2. Caracterizarea microprocesoarelor pe 8 biţi

În continuare se va descrie schema bloc funcţională a unui μP standard al cărui cuvânt de date are lungimea 8 biţi (μP8). Noţiunea *standard* se referă la faptul că μP îndeplineşte rolul UCP într-o maşină de tip von Neumann. După cum s-a arătat asemenea maşină este structurată în trei unităţi (*UAL, memorie, dispozitive de intrare / ieşire*) care comunică între ele printr-o unică magistrală cu secţiuni de *date, adrese şi comenzi*.

Presupunând existenţa unui program stocat în memorie al cărui sfârşit este marcat de o variabilă logică **SFP,** structura μP8 rezultă din necesitatea execuţiei programului conform următorului algoritm:

Repetă

- adresare şi aducere din memorie a codului;
- decodificare instrucţiune;
- execuţie instrucţiune;

până când SFP=adevarat.

Pornind de la această secvenţiere structura unui μP8 va fi descrisă pe 5 niveluri de detaliere .

2.1. Nivelul 1 de caracterizare

Nivelul 1 este asociat registrelor de date (RD) şi de adrese (RA). Aceste registre apar la interfaţa µP cu magistrala de date (MD) şi respectiv de adrese (MA), figura 2.1.

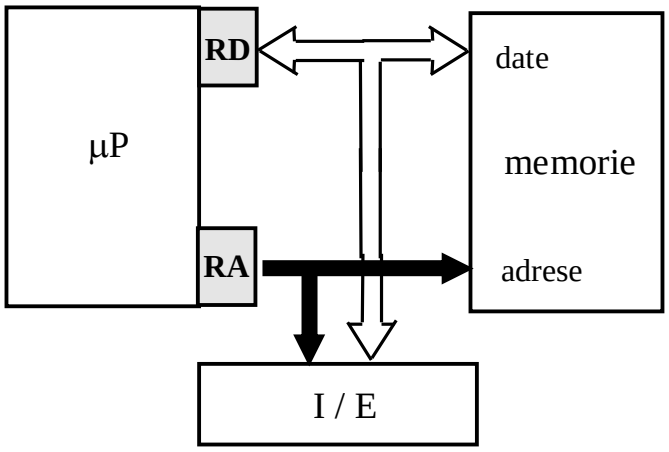


Fig. 2.1. Nivelul 1 de detaliere al unui µP8

RD este bidirecţional ca şi MD şi are lungimea egală cu a acesteia (8 biţi). O informaţie provenită din P este disponibilă unităţilor conectate la MD numai după înscrierea acesteia în RD. Invers, o informaţie destinată P este accesibilă acestuia tot numai după înscrierea în RD.

RA este unidirecţional şi are lungimea impusă de caracteristicile unităţii de control a adresării memoriei. RA are rolul de a menţine ferm pe MA adresa furnizată de UCP până la localizarea corectă a informaţiei în memorie sau în porturile intrare-ieşire. În acest context *portul* reprezintă o adresă de memorie care identifică circuitul fizic utilizat la transferul informaţiei între P şi periferic.

Atât RD cât şi RA sunt transparente pentru utilizator, acestea nefiind atribute de arhitectură.

2.2. Nivelul 2 de caracterizare

Acest nivel este înglobează registrele generale (RG). Acestea reprezintă practic memoria internă a P şi constituie nivelul de memorie cel mai rapid adresabilă într-un sistem. Funcţia lor este de stocare temporară a datelor (operanzi şi rezultate). După cum se observă din figura 2.2, accesul fizic la RG se face cu ajutorul unui multiplexor (MUX) pentru a citi un registru respectiv cu ajutorul unui demultiplexor (DMUX) pentru a înscrie un registru.

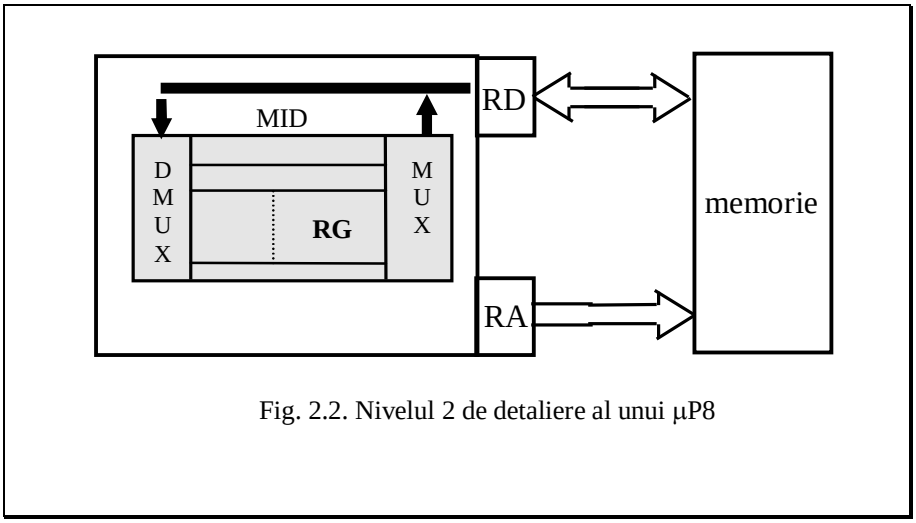


Fig. 2.2. Nivelul 2 de detaliere al unui μP8

După cum se observă din figura 2.2, legătura internă dintre RD şi RG este realizată prin *magistrala internă de date* (MID) care constituie o prelungire a magistralei de date (MD) a sistemului în interiorul P. La MID se vor conecta toate blocurile interne care au acces la informaţia vehiculată prin MD.

Numărul de RG şi lăţimea MID constituie criterii de performanţă pentru orice P. În ceea ce priveşte lătimea MID, nu este obligatoriu ca aceasta să fie egală cu a MD externe. RG sunt în totalitate la dispoziţia utilizatorului ele constituind *elemente de arhitectură.*

De exemplu, P 8080 avea un număr de 6 RG pe câte 8 biţi, *B,C,D,E,H,L* care pot fi utilizate ca atare dar şi în perechi, pentru a forma 3 registre de 16 biţi (*B-C* denumit registrul *B*, *D-E* denumit registrul *E şi H-L* care constituie registrul *H*.

2.3. Nivelul 3 de caracterizare

Nivelul 3 include unitatea aritmetică de procesare (UAP). Acest bloc funcţional, a cărui structură este prezentată în figura 2.3, reprezintă suportul activităţii de prelucrare a datelor.

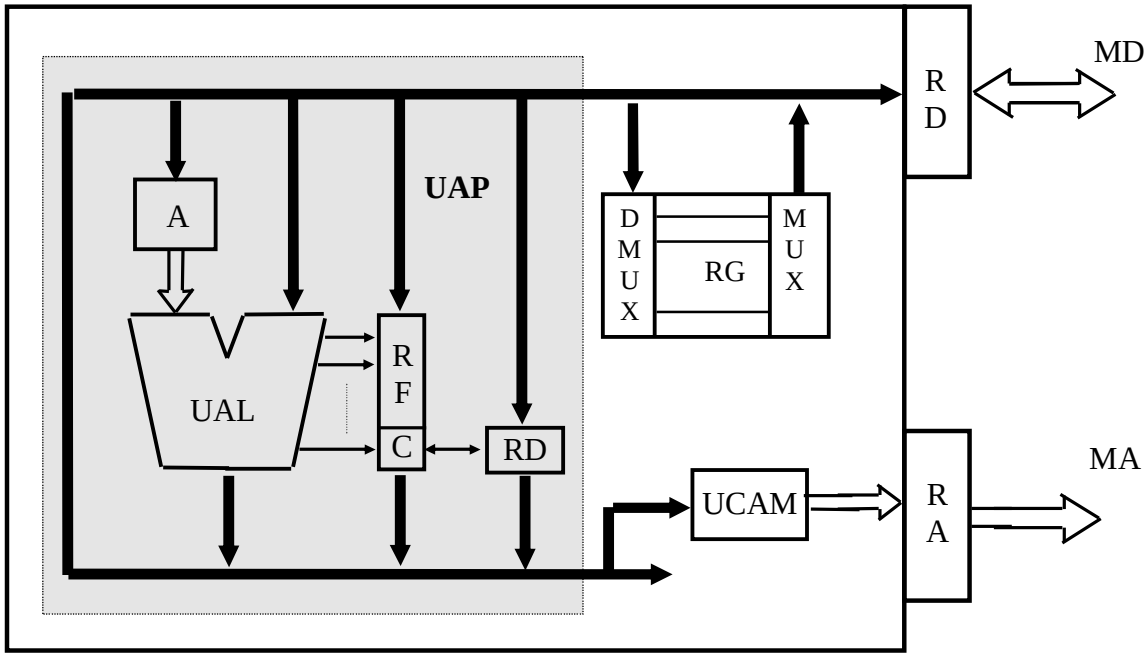


Fig. 2.3. Nivelul 3 de caracterizare a unui μP8:
UAL - unitate aritmetico - logică; A - registru acumulator; RF - registru al fanioanelor; C - fanion (indicator) de transport; RD - registru de deplasare; UCAM - unitate de control a aresării memoriei.

După cum s-a arătat, UAL reprezintă un circuit combinaţional care asigură realizarea unor funcţii aritmetice (*adunare, scădere, complementare faţă de 2, incrementare, decrementare, ajustare zecimală etc*.) şi logice (*SI, SAU, NICI, NUMAI, SAU EXCLUSIV, complementare faţă de 1, etc*).

Tipul şi numărul funcţiilor realizate de UAL constituie un criteriu de performanţă al P care se reflectă într-un atribut de arhitectură şi *anume subsetul de instrucţiuni de prelucrare a datelor*. În afara intrărilor şi ieşirilor de date, UAL mai are şi intrări de selecţie a funcţiilor, care însă nu sunt reprezentate în figura 2.3.

Acumulatorul (A) este un registru asemănător cu cele din setul de RG. Prin definiţie A conţine un operand al UAL şi în urma efectuării prelucrării, rezultatul. Având în vedere această dublă funcţionalitate a A, precum şi caracterul combinaţional al UAL, structura din figura 2.3 este nefuncţională datorită faptului că UAL se alimentează la infinit cu rezultatul propriei prelucrări. Eliminarea acestei situaţii critice se face prin includerea în structură a câte unui acumulator şi registru (temporare), evidenţiate în figura 2.4. Cei doi operanzi sunt menţinuţi în

registrele temporare până când rezultatul prelucrării se înscrie în A. Cele două registre sunt complet transparente pentru utilizator, ele neconstituind elemente de arhitectură.

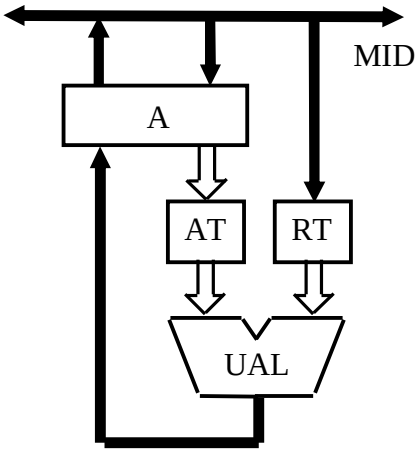


Fig. 2.4. Registre temporare aferente UAP:
AT - acumulator temporar;
RT - registru temporar.

Fanioanele (indicatorii de condiţie) reprezintă bistabili pentru memorarea unor condiţii speciale apăruta în funcţionarea P şi în special a UAL. Ele pot fi grupate într-un registru al fanioanelor, accesibil utilizatorului.

În figura 2.5 se reprezintă conţinutul acestui registru pentru P INTEL 8080, reprezentativ pentru clasa P8, în care semnificaţiile fanioanelor sunt următoarele:

- S (SIGN) - are valoarea **1** dacă rezultatul prelucrării din *A* este pozitiv (bitul cel mai semnificativ este **1**), altfel S are valoarea **0**;

- Z (ZERO) - este pus în **1** dacă în urma prelucrării, *A* conţine valoarea **0**, altfel Z este **0**;

- P (PARITY) - are valoarea **1** dacă în urma execuţiei unei instrucţiuni, *A* conţine un număr par de **1**, altfel P este **0**;

- C (CARRY) - este **1**, dacă din *A* s-a efectuat un transport, altfel C este **0**.

- AC (AUXILIARY CARRY) - este **1** în condiţiile existenţei unui transport dinspre bitul 3 spre bitul 4 al *A*, altfel AC este **0**.

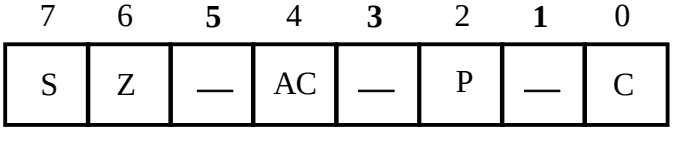


Fig. 2.5. Structura registrului fanioanelor pentru μP 8080.

Fanioanele joacă un rol important în structurarea programelor, întrucât instrucţiunile de salt condiţionat testează starea acestora. RF constituie element de arhitectură, iar numărul de

fanioane un criteriu de performanţă pentru P. Uzual A şi RF se asamblează într-un unic registru de 16 biţi cunoscut sub denumirea de *registru **PSW*** (***P***rogramm ***S***tatus ***W***ord).

Registrul de deplasare (RD) ocupă un loc aparte în structura oricărui P8 deoarece cu ajutorul lor se realizează operaţiile de înmulţire şi împărţire cu puteri ale lui 2. Deplasările presupun de regulă salvarea bitului extrem (0 sau 7) în fanionul C. O variantă interesantă o reprezintă *rotaţia* în care conţinutul fanionului C este înscris în celălalt bit extrem al RD. În figura 2.6 sunt evidenţiate cele două maniere de realizare a deplasării. RD nu este un atribut de arhitectură, el este implicit utilizat de instrucţiunile specifice ale P, dar operatorul nu are acces nemijlocit la acest registru special.

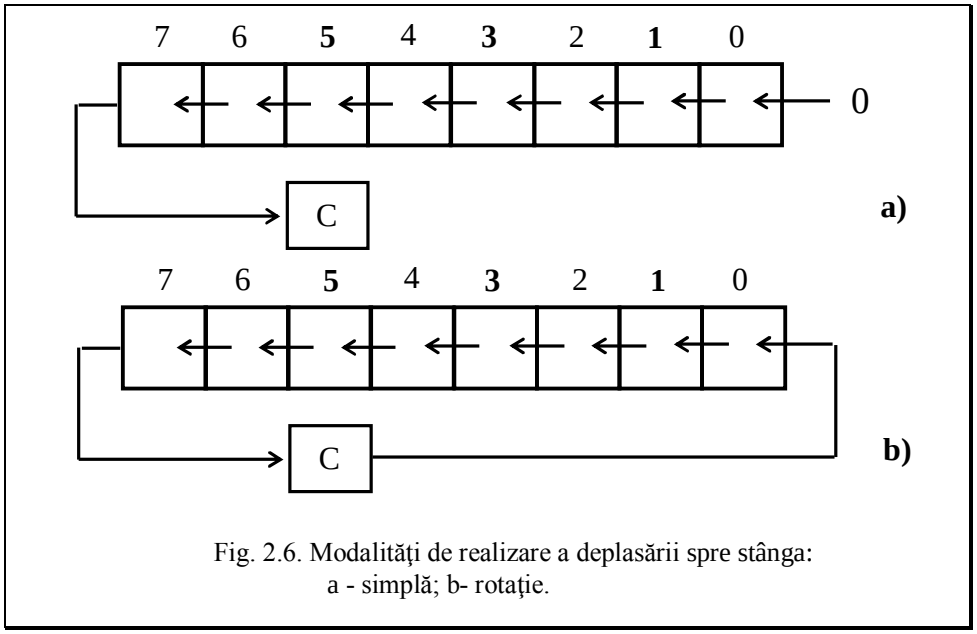


Fig. 2.6. Modalităţi de realizare a deplasării spre stânga:
a - simplă; b- rotaţie.

O secvenţă de utilizare implicită a RD ar putea fi următoarea:

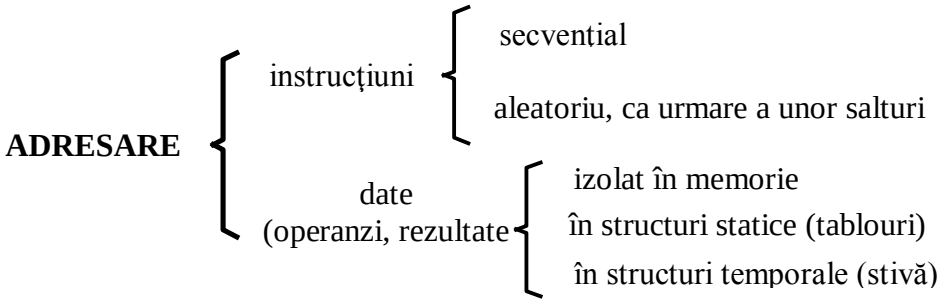
1 - se selectează registrul care conţine operandul;
2 - conţinutul acestuia este adus pe MID;
3 - se înscrie operandul în RD;
4 - se comandă acestuia funcţia de deplasare dorită;
5 - conţinutul RD (deplasat) este transferat pe MID;
6 - se selectează din nou registrul de la pasul 1;
7 - se înscrie în acesta rezultatul operaţiei de deplasare care ia locul operandului iniţial.

Funcţionarea RD este posibilă, dacă acesta este conectat la fanionul C, aspect evidenţiat şi de figura 2.3.

2.4. Nivelul 4 de caracterizare

Acest nivel defineşte unitatea de control a adresării memoriei (UCAM). Această unitate, a cărei conectare cu magistralele interne este evidenţiată în figura 2.7 realizează încărcarea unei adrese în RA în vederea localizării unei informaţii în memoria sau porturile calculatorului.

Structura sa derivă din funcţiile acesteia sintetizate mai jos.



Realizarea acestor funcţii, implică existenţa în structura UCAM, în totalitate sau nu, a următoarelor elemente principale:

- numărător de program;
- indicator de stivă;
- registru index,

pentru care în figura 2.7 se prezintă o posibilitate de interconectare.

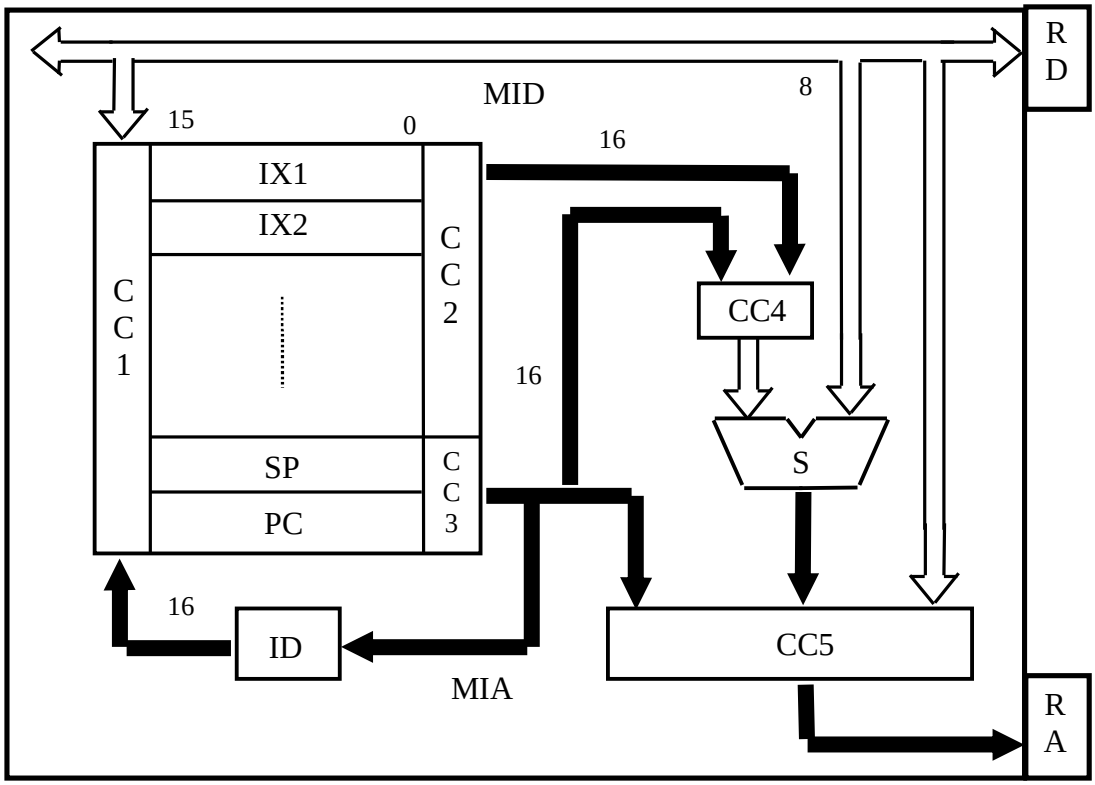


Fig. 2.7. Structura posibilă a unei unităţi de control a adresării memoriei:
PC - numărător de program; SP - registru indicator de stivă; IX - registre index; S - sumator; IDN - circuit de incrementare/decrementare; CC - circuite de conectare ; MIA - magistrală internă pentru formarea adresei.

Numărătorul de program (PC-Programm Counter) conţine adresa fizică (AF) a instrucţiunii ce urmează a fi executată. AF reprezintă forma sub care se furnizează o adresă pe MA pentru a se face identificarea fizică a locaţiei de memorie vizate.

Lungimea PC impune capacitatea maximă a memoriei ce poate fi adresată în sistem. Uzual pentru un μP8, PC are 16 biţi de unde rezultă o hartă a memoriei direct adresabile de *2^{16}ş65536 octeţi ş 64KB (1 Koctet ş 1 Kbyte ş 1024 octeţ*i).

Adresarea secvenţială a memoriei presupune transmiterea conţinutului registrului PC (respectiv a adresei locaţiei adresate) pe calea CC3 - CC5 la RA, urmată de incrementarea PC de către circuitul ID.

Schema din figura 2.7 permite saltul în memoria program (*noua adresă însoţeşte codul instrucţiunii ce urmează a se executa*) conform următoarei secvenţe:

- RA se încarcă prin CC5 direct de pe MID cu adresa instrucţiunii care urmează a se executa;

- concomitent PC se va încărca prin CC1 cu aceeaşi adresă, de unde îşi va continua funcţionarea secvenţială.

PC nu este un atribut de arhitectură, programatorul neputând modifica direct conţinutul acestuia.

Indicatorul de stivă (SP - Stack Pointer). Stiva (*Stack*) reprezintă o structură de date utilizată pentru păstrarea temporară a datelor. Stiva este organizată pe principiul **LIFO** (**L**ast **I**nput - **F**irst **O**utput — ultimul intrat - primul ieşit). Poziţia ocupată în stivă de ultimul element introdus constituie vrful stivei, încărcarea şi descărcarea acesteia putându-se efectua numai prin acest punct. După cum se observă din figura 2.8 prin operaţiile de încărcare descărcare se modifică adresa *vârfului* stivei.

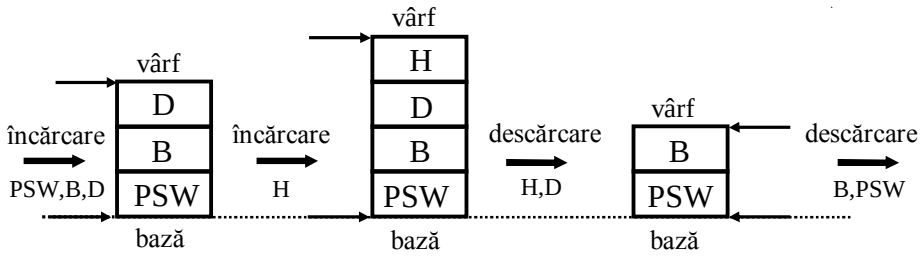


Fig. 2.8. Încărcarea şi descărcarea unei stive:
PSW - registrul stare program; B,C,D - registre duble.

Încărcarea şi descărcarea stivei se realizează prin instrucţiuni specifice (**PUSH** pentru scriere în stivă, **POP** pentru extragere din stivă). De exemplu pentru situaţia din figura 2.8 succesiunea de instrucţiuni este următoarea:

.
.
.

PUSH PSW; înscrie în stivă conţinutul PSW

PUSH B; idem B

PUSH D; idem D
PUSH H; idem H
.
.
.
POP H; extrage din stivă conţinutul H
POP D; idem D
POP B; idem B
POP PSW; idem PSW
(reamintim că PSW, B, D, H sunt registre duble pe câte 16 biţi).

SP conţine adresa curentă a vârfului stivei. Configurarea unei zone de memorie ca stivă se face prin înscrierea în SP a adresei bazei. Stiva se organizează astfel încât creşterea ei să se facă "în jos" adică în sensul descreşterii adreselor. La fiecare înscriere în stivă, SP este decrementat iar la fiecare extragere, acesta este incrementat, astfel încât în orice moment va conţine adresa primei locaţii disponibile din memoria stivă. Incrementarea/ decrementarea SP este realizată cu ajutorul circuitului ID iar încărcarea lui RA cu adresa din SP se face pe calea CC3 - CC5. Figura 2.9 constituie o ilustrare a operaţiunilor aferente lucrului cu stiva.

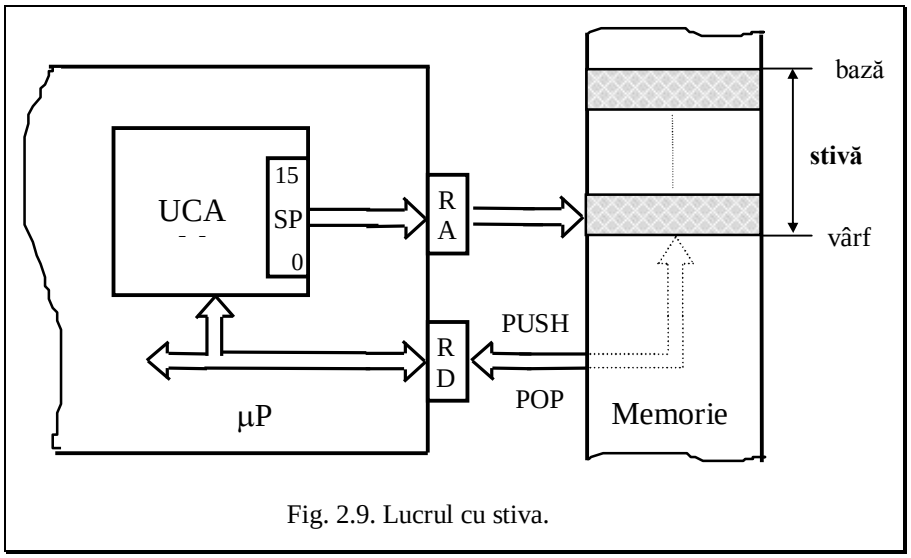


Fig. 2.9. Lucrul cu stiva.

În afara stocării temporare a datelor, stiva mai este utilizată în mecanismele de apelare a subprogramelor şi de răspuns la cererile de întrerupere. Registrul SP constituie un element de arhitectură, utilizatorul având posibilitatea să definească iniţial baza stivei. Există □P la care stiva este implementată *hardware* cu ajutorul unor registre speciale. Acest tip de stivă prezintă avantajul accesului rapid şi dezavantajul limitării severe a mărimii. În cazul stivei hardware SP devine transparent pentru utilizator întrucât nu mai este necesară o definire a bazei stivei.

Registrele index sunt opţionale în structura unui □P8 standard şi permit localizarea rapidă a informaţiei într-un bloc pe baza adresei fizice, care se poate obţine prin efectuarea unei operaţii de adunare:

$AF_{element} = AF_{bază} +$ deplasament,

unde AF reprezintă adresa fizică. Această operaţie este realizată în sumatorul S iar cei doi termeni sunt furnizaţi de un registru index (adresa de bază pe calea CC2 - CC4) şi de MID. Deplasamentul reprezintă adresa relativă a unui element în cadrul tabloului şi însoţeşte codul instrucţiunilor care folosesc date astfel structurate. Mărimea deplasamentului indică dimensiunea maximă a tabloului ce poate fi construit în memorie. Uzual pentru P8 deplasamentul are *8 biţi,* astfel încât se pot construi maxim $2^8 = 256$ elemente. Adresarea indexată permite localizarea rapidă printr-o singură instrucţiune a unui element de tablou şi creează premisele dezvoltării de noi metode pentru obţinerea adresei fizice prin calcul.

2.5. Nivelul 5 de caracterizare

Acest nivel este asociat unităţii de control a μP (UCμP8). După cum s-a arătat execuţia unui program comportă pentru fiecare instrucţiune parcurgerea următoarelor etape: *localizare, decodificare, execuţie propriu-zisă.* La rândul său fiecare etapă se descompune în acţiuni elementare care vor fi detaliate în continuare.

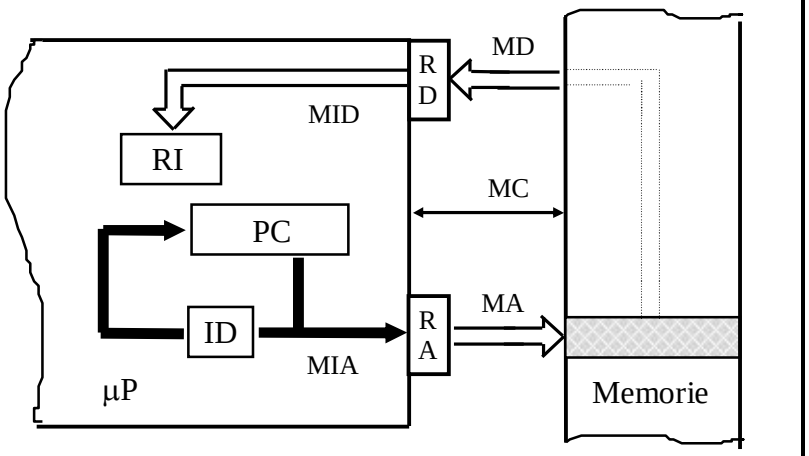


Fig. 2.10. Elemente implicate în extragerea unei instrucţiuni:
RI - registru de instrucţiuni; PC - numărător de program;
ID - circuit de incrementare/decrementare.

1. *Localizarea* şi aducerea în memorie a unei instrucţiuni, etapă la care participă elementele prezentate în figura 2.10, presupune parcurgerea următoarelor faze:

- încărcarea lui RA cu adresa din PC (în cazul uzual al parcurgerii secvenţiale a programului), în urma căreia adresa devine disponibilă pe MA;
- incrementarea lui PC pentru a se crea posibilitatea accesării următoarei locaţii de memorie;
- generarea pe magistrala de control a unui semnal de citire din memorie (READ);
- transferul pe MD şi de aici în RD a conţinutului locaţiei identificate;
- transferul codului instrucţiunii, din RD prin intermediul MID, într-un registru denumit *registru de instrucţiuni (RI).*

2. *Decodificarea* presupune recunoaşterea şi interpretarea conţinutului RI urmată de iniţierea acţiunilor aferente execuţiei.

3. *Execuţia* propriu-zisă implică activarea diverselor blocuri ale μP într-o ordine prestabilită şi/sau schimburi de informaţie cu memoria şi/sau porturile de intrare - ieşire.

Secvenţa de acţiuni elementare este dependentă de semantica fiecărei instrucţiuni. Coordonarea derulării în timp a fiecărei etape şi faze este asigurată de către UCμP8. O primă sarcină a acesteia o reprezintă stabilirea formatului instrucţiunii în funcţie de codul primit. Se are în vedere desfăşurarea în locaţii de memorie succesive a întregii informaţii necesare execuţiei instrucţiunii.

Având în vedere caracterul de automat sincron al unui μP, desfăşurarea în timp a etapelor şi fazelor unei instrucţiuni este nemijlocit legată de frecvenţa impulsurilor de sincronizare. Legat de execuţia în timp a instrucţiunilor se definesc următoarele noţiuni:

- *starea* ca timp maxim de efectuare a unei acţiuni elementare, reprezintă o durată egală cu perioada impulsurilor de sincronizare;

- *ciclul maşină* grupează mai multe acţiuni elementare în vederea conturării unei etape din execuţia unei instrucţiuni. Un ciclu maşină are mai multe stări şi are o semnificaţie strict funcţională fiind impus de necesităţi de sistematizare a activităţii a μP.

Atribuţiile legate de supervizarea funcţionării corecte a ansamblului de elemente reunite în structura unui μP impun prezenţa în cadrul UC□P8 a elementelor evidenţiate în figura 2.11. RI este conectat la MID, de unde preia codul instrucţiunii curente pe care îl transmite decodificatorului. Acesta identifică instrucţiunea din setul de instrucţiuni potenţial executabile de către μP. Informaţia rezultată la ieşirea decodificatorului este transmisă blocului de control şi sincronizare (BCS). Acesta este un automat finit microprogramat, care generează semnale de comandă pentru elementele implicate în execuţia instrucţiunii curente.

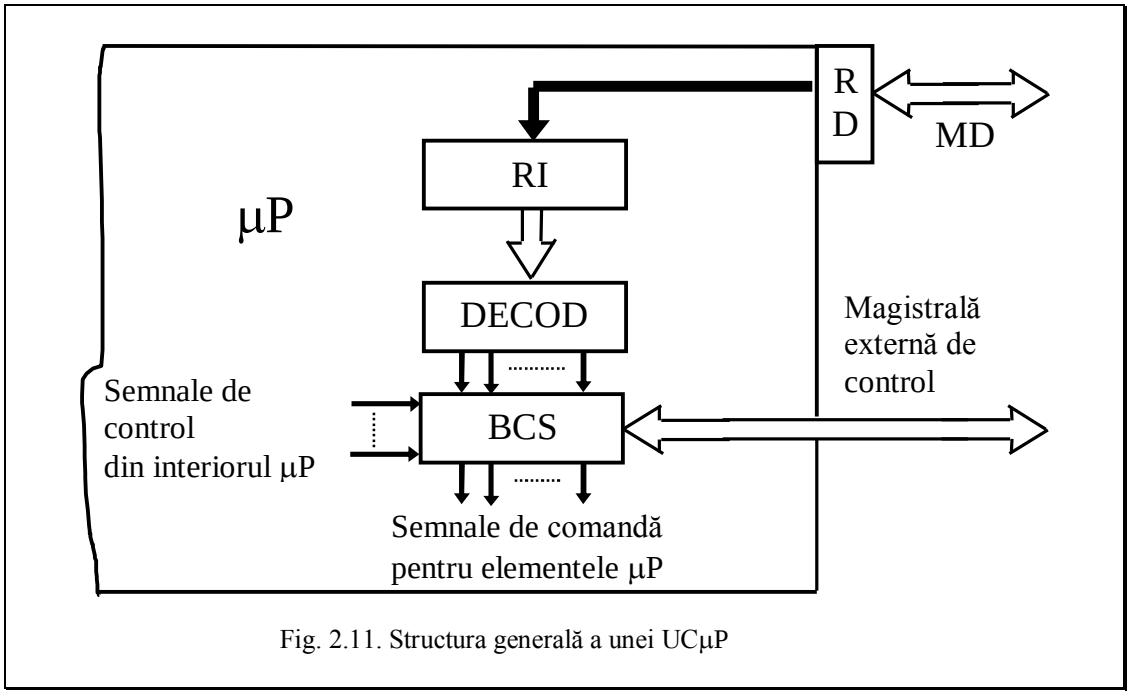


Fig. 2.11. Structura generală a unei UCμP

Microprogramul BCS va trebui să ţină cont, printre altele de:

- setul de instrucţiuni al μP;

- semantica fiecărei instrucţiuni;
- sistematizarea desfăşurării în timp a fiecărei instrucţiuni pe stări şi cicluri maşină;
- formatul fiecărei instrucţiuni;
- structura fizică concretă a blocurilor μP;
- semnalele de control necesare sau impuse μP (in interiorul sau din exteriorul său pe magistrala externă de comenzi).

BCS nu constituie un element de arhitectură şi prin urmare utilizatorul nu are acces la microprogramul acestuia. Datorită acestui fapt μP standard nu pot fi adaptate exact cerinţelor unei sarcini concrete prin optimizarea la nivel de microprogram a instrucţiunilor sale.

Din analiza efectuată pentru μP8 a reieşit că pe lângă neajunsul unei lungimi de numai 8 biţi a MD, acestea sunt caracterizate de absenţa oricărui paralelism în realizarea celor trei faze aferente execuţiei unei instrucţiuni.

3. Caracterizarea funcţională a unui microprocesor pe 16 biţi

La patru ani de la prezentarea primului μP8 - I8080 (MD pe 8 biţi, MA pe 16 biţi), firma Intel realizează μP8086 (magistrală unică multiplexată: 16 linii de date, 20 linii de adresă)

Bazat tot pe conceptul clasic al maşinii von Neumann noul μP a preluat majoritatea atributelor de structură şi arhitectură specifice generaţiei precedente (μP8). Dincolo de dublarea capacităţii numărului liniilor de date, în concepţia şi realizarea μP pe16 biţi (μP16) apar elemente de noutate care vor fi prezentate în continuare pe baza schemei funcţionale din figura 3.1.

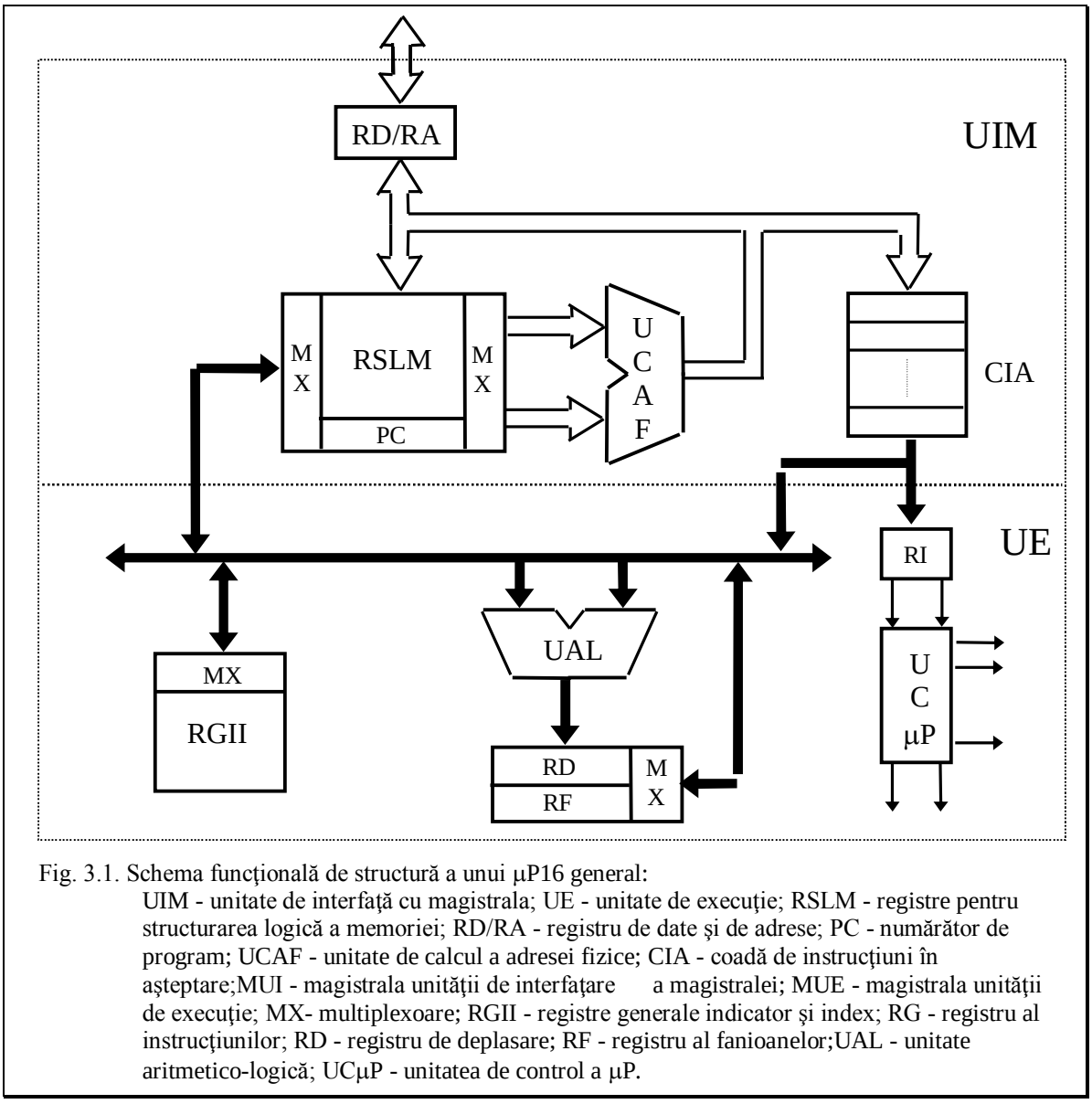


Fig. 3.1. Schema funcţională de structură a unui μP16 general:
UIM - unitate de interfaţă cu magistrala; UE - unitate de execuţie; RSLM - registre pentru structurarea logică a memoriei; RD/RA - registru de date şi de adrese; PC - numărător de program; UCAF - unitate de calcul a adresei fizice; CIA - coadă de instrucţiuni în aşteptare;MUI - magistrala unităţii de interfaţare a magistralei; MUE - magistrala unităţii de execuţie; MX- multiplexoare; RGII - registre generale indicator şi index; RG - registru al instrucţiunilor; RD - registru de deplasare; RF - registru al fanioanelor;UAL - unitate aritmetico-logică; UCμP - unitatea de control a μP.

Deosebirea esențială față de μP8 o constituie existența a două *procesoare specializate* care lucrează în paralel, pe care firma Intel, care a lansat primul μP16, le-a denumit **unitate de execuţie (UE) şi unitate de interfaţă cu magistrale (UIM).**

UE are ca principală sarcină execuţia instrucţiunilor, pe care împreună cu operanzii le primeşte prin intermediul UIM şi nu direct din memorie. Rezultatele prelucrării sunt trimise în memorie sau la porturi tot prin intermediul UIM.

UIM are drept scop mărirea cantităţii de informaţie vehiculată pe magistrală în unitatea de timp. Printre altele această funcţie presupune: furnizarea adreselor pentru instrucţiuni şi pentru date, calculul adreselor, realizarea structurării logice a memoriei, încărcarea cozii cu instrucţiuni, de unde vor fi preluate şi executate de către UE.

Setul de registre generale este completat cu registre de tip indicator şi index. Pentru fiecare din aceste registre există atât o utilizare implicită sugerată de fabricant, cât şi una alternativă.

În contextul versatilităţii registrelor generale, UAL nu mai are asociat un registru acumulator dedicat, oricare din RG putând îndeplini acest rol.

UCAM nu mai apare ca un bloc unitar, funcţiile fiind descentralizate astfel:

- registrele indicator şi index se găsesc în UE;
- numărătorul de program este asociat unui bloc de registre destinat structurării logice a memoriei;
- în UE apare un bloc special pentru calculul adreselor care dezvoltă sumatorul destinat acestui scop în structura unui μP8.

Paralelismul în funcţionare al UE şi UIM este asigurat de un bloc de registre asociat *cozii de instrucţiuni*. Aceasta se alimentează de către UIM şi se descarcă în UE.

Existenţa unei magistrale externe unice multiplexate conduce la existenţa unui unic registru tampon de date şi de adrese.

Sintetic, saltul calitativ care va conferi μP16 şi noi atribute de arhitectură are în vedere următoarele aspecte:

- existenţa a două procesoare care lucrează în paralel;
- versatilitatea funcţiilor registrelor;
- existenţa blocului pentru calculul adreselor;
- existenţa cozii de instrucţiuni;
- posibilitatea de structurare logică a memoriei.