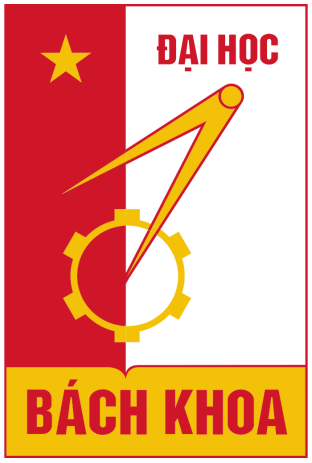
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

———————— * ————————



BÀI TẬP MÔN HỌC
NGUYÊN LÝ HỆ ĐIỀU HÀNH

QUẢN LÝ BỘ NHỚ TRONG VI XỬ LÝ INTEL 80486

Nguyễn Trọng Tuấn 20210906
Đào Quốc Tuấn 20210891
Nguyễn Hoàng Lâm 20210517

Giáo viên hướng dẫn: **TS.Phạm Đăng Hải**

HÀ NỘI
Ngày 8 tháng 7 năm 2023

Mục lục

Lời nói đầu

Quản lý bộ nhớ là một trong những nhiệm vụ quan trọng bậc nhất của một hệ điều hành. Trong hệ thống máy tính, bộ nhớ là một tài nguyên khan hiếm. Cho dù có cải tiến bộ nhớ tăng thêm nhiều đến bao nhiêu chăng nữa thì vẫn không đáp ứng được nhu cầu của con người. Các máy tính hiện nay có bộ nhớ ít nhất là 4Gb. Các máy chủ server có thể có bộ nhớ lên tới hàng Terabyte bộ nhớ. Thế nhưng nhu cầu bộ nhớ vẫn là quá ít. Do đó nhiệm vụ tối quan trọng của các hệ điều hành là tìm ra cách quản lý lượng bộ nhớ đó một cách hợp lý và tối ưu nhất có thể.

Linux là một hệ điều hành họ UNIX miễn phí được sử dụng rộng rãi. Được viết vào năm 1991 bởi Linus Toward, hệ điều hành Linux đã thu được những thành công nhất định. Là một hệ điều hành đa nhiệm, đa người dùng, Linux có thể chạy trên nhiều nền phần cứng khác nhau. Với tính năng ổn định và mềm dẻo, Linux đang dần được sử dụng nhiều trên các máy chủ cũng như các máy trạm trong các mạng máy tính. Linux còn cho phép dễ dàng thực hiện việc tích hợp nó và các hệ điều hành khác trong một mạng máy tính như Windows, Novell, Apple ... Ngoài ra, với tính năng mã nguồn mở, hệ điều hành này còn cho phép khả năng tùy biến cao, thích hợp cho các nhu cầu sử dụng cụ thể .Intel 486 (tên gọi chính thức i486, đôi khi là 80486) là một vi xử lý 32-bit. Nó là phiên bản kế tiếp của Intel 386 với hiệu năng cao hơn. i486 được giới thiệu vào năm 1989. Nó đại diện cho thế hệ thứ tư trong dòng vi xử lý x86 kể từ 8086 được giới thiệu vào năm 1978. Một chip i486 với xung 50 MHz xử lý khoảng 40 triệu dòng lệnh mỗi giây, tức nhanh hơn gấp đôi i386 trong cùng một chu kỳ. Tốc độ cải tiến của i486 đến từ pipeline năm giai đoạn của nó, tất cả các giai đoạn nằm trong một chu kỳ. Bộ xử lý dấu phẩy động của nó, i487, nhanh hơn đáng kể so với i387.

Trong bài báo cáo này, chúng em sẽ giới thiệu những cách mà hệ điều hành Linux và vi xử lý Intel 80486 quản lý bộ nhớ của hệ thống máy tính và tính hiệu quả của chiến lược này.

Đề tài: Quản lý bộ nhớ trong vi xử lý Intel 80486
Môn học: Nguyên lý hệ điều hành
Giáo viên hướng dẫn: Phạm Đăng Hải
Phát hành: 28/06/2023

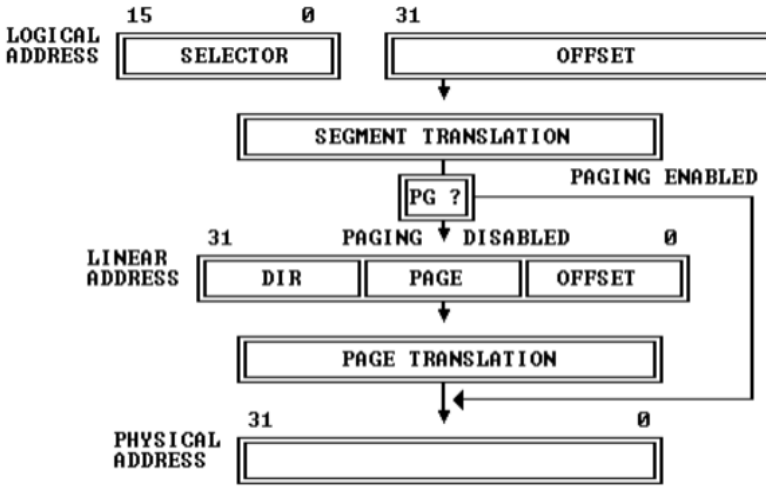
Chương 1

Tổng quan về quản lí bộ nhớ trong vi xử lí Intel 80486

Về cơ chế chung quản lý bộ nhớ của vi xử lý Intel 80486, vi xử lý này cũng sử dụng cơ chế Bộ nhớ ảo với địa chỉ logic và địa chỉ vật lý. Intel 80386 thực hiện việc dịch địa chỉ logic (tức là địa chỉ được các lập trình viên xem) thành địa chỉ vật lý (địa chỉ thực trong bộ nhớ vật lý) bằng 2 bước:

- Phiên dịch đoạn: Ở bước này, địa chỉ logic (bao gồm 2 thành phần segment selector và segment offset) được chuyển đổi thành địa chỉ tuyến tính.
- Phiên dịch trang: Ở bước này, địa chỉ tuyến tính sẽ được chuyển đổi thành địa chỉ vật lý. Đây là một bước mang tính tùy chọn, có nghĩa là nó sẽ được áp dụng tùy theo quyết định của nhà thiết kế hệ thống-phần mềm

Các bước trên được thực hiện một cách ngầm, tức là các lập trình viên sẽ không thể nhìn thấy các bước này thực hiện. Hình dưới đây minh họa 2 bước này một cách trừu tượng:



Hình 1.1: Quản lí bộ nhớ trong Intel 80486

Chương 2

Phân đoạn

Địa chỉ logic bao gồm một bộ chọn đoạn gồm 16 bit và độ lệch gồm 32 bit. Một địa chỉ logic được dịch sang địa chỉ tuyến tính bằng cách cộng độ lệch vào địa chỉ cơ sở của đoạn. Địa chỉ cơ sở được lấy từ bảng mô tả đoạn, một cấu trúc dữ liệu trong bộ nhớ cung cấp kích thước và vị trí của một đoạn, cũng như thông tin kiểm soát truy cập. Bảng mô tả đoạn được lấy từ một trong hai bảng, bảng mô tả chung (GDT) hoặc bảng mô tả cục bộ (LDT). Có một GDT cho tất cả các chương trình trong hệ thống và một LDT cho mỗi chương trình riêng biệt đang chạy. Nếu hệ điều hành cho phép, các chương trình khác nhau có thể chia sẻ cùng một LDT. Hệ thống cũng có thể được thiết lập mà không có LDT; tất cả các chương trình sau đó sẽ sử dụng GDT.

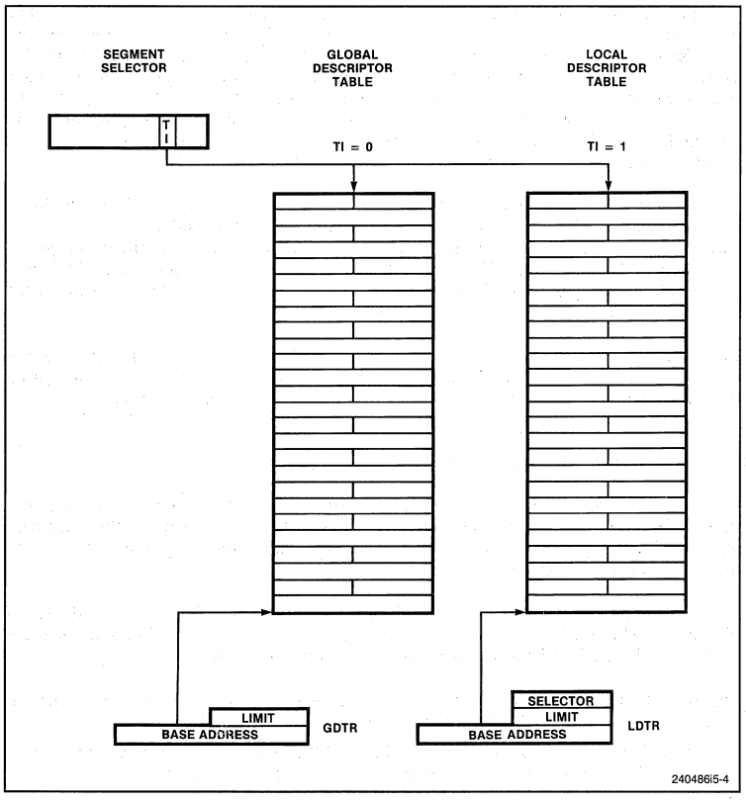
Khi một bộ chọn đoạn được tải, địa chỉ cơ sở, giới hạn đoạn và thông tin kiểm soát truy cập cũng được tải vào thanh ghi đoạn. Bộ xử lý không tham chiếu lại các bảng mô tả cho đến khi một bộ chọn đoạn khác được tải. Thông tin được lưu trữ trong bộ xử lý cho phép nó dịch địa chỉ mà không cần thực hiện thêm chu kỳ bus. Trong các hệ thống mà nhiều bộ xử lý có quyền truy cập vào cùng các bảng mô tả, việc tải lại các thanh ghi đoạn khi các bảng mô tả bị thay đổi là trách nhiệm của phần mềm. Nếu không thực hiện điều này, một bảng mô tả đoạn cũ được lưu trong thanh ghi đoạn có thể được sử dụng sau khi phiên bản lưu trong bộ nhớ của nó đã được thay đổi.

Bộ chọn đoạn chứa một chỉ mục 13 bit trong một trong các bảng mô tả. Chỉ mục này được nhân lên tám (số byte trong một bảng mô tả đoạn) và cộng vào địa chỉ cơ sở 32 bit của bảng mô tả. Địa chỉ cơ sở được lấy từ thanh ghi bảng mô tả chung (GDTR) hoặc thanh ghi bảng mô tả cục bộ (LDTR). Các thanh ghi này giữ địa chỉ tuyến tính của đầu bảng mô tả. Một TI bit trong bộ chọn đoạn xác định bảng nào được sử dụng, như được mô tả trong Hình 2-1.

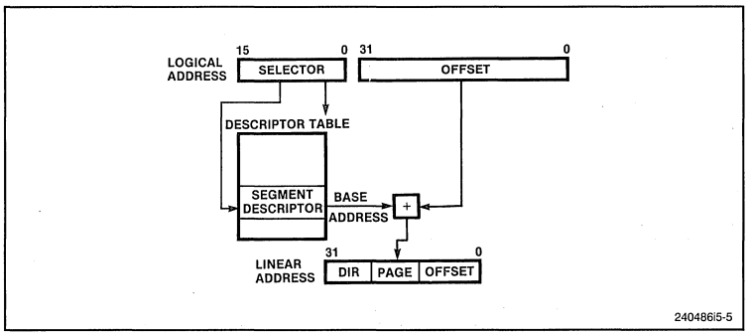
Địa chỉ được dịch là địa chỉ tuyến tính, như được mô tả trong Hình 2-2. Nếu không sử dụng phân trang, nó cũng là địa chỉ vật lý. Nếu sử dụng phân trang, một cấp độ dịch thuật địa chỉ thứ hai sẽ tạo ra địa chỉ vật lý. Quá trình dịch thuật này được mô tả cụ thể trong Mục 2.3.

2.1 Thanh ghi phân đoạn

Mỗi loại tham chiếu bộ nhớ được liên kết với một thanh ghi đoạn. Các tham chiếu mã, dữ liệu và ngăn xếp mỗi lần truy cập các đoạn được chỉ định bởi nội dung của thanh ghi đoạn tương ứng. Có thể có nhiều đoạn khác được sẵn có bằng cách tải các bộ chọn đoạn của chúng vào các thanh ghi này trong quá trình thực thi chương trình.



Hình 2-1. TI Bit Selects Descriptor Table



Hình 2-2. Segment Translation

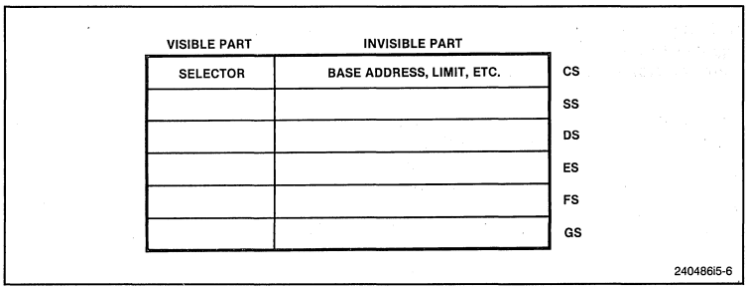
Mọi thanh ghi phân đoạn đều có một phần "nhìn thấy được" và một phần "ẩn", như Hình 2-3 minh họa. Các phần hiển thị của các thanh ghi địa chỉ phân đoạn này được các chương trình thao tác như với các thanh ghi 16 bit. Các phần vô hình được thao tác bởi bộ xử lý.

Các hoạt động tải các thanh ghi này là các lệnh chương trình bình thường. Các lệnh này gồm hai lớp:

- **Tải trực tiếp**: ví dụ là MOV, POP, LDS, LSS, LGS, LFS. Các lệnh này tham chiếu rõ ràng đến các thanh ghi phân đoạn.
- **Tải ngầm**: Các lệnh này tham chiếu ngầm đến thanh ghi CS và tải nó với một giá trị mới.

Khi sử dụng các lệnh này, phần nhìn thấy của thanh ghi đoạn được tải với một bộ chọn đoạn. Bộ xử lý tự động truy xuất địa chỉ cơ sở (Base), Limit, Type và các thông tin khác từ bảng mô tả và tải phần ẩn của thanh ghi đoạn.

Bởi vì hầu hết các lệnh tham chiếu đến các đoạn mà các bộ chọn của chúng đã được tải vào các thanh ghi đoạn, bộ xử lý có thể cộng địa chỉ logic offset với địa chỉ cơ sở của đoạn mà không làm giảm hiệu suất.



Hình 2-3. Segment Registers

2.2 Bộ chọn

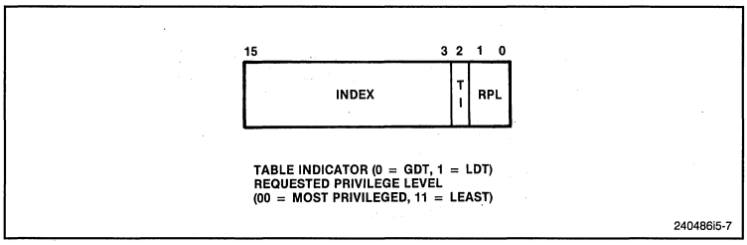
Một bộ chọn đoạn trỏ đến thông tin xác định một đoạn, được gọi là mô tả đoạn. Một chương trình có thể có nhiều đoạn hơn so với sáu bộ chọn đoạn mà các thanh ghi đoạn chiếm. Khi điều này đúng, chương trình sử dụng các dạng của lệnh MOV để thay đổi nội dung của các thanh ghi này khi cần truy cập vào một đoạn mới.

Một bộ chọn đoạn xác định một mô tả đoạn bằng cách chỉ định một bảng mô tả và một mô tả trong bảng đó. Các bộ chọn đoạn có thể nhìn thấy cho các chương trình ứng dụng như một phần của một biến con trỏ, nhưng các giá trị của bộ chọn thường được gán hoặc sửa đổi bởi trình biên dịch liên kết hoặc trình tải liên kết, không phải chương trình ứng dụng. Hình 5-7 hiển thị định dạng của một bộ chọn đoạn.

Index: Chọn một trong 8192 mô tả trong một bảng mô tả. Bộ xử lý nhân giá trị chỉ mục với 8 (số byte trong một mô tả đoạn) và cộng kết quả vào địa chỉ cơ sở của bảng mô tả (từ thanh ghi GDTR hoặc LDTR).

Table Indicator bit: Chỉ định bảng mô tả mà bộ chọn tham chiếu. Số 0 cho biết GDT; 1 cho biết LDT.

Requester Privilege Level (Mức đặc quyền của người yêu cầu): Khi trường này chứa một mức đặc quyền có giá trị cao hơn (tức là ít đặc quyền hơn) so với chương trình, nó ghi đè mức đặc quyền của chương trình. Khi một chương trình sử dụng một bộ chọn đoạn có đặc quyền thấp hơn, các truy cập bộ nhớ sẽ xảy ra ở mức đặc quyền thấp hơn. Điều này được sử dụng để bảo vệ chống lại vi phạm bảo mật khi một chương trình ít đặc quyền sử dụng một chương trình có đặc quyền cao hơn để truy cập vào dữ liệu được bảo vệ.



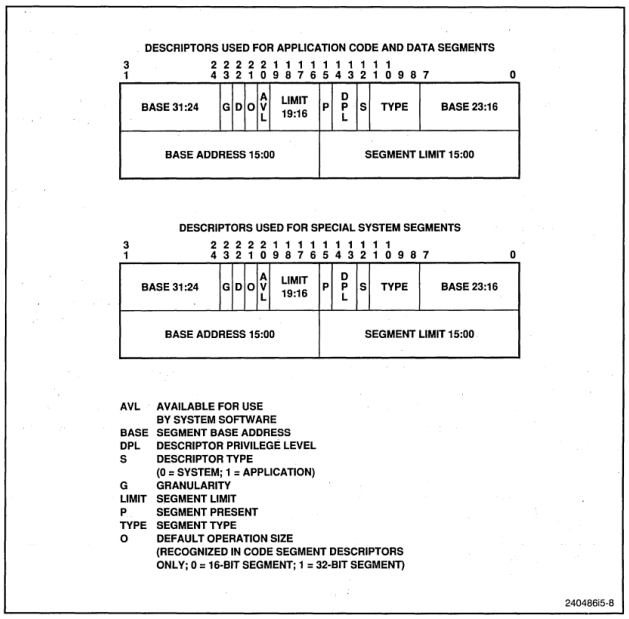
Hình 2-4. Segment Selector

Ví dụ, các tiện ích hệ thống hoặc trình điều khiển thiết bị phải chạy với mức đặc quyền cao để truy cập vào các cơ sở vật lý được bảo vệ như các thanh ghi điều khiển của giao diện ngoại vi. Tuy nhiên, chúng không được can thiệp vào các cơ sở vật lý khác, ngay cả khi nhận được yêu cầu từ một chương trình ít đặc quyền hơn. Nếu một chương trình yêu cầu đọc một sector trên đĩa vào bộ nhớ được chiếm bởi một chương trình có đặc quyền cao hơn, như hệ điều hành, RPL có thể được sử dụng để tạo ra một ngoại lệ bảo vệ tổng quát khi sử dụng bộ chọn đoạn ít đặc quyền. Ngoại lệ này xảy ra ngay cả khi chương trình sử dụng bộ chọn đoạn sẽ có mức đặc quyền đủ để thực hiện thao tác một cách riêng.

2.3 Mô tả

Một bộ mô tả phân đoạn là một cấu trúc dữ liệu trong bộ nhớ cung cấp cho bộ xử lý thông tin về kích thước và vị trí của một đoạn, cũng như thông tin điều khiển và trạng thái. Thông thường, Bộ mô tả được tạo bởi trình biên dịch,trình liên kết, trình tải hoặc hệ điều hành, không phải bởi lập trình viên ứng dụng. Hình 2-5 minh họa hai định dạng mô tả tổng quát. Các trường mô tả phân đoạn là:

- **Base**: Xác định vị trí của phân đoạn trong không gian địa chỉ tuyến tính 4 gigabyte. Bộ xử lý nối ba đoạn của địa chỉ cơ sở để tạo thành một giá trị 32 bit duy nhất.



Hình 2-5. Segment Descriptors

- **Granularity bit**: Sử dụng để điều chỉnh cách giới hạn đoạn (Limit) được hiểu và đo lường. Khi Granularity bit là 1, giới hạn đoạn (Limit) được chia thành các đơn vị 4K byte (hay một trang). Khi bit này là 0, giới hạn đoạn được hiểu theo đơn vị byte đơn lẻ. Tuy nhiên, chỉ có trường Limit bị ảnh hưởng bởi Granularity bit. Địa chỉ cơ sở (Base) vẫn được hiểu và xử lý theo đơn vị byte.

- **Limit**: Xác định kích thước của phân đoạn. Bộ xử lý nối hai phần của trường Limit thu được một giá trị 20 bit. Bộ xử lý hiểu giới hạn theo hai cách, tùy thuộc vào cài đặt của Granularity bit:

  – Theo đơn vị một byte, Limit có thể đạt được giá trị lên tới 1 megabyte.

  – Theo đơn vị 4 Kilobyte, Limit có thể đạt được giá trị lên tới 4 Gigabyte. Limit được dịch chuyển bit sang trái 12 bit khi được load, và sau đó các bit bậc thấp sẽ được thêm vào.

- **S bit**: Xác định xem đoạn được cho là một đoạn hệ thống hay một đoạn mã hoặc dữ liệu. Nếu bit S là 1, thì đoạn là một đoạn mã hoặc dữ liệu. Nếu bit này là 0, thì đoạn là một đoạn hệ thống.

- **D bit**: Chỉ ra độ dài mặc định cho các toán hạng và chế độ địa chỉ hiệu quả. Nếu bit D là 1, thì giả định sử dụng toán hạng 32 bit và các chế độ địa chỉ 32 bit. Nếu bit này là 0, thì giả định sử dụng toán hạng 16 bit và chế độ địa chỉ 16 bit.

- **Type**: Phân biệt giữa các loại bộ mô tả, phụ thuộc vào việc xem xét xem bộ mô tả phân đoạn là một mô tả đoạn ứng dụng hay một mô tả đoạn hệ thống (Từ S bit). Trường Type của bộ mô tả địa chỉ bộ nhớ xác định loại truy cập có thể được thực hiện vào một đoạn.

Bảng 2-6. Application Segment Types

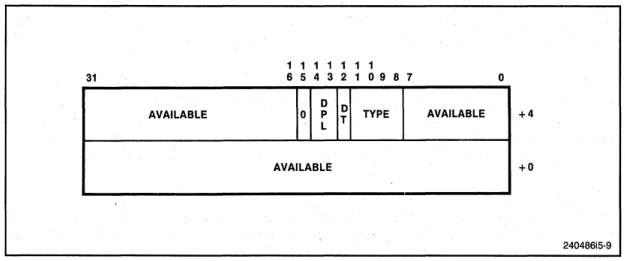
Number	E	W	A	Descriptor Type	Description
0	0	0	0	Data	Read-Only
1	0	0	1	Data	Read-Only, accessed
2	0	1	0	Data	Read/Write
3	0	1	1	Data	Read/Write, accessed
4	1	0	0	Data	Read-Only, expand-down
5	1	0	1	Data	Read-Only, expand-down, accessed
6	1	1	0	Data	Read/Write, expand-down
7	1	1	1	Data	Read/Write, expand-down, accessed
Number	**C**	**R**	**A**	**Descriptor Type**	**Description**
8	0	0	0	Code	Execute-Only
9	0	0	1	Code	Execute-Only, accessed
10	0	1	0	Code	Execute/Read
11	0	1	1	Code	Execute/Read, accessed
12	1	0	0	Code	Execute-Only, conforming
13	1	0	1	Code	Execute-Only, conforming, accessed
14	1	1	0	Code	Execute/Read-Only, conforming
15	1	1	1	Code	Execute/Read-Only, conforming, accessed

Đối với các đoạn dữ liệu, ba bit cuối cùng của trường Type có thể hiểu là expand-down (E), write enable (W) và accessed (A). Đối với các đoạn mã, ba bit cuối cùng của trường loại có thể hiểu là conforming (C), read enable (R) và accessed (A).

Trường Type cũng cho biết liệu đoạn đã được truy cập hay chưa. Ban đầu, bộ mô tả đoạn cho biết rằng đoạn đã được truy cập hay chưa. Nếu trường Type sau đó được đặt

thành một giá trị ứng với đoạn chưa được truy cập, bộ xử lý sẽ khôi phục giá trị nếu đoạn đã được truy cập. Bằng cách xóa và kiểm tra bit thấp nhất của trường Type, phần mềm có thể giám sát việc sử dụng đoạn (bit thấp nhất của trường Type cũng được gọi là Accessed bit).

- **Descriptor Privilege Level** (Mức đặc quyền mô tả): Xác định cấp độ đặc quyền của đoạn. Điều này được sử dụng để kiểm soát quyền truy cập vào đoạn, sử dụng cơ chế bảo vệ.



Hình 2-7. Segment Descriptor

- **Segment-Present bit**: Nếu bit này có giá trị là 0, điều này đồng nghĩa với việc bộ mô tả không hợp lệ để sử dụng. Bộ xử lý sẽ bảo lỗi khi bộ mô tả được chọn để tải vào một thanh ghi phân đoạn. Hình 2-7 cho thấy định dạng bộ mô tả phân đoạn khi bit này nhận giá trị là 0. Hệ điều hành được truy cập vào những vị trí có nhãn AVAILABLE. Những điều hành triển khai bộ nhớ ảo dựa trên cơ chế phân đoạn sẽ xóa bit segment-present trong trường hợp sau:
  - Khi không gian tuyến tính sinh ra bởi phân đoạn không được ánh xạ bởi cơ chế phân trang.
  - Khi phân đoạn không có trong bộ nhớ.

  Việc tạo và duy trì các bộ mô tả là trách nhiệm của hệ thống, thường đòi hỏi sự hợp tác của trình biên dịch, trình tải chương trình hoặc trình xây dựng hệ thống và hệ thống vận hành.

2.4 Bảng mô tả phân đoạn

Các bộ mô tả phân đoạn được lưu trữ trong một trong hai loại bảng mô tả:

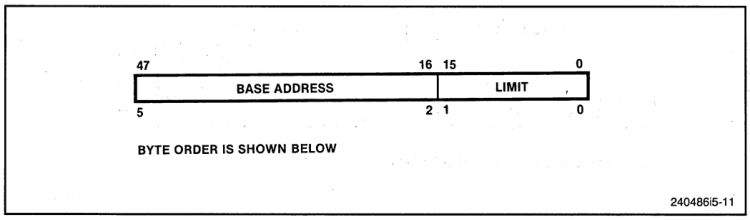
- Bảng mô tả chung (GDT)
- Bảng mô tả cục bộ (LDT)

Có một bảng mô tả chung (Global Descriptor Table) cho tất cả các tác vụ (tasks), và một bảng mô tả cục bộ (Local Descriptor Table) cho mỗi tác vụ được chạy. Bảng mô tả (descriptor table) là một mảng các mô tả phân đoạn (segment descriptors). Một bảng mô tả có độ dài thay đổi và có thể chứa tối đa 8192 (2^{13}) mô tả (descriptors). Mô tả đầu tiên trong bảng mô tả chung không được sử dụng bởi bộ xử lý. Bộ xử lý định vị vị trí GDT và LDT hiện tại trong bộ nhớ bằng các thanh ghi GDTR và LDTR. Các thanh ghi này lưu trữ địa chỉ cơ sở của các bảng trong không gian địa chỉ tuyến tính và lưu trữ giới hạn phân

đoạn. Các chỉ dẫn LGDT và SGDT cấp quyền truy cập vào GDTR; các chỉ dẫn LLDT và SLDT cấp quyền truy cập vào LDTR. Mô tả cụ thể về thanh ghi GDTR được giới thiệu ở mục tiếp theo.

2.5 Thanh ghi bảng mô tả phân đoạn

Bộ xử lý tìm thấy bảng mô tả toàn cục (GDT) và bảng mô tả ngắn đặc tuyến (IDT) bằng cách sử dụng các thanh ghi GDTR và IDTR. Các thanh ghi này chứa địa chỉ cơ sở 32-bit cho các bảng trong không gian địa chỉ tuyến tính. Chúng cũng chứa giá trị giới hạn 16-bit cho kích thước của các bảng này. Khi các thanh ghi được nạp hoặc lưu trữ, một "pseudo-descriptor" 48-bit được truy cập trong bộ nhớ, như được hiển thị trong Hình 2-8. GDT và IDT nên được căn chỉnh trên một ranh giới 16 byte để tối đa hóa hiệu suất do việc lấp đầy dòng cache.



Hình 2-8. Pseudo-Descriptor Format

Giá trị giới hạn được biểu thị bằng byte. Giống như các đoạn, giá trị giới hạn được cộng vào địa chỉ cơ sở để có được địa chỉ của byte cuối cùng hợp lệ. Một giá trị giới hạn là 0 sẽ cho kết quả là chính xác một byte hợp lệ. Bởi vì mô tả đoạn luôn có kích thước là tám byte, giới hạn nên luôn là một ít hơn một bội số nguyên của tám (tức là 8N - 1). Các lệnh LGDT và SGDT đọc và ghi thanh ghi GDTR; các lệnh LIDT và SIDT đọc và ghi thanh ghi IDTR.

Bảng mô tả thứ ba là bảng mô tả cục bộ (LDT). Nó được xác định bằng cách sử dụng một bộ chọn đoạn 16-bit nằm trong thanh ghi LDTR. Các lệnh LLDT và SLDT đọc và ghi bộ chọn đoạn trong thanh ghi LDTR. Thanh ghi LDTR cũng chứa địa chỉ cơ sở và giới hạn cho LDT, nhưng chúng được tự động nạp bởi bộ xử lý từ bảng mô tả đoạn cho LDT. LDT nên được căn chỉnh trên một ranh giới 16 byte để tối đa hóa hiệu suất do việc lấp đầy dòng cache.

Chương 3

Phân trang

Phân trang khác biệt với phân đoạn thông qua việc sử dụng các trang có kích thước nhỏ cố định. Khác với các đoạn, thường có cùng kích thước với cấu trúc dữ liệu mà chúng giữ, trên bộ xử lý Intel486, các trang luôn có kích thước 4K byte. Nếu chỉ sử dụng phân đoạn là hình thức duy nhất của việc dịch địa chỉ được sử dụng, một cấu trúc dữ liệu hiện diện trong bộ nhớ vật lý sẽ có tất cả các phần của nó trong bộ nhớ. Nếu sử dụng phân trang, một cấu trúc dữ liệu có thể chỉ được một phần trong bộ nhớ và một phần trong lưu trữ đĩa.

Thông tin ánh xạ địa chỉ tuyến tính thành địa chỉ vật lý và ngoại lệ được lưu trữ trong các cấu trúc dữ liệu trong bộ nhớ được gọi là bảng trang (page tables). Tương tự như phân đoạn, thông tin này được lưu trong bộ đệm (cache) của các thanh ghi bộ xử lý để giảm thiểu số chu kỳ bus cần thiết cho việc dịch địa chỉ. Tuy nhiên, khác với phân đoạn, các thanh ghi bộ xử lý này hoàn toàn không thể nhìn thấy bởi các chương trình ứng dụng.

Cơ chế phân trang xem xét địa chỉ tuyến tính 32-bit như có ba phần, hai chỉ mục 10 bit vào bảng trang và một offset 12 bit vào trang được chỉ định bởi bảng trang. Vì cả các trang ảo trong không gian địa chỉ tuyến tính và các trang vật lý trong bộ nhớ được căn chỉnh theo ranh giới trang 4K-byte, không cần thiết phải thay đổi 12 bit thấp nhất của địa chỉ. 12 bit này truyền thẳng qua phần cứng phân trang. Lưu ý rằng điều này khác với phân đoạn, vì các đoạn có thể bắt đầu từ bất kỳ địa chỉ byte nào.

Trong đó, 20 bit cao nhất của địa chỉ được sử dụng để chỉ mục vào bảng trang. Nếu mỗi trang trong không gian địa chỉ tuyến tính được ánh xạ bằng một bảng trang duy nhất trong RAM, sẽ cần 4 megabyte. Tuy nhiên, điều này không được thực hiện. Thay vào đó, sử dụng hai cấp bảng trang. Bảng trang cấp cao nhất được gọi là bảng thư mục trang (page directory). Nó ánh xạ 10 bit cao nhất của địa chỉ tuyến tính vào cấp thứ hai của bảng trang. Cấp thứ hai của bảng trang ánh xạ 10 bit giữa của địa chỉ tuyến tính vào địa chỉ cơ sở của một trang trong bộ nhớ vật lý (gọi là địa chỉ khung trang).

Thanh ghi CR3 lưu trữ địa chỉ khung trang của bảng thư mục trang. Vì lý do này, nó cũng được gọi là thanh ghi cơ sở bảng thư mục trang hoặc PDBR (Page Directory Base Register).Trong đó, 10 bit cao nhất của địa chỉ tuyến tính được nhân với bốn (số byte trong một mục nhập bảng trang) và được cộng vào giá trị trong thanh ghi PDBR để lấy địa chỉ vật lý của một mục nhập trong bảng thư mục trang. Vì địa chỉ khung trang luôn được xóa trong 12 bit thấp nhất của nó, phép cộng này được thực hiện bằng cách ghép nối (thay thế

12 bit thấp nhất bằng chỉ số đã được nhân).

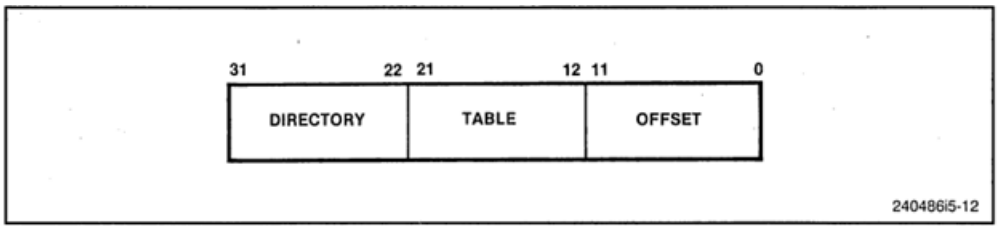
3.1 Bit PG Cho phép Phân trang

Khi phân trang được kích hoạt, một giai đoạn thứ hai của việc dịch địa chỉ được sử dụng để tạo ra địa chỉ vật lý từ địa chỉ tuyến tính. Nếu phân trang không được kích hoạt, địa chỉ tuyến tính được sử dụng làm địa chỉ vật lý.

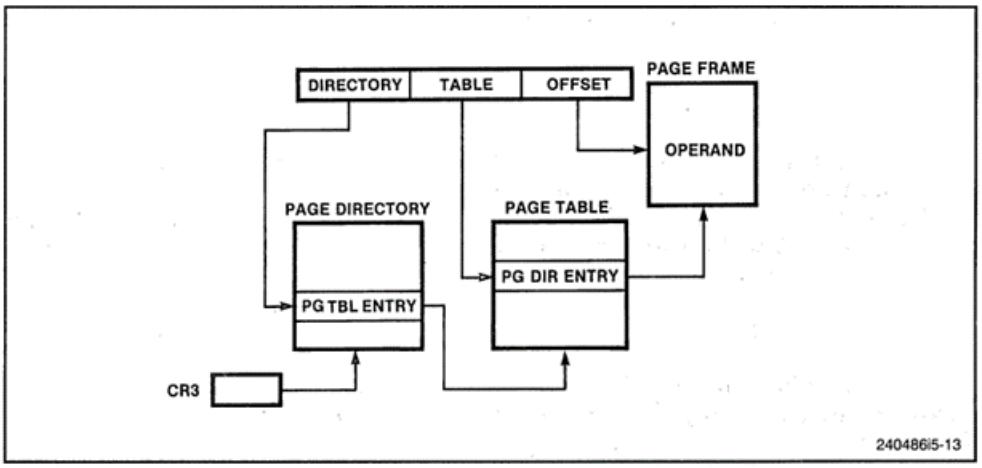
Phân trang được kích hoạt khi bit 31 (bit PG) của thanh ghi CR0 được thiết lập. Bit này thường được thiết lập bởi hệ điều hành trong quá trình khởi tạo phần mềm. Bit PG phải được thiết lập nếu hệ điều hành đang chạy nhiều chương trình trong chế độ ảo hóa 8086 hay nếu sử dụng bộ nhớ ảo được yêu cầu phân trang.

3.2 Địa chỉ tuyến tính

Hình 5-12 Mô tả định dạng của một địa chỉ tuyến tính.



Hình 5-12. Format of a Linear Address



Hình 5-13. Page Translation

Hình 5-13 mô tả cách bộ xử lý dịch các trường DIRECTORY, TABLE và OFFSET của một địa chỉ tuyến tính thành địa chỉ vật lý bằng cách sử dụng hai cấp bảng trang. Cơ chế phân trang sử dụng trường DIRECTORY làm chỉ mục vào một bảng thư mục trang, trường TABLE làm chỉ mục vào bảng trang được xác định bởi bảng thư mục trang, và trường OFFSET để địa chỉ một toán hạng trong trang được chỉ định bởi bảng trang.

3.3 Bảng trang

Một bảng trang là một mảng các mục nhập 32 bit. Một bảng trang cũng là một trang và chứa 4096 byte bộ nhớ hoặc tối đa là 1K mục nhập 32 bit. Tất cả các trang, bao gồm bảng thư mục trang và bảng trang, được căn chỉnh theo ranh giới 4K byte.

Để địa chỉ một trang bộ nhớ, hai cấp bảng được sử dụng. Cấp cao nhất được gọi là bảng thư mục trang. Nó địa chỉ tối đa 1K bảng trang ở cấp thứ hai. Một bảng trang ở cấp thứ hai địa chỉ tối đa 1K trang trong bộ nhớ vật lý. Tất cả các bảng được địa chỉ bởi một bảng thư mục trang, do đó, có thể địa chỉ 1M trang hoặc 220 trang. Vì mỗi trang chứa 4K hoặc 212 byte, các bảng của một bảng thư mục trang có thể trải dài trên toàn bộ không gian địa chỉ tuyến tính của bộ xử lý Intel486 (220 x212 = 232).

Địa chỉ vật lý của bảng thư mục trang hiện tại được lưu trữ trong thanh ghi CR3, còn được gọi là thanh ghi cơ sở của bảng thư mục trang (PDBR). Phần mềm quản lý bộ nhớ có tùy chọn sử dụng một bảng thư mục trang cho tất cả các tác vụ, một bảng thư mục trang cho mỗi tác vụ, hoặc một số sự kết hợp của hai cách trên.

3.4 Mục nhập ở bảng trang

Các mục trong cả bảng thư mục trang và bảng trang ở cấp độ tương ứng có cùng định dạng, chỉ trừ bảng thư mục trang không có bit Dirty. Hình 5-14 minh họa định dạng này. Vị trí bit của bit D đã được dành cho mục đích sử dụng của Intel trong tương lai.

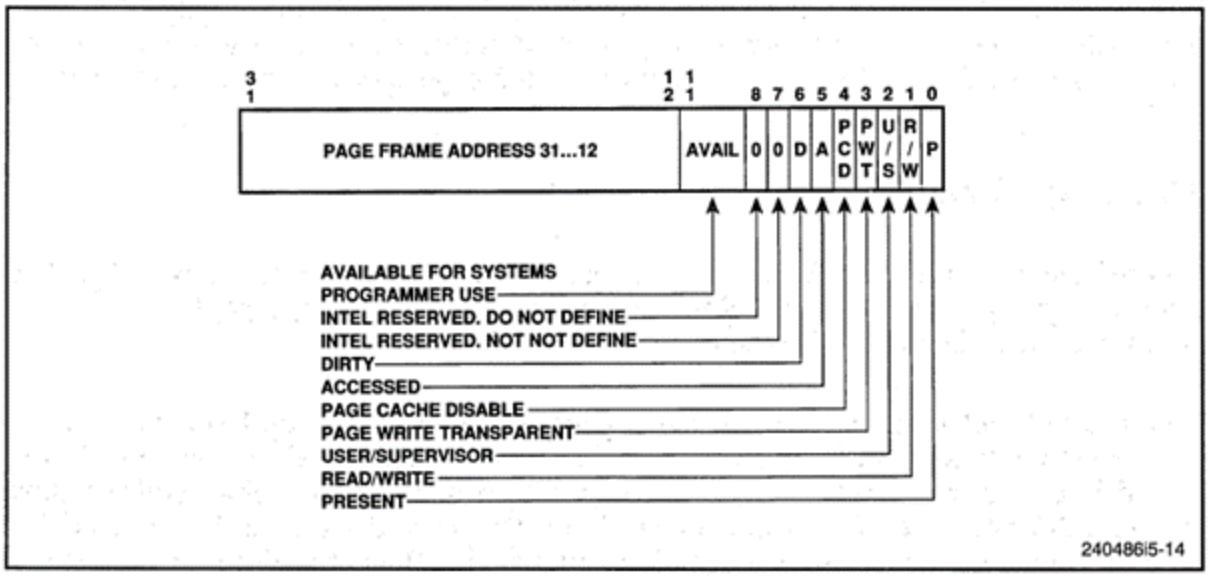
3.4.1 Địa chỉ khung trang

Địa chỉ khung trang là địa chỉ cơ sở của một trang. Trong mục nhập bảng trang, 20 bit cao được sử dụng để chỉ định địa chỉ khung trang, và 12 bit thấp chỉ định các bit điều khiển và trạng thái cho trang. Trong bảng thư mục trang, địa chỉ khung trang là địa chỉ của một bảng trang. Trong bảng trang cấp thứ hai, địa chỉ khung trang là địa chỉ của một trang chứa các chỉ thị hoặc dữ liệu.

3.4.2 PRESENT BIT

Present bit cho biết liệu địa chỉ khung trang trong mục nhập bảng trang ánh xạ tới một trang trong bộ nhớ vật lý hay không. Khi được thiết lập, trang đó có mặt trong bộ nhớ.

Khi bit Có mặt không được thiết lập, trang không có trong bộ nhớ và phần còn lại của mục nhập bảng trang có sẵn cho hệ điều hành, ví dụ như lưu trữ thông tin về vị trí của trang bị thiếu. Hình 5·15 mô tả định dạng của một mục nhập bảng trang khi bit Có mặt không được thiết lập.



Hình 5·14. Format of a Page Table Entry

Nếu Present bit không được đặt trong bất kỳ cấp độ nào của bảng trang khi có một nỗ lực sử dụng mục nhập bảng trang cho việc dịch địa chỉ, một ngoại lệ page-fault được tạo ra. Trong các hệ thống hỗ trợ bộ nhớ ảo theo yêu cầu, sau đây là chuỗi sự kiện xảy ra:

1. Hệ điều hành sao chép trang từ bộ nhớ đĩa vào bộ nhớ vật lý.

2. Hệ điều hành tải địa chỉ khung trang vào mục nhập bảng trang và đặt Present bit của nó. Có thể đặt các bit khác như bit R/W.

3. Do một bản sao của mục nhập bảng trang cũ có thể vẫn tồn tại trong bộ đệm tìm kiếm dịch (TLB), hệ điều hành xóa nó.

4. Chương trình gây ra ngoại lệ sau đó được khởi động lại.

Vì không có Present bit trong CR3 để chỉ ra khi bảng thư mục không cư trú trong bộ nhớ, bảng thư mục được chỉ định bởi CR3 nên luôn có mặt trong bộ nhớ vật lý.

3.4.3 ACCESSED VÀ DIRTY BITS

Các bit này cung cấp dữ liệu về việc sử dụng trang trong cả hai cấp độ của bảng trang. Accessed bit được sử dụng để báo cáo việc truy cập đọc hoặc ghi vào một trang hoặc bảng trang cấp hai. Dirty bit được sử dụng để báo cáo việc ghi vào một trang.

Trừ Dirty bit trong mục nhập bảng thư mục, các bit này được thiết bị phần cứng đặt; tuy nhiên, bộ xử lý không xóa bất kỳ bit nào trong số này. Bộ xử lý đặt Accessed bit trong cả hai cấp độ của bảng trang trước khi thực hiện hoạt động đọc hoặc ghi vào một trang. Bộ xử lý đặt bit Dirty trong bảng trang cấp hai trước khi thực hiện hoạt động ghi vào địa chỉ được ánh xạ bởi mục nhập bảng trang đó. Dirty bit trong các mục nhập thư mục được định nghĩa chưa rõ ràng.

Hệ điều hành có thể sử dụng Accessed bit khi cần giải phóng một số bộ nhớ trống bằng cách gửi một trang hoặc bảng trang cấp hai vào bộ nhớ đĩa. Bằng cách định kỳ xóa các

Accessed bit trong các bảng trang, nó có thể xem xét xem các trang đã được sử dụng gần đây. Các trang chưa được sử dụng là ứng viên để gửi ra đĩa.

Hệ điều hành có thể sử dụng Dirty bit khi một trang được gửi trở lại đĩa. Bằng cách xóa Dirty bit khi trang được đưa vào bộ nhớ, hệ điều hành có thể xem xét xem trang đã nhận bất kỳ quyền truy cập ghi nào. Nếu có một bản sao của trang trên đĩa và bản sao trong bộ nhớ chưa nhận bất kỳ ghi nào, không cần cập nhật đĩa từ bộ nhớ.

3.4.4 Bit Read/Write và User/Supervisor

Bit Read/Write và User/Supervisor được sử dụng để thực hiện kiểm tra bảo vệ trên các trang, mà bộ xử lý thực hiện cùng lúc với việc dịch địa chỉ.

3.4.5 Các Bit Điều khiển Bộ nhớ Cache Cấp Trang

Các bit PCD và PWT được sử dụng để quản lý bộ nhớ cache cấp trang. Phần mềm có thể kiểm soát việc lưu trữ cache của các trang hoặc bảng trang cấp hai riêng lẻ bằng cách sử dụng các bit này.

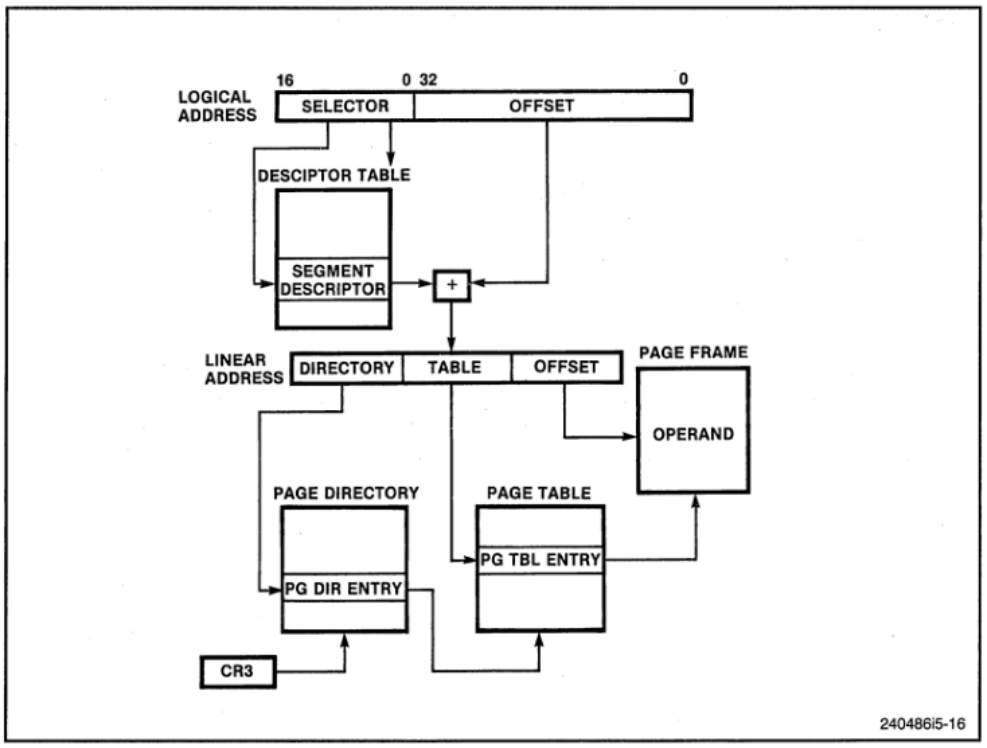
Chương 4

Phân đoạn kết hợp phân đoạn

4.1 Mô hình phẳng (Flat mode)

Khi sử dụng bộ xử lý Intel 486 để thực thi phần mềm không sử dụng phân đoạn (segmentation), có thể có nhu cầu loại bỏ các tính năng phân đoạn của bộ xử lý Intel 486. Bộ xử lý Intel 486 không có một bit chế độ để vô hiệu hóa phân đoạn, tuy nhiên, hiệu ứng tương tự có thể được đạt được bằng cách ánh xạ không gian stack, mã và dữ liệu vào cùng một phạm vi địa chỉ tuyến tính. Các hệ số 32-bit được sử dụng bởi các chỉ thị của bộ xử lý Intel 486 có thể bao phủ toàn bộ không gian địa chỉ tuyến tính.

Khi sử dụng phân trang (paging), các phân đoạn có thể được ánh xạ vào toàn bộ không gian địa chỉ tuyến tính. Trong trường hợp có nhiều hơn một chương trình được thực thi đồng thời, cơ chế phân trang có thể được sử dụng để cung cấp cho mỗi chương trình một không gian địa chỉ riêng biệt.



Hình 4.1: Chiến lược phân đoạn kết hợp phân trang

4.2 Phân đoạn mở rộng phân trang

Kiến trúc của Intel 80386 cho phép các phân đoạn lớn hơn hoặc nhỏ hơn kích thước của một trang (4 Kilobyte). Ví dụ: giả sử một phân đoạn được sử dụng để giải quyết và bảo vệ cấu trúc dữ liệu lớn kéo dài 132 Kilobyte. Trong một hệ thống phần mềm hỗ trợ bộ nhớ ảo được phân trang, không nhất thiết toàn bộ cấu trúc phải nằm trong bộ nhớ vật lý cùng một lúc. Cấu trúc được chia thành 33 trang, bất kỳ trang nào trong số đó có thể không có. Người lập trình ứng dụng không cần biết rằng hệ thống con bộ nhớ ảo đang phân trang cấu trúc theo cách này.

4.3 Phân trang mở rộng phân đoạn

Các phân đoạn cũng có thể nhỏ hơn kích thước của một trang. Trong trường hợp một phân đoạn được đặt trong một trang không được chia sẻ với phân đoạn khác, bộ nhớ dư thừa sẽ bị lãng phí. Ví dụ, giả sử chúng ta có 10 semaphore, mỗi semaphore chỉ chiếm 1 byte, tuy nhiên, mỗi semaphore lại được đặt trong một trang riêng biệt có kích thước 4K byte. Khi đó, tổng cộng chúng ta sẽ sử dụng 10 trang, tốn 40K byte bộ nhớ. Trong trường hợp này, chúng ta có thể gói 10 semaphore vào cùng một trang. Như vậy, chúng ta chỉ cần 1 trang (4K byte) để lưu trữ toàn bộ 10 semaphore, giúp tiết kiệm bộ nhớ.

Bằng cách gói những cấu trúc dữ liệu nhỏ vào cùng một trang, chúng ta có thể tận dụng bộ nhớ hiệu quả hơn và tránh lãng phí không gian bộ nhớ

4.4 Ranh giới phân trang và phân đoạn không tương ứng

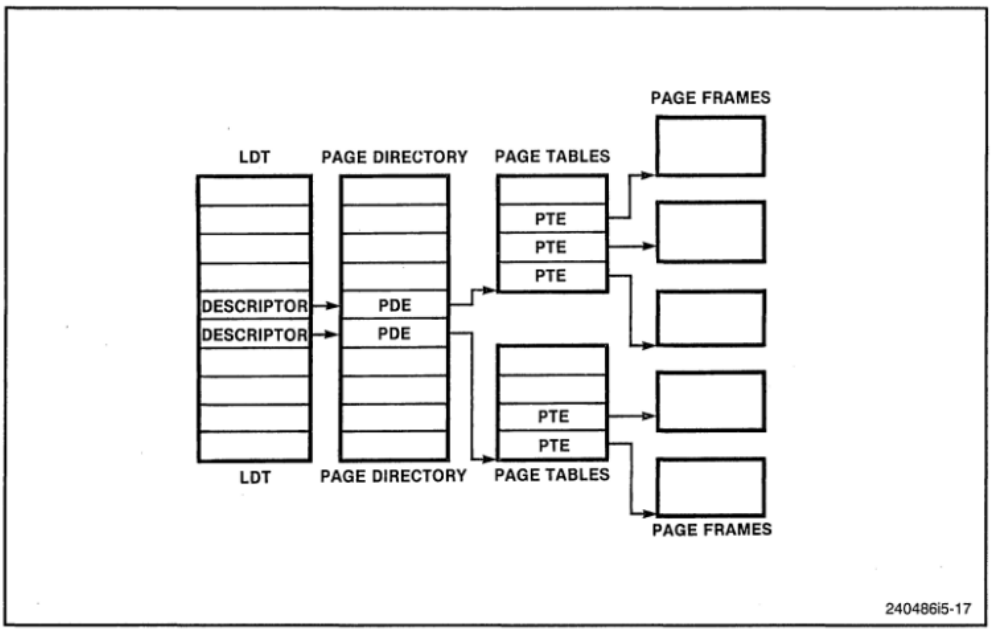
Kiến trúc của 80846 không thực thi bất kỳ sự tương đương nào giữa các ranh giới của các trang và các phân đoạn. Hoàn toàn có thể cho phép một trang chứa phần cuối của một phân đoạn và phần đầu của một phân đoạn khác. Tương tự như vậy, một phân đoạn có thể chứa phần cuối của một trang và phần đầu của trang khác.

4.5 Ranh giới phân đoạn và trang tương ứng

Tuy nhiên, phần mềm quản lý bộ nhớ có thể đơn giản hơn nếu nó thực thi một số tương ứng giữa ranh giới trang và phân đoạn. Ví dụ: nếu các phân đoạn chỉ được phân bổ theo đơn vị của một trang, thì logic phân bổ phân đoạn và trang có thể được kết hợp.

4.6 Bảng trang trên mỗi phân đoạn

Một cách tiếp cận để quản lý không gian cung cấp sự đơn giản hóa hơn nữa của phần mềm quản lý không gian là duy trì sự tương ứng 1-1 giữa các bộ mô tả phân đoạn và các mục nhập thư mục trang, như hình dưới đây minh họa. Mỗi bộ mô tả có một địa chỉ cơ sở trong đó 22 bit bậc thấp bằng 0; nói cách khác, địa chỉ cơ sở được ánh xạ bởi mục nhập đầu tiên của bảng trang. Một phân đoạn có thể có bất kỳ limig nào từ 1 đến 4 megabyte. Tùy thuộc vào giới hạn, phân đoạn được chứa trong từ 1 đến 1K khung trang. Do đó, một tác vụ được giới hạn trong 1K phân đoạn (một con số đủ cho nhiều ứng dụng), mỗi phân đoạn chứa tối đa 4 Mbyte. Bộ mô tả, mục nhập thư mục trang tương ứng và bảng trang tương ứng có thể được cấp phát và phân bổ đồng thời.



Hình 4.2: Mỗi đoạn sẽ có 1 bảng phân trang

Tổng kết

Quá trình làm việc và nghiên cứu về cách quản lý bộ nhớ của hệ điều hành Linux và **vi xử lý Intel 80486** đã giúp nhóm em rất nhiều trong việc hiểu rõ hơn và củng cố thêm về nội dung được giới thiệu trong bộ môn Nguyên lý hệ điều hành.

Để hoàn thành được báo cáo này, chúng em xin chân thành bày tỏ lời cảm ơn tới thầy ***Phạm Đăng Hải***, người đã cung cấp cho chúng em những kiến thức vô cùng bổ ích, những sự hướng dẫn tận tình và những tiết dạy tuyệt vời và đáng quý ở học phần Nguyên lý hệ điều hành.

Mặc dù chúng em đã đưa ra nỗ lực tìm hiểu và nghiên cứu, song bài báo cáo của chúng em vẫn có thể không trách khỏi những thiếu sót cần được cải thiện và bổ sung. Kính mong nhận sự góp ý của thầy và các bạn.

Tài liệu tham khảo

[1] Intel, Intel486 Microprocessor Family Programmer's Reference Manual.

http://bitsavers.informatik.uni-stuttgart.de/components/intel/_dataBooks/1992_Intel_486_Programmers_Reference_Manual.pdf.