

PROCESSADORES 8086

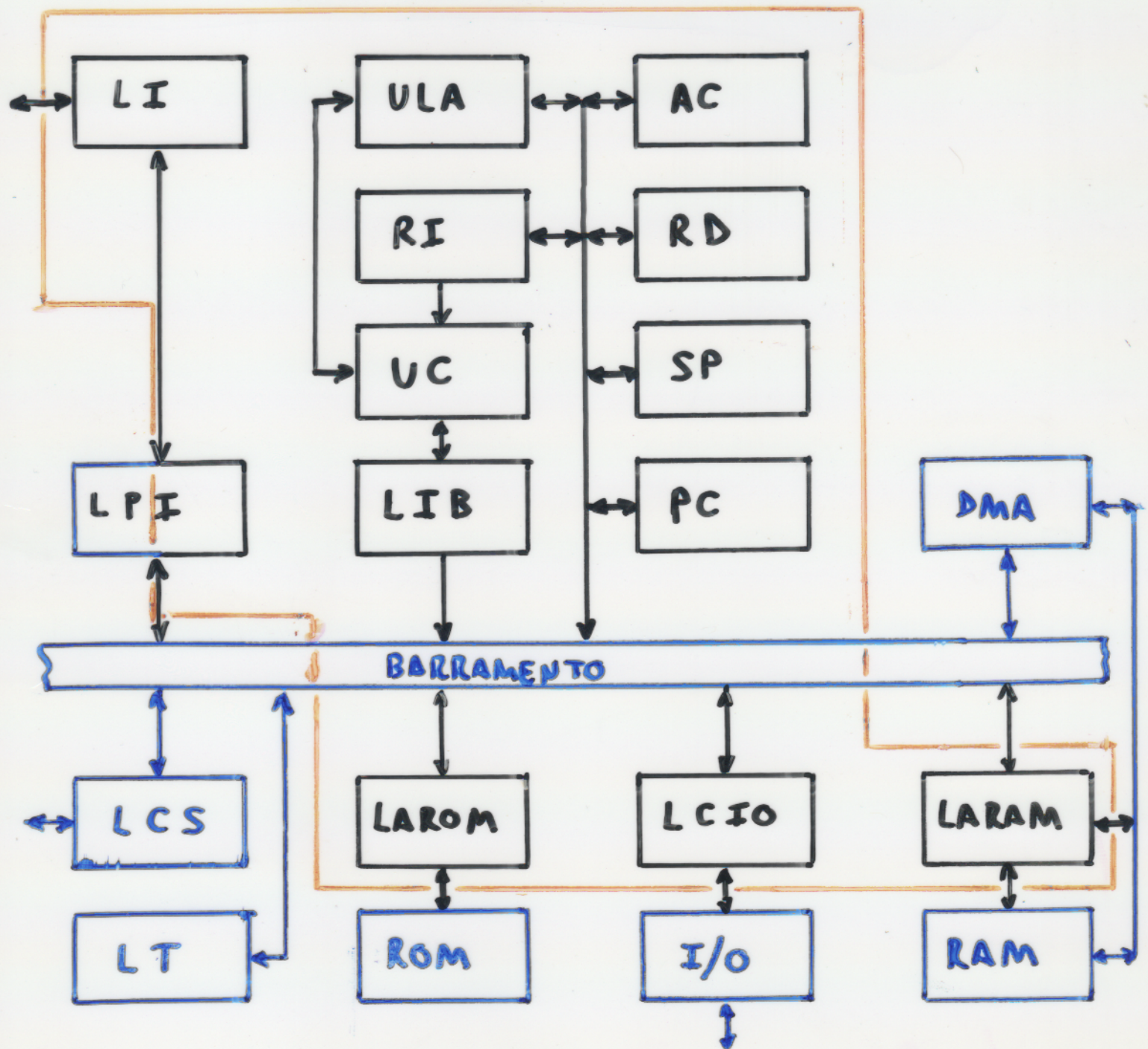
- * ARQUITETURA INTERNA
- * REGISTROS PROGRAMÁVEIS
- * MODOS DE ENDEREÇAMENTO
- * EXECUTION UNIT / BUS INTERFACE UNIT
- * I/O
- * MODOS DE INTERRUPÇÃO
- * PROCESSADOR 8088
- * EXEMPLOS DE INSTRUÇÕES

* ARQUITETURA INTERNA (CONT.)

LI - LÓGICA DE INTERRUPÇÃO
ULA - UNIDADE LÓGICA E ARITMÉTICA
AC - REGISTRO ACUMULADOR
RI - REGISTRO DE INSTRUÇÃO
RD - REGISTROS DE DADOS
VC - UNIDADE DE CONTROLE
SP - PONTEIRO DA STACK (STACK POINTER)
LIB - LÓGICA DE INTERFACE COM O BARRAMENTO
PC - CONTADOR DE PROGRAMA (PROGRAM COUNTER)

LPI - LÓGICA DE PRIORIDADE DE INTERRUPÇÃO
DMA - LÓGICA DE ACESSO DIRETO À MEMÓRIA
LCS - LÓGICA DO CIRCUITO DE SERIALIZAÇÃO
LT - LÓGICA DE TEMPORIZAÇÃO (TIMER)
LROM - LÓGICA DE ACESSO À ROM
ROM - MEMÓRIA DE LEITURA EXCLUSIVA
LARAM - LÓGICA DE ACESSO À RAM
RAM - MEMÓRIA DE ESCRITA E LEITURA
LCIO - LÓGICA DE CONTROLE DAS PORTAS DE I/O
I/O - PORTAS DE ENTRADA E SAÍDA (INPUT/OUTPUT)

* ARQUITETURA INTERNA



* REGISTROS PROGRAMÁVEIS

AX	AH	AL	PRIMARY ACCUMULATOR(S)
BX	BH	BL	ACCUMULATOR(S)/BASE REGISTER
CX	CH	CL	ACCUMULATOR(S)/COUNTER REGISTER
DX	DH	DL	ACCUMULATOR(S)/ I-O DATA COUNTER

SP	STACK POINTER
BP	BASE POINTER

SI	SOURCE INDEX
DI	DESTINATION INDEX

PC	PROGRAM COUNTER
----	-----------------

CS	CODE SEGMENT
DS	DATA SEGMENT
SS	STACK SEGMENT
ES	EXTRA SEGMENT

STATUS	STATUS
--------	--------

* REGISTROS PROGRAMÁVEIS (CONT.)

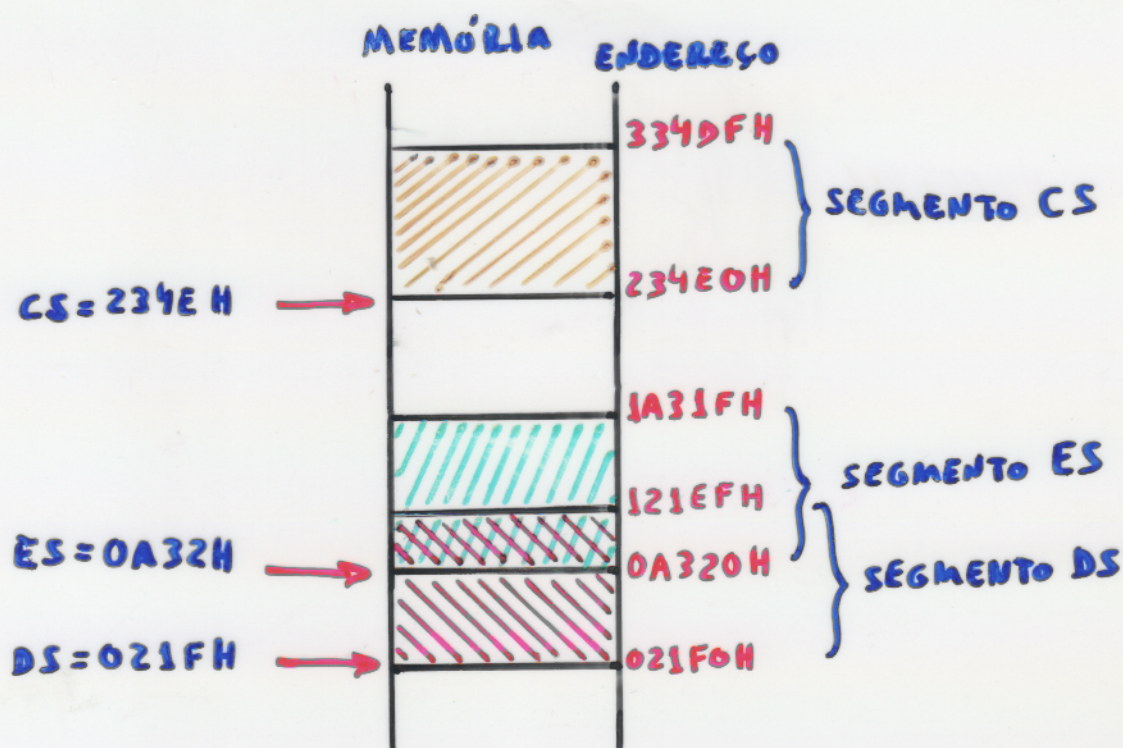
— TODOS OS ENDEREÇOS DE MEMÓRIA NO 8086 SÃO CALCULADOS PELA SOMA DO CONTEÚDO DE UM REGISTRO DE SEGMENTO COM O ENDEREÇO DE MEMÓRIA EFETIVO:

$$\begin{array}{rcl} \text{REGISTRO DE SEGMENTO} & & \text{XXXXXXXXXXXXXXXXXX 0000} \\ \text{ENDEREÇO EFETIVO} & + & \underline{0000\text{YYYYYYYYYYYYYYYY}} \\ \text{ENDEREÇO REAL} & & \text{XXxz2222222222222222Yyyy} \end{array}$$

— 20 BITS PARA ENDEREÇAMENTO $\rightarrow$ 1M bytes PARA ACESSAR

— CADA REGISTRO DE SEGMENTO IDENTIFICA O INÍCIO DE UM SEGMENTO DE 64K bytes DE MEMÓRIA

— CADA SEGMENTO DE MEMÓRIA PODE OU NÃO SE SOBREPOR A OUTRO SEGMENTO



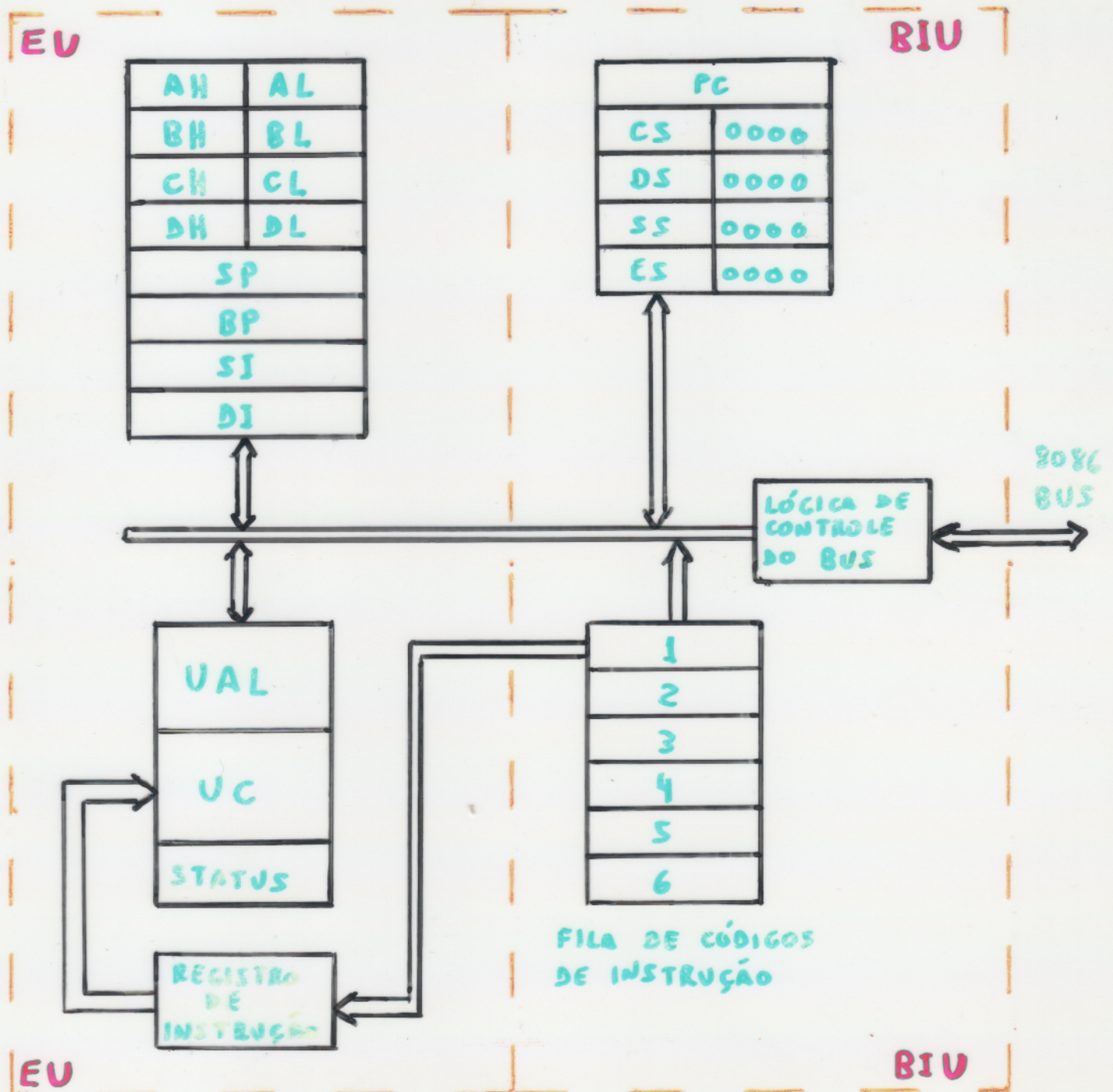
* MODOS DE ENDEREÇAMENTO (CONT)

— RESUMO GERAL

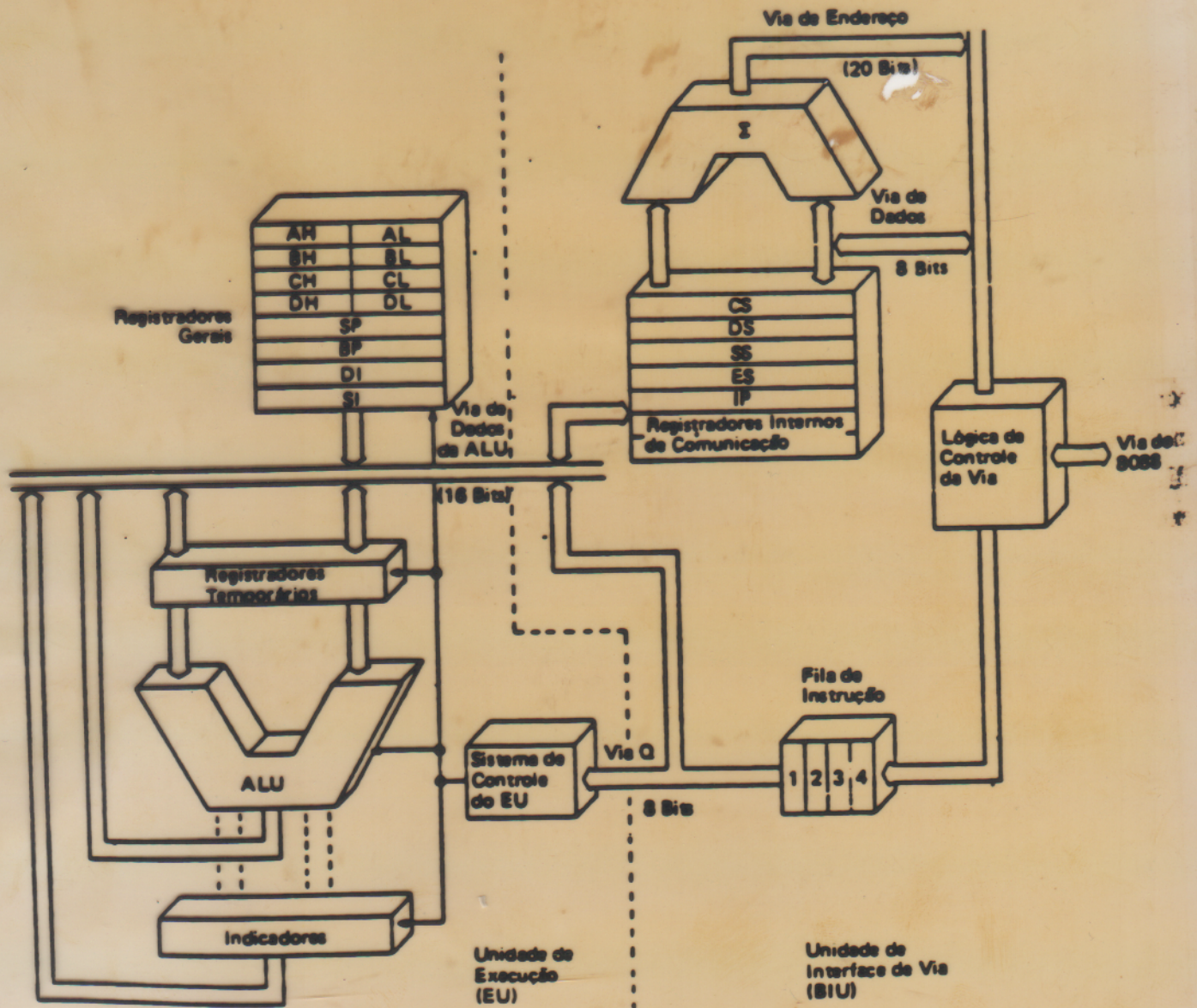
REFERÊNCIA à MEMÓRIA	REGISTRO DE SEGMENTO	REGISTRO BASE	REGISTRO ÍNDICE	DESLOCAMENTO POSSÍVEL		
				16 BITS	8 BITS	NENHUM
DADOS COMUNS	DS	NENHUM	SI	X	X	X
			DI	X	X	X
		BX	SI	X	X	X
			DI	X	X	X
			NENHUM	X	X	X
	DS	NENHUM	NENHUM	X		
	SS	BP	SI	X	X	X
			DI	X	X	X
			NENHUM	X	X	
STACK	SS	SP	NENHUM			
STRING	DS	NENHUM	SI			
	ES	NENHUM	DI			
CÓDIGO	CS	PC	NENHUM			
DESVIO	CS	PC	NENHUM		X	
I/O	DS	DX	NENHUM			

* EXECUTION UNIT (EU) E BUS INTERFACE UNIT (BIU)

- ARQUITETURA



UNIDADE DE EXECUÇÃO e UNIDADE DE INTERFACE DE BARRAMENTO



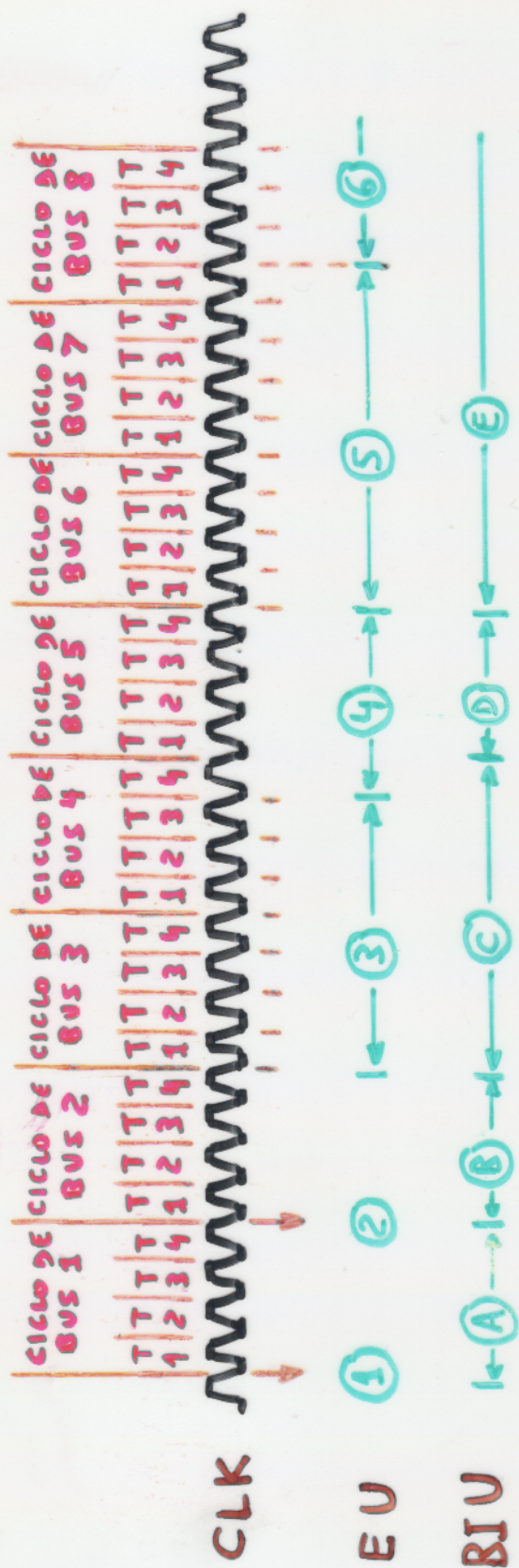
* EU & BIU (CONT.)

- EXECUTION UNIT:
 - REGISTROS DE DADOS E ENDEREÇOS
 - UNIDADE ARITMÉTICA E LÓGICA (UAL)
 - UNIDADE DE CONTROLE (UC)
- BUS INTERFACE UNIT:
 - LÓGICA DE CONTROLE E LIGAÇÃO COM O BARRAMENTO (BUS)
 - REGISTROS DE SEGMENTO
 - LÓGICA DE ENDEREÇAMENTO DA MEMÓRIA
 - FILA DE 6 BYTES PARA CÓDIGOS DE INSTRUÇÃO
- EU E BIU OPERAM ASSINCRONAMENTE:
 - QUANDO EU ESTÁ PRONTO PARA EXECUTAR UMA NOVA INSTRUÇÃO ELA É OBTIDA NO TOPO DA FILA DE INSTRUÇÕES E EXECUTADA EM UM CERTO NÚMERO DE PERÍODOS DE CLOCK
 - SE A FILA ESTIVER VAZIA BIU EXECUTA UM CICLO DE MÁQUINA PARA OBTER A INSTRUÇÃO (FETCH), ENQUANTO EU ESPERA PELO CÓDIGO DA INSTRUÇÃO
 - NORMALMENTE EU NÃO PRECISA ESPERAR PELO "FETCH" DA INSTRUÇÃO, UMA VEZ QUE A FILA RARAMENTE FICA VAZIA
 - SE MEMÓRIA OU I/O DEVEM SER ACESSADOS DURANTE A EXECUÇÃO DE UMA INSTRUÇÃO EU INFORMA SUAS NECESSIDADES AO BIU QUE EXECUTA UM CICLO DE MÁQUINA DE ACESSO EXTERNO

* EU & BIU (CONT.)

- BIU PROCURA MANTER A FILA DE INSTRUÇÕES CHEIA, SE DUAS OU MAIS POSIÇÕES DA FILA ESTIVEREM VAZIAS BIU EXECUTA CICLOS DE MÁQUINA DE "FETCH" PARA COMPLETAR A FILA
 - SE BIU ESTIVER NO MEIO DE UM CICLO DE "FETCH" E EU LHE PEDIR UM CICLO DE ACESSO EXTERNO, O "FETCH" SERÁ CONCLUÍDO ANTES DE ATENDER AO PEDIDO DE EU
 - UM PEDIDO DE EU TEM PRIORIDADE EM RELAÇÃO A UM CICLO DE "FETCH" CASO OCORRAM AO MESMO TEMPO
 - DESTA FORMA O 8086 PRATICAMENTE ELIMINOU O TEMPO DE ESPERA PELO "FETCH" DA INSTRUÇÃO. A ÚNICA EXCESSÃO OCORRE QUANDO É EXECUTADA UMA INSTRUÇÃO DE "DESVIO SOB CONDIÇÃO" PARA UMA INSTRUÇÃO FORA DA SEQUÊNCIA DA FILA
- EU NÃO USA "CICLOS DE MÁQUINA", UMA INSTRUÇÃO É EXECUTADA EM UM NÚMERO DE PERÍODOS DO CLOCK.
 - BIU UTILIZA UM "CICLO DE BUS" PARA FAZER "FETCH" DE INSTRUÇÕES OU ACESSAR MEMÓRIA OU UNIDADES DE I/O, ONDE 1 CICLO DE BUS É IGUAL A 4 PERÍODOS DO CLOCK. AS VEZES "CICLO DE BUS" E "CICLO DE MÁQUINA" SE CONFUNDEM.

* EU & BIU (CONT.)



* EU & BIU (CONT.)

- ① EU PEDÊ UMA INSTRUÇÃO NA FILA. COMO A FILA ESTÁ VAZIA, BIU FAZ "FETCH" DE UMA
- ② BIU FAZ "FETCH" DE UMA INSTRUÇÃO NA MEMÓRIA EM UM CICLO DE BUS
- ③ EU PRECISA DE UM SEGUNDO BYTE DA INSTRUÇÃO
- ④ BIU FAZ "FETCH" DO SEGUNDO BYTE DA INSTRUÇÃO EM UM CICLO DE BUS
- ⑤ EU CALCULA O ENDEREÇO DO DADO NA MEMÓRIA EM SETE PERÍODOS DE CLOCK, E AO FINAL DO SÉTIMO PERÍODO PEDE UM ACESSO EXTERNO (À MEMÓRIA)

* EU & BIU (CONT.)

- ③ Como EU não está pedindo acessos externos (ao BUS), BIU faz "FETCH" dos próximos dois bytes de instrução e os coloca na fila. Ao final do ciclo de BUS 4 o EU está pedindo um acesso ao BUS, logo no próximo ciclo BIU atenderá ao pedido de EU ao invés de fazer "FETCH" de instrução
- ④ EU espera pelo dado que está sendo providenciado pelo BIU
- ⑤ BIU faz "FETCH" do dado na posição de memória dada pelo EU
- ⑥ EU usa nove períodos do clock para executar a operação aritmética requerida

* EU & BIU (CONT.)

- ⑤ BIU CONTINUA FAZENDO "FETCH" DE INSTRUÇÕES PARA ENCHER A FILA DE BYTES DE INSTRUÇÃO
- ⑥ EU LÊ O PRÓXIMO BYTE DE INSTRUÇÃO NA FILA DO BIU E INICIA A EXECUÇÃO DA PRÓXIMA INSTRUÇÃO

* I/O

AS INSTRUÇÕES IN e OUT SÃO REALMENTE OITO INSTRUÇÕES POIS CADA UMA TEM DOIS CONJUNTOS DE OPÇÕES.

CADA INSTRUÇÃO DE I/O PODE ACESSAR 16 OU 8 BITS, E EM CADA CASO PODE TER UM ENDEREÇO CURTO OU LONGO.

ENDEREÇAMENTO CURTO REQUER UMA INSTRUÇÃO DE DOIS BYTES ONDE O SEGUNDO BYTE É O ENDEREÇO A SER ACESSADO (256 PORTAS DE I/O POSSÍVEIS).

ENDEREÇAMENTO LONGO USA INSTRUÇÃO DE APENAS UM BYTE ONDE O ENDEREÇO É DADO NO REGISTRO DX (65536 POSSÍVEIS).

INSTRUÇÕES DE I/O PARA ACESSAR 8 BITS USAM O REGISTRO AL PARA ARMAZENAR O DADO. INSTRUÇÕES PARA ACESSAR 16 BITS USAM O REGISTRO AX.

EXEMPLOS:

IN AL, PORTA - CARREGA UM BYTE DA PORTA "PORTA" EM AL

OUT AL, [DX] - CARREGA UM BYTE DE AL PARA A PORTA CUJO VALOR ESTÁ EM DX.

IN AX, PORTA - [PORTA] → AL e [PORTA+1] → AH

OUT AX, [DX] - AL → [DX] e AH → [DX+1]

* MODOS DE INTERRUPÇÃO

AS INTERRUPÇÕES PODEM SER:

- DO SOFTWARE OU PRODUZIDO PELO PROGRAMA
- DE LÓGICA EXTERNA MASCARÁVEL (PODE SER INIBIDA)
- DE LÓGICA EXTERNA NÃO MASCARÁVEL (NÃO PODE SER INIBIDA POR SOFTWARE)

UMA INTERRUPÇÃO POR SOFTWARE OCORRE QUANDO:

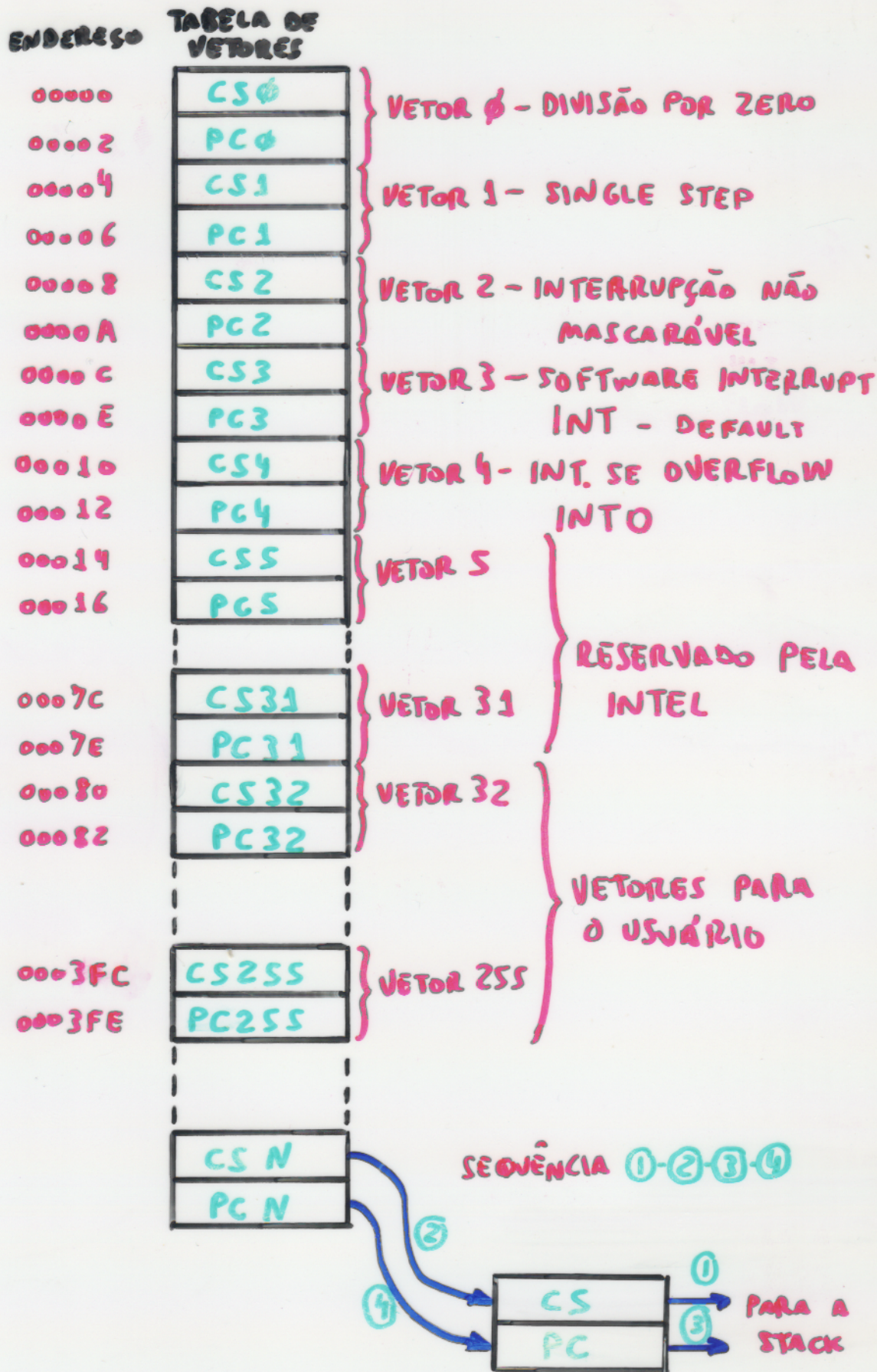
- OCORRE UMA TENTATIVA DE DIVISÃO POR ZERO
- SE EXECUTA A INSTRUÇÃO INTERRUPÇÃO (INT)
- SE EXECUTA A INSTRUÇÃO INTERRUPÇÃO-SE-OVERFLOW (INTO) E O STATUS DE OVERFLOW ESTIVER LIGADO

○ 8086 UTILIZA INTERRUPÇÃO VETORADA, ESTANDO A TABELA DE VETORES NO ENDEREÇO REAL DE MEMÓRIA 00000H ATÉ 003FFH (1024 BYTES DE TAMANHO)

A TABELA DE VETORES É FORMADA DE 256 ENTRADAS, CADA UMA DELAS CONTEM DOIS ENDEREÇOS DE 16 BITS QUE SERÃO CARREGADOS NO REGISTRO DE SEGMENTO CS E NO PC. OS VALORES ANTERIORES DE CS E PC SERÃO SALVOS NA STACK

ALGUMAS ENTRADAS NA TABELA DE VETORES TÊM FUNÇÕES FIXAS, RESTANDO 224 VETORES PARA INTERRUPÇÕES MASCARÁVEIS EXTERNAS

* MODOS DE INTERRUPÇÃO (CONT.)



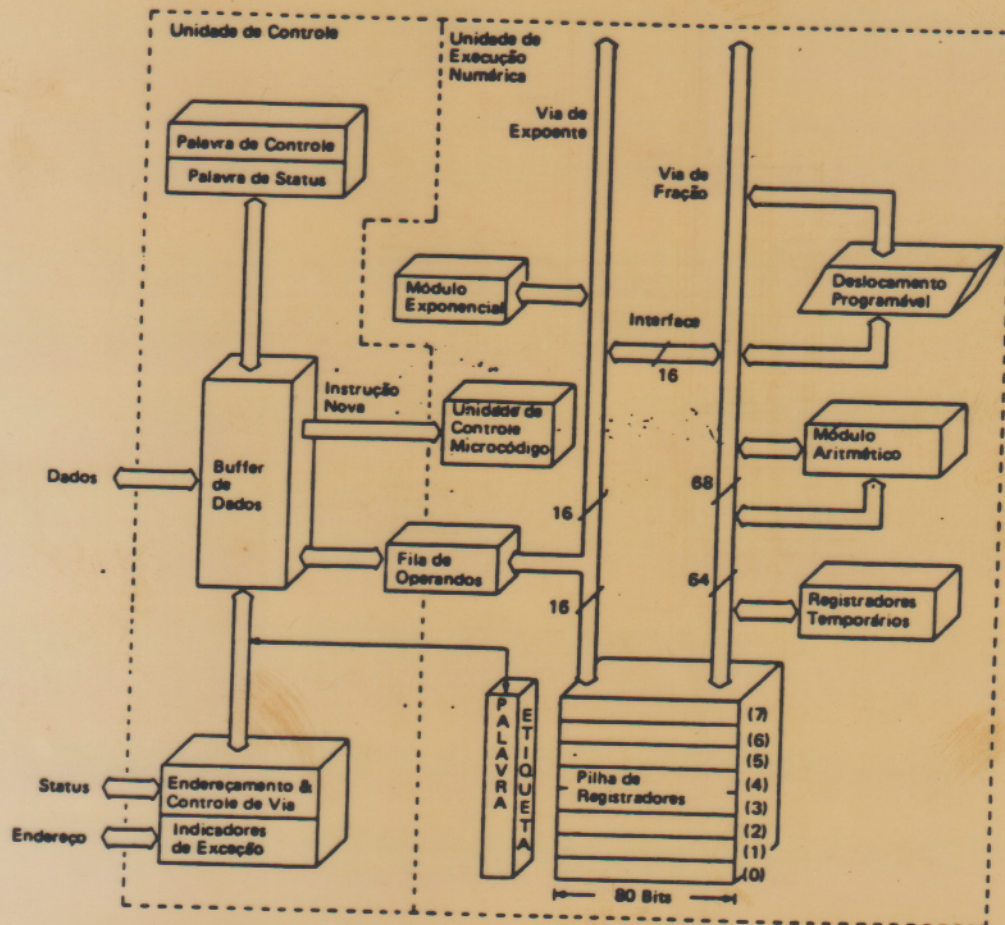


Figura 4.1 O PDN 8087 – pinagem e estrutura interna.

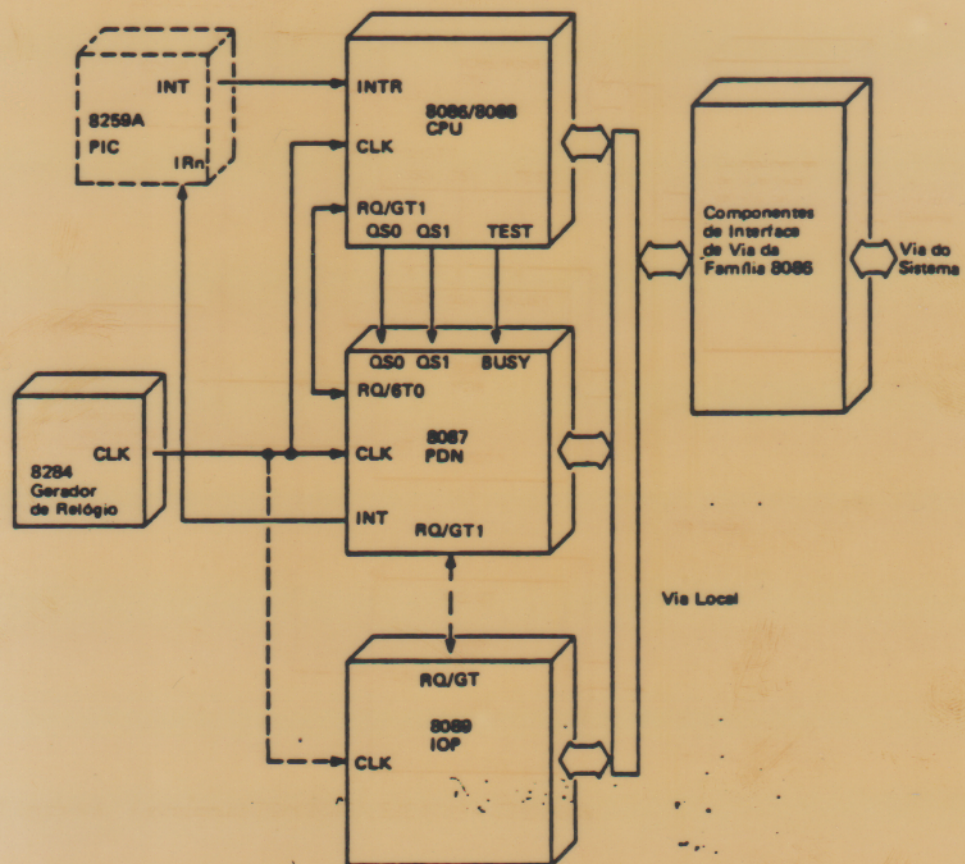
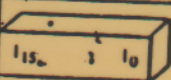
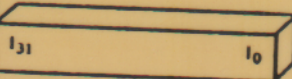
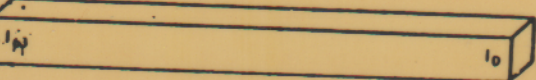
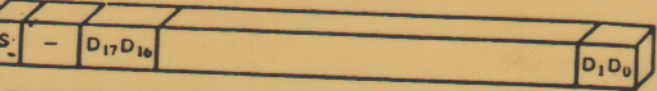
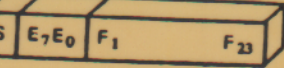
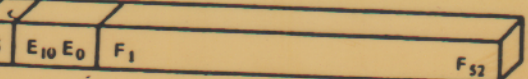
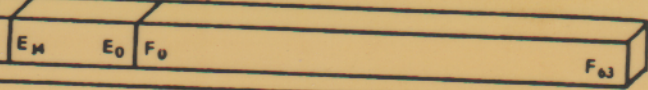


Figura 4.6 Aglomerado PDN 8087 - E/S 8089 - CPU 8086.

Tipos de Dados - 8087

Formatos dos Dados	Faixa	Precisão	Byte Mais Significativo											
			7	07	07	07	07	07	07	07	07	07	07	0
Inteiro Palavra	10^4	16 Bits												
Inteiro Curto	10^9	32 Bits												
Inteiro Longo	10^{19}	64 Bits												
BCD Empacotado	10^{18}	18 Dígitos												
Real Curto	$10^{\pm 38}$	24 Bits												
Real Longo	$10^{\pm 308}$	53 Bits												
Real Temporário	$10^{\pm 4932}$	64 Bits												

Inteiro: I
 BCD Empacotado: $(-1)^S (D_{17} \dots D_0)$
 Real: $(-1)^S (2^{E-Bias}) (F_0 \dots F_{n-1})$
 Polarizador = 127 para Real Curto
 1023 para Real Longo
 16383 para Real Temporário

Figura 4.7 Os tipos de dados do 8087.

* PROCESSADOR 8088

O MICRO 8088 É IGUAL AO 8086 PORÉM COM O BARRAMENTO (BUS) DE DADOS DE 8 BITS, LOGO PARA OBTER DADOS NA MEMÓRIA O 8088 DEVE FAZER DOIS CICLOS DE BUS, ENQUANTO O 8086 SÓ NECESSITA DE UM CICLO.

A ÚNICA DIFERENÇA DE ARQUITETURA É QUE A FILA DE INSTRUÇÕES NO 8088 TEM QUATRO POSIÇÕES, E O BIU EXECUTA UM CICLO DE "FETCH" PARA COMPLETAR A FILA SE APENAS UMA POSIÇÃO DELA ESTIVER VAZIA.

ACESSO À MEMÓRIA E I/O SÃO IDENTICOS. O CONJUNTO DE INSTRUÇÕES É O MESMO PARA 8086 E 8088, COM EXCESSÃO DO TEMPO DE EXECUÇÃO PELA NECESSIDADE DO 8088 FAZER DOIS CICLOS DE BUS PARA ACESSAR DADOS DE 16 BITS NA MEMÓRIA.

* EXEMPLOS DE INSTRUÇÕES

MOV CL, ADDR - CARREGA EM CL UM BYTE ENDETEREÇADO PELA POSIÇÃO "ADDR"
↑
OU OUTROS

MOV BX, ADDR - CARREGA EM BX UMA PALAVRA (16 BITS) ENDETEREÇADOS PELA POSIÇÃO "ADDR"
↑
OU OUTROS

ADD AH, ADDR - $[AH] \leftarrow [ADDR] + [AH]$

SUB ADDR, CX - $[ADDR] \leftarrow [ADDR] - [CX]$

MUL AL, ADDR - $[AX] \leftarrow [AL] * [ADDR]$
16 8 8

MUL DX, ADDR - $[DX][AX] \leftarrow [AX] * [ADDR]$
32 16 16

DIV AX, ADDR - $[AL] \leftarrow [AX] / [ADDR]$ e $[AH] \leftarrow$ RESTO
8 16 8

DIV DX, ADDR - $[AX] \leftarrow [DX][AX] / [ADDR]$ e $[DX] \leftarrow$ RESTO
16 32 16 16

SEG DS - O PRÓXIMO REGISTRO DE SEGMENTO SERÁ DS

MOV ADDR, DATA 16 - $[ADDR] \leftarrow$ DATA 16
16

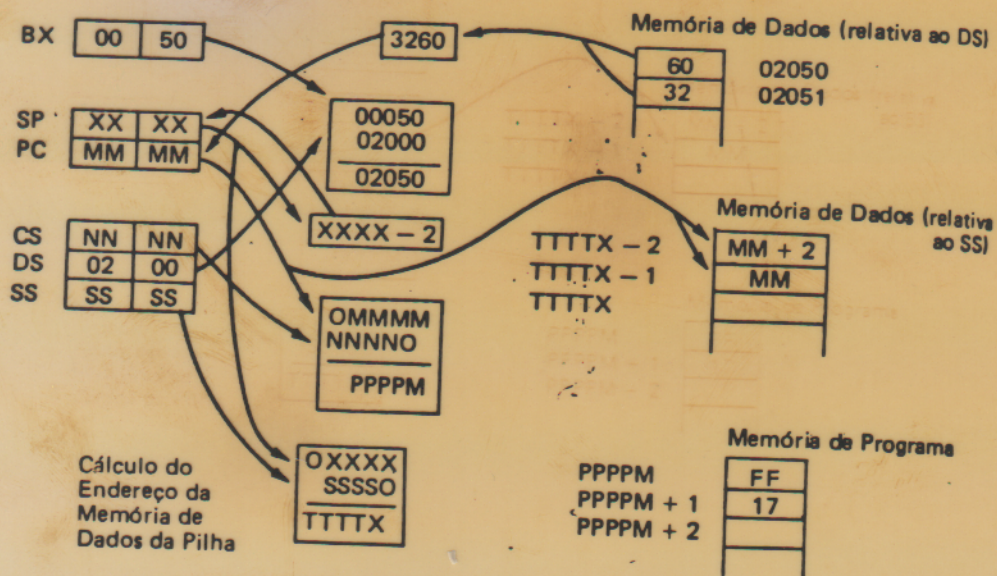
MOV AX, CX - $[AX] \leftarrow [CX]$

XCHG BH, DL - $[BH] \leftrightarrow [DL]$

XCHG DX, ADDR - $[DX] \leftrightarrow [ADDR]$

CALL [BX]

Depois da instrução ser executada, o registrador PC conterá 3260H, que executará a partir dessa posição.



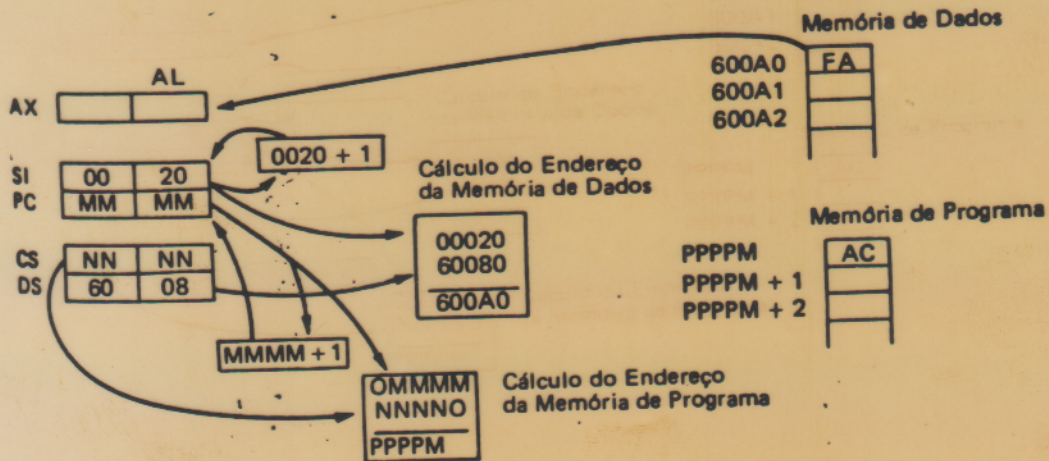
Comentários

- 1) É uma instrução imediata intra-segmentada através da memória.
- 2) O formato: $aa = 00$ e $bbb = 111 = (BX) + DISP$

7.2) LODS BS

Suponha que o registrador SI contém 0020H, o registrador DS contém 6008H, o conteúdo da posição de memória 600A0H é FAH e DF = 0.

Após a execução da instrução, o conteúdo do registrador AL será FAH e o conteúdo do registrador SI será 0021H.

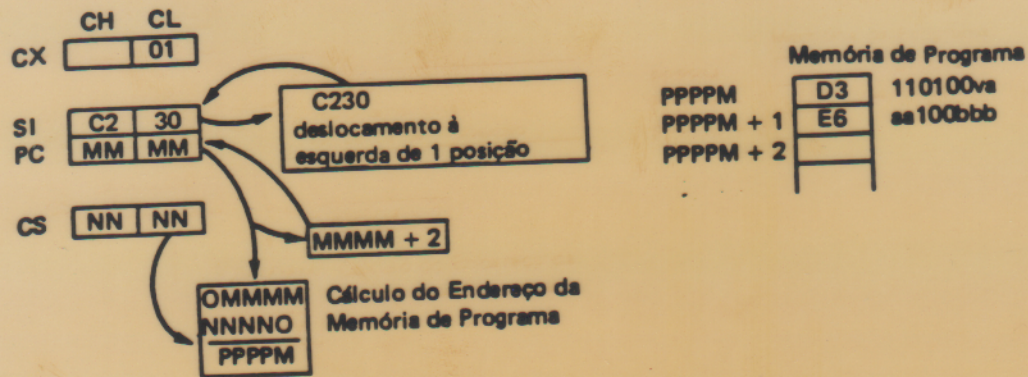


Comentários

1) Esta instrução move o conteúdo da posição de memória endereçada pelo registrador SI para o registrador AL.

2) Como DF = 0 e é uma transferência de 8 bits, o registrador SI é incrementado de 1.

SHL SI, CL



Comentários

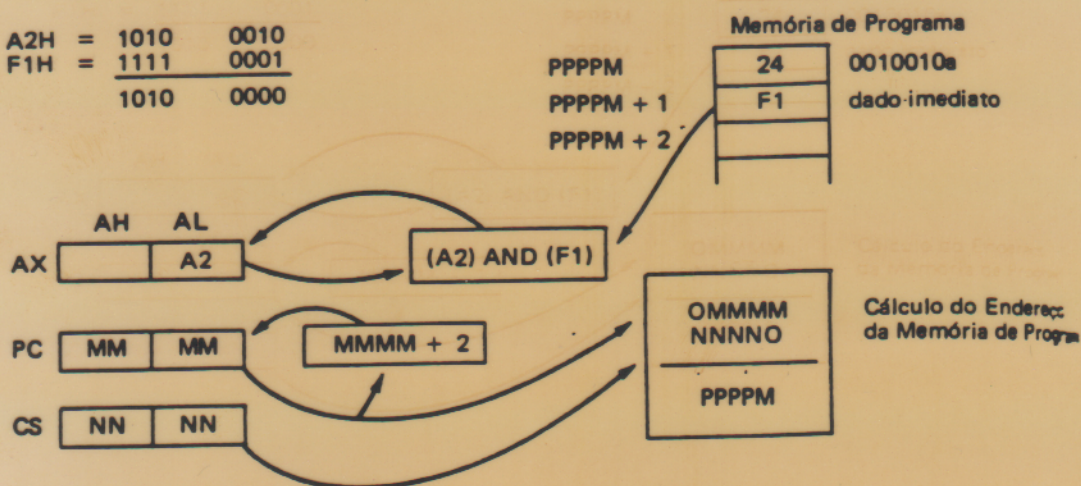
- 1) Desloca SI à esquerda de 1 posição. O deslocamento está contido no registrador CL.
- 2) O formato: $v = 1$ = desloca o número de bits especificados por CL, à esquerda, $a = 1$ = operação de 16 bits, $aa = 11$ = seleção do registrador, e $bbb = 110$ = registrador SI de 16 bits.
- 3) O indicador CF será 0.

3.2) AND AL, F1

Suponha que o registrador AL contém A2H. Depois da instrução ser executada, o conteúdo do registrador AL será A0H.

A2H = 1010 0010
F1H = 1111 0001

1010 0000

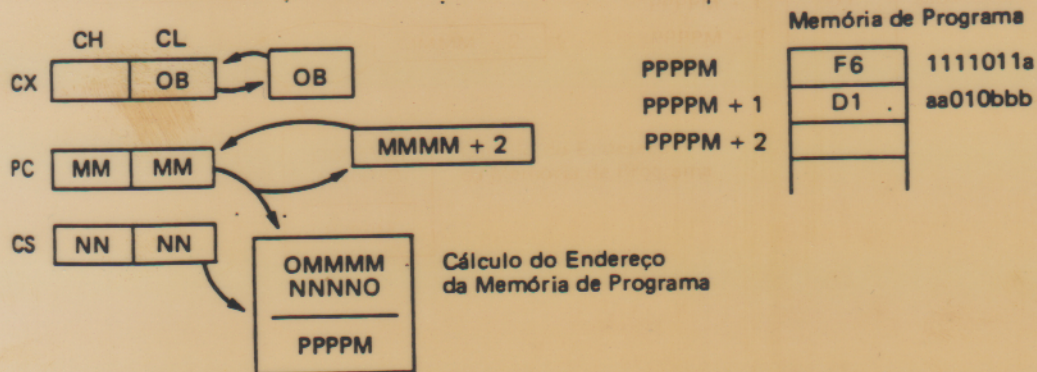


Comentários

- 1) Realiza um AND com o dado imediato F1 presente na posição seguinte ao código de instrução, com o conteúdo do registrador AL.
- 2) O formato: a = 0 = operação de 8 bits e na posição de memória de programa PPPPM + 1, contém os 8 bits do dado imediato.
- 3) Os indicadores ficarão assim:
3 bits "1", leva PF a 0; AF indeterminado; resultado não nulo, leva ZF para 0; OF vai a 0; SF vai a 0 e CF vai a 0.

3.1) NOT CL

Suponha que o registrador CL contém 0BH. Depois da instrução ser executada, o conteúdo do registrador CL será F4H.

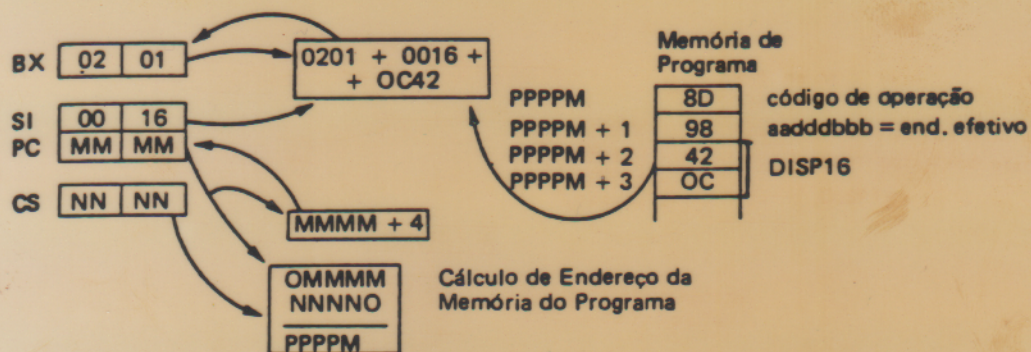


Comentários

- 1) Complementa o conteúdo do registrador CL.
- 2) O formato: a = 0 = operação de 8 bits, aa = 11 = seleção do registrador e bbb = 001 = registrador CL de 8 bits.
- 3) Os indicadores não são afetados.

1.2) LEA BX, (BX + SI + DISP16)

Suponha que o registrador BX contém 0201H, SI contém 016H e o DISP16 contém 0C42H. Após a execução da instrução, o registrador BX contém OE59H, que é a soma dos conteúdos dos registradores BX e SI e o DISP16.

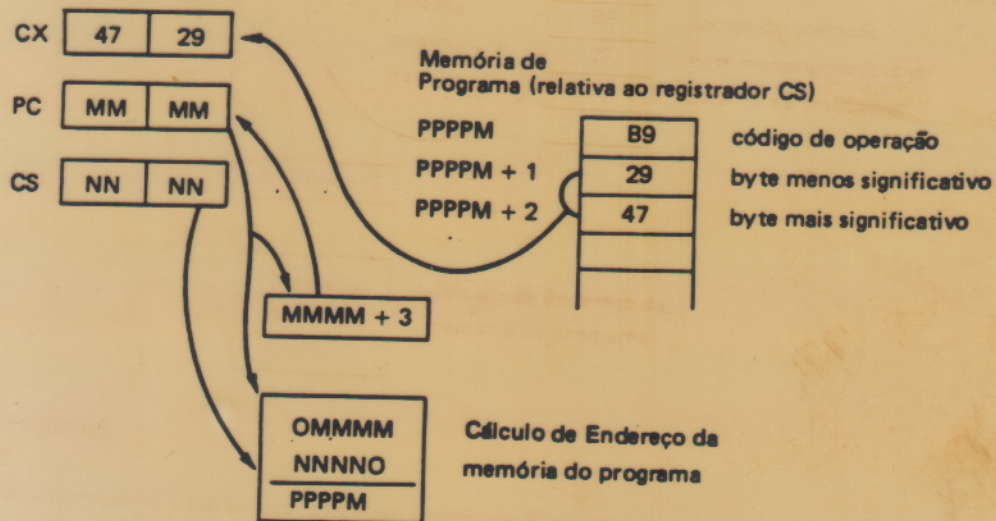


Comentário

- 1) Carrega o endereço de 16 bits (OE59H) no registrador BX.
- 2) O formato: aa = 10 = 2 bytes DISP, ddd = 011 = registrador BX e bbb = 000 = (BX) + (SI) + (DISP16), indicando o endereço efetivo.
- 3) Os indicadores não são afetados.

1.1) MOV CX, 4729 H

move o dado 4729 H de 16 bits para o registrador CX.



Comentários

- 1) Esta instrução é usada para carregar dados de 8 ou 16 bits no registrador de modo imediato.

6.2) SUB AL, 06H

DAS

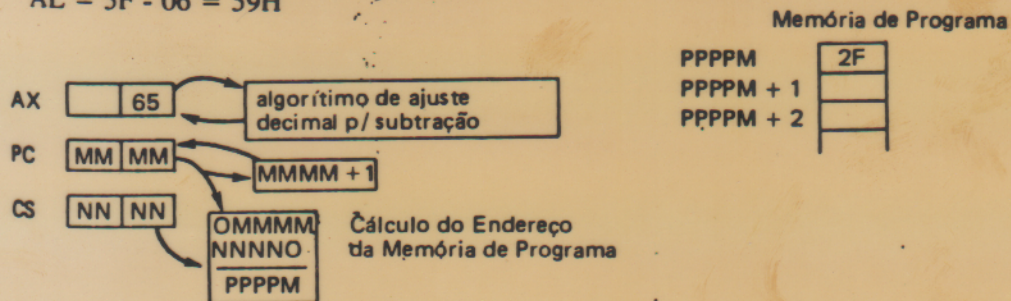
Suponha que o registrador AL contém 65H. Executando a sequência de instruções acima, resultará que AL conterá 59H.

Após a instrução SUB, teremos:

AL = 65H	=	0110	0101	
dado = 06H	=	0000	0110	+
complemento	=	1111	1010	
de dois do dado		0101	1111	
		5	F	

Como os 4 bits menos significativos do registrador, AL é igual a FH, então 6H é subtraído de AL, resultando:

$$AL = 5F - 06 = 59H$$



Comentários

- 1) Esta instrução é usada para ajustar um resultado no registrador AL em um decimal, gerado pela subtração de dois números em BCD.
- 2) O carry é complementado, levando CF a 0 e o indicador AF é levado para 1.

6) Exemplos de Instruções Aritméticas Codificadas em Binário

6.1) ADD AL, 28H

AAA

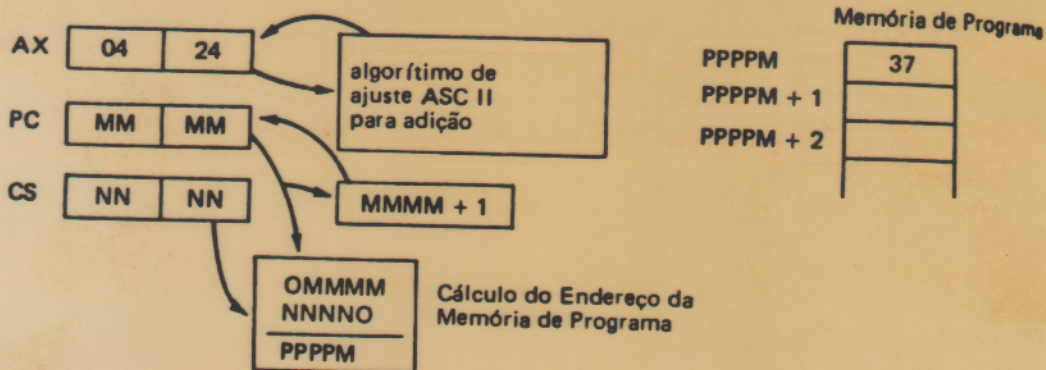
Suponha que o registrador AX contém 0424H. Executando a sequência de instruções acima, resultará que AX conterá 0502H.

Após a instrução ADD, teremos:

$$\begin{array}{rcl}
 \text{AL} & = & 24 \text{ H} = 0010 \quad 0100 \\
 \text{dado} & = & 28 \text{ H} = 0010 \quad 1000 \\
 & & \underline{0100} \quad \underline{1100} \\
 & & 4 \quad \quad \quad \text{C}
 \end{array}$$

O valor 4C é armazenado em AL. Usando o algoritmo para a instrução AAA, resultará:

$\text{AL} = \text{AL} + 6 = 4\text{C} + 6 = 52$, e zera-se os 4 bits mais significativos de AL, então: $\text{AL} = 02$ e AH é incrementado resultando: $\text{AH} = 05$.



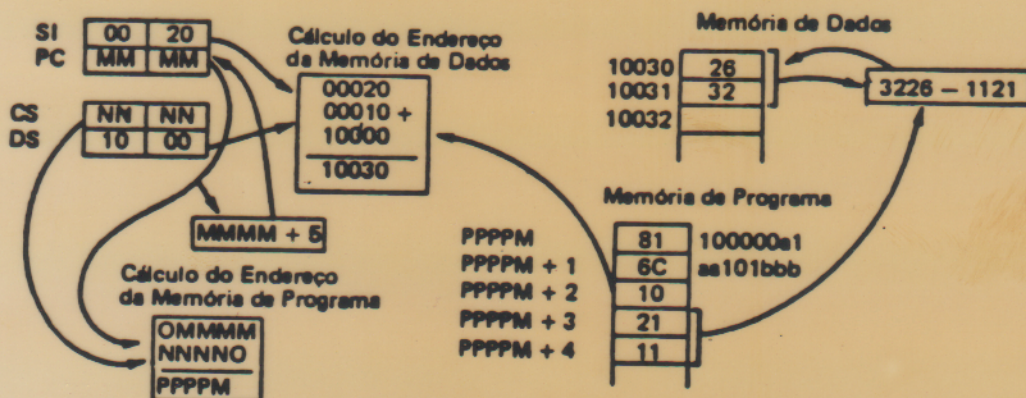
2.2) SUB (SI + 10), 1121H

Suponha que o registrador SI contém 0020H, DS contém 1000H e a posição de memória 10030H contém 3226H. Após a execução da instrução, o conteúdo da posição de memória 10030H conterá 2105H.

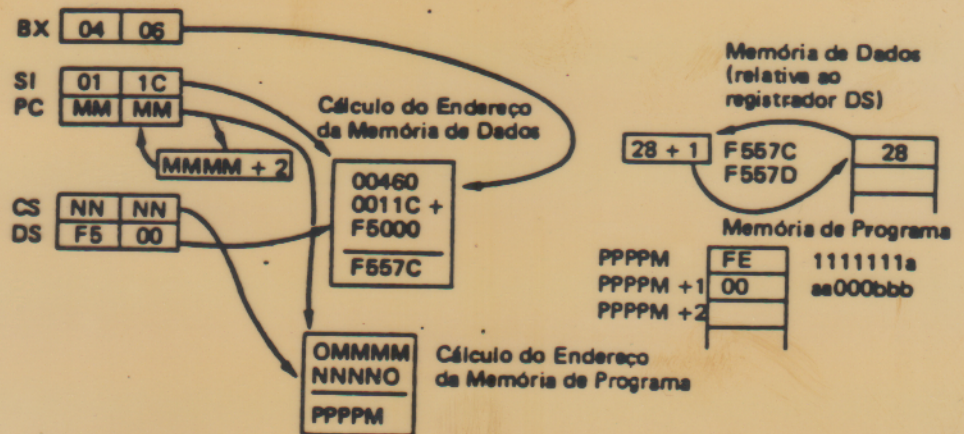
3226H = 0011 0010 0010 0110

Complemento de dois

de 1121H =
$$\begin{array}{r} 1110 \quad 1110 \quad 1101 \quad 1111 \\ 0010 \quad 0001 \quad 0000 \quad 0101 \\ \hline \end{array}$$



INC [BX+SI]



Comentários

- 1) Soma 1 ao conteúdo da posição de memória especificada.
- 2) O formato: $a = 0 = 1$ byte de dados, $aa = 00 =$ nenhum DISP e $bbb = 00 + (SI)$
- 3) O indicador de carry não é afetado.

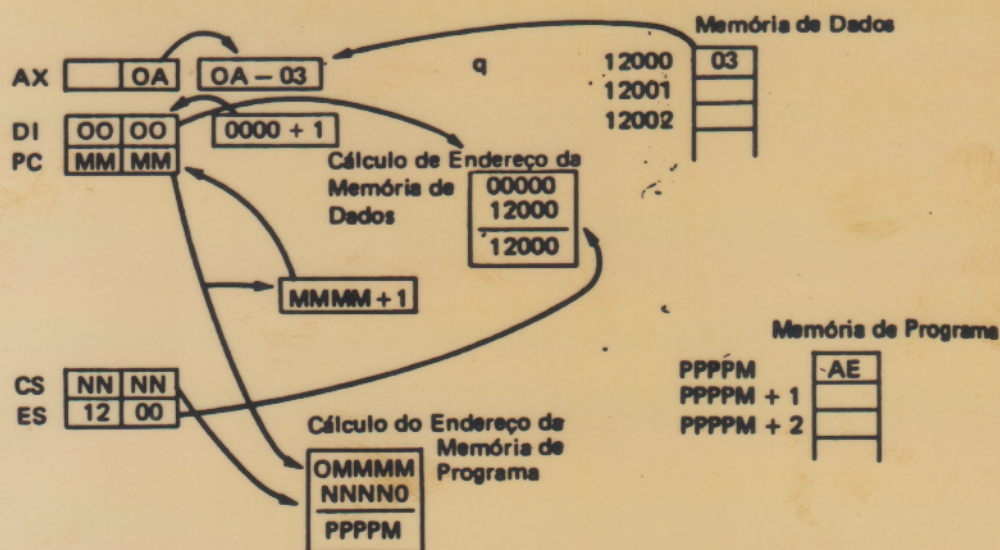
SCAS BD

registor ES contém 1200H, o conteúdo da posição de memória 12000H é 03H
DF = 0.

Após a execução da instrução, o registrador DI é incrementado de 1 e os indicadores estarão da seguinte maneira:

AL = 0AH =	0000	1010
Complemento de 2	1111	1101
03H	0000	0111

3 bits "1", leva PF a 0; não existe carry entre os bits 3 e 4, portanto, AF é levado a 0; OF vai a 0; SF vai a 0; carry do bit mais significativo é complementado, portanto CF é levado a 0 e resultado não nulo, leva ZF a 0.



Comentários

- 1) Esta instrução compara o conteúdo da posição de memória endereçada por DI com

MOV AX, [1000]

deslocamento

ou INDEXADO

MOV AX, [SI]

índice

← desloc. dentro do seg. de texto
← endereço base do seg. de texto

← desloc. dentro do seg de pilha
← endereço base do seg. de pilha