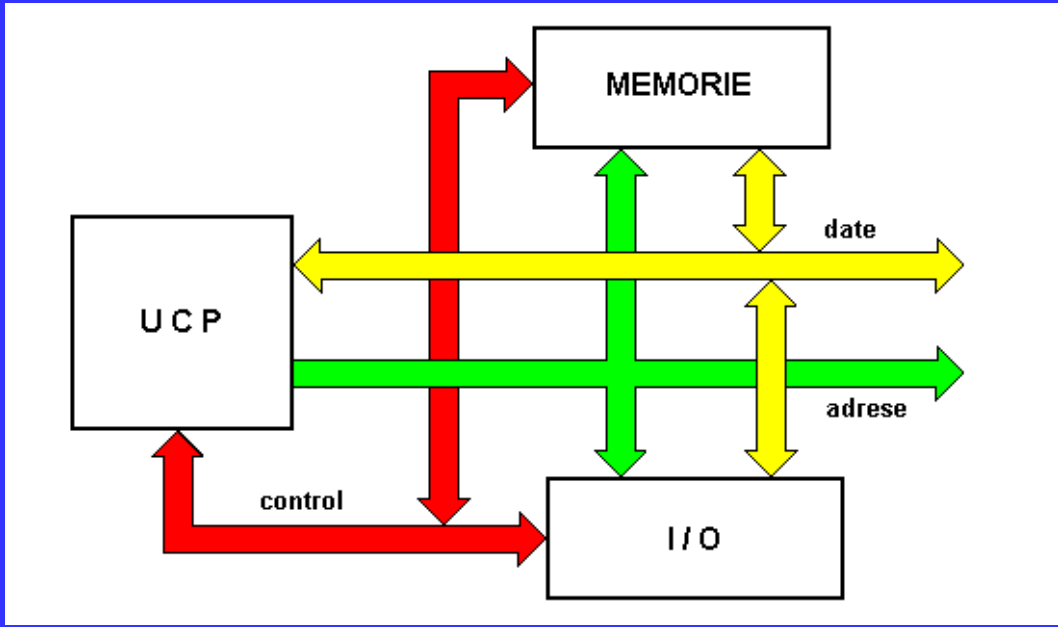
1. STRUCTURA UNUI MICROCALCULATOR. DEFINIȚII

1.1. Componentele funcționale ale unui microcalculator

Microcalculatorul, structurat ca o maşină "VON NEUMANN", este un sistem programabil de prelucrarea informației care are două componente inseparabile şi definitorii:

- **hardware**
- **software**



A. Componenta hardware;

blocurile funcţionale sunt:

1. UNITATEA CENTRALĂ DE PRELUCRARE (UCP); două funcţii esenţiale:

- **prelucrarea datelor;**
- **controlul activităţii întregului microcalculator.**

O Unitate centrală de prelucrarea informaţiei, având funcţiile enunţate mai sus, care coordonează un sistem structurat funcţional ca în figură şi care, fizic, se prezintă sub forma unui singur cip se numeşte MICROPROCESOR (µP)

2. MEMORIA este o secvenţă de locaţii pentru stocarea informaţiei. Fiecare locaţie este definită prin două entităţi informaţionale:

• *Conţinutul*, reprezentat de o înşiruire de cifre binare **0** sau **1** ("**biţi**");

- numere
- coduri etc.

Numărul de cifre binare conținute într-o locație depinde de modul în care microprocesorul organizează informația în memorie; mărimea unei locații va fi denumită **formatul memoriei**, exprimat în număr de biți (de regulă 8, 16, 32 sau 64 biți).

• *Adresa*, reprezentând numărul de ordine al locației, care permite identificarea sa în cadrul secvenței de locații (există o corespondență biunivocă între fiecare locație de memorie şi adresa sa).

Noțiuni aferente:

- "**Harta memoriei**": totalitatea locațiilor de memorie pe care le poate adresa un microprocesor.

- "**Pagini**" şi/sau "**segmente**": subdiviziuni *logice* ale hărții memoriei, ale căror dimensiuni, fixe sau dinamice, sunt specifice modului în care un microprocesor anume organizează memoria.

Structura fizică a memoriei unui microcalculator este formată din unul sau mai multe cipuri, cu capacităţi diverse; capacitatea totală de stocare a informaţiei pe care o realizează fizic cipurile de memorie într-un microcalculator este definită ca "memorie internă". Aceasta nu acoperă, în mod necesar, harta memoriei aferentă microprocesorului respectiv

Semnificaţia conţinutului memoriei microcalculatorului → două zone:

- Memoria de date conţine operanzi şi/sau rezultate; fizic, această porţiune de memorie este de tip RAM (cu scriere/citire).

- Memoria de program care conţine instrucţiuni; de regulă, (dar nu obligatoriu) această zonă este o memorie de tip ROM (memorie din care se poate doar citi).

Instrucţiunea: informaţia codificată (binar) prin care se impune microprocesorului desfăşurarea unei acţiuni specifice.

Observaţii:

• Fiecare instrucţiune este asociată în mod biunivoc cu un şir de cifre binare; deoarece acestea "codifică" instrucţiunile, vor fi denumite coduri.

• O instrucţiune reprezintă cea mai simplă acţiune, cu rezultat bine precizat, din activitatea unui microcalculator a cărui *unitate centrală de prelucrare a informaţiei* este un microprocesor anume.

• Un microprocesor concret poate "recunoaşte" şi executa numai codurile corespunzătoare instrucţiunilor pentru care a fost construit; totalitatea instrucţiunilor pe care un microprocesor le poate recunoaşte şi executa alcătuieşte setul de instrucţiuni al microprocesorului respectiv.

• Înşiruirea instrucţiunilor în memoria de program nu este haotică ci sub formă de programe, noţiune definită ca fiind o secvenţă de coduri de instrucţiuni organizate în mod logic şi coerent după un anumit algoritm, astfel încât întregul microcalculator să execute o "sarcină" prestabilită (task).

Semnificația conținutului locațiilor de memorie este conferită de programator în concordanță cu funcțiile specifice realizate de microprocesor:

- numere binare atunci când ne referim la date (operanzi/rezultate);

- coduri când ne referim la instrucțiuni.

În schema bloc funcțională propusă, memoria nu are nici un control asupra semnificației informației pe care o conține

3. DISPOZITIVELE DE INTRARE/IEŞIRE (I/O): circuitele prin care se realizează legătura între microcalculator şi lumea exterioară. O unitate elementară de conversaţie cu exteriorul poartă numele de "port de intrare/ieşire".

Între porturi şi locaţiile din MEMORIE există nişte similitudini:

- Porturile sunt în esenţă tot locaţii de memorare a informaţiei, adresabile; informaţia care se foloseşte uzual aici este alcătuită din operanzi/rezultate (date).

- Există o "hartă a porturilor" care poate sau nu să facă parte din harta memoriei.

Singura deosebire esenţială faţă de locaţiile de memorie este *legătura fizică pe care porturile o asigură cu exteriorul*; pentru microprocesor, de multe ori, această legătură fizică este transparentă şi nesemnificativă

"Magistrală": un set de conexiuni fizice între blocuri prin care informaţia care circulă are o semnificaţie prestabilită. Sistemele la care ne referim au o magistrală unică, ce le caracterizează; din punct de vedere funcţional, există trei componente ale acestei magistrale:

1. Magistrala de date, bidirecţională, permite circulaţia datelor (operanzi/rezultate), a instrucţiunilor şi chiar a adreselor.

2. Magistrala de adrese, unidirecţională, permite microproceso-rului să localizeze informaţia în MEMORIE sau în DISPOZITIVELE DE INTRARE/IEŞIRE; deci pe această magistrală circulă numai adrese.

3. Magistrala de control permite circulaţia, bidirecţională, a semnalelor de comandă şi control de la/la microprocesor, în calitatea sa de Unitate centrală.

B. Componenta software: o serie de programe organizate în moduri specifice.

Două categorii de software:

1. Sistemul de operare: totalitatea programelor care permit utilizatorului accesul complet la resursele sistemului (exemple: MS-DOS, UNIX etc.). Poate fi: rezident (permanent în memoria internă) sau încărcabil dintr-o memorie externă (operație denumită "bootstrap").

2. Software-ul utilizatorului, alcătuit din totalitatea programelor folosite pentru îndeplinirea unor sarcini specifice.

Caracteristicile arhitecturii "Von Neumann":

- **Microprocesorul constituie Unitatea centrală de prelucrare a unui sistem având schema bloc funcțională din figură. El concentrează şi funcția de prelucrare şi pe cea de comandă.**

- **Toate celelalte componente ale sistemului nu au putere de decizie. Memoria nu controlează şi nici nu e necesar să controleze semnificația informației pe care o deține şi modul în care este organizată logic.**

- **Legătura dintre blocuri este asigurată de o magistrală unică cu trei componente funcționale; pe magistrala de date circulă toate tipurile de informații.**

- **Funcționarea sistemului se face pe baza unor programe alcătuite din secvențe de instrucțiuni. Acestea sunt citite din memorie de către microprocesor, recunoscute şi apoi executate.**

Arhitectură:
totalitatea atributelor sistemului (în cazul de față, microprocesorul) care sunt disponibile ("vizibile") utilizatorului (ca, de pildă: registrele, modurile de adresare, tipurile de transferuri de date, modul de organizare logică a memoriei, tehnicile de intrare/ieşire, setul de instrucțiuni etc).

1.2. Definiții; microprocesoare CISC şi RISC

• **Microprocesor, microcalculator, minicalculator**

Asemănarea: caracteristicile globale ale atributelor de arhitectură.

Deosebirile între ultimele două: resurse (memorie internă şi externă, echipamente periferice) şi performanțe (viteză de prelucrare, cost, număr de componente, gabarit).

• **Definiția microprocesorului ca Unitate centrală de prelucrare; am presupus implicit că sistemul din care face parte este un micro(mini)calculator, deci un sistem de calcul. Putem extinde însă noțiunea şi asupra sistemelor de comandă şi control (de tip "controler"), ceea ce măreşte aria de cuprindere a noțiunii de microprocesor**

• **Noțiunea de "logică programată". Sistemele cu logică programată nu înseamnă, în mod automat, sisteme cu microprocesor. Microprocesorul poate constitui una dintre modalitățile de proiectare a sistemelor cu logică programată. Nu se va face confuzia "sistem cu logică programată cu microprocesor" ≠ "sistem microprogramat"**

• **Clasificări ale noțiunii de microprocesor:**

a) După lățimea magistralei de date: microprocesoare pe 8, 16, 32 sau pe 64 de biți

b) După tipul de sarcini eficient realizabile:

- **microprocesoare de uz general (µPUG), nespecializate;**
- **microprocesoare specializate, ca de pildă:**
  - **procesoare de intrare/ieşire, pentru conversații complexe între microcalculator şi lumea exterioară; exemplu: Intel 8089;**
  - **coprocesoare aritmetice, specializate pentru funcții aritmetice de utilitate generală (exponențiale, trigonometrice etc); exemplu: Intel 80387;**
  - **procesoare digitale de semnal, specializate pentru algoritmii specifici prelucrării semnalelor (FFT, produse de corelație, filtre digitale, calcul matriceal etc.); exemplu: Texas Instruments TMS 320**

c) După principiile de bază ale arhitecturii care guvernează funcționarea:

- **procesoare cu set complex de instrucțiuni (CISC) numite microprocesoare "standard" sau simplu "microprocesoare";**
- **procesoare cu set redus de instrucțiuni (RISC)**

1.3. Reprezentarea informației în sistemele digitale

- **bit** (prescurtat **b**) pentru o cifră binară 0 sau 1;
- **nibble** (prescurtat **n**) pentru o înşiruire de 4 biți;
- **byte** sau octet (prescurtat **B**) pentru o înşiruire de 8 biți;
- **cuvânt** sau **word** (prescurtat **w**) pentru o înşiruire de 2 octeți;
- **cuvânt dublu** sau **double word** (prescurtat **dw**) pentru o înşiruire de 4 octeți;
- prefixele:
  - **k** pentru $2^{10} \approx 10^3$;
  - **M** pentru $2^{20} \approx 10^6$;
  - **G** pentru $2^{30} \approx 10^9$;
  - **T** pentru $2^{40} \approx 10^{12}$

1.3.1 Reprezentarea internă

a) Reprezentarea programelor

- Fiecare instrucțiune este reprezentată în memorie de un **cod binar**.
- **Formatul instrucțiunilor**, adică totalitatea cifrelor binare necesare pentru codificare, are, de regulă, drept cuantă de informație, octetul. Pentru fiecare instrucțiune există un număr prestabilit de octeți cu care e codificată (de pildă, pentru Intel 8086, este între 1 şi 6 octeți)

b) Reprezentarea numerelor

1) Reprezentarea întregilor fără semn în "binar natural": este reprezentarea uzuală, "naturală" a numerelor binare:

$$\text{număr binar cu 8 cifre} = \sum_{i=0}^{7} b_i 2^i \quad \text{cu } b_i \in \{0, 1\}$$

$$\text{număr binar cu 8 cifre} = b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$$

b_0 : lsb

b_7 : msb

2) Reprezentarea întregilor cu semn în "binar natural":

Semnul numărului este reprezentat de msb cu următoarea convenție:

msb = 0 semnifică număr pozitiv;

msb = 1 semnifică număr negativ

Pentru un număr fără semn cu 8 biți, plaja numerelor reprezentabile acoperă 256 de poziții, între 0 şi 255, în zecimal.

Pentru un număr cu semn, plaja numerelor reprezentabile acoperă tot 256 de poziții, dar în intervalul -128 ÷ +127, presupunând 0 număr pozitiv.

Convenţii de reprezentare:

Tipul reprezentării	+ 5	- 5
"mărime şi semn"	**00000101**	**10000101**
"complement față de 1"	**00000101**	**11111010**
"complement față de 2"	**00000101**	**11111011**

Regulile de reprezentare în aceste trei convenții:

- Numerele pozitive se reprezintă identic.

- În "mărime şi semn", numerele negative diferă de cele pozitive numai prin bitul de semn.

- În "complement față de 1", mărimea numărului negativ se obține din reprezentarea precedentă prin complementare bit cu bit; convenția pentru bitul de semn se păstrează.

- În "complement față de 2", mărimea numărului negativ se obține din reprezentarea precedentă prin adunarea unei cifre binare 1 la lsb.

“Fanioane”

“Transportul” care apare între rangul unui număr binar şi cel imediat superior în operaţiile aritmetice (la scădere, îl vom numi "împrumut"): C (de la "carry")

“Depăşirea”: O (de la “overflow”). După cum numărul are semn sau nu, se poate scrie că:

$$O = C_{msb\nearrow} \quad SAU \quad C_{msb-1\nearrow\ msb}$$

Extinderea numerelor cu semn reprezentate în complement faţă de 2, de la 8 la 16 biţi

	Reprezentare cu 8 biţi	Reprezentare cu 16 biţi
+ 1	00000001	0**0000000**00000001
- 1	11111111	1**1111111**11111111

Regulile de "extindere a numerelor cu semn, în complement faţă de 2":
- Bitul de semn rămâne pe poziţia cea mai semnificativă.
- Partea care reprezintă mărimea numărului va ocupa poziţiile cele mai puţin semnificative ale numărului extins.
- Restul poziţiilor din numărul extins se completează cu cifre binare identice cu cea care reprezintă semnul (0 pentru numere pozitive şi 1 pentru numere negative).

3) Reprezentarea întregilor în "zecimal codificat binar" (ZCB): se reprezintă fiecare cifră zecimală separat, în binar natural, cu un nibble

Microprocesoarele folosesc două tipuri de reprezentări ZCB:

- Reprezentarea "ZCB împachetat" în care fiecare octet din memorie cuprinde câte două cifre zecimale, una pe nibble-ul mai puțin semnificativ şi cealaltă pe nibble-ul superior. Plaja de numere zecimale acoperită de o reprezentare cu 8 biți se micşorează de la 256 la 100 de numere: 0 ÷ 99.

- Reprezentarea "ZCB neîmpachetat" în care fiecare octet cuprinde o singură cifră zecimală pe nibble-ul mai puțin semnificativ. Restul cifrelor binare se completează cu 0.

4) Reprezentarea numerelor cu zecimale "cu virgulă fixă": se foloseşte principiul de a aloca un număr fix, prestabilit, de cifre binare pentru a reprezenta partea întreagă şi respectiv partea zecimală a unui număr.

Se poate folosi fie reprezentarea în binar natural fie în ZCB. Pentru partea întreagă se foloseşte regula de reprezentare a numerelor întregi cu semn, iar pentru partea zecimală regula de reprezentare a întregilor fără semn. (Apar: "trunchierea" sau "rotunjirea" numărului).

Modul de reprezentare foloseşte următoarele convenţii:

- **Se rezervă un şir de biţi cu care se exprimă numărul total de cifre ale numărului care urmează să fie reprezentat.**
- **Se rezervă, apoi, un şir de biţi în care se înscrie numărul de zecimale cu care se va reprezenta numărul.**
- **Urmează reprezentarea propriu-zisă a numărului înşiruind reprezentările pentru partea întreagă şi cea zecimală fără o altă delimitare explicită între ele.**

Un exemplu în ZCB împachetat:

.... 0100 0010 0000 1001 0110 0001 0101 ....

numărul reprezentat este + 96.15

5) Reprezentarea numerelor cu zecimale "cu virgulă mobilă"; reprezentare normalizată

Două entități informaționale: "mantisa" **M** şi "exponentul" **EXP**:

$$\text{număr binar} = M * 2^{EXP}$$

$$2^{-1} \leq M < 2^{0}$$

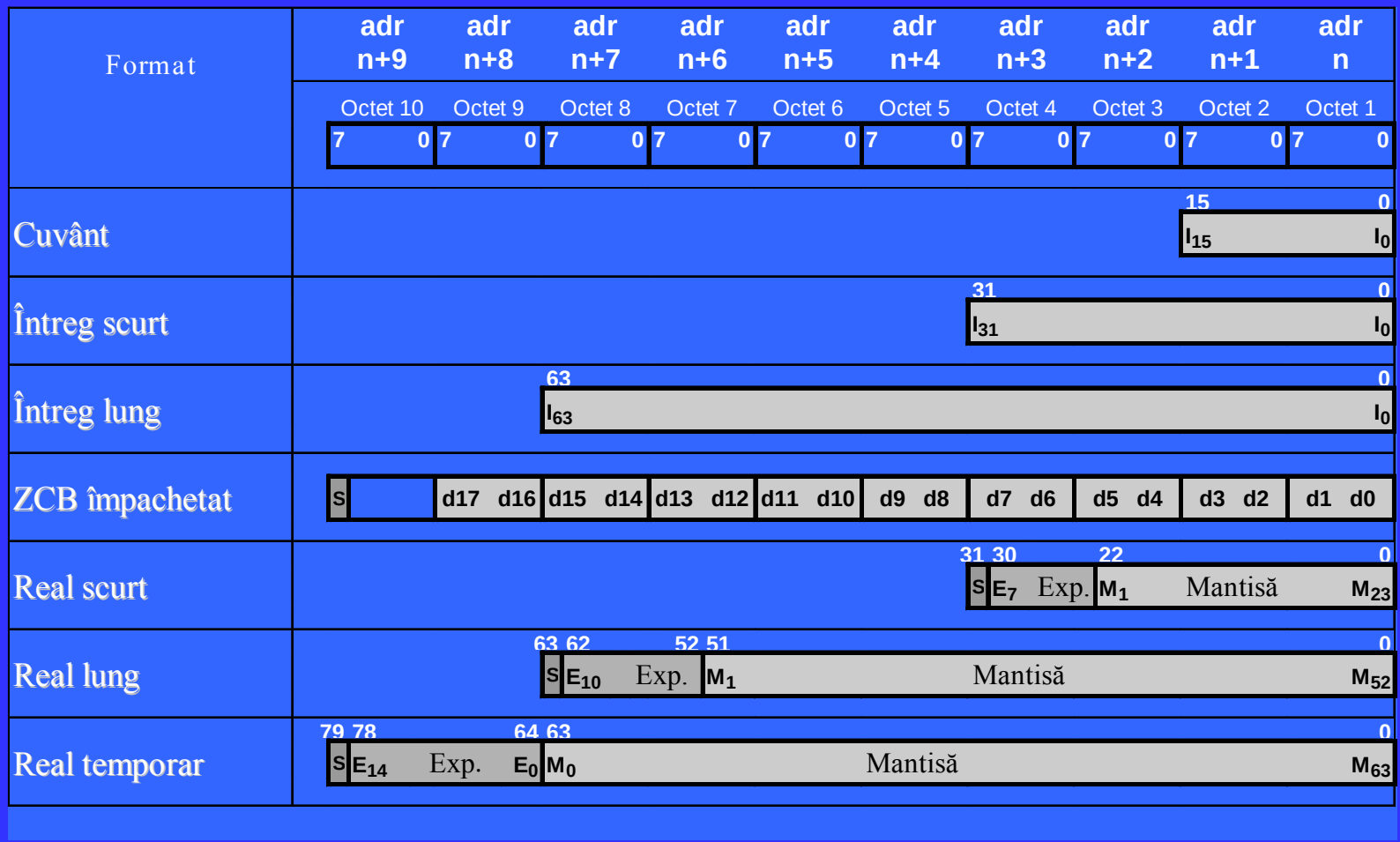
Un exemplu:

$$b_{31}.....b_{24}b_{23}.....b_0 ,$$

în care: - $b_{31} \div b_{24}$ reprezintă exponentul, având semnul în poziția b_{31}.
- $b_{23} \div b_0$ reprezintă mantisa cu semnul la b_{23}.

Plaja numerelor reprezentabile în acest fel: $M * 2^{\pm 128}$

Formatul datelor pentru coprocesoarele aritmetice



Format	Gamă	Precizie	Mod de interpretare
Cuvânt	10^4	16 biți	$I_{15}\ldots I_0$
Întreg scurt	10^9	32 biți	$I_{31}\ldots I_0$
Întreg lung	10^{18}	64 biți	$I_{63}\ldots I_0$
ZCB împachetat	$\pm 10^{\pm 18}$	18 cifre	$(-1)^S$ (d17… d0)
Real scurt	$\pm 10^{\pm 38}$	24 biți	$(-1)^S$ $(2^{Exp-127})(M_0.M_1 \ldots M_{23})$, M_0 implicit 1
Real lung	$\pm 10^{\pm 308}$	53 biți	$(-1)^S$ $(2^{Exp-1023})(M_0.M_1 \ldots M_{52})$, M_0 implicit 1
Real temporar	$\pm 10^{\pm 4932}$	64 biți	$(-1)^S$ $(2^{Exp-16383})(M_0.M_1 \ldots M_{63})$

Semn	Exponent	Mantisă	Valoare
x	11 … 11	1.xx … xx (cel puțin un x este 1)	Non-validă
x	11 … 11	1.00 … 00	Infinit
x	00 … 00	0. xx … xx	Nestandard (prea mică)
x	00 … 00	0.00 … 00	Zero

c) Reprezentarea datelor alfanumerice

Vom înțelege prin "date alfanumerice" sau "caractere" oricare dintre semnele care pot fi tipărite de la tastatura unui calculator.

"Seturi de caractere": grupuri minime de simboluri considerate suficiente pentru a asigura o editare cât mai completă a unui text.

Pentru fiecare caracter se va folosi o reprezentare binară, un cod, cu care caracterul (dintr-un set prestabilit) este în relație biunivocă.

Standardul "ASCII" (de la "American Standard Code for Information Interchange") cu care se codifică următorul set de caractere:

- **26 de litere mari ale alfabetului latin;**
- **26 de litere minuscule corespunzătoare;**
- **10 simboluri numerice: 0 ÷ 9;**
- **20 de simboluri speciale adiționale: +, -, (,), [,], {, }, * , # , $ etc.**

Codificare cu 7 biți;

msb : "bit de paritate". Convenția folosită este următoarea:

msb = 0 dacă codul are un număr par de cifre binare 1;
msb = 1 dacă codul are un număr impar de cifre binare 1

De exemplu:

A = 01000001;
B = 01000010;
C = 11000011 etc.

Bitul de paritate → fanion dedicat (P)

1.3.2 Reprezentarea externă

Reprezentarea externă se referă la modul în care informația prelucrată de un microcalculator apare utilizatorului (programatorului)

a) Pentru codurile instrucţiunilor se vor folosi abrevierile sugestive pe care, de regulă, fabricantul le impune şi pe care limbajul de asamblare le foloseşte ca atare: "mnemonice".

b) Pentru numere se utilizează mai multe tipuri de reprezentări:

- **Reprezentarea binară, imagine fidelă a conținutului locațiilor de stocare a informațiilor.**
- **Reprezentarea octală, care transformă numerele binare în baza de numerație 8.**
- **Reprezentarea hexazecimală : un simbol reprezentând o cifră în baza de numerație 16 înlocuieşte 4 cifre binare. Caracterele folosite sunt cifrele zecimale 0 ÷ 9 şi literele A ÷ F. Vom folosi convenția de a utiliza litera H ca sufix pentru numerele reprezentate în hexazecimal (de pildă 1199H).**

c) Pentru caractere se vor folosi chiar simbolurile cu care ele sunt individualizate. Programele utilitare folosite pentru examinarea conţinutului locațiilor de stocare a informațiilor fac conversia ASCII → simbol caracter dacă programatorul stabileşte că semnificația informației vizate impune această conversie

1.4. Convenţii pentru notaţii

1. Neterminali:

r	un registru oarecare;
r8	un registru de 8 biţi;
r16	un registru de 16 biţi;
r_i , r_j	registre individualizate, diferite;

mem	o locaţie de memorie oarecare (sau mai multe locaţii succesive);
mem8	o locaţie de memorie de un octet;
mem16	o locaţie de memorie de 16 biţi (pot fi două locaţii succesive dacă formatul este octetul);
mem32	o locaţie de memorie de 32 de biţi (pot fi patru locaţii succesive dacă formatul este octetul);
mem_i	o locaţie de memorie individualizată (în scopul de a o deosebi de alte locaţii de memorie);

adr	o adresă oarecare;
adr16	o adresă pe 16 biți;
adr24	o adresă pe 24 de biți;
adr_i	o adresă individualizată (în scopul de a o deosebi de alte adrese);
(r)	conținutul unui registru oarecare;
(r_i, r_j)	conținutul a două registre concatenate;
$(r)_l$	conținutul jumătății inferioare (mai puțin semnificativă) a unui registru;
$(r)_h$	conținutul jumătății superioare (mai semnificativă) a unui registru;
((r))	conținutul unei locații de memorie a cărei adresă se află într-un registru (adresare indirectă);
(mem)	conținutul unei locații de memorie oarecare;
adr_l	jumătatea inferioară a unei adrese;
adr_h	jumătatea superioară a unei adrese;
data	un operand oarecare;
data8	un operand de 8 biți;
data16	un operand de 16 biți;
disp8	un deplasament pe 8 biți;
disp16	un deplasament pe 16 biți;
port	un port de intrare/ieşire oarecare

2. Terminali:

R1, R2, A, AX, BP, A6, Dn, An	nume de registre;
(R1)	conținutul registrului R1;
(R1, R2)	conținutul perechii de registre R1 şi R2;
((R1))	conținutul locației de memorie a cărei adresă se află în registrul R1;
MEM, MEM1	nume de locații de memorie;
ADR, ADRn	nume de adrese

3. Operatori

←	atribuire;
↑	concatenare;
not	complementare (negație);
∀	operația logică SAU;
&	operația logică }I;
⊕	operația logică SAU EXCLUSIV;
+	adunare;
-	scădere;
*	înmulțire;
DIV	împărțire între numere întregi;
MOD	restul împărțirii între numere întregi

4. Alte simboluri

[ ]	încadrează elemente de sintaxă opționale;
|	delimitează elemente de sintaxă alternative