

## Curs 7. Managementul dispozitivelor de I/O

Dispozitivele de intrare/ iesire(I/O) va sunt cunoscute din utilizarea unui calculator si pot fi impartite pe 3 categorii: dispozitive pentru interfata umana cum ar fi tastatura, mouse, imprimanta, monitor, dispozitive pentru interfata cu diferite echipamente (de ex senzori) si dispozitive utilizate pentru comunicare (modem).

Dincolo de diferentierea facuta in functie de interfatarea realizata de un dispozitiv o alta caracteristica importanta in cazul dispozitivelor de I/O este cea referitoare la viteza acestora. In figura urmatoare aveti o reprezentare a vitezelor in cazul diferitelor dispozitive de I/O:

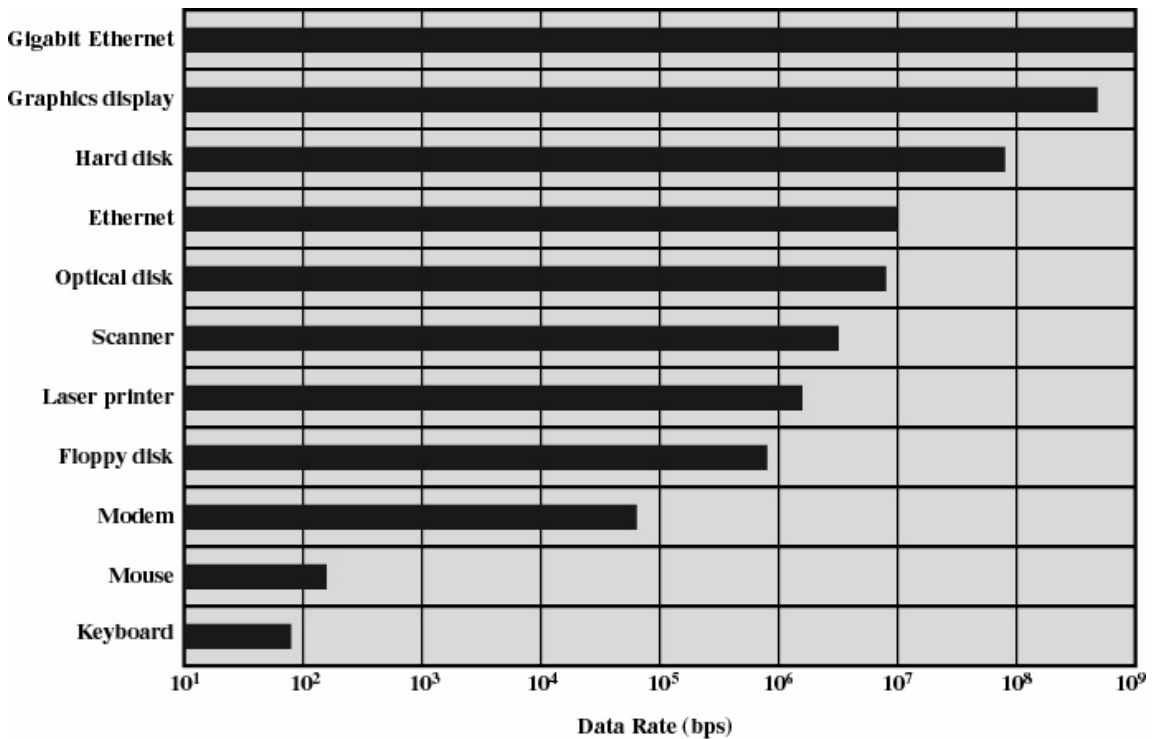


Figura 7.1. Comparatie intre vitezele dispozitivelor I/O

Principala tehnica de lucru cu un dispozitiv de I/O este reprezentata de DMA(Direct Memory Access) si se refera in principal la controlul schimbului de date intre memorie si dispozitivul de I/O. Pentru un transfer de date utilizand procesorul fara sa folosim un controler de DMA pot fi realizate foarte usor mici aplicatii in limbaj de asamblare in urma carora veti putea vedea ca viteza de acces la dispozitivul de I/O este total necorespunzatoare. Din aceasta cauza

s-a ajuns la utilizarea unui controler de DMA. Acesta acceseaza magistrala de date ceruta de la procesor iar ciclul instructie curent este suspendat urmand sa se transfere date cu dispozitivul de I/O in cadrul ciclului respectiv. In figurile 2 si 3 sint reprezentate controlerul DMA ca schema bloc si respectiv cum arat un ciclu de instructie si unde poate interveni DMA:

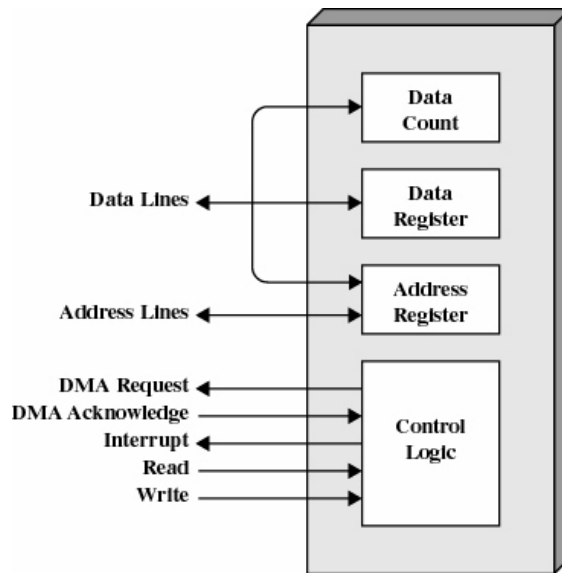


Figura 7.2. Logica DMA

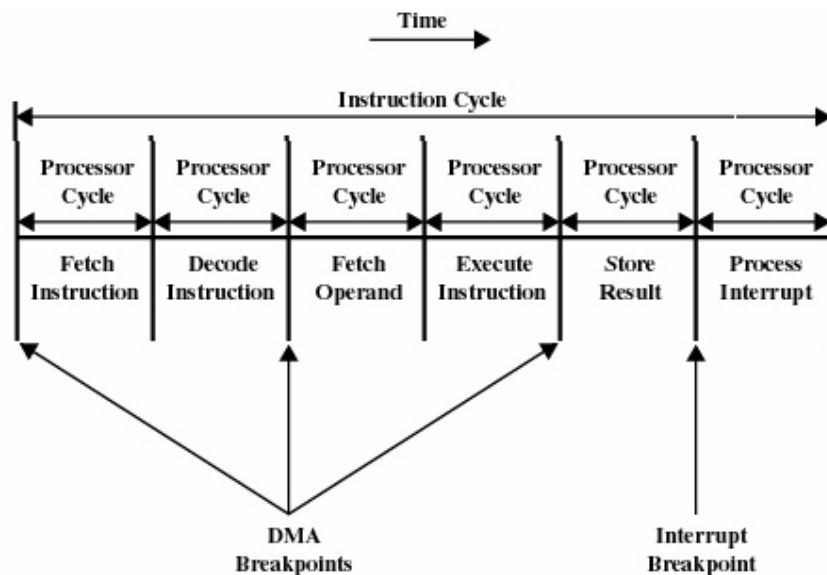


Figura 7.3. DMA si ciclul instructie

Principalele configuratii pentru plasarea controlerului de DMA le aveti in figura 4. Nu voi insista asupra lui pentru ca informatii detaliate le gasiti in cursul de microprocesoare.

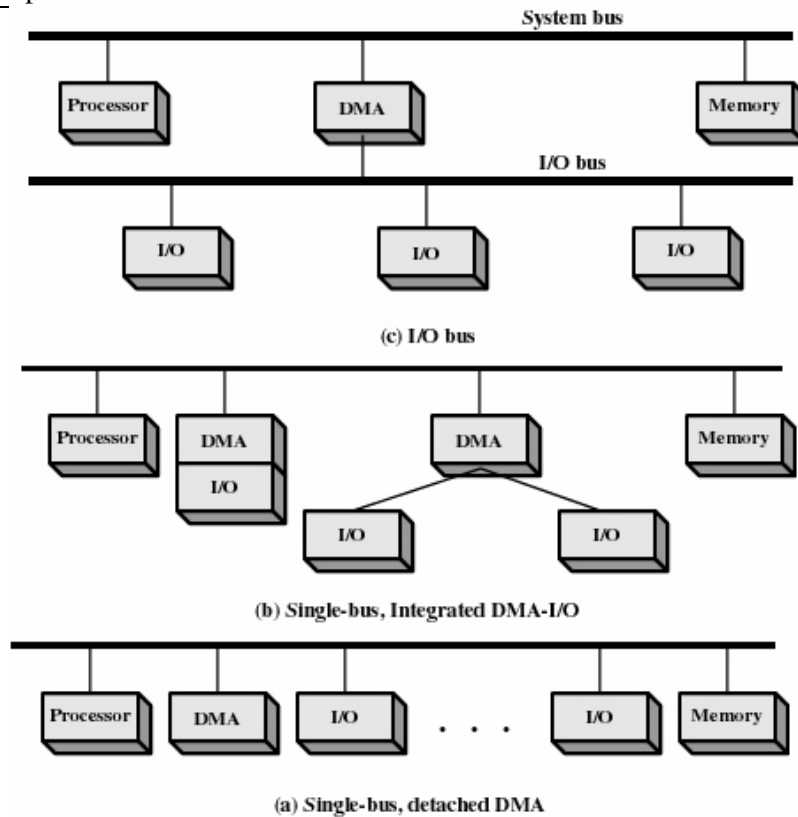


Figura 7.4. Configuratii alternative pentru DMA

### Structura logica a unei functii I/O

Vom considera 3 cazuri pentru dispozitivele I/O si variantele structurii logice in toate cele 3 cazuri. Cazurile sunt reprezentate sub forma de diagrama in figura 7.5. Primul caz, a), cel mai simplu, este cel al unui dispozitiv care comunica printr-o insiruire de bytes cu sistemul de calcul. Nivelele care ar trebui sa apara sunt urmatoarele:

- Modulul logic I/O : “discuta” cu dispozitivul ca o resursa logica In principal se ocupa de identificarea dispozitivului si operatii simple ca open, close, read write;
- Dispozitiv I/O : datele si operatiile solicitate sunt convertite in secvente de instructii pentru dispozitiv si comenzi pentru controler. Tehnica de buffer poate fi utilizata pentru imbunatatirea utilizarii;
- Dispecerizare si control: la acest nivel sunt utilizate intreruperile iar starile dispozitivului sunt raportate sistemului. La acest nivel se interactioneaza efectiv cu partea de hardware a dispozitivului.

Pentru un dispozitiv folosit la comunicatii am reprezentat diagrama din figura 7.5. b). Diferenta consta in faptul ca modulul logic este inlocuit de o arhitectura de comunicare, care poate fi constituita din mai multe straturi (exemplu OSI cu 7 straturi). In ultima figura este

reprezentat cazul managementului unui dispozitiv de stocare secundar. Cele 3 nivele care apar fata de primul caz sunt:

- Managementul directoarelor: la acest nivel numele simbolice sunt convertite in identificatori care referentiaza direct fisiere sau indirect printr-un descriptor sau index;
- Sistemul de fisiere: lucreaza cu structura logica a fisierelor si cu operatiile care pot fi specificate de useri cum ar fi: open, close, etc. Drepturile de acces sunt controlate tot la acest nivel;
- Organizarea fizica: referintele logice la fisiere sunt convertite la adrese fizice conforme cu cele folosite de catre dispozitiv.

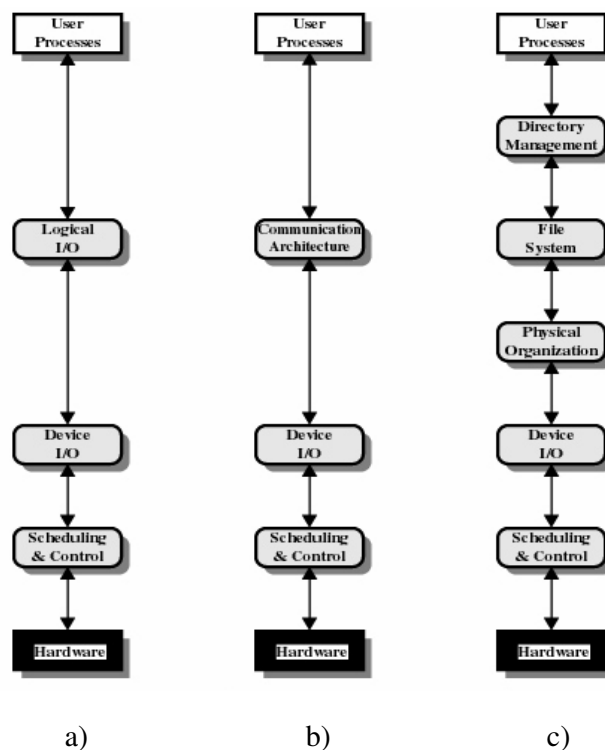


Figura 7.5. Modelul de organizare pentru dispozitive I/O

### Tehnica de buffering

Sa presupunem ca un proces utilizator doreste sa citeasca blocuri de date de la un dispozitiv de stocare secundar. Datele trebuie citite intr-o zona de date a procesului. Cel mai simplu mod este de a executa o comanda catre dispozitiv si sa asteptam ca datele sa devina disponibile. Ar putea aparea doua probleme importante in acest caz. Prima e ca procesul trece intr-o stare de asteptare pentru ca dispozitivul relativ incet de I/O sa furnizeze datele. A doua, si mai daunatoare, este ca aceasta asteptare poate interfera cu decizia sistemului de operare de

a face un swapping pentru acel proces. Zona de memorie trebuie insa sa ramana alocata in memoria principala pana la realizarea transferului de date. In cazul paginarii doar pagina tinta trebuie sa ramana in memoria principala. Exista si posibilitatea unui deadlock cand procesul este suspendat si mutat, in acesta caz el asteptand dupa dispozitivul de I/O iar acesta asteptand ca procesul sa fie reincarcat in memoria principala. Aceleasi consideratii pot fi facute si in cazul unei 'scrieri' de date catre un dispozitiv I/O.

Cele cateva moduri de rezolvare a problemei sunt prezentate in figura 7.6. Inainte de a le explica precizam ca transferul datelor de la/catre un dispozitiv de I/O se poate face orientat pe blocuri (cazul harddisk sau alte dispozitive de stocare secundare) sau pe streamuri (imprimante, porturi, terminale, mouse).

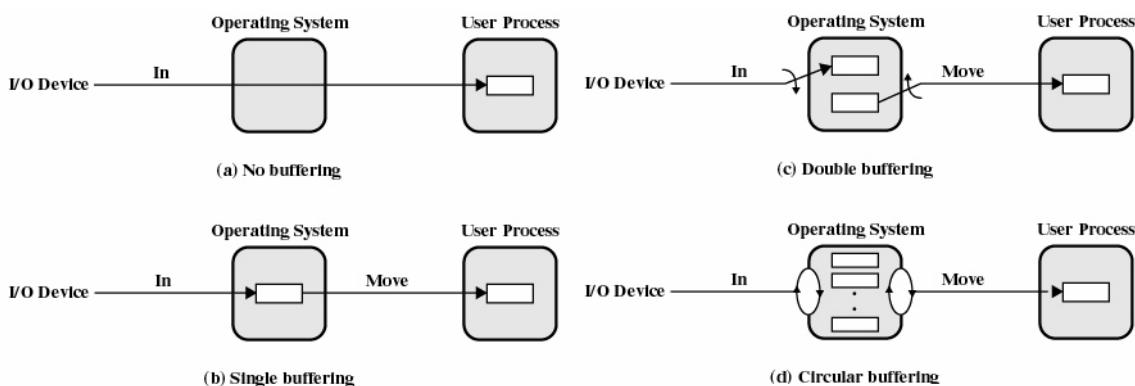


Figura 7.6. Scheme de buffering pentru dispozitivele I/O

Schemele de buffering prezentate sunt urmatoarele:

- Un singur buffer. In momentul in care un proces utilizator realizeaza o cerere catre un dispozitiv I/O, sistemul de operare pregateste un buffer pentru aceasta operatie in memoria principal (practic rezerva o mica portiune de memorie). Pentru cazul transferurilor orientate pe blocuri, dispozitivul executa transferul catre buffer iar in momentul in care transferul e complet, datele sunt mutate in spatiul utilizator si este solicitat un alt bloc de date. Aceasta abordare are ca rezultat o crestere a vitezei in raport cu cazul lipsei oricarui tip de buffer. Sistemul de operare poate face swap pentru proces deoarece operatia se executa in memoria principala si nu in memoria procesului utilizator. Tehnica bineinteles complica logica in cadrul sistemului de operare pentru ca trebuie pastrate date legate de asocierea buffer-proces. In cazul scrierii de blocuri consideratiile sunt similare. In cazul transferurilor bazate pe stream

se pot folosi linii de date care se pot termina prin CR si astfel bufferul poate fi utilizat pentru a stoca o singura linie.

- Buffer dublu. In figura 7.6. c puteti vedea exemplificata aceasta tehnică. Astfel procesul transfera date dintr-un buffer in timp ce alt bloc de date alimenteaza al doilea buffer. In cazul transferurilor bazate pe stream acest mod nu aduce imbunatatiri substantiale.
- Buffer circular. In ultima parte a figurii 7.6. am exemplificat un buffer circular. Poate fi utilizat cu succes in cazul dispozitivel de I/O cu o viteza foarte ridicata dar complica usor mecanismul.

Sistemul de buffering este foarte util in cazul sistemelor de operare de tip multitasking in care avem o varietate de dispozitive I/O si o varietate de procese.

### Managementul discurilor.

Pe parcursul dezvoltarii sistemelor de calcul s-a accentuat diferenta dintre viteza de lucru a procesorului si a memoriei si cea a dispozitivelor secundare de memorie cum ar fi harddisk-urile. In figura 7.7. am prezentat o diagrama de timp general in cazul lucrului cu un dispozitiv de stocare secundar de tip harddisk:

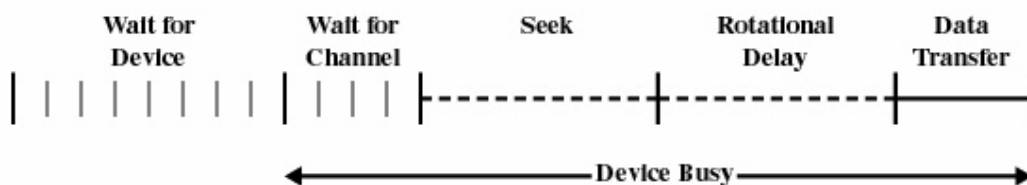


Figura 7.7. Diagrama temporală pentru lucrul cu un harddisk

Principali parametri care sunt luați în considerare în cazul dispozitivelor I/O de tip harddisk sunt:

- Timp de cautare. Este timpul necesar pentru mutarea la pista pe care se găsesc stocate datele. O formulă care poate caracteriza acest parametru este:  $T_s = m \times n + s$  în care:  $T_s$  reprezintă timpul de cautare estimat,  $n$  numărul de piste traversate,  $m$  o constantă care depinde de tipul diskului iar  $s$  întârzierea inițială;
- Întârzierea rotațională. Este întârzierea care apare față de numărul de rotații pe minut prevăzute.

- Timp de transfer.  $T = b/rN$  in care  $T$  reprezinta timpul de transfer,  $b$  numarul de octeti care este transferat,  $N$ , numarul de octeti de pe o pista iar  $r$  viteza de rotatie;

In continuare vom considera situatia tipica a unui mediu multitasking in care sistemul de operare intretine o coada de cereri pentru fiecare dispozitiv I/O. Dece pentru un singur harddisk vor fi un numar de cereri de la diverse procese aflate in coada. In cazul in care aceste procese sunt selectate intr-o ordine aleatoare, pisteles vor fi accesate aleator obtinand astfel cea mai proasta performanta.

Cea mai corecta abordare este cea de tipul FIFO in care cererile sunt satisfacute in ordinea in care se gasesc in coada respectiva. Aceasta poate functiona doar in cazul in care exista putine procese aflate in coada de asteptare. Complementara ei putem mentiona LIFO in care ultimul proces primeste controlul. Principala problema in acest caz este ca anumite procese pot ramane foarte mult in coada pana sunt satisfacute cererile dinaintea lor.

O alta tehnica folosita este SSTF (shortest service time first) care se bazeaza pe selectarea cererii care presupune cea mai mica miscare intre pisteles discului implicand cea mai mica durata pentru executarea operatiunii. Performantele sunt evident mai bune decat in celelalte doua cazuri. In cazul in care exista distante egale datorita faptului ca o pista se afla la aceeasi distanta fata de alte doua poate fi utilizat un mecanism aleator de selectie.

In cazul prezentat anterior exista posibilitatea ca anumite cereri sa nu fie satisfacute pana cand coada nu e goala. Pentru a elimina acest inconvenient a fost utilizat algoritmul SCAN. In acest caz citirea se face intr-o singura directie satisfacand toate cererile pana cand intalneste ultima pista in directia respectiva sau pana cand nu mai sunt cereri de rezolvat in acea directie. C-SCAN este o derivare a acesteia si cautarea se face intr-o singura directie. Astfel, cand se ajunge la ultima pista, cautarea este reincepta de la prima pista, in aceeasi directie.

In figura 7.8. sunt prezentate rezultatele in cazul celor 4 algoritmi mai importanti. Am considerat un disc cu 200 de piste. Cererile de piste s-au facut in urmatoarea ordine: 55, 58, 39, 18, 90, 160, 150, 38, 184. Se observa ca timpii obtinuti in cazul algoritmilor SSTF si SCAN sunt cei mai buni iar cel in cazul FIFO este cel mai slab.

Au mai fost dezvoltati si alti algoritmi pornind de la cei prezentati pentru diverse optimizari. Acestea se pot face cu complicarea mecanismului de acces la dispozitivul de stocare. Printre acestia mentionam variantele SCAN : FSCAN si N-step-SCAN mai cunoscute.

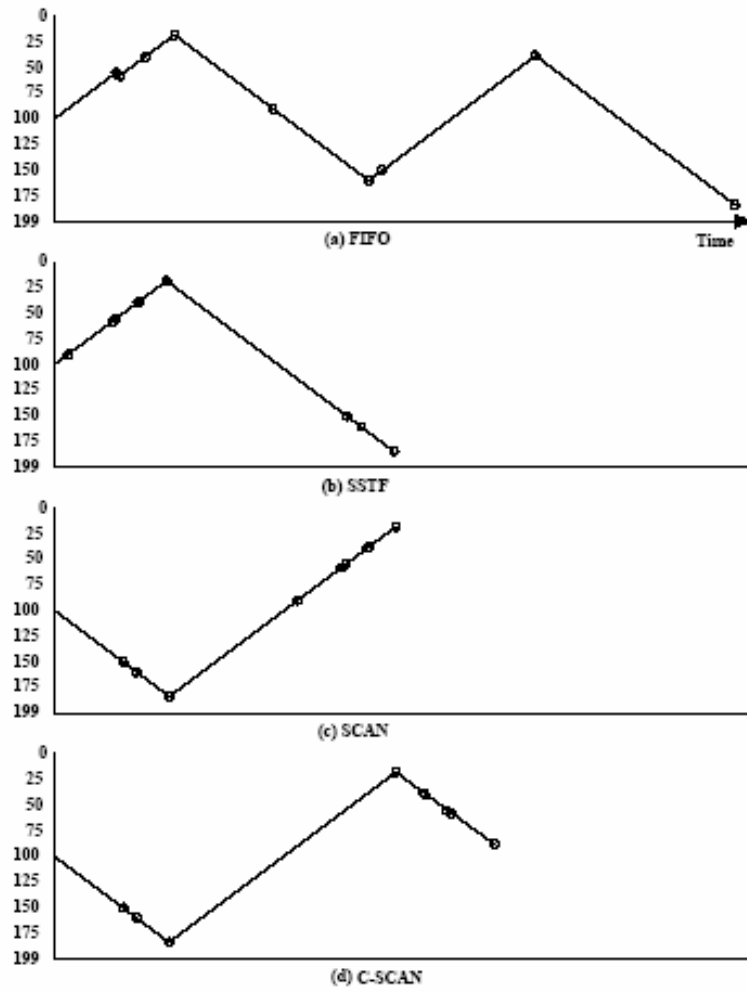


Figura 7.8. Comparatie intre algoritmi