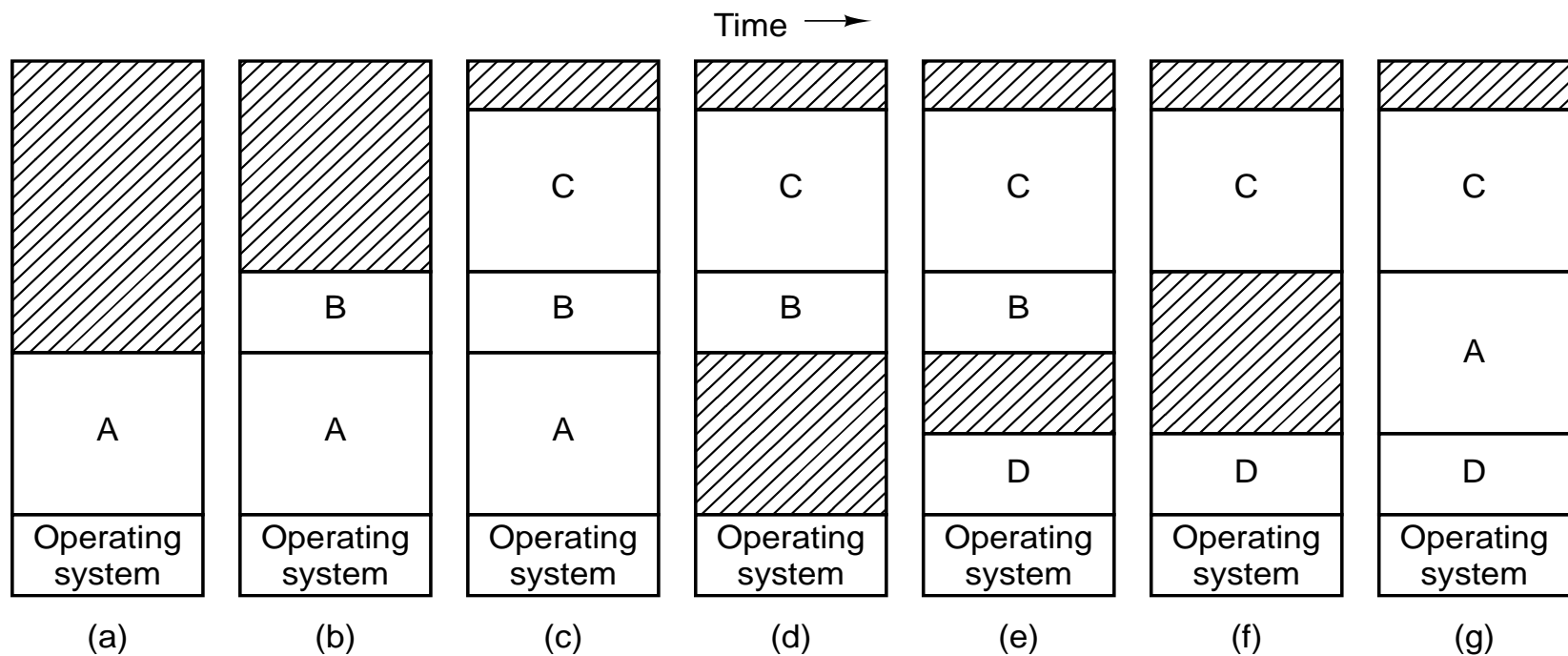
(b)



(c)

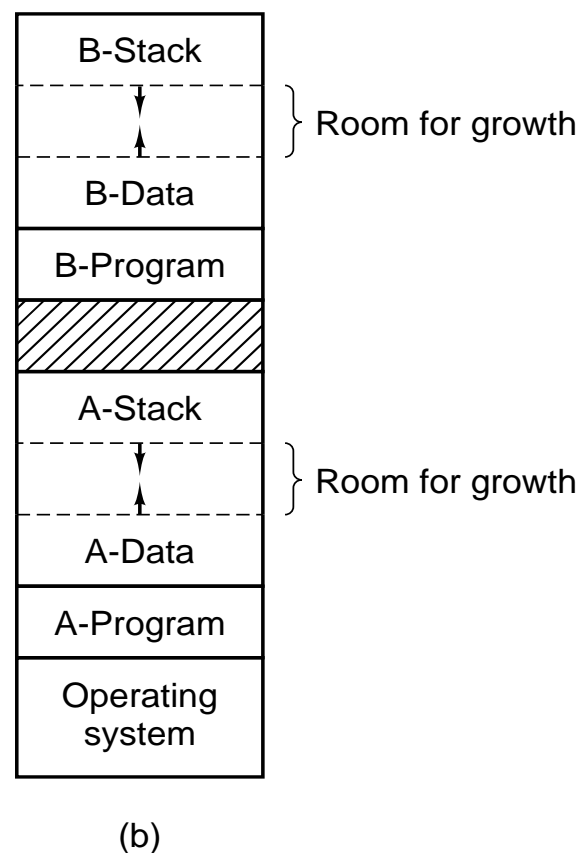
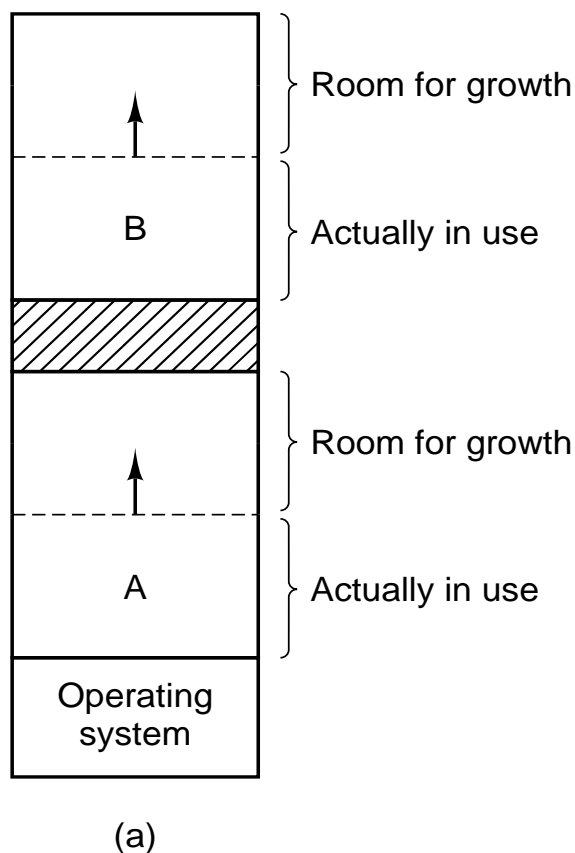
Dynamiczna alokacja pamięci (1)

- wymaga relokacji i ochrony
 - "przesuwalny" kod i dane
 - ograniczenie dostępu do przydzielonej partycji
 - adresowanie fizyczne: adres bazowy + przesunięcie, adres graniczny
- przykładowa zajętość pamięci w funkcji czasu



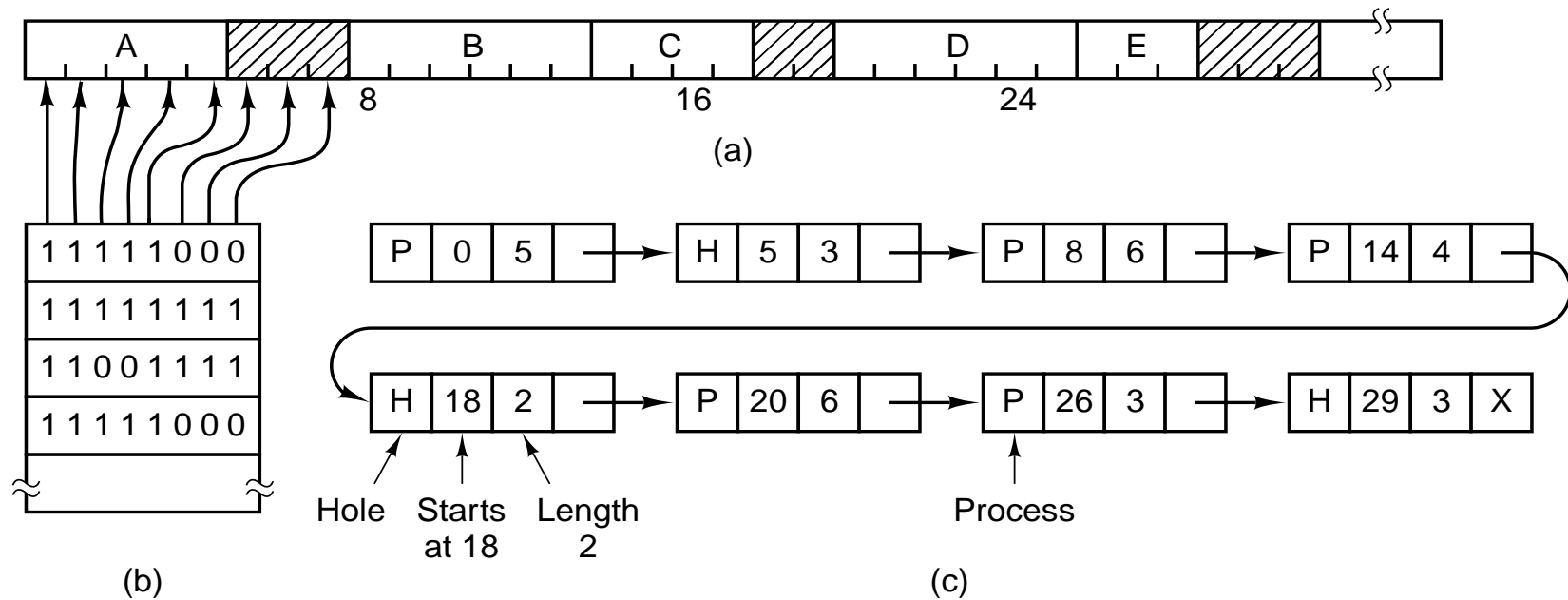
Dynamiczna alokacja pamięci (2)

- prealokacja – przydział "na zapas"
 - (a) – dla rosnącego segmentu danych
 - (b) – dla rosnącego segmentu danych i stosu



Dynamiczna alokacja pamięci (3)

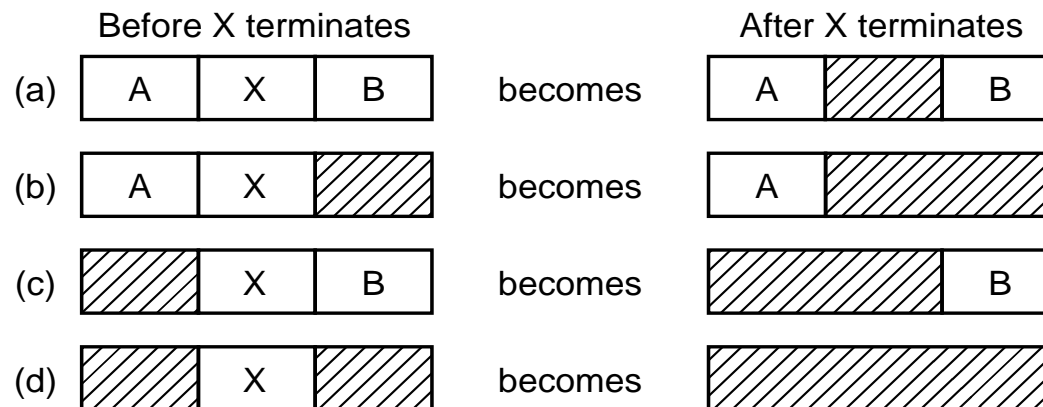
- przydział wg. mapy bitowej
 - (a) – 5 procesów (A–E), 3 "dziury", ustalone jednostki alokacyjne
 - (b) – odpowiednia mapa bitowa
 - (c) – reprezentacja w postaci listy połączonych bloków



Dynamiczna alokacja pamięci (4)

□ zwalnianie pamięci

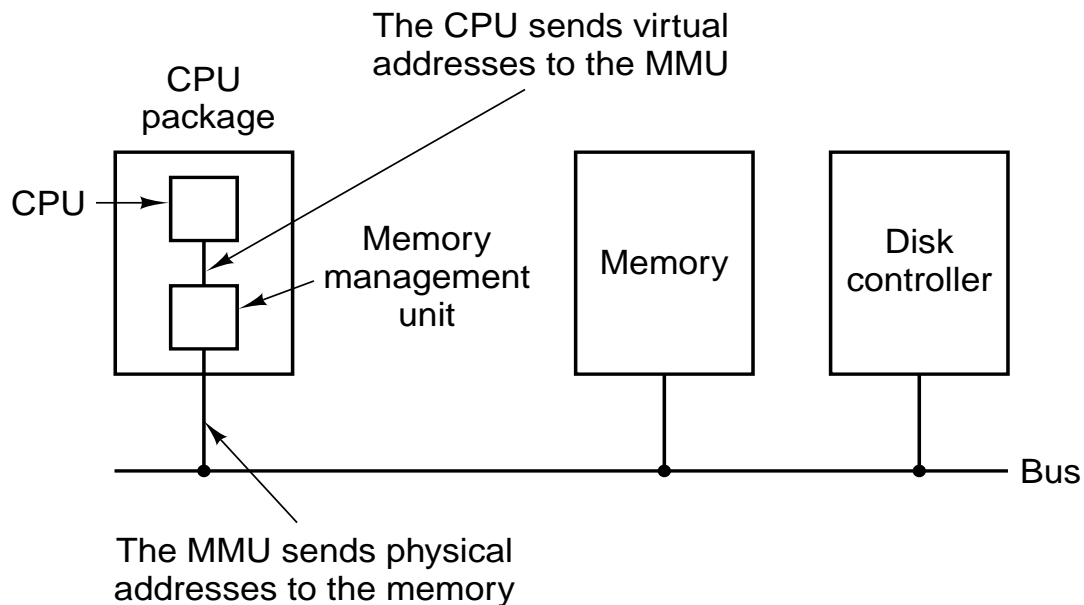
- cztery możliwości dla usuwanego procesu X



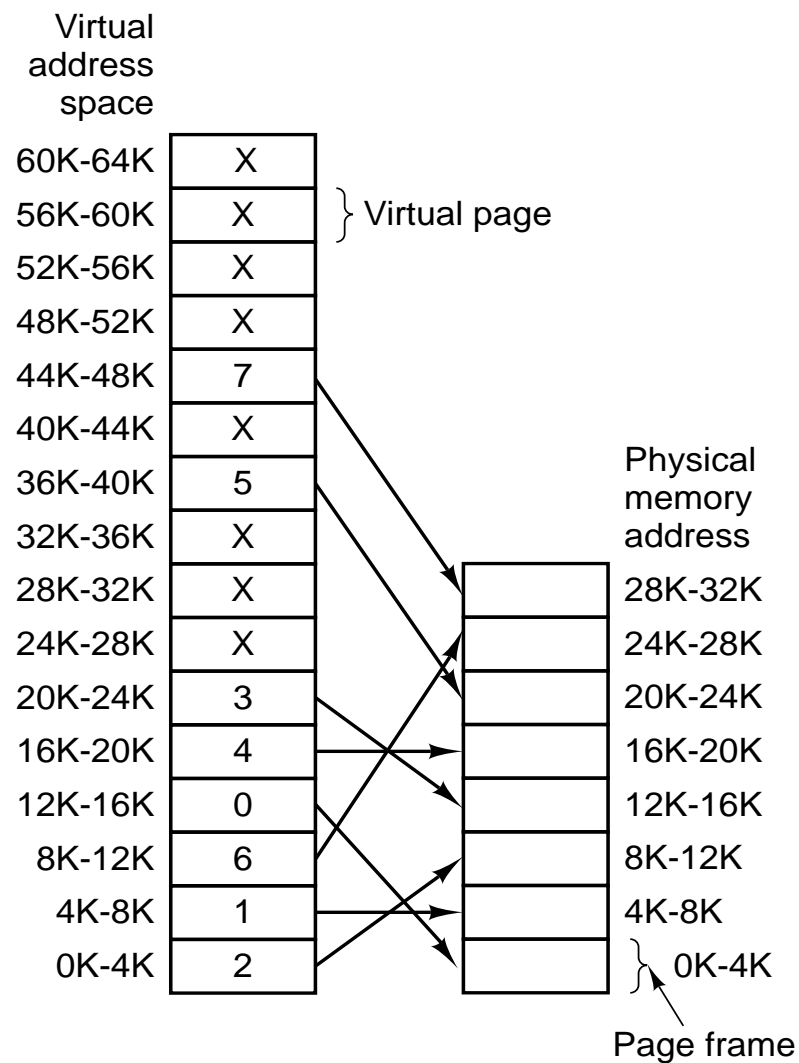
- reprezentacja w postaci listy połączonych bloków zapewnia tu lepszą funkcjonalność w porównaniu z mapą bitową
- jednostki alokacyjne: "strony" pamięci obsługiwane sprzętowo

Pamięć wirtualna

- abstrakcja pamięci fizycznej
 - obsługiwana sprzętowo przez MMU (Memory Management Unit)
 - CPU wykorzystuje (abstrakcyjne) adresy wirtualne
 - MMU zapewnia odpowiednią konwersję adresów

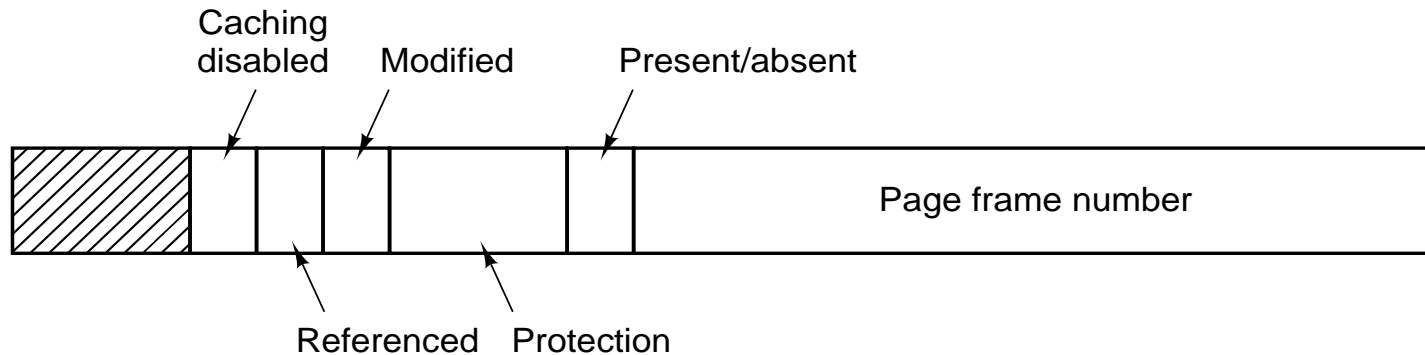


Stronicowanie



Tablica stron

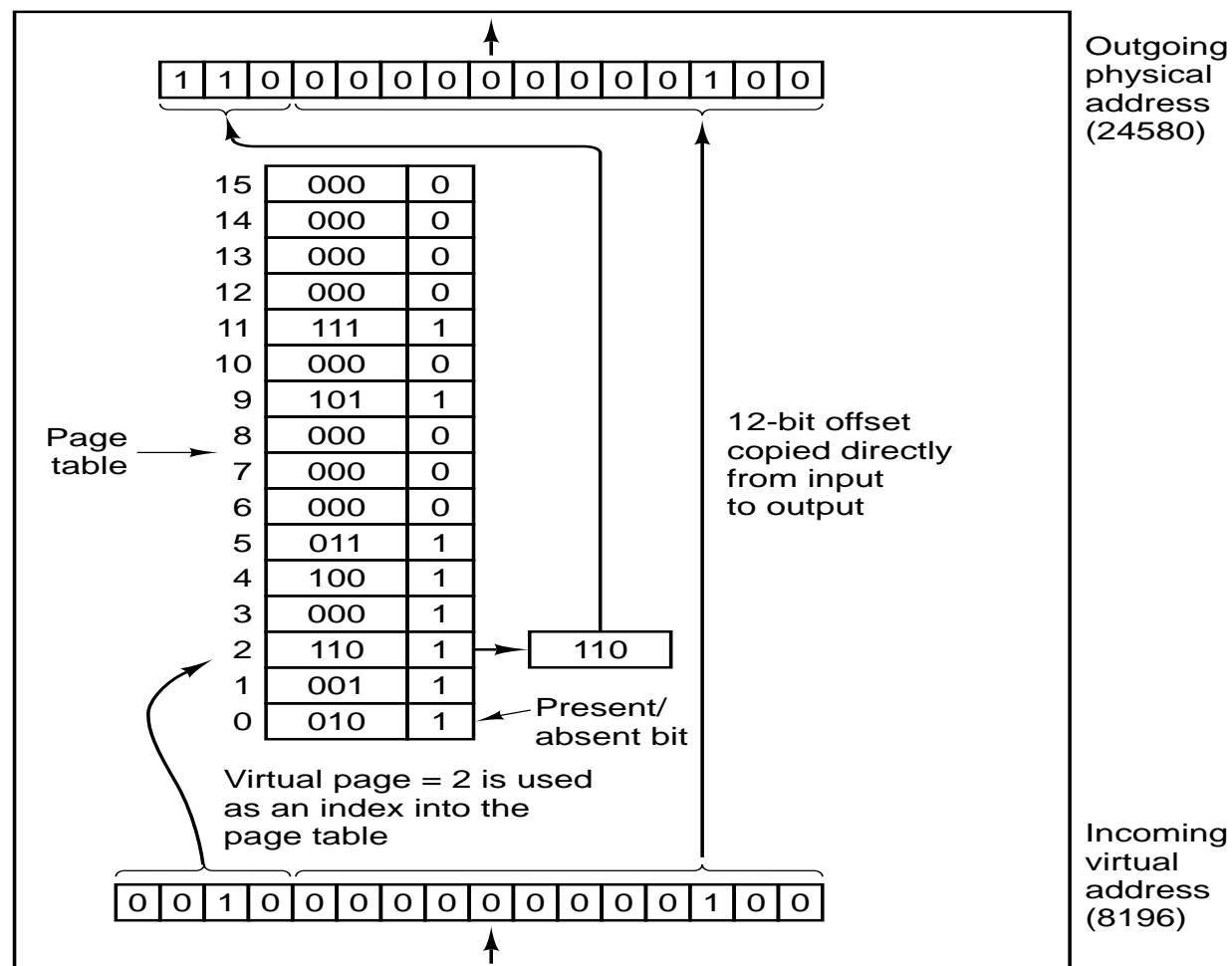
- odwzorowanie: strona wirtualna \longleftrightarrow strona fizyczna
 - typowa struktura elementu tablicy stron



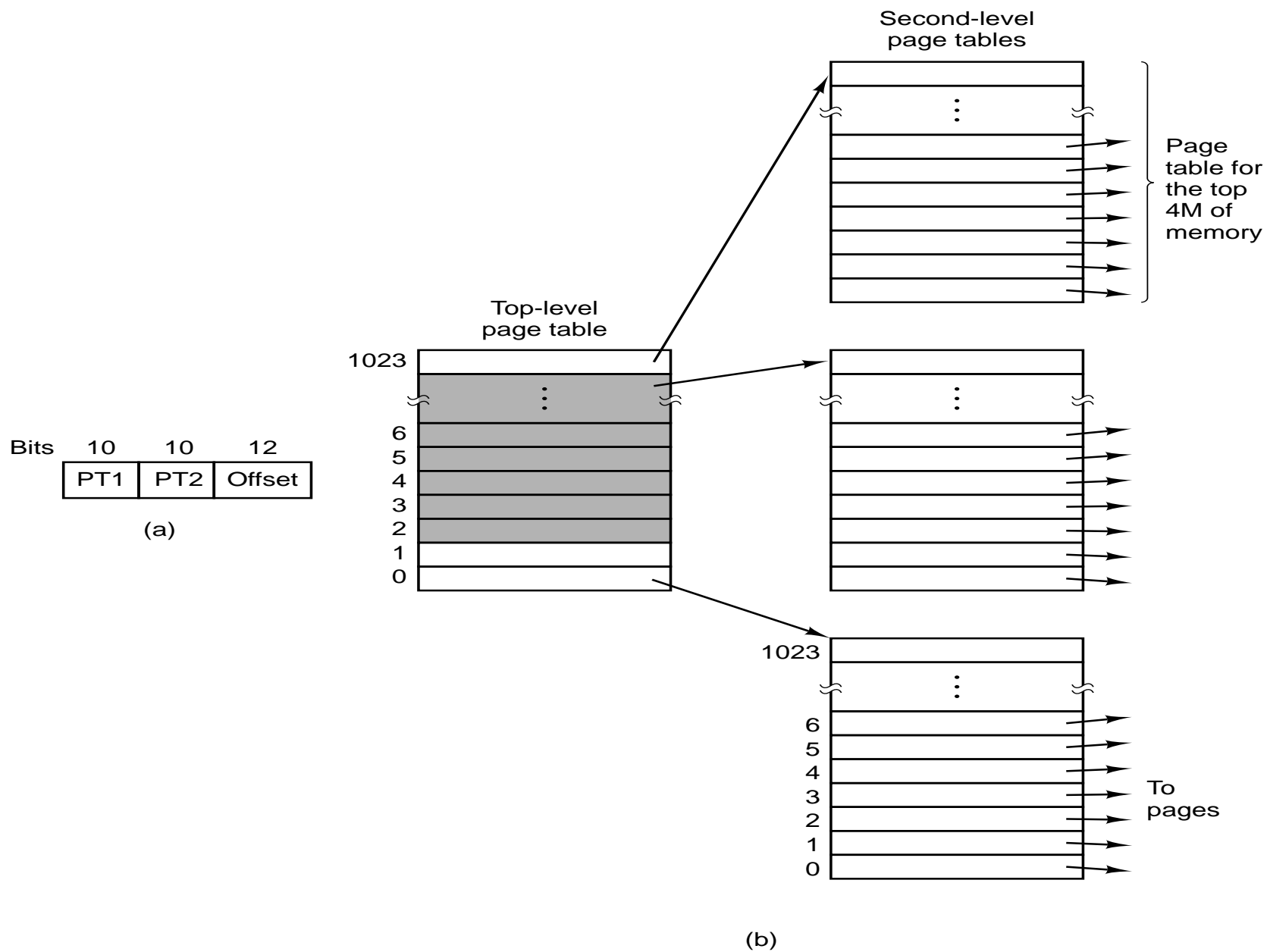
- znaczniki umożliwiają określenie stanu strony/ramki
- uprawnienia określają tryb dostępu (np. R,W,X)
- struktura wielopoziomowa umożliwia zmniejszenie wielkości tablicy stron

MMU – konwersja adresów

- przykład: tabela 16 stron po 4 KB



Stronicowanie dwupoziomowe



Bufory TLB

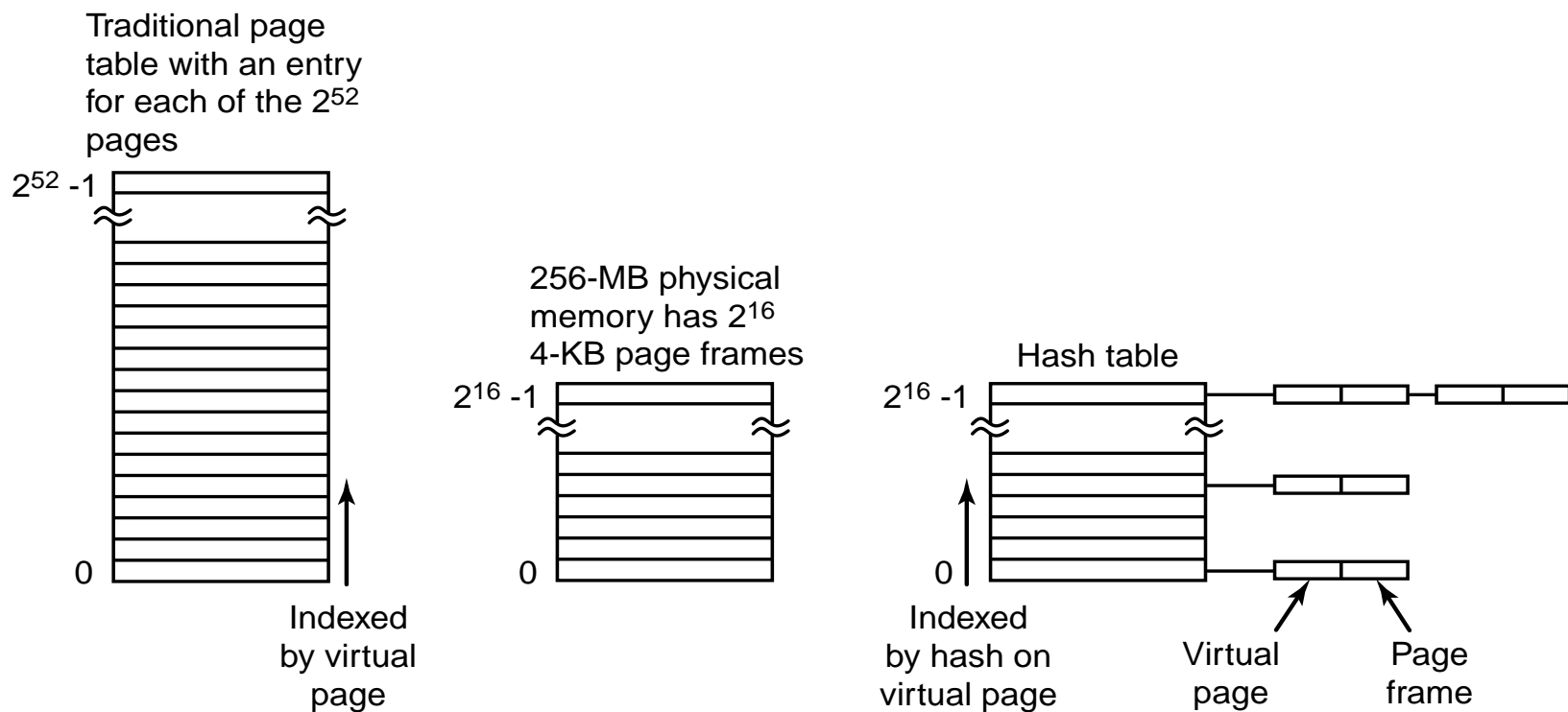
□ Translation Lookaside Buffers

- sprzętowo obsługiwana, bardzo szybka pamięć podręczna
- umożliwia znaczące przyspieszenie pracy MMU

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Odwrócona tablica stron

- powiązanie obszaru pamięci z odpowiednimi stronami
 - umożliwia dopasowanie stronicowania do wielkości pamięci fizycznej
 - umożliwia łatwą lokalizację stron odpowiadających np. procesowi
 - obsługa odbywa się zazwyczaj na poziomie systemu operacyjnego



Zastępowanie stron (1)

- konieczne po błędzie strony/stronicowania
 - wybór strony do usunięcia, lub
 - przydzielenie wolnego miejsca
 - strona zmodyfikowana musi zostać wpierw zapisana
 - często używana strona nie powinna zostać usunięta
- rozwiązania optymalne
 - przygotowanie zawczasu miejsca dla wszystkich potrzebnych stron (niemożliwe w ogólnym przypadku)
 - śledzenie wykonania procesu z zapisem historii stronicowania, do wykorzystania przy kolejnym uruchomieniu (niepraktyczne)
- algorytm NRU (Not Recently Used)
 - problem: określenie czasu ostatniego użycia ramki
 - typowe rozwiązanie: klasyfikacja ramek wg. sposobu użycia
 - najczęściej: klasyfikacja wg. znaczników 'Referenced' i 'Modified'
 - decyzja NRU: $R=M=0$ lub tylko $R=0$ (!)

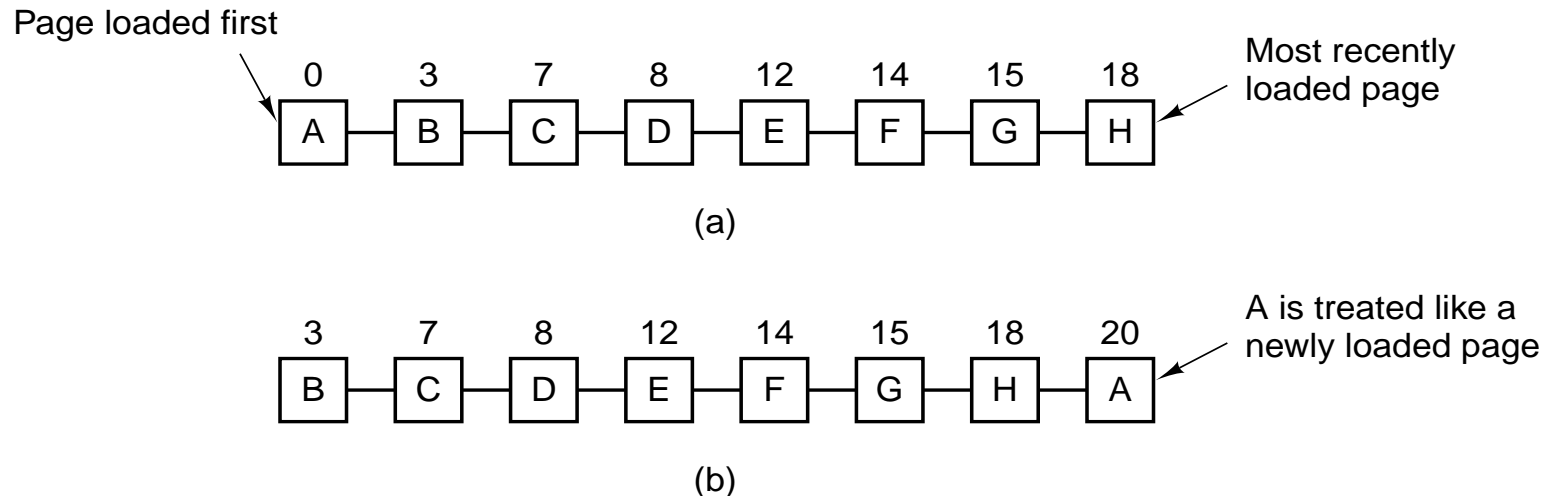
Zastępowanie stron (2)

□ algorytm FIFO

- wymaga utworzenia listy stron/ramek w kolejności użycia
- zastępowanie stron odbywa się wg. porządku na tej liście
- główna wada: strona najdłużej utrzymywana w pamięci może być również najczęściej używana (!)

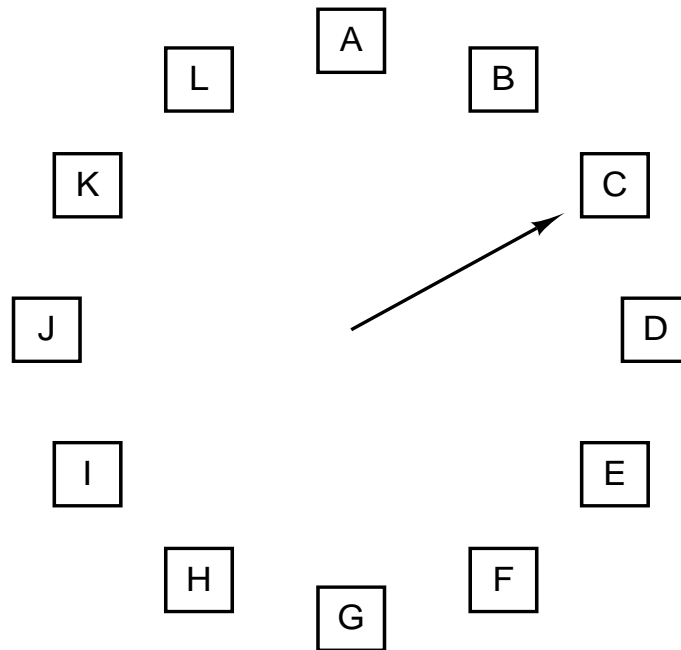
□ algorytm drugiej szansy

- (a) strony uporządkowane wg. czasu wprowadzenia do pamięci
- (b) strona A dostaje "drugą szansę", jeśli bit R=1 (Referenced)



Zastępowanie stron (3)

- wersja cykliczna ("zegarowa") algorytmu drugiej szansy



When a page fault occurs,
the page the hand is
pointing to is inspected.
The action taken depends
on the R bit:

R = 0: Evict the page

R = 1: Clear R and advance hand

Zastępowanie stron (4)

□ algorytm LRU (Least Recently Used)

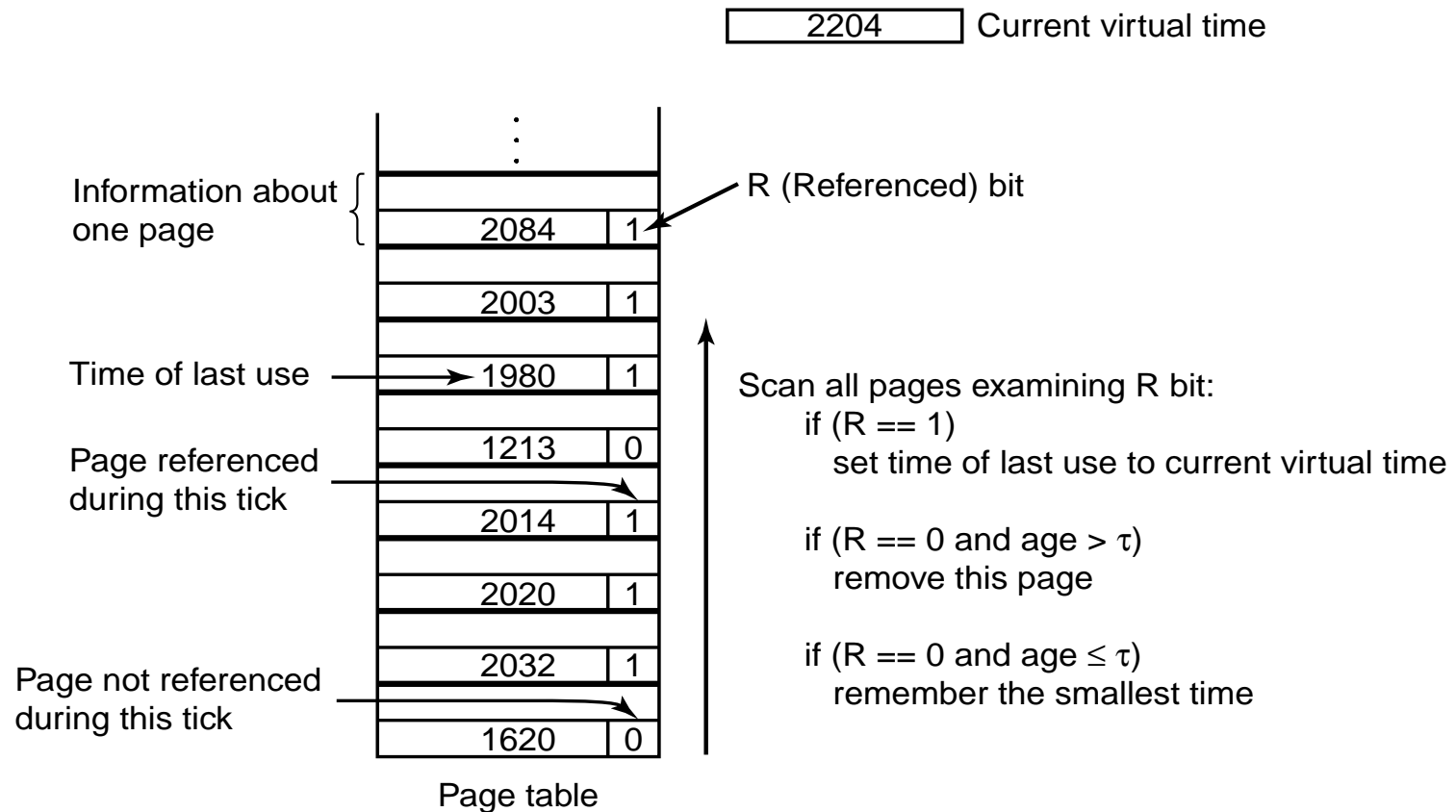
- założenie: strony/ramki ostatnio używane będą wkrótce potrzebne
- konieczność utrzymywania listy stron/ramek uporządkowanej w kolejności użycia
- lista musi być aktualizowana przy każdym odwołaniu do pamięci (!)
- alternatywnie: licznik czasu dla każdego elementu tablicy stron
- złożona implementacja
- generuje stosunkowo duży narzut systemowy

□ model zbioru roboczego (Working–Set Model)

- założenie: przetwarzanie odbywa się strefowo
- zbiór roboczy odpowiada strefie procesu/zadania
- wielkość strefy szacuje parametr "WS–window", określany jako ilość odwołań do stron/ramek strefy lub odpowiedni przedział czasowy

Zastępowanie stron (5)

- algorytm zastępowania wg. modelu zbioru roboczego
 - wersja wykorzystująca automatycznie ustalany przedział czasowy



Zastępowanie stron – porównanie

Algorithm	Comment
Optimal	Not implementable, but useful as a benchmark
NRU (Not Recently Used)	Very crude
FIFO (First-In, First-Out)	Might throw out important pages
Second chance	Big improvement over FIFO
Clock	Realistic
LRU (Least Recently Used)	Excellent, but difficult to implement exactly
NFU (Not Frequently Used)	Fairly crude approximation to LRU
Aging	Efficient algorithm that approximates LRU well
Working set	Somewhat expensive to implement
WSClock	Good efficient algorithm

Szamotanie

□ definicja

- występuje, jeśli obsługa stronicowania zajmuje więcej czasu niż wykonanie procesu

□ przyczyny

- nadmierna ilość równocześnie wykonujących się procesów (stopień wieloprogramowości)
- niewłaściwa praca podsystemu szeregowania zadań, automatycznie zwiększając stopień wieloprogramowości jeśli spada obciążenie CPU
- niewłaściwie dobrany algorytm zastępowania stron
- nadmierne obciążenie dysku wykorzystywanego do wymiany stron

□ rozwiązania

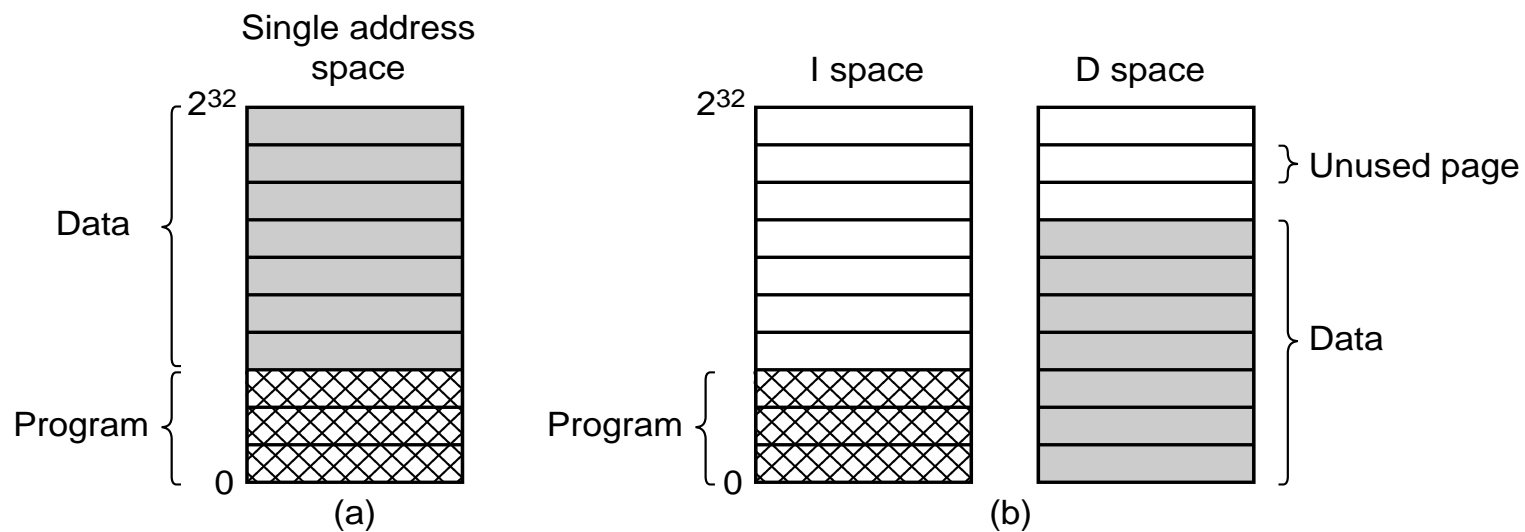
- generalnie: ograniczenie ilości procesów wymagających pamięci
- stronicowanie na żądanie (paging on demand), najlepiej od najwcześniejszych etapów wykonania
- stronicowanie wstępne (pre-paging), zmniejsza ilość błędów braku strony na początku wykonania

Rozmiar strony

- ustalany na etapie projektowania sprzętu
- niekiedy zmienny w określonych granicach
 - IBM S/370 – 2 KB i 4 KB
 - Intel IA32 – 4 KB i 4 MB
 - Motorola 68030+ – 256 B do 32 KB
- mały rozmiar strony
 - mniejsza fragmentacja (wewnętrzna)
 - lepsze wykorzystanie pamięci
 - dopasowanie do wielkości bloku dyskowego
 - duże, wielopoziomowe tablice stron
 - kosztowna obsługa stronicowania

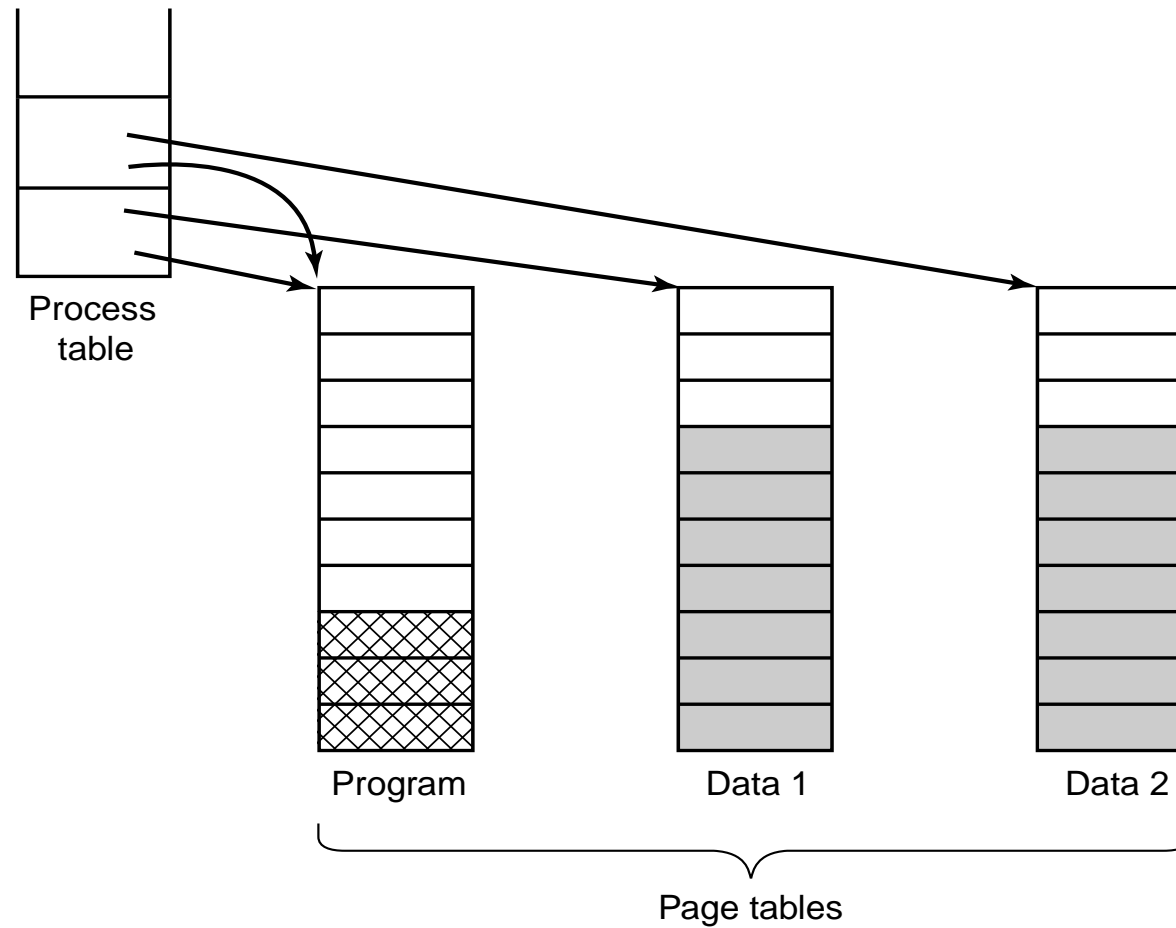
Przydział pamięci

- dwa modele
 - (a) – wspólny dla instrukcji i danych
 - (b) – rozdzielny



Współdzielenie stron pamięci

- dwa procesy wykonujące ten sam program

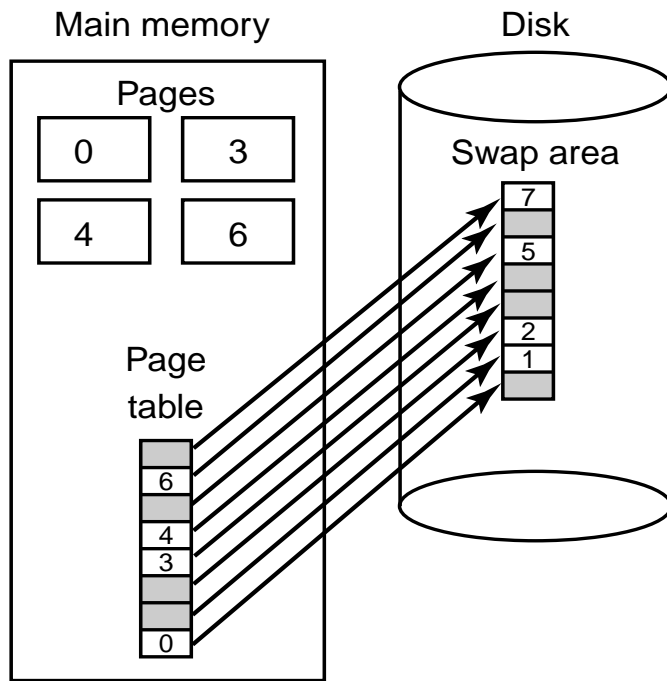


Stronicowanie a system operacyjny

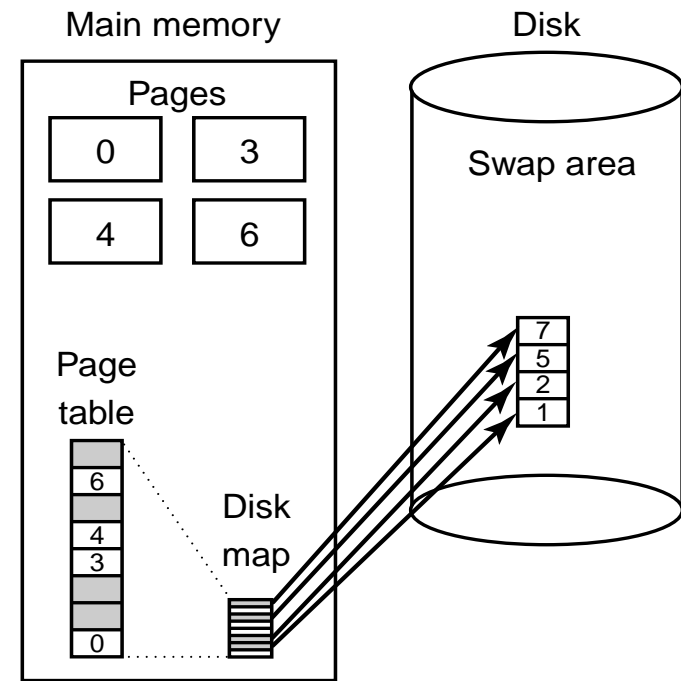
- tworzenie nowego procesu
 - określenie (inicjalnego) rozmiaru programu
 - zbudowanie tablicy stron, wprowadzenie stron do pamięci
- rozpoczęcie wykonania
 - inicjowanie sprzętu (MMU) i pamięci podręcznej (TLB)
- obsługa błędu strony
 - obsługa odpowiedniego wyjątku/przerwania
 - określenie przyczynowego adresu wirtualnego
 - zastąpienie odpowiedniej strony wymaganą stroną (np. z dysku)
 - aktualizacja tablicy stron, odtworzenie rejestrów, etc.
 - wznowienie wykonania
- zakończenie wykonania
 - zwolnienie odpowiednich stron
 - likwidacja tablicy stron

Wymiana stron z udziałem dysku

- podstawowe dwa rodzaje
 - (a) – statyczny obszar wymiany (partycja wymiany)
 - (b) – dynamiczny obszar wymiany (plik wymiany)



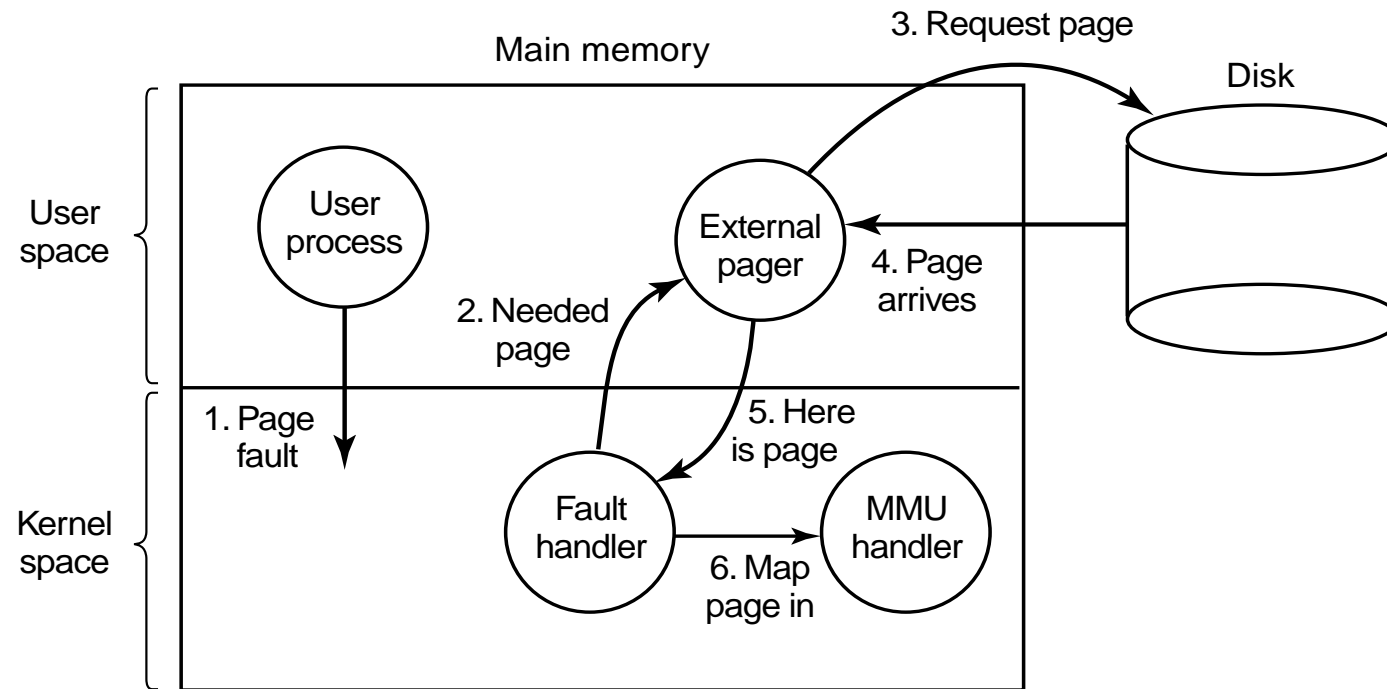
(a)



(b)

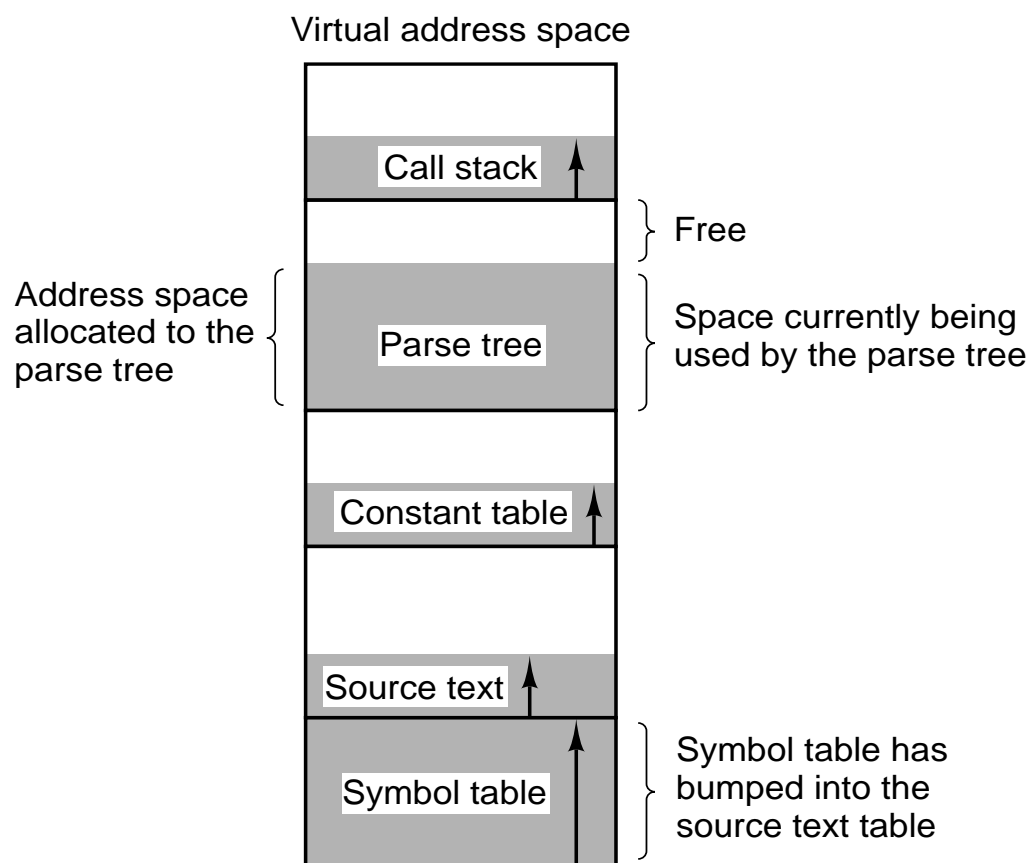
Obsługa błędu strony z wymianą

- separacja obsługi wymiany dyskowej



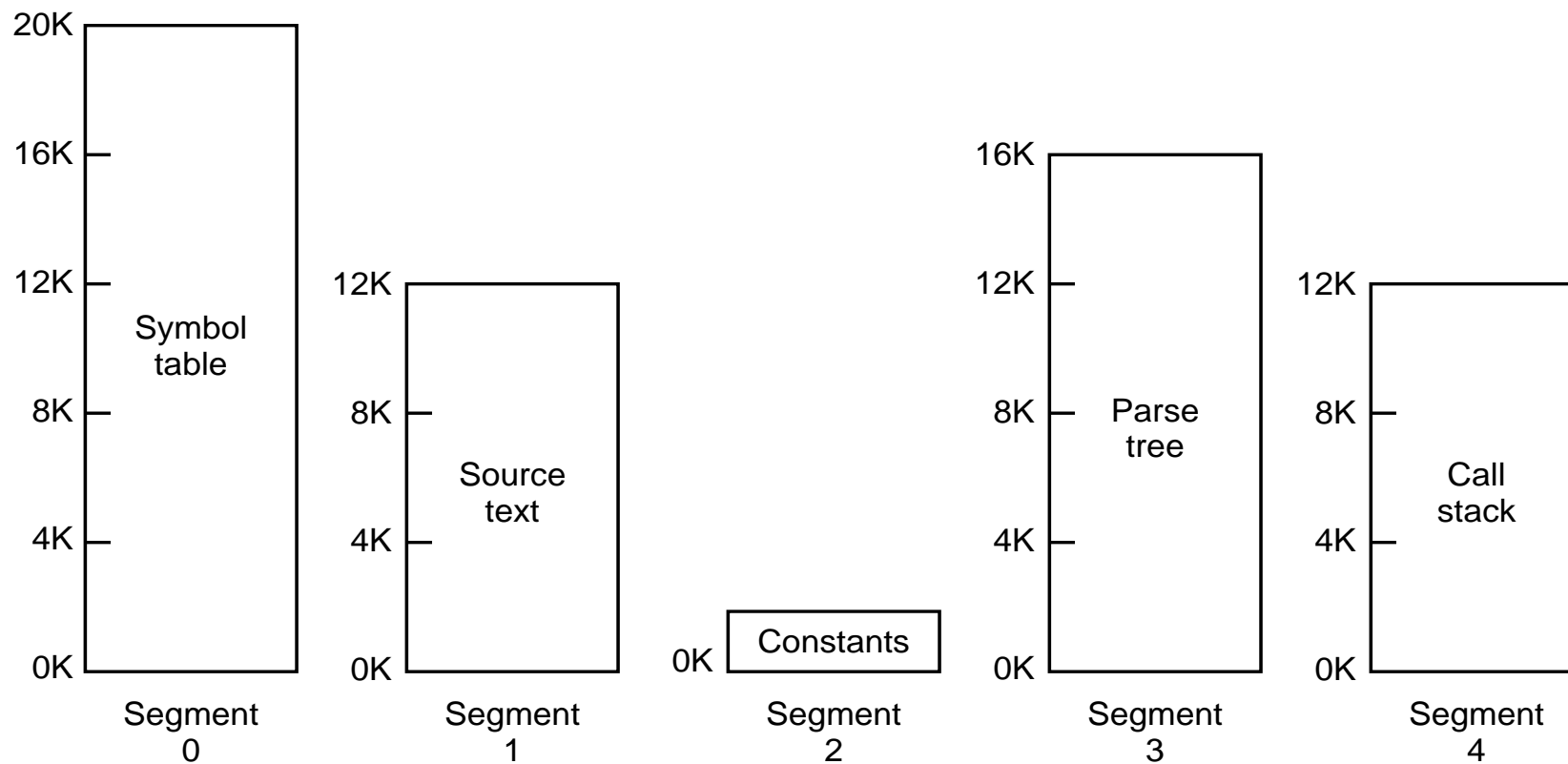
Segmentacja pamięci wirtualnej (1)

- jednowymiarowa, "płaska" przestrzeń adresowa
- możliwe nakładanie segmentów (!!!)



Segmentacja pamięci wirtualnej (2)

- rozłączne adresowanie segmentów

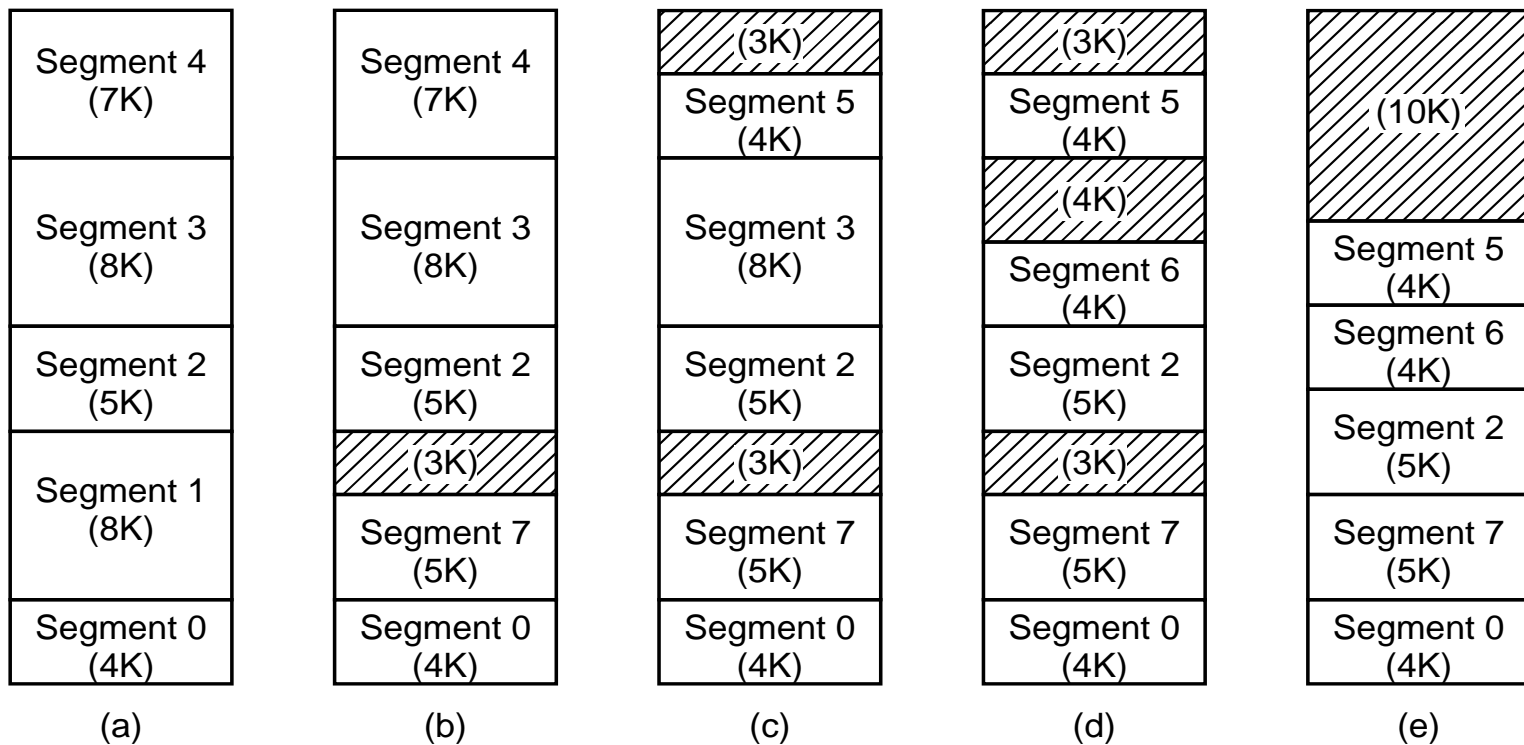


Segmentacja a stronicowanie

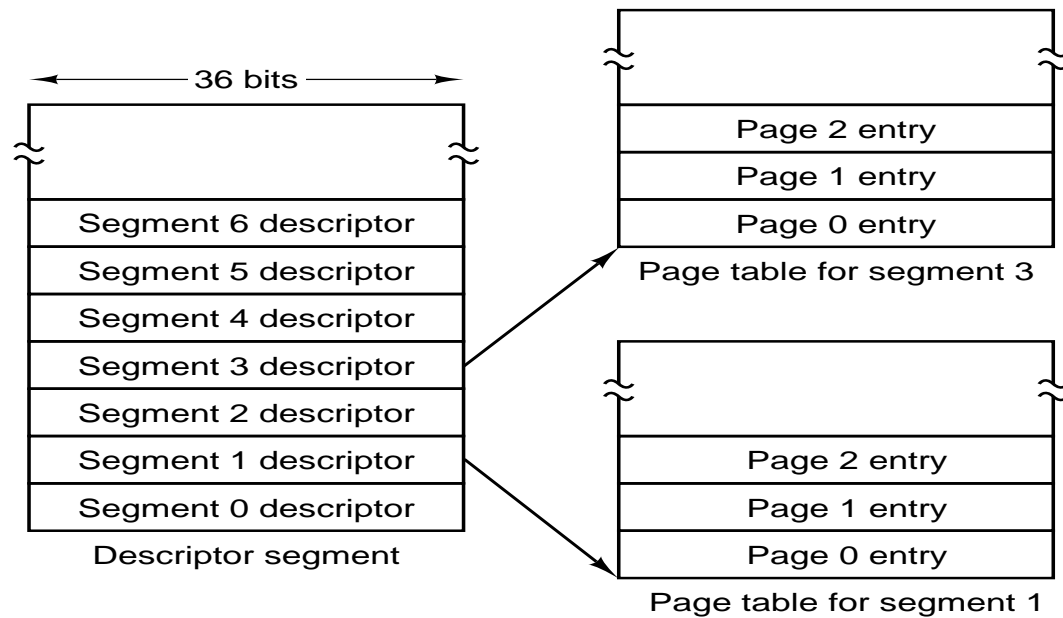
Consideration	Paging	Segmentation
Need the programmer be aware that this technique is being used?	No	Yes
How many linear address spaces are there?	1	Many
Can the total address space exceed the size of physical memory?	Yes	Yes
Can procedures and data be distinguished and separately protected?	No	Yes
Can tables whose size fluctuates be accommodated easily?	No	Yes
Is sharing of procedures between users facilitated?	No	Yes
Why was this technique invented?	To get a large linear address space without having to buy more physical memory	To allow programs and data to be broken up into logically independent address spaces and to aid sharing and protection

Segmentacja bez stronicowania

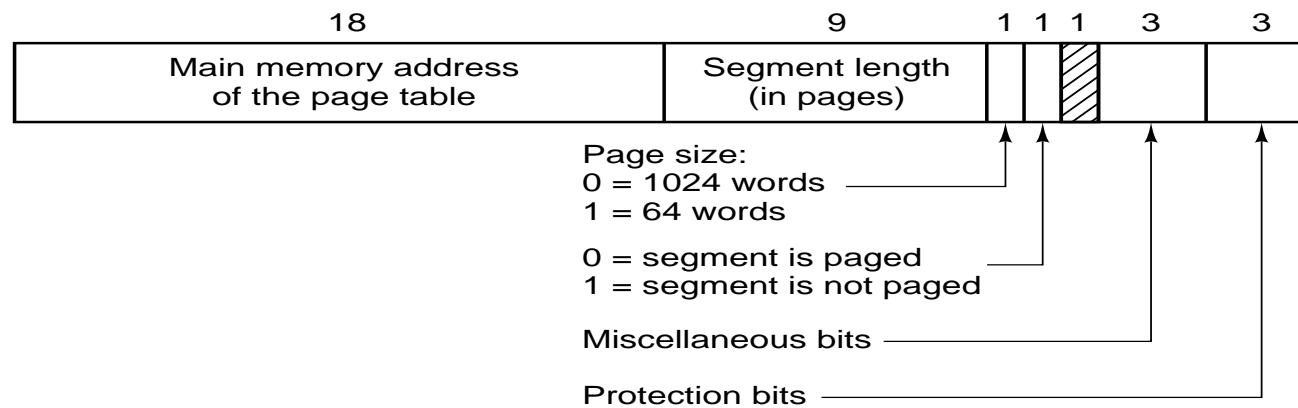
- główny problem: fragmentacja pamięci
 - (a)–(d) – typowa ewolucja fragmentacji
 - (e) – rozwiązanie: scalanie fragmentów (kompaktyfikacja)



Segmenty stronicowane: MULTIX (1)



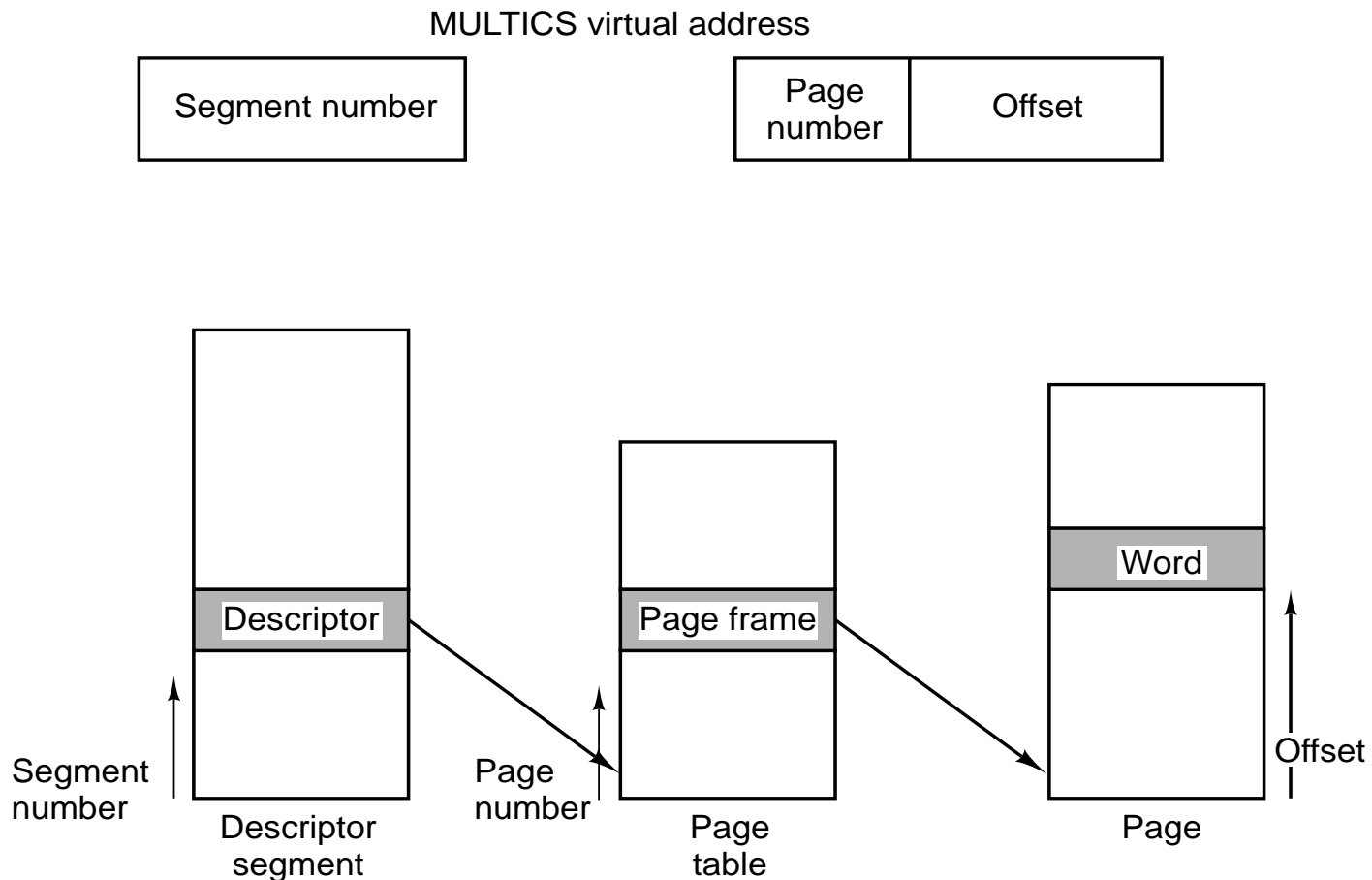
(a)



(b)

Segmenty stronicowane: MULTIX (2)

- konwersja dwuczęściowego adresu wirtualnego (34b)
 - Segment# (18b) : { Page# (6b) | Offset (10b) }



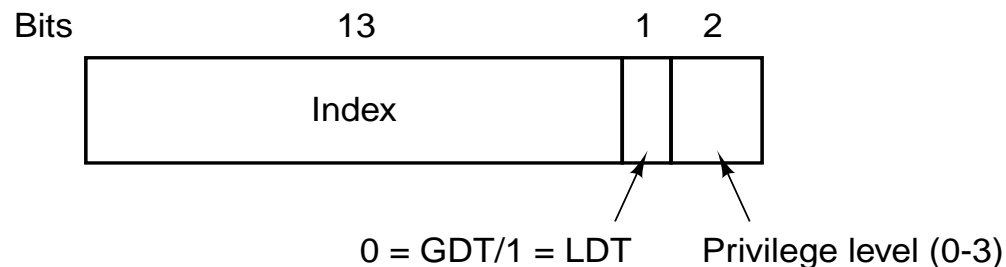
Segmenty stronicowane: MULTIX (3)

- uproszczona wersja TLB
 - nie uwzględniono wielkości stron (64/1024 W)

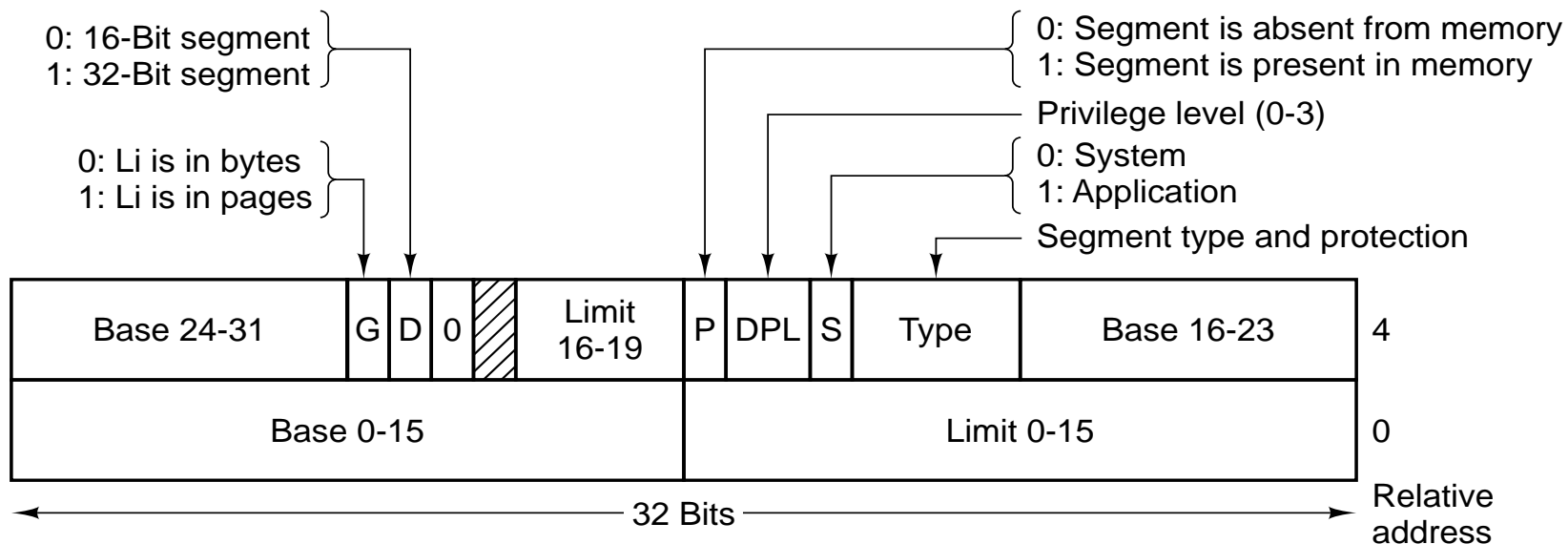
Comparison field		Page frame	Protection	Age	Is this entry used? ↓
Segment number	Virtual page				
4	1	7	Read/write	13	1
6	0	2	Read only	10	1
12	3	1	Read/write	2	1
					0
2	1	0	Execute only	7	1
2	2	12	Execute only	9	1

Segmenty stronicowane: IA32 (1)

□ selektor segmentu (Pentium)

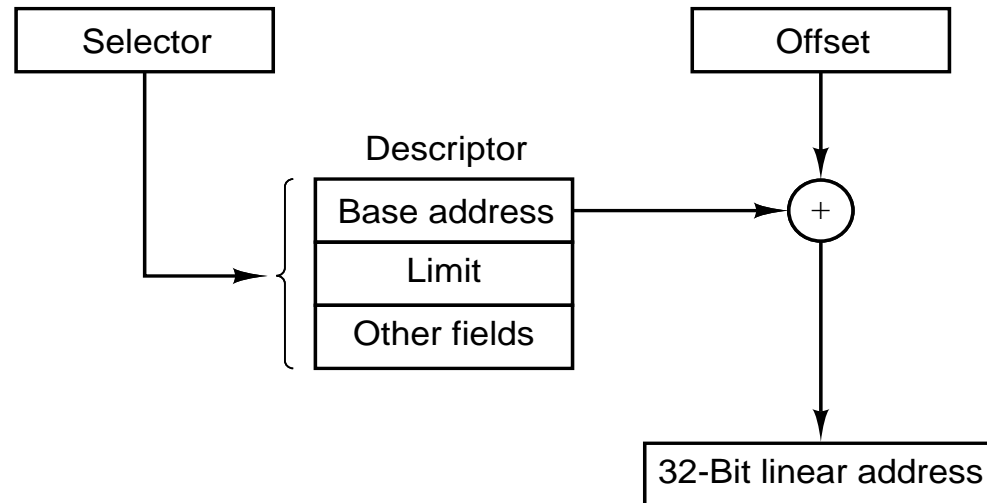


□ deskryptor segmentu kodu wykonywalnego



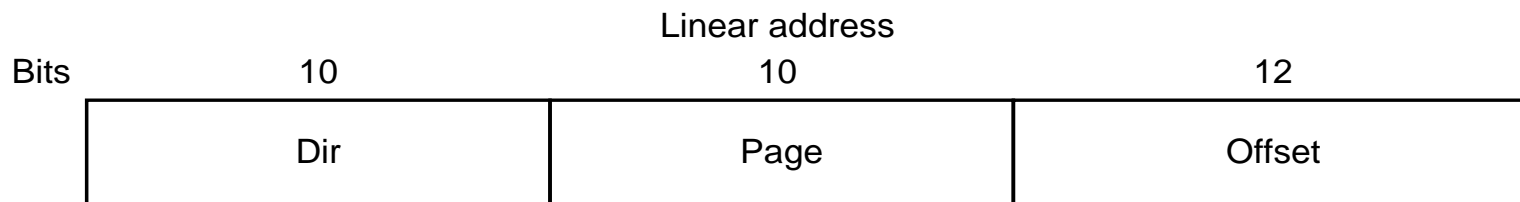
Segmenty stronicowane: IA32 (2)

- konwersja (selektor:offset) do adresu liniowego

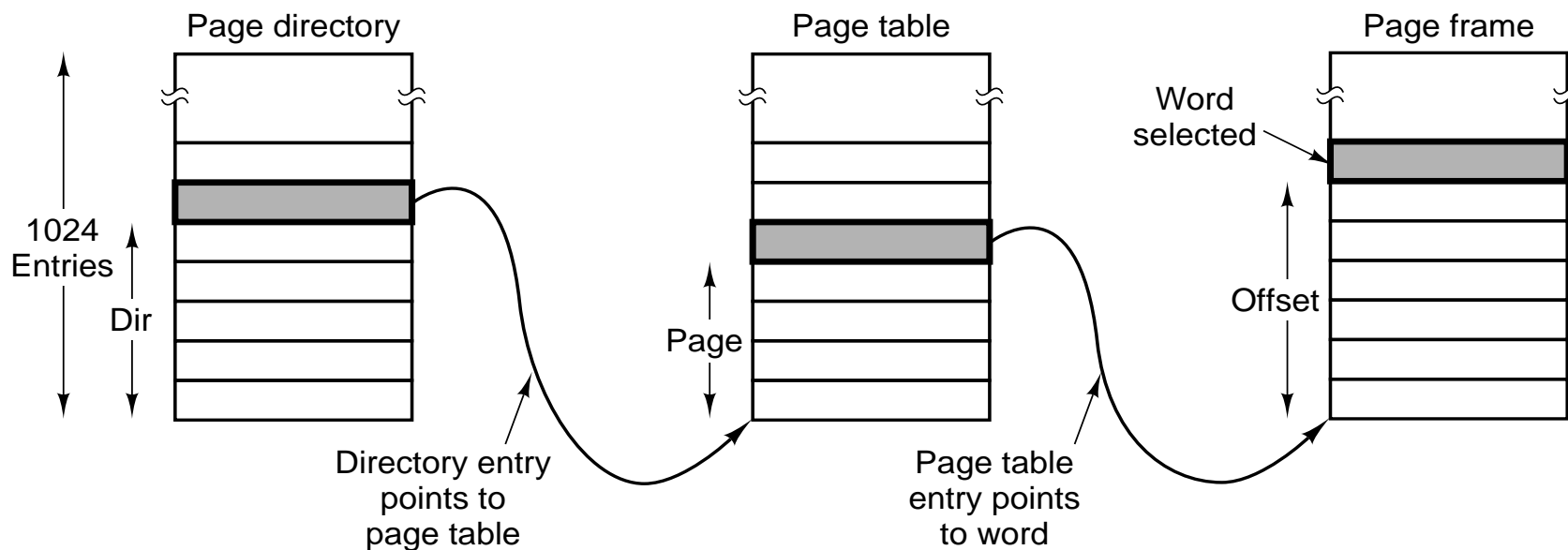


Segmenty stronicowane: IA32 (3)

- konwersja adresu liniowego (a) na adres fizyczny (b)



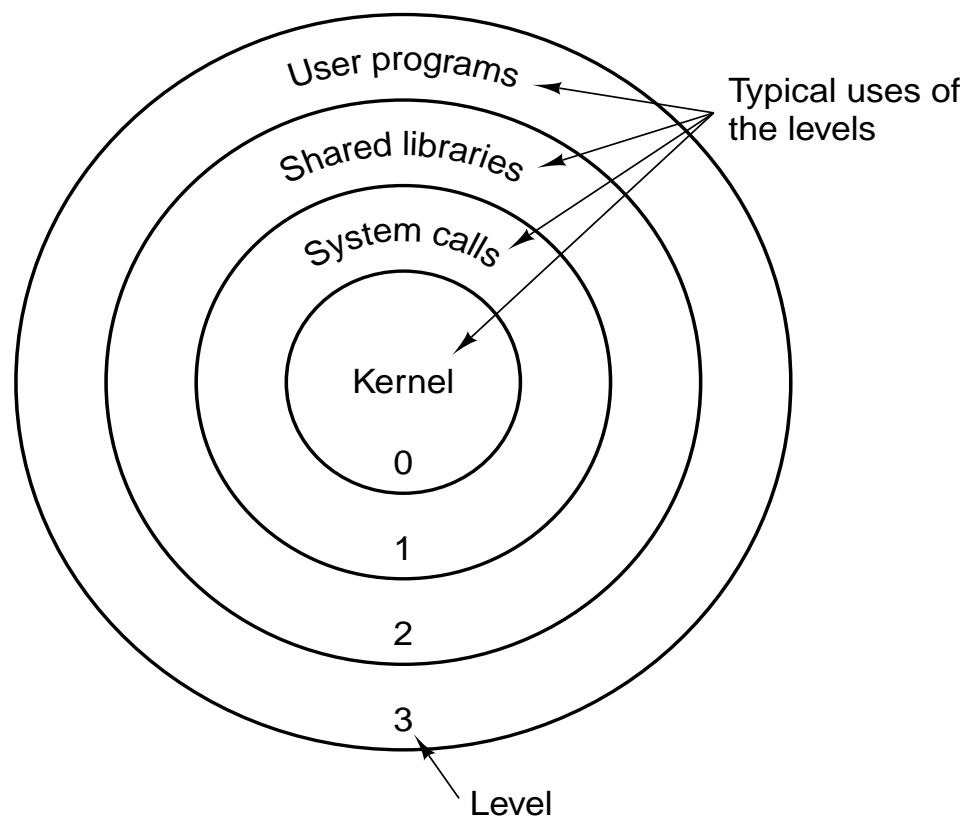
(a)



(b)

Segmenty stronicowane: IA32 (4)

- poziomy ("pierścienie") ochrony



===

05 – Wejście/Wyjście

- ☐ Podstawy – aspekty sprzętowe i programowe
- ☐ Warstwowa struktura obsługi urządzeń wejścia/wyjścia
- ☐ Urządzenia blokowo–zorientowane (dyski, etc.)
- ☐ Urządzenia znakowo–zorientowane (terminale, etc.)
- ☐ Graficzne interfejsy użytkownika
- ☐ Sieci komunikacyjne

>>>

Typowe prędkości transmisji danych

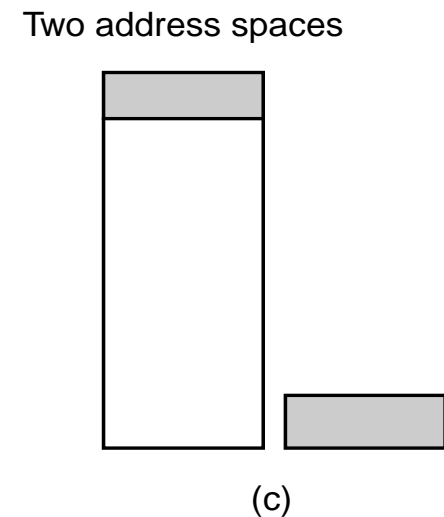
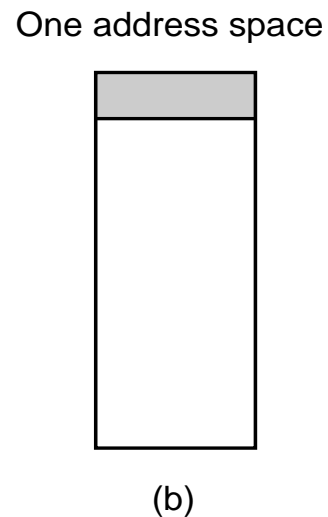
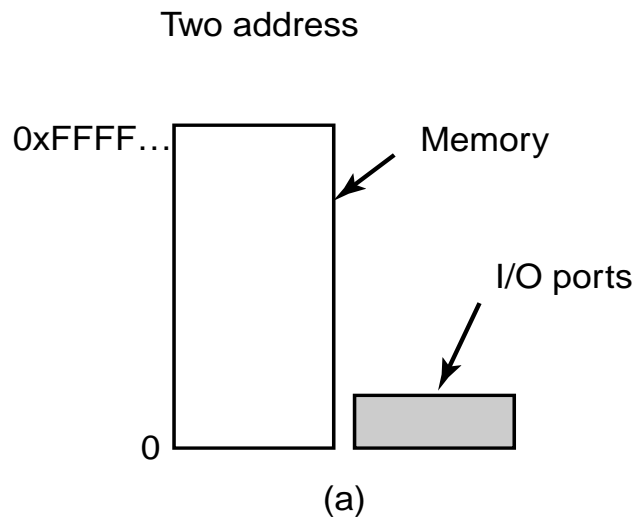
Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Telephone channel	8 KB/sec
Dual ISDN lines	16 KB/sec
Laser printer	100 KB/sec
Scanner	400 KB/sec
Classic Ethernet	1.25 MB/sec
USB (Universal Serial Bus)	1.5 MB/sec
Digital camcorder	4 MB/sec
IDE disk	5 MB/sec
40x CD-ROM	6 MB/sec
Fast Ethernet	12.5 MB/sec
ISA bus	16.7 MB/sec
EIDE (ATA-2) disk	16.7 MB/sec
FireWire (IEEE 1394)	50 MB/sec
XGA Monitor	60 MB/sec
SONET OC-12 network	78 MB/sec
SCSI Ultra 2 disk	80 MB/sec
Gigabit Ethernet	125 MB/sec
Ultrium tape	320 MB/sec
PCI bus	528 MB/sec
Sun Gigaplane XB backplane	20 GB/sec

Sprzęt wejścia/wyjścia

- typowy sprzęt/urządzenie We/Wy
 - elementy mechaniczne (!)
 - układy elektroniczne
 - firmware
- typowy sterownik urządzeń (ang.: device controller)
 - układy elektroniczne, w tym:
 - interfejs(y) obsługiwanych urządzeń
 - interfejs systemu komputerowego
 - układy wykrywania i korekcji błędów
 - układy buforowania i konwersji danych
 - firmware
 - opcjonalnie: odpowiedni koprocesor We/Wy, z pamięcią, etc.

Adresowanie urządzeń We/Wy (1)

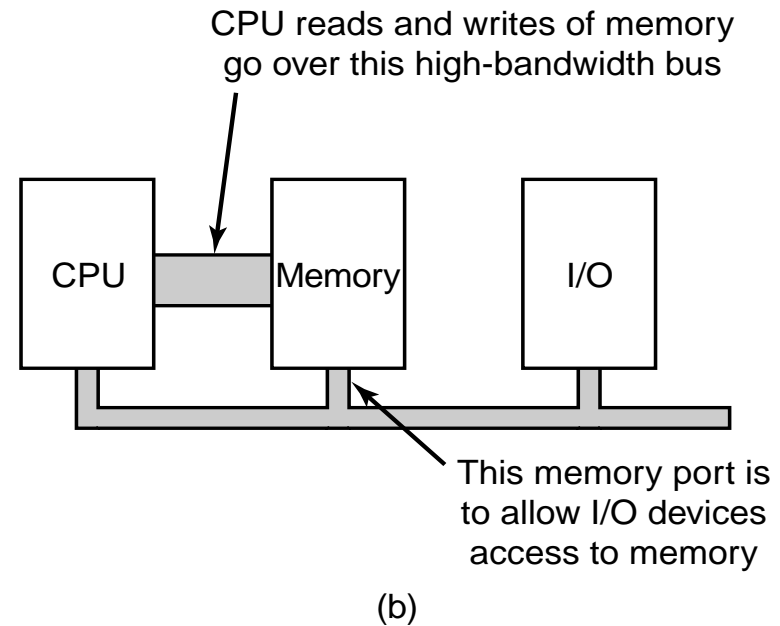
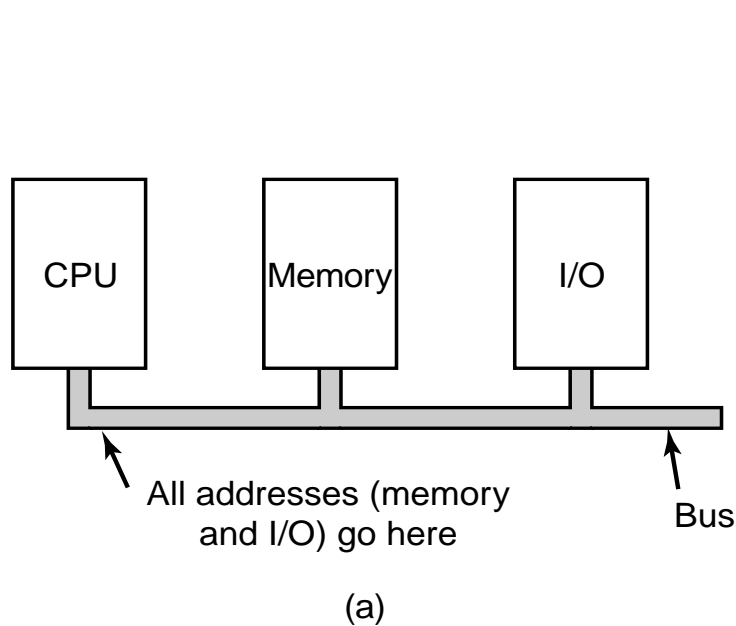
- odwzorowanie urządzeń I/O w systemie komputerowym
 - "memory-mapped I/O"
 - (a) – odrębna przestrzeń adresowa I/O
 - (b) – jednolite adresowanie pamięci i urządzeń I/O
 - (c) – model hybrydowy



Adresowanie urządzeń We/Wy (2)

□ architektura systemowa

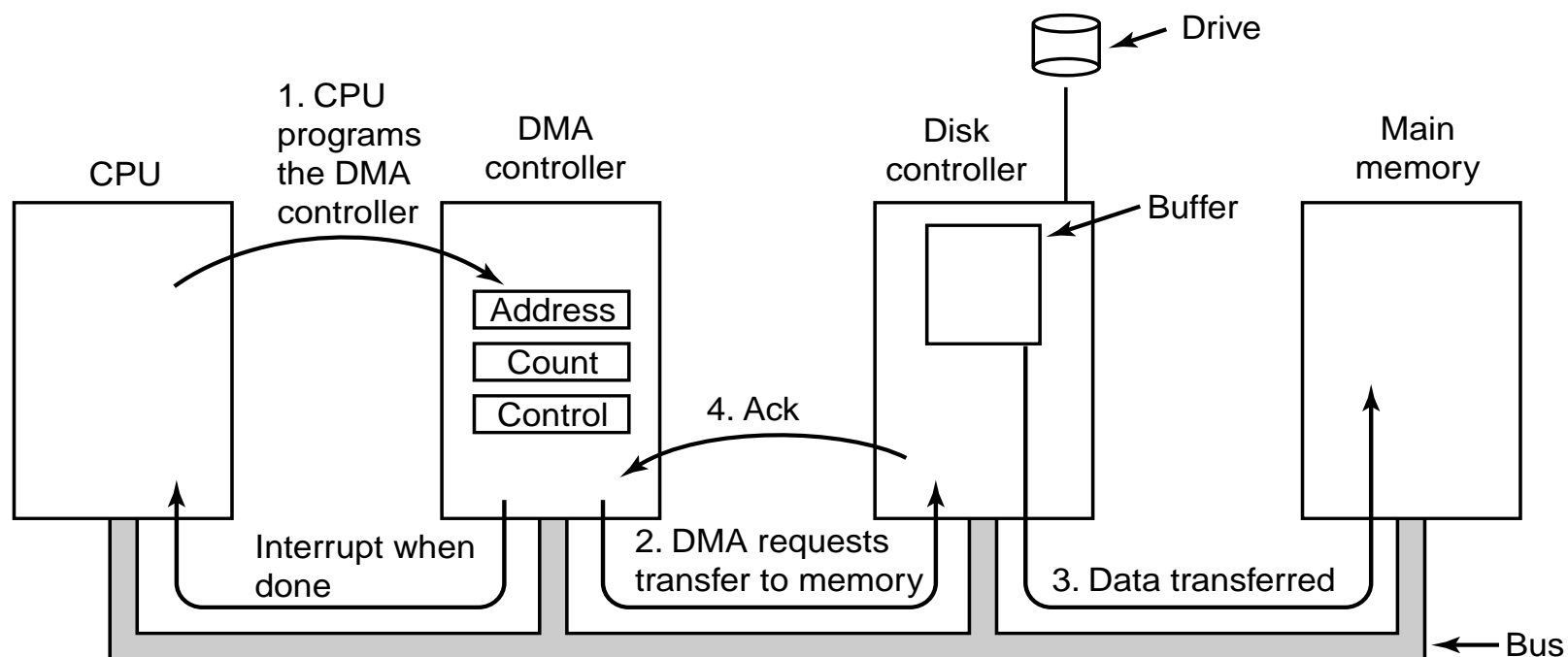
- (a) – wspólna (uniwersalna) magistrala (np. OmniBus/PDP-8)
- (b) – oddzielna magistrala dla obsługi I/O
- inna, najczęściej hybrydowa z przełącznicą systemową



Bezpośredni dostęp do pamięci

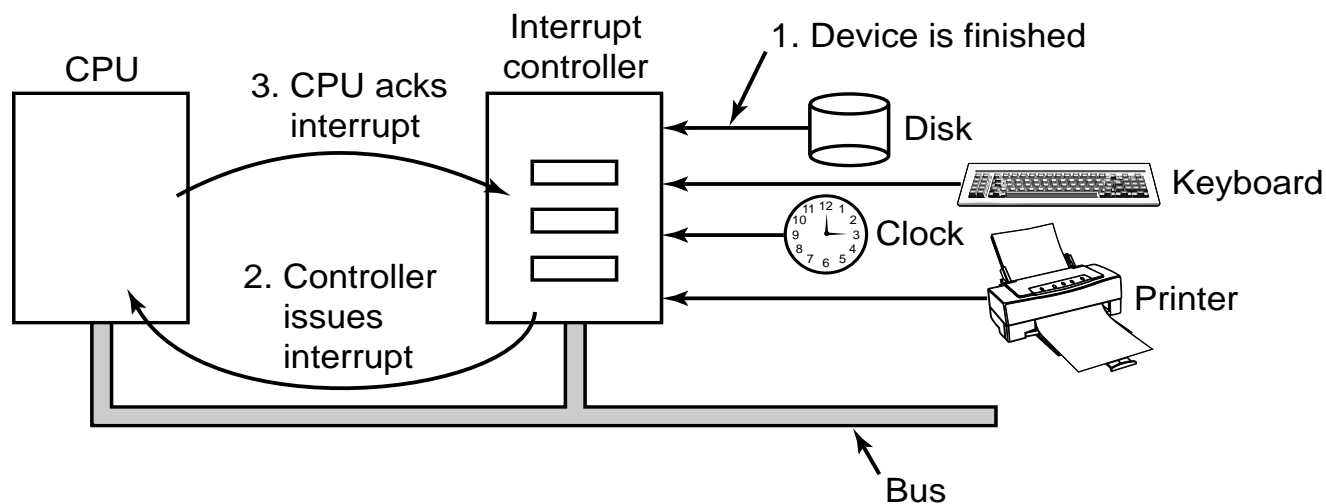
□ DMA – "Direct Memory Access"

- wymaga odpowiedniego sprzętu – sterownika DMA
- wymaga współpracy ze sterownikiem przerwań



Urządzenia We/Wy a przerwania

- sygnalizacja stanu urządzenia
 - urządzenia wykorzystują magistralę systemową do komunikacji ze sterownikiem przerwań

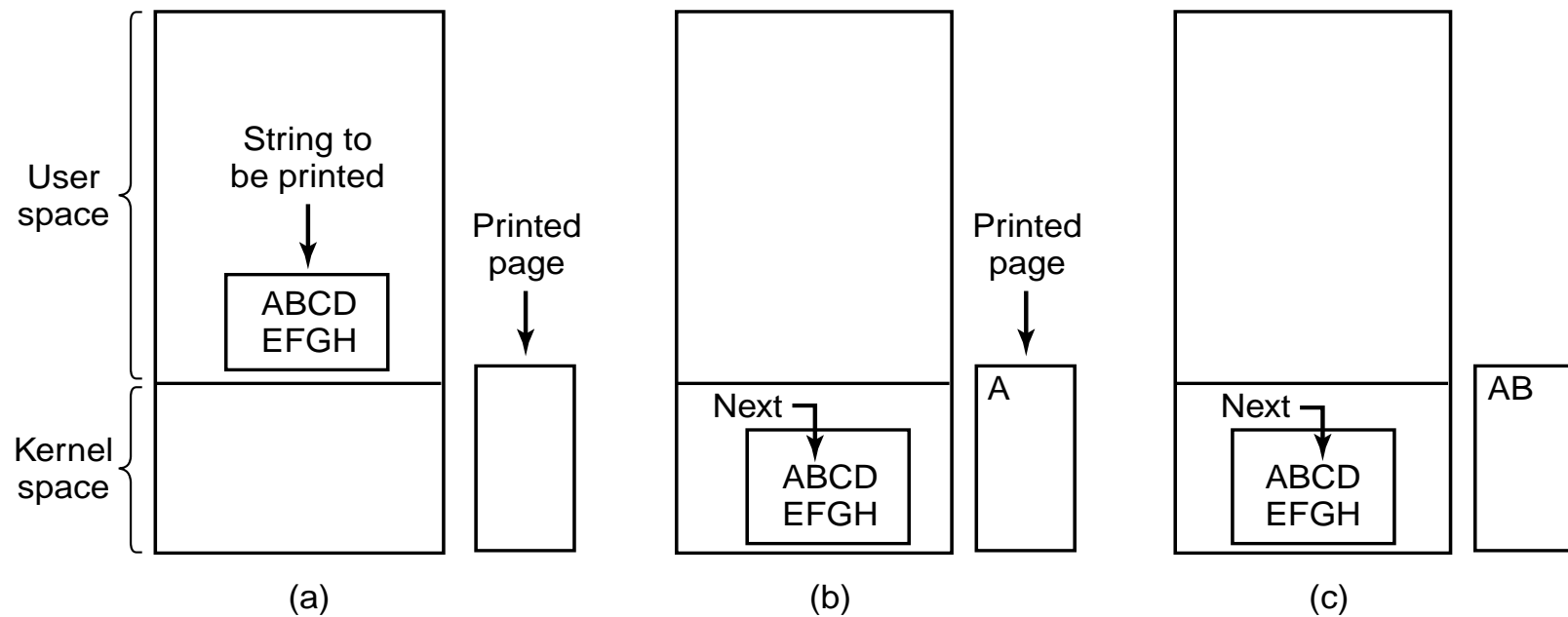


Oprogramowanie systemowe We/Wy

- niezależny dostęp do poszczególnych urządzeń
- jednolita identyfikacja/nazewnictwo w systemie
- możliwość współdzielenia/dedykowania urządzeń
- dobór właściwego trybu pracy
 - ogólny: synchroniczny/asynchroniczny
 - programowane We/Wy (PIO, "Programmed I/O")
 - We/Wy sterowane przerwaniem (IDIO, "Interrupt-Driven I/O")
 - We/Wy z bezpośrednim dostępem do pamięci (DMA)
- buforowanie danych
- wykrywanie i obsługa błędów I/O (niskopoziomowa)

Programowane We/Wy (1)

- przykład: drukowanie znak po znaku
 - (a) – inicjowanie wydruku, tryb użytkownika
 - (b),(c) – sterowanie wydrukiem, tryb jądra



Programowane We/Wy (2)

- przykładowa procedura wydruku (PIO)
 - fragment kodu procedury obsługi wywołania systemowego,
 - praca na poziomie jądra systemu (!)

```
copy_from_user(buffer, p, count);  
for (i = 0; i < count; i++) {  
    while (*printer_status_reg != READY) ;  
    *printer_data_register = p[i];  
}  
return_to_user();
```

/* p is the kernel bufer */
/* loop on every character */
/* loop until ready */
/* output one character */

We/Wy sterowane przerwaniem

- wywołanie procedury drukującej (IDIO)
 - (a) – systemowa obsługa wywołania
 - (b) – procedura obsługi przerwania

```
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)

We/Wy w trybie DMA

- wywołanie procedury drukującej (DMA)
 - (a) – systemowa obsługa wywołania
 - (b) – procedura obsługi przerwania

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

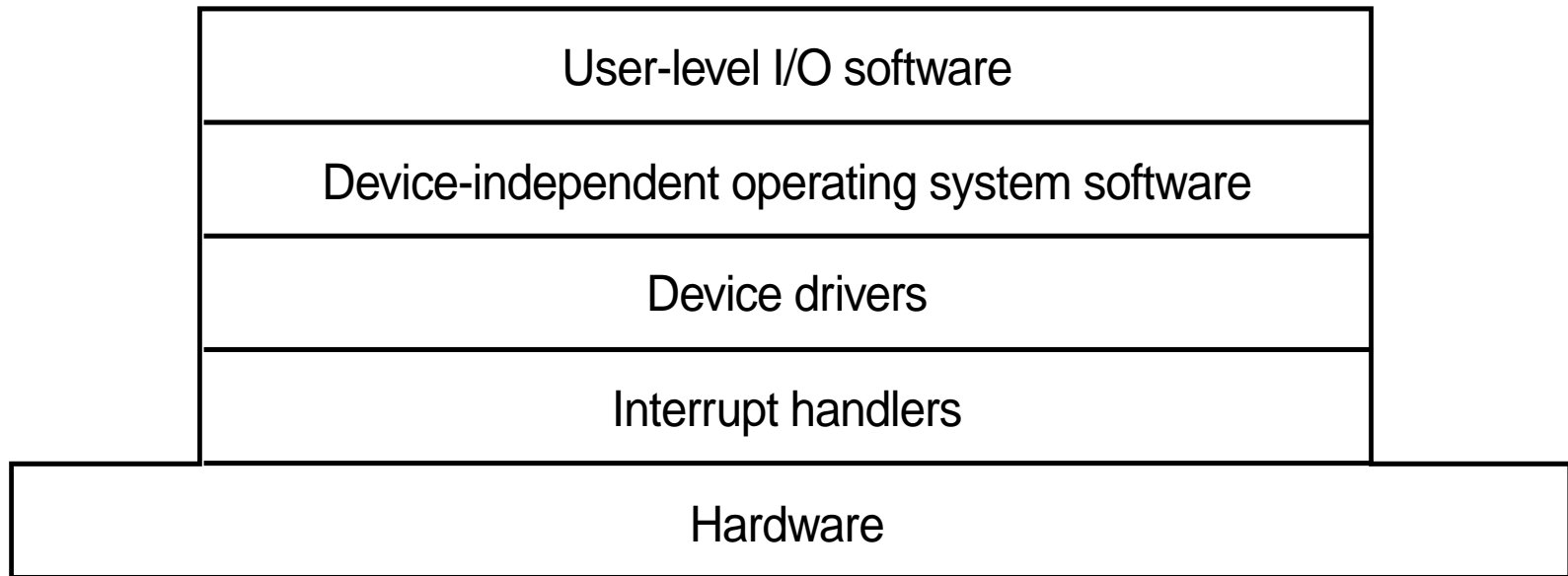
(a)

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

(b)

Warstwowa struktura oprogramowania

- typowy system wykorzystujący przerwania



Sterowniki urządzeń i obsługa przerwań

□ ścisła współpraca:

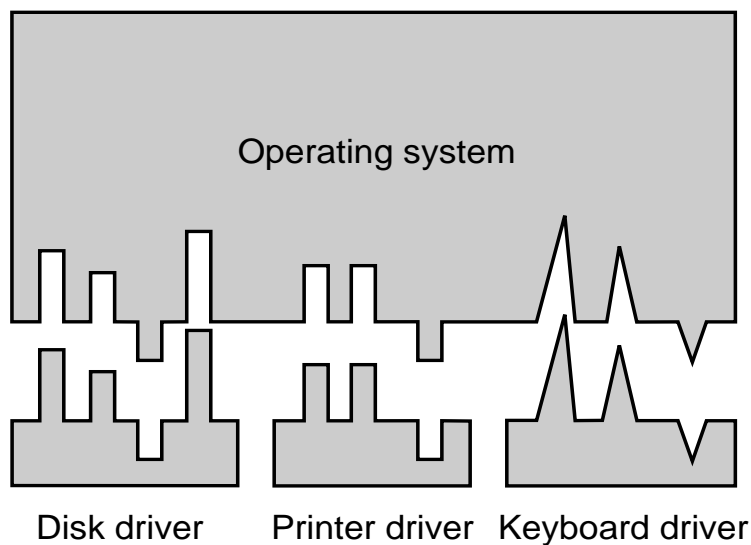
- sterowniki urządzeń inicjują operację We/Wy
- zdarzenia (gotowość, koniec, błąd, etc.) generują odpowiednie sygnały przerwań na magistrali lub przełącznicy systemowej
- procesor przekazuje sterowanie odpowiedniej procedurze obsługi przerwania
- sterownik urządzenia oczekuje na zakończenie pracy procedury obsługi przerwania
- procedura obsługi przerwania informuje sterownik urządzenia o zakończeniu swojej pracy i umożliwia mu kontynuację wykonania (programowe odtworzenie zawartości rejestrów, etc., jeśli nie jest to wykonywane sprzętowo)

□ warstwowa (sub)struktura:

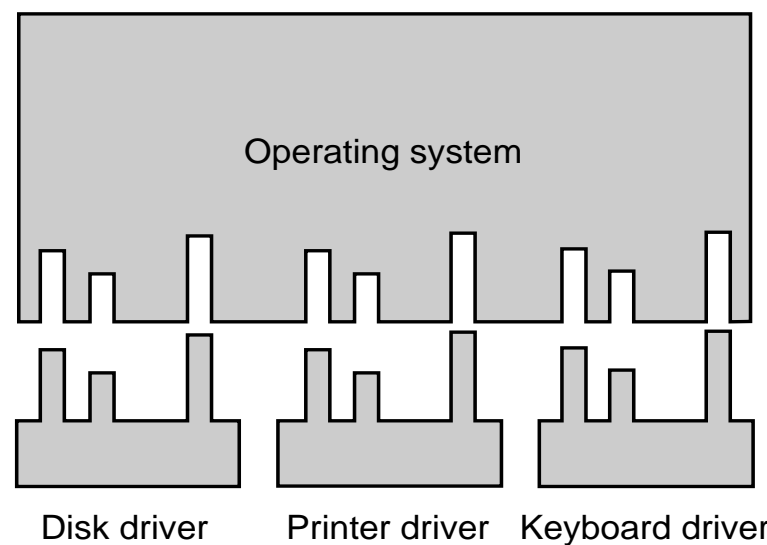
- obsługa interfejsu systemowego, niezależna od sprzętu
- obsługa magistrali/przełącznicy (np. PCI)
- obsługa grupy urządzeń (np. SCSI)
- obsługa konkretnego urządzenia (np. HDD)

Systemowy interfejs We/Wy

- możliwie niezależny od sprzętu
 - jednorodna obsługa wszystkich (?) sterowników urządzeń
 - obsługa urządzeń dedykowanych
 - buforowanie
 - wspólna, jednolita wielkość bloku
 - sygnalizacja i obsługa błędów



(a)

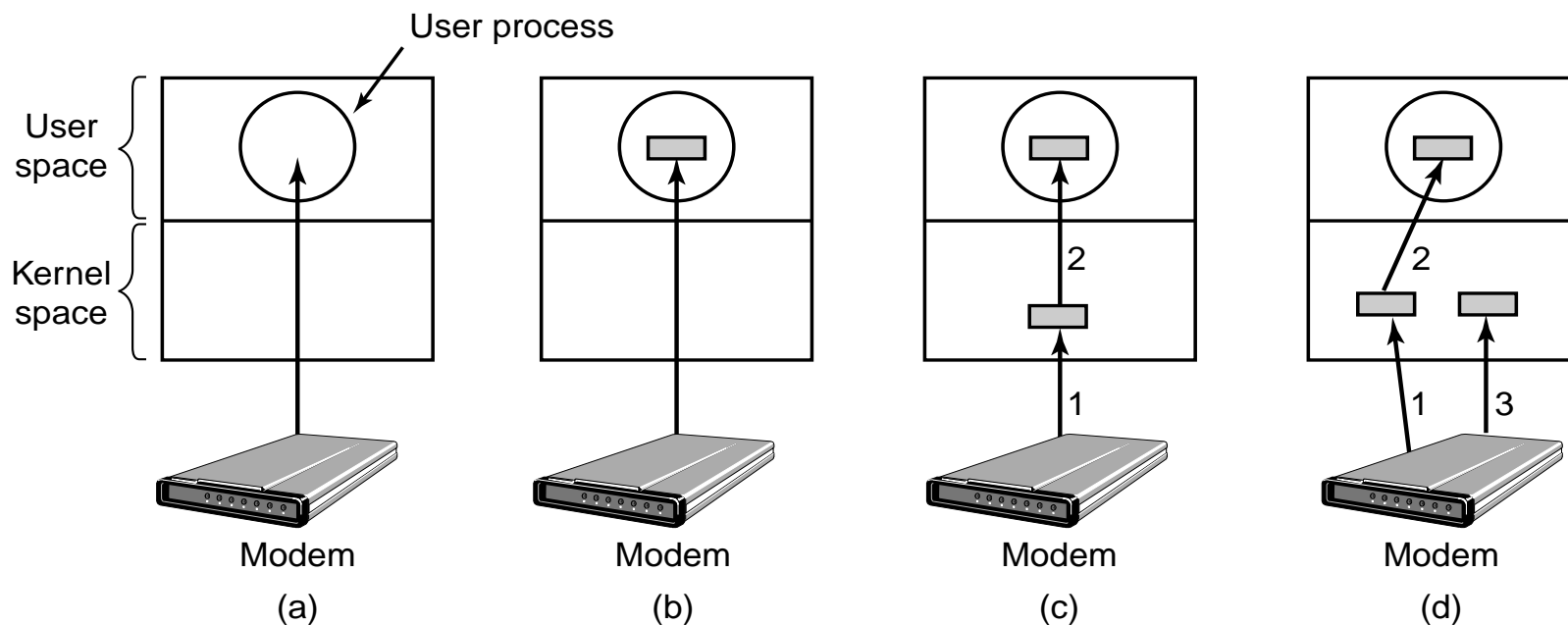


(b)

Buforowanie i replikacja (1)

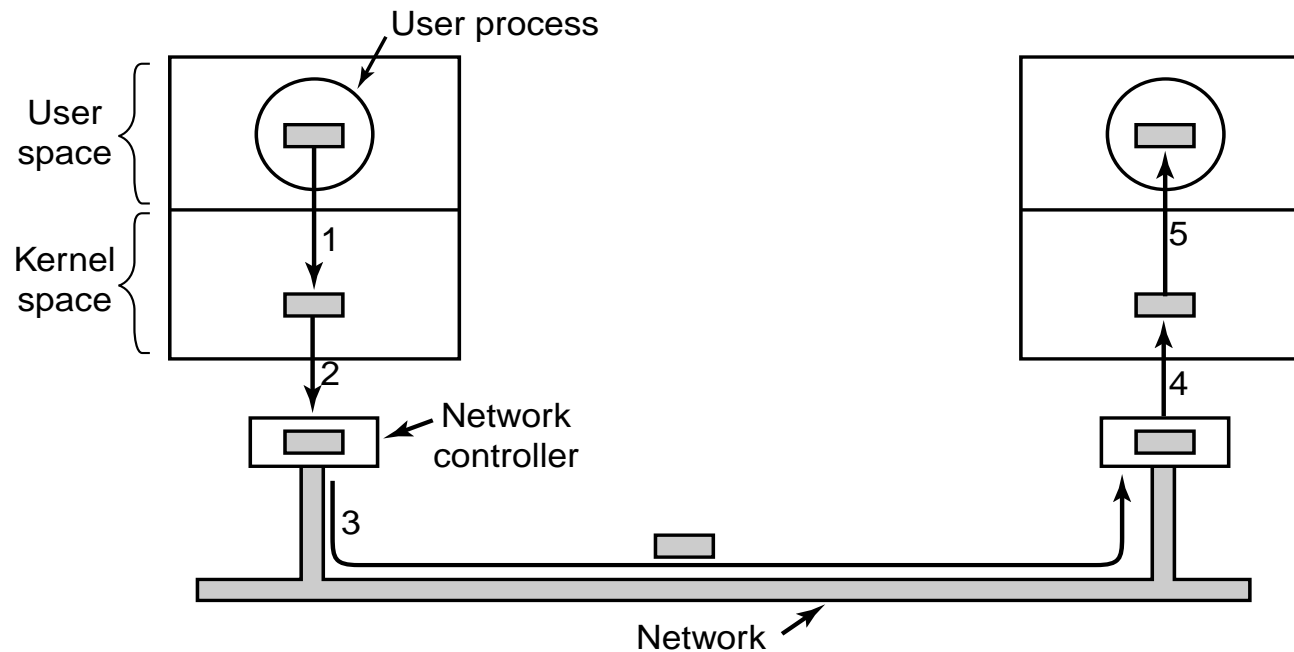
□ buforowanie We/Wy

- (a) – brak
- (b) – w przestrzeni użytkownika
- (c) – w przestrzeni jądra z replikacją w przestrzeni użytkownika
- (d) – buforowanie podwójne w przestrzeni jądra, z replikacją i bez



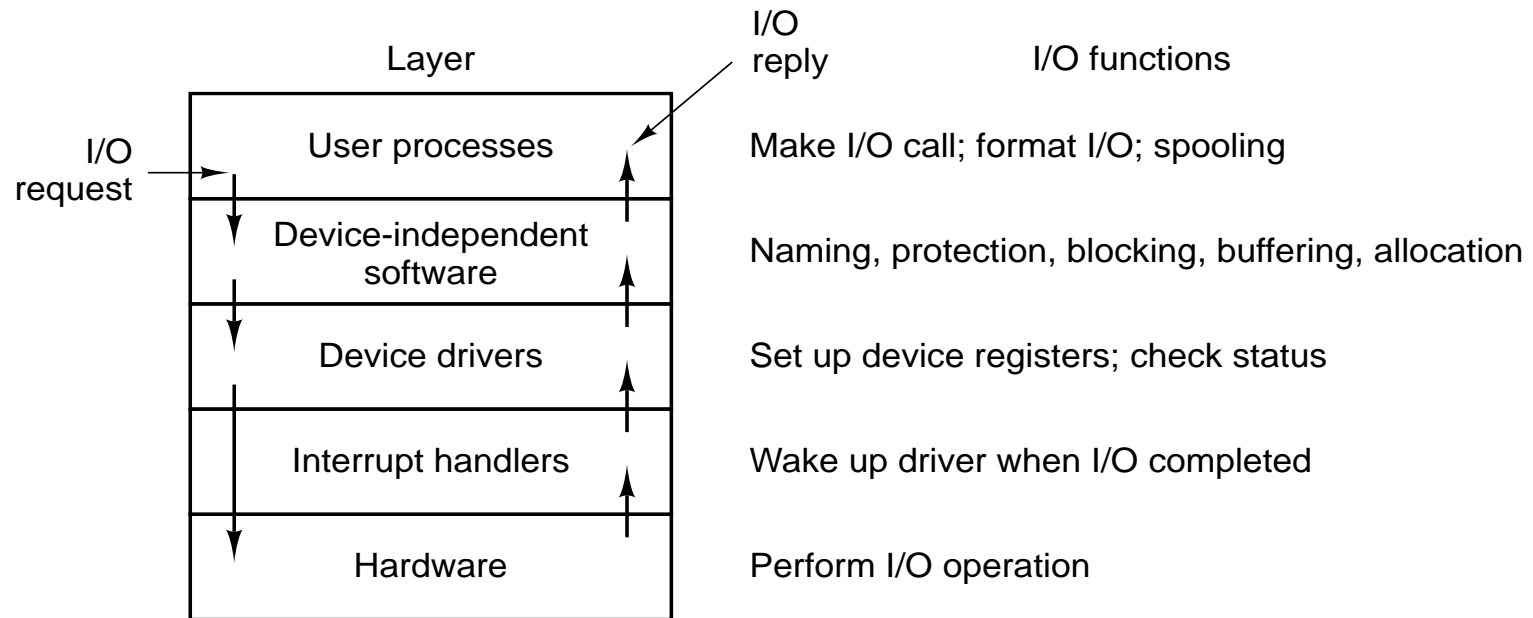
Buforowanie i replikacja (2)

- wielokrotna replikacja i buforowanie
 - typowa sytuacja podczas pracy w sieci komputerowej



We/Wy w przestrzeni użytkownika

- obsługa operacji We/Wy



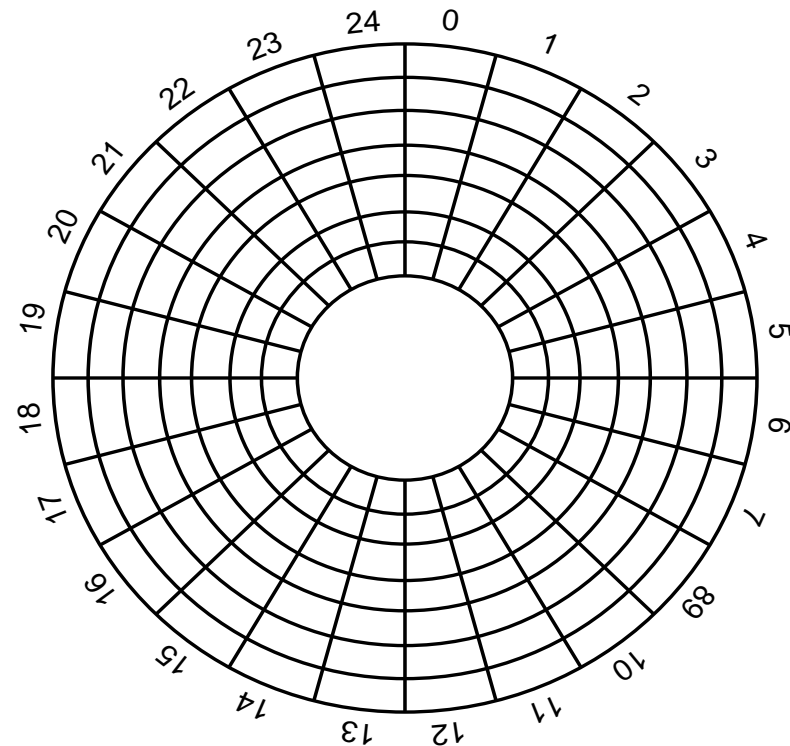
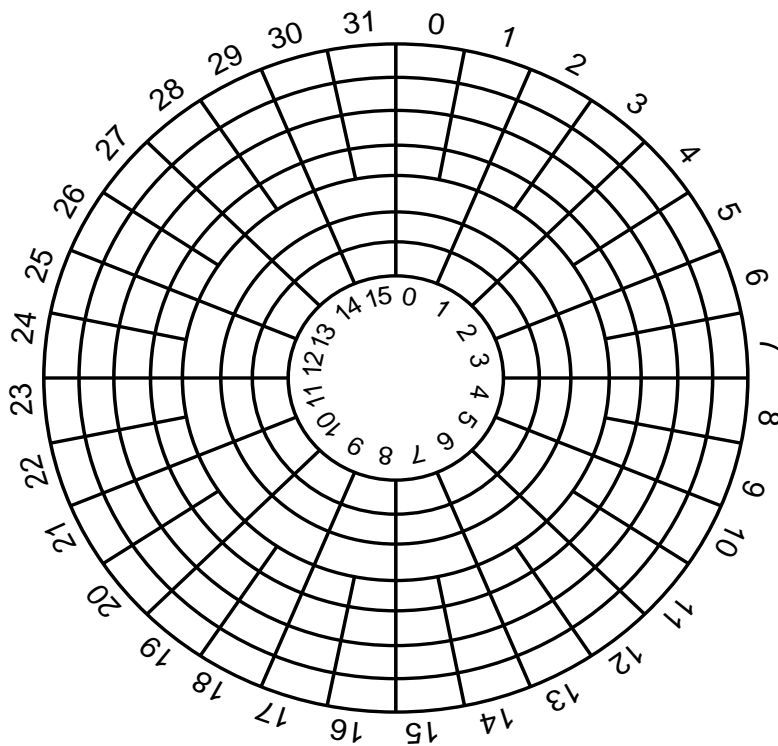
Urządzenia dyskowe (1)

- przykładowe parametry fizyczne

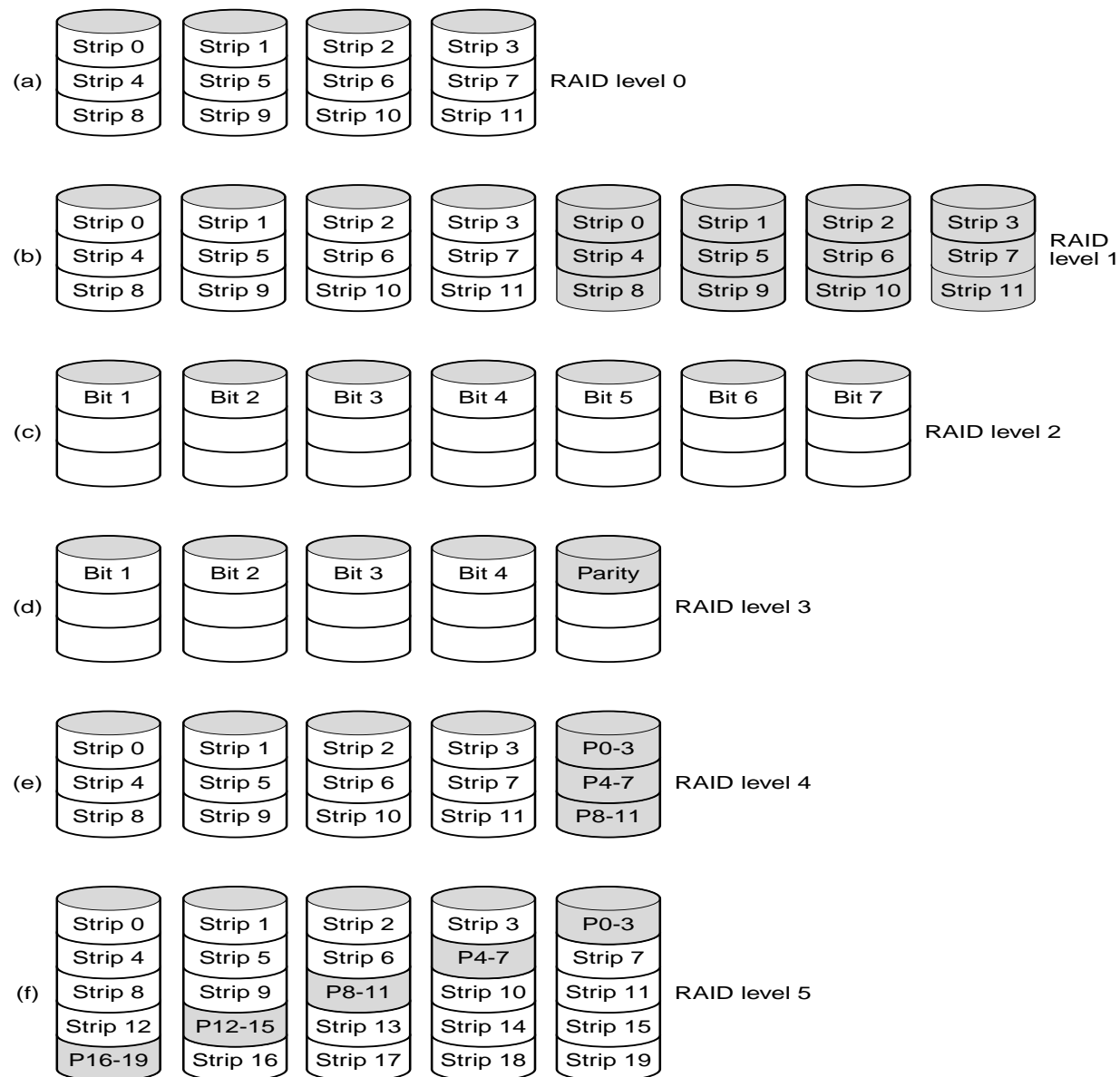
Parameter	IBM 360-KB floppy disk	WD 18300 hard disk
Number of cylinders	40	10601
Tracks per cylinder	2	12
Sectors per track	9	281 (avg)
Sectors per disk	720	35742000
Bytes per sector	512	512
Disk capacity	360 KB	18.3 GB
Seek time (adjacent cylinders)	6 msec	0.8 msec
Seek time (average case)	77 msec	6.9 msec
Rotation time	200 msec	8.33 msec
Motor stop/start time	250 msec	20 sec
Time to transfer 1 sector	22 msec	17 μ sec

Urządzenia dyskowe (2)

- "geometria dysku" a zapis fizyczny
 - zapis strefowy – Zone Bit Recording (ZBR), 2 strefy o różnej gęstości
 - wirtualna "geometria dysku" – przykład translacji zapisu fizycznego

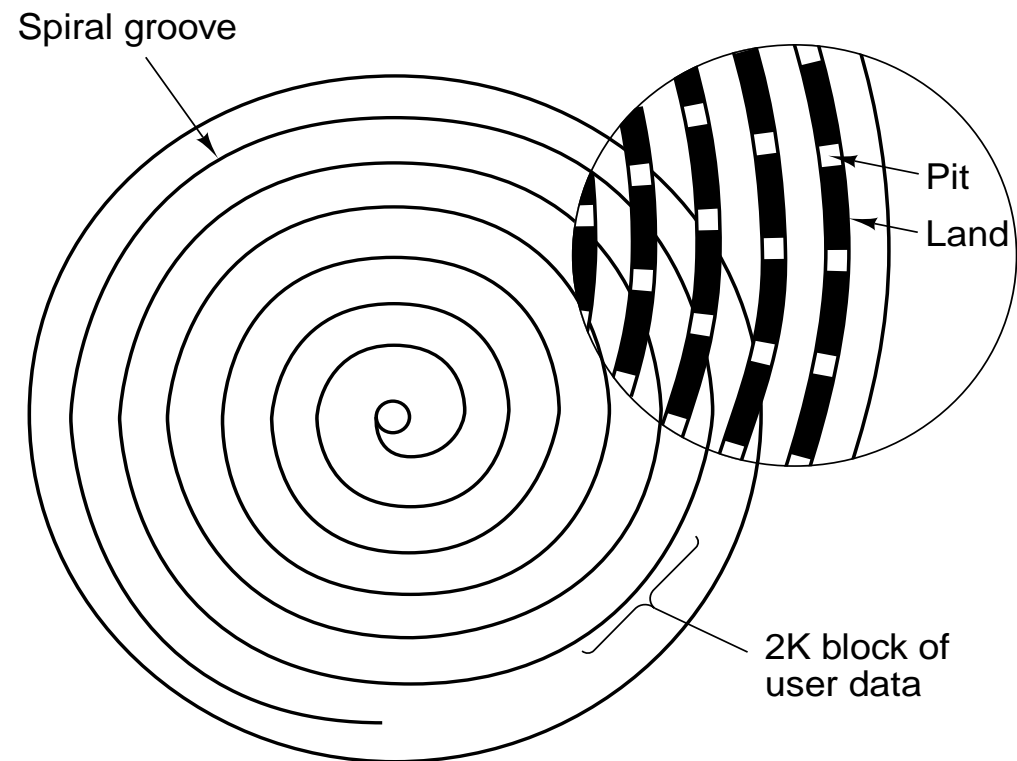


Macierze dyskowe (RAID)



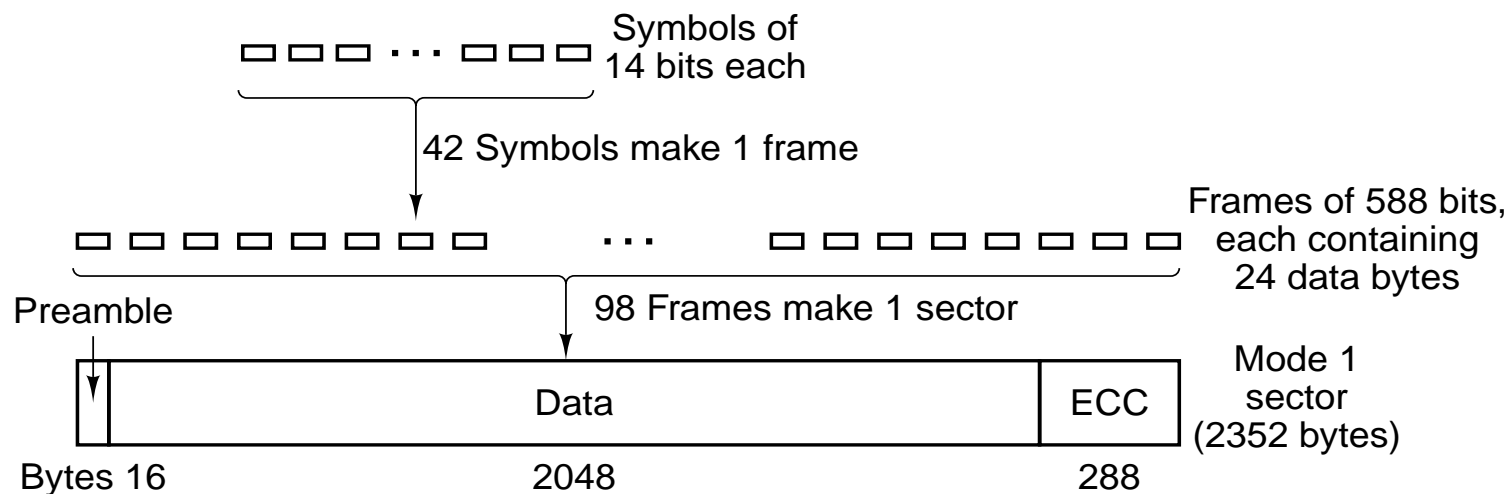
Dyski kompaktowe (1)

- zapis spiralny



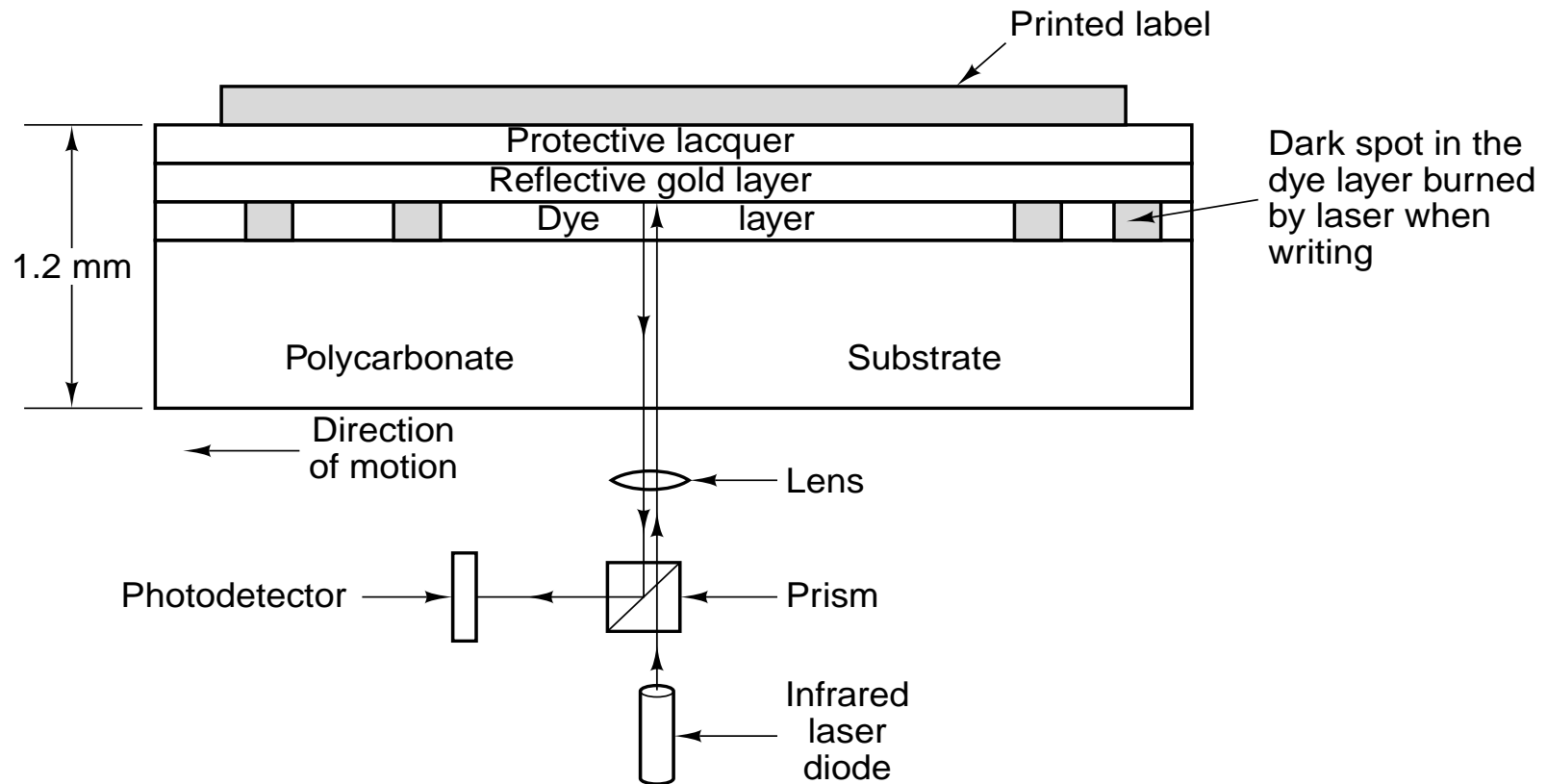
Dyski kompaktowe (2)

□ logiczne rozmieszczenie danych (CD-ROM)



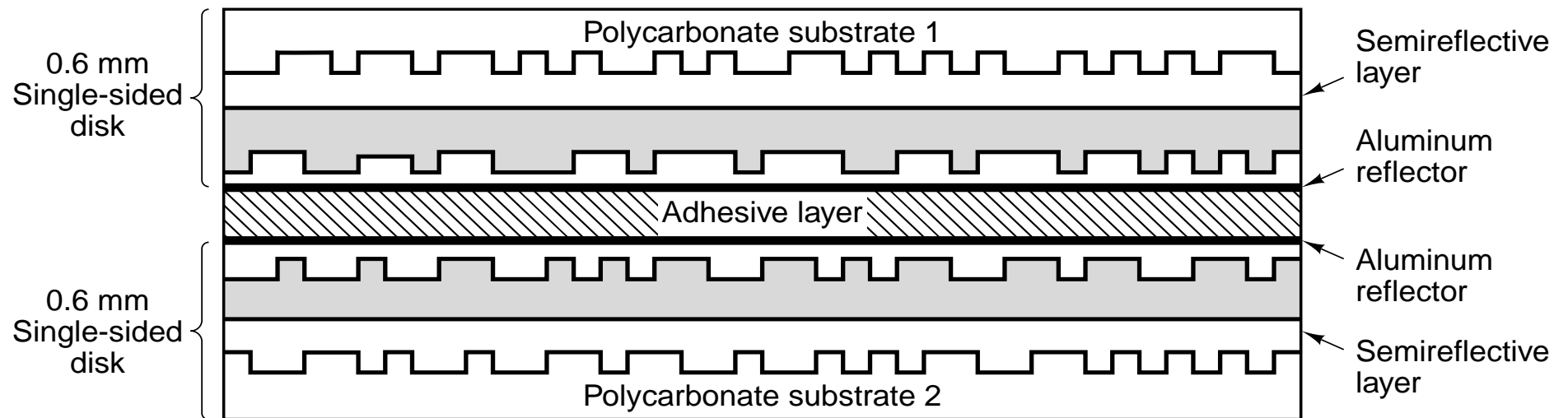
Dyski kompaktowe (3)

- schemat struktury fizycznej nośnika i napędu (CD-ROM)



Dyski kompaktowe (4)

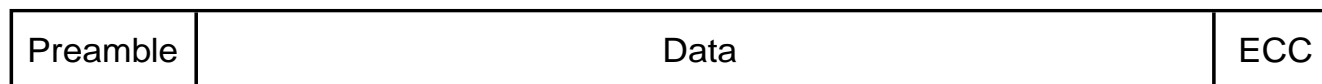
- dwustronny dysk DVD



Obsługa urządzeń dyskowych

- **wczesne systemy operacyjne**
 - bezpośredni zapis danych na dysku, wg. odpowiedniej "geometrii", i/lub
 - wg. zainicjowanej struktury rekordowej/blokowej (CKD, FB, etc.)
 - konieczność obsługi defektów nośnika

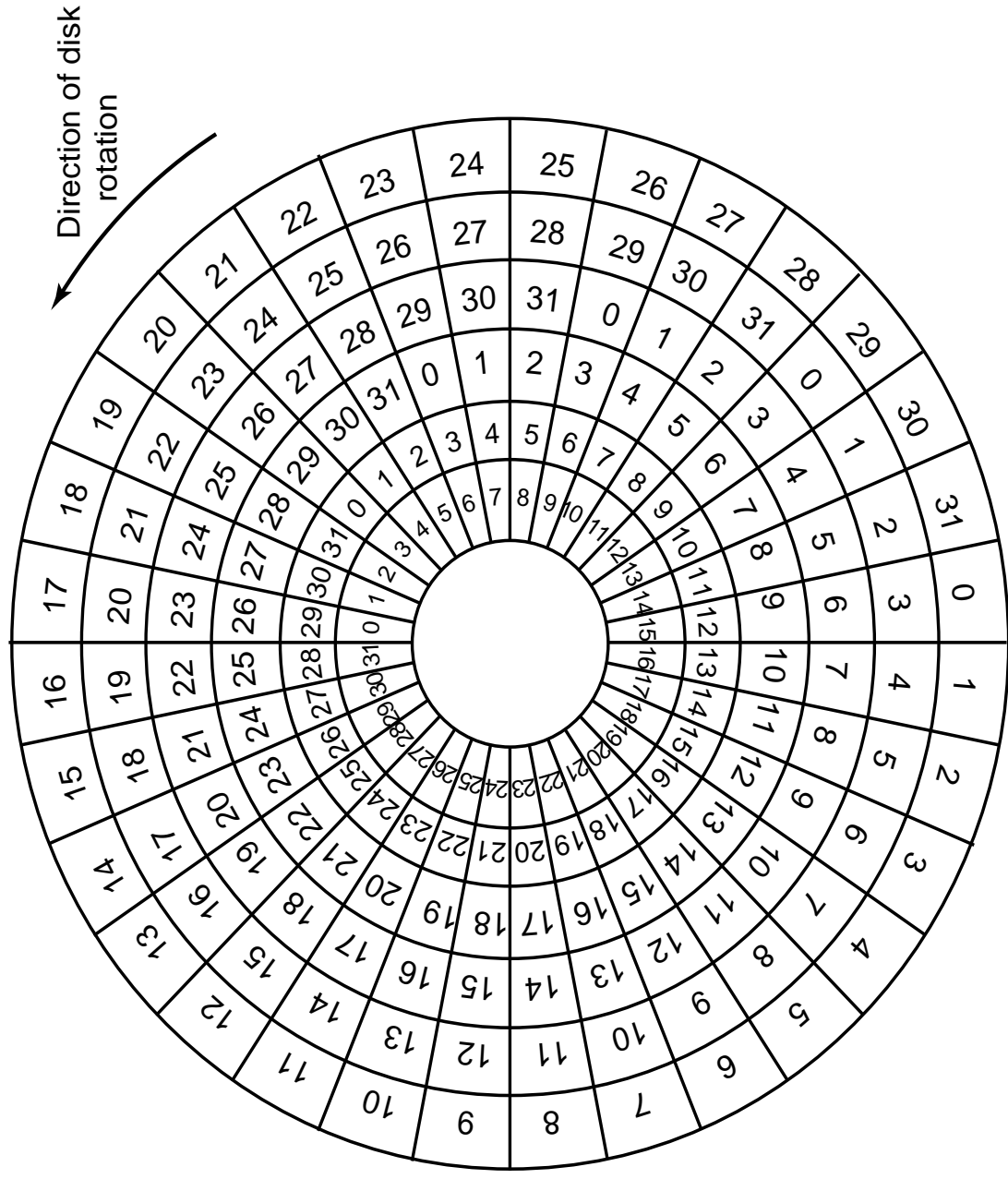
- **współcześnie:**
 - urządzenia dyskowe posiadają fabryczną, standardową strukturę sektorową



- defekty nośnika koryguje zazwyczaj firmware urządzenia

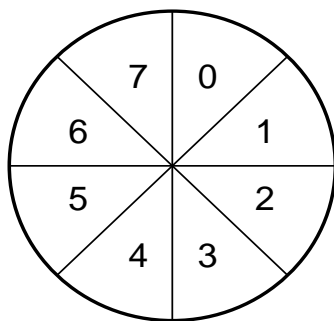
Fizyczny format dysku (1)

□ "skośność" zapisu (cylinder skew)

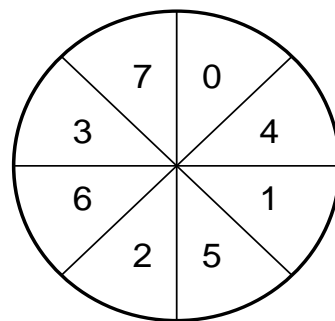


Fizyczny format dysku (2)

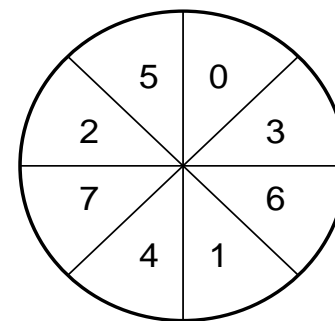
- "przeplot" sektorów
 - (a) – bez przeplotu (1:1)
 - (b) – pojedynczy (1:2)
 - (c) – podwójny (1:3)



(a)



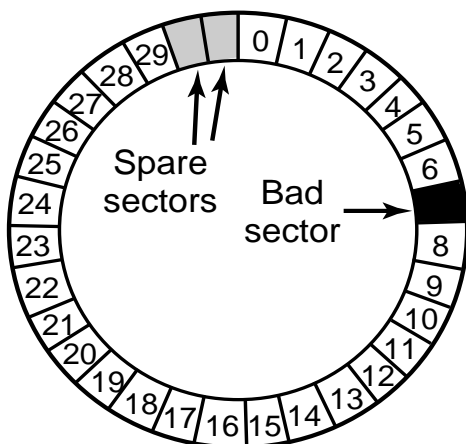
(b)



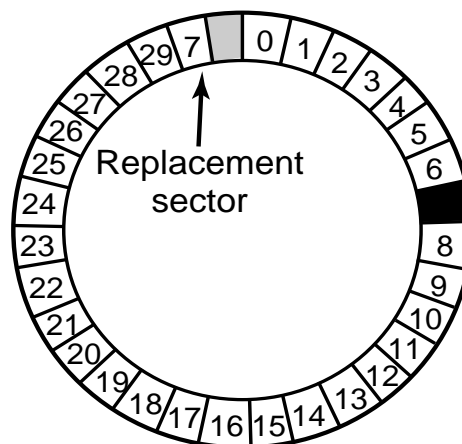
(c)

Fizyczny format dysku (3)

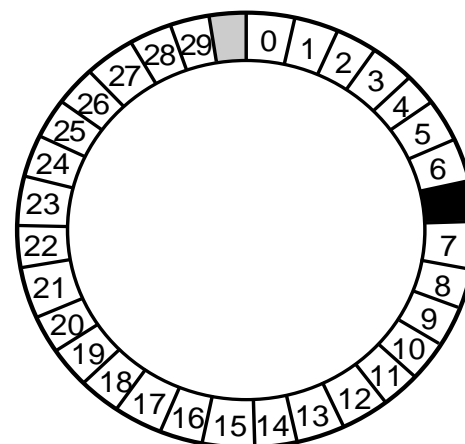
- obsługa defektów nośnika
 - (a) – ścieżka z defektem
 - (b) – przydział sektora zastępczego
 - (c) – modyfikacja formatu ścieżki



(a)



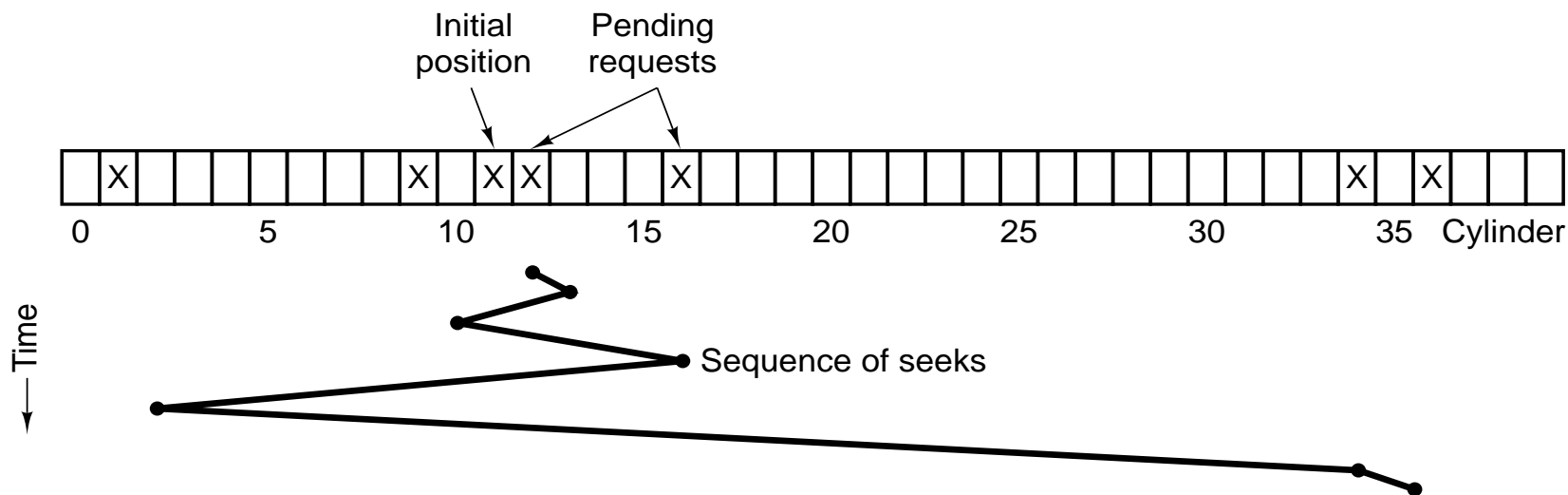
(b)



(c)

Szeregowanie dostępu do dysku (1)

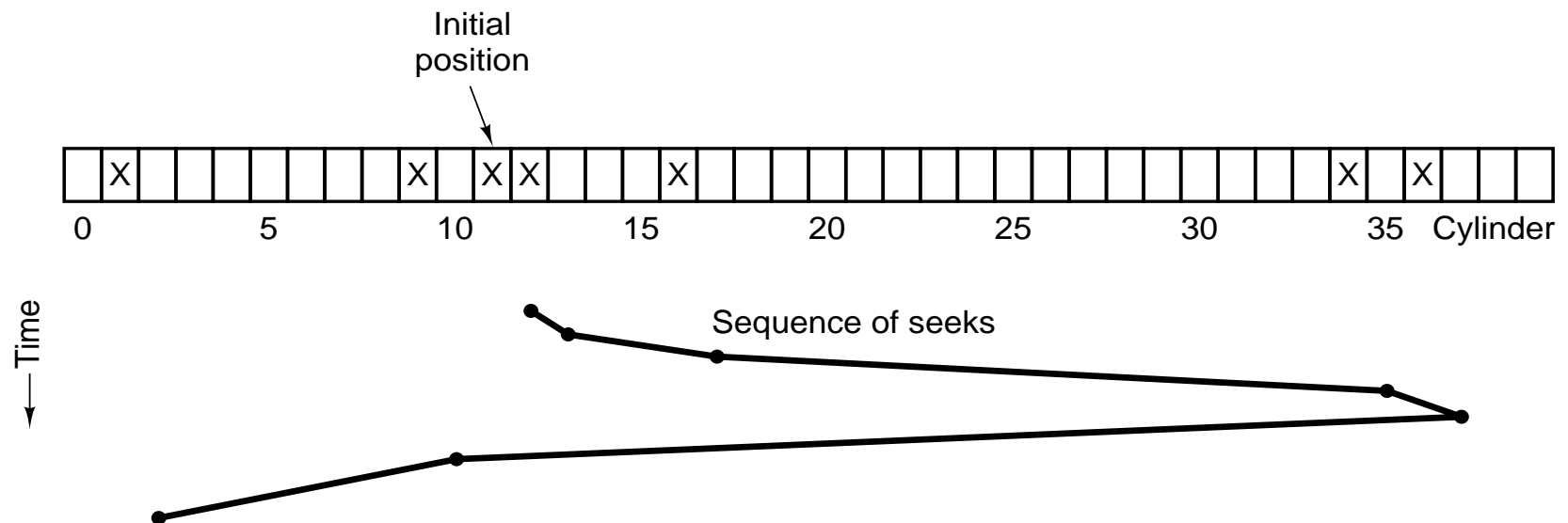
- składowe efektywnego czasu dostępu
 - czas na ustawienie głowicy (seek time)
 - opóźnienie związane z rotacją (rotational delay)
 - czas rzeczywistego transferu danych (transfer time)
 - największy wkład: czas ustawienia głowicy (!)
- algorytm SSTF (Shortest Seek Time First)
 - znacznie lepsza wydajność w porównaniu do kolejkowania (FCFS)
 - podstawowa wada: "głodzenie" zamówień do odległych cylindrów



Szeregowanie dostępu do dysku (2)

□ algorytmy SCAN i LOOK

- "algorytm windy" (elevator algorithm)
- wersja LOOK: zmiana kierunku bez osiągnięcia kresów



□ algorytmy C-SCAN i C-LOOK

- cykliczne wersje algorytmów SCAN i LOOK

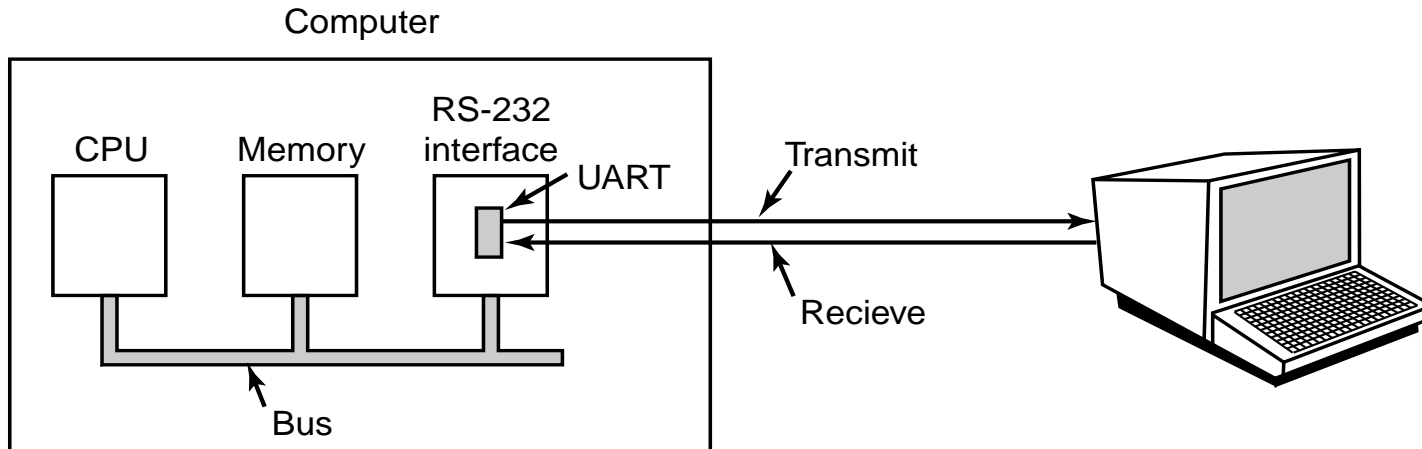
Szeregowanie dostępu do dysku (3)

□ wybór algorytmu

- zależy od rodzaju napędu dyskowego
- zależy od struktury stosowanego systemu plikowego
- może zależeć od stanu systemu plikowego, np. stopnia fragmentacji
- może zależeć od obecności obszaru wymiany/stronicowania
- może zależeć od mechanizmów obsługi pamięci wirtualnej
- może zależeć od funkcjonowania kontrolera/napędu dyskowego (!)
- może zależeć od funkcjonowania pamięci podręcznej i buforowania
- systemy serwerowe: algorytmy SCAN/LOOK i pokrewne
- systemy stacji roboczych: algorytmy typu SSTF
- najlepiej: kilka algorytmów, przełączanych adaptacyjnie

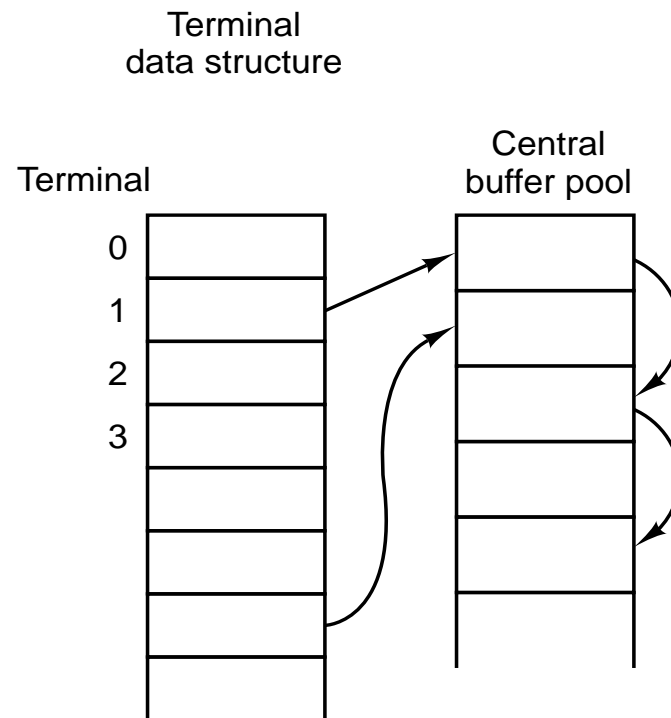
Znakowe urządzenia We/Wy

- przykład: terminal RS-232
 - transfer danych przy pomocy pojedynczych bitów (!!!)
 - komunikacja szeregową: synchroniczna lub asynchroniczna
 - duże obciążenie dla procesora i sterownika przerwań (!)

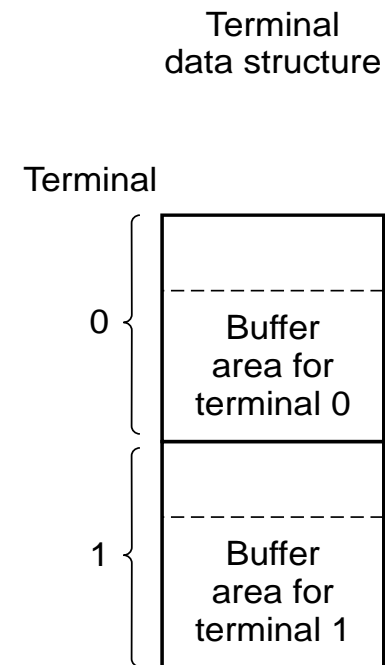


Obsługa terminali (1)

- bufory dla struktur danych
 - (a) – przydzielane z puli systemowej
 - (b) – dedykowane



(a)



(b)

Obsługa terminali (2)

- wejście terminalowe sterowane znakowo
 - przykład: znaki sterujące rodziny VTxxx (w trybie kanonicznym)

Character	POSIX name	Comment
CTRL-H	ERASE	Backspace one character
CTRL-U	KILL	Erase entire line being typed
CTRL-V	LNEXT	Interpret next character literally
CTRL-S	STOP	Stop output
CTRL-Q	START	Start output
DEL	INTR	Interrupt process (SIGINT)
CTRL-\	QUIT	Force core dump (SIGQUIT)
CTRL-D	EOF	End of file
CTRL-M	CR	Carriage return (unchangeable)
CTRL-J	NL	Linefeed (unchangeable)

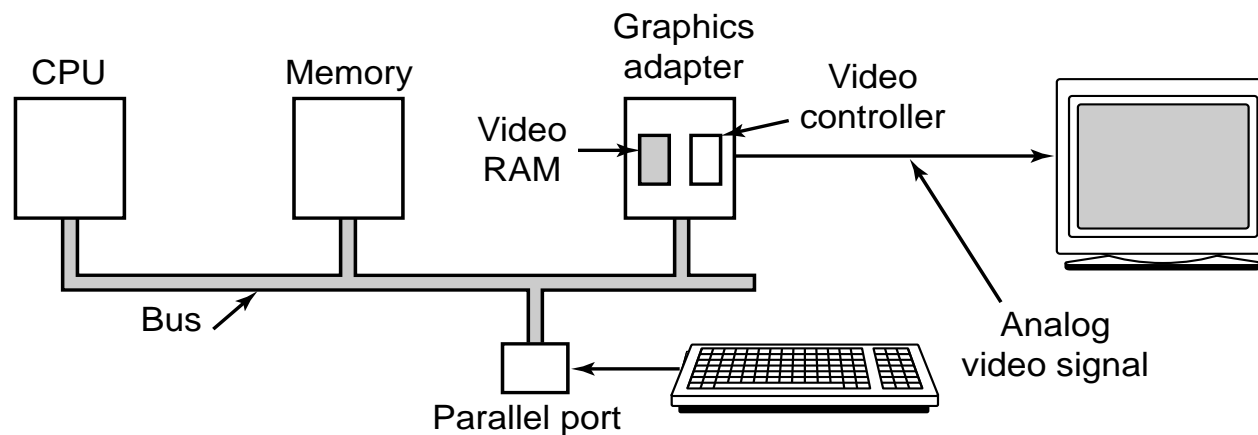
Obsługa terminali (3)

- wyjście terminalowe sterowane znakowo
 - przykład: sekwencje sterujące ANSI (ESC=0x1b ASCII)

Escape sequence	Meaning
ESC [<i>n</i> A	Move up <i>n</i> lines
ESC [<i>n</i> B	Move down <i>n</i> lines
ESC [<i>n</i> C	Move right <i>n</i> spaces
ESC [<i>n</i> D	Move left <i>n</i> spaces
ESC [<i>m</i> ; <i>n</i> H	Move cursor to (<i>m</i> , <i>n</i>)
ESC [<i>s</i> J	Clear screen from cursor (0 to end, 1 from start, 2 all)
ESC [<i>s</i> K	Clear line from cursor (0 to end, 1 from start, 2 all)
ESC [<i>n</i> L	Insert <i>n</i> lines at cursor
ESC [<i>n</i> M	Delete <i>n</i> lines at cursor
ESC [<i>n</i> P	Delete <i>n</i> chars at cursor
ESC [<i>n</i> @	Insert <i>n</i> chars at cursor
ESC [<i>n</i> m	Enable rendition <i>n</i> (0=normal, 4=bold, 5=blinking, 7=reverse)
ESC M	Scroll the screen backward if the cursor is on the top line

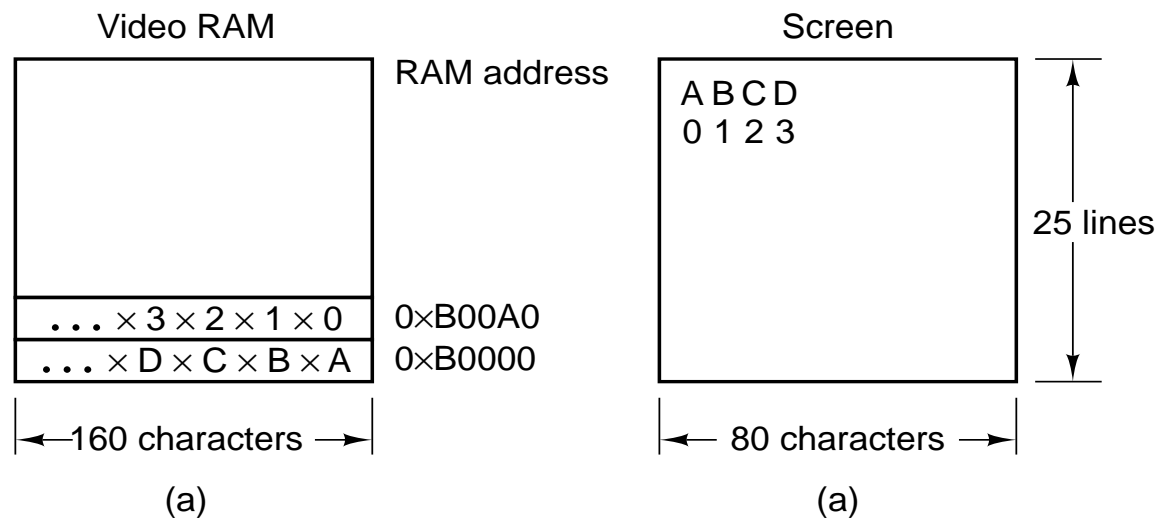
Obsługa terminali (4)

- typowy sprzęt terminalowy
 - We/Wy odwzorowane w pamięci (memory-mapped I/O)
 - sterownik zapisuje dane bezpośrednio do pamięci video
 - klawiatura generuje kody, poddawane odpowiedniej konwersji
 - wymienne tabele konwersji dla klawiatury i wyświetlacza
 - standaryzowane mapy klawiatury i tzw. 'strony kodowe'



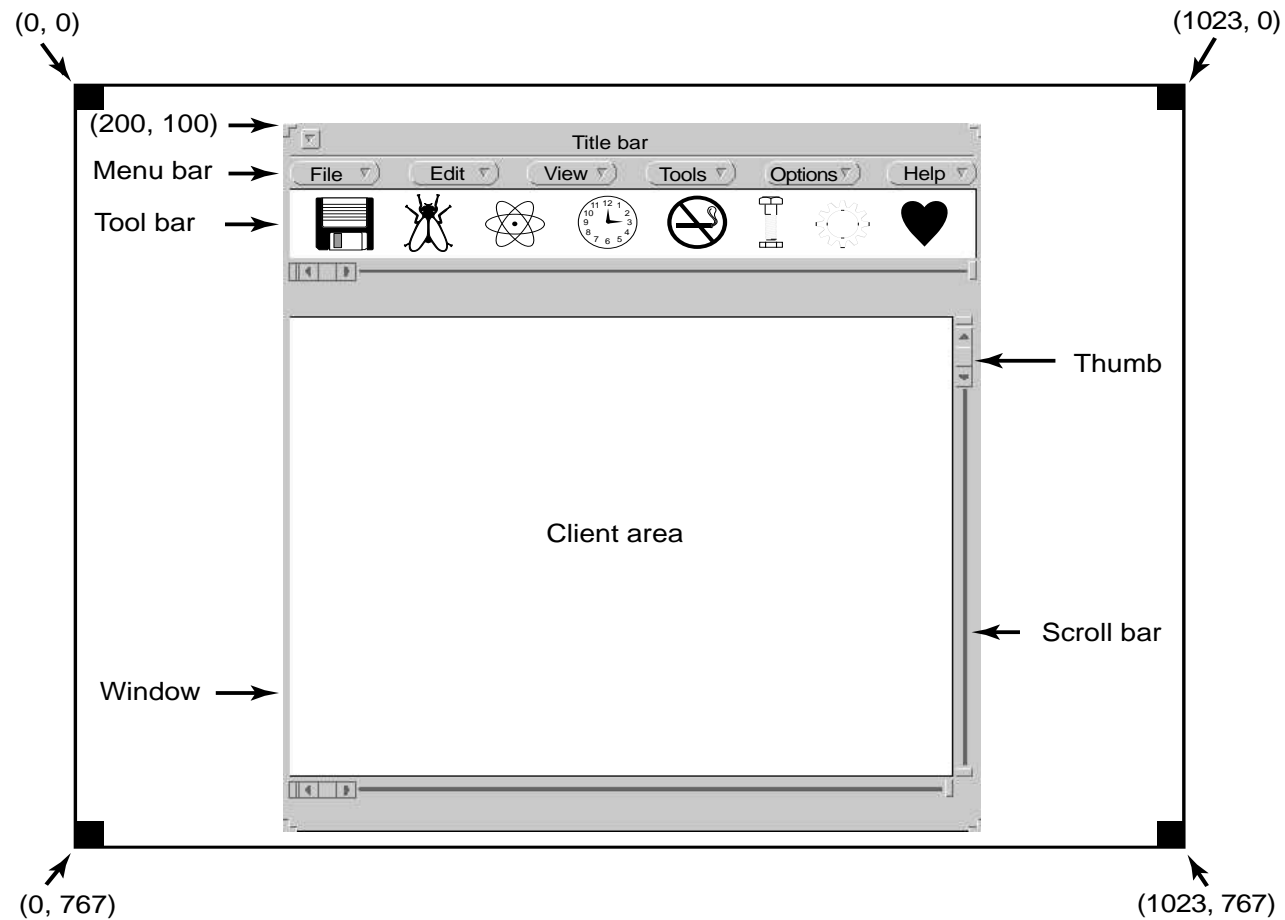
Obsługa terminali (5)

- przykład: wyświetlacz monochromatyczny
 - IBM PC MDA (Mono Display Adapter), tryb znakowy
 - dwa bajty pamięci video kodują jeden wyświetlany znak
 - poniżej: 'x' oznacza odpowiedni bajt atrybutów



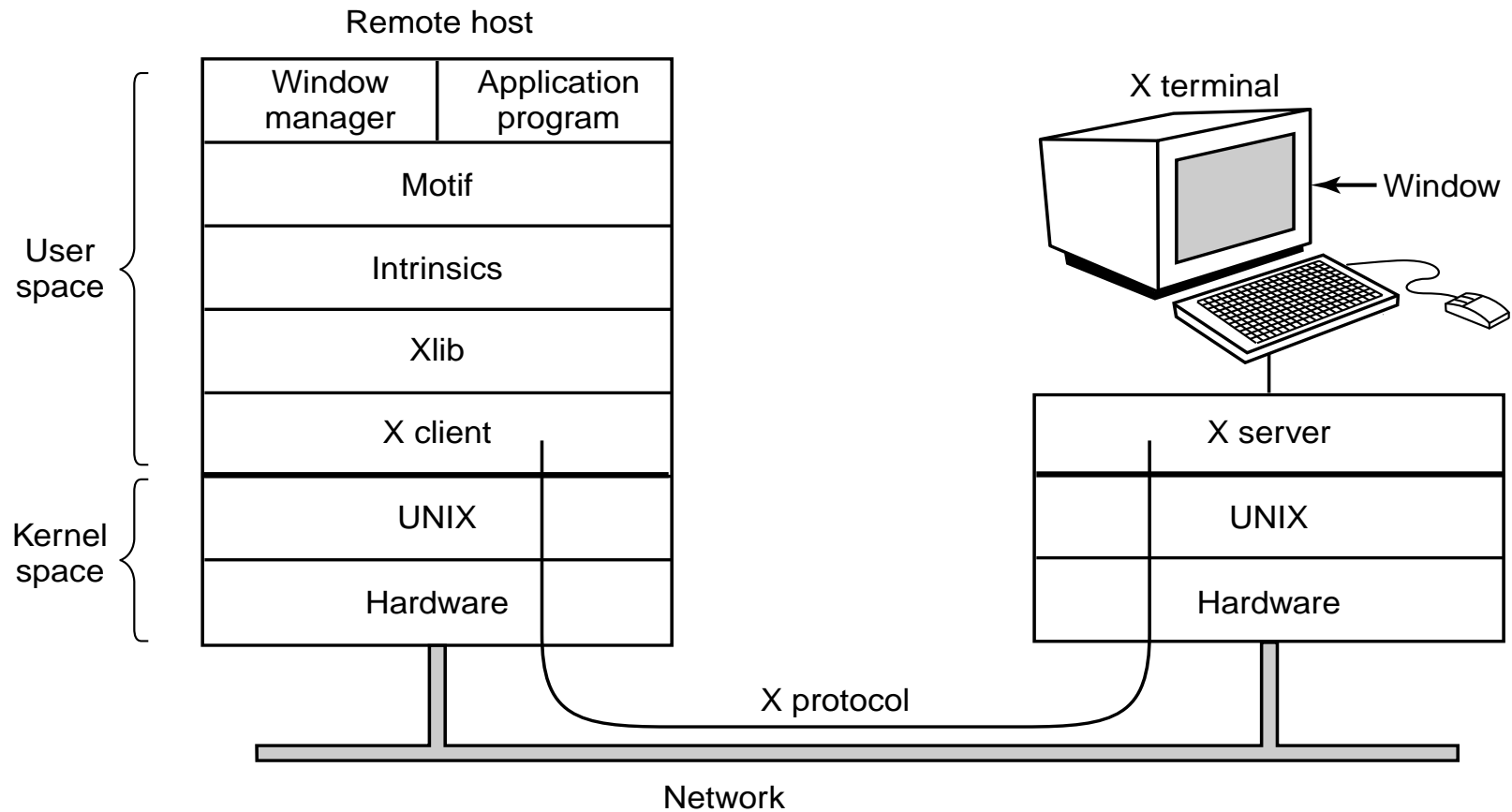
Obsługa terminali (6)

- przykład: wyświetlacz IBM XGA (1024x768)
 - typowe 'okno' (X Window)



Obsługa terminali (7)

- system X Window (MIT)



Obsługa terminali (8)

□ 'szkielet' typowej aplikacji X Window

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>

main(int argc, char *argv[])
{
    Display disp;                /* server identifier */
    Window win;                  /* window identifier */
    GC gc;                       /* graphic context identifier */
    XEvent event;                /* storage for one event */
    int running = 1;

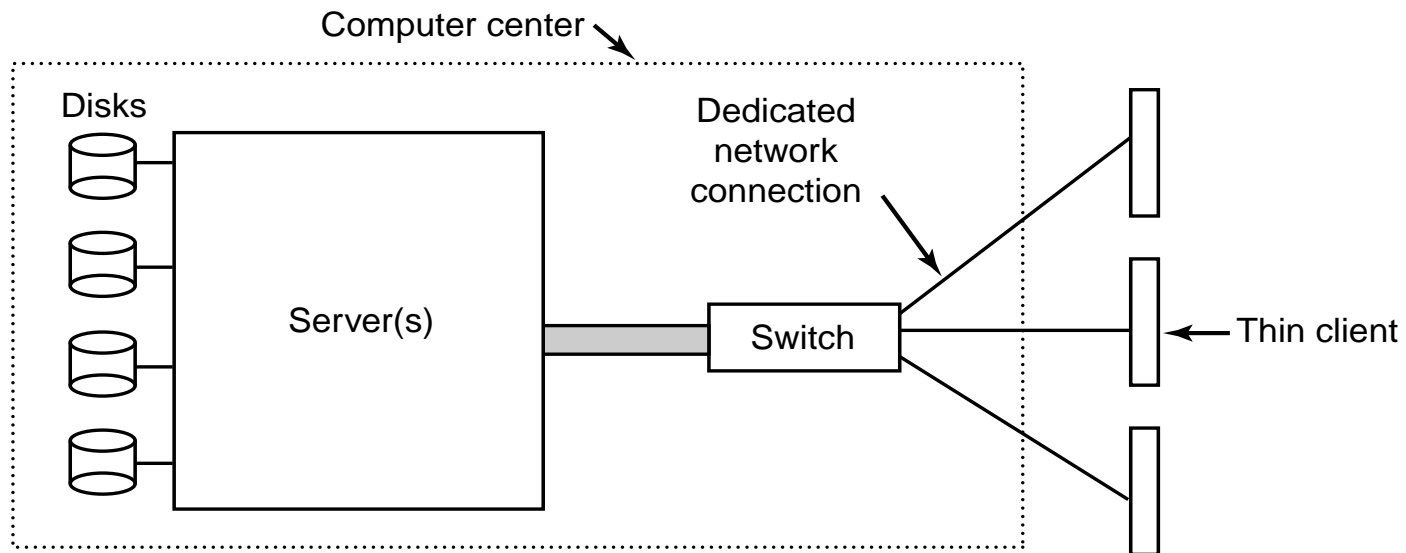
    disp = XOpenDisplay("display_name"); /* connect to the X server */
    win = XCreateSimpleWindow(disp, ... ); /* allocate memory for new window */
    XSetStandardProperties(disp, ...);     /* announces window to window mgr */
    gc = XCreateGC(disp, win, 0, 0);       /* create graphic context */
    XSelectInput(disp, win, ButtonPressMask | KeyPressMask | ExposureMask);
    XMapRaised(disp, win);                /* display window; send Expose event */

    while (running) {
        XNextEvent(disp, &event);         /* get next event */
        switch (event.type) {
            case Expose:    ...; break;    /* repaint window */
            case ButtonPress: ...; break;  /* process mouse click */
            case Keypress:  ...; break;    /* process keyboard input */
        }
    }

    XFreeGC(disp, gc);                /* release graphic context */
    XDestroyWindow(disp, win);         /* deallocate window's memory space */
    XCloseDisplay(disp);               /* tear down network connection */
}
```

Obsługa terminali (9)

- dedykowane sprzętowe terminale sieciowe
 - typowy model 'szczupłego klienta' (thin client)



===

06 – Systemy plikowe

- ☐ Pliki – struktura, organizacja, etc.
- ☐ Struktury katalogowe
- ☐ Implementacja systemu plikowego
- ☐ Przykładowe systemy plikowe

>>>

Systemy plikowe – założenia

- zorganizowane przechowywanie danych
 - plik --> (podstawowa) jednostka przetwarzania
 - odpowiednio duża pojemność pliku i systemu plikowego
 - adekwatny opis zawartości (nazewnictwo, atrybuty, etc)
 - adekwatna organizacja strukturalna (strefy, katalogi, etc.)
 - możliwość klasyfikacji (wg. nazwy, atrybutów, etc.)
 - możliwość dedykowania/współdzielenia/ochrony dostępu
 - możliwość tworzenia/wykorzystania kopii zapasowych (backup)
 - możliwość sprawdzenia poprawności strukturalnej i ew. naprawy
 - niezależność od rodzaju medium i sposobu zapisu/odczytu
- dostęp do zawartości pliku
 - sekwencyjny
 - swobodny
- typowe operacje plikowe (funkcje systemowe)
 - create(), delete(), open(), close(), read(), write()
 - append(), seek(), get_attr(), set_attr(), rename()

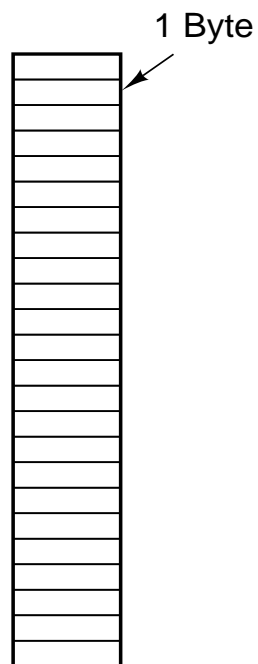
Klasyfikacja zawartości

- typowo: wg. tzw. "rozszerzenia" nazwy pliku

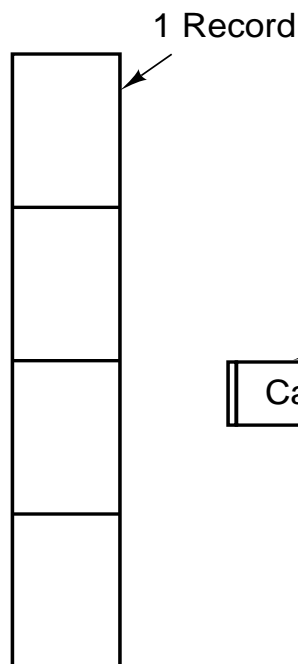
Extension	Meaning
file.bak	Backup file
file.c	C source program
file.gif	Compuserve Graphical Interchange Format image
file.hlp	Help file
file.html	World Wide Web HyperText Markup Language document
file.jpg	Still picture encoded with the JPEG standard
file.mp3	Music encoded in MPEG layer 3 audio format
file.mpg	Movie encoded with the MPEG standard
file.o	Object file (compiler output, not yet linked)
file.pdf	Portable Document Format file
file.ps	PostScript file
file.tex	Input for the TEX formatting program
file.txt	General text file
file.zip	Compressed archive

Struktura pliku

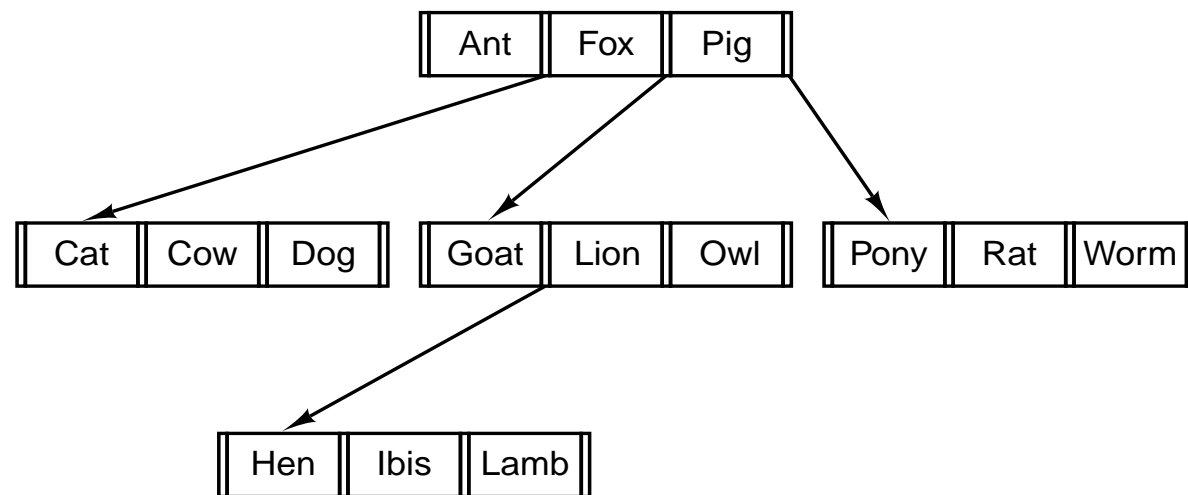
- trzy podstawowe rodzaje organizacji
 - (a) – bajtowa sekwencyjna
 - (b) – rekordowa sekwencyjna
 - (c) – rekordowa drzewiasta



(a)



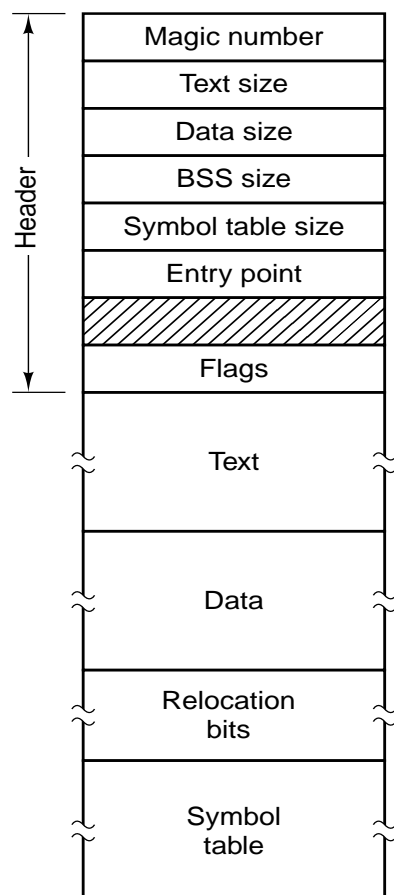
(b)



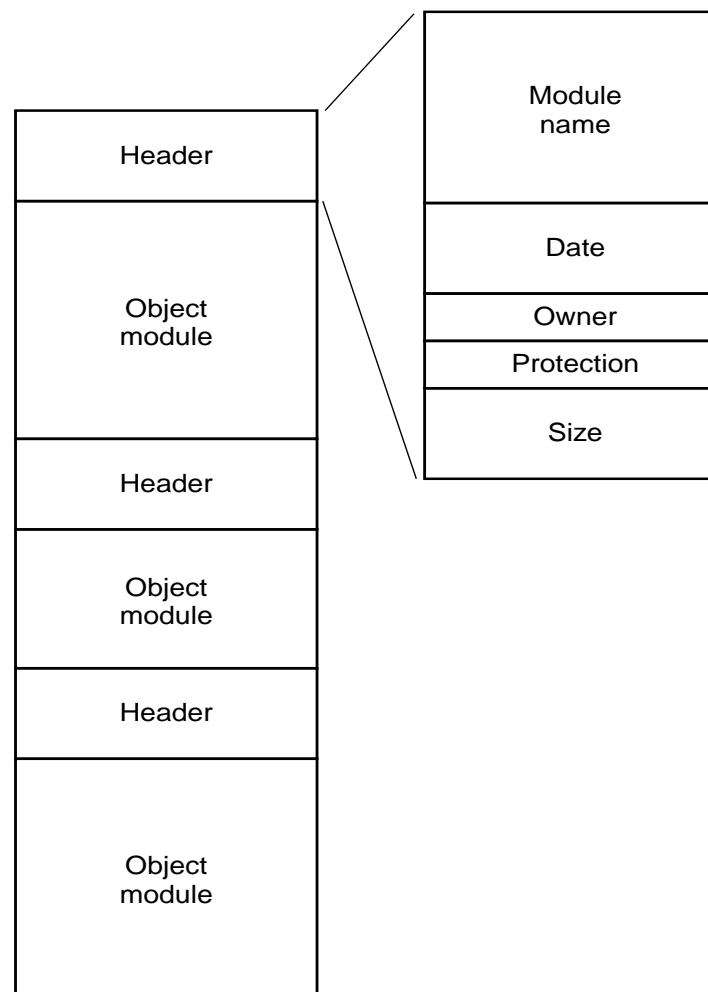
(c)

Rodzaje plików

- typowe przykłady
 - (a) – plik wykonywalny
 - (b) – plik biblioteczny (archiwum)



(a)



(b)

Atrybuty pliku

Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file has last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

Operacje plikowe – przykład

```
/* File copy program. Error checking and reporting is minimal. */

#include <sys/types.h>                /* include necessary header files */
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[]);    /* ANSI prototype */

#define BUF_SIZE 4096                /* use a buffer size of 4096 bytes */
#define OUTPUT_MODE 0700             /* protection bits for output file */

int main(int argc, char *argv[])
{
    int in_fd, out_fd, rd_count, wt_count;
    char buffer[BUF_SIZE];

    if (argc != 3) exit(1);           /* syntax error if argc is not 3 */

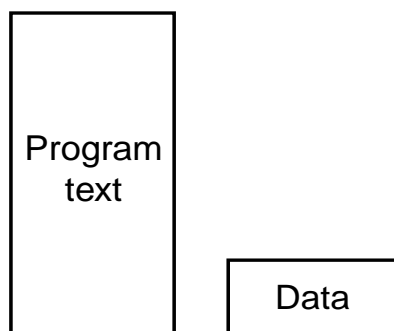
    /* Open the input file and create the output file */
    in_fd = open(argv[1], O_RDONLY);  /* open the source file */
    if (in_fd < 0) exit(2);           /* if it cannot be opened, exit */
    out_fd = creat(argv[2], OUTPUT_MODE); /* create the destination file */
    if (out_fd < 0) exit(3);          /* if it cannot be created, exit */

    /* Copy loop */
    while (TRUE) {
        rd_count = read(in_fd, buffer, BUF_SIZE); /* read a block of data */
        if (rd_count <= 0) break;        /* if end of file or error, exit loop */
        wt_count = write(out_fd, buffer, rd_count); /* write data */
        if (wt_count <= 0) exit(4);      /* wt_count <= 0 is an error */
    }

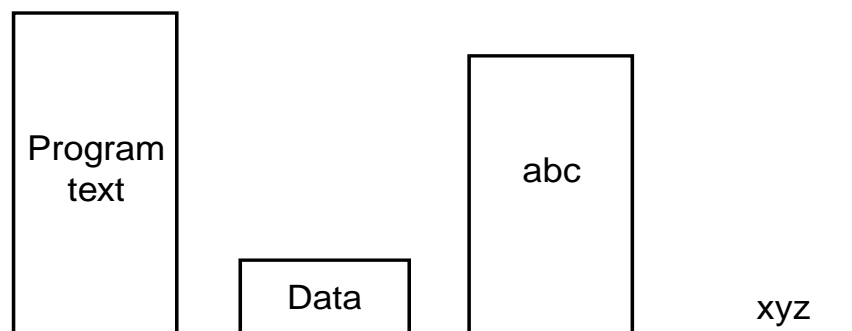
    /* Close the files */
    close(in_fd);
    close(out_fd);
    if (rd_count == 0)                  /* no error on last read */
        exit(0);
    else
        exit(5);                      /* error on last read */
}
```

Odwzorowanie zawartości pliku

- bufory i wskaźniki plikowe
 - standardowe funkcje biblioteczne
 - konieczność niezależnego przydziału pamięci dla buforów
- pliki odwzorowane w pamięci
 - plik odpowiada segmentowi pamięci
 - jednorodny dostęp do pliku i do pamięci
- przykład:
 - (a) – proces wykorzystujący standardowe funkcje plikowe
 - (b) – odwzorowanie plików 'abc' i 'xyz' (nowo utworzony)



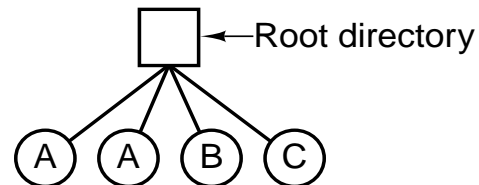
(a)



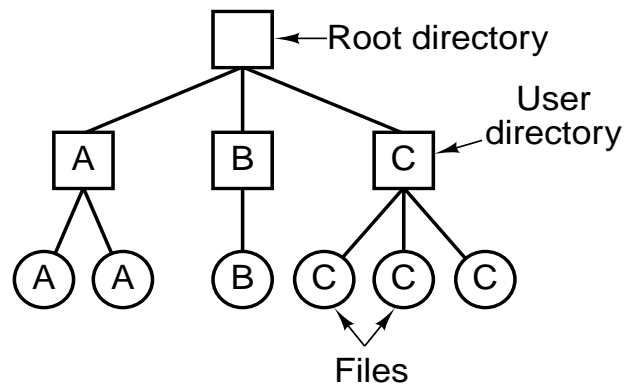
(b)

Katalogi (1)

- pojedynczy katalog
 - cztery pliki, trzech użytkowników (A,B i C)

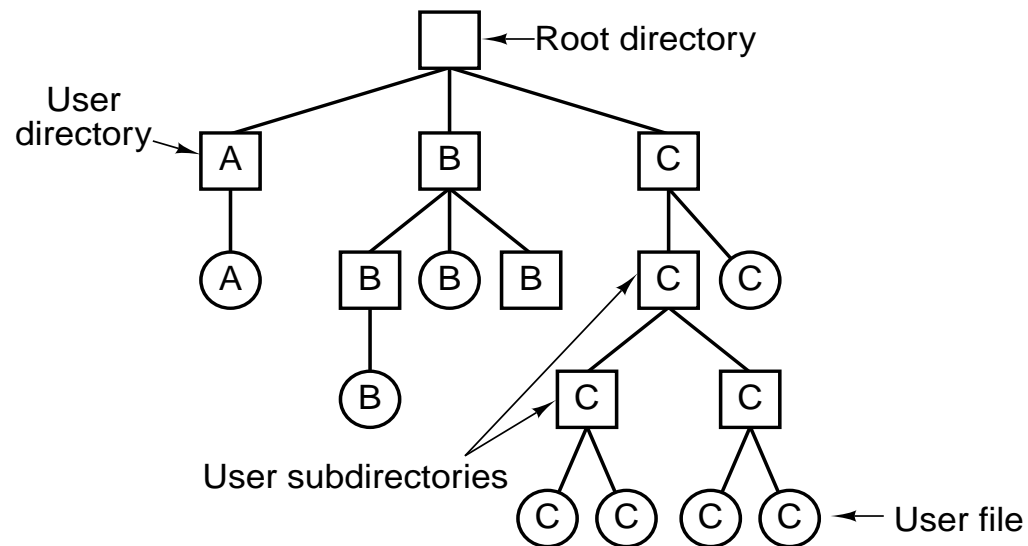


- równorzędne katalogi
 - każdy użytkownik dysponuje własnym katalogiem



Katalogi (2)

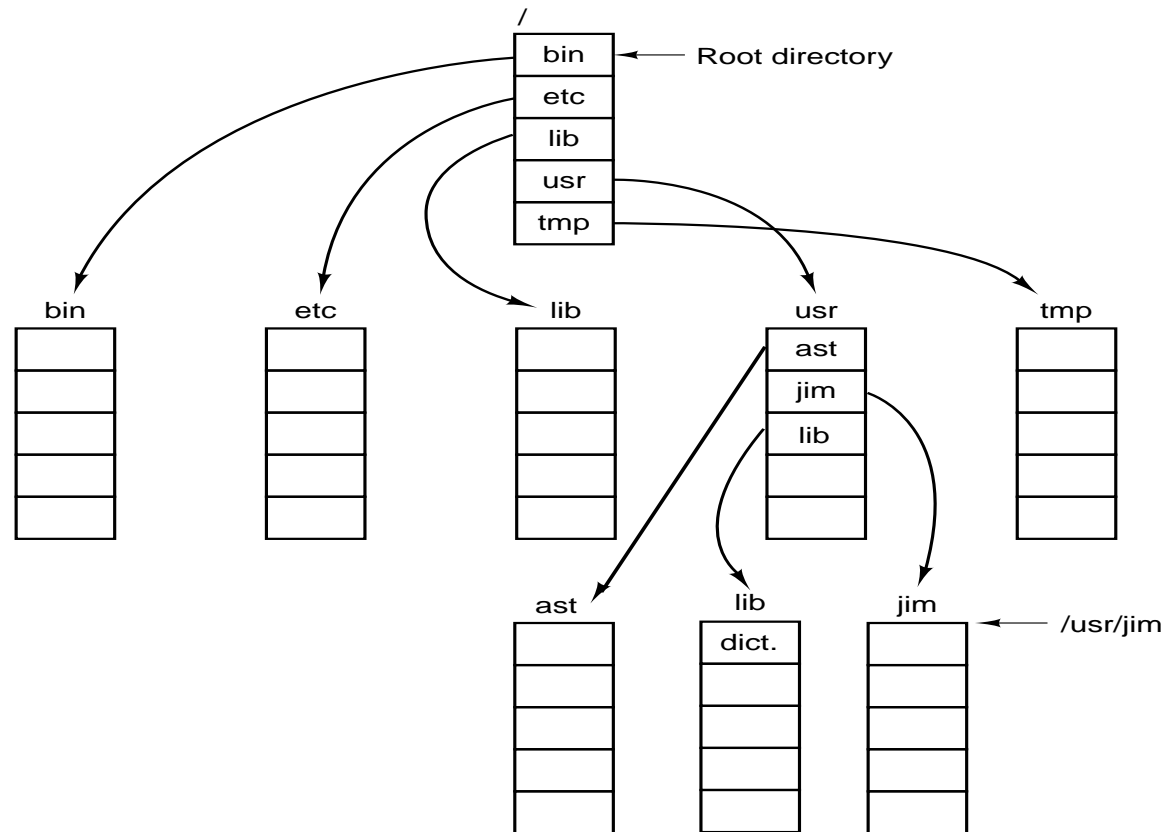
- hierarchiczny system katalogowy
 - użytkownicy mogą tworzyć podkatalogi



- typowe operacje katalogowe (funkcje systemowe)
 - `create()`, `delete()`, `open_dir()`, `close_dir()`
 - `read_dir()`, `rename()`, `link()`, `unlink()`

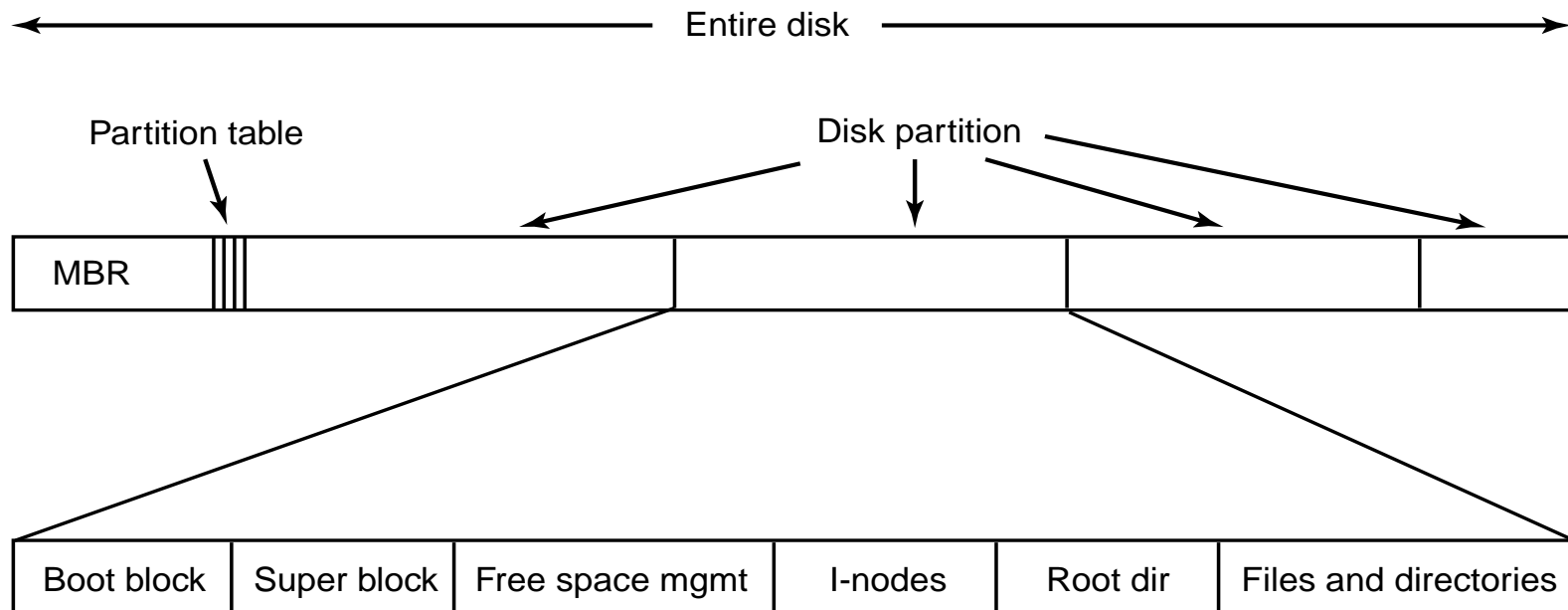
Katalogi (3)

- jednorodne drzewo katalogów (UNIX)



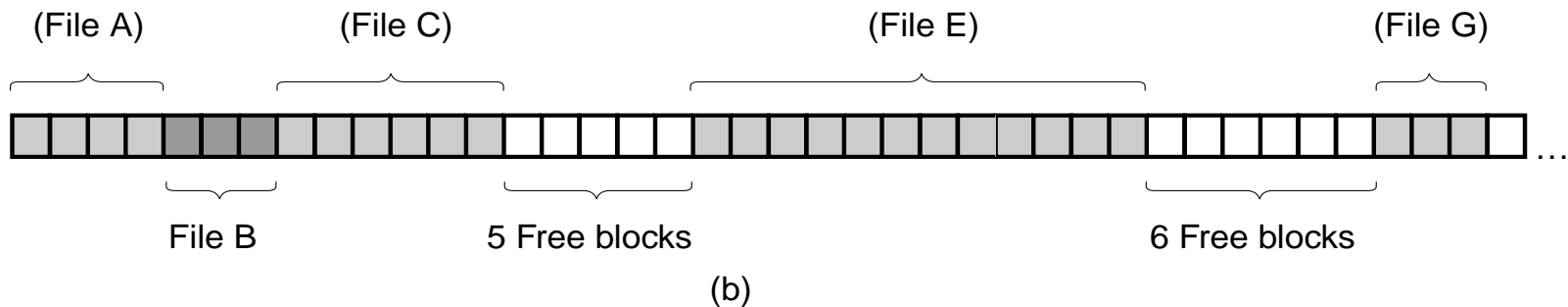
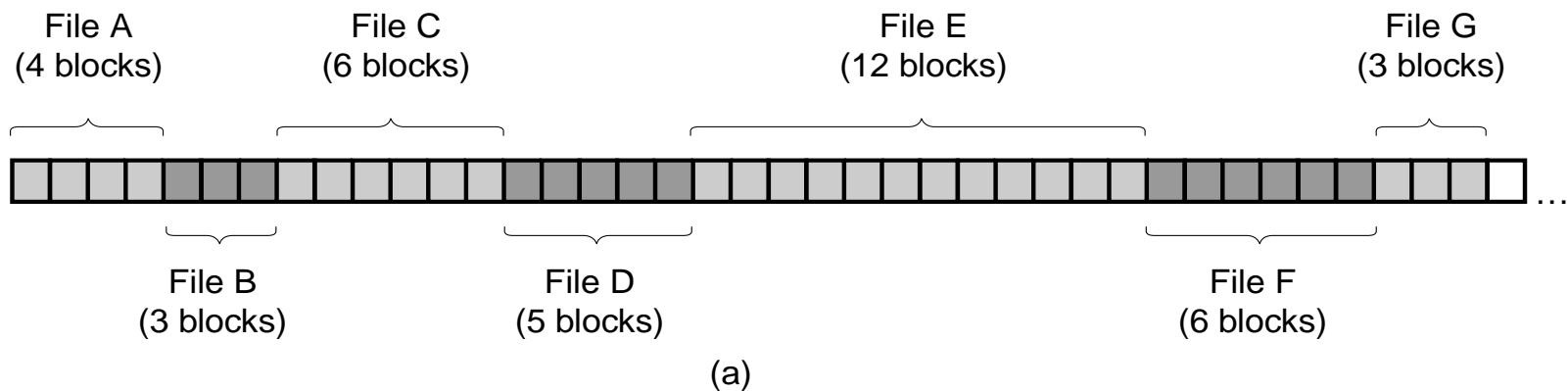
Dyski a systemy plikowe

- typowa struktura strefowa dysku
 - podział powierzchni dysku na partycje (wolumeny)
 - zainicjowany system plikowy (UNIX)



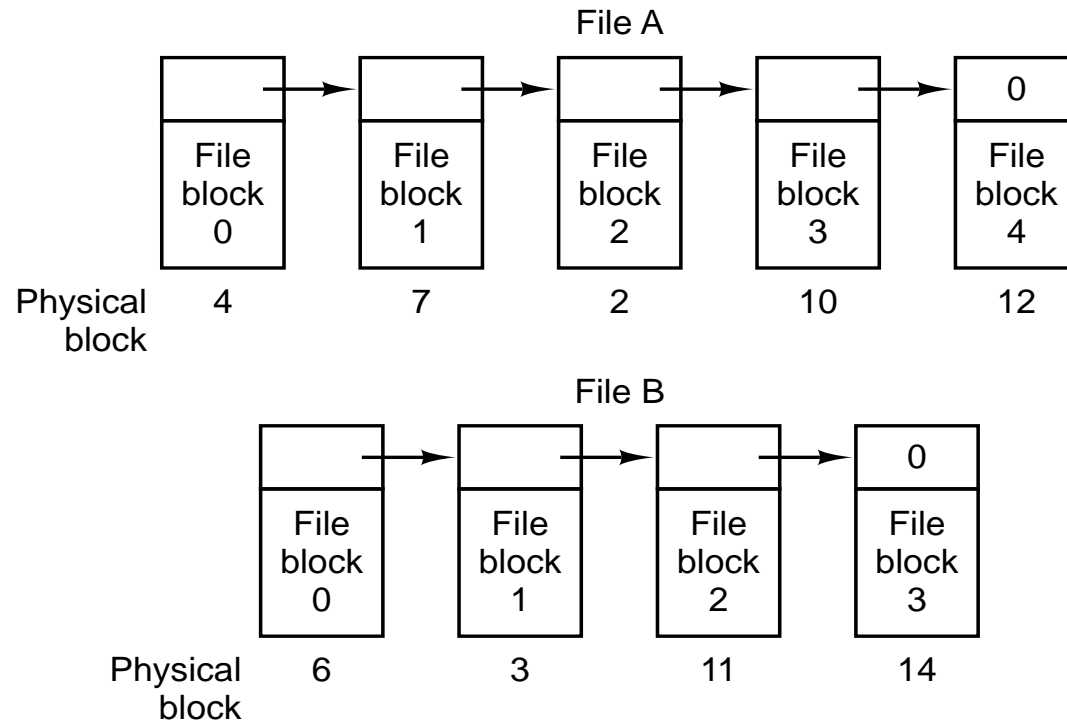
Alokacja plików (1)

- przydział ciągły (spójny) powierzchni dyskowej
 - (a) – spójny zapis 7 plików (A–G)
 - (b) – stan po usunięciu plików D i E



Alokacja plików (2)

- przydział wg. połączonej listy bloków alokacyjnych



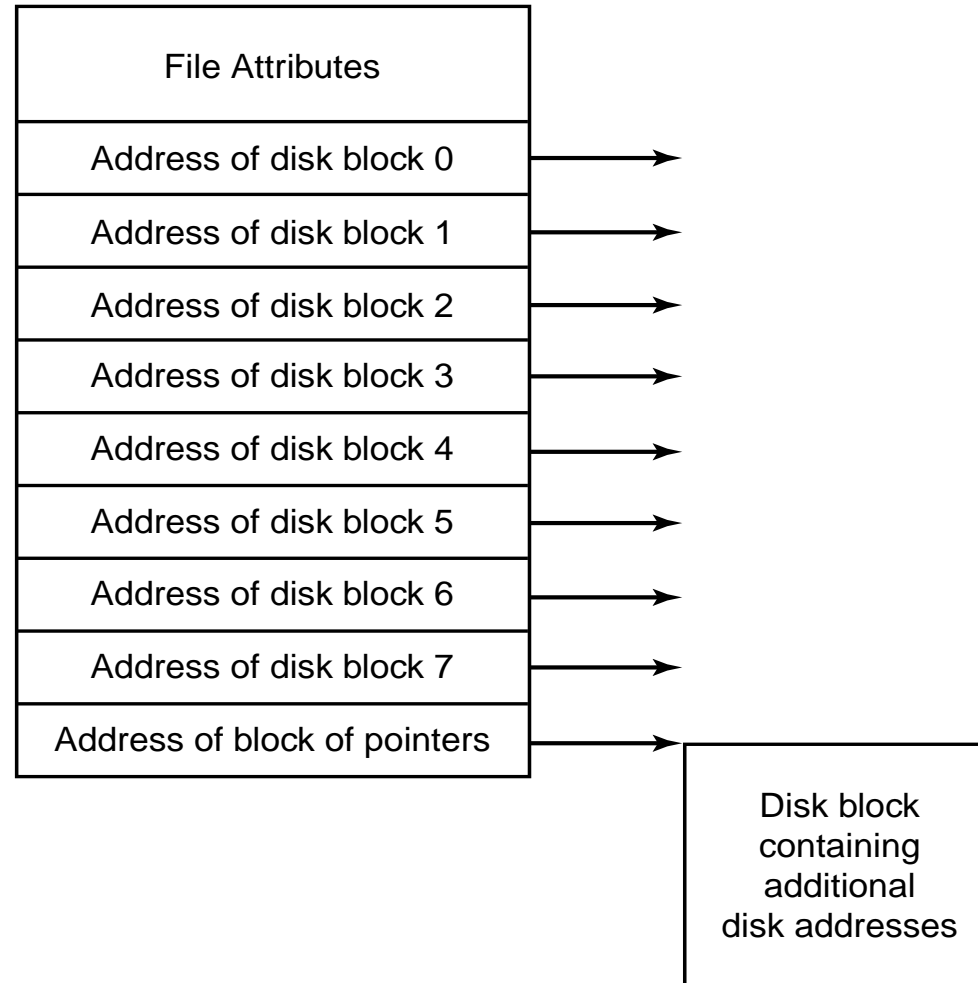
Alokacja plików (3)

- tablica alokacji plików w pamięci

Physical block		
0		
1		
2	10	
3	11	
4	7	← File A starts here
5		
6	3	← File B starts here
7	2	
8		
9		
10	12	
11	14	
12	-1	
13		
14	-1	
15		← Unused block

Alokacja plików (4)

- przykładowy blok indeksowy (i-node) na dysku



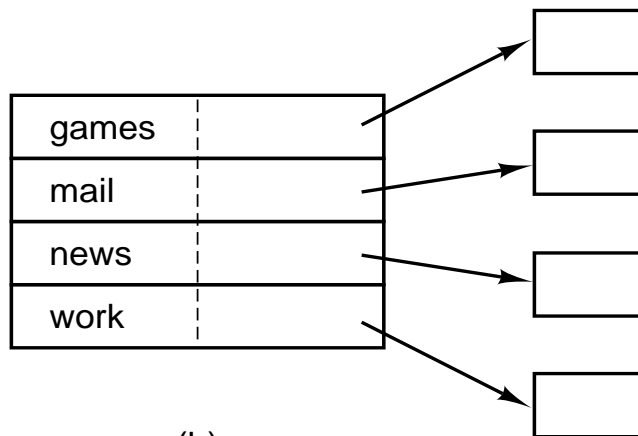
Implementacja katalogu (1)

□ typowe przykłady

- (a) – prosty katalog, ustalony format opisu
- (b) – katalog z odniesieniami do bloków indeksowych

games	attributes
mail	attributes
news	attributes
work	attributes

(a)

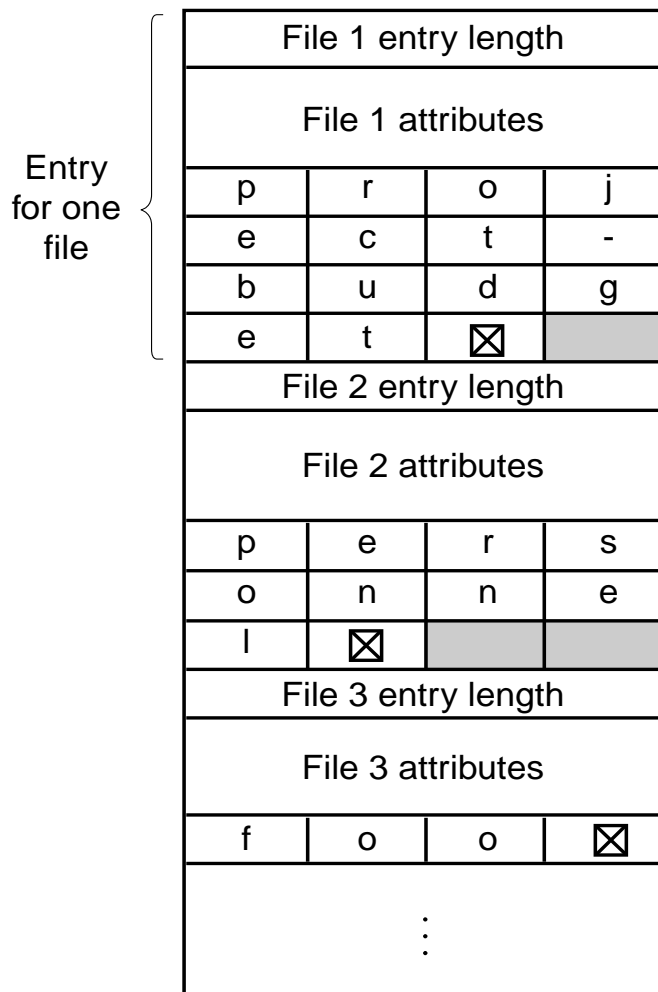


(b)

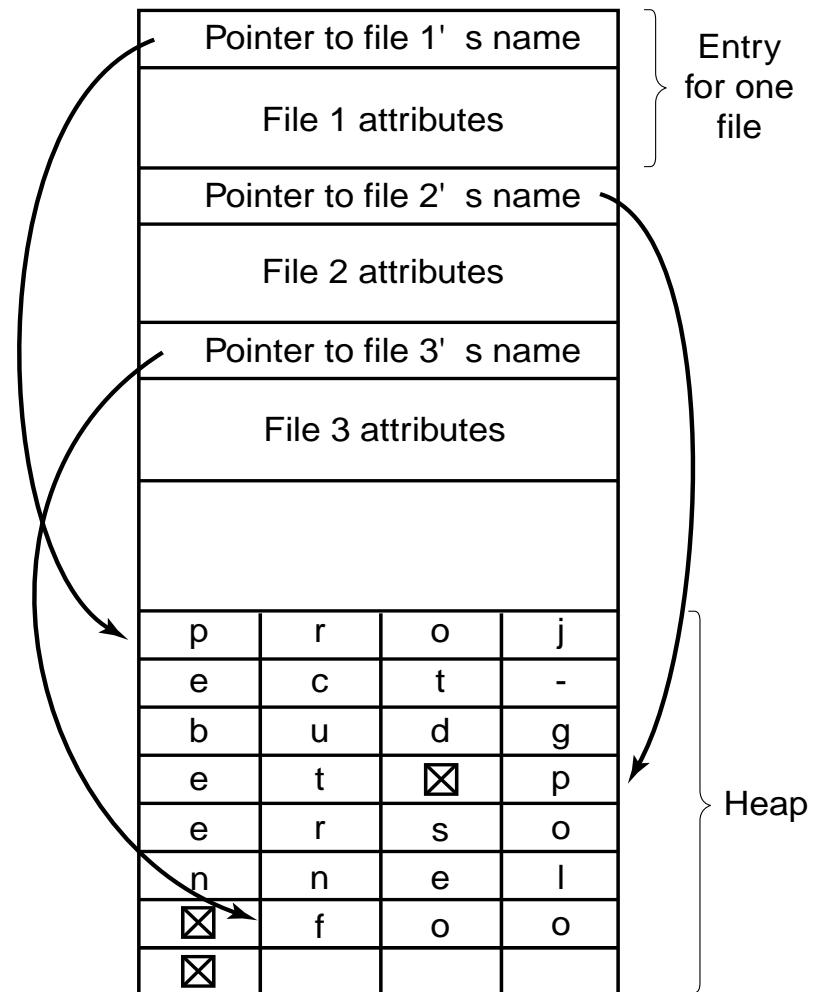
Data structure
containing the
attributes

Implementacja katalogu (2)

- różne sposoby zapisu "długich" nazw plików



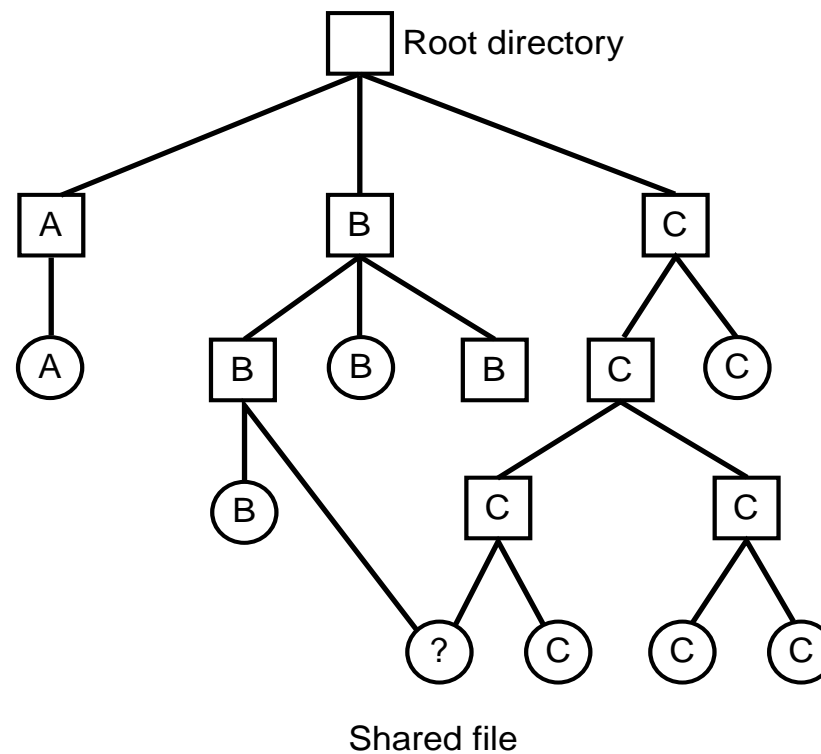
(a)



(b)

Implementacja katalogu (3)

- odniesienia wielokrotne i cykliczne
 - przykład: plik zapisany w dwóch katalogach

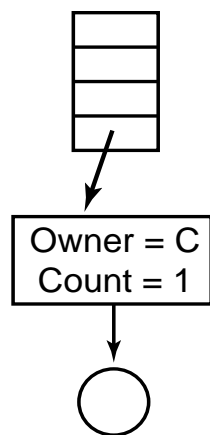


Implementacja katalogu (4)

□ dowiązania wielokrotne

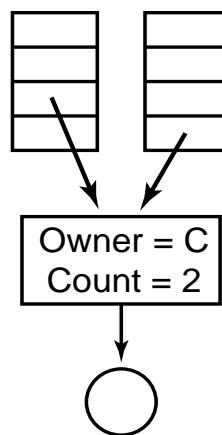
- (a) – plik utworzony przez użytkownika C
- (b) – dodatkowe dowiązanie do katalogu użytkownika B
- (c) – użytkownik C usunął dowiązanie w swoim katalogu

C' s directory



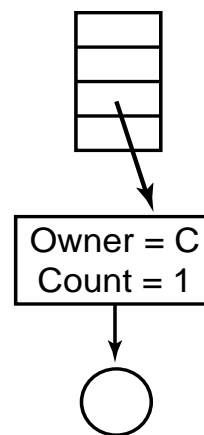
(a)

B' s directory C' s directory



(b)

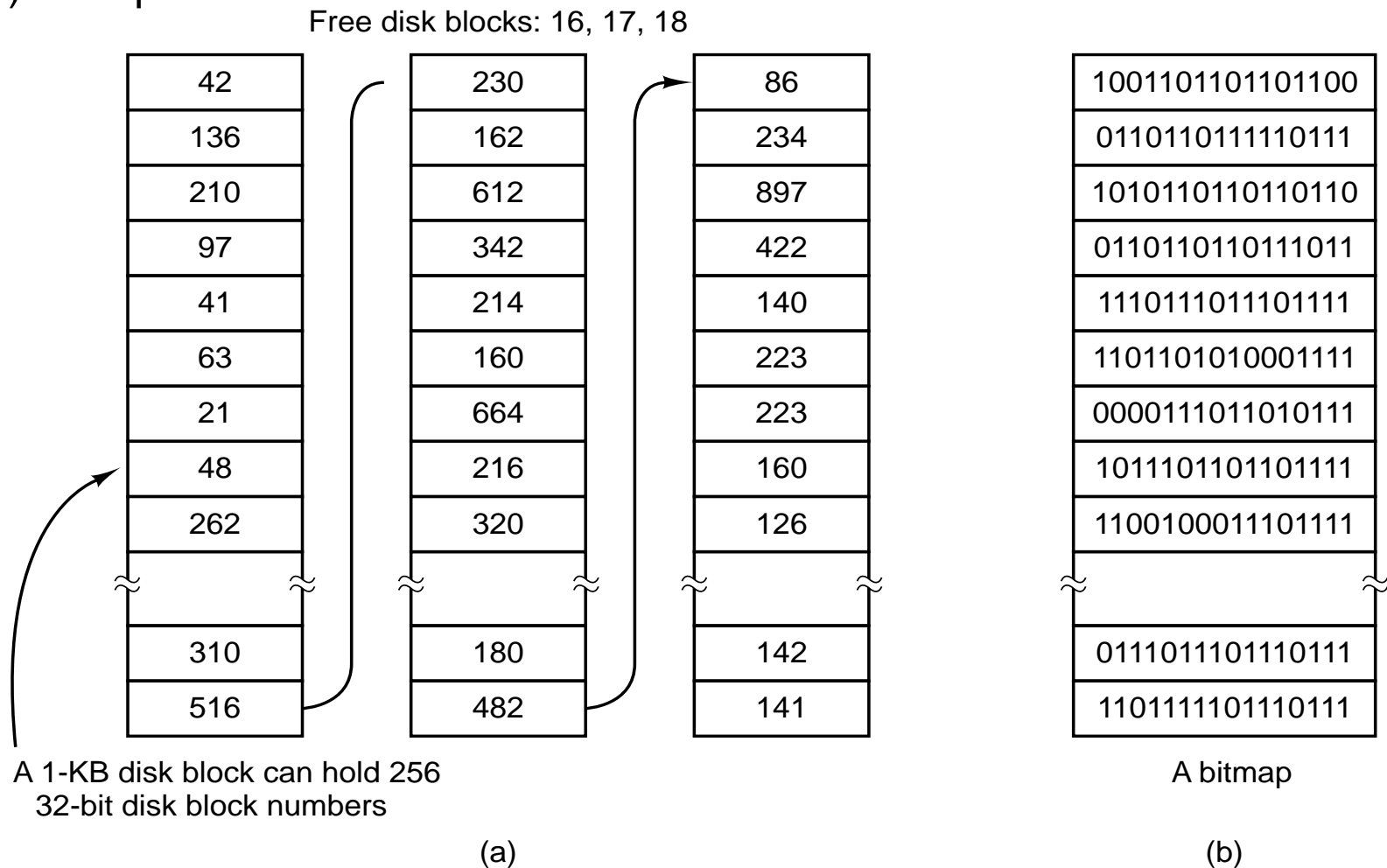
B' s directory



(c)

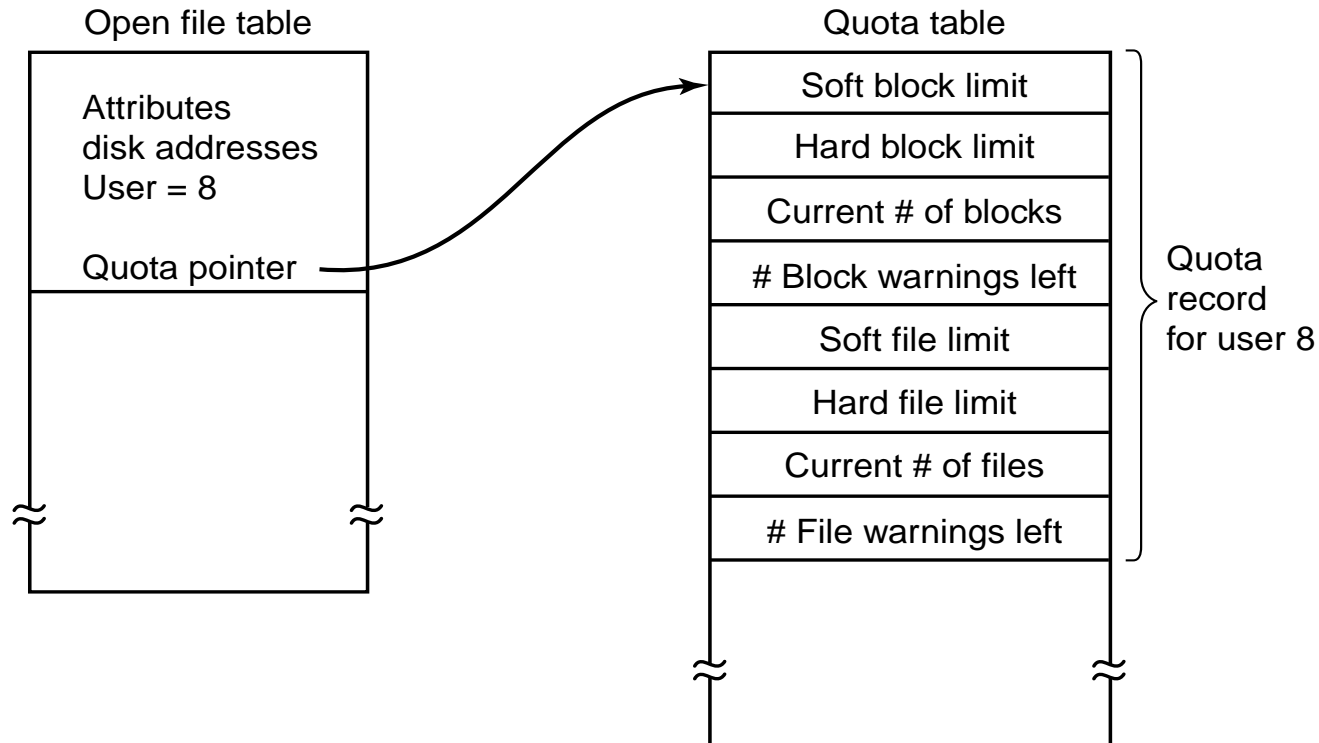
Zarządzanie powierzchnią dyskową (1)

- opis rozmieszczenia wolnych bloków alokacyjnych
 - (a) – powiązana lista bloków
 - (b) – mapa bitowa



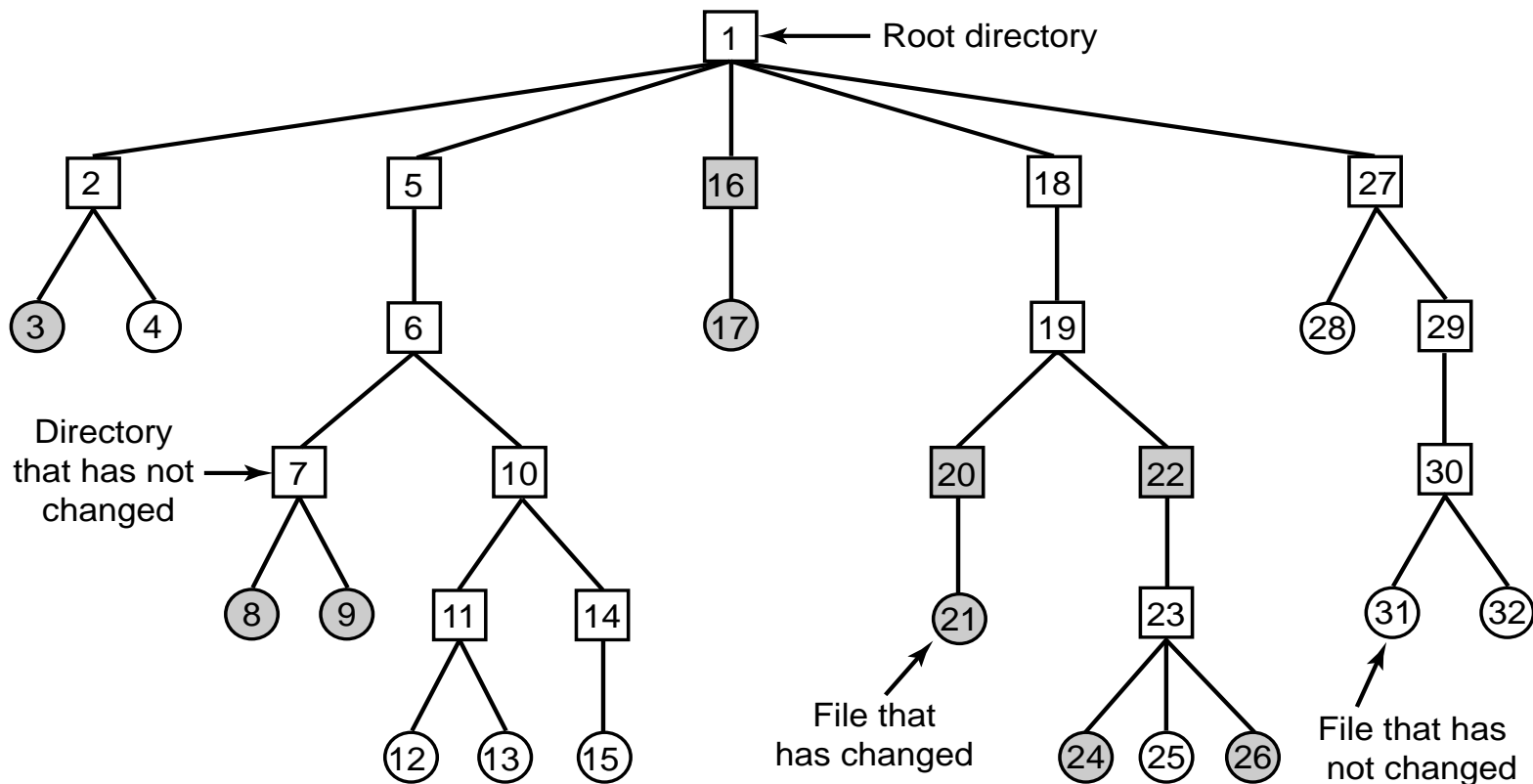
Zarządzanie powierzchnią dyskową (2)

- ograniczony przydział dla użytkownika (quota)



Kopiowanie i odtwarzanie (1)

- przykład: kopiowanie selektywne (dumping)
 - pliki i katalogi oznaczone wg. numeru bloku indeksowego
 - zacienione elementy uległy modyfikacji



Kopiowanie i odtwarzanie (2)

- kopiowanie według map bitowych bloków indeksowych
 - przykładowa sekwencja (a) – (d)

(a)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

(b)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

(c)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

(d)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Kopiowanie i odtwarzanie (3)

- sprawdzanie spójności systemu plikowego
 - porównanie listy bloków wolnych i zajętych
 - (a) – spójny system plikowy
 - (b) – brak bloku (2)
 - (c) – duplikat na liście wolnych bloków (4)
 - (d) – duplikat na liście zajętych bloków (5)

Block number															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
Blocks in use															
0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1
Free blocks															

(a)

Block number															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
Blocks in use															
0	0	0	0	1	0	0	0	0	1	1	0	0	0	1	1
Free blocks															

(b)

Block number															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
Blocks in use															
0	0	1	0	2	0	0	0	0	1	1	0	0	0	1	1
Free blocks															

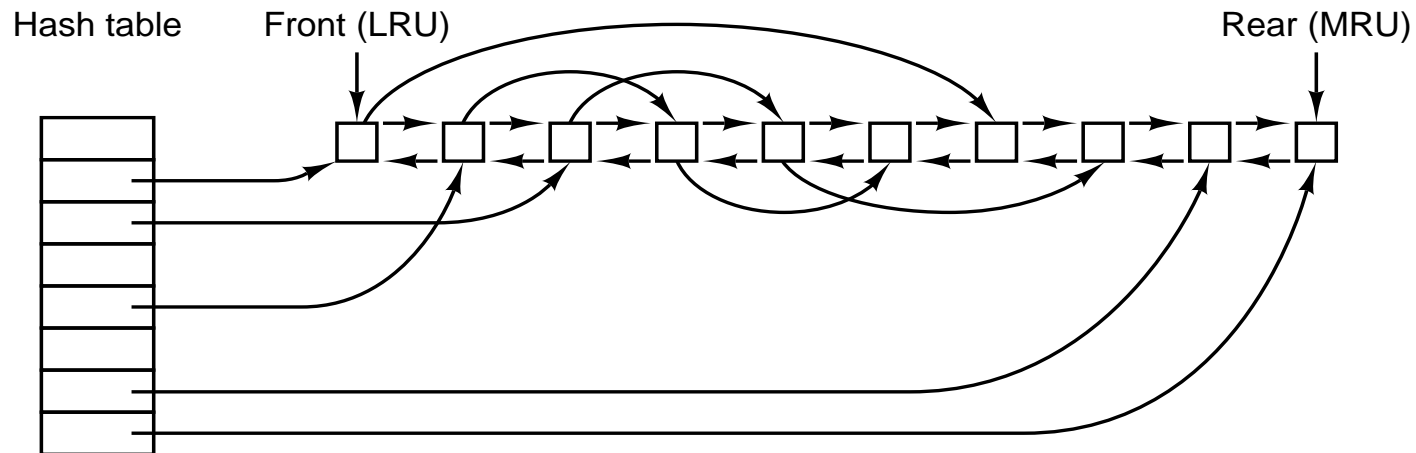
(c)

Block number															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	2	1	1	1	0	0	1	1	1	0	0
Blocks in use															
0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1
Free blocks															

(d)

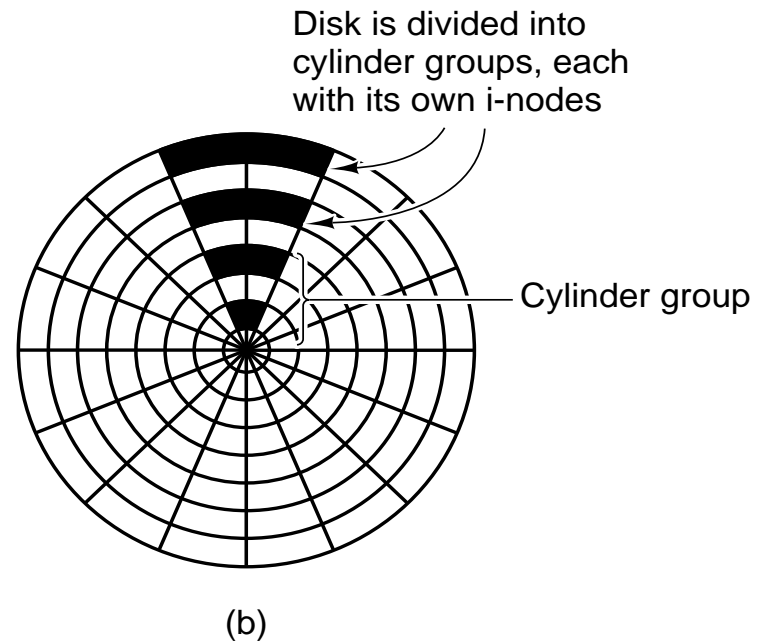
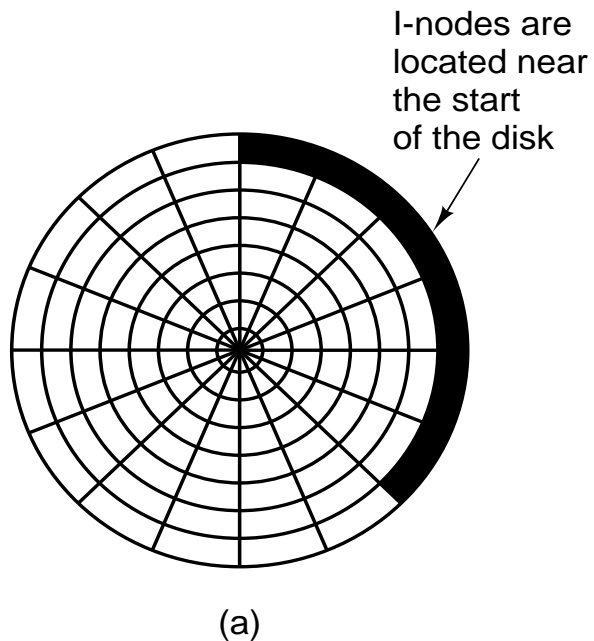
Wydajność systemu plikowego (1)

- pamięć podręczna bloków dyskowych
 - organizacja: lista uporządkowana wg. częstotliwości użycia



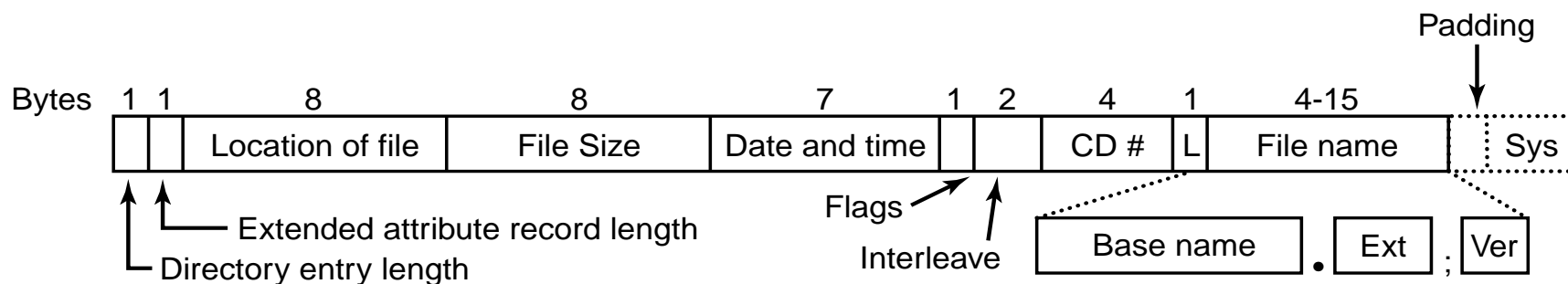
Wydajność systemu plikowego (2)

- umiejscowienie bloków indeksowych
 - (a) – na początku dysku
 - (b) – w odpowiedniej grupie cylindrów



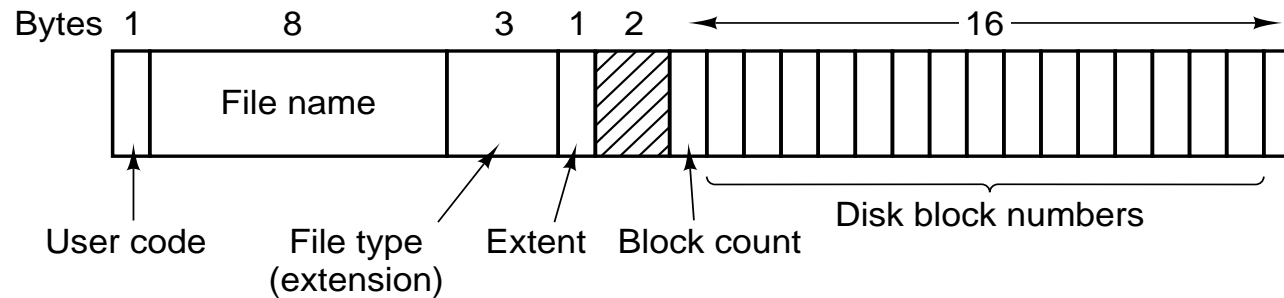
Przykład: CD-ROM FS

- element katalogu wg. ISO 9660



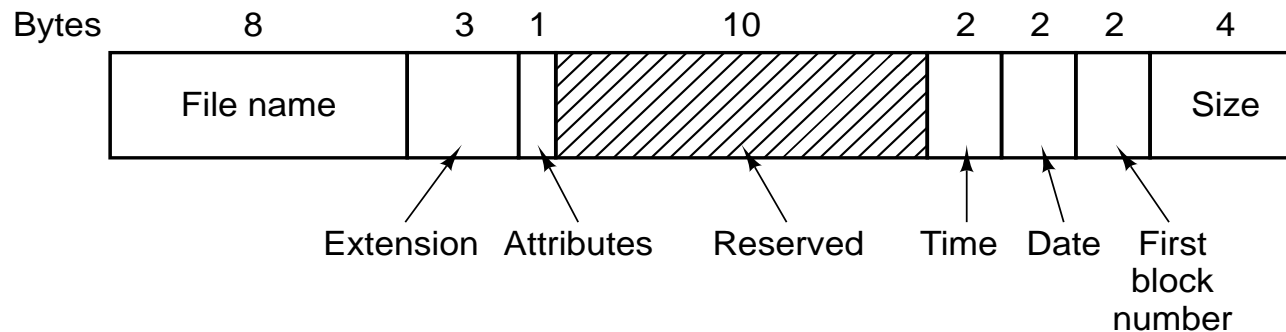
Przykład: CP/M FS

- element katalogu głównego



Przykład: MSDOS FS (1)

- element katalogu (FAT12/16)



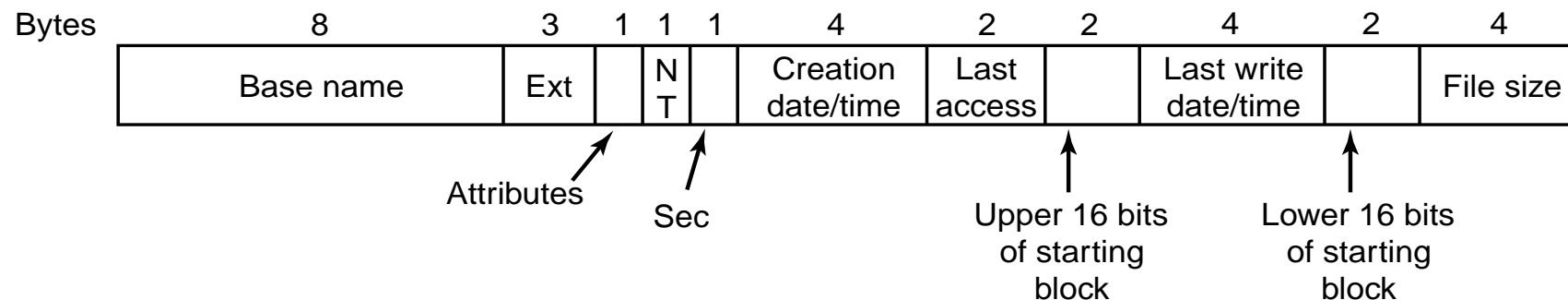
Przykład: MSDOS FS (2)

- maksymalna wielkość wolumenu dyskowego

Block size	FAT-12	FAT-16	FAT-32
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

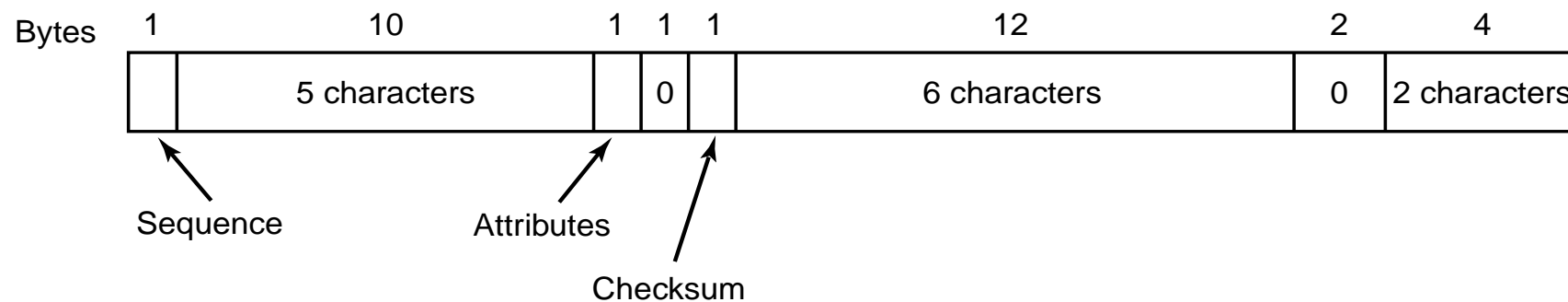
Przykład: MSDOS FS (3)

□ FAT32/VFAT – Windows 98 (Win95 SP2)



Przykład: MSDOS FS (4)

- element "długiej nazwy" pliku (Win98/Win95SP2)



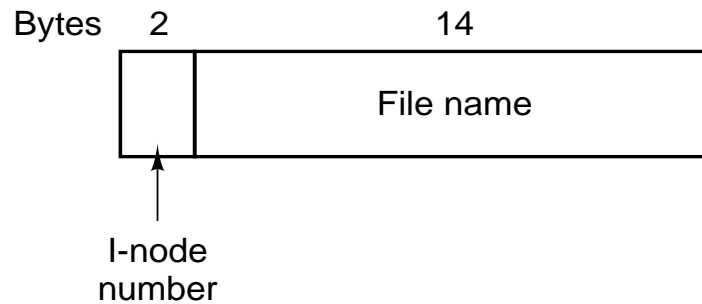
Przykład: MSDOS FS (5)

- kompletna pozycja katalogowa (Win98/Win95SP2)

Bytes	68	d o g										A	0	C	K									0						
	3	o v e										A	0	C	K	t h e				l a				0	z y					
	2	w n f o										A	0	C	K	x j u m p				0	s									
	1	T h e q										A	0	C	K	u i c k				b				0	r o					
	T	H E Q U I ~ 1										A	N	S	Creation time			Last acc		Upp		Last write			Low		Size			

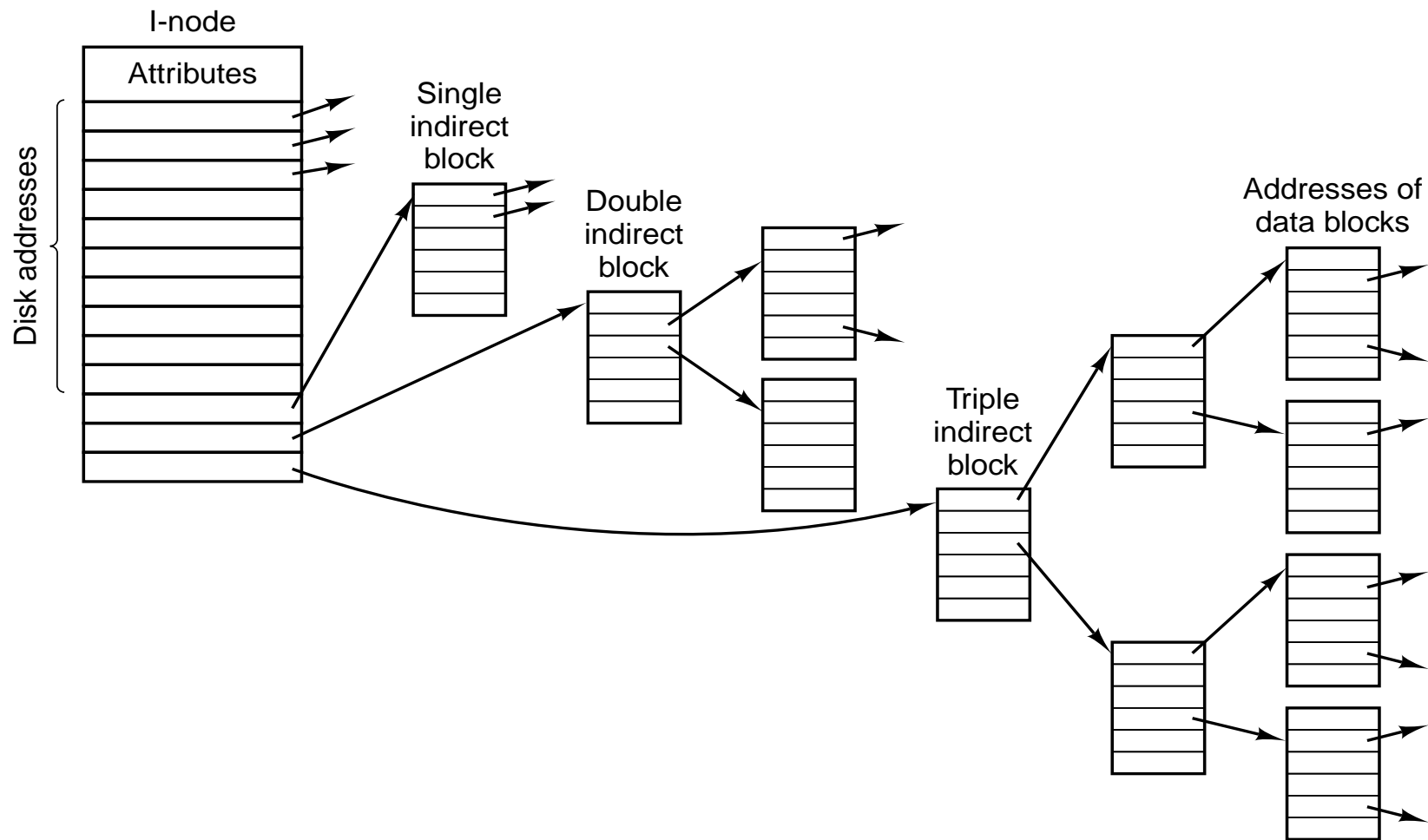
Przykład: UNIX v7 FS (1)

- pozycja katalogowa



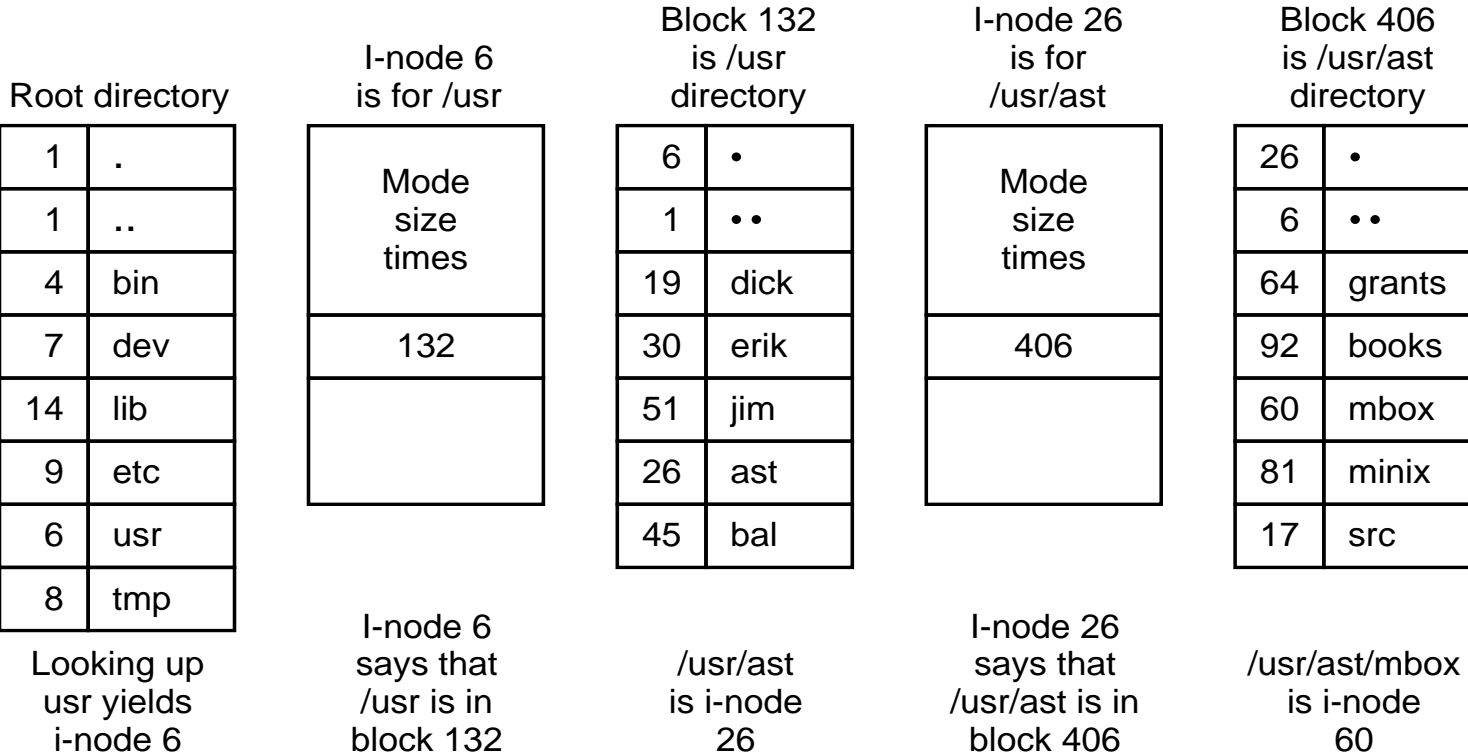
Przykład: UNIX v7 FS (2)

□ bloki indeksowe



Przykład: UNIX v7 FS (3)

□ lokalizacja pliku /usr/ast/mbox



===

07 – Multimedia i ich przetwarzanie

- ☐ Wstęp
- ☐ Pliki i dane multimedialne
- ☐ Kompresja/dekompresja danych multimedialnych
- ☐ Zagadnienia związane z systemowym szeregowaniem zadań i obsługą pamięci podręcznej

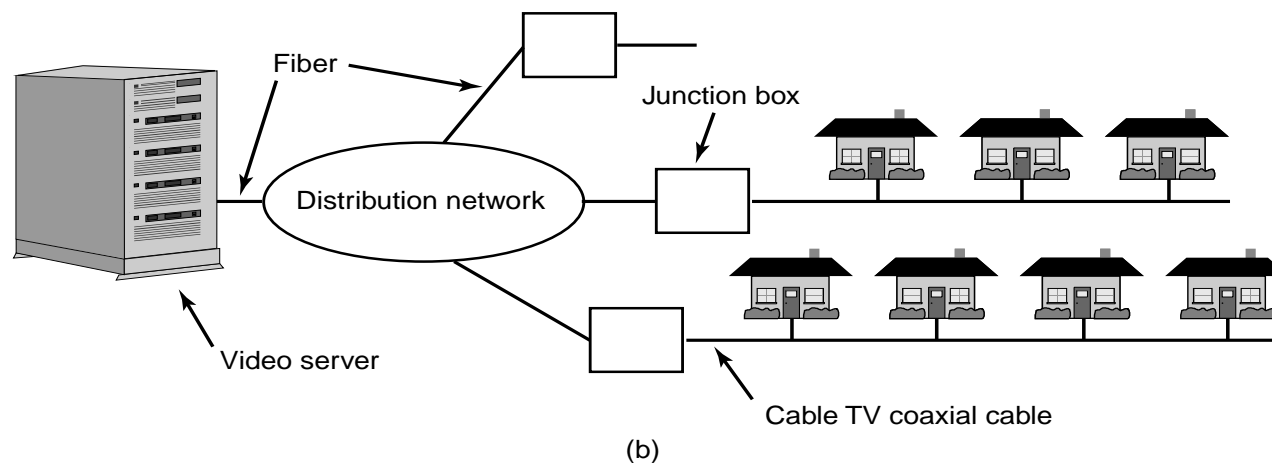
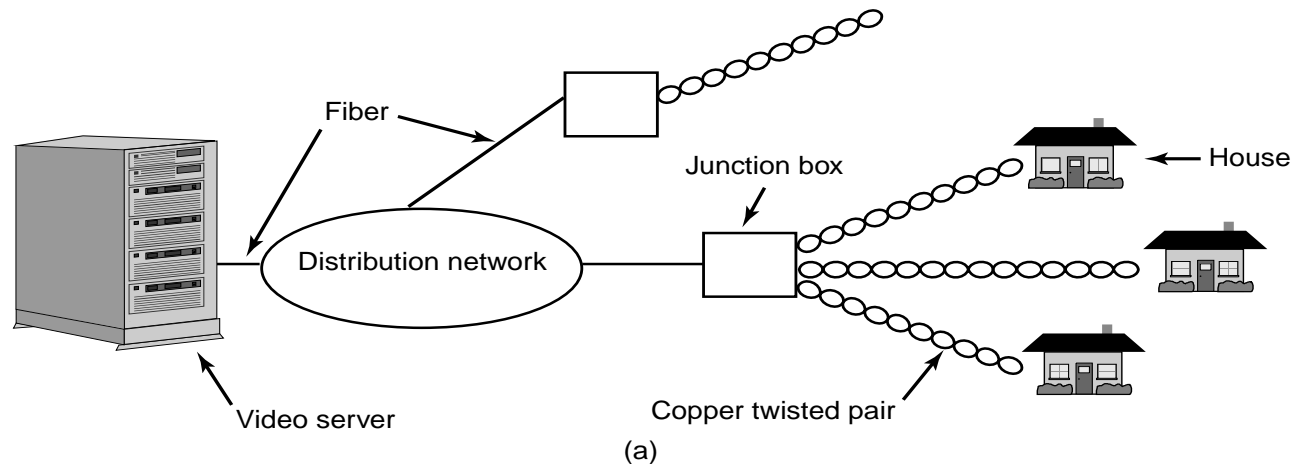
>>>

Typowe prędkości transmisji danych

Source	Mbps	GB/hr
Telephone (PCM)	0.064	0.03
MP3 music	0.14	0.06
Audio CD	1.4	0.62
MPEG-2 movie (640 \times 480)	4	1.76
Digital camcorder (720 \times 480)	25	11
Uncompressed TV (640 \times 480)	221	97
Uncompressed HDTV (1280 \times 720)	648	288
Fast Ethernet	100	
EIDE disk	133	
ATM OC-3 network	156	
SCSI UltraWide disk	320	
IEEE 1394 (FireWire)	400	
Gigabit Ethernet	1000	
SCSI Ultra-160 disk	1280	

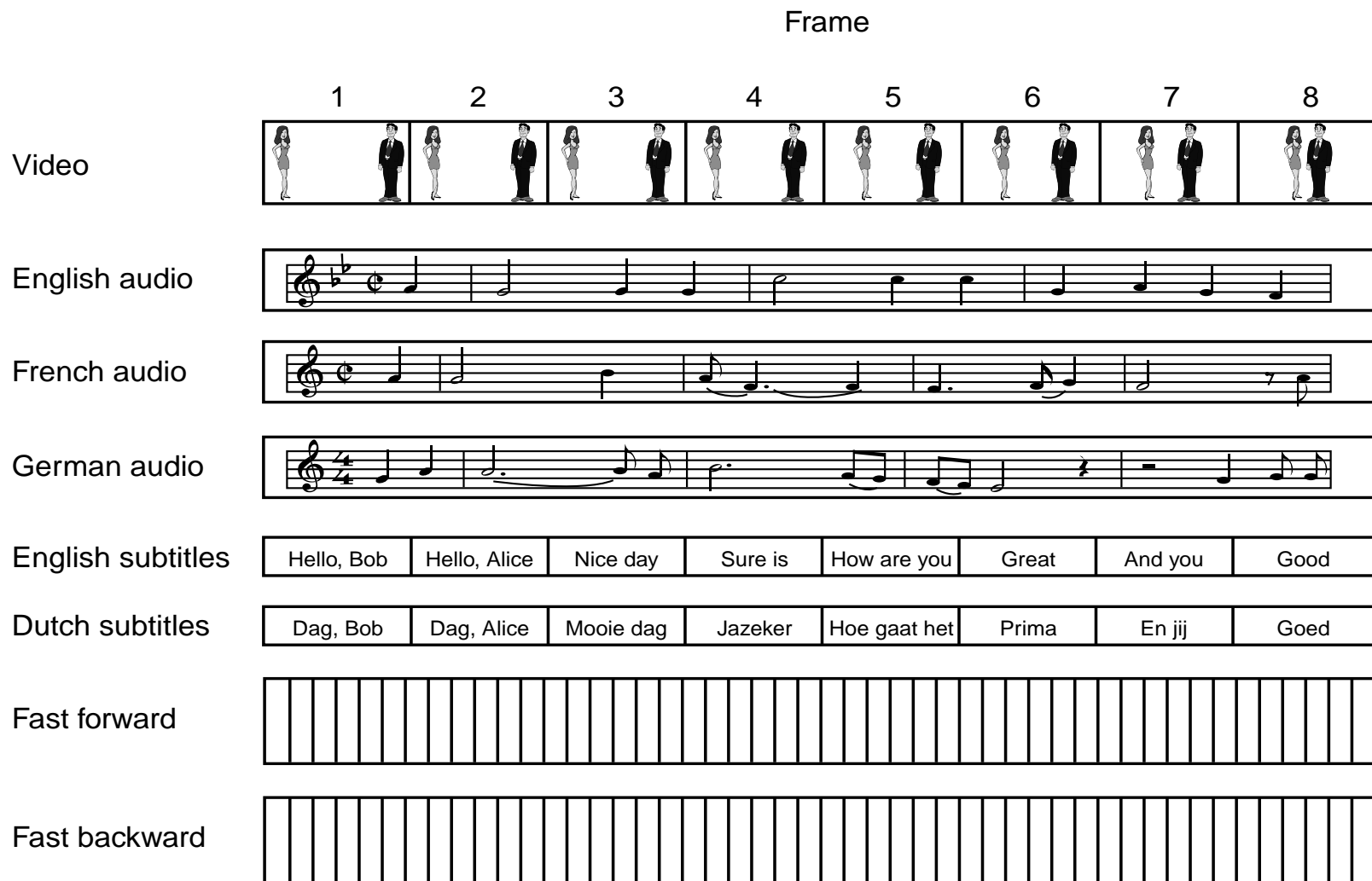
Infrastruktura

- przykład: realizacja "Video On Demand" (VoD)
 - (a) – ADSL (Asymmetric Digital Subscriber Line)
 - (b) – CaTV (Cable TV)



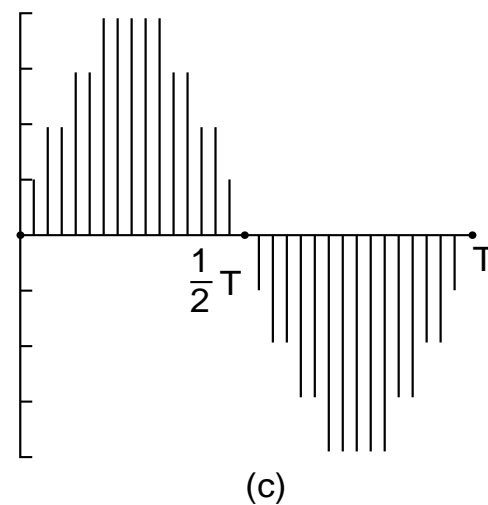
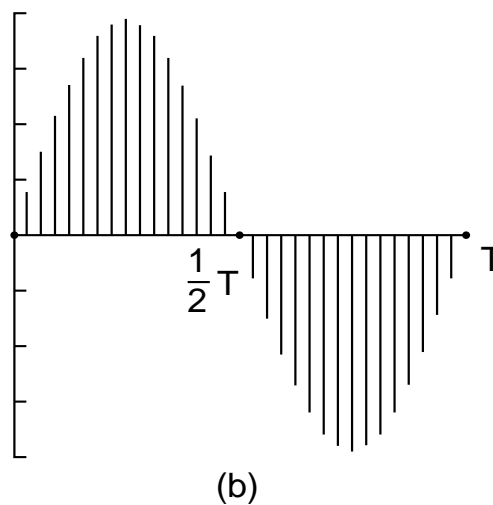
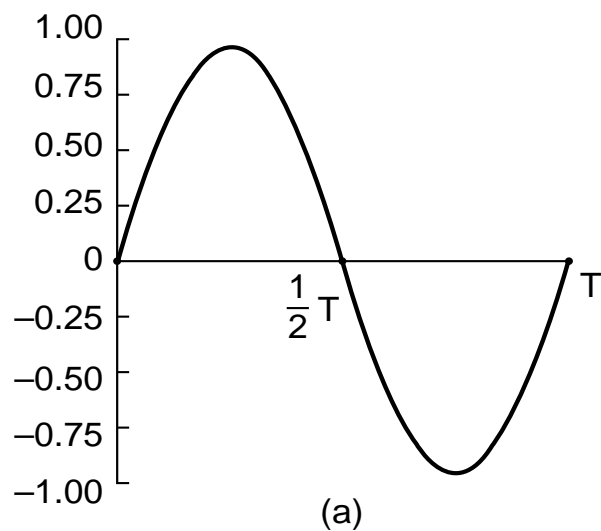
Multimedia a pliki

- przykład: typowy film



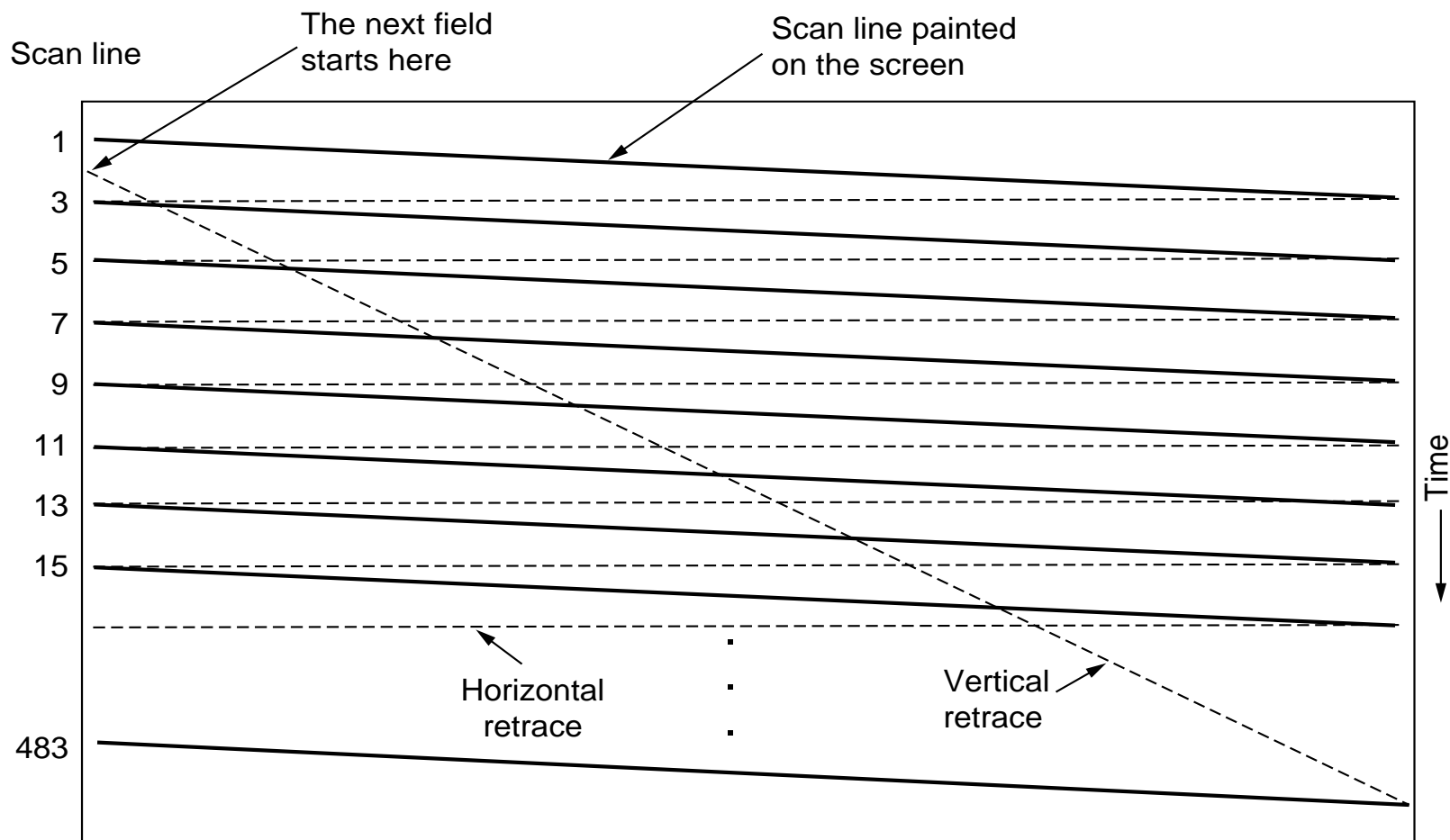
Dźwięk

- konwersja do postaci cyfrowej
 - (a) – amplituda sygnału analogowego
 - (b) – próbkowanie sygnału
 - (c) – próbkowanie i kwantyzacja



Obraz

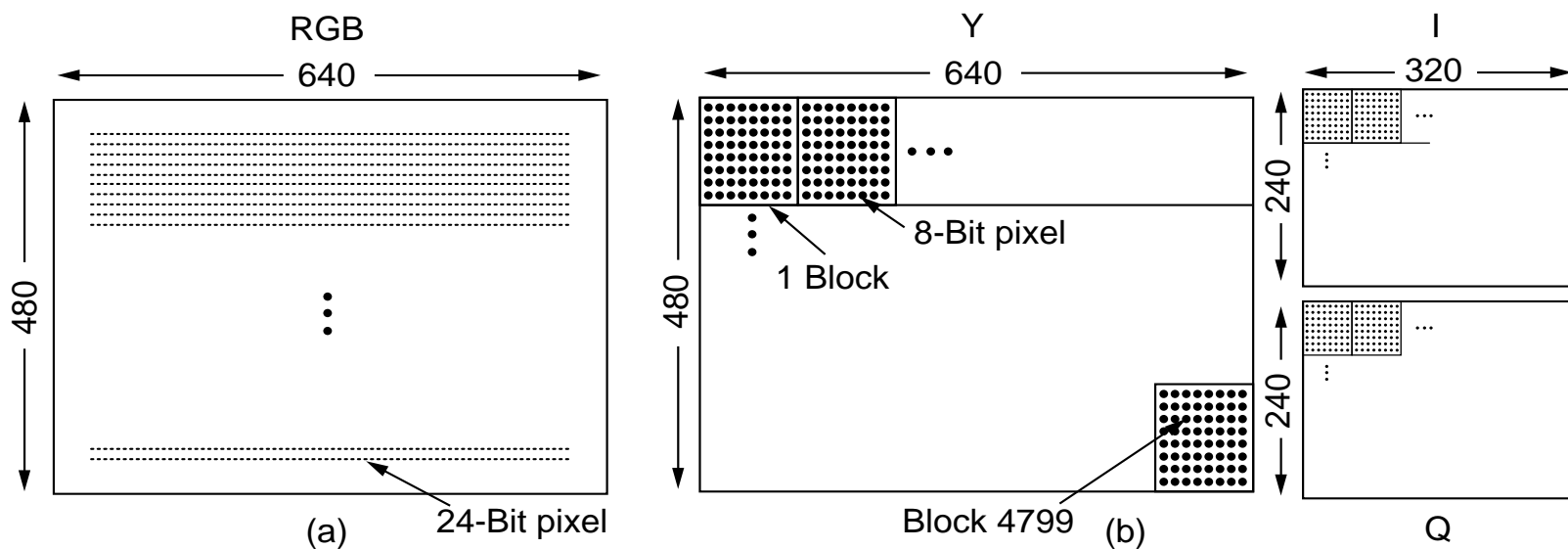
□ skanowanie obrazu w standardzie NTSC



Kompresja obrazu (1)

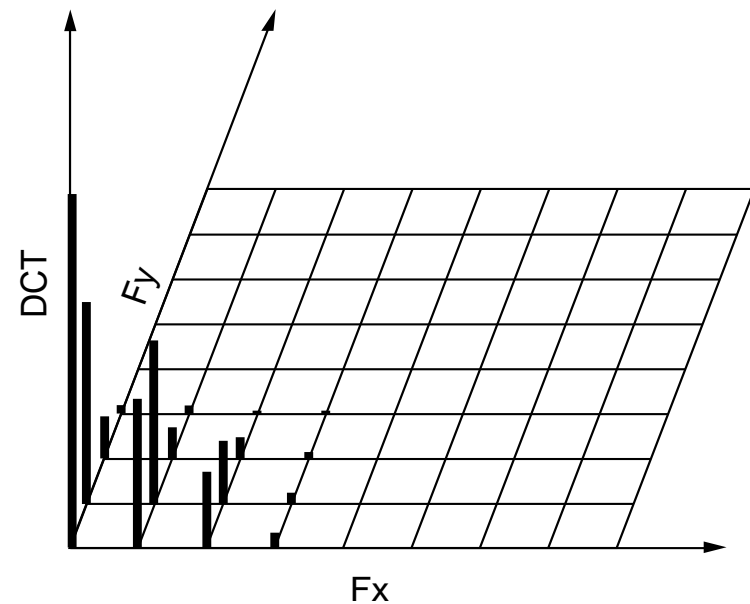
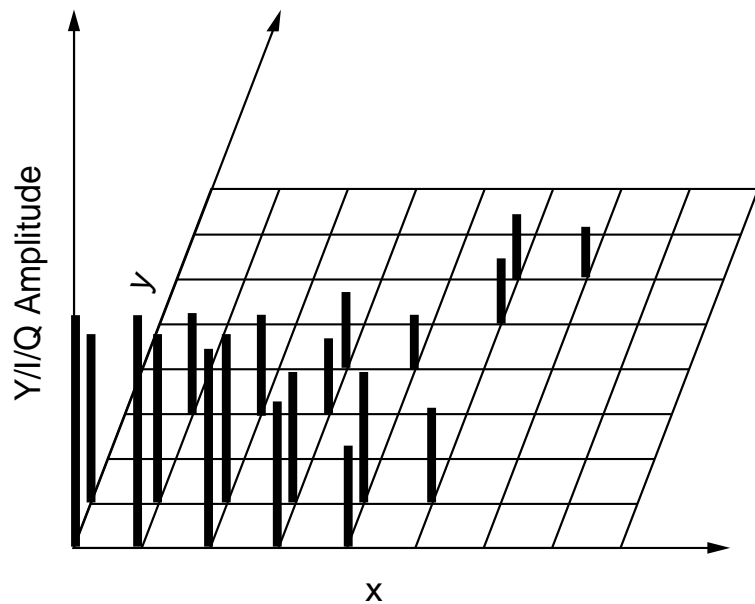
□ przykład: JPEG/MPEG

- (a) – dane wejściowe (RGB)
- (b) – struktura blokowa, macierze Y, I, Q



Kompresja obrazu (2)

- transformacja DCT
 - Discrete Cosine Transform
 - przykładowy blok macierzy Y



Kompresja obrazu (3)

□ kwantyzacja DCT/Q

DCT Coefficients

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantized coefficients

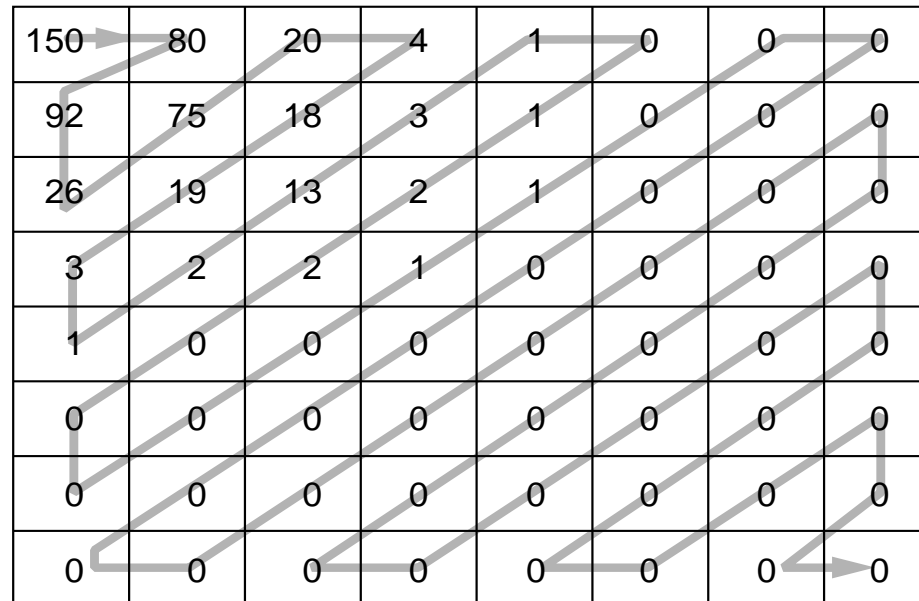
150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantization table

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

Kompresja obrazu (4)

- ruchomy obraz: MPEG
 - określenie kolejności transmisji po kwantyzacji



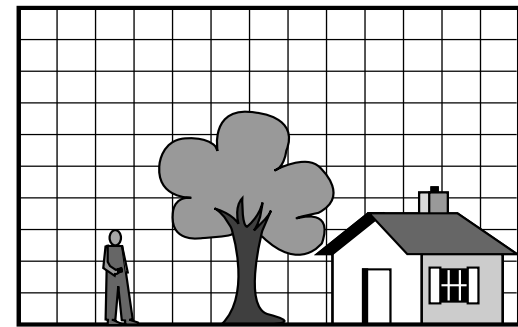
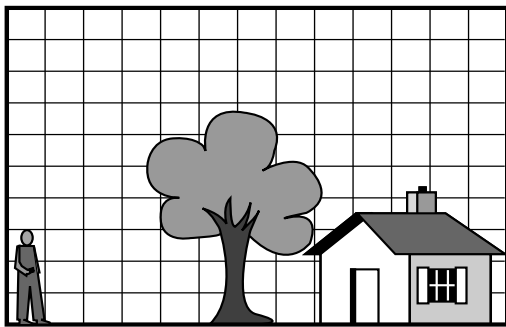
150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Kompresja obrazu (5)

□ MPEG-2:

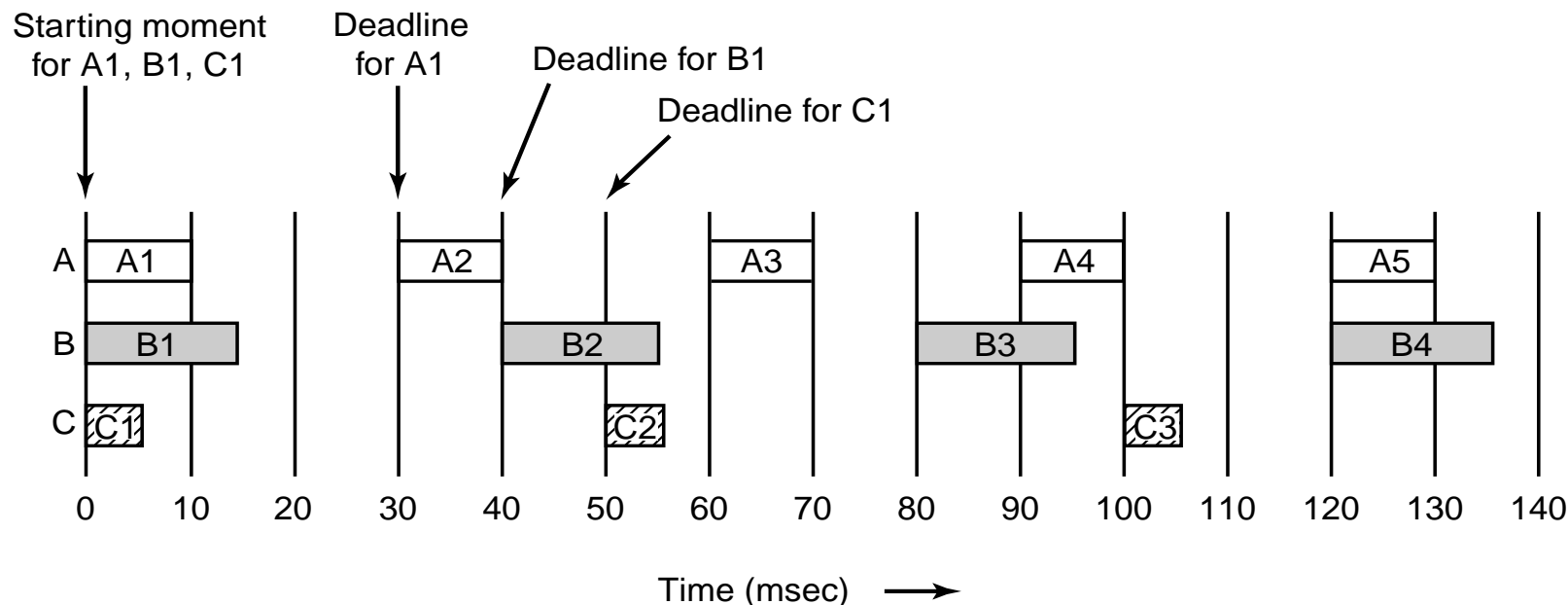
- I – Intracoded frames – pełne obrazy JPEG
- P – Predictive frames – różnice względem poprzedniej ramki
- B – Bi-directional frames – różnice wzgl. poprzedniej i następnej ramki

□ przykładowe następstwo ramek (I)



Multimedia a szeregowanie zadań (1)

- przykład: trzy procesy "multimedialne" A, B, C
 - ustalone czasy graniczne (deadline)
 - ustalone parametry pracy (prędkość wyświetlania ramek, etc.)



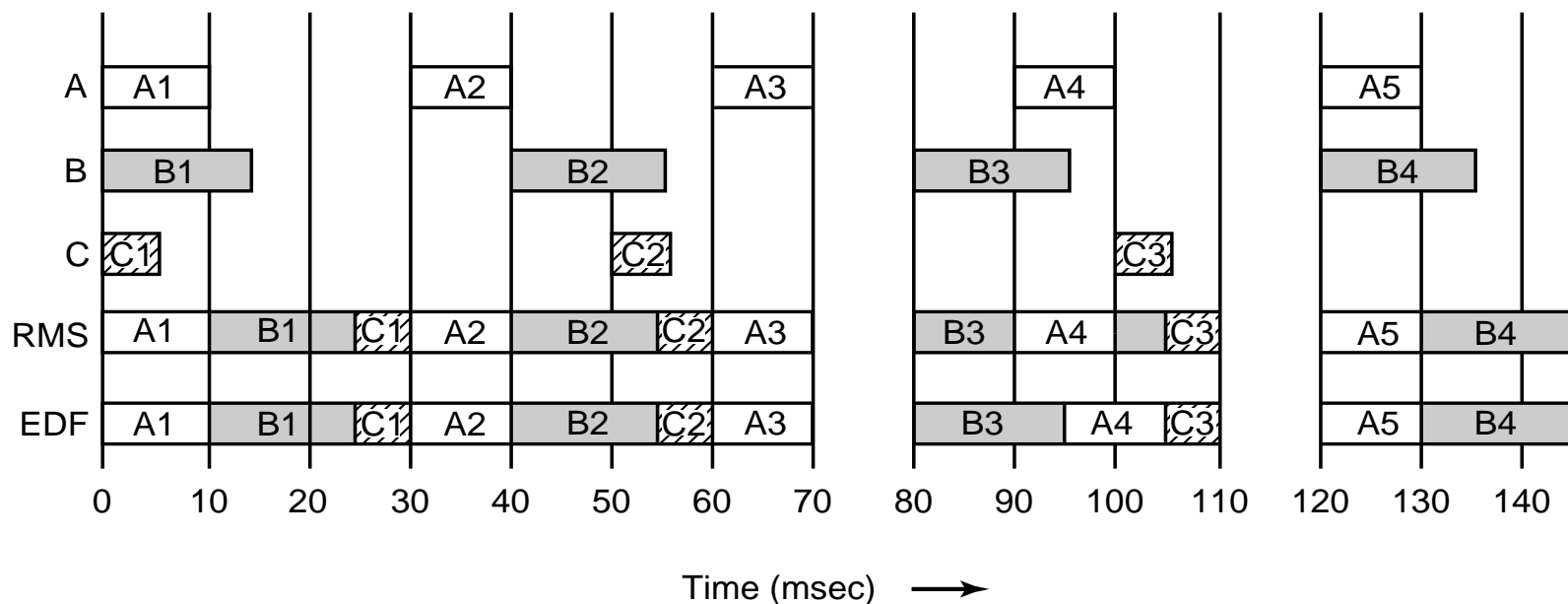
- Rate Monotonic Scheduling (RMS)
 - stałe długości okresów pracy i przestojów
 - stałe wymagania odnośnie zasobów
 - możliwość natychmiastowego wyłączenia

Multimedia a szeregowanie zadań (2)

□ Earliest Deadline First (EDF)

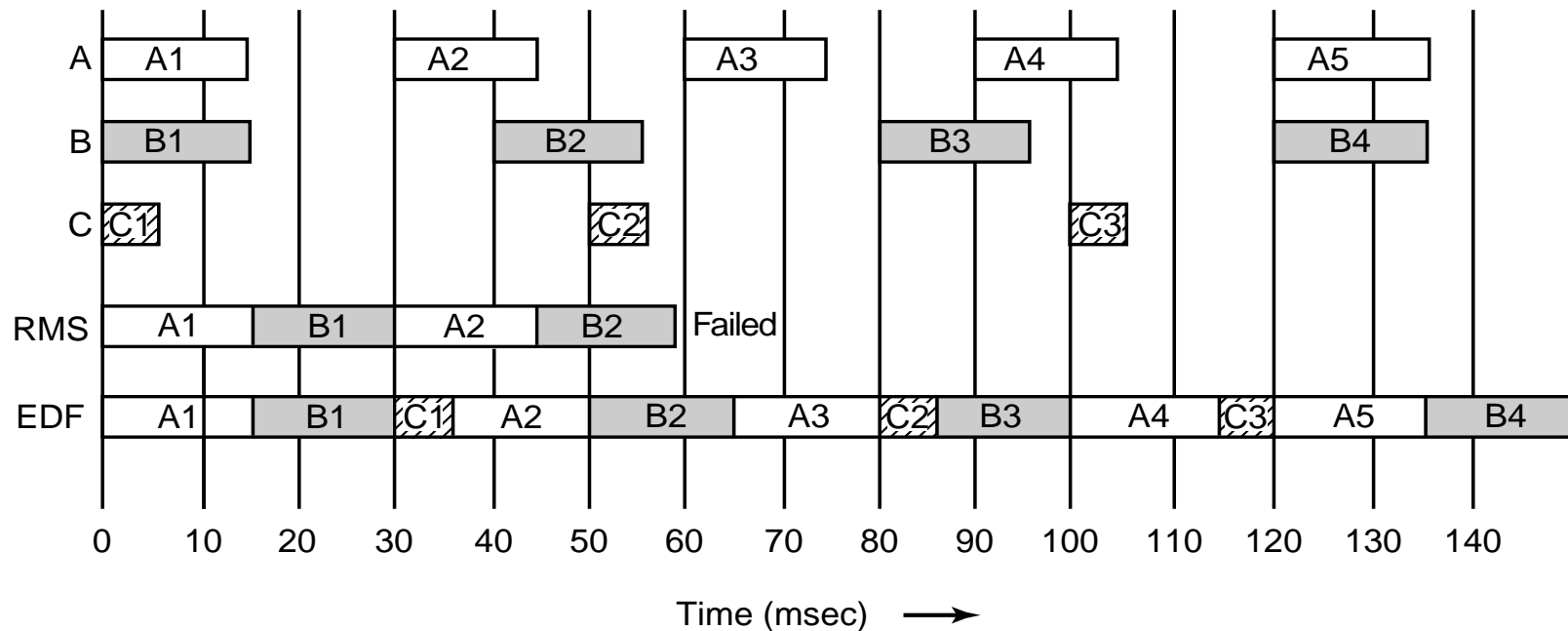
○ umożliwia dopasowanie do zmian reżimu czasowego

□ porównanie działania RMS i EDF



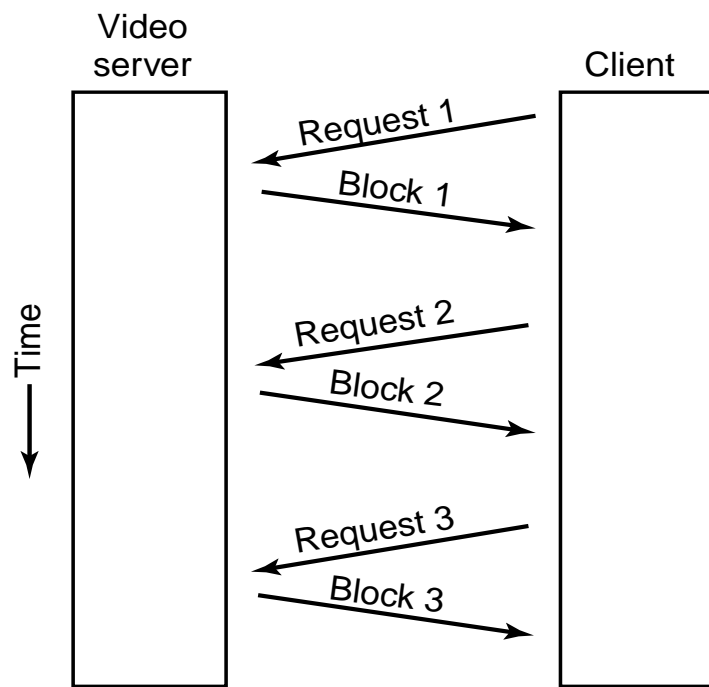
Multimedia a szeregowanie zadań (3)

- inne porównanie RMS z EDF
 - zastosowanie RMS prowadzi do załamania wykonania (C)

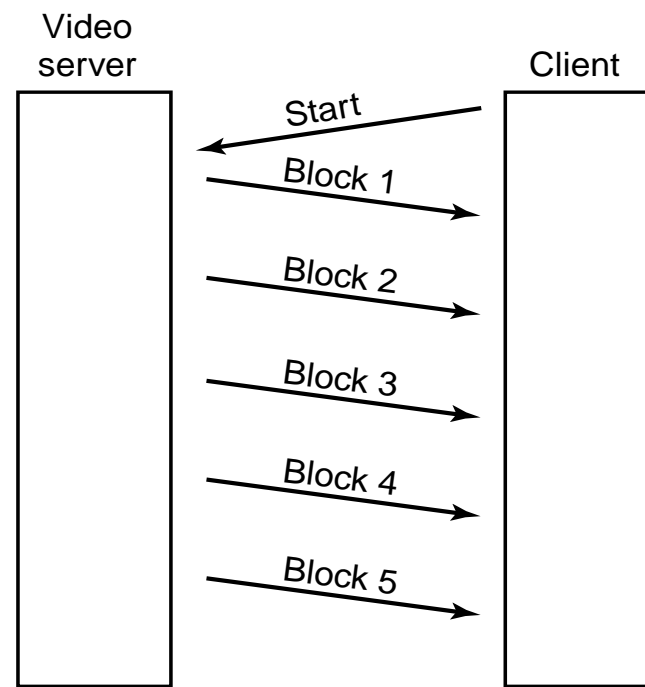


Multimedialne systemy plikowe (1)

- dostosowane do rodzaju transmisji
 - (a) – "Pull" – transmisja bloku na żądanie
 - (b) – "Push" – transmisja sekwencji bloków (strumienia)



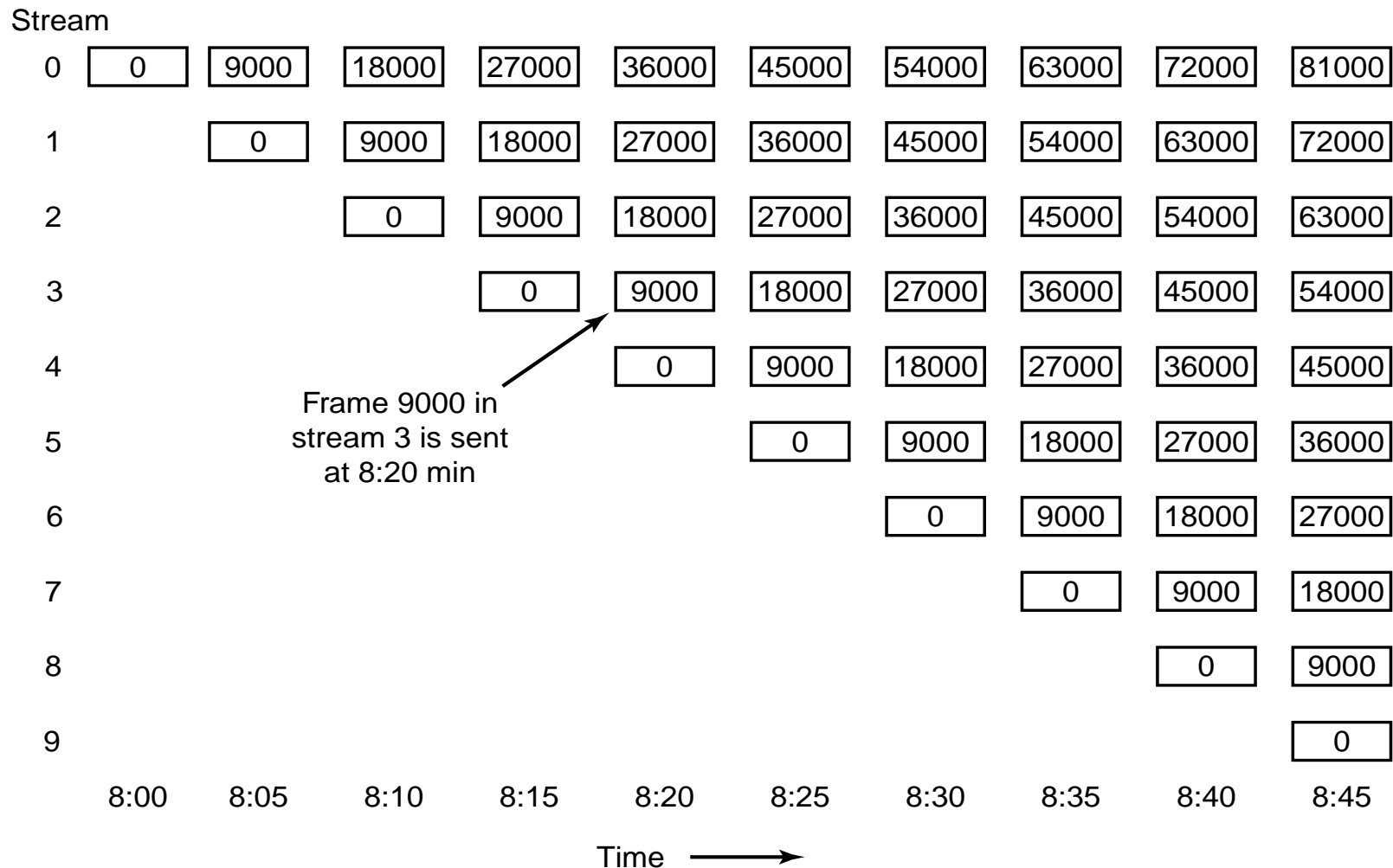
(a)



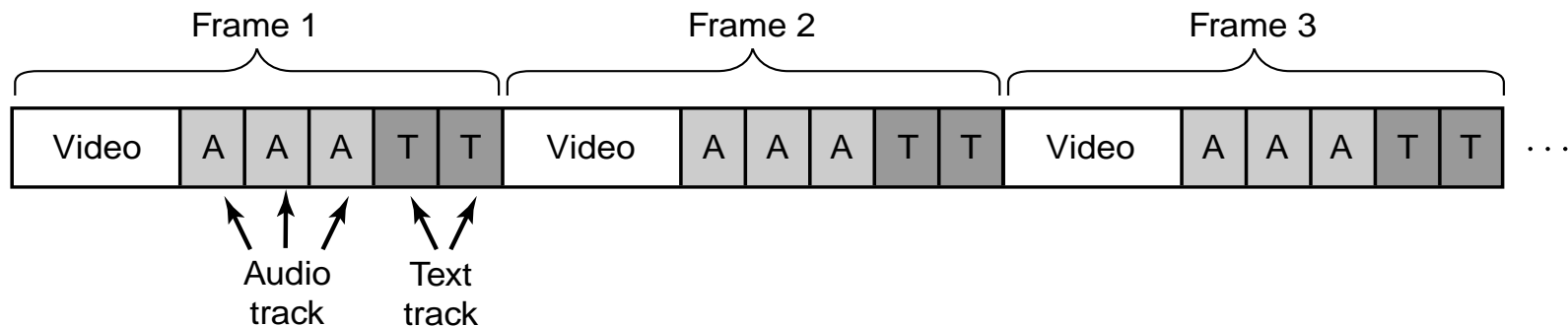
(b)

Multimedialne systemy plikowe (2)

- przeplatanie transmisji kilku strumieni (VoD)

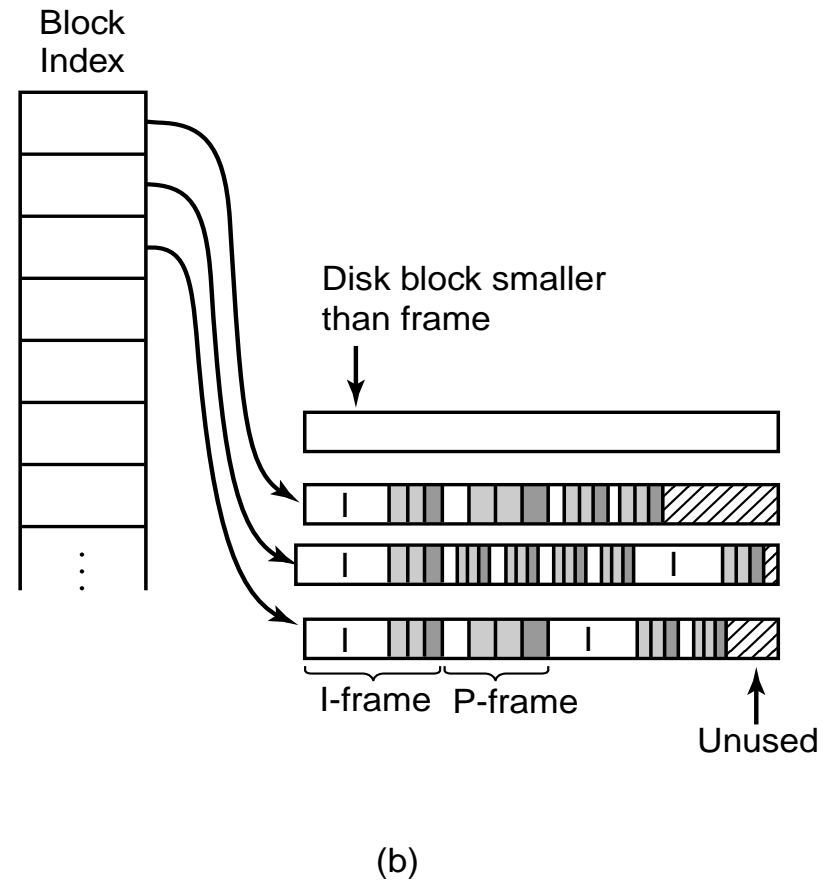
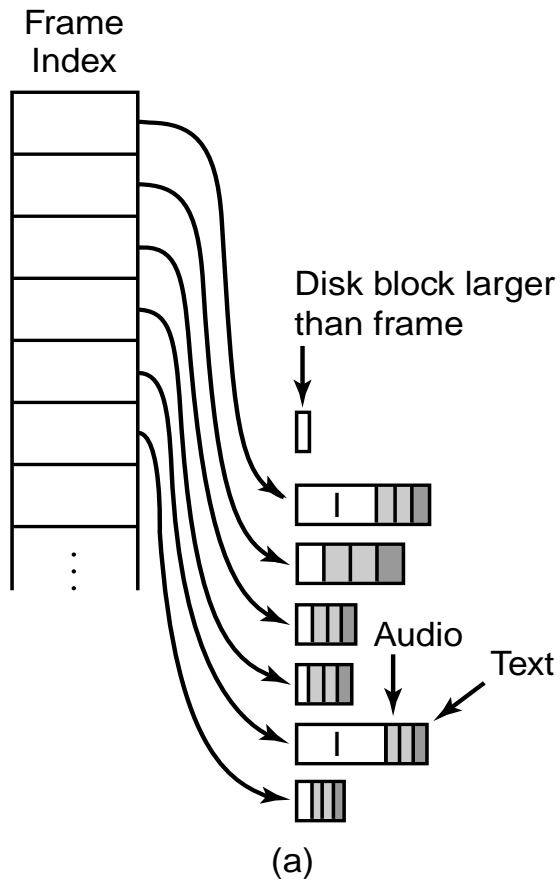


- plik jako sekwencja kompletnych ramek
 - przeplot ramek Video, Audio, Text



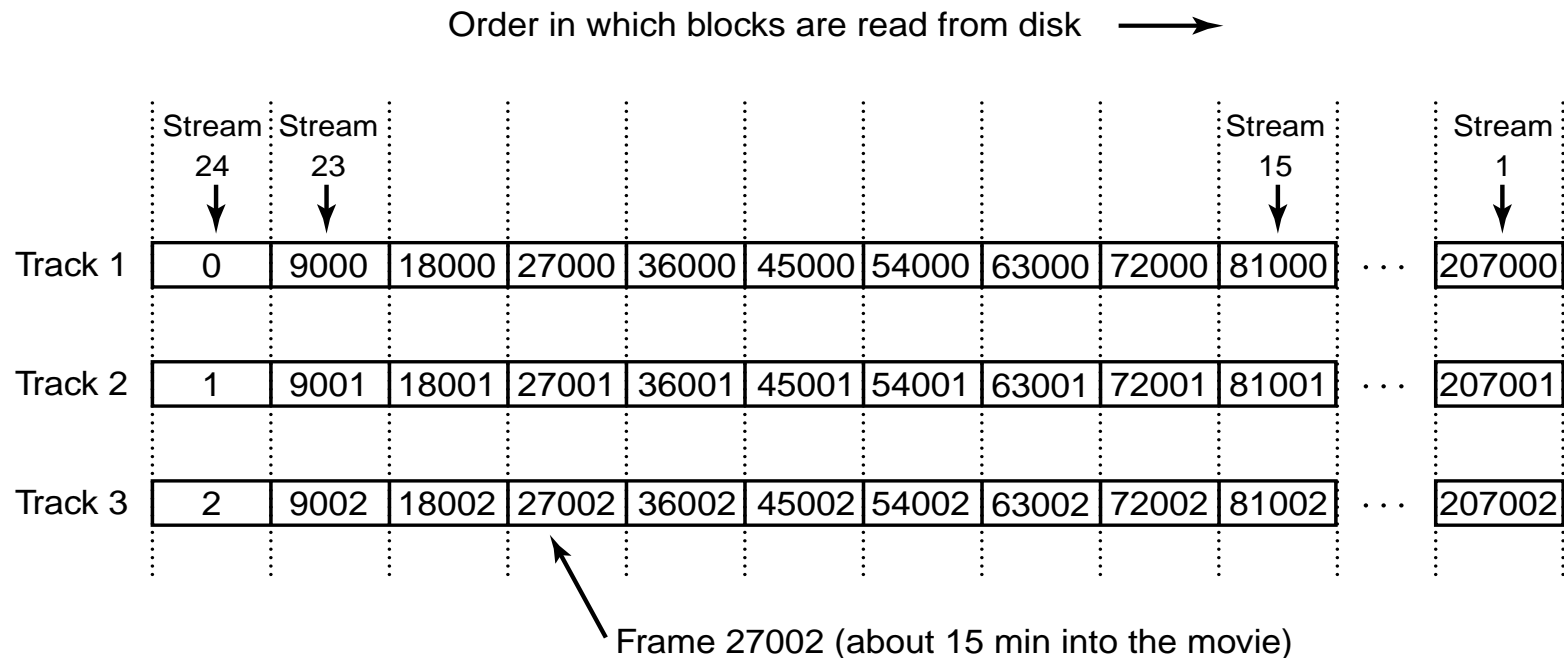
Multimedialne systemy plikowe (4)

- indeksowanie ramek i bloków dyskowych



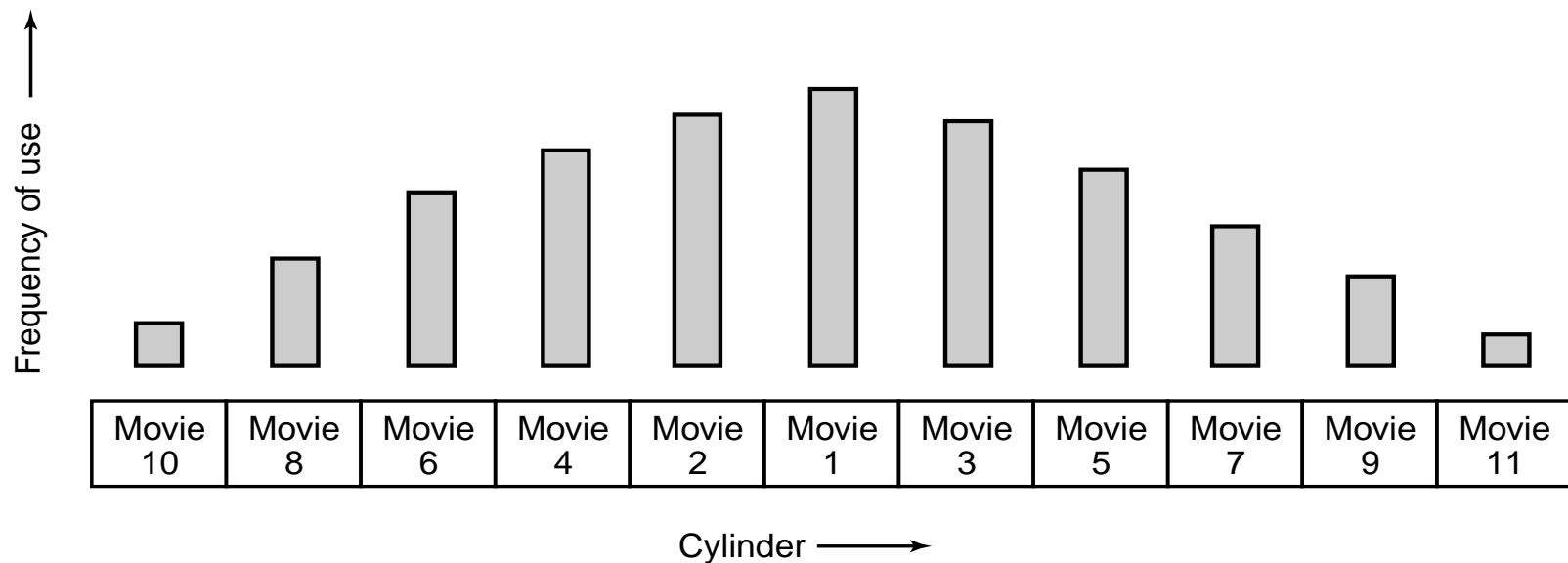
Multimedialne systemy plikowe (5)

- optymalne fizyczne rozmieszczenie ramek na dysku



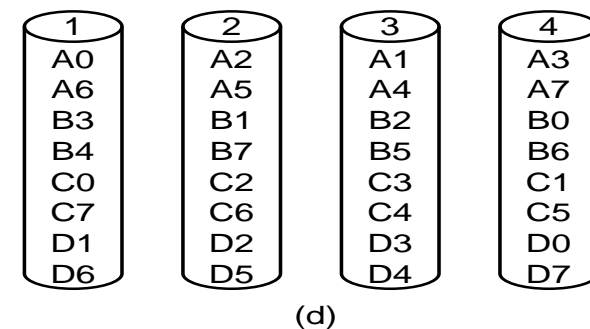
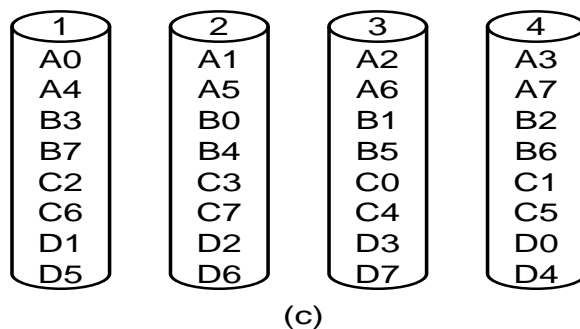
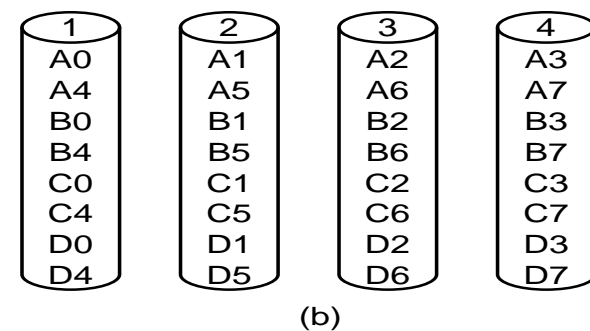
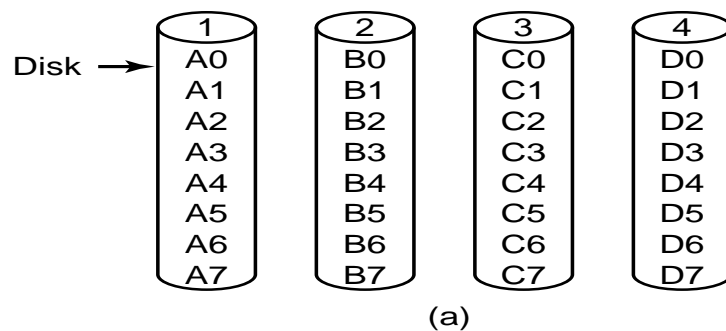
Multimedialne systemy plikowe (6)

- optymalne fizyczne rozmieszczenie plików na dysku
 - najczęściej wykorzystywane pliki pośrodku



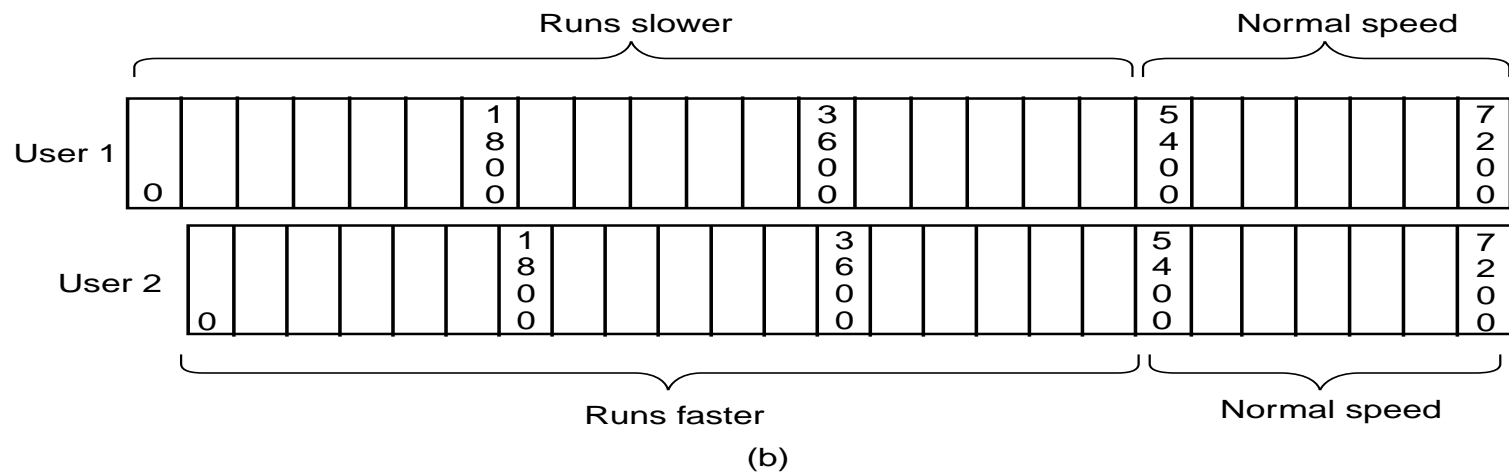
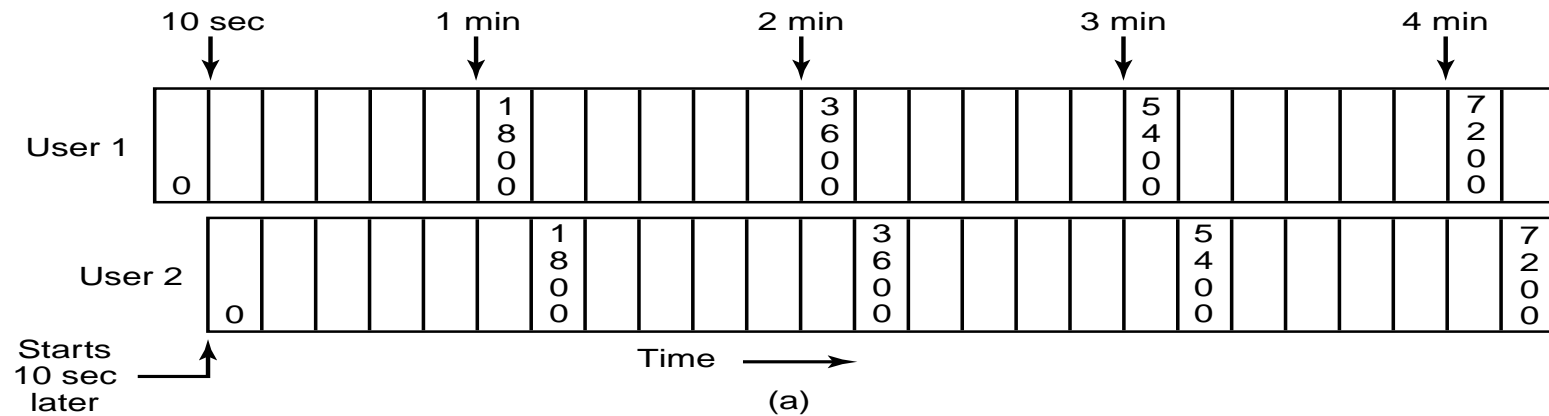
Multimedialne systemy plikowe (7)

- rozmieszczenie na kilku fizycznych dyskach
 - (a) – pliki na osobnych dyskach
 - (b)–(d) – różne sposoby podziału pomiędzy dyski



Buforowanie i pamięć podręczna (1)

- konwergencja strumieni



Buforowanie i pamięć podręczna (2)

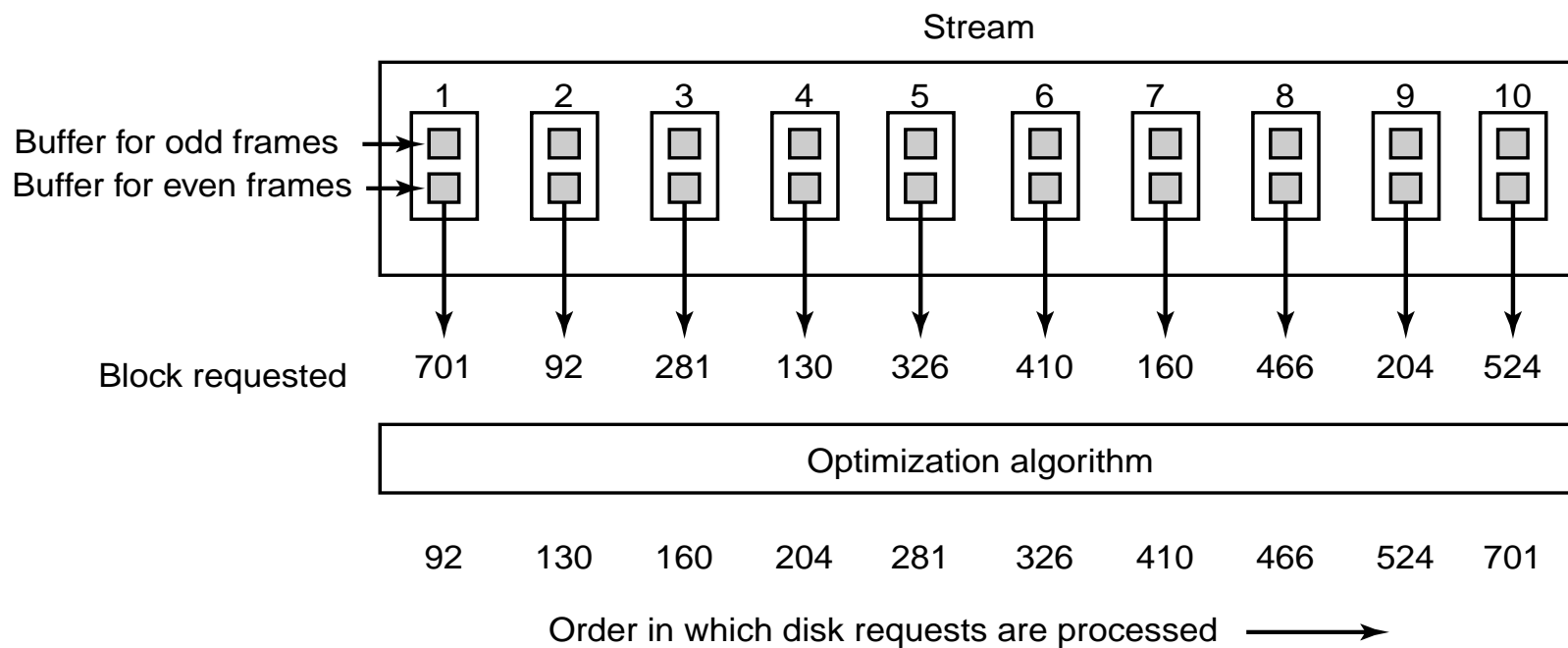
- wielopoziomowa obsługa archiwum
 - podstawowe media: taśmy, DVD, etc.
 - pamięć podręczna: HDD (RAID)
 - dodatkowo: początkowe sekwencje do natychmiastowego odtworzenia

- obsługa typowych funkcji VCR
 - dwukierunkowe "szybkie przewijanie"
 - odtwarzanie w zwolnionym tempie
 - zatrzymanie obrazu i powtórka

Dostęp do dysku (1)

□ Static Disk Scheduling

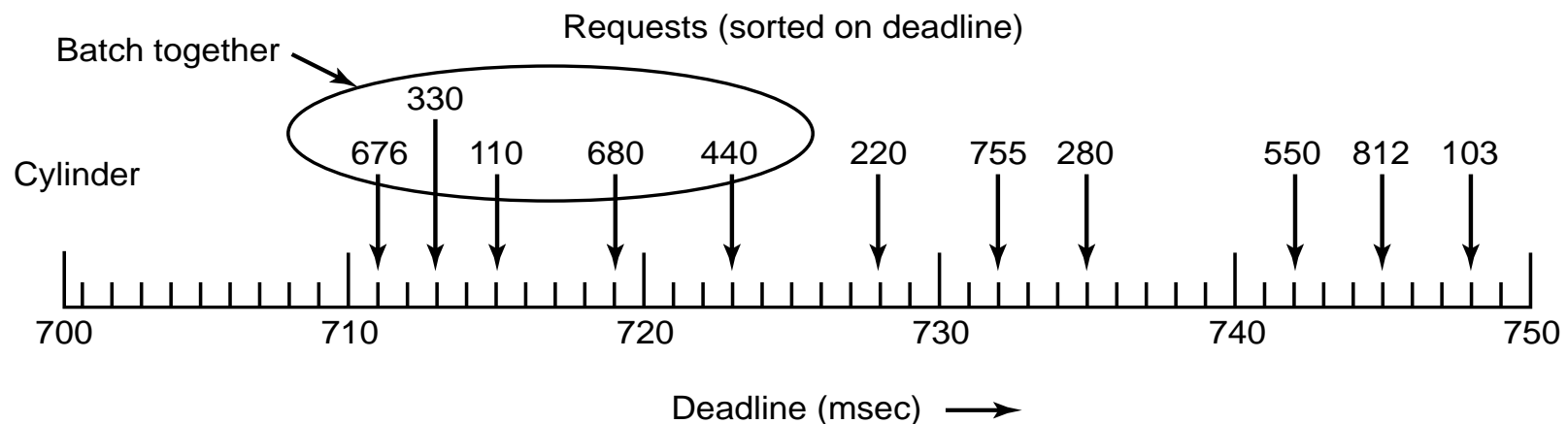
- każdy proces zamiawia tylko pojedyncze ramki/bloki
- zamówienia są poddawane optymalizacji
- realizacja zamówień odbywa się cyklicznie



Dostęp do dysku (2)

□ Dynamic Disk Scheduling

- zamówienia są kolejgowane wg. czasu granicznego (deadline)
- możliwe jest grupowanie zamówień
- współpraca z algorytmem szeregowania zadań (EDF)
- przykład: algorytm SCAN-EDF



===

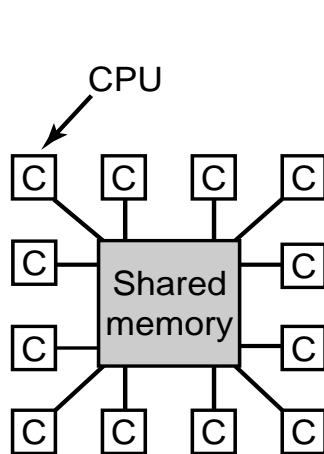
08 – Systemy wieloprocessorowe

- ☐ Podstawy
- ☐ Zagadnienia związane z obsługą sprzętu
- ☐ Systemowa obsługa wieloprocessorowości
- ☐ Multikomputery i systemy rozproszone

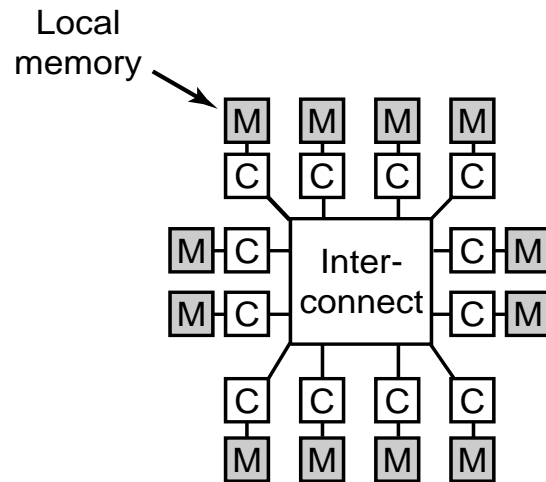
>>>

Podstawowe modele

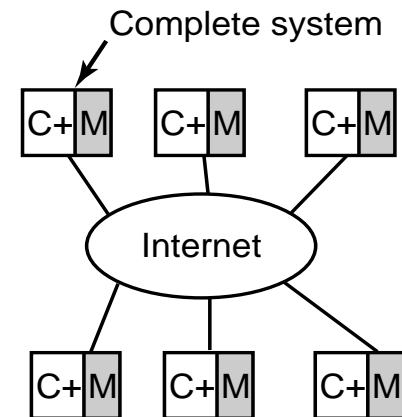
- klasyfikacja wg. sposobu komunikacji
 - (a) – współdzielona pamięć (lokalna)
 - (b) – lokalna sieć komputerowa
 - (c) – rozległy system rozproszony



(a)



(b)



(c)

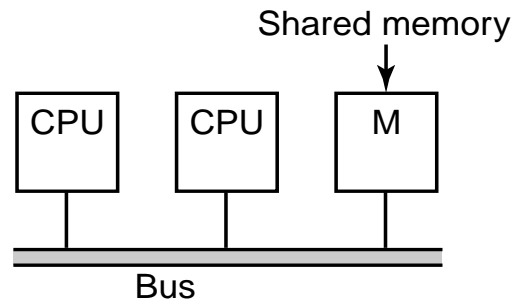
Wieloprocесory (1)

□ definicja

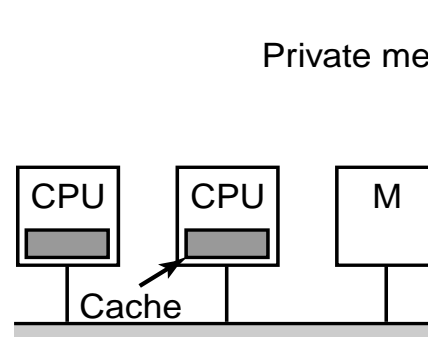
- system komputerowy w którym dwa lub więcej procesorów (CPU) posiada pełny dostęp do współdzielonej pamięci operacyjnej

□ wieloprocесory szynowe

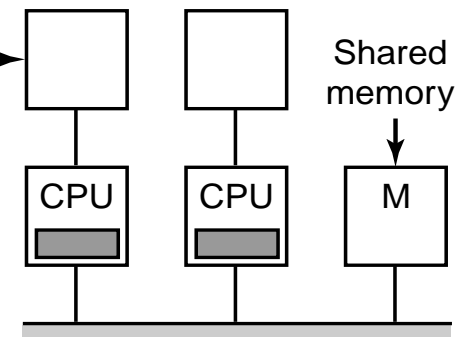
- (a) – kanoniczny
- (b) – z pamięcią podręczną procesora
- (c) – z dodatkową pamięcią własną ("prywatną")



(a)



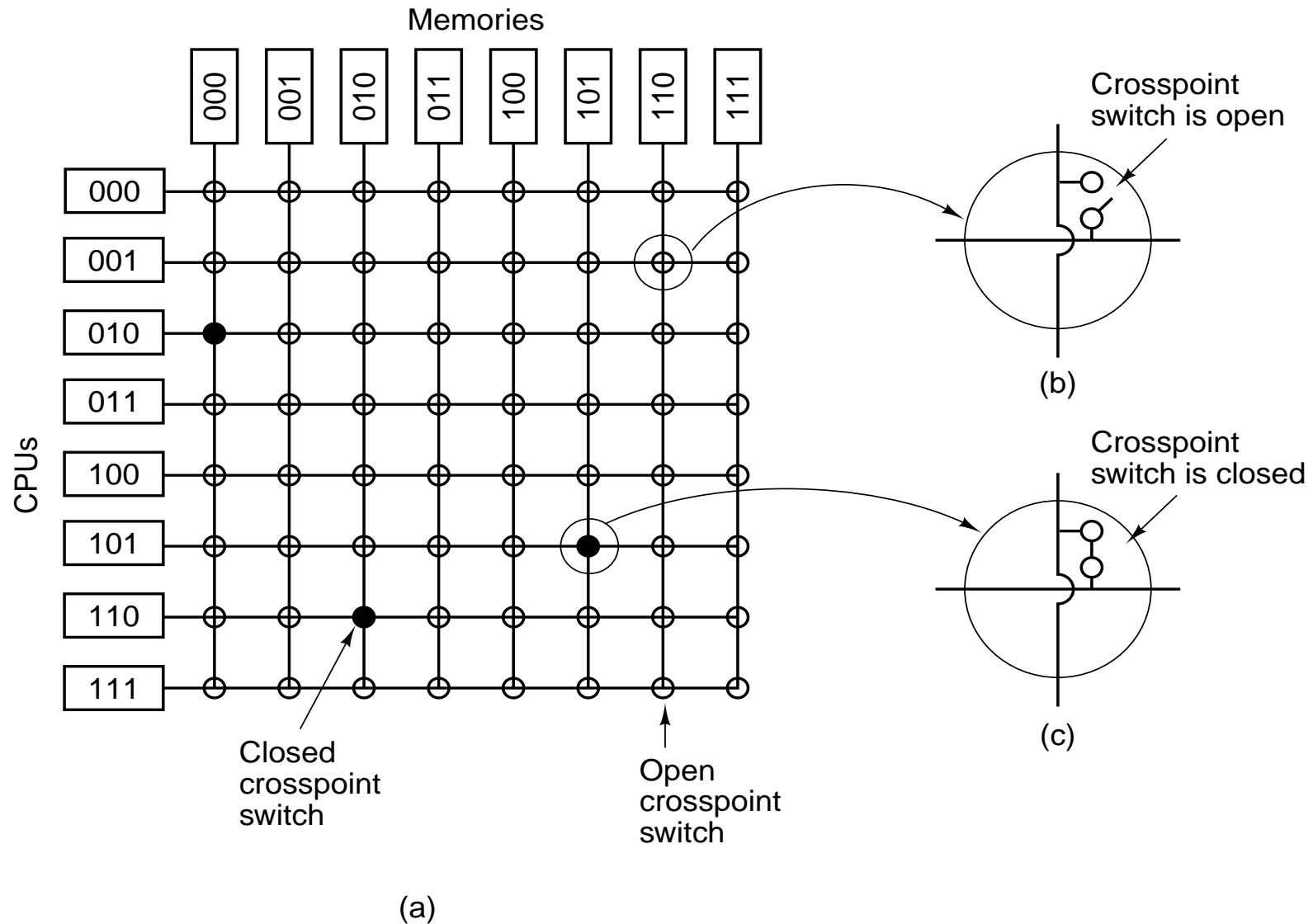
(b)



(c)

Wieloprocесory (2)

- z przełącznicą (crossbar switch)
 - Uniform Memory Access (UMA)

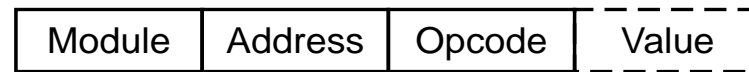


Wieloprocесory (3)

- z przełącznicą wielostopniową (multistage switch)
 - (a) – element sieci przełączającej (2x2 switch)
 - (b) – typowy format komunikatu w takiej sieci



(a)

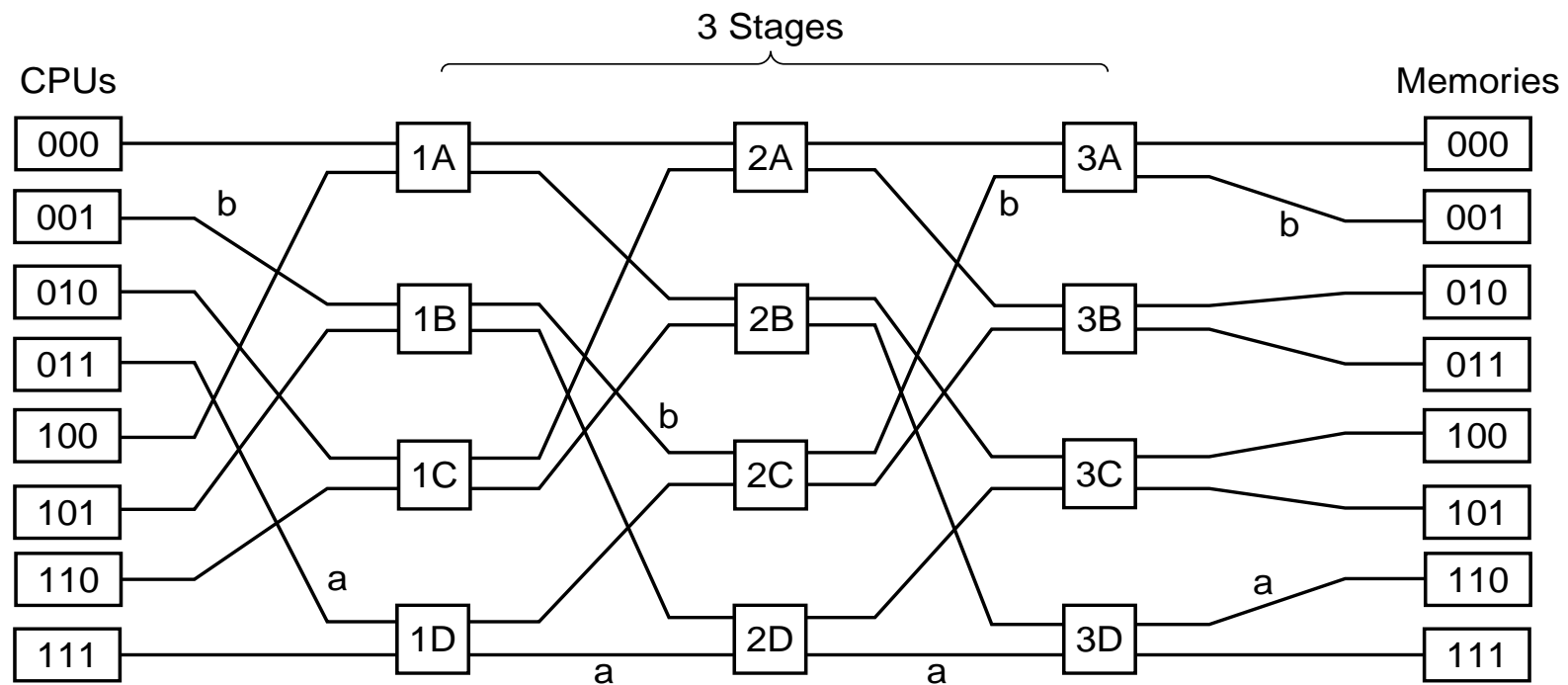


(b)

- różne sposoby połączenia w sieć
- mniejsza liczba punktów przełączania
- możliwe konflikty/blokady

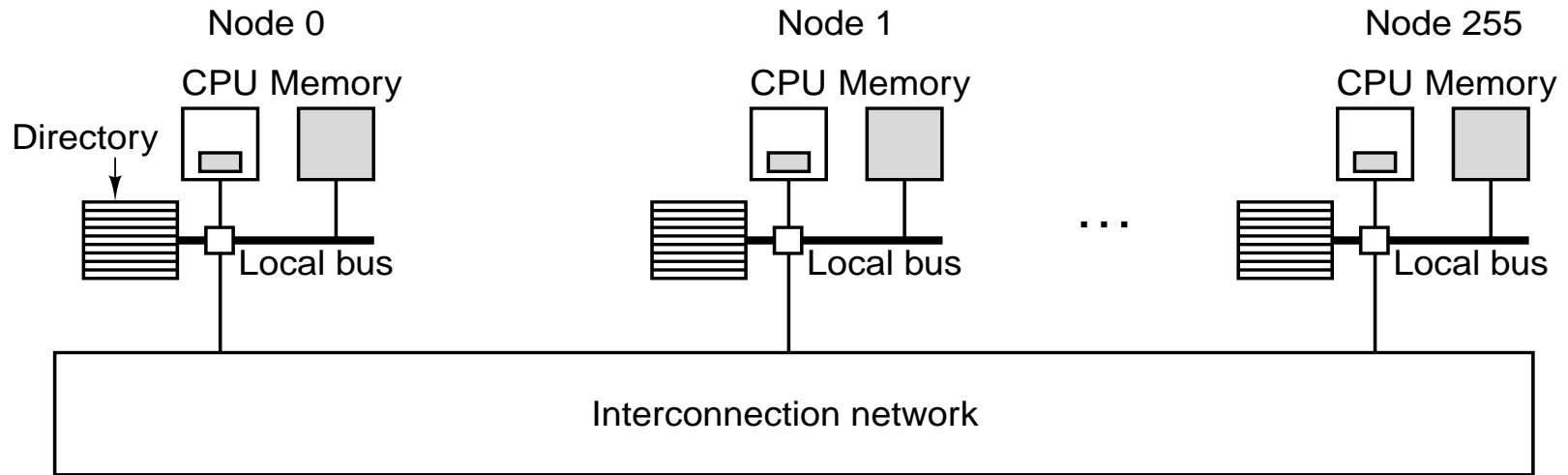
Wieloprocесory (4)

- z przełącznicą wielostopniową typu omega
 - bity adresu pamięci odp. przełączeniom na kolejnych etapach
 - a – trasa CPU(011)→Mem(110), b – trasa CPU(001)→Mem(001)

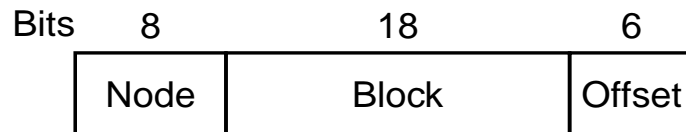


Wieloprocесory (5)

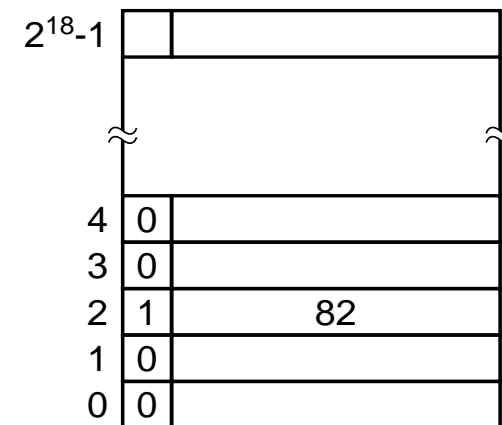
- z niejednorodnym dostępem do pamięci (NUMA)



(a)



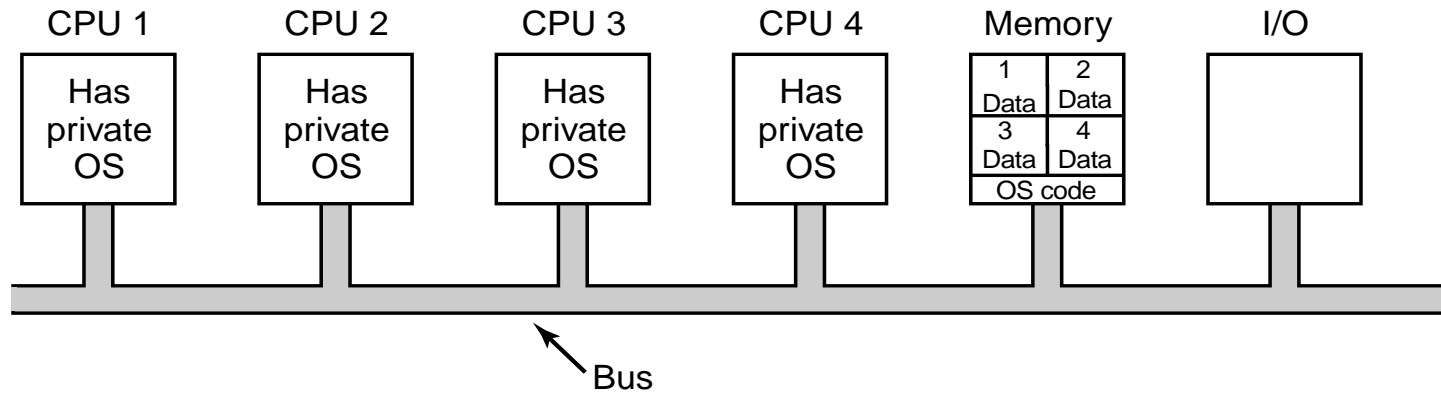
(b)



(c)

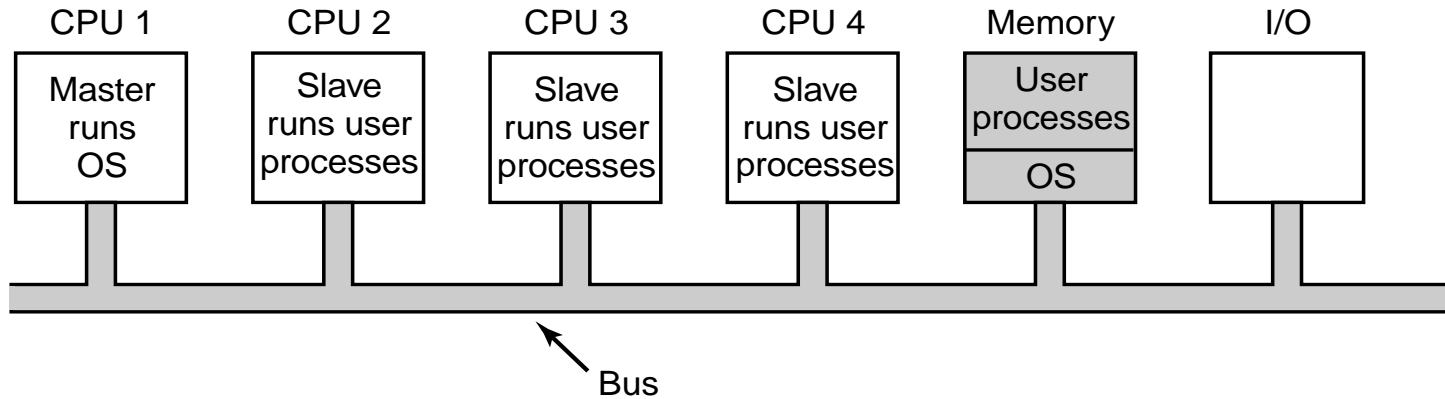
Systemy operacyjne (1)

- niezależny system operacyjny dla każdego CPU



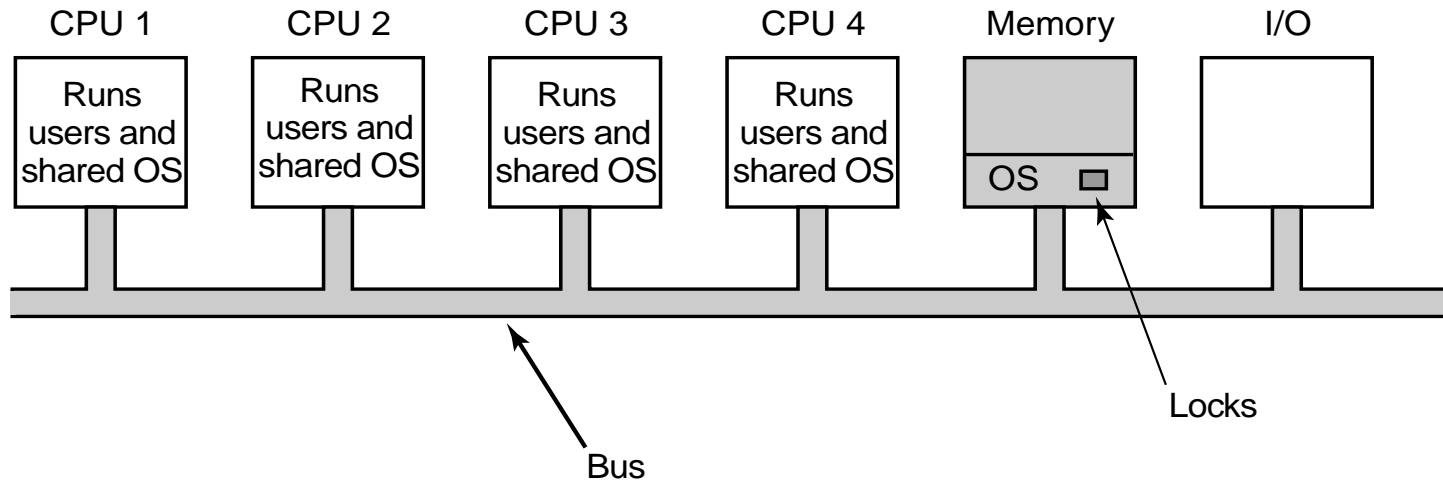
Systemy operacyjne (2)

- konfiguracja typu nadrzędny/podrzędny (Master/Slave)



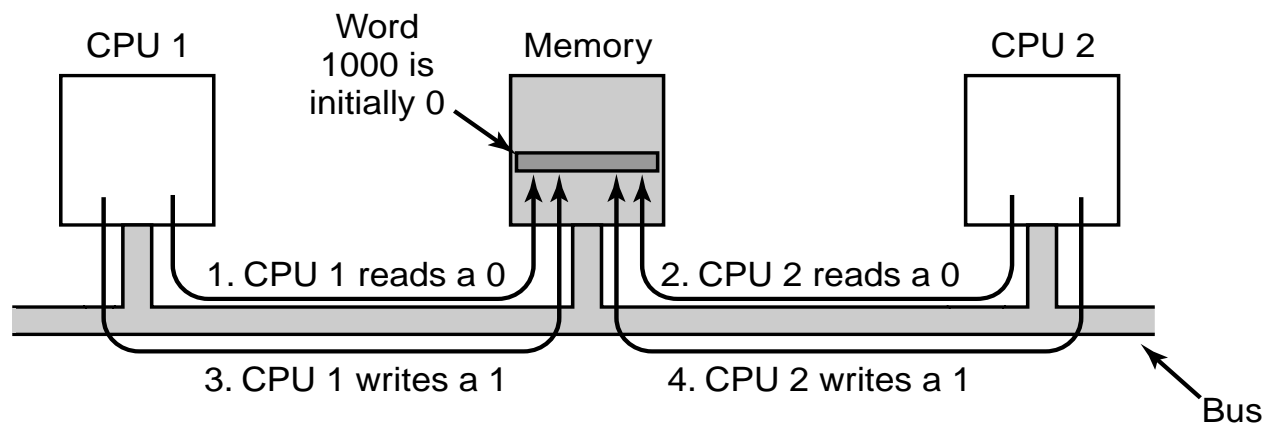
Systemy operacyjne (3)

- konfiguracja symetryczna (SMP)



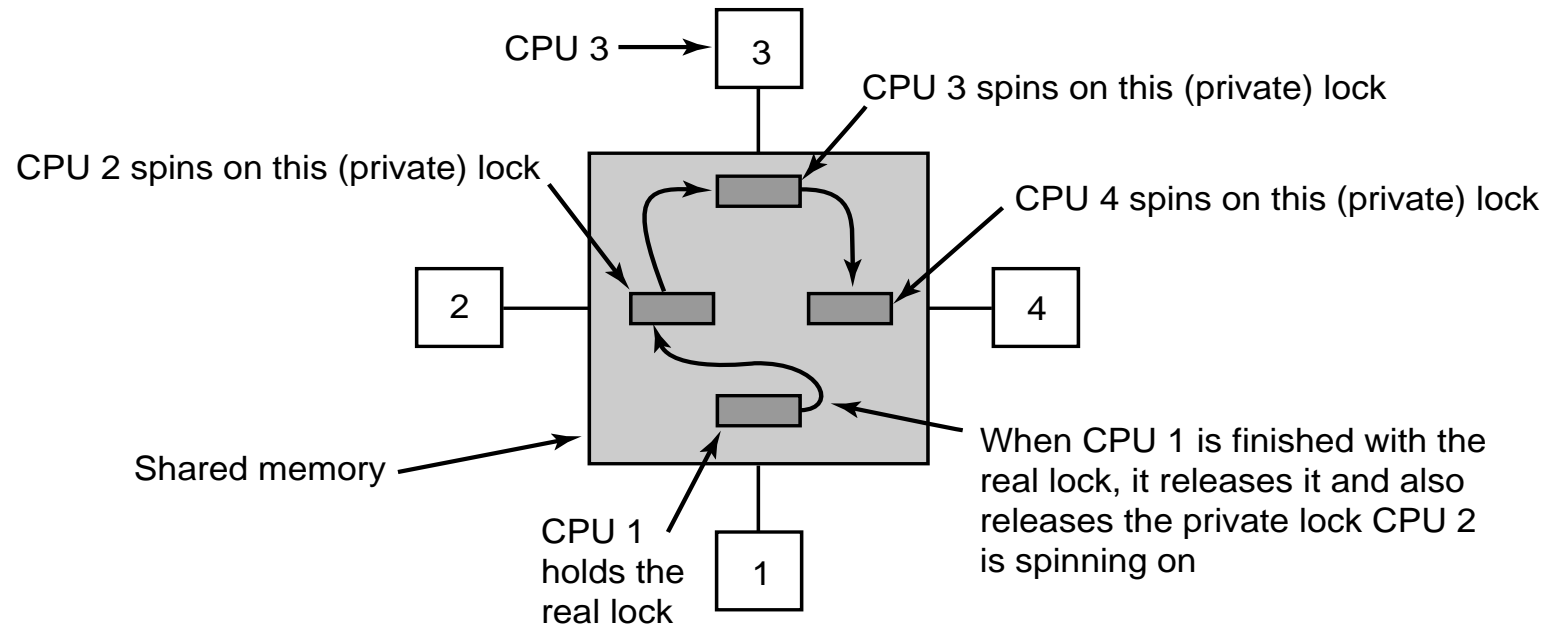
Synchronizacja (1)

- jednoczesny dostęp do pamięci
 - 1,2 – odczyt
 - 3,4 – zapis – wymaga wyłączenia
 - CPU: instrukcja TSL (Test and Set Lock)



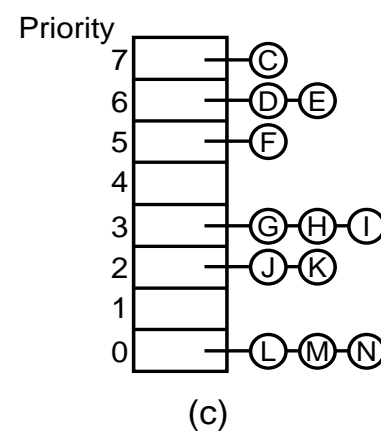
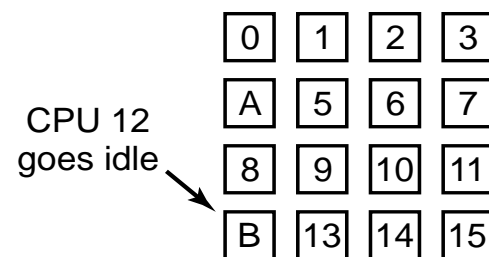
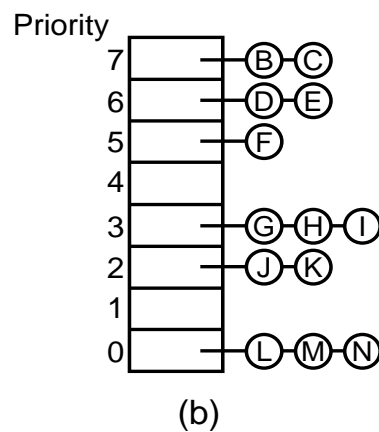
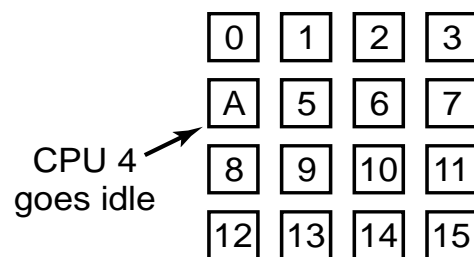
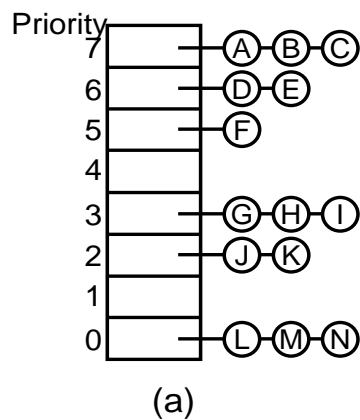
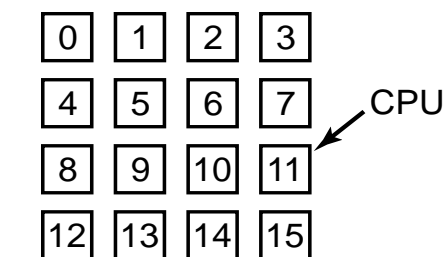
Synchronizacja (2)

□ zamki (locks)



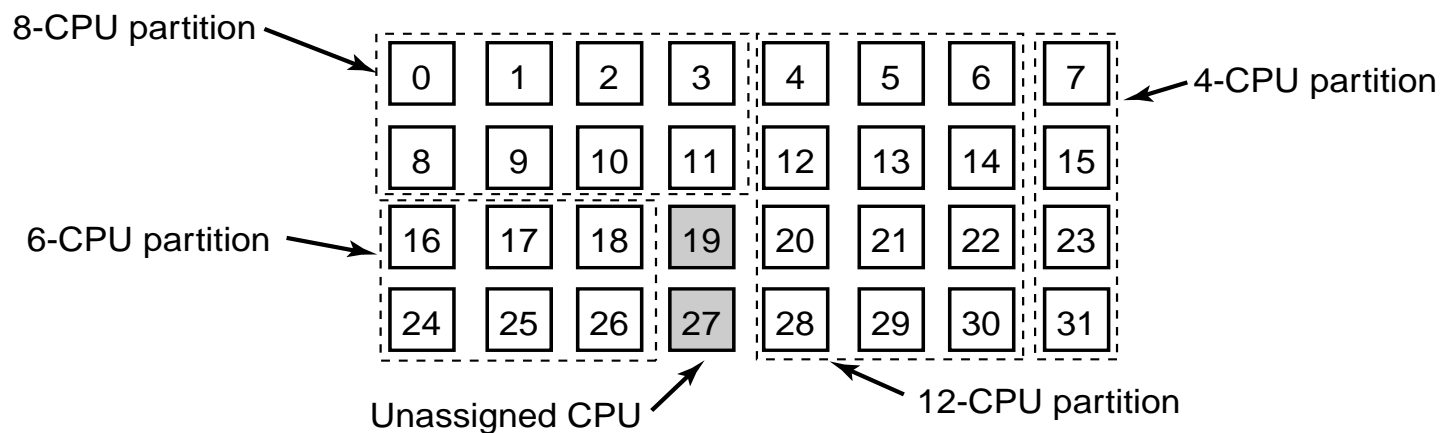
Szeregowanie (1)

- przydział wolnego procesora
 - wg. priorytetu i kolejności



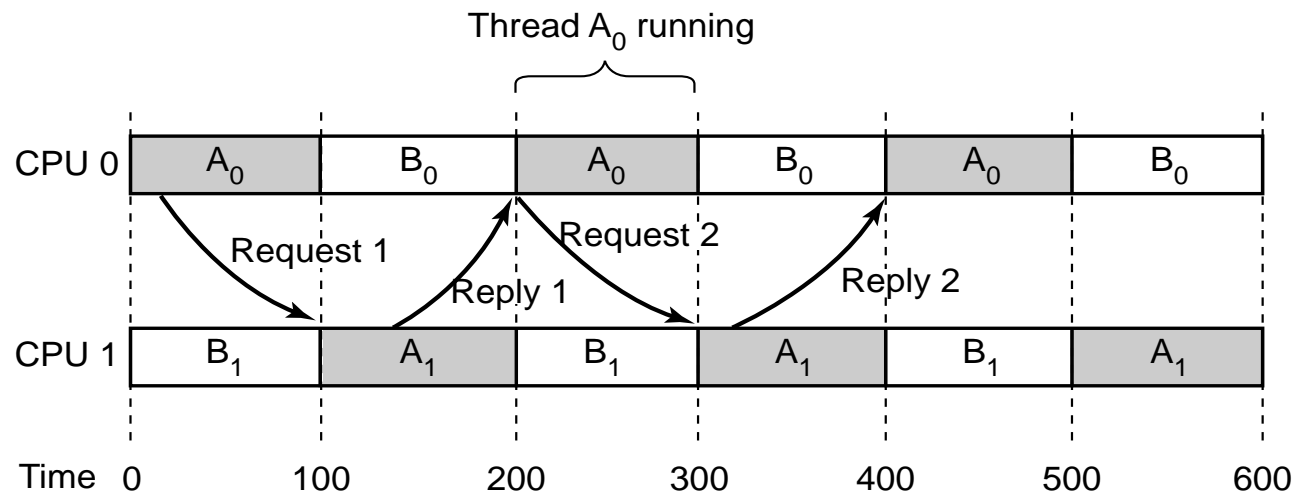
Szeregowanie (2)

- przydział partycji wieloprocessorowej
 - dla niezależnie pracującego systemu operacyjnego
 - dla wielowątkowego procesu/zadania, w trybie wyłączności



Szeregowanie (3)

- problemy komunikacyjne pomiędzy wątkami procesu
 - przykład: dwa procesy wielowątkowe A i B, na dwóch procesorach
 - przeplot wątków w przeciwności



- rozwiązanie: szeregowanie grupowe
 - następny slajd

Szeregowanie (4)

- szeregowanie grupowe – Gang Scheduling
 - grupa wątków lub procesów jest kolejkowana jako całość
 - wątki danego procesu pracują równocześnie na różnych procesorach
- przykładowy schemat przydziału CPU

		CPU					
		0	1	2	3	4	5
Time slot	0	A ₀	A ₁	A ₂	A ₃	A ₄	A ₅
	1	B ₀	B ₁	B ₂	C ₀	C ₁	C ₂
	2	D ₀	D ₁	D ₂	D ₃	D ₄	E ₀
	3	E ₁	E ₂	E ₃	E ₄	E ₅	E ₆
	4	A ₀	A ₁	A ₂	A ₃	A ₄	A ₅
	5	B ₀	B ₁	B ₂	C ₀	C ₁	C ₂
	6	D ₀	D ₁	D ₂	D ₃	D ₄	E ₀
	7	E ₁	E ₂	E ₃	E ₄	E ₅	E ₆

Multikomputery (1)

- definicja

- grupa powiązanych ze sobą procesorów, nie współdzielących ze sobą pamięci operacyjnej

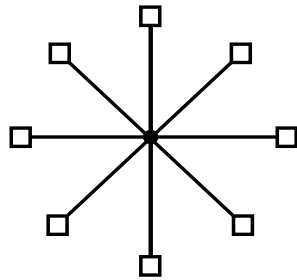
- klastery

- obliczeniowe (dedykowane)
- wysokiej dostępności (High Availability, Failover)
- grupa stacji roboczych – Cluster of Workstations (COW)

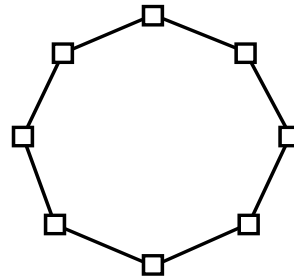
Multikomputery (2)

□ topologia połączeń

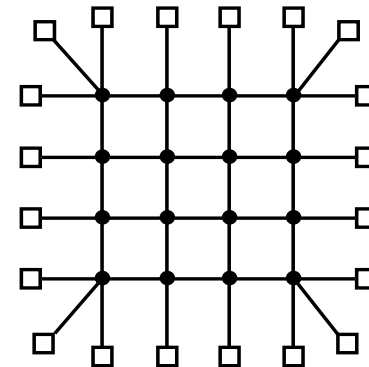
- (a) gwiazda, (b) pierścień, (c) krata (grid), (d) podwójny torus,
- (e) sześcián, (f) hipersześcián



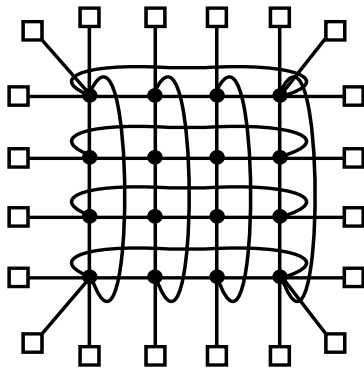
(a)



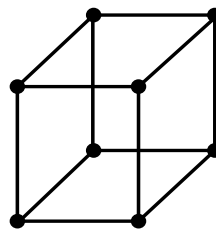
(b)



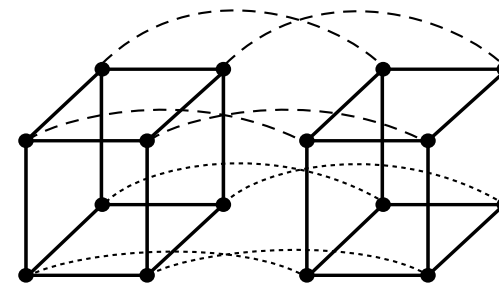
(c)



(d)



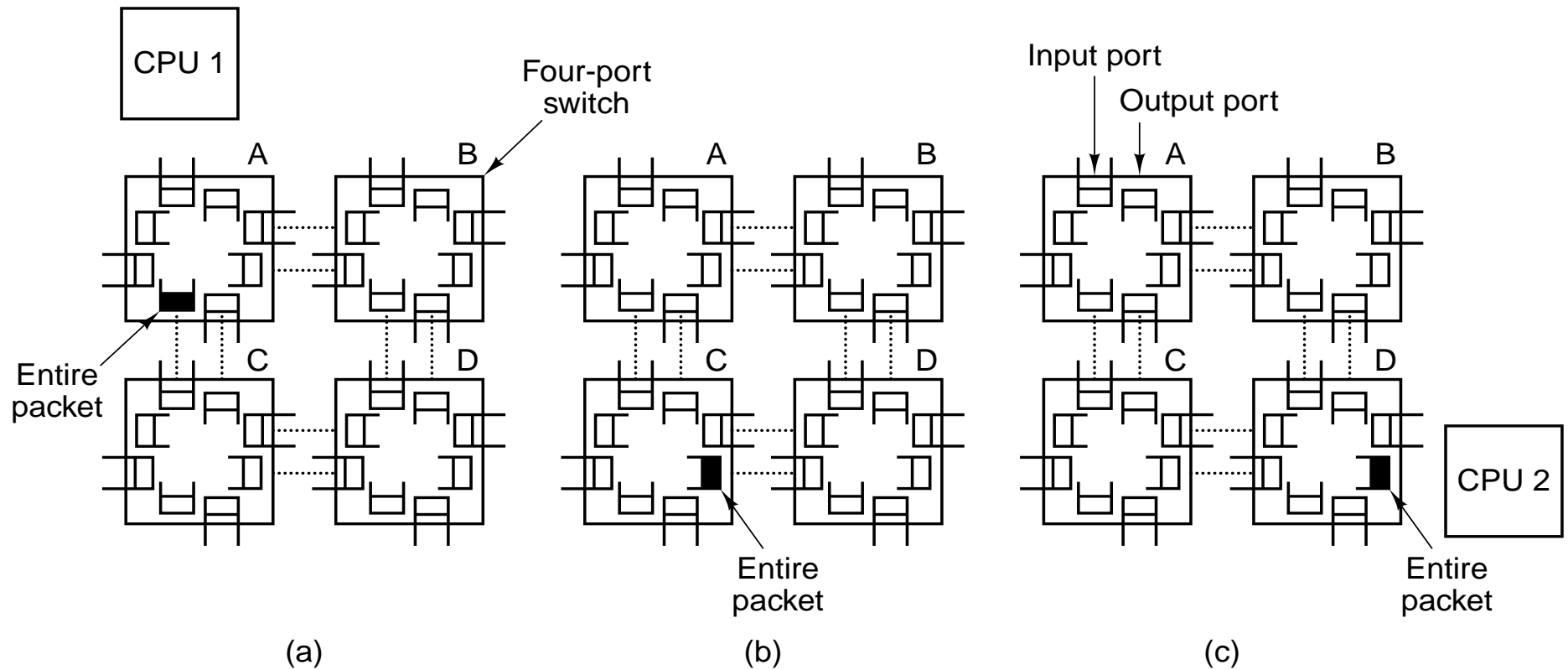
(e)



(f)

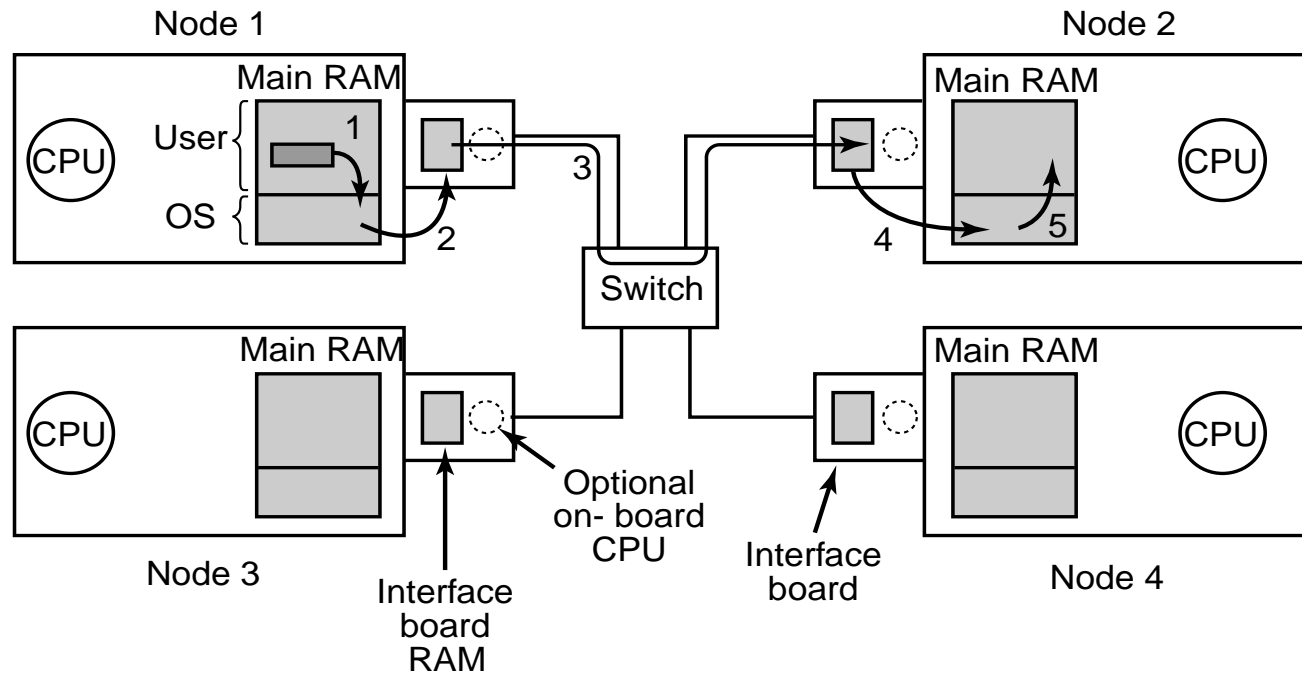
Multikomputery (3)

- sposób przełączania
 - przykład: store-and-forward



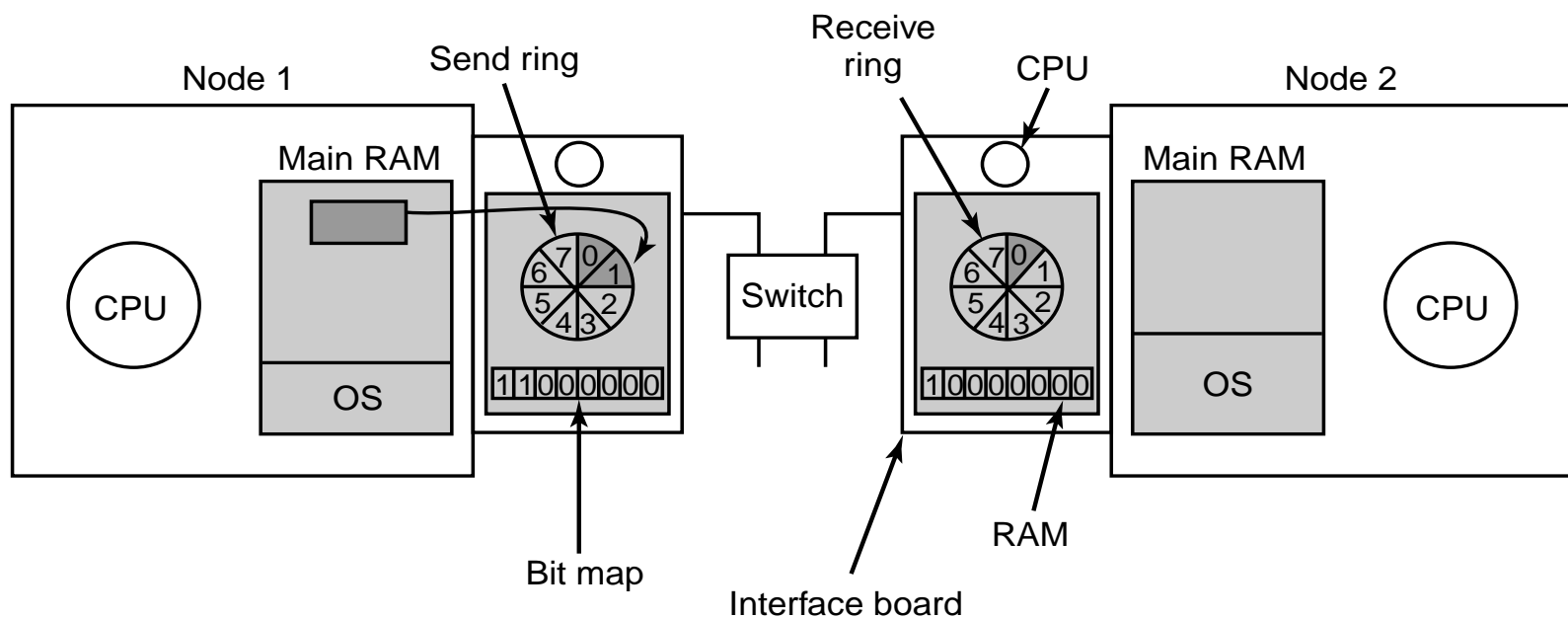
Multikomputery (4)

□ interfejsy sieciowe



Niskopoziomowe We/Wy

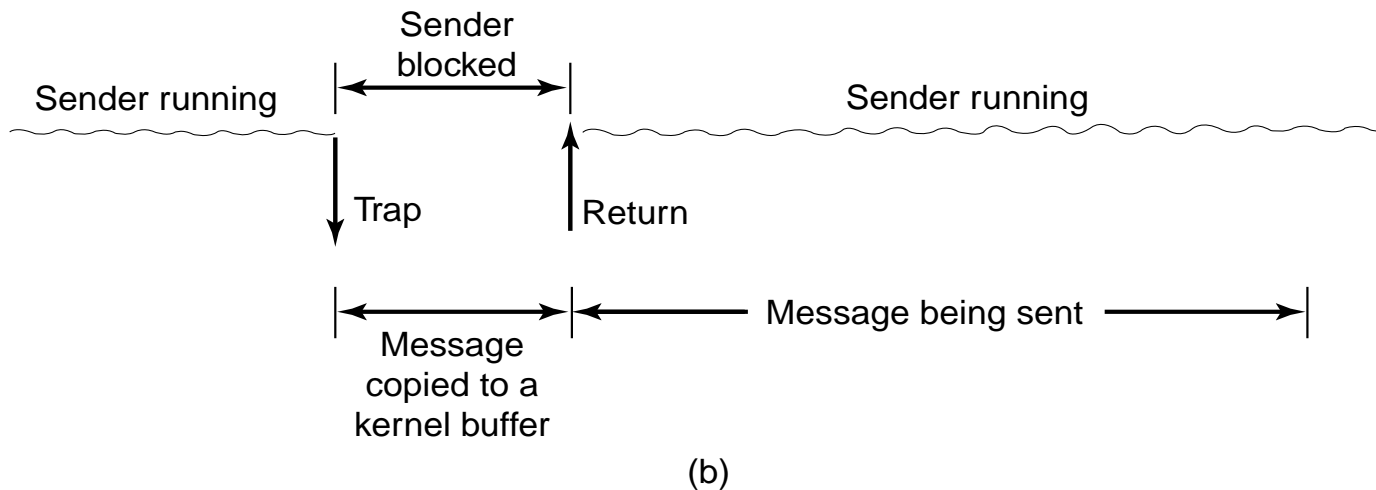
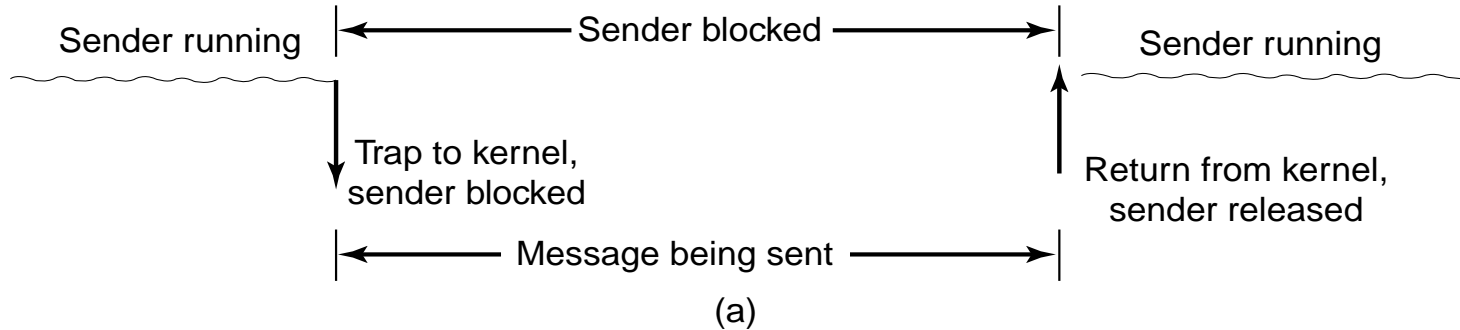
- komunikacja międzywęzłowa
 - cykliczne bufory nadawczo–odbiorcze (send/receive rings)
 - koordynacja pracy CPU i procesora We/Wy na karcie sieciowej



Wy/Wy na poziomie użytkownika

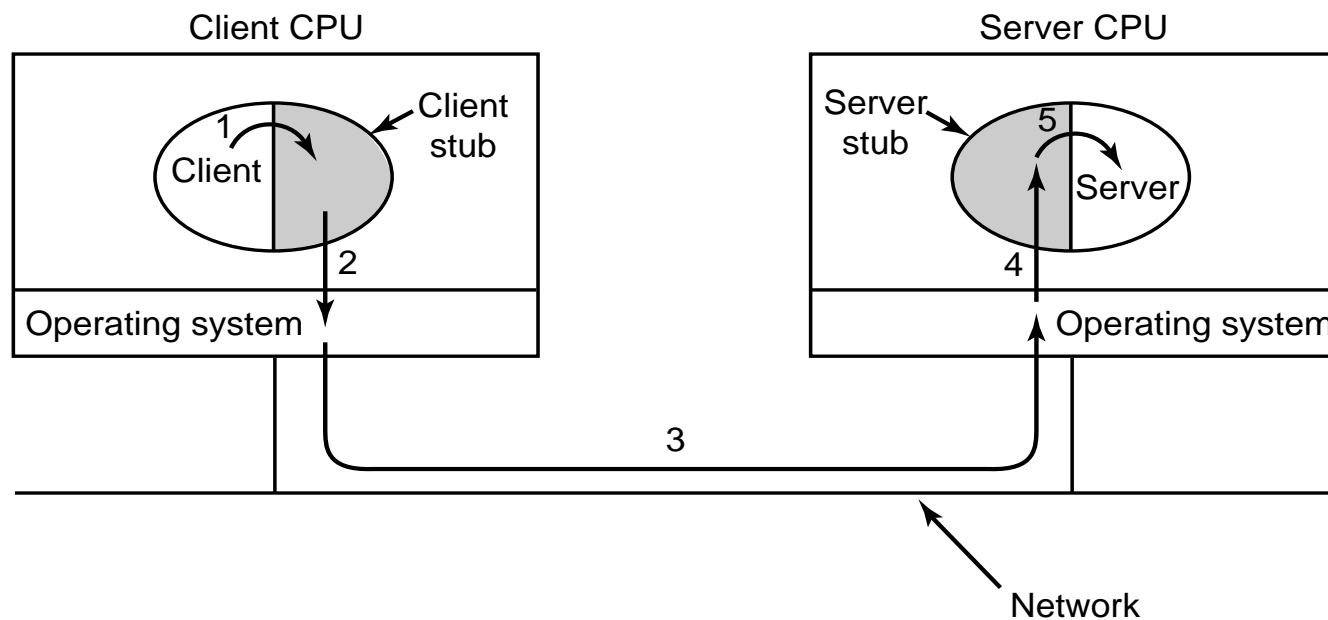
□ funkcje send()/receive()

- wersje: (a) – "blocking", (b) – "nonblocking"



Zdalne wywołanie procedury

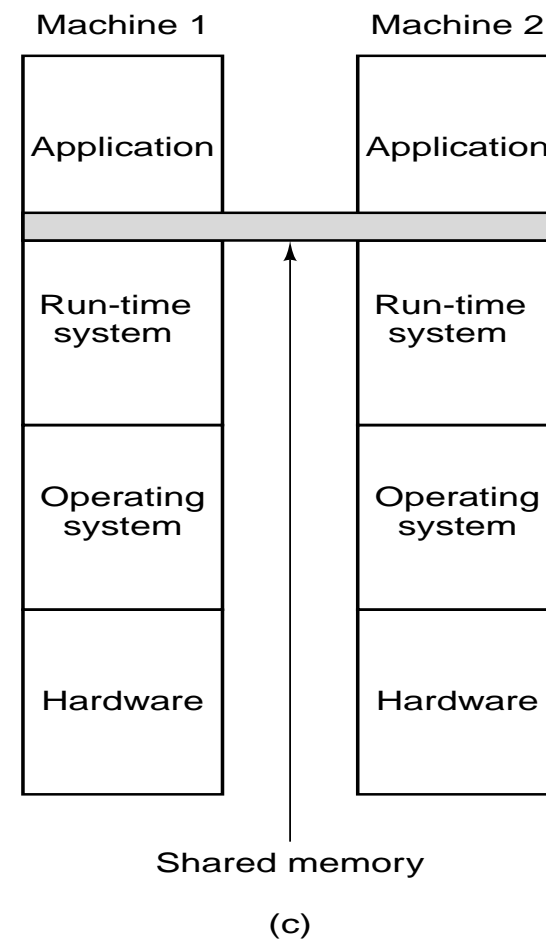
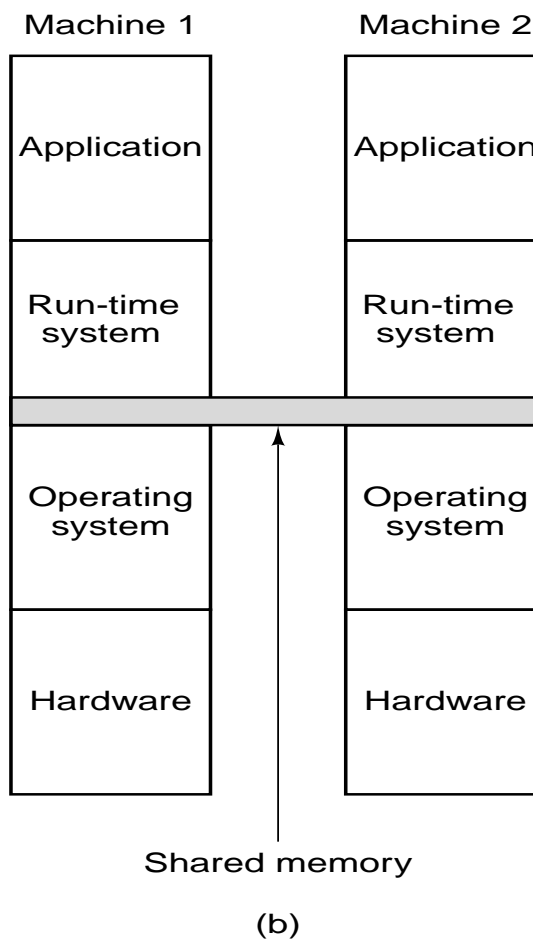
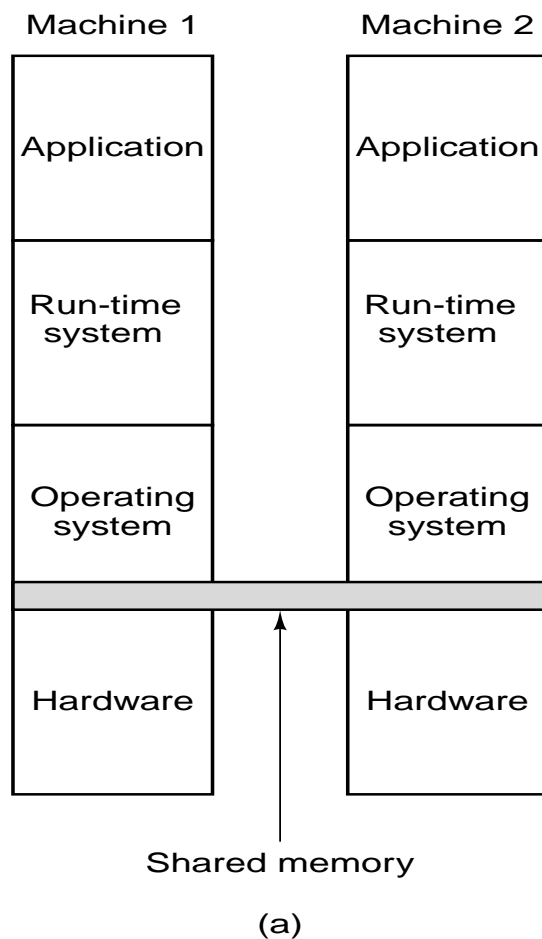
- Remote Procedure Call (RPC)
 - kolejne etapy realizacji
 - wyróżnione: namiastki (stubs) procedur



Rozproszona pamięć współdzielona (1)

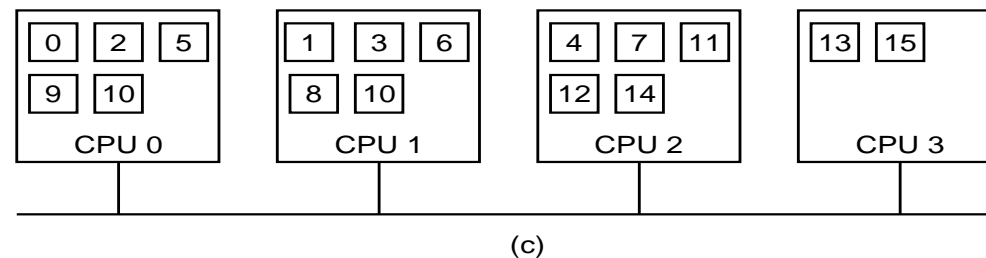
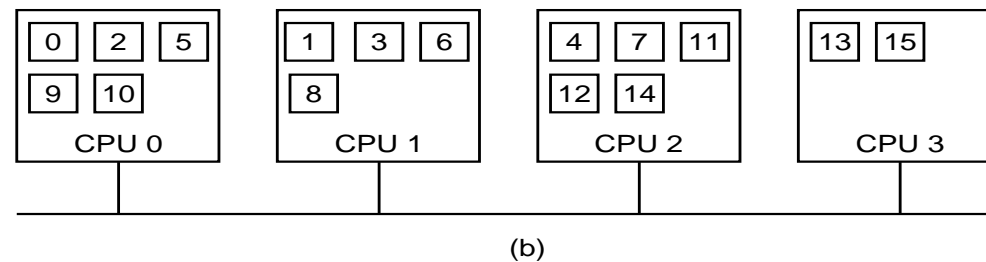
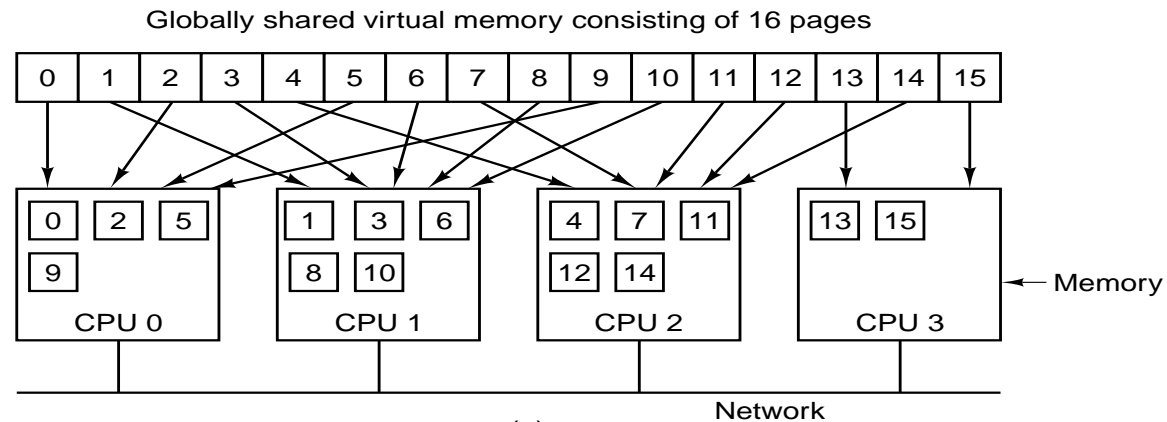
□ możliwości implementacji

- (a) – sprzęt, (b) – system operacyjny, (c) – aplikacje



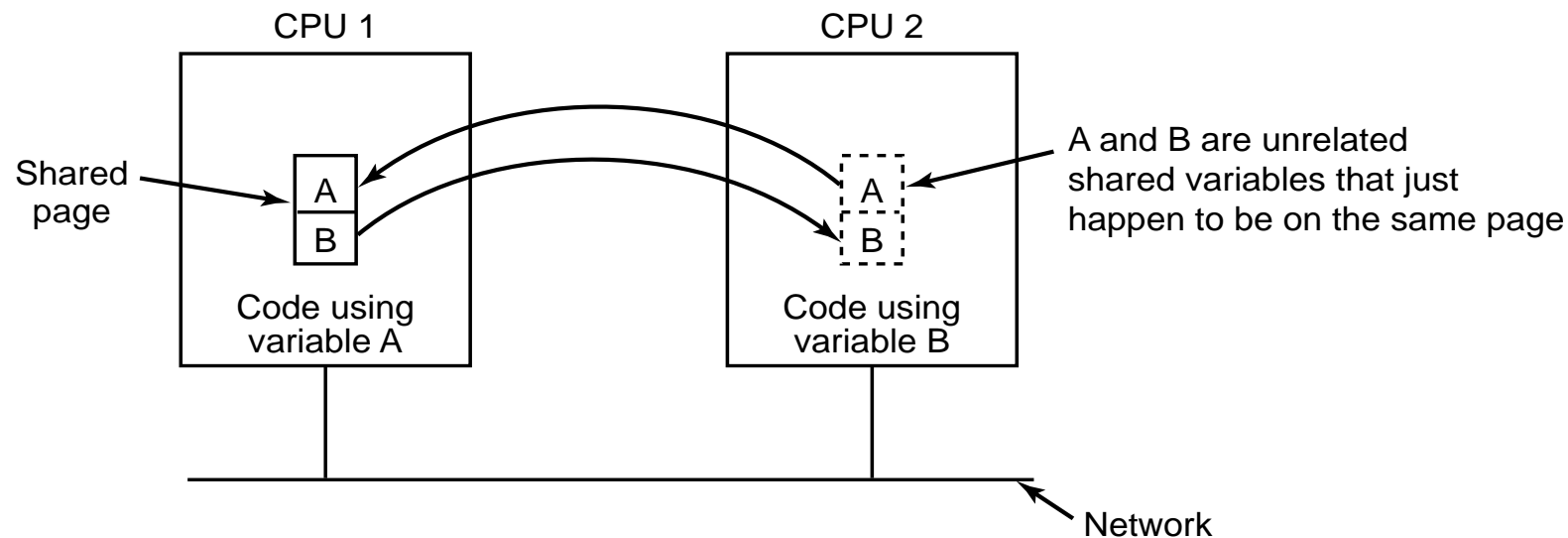
Rozproszona pamięć współdzielona (2)

□ replikacja stron pamięci



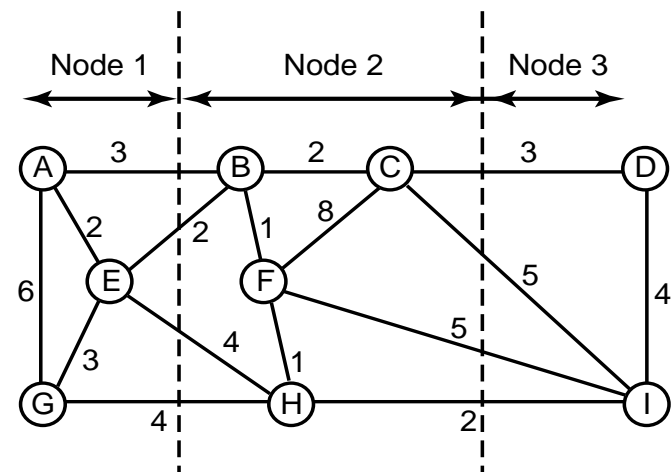
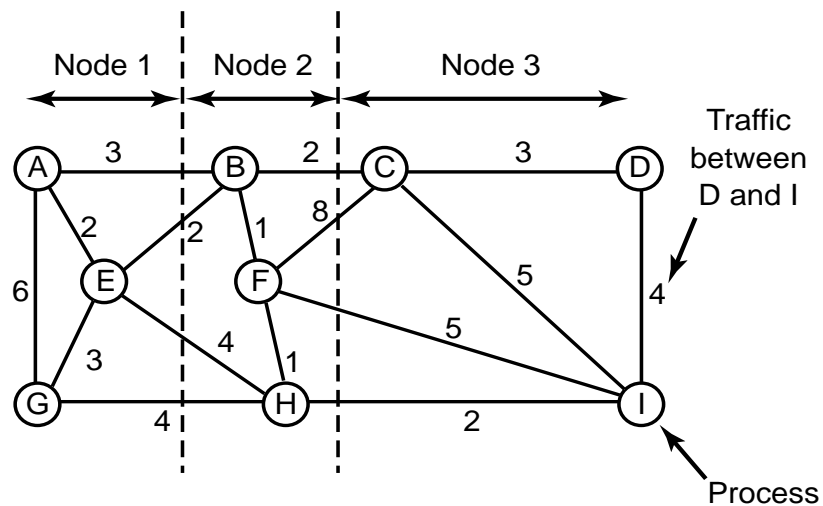
Rozproszona pamięć współdzielona (3)

- niezamierzona globalizacja



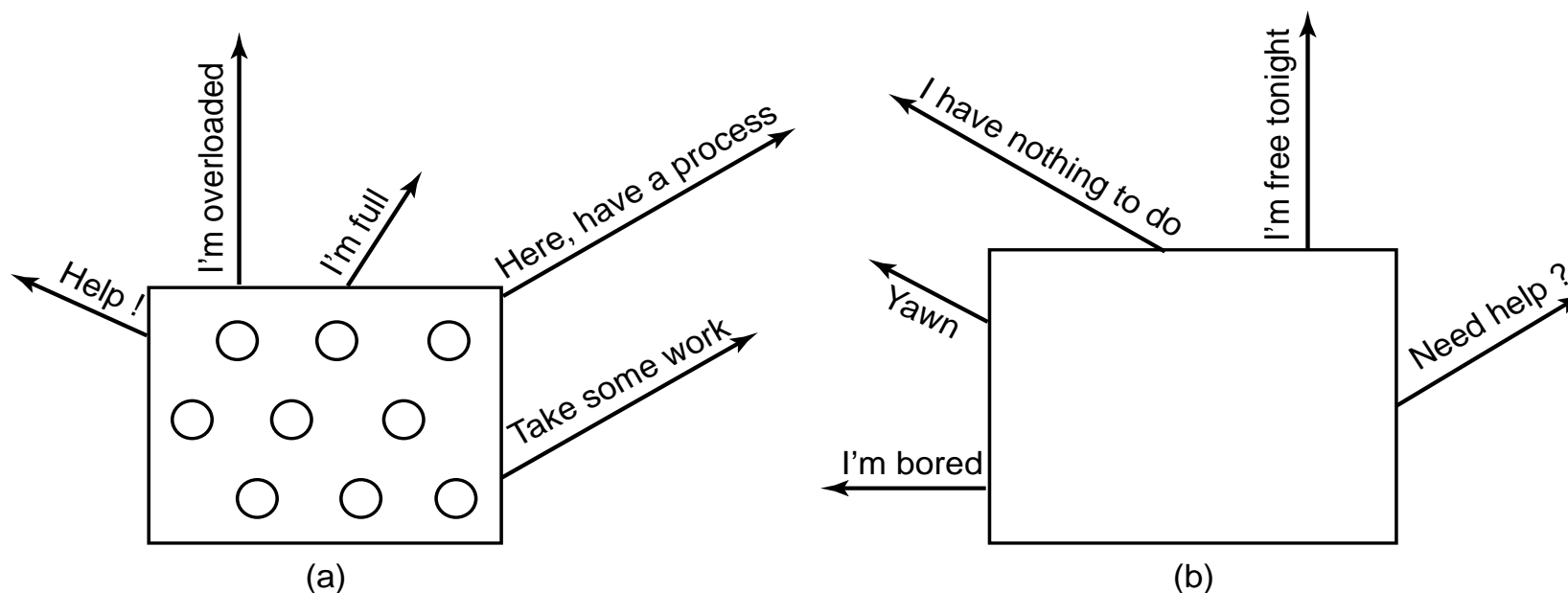
Szeregowanie zadań (1)

- deterministyczne wyrównywanie obciążenia
 - algorytm optymalizacji przepływów



Szeregowanie zadań (2)

- heurystyczne wyrównywanie obciążenia
 - (a) – inicjowane przez przeciążonego nadawcę
 - (b) – inicjowane przez przeciążonego odbiorcę



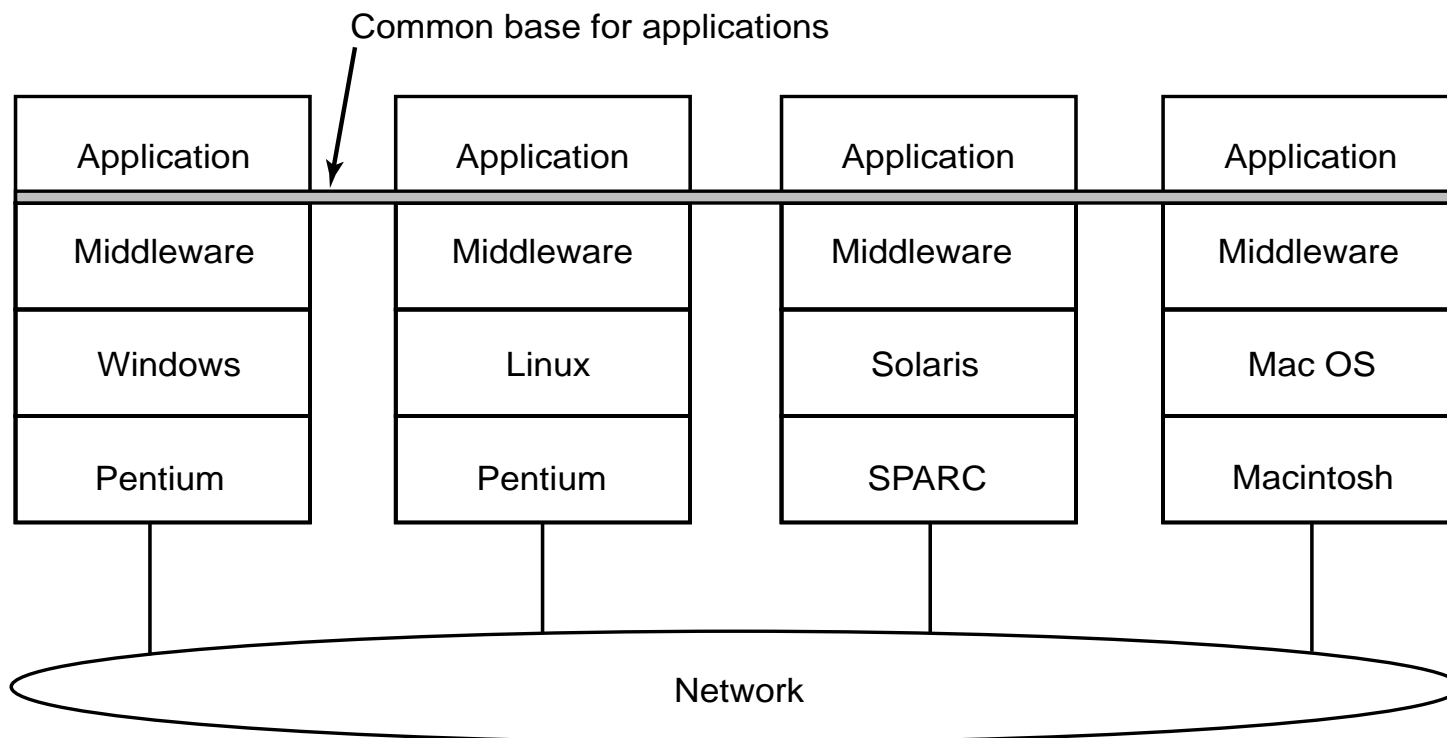
Rozproszone systemy komputerowe (1)

□ porównanie

Item	Multiprocessor	Multicomputer	Distributed System
Node configuration	CPU	CPU, RAM, net interface	Complete computer
Node peripherals	All shared	Shared exc. maybe disk	Full set per node
Location	Same rack	Same room	Possibly worldwide
Internode communication	Shared RAM	Dedicated interconnect	Traditional network
Operating systems	One, shared	Multiple, same	Possibly all different
File systems	One, shared	One, shared	Each node has own
Administration	One organization	One organization	Many organizations

Rozproszone systemy komputerowe (2)

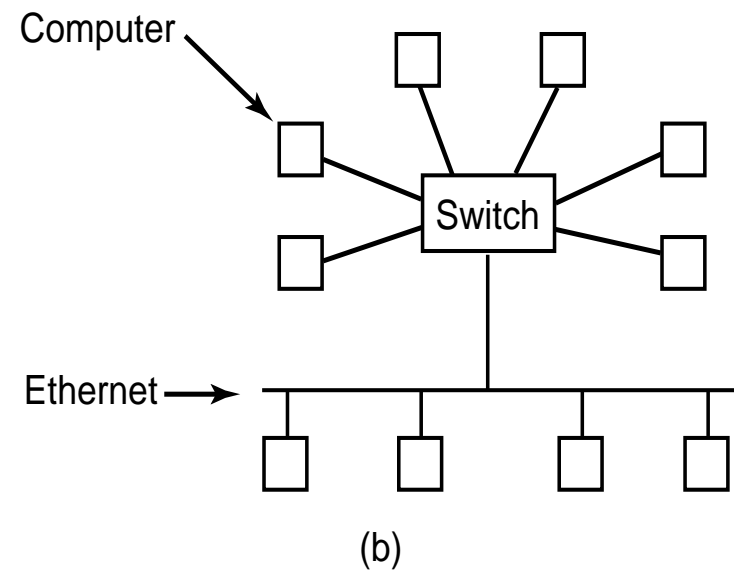
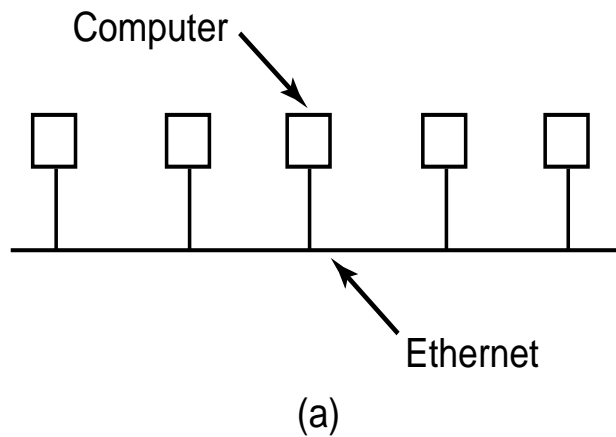
- jednorodność dzięki "middleware"



Infrastruktura sieciowa (1)

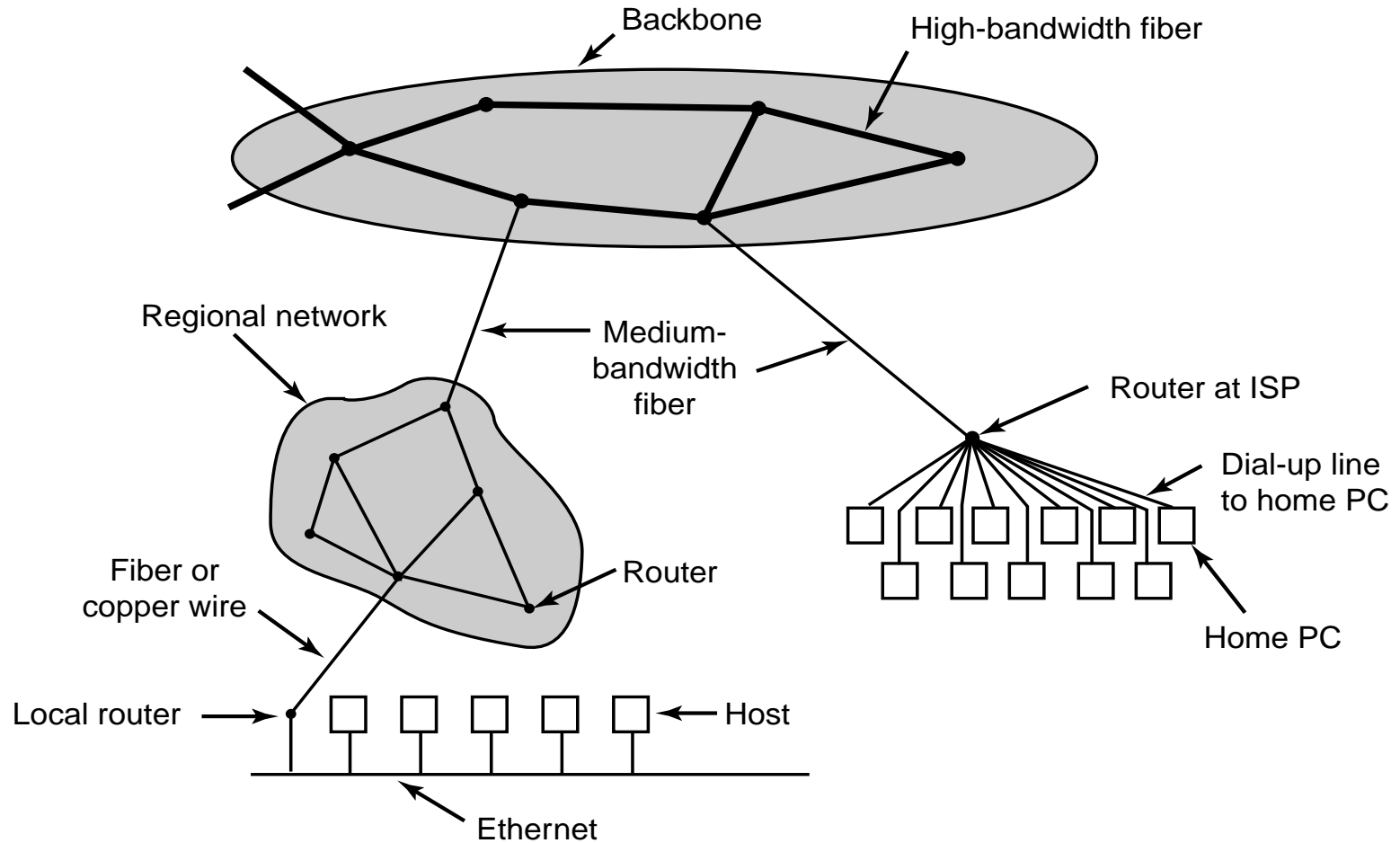
□ Ethernet

- (a) – klasyczny (magistrala)
- (b) – przełączany (drzewo)



Infrastruktura sieciowa (2)

□ Internet



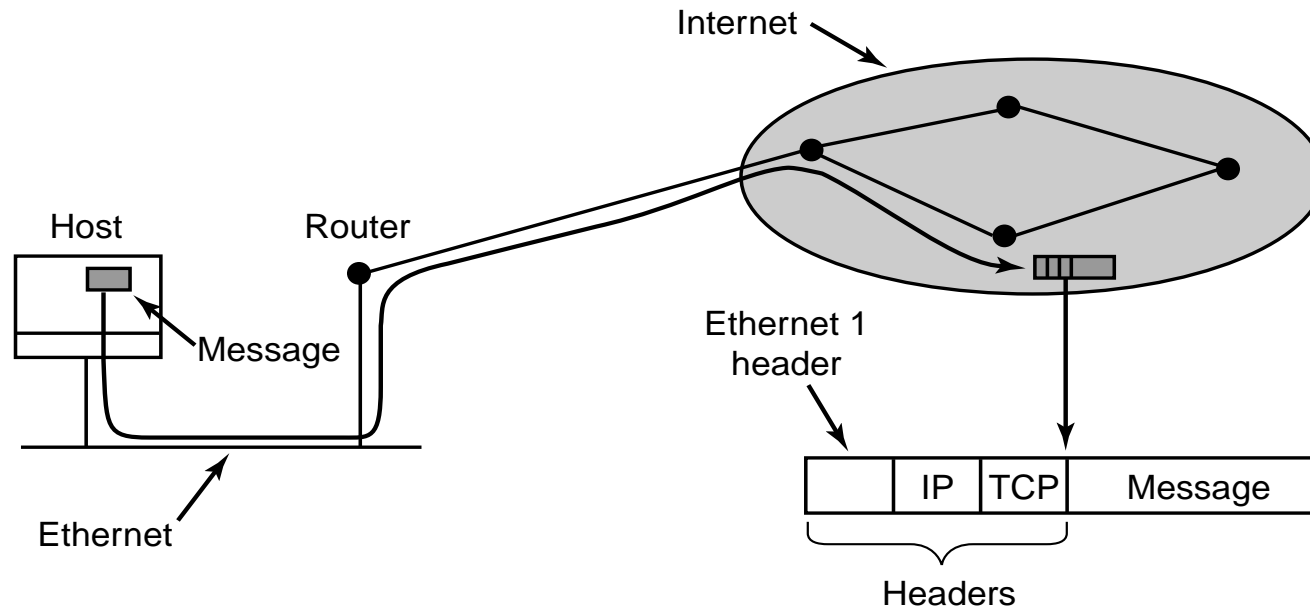
Usługi i protokoły sieciowe (1)

□ podstawowa klasyfikacja

	Service	Example
Connection-oriented	Reliable message stream	Sequence of pages of a book
	Reliable byte stream	Remote login
	Unreliable connection	Digitized voice
Connectionless	Unreliable datagram	Network test packets
	Acknowledged datagram	Registered mail
	Request-reply	Database query

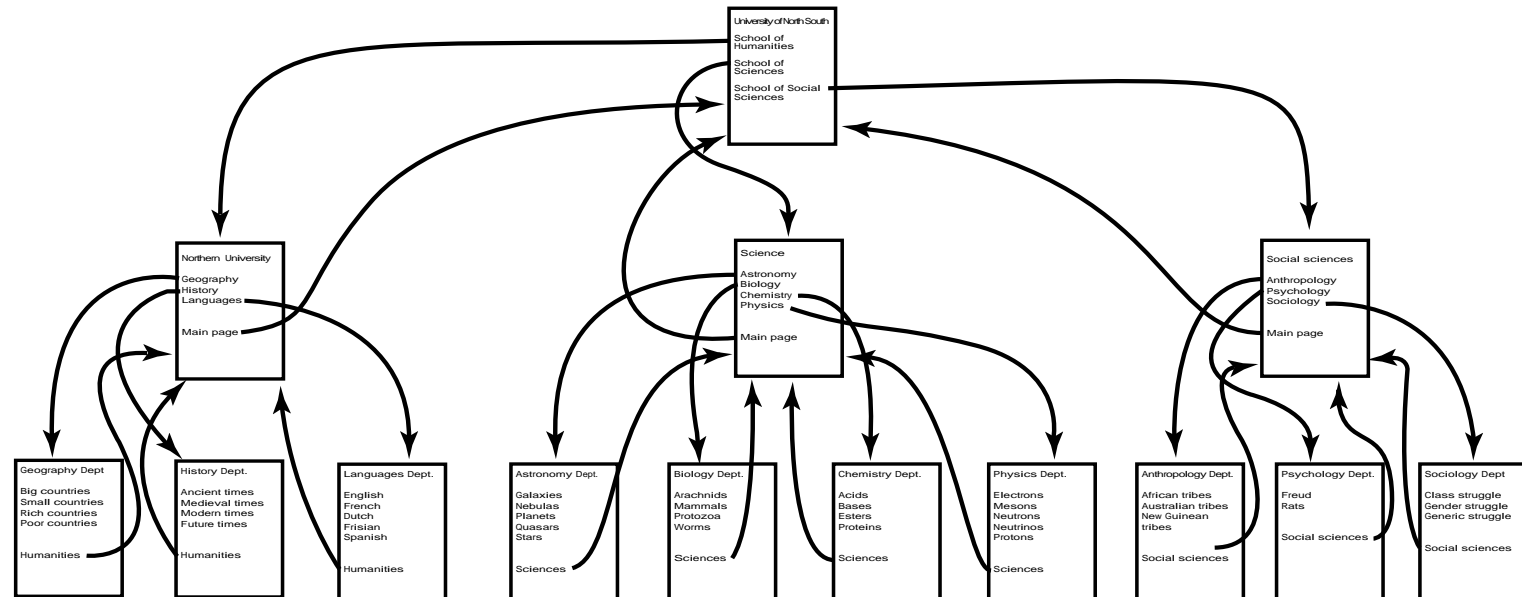
Usługi i protokoły sieciowe (2)

- współdziałanie protokołów (TCP/IP/Ethernet)



Middleware: WWW

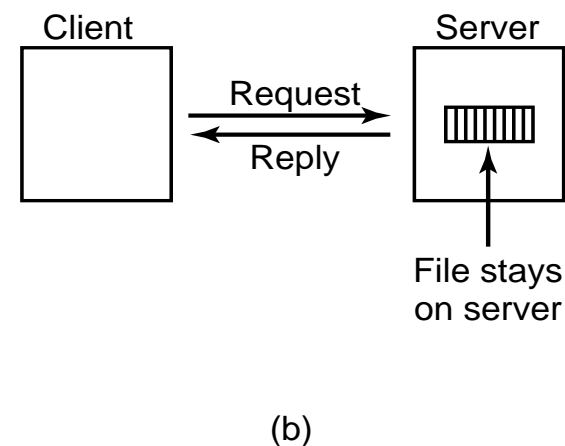
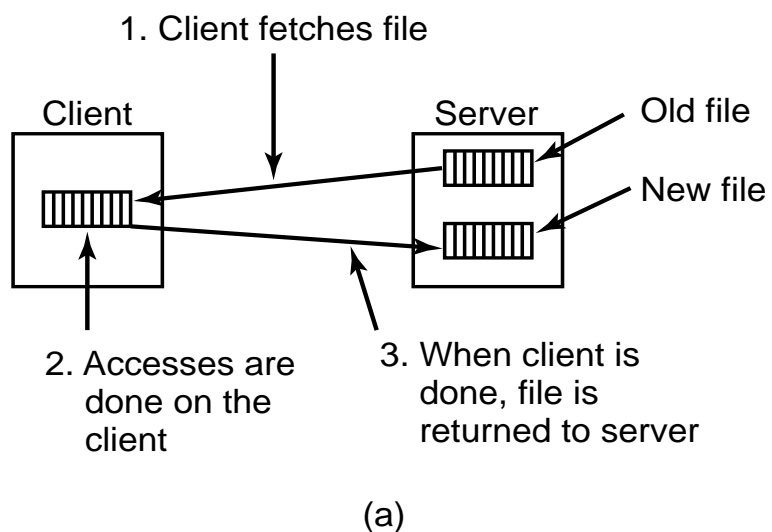
- skierowany graf standardowych dokumentów



Middleware: system plików (1)

□ transfer plików

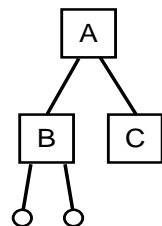
- (a) – download/upload
- (b) – zdalny dostęp do pliku



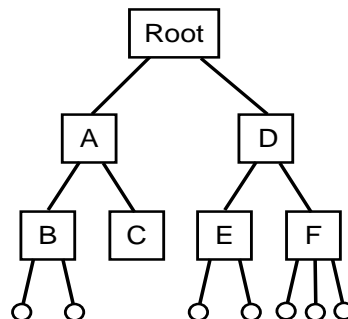
Middleware: system plików (2)

- przeźroczystość nazewnictwa/reprezentacji

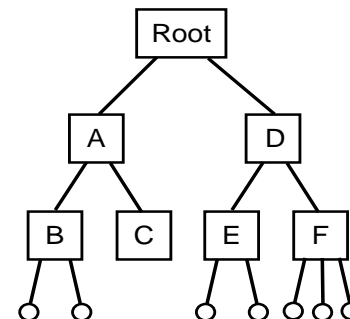
File server 1



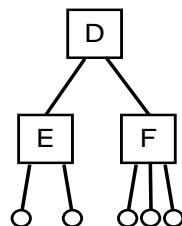
Client 1



Client 1

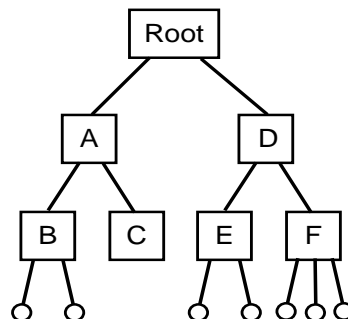


File server 2



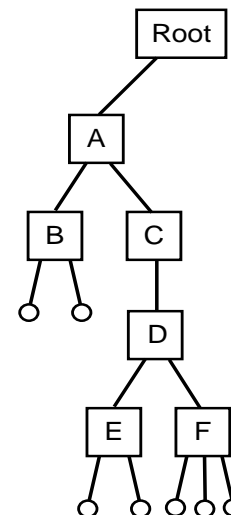
(a)

Client 2



(b)

Client 2

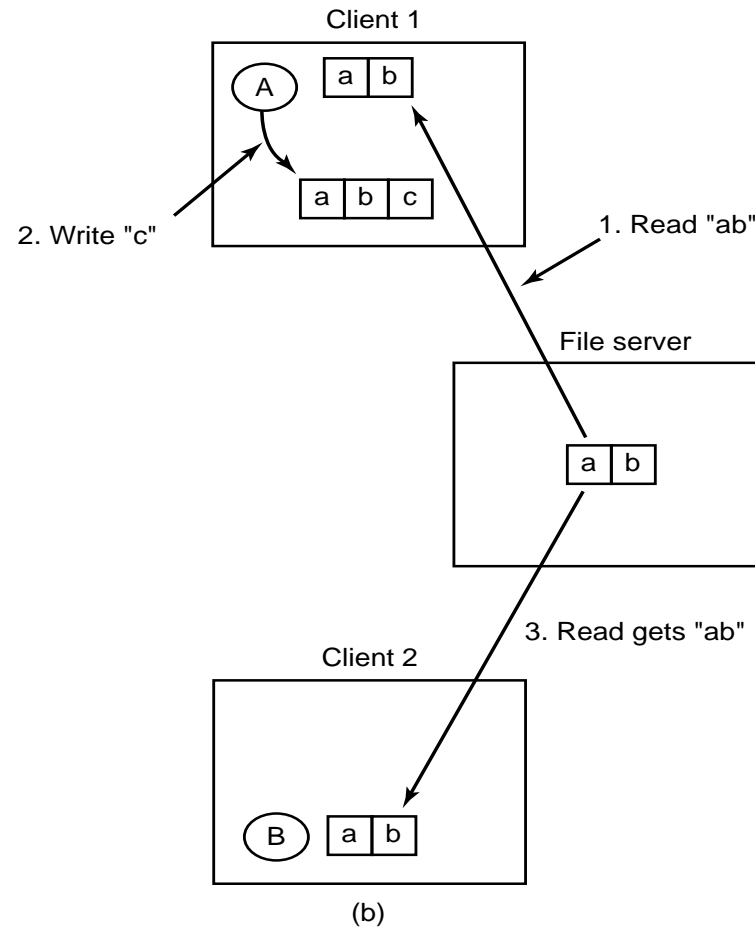
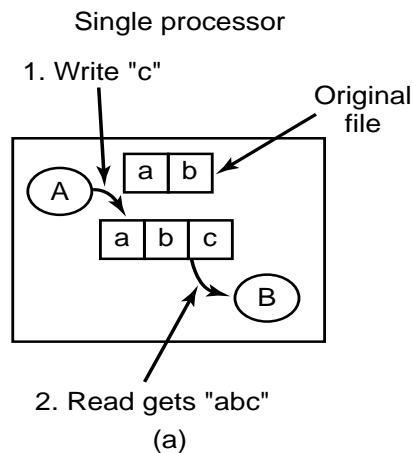


(c)

Middleware: system plików (3)

□ semantyka współdzielenia plików

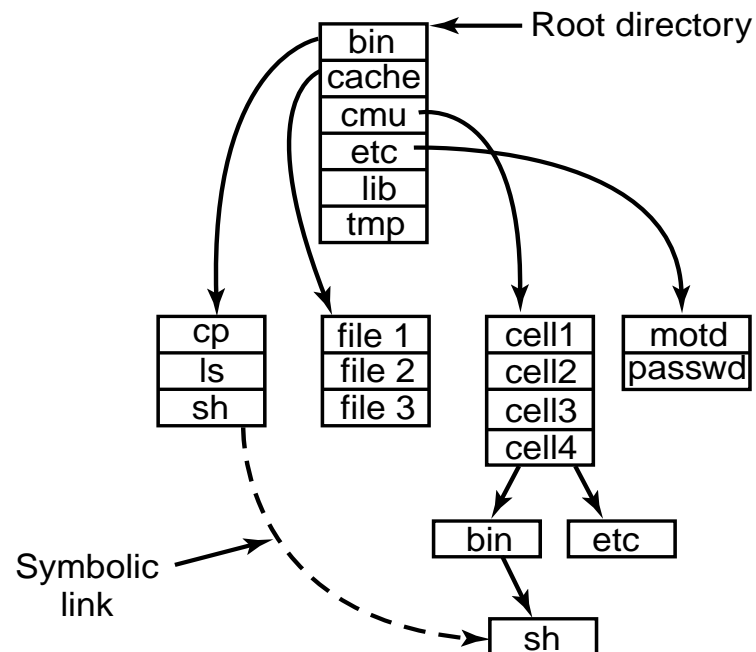
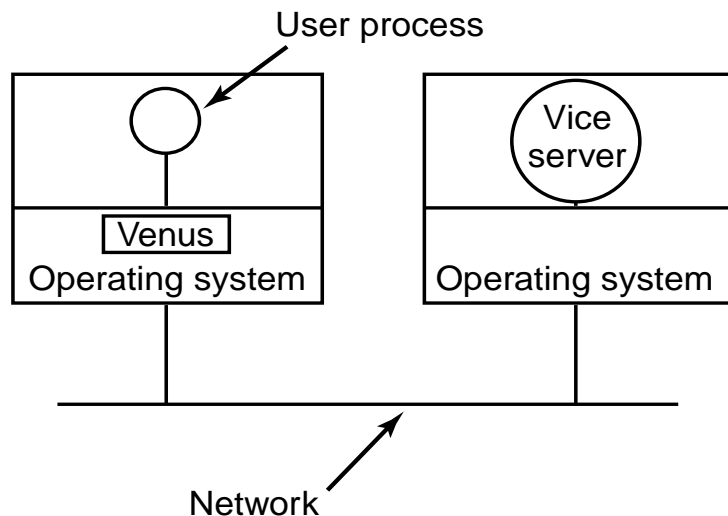
- (a) uniprocessor gwarantuje spójność sekwencyjną
- (b) system rozproszony może udostępnić nieaktualne dane



Middleware: system plików (4)

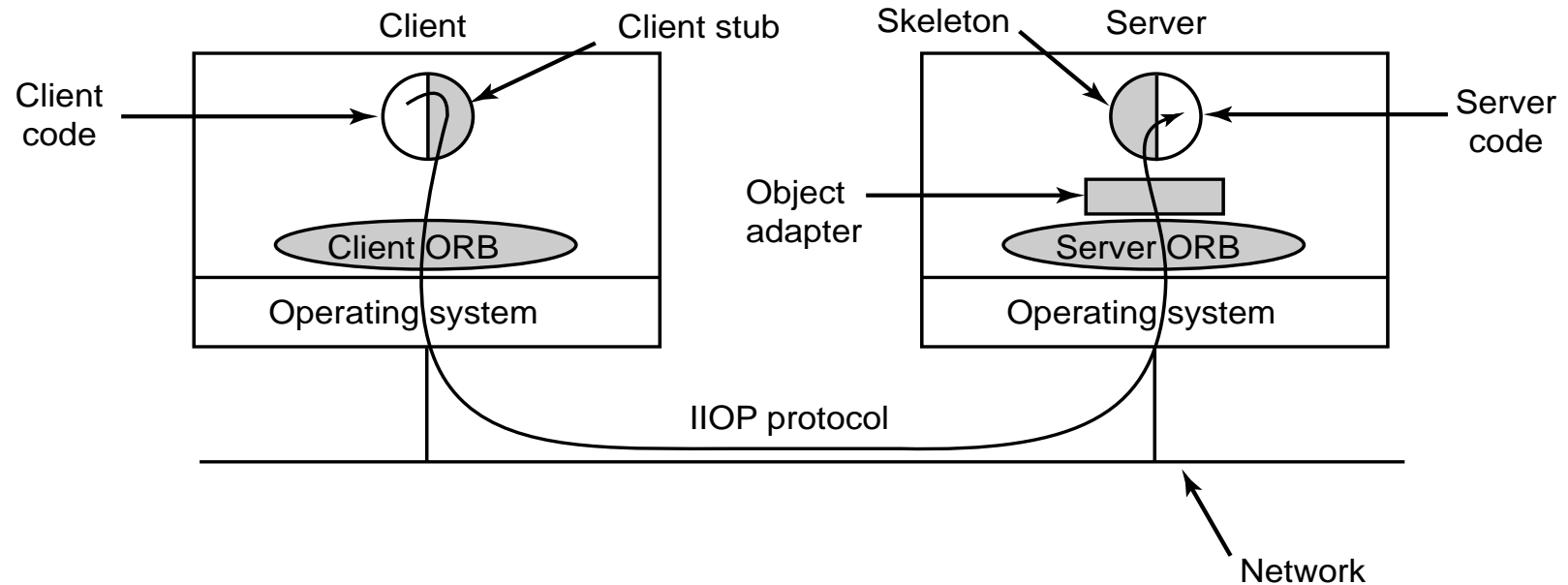
□ Andrew File System (AFS)

- serwer AFS (vice) pracuje w trybie użytkownika
- klient AFS (venus) pracuje w trybie jądra (!)
- stacje robocze są grupowane w "komórki" (cells)



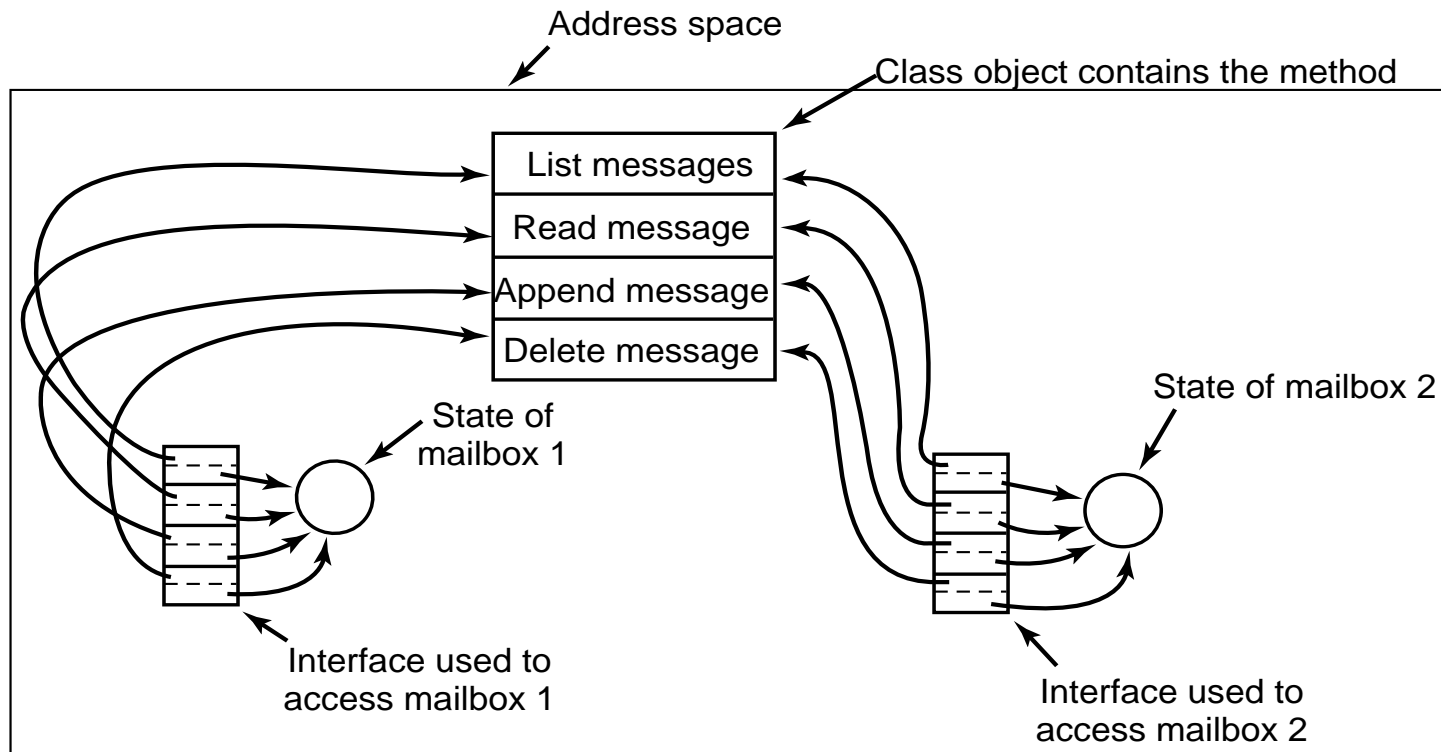
Middleware: CORBA

□ Common Object Request Broker Architecture



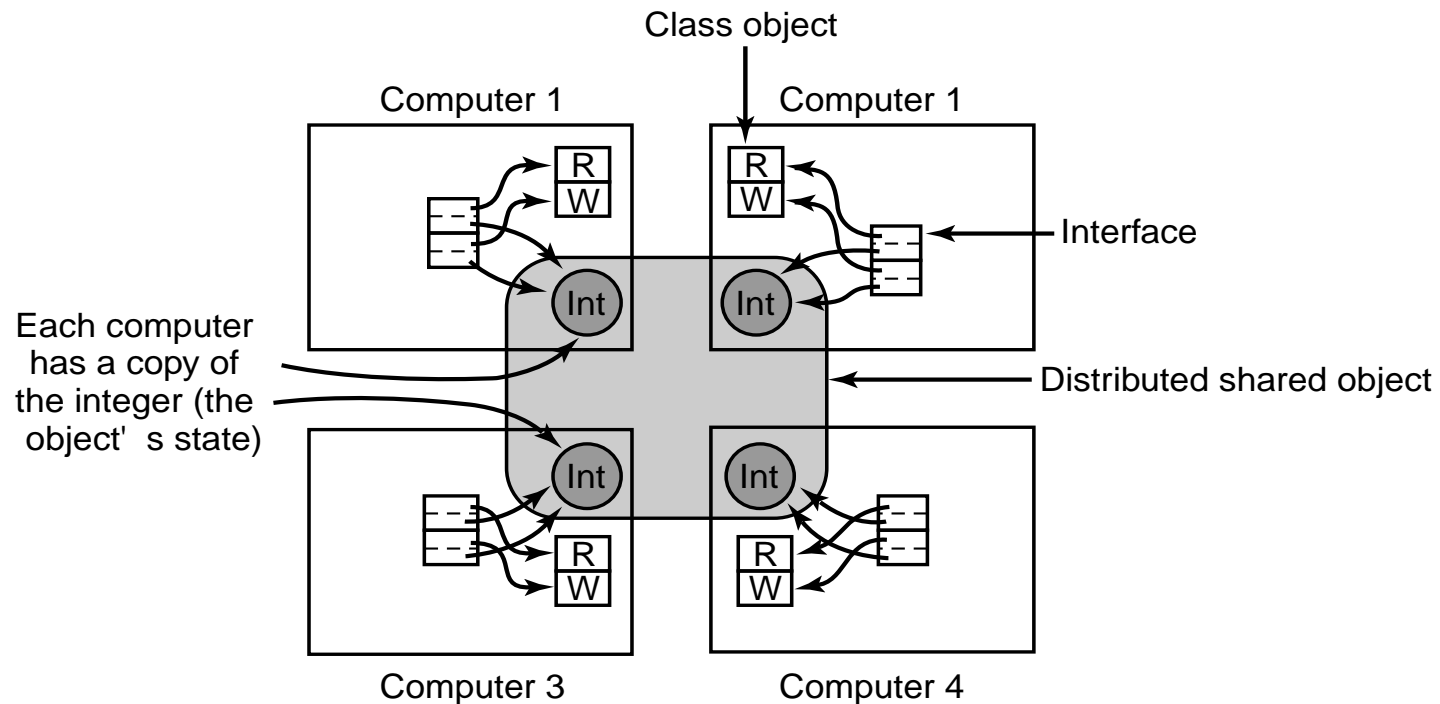
Middleware: Globe (1)

- globalny system obiektowy z replikacją
 - zaprojektowany dla miliardów użytkowników (!)



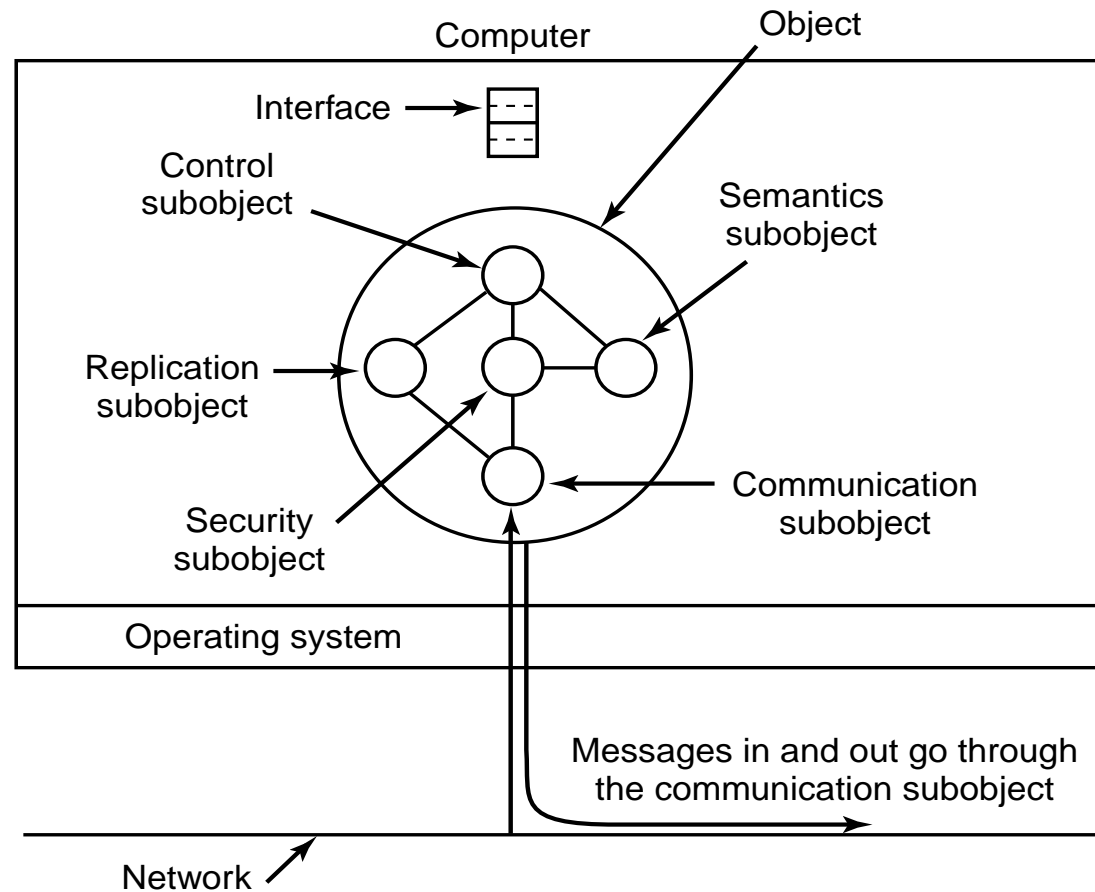
Middleware: Globe (2)

- rozproszony współdzielony obiekt
 - każdy węzeł posiada informację o stanie danego obiektu



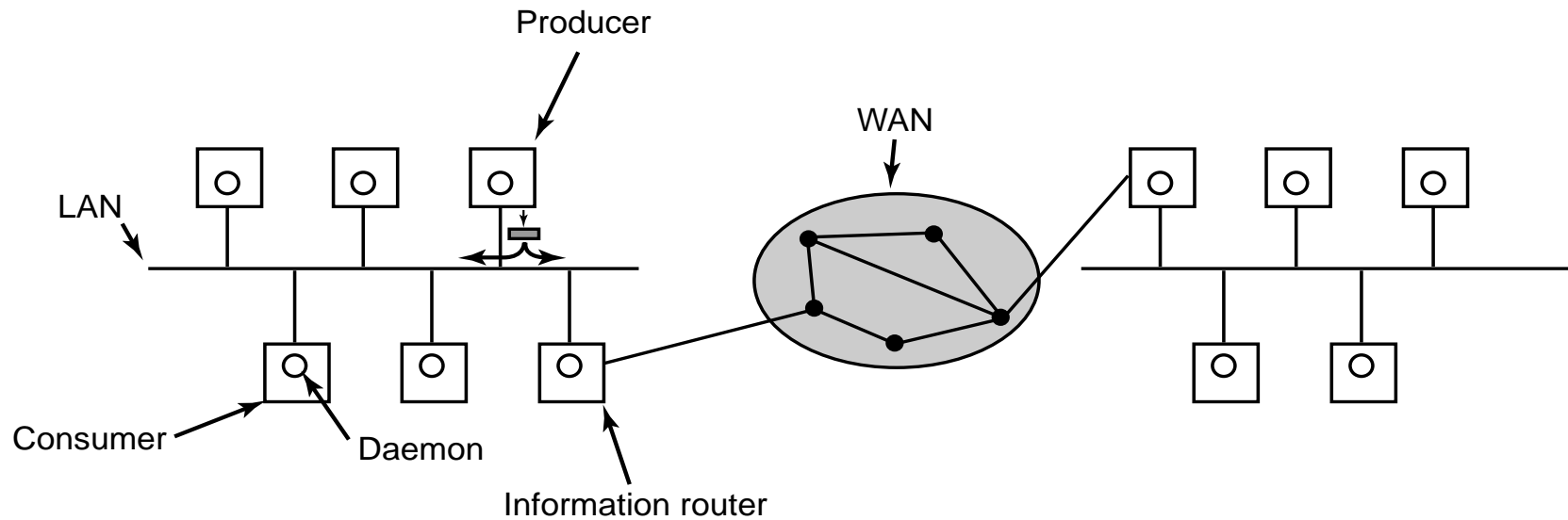
Middleware: Globe (3)

□ wewnętrzna struktura obiektu



Middleware: koordynacja

- architektura typu Publish-Subscribe



===