PLANIFICAREA EXECUŢIEI PROCESELOR ÎN SISTEMELE MULTIPROCES CU COMANDĂ PROGRAMATĂ

1.INTRODUCERE

Sistemele multiproces cu comandă programată (SMCP) se realizează prin n>1 procese secvenţiale care se desfăşoară în paralel.

Realizarea SMCP necesită un sistem de operare de timp real (SOTR) sau monitor de timp real sau executiv de timp real (variante simplificate), care trebuie să rezolve problemele:

- planificarea execuţiei proceselor (iniţiere, execuţie,suspendare, reluare -conform unei priorităţi, operaţiile de I/O,tratarea întreruperilor);
- pentru procesele interdependente: accesul exclusiv la resurse comune, sincronizare, comunicaţie;
- în sistemele complexe: gestiunea dinamică a memoriei.

În cadrul SMCP, procesele secvenţiale se pot afla la un moment de timp dat, între punctul iniţial şi punctul final al execuţiei lor şi concurează pentru resurse hardware şi software comune.

Paralelismul proceselor poate fi **fizic** (procesele se desfaşoară efectiv simultan în două unităţi centrale de prelucrare (UCP) diferite) sau **virtual** (procesele se execută cu diviziune în timp,prin alocarea succesivă de către un program planificator a unor intervale de timp (cuante) - pentru fiecare proces).

În cadrul SMCP, procesele pot fi independente sau interdependente (manevrează resurse hardware şi software comune, comunică între ele).

Pentru supervizarea execuţiei proceselor în sistemele multiproces, se utilizează un program (rezident în memoria calculatorului) care conduce la realizarea unui **microprocesor virtual** mult mai puternic decât cel original şi mult mai simplu de utilizat.Acest program (**multiplexor sau planificator de procese**) asigură următoarele avantaje:

- simplifică descrierea proceselor şi programarea lor;
- permite realizarea mai simplă a sistemelor cu multe procese ierarhizate.

Dezavantajele acestei soluţii sunt:

- necesitatea unui consum de memorie şi de timp de prelucrare suplimentar;
- generalitatea nejustificată pentru multe aplicaţii particulare;
- imposibilitatea de a trata un număr foarte mare de procese simple.

Din punct de vedere al programului de planificare, toate procesele din SMCP sunt interdependente (concurează pentru resurse comune: procesor, memorie etc., chiar dacă la nivelul aplicaţiei ele sunt total independente).

2.PRINCIPII DE PLANIFICARE A EXECUŢIEI PROCESELOR ÎN SMCP

Criteriile de clasificare a metodelor de planificare în SMCP sunt următoarele:

A. - diviziunea în timp a prelucrărilor:
- uniformă;
- neuniformă;

B. - priorităţile proceselor:
- egale;
- inegale: - fixe;
- variabile;

C. - tipul întreruperilor utilizate pentru iniţierea prelucrărilor şi numărul de nivele de întreruperi:
- periodice (IP): - un nivel;
  - n>1 nivele;
- asincrone: - generate de intrări;
  - n>=1 nivele;
- mixte: IP + asincrone.

D. - modul de suspendare a execuţiei unui proces:

- voluntară : autosuspendat după efectuarea unei faze de execuţie a sa (exemplu - efectuarea unei tranziţii pe GHT)
- forţată: - întreruptă de procese prioritare;
  - la expirarea timpului alocat de SOTR procesului;
  - în aşteptarea îndeplinirii unei condiţii;
- mixtă.

2.1 SISTEME CU PROCESE INDEPENDENTE

a) Planificarea cu diviziune uniformă în timp pentru procese de priorităţi egale

Caracteristici :

A - diviziune uniformă
B - priorităţi egale
C - un nivel de întreruperi periodice (cu perioada T/N) ; se activează procedura de planificare
D - suspendare mixtă - voluntară sau la expirarea unei limite de timp

Presupunem îndeplinite următoarele ipoteze :
- numărul proceselor este fix (egal cu N)
- perioada de activare pentru fiecare proces este egală cu T
- cuanta de timp alocată pentru un proces este T/N

Metoda de planificare constă în :

1. Se alocă fiecărui proces P0,P1, ... , Pn-1 ,maximum o cuantă de timp T/N, pentru execuţie , după care procesul este suspendat (şi reluat la ciclul T următor). După suspendarea unui proces se pune în execuţie procesul următor.
2. Procesul poate ceda voluntar UCP , dacă îşi termină execuţia fazei curente înainte de T/N (de exemplu pentru procese descrise prin grafuri hibride de tranziţii (GHT) cu timpul de prelucrare maxim < T/N)
3. Activarea planificatorului se realizează în două moduri:
   - în întreruperi periodice (la T/N) şi
   - prin chemări din procese (care s-au autosuspendat)

Avantajele acestei metode de planificare sunt:

- procesele se pot programa liber (cu sau fără GHT)
- planificarea este independentă de duratele de execuție ale tranzițiilor pe GHT datorită limitelor de timp impuse de planificator fiecărui proces)

Structura de date a planificatorului

Pentru fiecare proces Pi cu i=0,N-1 există un bloc de control al procesului BCPi cu următoarea structură:

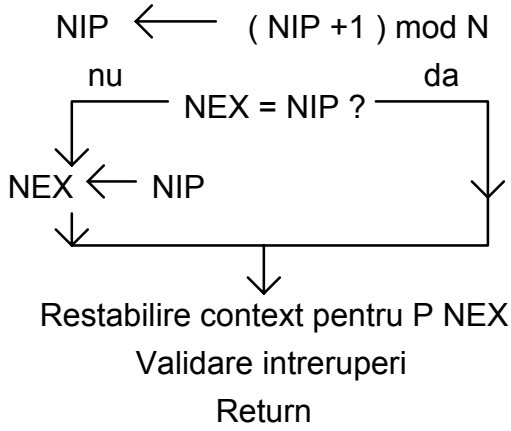
zona salvare context la suspendare	se salveaza parametri necesari reluarii executiei
adresa de start Pi *	la initializare; in timpul functionarii contine adresa de reluare
adresa virfului de stiva curent	SP pentru lucrul cu stiva procesului Pi
contoare de timp	pentru verificarea unor expirari de timp

Algoritmul de planificare este următorul:

Inițializări BCPi (pentru P0,P1,...,P N-1);
NIP ← N-1;
NEX ← N-1; (N-1 proces numit proces de bază)
Pune în execuție PNEX;

Subrutina de tratare a întreruperilor periodice
(activarea planificatorului)

Salvează context P NEX în stiva acestuia;



În cazul autosuspendării procesului curent se apelează următoarea subrutină:

Invalidare întreruperi;
Salvare context P NEX;
NEX ← (NEX + 1) mod N; (sau NEX ← N-1;)
Restabilește contextul P NEX;
Validare întreruperi;

Return;

Notații: NEX - numărul procesului curent în execuție;
NIP - contor de întreruperi periodice;

b) Planificarea cu diviziune neuniformă în timp pentru procese de priorități egale

Caracteristici: A - diviziune neuniformă;
B,C,D - la fel ca la planificarea cu diviziune uniformă;

Sunt valabile următoarele ipoteze:

- Numărul proceselor este fix (egal cu N);
- Perioada de activare pentru un proces este >=T ;

Metoda de planificare este următoarea:

1. Se defineşte un macrociclu MT după care funcționarea sistemului este periodică;
2. În fiecare ciclu T numărul proceselor activate poate fi diferit; numărul de cuante de timp alocate pentru fiecare proces variază - deci diviziunea în timp a prelucrărilor este neuniformă;
3. Suspendarea unui proces se realizează :
   - la expirarea limitei de timp alocată (suspendarea este forțată) sau
   - la cerere (autosuspendare);

Timpul rămas disponibil după autosuspendare se alocă fie procesului următor , fie procesului N-1 - numit proces de bază;

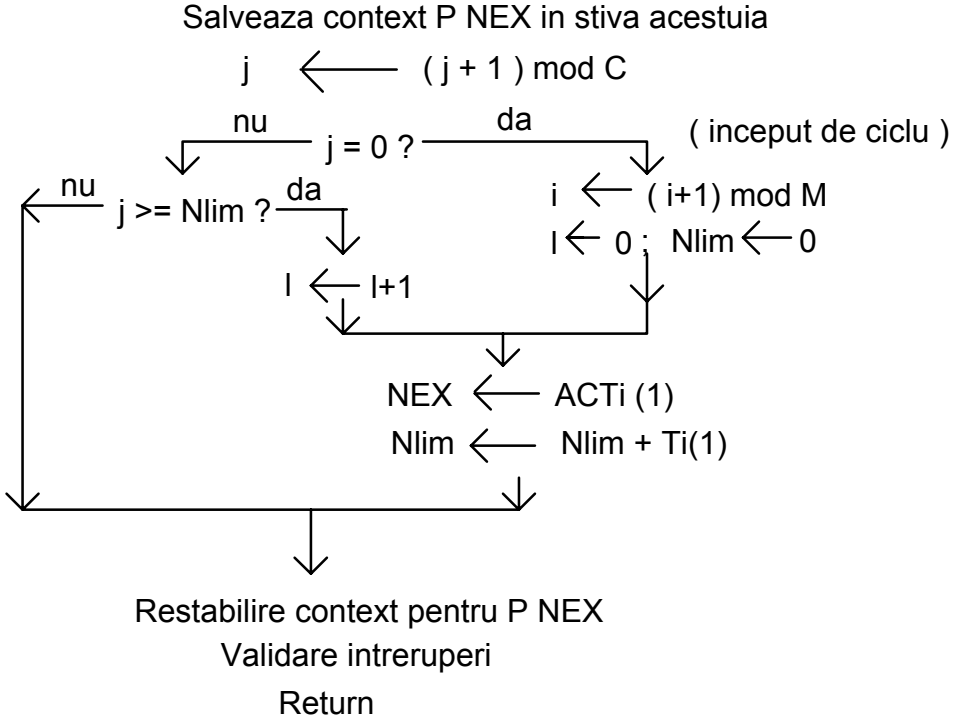
Structura de date a planificatorului :

1. Blocurile de control ale proceselor (ca la varianta anterioară);
2. Tabele de activare pentru fiecare ciclu i : ACT i (i=0,N-1) cu numerele proceselor active în ciclul i;
3. Tabele cu cuante de timp alocate fiecarui proces activat în ciclul i : T i;

Algoritmul de planificare este următorul:

Inițializări BCPi (pentru P0,P1,...,P N-1);
NEX ← N-1;
i ← M-1;
j ← C-1;
Pune în execuție P NEX;

Subrutina de tratare a întreruperilor periodice
(activarea planificatorului)



În cazul autosuspendării procesului curent se apelează următoarea subrutină:

Invalidare întreruperi;		Invalidare întreruperi;
Salvare context P NEX;		Salvare context P NEX;
NEX ← ACTi(l);	sau	NEX ← N-1;
Nlim ← Nlim + Ti(l);		{procesul de bază}
Restabileşte contextul P NEX;		Restabileşte contextul P NEX;
Validare întreruperi;		Validare întreruperi;
Return;		Return;

Notații: i - contor de cicluri în macrociclu;
j - contor de interval T/C (cuantă de timp) în ciclu;
l - pozitia curentă în tabelul de activare ACTi şi în tabelul cu numărul de cuante de timp Ti;
Nlim - momentul limită pentru procesul curent (măsurat în număr de cuante de timp);
NEX - numărul procesului curent în execuție;

c) Planificarea proceselor independente ierarhizate

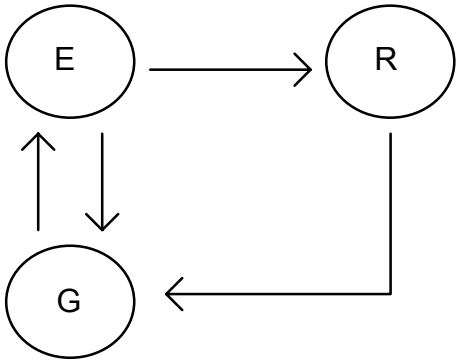
Realizarea sistemelor complexe impune utilizarea de procese ierarhizate (cu priorități diferite) în care o prelucrare de prioritate Pi este întreruptă de o prelucrare de prioritate
Pj > Pi.

Planificarea proceselor independente ierarhizate se realizează prin intermediul:
- programelor rezidente standard (SOTR)

sau al

- planificatoarelor scrise de proiectantul sistemului şi adaptate la aplicație.

Situațiile în care se poate afla un proces din punct de vedere al planificatorului pot fi descrise printr-un graf ca cel din figură.Trecerea unui proces dintr-o situație (stare) în alta este executată de către procedurile planificatorului.



unde E,G,R sunt stări ale proceselor (din punct de vedere al planificatorului):

- E = execuție;
- G = gata de execuție;
- R = repaus (procesul nu există din punct de vedere al planificatorului).

Vom da un exemplu simplu de planificator cu priorități.

Din punct de vedere al priorităților prelucrărilor în UCP, procesele se pot clasifica în:

a. - procese de prioritate maximă:
- planificatorul (activat prin IP);
- prelucrările "rapide" - utilizate pentru a răspunde la evenimentele semnalate de întreruperile asincrone (IA)(se mai numesc "handlere" de întreruperi);

b. - procese prioritare activate sincron, la intervale fixe, cu diviziune neuniformă;

c. - procese de prioritate minimă :

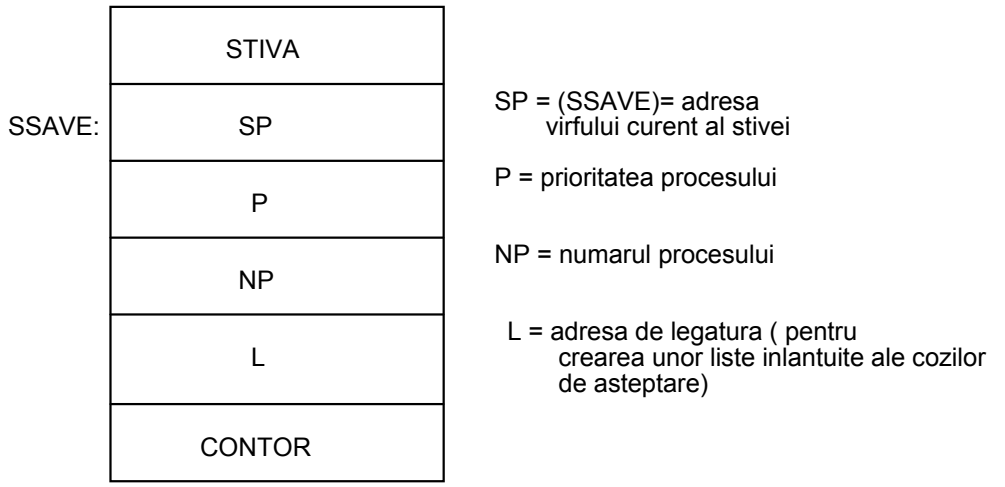
- procese neprioritare activate asincron, în intervalele disponibile, în ordinea priorităților (numărul de nivele de prioritate = k).

Suspendarea execuției proceselor se realizează fie prin autosuspendarea proceselor aflate în execuție, fie prin suspendarea forțată:
- la expirarea timpului alocat de SOTR procesului (pentru procesele prioritare);
- în situația în care trebuie executat un proces prioritar.

Ipoteze simplificatoare:

-se presupune un număr fix de procese sincrone şi asincrone:
- N procese sincrone;
- R procese asincrone.
- alocarea memoriei este prestabilită (fixă)

Structura de date este alcătuită din N+R blocuri de control proces (BCP):
Cozile de aşteptare (activare) Q1...Qk, în care procesele neprioritare aşteaptă punerea în execuție, sunt de tip FIFO. Fiecare Qi este organizată ca listă înlănțuită. Tablourile pentru activările sincrone sunt similare cu cele de la pct.b.



Algoritmul de planificare este o extensie a algoritmului de la diviziunea neuniformă cu priorități egale, cu următoarele modificări:

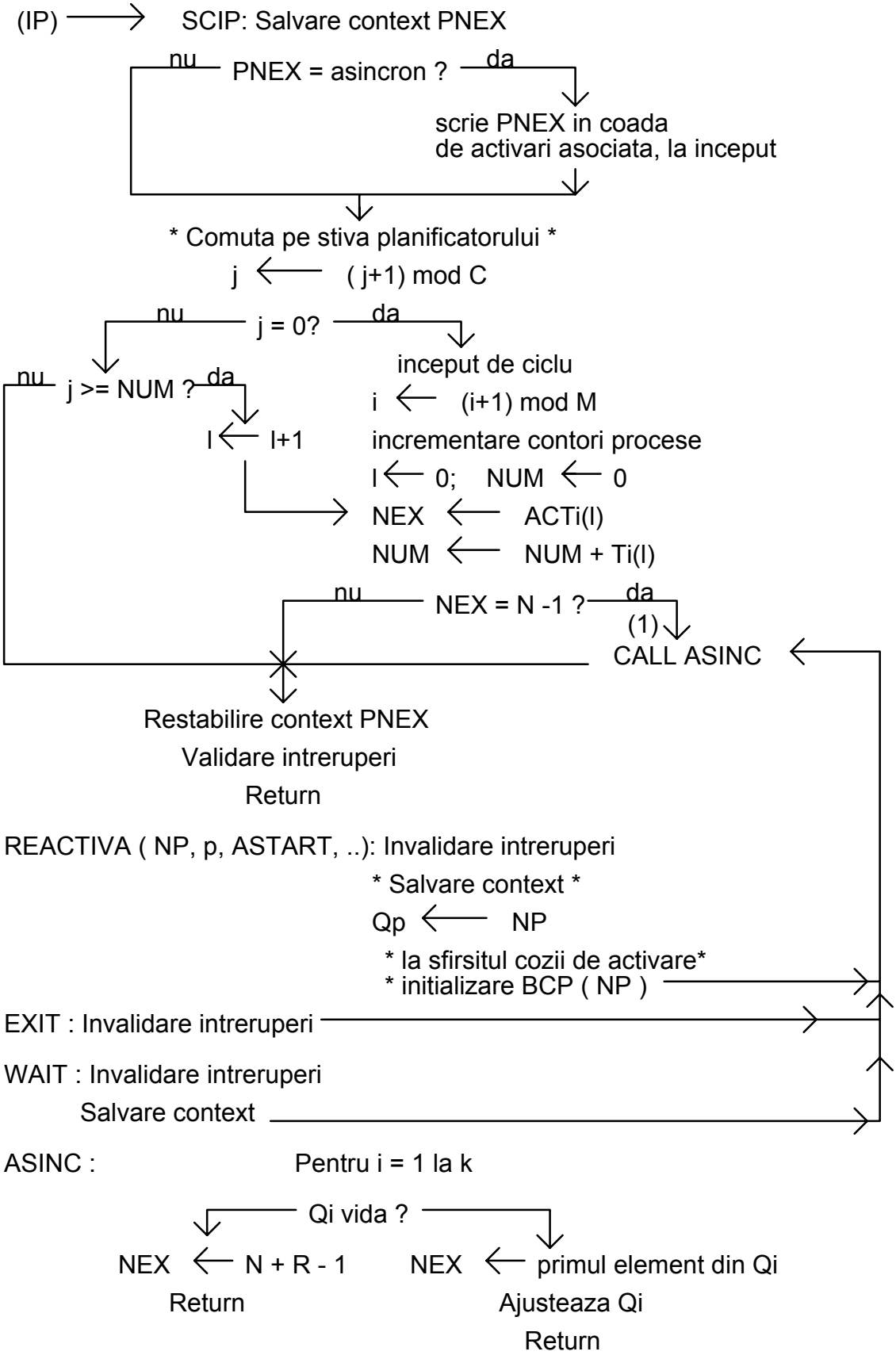
- timpul alocat unui proces fictiv sincron P_{N-1} se alocă pentru procesele neprioritare; pentru fiecare nivel de prioritate (pentru procesele asincrone), procesele sunt executate în ordinea sosirii lor în cozile de activare a nivelului respectiv;
- un proces asincron intrat în execuție şi suspendat datorită unui proces prioritar este repus în coada pe nivelul sau (dar la început), astfel încât la reluarea execuției pentru nivelul respectiv va fi primul repus în execuție.

Sunt definite următoarele proceduri:

SCIP = procedura principală de planificare (activată la fiecare IP);
ASINC = procedura care întoarce în NEX numărul celui mai prioritar proces asincron aflat în cozile de activare;
ACTIVA = procedura care activează un proces asincron (de prioritate mică) prin chemarea sa dintr-un proces sincron sau SCIA;
SCIA = procedura de servire a întreruperilor "rapide" şi eventual pentru activări ale unor procese asincrone;
REACTIVA = procedura de reactivare a unui proces asincron la inițiativa proprie (după terminarea unei faze,procesul se repune el însuşi în coada de activare, pe nivelul corespunzător de prioritate);
EXIT = procedura de autosuspendare (terminare a execuției) pentru un proces asincron;
WAIT = procedura de autosuspendare (temporară) a unui proces sincron (până la ciclul următor).

Algoritmul de planificare este următorul:

```
Inițializare generală
Pentru i=0 la N+R-1
      inițializează BCPi
NEX ← N+R-1
i ← N-1
j ← C-1
Restabilire context PNEX
Validare întreruperi
     Return
```



(1) Timpul alocat procesului N-1 se alocă de fapt proceselor asincrone, dacă cozile Qi de aşteptare a intrării în execuţie sunt nevide.

(IA) → SCIA: Salvari în stiva curentă
Prelucrări "rapide"
("handler" întreruperi)
* CALL ACTIVA(...) *
Restabiliri
Validare întreruperi
Return

ACTIVA (NP,p,ASTART,param): Invalidare întreruperi
Salvări în stiva curentă
Qp ← NP
* la sfârşitul cozii de aşteptare *
* Reinițializare BCP(NP) *
Restabiliri
Validare întreruperi
Return

Notații: NP - numărul procesului;
p - prioritatea;
ASTART - adresa de start;
param - alti parametrii;

3.DESFĂ□URAREA LUCRĂRII

Se va studia funcționarea unui program de planificare a proceselor (corespunzător variantei c) ; se vor modifica tabelele de activare a proceselor şi se va studia efectul acestor modificări.

DESCRIEREA PROGRAMULUI DE SIMULARE A PLANIFICARTORULUI DE PROCESE

Procesele care se pot executa de către planificator sunt:

- sincrone (1,2,3,4,5);
- asincrone (6,7,8,9,10).

Procesul 5 este procesul de bază.
Procesele asincrone sunt organizate pe două nivele de prioritate
(0 - cea mai prioritară; 1 - mai puțin prioritară).
Primitivele planificatorului sunt:

- WAIT - autosuspendare procese sincrone; se pune în execuție primul proces asincron din coada prioritară;
- REACTIV - autosuspendare procese asincrone; se pune în execuție primul proces asincron din coada prioritară.

În cazul primitivelor WAIT şi REACTIV, dacă cozile de aşteptare ale proceselor asincrone sunt vide, se repune în execuție procesul care apelează primitiva.
- EXIT - procesul trece în starea "repaus";
- ACTIV(p)- procesul "p" este trecut din starea de "repaus" în starea "gata de execuție" (este activat).

Primitiva WAIT trebuie apelată numai de către procese sincrone, iar primitiva REACTIV trebuie apelată numai de către procese asincrone.
Primitivele EXIT şi ACTIV(p) pot fi apelate atât de către procesele sincrone, cât şi de cele asincrone.

Limitări ale programului de simulare

1. După apelul uneia dintre primitivele WAIT, REACTIV, EXIT, nu se mai permite un alt apel de primitivă.
2. Primitiva ACTIV(p) poate fi apelată de mai multe ori şi poate fi urmată de apelul uneia dintre primitivele WAIT, REACTIV, EXIT.
3. Primitivele WAIT, REACTIV, ACTIV(p), EXIT sunt funcționale numai dacă sunt apelate dintr-un proces care este planificat să ocupe integral o cuantă de timp.
Dacă un proces este pus în execuție ca urmare a uneia dintre primivele WAIT sau REACTIV şi conține el însuşi un apel al primitivelor WAIT, REACTIV, ACTIV(p), EXIT, acestea nu vor fi funcționale.

Observații:
1. Pentru a avea controlul asupra cuantei de timp în care un proces apelează o primitivă, s-au introdus contorii CNT[i], i=1,..,10; fiecare proces trebuie să-şi incrementeze contorul asociat şi să testeze acest contor înainte de apelul primitivelor.
De exemplu:

```
Proces i:       CNT[i]:=CNT[i] + 1;
                IF CNT[i]=N THEN "PRIMITIVA";
```

Aceasta va avea ca efect execuția primitivei la cuanta N, alocată procesului i.

2. Întreruperile periodice sunt simulate prin apasarea tastei "SPACE". Pentru modul de lucru automat este necesară acționarea tastei "SPACE" o singură data, la început.

3. La introducerea numărului de cuante pentru fiecare proces activ, dacă suma cuantelor alocate proceselor este mai mare decât numărul de cuante pe ciclu, atunci se alocă câte o cuantă pentru fiecare proces, cu excepția ultimului, căruia i se alocă restul de cuante. Dacă suma cuantelor alocate proceselor este mai mică decât numărul de cuante pe ciclu, diferența de cuante se alocă ultimului proces.

4. Trebuie îndeplinită condiția:

(număr cuante/ciclu) * (număr cicluri) <= 15.

În caz contrar, valoarea introdusă prima (dintre nr.cuante/ciclu şi nr.cicluri) se păstrează, cealaltă valoare fiind calculată astfel încât să se respecte condiția enunțată.

5. Dacă un proces sincron este în repaus şi planificatorul trebuie să-l pună în execuție, atunci trebuie executat procesul de bază (5).

Pentru utilizarea programului, a se vedea şi opțiunea "HELP".