8 Microprocesorul Intel 8080

Apărut în 1971, a fost cel mai răspândit microprocesor de 8 biți.

Realizat în tehnologie NMOS, cu 40 de terminale, utilizează 3 tensiuni de alimentare: +5 Vcc, -5 Vcc, +12 Vcc şi două semnale de tact de aceeaşi frecvență (tipic 2 MHz), dar defazate.

- Toate intrările şi ieşirile sunt compatibile TTL;
- Magistrala de date este de 8 biți (D0 - D7) iar cea de adrese este de 16 biți (A0 - A15), ceea ce permite adresarea unei memorii de maxim 64 kB;
- Poate adresa 512 dispozitive de intrare/ieşire (porturi) distincte;
- Procesează cuvinte de date de 8 biți şi adrese de 16 biți;
- Formează memoria stivă în exterior, în spațiul de memorie adresabil;
- Setul de instrucțiuni conține 72 de tipuri, în total poate executa 244 de instrucțiuni distincte.

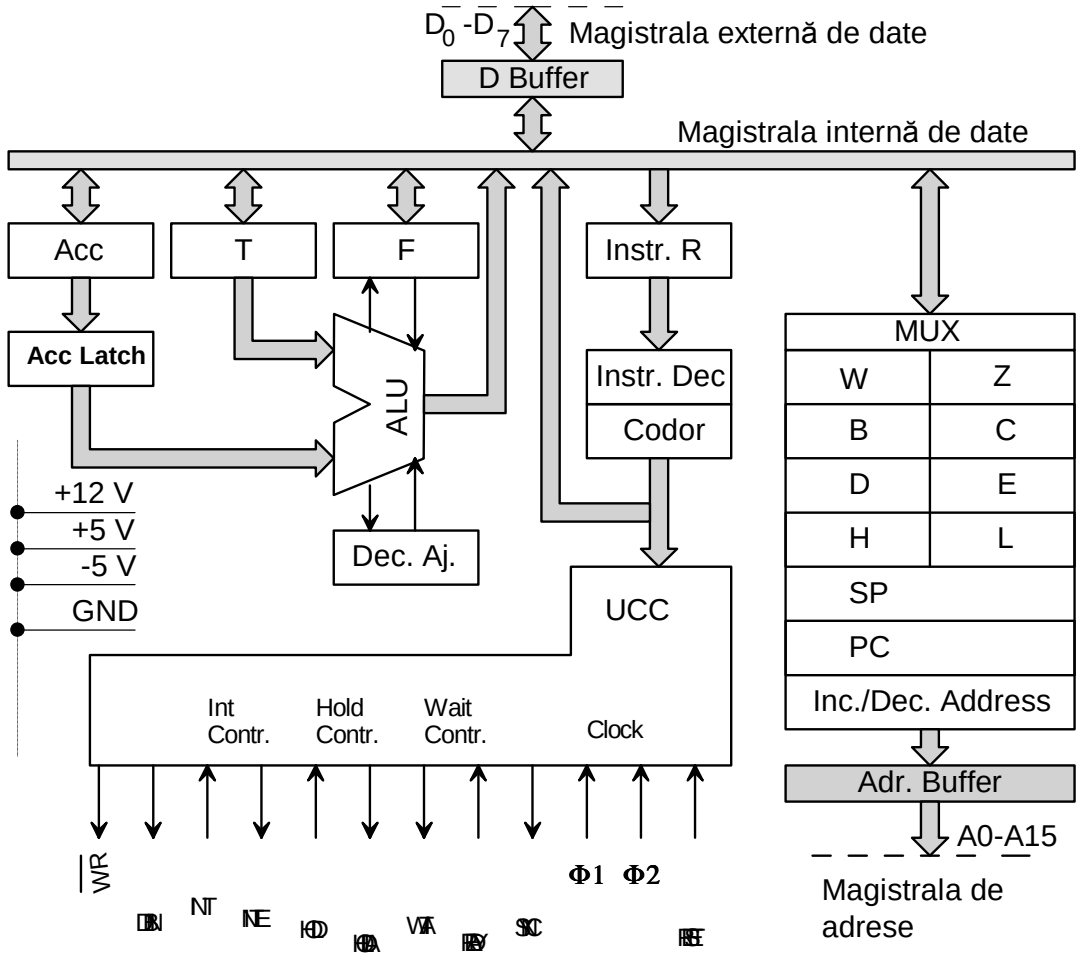

Fig. 1. Arhitectura internă a microprocesorului Intel 8080

8.1 Registrele interne

Constituie memoria internă de tip SRAM - MOS, formată din 6 registre de 16 biți, dintre care 4 pot fi utilizate şi ca 8 registre de 8 biți.

Registrele de uz general sunt: A (Acumulatorul), B, C, D, E, H, L, ca registre de 8 biți sau B, D, H ca registre de 16 biți. Sunt la dispoziția utilizatorului prin intermediul setului de instrucțiuni. Acumulatorul are şi rol de registru de deplasare.

Registrele W, Z nu sunt disponibile utilizatorului, ele fiind folosite de CPU în operații interne.

Registrele de adresă:

PC (*Program Counter*) conține adresa de 16 biți a instrucțiunii următoare din program; pentru extragerea instrucțiunii, conținutul registrului PC se încarcă pe magistrala externă de adrese. După transferul fiecărui octet, conținutul lui PC creşte cu o unitate, pentru adresarea octetului următor.

SP (*Stack Pointer*) - indicator de stivă, conține adresa ultimului operand introdus în memoria stivă, organizată în memoria RAM externă; operațiile de scriere/citire cu stiva se realizează prin intermediul registrului SP.

Un operand de 8 biți poate fi transferat între registrele interne prin intermediul magistralei interne de date de 8 biți, sub controlul unei instrucțiuni de transfer. Registrul sursă (care conține operandul), specificat în instrucțiune, se conectează la magistrala internă prin intermediul unui multiplexor de 8 biți; apoi se selectează registrul destinație în care se transferă operandul de pe magistrală. Tot prin magistrala internă se realizează transferuri între un registru şi o locație de memorie.

Adresele sunt operanzi de 16 biți care se transferă magistralei externe de adrese prin intermediu unui registru special de incrementare / decrementare şi unui registru tampon, de memorare temporară, conectat direct cu magistrala externă. Registrul tampon are rolul de a menține adresa pe magistrală un interval de timp suficient ca aceasta să fie recepționată de memorie sau porturi.

8.2 Unitatea aritmetică şi logică (ALU - Arithmetic and Logic Unit)

Este o structură formată din circuite logice combinaționale, care realizează sub controlul UCC operații aritmetice (adunări, scăderi) şi operații logice (Şi, Sau, Sau Exclusiv) bit cu bit, între doi operanzi de 8

biți. Pentru memorarea celor doi operanzi, sunt necesare două registre de stocare temporară (pe durata operației); acestea sunt T şi Acc. Latch. Registrul Acc. este utilizat inițial pentru unul din operanzi, care trece în Acc. Latch şi în final pentru rezultatul operației.

Pentru operații aritmetice în cod BCD sunt necesare corecții ale rezultatelor, care sunt realizate de un bloc special DAj. (*Decimal Adjust*), iar pentru informații referitoare la rezultatul operației este necesar registrul F (*Flags*) al indicatorilor de condiții sau registrul de stare al UAL (fig.2).

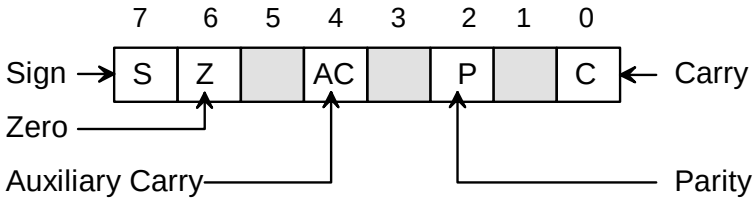

Fig. 2 Registrul indicatorilor de condiții (F) la microprocesorul Intel 8080

8.3 Registrul şi decodorul de instrucțiuni

Instrucțiunile microprocesorului Intel 8080 conțin 1, 2 sau 3 octeți, dintre care primul octet reprezintă codul operației ce se efectuează la execuția instrucțiunii şi este transferat din memorie în registrul IR (*Instruction Register*). Codul operației este interpretat de decodor (*Instruction Decoder*) ale cărui ieşiri, sincronizate cu impulsurile de tact, generează sub controlul UCC toate comenzile interne şi externe necesare efectuării operației.

8.4 Semnale generate şi primite de microprocesor

Semnalele de tact: Φ_1, Φ_2 , sunt generate de un circuit specializat din familia Intel, 8224 - generator de tact. Sunt semnale cu amplitudinea de 12 V şi frecvența de 2MHz, defazate (fig.3). Perioada T, adică intervalul de timp dintre două treceri succesive din "0" în "1" este numită *stare* şi determină durata unei operații elementare ($T = \frac{1}{2 \cdot 10^6} = 0,5 \mu s$).

Semnalul SYNC delimitează durata unui ciclu maşină (o singură referire la memorie sau la dispozitive I/O), fiind generat de fronturile semnalului de tact Φ_2, sub controlul UCC.

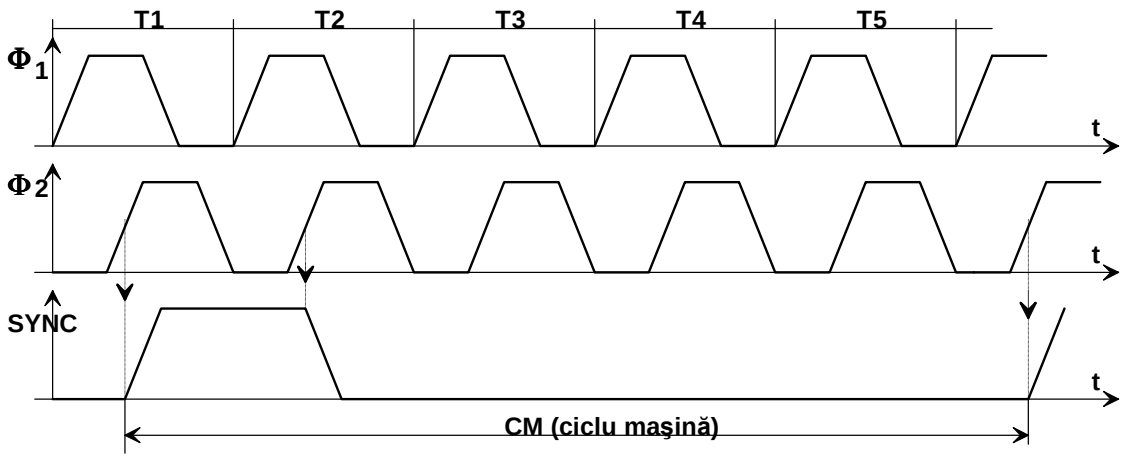

Fig.3 Diagramele semnalelor de tact şi SYNC

Un ciclu maşină are o durată variabilă de 3 - 5 stări în funcție de tipul operației ce se efectuează; o instrucțiune se execută în 1 - 5 cicluri maşină.

Magistrala de date: $\mathbf{D_0}$ **-** $\mathbf{D_7}$ - linii bidirecționale cu trei stări, prin care se realizează schimbul de informație dintre procesor şi memorie sau porturi de intrare / ieşire.

Magistrala de adrese: $\mathbf{A_0}$ **-** $\mathbf{A_{15}}$ - linii unidirecționale cu trei stări (*three state*), prin care se transmite o adresă de 16 biți către memorie sau de 8 biți către porturi. Este orientată de la microprocesor spre exterior.

HOLD : intrare pentru cereri de cedare magistrale. Un dispozitiv inteligent extern solicită controlul total asupra magistralelor în vederea accesului direct la memorie (tehnică *DMA - Direct Access Memory*).

HLDA : ieşire, răspuns la cererea HOLD, confirmând acceptarea acestei cereri, după trecerea magistralelor de adrese, date şi control în starea SIR (starea înaltă impedanță sau starea "a treia")

WR ieşire cu trei stări (*Write*) activă în "0" logic; determină o operație de scriere în memorie sau în porturi.

INT (*Interrupt request*) cerere de întrerupere emisă de un dispozitiv extern pentru tratare prioritară (procesorul trece la execuția unei subrutine de tratare întrerupere, după terminarea instrucțiunii în execuție).

INTE : ieşire cu trei stări (*Interrupt Enable*), activă în "1" logic, când microprocesorul execută un ciclu maşină de acceptare întrerupere.

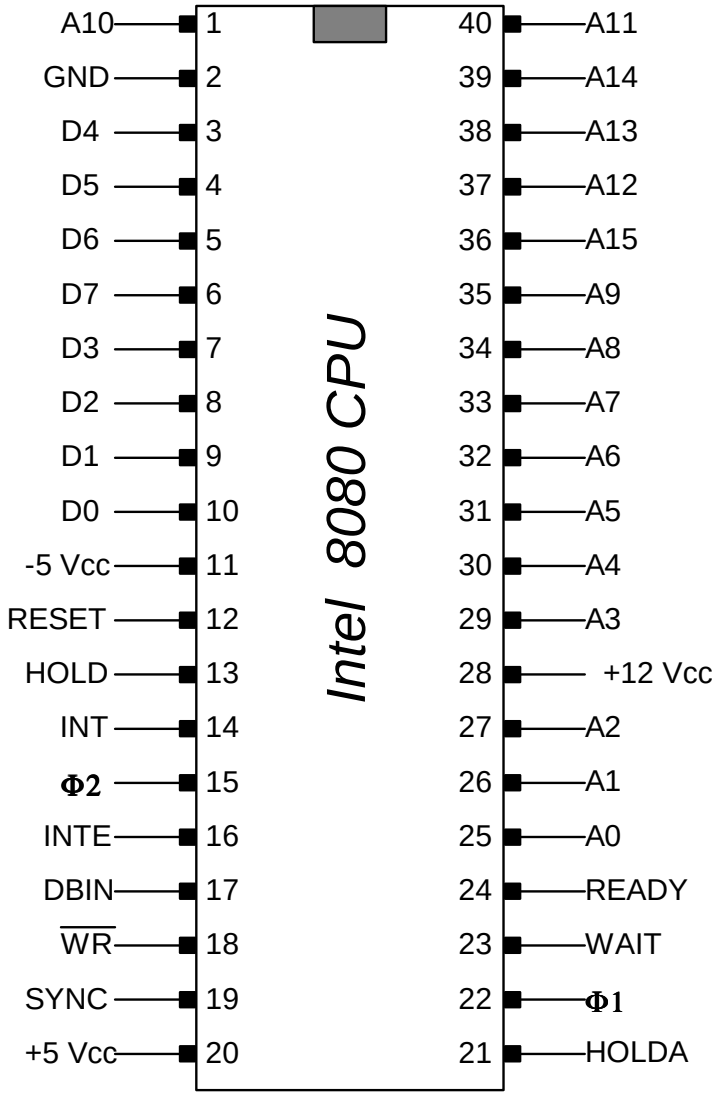

Fig.4 Intel 8080 - Configurația semnalelor la pini

DBIN : ieşire cu trei stări (*Data Bus Input*), care indică sensul transferului pe magistrala de date. Dacă este "1" logic, magistrala de date este orientată către microprocesor (intrare date).

WAIT : semnal generat de microprocesor prin care informează dispozitivele externe că se află în starea de aşteptare date. În această stare, procesorul nu execută operații interne.

READY : semnal generat extern, activ în "1"; informează procesorul că o dată de 8 biți este disponibilă şi stabilă pe magistrala de date. Este confirmarea aşteptată de procesor în starea WAIT pentru a transfera data de pe magistrală în registrele interne. Perechea de semnale WAIT-READY permite schimbul de informație cu memorii sau porturi "lente", care necesită mai mult de o perioadă de tact pentru încărcarea unui octet pe magistrala de date.

RESET : semnal de inițializare, generat din exterior, activ în "1". Are ca efect principal inițializarea registrului PC cu 0000 H, ceea ce duce la reluarea execuției de la adresa 0000 H. De regulă, la această adresă începe un program de inițializare a sistemului de calcul. Semnalul RESET este generat automat la apariția tensiunilor de alimentare (la pornirea sistemului) dar poate fi generat în orice moment prin apăsarea unui buton, deci la intervenția operatorului.

8.5 Organigrama stărilor unui *ciclu maşină* (CM)

Cu excepția unor instrucțiuni speciale (ex. DAD), numărul de cicluri maşină ce corespund unei instrucțiuni este dat de numărul de apeluri la memorie sau porturi, deoarece Intel 8080 transmite o singură adresă / ciclu maşină.

În structura unei instrucțiuni, primul ciclu maşină este totdeauna de citire memorie, în vederea "aducerii" primului octet (de cod) în registrul de instrucțiuni; de aceea se mai numeşte "ciclu maşină de aducere".

Cele 244 de instrucțiuni ale lui Intel 8080 utilizează numai 10 tipuri de ciclu maşină:

- CM de aducere instrucțiune (citire instrucțiuni din memorie);
- CM de citire memorie (MEMR - *Memory Read*);
- CM de scriere în memorie (MEMW - *Memory Write*);
- CM de citire stivă (MEMW);
- CM de scriere în memoria stivă (MEMW);
- CM de citire date dintr-un port de intrare/ieşire (I/OR - *Input/Output Read*);
- CM de scriere date într-un port de intrare/ieşire (I/OW - *Input/Output Write*);
- CM de acceptare întrerupere (INTA - *Interrupt Acknowledge*);
- CM de oprire (HALT);
- CM de întrerupere/oprire.

Toate tipurile de ciclu maşină au aceeaşi structură logică (fig.5) formată din maxim 5 stări (5 perioade ale impulsurilor de tact).

În starea **T1**, o adresă (PC, WZ sau HL) este încărcată pe magistrala de adrese, în vederea citirii unui octet de program sau de date.

Starea **T2** este rezervată pentru acces al memorie, magistrala de adrese conține încă adresa încărcată în starea T1 iar magistrala de date este orientată către procesor în vederea transferului de date. Registrul PC este incrementat (PC+1), fiind astfel pregătit pentru citirea octetului

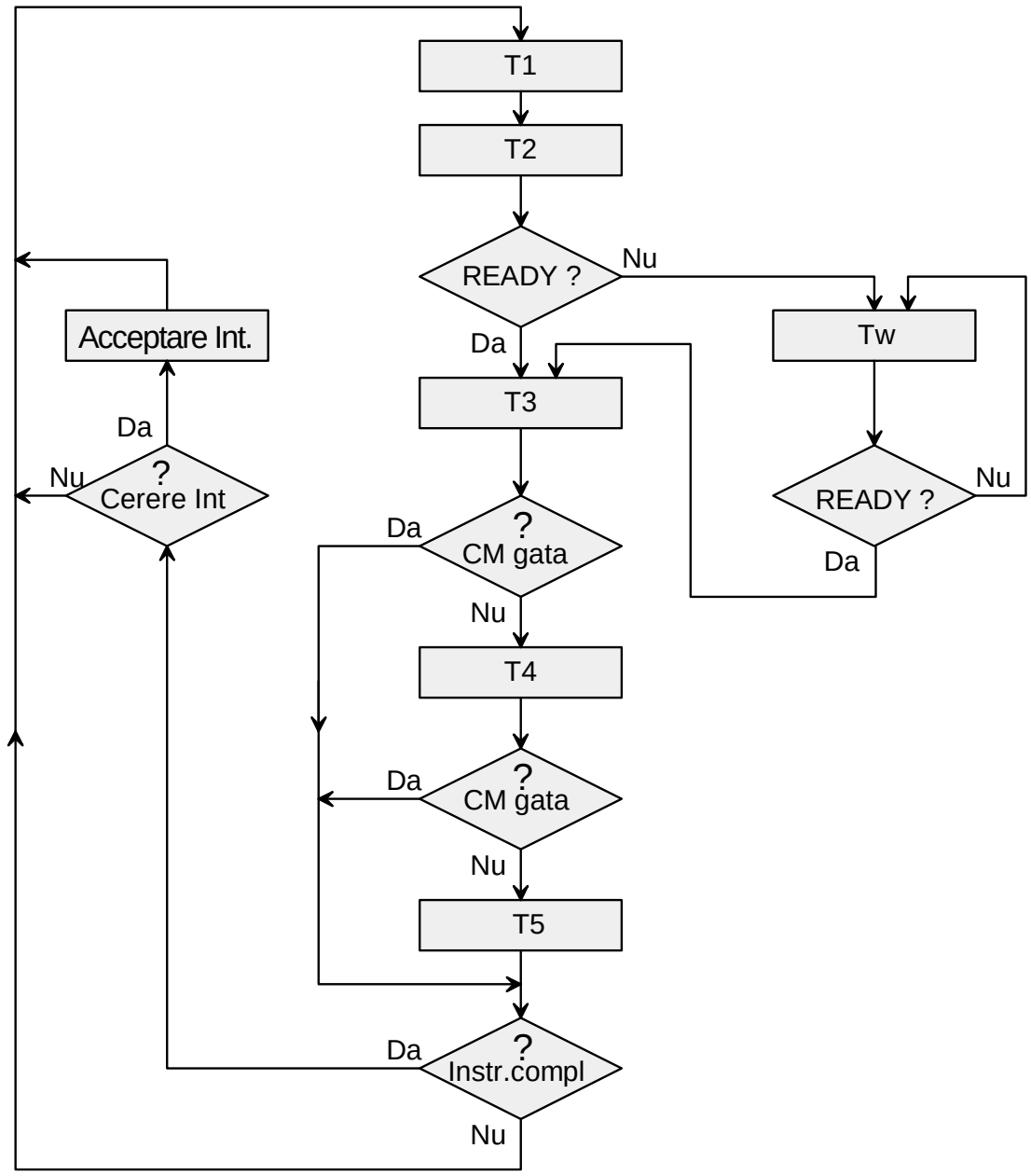

Fig. 5 Organigrama stărilor într-un ciclu maşină

următor din program; dacă se citeşte o dată de 16 biți, se incrementează registrul WZ.

După T2, se testează valoarea logică a semnalului READY, care confirmă sau nu existența unui octet de date pe magistrală. Dacă READY="0", procesorul intră în starea WAIT, de aşteptare, încă o perioadă de tact (Tw), după care se testează din nou intrarea READY. Procesorul va trece la starea T3 numai dacă intrarea READY="1", ceea ce reprezintă confirmarea existenței unei date de 8 biți pe magistrală.

În starea **T3** octetul de date disponibil pe magistrală este transferat într-un registru intern (RI, A, etc.). Se efectuează operații interne.

După T3, se testează tipul de ciclu maşină; dacă acesta are mai mult de 3 stări, se trece la starea T4.

În starea **T4** sunt generate semnale pentru operații interne.

După T4, se testează tipul de ciclu maşină; dacă acesta are mai mult de 4 stări, se trece la starea T5.

În starea **T5**, sunt generate semnale pentru alte operații interne.

Un ciclu maşină are maxim 5 stări; deci după starea T5, ciclul maşină curent este încheiat. Dacă este ultimul ciclu maşină al instrucțiunii, se testează existența unei cereri de întrerupere de tip INT şi în caz afirmativ aceasta este înregistrată şi se inițiază un ciclu maşină special, de tratare întrerupere. Dacă ciclul maşină curent nu încheie istrucțiunea, se continuă cu CM următor, fără a fi testate cererile de întrerupere.

8.6 Unitate centrală cu Intel 8080

Pentru realizarea unității centrale (fig.6) a unui sistem de calcul bazat pe Intel 8080, sunt necesare trei circuite din familia Intel: 8080 CPU, 8224 (generator de tact) şi 8228 (controler de magistrală).

Generatorul de tact 8224 conține un oscilator cu cuarț şi un divizor de frecvență la ieşirea căruia se obțin impulsurile de tact aplicate microprocesorului 8080. Circuitul formează de asemenea semnalele RESET şi READY pe baza semnalelor de intrare de acelaşi tip din sistem (RESIN, RDYIN).

Controlerul de sistem 8228, are două funcții principale:

- Amplificator de magistrală de date;
- Generează magistrala de control pe baza semnalelor WR, DBIN, HOLDA şi a cuvântului de stare transferat de 8080 pe liniile de date în starea T1 a CM.

Execuția unei instrucțiuni constă în efectuarea consecutivă a maxim 5 operații (cicluri maşină). Cuvântul de stare (format din 8 semnale de comandă) ce corespunde ciclului maşină curent este încărcat de microprocesor pe durata semnalului SYNC="1", pe magistrala de date, operația fiind validată de semnalul STSTB (*Status Strob*) - validare stare.

Începând cu starea T3 a ciclului maşină, magistrala de date este utilizată pentru transfer de date între microprocesor şi circuitele externe, iar cuvântul de stare trebuie păstrat pe întreaga durată a ciclului maşină. Este deci necesară păstrarea cuvântului de stare într-un registru latch.

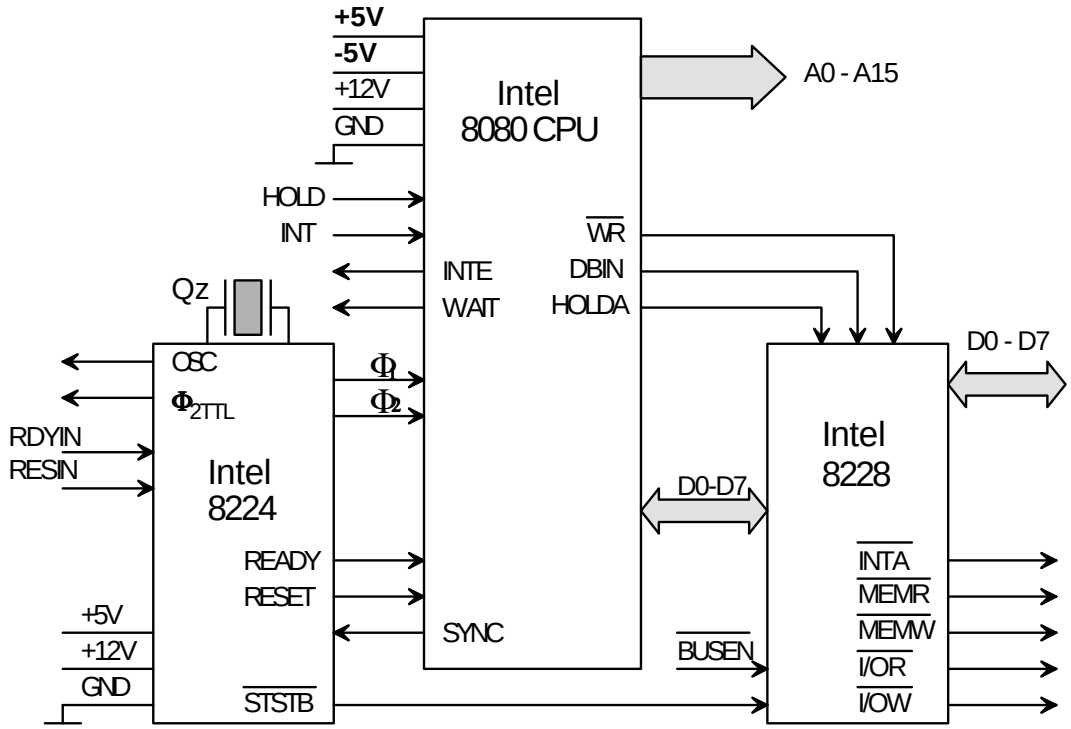

Fig.7 Unitate centrală cu Intel 8080

Acesta se află în structura circuitului integrat 8228 care este şi amplificator de magistrală bidirecțională (buffer). Pe baza cuvântului de stare şi a semnalelor WR, DBIN, HOLDA, 8228 generează cele 5 semnale de comandă active în "0", ale magistralei de control:

MEMR (*Memory Read*) - citire memorie;
MEMW (*Memory Write*) - scriere în memorie;
I/OR (*Input/Output Read*) - citire date din port I/O;
I/OW (*Input/Output Write*) - scriere date în port I/O;
INTA (*Interrupt Acknowledge*) - acceptare întrerupere.

La microprocesoarele 8085, Z80, M6800, controlerul de sistem echivalent cu 8228 este inclus în CPU.

8.7 Formatul instrucțiunilor

Instrucțiunile sunt formate din 1, 2 sau 3 octeți. Primul octet este codul operației caracteristice instrucțiunii. Octeții 2 şi 3 reprezintă date sau adrese. Codul fiind pe 8 biți, rezultă 256 combinații distincte, deci 256 instrucțiuni diferite (8080 execută 244 instrucțiuni diferite care se pot grupa în 72 de tipuri).

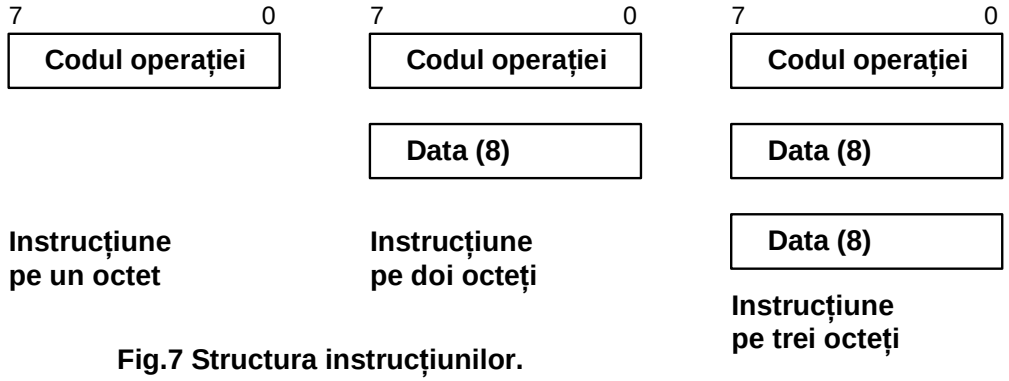

Fig.7 Structura instrucțiunilor.

Octeții unei instrucțiuni sunt memorați în memoria program în locații succesive, începând cu codul instrucțiunii. Instrucțiunile intră în execuție în ordinea în care apar în program. Adresa primului octet este şi adresa instrucțiunii.

În procesul de execuție, prin decodificarea primului octet, unitatea centrală obține informația privind numărul de octeți ai instrucțiunii şi tipul operației.

8.8 Execuția instrucțiunilor

Execuția fiecărei instrucțiuni începe cu un ciclu maşină *de aducere* în care se citeşte octetul de cod şi se transferă în registrul de instrucțiuni. Dacă instrucțiunea conține 2 sau 3 octeți, se repetă ciclul maşină de aducere pentru fiecare octet dar ei se depun în registre de uz general (numai codul se depune în RI).

În figurile 8, 9 şi 10, sunt prezentate stările şi ciclurile maşină pentru 3 instrucțiuni: o instrucțiune simplă (1 CM, 5 stări), o instrucțiune de medie complexitate (2 CM, 7 stări) şi o instrucțiune de maximă complexitate (5 CM, 16 stări).

```
MOV  A, B      ;se transferă conținutul registrului B în A, registrul B
                nu se modifică
```

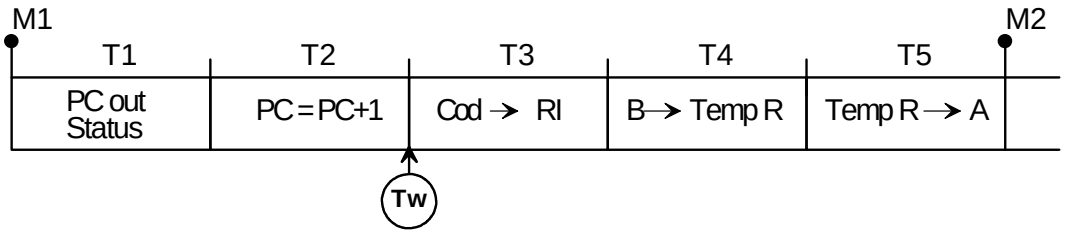

Fig. 8 Structura CM la instrucțiunea MOV A,B

În starea T1, conținutul registrului PC se transferă pe magistrala de adrese în vederea extragerii codului instrucțiunii (*PC out*) şi cuvântul de stare corespunzător ciclului maşină se transferă pe magistrala de date în 8228 pentru generarea semnalelor de comandă de citire memorie (*Status*).

În starea T2, se incrementează PC în vederea extragerii octetului următor din memorie. Dacă memoria nu răspunde pe durata stării T2, microprocesorul intră într-o stare suplimentară, de aşteptare, Tw.

În starea T3, codul instrucțiunii, extras din memorie, se transferă în registrul de instrucțiuni, este decodificat şi în T4, T5 se efectuează operația de transfer a conținutului lui B în A, prin intermediul unui registru temporar.

Instrucțiunea conține un singur ciclu maşină şi se încheie după 5 stări (fig.8).

```
MOV  A, (HL)        ;se transferă conținutul locației de memorie adresate
                    ;cu HL în acumulator; conținutul locației nu se schimbă.
```

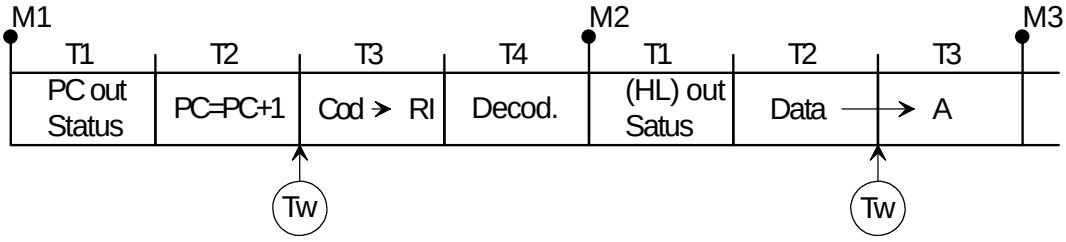

Fig. 9 Structura CM la instrucțiunea MOV A, (HL)

În fig.9, primul ciclu maşină (de *aducere*) are 4 stări, deoarece instrucțiunea fiind mai complexă, necesită o stare (T4) pentru decodificare, iar operația codificată în instrucțiune se execută într-un ciclu maşină separat, M2, de citire date din memorie. În starea T1 din M2, conținutul perechii de registre HL se transferă pe magistrala de adrese în vederea citirii locației de memorie cu adresa egală cu HL. După starea T2, de acces la memorie şi eventual după una sau mai multe stări Tw, octetul de date (Data) din memorie se transferă prin magistrala de date direct în registrul A.

În fig. 10, este prezentată succesiunea ciclurilor maşină pentru o instrucțiune de complexitate maximă, care necesită 5 operații consecutive de acces la memorie şi are trei octeți (un octet de cod şi doi octeți de date care reprezintă o adresă).

Primul ciclu maşină este identic cu cel din instrucțiunea precedentă.

În M2, M3 - cicluri maşină de *aducere*, se extrag din memorie al doilea octet al instrucțiunii (Byte 2) şi al treilea (Byte 3), care sunt depozitate în registrele speciale Z şi respectiv W (Z=25H, W=9AH).

În M4 - citire memorie, octetul de la adresa WZ=9A25H se transferă în L şi în M5 - citire memorie, octetul de la adresa WZ+1=9A26H se transferă în H.

Se observă că în M4 şi M5, adresarea memoriei se face cu perechea de registre WZ, registrul de adresare PC fiind utilizat numai pentru transferul octeților din program (instrucțiuni).

LHLD 9A25 ;se încarcă perechea de registre HL din memorie
;de la adresa 9A25: (9A25)→L, (9A26)→H

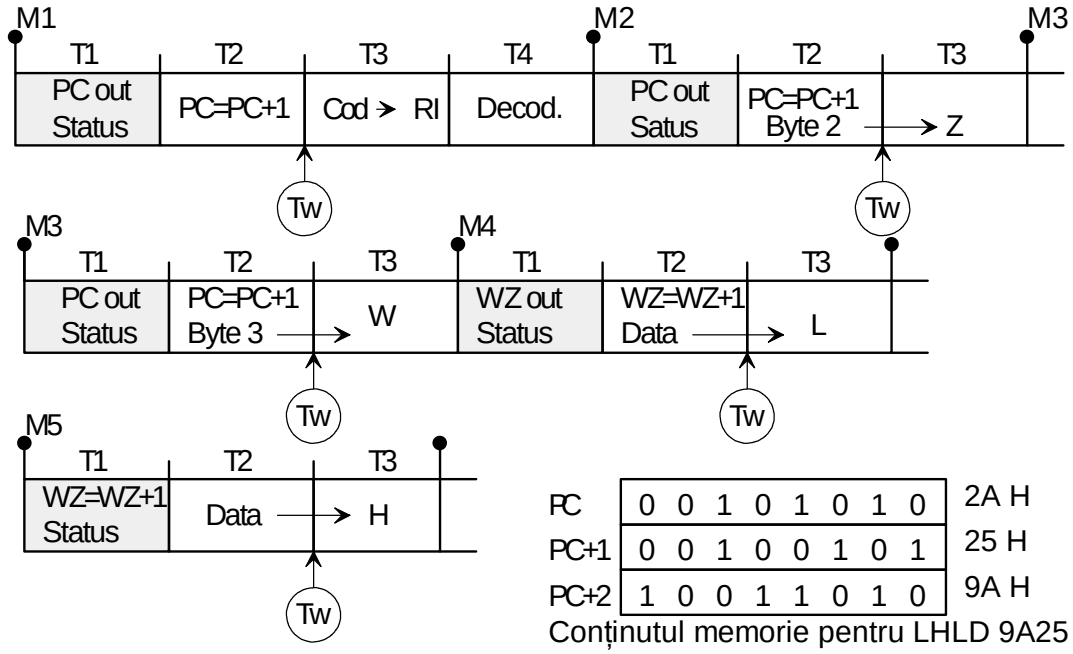

Fig. 10 Structura CM la instrucțiunea LHLD 9A25

9 Microprocesorul Zilog 80 (Z 80)

Apărut în 1972, este un microprocesor clasic de 8 biți, celebru în perioada sa de glorie pentru calitățile sale. A fost proiectat şi realizat de colectivul de ingineri care a realizat 8080 la Intel şi care a migrat la Zilog pentru importante avantaje financiare. Concurenții importanți ai lui Z80 au fost: Intel 8080, Intel 8085, Motorola 6800 şi MOSTech 6502 utilizat în primele calculatoare *Commodore*.

Realizat în tehologie NMOS, cu 40 de pini, necesită o singură tensiune de alimentare (+5 Vcc). Are magistrala de date de 8 biți şi cea de adrese de 16 biți.

Față de Intel 8080, prezintă importante perfecționări din punct de vedere hardware şi software:

- Includerea logicii de tact în structura internă simplifică generatorul de tact extern, care trebuie să fie doar monofazic şi nu necesită un circuit specializat;
- Include logica de generare a semnalului *Refresh* şi un registru intern, R, necesare pentru reîmprospătarea memoriilor DRAM;
- Dispune de linie de întreruperi nemascabile - NMI;
- Dublarea setului de registre de uz general şi indicatorilor de condiții, ceea ce permite tratarea cererilor de întrerupere prin schimbarea setului de bază cu cel suplimentar, fără a utiliza memoria stivă.
- Un registru special - I pentru stocarea octetului superior al vectorului de întrerupere, ceea ce permite plasarea subrutinelor de tratare la orice adresă;
- Adresarea indexată a memoriei, preluată de toate microprocesoarele ulterioare, realizată cu două registre index, IX şi IY, de 16 biți;
- Extinderea setului de instrucțiuni de la 72 de tipuri la 158 tipuri, cu menținerea compatibilității codului de instrucțiuni cu cel de la 8080;
- Apar tipuri noi de instrucțiuni: instrucțiuni pentru transferul unor blocuri de date organizate în zone de memorie şi instrucțiuni care testează sau modifică biți individuali în registre sau în locații de memorie.
- Z80 este compatibil în funcționare cu circuitele familiei Intel: 8255, 8251, 8253 dar are propria familie Zilog: PIO, SIO, CTC, DMA.

Z80 şi familia Zilog au fost fabricate peste 10 ani la Microelectronica din Bucureşti sub denumirea MMN 80, MMN 8x (sub licență Zilog).

9.1 Structura internă

Este în principiu aceeaşi cu cea de la Intel 8080, cu dezvoltările menționate mai sus privind setul de registre şi noile registre speciale introduse.

Magistrala de control este generată intern şi ca urmare nu este necesar un circuit specializat (tip 8228), cu funcția de controler de magistrală.

La indicatorii de condiții 8080 se adaugă unul singur, N (Negative).

Magistrala de adrese, A_0-A_{15} de 16 biți permite adresarea unei memorii externe de 64 kB şi 512 porturi de intrare/ieşire, adresate cu octetul inferior de adresă.

Magistrala de date D_0-D_7 de 8 biți este bidirecțională şi permite organizarea memoriei şi porturilor pe octeți.

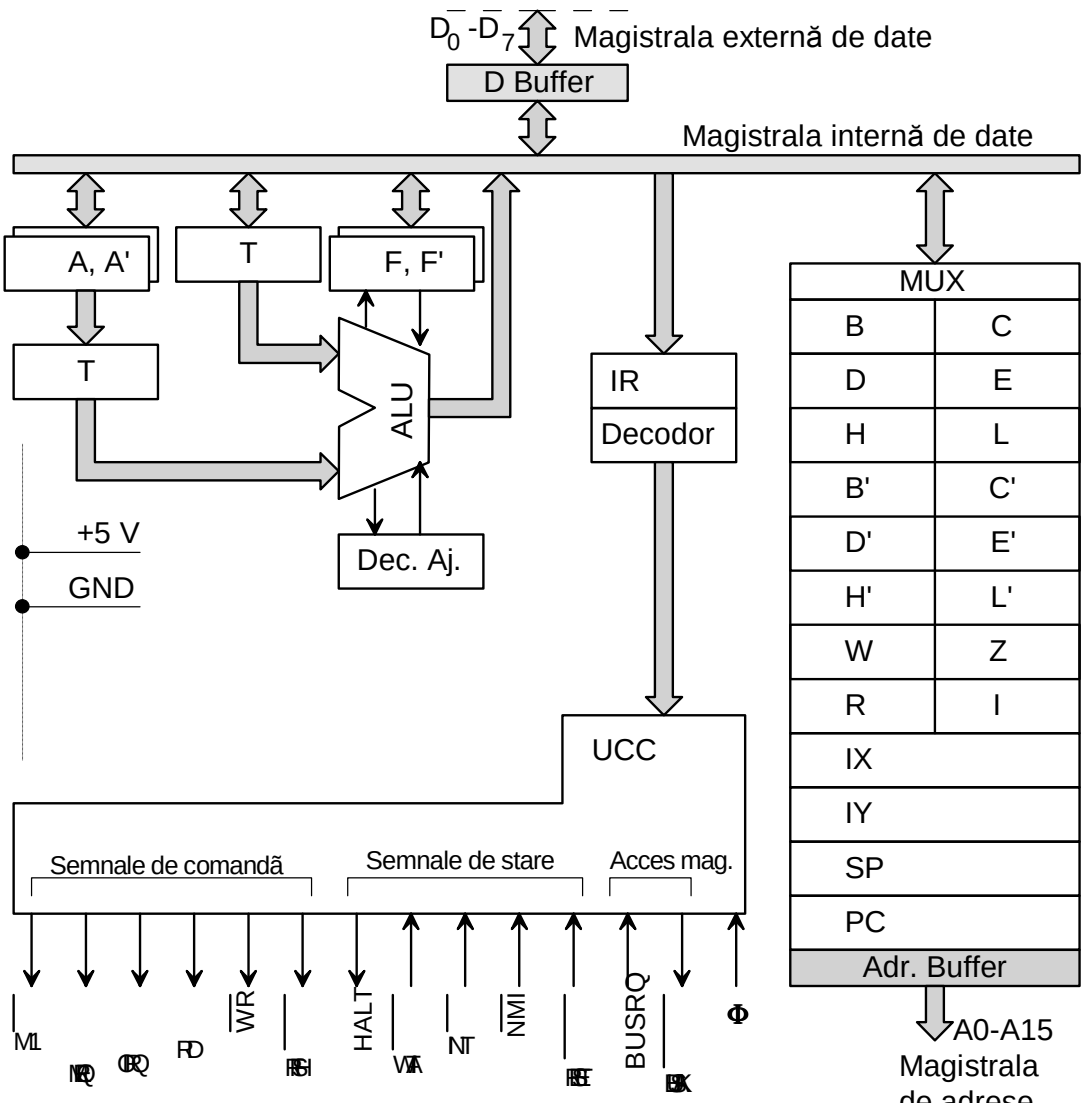

Fig. 11. Arhitectura intenă a microprocesorului Z80

1. Semnificaţia semnalelor de comandă

M1 (*Machine Cycle One)* - semnal de ieşire, activ în "0" indică (atunci când apare simultan cu **MREQ**) primul ciclu maşină al unei instrucţiuni – ciclul de extragere a codului operaţiei din memorie sau (când apare simultan cu **IORQ**) începutul unui ciclu maşină de tratare a unei întreruperi.

MREQ (*Memory Request*) - semnal de ieşire, activ în "0", care arată că pe magistrala de adrese este încărcată şi stabilă o adresă de memorie ce va fi utilizată
într-o operaţie de citire sau scriere.

IORQ *(Input/Output Request)* semnal de ieşire, activ în "0" care indică încărcarea pe octetul inferior al magistralei de adrese (A_0-A_7) a unei adrese ce va fi utilizată într-o operaţie de citire sau scriere cu un port de intrare/ieşire. Dacă apare simultan cu M1, marchează, pe durata unui

ciclu de tratare a unei întreruperi, momemtul în care adresa de răspuns poate fi plasată pe magistrala de date în vederea accesului la subrutina de tratare.

RD *(Read)* semnal de ieşire activ în "0", ce reprezintă comanda de citire date din memorie sau port I/O. Semnalul valideaza transferul datelor pe magistrală.

WR *(Write)* semnal de ieşire activ în "0" ce reprezintă comanda de scriere date în memorie sau port I/O. Semnalul valideaza transferul datelor pe magistrală.

RFSH *(Refresh)* – reîmprospătare. Ieşire activă în "0" care apare simultan cu MREQ indicând că pe cei mai puțin semnificativi 7 biți ai magistralei de adrese (A_0-A_6) se găseşte o adresă de memorie DRAM; bitul A_7=0 iar pe A_8-A_{15} se plasează conținutul registrului R.

2. Semnalele de stare

HALT *(Halt State)* starea de oprire; ieşire activă în "0", generată ca urmare a execuției instrucțiunii HALT. Microprocesorul execută în mod repetat instructiuni NOP (nici o operație) pentru a asigura funcția de reîmprospătare a memoriei DRAM. Din starea de oprire poate ieşi la apariția unei cereri de întrerupere (dacă au fost validate în prealabil) sau la comanda RESET.

WAIT – aşteaptă !... Intrare activă în "0" care indică microprocesorului că memoria sau portul adresate în prealabil nu sunt gata pentru a efectua transfer de date; procesorul rămâne în stare de aşteptare până când semnalul devine inactiv. Pe durata în care semnalul este activ nu se generează RFSH.

INT *(Interrupt Request)* – cerere de întrerupere externă. Semnal de intrare activ în "0" generat de un dispozitiv extern prin care acesta solicită execuția unei subrutine specifice din memorie. Cererea este luată în considerație la sfârşitul instrucțiunii în curs de execuție cu condiția ca întreruperile de tip INT (mascabile) să fi fost validate în prealabil de către program (cu instrucțiunea EI - *Enable Interrupt*) şi semnalul BUSRQ sa nu fie activ. Achitarea unei cereri INT se poate face în 3 moduri printr-un ciclu maşină de achitare, recunoscut prin generarea lui IORQ simultan cu M1. De obicei la intrarea INT se conectează prin "sau cablat" mai multe intrări.

NMI *(Non Maskable Interrupt)* cerere de întrerupere nemascabilă. Semnal de intrare activ pe frontul coborâtor generat la apariția unui eveniment care trebuie tratat imediat indiferent de importanța programului în execuție. Cererile NMI nu pot fi desconsiderate prin software; acestea vor fi obligatoriu deservite în cel mai scurt timp posibil (la terminarea

instrucțiunii curente) şi vor întrerupe chiar şi tratarea unei cereri de tip INT.

Evenimente care pot genera NMI: avertizare asupra căderii tensiunii de alimentare, intervenția operatorului.

Datorită importanței sale, semnalul NMI nu se consideră activ pe nivelul 0 sau 1, ci pe frontul $1 \rightarrow 0$.

RESET - inițializare microprocesor. Intrare activă în "0", cu durata de minim 3 perioade de tact care are următoarele efecte:

- Numărătorul de program este anulat: PC = 0000 H;
- Se invalidează întreruperile de tip INT;
- Se anulează conținutul registrelor speciale: R = 00 H, I = 00 H;
- Se instalează *modul zero* de tratare a întreruperilor INT;
- Pe durata în care RESET este activ, magistralele de date şi adrese trec în starea de înaltă impedanță (SIR), semnalele de comandă devin inactive, nu se generează *Refresh*.

3. Semnale de acces la magistrale

BUSRQ (*Bus Request)* – semnal de intrare, activ în "0", cerere de acces la magistralele sistemului. Z80 termină ciclul maşină în execuție, trece magistralele în stare SIR şi generează semnalul de acceptare a cererii, BUSAK. Dispozitivul care a adresat cererea, prin comanda magistralelor, având astfel acces la toate resursele sistemului. De regulă, în acest mod se realizaează transfer rapid de date cu memoria.

BUSAK *(Bus Acknowledge)* - acceptarea cererii BUSRQ. Semnal activ în "0" pe toată durata cedării magistralelor.

Φ - semnal de tact, de nivel TTL, generat extern (2,5 MHz – 6 MHz).

Întreruperile externe

O întrerupere opreşte temporar execuția unui program şi transferă controlul unei subrutine specifice de tratare, care corespunde cauzei ce a generat întreruperea. Mecanismul prin care se face acest transfer este în esență de tip apel de procedură, ceea ce implică revenirea în programul întrerupt după execuția subrutinei de tratare.

Microprocesorul are două intrări pentru întreruperi, NMI şi INT, dintre care NMI are prioritate absolută.

Cererea INT este mascabilă prin introducerea în program a instrucțiunii DI - de invalidare întreruperi (*Disable Interrupt*). Pentru validare întreruperi se utilizează instrucțiunea EI (*Enable Interrupt*).

Cererea INT nu va fi luată în considerație dacă semnalul BUSRQ este activ, deci cererile de acces la magistrale au prioritate față de cererile de întrerupere mascabile.

Recunoaşterea unei cereri INT este realizată prin generarea semnalelor active M1 şi IORQ care apar în cadrul unui ciclu maşină special, de întrerupere, care conține două stări T_w între stările T_2 şi T_3, ceea ce va permite logicii externe să plaseze un vector (adresă) de întrerupere pe magistrala de date. Vectorul de întrerupere este interpretat în funcție de modul de lucru determinat de program.

Modul 0 – va interpreta vectorul de întrerupere ca un cod pe 8 biți ce va forța contorul de program PC la una din adresele: 0000, 0008, 0010, 0018, 0020, 0028, 0030, 0038 (toate în Hexazecimal). Codul este 11xxx111 unde xxx ia toate valorile între 000 şi 111, corespunzătoare locațiilor menționate. Codul este cel al instrucțiunii ***RST n***, (*Restart*) unde n = 0, 1, 2, . . . ,7.

Întrerupere INT n	*Codul întreruperii*	*Instrucțiunea ce se execută*	*Adresa la care se face saltul*
INT 0	11 **000** 111	RST 0	0 0 0 0 H
INT 1	11 **001** 111	RST 1	0 0 0 8 H
INT 2	11 **010** 111	RST 2	0 0 1 0 H
INT 3	11 **011** 111	RST 3	0 0 1 8 H
INT 4	11 **100** 111	RST 4	0 0 2 0 H
INT 5	11 **101** 111	RST 5	0 0 2 8 H
INT 6	11 **110** 111	RST 6	0 0 3 0 H
INT 7	11 **111** 111	RST 7	0 0 3 8 H

Microprocesorul poate deservi 8 cereri de întrerupere diferite, generate de 8 dispozitive inteligente, capabile să respecte protocolul de lucru impus. Fiecare dispozitiv se identifică prin codul pe care îl încarcă pe magistrala de date în momentul acceptării cererii de întrerupere. Acest cod este interpretat de procesor ca instrucțiune pe un octet; în setul de instrucțiuni, codul corespunde unei instrucțiuni de salt necondiționat la adresa din tabelul de mai sus. Între locațiile adresate sunt câte 8 locații libere, în care programatorul trebuie să încarce subrutinele de tratare pentru fiecare întrerupere. Dacă subrutinele necesită mai mult de 8 locații, se introduc instrucțiuni de salt în zone de memorie libere.

Modul 1 determină ca prima instrucțiune executabilă după acceptarea întreruperii va fi RST 7, care forțează execuția la adresa 0038 H.

Modul 2 utilizează cel mai eficient resursele hard şi soft ale lui Z80. Dispozitivul ce solicită întrerupere plasează un vector de adresă de 8 biți pe magistrala de date, pe durata ciclului maşină de recunoaştere întrerupere. Vectorul de 8 biți reprezintă octetul inferior de adresă (A_0-A_7) al subrutinei de tratare. Octetul superior (A_8-A_{15}) este furnizat de registrul intern I. În acest mod, subrutinele de tratare întrerupere pot fi plasate la orice adresă de memorie.

La recunoaşterea unei cereri de întrerupere de tip NMI, procesorul va executa salt necondiționat la adresa 0066 H, unde se află subrutina de tratare a cererilor NMI. Aşadar, cererile de întrerupere nemascabile sunt tratate numai în modul 1 (salt la adresă fixă).

Modul de tratare a cererilor INT se instalează prin instrucțiunile de comandă:

IM0 (*Interrupt Mode 0*) - se instalează modul M0;
IM1 (*Interrupt Mode 1*) - se instalează modul M1;
IM2 (*Interrupt Mode 2*) - se instalează modul M2;

Setul de instrucțiuni

Z80 recunoaşte şi execută 800 de instrucțiuni diferite. Este compatibil la nivel de cod cu Intel 8080 (recunoaşte şi execută toate instrucțiunile lui I 8080 sub formă binară, dar nu şi în limbaj de asamblare).

Principalele categorii de instrucțiuni sunt:

- Instrucțiuni de transfer de date pe 8 şi 16 biți;
- Instrucțiuni aritmetice şi logice;
- Deplasări stânga / dreapta şi rotații;
- Instrucțiuni orientate pe bit;
- Apel subrutine, revenire din subrutine;
- Operații de intrare/ieşire cu porturile (*Input, Output*);
- Instrucțiuni de control.

Instrucțiunile lui Z80 au o structură pe 1, 2, 3 sau 4 octeți.

1. Instrucțiuni pe un octet

Codul operației

2. Instrucțiuni pe doi octeți:

Cod op.	Cod op.	Cod op.	Cod op.
Cod op.	Adresă (8)	Data (8)	Deplasament (8)

3. Instrucțiuni pe trei octeți:

	Cod op.	Cod op.	Cod op.
	Data (8)	Adresă (8) inf.	Cod op.
	Data (8)	Adresă (8) sup.	Deplasament (8)

4. Instrucțiuni pe patru octeți:

Cod op.	Cod op.	Cod op.	Cod op.
Cod op.	Cod op.	Cod op.	Cod op.
Data (8)	Adresă (8) inf.	Deplasament (8)	Deplasament (8)
Data (8)	Adresă (8) sup.	Data (8)	Cod op.

Fig.12 Structura instrucțiunilor Z80.

10 Microprocesoare pe 16 biți

La elaborarea arhitecturii microprocesoarelor organizate pe opt biți, proiectanții au avut în vedere aproape în mod exclusiv aspecte hardware.

S-a urmărit, ca prin prin folosirea microprocesorului să se obțină reducerea numărului de componente electronice integrate pe scară joasă şi medie. Este adevărat că anumite soluții, adoptate în acea etapă, şi-au păstrat în mare măsură valabilitatea şi în cadrul microprocesoarelor dezvoltate ulterior. În acest sens, se pune în evidență în primul rând modul de comunicare cu echipamentele periferice. S-au dezvoltat interfețe specializate, programabile, care au fost preluate şi în generația microprocesoarelor pe 16 biți. În curând însă, microprocesoarele pe opt biți au arătat că prezintă limitări de principiu în atingerea unor performanțe superioare şi în primul rând datorită comunicării cu memoria principală, care se făcea pe o magistrală de date limitată la opt biți. S-au făcut atunci aprecieri potrivit cărora generația următoare de microprocesoare va produce o adevărată revoluție.

În arhitectura microprocesoarelor pe 16 biți nu apare de fapt o revoluție datorită dezvoltării circuitelor VLSI aşa cum se anticipase, ci se preiau concepte folosite la minicalculatoare, în fond respectându-se în mare măsură instoria dezvoltării calculatoarelor mari. Acelaşi lucru se remarcă şi pe partea software, în sensul că setul de instrucțiuni al microprocesoarelor a fost orientat spre implementarea limbajelor de programare de nivel înalt, a compilatoarelor etc.

10.1 Organizarea magistralelor

La microprocesoarele pe 16 biți, trăsătura fundamentală din acest punct de vedere este faptul că magistrala de date a fost extinsă la 16 biți.

Apare şi o extindere corespunzătoare a magistralei de adrese pentru a acoperi un spațiu de cel puțin un megaoctet. Magistrala de date poate să apară complet separată față de magistrala de adrese sau acestea pot fi multiplexate. În această ultimă soluție, aceeaşi magistrală este folosită atât pentru adrese cât şi pentru date, reducându-se astfel numărul de terminale externe ale microprocesorului.

În cazul magistralelor multiplexate se impune introducerea de registre pentru reținerea informației de adresă. În acest scop sunt

prevăzute semnale prin care se precizează natura informației poziționată la un moment dat pe magistrală.

Comunicarea cu memoria se poate face la nivelul unui cuvânt de 16 biți, dar s-a menținut şi posibilitatea ca schimbul de date între unitatea centrală şi memorie să se poată face la nivel de octet. În acest scop, s-au prevăzut semnale prin care se precizează dacă schimbul se face la nivel de cuvânt sau la nivel de octet.

10.2 Registrele interne

Prezența registrelor generale în arhitectura unui procesor vizează în primul rând creşterea vitezei de procesare a instrucțiunilor. Datele care sunt necesare în execuția operațiilor, nu se mai extrag din memorie ci sunt stocate temporar în registrele interne.

La microprocesoarele pe 16 biți, creşte numărul registrelor interne şi se diversifică posibilitățile de prelucare a datelor stocate în aceste registre, la nivel de bit, de octet, de cuvânt, în cod BCD etc.

Se estompează caracterul specializat al unor registre, în sensul că acestea pot fi utilizate în orice tip de operație, deşi se mențin anumite specializări (registre de date, registre de adresare).

Se remarcă, de asemenea, creşterea lungimii registrului de stare program, în care se introduc indicatori de condiții suplimentari față de cei existenți la microprocesoarele pe opt biți.

10.3 Facilități pentru întreruperi

La microprocesoarele pe 16 biți se regăsesc facilitățile prevăzute pentru generarea întreruperilor de la microprocesoarele pe opt biți, cu mențiunea că sporesc posibilitățile pentru tratarea acestora şi acest proces se desfăşoară în modul de operare “supervizor”.

O caracteristică inexistentă la microprocesoarele pe opt biți, dar prezentă la cele pe 16 biți, se referă la întreruperile de tip software. Spre deosebire de întreruperile externe care apar aleatoriu, fiind dependente de momentul apariției unui eveniment extern, întreruperile provocate pe cale software sunt sincrone. Ele se produc întotdeauna atunci când are loc execuția unor intrucțiuni introduse în acest scop.

10.4 Moduri de operare

Majoritatea microprocesoarelor pe 16 şi 32 de biți pot să funcționeze fie în modul numit “utilizator”, fie în modul denumit “supervizor”. În funcție de firma producătoare, cele două moduri se

regăsesc sub alte nume, spre exemplu: “normal” şi respectiv “sistem”.

Fiecare din aceste două moduri de operare este precizat prin starea unui bit dintr-un registru intern. La microprocesorul Motorola 68000, acest bit este notat cu *S* în registrul de date, în timp ce la Z8000 este notat cu *S/N*. În fiecare ciclu maşină, microprocesorul informează celelalte componente din sistem asupra modului de operare în care el funcţionează.

La Z8000, în acest scop, s-a prevăzut un semnal de ieşire cu semnificaţia *Normal/Sistem*. La microprocesorul Motorola 68000, se folosesc în acest scop trei semnale notate *FC2*, *FC1*, *FC0* prin a căror poziţionare se fac precizările necesare.

Acest mecanism bazat pe două moduri de operare, asigură securitatea sistemului. Programele utilizatorilor curenţi se execută în modul de operare “utilizator” şi prin aceasta li se permite accesul numai în zonele proprii alocate pentru cod şi data. Sistemul de operare se execută însă în modul “supervizor” şi are acces la toate resursele sitemului. Un număr de instrucţiuni care au efecte importante asupra sistemului, numite “instrucţiuni privilegiate”, pot fi executate numai când microprocesorul funcţionează în modul “supervizor”. În acestă categorie intră instrucţiunile de intrare / ieşire, instrucţiunile de validare şi invalidare a întreruperilor, instrucţiunile prin care se modifică indicatorii de stare şi alte instrucţiuni. S-au prevăzut instrucţiuni prin care dintr-un program utilizator se poate trece într-un mod controlat în modul “supervizor”. În acest scop, la Z8000 s-a introdus instrucţiunea *SC (Supervizor Call)*, iar la Motorola 68000 instrucţiunea *TRAP*. Trecerea din modul de lucru “supervizor” în modul de lucru “utilizator” poate fi făcută prin executarea unor instrucţiuni privilegiate ce modifică starea acelui bit indicator din registrul de stare care precizează modul de lucru al microprocesorului.

10.5 Prelucrarea instrucţiunilor

O caracteristică arhitecturală specifică microprocesoarelor pe 16 biţi este extragerea anticipată a instrucţiunilor şi executarea acestora în regim “*pipeline*” , ceea ce în limba română se traduce adesea prin “*bandă de asamblare*”. Procesul are la bază o idee simplă. Un proces de calcul mai complex se subdivide în procese de complexitate mai redusă, fiecare subproces fiind executat într-o subunitate hardware specializată.

Subunităţile hardware se succed una după alta, încât se poate spune că formează o conductă sau o bandă de asamblare. Prima subunitate recepţionează continuu informaţia de intrare, o prelucrează conform atribuţiilor sale şi furnizează la ieşire rezultatul într-un anumit moment. Ieşirea primei subunităţi consituie intrare pentru unitatea a doua. La un

moment dat, conducta este “plină” şi toate subunităţile hardware ce o compun funcţionează simultan. Acest principiu referitor la prelucrarea instrucţiunilor, este exemplificat în fig.1. Procesul de prelucrare a fost divizat în cinci subprocese: extragerea instrucţiunii (F), decodificarea (D), calculul adresei efective (A), extragerea operandului (O) şi execuţia propriu-zisă a operaţiei (E).

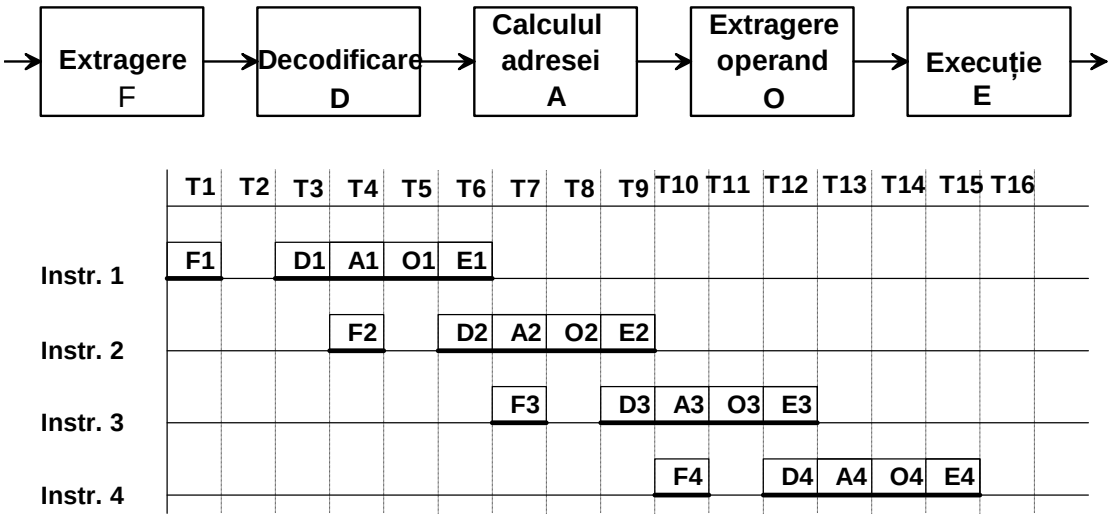

Fig. 1 Prelucrarea "pipeline" a instrucţiunilor

În fig.1 au fost luate în consideraţie 4 istrucţiuni. Rezultă că sunt necesare numai 15 perioade de tact (stări) pentru execuţia lor completă, faţă de 24 de stări în cazul în care anumite operaţii nu se execută simultan. Desfăşurarea în timp a operaţiilor este bazată pe utilizarea magistralelor externe pentru extragerea instrucţiunii următoare pe durata operaţiilor de decodificare şi calcul adresă din instrucţiunea precedentă. De asemenea, decodificarea unei instrucţiuni se poate face simultan cu execuţia celei precedente, deoarece operaţiile sunt realizate de unităţi diferite din structura microprocesorului. Practic, procesul este mai complex, deoarece nu toate instrucţiunile au aceeaşi lungime şi durată de execuţie.

O altă caracteristică arhitecturală a microprocesoarelor pe 16 biţi o reprezintă extragerea anticipată (*look-ahead*) a instrucţiunilor. La baza acestui mecanism se găseşte o memorie tampon în care se încarcă un număr de instrucţiuni şi operanzi din memoria principală în avans faţă de instrucţiunea curentă. Acest proces se declanşează în timpul execuţiei unei instrucţiuni, dacă magistralele externe sunt temporar libere.

Procedându-se în acest mod, numărul de instrucţiuni prelucrate într-o secundă creşte, deoarece instrucţiunile vor fi preluate direct din memoria tampon internă, fiind eliminat timpul relativ lung de aducere din memoria principală. Creşterea de viteză este însă de tip statistic, pentru că

intervin probleme greu de soluționat la modul general. De exemplu, efectul de aducere anticipată a instrucțiunilor poate fi anulat când instrucțiunea curentă este de salt iar instrucțiunea țintă nu se găseşte în memoria tampon; în acest caz, informația din memoria tampon este anulată şi începe încărcarea acesteia cu instrucțiuni de la adresa de salt. Dacă apar frecvent instrucțiuni de salt, extragerea anticipată devine inutilă.

La microprocesorul 8086, pentru procesarea instrucțiunilor există două unități (fig.2). Unitatea de interfață cu magistrala, BIU (*Bus Interface Unit*), care asigură conectarea cu memoria, realizează extragerea instrucțiunilor şi operanzilor din memorie; instruțiunile sunt stocate într-o memorie tampon de şase octeți. Unitatea de execuție, notată prescurtat EU (*Execution Unit*) realizează decodificarea, calculul adresei efective, preluarea operandului de la BIU şi execuția propriu-zisă a instrucțiunilor pe care le preia din fişierul de instrucțiuni. De asemenea, unitatea EU transmite spre BIU datele care trebuie stocate în memoria externă.

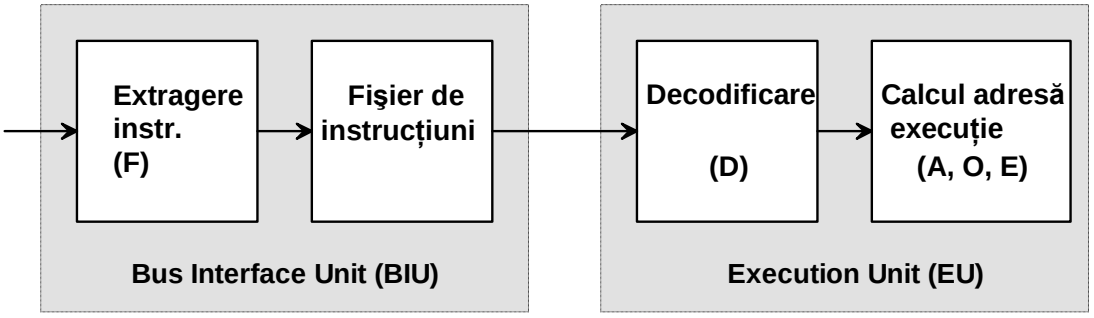

Fig.2 Unitățile de prelucrare la Intel 8086.

La microprocesorul Z8000, proiectanții nu au adoptat o arhitectură cu memorie tampon în care să se formeze un şir de aşteptare pentru câteva instrucțiuni, ci au prevăzut un mecanism tip *pipeline* cu extragerea anticipată a unei singure instrucțiuni, suprapunând prelucrarea datelor din instrucțiunea curentă cu extragerea din memorie a următoarei instrucțiuni.

Motorola 68000 este proiectat, de asemenea, pe baza conceptului de "bandă de asamblare", în sensul că extragerea instrucțiunii se suprapune în timp cu execuția acesteia.

10.6 Administrarea memoriei

Memoria a constituit resursa principală calculatoarelor şi conțină încă să se mențină în această poziție. Din punct de vedere al unității centrale în raport cu memoria, se manifestă două cerințe:

- memoria să nu constituie un factor limitativ al vitezei de execuție, adică, schimbul de informații să nu micşoreze numărul de instrucțiuni executate în unitatea de timp. Dacă, de exemplu, timpul de acces la memorie este de 100 nanosecunde şi o instrucțiune conține 4 octeți, care se extrag consecutiv din memorie, pentru extragerea instrucțiunii se consumă 400 ns.

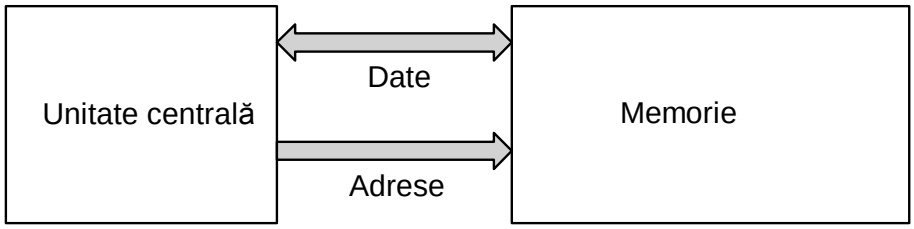

Fig.3 Canalul de comunicație cu memoria

- a doua cerință se referă la capacitatea memoriei; creşterea continuă a complexității aplicațiilor ce operează pe un mare volum de date conduce la necesitatea de a dispune de memorie cu capacitate de stocare tot mai mare, în cazul ideal de capacitate practic nelimitată, pentru utilizator.

Aceste două cerințe privind memoria, de viteză şi de capacitate de stocare, se soluționează printr-un compromis, întrucât problemele tehnologice ce apar sunt greu de rezolvat.

La microprocesoarele organizate pe 16 biți, schimbul de date cu memoria se face pe o magistrală de 16 linii. Față de microprocesoarele organizate pe opt biți, debitul de transfer al datelor pe canalul de legătură cu memoria, se dublează.

În ceea ce priveşte capacitatea de stocare a memoriei, la microprocesoarele pe 16 biți, magistrala de adrese a fost, de asemenea, extinsă la 20 de linii, pentru a acoperi un spațiu de adresare de un megaoctet. Extinderea cu mult peste această limită a magistralei de adrese determină creşterea numărului de terminale, care la microprocesoarele de 8 şi 16 biți a rămas la 40 din motive tehnologice.

În stabilirea arhitecturii microprocesoarelor de 16 biți s-a avut în vedere necesitatea existenței unor mecanisme prin care să se asigure o administrare eficientă a memoriei.

10.7 Spațiul adreselor logice

De regulă, la microprocesoarele pe opt biți, adresa generată în timpul execuției unui program şi poziționată pe magistrala de adrese de către microprocesor, se foloseşte direct în adresarea unei locații de memorie sau a unui dispozitiv de intrare/ieşire.

La microprocesoarele pe 16 biţi se introduc noţiunile de **adresă logică** şi de **adresă fizică**. Adresele generate de un program sunt considerate adrese logice şi totalitatea acestora formează spaţiul adreselor logice. Totalitatea adreselor ce corespund memoriei (sau dispozitivelor de intrare / ieşire) formează spaţiul adreselor fizice.

Cele două spaţii, al adreselor logice şi al adreselor fizice, pot să nu fie egale. Arhitectura microprocesoarelor pe opt biţi nu a fost concepută a suporta acest mod de tratare a adreselor. La aceste microprocesoare, adresele logice coincid cu cele fizice.

La microprocesoarele pe 16 biţi este necesar să existe un mecanism de conversie a adreselor, prin care adresele logice să fie translatate în adrese fizice. Adresa instalată de microprocesor pe magistrala de adrese poate să coincidă cu adresa fizică sau poate fi diferită de aceasta. Astfel, la microprocesorul 8086, mecanismul de translatare este inclus în microprocesor şi adresa instalată pe magistrala de adrese se foloseşte direct în adresarea memoriei.

La microprocesoarele Zilog Z8000, Motorola 68000, NS16032, mecanismul de translatarea adreselor logice în adrese fizice este preluat de un circuit extern, denumit “unitate de administrare a memoriei” (*Memory Management Unit, MMU*). În acest caz, adresa logică generată de către microprocesor nu este aceeaşi cu adresa fizică ce va fi transmisă spre memorie sau spre dispozitivele de intrare/ieşire.

De regulă, la microprocesoarele pe opt biţi, spaţiul adreselor este considerat *liniar*, în sensul că acesta începe de la adresa cu valoarea zero şi continuă până la valoarea maximă ce rezultă din numărul total de biţi cu care se exprimă adresa.

Acest mod de organizare a spaţiului de adresare prezintă unele dificultăţi atunci când mai mlte programe trebuie să-şi împartă memoria sistemului, respectiv atunci când aceste programe şi zonele de date asociate trebuie să fie protejate unele de celelalte.

Dacă nu există un mecanism de administrare a memoriei, atunci aceste sarcini trebuie să fie preluate de către programele utilizator. Una din cerinţele fundamentale pentru o administrare şi alocare eficientă a memoriei o reprezintă distincţia ce trebuie introdusă între spaţiul adreselor logice şi spaţiul adreselor fizice, respectiv independenţa adreselor logice de adresele fizice.

Această cerinţă este totodată şi o primă condiţie în realizarea sistemelor cu memorie virtuală. În acest caz, spaţiul logic este pus în corespondenţă atât cu spaţiul fizic oferit de memoria principală, cât şi cu spaţiul oferit de memoriile externe pe discuri. Programatorul, în aceste condiţii îşi elaborează programele fără a fi constrâns de capacitatea de stocare oferită de memoria principală. La realizarea unui sistem cu

memorie virtuală se va introduce un mecanism care să constate automat dacă elementul adresat se găseşte în memoria principală sau în memoria auxiliară. În cazul când accesul vizează un element ce nu se găseşte în memoria principală, atunci execuția instrucțiunii este suspendată şi intervine sistemul de operare ce va aduce elementul rspectiv în memoria principală. După executarea acestor transferuri, se revine la programul aflat în execuție curentă prin reexecuția instrucțiunii anterior suspendate.

O caracteristică importantă ce trebuie să existe în arhitectura microprocesoarelor pentru a se putea realiza o memorie virtuală, se referă la posibilitatea ca execuția unei instrucțiuni să fie întreruptă înainte ca aceasta să fie complet executată. După aducerea elementului adresat din memoria externă în memoria principală, instrucțiunea respectivă va fi reluată. Transferurile între memoria principală şi cea externă (pe disc) nu se face la nivelul unui singur element, ci pe blocuri de elemente (segmente sau pagini).

Procedând astfel, rezulă că adresele folosite de un programator, respectiv cele generate în cadrul unui program pentru localizarea unui operand vor fi adrese logice. În programele utilizatorilor, niciodată nu vor apărea adrese fizice. Programele utilizator şi datele asociate acestora se găsesc în spațiul adreselor logice şi nu depind de capacitatea memoriei operative, fizic prezentă într-un sistem. Există câteva modalități de organizare a spațiului adreselor logice şi de formare a adreselor logice. O continuitate de adrese logice nu conduce la necesitatea de a avea şi o continuitate în adresele fizice.

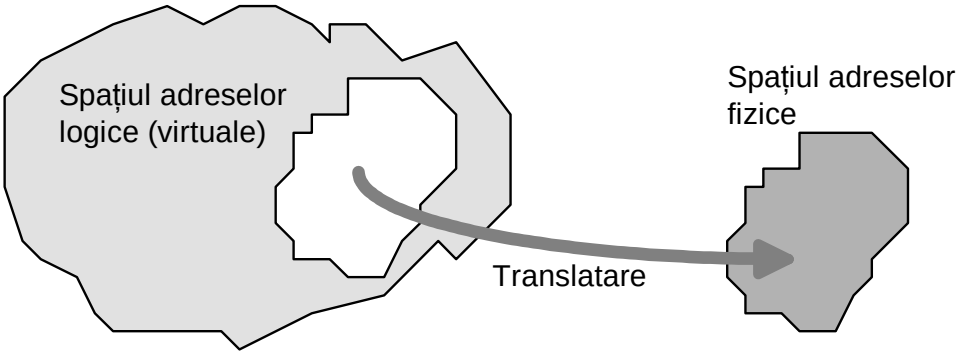

Fig. 4 Relația dintre spațiul adreselor virtuale şi spațiul adreselor fizice

În figura 4 se pune în evidență faptul că prin crearea a două spații în ceea ce priveşte adresele, unul logic şi celălalt fizic, este necesar să apară un mecanism prin care să se asigure o corespondență între cele două tipuri de adrese.

Dacă în sistem sunt active două programe diferite, A şi B, fiecare are un spațiu propriu de adrese fizice, chiar dacă spațiile de adrese logice cincid total sau parțial. În acest caz, mecanismul de translatare a adreselor

este mai complex, fiind asistat de sistemul de operare, care gestionează memoria fizică şi dispune în orice moment de informaţii referitoare la zonele libere, zonele rezervate şi zonele temporar ocupate, de memorie.

Pentru utilizator, este important modul în care este structurat spaţiul adreselor logice, deoarece acesta determină scrierea adreselor în program.

- Dacă spaţiul logic este liniar, adică are o singură dimensiune, adresele logice se exprimă prin numere naturale 0, 1, 2, . . . ,n. La Motorola 68000 şi NS 16032, spaţiul adreselor logice este liniar.
- Dacă spaţiul logic este neliniar, atunci are cel puţin două dimensiuni şi ca urmare o adresă logică se exprimă prin doi sau mai mulţi parametri. La Intel 8086 şi Z 8000, adresa logică se exprimă prin doi parametri: *segment* şi *offset*.

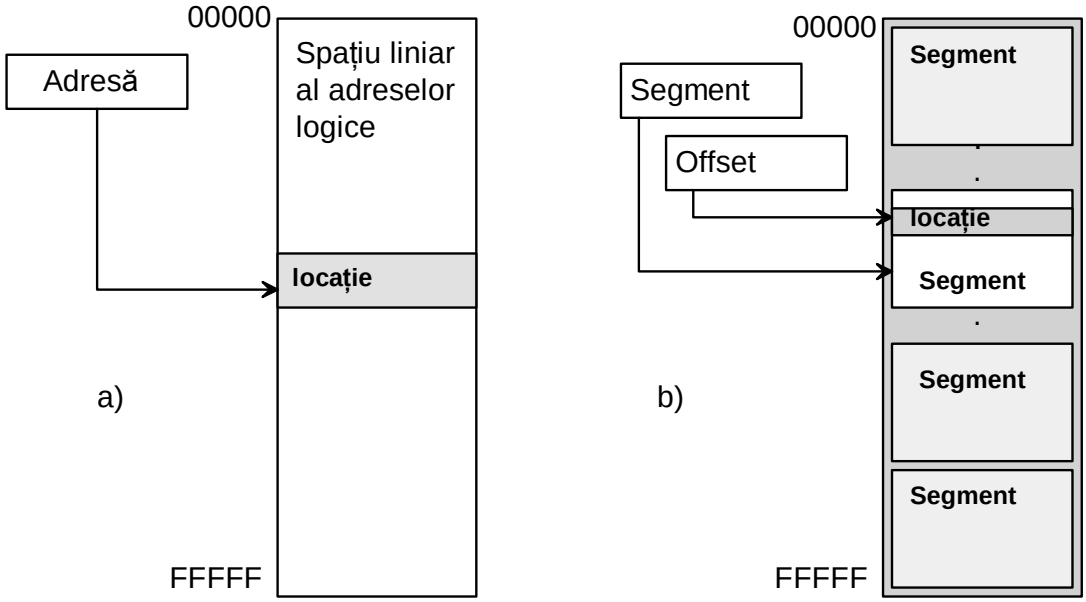

Fig. 5 Spaţiul adreselor logice în cazul liniar (a) şi în cazul neliniar (b)

10.8 Alocarea memoriei

La scrierea programelor se utilizează adrese logice, adrese ce sunt independente de spaţiul adreselor fizice ocupate atunci când programul se va încărca efectiv în memorie.

Înainte ca un program să fie executat, este necesar ca acestuia să i se aloce un spaţiu în memoria fizic prezentă în sistem.

Dacă această atribuire de spaţiu se face la încărcarea programului în memoria operativă, atunci procedeul e numit **alocare statică** şi prezintă o serie de dezavantaje. În sistemele evoluate, se foloseşte **alocarea dinamică** a memoriei ce se caracterizează prin atribuirea spaţiului

necesar programelor sau proceselor în faza de execuţie a acestora. La alocarea dinamică a memoriei, se utilizează diferite procedee. Fiecare din acestea se bazează pe o divizare a spaţiului adreselor logice în blocuri ce cuprind adrese logice consecutive.

Într-o primă metodă, divizarea se face în blocuri numite segmente, de lungime arbitrară. Fiecare segment este adresat printr-un nume interpretat software. Segmentul constituie unitatea de bază în alocarea memoriei şi în transferul ce are loc între memoria operativă şi cea externă.

În cea de-a doua metodă, bazată pe pagini, spaţiul fizic de adresare este divizat în blocuri de aceeaşi lungime, "cadru-pagină" iar spaţiul logic în blocuri de aceiaşi mărime, numite "pagini". Subdivizarea spaţiului logic în pagini o face în mod efectiv sistemul de operare, în timp ce numele paginilor este interpretat prin hardware. În acest caz, pagina reprezintă unitatea de bază în alocarea memoriei operative şi în transferurile cu memoria externă. Se reţine faptul că în această metodă, unitatea folosită în alocarea memoriei este constantă, spre deosebire de unitatea segment. În mod uzual, paginile au dimensiuni mai mici comparativ cu dimensiunea adoptată pentru un segment.

Combinând avantajele celor două metode, se procedează la folosirea combinată a segmentelor şi a paginilor, spaţiul logic fiind organizat în segmente de lungime variabilă, şi acestea, la rândul lor, în pagini cu dimensiune fixă.

10.9 Translatarea adreselor

În toate metodele de alocare dinamică a memoriei, este necesar să existe un mecanism prin care să se realizeze o translatare a adreselor logice în adrese fizice ale memoriei operative. Această translatare are loc la faza de execuţie a programului şi mecanismul, printre altele, cuprinde o tabelă prin care se face corespondenţa între adresele fizice şi cele logice.

Principiul de funcţionare este pus în evidenă în figura 6.

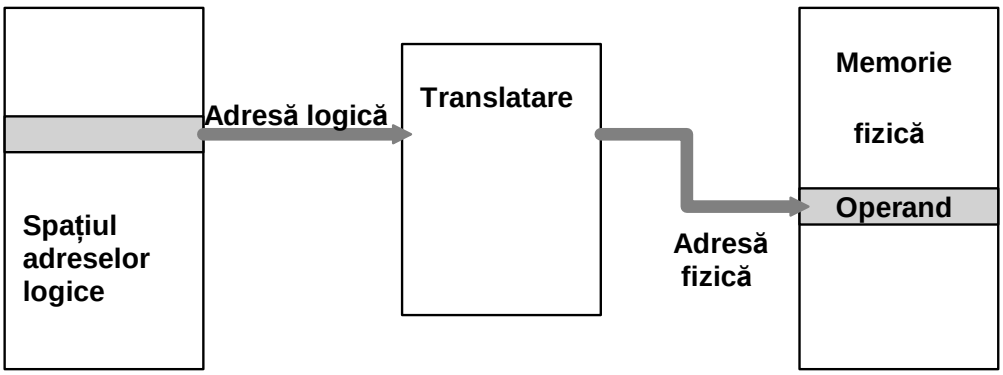

Fig.6 Principiul translatării adreselor logice în adrese fizice

În cazul cel mai general, tabela de translatare poate fi interpretată ca un set de registre ce se interpune între microprocesor şi memoria operativă. Unitatea de adresare generează adresa logică; în cadrul unității de translatare se realizează următoarele acțiuni:

1. Dacă operandul adresat nu se găseşte în memoria operativă, atunci se face apel la memoria externă; sistemul de operare va fi informat în acest sens pentru a întreprinde acțiunile ce se impun. Acestea se referă la extragerea operandului din memoria externă, aducerea lui în memoria operativă şi transmiterea către unitatea de translatare a adresei fizice unde a fost depus.

2. Dacă operandul adresat se găseşte în memoria operativă, atunci unitatea de translatare determină adresa fizică a acestuia şi o transmite spre memorie.

Un mecanism de translatare cu un singur nivel este ineficient din punct de vedere practic, întrucât tabela de translatare necesită un spațiu de memorare comparabil cu memoria principală.

Soluția este divizarea spațiului de adresare în segmente şi pagini, ceea ce are ca efect reducerea considerabilă a dimensiunii tabelei de translatare, întrucât referințele se fac la nivel de blocuri (segmente). Numărul locațiilor din tabela de translatare determină numărul maxim de segmente sau pagini din memoria operativă. O locație din tabelă corespunde unui segment de memorie.

În principiu, tabela de translatare poate fi realizată efectiv fie prin mijloace software (o zonă din memorie), fie prin mijloace hardware.

Sistemul de operare va reactualiza conținutul tabelei de translatare ori de câte ori va avea loc o schimbare a amplasării programelor şi datelor în memorie.

În sistemele cu multiprogramare, unde mai multe programe sunt active simultan, vor exista mai multe tabele de translatare, câte una pentru fiecare program. Dacă se foloseşte o singură tabelă de translatare, atunci sistemul de operare va proceda la actualizarea tabelei, atunci când se întrerupe execuția unui program şi se comută execuția către altul.

Când se folosesc mai multe tabele, realizate fie software, fie hardware, este necesar să se prevadă posibilitatea de a selecta la un moment dat tabela de translatare corespunzătoare programului de execuție.