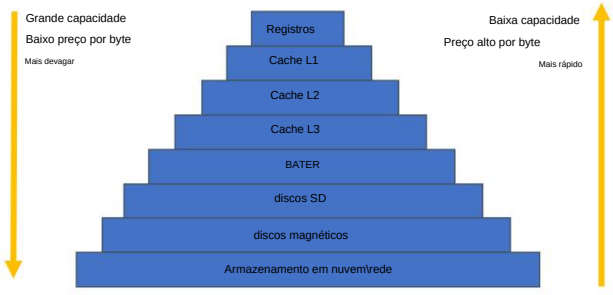
Registros
Instrução MOV e ADD

1

Registros

Os registradores são o nível superior da hierarquia de memória



2

Registros

O número de registradores em cada CPU é muito pequeno

Esse número tende a aumentar com o tempo

Além disso, o tamanho dos registradores tende a aumentar com o

tempo. **Cada registrador é identificado pelo seu nome, não por um endereço.**

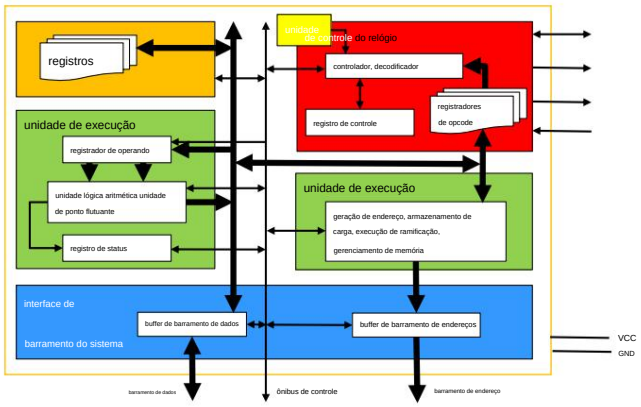
O nome não diferencia maiúsculas de minúsculas.

Alguns **registros diferentes** podem compartilhar o mesmo **espaço físico.**

Nesses casos, a alteração em um registrador implica em alteração no outro, pois compartilham o mesmo espaço físico **Cada nova UCP que**

implementa novos registradores deve manter o nome e o tamanho de todos os registradores anteriores a esta UCP para que seja compatível com o software anterior.

3

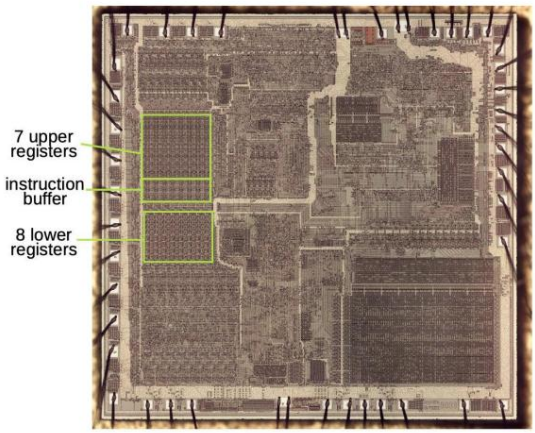


4

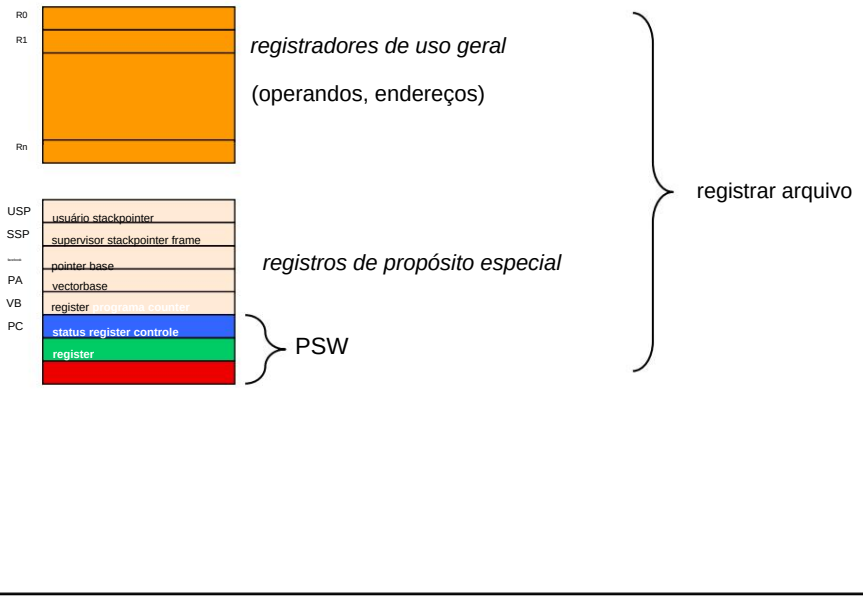
Registros

- Memória muito rápida com tempo de acesso muito baixo (< 1 ns)
- Seleção direta de registros únicos por meio de linhas de controle dedicadas
  - Nenhum decodificador/decodificação de endereço é necessário
- Todos os registradores estão no chip
  - Nenhum acesso externo necessário com atrasos devido ao tempo de execução, multiplexação, buffer, etc.
- Pode oferecer funções adicionais
  - Incremento/diminuição
  - Mudança
  - Definido como zero, conectado a zero
- Várias portas de entrada/saída independentes
  - Possibilidade de escrita e leitura simultânea de vários (diferentes) registradores
  - Os processadores superescalares de hoje são capazes de escrever 4 registradores e ler 8 registradores em um ciclo de clock

5

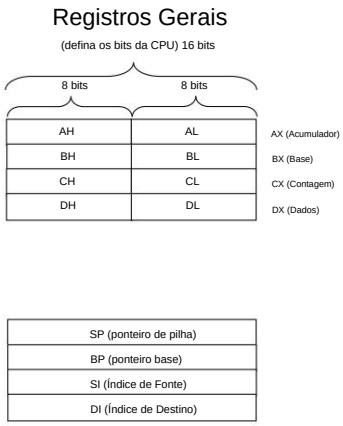


6



7

Registros 8086/286



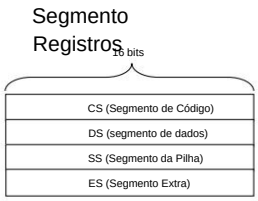
8

SP, BP, SI e DI

- **Stack Pointer (SP)** ÿ O registro SP de 16 bits fornece o valor de deslocamento dentro da pilha do programa.
- **Base Pointer (BP)** ÿ O registro BP de 16 bits ajuda principalmente na referência as variáveis de parâmetro passadas para uma sub-rotina.
- **Source Index (SI)** ÿ É usado como índice de origem para operações de string.
- **Índice de Destino (DI)** ÿ É usado como índice de destino para string operações.

9

Registros 8086/286



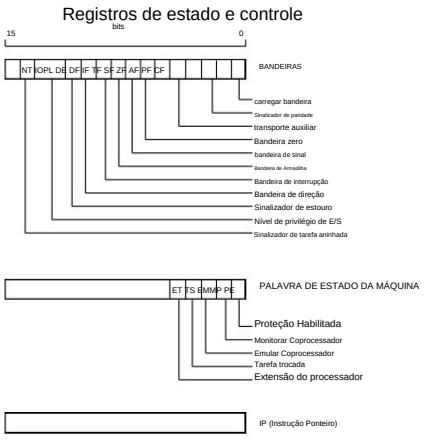
10

CS, DS e SS

- **Segmento de Código** ÿ Contém todas as instruções a serem executadas. Um código de 16 bits O registrador de segmento ou registrador CS armazena o endereço inicial do segmento de código.
- **Segmento de Dados** ÿ Contém dados, constantes e áreas de trabalho. Dados de 16 bits O registrador de segmento ou registrador DS armazena o endereço inicial do segmento de dados.
- **Segmento Stack** ÿ Contém dados e endereços de retorno de procedimentos ou sub-rotinas. Ele é implementado como uma estrutura de dados de 'pilha'. O registrador de segmento de pilha ou registrador SS armazena o endereço inicial da pilha.

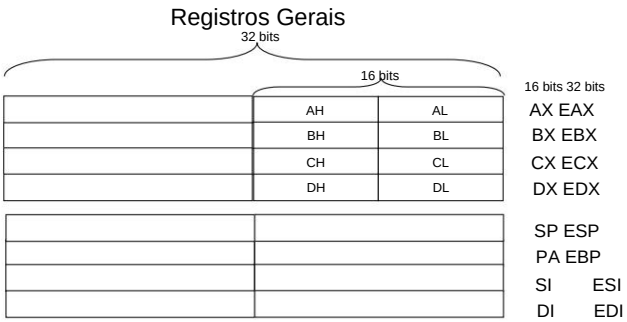
11

Registros 8086/286



12

386 registros



Encontre o valor de AX, AH e AL em decimal se colocarmos 04 03 02 01 H (hex) em EAX

13

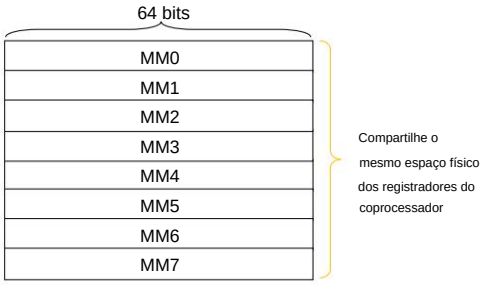
Encontre o valor de AX, AH e AL em decimal se colocarmos 04 03 02 01 (hex) em EAX

- EAX: **04 03 02 01**
- AX: 02 01
- AH: 02
- AL: 01

| 0000 0001 0010 0011 0100 0101 0110 0111 | ------> EAX
| 0100 0101 0110 0111 | ------> AX |
0110 0111 | ------> AL
| 0100 0101 | ------> AH

14

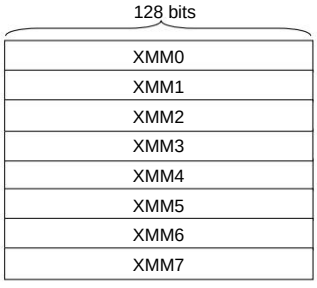
Registros de notícias em Pentium MMX (Não
registros gerais. A CPU é uma CPU de 32 bits)



15

Registros de notícias em Pentium II (Não
registros gerais. A CPU é uma CPU de 32 bits)

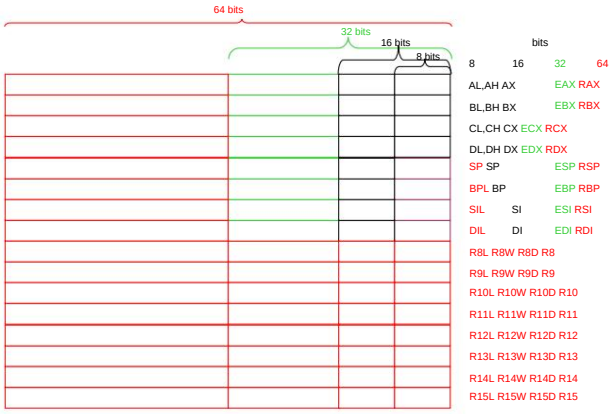
registradores XMM



16

Tecnologia EM64T

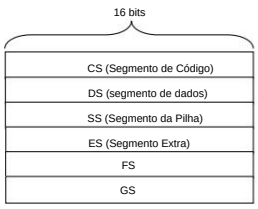
Registros Gerais



17

Tecnologia EM64T

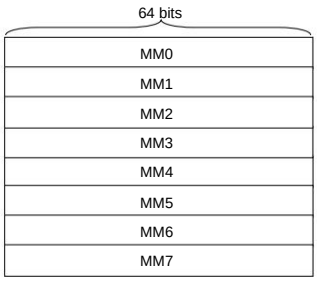
registradores de segmento
(tudo no mesmo)



18

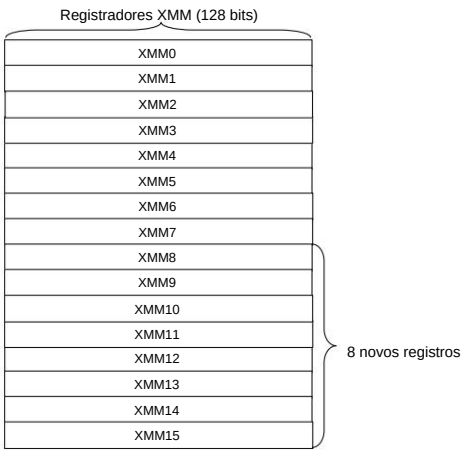
Tecnologia EM64T

Registradores
MMX (todos no mesmo)



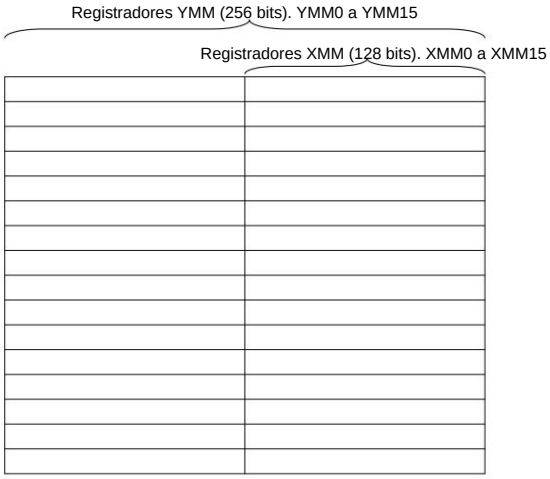
19

Tecnologia EM64T



20

Tecnologia AVX (Advanced Vector eXtension)



21

Registro

- Um registrador é uma área de armazenamento temporário embutida em uma CPU.
- Uma CPU registra um pequeno conjunto de locais de armazenamento de dados incorporados a uma CPU.

- Um registrador pode conter uma instrução, um endereço de armazenamento ou qualquer tipo de dados (como uma sequência de bits ou caracteres individuais). • Algumas instruções especificam registradores como parte da instrução.
  Uma instrução pode especificar que o conteúdo de dois registradores definidos ser somados e então colocados em um registrador especificado.

- Um registrador deve ser grande o suficiente para conter uma instrução.
  Um computador de 64 bits, um registro deve ter 64 bits de comprimento.

22

Instrução MOV

MOV é a instrução obrigatória na montagem

O formato MOV é:

destino MOV , *fonte*

Esta instrução move os conteúdos da *fonte* para o *destino*

Origem e *destino* devem ter o mesmo tamanho

O destino pode ser um registro ou referência de memória (variável ou endereço)

A fonte pode ser um registrador, uma referência de memória ou um valor

Simultâneo não é possível usar referência de memória como *destino* e *fonte*

23

Exemplos de instrução MOV

MOV AH,AL; coloca os conteudos de AL em AH (registradores de 8 bits)

MOVAX,BX; coloca os conteudos de BX em AX (registradores de 16 bits)

MOVECX,EDX; coloque os conteudos de EDX em ECX (registradores de 32 bits)

MOV EAX,BX; instrução inválida. Parâmetros com tamanho diferente

MOV AL,10; coloque o decimal 10 em AL

MOV AL,10B; coloque o binário 00000010 em AL

MOV AL,2FH; coloque o hexadecimal 2F em AL

MOV AL,0A2H; coloque o hex A2 em AL. (se o primeiro dígito hexadecimal for uma letra deve ser precedido por 0.

MOV AL,'A'; coloque o ASCII 'A' (decimal 65), em AL

MOV AL,256; 256 como 9 bits (100000000B). Esta instrução é válida mas só coloca o valor 00000000B em AL. Todos os bits à esquerda do bit 7 são ignorados

24

Instrução MOV
exemplos

Se você definir as variáveis int x,y; // inteiro de 32 bits

MOV x,0; colocar 0 na variável x;

MOVx,EAX; coloque os conteúdos de EAX na variável x

MOVx,AX; instrução inválida. Parâmetros com tamanho diferente

MOVEBX,x; coloque o conteúdo da variável x no EBX

MOV AL,x; instrução inválida. Parâmetros com tamanho diferente

MOV x,y; instrução inválida. Os dois parâmetros não podem ser referência de memória simultânea

25

Instrução MOV
exemplos

Se EDI como um valor de endereço (32 bits)

MOV [EDI], AL; coloca os conteudos de AL na memoria no endereço EDI

Endereço	contenda (HEX)
EDI-2	??
EDI-1	??
EDI	AL
EDI+1	??
EDI+2	??
EDI+3	??
EDI+4	??
EDI+5	??

SI = Índice de Fonte

DI = Índice de destino

26

ESI e EDI

• ESI e EDI são registradores de uso geral. • Se uma

variável deve ter classe de armazenamento de registro, ela geralmente é armazenada em ESI ou EDI.

- Algumas instruções usam ESI e EDI como ponteiros para endereços de origem e destino ao copiar um bloco de dados.

27

Instrução MOV

exemplos

Se EDI como um valor de endereço (32 bits)

MOV [EDI], AX; coloque os conteudos de AX na memoria no endereco EDI

Endereço	contenda (HEX)
EDI-2	??
EDI-1	??
EDI	AL
EDI+1	AH
EDI+2	??
EDI+3	??
EDI+4	??
EDI+5	??

MACHADO

28

Instrução MOV
exemplos

Se EDI como um valor de endereço (32 bits)

MOV [EDI],EAX; coloca os conteudos do EAX na memoria no endereço EDI

Endereço	contenda (HEX)	
EDI-2	??	
EDI-1	??	
EDI	AL	EAX
EDI+1	AH	
EDI+2	XX	
EDI+3	XX	
EDI+4	??	
EDI+5	??	

29

Instrução MOV
exemplos

Se EDI como um valor de endereço (32 bits)

MOV [EDI],1; Erro se o compilador não souber o tamanho do ponteiro [EDI]

MOV BYTE PTR [EDI],1; Sempre funciona. BYTE PTR define [EDI] como um ponteiro para 1 byte

Endereço	contenda (HEX)	
EDI-2	??	
EDI-1	??	
EDI	1	1 representado com 1 byte
EDI+1	??	
EDI+2	??	
EDI+3	??	
EDI+4	??	
EDI+5	??	

30

Instrução MOV
exemplos

Se EDI como um valor de endereço (32 bits)

PALAVRA MOV PTR [EDI],1; WORD PTR define [EDI] como um ponteiro para 2 bytes

Endereço	contenda (HEX)
EDI-2	??
EDI-1	??
EDI	1
EDI+1	0
EDI+2	??
EDI+3	??
EDI+4	??
EDI+5	??

1 representado com 2 bytes

31

Instrução MOV
exemplos

Se EDI como um valor de endereço (32 bits)

MOV DWORD PTR [EDI],1; DWORD PTR define [EDI] como um ponteiro para 4 bytes

Endereço	contenda (HEX)
EDI-2	??
EDI-1	??
EDI	1
EDI+1	0
EDI+2	0
EDI+3	0
EDI+4	??
EDI+5	??

1 representado com 4 bytes

32

Instrução MOV
exemplos

Se EDI como um valor de endereço (32 bits)

MOV [EDI+1],AL; coloque os conteudos de AL na memoria no endereço EDI+1

Endereço	contenda (HEX)
EDI-2	??
EDI-1	??
EDI	??
EDI+1	AL
EDI+2	??
EDI+3	??
EDI+4	??
EDI+5	??

33

Instrução MOV
exemplos

Se EDI como um valor de endereço e EBX um valor de índice (4)

MOV [EDI+EBX],AL; coloque os conteudos de AL na memoria no endereço EDI+EBX ou neste caso EDI+4

Endereço	contenda (HEX)
EDI-2	??
EDI-1	??
EDI	??
EDI+1	??
EDI+2	??
EDI+3	??
EDI+4	AL
EDI+5	??

34

Instrução MOV
exemplos

É possível multiplicar o índice por 2, 4 ou 8

Se EDI como um valor de endereço e EBX um valor de índice (2)

MOV [EDI+EBX*2],AL; coloca os conteudos de AL na memoria no endereço EDI+EBX*2 ou neste caso EDI+2*2=EDI+4

Endereço	contenda (HEX)
EDI-2	??
EDI-1	??
EDI	??
EDI+1	??
EDI+2	??
EDI+3	??
EDI+4	AL
EDI+5	??

35

Instrução MOV
exemplos

Se você definir a variável char lista[100]; //array de 100 caracteres

MOV lista[0],AL; coloque os conteudos de AL na memoria no endereço lista + 0

Endereço	contenda (HEX)
+0	AL
lista+1	??
lista+2	??
lista+3	??
lista+4	??
lista+5	??
lista+6	??
lista+7	??

36

Instrução MOV
exemplos

Se você definir a variável char lista[100]; //array de 100 caracteres

MOV lista[0],AX; Instrução inválida. Tamanho de parâmetro diferente

37

Instrução MOV
exemplos

Se você definir a variável char lista[100]; //array de 100 caracteres

MOV lista[EDI],AL; coloque os conteudos de AL na memoria no endereço lista + EDI.
Se EDI for 3 significa lista+3

Endereço	contenda (HEX)
lista+0	??
lista+1	??
lista+2	??
lista+3	AL
lista+4	??
lista+5	??
lista+6	??
lista+7	??

O índice EDI, neste caso, pode ser multiplicado por 2, 4 ou 8

38

Instrução MOV
exemplos

Se você definir a variável int lista[100]; //array de 100 inteiros

MOV EAX,1

MOV lista[0],EAX; coloca os conteudos do EAX na memoria no endereço lista + 0

Endereço	contenda (HEX)
lista+0	1
lista+1	0
lista+2	0
lista+3	0
lista+4	??
lista+5	??
lista+6	??
lista+7	??

39

Instrução MOV
exemplos

Existe uma grande diferença entre lista[n] usado em linguagens de alto nível e lista[n] usado em assembly

Em linguagens de alto nível, lista[n] identifica o elemento n na lista.

A posição da memória é lista+n*(tamanho do elemento) = lista+n*4 para elementos int

lista[1]=1;
Escreva o valor 1 no endereço lista+1*4

Endereço	contenda (HEX)
+0	??
lista+1	??
lista+2	??
lista+3	??
lista+4	1
lista+5	0
lista+6	0
+7	0

Na montagem lista[n] identifique o endereço na posição lista+n

Lista de MOV[1],1;
Escreva o valor 1 no endereço lista+1

Endereço	contenda (HEX)
+0	??
lista+1	1
lista+2	0
lista+3	0
lista+4	0
lista+5	??
lista+6	??
+7	??

40

Instrução ADICIONAR

O formato ADD é:

ADICIONAR *destino, origem* (destino=destino+origem)

ADD soma o *destino* com *a origem* e coloca o resultado no *destino*

A instrução ADD afeta os FLAGS

O ADD usa todas as regras apresentadas para a instrução mov

41

Mova os conteúdos da variável y para a variável x

MOV EAX, y;	coloque os conteúdos da variável y em EAX
MOV x, EAX;	coloque os conteúdos de EAX na variável x

42