

5. STRUCTURA ȘI ARHITECTURA CALCULATORULUI DIDACTIC

OBIECTIVE

În capitolul 5 se definesc structura și arhitectura calculatorului didactic, care va fi dezvoltat în continuare și în capitolele care urmează. Astfel sunt elaborate specificațiile primitivelor funcționale și modul de interconectare a acestora. De asemenea în acest capitol se urmărește funcționarea detaliată a unui calculator numeric prin parcurgerea tuturor etapelor de specificare, descriere, proiectare și evaluare a unității de execuție și a unității de comandă a calculatorului didactic.

CUPRINS

5. STRUCTURA ȘI ARHITECTURA CALCULATORULUI DIDACTIC	1
5.1 Primitivele funcționale ale calculatorului didactic	2
5.2 Schema bloc și fluxul informației	4
5.3 Setul de instrucțiuni	6
5.3.1 Instrucțiuni de transfer date	7
5.3.2 Instrucțiuni aritmetice	8
5.3.3 Instrucțiuni de transfer control	13
5.4 Formatul instrucțiunilor calculatorului didactic	14
5.5 Moduri de adresare	16
5.6 Codificarea instrucțiunilor	28
5.7 Faza de citire interpretare a instrucțiunilor mașină	32
5.8 Faza de execuție a instrucțiunilor mașină	37
5.9 Completarea setului de instrucțiuni	46
5.9.1 Introducerea instrucțiunilor RCL, RCR, ROL, ROR	47
5.9.1.1 Formatul și funcțiile instrucțiunilor RCL, RCR, ROL, ROR	47
5.9.1.2 Codificarea instrucțiunilor RCL, RCR, ROL, ROR	47
5.9.1.3 Modificarea fazei de citire interpretare în scopul recunoașterii acestor instrucțiuni	48
5.9.1.4 Descrierea fazei de execuție	48
5.9.2 Introducerea instrucțiunilor LOOP, LOOPZ	48
5.9.2.1 Formatul și funcțiile instrucțiunilor LOOP și LOOPZ.	48
5.9.2.2 Codificarea instrucțiunilor LOOP și LOOPZ	49
5.9.2.3 Modificarea fazei de citire interpretare în scopul recunoașterii acestor instrucțiuni	49
5.9.2.4 Descrierea fazei de execuție	49
5.9.3 Introducerea instrucțiunii XCHG	50
5.9.3.1 Formatul și funcția instrucțiunii XCHG.	50
5.9.3.2 Codificarea instrucțiunii XCHG	50

Pentru a facilita însușirea principalelor aspecte privind proiectarea unui calculator numeric se va considera un calculator didactic simplu dar care acoperă, din punct de vedere conceptual, principiile ce stau la baza proiectării și realizării calculatoarelor numerice moderne. Considerarea unui calculator didactic ca primă etapă în însușirea conceptelor privind funcționarea și realizarea unui calculator numeric oferă, față de analiza unui calculator real, mai multe avantaje cum ar fi:

- participarea directă la proiectarea calculatorului;
- o mare elasticitate în ceea ce privește opțiunile proiectului în diferite etape ale procesului de proiectare;
- o mai mare independență față de constrângerile și detaliile de implementare.

Calculatorul didactic este conceput ca un calculator universal adecvat prelucrărilor numerice și logice asupra unor structuri de date întâlnite, în general, în limbajele de nivel înalt.

5.1 Primitivele funcționale ale calculatorului didactic

În continuare se vor prezenta, pe scurt, primitivele funcționale ale calculatorului didactic și funcțiile acestora.

Memoria M

$$r_1M=16 ; r_2M=65536$$

Memoria M [65536;16] este o matrice de elemente de memorare organizată într-un spațiu de adresare unic de 65536 cuvinte a câte 16 biți fiecare.

Memoria este utilizată pentru a păstra informații neinterpretate reprezentând date sau instrucțiuni. Oricare două celule de memorie sunt accesibile în mod echivalent. Totuși, memoria poate fi logic împărțită în segmente de instrucțiuni, date, stivă, și segmente de memorie disponibilă. Pentru segmentul de instrucțiuni se poate prevedea și o memorie cu conținut permanent de tip "numai citire".

Citirea și scrierea se fac asincron sub controlul unității de comandă.

Registrul AM

$$r_{AM}=16$$

Registrul de adresare a memoriei, AM păstrează adresa celulei de memorie la care se face acces la un moment dat. Lungimea acestui registru se alege astfel ca $2^{r_{AM}} \geq r_2M$. Adresa calculată (adresa efectivă) este memorată în AM selectând, prin decodificare, cuvântul din memorie la care se va face accesul.

Registreele RG

$$r_1RG=16 ; r_2RG=8$$

Deoarece timpul de acces la memoria M este relativ mare (150ns-400ns) se va prevedea o memorie rapidă organizată sub forma a 8 registre de câte 16 fiecare. Registreele RG conțin unul sau ambii operanzi necesari pentru execuția instrucțiunilor CD. Unele din aceste registre vor fi utilizate și pentru calculul adresei efective a operanzilor din memorie. Astfel:

- BA, BB sunt utilizate ca registre de bază;
 - XA, XB sunt utilizate ca registre index iar
 - IS este utilizat ca indicator pentru adresarea unor structuri de date de tip stivă.
- RA, RB, RC sunt utilizate numai pentru păstrarea operanzilor.

Adresa registrului selectat este specificată în codul instrucțiunii. Selectarea registrelor generale va fi prezentată detaliat în paragraful 5.4.

Registrul CP

$r_{CP}=16$

Registrul contor program CP este utilizat pentru păstrarea adresei instrucțiunii ce urmează să se execute după terminarea execuției instrucțiunii curente. Lungimea lui CP se alege astfel ca $2^{CP} \geq r_2M$. Registrul CP poate fi inițializat cu o valoare dată la inițializarea sistemului, cu o valoare oarecare prin execuția instrucțiunilor de transfer control (vezi paragraful 5.3) sau poate fi incrementat în cazul execuției instrucțiunilor (operaționale) ce nu specifică transferul controlului la o altă secvență.

Unitatea aritmetică logică UAL

$r_{UAL}=16$

Unitatea aritmetică logică (UAL) realizează operațiile aritmetice și logice ale CD și este utilizată pentru prelucrarea datelor și pentru calculul adresei efective. UAL este de tip paralel, prelucrează operanzi pe 16 biți reprezentați în cod complementar. UAL implementează direct toate operațiile elementare necesare execuției instrucțiunilor aritmetice și logice ale CD. Condițiile în care s-a efectuat o operație în UAL și caracteristicile rezultatului sunt păstrate într-un registru de indicatori IND.

Registrele T1, T2

$r_{T1}=r_{T2}=16$

Registrele temporare T1 și T2 sunt utilizate pentru a păstra operanzii unei operații executate în UAL, rezultate intermediare la calcularea adresei efective și nu sunt accesibile în mod explicit de programator.

Indicatorii de condiții IND

$r_{IND} = 16$

Registrul de indicatori constituie o grupare a unor bistabili cu funcții individuale, poziționați la execuția instrucțiunilor în funcție de rezultatul din UAL. Registrul IND permite alegerea unei secvențe de execuție următoare unei operații aritmetice/logice în funcție de rezultatul acestei operații. Lungimea registrului IND este de 16 deși numai o parte din aceștia sunt utilizați în mod efectiv. Funcțiile acestor bistabili indicator vor fi prezentate în paragraful 5.3.2.

Registrul RI

$r_{RI} = 16$

Registrul de instrucțiuni RI păstrează codul instrucțiunii în curs de execuție. Conținutul său este decodificat și transmis secțiunii de generare comenzi/verificare stări din unitatea de comandă. În RI se păstrează și informațiile necesare pentru selecția registrelor generale în funcție de instrucțiunea în curs de execuție.

Magistrala MAG $rMAG = 16$

Interconectarea resurselor prezentate mai sus se realizează prin intermediul unei magistrale multiplexate în timp, MAG, care constituie suportul fizic de comunicație între aceste resurse. Dimensiunea magistralei este de 16 și este formată din 16 linii de interconectare, fiind astfel în totalitate pasivă. Fiecare resursă conectată la magistrală va include și circuitele de interfață necesare cuplării la magistrală. Transmisia pe MAG se face astfel încât un singur cuvânt de informație circulează pe magistrală la un moment dat.

Instrucțiunile calculatorului didactic

Specificarea transferurilor de informații între resursele CD și transformările suferite în timpul transferului se face prin intermediul instrucțiunilor. Instrucțiunile CD sunt grupate astfel:

- instrucțiuni de transfer date;
- instrucțiuni aritmetice/logice;
- instrucțiuni de transfer control.

Instrucțiunile de transfer date vor fi utilizate atunci când este necesară o transformare spațială a datelor fără a modifica conținutul informațiilor transferate. Acest grup include instrucțiuni de transfer între resursele interne ale unității de execuție sau între unitatea de execuție și subsistemul de intrări/ieșiri.

Instrucțiunile aritmetice și logice trebuie să permită efectuarea operațiilor aritmetice/logice uzuale.

Grupul de instrucțiuni de transfer control este necesar pentru implementarea structurilor de control care asigură înlănțuirea secvențelor de execuție a instrucțiunilor conform cu un algoritm dat, descris prin program.

5.2 Schema bloc și fluxul informației

Schema bloc a calculatorului didactic este prezentată în Fig. 5.1. În schemă se evidențiază principalele subansambluri ale calculatorului didactic:

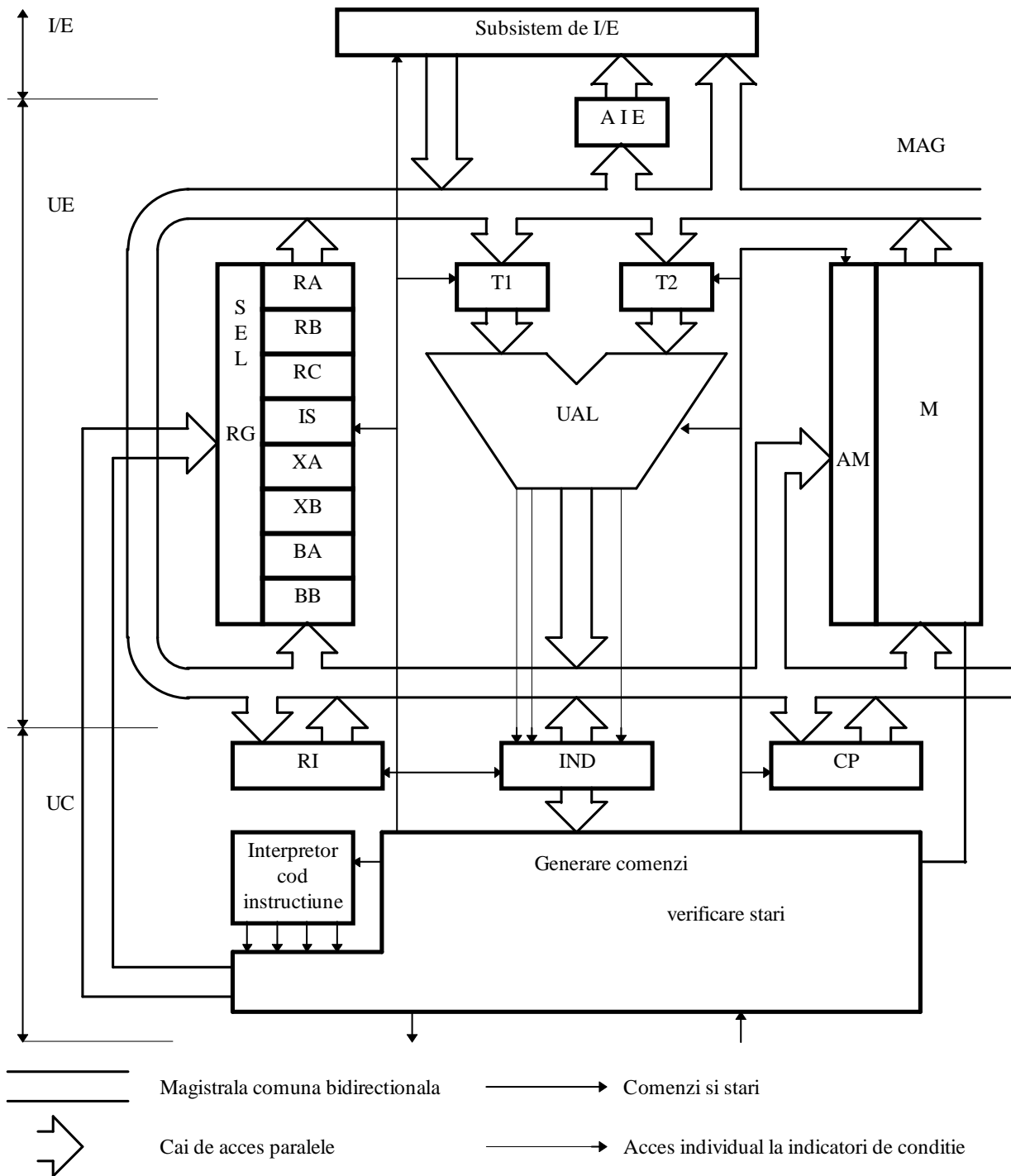
- unitatea de execuție (UE);
- unitatea de comandă (UCda);
- subsistemul de intrări /ieșiri (I/E).

Informațiile care se transferă între resursele CD reprezintă date sau comenzi. Informațiile reprezentând datele pot fi interpretate ca instrucțiuni sau operanzi în funcție de contextul în care acestea sunt transferate și prelucrate. Datele circulă în principal pe magistrala comună MAG iar comenzile circulă prin conexiuni dedicate între UC și resursele UE și subsistemului de intrare/ieșire.

Fluxul datelor trebuie astfel organizat încât să permită preluarea datelor primare din mediul extern prin intermediul subsistemului de intrări/ieșiri, prelucrarea acestora în unitatea de execuție UE și transmiterea rezultatelor spre mediul extern.

Datele preluate din exterior prin subsistemul de intrări/ieșiri sunt transferate prin magistrala MAG în registrele generale sau în memorie. Având în vedere viteza de lucru ridicată a registrelor generale RG în raport cu memoria M este de dorit ca prelucrările să se efectueze, pe cât posibil, între operanzi memorati în RG. În procesul de prelucrare operanzii sunt preluați din RG și transferați în registrele de lucru T1, T2 ca intrări pentru unitatea aritmetică logică UAL, iar rezultatul se depune în unul din registrele sursă din RG. Datorită numărului redus de registre generale RG, apare necesitatea transferului de operanzi între memorie și

acestea. Există și posibilitatea preluării ambilor operanzi din memorie sau a unui operand din memorie și a celuilalt din RG. Rezultatul este memorat în locul unuia din cei doi operanzi.



Schema bloc a calculatorului didactic

Figura 5.1

Registrul de indicatori este poziționat direct de către UAL. Există posibilitatea transferului direct, prin MAG, între IND și memorie pentru salvarea/restaurarea indicatorilor de condiții.

Registrul contor program CP poate fi încărcat cu o valoare din RG sau memorie, iar conținutul său poate fi transferat în registrul AM sau în memorie.

Posibilitățile de transfer a datelor între resursele calculatorului didactic sunt prezentate în tabelul 5.1.

Tabelul 5.1 Posibilități transfer date în CD

Sursă	Destinație										
	RG	M	T1	T2	RI	CP	I/E	AIE	UAL	AM	IND
RG	x	x	x	x		x	x			x	
M	x	x	x	x	x	x	x			x	x
T1									x		
T2									x		
RI								x			
CP	x	x				x				x	
I/E	x	x									
AIE							x				
UAL	x	x	x	x		x				x	x
AM											
IND		x									

Registrul de instrucțiuni RI este încărcat din memorie cu codul instrucțiunii curente iar conținutul său este accesibil unității de comandă pentru interpretare.

Fluxul informațiilor în calculatorul didactic, pentru execuția unui program, poate fi rezumat în felul următor.

Se citește din memorie instrucțiunea a cărei adresă se găsește în CP. Primul cuvânt al instrucțiunii se citește în RI și este utilizat de UC pentru a controla mai departe operațiile ce urmează să se execute. Se citesc, dacă este cazul, următoarele cuvinte ale instrucțiunii (instrucțiunile pot fi formate din 1,2 sau 3 cuvinte de 16) și se citesc operanzii. Aceste operații formează faza de citire - interpretare a instrucțiunii. Urmează apoi faza de execuție care execută prelucrarea efectivă a operanzilor conform cu codul operației specificat în RI. Rezultatul este depus în memorie sau în RG după cum este specificat în RI. Pentru execuția operațiilor de I/E datele sunt transferate între registrul RA și subsistemul de I/E.

5.3 Setul de instrucțiuni

Pentru rezolvarea pe calculator a unei probleme date, programatorul specifică prin instrucțiuni operațiile ce trebuiesc efectuate asupra datelor și ordinea în care acestea trebuie executate de către calculator.

Stabilirea setului de instrucțiuni al unui calculator este o operație dificilă și trebuie corelată cu o serie de aspecte ca: aplicațiile pentru care este destinat calculatorul, resursele calculatorului și structura acestora (registre generale, memorie), modul de acces la operanzii din registre sau memorie etc.

Pentru specificarea operațiilor logice și de calcul sunt necesare instrucțiuni aritmetice și logice care să permită realizarea unui set minim de operații cum ar fi: *adunarea și scăderea* (sau adunarea și schimbarea semnului); *conjuncția, disjuncția și negația* (sistem complet de funcții); *deplasarea aritmetică și logică*.

Pentru implementarea structurilor de control ale unui algoritm sunt necesare instrucțiuni de transfer control cum ar fi: *salt necondiționat* la o anumită secvență de instrucțiune; *testarea condițiilor de terminare* a operațiilor aritmetice/ logice și ramificarea execuției programului în funcție de aceste condiții.

Pentru transferul datelor primare și a rezultatelor între calculator și mediul extern sunt necesare instrucțiuni de intrare/ieșire.

Pentru structurarea datelor în memorie sau registrele generale sunt necesare instrucțiuni de transfer a datelor între aceste resurse.

Setul de instrucțiuni pentru calculatorul didactic poate fi format din următoarele tipuri de instrucțiuni: de transfer date; aritmetice; logice; transfer control.

5.3.1 Instrucțiuni de transfer date

Cu instrucțiunile de transfer date se poate specifica transferul datelor între memorie și registrele generale sau între registrul RA și porturi de I/E. Instrucțiunile de manipulare a stivei sunt incluse tot în acest grup. În tabelul 5.2 se prezintă instrucțiunile de transfer date:

Tabelul 5.2 Instrucțiuni de transfer date

Instrucțiuni CU SCOP GENERAL	MOV PUSH POP	Transferă cuvânt Transferă cuvânt în stivă Transferă cuvânt din stivă
Instrucțiuni INTRARI / IEȘIRI	IN OUT	intrare cuvânt ieșire cuvânt
Instrucțiuni TRANSFER INDICATORI	PUSHF POPF	Transferă indicatori în stivă Transferă indicatori din stivă

În continuare, se prezintă funcțiile efectuate la execuția acestora (Tabelul 5.3).

Tabelul 5.3 Instrucțiuni cu scop general

Format instrucțiune	Funcție
MOV <i>destinație, sursă</i>	MOV transferă un cuvânt specificat de operandul sursă la operandul destinație. Conținutul indicatorilor nu se schimbă.
PUSH <i>sursă</i>	PUSH decrementează indicatorul stivei IS apoi transferă un cuvânt specificat de operandul sursă în vârful stivei, indicat de IS. PUSH permite într-un mod convenabil memorarea temporară a datelor în stivă, de exemplu transferul parametrilor la chemarea unei proceduri. PUSH nu afectează indicatorii.
POP <i>destinație</i>	POP transferă cuvântul din vârful stivei indicat de IS la operandul destinație apoi incrementează pe IS pentru a indica noul vârf al stivei. POP permite transferarea datelor temporare din stivă în registre sau memorie. POP nu afectează indicatorii.
PUSHF	PUSHF decrementează indicatorul stivei IS, apoi transferă toți indicatorii în cuvântul din vârful stivei indicat de IS. Conținutul indicatorilor nu se schimbă.
POPF	POPF transferă biții corespunzători din cuvântul aflat în vârful stivei în indicatorii de condiții înlocuind vechile valori ale acestora. IS este incrementat pentru a indica noul vârf al stivei.
IN <i>port</i>	IN transferă un cuvânt de la portul de intrare specificat prin adresa "port" în registrul RA. Adresele porturilor de I/E sunt cuprinse între 0 și 255.
OUT <i>port</i>	OUT transferă un cuvânt din RA la portul de ieșire specificat prin adresa "port".

Observații:

PUSHF și POPF permit unei proceduri să salveze și să restaureze indicatorii corespunzători programului chemător și de asemenea permit modificarea prin program a conținutului indicatorilor.

IN și OUT nu modifică indicatorii.

5.3.2 Instrucțiuni aritmetice

Instrucțiunile aritmetice ale calculatorului didactic operează asupra unor numere reprezentate pe 16 biți sub forma de numere binare fără semn sau numere binare cu semn, numerele negative fiind reprezentate în cod complementar. Numerele sunt considerate întregi, poziția virgulei fiind implicit după cea mai puțin semnificativă cifră a numărului. Astfel domeniul de valori pentru numerele fără semn este între 0 și 65535, iar pentru numerele cu semn între -32768 și +32767.

În tabelul 5.4 se specifică instrucțiunile aritmetice care intră în setul de instrucțiuni al calculatorului didactic.

Tabelul 5.4 Instrucțiuni aritmetice

Instrucțiuni ADUNARE	ADD	Adună cuvânt
	ADC	Adună cuvânt și transport
	INC	Incrementează cuvânt
Instrucțiuni SCADERE	SUB	Scade cuvânt
	SBB	Scade cuvânt cu împrumut
	DEC	Decrementează cuvânt
	NEG	Schimbă semnul la cuvânt
	CMP	Compară cuvânt

Pentru a analiza rezultatul operațiilor aritmetice se utilizează indicatorii de condiții prezentați în tabelul 5.5.

Indicator	Funcție
T (transport):	<ul style="list-style-type: none"> – este setat (pozionat în "1") dacă în urma unei adunări rezultă un transport dinspre rangul cel mai semnificativ, altfel T este șters (trecut în "0"). Transportul T este setat dacă în urma unei scăderi rezultă un împrumut în cel mai semnificativ bit al rezultatului, altfel este șters. – poate fi interpretat ca depășire în operațiile cu numere întregi fără semn. – poate fi utilizat în instrucțiunile ADC și SBB pentru a efectua operații aritmetice în precizie multiplă (32 de biți sau 64 biți). – poate fi testat cu instrucțiuni de salt condiționat.
S (semn):	<ul style="list-style-type: none"> – la execuția instrucțiunilor aritmetice și logice indicatorul S este setat la valoarea bitului cel mai semnificativ al rezultatului (bitul de semn). Pentru numere cu semn în absența depășirii S=0 indică rezultat pozitiv iar S=1 indică rezultat negativ. – poate fi testat cu instrucțiuni de salt condiționat. În cazul operațiilor cu numere fără semn S poate fi ignorat deoarece în acest caz specifică cel mai semnificativ bit al rezultatului.
Z (zero):	<ul style="list-style-type: none"> – este setat dacă în urma unei operații aritmetice sau logice se obține rezultat egal cu 0, altfel Z este șters. – poate fi testat cu instrucțiuni de salt condiționat pentru a dirija secvența de execuție a instrucțiunilor în funcție de valoarea rezultatului.
P (paritate):	<ul style="list-style-type: none"> – indicatorul de paritate P este setat dacă în urma execuției unei operații aritmetice sau logice rezultatul conține un număr par de biți egali cu 1, altfel P este șters. – poate fi testat cu instrucțiuni de salt condiționat.
D (depășire):	<ul style="list-style-type: none"> – indicatorul de depășire D este setat dacă în urma execuției unei operații aritmetice rezultatul este un număr pozitiv prea mare sau un număr negativ prea mic pentru a putea fi reprezentat în operandul destinație (exclusiv bitul de semn); altfel D este șters. – poate fi interpretat ca depășire în operațiile cu numere întregi cu semn și poate fi testat cu instrucțiuni de salt condiționat. – poate fi ignorat în operațiile aritmetice cu numere întregi fără semn.

În continuare se prezintă formatul instrucțiunilor aritmetice și funcțiile efectuate la execuția acestora (tabelul 5.6).

Tabelul 5.6 Instrucțiuni aritmetice

Format instrucțiune	Descriere
ADD <i>destinație, sursă</i>	Suma celor doi operanzi trece în locul operandului destinație. Operanzii pot fi numere binare cu semn sau fără semn. ADD poziționează indicatorii T, S, Z, P, D.
ADC <i>destinație, sursă</i>	ADC adună cei doi operanzi și mai adună 1 dacă T=1 iar rezultatul îl trece în locul operandului destinație. Operanzii pot fi numere binare cu sau fără semn. ADC se utilizează pentru operații în precizie multiplă. ADC poziționează toți indicatorii de condiții.
INC <i>destinație</i>	INC adună 1 la operandul destinație considerat ca un număr pe 16 biți fără semn. INC poziționează bistabilii de condiții cu excepția lui T.
SUB <i>destinație, sursă</i>	Operandul sursă este scăzut din operandul destinație și rezultatul este trecut în locul operandului destinație. Operanzii pot fi numere binare cu sau fără semn. SUB poziționează toți indicatorii de condiții.
SBB <i>destinație, sursă</i>	SBB scade operandul sursă din operandul destinație și mai scade 1 dacă T=1. Rezultatul trece în locul operandului destinație. Operanzii pot fi numere binare cu sau fără semn. SBB se utilizează pentru operații în precizie multiplă. SBB poziționează toți indicatorii de condiții.
DEC <i>destinație</i>	DEC scade 1 din operandul destinație considerat ca un număr pe 16 biți fără semn. DEC poziționează indicatorii cu excepția lui T.
NEG <i>destinație</i>	Operandul destinație se scade din 0 și rezultatul trece în locul operandului destinație. Aceasta operație inversează semnul unui întreg, trecându-l sub formă de complement față de doi. Dacă se încearcă negarea lui -32768, operandul rămâne neschimbat și se setează indicatorul de depășire, D=1. NEG poziționează toți indicatorii de condiții. T este întotdeauna setat exceptând situația în care operandul este zero.
CMP <i>destinație, sursă</i>	CMP scade operandul sursă din operandul destinație dar rezultatul nu se reține. Operanzii rămân neschimbați iar indicatorii de condiții sunt poziționați corespunzător și pot fi testați cu instrucțiuni de salt condiționat.

5.3.3 Instrucțiuni logice și de deplasare

În acest grup au fost incluse instrucțiunile care operează asupra biților unui cuvânt prin funcții logice și de deplasare și sunt prezentate în Tabelul 5.7.

Instrucțiunile logice afectează indicatorii astfel: T și D sunt întotdeauna zero în urma unei operații logice iar Z, S și P sunt poziționați în funcție de rezultatul operației și pot fi testați cu instrucțiunile de salt condiționat, ca și în cazul instrucțiunilor aritmetice.

Instrucțiuni LOGICE	NOT AND OR XOR TEST	"Nu" cuvânt "Și" cuvânt "Sau" cuvânt "Sau exclusiv" cuvânt "Test" cuvânt
Instrucțiuni DEPLASARE	SHL/SAL SHR SAR	Deplasare logică/aritmetică la stânga cuvânt Deplasare logică la dreapta cuvânt Deplasare aritmetică la dreapta cuvânt

În continuare se prezintă formatul instrucțiunilor logice și de deplasare și funcțiile efectuate la execuția acestora.

Tabelul 5.8 Formatul instrucțiunilor logice

Formatul instrucțiunii	Funcție
NOT <i>destinație</i>	NOT inversează (complementul față de 1) biții cuvântului dat de operandul destinație.
AND <i>destinație,sursă</i>	AND efectuează "și" logic între biții celor doi operanzi și trece rezultatul în locul operandului destinație.
OR <i>destinație,sursă</i>	OR efectuează "SAU" logic între biții celor doi operanzi și trece rezultatul în locul operandului destinație.
XOR <i>destinație,sursă</i>	XOR efectuează "SAU-exclusiv" între biții celor doi operanzi și trece rezultatul în locul operandului destinație.
TEST <i>destinație,sursă</i>	TEST efectuează "și" logic între cei doi operanzi și nu se reține rezultatul. Cei doi operanzi rămân neschimbați iar indicatorii sunt poziționați corespunzător, putând fi testați cu instrucțiunile de salt condiționat.

Instrucțiunile de deplasare se utilizează pentru a deplasa aritmetic sau logic biții unui cuvânt. Deplasarea aritmetică se utilizează pentru înmulțirea/împărțirea numerelor cu puteri ale lui 2 iar deplasarea logică se utilizează pentru izolarea unor biți dintr-un cuvânt. Bistabilii de condiții sunt afectați de instrucțiunile de deplasare în felul următor: S, Z, P sunt poziționați în mod obișnuit, ca și în cazul instrucțiunilor logice. T conține întotdeauna bitul deplasat în afara operandului destinație iar D este setat dacă în urma operației de deplasare bitul cel mai semnificativ (semn) și-a schimbat valoarea, altfel D este șters.

Tabelul 5.8. Formatul instrucțiunilor de deplasare

Formatul instrucțiunii	Funcție
SHL/SAL <i>destinație</i>	SHL/SAL realizează deplasarea la stânga cu o poziție a operandului destinație. În bitul cel mai puțin semnificativ se introduce zero. Deplasarea logică și aritmetică la stânga cu o poziție produc același rezultat. S-au prevăzut două mnemonice pentru a mări flexibilitatea de exprimare a utilizatorului, în funcție de contextul de prelucrare a datelor.
SHR <i>destinație</i>	SHR deplasează logic la dreapta biții operandului destinație introducând zero în bitul cel mai semnificativ.
SAR <i>destinație</i>	SAR deplasează aritmetic la dreapta biții operandului destinație. Deplasarea se face cu extensia bitului de semn (bitul de semn rămîne neschimbat iar bitul cel mai semnificativ de date preia conținutul bitului de semn).

5.3.3 Instrucțiuni de transfer control

Ordinea în care se execută instrucțiunile unui program pentru calculatorul didactic este determinată de secvența de valori reprezentând conținutul registrului CP.

Tabelul 5.9 Instrucțiuni de transfer control

Instrucțiuni TRANSFER NECONDIȚIONAT	CALL	Salt necondiționat la o procedură	
	RET	Revenire necondiționată dintr-o procedură	
	IRET	Revenire necondiționată dintr-o procedură de întrerupere	
	JMP	Salt necondiționat la o anumită adresă	
Instrucțiuni TRANSFER CONDIȚIONAT		condiția testată	salt dacă
	JA	$(T \text{ OR } Z)=0$	"above"; $d>s$ (d,s fără semn)
	JAE	$T=0$	"above or equal"; $d\leq s$ (d,s fără semn)
	JB	$T=1$	"below"; $d<s$ (d,s fără semn)
	JBE	$(T \text{ OR } Z)=1$	"below or equal"; $d\geq s$ (d,s fără semn)
	JC	$T=1$	"carry"; transport
	JE	$Z=1$	"equal"; $d=s$
	JG	$((S \text{ XOR } D)\text{OR } Z)=0$	"greater"; $d>s$ (d,s cu semn)
	JGE	$(S \text{ XOR } D)=0$	"greater or equal"; $d\geq s$ (d,s cu semn)
	JL	$(S \text{ XOR } D)=1$	"less"; $d<s$ (d,s cu semn)
	JLE	$((S \text{ XOR } D)\text{OR } Z)=1$	"less or equal"; $d\leq s$ (d,s cu semn)
	JNC	$T=0$	"not carry"; nu există transport
	JNE	$Z=0$	"not equal"; d diferit s
	JNO	$D=0$	"not overflow"; nu există depășire
	JPO	$P=0$	"parity odd"; paritate impară
	JNS	$S=0$	"not sign"; rezultat pozitiv
	JO	$D=1$	"overflow"; depășire
JPE	$P=1$	"parity even"; paritate pară	
JS	$S=1$	"sign"; rezultat negativ	
Nota s=sursă ; d=destinație			

Secvența normală de execuție a unui program se obține prin incrementarea contorului program CP, la fiecare instrucțiune pentru a obține instrucțiunea următoare.

Instrucțiunile de transfer control operează asupra contorului program. Incărcarea registrului CP cu o valoare oarecare determină alterarea secvenței normale de execuție a programului. Transferul controlului la o altă secvență se poate face necondiționat sau condiționat de valorile indicatorilor de condiții. Instrucțiunile de transfer control sunt prezentate în Tabelul 5.5.

Instrucțiunea HLT a fost inclusă în acest grup deși efectul execuției acestei instrucțiuni este suspendarea execuției programului și trecerea UC a calculatorului didactic într-o stare de așteptare (se așteaptă un reset sau o întrerupere).

Tabelul 5.9 Formatul instrucțiunilor de transfer control

Formatul instrucțiunii	Funcție
CALL nume-procedura	CALL transferă controlul la procedura definită în cadrul programului și salvează în stivă informațiile necesare revenirii în programul principal (chemător). Revenirea se face la instrucțiunea imediat următoare instrucțiunii CALL, prin executarea, în procedura chemată, a unei instrucțiuni RET. Indiferent de modul în care se calculează adresa procedurii, UC decrementează registrul IS și salvează apoi în stivă adresa instrucțiunii următoare lui CALL. Valoarea calculată, reprezentând adresa procedurii înlocuiește apoi conținutul lui CP și secvența continuă cu execuția procedurii
RET	RET transferă controlul dintr-o procedură înapoi în programul chemător, la instrucțiunea imediat următoare instrucțiunii care a activat, prin CALL, procedura. RET extrage conținutul celulei indicată de IS (vârful stivei) și-l introduce în CP apoi incrementează pe IS.
IRET	IRET transferă controlul dintr-o procedură de tratare a unei întreruperi înapoi în programul întrerupt prin următoarea secvență de operații, conținutul celulei din vârful stivei, indicată de IS, este transferat în CP, se incrementează IS și conținutul celulei indicată de IS (actualizat) este transferat în registrul de indicatori după care IS se incrementează din nou și programul continuă cu instrucțiunea de la adresa aflată în CP. IRET poziționează indicatorii de condiții conform conținutului din stivă.
JMP destinație	JMP transferă controlul la locația indicată de destinație. JMP nu salvează nimic în stivă, execuția programului continuă cu instrucțiunea indicată de adresa calculată (adresa destinație).
Jcond "destinație"	Instrucțiunile de salt condiționat transferă controlul la altă secvență numai dacă se îndeplinește condiția de test specificată în instrucțiune. Dacă nu este îndeplinită condiția de salt se continuă execuția programului cu instrucțiunea următoare. Adresa "destinație" trebuie să fie cuprinsă în limitele - 128 și +127. Saltul se face relativ la contorul program : CP + " destinație ".
HLT	HLT trece UCP într-o stare de așteptare a unei întreruperi sau reinițializare a secvenței de comandă. Altfel spus din starea de HALT se poate ieși numai în urma unei întreruperi (continuându-se cu instrucțiunea următoare lui HLT) sau cu un reset (care introduce 0 în CP).

5.4 Formatul instrucțiunilor calculatorului didactic

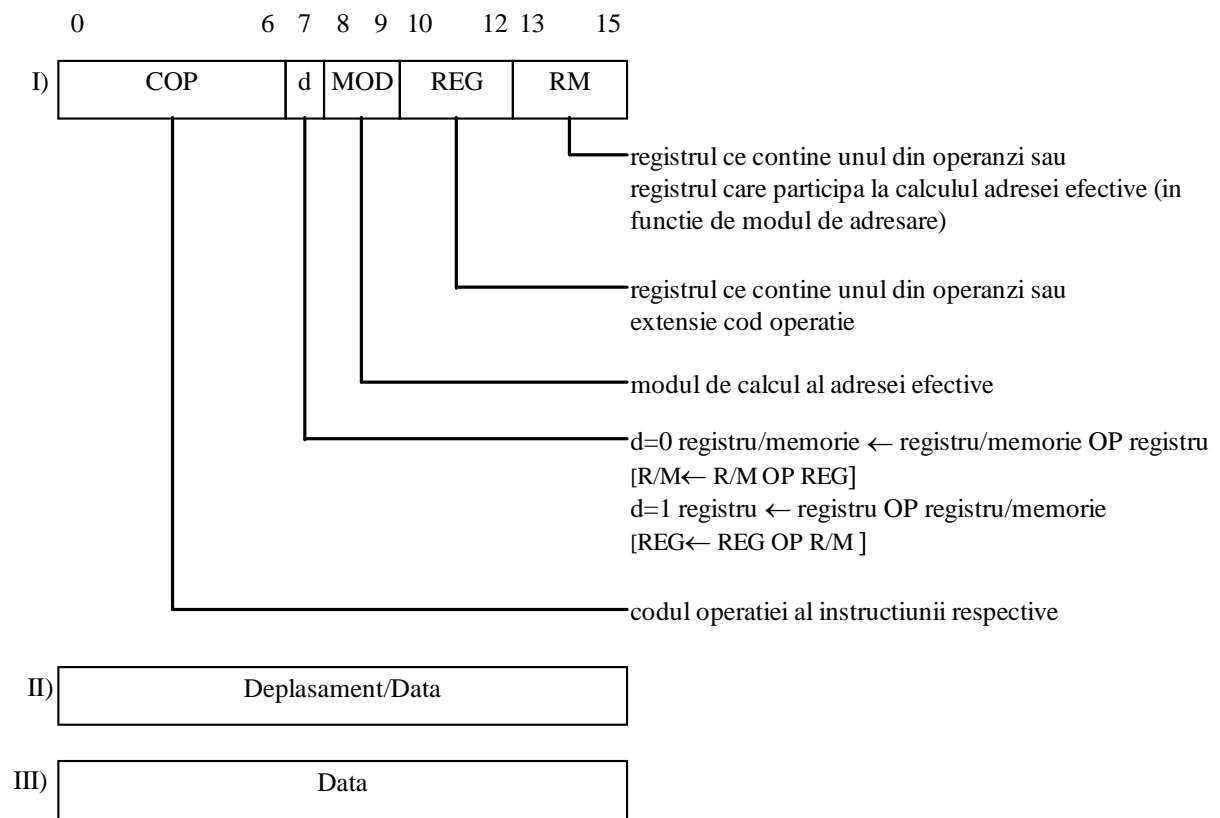
Repertoriul de instrucțiuni al calculatorului didactic a fost alcătuit în vederea ilustrării principalelor categorii de instrucțiuni întâlnite în calculatoarele moderne și pentru a satisface cerințele privind transferul

informației, operațiile aritmetice și logice, transferul comenzii în cadrul programului, comanda procesorului, etc. Aceste instrucțiuni, prin formatul pe care îl au, prin modalitățile extrem de variate pentru calculul adreselor efective ale operanzilor asigură facilități pentru manipularea unor structuri complexe de date, cum sunt tablourile unidimensionale și bidimensionale de date, diverse tipuri de înregistrări, fișiere, etc.

Instrucțiunile calculatorului didactic pot manipula operanzi aflați în registrele unității centrale, operanzi conținuți în instrucțiuni sau operanzi aflați în memorie, adresele efective ale acestor operanzi fiind calculate în diferite moduri, asigurând o mare suplețe în adresare.

Instrucțiunile au lungimi variabile de 1, 2 sau 3 cuvinte de câte 16 biți, lungimea fiind dictată de tipul operației codificate, de numărul de operanzi, de modul de adresare, etc.

În Fig. 5.2 este prezentată structura instrucțiunii calculatorului didactic.



Primul cuvânt din instrucțiune conține 5 câmpuri distincte :
COP, d, MOD, REG, RM

Tabelul 5.10 Câmpurile primului cuvânt al instrucțiunii CD

Câmp	Descriere
COP	reprezintă codul operației, are 7 biți și reprezintă funcția ce trebuie efectuată
d	specifică destinația rezultatului ținând seama de faptul că cei doi operanzi ai instrucțiunii sunt indicați cu ajutorul câmpurilor REG și RM. $d = 0$, RM \leftarrow RM COP REG $d = 1$, REG \leftarrow REG COP RM
MOD	arată modul de calcul al adresei efective a operandului și necesită 2 biți
REG	specifică registrul care conține operandul, sau se folosește ca extensie a câmpului COP, ocupă 3 biți
RM	indică registrul care conține un operand sau registrul folosit pentru calculul adresei efective.

Al doilea cuvânt al instrucțiunii conține deplasamentul (depls) folosit în calculul adresei efective sau data ca operand imediat, în cazul instrucțiunilor fără deplasament sau fără adresare la memorie.

Al treilea cuvânt al instrucțiunii, în cazul în care există, specifică data ca operand imediat.

5.5 Moduri de adresare

Modul de adresare reprezintă modalitatea în care se calculează adresa efectivă (AE) a operandilor implicați în instrucțiunea curentă.

Instrucțiunile calculatorului didactic pot prelucra maxim doi operanzi. Aceștia se pot găsi :

- ambii în registrele generale RG ;
- unul în registrele generale RG și altul în memorie ;
- unul în registrele generale RG și altul în cadrul instrucțiunii respective (operand imediat) ;
- unul în memorie și altul imediat.

În funcție de codificarea câmpurilor MOD și RM se precizează modul în care se calculează adresa efectivă (AE) a operandului. Tabelul 5.6 specifică modul în care se calculează adresa efectivă și codurile din câmpurile MOD, RM. Se observă că pentru MOD = 11, operandii se găsesc în registrele generale.

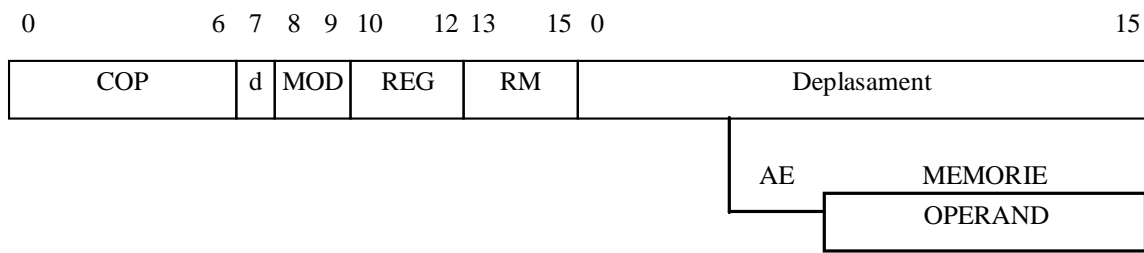
Pentru a asigura o mare flexibilitate în prelucrarea datelor alegem, pentru calculatorul didactic, următoarele moduri de adresare:

1. Adresare directă

Adresa efectivă este prevăzută în instrucțiune, în cuvântul al doilea, Fig. 5.3.

AE = Depls

Exemplu: mov RA,adresă

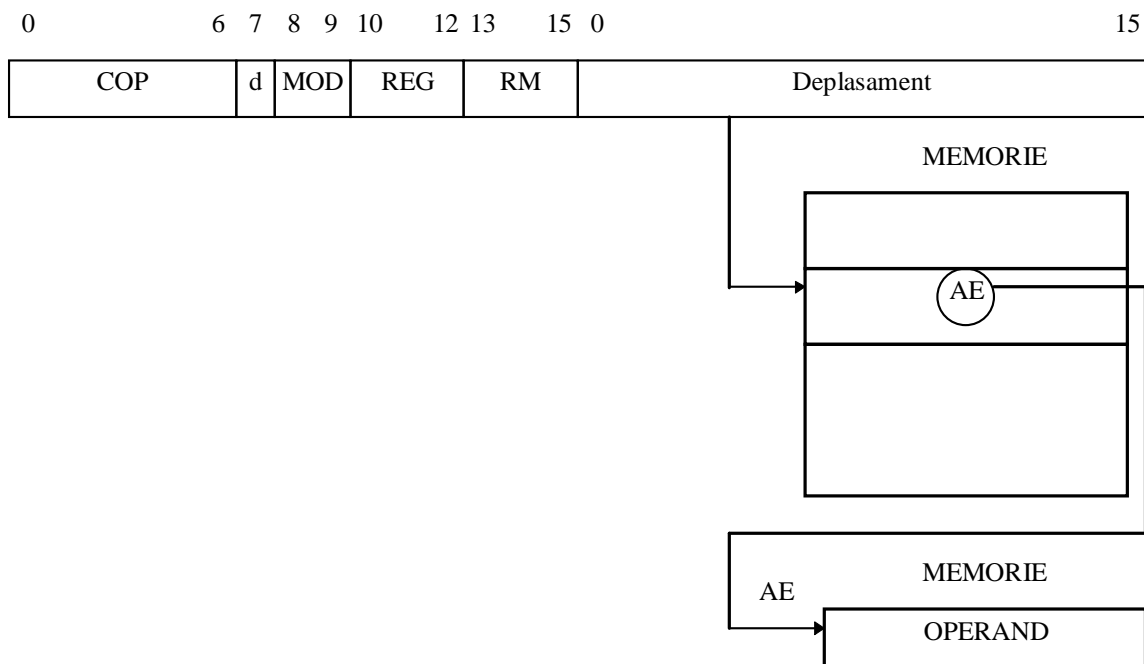


2. Adresare indirectă

Adresa efectivă se citește din memorie, din locația a cărei adresă este conținută în al doilea cuvânt al instrucțiunii, Fig.5.4.

$AE = [Depls]$

Exemplu: mov RA,[adresa]

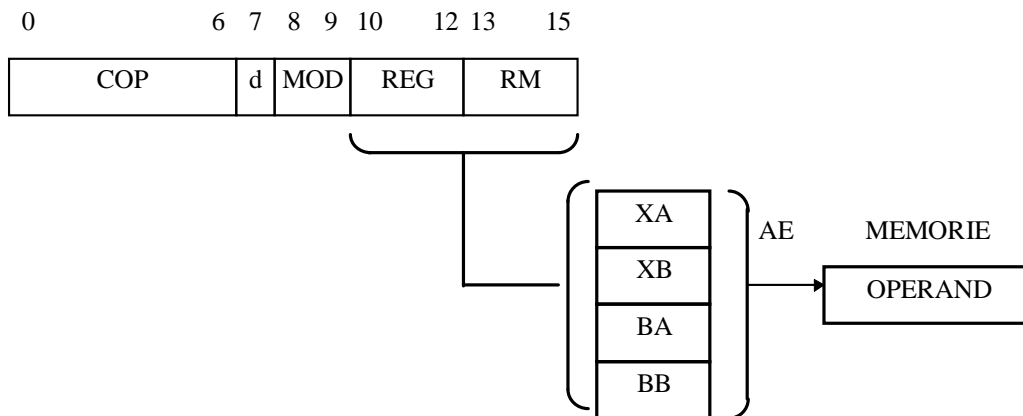


3. Adresare indirectă prin registru

Adresa efectivă se găsește în unul din registrele XA, XB,BA, BB, Fig. 5.5.

$$AE = \begin{bmatrix} XA \\ XB \\ BA \\ BB \end{bmatrix}$$

Exemplu: `mov RA,[BA]`

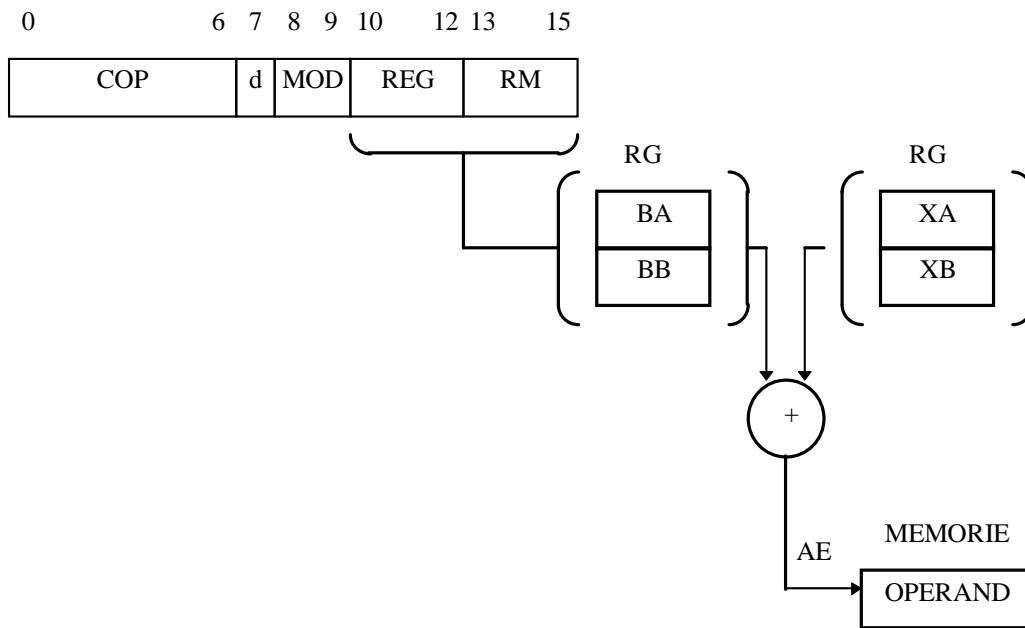


4. Adresare indirectă prin sumă de registre

Adresa efectivă se obține ca sumă a conținutului unui registru de bază cu conținutul unui registru index, Fig. 5.6.

$$AE = \begin{bmatrix} BA+XA \\ BA+XB \\ BB+XA \\ BB+XB \end{bmatrix}$$

Exemplu: `mov RA,[BA] [XA]`
`mov RA,[BA + XA]`



5. Adresare indirectă prin sumă de registre cu autoincrementare a registrelor index după calculul adresei efective.

Față de modul precedent de adresare apare deosebirea că registrele index se incrementează după generarea adresei efective, Fig. 5.7.

$$AE = \begin{bmatrix} BA+XA+ \\ BA+XB+ \\ BB+XA+ \\ BB+XB+ \end{bmatrix}$$

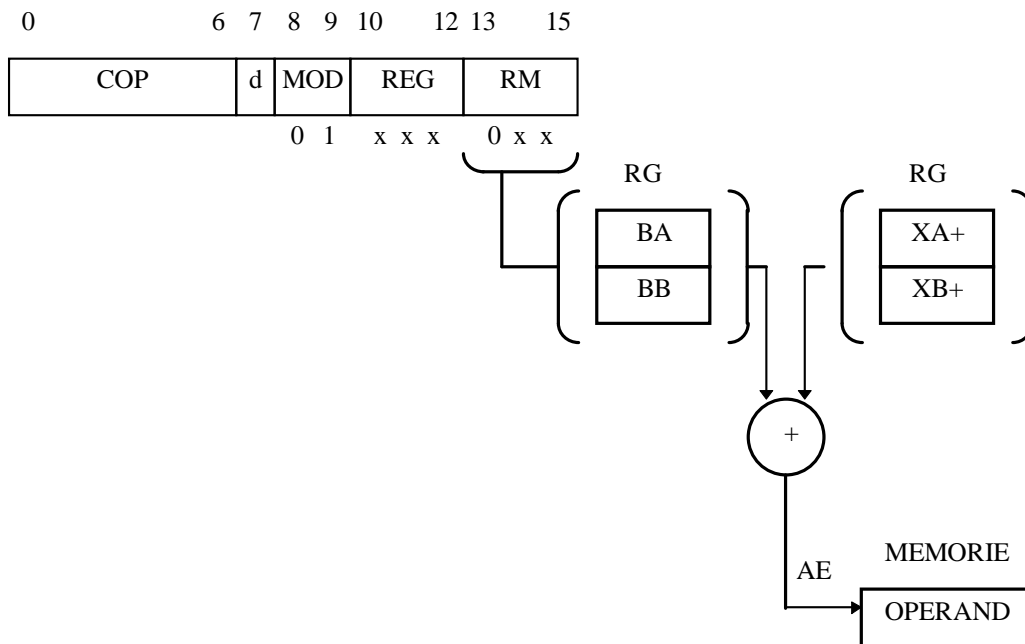
ceea ce este echivalent cu :

$$AE = \begin{bmatrix} BA+XA \\ BA+XB \\ BB+XA \\ BB+XB \end{bmatrix}$$

; XA ← XA + 1
; XB ← XB + 1

Incrementarea registrului index XA sau XB are loc după participarea la calculul adresei efective.

Exemplu: mov RA,[BA][XA+]



6. Adresare indirectă prin sumă de registre cu autodecrementare a registrului index înainte de calculul adresei efective.

În acest caz se folosește doar un singur registru index și anume registrul XA, Fig. 5.8.

$$AE = \begin{bmatrix} BA+XA- \\ BB+XA- \end{bmatrix}$$

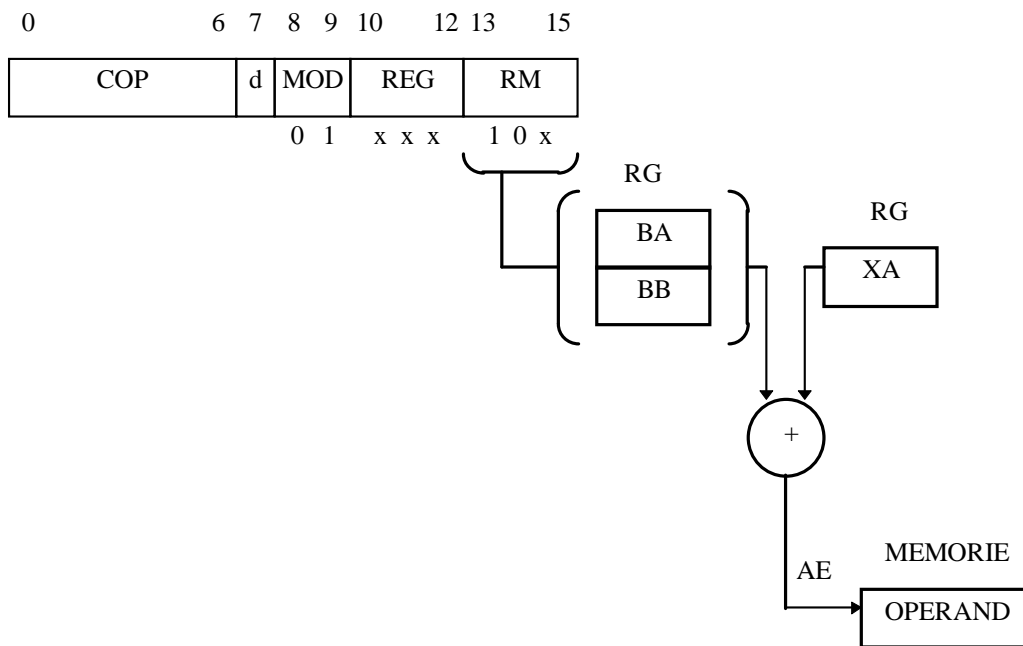
ceea ce este echivalent cu:

$$XA \leftarrow XA - 1 \text{ și}$$

$$AE = \begin{bmatrix} BA+XA \\ BB+XA \end{bmatrix}$$

Decrementarea registrului index XA are loc înainte de a participa la calculul adresei efective.

Exemplu: `mov RA,[BA][XA-]`

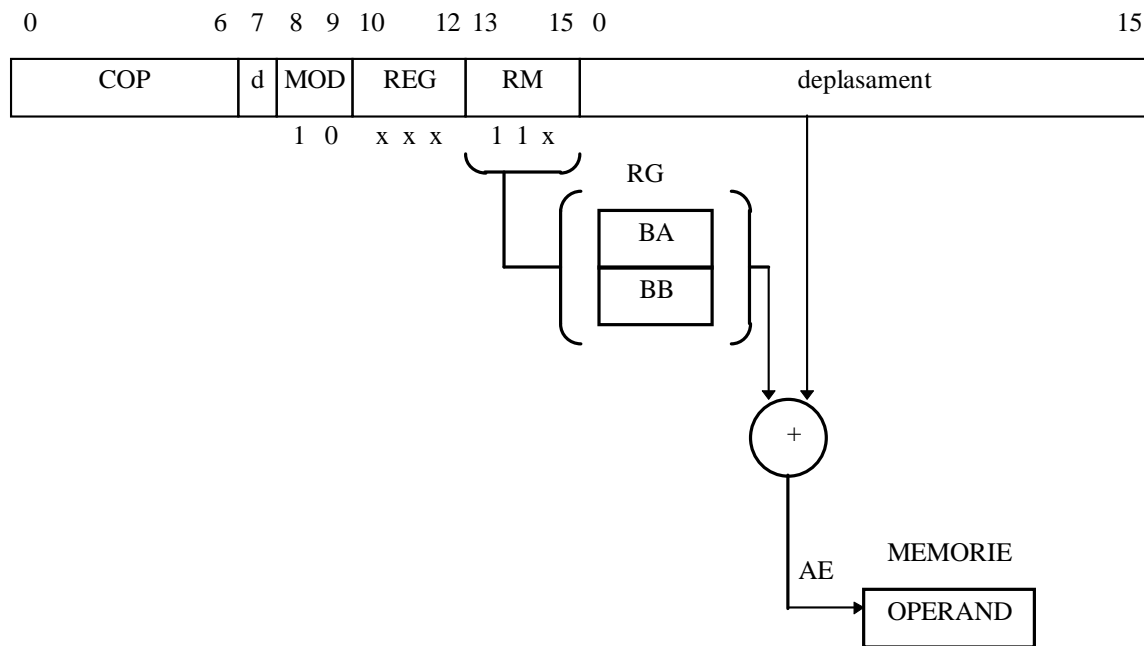


7. Adresare bazată

Adresa efectivă se obține prin adunarea conținutului celui de-al doilea cuvânt al instrucțiunii cu unul din registrele bază, Fig. 5.9.

$$AE = \begin{bmatrix} BA \\ BB \end{bmatrix} + \text{deplasament}$$

Exemplu: mov RA,[BA]+adresa
 mov RA,adresa[BA]
 mov RA,[BA+adresa]

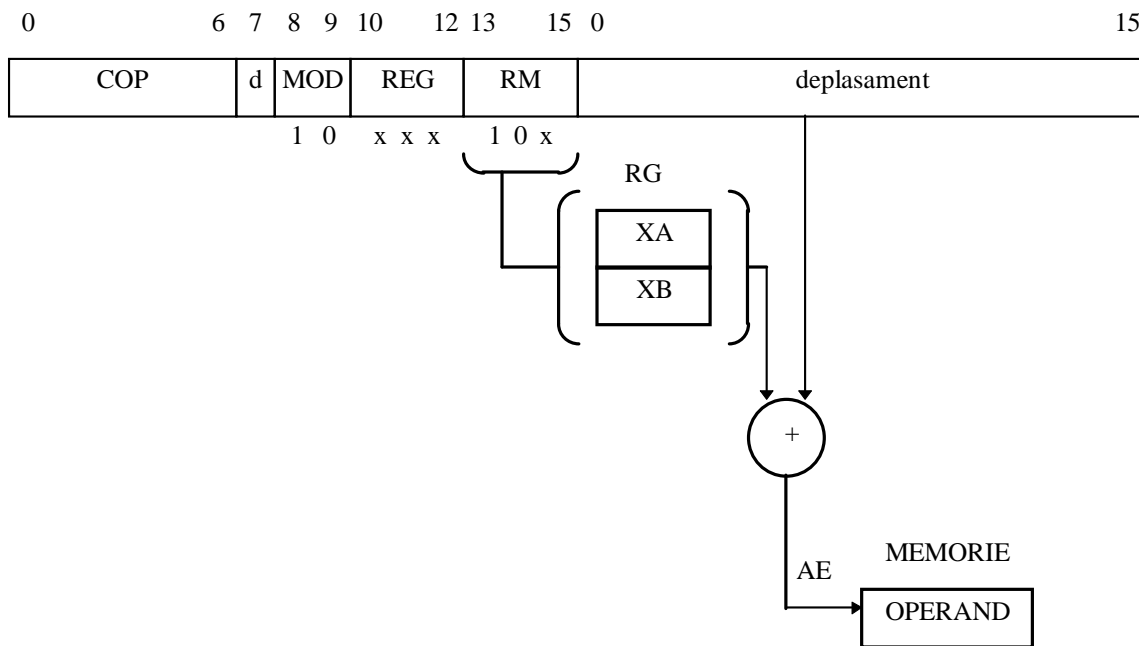


8. Adresare indexată

Adresa efectivă se obține prin adunarea conținutului celui de-al doilea cuvânt al instrucțiunii cu unul din registrele index, Fig. 5.10.

$$AE = \begin{bmatrix} XA \\ XB \end{bmatrix} + \text{deplasament}$$

Exemplu: mov RA,[XA]+adresa
 mov RA,adresa[XA]
 mov RA,[XA+adresa]

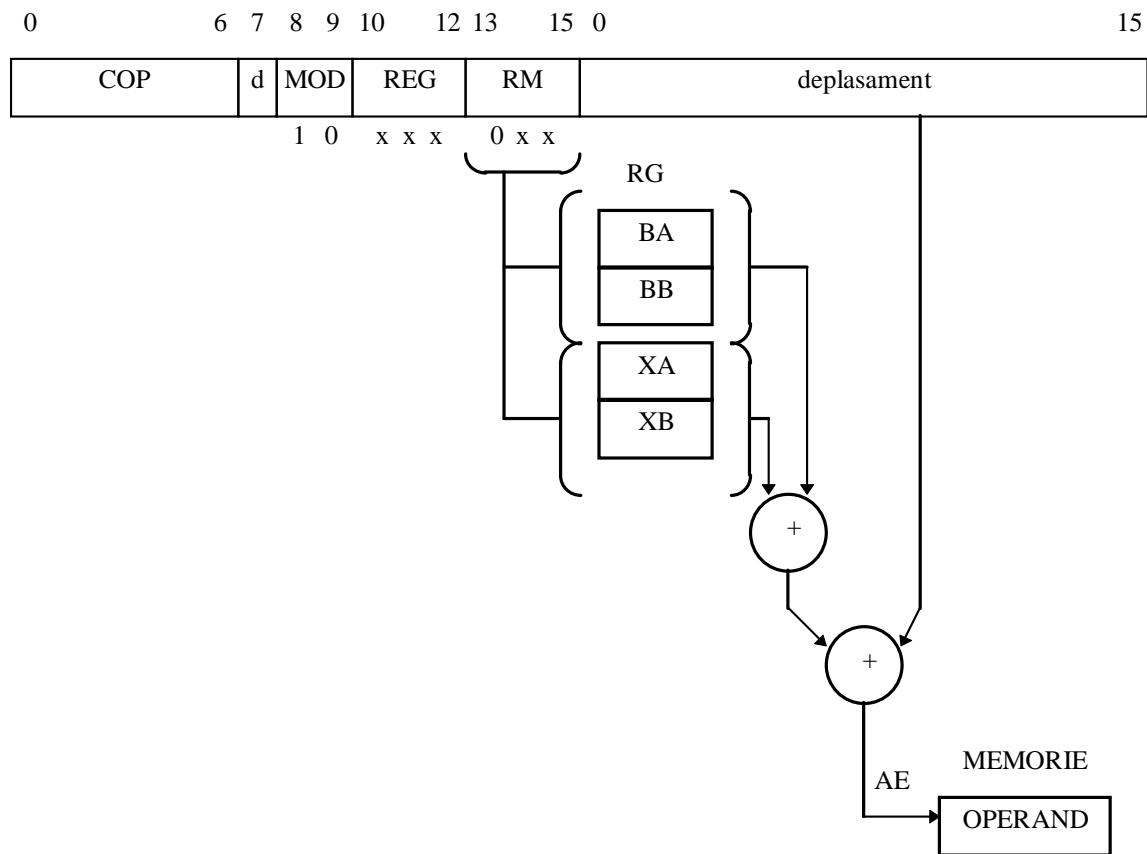


9. Adresare bazată indexată

Adresa efectivă se obține prin adresarea conținutului celui de-al doilea cuvânt al instrucțiunii cu suma dintre un registru de bază și un registru index, Fig. 5.11.

$$AE = \begin{bmatrix} BA+XA \\ BA+XB \\ BB+XA \\ BB+XB \end{bmatrix} + \text{deplasament}$$

Exemplu: mov RA,[BA][XA]+adresa
 mov RA,adresa[BA][XA]
 mov RA,[BA+XA+adresa]
 mov RA,[BA][XA].adresa

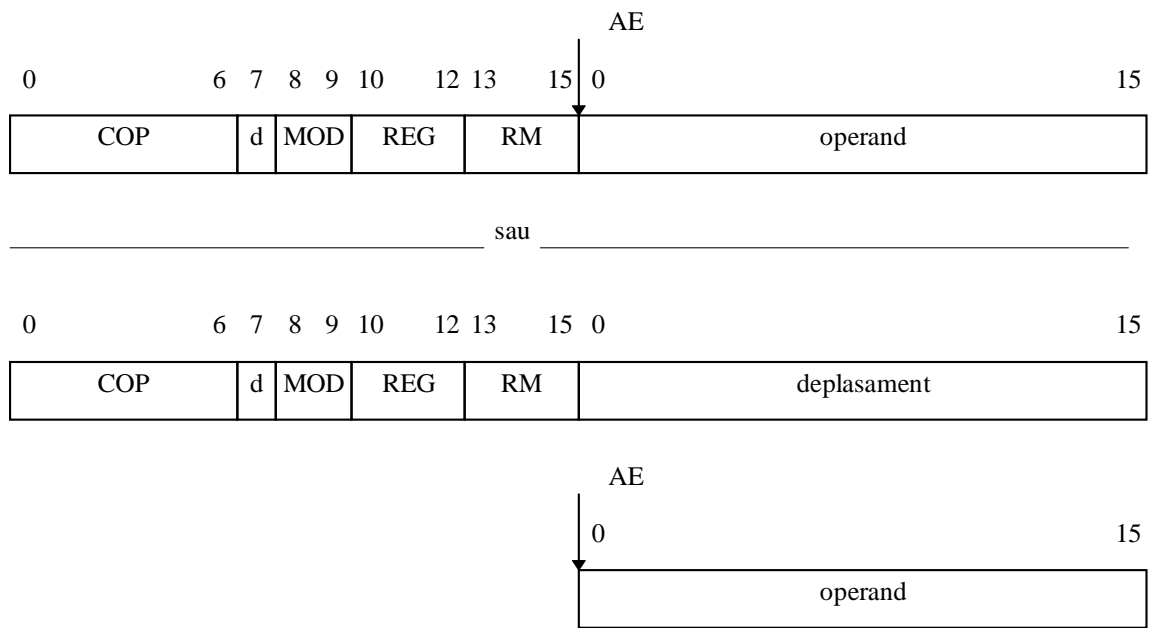


10. Adresare imediată

Operandul se găsește în al doilea cuvânt al instrucțiunii sau al treilea cuvânt al instrucțiunii, Fig. 5.12.

$$AE = \begin{bmatrix} CP + 1 \\ CP + 2 \end{bmatrix}$$

Exemplu: `mov RA,7`

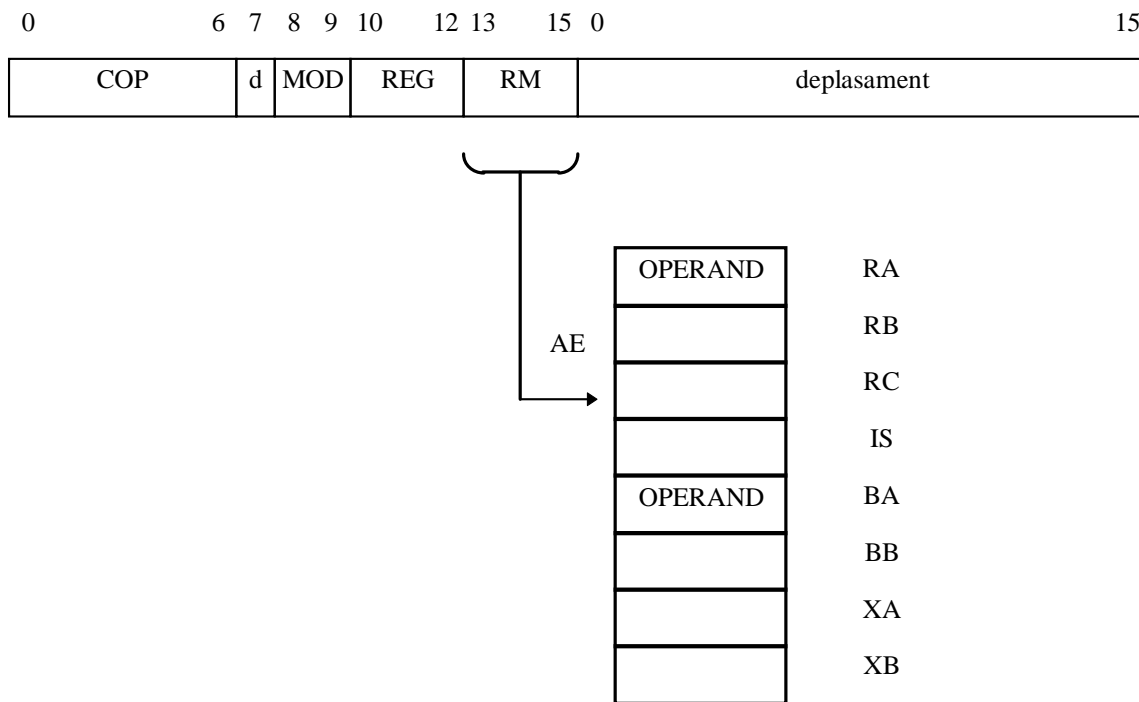


11. Adresare directă la registru

Operandul se găsește într-unul din registrele specificate în câmpurile REG sau RM, Fig 5.13.

$$AE = \begin{bmatrix} REG \\ RM \end{bmatrix}$$

Exemplu: mov RA,RB



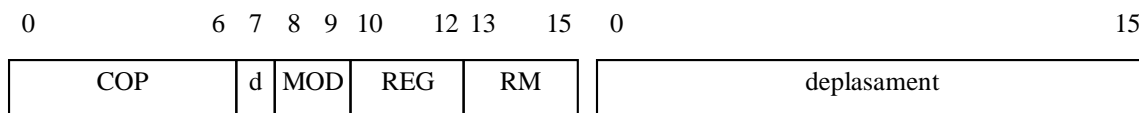
Clasificarea instrucțiunilor

Din punctul de vedere al adresării, putem clasifica instrucțiunile în următoarele mari categorii:

a) - **instrucțiuni cu calcul de adresă efectivă** în care distingem:

a1) - **instrucțiuni cu o singură adresă efectivă** (cu un singur operand), Fig 5.14 :

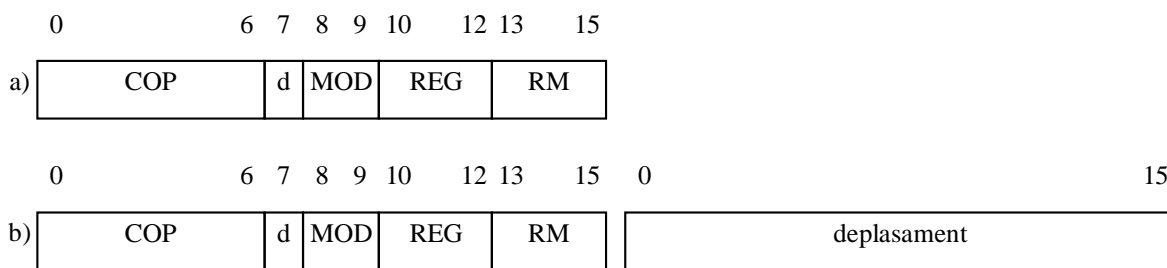
**INC; DEC; NEG; NOT; SHL/SAL; SHR; SAR;
PUSH; POP; CALL; JMP**



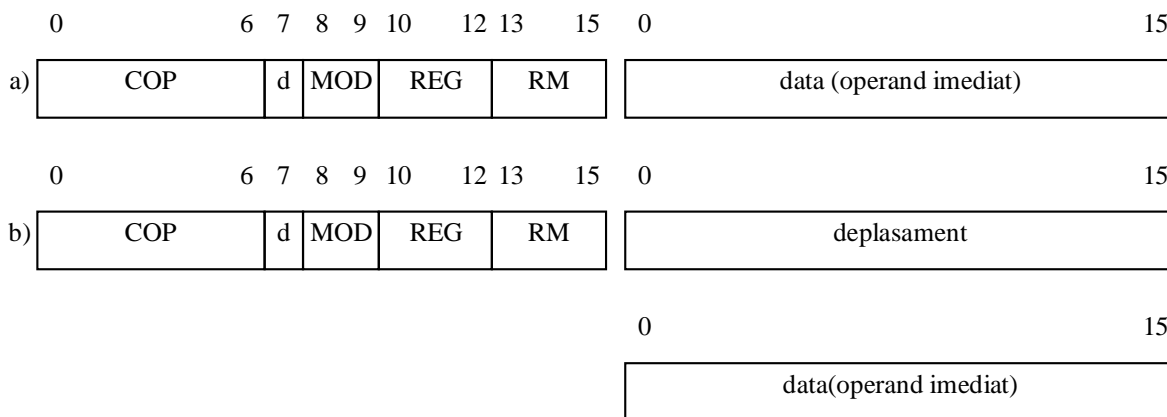
a2) - **instrucțiuni cu 2 adrese efective** (cu 2 operanzi)

a2.1) - cu **2 operanzi** ale căror adrese sunt specificate în instrucțiune, Fig. 5.15 :

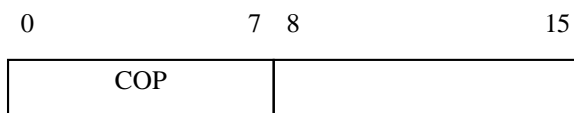
MOV; ADD; ADC; SUB; SBB; CMP; AND; OR; XOR; TEST



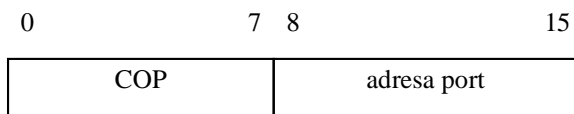
a2.2) - cu 2 operanzi dintre care unul este imediat, Fig. 5.16 :
MOV; ADD; ADC; SUB; SBB; CMP; AND; OR; XOR; TEST



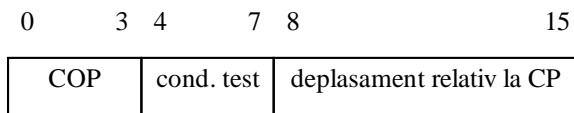
b) - instrucțiuni fără calcul de adresă efectivă, Fig. 5.17:
PUSHF; POPF; RET; IRET; HLT; IN; OUT; Jcond



a) PUSHF; POPF; RET; IRET; HLT



b) IN; OUT



b) J cond

Având în vedere clasificarea instrucțiunilor mașină ale calculatorului didactic se propune codificarea modurilor de adresare din Tabelul 5.6.

RM(RI _{13:15})	MOD(RI ₈ ,RI ₉)				
	00	01	10	11	
000	[BA+XA]	[BA+XA+]	[BA+XA] + Depls	RA*	0
001	[BA+XB]	[BA+XB+]	[BA+XB] + Depls	RB*	1
010	[BB+XA]	[BB+XA+]	[BB+XA] + Depls	RC*	2
011	[BB+XB]	[BB+XB+]	[BB+XB] + Depls	IS*	3
100	[XA]	[BA+-XA-]	[XA] + Depls	XA*	4
101	[XB]	[BB+-XA-]	[XB] + Depls	XB*	5
110	[BA]	Depls	[BA] + Depls	BA*	6
111	[BB]	[Depls]	[BB] + Depls	BB*	7

* registrul general conține chiar operandul

Tabelul 5.6 Calculul adresei efective

5.6 Codificarea instrucțiunilor

Ținând seama de clasificarea instrucțiunilor mașină ale calculatorului didactic se propune ca instrucțiunile din aceeași categorie să fie codificate cu coduri adiacente pentru a simplifica faza de interpretare a acestora.

În codificarea instrucțiunilor mașină se va ține cont de următoarele criterii generale :

Bitul RI₀ va separa grupul de instrucțiuni cu adresă efectivă de cele fără calcul de adresă efectivă.

Astfel:

RI₀=0 va caracteriza grupul de instrucțiuni cu calcul de adresă efectivă, iar

RI₀=1 va caracteriza grupul de instrucțiuni fără calcul de adresă efectivă .

În cadrul primului grup, bitul RI₁ va separa instrucțiunile cu o adresă de cele cu două adrese:

RI₁=0 caracterizează instrucțiunile cu o singură adresă

RI₁=1 caracterizează instrucțiunile cu două adrese

Bitul RI₂ din codul operației separă grupul de instrucțiuni cu operand imediat de celelalte:

RI₂=0 caracterizează instrucțiunile fără operand imediat,

RI₂=1 caracterizează instrucțiunile cu operand imediat.

În cadrul grupului de instrucțiuni cu o singură adresă bitul RI₃ separă instrucțiunile de transfer date sau transfer control de celelalte.

Astfel:

RI₃=0 caracterizează instrucțiunile de transfer date/control, iar

RI₃=1 caracterizează instrucțiunile operaționale cu un singur operand.

În cadrul grupului de instrucțiuni cu două adrese bitul RI₃=0 specifică instrucțiunile care nu produc rezultat ci poziționează numai indicatorii de condiții (CMP, TEST).

Biții RI_{4:6} codifică individual instrucțiunile din fiecare subgrup.

În cadrul clasei de instrucțiuni fără calcul de adresă efectivă bitul RI₃=1 separă instrucțiunile de salt condiționat de celelalte. Biții RI_{4:6} codifică individual instrucțiunile iar în cazul instrucțiunilor de salt condiționat RI_{4:7} specifică condiția de test.

Având în vedere aspectele prezentate privind codificarea instrucțiunilor mașină rezultă codificarea din Tabelul 5.11.

Tabelul 5.11 Codificarea instrucțiunilor

	RI ₀ =0 cu adresă efectivă		RI ₀ =1 fără adresă efectivă	
	RI ₁ =0 cu o adresă	0=MOV	0=MOV	0=IN
	1=	1=	1=OUT	1=
	2=PUSH	2=	2=PUSHF	2=
RI ₃ =0 transfer date/contr	3=POP	3=	3=POPF	3=
	4=CALL	4=	4=RET	4=
	5=JMP	5=	5=IRET	5=
	6=	6=	6=HLT	6=
	7=	7=	7=	7=
RI ₃ =1 operații	0=INC	0=	Jcond[RI _{4:7}]	0=
	1=DEC	1=	Jcond[RI _{4:7}]	1=
	2=NEG	2=	Jcond[RI _{4:7}]	2=
	3=NOT	3=	Jcond[RI _{4:7}]	3=
	4=S HL/SAL	4=	Jcond[RI _{4:7}]	4=
	5=SHR	5=	Jcond[RI _{4:7}]	5=
	6=SAR	6=	Jcond[RI _{4:7}]	6=
	7=	7=	Jcond[RI _{4:7}]	7=
RI ₁ =1 cu două adrese	0=	0=	0=	0=
	1=	1=	1=	1=
	2=CMP	2=CMP	2=	2=
	3=	3=	3=	3=
RI ₃ =0 nu produc rezultat	4=TEST	4=TEST	4=	4=
	5=	5=	5=	5=
	6=	6=	6=	6=
	7=	7=	7=	7=
RI ₃ =0 produc rezultat	0=ADD	0=ADD	0=	0=
	1=ADC	1=ADC	1=	1=
	2=SUB	2=SUB	2=	2=
	3=SBB	3=SBB	3=	3=
	4=AND	4=AND	4=	4=
	5=OR	5=OR	5=	5=
	6=XOR	6=XOR	6=	6=
	7=	7=	7=	7=
RI ₂ =0 operand neimediat			RI ₂ =1 operand imediat	

Obs: în cadru codificării instrucțiunea MOV a fost grupată împreună cu cele cu o adresă PUSH, POP, CALL, JMP, considerându-se faptul că se citește un singur operand în faza de preluare operanzi.

Codurile operație specifice instrucțiunilor mașină, împreună cu bitul ce precizează destinația sunt prezentate în tabelul 5.12.

Tabelul 5.12 Codurile instrucțiunilor mașină

Bit RI									
0	1	2	3	4	5	6	7		
0	0	0	0	0	0	0	d	MOV	
0	0	0	0	0	1	0	x	PUSH	
0	0	0	0	0	1	1	x	POP	
0	0	0	0	1	0	0	x	CALL	
0	0	0	0	1	0	1	x	JMP	
0	0	0	1	0	0	0	x	INC	
0	0	0	1	0	0	1	x	DEC	
0	0	0	1	0	1	0	x	NEG	
0	0	0	1	0	1	1	x	NOT	
0	0	0	1	1	0	0	x	SHL/SAL	
0	0	0	1	1	0	1	x	SHR	
0	0	0	1	1	1	0	x	SAR	
0	0	1	0	0	0	0	x	MOV	cu operand imediat
0	1	0	0	0	1	0	d	CMP	
0	1	0	0	1	0	0	d	TEST	
0	1	0	1	0	0	0	d	ADD	
0	1	0	1	0	0	1	d	ADC	
0	1	0	1	0	1	0	d	SUB	
0	1	0	1	0	1	1	d	SBB	
0	1	0	1	1	0	0	d	AND	
0	1	0	1	1	0	1	d	OR	
0	1	0	1	1	1	0	d	XOR	
0	1	1	0	0	1	0	x	CMP	cu operand imediat
0	1	1	0	1	0	0	x	TEST	cu operand imediat
0	1	1	1	0	0	0	x	ADD	cu operand imediat
0	1	1	1	0	0	1	x	ADC	cu operand imediat
0	1	1	1	0	1	0	x	SUB	cu operand imediat
0	1	1	1	0	1	1	x	SBB	cu operand imediat
0	1	1	1	1	0	0	x	AND	cu operand imediat
0	1	1	1	1	0	1	x	OR	cu operand imediat
0	1	1	1	1	1	0	x	XOR	cu operand imediat
1	0	0	0	0	0	0	x	IN	
1	0	0	0	0	0	1	x	OUT	
1	0	0	0	0	1	0	x	PUSHF	
1	0	0	0	0	1	1	x	POPF	
1	0	0	0	1	0	0	x	RET	
1	0	0	0	1	0	1	x	IRET	
1	0	0	1	1	1	0	x	HLT	
1	0	0	1	conditie			x	Jcond	

5.7 Faza de citire interpretare a instrucțiunilor mașină

Faza de citire interpretare a instrucțiunilor mașină are ca obiective principale:

- citirea instrucțiunii curente;
- separarea instrucțiunilor în clase mari;
- calculul adresei efective dacă este cazul;
- pregatirea adresei efective în registrele AM și T1.

Pentru a permite o implementare hardware simplă și o descriere AHPL sub o formă cât mai compactă, vom considera că adresa registrelor generale implicate în transfer, prelucrare sau calcul de adresă efectivă va fi generată cu ajutorul unei memorii denumită MAP, Fig.5.2.

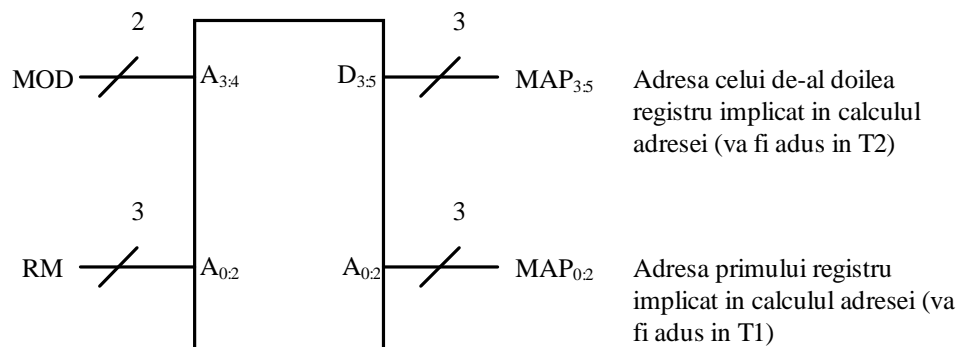


Figura 5.2

Această memorie va avea memorate adresele registrelor generale implicate, în funcție de modul de adresare.

Adresele registrelor generale sunt prezentate în tabelul 5.13.

Tabelul 5.13 Adresele registrelor generale RG

Registru	Adresă
RA	0
RB	1
RC	2
IS	3
XA	4
XB	5
BA	6
BB	7

Conținutul memoriei MAP, care generează adrese de registre generale implicate în calculul adresei efective, este prezentat în tabelul 5.14.

Descrierea fazei de citire interpretare a instrucțiunilor mașină este realizată prin programul AHPL (pașii 1...25), conform cu organigrama din Fig. 5.3.

Pentru simplificarea descrierii AHPL vom considera lucrul cu memoria principală de tip sincron.

ADR	MAP0:2	MAP3:5	ADR	MAP0:2	MAP3:5
0	6	4	10	6	4
1	6	5	11	6	5
2	7	4	12	7	4
3	7	5	13	7	5
4	4	x	14	4	x
5	5	x	15	5	x
6	6	x	16	6	x
7	7	x	17	7	x
8	6	4	18	x	x
9	6	5	19	x	x
A	7	4	1A	x	x
B	7	5	1B	x	x
C	6	4	1C	x	x
D	7	4	1D	x	x
E	x	x	1E	x	x
F	x	x	1F	x	x

Programul AHPL pentru descrierea fazei de citire interpretare.

MODULE : UCD-CD Unitatea de comandă a calculatorului didactic

MEMORY: M [65536;16]; RG [8;16]; AM [16]; CP [16]; T1 [16]; T2 [16]; IND [16]; RI [16]; AIE[8]

INPUTS: <semnalele de intrare>

COMBUS: MAG [16]

1. $\rightarrow SL(SYN(starta))/(1)$

/* se așteaptă pornirea CD, ce face și o inițializare a contorului program CP.

2. $AM \leftarrow CP$

/* se aduce adresa instrucțiunii ce trebuie citită în registrul de adrese al memoriei.

3. $RI \rightarrow BUSFN(M;DCD(AM))$

/* se citește în registrul de instrucțiuni, instrucțiunea ce se va executa.

4. $\rightarrow RI_0/(66)$

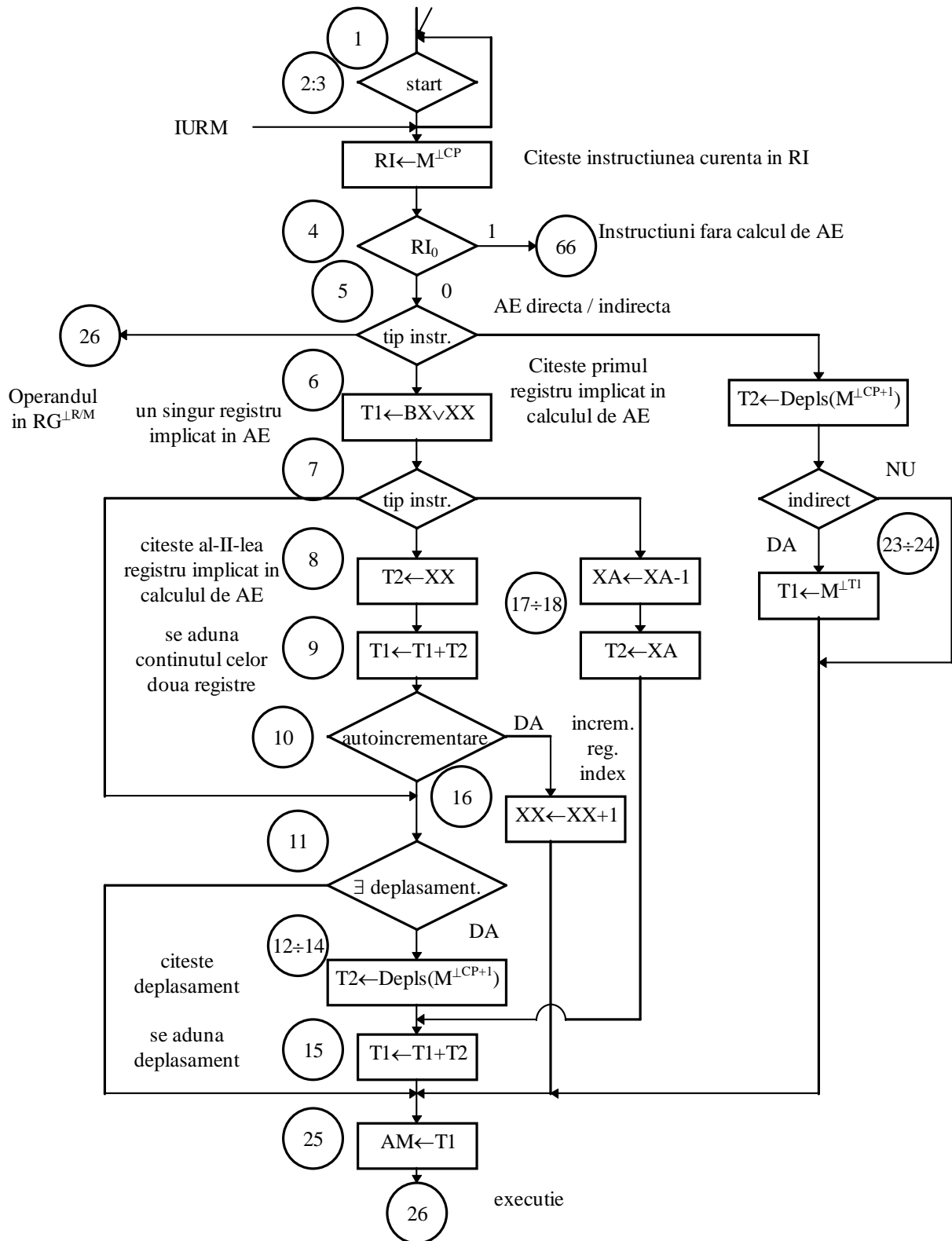
/* se separă instrucțiunile fără calcul de adresă (pas 66) de cele cu calcul de adresa și se continuă cu cele din urmă.

/* Se calculează adresa efectivă.

5. $\rightarrow ((RI_8 \wedge RI_9), (RI_8 \wedge RI_9 \wedge RI_{13} \wedge RI_{14}))/ (26, 19)$

/* se separă instrucțiunile pentru care MOD=11 (pas 26) și cele cu adresare directă sau indirectă (pas 19) de celelalte.

6. $T1 \leftarrow BUSFN(RG;DCD(BUSFN_{0,2}(MAP;DCD(MOD,RM))))$



Organigrama fazei de citire interpretarea ainstruciunilor masina

Figura 5.3

/* Se citește în registrul temporar T1, conținutul primului registru general care este utilizat la calculul adresei efective. Adresa registrului general este preluată din memoria MAP de la adresa specificată de MOD și RM

7. $\rightarrow ((RI_9 \wedge RI_{13}), \overline{(RI_9 \wedge RI_{13})}) / (11, 17)$

/* Se separă modul de adresare indirect prin registre, indexat, bazat (pas 11) și cu autodecrementare (pas 17) de celelalte moduri

8. $T2 \leftarrow \text{BUSFN}(\text{RG}; \text{DCD}(\text{BUSFN}_{3;5}(\text{MAP}; \text{DCD}(\text{MOD}, \text{RM}))))$

/* Pentru modurile de adresare rămase se citește în T2 cel de al doilea registru general ce participă la calculul adresei.

9. $T1 \leftarrow \text{ADD}(T1; T2; 0)$

/* se însumează cele două registre ce participă la calculul adresei.

10. $\rightarrow RI_9 / (16)$

/* se separă modul de adresare cu autoincrementare de celelalte

11. $\rightarrow RI_8 / (25)$

/* se separă modul de adresare cu deplasament de cel fără deplasament

12. $CP \leftarrow \text{INC}(CP)$

/* se pregătește adresa pentru citirea deplasamentului

13. $AM \leftarrow CP$

14. $T2 \leftarrow \text{BUSFN}(M; \text{DCD}(AM))$

/* se citește deplasamentul

15. $T1 \leftarrow \text{ADD}(T1; T2; 0);$

$\rightarrow (25)$

/* se adună deplasamentul la adresa efectivă și se trece controlul la faza de execuție

16. $RG * \text{DCD}(\text{BUSFN}_{3;5}(\text{MAP}; \text{DCD}(\text{MOD}, \text{RM}))) \leftarrow \text{ADD}(0; T2; 1);$

$\rightarrow (25)$

/* se incrementează registrul index utilizat în modul de adresare cu autoincrementare și se trece la faza de execuție

17. $T2 \leftarrow \text{BUSFN}(\text{RG}; \text{DCD}(\text{BUSFN}_{3;5}(\text{MAP}; \text{DCD}(\text{MOD}, \text{RM}))))$

18. $T2 \leftarrow \text{ADD}(0\text{FFFFH}; T2; 0);$

$RG * \text{DCD}(\text{BUSFN}_{3;5}(\text{MAP}; \text{DCD}(\text{MOD}, \text{RM}))) \leftarrow \text{ADD}(0\text{FFFFH}; T2; 0);$

$\rightarrow (15)$

/* se tratează modul de adresare cu autodecrementare prin decrementarea registrului index specificat și utilizarea sa la calculul adresei efective

19. $CP \leftarrow INC(CP)$

/* începe tratarea modului de adresare direct/indirect

20. $AM \leftarrow CP$

21. $T1 \leftarrow \mathbf{BUSFN}(M;DCD(AM))$

/* se citește deplasamentul

22. $\rightarrow \overline{RI}_{15}/(25)$

/* se separă modul de adresare direct de indirect

23. $AM \leftarrow T1$

24. $T1 \leftarrow \mathbf{BUSFN}(M;DCD(AM))$

/* se aduce în T1 adresa calculată indirect

25. $AM \leftarrow T1$

/* se aduce adresa efectivă în registrul AM

5.8 Faza de execuție a instrucțiunilor mașină

Faza de execuție a instrucțiunilor mașină este descrisă prin programul AHPL (pașii 26...88) conform cu organigramele din Fig.5.4, 5.5, 5.6.

/* Faza de execuție a instrucțiunilor cu calcul de adresă efectivă

/* Se separă instrucțiunile cu două adrese ADD,ADC,etc de cele cu o adresă (cu un operand) INC, DEC, etc, formându-se un grup separat cu cele de transfer date și chemare de subrutină sau salt necondiționat deoarece aceste trei mari clase au părți comune în faza de execuție.

26. $\rightarrow (RI_1)/(57)$

/* se separă instrucțiunile cu două adrese de celelalte

27. $\rightarrow (RI_3)/(54)$

/* se separă instrucțiunile cu o adresă de cele de transfer date și ramificație.

/* Execuția instrucțiunilor MOV; CALL; JMP; PUSH; POP (organigrama din Fig. 5.4)

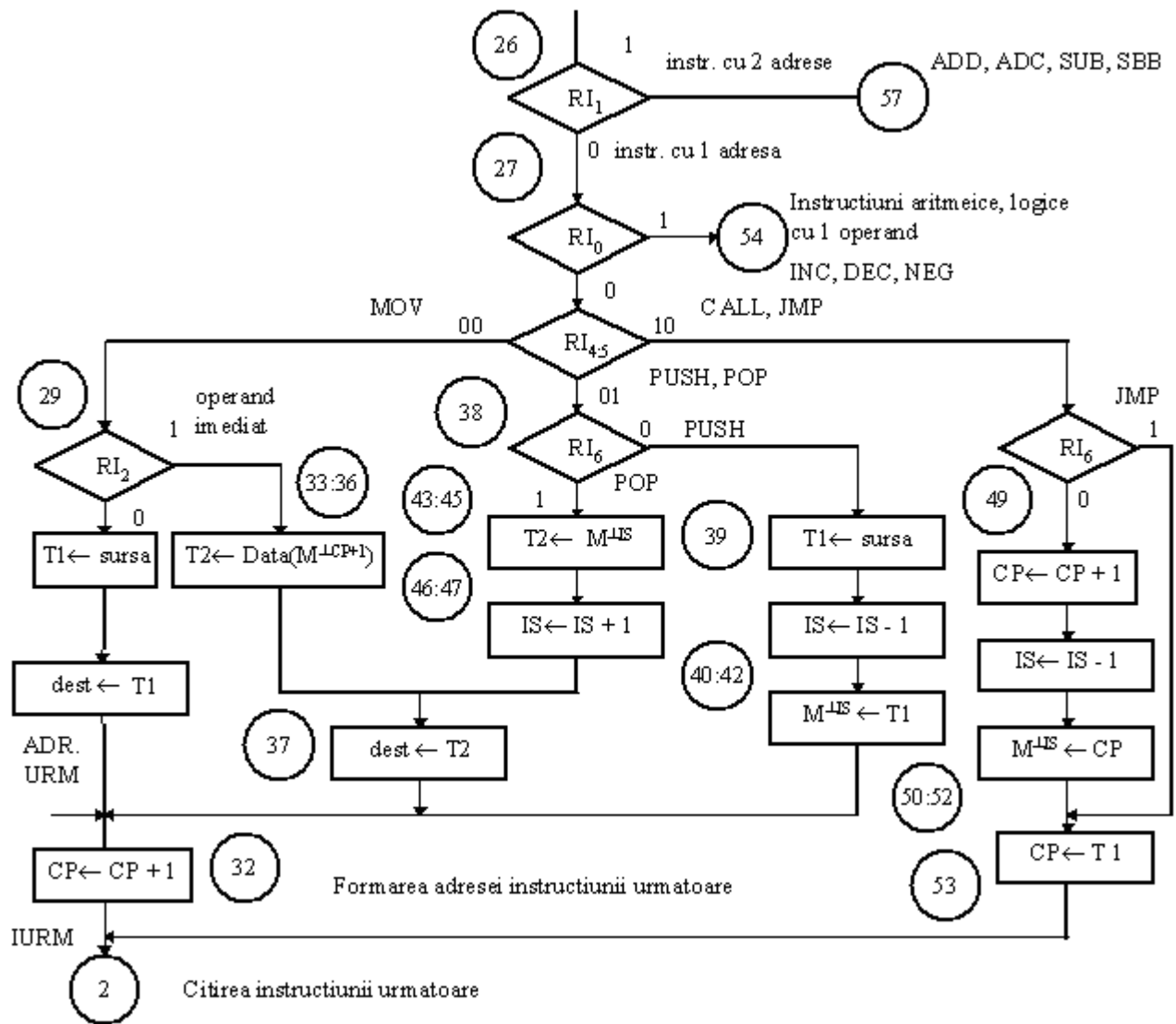


Figura 5.4

28. → (RI₄,RI₅)/(48,38)

/* se separă instrucțiunile CALL;JMP (pas48) și PUSH;POP (pas 38) de instrucțiunile MOV care se tratează în continuare

/* **Execuția instrucțiunilor MOV**

29. → (RI₂)/(33)

/* se tratează separat instrucțiunea MOV cu operand imediat de cea cu operand de la adresă efectivă; ținând seama de sensul bitului d=RI₇ și modul de adresare prezentate în formatul instrucțiunilor, operandul sursă al instrucțiunii MOV poate fi într-un registru general specificat de biții REG sau RM sau într-o celulă de memorie specificată de adresa efectivă.

dacă d=0 sursa este în RG specificată de REG iar destinația este în RG specificata de RM (MOD=11) sau în memorie la adresa efectivă (MOD=11)

dacă $d=1$ sursa este în RG specificată de RM sau în memorie la adresa efectivă iar destinația în RG specificată de REG

30. $T1 \leftarrow (\text{BUSFN}(\text{RG};\text{DCD}(\text{REG}))!\text{BUSFN}(\text{M};\text{DCD}(\text{AM}))!\text{BUSFN}(\text{RG};\text{DCD}(\text{RM})))$

$\overline{*(RI_7, RI_7 \wedge RI_8 \wedge RI_9, RI_7 \wedge RI_8 \wedge RI_9)}$

31. $((\text{RG} * \text{DCD}(\text{REG}))!(\text{M} * \text{DCD}(\text{AM}))!(\text{RG} * \text{DCD}(\text{RM})))$

$\overline{*(RI_7, RI_7 \wedge RI_8 \wedge RI_9, RI_7 \wedge RI_8 \wedge RI_9)} \leftarrow T1$

/* operandul sursă citit în registrul temporar T1 se depune la adresa efectivă sau într-un registru general

32. $CP \leftarrow \text{INC}(CP)$

$\rightarrow (2)$

/* s-a terminat de executat instrucțiunea MOV cu operandul de la adresa efectivă. Se pregătește adresa pentru instrucțiunea următoare și se reia faza de citire interpretare.

/* **Execuția instrucțiunii MOV cu operand imediat**

33. $CP \leftarrow \text{INC}(CP)$

/* se stabilește adresa pentru operandul imediat și se aduce în registrul de adrese

34. $AM \leftarrow CP$

35. $T2 \leftarrow \text{BUSFN}(\text{M};\text{DCD}(\text{AM}))$

/* Se citește operandul imediat în T2 pentru a nu pierde adresa efectivă ce se găsea în T1

36. $AM \leftarrow T1$

/* Se reface în registrul de adrese al memoriei adresa efectivă

37. $((\text{M} * \text{DCD}(\text{AM}))!(\text{RG} * \text{DCD}(\text{RM}))\overline{*(RI_8 \wedge RI_9, RI_8 \wedge RI_9)}) \leftarrow T2 ;$

$\rightarrow (32)$

/* se depune operandul sursă la adresa efectivă și se trece la formarea adresei pentru instrucțiunea următoare.

/* **Execuția instrucțiunilor PUSH și POP**

38. $\rightarrow (RI_6)/(43)$

/* se separă instrucțiunea POP. Se citește în T1 operandul ce trebuie dus în stivă

39. $T1 \leftarrow \text{BUSFN}(\text{M};\text{DCD}(\text{AM}))!(\text{BUSFN}(\text{RG};\text{DCD}(\text{RM}))\overline{*(RI_8 \wedge IR_9, RI_8 \wedge RI_9)})$

40. $T2 \leftarrow \text{BUSFN}(\text{RG};\text{DCD}(\text{ADRS}))$

/* se decrementează IS și se duce și în registrul de adrese al memoriei

41. $\text{RG} * \text{DCD}(\text{ADRS}) \leftarrow \text{ADD}(\text{0FFFFH}; T2; 0)$

42. M*DCD(AM) ← T1
→ (32)

/* se salvează operandul sursă în stivă și se merge la formarea adresei instrucțiunii următoare
/* se execută POP

43. AM ← **BUSFN**(RG;DCD(ADRIS))

44. T2 ← **BUSFN**(M;DCD(AM))
/* se citește din stivă

45. AM ← T1
/* se reface adresa efectivă din registrul de adrese al memoriei

46. T1 ← **BUSFN**(RG;DCD(ADRIS))
/* se incrementează

47. RG*DCD(ADRIS) ← ADD(T1;0;1)
→ (37)

/* **Execuția instrucțiunilor CALL; JMP**

48. → (RI6)/(53)
/* se separă instrucțiunea JMP

49. CP ← INC(CP)
/* se formează adresa instrucțiunii următoare și se salvează în tivă

50. T2 ← **BUSFN**(RG;DCD(ADRIS))

51. RG*DCD(ADRIS) ← ADD(0FFFFH;T2;0)
AM ← ADD(0FFFFH;T2;0)

52. M*DCD(AM) ← CP

53. CP ← T1
→ (2)
/* adresa efectivă se depune în contorul de program
/* Execuția instrucțiunilor cu un operand
INC; DEC; NEG; NOT; SHL/SAL; SHR; SAR
(organigrama din Fig.5.5)

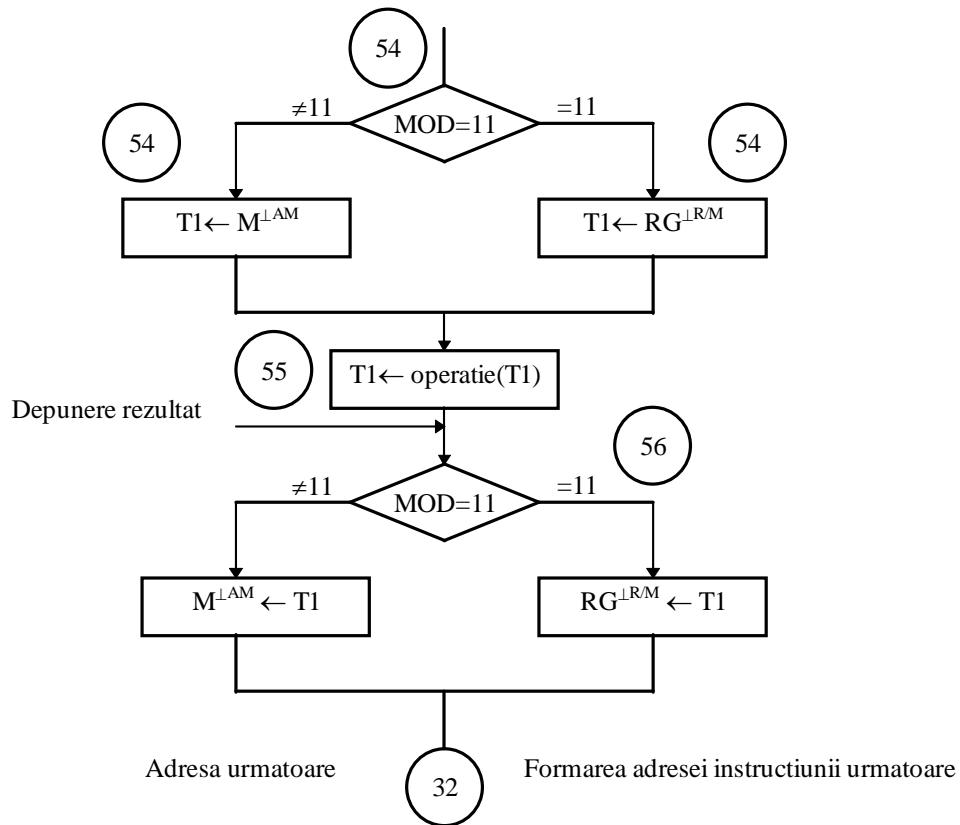


Figura 5.5

54. $T1 \leftarrow (\overline{\text{BUSFN}(M; \text{DCD}(AM))})! (\overline{\text{BUSFN}(RG; \text{DCD}(RM))}) * (\overline{RI_8 \wedge RI_9}, \overline{RI_8 \wedge RI_9}))$

/* s-a citit operandul în registrul temporar T1

55. $T1 \leftarrow (\overline{\text{ADD}(T1; 0; 1)}! \overline{\text{ADD}(T1; 0FFFFH; 0)}! \overline{\text{ADD}(T1; 0; 1)}! \overline{T1! T1_{1:15}, 0! 0, T1_{0:14}}! \overline{T1_0, T1_{0:14}}) * \overline{\text{DCD}_{0:6}(RI_{4:6})}$

/* Z,S,D,T,P-se pozitionează corespunzător rezultatului

56. $(\overline{M * \text{DCD}(AM)}! \overline{RG * \text{DCD}(RM)}) * (\overline{RI_8 \wedge RI_9}, \overline{RI_8 \wedge RI_9}) \leftarrow T1$

→ (32)

/* Execuția instrucțiunilor cu doi operanzi ADD;ADC;SUB;SBB;AND;OR;XOR;CMP;TEST (organigrama din Fig.5.6)

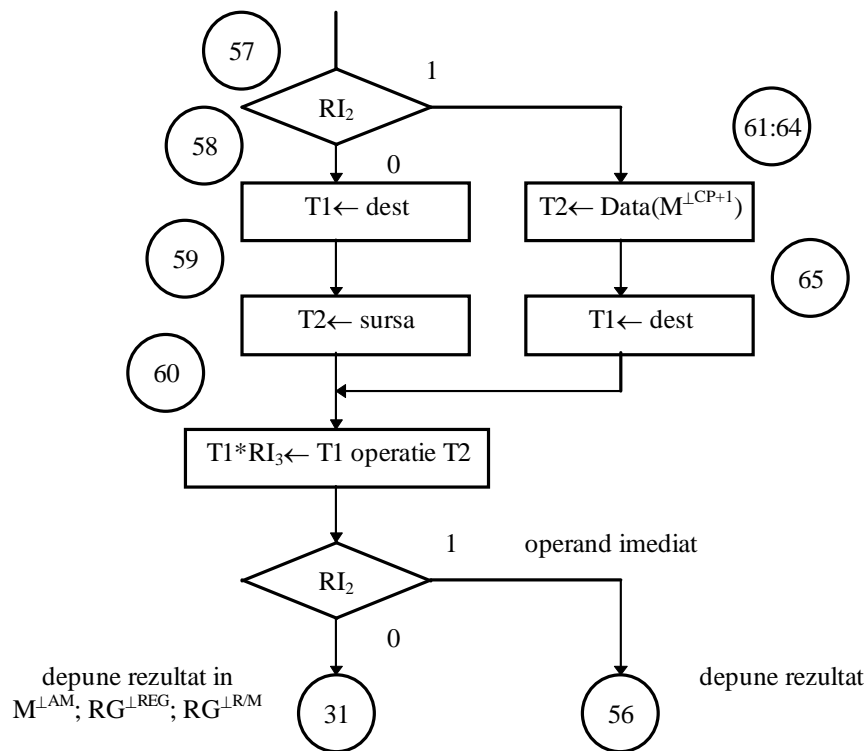


Figura 5.6

57. $\rightarrow (RI_2)/(61)$

/* se separă instrucțiunile cu operand imediat

58. $T1 \leftarrow (\overline{BUSFN}(RG;DCD(REG))!BUSFN(M;DCD(AM))!BUSFN(RG;DCD(RM)))$

$*(\overline{RI_7}, \overline{RI_7 \wedge RI_8 \wedge RI_9}, \overline{RI_7 \wedge RI_8 \wedge RI_9})$

/* d=0 R/M ← R/M cop REG

/* d=1 REG ← REG cop R/M

/* se citește în T1 operandul 1

59. $T2 \leftarrow (\overline{BUSFN}(RG;DCD(REG))!BUSFN(M;DCD(AM))!BUSFN(RG;DCD(RM)))$

$*(\overline{RI_7}; \overline{RI_7 \wedge RI_8 \wedge RI_9}); \overline{RI_7 \wedge RI_8 \wedge RI_9}$

/* se citește în T2 operandul 2

/* $RI_3=0$ caracterizează instrucțiunile ce nu produc rezultat și poziționează indicatorii de stare (CMP;TEST)

60. $T1*RI_3 \leftarrow (\overline{ADD}(T1;T2;0)!ADD(T1;T2;T)!ADD(T1;T2;1)!ADD(T1;T2;T) \wedge \overline{AND}(T1;T2)!OR(T1;T2)!XOR(T1;T2))*DCD_{0:6}(RI_{4:6})$

/* Z,S,D,T,P-se pozitionează conform rezultatului

$\rightarrow \overline{(RI_2, RI_2)} / (31, 56)$

/* execuția instrucțiunilor cu doi operanzi, dintre care unul este imediat

61. $CP \leftarrow INC(CP)$

/* se citește operandul imediat

62. $AM \leftarrow CP$

63. $T2 \leftarrow \mathbf{BUSFN}(M; DCD(AM))$

/* se refăce adresa efectivă

64. $AM \leftarrow T1$

65. $T1 \leftarrow (\mathbf{BUSFN}(M; DCD(AM)) \mathbf{!BUSFN}(RG; DCD(RM)) * \overline{(RI_8 \wedge RI_9, RI_8 \wedge RI_9)})$

$\rightarrow (60)$

**/*Execuția instrucțiunilor fără calcul de adresă efectivă IN;OUT;PUSHF;POPF;RET;IRET;HLT
(organigrama din Fig.5.7)**

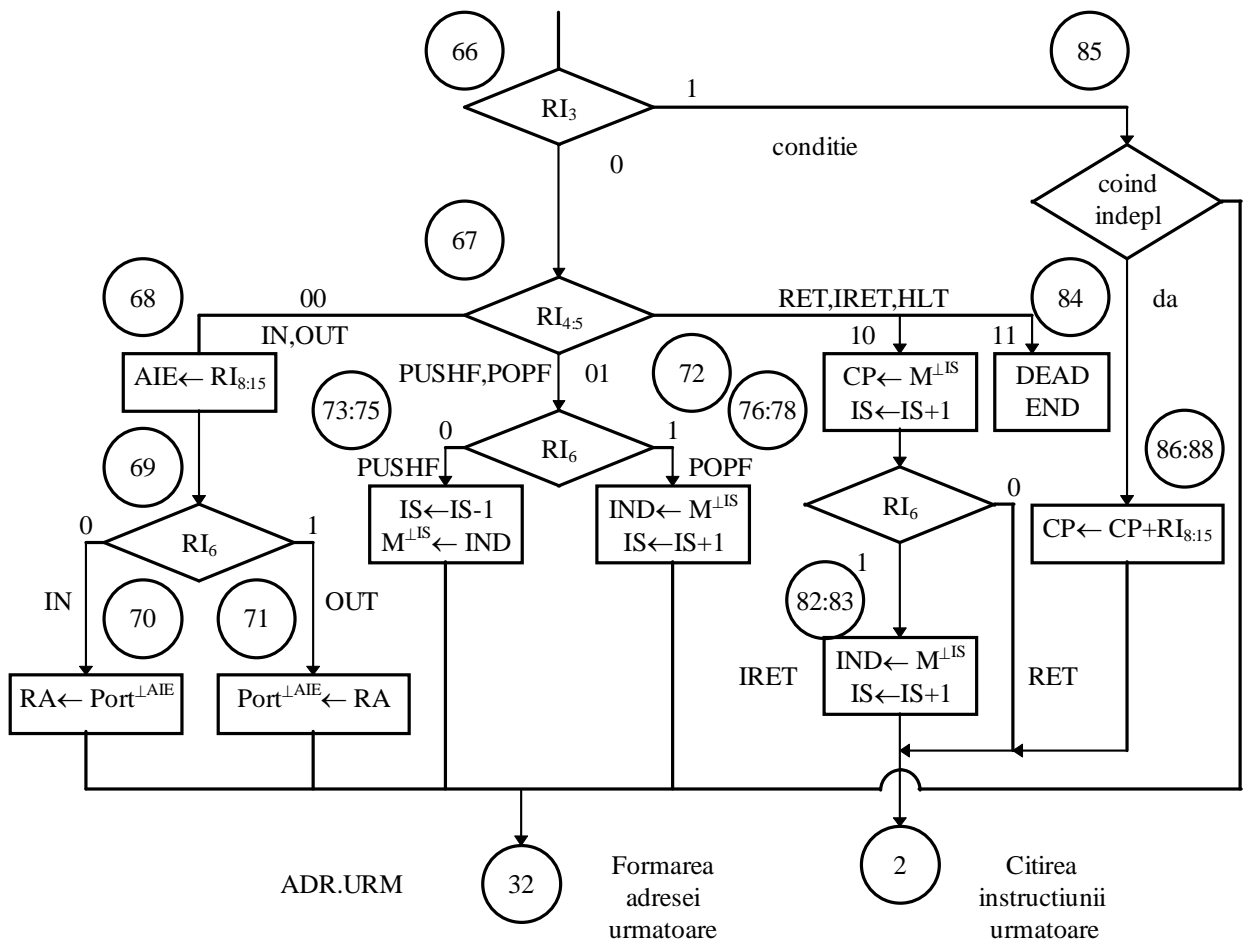


Figura 5.7

66. → RI₃/(85)

/* se separă instrucțiunile de salt condiționat

67. → (RI₄∧RI₅, RI₄∧RI₅, RI₄∧RI₅)/(72,79,84)

/* se separă instrucțiunile PUSHF; POPF (pas 72), instrucțiunile RET; IRET (pas 79) și instrucțiunea HLT (pas(84)

/* Execuția instrucțiunilor IN, OUT

68. AIE ← RI_{8:15}

69. → RI₆/(71)

/* se separă instrucțiunea OUT

70. RG*DCD(ADRR) ← BUSFN(PORT;DCD(AIE))
→ (32)

/* se execută instrucțiunea IN

71. $\text{PORT} * \text{DCD}(\text{AIE}) \rightarrow \text{BUSFN}(\text{RG}; \text{DCD}(\text{ADRR}))$

$\rightarrow (32)$

/* se execută instrucțiunea OUT

/* Execuția instrucțiunilor PUSHF; POPF

72. $\rightarrow (\text{RI}_6)/(76)$

/* se separă POPF

/* în continuare se execută PUSHF

7

3. $\text{T1} \leftarrow \text{BUSFN}(\text{RG}; \text{DCD}(\text{ADRS}))$

74. $\text{RG} * \text{DCD}(\text{ADRS}) \leftarrow \text{ADD}(\text{T1}; \text{OFFFH}; 0);$

$\text{AM} \leftarrow \text{ADD}(\text{T1}; \text{OFFFH}; 0)$

75. $\text{M} * \text{DCD}(\text{AM}) \leftarrow \text{IND}$

$\rightarrow (32)$

/* s-a executat instrucțiunea PUSHF

76. $\text{AM} \leftarrow \text{BUSFN}(\text{RG}; \text{DCD}(\text{ADRS}))$

$\text{T1} \leftarrow \text{BUSFN}(\text{RG}; \text{DCD}(\text{ADRS}))$

77. $\text{IND} \leftarrow \text{BUSFN}(\text{M}; \text{DCD}(\text{AM}))$

78. $\text{RG} * \text{DCD}(\text{ADRS}) \leftarrow \text{ADD}(\text{T1}; 0; 1)$

$\rightarrow (32)$

/* s-a executat instrucțiunea POPF

/* Execuția instrucțiunilor RET; IRET

79. $\text{AM} \leftarrow \text{BUSFN}(\text{RG}; \text{DCD}(\text{ADRS}))$

$\text{T1} \leftarrow \text{BUSFN}(\text{RG}; \text{DCD}(\text{ADRS}))$

80. $\text{CP} \leftarrow \text{BUSFN}(\text{M}; \text{DCD}(\text{AM}))$

/* se reface contorul program

81. $\text{RG} * \text{DCD}(\text{ADRS}) \leftarrow \text{ADD}(\text{T1}; 0; 1)$

$\text{AM} \leftarrow \text{ADD}(\text{T1}; 0; 1)$

$\text{T1} \leftarrow \text{ADD}(\text{T1}; 0; 1)$

$\rightarrow \overline{\text{RI}}_6/(2)$

/* se separă RET a cărei execuție s-a terminat

82. $\text{IND} \leftarrow \text{BUSFN}(\text{M}; \text{DCD}(\text{AM}))$

/* se execută IRET

83. $RG*DCD(ADRI) \leftarrow ADD(T1;0;1)$

→ (2)

/* Execuția instrucțiunii HLT

84. DEAD END

/* Execuția instrucțiunilor de salt condiționat

85. → condiție / (32)

/* Condiție neîndeplinită, salt la instrucțiunea următoare

86. $T1 \leftarrow CP$

87. $T2 \leftarrow RI_{8:15}$

/* transferul se face cu extensia semnului RI_8 pe biții 0:7)

88. $CP \leftarrow ADD(T1;T2;0)$

→ (2)

5.9 Completarea setului de instrucțiuni

În procesul de elaborare programe, în limbaj de asamblare, poate apare necesitatea unor operații de rotire (stânga/dreapta), de repetare a unei secțiuni de program specificate, de transfer a unui șir de date dintr-o zonă de memorie în alta, etc.

Aceste operații se pot realiza prin setul de instrucțiuni mașină al calculatorului didactic, însă dacă apar foarte des în programele utilizatorilor, ar fi de preferat (pentru a mări viteza de lucru) ca aceste operații să fie rezolvate direct de instrucțiuni mașină .

În cele ce urmează vom încerca să completăm setul de instrucțiuni mașină pentru implementarea directă a unor operații. Se vor analiza aspectele implicate de introducerea de noi instrucțiuni mașină în setul existent .

Aspectele care apar în procesul de introducere a unor instrucțiuni noi sunt următoarele :

- alegerea funcției și a mnemonice;
- stabilirea formatului instrucțiunii;
- codificarea instrucțiunii;
- modificarea fazei de citire interpretare în scopul recunoașterii instrucțiunii;
- descrierea fazei de execuție a instrucțiunii.

În vederea prezentării mecanismului de completare a setului instrucțiuni vom introduce instrucțiuni ce realizează operațiile:

- rotire stânga/dreapta cu sau fără transport;
- repetare a unei secțiuni de program;
- interschimbarea conținutului a doi operanzi.

Alegerea funcției și a mnemonice

Pentru realizarea operațiilor propuse alegem :

Mnemonică	Funcție
RCL	-rotire stânga de un număr de ori
RCR	-rotire dreapta de un număr de ori
ROL	-rotire stânga prin indicatorul de transport de un număr de ori
ROR	-rotire dreapta prin indicatorul de transport de un număr de ori
LOOP	-repetarea unei secțiuni de program de un număr de LOOPZ ori
XCHG	-interschimbă conținutul a doi operanzi

5.9.1 Introducerea instrucțiunilor RCL, RCR, ROL, ROR

5.9.1.1 Formatul și funcțiile instrucțiunilor RCL, RCR, ROL, ROR

Formatul instrucțiunilor RCL, RCR, ROL, ROR este :

COP c MOD --- RM

Instrucțiunea RCL realizează funcția :

dacă c=0 atunci CONTOR = 1
altfel CONTOR = (RC)

atât timp cât CONTOR ≠ 0 execută

$T, OPERAND_{0:15} \leftarrow OPERAND_{0:15}, OPERAND_0$
CONTOR = CONTOR - 1

Instrucțiunea RCR realizează funcția :

dacă c=0 atunci CONTOR = 1
altfel CONTOR = (RC)

atât timp cât CONTOR ≠ 0 execută

$T, OPERAND_{0:15} \leftarrow OPERAND_{15}, OPERAND_{15}, OPERAND_{0:14}$
CONTOR = CONTOR - 1

Instrucțiunea ROL realizează funcția :

dacă c=0 atunci CONTOR = 1
altfel CONTOR = (RC)

atât timp cât CONTOR ≠ 0 execută

$T, OPERAND_{0:15} \leftarrow OPERAND_{0:15}, T$
CONTOR = CONTOR - 1

Instrucțiunea ROR realizează funcția :

dacă c=0 atunci CONTOR = 1
altfel CONTOR = (RC)

atât timp cât CONTOR ≠ 0 execută

$T, OPERAND_{0:15} \leftarrow OPERAND_{15}, T, OPERAND_{0:14}$
CONTOR = CONTOR - 1

5.9.1.2 Codificarea instrucțiunilor RCL, RCR, ROL, ROR

Analizând codificarea instrucțiunilor mașină ale calculatorului didactic se observă că aceste instrucțiuni fac parte din clasa de instrucțiuni cu un singur operand.

Codul 0001111 din această categorie de instrucțiuni nu este utilizat. Vom folosi acest cod pentru codificarea acestui grup de instrucțiuni iar biții $RI_{10:12} = \text{REG}$, de asemenea neutilizați îi vom folosi pentru codificarea individuală.

Rezultă codurile:	0001111 c MOD 000 RM	- RCL
	0001111 c MOD 001 RM	- RCR
	0001111 c MOD 010 RM	- ROL
	0001111 c MOD 011 RM	- ROR

5.9.1.3 Modificarea fazei de citire interpretare în scopul recunoașterii acestor instrucțiuni

Pentru a lua în considerare grupul de instrucțiuni ROL, ROR, RCL, RCR se introduce după pasul AHPL 54. care a citit operandul în registrul temporar T1, un pas AHPL care separă aceste instrucțiuni și face legătura cu secvența de execuție a acestora.

54.1 $\rightarrow \text{DCD}_7(RI_{4:6}) / (90)$

Se consideră că execuția acestor instrucțiuni începe la pasul AHPL 90.

5.9.1.4 Descrierea fazei de execuție

90. $T2 \leftarrow (\text{BUSFN}(\text{RG}; \text{DCD}(\text{ADRRC}))!16T1) * (RI_7, RI_7)$

91. $T, T1 \leftarrow ((T1_0, T1_{1:15}, T1_0)!(T1_{15}, T1_{15}, T1_{0:14})!(T1_0, T1_{1:15}, T)!$
 $(T1_{15}, T, T1_{0:14})) * \text{DCD}_{0:3}(RI_{11:12})$

92. $T2 \leftarrow \text{ADD}(\text{OFFFH}; T2; 0);$
 $\rightarrow ((\sqrt{T2}), (\sqrt{T2})) / (91, 56)$

5.9.2 Introducerea instrucțiunilor LOOP, LOOPZ

Instrucțiunile LOOP și LOOPZ sunt utilizate pentru a realiza repetarea unei secțiuni de program de un număr prestabilit de ori, altfel spus realizarea ciclurilor în program.

5.9.2.1 Formatul și funcțiile instrucțiunilor LOOP și LOOPZ.

Instrucțiunile LOOP și LOOPZ realizează un salt relativ la CP stabilit de valoarea deplasamentului din cadrul instrucțiunii.

Aceste instrucțiuni se pot introduce în grupul instrucțiunilor de salt condiționat. Fac parte din categoria instrucțiunilor fără calcul de adresă efectivă.

Formatul general este :
 COP Deplasament

Instrucțiunea LOOP realizează funcția:

$RC \leftarrow RC - 1$

dacă $RC \neq 0$ atunci $CP \leftarrow CP + \text{Deplasament}$
 altfel $CP \leftarrow CP + 1$

registru RC este decrementat și se realizează saltul relativ la CP cu o valoare specificată de Deplasament dacă conținutul registrului RC este diferit de zero, iar în caz contrar se continuă cu instrucțiunea următoare.

Deplasamentul trebuie să aibă o valoare între -128 și 127 și reprezintă o adresă relativă față de adresa instrucțiunii curente LOOP.

Instrucțiunea LOOPZ realizează funcția:

$RC \leftarrow RC - 1$
 dacă $(RC \neq 0) \wedge (Z = 1)$ atunci $CP \leftarrow CP + \text{Deplasament}$
 altfel $CP \leftarrow CP + 1$

registru RC se decrementează și se realizează saltul relativ la CP cu o valoare specificată de Deplasament dacă conținutul registrului RC este diferit de zero și indicatorul Z este unu, iar în caz contrar ($RC = 0$ sau $Z = 0$) se continuă cu instrucțiunea următoare. Deplasamentul trebuie să fie aibă o valoare între -128 și 127 și reprezintă o adresă relativă față de adresa instrucțiunii curente LOOPZ.

5.9.2.2 Codificarea instrucțiunilor LOOP și LOOPZ

Deoarece aceste instrucțiuni fac parte din grupul instrucțiunilor de salt condiționat vom stabili o codificare adiacentă cu a acestora. Propunem codificarea:

1 0 1 1 0 x x x Deplasament pentru instrucțiunea LOOP
 1 0 1 1 1 x x x Deplasament pentru instrucțiunea LOOPZ

5.9.2.3 Modificarea fazei de citire interpretare în scopul recunoașterii acestor instrucțiuni

Pentru a lua în considerare și acest grup de instrucțiuni pasul AHPL 85, care realizează execuția instrucțiunilor de salt condiționat, se modifică astfel:

85. $\rightarrow (\overline{RI_2}, RI_2 \wedge \text{condiție}) / (93, 32)$

S-a ținut seama că cele două grupuri de instrucțiuni diferă numai prin bitul RI_2 din codul de operație. La pasul 32 se merge în cazul instrucțiunilor de salt condiționat când condiția nu este îndeplinită. Pasul 86 constituie continuarea execuției instrucțiunilor de salt condiționat când condiția este îndeplinită, iar pasul 93 reprezintă începutul secvenței de execuție a instrucțiunilor LOOP și LOOPZ.

5.9.2.4 Descrierea fazei de execuție

93. $T2 \leftarrow \text{BUSFN}(\text{RG}; \text{DCD}(\text{ADRRC}))$

94. $\text{RG} * \text{DCD}(\text{ADRRC}) \leftarrow \text{ADD}(0\text{FFFFH}; T2; 0)$

95. $\rightarrow ((\sqrt{T2}) \vee \overline{RI_4} \wedge \overline{Z}) / (32)$

96. $T1 \leftarrow CP$

97. $T2 \leftarrow RI_{8:15}$

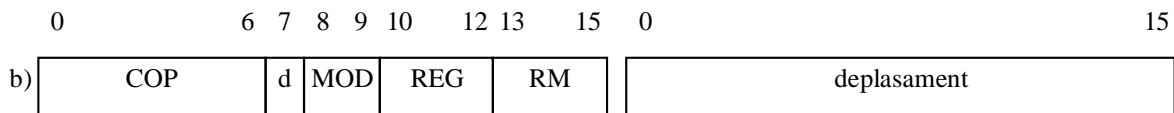
98. $CP \leftarrow ADD(T1;T2;0)$
 $\rightarrow (93)$

5.9.3 Introducerea instrucțiunii XCHG

Instrucțiunea XCHG are rolul de a interschimba doi operanzi ce se găsesc, fie ambii în registrele generale, fie unul într-un registru general și cel de al doilea în memorie.

5.9.3.1 Formatul și funcția instrucțiunii XCHG.

Instrucțiunea XCHG face parte din grupul instrucțiunilor cu două adrese. Formatul general al instrucțiunii este:



Funcția îndeplinită de instrucțiunea XCHG este:

dacă $MOD = 11$ atunci $RG^{DCD(REG)} \leftrightarrow RG^{DCD(RM)}$
 altfel $RG^{DCD(REG)} \leftrightarrow M^{DCD(AM)}$

5.9.3.2 Codificarea instrucțiunii XCHG

Făcând parte din grupul instrucțiunilor cu două adrese vom stabili o codificare în clasa ADD, ADC, etc

Codificarea propusă este:

0 1 0 1 1 1 1 x MOD REG RM

5.9.3.3 Modificarea fazei de citire interpretare în scopul recunoașterii acestei instrucțiuni

Pentru a lua în considerare această instrucțiune vom introduce înaintea începutului execuției grupului de instrucțiuni ADD, ADC, SUB, SBB, etc, adică înaintea pasului AHPL 58, un pas 57.1 care separă această instrucțiune de grupul respectiv.

57.1 $\rightarrow (DCD(RI_{4:6})) / (99)$

S-a considerat ca la pasul AHPL 99 va începe execuția instrucțiunii XCHG.

5.9.3.4 Descrierea fazei de execuție

99. $T1 \leftarrow (\mathbf{BUSFN}(M;DCD(AM))!\mathbf{BUSFN}(RG;DCD(RM))) * \overline{(RI_8 \wedge RI_9, RI_8 \wedge RI_9)}$

100. $T2 \leftarrow \mathbf{BUSFN}(RG;DCD(REG))$

101. $RG * DCD(REG) \leftarrow T1$

102. $((M*DCD(AM))!(RG*DCD(RM))) * \overline{(RI_8 \wedge RI_9, RI_8 \wedge RI_9)} \leftarrow T2$

$\rightarrow (32)$

S-a citit primul operand în T1, al doilea operand în T2, după care depunerea s-a făcut în ordine inversă. În final se merge la pasul AHPL 32 unde se pregătește adresa pentru instrucțiunea următoare.

În acest mod se pot introduce orice instrucțiune care este absolut necesară în setul de instrucțiuni mașină.