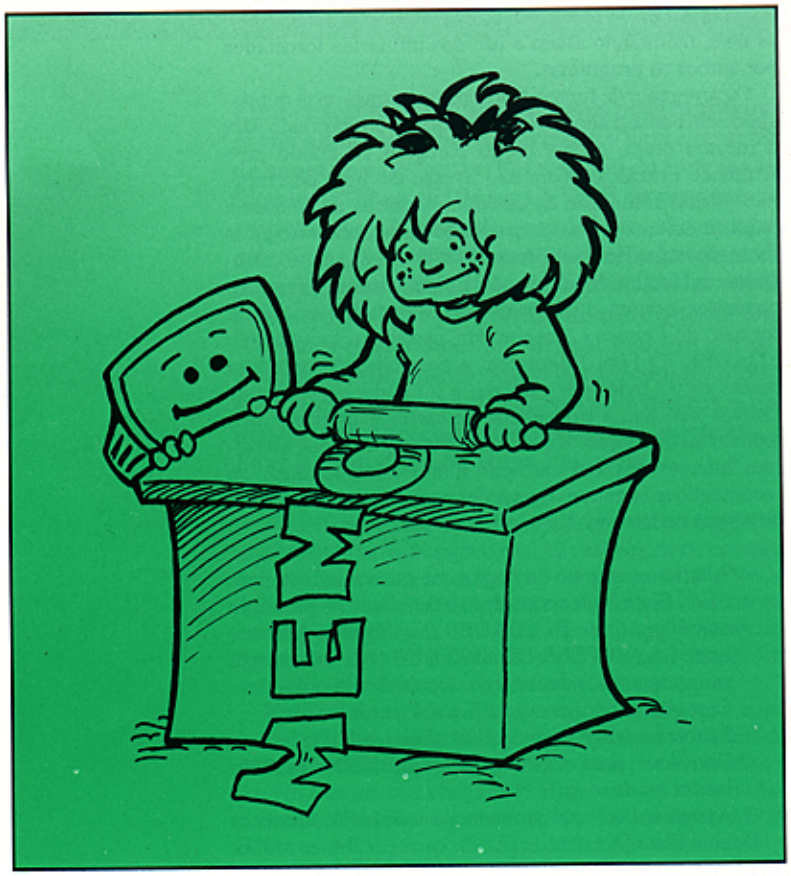
# MEMÓRIA ESTENDIDA

## PARTE III



Neste artigo vamos continuar a delicadíssima missão de apresentar ao leitor os princípios básicos que regem o funcionamento dos processadores Intel de 32 bits em **Modo Protegido**.

Era nossa ideia, no início, que 2 artigos deveriam bastar para apresentar o tema. Porém, apercebemo-nos depressa que, mesmo a um nível introdutório, ficaria um enorme vácuo por preencher. Nesse sentido, optámos – bem ou mal – por aprofundar um pouco mais esta matéria com um terceiro artigo e que tratará do suporte interno dos processadores Intel de 32 bits a actividades de Multitarefa (*Multitasking*). Depois disso, e para completar o quadro panorâmico que nos propusemos, abordaremos um a um os vários protocolos e *standards* existentes na actualidade para alocação de memória *extended* e/ou execução de programas nessa mesma memória.

O programa para este número versa dois assuntos cujo conhecimento é muito relevante para o entendimento do funcionamento dos processadores Intel de 32 bits em **Modo Protegido**.

O primeiro deles aborda os *Interrupts* (ou Interrupções) e *Exceptions* (ou Excepções) e o segundo trata de *Gates* (ou Portas). O leitor interessado em obter um conhecimento substancialmente mais aprofundado desta matéria deverá também analisar o código fonte do programa ilustrativo que incluímos na disqueta **Spooler** e que dá pelo nome de P32INTR. Naturalmente que serão necessários conhecimentos de linguagem *Assembly*.

### INTERRUPTS E EXCEPTIONS

Todos os programas em fase de execução são frequentemente «interrompidos» para atenderem a sinais de *Hardware* que se designam por Interrupções). Uma Interrupção faz derivar temporáriamente o fluxo de execução do programa em curso para uma ou mais (caso estejam várias encadeadas) rotinas, normalmente exteriores a esse programa e que são designadas, entre outros nomes, por *Handlers* do *Interrupt*.

No final da sua execução, o último *Handler* da cadeia devolverá o controlo ao programa interrompido, o qual pode então prosseguir a partir do ponto de interrupção como se nada se tivesse passado.

As excepções, por sua vez, têm um comportamento algo semelhante às Interrupções, mas não são originadas por sinais de *Hardware*, mas sim por instruções do nosso programa.

As Interrupções são despoletados por acontecimentos exteriores ao programa; as Excepções são respostas do processador a certas condições detectadas durante a execução de uma instrução do nosso programa. Uma Excepção ocorre sempre no mesmo ponto de um programa; um Interrupção é assíncrona com a execução de um dado programa.

Algumas Excepções, tal como o conhecido INT 21H que rege a maioria dos serviços do DOS, e de um modo geral todas as outras Excepções que no seio dos nossos programas se iniciam com a mnemónica *Assembly* INT, são mais vulgarmente designadas por *Software Interrupts* (ou Interrupções por *Software*). Para além das Interrupções por *Software*, existem Excepções cuja origem é uma reacção

interna do microprocessador a erros detectados durante a execução de instruções do nosso programa. Estas Excepções do microprocessador são classificadas como se segue:

***Faults***: São as que acontecem antes da execução da instrução de código máquina que lhes dá origem. O endereço de retorno do *Handler* apontará, por conseguinte, para a instrução que originou a *Fault* e não para a instrução que se lhe segue. Isto permite que a instrução possa ser repetida de novo (em princípio em novas condições que já não produzam erro).

***Traps***: Acontecem após a execução da instrução que lhes dá origem.

***Aborts***: Este tipo de Excepções muitas vezes não possibilita a localização da instrução que as originou, pois acontece sempre após grandes «catástrofes» tais como falhas no *Hardware* ou valores disparatados em tabelas do sistema tais como a *Global Descriptor Table*.

Nada mais nada menos do que 32 Excepções do microprocessador estão previstas pela Intel, a qual reservou para o seu tratamento os vectores da tabela de interrupções com números compreendidos entre 00H e 1FH. A lista dessas Excepções, a definição dada pela Intel e o respectivo tipo constam da tabela seguinte.

| Número do Vector | Definição/Tipo |
|---|---|
| 0 | Divide Error (*FAULT*) |
| 1 | Debug Exception (*TRAP ou FAULT*) |
| 2 | NMI (Não é *Exception* mas *Interrupt*) |
| 3 | Breakpoint (*TRAP*) |
| 4 | Overflow (*TRAP*) |
| 5 | Bounds (*FAULT*) |
| 6 | Invalid Opcode (*FAULT*) |
| 7 | Coprocessor Not Available (*FAULT*) |
| 8 | Double Fault (*ABORT*) |
| 9 | Coprocessor Segment Overrun (*ABORT*) |
| 10 | Invalid TSS (*FAULT*) |
| 11 | Segment Not Present (*FAULT*) |
| 12 | Stack Fault (*FAULT*) |
| 13 | General Protection (*FAULT* ou *TRAP*) |
| 14 | Page Fault (*FAULT*) |
| 15 | Reservado pela INTEL para uso futuro. |
| 16 | Coprocessor Error (*FAULT*) |
| 17 a 31 | Reservado pela INTEL para uso futuro. |

Claro que uma coisa é a Intel ter decidido reservar para si os vectores da tabela de interrupções compreendidas entre 00H e 1Fh, e outra é os outros intervenientes no processo – e que têm voto na matéria – aceitarem e cumprirem esse desejo. E tanto assim é, que já nos idos tempos do IBM XT, a poderosa IBM tomara a liberdade de se apropriar para seu próprio uso (e abuso) de alguns dos vectores reservados pela Intel. Isso não constituiu um problema sério nos longínquos tempos do IBM XT, pois o velho processador 8088 apenas tinha capacidade para compreender as Excepções de 00H a 04H e, portanto, se o BIOS ou o DOS utilizassem vectores de interrupção entre 05H e 1FH, o 8088 não ficava mais pobre por isso.

Contudo, quando o IBM AT apareceu, e com ele o «revolucionário» processador Intel 80286, as «coisas» começaram a ficar subitamente muito mais complicadas. Por exemplo, a Interrupção 5 estava a ser utilizada pelo BIOS para a função de fazer sair na impressora o conteúdo do ecrã aquando da digitação da tecla «Print Screen» no teclado. Mas esse mesmo vector 5 já disponha primáriamente de outra missão (que podia ser utilizada pelo *chip* 80286), e que era a de detectar durante a execução da instrução de código máquina correspondente à mnemónica *Assembly* BOUND, se o operando ultrapassava os limites de um dado bloco de memória.

E não só – muito mais grave que tudo isto – o primeiro bloco de Interrupções de *Hardware* (IRQ0 a IRQ7) apropriava-se muito «desrespeitosamente», durante a fase de inicialização do sistema dos vectores de interrupção compreendidos entre 8 e 15, os quais são, como se sabe, parte integrante da «coutada» reservada da Intel. Uma verdadeira calamidade, pensarão alguns leitores que estão a ficar inquietos com toda esta anarquia.

Bem, moderemo-nos um pouco. Efectivamente, o mal existe mesmo e é muito concreto, mas os potenciais malefícios só se fazem normalmente sentir quando o processador trabalha em **Modo Protegido**.

E tanto é assim que uma das primeiras preocupações que devemos ter quando construímos um programa para trabalhar (autonomamente) em **Modo Protegido** é dar a «César o que é de César», isto é, restituir à Intel todos os seus vectores da tabela de interrupções. Isso implica, de um modo geral (existe outra solução mas é menos recomendável e inibimo-nos de a referenciar), reimplantar todo o primeiro nível de Interrupções de *Hardware* para outra zona da tabela de interrupções.

E a partir de agora, quando algum «sabichão» afirmar soberbamente ao caro leitor que o INT 8 é uma Interrupção que é emitida 18,2 vezes por segundo com a finalidade de actualizar o Relógio do Sistema, ou que o INT 9 é uma Interrupção que é emitida pelo *Hardware* em resposta à digitação do teclado, o leitor pode replicar, mais soberbamente ainda, que em **Modo Protegido** isso normalmente não corresponde à verdade (e em **Modo Real** também pode não corresponder, embora normalmente isso não aconteça).

E uma demonstração da possibilidade de reprogramação do PIC (*Programmable Interruption Controller*) de modo a distribuir os vectores da tabela de interrupções com valores compreendidos entre 20H e 27H ao primeiro nível de Interrupções de *Hardware* é feita pelo programa ilustrativo P32INTR.EXE (que deverá ser analizado pelos leitores interessados na compreensão do «modus faciendi» desta e de algumas outras técnicas pouco conhecidas.).

Outro aspecto interessante e também pouco conhecido tem a ver com a localização física em memória da tabela de interrupções (**IDT** ou *Interrupt Descriptor Table*). Normalmente, em **Modo Real**, a tabela é localizada nos primeiros 1000 bytes da memória física, ou seja, nos endereços físicos que vão de 0H a 400H (ocupando cada um dos 256 vectores de interrupção 4 bytes de espaço nessa tabela).

Nos processadores Intel de 32 bits a **IDT** pode estar localizada em qualquer ponto da memória física. O tamanho da tabela também pode variar. Quer a informação sobre a localização física em memória do início da **IDT**, quer sobre o respectivo tamanho, constam de um registo do processador designado por **IDTR** (*Interruption Descriptor Table Register*).

Em **Modo Protegido** cada vector de interrupção tem uma entrada de 8 bytes (que corresponde ao tamanho de um *Descriptor*) na **IDT** (em **Modo Real** essa entrada é, como se sabe, de apenas 4 bytes). Por conseguinte, para formar um índice na **IDT** o processador multiplica o número da

Interrupção ou da Excepção por 8. A **IDT** contém um dos seguintes 3 tipos de *Descriptors*:

- *Interruption Gates*
- *Trap Gates*
- *Task Gates*

Vamos então ver o que são essas famosas *Gates*.

## GATES

As *Gates* (ou Portas) são utilizadas (e não só na **IDT**) com a finalidade de transferir controlo entre segmentos executáveis, eventualmente com diferentes níveis de previlégio. Todas as Portas têm *Descriptors* quer na **GDT** (*Global Descriptor Table*) quer numa das **LDT** (*Local Descriptor Table*) quer na **IDT**.

Existe contudo um tipo de Porta, a *Call Gate* que não pode ter *Descriptors* na IDT. Os *Descriptors* das Portas pertencem à categoria de «*Descriptors* para Segmentos de Sistema e Portas» (é favor conferir com o explicado a este respeito na Spooler Nº 24) e recebem o valor 0 no campo **DT** (*Descriptor Type*).

Nos *Descriptors* para Segmentos de Sistema e Portas os bytes 40 a 43 definem um dos tipos constantes da tabela seguinte.

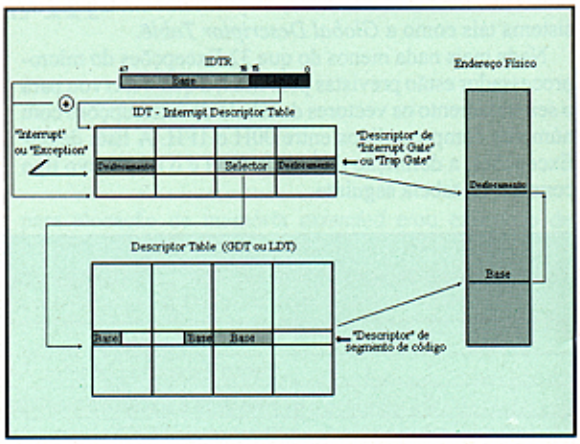
| TYPE | Designação |
|---|---|
| 0 | Reservado |
| 1 | 80286 TSS (*Task State Segment) disponível* |
| 2 | LDT *(Descriptor Table)* |
| 3 | 80286 TSS indisponível. |
| 4 | 80286 *Call Gate* |
| 5 | *Task Gate* |
| 6 | 80286 *Interrupt Gate* |
| 7 | 80286 *Task Gate* |
| 8 | Reservado |
| 9 | 80386/80486 TSS disponível. |
| 10 | Reservado |
| 11 | 80386/80486 TSS indisponível |
| 12 | 80386/80486 *Call Gate* |
| 13 | Reservado |
| 14 | 80386/80486 *Interrupt Gate* |
| 15 | 80386/80486 *Trap Gate* |

Para o caso das *Interruption Gates* e *Trap Gates* o *Descriptor* terá o aspecto da figura abaixo:

Os *Descriptors* das *Call Gates* e *Task Gates* são algo diferentes e serão abordados no próximo artigo, quando nos debruçarmos sobre o tema da Multitarefa.

Como se viu atrás, tanto as *Interruption Gates* como as *Trap Gates* são habitualmente activadas por Interrupções ou Excepções. Tanto umas como outras fazem referência indirecta a uma rotina que é localizada pelo microprocessador do seguinte modo:

- O campo da Porta designado por «Selector do Segmento» aponta para o *Descriptor* de um segmento executável quer na **GDT** quer na **LDT**.
- Os campos da Porta designados por «Deslocamento no Segmento» apontam para o início do *Handler* da Interrupção ou Excepção. Vejamos em esquema como isso se passa:




A única diferença entre uma *Interruption Gate* e o uma *Trap Gate* reside na sua actuação em relação à *Interruption Flag* do registo *FLAGS* do microprocessador. Um Interrupção ou Excepção que utilize uma *Interruption Gate* coloca a zero a *Interruption Flag* impedindo que outras Interrupções ou Excepções interrompam a execução do *Handler*, enquanto que se for utilizado uma *Trap Gate* o *Handler* pode ser interrompido.

E ficamos por aqui, pois o objectivo que nos propusemos para esta etapa foi atingido. No próximo número dê-nos um pouco do seu tempo, pois vamos tentar explicar-lhe o potencial que os processadores Intel de 32 bits incorporam a nível de *Hardware*, possibilitando o desenvolvimento de verdadeiros sistemas operativos de Multitarefa.

**José Páscoa**