UNIVERSIDADE FEDERAL DE SANTA CATARINA

KAA

Um Núcleo Experimental para um Sistema Operacional Distribuído

Rafael Volkmann

Universidade Federal de Santa Catarina
Departamento de Informática e de Estatística
Curso de Ciência da Computação

TÍTULO : Um Núcleo Experimental para um
Sistema Operacional Distribuído

AUTOR : Rafael Volkmann

ORIENTADOR : Antônio Augusto Medeiros Fröhlich

CO-ORIENTADOR : Thadeu Botteri Corso

BANCA EXAMINADORA : Ana Elisa Schmidt
Antônio Augusto Medeiros Fröhlich
Thadeu Botteri Corso

Florianópolis, 08 de dezembro de 1994.

DEDICAÇÃO

Este trabalho é dedicado em memória de Ulisses Dobnier Daniel.

AGRADECIMENTOS

Agradeço a todos que tornaram possível a realização deste trabalho, especialmente aos meus pais e ao meu orientador Antônio Augusto.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

INTRODUÇÃO

O velho modelo de um único computador central servindo toda uma organização foi rapidamente trocada por vários computadores menores interconectados realizando o mesmo trabalho, caracterizando uma rede de computadores.

As redes de computadores possuem, atualmente, sistemas operacionais em rede ou sistemas operacionais distribuídos. Em um sistema operacional em rede o usuário conhece a existência de vários computadores. Em um sistema operacional distribuído o usuário trabalha como em uma máquina com um único processador.

Em um sistema distribuído, o usuário não sabe onde seus programas estão executando, nem conhece a localização de seus arquivos, e esta transparência de localidade é manipulada automatica e eficientemente pelo sistema operacional.

Um sistema operacional é construído subdividindo-o em servidores mais um núcleo. O núcleo deve ser o mais simples possível, apenas dando suporte a execução de servidores, que implementarão toda a funcionalidade do sistema operacional.

Com as características de distribuição implementadas pelos servidores, o núcleo não difere muito em modelagem dos núcleos já existentes, adicionando-se apenas características de suporte aos servidores.

Este trabalho propõe a realização de um projeto e implementação de um núcleo experimental para um sistema operacional distribuído. O núcleo aqui apresentado não implementa a funcionalidade completa para suporte ao desenvolvimento de um sistema operacional distribuído sobre computadores 80386 e compatíveis. Este núcleo apenas não impede que em futuras implementações o sistema suporte um sistema distribuído.

O núcleo possue a característica de isolar os servidores e as aplicações de um sistema operacional das características dependentes da arquitetura da máquina. Um núcleo deve prover :

- um gerenciamento de processos e *threads*;
- um gerenciamento de memória (gerenciamento de baixo nível);
- um mecanismo de entrada e saída(E/S);
- um mecanismo de comunicação entre processos.

O capítulo 1 apresenta o estado-da-arte referente ao projeto e os aspectos teóricos envolvidos no projeto. O capítulo 2 apresenta uma visão global do sistema, descrevendo as características gerais do projeto.

O capítulo 3 descreve de forma sucinta a arquitetura do microcomputador 80386 executando em modo protegido (modo supervisor).

O capítulo 4 apresenta a necessidade e as características do carregador do sistema. Finalmente, o capítulo 5 apresenta o núcleo propriamente dito.

2. ORGANIZAÇÃO DO SISTEMA

O trabalho tem como escopo um projeto experimental, servindo de teste e suporte para futuras aplicações. A escolha de utilização de máquinas 80386 foi devido sua grande disponibilidade e difusão e vasta quantidade de material bibliográfico. Os compiladores utilizados no projeto foram *Borland C++ 3.1* (*DOS*) e *GNU C++ 2.1*(*Linux*).

Como o núcleo deve dar suporte a um sistema multitarefa, e deve implementar características pertinentes a um sistema operacional distribuído, ele não poderia ser executado no modo real 80386, sendo então projetado para utilizar o modo protegido 80386, onde se pode usufruir as características de multitarefa implementadas em hardware.

Para que o sistema fosse executado independente de plataformas já existentes, por exemplo executando sob o *DOS*, houve a necessidade de se escrever um carregador para o sistema. O carregador do sistema tem a função de ler o código relativo ao núcleo (a partir de um disco de carga) alocá-lo em memória e executá-lo.

O modelo básico para construção de sistemas operacionais é subdividí-lo em vários servidores e um núcleo. O núcleo deve ser o mais simples possível, implementando apenas o essencial para a execução dos servidores.

O núcleo realiza toda a inicialização do sistema e possue dados e código executando em modo supervisor. Os serviços do núcleo são utilizados através de interrupções de software ou na ocorrência de uma interrupção de hardware. Cada controlador de dispositivo externo deve ser implementado como um servidor, assim, as interrupções de hardware devem ser convertidas em mensagens para seus respectivos servidores.

O núcleo primeiramente monta as estruturas de dados e executa as instruções necessárias para a troca do modo real para o modo protegido. Para a alocação de segmentos para as tarefas é feito um gerenciamento de memória em baixo nível.

Todas as exceções são tratadas e a única interrupção de hardware utilizada pelo próprio núcleo é a interrupção do relógio. Esta interrupção é usada para implementar o escalonamento dos processos.

3. ARQUITETURA 80386

O 80386 é fundamentalmente um computador multitarefa. Ele possui estruturas e mecanismos em hardware que dão suporte a criação de um sistema multitarefa. A implementação de tais estruturas em hardware possibilita a troca rápida do contexto das tarefas, e a característica de se ter várias tarefas executando em um computador está centrado basicamente numa estrutura chamada TSS (*Task State Segment*). Gerenciamento de memória, proteção, exceções e interrupções são mecanismos baseados em tarefas.

No 80386 o mecanismo de gerenciamento de memória traduz os endereços lógicos em endereços físicos. O gerenciamento de memória pode ser dividido em :

- segmentação - permite o isolamento de tarefas;
- paginação - implementa memória virtual (opcional e não abordado no projeto).

A segmentação traduz os endereços lógicos em endereços lineares. A paginação traduz os endereços lineares em endereços físicos. As facilidades de paginação e segmentação do 80386 permitem implementar qualquer modelo de memória comumente usado, incluindo *flat*, segmentado, paginado, e segmentado-paginado. O gerenciamento de memória do 80386 utiliza-se de tabelas (*Global Descriptor Table* e *Local Descriptor Table*) que descrevem os segmentos de memória sendo utilizados.

O 80386 possui um complexo mecanismo de proteção. Este mecanismo pode isolar o sistema operacional das aplicações, tão bem quanto ajudar a detectar e identificar erros ocorridos em uma determinada tarefa. Para isto o 80386 possui mecanismos para verificar acessos a memória e execução de instruções de acordo com os critérios de proteção. Estes mecanismos podem ser usados ou ignorados de acordo com o sistema em questão.

No 80386, interrupções e exceções são eventos não programados que alteram o fluxo normal de execução das instruções em uma tarefa. As interrupções independem da execução de

instruções e tipicamente são disparadas por dispositivos externos. Exceções resultam basicamente da execução de instruções. Há duas fontes para interrupções externas e duas para exceções :

1. Interrupções
    - mascaráveis, que são sinalizadas via pino INTR;
    - nao-mascaráveis, que são sinalizadas via pino NMI.

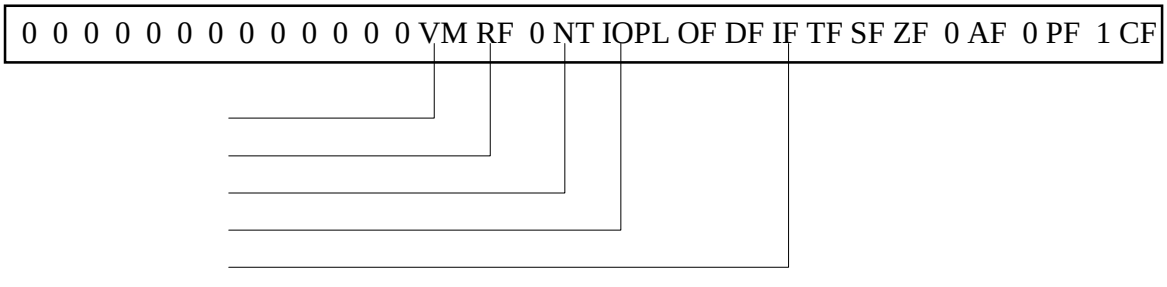
2. Exceções
    - de detecção de erros (*faults*, *traps* e *aborts*)
    - programadas. As instruções INTO, INT 3, INT 'n' e BOUND podem disparar exceções. Estas instruções são freqüentemente chamadas de interrupções de software, mas o processador as manipula como exceções.

3.1 REGISTRADORES

Os registradores no 80386, excetuando-se os já conhecidos do 8086, designados para uso por programadores de sistemas são os seguintes :

- EFLAGS
- Registradores de Gerenciamento de Memória
- Registradores de Controle
- Registradores de *Debug*
- Registradores de Teste

A figura 3.1 mostra o registrador EFLAGS. Ele controla entrada/saída, interrupções mascaráveis, *debug*, troca de tarefas, e habilitação do modo virtual 8086, entre outras características de um sistema multitarefa. Maiores detalhes podem ser encontrados em [INT 86].



Virtual 8086 Mode
Resume Flag
Nested Task Flag
I/O Privilege Level
Interrupt Flag

Figura 3.1 - Formato do registrador EFLAGS

Quatro registradores do 80386 alocam estruturas de dados que controlam o gerenciamento de memória segmentado :

- GDTR - *Global Descriptor Table Register*
- LDTR - *Local Descriptor Table Register*
- IDTR - *Interrupt Decriptor Table Register*
- TR - *Task Register*

A figura 3.2 mostra o formato dos registradores de controle do 80386. Estes registradores são acessíveis para os programadores de sistemas somente via instruções MOV, que permitem carregá-los e armazená-los em registradores de propósito geral.

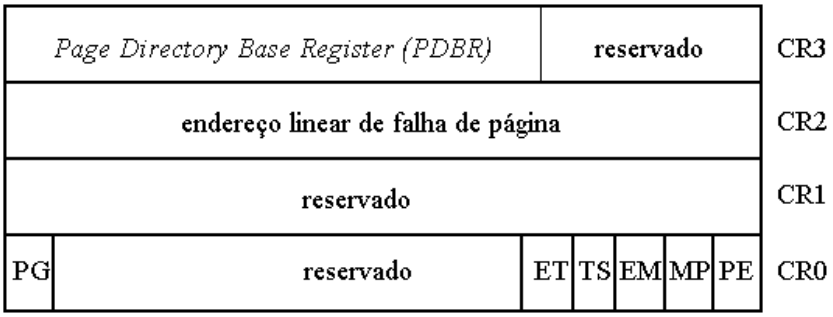


Figura 3.2 - Registradores de Controle

O registrador CR0 contem *flags* de controle do sistema, que controlam ou indicam condições que se aplicam ao sistema como um todo, e não para uma tarefa individual.

EM (*Emulation*)	indica se as funções do coprocessador estão sendo emuladas
ET (*Extension Type*)	indica o tipo de coprocessador presente (80287 ou 80387)
MP (*Math Present*)	controla a função da instrução WAIT, que é usada para coordenar o coprocessador
PE (*Protected Enable*)	indica se o processador esta executando em modo protegido
PG (*Paging*)	indica se o processador utiliza tabelas de página para traduzir endereço linear em endereço físico
TS (*Task Switched*)	utilizado para troca de tarefas

CR2 é usado para manipular falhas de página quando PG está setado. O processador armazena em CR2 o endereço linear que disparou a falha. CR3 é usado quando PG está setado, e contem o registrador do diretório de páginas utilizado na paginação.

Maiores informações sobre os registradores acima descritos, os registradores de teste, e os registradores de *debug* são encontrados em [INT 86].

3.2 GERENCIAMENTO DE MEMÓRIA: SEGMENTAÇÃO E PROTEÇÃO

A figura 3.3 mostra em detalhes como o 80386 converte endereços lógicos em endereços lineares. Para realizar esta tradução o processador utiliza as seguintes estruturas de dados :

- Descritores
- Tabelas de descritores
- Seletores
- Registradores de segmento

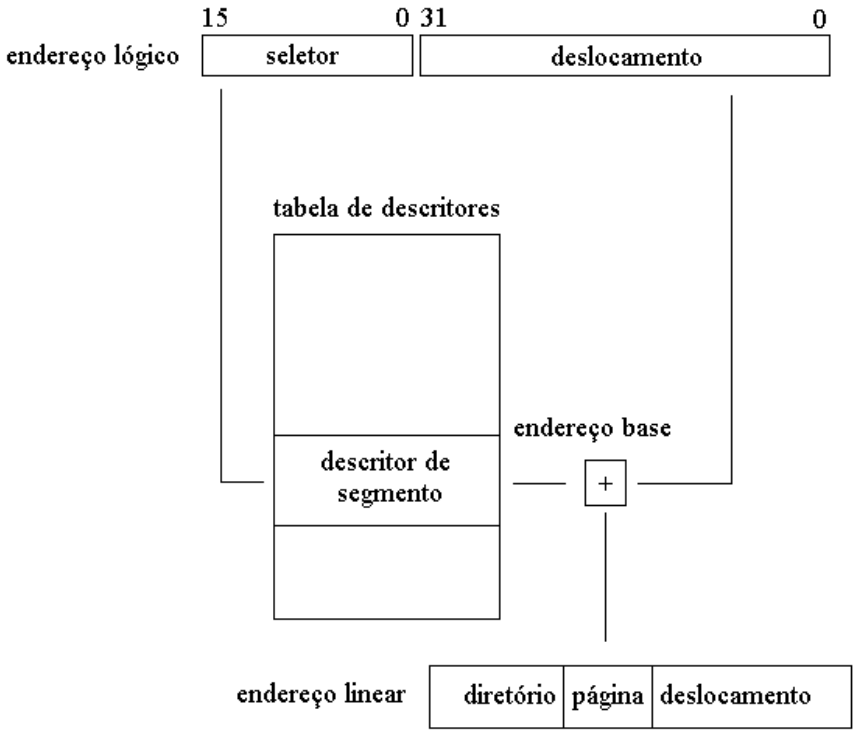


Figura 3.3 - Conversão de endereços

3.2.1 DESCRITORES

Os descritores de segmentos provem ao processador os dados necessários para mapear um endereço lógico em endereço linear. A figura 3.4 ilustra os dois formatos de descritores gerais. Todos os tipos de descritores de segmento possuem um destes formatos. Os campos são :

- BASE : define a localização do segmento no espaço de endereçamento linear de 4 Gbytes.
- LIMIT : define o tamanho do segmento (20 bits). O processador interpreta o campo LIMIT dependendo do conteúdo do bit de granularidade :
    1. em unidades de 1 byte, definindo um limite superior de 1 Mbyte
    2. em unidades de 4 Kbytes, definindo um limite superior de 4 Gbytes.

- GRANULARITY BIT : especifica a unidade utilizada no campo LIMIT.
- DPL (*Descriptor Privilege Level*) : usado pelo mecanismo de proteção.
- SEGMENT-PRESENT BIT : quando a paginação está habilitada indica a presença de um segmento em memória.
- TYPE : este campo indica que tipo de descritor está sendo utilizado. Possui diferentes configurações para os diversos tipos de descritores. Os descritores de dados e descritores de código tem o campo TYPE definido como :

  DATA : 1 0 E W A

  E - expand down (usado para segmentos de pilha)

  W - indica que o segmento permite escrita

  A - accessed bit

  CODE : 1 1 C R A

  C - segmento conformante

  R - indica que o segmento permite leitura

  A - accessed bit
- ACCESSED BIT : o processador seta este bit quando o segmento é acessado (usado em sistema com memória virtual).
- AVL BIT : avaliável para programadores de sistema.

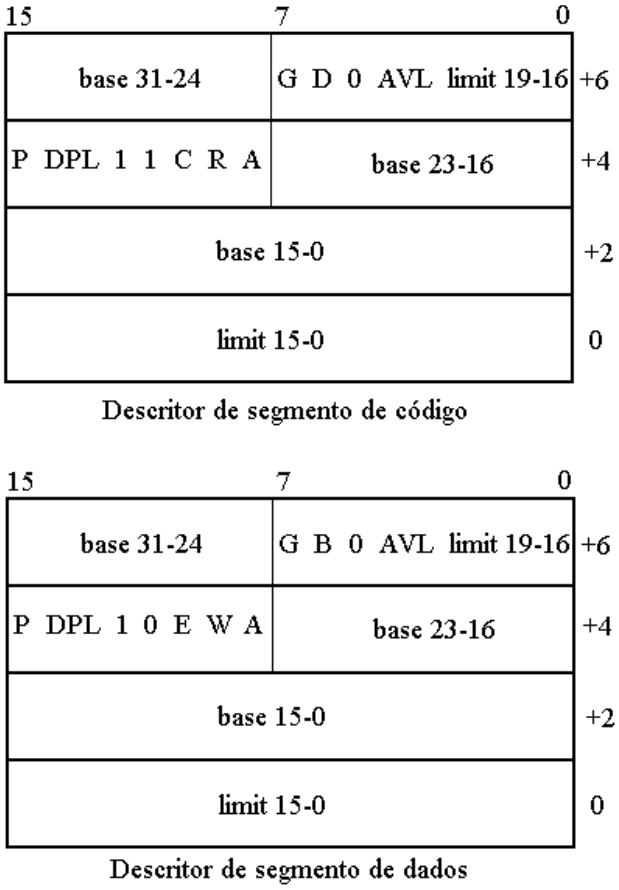


Figura 3.4 - Descritores de segmento de código e de dados

3.2.2 TABELAS DE DESCRITORES

Descritores de segmentos são armazenados em 2 tipos de tabelas de descritores :

- *Global Descriptors Table* (GDT);
- *Local Descriptors Table* (LDT).

Uma tabela de descritores é simplesmente um vetor de memória que contem descritores (8 bytes), como mostrado na figura 3.5. Uma tabela de descritores é variável em comprimento e pode conter 8192 (2^{13}) descritores. A primeira entrada da GDT (índice=0) não é

utilizada pelo processador e um seletor apontando para a GDT com índice 0 é chamado de seletor nulo, e uma tentativa de acessá-lo resulta em uma exceção.

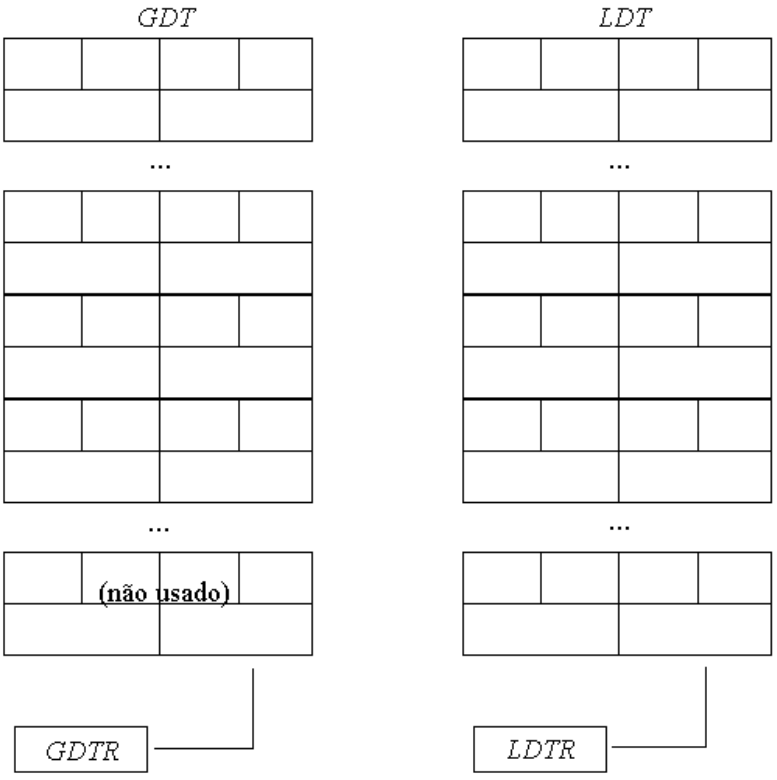


Figura 3.5 - Tabelas de descritores

Os mecanismos de segmentação utilizando GDT e LDT são mostrados nas figuras 3.6 e 3.8.

Os registradores GDTR e LDTR apontam para as tabelas de descritores globais e tabelas de descritores locais, respectivamente. O GDTR contem um endereço linear de 32 bits da GDT e um limite de 16 bits. O LDTR contem um seletor para a LDT corrente, já que podem existir varias LDT's em um sistema. Este seletor deve referenciar um descritor de LDT na GDT.

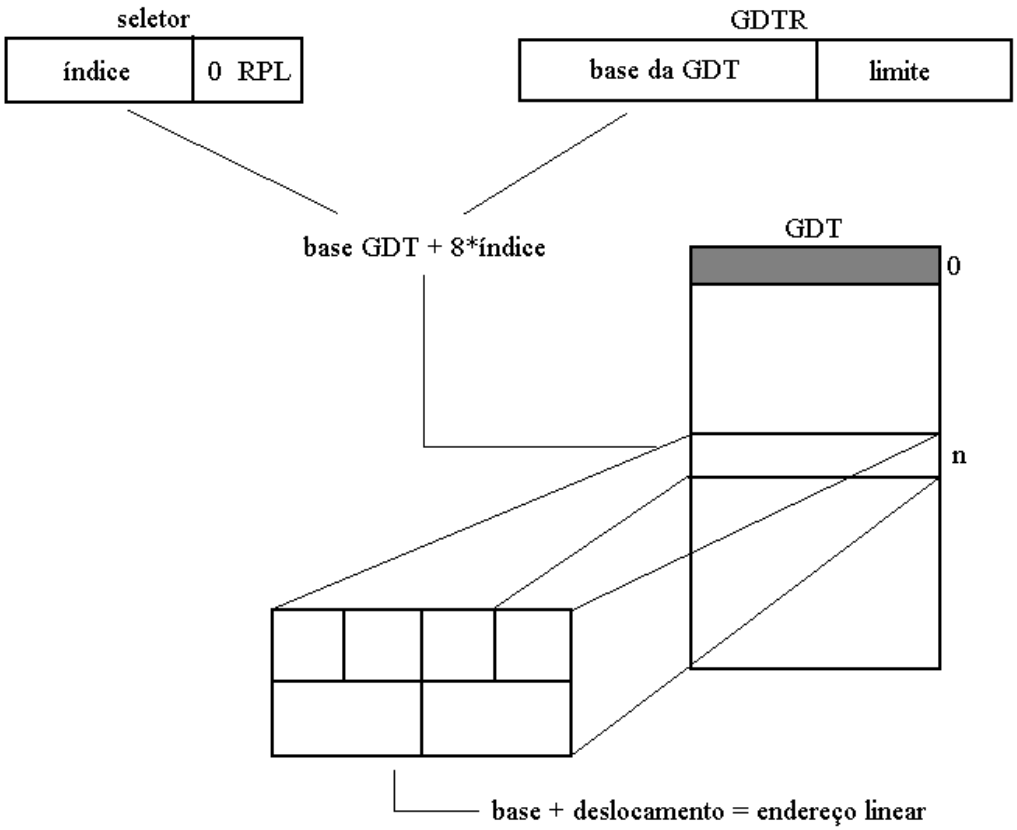


Figura 3.6 - Endereçamento utilizando GDT.

3.2.3 SELETORES

O seletor indexa um descritor em uma tabela de descritores. A figura 3.7 mostra o formato de um seletor.

- índice : seleciona um dos 8192 descritores em uma tabela.
- indicador de tabela (TI - Table Indicator) : especifica a que tabela o seletor se refere. Zero indica a GDT, e um indica a LDT corrente.
- RPL : usado pelo mecanismo de proteção

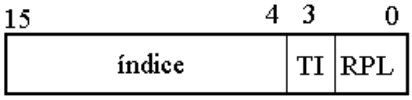


Figura 3.7 - Formato de um seletor.

3.2.4 REGISTRADORES DE SEGMENTO

O 80386 armazena as informações dos descritores em registradores de segmento, eliminando a necessidade de consultar uma tabela de descritores a cada acesso a memória.

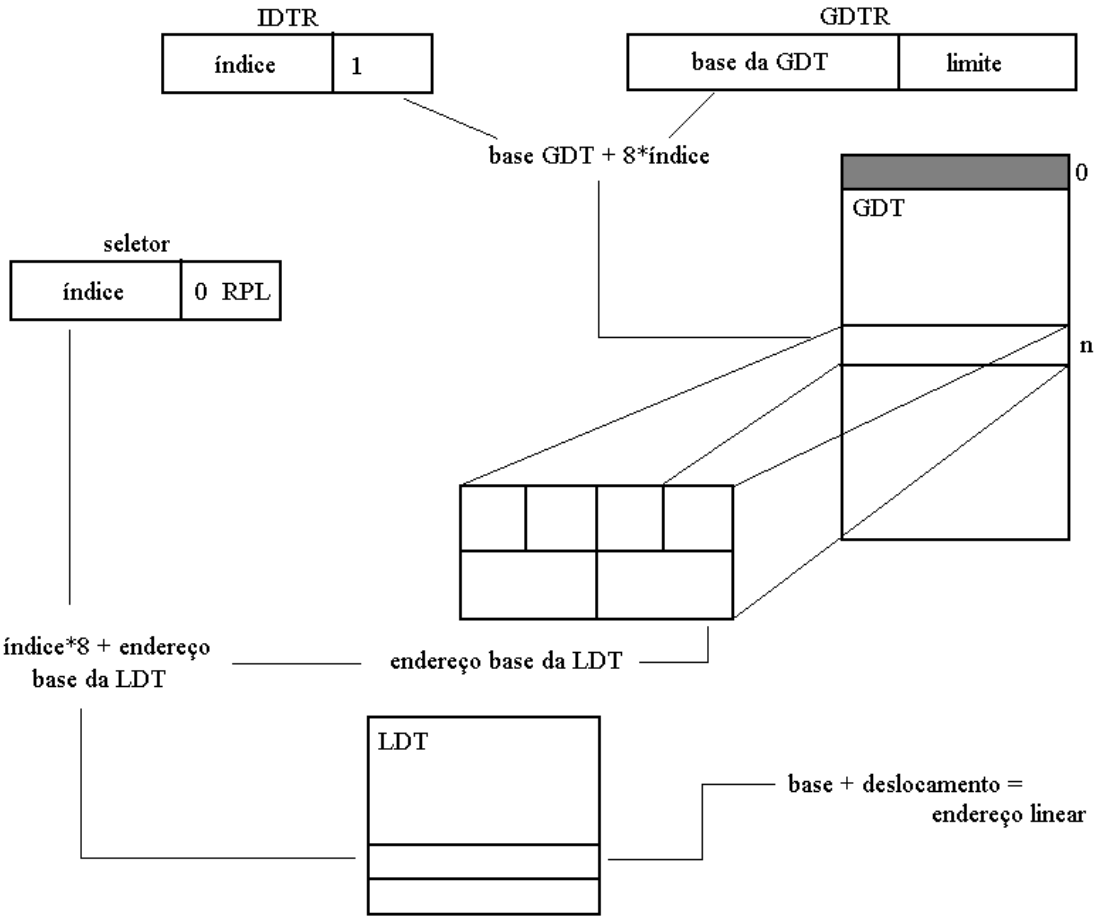


Figura 3.8 - Endereçamento utilizando LDT.

Cada registrador de segmento possui uma porção “visível” e uma porção “invisível”. As operações que carregam estes registradores estão em uma das duas classes :

- Instruções de carga direta, por exemplo MOV, POP, ...
- Instruções de carga implícita, por exemplo *far* CALL e JMP. Estas instruções referem-se ao registrador CS, carregando-o com um novo valor.

Usando estas instruções, o programa carrega a parte “visível” do registrador de segmento com um seletor de 16 bits. O processador automaticamente carrega o descritor associado na parte “invisível”. Com este mecanismo, referências a dados em um mesmo segmento, não precisam realizar a tradução de endereços repetidamente.

3.2.5 PROTEÇÃO

A proteção de hardware do 80386 é parte integral do gerenciamento de memória realizado pela máquina. Cada referência a memória é verificada pelo hardware se satisfaz os critérios de proteção.

Toda verificação é feita a cada ciclo de memória e qualquer violação resulta em uma exceção. Como as verificações são feitas concorrentemente com a formação do endereço, portanto, não há penalidades na performance.

O conceito de privilégio é central para os aspectos de proteção. Aplicado a procedimentos, privilégio é o grau de confiabilidade dado ao procedimento de forma que ele não afete outros procedimentos ou dados. Aplicados a dados, privilégio é o grau de proteção de uma estrutura de dados que um procedimento deve ter para acessá-la.

Proteção no 80386 tem cinco aspectos :

- verificação de tipo;
- verificação de limite;
- restrição do domínio de endereçamento;
- restrição dos pontos de entrada em procedimentos;
- restrição do conjunto de instruções.

O segmento é a unidade de proteção, e os descritores de segmentos armazenam as informações de proteção. As verificações de proteção são feitas automaticamente pela CPU quando um seletor de um descritor de segmento é carregado em um registrador de segmento e em cada acesso ao segmento.

O 80386 apresenta em sua arquitetura 4 níveis de privilégio distintos, sendo que a maioria dos sistemas, inclusive este, utiliza-se de somente 2 níveis (nível 0 para execução do núcleo e nível 3 para as demais tarefas). Os seguintes “objetos” reconhecidos pelo processador contem níveis de privilégio :

- os descritores contem um campo chamado DPL (*Descriptor Privilege Level*);
- os seletores contem um campo chamado RPL (*Requestor's Privilege Level*). O RPL é intendido para representar o nível de privilégio do procedimento que originou o seletor;
- o registrador CS contem armazenado o CPL (*Current Privilege Level*).

Os detalhes dos critérios utilizados nos mecanismos de proteção são encontrados em [INT 86].

3.3 MULTITAREFA

Para prover um sistema multitarefa eficiente e protegido o 80386 possui estruturas de dados especiais. Os registradores e estruturas de dados que suportam multitarefa são :

- segmento de estado de tarefa;
- descritor de segmento de estado de tarefa;
- registrador de tarefa;
- descritor de portão de tarefa.

Com estas estruturas o 80386 pode rapidamente realizar a troca de tarefas, salvando o contexto da tarefa original para que esta possa ser reinicializada mais tarde.

3.3.1 SEGMENTO DE ESTADO DE TAREFA

Todas as informações que o processador necessita a fim de gerenciar uma tarefa são armazenadas em um tipo especial de segmento, um segmento de estado de tarefa (TSS - *Task Segment State*). A figura 3.9 mostra o formato de uma TSS executando em tarefas no 80386.

mapa de E/S	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	T
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	LDT	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	GS	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	FS	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	DS	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	SS	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	CS	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	ES	
EDI		
ESI		
EBP		
ESP		
EBX		
EDX		
ECX		
EAX		
EFLAGS		
EIP		
CR3		
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	SS2	
ESP2		
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	SS1	
ESP1		

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	SS0
ESP0	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	*back link*

Figura 3.9 - Segmento de estado de tarefa.

Os campos de uma TSS são classificados em :

1. Conjunto dinâmico que o processador atualiza a cada troca de tarefa. Este conjunto inclui :
   - registradores de propósito geral (EAX, EBX, ECX, EDX, ESP, EBP, ESI, EDI);
   - registradores de segmento (CS, DS, ES, FS, GS, SS);
   - registrador de flags (EFLAGS);
   - apontador de instruções (EIP);
   - o seletor da TSS da tarefa executando anteriormente.

2. Conjunto estático que o processador lê e não altera. Este conjunto inclui :
   - o seletor da LDT da tarefa;
   - o registrador que contem o endereço base do diretório de páginas da tarefa (PDBR - CR3);
   - apontadores para as pilhas dos níveis de privilégio 0 à 2;
   - o bit T, que coloca o processador em modo *debugging*;
   - o mapa de entrada/saída do 80386.

3.3.2 DESCRITOR DE SEGMENTO DE ESTADO DE TAREFA

O segmento de estado de tarefa, como qualquer outro segmento, é definido por um descritor. A figura 3.10 mostra o formato de um descritor de TSS.

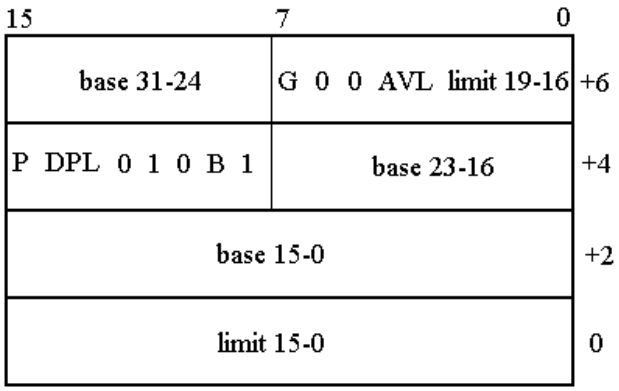


Figura 3.10 - Descritor de uma TSS

O bit B indica que a tarefa está ocupada. Tarefas não são reentrantes e o bit B permite ao processador detectar uma tentativa de troca de tarefa para uma tarefa já ocupada.

Os campos BASE, LIMIT, DPL e os bits G e P tem as mesmas funções de outros descritores. O campo LIMIT deve ter um valor maior ou igual a 103, por causa do tamanho mínimo de uma TSS. Descritores de TSS podem somente residir em GDT.

3.3.3 REGISTRADOR DE TAREFA

O registrador de tarefa identifica a tarefa em corrente execução, e aponta para uma TSS (figura 3.11). O registrador de tarefa tem uma porção visível e uma porção invisível idêntica aos registradores de segmento.

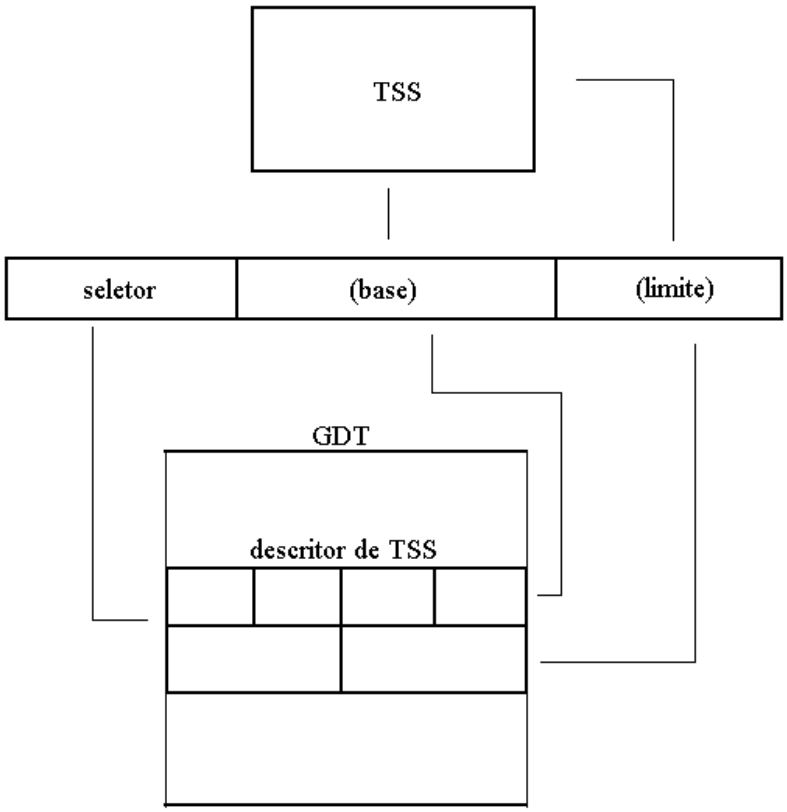


Figura 3.11 - Registrador de tarefa.

3.3.4 DESCRITOR DE PORTÃO DE TAREFA

Um descritor de um portão de tarefa provê uma referência indireta e protegida para uma TSS. A figura 3.12 ilustra o formato de um portão de tarefa. O campo SELETOR de um portão de tarefa refere-se a um descritor de TSS. O valor de RPL neste seletor não é usado pelo processador.

3.3.5 TROCA DE TAREFAS

O 80386 troca a execução para uma outra tarefa nos seguintes casos :

- a corrente tarefa executa um JMP ou CALL que refere-se a um descritor de TSS;
- a corrente tarefa executa um JMP ou CALL que refere-se a um portão de tarefa;
- uma interrupção ou exceção vetorizam para um portão de tarefa na IDT;
- a corrente tarefa executa um IRET quando o flag NT (*Nested Task*) está setado.

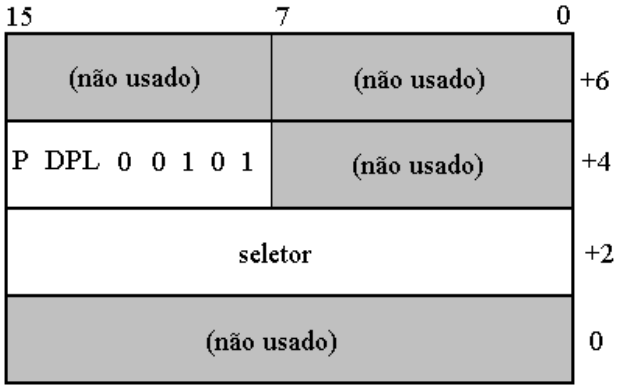


Figura 3.12 - Descritor de portão de tarefa.

3.3.6 SEQÜÊNCIA DE TAREFAS

O campo de *back-link* e o *flag* NT em EFLAGS permitem ao 80386 retornar automaticamente para a tarefa interrompida. Quando uma instrução CALL, uma interrupção, uma exceção causam uma troca de tarefa o 80386 automaticamente preenche o campo de *back-link* da nova TSS com o seletor da antiga TSS, e ao mesmo tempo, seta o *flag* NT no registrador de *flags* da nova tarefa.

3.4 ENTRADA e SAÍDA

Dois mecanismos provem proteção para as funções de entrada e saída :

- o campo IOPL no registrador de *flags*;
- o mapa de permissão de E/S na TSS.

Quando ocorre uma instrução de E/S, o processador primeiro verifica se o CPL é menor ou igual ao IOPL. Se a condição for verdadeira, a operação procede. Caso contrário, o processador verifica o mapa de bits da TSS da tarefa.

Cada bit no mapa corresponde a uma porta de E/S. Se o bit correspondente a porta de E/S está em zero a operação procede.

3.5 INTERRUPÇÕES E EXCEÇÕES

O processador associa um número identificador a cada tipo diferente de interrupção ou exceção. A NMI e as exceções reconhecidas pelo processador são identificadas no campo de 0 a 31. Nem todos os números são usados pelo 80386, e são reservados pela Intel para aplicações futuras.

Os identificadores das interrupções mascaráveis são determinados por controladoras de interrupções externas (Intel 8259-A) e comunicadas ao processador durante a seqüência de reconhecimento de interrupção do processador. Os números atribuídos ao 8259-A PIC podem ser especificados por software, e podem estar nos campos de 32 a 255. A tabela 3.1 mostra a relação de interrupções e exceções.

Identificador	**Descrição**
0	Erro de divisão
1	Exceções de *debug*
2	Interrupção não-mascarável
3	Ponto de parada

4	*Overflow*
5	Verificação de BOUND
6	Código inválido
7	Coprocessador não avaliável
8	Falha dupla
9	(reservado)
10	TSS inválida
11	Segmento não presente
12	Exceção de pilha
13	Proteção geral
14	Falha de página
15	(reservado)
16	Erro de coprocessador
17 - 31	(reservado)
32 - 255	Avaliável para interrurpções externas via pino INTR

Tabela 3.1 - Interrupções e exceções.

3.5.1 TABELA DE INTERRUPÇÃO

A IDT (*Interrupt Descriptor Table*) é a ligação entre os números de interrupções e exceções e o manipulador que o sistema operacional possui para tratar aquele tipo de interrupção ou exceção.

O processador usa o número como um índice para a IDT. Como uma LDT ou GDT, a IDT é uma tabela de descritores e pode estar localizada em qualquer espaço de endereçamento linear. A IDT e seu registrador IDTR são mostrados na figura 3.13. Uma IDT não pode conter mais que 256 descritores, limitado ao numero de exceções e interrupções válidas.

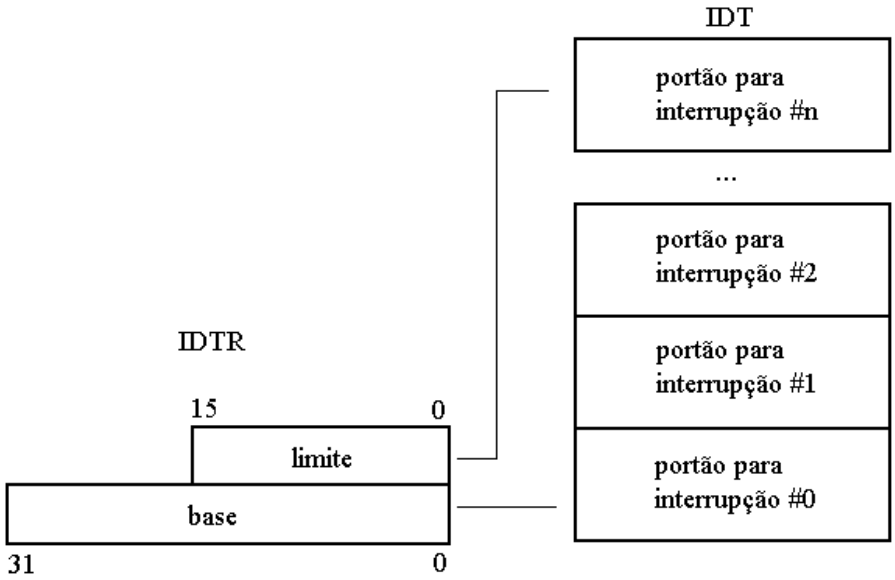


Figura 3.13 - Registrador e tabela da IDT.

3.5.2 DESCRITORES DA IDT

A IDT pode conter 3 tipos de descritores :

- *task gate*;
- *interrupt gate*;
- *trap gate*.

A figura 3.14 ilustra o formato dos portões no 80386.

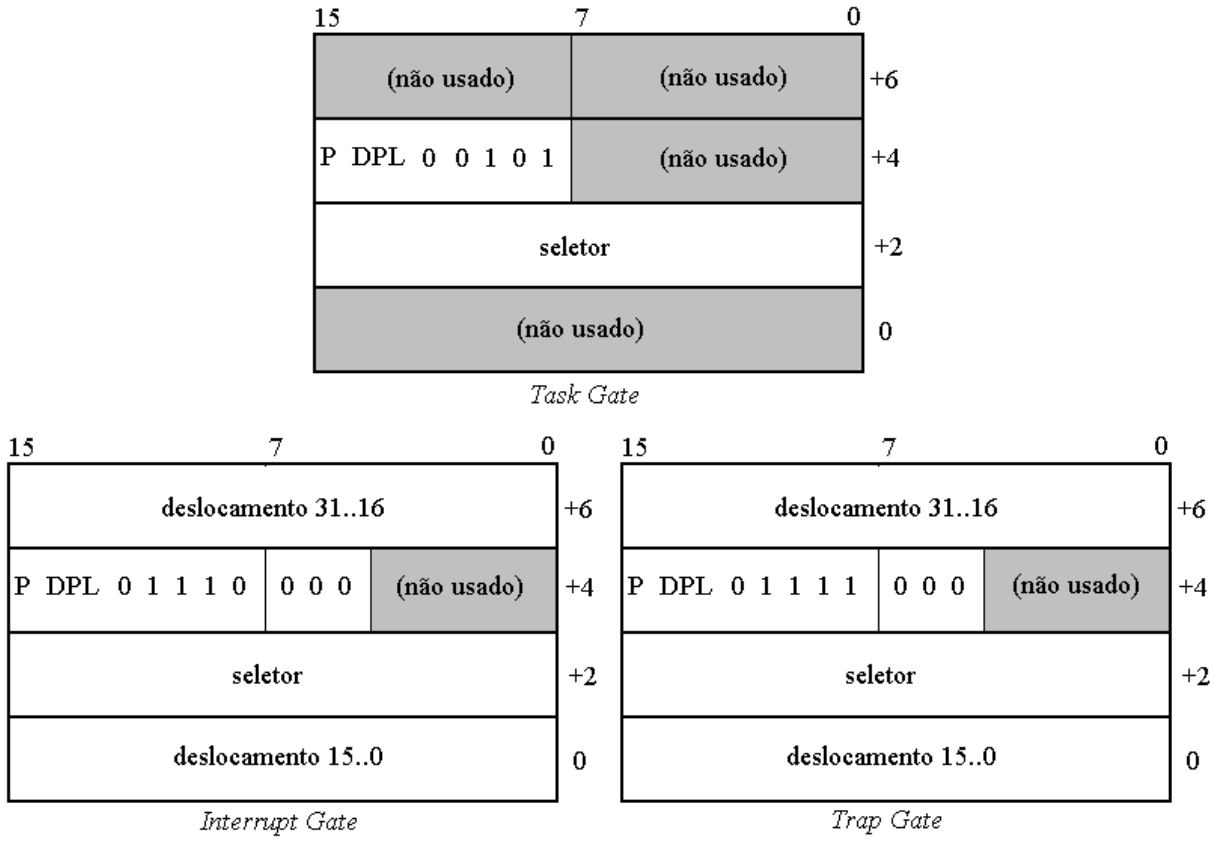


Figura 3.14 - Descritores da IDT.

Se o processador encontra um *task gate* no tratamento de uma interrupção, ele causa uma troca de tarefa similar a uma instrução CALL para um *task gate*. O seletor do portão aponta para um descritor de TSS na GDT. O retorno para a tarefa interrompida ocorre com a execução da instrução IRET.

Um *interrupt gate* ou *trap gate* apontam indiretamente para um procedimento que irá ser executado no contexto da tarefa corrente. O seletor aponta para um descritor de segmento executável na GDT ou LDT corrente. O campo OFFSET aponta para o início do procedimento que manipula a interrupção.

Detalhes de tratamento de pilha, códigos de erro, *flags* e proteção são encontrados em [INT 86] e [INT 87a].

Em geral, manipuladores de exceções devem ser implementados como procedimentos, tal que possam manipular a exceção no contexto da tarefa que a gerou. Manipuladores de interrupções são candidatos a implementação em tarefas separadas, executando em um contexto distinto, próprio para o tratamento da interrupção.

3.6 INICIALIZAÇÃO PARA O MODO PROTEGIDO

Inicialização é a seqüência de instruções que um sistema operacional deve executar para iniciar a primeira tarefa. Quando sua linha de RESET é ativada, o 80386 responde no modo real de endereçamento, e o sistema operacional deve realizar a troca do modo real para o modo protegido.

Antes de qualquer troca para o modo protegido, o registrador da GDT (GDTR) deve apontar para uma GDT válida, e as interrupções devem estar desabilitadas. O IDTR, a pilha e as TSS's podem ser carregadas tanto em modo real quanto em modo protegido.

Para mudar o processador para o modo protegido basta setar o bit PE da MSW (*Machine Status Word*) do registrador CR0. Imediatamente após a troca, uma instrução JMP deve ser executada para que o processador descarregue as instruções de pré-busca na fila de execução. Uma instrução JMP força o processador a descartar uma informação inválida.

4. CARREGADOR DO SISTEMA

O núcleo desenvolvido no projeto tem por objetivo ser testado e carregado facilmente sob computadores 80386. Decidiu-se que a forma mais eficiente e independente de sistema seria carregá-lo a partir de um disco de carga. Assim, um pequeno software (carregador do sistema) teria a função de ler o código relativo ao núcleo, residente em disco, alocá-lo em uma área de memória e finalmente executá-lo.

Utilizando um disco de carga houve a possibilidade de testar a execução de diversas tarefas no sistema. As código relativo às tarefas podem ser gravadas no disco e podem ser carregadas em memória antes da execução do núcleo. Assim, as tarefas que devem ser executadas estão isoladas completamente do código do núcleo, deixando-o mais compacto e eficiente, convergindo para o conceito de um núcleo pequeno e funcional.

Quando o 80386 é ligado ou reinicializado são executadas rotinas de iniciação e teste do BIOS, mostrados abaixo :

- teste do processador;
- teste da memória que contem o BIOS;
- teste do *timer* 8253;
- teste do controlador de DMA;
- teste da memória RAM;
- iniciação dos controladores de interrupções;
- iniciação do vetor de interrupção;
- leitura das chaves de configuração;
- teste da interface de vídeo;
- teste do teclado;
- procura de expansões;
- iniciação da interface de disco;
- verificação da presença de interfaces seriais e paralelas;

- desvio para a leitura do carregador do sistema.

O leitor do carregador do sistema lê o setor 1 da face 0 da trilha 0 para o endereço 07C00h e executa um desvio para 0:7C00h. O mapeamento do disco deveria ter a configuração mostrada na figura 1.

Como o carregador do sistema ocupa um espaço maior que um setor (512 bytes) de disco, foi escrito um programa (*bootstrap*) somente para ler o carregador do disco e executá-lo em memória, pequeno o suficiente a ser colocado em um único setor.

Desta forma o carregador do sistema poderia ser alocado em qualquer setor do disco, o mesmo aplicando-se ao núcleo e às tarefas a serem executadas posteriormente.

Com esta flexibilidade de alocação a estrutura do disco pode ser compatibilizada com o sistema de arquivos distribuídos PYXIS, descrito em [___], para qual o núcleo possui funcionalidade.

Para realizar a leitura dos setores do disco, o *bootstrap* utiliza-se unicamente das funções disponíveis pelo BIOS. O código correspondente ao *bootstrap* é mostrado em anexo e as funções do BIOS são encontradas em [4].

Com a flexibilidade de alocação de quaisquer setores do disco para o carregador do sistema, para o núcleo e para as tarefas, houve a possibilidade da compatibilização das estruturas do disco com o sistema de arquivos PYXIS. O PYXIS requer a existência de uma tabela de arquivos no setor 2 da face 0 da trilha 0 do disco. Esta tabela contem informações sobre a localização de arquivos no disco, entre outras informações.

Desta forma, o *bootstrap* carrega o *carregador do sistema* para memória através das informações contidas na primeira entrada da tabela de arquivos do disco. O *carregador do sistema*, por sua vez, carrega os demais arquivos existentes na tabela, que são o núcleo e as tarefas a serem executadas.

Com a carga das tarefas a partir do *carregador do sistema*, o núcleo não executa nenhuma tarefa no sentido de interfaceamento com dispositivos externos, como com o disco. O código referente ao *carregador do sistema* é encontrado em anexo.

REFERÊNCIAS BIBLIOGRÁFICAS

[EGG 83] EGGEBRECHT, Lewis C., *Interfacing to the IBM Personal Computer*, Howard W. Sams & Co., 1983.

[INT 86] INTEL Corporation, *80386 Programmer's Reference Manual*, 1986.

[INT 87a] INTEL Corporation, *80386 System Software Writer's Guide*, 1987.

[INT 87b] INTEL Corporation, *Hardware Reference Manual*, 1987.

[QUA 87] QUADROS, Daniel G. A., *PC Assembler; usando o BIOS*, Rio de Janeiro: Campus, 1987.

[TAN 89] TANENBAUM, A. S., *Computer networks*, Englewood Cliffs: Prentice-Hall, 1989.

[TAN 92] TANENBAUM, A. S., *Modern operating systems*, Englewood Cliffs: Prentice Hall, 1992.

[TIS 92] TISCHER, Michael, *PC Intern - System Programming*, Abacus, 1992.