UNIUNEA EUROPEANĂ
GUVERNUL ROMÂNIEI
Instrumente Structurale 2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Programare în limbaj de asamblare

9. Descriptori de segment.

Descriptori de segment

CPU face în mod automat referire la tabele ori de câte ori un registru segment este încărcat cu un selector. Toate instrucțiunile care încarcă un registru segment vor face referire la tabelele din memorie, fără soft suplimentar. Aceste tabele conțin valori de 8 octeți, denumiți descriptori. Există descriptori de segment pentru segmentele de cod, stivă, date, descriptori de control ai sistemului pentru segmente de date speciale şi operații de transfer al controlului (comutare de task).

Tabela descriptorilor segment face asocierea între un selector şi un segment fizic din memorie, convertind astfel adresa virtuală într-o adresă fizică. Pe baza tabelei descriptorilor TD, procesorul determină poziția în memoria fizică a segmentului corespunzător unui anumit selector. Unele intrări din tabela DT sunt marcate ca *„segment neîncărcat“*. Pentru aceste intrări din DT, procesorul generează o întrerupere. Rutina de tratare a întreruperii încarcă segmentul de pe disc în memorie, permițând apoi accesul la informația din acel segment. Un segment care nu mai este folosit este salvat pe disc de către sistemul de operare. Sistemul realizează operația de interschimb de segmente între memorie şi disc. Pentru a gestiona eficient memoria, sistemul de operare păstrează în memorie segmentele cel mai recent utilizate. Pentru celelalte segmente, marchează intrările în tabela DT ca *„segment neîncărcat“*, utilizând spațiul disponibil pentru alte segmente ce urmează a fi încărcate.

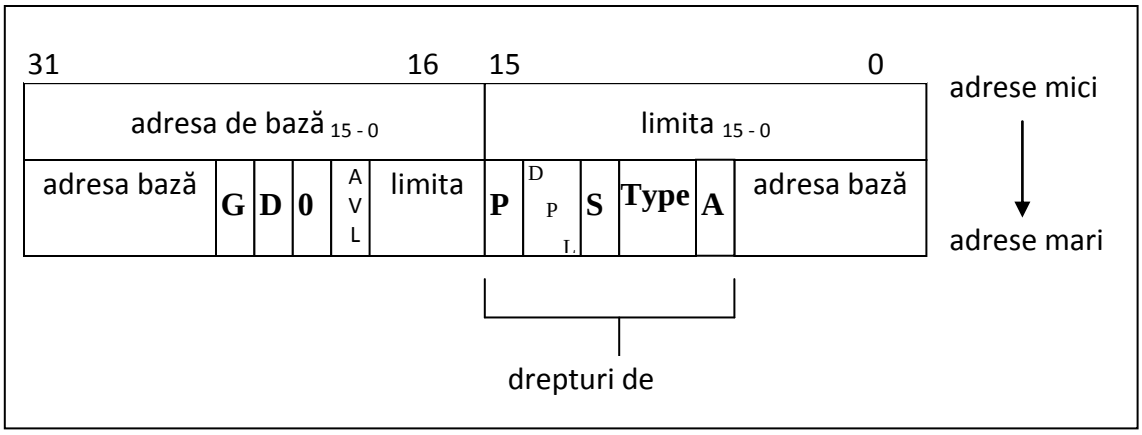
Descriptorii de segment de date şi cod conțin, pe lângă adresa de bază a segmentului, şi alte atribute de segment, incluzând:

- dimensiunea segmentului (1 ÷ 64 Ko);
- drepturi de acces (read only, read/write, execute only, execute/read);
- prezența în memorie (pentru sistemele cu memorie virtuală).

Segmentele de memorie reprezintă un tip de obiecte sistem. Alte obiecte sistem includ tabelele ce suportă mecanismul de protecție, segmente speciale ce memorează starea procesorului şi obiecte de control al accesului, denumite porți (*gates*). Descriptorii sunt grupați în tabele de descriptori. Examinând selectorul, CPU determină descriptorul asociat selectorului şi obiectul referit de descriptor. Într-un câmp al descriptorului se memorează nivelul de privilegiu al obiectului (DPL). Când programul solicită accesul la un obiect sistem, cu un selector, pot să apară următoarele situații:

- accesul nu este permis; dacă cererea violează o regulă de protecție, controlul este transferat de la program la o rutină din sistemul de operare care, de obicei, sistează execuția procesului;
- accesul este permis, dar imposibil de recunoscut; de exemplu, dacă obiectul nu este, curent, în memorie, atunci este apelată o rutină a sistemului de operare pentru a aduce obiectul în memorie (operație denumită „interschimb“ – swapping), şi returnează controlul programului. Este permis, apoi, programului să reîncerce accesul la obiect;
- accesul este recunoscut, la nivelul de privilegiu cerut (RPL);

Structura unui descriptor de segment este prezentată în figură.



Structura generală a unui descriptor de segment (sistem, memorie sau poartă)

Câmpurile din structura unui descriptor de segment sunt descrise în continuare.

Adresa de bază reprezintă adresa de start a segmentului, față de offsetul 0; ea este o adresă de 32 de biți, reprezentată de octeții 2, 3, 4 şi 7 din descriptor.

Limita determină ultima unitate adresabilă a segmentului; acest câmp are 20 de biți, alcătuit din octeții 0 şi 1 ai descriptorului, şi ultimii 4 biți ai octetului 6 din descriptor. Limita poate fi definită în octeți sau în entități mai mari, denumite pagini, de 4 Ko. Aceasta poate conduce la o dimensiune maximă de 1 Mo, în primul caz, şi 4 Go, în cel de-al doilea caz.

Bitul *G* reprezintă granularitatea sau rezoluția în care este definită limita unui segment. Limita unui segment cu rezoluție (granularitate) octet, este măsurată în octeți; un segment cu rezoluția pagină este măsurat în entități mai mari, denumite pagini. O pagină cuprinde 2^{12} = 4 Kocteți. Aceasta conduce la o limită a dimensiunii unui segment de 2^{20} pagini de câte 4 Ko, deci în total 4 Go.

Bitul *P (present)* este pus pe 1, când segmentul indicat de selector este prezent în memoria fizică. Într-un sistem cu memorie virtuală, sistemul de operare poate muta conținutul unor segmente pe disc, dacă memoria fizică este plină. Atunci, sistemul de operare marchează descriptorul ca „not-present“, prin resetarea bitului P, pe 0. Dacă o aplicație încarcă un selector într-un registru segment şi descriptorul asociat cu selectorul are P=0, se generează o întrerupere „not-present“ (11-nivelul/tipul întreruperii). Sistemul de operare caută apoi o zonă de memorie fizică liberă, copiază conținutul segmentului de pe disc înapoi în memorie, actualizează descriptorul cu noua adresă de bază, setează P=1, şi reîncepe instrucțiunea întreruptă.

DPL – privilegiul unui descriptor: 0 – cel mai privilegiat, 3 – cel mai puțin privilegiat.

Un task poate obține acces la segmente cu acelaşi privilegiu sau mai mic. El poate citi/scrie date în segmente cu privilegiu egal sau mai mic. Un task poate apela segmente de cod de acelaşi privilegiu. Totuşi, accesele la segmente cu privilegiu mai mare pot fi recunoscute, indirect, prin porți (o caracteristică a mecanismului de protecție). Un task nu poate obține niciodată un acces la un segment de cod cu privilegiu mai mic.

Nivelul de privilegiu al unui task, numit CPL (Current Privilege Level) este nivelul de privilegiu al segmentului de cod ce se execută. De obicei, cele mai importante porțiuni ale sistemului de operare rulează la nivelul 0. Alte componente de sistem pot rula la un nivel mai puțin privilegiat, iar aplicațiile rulează, tipic, pe nivelul 3.

Bitul *A (accessed)* este setat când selectorul pentru descriptor este încărcat într-un registru segment. Sistemul de operare poate utiliza acest bit pentru a determina care segmente nu sunt frecvent utilizate şi pot fi transferate pe disc, dacă este necesar.

Bitul 6, din câmpurile adiționale, este denumit bitul *D* sau *B*:

- *D (default)*, dacă descriptorul este pentru un segment executabil; D=1 indică modul

nativ, adică setul implicit de instrucțiuni, iar D=0, un segment de cod 286, care rulează cu offseturi de 16 biți şi setul de instrucțiuni de la 286.

- *B (big)*, dacă descriptorul este pentru un segment de date; B=1, pentru orice segment de date a cărui dimensiune este mai mare de 64 Ko.

AVL poate fi utilizat pentru:

- marcarea de segmente „inutile“ (garbage collection);
- indicarea segmentelor ale căror adrese de bază nu trebuie modificate.

Semnificația biților din octetul de acces este următoarea:

- P – (Present); dacă este 1, segmentul este mapat în memoria fizică; în caz contrar, nu este mapat în memoria fizică, iar baza şi limita nu sunt utilizate;
- DPL – (Descriptor Privilege Level) este atributul de privilegiu, utilizat în teste de privilegiu;
- S – (Segment Descriptor) specifică tipul de segment definit de descriptor:
  - = 1, este descriptor de segment de cod sau date;
  - = 0, nu este descriptor de segment de cod sau date, ci descriptor de sistem;
- Type – specifică tipul segmentului: de cod sau de date; cei trei biți au următoarea semnificație:
- (3) Executable; dacă este 0 este un descriptor de segment de date, şi în acest caz semnificația biților următori este:
  - (2) Expansion Direction (ED); dacă este 0, segmentul se extinde în sus, adică spre adrese mari (deci pentru date), astfel încât offset<=limita; dacă este 1, segmentul se extinde în jos, adică spre adrese mici (stivă), şi deci offset > limita;
  - (1) Writeable; dacă este 0, nu se poate scrie în acel segment, în caz contrar se poate scrie;
- (3) Executable; dacă este 1, este un descriptor de cod, şi atunci semnificația biților următori este:
  - (2) Conforming; dacă este 1, acest segment poate fi executat numai când CPL >= DPL, adică „în conformitate“ cu criteriile de protecție (privilegiu);
  - (1) Readable; dacă este 0, segmentul nu poate fi citit, adică este numai executabil, altfel el poate fi citit;
- A – (Accessed); dacă este 0, segmentul nu a fost adresat, iar dacă este 1, selectorul de segment a fost încărcat în registrul segment sau utilizat de instrucțiuni de test selector.

Formatele descriptorilor

Sunt definite formate pentru cele trei tipuri de descriptori: program, sistem şi porți. Descriptorii de segment de program au fost descrişi anterior. Descriptorii de segment sistem descriu LDT şi TSS. Nu se pot încărca descriptorii pentru un LDT şi un TSS într-un registru segment şi nici citi sau scrie conținutul sub formă de date.

Pentru ca sistemul de operare să actualizeze un LDT sau TSS, el trebuie să creeze un descriptor de segment de memorie cu aceeaşi adresă de bază şi limită, denumit „alias“. Programele de depanare care permit modificarea segmentelor de cod al programului trebuie să creeze „alias“-uri, deoarece segmentele de cod nu pot fi scrise în condițiile regulilor de protecție ale familiei de procesoare Intel. Descriptorul de memorie (S=1) este descris în manual.

Un descriptor de poartă nu defineşte o zonă de memorie, şi deci nu are câmpurile: adresă de bază şi limită. În schimb, o poartă face referire la un alt descriptor printr-un selector. Porțile de

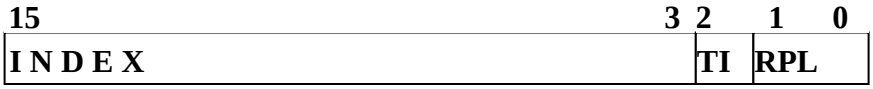
apel, întrerupere şi capcană, trebuie să conțină selectorul pentru un segment de cod şi un offset în segment. Porțile task păstrează un selector pentru un TSS, iar porțiunea de offset a descriptorului este neutilizată.

Descriptorii de poartă, la fel ca şi cei de sistem, au S = 0, şi pot conține, de exemplu, una din valorile (în câmpul „type“):

5 – poartă TSS, 12 (4) – poartă apel 486/386 (286), 14 (6) – poartă întrerupere 486/386 (286), 15 (7) – poartă capcană 486/386 (286).

Tipurile de descriptori 1, 3, 4, 6 şi 7 sunt utilizați la 286. Se pot utiliza descriptori de cod şi date pentru sisteme de 16 biți, într-un sistem de 32 de biți, dar utilizarea descriptorilor de sistem de 16 biți poate crea probleme.

Există, totuşi, o diferență importantă între modul real de adresare şi modul protejat, în ceea ce priveşte formatul actual şi informația conținută de selectoarele de segment. În modul real, selectorul de 16 biți constituie biții cei mai semnificativi ai adresei de bază a segmentului fizic. În schimb, selectorul de segment în modul protejat are următorul format:
Semnificația ultimilor doi biți este următoarea:



Formatul unui Selector

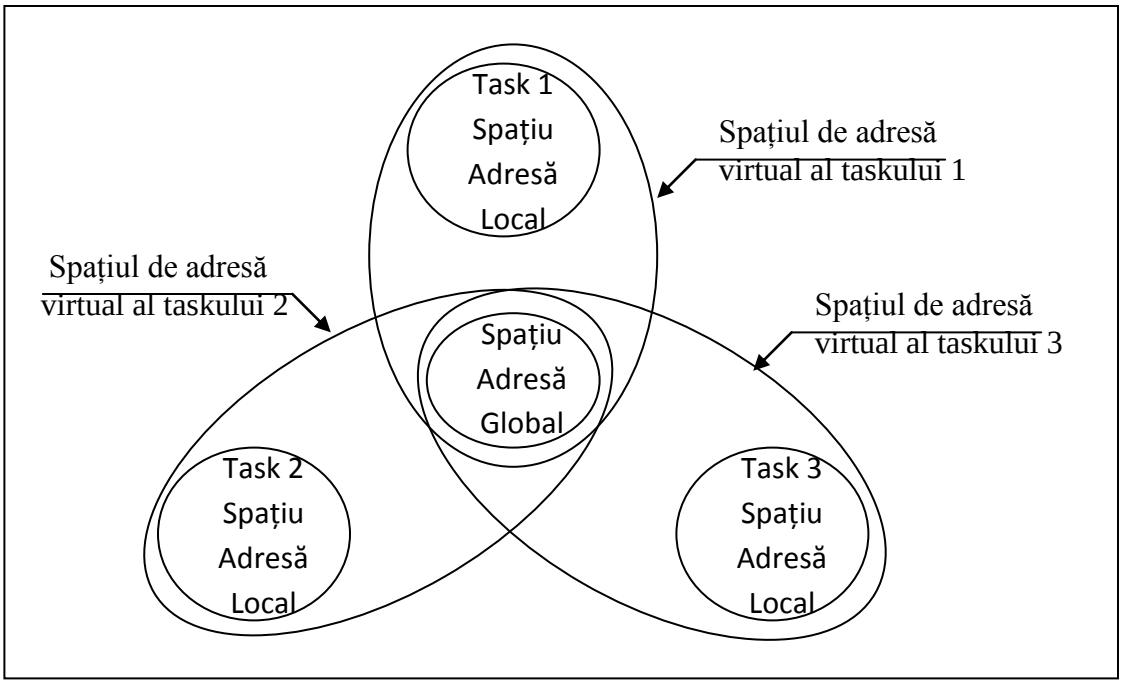
- RPL – (Requested Privilege Level) – nivelul de privilegiu dorit;
- TI – (Table Indicator) – acest bit face selecția între tabela de descriptor globală / locală.

Ceilalți biți (15÷3) selectează intrarea respectivă din tabelă. Deci biții RPL, din câmpul 1÷0, nu sunt implicați în selectarea şi specificarea segmentelor.

Ceilalți 14 biți din componența selectorului desemnează în mod unic un segment particular. Spațiul de adresă virtuală al unui program va fi format din cel mult 2^{14} segmente distincte. Întrucât dimensiunea maximă a unui segment este de 64 Ko (2^{16}), rezultă că spațiul virtual de adresă al unui program poate ajunge la 1 Go ($2^{14}\times2^{16} = 2^{30}$) de locații adresabile individual. La 386/486, întrucât dimensiunea unui segment este de 32 de biți, se ajunge la un spațiu total de adresă virtuală de 64 To ($2^{14} \times 2^{32} = 2^{46}$).

Spațiul total de adresare virtual este ulterior subdivizat în două jumătăți separate, deosebite prin bitul TI. Aceste două jumătăți sunt spațiul global de adrese şi spațiul local de adrese. Spațiul global este utilizat pentru date şi procese de sistem, incluzând softul sistemului de operare, rutine de bibliotecă şi alte servicii de sistem, de obicei partajate. Pentru programele de aplicație sistemul de operare apare ca un set de rutine, de servicii, ce sunt accesibile tuturor task-urilor. Spațiul global este partajat de toate task-urile, pentru a evita duplicate inutile ale rutinelor de servicii de sistem şi pentru a facilita partajarea datelor şi a rutinelor de întrerupere.

Cealaltă jumătate a spațiului virtual este mapată separat pentru fiecare task din sistem şi este cunoscută sub numele de spațiu local de adresă. În general, segmentele de cod şi date dintr-un spațiu local de adrese a unui task sunt specifice task-ului sau utilizatorului respectiv, ca în figura următoare.



Spațiile virtuale de adrese pentru trei taskuri

Tabela GDT (Global Descriptor Table) conține o descriere completă a spațiului global de adrese şi este unică. Tabelele LDT (Local Descriptor Table) conțin descriptori ai spațiului de adrese local fiecărui task. Un task se caracterizează prin perechea de tabele GDT (comună tuturor task-urilor) şi LDT (specific fiecărui task). Tabela GDT, fiind unică pentru toate task-urile, nu are nevoie de descriptor. Adresa de bază şi dimensiunea ei sunt păstrate într-un registru special, GDTR (Global Descriptor Table Register). Analog există un registru LDTR, care memorează adresa de bază şi mărimea tabelei LDT curente asociată cu task-ul în execuție. Registrele GDTR şi LDTR sunt accesibile numai sistemului de operare, prin intermediul unor instrucțiuni specifice.

Fiecare task îşi are propria tabelă LDT, care conține descriptorii segmentelor de cod şi date utilizate. De asemenea, task-ul poate avea propria tabelă a vectorilor de întrerupere (IDT – Interrupt Descriptor Table), care conține descriptorii vectorilor de întrerupere specifici task-ului. Tabela IDT are aceeaşi structură ca tabela DT. Accesul la aceste tabele pentru citire şi scriere se face prin instrucțiuni specifice: LGDT, SGDT, LLDT, SLDT, LIDT, SIDT (Load /Store Global / Local / Interrupt Descriptor Table Register).

Termenul de multitasking se referă la capacitatea de a executa simultan mai multe programe separate (task-uri). Deoarece calculatorul dispune de un singur procesor, task-urile nu se execută în paralel, ci procesorul comută de la un task la altul, creând astfel impresia de execuție simultană. Procesoarele familiei IA, începând de la 286, implementează acest concept prin mijloace hard. Ele asigură protecția task-urilor astfel încât memoria utilizată de un task să nu fie accesibilă unui alt task.

Instrucțiunea ARPL (Adjust Requested Privilege Level) folosită de sistemul de operare modifică nivelul de privilegiu al selectorului destinație astfel încât să coincidă cu acela al sursei.

Descriptorul de segment conține un octet ce specifică drepturile de acces la acel segment. Sistemul de operare poate modifica acest octet folosind instrucțiunea LAR (Load Access Rights), iar task-urile pot determina dreptul de acces prin instrucțiunile VERR, VERW (Verify Segment for Reading / Writing).

Instrucțiunile specifice modului protejat permit gestionarea corectă a task-urilor. La comutarea de la un task la altul, procesorul trebuie să salveze starea task-ului curent. Starea task-ului curent se

referă la conținutul registrelor generale, registrul indicatorilor de condiții, descriptorii segmentelor pentru cod şi date, selectorul pentru tabela LDT şi legătura către task-ul executat anterior. Starea task-ului se memorează într-un segment în memorie, numit segment de stare task, TSS (Task State Segment). Descriptorul pentru segmentul de stare task este un descriptor special ce se află în tabela GDT. Procesorul are un registru task, TR (Task Register), care conține un selector pentru acel descriptor special, selector care identifică în mod unic task-ul. Accesul la registrul de stare task se realizează prin instrucțiunile STR şi LTR (Store/ Load Task Register). Aceste instrucțiuni specifice modului protejat nu se folosesc, de regulă, când se lucrează sub sistemul de operare MS-DOS. Codul BIOS conține însă, instrucțiuni care testează procesorul în modul protejat. Secvența respectivă de instrucțiuni comută procesorul din modul real în modul protejat şi apoi invers.