5. STRUCTURA ŞI ARHITECTURA CALCULATORULUI DIDACTIC

OBIECTIVE

În capitolul 5 se definesc structura şi arhitectura calculatorului didactic, care va fi dezvoltat în continuare şi în capitolele care urmează. Astfel sunt elaborate specificaţiile primitivelor funcţionale şi modul de interconectare a acestora. De asemenea în acest capitol se urmăreşte funcţionarea detaliată a unui calculator numeric prin parcurgerea tuturor etapelor de specificare, descriere, proiectare şi evaluare a unităţii de execuţie şi a unităţii de comandă a calculatorului didactic.

CUPRINS

5. STRUCTURA ŞI ARHITECTURA CALCULATORULUI DIDACTIC 1

5.1 Primitivele funcţionale ale calculatorului didactic 2

5.2 Schema bloc şi fluxul informaţiei 4

5.3 Setul de instrucţiuni 6
5.3.1 Instrucţiuni de transfer date 7
5.3.2 Instrucţiuni aritmetice 8
5.3.3 Instrucţiuni de transfer control 13

5.4 Formatul instrucţiunilor calculatorului didactic 14

5.5 Moduri de adresare 16

5.6 Codificarea instrucţiunilor 28

5.7 Faza de citire interpretare a instrucţiunilor maşină 32

5.8 Faza de execuţie a instrucţiunilor maşină 37

5.9 Completarea setului de instrucţiuni 46
5.9.1 Introducerea instrucţiunilor RCL, RCR, ROL, ROR 47
5.9.1.1 Formatul şi funcţiile instrucţiunilor RCL, RCR, ROL, ROR 47
5.9.1.2 Codificarea instrucţiunilor RCL, RCR, ROL, ROR 47
5.9.1.3 Modificarea fazei de citire interpretare în scopul recunoaşterii acestor instrucţiuni 48
5.9.1.4 Descrierea fazei de execuţie 48
5.9.2 Introducerea instrucţiunilor LOOP, LOOPZ 48
5.9.2.1 Formatul şi funcţiile instrucţiunilor LOOP şi LOOPZ. 48
5.9.2.2 Codificarea instrucţiunilor LOOP şi LOOPZ 49
5.9.2.3 Modificarea fazei de citire interpretare în scopul recunoaşterii acestor instrucţiuni 49
5.9.2.4 Descrierea fazei de execuţie 49
5.9.3 Introducerea instrucţiunii XCHG 50
5.9.3.1 Formatul şi funcţia instrucţiunii XCHG. 50
5.9.3.2 Codificarea instrucţiunii XCHG 50

Pentru a facilita însuşirea 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 şi realizării calculatoarelor numerice moderne. Considerarea unui calculator didactic ca primă etapă în însuşirea conceptelor privind funcţionarea şi realizarea unui calculator numeric oferă, faţă de analiza unui calculator real, mai multe avantaje cum ar fi:

- participarea directă la proiectarea calculatorului;
- o mare elasticitate în ceea ce priveşte opţiunile proiectului în diferite etape ale procesului de proiectare;
- o mai mare îndependenţă faţă de constrîngerile şi detaliile de implementare.

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

5.1 Primitivele funcţionale ale calculatorului didactic

În continuare se vor prezenta, pe scurt, primitivele funcţionale ale calculatorului didactic şi funcţiile acestora.

Memoria M

$r_1M=16$; $r_2M=65536$

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

Memoria este utilizată pentru a păstra informaţii neinterpretate reprezentând date sau instrucţiuni. Oricare două celule de memorie sunt accesibile în mod echivalent. Totuşi, memoria poate fi logic împărţită în segmente de instrucţiuni, date, stivă, şi segmente de memorie disponibilă. Pentru segmentul de instrucţiuni se poate prevedea şi o memorie cu conţinut permanent de tip "numai citire".

Citirea şi scrierea se fac asincron sub controlul unităţii de comandă.

Registrul AM

rAM=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 $2^{rAM} \geq r_2M$. Adresa calculată (adresa efectivă) este memorată în AM selectând, prin decodificare, cuvântul din memorie la care se va face accesul.

Registrele 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. Registrele RG conţin unul sau ambii operanzi necesari pentru execuţia instrucţiunilor CD. Unele din aceste registre vor fi utilizate şi pentru calculul adresei efective a operanzilor din memorie. Astfel:

- BA,BB sunt utilizate ca registre de bază;
- XA,XB sunt utilizate ca registe 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 instrucţiunii. Selectarea registrelor generale va fi prezentată detaliat în paragraful 5.4.

Registrul CP

rCP=16

Registrul contor program CP este utilizat pentru păstrarea adresei instrucţiunii ce urmează să se execute după terminarea execuţiei instrucţiunii curente. Lungimea lui CP se alege astfel ca $2^{rCP} \geq r_2M$. Registrul CP poate fi iniţializat cu o valoare dată la iniţializarea sistemului, cu o valoare oarecare prin execuţia instrucţiunilor de transfer control (vezi paragraful 5.3) sau poate fi incrementat în cazul execuţiei instrucţiunilor (operaţionale) ce nu specifică transferul controlului la o altă secvenţă.

Unitatea aritmetică logică UAL

rUAL=16

Unitatea aritmetică logică (UAL) realizează operaţiile aritmetice şi logice ale CD şi este utilizată pentru prelucrarea datelor şi pentru calculul adresei efective. UAL este de tip paralel, prelucrează operanzi pe 16 biţi reprezentaţi în cod complementar. UAL implementează direct toate operaţiile elementare necesare execuţiei instrucţiunilor aritmetice şi logice ale CD. Condiţiile în care s-a efectuat o operaţie în UAL şi caracteristicile rezultatului sunt păstrate într-un registru de indicatori IND.

Registrele T1,T2

rT1=rT2=16

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

Indicatorii de condiţii IND

rIND = 16

Registrul de indicatori constituie o grupare a unor bistabili cu funcţii individuale, pozitionaţi la execuţia instrucţiunilor în funcţie de rezultatul din UAL. Registrul IND permite alegerea unei secvenţe de execuţie următoare unei operaţii aritmetice/logice în funcţie de rezultatul acestei operaţii. Lungimea registrului IND este de 16 deşi numai o parte din aceştia sunt utilizaţi în mod efectiv. Funcţiile acestor bistabili indicator vor fi prezentate în paragraful 5.3.2.

Registrul RI

rRI = 16

Registrul de instrucţiuni RI păstrează codul instrucţiunii în curs de execuţie. Conţinutul său este decodificat şi transmis secţiunii de generare comenzi/verificare stări din unitatea de comandă. în RI se păstrează şi informaţiile necesare pentru selecţia registrelor generale în funcţie de instrucţiunea în curs de execuţie.

Magistrala MAG

rMAG = 16

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

Instrucţiunile calculatorului didactic

Specificarea transferurilor de informaţii între resursele CD şi transformările suferite în timpul transferului se face prin intermediul instrucţiunilor. Instrucţiunile CD sunt grupate astfel:

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

Instrucţiunile de transfer date vor fi utilizate atunci când este necesară o transformare spaţială a datelor fără a modifica conţinutul informaţiilor transferate. Acest grup include instrucţiuni de transfer între resursele interne ale unităţii de execuţie sau între unitatea de execuţie şi subsistemul de intrări/ ieşiri.

Instrucţiunile aritmetice şi logice trebuie să permită efectuarea operaţiilor aritmetice/logice uzuale.

Grupul de instrucţiuni de transfer control este necesar pentru implementarea structurilor de control care asigură înlănţuirea secvenţelor de execuţie a instrucţiunilor conform cu un algoritm dat, descris prin program.

5.2 Schema bloc şi fluxul informaţiei

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

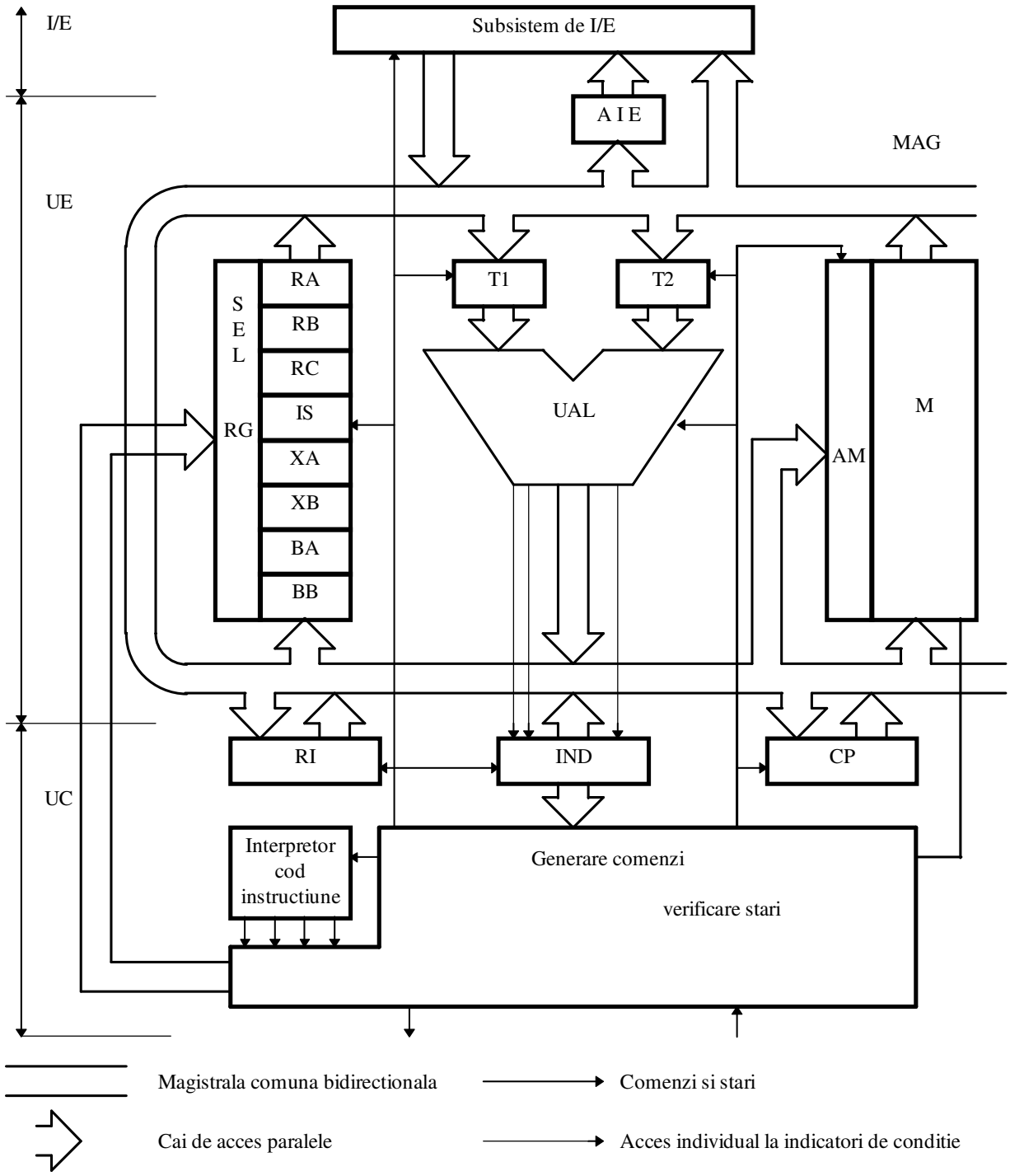
- unitatea de execuţie (UE);
- unitatea de comandă (UCda);
- subsistemul de intrări /ieşiri (I/E).

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

Fluxul datelor trebuie astfel organizat încît să permită preluarea datelor primare din mediul extern prin intermediul subsistemului de intrări/ieşiri, prelucrarea acestora în unitatea de execuţie UE şi transmiterea rezultatelor spre mediul extern.

Datele preluate din exterior prin subsistemul de intrări/ieşiri 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 memoraţi în RG. În procesul de prelucrare operanzii sunt preluaţi din RG şi transferaţi 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 şi

acestea. Există şi posibilitatea preluării ambilor operanzi din memorie sau a unui operand din memorie şi 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 poziţionat direct de către UAL. Există posibilitatea transferului direct, prin MAG, între IND şi memorie pentru salvarea/restaurarea indicatorilor de condiţii.

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

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

Tabelul 5.1 Posibilităţi transfer date în CD

	Destinaţie										
Sursă	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 instrucţiuni RI este încărcat din memorie cu codul instrucţiunii curente iar conţinutul său este accesibil unităţii de comandă pentru interpretare.

Fluxul informaţiilor în calculatorul didactic, pentru execuţia unui program, poate fi rezumat în felul următor.

Se citeşte din memorie instrucţiunea a cărei adresă se găseşte în CP. Primul cuvânt al instrucţiunii se citeşte în RI şi este utilizat de UC pentru a controla mai departe operaţiile ce urmează să se execute. Se citesc, dacă este cazul, următoarele cuvinte ale instrucţiunii (instrucţiunile pot fi formate din 1,2 sau 3 cuvinte de 16) şi se citesc operanzii. Aceste operaţii formează faza de citire - interpretare a instrucţiunii. Urmează apoi faza de execuţie care execută prelucrarea efectivă a operanzilor conform cu codul operaţiei specificat în RI. Rezultatul este depus în memorie sau în RG după cum este specificat în RI. Pentru execuţia operaţiilor de I/E datele sunt transferate între registrul RA şi subsistemul de I/E.

5.3 Setul de instrucţiuni

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

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

Pentru specificarea operaţiilor logice şi de calcul sunt necesare instrucţiuni aritmetice şi logice care să permită realizarea unui set minim de operaţii cum ar fi: *adunarea şi scăderea* (sau adunarea şi schimbarea semnului); *conjuncţia, disjuncţia şi negaţia* (sistem complet de funcţii); *deplasarea arimetică şi logică* .

Pentru implementarea structurilor de control ale unui algoritm sunt necesare instrucţiuni de transfer control cum ar fi: *salt necondiţionat* la o anumită secvenţă de instrucţiune; *testarea condiţiilor de terminare* a operaţiilor aritmetice/ logice şi ramificarea execuţiei programului în funcţie de aceste condiţii.

Pentru transferul datelor primare şi a rezultatelor între calculator şi mediul extern sunt necesare instrucţiuni de intrare/ieşire.

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

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

5.3.1 Instrucţiuni de transfer date

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

Tabelul 5.2 Instrucţiuni de transfer date

Instrucţiuni CU SCOP GENERAL	MOV PUSH POP	Transferă cuvânt Transferă cuvânt în stivă Transferă cuvânt din stivă
Instrucţiuni INTRARI / IEŞIRI	IN OUT	intrare cuvânt ieşire cuvânt
Instrucţiuni TRANSFER INDICATORI	PUSHF POPF	Transferă indicatori în stivă Transferă indicatori din stivă

În continuare, se prezintă funcţiile efectuate la execuţia acestora(Tabelul 5.3).

Tabelul 5.3 Instrucţiuni cu scop general

Format instrucţiune	Funcţie
MOV *destinaţie,sursă*	MOV transferă un cuvânt specificat de operandul sursă la operandul destinaţie. Conţinutul indicatorilor nu se schimbă.
PUSH *sursă*	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 *destinaţie*	POP transferă cuvântul din vârful stivei indicat de IS la operandul destinaţie 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 toţi indicatorii în cuvântul din vârful stivei indicat de IS. Conţinutul indicatorilor nu se schimbă.
POPF	POPF transferă biţii corespunzători din cuvântul aflat în vârful stivei în indicatorii de condiţii înlocuind vechile valori ale acestora. IS este incrementat pentru a indica noul vârf al stivei.
IN *port*	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 şi 255.
OUT *port*	OUT transferă un cuvânt din RA la portul de ieşire specificat prin adresa "port".

Observaţii:

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

IN şi OUT nu modifică indicatorii.

5.3.2 Instrucţiuni aritmetice

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

În tabelul 5.4 se specifică instrucţiunile aritmetice care intră în setul de instrucţiuni al calculatorului didactic.

Tabelul 5.4 Instrucţiuni aritmetice

Instrucţiuni ADUNARE	ADD	Adună cuvânt
	ADC	Adună cuvânt şi transport
	INC	Incrementează cuvânt
Instrucţiuni SCADERE	SUB	Scade cuvânt
	SBB	Scade cuvânt cu împrumut
	DEC	Decrementează cuvânt
	NEG	Schimbă semnul la cuvânt
	CMP	Compară_ cuvânt

Pentru a analiza rezultatul operaţiilor aritmetice se utilizează indicatori de condiţii prezentaţi în tabelul 5.5.

Tabelul 5.5 Indicatorii de condiţii

Indicator	Funcţie
T (transport):	– este setat (pozitionat în "1") dacă în urma unei adunări rezultă un transport dinspre rangul cel mai semnificativ, altfel T este şters (trecut în "0"). Transportul T este setat dacă în urma unei scăderi rezultă un împrumut în cel mai semnificativ bit al rezultatului, altfel este şters. – poate fi interpretat ca depăşire în operaţiile cu numere întregi fără semn. – poate fi utilizat în instrucţiunile ADC şi SBB pentru a efectua operaţii aritmetice în precizie multiplă (32 de biţi sau 64 biţi). – poate fi testat cu instrucţiuni de salt condiţionat.
S (semn):	– la execuţia instrucţiunilor aritmetice şi logice indicatorul S este setat la valoarea bitului cel mai semnificativ al rezultatului (bitul de semn). Pentru numere cu semn în absenţa depăşirii S=0 indică rezultat pozitiv iar S=1 indică rezultat negativ. – poate fi testat cu instrucţiuni de salt condiţionat. În cazul operaţiilor cu numere fără semn S poate fi ignorat deoarece în acest caz specifică cel mai semnificativ bit al rezultatului.
Z (zero):	– este setat dacă în urma unei operaţii aritmetice sau logice se obţine rezultat egal cu 0, altfel Z este şters. – poate fi testat cu instrucţiuni de salt condiţionat pentru a dirija secvenţa de execuţie a instrucţiunilor în funcţie de valoarea rezultatului.
P (paritate):	– indicatorul de paritate P este setat dacă în urma execuţiei unei operaţii aritmetice sau logice rezultatul conţine un număr par de biţi egali cu 1, altfel P este şters. – poate fi testat cu instrucţiuni de salt condiţionat.
D (depăşire):	– indicatorul de depăşire D este setat dacă în urma execuţiei unei operaţii aritmetice rezultatul este un număr pozitiv prea mare sau un număr negativ prea mic pentru a putea fi reprezentat în operandul destinaţie (exclusiv bitul de semn); altfel D este şters. – poate fi interpretat ca depăşire în operaţiile cu numere întregi cu semn şi poate fi testat cu instrucţiuni de salt condiţionat. – poate fi ignorat în operaţiile aritmetice cu numere întregi fără semn.

În continuare se prezintă formatul instrucţiunilor aritmetice şi funcţiile efectuate la execuţia acestora (tabelul 5.6).

Tabelul 5.6 Instrucţiuni aritmetice

Format instrucţiune	Descriere
ADD *destinaţie, sursă*	Suma celor doi operanzi trece în locul operandului destinaţie. Operanzii pot fi numere binare cu semn sau fără semn. ADD pozitionează indicatorii T, S, Z, P, D.
ADC *destinaţie, sursă*	ADC adună cei doi operanzi şi mai adună 1 dacă T=1 iar rezultatul îl trece în locul operandului destinaţie. Operanzii pot fi numere binare cu sau fără semn. ADC se utilizează pentru operaţii în precizie multiplă. ADC poziţionează toţi indicatorii de condiţii.
INC *destinaţie*	INC adună 1 la operandul destinaţie considerat ca un număr pe 16 biţi fără semn. INC poziţionează bistabilii de condiţii cu excepţia lui T.
SUB *destinaţie,sursă*	Operandul sursă este scăzut din operandul destinaţie şi rezultatul este trecut în locul operandului destinaţie. Operanzii pot fi numere binare cu sau fără semn. SUB pozitionează toţi indicatorii de condiţii.
SBB *destinaţie,sursă*	SBB scade operandul sursă din operandul destinaţie şi mai scade 1 dacă T=1. Rezultatul trece în locul operandului destinaţie. Operanzii pot fi numere binare cu sau fără semn. SBB se utilizează pentru operaţii în precizie multiplă. SBB pozitionează toţi indicatorii de condiţii.
DEC *destinaţie*	DEC scade 1 din operandul destinaţie considerat ca un număr pe 16 biţi fără semn. DEC poziţionează indicatorii cu excepţia lui T.
NEG *destinaţie*	Operandul destinaţie se scade din 0 şi rezultatul trece în locul operandului destinaţie. Aceasta operaţie inversează semnul unui întreg, trecându-l sub formă de complement faţă de doi. Dacă se încearcă negarea lui -32768, operandul rămâne neschimbat şi se setează indicatorul de depăşire, D=1. NEG poziţionează toţi indicatorii de condiţii. T este întotdeauna setat exceptând situaţia în care operandul este zero.
CMP *destinaţie,sursă*	CMP scade operandul sursă din operandul destinaţie dar rezultatul nu se reţine. Operanzii rămân neschimbaţi iar indicatorii de condiţii sunt poziţionaţi corespunzător şi pot fi testati cu instrucţiuni de salt condiţionat.

5.3.3 Instrucţiuni logice şi de deplasare

În acest grup au fost incluse instrucţiunile care operează asupra biţilor unui cuvânt prin funcţii logice şi de deplasare şi sunt prezentate în Tabelul 5.7.

Instrucţiunile logice afectează indicatorii astfel: T şi D sunt întotdeauna zero în urma unei operaţii logice iar Z, S şi P sunt poziţionati în funcţie de rezultatul operaţiei şi pot fi testaţi cu instrucţiunile de salt condiţionat, ca şi în cazul instrucţiunilor aritmetice.

Tabelul 5.7 Instrucţiuni logice şi de deplasare

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

În continuare se prezintă formatul instrucţiunilor logice şi de deplasare şi funcţiile efectuate la execuţia acestora.

Tabelul 5.8 Formatul instrucţiunilor logice

Formatul instrucţiunii	Funcţie
NOT *destinaţie*	NOT inversează (complementul faţă de 1) biţii cuvântului dat de operandul destinaţie.
AND *destinaţie,sursă*	AND efectuează "şi" logic între biţii celor doi operanzi şi trece rezultatul în locul operandului destinaţie.
OR *destinaţie,sursă*	OR efectuează "SAU" logic între biţii celor doi operanzi şi trece rezultatul în locul operandului destinaţie.
XOR *destinaţie,sursă*	XOR efectuează "SAU-exclusiv" între biţii celor doi operanzi şi trece rezultatul în locul operandului destinaţie.
TEST *destinaţie,sursă*	TEST efectuează "şi" logic între cei doi operanzi şi nu se reţine rezultatul. Cei doi operanzi rămân neschimbaţi iar indicatorii sunt pozitionaţi corespunzător, putând fi testaţi cu instrucţiunile de salt condiţionat.

Instrucţiunile de deplasare se utilizează pentru a deplasa aritmetic sau logic biţii unui cuvânt. Deplasarea aritmetică se utilizează pentru înmulţirea/împărţirea numerelor cu puteri ale lui 2 iar deplasarea logică se utilizează pentru izolarea unor biţi dintr-un cuvânt. Bistabilii de condiţii sunt afectaţi de instrucţiunile de deplasare în felul următor: S, Z, P sunt pozitionaţi în mod obişnuit, ca şi în cazul instrucţiunilor logice. T conţine întotdeauna bitul deplasat în afara operandului destinaţie iar D este setat dacă în urma operaţiei de deplasare bitul cel mai semnificativ (semn) şi-a schimbat valoarea, altfel D este şters.

Tabelul 5.8. Formatul instrucţiunilor de deplasare

Formatul instrucţiunii	Funcţie
SHL/SAL *destinaţie*	SHL/SAL realizează deplasarea la stânga cu o poziţie a operandului destinaţie. În bitul cel mai puţin semnificativ se introduce zero. Deplasarea logică şi aritmetică la stânga cu o poziţie produc acelaşi rezultat. S-au prevăzut două mnemonice pentru a mări flexibilitatea de exprimare a utilizatorului, în funcţie de contextul de prelucrare a datelor.
SHR *destinaţie*	SHR deplasează logic la dreapta biţii operandului destinaţie introducînd zero în bitul cel mai semnificativ.
SAR *destinaţie*	SAR deplasează aritmetic la dreapta biţii operandului destinaţie. Deplasarea se face cu extensia bitului de semn (bitul de semn rămîne neschimbat iar bitul cel mai semnificativ de date preia conţinutul bitului de semn).

5.3.3 Instrucţiuni de transfer control

Ordinea în care se execută instrucţiunile unui program pentru calculatorul didactic este determinată de secvenţa de valori reprezentând conţinutul registrului CP.

Tabelul 5.9 Instrucţiuni de transfer control

Instrucţiuni TRANSFER NECONDIŢIONAT	CALL	Salt necondiţionat la o procedură	
	RET	Revenire necondiţionată dintr-o procedură	
	IRET	Revenire necondiţionată dintr-o procedură de întrerupere	
	JMP	Salt necondiţionat la o anumită adresă	
Instrucţiuni TRANSFER CONDIŢIONAT		condiţia testată	salt dacă
	JA	(T OR Z)=0	"above";d>s (d,s fără semn)
	JAE	T=0	"above or equal";d≤s(d,s fără semn)
	JB	T=1	"below";d<s (d,s fără semn)
	JBE	(T OR Z)=1	"below or equal";d≥s(d,s fără semn)
	JC	T=1	"carry"; transport
	JE	Z=1	"equal"; d=s
	JG	((S XOR D)OR Z)=0	"greater";d>s (d,s cu semn)
	JGE	(S XOR D)=0	"greater or equal";d≥s (d,s cu semn)
	JL	(S XOR D)=1	"less"; d<s (d,s cu semn)
	JLE	((S XOR D)OR Z)=1	"less or equal";d≤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 depăşire
	JPO	P=0	"parity odd";paritate impară
	JNS	S=0	"not sign";rezultat pozitiv
	JO	D=1	"overflow";depăşire
	JPE	P=1	"parity even";paritate pară
	JS	S=1	"sign";rezultat negativ
Nota s=sursă ; d=destinaţie			

Secvenţa normală de execuţie a unui program se obţine prin incrementarea contorului program CP, la fiecare instrucţiune pentru a obţine instrucţiunea următoare.

Instrucţiunile de transfer control operează asupra contorului program. Incărcarea registrului CP cu o valoare oarecare determină alterarea secvenţei normale de execuţie a programului. Transferul controlului la o altă secvenţă se poate face necondiţionat sau condiţionat de valorile indicatorilor de condiţii. Instrucţiunile de transfer control sunt prezentate în Tabelul 5.5.

Instrucţiunea HLT a fost inclusă în acest grup deşi efectul execuţiei acestei instrucţiuni este suspendarea execuţiei programului şi trecerea UC a calculatorului didactic într-o stare de aşteptare (se aşteaptă un reset sau o întrerupere).

Tabelul 5.9 Formatul instrucţiunilor de transfer control

Formatul instrucţiunii	**Funcţie**
CALL nume-procedura	CALL transferă controlul la procedura definită în cadrul programului şi salvează în stivă informatiile necesare revenirii în programul principal (chemător). Revenirea se face la instrucţiunea imediat următoare instrucţiunii CALL, prin executarea, în procedura chemată, a unei instrucţiuni RET. Indiferent de modul în care se calculează adresa procedurii, UC decrementează registrul IS şi salvează apoi în stivă adresa instrucţiunii următoare lui CALL. Valoarea calculată, reprezentând adresa procedurii înlocuieşte apoi conţinutul lui CP şi secvenţa continuă cu execuţia procedurii
RET	RET transferă controlul dintr-o procedură înapoi în programul chemător, la instrucţiunea imediat următoare instrucţiunii care a activat, prin CALL, procedura. RET extrage conţinutul celulei indicată de IS (vârful stivei) şi-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 secvenţă de operaţii, conţinutul celulei din vârful stivei, indicată de IS, este transferat în CP, se incrementează IS şi conţinutul celulei indicată de IS (actualizat) este transferat în registrul de indicatori după care IS se încrementează din nou şi programul continuă cu instrucţiunea de la adresa aflată în CP. IRET poziţionează indicatorii de condiţii conform conţinutului din stivă.
JMP destinaţie	JMP transferă controlul la locaţia indicată de destinaţie. JMP nu salvează nimic în stivă, execuţia programului continuă cu instrucţiunea indicată de adresa calculată (adresa destinaţie).
Jcond "destinaţie"	Instructiunile de salt condiţionat transferă controlul la altă secvenţă numai dacă se îndeplineşte condiţia de test specificată în instrucţiune. Dacă nu este îndeplinită condiţia de salt se continuă execuţia programului cu instrucţiunea următoare. Adresa "destinaţie" trebuie să fie cuprinsă în limitele - 128 şi +127 . Saltul se face relativ la contorul program : CP + " destinaţie ".
HLT	HLT trece UCP într-o stare de aşteptare a unei întreruperi sau reiniţializare a secvenţei de comandă. Altfel spus din starea de HALT se poate ieşi numai în urma unei întreruperi (continuându-se cu instrucţiunea următoare lui HLT) sau cu un reset (care introduce 0 în CP).

5.4 Formatul instrucţiunilor calculatorului didactic

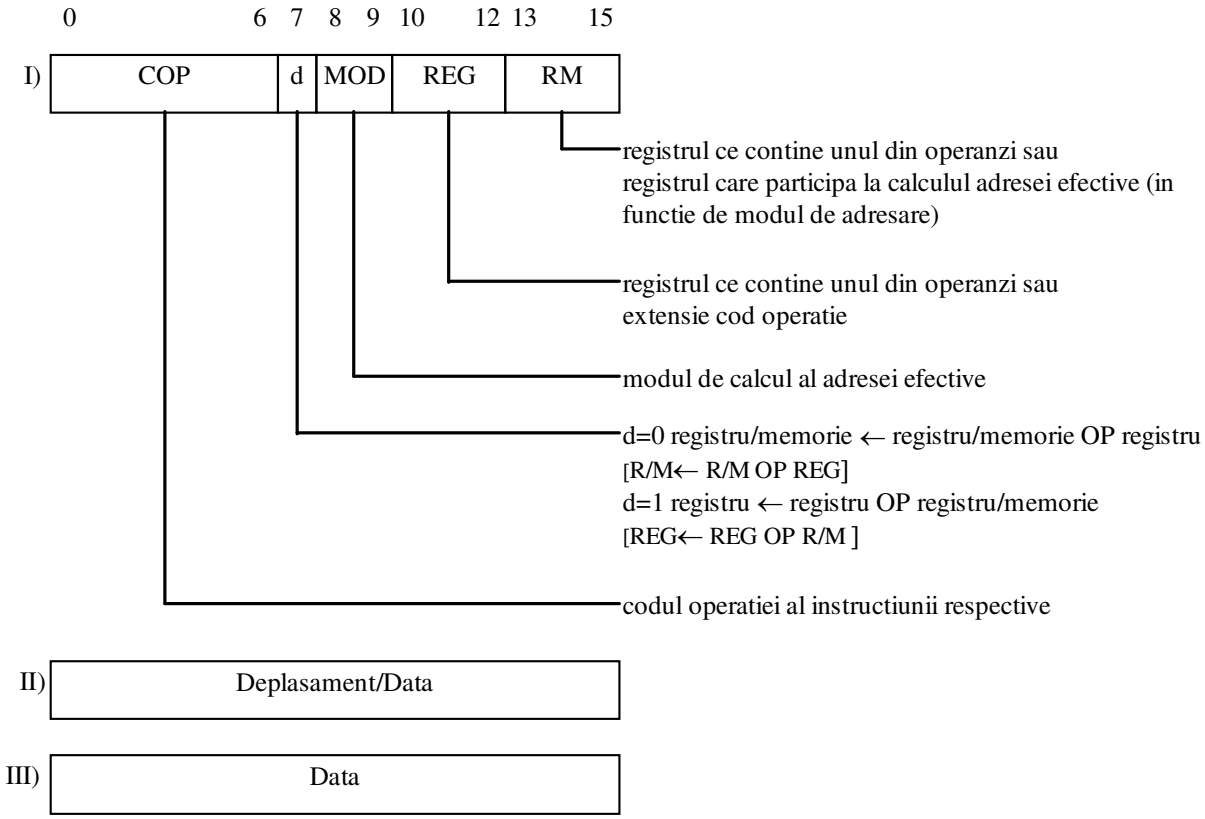
Repertoriul de instrucţiuni al calculatorului didactic a fost alcătuit în vederea ilustrării principalelor categorii de instrucţiuni întâlnite în calculatoarele moderne şi pentru a satisface cerinţele privind transferul

informaţiei, operaţiile aritmetice şi logice, transferul comenzii în cadrul programului, comanda procesorului, etc. Aceste instrucţiuni, prin formatul pe care îl au, prin modalităţile extrem de variate pentru calculul adreselor efective ale operanzilor asigură facilităţi pentru manipularea unor structuri complexe de date, cum sunt tablourile unidimensionale şi bidimensionale de date, diverse tipuri de înregistrări, fişiere, etc.

Instrucţiunile calculatorului didactic pot manipula operanzi aflaţi în registrele unităţii centrale, operanzi continuţi în instrucţiuni sau operanzi aflaţi în memorie, adresele efective ale acestor operanzi fiind calculate în diferite moduri, asigurând o mare supleţe în adresare.

Instrucţiunile au lungimi variabile de 1, 2 sau 3 cuvinte de cîte 16 biţi, lungimea fiind dictată de tipul operaţiei codificate, de numărul de operanzi, de modul de adresare, etc.

În Fig. 5.2 este prezentată structura instrucţiunii calculatorului didactic.



Primul cuvânt din instrucţiune conţine 5 câmpuri distincte :

COP, d, MOD, REG, RM

Tabelul 5.10 Câmpurile primului cuvânt al instrucţiunii CD

Câmp	Descriere
COP	reprezintă codul operaţiei, are 7 biţi şi reprezintă funcţia ce trebuie efectuată
d	specifică destinaţia rezultatului tinînd seama de faptul că cei doi operanzi ai instrucţiunii sunt indicaţi cu ajutorul câmpurilor REG şi RM. d = 0 , RM ← RM COP REG d = 1 , REG ← REG COP RM
MOD	arată modul de calcul al adresei efective a operandului şi necesită 2 biţi
REG	specifică registrul care conţine operandul, sau se foloseste ca extensie a câmpului COP, ocupă 3 biţi
RM	indică registrul care conţine un operand sau registrul folosit pentru calculul adresei efective.

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

Al treilea cuvânt al instrucţiunii, 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 operanzilor implicaţi în instrucţiunea curentă.

Instrucţiunile calculatorului didactic pot prelucra maxim doi operanzi. Aceştia se pot găsi :

-ambii în registrele generale RG ;

-unul în registrele generale RG şi altul în memorie ;

-unul în registrele generale RG şi altul în cadrul instrucţiunii respective (operand imediat) ;

-unul în memorie şi altul imediat.

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

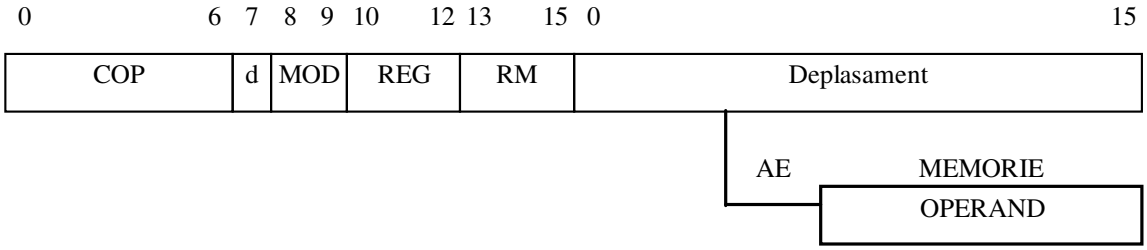
Pentru a asigura o mare flexibilitate în prelucrarea datelor alegem, pentru calculatorul didactic, următoarele moduri de adresare:

1. **Adresare directă**

Adresa efectivă este prevăzută în instrucţiune, în cuvîntul al doilea, Fig. 5.3.

AE = Depls

Exemplu: mov RA,adresă

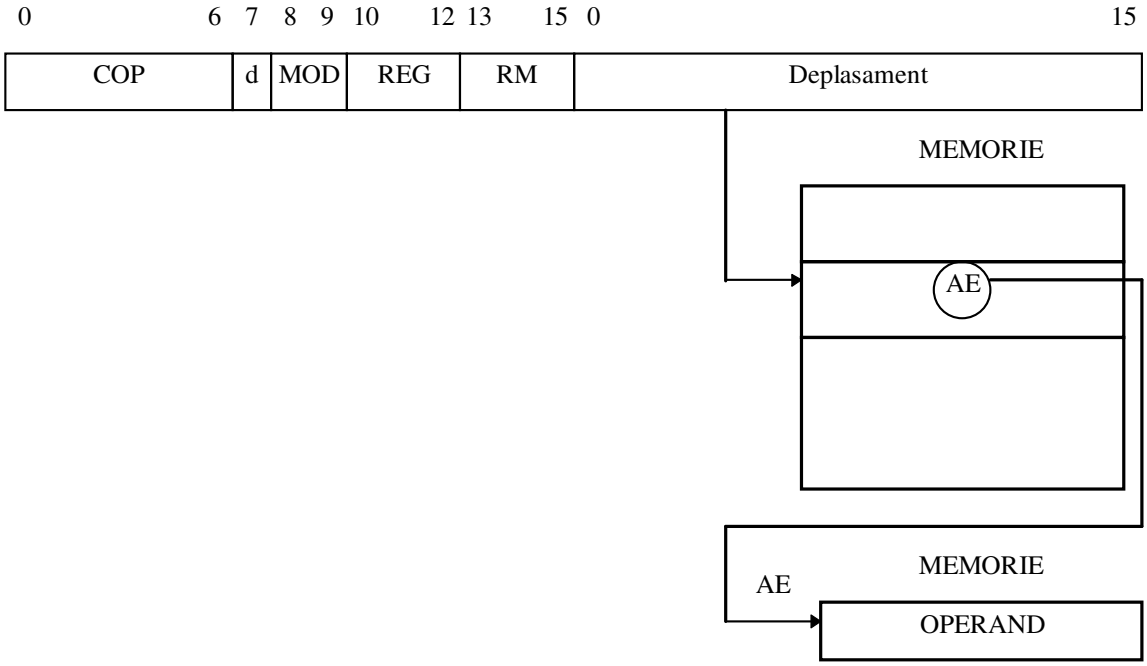


2. **Adresare indirectă**

Adresa efectivă se citeşte din memorie, din locaţia a cărei adresă este conţinută în al doilea cuvânt al instrucţiunii, Fig.5.4.

AE = [Depls]

Exemplu: mov RA,[adresa]

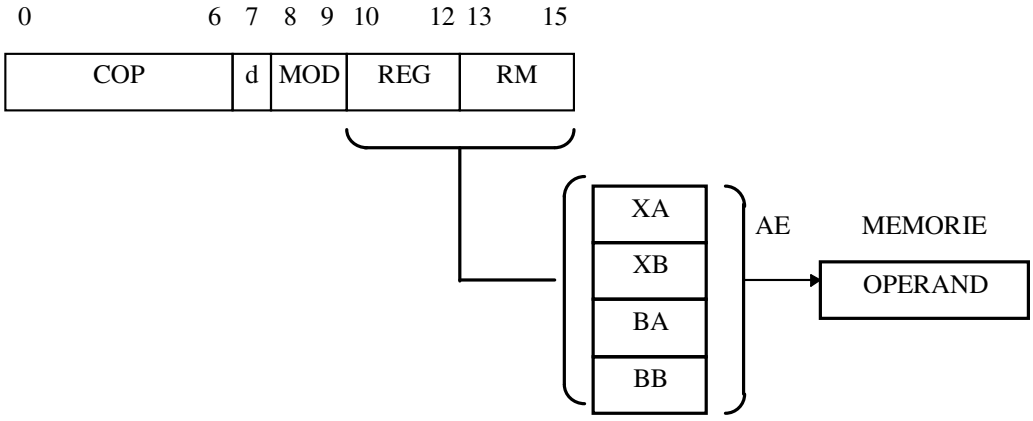


3. **Adresare indirectă prin registru**

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

AE = [XA, XB, BA, BB]

Exemplu: mov RA,[BA]

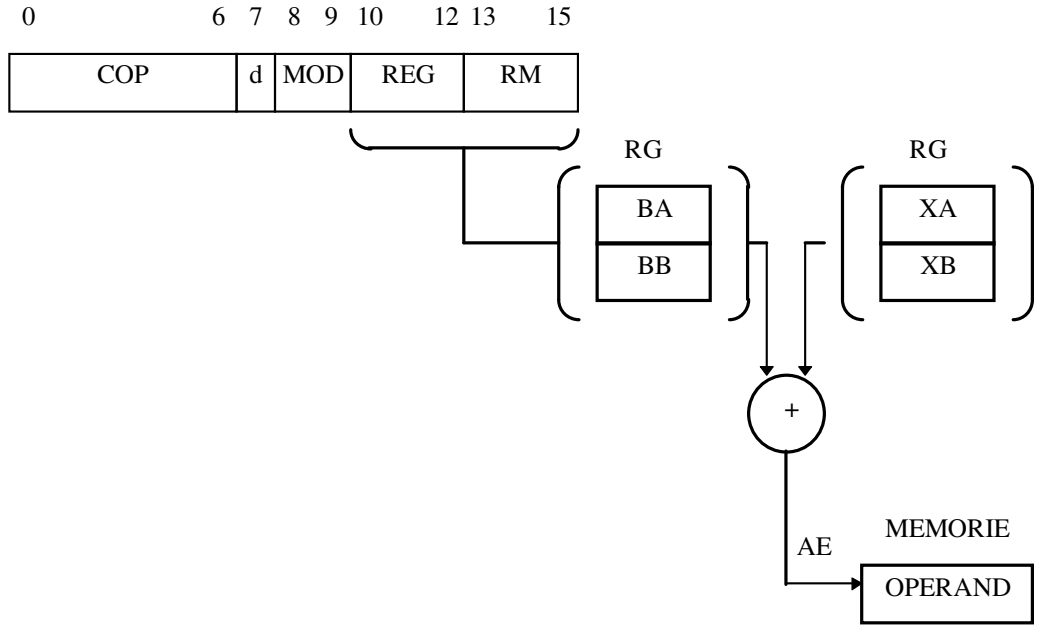


4. **Adresare indirectă prin sumă de registre**

Adresa efectivă se obţine ca sumă a conţinutului unui registru de bază cu conţinutul unui registru index, Fig. 5.6.

AE = [BA+XA, BA+XB, BB+XA, BB+XB]

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.**

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

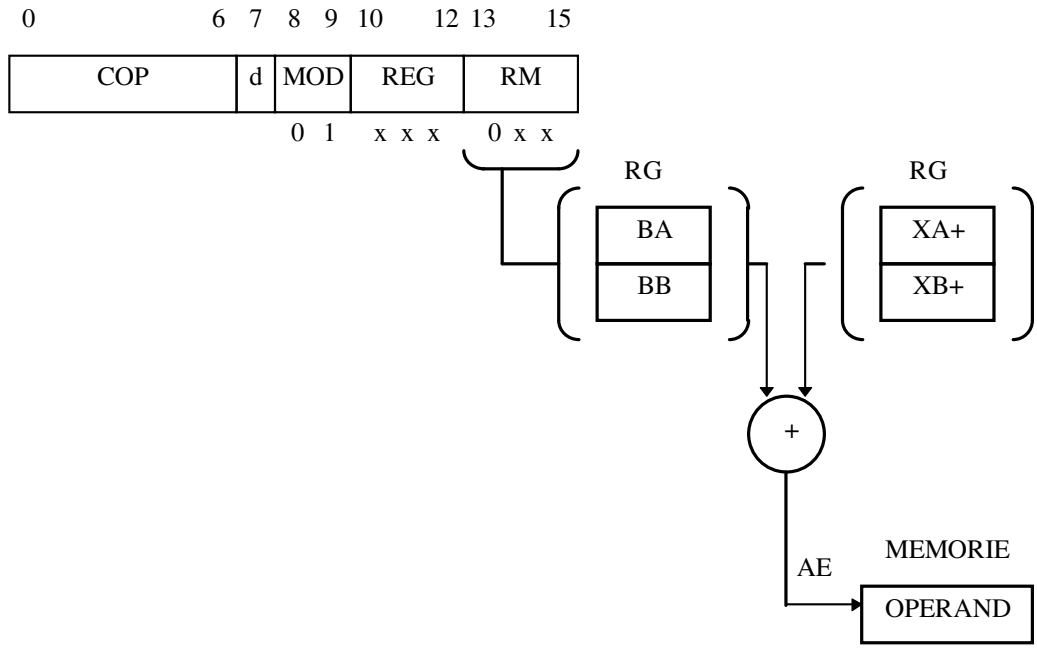
$$AE = \begin{bmatrix} BA+XA+ \\ BA+XB+ \\ BB+XA+ \\ BB+XB+ \end{bmatrix}$$

ceea ce este echivalent cu :

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

; XA $\leftarrow$ XA + 1
; XB $\leftarrow$ 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 foloseşte doar un singur registru index şi anume registrul XA, Fig. 5.8.

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

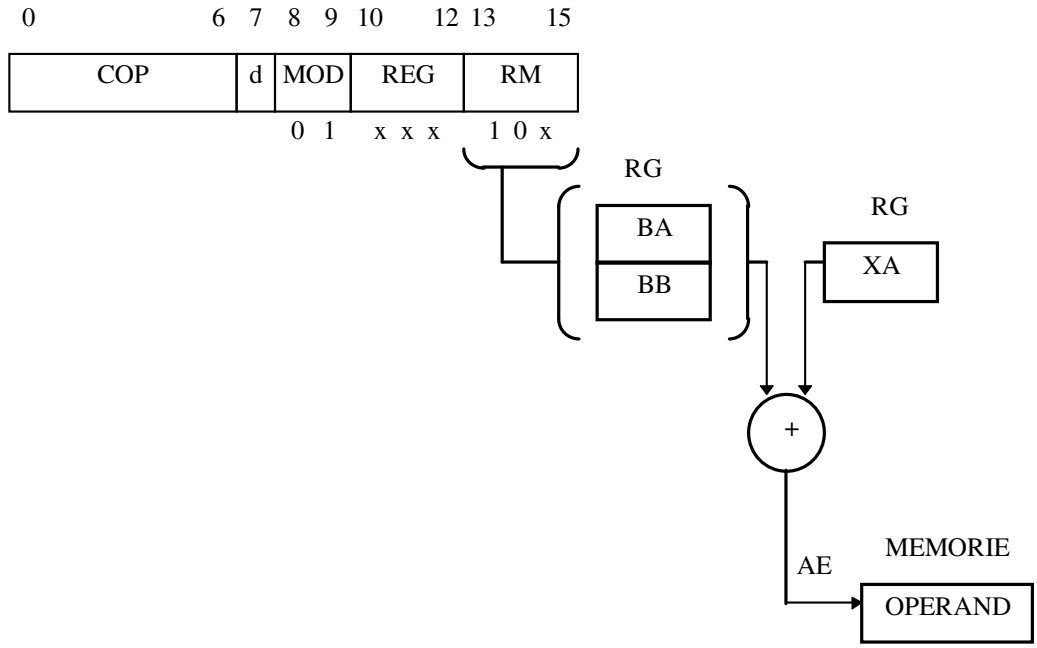
ceea ce este echivalent cu:

XA ← XA -1 şi

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

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

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

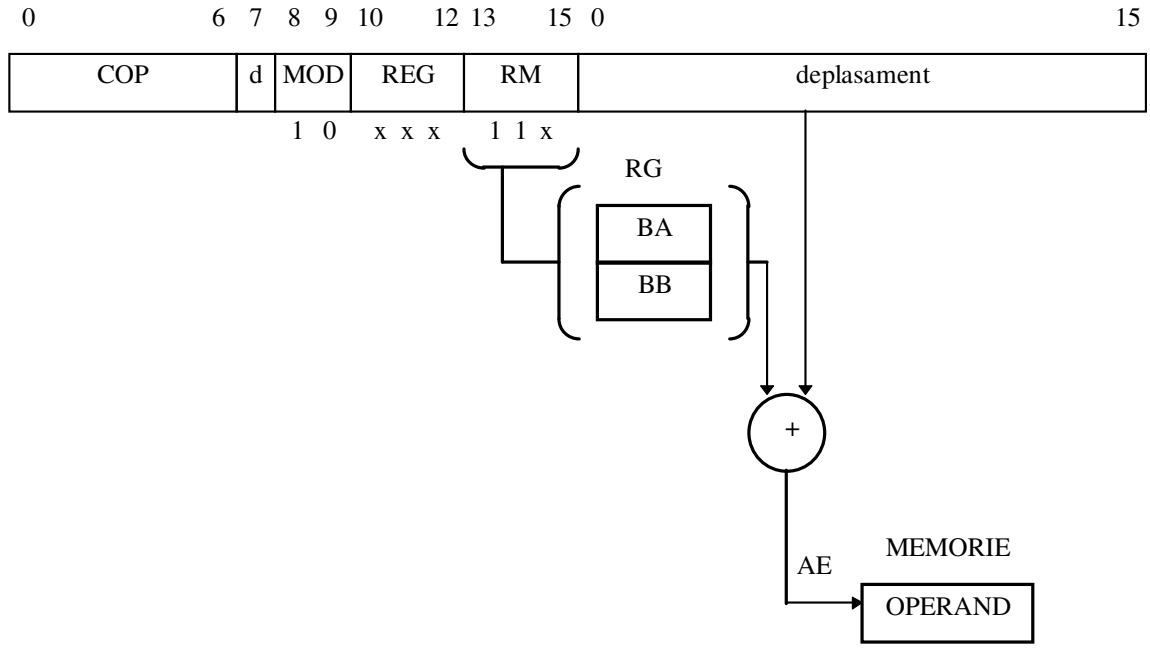


7. **Adresare bazată**

Adresa efectivă se obţine prin adunarea conţinutului celui de-al doilea cuvânt al instrucţiunii cu unul din registrele bază, Fig. 5.9.

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

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

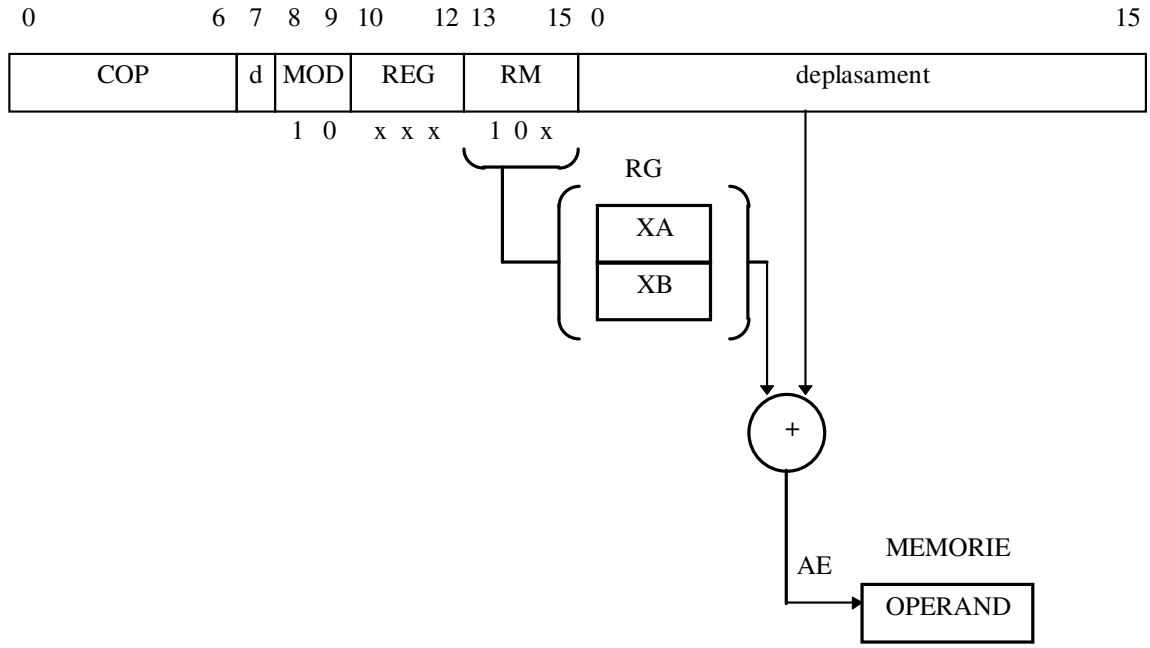


8. **Adresare indexată**

Adresa efectivă se obţine prin adunarea conţinutului celui de-al doilea cuvânt al instrucţiunii 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]
```

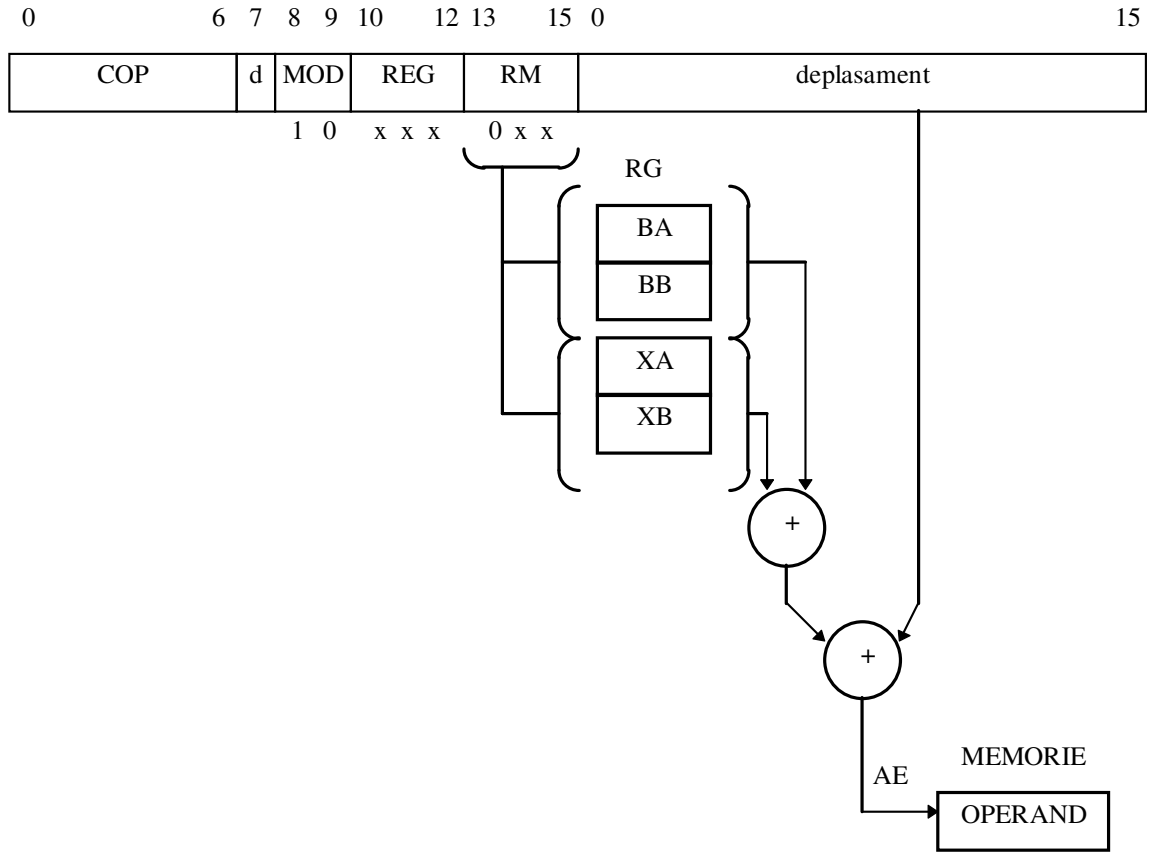


9. **Adresare bazată indexată**

Adresa efectivă se obţine prin adresarea conţinutului celui de-al doilea cuvânt al instrucţiunii cu suma dintre un registru de bază şi 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
```

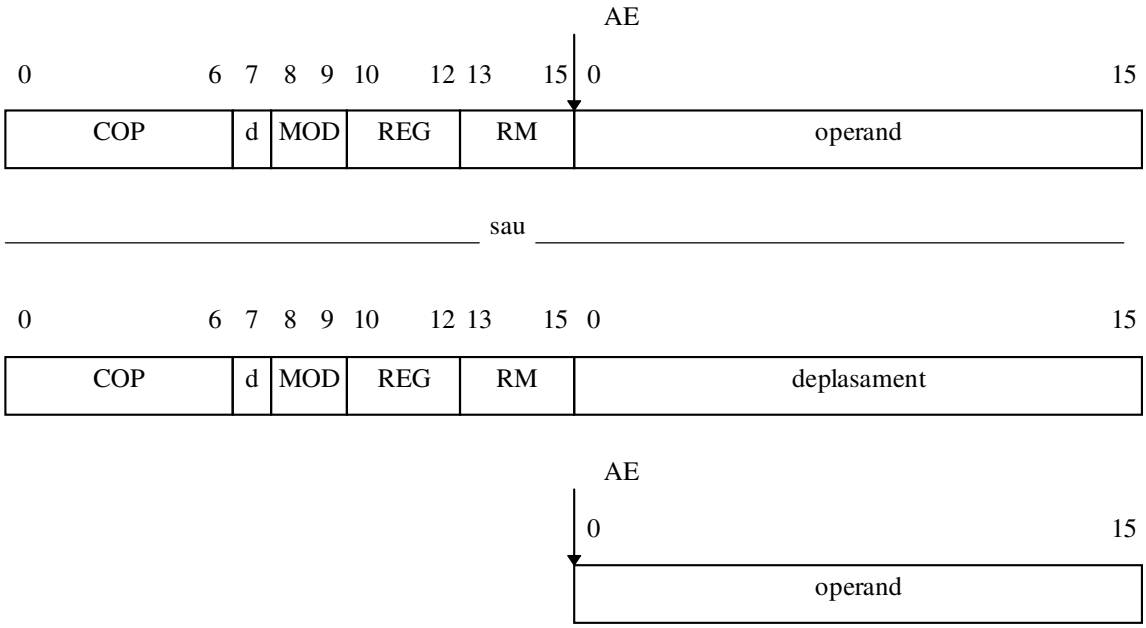


10. **Adresare imediată**

Operandul se găseşte în al doilea cuvânt al instrucţiunii sau al treilea cuvânt al instrucţiunii, Fig. 5.12.

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

Exemplu: mov RA,7

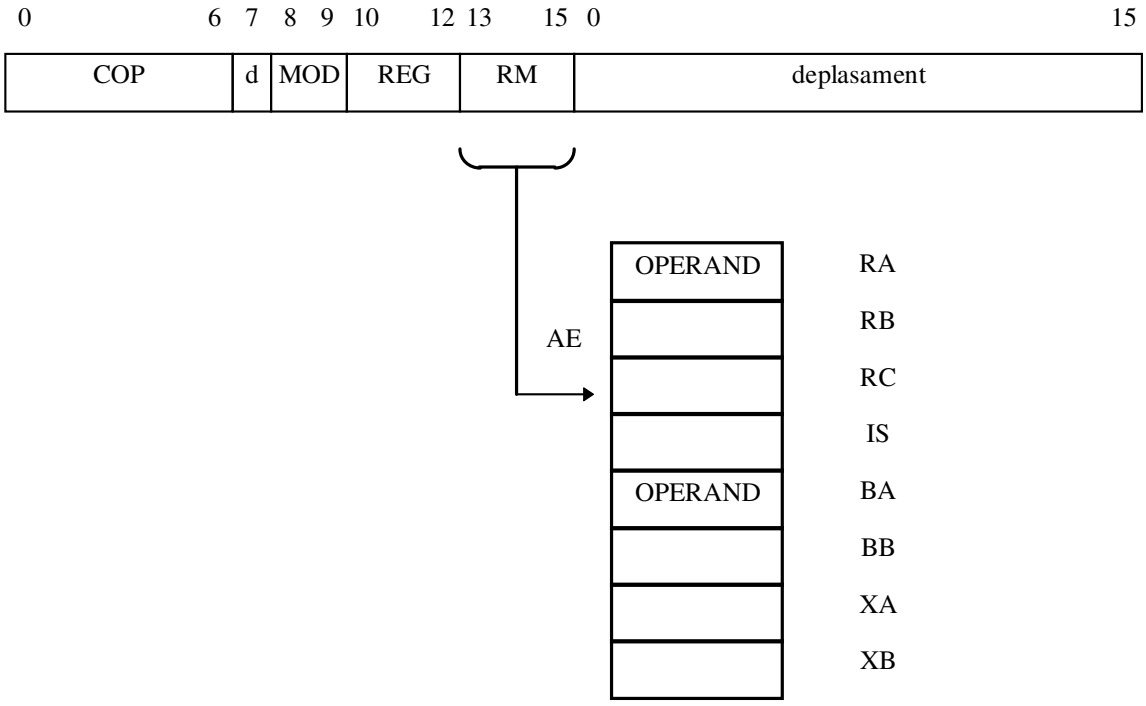


11. **Adresare directă la registru**

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

AE = [REG / RM]

Exemplu: mov RA,RB



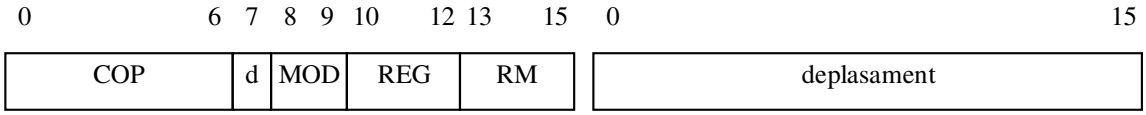
Clasificarea instrucţiunilor

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

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

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

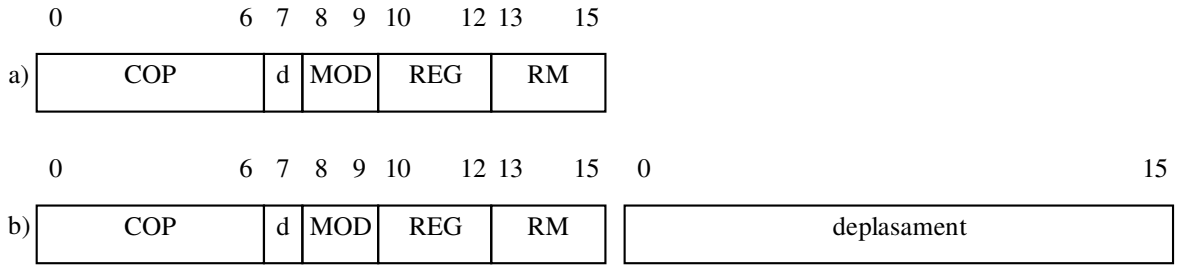
INC; DEC; NEG; NOT; SHL/SAL; SHR; SAR;
PUSH; POP; CALL; JMP



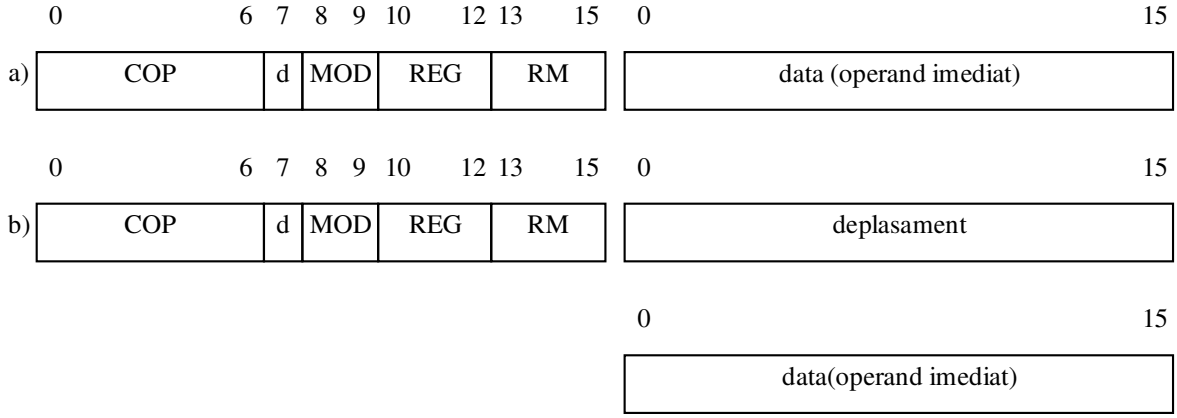
a2) - instrucţiuni **cu 2 adrese efective** (cu 2 operanzi)

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

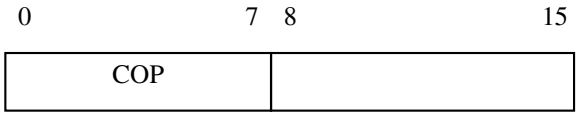
MOV; ADD; ADC; SUB; SBB; CMP; AND; OR; XOR; TEST



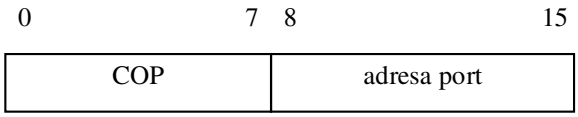
a2.2) - cu **2 operanzi dintre care unul este imediat,** Fig. 5.16 :
MOV; ADD; ADC; SUB; SBB; CMP; AND; OR; XOR; TEST



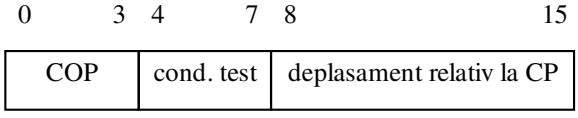
b) - instrucţiuni fără calcul de adresă efectivă, Fig. 5.17:
PUSHF; POPF; RET; IRET; HLT; IN; OUT; Jcond



a) PUSHF; POPF; RET; IRET; HLT



b) IN; OUT



b) J cond

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

	MOD(RI_8,RI_9)				
RM($RI_{13:15}$)	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 conţine chiar operandul

Tabelul 5.6 Calculul adresei efective

5.6 Codificarea instrucţiunilor

Ţinând seama de clasificarea instrucţiunilor maşină ale calculatorului didactic se propune ca instrucţiunile din aceeaşi categorie să fie codificate cu coduri adiacente pentru a simplifica faza de interpretare a acestora.

În codificarea instrucţiunilor maşină se va ţine cont de următoarele criterii generale :

Bitul RI_0 va separa grupul de instrucţiuni cu adresă efectivă de cele fără calcul de adresă efectivă.

Astfel:

RI_0=0 va caracteriza grupul de instrucţiuni cu calcul de adresă efectivă, iar

RI_0=1 va caracteriza grupul de instrucţiuni fără calcul de adresă efectivă .

În cadrul primului grup, bitul RI_1 va separa instrucţiunile cu o adresă de cele cu două adrese:

RI_1=0 caracterizează instrucţiunile cu o singură adresă

RI_1=1 caracterizează instrucţiunile cu două adrese

Bitul RI_2 din codul operaţiei separă grupul de instrucţiuni cu operand imediat de celelalte:

RI_2=0 caracterizează instrucţiunile fără operand imediat,

RI_2=1 caracterizează instrucţiunile cu operand imediat.

În cadrul grupului de instrucţiuni cu o singură adresă bitul RI_3 separă instrucţiunile de transfer date sau transfer control de celelalte.

Astfel:

RI_3=0 caracterizează instrucţiunile de transfer date/control, iar

RI_3=1 caracterizează instrucţiunile operaţionale cu un singur operand.

În cadrul grupului de instrucţiuni cu două adrese bitul RI_3=0 specifică instrucţiunile care nu produc rezultat ci poziţionează numai indicatorii de condiţii (CMP, TEST).

Biţii $RI_{4:6}$ codifică individual instrucţiunile din fiecare subgrup.

În cadrul clasei de instrucţiuni fără calcul de adresă efectivă bitul RI_3=1 separă instrucţiunile de salt condiţionat de celelalte. Biţii $RI_{4:6}$ codifică individual instrucţiunile iar în cazul instrucţiunilor de salt condiţionat $RI_{4:7}$ specifică condiţia de test.

Având în vedere aspectele prezentate privind codificarea instrucţiunilor maşină rezultă codificarea din Tabelul 5.11.

Tabelul 5.11 Codificarea instrucţiunilor

	RI_0=0 cu adresă efectivă		RI_0=1 fără adresă efectivă	
RI_1=0 cu o adresă	0=MOV	0=MOV	0=IN	0=
	1=	1=	1=OUT	1=
	2=PUSH	2=	2=PUSHF	2=
RI_3=0 transfer	3=POP	3=	3=POPF	3=
date/contr	4=CALL	4=	4=RET	4=
	5=JMP	5=	5=IRET	5=
	6=	6=	6=HLT	6=
	7=	7=	7=	7=
RI_3=1 operaţii	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=
	4=TEST	4=TEST	4=	4=
RI_3=0 nu produc	5=	5=	5=	5=
rezultat	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 instrucţiunea MOV a fost grupată
împreună cu cele cu o adresă PUSH, POP, CALL, JMP, considerându-se faptul că se citeşte un singur operand în faza de preluare operanzi.

Codurile operaţie specifice instrucţiunilor maşină, împreună cu bitul ce precizează destinaţia sunt prezentate în tabelul 5.12.

Tabelul 5.12 **Codurile instrucţiunilor maşină**

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 instrucţiunilor maşină

Faza de citire interpretare a instrucţiunilor maşină are ca obiective principale:
-citirea instrucţiunii curente;
-separarea instrucţiunilor în clase mari;
-calculul adresei efective dacă este cazul;
-pregatirea adresei efective în registrele AM şi T1.

Pentru a permite o implementare hardware simplă şi o descriere AHPL sub o formă cît 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.

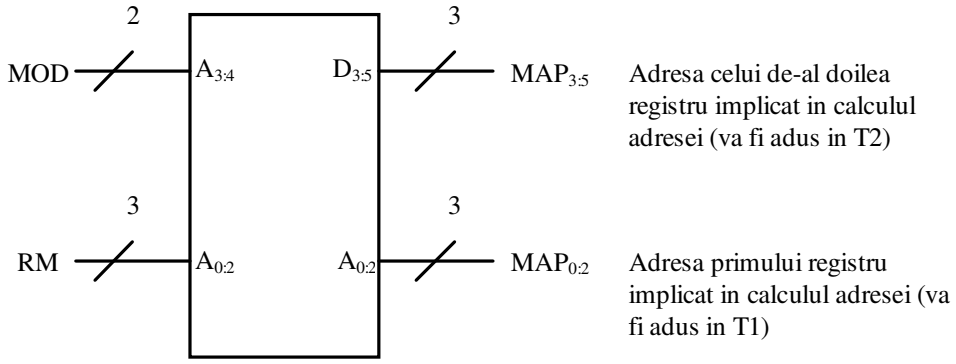


Figura 5.2

Această memorie va avea memorate adresele registrelor generale implicate, în funcţie 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

Conţinutul 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 instrucţiunilor maşină este realizată prin programul AHPL (pasii 1...25), conform cu organigrama din Fig. 5.3.

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

Tabelul 5.14 Conţinutul memoriei MAP

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 aşteaptă pornirea CD, ce face şi o iniţializare a contorului program CP.

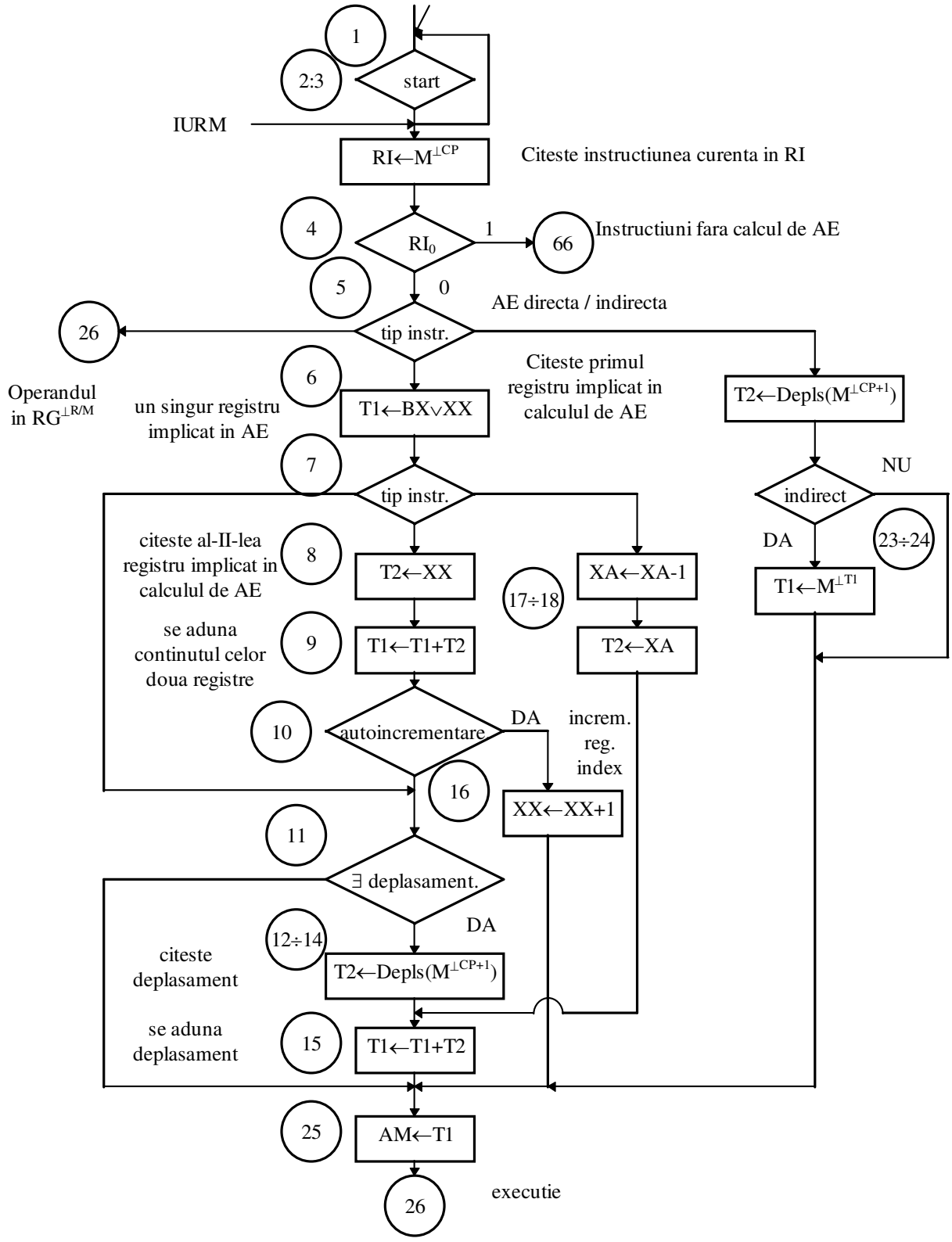
2. AM $\leftarrow$ CP
/* se aduce adresa instrucţiunii ce trebuie citită în registrul de adrese al memoriei.

3. RI $\rightarrow$ **BUSFN**(M;DCD(AM))
/* se citeşte în registrul de instrucţiuni, instrucţiunea ce se va executa.

4. $\rightarrow RI_0$ /(66)
/* se separă instrucţiunile fără calcul de adresă (pas 66) de cele cu calcul de adresa şi se continuă cu cele din urmă.
/* Se calculează adresa efectivă.

5. $\rightarrow ((RI_8 \wedge RI_9),(RI_8 \wedge \overline{RI_9} \wedge RI_{13} \wedge RI_{14}))$/(26,19)
/* se separă instrucţiunile pentru care MOD=11 (pas 26) şi 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 ainstructiunilor masina

Figura 5.3

/* Se citeşte în registrul temporar T1, conţinutul primului registru general care este utilizat la calculul adresei efective. Adresa registrului general este preluată din memoria MAP de la adresa specificată de MOD şi 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) şi cu autodecrementare (pas17) de celelalte moduri

8. T2 ← **BUSFN**(RG;DCD(**BUSFN**$_{3:5}$(MAP;DCD(MOD,RM))))
/* Pentru modurile de adresare rămase se citeşte în T2 cel de al doilea registru general ce participă la calculul adresei.

9. T1 ← 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. $\rightarrow \overline{RI_8}/(25)$
/* se separă modul de adresare cu deplasament de cel fără deplasament

12. CP ← INC(CP)
/* se pregăteşte adresa pentru citirea deplasamentului

13. AM ← CP

14. T2 ← **BUSFN**(M;DCD(AM))
/* se citeşte deplasamentul

15. T1 ← ADD(T1;T2;0) ;
$\rightarrow$ (25)
/* se adună deplasamentul la adresa efectivă şi se trece controlul la faza de execuţie

16. RG * DCD(**BUSFN**$_{3:5}$(MAP;DCD(MOD,RM))) ← ADD(0;T2;1) ;
$\rightarrow$ (25)
/* se incrementează registrul index utilizat în modul de adresare cu autoincrementare şi se trece la faza de execuţie

17. T2 ← **BUSFN**(RG;DCD(**BUSFN**$_{3:5}$(MAP;DCD(MOD,RM))))

18. T2 ← ADD(0FFFFH;T2;0);
RG * DCD(**BUSFN**$_{3:5}$(MAP;DCD(MOD,RM))) ← ADD(0FFFFH;T2;0) ;
$\rightarrow$ (15)

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

19. CP ← INC(CP)
/* începe tratarea modului de adresare direct/indirect
20. AM ← CP
21. T1 ← **BUSFN**(M;DCD(AM))
/* se citeşte deplasamentul

22. → $\overline{RI_{15}}$/(25)
/* se separă modul de adresare direct de indirect

23. AM ← T1

24. T1 ← **BUSFN**(M;DCD(AM))
/* se aduce în T1 adresa calculată indirect

25. AM ← T1
/* se aduce adresa efectivă în registrul AM

5.8 Faza de execuţie a instrucţiunilor maşină

Faza de execuţie a instrucţiunilor maşină este descrisă prin programul AHPL (pasii 26...88) conform cu organigramele din Fig.5.4, 5.5, 5.6.
/* Faza de execuţie a instrucţiunilor cu calcul de adresă efectivă
/* Se separa instrucţiunile cu doua adrese ADD,ADC,etc de cele cu o adresă (cu un operand) INC, DEC, etc, formîndu-se un grup separat cu cele de transfer date şi chemare de subrutină sau salt necondiţionat deoarece aceste trei mari clase au părţi comune în faza de execuţie.

26. → (RI_1)/(57)
/* se separă instrucţiunile cu două adrese de celelalte

27. → (RI_3)/(54)
/* se separă instrucţiunile cu o adresă de cele de transfer date şi ramificaţie.
/* Execuţia instrucţiunilor MOV; CALL; JMP; PUSH; POP (organigrama din Fig. 5.4)

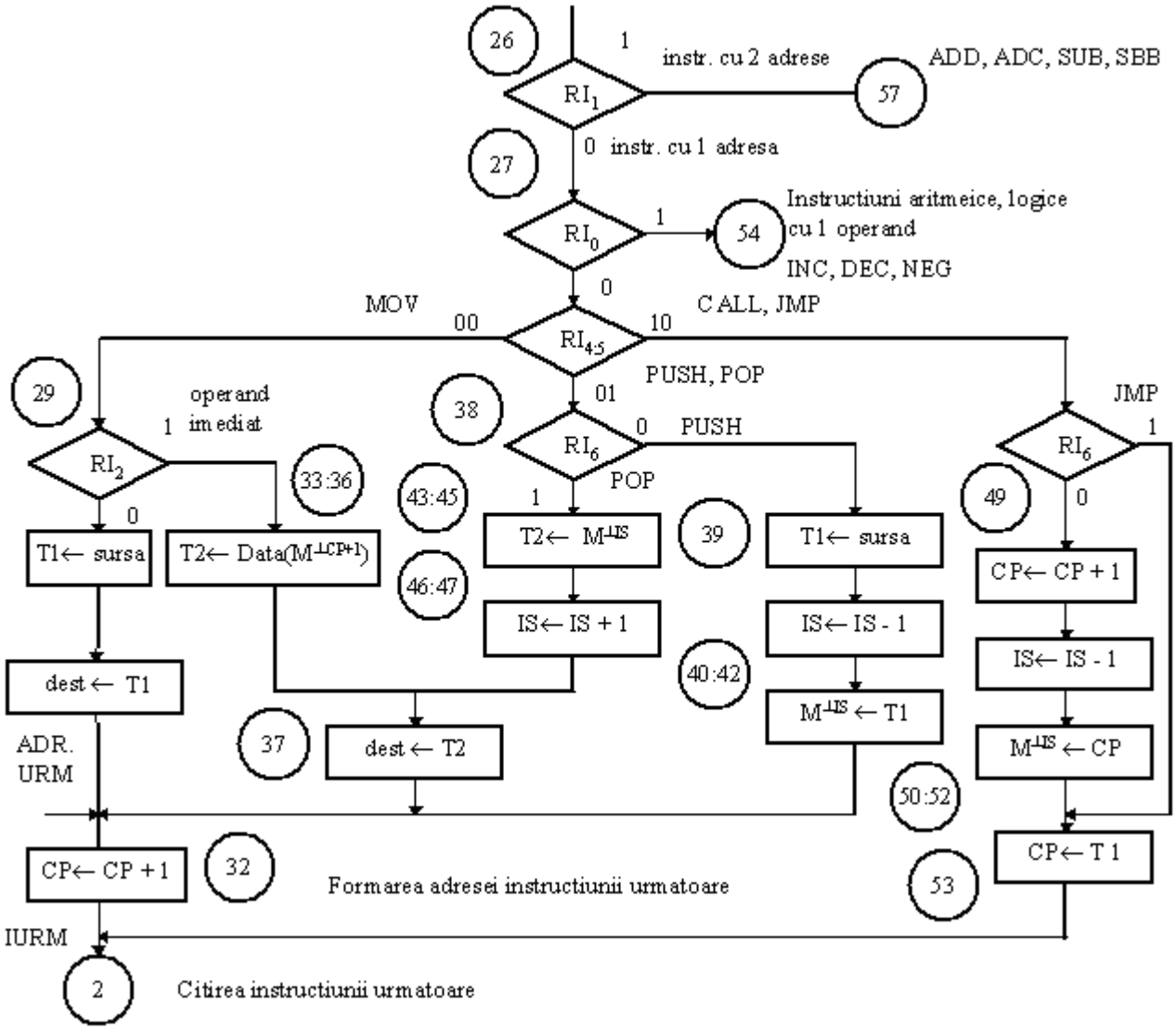


Figura 5.4

28. → (RI_4,RI_5)/(48,38)
/* se separă instrucţiunile CALL;JMP (pas48) şi PUSH;POP (pas 38) de instrucţiunile MOV care se tratează în continuare

/* Execuţia instrucţiunilor MOV

29. → (RI_2)/(33)
/* se tratează separat instrucţiunea MOV cu operand imediat de cea cu operand de la adresă efectivă;
ţinând seama de sensul bitului d=RI_7 şi modul de adresare prezentate în formatul instrucţiunilor, operandul sursă al instrucţiunii MOV poate fi într-un registru general specificat de biţii REG sau RM sau într-o celulă de memorie specificată de adresa efectivă.

dacă d=0 sursa este în RG specificată de REG iar destinaţia 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 destinaţia în RG specificată de REG

30. T1 ← (**BUSFN**(RG;DCD(REG))!**BUSFN**(M;DCD(AM))!**BUSFN**(RG;DCD(RM)))

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

31.(((RG*DCD(REG))!(M*DCD(AM))!(RG*DCD(RM))

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

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

32. CP ← INC(CP)
→ (2)

/* s-a terminat de executat instrucţiunea MOV cu operandul de la adresa efectivă. Se pregăteşte adresa pentru instrucţiunea următoare şi se reia faza de citire interpretare.

/* Execuţia instrucţiunii MOV cu operand imediat

33. CP ← INC(CP)

/* se stabileşte adresa pentru operandul imediat şi se aduce în registrul de adrese

34. AM ← CP

35. T2 ← **BUSFN**(M;DCD(AM))

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

36. AM ← T1

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

37. ((M*DCD(AM))!(RG*DCD(RM))*($\overline{RI_8 \wedge RI_9}$,$RI_8 \wedge RI_9$)) ← T2 ;
→ (32)

/* se depune operandul sursă la adresa efectivă şi se trece la formarea adresei pentru instrucţiunea următoare.

/* Execuţia instrucţiunilor PUSH şi POP

38. → (RI_6)/(43)

/* se separă instrucţiunea POP. Se citeşte în T1 operandul ce trebuie dus în stivă

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

40. T2 ← **BUSFN**(RG;DCD(ADRIS))

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

41.RG*DCD(ADRIS) ← ADD(0FFFFH;T2;0)

AM ← ADD(0FFFFH;T2;0)

42. M*DCD(AM) ← T1
→ (32)
/* se salvează operandul sursă în stivă şi se merge la formarea adresei instrucţiunii următoare
/* se execută POP

43. AM ← **BUSFN**(RG;DCD(ADRIS))

44. T2 ← **BUSFN**(M;DCD(AM))
/* se citeşte 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)

/* **Execuţia instrucţiunilor CALL; JMP**

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

49. CP ← INC(CP)
/* se formează adresa instrucţiunii următoare şi 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
/* Execuţia instrucţiunilor cu un operand
INC; DEC; NEG; NOT; SHL/SAL; SHR; SAR
(organigrama din Fig.5.5)

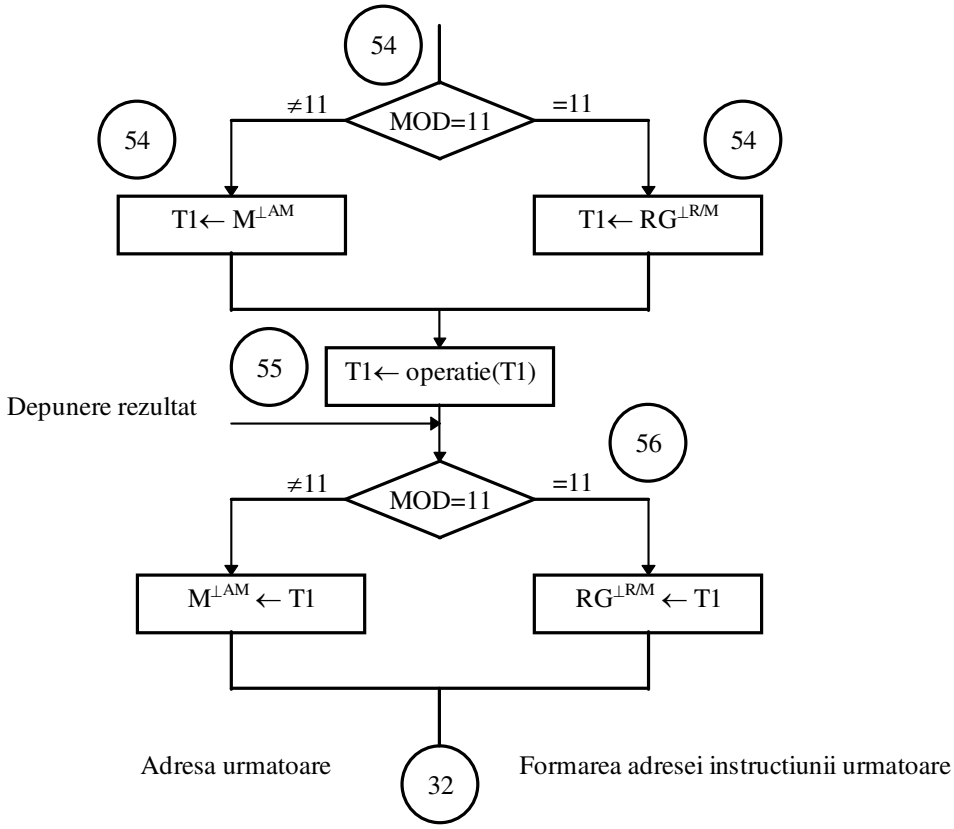


Figura 5.5

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

/* s-a citit operandul în registrul temporar T1

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

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

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

$\rightarrow$ (32)

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

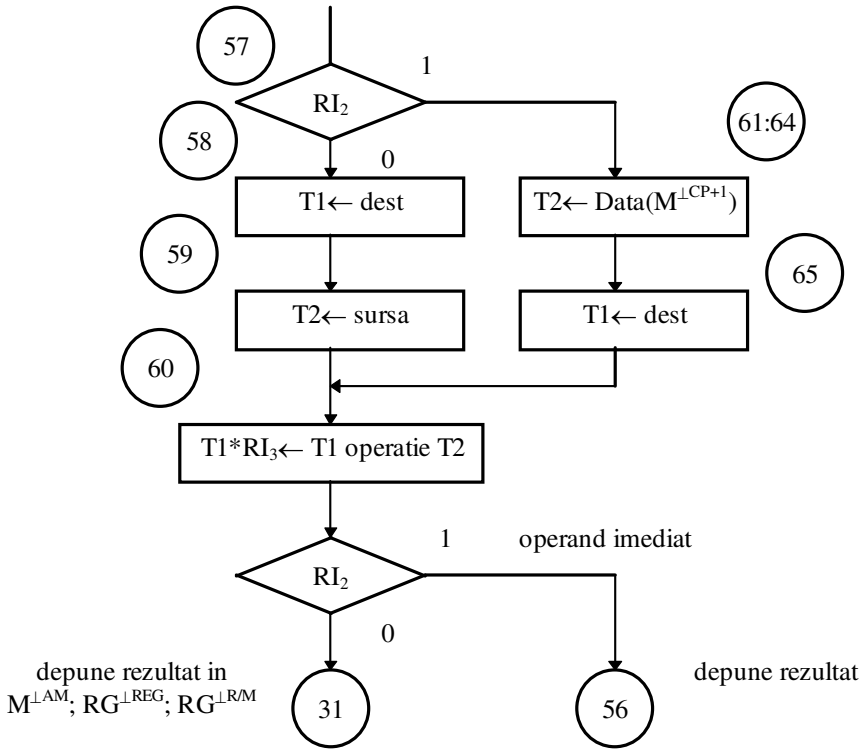


Figura 5.6

57. → (RI_2)/(61)
/* se separă instrucţiunile cu operand imediat
58. T1 ← (**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 citeşte în T1 operandul 1

59. T2 ← (**BUSFN**(RG;DCD(REG))!**BUSFN**(M;DCD(AM))!**BUSFN**(RG:DCD(RM)))

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

/* se citeşte în T2 operandul 2
/* RI_3=0 caracterizează instrucţiunile ce nu produc rezultat şi pozitionează indicatorii de stare (CMP;TEST)

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

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

$\rightarrow (\overline{RI_2},RI_2)/(31,56)$
/* execuţia instrucţiunilor cu doi operanzi, dintre care unul este imediat

61. CP ← INC(CP)
/* se citeşte operandul imediat

62. AM ← CP

63. T2 ← **BUSFN**(M;DCD(AM))
/* se reface adresa efectivă

64. AM ← T1

65. T1 ← (**BUSFN**(M;DCD(AM))!**BUSFN**(RG;DCD(RM))*$(\overline{RI_8 \wedge RI_9},RI_8 \wedge RI_9)$
$\rightarrow$ (60)

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

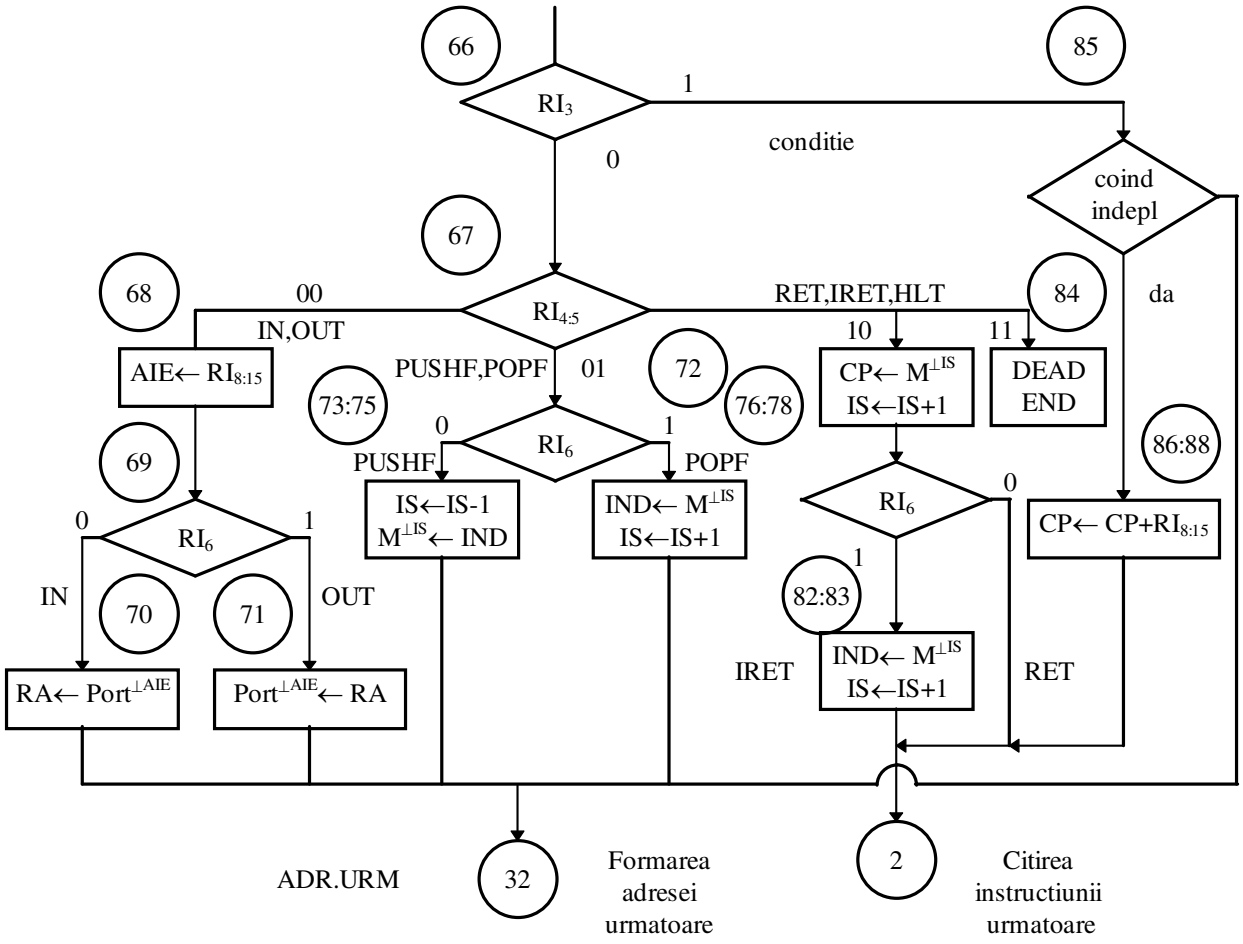


Figura 5.7

66. → RI_3/(85)
/* se separă instrucţiunile de salt condiţionat

67. → ($\overline{RI_4} \wedge RI_5$,$RI_4 \wedge \overline{RI_5}$,$RI_4 \wedge RI_5$)/(72,79,84
)
/* se separă instrucţiunile PUSHF; POPF (pas 72), instrucţiunile RET; IRET (pas 79) şi instrucţiunea HLT (pas(84)
/* Execuţia instrucţiunilor IN, OUT

68. AIE ← $RI_{8:15}$

69. → RI_6/(71)
/* se separă instrucţiunea OUT

70. RG*DCD(ADRRA)) ← **BUSFN**(PORT;DCD(AIE))
→ (32)
/* se execută instrucţiunea IN

71. PORT * DCD(AIE) → **BUSFN**(RG;DCD(ADRRA))
→ (32)
/* se execută instrucţiunea OUT

/* Execuţia instrucţiunilor PUSHF; POPF

72. → (RI_6)/(76)
/* se separă POPF
/* în continuare se execută PUSHF
7
3. T1 ← **BUSFN**(RG;DCD(ADRIS))

74. RG*DCD(ADRIS) ← ADD(T1;OFFFFH;0);
AM ← ADD(T1;0FFFFH;0)

75. M*DCD(AM) ← IND
→ (32)
/* s-a executat instrucţiunea PUSHF

76. AM ← **BUSFN**(RG;DCD(ADRIS))
T1 ← **BUSFN**(RG;DCD(ADRIS))

77. IND ← **BUSFN**(M;DCD(AM))
78. RG*DCD(ADRIS) ← ADD(T1;0;1)
→ (32)
/* s-a executat instrucţiunea POPF

/* **Execuţia instrucţiunilor RET;IRET**

79. AM ← **BUSFN**(RG;DCD(ADRIS))
T1 ← **BUSFN**(RG;DCD(ADRIS))

80. CP ← **BUSFN**(M;DCD(AM))
/* se reface contorul program

81. RG*DCD(ADRIS) ← ADD(T1;0;1)
AM ← ADD(T1;0;1)
T1 ← ADD(T1;0;1)

→ $\overline{RI_6}$/(2)

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

82. IND ← **BUSFN**(M;DCD(AM))

```
/* se execută IRET

83. RG*DCD(ADRIS) ← ADD(T1;0;1)
      → (2)

/*                Execuţia instrucţiunii HLT
84. DEAD END

/*          Execuţia instrucţiunilor de salt condiţionat

85.→ condiţie / (32)

/* Condiţie neândeplinită, salt la instrucţiunea următoare
86. T1 ← CP

87. T2 ← RI8:15
/* transferul se face cu extensia semnului RI8 pe biţii 0:7)

88. CP ← ADD(T1;T2;0)
      → (2)
```

5.9 Completarea setului de instrucţiuni

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

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

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

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

- alegerea funcţiei şi a mnemonicei;
- stabilirea formatului instrucţiunii;
- codificarea instrucţiunii;
- modificarea fazei de citire interpretare în scopul recunoaşterii instrucţiunii;
- descrierea fazei de execuţie a instrucţiunii.

În vederea prezentării mecanismului de completare a setului instrucţiuni vom introduce instrucţiuni ce realizează operaţiile:

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

Alegerea funcţiei şi a mnemonicei

Pentru realizarea operaţiilor propuse alegem :

Mnemonică	Funcţie
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 secţiuni de program de un număr de LOOPZ ori
XCHG	-interschimbă conţinutul a doi operanzi

5.9.1 Introducerea instrucţiunilor RCL, RCR, ROL, ROR

5.9.1.1 Formatul şi funcţiile instrucţiunilor RCL, RCR, ROL, ROR

Formatul instrucţiunilor RCL, RCR, ROL, ROR este :

COP c MOD --- RM

Instrucţiunea RCL realizează funcţia :

dacă c=0 atunci CONTOR = 1
altfel CONTOR = (RC)
atât timp cât CONTOR ≠ 0 execută T,$OPERAND_{0:15}$ ← $OPERAND_{0:15}$,$OPERAND_0$
CONTOR = CONTOR - 1

Instrucţiunea RCR realizează funcţia :

dacă c=0 atunci CONTOR = 1
altfel CONTOR = (RC)
atât timp cât CONTOR ≠ 0 execută
T,$OPERAND_{0:15}$ ← $OPERAND_{15}$,$OPERAND_{15}$,$OPERAND_{0:14}$
CONTOR = CONTOR - 1

Instrucţiunea ROL realizează funcţia :

dacă c=0 atunci CONTOR = 1
altfel CONTOR = (RC)
atât timp cât CONTOR ≠ 0 execută T,$OPERAND_{0:15}$ ← $OPERAND_{0:15}$,T
CONTOR = CONTOR - 1

Instrucţiunea ROR realizează funcţia :

dacă c=0 atunci CONTOR = 1
altfel CONTOR = (RC)
atât timp cât CONTOR ≠ 0 execută T,$OPERAND_{0:15}$ ← $OPERAND_{15}$,T,$OPERAND_{0:14}$
CONTOR = CONTOR - 1

5.9.1.2 Codificarea instrucţiunilor RCL, RCR, ROL, ROR

Analizând codificarea instrucţiunilor maşină ale calculatorului didactic se observă că aceste instrucţiuni fac parte din clasa de instrucţiuni cu un singur operand.

Codul 0001111 din această categorie de instrucţiuni nu este utilizat. Vom folosi acest cod pentru codificarea acestui grup de instrucţiuni iar biţii $RI_{10:12}$ =REG, de asemenea neutilizaţi 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 recunoaşterii acestor instrucţiuni

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

54.1 $\rightarrow DCD_7(RI_{4:6})$ / (90)

Se consideră că execuţia acestor instrucţiuni începe la pasul AHPL 90.

5.9.1.4 Descrierea fazei de execuţie

90. T2 ← (**BUSFN**(RG;DCD(ADRRC))!16T1)*(RI_7,RI_7)

91. T,T1 ← (($T1_0$,$T1_{1:15}$,$T1_0$)!($T1_{15}$,$T1_{15}$,$T1_{0:14}$)!($T1_0$,$T1_{1:15}$,T)!
($T1_{15}$,T,$T1_{0:14}$))*$DCD_{0:3}$($RI_{11:12}$)

92. T2 ← ADD(0FFFFH;T2;0);
→ ((∨/T2),(∨/T2))/(91,56)

5.9.2 Introducerea instrucţiunilor LOOP, LOOPZ

Instrucţiunile LOOP şi LOOPZ sunt utilizate pentru a realiza repetarea unei secţiuni de program de un număr prestabilit de ori, altfel spus realizarea ciclurilor în program.

5.9.2.1 Formatul şi funcţiile instrucţiunilor LOOP şi LOOPZ.

Instrucţiunile LOOP şi LOOPZ realizează un salt relativ la CP stabilit de valoarea deplasamentului din cadrul instrucţiunii.

Aceste instrucţiuni se pot introduce în grupul instrucţiunilor de salt condiţionat. Fac parte din categoria instrucţiunilor fără calcul de adresă efectivă.

Formatul general este :
COP Deplasament

Instrucţiunea LOOP realizează funcţia:

RC ← RC - 1

dacă RC ≠ 0 atunci CP ← CP + Deplasament
altfel CP ← CP + 1

registrul RC este decrementat şi se relizează saltul relativ la CP cu o valoare specificată de Deplasament dacă conţinutul registrului RC este diferit de zero, iar în caz contrar se continuă cu instrucţiunea următoare.

Deplasamentul trebuie să aibă o valoare între -128 şi 127 şi reprezintă o adresă relativă faţă de adresa instrucţiunii curente LOOP.

Instrucţiunea LOOPZ realizează funcţia:

RC ← RC - 1
dacă (RC ≠ 0) ∧ (Z = 1) atunci CP ← CP + Deplasament
altfel CP ← CP + 1

registrul RC se decrementează şi se realizează saltul relativ la CP cu o valoare specificată de Deplasament dacă conţinutul registrului RC este diferit de zero şi indicatorul Z este unu, iar în caz contrar (RC = 0 sau Z = 0) se continuă cu intrucţiunea următoare. Deplasamentul trebuie să fie aibă o valoare între -128 şi 127 şi reprezintă o adresă relativă faţă de adresa instrucţiunii curente LOOPZ.

5.9.2.2 Codificarea instrucţiunilor LOOP şi LOOPZ

Deoarece aceste instrucţiuni fac parte din grupul instrucţiunilor de salt condiţionat vom stabili o codificare adiacentă cu a acestora. Propunem codificarea:

1 0 1 1 0 x x x Deplasament pentru instrucţiunea LOOP
1 0 1 1 1 x x x Deplasament pentru instrucţiunea LOOPZ

5.9.2.3 Modificarea fazei de citire interpretare în scopul recunoaşterii acestor instrucţiuni

Pentru a lua în considerare şi acest grup de instrucţiuni pasul AHPL 85, care realiza execuţia instrucţiunilor de salt condiţionat, se modifică astfel:

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

S-a ţinut seama că cele două grupuri de instrucţiuni diferă numai prin bitul RI_2 din codul de operaţie. La pasul 32 se merge în cazul instrucţiunilor de salt condiţionat când condiţia nu este îndeplinită. Pasul 86 constituie continuarea execuţiei instrucţiunilor de salt condiţionat când condiţia este îndeplinită, iar pasul 93 reprezintă începutul secvenţei de execuţie a instrucţiunilor LOOP şi LOOPZ.

5.9.2.4 Descrierea fazei de execuţie

93. T2 ← **BUSFN**(RG;DCD(ADRRC))

94. RG * DCD(ADRRC) ← ADD(0FFFFH;T2;0)

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

96. T1 ← CP

97. T2 ← $RI_{8:15}$

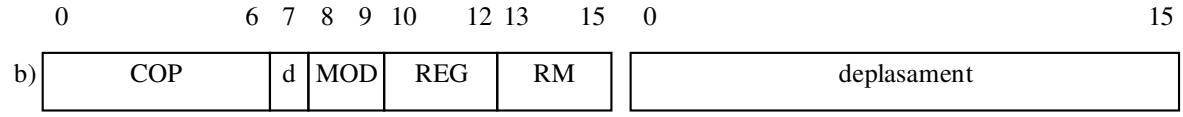
98. CP ← ADD (T1;T2;0)
→ (93)

5.9.3 Introducerea instrucţiunii XCHG

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

5.9.3.1 Formatul şi funcţia instrucţiunii XCHG.

Instrucţiunea XCHG face parte din grupul instrucţiunilor cu două adrese. Formatul general al instrucţiunii este:



Funcţia îndeplinită de instrucţiunea XCHG este:
dacă MOD = 11 atunci $RG^{DCD(REG)}$ <--> $RG^{DCD(RM)}$
altfel $RG^{DCD(REG)}$ <--> $M^{DCD(AM)}$

5.9.3.2 Codificarea instrucţiunii XCHG

Făcând parte din grupul instrucţiunilor 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 recunoaşterii acestei instrucţiuni

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

57.1 → (DCD($RI_{4:6}$))/(99)
S-a considerat ca la pasul AHPL 99 va începe execuţia instrucţiunii XCHG.

5.9.3.4 **Descrierea fazei de execuţie**

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

100.T2 ← **BUSFN**(RG;DCD(REG))

101.RG * DCD(REG) ← T1

102.((M*DCD(AM))!(RG*DCD(RM))) * $\overline{(RI_8 \wedge RI_9, RI_8 \wedge RI_9)}$ ← T2
→ (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 pregăteşte adresa pentru instrucţiunea următoare.

În acest mod se pot introduce orice instrucţiune care este absolut necesară în setul de instrucţiuni maşină.