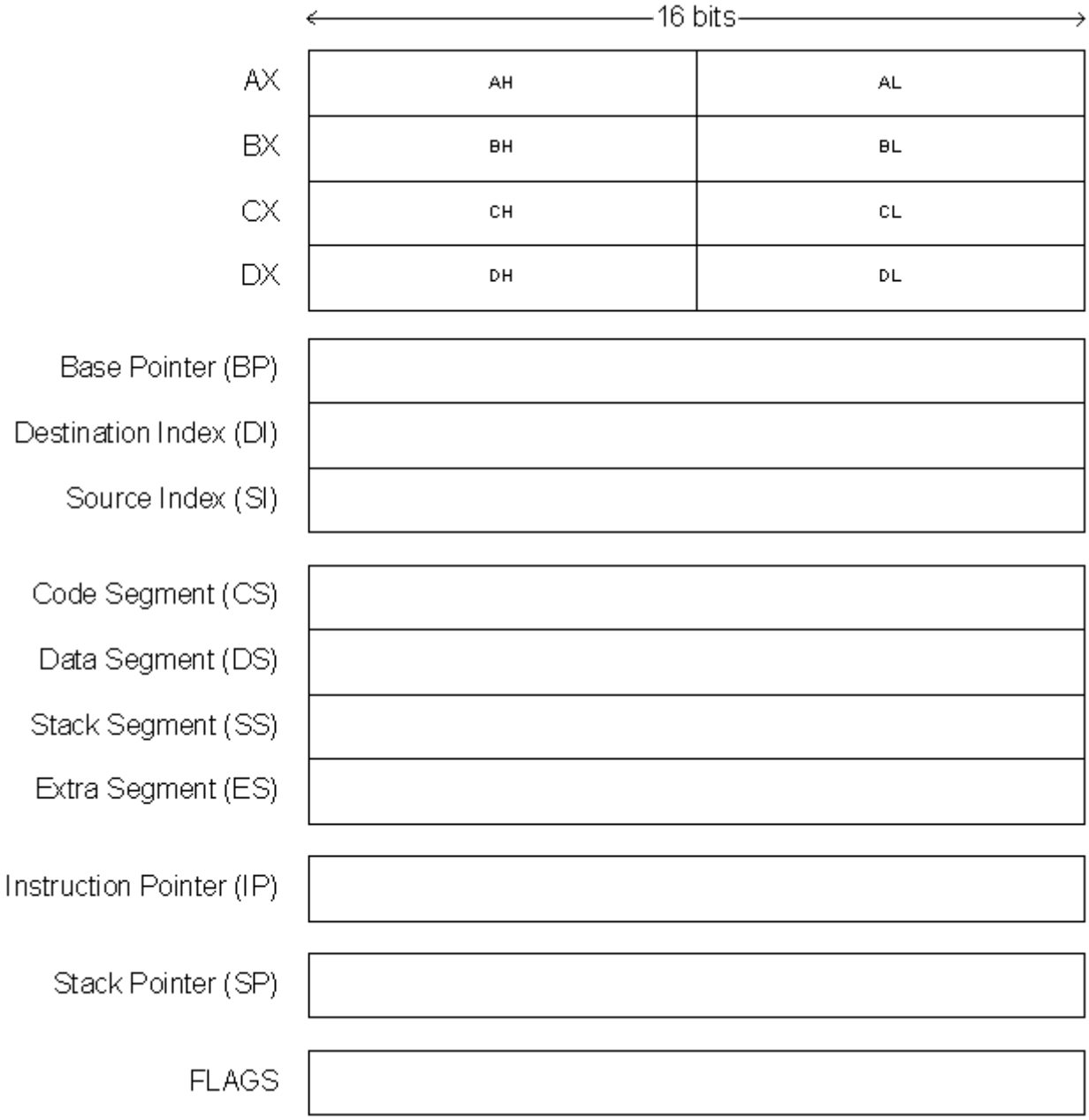
Intel 80x86

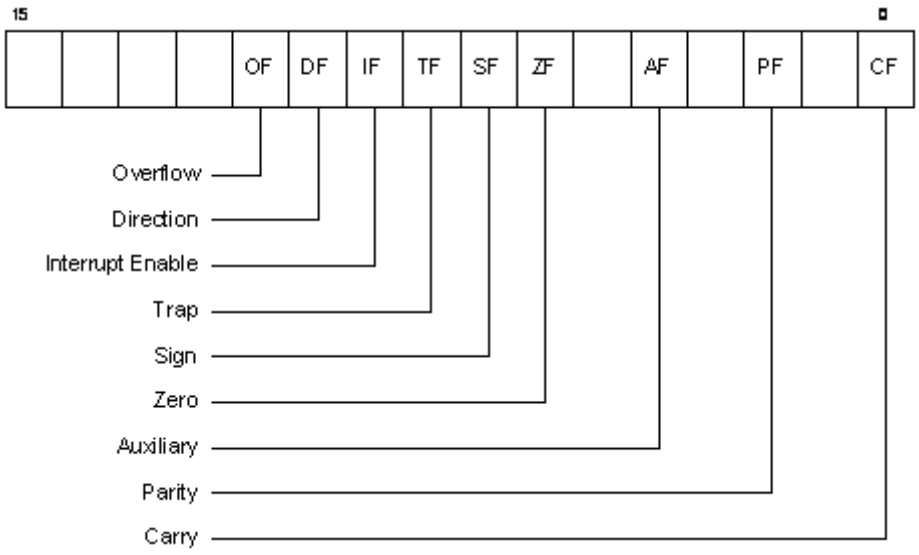
Intel 8086

Registos

Tamanho: 16 bits (word de a6 bits), sendo que quarto registos podem ser acedidos através das suas partes mais e menos significativa (ex. AH e AL são respectivamentes os 8 bits mais e menos significativos do registo AX).



Flags



Segmentação

Endereçamento

As posições de memória a aceder (ler ou escrever) são definidas pelo programa. O resultado do programa depende das características do processador. Em particular, a largura do barramento de endereços (bits) está directamente relacionada com a quantidade de memória a ser acedida. Quanto mais largo este for, maior é o endereço que pode ser especificado, logo maior é a quantidade de memória que pode ser acedida.

Largura de barramento de endereços (n)	Memória endereçável (2^n)
8	256B
16	64KB
20	1024KB = 1MB
32	4GB

Outro dos factores importantes é o tamanho dos registos do processador (word), pois condiciona o cálculo dos valores das posições de memória.

Assim, basicamente o acesso à memória é uma função de dois elementos:

- Tamanho da word;
- Largura do barramento de endereços.

No entanto, existe a necessidade de aceder a memória cujos endereços vão para além da capacidade de representação da word.

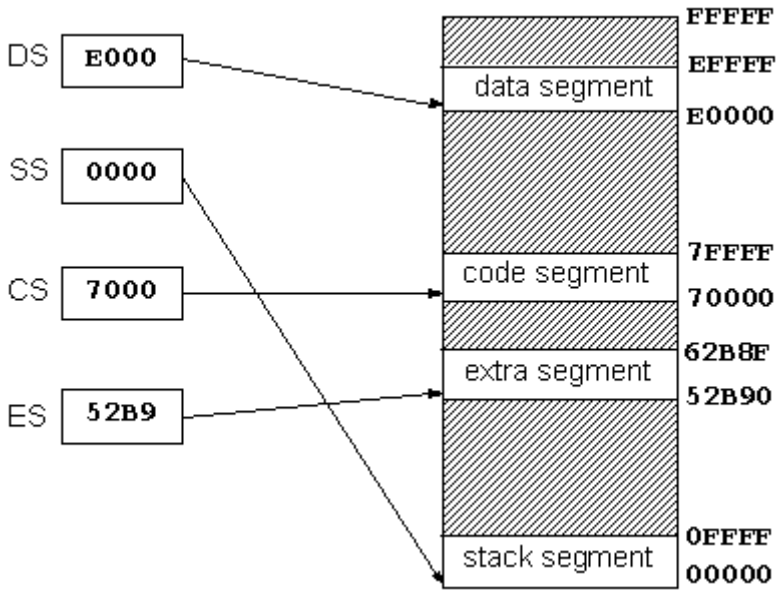
No caso do 8086, cuja word é de 16 bits, seria possível endereçar (aceder) a apenas 64KB de memória, o que é manifestamente insuficiente, mesmo em 1979.

Funcionamento

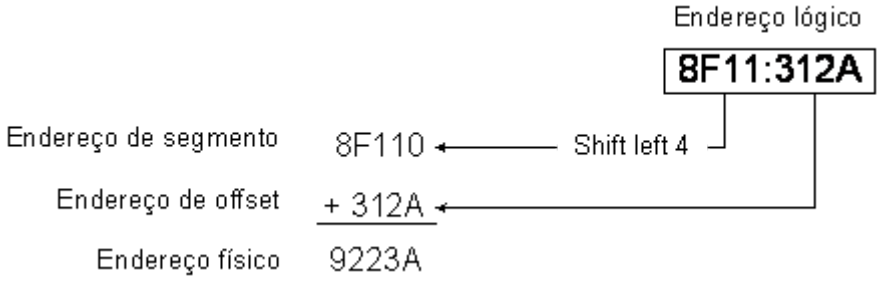
Aumentar o tamanho da word é complexo, quer para a engenharia relacionada com o desenvolvimento do processador quer para a manutenção da compatibilidade com o software desenvolvido para processadores com menor word.

Por outro lado, aumentar ao barramento de endereços não é tão complexo, e a sua evolução não é tão afectada por razões de compatibilidade com processadores com menor largura de barramento.

Surge o conceito de segmentação de memória, partindo do princípio de que a memória usada por um programa é de diferentes tipos e deve ou pode ser físicamente separada.



O endereço de uma posição de memória é calculado pela “combinação” de endereço do segmento e da deslocação (offset) dentro do segmento, segundo o esquema seguinte:



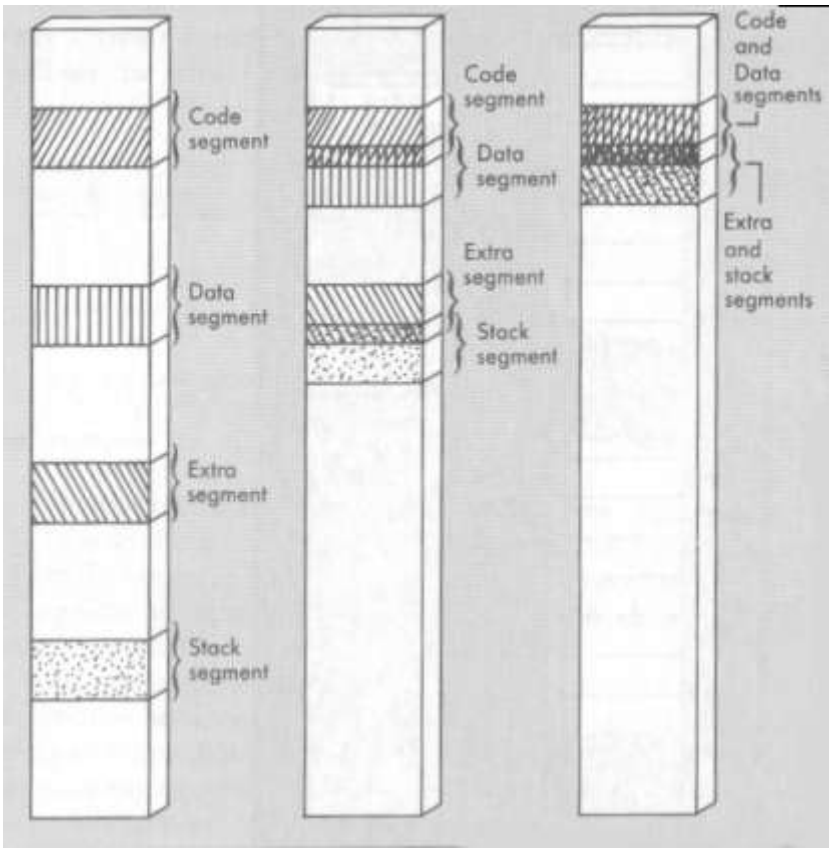
Portanto, como:

- Existem 4 registos de segmentos:
  - CS - Code Segment Register
  - DS - Data Segment Register
  - SS - Stack Segment Register
  - ES - Extra Segment Register
- Cada segmento inicia-se num endereço múltiplo de 16 (shift left do registo de segmento);
- Cada segmento prolonga-se por 2^{16} posições de memória (valores de offset);
- Um segmento tem portanto um tamanho fixo de 64 KB;
- Num determinado contexto (valores dos registos de segmentos), apenas 256 KB (4*64KB) são acessíveis simultaneamente.

Adicionalmente, existe mais do que um endereço lógico para o mesmo endereço físico, i.e. os segmentos podem sobrepor-se parcial ou completamente.

Os seguintes endereços lógicos correspondem todos à mesma posição de memória:

- 0000:0032
- 0001:0016
- 0002:0000



Modos de endereçamento

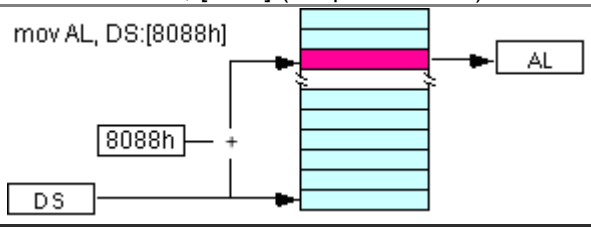
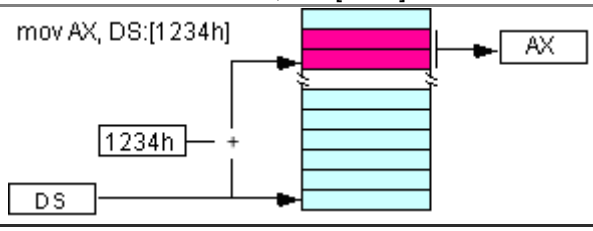
Considere-se a instrução MOV, que copia o valor do local indicado por “source” para o local indicado por “destination”:

mov destination, source

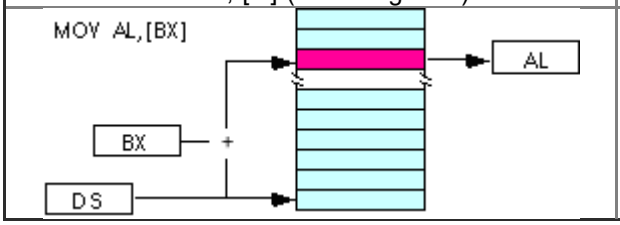
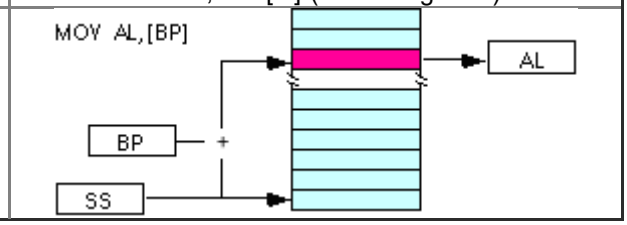
Existem cinco tipos distintos de endereçamento de memória.

Através de variável (Displacement Only)

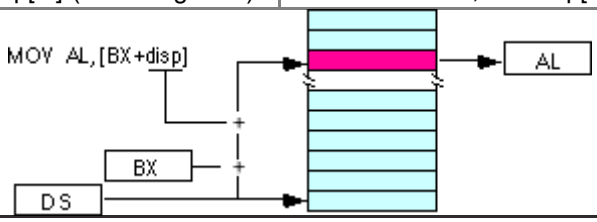
Acede ao valor da posição de memória representado pelo nome da variável (deslocamento). Na prática, a variável é representada no código pelo deslocamento dentro do segmento respectivo.

8 bits	2 bytes (word)
mov al, var1 (DS por omissão)	mov ax, var2
mov al, DS:var1 (DS explícito)	mov ax, DS:var2
mov al, [8088] (DS por omissão)	mov ax, DS:[1234]
	

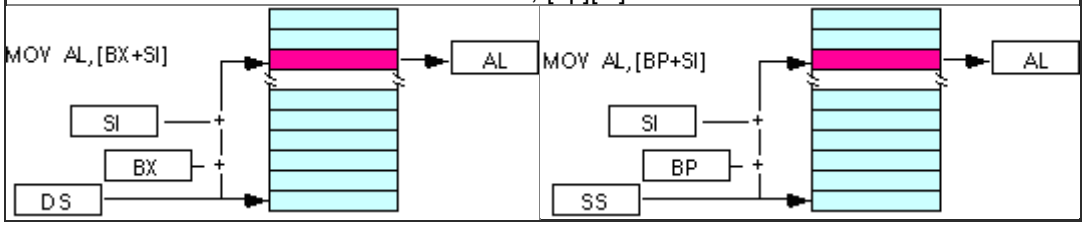
Através de valor de registo (Register Indirect)

Segmento usado por defeito	Definição do segmento a usar
mov al, [bx] (Data Segment)	mov al, CS:[bx] (Code Segment)
mov al [bp] (Stack Segment)	mov al, DS:[bp] (Data Segment)
mov al, [si] (Data Segment)	mov al, SS:[si] (Stack Segment)
mov al, [di] (Data Segment)	mov al, ES:[di] (Extra Segment)
	

Indexado (Indexed Addressing)

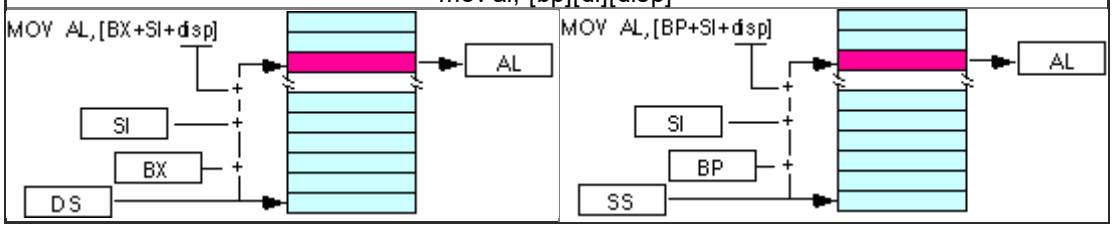
Segmento usado por defeit	Definição do segmento a usar
mov al, disp[bx] (Data Segment)	mov al, SS:disp[bx]
mov al, disp[bp] (Stack Segment)	mov al, ES:disp[bp]
mov al, disp[si] (Data Segment)	mov al, CS:disp[si]
mov al, disp[di] (Data Segment)	mov al, SS:disp[di]
	

Indexado mais base (Based Indexed)

Segmento usado por defeiro
mov al, [bx][si]
mov al, [bx][di]
mov al, [bp][si]
mov al, [bp][di]


Indexado mais base mais variável (Based Indexed Plus Displacement)

Segmento usado por defeiro
mov al, disp[bx][si]
mov al, disp[bx+di]
mov al, [bp+si+disp]
mov al, [bp][di][disp]



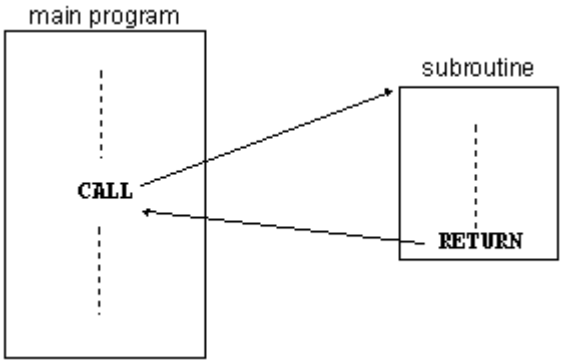
Regra

Assim, pela combinação dos valores de cada uma das seguintes colunas é possível definir um endereço válido: pelo menos o valor de uma coluna deve ser usado, e no máximo um valor por coluna.

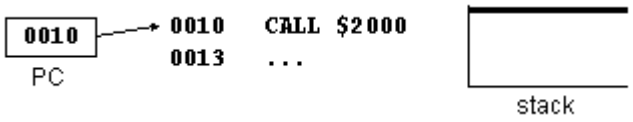
Deslocamento (variável)	[bx]	[si]
	[bp]	[di]

Sub-rotina

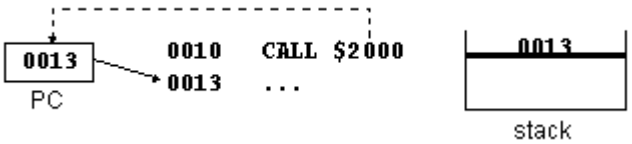
- CALL
- RET



1. CALL



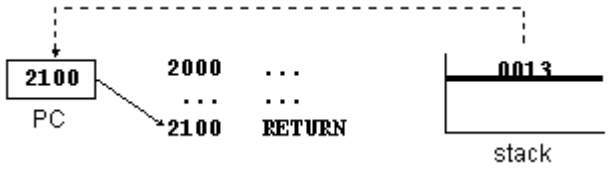
2. Processo preparatório (1)



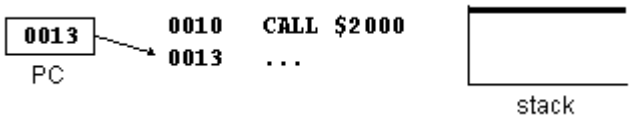
3. Processo preparatório (2)

4. Execução da sub-rotina

5. RET



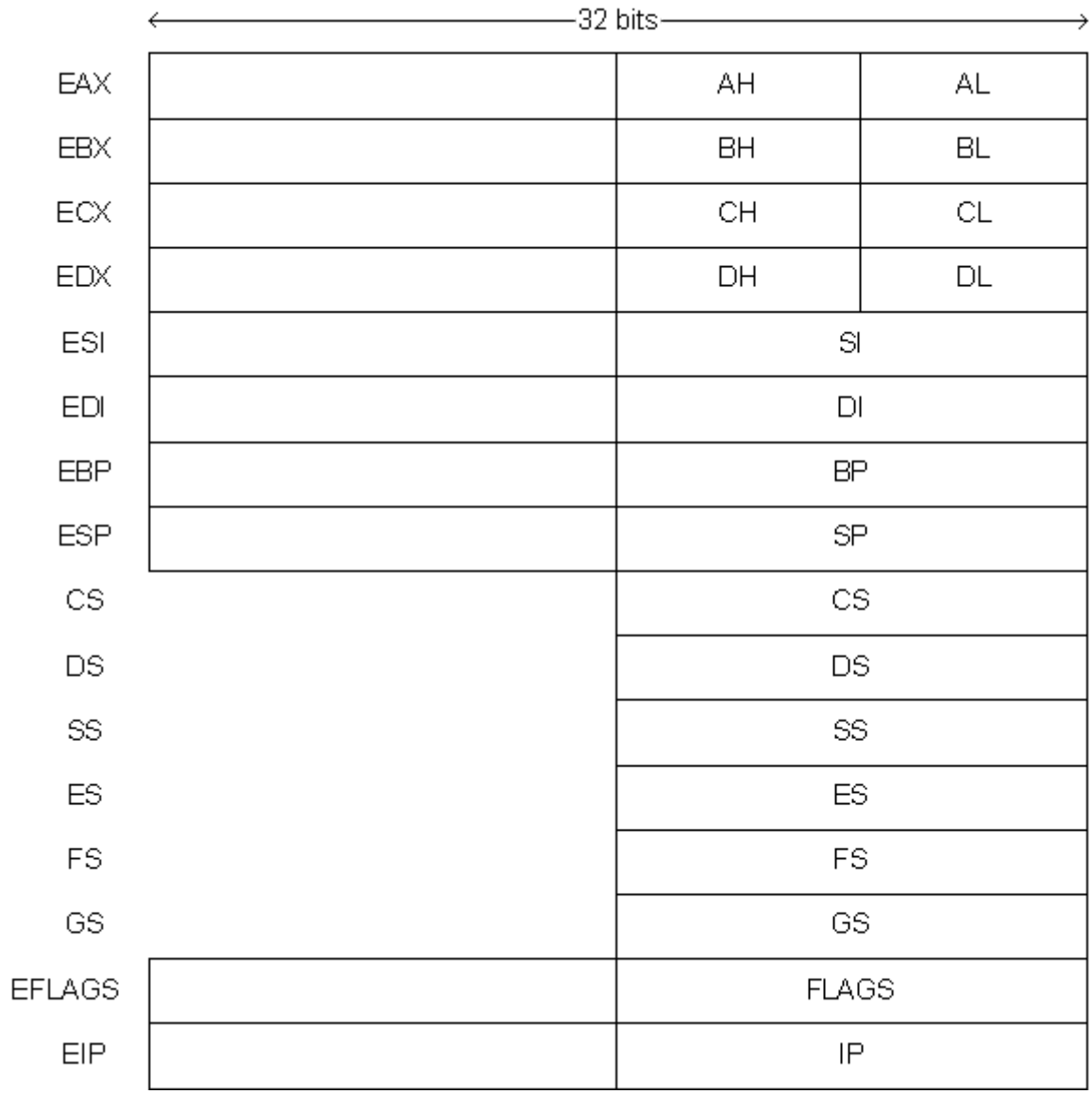
6. Continuação da execução principal



Intel 80386

Registos

- Tamanho: 32 bits (word de 32 bits);
- Barramento de endereços: 32 bits;
- Barramento de dados: 32 bits;
- Registos de segmentos de 16 bits;
- Registo de flags de 32 bits, sendo incluídas diversas novas flags.



Por razões de compatibilidade com os anteriores processadores da família, existe a possibilidade de aceder independentemente aos 16 bits menos significativos (ex. AX), bem como dentro destes, aos 8 bits mais e menos significativos (ex. AH, AL) se já existisse essa possibilidade nos anteriores processadores da família.

Segmentação

Ao contrário do que sucede no processador 8086, o tamanho da word não é (normalmente) limitativa da quantidade de memória endereçável:

$$2^{32} = 4GB$$

Por questões de compatibilidade e de facilidade de modelação do problema, a segmentação continua a fazer sentido no 80386.

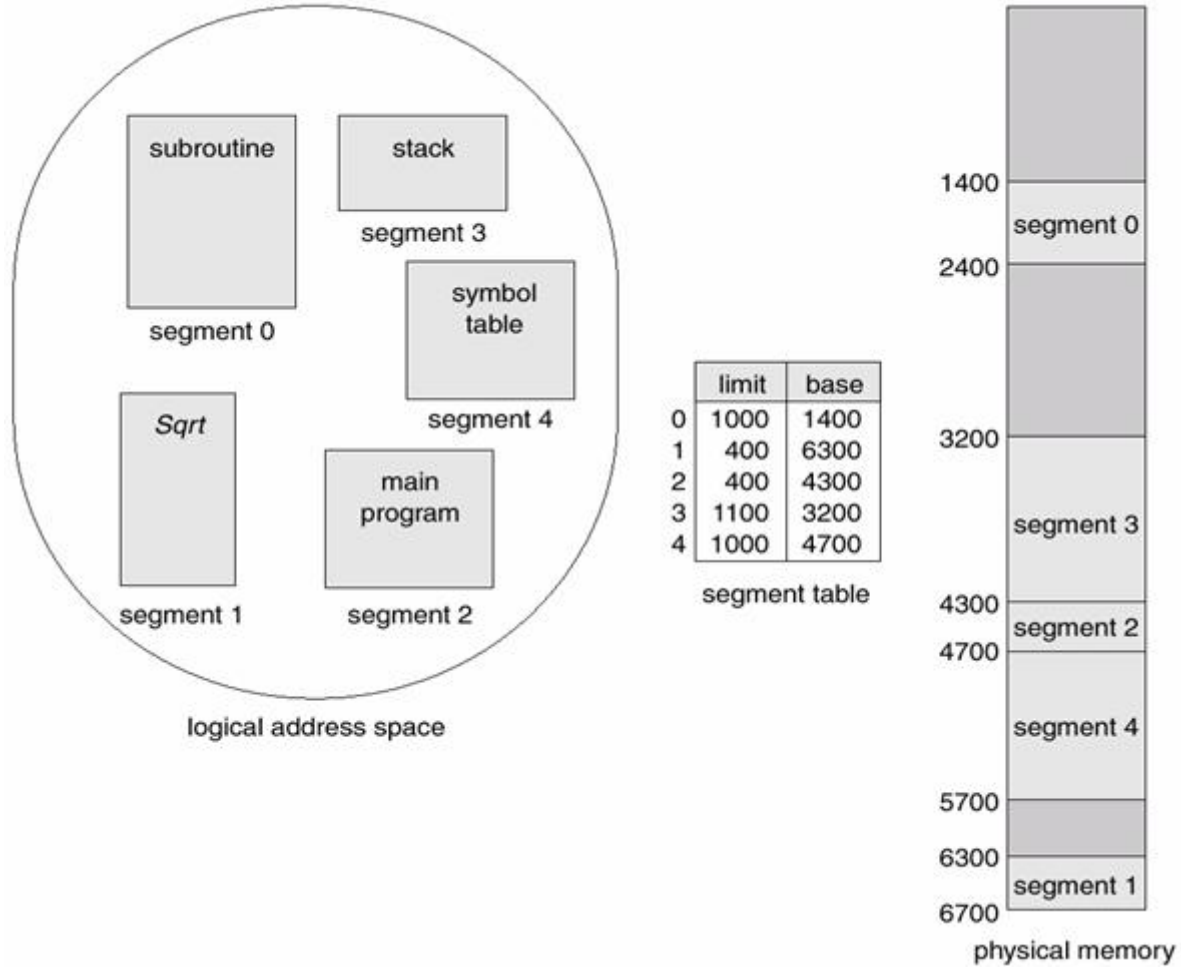
No entanto, a segmentação, tal como proposta no 8086 é extremamente limitativa em termos de tamanho dos segmentos.

A esse respeito deram-se duas alterações:

- Aparecimento de dois novos registos de segmento: FS e GS o que permite aceder a mais dois segmentos sem alterar os registos de segmentos;
- Alteração do funcionamento da segmentação pela inclusão de uma tabela de segmentos.

Esta tabela é responsável por guardar descritores de segmentos, que é a informação acerca do segmentos existentes, o que permite definir entre outras:

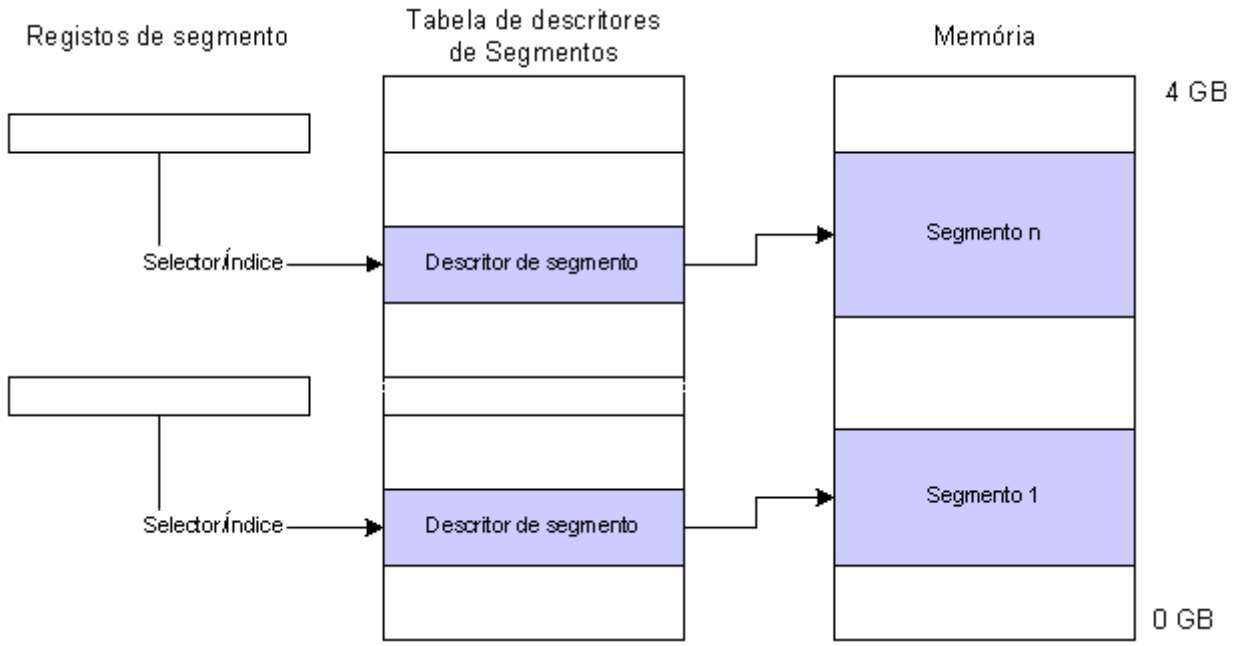
- Endereço de início do segmento;
- Tamanho do segmento;
- Tipo de segmento: dados, código, stack.



No entanto, a segmentação é mais do que a capacidade de segmentação de maiores blocos de memória. Nomeadamente, a segmentação fornece funcionalidades fundamentais de controlo de acesso de acordo com o nível de privilégio definido para o segmento.

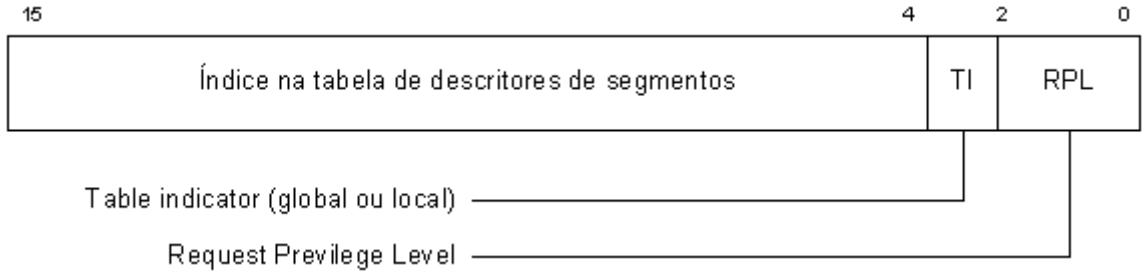
Registos de segmentos

Para isso, o valor dos registos de segmentos deixam de corresponder a um endereço físico na memória, e passam a corresponder a um índice numa tabela de informação sobre os segmentos existentes. O Valor do registo de segmento determina portanto a posição da tabela onde o segmento está descrito, sendo a sua localização determinada pela informação na tabela.



No entanto, o conteúdo dos registos de segmentos não é todo usado como valor para índice na tabela. De facto, os valores dos registos têm a seguinte semântica:

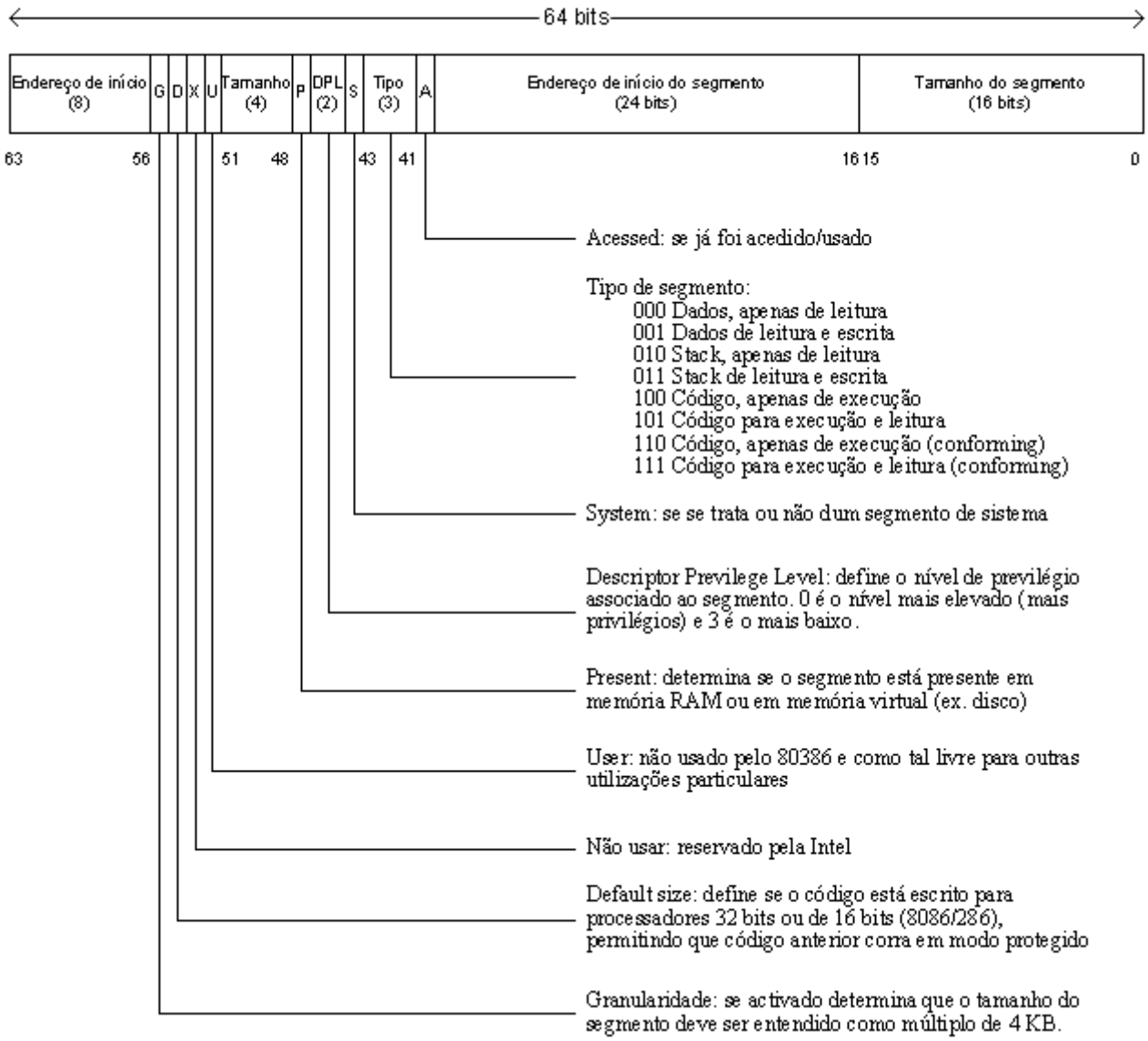
- 0-2 bits corresponde ao nível de privilégio com que é pedido acesso ao segmento;
- 3 bit define se o segmento deve ser procurado na tabela global de segmentos ou na tabela local;
- 4-15 bits corresponde ao índice na tabela de descritores de segmento.



Como o índice é representado por 12 bits, o número de segmentos acedidos por uma determinada tarefa é de 2^{12} = 8192.

4.2.2 Descritores de segmentos

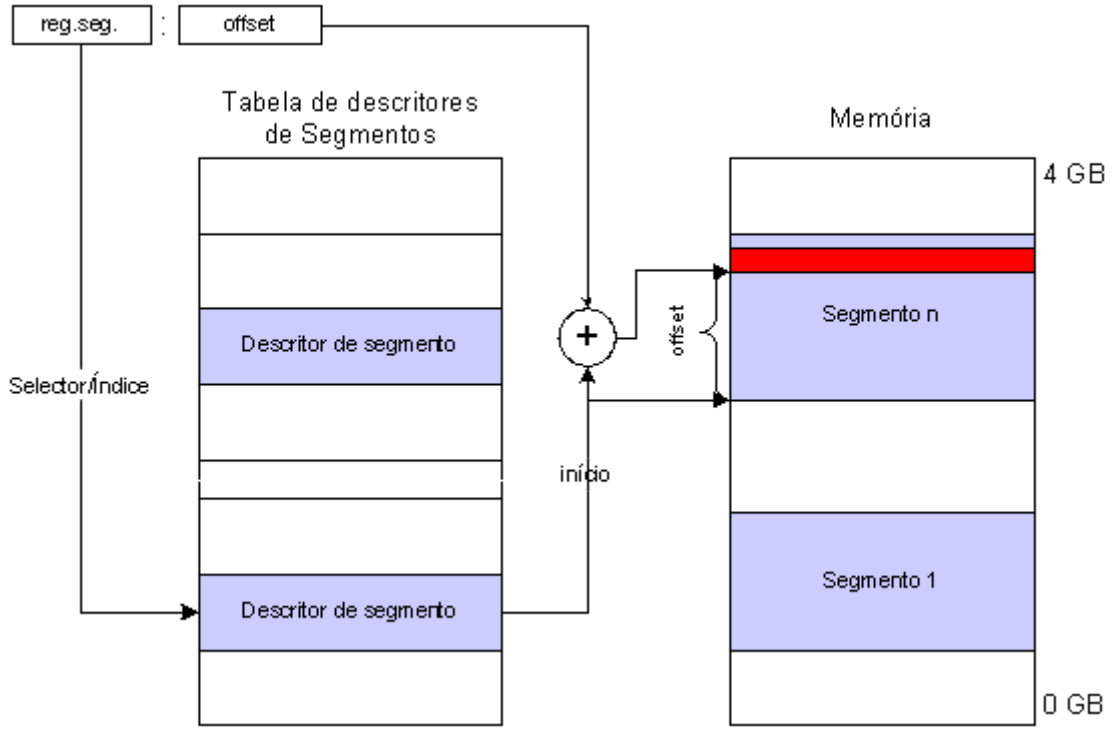
Cada descritor de segmento é uma estrutura com 64 bits de comprimento, dividida segundo os seguintes elementos:



4.2.3 Offset (Deslocamento)

Para determinar dentro dum segmento qual a posição de memória que se deseja aceder é necessário combinar o valor do endereço de início do segmento com o valor de endereço disponibilizado como deslocamento.

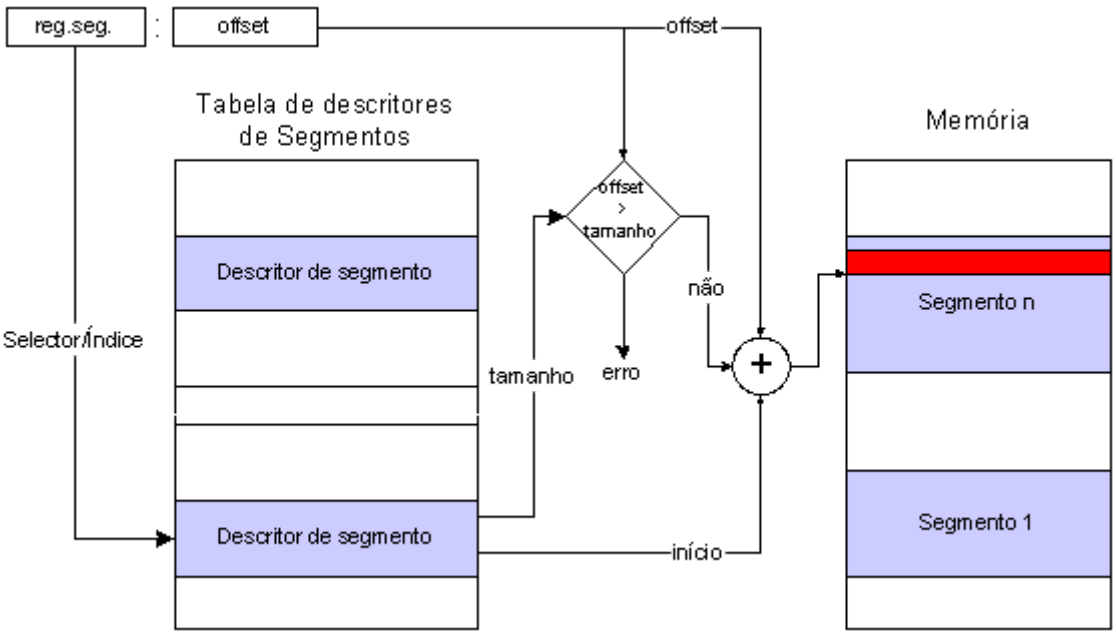
Ao contrário do processo de combinação de endereços no 8086, no caso do 80386 a combinação é uma simples adição de valores.



Como o valor de offset é um valor de 32 bits, é possível existir um único segmento no sistema e mesmo assim ser todo endereçável, pois o valor de offset permite aceder a qualquer endereço de memória.

No entanto, quando o segmento é menor que os 4 GB máximos (uma situação típica), o valor de offset poderá corresponder a uma posição de memória para além do limite máximo do segmento. Esta é uma situação altamente indesejável em ambientes multi-tarefa e em que diferentes níveis de privilégio sejam usados entre tarefas, como é o caso dos modernos sistemas operativos (ex. Linux, Windows).

No entanto, pela combinação do valor de endereço de início do segmento com o valor do tamanho do segmento, é possível determinar (validar) se determinado acesso ao segmento é realizado fora dos seus limites, e assim proteger o sistema.



4.2.4 Registos para tabelas de segmentos

Os descritores de tabelas têm registos associados para que possam ser localizadas na memória:

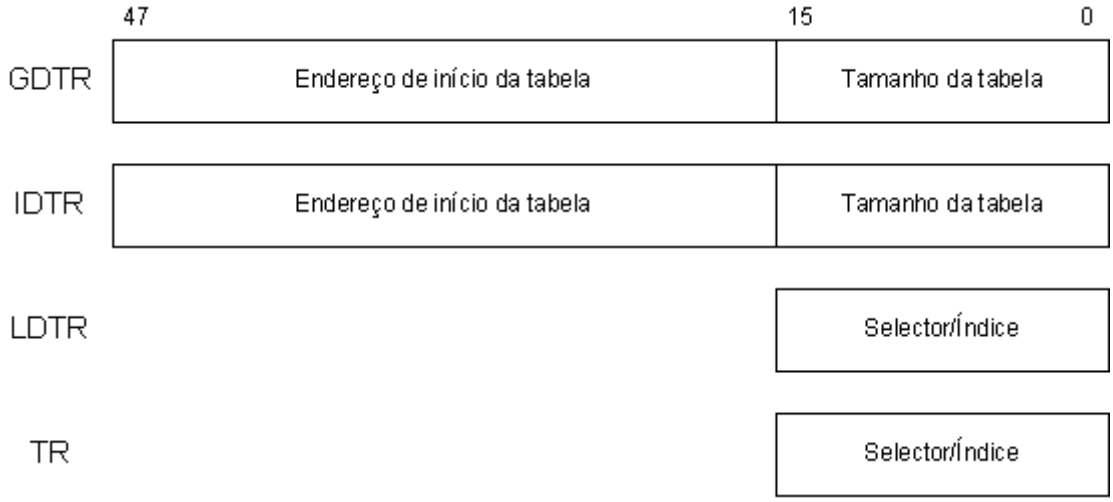
- GDTR: Global Descriptor Table Register, é o registo que contém o endereço de início e o tamanho da tabela de descritores de segmentos;

- IDTR: Interrupt Descriptor Table Register, é o registo que contém o endereço de início e o tamanho da tabela de descritores de rotinas de interrupções.

Trata-se de registos de 48 bits, usados da seguinte forma:

- 0-15 bits, contém o tamanho da tabela;
- 16-47 bits, contém o endereço linear de início da tabela.

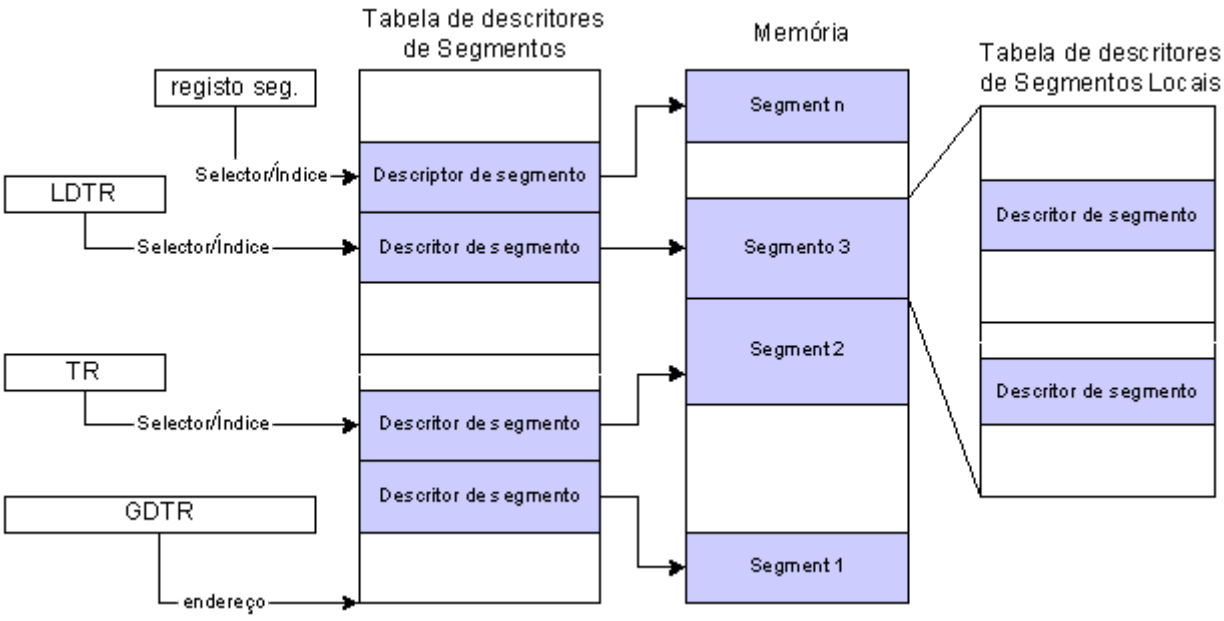
Ambos são inicializados assim que as tabelas de descritores são definidas, e não são alteradas durante toda a sessão.



Além destes dois registos visíveis globalmente por todas as tarefas, existem dois outros registos associados com as tarefas propriamente ditas:

- LDTR é o registo que determina o endereço da tabela de segmentos usados pela tarefa. Na realidade, o valor do registo é o valor correspondente a um índice na GDT, que por sua vez define o endereço da tabela de descritores dos segmentos associados com a tarefa. O bit 3 do registo de segmento determina se a tabela de segmentos a usar é a global ou local;
- TR é o registo que define índice na GDT referente ao TSS (Task-State Segment). Este segmento serve para armazenar informação referente ao estado dos registos do CPU para cada tarefa do sistema. Assim que determinada tarefa deixa de ser executada, os valores dos registos de CPU são copiados para o TSS da tarefa para que sejam acessíveis quando voltar a ser executada. O TR é então carregado com o descritor da "nova" tarefa, o que provoca o carregamento dos registos de CPU com os valores existentes no seu TSS (anteriormente armazenados).

Ambos são inicializados sempre que ocorre alteração da tarefa em execução.



4.2.5 Limitações da segmentação

- Segmentação pode causar/necessitar de fragmentação
- Acontece quando os blocos disponíveis são demasiado pequenos para conter o segmento inteiro;
- Potenciais soluções:
    - Compactação (ordenação de segmentos ou páginas);
    - Protelar execução do processo que necessita do segmento;
    - Fragmentar o segmento por vários blocos.

4.3Paginação

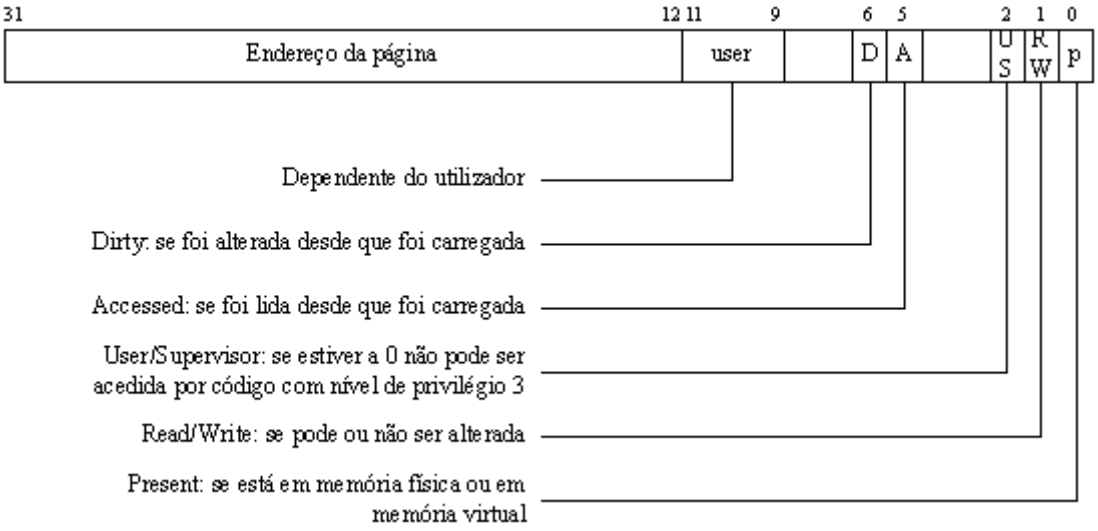
A paginação é um mecanismo de gestão de memória distinto da segmentação. Através da paginação é possível:

- Usar a memória em pedaços mais pequenos, denominadas páginas, permitindo um (potencialmente) melhor aproveitamento da memória. No Intel 80386 estas páginas são de 4KB;
- Implementar memória virtual, que corresponde a transferir de/para suportes de armazenamento secundários (ex. discos) pedaços de dados da “memória” desnecessários no momento, libertando memória RAM para dados mais relevantes.

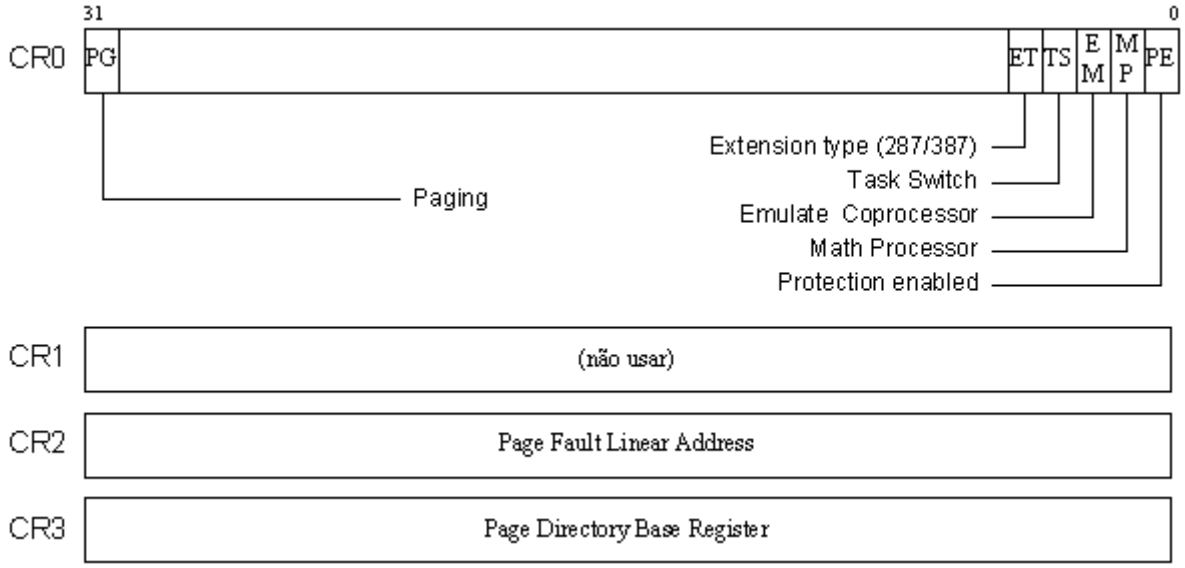
4.3.1 Tabelas de páginas

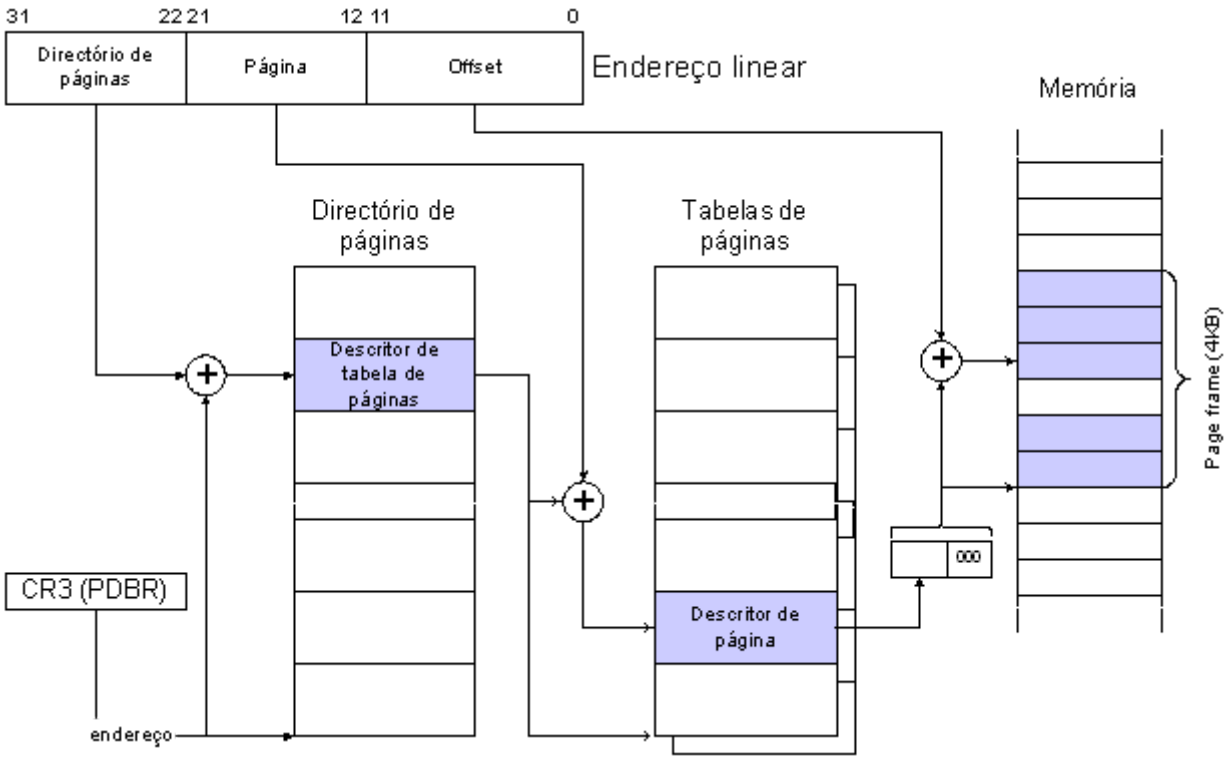
Tal como acontece com os segmentos, é necessário manter informação acerca das suas características. É portanto necessário definir tabelas de descrição de páginas.

Uma página é descrita através da informação descrita na seguinte estrutura de 32 bits:



Tal como para o segmentos, é necessário definir a localização da tabela de descritores páginas. Para isso foram introduzidos quatro novos registos de 32 bits:





4.4 Segmentação com Paginação

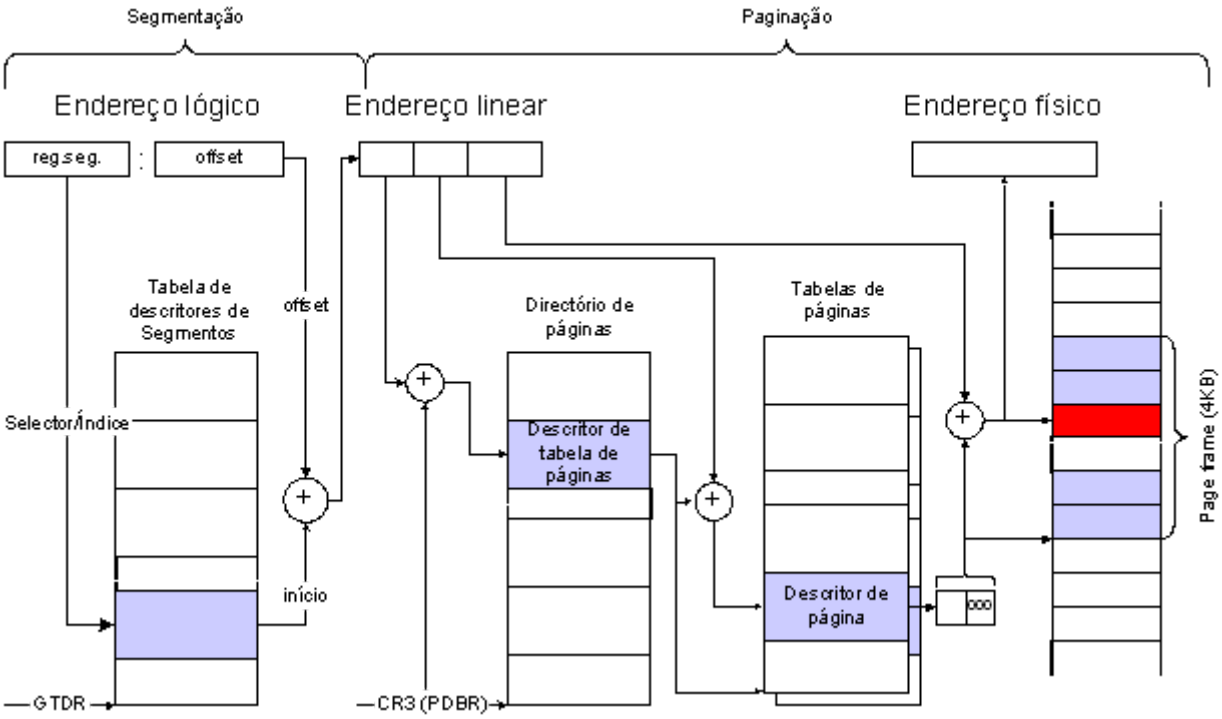
Através da conjugação da segmentação com a paginação é possível potenciar os benefícios e reduzir as desvantagens associadas a cada uma duas técnicas de organização de memória.

Vejamos:

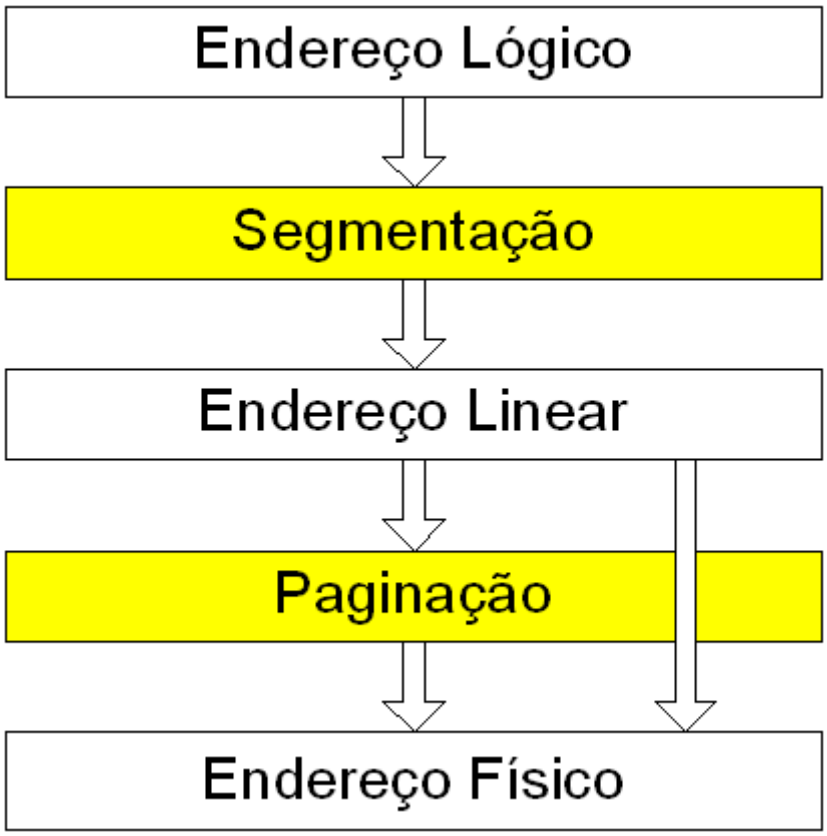
	Segmentação	**Paginação**
Início	Qualquer	De 4KB em 4KB
Tamanho	Qualquer	4KB
Níveis de Privilégio	4	2
Protecção contra escrita	Sim	Sim
Informação sobre presença	Sim	Sim
Informação sobre acesso	Sim	Sim
Tipo de conteúdo	Sim	Não
Informação sobre alteração	Não	Sim

A combinação da segmentação com a paginação resulta num processo transparente para cada uma das fases. Isto é, para funcionar não há necessidade de alterações a cada um dos processos.

Corresponde ao seguinte esquema:



No entanto, enquanto a segmentação não pode ser desligada, a paginação pode, pelo que quando isso acontece, o endereço linear resultante da segmentação corresponde ao endereço físico. Ou seja:



Última actualização: 28-04-2006