ELE-8575

Modelo de Programação

Slides baseados no material do curso do Prof. Chen Lian Kuan

Objetivos

- ✓ Introduzir os modo de programação do 8086/8088;
- ✓ Descrever mais detalhadamente os registradores internos e suas funcionalidades;
- ✓ Descrever a diferença entre endereço lógico e endereço físico para memória e para portas de entrada/saída – E/S (ou input/output – I/O).

Registradores x86

Registradores no 8088/8086 podem ser agrupados da seguinte forma:

(i) General registers (AX, BX, CX, DX)

(ii) Pointer and Index registers (SP, BP, SI, DI, IP)

(iii) Flag Register (FLAGS)

(iv) Segment Registers (CS, DS, SS, ES)

Os microprocessadores 80386, 80486, Pentium e superiores têm registros de 32 bits ou maior, e os registros de 16 bits do 8088 formam um subconjunto.

Diagrama simplificado do 8086
Registradores de Segmento (BIU)

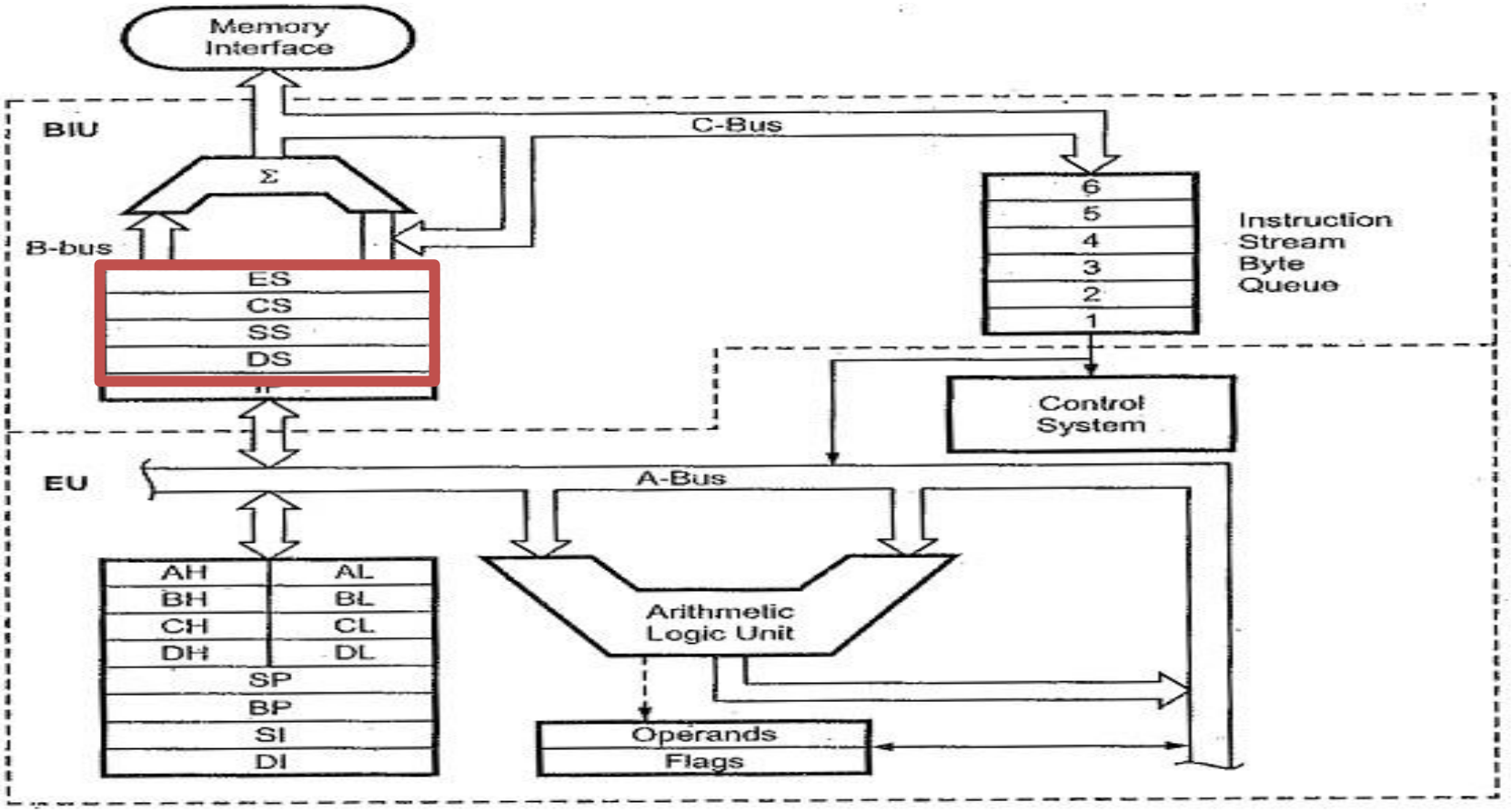


Fig. 6.2 8086 Internal block diagram

Diagrama simplificado do 8086
Ponteiro de Próxima Instrução (BIU)

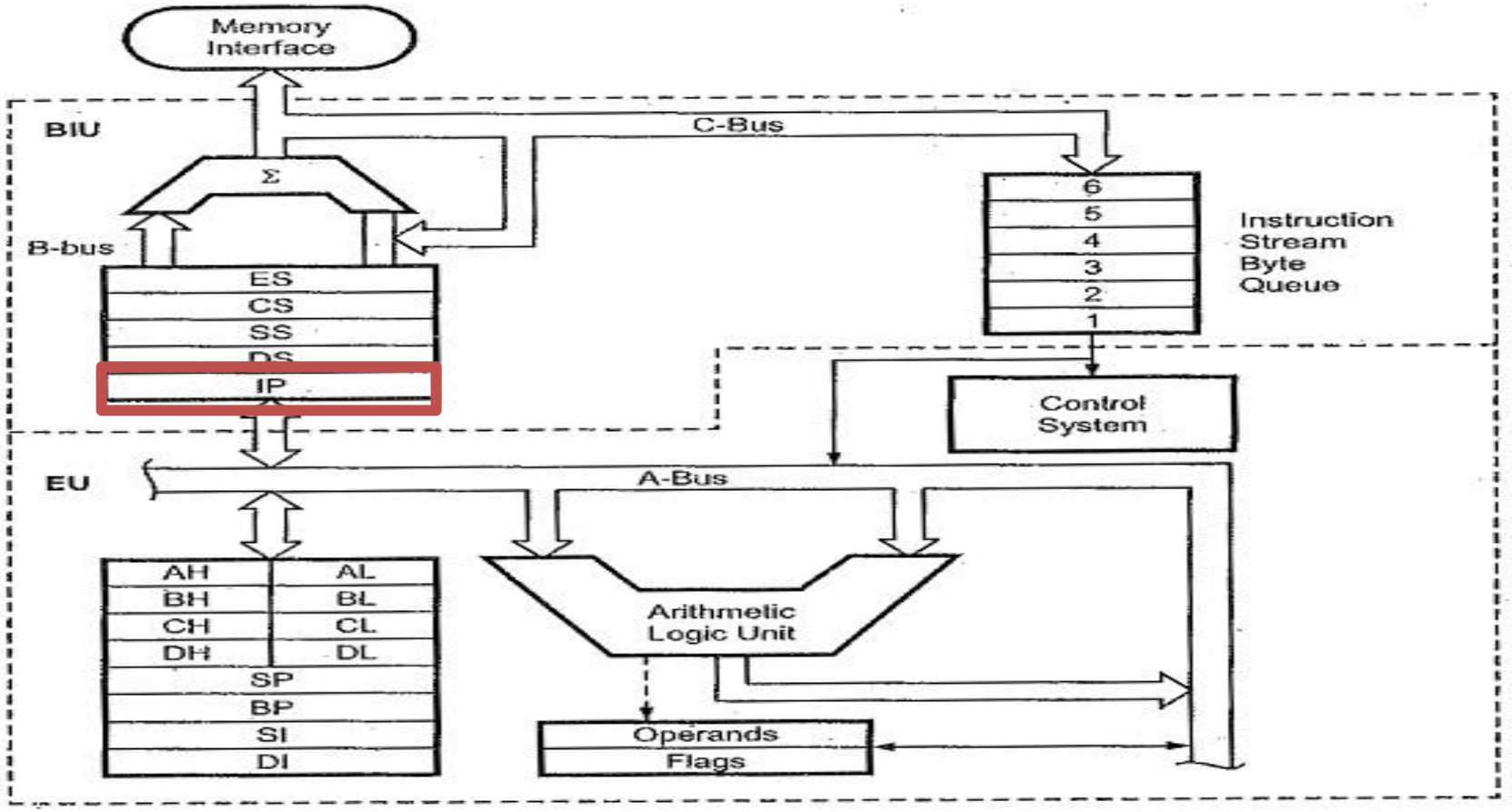


Fig. 6.2 8086 Internal block diagram

Diagrama simplificado do 8086
Registradores de Propósito Geral (EU)

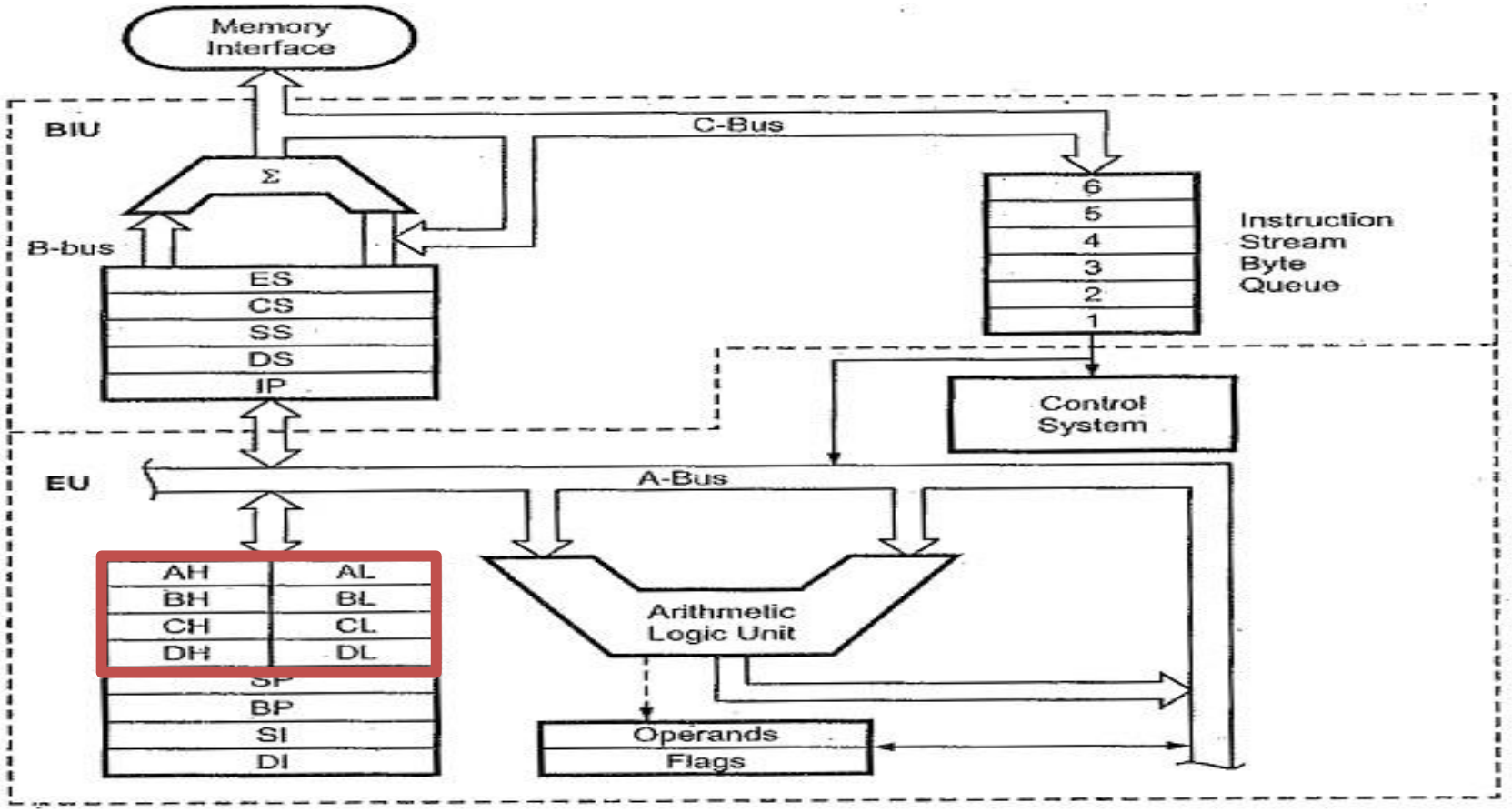


Fig. 6.2 8086 Internal block diagram

Diagrama simplificado do 8086
Pilha (SP e BP) e String (SI e DI) (EU)

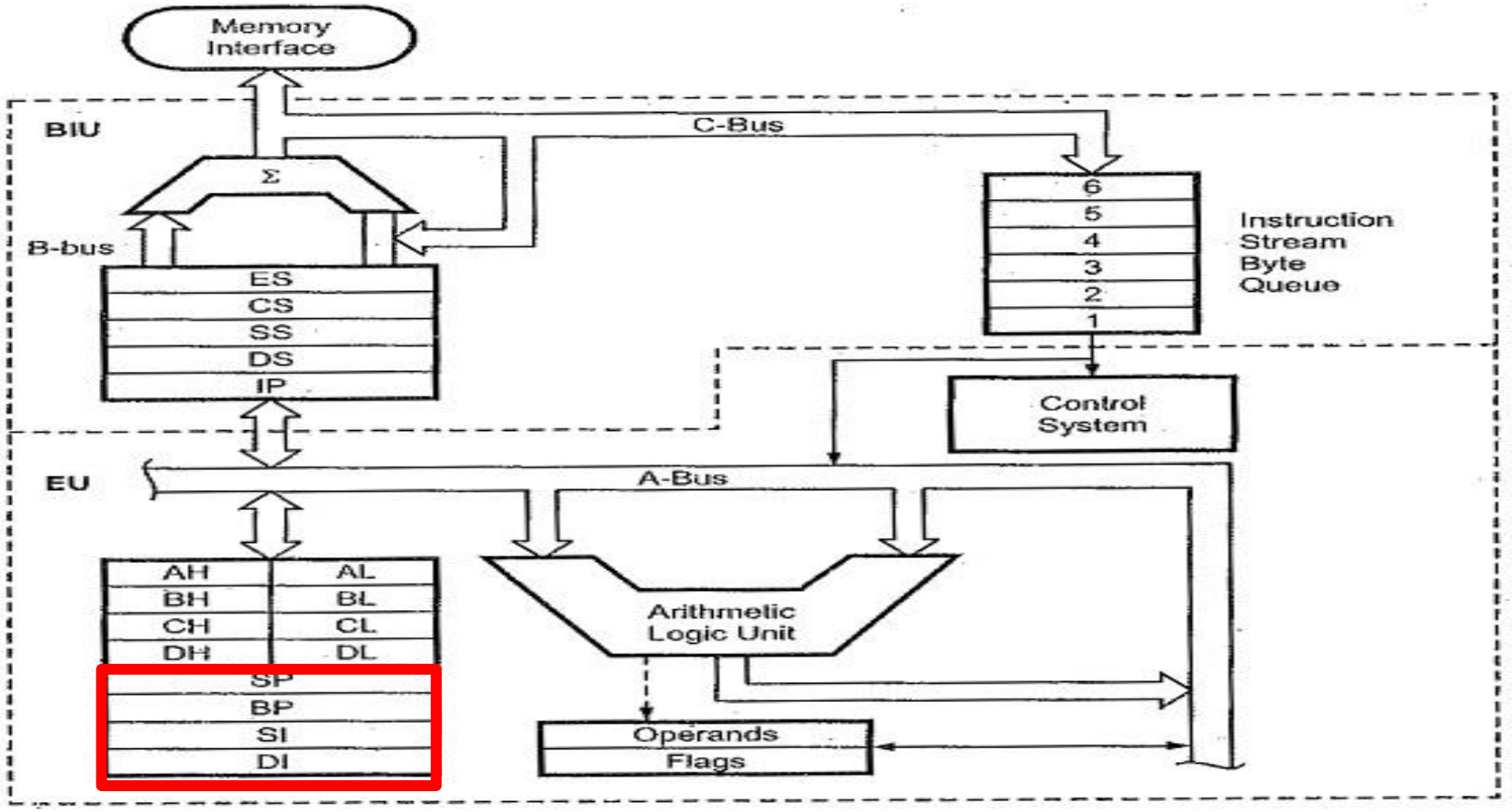


Fig. 6.2 8086 Internal block diagram

Diagrama simplificado do 8086
Registrador de *Flags* (EU)

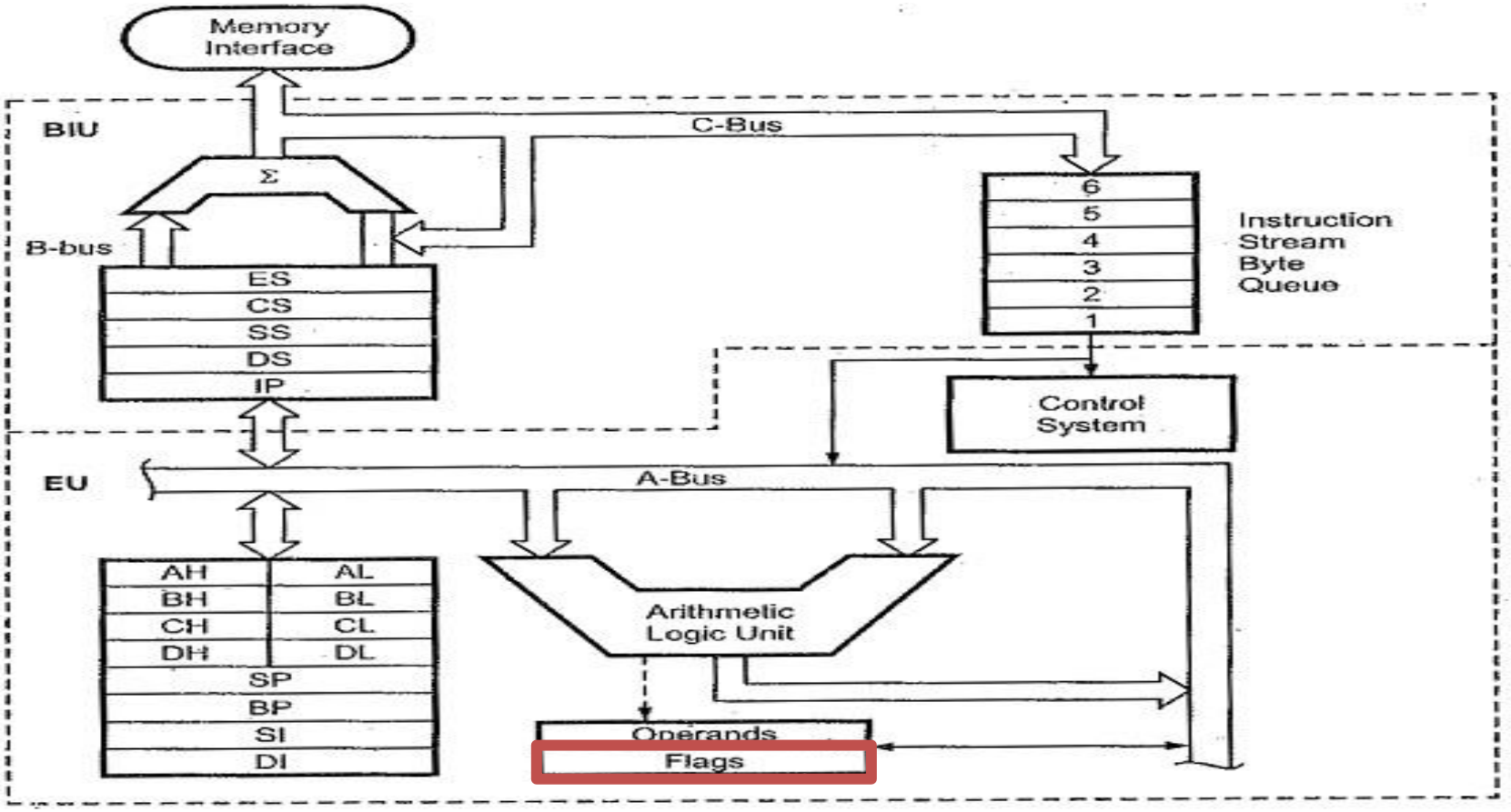
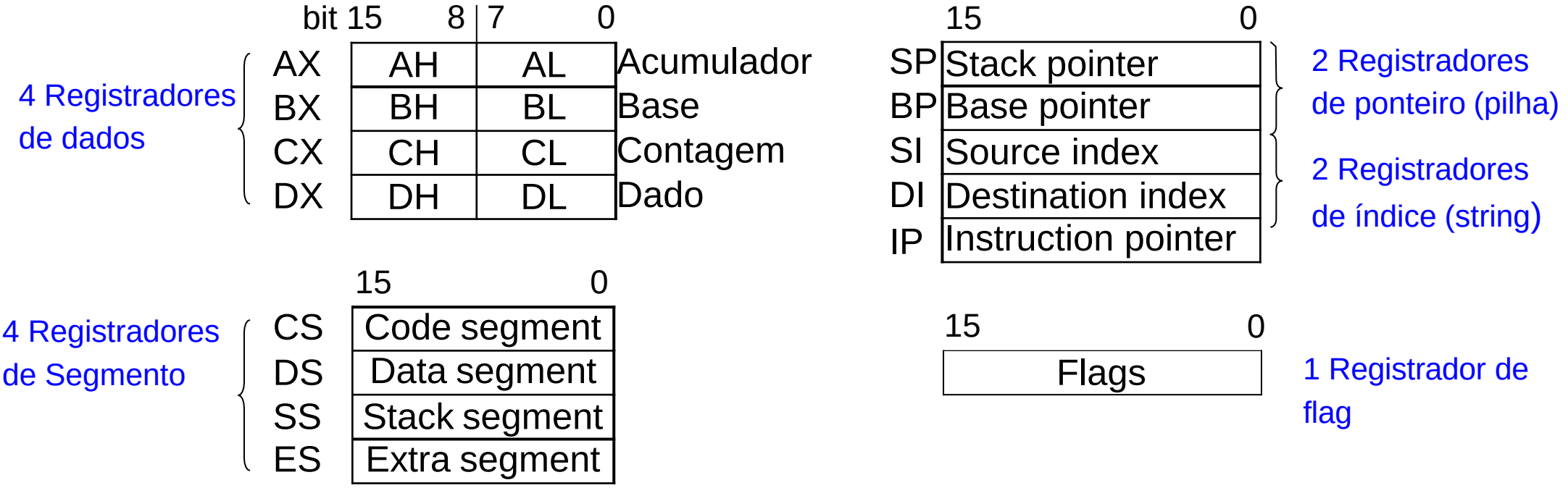


Fig. 6.2 8086 Internal block diagram

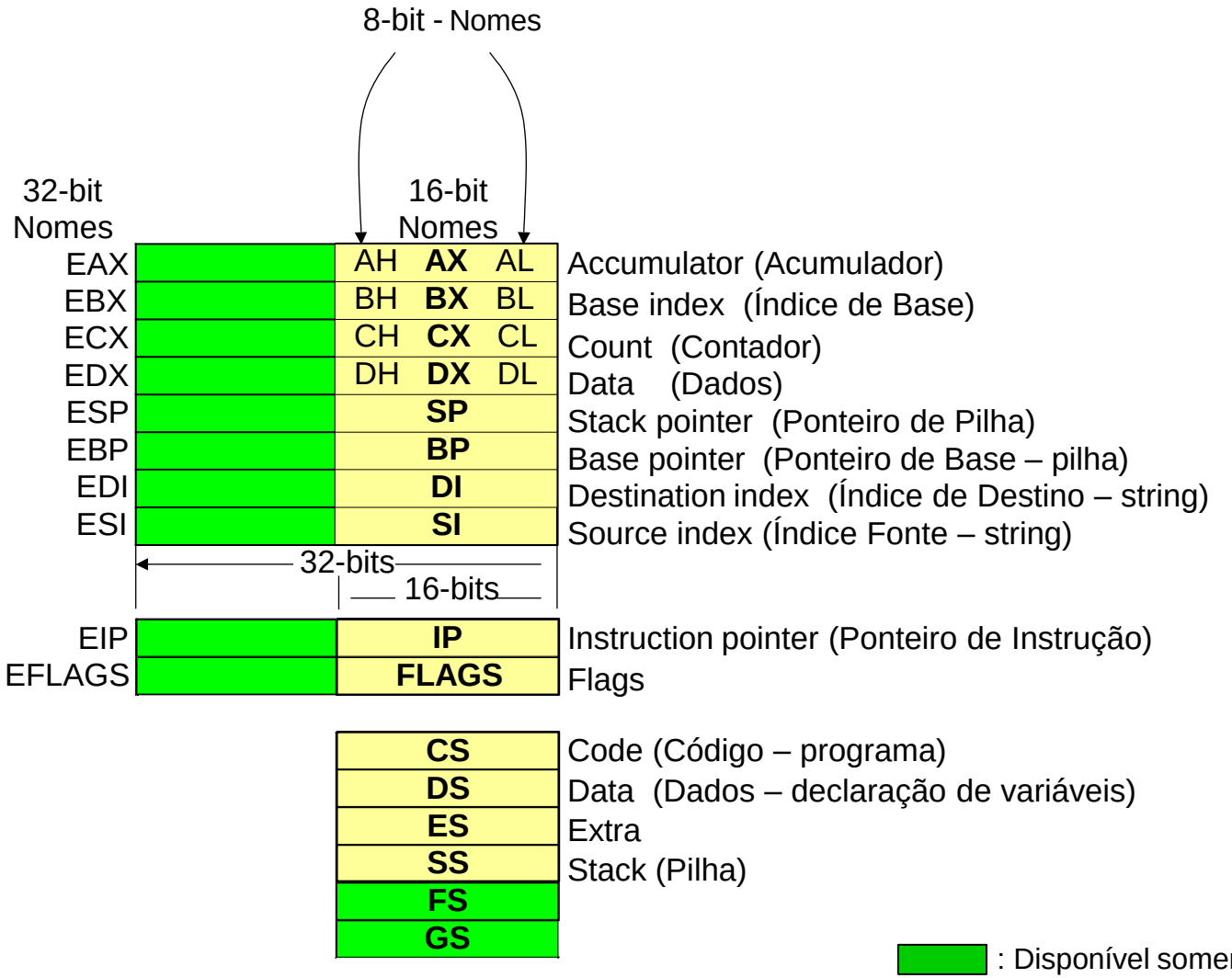
Modelo de Programação (visão do programador)

O modelo de programação (software) resume todas as informações necessárias para a programação do microprocessador.

Exemplo: Modelo de programação 8088



Modelo de Programação 8086/8088



Registradores Visíveis e não Visíveis

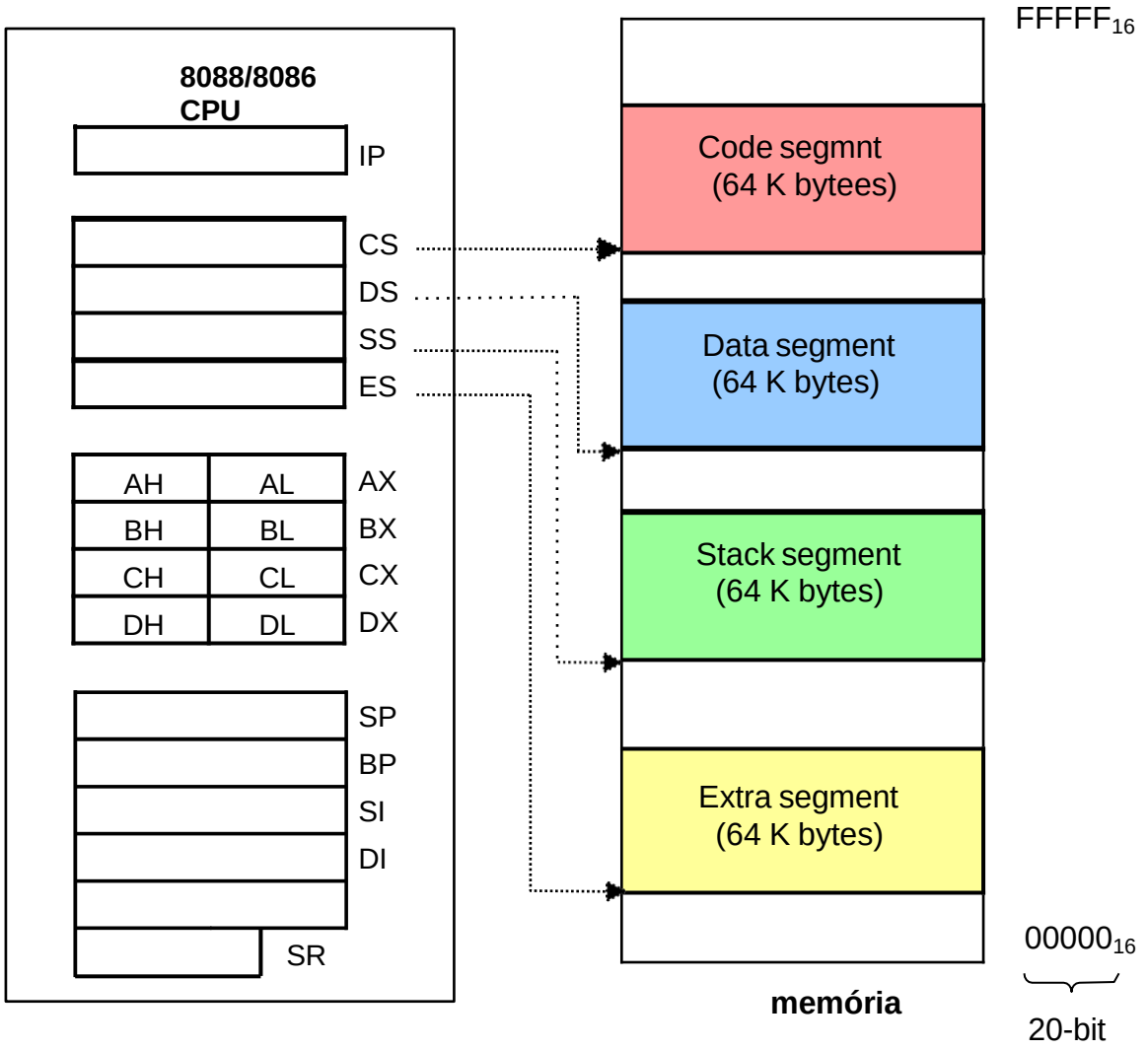
- ✓ O modelo de programação de um microprocessador contém uma descrição concisa dos registros internos e de suas funções.
- ✓ Somente os registros visíveis (ou seja, diretamente acessíveis por aplicações) são incluídos no modelo de programação.
- ✓ Todos os registros no modelo de programação para o 8086/8088 são afetados pelo conjunto de instruções.

Registradores de segmento e acesso à memória

Os registradores de segmento da CPU armazenam os endereços iniciais de memória onde serão carregados:
Programa (CS)
Declaração de variáveis (DS)
Pilha (SS)
Extra (por exemplo: outra RAM, heap ou usado para apontar para qualquer região da RAM do sistema sem precisar de mudar o conteúdo de CS, DS, SS)

Como fazer 16-bit → 20-bit ???
Os registradores de segmento são de 16 bits, mas o endereço físico é de 20 bits!!!!!

SR: status (flag)



Estado interno de µP x86 após a sua inicialização (reset)

CPU COMPONENT	CONTENT
Flags	Clear
Instruction Pointer	0000h
CS Register	FFFFh
DS Register	0000h
SS Register	0000h
ES Register	0000h
Queue	Empty

Registradores AX, BX, CX, DX

AX, BX, CX, DX são registradores de 16 bits e somente esses 4 registradores podem ser divididos e 2 outros de 8 bits cada.

	15 — 8	7 — 0	
AX	AH	AL	**accumulator**
BX	BH	BL	**base**
CX	CH	CL	**count**
DX	DH	DL	**data**

Além de serem usados como registros gerais, eles são usados como memória padrão (*default*) para algumas instruções:

AX (acumulador) armazena o resultado de muitas instruções aritméticas e lógicas.

BX (Base) armazena os dados de endereço da base (offset) na memória e o endereço da base de uma tabela de dados referenciados pela instrução de tradução (XLATB).

CX armazena a contagem para certas instruções (por exemplo, contador na instrução LOOP e em instruções de strings).

DX guarda a parte mais significativa do resultado de uma multiplicação de 16 bits, a parte mais significativa de um dividendo antes da divisão, ou o endereço de uma porta de E/S.

Registradores SP, BP, SI, DI

	15 … 0	
SP		Stack pointer
BP		Base pointer
DI		Destination Index
SI		Source Index
IP		Instruction Pointer

Este conjunto de registradores geralmente armazena endereços de *offset* de memória. IP armazena o endereço de *offset* da próxima instrução na memória. SP, BP, DI e SI normalmente armazenam o endereço de *offset* dos dados na memória.

SP, BP, DI e SI também podem ser usados para fins gerais (×, /, +, -, AND, OR, NOT, XOR, NEG)

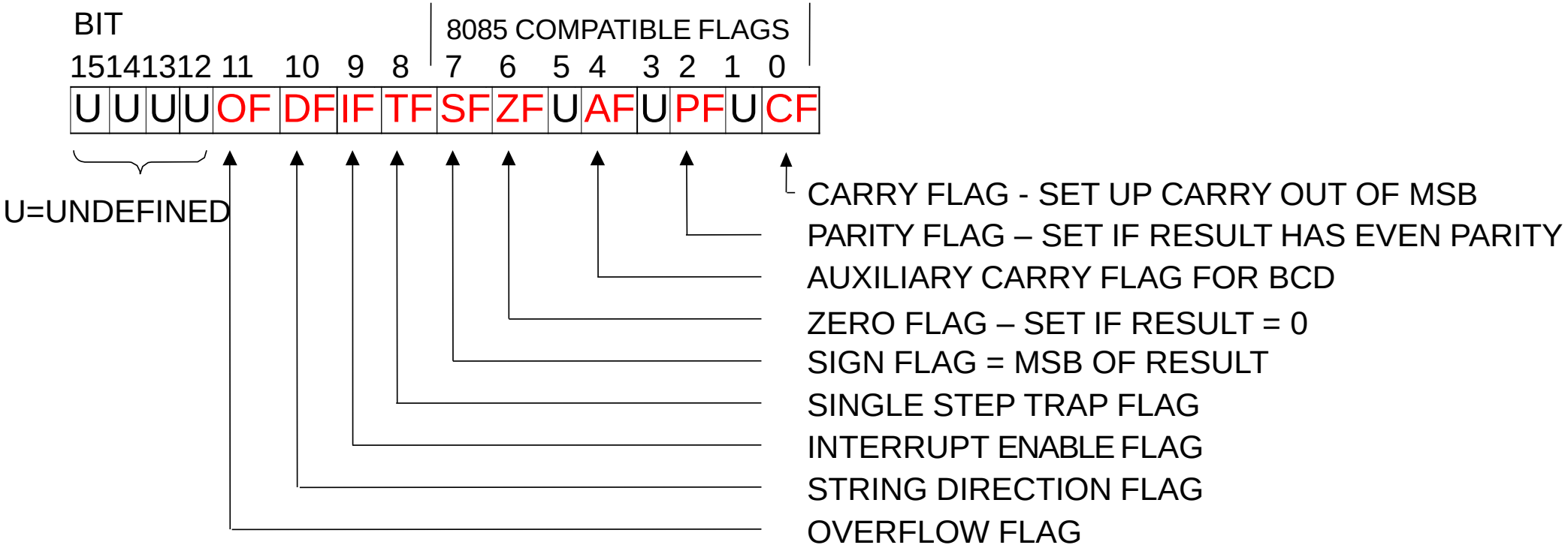
SP (Stack Pointer) é usado nas instruções PUSH e POP para operações em uma pilha LIFO (Last-In, First-Out). Na verdade SP é usado todas as vezes na qual se usa a pilha (interrupção e chamada de rotinas). Aponta para o topo da pilha.

BP (Base Pointer) é frequentemente usado no endereçamento de uma série de dados na memória da pilha.

DI (Destination Index) geralmente armazena o endereço de destino indireto dos dados de uma instrução, em geral de string.

SI (Source Index) é usado ao endereçar indiretamente os dados de origem em certas instruções de string.

8086/8080 – Bits do Registrador de Flags



Flags Register

9 dos 16 bits no registrador de flags podem ser = “1” quando certas situações ocorrem (ou seja, eles sinalizam um evento.) Os outros 7 bits não são utilizados.

FLAGS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				O	D	I	T	S	Z		A		P		C

Todos os bits valem “0” quando o µP é energizado.

Instruções de salto condicional testam os bits de flag O, S, Z, P e C.

Instruções (pushf, popf) ou (LAHF, SAHF) estão disponíveis para transferir o conteúdo do registrador de flags de/para a pilha ou de/para o registro AH (apenas os 8 bits inferiores de flag). Outras instruções estão disponíveis para manipulação de certos bits de flag (por exemplo, STI, CLI, STD, CLD, STC, CLC, CMC). Por exemplo, STI faz: I=“1”, CLI faz: I=“0”, CMC complementa a flag C (operação NOT).

FLAGS

FLAGS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					O	D	I	T	S	Z		A		P		C

As flags no registro de FLAGS ou são de condição ou de controle.

As flags de condição consistem em :

C (flag de transporte) - definida como “1” quando o resultado de uma adição tem uma execução do byte mais significativo. Outras instruções também podem afetar o C (por exemplo, subtração).

P (flag de paridade) - definida para “1” se o byte de baixa ordem do resultado contiver um número par de bytes; caso contrário, é definida para zero.

A (flag de transporte auxiliar) - definida para “1” se houver uma situação de vai-1 do bit-3 durante uma adição ou um empréstimo para o bit-3 durante a subtração.

Z (flag de zero) - definida para “1” se o resultado for zero; caso contrário, Z é “0”.

S (flag de sinal) - igual ao bit mais significativo do resultado (ou seja, ajustado para “1” se o resultado for negativo)

O (flag de transbordo) - definida se um resultado estiver fora do alcance (por exemplo, ao adicionar dois números positivos e o resultado é um número negativo).

Flags de Controle

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLAGS					O	D	I	T	S	Z		A		P		C

Três bits (D, I, T) nos registros de bandeiras controlam a operação do microprocessador nas seguintes circunstâncias:

D (direction flag) - em certas instruções de manipulação de strings, D determina se a string é processada a partir do endereço mais baixo (D=0) ou endereço mais alto (D=1). Ou seja, o endereço:

D="0": auto-incrementa e D="1": auto-decrementa.

I (interrupt flag) – determina quando uma interrupçãodo tipo mascarável (INT) será reconhecida pelo processador. Se I="1", a interrupção será aceita, caso contrário, a interrupção será ignorada.

T (trap flag) - se T="1", o programa é executado passo a passo.

Exemplo de efeito de uma operação de adição no registro de FLAGS

Exemplo 1

```
  0010 0011 0100 0101
+ 0011 0010 0001 1001
  0101 0101 0101 1110
```

Os bits de flags ficam:

S (sign) =0, Z (zero) =0, P (parity) =0, C (carry) =0,

A (aux carry) =0, O (overflow) =0

Exemplo 2

```
  0101 0100 0011 1001
+ 0100 0101 0110 1010
  1001 1001 1010 0011
```

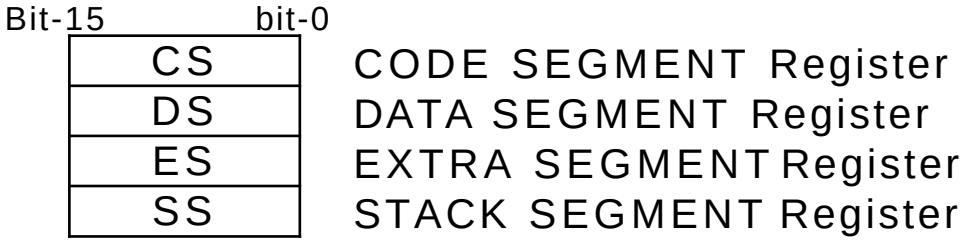
Os bits de flags ficam:

S (sign)=1 , Z (zero)=0, P (parity) =1, C (carry) =0,

A (aux carry) =1, O (overflow) =1

Regs de Segmento e Segmentos de Memória

Registradores de segmento são de 16-bit registers e quando usados em conjunto com registradores de índice e ponteiros geram o endereço físico de 20 bits



Segmentos de Memória:

Code Segment é a seção de memória usada para armazenar as instruções e procedimentos do programa. O registro CS armazena o endereço inicial do código do programa. No 8086 (e 80286) o segmento de código é limitado a 64K bytes de comprimento (no 80386 o comprimento máximo é de 4G bytes).

Data Segment contém os dados usados pelo programa. O registrador DS, como qualquer outro registrador de segmento, armazena o endereço inicial do segmento.

Regs de Segmento e Segmentos de Memória

Extra Segment segmento usado em operações específicas de manipulação string.

Stack Segment define a localização da pilha na memória.

- ✓ Um segmento no sistema 8088/8086 é uma seção contínua de memória de até 64K bytes de comprimento (64K de informações de 1 byte; ou 64 KB).

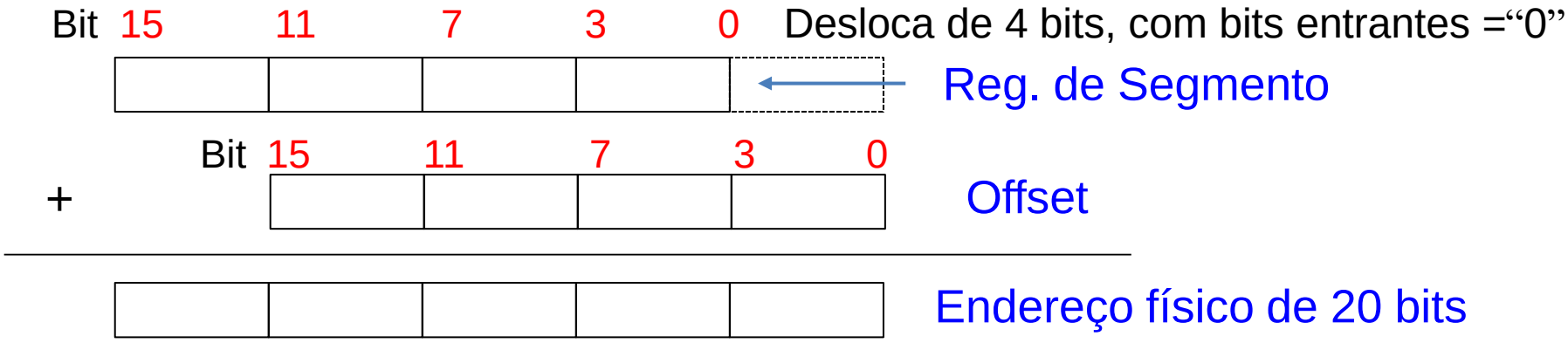
- ✓ O bloco de 64 Kbyte definido pelo endereço inicial de um segmento pode se sobrepor a outros segmentos ou estar completamente em sua própria área de memória separada.

- ✓ Os segmentos permitem que pacotes de informações (por exemplo, uma tabela de dados, ou uma sub-rotina) sejam mantidos separadamente - não é necessário preencher todos os 64K do segmento e o programador pode fazer o segmento de tamanho arbitrário (até 64K bytes, em incrementos de 16 bytes).

Segmentos e Offsets

- ✓ O endereço de 20 bits para a memória é gerado pela adição de um offset de 16 bits ao conteúdo de um reg. de segmento de 16 bits deslocado. O endereço do segmento define o início de um bloco de memória de 64K-byte dentro do espaço de 1Mbyte, e o offset define a localização exata da memória dentro desse bloco de 64Kbyte.

- ✓ Na verdade, o endereço de 20 bits é formado pelo deslocamento 4 bits para a esquerda do conteúdo do reg. segmento de 16 bits, somado a um offset de 16 bits.



Segments and Offsets (Cont.)

- ✓ Uma vez que o endereço do segmento é derivado do deslocamento de 4 bits, formando um endereço de 20 bits, o endereço inicial de um segmento só pode ocorrer em intervalos de 16 bytes. Endereço de início válido 00000h, 00010h, 00020h, ...
- ✓ Aumentando-se em 1 o valor do registro de segmento, aumenta-se em 16 o endereço físico inicial.

- ✓ As vantagens do método segmento + offset são:
- ✓ O código do programa pode ser facilmente realocado na memória (útil para multitarefas);
- ✓ A maioria das operações podem ser realizadas alterando-se apenas o offset de 16 bits. O offset é armazenado em registros de 16 bits permitindo uma interface mais fácil para memória de 8 e 16 bits de largura.

- ✓ Pergunta: Como sabemos qual endereço do segmento deve ser usado para um determinado endereço de offset?

Regras de combinação de Registros de Segmento e Offset

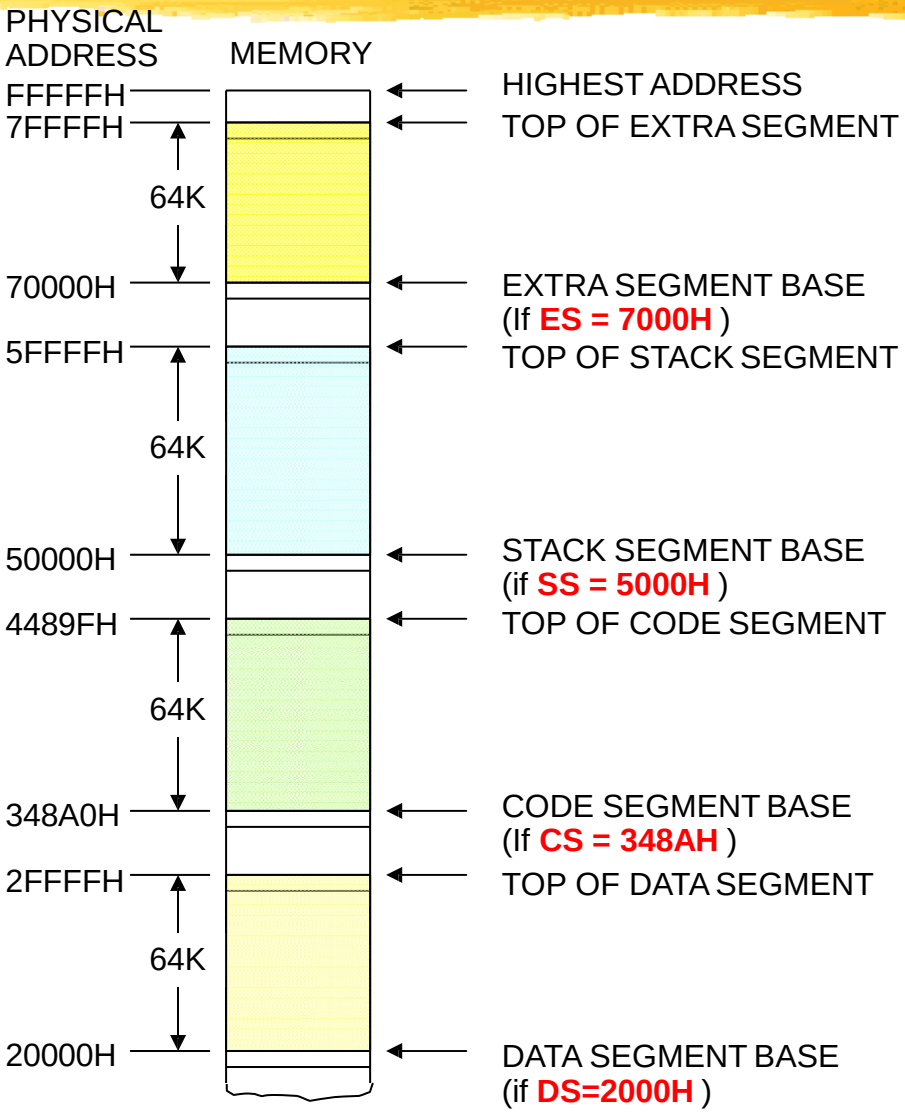
O microprocessador possui um conjunto de regras que definem a combinação:
registro de segmento: registro de offset (exemplos: CS:IP, SS:SP, ES:DI, DS:SI)
utilizada por certos modos de endereçamento. Observe que na maioria das vezes não é necessário especificar o registro de segmento utilizado pois ele é assumido por default.
No entanto, o padrão pode ser modificado explicitando-se na instrução o registro de segmento a ser utilizado.

MOV CL, [BP] ; faz: CL← memória(SS0 + BP), SS0 = (SS deslocado de 4 bits entrantes iguais a 0)

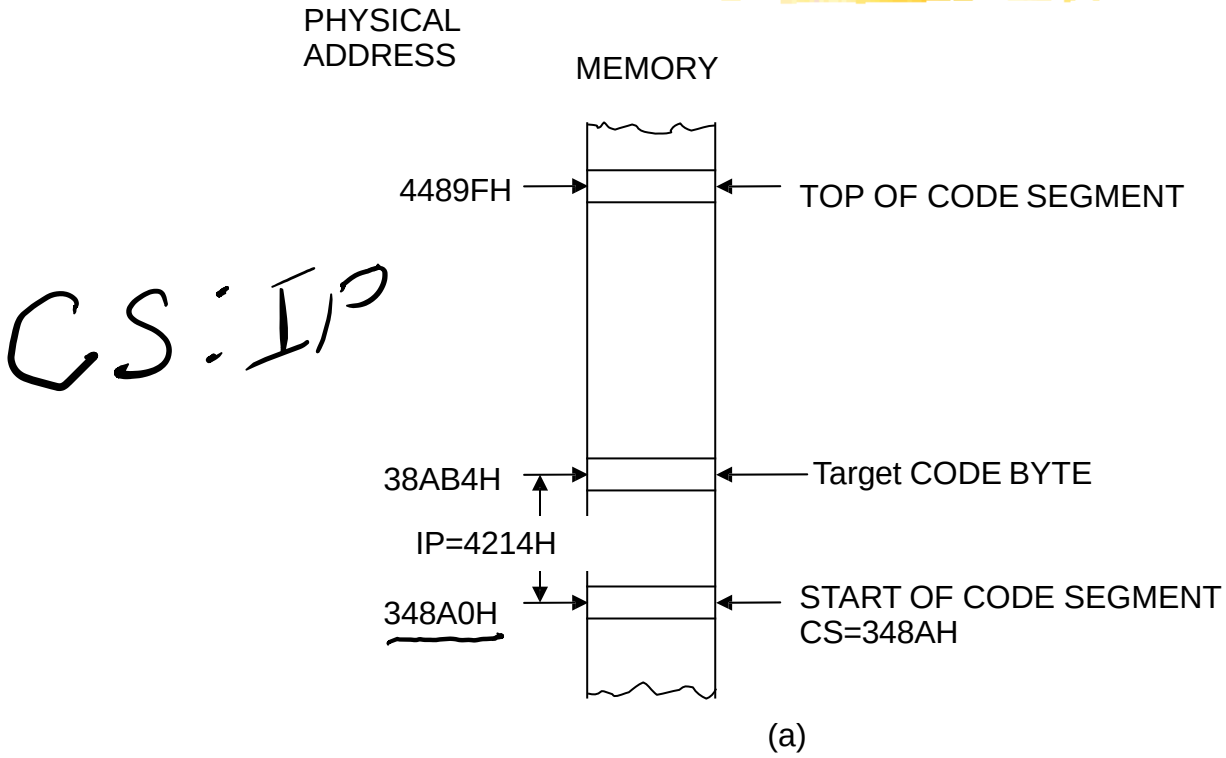
MOV CL, [DS:BP] ; faz: CL← memória(DS0 + BP), DS0 = (DS deslocado de 4 bits entrantes iguais a 0)

Registrador de Offset	Registrador de Segmento Default	Pode-se trocar de reg. de segmento?
IP	CS	Nunca
SP	SS	Nunca
BP	SS	DS, ES ou CS
SI, DI, (exceto para strings), BX, BP	DS	ES, SS ou CS
DI (para instruções de string)	ES	Nunca

Um exemplo de alocação de segmentos de 64 Kbyte dentro do espaço de endereços de 1Mbyte de um 8086/8088

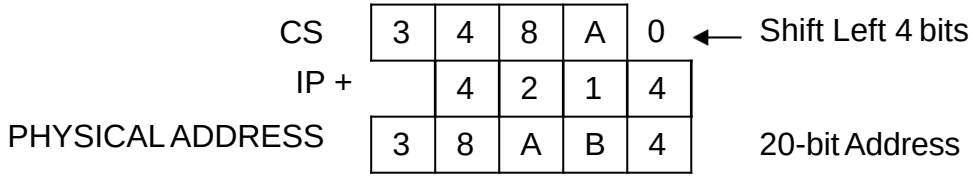


Exemplo de uso do par CS:IP



Diagram

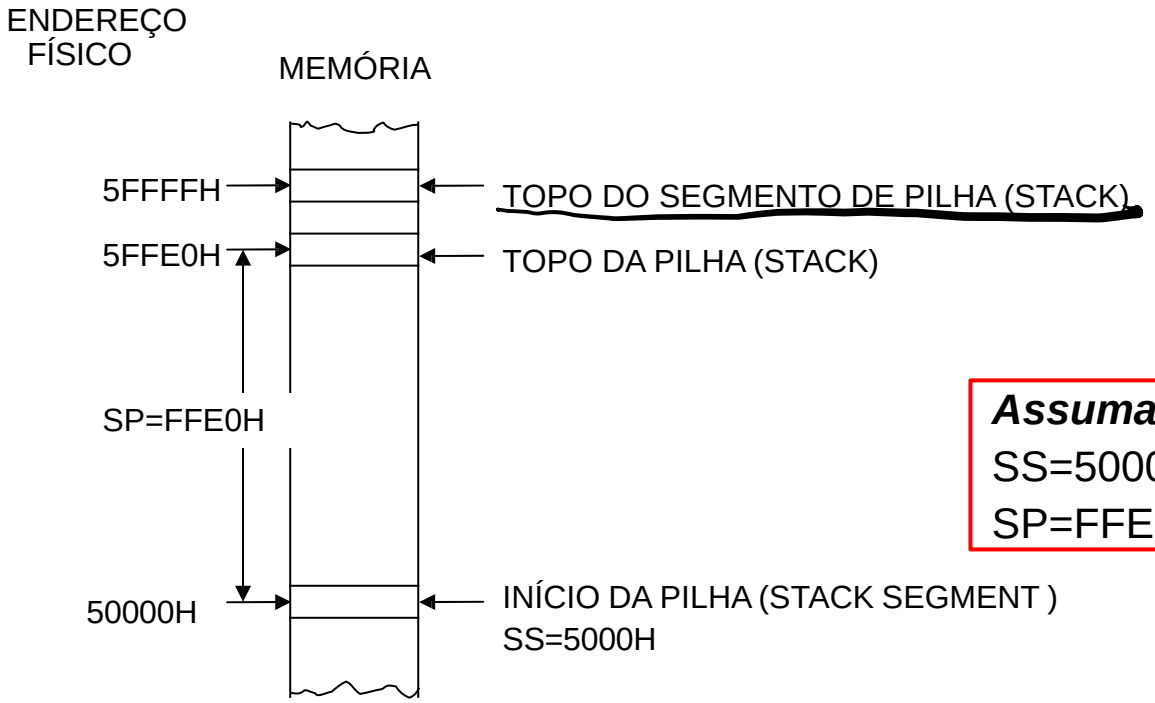
Assume
CS=348AH
IP=4214H



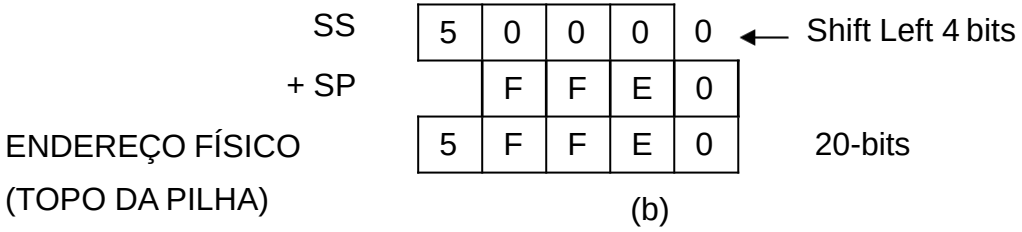
Computation

Par SS:SP apontando para o topo da pilha (stack)

Diagrama



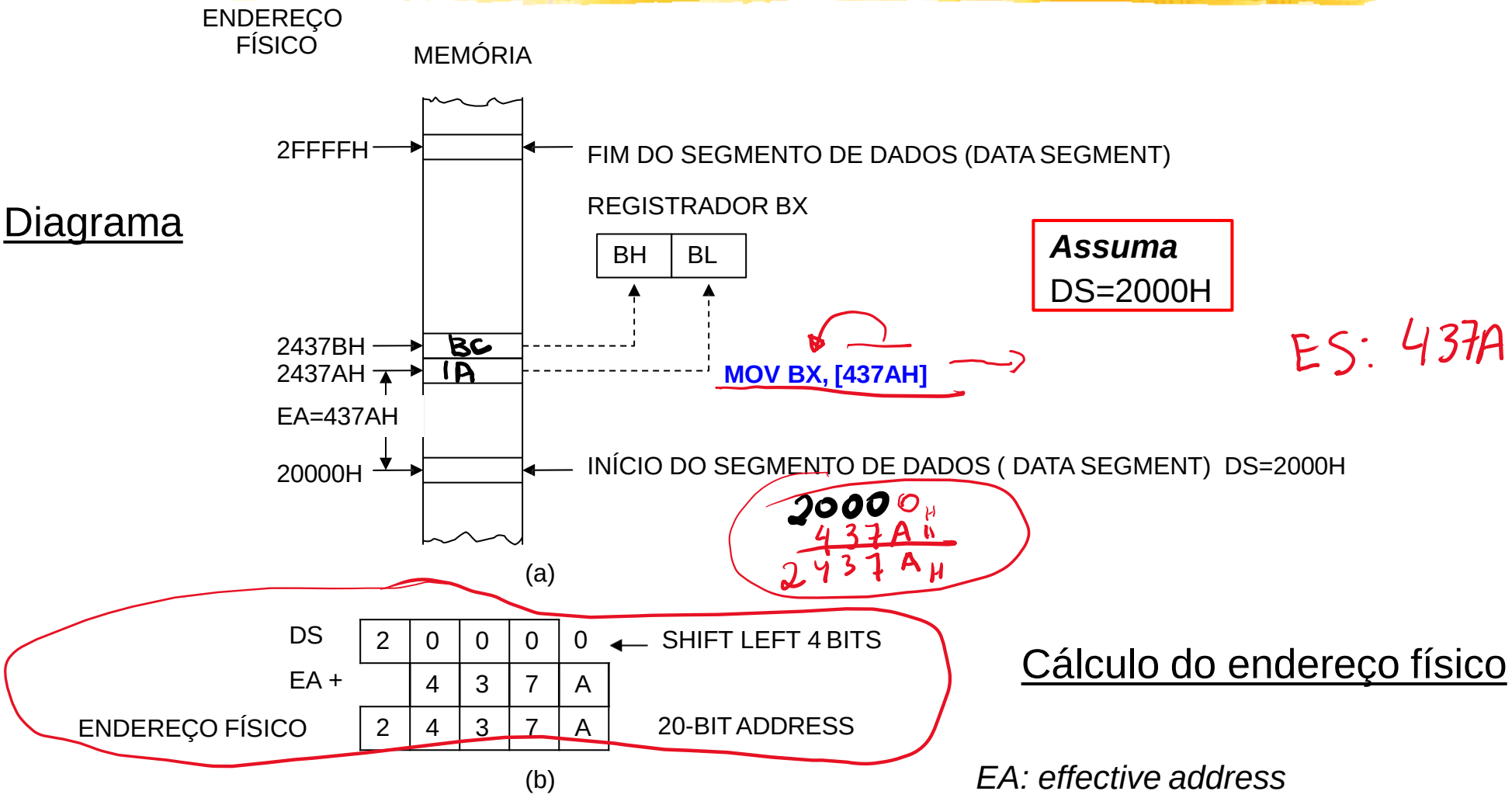
Assuma
SS=5000H
SP=FFE0H



Cálculo do endereço físico

Adição de DS e um endereço efetivo (EA) para produzir o endereço físico do dado em memória

Diagrama



Cálculo do endereço físico

EA: effective address

Representação de Números em distintas arquiteturas

Um número com vários bytes pode ser armazenado de 2 maneiras:

Arquitetura *Low Order Byte First* (LOBF - também chamada de *little endian*) - o byte menos significativo é colocado no primeiro (mais baixo) endereço de memória. Exemplos de microprocessadores que utilizam esta convenção incluem o Intel 8086/8088.

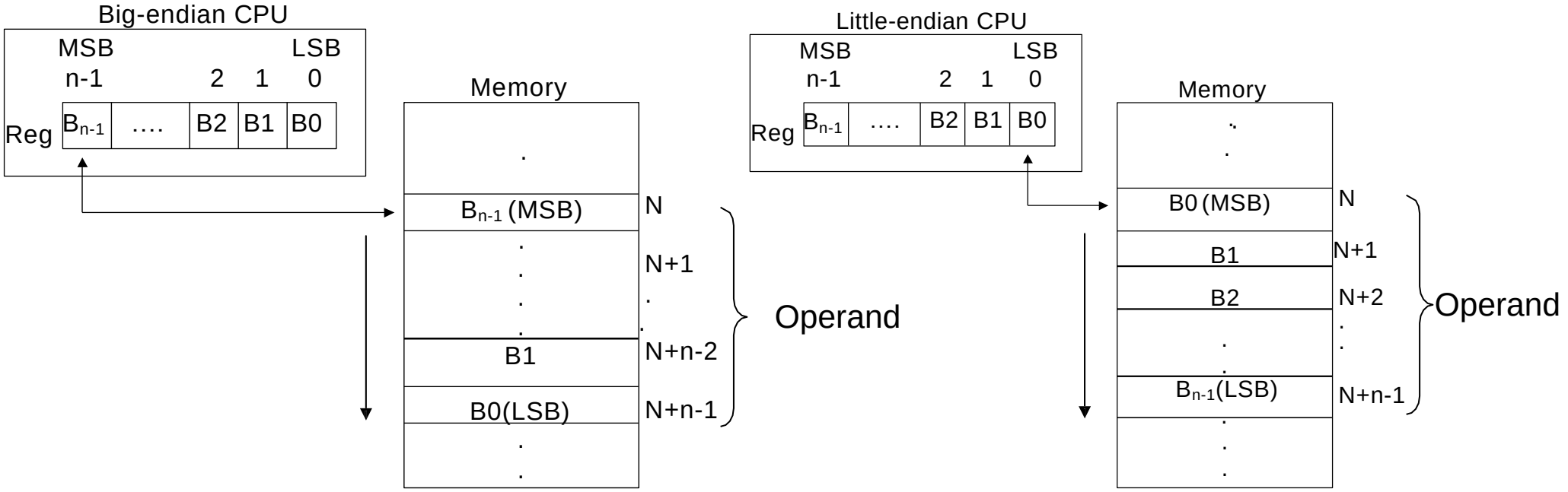
Arquitetura *High Order Byte First* (HOBF - também chamado de *big endian*) - o byte mais significativo é colocado em primeiro lugar. Exemplos incluem os microprocessadores Motorola 68x

Por exemplo: Colocar no endereço de memória 0xABCDE o valor 0x3A4B:

Little Endian: 0xABCDE recebe 0x4B e 0xABCDF recebe 0x3A

Big Endian: 0xABCDE recebe 0x3A e 0xABCDF recebe 0x4B

Armazenamento numérico em memória



Big-endian (high-order-byte-first) architecture

Little-endian (low-order-byte-first) architecture

Usado nos processadores 8086 and 8088