1 Structura unui microprocesor Definiții

Sunt prezentate câteva definiții strict necesare pentru înțelegerea noțiunilor ce vor urma. De asemenea, este prezentată arhitectura foarte generală a unui sistem de calcul, care nu necesită multe cunoştințe pentru a fi bine înțeleasă. Pornind de la aceasta, vor fi dezvoltate arhitecturi reale foarte diverse şi complicate.

1.1 Câteva definiții

Vom defini cuvintele ce vor fi utilizate foarte frecvent în lucrare, care sunt de fapt larg folosite în informatică. Chiar dacă definițiile nu coincid total cu cele ale Academiei, au meritul de a fi simple şi clare.

Circuit integrat - componentă electronică de mici dimensiuni, (cu suprafața măsurată în milimetri pătrați) ce conține un număr de tranzistoare elementare interconectate; el poate avea funcții cablate sau programate. Circuitele integrate sunt clasificate în mai multe categorii, în funcție de densitatea de integrare: circuite integrate simple, circuite integrate pe scară medie (MSI - *Medium Scale Integration*), circuite integrate pe scară largă (LSI - *Large Scale Integration*), circuite integrate pe scară foarte largă (VLSI - *Verry Large Scale Integration*), etc. După dezvoltarea explozivă a circuitelor integrate în ultimii ani, aceste noțiuni au început să devină desuete.

Unitate centrală sau **Unitate centrală de procesare** este acea componentă a unui calculator care realizează prelucrarea datelor pe baza unui program şi controlul întregului sistem. Se utilizează frecvent notația CPU (*Central Processing Unit*) ceea ce desemnează de data aceasta în mod exclusiv un microprocesor.

Microprocesor - este un circuit integrat (LSI, VLSI) cu mii de tranzistoare în structură care realizează funcțiile unității centrale dintr-un calculator.

Microcalculator - sistem de calcul în care unitatea centrală este un microprocesor; un microcalculator mai cuprinde: blocuri de memorie, circuite pentru transferul informației (porturi de intrare - ieşire) şi dispozitive periferice - tastatură, monitor, unitate de discuri, imprimantă etc.

Hardware - totalitatea componentelor materiale ale unui sistem de calcul (structuri mecanice, cabluri, cutii, circuite, etc.).

Software - totalitatea componentelor imateriale (programe de sistem şi de aplicaţii) cu care este dotat un sistem de calcul. Pentru programele complexe, nu există nici o metodă de verificare care să garanteze că este fără erori de concepţie. Doar utilizarea îndelungată în practică a unui program poate duce la detectarea şi eliminarea totală a erorilor.

Sistem de operare - totalitatea programelor care permit utilizatorului accesul deplin la toate resursele sistemului. Conţine programe cu denumiri specifice după funcţiile realizate, care asigură accesul la echipamentele periferice (tastatură, monitor, unităţi externe de memorie etc.) organizarea informaţiei sub formă de fişiere şi o gamă largă de operaţii asupra acestora (deschiderea, închiderea, transferul, afişarea, crearea, ştergerea, modificarea şi altele). Cele mai răspândite sisteme de operare sunt MS-DOS, WINDOWS şi UNIX.

Noţiunea de memorie. Nu se poate înţelege funcţionarea unui sistem programabil, ca microprocesorul de exemplu, fără a stăpâni noţiunea de memorie.

Să considerăm, de exemplu, o comodă cu mai multe sertare suprapuse, pe care le numerotăm. Numărul fiecărui sertar este ceea ce numim în general *adresă* iar sertarul este în informatică *locaţie de memorie*. Pentru sistemul de calcul, memoria este un şir finit de locaţii numerotate (fig.1).

Celule (locaţii) de memorie	Adrese
1 1 1 1 0 0 0 0	0011
0 0 0 0 1 1 1 1	0010
1 1 0 0 0 0 1 1	0009
0 0 1 1 0 0 1 1	0008
1 1 0 1 1 1 0 1	0007
1 1 0 1 0 1 1 1	0006
1 1 0 0 0 1 0 1	0005
0 1 1 1 0 1 1 1	0004
0 1 0 1 1 1 1 1	0003
0 0 0 0 0 0 0 0	0002
0 0 0 1 1 1 0 1	0001
0 1 0 1 0 1 0 1	0000

Fig. 1. Reprezentarea memoriei (locaţii de 8 biţi)

O locaţie este definită de două entităţi informaţionale: conţinutul şi adresa.

Conţinutul este un şir de cifre binare *0* sau *1* (**Bi**nary Digi**t** = bit) care poate reprezenta *o dată* (un număr sau un caracter în cod binar, o stare, etc.) sau o *comandă* (instrucţiune). Numărul de cifre binare dintr-o locaţie reprezintă *dimensiunea locaţiei* (*formatul memoriei)* (pentru 8 biţi se utilizează denumirea "octet" iar pentru 16 biţi - "cuvânt").

Adresa este numărul de ordine al unei locații de memorie; adresa permite identificarea fiecărei locații în şirul ordonat de locații ce alcătuiesc memoria unui sistem de calcul.

Structura memoriei organizată pe locații, cu delimitarea anumitor zone este numită "harta memoriei".

Se pot evidenția două astfel de zone:

- Memoria de date (locațiile conțin "date");
- Memoria de programe (locațiile conțin instrucțiuni codificate).

Instrucțiunea reprezintă cea mai simplă operație (comandă) pe care o poate transmite programatorul către o unitate centrală (care poate fi un microprocesor). Unitatea centrală poate recunoaşte şi executa numai instrucțiunile codificate pentru care a fost construită; acestea formează **setul de instrucțiuni** caracteristic unității centrale.

Un şir de instrucțiuni, organizate logic după un algoritm, formează un program; prin intermediul programului, utilizatorul transmite sistemului de calcul o anumită *sarcină privind prelucrarea datelor* (**task**). Noțiunea de task este mai largă decât cea de program: există sarcini pentru a căror îndeplinire sunt necesare mai multe programe.

Magistrala este un ansamblu de conexiuni electrice prin care circulă *informație de acelaşi tip* având ca suport semnale electrice; în funcție de tipul informației, magistralele sunt de trei categorii: de date, de adrese şi de control. O caracteristică de bază este *dimensiunea magistralei*, adică numărul liniilor de conectare; avem astfel magistrale de 8 biți (cu 8 linii de conectare), magistrale de 16 biți (cu 16 linii de conectare), etc. Dimensiunea fiecărei magistrale este determinată de structura unității centrale şi determină la rândul său structura memoriei (numărul de biți pe locație) şi a porturilor de intrare / ieşire.

Prin cuvântul *magistrală* se înțelege de regulă şi ansamblul de circuite electronice (amplificatoare uni- sau bi-direcționale) care sporesc puterea semnalelor electrice (pentru *fan-out* mai mare) şi aduc nivelul (tensiunea) la valoarea standard.

Magistralele unidirecționale pot transmite informația într-un singur sens iar cele bidirecționale, în ambele sensuri (sensul de transmisie este controlat de unitatea centrală). Magistrala de adrese este unidirecțională (de la unitatea centrală spre sistem) iar magistralele de date şi de control sunt bidirecționale.

- La o magistrală se cuplează în paralel mai multe blocuri de acelaşi tip sau de tipuri diferite; acestea devin active succesiv sub comanda unității centrale; astfel UC coordonează toate transferurile din sistem.

1.2 Structura sistemelor de calcul

Orice sistem numeric de prelucrare a datelor, primeşte din exterior date binare pe care le prelucrează pe baza unor programe de lucru existente în memorie. Rezultatele prelucrării sunt transmise către exterior prin unități specializate.

Principalele componente **hardware** sunt vizibile în structura din figura 2, care este valabilă atât pentru sisteme simple (calculator de buzunar), cât şi pentru sisteme complexe de calcul.

- UCP - unitate centrală de procesare (microprocesor în cazul în care sistemul de calcul este un microcalculator). Ea conține o unitate centrală de comandă - UCC şi o unitate aritmetică şi logică - UAL;
- Memoria;
- Unități de intrare / ieşire - I/E;

1. UCC are rol de prelucrare a datelor şi de coordonare a întregului sistem. Prin intermediul magistralelor extrage succesiv din memorie instrucțiuni, le interpretează şi generează semnale de comandă către unitățile de prelucrare a datelor. Operațiilor aritmetice şi logice sunt efectuate de unitatea UAL.

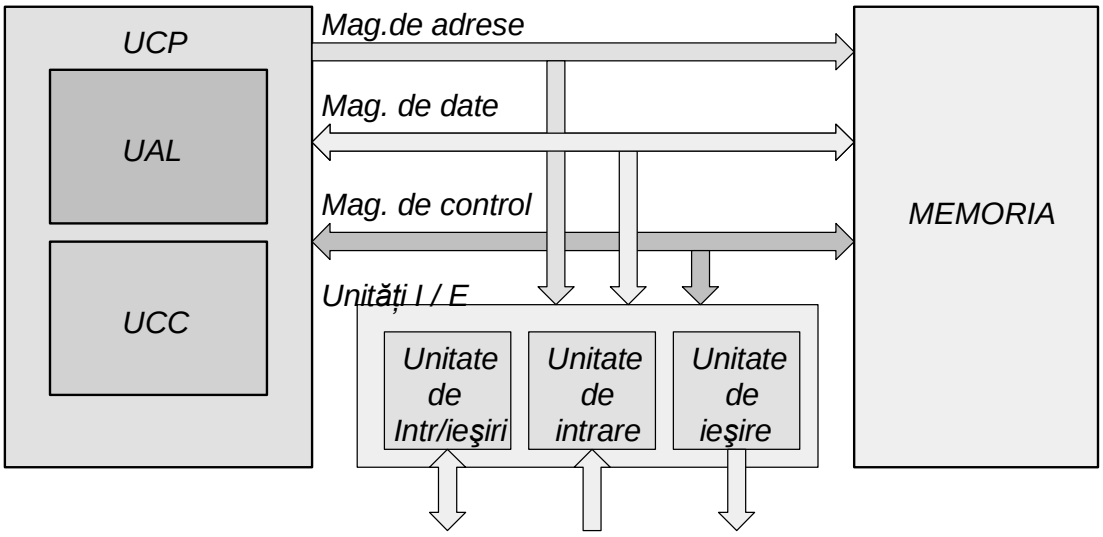

Figura 2. Structura unui sistem de calcul

2. Memoria păstrează programe şi date. Programele sunt stocate în memorie sub formă de şiruri de instrucțiuni iar datele sunt operanzi sau rezultate ale prelucrărilor. Datele sunt *numere binare* (şiruri de "0" şi "1") iar instrucțiunile sunt *comenzi de prelucrare,* care deşi se prezintă tot ca

şiruri de cifre binare, vor fi numite*coduri*. Anumite zone de memorie vor fi utilizate pentru programe iar altele, pentru date.

Controlul asupra conţinutului memoriei revine exclusiv unităţii centrale; blocul de memorie nu are nici un control asupra semnificaţiei informaţiei pe care o conţine.

3. Unităţile sau dispozitivele de intrare/ieşire (notate I/E sau Input/Output=I/O) realizează legătura dintre sistemul de calcul şi lumea exterioară. O unitate elementară de tip I/O este numită în mod curent **port de intrare/ieşire**. Între porturi şi locaţiile de memorie există asemănări dar şi deosebiri fundamentale.

Ca şi locaţiile de memorie, porturile sunt adresabile (fiecare port are o adresă proprie de identificare); operaţiile pe care unitatea centrală le poate efectua cu porturile sunt: "scriere port" - transfer de date la port şi "citire port" - transfer de date de la port la UCC (aceleaşi operaţii se efectuează şi cu locaţiile de memorie).

Deosebirea esenţială faţă de locaţiile de memorie este *legătura pe care porturile o realizează cu echipamentele externe* (periferice): tastatură, monitor, unităţi de memorie externe (disc, bandă), imprimantă, alte echipamente specifice unor procese industriale (traductoare, echipamente de forţă, de semnalizare etc.). Această funcţie a porturilor de "puncte de frontieră" determină şi alte deosebiri faţă de locaţiile de memorie:

- Informaţia primită prin intermediul porturilor este tot timpul "de actualitate" - informaţie nouă pentru UCC.
- Informaţia "scrisă" într-un port nu este stocată în mod pasiv, ca într-o locaţie de memorie ci are efect asupra unui periferic (aprinde luminile oraşului, declanşază sistemul de propulsie al unei rachete cosmice).
- Operaţiile cu porturile sunt realizate de UCC prin instrucţiuni specifice, de tip IN (*Input*) sau OUT (*Output*), altele decât cele cu memoria, MOV, PUSH, POP etc.

Din punct de vedere **software**, un sistem de calcul dispune de două componente fundamentale.

1. *Sistemul de operare* - totalitatea programelor care asigură accesul utilizatorului la resursele sistemului (MS-DOS, WINDOWS, UNIX etc.). Programele, cu denumiri specifice după funcţiile realizate, asigură accesul şi controlul dispozitivelor periferice, organizarea informaţiei în memoria internă, încărcarea şi execuţia programelor de aplicaţii etc.

Fără un sistem de operare, calculatorul este o cutie inutilă.

2. Programele de aplicaţii - introduse de utilizator în scopul rezolvării sarcinilor proprii. Acestea au o varietate practic nelimitată.

1.3 Arhitectura de bază a unui microprocesor

Acest capitol se referă la structura internă a unui microprocesor "standard", unitățile interne fiind prezente sub formă mai simplă sau mai complexă la toate tipurile de microprocesoare.

Deşi există o mare varietate de microprocesoare, produse de diferite firme, cu multe deosebiri în structura şi tehnologia lor de fabricație, toate au o schemă structurală comună, rezultată din operațiile de bază pe care le efectuează. Caracteristicile structurale şi funcționale comune rezultă din filosofia proiectării microprocesoarelor, ca instrumente complexe pentru realizarea unor sisteme numerice flexibile, rapide, puternice şi la un preț de cost deosebit de avantajos.

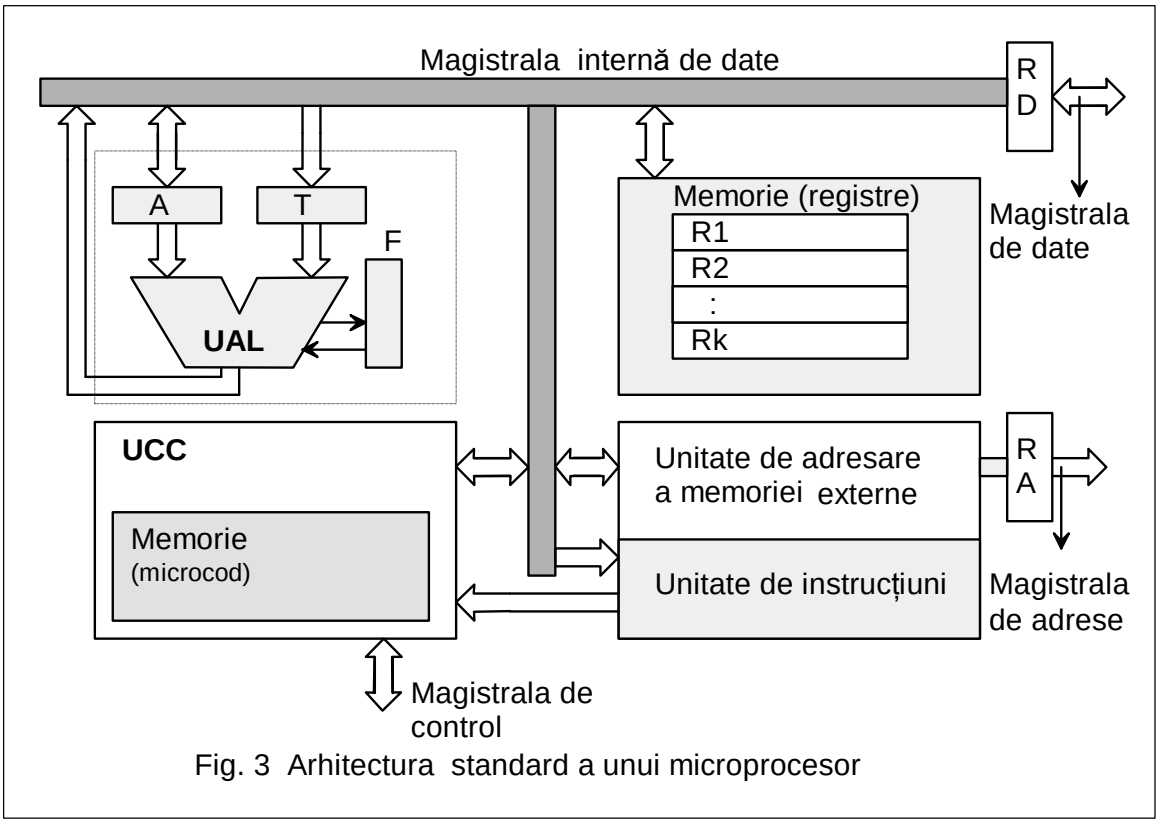

Fig. 3 Arhitectura standard a unui microprocesor

Structura de bază conține 5 unități cu funcții specifice: unitatea de comandă şi control - UCC, unitatea aritmetică şi logică - UAL, memoria internă (formată din registre = locații), unitatea de adresare a memoriei externe şi unitatea de instrucțiuni.

1.3.1. Memoria internă

Este formată din registre cu dimensiunea (număr de celule de memorie) egală cu cea a magistralei de date.

Registrele, notate R1, R2, . . . ,Rk, sunt numite "de uz general" deoarece, prin intermediul instrucțiunilor, în acestea se pot stoca temporar *date* de orice tip (numerice, alfanumerice, date de intrare - ieşire, adrese, instrucțiuni etc.). În mod frecvent, în registrele de uz general se stochează operanzi şi rezultate intermediare ale prelucrărilor numerice; registrele fiind conectate la magistrala internă de date, transferul datelor este rapid şi facil.

Setul de registre de uz general constituie un atribut de arhitectură deoarece aceste registre sunt la dispoziția programatorului; acesta le utilizează prin intermediul instrucțiunilor. Pentru a fi uşor de utilizat, registrele au un nume format din una sau mai multe litere: A, B, C, . . , AX, BX, . . , EAX, EDI, EDS etc.

1.3.2. Registrul de date şi registrul de adrese

Două registre, RA şi RD prin care se realizează conectarea cu magistralele externe de date şi adrese, au rol cu totul special în structură.

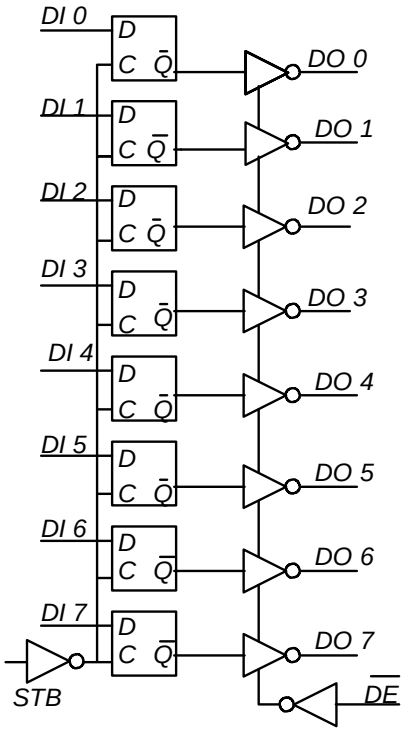

Fig. 4. Schema unui registru de 8 biți

Registrul de date RD memorează temporar datele magistralei pe care o deserveşte. Datele ce se transferă spre exterior sunt menținute pe magistrală până când dispozitivele externe (de regulă mai lente decât procesorul) le recepționează în registrele proprii. Datele transferate prin magistrală spre microprocesor se consideră recepționate după înscrierea în RD, care fiind conectat la magistrala internă de date devine sursă de date pentru blocurile interne.

Similar, registrul de adrese RA are rolul de a menține o adresă pe magistrala externă de adrese un timp suficient pentru ca memoria şi porturile să o poată înregistra pentru realizarea funcției de selecție.

Cele două registre, RA şi RD sunt invizibile pentru utilizator.

În figura 4 este prezentată schema de principiu a unui registru de 8 biți care poate fi utilizat ca tampon pentru magistrala de adrese (RA). Pentru o magistrală de 16 biți, se utilizează două registre de 8 biți.

Registrul din figură are şi rol de amplificator de magistrală, asigurând un *fan - out* de 20 intrări TTL. Informația de la intrările DI apare al ieşirile DO pe nivelul 1 logic al semnalului *STB* şi este memorată în cele 8 circuite basculante bistabile de tip D. Pentru ca informația să fie disponibilă la ieşiri, este necesar ca semnalul de validare $\overline{DE} = 0$.

1.3.3. Unitatea aritmetică şi logică (UAL)

Acest bloc funcțional execută prelucrarea datelor. Funcțiile realizate de unitate sunt:

- funcții aritmetice: adunare, scădere, înmulțire, împărțire;
- funcții logice: ŞI, SAU, SAU EXCLUSIV, NU şi complement.

Fiecare funcție este activată de o instrucțiune corespunzătoare care furnizează şi operanzii implicați în operație.

Pentru realizarea funcțiilor sale, unitatea aritmetică şi logică utilizează câteva registre speciale care fac parte integrantă din UAL:

- **Acumulatorul**

Registru de uz general care este utilizat de UAL pentru stocarea unuia dintre operanzi şi pentru rezultatul operației; din acest punct de vedere are un rol cu totul special în comparație cu celelalte registre de uz general.

- **Registrul F**

Este registrul fanioanelor de condiții (*Flags*) şi conține celule de memorie independente, cu funcții specifice, pentru înregistrarea unor informații ce rezultă din operațiile aritmetice şi logice (semnul rezultatului, paritatea, existența bitului de transport sau împrumut, depăşirea domeniului şi altele). În ansamblu, indicatorii de condiții exprimă starea unității aritmetice şi logice.

- **Registrul de deplasare**

Este utilizat pentru deplasări spre stânga sau spre dreapta a unui operand. Deplasările se pot face cu unul sau mai mulți biți. Deplasarea spre stânga cu un bit este echivalentă cu înmulțirea cu 2 iar cea spre dreapta, cu împărțirea prin 2.

La microprocesoarele de 8 biți, acumulatorul este folosit şi ca registru de deplasare însă la microprocesoarele evoluate există un registru special cu această funcție, care nu este vizibil pentru programator.

Deplasările spre stânga sau spre dreapta se realizează sub comanda unor instrucțiuni specifice care acționează asupra unui registru de uz general sau unei locații de memorie. Pentru realizarea operației, conținutul registrului sau locației se transferă în registrul de deplasare, se execută deplasarea şi apoi rezultatul se transferă înapoi în registru sau locație. Pentru programator operațiile secundare sunt invizibile.

1.3.4. Unitatea de adresare a memoriei externe

Rolul acestei unități este calcularea adresei unui operand aflat în memoria externă, încărcarea acesteia pe magistrala de adrese şi controlul transferului între memorie şi microprocesor.

În memoria externă se adresează instrucțiuni şi operanzi. Pentru instrucțiuni se utilizează un registru special de adresă, PC (*Program Counter*) - numărător de program sau IP (*Instruction Pointer*) - indicator de instrucțiuni; conținutul său creşte cu o unitate după citirea fiecărui octet.

Pentru adresarea operanzilor (datelor) se utilizează registre de adresare numite "index". Adresa se poate obține direct din registru sau prin adunarea (scăderea) unui deplasament (constantă specificată în instrucțiune). Adresarea datelor se poate face şi direct, prin încărcarea adresei în registrul RA; în acest caz, adresa este furnizată de instrucțiune.

1.3.5. Unitatea de comandă şi control

Coordonează funcționarea tuturor unităților interne pentru execuția operațiilor conținute în mod codificat în instrucțiuni.

Funcțiile unității de comandă sunt:

- Extragerea instrucțiunii din memoria externă.

Se "citeşte" instrucțiunea din zona care conține programul aflat în execuție. Instrucțiunea are două zone de informație: zona de cod, care conține operația caracteristică instrucțiunii şi zona de date (operanzi). Zona de cod se încarcă în registrul de instrucțiuni, aflat în unitatea de instrucțiuni.

- Decodificarea instrucțiunii.

Fiecare instrucțiune are ca efect o succesiune specifică de operații elementare, numite microoperații. Secvența de microoperații este generată de unitatea de comandă pe baza codului instrucțiunii; determinarea acestei secvențe în funcție de cod, este numită "decodificare".

- Execuția propriu-zisă constă în activarea succesivă a unităților interne pentru efectuarea operațiilor din secvența corespunzătoare instrucțiunii.

Codul instrucțiunii permite obținerea tuturor informațiilor necesare execuției operațiilor impuse de instrucțiune:

- numărul de octeți din formatul instrucțiunii;
- tipul operației principale (adunare, scădere, transfer, salt etc.);
- numărul operanzilor implicați în operație;
- adresa fiecărui operand (dacă este operand aflat temporar într-un registru intern, se specifică acest registru; dacă este operand în memorie, se specifică adresa sau cum se obține adresa lui).

Fiecare instrucțiune corespunde unei operații fundamentale, care, în general se realizează în mai multe etape. Execuția unei instrucțiuni este un şir de operații elementare:

- *starea* - corespunde unei perioade de tact (T) şi este durata unei operații elementare (de exemplu, incrementarea unui registru);
- *ciclul maşină* - conține 3 - 5 stări şi corespunde unei etape din execuția unei instrucțiuni (de exemplu, citirea unei locații de memorie, transferul de date între un registru intern şi o locație de memorie etc.); o instrucțiune conține 1 - 5 cicluri maşină, dintre care primul ciclu maşină este de citire memorie (citirea codului instr.).

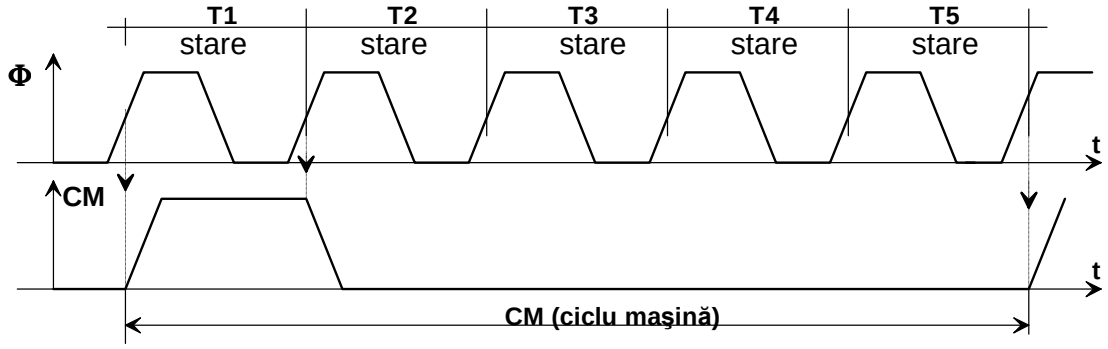

Fig.4 Diagrama semnalului de tact şi delimitarea unui ciclu maşină

În desfăşurarea în timp a unei instrucțiuni, unitatea de comandă şi control selectează şi adresează unitățile interne ale microprocesorului care realizează funcții specifice rolului lor.

Realizarea concretă a unității de comandă şi control (UCC) este specifică fiecărui tip de microprocesor şi determină multe din performanțele sale.

Ca structură, UCC este un automat finit, care funcționează pe baza unui microprogram introdus în procesul de fabricație. Acest microprogram nu poate fi modificat de utilizator şi corespunde setului de instrucțiuni, fiind un interpretor de instrucțiuni. De aceea, setul de instrucțiuni este de asemenea fix pentru fiecare tip de microprocesor.

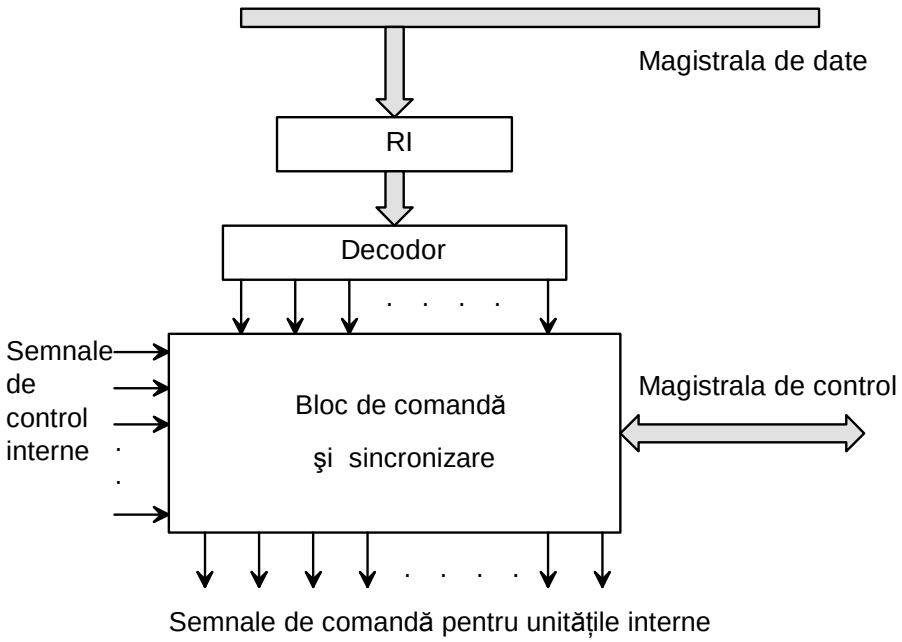

Fig. 5 Structura unității de comandă şi control

În figura de mai sus este prezentată structura generală a unității de comandă şi control. Registrul de instrucțiuni (RI) memorează temporar codul instrucțiunii, care este încărcat din memorie prin intermediul magistralei externe de date. Decodorul identifică instrucțiunea în cadrul setului de instrucțiuni; informația privind instrucțiunea curentă este transferată blocului de comandă.

Blocul de comandă şi sincronizare, pe baza unui microprogram de interpretare, generează semnalele de comandă către unitățile interne de execuție.

1.4 Principiul de funcționare al unui microprocesor

Programul este compus din instrucțiuni care se află în memorie. Citirea instrucțiunilor din memorie se face în sensul crescător al adreselor la care sunt memorate. Activitatea microprocesorului constă, în principal, în execuția instrucțiunilor una câte una, în ordinea în care se află în program. Principalele etape sunt aşadar:

- citirea instrucțiunii din memorie şi stocarea sa într-un registru intern;
- decodarea instrucțiunii, adică identificarea operațiilor conținute sub formă codificată în instrucțiune;

- executarea operațiilor într-o anumită ordine.

Fiecare instrucțiune are o anumită adresă. Pentru citirea instrucțiunii este necesară încărcarea adresei pe magistrala de adrese pentru a se realiza accesul la locația de memorie. Microprocesorul trebuie să dispună în permanență de adresa instrucțiunii curente, din care, prin incrementare obține adresa instrucțiunii următoare. În acest scop se utilizează un registru de adresare, numit, în general, *numărător de program* (PC - *Program Counter*) sau *indicator de instrucțiuni* (IP - *Instruction Pointer*) care conservă în permanență adresa curentă.

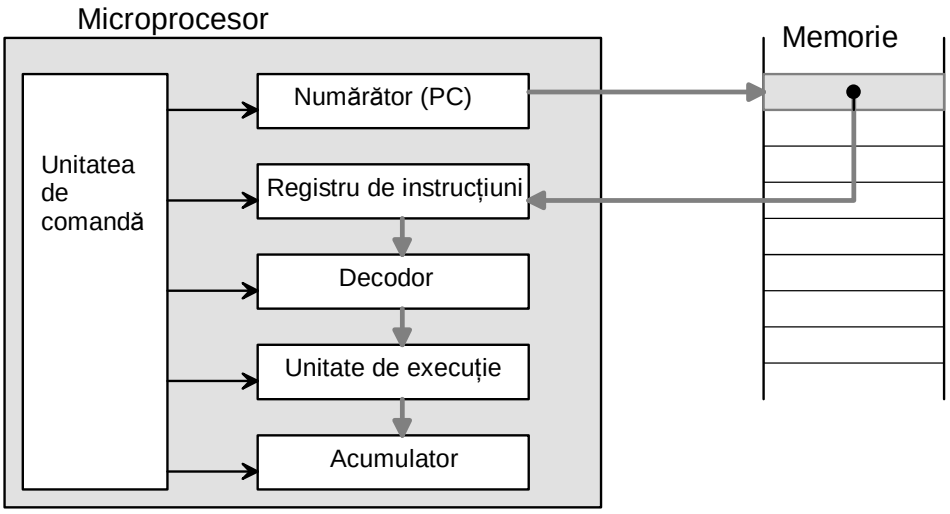

Fig.6 Etapele tratării unei instrucțiuni

Primul octet al unei instrucțiuni (uneori şi al doilea) este totdeauna octet de cod; el este încărcat temporar în registrul RI (registrul de instrucțiuni). Un octet dă o informație directă de 8 biți, adică 16 stări, ceea ce este insuficient pentru setul de instrucțiuni. De aceea, instrucțiunile sunt codificate pe 8 biți, ceea ce permite utilizarea tuturor combinațiilor binare, în total 256; dacă se utilizează doi octeți de cod, numărul maxim de instrucțiuni codificabile este 256 x 256.

Pentru obținerea informației din codul instrucțiunii, este necesară operația de decodare (decodorul are 8 intrări şi 256 ieşiri, câte una pentru fiecare instrucțiune codată) şi transformarea ieşirilor decodorului în comenzi electrice care urmează să activeze unitățile interne care vor executa operațiile prestabilite. De exemplu, o comandă de adunare activează unitatea aritmetică pentru operația de adunare.

Decodarea instrucțiunii furnizează şi informația privind numărul de octeți pe care îl conține. Astfel, unitatea centrală poate separa instrucțiunile din şirul de octeți al programului. Unele instrucțiuni conțin pe lângă unul sau doi octeți de cod şi operanzi. Aceştia sunt transferați în registrele de uz general, RI fiind rezervat exclusiv pentru coduri. După citirea unui octet din program, numărătorul de program (PC) este

incrementat (conținutul creşte cu o unitate: PC+1), fiind astfel pregătit pentru extragerea octetului următor, care se află în memorie la adresa următoare în sens crescător.

Registrul PC se comportă asemănător cu indicatorul kilometric al unui automobil: el indică permanent numărul de kilometri parcurşi şi creşte cu o unitate imediat ce a fost parcurs încă un kilometru.

1.4.1. Ceasul microprocesorului

Circuitele de comandă şi cele care generează secvențele de operare, pun în funcțiune diferite unități interne ale microprocesorului la anumite momente. UCC fiind un automat secvențial cu număr finit de stări, funcționează pe baza unor impulsuri de tact. Acestea sunt produse de un generator electronic pilotat de un cristal de cuarț, care asigură stabilitatea frecvenței la variația tensiunii de alimentare şi a temperaturii. Generatorul de tact poate fi un circuit specializat extern sau poate fi conținut în structura internă a microprocesorului (Fig.7).

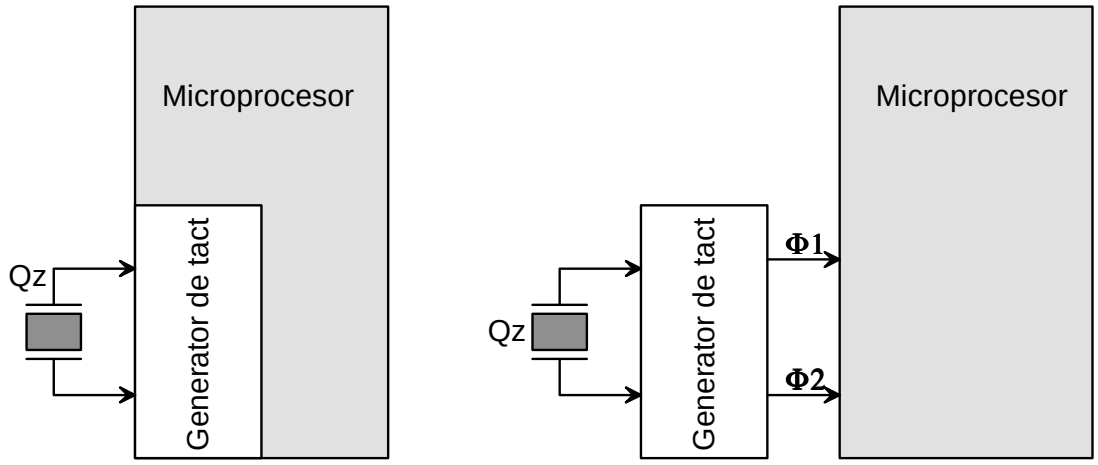

Fig.7 Relația generator de tact - microprocesor

Frecvența impulsurilor de tact determină viteza de execuție a instrucțiunilor. O operație elementară se efectuează într-o singură perioadă de tact. Viteza de operare, exprimată în operații/secundă, va fi:

$$v = f = \frac{1}{T}\ op/\sec .$$

unde f este frecvența (în Herz) iar T este perioada (în secunde).

De exemplu, la o frecvență de 100 MHz, rezultă o viteză de operare de 100 000 000 de operații elementare pe secundă.

Nu se pot măsura foarte riguros performanțele unui microprocesor, deoarece nu există un instrument cu asemenea funcție.

Evaluarea performanțelor se face prin compararea diferitelor tipuri de microprocesoare între ele, pe baza unor criterii unanim recunoscute.

Viteza de operare, de exemplu, se exprimă prin mai mulți parametri:

- Frecvența de tact, exprimată în MHz; cu cât este mai mare, cu atât viteza de execuție creşte. Frecvența nu exprimă însă foarte exact viteza de execuție a instrucțiunilor; în cazul apelării la subrutine, este necesar un interval de timp pentru schimbarea adresei curente. Unui microprocesor îi sunt necesare 10 perioade de tact pentru a executa saltul iar altuia, 20 de perioade. Dacă s-ar dubla frecvența de tact la cel de-al doilea, durata saltului rămâne aceeaşi!
- Numărul de instrucțiuni pe secundă; se exprimă în MIPS (Milioane de Instrucțiuni Pe Secundă). Durata de execuție (exprimată în perioade de tact), diferă foarte mult în funcție de tipul instrucțiunii. Cele mai lungi instrucțiuni sunt cele pentru operații aritmetice în virgulă mobilă; de aceea, unitatea recunoscută este FLOPS (Floating Point Operation per Second).

La compararea calculatoarelor, totul se complică, deoarece pe lângă performanțele unității centrale intervin cele de sistem: viteza de transfer pe magistrale, viteza de operare a discului, a monitorului. Au fost concepute programe test speciale pentru evaluarea performanțelor prin măsurarea timpului de execuție pe diferite calculatoare. Orice program, însă, favorizează o anumită arhitectură în detrimentul alteia.

1.4.2. Exemplu de execuție a unui program simplu

Exemplul următor este real dar corespunde microprocesoarelor de 8 biți, care deşi au performanțe modeste, prezintă avantajul simplității.

Urmărirea etapelor de execuție nu este foarte dificilă iar valoarea pedagogică este considerabilă.

Problema constă în adunarea 5+12, rezultatul fiind încărcat în registrul acumulator.

Vom privi problema din punctul de vedere al programatorului, care are acces doar la registrele interne şi la locațiile de memorie.

Instrucțiunile şi operanzii (5 şi 12) se află în memorie.

Secvența completă de operații este următoarea:

- Prima instrucțiune transferă numărul 5 din memorie în acumulator;
- A doua, comandă adunarea lui 12 la conținutul acumulatorului;
- Rezultatul adunării rămâne în acumulator.

Organigrama programului (schema logică) este dată în fig.8.

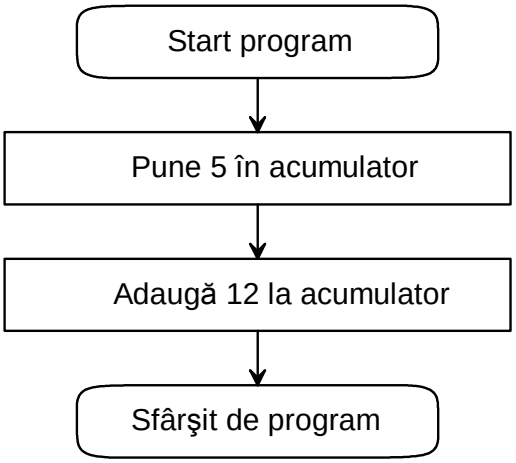

Fig. 8 Schema logică a programului

Programul din memorie va conține aşadar două instrucțiuni, care conțin şi operanzii. Vom presupune că programul începe la adresa 0105.

Pentru a fi executat de microprocesor, programul trebuie scris în cod maşină (în sistem binar sau în hexazecimal) şi încărcat în memorie, octet cu octet începând cu locația de memorie cu adresa 0105.

În setul de instrucțiuni al microprocesorului căutăm instrucțiunile corespunzătoare. Pentru aceasta, este necesar manualul de utilizare; vom folosi manualul de utilizare al microprocesorului Intel 8085, unde găsim toate informațiile necesare, cu privire la instrucțiuni.

```
MOV  A,  d8      ; încarcă în acumulator data de 8 biți
ADD  A,  d8      ; adună data la acumulator, rezultatul în A.
```

Vom înlocui în corpul fiecărei instrucțiuni operanzii generici, cu 5 şi 12.

Instrucțiunea	Descrierea	Cod Hexa	Cod binar
MOV A, 05H	Încarcă în acumulator numărul 05 (05 în Hexa este tot 05)	3E 05	0011 1110 0000 0101
ADD A, 0CH	Adună 12 la acumulator (12 în Hexa este C)	C6 0C	1100 0110 0000 1100

Cei 4 octeți (din ultimele două coloane ale tabelului) definesc în totalitate programul.

Programul se încarcă în memorie, octet cu octet, de la adresa 0105. Celulele de memorie fiind circuite bistabile (binare), în memorie vom găsi programul sub formă de cod binar (ultima coloană).

Conținutul locației	Adresa
0 0 1 1 1 1 1 0	0105
0 0 0 0 0 1 0 1	0106
1 1 0 0 0 1 1 0	0107
0 0 0 0 1 1 0 0	0108
	0109

Fig. 9 Aspectul programului în memorie

În continuare vom derula "filmul" operațiilor care au loc pentru execuția programului. Este pus în evidență principiul fundamental al rulării programelor într-un sistem numeric de calcul.

Etapa 0: situația inițială

Ne interesează conținutul numărătorului de program (PC), al registrului de instrucțiuni (RI) şi al registrului acumulator (A).

PC conține adresa primei instrucțiuni (0105), acumulatorul conține data rămasă din operații anterioare, care nu mai interesează pe nimeni iar registrul RI conține codul unei instrucțiuni anterioare, care, de asemenea nu ne interesează.

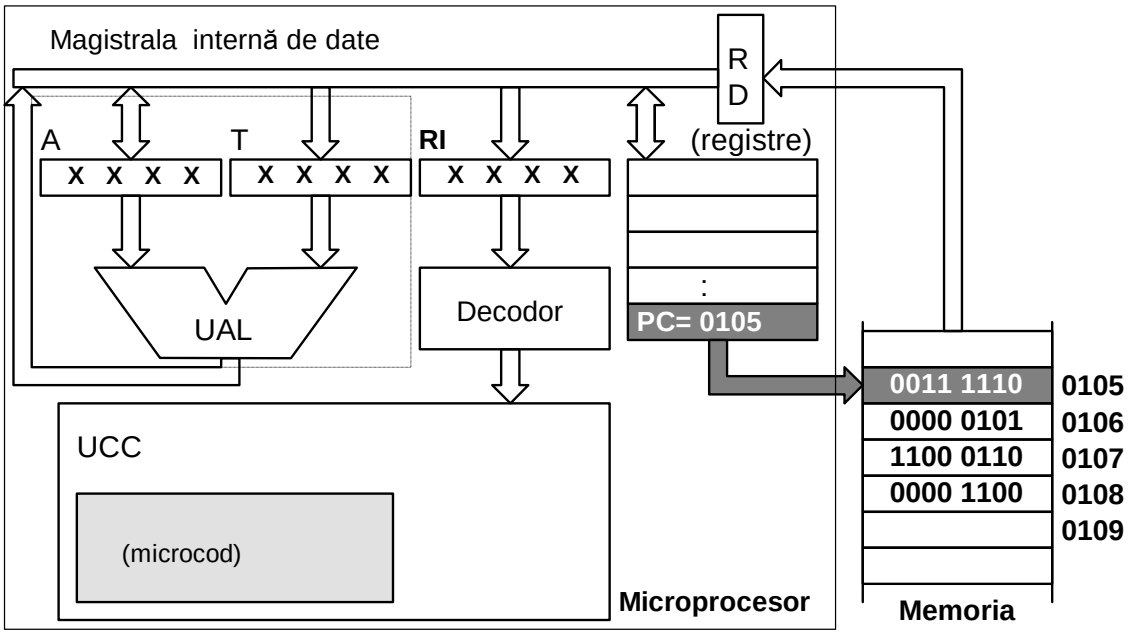

Fig. 10 Adresarea memoriei pentru extragerea primului octet

Etapa 1: Citirea primului octet. Conținutul contorului (PC) se încarcă pe magistrala de adrese, ceea ce are ca efect selectarea locației cu adresa 0105, care conține primul octet al instrucțiunii (de cod).

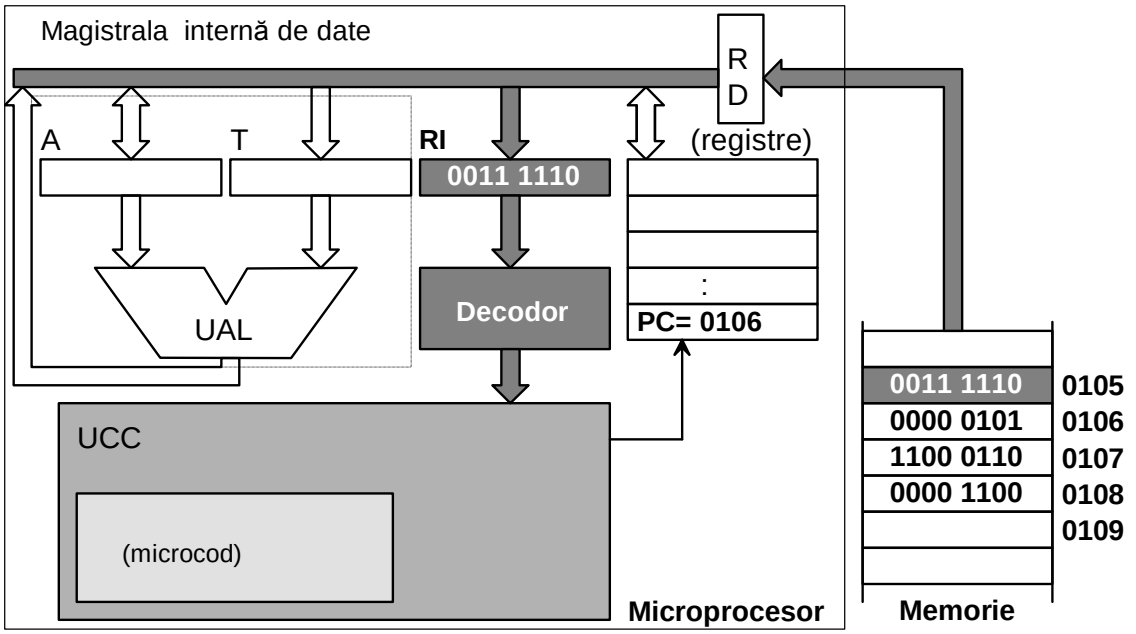

Fig. 11 Încărcarea octetului de cod în RI şi decodarea

După citirea primului octet, contorul de program este incrementat automat, aşadar conținutul lui devine PC = 0106.

După decodare, UCC obține informația privind octetul de date şi comandă citirea lui şi încărcarea în registrul A; UCC "ştie" că al doilea octet este ultimul şi deci va urma eventual altă instrucțiune.

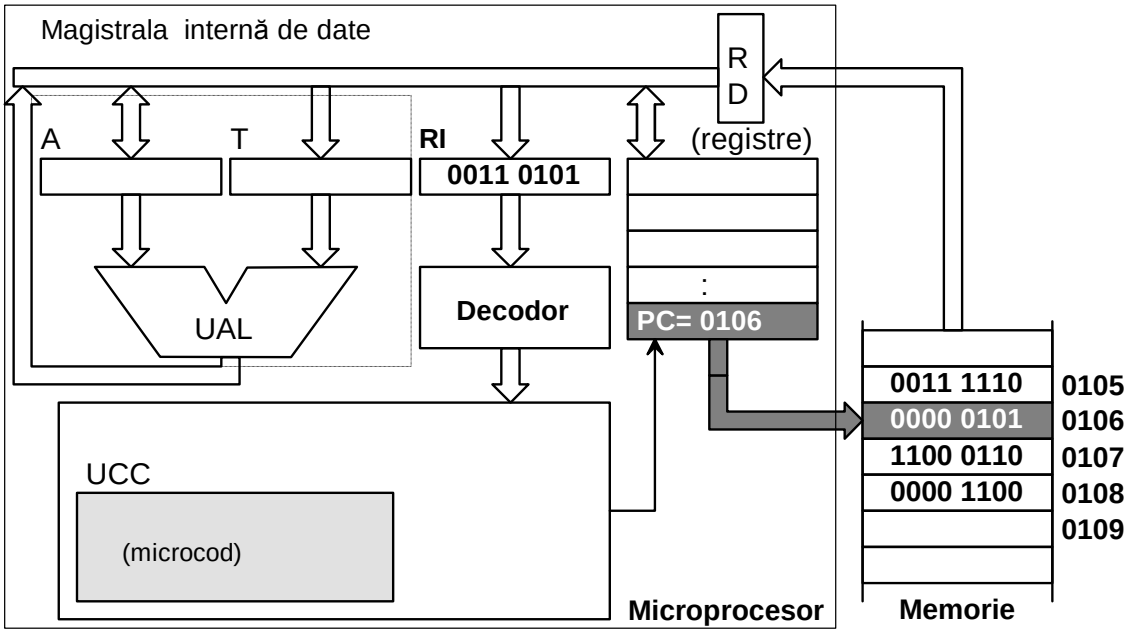

Fig. 12 Citirea celui de-al doilea octet al instrucțiunii (primul operand)

Etapa 2: Citirea octetului al doilea (operandul 05). Conținutul contorului (PC) se încarcă pe magistrala de adrese, ceea ce are ca efect selectarea locației cu adresa 0106, care conține al doilea octet al

instrucțiunii - octetul de date. Acesta va fi încărcat în registrul A, conform comenzii primei instrucțiuni. După citirea octetului de date, în mod automat, conținutul contorului devine PC = 0107.

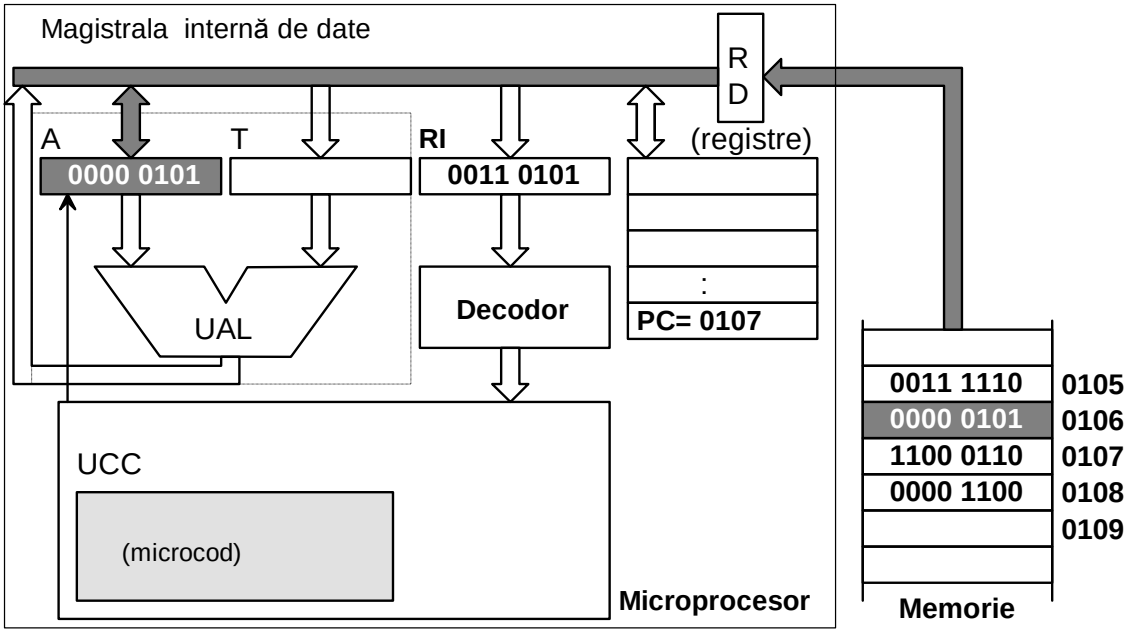

Fig. 13 Încheierea execuției primei instrucțiuni

Etapa 3: Citirea octetului de cod (instrucțiunea a doua).

Se efectuează aceleaşi operații ca în etapa 1, dar locația de memorie selectată este 0107, în care se află codul instrucțiunii a doua.

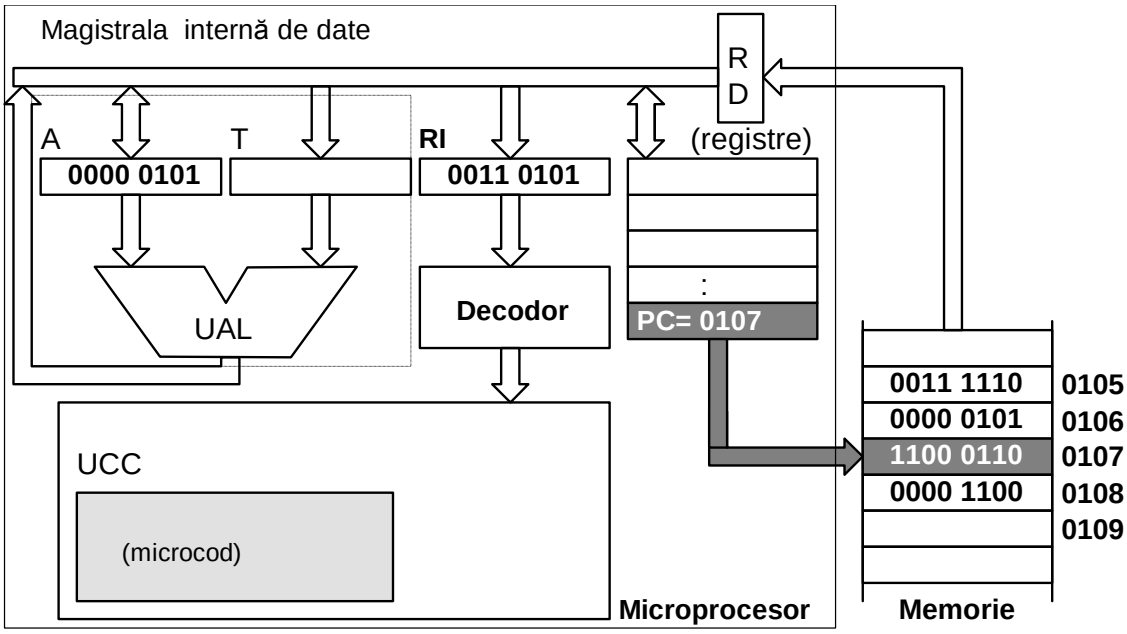

Fig.14 Citirea locației cu adresa 0107, codul instrucțiunii a doua

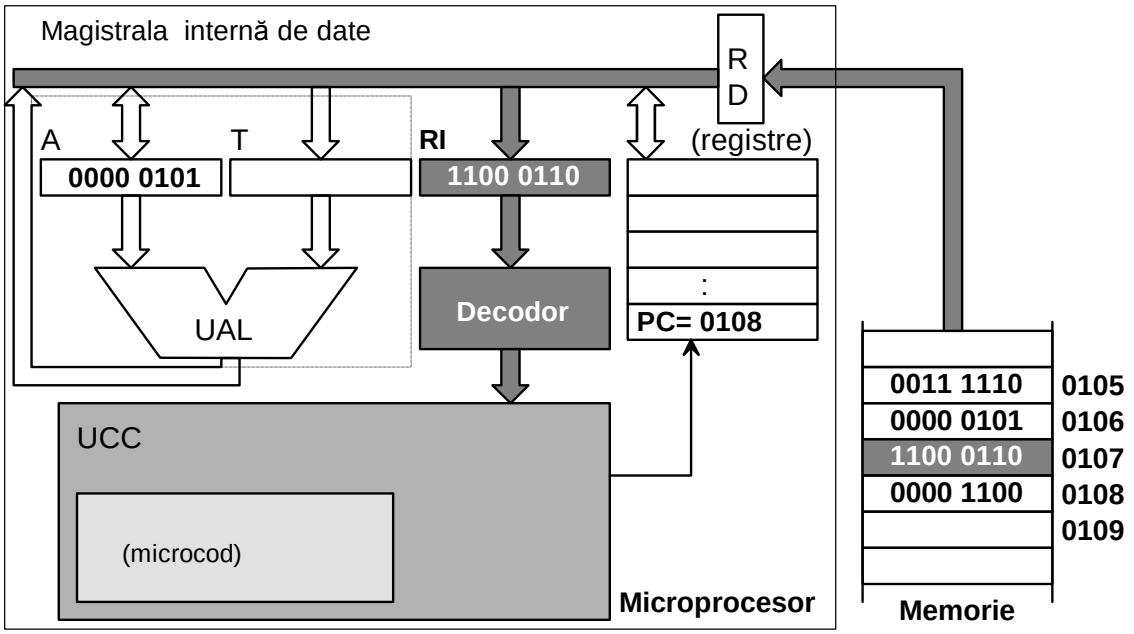

Fig. 15 Decodarea instrucțiunii a doua (de adunare)

Etapa 4: Citirea octetului al doilea.

Acesta reprezintă al doilea operand ce va fi încărcat în registrul temporar T, al unității aritmetice UAL (fig. 16, 17). Transferul în T se execută pe baza comenzii codificate în instrucțiunea ADD A, 0C - de adunare a operandului 0C la acumulator; aceeaşi instrucțiune comandă operația de adunare A+T şi plasarea rezultatului în registrul A (fig.18).

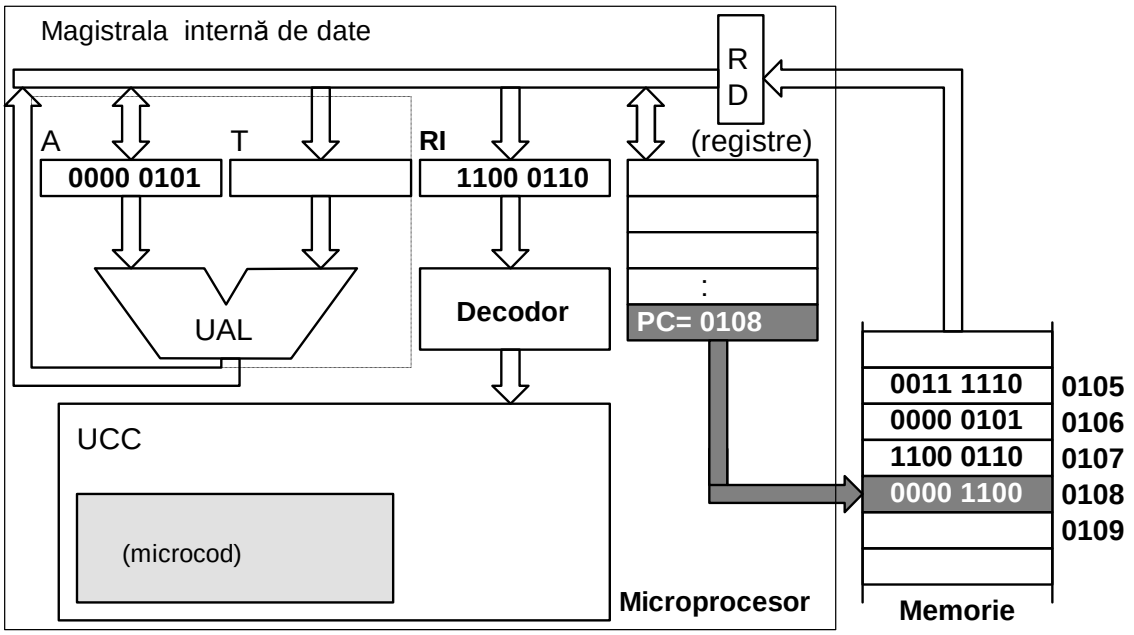

Fig. 16 Citirea octetului al doilea al instrucțiunii de adunare.

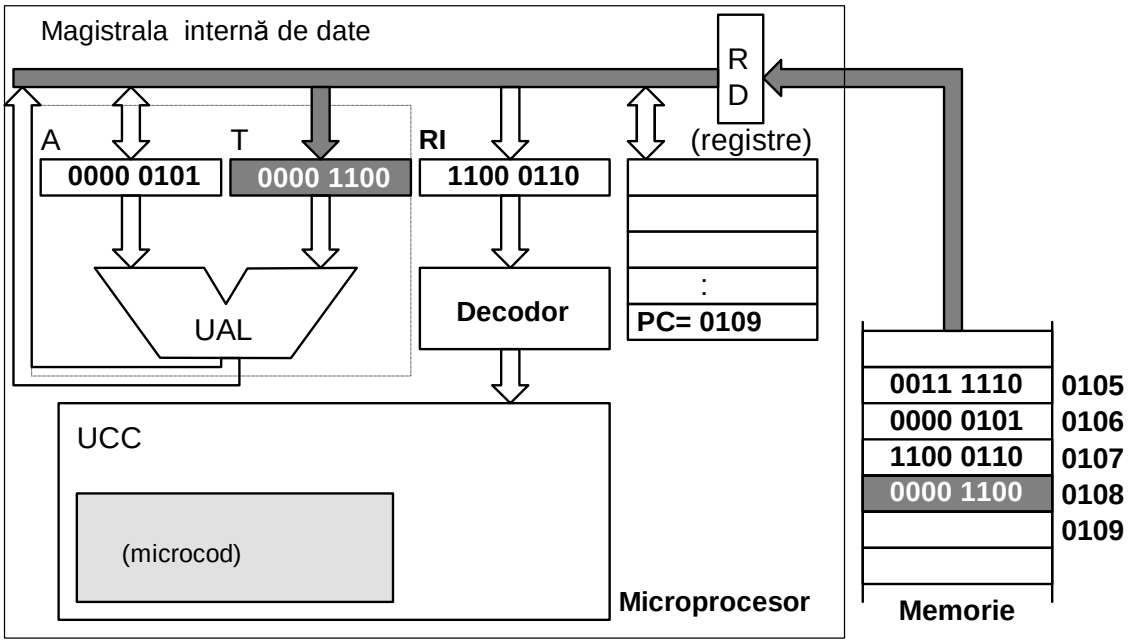

Fig. 17 Încărcarea octetului 0C în registrul temporar T

Operația de adunare A+T se face într-o singură perioadă de tact iar rezultatul este plasat în A prin intermediul magistralei interne de date la care registrul A este conectat.

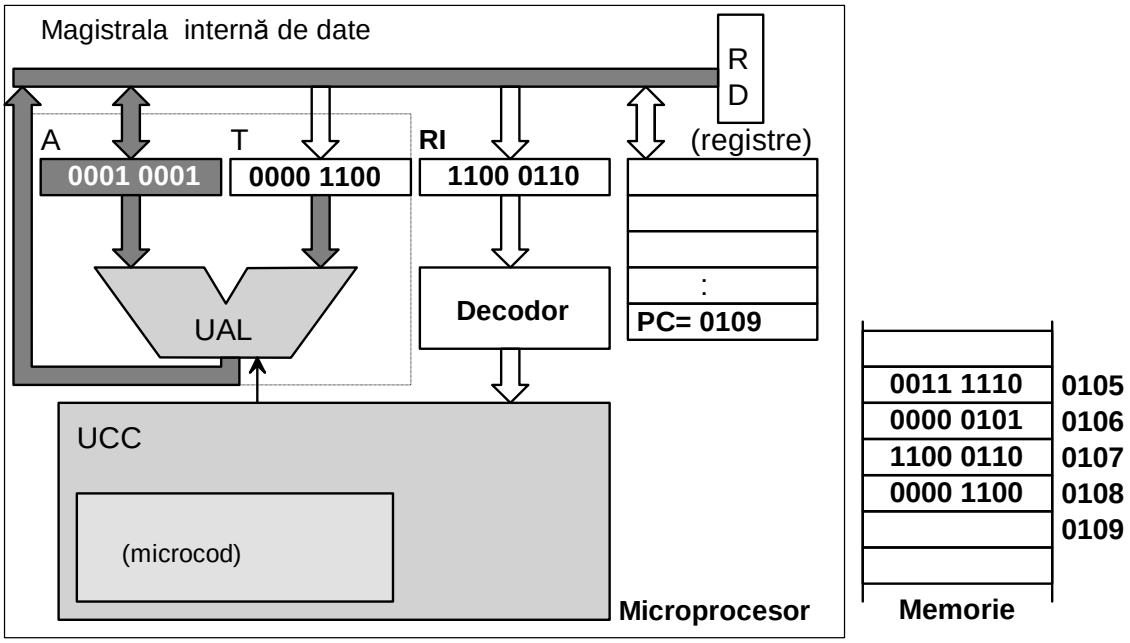

Fig. 18 Efectuarea adunării A+T, rezultatul fiind 0001 0001=17

În fig.19, 20, sunt prezentate cele două instrucțiuni descompuse în operații elementare (conform manualului de utilizare Intel); sunt delimitate perioadele de tact şi ciclurile maşină. Pentru prima instrucțiune, ciclul maşină CM1:

T1: conținutul numărătorului PC este încărcat pe magistrala de adrese iar pe magistrala de date se încarcă temporar cuvântul de stare (*Status*).

T2: se incrementează PC pentru citirea locației următoare.

Tw: dacă citirea memoriei durează mai mult de o stare, se inserează automat o stare de aşteptare (*wait*).

T3: codul instrucțiunii se transferă în registrul de instrucțiuni.

T4: se face decodarea octetului de cod.

Ciclul maşină CM2:

T1: conținutul numărătorului PC este încărcat pe magistrala de adrese iar pe magistrala de date se încarcă temporar cuvântul de stare (*Status*).

T2, (Tw), T3: data din locația adresată se copiază în A (acumulator).

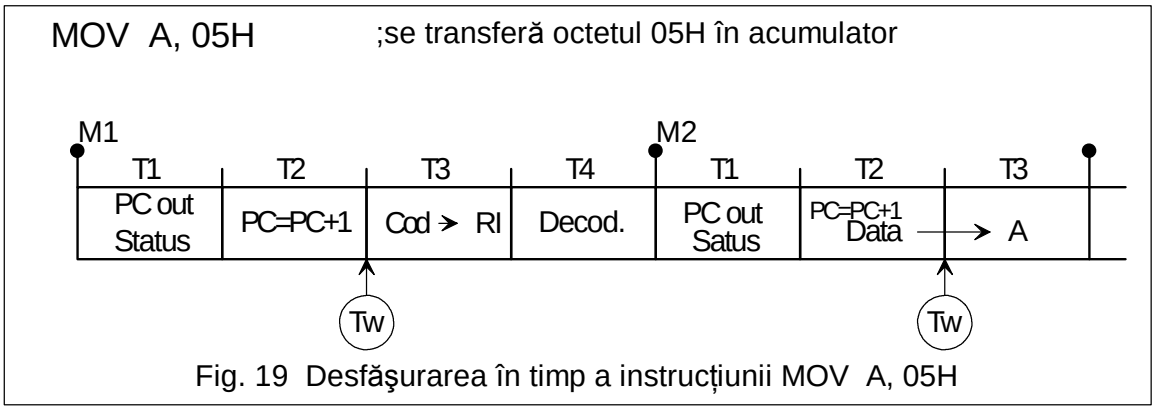

Fig. 19 Desfăşurarea în timp a instrucțiunii MOV A, 05H

În cazul instrucțiunii ADD A, 0CH, desfăşurarea în timp este asemănătoare (fig.20), cu deosebirea că al doilea ciclu maşină are 4 stări, copierea operandului se face în registrul T iar în ultima stare se face adunarea A+T cu depunerea rezultatului în A.

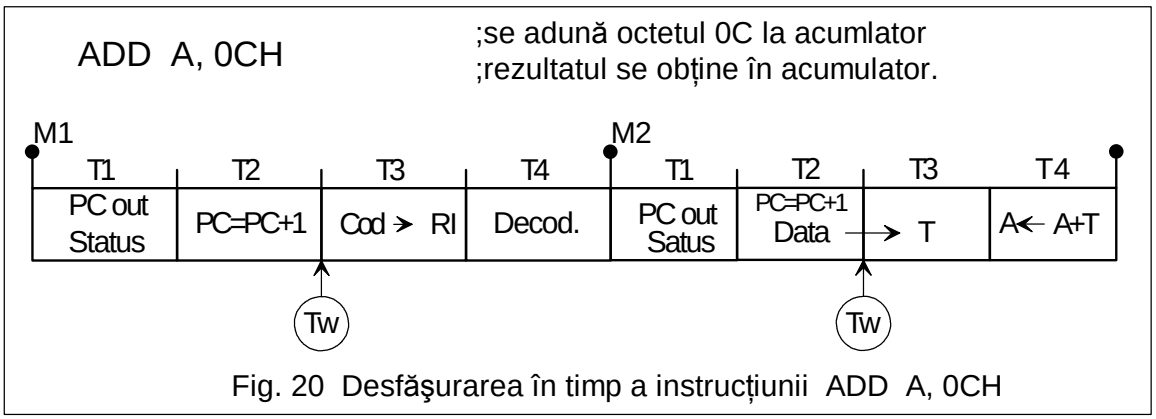

Fig. 20 Desfăşurarea în timp a instrucțiunii ADD A, 0CH

2 Magistrale şi standarde de magistrală

2.1 Conceptul de magistrală

Microprocesorul schimbă informaţii nu numai cu memoria dar şi cu lumea perifericelor (tastatură, mouse, ecran, imprimantă, modem etc.). El dialoghează cu aceste periferice prin intermediul circuitelor de interfaţă, numite încă de "intrare/ieşire" (I/O - input/output). Toate perifericele trebuie să poată fi nominalizate individual prin adrese specifice; aceste adrese nu se confundă cu cele de memorie, deoarece ele sunt active simultan cu comenzi specifice numai circuitelor de interfaţă.

Comunicarea cu memoria şi perifericele se realizează prin linii paralele de transfer de date, magistrale.

Magistrala este un mediu comun de comunicaţie între componentele unui sistem de calcul; fizic, este formată din linii electrice prin care circulă semnale de acelaşi tip şi amplificatoare electronice pentru menţinerea nivelului de tensiune în condiţiile creşterii numărului de componente conectate.

Orice magistrală este generată şi controlată de o unitate specializată, de regulă unitatea centrală a sistemului. Această unitate iniţiază dialogul cu alte unităţi conectate la magistrală (unităţi de intrare/ieşire, memorie). Dialogul este totdeuna de tipul *UC* ↔ *unitate secundară* şi nu între două unităţi secundare.

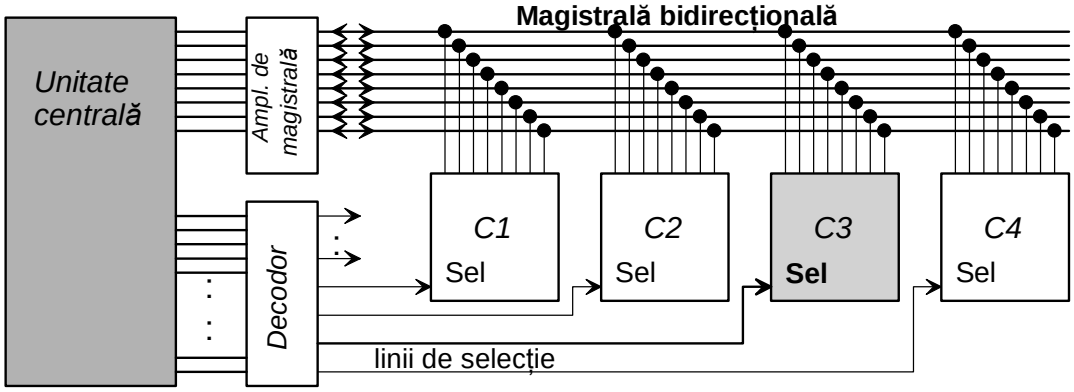

Fig. 1 Comunicare între UC şi componenta C3, prin magistrală.
Celelalte componente conectate sunt pasive

În fig.2, este prezentat amplificatorul de magistrală Intel 8286, bidirecţional, pe 8 biţi, cu ieşiri cu trei stări şi *fan-out* 20 pentru ieşirile B, 6 pentru ieşirile A.

Circuitul realizează funcţiile de amplificare şi transfer atâta timp cât semnalul $\overline{DE} = 0$. Dacă $\overline{DE} = 1$, ieşirile trec în starea de înaltă impedanţă. Sensul transferului este stabilit de intrarea T: dacă *T=1*, transferul se face de la A la B iar dacă *T=0*, de la B la A.

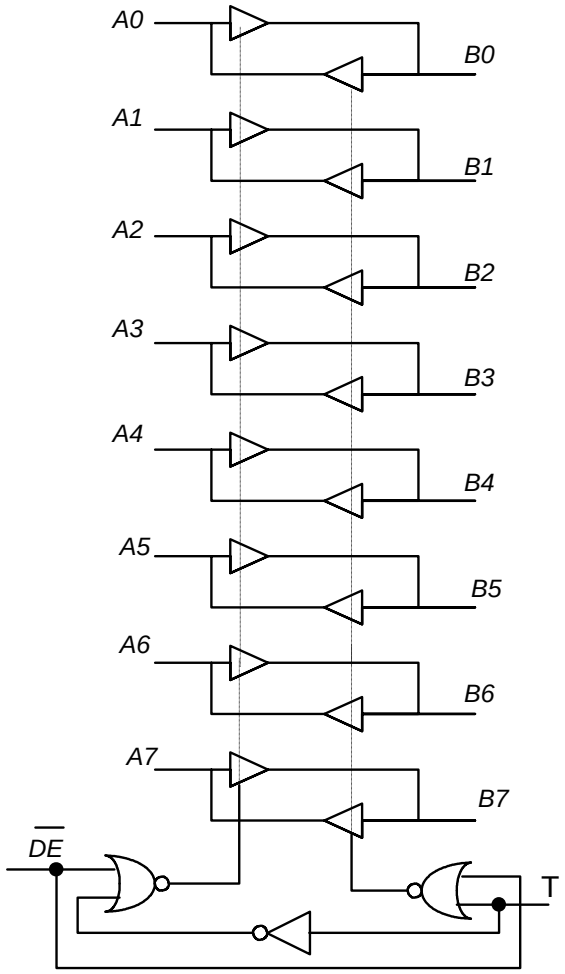

Fig. 2. Schema unui amplificator bidirecțional de magistrală (Intel 8286)

Introducerea conceptului de magistrală a revoluționat modul de concepere şi proiectare a sistemelor de calcul. Modelul de calculator bazat pe magistrală a fost preluat mai ales de familiile de mini şi micro calculatoare. Prin standardizarea magistralelor, sistemele de calcul au devenit deschise, în sensul că producătorii de componente au putut să realizeze module de memorie, interfețe de intrare / ieşire, echipamente periferice, pentru familii mari de calculatoare, bazându-se numai pe specificațiile magistralei.

Au fost dezvoltate diferite standarde de magistrală, urmărind evoluția microprocesoarelor utilizate ca unități centrale şi implicit a necesităților lor de comunicare (viteză, mod de transfer, sincronizare etc.).

S-au dezvoltat magistrale specializate pentru anumite tipuri de echipamente periferice (unități de disc, console grafice etc.).

Din acest punct de vedere se pot distinge două clase de magistrale:

- ***Magistrale de sistem*** - dezvoltate în special pentru conectarea unității centrale cu celelalte componente de bază ale sistemului: ISA, EISA, MULTIBUS, PCI;
- ***Magistrale specializate*** - care sunt destinate optimizării transferului de date între sistem şi anumite tipuri de periferice: VESA, SCSI, GPIB.

Apariția şi evoluția rapidă a microprocesoarelor a consacrat modelul de calculator bazat pe magistrală. Semnalele generate de microprocesor sunt concepute tocmai pentru conectarea la o magistrală de sistem.

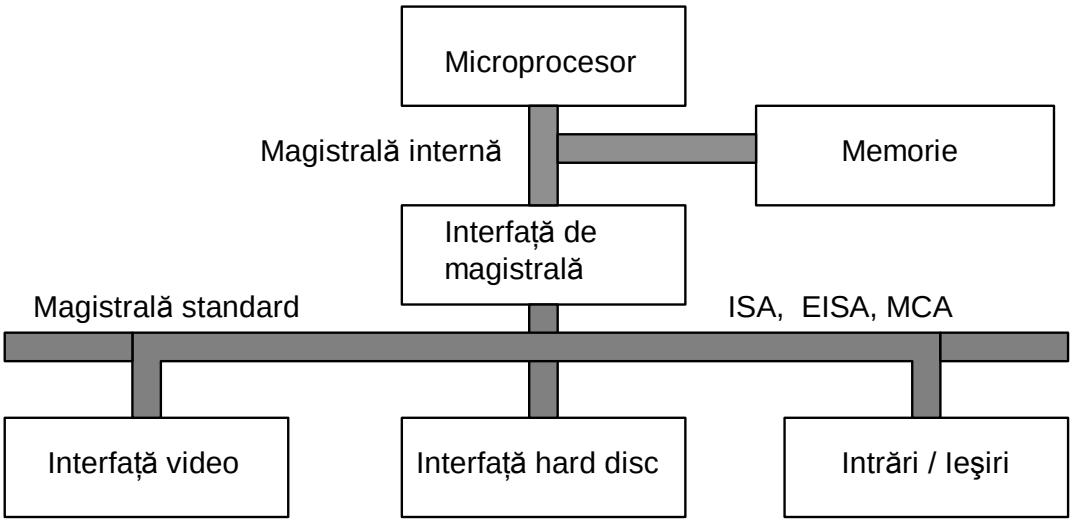

Fig.3 Microprocesorul generează magistrala internă (de date, adrese, şi de control). Interfața de magistrală face corespondendența cu magistrala standard (extinsă).

Deşi familiile de microprocesoare sunt diferite, în funcție de producător, totuşi numeroase caracteristici ale magistralelor lor sunt comune (principii de transfer, tipuri de semnale, dimensiune), astfel că modulele proiectate pentru o anumită magistrală pot fi adaptate uşor pentru alta.

2.2 Elementele de bază ale unei magistrale

O magistrală, care din punct de vedere fizic este constituită din linii conductoare şi amplificatoare de semnal, din punct de vedere funcțional se

compune dintr-un *set de semnale* electrice şi un *set de reguli* care guvernează accesul la comunicaţie şi transferul de informaţii.

Informaţiile transferate pot fi: date, instrucţiuni, informaţii de control şi sincronizare. Regulile de funcţionare se referă la:

- caracteristicile fizice şi electrice ale componentelor conectate la magistrală (niveluri de tensiune, curenţi, încărcare, tip conectori);
- secvenţa de generare a semnalelor necesare pentru transfer;
- timpi limită pentru diferite faze ale unui transfer şi timpi de menţinere a unui anumit semnal;
- dependenţe funcţionale şi temporare între diferite tipuri de semnale.

În funcţie de numărul liniilor utilizate pentru transferul de date, magistralele sunt de două tipuri: *paralele şi seriale*.

Cele seriale se utilizează în principal pentru comunicaţie între sisteme de calcul aflate la distanţă (reţele de calculatoare).

2.3 Magistrale paralele

O magistrală paralelă se compune din următoarele tipuri de semnale:

- *semnale de date* - utilizate pentru transfer de date în ambele sensuri; la un moment dat o singură unitate poate să emită pe liniile de date; numărul liniilor de date (8, 16, 32, 64) determină dimensiunea cuvântului de date ce poate fi transferat şi viteza de transfer;
- *semnale de adresă* - având rolul de adresare a modulului destinaţie sau sursă; numărul liniilor de adresă determină dimensiunea spaţiului de adresare al magistralei;
- *semnale de comandă* - specifică direcţia de transfer (dinspre procesor sau către procesor) şi tipul componentei adresate (modul de memorie, dispozitiv de intrare/ieşire, controler de întreruperi etc.).
- *semnale de control* - stabilesc condiţiile de transfer pe magistrală;
- *semnale de întrerupere* - semnalizează apariţia unor evenimente interne sau externe şi determină întreruperea execuţiei programului curent;
- *semnale de tact* - au rol de sincronizare şi permit generarea unor semnale de frecvenţă programabilă;
- *semnale de alimentare* - utilizate ca tensiuni de alimentare pentru componentele sitemului;
- *semnale de control acces* - utilizate pentru arbitraj şi pentru controlul accesului la magistrală, în cazul magistralelor multimaster.

Numărul şi semnificaţia particulară a semnalelor depinde de tipul şi destinaţia magistralei. Anumite grupe de semnale pot lipsi şi alte grupe pot fi adăugate (semnale de eroare, de control paritate etc.).

Clasificare

1. După modul de lucru, în raport cu semnalul de tact:

- *magistrale sincrone* - ciclurile de transfer sunt corelate cu semnalul de tact. Dimensiunea magistralei este limitată de frecvenţa semnalului de tact.
- *magistrale asincrone* - nu există o legătură directă între ciclul de transfer şi semnalul de tact al sistemului; majoritatea magistralelor actuale lucrează pe acest principiu (ISA, EISA, MULTIBUS).

2. După numărul de unităţi master conectate la magistrală:

- *magistrale unimaster* - există o singură unitate master pe magistrală; nu sunt necesare mecanisme de arbitraj. Unitatea master iniţiază orice transfer de informaţie şi are permanent controlul deplin asupra stării magistralei; unităţile slave conectate la magistrală nu dispun de elementele necesare pentru controlul magistralei.
- *magistrale multimaster* - sunt controlate de mai multe unităţi master dar nu simultan; magistrala trebuie să conţină semnale de arbitraj şi un anumit protocol de transfer al controlului între unităţile master.

3. După modul de realizare a transferului de date:

- *magistrale cu transfer prin cicluri* (secvenţiale); se aplică regula ca la un moment dat cel mult un ciclu de transfer să se afle în desfăşurare; majoritatea magistralelor folosesc acest principiu de transfer.
- *magistrale tranzacţionale* - transferul de date se efectuează prin tranzacţii; o tranzacţie este divizată în mai multe faze care se pot desfăşura simultan dacă ele utilizează grupuri de semnale diferite.

Pentru compatibilizarea modulelor cu sistemele de calcul, au fost standardizate multe tipuri de magistrale.

În cadrul unui sistem de calcul pot să coexiste ma imulte tipuri de standarde, specializate pe transfer de date între diferite tipuri de componente de sistem. Astfel se poate utiliza o magistrală de mare viteză pentru transferul între procesor şi memorie, o magistrală cu acces multiplu pentru conectarea unor periferice de mare viteză (disc, interfaţă video) şi o magistrală de viteză redusă pentru periferice lente.

În domeniul calculatoarelor personale s-au dezvoltat mai multe magistrale care au devenit standarde în fapt, înainte de standardizarea lor formală:

- ISA (*Industrial Standard Architecture*) - magistrala de sistem a primelor calculatoare personale compatibile IBM PC şi care încă se utilizează în majoritatea calculatoarelor personale.
- EISA (*Extended ISA*) - varianta extinsă a magistralei ISA.
- VESA Local Bus (*Video Electronics Standard Association*) - magistrală proiectată inițial pentru pentru accelerarea transferului între procesor şi interfața grafică, s-a dovedit utilă şi pentru alte tipuri de interfețe de mare viteză (*Hard Disc*).
- PCI (*Peripheral Component Interconnect*) - magistrală de mare viteză adaptată cerințelor procesoarelor evoluate din familia Intel. A fost adoptată în arhitectura generală *Power PC* definită de *Apple -IBM - Motorola*.
- PCMCIA (*Personal Computer Memory Card International Association*) în format de carte de credit.

2.4 Caracteristici funcționale ale magistralei ISA

Magistrala ISA (*Industrial Standard Architecture*) a fost definită pentru calculatoarele personale IBM PC. Se utilizează pentru conectarea în sistem a interfețelor de intrare/ieşire specifice unui calculator personal: interfața de disc, interfața video, interfața de sunet şi diferite alte interfețe de utilizator.

Este o magistrală asincronă, care poate transfera date pe 8 şi 16 biți.

Semnalele magistralei se regăsesc pe sloturile de extensie ale calculatorului. Un slot se compune din doi conectori, unul de 64 de pini şi celălalt de 36 de pini.

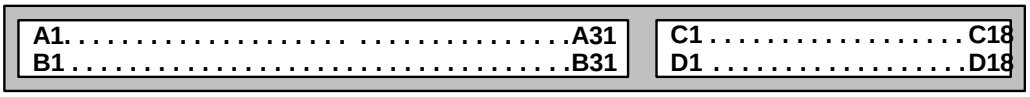

Fig.4 Conectorii magistralei ISA (62 +36 pini)

Pe o magistrală ISA transferul de date se realizează pe bază de cicluri de magistrală. În funcție de tipul transferului, există 4 tipuri de ciclu:

- ciclu de citire memorie ($\overline{MRDC}$ - *Memory Read Command*);
- ciclu de scriere memorie ($\overline{MWTC}$ - *Memory Write Command*);
- ciclu de citire port intrare ($\overline{IORC}$ - *Input Output Read Command*);
- ciclu de scriere port ieşire ($\overline{IOWC}$ - *Input Output Write Command*);

în paranteze sunt menționate denumirile semnalelor care se activează la execuția fiecărui tip de ciclu, toate fiind active în "0".

Conectorii magistralei ISA se compun din două secțiuni:

- Secțiunea de origine, pe 8 biți (PC XT), cu 62 de pini;
- Secțiunea de extensie la 16 biți (PC AT), cu 36 de pini.

Liniile de semnal sunt:

- **SA0 . .SA19** (*System Address*) linii de adresă pentru memorie şi porturi;
- **LA17 . . LA23** (*Lachable Address*) linii de adresă nememorate, valide numai pe durata semnalului **BALE**.

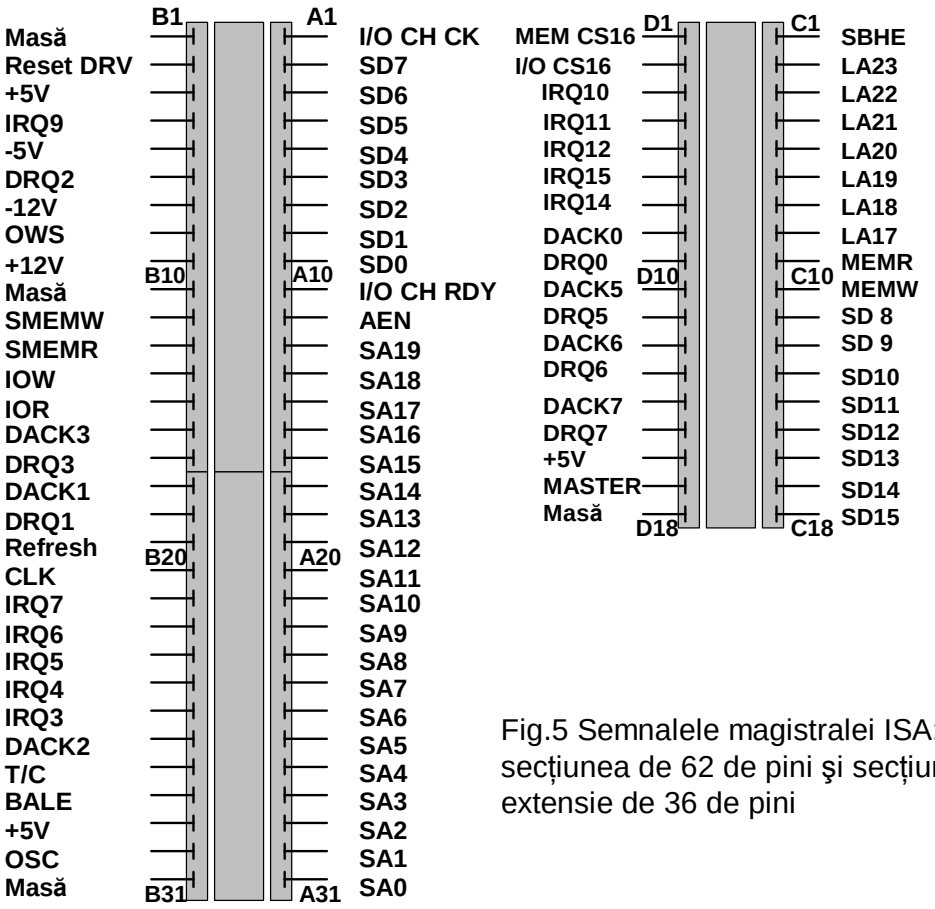

Fig.5 Semnalele magistralei ISA: secțiunea de 62 de pini şi secțiunea de extensie de 36 de pini

- **CLK** linia semnalului de tact al sistemului;
- **Reset DRV** (*Reset Driver*) comanda de inițializare;
- **SD0 . . SD15** (*System Data Lines*) linii bidirecționale de date;
- **I/O CH CK** (*I/O Channel Check*) indică o eroare de paritate;
- **I/O CH RDY** (*I/O Channel Ready*) prin dezactivare, semnalul introduce stări de aşteptare pentru dispozitive lente;
- **IRQ3 . .IRQ7, IRQ9 . .IRQ15** (*Interrupt Request*) cereri de întrerupere;
- **IORC** (*Input/Output Read Command*) citire port de intrare (periferic);
- **IOWC** (*Input/Output Write Command*) scriere port de ieşire (periferic);
- **SMEMR** (*System Memory Read*) citire memorie din spațiul sub 1 MB;

- **SMEMW** (*System Memory Write*) scriere memorie în spațiul de adresare sub 1 MB;
- **MRDC** (*Memory Read Command*) citire memorie în tot spațiul;
- **MWTC** (*Memory Write Command*) scriere memorie în tot spațiul;
- **DRQ0 . . DRQ7** (*DMA Request*) cereri de acces direct la memorie (0..3 pentru transferuri de 8 biți, 4..7 pentru transferuri de 16 biți);
- **DACK0 . . DACK7** (*DMA Acknowledge*) achitare cereri DMA (0..3 pentru transferuri de 8 biți, 4..7 pentru transferuri de 16 biți);
- **AEN** (*Address Enable*) invalidează selecția porturilor I/O pe durata desfăşurării unui ciclu DMA;
- **REFRESH** - indică ciclu maşină de reîmprospătare memorie DRAM;
- **T/C** (*Terminal Count*) indică terminarea transferului DMA;
- **SBHE** (*System Bus Hight Enable*) indică transfer pe octetul superior de date (SD8 . . 15);
- **MASTER** semnal generat de un modul master (procesor de intr./ieşire) când inițiază un ciclu de transfer;
- **MEMCS 16** (*Memory Size 16*) indică selectarea unui modul de memorie pe 16 biți;
- **I/O CS 16** (*Input/Output Size 16*) indică selecția unui port pe 16 biți;
- **OSC** (*Oscilator*) semnal generat de oscilatorul sitemului; acest semnal este divizat cu diferiți factori pentru obținerea frecvenței de transfer, de reîmprospătare memorie etc.

2.5 Arbitrajul magistralelor

Într-un sistem cu mai multe procesoare (sistem multiprocesor), şi o singură magistrală de sistem, un singur procesor trebuie să aibă controlul magistralei, altfel apar conflicte de acces. Este necesară aşadar, prezența unui arbitru, care să atribuie magistrala unui singur procesor la un moment dat.

Gestiunea magistralelor este numită "arbitrajul magistralelor" şi este asigurat de circuite logice speciale; aceste circuite sunt numite generic *ASIC* (*Application Specific Integrated Circuit*) şi au ca sarcină rezolvarea conflictelor de acces la magistrală. Arbitrajul se face în funcție de nivelul de prioritate al fiecărui concurent. În general, dispozitivele de reîmprospătare a memoriei DRAM şi cele de acces direct la memorie, DMA, au nivelul de prioritate maximă.

Problemele de acces la magistrală sunt evitate dacă fiecare procesor are memorie internă "cache"; este unul din motivele pentru care generațiile evoluate de microprocesoare dispun de memorie internă, integrată pe *chip*.