1. INTRODUCERE

1.1. Calculatoare numerice

Calculatorul numeric reprezintă un sistem digital, format din dispozitive fizice conectate în vederea prelucrării informațiilor numerice.

Un calculator numeric (sistem de calcul) cuprinde două categorii de componente:

- Componente fizice (echipamente), ansamblul lor fiind cunoscut sub numele de *hardware*;
- Componente logice (programe), prin intermediul cărora sunt utilizate echipamentele, ansamblul componentelor logice fiind cunoscut sub numele de *software*.

Sistemul de calcul reprezintă deci un ansamblu de componente hardware şi software în interacțiune, destinat prelucrării datelor. Un asemenea sistem se caracterizează prin următoarele:

- Componentele funcționale sunt realizate cu ajutorul circuitelor electronice, majoritatea fiind circuite integrate, asigurând o viteză ridicată în efectuarea operațiilor aritmetice şi fiabilitate în funcționare.
- Funcționează pe baza unui program memorat, format dintr-o succesiune de instrucțiuni introduse în memoria calculatorului, instrucțiuni care sunt extrase din memorie, interpretate şi executate.
- Informația memorată şi prelucrată este una discretă, fiind codificată astfel încât mărimile asupra căreia operează pot lua numai două valori distincte (0 şi 1). O astfel de informație se numeşte *informație binară*.

1.2. Programarea calculatoarelor numerice

Pentru rezolvarea unei probleme cu ajutorul calculatorului, este necesară descrierea acesteia printr-o succesiune de operații, care constituie *algoritmul* de

rezolvare. Descrierea se poate realiza, la început, în limbaj natural, transpunându-se apoi într-un limbaj artificial, numit *limbaj de programare*.

Limbajul de programare reprezintă un set de instrucțiuni, împreună cu regulile de organizare ale acestora într-un program. Totalitatea simbolurilor şi cuvintelor folosite în limbajul respectiv constituie *vocabularul* limbajului. Totalitatea regulilor de scriere ale instrucțiunilor reprezintă *sintaxa* limbajului, iar totalitatea regulilor prin care se asociază o semnificație instrucțiunilor reprezintă *semantica* limbajului.

Calculatorul poate executa numai instrucțiuni exprimate intern sub forma unor şiruri de cifre binare. Programele în care instrucțiunile sunt scrise sub această formă se numesc programe în *limbaj maşină*. Limbajul maşină este caracteristic fiecărui tip de calculator.

Scrierea şi introducerea programelor sub această formă este dificilă. Pentru simplificarea scrierii programelor, au fost create limbaje în care fiecărui cod de instrucțiune i s-a ataşat un *nume mnemonic*. Acest mod de reprezentare a instrucțiunilor maşină se numeşte *limbaj simbolic* sau *limbaj de asamblare*. Acest limbaj permite utilizatorului să realizeze codificări simbolice ale instrucțiunilor şi ale adreselor de memorie.

Pentru a se executa un program în limbaj de asamblare, acesta trebuie translatat în prealabil în limbaj maşină, printr-un proces numit *asamblare*. Programul care se translatează se numeşte *program sursă*, iar cel rezultat în urma translatării se numeşte *program obiect*. Operația de translatare a programului sursă în program obiect este executată de un program special numit *asamblor*.

O instrucțiune în limbaj de asamblare corespunde unei instrucțiuni în limbaj maşină, deosebirea dintre ele constând numai în modul de reprezentare. Deci, fiecare instrucțiune în limbaj de asamblare este translatată într-o singură instrucțiune în limbaj maşină.

Un limbaj de asamblare este specific unui calculator, astfel încât trebuie să se cunoască instrucțiunile şi organizarea internă a acelui calculator. Din acest motiv, s-au elaborat limbaje pentru programe care se pot executa pe orice calculator. Acestea sunt limbaje orientate pe probleme sau *limbaje de nivel înalt*, spre deosebire de limbajele de asamblare, care sunt limbaje orientate pe calculator sau *limbaje de nivel scăzut*. Asemenea limbaje de nivel înalt sunt *Pascal*, C, BASIC, FORTRAN, LISP, COBOL, PROLOG etc.

Unei instrucțiuni într-un limbaj de nivel înalt îi corespunde o succesiune de instrucțiuni în limbaj maşină. Translatarea în limbajul maşină se poate realiza cu ajutorul unui *compilator*, care generează din programul sursă un *program executabil*, acesta fiind executat după ce întregul program a fost compilat.

O altă posibilitate este utilizarea unui *interpretor*, care translatează fiecare instrucțiune în limbajul de nivel înalt într-o succesiune de instrucțiuni maşină, acestea fiind executate imediat. În acest caz nu se generează un program executabil. Viteza de execuție este însă redusă, deoarece fiecare instrucțiune trebuie interpretată chiar dacă ea este executată în mod repetat.

Uneori este mai convenabil să se considere că există un calculator ipotetic sau o *maşină virtuală*, a cărui limbaj maşină este un anumit limbaj de nivel înalt. Un asemenea calculator ar executa direct instrucțiunile limbajului de nivel înalt, fără a fi necesară utilizarea unui translator (compilator) sau interpretor. Chiar dacă implementarea unei

maşini virtuale care să lucreze direct cu un limbaj de nivel înalt ar fi prea costisitoare, se pot scrie programe pentru această maşină, deoarece aceste programe pot fi translatate sau interpretate cu un program care poate fi executat direct de calculatorul existent.

1.3. Modelul unui calculator numeric

Un model posibil al unui calculator numeric modern reprezintă o ierarhie de maşini virtuale pe mai multe nivele (Figura 1.1).

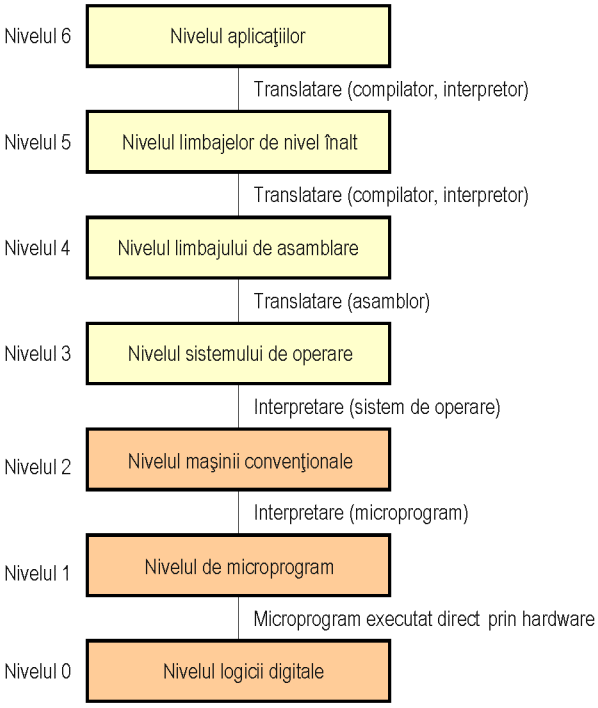

Figura 1.1. Ierarhia de nivele a unui calculator modern.

Nivelul 0, numit *nivelul logicii digitale*, este reprezentat de componentele hardware ale calculatorului (maşina fizică). Circuitele acestui nivel execută instrucțiunile maşină ale nivelului 1. Elementele de bază ale acestor circuite sunt *porțile logice*, fiecare poartă fiind formată la rândul ei dintr-un număr de tranzistoare. O poartă logică are una sau mai multe intrări digitale (semnale reprezentând 0 logic sau 1 logic), şi are ca ieşire o funcție simplă a acestor intrări, de exemplu ŞI logic, SAU logic.

Nivelul 1, numit *nivelul de microprogram*, interpretează instrucțiunile nivelului 2, pentru fiecare instrucțiune a acestui nivel existând câte un microprogram.

Fiecare microprogram defineşte în mod implicit un limbaj de nivel 2 şi o maşină virtuală. De obicei există multe similarităţi între maşinile virtuale de nivel 2, chiar şi în cazul calculatoarelor diferiţilor producători. Acest nivel se numeşte *nivelul maşinii convenţionale*. Atunci când se descrie setul de instrucţiuni al unui calculator, se descrie de fapt maşina virtuală de nivel 2, şi nu maşina reală de nivel 1. Sunt descrise deci instrucţiunile interpretate de către microprogram, şi nu instrucţiunile executate direct prin hardware.

De menţionat că la anumite calculatoare nivelul de microprogram lipseşte. La aceste calculatoare, instrucţiunile maşinii convenţionale sunt executate direct de circuitele electronice ale nivelului 0.

Nivelul 3 este de obicei un nivel hibrid, deoarece multe din instrucţiunile limbajului său sunt prezente în cadrul instrucţiunilor nivelului 2. Există în plus un set de noi instrucţiuni, o organizare diferită a memoriei, posibilitatea de execuţie a mai multor programe în paralel şi alte facilităţi. Noile facilităţi adăugate la nivelul 3 sunt realizate cu ajutorul unui interpretor, numit *sistem de operare*, iar instrucţiunile identice cu cele ale nivelului 2 sunt executate direct prin microprogram. Nivelul 3 este numit *nivelul sistemului de operare*. Un sistem de operare reprezintă un ansamblu de programe care asigură exploatarea optimă a resurselor hardware şi software ale unui sistem de calcul. Sistemele de operare au fost create pentru simplificarea activităţii de programare, utilizarea optimă a posibilităţilor de lucru ale echipamentelor şi reducerea intervenţiei utilizatorilor în cursul execuţiei programelor.

Nivelele 0-3 nu sunt destinate utilizării directe de către programatorii obişnuiţi, ci pentru rularea translatoarelor şi interpretoarelor scrise de programatorii de sistem. Nivelul 4 şi nivelele superioare sunt destinate programatorilor de aplicaţii.

Nivelul 4 este *nivelul limbajului de asamblare*. Programele scrise în limbaj de asamblare sunt translatate în limbajul nivelului 1, 2 sau 3 şi apoi interpretate de către maşina virtuală corespunzătoare.

Nivelul 5 constă din limbajele destinate programatorilor de aplicaţie, fiind numit *nivelul limbajelor de nivel înalt*. Programele scrise în aceste limbaje sunt translatate în limbajele nivelului 3 sau 4 cu ajutorul compilatoarelor sau interpretoarelor.

Nivelul 6 reprezintă *nivelul aplicaţiilor*. Constă din colecţii de programe destinate unor domenii specializate, de exemplu pentru administraţie, economie, proiectare asistată de calculator, grafică etc.

Fiecare nivel reprezintă o abstractizare distinctă, cu diferite obiecte şi operaţii. Setul tipurilor de date, a operaţiilor şi facilităţilor fiecărui nivel reprezintă arhitectura nivelului respectiv. Arhitectura tratează acele aspecte care sunt vizibile utilizatorului nivelului respectiv, ca de exemplu dimensiunea memoriei disponibile. Aspectele de implementare, ca de exemplu tehnologia utilizată pentru implementarea memoriei, nu fac parte din arhitectură. *Arhitectura calculatorului* reprezintă studiul proiectării acelor părţi ale unui sistem de calcul care sunt vizibile pentru programatori.

1.4. Structura maşinii fizice

În 1945, *John von Neumann* a stabilit structura logică a *calculatorului cu program memorat*. Majoritatea calculatoarelor actuale respectă această structură (Figura 1.2).

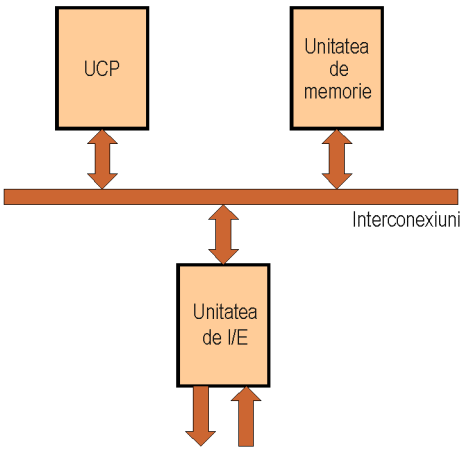

Figura 1.2. Structura maşinii fizice.

Ideea principală a calculatorului cu program memorat este că atât *instrucţiunile*, cât *şi datele sunt păstrate în aceeaşi memorie*. Datele sunt cele asupra cărora se efectuează prelucrări. Instrucţiunile sunt interpretate ca şi coduri pentru generarea semnalelor de control necesare funcţionării calculatorului.

Componentele principale ale maşinii fizice sunt următoarele:

1. *Unitatea centrală de prelucrare* (UCP): Execută prelucrarea datelor şi controlează funcţionarea calculatorului. De multe ori se numeşte *procesor*.
2. *Unitatea de memorie* (*memoria internă* sau *principală*): Păstrează datele şi instrucţiunile.
3. *Unitatea de intrare/ieşire* (I/E): Efectuează transferul datelor între calculator şi mediul exterior acestuia.
4. *Interconexiunile*: Permit comunicaţia între UCP, memoria internă şi unitatea de I/E.

1.4.1. Unitatea centrală de prelucrare

Componenta cea mai complexă este unitatea centrală de prelucrare, cu structura din Figura 1.3.

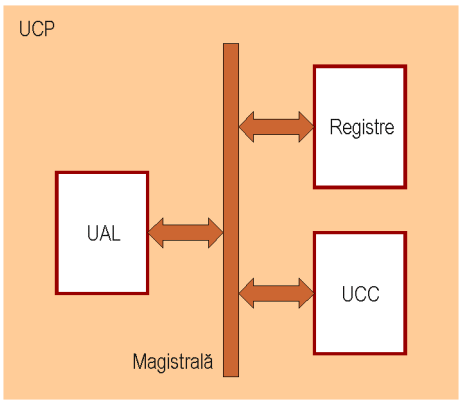

Figura 1.3. Structura unității centrale de prelucrare (UCP).

Componentele principale ale UCP sunt următoarele:

- *Unitatea aritmetică şi logică* (UAL): Execută prelucrările asupra datelor.
- *Registrele*: Reprezintă o memorie internă pentru UCP.
- *Unitatea de comandă şi control* (UCC): Controlează funcționarea UCP şi deci a calculatorului.
- *Interconexiunile* din cadrul UCP: Asigură comunicația dintre UAL, registre şi UCC. Sunt realizate sub forma unei *magistrale*, numită *magistrală internă a* UCP.

1.4.1.1. Unitatea aritmetică şi logică

UAL implementează diferite operații aritmetice şi logice asupra operanzilor obținuți din memorie. Conține, în principal, un circuit logic pentru adunare, numit *sumator*, toate operațiile aritmetice reducându-se la o succesiune de operații de adunare.

Operațiile efectuate pot fi: transferuri de date între registre şi între acestea şi memorie; operații aritmetice; operații logice (ŞI, SAU, NU); operații de deplasare a conținutului unui registru sau locație de memorie; operații de comparație a doi operanzi.

UAL generează informații referitoare la rezultatul ultimei instrucțiuni aritmetice şi logice executate. Acestea se referă la semnul rezultatului, la paritatea acestuia, la cazurile în care a apărut un transport sau un împrumut în cursul prelucrării. Fiecare din aceste informații se păstrează în câte un bistabil, bistabilele fiind reunite într-un *registru de stare*. Acest registru conține şi alte informații referitoare la starea programului.

1.4.1.2. Registrele

Setul de registre din cadrul UCP păstrează temporar operanzii unei operații aritmetice sau logice, rezultatele intermediare şi finale, sau adresele acestora. Utilizarea registrelor creşte viteza de prelucrare, eliminând necesitatea accesului repetat la memorie. Ele reprezintă deci o memorie internă foarte rapidă.

Unele registre pot avea funcții *dedicate*, altele se pot utiliza pentru orice operații, fiind *registre generale*. Un registru special îl reprezintă *registrul acumulator* (sau *acumulator*), care păstrează de obicei unul din operanzii care participă la o operație, ca şi rezultatul operației. O parte din registre nu sunt accesibile prin program, fiind registre de lucru. Un asemenea registru este, de exemplu, *registrul de instrucțiuni*, care păstrează instrucțiunea curentă (cea care se execută la un moment dat).

1.4.1.3. Unitatea de comandă şi control

UCC coordonează activitatea calculatorului: extrage instrucțiunile programului din memorie, le decodifică (le interpretează) şi generează secvența semnalelor de comandă necesare execuției. Separarea în timp a etapelor de execuție se asigură de către un *dispozitiv de secvențiere*.

Pe parcursul execuției unei instrucțiuni, unitatea de comandă primeşte de la unitatea de calcul *informații de stare*, în funcție de care selectează una din alternativele de continuare a operației.

La terminarea execuției, se trece la instrucțiunea următoare. Adresa acestei instrucțiuni este păstrată într-un registru numit *contor de program* sau *numărător de instrucțiuni*.

În funcție de modul de implementare a dispozitivului de secvențiere, dispozitivele de comandă pot fi de două tipuri:

- Realizate în *logică cablată*, de exemplu cu un numărător şi un decodificator;
- *Microprogramate*, care păstrează secvențele semnalelor de comandă într-o memorie de microprogram.

1.4.2. Unitatea de memorie

Memoria reprezintă sursa sau destinația tuturor informațiilor. În memorie sunt încărcate informațiile inițiale (date şi instrucțiuni) prin dispozitivele de intrare, şi de la memorie sunt preluate rezultatele prin intermediul dispozitivelor de ieşire.

Memoria este organizată ca o colecție de locații de memorie. Fiecărei locații i se asociază o *adresă*, prin intermediul căreia se poate selecta locația respectivă. Adresarea se realizează cu ajutorul unor *linii de adresă*, numărul acestor linii determinând capacitatea maximă adresabilă a memoriei. De exemplu, cu 16 linii de adresă se pot selecta maxim 2^{16} locații de memorie.

O locație de memorie se caracterizează prin:

- *Adresă*: poziția locației în cadrul memoriei.
- *Conținut*: valoarea memorată la această adresă.

Cantitatea de informație care poate fi memorată într-o locație adresabilă individual, exprimată ca număr de *cifre binare* (*biți*), se numeşte *lungime a cuvântului de memorie*. De obicei, memoria este organizată pe cuvinte de 16, 32 sau 64 de biți, unitatea adresabilă fiind *octetul*.

Capacitatea memoriei se exprimă în Kocteți (KB) sau multipli ai acestuia:

$1\ KB = 2^{10}\ B = 1024\ B$
$1\ MB = 2^{10}\ KB = 2^{20}\ B$
$1\ GB = 2^{10}\ MB = 2^{30}\ B$

Memoria trebuie să aibă o capacitate cât mai mare şi o viteză cât mai ridicată, adică un timp de acces cât mai redus. Viteza este direct proporțională cu costul. De aceea, majoritatea calculatoarelor au două tipuri de memorii, care lucrează pe principii diferite:

- O *memorie internă* rapidă, numită memorie principală, care comunică direct cu unitatea de calcul şi cea de comandă, cu un cost pe bit relativ ridicat;
- O *memorie externă* mai lentă, cu o capacitate mult mai mare, şi cu un cost pe bit mai redus (disc magnetic, bandă magnetică).

Această realizare a memoriei nu afectează sensibil viteza de calcul, deoarece prelucrarea datelor şi transferul de informații se efectuează la viteza de acces a memoriei interne.

Operațiile efectuate cu memoria sunt cele de *citire* şi de *scriere*. Aceste operații, ca şi cea de *selecție* a unor locații de memorie pe baza adresei, se realizează cu un ansamblu de circuite care formează, împreună cu memoria, *unitatea de memorie*.

1.4.3. Unitatea de intrare/ieşire

Această unitate asigură comunicația dintre calculator şi mediul exterior. Utilizatorii comunică sistemului informațiile înregistrate pe suporturi externe de informație, iar rezultatele prelucrărilor sunt furnizate utilizatorilor pe asemenea suporturi. Transmiterea informațiilor de pe suporturile externe în memorie şi înregistrarea informațiilor pe asemenea suporturi sunt efectuate de *echipamentele periferice*. Acestea sunt conectate la calculator prin *interfețe de I/E*, existente în cadrul unităților de I/E, care îndeplinesc două funcții importante: de conversie de date şi de memorie tampon.

Conversia este necesară deoarece informația este reprezentată diferit şi pe medii diferite, fiind necesară compatibilizarea dispozitivelor respective.

Funcția de memorie tampon este necesară pentru că viteza de lucru a calculatorului (UAL şi UCC) este cu câteva ordine de mărime mai mare decât a dispozitivelor exterioare. Memoria tampon asigură sincronizarea funcționării calculatorului cu aceste dispozitive.

1.4.4. Interconexiuni

Interconectarea unităţilor componente ale calculatorului se realizează prin una sau mai multe *magistrale*. O magistrală este formată dintr-un grup de linii destinate transferului paralel al informaţiilor de la una sau mai multe surse la una sau mai multe destinaţii. Numărul liniilor magistralei este egal de obicei cu lungimea cuvântului transferat.

Magistralele pot fi unidirecţionale sau bidirecţionale. La un moment dat, nu poate fi selectată decât o sursă şi una sau mai multe destinaţii. În cazul selecţiei mai multor surse simultan, rezultatul poate fi imprevizibil.

Selecţia sursei şi a destinaţiei se realizează cu multiplexoare şi decodificatoare. Multiplexorul specifică sursa care depune informaţia pe magistrală, iar decodificatorul selectează destinaţiile care vor fi cuplate la magistrală.

În funcţie de semnalele vehiculate, magistralele pot fi de *adrese*, de *date* şi de *control*.