Registros

Registros visíveis

Registros Gerais

Registros de 32-bit:

* EAX => acumula resultado de operações;
* EDX => segura dados da memória;
* ECX => contador utilizado em loops;
* EBX => base para endereçamentos;

Registros de 16-bit:

* AX => word menos significante de EAX;
* DX => word menos significante de EDX;
* CX => word menos significante de ECX;
* BX => word menos significante de EBX;

Registros de 8-bit:

* AH => byte mais significante de AX;
* AL => byte menos significante de AX;
* DH => byte mais significante de DX;
* DL => byte menos signficante de DX;
* CH => byte mais significante de CX;
* CL => byte menos significante de CX;
* BH => byte mais significante de BX;
* BL => byte menos significante de BX;

Registros de segmentos (16-bit):

* CS => índice seletor para segmento de código;
* SS => índice seletor para segmento de pilha;
* DS => índice seletor para segmento de dados;
* ES => índice seletor extra;
* FS => índice seletor extra;
* GS => índice seletor extra;

Registros de estado e controle (32-bit):

* EFLAGS => armazena estado de programa;
* EIP => armazena posição atual no programa;

Registros invisíveis

Registros de gerenciamentos de Memória

* GDTR => Descritor de segmento, indica lista de segmentos globais;
  + endereço base => 32-bit;
  + limite => 16-bit;
* LDTR => Descritor de segmento, indica lista de segmentos locais;
  + endereço base => 32-bit;
  + limite => 16-bit;
* TR => Descritor de segmento, indica lista de tarefas;
  + seletor => 16-bit;
  + endereço base => 32-bit;
  + limite => 16-bit;
  + atributos => 16-bit;
* IDTR => Descritor de segmento, indica lista de interrupções;
  + seletor => 16-bit;
  + endereço base => 32-bit;
  + limite => 16-bit;
  + atributos => 16-bit;

Registros de controle (32-bit)

* CR0 => Contém flags que controlam certas funções da cpu;
* CR1 => [Reservado];
* CR2 => Endereço linear, da falha de acesso a página;
* CR3 => Página de diretório, descritor;

Registros de Teste e Debug (32-bit)

* DR0 => Breakpoint 0, endereço linear;
* DR1 => Breakpoint 1, endereço linear;
* DR2 => Breakpoint 2, endereço linear;
* DR3 => Breakpoint 3, endereço linear;
* DR4 => [reservado];
* DR5 => [reservado];
* DR6 => Flags de debug;
* DR7 => Flags de debug;
* TR3 => TLB teste, dado;
* TR4 => TLB teste, endereço linear;
* TR5 => TLB teste, não utilizado, descritor de segmento;
* TR6 => TLB teste, endereço linear;
* TR7 => TLB teste, endereço físico;

Flags do Sistema

EFLAGS

* CF, 0 => Carry Flag, é definido na transbordação de uma operação de soma ou subtração na qual o valor resultante não cabe no operador de 32, 16 ou 8-bit da instrução em específico;
* PF, 2 => Parity Flag, definido caso o resultado de uma operação resulte em um valor par, é limpo do contrário;
* AF, 4 => Auxiliary Flag, {analise};
* ZF, 6 => Zero Flag, é definido caso a operação resulte em zero;
* SF, 7 => Signed Flag, é definido caso a operação resulte em um valor negativo (o bit mais significante está definido);
* TF, 8 => Trap Flag, {analise};
* IF, 9 => INTR Flag, caso ativo, permite a cpu o reconhecimento de interrupções externas por INTR, do contrário, são ingnorados;
* DF, 10 => Direction Flag, caso definido, operações com string realizam decremento de registros ESI/EDI, do contrário, configura incremento dos mesmos;
* OF, 11 => Overflow Flag, é definido em operações que ocorram transbordamento do valor do resultado para o sinal;
* IOPL, 12-13 => Input/Output Privilege Level, define o CPL (current privilege level) do programa atual;
* NT, 14 => Nested Flag, definido caso o programa tenha realizado um pulo aninhado, isto é, com um valor de retorno;
* RF, 16 => Resume Flag, {analise};
* VM, 17 => Virtual Mode, {analise};
* AC, 18 => Alignment Check, caso ativo, permite a verificação de alinhamento de endereço no acesso á memória;
* VIF, 19 => Virtual Interrupt Flag, {analise};
* VIP, 20 => Virtual Interrupt Pending, {analise};
* ID, 21 => Identification Flag, {analise};

CR0

* PE, 0 => Protection Enable, ativa a proteção por segmento;
* MP, 1 => Math Present, {analise};
* EM, 2 => Emulation, {analise};
* TS, 3 => Task Switched, indica se houve uma troca de tarefa;
* ET, 4 => Extension Type, {analise};
* NE, 5 => Numeric Error, caso ativo, configura a possibilidade de receber erro reportado de operações numéricas de ponto flutuante;
* WP, 16 => Write Protect, {analise};
* AM, 18 => Alignmente Mask, utilizado em conjunto com AC [{analise}];
* NW, 29 => Not Write Through, {analise};
* CD, 30 => Cache Disable, {analise};

Segmentação

Um modo de gerenciamento de memória do sistema é pelo acesso segmentado dos blocos de dados, isso permite gerir a segurança e restringir o acesso de programas de nível alto á dados do sistema ou mesmo de outros programas.

Tradução

Um endereço fornecido pelo programa se chama “Logical Address”, isto é, endereço lógico, que precisa ser traduzido para “Linear Address”, isto é, endereço linear, para isso, ele precisa passar pela tradução da segmentação.

Para isso, ele acessa uma das tabelas de segmento, por meio dos registros GDTR ou LDTR, que respectivamente correspondem á tabela global de segmentos, e tabela local de segmentos, cada um armazena um descritor de segmento que será utilizado para acessar ambas as tabelas na memória.

Primeiramente, é fornecido o *Logical Address*, e a depender do tipo de dado que será acesso, um dos descritores serão utilizados (CS, SS ou DS), o descritor fornecerá um índice para a tabela (não considerando seus 3 primeiros bits), e fornecerá informação sobre qual das tabelas (GDTR e LDTR) será acesso, por meio do seu bit 2 (1 para local, 0 para global), a memória é acessada neste local fornecido, obtém dois *dwords* e converte para um descritor, que traduzirá o endereço fornecido pelo programa.