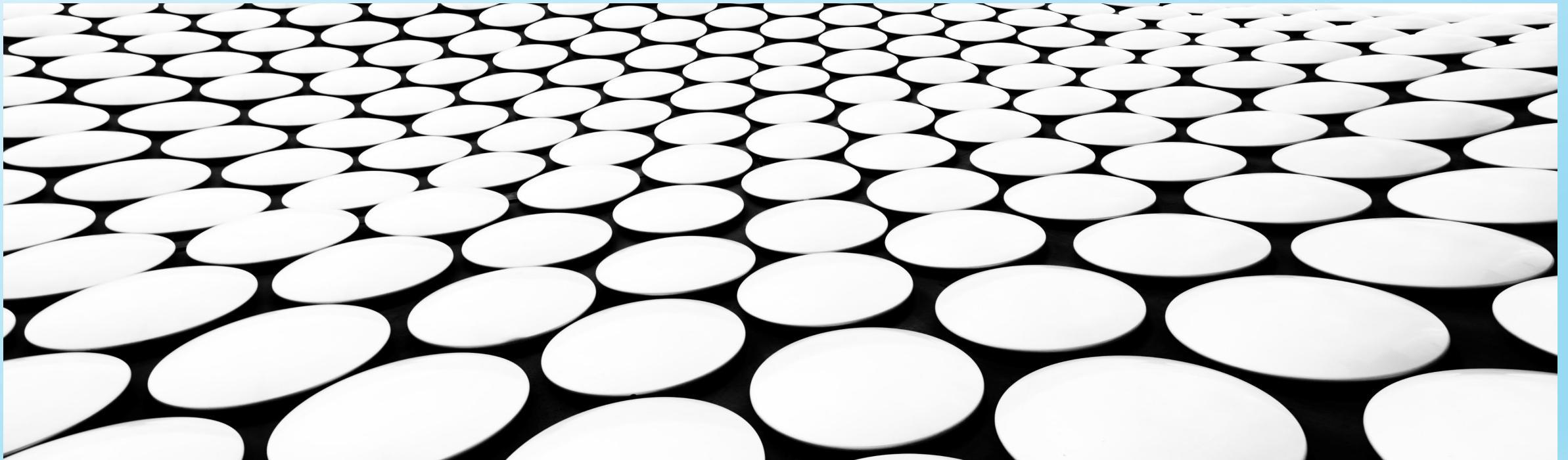

ARHITECTURA SISTEMELOR DE CALCUL

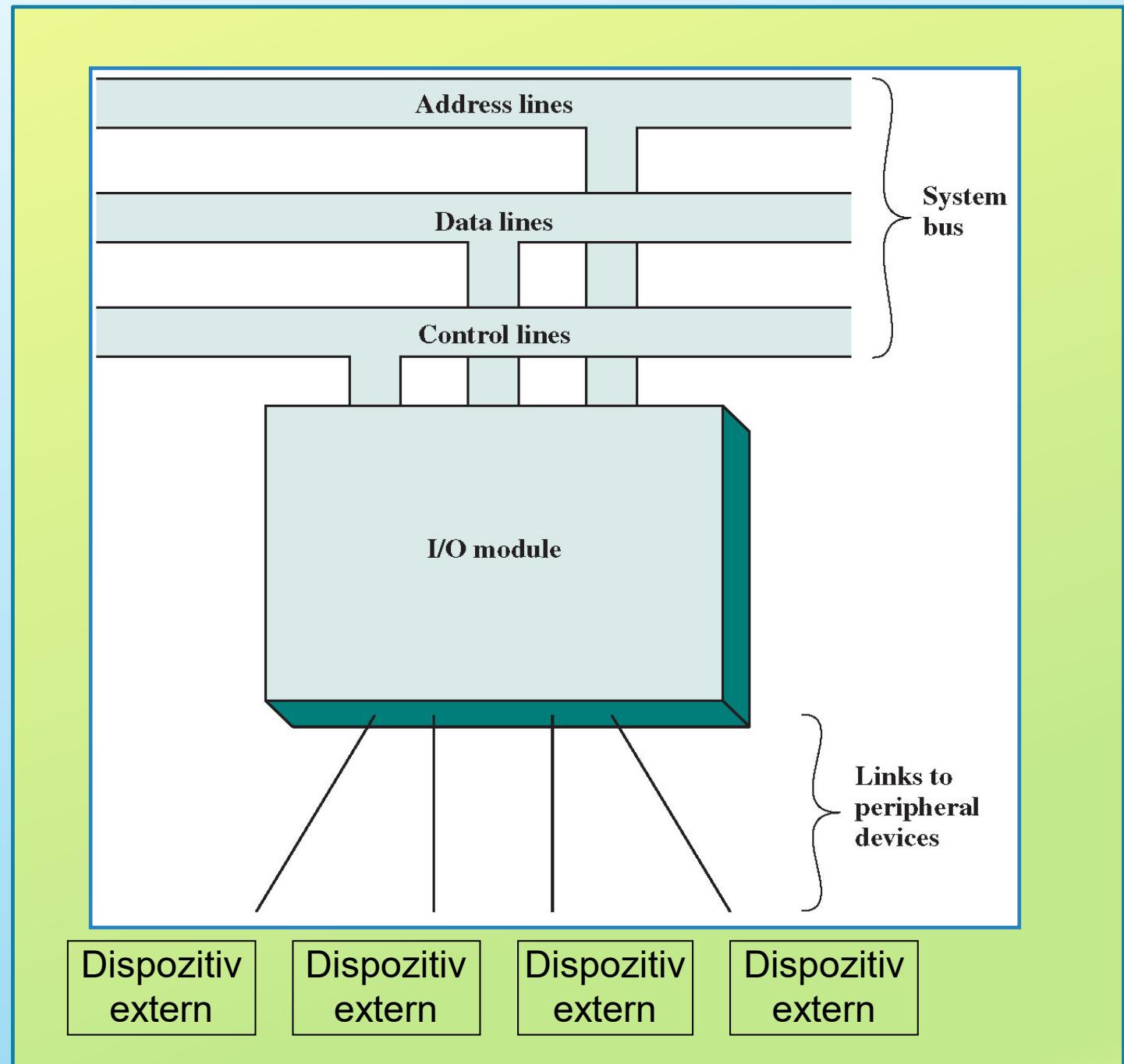
UB, FMI, CTI, ANUL III, 2022-2023



**BLOCURI
DE
INTRARE/IESIRE
(I/O)**

Modelul generic al unui bloc I/O

= modul I/O +
dispozitiv extern



Dispozitive externe

- Operațiile I/O sunt realizate printr-o gamă largă de dispozitive externe care asigura schimbul de date între mediul extern și computer.
- Un dispozitiv extern se atașează la computer printr-o legătură cu un modul I/O.
 - Legătura este utilizată pentru a transfera controlul, starea și datele între modulul **I/O** și dispozitivul extern.
 - Un dispozitiv extern conectat la un modul **I/O** este adesea denumit dispozitiv periferic sau pur și simplu periferic.

În general, putem clasifica **dispozitivele** externe în trei categorii:

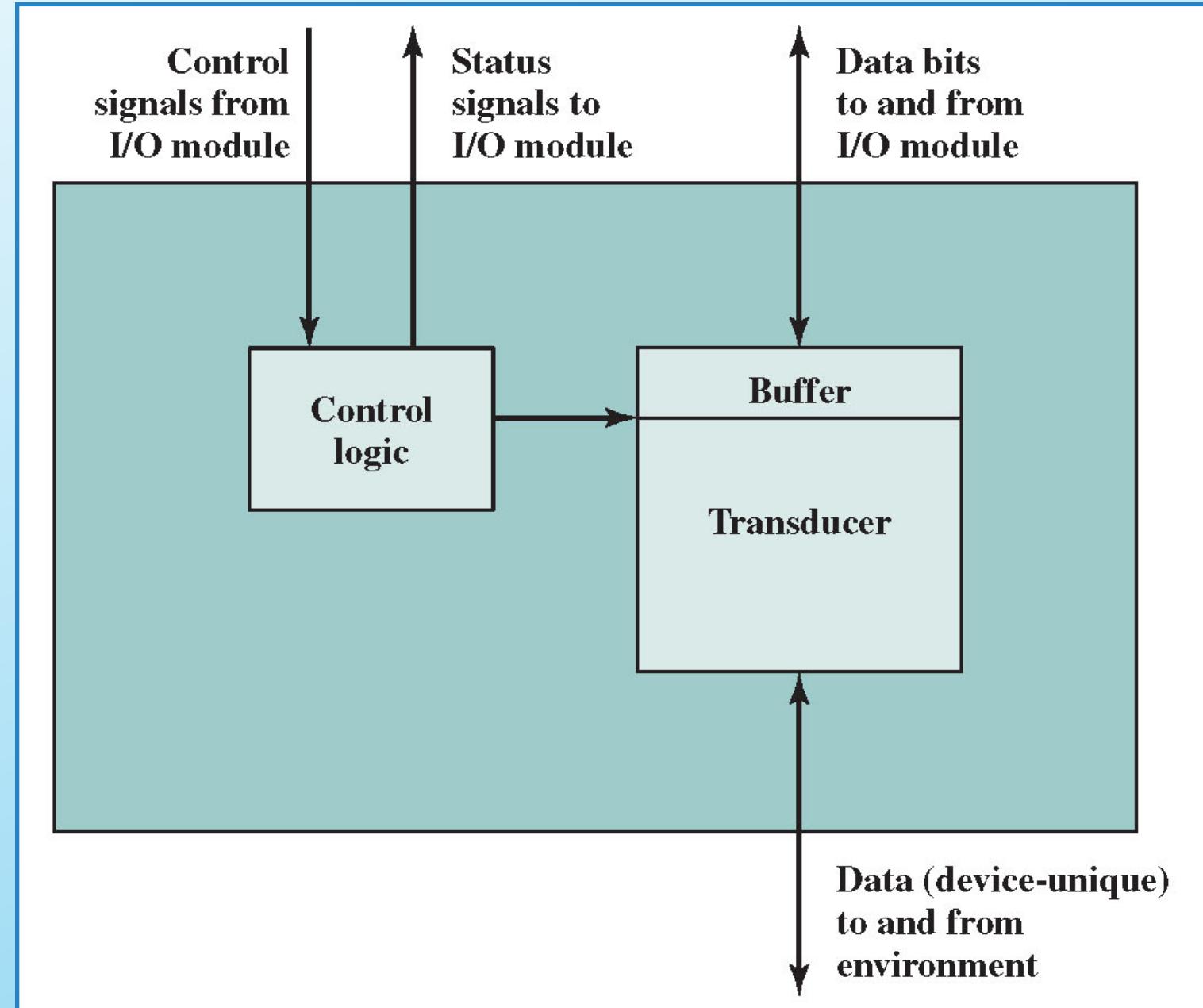
- **dispozitive care pot fi citite de om:** Adequate pentru comunicarea între utilizator si calculator
 - Video-terminal, imprimante
- **dispozitive care pot fi citite de masina:** specializate în comunicarea între echipamente si calculator
 - Discuri magnetice, senzori
- **dispozitive pentru comunicare:** Potrivite pentru comunicarea cu dispozitivele aflate la distanță

Structura unui dispozitiv extern

Blocul de control logic

Memorie tampon

Traductor



Interfața cu modulul I/O transmite semnale de control, stare și date.

- **Semnalele de control**

- determină operația pe care dispozitivul o va efectua:
 - trimitera datelor către modulul I/O (INPUT sau READ),
 - acceptarea datelor din modulul I/O (OUTPUT sau WRITE),
- indică starea rapoartelor
- comanda efectuarea unei anumite funcții de control la dispozitiv (de exemplu, poziționarea unui cap pe disc).

- **Semnalele de stare** indică starea dispozitivului.

- READY / NOT-READY pentru a arăta dacă dispozitivul este pregătit pentru transferul de date.
- **Datele** sunt grupate sub forma unor seturi de biți care trebuie trimise sau primite de la modulul I/O.

- **Blocul de control logic** controlează funcționarea dispozitivului ca răspuns la directiva primită de la modulul I/O.
- **Traductorul** convertește datele:
 - de la cele cu semnal electric (digital) la celelalte forme de semnal în timpul ieșirii și
 - de la alte forme de semnal la cele cu semnal electric (digital) în timpul intrării.
- **Memoria tampon** este asociată cu traductorul pentru a ține temporar datele transferate între modulul I/O și mediul extern; o dimensiune de buffer de 8 până la 16 biți este uzuala.

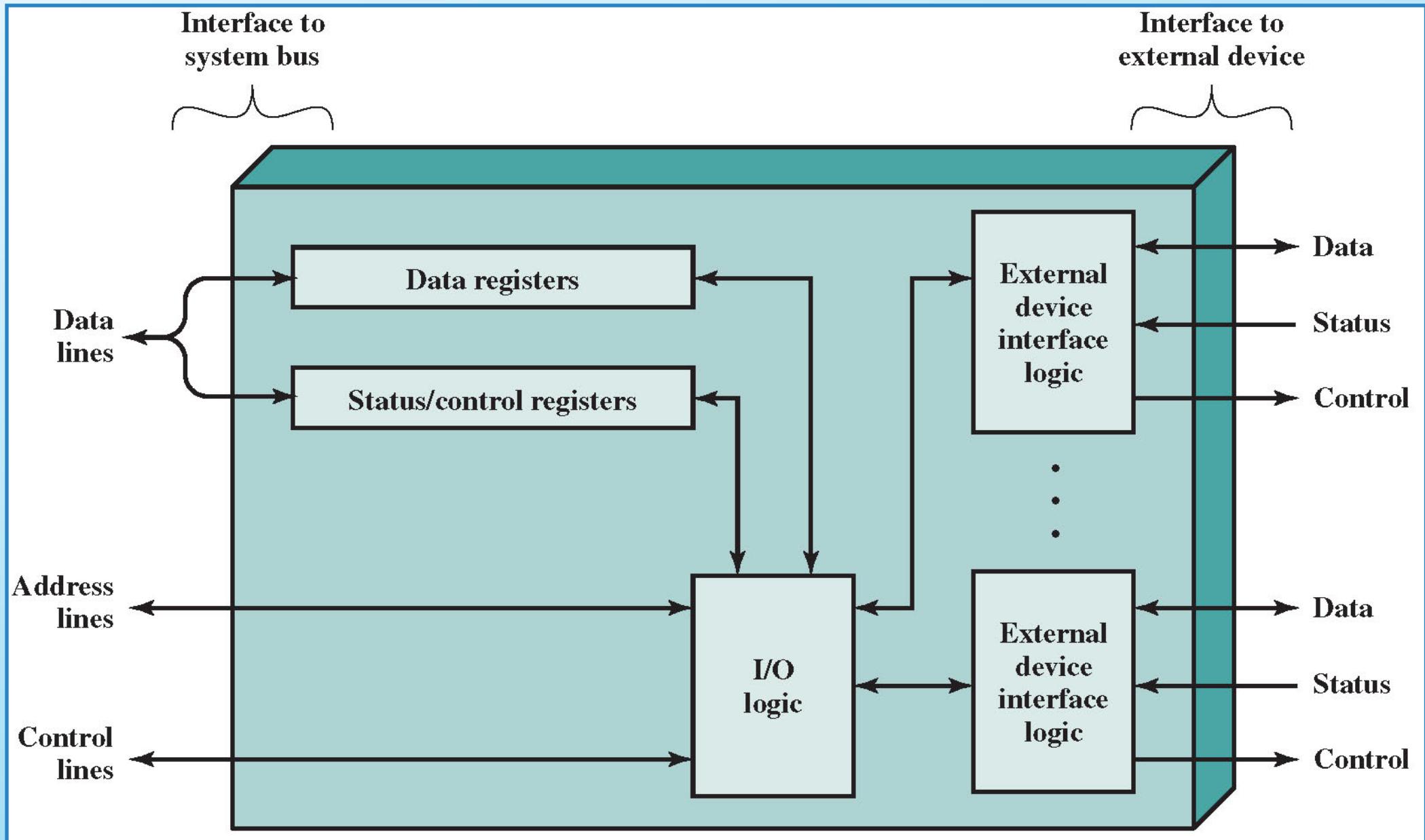
Module I/O

- Funcțiile modulului
 - Funcțiile sau cerințele majore pentru un **modul I/O** se încadrează în următoarele categorii:
 - Control și secvențiere
 - Comunicarea cu procesorul
 - Comunicarea cu dispozitivul periferic
 - Stocarea datelor în memorii tampon
 - Detectarea erorilor

Structura modulului I/O



Ex.:



- În orice moment de timp, procesorul poate (trebuie) să comunice cu unul sau mai multe dispozitive externe, în mod imprevizibil, dar în funcție de comenzi de I/O din program.
- Resursele interne, cum ar fi memoria principală și magistrala de sistem, trebuie partajate între mai multe activități, inclusiv transferul de date I/O.
- Astfel, *funcția I/O include o cerință de control și de sincronizare*, pentru a coordona fluxul de date între componente interne și dispozitivele externe.

Transferul de date **de la** un dispozitiv extern către procesor implica următoarea secvență de pași:

- 1.** Procesorul interoghează modulul I/O pentru a verifica starea dispozitivului atașat.
- 2.** Modulul I/O returnează starea dispozitivului extern.
- 3.** Dacă dispozitivul este operațional și este gata să transmită, procesorul solicită transferul de date prin intermediul unei comenzi către modulul I/O.
- 4.** Modulul I/O obține un calup de date (de exemplu, 8 sau 16 biți) de la dispozitivul extern.
- 5.** Datele sunt transferate de la modulul I/O la procesor.

Dacă sistemul utilizează o magistrală, atunci fiecare dintre interacțiunile dintre procesor și modulul I/O implică una sau mai multe arbitraje de magistrală.

Scenariul simplificat precedent ilustrează de asemenea faptul că modulul I/O trebuie să comunice **cu procesorul** și **cu dispozitivul extern**.

Comunicarea cu procesorul

Comenzi: Modulul I/O acceptă comenzi de la procesor, ca semnale pe magistrala de comandă.

- De exemplu, un modul I/O pentru o unitate de disc ar putea accepta următoarele comenzi: 'READ SECTOR', 'WRITE SECTOR', 'SEEK track number' și "CAN record ID". Ultimele două comenzi includ fiecare un parametru care este trimis pe magistrala de date.

Date: Datele sunt schimbate între procesor și modulul I/O pe magistrala de date.

Raportarea stării: deoarece perifericele sunt foarte lente, este important să se cunoască starea modulului I/O (respectiv a perifericului).

- De exemplu, dacă un modul de intrare / ieșire este rugat să trimită date procesorului (citire), este posibil să nu fie gata să facă acest lucru deoarece lucrează încă la o comandă I/O anterioară. Acest fapt poate fi raportat cu un semnal de stare.

Semnalele generale de stare sunt 'BUSY' și 'READY'. Pot să existe și semnale de raportare pentru diferite erori.

Recunoașterea adresei: Un modul I/O trebuie să recunoască cate o adresă unică pentru fiecare periferic pe care il controleaza.

Comunicarea cu dispozitivul extern

Această comunicare implică transfer de comenzi, informații despre stare și date

O sarcină esențială a unui modul I/O este **stocarea datelor în memorii tampon**. În timp ce rata de transfer catre sau dinspre memoria principală sau procesor este destul de mare, pentru multe dispozitive periferice rata este cu ordine de mărime mai mică și acoperă un domeniu larg.

- Datele provenite din memoria principală sunt trimise catre un modul I/O într-un ritm exploziv. Datele sunt stocate temporar în modulul I/O și apoi trimise la dispozitivul periferic într-o rata de transfer specifică.
- În direcția opusă, datele sunt stocate temporar astfel încât să nu oblige memoria la o operație de transfer lent. Astfel, modulul I/O trebuie să poată funcționa atât la viteze ale dispozitivului, cât și la memorie.
- În mod similar, dacă dispozitivul I/O funcționează cu o rată mai mare decât rata de acces la memorie, atunci modulul I/O efectuează operația de stocare temporara necesară.

- Un modul I/O este adesea responsabil pentru **detectarea erorilor** și pentru **răportarea lor** catre procesor.
 - O clasă de erori include **defecțiuni mecanice și electrice** raportate de dispozitiv (de exemplu, blocaj de hârtie, disc DVD defect).
 - O altă clasă constă în **modificări neintenționate ale informației**, în timpul transmisiei de la dispozitiv la modulul I/O. Unele forme de codificare permit detectarea erorilor de transmisie.
 - Un exemplu simplu este utilizarea unui bit de paritate pe fiecare caracter transmis. De exemplu, codul de caractere IRA ocupă 7 biți dintr-un octet. Al 8-lea bit este setat astfel încât suma totală a bitilor din byte să fie par (paritate pară) sau impară (paritate impară). Când un octet este primit, modulul I/O verifică paritatea pentru a determina dacă a apărut o eroare.

- Modulele I/O variază considerabil în ceea ce privește complexitatea și numărul de dispozitive externe pe care le controlează.
 - Modulul se conectează la restul calculatorului printr-un set de linii de semnal (de exemplu, linii de magistrala de sistem).
 - Logica din cadrul modulului interacționează cu procesorul printr-un set de linii de control. Procesorul utilizează liniile de control pentru a emite comenzi către modulul I/O.
 - Unele dintre liniile de control pot fi utilizate de către modulul I/O (de exemplu, pentru semnale de arbitraj și stare).
 - Datele transferate către și din modul sunt stocate în unul sau mai mulți registri de date.
 - Pot exista, de asemenea, unul sau mai multe registri de stare care furnizează informații despre starea curentă.
 - Un registru de stare poate funcționa, de asemenea, ca registru de control, pentru a accepta informații detaliate de control de la procesor.
 - De asemenea, modulul trebuie să poată recunoaște și genera **adrese asociate** cu dispozitivele pe care le controlează.
 - **Fiecare modul I/O are o adresă unică sau, dacă controlează mai multe dispozitive externe, un set unic de adrese.**
 - Modulul I/O conține logică specifică interfeței cu fiecare dispozitiv pe care îl controlează.

Un modul I/O funcționează permitând procesorului să vizualizeze o gamă largă de dispozitive **într-un mod simplu**.

- Modulul I/O poate ascunde detaliile de sincronizare, formate și de electromecanică ale unui dispozitiv extern astfel încât procesorul să poată funcționa în termeni simpli de comenzi de citire și scriere și, eventual comenzi de deschidere și închidere a fișierelor.
 - În forma sa cea mai simplă, modulul I/O poate lăsa încă o mare parte din munca de control (de exemplu, derularea inversă a benzii) vizibilă procesorului.
- Un modul I/O avansat, care preia cea mai mare parte a operațiilor detaliate de procesare, prezintând o interfață de nivel înalt procesorului, este denumit de obicei un **canal I/O** sau un **procesor I/O**.
- Un modul I/O care este destul de primitiv și necesită un control detaliat este denumit de obicei un **controler I/O** sau un **controler de dispozitiv**.
 - Controlerele I/O sunt frecvent observate pe microcomputere, în timp ce canalele I/O sunt utilizate pe mainframe.

■ Sunt posibile trei tehnici de operare I/O.

- **operarea I/O programata:** datele sunt schimbată între procesor și modulul I/O. Procesorul execută un program care îi oferă controlul direct al operației I/O, incluzând sesizarea stării dispozitivului, trimiterea unei comenzi de citire sau scriere și transferarea datelor.
 - Când procesorul emite o comandă către modulul I/O, trebuie să aștepte până la terminarea operației I/O.
Dacă procesorul este mai rapid decât modulul I/O, apare o pierdere de timp procesor.
- **operarea I/O cu întreruperi:** procesorul emite o comandă I/O, continuă să execute alte instrucțiuni și este **întrerupt** de modulul I/O când acesta din urmă și-a încheiat activitatea.
 - În operarea **I/O programata și cu întreruperi**, procesorul este responsabil:
 - cu extragerea datelor din memoria principală pentru ieșire și
 - cu stocarea datelor în memoria principală pentru intrare.
 - A treia alternativa de operare este cunoscută sub denumirea de **acces direct la memorie (DMA)**. În acest mod, modulul I/O și memoria principală schimbă datele direct, fără implicarea procesorului.

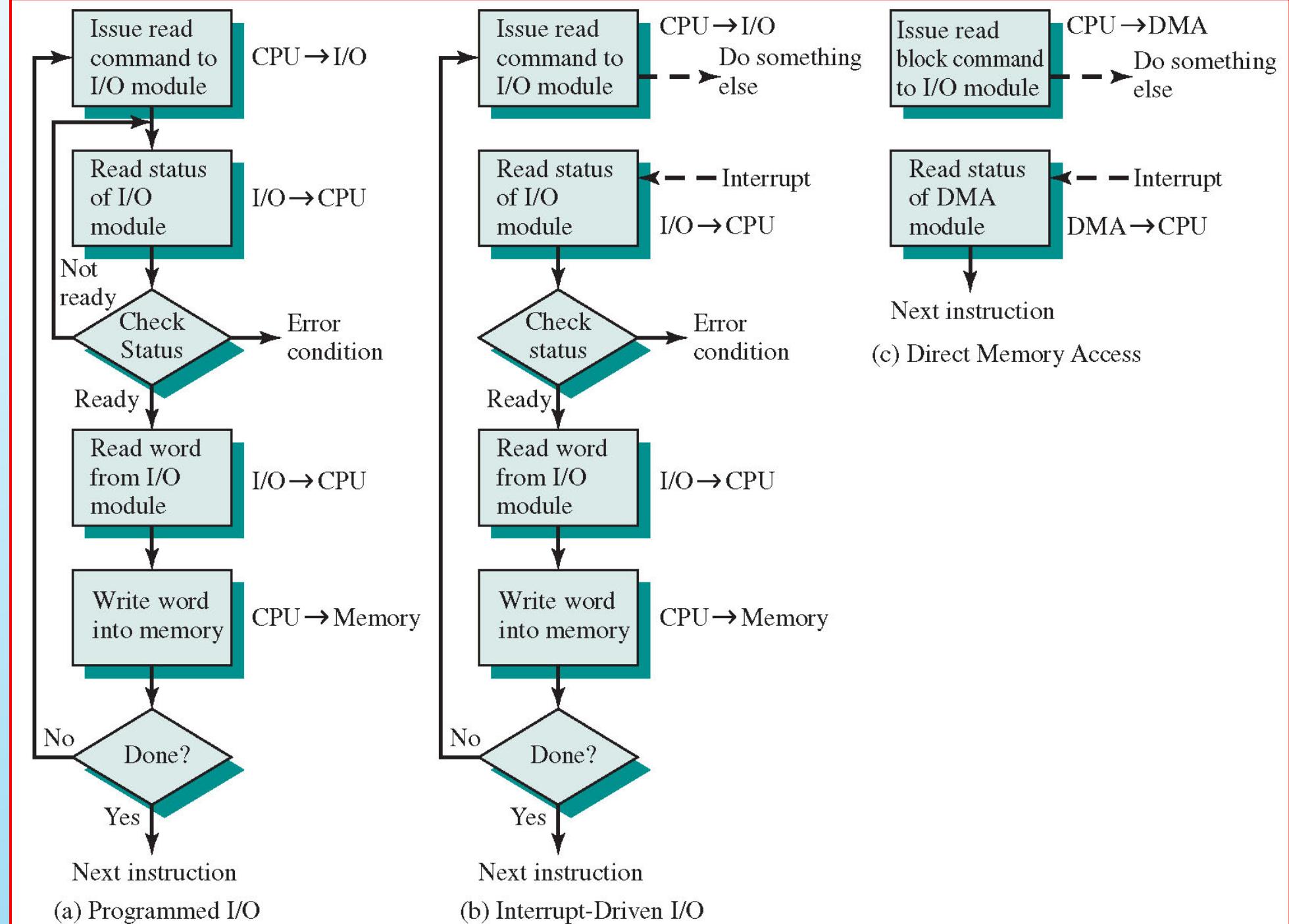
Operatiuni I/O programate

- Când procesorul execută un program și întâlnește o instrucțiune referitoare la I/O, execută acea instrucțiune prin emiterea unei comenzi către modulul I/O corespunzător.
 - Cu I/O programate, modulul I/O va efectua acțiunea solicitată și apoi va seta biții corespunzători în registrul de stare I/O.
 - Modulul I/O nu mai întreprinde acțiuni de alertare a procesorului. În special, **nu îintrerupe procesorul**.
 - Astfel, este responsabilitatea procesorului să verifice periodic starea modulului de intrare / ieșire până când constată că operația este completă.

Comenzi I/O

- Pentru a executa o instrucțiune referitoare la I/O, procesorul emite
 - o adresă, specificând modulul I/O și dispozitivul extern și
 - o comandă I/O.
- Există patru **tipuri** de comenzi I/O pe care un modul I/O le poate primi atunci când este adresat de un procesor:
 - **de Control:** Se utilizează pentru a activa un dispozitiv periferic și pentru a-i comunica ce să facă.
 - De exemplu, o unitate de bandă magnetică poate fi instruită pentru a derula înapoi sau în avans avans o banda. Aceste comenzi sunt adaptate tipului particular de dispozitiv periferic.
 - **de Test:** Folosit pentru a testa diferitele condiții de stare asociate cu un modul I/O și perifericele acestuia.
 - Procesorul va dori să știe că **perifericul** de interes este alimentat și **disponibil** pentru utilizare.
 - De asemenea, va dori să știe **dacă operațiunea I/O** cea mai recentă **este finalizată** și dacă au apărut erori.
 - **de Citire:** determină modulul I/O să obțină un calup de date de la periferic și să îl plaseze într-un buffer intern.
 - Procesorul poate obține apoi calupul de date solicitând ca modulul I/O să îl plaseze pe magistrala de date.
 - **de Scriere:** Obliga modulul I/O să preia un calup de date (octet sau cuvânt) din magistrala de date și apoi să transmită acel element de date către periferic.

Figura 1



- Figura 1a prezintă un exemplu de utilizare a I/O programate pentru citirea într-un bloc de date de la un dispozitiv periferic (de exemplu, o înregistrare din bandă) cu scriere în memorie.
 - Datele sunt citite sub forma unui singur cuvânt (ex.: 16 biți) la un moment dat.
 - Pentru fiecare cuvânt citit, procesorul trebuie să rămână într-un ciclu de verificare a statutului până când determină că cuvântul este disponibil în registrul de date al modulului I/O.
 - Această diagramă evidențiază principalul dezavantaj al acestei tehnici: este un proces consumator de timp care menține procesorul ocupat inutil.

Instructiuni I/O

- În cazul I/O programate, există o corespondență strânsă între instrucțiunile I/O pe care procesorul le extrage din memorie și comenzi I/O pe care procesorul le emite către un modul I/O pentru a executa instrucțiunile.
 - Instrucțiunile sunt ușor de tradus în comenzi I/O, și există adesea o relație unu-la-unu simplă. Forma instrucțiunii depinde de modul în care sunt abordate dispozitivele externe.
- Există multe dispozitive I/O conectate prin intermediul modulelor I/O la sistem.
 - Fiecare dispozitiv are un identificator sau o adresă unică. Când procesorul emite o comandă I/O, comanda conține adresa dispozitivului dorit.
 - Fiecare modul I/O trebuie să interpreteze liniile de adresă pentru a determina dacă comanda este pentru el însăși.

- Când procesorul, memoria principală și I/O partajeaza acceasi magistrala, sunt posibile două moduri de adresare:
 - **memorie mapată**
 - **memorie izolată.**
- Pentru **I/O cu memorie mapata**, există un singur spațiu de adrese pentru locațiile de memorie și dispozitivele I/O.
 - Procesorul tratează registrele de stare și date ale modulelor I/O ca locații de memorie și utilizează aceleași instrucțiuni pentru a accesa atât dispozitivele de memorie, cât și dispozitivele I/O.
 - De exemplu, o magistrala cu 10 linii de adrese (10 biti), poate adresa un total (combinat) de $2^{10} = 1024$ de locații de memorie și adrese I/O, în orice combinație.
 - Pentru I/O cu memorie mapata, sunt necesare: o singură linie de citire și o singură linie de scriere pe magistrala

I/O cu memorie izolata

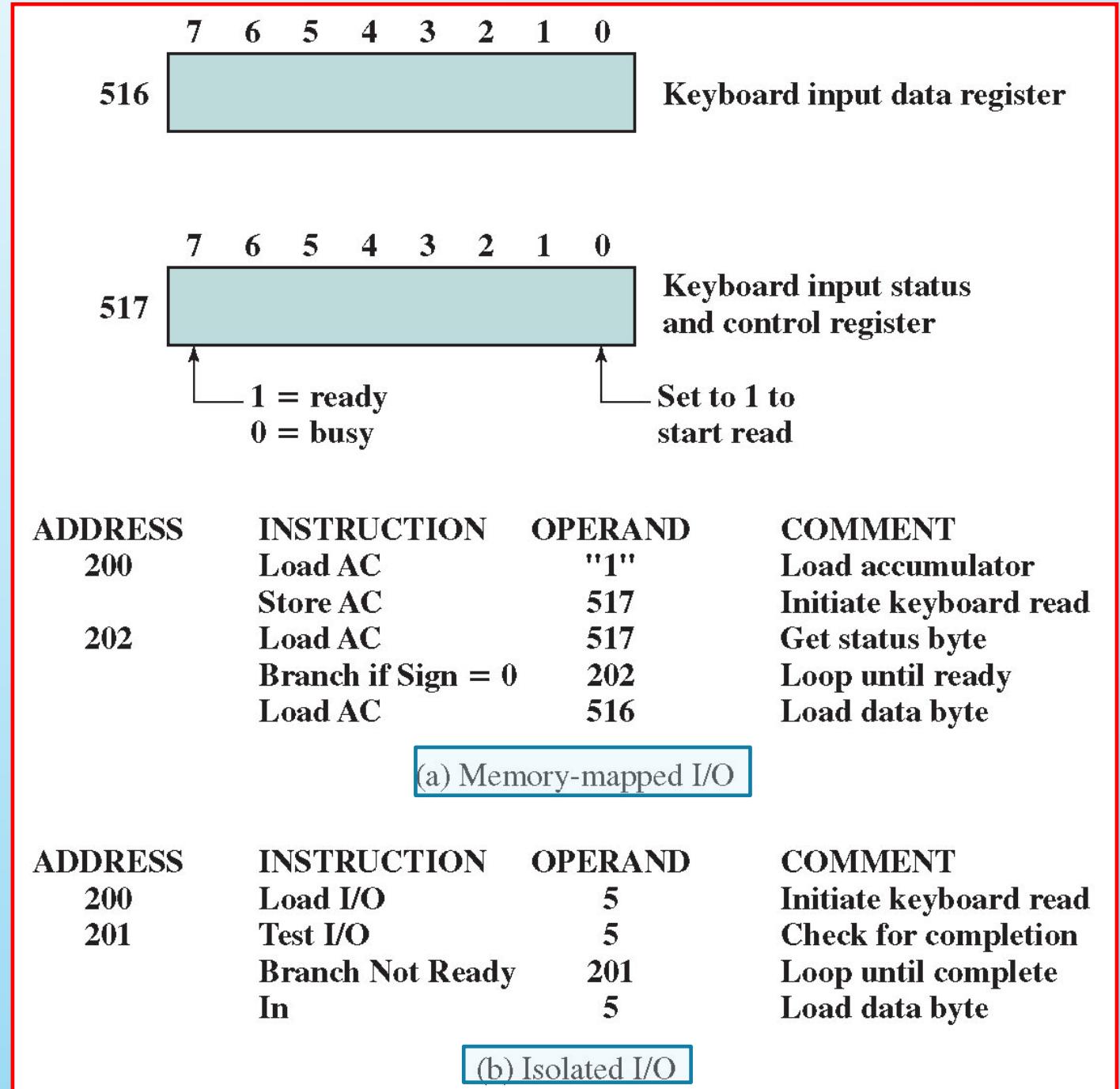
- Alternativ, magistrala poate fi echipată cu: linii de comandă de citire și scriere în memorie, plus linii de comandă de intrare și ieșire.
 - Acum, linia de comandă specifică dacă adresa se referă la o locație de memorie sau la un dispozitiv I/O.
 - Gama completă de adrese este disponibilă pentru ambele.
 - Din nou, cu 10 linii de adrese, sistemul poate suporta acum atât 1024 de locații de memorie, cât și 1024 adrese I/O.
- Deoarece spațiul de adrese pentru I/O este izolat de cel pentru memorie, acest lucru este denumit **I/O izolat**.

Exemplu

Figura (care urmeaza) compara aceste două tehnici I/O programate.

- Figura 2a arată modul în care interfața pentru un dispozitiv de intrare simplu, cum ar fi o tastatură, poate apărea unui programator care utilizează I/O cu memorie mapata.
 - Să presupunem o adresă pe 10 biți, cu memorie de 512-biți (locații 0-511) și până la 512 adrese I/O (locații 512-1023).
 - Două adrese sunt dedicate intrării de la tastatură. Adresa 516 se referă la registrul de date, iar adresa 517 se referă la registrul de stare, care funcționează și ca registrul de control pentru primirea comenziilor procesorului.
 - Programul prezentat va citi 1 byte de date de la tastatură și îl va introduce într-un registru acumulator din procesor. Rețineți că procesorul ramane în bucla până când octetul de date este disponibil.

Figura 2



- Pentru I/O cu memorie izolata (**Figura 2 b**), porturile I/O sunt accesibile numai prin comenzi I/O speciale, care activează liniile de comandă I/O din magistrală.
- Pentru majoritatea tipurilor de procesoare, există un set relativ mare de instrucțiuni diferite pentru a adresa memoria. Dacă se utilizează I/O cu memorie izolată, există doar câteva instrucțiuni I/O.
- Astfel, un avantaj al I/O cu memorie mapată este că acest repertoriu mare de instrucțiuni poate fi folosit, permitând o programare mai eficientă.
- Un dezavantaj este că spațiul valoros de adrese de memorie este blocat. Atât I/O cu memorie mapată, cât și cele memorie izolată cu sunt utilizate în mod obișnuit.

Operațiuni I/O cu intrerupere

- Problema asociată cu I/O programată este că procesorul trebuie să aștepte mult timp modulul de I/O care este ocupat cu receptia sau transmiterea datelor.
 - Procesorul, în timp ce așteaptă, trebuie să interogheze în mod repetat starea modulului I/O.
 - Ca urmare, nivelul performanței întregului sistem este puternic degradat.
- O alternativă este ca procesorul să emită o comandă de I/O către modul și apoi să continue să facă alte activități utile.
- Modulul I/O va îintrerupe apoi procesorul pentru a solicita serviciul atunci când este gata să facă schimb de date cu procesorul.
- Procesorul execută apoi transferul de date, ca mai înainte, și apoi își reia prelucrarea anterioară.

Actiunea modulului I/O.

- Pentru operatiunea de intrare, modulul I/O primește o comandă READ de la procesor.
 - Modulul I/O va citi (in consecinta) date de la un periferic asociat.
- Odată ce datele se află în registrul de date al modulului, modulul semnalează printr-o îñtrerupere pe o linie de control spre procesor.
- Modulul așteaptă până când datele sale sunt solicitate de către procesor.
 - Când CPU face cererea, modulul își pune datele pe magistrala de date și este apoi pregătit pentru o altă operație I/O.

Acțiunea procesorului.

- Procesorul emite o comandă READ.
 - Apoi se opreste și face altceva (procesorul poate lucra la mai multe programe diferite în același timp).
- La sfârșitul fiecărui ciclu de instrucțiune, procesorul verifică întreruperile
- Când se produce întreruperea de la modulul I/O, procesorul salvează contextul (de exemplu, registrul de programe și registrul procesorului) programului curent și procesează întreruperea.
 - În acest caz, procesorul citește cuvântul de date din modulul I/O și îl stochează în memorie.
- Apoi restabilește contextul programului pe care lucra (sau alt program) și reia execuția.

Figura 1b prezintă utilizarea interogării I/O pentru citirea într-un bloc de date.

- I/O cu întrerupere este mai eficientă decât I/O cu programare, deoarece elimină aşteptarea inutilă.
- I/O cu întrerupere consumă totusi mult timp procesor, deoarece fiecare cuvânt de date care trece de la memorie la modulul I/O sau de la modulul I/O la memorie trebuie să treacă prin procesor.

Accesul direct la memorie (DMA)

Dezavantaje ale I/O programate și întrerupte

- I/O cu întreruperi, deși mai eficient decât I/O programat, necesită intervenția activă a procesorului pentru a transfera date între memorie și un modul I/O, iar orice transfer de date trebuie să traverseze o cale prin procesor.
- Astfel, ambele forme de I/O suferă de două dezavantaje inerente:
 - 1. Rata de transfer I/O este limitată de viteza cu care procesorul poate testa și servi un dispozitiv.
 - 2. Procesorul este legat la gestionarea unui transfer I/O; un număr de instrucțiuni trebuie executate pentru fiecare transfer I/O

Există un fel compromis între aceste două dezavantaje.

- Sa luam în consideratie transferul unui bloc de date.
 - Folosind I/O programate, procesorul este dedicat sarcinii I/O și poate muta date cu o rată destul de ridicată, cu costul de a nu face altceva.
 - Întreruperea I/O eliberează procesorul într-o oarecare măsură în detrimentul ratei de transfer I/O.
 - Cu toate acestea, ambele metode au un impact negativ atât asupra activității procesorului, cât și asupra ratei de transfer I/O.

Atunci când trebuie mutate volume mari de date, este necesară o tehnică mai eficientă: accesul direct la memorie (DMA).

Functia DMA

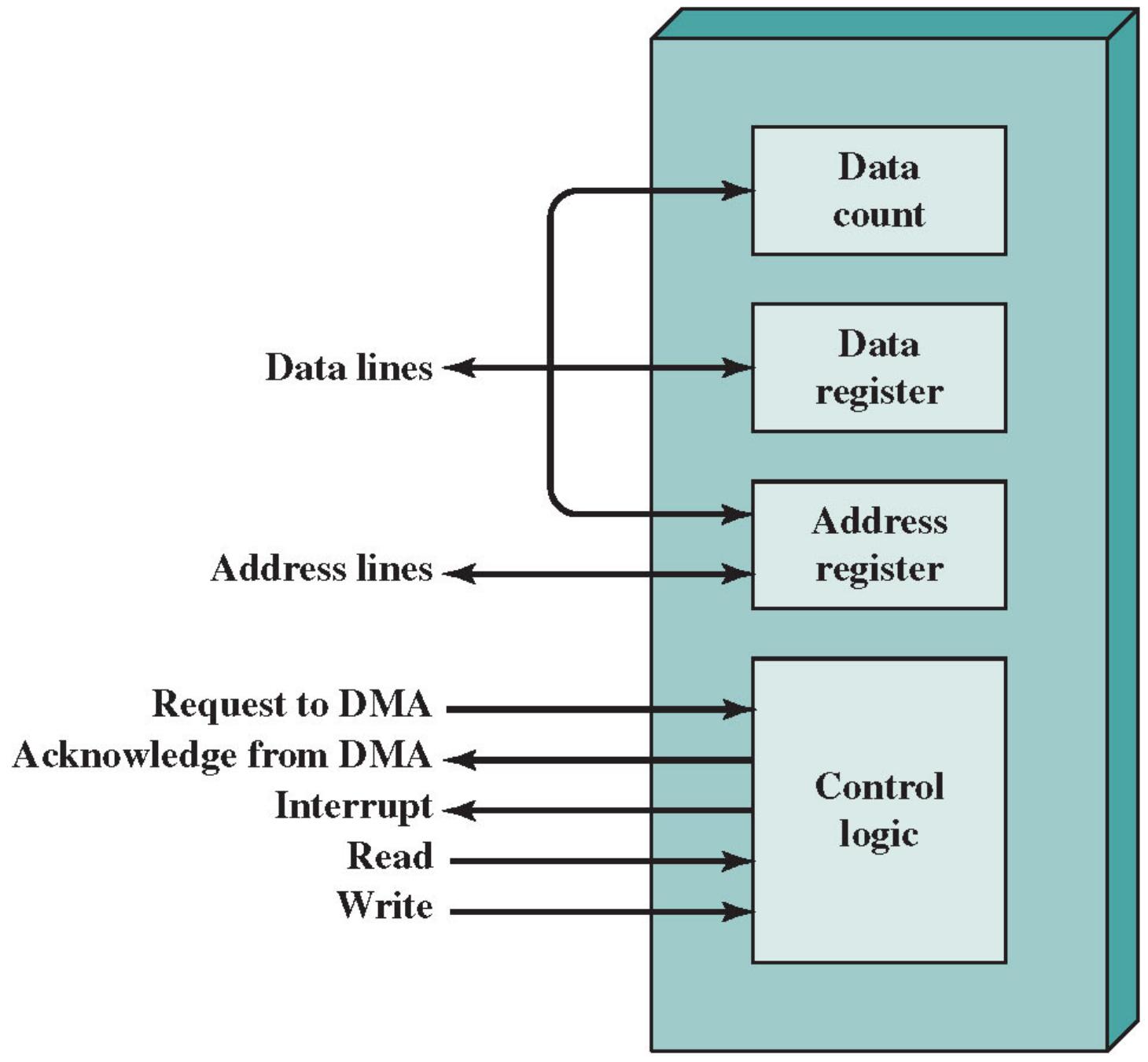
DMA implică existența unui modul suplimentar pe magistrala de sistem.

- **Modulul DMA** este capabil să imită procesorul și să preia controlul asupra sistemului de la procesor.
 - Trebuie să facă acest lucru pentru a transfera date către și din memorie prin magistrala de sistem.
 - În acest scop, modulul DMA
 - trebuie să utilizeze magistrala numai atunci când procesorul nu are nevoie de ea sau
 - trebuie să forțeze procesorul să își suspende temporar funcționarea.
 - Ultima tehnică este mai frecventă și este cunoscută ca furt de ciclu, deoarece modulul DMA în mod efectiv fură un ciclu de magistrala.

Când procesorul dorește să citească sau să scrie un bloc de date, emite o comandă către modulul DMA, trimițând către modulul DMA următoarele informații:

- Dacă se solicită citirea sau scrierea, folosind linia de control pentru citire sau scriere între procesor și modulul DMA
- Adresa dispozitivului I/O implicat, comunicat pe liniile de date
- Locația de pornire din memorie pentru citirea sau scrierea, comunicarea pe liniile de date și stocata de modulul DMA în registrul său de adrese
- Numărul de cuvinte care trebuie citite sau scrise, comunicate din nou prin linii de date și stocate în registrul de numărare a datelor

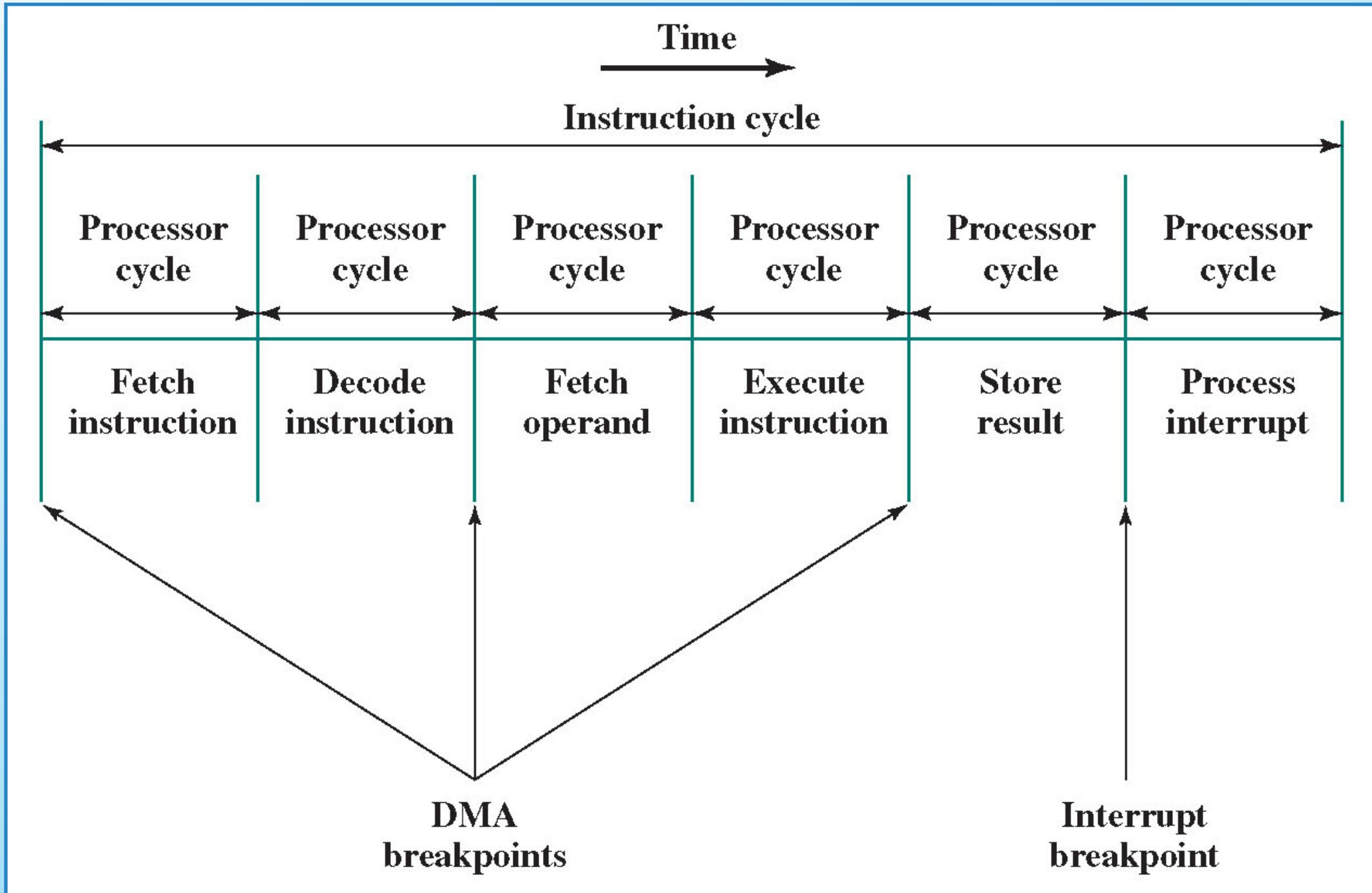
Figura 3



Procesorul continuă apoi cu alte lucrări.

- A delegat această operație I/O catre modulul DMA.
- Modulul DMA transferă întregul bloc de date, câte un cuvânt la un moment dat, direct către sau din memorie, fără a trece prin procesor.
- Când transferul este finalizat, modulul DMA trimite un semnal de întrerupere procesorului.
- Astfel, procesorul este implicat doar la începutul și la sfârșitul transferului.

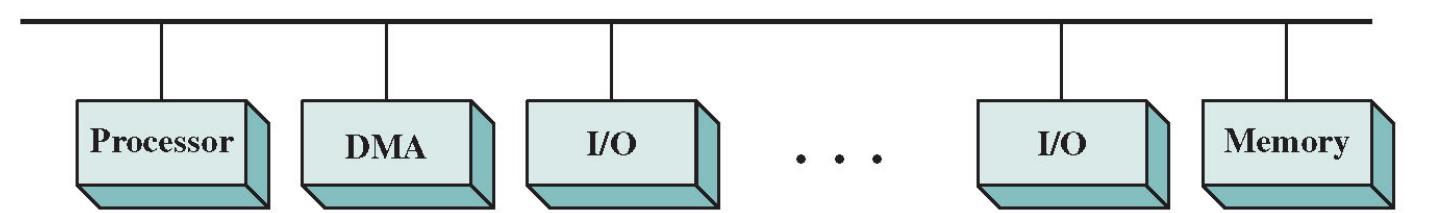
Figura 4



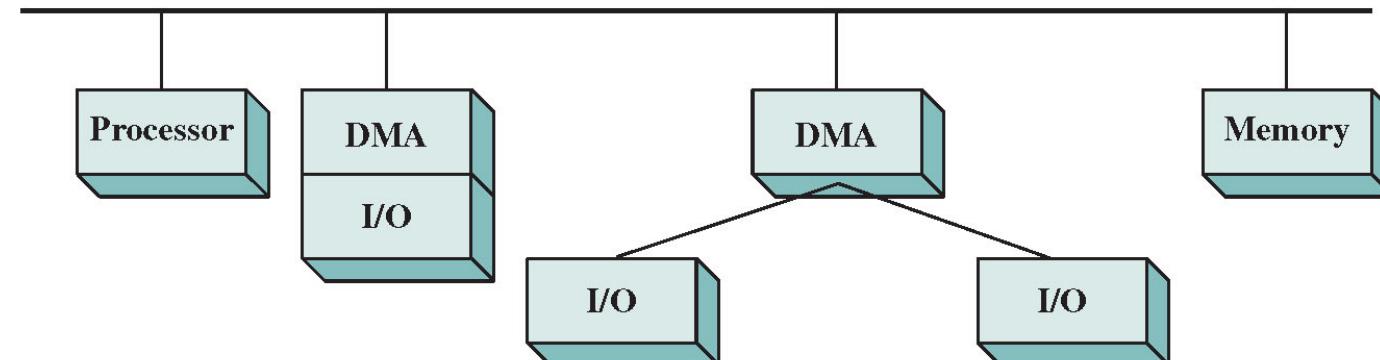
- Figura 4 arată unde, în ciclul instrucțiunii, procesorul poate fi suspendat.
 - În fiecare caz, procesorul este suspendat chiar înainte de a folosi magistrala.
- Modulul DMA transferă apoi un cuvânt și readuce controlul la procesor.
 - **Rețineți că aceasta nu este o întrerupere; procesorul nu salvează un context și nu face altceva.**
 - procesorul se oprește pentru un ciclu de magistrală.
- Efectul global este ca procesorul executa operațiile mai lent.
 - Pentru un transfer I/O cu mai multe cuvinte, DMA este mult mai eficient decât I/O programate sau cu întreruperi

- Mecanismul DMA poate fi configurat într-o varietate de moduri.
 - Unele posibilități sunt prezentate în Figura 5.
- În primul exemplu, toate modulele partajează aceeași magistrală de sistem.
 - Modulul DMA, care acționează ca un procesor surrogat, utilizează I/O programat pentru a schimba date între memorie și un modul I/O prin modulul DMA.
 - Această configurație, deși poate fi ieftină, este în mod clar ineficientă.
 - Ca și în cazul I/O programate controlate de procesor, fiecare transfer al unui cuvânt consumă două cicluri de magistrală. (un transfer memorie-DMA și un transfer DMA-I/O)

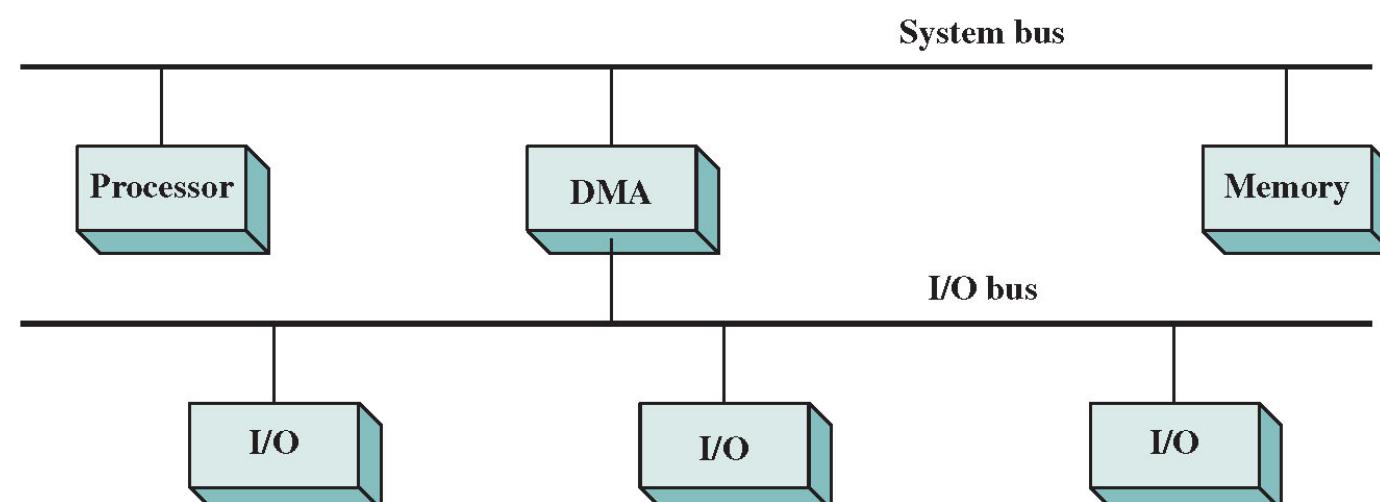
Figura 5



(a) Single-bus, detached DMA



(b) Single-bus, integrated DMA-I/O



(c) I/O bus

- Numărul necesar de cicluri de magistrala poate fi redus substanțial prin integrarea funcțiilor DMA și I/O.
 - După cum indică **Figura 5b**, aceasta înseamnă că există o cale între modulul DMA și unul sau mai multe module I/O care nu include magistrala de sistem.
 - Logica DMA poate fi de fapt o parte a unui modul I/O sau poate fi un modul separat care controlează unul sau mai multe module I/O.
- Acest concept poate fi preluat cu un pas prin conectarea modulelor I/O la modulul DMA folosind o magistrală I/O (**Figura 5c**).
 - Aceasta reduce numărul de interfețe I/O din modulul DMA la una și asigură o configurație ușor de expandat.
 - În ambele cazuri (**Figurile 5b și 5c**), magistrala de sistem pe care modulul DMA o partajează cu procesorul și memoria este folosită de modulul DMA numai pentru a schimba datele cu memoria.
 - Schimbul de date între modulele DMA și I/O are loc în afara magistralei de sistem.

