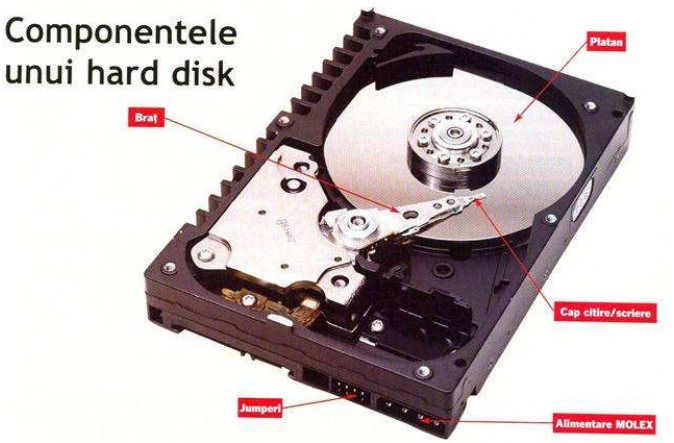
UNIVERSITATEA TEHNICĂ A MOLDOVEI

VICTOR ABABII
VIORICA SUDACEVSCHI

ECHIPAMENTE PERIFERICE

PROGRAMAREA OPERAŢIILOR DE INTRARE/IEŞIRE

Îndrumar de laborator



Chişinău 2016

UNIVERSITATEA TEHNICĂ A MOLDOVEI

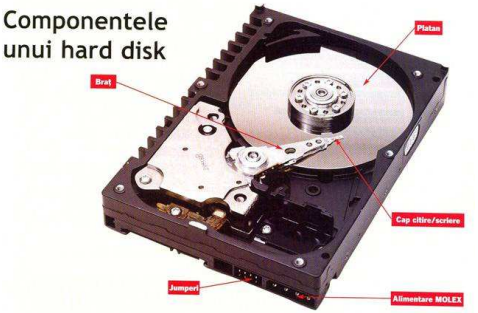
FACULTATEA CALCULATOARE, INFORMATICĂ ŞI MICROELECTRONICĂ

CATEDRA CALCULATOARE

ECHIPAMENTE PERIERICE

PROGRAMAREA OPERAŢIILOR DE INTRARE/IEŞIRE

Îndrumar de laborator



Chişinău
U.T.M.
2016

Îndrumarul este adresat studenţilor cu specializarea ***526.1 „Calculatoare”*** de la Facultatea Calculatoare, Informatică şi Microelectronică pentru a fi utilizat la executarea lucrărilor de laborator la disciplina ***ECHIPAMENTE PERIFERICE***.

Fiecare lucrare de laborator conţine consideraţii teoretice necesare executării lucrării respective, tema pentru acasă şi etapele pentru executarea lucrării. Lucrările conţin câte două părţi: prima bazată pe analiza sistemelor hard, a doua pe elaborarea şi testarea unui produs soft.

Prin problematica pe care o tratează îndrumarul de laborator poate fi utilizat şi de alte specializări care studiază structura şi modul de funcţionare a sistemelor hardware ale PC (**525.1 *„Microelectronică şi Nanotehnologii”*, 526.2 *„Tehnologii Informaţionale”*, 526.3 *„Automatica şi Informatica”*** şi **526.4 *„Ingineria Sistemelor Biomedicale”***).

Elaborare:	conf. dr. Victor Ababii conf. dr. Viorica Sudacevschi
Redactor responsabil:	prof. dr.hab Emilian Guţuleac
Recenzent:	conf. dr. Sergiu Zaporojan
Redactor:	
Culegere computerizată:	conf. dr. Victor Ababii conf. dr. Viorica Sudacevschi

ECHIPAMENTE PERIFERICE

Îndrumar de laborator

Autori: Victor Ababii
Viorica Sudacevschi

Bun de tipar 01.12.2016 | Formatul hârtiei 60*84 1/16
Hârtie ofset. Tipar ofset. | Coli de tipar 8.5
Tiraj 100 ex. | Comanda nr. 1

U.T.M., 2004, Chişinău, bd. Ştefan cel Mare, 168
Secţia de Redactare, Editare şi Multiplicare a U. T. M.
Chişinău, str. Studenţilor, 11

INTRODUCERE

Scopul cursului ***Pchipamente periferice*** (EP) constă în acumularea de către studenţi a cunoştinţelor legate de componenţa, caracteristicile tehnice şi principiile de funcţionare ale componentelor hardware ale calculatoarelor personale (PC) şi metodele de programare ale acestora.

Ca rezultat al studiului cursului studentul v-a obţine cunoştinţe teoretice şi practice în:

- clasificarea şi destinaţia EP;
- terminologia şi caracteristicile tehnico-economice ale EP;
- principiile de conectare şi comunicare ale EP cu procesorul şi alte EP ale sistemelor complexe de calcul;
- componenţa structurală a principalelor tipuri de PC şi EP;
- bazele fizice şi principiile de funcţionare ale EP;
- tendinţele de dezvoltare ale EP.

În rezultatul studierii cursului EP studentul este dator să poată:

- elabora structuri ale EP, determinând particularităţile de conexiune ale acestora în sistem;
- elabora scheme logice şi funcţionale a blocurilor electronice ale EP (bloc de racordare, convertoare de semnale, dirijare, control);
- efectua calcule ale parametrilor EP;
- elabora scheme bloc şi programe de comandă cu EP;
- elaborarea produselor program pentru gestiunea operaţiilor de intrare/ieşire.

Cursul se bazează pe cunoştinţele acumulate la studierea următoarelor discipline: “Matematica superioară“, „Mecanica teoretică”, ”Fizica”, ”Grafica pe calculator”, ”Electronica”, ”Circuite integrate digitale”, ”ASDN”, ”Circuite analoge si de conversie”, „Programarea calculatorelor”.

Cunoştinţele acumulate la studierea cursului EP vor fi utilizate de către studenţi la studierea următoarelor discipline: „Sisteme cu microprocesoare”, „Arhitecturi avansate”, „Bazele transmiterii de date”, „Reţele de calculatoare”, etc.

I. NOŢIUNI TEORETICE GENERALE

I.1 Structura sistemului de calcul (PC)

Un sistem de calcul este un ansamblu de resurse hardware şi software care permit execuţia eficientă a unor programe aplicative. Acest ansamblu este stratificat pe mai multe nivele ierarhice de abstractizare (Tabelul 1.1). Pentru soluţionarea unei anumite probleme, un utilizator poate accesa resursele sistemului la diferite nivele. Nivelele inferioare oferă o mai mare libertate şi flexibilitate, însă sunt necesare cunoştinţe detaliate referitoare la construcţia internă a calculatorului; programarea pe nivelele inferioare este dificilă, ineficientă şi necesită experienţă. Pe nivelele superioare detaliile constructive sunt ascunse, activitatea de programare este mult mai eficientă, însă programul rezultat este mai lent şi programatorul are posibilităţi limitate de acces direct la resurse.

Tabelul 1.1 Niveluri de abstractizare a sistemelor de calcul.

Nivel de abstractizare	Mod de acces	Concepte de modelare a resurselor	Tip utilizator
Programe aplicative	Comenzi interactive	Specifice aplicaţiei	Ne-specialist în tehnica de calcul
Limbaje de nivel înalt	Funcţii de intrare/ieşire (biblioteci)	Fişier, structuri de date, obiecte	Programator de aplicaţii
Sistem de operare (nivelul de sistem)	Comenzi de operare sau apeluri de sistem	Fişier, canale de intrare/ieşire	Administrator de sistem
Nucleul sistemului de operare	Apel drivere de nivel inferior şi variabile de sistem	Interfeţe de intrare/ieşire, întreruperi	Programator de sistem
Maşina fizica	Instrucţiuni de intrare/ieşire în limbaj maşină	Registre (porturi) de intrare/ieşire, locaţii de memorie	Proiectant de sisteme dedicate

I.2 Programarea EP

O dată cu apariţia sistemelor de operare moderne au fost restricţionate o mare parte din metodele clasice de accesare şi de lucru cu EP. Metodele clasice, prezente în sistemele de operare MS-DOS, Windows 95/98/ME, oferă un avantaj foarte mare pentru programatorii de sistem asigurându-le un acces deplin la toate resursele PC. Odată cu apariţia sistemelor de operare bazate pe tehnologia NT unele funcţii au fost restricţionate oferind noi metode de gestiune a EP.

Majoritatea limbajelor de programare de nivel înalt prevăd prezenţa a diferitor funcţii sau proceduri de gestiune a EP, în acelaşi timp, oferă posibilitatea de a include segmente de cod Assembler care simplifică esenţial operaţiile de gestiune a EP. În procesul de elaborare a lucrărilor de laborator v-om utiliza limbajul C/C++ în combinare cu limbajul Assembler [1, 2].

Gestiunea EP sub sistemul de operare MS-DOS

Limbajul C oferă posibilitatea programării mixte atât în limbajul C cât şi în limbajul de asamblare, astfel beneficiind de structurile de date ale limbajului de nivel înalt cât şi de viteza de execuţie a programelor scrise în limbajul de asamblare.

Accesul fizic la registrele microprocesorului se poate face folosind variabilele puse la dispoziţie de mediul C:

_AX, _AH, _AL, _BX, _BH, _BL, _CX, _CH, _CL, _DX, _DH, _DL, _SI, _DI, _SP, _CS, _DS, _ES, _SS, _FLAGS.

Exemplu:

```
_AX = 0x4520;
_CX = 0x7693;
_AH = 83;
```

O altă metodă de accesare a regiştrilor microprocesorului se face prin folosirea prefixului asm.

Exemplu:

```
asm mov ax, 4356h;
asm mov dx, ax;
asm mov es, ax;
```

Pentru includerea a mai multor instrucţiuni succesive în limbajul de asamblare se pot folosi acoladele.

Exemplu:

```
asm {
```

```
xor eax, eax;
mov dx, 378h;
mov al, 25;
out dx, al;
inc dx;
in ah, dx;
}
```

Programele care conţin linii în limbajul de asamblare asm trebuie să anunţe acest lucru compilatorului prin directiva:

```
#pragma inline
```

Setul de instrucţiuni accesibile pentru elaborarea programelor de gestiune a EP în limbajul C/C++ se prezintă în Anexa 1 şi respectiv pentru limbajul de asamblare se prezintă în Anexa 2.

I.3 Accesul la maşina fizică

Un calculator – maşină fizică - dispune de următoarele resurse accesibile utilizatorului: registre interne (de date, adrese sau cu funcţii speciale), indicatoare de condiţie, locaţii de memorie şi porturi (registre) de intrare/ieşire. O resursă specială, se poate considera sistemul de întreruperi. Programatorul are acces la aceste resurse prin intermediul instrucţiunilor în cod maşină. Orice calculator dispune de un limbaj cod maşină unic ce permite transferul de informaţii între diferitele tipuri de resurse, controlul secvenţei de execuţie a programului şi efectuarea unor operaţii aritmetico-logice cu datele conţinute în resurse. Accesul la resurse se face prin diferite moduri de adresare (directă, indexată, bazată, etc.), însă datele sunt de obicei ne-structurate (bit, octet, cuvânt, dublu-cuvânt).

Pentru procesoarele care facilitează programarea multitasking pot să existe mecanisme prin care se îngrădeşte accesul programelor utilizator la anumite resurse ale sistemului. La aceste procesoare există mai multe regimuri de lucru şi un sistem de acordare a drepturilor de acces. Regimul protejat sau supervizor este utilizat de către sistemul de operare pentru partajarea resurselor între taskurile concurente. Programele utilizator au acces numai la resursele alocate de către sistemul de operare.

Accesul la interfeţele de intrare/ieşire se face printr-un set relativ redus de instrucţiuni de citire (IN) şi scriere (OUT). Programatorul trebuie să cunoască modul de funcţionare a interfeţei, adresa de bază şi

adresa relativă a registrelor interfeţei. De asemenea trebuie să cunoască semnificaţia biţilor conţinuţi în registrele de comandă şi stare. Orice modificare intervenită în structura interfeţei presupune de obicei re-scrierea rutinei de intrare/ieşire.

Accesul la locaţiile de memorie este mult mai flexibil; cele mai multe tipuri de instrucţiuni acceptă ca operand o locaţie de memorie. În cadrul arhitecturii Intel o locaţie se adresează prin specificarea (implicită sau explicită) a unui registru segment şi a unei adrese relative faţă de începutul segmentului. Funcţie de regimul de lucru, conţinutul registrului segment este interpretat ca adresa de început a segmentului (modul real) sau ca un descriptor din care se poate determina (pe baza unor tabele) adresa de început a segmentului. Adresarea se poate face la nivel de octet, cuvânt (doi octeţi) sau dublu-cuvânt (patru octeţi).

I.3.1 Accesul la resurse prin driverele sistemului de întreruperi

Majoritatea sistemelor de operare dispun de un set de rutine (drivere) prin care facilitează accesul utilizatorului la resursele calculatorului. Rolul acestor rutine este de a oferi o cale eficientă şi uniformă (standardizată) de acces, prin care detaliile constructive ale resurselor sunt ascunse. Aceste rutine au o specificaţie de interfaţă (parametri de intrare şi ieşire) care nu se modifică nici în cazul modificării resurselor pe care le deservesc.

În cazul sistemului de operare MS-DOS accesul la driverele resurselor de sistem se realizează prin apeluri de întreruperi software (instrucţiunea INT n). Fiecărui tip de resursă îi este alocat câte un nivel de întrerupere. În tabelul 1.2 sunt indicate principalele întreruperi software utilizate pentru accesul la resurse. Adresa rutinei de tratare a întreruperii este înscrisă în tabela de întreruperi a sistemului, pe poziţia corespunzătoare nivelului de întrerupere utilizat; adresa fizica se calculează după formula: *0000 :<nivel intrerupere>*4*.

Tabelul 1.2 Principalele întreruperi software.

Nivel întrerupere	Funcţia
INT 10h	Servicii pentru interfaţa video
INT 11h	Informaţii despre echipamentele conectate în sistem
INT 12h	Informaţii despre memorie
INT 13h	Servicii pentru interfaţa de disc (FDD şi HDD)
INT 14h	Servicii pentru canalul serial

INT 15h	Servicii sistem
INT 16h	Servicii de tastatură
INT 17h	Servicii pentru imprimantă
INT 18h	ROM Basic
INT 19h	Încărcare sistem de operare (Bootstrap)
INT 1Ah	Servicii de timp
INT 1Bh	Handler pentru CTRL-Breack
INT 1Ch	Întrerupere de ceas (folosita pentru lansarea unor rutine utilizator la fiecare întrerupere de ceas)

Rutinele de întrerupere software se regăsesc în porţiunea BIOS (Basic Input Output System) a sistemului de operare, rezidentă în memoria EPROM a sistemului. Pentru anumite aplicaţii speciale, utilizatorul poate să înlocuiască driverele de sistem cu driverele proprii prin înscrierea în tabela de întreruperi a adresei noii rutine de tratare a întreruperii. Se recomanda ca la terminarea aplicaţiei să se refacă adresa rutinei iniţiale. Aceasta este şi modalitatea prin care producătorii de interfeţe evoluate pot să înlocuiască driverele originale cu drivere adaptate noilor placi.

I.3.2 Accesul prin apeluri de sistem

Nivelul apelurilor de sistem oferă o interfaţă mai evoluată de acces la resursele sistemului. Se utilizează entităţi logice pentru modelarea resurselor. În acest fel se face abstracţie de detaliile constructive şi funcţionale ale interfeţelor adresate. De exemplu la o interfaţă de disc transferul de date se face la nivel de bloc de date, fără să se ţină cont de organizarea fizică a datelor pe sectoare, piste şi discuri. Adresarea se face prin nume de fişier sau prin handler (handler = structură de date utilizată pentru identificarea ţi lucrul cu un fişier sau canal de comunicaţie). În schimb datele nu pot fi citite sau scrise la nivel de sector fizic.

În sistemul MS-DOS apelurile de sistem sunt implementate prin întreruperea software INT 21h. Fiecare tip de funcţie are un identificator (un număr care la apel trebuie înscris în registrul AH) şi un număr de parametri de intrare. Aceşti parametri se înscriu în registrele interne ale procesorului înaintea instrucţiunii INT 21h.

Funcţiile de sistem acoperă majoritatea resurselor hardware şi software ale sistemului. Pentru aceeaşi interfaţă pot exista mai multe tipuri de funcţii, cu diferite grade de abstractizare. Numărul acestor funcţii a crescut continuu, la fiecare noua versiune a sistemului MS-DOS. În tabelul 1.3 se prezintă câteva exemple de funcţii de sistem.

Tabelul 1.3 Funcţii de sistem MS-DOS.

Cod funcţie (AH)	**Descrierea funcţiei**	**Parametri de intrare şi ieşire**
00h	Terminarea programului	Intrare: CS - adresa de segment a PSP-ului programului de terminat
01h	Citirea cu ecou de la tastatură	Ieşire: AL - caracterul citit
02h	Afişarea unui caracter	Intrare: DL- caracterul de afişat
03h	Intrare auxiliară (implicit COM1)	Ieşire: AL - caracterul recepţionat
04h	Ieşire auxiliară (Implicit COM1)	Intrare: DL – caracterul de transmis
05h	Tipărirea unui caracter	Intrare: DL – caracterul de transmis
08h	Citirea fără ecou de la tastatură	Ieşire: AL - caracterul citit
0fh	Deschiderea unui fişier	Intrare: DS:DX – pointerul la FCB-ul fişierului ne-deschis; Ieşire: AL=0 – s-a găsit fişierul, AL=FF – eroare
10h	Închiderea unui fişier	idem
14h	Citirea secvenţială a unui fişier folosind FCB (File Control Block)	Intrare: DS:DX – pointerul la FCB-ul fişierului deschis; Ieşire: AL=0 – citire corecta, AL#0 – eroare
3ch	Crearea unui fişier	Intrare: DS:DX- pointerul la numele fişierului (şir ASCII), CX- atributul fişierului (00- normal; 01- numai citire; 02-

		ascuns; 03- sistem); Ieşire: CF=0- operaţie reuşită, AX- identificatorul logic al fişierului, CF=1 Eroare
3dh	Deschiderea unui fişier folosind identificatorul logic	Intrare: DS:DX- pointerul la numele fişierului (şir ASCII), AL- codul de acces; Ieşire : CF=0- operaţie reuşită, AX- identificatorul logic al fişierului, CF=1 - eroare
3eh	Închiderea unui fişier folosind identificatorul logic	Intrare : BX – identificatorul logic al fişierului; Ieşire : CF=0- operaţie reuşită, CF=1 - eroare
3fh	Citirea unui fişier folosind identificatorul logic	Intrare : BX – identificatorul logic al fişierului, CX- nr. de octeţi citiţi, DS :DX- pointerul la zona de citire; Ieşire : CF=0- operaţie reuşită, AX- numărul de caractere citite, CF=1 - eroare

I.3.3 Accesul la resurse prin funcţii şi proceduri de intrare/ieşire conţinute în limbaje de programare de nivel înalt

Limbajele de nivel înalt permit accesul la resursele sistemului prin intermediul unor instrucţiuni de intrare/ieşire sau funcţii şi proceduri standard de intrare/ieşire. În limbajele orientate obiect există biblioteci de clase standard care modelează interfaţa cu diferitele echipamente periferice. De obicei se lucrează cu entităţi logice de tip fişier, soclu, canal de comunicaţie, fereastră, etc. Numărul şi tipul de operaţii permise este dictat de specificul limbajului de programare [1, 2, 3].

II. LUCRAREA DE LABORATOR NR. 1.
„*Studierea şi programarea tastaturii*”

II.1 Scopul lucrării

Studierea şi programarea tastaturii. Se urmăreşte de a studia structura şi modul de realizare al tastaturi IBM PC-AT, modul de funcţionare a acesteia şi a interfeţei cu calculatorul. Studierea funcţiilor standard BIOS şi DOS, destinate pentru gestiunea tastaturii şi metodele de utilizare a acestora în tehnicile de programare.

II.2 Resurse hardware şi software necesare pentru efectuarea lucrării

1. Calculator PC;
2. Tastaturi de diferite modele;
3. Osciloscop;
4. Compilator C / C++;
5. Compilator Assembler;
6. Turbo Help DOS/BIOS 3 / 6.

II.3 Consideraţii teoretice

II.3.1 Tastatura IBM PC-AT

Tastatura este realizată in variantele cu 84 taste, 101 sau 102 taste. Tastatura calculatoarelor IBM PC-XT, având 84 taste, nu este compatibila cu cea a calculatoarelor IBM PC-AT, dar unele tastaturi pot fi comutate in modul XT sau AT. Tastatura de tip U.S.A. (*US English*) are 101 taste, iar cea de tip *UK English* si cele pentru limbile franceza, germana, italiana, spaniola au in general 102 taste.

Între tastatură si calculator datele sunt transmise pe o linie serială bidirecţională. Calculatorul poate transmite comenzi către tastatură, iar aceasta transmite codurile de scanare ale tastelor apăsate. Interfaţa cu tastatura este realizată cu ajutorul unui microprocesor I8042. Dacă interfaţa cu tastatura nu este liberă, codurile tastelor apăsate sunt memorate intr-un buffer FIFO de 16 caractere. Dacă bufferul s-a umplut, se memorează intr-un octet rezervat in acest scop un cod de depăşire al bufferului (00h), codurile altor taste apăsate fiind pierdute pana la golirea bufferului.

Fiecare tastă are asociat un număr numit cod de scanare (*scan code*), care indica poziţia tastei in cadrul tastaturii. Toate tastele funcţionează in

modul *"make/break*". La apăsarea unei taste, se generează codul de apăsare al tastei (codul "*make*"), care este codul de scanare, cu bitul 7 egal cu 0. La eliberarea tastei, se generează codul de eliberare al tastei (codul "*break*"). La calculatoarele XT acest cod este cel de apăsare cu diferenţa că bitul 7 este setat, iar la calculatoarele AT codul este format din doi octeţi, primul cu valoarea F0h, urmat de codul de apăsare.

Tastele dispun de o funcţie de repetare automată (*typematic*). La apăsarea unei taste un timp mai îndelungat (de exemplu, peste 0.5 s), se generează codul de apăsare cu o anumita frecvenţa (2 - 30 ori pe secunda), pană la eliberarea tastei.

La cuplarea tensiunii, logica tastaturii generează un semnal de resetare (*POR - Power-On Reset*), cu durata minima de 300 ms. Tastatura efectuează apoi un test numit *BAT* (*Basic Assurance Test*),calculând o sumă de control pentru conţinutul memoriei ROM, si verificând memoria RAM a tastaturii. Indicatoarele LED pentru starea tastelor *Num Lock*, *Caps Lock* si *Scroll Lock* se aprind, iar apoi se sting. Durata de execuţie a testului este de 600-900 ms. După efectuarea testului, daca interfaţa este libera, tastatura transmite un cod de stare care indica rezultatul testului: 55h dacă testul este corect, sau o alta valoare daca a apărut o eroare.

Legătura cu calculatorul se realizează printr-un cablu prevăzut cu o mufă DIN cu 5 contacte, care se cuplează la un conector de pe placa de baza. Acest cablu conţine următoarele linii:

1.	+KBD CLOCK	Semnal de ceas, pentru sincronizarea transferului de date
2.	+KBD DATA	Datele în cod secvenţial
3.	NC	Liber
4.	GND	Comun
5.	+5 V	Tensiunea de alimentare a tastaturii de +5 V

Schema logică a tastaturii conţine un microprocesor, de exemplu HD6805V1 (se pot utiliza si alte procesoare, ca I8049 sau Z80), care are sarcina de a urmări si de a raporta in permanenţă unităţii centrale activitatea la nivelul tastelor. Microprocesorul HD6805 conţine o memorie ROM, o memorie RAM si 4 porturi de I/E de 8 biţi (PA0 .. PA7, .. PD0 .. PD7). Semnalul de ceas este generat de procesor cu ajutorul unui oscilator cu cuarţ de 4 MHz. Semnalul +KBD CLOCK este

generat de microprocesorul 6805 (ieşirea PA0), dar calculatorul poate solicita o întrerupere prin aducerea acestei linii la 0 logic, de aceea microprocesorul trebuie să testeze periodic starea acestei linii (intrarea PA1).

Semnalul +KBD DATA este transmis pe ieşirea PA2, şi este recepţionat pe intrarea INT . Dacă tastatura nu transmite date spre calculator (+KBD CLOCK = 1 şi +KBD DATA = 1), calculatorul poate semnaliza prin aducerea liniei de date la 0 logic faptul că urmează să transmită o comandă spre tastatură.

Liniile PA3, PA4 si PA5 ale portului A sunt definite ca ieşiri, comandând indicatoarele *NumLock, ScrollLock* si *CapsLock* ale tastaturii.

Conectarea la matricea tastaturii se realizează prin porturile B, C şi D. Cele 16 coloane ale matricei sunt legate la porturile B si C, la ieşirile acestor porturi transmiţându-se impulsuri. Cele 8 linii ale matricei sunt conectate la portul de intrare D. La apăsarea unei taste, aceasta efectuează o conexiune între linia si coloană corespunzătoare. Pe baza informaţiilor transmise la porturile B şi C şi a semnalelor recepţionate la portul D, microprocesorul poate identifica tasta apăsată.

II.3.2 Modul de comunicare PC – Tastatura

Comunicaţia cu calculatorul se realizează prin liniile +KBD CLOCK şi +KBD DATA. Liniile sunt comandate prin inversoare cu colector in gol. Atunci când comunicaţia este întreruptă, ambele linii sunt in starea 1 logic. Datele sunt transmise serial, pe 11 biţi. Semnificaţia acestora este următoarea:

Bit 1	bit de START (0)
Bit 2	D0 (bit c.m.p.s.)
Bit 3	D1
Bit 4	D2
Bit 5	D3
Bit 6	D4
Bit 7	D5
Bit 8	D6
Bit 9	D7 (bit c.m.s.)
Bit 10	bit de paritate impară
Bit 11	bit de STOP

Transmisia datelor de la tastatură. Dacă tastatura este pregătită pentru transmisia unei date, verifică mai întâi dacă tastatura nu este invalidată (+KBD CLOCK = 0), si daca sistemul nu doreşte transmiterea unei date (+KBD CLOCK = 1, +KBD DATA = 0). In aceste cazuri caracterul pregătit pentru transmisie este depus în buffer, si în ultimul caz se preia data. Dacă liniile de ceas şi de date sunt ambele in starea 1 logic, tastatura transmite cei 11 biţi în ordinea indicata. Datele sunt valide pentru valoarea 0 a semnalului de ceas. In cursul transmisiei tastatura verifică la intervale de 60 ms daca semnalul de ceas are valoarea 1. Dacă sistemul aduce semnalul de ceas la valoarea 0 după începerea transmisiei, dar înainte de transmisia bitului de paritate, tastatura opreşte transmisia datelor.

Transmisia datelor de la calculator. Atunci când sistemul doreşte transmisia unei date la tastatură, verifică dacă nu există o transmisie in curs de la tastatură. Dacă s-a început o transmisie, dar nu s-a transmis încă bitul 10, sistemul poate întrerupe transmisia prin aducerea semnalului de ceas la valoarea 0. Dacă acest bit a fost transmis, sistemul trebuie sa preia data. Dacă nu are loc o transmisie, sistemul aduce semnalul de ceas la 0 logic pentru o perioadă mai mare de 60 μs. Tastatura verifică dacă această perioadă este mai mare de 60 μs, si in caz afirmativ recepţionează data transmisă.

Pentru fiecare dată transmisă de sistem, tastatura trebuie sa răspundă printr-un cod de achitare in timp de 20 ms. Dacă achitarea este eronată sau lipseşte, sistemul retransmite data.

II.3.3 Interfaţa cu Tastatura

Interfaţa este realizată cu ajutorul unui microprocesor I8042, care poate comunica serial cu tastatura. Interfaţa verifică paritatea datelor recepţionate de la tastatură, interpretează şi transformă codurile de scanare, iar apoi le depune în bufferul de ieşire, de unde sistemul le poate prelua. Microprocesorul I8042 poate genera o cerere de întrerupere dacă a fost depusa o dată in bufferul de ieşire. Prin intermediul interfeţei se pot transmite si date către tastatură. În acest caz programul verifica prin citirea registrului de stare daca bufferul de intrare poate primi date. Daca aceasta condiţie este îndeplinita, înscrie data în bufferul de intrare. Datele transmise din buffer către tastatură se completează automat cu un bit de paritate impară. Este necesară confirmarea datelor şi în cazul

transmisiei datelor de la interfaţă la tastatură. Programul nu poate transmite un nou octet pană când nu primeşte confirmarea recepţiei octetului precedent de către tastatură. Linia de întrerupere "*Output Buffer Full*" (IRQ1) se poate utiliza atât pentru rutinele de transmisie, cat şi pentru cele de recepţie.

Schimbul de date dintre procesor şi tastatură are loc prin adresele 60h sau 64h. Daca se utilizează adresa 60h, informaţiile înscrise sunt interpretate ca date, iar daca se utilizează adresa 64h, ele sunt interpretate ca şi comenzi.

Operaţia de scriere la adresa 64h setează bitul 3 de stare, iar cea de scriere la adresa 60h resetează acest bit. I8042 utilizează acest bit pentru a determina daca bufferul de intrare conţine o dată sau o comanda.

În continuare sunt prezentate principalele comenzi care pot fi transmise la adresa 64h:

- **20**h - *Citirea octetului de comandă* - Controlerul tastaturii depune octetul actual de comandă in bufferul de ieşire;
- **AA**h - *Autotest* - Controlerul tastaturii executa un program de diagnoză. In caz de succes, se depune valoarea 55h in bufferul de ieşire;
- **AB**h - *Test de interfaţă* - Controlerul tastaturii testează liniile KBD CLOCK si KBD DATA. Rezultatul testului se depune in bufferul de ieşire, si are semnificaţia următoare:

00 - Test terminat cu succes.
01 - Linia KBD CLOCK este în permanenţă la nivelul 0 TTL.
02 - Linia KBD CLOCK este în permanenţă la nivelul 1 TTL.
03 - Linia KBD DATA este în permanenţă la nivelul 0 TTL.
04 - Linia KBD DATA este în permanenţă la nivelul 1 TTL.

- **AD**h - *Invalidarea tastaturii* - Linia KBD CLOCK ajunge la nivelul 0 TTL, funcţionarea interfeţei fiind invalidată;
- **AE**h - *Validarea tastaturii* - Interfaţa devine operaţională;
- **C0**h - *Citirea portului de intrare* - Controlerul citeşte conţinutul portului de intrare si îl depune in bufferul de ieşire. Comanda poate fi transmisă numai daca bufferul de ieşire este gol.
- **D0**h - *Citirea portului de ieşire* - Controlerul citeşte conţinutul portului de ieşire şi îl depune in bufferul de ieşire. Comanda poate fi transmisa numai daca bufferul de ieşire este gol.

- **D1**h - *Înscrierea portului de ieşire* - Octetul următor transmis la portul cu adresa 60h va fi înscris in portul de ieşire.
- **E0**h - *Citirea intrărilor de test* - Se citesc semnalele de pe intrările T0 si T1 şi se depun in bufferul de ieşire. Pe aceste intrări se transmit sistemului semnalele KBD CLOCK, respectiv KBD DATA. Intrării T0 îi corespunde bitul 0, iar intrării T1 ii corespunde bitul 1.

II.3.4 Comenzi pentru Tastatură

Comenzile următoare pot fi transmise de la sistem la tastatură, care trebuie să le execute intr-un timp de 20 ms generând codul ACK (**FA**h).

- **RESET** (*Iniţializare*, **FF**h)

Procesorul tastaturii execută programul de iniţializare si cel de autotest, transmite octetul care indică rezultatul autotestului şi continuă urmărirea tastaturii.

- **RESEND** (*Retransmisie*, **FE**h)

Comanda se transmite tastaturii dacă a apărut o eroare la transmisia datelor de la tastatură. Aceasta va retransmite ultima dată transmisă.

- **DEFAULT DISABLE** (*Stare iniţială, invalidare*, **F5**h)

Se iniţializează starea indicatoarelor si parametrii de repetare, se goleşte bufferul, după care se invalidează urmărirea tastelor.

- **ENABLE** (*Validare*, **F4**h)

Se goleşte bufferul şi se începe urmărirea tastelor.

- **SET TYPEMATIC RATE/DELAY** (*Setare rata/întârziere de repetare*, **F3**h)

Cu aceasta comandă şi cu parametrul care urmează se poate seta rata de repetare a tastelor (RATE) şi întârzierea cu care se începe repetarea (DELAY). Rata de repetare se poate calcula cu ajutorul a două valori RATE1 si RATE2, care se transmit in cadrul parametrului:

Bit 7: 0
Biţii 6-5: DELAY (bitul 5 c.m.p.s.)
Biţii 4-3: RATE2 (bitul 3 c.m.p.s.)
Biţii 2-0: RATE1 (bitul 0 c.m.p.s.)
Întârzierea se calculează cu relaţia:
TDELAY = (DELAY + 1) * 250 [ms] ± 20%

- **ECHO** (*Ecou*, **EE**h)

Comanda utilizată pentru testare. Tastatura răspunde prin acelaşi cod, după care continua urmărirea tastelor.

- **SET/RESET MODE INDICATORS** (*Setare/resetare indicatoare de mod*, **ED**h)

Permite modificarea stării LED-urilor tastaturii. Evidenţa stării acestora este realizata de sistem. După achitarea comenzii, se poate transmite un parametru de un octet, ai căror biţi au semnificaţia următoare:

Bit 0: stare *Scroll Lock*
Bit 1: stare *Num Lock*
Bit 2: stare *Caps Lock*
Bioi 3-7:neutilizaoi

Dacă un bit este 1, indicatorul corespunzător va fi aprins.

- **SELECT ALTERNATE SCAN CODES** (*Selecţie coduri de scanare*, **F0**h)

Tastatura poate genera trei tipuri (seturi) de coduri de scanare, această comandă permiţând selecţia tipului dorit. Se transmite un parametru cu valoarea 00, 01, 02 sau 03. Valorile 01, 02 si 03 determină selecţia codurilor de scanare cu tipul 1, 2 sau 3, iar valoarea 00 are ca efect informarea sistemului asupra tipului curent.

- **READ ID** (*Citire identificator*, **F2**h)

După achitarea comenzii, tastatura va transmite sistemului un cod de identificare format din 2 octeţi.

- **SET KEY TYPE** (*Setare tip pentru o tastă*, **FB**h, **FC**h, **FD**h)

După recepţionarea comenzii, tastatura aşteaptă selecţia unei taste. Selecţia se realizează prin transmiterea codului de scanare al tastei respective, conform asignării din setul 3. Tasta selectată va fi setată în modul de lucru indicat de comandă, astfel:

FBh - Tasta va genera cod în mod repetat la apăsare îndelungată (*Typematic*);

FCh - Tasta va genera cod de scanare atât la apăsare cat şi la eliberare (*Make/Break*);

FDh - Tasta va genera cod numai la apăsare (*Make*).

II.3.5 Tipuri de taste

Orice apăsare şi eliberare a unei taste provoacă o întrerupere hardware, fiind activată rutina de tratare a întreruperii 09h. Aceasta

rutină depune pentru fiecare tasta apăsată cate 2 octeţi intr-un buffer din zona de date BIOS, la adresa 0000:041Eh. Bufferul poate conţine maxim 15 cuvinte, depăşirea capacităţii fiind semnalizată sonor. Programele care citesc tastatura citesc de fapt din acest buffer circular şi aşteaptă apăsarea unei taste numai dacă bufferul este gol.

Tastele se împart in doua categorii:

- Taste obişnuite;
- Taste de tip *'shift'*.

Fiecare tastă, indiferent de tipul acesteia, generează o întrerupere hardware (INT 09h) la apăsare şi la eliberare, dar rutina de întrerupere tratează diferit aceste taste.

1) *Tastele obişnuite* se împart în două grupe:

- Grupa tastelor care au un cod ASCII;
- Grupa tastelor funcţionale (F1..F10 si tastele numerice).

La apăsarea tastelor normale, rutina de întrerupere înscrie in bufferul circular codul de scanare şi codul ASCII (pentru tastele din grupa ASCII) sau 0 (pentru tastele funcţionale). Tastele funcţionale combinate cu *Shift, Alt* sau *Ctrl* generează coduri speciale de scanare.

2) *Tastele de tip 'shift'* se împart si ele in doua grupe:

- Grupa tastelor 'shift': *Shift Left, Shift Right, Ctrl* si *Alt;*
- Grupa tastelor comutatoare: *Caps Lock, Scroll Lock, Num Lock* şi *Ins.*

Tastele din grupa *'shift'* au efect atât timp cat sunt apăsate (întreruperile repetate pentru aceste taste sunt ignorate de către rutina de întrerupere).

Cele din grupa comutatoarelor au efect cu reţinere (deci sunt active între două apăsări succesive ale tastelor respective).

Rutina de tratare a întreruperii 09h realizează de asemenea interpretarea şi execuţia unor comenzi imediate care corespund unor combinaţii de taste speciale apăsate simultan:

Ctrl-NumLock - Comanda suspendarea execuţiei unui program pană la apăsarea unei alte taste.

Shift-PrintScreen - Se apelează întreruperea 05h, care efectuează copierea la imprimantă a conţinutului ecranului (hardcopy).

Ctrl-Break - Se goleşte bufferul tastaturii, se plasează cuvantul 0000h in buffer, se apelează întreruperea 1Bh, care determina abandonarea programului şi revenirea în sistemul de operare. Este setat bitul 7 al octetului de la adresa 0040:0071h.

Ctrl-Alt-Del - Comanda reiniţializarea sistemului, executându-se saltul la adresa F0000:FFF0h sau la adresa de destinaţie a instrucţiunii de salt de la această adresă.

SysReq (Alt-PrintScreen) - Se apelează întreruperea 15h, căreia i se transmite valoarea 8500h în registrul AX (funcţia 85h). La eliberarea tastei, in registrul AX se înscrie valoarea 8501h. La pornirea sistemului, funcţia 85h a întreruperii 15h constă numai dintr-o instrucţiune IRET; apăsarea acestei taste nu va avea un efect vizibil.

II.3.6 Zone de date BIOS pentru Tastatură

Zona de memorie intre adresele 0000:0400h şi 0000:0500h este utilizată de BIOS pentru păstrarea unor variabile interne. Unele variabile pot fi citite prin funcţii BIOS, iar altele numai prin acces la zona respectivă de memorie. În această zonă se afla următoarele informaţii utilizate de tastatură:

0000:0417h - Reprezintă primul octet de stare al tastaturii. Funcţia 02h a întreruperii 16h permite citirea acestui octet. Accesul la acest octet permite comutarea stării tastelor *Scroll Lock, Num Lock, Caps Lock* si *Insert*. Biţii 0-3 ai acestui octet nu trebuie modificaţi.

- Bit 7 = 1: stare *Insert* activă;
- Bit 6 = 1: stare *Caps Lock* activă;
- Bit 5 = 1: stare *Num Lock* activă;
- Bit 4 = 1: stare *Scroll Lock* activă;
- Bit 3 = 1: tasta *Alt* apăsată;
- Bit 2 = 1: tasta *Ctrl* apăsată;
- Bit 1 = 1: tasta *Left Shift* apăsată;
- Bit 0 = 1: tasta *Right Shift* apăsată.

0000:0418h - Reprezintă al doilea octet de stare al tastaturii.

- Bit 7 = 1: tasta *Insert* apăsată;
- Bit 6 = 1: tasta *Caps Lock* apăsată;
- Bit 5 = 1: tasta *Num Lock* apăsată;
- Bit 4 = 1: tasta *Break* apăsată;
- Bit 3 = 1: mod de suspendare activ (tasta *Pause* a fost apăsată);
- Bit 2 = 1: tasta *SysReq* apăsată;
- Bit 1 = 1: tasta *Alt* apăsată;
- Bit 0 = 1: tasta *Ctrl* apăsată.

0000:041Ah - Acest cuvânt conţine adresa următorului caracter care va fi citit din bufferul circular (pointer la inceputul bufferului).

0000:041Ch - Cuvântul de la această adresă conţine adresa ultimului caracter din bufferul circular (pointer la sfârşitul bufferului).

0000:041Eh - 0000:043Dh - Aceasta zona conţine bufferul tastaturii. Fiecare caracter necesită 2 octeţi, deci zona permite memorarea a maxim 15 caractere. Pentru o tastă cu cod ASCII, în buffer se memorează codul ASCII şi apoi codul de scanare. Pentru o tastă cu cod extins, primul octet conţine valoarea 0, iar al doilea conţine codul extins al tastei.

Pointerii de citire din buffer şi de scriere în buffer se află la adresele 0000:041Ah, respectiv 0000:041Ch. La citirea unui caracter din buffer, pointerul de citire este incrementat cu 2. Dacă se citeşte un caracter din ultima locaţie de memorie a bufferului, pointerul se resetează la începutul bufferului. La depunerea unui caracter în buffer, pointerul de scriere este incrementat cu 2. Dacă noul caracter este memorat în ultima locaţie de memorie a bufferului, acest pointer este resetat la începutul bufferului.

II.3.7 Funcţii DOS şi BIOS pentru Tastatură

În Tabelul 2.1 se prezintă unele funcţii DOS pentru citirea de la tastatură.

Tabelul 2.1 Funcţii DOS pentru tastatură.

AH	Funcţia realizată	Parametri de intrare	Parametri de ieşire
01h	Citire caracter de la tastatură cu afişare în ecou pe ecran.		AL = caracterul citit de la tastatură.
06h	Citire/scriere directă: - de la tastatură - pe ecran.	DL = 0FFh. DL = cod caracter.	AL = caracter citit. AL = 0.
07h	Citire caracter de la tastatură fără ecou.		AL = caracter citit.
08h	Citire caracter de la tastară fără ecou.		AL = caracter citit.

0Ah	Citire de la tastatură a unui şir de caractere până la *CR* si introducere in memorie.	DS:DX = adresa zonei de memorie.	
0Bh	Test stare tastatură.		AL = FFh: există tastă apăsată; AL = 00h: nu există tastă apăsată.
0Ch	Iniţializare buffer tastatură şi apelare funcţie specificată. Se aşteaptă apăsarea unei taste.	AL = nr. funcţie cerută (01h, 06h, 07h, 08h, 0Ah).	

În Tabelul 2.2 se prezintă principalele funcţii BIOS pentru tastatură, accesibile prin întreruperea INT 16h.

Tabelul 2.2 Funcţii BIOS pentru tastatură.

AH	**Funcţia realizată**	**Param. de ieşire**
00h	Citire caracter din buffer. Dacă bufferul este gol, se aşteaptă introducerea unui caracter in buffer de către rutina INT 09h	AL = cod ASCII caracter; AH = cod "scan" caracter.
01h	Test caracter disponibil	ZF* = 0: exista caracter; ZF* = 1: buffer gol; AX = ca şi la funcţia 00h.
02h	Citire stare taste *'shift'*	AL = octet de stare al tastelor *'shift'* (primul octet de stare al tastaturii).

* registrul de fanioane bitul ZF.

II.4 Desfăşurarea lucrării

1. Să se determine resursele rezervate de sistem pentru gestiunea tastaturii.

2. Să se descrie câteva metode de identificare prin program a unei taste apăsate dintr-o matrice de taste.

3. Să se descrie modul de funcţionare al interfeţei şi comunicare cu tastatura.

4. Să se scrie un program care permite citirea datelor din portul tastaturii şi afişarea acestuia pe ecran în formatul „Scan cod", „ASCII cod" şi „Cod grafic".

5. Să se scrie un program care permite afişarea informaţiei pe ecran utilizând întreruperea BIOS 16H. Informaţia se va afişa indicând octetul de stare a tastelor şi „Scan cod", „ASCII cod" şi „Cod grafic".

6. Să se verifice funcţionarea comenzilor care pot fi transmise interfeţei, ca executarea unui autotest şi a testului de interfaţă, citirea porturilor interfeţei şi resetarea prin program a sistemului.

7. Să se scrie un program pentru aprinderea şi stingerea succesivă a indicatoarelor ale tastaturii. Operaţia se va efectua în mod repetat pană la apăsarea unei taste.

8. Să se scrie un program pentru aprinderea unor indicatoare ale tastaturii, completând şi în zona de date a BIOS noua stare a indicatoarelor.

9. Să se scrie un program care permite introducerea unor linii de la tastatura, afişând intr-o zona a ecranului starea tastelor *Caps Lock* şi *Insert*. Starea afişată pe ecran va fi actualizată la fiecare tastare.

II.5 Conţinutul dării de seamă

1. Sarcina şi scopul lucrării;
2. Schema de structură a tastaturii examinate;
3. Schema bloc a algoritmului de funcţionare a tastaturii examinate;
4. Descrieţi modul de funcţionare a tastaturii;
5. Analiza comparativă a tastaturilor moderne. Avantajele şi dezavantajele acestora;
6. Formatul blocului de date de comunicare a tastaturii cu calculatorul;
7. Schema bloc a programului de achiziţie a datelor de la tastatură;
8. Programul de achiziţie a datelor de la tastatură şi afişare pe ecran;
9. Concluzii.

III. LUCRAREA DE LABORATOR NR. 2.
„*Studierea şi programarea imprimantelor*”

III.1 Scopul lucrării

Studierea şi programarea imprimantei. Se urmăreşte de a studia structura şi tehnologiile de realizare al imprimantelor, modul de funcţionare a acesteia şi a interfeţei de comunicare cu calculatorul. Studierea funcţiilor standard BIOS şi DOS, destinate pentru gestiunea imprimantei şi metodele de utilizare a acestora în tehnicile de programare.

III.2 Resurse hardware şi software necesare pentru efectuarea lucrării

1. Calculator PC;
2. Imprimante de diferite modele;
3. Osciloscop sau analizor logic;
4. Compilator C / C++;
5. Compilator Assembler;
6. Turbo Help DOS/BIOS 3 / 6.

Lucrarea este dedicată studierii portului paralel standard şi extins al calculatoarelor compatibile IBM PC şi urmăreşte familiarizarea cu folosirea instrucţiunilor de intrare/ieşire pentru citirea unor stări şi pentru generarea unor semnale de comandă. Sunt prezentate registrele şi semnalele portului paralel, porturile bidirecţionale şi posibilităţile de utilizare ale portului paralel pentru transferul datelor intre doua calculatoare.

III.3 Consideraţii teoretice

III.3.1 Portul paralel al calculatoarelor IBM PC

Începând cu calculatorul IBM PC original (introdus în anul 1981), IBM a definit un port paralel standard pentru imprimantă. Acest port reprezenta o alternativă la portul serial mai lent, fiind utilizat mai ales pentru cuplarea imprimantelor mai rapide. Definiţia iniţială s-a regăşit în adaptorul pentru imprimanta (“*IBM Printer Adaptor*”) şi adaptorul pentru monitor monocrom şi imprimantă (“*IBM Monochrome Display and Printer Adaptor - MDPA*”).

Pe măsura creşterii numărului de echipamente care trebuie conectate

la calculator, portul paralel a devenit mijlocul prin care se pot conecta periferice cu performanţe mai ridicate. Printre acestea se număra unităţi de partajare a imprimantelor, unităţi portabile de disc, unităţi încasetate de bandă, adaptoare de reţea şi unităţi CD-ROM. Pentru aceste periferice se utilizează porturile paralele îmbunătăţite (ECP, EPP).

Conectori. Portul paralel utilizează un conector mama DB-25S aflat pe panoul din spate al calculatorului. Pentru conectarea imprimantelor paralele care respectă standardul *Centronics* se utilizează un cablu cu un conector tata DB-25P şi un conector tata *Centronics* cu 36 de contacte. Se utilizează 17 linii de semnal şi 8 linii de masă. Liniile de semnal sunt împărţite în trei grupe:

- Control (4 linii);
- Stare (5 linii);
- Date (8 linii).

Registrele portului paralel. Semnalele din grupele de sus sunt aşignate biţilor din registrele care constituie interfaţa hardware/software la portul paralel. Adresele registrelor sunt mapate în spaţiul de I/E al procesorului. Există un număr de 3 registre cu adrese consecutive, începând cu adresa de baza a portului paralel. Adresele de bază obişnuite ale portului paralel sunt: 3BCh, 378h şi 278h. Adresa 3BCh este utilizată de adaptoarele pentru monitor monocrom şi imprimanta, iar adresele 378h şi 278h de adaptoarele pentru imprimantă. Implementările mai noi ale portului paralel, care permit funcţionarea in modurile avansate definite de standardul *IEEE 1284*, utilizează intre 8 şi 16 registre, cu adresele de bază 378h sau 278h. Uneori adresele acestora pot fi relocatabile, ca in cazul adaptoarelor de tip "*Plug and Play*". Registrele portului paralel sunt de 8 biţi, accesul direct la acestea realizându-se prin instrucţiunile IN şi OUT (sau prin funcţiile inportb(), outportb()).

Registrele SPP (*Standard Parallel Port*) sunt descrise in Tabelul 3.1.

Tabelul 3.1. Registrele SPP.

Nume Registru	Offset	R/W	Funcţie
Registru de date	0	W	8 ieşiri LS TTL
Registru de stare	1	R	5 intrări LS TTL
Registru de control	2	W	4 ieşiri TTL cu colector deschis

Observaţii:

1. *Offset* indică deplasamentul faţă de adresa de baza a portului

paralel.

2. Registrul de date poate fi citit în cazul tuturor porturilor, dar se poate utiliza pentru o operaţie de intrare numai in cazul porturilor bidirecţionale.

3. Registrul de control poate fi citit ţn scopuri de testare. Valoarea citită trebuie să fie egală cu cea înscrisa anterior in acest registru.

ROM BIOS rezervă în memoria RAM, începând de la adresa 0000:0408h, 4 cuvinte de 16 biţi pentru adresa de bază a 4 porturi paralele. La pornirea sistemului, BIOS testează existenţa porturilor paralele cu adresele de bază 3BCh, 378h şi 278h (în această ordine), şi memorează adresele de bază ale porturilor găsite în cuvinte consecutive din această tabelă. ROM BIOS memorează de asemenea numărul porturilor paralele care au fost găsite in cei doi biţi superiori ai octetului de la adresa 0000:0411h
(ca o valoare intre 0 şi 3).

Sistemul de operare DOS şi funcţiile BIOS pentru imprimantă (accesate prin întreruperea INT 17h) asignează adresele acestor porturi la dispozitivele LPTn. Prima valoare din tabelă va fi adresa de bază pentru LPT1, a doua pentru LPT2, iar a treia pentru LPT3. Nu exista o adresa standard pentru al patrulea port. Prin inversarea intrărilor în tabela BIOS, se pot schimba porturile fizice care sunt asignate dispozitivelor LPT.

Semnalele portului paralel.

Semnalele sunt prezentate în Tabelul 3.2. Funcţia se referă la semnalul aflat în starea activă.

Tabelul 3.2 Semnalele portului paralel standard.

I/O	Pin	Nume semnal	Bit Rg	Funcţie
O	1	- Strobe	C0	Activat de calculator pentru a indica prezenta datelor valide pe liniile D0..D7.
O	2	Data 0	D0	Linia c.m.p.s. de date.
O	3	Data 1	D1	…
O	4	Data 2	D2	…
O	5	Data 3	D3	…
O	6	Data 4	D4	…
O	7	Data 5	D5	…
O	8	Data 6	D6	…

O	9	Data 7	D7	Linia c.m.s. de date.
I	10	-Ack	S6	Impuls negativ care indica recepţia ultimului caracter.
I	11	Busy	S7	Semnal activat de imprimanta atunci când nu poate primi date (buffer plin, eroare).
I	12	PaperEnd	S5	Indică lipsa hârtiei.
I	13	SelectOut	S4	Indică starea Online a imprimantei.
O	14	-AutoFeed	C1	Activat de calculator pentru ca imprimanta să insereze automat caracterul LF după CR.
I	15	-Error	S3	Indică o condiţie de eroare a imprimantei.
O	16	-Init	C2	Utilizat pentru iniţializarea imprimantei.
O	17	-SelectIn	C3	Activat pentru selectarea imprimantei.
	18 – 25	Ground		Linii de masă.

Configuraţia registrelor.

Se prezintă în continuare configuraţia registrelor portului paralel.

Registrul de date (3BCh, 378h, 278h)

D7	D6	D5	D4	D3	D2	D1	D0
Data7	Data6	Data5	Data4	Data3	Data2	Data1	Data0

La scriere, datele sunt transmise (neinversate) la pinii 2-9. Valoarea înscrisă va fi memorată şi va rămâne stabilă pană la înscrierea unei noi valori.

La citire, în cazul în care bitul 5 al registrului de control (C5) este 0, se citeşte valoarea care a fost înscrisă în prealabil în registrul de date. In cazul în care bitul C5 este 1, valoarea citită reprezintă datele preluate de la pinii 2-9 (numai la porturile bidirecţionale).

Registrul de stare (3BDh, 379h, 279h)

S7	S6	S5	S4	S3	S2	S1	S0
Busy	-Ack	PaperEnd	SelectOut	-Error	X	X	X

- Biţii 0-2 nu sunt utilizaţi (nedefiniţi).
- Bitul 3, - Error (pin 15): Dacă este 0, a apărut o eroare la tipărire, imprimanta este neoperaţională sau nu are hârtie (neinversat).
- Bitul 4, SelectOut (pin 13): Dacă este 1, imprimanta este operaţională (neinversat).
- Bitul 5, PaperEnd (pin 12): Dacă este 1, imprimanta nu are hârtie (neinversat).
- Bitul 6, -Ack (pin 10): Printr-un impuls negativ al semnalului *Acknowledge*, cu durata de aproximativ 10 ms, imprimanta indică preluarea ultimului caracter (neinversat).
- Bitul 7, Busy (pin 11): Daca este 0, tipărirea este in curs sau bufferul imprimantei este plin (*inversat*).

Registrul de control (3BEh, 37Ah, 27Ah)

C7	C6	C5	C4	C3	C2	C1	C0
X	X	Dir	IRQEn	-SelectIn	-Init	-AutoFeed	-Strobe

- Bitul 0, -Strobe (pin 1): Prin setarea acestui semnal la 1 pentru cel puţin 1 μs se indică imprimantei că exista o nouă data validă pe liniile de date (*inversat*).
- Bitul 1, -AutoFeed (pin 14): Daca semnalul este setat la 1, imprimanta va trece la o linie nouă după recepţionarea fiecărui caracter CR (*inversat*).
- Bitul 2, -Init (pin 16): Prin setarea acestui semnal la 0 pentru cel puţin 50 μs, imprimanta va fi iniţializată (neinversat).
- Bitul 3, -SelectIn (pin 17): Dacă semnalul este setat la 1, codurile DC1 şi DC3 pot activa şi dezactiva imprimanta (*inversat*).
- Bitul 4, IRQEn: Dacă este setat la 1, un front crescător al semnalului -Ack va genera o cerere de întrerupere (nu exista un semnal extern asociat acestui bit).
- Bitul 5, Dir: La porturile bidirecţionale (calculatoare PS/2 şi unele calculatoare AT), acest bit indică direcţia liniilor de date: 0 - ieşire; 1 - intrare (nu exista un semnal extern asociat acestui bit).
- Biţii 6-7 nu sunt utilizaţi (ignoraţi la scriere, nedefiniţi la citire).

Pentru ieşirile C0-C3 ale registrului de control se utilizează un buffer inversor 7405 cu colector deschis, ieşirile fiind conectate la +5 V prin rezistenţe (in mod obişnuit de 4.7 K.), astfel încât un dispozitiv extern le

poate forţa in starea 0, fără suprasolicitarea circuitelor din calculator. Biţii C0-C3 ai registrului de control se pot utiliza şi ca intrări. Pentru aceasta se vor seta ieşirile corespunzătoare la 1 logic, prin înscrierea valorii binare 0100 in biţii c.m.p.s. ai registrului, după care se va putea citi starea semnalelor externe aplicate la pinii respectivi, prin citirea portului de intrare (feedback) al registrului de control. Se va ţine cont de inversarea biţilor C0, C1 şi C3.

Întreruperi.

Portul paralel poate genera o întrerupere prin setarea bitului IRQEn din registrul de control, dar transferul prin întreruperi nu este utilizat de BIOS şi DOS. Versiunile sistemului de operare OS/2 anterioare versiunii 3.0 (*Warp*) utilizează întreruperea pentru tipărire, dar începând cu următoarele versiuni întreruperea nu este necesară (deşi ea poate fi utilizată). Prin setarea bitului IRQEn este validat un buffer cu trei stări al liniei IRQ, fiind generată o întrerupere la fiecare front crescător al semnalului –Ack. Pentru generarea întreruperii trebuie ca nivelul respectiv de întrerupere sa fie validat prin resetarea bitului corespunzător din registrul măştilor de întrerupere.

Adaptorul MDPA (cu adresa de baza 3BCh) şi adaptorul primar de imprimantă (cu adresa de baza 378h) utilizează de obicei întreruperea hardware de nivel 7 (IRQ 7), corespunzătoare INT 0Fh. Ambele adaptoare utilizează aceeaşi întrerupere, şi deoarece magistrala ISA nu permite partajarea întreruperilor, numai unul din cele doua adaptoare trebuie sa fie validat la un moment dat. Nivelul 7 de întrerupere este utilizat şi de unele placi de sunet, deci trebuie evitate conflictele intre acestea şi portul paralel.

Adaptorul secundar de imprimantă (cu adresa de baza 278h) utilizează în mod normal întreruperea hardware de nivel 5 (IRQ 5), corespunzătoare INT 0Dh. Acest nivel este utilizat de multe ori de unele adaptoare EGA, placi de sunet sau placi de reţea.

Porturi bidirecţionale.

Seria de calculatoare IBM PS/2 a adăugat posibilitatea utilizării porturilor paralele în mod bidirecţional. Prin utilizarea bitului 5 al registrului de control (C5 = 1) se pot trece ieşirile portului de date în starea de înaltă impedanţă, acest port fiind deconectat de la pinii conectorului. Astfel în registrul de date (feedback) se pot citi semnalele TTL externe aplicate la pinii corespunzători. Valorile înscrise în registrul de date sunt memorate, dar nu sunt transmise la pini. Există calculatoare

AT cu porturi paralele bidirecţionale compatibile PS/2. Anumite adaptoare necesită setarea unui jumper pentru validarea modului de intrare al portului. Calculatoarele cu portul paralel integrat pe placa de bază pot pune la dispoziţie o setare BIOS pentru validarea sau invalidarea funcţionării bidirecţionale a portului.

Portul paralel IBM original dispunea de circuitele necesare pentru a fi bidirecţional. Circuitul 74LS374 are un pin de validare (pinul 1: -OE) care permite trecerea ieşirilor de date in starea de înaltă impedanţa, dar este conectat de obicei la masă pentru validarea în permanenţă a driverelor de ieşire. Circuitul latch 74LS174 utilizat pentru biţii C0-C4 ai registrului de control este un latch hexa, bitul 5 al magistralei de date a procesorului fiind conectat la a şasea intrare (pinul 15) a circuitului. Acest bit este memorat la o scriere în registrul de control, dar ieşirea Q6 corespunzătoare a circuitului latch nu este utilizată. Dacă se deconectează de la masă pinul 1 (-OE) al circuitului 74LS374 şi acesta se conectează la pinul 15 (Q6) al circuitului 74LS174, se obţine un port paralel bidirecţional compatibil PS/2.

Există posibilitatea utilizării portului paralel standard (unidirecţional) ca port de intrare. Prin înscrierea valorii FFh în registrul de date, orice semnal extern TTL “low” va fi citit ca 0 în registrul de date (feedback). Un semnal extern TTL “high”, sau lipsa unui semnal, va permite ca pinul (şi bitul de date) să rămână la 1 logic. O asemenea utilizare nu este insa recomandată, deoarece poate determina forţarea circuitului 74LS374 (sau echivalent) şi deteriorarea acestuia. De notat că spre deosebire de porturile IBM iniţiale, realizate cu circuite MSI TTL, porturile paralele recente sunt implementate prin controlere LSI intr-un singur cip, cu caracteristici electrice şi costuri de înlocuire diferite. Uneori controlerul LSI integrează şi alte circuite, ca un port serial sau un adaptor pentru monitor monocrom, astfel ca la o supraîncălzire se pot distruge şi acestea. În cazul utilizării oricărui port paralel bidirecţional pentru intrare, se recomandă trecerea ieşirilor in starea de înaltă impedanţa de fiecare data când portul este conectat la dispozitive externe. Se pot utiliza şi rezistenţe pentru limitarea curentului. Anumite circuite pentru porturi paralele blochează bitul C5 de control al direcţiei, pentru a preveni schimbarea accidentală a direcţiei (deblocarea se realizează prin înscrierea unei valori intr-un alt registru).

Transferul datelor utilizând portul paralel.

Există 3 moduri principale de conexiune intre doua porturi paralele

pentru transferul datelor între două sisteme utilizând aceste porturi. Cel mai utilizat mod (*Modul 1*) este cel în care datele sunt transferate pe 4 biţi, conectând biţii D0-D4 (sau D3-D7) ai registrului de date din primul port cu biţii S3-S7 ai registrului de stare din cel de-al doilea port, şi invers. Cei 4 biţi de date (de exemplu D0-D3) transmişi de la un port sunt citiţi de al doilea port (de exemplu S3-S6), celalalt bit (D4/S7) fiind utilizat pentru sincronizare (*Data Ready* ca strob de date intr-un sens sau *Acknowledge* in celalalt sens). Acest mod de conectare funcţionează cu oricare port standard. Este necesară implementarea unui protocol pentru controlul direcţiei de transfer.

În Tabelul 3.3 se prezintă o lista de conexiuni a unui cablu care se poate utiliza pentru transferul in *Modul 1* (există şi alte variante de conexiuni pentru acest mod). Varianta prezentată poate fi utilizata şi cu diferite programe DOS pentru transferul datelor, ca *INTERLNK*, *LapLink* şi *FastLynx* (*FastWire*).

Tabelul 3.3 Lista de conexiuni pentru transferul in Modul 1.

Port 1	**Pin**	**Direcţie**	**Pin**	**Port 2**	**Conexiune**
D0+	2	→	15	S3+	directă
D1+	3	→	13	S4+	directă
D2+	4	→	12	S5+	directă
D3+	5	→	10	S6+	directă
D4+	6	→	11	S7-	inversată
S7-	11	←	6	D4+	inversată
S6+	10	←	5	D3+	directă
S5+	12	←	4	D2+	directă
S4+	13	←	3	D1+	directă
S3+	15	←	2	D0+	directă
Gnd	25	--	25	Gnd	(masa)

Modul 2 poate fi utilizat numai la porturile bidirecţionale, permiţând transferuri pe 8 biţi. Portul receptor trebuie să aducă registrul de ieşire date în starea de înaltă impedanţă (prin setarea bitului C5 la 1) pentru a utiliza registrul de date (feedback) ca registru de intrare.

În *Modul 3* se utilizează pentru intrare 4 biţi ai registrului de control şi 4 biţi ai registrului de stare. Acest mod permite transferuri bidirecţionale pe 8 biţi cu oricare port standard. Biţii registrului de control utilizaţi ca intrări trebuie setaţi la 1 logic: C0=C1=C3=0 şi C2=1.

Există şi posibilitatea utilizării pentru intrare a 5 biţi ai registrului de stare şi a 3 biţi ai registrului de control. Cealaltă ieşire a registrului de control va fi conectata cu ieşirea corespunzătoare a registrului de control din al doilea port. Daca această ieşire este implicit la 1 logic, fiecare port o poate aduce la 0 logic.

III.3.2 Prezentare generala a standardului *IEEE 1284*

La apariţia perifericelor cu performanţe mai ridicate, portul paralel standard a fost utilizat şi pentru conectarea acestor periferice. Problemele întâmpinate de proiectanţii şi utilizatorii acestor periferice se pot împărţi în trei categorii. In primul rând, deşi performanţele calculatoarelor au crescut considerabil, arhitectura şi performanţele portului paralel au rămas practic neschimbate. Rata maximă de transfer care se poate obţine cu această arhitectură este in jur de 150 KB/s, procesorul fiind solicitat in mod intens. În al doilea rând, nu exista un standard pentru interfaţa electrică, ceea ce conducea la numeroase probleme atunci când se încerca garantarea operaţiilor de transfer pentru diferite platforme. În sfârşit, lipsa standardelor de proiectare a determinat o limitare a lungimii cablurilor externe sub 2 m.

In 1991 a avut loc o întâlnire a producătorilor de imprimante cu scopul iniţierii unui nou standard pentru controlul inteligent al imprimantelor conectate intr-o reţea. Aceşti producători, printre care *IBM, Lexmark, Texas Instruments*, au format o asociaţie numită "*Network Printing Alliance*" (*NPA*). *NPA* a definit un set de parametri care, dacă sunt implementaţi de sistemul de calcul şi de imprimantă, asigura controlul complet al operaţiilor cu imprimanta. Pentru implementarea acestui standard este necesară însă o conexiune bidirecţională cu performanţe ridicate cu sistemul de calcul. Conexiunea obişnuită utilizată, portul paralel standard, nu avea performanţele necesare pentru a îndeplini cerinţele acestui standard.

NPA a propus institutului *IEEE* crearea unui comitet pentru dezvoltarea unui nou standard al unui port paralel bidirecţional de viteză ridicată pentru calculatoarele personale. Cerinţa era ca noul standard să fie în totalitate compatibil cu portul paralel original, permiţând însă creşterea ratei de transfer la peste 1 MB/s, în ambele sensuri. Acest comitet a elaborat standardul *IEEE 1284*, numit "*Standard Şignaling Method for a Bi-directional Parallel Peripheral Interface for Personal Computers*", care a fost aprobat in 1994.

Standardul *IEEE 1284* defineşte 5 moduri de transfer, care asigura crearea unui canal de conexiune în sens direct (PC la periferic), în sens invers (periferic la PC) sau bidirecţional între calculator şi periferic. În ultimul caz, deoarece exista un singur set de linii de date, conexiunea este semiduplex, datele fiind transferate intr-un singur sens la un moment dat.

Modurile de transfer definite sunt următoarele:

- Modul standard, numit şi mod "*Centronics*" sau mod de compatibilitate;
- Modul de transfer pe 4 biţi (*Nibble Mode*), utilizând liniile de stare pentru transferul datelor;
- Modul de transfer pe octet (*Byte Mode*), utilizând liniile de date;
- EPP (*Enhanced Parallel Port*), utilizat mai ales de periferice ca unităţi de bandă, CD-ROM, unităţi de disc, adaptoare de reţea;
- ECP (*Extended Capability Port*), utilizat în special de noile generaţii de imprimante şi scannere.

III.3.3 Moduri de transfer

Modul de compatibilitate.

Acest mod defineşte protocolul utilizat de portul paralel standard pentru transferul datelor la imprimantă. Datele sunt plasate pe liniile de date, se testează starea imprimantei pentru a determina daca aceasta este libera sau ocupată, şi apoi se generează prin program un strob de date.

Operaţiile efectuate sunt descrise în continuare:

1. Se înscrie un octet de date in registrul de date.
2. Se citeşte registrul de stare pentru a testa semnalul Busy.
3. Daca semnalul Busy nu este activ, se înscrie în registrul de control configuraţia corespunzătoare pentru activarea semnalului -Strobe.
4. Se înscrie in registrul de control valoarea necesara pentru dezactivarea semnalului -Strobe.

Pentru transferul unui octet, sunt necesare deci patru instrucţiuni de I/E şi cel puţin un număr egal de instrucţiuni suplimentare. Rata de transfer rezultată este suficientă pentru comunicaţia cu imprimantele matriciale şi imprimantele cu laser mai vechi, dar nu şi pentru unităţile de disc portabile, adaptoare de reţea şi noile generaţii de imprimante cu laser.

Acest mod asigură doar realizarea unui canal în sens direct şi trebuie combinat cu un mod care asigură un canal în sens invers pentru a realiza o comunicaţie bidirecţională. Includerea acestui mod in cadrul standardului s-a realizat pentru a asigura compatibilitatea cu numeroasele tipuri de imprimante şi alte periferice existente.

Unele controlere de I/E integrate implementează un mod care utilizează un buffer FIFO pentru transferul datelor cu protocolul modului de compatibilitate. Acest mod este numit "*Fast Centronics*" sau "*Mod* FIFO". Atunci când acest mod este validat, datele înscrise în portul FIFO vor fi transferate la imprimantă utilizând stroburi de date generate prin hardware. Deoarece exista o întârziere redusa intre transferuri, iar protocolul
nu este realizat prin program, cu anumite sisteme pot fi atinse rate
de transfer de peste 500 KB/s. Acest mod nu este însă unul din modurile definite de standardul *IEEE 1284*.

Modul de transfer pe 4 biti ("Nibble").

Acest mod permite obţinerea unui canal în sens invers de la un periferic, fiind combinat de obicei cu modul de compatibilitate pentru a crea un canal bidirecţional.

Toate porturile paralele standard dispun de 5 linii de la periferic la calculator, care pot fi utilizate pentru citirea stării perifericului. Prin utilizarea acestor linii un periferic poate transmite un octet de date in doua cicluri de transfer de cate 4 biţi. Deoarece linia -Ack este utilizata de obicei pentru generarea unei întreruperi, biţii utilizaţi pentru transfer nu sunt împachetaţi în mod convenabil în cadrul octetului definit de registrul de stare, fiind necesare operaţii pe biţi pentru a obţine octetul corect.

Fazele în modul de transfer pe 4 biţi sunt următoarele:

1. Calculatorul indică faptul că este pregătit pentru preluarea datelor prin trecerea semnalului HostBusy în 0.
2. Perifericul răspunde prin plasarea primilor 4 biţi pe liniile de stare.
3. Perifericul semnalează datele valide prin semnalul PtrClk = 0.
4. Calculatorul setează la 1 semnalul HostBusy pentru a indica recepţia datelor şi faptul că nu este încă pregătit pentru următorii 4 biţi.
5. Perifericul setează la 1 semnalul PtrClk ca semnal de achitare pentru calculator.
6. Stările 1-5 se repeta pentru următorii 4 biţi ai octetului.

Modul de transfer pe 4 biţi, ca şi modul de compatibilitate, necesită

implementarea protocolului prin program. Acest mod este cel mai solicitant din punct de vedere software, motiv pentru care rata de transfer este limitata la aproximativ 50 KB/s. Aceasta limitare nu are un efect vizibil în cazul perifericelor cu cerinţe reduse pentru canalul în sens invers, ca imprimantele, dar nu este acceptabilă în cazul adaptoarelor de reţea, unităţilor de disc sau unităţilor CD-ROM.

Avantajul principal al combinării modului de compatibilitate şi a modului de transfer pe 4 biţi consta în posibilitatea utilizării lor pentru orice calculator cu un port paralel standard.

Modul de transfer pe octet.

Prin implementările ulterioare ale portului paralel, anumiţi producători, ca IBM cu portul paralel al calculatoarelor PS/2, au adăugat posibilitatea de a dezactiva driverele utilizate pentru liniile de date, permiţând portului de date să devină un port de intrare. Aceasta permite transmiterea unui octet întreg de date de către un periferic într-un singur ciclu de transfer prin utilizarea celor 8 linii de date, spre deosebire de cele doua cicluri necesare in modul "Nibble".

Prin utilizarea modului de transfer pe octet pentru transferurile pe canalul invers se pot obţine rate de transfer de la periferic la calculator apropiate de cele ale modului de compatibilitate, de la calculator la periferic. Acest tip de port este denumit uneori ca "port bidirecţional îmbunătăţit", creând anumite confuzii cu portul paralel îmbunătăţit EPP (*Enhanced Parallel Port*).

Operaţiile efectuate pentru un transfer sunt următoarele:

1. Calculatorul indică faptul că este pregătit pentru preluarea datelor prin setarea la 0 a semnalului HostBusy.

2. Perifericul răspunde prin plasarea primului octet pe liniile de date.

3. Perifericul indică un octet valid prin semnalul PtrClk = 0.

4. Calculatorul setează semnalul HostBusy la 1 pentru a indica preluarea unui octet şi faptul că nu este pregătit pentru un nou octet.

5. Calculatorul generează un impuls negativ al semnalului HostClk ca un semnal de achitare pentru periferic.

6. Operaţiile 1-5 sunt repetate pentru următorii octeţi.

Modul EPP.

Protocolul pentru portul EPP a fost dezvoltat iniţial de firmele *Intel, Xircom* şi *Zenith Data Systems*, ca un mijloc de a asigura o legătură paralelă cu performanţe ridicate, fiind în acelaşi timp compatibilă cu portul paralel standard. Acest protocol a fost implementat de *Intel* in

setul de cipuri 386SL (circuitul de I/E 82360), înainte de începerea elaborării standardului *IEEE 1284*. Protocolul EPP a oferit numeroase avantaje producătorilor de periferice şi a fost adoptat ca o metodă opţională de transfer, fiind formată o asociaţie pentru promovarea acestui protocol. Asociaţia a promovat adoptarea acestui protocol ca unul din modurile avansate ale standardului *IEEE 1284*.

Protocolul EPP defineşte patru tipuri de cicluri de transfer:

1. Ciclu de scriere date.
2. Ciclu de citire date.
3. Ciclu de scriere adrese.
4. Ciclu de citire adrese.

Ciclurile de date sunt utilizate pentru transferul datelor între calculator şi periferic. Ciclurile de adrese pot fi utilizate pentru transferul adreselor, a informaţiilor de comandă şi control.

Fazele ciclului de scriere date sunt următoarele:

1. Programul execută un ciclu de I/E de scriere în registrul 4 (port de date EPP).
2. Se activează semnalul -Write şi datele se înscriu ţn portul paralel.
3. Strobul de date este activat, deoarece -Wait este activ.
4. Portul aşteaptă pentru achitarea de la periferic (-Wait dezactivat).
5. Strobul de date este dezactivat şi ciclul EPP se termină.
6. Ciclul de I/E al magistralei ISA se termina.
7. Semnalul -Wait este activat pentru a indica posibilitatea începerii următorului ciclu.

Se subliniază faptul ca întregul transfer de date are loc intr-un singur ciclu ISA. Efectul este ca prin utilizarea protocolului EPP pentru transferul datelor se pot atinge rate de transfer intre 500 KB/s şi 2 MB/s. Astfel, perifericele care utilizează portul paralel pot atinge nivele de performanţă apropiate de cele ale plăcilor de extensie ISA.

Setul de registre EPP reprezintă o extensie a registrelor portului paralel standard. Registrele EPP sunt prezentate în Tabelul 3.4.

Tabelul 3.4 Registrele EPP.

Nume port	Offset	Mod	R/W	Descriere
Port de date SPP	+0	SPP/ EPP	W	Port de date standard.
Port de stare SPP	+1	SPP/ EPP	R	Citeşte liniile de stare ale interfeţei.
Port de control	+2	SPP/	W	Setează liniile de

SPP		EPP		control.
Port de adresă EPP	+3	EPP	R/W	Generează un ciclu de citire sau scriere adrese.
Port de date EPP	+4	EPP	R/W	Generează un ciclu de citire sau scriere date.
Nedefinite	+5..+7	EPP	n/a	Utilizate în mod diferit de diferitele implementări. Pot fi utilizate pentru I/E de 16 sau 32 biţi.

Prin generarea unei singure instrucţiuni de scriere in portul de date EPP, controlerul EPP va genera semnalele de protocol pentru transferul datelor într-un ciclu de scriere date EPP. Instrucţiunile de I/E la porturile cu adrese +0..+2 faţă de adresa de bază au acelaşi efect cu cele la portul paralel standard. Astfel se garantează compatibilitatea cu perifericele portului paralel standard. Ciclurile de adrese sunt generate atunci când operaţiile de citire sau scriere se efectuează la portul cu adresa +3 faţă de adresa de baza.

Porturile +5..+7 sunt utilizate in mod diferit de diversele implementări hardware. Acestea pot fi utilizate pentru implementarea interfeţelor pe 16 sau 32 de biţi, sau ca registre de configuraţie. De exemplu, exista porturi paralele care pot fi accesate utilizând instrucţiuni de I/E pe 32 de biţi, chiar dacă au interfeţe de 8 biţi. Controlerul ISA va intercepta operaţia pe 32 de biţi şi va genera 4 cicluri rapide pe 8 biţi. Aceste cicluri suplimentare sunt generate prin hardware şi sunt transparente pentru programe. Timpul total pentru aceste cicluri va fi mai mic decât cel pentru 4 cicluri independente de 8 biţi.

Posibilitatea de a transfera datele prin utilizarea unei singure instrucţiuni permite porturilor paralele în modul EPP să transfere datele la viteza magistralei ISA. În funcţie de implementarea portului calculatorului şi de performanţele perifericului, un port EPP poate atinge o rată de transfer între 500 KB/s şi 2 MB/s. Această rată de transfer este mai mult decât suficientă pentru a permite adaptoarelor de reţea, unităţilor CD-ROM, unităţilor de bandă şi altor periferice să funcţioneze

la o viteza apropiată de cea a magistralei ISA.

Modul ECP.

Protocolul ECP a fost propus de firmele *Hewlett Packard* şi *Microsoft* ca un mod avansat pentru comunicaţia cu periferice ca imprimante şi scannere. Ca şi modul EPP, modul ECP asigură o comunicaţie bidirecţională între calculator şi periferic.

Protocolul ECP pune la dispoziţie următoarele tipuri de cicluri atât în sens direct, cât şi invers:

1. Cicluri de date.
2. Cicluri de comenzi.

Spre deosebire de modul EPP, atunci când a fost propus protocolul ECP, a fost propusă de asemenea o implementare standard a registrelor. Aceasta propunere defineşte caracteristici care sunt dependente de implementare şi care nu sunt specificate de standardul *IEEE 1284*. Aceste caracteristici cuprind compresia de date *Run Length Encoding* (RLE) pentru adaptoarele din calculator, adresarea canalelor, memorii FIFO pentru canalul direct şi cel invers, DMA şi I/E programate pentru interfaţa calculatorului.

Facilitatea RLE permite decompresia şi (opţional) compresia în timp real a datelor, cu rate de compresie de pana la 64:1 (4:1 in mod tipic). Acest lucru este util în special în cazul imprimantelor şi scannerelor care transferă imagini de dimensiuni mari, cu şiruri identice de date. Compresia este realizată prin contorizarea octeţilor identici şi transmiterea unui octet RLE care indică numărul de repetări ale următorului octet. Decompresia este realizata prin interceptarea octetului RLE şi repetarea corespunzătoare a următorului octet. Un contor RLE egal cu 0 specifică faptul că următorul octet reprezintă un singur octet de date, iar un contor egal cu 127 indică necesitatea repetării următorului octet de 128 de ori.

Adresarea canalelor este diferită conceptual de adresarea în cazul modului EPP, fiind utilizată pentru adresarea mai multor dispozitive logice ale unui singur dispozitiv fizic. De exemplu, in cadrul unui singur dispozitiv multifuncţional poate exista un fax, o imprimantă şi un modem, cu un singur port paralel. Fiecare din aceste funcţii separate sunt realizate de câte un dispozitiv logic separat. Utilizând adresarea canalelor pentru accesul la fiecare din aceste dispozitive, se pot recepţiona date de la modem în timp ce canalul de date al imprimantei este ocupat cu prelucrarea unei imagini. In cazul protocolului modului de

compatibilitate, dacă imprimanta devine ocupată, nu se mai pot realiza alte operaţii de comunicaţie până la eliberarea canalului de date al imprimantei. In cazul modului ECP, driverul software adresează un alt canal şi comunicaţia poate continua.

Fazele transferului în sens direct sunt următoarele:

1. Calculatorul plasează datele pe liniile de date şi indică un ciclu de date prin setarea HostAck la 1.

2. Calculatorul setează HostClk la 0 pentru a indica existenţa unei date valide.

3. Perifericul răspunde prin setarea PeriphAck la 1.

4. Calculatorul setează HostClk la 1. Acest front al semnalului HostClk trebuie utilizat pentru înscrierea datei la periferic.

5. Perifericul setează PeriphAck la 0, indicând faptul că este pregătit pentru următorul octet.

6. Ciclul se repeta, dar de data aceasta este un ciclu de comenzi deoarece HostAck este 0.

Modelul registrelor ECP defineşte 6 registre care necesită numai 3 porturi de I/E. Aceste registre sunt descrise in Tabelul 3.5.

Tabelul 3.5 Descrierea registrelor ECP.

Offset	Nume	R/W	Mod ECP	Funcţie
000	Data	R/W	000-001	Data Register
000	ecpAFifo	R/W	011	ECP Address FIFO
001	dsr	R0	Toate modurile	Status Register
002	dcr	R/W	Toate modurile	Control Register
400	cFifo	R/W	010	Parallel Port Data FIFO
400	ecpDFifo	R/W	011	ECP Data FIFO
400	tFifo	R/W	110	Test FIFO
400	cnfgA	R	111	Configuration Register A
401	cnfgB	R/W	111	Configuration Register B
402	ecr	R/W	Toate modurile	Extended Control Register

Daca portul ECP este în modul standard sau în modul bidirecţional, primele 3 registre operează identic cu cele ale unui port standard, fiind menţinută compatibilitatea cu dispozitivele şi driverele mai vechi.

Descrierea registrelor portului ECP. Funcţionarea registrelor depinde de câmpul de mod din registrul ecr. Pentru alte moduri decât cele specificate, operarea registrelor este nedefinita.

ecpAFifo (mod 011)

Un octet de date scris in acest registru este plasat in memoria FIFO şi este marcat ca o adresă ECP. Acest octet va fi transmis la periferic în mod automat (prin hardware). Utilizarea acestui registru este definită numai pentru sensul direct.

cFifo (mod 010)

Cuvintele înscrise în acest registru, sau transferate prin DMA de la sistem, sunt transmise prin hardware la periferic prin protocolul portului paralel standard. Dacă nu trebuie transferate cuvinte complete, operaţia trebuie efectuată în modul 000. Acest mod este definit numai pentru sensul direct.

ecpDFifo (mod 011)

Pentru sensul direct, cuvintele înscrise în acest registru, sau transferate prin DMA de la sistem, sunt transmise prin hardware la periferic prin protocolul portului ECP. Dacă nu trebuie transferate cuvinte complete, operaţia trebuie efectuată în modul 000. Pentru sensul invers, se citesc cuvintele de la periferic în memoria FIFO de date.

tFifo (mod 110)

Se pot citi, scrie sau transfera prin DMA cuvinte între sistem şi acest registru, în ambele sensuri. Datele înscrise în acest registru nu vor fi transmise pe liniile portului paralel prin utilizarea protocolului hardware. Datele vor fi transferate la rata maximă a magistralei ISA, astfel că se pot efectua măsurători de performanţă prin program.

cnfgA (mod 111)

Acest registru permite obţinerea unor informaţii specifice implementării portului paralel. Bitul 7 al acestui registru arată dacă întreruperile sunt generate prin impulsuri (bit 7 = 0) sau pe nivel (bit 7 = 1). Biţii 6-4 (impID) pot identifica implementarea portului (printr-un număr ID), şi indică dimensiunea cuvântului:

- 000: 16 biţi,
- 001: 8 biţi,
- 010: 32 biţi.

cnfgB (mod 111)

Acest registru permite selectarea prin program a întreruperilor şi a canalelor DMA. O implementare R/W a acestui registru implică un dispozitiv configurabil prin program. Toate porturile ISA trebuie să implementeze acest registru cel puţin ca un registru R/O. Bitul 7 (compress) este de tip R/O în cazul în care facilitatea de comprimare nu este implementată, la citire returnând în acest caz valoarea 0. Dacă bitul este setat la 1, datele vor fi comprimate înainte de transmiterea lor. Dacă bitul este 0, vor fi transmise numai date necomprimate. Bitul 6 (intrValue) returnează valoarea de pe linia IRQ a magistralei ISA pentru a se putea determina conflictele posibile. Biţii 5-3 (intrLine) permit selectarea întreruperii. Dacă aceşti biţi conţin valoarea 000, întreruperea trebuie selectată prin jumper. Valorile 001..111 selectează, în ordine, IRQ 7 (implicit), IRQ 9, IRQ 10, IRQ 11, IRQ 14, IRQ 15, respectiv IRQ 5. Biţii 2-0 (dmaChannel) permit selectarea canalului DMA. Dacă aceşti biţi conţin valoarea 000 sau 100, canalul DMA trebuie selectat prin jumper, fiind de 8 biţi, respectiv de 16 biţi. Valorile 001, 010, 011 selectează canalul 1, 2, respectiv 3 (8 biţi). Valorile 101, 110, 111 selectează canalul 5, 6, respectiv 7 (16 biţi).

ecr

Registrul ecr (*Extended Control Register*) controlează funcţiile extinse ale portului ECP. Biţii 7-5 indică modul de funcţionar a portului ECP. Bitul 4 (-ErrIntrEn), valid numai în modul ECP, permite prin valoarea 0 generarea întreruperilor la frontul descrescător al semnalului -PeriphRequest (-Error). Prin valoarea 1 generarea întreruperilor este invalidată. Bitul 3 (dmaEn) validează prin valoarea 1 transferurile DMA. Un transfer DMA va începe atunci când bitul serviceIntr devine 0. Bitul 2 (serviceIntr) dezactivează prin valoarea 1 transferurile DMA şi toate cazurile de întrerupere. Bitul 1 (full) indică prin valoarea 1 faptul ca memoria FIFO nu poate accepta un alt cuvânt (daca bitul Dir al registrului de control este 0) sau este complet plină (Dir = 1). Dacă bitul full este 0, memoria FIFO are cel puţin un cuvânt liber (Dir = 0) sau cel puţin un octet liber (Dir = 1). Acest bit este de tip R/O. Bitul 0 (empty) indică prin valoarea 1 faptul că memoria FIFO este complet goala (Dir = 0) sau conţine mai puţin de un cuvânt (Dir = 1). Dacă bitul empty este 0, memoria FIFO conţine cel puţin un octet (Dir = 0) sau cel puţin un cuvânt (Dir = 1). Acest bit este de tip R/O.

Utilizarea portului ECP este similară cu cea a portului EPP. Se

înscrie un mod de operare în registrul **ecr**, iar apoi transferul de date este realizat prin scrierea sau citirea portului de I/E corespunzător. Toate semnalele de protocol sunt generate automat de controlerul interfeţei. Diferenţa principală este că în cazul portului ECP transferurile sunt realizate de obicei prin DMA şi nu prin I/E programate.

III.3.4 Negocierea modului de transfer

Perifericele nu trebuie să implementeze toate modurile interfeţei *IEEE 1284*. De aceea, este necesară o metodă prin care calculatorul poate determina posibilităţile perifericului şi poate seta interfaţa intr-unul din modurile disponibile. În acest scop s-a elaborat conceptul de *negociere*.

Negocierea este o secvenţă de operaţii ale interfeţei portului paralel, care nu afectează un dispozitiv mai vechi, dar asigură identificarea unui periferic compatibil cu standardul *IEEE 1284*. Ideea este că un periferic mai vechi nu va răspunde la secvenţa de negociere, iar calculatorul va rămâne intr-un mod compatibil cu perifericul, în timp ce un periferic *1284* va răspunde la această secvenţă şi va putea fi setat in oricare mod disponibil.

În timpul fazei de negociere, calculatorul plasează un cod de cerere pe liniile de date şi apoi iniţiază secvenţa de negociere. Cererea poate fi de a plasa interfaţa intr-un anumit mod, sau de a solicita un identificator de dispozitiv (ID) de la periferic, care va permite calculatorului identificarea tipului de periferic ataşat. Identificatorul poate fi returnat în oricare mod de transfer în sens invers, diferit de EPP. În timpul negocierii se utilizează un *octet de extensibilitate*. Semnalul XFlag este utilizat de periferic pentru a confirma faptul că modul cerut este disponibil. Acest semnal va fi setat la 1 ca o confirmare pozitivă pentru toate cererile, cu excepţia modului de transfer pe 4 biţi ("Nibble"). Toate dispozitivele compatibile cu standardul *IEEE 1284* trebuie să admită modul "Nibble" pentru transferurile prin canalul invers.

În Tabelul 3.6 se descriu biţii octetului de extensibilitate. Bitul Extensibility Link este prevăzut pentru a asigura un mecanism de adăugare ulterioară a noi caracteristici şi moduri de operare.

Tabelul 3.6 Valori ale octetului de extensibilitate.

Bit	Descriere	Valori valide
7	Cerere "Extenşibility Link"	1000 0000
6	Cerere mod EPP	0100 0000

5	Cerere mod ECP cu RLE	0011 0000
4	Cerere mod ECP fără RLE	0001 0000
3	(Rezervat)	0000 1000
2	Cerere ID dispozitiv	Returnează datele utilizând modul: Mod “Nibble” 0000 0100 Mod “Byte” 0000 0101 Mod ECP cu RLE 0011 0100 Mod ECP fără RLE 0001 0100
1	(Rezervat)	0000 0010
0	Mod “Byte”	0000 0001
-	Mod “Nibble”	0000 0000

Negocierea şi identificatorii de dispozitiv sunt elemente importante pentru posibilitatea viitoare a sistemelor de a determina perifericele conectate la portul paralel.

III.4 Zone de date BIOS pentru gestiunea imprimantei

Zona de memorie BIOS rezervată pentru gestiunea imprimantei ocupă spaţiul de adrese 0000:0408 – 0000:040F.

0000:0408 – 0000:0409 – adresa de bază a portului imprimantei LPT1.

0000:040A – 0000:040B – adresa de bază a portului imprimantei LPT2.

0000:040C – 0000:040D – adresa de bază a portului imprimantei LPT3.

0000:040E – 0000:040F – adresa de bază a portului imprimantei LPT4.

III.5 Funcţii DOS şi BIOS pentru gestiunea imprimantei

Pentru gestiunea imprimantei sunt rezervate două întreruperi BIOS: INT 05h – imprimă conţinutul memoriei video regim text şi INT 17h – acces la porturile paralele, şi întreruperea DOS INT 21h funcţia 05h care permite imprimarea unui caracter.

INT 21h funcţia 05h – AH = 05h – numărul funcţiei şi DX = codul ASCII al caracterului.

INT 05h – lansează procedura BIOS pentru imprimarea conţinutului memoriei video pagina activă. Această întrerupere este lansată de procedura întreruperii hard BIOS INT 09h la identificarea tastării tastei *PrtSc*. Această întrerupere poate fi lansată şi din programul utilizatorului sau poate fi modificată prin înlocuirea cu procedura proprie de imprimare a memoriei video.

În Tabelul 3.7 se prezintă principalele funcţii BIOS pentru gestiunea imprimantei, accesibile prin întreruperea **INT 17h**.

Tabelul 3.7 Funcţii BIOS pentru gestiunea imprimantei.

AH	Funcţia realizată	Parametru de intrare	Parametru de ieşire
00h	Imprimă un caracter	AL = codul ASCII al caracterului pentru imprimare; DX = numărul imprimantei (0, 1 sau 2).	AH = 01h –eroare, caracterul nu este imprimat, sau codul cuvântului de stare a imprimantei după imprimarea caracterului.
01h	Iniţierea portului imprimantei.	DX = numărul imprimantei (0, 1 sau 2).	AH = codul cuvântului de stare a imprimantei.
02h	Citeşte cuvântul de stare al imprimantei.	DX = numărul imprimantei (0, 1 sau 2).	AH = codul cuvântului de stare a imprimantei. D0 = 1 – TimeOut; D3 = 1 – eroare de intrare/ieşire; D4 = 1 – imprimanta este selectată; D5 = 1 – lipseşte suportul; D6 = 1 – confirmare; D7 = 1 – imprimanta este liberă.

III.6 Desfăşurarea lucrării

1. Să se determine resursele rezervate de sistem pentru gestiunea

imprimantei.

2. Să se scrie un program pentru detectarea porturilor SPP ale unui calculator.

3. Să se scrie un program pentru transmiterea unui bloc de date, cules la tastatură, la imprimantă utilizând adresarea directă a porturilor de intrare/ieşire.

4. Să se scrie un program care permite transmiterea unui fişier la imprimantă utilizând adresarea directă a porturilor de intrare/ieşire.

5. Să se scrie un program care permite imprimarea unui bloc de date (fişier) utilizând întreruperea INT 17h.

III.7 Conţinutul dării de seamă

1. Sarcina şi scopul lucrării.

2. Schema de structură a imprimantei examinate.

3. Schema bloc a algoritmului de funcţionare a imprimantei examinate.

4. Analiza comparativă a imprimantelor moderne. Avantajele şi dezavantajele acestora.

5. Formatul blocului de date de comunicare a imprimantei cu calculatorul. Diagramele de timp.

6. Schema bloc a programului de imprimare a datelor.

7. Programul elaborat.

8. Concluzii.

IV. LUCRAREA DE LABORATOR NR. 3 ŞI 4.
„*Studierea şi programarea sistemului video. Regim test şi grafic*”

IV.1 Scopul lucrării

Studierea şi programarea sistemului video în regim text şi grafic. Se urmăreşte de a studia structura, standardele şi tehnologiile de realizare a sistemelor video, interfaţa video şi modul de funcţionare. Studierea funcţiilor standard BIOS şi DOS, destinate pentru gestiunea sistemului video şi metodele de utilizare a acestora în tehnicile de programare.

IV.2 Resurse hardware şi software necesare pentru efectuarea lucrării

1. Calculator PC;
2. Videomonitoare de diferite modele;
3. Osciloscop;
4. Compilator C / C++;
5. Compilator Assembler;
6. Turbo Help DOS/BIOS 3 / 6.

IV.3 Consideraţii teoretice

Dispozitivele de afişare video sunt alcătuite din două componente de bază:

- monitorul – video display care realizează afişarea efectivă pe baza comenzilor primite de la adaptor;
- adaptorul video – generează comenzile de afişare către monitor, efectuând totodată şi prelucrarea finală a informaţiilor ce vor fi afişate.

Monitoarele se clasifică în monitoare cu tub catodic, plasmă şi cristale lichide. În funcţie de tehnologia utilizată la afişare, PC-urile se clasifică în două categorii:

- desktop-uri;
- laptop-uri şi notebook-uri.

Cele mai multe PC-uri desktop folosesc monitoare bazate pe tehnologia tuburilor cu raze catodice sau cristale lichide, laptop-urile şi notebook-urile folosesc o tehnologie bazată pe cristale lichide.

Monitoarele bazate pe tehnologia **CRT (Cathode Ray Tub)** conţin următoarele blocuri funcţionale de bază: cinescopul umplut cu un gaz inert cu catodul (tun de electroni) dintr-o parte şi ecranul (anodul) cu un

înveliş de fosfor din cealaltă parte. Diferenţa de potenţial de 25000 volţi dintre anod şi catod generează o viteză de deplasare a electronilor aproape de viteza luminii. Lovindu-se de anod (învelişul de fosfor) energia cinetică se transformă în lumină vizibilă. Calitatea imaginii unui monitor CRT este influenţată de patru factori:

1) Tipul de fosfor care determină culoare imaginii ecranului:
   - monocrome – folosesc un singur strat omogen de fosfor de culoare verde, negru sau alb;
   - color – utilizează trei tipuri de fosfor roşu, verde şi albastru. Variind intensitatea fiecărei culori se poate obţine un spectru extins de nuanţe.
2) Generatorul de electroni care deplasează electronii sub forma unui fascicol de mare precizie şi viteză. Pentru deplasarea fascicolului pe ecran sunt utilizaţi electromagneţi puternici. Monitoarele monocrom includ un singur generator de electroni, cele color includ trei aşezate în triunghi care funcţionează sincron iar fluxurile de electroni converg către acelaşi pixel.
3) Masca de umbre sau grila de deschidere se utilizează pentru a exclude împrăştierea fascicolului de electroni pe alte culori. În acest scop în spatele stratului de fosfor la o distanţă mică este plasat un strat de metal cu orificii foarte fine care permite fascicolului de electroni să pătrundă numai la particulele de fosfor de o anumită culoare, celelalte două fiind mascate.
   Tuburile Trinitron folosesc o grilă de deschidere plasată într-o reţea de tuburi verticale separate prin nişte şanţuri. Particulele de fosfor se găsesc pe partea interioară a tuburilor sub forma unor benzi ale celor trei culori de bază, grila nu permite ca fluxul de electroni să ajungă la o altă bandă de culoare decât la cea solicitată.
4) Reflexia luminii. Cu cît monitorul este mai plat, cu atât reflexia este mai mică.

Dimensiunile cele mai frecvente ale monitoarelor sunt: 14’, 15’, 16’, 17’, 18’, 19’, 20’, 21’ şi 22’.

Nivelul de detaliere a unei imagini pe ecran este dat de rezoluţia acesteia măsurată în pixeli.

Principalele caracteristici ale monitoarelor CRT sunt:

- rezoluţia;
- rata de scanare;

- rata de împrospătare;
- standardul video;
- frecvenţa de scanare;
- dimensiunea benzii.

LCD (Liquid Cristal Display). LCD-urile au un ecran plat, cu o strălucire mai mică şi putere de consum redusă. Calitatea culorii este mai înaltă iar rezoluţia mai redusă faţă de cele CRT. Există trei categorii de monitoare LCD: matrice pasivă monocrom, matrice pasivă color şi matrice activă color.

LCD cu matrice pasivă – fiecare celulă este controlată de încărcături electrice transmise de tranzistor în raport cu rândul şi coloana poziţiei pe marginea ecranului. Pentru a creşte strălucire ecranul se împarte în două părţi care se regenerează în paralel.

LCD cu matrice activă – fiecare celulă are tranzistorul propriu. O performanţă mai înaltă oferă monitoarele LCD cu matrice activă cu tranzistor pe film subţire TFT în care fiecare pixel este controlat de trei tranzistori, câte unul pentru fiecare culoare. Ecranul este împrospătat şi redesenat imediat.

Adaptorul video este un ansamblu de circuite care realizează procesarea informaţiilor care se afişează pe monitor şi formează semnalele de comandă pentru monitor:

- culoare fiecărui pixel RGB;
- semnalele de sincronizare a baleierii de sus în jos şi de la stânga la dreapta;
- semnalul de sincronizare a liniilor pe orizontală HSYN;
- semnalul de sincronizare a cadrelor pe verticală VSYN.

Componentele de bază ale unui adaptor video sunt:

- memoria video;
- controlorul grafic;
- controlorul de atribute;
- registrele de control;
- convertorul numeric-analog;
- conectorul pentru cuplarea monitorului;
- conectorul de magistrală;
- conectorul de extensie;
- video BIOS.

Standarde IBM şi VESA. O placă video furnizează semnale în unul dintre următoarele standarde de fabricaţie:

- MDA – monochrome display adapter;
- CGA – color graphics adapter;
- EGA – enhanced graphics adapter;
- VGA – video graphics array;
- SVGA – super VGA;
- XGA – extended graphics array.

IV.4 Funcţii DOS şi BIOS pentru sistemul video

În sistemul BIOS s-a rezervat întreruperea INT 10h care permite setarea regimului de funcţionare a video-sistemului, afişarea caracterelor pe ecran, precum şi pentru utilizarea modurilor grafice. Pentru modurile grafice cu rezoluţii mai mari nu este recomandabilă utilizarea acestei întreruperi pentru că este lentă; se recomandă scrierea directă în memoria video. Să luăm ca exemplu afişarea unui caracter pe ecran: apelarea întreruperii 10h implică execuţia unui cod destul de mare (interpretarea parametrilor transmişi în registre, stabilirea sub-funcţiei apelată etc.) , pe când pentru scrierea directă în memoria video este necesară o singură instrucţiune de tip MOV (eventual două pentru stabilirea atributelor caracterului) .

Totuşi, această întrerupere este foarte practică pentru programele care nu afişează pe ecran cantităţi mari de informaţie la un moment dat. Folosind această întrerupere programatorul nu mai trebuie să calculeze adresele memoriei video în care să scrie fiecare caracter.

În tabelul 4.1. sunt prezentate descrierea funcţiilor întreruperii INT 10h.

Tabelul 4.1 Funcţii BIOS pentru sistemul video.

(AH)	**Funcţia realizată**	**Parametrii de intrare**	**Parametrii ieşire**
00h	Selecţia modului de lucru a sistemului video	Conform tabelului 4.2.	
01h	Selecţia formei şi a dimensiunii cursorului	(CH) biţii 0-4 linia de început a cursorului; (CH) biţii 5-7=0; (CL) biţii 0-4 linia de sfârşit a cursorului; (CL) biţii 5-7=0.	

02h	Poziţionarea cursorului pe ecran	(DH, DL) numărul liniei şi coloanei, (0, 0) – colţul sus stânga; (BH) – numărul paginii, =0 pentru mod grafic.	
03h	Citire coordonate cursor	(BH) – numărul paginii; =0 pentru mod grafic	(DH, DL) lin. , col. (CH, CL) forma
04h	Citire poziţie indicator optic	(AH) =0 light pen inactiv; =1 light pen activ; (DH, DL) lin. , col. cursor; (CH) linie pixel (0-199); (BX) col pixel (0-319/639)	
05h	Selecţie pagină activă	(AL) – nr. pagină 0-7 pt mod 0 şi 1 0-3 pt mod 2 şi 3	
06h	Execuţie operaţie “scroll up”	(AL) – nr. de linii; (AL) =0 ştergere fereastră; (CH, CL) – lin. , col. colţului stânga sus; (DH, DL) – lin. , col. colţului dreapta jos; (BH) – atributul unei linii albe.	
07h	Execuţia operaţiei “scroll down”	(AL) – nr. de linii; (AL) =0 ştergere fereastră; (CH, CL) – lin. , col. colţului stânga sus; (DH, DL) – lin. , col. colţului dreapta jos; (BH) – atributul unei linii albe.	
08h	Citire caracter de pe ecran şi determinarea atributului sau. (se citeşte caracterul din poziţia curentă a cursorului)	(BH) – nr. pagină referită (numai pentru mod alfanumeric)	(AL) – caracterul citit; (AH) – atributul caracterului.

09h	Afişarea caracterului pe ecran (în poziţia curentă a cursorului)	(BH) – nr. pagină referită (numai pt mod alfanumeric); (BL) – atributul caracterului (în mod alfanumeric) ; - culoarea (în mod grafic); (CX) – nr. caractere de afişat; (AL) – caracterul.	
0Ah	Înlocuire caractere pe ecran cu păstrarea caracteristicilor de culoare	(BH) – nr. pagină referinţă; (CX) – nr. caractere de afişat; (AL) – caracterul.	
0Bh	Fixarea caracteristicii de culoare (numai pt modul grafic 320x200 pixeli)	(BH) – index paletă (conf doc IBM-PC); (BL) – valoarea culorii în paletă.	
0Ch	Afişarea unui punct pe ecran (pt modul grafic)	(DX) – nr. liniei; (CX) – nr. coloanei; (AL) – valoarea culorii (0, 1, 2, 3).	
0Dh	Citire culoare punct de pe ecran (pt modul grafic)	(DX) – nr. liniei; (CX) – nr. coloanei.	(AL) – culoarea citită
0Eh	Afişare caracter pe ecran cu actualizarea poziţiei cursorului	(AL) – caracterul de afişat; (BL) – culoare fond (pt mod grafic); (BH) – pagina (pt mod alfanumeric);	
0Fh	Citire caracteristici mod de lucru curent		(AL) – modul curent; (AH) – nr coloane caracter; (BH) – nr paginii.

În tabelul 4.2. sunt prezentate modurile de funcţionare a sistemului video.

Tabelul 4.2 Modurile de funcţionare a sistemului video.

Mod	**Regim**	**Rezoluţie text**		**Rezoluţie grafică**	**Culori**	**Nr. pagini**	**RAM video**	**Standard SVGA**
00H	Text	40x25	8x8	320x200	16	8	B8000H	CGA
			8x14	320x350				EGA
			9x16	360x400				VGA
01H	Text	40x25	8x8	320x200	16	8	B8000H	CGA
			8x14	320x350				EGA
			9x16	360x400				VGA
02H	Text	80x25	8x8	640x200	16	8	B8000H	CGA
			8x14	640x350				EGA
			9x16	720x400				VGA
03H	Text	80x25	8x14	640x350	16	8	B8000H	CGA
			9x14	720x350				EGA
			9x16	720x400				VGA
04H	Grafic	40x25	8x8	320x200	4	1	B8000H	CGA
05H	Grafic	40x25	8x8	320x200	4	1	B8000H	CGA
06H	Grafic	80x25	8x8	640x200	a/n	1	B8000H	CGA
07H	Text	40x25	9x14	720x350	mono	1	B8000H	MDA
08H	Grafic	20x25	8x8	160x200	16	1	B8000H	PCjr
09H	Grafic	40x25	8x8	320x200	16	1	B8000H	PCjr
0AH	Grafic	80x25	8x8	640x200	4	1	B8000H	PCjr
0DH	Grafic	40x25	8x8	320x200	16	8	A0000H	EGA
0EH	Grafic	80x25	8x8	640x200	16	4	A0000H	EGA
0FH	Grafic	80x25	8x14	640x350	mono	2	A0000H	EGA
10H	Grafic	80x25	8x14	640x350	16	2	A0000H	EGA
11H	Grafic	80x30	8x16	640x480	mono	1	A0000H	VGA
12H	Grafic	80x30	8x16	640x480	16	1	A0000H	VGA
13H	Grafic	40x30	8x8	320x200	256	1	A0000H	VGA
4EH	Text	80x60	8x8	640x480	16	1	B8000H	SVGA
4FH	Text	132x60	8x8	1056x480	16	1	B8000H	SVGA
50H	Text	132x25	8x14	1056x350	16	1	B8000H	SVGA
51H	Text	132x43	8x8	1056x344	16	1	B8000H	SVGA
52H	Grafic	100x37	8x16	800x600	16	1	A0000H	SVGA
53H	Grafic	80x30	8x16	640x480	256	1	A0000H	SVGA
54H	Grafic	100x37	8x16	800x600	256	1	A0000H	SVGA
55H	Grafic	128x48	8x16	1024x768	4	1	A0000H	SVGA
56H	Grafic	128x48	8x16	1024x768	16	1	A0000H	SVGA
57H	Grafic	96x64	8x16	768x1024	16	1	A0000H	SVGA
58H	Grafic	160x64	8x16	1280x1024	16	1	A0000H	SVGA
59H	Grafic	128x48	8x16	1024x768	256	1	A0000H	SVGA

În tabelul 4.3 sunt prezentate funcţiile DOS pentru sistemul video.

Tabelul 4.3 Funcţiile DOS – INT21h pentru sistemul video.

(AH)	Funcţia realizată	Parametrii de intrare	Parametrii de ieşire
01h	Citeşte un caracter de la tastatură şi-l trimite în ecou la ecran. Dacă se apasă CTRL-BREAK se execută INT 23h		(AL) – caracterul introdus
02h	Afişare caracter la ecran. Dacă se apasă CTRL-BREAK se execută INT 23h	(DL) – caracterul	
09h	Afişarea unui şir din memorie care se termină cu $ (24h)	(DS:DX) – adresa şir	

IV.5 Desfăşurarea lucrării „Studierea sistemului video în regim text”

1. Să se determine resursele rezervate de sistem pentru gestiunea sistemului video regim text.
2. Să se scrie un program „redactor textual” care permite efectuarea tuturor operaţiilor specifice (afişare text, redactare text, deplasare cursor, modificare culoare text şi fundal) utilizând adresarea directă a memoriei video.
3. Să se scrie un program „redactor textual” care permite efectuarea tuturor operaţiilor specifice (afişare text, redactare text, deplasare cursor, modificare culoare text şi fundal) utilizând funcţii ale întreruperii INT 10h.

IV.6 Desfăşurarea lucrării „Studierea sistemului video în regim grafic”

1. Să se determine resursele rezervate de sistem pentru gestiunea sistemului video regim grafic.
2. Să se scrie un program „redactor grafic” care permite efectuarea tuturor operaţiilor specifice (desenare, ştergere, modificare culoare)

utilizând adresarea directă a memoriei video şi funcţii de identificare a poziţiei cursorului grafic utilizând întreruperea INT 33h.

3. Să se scrie un program „redactor grafic” care permite efectuarea tuturor operaţiilor specifice (desenare, ştergere, modificare culoare) utilizând funcţii ale întreruperii INT 10h şi funcţii de identificare a poziţiei cursorului grafic utilizând întreruperea INT 33h.

IV.7 Conţinutul dării de seamă

1. Sarcina şi scopul lucrării.

2. Schema de structură a sistemului video (regim text, regim grafic).

3. Schema bloc a algoritmului de funcţionare a sistemului video.

4. Analiza comparativă a standardelor de sisteme video. Avantajele şi dezavantajele acestora.

5. Schema bloc a programului de funcţionare a produsului program (regim text, regim grafic).

6. Programul elaborat.

7. Concluzii.

V. LUCRAREA DE LABORATOR NR. 5 ŞI 6.
„*Studierea şi programarea FDD şi HDD*”

V.1 Scopul lucrării

Studierea şi programarea dispozitivelor FDD şi HDD. Se urmăreşte de a studia structura, standardele şi tehnologiile de realizare a dispozitivelor de înregistrare a informaţiei pe suport magnetic FD şi HD. Studierea funcţiilor standard BIOS şi DOS, destinate pentru gestiunea dispozitivelor FDD şi HDD şi metodele de utilizare a acestora în tehnicile de programare.

V.2 Resurse hardware şi software necesare pentru efectuarea lucrării

1. Calculator PC;
2. Dispozitive FDD şi HDD de diferite modele;
3. Osciloscop;
4. Compilator C / C++;
5. Compilator Assembler;
6. Programul DISKEDIT.EXE;
7. Turbo Help DOS/BIOS 3 / 6.

V.3 Consideraţii teoretice

Discul flexibil (FD) este confecţionat din material plastic acoperit cu un strat de oxid magnetizabil. Este împachetat din construcţie cu scopul protejării de mediul extern.

În funcţie de dimensiune există două tipuri de FD:

- FD de 5 ¼ inches;
- FD de 3 ½ inches.

Fizic un FD este realizat în piste concentrice numerotate de la exterior către interior. Fiecare pistă fiind compusă dintr-un număr de sectoare. Un sector conţine un număr de bytes (128, 256, 512, 1024). Înregistrare datelor pe un FD se efectuează pe ambele suprafeţe.

Majoritatea dischetelor sunt construite din aceleaşi materiale de bază un material plastic acoperit cu un înveliş de oxid de fier, cobalt-fier sau bariu-fier.

Densitatea este o măsură a cantităţii de informaţie care poate fi înregistrată într-un anumit spaţiu pe suprafaţa de înregistrare. Există două tipuri de densitate: longitudinală şi liniară. Densitatea longitudinală

este caracterizată prin numărul de piste ce pot fi înregistrate pe dischetă, exprimată ca număr de piste pe inch (TPI). Densitatea liniară este capacitatea unei singure piste de a înregistra informaţii, indică ca număr de biţi pe inch (BPI).

Pentru a stoca cît mai multă informaţie tot mai multe firme au început să folosească formatul DMF care permite stocarea a 1.7 MB pe dischetele de 3.5". Formatul DMF diferă de formatul standard prin faptul că foloseşte 21 de sectoare pe pistă prin reducerea pauzelor dintre înregistrări la 9 biţi. Fiecare pistă utilizată este intercalată cu alta, astfel încât sectoarele nu apar în ordine consecutivă. Neajunsul acestui format este lipsa posibilităţii de înregistrare a informaţiei de FDD normale, iar sistemul DOS nu poate citi dischetele DMF.

Pentru dischete sunt cunoscute 4 tipuri de formatări. Driverul dischetelor şi sistemul de operare se ajustează automat la formatul dischetelor. Toate driverele de capacităţi mari pot citi dischete formatate la o capacitate mai mică.

Unitatea de disc flexibil îndeplineşte următoarele funcţii: rotirea dischetei cu o viteză constantă, deplasarea capului de citire/înscriere cu o precizie suficientă pentru a localiza fiecare pistă de pe disc, recunoaşterea locului de început şi deplasarea relativă la adresa de bază.

O unitate de disc flexibil este alcătuită din următoarele componente:

- capete de citire/înscriere;
- mecanism de acţionare a capetelor de citire/înscriere;
- motorul de rotire a discului;
- linii de ghidare dreapta/stînga FD;
- comutator de protejare a scrierii;
- placa cu circuite integrate;
- conector de alimentare;
- cabluri conectoare pentru transferul datelor şi a semnalelor de control.

Interfaţa unităţii de disc flexibil asigură conectarea FDD la ansamblul componentelor PC-ului pentru a deveni funcţional. Portul adaptor de FDD admite conectarea a două FDD. FDD dispun de blocuri selectoare ce se configurează pentru a selecta numărul unităţii.

Înscrierea şi citirea informaţiei se efectuează prin apelul la întreruperea standard a BIOS-ului destinată pentru deservirea FD. La iniţierea unei operaţii cu FD BIOS-ul trimite un cod corespunzător aplicaţiei ce a solicitat accesul. Sistemul de operare trimite instrucţiunile

către BIOS, care trimite codul portului la controlorul de FD. Controlorul indică FDD-ului unde să deplaseze capetele de citire/scriere şi ce operaţie să execute. Controlorul FDD-ului are două funcţii principale: conversia comenzilor generate de BIOS în semnale ce controlează FDD şi conversia semnalelor generate de capetele de citire/scriere într-o formă înţeleasă de celelalte componente ale PC-ului.

Două semnale controlează poziţia capetelor de citire/scriere: step pulse – activează motorul de deplasare a capetelor iar semnalul de direcţie indică în ce direcţie să se deplaseze capetele spre centrul (1) sau spre marginea (0) dischetei. Pentru a selecta faţa cu care se efectuează operaţia se utilizează semnalul numit selector de scriere, pentru (1) se selectează faţa de sus, pentru (0) se selectează faţa de jos a dischetei. Scrierea pe dischetă este efectuată de două semnale: Write Data care conţine informaţia de înscriere şi Write Enable permite operaţia de înscriere pe dischetă. Rata transferului de date variază funcţie de tipul driverului.

Controlerul primeşte patru semnale de la FDD. Două determină locul poziţionării capului, al treilea determină prezenţa dischetei în FDD, al patrule protecţia informaţiei.

Controlorul FDD ţinând cont de instrucţiunea generată de BIOS şi datele recepţionate de la FDD formează o consecutivitate de semnale pentru a poziţiona capul pe pista respectivă şi identifică sectorul respectiv, după aceea începe operaţia de citire/scriere.

Discul fix (HD) este principalul dispozitiv de stocare a datelor şi extindere a capacităţii memoriei RAM cu memoria virtuală. El oferă o viteză sporită de comunicare, capacitate mare şi facilităţi de instalare. HD diferă ca tehnologie de fabricaţie, interfaţă, viteză şi capacitate de stocare a datelor.

Mediul de memorie a HD este alcătuit dintr-o colecţie de platane circulare din aluminiu acoperit cu o strat de material magnetic. Modul de funcţionare al unei unităţi de HD este similar cu cel al unei unităţi de FD. Are platane care se rotesc în jurul unui ax şi capete care se deplasează. Informaţia se stochează pe piste şi sectoare logice. Majoritatea HD au mai multe platane fixate pe o axă. Mulţimea pistelor care se află la aceiaşi distanţă faţă de centru formează un cilindru.

Platanele sunt confecţionate din aliaj de aluminiu cu duritate mare şi greutate mică. Unele unităţi folosesc platane confecţionate din sticlă. Platanele sunt acoperite cu un strat de substanţă magnetică care

constituie suportul magnetic ce memorizează informaţia. La platanele HD sunt utilizate două tipuri de suport magnetic: strat de oxid şi peliculă subţire. Tehnologia de peliculă subţire permite obţinerea unei generaţii de HD cu distanţe mai mici între capete şi disc, ceea ce duce la o densitate mai mare a datelor pe disc. Acoperirea platanelor se realizează prin două modalităţi: placare electrolitică şi metalizare în vid.

Viteza de rotaţie a discurilor aste 3600 rpm, 4500rpm, 5400 rpm, 7200 rpm şi 10000 rpm. Viteza de rotaţie mare, combinată cu un mecanism de poziţionare a capetelor foarte rapid şi mai multe sectoare pe pistă fac ca HD să fie foarte rapide. Capetele de citire/scriere nu ating platanele în timpul funcţionării. La scoaterea de sub tensiune a calculatorului, capetele coboară pe suprafaţa platanelor în momentul când acestea se opresc din rotaţie. Când unitatea este pusă în funcţiune, se formează o pernă de aer care ţine fiecare cap suspendat la o distanţă mică deasupra sau dedesubtul platanului.

Unitatea de disc fix HDD. Există multe tipuri de HD, dar aproape toate au aceleaşi componente de bază, diferă modul de realizare sau materialele folosite la proiectarea lor. Componentele de bază ale unei unităţi de HD tipice sunt:

- platanele – mediul de memorare;
- capete de citire/scriere;
- mecanism de poziţionare a capetelor;
- motorul de rotaţie a platanelor;
- circuitul electronic de comandă şi control a unităţii HD;
- cabluri şi conectori;
- elemente de configurate;
- carcasă.

Tipurile de capete de citire/scriere care se folosesc în unităţile de HDD:

- capete de ferită;
- capete cu pelicule subţiri;
- capete magneto-rezistive.

Organizarea şi adresarea datelor. Datele informaţionale sunt memorizate pe sectoare. Fiecare sector are adresa proprie care se defineşte din: numărul cilindrului, numărul platanei, suprafaţa platanei, numărul capului, numărul sectorului pe pistă.

Interfeţe pentru unităţile de HD sunt următoarele:

- ST-506/412;

- ESDI;
- IDE, ADA;
- SATA;
- SCSI.

Interfaţa IDE este componentă a standard a plăcii de bază şi formează un minimum de semnale de dirijare cu HD. Schimbul de date este efectuat la nivel de instrucţiuni şi blocuri de date în cod paralel. Pe parcurs au fost implementate mai multe standarde ale interfeţei IDE cum ar fi ATA IDE, ATA-1, ATA-2, ATA-3.

Interfaţa SCSI este o interfaţă la nivel de sisteme cu o magistrală care acceptă până la 8 echipamente. Magistrala SCSI nu comunică direct cu echipamentele periferice ci cu controlorul care este inclus în unitate. Pe parcurs sau dezvoltat mai multe standarde SCSI. Cele mai răspândite sunt: SCSI-1, SCSI-2, SCSI-3.

Capacitatea HD cu interfaţă IDE, SATA şi SCSI variază de la sute de GB până la zeci de TB.

V.4 Funcţii DOS şi BIOS pentru FDD şi HDD

Accesul direct la HDD şi la FDD este asigurat de întreruperea INT 13h. INT 13h permite citirea şi scrierea sectoarelor pe disc în mod direct, fără a ţine cont de sistemul de fişiere existent pe disc. De aceea nu este recomandată folosirea acestei întreruperi pentru lucrul cu fişiere, fiind preferabile funcţiile DOS pentru aceste operaţii. Există însă situaţii când utilizarea acestei întreruperi este singura alternativă: citirea unui disc pe care se află alt sistem de fişiere decât FAT.

Cea mai importantă diferenţă între accesul la discheta şi accesul la hard disk, pentru citirea mai multor sectoare, este că la hard disk se incrementează automat capul / cilindrul curent, în mod automat.

Adresarea dispozitivelor are loc conform datelor de mai jos:

DL = 00h - selectează dispozitivul FDD A:

DL = 01h - selectează dispozitivul FDD B:

DL = 80h - selectează dispozitivul HDD 1

DL = 81h - selectează dispozitivul HDD2

Funcţiile întreruperii INT 13h:

AH = 00h - resetează dispozitivul selectat

AH = 01h - identifică starea dispozitivului selectat

AH = 02h - citeşte sectorul de pe dispozitiv

AH = 03h - scrie sectorul pe dispozitiv

AH = 04h - verifică sectorul

AH = 05h - formatare pistă

AH = 08h - citeşte parametrii dispozitivului

AH = 09h - iniţierea controlorului

V.5 Desfăşurarea lucrării

1. Să se determine resursele rezervate de sistem pentru gestiunea FDD şi HDD;

2. Faceţi cunoştinţă cu dispozitivul FDD şi HDD propus spre examinare;

3. Să se determine structura informaţiei pe FD şi HD utilizând programul DISKEDIT.EXE;

4. Să se elaboreze programul care permite scrierea şi citirea datelor de pe sectorul cu adresa indicată utilizând funcţii ale întreruperii INT 13h;

5. Să se elaboreze programul care permite: crearea unui directorii, eliminarea unui directorii, trecerea în directorii, vizualizarea conţinutului unui directorii, coprierea unui fişier, ştergerea unui fişier şi modificarea numelui fişierului utilizând funcţii DOS;

V.6 Conţinutul dării de seamă

1. Sarcina şi scopul lucrării;
2. Resursele rezervate de sistem pentru gestiunea FDD şi HDD;
3. Schema de structură a dispozitivului examinat;
4. Bloc schema algoritmului programului elaborat;
5. Programul elaborat;
6. Concluzii.

VI. Lucrarea de laborator Nr. 7
„*Studierea şi programarea sistemului de întreruperi*”

VI.1 Scopul lucrării

Familiarizarea studenţilor cu sistemul de întrerupere, prezentarea principiului de funcţionare al unui sistem de întreruperi, analiza sistemului de întreruperi al familiei de procesoare Intel x86 şi studierea modului de utilizare a întreruperilor la un calculator personal compatibil IBM-PC AT.

VI.2 Resurse hardware şi software necesare pentru efectuarea lucrării

1. Calculator PC;
2. Compilator C / C++;
3. Compilator Assembler;
4. Turbo Help DOS/BIOS 3 / 6.

VI.3 Consideraţii teoretice

Principiul de funcţionare al sistemului de întreruperi.

Sistemul de întreruperi este acea parte a unui sistem de calcul care permite detecţia unor evenimente externe sau interne şi declanşarea unor acţiuni pentru tratarea lor. Astfel de evenimente pot fi: recepţia unui caracter pe un canal serial, golirea unui registru de transmisie, impuls generat de un contor de timp, tentativa de execuţie a unui cod de instrucţiune ne-permis (inexistent sau protejat), terminarea unei anumite operaţii de către o interfaţă, eroare în timpul execuţiei unei operaţii aritmetice (împărţire la zero) şi multe altele. Întreruperile permit calculatorului să reacţioneze rapid la aceste evenimente, să se sincronizeze cu ele şi să le trateze în timp util. O alternativă la sistemul de întreruperi ar fi testarea periodică prin program a tuturor indicatorilor de stare şi a semnalelor de intrare. Această soluţie este ineficientă în cazul în care numărul de elemente care trebuie testate este mare.

Pentru un sistem de calcul, politica de soluţionare a întreruperilor caracterizează adaptabilitatea sistemului la stimuli interni şi externi. Majoritatea sistemelor de întrerupere utilizează un sistem de priorităţi pentru a stabili ordinea de deservire a cererilor concurente de întrerupere. Prioritatea se stabileşte pe baza importanţei acordate evenimentului tratat şi a restricţiilor de timp în soluţionarea întreruperii. Politica de priorităţi

trebuie să asigure soluţionarea echitabilă şi în timp util a tuturor cererilor.

Un calculator poate să identifice mai multe tipuri de întrerupere (numite nivele de întrerupere). Pentru fiecare nivel se poate defini câte o rutină de tratare a întreruperii respective. Adresele de început ale acestor rutine se păstrează intr-o tabela de pointeri denumită tabela de întreruperi. Aceste rutine sunt activate la apariţia şi acceptarea de către calculator a întreruperii corespunzătoare. În funcţie de cerinţele aplicaţiei executate, anumite nivele de întrerupere pot fi invalidate, temporar sau pe toată durata aplicaţiei. O întrerupere invalidată (sau mascată) nu este recunoscută de către calculator.

La activarea unui semnal de întrerupere se testează dacă nivelul corespunzător este validat şi dacă nu sunt în curs de deservire alte întreruperi mai prioritare; în caz afirmativ are loc întreruperea temporară a secvenţei curente de execuţie, se salvează pe stivă adresa instrucţiunii următoare şi se face salt la rutina de tratare a întreruperii. După execuţia rutinei de întrerupere se revine la secvenţa întreruptă prin încărcarea adresei salvate pe stivă.

Adesea, pentru controlul întreruperilor externe se utilizează un circuit specializat denumit controlor de întreruperi. Un astfel de circuit deserveşte un set de semnale de întrerupere. Funcţiile tipice ale unui astfel de controlor sunt: detecţia condiţiei de producere a unei întreruperi (front crescător al semnalului de întrerupere), arbitrarea cererilor multiple, mascarea unor întreruperi, evidenţa întreruperilor deservite, etc.

În cadrul setului de instrucţiuni al unui procesor pot să existe instrucţiuni dedicate pentru tratarea întreruperilor (validarea / invalidarea întreruperilor, revenirea din rutina de întrerupere, simularea software a unor întreruperi hardware, etc.).

Un sistem de întreruperi poate să aibă mai multe moduri de funcţionare; aceste moduri diferă prin: modul de alocare a priorităţilor (fixa, rotativă), modul de detecţie a semnalului de întrerupere (pe front sau pe nivel), acceptarea sau nu a întreruperilor imbricate (întrerupere imbricată = întrerupere acceptată în timpul execuţiei unei alte rutine de întrerupere), numărul de nivele de imbricare, etc.

Întreruperile sunt o soluţie de implementare a unor activităţi concurente şi joacă un rol important în realizarea sistemele de operare multitasking şi de timp-real.

Sistemul de întreruperi al arhitecturii Intel x86.

Arhitectura Intel x86 accepta 256 de nivele de întrerupere. Întreruperile pot fi provocate de evenimente interne sau externe procesorului. Sursele interne de întrerupere sunt: tentativa de divizare cu zero, execuţia unei instrucţiuni de întrerupere software (INT n, sau INTO – interrupt on overflow), tentativa de execuţie a unei instrucţiuni ne-permise (protejate) şi indicator de trasare (Trace) setat. Întreruperile externe sunt de doua tipuri:

- ne-mascabile – generate prin activarea semnalului NMI;
- mascabile – generate prin activarea semnalului INT.

Întreruperile mascabile pot fi invalidate la nivelul procesorului prin resetarea indicatorului IF (Interrupt Flag) din registrul de stare program (IF=0); resetarea se realizează implicit la lansarea unei rutine de întrerupere sau în mod explicit prin instrucţiunea CLI (Clear Interrupt). Validarea întreruperilor mascabile se realizează implicit la revenirea dintr-o rutina de întrerupere (prin refacerea registrului de stare) sau în mod explicit prin instrucţiunea STI (Set Interrupt).

Întreruperea ne-mascabilă nu este afectată de starea indicatorului IF. Aceasta întrerupere este de obicei utilizată pentru a indica situaţii critice în funcţionarea unui sistem de calcul (eroare de paritate, fluctuaţii ale tensiunii de alimentare, etc.).

Pentru gestionarea întreruperilor externe mascabile se utilizează un controlor specializat de întreruperi de tip I8259A. La versiunile de procesoare Pentium funcţionalitatea acestui controlor a fost integrată în circuitul procesorului.

Adresele rutinelor de întrerupere sunt păstrate într-o tabelă de întreruperi plasata la începutul spaţiului de adresare (adresa 0000:0000). Tabela conţine 256 de intrări corespunzătoare celor 256 de nivele de întrerupere recunoscute de procesor; o intrare conţine 4 octeţi şi păstrează adresa de offset şi adresa de segment a rutinei de întrerupere. În acest tabel întreruperile interne au poziţii predefinite. Restul poziţiilor pot fi utilizate de sistemul de operare sau de aplicaţiile utilizator. Adresa la care se afla pointer-ul către rutina de întrerupere a întreruperii de nivel *k* se determină prin multiplicarea cu 4 a numărului de nivel *(k*4)*.

Orice întrerupere poate să fie simulată prin program, folosindu-se instrucţiunea INT n (n specifică nivelul de întrerupere). Utilizarea instrucţiunii INT n s-a dovedit utilă în apelarea unor rutine / funcţii ale sistemului de operare. Avantajul utilizării acestor instrucţiuni în locul

instrucţiunilor clasice de apel de procedura (CALL) constă în posibilitatea relocării şi redirectării rutinelor prin mecanismul tabelei de întreruperi. Un utilizator al unei funcţii de sistem nu trebuie să cunoască adresa rutinei care îndeplineşte funcţia respectivă ci numai intrarea în tabela de întreruperi alocată pentru această funcţie. Astfel, versiunile mai noi ale sistemului de operare, în care se modifica adresele funcţiilor de sistem nu afectează programele scrise pentru versiunile mai vechi, deoarece adresa este actualizată în tabela de întreruperi. Sistemul de operare MS-DOS beneficiază din plin de această facilitate. De exemplu: indiferent de versiunea constructivă a calculatorului personal şi a versiunii MS-DOS, funcţia de afişare se poate apela prin instrucţiunea INT 10H.

Sistemul de întreruperi al calculatorului IBM-PC AT.

La calculatoarele personale de tip IBM-PC AT sistemul de întreruperi are un dublu rol:

- folosit în mod clasic pentru sincronizarea procesorului cu o serie de evenimente externe, care au loc mai ales în interfeţele de intrare / ieşire;
- este un mijloc de implementare a apelurilor de funcţii de sistem.

Pentru tratarea întreruperilor hardware se folosesc două controloare de întrerupere conectate în cascada. Împreună controlează 16 cereri externe de întrerupere. În tabelul 7.1 se prezintă modul de alocare şi utilizare a nivelelor de întrerupere.

Dacă se doreşte tratarea în mod diferit a unei anumite întreruperi, atunci întreruperea respectivă trebuie redirectată către o rutină scrisă de utilizator. În acest scop se va înscrie în tabela de întreruperi la intrarea corespunzătoare întreruperii respective, adresa de offset şi adresa de segment a rutinei utilizator. Scrierea se poate face direct printr-o secvenţă de instrucţiuni sau prin apelarea unei funcţii de sistem (INT 21h, funcţia 25H). Se recomandă salvarea vechii adrese înainte de scrierea celei noi, astfel încât la sfârşitul programului utilizator să se poată reface legătura la vechea rutină.

În continuare sunt descrise funcţiile de sistem utilizabile în acest scop:

INT 21h – funcţia 25H – poziţionarea vectorului de întrerupere:

Intrări: AH=25h – numărul funcţiei,

AL=n – numărul întreruperii,

DS:DX – pointer-ul la noua rutină de întrerupere.

INT 21h – funcţia 35h – citirea vectorului de întrerupere:

Intrări: AH=35h – numărul funcţiei,

AL=n – numărul întreruperii,
Ieşire: ES:BX – pointer-ul la rutina de întrerupere.

Tabelul 6.1 Alocarea nivelelor de întrerupere la calculatoarele compatibile IBM PC AT.

Nivel	**Tip întrerupere**	**Semnal**	**Con-trolor**	**Cauza întreruperii / utilizare**
1	2	3	4	5
00h	internă	-	-	împărţire cu zero
01h	internă			execuţie pas cu pas
02h	internă	NMI		întrerupere nemascabila
03h	internă			Breackpoint
04h	internă			depăşire capacitate
05h	software			tipărire ecran
06h	internă			rezervat
07h	internă			rezervat
08h	externă (hardware)	IRQ0	1	ceas sistem
09h	externă (hardware)	IRQ1	1	tastatura
0Ah	externă (hardware)	IRQ2	1	cascadare controlor 2
0Bh	externă (hardware)	IRQ3	1	COM2
0Ch	externă (hardware)	IRQ4	1	COM1
0Dh	externă (hardware)	IRQ5	1	LPT2
0Eh	externă (hardware)	IRQ6	1	controlor de disc flexibil
0Fh	externă (hardware)	IRQ7	1	LPT1
10h	software			interfaţa video
11h	software			determinare configuraţie sistem
12h	software			determinare dimensiune

				memorie
13h	software			interfaţa de disc FD, HD
1	2	3	4	5
14h	software			interfaţa serială
15h	software			servicii extinse
16h	software			interfaţa de tastatură
17h	software			interfaţa de imprimantă
18h	software			salt ROM-Basic
19h	software			încărcător (Bootstrap)
1Ah	software			ceas de timp real
1Bh	software			apăsare CTRL+C
1C	software			întrerupere periodică la dispoziţia utilizatorului
1D	pointer			adresa tabelei de parametri video
1E	pointer			adresa tabelei de parametri pentru disc
1F	pointer			adresa tabelei de caractere grafice
20h	software			terminare program
21h	software			apeluri sistem (DOS)
….				
25-				Citire / scriere

26h				fizica disc
1	2	3	4	5
27h	software			terminare program cu menţinere în memorie
…				
33h	software			suport mouse
…				
70h	externă (hardware)	IRQ8	2	ceas de timp real
71h	externă (hardware)	IRQ9	2	redirectare IRQ2
72h	externă (hardware)	IRQ10	2	liber
73h	externă (hardware)	IRQ11	2	liber
74h	externă (hardware)	IRQ12	2	liber
75h	externă (hardware)	IRQ13	2	coprocesor aritmetic
76h	externă (hardware)	IRQ14	2	controlor disc fix
77h	externă (hardware)	IRQ15	2	liber
78h	software			liber
...	...			...
FFh	software			liber

În cazul în care se doreşte utilizarea unei întreruperi hardware (provocată de un semnal de tip IRQn) atunci trebuie să se valideze nivelul respectiv de întrerupere în controlorul de întreruperi 8259A. Acest lucru se realizează prin resetarea bitului din registrul de mascare, corespunzător intrării utilizate. La terminarea programului întreruperea va fi invalidată prin setarea aceluiaşi bit.

Exemlul 6.1: redirectarea unei întreruperi software:

```
VECT DW      2 DUP(?) ; aici se salvează vechiul vector de întrerupere
;iniţializare
……..
MOV  AH, 35h
```

```
MOV  AL, n
INT  21h              ; citirea vechiului vector de întrerupere
MOV  VECT, BX
MOV  BX, ES
MOV  VECT+2, BX              ; salvare vector
MOV  AX, SEG RUT_INT
MOV  DS, AX           ;DS <- adresa de segment a rutinei de tratare a
                      ; întreruperii
MOV  DX, OFFSET RUT_INT  ; DX <- adresa de offset a rutinei de
                               ; întrerupere
MOV  AH, 25h                   ; funcţia de scriere vector
MOV  AL,n                      ; n – nivel întrerupere
INT  21h              ; înscrierea noului vector în tabela de întreruperi
…….
; program
        ……
; sfârşit program
MOV  AX, VECT+2
MOV  DS, AX
MOV  DX, VECT
MOV  AH, 25h          ; funcţia de scriere vector
MOV  AL, n            ; n – nivel întrerupere
INT  21h              ; înscrierea vechiului vector în tabela de întreruperi
…….
;rutina de tratare a întreruperii
RUT_INT      PROC  FAR      ; noua rutina de tratare a întreruperii
PUSH         r       ; salvarea registrelor utilizate în cadrul rutinei
STI
        ……..
; corpul rutinei
        ……..
; sfârşitul rutinei
POP  r                         ; refacere registre salvate
IRET
RUT_INT      ENDP
```

Exemplul 6.2: programarea unei întreruperi hardware:

Se presupune că întreruperea nu este tratată în sistemul de operare, fapt pentru care nu se va salva vechiul vector de întrerupere:

```
INTA00  EQU     20H     ; adresa portului 0 din controlorul de întreruperi
INTA01  EQU     21H     ; adresa portului 1 din controlorul de întreruperi
EOI             EQU     20h     ; comanda de încheiere întrerupere
MASCA EQU       11011111B       ; masca pentru validarea intrării nr. 5
MASCA1          EQU     00100000B  ; masca pentru invalidarea intrării nr. 5
        ……..
```

```
; iniţializare întrerupere
   CLI                    ; invalidarea întreruperilor mascabile (INTR)
   MOV  AX, SEG RUT_INT
   MOV  DS, AX  ; DS <- adresa de segment a rutinei de tratare a întreruperii
   MOV  DX, OFFSET RUT_INT    ; DX <- adresa de offset a rutinei de
                              ;întrerupere
   MOV  AH, 25h           ; funcţia de scriere vector
   MOV  AL, 5+8           ; 5+8= nivel de întrerupere programat ( primele 8
                  ;întreruperi hardware sunt plasate începând de la nivelul 8)
   INT  21h               ; înscrierea noului vector în tabela de întreruperi
   MOV  DX, INTA01
   IN   AL, DX            ; citire registru de masca
   AND  AL, MASCA         ; şterge bitul corespunzător întreruperii care se
                          ; validează
   OUT  DX, AL            ; validare intrare de întrerupere
   STI
            …….
   ; program
            ……
   ; sfârşit program
   CLI
   MOV  DX, INTA01
   IN   AL, DX            ; citire registru de masca
   OR   AL, MASCA1        ; setează bitul corespunzător întreruperii utilizate
   OUT  DX, AL            ; invalidarea intrării de întrerupere
   STI
            …….
   ;rutina de tratare a întreruperii
RUT_INT         PROC  FAR      ; noua rutina de tratare a întreruperii
   PUSH         r         ; salvarea registrelor utilizate in cadrul rutinei
   STI
            ……..
; corpul rutinei
            ……..
;sfârşitul rutinei
   MOV  DX, INTA00
   MOV  AL, EOI
   OUT  DX, AL            ; comanda de sfârşit întrerupere
   POP  r                 ; refacere registre salvate
   IRET
RUT_INT         ENDP
```

În limbajul de programare C++ sunt rezervate funcţii de gestiune a întreruperilor:

getvect(INTR) – determină adresa vectorului de întrerupere INTR;

setvect(INTR, handler) – modifică vectorul de întrerupere INTR cu o nouă rutină handler.

Exemplul 6.3: programarea unei întreruperi hardware în limbajul C++:

```
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#define INTR 0X1C    /* întrerupere de ceas */

void interrupt ( *oldhandler);
int count=0;
void interrupt handler(__CPPARGS)
{
/* rutina de deservire a întreruperii */
  oldhandler();
}
int main(void)
{
/* salvează vectorul vechi de întrerupere */
  oldhandler = getvect(INTR);
/* setează un nou vector de întrerupere */
  setvect(INTR, handler);
/* corpul programului principal */
/* restituie vectourl vechi de întrerupere */
  setvect(INTR, oldhandler);
  return 0;
}
```

VI.4 Desfăşurarea lucrării

1. Să se determine resursele rezervate de sistem pentru gestiunea sistemului de întreruperi;

2. Să se identifice în tabela de întreruperi a unui calculator compatibil IBM-PC AT adresele unor rutine de întrerupere care oferă servicii de baza ale sistemului de operare MS-DOS (exemplu: INT 13H – driverul de disc, INT 10H – driverul video, INT 1CH – întrerupere de periodică pentru programe utilizator, etc.);

3. Să se scrie o rutină de întrerupere care se activează la fiecare întrerupere a ceasului de sistem (se va folosi întreruperea 1CH);

4. Să se scrie o rutina de întrerupere care filtrează cererile de scriere pe disc, în sensul că nu se va permite scrierea pe sectorul 0, pista 0, capul 0.

Pentru aceasta se redirectează întreruperea 13H. Parametri de apel ai rutinei corespunzătoare întreruperii 13h se pot determina cu ajutorul utilitarului THelp sau TecHelp;

5. Să se scrie un program de tipărire care lucrează pe întreruperi. Programul va conţine: un modul de iniţializare a întreruperii de imprimantă, o rutină de tratare a întreruperii hardware şi o rutină de întrerupere software care îndeplineşte funcţia de tipărire;

6. Să se scrie un program care permite înlocuirea rutinei standard a tastaturii;

7. Să se scrie un program care permite înlocuirea rutinei standard a mouse-ului.

VI.5 Conţinutul dării de seamă

1. Sarcina şi scopul lucrării;
2. Resursele rezervate de sistem pentru gestiunea sistemului de întreruperi;
3. Tabelul de repartizare a canalelor de întrerupere pentru calculatorul respectiv;
4. Bloc schema algoritmului programului elaborat;
5. Programul elaborat;
6. Concluzii.

SURSE BIBLIOGRAFICE

1. Geber T. şi alţii. *Echipamente periferice.* Bucureşti, Editura ştiinţificş şi enciclopedică. 1981.
2. Ogruţan Petre. *Interfeţe şi echipamente periferice.* Note de curs. Universitatea Transilvania Braşov. 1994.
3. Philips Semiconductors – *80C51 – Based 8-Bit Microcontroller*, Integrated Circuits, Data Handbook, SUA, February 1994.
4. Hennessy, J., Patterson, D. – *Computer Architecture*: *A Quantitative Approach*, Morgan Kaufmann Publishers, 2-nd edition, 1996.
5. ******* – *Computer Review. The Future of Microprocessors*, IEEE Computer Society Press, September, 1997.
6. Florian Mircea Boian, *Sisteme de operare interactive*, ed. Libris, Cluj-Napoca, 1994.
7. Petru Eles, Horia Ciocârlie, *Programarea concurenta în limbaje de nivel înalt*, Editura Stiintifica, Bucuresti, 1991.
8. Bjarne Stroustrup, *The C++ Programming Language (second edition)*, Addison Wesley, 1991.
9. Baruch, Z., *Sisteme de intrare/ieşire, Îndrumător de lucrări de laborator*, Editura U.T.PRES, Cluj-Napoca, 1998.
10. Baruch, Z., *Sisteme de intrare/ieşire ale calculatoarelor*, editura Albastră, Cluj-Napoca, 2000.
11. Rosch, W. L., *Hardware Bible*, Sixth Edition, Que Publishing, 2003.
12. Mueller, S., *Upgrading and Repairing PCs*, 14th Edition, Que, 2003.
13. Serial ATA International Organization, "Serial ATA: A Comparison with Ultra ATA Technology", 2002, http://www.sata-io.org/docs/
14. Feldman, J. M., *Computer Architecture, A Designer's Text Based on a Generic RISC,* McGraw-Hill, 1994.
15. IBM Research, "The Giant Magnetoresistive Head: A Giant Leap for IBM Research", http://www.research.ibm.com/research/gmr.html.
16. PC Tech Guide, "Hard Disks", 2003, http://www.pctechguide.com/04disks.htm.

17. Microsoft Corp., “Plug and Play Parallel Port Devices”, Version 1.0b, 1996.
18. Eugen Lupu. *Sisteme cu microprocesoare. Resurse hardware, prezentare. Programare şi aplicaţii.* Editura Albastra, 2010.
19. John Woram. *The PC Configuration Handbook.* Random House, New York 1990.
20. Scott Mueller. *PC – Depanare şi modernizare.* Editura Teora, Bucureşti 1995.
21. Tanenbaum, A., *Organizarea structurată a calculatoarelor,* ediţia a IV-a, Computer Press AGORA, Tg. Mureş, 1999.
22. D. Nicula, A. Piukovici, R. Gavrus. *Microprocesoare - Îndrumar de laborator*. Universitatea TRANSILVANIA, Brasov, 2000.
23. Lungu, V., *Procesoare INTEL, Programare în limbaj de asamblare,* Ediţia a II-a, Teora, 2007.
24. Lupu, E., Sisteme cu microprocesoare. Resurse hardware. Prezentare, programare şi aplicaţii. Ed. Albastră, Cluj-Napoca, 2003.
25. Muscă, Ghe., Programare în limbaj de asamblare, Ed. Teora, Bucureşti, 1999.
26. Брэй Барри. *Микропроцессоры Intel: 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, Pentium Pro Processor, Pentium 4. Архитектура, программирование и интерфейсы*. БХВ-Петербург, 2005.
27. Дармаван Салихан. *BIOS. Дизассемблирование, модификация, программирование.* БХВ-Петербург,2007.
28. Сергей Петров. *Шины PCI, PCI Express. Архитектура, дизайн, принципы функционирования.* БХВ-Петербург, 2006.
29. Холмогоров В. *Персональный компьютер.* Олма-Пресс. 2006.
30. Авдеев В.А. *Периферийные устройства: интерфейсы, схемотехника, программирование*. ДМК Пресс. 2009.
31. Иванов Ю.И., Югай В.Л. *Микропроцессорные устройства систем управления.* Таганрог. ТРТУ. 2005.
32. Мячев А.А. *Персональные ЭВМ и микроЭВМ. Основы организации.* Справочник. М. Радио и связь. 1991.

33. Майоров В. Г. *Практический курс программирования микропроцессорных систем.* М. Машиностроение. 1989.
34. *Сопряжение датчиков и устройств ввода данных с компьютерами IBM PC*. М. Мир. 1992.
35. Фигурнов В.Э. *IBM PC для пользователя.* М: ИНФРА-М, 1997.
36. http://www.xilinx.com.
37. http://www.altera.com.
38. http://www.microchip.com.
39. http://www.intel.com.
40. http://www.sygnal.com.
41. http://www.atmel.com.

ANEXE

Anexa 1. Setul de instrucţiuni ale limbajului C/C++

Tipuri de date:

- **numerice**, care pot fi **intregi** sau **reale** ;
- **logice**, care au două valori **TRUE (adevarat)** şi **FALSE (fals)** ;
- **sir de caractere**, reprezintă un şir de caractere cuprins între apostrofuri

Variabile:

- variabile de tip **intreg** notate **integer** ;
- variabile de tip **real** notate **real** ;
- variabile de tip **logic** notate **boolean** ;
- variabile de tip **sir** notate **string** ;

Exemplu:

```
integer a, b;
real c;
string b.
```

Operatori aritmetici:

+ (adunare) ; - (scadere) ; * (inmultire) ; / (împărţire)

- div (împărţire întreagă) - operanzii trebuie să fie de tip intreg şi furnizează rezultatul corect dacă ambele valori ale operanzilor sunt naturale.

- mod (rest al împărţirii) - operanzii trebuie să fie de tip intreg şi furnizează rezultatul corect dacă ambele valori ale operanzilor sunt naturale

Operatori relationali:

<(mai mic); >(mai mare); =(egal); <>(diferit); <=(mai mic sau egal); >=(mai mare sau egal)

Operatori logici:

NOT (negare) ; AND (şi) ; OR (sau) ; XOR (sau exclusiv)

Operatii de intrare / iesire:

Operaţia de intrare (**citire)** este **read**

Operaţia de ieşire (**scriere)** este **write**

Exemplu:

```
real a,b,c;          // se declara variabilele a,b,c//
read a,b,c           // se citesc variabilele a,b,c//
```

write a,b,c // se afiseaza valorile variabilelor a,b,c introduse de la tastatură//

Operatii de decizie:

Forma generala:

if expresie logica **then** operatia1 **else** operatia2 **endif**

Mod de executie: se evalueaza expresia logica, daca este **adevarata** se executa **operatia 1**, iar daca este **falsa** se executa **operatia 2**

Exemplu:

```
integer a, b;
read a read b
if a>b then write a else write b
endif
```

Exemplu:

```
real a, b, c, d, rez;
read a, b, c, d
if c+d>0 then rez:=a+b
else
if c+d=0 then rez:=a-b
else
rez:=a*b
endif endif
write rez
```

Structura WHILE DO:

Forma generală. Fie **E** o expresie si **S** o structura.

while E
do S
endwhile

Se evalueaza expresia logica **E**, dacă este adevarată se execută structura **S** apoi se repeta executia pină ce expresia logică devine falsă.

Exemplu:

```
integer n1, n2, s, i;
read n1 read n2
s:=0 i:=1
while i <= n2 do
                        s:=s+n1
                        i:=i+1
```

```
endwhile
write s
```

Structura FOR:

Forma generală. Fie o variabilă ***i*** (variabila de ciclare) şi două valori intregi **a(valoare initială)** şi **b(valoare finală)** şi o structură **S**

for i:=a, b
S
repeat

Exemplu:

```
integer n, s, i;
read n
s:=0
for i:=1, n
s:=s+i
repeat
write s
```

Structura REPEAT UNTIL

Forma generală. Fie o structură **S** şi o expresie logică **E**

repeat
S
until E

Exemplu:

```
integer n, i, s;
read n
i:=1    s:=0
do
s:=s+i
i:=i+1
until  i > n
write s
```

Elementele de bază ale limbajului C++

Un program scris in C++ este alcatuit din una sau mai multe funcţii. Fiecare funcţie are mai multe instrucţiuni în C++ care codifică algoritmul programului. Instrucţiunile unei functii reprezintă corpul

funcţiei şi sunt cuprinse între **{ }**. După fiecare instrucţiune din corpul funcţiei se pune semnul **;**

La începutul fiecărui program se specifică fişierele care conţin funcţiile ce se utilizează în program astfel: **# include <numefisier.h>.**

Exemple:

<stdio.h> - conţine funcţii standard de intrare / ieşire (I/O) (printf, scanf, getch, ...);

<stdlib.h> - conţine funcţii standard precum (abort, exit, rand, atoi, itoa,...);

<string.h> - conţine funcţii pentru prelucrarea şirurilor de caractere (strcpy, strlen, strcmp, strcat,...);

<math.h> - conţine funcţii matematice (cos, sin, pow, fabs, abs, ...);

<graphics.h> - conţine funcţii pentru gestiunea ecranului în mod grafic (initgraph, putpixel, line, outtext, outtextxy , cleardevice, ...);

<conio.h> - conţine funcţii standard de intrare ieşire de la consolă (clrscr, getch, getche, gotoxy, putch, wherex, wherey,...);

<time.h> - conţine funcţii pentru gestiunea orei şi a datei sistemului de calcul (gettime, settime, ctime,...);

<dos.h> - conţine funcţii pentru interacţiunea cu sistemul de operare DOS (setdrive, inport, outport);

După specificarea directivelor trebuie scrisă functia rădăcină care se numeste **main( )** sau **void main( ).** După numele directivelor sau a funcţiilor **nu se pune** semnul **;**

Citiri, scrieri:

- pentru realizarea **citirii** se utilizeaza : **cin>>nume variabila**
- pentru realizarea **scrierii** se utilizeaza: **cout<<nume variabila**

Exemplu:

```
#include<iostream.h>
#include<conio.h>
void main()
{
int L,l,h;
clrscr();                              // şterge ecranul //
cout<<"Lungimea=" ; cin>>L;
cout<<"Latimea="; cin>>l;
cout<<"Inaltimea="; cin>>h;
getch();
```

}

TIPURI DE DATE

Tipuri întregi:

- **int** (tip intreg care ocupă 16 biţi)
- **long** (tip intreg care ocupă 32 de biţi)
- **unsigned int** sau **unsigned long** (valorile datelor sunt fără semn, adică pozitive)
- **char** (tip caracter, aceste date se pun între două apostrofuri ' ')

Tpuri reale:

- **float** (tip real care reţin si numerele zecimale , ocupă 32 biti)
- **double** (tip real care ocupă 64 biti)
- **long double** (tip real care ocupă 80 biti)

Constante:

Pentru a da un nume constantelor se foloseste declaraţia **const** care are forma:

const [tip] nume=valoare;

[tip] - tipul constantei ; **nume** -numele constantei ; **valoare** - valoarea constantei.

Exemplu:

const float a=12.6;

constanta este de tip **float**, poartă denumirea **a**, are valoarea **12,6.**

Operatori aritmetici:

+ (adunare); **-** (scadere); ***** (inmulţire); **/** (impărţire); **%** (restul impărţirii întregi).

Operatori relaţionali:

< (mai mic); <= (mai mic sau egal); > (mai mare); >= (mai mare sau egal).

Operatori de egalitate:

== (egalitate); **!=** (inegalitate).

Operatori de incrementare şi decrementare:

++ (incrementare); -- (decrementare).

Operatorii pot fi : **prefixaţi** (în faţa operandului) situaţie în care variabila este incrementată sau decrementată înainte ca valoarea retinută de ea să intre în calcul;

postfixaţi (după operand) situaţie în care variabila este incrementată sau decrementată după ce valoarea reţinută de ea intră în calcul.

Exemplu:

a++*b++;
++a*++b;

Operatori logici:

! - negare logică; **&&** - ŞI logic; || SAU logic.

Operatori de atribuire:

Este reprezentata prin semnul **=.**

Se mai utilizează **operatori de atribuire combinaţi:**

+= ; -= ; *= ; /= ; %= ; &= ; <<= ; >>=

Exemplu:

a=a+b este echivalent cu **a+=b**;
a=a*b este echivalent cu **a*=b;**

Operatorul condiţional:

Forma generală **e1 ? e2 : e3**

Se evaluează **e1**, dacă este **adevarată** se execută **e2**, dacă este **falsa** se executa **e3.**

Exemplu:

Citirea unuui numar **x** şi tipărirea numarului **|x|** (modulul numarului **x**).

```
#include<iostream.h>
#include<conio.h>
void main()
{
 float x;
 clrscr();
 cout<<"x=" ; cin>>x;
 cout<<"|x|="<<" "<<(x>=0?x:-x);
 getch();
 }
```

Instrucţiunea IF:

Forma generală:

if (*expresie*) *instructiune1* else *instructiune2*

Se evaluează ***expresia***, dacă este **adevarată** se execută ***instructiune1***, dacă este **falsă** se execută ***instructiune2.***

Exemplu:

```
#include<iostream.h>
#include<conio.h>
 void main()
{
int a,b,max;
clrscr();
cout<<"a="; cin>>a;
cout<<"b="; cin>>b;
if (a>b) max=a;
else max=b;
cout<<"numarul mai mare este "<<" "<<max;
getch();
}
```

Instrucţiunea SWITCH:

Forma generală a instrucţiunii:

switch (*expresie*) {
case e1 **: *secventa 1* ; break;**
case e2 **: *secventa 2* ; break;**
.....
case en **: *secventa n* ; break;**
default : *secventa n+1*;
}

Se evaluează ***expresie***, dacă **este egală** cu una din expresiile **e1, e2, ...en** se execută ***secvenţa*** corespunzatoare expresiei **s1, s2, ...sn**, iar dacă **nu este egală** cu una din aceste expresii se execută numai ***secventa n+1.***

Exemplu:

```
#include<iostream.h>
#include<conio.h>
void main()
{
int i;
```

```
clrscr();
cin>>i;
switch (i)
    { case 1: cout<<"Am citit 1";break;
      case 2: cout<<"Am citit 2";break;
      default: cout<<"Am citit altceva";
    }
getch();
}
```

Instrucţiunea WHILE:

Această instrucţiune permite programarea ciclurilor cu test iniţial. Forma generală este:

while (*expresie*)

{..... instructiuni }

Se evalueaza ***expresie***, dacă este **adevarată** se execută **{....instructiuni}** după care se revine la evaluarea expresiei, dacă este **falsă** se trece la instrucţiunea următoare.

Exemplu:

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
 void main()
{
float a,b,c,d,x1,x2,x;
int tasta;
while (tasta!='q')
    {
clrscr();
cout<<"a=" ; cin>>a;
cout<<"b="; cin>>b;
cout<<"c=" ; cin>>c;
d=float( b*b-4*a*c);cout<<"discriminantul ecuatiei este"<<"
"<<sqrt(d)<<endl;
if(d<0) {cout<<"ecuatia nu are solutii reale";}
else
if (d>0)
{ x1=(-b+sqrt(d))/(2*a) ; x2=(-b-sqrt(d))/(2*a);
```

```
 cout<<"x1="<<x1<<endl;cout<<"x2="<<x2<<endl;}
 else
 {x=float(-b/2*a);cout<<"ecuatia are solutie unica
x=x1=x2="<<" "<<x<<endl;}
 cout<<"Pentru continuare apasa o tasta"<<endl;
 cout<<"Pentru iesire apasa tasta q";
 tasta=getch();
    } }
```

Instrucţiunea DO WHILE:

Instrucţiunea permite programarea ciclurilor cu test final.

Forma generală este:

do
{ instructiuni }
while (*expresie*)

Se execută **{ instructiuni }**, se evaluează ***expresie***, dacă este **adevarată** se execută din nou **{instructiuni}**, iar dacă este **falsă** execuţia instrucţiunii **do** se termina.

Exemplu:

```
#include<iostream.h>
#include<conio.h>
void main()
{
long n, tasta, s=0, i=1;
while (tasta!='q')  {
clrscr();
cout<<"n=";cin>>n;
do
   {
     s=s+i; i=i+1;
   }
while(i<=n);
cout<<"Suma primelor n numere naturale este"<<" "<<s<<endl;
cout<<"Pentru a continua apasa o tasta"<<endl<<"Pentru a iesi
din program apasa tasta 'q'";
tasta=getch();   } }
```

Instrucţiunea FOR:

Se utilizează cel mai fracvent pentru programarea ciclurilor cu test iniţial.

Forma generala:

for($e_{INITIALIZARE}$; e_{TEST}; $e_{INCREMENTARE}$) instrucţiune

$e_{INITIALIZARE}$ - se evalueaza o singura data pentru **initializarea** variabilei de ciclare inaintea primului ciclu;

e_{TEST} - este evaluată inaintea fiecărui ciclu pentru **a testa** dacă se execută instrucţiunea subordonată şi reprezintă condiţia de ieşire din ciclu;

$e_{INCREMENTARE}$ - se evaluează la sfirşitul fiecărui ciclu pentru **incrementarea** variabilei de ciclare.

Exemplu:

```
#include<iostream.h>
#include<conio.h>
void main()
{
int i,n,tasta;
long double a,b;
while(tasta !='q') {
clrscr();
cout<<"Introduceti numarul"<<"";cin>>n;
a=b=1;
for(i=2;i<=n;i++)
                    {a*=i;b+=i;}
cout<<"suma="<<b<<endl;cout<<"produsul="<<a<<endl;
cout<<"Pentru iesire apasa tasta q";
tasta=getch();   }
}
```

Instrucţiunea BREAK:

Se utilizează pentru intreruperea necondiţionată a unei secvenţe.

Instrucţiunea CONTINUE:

Se utilizează numai în blocul instrucţiunilor de ciclare pentru a întrerupe execuţia iteraţiei curente.

Instrucţiunea GO TO:

Are ca efect întreruperea secvenţei curente şi continuarea execuţiei de la instrucţiunea care este specificată după **go to**.

Tablouri de date structurate:

Tabloul este o listă de elemente de acelaşi tip plasate succesiv intr-o zona de memorie.

Tablourile por fii : **simple (vector)** sau **multiple (matrice)**

Exemplu:

```
int a[5]={-2,4,8,1,9};
int b[3][4]={ {11,12,13,14}, {21,22,23,24}, {31,32,33,34} };
```

Şiruri de caractere:

Pentru a **declara un şir de caractere** se utilizează funcţia:

char nume shir[numărul de elemente în şir]

Exemplu: **char shir1[100]** – s-a declarat şirul cu numele **shir1** care poate lista **100 caractere.**

```
char s1[20],s2[20];
cin.get(s1,20);
cin.get();
cin.get(s2,20);
cout<<s1<<endl<<s2;
```

Dacă ar lipsi funcţia **cin.get()** a doua citire nu ar putea fi efectuată, deoarece la apăsarea tastei **Enter** în memorie este păstrat caracterul **'\n'**, fapt care duce la intreruperea citirii.

Exemplu:

```
#include<iostream.h>
#include<conio.h>
void main() {
char s1[20],s2[20];
cin.get(s1,20); cin.get(); cin.get(s2,20);
clrscr();
cout<<s1<<endl<<s2;
getch(); }
```

Funcţii de acces direct la porturile de intrare / ieşire INPORT / OUTPORT:

Inport (InpW)/ Inportb (Inp) - citeşte un cuvânt/octet de la un port;

Outport (OutpW)/ Outportb (Outp) - scrie un cuvint/octet la un port;

Declaraţia funcţiilor :

unsigned inport (*unsigned portid*);

unsigned char inportb (*unsigned portid*);

void outport (*unsigned portid, unsigned value*);
void outportb (*unsigned portid, unsigned char value*);

Semnificaţie:

portid - adresa portului;

value - cuvântul / octetul citit / scris.

Citirea unui cuvint se efectueaza astfel: octetul inferior de la adresa portului indicat, iar cel superior de la adresa potrului indicat + 1.

Scrierea unui cuvint se efectueaza astfel: octetul inferior al cuvintului la adrresa portului indicat, iar cel superior la adresa portului indicat + 1.(Portul de iesire de 16 biti).

Pentru funtcţiile **inport, inportb, outpotr** şi **outportb** se declară biblioteca #include <dos.h>.

Pentru funtcţiile **inp, inpw, outp** şi **outpw** se declară biblioteca #include <conio.h>.

Exemplu:

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>
void main(void)
{
  int portA = 0x378;
  int portB = 0x379;
  int portC = 0x37A;
  int A, B, C;
  int value = 'C';
  outport (portA, value);
  outport (portB, 0x10);
  outportb (portC, 36);
  outportb (portA, 0xAB);
  A = inport (portA);
  B = inport (portB);
  C = inportb (portC);
  }
```

Funcţii de grafică:

Getpixel(int x, int y); - determină culoarea pixelului cu coordonatele **x** şi **y**.

Putpixel(int x, int y, int color); - afişează un pixel pe ecran cu coordonatele **x, y** cu culoarea **color.**
Line(int x1, int y1, int x2, int x2); - desenează o linie din punctul **x1, y1** în punctul **x2, y2** cu culoarea predefinită**.**
Lineto(x, y); - desenează o linie din punctul curent în punctul cu coordonata **x, y** cu culoarea predefinită.
Outtextxy(x, y, msg); - afişează textul **msg** începând cu coordonata **x, y**.

Exemplu:

```
#include <graphics.h>  /* include biblioteca grafică */
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
        int main(void)
        {
  int gdriver = DETECT, gmode, errorcode;
  int xmax, ymax;
  initgraph(&gdriver, &gmode, "");
  errorcode = graphresult();
  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key to halt:");
    getch();
    exit(1);
  }
  setcolor(getmaxcolor());
  xmax = getmaxx();
  ymax = getmaxy();
  int color = getpixel(xmax, ymax);
  putpixel(xmax, ymax);
  line(0, 0, xmax, ymax);
  lineto(xmax, ymax);
  getch();
  closegraph();
  return 0;
}
```

Anexa 2. Setul de instrucţiuni ale limbajului de asamblare

Metode de acces la date:

Metoda de acces la date şi instrucţiuni reprezintă modalitatea de specificare (exprimare) a locaţiei instrucţiunii sau a operandului care urmează să se acceseze.

- **Metode de acces pentru instrucţiuni:**
  - **secvenţială**: se citeşte instrucţiunea de la adresa următoare
    - PC=PC+lungimea_instrucţiunii_curente
    - ex: instrucţiuni aritmetice si logice
  - **directă**: se citeşte instrucţiunea de la adresa specificată în instrucţiunea curentă
    - PC= adresa_directa
    - ex: instrucţiuni de salt, apel de rutine
  - **relativă**: se citeşte instrucţiunea de la o adresă relativă faţă de adresa curentă (deplasamentul poate fi pozitiv sau negativ)
    - PC= PC+deplasament
    - ex: instrucţiuni de salt, apel de rutine
  - **indirectă:** instrucţiunea curentă specifică adresa adresei instrucţiunii următoare.

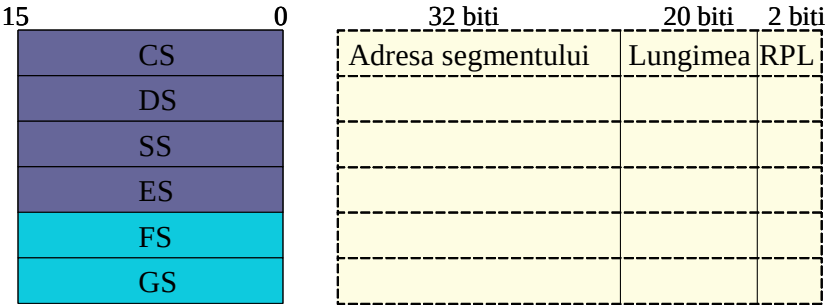
- **Metode de acces la date:**
  - **imediată:** valoarea operandului este prezentă în codul instrucţiunii
    - ex: MOV AX, 1234H
  - **directă:** adresa operandului este specificată în codul instrucţiunii
    - ex: MOV AX, [1234H]
  - **registru:** operandul se află într-un registru intern al UCP
    - ex: MOV AX, BX
  - **indirectă**: instrucţiunea conţine adresa adresei operandului
    - ex: MOV AX, [SI+1234H]
  - **implicită:** codul instrucţiunii implică utilizarea unui anumit registru, adresă de memorie sau adresă de adresă de memorie
    - ex: PUSH AX; implicit se foloseste SP ca registru de adresare

– **indexată**: se accesează succesiv elementele unei structuri de date de tip tablou
  - ex: MOV AX, [SI]
– **altele:** bazată, relativă, stivă.

Modelul ISA x86 de Registre:

- Registre generale:

- Registre segment:

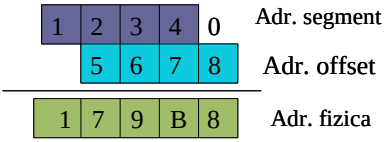


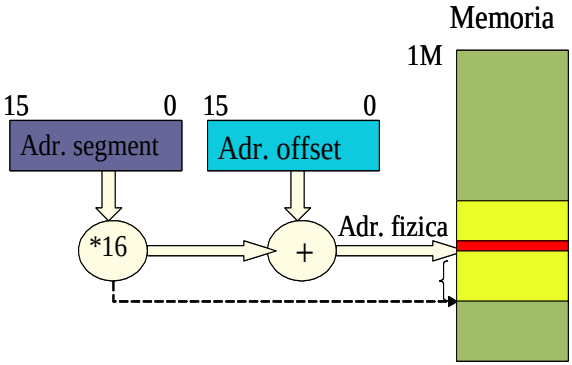
- Registre speciale:
  IP – registru indicator de program;
  SP – registru indicator de stiva;

PSW – cuvântul de stare, se generează după execuţia fiecărei instrucţiuni.

PSW de bază		**PSW extins**	
flags	**destinaţie**	**flags**	**destinaţie**
CF	transport	IOPL	nivel de privilegiu I/E
PF	paritate	NT	task imbricat
AF	transport auxiliar	RF	continuare în caz eroare
ZF	rezultat zero	VM	mod virtual 8086
SF	semn	AC	verificare aliniere
TF	trasare	VIF	intrerupere virtuala
IF	validare intreruperi mascabile	VIP	validare intrerupere virtuala
DF	directia de transfer	ID	indicator
OF	depasire capacitate		

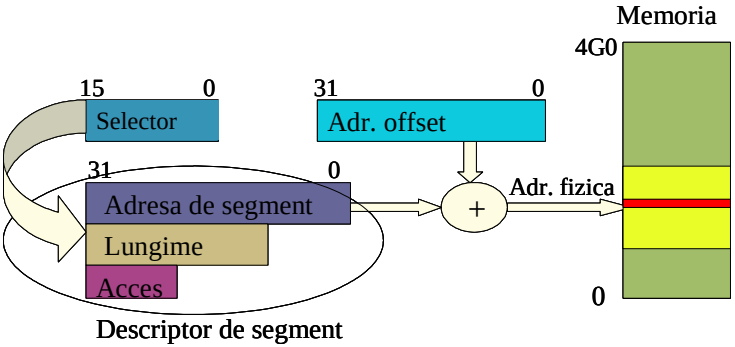
Adresarea memoriei
modul real:





Observatii:
- lungime segment = 64Ko
- spaţiu de memorie = 1Mo
- nu exista mecanisme de protecţie a datelor

modul protejat:



Observaţii:
- spatiu de adresare = 4 Go
- dimensiune segment = 1o .. 4Go
- o mai bună protecţie a segmentelor
- o utilizare mai eficientă a memoriei

Caracteristici ale modurilor de lucru:
- **Modul real** – modul de lucru specific 8086:
  – organizare pe 16 biti
  – spatiu maxim de memorie – 1 Mo
  – resursele hardware sunt direct accesibile programului utilizator
- **Modul protejat**: modul natural de lucru al procesoarelor 386 şi superioare
  – toate extensiile (instrucţiuni, registre, etc.) sunt accesibile
  – asigura un management mai bun al memoriei
  – procesorul lucreaza cel mai eficient si la capacitatea maxima
  – permite simularea în mediu multitasking a modului real => Modul virtual V86 (setat printr-un atribut al taskului)
- **Modul "Administrare sistem"** (SMM – System Management Mod)
  – se intra în acest mod printr-o intrerupere.

Sintaxa instrucţiunilor x86:
- **<linie_program>**:= [<eticheta>:] [<prefix>] [<instrucţiune>|<directiva>] [;<comentariu>]
- **<eticheta>** - şir de litere şi cifre, care începe cu o literă

- **<prefix>** - cuvânt cheie care modifică regimul de execuţie al instrucţiunii care urmează (ex: REP – repetă instrucţiunea următoare de un nr. de ori)
- **<instrucţiune>**:= <mnemonica> [<operand1> [,<operand2>]]
- **<mnemonica>** - grup de litere care simbolizează o instrucţiune
- **<operand1>**:= <registru>|<variabilă>
- **<operand2>**:= <val_imediată>|<operand>
- **<registru>** := EAX|EBX|...AX|BX| .. |AH|BH|.. |AL|BL| ...|CS|DS, ...|GS => nume de registru
- **<variabila>**:= <nume_var>|<adresa_var>
- **<adresa_var>**:=[<nume_var>]'['[reg_index][+<reg_baza>] [+<deplasament>]']'
- **<val_imediata>**:=<număr>|<expresie>
  - un număr sau o expresie aritmetico-logică ce se poate evalua în momentul compilării; se poate exprima în zecimal, hexazecimal (indicativul H) sau binar (indicativul B)
- **<deplasament>** := <val_pe_16biţi>|<val_pe_32biţi>
  - valoare exprimabilă pe 16 sau 32 biţi
- **<comentariu>** - text (cu caracter explicativ) ignorat de compilator

Exemple:

- instrucţiuni fără operand

```
NOP
MOVSB
```

- instrucţiuni cu un operand

```
PUSH AX
ROR  DX
```

- instrucţiuni cu doi operanzi

```
MOV AX, BX
```

- linie cu etichetă instrucţiune şi comentariu

```
START:    MOV AX, BX   ;mută conţ. AX în BX
```

- linie de comentariu

```
; aceasta este o linie de comentariu
```

- linie cu etichetă

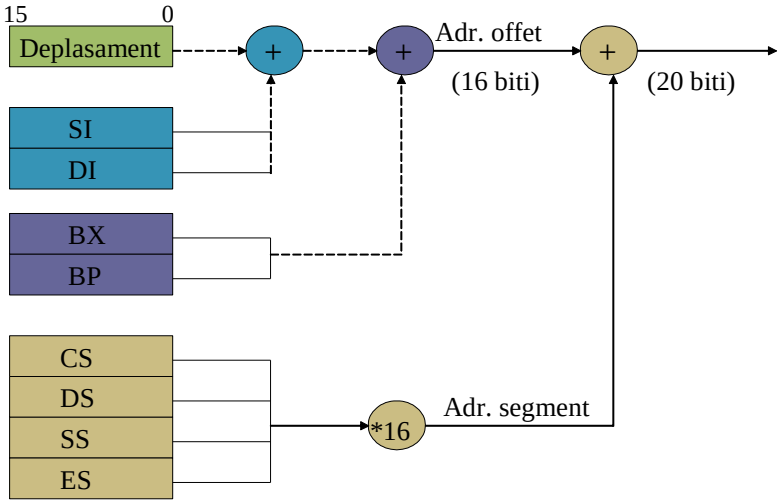
```
ETICHETA:
```

- **Registre interne:**
  - registre generale:
    - (8 biti) AH,AL,BH,BL,CH,CL,DH,DL

- (16 biti) AX, BX,CX,DX, SI,,DI,SP, BP
- (32 biti) EAX, EBX,ECX,EDX, ESI,EDI,ESP, EBP

– registre speciale: CS,DS, SS, ES, FS,GS, GDTR, LDTR, CR0,..CR4, PSW

Calculul adresei unei variabile (în modul real)



- **Porturi de Intrare / Ieşire:**
  - registre continute în interfetele de intrare/iesire
  - spatiul de adresare maxim: 64Ko (adr. maxima 0FFFFH)
  - la PC-uri spatiul este limitat la 1ko (adr. maxima 3FFH)
  - pe aceeasi adresa pot fi 2 registre:
    - un reg. de intrare şi unul de iesire
  - porturile apar doar în instrucţiunile IN şi OUT
  - specificare:
    - direct, prin adresa fizica (daca adresa este exprimabila pe un octet) sau nume simbolic

```
ex:   IN     AL, 12h
      OUT    33h, AL
```

    - indirect, prin adresa continuta în registrul DX

```
ex:   IN     AL,DX
      OUT    DX,AL
```

Clasificarea instrucţiunilor:

Există mai multe criterii de clasificare a instructiunilor:

- dupa tipul de procesor:
  - 8086 (modul real), '386 (modul protejat), x87 (instructiuni flotante), Pentium II (MMX)
- dupa tipul operanzilor:
  - 8/16 biti, 32 biti
- dupa complexitate:
  - simple (o singura operatie, mod simplu de adresare, lungime scurta, timp redus de executie)
- dupa tipul operaţiei efectuate:
  - Instructiuni de transfer
    - mov, lea, les , push, pop, pushf, popf
  - Instructiuni de conversie
    - cbw, cwd, xlat
  - Instructiuni aritmetice
    - add, inc, sub, dec, cmp, neg, mul, imul, div, idiv
  - Instructiuni logice, de rotaţie, deplasare (shift) şi pe bit
    - and, or, xor, not, shl, shr, rcl, rcr
  - Instructiuni de Intrare/Iesire (I/O)
    - in, out
  - Instructiuni pe şiruri
    - movs, stos, lods
  - Instructiuni de control al fluxului de program
    - jmp, call, ret, salturi conditionate

Instructiuni de transfer:

Instructiunea MOV

- Transferă o dată pe 8, 16 sau 32 biţi: între două registre, între un registru şi o locaţie de memorie sau o dată imediată într-un registru sau locaţie de memorie

Sintaxa: mov reg, reg

Exemple:

```
mov mem, reg
mov reg, mem
mov mem, data_imediata
mov reg, data_imediata
```

mov reg_segm, mem16
mov reg_segm, reg16
mov mem16,reg_segm
mov reg16, reg_segm
obs: mov mem,mem – eroare
mov reg_segm, data_imediata - eroare

Instrucţiunile LDS, LES, LFS, LGS, şi LSS

Sintaxa:

LxS <reg16>, mem32

unde x Є [D, E, F, G, S]

- Semnificaţia: - încărcare pointer "far" (adresa variabilei)

< reg16 > <= [mem32]16
<reg_segm> <=[mem32+2]16

- Exemple:

```
lds  si, p1      ; ds<=mem(p1+2)    si<=mem(p1)
lss  bp, p2      ; ss<=mem(p2+2)    bp<=mem(p2)
```

Instructiunea LEA

- Sintaxa:

lea reg16, mem
lea reg32, mem ; doar pt. '386 si mai noi

- Semnificatia: incarca in registru "adresa efectiva" a variabilei de memorie; incarcare pointer pe 16 sau 32 biti

<reg16/32> = adr_offset(mem)

- exemple:

```
lea ax, var1                ; ax=offset(var1)
lea bx, [bx+5]              ; bx=bx+5
lea  di, var[bx+si]         ; di=bx+si+offset(var)
lea  si, -100[di]           ; si=di-100
```

Instructiunile PUSH si POP

- sintaxa:

push <operand>
pop <operand>
<operand> - reg16, reg32, reg_segm, data_imediata, mem

- semnificatia: scriere/citire pe/de pe stiva

sp=sp – 2 sau 4 <operand>=ss:[sp]
ss:[sp]=<operand> sp=sp + 2 sau 4

PUSHAD/POPAD introd/extrage în/din stivă EAX, ECX, EDX, EBX, ESP, EBP, ESI şi EDI decrementând/incrementând the stack pointer cu 32.

- Exemple:

```
push bx
PUSH ES
PUSH  [BX]
PUSH [BP+SI]
PUSH ES :[SI+4]
```

Instructiunile XCHG, LAHF, SAHF, BSWAP

- XCHG - schimba conţinutul operanzilor intre ei
  sintaxa: xchg <operand1>, <operand2>
  <operand1/2> - reg16, reg32, mem16, mem32
- LAHF SAHF
  ah=<flags> <flags>=ah
- BSWAP (byte swap - numai pt. >'486)
  - Schimbă ordinea celor 4 octeţi dintr-un registru de 32 de biţi

  sintaxa: bswap <reg32>
  ex: bsawp eax ; byte0-7⇔byte24-31 si byte8-15⇔byte16-23

Instructiuni de conversie MOVZX, MOVSX, CBW, CWD, CWDE, and CDQ

- MOVZX <dest>, <sursa>
  - mută <sursa> în <dest> cu extensie 0
  - lungime(<dest>)=2*lungime(<sursa>)
  - <sursa> - reg8, reg16, mem8, mem16
  - <dest> - reg16, reg 32
- MOVSX <dest>, <sursa>
  - muta <sursa> in <dest> cu extensie de semn
- CBW - converteşte **al** (byte) în **ax** (word) cu extensie de semn
- CWD - converteşte **ax** (word) în **dx:ax** (dword) cu extensie de semn
- CWDE – converteşte **ax** (word) în **eax** (dword) cu extensie de semn
- CDQ – converteşte **eax** (dword) în **edx:eax** (qword) cu extensie de semn

Instructiuni aritmetice ADD, ADC, SUB, SBB

add <dest>, <src> ;<dest> := <dest> + <src>

adc <dest>, <src> ;<dest> := <dest> + <src> + C
SUB <dest>, <src> ; <dest> := <dest> - <src>
sbb <dest>, <src> ; <dest> := <dest> - <src> - C

- <dest> - reg8/16/32, mem8/16/32,
- <src> - reg8/16/32, mem8/16/32,, data_imediata
- operaţiile aritmetice şi logice afectuează următorii indicatori de condiţie: C, AC, Z, S, O, P
- eventuala depaşire a capacităţii se verifică de programator (atenţie la forma de reprezentare cu/fara semn)
  - C – depaşire la operaţiile fără semn
  - O – depaşire la operaţiile cu semn

Instructiuni aritmetice MUL, IMUL

- mul – inmulţire nr. intregi fără semn
- imul – inmultire nr. intregi cu semn

```
mul src                     ; acc := accLO * src
imul src                    ; acc := accLO * src
imul dest, src1, imm_src    ; dest := src1 * imm_src
imul dest, imm_src          ; dest := dest * imm_src
imul dest, src              ; dest := dest * src
```

- src – reg8/16/32, mem8/16/32
- acc – ax, dx:ax , edx:eax (dim(src)*2)
- dest – reg16/32
- src1 –reg 16/32, mem16/32
- imm_src – data_imediata8/16/32

Instructiuni aritmetice DIV, IDIV

- div – împărţire nr. intregi fără semn
- idiv – împărţire nr. intregi cu semn

```
div src                     ; accLO := acc / src
                            ; accHI := acc MOD src
idiv src                    ; accLO := acc / src
                            ; accHI := acc MOD src
```

- src – reg8/16/32, mem8/16/32
- acc – ax, dx:ax, edx:eax (2*dim(src))
- nu se pot împărţi 2 operanzi de aceeaşi lungime

Instructiuni aritmetice NEG, INC, DEC

```
neg dest                    ; dest := - dest
inc dest                    ; dest := dest + 1
dec dest                    ; dest := dest – 1
```

- dest – reg16/32, mem16/32
- instrucţiuni scurte şi rapide
- inc şi dec :
  - utile pt. parcurgerea unor şiruri prin incrementarea sau decrementarea adresei;
  - utile pentru contorizare (numarare).

Instructiuni logice AND, OR, XOR, NOT

- operatii logice pe bit
- sintaxa: <operator> <operand1>, <operand2>
- <operand1/2> := <reg8/16/32>|<mem8/16/32>|<val_imed>
- semnificatia: <operand1>=<operand1><operator><operand2>
- exemple:

```
and ax, bx                  mov al,10101111b
or  ax, 33h                 and   al, 00100000b
not  var1                   ;in al va fi  00100000b
xor si,si
```

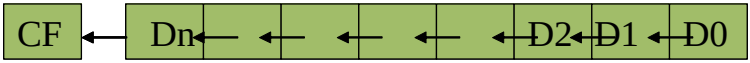
Instructiuni de deplasare SHL/SAL, SHR, SAR

- instrucţiuni pentru deplasarea continutului unui operand la stanga şi la dreapta cu un numar de pozitii binare;
- deplasari:
- logice: deplasare bit cu bit;
- "aritmetice": echivalente cu operatiile de inmultire si impartire.

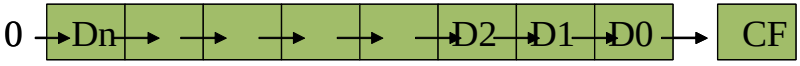
Sintaxa:

<instructiune> <operand>, <contor>
<operand>:=<reg>|<mem>
<contor>:=1| cl |<val_imediata>
(<val_imediata> doar la procesoarele mai noi)

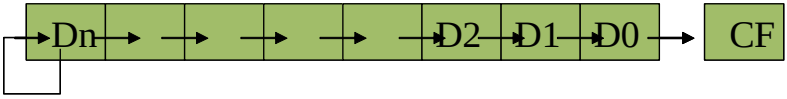
SHL



SHR



SAR



Instructiuni de rotaţie RCR, RCL, ROR, ROL

– instructiuni de rotatie la stanga şi la dreapta, cu si fara CF

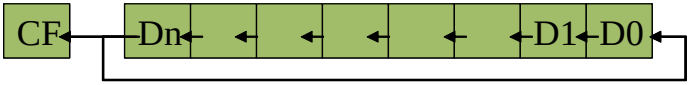
Sintaxa:

<instructiune> <operand>,<contor>

<operand>:=<reg>|<mem> <contor>:= 1| cl| <val_imed.>

RCL

ROR



Instructiuni pe bit

- ***TEST***
  - sintaxa: TEST <operand1>, <operand2>;
  - executa un ŞI logic fără a memora rezultatul;
  - se foloseşte doar pt. pozitionarea indicatorilor de condiţie.
- ***BT, BTC, BTR, BTS***
  - sintaxa: <instr> <operand>, <index>;
  - copiaza bitul specificat de <index> din <operand> în CF
  - BTC complementeaza, BTR reseteaza, iar BTS seteaza bitul dupa copiere;
- ***SETcc***
  - sintaxa SETcc <reg8>|<mem8>
  - seteaza <reg8>|<mem8> dacă o conditia este îndeplinită
  - ex: SETC, SETNC, SETZ, SETNZ, ... SETA, SETB, SETE, SETGE, SETBE, ...

Instructiuni de intrare/iesire IN, OUT

Sintaxa: IN <destinatie>, <port>

OUT <port>, <sursa>

<destinatie>, <sursa> := AL| AX|EAX

<port>:= <valoare8>| DX

Exemplu:

```
IN al, 378H
MOV DX, 378H
```

```
OUT DX, AL
```

Instrucţiuni de control al programului (instrucţiuni de salt):

- ***JMP - salt necondiţionat***
  - sintaxa: JMP <eticheta>|<var_pointer>|<registru>;
- tipul saltului şi distanţa sau adresa de salt se determina de către compilator;
- se preferă utilizarea etichetelor, pt. a marca ţinta saltului;
- salturile necondiţionate nu sunt agreate în programarea structurată.

Exemple:

- salt inainte

```
JMP   ET1
....
ET1:   ....
```

- salt inapoi

```
ET1:   ....
......
JMP   ET1
```

- salt relativ

```
pointer    word  adr_tinta
JMP    pointer
JMP    tabela_de_salt[BX]
JMP    AX
```

- salt intersegment

```
cod1   segment
JMP   ET5
cod1   ends
cod2   segment
.......
ET5:   MOV AX,BX
cod2   ends
```

Instrucţiuni de salt la rutină şi revenire din rutină:

- ***CALL***
  - sintaxa: CALL <eticheta>|<var_pointer>|<registru>
  - aceleaşi tipuri ca şi la instr. JMP, dar fără salt scurt

Apel intrasegment - "near"

- se salvează pe stivă adresa instrucţiunii următoare:

SS:[SP] = IP+<lung_instr_curenta> ; SP=SP-2

- se încarcă numărătorul de instrucţiuni cu adresa rutinei

IP = <adr_offset_rutina>

- se continuă execuţia de la adresa nouă

Apel intersegment - "far"

- se salvează pe stivă CS

SS:[SP] = CS ; SP=SP-2

- se salvează pe stivă adresa instrucţiunii următoare:

SS:[SP] = IP+<lung_instr_curenta> ; SP=SP-2

- se încarcă adresa rutinei în CS:IP

CS:IP = <adr_segment>:<adr_offset_rutina>

- se continuă execuţia de la adresa nouă

Rutina se declară cu directive (proc, endp) sau printr-o simplă etichetă

Revenirea din rutina RET, RETN, RETF

- sintaxa: RET [<deplasament>]
- semnificaţia:
  - RET - revenire din rutina 'near' sau 'far';
  - RETN - revenire din rutina 'near';
  - RETF - revenire din rutina 'far';
  - RET***x*** <deplasament> - revenire cu descărcarea stivei; ***x*** indică numărul de poziţii cu care trebuie să se descarce stiva înainte de revenirea din rutină; în mod uzual acest parametru lipseşte
    - SP=SP+<deplasament> ; pt. descarcarea parametrilor de apel.
- revenirea din rutina:
  - se reface în IP (pt. 'near') sau CS:IP (pt. 'far') adresa de revenire prin descărcarea stivei;
  - se continuă cu instrucţiunea de la adresa refăcută.

Exemple:

```
rut1    proc    near
                push    ax
                ......
                pop     ax
                ret
```

```
rut1    endp
                .....
                call rut1
                mov bx,cx

rut2:   mov  dx,ax
                .......
                ret   2
                ....
                push  param1
                call rut2
                .......
                call  ax
                call  tabela[BX]
```

Instrucţiuni de salt conditionat:

Jcc - salt dacă condiţia ‘cc’ este îndeplinită; în caz contrar se trece la instrucţiunea următoare.

Sintaxa: Jcc <eticheta>

unde:

- <eticheta> - se traduce printr-o distanţă relativă pe 8 biţi
- condiţia este dată de starea unui sau a unor indicatoare de condiţie (flaguri): CF, ZF, SF, PF, OF;
- pentru aceeaşi condiţie pot exista mnemonici diferite (ex: JZ, JE) ;
- Atentie: la 8086/286 salturile pot fi doar în intervalul -128 .. +127;
- de la ‘386 salturile se pot face oriunde în interiorul unui segment.

Salturile condiţionate se pot face în raport de indicatorii de condiţie.

În continuare sunt prezentate variantele de salt condiţionat care implică testarea unui singur indicator de condiţie.

Instruc-ţiunea	**Condiţia**	**Instrucţiuni echivalente**	**Explicaţii**
JC	CF=1	JB, JNAE	salt dacă a fost un transport
JNC	CF=0	JNB, JAE	salt dacă nu a fost un transport

JZ	ZF=1	JE	salt dacă rezultatul este zero
JNZ	ZF=0		salt dacă rezultatul nu este zero
JS	SF=1		salt dacă rezultatul este negativ
JNS	SF=0		salt dacă rezultatul este pozitiv
JO	OF=1	JPE	salt la depăşire de capacitate
JNO	OF=0	JP	salt daca nu este depăşire
JP	PF=1		salt dacă rezultatul este par
JNP	PF=0		salt dacă rezultatul nu este par

Instrucţiuni de salt condiţionat utilizate după compararea a două numere fără semn:

Instruc-ţiunea	**Condiţie**	**Indicatori testaţi**	**Instrucţiuni echivalente**	**Explicaţii**
JA	>	CF=0,ZF=0	JNBE	salt la mai mare
JAE	>=	CF=0	JNB, JNC	salt la mai mare sau egal
JB	<	CF=1	JNAE, JC	salt la mai mic
JBE	<=	CF=1 sau ZF=1	JNA	salt la mai mic sau egal
JE	=	ZF=1	JZ	salt la egal
JNE	!=	ZF=0	JNZ	salt la diferit

Instrucţiuni de salt utilizate după compararea a două numere cu semn (reprezentate în complement faţă de doi).

Instruc-ţiune	**Condiţie**	**Indicatori testaţi**	**Instrucţiuni echivalente**	**Explicaţii**
JG	>	SF=OF sau ZF=0	JNLE	salt la mai mare
JGE	>=	SF=OF	JNL	salt la mai mare sau egal
JL	<	SF!=OF	JNGE	salt la mai mic
JLE	<=	SF!=OF sau ZF=1	JNG	salt la mai mic sau egal

JE	=	ZF=1	JZ	salt la egal
JNE	!=	ZF=0	JNZ	salt la diferit

Exemplu:

Instrucţiunile de salt condiţionat se utilizează după o instrucţiune care modifică bistabilii de condiţie (registrul PSW). Cea mai des întâlnită situaţie este instrucţiunea de comparaţie CMP.

```
MOV   AL, OFFH
MOV   BL, 1
CMP   AL, BL
JA    ET_1
```

Instrucţiunile de ciclare LOOP, LOOPZ, LOOPNZ:

Aceste instrucţiuni permit implementarea unor structuri de control echivalente cu instrucţiunile "for", "while" "do-until" din limbajele de nivel înalt. Aceste instrucţiuni efectuează o secvenţă de operaţii: decrementarea registrului CX folosit pe post de contor, testarea condiţiei de terminare a ciclului şi salt la etichetă (la începutul ciclului) în cazul în care condiţia nu este îndeplinită.

Sintaxa instrucţiunilor:

LOOP <adresă>
LOOPZ <adresă>
LOOPNZ <adresă>

unde: <adresa> este o etichetă sau o expresie evaluabilă la o adresă.

Pentru instrucţiunea LOOP condiţia de terminare a ciclului este CX=0, adică contorul ajunge la 0. Pentru instrucţiunea LOOPZ în plus ciclul se încheie şi în cazul în care ZF=0. La instrucţiunea LOOPNZ ieşirea din buclă se face pentru ZF=1.

Semnificaţie pentru LOOP

CX=CX-1
If(CX!=0) “salt la <etichetă>”
else “continua cu instructiunea urmatoare”.

Sintaxa: LOOPZ|LOOPE <eticheta>

– semnificaţia: asemănător cu LOOP,
CX=CX-1;
if((CX!=0) si (ZF=1) “salt la <eticheta”
else “continua”.

Sintaxa: LOOPZ|LOOPE <eticheta>

– semnificaţia: CX=CX-1
if((CX!=0) si (ZF!=1) “salt la <eticheta”
else “continua”;

- JCXZ, JECXZ -salt daca CX (respectiv ECX) este 0;
- se foloseşte înaintea unei instrucţiuni de buclare (LOOP), pentru a preintâmpina executia de ~65.000 ori a buclei, in cazul in care CX=0;

Exemple:

```
    mov   cx, 100
et1:  .....
      loop  et1     ; ciclul se execută de 100 de ori

            MOV   CX, l_vector
            LEA   SI, vector
            MOV   AL, 0
bucla: ADD  AL, [SI]
            INC   SI
            LOOP  bucla
```

Exemplu cu bucle imbricate:

```
            MOV   CX, numar1
ET2:        PUSH  CX
            MOV   CX, numar 2
ET1:  .....
            LOOP ET1
            POP CX
            LOOP ET2
```

Exemplu cu ieşire forţată din buclă:

```
            MOV   CX, nr_maxim
et4:  .....
            CMP AX, BX
            LOOPNE   et4
```

Instrucţiuni de întrerupere:

Noţiunea de întrerupere presupune (aşa cum îi arată şi numele) întreruperea programului în curs de execuţie şi transferul controlului la o anumită rutină specifică, numită rutină de tratare, dictată de cauza care a generat întreruperea.

Mecanismul prin care se face acest transfer este, în esență, de tip apel de procedură, ceea ce înseamnă revenirea în programul întrerupt, după terminarea rutinei de tratare.

Instrucţiunea INT

Sintaxa: INT <nivel_intrerupere>

UNDE <nivel_intrerupere> = 0 ..255:

- semnificaţia: apelul prin program a unei rutine de intrerupere (întreruperi software);
- adresa rutinei - pastrata intr-o tabela de intreruperi;
- tabela contine 256 intrari (adrese) pt. cele 256 de nivele de intrerupere acceptate de un procesor x86;
- INT - forma speciala de instrucţiune “CALL”;
- mod de executie:
  - se salvează pe stivă registrul de stare program (indicatorii de condiţie);
  - se salvează pe stivă CS şi IP;
  - se copiază în CS:IP adresa rutinei de întrerupere din tabela de întreruperi de la adresa: <nivel_intrerupere>*4;
- adresa rutinei poate fi modificată în timpul execuţiei programului - legare dinamică;
- este o modalitate de acces la resursele (procedurile) sistemului de operare.

Exemple de apeluri de întreruperi:

Apeluri de întreruperi BIOS

```
;citeste un caracter de la tastatura
        MOV   AH,0
        INT     16H
; in Al va fi codul caracterului citit
; scrierea unui caracter pe ecran
        MOV   AH, 0EH
        MOV   AL, ‘X’
        INT     10H
; Apeluri sistem (INT 21H)
; terminarea programului utilizator
        MOV   AX, 4C00H
        INT 21H
```

; Scrierea unui caracter pe ecran

```
MOV   AH, 02
MOV   DL,'X'
INT   21H
```

Instructiuni pentru indicatorii de condiţie:

- ***CLC, STC, CMC***
  - semnificaţia:
    - CLC - clear carry - CF=0;
    - STC - set carry - CF=1;
    - CMC - complement carry - CF=NOT CF;
- ***CLD, STD***
  - semnificatia:
    - controlează direcţia de avans pt. instrucţiunile pe şiruri:
      - DF=0 - incrementare registre index;
      - DF=1 - decrementare registre index;
    - CLD - clear direction - DF=0;
    - STD - set direction - DF=1;
- ***CLI, STI***
  - semnificaţia: permit validarea şi invalidarea întreruperilor mascabile;
  - CLI - clear IF - IF=0 - invalidare întreruperi mascabile;
  - STI - set IF - IF=1 - validare întreruperi mascabile;
- ***NOP*** - no operation - instructiune de temporizare
  - (este de fapt xchg ax,ax care nu are nici un efect);
- ***HLT*** - halt - oprirea procesorului;
- ***HIT*** - halt until interrupt or reset -oprirea temporară a procesorului.

CUPRINS