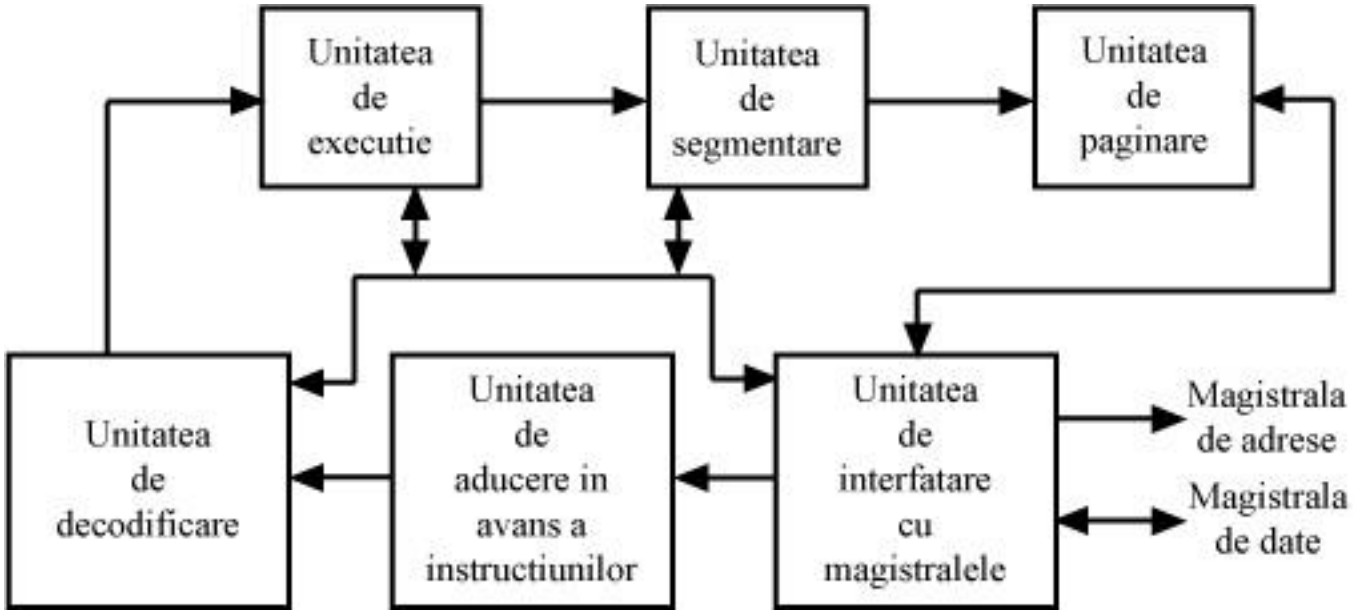
Curs 3

2.2. Microprocesorul 80386

- Este primul microprocesor pe 32 biţi, atât la nivelul magistralei de date cât şi la nivelul registrelor interne;
- Conţine un mecanism de gestiune a memoriei performant care preia de la microprocesorul anterior al familiei conceptul şi soluţia segmentării memoriei şi îi adaugă conceptul şi soluţia paginării memoriei; toate microprocesoarele din familie care i-au urmat au preluat acest mecanism de gestiune a memoriei;
- Viteza a crescut nu numai prin creşterea frecvenţei tactului dar şi prin îmbunătăţiri structurale, ca de exemplu: creşterea numărului blocurilor interne care lucrează independent şi ciclurile maşină cu generarea în avans a adreselor;
- Include facilităţi de depanare şi autotestare;
- Creşterea capacităţii de memorie gestionabilă: 64 To/ task memorie virtuală ce poate fi mapată în 4 Go memorie fizică.

Proiectarea cu Microprocesoare

- Structura internă



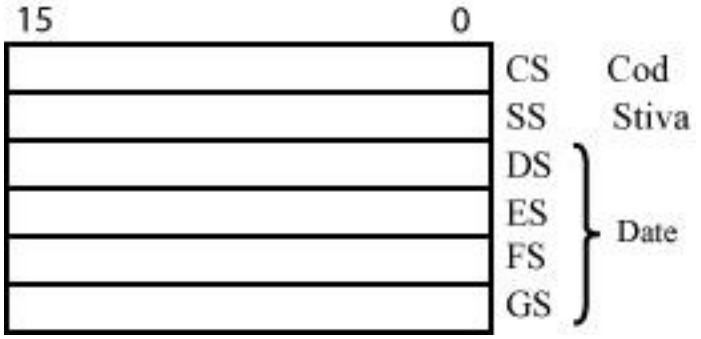
Proiectarea cu Microprocesoare

- **Unitatea de interfaţare la magistrală** asigură legătura microprocesorului cu exteriorul; este solicitată de toate celelalte unităţi;

- **Unitatea de aducere în avans a instrucţiunilor** implementează o facilitate existentă şi la microprocesorul 8086: atunci când nu se execută vreun acces la magistrală, unitatea aduce în avans instrucţiuni şi le depune într-un şir de instrucţiuni, de 16 octeţi, economisindu-se astfel timpul de aducere a codului instrucţiunii;
- **Unitatea de decodificare a codului instrucţiunii** extrage codurile de instrucţiune din unitatea de aducere în avans a instrucţiunilor şi le decodifică; necesită o perioadă de tact pentru a decodifica un octet de cod;
- **Unitatea de execuţie** cuprinde unitatea aritmetică şi logică, setul de registre şi dispozitivul de comandă şi control care conduce întreaga activitate a microprocesorului;

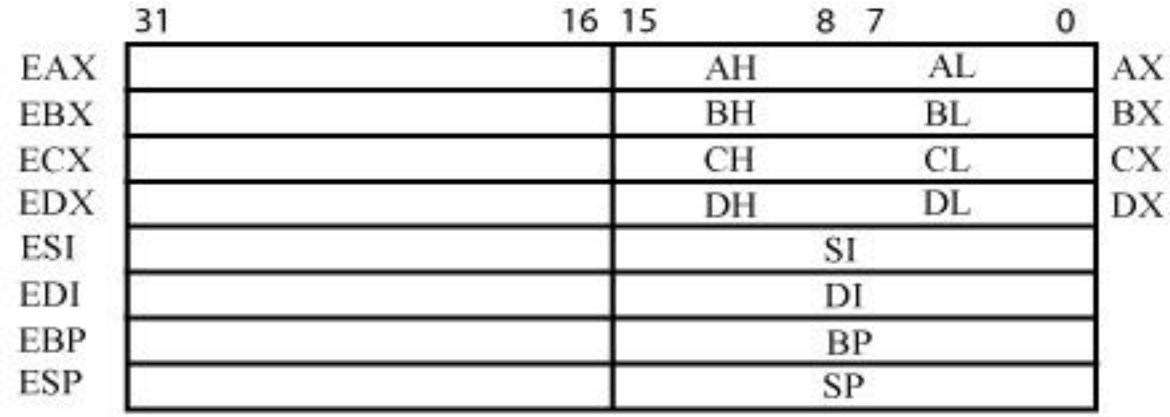
- **Unitatea de segmentare** translatează adresa logică în adresă liniară şi realizează protecţia;
- **Unitatea de paginare** preia, dacă este activată, adresa liniară furnizată de unitatea de segmentare şi o transformă în adresă fizică; dacă nu este activată, adresa liniară devine adresă fizică; cuprinde o memorie cache pentru paginare, în scopul creşterii vitezei.

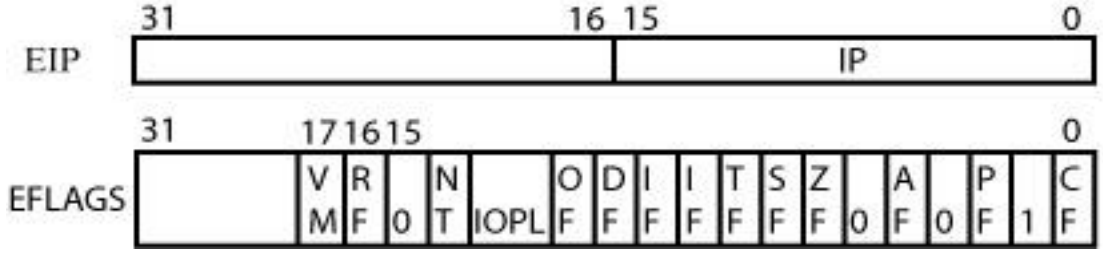
- Setul de registre:
  - registre de segment,
  - registre generale şi
  - registre sistem.
- Registrele de segment



Proiectarea cu Microprocesoare

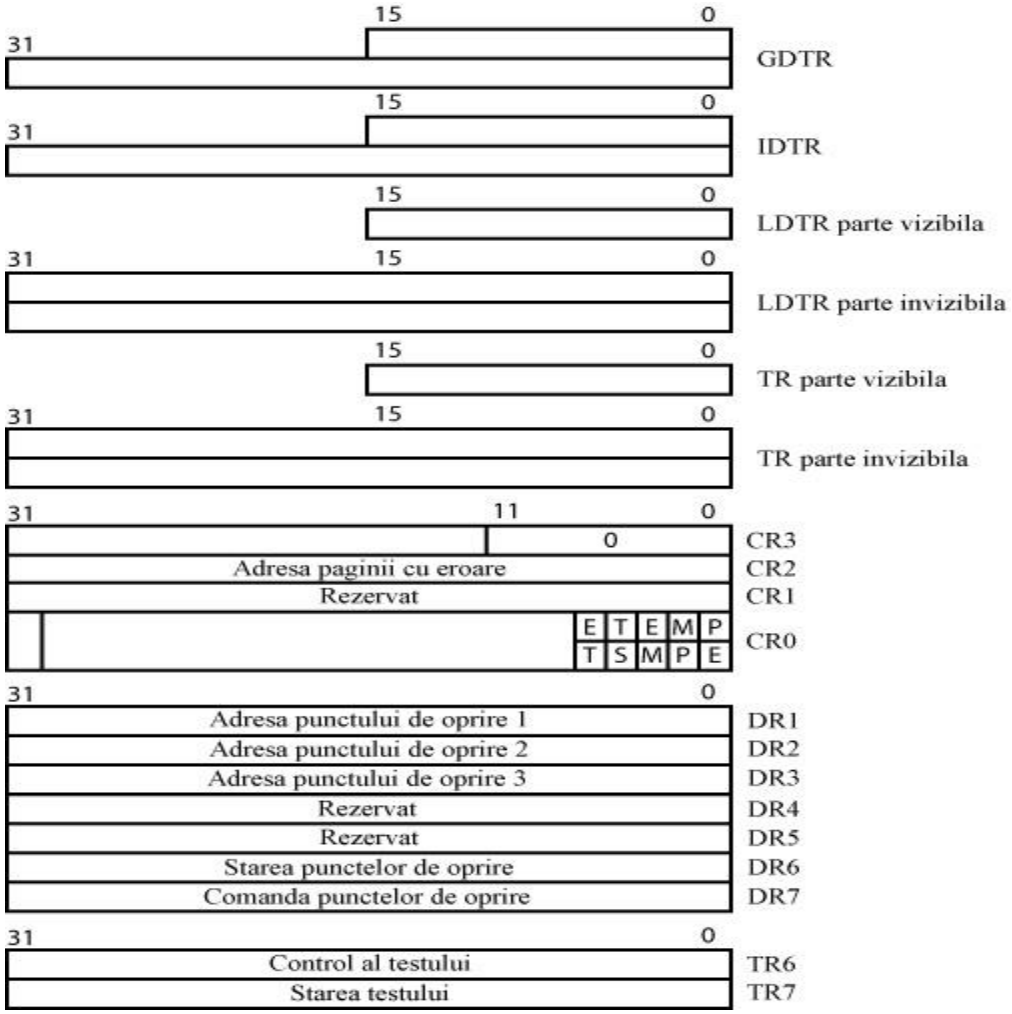
- Registrele generale





Proiectarea cu Microprocesoare

- Registrele sistem



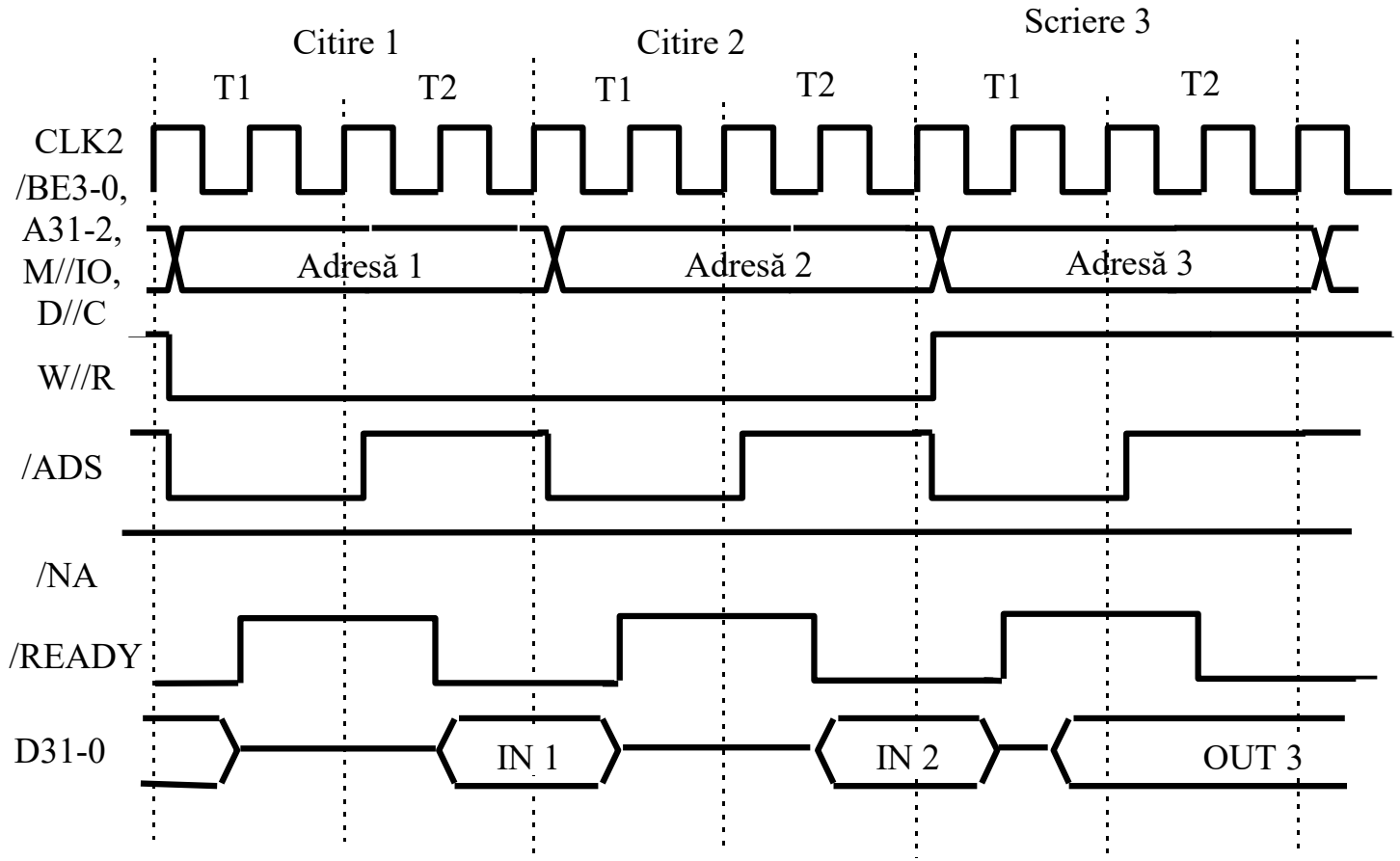
Proiectarea cu Microprocesoare

- Ciclurile maşină
  - tipuri de cicluri

M//IO	D//C	W//R	Tip de ciclu
0	0	0	Acceptare a cererii de întrerupere
0	0	1	Nu apare
0	1	0	Citire de la porturi
0	1	1	Scriere la porturi
1	0	0	Aducere cod de instrucţiune
1	0	1	Halt sau Shutdown
1	1	0	Citire date din memorie
1	1	1	Scriere date in memorie

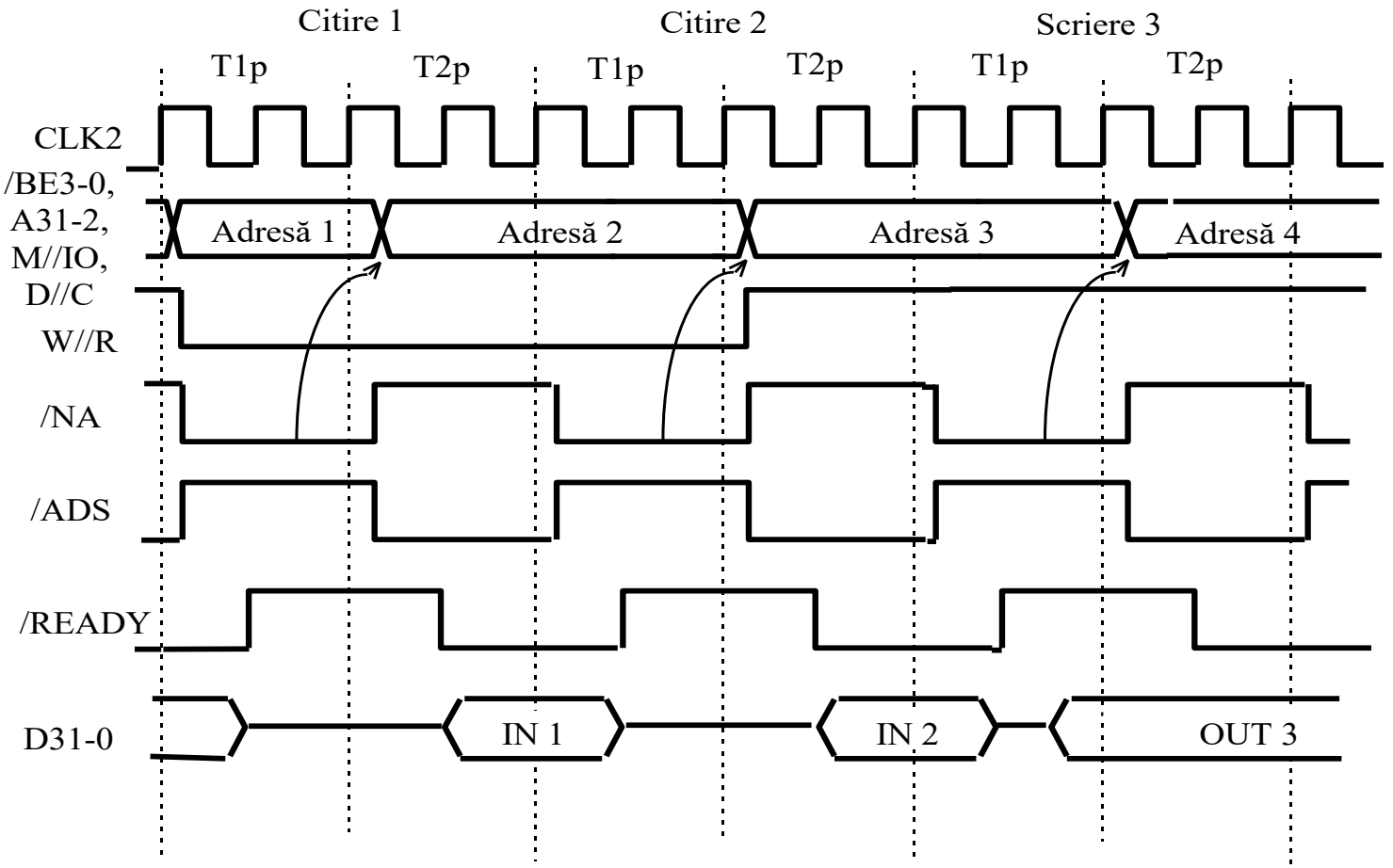
Proiectarea cu Microprocesoare

- Cicluri normale şi cu generare în avans a adreselor
  - succesiune de cicluri normale



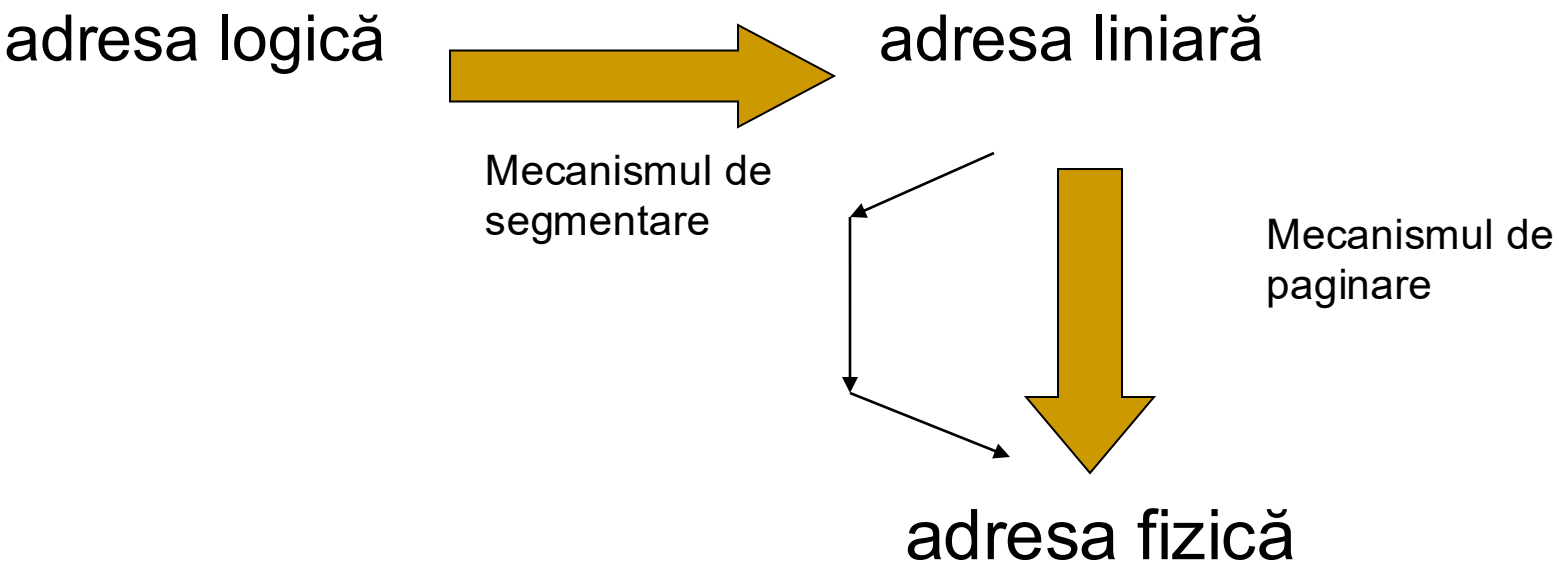
Proiectarea cu Microprocesoare

- succesiune de cicluri cu generare în avans a adreselor



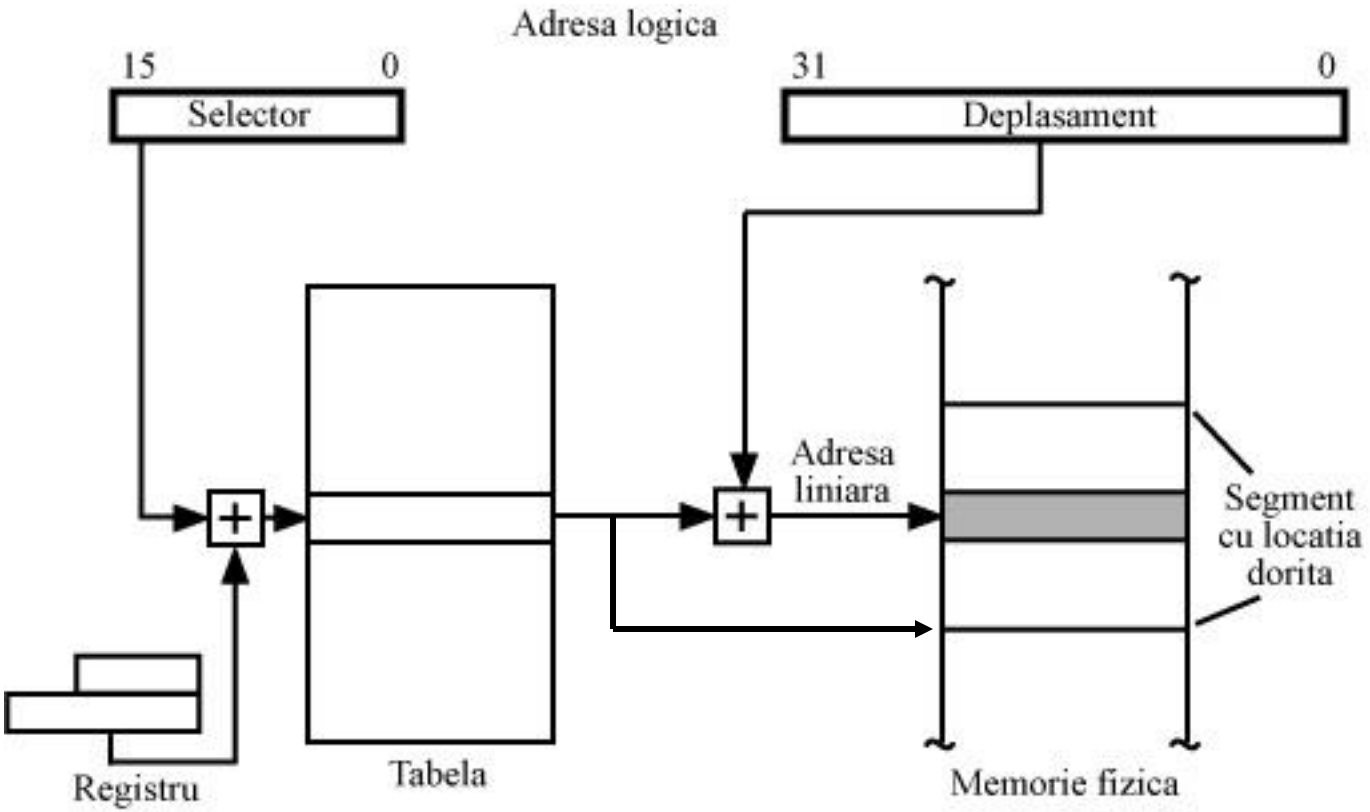
- Organizarea şi gestionarea memoriei
  - Modul real:
    - emulează microprocesorul 8086, ca urmare va avea aceeaşi arhitectură de bază dar va permite şi accesul la registre pe 32 biţi;
  - Modul protejat:
    - oferă performanţe deosebite şi suport hardware complet şi rapid pentru implementarea sistemelor multitasking;
    - asigură comutarea rapidă a taskurilor, separarea şi protecţia zonelor de memorie proprii taskurilor, între ele precum şi separarea şi protecţia zonelor de memorie proprii sistemului de operare;
    - oferă un mecanism de privilegii, pe 4 nivele, care ordonează zonele de memorie şi împiedică accesele neautorizate la memorie;
    - orice acces la memorie este posibil doar respectând un set de reguli bine determinat care include controale de tip de acces, controale de drepturi de acces şi controale de nivel de privilegiu;
    - permite ca fiecărui task să îi fie alocaţi 64 To memorie virtuală şi această memorie virtuală va putea fi mapată în doar 4 Go memorie fizică.

- În modul protejat microprocesorul are 3 spaţii de adrese:
  - logice, date de programator,
  - liniare, obţinute din mecanismul de segmentare şi
  - fizice, obţinute din mecanismul de paginare.



Proiectarea cu Microprocesoare

- Mecanismul de segmentare

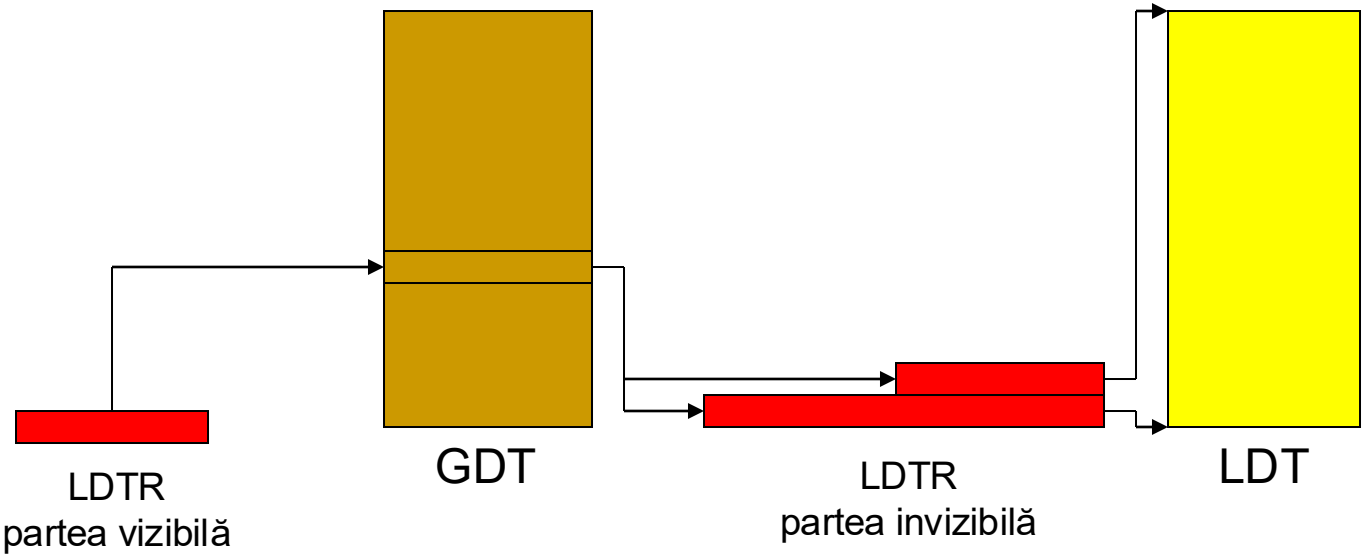


- Adresa logică = Deplasament + Registru selector;
- Deplasament: 32 biţi, furnizat de instrucţiune;
- Registru selector = Registru de segment;
- Conţinutul registrului selector: index în Tabela;
- O intrare în Tabel = grup de 8 octeţi numit descriptor;

- 3 tipuri de tabele cu descriptori:
  - Tabela Descriptorilor Globali (GDT) – comună tuturor taskurilor;
  - Tabela Descriptorilor Locali (LDT) – proprie unui task;
  - Tabela Descriptorilor de Întreruperi (IDT) – comună tuturor taskurilor;

Proiectarea cu Microprocesoare

- GDTR (Global Descriptor Table Registers) indică adresa de bază şi limita GDT;
- IDTR (Interrupt Descriptor Table Registers) indică adresa de bază şi limita IDT;
- LDTR (Local Descriptor Table Registers): microprocesorul vede LDT – urile ca segmente de sine stătătoare;



- Structura unui selector de segment:

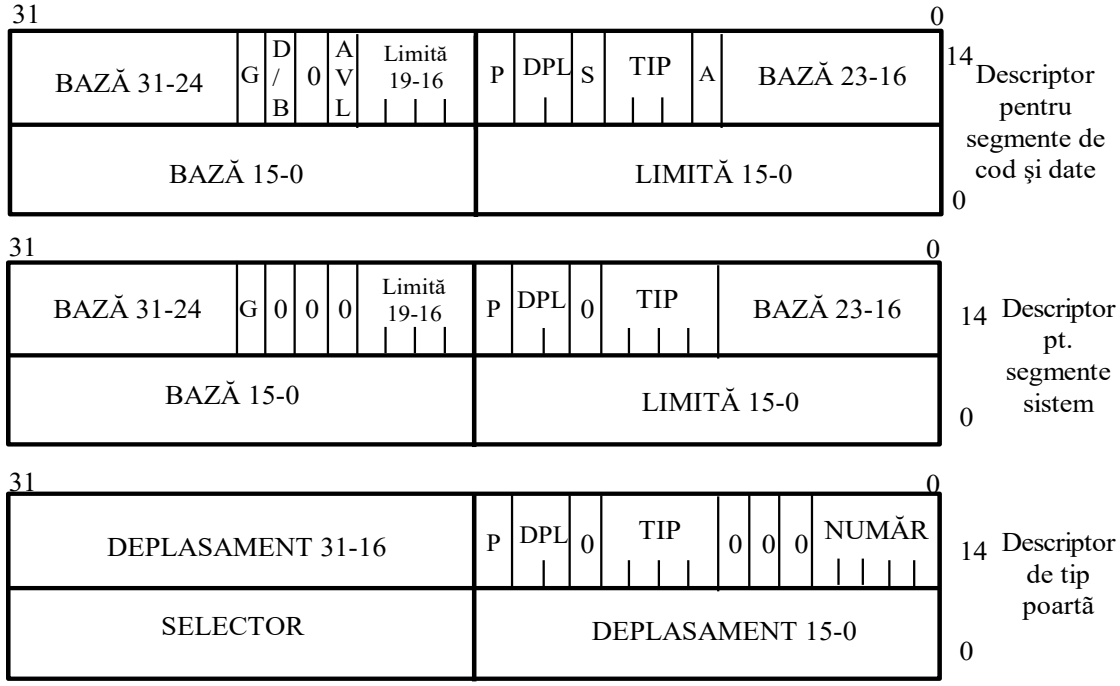
- INDEX = 13 biţi → GDT sau LDT are 2^{13} = 8192 intrări a câte 8 octeţi → o tabelă are 8 – 65536 octeţi;

- Un task poate accesa maxim:

8192 segmente comune + 8192 segmente proprii = 16384 segmente.

Proiectarea cu Microprocesoare

- Descriptor:
  - pentru segmente de cod şi date;
  - pentru segmente sistem (pentru LDT și TSS), un tip aparte fiind descriptorii de tip poartă;
  - structura descriptorilor:



- P ("Present"): bitul de prezenţă; 1 înseamnă ca segmentul este în memoria fizică iar 0 înseamnă ca segmentul nu se găseşte în memoria fizică deci o încercare de acces la el va provoca o excepţie;

- DPL ("Descriptor Privilege Level"): este nivelul de privilegiu al segmentului;
- S ("Segment Descriptor"): indică tipul descriptorului; 1 pentru segmente de cod şi date şi 0 pentru segmente sistem;
- G (“Granularity”): unitatea de măsură este octetul sau pagina de 4 Ko;
- TIP: se interpretează diferit funcţie de tipul segmentului;

- A ("Accessed"): segmentul a mai fost, 1, sau nu a mai fost, 0, accesat; este folosit de mecanismul de memorie virtuală pentru a selecta segmentul care va fi eliminat din memoria fizică; sistemul de operare va şterge acest rang după fiecare accesare a segmentului, contorizând simultan numărul de accesări ale segmentului; în acest fel sistemul de operare poate afla numărul de accesări ale fiecărui segment într-o perioadă de timp determinată; dacă segmentul care se elimină din memoria fizică este ales după criteriul "cel mai puţin folosit" într - o anumită perioadă, atunci contorizările realizate de sistemul de operare indică rapid segmentul ce va fi eliminat.

- Descriptorii pentru segmente de cod şi date: câmpul TIP:
  - E = 1 - descriptor pentru segment de cod:
  - C ("Conforming"): bit de conformitate, oferă posibilitatea ca un segment de cod să fie accesat din segmente de cod mai puţin privilegiate; dacă C = 1 atunci segmentul se numeşte conform şi poate fi accesat din segmente de cod mai puţin privilegiate;
  - R ("Readable"): 0 înseamnă că segmentul nu poate fi citit ci doar executat iar 1 înseamnă că segmentul poate fi şi citit şi executat; un segment de cod ce poate fi şi citit trebuie folosit atunci când conţine şi constante şi instrucţiuni; acest rang poate fi folosit şi la protecţie contra furturilor de cod;
  - E = 0 - descriptor pentru segment de date sau stivă:
  - ED ("Expansion Direction"): specifică dacă segmentul este expandabil spre adrese mai mici (cazul segmentelor de stivă), 1 sau spre adrese mai mari (cazul segmentelor de date), 0; prin acest mecanism se stabileşte până unde se poate întinde un segment de date/ stivă;
  - W ("Writeable"): specifică dacă segmentul respectiv este protejat, 0, sau nu, 1, la scriere.

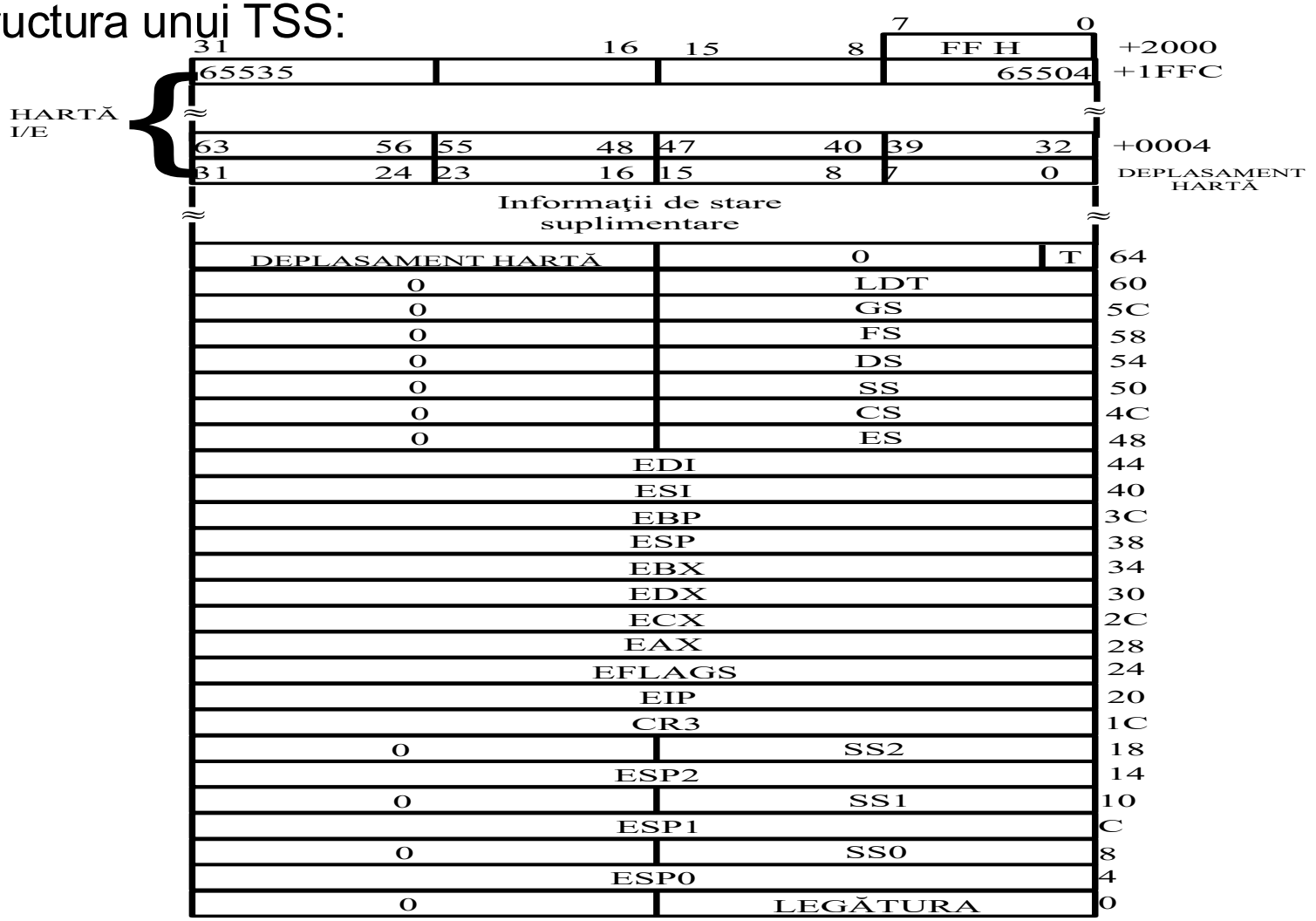
Proiectarea cu Microprocesoare

- Dezavantaj al mecanismului de segmentare:
  - citirea unui descriptor adică 2 cicluri suplimentare; soluţia: registre cache pentru descriptori.
- Avantaje ale mecanismului de segmentare:
  - facilitează implementarea sistemelor multitasking prin asignarea segmentelor de cod şi date propri,
  - facilităţile de protecţie oferite de microprocesor asigură separarea segmentelor corespunzătoare unui task de cele corespunzătoare altui task şi de cele corespunzătoare sistemului de operare; în acest fel pot rula concurent mai multe taskuri, izolate unul de celălalt, fiecare având acces doar la propriile resurse şi izolate de resursele sistemului de operare,
  - un program oarecare nu poate modifica resursele sistemului de operare şi o eroare dintr-un program nu poate afecta decât propriile resurse,
  - în cazul existenţei unei erori într-un program care duce la abandonarea sa, alte programe pot rula pentru a diagnostica şi, eventual corecta, eroarea;
  - verificările care asigură protecţia sunt realizate de microprocesor înainte de fiecare acces la memorie, în timpul translatării adresei, deci nu necesită timp suplimentar; verificări:
    - de tip: verificarea restricţiilor impuse de octetul de atribute din descriptor,
    - de limită: împiedică un acces în afara segmentului,
    - de puncte de intrare în segmentul destinaţie,
    - de nivele de privilegiu:
      - 4 nivele de privilegiu: 0 (cel mai privilegiat) – 3 (cel mai puţin privilegiat),
      - nivelul 0 se alocă nucleului sistemului de operare iar nivelul 3 programelor aplicative.

- Transferul controlului în alt segment de cod (intersegment) se realizează prin încărcarea registrului CS utilizând o instrucţiune de tip JMP sau CALL intersegment şi în cazul comutării taskurilor;
- Comutarea taskurilor înseamnă transferul controlului într-un alt segment de cod, împreună cu salvarea stării curente a microprocesorului şi încărcarea noii stări a microprocesorului;
- Fiecărui task îi corespunde un segment special, Segment Stare Task, TSS, care conţine toate informaţiile necesare actualizării stării microprocesorului şi revenirii în taskul de unde s-a făcut comutarea:
  - indicatorii de stivă şi valorile selectorilor pentru stivele corespunzătoare nivelelor de privilegiu 0, 1 şi 2,
  - conţinuturile registrelor generale, inclusiv ale registrelor EIP şi EFLAGS; registrul EIP conţine punctul de intrare în taskul corespunzător,
  - selectorul pentru LDT al taskului respectiv,
  - legătura cu taskul apelant,
  - harta de intrare/ ieşire,
  - câmp pentru informaţii suplimentare la dispoziţia sistemului de operare sau a programelor aplicative.

Proiectarea cu Microprocesoare

- Structura unui TSS:



- Fiecărui task îi corespund segmente de date proprii ale căror descriptori se găsesc într-o LDT proprie; aceasta asigură o izolare a datelor corespunzătoare taskurilor; selectorul care va fi încărcat în LDTR, partea vizibilă, va fi preluat din TSS;
- Câmpul HARTĂ I/E completează protecţia la operaţiile de I/E:
  - verificările menţionate,
  - verificarea de nivel de privilegiu,
  - biţii din HARTĂ I/E.

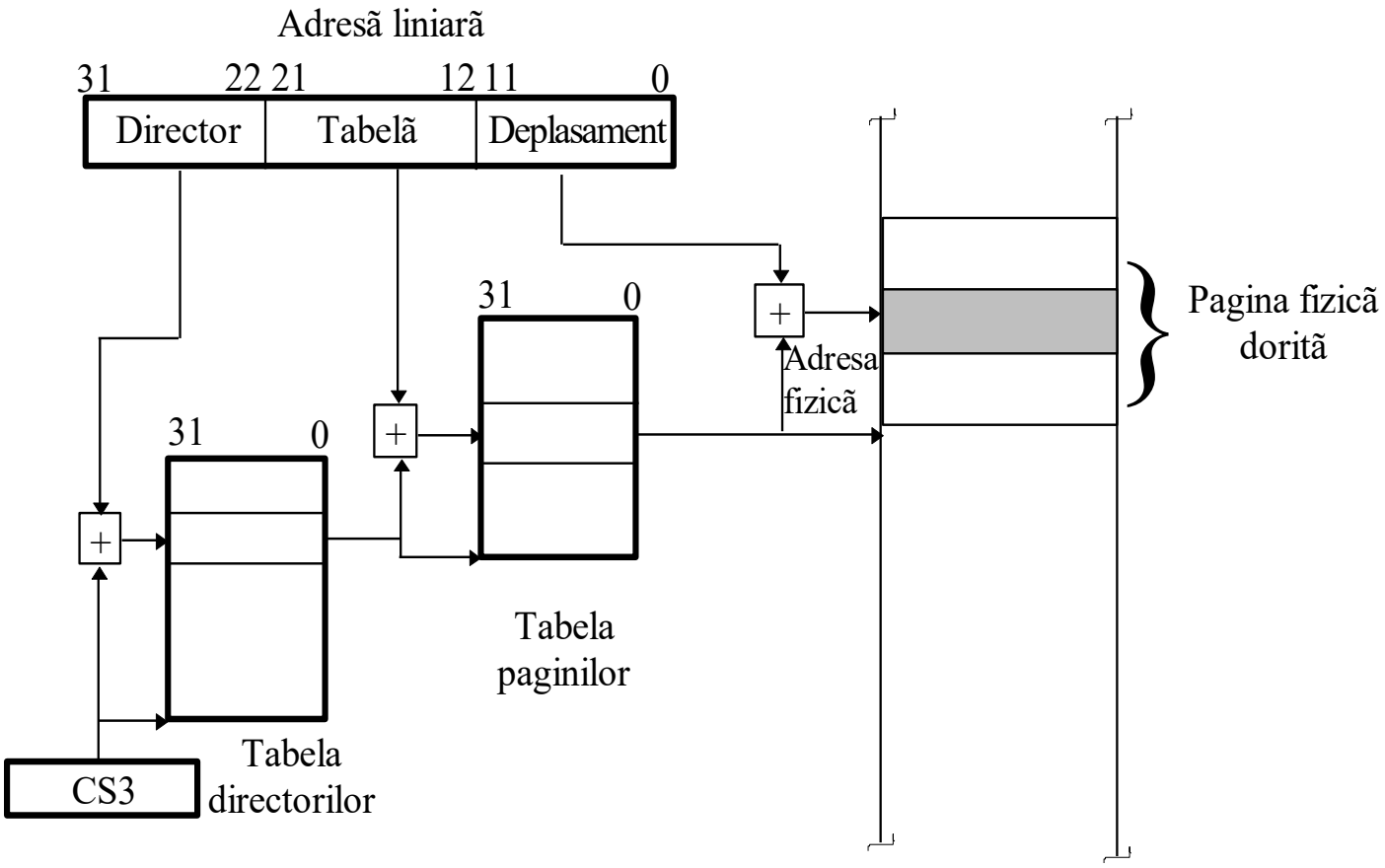
- Adresa de început şi limita unui TSS se obţin dintr-un descriptor pentru un TSS, prezent în GDT;
- Accesul la descriptor se face printr-un selector conţinut în registrul de task, TR, în partea vizibilă a sa;
- În partea invizibilă a sa se încarcă, din descriptor, adresa de bază şi limita Segmentului Stare Task, realizându-se accesul la acesta.

- Operaţiile executate de microprocesor la comutarea taskurilor sunt următoarele:
  - se memorează valorile curente pentru registrele generale, EAX, EBX, ECX, EDX, ESP, EBP, ESI, EDI, CS, SS, DS, ES, FS, GS, EIP şi EFLAGS în TSS activ şi acesta este trecut în stare de liber;
  - se încarcă registrul de task, TR, partea vizibilă, cu valoarea corespunzătoare noului task şi se trece noul task în starea activ;
  - se încarcă registrele şi LDTR, partea vizibilă cu valorile din noul TSS; se încarcă şi partea invizibilă a LDTR;
  - se intră în noul task în punctul CS:EIP.

- Mecanismul de paginare:
  - a apărut ca urmare a dimensiunilor mari ale segmentelor (până la 4 Go), în scopul gestionării mai eficiente a lor; priveşte segmentul ca fiind alcătuit din pagini fixe de 4 Ko şi oferă facilităţi suplimentare de protecţie;
  - activarea / dezactivarea mecanismului de paginare se face prin setarea / resetarea rangului PG, cel mai semnificativ, din registrul CR0;
  - mecanismul de paginare preia adresa generată de cel de segmentare, adresa liniară, şi o transformă în adresă fizică;
  - dacă mecanismul de paginare nu este activat, atunci adresa liniară devine adresă fizică;

Proiectarea cu Microprocesoare

- Structura mecanismului de paginare:

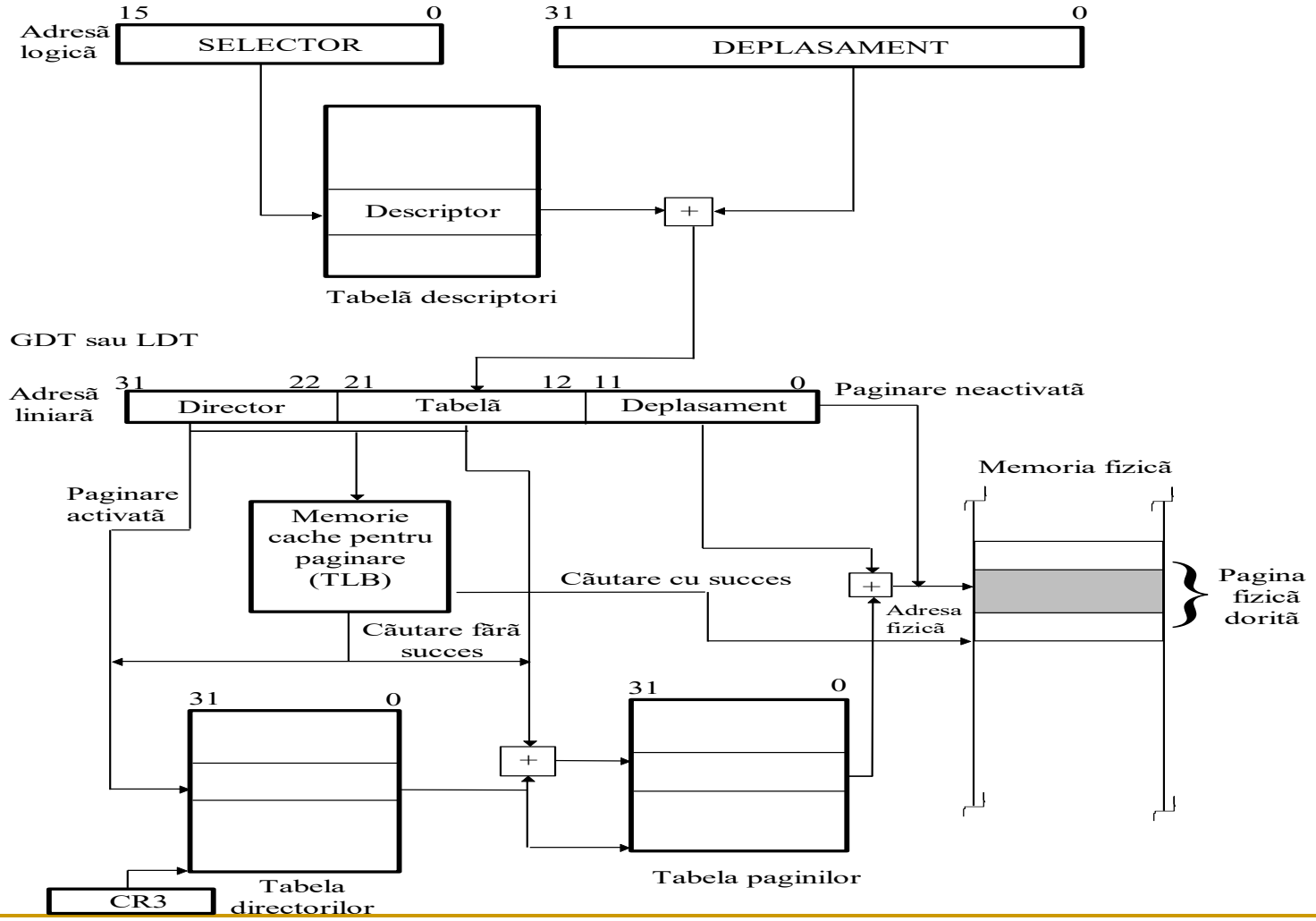


- Avantaje:
  - se pot gestiona 1024 x 1024 = 1 M pagini a câte 4 Ko;
  - tabelele sunt ele însele pagini de câte 4 Ko;
  - cere zone de memorie compacte de dimensiuni mici: 4 Ko faţă de un mecanism pe un singur nivel, cu pagini de 4 Ko, pentru care ar fi fost necesară o tabelă cu 1 M intrări şi considerând 4 octeţi/ intrare ar fi fost necesară o zonă de memorie compactă de 4 Mo.

- Dezavantaje:
  - necesitatea execuţiei a două accese suplimentare la memorie pentru fiecare acces dorit, ceea ce înseamnă timp suplimentar;
  - soluţia: prevederea în microprocesor a unei memorii cache pentru paginare, cunoscută sub denumirea de TLB;
  - conţine adresele celor mai recent folosite 32 pagini;
  - tabela are 32 intrări şi poate acoperi 128 Ko de memorie.

Proiectarea cu Microprocesoare

- Adresarea memoriei la microprocesorul 80386:



- Sistemul de întreruperi:
  - întreruperi: provocate de evenimente externe şi instrucţiuni INT nn;
  - excepţii: provocate de evenimente interne;
  - 3 grupe:
    - erori ("faults"): o eroare este o excepţie detectată înaintea execuţiei instrucţiunii care a provocat-o; exemple sunt: violare de privilegii, segment sau pagină lipsă din memoria fizică etc.; adresa de revenire din rutina de tratare este adresa instrucţiunii în timpul căreia s-a detectat excepţia deci are loc o restartare a instrucţiunii;
    - traps: este un tip de excepţie luată în considerare la sfârşitul execuţiei instrucţiunii care a provocat-o; exemple sunt: eroare la divizare, ieşire din domeniu; adresa de revenire din rutina de tratare este adresa instrucţiunii următoare instrucţiunii în timpul căreia s-a detectat excepţia;
    - abandonuri: apare ca urmare a unor situaţii deosebite, catastrofale, de exemplu o eroare hardware; nu se cunoaşte instrucţiunea care a determinat abandonul şi nu se poate asigura nici continuarea execuţiei programului;
  - tratare în mod real şi în mod protejat;
  - Tabela vectorilor: în mod protejat conţine descriptori speciali, numiți porţi, de tip întrerupere sau trap şi este IDT.

Proiectarea cu Microprocesoare

- Accesul la rutina de tratare a unei întreruperi sau excepţii printr-o poartă de tip întrerupere sau trap: