CURSUL 13

SISTEMUL DE INTRARE - IEŞIRE

Conţinut:

I. CIRCUITE DE INTERFAŢĂ

Subsistemul de intrare-ieşire (I/O) al unui calculator asigură calea de comunicaţie a informaţiilor între calculator şi mediul extern (logica externă care nu lucrează direct cu UCP). Un sistem de calcul de uz general, nu are utilitate practică dacă nu poate recepţiona şi transmite informaţii cu exteriorul, într-o formă accesibilă utilizatorului uman. Prin intermediul acestui subsistem, utilizatorul introduce programele şi datele dorite în memoria calculatorului, pentru prelucrare, şi tot cu ajutorul său rezultatele se înregistrează, sau se afişează în exterior.

Legătura între UCP şi dispozitivele de I/O (dispozitivele periferice) se face prin intermediul unor *circuite de interfaţă de I/O* care asigură, din punct de vedere hardware, schimbul corect de date. Aceste circuite de interfaţă se cuplează la magistralele calculatorului şi ele sunt adresabile la nivel de *registru-port* de intrare-ieşire. Prin *port* înţelegem aici un loc (în general un registru), cu adresă specifică, adresă care constituie o "poartă" prin care calculatorul realizează schimb de informaţie cu

exteriorul; fie culege informaţia, iar portul este port de intrare (PI), fie transmite informaţia la un port de ieşire (PO). Registrele port pot fi adresate în spaţiul de adrese al memoriei (vorbim atunci de *mapare-organizare a porturilor în spaţiul de memorie*) sau pot fi adresate în spaţiu separat de cel de memorie (*mapare în spaţiul de I/O*). Fiecare port de intrare sau ieşire are o adresă specifică.

Ca urmare adresele porturilor de intrare-ieşire pot fi tratate în două moduri:

1. *ca porturi cu adrese distincte faţă de adresele de memorie* (semnalele de control şi selecţie fiind specifice pentru memorie, respectiv pentru spaţiul de intrare-ieşire) - *mapare în spaţiu separat de I/O*;
2. *adresele porturilor sunt înglobate în spaţiul de adrese al memoriei principale*. Această organizare a adreselor (numita şi "*mapare a adreselor de I/O în spaţiul de memorie*") face ca porturile să fie selectate prin aceleaşi semnale de adresă şi control ca şi memoria. În lucrul cu porturile de I/O se vor utiliza instrucţiunile de transfer specifice memoriei.

Este important de observat că, exceptând procesorul şi memoria principală, toate circuitele conectate la magistrala de date a sistemului, deci care partajează această resursă a sistemului, sunt privite de procesorul central (UCP) ca porturi de I/O, chiar dacă funcţiile acestora nu se limitează strict la operaţii de intrare / ieşire. Este o manieră unitară de lucru a UCP cu orice circuit de interfaţă, circuit care poate include mai multe porturi de intrare / ieşire , indiferent de funcţiile specifice ale acestuia (figura 7.1). Aceasta face ca din punctul de vedere al UCP să se lucreze cu porturile adresabile într-un mod asemănător cu cel folosit la adresarea locaţiilor de memorie.

Unele circuite de interfaţă sunt incluse în circuitele specializate de control (“controllere”) ale perifericelor. De exemplu, controllerul unităţii de disc, controlează operaţiile fizice efectuate de discul magnetic. Pentru ca să se poată face cuplarea la calculator controlerul trebuie să respecte nişte specificaţii standardizate de interfaţare. Standardul IDE ("Integrated Drive Electronics") a realizat transferarea circuitelor de control către mecanismul discului hard, circuitele cuplate direct la magistralele calculatorului având o structură simplă. Pentru cazul specific al discului hard, standardul de interfaţă stabileşte nu numai modul de lucru cu partea electronică de control, ci şi modul în care se face codificarea datelor pe suportul magnetic. Funcţia principală a circuitelor interfaţă de I/O este de a rezolva diferenţele *funcţionale, electrice şi informaţionale* dintre calculator şi periferic. Circuitele controler au în plus şi sarcina specifică de control a unui anumit periferic. Principalele diferenţe între UCP şi periferie, care impun folosirea circuitelor de interfaţă, constau în următoarele:

a. perifericele sunt dispozitive a căror funcţionare se bazează pe diferite tehnologii (electromecanice, electromagnetice, electronice). De aceea trebuie să existe dispozitive de conversie a valorilor semnalului, pentru o adaptare din punct de vedere electric cu calculatorul;
b. ritmul de transfer al datelor este mult mai scăzut la periferice faţă de UCP. Pentru transferul de date între periferice şi UCP sau memorie trebuie deci să existe mecanisme de sincronizare.
c. codurile şi formatele datelor în echipamentele periferice pot fi diferite faţă de codurile şi formatele folosite în UCP şi memorie.
d. există o varietate de periferice, cu moduri de funcţionare diferite şi de aceea acestea trebuie controlate adecvat, pentru a nu perturba celelalte periferice conectate la UCP.

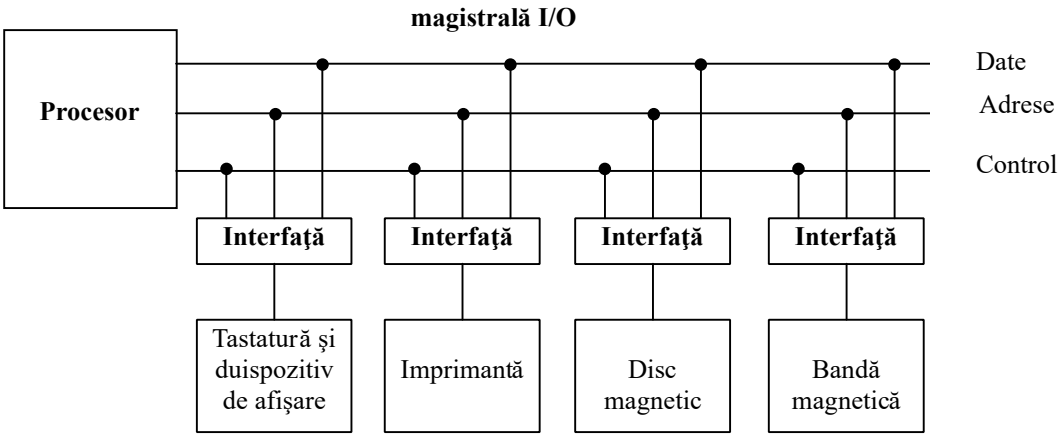


Figura 7.1. Conectarea la magistrala de I/O a dispozitivelor de intrare - ieşire.

Circuitul de interfaţă rezolvă toate aceste diferenţe, fiind inclus între UCP şi periferice, pentru a superviza şi sincroniza toate transferurile de intrare / ieşire. În plus aşa cum am amintit, fiecare periferic poate avea propriul controller care supervizeaza funcţionarea corectă, specifică a respectivului periferic. Interfaţarea a două dispozitive fizice (*hardware interfacing*) constă în a proiecta circuitele de interconectare fizică dintre aceste două dispozitive. Iată câteva exemple obisnuite de sarcini de "hardware interfacing":

- interfaţarea unei imprimante (printer) la un calculator (de exemplu, pentru PC, posibilă doar prin respectarea standardului de interfaţare USB[1] sau Centronix);
- interfaţarea unei legaturi de comunicaţie seriala la un calculator
- interfaţarea unui port de I/O digital la un calculator
- interfaţarea unor convertoare D/A sau A/D la un calculator
- interfaţarea unei unităţi de dischetă sau a unui controler de disc fix (hard disc) la un calculator.

Notiunea de interfaţare poate fi folosită însă şi dacă ne referim la programele rulate de calculator (*software interfacing*). Când cele două obiecte ce definesc interfaţa sunt programe de calculator, sarcinile de interfaţare constau în proiectarea unui alt program ce asigură comunicarea dintre primele două programe. De exemplu, un program poate fi un program de control al imprimantei (printer-driver), iar celălalt poate fi un program general de aplicaţie al utilizatorului. Programul de control al imprimantei este o procedură care tipăreşte un caracter la imprimantă ori de câte ori este apelat. Interfaţa software se referă la faptul ca programul de aplicaţie trebuie să cunoască locul în care trebuie să plaseze caracterul de tipărit, înainte de a invoca programul printer driver. După cum se vede din acest exemplu, interfaţarea software face ca parametrii să fie transmişi corect de la un program la celălalt. Protocolul de transmitere a parametrilor este stabilit doar de unul din cele doua programe ce comunică prin intermediul interfeţei. De exemplu la un PC[2], ce rulează sub sistemul de operare DOS, dintre cele doua programe ce se interfaţează, un program este, în general, un program de aplicaţie; celălalt program poate fi un program de control pentru un dispozitiv (“device driver”), de exemplu un printer-driver, sau poate realiza doar o anumită funcţie simplă (ca de exemplu alocarea memoriei). Al doilea program este cel care defineşte protocolul pentru transmiterea parametrilor. Distincţia dintre

[1] USB = Universal Serial Bus
[2] PC = Personal Computer

un hardware driver (program de control al unui dispozitiv fizic) şi o funcţie nu prezintă importanţă aici, elementul important este modul cum se transmit parametrii.

Circuitul de interfaţă fiind legat între magistrala de I/O a sistemului şi echipamentul periferic, are semnale specifice de cuplare - interfaţare cu magistrala, respectiv cu echipamentul periferic. Dacă spre periferic tipurile de semnale şi modul de lucru cu acestea depind în mare măsură de caracteristicile dispozitivului periferic, partea dinspre magistrală, face legătura cu memoria şi UCP şi cuprinde semnale pentru selecţia circuitelor de I/O, pentru controlul transferului datelor şi semnale de cerere de servicii către UCP.

Transferul datelor între un port de I/O şi UCP are loc, în principiu, asemănător cu transferul datelor între procesor şi memorie.

Există patru tipuri de semnale de comandă pe care le poate recepţiona/transmite o interfaţă:

1. semnale de control
2. semnale de stare
3. date de ieşire
4. date de intrare

Semnalele de control sunt transmise de UCP pentru a activa perifericul şi pentru a-l informa ce operaţie trebuie să efectueze. De exemplu, o unitate de bandă magnetică poate fi comandată să deplaseze banda înainte cu o înregistrare de date, să deruleze rapid banda la început, sau să pornească citirea unui bloc de înregistrări. Semnalele de control emise sunt specifice pentru fiecare tip de periferic.

Semnalele de stare sunt utilizate pentru a testa diferite condiţii ale interfeţei şi ale perifericului. De exemplu, calculatorul poate testa dacă un periferic este gata pentru un viitor transfer de date. În timpul transferului se pot produce erori care sunt detectate de interfaţă. Aceste erori sunt marcate prin setarea unor biţi dintr-un registru de stare al interfeţei, registru ce poate fi citit de procesor.

Semnalele de comandă pentru date de ieşire fac ca interfaţa să răspundă prin transferarea datelor de la magistrala de date către registrele sale interne. Consideraţi exemplul cu unitatea de bandă. Calculatorul porneşte derularea benzii prin semnale de control. Apoi procesorul monitorizează efectuarea comenzii trimise prin citirea informaţiilor de stare. Când banda a ajuns la poziţia corectă, procesorul transmite comenzile (adresă şi control) şi datele de ieşire, iar interfaţa transferă informaţia către registrele interne, iar apoi către controllerul unităţii de bandă, pentru stocare.

Comenzile pentru date de intrare sunt similare cu cele pentru ieşirea datelor, diferind doar sensul de circulaţie al informaţiei. Procesorul testează starea interfeţei pentru a verifica dacă datele cerute pot fi transferate către magistrala de date.

II. ORGANIZAREA IERARHICĂ A MAGISTRALELOR

În calculatoarele moderne transferul dintre UCP şi memorie, respectiv dispozitivele de I/O se face prin magistrale organizate ierarhic, în funcţie de viteza dispozitivelor cuplate la fiecare magistrală.

Există mai multe variante de organizare, în funcţie de tipul calculatorului (de uz general sau pentru aplicaţii specifice) şi diferenţele de viteză între dispozitive. Dacă la o magistrală se conectează mai multe dispozitive performanţa sistemului poate scădea. Cu cât sunt mai multe dispozitive conectate la magistrală cu atât creşte lungimea acesteia şi deci întârzierea la propagarea semnalelor.

În plus magistrala unică poate deveni o resursă hardware extrem de solicitată dacă este multiplexată în timp de multe dispozitive, cu viteze de funcţionare mult diferite.

De aceea marea majoritate a calculatoarelor utilizează mai multe magistrale, organizate ca o ierarhie de magistrale cu viteze de operare diferite. O structură clasică de magistrale multiple organizate pe niveluri este prezentată în figura 7.2 [Stallings00]. Există o magistrală locală care conectează procesorul cu memoria cache şi la care se pot conecta şi câteva dispozitive de I/O (locale) de mare viteză. Controllerul de memorie cache interfaţează această memorie cu procesorul şi de asemenea cu magistrala sistem la care este conectată memoria principală. Avantajul acestui mod de organizare este că transferul direct între dispozitivele de I/O şi memorie nu interferează cu activitatea procesorului care lucrează direct cu memoria cache. Dispozitivele de I/O sunt conectate la al treilea nivel ierarhic numit magistrală de extensie. Acest ultim mod de aranjare permite conectarea la magistrala de extensie a unui mare număr şi o mare diversitate de dispozitive de I/O şi în acelaşi timp izolează traficul procesor - memorie faţă de traficul de I/O.

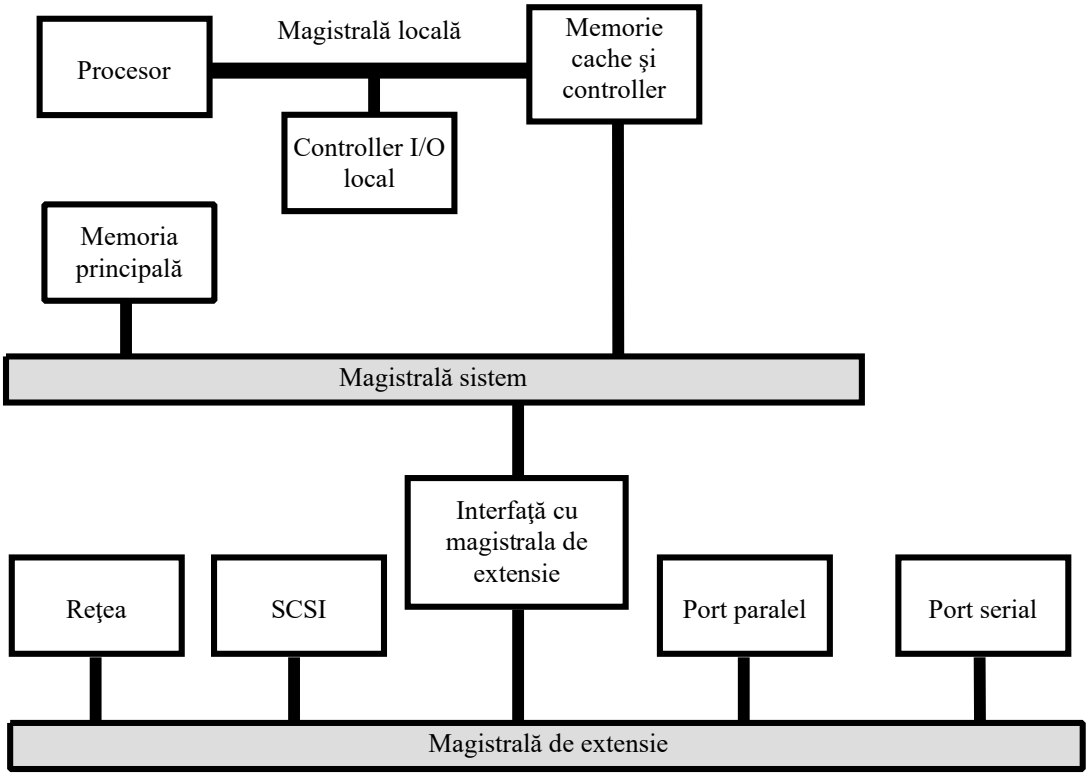


Figure 7.2. Exemplu de organizare ierarhică a magistralelor calculatorului

În figura 7.2 se exemplifică câteva dispozitive de I/O ce pot fi cuplate la magistrala de extensie. Conexiunile de tip reţea includ reţele locale (LAN - local area network), sau conexiuni la reţele pe arii extinse. Controllerul SCSI (small computer system interface) conectează la magistrala de extensie o magistrală SCSI la care se pot conecta controllere de hard disc locale şi alte periferice. Portul serial poate fi folosit pentru conectarea unei imprimante sau a unui scanner. Magistralele de extensie (numite şi magistrale de I/O) suportă o gamă largă de rate de transfer, pentru a permite conectarea unei game largi de dispozitive I/O.

Arhitectura clasică a magistralelor este eficientă, dar nu face faţă la noile dispozitive de I/O ce funcţionează la viteze din ce în ce mai mari. Ca urmare au apărut noi organizări, care introduc un nivel intermedia de magistrală de mare viteză (numit uneori magistrală la mezanin) care se interfaţează cu o punte (bridge) cu magistrala locală a procesorului. Figura 7.3 prezintă această abordare, în care controllerul de cache este integrat într-o punte, sau dispozitiv tampon, care se

conectează la magistrala de mare viteză. La această magistrală de mare viteză se pot conecta circuite controller de mare viteză pentru LAN, (cum ar fi Fast Ethernet la 100 Mbps, controller videografic). Dispozitivele de I/O cu viteză mică se cuplează în continuare la magistrala de extensie conectată printr-o interfaţa cu magistrala de mare viteză. Avantajul acestei aranjări este că dispozitivele de I/O de mare viteză sunt integrate mai aproape de procesor şi în acelaşi timp pot funcţiona în paralel cu procesorul.

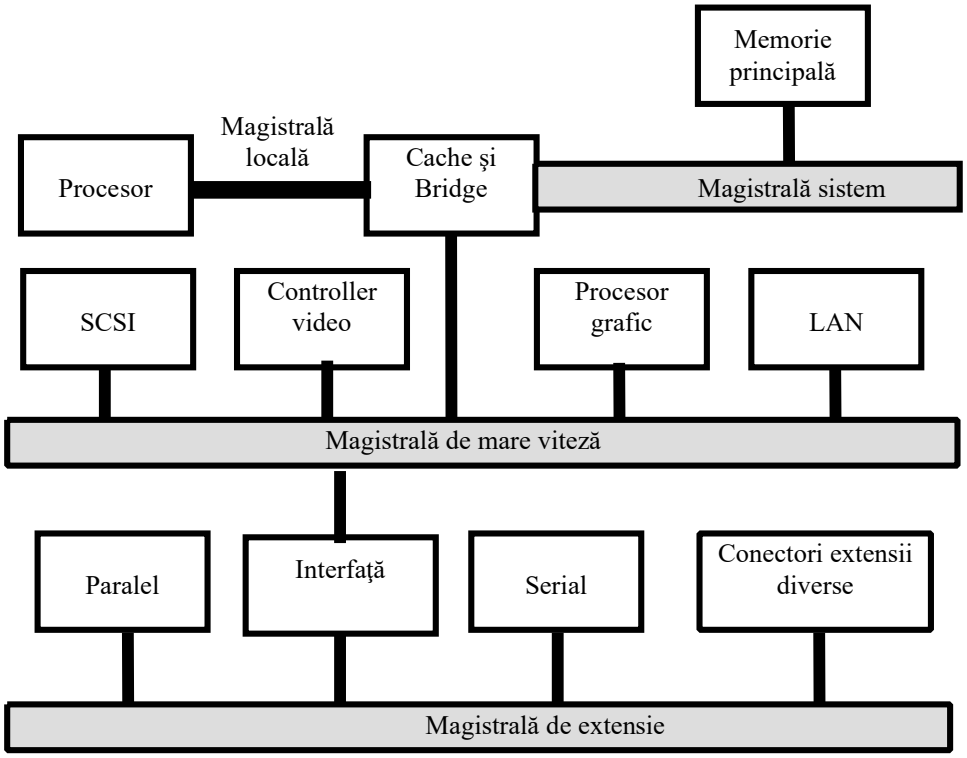


Figura 7.3. Exemplu de organizare ierarhică a magistralelor la care se introduce magistrala de mare viteză "de la mezanin"

Din punctul de vedere al modului de transfer a informaţiilor pe magistrale, acestea pot fi sincrone, sau asincrone. Magistralele sincrone sunt comandate de un semnal de ceas local. Toate transferurile pe aceste magistrale, numite cicluri de magistrală, respectă un protocol fix ce impune un anumit număr de impulsuri de ceas. Datorită sincronizării controlul transferurilor este extrem de simplu. Blocurile de interfaţă (bridge) între magistralele sincrone ce funcţionează la diferite frecvenţe de ceas trebuie să realizeze adaptarea de viteză în aşa fel încât transferurile între magistrale să se facă corect şi cât mai rapid. La magistralele asincrone transferurile nu mai trebuie să se încadreze într-un interval fix de timp, iar controlul transferurilor se face cu ajutorul unor semnale de control între cei doi corespondenţi (*handshaking)*.

Transferurile cu porturile de I/O sunt în general de tip asincron, cu posibilitatea de a introduce stări suplimentare de aşteptare ale UCP pentru a mări durata unui ciclu maşină.

Pentru majoritatea procesoarelor ce pot organiza spaţiu separat de adrese pentru porturile de I/O, în ciclurile de transfer cu porturile se introduc automat (de către UCP) stări de aşteptare (wait), care adaptează viteza UCP la viteza scăzută a dispozitivelor de I/O. Dacă porturile sunt organizate în spaţiul de memorie, acest mecanism de sincronizare, cu ajutorul stărilor de wait, trebuie construit în exteriorul UCP şi el trebuie să acţioneze ori de câte ori se face acces la o adresă ce corespunde spaţiului de I/O.

III. TRANSFERURI ASINCRONE DE DATE

Transferurile asincrone de date între două unităţi independente cer să se transmită semnale de control între unităţile ce comunică, pentru a se indica momentul la care datele sunt disponibile. Transferul asincron poate fi controlat în două moduri:

1. *cu un semnal de control*, printr-un impuls de *strobe (marcare)* furnizat de unul din corespondenţi pentru a indica celuilalt momentul la care datele vor fi transferate.
2. *cu două semnale de control,* la care datele ce trebuie transferate sunt însoţite de un semnal de control care indică prezenţa datelor pe magistrală. Unitatea ce recepţionează datele răspunde cu un alt semnal de control pentru a confirma recepţia datelor. Se stabileşte astfel o legătură de date cu confirmare ("*handshaking*").

Cele două metode de control a transferului asincron pot fi utilizate nu numai la transferurile de I/O, ele fiind folosite în unele sisteme de calcul şi la transferurile asincrone procesor-memorie, procesor - port, port - echipament periferic. În continuare vom nota cele două unităţi care comunică ca "*sursă*" şi "*destinaţie*" a datelor fără a specifica care sunt cei doi interlocutori.

III.I. Transmisie asincronă cu un singur semnal de control

Aceasta metodă de transfer asincron a datelor foloseşte o singură linie de control pentru a indica fiecare transfer. Semnalul de marcare (notat "Strobe" în figura 7.4) poate fi activat fie de unitatea sursă (figura 7.4.a) fie de cea destinaţie (figura 7.4.b). La transferul iniţiat de sursă, Strobe indică prezenţa datelor pe magistrala comună şi de asemenea poate servi ca semnal ce controlează memorarea datelor de către destinaţie. Vitezele de lucru ale sursei *S* şi destinaţiei *D* trebuie cunoscute reciproc, la proiectarea şi construcţia sistemului. În funcţie de cel mai lent dintre interlocutori, se dimensionează intervalele de timp t1, t2 şi t3, astfel încât să se efectueze un transfer corect. Dacă destinaţia este un port de ieşire, acesta trebuie sa aibă capacitatea de memorare a datelor. Pentru transferul iniţiat de sursă sursa plasează întâi datele pe magistrala comună şi după stabilizarea valorilor pentru date, sursa activează linia de strobe. Datele şi semnalul de strobe trebuie să rămână active cel puţin un timp notat t2, suficient destinaţiei pentru a citi corect datele. Adesea destinaţia foloseşte ultimul front al semnalului de strobe pentru a copia conţinutul datelor într-un registru intern. În cazul transferului iniţiat de destinaţie, dacă sursa datelor este un port de intrare, în principiu, nu este obligatorie funcţia de memorare locală. Ieşirile portului către magistrala locală de date pot fi realizate prin operatori TSL (trei stări logice), activaţi de semnalul de strobe. Pentru transfer iniţiat de destinaţie, semnalul de marcare are funcţia unei cereri de transfer de date. În multe din aceste tipuri de transferuri în calculator, impulsurile de strobe sunt controlate tot de impulsurile de ceas ale UCP (tranzitiile strobe sunt sincronizate cu tranziţii ale semnalului de ceas). Astfel că strobe ar putea fi semnalul de validare al adresei pe magistrală, sau de validare date, semnal de citire-scriere etc.

Inconvenientele acestui tip de transfer constau în lipsa totală a oricărei confirmări privind acceptarea şi / sau corectitudinea datelor recepţionate, respectiv transmise. Acest mod de transfer se poate utiliza doar între unităţile la care vitezele de lucru sunt cunoscute reciproc. Transferul de acest tip este specific pentru iniţializarea unor dispozitive programabile de I/O, la care datele transmise se înscriu în registre de control ale dispozitivelor respective.

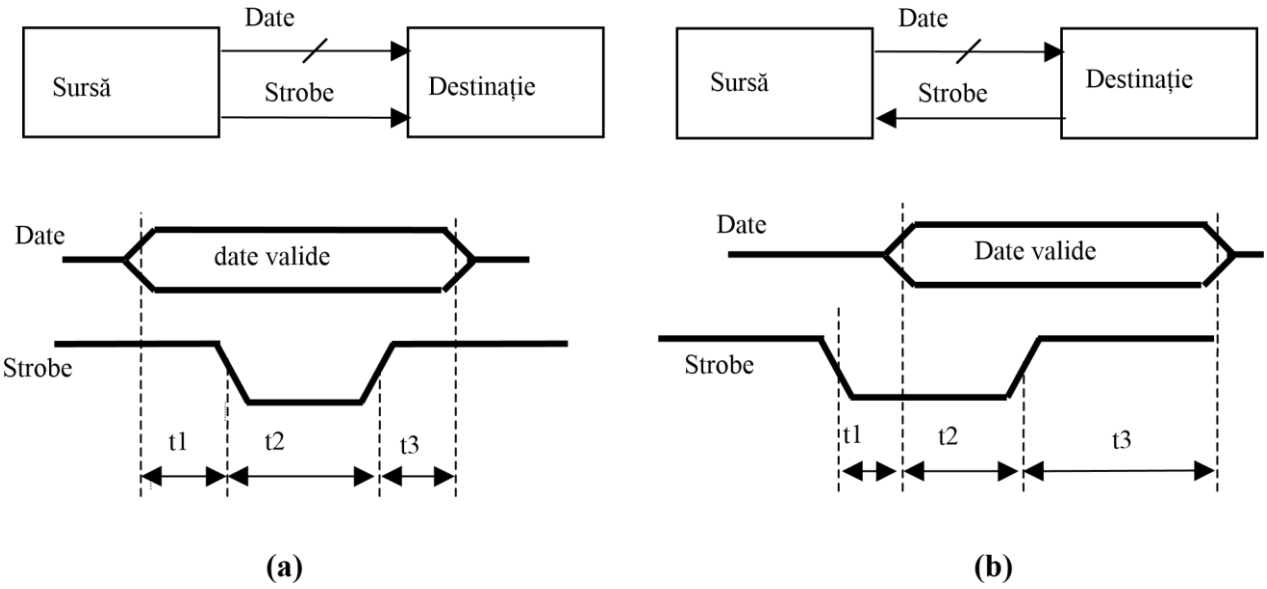


Figura 7.4. Transfer asincron de date cu un singur semnal de control. (a) diagrama bloc şi diagrama semnalelor în timp pentru transfer iniţiat de sursă. (b) diagrama bloc şi diagrama semnalelor în timp pentru transfer iniţiat de destinaţie.

III.II. Transmisie asincronă cu două semnale de control

Unul dintre semnalele de control realizează iniţierea transferului de date, iar cel de-al doilea confirmă recepţia acestora. Procedeul este numit *transfer asincron al datelor cu confirmare* ("handshaking"). Se realizează astfel un protocol de transmisie la nivel electric, între sursă şi destinaţie. Şi aici, transferul poate fi iniţiat de sursă (cu strobe - marcare date de transmis), sau de către destinaţie (prin strobe - cerere de date). În ambele cazuri interlocutorul va răspunde cu un semnal care indică că datele au fost acceptate, respectiv că datele au fost furnizate (Ready).

Secvenţa evenimentelor pentru transferul iniţiat de sursa datelor este următoarea (figura 7.5):

- sursa plasează datele pe magistrala de date comună
- sursa validează datele prin semnalul de marcare (Strobe)
- destinaţia confirmă că este gata (Ready) pentru transfer
- sursa dezactivează semnalul Strobe, care de obicei produce şi memorarea datelor la destinaţie iar după un timp dezactivează şi datele de pe magistrală
- destinaţia dezactivează semnalul Ready

Se observă că sursa menţine semnalul de strobe activ până când destinaţia este în măsură să răspundă cu semnalul de ready. Duratele semnalelor active se pot prelungi până când cel mai lent dintre corespondenţi poate efectua corect transferul de date.

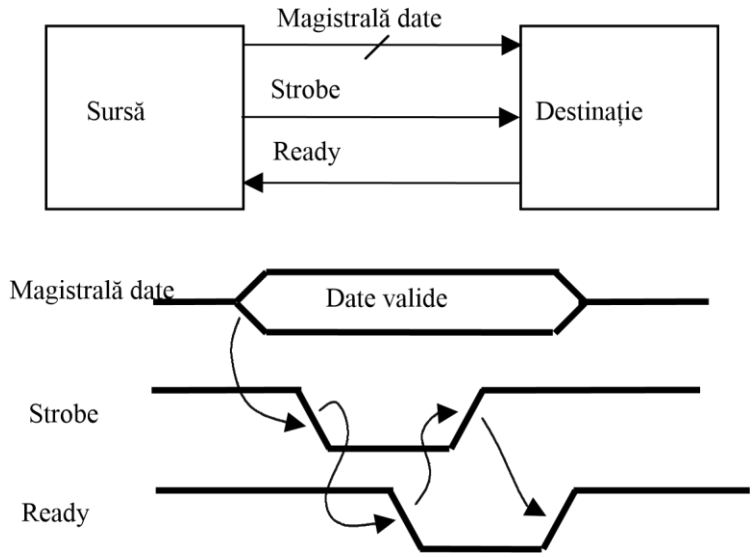


Figura 7.5. Diagrama bloc şi evoluţia semnalelor în timp pentru transfer de tip handshaking iniţiat de sursa datelor.

În cazul când iniţierea transferului se face de către destinaţie formele de undă corespunzătoare controlului transferului sunt prezentate în figura 7.6.

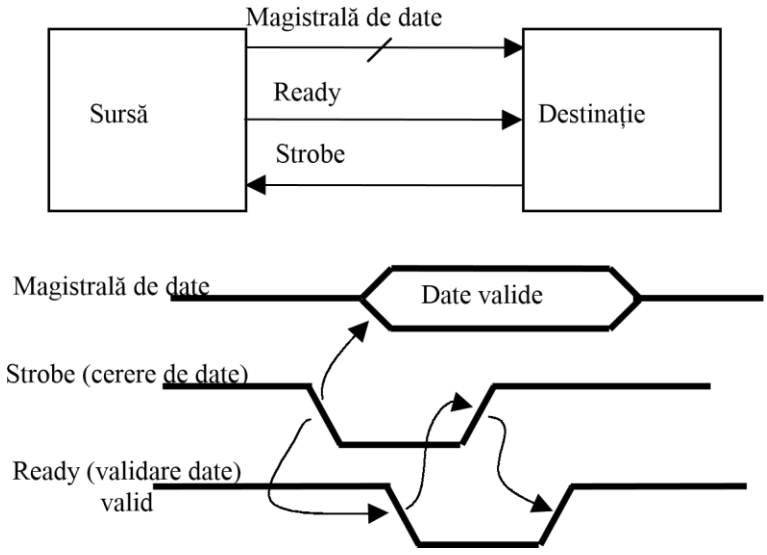


Figura 7.6. Diagrama bloc şi evoluţia semnalelor în timp pentru transfer de tip handshaking iniţiat de destinaţia datelor.

Secvenţa evenimentelor pentru transferul iniţiat de destinaţie este următoarea:

- destinaţia lansează semnalul Strobe, cu semnificaţia "cerere de date"
- sursa răspunde prin punerea datelor pe magistrala comună şi apoi prin activarea semnalului de confirmare (Ready)
- destinaţia citeşte datele şi confirmă acest lucru prin dezactivarea semnalului de strobe
- sursa consideră transferul încheiat şi dezactivează semnalul de ready.

IV. MODALITĂŢI DE TRANSFER DE INTRARE-IEŞIRE

Din punctul de vedere al circuitelor care controlează transferul şi de asemenea al dispozitivului care iniţiază un transfer de I/O, modalităţile de transfer se pot clasifica în:

1. *transfer prin program* - transfer iniţiat şi controlat în totalitate de programul rulat de UCP;
2. *transfer prin întreruperi* - transferul este controlat de UCP ca răspuns la o cerere de întrerupere externă, care iniţiază transferul;
3. *transfer prin acces direct la memorie* (DMA - "Direct Memory Acces") - transferul este controlat de un circuit controler DMA (care preia controlul magistralelor sistemului), iar iniţierea transferului este făcută fie de o cerere de transfer de la un periferic, fie la iniţiativa programului rulat de UCP.

IV.I. Transferul de I/O prin program

La acest tip de transfer, toate transferurile se iniţiază şi se controlează prin program. Transferul poate fi *direct*, caz în care procesorul citeşte un port de intrare (PI) sau scrie un port de ieşire (PO), fără nici o verificare prealabilă a stării perifericului corespunzător portului. Acest mod de transfer se foloseşte pentru sistemele simple de control numeric, dedicate unor aplicaţii fixe, în faza de proiectare şi testare a transferului (program plus hardware), sau în cazul rulării unor rutine de iniţializare a circuitelor programabile de interfaţă.

Al doilea mod de transfer prin program este transferul prin *interogare*. Intrarea în conversaţie cu un periferic se face sub controlul programului, de obicei într-o *buclă de interogare* (*polling* - scrutare a stării circuitelor periferice implicate în transfer). UCP interoghează dispozitivele periferice, după o anumită strategie stabilită prin program, dacă doresc sau nu schimb de date sau mesaje, sau dacă sunt active (gata pentru a primi informaţii). Pentru a determina dacă o operaţie de I/O este cerută, sau dacă poate avea loc, se pot folosi bistabile de condiţie (fanioane) locale, setate conform condiţiilor portului. Aceste fanioane se implementează fizic fie ca bistabile singulare, fie sunt incluse în registre de stare ale porturilor. De exemplu un program poate controla transferul de date de la un circuit de conversie analog-numerică. Procesorul declanşează conversia, iar apoi testează periodic o ieşire (un bit) de stare al convertorului. Când bitul indică terminarea conversiei, se încheie rularea buclei de aşteptare, preluându-se datele convertite.

Datorită simplităţii ei, tehnica de comunicare prin interogare programată este recomandată în sistemele în care timpul pierdut în rutina de interogare nu este critic pentru aplicaţia respectivă. Atunci când programul are de controlat mai multe periferice, pe baza unei liste a porturilor asociate perifericelor, procesorul testează pe rând starea perifericelor şi le serveşte din punctul de vedere al transferului, dacă starea citită permite acest lucru. În acest fel perifericele sunt deservite în mod secvenţial şi implicit apar întârzieri la servirea acestora, mai ales dacă numărul de periferice controlate este mare. Dacă unele dintre periferice se consideră mai importante decât altele, adresa acestora se poate introduce de mai multe ori în lista de testare din bucla de interogare. Dezavantajul principal al modului de transfer programat este constituit de timpul pierdut de UCP pentru testarea stării perifericelor, chiar dacă acestea nu cer servicii la momentele de timp respective. Acest mod de transfer prezintă însă avantaje din punctul de vedere al costurilor (minim de echipament fizic suplimentar pentru transfer). Alt avantaj este că se cunoaşte exact momentul de timp când se testează sau când se face transfer de date cu un periferic; de aceea se spune că acest tip de transfer este *sincron cu programul*.

Ca exemplu al transferului prin program vom presupune mai întâi un transfer între memorie şi un port de ieşire, indicând, mai întâi printr-o listă secvenţa de comenzi folosite:

```
                    START

                    iniţializare POINTER adresă de memorie;
                    iniţializare CONTOR al lungimii transferului;
CITIRE_MEM:         UCP citeşte date din memorie
TEST_STARE:         citire stare port
                    dacă e gata de transfer sari la TRANSF
                    dacă nu, incrementează CONTOR_RATĂRI
                    dacă CONTOR_RATĂRI = W salt la EROARE
                    altfel, salt la TEST_STARE
TRANSF:             UCP scrie data la port, şi actualizează CONTOR şi POINTER
                    dacă mai sunt date de transmis, salt la CITIRE_MEM altfel
                    salt la FINAL
EROARE:             .....

FINAL:      .....
                    END
```

Pentru I8086, dacă la adresa simbolica "**port**" se găseşte informaţia de stare (bitul 0 indicând prin 1 logic, stare gata de transfer), iar la adresa **port+1** se vor transfera datele, fragmentul de program arată astfel:

```
start:      mov dx, port
            lea  bx,  buffer
            mov cx, count
            mov si, wait
test_st:    in al,dx
            test al,1
            jnz transf
            dec si
            jnz test_st
transf:     inc dx
            mov al,[bx]
            out dx,al
            dec dx
            inc bx
            dec cx
            jnz test_st
```

Dacă sunt mai multe porturi, (portul I având adresa registrului de stare "port1", iar bitul 0 al cuvântului de stare arată dacă portul e gata (1) sau nu (0) de transfer), bucla de interogare ar putea arata astfel:

```
test1:      in al,port1
            test al,1
            jz test2
            call transf1
test2:      in al,port2
            test al,1
            jz test3
            call transf2
```

```
                :::::::
testN:          in al,portN
                test al,1
                jz test1
                call transfN
```

IV.II. Transferul I/O prin întreruperi

Aşa cum s-a arătat în capitolul 5 evenimentele externe procesorului, deci independente de programul rulat, pot produce *cereri de întrerupere.* Dacă acestea sunt acceptate, se produce *întreruperea* (suspendarea temporară a) programului rulat şi *saltul la o rutină specifică de tratare* a cererii de întrerupere. După execuţia rutinei de tratare se revine la execuţia programului întrerupt. Subsistemul de întreruperi al calculatorului nu este destinat special doar pentru operaţii de transfer de I/O, dar aici ne vom referi doar la acest aspect. Din punctul de vedere al transferurilor de I/O, avantajele sistemului de întreruperi sunt următoarele [Sztojanov87]:

- permite sincronizarea procesorului cu evenimente externe;
- eliberează procesorul de sarcina testării periodice a perifericelor, sarcină consumatoare de timp. Ca urmare transferul prin întreruperi prezintă o viteză de răspuns mai mare decât transferul prin program;
- posibilitatea de tratare ierarhizată a cererilor de întrerupere simultane sau succesive;

Cererile de întrerupere pot fi recunoscute doar la sfârşitul ciclului instrucţiune curent. În general, pentru efectuarea transferurilor de I/O se folosesc cererile de întrerupere mascabile.

Răspunsul UCP la acceptarea unei cereri de întrerupere mascabilă, consta în următoarele operaţii succesive:

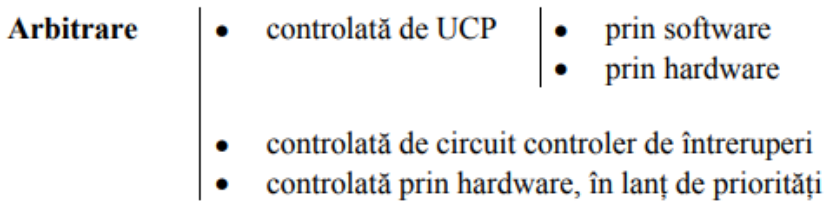
1. Salvarea stării programului întrerupt. În această fază se salvează automat în stivă cel puţin adresa de revenire la programul întrerupt. Alte informaţii privind starea programului se pot salva prin instrucţiuni introduse la începutul rutinei de tratare, iar înainte de revenirea la programul întrerupt trebuie introduse instrucţiuni pentru restaurarea acestor informaţii.
2. Confirmarea acceptării şi identificarea întreruptorului. Confirmarea acceptării întreruperii se face prin transmiterea de UCP a unui semnal de acceptare, în urma căruia perifericul trimite, de obicei, un vector de întrerupere pentru identificare. Există şi sisteme de identificare care nu folosesc vector de întrerupere.
3. Calcularea adresei de început a rutinei de tratare a întreruperii. Pe baza informaţiei de identificare se calculează unde se va face saltul pentru ca cererea de întrerupere, pentru transferul de date, să fie servită.
4. La terminarea rutinei de tratare, se execută o instrucţiune specială, care comanda refacerea conţinutului contorului de program şi deci revenirea la programul întrerupt.

Generarea vectorului de întrerupere se face fie direct de către dispozitivul întreruptor (de fapt de către circuitul de interfaţă cu un anumit periferic), fie centralizat, pentru toate întreruperile mascabile, de către un dispozitiv special numit controller de întreruperi.

Arbitrarea întreruperilor multiple.

În cazul întreruperilor multiple, simultane sau succesive, subsistemul de întreruperi trebuie să realizeze un arbitraj al acestora pentru a stabili care dintre cererile de întrerupere multiple va fi servită la un moment dat. Sistemul de arbitrare alocă priorităţi cererilor de întrerupere, servind întotdeauna cererile cu prioritatea cea mai mare. Se permite de asemenea ca o rutină de tratare să poată fi la rândul

ei întreruptă de o cerere cu prioritate mai mare. De obicei, pentru un procesor de uz general, priorităţile intrinseci sunt atribuite în următoarea ordine: întreruperile interne (excepţiile) cu prioritatea cea mai mare, apoi urmează întreruperile externe (hardware) nemascabile şi în final cele mascabile. Arbitrarea întreruperilor multiple se poate realiza prin mai multe metode:



În cazul arbitrării *controlate de UCP prin hardware*, UCP conţine intern circuitele necesare realizării arbitrării. Circuitul de arbitrare primeşte mai multe intrări de cereri de întrerupere, acestea, sau combinaţii ale acestora, corespunzând la un anumit nivel prefixat de prioritate. Adresele rutinelor de tratare pot fi fixe (cum este de exemplu cazul microprocesorului I8085) sau variabile (de exemplu la microprocesorul MC68000).

De exemplu, la microprocesorul Intel 8085 există 5 intrări de cereri de întrerupere. Una dintre intrări este numită INT şi ea funcţionează pentru întreruperi vectorizate, deci pe bază de vector de întrerupere. Celelalte intrări nu au nevoie de vector de întrerupere, pentru că intern ele au alocată câte o prioritate fixă, iar saltul către rutina de tratare se face tot la o adresă fixă. În tabelul 7.1. se indică intrările de întrerupere la microprocesorul I8085, împreună cu ordinea de prioritate alocată (prioritatea 1 fiind maximă) şi adresele de salt.

Tabel 7.1. Întreruperi la I8085.

Prioritate fixă	**Nume intrare întrerupere**	**Adresă de salt**
1	TRAP	24h
2	RST 5.5	2Ch
3	RST 6.5	34h
4	RST 7.5	3Ch
5	RST 7.5	prin Vector

Arbitrarea *controlată de UCP prin software*, constă într-un suport hardware minimal, extern UCP, care să sprijine operaţiile de recepţie a cererilor de întrerupere şi respectiv de identificare şi validare sau nu a întreruptorului, în funcţie de prioritatea alocată. Circuitele au ca element principal un registru-port de intrare, în care se alocă câte un bit pentru fiecare dispozitiv întreruptor. Dacă acest bit este setat, dispozitivul a cerut cerere de întrerupere. Simultan cu setarea bitului din registru, semnalul de cerere de întrerupere se transmite şi către UCP. Prin citirea registrului port de intrare şi compararea cu situaţia anterioară (memorată într-un registru intern), UCP detectează dacă a apărut o cerere de întrerupere cu prioritate mai mare decât cea a programului curent. În cazul unei întreruperi cu prioritate mai mică se continuă programul curent (după eventuala înregistrare a cererilor în vederea servirii ulterioare). În cazul priorităţii mai mari, UCP începe execuţia subrutinei de servire asociată noii cereri, întrerupând astfel programul curent şi actualizând registrul de serviciu. După terminarea execuţiei subrutinei de servire se reia ultimul program întrerupt. De asemenea, după comutarea contextului, UCP trebuie să revalideze întreruperile pentru a permite unor eventuale cereri de întrerupere prioritare să fie luate în considerare. Cererile de întrerupere ale dispozitivelor individuale de I/O, pot fi activate sau dezactivate prin program, prin intermediul unor bistabili de mască, asamblaţi

de obicei într-un registru de mascare. Astfel că la momente diferite de timp, prin aplicarea unor măşti diferite, programul poate modifica ordinea de (prioritate de) servire a cererilor de întrerupere.

Arbitrarea controlată de circuit controler de întreruperi, este cea mai des întâlnită la microcalculatoarele moderne. Controlerul de întreruperi programabil (PIC[3]) este un circuit specializat care preia, în principiu, toate sarcinile pe care le avea UCP la arbitrarea prin software: acceptă cereri de întrerupere de la mai multe surse, determină dacă există cel puţin o cerere de întrerupere cu prioritate superioară celei a programului curent, iar în caz afirmativ, generează către UCP o cerere de întrerupere. La primirea confirmării acceptării cererii de întrerupere, circuitul PIC generează pe magistrala de date vectorul de întrerupere asociat celei mai prioritare cereri existente. UCP întrerupe programul în curs şi serveşte cererea prin mecanismul descris anterior. Circuitul controler, este numit şi "programabil", pentru că poate funcţiona în mai multe moduri de lucru programabile prin înscrierea de către UCP a codurilor corespunzătoare modurilor de lucru în registrele de control ale PIC. De obicei programarea modului de lucru se face printr-o secvenţă de iniţializare, executată de UCP, la pornirea calculatorului. Modul de lucru poate însă să fie modificat ulterior, tot prin program. Circuitul PIC estre văzut de UCP ca un circuit de interfaţă ce cuprinde mai multe porturi de I/O. Modul de alocare a priorităţilor poate fi:

1. fix
2. rotitor. Acest mod implementează "politeţea" în servirea cererilor de întrerupere. Iniţial priorităţile sunt ordonate în ordine descrescătoare, dar după servire, fiecare nivel de prioritate devine automat cel mai puţin prioritar.
3. cu mascare. În acest caz priorităţile pot fi modificate dinamic, masca invalidând temporar anumite niveluri de prioritate.

Arbitrarea realizată prin PIC asigură un timp mic de răspuns pentru întreruperile hardware şi o flexibilitate mare în alocarea sau modificarea priorităţilor. În plus, multe dintre circuitele de tip PIC pot fi cascadate, ceea ce permite extinderea uşoară a sistemului într-un calculator.

Metoda de *arbitrare descentralizată prin lanţ de priorităţi* a întreruperilor presupune existenţa în fiecare dispozitiv întreruptor a unei logici capabile să asigure protocolul electric de tratare a întreruperilor. Dispozitivele întreruptoare se conectează într-un "lanţ de priorităţi" (daisy chain), ca în figura 7.7.

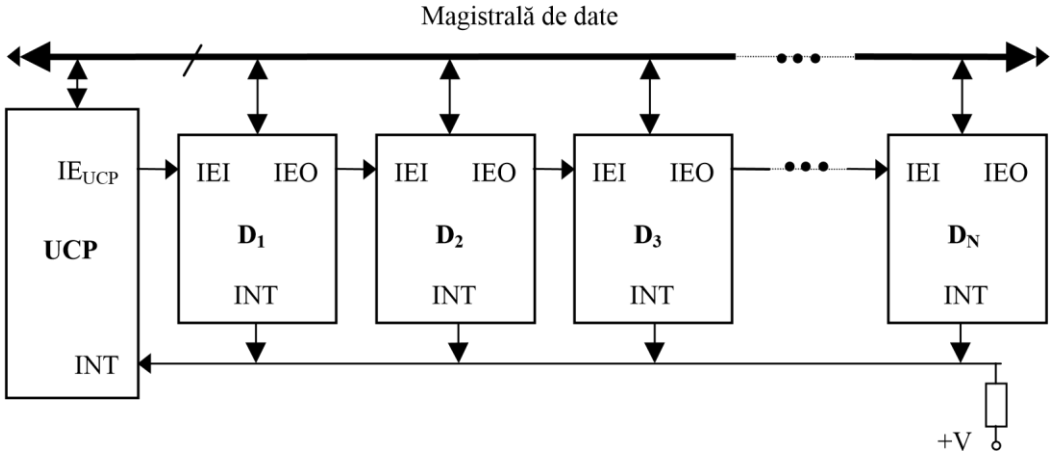


Figura 7.7. Exemplu de arbitrare a cererilor multiple de întrerupere prin lanţ de priorităţi

[3] Programmable Interrupt Controller

În figură fiecare dispozitiv D_i , i = 1,N dispune de câte o ieşire $\overline{INT}$(cu colector / drenă în gol), ceea ce permite legarea împreună a acestor ieşiri la linia de intrare $\overline{INT}$ a UCP. Se realizează astfel un SAU cablat pe această intrare a UCP. (de fapt legând ieşirile împreună avem un SI cablat între ieşiri active în stare JOS, adică:

$$\overline{A}\cdot\overline{B}\cdot\overline{C}\cdots = \overline{A+B+C+\cdots} \qquad (7.1)$$

Lanţul de dispozitive conduce la priorităţi fixe, ce ţin de poziţia dispozitivului în lanţ. Prioritatea cea mai mare este alocată celui mai apropiat dispozitiv faţă de UCP (D_1), iar prioritate minimă o are ultimul dispozitiv din lanţ.

UCP generează semnalul de validare a întreruperilor externe (IE_{UCP}) care se propagă prin lanţ, trecând prin celulele succesive ale lanţului. Pentru fiecare dispozitiv, starea intrării de validare IEI (Interrupt Enable Input) este copiată la ieşirea IEO (Interrupt Enable Output) a fiecărui dispozitiv, dacă dispozitivul respectiv nu cere întrerupere către UCP. Dacă un dispozitiv a cerut întrerupere, care a fost recunoscută şi se face tratarea acesteia, pe tot timpul servirii între intrarea IEI şi ieşirea IEO ale dispozitivului servit nu mai există egalitate. Pe toată această perioadă dispozitivul aduce IEO la zero pentru a nu permite dispozitivelor cu prioritate mai mică să-i întrerupă rutina de tratare. În momentul în care un dispozitiv D_i cere întrerupere, pot exista două situaţii:

⇒ dacă IEI_i=0, atunci un dispozitiv mai prioritar decât D_i a cerut întrerupere şi servirea sa de către rutina corespunzătoare nu s-a încheiat. Ca urmare dispozitivul D_i va aştepta până când IEI_i trece în stare activă "1".

⇒ dacă IEI_i=1, atunci D_i poate genera cererea de întrerupere către INT (aducând această linie la zero logic); aceasta pentru că D_i este dispozitivul cel mai prioritar care cere întrerupere la momentul respectiv de timp. Ca urmare, după lansarea cererii de întrerupere, IEO_i devine zero, invalidând cererile de întrerupere de la dispozitivele cu prioritate mai mică din lanţ.

Din cele de mai sus rezultă că dacă a apărut o cerere de întrerupere către UCP, în lanţ, un singur dispozitiv are IEI = 1 şi IEO = 0, şi anume cea servită. Pentru toate celelalte dispozitive ieşirea IEO are valoare logică identică cu intrarea IEI. Dacă UCP acceptă cererea de întrerupere, dispozitivul întreruptor depune pe magistrala de date a sistemului vectorul de întrerupere, pentru a fi identificat şi servit corespunzător.

IV.III. Transferul prin acces direct la memorie

Transferul, prin intermediul UCP, a blocurilor mari de date între memoria principală şi periferice, se face relativ lent, pentru că de fiecare dată informaţia trece şi prin registre ale UCP. În plus pentru o cantitate mare de date transferate între memorie şi periferic UCP va consuma foarte mult timp.

Transferul prin acces *direct la memorie* (*DMA* - Direct Memory Acces) este executat direct între memoria principală şi dispozitivul periferic, fără ca datele să mai treacă prin UCP, sub comanda unui circuit controler DMA, care controlează temporar magistralele sistemului. În figura 7.8. se presupune că iniţierea transferului se face de către periferic (prin intermediul interfeţei) care efectuează o cerere de transfer prin acces DMA spre circuitul controler de DMA (DMAC). Acesta solicită de la UCP controlul magistralelor prin semnalul de *cerere de control a magistralelor*, *BR*, (Bus Request). Cu o mică întârziere (la sfârşitul ciclului maşină curent) UCP cedează controlul magistralelor, îşi trece ieşirile către acestea în HiZ (stare de înaltă impedanţă) şi informează despre aceasta prin semnalul *acordare a controlului magistralelor*, *BG*, (Bus Grant). Circuitul controler

DMAC furnizează adresele pe magistrala de adrese, preia controlul semnalelor de scriere (WR) şi citire (RD) şi trimite către periferic semnalul de acceptare a transferului prin DMA. Când dispozitivul I/O primeşte acest semnal de acceptare, el pune un cuvânt pe magistrala de date (pentru scriere în memoria principală) sau citeşte un cuvânt de pe magistrala de date (pentru citire din memoria principală).

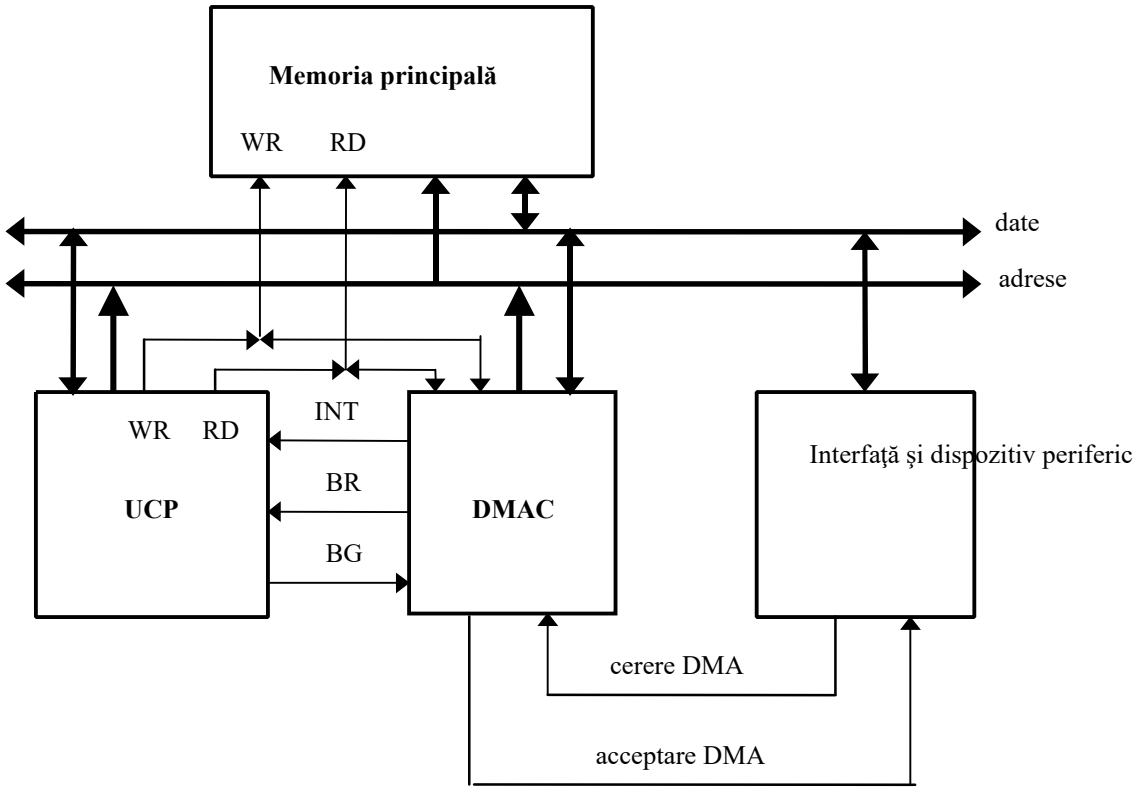


Figura 7.8. Exemplu de transfer DMA (DMAC = Controller DMA)

Transferul de date se va face *direct* între periferic şi memoria principală, fără ca UCP să mai fie intermediar al acestui transfer (aşa cum se întâmpla la transferul prin program şi prin întreruperi). Un *ciclu DMA* reprezintă transferul unui cuvânt din sau în memorie. După terminarea unui ciclu, controllerul DMA poate continua cu alte cicluri DMA (dacă a fost programat să facă acest lucru şi dacă cererea DMA de la periferic se păstrează activă), sau poate returna controlul magistralelor către UCP prin inactivarea semnalului BR. Pentru transfer *pe cuvânt*, secvenţa de semnale de cerere DMA (cerere → BR → BG → acceptare) conduce la desfăşurarea unui singur ciclu DMA. Pentru un alt ciclu DMA, secvenţa de semnale trebuie repetată şi abia apoi se transferă un nou cuvânt. Cealaltă variantă de transfer, în *mod rafală-bloc*, presupune transferul mai multor cuvinte pentru fiecare cerere de transfer DMA. Este un mod de mare viteză, care însă dacă controllerul şi UCP sunt legate la aceleaşi magistrale poate menţine procesorul în inactivitate pentru o perioadă de timp. În cazul transferului pe cuvânt, întreruperea poate fi insesizabilă, ca timp, pentru procesor, şi de aceea uneori acest mod de transfer este numit "*cu furt de ciclu*".

De obicei circuitele controler DMA pot gestiona transferuri cu mai multe periferice simultan, pentru fiecare periferic existând (intern controllerului) un *canal DMA*.

Conform funcţiilor pomenite pe scurt mai sus, structura controllerului DMA trebuie să cuprindă:

- Logică de *comandă şi sincronizare*. Modul de lucru al controllerului DMA este programabil. De aceea el este iniţializat printr-o rutină de iniţializare rulată de UCP, fixându-se astfel: modul de transfer (cuvânt sau bloc-rafală), sensul transferului (la memoria principală sau de la memoria principală), nivelul activ al semnalelor de interfaţă cu echipamentul periferic, priorităţile acordate canalelor, modul de tratare al sfârşitului de transfer etc. Circuitul controller DMA are de asemenea o interfaţă cu UCP, care permite iniţializarea registrelor controllerului, comanda ulterioară de către UCP a controllerului, precum şi realizarea protocolului BR/BG;
- *Circuite tampon* pentru conectarea la magistralele calculatorului;
- *Logică de arbitrare a priorităţilor canalelor*;
- *Logică specifică fiecărui canal DMA*, care cuprinde cel puţin următoarele componente:
  - ⇒ un registru *numărător de adrese* care generează adresa curentă (din memoria principală) de transfer;
  - ⇒ un *numărător de cuvinte de transferat*, care se decrementează la fiecare ciclu DMA efectuat (valoarea iniţială a acestui contor este egală cu numărul de cuvinte ce se doreşte a fi transferat. Atunci când conţinutul a ajuns la zero, ciclul de transfer s-a sfârşit);
  - ⇒ un *registru de stare a canalului*, care poate fi citit de UCP şi care indică: sensul transferului prin canal, canal activat / dezactivat, prioritatea alocată etc.

Transferul prin DMA prezintă avantaje din punctul de vedere al vitezelor mari de transfer, pentru blocuri mari de date. Este un transfer folosit în aplicaţii de genul: transfer cu discurile magnetice, transfer cu plăci periferice ce conţin convertoare AD sau DA rapide etc.

BIBLIOGRAFIE

1. [Borcoci95] Borcoci E., Zoican S., Popovici E., Arhitectura microprocesoarelor, partea I, Ed. Media Publishing, Bucureşti, 1995.
2. [Burileanu94] Burileanu,C., Arhitectura microprocesoarelor, Editura DENIX, Bucureşti, 1994;
3. [Crutu87] Crutu, Gh., Romanca, M., Fratu, A., Calculatoare, microsisteme de calcul, Universitatea din Brasov, 1987;
4. [Dodescu80] Dodescu, Gh., Ionescu, D., Misdolea, T., Nisipeanu, L., Pilat, F., Sisteme electronice de calcul şi teleprelucrare, Ed.Didactică şi Pedagogică, Bucuresti, 1980;
5. [Furht87] Furht, B., Milutinovic, V., A survey of microprocessor architecture for memory management, IEEE Computer, March 1987, vol.20, no3, pp. 48-66.
6. [Hayes88] Hayes, J., Computer Architecture and Organisation, McGrow Hill Comp., 1988.
7. [Lupu86] Lupu, C., s.a., Microprocesoare, Ed. Militară, Bucureşti 1986;
8. [Mano93] Mano, M., Computer System Architecture, Prentice-Hall Inc. 1993.
9. [MDE72] Mic dicţionar enciclopedic, Editura enciclopedică română, Bucureşti, 1972
10. [Nicula97] Nicula, D., Arhitecturi de microprocesoare de performanţe ridicate, Teză de doctorat, Universitatea TRANSILVANIA Braşov, 1997
11. [Patterson90] Patterson, D., Hennessy, J., Computer Architecture A Quantitative Approach, Morgan Kaufmann Publishers, Inc. 1990;
12. [Patterson96] Patterson, D., Hennessy, J., Computer Architecture - A Quantitative Approach, second edition, Morgan Kaufmann Publishers, Inc. 1996;
13. [Patterson94] Patterson, D., Hennessy, J., Computer Organization & Design, the Hardware Software Interface, Morgan Kaufmann Publishers, Inc. 1994;
14. [Pfaffenberger96] Pfaffenberger, B., Dicţionar explicativ de calculatoare, Ed. Teora, Bucureşti, 1996
15. [Pop2000] http://www.ida.liu.se/~paupo, Slides for Advanced Computer Architecture, Paul Pop, Institutionen för Datavetenskap, Linköpings Universitet
16. [SPEC] The Standard Performance Evaluation Corporation, http://www.spec.org
17. [Stallings00] Stallings, W., Computer Organization and Architecture, 5th edition, Prentice Hall International, Inc., 2000.
18. [Stefan91] Ştefan, Gh., Funcţie şi structură în sistemele digitale Ed. Academiei Române, 1991;
19. [Stefan83] Ştefan, Gh., Drăghici, I., Mureşan, T., Barbu, E., Circuite integrate digitale, Ed.Didactică şi Pedagogică, Bucureşti 1983;
20. [Strugaru92] Strugaru, C., Popa, M., Microprocesoare pe 16 biţi, Editura TM, Timişoara 1992;
21. [Sztojanov87] Sztojanov, I., Borcoci, E., Tomescu, N., Bulik, D., Petrec, M., Petrec, C., De la poarta TTL la microprocesor, Ed.Tehnică, Bucureşti, 1987;
22. [Tanenbaum99] Tanenbaum, A., Organizarea structurată a calculatoarelor, ediţia a IV-a, Computer Press AGORA, Tg. Mureş, 1999.
23. [Toacse85] Toacse, Gh., Introducere în microprocesoare, Ed. Ştiinţifică şi Enciclopedică, 1985;
24. [Toacse96] Toacşe, Gh. Nicula, D. Electronică digitală, Ed. Teora, 1996
25. [Weems96] Weems, Ch. Jr., Computer Science Course 635, Notes from Lecture 3, at www.cs.umass.edu/~weems/index.hrml