8. MODALITĂȚI DE REPREZENTARE A CALCULATOARELOR

Calculatoarele reprezintă sisteme complexe care pot fi examinate la diverse niveluri de abstractizare, în funcție de scopul urmărit. Adesea un calculator poate fi examinat sub aspect funcțional/comportamental sau structural.

8.1 REPREZENTAREA FUNCȚIONAL/COMPORTAMENTALĂ

Din punct de vedere funcțional/comportamental, sub forma cea mai generală, un calculator se poate reprezenta prin tripletul **<I, E, C >** unde:

- **I** constituie mulțimea intrărilor,
- **E** corespunde mulțimii ieşirilor,
- **C** $\subset$ **I x E** reprezintă o submulțime a produselor carteziene între elementele mulțimii **I** şi elementele mulțimii **E**. **C** realizează aplicații din mulțimea intrărilor **I**, în mulțimea ieşirilor **E**.

Pentru a facilita studiul unui calculator sub aspect funcțional/comportamental, adesea se recurge la reprezentarea lui sub forma unei *ierarhii de niveluri imbricate*.

Un *nivel* este constituit din *mulțimea aplicațiilor asupra elementelor mulțimii de intrare* pentru nivelul dat, cât şi asupra elementelor *mulțimilor de intrare şi ieşire de la nivelul inferior*, imbricat. Aplicațiile de la un nivel dat pot constitui aplicații şi pentru nivelul superior următor.

O posibilitate de reprezentare este dată mai jos, unde se pot distinge trei niveluri imbricate:

- aplicațiile/funcțiile primitive ale maşinii de bază implementate în hardware, capabilă să execute operații elementare,
- funcțiile standard/predefinite, reprezentate prin nivelul instrucțiunilor maşinii convenționale,
- funcțiile construite de utilizator, pe baza instrucțiunilor maşinii convenționale, pentru diverse tipuri de aplicații concrete.

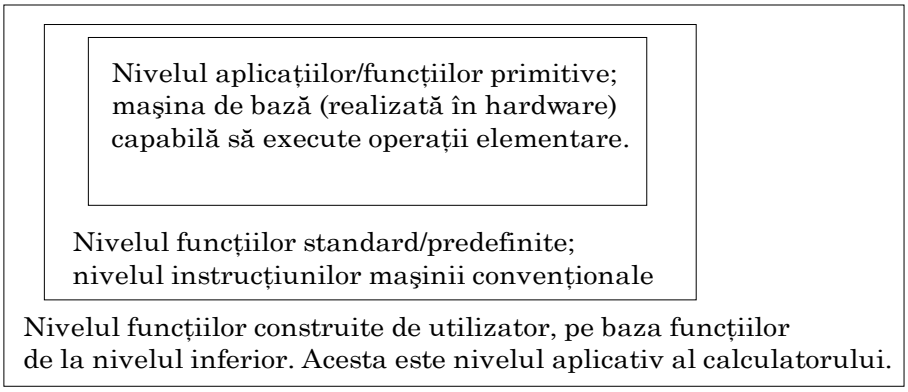

Figura 8.1. Reprezentarea unui calculator la nivel funcțional/comportamental sub forma unei ierarhii de module îmbricate.

8.2 ARHITECTURA CALCULATORULUI NUMERIC

La nivelul maşinii convenționale se defineşte noțiunea de arhitectură a unui calculator numeric (procesor) prin cuadruplul **A = <PI, PE, RG, I>** unde:

PI = **{PI_0, …, PI_i}** este mulțimea porturilor de intrare,

PE = **{PE_0, …, PE_j}** este mulțimea porturilor de ieşire,

RG = **{RG_0, …, RG_k}** este ansamblul registrelor generale din unitatea de execuție

I = **{I_0, …, I_l}** este setul instrucțiunilor calculatorului.

Porturile de intrare şi ieşire sunt utilizate pentru schimbul de informații cu mediul înconjurător, prin intermediul echipamentelor periferice, în timp ce registrele generale sunt folosite pentru stocarea diferitelor variabile de stare.

Sub forma unui prim exemplu, se prezintă mai jos arhitectura microprocesorului Intel 8080.

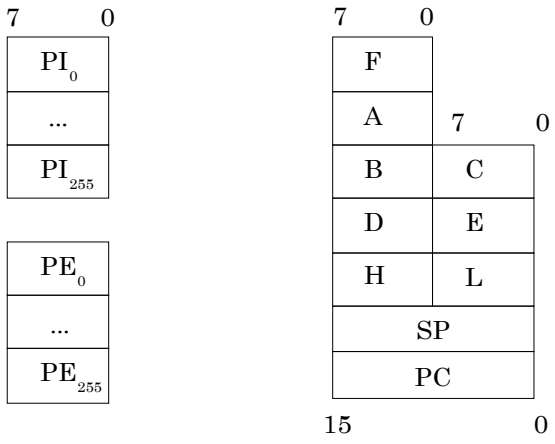

Figura 8.2. Arhitectura microprocesorului Intel 8080

Atât porturile de intrare **PI_0, …, PI_{255}**, cât şi porturile de ieşire **PE_0, …, PE_{255}** au câte 8 biți. În cadrul registrelor generale se întâlnesc şi o serie de registre cu caracter specializat în ceea ce priveşte utilizarea:

- **A** este registrul acumulator principal, folosit în cele mai multe operații aritmetice şi logice;
- **B, C, D, E, H, L** sunt registre de uz general, deşi unele au şi utilizări speciale; **H** şi **L** sunt folosite adesea pentru formarea unor adrese de operanzi pe 16 biți;
- **F** este registrul indicatorilor de condiții, furnizați de către unitatea de execuție după fiecare operație (**CY** - transport în afara rangului de semn, **AC –** transport auxiliar între tetradele octetului rezultat, **S** - semnul rezultatului, **Z** - indicator de rezultat zero, **P** - indicator privind paritatea numărului de unități din rezultat). Pozițiile ocupate de aceşti indicatori de condiții în registrul **F** sunt date mai jos:

7	6	5	4	3	2	1	0
S	Z	*	AC	*	P	*	CY

Figura 8.3. Indicatorii de condiții

- **SP** este indicatorul de stivă, care asigură accesul la o structură de tip stivă organizată în memorie;
- **PC** contor program, pentru adresarea instrucțiunilor din memorie. Lista de instrucțiuni a microprocesorului 8080 conține 78 de instrucțiuni.

Un alt exemplu îl constituie arhitectura microprocesorului 8086.

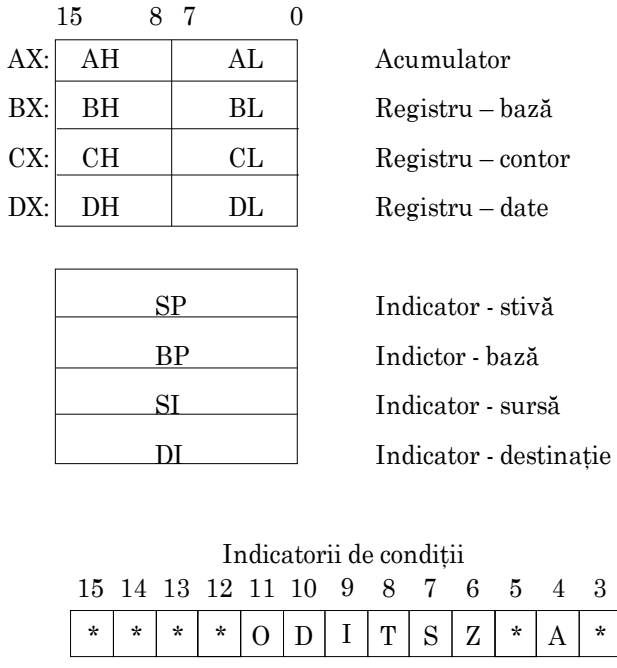

Figura 8.4. Arhitectura microprocesorului Intel 8086; registrele generale şi indicatorii.

Semnificațiile indicatorilor de condiții sunt următoarele: **O** - depăşire aritmetică; **D** - direcția la explorarea şirurilor; **I** - activare/dezactivare întreruperi; **T** - capcană pentru lucrul pas cu pas; **S** – semn; **Z** – zero; **A** - transport auxiliar; **P** – paritate; **C** - transport în afara rangului de semn.

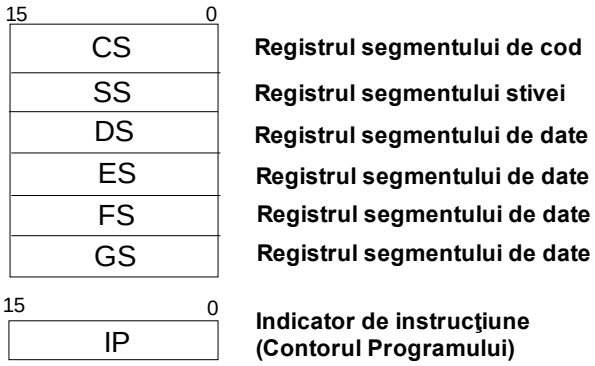

Figura 8.5. Arhitectura microprocesorului 8086: registrele segmentelor şi indicatorul instrucțiunii.

Microprocesorul 8086 mai posedă câte două tablouri de porturi de intrare/ieşire, a câte 65536 octeți fiecare (figura 8.6):

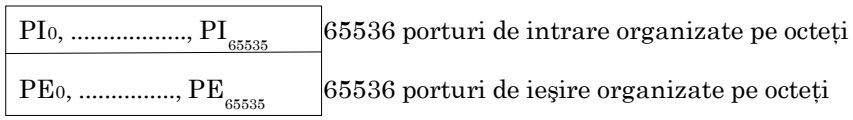

Figura 8.6. Arhitectura microprocesorului 8086: porturile de intrare şi de ieşire.

Notă: Porturile de I/E pot fi organizate şi pe cuvinte de 16 biți, la dimensiunea de 32768 cuvinte pentru fiecare tablou de porturi. Ținând seama de toate modurile de adresare, microprocesorul 8086 dispune de peste 300 de instrucțiuni diferite.

Ultimul exemplu se referă la unitatea centrală a **calculatorului FELIX 5000** care este văzută de către programator sub forma a 16 registre generale **RG_0, ..., RG_{15},** de câte 32 de biți, şi a unui cuvânt dublu de stare program (**PSW** *Program Status Word*), care stochează atât contorul programului, cât şi o serie de indicatori de condiții:

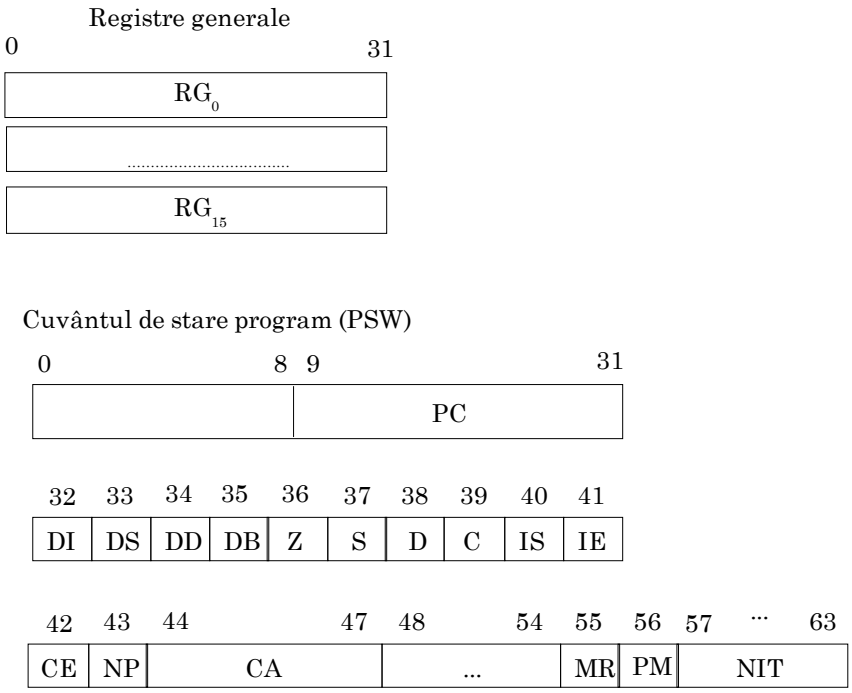

Figura 8.7. Arhitectura procesorului FELIX 5000 văzută de programator.

Câmpurile **PSW** au următoarele semnificații:

PC - contor program;

Măşti de depăşire:

DI, DS - depăşire inferioară/superioară în virgulă mobilă:

- 64 < **E** < 64 , (**E** = exponent);

DD - depăşire zecimală;

DB - depăşire binară: -1 < **n** < 1, (**n** = mantisa).
Dacă masca este unu atunci derutarea este interzisă.

Indicatorii de condiții ai rezultatului:

Z = 1 - rezultat nul;

S = 1 - rezultat < 0;

D = 1 - depăşire;

C = 1 - transport.

Măşti de întrerupere:

IS - mască întreruperi de I/E;

IE - mască întreruperi externe;

CE - mască întreruperi contor nul.
Daca masca este unu, întreruperea este inhibată şi rămâne în aşteptare.

Nivelul programului:

NP = 0 - unitatea centrală operează în modul privilegiat (poate executa toate instrucțiunile);

NP = 1 - unitate centrală operează în modul normal (încercarea de a executa instrucțiunile de sistem generează o derutare).

Cheia de acces:

CA = cheia de acces la paginile de memorie, de câte 2Ko, protejate prin chei de protecție.

Masca de rotunjire:

MR = 0 - se rotunjeşte rezultatul operației în virgulă mobilă;

MR = 1 - rezultatul nu se rotunjeşte.

Paritate memorie:

PM = 0 - eroarea de paritate la memorie este tratată prin derutare;

PM = 1 - eroarea se tratează prin întrerupere.

Nivelul de întrerupere asociat programului:

NIT- defineşte nivelul de prioritate al programului în curs de execuție (pot exista cel mult 128 niveluri de prioritate).

Calculatorul FELIX 5000 are implementate 102 instrucțiuni, din 128 instrucțiuni posibile.

Cunoscând arhitectura unui calculator (porturile de I/E, setul registrelor generale, indicatorii de condiții, lista de instrucțiuni - inclusiv modurile de calcul pentru adresa efectivă) se pot scrie programe la nivelul maşinii convenționale (în limbaj de asamblare).

8.3 REPREZENTAREA STRUCTURALĂ A UNUI CALCULATOR

În acest caz se pleacă de la ideea că un calculator reprezintă un *agregat/sistem*, constituit din *componente primitive* (primitive funcționale) *interconectate* într-o manieră dată, pentru a putea executa operații specifice, de prelucrare a informației.

Componentele se caracterizează printr-o serie de *atribute*, iar atributele - prin *valori*. Informația se măsoară în biți (uneori în ranguri zecimale, caractere alfanumerice etc). Pe lângă funcțiile specifice de prelucrare a informației, *componentele primitive se mai caracterizează prin debit de transfer şi capacitate de stocare a informațiilor.*

Aceste caracteristici prezintă importanță pentru studiul sistemelor de calcul cu ajutorul metodelor cercetării operaționale, în numeroase aplicații (multiprogramare, multiprelucrare, timp divizat, rețele de calculatoare etc.), în care intervin congestii, cozi de aşteptare, zone tampon (buffer), debite de transfer al datelor etc.

Din *punct de vedere structural* sistemul de calcul poate fi descompus (figura 8.8) în:

- unitatea de intrare (**UI**),
- unitatea centrală (**UC_e**),
- unitatea de ieşire (**UE**).

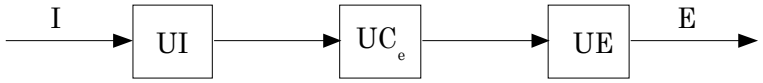

Figura 8.8. Structura generală a unui sistem de calcul.

Unitățile de intrare şi ***de ieşire*** asigură legătura sistemului cu echipamentele periferice primare (traductoarele/elementele de execuție), care preiau informația din mediul extern şi o furnizează în sistem sau care execută diferite acțiuni asupra mediului extern, ca urmare a interpretării informației prelucrate de calculator.

Unitatea centrală asigură stocarea programului, a datelor şi realizează prelucrarea automată a acestora pe baza interpretării programului dat.

Pentru a putea înțelege organizarea şi operarea unor sisteme complexe cum sunt calculatoarele se impune descompunerea lor în unități funcționale mai simple, cu precizarea semnalelor manipulate şi al modului lor de dialog.

Astfel, rafinarea structurii sistemului de calcul evidențiază în zonele **UI** şi **UE**:

- subsistemul de intrare (**SI**),
- subsistemul de ieşire (**SE**),
- echipamentele periferice de intrare (**EPI**) şi echipamentele periferice de ieşire (**EPE**).

EPI au rolul de a prelua datele de la diverse traductoare, eventual stocate pe diverşi purtători fizici, şi a le aduce la o formă compatibilă cu intrările **SI** (ca natură fizică, format de reprezentare etc).

EPE au funcția de a prelua datele prelucrate, furnizate prin **SE**, şi de a le aduce la o formă corespunzătoare destinației: semnale specifice diferitelor elemente de execuție (în cazul conducerii proceselor tehnologice) sau de a le plasa pe un anumit suport fizic etc.

Subsistemele de **I/E** asigură transferul, autonom sau sub controlul **UCe**, al informațiilor între **UCe** şi **EP**, prin realizarea unei adaptări de debite de transfer (vitezele de operare ale acestora diferă cu mai multe ordine de mărime). Uneori sunt implementate şi alte operații: modificarea formatului de reprezentare a datelor, validări, verificări etc.

La nivelul **UCe** se pot distinge următoarele unități funcționale:

- unitatea de memorie (**UM**),
- unitatea de execuție (**UE**),
- unitatea de comandă (**UC**).

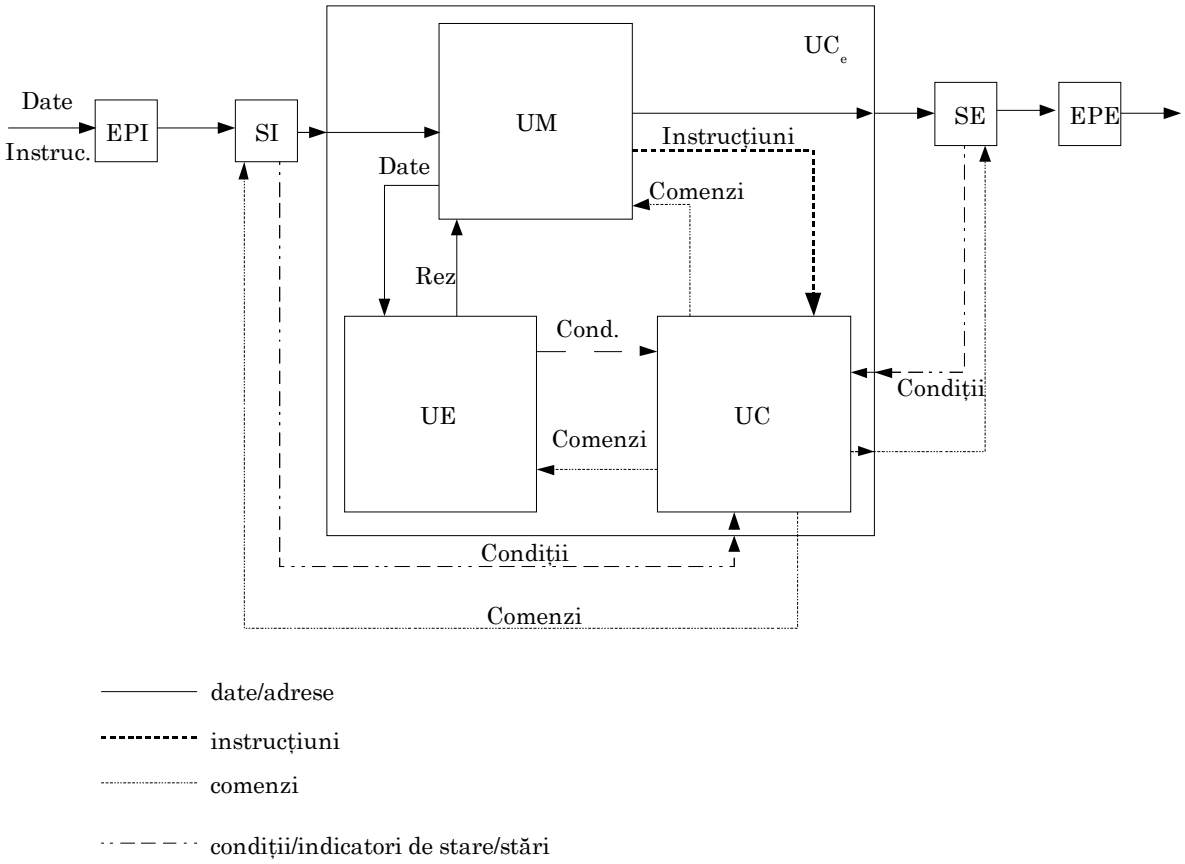

Figura 8.9. Structura unui calculator numeric cu precizarea unităților componente şi a conexiunilor lor.

Unitatea de memorie are funcția de stocare a datelor inițiale, a programului, a rezultatelor intermediare şi finale. În sistemele moderne ea poate opera autonom, atât cu **SI/SE**, cât şi cu procesorul (ansamblul unitate de comandă - unitate de execuție). Unitatea de memorie are o organizare liniară, constând în celule de stocare a informației, al căror conținut poate fi manipulat prin specificarea adresei celulei date.

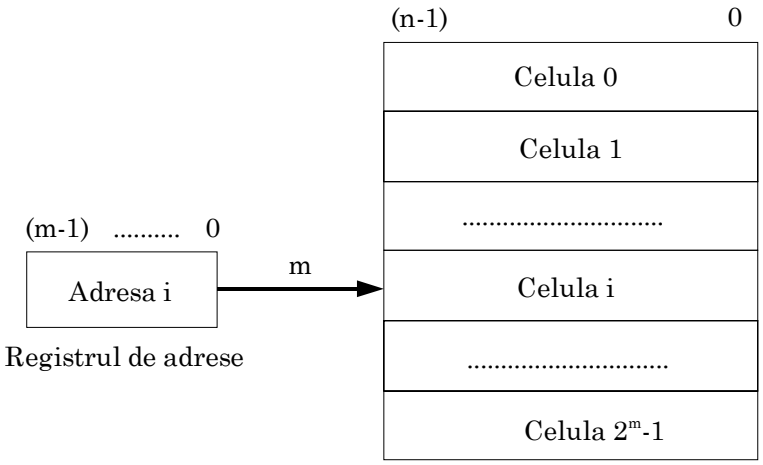

Figura 8.10. Organizarea unității de memorie.

Adresa ia valori cuprinse între **0** şi $2^m - 1$, unde **m** este numărul de ranguri binare ale registrului de adrese.

Unitatea de execuție asigură, sub controlul *unității de comandă*, o succesiune de operații aritmetice şi logice asupra datelor preluate din unitatea de memorie sau din memoria locală-proprie (implementată sub forma unor registre generale -RG-), rezultatele fiind returnate în unitatea de memorie sau în registrele generale.

După fiecare operație **UE** actualizează starea unor *indicatori de condiții*, care reflectă caracteristicile rezultatului curent (< 0, > 0, = 0, paritate, transport, depăşire etc.).

Unitatea de comandă prelucrează fluxul de instrucțiuni, care constituie programul. Ea furnizează semnale de comandă pentru celelalte unități, coordonând funcționarea lor în conformitate cu cerințele programului. UC poate examina, în cadrul execuției unei instrucțiuni date, informația de stare asociată fiecărei unități, luând, în continuare, deciziile corespunzătoare pentru executarea corectă a programului.

Unitatea de comandă se poate implementa sub formă *convențională* (*ca automat secvențial cu stări codificate sau cu stări complet decodificate*) sau sub formă *microprogramată* (cu stocarea semnalelor de comandă, în manieră statică, într-o memorie rapidă).

În unele cazuri **UC** este denumită şi *unitate de prelucrare a instrucțiunilor* - unitatea **I**, iar unitățile de execuție şi de memorie sunt notate cu **E** şi respectiv cu **M**. Astfel, un calculator C (unitatea centrală) poate fi reprezentat prin expresia **C = I [E - M].**

În cadrul **notației structurale PMS** (în care sunt folosite drept componente primitive: **P** - *procesorul*, **M** - *memoria*, **S** - *comutatorul*, **D** - *operatorul asupra datelor*, **K** - *operatorul de comandă*, **L** - *legătura*, **T** - *terminalul/traductorul*) un calculator **C** se descrie prin notația: **C = M - P – T.**

Prima notație este utilă în cazul sistemelor constituite din mai multe unități de execuție şi memorii asociate, controlate de o singură unitate de comandă (*Sisteme cu un singur Flux de Instrucțiuni şi mai Multe Fluxuri de Date - SIMD - Single Instruction Stream Multiple Data Stream*). Sistemele obişnuite au un singur flux de instrucțiuni şi un singur flux de date SISD *(Single Instruction Stream Single Data Stream*).

În rezumat, se poate observa că:

- ansamblul: **UE + UC = P** (Procesor sau Unitate Centrală de Prelucrare – UCP/CPU),
- ansamblul: **P + M = UCe** (Unitate Centrală),
- ansamblul: **UCe + SI + SE + Software** = **Sistem de calcul**.

În figurile de mai jos 8.11 şi 8.12 se vor prezenta câteva exemple de sisteme de calcul, folosind notația PMS:

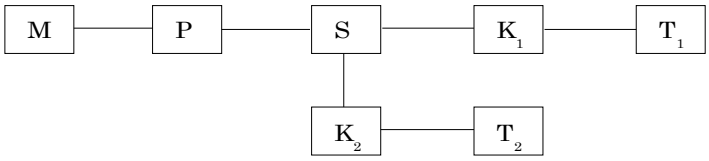

Figura 8.11. Sistem de calcul în care terminalele (echipamentele de I/E) transferă datele cu memoria prin intermediul procesorului.

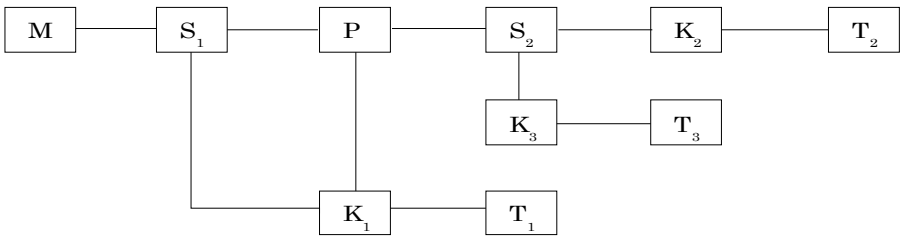

Figura 8.12. Sistem de calcul în care unele terminale (echipamentele de I/E) transferă date cu memoria direct (T_1) sau prin intermediul procesorului (T_2, T_3).

Un *alt exemplu* îl constituie **Microcalculatorul Felix PC**, care având o structură modulară, constă într-un modul de bază şi mai multe module-extensii (Figura 4.13).

Modulul de bază conținea resursele hardware, care asigurau funcționarea sa ca sistem universal, cu o configurație redusă, incluzând: unitatea centrală, tastatura, consola serială, imprimanta şi unitățile de discuri flexibile.

Modulele-extensii aveau un caracter opțional, fiind utilizate în realizarea unor configurații orientate pe aplicații sau în vederea măririi disponibilității şi resurselor sistemului.

Modulul de bază poseda următoarele resurse:

- unitate de prelucrare, bazată pe microprocesoarele 8086/8088 şi 8087;
- memorie RAM de 256Ko;
- memorie EPROM de 8 Ko – 64Ko;
- cuplor pentru discuri flexibile de 5 ¼" sau 8";
- interfețe pentru:
  - tastatură;
  - imprimantă serială;
  - comunicație asincronă-sincronă;
  - casetă magnetică audio;
  - generator de tonuri;
- ceas de timp real;
- numărătoare programabile;
- sistem de întreruperi;
- canal de acces direct la memorie;
- conectori pentru module de extensie;
- conectori pentru periferice.

Modulul de bază era organizat în jurul a doua magistrale: magistrala sistemului, care permitea cuplarea extensiilor şi magistrala locală la care se conectau resursele locale ale sistemului.
Descrierea PMS a modulului de bază este data mai jos, cu următoarele notații:

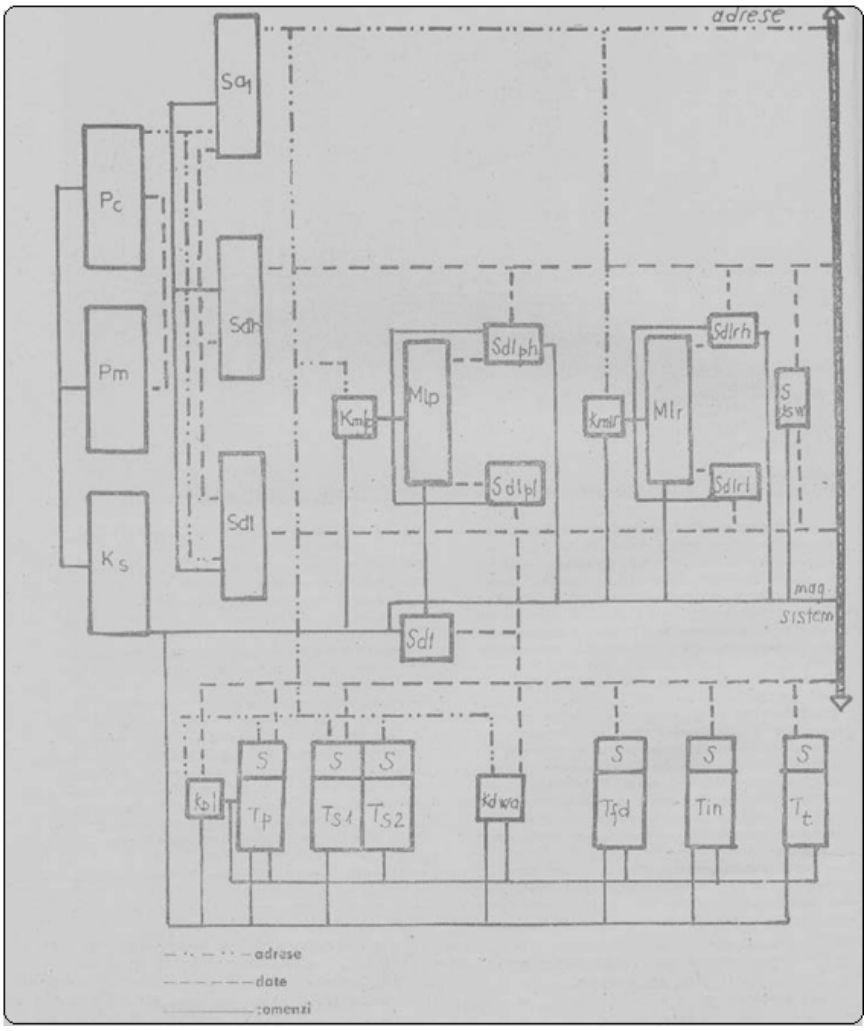

Figura 8.13. Structura microcalculatorului FELIX PC.

P_c – procesor central (8086/8088);

P_n – procesor numeric (8087);

K_s – unitate de control a sistemului;

K_{pl} – unitate de control a resurselor locale;

K_{dma} - unitate de control a accesului direct la memorie;

K_{mlr} – unitate de control a memoriei locale RAM;

K_{mlp} - unitate de control a memoriei locale EPROM;

S_{di} – comutatoare pentru conectarea datelor la magistrală;

S_{ai} - comutatoare pentru conectarea adreselor la magistrală;

S_{dsw} – comutatoare pentru configurarea magistralei 8/16 biți;

M_{lp} – memorie locală EPROM;

M_{lr} – memorie locală RAM;

T_p – interfața paralelă programabilă;

T_{si} – interfețe seriale sincrone/asincrone programabile;

T_{fd} – cuplor pentru discul flexibil;

T_{in} – sistem pentru gestionarea întreruperilor;

T_t – generator de tact programabil.

8.4 INSTRUCȚIUNILE CALCULATORULUI

Instrucțiunile calculatorului reprezintă o colecție de informații privind operațiile care se pot efectua într-un calculator dat.

Sub forma cea mai generală, instrucțiunile se împart în două categorii:

- instrucțiuni *operaționale şi de transfer* al informațiilor, inclusiv instrucțiunile de I/E;
- instrucțiuni *cu caracter de decizi*e, care modifică secvența de execuție a programului în mod condiționat, de o serie de indicatori, sau în mod necondiționat.

Instrucțiunile sunt formate din mai multe câmpuri, dintre care se menționează următoarele:

- *codul de operație/funcția*, care specifică operația/acțiunea elementară evocată de instrucțiune: aritmetică, logică, transfer de date, transfer al comenzii etc.;
- *adresa operandului/instrucțiunii*, care specifică locația de memorie cu care se face transferul de date sau de la care se citeşte o instrucțiune.

În cazul instrucțiunilor cu caracter operațional, structura instrucțiunii poate conține, pe lângă câmpul codului de operație, unul sau mai multe câmpuri pentru adrese.

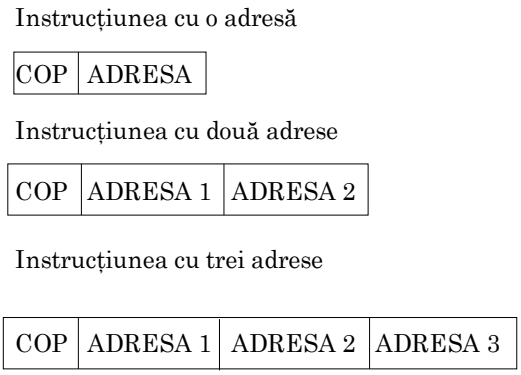

Figura 8.14. Formate simple de instrucțiuni.

În primul caz instrucțiunea specifică adresa unui singur operand, cel de-al doilea operand (în cazul operațiilor ce implică doi operanzi) fiind deja adus în UE. De regulă, rezultatul rămâne în UE.

În cazul al doilea sunt prezente adresele celor doi operanzi (adresa operandului destinație şi adresa operandului sursă - rezultatul operației se stochează la adresa operandului destinație).

Câmpul ADRESA3, în cel de-al treilea caz, poate specifica adresa la care se trimite rezultatul. Instrucțiunile cu trei adrese sunt mai rar întâlnite. Câmpul de adresă din instrucțiune poate avea mai multe semnificații:

- conține chiar operandul, în cazul *operandului imediat*;
- reprezintă adresa unui operand din memorie, în situația *adresării directe*;
- reprezintă adresa unei celule de memorie în care se află adresa unui operand, în cazul *adresării indirecte*.

În unele situații, la conținutul câmpului de adresă din instrucțiune se adună conținuturile unor *registre speciale*:

- *registrele bază,* la *adresarea bazată*;
- *registrele index*, la *adresarea indexată*.

Cu ajutorul acestor facilități se pot adresa diverse date stocate în memorie, care sunt structurate sub formă de masive, multidimensionale, în cazuri particulare: matrici şi vectori.

Registrul bază conține adresa de start (baza) a structurii, adresa din instrucțiune reprezintă o deplasare în structura dată, iar registrul index asigură adunarea unei cantități variabile (incrementabile/decrementabile - eventual automat).

Adresa obținută în urma unor asemenea operații, adesea combinate, poartă numele de *adresă efectivă*. În consecință, instrucțiunea va conține câmpuri adiționale, care vor specifica adresarea *bazată, indirectă, indexată* etc, deci, *modul de calcul al adresei efective*.

Aceste artificii sunt impuse de faptul că, în calculatoarele convenționale, memoria este organizată liniar, în cadrul ei informația fiind stocată sub forma unor structuri date. Diversele moduri de adresare facilitează accesul la componentele acestor structuri.

Trebuie menționat faptul că listele de instrucțiuni pentru cele mai multe calculatoare conțin şi *instrucțiuni fără adresă*. Acestea se bazează pe facilitățile implementate în hardware care permit

adresarea şi manipularea datelor plasate într-o structură specială numita *stivă* (stack - **LIFO** - **L**ast **I**n **F**irst **O**ut), organizată în memorie sau (mai rar) în asamblaje de registre aflate în unitatea de execuție. Accesul la informația plasată în stivă se realizează numai prin intermediul vârfului stivei, folosind doua operații de bază: plasarea unui nou cuvânt în stivă (**PUSH**) şi extragerea cuvântului din vârful stivei (**POP**). Adresa celulei ocupate din vârful stivei este stocată într-un registru indicator de stivă (**Stack Pointer - SP**) al cărui conținut este automat incrementat după o operație **POP** şi respectiv - decrementat, după o operație **PUSH**.

Instrucțiunile, care afectează execuția programului, trebuie să specifice în câmpul codului de operație condițiile testate, iar în câmpul de adresă deplasarea necesară calculului adresei efective. Şi în acest caz, *adresa efectivă se poate obține direct, indirect, bazat, indexat sau ca urmare a unor operații combinate* (figura 8.15).

Formatul instrucțiunii calculatorului FELIX 5000, conține în cadrul a 32 de biți mai multe câmpuri:

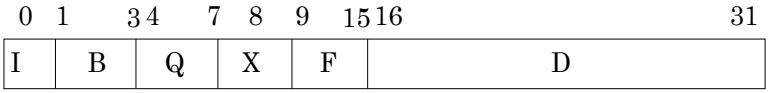

Figura 8.15. Formatul instrucțiunii calculatorului FELIX 5000

unde:

- **I** poate lua valoarea 0 în cazul adresării directe şi 1, în cazul adresării indirecte;
- **B** ia valori între 0 şi 7, specificând pentru valori mai mari decât 0, adresarea bazată cu unul din registrele generale $\mathbf{RG_9 - RG_{15}}$, iar pentru 0 adresarea nebazată;
- **Q** specifică: registrul general, care conține unul din operanzi, o constanta sau un vector logic sau adresa de origine şi lungimea unui operand, în cazul manipulării şirurilor, şi registrul general $\mathbf{RG_0 - RG_{15}}$ folosit la indexare;
- **X** ia valoarea 0, în cazul adresării neindexate şi valoarea 1, în cazul adresării indexate;
- **D** specifică deplasarea, folosită pentru calculul adresei efective, prin adunarea cu registrul bază. Dacă registrul bază nu este specificat, **D** asigură adresarea pe o zonă de 64 Ko. Pentru explorarea întregului spațiu de memorie se adună în mod curent conținutul lui **D** cu conținutul registrului bază**.**

Modurile de adresare întâlnite la FELIX 5000:

- *Adresare directă*: **I = 0; X = 0; B ≠ 0**.

  $\mathbf{A_{ef} = (B + 8) + D}$; unde **(B + 8)** specifică registrul general utilizat ca bază.

- *Adresarea indirectă*: **I = 1; X = 0; B ≠ 0**.

Prima adresă efectivă, $\mathbf{A_{ef}}$, calculată va reprezenta adresa unui cuvânt din memorie în care se găseşte o informație, care este tratată, din punctul de vedere al calculului unei noi adrese efective, ca o instrucțiune. Astfel, la adresa efectivă:

$\mathbf{A_{ef1} = (B + 8) + D}$ se va găsi un cuvânt care va avea câmpurile: $\mathbf{I_1, B_1, Q_1, X_1, F_1, D_1}$. Dacă $\mathbf{I_1 = 1}$, atunci se va calcula noua adresa efectivă:

$\mathbf{A_{ef2} = (B_1 + 8) + D_1}$, de la care se va citi un nou cuvânt, procesul fiind continuat pe maximum 5 niveluri de adresare indirectă, până când **Ii = 0**.

- *Adresarea indexată directă*: **I = 0; X = 1**.

  $\mathbf{A_{ef} = (Q) + (B + 8) + D}$, unde **(Q)** specifică **RG** folosit ca registru index.

- *Adresarea indexată indirectă*: **I = 1; X =** 1.

  $\mathbf{A_{ef1} = (B + 8) + D,}$

  $\mathbf{A_{ef(k+1)} = (B_k + 8) + D_k,}$

  $\mathbf{A_{ef} = (Q) + A_{ef(k+1)} = (Q) + (B_k + 8) + D_k.}$