CONCEPTUL DE MAGISTRALA

În funcționarea unui sistem microprocesor sunt necesare legături multiple și complexe între diferitele părți funcționale ale acestuia. ***Grupurile de interconexiuni, cu funcții similare, care fac legătura între diversele secțiuni ale unui sistem microprocesor se numesc magistrale.*** Ele conectează *semnalele de date, adrese, control*, ale unei secțiuni, cu semnalele omologe ale celorlalte secțiuni, constituind calea de legătura dintre ele.

Există două seturi de magistrale în orice sistem microprocesor:

- ***magistrale interne*** (canalele de legătura între diversele unități funcționale din unitatea centrală - *CPU* a sistemului)
- ***magistrale externe*** (căile de comunicație între unitatea centrală - *CPU* și componentele externe acestuia).

Fiecare grup de magistrale poate fi subdivizat în trei categorii, după tipul informației transferate:

- ***magistrale de adrese***
- ***magistrale de date***
- ***magistrale de control.***

O magistrală poate avea mai multe linii, permițând transferul simultan al unui cuvânt de informație (adresă sau dată), caz în care magistrala se numește *magistrală paralelă.* Există de asemenea, *magistrale seriale*, pentru care datele sunt transmise multiplexat în timp (un cuvânt este transmis bit cu bit, pe aceeași linie), care sunt, evident, mult mai lente decât magistralele paralele. Practic, toate magistralele interne ale unității centrale *(CPU)*, cât și cele externe care conectează unitatea centrală *(CPU)* cu memoria și cu majoritatea interfețelor de intrare / ieșire ale sistemului, sunt organizate ca magistrale paralele. Legături seriale se utilizează mai ales pentru transmiterea datelor la / de la echipamente aflate la distanță de unitatea centrală (terminale, linii telefonice, etc.).

În figura 1 se reprezintă o structură de principiu a modului de interconectare între diferitele elemente componente ale sistemului microprocesor, prin intermediul diferitelor magistrale ale acestuia. Deoarece unitatea centrală nu poate alimenta simultan un număr prea mare de circuite exterioare (există valori limitate ale curenților pe care aceasta îi suportă sau generează), este necesară amplificarea semnalelor de pe magistralele sistemului. Această adaptare a semnalelor este efectuată cu diferite tipuri de circuite, dintre care pot fi enumerate următoarele tipuri:

- ***circuite tampon amplificatoare de magistrală (drivere);***
- ***circuite tampon bidirecționale (buffere bidirecționale);***
- ***circuite pentru reținerea datelor (latch).***

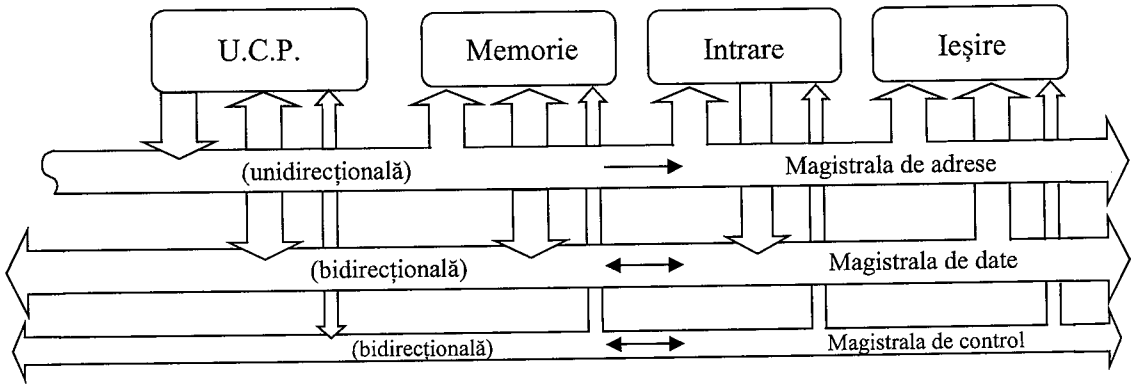


Fig. 1. Structura unui sistem microprocesor. Magistralele sistemului

Figura 2 prezintă utilizarea *circuitelor tampon (uni și bidirecționale)*, pe magistralele sistemului. Semnalele obținute la ieșirea acestor circuite formează magistrale similare celor din care provin. Uzual, se utilizează circuite tampon unidirecționale pentru semnalele de pe magistralele de adrese și de control, și circuite tampon bidirecționale, pentru magistralele de date. În cazurile în care microprocesorul se interfațează cu periferice mult mai lente decât unitatea centrală, este necesar ca informația să fie prezentă la intrarea acestor circuite, până la preluarea ei de către respectivul circuit. În aceste cazuri, se pot utiliza circuite pentru reținerea datelor (latch - reprezentând în esență, registre paralele), în care informația prezentă pe pinii de intrare, de pe o magistrala a sistemului, este înscrisă la primirea unui semnal de control (de scriere), generat direct de către microprocesor sau de către un circuit decodificator.

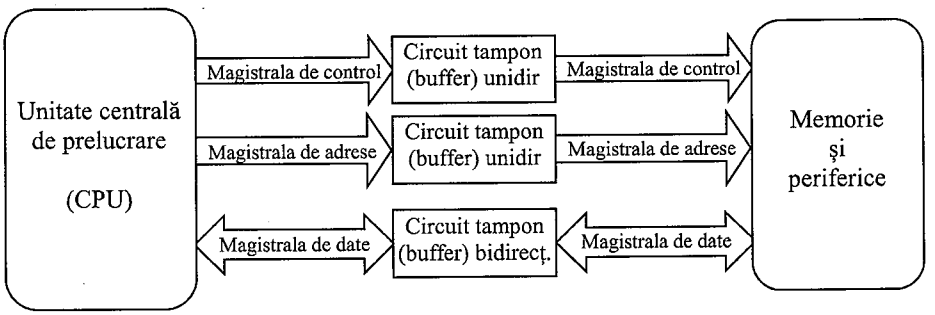


Fig. 2. Utilizarea circuitelor tampon (buffere)

Figura 3 prezintă structura unui asemenea montaj, care menține la ieșirea către un afișaj cu segmente informația dorită, până la înscrierea altor date, la un acces ulterior al circuitului latch din schemă.

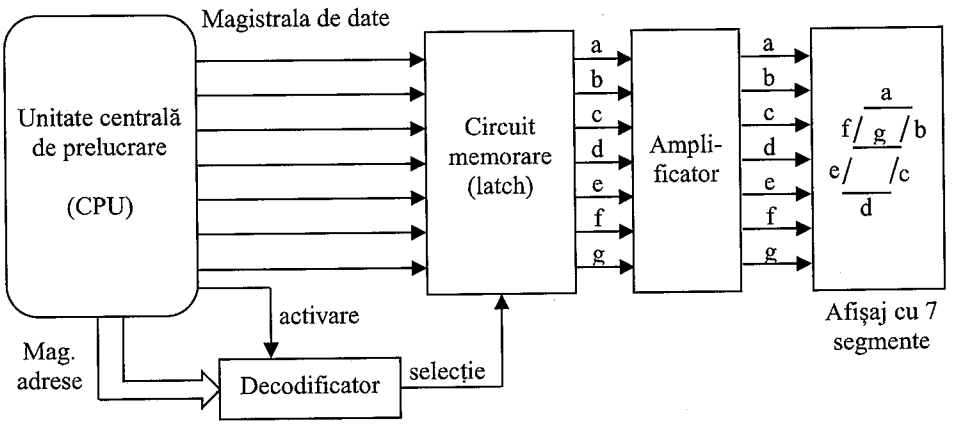


Fig. 3. Utilizarea circuitelor de tip latch

Magistrala de date

Magistrala de date este destinată atât transferului *unidirecțional* de instrucțiuni de la memorie (citire a programului), cât și celui *bidirecțional*, de date între memorie, unitatea centrală și/sau interfețele de intrări/ieșiri (informația parcurge magistrala în ambele sensuri, sau de la procesor la una dintre unitățile externe acestei unități, sau de la una dintre aceste unități spre procesor). Direcția transferului de informație este supervizată de către secțiunea de *CONTROL* a microprocesorului, prin generarea de semnale specifice (citire sau scriere). Lungimea cuvântului microprocesorului determină numărul de linii de conexiune din magistrala de date (8, 16, 32, etc.).

Dacă în structurile de tip *von Neuman*, există o singură magistrală de date, pe care se vehiculează, la momente de timp diferite, atât cuvintele de program (coduri de instrucțiune), citite din memoria program, către microprocesor, cât și cuvintele de date, între memorie sau periferice și microprocesor, la execuția instrucțiunilor, în structurile de tip *Harvard* (specifice procesoarelor *DSP*), există magistrale independente pentru transferul cuvintelor de instrucțiune, respectiv a datelor propriu-zise ale programului. Astfel, este mărită eficiența globală a sistemului.

Magistrala de adrese

Magistrala de adrese este o magistrala unidirecțională. Ea vehiculează codul binar reprezentând locația (adresa) datei ce se va utiliza în cadrul operației ce se execută. Astfel, ea poate selecta adresa celulei de memorie de la care se citește o nouă instrucțiune, o dată, sau la care se memorează un rezultat. Multe microprocesoare utilizează magistrala de adrese și pentru configurarea adresei echipamentului de intrare-ieșire selectat la o operație de *INPUT* sau *OUTPUT*. Este sarcina secțiunii de *CONTROL* de a delimita și indica, prin generarea de semnale de control specifice, dacă este vorba de o operație de citire sau scriere din memorie sau intrare-ieșire.

Adresabilitatea unui microprocesor este dată de numărul de biți al magistralei de adrese. Un număr uzual de 65536 (64 ko) celule de memorie vor necesita 16 linii de adresă pe această magistrală (2^{16}=65536). În general, se vor putea adresa 2^N celule de memorie prin intermediul a N linii de adresă.

Menționăm și aici că structurile *von Neuman* au o singură magistrală de adrese, vehiculând la momente de timp diferite, adresa de unde se citește cuvântul de instrucțiune, respectiv adresa la/de la care se efectuează transferul datelor pentru execuția propriu-zisă a instrucțiunii. Structurile cu arhitectura *Harvard* au două magistrale separate de adrese, pentru adresarea memoriei de program, respectiv a celei de date (ceea ce permite ca în paralel cu execuția unei instrucțiuni să se poate adresa și citi cuvântul de instrucțiune următor).

Magistrala de control

Această magistrală furnizează informații suplimentare necesare pentru indicarea operației ce se efectuează. Numărul de semnale de pe această magistrală depinde de numărul de semnale de control necesare pentru microprocesorul utilizat. Astfel de semnale de control pot fi :

- *semnalul de ceas* al sistemului (care asigură funcționarea secvențială cu o periodicitate fixă a întregului sistem microprocesor)
- *semnalele de citire/scriere în memorie*
- *semnalele de citire/scriere pentru intrare/ieșire din sistem* etc.

Corelate cu magistrala de adrese, semnalele de pe magistrala de control permit selecția unică a echipamentului sau a celulei de memorie căreia/de la care, prin intermediul magistralei de date, se transmite/preia informația, conform operației ce se execută.

Un aspect important este cel ridicat de problema desfășurării operațiilor în sistem, pe aceste magistrale, în mod corect, fără probleme de interferență a informațiilor deoarece, pe aceleași căi, sunt transferate în sensuri și spre/de la echipamente diferite.

Răspunsul este dat de însăși funcționarea secvențială, sub supravegherea funcției de *CONTROL*, a întregului sistem. Este asigurată activarea unică, după cum s-a menționat, a celulelor de memorie sau a elementelor de intrare/ieșire cu care se operează la un moment dat. Acestea sunt dotate cu *circuite tampon (buffere)* proprii. Conectarea pe liniile magistralelor, în sensul transferului de informație pe magistrală, este efectuată prin circuitele tampon respective, care sunt selectate doar în cazul adresării circuitului respectiv. La fel, înscrierea informației într-o anumită celulă de memorie sau element de ieșire se face prin activarea funcției de memorare doar pentru elementul respectiv. Astfel, la efectuarea unei operații de transfer de informații în sistem, va exista o singură sursă a semnalelor de pe fiecare magistrală, și va fi selectat un singur element în care datele se vor înscrie la operația respectivă. Aceste selecții unice vor trebui asigurate prin proiectarea corespunzătoare a circuitelor de adresare și decodificare.

Ceea ce poate fi evidențiat acum este faptul că microprocesorul nu trebuie să fie privit ca o colecție de circuite digitale, conectate într-o manieră convențională. Această conectare între circuite este în primul rând o funcție a programului (*software*-ului) microprocesorului.

Selecții, activări și operații specifice ale elementelor sistemului microprocesor apar ca urmare a execuției instrucțiunilor programului. Se evidențiază deci necesitatea stăpânirii unei noi tehnici de "realizare" a acestor conexiuni logice funcționale, aceea a *programării* microprocesorului. Prin program se implementează astfel echivalentul unor structuri logice complexe, variabile în timp.

ELEMENTE DE INTRARE/IEŞIRE ALE SISTEMELOR CU MICROPROCESOR

Pentru a-şi realiza funcţiile la bordul aeronavei, un calculator trebuie să preia date din mediul exterior şi să trimită rezultatele prelucrărilor la elementele de execuţie sau afişare corespunzătoare. Acest schimb de date cu exteriorul poartă numele de proces de intrare-ieşire.

Pentru ca datele să poată fi preluate de calculator trebuie ca ele să fie prezentate deja în format numeric binar. La ieşire, calculatorul oferă rezultatele tot în format numeric binar. Conversia mărimilor de intrare în format numeric şi a rezultatelor în mărimi analogice se realizează cu ajutorul interfeţelor de intrare-ieşire.

Unitatea centrală (CPU) a unui calculator poate efectua multiple operaţii interne dar trebuie să asigure şi comunicaţia acesteia cu exteriorul, fără de care funcţionalitatea sistemului ar fi practic inexistentă. Deoarece singura cale de comunicaţie dintre procesor şi exterior este reprezentată de *magistrala de date* a sistemului, (utilizând şi magistralele de adrese şi control), este necesar să se utilizeze *circuite speciale* care să *convertească* informaţiile diverse, de la o gamă foarte largă de echipamente şi componente de intrare şi ieşire, la semnale *compatibile* cu cele de pe magistralele sistemului microprocesor. Aceste *circuite*, care implementează o *funcţie de compatibilizare*, se numesc *module de interfaţă* (*module de intrare-ieşire*). Astfel, comunicaţia cu unitatea centrală reprezintă transferul datelor prin aceste interfeţe, realizată într-un mod selectiv şi controlat uzual de către microprocesor. *Convenţiile* utilizate pentru a implementa această comunicaţie, incluzând temporizările semnalelor, controlul procesului de transfer de date, modul de reprezentare (codificare) a datelor, alcătuiesc aşa-numitul *protocol al comunicaţiei*.

Funcţia de bază a modulelor de interfaţă va fi aceea de a *converti* semnalele de pe magistralele CPU pentru seturile de *porturi de intrare sau ieşire*. Un *port* reprezintă astfel o colecţie de componente ale sistemului la care se pot conecta echipamente externe (*periferice*). Gama arhitecturilor şi a implementărilor diverselor sisteme cu microprocesoare este extrem de largă, fără o standardizare a funcţiilor şi/sau semnalelor magistralelor acestora. Aceste sisteme diverse vor trebui să interacţioneze cu o gamă largă şi nu neapărat unitară a echipamentelor periferice, utilizând o paletă importantă de semnale, viteze de lucru şi protocoale de comunicaţie.

Compatibilitatea între operaţiile interne ale procesorului şi semnalele externe se va asigura de o mare varietate de tipuri de *interfeţe*. Uzual, firmele producătoare de microprocesoare vor asigura şi *circuitele specializate*, compatibile cu tipurile de microprocesoare produse de către aceştia, pentru a se implementa diversele *interfeţe specializate*.

Astfel, comunicaţia între microprocesor şi echipamentele externe are loc în *două etape succesive*:

1. transferul are loc *între CPU şi interfaţă*, denumită *operaţie de intrare-ieşire*;
2. comunicaţia are loc între *interfaţă şi periferic*, denumită *operaţie de transmisie* a informaţiilor (fig. 4.).

În consecinţă, ***interfeţele de intrare/ieşire*** reprezintă calea de legătură ce asigură transferul de date între microprocesor şi exteriorul lui. Există numeroase posibilităţi pentru realizarea acestor legături. În funcţie de configuraţia sistemului şi de cerinţele aplicaţiei, se pot utiliza structuri mai simple sau deosebit de complexe.

Elementele de intrare asigură preluarea informaţiei dintr-o formă accesibilă omului (reprezentări: caractere, litere, cifre, până la imagini tv.), sau de la diverse elemente ale instalaţiilor industriale (semnalizări, alarme etc.), iar elementele de ieşire furnizează informaţii utilizabile de către operatorul uman (mesaje tipărite, semnalizări audio-vizuale, sinteză de voce), cât şi pentru comanda şi acţionarea elementelor echipamentelor industriale.

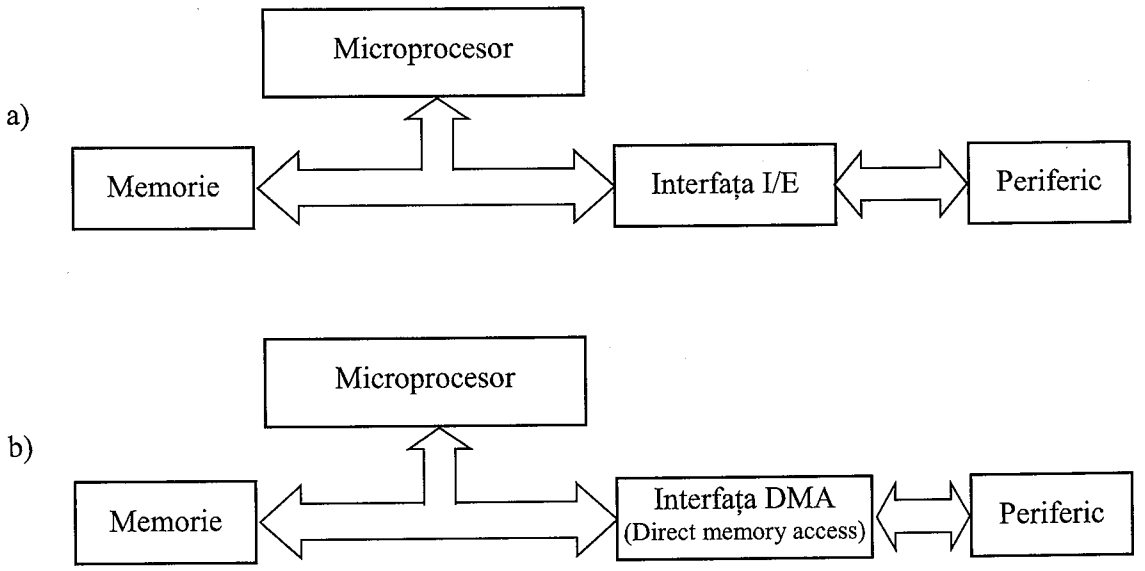


Fig. 4. Moduri de interfațare:
(a) sub controlul microprocesorului; (b) prin acces direct la memorie

Semnale exterioare de o mare varietate a formelor de prezentare (analogice, numerice, de alta natură decât electrice) trebuie convertite la formele de informație, limitate, acceptate de către microprocesor. Conversii similare trebuie efectuate la ieșirea spre exteriorul sistemului microprocesor. Trebuie adaptate și vitezele de lucru diferite ale perifericelor, respectiv ale microprocesorului. Uzual, acesta lucrează mult mai rapid, putând efectua numeroase operații, în timp ce, de exemplu, se preia un caracter de la o consola (prin apăsarea unei clape a acesteia) și se pregătește într-o formă adecvată citirii de către microprocesor. De asemenea, există și echipamente foarte rapide (discuri magnetice, de exemplu), care pot furniza/prelua informații către/de la memoria sistemului, cu viteze mai mari decât cele ce pot fi obținute dacă transferul se efectuează de către microprocesor (sub comanda acestuia). În aceste cazuri, se utilizează așa-numitele tehnici de transfer al informației prin ***acces direct la memorie (DMA - Direct memory access)***, pentru care accesul între interfață și memoria sistemului se face sub controlul unui circuit specializat, fără controlul programului microprocesorului.

Se pot adopta soluții *hardware-software* care să asigure o performanță optimă a sistemului. În funcție de cerințele acestuia, de numărul de elemente pe care le supervizează microprocesorul, se vor adopta diverse tehnici de lucru.

Exista trei *moduri principale de lucru cu elementele de intrare/ieșire*, permițând controlul și sincronizarea transferului de date:

(a) Operații de intrare/ieșire efectuate sub controlul programului;
(b) Operații de intrare/ieșire prin întreruperi;
(c) Operații de intrare/ieșire prin acces direct la memorie.

Tipul de operație I/E utilizată într-o aplicație va depinde de rata de transmitere a datelor, de întârzierea maximă cu care microprocesorul poate prelua/transmite datele din momentul disponibilității echipamentului care le vehiculează, de posibilitatea executării (intercalat cu operațiile I/E) a altor operații ale unității centrale.

Din punctul de vedere al programării microprocesorului pentru *operații de I/E*, acestea se pot efectua fie prin executarea de ***instrucțiuni specifice de I/E***, fie prin ***instrucțiuni proprii operării cu elemente de memorie***. Pentru diferențierea între diversele echipamente ale sistemului, acestea sunt selectate în mod unic, conform informației de pe magistralele de adrese și de control, în mod asemănător selecției celulelor de memorie.

Porturile sistemului vor prelua sau transmite informația sub formă de *cuvinte de date*, de lungime specifică procesorului (8, 16 biți). Accesul la magistrala de date se va face tot prin *circuite tampon*. Astfel, tehnicile generale de *adresare și selecție de cip* ale *porturilor* vor fi similare celor de la *memorii*, cu unele particularități specifice.

Ca o *mențiune specială*, trebuie remarcat că, spre deosebire de memorii, unde la o adresă se va găsi o singură celulă de memorie, aceeași și pentru operațiile de scriere și pentru cele de citire, la operațiile cu *porturi*, este permis, și se și întâlnesc situații în care o *aceeași adresă* este utilizată pentru *două interfețe diferite*, una de *intrare* sau alta de *ieșire*. Aceasta deoarece pot exista porturi doar de intrare, și altele doar de ieșire, deci cu flux unic al datelor, doar *de la* sau doar *spre* microprocesor.

În continuare se vor analiza cele trei tipuri de bază de organizare a transferurilor de date în operațiile de I/E ale microprocesorului.

1 Operații I/E efectuate sub controlul programului (tehnici polling)

Una dintre cele mai accesibile și utilizate metode de efectuare a operațiilor de I/E este sub controlul nemijlocit al programului microprocesor. Astfel, prin instrucțiuni specifice, care determina selecția porturilor de I/E ale sistemului, microprocesorul efectuează operații de transfer de date între aceste interfețe și CPU.

Există două *tipuri de bază* a informației transmisă între *microprocesor și portul de intrare/ieșire*, și anume ***cuvintele de control*** și ***cuvintele de date*** (fig. 5).

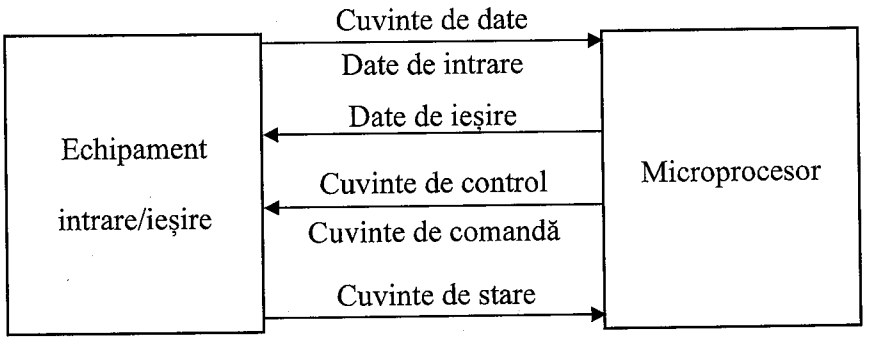


Fig. 5. Fluxul de informații între microprocesor și o interfață de I/E

Cuvintele de control sunt utilizate pentru *sincronizarea* operațiilor portului și ale microprocesorului și *setarea/testarea* unor parametri, operații necesare în vederea transmiterii ***cuvintelor de date***.

Folosite în operații de *scriere spre port*, *cuvintele de control* sunt denumite ***cuvinte de comandă*** a *portului* și permit, prin afectarea pentru fiecare bit al cuvântului, a unei semnificații unice, *programarea portului* (pentru porturile care pot fi programate). Astfel, se pot *iniția* unele operații sau *seta* unii parametri (oprirea unui motor, modificarea ratei de transmisie a unor date, ștergerea unui indicator de eroare, etc.).

Citirea cuvintelor de control, denumite în acest caz, ***cuvinte de stare***, permite, prin testarea prin program a biților cuvântului citit de la port, *analizarea* stării acestuia (dacă are sau nu informație disponibilă, dacă poate prelua alt cuvânt de date, dacă transferul s-a efectuat corect etc., în funcție de posibilitățile și funcțiile portului) și luarea de *decizii* adecvate de către microprocesor (se vor executa în mod corespunzător acele secțiuni din programul sistemului care tratează situația existentă).

La efectuarea *operațiilor de intrare/ieșire sub controlul programului*, se pot utiliza:

1. ***Instrucțiuni specifice de I/E.*** În acest caz, microprocesorul este prevăzut cu semnale pe magistrala de control, diferite pentru operațiile cu memoria, respectiv cu porturile. Se conectează atât memoria cât și porturile la magistralele de adrese și de date ale sistemului, dar se utilizează semnale de control distincte pentru cele doua categorii de componente ale sistemului. Figura 6 prezintă modul de structurare a sistemului în acest caz. Utilizând instrucțiunile specifice de I/E, se generează, prin program, la execuția acestor instrucțiuni semnalele de control asociate, selectând porturile de I/E și efectuând transferul dorit al informațiilor.

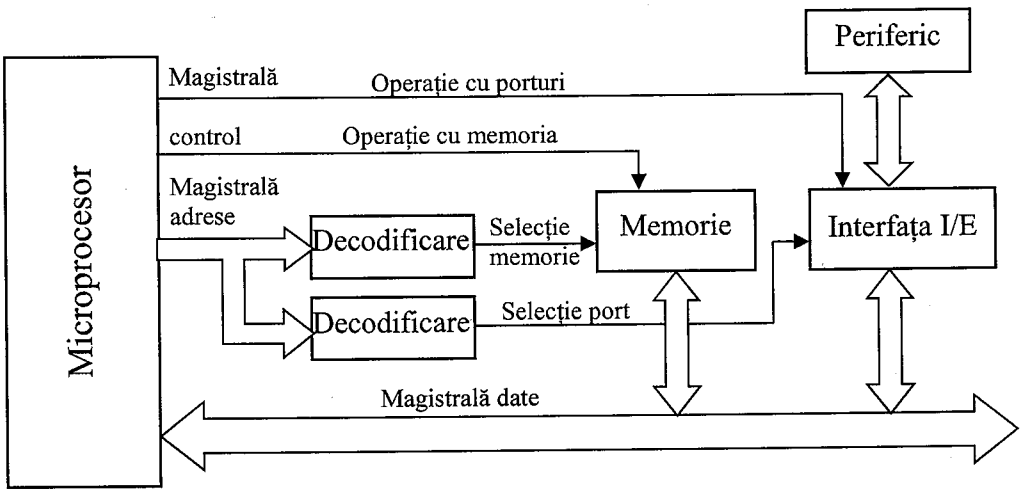


Fig. 6. Utilizarea operațiilor specifice de I/E, distincte de cele cu memoria

Există două moduri principiale de implementare a acestui tip de operații I/E:

1.1. Câte o instrucțiune unică pentru fiecare operație de I/E menționată, ***utilizând minimum o singură adresă*** pentru un port. În acest caz, există *patru instrucțiuni tipice*:

- ***citire de date (citire cuvânt de date);***
- ***scriere de date (scriere cuvânt de date);***
- ***transmitere comandă (scriere cuvânt de comandă);***
- ***citire stare (citire cuvânt de stare).***

Acest tip de operație cu porturile ar necesita un număr de *minimum 2 semnale de control*, pentru a diferenția, la o operație cu același port, *sensul* și *tipul* instrucțiunii executate la un moment dat.

Figura 7 prezintă structura sistemului în acest caz. Metoda ***nu*** este utilizată în mod tipic, în sistemele microprocesor uzuale.

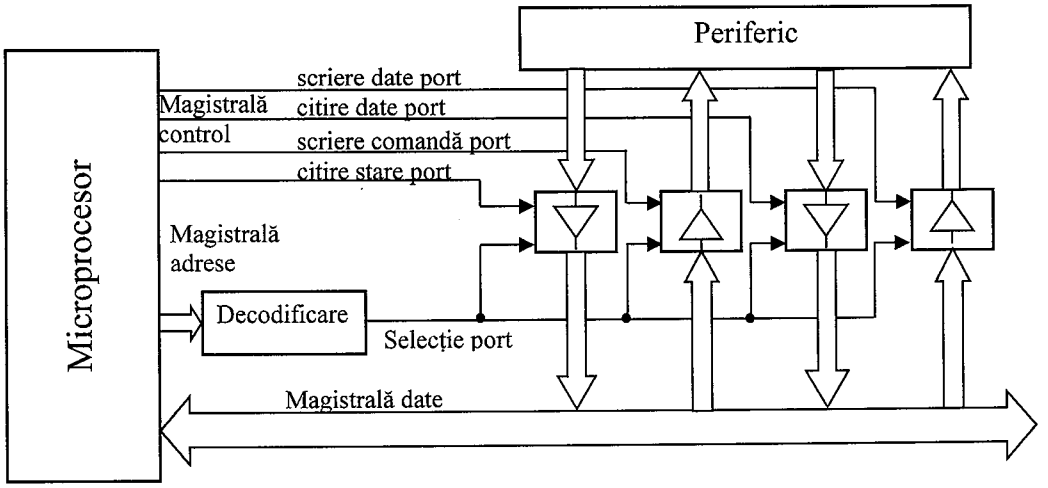


Fig. 7. Utilizarea unei singure adrese de port și a patru semnale de control I/E

1.2. Două instrucțiuni I/E, una pentru ***intrare***, alta pentru ***ieșire***, *atât pentru cuvintele de date, cât și pentru cele de control.* Se vor utiliza, în acest caz, ***minimum două adrese pentru un port,*** pentru a *diferenția* operația ce se efectuează, referitor la *date* sau *control*. Cele *două instrucțiuni tipice*, ce se definesc în acest caz, vor fi:

- citire informație (cuvânt de dată sau de stare);

- scriere informație (cuvânt de dată sau de comandă).

Pentru acest tip de operație de I/E, este necesar *minimum un semnal de control* care să diferențieze *sensul* transferului datelor (*citire* sau *scriere*), care corelat cu *adresa* de la/la care se face transferul, să poată defini în mod unic operația ce se execută. Metoda este una dintre cele mai întâlnite tehnici de utilizare a porturilor de I/E. Figura 8 prezintă structura de principiu a sistemului în acest caz.

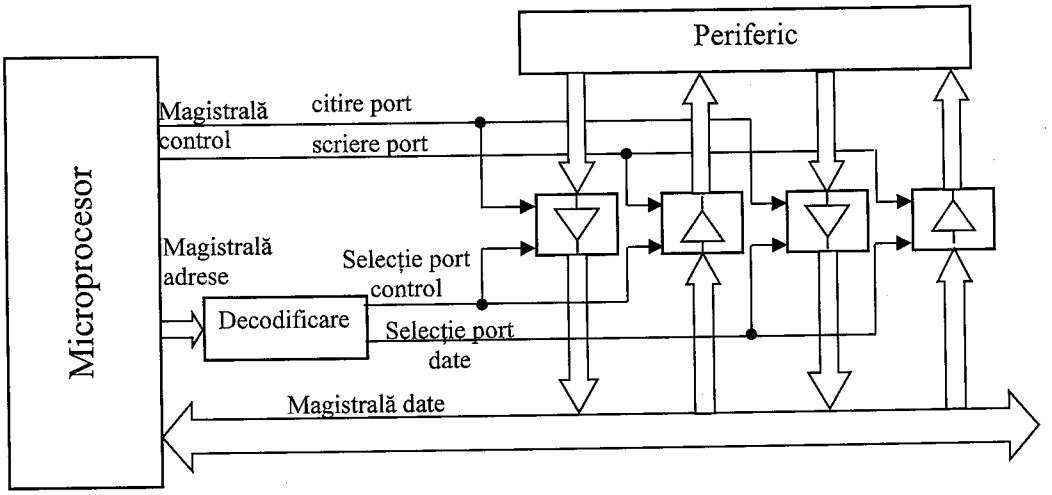


Fig. 8. Utilizarea a două adrese I/E port și a două semnale de control

2. ***Nu se folosesc instrucțiuni separate de I/E.*** Se va ***adresa portul ca o celulă de memorie.***

Se configurează memorie în sistem, mai puțină decât capacitatea maximă a acestuia. *La adresele rămase libere se va asigura selecția portului respectiv.* (Deci, la apariția semnalelor de adresă corespunzătoare și a celor de control, proprii lucrului cu celula de memorie respectivă, se va selecta de fapt și se va vehicula informație cu portul dorit). Deși memoria sistemului devine astfel mai mică, acesta este un impediment minor, compensat în primul rând de faptul că, paleta de posibilități oferită de instrucțiunile de tip lucru cu memoria este mult mai largă deoarece, microprocesorul va efectua operații cu o celulă de memorie, nu cu un port.

Operațiile tipice pentru a efectua transferul datelor vor fi în acest caz:

(1) încărcare dată (citire cuvânt de dată sau stare);

(2) memorare dată (scriere cuvânt de dată sau comandă).

Metoda este desemnată cu termenul de "memory-mapped I/O" - intrare/ieșire "mapată" în memorie. Figura 9 prezintă structura de principiu a sistemului în acest caz.

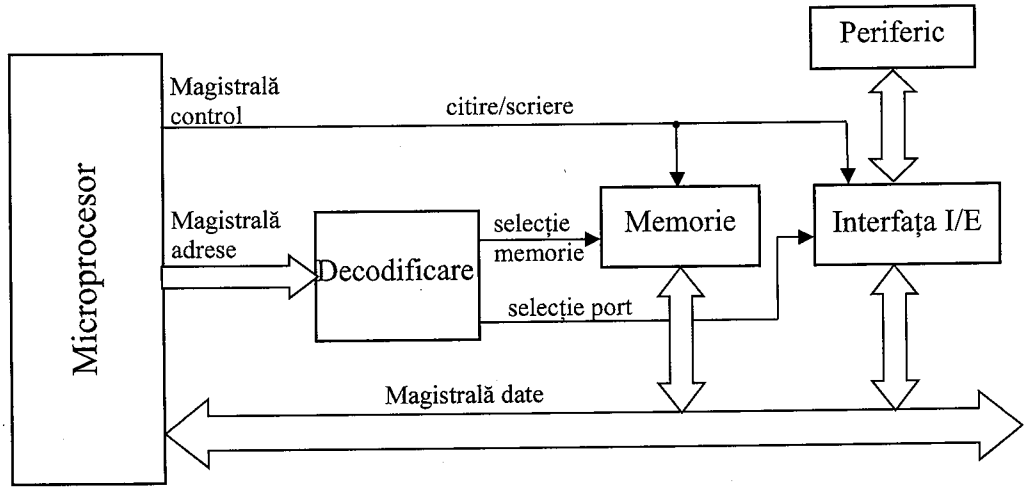


Fig. 9. Utilizarea operațiilor I/E, de tipul „memory mapped I/O"

O *comparație* succintă a celor două metode prezentate anterior poate evidenția elementele esențiale ale acestora, de care se ține seama la configurarea structurii sistemului microprocesor.

Astfel, utilizarea *instrucțiunilor și a semnalelor specifice de I/E*, are avantajul unei mai bune organizări și clarificări a programului, cât și a configurării schemei de selecție a memoriei și porturilor sistemului. În schimb, metoda "*memory mapped I/O*" permite utilizarea instrucțiunilor uzuale de lucru cu memoria, cu avantajele aferente (mult mai multe instrucțiuni aferente, moduri de adresare complexe, etc.). Pe de altă parte însă, metoda implică complicații în ceea ce privește schemele de adresare și selecție a memoriei și interfețelor sistemului. În plus, o problemă suplimentară în acest caz o reprezintă metodologia utilizată la *depanarea* programului, fiind mai dificilă diferențierea operațiilor cu memoria de cele cu porturi, de vreme ce instrucțiunile utilizate la aceste operații coincid (doar adresele diferențiind cele două tipuri de componente).

Capacitatea de adresare a porturilor, pentru tipurile uzuale de microprocesoare, variază de la 256 de porturi I/O (uzual la microprocesoarele de 8 biți), până la 64k porturi de I/O sau mai mult. În cazul utilizării porturilor organizate ca "*memory mapped I/O*", din totalul spațiului de memorie configurabil pentru microprocesorul respectiv se va decide combinația *memorie/porturi* optimă pentru aplicația respectivă. Uzual, atât memoria cât și porturile sistemului se organizează în acest caz în *zone compacte de adrese (blocuri)*.

Modalitățile de utilizare în program a *cuvântului de control* sunt, în principal, date de o secvență de program, de genul:

1. *se scrie cuvântul de comandă la port, pentru a cere transferul de cuvânt de date;*
2. *se citește cuvântul de stare de la port;*
3. *se verifică biții de stare ce indică posibilitatea transferării datelor;*
4. *se reiau pașii 2-3, până ce portul este gata de transfer;*
5. *se citește (scrie) cuvântul de date.*

O astfel de secvență poate apărea pe parcursul programului principal al microprocesorului.

O soluție mult mai avantajoasă, din punctul de vedere al timpului de răspuns la evenimente (când este posibil transferul), care permite implementarea unor programe mai clar și corect elaborate, este oferită de către efectuarea operațiilor I/E prin întreruperi.

2. Operații de intrare/ieșire efectuate prin întreruperi

Operarea prin ***întreruperi*** *este bazată pe utilizarea unor semnale de control specifice, lansate de la periferice, către microprocesor, anunțând un eveniment semnificativ la nivelul acestora* (date disponibile pentru a fi preluate, posibilitatea acceptării de noi date de la procesor, semnalizări primite din exteriorul sistemului etc.).

Lucrul în întreruperi permite *operarea aparent simultană a mai multor programe* distincte ale microprocesorului.

În principal, această tehnică este aplicată pentru controlul și lucrul cu echipamentele periferice și pentru implementarea sistemelor ce operează ***în timp real*** (efectuând anumite operații măsurători, calcule, comenzi), la intervale predeterminate de timp. Bazându-se pe faptul că, în general, procesorul are viteza de lucru net superioară perifericelor, el lucrează cu fiecare doar la "*cererea*" acestuia, putând astfel să servească mai multe dintre ele, la vitezele lor de lucru.

Sistemul de întreruperi va permite acest mod de soluționare a problemei. Într-o formă simplificată, apariția unei întreruperi la procesor determină următoarele evenimente:

1. terminarea instrucțiunii curente a microprocesorului;
2. memorarea valorii contorului de program (PC) în stivă;
3. încărcarea contorului de program (PC) cu o adresă de program, predefinită;
4. inhibarea întreruperilor și continuarea execuției de la nouă adresă a contorului de program (PC).

După tratarea întreruperii (zona de program la care s-a sărit la întrerupere este scrisă special pentru evenimentul apărut) se execută o instrucțiune specifică de *revenire în programul întrerupt (se reface contorul de program, PC, din stivă)*, continuându-se execuția acestuia (printr-un proces asemănător celui întâlnit la revenirea din subrutinele obișnuite, apelate prin program). Figura 10 prezintă un exemplu în acest sens.

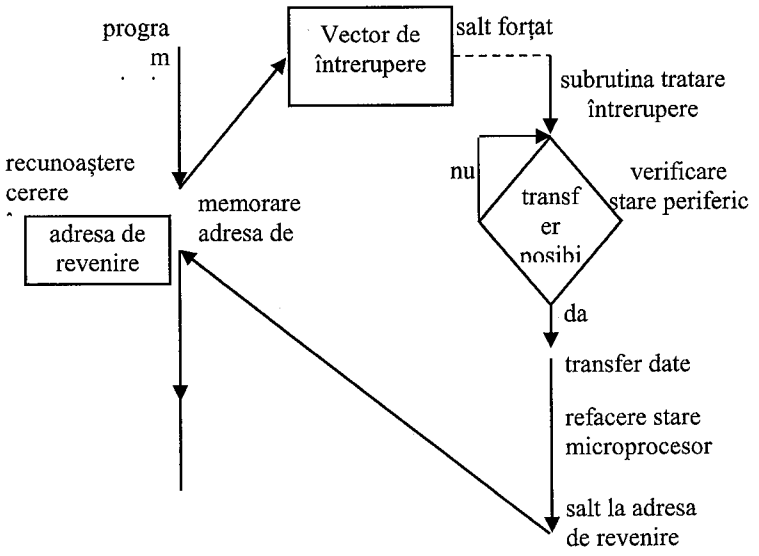


Fig. 10. Operații I/E controlate prin întreruperi

Inhibarea întreruperilor, la acceptarea cererii de întrerupere, evită tratarea unei noi întreruperi până ce nu s-a rezolvat cea în curs de tratare. Uzual, sistemul permite existența *mai multor întreruperi*, de la diverse echipamente. Există diverse *modalități* pentru a determina saltul la adresa subrutinei aferente întreruperii respective. Astfel, este posibilă existența *mai multor pini distincți ai procesorului*, fiecare corespunzător unei anumite întreruperi. Apariția pe unul dintre aceștia a unui semnal activ va implica *saltul programului curent, la o adresă fixă, cunoscută de către programator*, și unde a fost *plasată subrutina de tratare a întreruperii* respective.

În cazul existenței unei *singure linii de întrerupere* la microprocesor, există *două moduri* principale de recunoaștere a sursei care a generat întreruperea:

1. ***Scanarea echipamentelor***. Saltul în subrutina de tratare a întreruperii este la adresă fixă. Aici se testează *cuvântul de stare* al fiecărui echipament de I/E pentru a se determina, prin program, care echipament de I/E a generat întreruperea.

2. ***Utilizarea vectorilor de întrerupere***. În aceste structuri, în momentul în care microprocesorul este disponibil pentru acceptarea întreruperii, el confirmă acest accept setând un semnal către echipamentele ce solicită întreruperea. Primind acceptarea, acestea vor genera pe magistrala de date un *cod de instrucțiune de apelare de subrutină*, cod generat de către *logica de întrerupere*. Acest *cod* conține câte o *adresă* pentru fiecare echipament în parte. În general, aceste adrese sunt *localizate secvențial în memoria program* și formează așa-numitul ***vector de întrerupere*** al sistemului.

O problemă importantă, la tratarea întreruperilor, este reprezentată de ***prioritatea*** cu care se execută acestea. Este posibilă apariția *simultană* a mai multor întreruperi, sau apariția unei întreruperi ce trebuie deservită, chiar în timp ce se deservea o întrerupere mai puțin urgentă.

Inhibarea, pur și simplu, a întreruperilor elimina aceasta posibilitate, ceea ce poate să nu ofere performanțe satisfăcătoare sistemului. De aceea, acordarea unei priorități fiecărei cereri de întrerupere și soluționarea acestora conform priorității avute sunt foarte importante.

Se poate rezolva problema priorității atât prin *software*, cât și prin *hardware*.

- ***Priorități software***. Ordinea în care se testează starea echipamentelor de I/E (Fig. 11) corespunde priorității acestora. Detectarea, în ordinea testării, a unei cereri de întrerupere implică saltul la rutina corespunzătoare acestei întreruperi, care va avea prioritate.

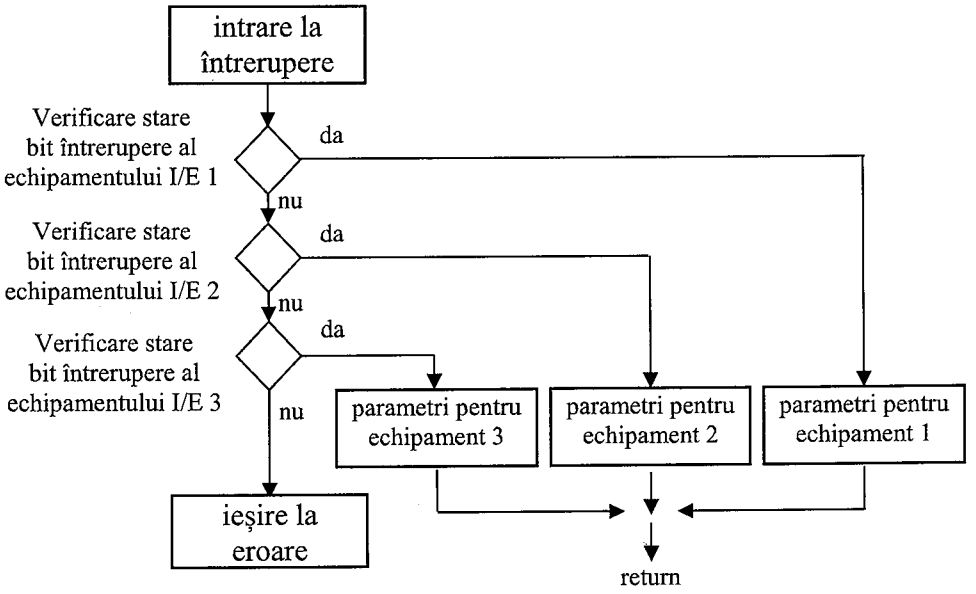


Fig. 11. Testarea cererii de întrerupere a echipamentelor de I/E în subrutina de întrerupere

- ***Priorități hardware***. Un semnal de control trimis de către microprocesor în exterior trece prin logica de control a fiecărui echipament (Fig. 12). Dacă întreruperile sunt *inhibate (mascate)*, acest semnal nu permite generarea de întreruperi către procesor. Când întreruperile sunt *activate*, semnalul trece pe rând prin echipamentele ce nu cer întrerupere. Ajungând la cel care cere întrerupere, logica implementată blochează generarea de întreruperi de la următoarele echipamente şi determină generarea întreruperii către procesor. Astfel, poziția în lanțul de echipamente determină prioritatea acestora.

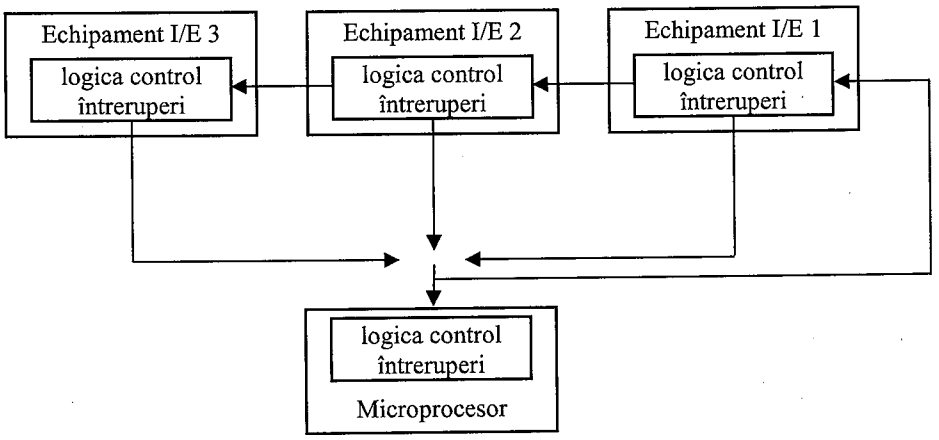


Fig. 12. Implementarea hardware a logicii de priorități la întrerupere, de la echipamentele de I/E

O mențiune trebuie făcută şi referitor la ***salvarea stării programului*** la apariția unei întreruperi. Saltul la rutina de întrerupere va implica alterarea, prin operațiile efectuate acolo, a informației din registrele procesorului.

Un **vector de întrerupere al microprocesorului *8086*** reprezintă un pointer (4 octeți) conținând adresa rutinei de tratare a întreruperii asociate. Cuvântul cel mai semnificativ al pointerului conține adresa de baza de segment, iar cuvântul mai puțin semnificativ conține ofsetul față de începutul segmentului al subrutinei respective, astfel încât, următoarea instrucțiune ce se va extrage și executa va fi prima instrucțiune din cadrul subrutinei de întrerupere. Cum fiecare intrare în tabela vectorilor de întrerupere are o lungime de 4 octeți, CPU calculează locația vectorului asociat unei întreruperi anume prin simpla înmulțire cu 4 a numărului (cuprins între 0 și 255) ce reprezintă **tipul** întreruperii respective.

Primii 5 vectori de întrerupere sunt **dedicați** întreruperilor generate prin software și unei întreruperi externe (*NMI*):

0—divide error
1—single-step
2—NMI
3—breakpoint
4—overflow,

următorii 27 vectori de întrerupere (deci până la locația 07FH) sunt **rezervați** de firma *INTEL*, iar restul, adică vectorii 32 până la 255, sunt la dispoziția utilizatorului.

Subrutina de tratare a întreruperii

Subrutina de tratare a unei întreruperi trebuie să salveze toate registrele pe care le utilizează înainte de a le inițializa și să le refacă înainte de terminarea rutinei. Ca urmare, în cazul procesării unor întreruperi simultane, stiva trebuie să aibă la dispoziție un spațiu suficient pentru salvările succesive ce pot apare. O altă problemă ce trebuie avută în vedere este faptul că, dezactivarea întreruperilor externe într-o rutină de tratare a unei întreruperi poate duce la pierderea acestora, dacă rutina are un cod prea mare.

Toate subrutinele de întrerupere trebuie să se termine cu instrucțiunea *IRET* (***I**nterrupt **RET**urn*), instrucțiune a cărei execuție se bazează pe ipoteza că, stiva este în aceeași condiție în care a fost la intrarea în procedură.

Ceea ce se procesează efectiv în cadrul unei rutine de tratare a întreruperii depinde de aplicația respectivă. De exemplu, în cazul unei proceduri care servește o cerere de întrerupere externă, prima acțiune ce trebuie executată (dacă acest lucru nu se întâmplă automat) este trimiterea unei comenzi către echipamentul ce este servit, prin care să se determine retragerea cererii de întrerupere. Următoarea acțiune uzuală este de a se citi starea echipamentului pentru a se identifica motivul pentru care s-a solicitat o întrerupere. În final, în funcție de cauza respectivă, se comandă execuția unor operații corespunzătoare.

3 Operații de I/E efectuate prin *acces direct la memorie*

Accesul direct la memorie (*Direct memory access - DMA*) este utilizat în cazul transferurilor foarte rapide de date, care se efectuează de la/la periferice cu viteze de lucru superioare celei a microprocesorului. Se permite astfel transferul de informații fără a se opera sub controlul programului procesorului. În cazul transferurilor *DMA*, transferul de date între periferice și memoria sistemului este coordonat de către un controler specializat pentru aceste transferuri, programat în prealabil de către microprocesor (în vederea cunoașterii adreselor unde/de unde să se facă transferul de date de la/la periferice) (Fig. 13.).

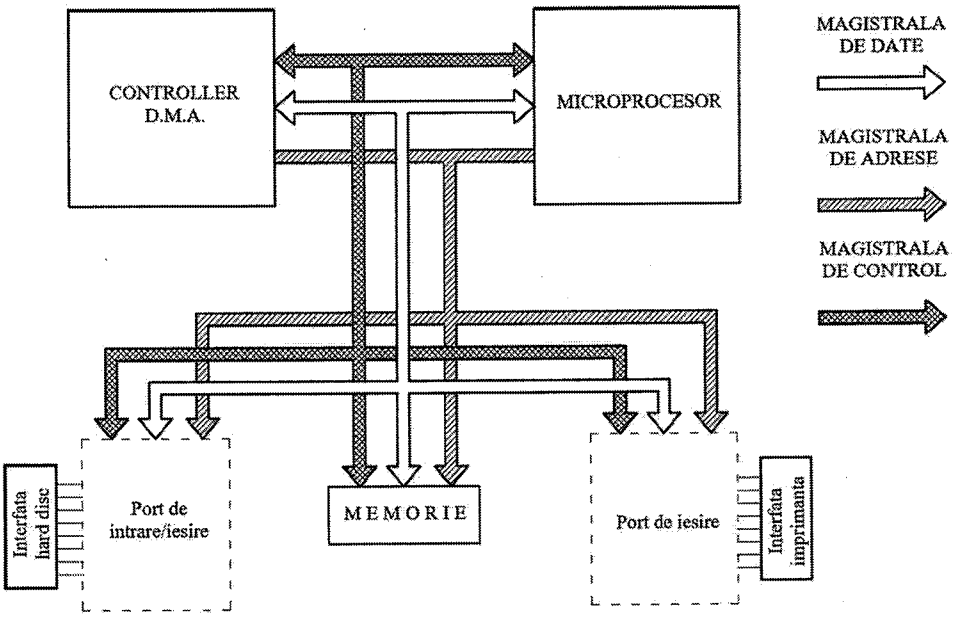


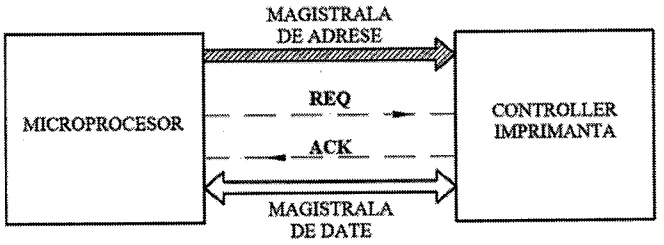
Fig. 13. Transferul de date prin acces direct la memorie

In cazul in care apare necesitatea transferului unui volum mare de date de la memorie la sau de la unul dintre periferice, microprocesorul inițializează controllerul DMA cu adresa de start a domeniului la/de la care trebuie să facă transferul, numărul de octeți și sensul în care se face transferul. Spre exemplu, dacă se dorește trimiterea unui fișier din memorie la imprimantă, se specifică adresa de început a fișierului în memorie și lungimea fișierului, precum și faptul că fișierul va fi trimis din memorie la portul de ieșire corespunzător imprimantei. În continuare este așteptat un moment în care microprocesorul nu are nevoie de transferuri de date pe magistrală și transferă controlul magistralelor controllerului DMA. Din acest moment, microprocesorul nu va mai utiliza magistralele până când se va finaliza transferul de date. Controllerul DMA este și el un microprocesor dar cu un ALU simplificat, deoarece nu are altă sarcină decât să facă transferuri de date. Acesta controlează funcționarea magistralelor la fel cum o face și microprocesorul, dar realizează doar operații de citire-scriere. În momentul când pe magistrala de control s-a specificat faptul că i s-a transferat controlul magistralelor, controllerul DMA începe să facă transferul de date cerut în momentul inițializării. Când a terminat transferul de date comunică microprocesorului tot pe magistrala de control faptul că a încheiat sarcina atribuită și microprocesorul poate utiliza din nou magistralele.

Un astfel de mecanism de transfer este foarte eficient atunci când microprocesorul este prevăzut cu memorie cache. În intervalul de timp în care se face transferul de date microprocesorul poate continua activitatea utilizând instrucțiunile și datele stocate în memoria cache, deci apare o îmbunătățire substanțială a gradului de utilizare a timpului de calcul al microprocesorului. De asemenea, acest mecanism este foarte util in cazul procesoarelor multitasking, când transferul de date asociat unui task poate fi suprapus peste timpul de calcul asociat unui alt task. Din punct de vedere software, coordonarea transferului DMA este realizata de către sistemul de operare.

Protocolul handshaking de transfer de date

Acest protocol este utilizat pentru transferul de date între microprocesor și periferice, mai ales atunci când frecvențele de ceas ale microprocesorului și perifericului respectiv diferă foarte mult. Spre exemplu frecvența de ceas a unui microprocesor poate fi în prezent de ordinul 1-2 GHz iar frecvența de ceas a unei imprimante matriceale cu ace poate fi de ordinul a 10-100 MHz. Unele dintre semnalele de comandă și de stare sunt de obicei activate pe durata unei perioade de ceas și sunt citite de echipamentul corespondent de obicei pe unul dintre fronturile impulsurilor sale de ceas. Datorita diferenței mari dintre cele două frecvențe, este posibil ca activarea unor semnale de comandă și de stare ale echipamentului mai rapid să treacă neobservate de echipamentul mai lent, astfel încât apar situații anormale în transferul de date între cele două echipamente.



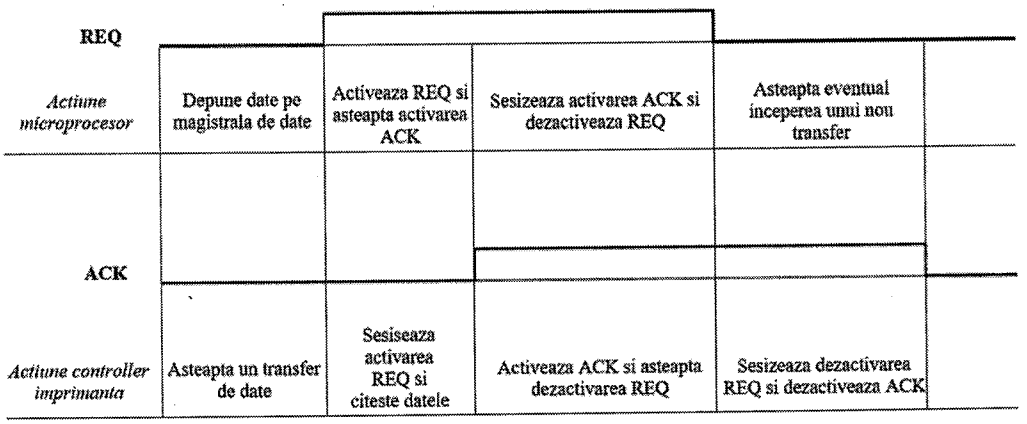


Fig. 14. Desfășurarea protocolului handshaking

Protocolul handshaking realizează sincronizarea transferului de date între astfel de periferice cu viteze foarte diferite de lucru. În cadrul magistralei de control vor exista două linii de semnal, denumite REQ ("request") și ACK ("acknoledge"). Linia REQ este controlată de echipamentul care emite datele și linia ACK este controlată de echipamentul receptor. Dacă transferul de date este bidirecțional atunci mai este necesară încă o pereche de linii REQ și ACK. După cum se poate observa este necesar ca ambele echipamente între care se comunică datele să fie echipamente inteligente, adică să fie dotate cu un microprocesor sau microcontroler care să îi coordoneze activitatea.

In figura 14 este exemplificat transferul între un microprocesor rapid și o imprimantă lentă. Atâta timp cât nu se desfășoară nici un transfer între microprocesor și controllerul de imprimantă semnalele REQ și ACK sunt dezactivate. În momentul în care microprocesorul are de transmis date la imprimantă pune adresa controllerului de imprimanta pe magistrala de adrese și datele de transferat pe magistrala de date, după care activează semnalul REQ, semnalizând astfel controlerului de imprimanta faptul că are de preluat date de pe magistrala de date. Când controllerul de imprimantă sesizează activarea semnalului REQ citește datele de pe magistrala de date, după care activează semnalul ACK și informează astfel microprocesorul că a preluat datele de pe magistrală. În acest interval de timp, microprocesorul trebuie să mențină fixe datele pe magistrala de date și adresa controlerului de imprimantă pe magistrala de adrese. Deoarece microprocesorul este foarte rapid, modificarea datelor și adresei după o singură perioadă de ceas ar putea face ca datele care trebuie trimise imprimantei să nu poată fi citite de aceasta, deoarece se modifică cu o viteză prea mare. Rezultatul citirii ar fi eronat. El trebuie să aştepte controlerul de imprimantă să activeze semnalul ACK. De asemenea, o menținere activa a semnalului REQ doar pe durata unei singure perioade de ceas a microprocesorului ar putea trece neobservata de controllerul de imprimanta. Trecerea la un nou transfer de date între microprocesor și controller trebuie făcută atunci când ambele echipamente au sesizat finalizarea transferului curent și de asemenea, au fost înștiințate de faptul că și echipamentul corespondent a luat la cunoștință de finalizarea transferului curent. Acceptarea trimiterii de noi date atât timp cât este activ semnalul ACK ar conduce la desincronizarea transferului. Activarea semnalului REQ și modificarea datelor și adresei de pe magistrale ar putea trece neobservate de controlerul de imprimantă sau s-ar modifica cu o viteză prea mare, astfel încât controlerul de imprimantă ar pierde din datele transmise. În consecință, atunci când microprocesorul sesizează activarea semnalului ACK dezactivează semnalul REQ, înștiințând controlerul de magistrala că din punctul său de vedere transferul curent s-a încheiat. Microprocesorul trebuie să aibă confirmarea din partea controllerului de magistrală a faptului că a primit acest mesaj. Confirmarea este realizată de controlerul de imprimantă în momentul în care a sesizat dezactivarea semnalului REQ, prin dezactivarea semnalului ACK. În acest moment transferul curent s-a încheiat și poate începe un nou transfer de date. Implementarea acestui protocol poate fi făcută în toate cele trei variante de transferuri de date – tehnica de polling, tehnica întreruperilor sau tehnica de acces direct la memorie.

Operațiile *DMA* se întrepătrund cu operațiile normale ale sistemului. Există *trei metode* uzuale pentru *transferuri DMA*. Una dintre ele oprește microprocesorul, pe când celelalte două interferează cu operațiile sale curente, după cum urmează:

1. ***Metoda de oprire a activității procesorului (HALT)*** implică, la apariția unei cereri de acces *DMA*, *terminarea efectuării instrucțiunii curente* a procesorului, urmată de *trecerea tuturor pinilor acestuia* (pe magistralele de date, adrese și control) *în starea de mare impedanță* și permite *efectuarea transferurilor DMA* pe aceste magistrale. *Dezavantajul* metodei constă în faptul că vor fi necesare câteva cicluri-mașină din momentul apariției cererii de transfer,

până la posibilitatea satisfacerii acesteia. Uneori, aceasta întârziere poate fi prea mare. *Avantajos* este faptul că, odată obținut, controlul magistralelor poate fi menținut pentru transferuri oricât de lungi.

2. ***Metoda de oprire a microprocesorului în cadrul instrucțiunii curente***, *la terminarea ciclului de instrucțiune curent*. Se obțin timpi de răspuns mult mai buni, deci performanțe sporite. La o astfel de întrerupere a execuției programului se pot transfera seturi mari de date.

3. ***Metoda prin care procesorul nu este oprit și nici încetinit la executarea programului***. Metoda este complet "transparentă" pentru procesor, este cea mai rapidă, dar și cea mai critică în același timp. ***Operațiile microprocesorului și ale DMA sunt multiplexate***, luând în considerare faptul că transferurile unității centrale apar doar pentru anumite faze ale ceasului sistemului.

Utilizarea unei anumite tehnici *DMA* se va face printr-un compromis între complexitatea structurii *hardware*, viteza de execuție necesară și rata de transfer *DMA* ce trebuie obținută.