CALCULATORUL DIDACTIC CD

1. Scopul lucrării

În lucrare se prezintă structura unui calculator numeric simplu, setul de instrucțiuni şi semnalele de comandă ale acestuia. Se urmăreşte scrierea unor programe simple şi vizualizarea execuției acestora cu ajutorul unui program de simulare.

2. Considerații teoretice

2.1. Prezentarea generală a calculatorului didactic

Calculatorul didactic CD este un calculator numeric simplu, cu funcționare sincronă, cu o unitate aritmetică binară de tip paralel. Cuvântul are lungimea fixă de 24 de biți. Capacitatea memoriei este de 64 K cuvinte, deci adresa unui cuvânt are dimensiunea de 16 biți. Se utilizează instrucțiuni cu o singură adresă. Modurile de adresare posibile sunt: adresarea directă, adresarea indirectă, adresarea imediată şi adresarea indexată.

Principalele operații care pot fi realizate de acest calculator sunt următoarele:

- Operații aritmetice: adunare, scădere, incrementare, decrementare;
- Operații logice: ŞI logic, SAU logic, complement logic;
- Operații de rotire la dreapta şi la stânga;
- Operații cu stiva;
- Operații cu subrutine;
- Salturi condiționate şi necondiționate.

Elementele principale ale calculatorului didactic sunt următoarele:

- Memoria internă M [65536, 24], căreia i se ataşează decodificatorul de adrese DA şi registrul de adrese RA [16];
- Unitatea aritmetică şi logică UAL;
- Numărătorul de adrese NA [16];
- Registrul de instrucțiuni RI [24];
- Registrul de date al memoriei RT [24], numit şi registru tampon, având şi rolul de a păstra unul din operanzi;
- Registrul acumulator A [24];
- Registrul de index X [16];
- Registrul indicator de stivă SP [16], stiva fiind organizată în memoria internă;
- Indicatorii de condiții Z (*Zero*) şi C (*Carry*);
- Unitatea de comandă, cu un numărător de secvențiere NS [4] şi un decodificator al codului operației DCO.

Se utilizează două magistrale cu 24 de linii, ABUS şi BBUS, conectate la intrările UAL, şi o altă magistrală RBUS cu 25 de linii, conectată la intrările registrelor. Linia $RBUS_{24}$ a magistralei RBUS este conectată la indicatorul *Carry*.

2.2. Instrucțiunile calculatorului didactic

2.2.1. Limbajul de descriere utilizat

Pentru descrierea structurii calculatorului şi a secvențelor de operații elementare necesare execuției instrucțiunilor se utilizează un limbaj de descriere simplu, ale cărui elemente principale sunt prezentate în continuare.

Constantele numerice utilizate sunt valorile binare `0`, `1` şi codurile binare de `n` biți ale unor numere zecimale `d`, de forma `n$d`. Caracterul `$` indică deci operația de codificare binară. De exemplu:

```
4$0 = 0000
4$6 = 0110
```

Variabilele pot fi scalare, vectoriale sau matriciale. Acestea pot desemna numele următoarelor elemente hardware:

- Semnale de comandă sau de stare: `read`, `write`;
- Bistabile: `C`, `Z`, `BS`;
- Registre sau numărătoare: `A[24]`, `X[16]`, `N[4]`;
- Memorii sau tablouri de registre: `M[65536,24]`, `ROM[4096,16]`, `R[16,32]`;
- Magistrale de comunicație: `ABUS`, `BBUS`, `RBUS`.

Operatorii de selecție permit selecția unor elemente ale variabilelor vectoriale sau matriciale: bistabile ale unui registru, biți ai unor cuvinte de memorie, registre ale unui tablou de registre sau cuvinte de memorie. Exemple:

- Pentru registre: A_i reprezintă bitul `i` din registrul `A`;
- Pentru tablouri de registre: R_m reprezintă registrul `m` din tabloul de registre `R`;
- Pentru memorii: M_m reprezintă cuvântul `m` din memoria `M`.

Este posibilă selecția mai multor elemente dintr-un vector, a mai multor registre dintr-un tablou de registre sau a mai multor cuvinte de memorie. Exemple:

- Pentru registre: $A_{i:k}$ reprezintă biții `i` până la `k` din registrul `A`;
- Pentru tablouri de registre: $R_{i:k}$ reprezintă registrele `i` până la `k` din tabloul de registre `R`;
- Pentru memorii: $M_{i:k}$ reprezintă cuvintele `i` până la `k` din memoria `M`.

În cazul memoriilor, este necesară de multe ori selecția unui cuvânt în funcție de conținutul unui registru. Acest registru conține adresa cuvântului. De exemplu,

```
M[RA]
```

reprezintă cuvântul de memorie a cărei adresă se află în registrul `RA`. Notația semnifică decodificarea conținutului registrului `RA` şi selecția cuvântului de memorie.

Operatorul & realizează concatenarea unor elemente scalare, vectoriale sau matriciale, pentru a obține variabile cu mai multe componente. De exemplu, dacă `A` este un registru cu 4 poziții, iar `S` este un bistabil, cu următorul conținut:

```
A = 1001
S = 0
```

atunci prin aplicarea operatorului de concatenare se pot obține, de exemplu, următoarele variabile:

$A_1\&A_0 = 01$
$S\&A_{2:0} = 0001$

Operatorii logici sunt următorii:

- $\overline{\ }$ NU logic;
- + SAU logic;
- * ŞI logic;

- ⊕ SAU EXCLUSIV.

Exemple:

```
A = 1011
B = 0101
_
A  = 0100
A+B = 1111
A∗B = 0001
A⊕B = 1110
```

Cu excepția operatorului NU logic, ceilalți operatori logici se pot aplica tuturor elementelor unui vector sau a unei matrici, simbolul operatorului fiind urmat de numele vectorului sau al matricii în paranteze. Rezultă o valoare binară sau un vector coloană. Exemple:

```
A = 1011
+(A) = (((1+0)+1)+1) = 1
∗(A) = (((1∗0)∗1)∗1) = 0
⊕(A) = (((1⊕0)⊕1)⊕1) = 1
```

Operatorii aritmetici permit adunarea, scăderea, incrementarea sau decrementarea variabilelor binare fără semn. Operatorii aritmetici de adunare şi scădere sunt operatori binari, iar cei de incrementare şi decrementare sunt operatori unari. Operatorii aritmetici sunt următorii:

- `add` Adunare;
- `sub` Scădere;
- `inc` Incrementare;
- `dec` Decrementare.

Exemple:

```
A add X
A sub B
dec N
```

Expresiile descriu funcția executată de o microoperație înaintea executării unui transfer. Expresiile sunt formate din constante, variabile, operatori şi, eventual, paranteze. Relația de precedență a operatorilor în cadrul expresiilor este următoarea:

1. Paranteze;
2. Operatori de selecție;
3. `inc`, `dec`;
4. `add`, `sub`;
5. NU logic;
6. ŞI logic;
7. SAU logic, SAU EXCLUSIV;
8. Concatenarea.

Microinstrucțiunile se utilizează pentru a descrie o secvență de comandă. Fiecare microinstrucțiune descrie o operație elementară, numită şi microoperație, care se execută pe durata unui impuls al generatorului de tact. O microoperație descrie operanzii şi operatorul printr-o expresie, transferul efectuat printr-o săgeată şi destinația transferului prin variabila în care se memorează rezultatul.

Microoperațiile se pot clasifica în următoarele categorii:

- De transfer propriu-zis, prin care informația nu este modificată în timpul transferului;
- Aritmetice;
- Logice;
- De deplasare în registre.

De exemplu, transferul registrului `B` în registrul `A` se indică prin următoarea microoperație:

```
A ← B;
```

Semnalul de comandă sau funcţia de comandă care validează o microoperaţie se indică între bare oblice, de exemplu:

```
/T0/   A ← A add B;
```

Transferul datelor prin intermediul magistralelor se execută în două etape. În prima etapă, datele sunt depuse pe magistrală. Această operaţie se deosebeşte de un transfer obişnuit prin faptul că datele nu sunt memorate pe magistrală; ele rămân pe magistrală cât timp semnalul de comandă care validează această operaţie este activ. Transferul datelor pe magistrală este indicat prin semnul "=":

```
/c1/   ABUS = A;
```

În a doua etapă, datele sunt transferate de pe magistrală în registrul destinaţie şi sunt memorate în acest registru. Această operaţie nu diferă de un transfer obişnuit între registre, de exemplu:

```
/c2/   RI ← RBUS;
```

Ambele etape ale transferului cu magistrala se pot executa pe durata aceleiaşi perioade a semnalului de tact. Astfel, cele două etape ale transferului precedent, executate pe durata unui impuls T_0 al semnalului de tact, se pot scrie astfel:

```
/T0*c1/ ABUS = A,
/T0*c2/ RI ← RBUS;
```

Dacă o microoperaţie trebuie executată doar în anumite condiţii, ea poate fi descrisă printr-o *microinstrucţiune condiţională*. O asemenea microinstrucţiune poate avea una din formele următoare:

```
if (e) then (m1);
if (e) then (m1) else (m2);
```

unde *e* este o expresie condiţională. Dacă condiţia este adevărată, microinstrucţiunea m_1 este executată. În caz contrar, se execută microinstrucţiunea m_2, dacă aceasta este specificată după cuvântul cheie `else`. Se pot utiliza operatorii logici pentru a forma o expresie condiţională. Dacă în cadrul expresiei nu apare simbolul "=" urmat de o valoare, expresia se evaluează ca fiind adevărată dacă valoarea logică a acesteia este 1. De exemplu, microinstrucţiunea condiţională:

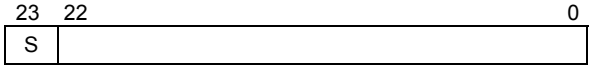
```
if (S) then (Q0 ← 0) else (Q0 ← 1);
```

va poziţiona bistabilul `Q0` din registrul `Q` la 0 sau la 1, după cum bistabilul `S` este poziţionat la 1, respectiv la 0.

2.2.2. Structura cuvântului

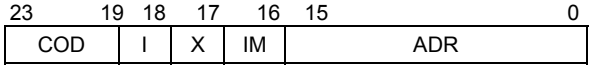
Cuvântul calculatorului reprezintă unitatea de informaţie prelucrată. Pentru simplitate, lungimea cuvântului este fixă, fiind egală cu 24 de biţi (3 octeţi). Cuvântul este interpretat ca o dată sau ca o instrucţiune.

Unitatea centrală tratează cuvântul ca o dată binară. Datele sunt reprezentate în `C2`, numerele fiind considerate întregi. Datele au următorul format:



unde `S` este bitul de semn.

Unitatea de comandă tratează cuvântul ca o instrucţiune. Formatul instrucţiunilor este următorul:



Câmpurile au semnificaţia următoare:

ADR Câmpul de adresă al operandului (16 biți);

IM Indicatorul de adresare imediată: dacă IM = 1, câmpul ADR conține o dată imediată;

X Indicatorul de adresare indexată: dacă X = 1, adresa operandului este suma dintre conținutul registrului de index şi conținutul câmpului ADR;

I Indicatorul de adresare indirectă: dacă I = 1, câmpul ADR conține adresa cuvântului de memorie care conține adresa operandului;

COD Câmpul de cod al operației (5 biți).

Dacă I = X = IM = 0, adresarea este directă, deci câmpul ADR conține adresa operandului de memorie.

2.2.3. Setul de instrucțiuni

Pentru descrierea instrucțiunilor se vor utiliza coduri simbolice (mnemonice) ale acestora şi unele notații, acestea formând un limbaj de asamblare primitiv. Adresele şi valorile numerice vor fi scrise în hexazecimal. Pentru simplitate, adresele se vor considera ca fiind adrese de cuvinte, şi nu de octeți, astfel că după fiecare instrucțiune extrasă din memorie, numărătorul de adrese se va incrementa cu 1.

Codul mnemonic corespunde codului operației. Dacă instrucțiunea este cu referire la memorie, mnemonicul este urmat de un operand care indică modul de adresare şi adresa de memorie.

Pentru descrierea instrucțiunilor se utilizează următoarele notații:

data Reprezintă o constantă numerică, baza implicită fiind cea hexazecimală.

adr Reprezintă o adresă de memorie, baza implicită fiind cea hexazecimală.

op Reprezintă un operand adresat direct, indirect, indexat sau imediat. Poate avea una din formele următoare:

adr	operand de memorie cu adresare directă;
@*adr*	operand de memorie cu adresare indirectă;
X+*adr*	operand de memorie cu adresare indexată;
#*data*	operand cu adresare imediată.

op1 Reprezintă un operand adresat direct, indirect sau indexat. Poate avea una din formele următoare:

adr	operand de memorie cu adresare directă;
@*adr*	operand de memorie cu adresare indirectă;
X+*adr*	operand de memorie cu adresare indexată.

Instrucțiunile sunt prezentate în Tabelul 1.1.

Tabelul 1.1. Instrucțiunile calculatorului didactic CD.

Nr.	Instrucțiune		Semnificație	Descriere simbolică
1.	LDSP	*op*	Încărcare registru SP	SP ← *op*
2.	LDA	*op*	Încărcare registru A	A ← *op*
3.	LDX	*op*	Încărcare registru X	X ← *op*
4.	STA	*op1*	Memorare registru A	*op1* ← A
5.	STX	*op1*	Memorare registru X	*op1* ← X
6.	ADD	*op*	Adunare	A ← A add *op*
7.	SUB	*op*	Scădere	A ← A sub *op*

Nr.	Instrucțiune	Semnificație	Descriere simbolică
8.	AND *op*	ŞI logic	A ← A ∗ *op*
9.	OR *op*	SAU logic	A ← A + *op*
10.	JMP *op*	Salt necondiționat	NA ← *op1*
11.	JZ *op*	Salt dacă e zero	**if** (Z = 1) **then** (NA ← *op1*)
12.	JNZ *op*	Salt dacă nu e zero	**if** (Z = 0) **then** (NA ← *op1*)
13.	JC *op*	Salt dacă e carry	**if** (C = 1) **then** (NA ← *op1*)
14.	JNC *op*	Salt dacă nu e carry	**if** (C = 0) **then** (NA ← *op1*)
15.	CALL *op1*	Apel de subrutină	SP ← dec SP, M [SP] ← NA, NA ← *op1*
16.	XCHG	Interschimbare A cu X	A ↔ X
17.	INA	Incrementare A	A ← inc A
18.	INX	Incrementare X	X ← inc X
19.	DCA	Decrementare A	A ← dec A
20.	DCX	Decrementare X	X ← dec X
21.	CMA	Complementare A	$A \leftarrow \overline{A}$
22.	RLA	Rotire A la stânga	$C\&A \leftarrow A\&A_{22:0}\&0$
23.	RRA	Rotire A la dreapta	$C\&A \leftarrow A_0\&C\&A_{23:1}$
24.	CLC	Ştergere indicator Carry	C ← 0
25.	STC	Setare indicator Carry	C ← 1
26.	PUSH	Memorare A în stivă	SP ← dec SP, M [SP] ← A
27.	POP	Extragere A din stivă	A ← M [SP], SP ← inc SP
28.	RET	Revenire din subrutină	NA ←M [SP], SP ← inc SP
29.	NOP	Nici o operație	
30.	HLT	Oprire	

2.2.4. Exemple de programe

Se indică în stânga fiecărei instrucțiuni adresa de memorie la care se află instrucțiunea respectivă.

1) Adunarea a două cuvinte duble de memorie

Cuvintele se află la adresele `200h` şi `202h`, primele cuvinte fiind cele mai puțin semnificative (c.m.p.s.). Rezultatul este memorat în locul primului cuvânt.

```
0100        LDA   0200        ; cuvântul 1, partea c.m.p.s.
0101        ADD   0202        ; adună la cuvântul 2, partea c.m.p.s.
0102        STA   0200        ; memorează partea c.m.p.s. a rezultatului
0103        LDA   0201        ; cuvântul 1, partea c.m.s.
0104        JNC   0106        ; salt dacă nu este transport
0105        INA               ; incrementează pentru adunarea transportului
0106        ADD   0203        ; cuvântul 2, partea c.m.s.
0107        STA   0201        ; memorează partea c.m.s. a rezultatului
0108        HLT
```

Limbajul de asamblare utilizat are dezavantajul că trebuie ținută evidența adreselor de memorie. Un limbaj de asamblare mai evoluat permite utilizarea unor nume simbolice pentru indicarea adreselor de memorie. În locul adreselor numerice, se utilizează etichete simbolice, care reprezintă adresa la care se află instrucțiunea respectivă în memorie, valoarea etichetei fiind calculată şi atribuită de asamblor. Referirea la o adresă simbolică se realizează prin utilizarea acesteia în câmpul destinat operandului instrucțiunii. Nu este necesară etichetarea fiecărei instrucțiuni, ele ocupând cuvinte succesive de memorie.

2) Înmulțirea a două cuvinte prin metoda directă

Cuvintele se află la adresele DEINM şi INMUL. Rezultatul se memorează în două cuvinte aflate la adresele REZ şi REZ+1 (primul fiind cuvântul c.m.p.s.). La adresa CONTOR se păstrează contorul de iterații.

```
MULT        LDA   #0
            STA   REZ          ; inițializează produsul cu 0
            STA   REZ+1
            LDA   #18          ; numărul de iterații (24 în zecimal)
            STA   CONTOR       ; inițializează contorul de iterații
BUCLA       LDA   INMUL        ; încarcă înmulțitorul
            RRA                ; roteşte înmulțitorul pentru testarea bitului 0
            STA   INMUL        ; memorează înmulțitorul
            JNC   CONTIN       ; salt dacă bitul testat este 0
            LDA   REZ+1        ; adună deînmulțitul la cuvântul c.m.s.
            ADD   DEINM        ; al rezultatului
            STA   REZ+1        ; memorează produsul parțial
CONTIN      LDA   REZ+1        ; urmează deplasarea la dreapta
            CLC                ; a produsului parțial
            RRA                ; deplasează la dreapta cuvântul c.m.s.
            STA   REZ+1        ; şi îl memorează
            LDA   REZ
            RRA                ; deplasează la dreapta cuvântul c.m.p.s.
            STA   REZ          ; şi îl memorează
            LDA   CONTOR
            DCA                ; decrementează contorul
            STA   CONTOR
            JNZ   BUCLA        ; repetă dacă nu este 0
            HLT

DEINM       DW    5            ; inițializează deînmulțitul (un cuvânt)
INMUL       DW    9            ; inițializează înmulțitorul (un cuvânt)
REZ         DD    0            ; rezervă două cuvinte pentru rezultat
CONTOR      DW    0            ; rezervă un cuvânt pentru contorul de iterații
```

2.3. Structura calculatorului didactic

2.3.1. Structura generală

Unitatea aritmetică şi logică UAL conține un sumator/scăzător paralel şi circuite pentru operațiile logice ŞI, SAU, complementare. Pentru executarea acestor operații, un operand trebuie încărcat în registrul acumulator A, iar al doilea operand trebuie încărcat în registrul tampon RT. De aceea, aceste registre sunt conectate la cele două intrări ale UAL.

UAL este utilizată şi pentru indexarea adresei, adunând conținutul registrului de index la partea de adresă din registrul de instrucțiuni. Pentru aceasta, registrul de index X este conectat la una din intrările UAL, iar registrul de instrucțiuni RI este conectat la cealaltă intrare a UAL. Deoarece la intrările UAL trebuie să se conecteze mai multe registre, este avantajos să se utilizeze câte o magistrală la fiecare intrare. Aceste magistrale, de câte 24 de linii, sunt ABUS şi BBUS.

Registrele se conectează la magistrale astfel încât argumentele aceleiaşi operații să nu fie plasate pe aceeaşi magistrală. Astfel, registrele RI şi RT se conectează la magistrala ABUS, iar registrele A şi X se conectează la magistrala BBUS.

Deoarece registrele pot reprezenta destinațiile mai multor surse, este avantajoasă utilizarea unei magistrale comune pentru încărcarea tuturor registrelor. Această magistrală, denumită RBUS, este conectată la intrările fiecărui registru. Un transfer va fi efectuat în două etape: depunerea sursei pe magistrala RBUS, şi apoi transferul de pe magistrala RBUS în registrul destinație specificat.

Din setul de instrucțiuni se observă că trebuie să existe posibilitatea incrementării şi decrementării registrelor A, X, NA şi SP. Pentru a se putea utiliza aceeaşi logică de incrementare, respectiv decrementare, aceste registre sunt amplasate pe aceeaşi magistrală, BBUS. Incrementarea se poate rea-

liza cu o logică specială în UAL, sau prin adunarea valorii 1 pe magistrala ABUS (linia c.m.p.s. fiind 1, iar celelalte linii fiind 0). Pentru decrementare se poate utiliza o logică separată, sau se poate aduna complementul față de 2 al valorii 1.

Toate registrele care pot fi surse ale unui transfer sunt conectate fie la magistrala ABUS, fie la magistrala BBUS. Transferurile directe între registre sunt implementate prin depunerea conținutului registrului sursă pe magistrala ABUS sau BBUS, transferul acestuia pe magistrala RBUS şi de aici în registrul destinație. Transferul magistralelor ABUS şi BBUS pe magistrala RBUS nu se efectuează direct, ci prin intermediul UAL. De exemplu, pentru transferul magistralei ABUS pe magistrala RBUS, se utilizează o logică specială în cadrul UAL pentru a obține la una din ieşirile acesteia funcția identică (ABUS sau BBUS).

Magistralele ABUS şi BBUS au 24 de linii. Registrele X, NA, SP şi RA, de câte 16 biți, sunt conectate la liniile 0-15 ale acestor magistrale. Pentru a se permite efectuarea operațiilor aritmetice cu transport şi a celor de rotație, care implică indicatorul de transport C, magistrala RBUS are 25 de linii. La indicatorul de transport se conectează linia 24 a magistralei RBUS, iar la registre se conectează liniile 0-15 sau 0-23 ale magistralei. La indicatorul de transport se conectează şi pozițiile 23 şi 0 ale registrului acumulator. Ieşirea indicatorului de transport se conectează la pozițiile 23 şi 0 ale registrului acumulator. La indicatorul Z se conectează o funcție logică generată de UAL.

Schema bloc a calculatorului didactic este prezentată în Figura 1.1.

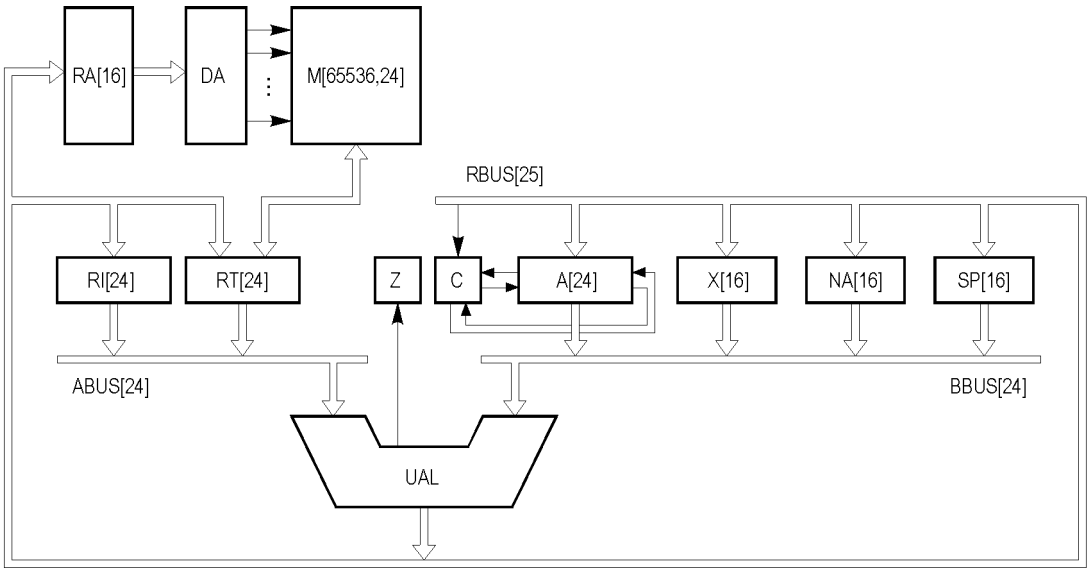


Figura 1.1. Schema bloc a calculatorului didactic.

2.3.2. Semnalele de comandă

Se prezintă în continuare semnalele de comandă care selectează principalele funcții şi operații.

c_1 Selectează funcția de adunare a UAL;
c_2 Selectează funcția de scădere a UAL;
c_3 Selectează funcția ŞI logic a UAL;
c_4 Selectează funcția SAU logic a UAL;
c_5 Selectează funcția identică pentru primul operand al UAL;
c_6 Selectează funcția de decrementare a UAL;
c_7 Selectează funcția de incrementare a UAL;
c_8 Selectează funcția NU logic a UAL;

c_9 Selectează funcția identică pentru al doilea operand al UAL;
c_{11} Poziționează indicatorul *Carry*;
c_{12} Incrementează numărătorul de secvențiere NS;
c_{18} Selectează intrarea multiplexorului MUX1:
0 – intrarea selectată este magistrala RBUS;
1 – intrarea selectată este cuvântul adresat din memorie;
c_{20} Validează citirea din memorie;
c_{21} Validează scrierea în memorie;
c_{22}, c_{23} Selectează intrarea multiplexorului MUX2:
00 - intrarea selectată este linia $RBUS_{24}$;
01 - intrarea selectată este bitul A_0 al registrului A;
10 - intrarea selectată este bitul A_{23} al registrului A;
c_{26}, c_{27} Selectează funcția registrului A:
00 - stare nemodificată;
01 - deplasare la dreapta cu o poziție;
10 - deplasare la stânga cu o poziție;
11 - încărcare paralelă;
c_{36} Șterge numărătorul de secvențiere NS;
c_{37} Validează încărcarea paralelă a numărătorului de secvențiere NS.

Celelalte semnale de comandă validează diferite operații de transfer sau încărcarea diferitelor registre.

2.3.3. Structura detaliată

Registrele care reprezintă diferite surse ale unui transfer sunt conectate la magistrale prin intermediul unor grupuri de porți ŞI, comandate prin semnale generate de unitatea de comandă. Registrele care se încarcă de la o singură sursă de date se conectează permanent la această sursă, dar înscrierea lor este sincronizată cu semnalul de tact condiționat de un semnal de comandă.

Schema bloc detaliată a calculatorului didactic este prezentată în Figura 1.2.

Unitatea de comandă a calculatorului conține un generator de faze, format dintr-un numărător de secvențiere NS şi un decodificator de secvențiere DS. Numărătorul de secvențiere, comandat de generatorul de tact, trece printr-o succesiune de stări. Dacă numărătorul este format din m bistabile, vor exista 2^m stări distincte. Ieşirile acestui numărător sunt decodificate cu ajutorul decodificatorului de secvențiere, obținându-se până la $n = 2^m$ ieşiri ale decodificatorului, care constituie semnalele de fază T_0, T_1, …, T_n. Acestea sunt impulsuri decalate în timp, şi fiecare din ele determină câte o fază. Durata fiecărei faze este egală cu perioada semnalului de tact, în acest timp fiind executată o microoperație. În cazul calculatorului didactic prezentat, există 11 semnale de fază, T_0, T_1, …, T_{10}.

Pot exista microoperații care necesită pentru execuție mai mult de o perioadă de tact, în aceste cazuri fiind necesare întârzieri de mai multe perioade de tact. O asemenea situație poate apare, de exemplu, în cazul unor memorii lente, la care datele citite devin disponibile după un timp mai mare decât o perioadă de tact. Un alt exemplu este cel al sumatorului din UAL, la care, de regulă, timpul necesar pentru formarea rezultatului este mult mai mare decât timpul necesar pentru un transfer simplu între registre. În acest ultim caz, datele trebuie menținute la intrarea sumatorului, pe magistralele ABUS şi BBUS, pe durata mai multor perioade de tact. Pentru întârzieri mai mari, se poate utiliza un numărător pentru contorizarea perioadei de întârziere.

Pentru simplitate se presupune că nu sunt necesare întârzieri suplimentare pentru operațiile cu memoria, iar sumatorul necesită numai o perioadă de tact pentru propagarea transportului şi formarea sumei.

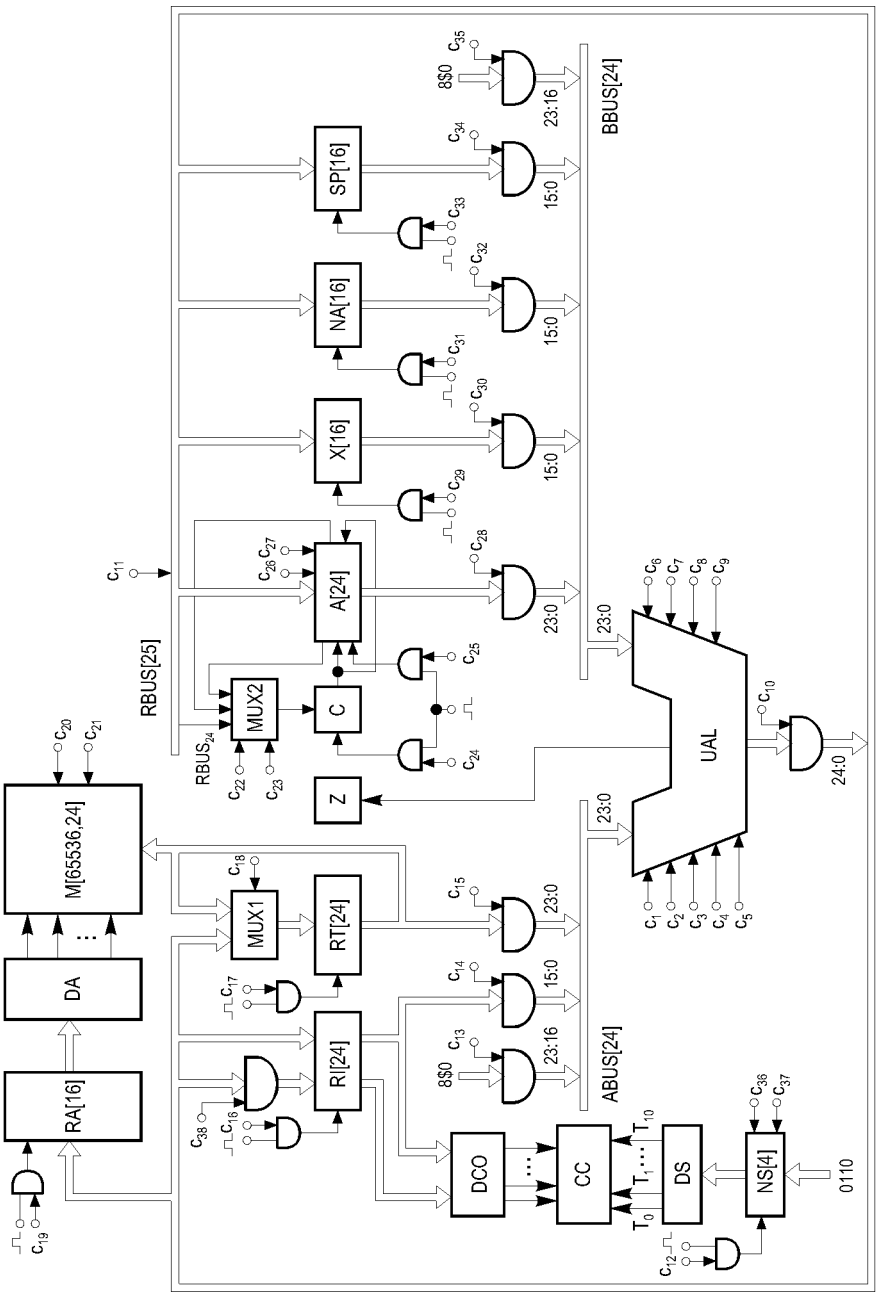


Figura 1.2. Schema bloc detaliată a calculatorului didactic.

Ieşirile generatorului de faze (T_0, T_1, …, T_{10}) sunt combinate cu semnalele rezultate în urma decodificării codului operației de către circuitul combinațional CC, şi astfel se generează succesiunea semnalelor de comandă necesare executării microoperațiilor în ordinea corectă.

2.4. Execuția instrucțiunilor

Secvența operațiilor necesare pentru execuția unei instrucțiuni se numeşte ciclu de instrucțiune. Fiecare ciclu de instrucțiune este format din ciclul de extragere şi ciclul de execuție. În ciclul de extragere, instrucțiunea este citită din memorie, iar în ciclul de execuție are loc execuția propriu-zisă a instrucțiunii.

Pentru instrucțiunile fără adresă, ciclul de extragere se termină atunci când instrucțiunea a fost transferată în registrul de instrucțiuni. Acelaşi lucru este valabil şi pentru instrucțiunile cu adresa-

re imediată. Pentru instrucțiunile cu o adresă care utilizează un alt mod de adresare, şi anume adresarea indirectă sau adresarea indexată, ciclul de extragere se continuă pentru a obține adresa efectivă care va fi utilizată în instrucțiune. Adresarea indirectă este cu un singur nivel.

În ciclul de extragere, adresa instrucțiunii următoare, conținută în numărătorul de adrese NA, este transferată în registrul de adrese RA. Se citeşte apoi cuvântul de memorie adresat de registrul RA, reprezentând codul instrucțiunii următoare, şi se transferă în registrul tampon RT, şi de aici în registrul de instrucțiuni RI. Numărătorul de adrese NA este incrementat. Descrierea simbolică a acestor microoperații este următoarea:

```
/T0/        RA ← NA;
/T1/        RT ← M[RA];
/T2/        RI ← RT,
            NA ← inc NA;
```

Se menționează că incrementarea numărătorului de adrese NA se realizează pe durata aceleiaşi faze T_2, ca şi transferul registrului tampon RT în registrul de instrucțiuni RI.

Pentru instrucțiunile cu adresare indirectă, la care indicatorul de adresare indirectă este poziționat (RI_{18} = 1), partea de adresă din registrul de instrucțiuni se utilizează pentru extragerea din memorie a adresei efective a operandului, care se transferă în registrul RT, iar apoi în registrul RI. Pentru aceste instrucțiuni, se execută deci în plus următoarea secvență de microoperații:

```
/T3*RI18/   RA ← RI15:0;
/T4*RI18/   RT ← M[RA];
/T5*RI18/   RI15:0 ← RT15:0;
```

Deoarece pentru instrucțiunile cu adresare indirectă sunt necesare 6 faze pentru ciclul de extragere, în cazul modurilor de adresare care necesită mai puține operații, numărătorul de secvențiere trebuie încărcat cu valoarea 6. În acest fel, ciclul de execuție va începe în faza T_6 pentru toate instrucțiunile.

Pentru instrucțiunile cu adresare indexată, la care indicatorul de adresare indexată este poziționat (RI_{17} = 1), adresa efectivă a operandului este calculată ca suma dintre partea de adresă din registrul de instrucțiuni şi registrul de index:

```
/T3*RI17/   RI15:0 ← RI15:0 add X,
            NS ← 6;
```

La sfârşitul acestor secvențe, partea de adresă a registrului de instrucțiuni ($RI_{15:0}$) conține deci adresa operandului.

Ciclul de execuție este specific pentru fiecare instrucțiune. Se presupune că în urma decodificării instrucțiunii curente, se activează un semnal de stare cu numele identic cu mnemonica instrucțiunii respective. De exemplu, dacă instrucțiunea curentă este STA, se activează semnalul STA. Pentru instrucțiunea STA, care memorează conținutul acumulatorului la adresa indicată în instrucțiune, se execută următoarea secvență de microoperații:

```
/T6*STA/    RT ← A;
/T7*STA/    RA ← RI15:0;
/T8*STA/    M[RA] ← RT,
            NS ← 0;
```

După execuția fiecărei instrucțiuni, numărătorul de secvențiere NS se resetează, pentru a începe un nou ciclu de extragere (corespunzător fazei T_0).

Ca un alt exemplu, se indică secvența de microoperații necesare execuției instrucțiunii CALL. Această instrucțiune apelează subrutina a cărei adresă este specificată în instrucțiune. Se decrementează mai întâi indicatorul de stivă SP, pentru a se putea memora un nou cuvânt în stivă. Indicatorul de stivă este transferat în registrul de adrese RA, pentru adresarea memoriei. Conținutul numărătorului de adrese NA, reprezentând adresa de la care se continuă programul după revenirea din subrutină, este transferat în registrul tampon RT, şi apoi în memorie. Pozițiile c.m.s. ale registrului tampon (biții 23-0)

se inițializează cu zerouri. În numărătorul de adrese se transferă apoi adresa de început a subrutinei, aflată în partea de adresă a registrului de instrucțiuni. Descrierea simbolică a acestor microoperații este următoarea:

```
/T6*CALL/     SP ← dec SP;
/T7*CALL/     RA ← SP;
/T8*CALL/     RT ← (8$0)&NA;
/T9*CALL/     M[RA] ← RT;
/T10*CALL/    NA ← RI15:0,
              NS ← 0;
```

În secvențele anterioare, microoperațiile de transfer sunt descrise în mod simplificat, fără a ține cont de faptul că transferurile se efectuează prin intermediul magistralelor. Aceste microoperații de transfer se pot detalia astfel încât să se țină cont de existența magistralelor. Pentru exemplificare, vom rescrie secvențele anterioare astfel încât transferurile să fie descrise în mod complet. Astfel, descrierea completă a microoperațiilor comune executate în ciclul de extragere de toate instrucțiunile este următoarea:

```
/T0/          BBUS15:0 = NA,              --
              RBUS = 0&BBUS,              -- RA ← NA
              RA ← RBUS15:0;              --
/T1/          RT ← M[RA];
/T2/          ABUS = RT,                  --
              RBUS = 0&ABUS,              -- RI ← RT
              RI ← RBUS23:0,              --
              NA ← inc NA;
```

Pentru instrucțiunile cu adresare indirectă, în continuarea ciclului de extragere se execută următoarea secvență de microoperații:

```
/T3*RI18/     ABUS15:0 = RI15:0,          --
              RBUS = 0&ABUS,              -- RA ← RI15:0
              RA ← RBUS15:0;              --
/T4*RI18/     RT ← M[RA];
/T5*RI18/     ABUS15:0 = RT15:0,          --
              RBUS15:0 = ABUS15:0,        -- RI15:0 ← RT15:0;
              RI15:0 ← RBUS15:0;          --
```

Microoperațiile detaliate executate la instrucțiunile cu adresare indexată sunt următoarele:

```
/T3*RI17/     ABUS15:0 = RI15:0,          --
              BBUS15:0 = X,               -- RI15:0 ← RI15:0 add X
              RBUS = ABUS add BBUS,       --
              RI15:0 ← RBUS15:0,          --
              NS ← 6;
```

Pentru instrucțiunea STA, se execută următoarea secvență de microoperații detaliate:

```
/T6*STA/      BBUS = A,                   --
              RBUS = 0&BBUS,              -- RT ← A
              RT ← RBUS23:0;              --
/T7*STA/      ABUS15:0 = RI15:0,          --
              RBUS = 0&ABUS,              -- RA ← RI15:0
              RA ← RBUS15:0;              --
/T8*STA/      M[RA] ← RT,
              NS ← 0;
```

În cazul instrucțiunii CALL, descrierea secvenței de microoperații executate este următoarea:

```
/T6*CALL/     BBUS = (8$0)&SP,            --
              RBUS = dec BBUS,            -- SP ← dec SP
              SP ← RBUS15:0;              --
/T7*CALL/     BBUS = (8$0)&SP,            --
              RBUS = 0&BBUS,              -- RA ← SP
```

```
                RA ← RBUS15:0;              --
/T8*CALL/       BBUS = (8$0)&NA,            --
                RBUS = 0&BBUS,              -- RT ← (8$0)&NA
                RT ← RBUS23:0;              --
/T9*CALL/       M[RA] ← RT;
/T10*CALL/      ABUS15:0 = RI15:0,          --
                RBUS = 0&ABUS,              -- NA ← RI15:0
                NA ← RBUS15:0,              --
                NS ← 0;
```

3. Desfăşurarea lucrării

3.1. Lansaţi în execuţie programul CALCDID.EXE care simulează funcţionarea calculatorului didactic. Utilizaţi opţiunea C (*Codificare instrucţiuni*), introducând mnemonicele instrucţiunilor din primul exemplu de program prezentat. După terminarea introducerii, indicată prin caracterul E, apăsaţi tasta S (*Start*) pentru începerea execuţiei instrucţiunilor. La fiecare apăsare a tastei T, calculatorul va executa operaţiile aferente unui semnal de tact. Identificaţi operaţiile executate pentru fiecare tact, vizualizând conţinutul registrelor şi semnalele de comandă active. Execuţia poate fi terminată prin apăsarea tastei E.

3.2. Codificaţi adresele simbolice din al doilea exemplu de program în adrese hexazecimale. Pentru aceasta întocmiţi mai întâi tabela de simboluri a programului, care va conţine toate numele simbolice utilizate şi valoarea numerică atribuită fiecărui simbol. Rescrieţi apoi programul sub forma în care este scris primul exemplu de program, înlocuind fiecare nume simbolic cu valoarea sa numerică. Editaţi programul cu un editor de texte, şi creaţi un fişier text. Lansaţi din nou în execuţie simulatorul, utilizând opţiunea F (*Fişier extern*) pentru încărcarea codului programului în memorie. Vizualizaţi execuţia programului.

3.3. Scrieţi un program pentru copierea unui tablou de 16 cuvinte de la o adresă sursă la o adresă destinaţie, utilizând adresarea indirectă. Codificaţi programul şi procedaţi apoi ca la punctul 3.2. Modificaţi apoi programul pentru a utiliza adresarea indexată.

3.4. Scrieţi secvenţele de microoperaţii sub formă simplificată pentru execuţia următoarelor instrucţiuni: LDA, STX, ADD, SUB, INA, DCX, JMP, PUSH, POP, RET. Scrieţi apoi secvenţele în mod detaliat pentru a ţine cont de transferurile efectuate prin intermediul magistralelor.