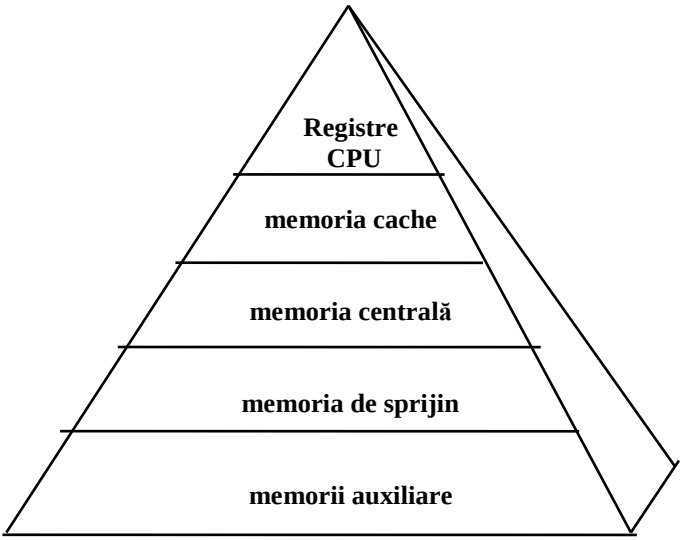
ARHITECTURA CLASICĂ VON NEUMANN

1. Memorii

O memorie este un dispozitiv capabil să înregistreze, să conserve şi să restituie informaţii (codificate în binar în sistemul de calcul).

Ierarhia memoriilor

Elementele diverse al memoriei unui sistem de calcul sunt ordonate în funcţie de următoarele criterii: timp de acces, capacitate şi cost pe bit. Figura următoare prezintă tipurile de memorie şi arată ierarhia existentă între diferitele nivele de memorie.



Se poate constata faptul că pe măsură ce ne îndepărtăm de CPU, timpul de acces şi capacitatea memoriei cresc, în timp ce costul pe bit se diminuează.

a) **registrele** sunt elementele de memorie situate în unitatea centrală de prelucrare (CPU) şi sunt caracterizate printr-o mare viteză, servind în principal stocării operanzilor şi a rezultatelor intermediare. Vom vorbi mai pe larg despre aceste memorii în subcapitolul următor.
b) **memoria cache** sau **antememoria** este o memorie rapidă, de capacitate redusă (în raport cu memoria centrală) utilizată ca

memorie intermediară între CPU şi memoria centrală. Această memorie permite minimizarea numărului de accese la memoria centrală, realizând astfel câştig considerabil de timp.

c) **memoria centrală** este organul principal de aranjare a informaţiilor utilizate de către CPU. Pentru execuţia unui program el trebuie să fie încărcat (instrucţiuni + date) în memoria centrală. Aceasta este o memorie pe semiconductoare al cărei timp de acces este mult mai mare decât cel al registrelor sau memoriei cache.

d) **memoria de sprijin** serveşte drept memorie intermediară între memoria centrală şi memoriile auxiliare. Memoria de sprijin este prezentă în sistemele de calcul cele mai evoluate şi permite creşterea vitezei de schimb a informaţiilor între cele două nivele.

e) **memoriile auxiliare** numite de asemenea şi **memorii de masă** (între care un loc important îl ocupă memoriile de arhivare), sunt memorii periferice de mare capacitate şi cost relativ scăzut. Ele servesc ca dispozitive de stocare permanentă şi utilizează pentru aceasta suporturi magnetice (discuri, cartuşe, benzi) şi suporturi optice (discuri optice) spre deosebire de nivelele mai apropiate de CPU care fac apel la tehnologia semiconductoarelor.

Organizarea informaţiilor

Informaţiile pe care le prelucrează un sistem de calcul trebuie să se adapteze unui anumit format, ale cărui caracteristici generale sunt următoarele:

a) **Bitul**, constituie unitatea de bază a informaţiei. Într-o memorie, cel mai mic element de stocare este numit adesea **punct de memorie**: el memorează un bit de informaţie.

b) **Octetul,** mai cunoscut sub termenul englez de **byte**, corespunde unei succesiuni de 8 biţi.

c) **Caracterul** este o grupare de 6, 7, 8, ... biţi, permiţând codificarea unui caracter alfanumeric sau a unui caracter special (!, #, $, %, ^, &, *, **“**, [,], ...) potrivit convenţiilor de codificare: ASCII, EBCDIC etc.

d) **Cuvântul** [word] este o grupare de biţi constituind unitatea de informaţie adresabilă în memoria centrală şi care variază de la un sistem de calcul la altul. Valorile 32 şi 64 au tendinţa de a deveni valori generale pentru lungimea unui cuvânt.

e) **Înregistrarea** [record] semnifică un bloc de date şi constituie unitatea de informaţie stocată în memoria auxiliară a sistemului de calcul (disc, bandă).

f) **Fişierul** [file] este o mulţime finită de înregistrări.

Caracterisicile memoriilor

a) **Adresa** este valoarea numerică desemnând un element fizic de memorie (de exemplu adresa unui cuvânt în memoria centrală).
b) **Capacitatea** unei memorii corespunde numărului de instrucţiuni pe care le poate conţine şi se poate exprima în funcţie de numărul de biţi, octeţi sau cuvinte.
c) **Timpul de acces** este timpul în care se realizează o operaţie de acces (citire sau scriere).
d) **Ciclul de memorie** este timpul minimal între două accese succesive la memorie. Acesta este mai lung decât timpul de acces deoarece cuprinde şi anumite operaţii de întreţinere, sincronizare, stabilizare de semnale în circuite etc.
e) **Debitul** este numărul de informaţii citite sau scrise pe secundă.
f) **Volatilitatea** caracterizează permanenţa informaţiilor într-o memorie. O memorie volatilă îşi pierde conţinutul la producerea unei întreruperi de curent, deci are nevoie de o alimentare constantă cu energie electrică pentru a-şi conserva informaţiile. Memoria centrală pe semiconductoare este volatilă spre deosebire de memoriile magnetice auxiliare.

Tipuri de acces la memorie

a) **Accesul secvenţial**: este accesul cel mai lent deoarece, pentru a accesa o informaţie particulară trebuie parcurse toate informaţiile care o preced (de exemplu, benzile magnetice);
b) **Accesul direct**: informaţiile posedă o adresă proprie care permite accesarea lor în mod direct (de ex., memoria centrală, registrele);
c) **Acces semi-secvenţial**: este o combinaţie între accesul direct şi cel secvenţial (de exemplu, pentru un disc magnetic, accesul la cilindru este direct şi accesul la un sector este secvenţial);
d) **Accesul prin conţinut (memorie asociativă)**: informaţiile sunt identificate printr-o cheie şi căutarea se efectuează simultan pentru toate poziţiile memoriei (de exemplu, memoria cache).

2 Unitatea centrală de prelucrare

Unitatea centrală de prelucrare sau procesorul central [Central Processing Unit = CPU], sau mai simplu procesor [processor], are rolul de a interpreta şi executa instrucţiunile programului pe un sistem de calcul.

CPU este un adevărat **"**creier**"** al sistemului, direct asociat cu memoria unde sunt stocate instrucţiunile şi datele de prelucrat şi în acest context, ansamblul CPU + memoria centrală este adesea specificat prin denumirea de **unitate centrală** a sistemului de calcul.

Unitatea centrală de prelucrare se compune din două unităţi funcţionale separate: **unitatea aritmetică şi logică (UAL)** şi **unitatea de comandă** şi **de control** [control unit].

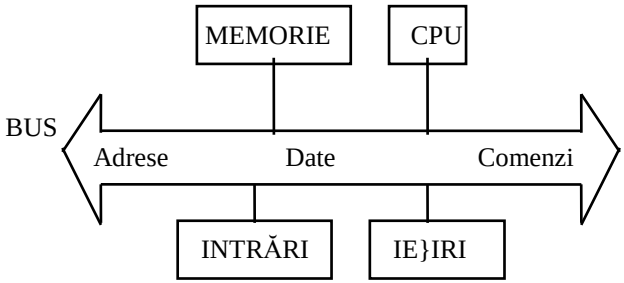
Unitatea aritmetică şi logică este acea componentă a unităţii centrale de prelucrare în care sunt executate operaţiile aritmetice şi logice.

Unitatea de comandă controlează funcţionarea tuturor celorlalte unităţi (UAL, memorie, intrări/ieşiri), furnizându-le semnale de ritmicitate a funcţionării şi de comandă.

Funcţionarea poate fi descrisă în modul următor: unitatea de comandă caută în memoria centrală o instrucţiune transmiţând în acest sens spre memorie o adresă şi o comandă. Instrucţiunea, înregistrată sub formă binară la adresa dată, este transferată către unitatea de comandă, unde decodificarea sa permite determinarea operaţiei cerute. Această informaţie este utilizată pentru a genera semnalele corespunzătoare către UAL pentru declanşarea execuţiei instrucţiunii. Datele de prelucrat vor fi de asemenea căutate în memorie de către unitatea de control şi transferate direct unităţii de calcul.

Diferitele unităţi ale sistemului de calcul sunt interconectate prin sisteme de cablare care transportă semnale electrice. Pentru a se evita legarea unei unităţi cu toate celelalte, se utilizează linii exploatate în comun de către toate unităţile sistemului de calcul. Se numeşte **bus**, ansamblul de linii capabile de a transmite semnale corespunzătoare celor trei tipuri de informaţii: adrese, date şi comenzi.

Există la ora actuală arhitecturi bazate pe bus unic, pe care sunt conectate toate organele sistemului de calcul. O asemenea structură, specifică microcalculatoarelor, este ilustrată în figura următoare:



În general, interconexiunile sunt asigurate prin bus-uri specializate: bus-memorie, bus de intrare/ieşire etc. Bus-ul poate fi utilizat de către toate unităţile care-i sunt ataşate dar nu mai mult decât două simultan, ceea ce ridică probleme de aşteptare şi arbitraj al cererilor de utilizare.

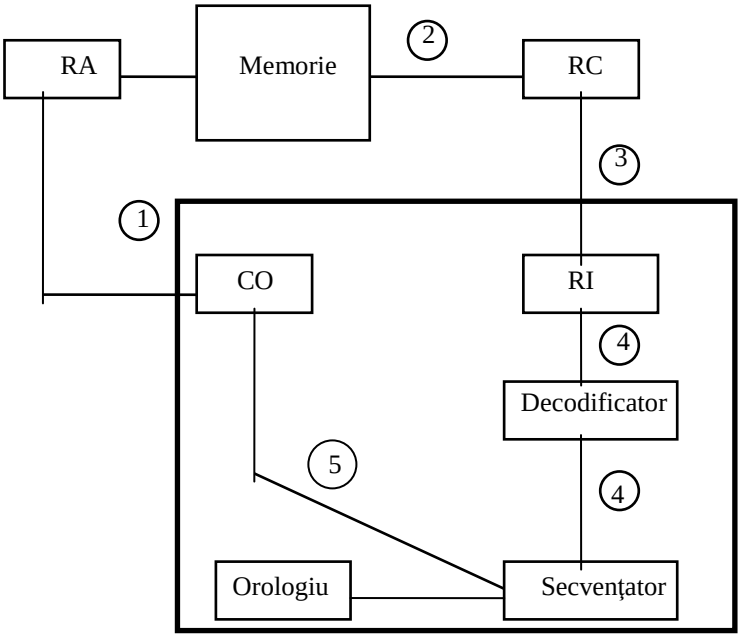
Unitatea de comandă

Unitatea de comandă este mulţimea tuturor dispozitivelor de coordonare a funcţionării sistemului de calcul în vederea executării secvenţei de operaţii specificate prin instrucţiunile programului.

Principalele dispozitive ale unităţii de comandă care vizează căutarea în memorie şi decodificarea unei instrucţiuni (ciclul de căutare), sunt:

a) **contorul ordinal** (CO), este un registru care conţine adresa din memorie unde este stocată instrucţiunea de căutat;
b) **registrul instrucţiune** (RI), primeşte instrucţiunea de executat;
c) **decodificatorul** codului operaţiei, care determină ce operaţie trebuie să fie efectuată, dintre toate cele posibile;
d) **secvenţatorul**, care generează semnale de comandă;
e) **ceasul intern**, sau **orologiul**, care emite impulsuri electrice uniforme, sincronizând astfel toate acţiunile unităţii centrale.

Circulaţia informaţiilor în timpul unui ciclu de căutare [fetch cycle]:



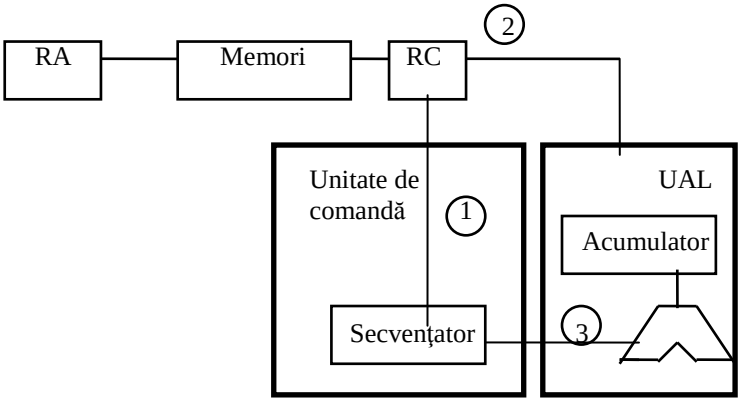
Etapele ciclului de căutare (corespunzând numerelor încercuite în figura de mai sus) pot fi rezumate astfel:

1) transferul adresei noii instrucţiuni din CO către RA, registrul de adresă al memoriei [memory adress register];

2) un semnal de citire, generat de către unitatea de comandă, provoacă transferul instrucţiunii căutate în RC (registrul cuvânt), care funcţionează ca un registru tampon pentru toate informaţiile citite (sau scrise în) din memorie;
3) transferul instrucţiunii în RI (instrucţiune = cod operaţie + adresă operand);
4) în timp ce adresa operandului este trimisă către RA, codul operaţiei este transmis decodificatorului care determină tipul operaţiei solicitate şi-l transmite secvenţatorului printr-un semnal pe linia de ieşire corespunzătoare;
5) CO este incrementat în vederea ciclului următor de căutare.

Ciclul de căutare este imediat urmat de **ciclul de execuţie**, în timpul căruia operaţia specificată prin instrucţiune este efectuată de către unitatea de calcul.

Secvenţa exactă a acţiunilor coordonate de către secvenţator va depinde de tipul operaţiei; în general, în timpul unui ciclu de execuţie, informaţia va circula potrivit schemei următoare:



Un ciclu de execuţie va cuprinde în mod normal următoarele etape:
1) secvenţatorul începe să trimită semnale de comandă către memorie pentru citirea operandului la adresa deja stocată în RA şi realizează transferul către RC;
2) transferul conţinutului din RC către UAL, mai precis către registrul acumulator, sau oricare alt registru destinat operaţiei specificate. În anumite cazuri, de exemplu, memorarea unui rezultat, conţinutul registrului acumulator se va transfera către RC, iar dacă este vorba despre o instrucţiune de salt, câmpul de adresă al instrucţiunii va trebui transferat în CO;
3) operaţia este efectuată sub controlul secvenţatorului.

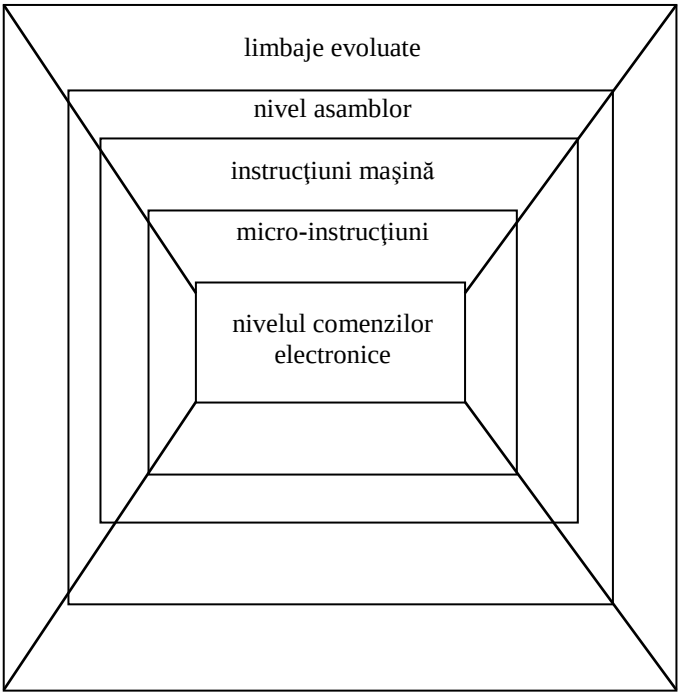
Atunci când ciclul de execuţie este încheiat, unitatea de comandă trece imediat la ciclul de căutare următor, luând în considerare noua instrucţiune indicată prin adresa conţinută în CO.

Nivele de programare

Pentru a scrie un program, utilizatorul poate utiliza unul dintre limbajele de programare cunoscute: **Fortran**, **Pascal**, **Ada**, **Asamblor** etc. Sistemul de calcul nu poate înţelege decât propriul său limbaj, **limbajul maşină**, cu mulţimea sa de instrucţiuni.

În programare, se utilizează termenul **limbaj** pentru a indica o mulţime de instrucţiuni şi de reguli sintactice care permit scrierea **codului sursă** al programului. Deoarece sistemul de calcul execută doar programe scrise în cod maşină sau cod obiect, apare necesitatea traducerii codului sursă în cod obiect, care se realizează automat cu ajutorul compilatoarelor şi asambloarelor.

Limbajele de programare se poate prezenta pe mai multe nivele, cele superioare fiind mai apropiate de limbajul utilizatorului, iar cele inferioare fiind mai bine adaptate caracteristicilor sistemului de calcul. Situaţia actuală este sintetizată în figura următoare:



Structura instrucţiunilor la nivel maşină

Sistemele de calcul sunt capabile să efectueze un anumit număr de operaţii simple, de exemplu adunarea a două numere, testarea semnului unei valori numerice, copierea conţinutului unui registru în alt registru, stocarea în memorie a rezultatului unei operaţii etc.

O instrucţiune maşină trebuie să furnizeze pentru CPU toate informaţiile pentru declanşarea unei operaţii elementare. Ea trebuie să conţină un cod al operaţiei care este esenţial pentru specificarea tipului acţiunii dorite. În plus, instrucţiunea mai trebuie să conţină una sau mai multe adrese, de exemplu, adresa operandului (sau operanzilor), adresa unde se depune rezultatul, adresa de căutare a instrucţiunii următoare.

De aceea, o instrucţiune în formatul unei instrucţiuni maşină va comporta un câmp de cod operaţie şi până la patru câmpuri de adresă.

Un sistem de calcul funcţionează pe **n adrese**, n = 0, 1, 2, 3, 4, dacă majoritatea instrucţiunilor sale conţin n adrese în câmpul de adrese.

Setul şi structura instrucţiunilor maşină caracterizează arhitectura fundamentală a sistemelor de calcul. În acest context, dacă sistemele primei generaţii utilizau frecvent formate pe mai multe adrese ale instrucţiunilor, astăzi, datorită creşterii considerabile a capacităţii de memorare, se preferă instrucţiuni pe o adresă.

Trecerea la instrucţiunea următoare de executat se realizează prin utilizarea unui contor ordinal care se incrementează cu 1 la fiecare operaţie şi acceptă execuţia secvenţială a instrucţiunilor.

Pentru a se evita reţinerea adresei rezultatului într-o locaţie separată, acesta se poate memora în locul unui operand, utilizând un registru special, **acumulatorul**: al doilea operand se găseşte deja în registrul acumulator (încărcat prin instrucţiunea precedentă) şi rezultatul operaţiei va fi stocat de asemenea în registrul acumulator.

Exemplul următor, de programare a unei maşini pe o adresă, prezintă secvenţa instrucţiunilor maşină / asamblor pentru evaluarea următoarei expresii aritmetice: $A = B \times (C + D \times E - F / G)$.

1.	LOAD	F	(LOAD = încarcă în registrul acumulator)
2.	DIV	G	(DIV = împarte conţinutul acumulatorului)
3.	STA	T_1	(STA = stochează conţinutul acumulatorului)
4.	LOAD	D	
5.	MPY	E	(MPY= multiplică conţinutul acumulatorului)
6.	ADD	C	(ADD = adaugă la conţinutul acumulatorului)
7.	SUB	T_1	(SUB = scade din conţinutul acumulatorului)
8.	MPY	B	
9.	STA	A	

Pentru a ilustra utilizarea unei maşini pe zero adrese va trebui utilizată o structură de date specială numită stivă [stack] (sau memorie LIFO = Last In, First Out), operanzii găsindu-se în cele două poziţii superioare şi rezultatul fiind plasat în vârful aceleiaşi stive.

Se poate descrie astfel secvenţa instrucţiunilor corespunzătoare evaluării expresiei din exemplul anterior: $A = B \times (C + D \times E - F / G)$.

1.	LOAD	B	Stiva = {B}
2.	LOAD	C	Stiva = {B; C}
3.	LOAD	D	Stiva = {B; C; D}
4.	LOAD	E	Stiva = {B; C; D; E}
5.	MPY		Stiva = {B; C; D × E}
6.	ADD		Stiva = {B; C + D × E}
7.	LOAD	F	Stiva = {B; C + D × E; F }
8.	LOAD	G	Stiva = {B; C + D × E; F; G }
9.	DIV		Stiva = {B; C + D × E; F / G }
10.	SUB		Stiva = {B; C + D × E – F / G }
11.	MPY		Stiva = {B × (C + D × E – F / G) }
12.	STA	A	Stiva = { }

Setul instrucţiunilor

Fiecare tip de sistem de calcul posedă un set de instrucţiuni de bază, care variază de obicei între 50 şi 250. La ora actuală există două mari tendinţe în ceea ce priveşte setul de instrucţiuni de bază:

a) **Arhitecturile RISC** [Reduced Instruction Set Computer] preconizează un număr mic de instrucţiuni elementare într-un format fix, uşor de realizat din punct de vedere material (hardware) şi cu o execuţie rapidă, ceea ce implică un secvenţator cablat şi un compilator capabil să exploateze bine caracteristicile sistemului (de exemplu, utilizarea pe o scară largă a registrelor şi limitarea acceselor la memoria centrală);

b) **Arhitecturile CISC** [Complex Instruction Set Computer] sunt bazate pe un set bogat de instrucţiuni, de talie variabilă, oferind astfel instrucţiuni compuse, ca de exemplu, calculul rădăcinii pătrate sau înmulţire în virgulă mobilă dublă precizie. În general aceste sisteme sunt prevăzute cu secvenţator multiprogramat.

Instrucţiunile care pot face parte din setul de instrucţiuni de bază ale oricărui sistem de calcul pot fi clasate în şase grupe, astfel:

a) **transfer de date** (Load, Move, Store, transfer de date între două registre sau între memoria principală şi un registru);

b) **operaţii aritmetice** (cele patru operaţii în virgulă fixă sau mobilă, în simplă sau multiplă precizie);
c) **operaţii logice** (AND, OR, NOT, XOR etc.);
d) **control de secvenţă** (salturi condiţionate şi necondiţionate, apel de subprograme etc.);
e) **intrări/ieşiri** (Read, Write, Print etc.);
f) **operaţii diverse** (decalări, conversii de format, incrementări de registre etc.).

Registrele CPU

Numărul şi tipul registrelor pe care le posedă CPU constituie o caracteristică determinantă a arhitecturii sale şi au o influenţă importantă asupra programării.

Deşi structura registrelor CPU diferă de la un constructor la altul, totuşi, funcţiunile de bază realizate prin diverse registre CPU sunt în general aceleaşi. În continuare vom descrie principalele registre CPU:

a) **Contorul ordinal** (CO) [PC=Program Counter] conţine adresa de memorie a următoarei instrucţiuni de executat. El este incrementat automat de către sistem după fiecare utilizare (programatorul nu are acces direct la CO). Programul este executat în secvenţă, cu excepţia situaţiilor în care conţine o instrucţiune de modificare a secvenţei (de exemplu, o instrucţiune de salt sau de ramificare), caz în care noua adresă va înlocui conţinutul contorului ordinal.
b) **Acumulatorul** (Rac) este un registru foarte important al UAL, care, în majoritatea operaţiilor aritmetice şi logice conţine unul dintre operanzi înainte de execuţie şi rezultatul după execuţie. Registrul acumulator poate fi de asemenea utilizat ca un registru tampon în operaţiile de intrare/ieşire. În general Rac are aceeaşi talie ca şi un cuvânt-memorie, dar cele mai multe sisteme de calcul admit extinderea registrului acumulator (registrul Q) pentru a conţine rezultatul unei înmulţiri, câtul şi restul unei împărţiri, sau poate conţine biţii cei mai puţin semnificativi în cadrul unei operaţii în virgulă mobilă dublă precizie;
c) **Registrele generale** [general purpose register] permit salvarea informaţiilor utilizate în mod frecvent de către program sau a rezultatelor intermediare, ceea ce conduce la accelerarea execuţiei programului prin reducerea numărului de accese la memoria centrală;
d) **Registrele de index** sau de **indice** (XR) [index register] se pot utiliza ca registre generale pentru salvări şi contorizări, posedând în plus o funcţie specială, de mare utilitate în manipularea tablourilor de date. Registrele de index pot fi utilizate pentru manipularea

adreselor corespunzător unei forme particulare de adresare numită **adresare indexată**;

e) **Registrele de bază** (Rb) [base registers] sunt utilizate ca registre de index pentru calculul adreselor efective (**adresare bazată**) şi în acest sens ele sunt concepute să conţină o adresă de referinţă care trebuie adăugată la conţinutul câmpului de adresă al instrucţiunii pentru a obţine adresa efectivă;

f) **Registrul de stare** [PSW = Program Status Word], numit de asemenea **registru condiţie**, conţine anumiţi biţi care indică starea unei condiţii particulare în CPU. De exemplu, bitul indicator Z indică dacă rezultatul operaţiei efectuate este egal cu zero, bitul indicator C indică o depăşire de capacitate în Rac etc. Aceşti biţi pot fi testaţi prin program pentru determinarea secvenţei de instrucţiuni de urmat;

g) **Registrul pointer de stivă** [SP = Stack Pointer] este utilizat pentru a simula o stivă în memoria centrală. Registrul SP funcţionează ca un registru de adresă de memorie (RA) şi conţine în orice moment adresa corespunzătoare vârfului stivei, deci, când se încarcă un cuvânt în stivă adresa sa se înscrie în SP, iar citirea din stivă se face pornind de la adresa indicată de pointerul SP;

h) **Registrele specializate** sunt specifice anumitor sisteme de calcul şi sunt destinate operaţiilor particulare, de exemplu, registre de decalare [shift registers], registre pentru operaţii aritmetice în virgulă mobilă [floating point registers] etc.

Adresarea operanzilor

Câmpul de adresă al unei instrucţiuni nu conţine de fiecare dată adresa efectivă a unui operand şi de aceea, pentru facilitarea activităţii de programare sunt puse la dispoziţie mai multe metode de adresare a operanzilor, cunoscute sub numele de **moduri de adresare**, specificate în formatul instrucţiunilor printr-un bit care indică modul folosit.

Printre modurile de adresare, cele mai importante sunt:

a) **adresare directă**: câmpul de adresă al instrucţiunii conţine adresa efectivă a operandului;

b) **adresare indirectă**: câmpul de adresă al instrucţiunii conţine adresa la care găseşte adresa efectivă a operandului (pot exista mai multe nivele de indirectare);

c) **adresare imediată**: câmpul de adresă al instrucţiunii conţine chiar operandul;

d) **adresare implicită**: codul operaţiei indică locul unde se găseşte operandul (de exemplu, maşini pe zero adrese);

e) **adresare indexată**: adresa efectivă = conţinutul câmpului de adresă al instrucţiunii + conţinutul registrului de index;
f) **adresare bazată**: adresa efectivă = conţinutul câmpului de adresă al instrucţiunii + conţinutul registrului de bază;
g) **adresare relativă**: asemănătoare adresării bazate, dar utilizează conţinutul registrului CO ca adresă de bază.

Exemple de adresare

Pentru simplificarea scrierii, în exemplele următoare vom utiliza simboluri ca: **a, b, c, alfa** etc. pentru a indica anumite date şi prescurtări precum: IM (adresare imediată), I (adresare indirectă), XR_1 (registrul de index nr. 1), B_1 (registrul de bază nr. 1). Ne propunem să arătăm care este efectul pe care-l au asupra conţinutului registrului acumulator diferitele moduri de adresare utilizate. Conţinutul memoriei şi al câtorva registre este prezentat în tabela următoare:

Starea registrelor şi a memoriei

adresa	*conţinut*		*adresa*	*conţinut*
100	a		300	alfa
101	b		301	beta
102	c		302	gama
103	d		XR_1	1
200	300	B_1	100	
201	302		B_2	200

În tabela următoare sunt prezentate exemple de operaţii care utilizează registrul acumulator, modurile de adresare fiind specificate după câmpul de adresă al instrucţiunii.

Efectul diferitelor moduri de adresare este prezentat în continuare:

Instrucţiune		*Conţinutul Rac*
LOAD	100	a
LOAD	100, IMM	100
LOAD	200, I	alfa
LOAD	200, XR_1	302
LOAD	200, XR_1, I	gama
LOAD	200, I, XR_1	beta
LOAD	3, B_1	d
LOAD	1, B_2	302

Notăm cu ADR numărul de biţi din câmpul de adresă al unei instrucţiuni şi fie 2^n capacitarea memoriei centrale. Pot exista trei situaţii:

a) ADR = n: memoria fizică este accesibilă în totalitate şi orice metodă de adresare poate fi utilizată;
b) ADR < n: ADR nu este suficient pentru a adresa toată memoria şi în acest caz se poate utiliza metoda de adresare bazată, dacă registrul de bază este suficient de mare (*n* biţi). În acest sens, se poate diviza memoria în blocuri de o asemenea mărime încât ADR să poată adresa complet un bloc. Câmpul de adresă al instrucţiunii va conţine deplasamentul [offset] în interiorul blocului, iar adresa de început a blocului va fi conţinută în registrul de bază. Adresa efectivă se va calcula prin însumarea conţinutului registrului de bază cu conţinutul câmpului de adresă al instrucţiunii.
c) ADR > n: ADR poate adresa poziţii de memorie care nu există în memoria fizică, ceea ce oferă posibilitatea extinderii memoriei principale spre memoriile auxiliare, de exemplu, discuri magnetice (noţiunea de **memorie virtuală)**.

Unitatea aritmetică şi logică (UAL)

Unitatea aritmetică şi logică, specifică sistemelor de calcul moderne, este capabilă să realizeze o mare varietate de operaţii.

Anumite operaţii nu utilizează decât un singur registru şi un singur operand, de exemplu, complementarea logică, decalarea, incrementarea etc., pe când alte operaţii folosesc doi operanzi, de exemplu, adunarea, scăderea, operaţiile logice AND, OR, XOR etc.

Sistemele de calcul specializate pe aplicaţii cu caracter tehnico-ştiinţific oferă o întreagă gamă de operaţii în virgulă mobilă în simplă şi dublă precizie. Microcalculatoarele posedă un număr mai restrâns de instrucţiuni specifice operaţiilor cele mai simple, lăsând în seama programării operaţiile mai complexe, ca de exemplu, înmulţirea, împărţirea, extragerea rădăcinii pătrate, operaţiile în virgulă mobilă etc.

Orice prelucrare de date are loc în cadrul UAL, unde se găsesc toate circuitele capabile să efectueze operaţiile elementare care stau la baza oricărui algoritm. UAL este total subordonată unităţii de comandă care declanşează, controlează şi sincronizează orice activitate a UAL.

3 Unităţile de intrare/ieşire

Funcţiunea de bază a unui sistem de calcul este prelucrarea informaţiei. Am văzut cum se realizează această funcţiune la nivelul memoriei centrale şi a unităţii centrale de prelucrare.

Vom prezenta în continuare modalităţile prin care sistemul de calcul achiziţionează informaţia furnizată de mediul său exterior şi restituie rezultatele prelucrării acestei informaţii.

Comunicarea sistemului de calcul cu lumea exterioară este realizată prin intermediul unităţilor de **intrare/ieşire** [input/output], sau [I/O].

În ultimii ani, tehnica intrărilor/ieşirilor a fost mult dezvoltată, echipamentul intrărilor/ieşirilor este astăzi mai costisitor chiar şi decât unitatea centrală şi ocupă mai mult spaţiu decât orice altă componentă a sistemului de calcul.

Dacă sistemele de calcul din prima generaţie utilizau informaţii preluate prin manipularea întreruptoarelor sau a benzii de hârtie perforată, în continuare se vor utiliza tastaturi alfanumerice, cartele perforate, benzi şi cartuşe magnetice, discuri şi dischete magnetice, discuri optice, ecrane video, mouse-uri, creioane optice, cititoare optice de caractere sau de coduri specializate.

Tehnica ieşirilor a fost de asemenea considerabil dezvoltată. S-a trecut astfel de la lămpile intermitente ale anilor ‘50, la ecranele color şi imprimantele laser ale sistemelor contemporane.

Terminale interactive

Un terminal interactiv este un echipament periferic permiţând utilizatorului o comunicare în ambele sensuri cu sistemul de calcul. Unitatea de intrare este un dispozitiv interactiv precum **tastatura** [keyboard] sau **mouse-**ul, iar unitatea de ieşire este un **ecran de vizualizare** [display] bazat de obicei pe tehnica tubului catodic [CRT].

Tastatura este dispozitivul interactiv prin excelenţă pentru toate aspectele referitoare la tratarea unui text. Pentru o manipulare mai uşoară a obiectelor pe ecran se utilizează mouse-ul.

Tastatura unui sistem de calcul este asemănătoare cu cea a unei maşini de scris care realizează imprimarea pe o foaie de hârtie a caracterelor în urma apăsării tastelor.

Prin acţionarea unei taste este lansat un semnal electronic care este codificat în mod specific [de exemplu în cod ASCII], iar caracterul corespunzător tastei este afişat pe ecranul sistemului de calcul.

Ecrane alfanumerice

Un ecran alfanumeric se limitează la afişarea caracterelor distribuite pe un anumit număr de linii, fiecare linie fiind compusă din mai multe sute de puncte.

Fiecare caracter este format dintr-o configuraţie de puncte alese conform unei anumite grile, generând astfel o matrice de puncte.

Ecranul conţine în general o pagină de text organizată în 24 de linii şi 80 de coloane.

Ecrane grafice

Ecranele grafice permit atât afişarea caracterelor cât şi afişare de imagini sau desene.

Prin apariţia şi dezvoltarea interfeţelor utilizatori - ecrane grafice bazate pe folosirea ferestrelor şi pictogramelor, ecranele grafice au înlocuit ecranele alfanumerice. Cele mai cunoscute ecrane grafice sunt:

a) **Ecrane cu baleiaj TV** la care ecranul este divizat în mici domenii elementare (**pixeli**) care formează un număr de linii şi de coloane. Evoluţia ecranelor cu baleiaj a cunoscut următoarele etape:
   - **ecranele monocrome**, care afişează imagini binare (**bitmap)**, unde fiecare element este reprezentat printr-un bit;
   - **ecranele cu nivele de gri**, care pot afişa imagini **pixmap**, în care fiecare element de imagine poate avea un anumit nivel de gri (8 biţi pot reprezenta 256 nivele de gri);
   - **ecranele color**, permit afişarea imaginilor în culori, dar pentru aceasta sunt necesare memorii-imagine mai mari, deoarece pentru fiecare element de imagine trebuie indicate valorile celor trei culori de bază (bleu, verde şi roşu).

b) **Ecrane vectoriale**, care realizează vizualizarea prin adresarea punctuală a ecranului în locul unui baleiaj sistematic. Se utilizează un fascicol ca **"**stilou**"** pentru trasare curbe conform unui parcurs definit printr-o secvenţă de comenzi stocate într-o memorie locală. Ecranele grafice sunt adaptate aplicaţiilor unde domină liniile, structurile filiforme, desenele prin trasaj (contur), fără o eficienţă deosebită în reproducerea imaginilor pline (fotografii);

c) **Ecrane plate**. Miniaturizarea produselor electronice a antrenat realizarea şi dezvoltarea afişajelor pe ecrane plate, mai mici şi mai puţin fragile decât tuburile catodice. Tehnici, ca de exemplu, **cristalele lichide** sau **ecrane cu plasmă** au devenit competitive.

Imprimante

Tehnicile de imprimare au evoluat considerabil, la ora actuală există o mare varietate de imprimante, acoperind un interval extins de preţuri şi de performanţe. Imprimarea poate fi **alb-negru** sau **color**, principalele procedee de imprimare color fiind jetul de cerneală, transfer termic sau sublimare termică.

Imprimantele color utilizează trei culori primare galben, cyan, magenta, ca şi negru, ceea ce permite crearea unei vaste palete de culori.

Putem prezenta o variantă de clasificare a imprimantelor:

a) **Imprimante cu impact**, încă utilizate în administraţie datorită posibilităţii de a realiza copii carbon:
   - **teleimprimatoarele**: capul mecanismului de imprimare are forma unui cilindru pe care sunt gravate caracterele; între cilindru şi hârtia de imprimat se găseşte o bandă îmbibată de cerneală;
   - **imprimantele Boole**: caracterele sunt înscrise pe o sferă;
   - **imprimantele matriciale (prin puncte)**: caracterele sunt compuse pornind de la punctele unei grile (**matrice**) de 7×9 sau 9×13, potrivit calităţii de imprimare alese;

b) **Imprimantele fără impact**, unde calitatea de imprimare este dată de densitatea punctelor imprimate, care se exprimă în **dpi** [dots per inch]:
   - **imprimante termice**: se aseamănă cu imprimantele matriciale dar în locul lovirii unei benzi, se realizează încălzirea unei suprafeţe de hârtie specială, sensibilă la căldură. Funcţionarea este silenţioasă, iar în ceea ce priveşte imprimarea termică color, sunt cunoscute două tehnologii: **transfer termic** şi **sublimare termică**, ambele utilizând un rulou de celofan acoperit de o cerneală în stare solidă, compusă dintr-o succesiune de regiuni de culoare galbenă, cyan, magenta;
   - **imprimante cu jet de cerneală**: sunt silenţioase şi au viteza de imprimare comparabilă cu cea a imprimantelor termice, putând imprima orice simbol sau grafism. Principiul de funcţionare constă în crearea unui fascicol de picături de cerneală dirijate asupra hârtiei cu o mare precizie. Calitatea de imprimare este foarte bună, rezoluţia variază între 200 dpi şi 1200 dpi;
   - **imprimante laser**: utilizează metode şi procedee electrostatice. Se formează o imagine electrostatică pe un tambur fotoconductor, tamburul trece prin faţa staţiei de dezvoltare unde cerneala încărcată electric este atrasă numai de punctele precedent încărcate. Imaginea este transferată pe hârtie prin frecarea acesteia pe tambur. Această tehnică nu necesită hârtie de calitate superioară, rezoluţia variază între 300 şi 2000 dpi (standard 600 dpi).

Digitalizoare

Digitalizoarele [scanners] sunt echipamente periferice care permit **"**numerizarea**"** unei imagini pornind de la o copie pe suport solid (hârtie). Rezultatul digitalizării este o imagine digitală stocată într-un fişier.

Arhitecturi şi proceduri de intrare/ieşire

Unitatea centrală comunică cu unităţile periferice prin intermediul subsistemului de intrare/ieşire.

Pentru a efectua o operaţie de intrare/ieşire, în CPU se execută o instrucţiune de intrare/ieşire. CPU preia iniţiativa realizării oricărei intrări sau ieşiri, şi în funcţie de natura legăturii între CPU şi echipamentul periferic implicat, se stabileşte o legătură directă CPU-periferic sau se atribuie sarcina schimbului unui dispozitiv subordonat dar capabil să lucreze de manieră autonomă, ca de exemplu, **canal de intrare/ieşire** sau **acces direct la memorie** [DMA: Direct Memory Access].

În cazul legăturii directe între CPU şi periferic apare problema diferenţei enorme între vitezele de lucru ale celor două dispozitive, CPU rămâne blocat pe toată durata transferului de informaţie.

În vederea utilizării raţionale a CPU, se utilizează tehnica **întreruperii programului**. Metoda constă în introducerea unui semnal numit **întrerupere** [interrupt] trimis spre CPU de către perifericul gata să efectueze un schimb elementar, de exemplu, transferul unui octet. Acest semnal provoacă întreruperea programului în curs de execuţie, CPU se ocupă de transfer prin activarea unui program special (**program de serviciu de întrerupere**), iar după efectuarea transferului, CPU reia programul întrerupt.

Abordarea cea mai economică constă în suspendarea programului în execuţie timp de un **ciclu-memorie**, fără un efect deosebit asupra timpului de execuţie. Această metodă se materializează prin utilizarea **canalelor de intrare/ieşire** sau **acces direct la memorie** [DMA].

Sistemul de întreruperi

Întreruperea este un semnal electronic generat de către o unitate funcţională, de exemplu, canal sau controler de periferice, şi acest semnal este transmis spre CPU pentru a provoca o ruptură de secvenţă în vederea execuţiei unui program prioritar, care tratează cauza întreruperii.

Sistemul de întrerupere, încorporat în CPU la nivelul secvenţatorului, este dispozitivul care înregistrează semnalele de întrerupere trimise către CPU, ale căror cauze pot fi:

a) **interne** în raport cu CPU, de exemplu: depăşire de capacitate (overflow), coduri de operaţii inexistente, erori de adresare, utilizare abuzivă a instrucţiunilor privilegiate, pană de curent.

b) **externe** în raport cu CPU, de exemplu: starea unei unităţi periferice, sfârşitul unui transfer de date.

Tratarea unei întreruperi se realizează prin următoarele acţiuni:

a) oprirea execuţiei programului în curs;
b) salvarea stării sistemului;
c) executarea programului de serviciu de întrerupere;
d) restaurarea stării sistemului;
e) reluarea execuţiei programului întrerupt.

Diversele cauze ale întreruperilor sunt afişate într-un **vector de indicatori** asociat sistemului de întrerupere, iar programul care tratează întreruperea trebuie să testeze aceşti indicatori.

În sistemele evoluate, fiecare întrerupere are asociată o adresă în memoria centrală, iar ruptura de secvenţă este realizată prin transferul în contorul ordinal a adresei unde se găseşte programul de serviciu.

Majoritatea sistemelor de calcul moderne sunt prevăzute cu sisteme de întrerupere ierarhizate [priority interrupt systems], acestea fiind sistemele cu nivele de prioritate.

Problemele care trebuiesc rezolvate sunt următoarele:

- sosirea mai multor semnale de întrerupere în timpul execuţiei unei instrucţiuni;
- sosirea unui semnal de întrerupere în timpul execuţiei unui program de prelucrare a unei întreruperi anterioare.

În aceste sisteme, fiecare nivel este asociat unui anumit număr de întreruperi, fiecărui nivel îi corespunde un anumit nivel de prioritate, iar orice program poate fi întrerupt în vederea realizării unei întreruperi mai prioritare.

Sistemele de întrerupere cele mai elaborate sunt cele care fac parte din sistemele de calcul orientate către aplicaţii de conducere a proceselor industriale [process control] sau de achiziţionare de date [data acquisition].

Un sistem de întrerupere modern trebuie să permită programatorului următoarele acţiuni:

- invalidarea/activarea [disable/enable] sistemului de întrerupere;
- mascarea/demascarea individuală a întreruperilor;
- stabilirea unei ierarhii în mulţimea cauzelor intreruperilor şi definirea mai multor nivele de prioritate, de preferinţă dinamic;
- asocierea unui program specific fiecărei întreruperi, permiţând acţiuni elementare realizabile într-o singură instrucţiune;
- posibilitatea de utilizare a tuturor registrelor care caracterizează starea maşinii şi refacerea acestora la sfârşitul programului de întrerupere.