Adelaida Mihaela DUINEA

INFORMATICĂ APLICATĂ

-notițe de curs -

CUPRINS

CURS 1

1. OBIECTUL INFORMATICII

Informatica cuprinde domeniile legate de proiectarea, construcția, evaluarea, utilizarea şi întreținerea sistemelor de prelucrare automată a datelor, incluzând componentele hardware, software, elementele organizaționale şi umane cu impactul lor în industrie, administrație, comerț, etc.

Rolul informaticii, ca mijloc de perfecționare a societății se concretizează în mai multe direcții:

- asigură o cunoaştere profundă şi o informare operativă asupra stării şi dinamicii relațiilor economice obiective dintre oameni şi procesele de producție;
- constituie un mijloc de modelare şi corelare optimă a factorilor de producție, asigurând creşterea nemijlocită a eficienței economice generale, dar şi pe fiecare întreprinzător în parte.

În procesele industriale complexe, variația rapidă a numeroşilor parametri, precum şi abaterea acestora de la limitele admise, pot fi urmărite şi sesizate operativ, numai cu ajutorul calculatorului electronic.

Principalele avantaje ale unui calculator electronic le constituie **viteza de lucru** a acestuia, dar şi posibilitatea de a se **adapta rapid oricărui domeniu** de utilizare prin executarea unui program corespunzător, eliminând astfel munca repetitivă.

Calculatoarele se bazează în funcționare pe îndeplinirea câtorva sarcini principale:

- ✓ prelucrarea (sau procesarea) informațiilor;
- ✓ stocarea (memorarea) informațiilor;
- ✓ transferul şi comunicarea informațiilor.

Prelucrarea (procesarea) informațiilor

Informațiile cu care operează permanent un calculator pot fi împărțite în 3 categorii: date, programe, parametri de configurare.

Datele – sunt acele informații care sunt procesate.

Programele – reprezintă o categorie specială de informații, care conțin algoritmii conform cărora calculatorul va procesa datele. Programele sunt alcătuite din instrucțiuni care sunt executate una

câte una, până când, pornind de la datele introduse, se ajunge la rezultatul final. Pentru calculator, aceste instrucțiuni sunt codificate în aşa-numitul *cod-maşină* – dat de producător.

Parametri de configurare – este vorba de acele informații care determină modul specific de funcționare pentru fiecare componentă fizică a calculatorului, sau pentru programele folosite de el. Prin aceşti parametri, care rămân memorați de calculator până la modificarea sau ştergerea lor, un calculator poate fi programat, de pildă, să accepte sau să ignore un anumit dispozitiv fizic (un hard-disk, un mouse etc.).

Stocarea (memorarea) informațiilor

Calculatorul poate stoca (memora) informații în mai multe forme diferite, astfel încât el va putea procesa nu numai informații introduse în momentul procesării, ci şi informații stocate în memoria lui. În acest fel, un calculator este adesea folosit şi pentru a găzdui baze de date sau arhive de informații şi documente diverse, în format digital (electronic). Memoria calculatorului se împarte în două tipuri de bază: memorie temporară (pe termen scurt, sau dinamică) şi memorie permanentă (pe termen lung, fixă). Memoria temporară este memoria care se şterge la oprirea calculatorului, şi este folosită numai în timpul funcționării lui, ca o zonă de memorie de lucru pentru programele aflate în funcțiune. Folosind memoria internă RAM (Random Access Memory), calculatorul execută mai rapid programele şi procesează mai eficient informațiile.

Transferul şi comunicarea informațiilor

Pentru a putea stoca şi procesa informații, calculatorul trebuie să le şi transfere de la un dispozitiv la altul, şi adesea în funcțiile sale de bază intră şi comunicarea informațiilor către/dinspre alte calculatoare. Există mai multe forme de transfer şi comunicare de informații în activitatea calculatorului:

I/O (Input/Output) este denumirea generică dată dispozitivelor de intrare/ieşire, adică acelor dispozitive care asigură introducerea (intrarea) informațiilor în calculator, şi afişarea (ieşirea) de informații prin diverse metode. De pildă, tastatura, mouse-ul sau scanner-ul sunt dispozitive tipice de intrare, prin care operatorul poate introduce texte sau poate da comenzi calculatorului, în vreme ce monitorul, imprimanta şi boxele audio sunt dispozitive tipice de ieşire, prin care informațiile din calculator ajung să fie văzute sau auzite de operator.

Transfer în/din memoria internă RAM – orice program, la lansarea sa, este transferat parțial în memoria RAM, de unde va fi executat pas cu pas. Tot în memoria RAM sunt plasate informațiile în curs de prelucrare, şi are loc un transfer continuu de informații între memoria RAM şi celelalte dispozitive din calculator.

Transfer între discuri – citirea informațiilor de pe un spațiu de stocare (disc) oarecare poate fi văzută tot ca o operație de intrare în procesul de prelucrare a informațiilor, iar scrierea informațiilor pe disc poate fi văzută şi ca o operație de ieşire în acelaşi proces.
Comunicația în rețea – pentru un calculator conectat la o rețea, fie prin dispozitive de rețea, fie prin modem, au loc şi transferuri de informații către/dinspre alte calculatoare.

2. METODE DE REPREZENTARE A INFORMAȚIEI

Sarcinile calculatorului implică operarea acestuia cu informații de cele mai diverse tipuri. Se pot distinge două metode de reprezentare a informației: analogică şi digitală.

Informația analogică este de tip continuu, şi este acea informație care poate avea un număr infinit de valori într-un domeniu definit.

Informația digitală are un număr finit de valori într-un domeniu limitat, şi calculatoarele folosesc acest tip de informație pentru ca toate operațiile lor să se deruleze în timp finit şi după algoritmi exacți.
Calculatoarele actuale folosesc o formă particulară de informație digitală şi anume **informația binară.**

Informația binară este informația digitală reprezentată prin folosirea unui set de numai două valori: 0 şi 1. Prin codificări adecvate, aproape orice tip de informație poate fi reprezentată în formă binară. Avantajele acestei forme de reprezentare a informației sunt mai multe: *simplitate***,** *expandabilitate, claritate şi viteză*.

3. STRUCTURA UNUI CALCULATOR PERSONAL

Un ***sistem electronic de calcul*** este o maşină automată de prelucrare a informației capabilă să execute secvențe complexe de operații cu ajutorul unui program înregistrat în memoria principală. Un sistem electronic de calcul (S.E.C) – denumit în mod curent ***calculator***, reuneşte din punct de vedere *fizic* şi *funcțional* două componente de bază: ***componenta hardware***; ***componenta software***.

3.1. Componenta hardware

Componenta hardware reprezintă ansamblul elementelor fizice, care compun calculatorul electronic: circuite electrice, componente electronice, dispozitive mecanice şi alte elemente materiale ce intră în structura fizică a calculatorului electronic care au rolul de a primii date, de a le

memora, de a le prelucra şi de a le reda într-o formă accesibilă utilizatorului. Hardware-ul este controlat de software în procesul transformării datelor în informații.

Componentele hardware sunt asamblate fizic pentru a îndeplini anumite funcții de bază ale calculatorului:

- funcția de introducere a datelor şi programelor în sistem (funcția de intrare);
- funcția de memorare şi regăsire a datelor;
- funcția de prelucrare a datelor, cunoscută şi sub denumirea de funcția aritmetică şi logică;
- funcția de comandă şi control
- funcția de afişare a mesajelor şi rezultatelor (funcția de ieşire).

3.2. Componenta software

Componenta software cuprinde totalitatea programelor, reprezentând "inteligența calculatorului", prin care se asigură funcționarea şi exploatarea sistemului de calcul. Prin intermediul acestor programe, utilizatorul are posibilitatea de a comunica cu sistemul de calcul, introducând date, programe şi comenzi, primind rezultatele prelucrării şi diverse mesaje. O parte din date, rezultate sau programe pot fi memorate pentru prelucrări ulterioare.

Componentele hardware ce formează arhitectura unui sistem electronic de date sunt:

- unitățile de intrare-ieşire (periferice)
- unitatea centrală care cuprinde:
  - unitatea de comandă-control;
  - unitatea de memorare;
  - unitatea aritmetico-logică

3.3. Arhitectura unității centrale

Unitatea centrală se realizează în jurul unui microprocesor (μp) şi îndeplineşte funcția de prelucrare şi stocare a informațiilor.

În alcătuirea unității centrale intra următoarele elemente:

1. microprocesorul;
2. memoria internă;
3. magistrala de date şi magistrala de comenzi

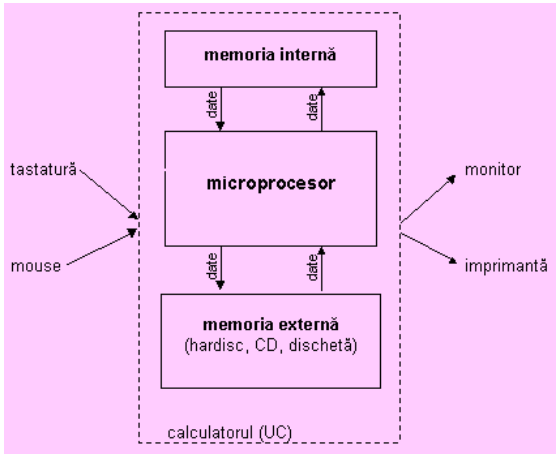


Figura 1. Alcătuirea unui calculator

Microprocesorul este un circuit integrat centralizat prevăzut cu un număr de terminale care permit integrarea lui în circuitul electric. El realizează prelucrarea informațiilor şi în acest scop conlucrează cu memoria internă. El este prevăzut constructiv cu un număr de zone de memorie internă chiar în circuitul respectiv numite *registre*. Fiecare registru în parte are o lungime fixă şi un nume distinct. Unul dintre registrele microprocesorului este *pointerul de instrucțiuni* notat IP. În acesta se stochează adresa din memoria internă în care se găseşte următoarea instrucțiune care trebuie executată.

Memoria internă RAM (Random Access Memory), este reprezentată de un număr de circuite integrate care pot stoca informațiile sub formă electrică. Din punct de vedere informațional informația stocată este pusă sub formă binară. Informațiile sunt transferate între elementele unui calculator personal folosind magistrala de date şi magistrala de comenzi.

Magistralele sunt circuitele conductoare paralele prin care circulă semnale electrice şi la care se corectează diferitele componente. O magistrală este caracterizată de lățimea magistralei respective.

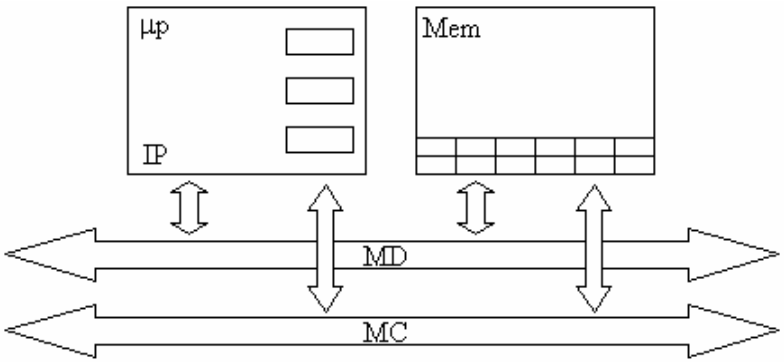


Figura 2.

Memoria internă este împărțită în zone de lungime fixă numite locație de memorie, fiecare locație având câte o adresă.

Pentru a recupera informațiile dintr-o locație, microprocesorul procedează în felul următor:

- microprocesorul pune pe magistralele de date adresa de locație din care să recupereze informații pe care o menține în continuare;
- microprocesorul pune pe marginala de comenzi, semnalul execută citirea;
- controlerul de memorie recunoaşte acest semnal şi recuperează de pe magistrala de date adresa, locația de memorie respectivă;
- controlerul de memorie se duce la locația respectivă şi preia informația stocată în aceasta, în acest timp microprocesorul eliberează magistrala de date;
- controlerul de memorie pune pe magistrala de date informația pe care o preluat-o de la locația de memorie;
- controlerul de memorie pune pe magistrala de comenzi care a fost prealabil eliberată de microprocesor, semnalul execută citirea;
- microprocesorul recunoaşte acest semnal preia informația şi o stochează într-unul din registrele interne.

Memoria externă este formată din discul fix numit HARD DISC, unitatea FLOPPY şi unitatea CD sau DVD. Suporturile de memorie externă servesc pentru memoria permanentă a unui număr mare de informații. Caracteristicile principale ale memoriei externe sunt volumul mare de informație care poate fi memorată şi timpul de acces foarte mare în comparație cu memoria internă.

CURS 2

PERFORMANȚELE MICROPROCESORULUI

Performanțele unui microprocesor sunt influențate de:

1. **Viteza de lucru** – depinde la rândul ei de alte caracteristici ale acestuia. Cele mai importante sunt următoarele:

- ***Frecvența ceasului intern a microprocesorului*** **–** toate fenomenele şi prelucrările de informații realizate în interiorul unui microprocesor nu se execută la întâmplare ci sunt comutate de un generator de impulsuri care funcționează cu frecvență fixă numit ceas intern. Deoarece fiecare operațiune are nevoie de un număr fix de impulsuri, cu cât frecvența generatorului intern este mai mare, cu atât durata fiecărei operațiuni va fi mai mică.

- ***Dimensiunea regiştrilor interni şi lățimea magistralelor calculatorului.*** O mare parte din timpul de lucru microprocesorul îl consumă pentru a schimba informații cu memoria internă. Cu cât regiştrii interni sunt mai mari şi lățimea magistralei este mai mare cu atât microprocesorul va schimba cu memoria internă o cantitate mai mare de informație la un singur transfer. În acest fel se reduce numărul de transferuri efectuate.

- **Modul de lucru al microprocesorului** – microprocesoarele moderne sunt capabile să suprapună parțial anumite informații şi instrucțiuni din program. Primele microprocesoare lucrau secvențial, instrucțiunile fiind încărcate şi executate una după alta. Microprocesoarele moderne sunt capabile să asigure o prelucrare paralelă, totală sau parțială a informației. Prin această operație, două instrucțiuni succesive pot fi parțial suprapuse.

- **Dimensiunea memoriei de prindere** – este o memorie internă sau externă a microprocesorului, realizată într-o tehnologie specială care asigură timp de acces foarte mic. În acest fel, durata operațiilor de transfer între microprocesor şi memorie poate fi redusă destul de mult. Atunci când microprocesorul dispune de memoria de prindere, dacă el trebuie să caute o instrucțiune, o caută mai întâi în memoria de prindere până când o găseşte; dacă nu o găseşte, o caută în memoria internă şi comandă transferul unei pagini de memorie complete din memoria internă în memoria de prindere. Ori de câte ori programele sunt bine structurate, transferurile din memoria internă în memoria de prindere sunt rare şi se obține o creştere a vitezei de lucru. Din contră, atunci când programul are multe instrucțiuni de salt, transferurile între memoria internă şi

memoria de prindere sunt foarte frecvente şi în loc să se obțină o creştere a vitezei, se obține o scădere a acesteia. Se consideră că memoria de prindere poate să conducă la creşterea vitezei microprocesorului cu până la 25 %.

2. **Volumul de memorie pe care îl poate accesa microprocesorul**. Pentru ca microprocesorul să schimbe informații cu anumite locație de memorie el trebui să precizeze care este adresa acestei locații. În acest scop, adresa locației trebuie memorată într-unul din regiştrii microprocesorului. Un alt aspect important care influențează perfecțiunea unui microprocesor este setul de instrucțiuni pe care acesta îl poate executa. Primele microprocesoare erau capabile să execute un număr foarte mare de instrucțiuni. Din analize ulterioare, s-a ajuns la concluzia că doar 15 % din aceste instrucțiuni erau folosite pentru realizarea de programe. În aceste condiții s-au realizat microprocesoare mai simple constructiv, care sunt capabile să execute decât instrucțiunile strict necesare, numit microprocesoare RISC (microprocesoare cu set redus de instrucțiuni.
3. **Memoria internă** – din punct de vedere al comportării, marea majoritate a memoriei interne este o memorie volatilă. Fac excepție de la această regulă componentele memoriei BIOS, care este o memorie de tip ROM (memorie nevolatilă, care poate fi doar citită de utilizator).

Ea conține programul de aplicație şi eventualele constante de program; memoria poate fi de tip PROM (se înscrie o singură dată), EPROM (cu posibilitate de înscriere multiplă, off-line) sau EEPROM (cu posibilitate de scriere în timpul funcționării programului); dimensiunea memoriei variază funcție de varianta constructivă de la 0 la 32ko; ea se poate extinde prin adăugarea unei memorii externe.

Primele calculatoare accesau un volum de 1 MB de memorie internă. Din aceştia, 640 KB erau utilizați pentru programele utilizatorului restul de 384 KB fiind rezervații pentru programe din BIOS (pentru sistemele de operare).

Pentru calculatoarele moderne, primul MB de memorare internă care păstrează împărțirea de la primele calculatoare personale se numeşte *memorie de bază*. Odată cu dezvoltarea programelor, cei 640 KB disponibili pentru programele utilizate au devenit insuficienți. O metodă pentru a creşte memoria adresabilă a constituit-o *memoria expandată*. Calculatoarele moderne, odată cu creşterea dimensiunilor regiştrilor interni, nu mai au probleme cu limitarea memoriei adresabilă. Volumul de memorie internă care depăşeşte primul MB de memorie internă se numeşte *memorie extinsă*. Primi 64 KB din memoria extinsă formează *memoria de nivel înalt*.

INTERFEȚE. PORTURI

- ✓ **Interfețele**

O interfață este un echipament specializat sau un program care face legătura între două componente diferite ale calculatorului. În mod obişnuit o interfață asigură legătura dintre unitatea centrală şi un periferic. Ea are rolul de a prelua datele de pe magistrala unității centrale şi a le pune într-un format care să fie acceptat de periferic. În acelaşi timp preia răspunsurile sau mesajele perifericului şi le transformă în formatul acceptat de unitatea centrală.

- ✓ **Porturile**

Un port reprezintă un punct „x" în care unitatea centrală face schimb de informație cu perifericele externe. După modul de transmitere al informației, porturile pot fi seriale sau paralele. În cazul porturilor seriale se transmite un singur bit odată, pentru aceasta folosindu-se două fire. Biții unui cuvânt sunt transmişi unul după celălalt. După fiecare octet se verifică de obicei corectitudinea transmisiei.

Transmiterea paralelă constă în utilizarea de cabluri cu mai multe fire pentru transmiterea semnalului de informare (în general 8) la care se adaugă un fir comun pentru masă. În acest fel fiecare din biții unui octet este reprezentat de valoarea firului corespunzător acestui bit în raport cu firul de masă; cei 8 biți sunt transmişi simultan. Deoarece fiecare operație de transmise consumă un număr mare de impulsuri de tact, viteza de transmisie în cazul porturilor seriale e mai mică decât în cazul porturilor paralele. Dezavantajul porturilor parale îl reprezintă costul ridicat.

În mod obişnuit, porturile seriale se reprezintă prin numele COM urmat de o cifră de la 1 la 4, iar porturile paralele prin LPT urmat de o cifră de la 1 la 3. În mod normal, un calculator are două porturi seriale şi unul paralel. Calculatoarele moderne utilizează o soluție unificată de transmitere serială a informației, reprezentată de porturile USB. La un port USB se poate conecta un dispozitiv extern sau un echipament specializat numit HUB, care are o ieşire spre calculator şi 5 ieşiri spre dispozitive externe. La fiecare din aceste ieşiri externe, se poate conecta un dispozitiv propriu-zis sau un alt HUB. În felul acesta, se obține o structură arborescentă, care permite conectarea la acelaşi calculator a unui număr foarte mare de periferice. La interfețe mecanice pentru porturile seriale se folosesc conectorii DB9, iar pentru cele paralele DB25.

REȚELE DE CALCULATOARE. COMPONENTE SOFTWARE

Primele calculatoare de tip PC, funcționau independent, acest lucru însemnând un consum mare de memorie externă pentru memorarea programelor şi un număr mare de periferice, câte unul pentru fiecare calculator. După extinderea utilizării acestora, a apărut necesitatea interconectării lor, având ca scop inițial posibilitatea de a partaja anumite resurse ale sistemului (memorie externă, diferite programe, echipamente periferice) între diferiți utilizatori. În plus, conectarea calculatoarelor în rețea permite schimbul direct de informații între utilizatorii ale căror calculatoare sunt conectate în rețea.

Din punct de vedere al extinderii geografice, rețelele pot fi:

a. rețele locale (LAN);
b. rețele extinse (WAN).

Rețelele locale conțin până la câteva sute de calculatoare conectate între ele prin rețele dedicate (nu mai sunt folosite în alt scop) din cabluri electrice sau fibră optică. Distanța între stațiile rețelei este relativ mică, stațiile aparținând, de obicei, aceluiaşi proprietar. Comunicarea între stații se face, în general, în banda de bază.

Rețelele extinse interconectează un număr foarte mare de calculatoarele, de diferite tipuri, aparținând unor proprietari diferiți şi situate la distanțe foarte mari. Comunicația se face prin rețele folosite şi în alte scopuri: rețele telefonice, radio, satelit. Schimbul de informații presupune modularea undei purtătoare şi utilizarea unui echipament special numit „modem”.

Structuri topologice de rețele de calculatoare

Există mai multe tipuri de structuri de rețele locale, şi anume:

a. *Rețele radiale (stelate)* – în care există un calculator principal iar toate celelalte stații sunt conectate cu acesta. Avantaje: preț de cost redus, simplitate. Dezavantaj: fiabilitate redusă.

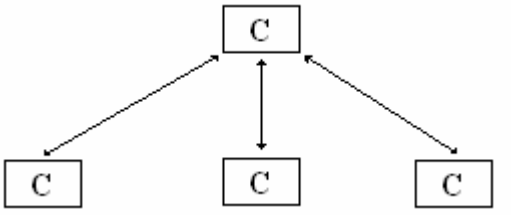


Figura 2.1. Rețea locală tip stelată

b. *Rețele inelare* – toate stațiile alcătuiesc un inel. Avantaj: fiabilitate mai bună. Dezavantaj: transmisie mai lentă a informației, necesită o supraveghere permanentă.

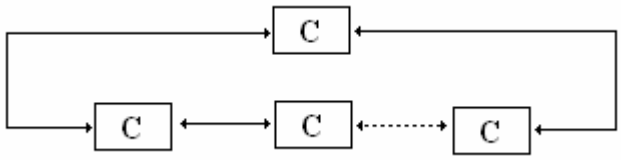


Figura 2.2. Rețea locală tip inelară

c. *Rețele de tip magistrale* – la care toate calculatoarele sunt conectate în paralel pe acelaşi mediu de transmisie. Avantaj: construcție simplă. Dezavantaj: comunicare extrem de dificilă.

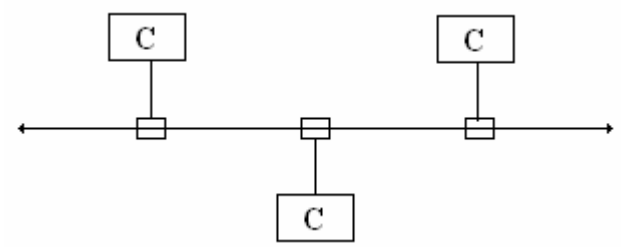


Figura 2.3. Rețea locală tip magistrală

d. *Rețele multinivel* – atunci când există un număr foarte mare de stații putem avea conexiuni diferite (la fiecare nivel se foloseşte un alt tip de structură):

- conexiune radială la nivel înalt;
- conexiune inelară la nivele inferioare.

Componente software pentru rețele de calculatoare

Principalele componente software, care permit interconectarea calculatoarelor de tip PC în rețele, sunt protocoalele de comunicație, driverele de dispozitiv şi software-ul de comunicații.

Protocolul de rețea – totalitatea regulilor care trebuie respectate de stațiile emițător-receptor astfel încât transmisia datelor să poată fi făcută cu garantarea recepției corecte a informației transmise. Aceste protocoale de rețea sunt standardizate la nivel internațional prin reglementări de I.E.E.E., astfel încât să devină posibilă interconectarea unor echipamente provenind de la diferiți producători, realizându-se structuri numite *sisteme deschise*.

Driverele de dispozitiv – componente speciale reprezentate de structuri care permit conectarea anumitor dispozitive fizice cu unitatea centrală. În esență, ele reprezintă interfețe specializate pentru a interconecta dispozitivele cu unitatea centrală (placa de rețea).

Software-ul de comunicație – programe specializate care asigură interconectarea şi compatibilitatea cu mediul de comunicație pe care se face transmiterea informației.

CURS 3

3.1. REȚELE ETHERNET

Ethernet-ul este cel mai utilizat protocol de rețea pentru rețele locale deoarece are avantaje multiple din punct de vedere funcțional şi structural. Astfel, ca structură topologică foloseşte o rețea de tip magistrală cu transmitere serială (fiabilitate ridicată şi extindere simplă). Informația circulă între stațiile rețelei sub forma unor pachete de date sau frame-uri cu o lungime variabilă între circa 70 şi 1500 de octeți. Structura pachetelor de date este bine stabilită prin reglementări internaționale şi trebuie respectată de toți producătorii de echipamente.

O rețea de tip ETHERNET poate interconecta până la 1024 de calculatoare diferite. Pentru a face posibilă comunicația, fiecare calculator are o adresă care poate fi scrisă în binar pe o lungime de 6 octeți şi care este unică în lume. Această adresă este alocată de producătorul plăcii de rețea şi nu se repetă niciodată.

Dezavantajul rețelelor ETHERNET constă în posibilitatea ca două stații din rețea să încerce să emită simultan, semnalele acestora suprapunându-se, ceea ce are ca efect obținerea unui semnal fără nici o semnificație logică şi care nu respectă regulile protocolului de rețea. Acest fenomen se numeşte „*coliziune de date*” (emit două stații simultan). Legătura între stații în sistemul ETHERNET se face prin două fire, deci bit după bit.

Limitarea superioară la circa 1500 de octeți a lungimii pachetului de date, are ca scop împiedicarea unei stații de a ocupa permanent rețeaua atunci când are de transmis un volum foarte mare de date.

Limitarea inferioară a fost făcută pentru a permite determinarea coliziunilor de date.

Rețeaua de tip ETHERNET e numită rețea cu acces concurențial: prima stație care găseşte rețeaua liberă poate să emită. Pentru a începe emisia o stație care vrea să transmită „ascultă” o perioadă rețeaua pentru a depista modificări de potențial care arată că o altă stație emite. Dacă în perioada de urmărire a rețelei nu sunt sesizate transmisii de date, stația poate să înceapă să emită. Această metodă nu exclude posibilitatea ca două stații să înceapă să emită simultan (coliziune de date). Pentru a detecta coliziunea de date, fiecare stație mai „ascultă” o perioadă rețeaua şi apoi începe transmisia, comparând datele de pe rețea cu ceea ce a transmis. Atunci când datele corespund, înseamnă că stația respectivă e singura care transmite, dacă sunt diferite înseamnă că mai transmite o stație. La producerea unei coliziuni de date ambele stații încetează emisia, fac

pauze aleatoare, după care reiau operațiunea. Deoarece pauzele sunt aleatoare, probabilitatea să se producă o nouă coliziune este foarte mică. Dacă totuşi se produce iar, stația îşi încetează emisia, dublează pauza de aşteptare şi repetă operațiunea. Intervalul de ascultare este limitat şi în funcție de acesta se stabileşte dimensiunea minimă a pachetului de date şi lungimea maximă a rețelei.

Limitarea inferioară a lungimii pachetului de date a fost aleasă astfel încât timpul de emisie la frecvența de 100 MHz să fie de circa 50 de nanosecunde, ceea ce corespunde unei lungimi a rețelei de circa 100m.

3.2. STRUCTURA UNUI CADRU DE DATE ETHERNET

Un cadru de date include atât informația care trebuie transmisă cât şi informații complementare care permit sincronizarea celor două stații şi verificarea corectitudinii transmisiei. Astfel, fiecare cadru de date are mai multe componente de lungime fixă sau variabilă:

1. **Preambulul** – care are lungimea de 7 octeți, cu o structură fixă, fiind formați din biți cu valoarea 1 şi 0 alternativ. Această structură are ca scop să permită sincronizarea stației emițător cu cea receptor.
2. **Secvența de început a cadrului** – cu lungimea de 1 octet care respectă aceeaşi structură de 1 şi 0 alternativ, cu excepția ultimilor 2 biți care au valoarea 1. (10101011)
3. **Adresa stației destinație** pe o lungime de 6 octeți. O situație particulară o reprezintă adresa 111....1 care este o adresă ce nu se va regăsi niciodată ca adresă de stație şi care semnifică în rețea faptul că mesajul care urmează este destinat tuturor stațiilor din rețea. Fiecare stație are o adresă proprie care de fapt e adresa plăcii de rețea. Aceasta este alocată de fabricant şi este unică în lume.
4. **Adresa stației sursă** – 6 octeți.
5. **Lungimea** – pe 2 octeți, care arată lungimea pachetului (structurii) de date care urmează să fie transmis.
6. **Pachetul (structura) de date** – cu lungime de 46 – 1500 octeți.
7. **Secvența (structura) de verificare a corectitudinii transmisiei** care are o lungime de 4 octeți.

Anumite fenomene externe pot să modifice valoarea tensiunii pe linie pe anumite perioade de timp, alterând datele transmise. Pentru verificarea corectitudinii transmisiei, se folosesc expresii matematice care au ca variabile valorile biților din pachetul de date. Expresia matematică este

cunoscută atât de stația care emite cât şi de cea care recepționează. La emisie, valorile biților sunt înlocuite în expresiile matematice, iar rezultatul este înscris în secvența de verificare. La recepție, datele recuperate sunt folosite pentru a recalcula valoarea expresiei iar rezultatul se compară cu valoarea din secvența de verificare. Dacă cele două valori coincid înseamnă că recepția a fost realizată corect; în caz contrar există o eroare pe lanțul de transmisie şi structura de date a fost alterată. În acest caz, stația care a recepționat emite o cerere de retransmisie a structurii de date către stația care a emis (stația sursă).

3.3. REȚELE BAZATE PE JETON

Spre deosebire de rețelele ETHERNET, rețele bazate pe jeton sunt rețele cu acces determinist: o stație emite doar când i se dă permisiunea să facă acest lucru; după ce a început emisia ştie sigur că nici o altă stație nu va mai începe să emită. Structura rețelei folosită în acest caz e o structură de inel fizic (rețea inelară) sau logic (rețea de tip magistrală).

Între stațiile rețelei circulă în mod continuu, un cadru de date cu o structură dată, „jeton". Dacă stația nu doreşte să emită, transmite jetonul nemodificat spre stația următoare. Dacă doreşte să transmită, modifică jetonul astfel încât stația următoare să nu-l mai recunoască; îl transmite spre stația următoare după care începe să transmită, tot spre stația următoare şi structura de date pe care vrea să o transmită. Fiecare mesaj primit este retransmis spre următoarea stație. Astfel, după un timp, mesajul revine la stația care l-a emis, stație care poate compara corectitudinea transmisiei, după care mesajul este blocat. De fiecare dată când primeşte un mesaj, fiecare stație compară adresa stației destinație cu propria adresă şi dacă îi este destinat, pe lângă retransmitere îl şi rememorează. Adresa unei stații în acest caz nu mai este unică în lume ci este unică în rețea.

Structura unui jeton are în general 3 octeți.

Secvență de start	Secvență de prioritate	Bitul jeton	Bitul de monitor	Secvența de rezervare	Sfârşit
1 octet	3 biți ceea ce asigură posibilitatea de a defini 2^3=8 nivele de prioritate ale stațiilor	1 bit	1 bit obişnuit=0	3 biți	1 octet

Jetonul are în general valoarea 1 dar când transmite are valoarea 0, el aflându-se înaintea mesajului transmis. Jetonul circulă continuu în rețea de la o stație la următoarea. Pentru a transmite în rețea, o stație trebuie să aştepte să primească jetonul cu bitul jeton de valoare 0. Atunci când îl

primeşte, transformă bitul jeton în valoare 1 şi transmite cadrul spre următoarea stație. După aceasta, stația respectivă începe să-şi emită pachetul de date. După un timp, jetonul parcurge rețeaua circular şi revine la stația care l-a modificat. Aceasta, dacă a terminat transmisia, transformă bitul jeton din 1 şi 0, dând dreptul următoarei stații să emită.

Jetonul fiind tot o structură de date, poate fi şi el alterat. Pentru a evita această situație, una din stațiile rețelei este definită ca stație monitor. În funcție de lungimea rețelei ea ştie după cât timp trebuie să treacă prin dreptul ei jetonul. La fiecare trecere porneşte un numărător care măsoară timpul. Dacă după trecerea timpului jetonul nu a revenit, stația monitor emite un nou jeton cu bitul monitor de valoare 1. Prima stație care-l primeşte, transformă bitul monitor din 1 în 0 şi retransmite jetonul. Revenirea la stația monitor a unui jeton cu bitul monitor de valoare 1 înseamnă că stația monitor a rămas singură şi în aceste condiții emite un mesaj de eroare iar rețeaua îşi încetează activitatea.

CURS 4

4.1. SISTEME DE OPERARE

Calculatoarele personale sunt structuri de echipamente cu un număr mare de componente care trebuie să funcționeze interconectat şi corelat între ele. Din această cauză, trebuie să existe anumite componente de nivel logic (programe) care să îndeplinească funcția de gestiune şi coordonare a funcționării componentelor de nivel fizic. Aceste componente de nivel logic alcătuiesc „***sistemul de operare***" al calculatorului. Aceste programe sunt încărcate de pe un suport de memorie externă; la pornirea calculatorului rămân active pe toată durata funcționării acestora.

Pe lângă funcția de gestiune a componentelor calculatorului, programele din sistemul de operare asigură şi interfața dintre utilizator şi calculator. Această interfață are un caracter de uniformitate, în sensul că ea depinde numai de versiunea sistemului de operare şi nu depinde de varianta constructivă a calculatorului, punând la dispoziția utilizatorului un set minimal de comenzi pe care calculatorul le poate executa.

Funcțiile sistemului de operare reprezintă în acelaşi timp o bază pentru celelalte programe ale utilizatorului: utilizatorul poate folosi aceste funcții în programele proprii fără să mai scrie propriile proceduri (de exemple citirea de caractere, afişarea de caractere pe ecran, crearea, deschiderea şi modificarea fişierelor, etc).

Sarcinile sistemului de operare constau în:

- gestiunea microprocesorului;
- gestiunea memoriei interne;
- gestiunea programelor şi a perifericelor.

Introducerea unui program în execuție se poate face în mai multe feluri:

- scrierea numelui programului (completată, eventual, cu calea care indică locul în care acesta este stocat în memoria internă) şi apăsarea tastei „Enter";
- alegerea programului dintr-o listă de programe;
- selectarea unei pictograme asociată programului respectiv.

După aceste operații, programul intră în execuție. Pentru a intra în execuție, programul trebuie adus din memoria externă în memoria internă iar adresa primei instrucțiuni a acestuia trebuie înscrisă în registrul pointer-ului de instrucțiuni. Aceste operații sunt executate de anumite componente ale sistemului de operare. Alte componente realizează transpunerea instrucțiunilor

scrise într-un limbaj de programare evoluat sub o formă care să poată fi înțeleasă şi interpretată de microprocesor (limbaj cod-maşină).

Limbajul fiecărui microprocesor depinde de constructorul acestuia. Anumite componente ale sistemului de operare preiau comanda utilizatorului (care au aceeaşi formă oricare ar fi tipul calculatorului utilizat) şi o transcrie astfel încât să poată fi analizată, interpretată şi executată de microprocesor.

În raport cu funcțiile îndeplinite, un calculator poate fi împărțit în componente de nivel fizic şi componente de nivel logic.

Componentele de nivel logic sunt programele şi datele.

Din programe fac parte componentele sistemului de operare, programele aplicative şi pachetele de programe. Datele pot fi memorate în fişiere şi baze de date.

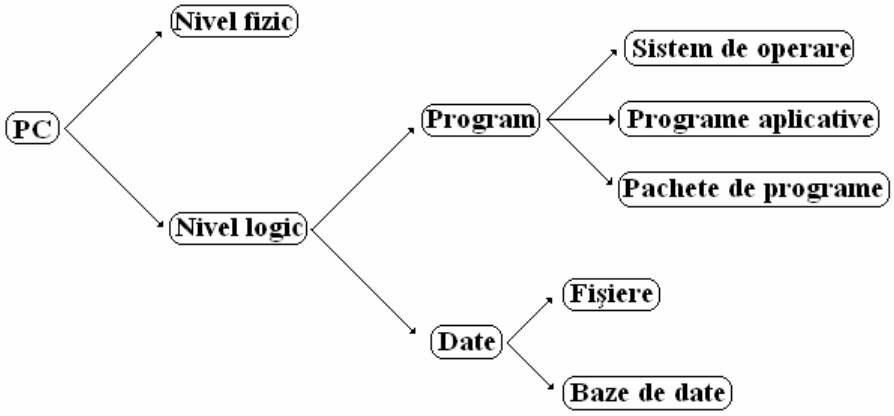


Figura 4.1. Structura unui calculator din punct de vedere al funcțiilor îndeplinite

Ca şi structură, un sistem de operare este în acelaşi timp un pachet de programe: mai multe programe independente care funcționează corelat astfel încât să asigure îndeplinirea unei anumite sarcini.

4.2. FUNCȚIILE ŞI COMPONENTELE SISTEMULUI DE OPERARE

Un sistem de operare are următoarele funcții principale:

1) ***Gestiunea microprocesorului*** – Sistemul de operare trebuie să cunoască numărul de microprocesoare cu care este dotat calculatorul respectiv şi să stabilească în fiecare moment ce sarcină primeşte microprocesorul sau fiecare dintre microprocesoarele sistemului multi-procesor. Atunci când este nevoie, sistemul de operare alocă unul dintre microprocesoare unui anumit program, înscriind o instrucțiune a acestuia în registrul pointer-ului de

instrucțiuni al microprocesorului. Odată cu această alocare, se precizează şi cât timp va putea executa microprocesorul acest program. Există două posibile situații:

a. programul se încheie înainte de consumarea timpului alocat. În acest caz, la terminarea programului, acesta transferă controlul microprocesorului către sistemul de operare;

b. la expirarea timpului alocat, programul nu s-a încheiat. În acest caz, sistemul de operare alocă microprocesorul unui alt program dacă este nevoie, dar înaintea acestei alocări salvează starea programului scos din execuție (memorează instrucțiunea care era executată şi valorile variabilelor programului). În acest fel, atunci când programul va reintra în execuție, el va fi reluat din punctul în care a fost întrerupt.

2) ***Gestiunea memorării*** – Sistemul de operare trebuie să cunoască valoarea şi tipul de memorie de care dispune calculatorul, volumul acesteia şi să realizeze alocarea acestei memorii în funcție de necesități, asigurând şi protecția informației memorate. De exemplu, atunci când un program defineşte o variabilă, acestei variabile i se alocă o zonă de memorie în care va fi stocată. După această alocare, zona va fi rezervată, informația neputând fi modificată decât de către programul în care s-a definit variabila respectivă. Nici un alt program nu va putea modifica informația, chiar dacă şi în cadrul acesteia se defineşte o variabilă cu acelaşi nume. Zona de memorie este rezervată până când programul în care a fost definită variabila se încheie sau acesta renunță la variabila respectivă şi eliberează zona de memorie.

3) ***Gestiunea echipamentelor periferice***. În cadrul acestei funcții sistemul de operare trebuie să cunoască toate perifericele care sunt conectate la calculatorul respectiv. Atunci când un program aplicativ are nevoie să utilizeze un periferic, el transmite o cerere către microprocesor. Acesta, prin intermediul unor componente ale sistemului de operare, analizează aceste cereri şi stabileşte cărui program şi eventual pentru cât timp îi va fi alocat perifericul respectiv.

✓ **Componentele unui sistem de operare**

Fiecare program al sistemului de operare îndeplineşte o anumită funcție. De aceea, ele pot fi împărțite în raport cu anumite criterii:

1. În funcție de poziția pe care o ocupă în raport cu utilizatorul şi cu componentele de nivel fizic:

a. Programe pentru prelucrarea comenzilor de consolă (C.C.P.) – au sarcina de a prelucra comenzile de la utilizator şi a le descompune în sarcini simple. Acestea sunt transmise către

componenta DOS. La nivelul acestei componente, sarcinile sunt descompuse în sarcini elementare care pot fi executate pas cu pas de componentele fizice ale calculatorului. Aceste sarcini elementare sunt preluate de componenta BIOS care este capabilă să le transmită componentelor fizice sub o formă care să fie acceptată de acestea.

b. Componenta DOS. Aceste componente descompun sarcinile simple în sarcini elementare.

c. Componenta BIOS – care ştie cum să interacţioneze cu componentele fizice pentru a executa aceste sarcini elementare. De exemplu, la introducerea unei comenzi de la tastatură, componenta BIOS preia caracterele unul câte unul. Componenta DOS reface comanda ca şir de caractere şi analizează corectitudinea acesteia. Componenta procesorului de comenzi de la consolă analizează comanda primită şi stabileşte care sunt acțiunile pe care trebuie să le facă.

2. În funcție de sarcinile pe care le au de îndeplinit:

a. *Monitorul sistemului* – cuprinde toate programele care au ca scop monitorizarea funcționării diferitelor componente şi eventual starea diferitelor programe;

b. *Programe de comandă şi control* – permit utilizatorului să-şi realizeze propriile aplicații;

c. *Programe speciale de intrare-ieşire*. Aceste programe au ca scop controlul funcționării echipamentelor speciale pentru transmiterea de date de la utilizator spre unitatea centrală sau invers, a rezultatelor de la unitatea centrală spre utilizator.

d. *Programe de lucru care îndeplinesc alte funcții specifice*. În această categorie intră:

- program de gestiune a fişierelor (bibliotecar) care are ca scop operațiile cu fişiere pe suporturile de memorie externă;
- editorul de legături;
- încărcătorul – program din sistemul de operare capabil să regăsească programe executabile, să le încarce în memorie internă şi să le lanseze în execuție.

CURS 5

5. REPREZENTAREA INTERNĂ A DATELOR

Caracteristicile tehnice şi funcționale ale sistemelor de calcul a făcut necesară adoptarea unui sistem specific de reprezentare a informațiilor cu caracter numeric sau alfanumeric.

Reprezentarea sub formă grafică a informației cu caracter numeric a generat diferite metode de reprezentare a acestora, numite metode numerice.

Totalitatea acestor metode, împreună cu un set de reguli care ajută la scrierea unui număr folosind un anumit tip de caractere numit *cifre,* se numeşte *sistem de numerație*.

După valoarea echivalentă asociată unei cifre, sistemele de numerație pot fi:

- nepoziționale;
- poziționale.

În cazul ***sistemelor nepoziționale*** valoarea unei cifre este aceeaşi, indiferent de poziția pe care aceasta o ocupă în reprezentarea numărului.

Reprezentarea prin sistemele de numerație nepoziționale, necesită utilizarea unor reguli dificile, greu de implementat şi din această cauză nu este folosită.

Sistemele de numerație poziționale se caracterizează prin faptul că valoarea unei cifre depinde de poziția pe care aceasta o ocupă în transcrierea numărului.

Un sistem de numerație pozițional foloseşte un număr numit *baza* sistemului respectiv. Numărul de simboluri al alfabetului unui sistem de numerație trebuie să fie egal cu baza sistemului de numerație respectiv.

Cunoscând baza unui sistem de numerație, notată cu „b”, un număr oarecare „N” format dintr-o parte întreagă şi o parte fracționară poate fi reprezentat sub forma:

$$N=N_i+N_F \qquad \text{sau } N=\sum_{i=-m}^{n-1} a_i b^{-i} \tag{5.1}$$

unde a_i sunt cifre din alfabetul de numerație respectiv.

Dacă se cunoaşte baza sistemului de numerație, numărul poate fi reprezentat sub forma:

$$N=\sum_{i=m}^{n-1} a_1 b^i \rightarrow \underbrace{a_{n-1}a_{n-2}.....a_1}_{N_i},\underbrace{a_0 a_{-1}...a_{-m}}_{N_F} \tag{5.2}$$

Exemplu:

$$43793{,}21 = 4\cdot10^4 + 3\cdot10^3 + 7\cdot10^2 + 9\cdot10^1 + 3\cdot10^0 + 2\cdot10^{-1} + 1\cdot10^{-2}$$

Având dată baza sistemului de numerație, alfabetul acestuia trebuie să cuprindă atâtea simboluri cât este baza sistemului de numerație.

Exemplu:

pentru baza 2 {0,1} – sistem binar;

pentru baza 8 {0, 1, ... 7} – sistem octal;

pentru baza 16 {0, 1, ..., 9, A, B, C, D, E, F} – sistem hexazecimal.

Reprezentarea unui număr într-un sistem de numerație depinde de sistemul de numerație respectiv. Trecerea dintr-un sistem de numerație pozițional caracterizat prin baza „b_1" într-un sistem de numerație caracterizat de baza „b_2" se face simplu, prin operații aritmetice elementare de înmulțire şi împărțire în baza inițială. Astfel, pentru transformarea părții întregi se face împărțirea la noua bază, iar pentru partea fracționară se face înmulțirea cu noua bază.

Pentru aceasta trebuie ținut cont că valoare numerică a numărului este aceeaşi indiferent de baza în care este reprezentată.

Se consideră numărul

$$N_i = a_{n-1}b_2^i + ... + a_1b_2 + a_0 \tag{5.3}$$

Determinarea cifrelor de la a_0 până la a_{n-1} se face exprimând baza 2 în baza 1 prin împărțiri succesive la baza 2.

i=n-1

$$N = a_{n-1}b_2^{n-1} + a_{n-2}b_2^{n-2} + ... + a_1b_2 + a_0 \tag{5.4}$$

Dacă această parte întreagă se va împărți la valoarea lui b_2 în baza inițială se va obține:

$$\frac{N}{b_2} = \underbrace{a_{n-1}b_2^{n-2} + ... + a_1b_2}_{N} + \underbrace{\frac{a_0}{b_2}}_{=a_0} \tag{5.5}$$

$$\frac{N_1}{b_2} = a_{n-1}b_2^{n-3} + ... + \underbrace{\frac{a_1}{b_2}}_{=a_1} \tag{5.6}$$

Pentru partea fracționară se vor opera înmulțiri succesive cu baza 2:

$$N_F = a_{-1}b_2^{-1} + ... + a_{-m}b_2^{n-1} \tag{5.7}$$

$$N_Fb_2 = a_{-1} + \underbrace{a_{-2}b_2^{-2} + ... + a_{-m}b_2^{n-2}}_{N'_F} \tag{5.8}$$

$$N'_Fb_2 = a_{-2} + ... + a_{-m}b^{n-2} \tag{5.9}$$

Datorită particularităților sistemului de calcul, caracterizat în principal de elemente cu două stări stabile, reprezentarea sistemelor de numerație se face în general folosind sistemul binar.

Exemplu:

$(17,31)_{10}=17+0,31$

17:2=8 rest 1 1

8:2=4 rest 0 0

4:2=2 rest 0 0

2:2=1 rest 0 0

1:2=0 rest 1 1 $\Rightarrow$ $N_i=(10001)_2$

0,31·2=0,62 0

0,62·2=1,24 1

1,24=1+0,24

0,24·2=0,48 0

0,48·2=0,96 0

0,96·2=1,92 1 $\Rightarrow$ $N_F=(01001)_2$

$(17,31)_{10}=(10001,01001)_2$

Reprezentarea unui număr într-un sistem de calcul trebuie să se limiteze la atâtea cifre câte celule au regiştrii interni.

Trebuie subliniat faptul că modul de reprezentare a unui număr într-o bază depinde de valoarea bazei respective. Astfel, un număr poate avea într-o bază un număr finit de zecimale, iar în altă bază un număr infinit:

$$\left(\frac{1}{3}\right)_{10}=0,333.....\left(\frac{1}{3}\right)_{3}=0,1$$

Este cunoscut faptul că fiecare cifră aflată în stânga virgulei reprezintă o valoare între 0 şi 9 înmulțită cu o putere pozitivă a lui 10, în timp ce la dreapta virgulei valorile se înmulțesc cu o putere negativă a lui 10. puterile cresc, respectiv scad, din unu în unu pornind de la virgulă spre stânga, respectiv spre dreapta.

Exemplu: reprezentarea numărului 123,456 este:

$$123,456=1\cdot10^2+2\cdot10^1+3\cdot10^0+4\cdot10^{-1}+5\cdot10^{-2}+6\cdot10^{-3}$$

Pentru reprezentarea internă în calculator se preferă baza 2, deoarece este strâns legată de modul de stocare a informațiilor în memoria internă. Se folosesc, de asemenea, baze multiplu de 2

(8, 16) datorită facilității de schimbare a bazei, care nu mai necesită calcule speciale: (10011, $1010001)_2$.

Reprezentarea unui număr se face ca şi în baza 10 cu deosebirea că de această dată valorile se înmulțesc cu puteri ale lui 2. ca urmare, este uşor de realizat conversia din baza 2 în baza 10.

Exemplu:

$$11001010_2 = 1\cdot 2^7 + 1\cdot 2^6 + 0\cdot 2^5 + 0\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 = 128 + 64 + 8 + 2 = 202\big|_{10}$$

Operația inversă, conversia din b_{10} în b_2 este mai laborioasă, numărul zecimal fiind împărțit succesiv la 2 atât timp cât câtul este supraunitar. Reprezentarea binară este dată de câtul ultimei împărțiri urmat de resturile împărțirilor anterioare în sens invers.

$673 : 2 = 336 rest 1$

$336 : 2 = 168 rest 0$

$168 : 2 = 84 rest 0$

$84 : 2 = 42 rest 0$

$42 : 2 = 21 rest 0$

$21 : 2 = 10 rest 1$

$10 : 2 = 5 rest 0$

$5 : 2 = 2 rest 1$

$2 : 2 = 1 rest 0$

Rezultatul conversiei este:

$673_{10}=1010100001_2$

CURS 6

6.1. REPREZENTAREA NUMERELOR ALGEBRICE

O problemă specială la reprezentarea internă a datelor o reprezintă reprezentarea semnului numărului şi a virgulei. În general, virgula este poziționată înaintea celei mai semnificative cifre a unui număr. Există trei modalități de reprezentare a numerelor algebrice:

- ✓ **reprezentarea prin valoare şi semn** – în acest caz se reprezintă separat valoarea numărului şi separat, printr-un bit special, semnul acestuia. Semnul se reprezintă printr-un bit situat cel mai la stânga, reprezentat în fața celui mai puțin semnificativ bit. Valoarea acestuia este „0" pentru numerele pozitive şi „1" pentru numerele negative.

$(13)_{10} = (1101)_2$

$(13)_{10} = \underline{0}1101$

$(-13)_{10} = \underline{1}1101$

$$N = a_n b^n \sum_{i=m}^{n-1} a_i b^i$$

$a_n = 0 \rightarrow nr \geq 0$

$a_n = 1 \rightarrow nr < 0$

Avantajul reprezentat prin valoare şi semn îl reprezintă simplitatea. Dezavantajul îl constituie faptul că înainte de operațiile aritmetice trebuie analizat bitul de semn.

- ✓ **reprezentarea în complement față de 2** – numerele pozitive sunt reprezentate ca şi cazul anterior prin valoare şi semn. Numerele negative sunt reprezentate prin bitul de semn, inversarea biților numărului respectiv şi adunarea valorii „1" la cel mai puțin semnificativ bit.

$$N = a_n b^n \sum_{i=m}^{n-1} a_i b^i + 1b^{-m}$$

$a_i = 1 \rightarrow a_i = 0$

$a_i = 0 \rightarrow a_i = 1$

$(13)_{10} = (1101)_2 \rightarrow 0010 + 0001 \rightarrow 0011$

$(-13)_{10} = 10011$

- ✓ **reprezentarea în complement față de 1** – este asemănătoare cu reprezentarea în complement față de 2 numai că nu se mai adaugă valoarea „1" la cel mai puțin semnificativ bit.

$$N = a_n b^n \sum_{i=m}^{n-1} a_i b^i$$

$(13)_{10} = 1101 \rightarrow 0010$
$(-13)_{10} = 10010$

În cazul reprezentării în complement nu mai este necesară verificarea semnului operanzilor înaintea efectuării operațiilor aritmetice.

6.2. REPREZENTAREA NUMERELOR ÎN VIRGULĂ MOBILĂ

Numerele reale se reprezintă în calculator în virgulă mobilă. Reprezentarea în virgulă fixă are ca principal dezavantaj faptul că numerele foarte mari nu pot fi reprezentate, iar numerele foarte mici devin egale cu „0”.

Reprezentarea în virgulă mobilă permite reprezentarea unor valori numerice pentru o gamă mult mai extinsă, păstrând precizia relativă. În acest scop se foloseşte proprietatea oricărui număr de a putea fi scris sub formă de „mantisă şi exponent”.

Numărul real R este pus sub forma

$$R = f * b^e$$

unde: f este un număr subunitar (mantisa), b este baza iar e este exponent.

Exponentul reprezintă o valoare întreagă care este egală cu puterea lui „10” şi caracterizează mărimea numărului respectiv.

Mantisa este întotdeauna subunitară şi caracterizează valorile acestuia.

$(157)_{10} = 0{,}157 \cdot 10^3$ M = 0,157 E = 3
$(0{,}00157)_{10} = 0{,}157 \cdot 10^{-2}$ M = 0,157 E = -2

Avantajul acestei reprezentări este că, întotdeauna în mantisă prima cifră după virgulă este nenulă, deci se păstrează în reprezentarea numărului cifrele cele mai semnificative, atât pentru valori mari cât şi pentru valori mici.

În cazul numerelor algebrice trebuie ținut cont de faptul că atât mantisa cât şi exponentul pot să fie negative. O posibilitate de reprezentare a celor două semne ar fi să se rezerve 2 biți, unul pentru semnul mantisei şi unul pentru semnul exponentului. Deoarece în acest fel s-ar pierde doi biți de memorie se preferă reprezentarea clară numai a semnului mantisei, iar exponentul se transformă într-o valoare derivată, numită caracteristică. Mantisa şi caracteristica sunt reprezentate în hexazecimal. După numărul de biți pe care se reprezintă numărul respectiv, avem reprezentare simplă precizie „SP” - se face pe 32 de biți şi reprezentare dublă precizie, „DP”, pe 64 de biți.

Pentru reprezentarea în simplă precizie, „SP”, din cei 32 de biți 24 sunt rezervați pentru reprezentarea mantisei, iar ceilalți 8 sunt folosiți pentru reprezentarea celorlalte informații (semn şi caracteristică):

$E \to 6b$ $\qquad 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 63$

E = -63 – 63

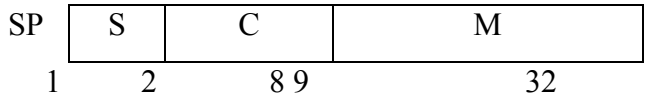
$E \to C = 64 + E$ *- caracteristică*

C = 0 ÷ 127

Pentru numerele negative în cazul mantisei se foloseşte codul complementar, iar pentru exponent codul inversat.

$(31)_{10}$ $(-31)_{10}$

Simplă precizie

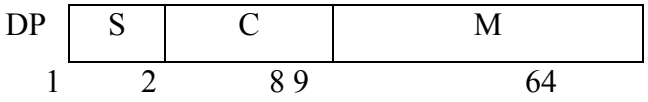


1 bit semn

2 ... 8 caracteristică

9 ... 32 biți pentru mantisă

Dublă precizie



1 bit semn

2 ... 8 caracteristică

9 ... 64 biți pentru mantisă

$(31)_{10} = 16^1 \cdot 1 + 16^0 \cdot 15 = (1F)_{16}$

0 .... 9 A B C D E F

10 11 12 13 14 15

$31 = 0{,}31 \cdot 10^2$ M = 0,31 E = 2

C = 64 + 2 = 66

$(66)_{10} = 16^1 \cdot 4 + 16^0 \cdot 2 = (42)_{16}$

0100 0010

$(1F)_{16}$

01 1111

$$(42)_{16} = \underbrace{0}_{S}\underbrace{1000001}_{C}\underbrace{00001111100...}_{M}$$

Operațiile cu numere reprezentate în virgulă mobilă sunt mai dificile decât operațiile cu numere în virgulă fixă. Pentru adunare se transformă cei doi operanzi astfel încât să aibă aceeaşi caracteristică:

$0{,}5\cdot10^{2} + 0{,}3\cdot10^{3} = 0{,}05\cdot10^{3} + 0{,}3\cdot10^{3} = (0{,}05+0{,}3)\cdot10^{3}$

După această transformare se adună mantisele celor două numere, iar caracteristicile se copiază; scăderea se face la fel.

Pentru înmulțire se adună caracteristicile şi se înmulțesc mantisele, urmând ca ulterior, dacă e cazul, să se modifice cele două valori astfel încât prima cifră după virgulă a mantisei să fie nenulă.

$0{,}1\cdot10^{2} \cdot 0{,}2\cdot10^{4} = 0{,}02\cdot10^{6}$

Pentru împărțire există posibilitatea ca mantisa să devină supraunitară; regula de bază pentru operație este că se scad caracteristicile şi se împart mantisele. Înainte de aceasta, dacă este cazul, se modifică deîmpărțitul astfel încât prin operații de împărțire a mantiselor să se obțină o valoare mai mică decât 1.

$$\frac{0{,}8\cdot10^{3}}{0{,}2\cdot10^{2}} = 4\cdot10^{1} \qquad \frac{0{,}08\cdot10^{4}}{0{,}2\cdot10^{2}} = 0{,}4\cdot10^{2}$$

APLICAȚII

A1. ALGORITM. PREZENTAREA GENERALĂ

Rezolvarea oricărei probleme, se poate realiza într-un număr de etape sau paşi. Cu toate că unii paşi par identici, momentul în care ei se execută, poate fi total diferit.

Metoda prin care se rezolvă o problemă, se numeşte *algoritm*.

Exemplul 1:

Se consideră că există două vase unul A de 3 litri şi unul B de 8 litri. Cum să facem să avem 4 litri de apă în vasul de 8 litri?

- **Pasul 1.** Se umple vasul B cu 8 litri de apă. Avem 0 litri în vasul A şi 8 litri în vasul B.
- **Pasul 2.** Se goleşte apa din vasul B până se umple vasul A de unde rezultă că în vasul A avem 3 litri de apă, iar în vasul B 5 litri de apă.
- **Pasul 3.** Se aruncă apa din vasul A de unde rezultă că în vasul A avem 0 litri de apă, iar în vasul B 5 litri de apă.
- **Pasul 4.** Se umple din nou vasul A cu apă şi astfel în vasul A avem 3 litri de apă, iar în vasul B 2 litri de apă.
- **Pasul 5.** Se aruncă apa din vasul A de unde rezultă că în vasul A avem 0 litri de apă, iar în vasul B 2 litri de apă.
- **Pasul 6.** Se goleşte apa din vasul B în vasul A de unde rezultă că în vasul A avem 2 litri de apă, iar în vasul B 0 litri de apă.
- **Pasul 7.** Se umple vasul B de unde rezultă că în vasul A rămân 2 litri de apă.
- **Pasul 8.** Se umple vasul A cu apă, de unde rezultă că în vasul A avem 3 litri de apă, iar în vasul B 7 litri de apă.
- **Pasul 9.** Se goleşte vasul A.
- **Pasul 10. S**e umple vasul A cu apă din vasul B de unde rezultă că acum avem 3 litri de apă în vasul A şi 4 litri în vasul B.

Exemplul 2:

Sunt două vase unul A de 3 litri şi unul B de 5 litri. Cum să facem să avem 4 litri în vasul de 5 litri. Se subînțelege că nu dispunem de nici un alt mijloc de măsurare a lichidelor precum şi de faptul că putem vehicula prin cele două vase o cantitate nelimitată de apă. Se umple vasul mare la început.

- **Pasul 1.** Se umple vasul B cu 5 litri de apă. Avem 0 litri în vasul A şi 5 litri în vasul B.

- **Pasul 2.** Se goleşte apa din vasul B până se umple vasul A de unde rezultă că în vasul A avem 3 litri de apă, iar în vasul B 2 litri de apă.
- **Pasul 3.** Se goleşte apa din vasul A de unde rezultă că în vasul A avem 0 litri de apă, iar în vasul B 2 litri de apă.
- **Pasul 4.** Punem apă din vasul B în vasul A, avem 2 litri de apă în vasul A, iar în vasul B 0 litri de apă.
- **Pasul 5.** Umplem cu apă vasul B de unde rezultă că în vasul A avem 2 litri de apă, iar în vasul B 5 litri de apă.
- **Pasul 6.** Se goleşte apa din vasul B în vasul A de unde rezultă că în vasul A avem 3 litri de apă, iar în vasul B 4 litri de apă.

A2. EXPRESII ARITMETICE. SCHEME LOGICE

În realizarea unui algoritm foarte folosit sunt expresii aritmetice care prezintă următoarele particularități.

a) componentele unei expresii:

- Operanzii - variabilele sau constantele;
- operații binare +, - , ·, /;
- operații uneori de exemplu: semnul „-„ în fața operanzilor –(x+y);
- paranteze (,);
- diferite funcții standard

b) liniarizarea – expresiile aritmetice utilizate într-un anumit limbaj de programare, trebuie să fie liniarizate, adică scrise pe un singur rând.

c) prioritățile de calcul – prelucrarea expresiilor aritmetice se realizează de studiul dreptei după următoarele reguli:

- ✓ se execută întâi funcțiile standard;
- ✓ urmează expresiile din paranteze;
- ✓ se execută exponențialele;
- ✓ se execută împărțirile şi înmulțirile;
- ✓ se execută adunările şi scăderile.

d) tirul operatorilor – variabilele şi constantele utilizate în limbajul de programare pot fi:

- ✓ tipul real
- ✓ tipul întreg

Exemple:

Să se liniarizeze expresiile:

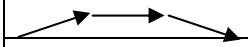
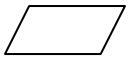
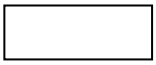
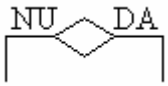
$$m = \frac{a}{2\left(\frac{x+y}{b}\right)} - \frac{b}{a} \Rightarrow m = a/(2(x+y)/b) - b/a$$

$$S = \frac{(a+b)\cdot c}{x^2 + y^2} \Rightarrow S=((a+b)\cdot c) / (x\text{^}2+y\text{^}2)$$

Scheme logice

Schema logică permite familiarizarea programatorilor cu un anumit algoritm şi schimbul de informații între aceştia.

Tabelul A2.1. Simboluri folosite în descrierea schemelor logice:

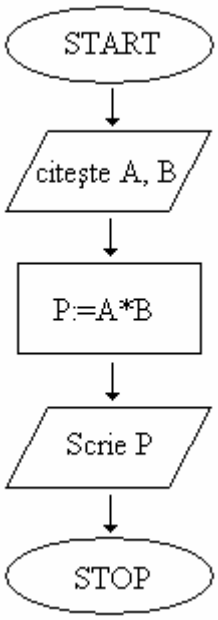
Simbol	Observație
	Fac legătura între blocurile schemei logice
	Conector plasat pe aceeaşi pagină care face legătura între diferitele blocuri ale schemei
	Conector de trecere pe următoarea pagină
	Bloc delimitator care marchează punctul de început şi de sfârşit al schemei
	Bloc intrare – ieşire utilizat pentru înscrierea informațiilor sau extragerea informațiilor
	Bloc de calcul care permite calculul unei variabile sau atribuirea unei valori, unei variabile
	Bloc de procedură utilizat pentru înscrierea procedurilor
	Bloc de decizie

Exemplul 1.

Să se construiască schema logică pentru un algoritm de scriere a produsului oricăror două numere.

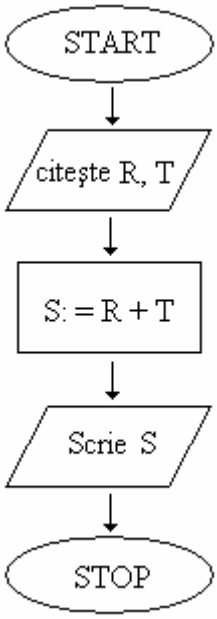
Se aleg două variabile: A, B

P = A * B



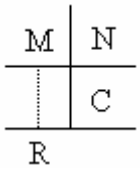
Exemplul 2.

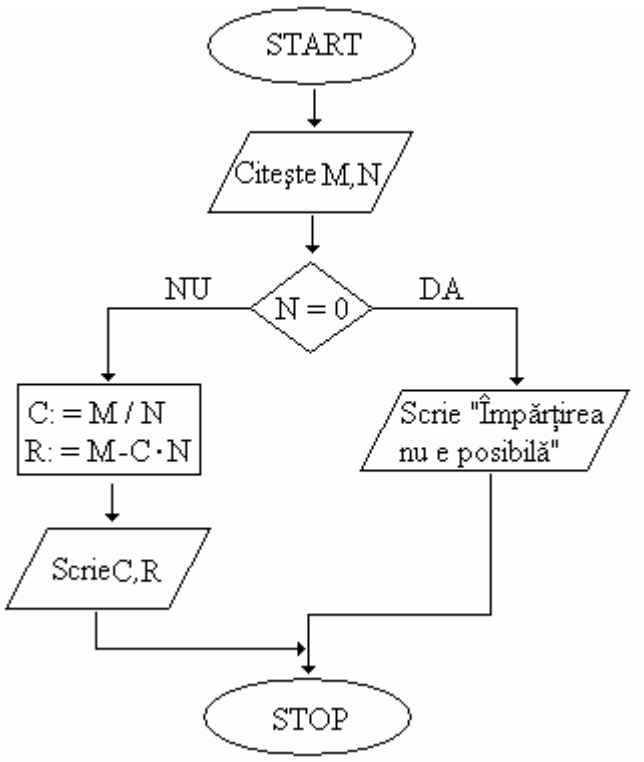
Să se realizeze o schemă care permite calcularea sumei a două numere.



Exemplul 3.

Să se construiască schema logică pentru calculul câtului şi restului a oricăror două numere. M, N M = N·C+R





Exemplul 4:

Să se realizeze schema logică pentru rezolvarea ecuației de gradul I. Forma generală e ecuației de gradul I este următoarea: $ax + b = 0$

Dacă $a = 0 \Rightarrow 0 \cdot x + b = 0 \Rightarrow b = 0 \Rightarrow$ ecuație imposibilă

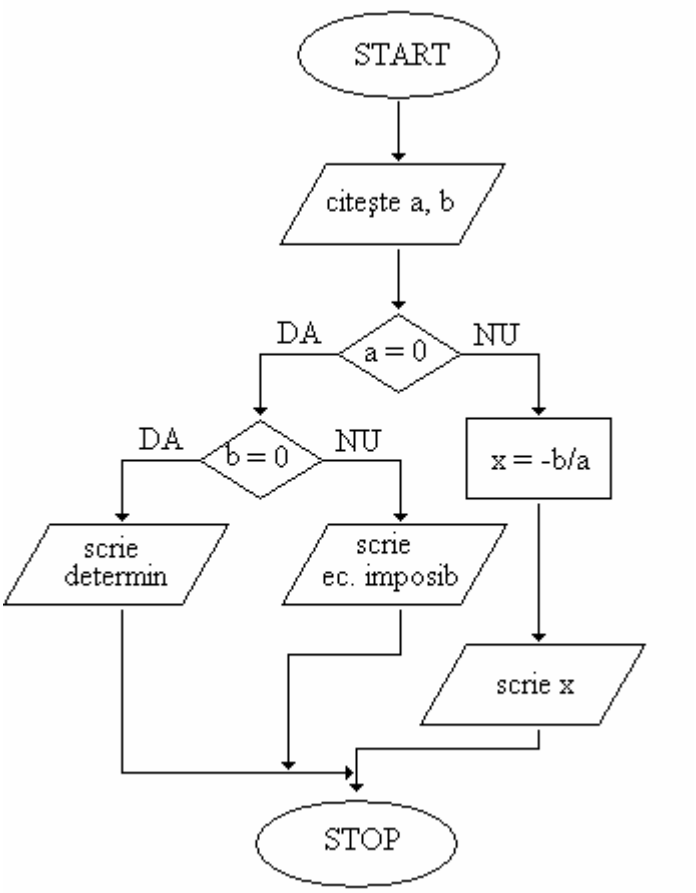
Dacă $a = b = 0 \Rightarrow 0 \cdot x + 0 = 0 \Rightarrow x = 0 \Rightarrow$ o nedeterminare

Dacă $a \neq 0 \Rightarrow x = \dfrac{-b}{a}$

$a = 5$

$\Rightarrow \; b = 10$

$x = \dfrac{-10}{5} = -2$



Exemplul 5:

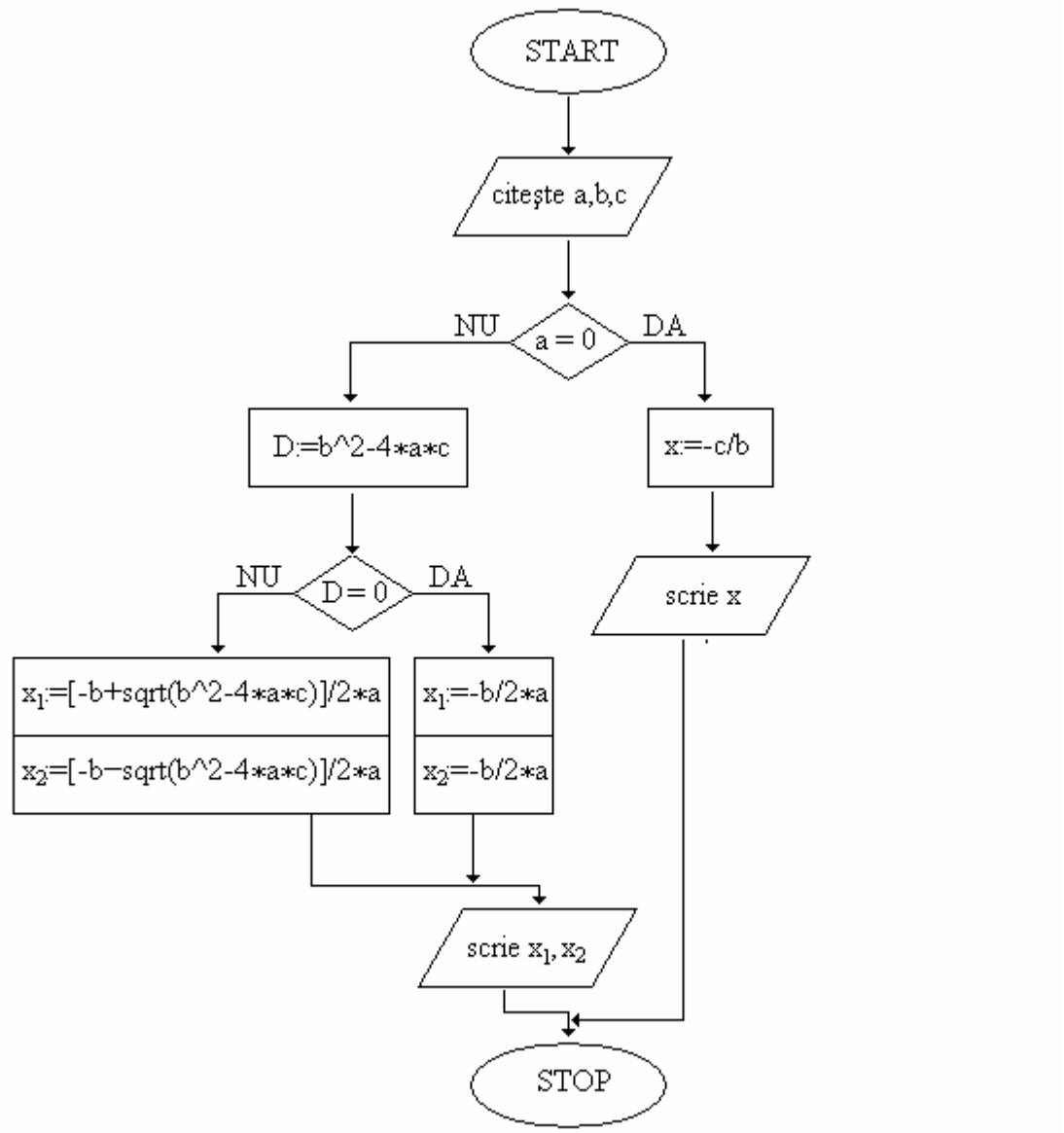
Să se realizeze schema logică pentru rezolvarea ecuației de gradul II. Forma generală a ecuației de gradul II este următoarea: $ax^2 + bx + c = 0$

Dacă $a = 0 \Rightarrow bx + c = 0 \Rightarrow x = -\frac{b}{c}$

Dacă $a \neq 0 \Rightarrow \Delta = b^2 - 4ac$

$$\Delta \neq 0 \Rightarrow x_1, x_2 = \frac{-b \pm \sqrt{\Delta}}{2a}$$

$$\Delta = 0 \Rightarrow x_1 = x_2 = \frac{-b}{2a} \quad \Delta = D$$



Exemplul 6:

Să se realizeze schema logică pentru determinarea sumei a de „n” termeni.

$$A=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \quad mxn \quad b=b_1b_2.....b_n$$

$$S=\sum_{i=1}^{n} b_i \quad n=4 \Rightarrow S=\sum_{i=1}^{4} b_i = b_1+b_2+b_3+b_4$$

$$n=4, i=1, S=0$$

$$b_{(1)}=3$$

$$S=S+b_{(1)}=0+3=3$$

$$i=i+1=1+1=2$$

$b_{(2)} = 7$

$S = S + b_{(2)} = 3 + 7 = 10$

$i = i + 1 = 2 + 1 = 3$

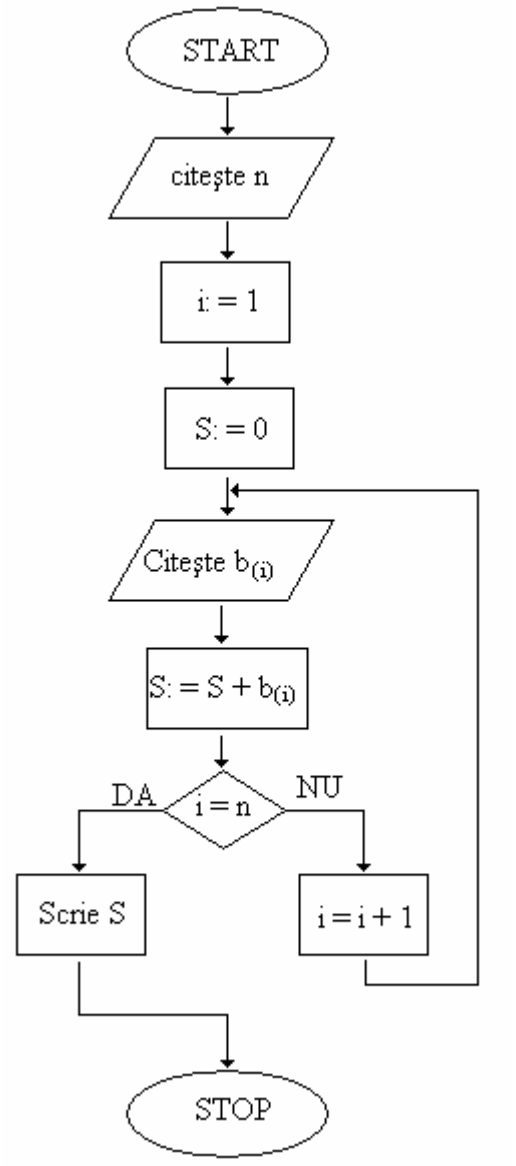
$b_{(3)} = -12$

$S = S + b_{(3)} = 10 + (-12) = -2$

$i = i + 1 = 3 + 1 = 4$

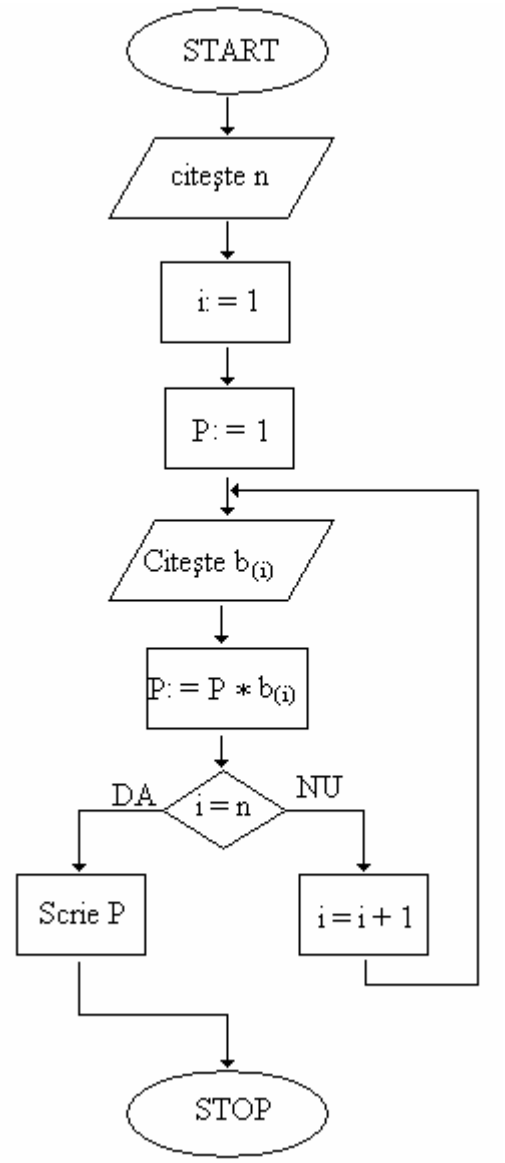
$b_{(4)} = 8$

$S = S + b_{(4)} = -2 + 8 = 6$



Exemplul 7:

Să se realizeze schema logică pentru determinarea produsului a „n” termeni.



Exemplul 8:

Fie n $\in$ R şi $a_{(1)}$, $a_{(2)}$ ……. $a_{(n)} \in$ R. Se cere să se realizeze schema logică pentru determinarea maximului celor „n” termeni.

2 7 4 9

$\max = 2 = a_{(1)}$

Fie: $\max = 7 = a_{(2)}$

$\max = 9 = a_{(4)}$

$n = 3$

$a_{(1)} = 2$

$a_{(2)} = 7$

$a_{(3)} = 4$

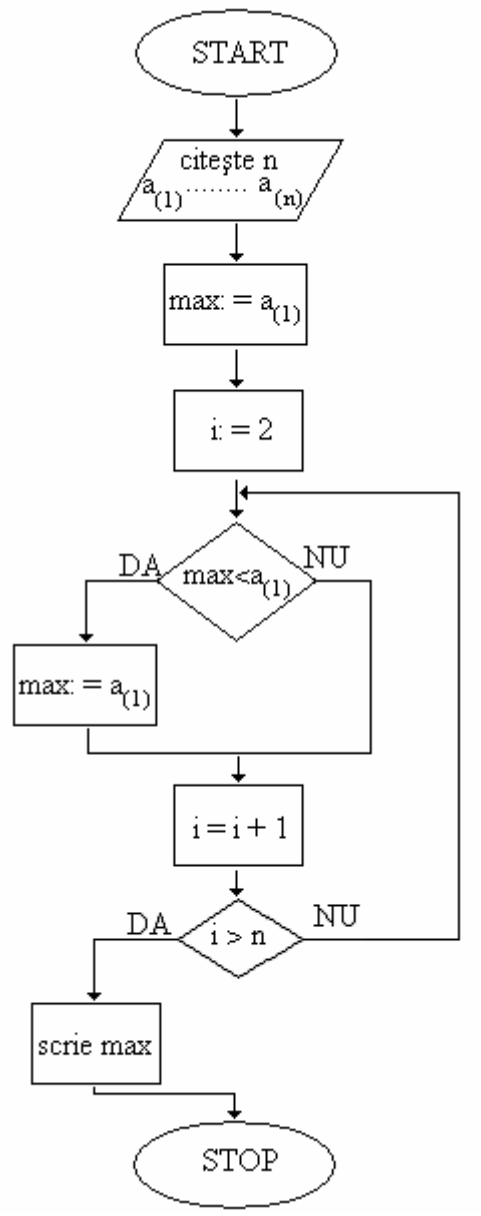
$\max = a_{(1)} = 2, i = 2$

$\max = a_{(2)} = 7$

$i = i + 1 = 2 + 1 = 3$

$i = i + 1 = 3 + 1 = 4$

$\max = 7$



Exemplul 9:

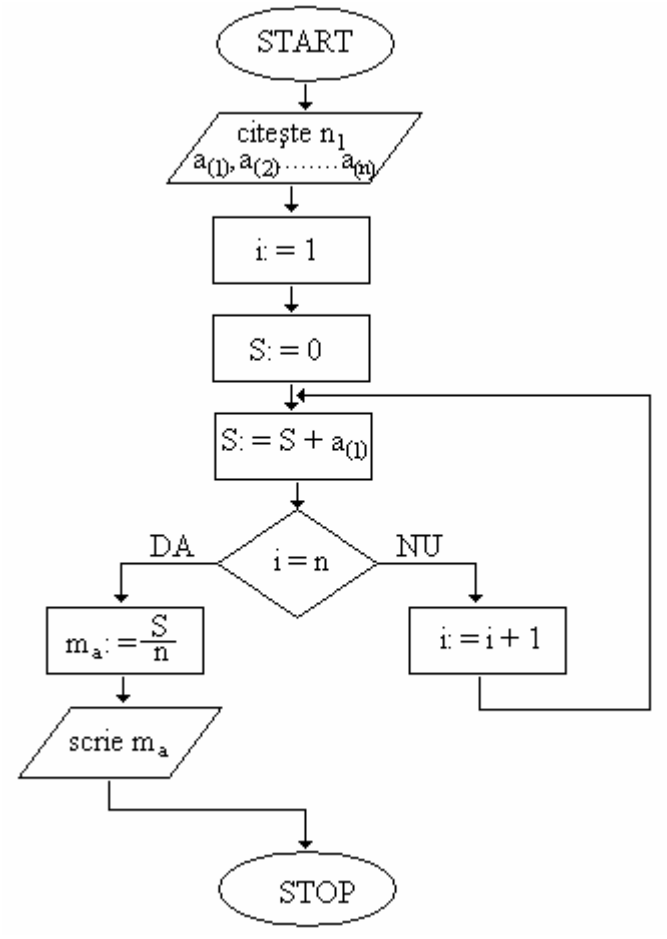
Să se realizeze schema logică pentru determinarea mediei aritmetice a unui şir.

$$m_a = \frac{\sum_{i=1}^{n} a_i}{n} = \frac{a_1 + a_2 + a_3 + ... + a_n}{n}$$

$n = 3$
$a_1 = 4$
$a_2 = 6$
$a_3 = -8$
$i = 1$
$S = 0$

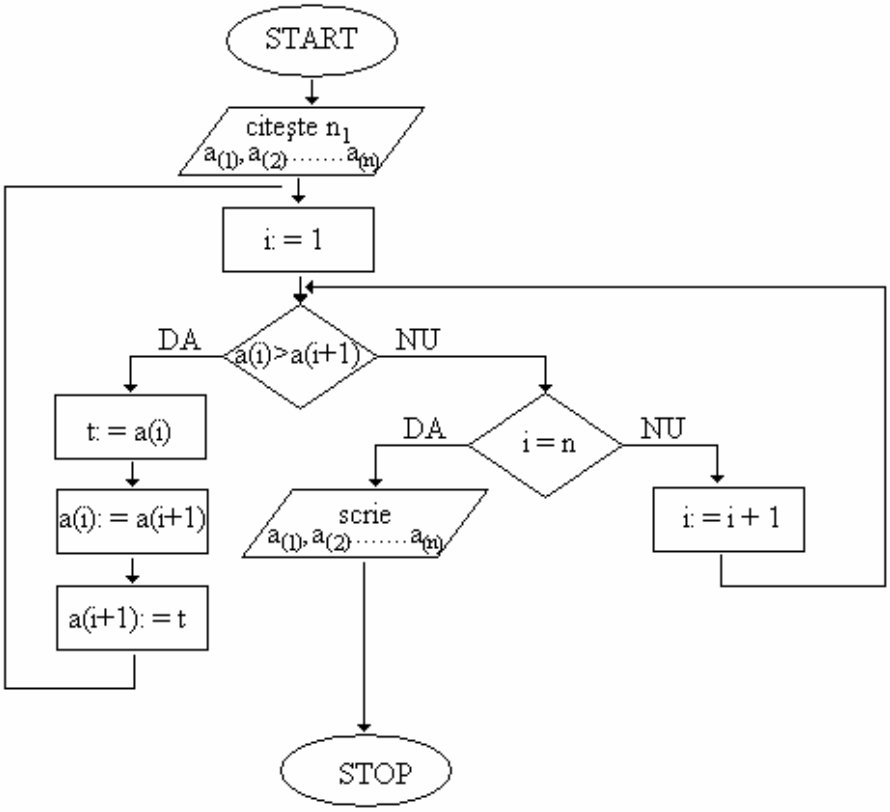
$S = S + a_1 = 0 + 4 = 4$
$i = i + 1 = 1 + 1 = 2$
$S = S + a_2 = 4 + 6 = 10$
$i = i + 1 = 2 + 1 = 3$
$S = S + a_3 = 10 + (-8) = 2$

$$m_a = \frac{S}{m} = \frac{2}{3} \cong 0{,}6$$



Exemplul 10:

Să se realizeze schema logică pentru ordonarea crescătoare a unui şir de „n” termeni.



2 4 3 5 1

$n=5, a_{(1)}=2, a_{(2)}=4, a_{(3)}=3, a_{(4)}=5$

$i=1$

$$a_{(1)} > a_{(2)} \xrightarrow{NU} i=i+1=2 \;\; a_{(2)} > a_{(3)} \xrightarrow{DA} \begin{vmatrix} t=a_{(2)}=4 \\ a_{(2)}=a_{(3)}=3 \\ a_{(3)}=t=4 \end{vmatrix}$$

2 3 4 5 1

$i=1$

$a_{(1)} > a_{(2)} \xrightarrow{NU}$

$i=i+1=2 \;\; a_{(2)} > a_{(3)} \xrightarrow{NU} i=i+1=3 \;\; a_{(3)} > a_{(4)} \xrightarrow{NU} i=i+1=4 \;\; a_{(4)} > a_1$

$$\begin{vmatrix} t=a_{(4)}=5 \\ a_{(4)}=a_{(5)}=1 \\ a_{(5)}=t=5 \end{vmatrix}$$

2 3 4 1 5

$i=1$

$a_{(1)} > a_{(2)} \xrightarrow{NU} i = i + 1 = 2 \;\; a_{(2)} > a_{(3)} \xrightarrow{NU} i = i + 1 = 3 \;\; a_{(3)} > a_{(4)} \xrightarrow{DA}$

$$\begin{vmatrix} t = a_{(3)} = 4 \\ a_{(3)} = a_{(4)} = 1 \\ a_{(4)} = t = 4 \end{vmatrix}$$

2 3 1 4 5

$i = 1$

$$a_{(1)} > a_{(2)} \xrightarrow{NU} i = i + 1 = 2 \;\; a_{(2)} > a_{(3)} \xrightarrow{DA} \begin{vmatrix} t = a_{(2)} = 3 \\ a_{(3)} = a_{(2)} = 1 \\ a_{(3)} = t = 3 \end{vmatrix}$$

2 1 3 4 5

$i = 1$

$$a_{(1)} > a_{(2)} \xrightarrow{DA} \begin{vmatrix} t = a_{(1)} = 2 \\ a_{(2)} = a_{(1)} = 1 \\ a_{(2)} = t = 2 \end{vmatrix}$$

1 2 3 4 5

BIBLIOGRAFIE

1. Knuth D. E., *Arta Programării Calculatoarelor,* vol 1, Editura Teora, 1999
2. Tanenbaum A. S., *Modern Operating Systems*, Prentice Hall, 2001
3. Stevens W. R., *TCP/IP Illustrated, vol 1 Protocols*, Addison Wesley, 1994
4. Tanenbaum A. S., *Rețele de Calculatoare ed. 4*, Biblos, 2003