

Blocuri multiple de memorie.

Cresterea capacitatii memoriei principale/interne, indiferent de tehnologia de implementare, conduce la cresterea timpului de acces, ceea ce poate avea drept consecinta intrarea procesorului in stari inactive, de asteptare.

Adesea, exista situatii in care se cunosc anticipat adresele de acces la memorie. Pornind de la aceasta constatare, memoria interna poate fi organizata pe blocuri, cu porturi proprii de acces, adresele fiind intretesute intre blocuri.

In figura de mai jos este prezentata o memorie cu capacitatea de 256 cuvinte x 32 biti, structurata sub forma a 4 blocuri cu adrese intretesute. Organizarea adreselor este data in hexa, la nivelul fiecarui bloc.

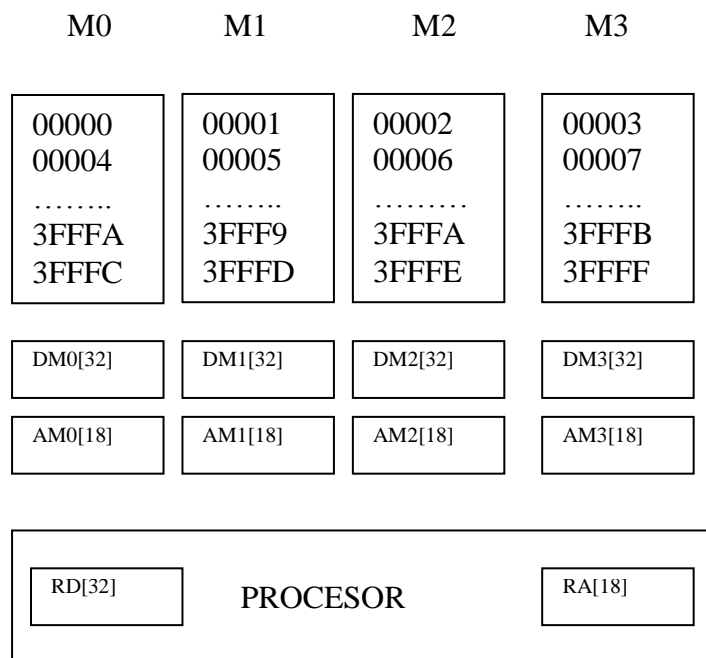


Fig. Memorie organizata in blocuri cu adrese intretesute.

Fiecare bloc va contine 2^{16} cuvinte de cate 32 de biti. In cazul unui procesor sofisticat este avantajos sa se citeasca simultan continutul mai multor adrese din memorie. Aceste cuvinte/informatii vor fi transferate catre o memorie rapida. In cadrul organizarii din figura de mai sus se pot extrage maximum 4 cuvinte simultan.

In continuare se va studia organizarea unei memorii pe blocuri, cu adrese intretesute si puncte multiple de acces.

Acest gen de organizare ofera avantajul ca asigura accesul simultan la mai multe blocuri, prin porturile specifice de acces.

La aceste porturi se pot conecta procesoare, in sistemele multiprocesor, sau echipamente de I/E, care lucreaza in regim de acces direct la memorie, ceea ce permite executia unor programe in paralel cu operatiile de I/E.

In continuare se va proiecta secventa de comanda pentru o memorie organizata pe blocuri cu porturi de acces. Fiecare bloc i de memorie poseda un registru de date DM_i si un registru de adrese AM_i . De asemenea, fiecare port de acces j este prevazut cu un registru de comunicatii de date CD_j si cu un registru de comunicatii de adrese CA_j .

Oricare din cele patru blocuri de memorie poate comunica cu oricare din cele patru porturi de acces. Unitatea de comanda a memoriei si procesorul/procesoarele sunt controlate cu acelasi semnal de ceas. Astfel, un procesor poate introduce un cuvint intr-un registru de comunicatie CA_j asociat, intr-o perioada de ceas, iar unitatea de control a memoriei il poate prelua in perioada imediat urmatoare. Printr-un registru CD_j transferul poate avea loc in ambele sensuri.

Cu fiecare registru CA_j sunt asociate doua bistabile de control C_j si S_j .

In cazul in care $C_j = 1$ se solicita o operatie de citire, iar in cazul in care $S_j = 1$ - una de scriere (un cuvint aflat in CD_j urmeaza sa fie stocat in memorie, la adresa specificata in CA_j). In cazul in care $S_j = C_j = 0$ registrele CA_j si CD_j sunt accesibile unitatii de comanda a portului de acces, ceea ce specifica unui eventual procesor activ posibilitatea de a trece la operatia urmatoare.

Transferul unidirectional al adreselor din registrele CA_j in registrele AM_i ($i, j = 0, \dots, 3$) se realizeaza prin intermediul magistralei $MAGA$. Transferul datelor intre registrele CD_i si DM_j , indiferent de sens, se efectueaza cu ajutorul magistralei $MAGD$.

Procesul de citire/scriere din/in memorie poate necesita mai multe perioade de ceas. Astfel, patru blocuri de memorie pot fi ocupate simultan cu diverse operatii, solicitate de diferite porturi de acces.

Pentru a controla corect fluxul de date este necesara introducerea, la fiecare bloc de memorie, a unor registre de cate doi biti: $DS_i[2]$ si $ST_i[2]$, care specifica destinatia

(portul de acces implicat), in cazul operatiei de citire, si respectiv starea blocului de memorie. Codificarea starii este urmatoarea:

- ST_i = 00 blocul i este inactiv;
- ST_i = 01 blocul i executa o operatie de scriere sau de citire;
- ST_i = 10 blocul a terminat o operatie de citire si datele asteapta sa fie preluate din registrul DM_i.

In figura de mai jos se prezinta organizarea unei memorii constand in patru blocuri M₀,...,M₃ si patru porturi de acces PA₀,...,PA₃, care comunica intre ele prin magistralele de date si adrese MAGD si, respectiv - MAGA.

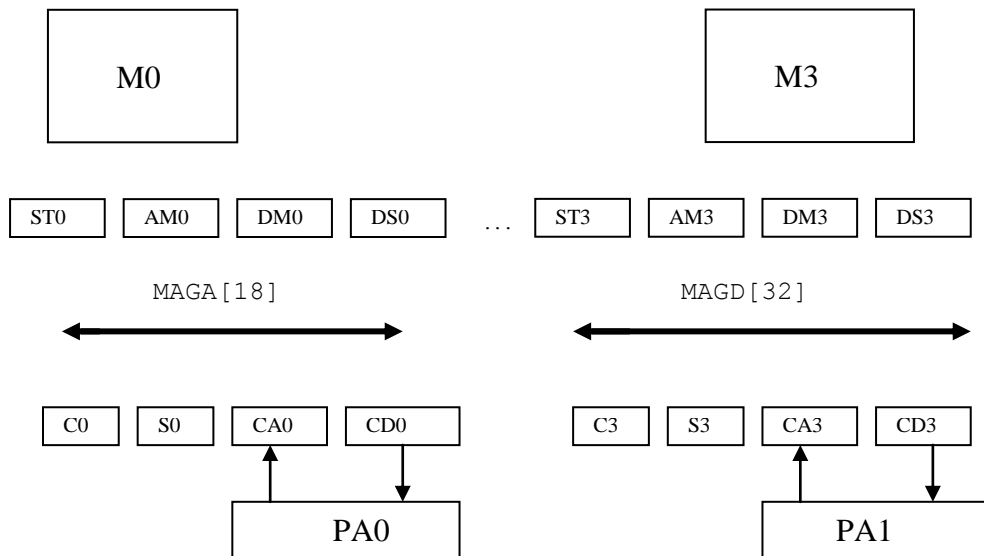


Fig. Organizarea unei memorii cu patru blocuri cu adrese intretesute si patru porturi de acces.

Presupunand o memorie cu capacitatea de 256 cuvinte, de cate 32 de biti fiecare, organizata in patru blocuri cu adrese intretesute, se poate stabili o structura a resurselor, la nivel de bloc si de port de acces, ca in figura urmatoare, unde se considera, spre exemplificare, blocul de memorie M₀ si portul de acces PA₂.

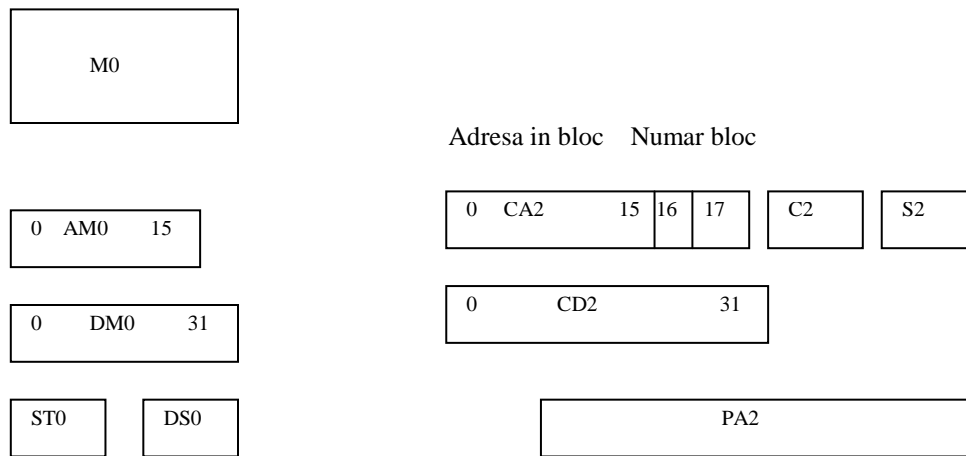


Fig. Resursele unui bloc de memorie si ale unui port de acces.

Se poate observa ca un registru CAi posedă 18 biti dintre care ultimii doi, mai puțin semnificativi, reprezintă numărul blocului, iar cei mai semnificativi 16 biti - adresa în bloc. Aceasta corespunde adresării întretesute. În cazul când primii doi biti mai semnificativi ai registrului CAi ar fi codificat numărul blocului, adresarea nu ar mai fi fost întretesută. Distribuția adreselor întretesute a fost prezentată mai sus.

Un procesor poate furniza la un moment dat patru adrese consecutive, pentru cele patru blocuri de memorie, numai în două perioade de tact pentru fiecare dacă porturile de acces sunt inactice. Se consideră că dacă un port de acces solicită ca patru cuvinte consecutive să fie citite din memorie portul de acces în cauză va fi pregătit să recepționeze cuvintele pe măsura ce acestea devin disponibile. În caz contrar cuvintele s-ar putea să fie transferate în registrul CA dat, în altă ordine decât cea în care au fost solicitate. Controlul porturilor de acces trebuie să țină seama de cererile de date pe care acestea le-au prezentat. În particular, un procesor nu trebuie să încerce să scrie într-un bloc de memorie, dacă o cerere de citire nu a fost satisfăcută.

Controlul unității va fi abordat sub aspectul traficului de informații pe magistralele de adrese și de date. Pentru a evita reducerea de viteză a procesului de transfer al informației trebuie să se permită transferuri simultane pe cele două magistrale.

Secvența de control este constituită din două părți (vezi organigrama). Prima parte va accepta o nouă cerere de citire și simultan va putea livra datele de la o operație de citire

deja terminata. Partea a doua trateaza cererile de scriere. Ramura din dreapta a primei parti specifica modul de interogare al fiecarui port de acces in scopul detectarii unei eventuale cereri de citire.

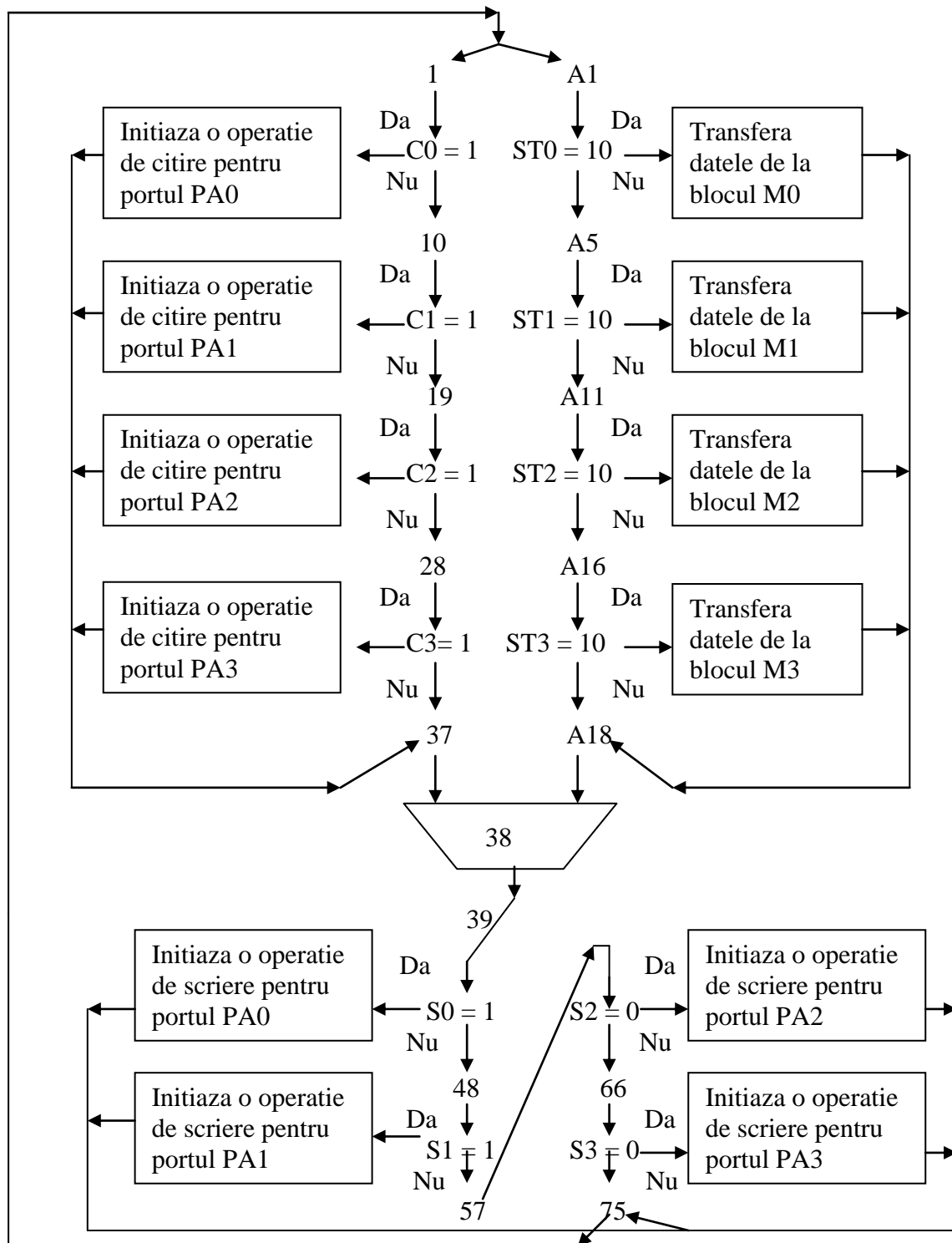


Fig. Secventa de control pentru memorie

Prima cerere intalnita este implementata prin transferul adresei catre blocul de memorie selectat, folosind MAGA. In acelasi timp un cuvint poate fi returnat de la un bloc de memorie catre un port de acces, prin magistrala MAGD - ramura din dreapta.

Dupa tratarea a cel mult unei cereri de citire si a unui transfer de date, comanda converge pentru a permite cautarea unei eventuale cereri de scriere. Secventa de scriere va fi implementata numai pentru o singura cerere - prima intalnita. Cererile de scriere sunt tratate separat, deoarece in cazul lor se transmit simultan la blocul de memorie atat adresa, cat si data. In organigrama a fost incorporat un sistem de prioritati pe baza caruia porturile de acces si blocurile de memorie cu indici mai mici au prioritati mai mari. Intervalele de timp pentru citirea si scrierea in blocurile de memorie sunt mult mai mari decat intervalul de timp necesar pentru efectuarea mai multor treceri prin secventa de comanda a magistralelor. Fiecare transfer de adresa sau de data intra-un procesor si un bloc de memorie necesita numai o perioada de tact.

In continuare se prezinta secventa detaliata pentru initializarea unei cereri de citire pentru PA0.

1. $\rightarrow(C0 \cap DCD(CA016:17), \overline{C0})/(2,4,6,8,10)$

2. $\rightarrow(ST00 \cup ST01)/(10)$; Fara Intarziere

3. $DS0, ST0, C0 \leftarrow 0,0,0,1,0$; $MAGA = CA0$;

$AM0 \leftarrow MAGA$; $cm0 = 1$;

$\rightarrow(37)$; Fara Intarziere

4. $\rightarrow(ST10 \cup ST11)/(10)$; Fara Intarziere

5. $DS1, ST1, C0 \leftarrow 0,0,0,1,0$; $MAGA = CA0$;

$AM1 \leftarrow MAGA$; $cm1 = 1$;

$\rightarrow(37)$; Fara Intarziere

6. $\rightarrow(ST20 \cup ST21)/(10)$; Fara Intarziere

7. $DS2, ST2, C0 \leftarrow 0,0,0,1,0$; $MAGA = CA0$;

$AM2 \leftarrow MAGA$; $cm2 = 1$;

$\rightarrow(37)$; Fara Intarziere

8. $\rightarrow(ST30 \cup ST31)/(10)$; Fara Intarziere

9. $DS3, ST3, C0 \leftarrow 0,0,0,1,0$; $MAGA = CA0$;

$AM3 \leftarrow MAGA$; $cm3 = 1$;

→(37); Fara Intarziere

Secventa detaliata pentru initializarea unei cereri de citire pentru portul PA1.

10. →($C1 \cap DCD(CA116:17), \overline{C1}$)/(11,13,15,17,19,)

11. →($ST00 \cup ST11$)/(19); fara Intarziere

12. $DS0, ST0, C1 \leftarrow 0, 10, 1, 0$; $MAGA = CA1$;

$AM0 \leftarrow MAGA$; $cm0 = 1$;

→(37); Fara Intarziere

13. →($ST10 \cup ST11$)/(19); Fara Intarziere

14. $DS1, ST1, C1 \leftarrow 0, 1, 0, 1, 0$; $MAGA = CA1$;

$AM1 \leftarrow MAGA$; $cm1 = 1$;

→(37); Fara Intarziere

15. →($ST20 \cup ST21$)/(19); Fara Intarziere

16. $DS2, ST2, C1 \leftarrow 0, 1, 0, 1, 0$; $MAGA = CA1$;

$AM2 \leftarrow MAGA$; $cm2 = 1$;

→(37); Fara Intarziere

17. →($ST30 \cup ST31$)/(19); Fara Intarziere

18. $DS3, ST3, C1 \leftarrow 0, 1, 0, 1, 0$; $MAGA = CA1$;

$AM3 \leftarrow MAGA$; $cm3 = 1$;

→(37); fara Intarziere

.....

La pasul 1 se constata efectuarea unui salt cu patru ramificatii in functie de blocul de memorie adresat de CA0. Daca blocul in cauza este inactiv, ceea ce se testeaza la pasul 2, pasul 3 va produce modificarile necesare de stare si va conecta magistrala MAGA la CA0, pentru a se efectua transferul adresei la AM0. La pasul 3 are loc transferul adresei si lansarea unei cereri de citire catre M0. Pentru cazurile in care sunt adresate blocurile de memorie M1, M2, M3 sunt necesare trei secvente separate a cate doi pasi: 4 si 5, 6 si 7, 8 si 9. Pasii 10 - 37 solutioneaza cererile de citire emise de porturile PA1, PA2, PA3 catre blocurile de memorie M0, M1, M2, M3.

Secventa concurenta pentru returnarea datelor consta din pasii A1 - A5, care trateaza transferurile de la M0, la PA0,...,PA3. Secvente similare (pasii A6,...,A20) vor fi elaborate si pentru blocurile de memorie M1,M2, M3.

A1. $\rightarrow(\text{ST00} \cap \text{DCD}(\text{DS0}), \overline{\text{ST00}}) / (\text{A2}, \text{A3}, \text{A4}, \text{A5}, \text{A6})$

A2. $\text{ST0} \leftarrow 0,0; \text{MAGD} = \text{DM0};$

$\text{CD0} \leftarrow \text{MAGD}; \text{gata} = 1;$

$\rightarrow(\text{A21});$ Fara Intarziere

A3. $\text{ST0} \leftarrow 0,0; \text{MAGD} = \text{DM0};$

$\text{CD1} \leftarrow \text{MAGD}; \text{gata} = 1;$

$\rightarrow(\text{A21});$ Fara Intarziere

A4. $\text{ST0} \leftarrow 0,0; \text{MAGD} = \text{DM0};$

$\text{CD2} \leftarrow \text{MAGD}; \text{gata} = 1;$

$\rightarrow(\text{A21});$ Fara Intarziere

A5. $\text{ST0} \leftarrow 0,0; \text{MAGD} = \text{DM0};$

$\text{CD3} \leftarrow \text{MAGD}; \text{gata} = 1;$

$\rightarrow(\text{A21});$ Fara Intarziere

A6. $\rightarrow(\text{ST10} \cap \text{DCD}(\text{DS1}), \overline{\text{ST10}}) / (\text{A7}, \text{A8}, \text{A9}, \text{A10}, \text{A11})$

A7. $\text{ST1} \leftarrow 0,0; \text{MAGD} = \text{DM1};$

$\text{CD0} \leftarrow \text{MAGD}; \text{gata} = 1;$

$\rightarrow(\text{A21});$ Fara Intarziere

A8. $\text{ST1} \leftarrow 0,0; \text{MAGD} = \text{DM1};$

$\text{CD1} \leftarrow \text{MAGD}; \text{gata} = 1;$

$\rightarrow(\text{A21});$ Fara Intarziere

A9. $\text{ST1} \leftarrow 0,0; \text{MAGD} = \text{DM1};$

$\text{CD2} \leftarrow \text{MAGD}; \text{gata} = 1;$

$\rightarrow(\text{A21});$ Fara Intarziere

A10. $\text{ST1} \leftarrow 0,0; \text{MAGD} = \text{DM1};$

$\text{CD3} \leftarrow \text{MAGD}; \text{gata} = 1;$

$\rightarrow(\text{A21});$ Fara Intarziere

A11.

Pasul A1 genereaza un salt, in functie de destinatia datelor, specificata in DS0.

Pasul A2 stabileste conectarea magistralei; inactiveaza starea blocului de memorie; transmite cuvantul solicitat catre CD0 si semnaleaza acest fapt prin semnalul gata = 1. De la pasii 37 si A21 controlul este directionat catre pasul de convergenta 38, de la care incepe secventa de servire a cererilor de scriere.

A18. →(38); Fara Intarziere.

37. →(38); fara Intarziere

38. CONVERGE(A18,37)

39. →($S0 \cap \overline{DCD(CA016:17)}, S0$)/(40,42,44,46,48)

40. →($ST00 \cup ST01$)/(48); Fara Intarziere

41. $ST0, S0 \leftarrow 0, 1, 0$; $MAGA = CA0$;

$MAGD = CD0$; $AM0 \leftarrow MAGA$; $DM0 \leftarrow MAGD$; $sm0 = 1$;

→(75)

42. →($ST10 \cup ST11$)/(48); Fara Intarziere

43. $ST1, S0 \leftarrow 0, 1, 0$; $MAGA = CA0$;

$MAGD = CD0$; $AM1 \leftarrow MAGA$; $DM1 \leftarrow MAGD$; $sm1 = 1$;

→(75)

44. →($ST20 \cup ST21$)/(48); Fara Intarziere

45. $ST2, S0 \leftarrow 0, 1, 0$; $MAGA = CA$

$MAGD = CD0$; $AM2 \leftarrow MAGA$; $DM2 \leftarrow MAGD$; $sm2 = 1$;

→(75)

46. →($ST30 \cup ST31$)/(48); Fara Intarziere

47. $ST3, S0 \leftarrow 0, 1, 0$; $MAGA = CA0$;

$MAGD = CD0$; $AM3 \leftarrow MAGA$; $DM3 \leftarrow MAGD$; $sm3 = 1$;

→(75)

48.

75. DIVERGE(1,A1)

Pasii 39 - 47 trateaza cazurile in care sunt adresate, pentru scriere, blocurile de memorie M0, M1, M2, M3 de catre portul de acces PA0. Pasii 48 - 74 trateaza cazurile de scriere generate de porturile de acces PA1, PA2, PA3.

Secventa de control prezentata nu este in mod necesar cea mai rapida sau eficienta. O trecere prin secventa va necesita cel mult patru perioade de ceas. Daca ciclul memoriei este de 50 - 100 de ori mai lung decat perioada ceasului, intervalul de parcurgere de patru perioade nu este semnificativ. Daca ciclul memorie se reduce la 10 perioade de ceas, atunci parcurgerea va introduce o Intarziere importanta.

In principiu se poate recurge la inlocuirea magistralelor prin circuite care sa asigure transferuri directe. O solutie mai eficienta ar consta in eliminarea completa a transferurilor si directionarea combinationala a adreselor. Aceasta va duce la o retea combinationala extrem de extinsa.