

## 9. MICROCONTROLADORES 8086/8088

### 9.1 Introdução

O 8088 é o microprocessador do PC-XT (*Personal Computer - eXtended Technology*), produzido inicialmente pela Intel, sendo equivalente ao microprocessador 8086 (que tem como principal diferença a extensão do barramento externo de dados). O 8086 tem os barramentos de dados externo e interno de 16 bits enquanto o 8088 apresenta o barramento interno de 16 bits e o externo de 8 bits.

Os processadores 80186 e 80188 são *upgrades* do 8086 e do 8088 respectivamente, onde foram incorporados de 10 a 15 componentes (gerador de clock, controlador de DMA, unidade de chip select, controlador de interrupção, temporizadores, etc.) mais comuns em sistemas baseados no 8088/86. Todas as instruções do 8086 são incluídas no 80186, além de 10 novas instruções.

Ainda, dentro da família dos microprocessadores Intel de 16 bits, temos o processador 80286 (registradores de 16 bits, 24 linhas de endereço). O 80286 incorporou um gerenciador de memória e agregou recursos para multitarefa (há novas instruções com este objetivo). A partir do 80286 os PCs passaram a se chamar PC-AT (PC - *Advanced Technology*).

A diferença no barramento externo de dados entre o 8088 e o 8086 poderia nos levar a supor este último é duas vezes mais rápido que o primeiro; tal fato, em geral, não é verdadeiro. Quando a Unidade de Execução do microprocessador (chamada **EU**: *Execution Unit*) executa uma instrução, outra unidade independente (chamada **BIU**: *Bus Interface Unit*) busca uma nova instrução na memória e a guarda em uma memória interna do microprocessador (*queue*). Com isso, há uma sobreposição de ciclos de execução e ciclos de busca de instrução. No caso do 8088, se a instrução é de 16 bits a BIU deve fazer duas leituras e a EU deverá esperar a instrução estar completa. No 8086 com apenas uma leitura na memória a instrução já estaria disponível para execução.

#### Portanto:

- para aplicações orientadas a 8 bits (byte) o 8088 desenvolve a tarefa tão bem quanto o 8086;
- para aplicações orientadas a 16 bits (word), o 8088 será menos eficiente que o 8086 mas não com metade da velocidade.

#### **Principais Características do 8088**

- Barramento de endereços com 20 linhas: proporciona endereçamento para até 1MB (1.048.576 bytes) de memória.
- Capacidade de uso de coprocessador (8087).
- Todos os registradores internos possuem 16 bits (alguns podem ser acessados pela metade - primeiros ou últimos 8 bits).
- Possui 2 modos de funcionamento: Mínimo e Máximo: no modo mínimo equivale a um 8085 acelerado. O modo máximo é usado em ambientes multiprocessados (quando o 8088 convive com outros 8088) ou em ambientes coprocessados (quando há processador aritmético). Desta forma, como o PC-XT tem soquete para utilização do 8087, sua CPU funciona com o 8088 no modo máximo.
- A CPU é dividida em 2 blocos: Unidade de Execução (**EU**) e Unidade de Interface com o Barramento (**BIU**).
- BIU com fila de instruções (*queue*) de 4 bytes (8088) ou 6 bytes (8086).
- Realiza instruções de divisão e multiplicação (para números com ou sem sinal), assim como operações com strings (blocos de bytes).
- Possui registradores de segmento, possibilitando segmentação de memória.
- No RESET o endereço de memória acessado é FFFF0H.

Para ilustrar melhor as características do 8088, a tabela a seguir foi criada, onde é feita uma comparação entre o 8085, o 8086 e o 8088. A seguir faz-se um paralelo entre os registradores do 8085 e os registradores do 8088 (os quais serão mais detalhadamente explicados nas próximas seções).

## Características dos Microprocessadores 8085, 8086 e 8088

Característica	Microprocessador 8085	Microprocessador 8088	Microprocessador 8086
Barramento de endereço	16 bits	20 bits	20 bits
Capacidade de endereçamento de memória	65.536 ( 64 kB )	1.048.576 ( 1 MB )	1.048.576 ( 1 MB )
Barramento de dados	8 bits	Interno: 16 bits Externo: 8 bits	Interno: 16 bits Externo: 16 bits
Manipulação de STRINGS	NÃO	SIM	SIM
Registradores Internos	8 bits e 16 bits	16 bits	16 bits
Uso de segmentação para endereçamento	NÃO	SIM	SIM
Aritmética Decimal completa	NÃO	SIM	SIM
Etapas de Busca e Execução	Em sequência:  <b>Busca → Executa</b>	Unidades Independentes: Unidade de Interfaceamento com Barramento ( <b>BIU</b> ) – responsável pela <b>Busca</b> e  Unidade de Execução ( <b>EU</b> )	Unidades Independentes: Unidade de Interfaceamento com Barramento ( <b>BIU</b> ) – responsável pela <b>Busca</b> e e Unidade de Execução ( <b>EU</b> )

## Registradores dos microprocessadores 8085, 8086 e 8088

## Registradores do 8085

	A	Acumulador
H	L	Apontador de dados
B	C	
D	E	
SP		Apontador de pilha
PC		Contador de Programa

## Registradores do 8088 / 8086

AH	AL (A)	AX – Acumulador Primário
BH	BL	BX – Acumulador e Registrador Base
CH	CL	CX – Acumulador e Contador
DH	DL	DX – Acumulador e Endereçador de I/O
SP		Apontador de pilha
BP		Apontador base – usado na pilha
SI		Índice da Fonte – usado para indexação
DI		Índice de Destino – usado para indexação
IP		Ponteiro de Instrução
CS		Segmento de Código
DS		Segmento de Dados
SS		Segmento de Pilha
ES		Segmento Extra
FLAGS		Registrador de Flags

Registradores de segmento. São usados para a formação do endereço absoluto.

## 9.2 Diagrama de Blocos do 8088

Uma das principais inovações do 8088 foi a separação entre lógica de execução e lógica de controle do barramento, criando dois blocos que funcionam de forma assíncrona: a Unidade de Execução (EU) e a Unidade de Interface com o Barramento (BIU).

A EU tem como função processar (decodificar e executar) instruções obtidas da BIU. A EU é constituída de:

- Registradores de Dados;
- Registradores de Endereços;
- ALU;
- Unidade de Controle.

A BIU tem apenas funções de hardware: controla o acesso ao barramento (linhas de endereçamento, linhas de dados e sinais de controle). A BIU é constituída de:

- Lógica de interface com o barramento;
- Registradores de segmento;
- Lógica para endereçamento de memória (somador);
- Fila de instruções (4 bytes para o 8088 e 6 bytes para o 8086).

### Procedimento de trabalho do 8088:

1. A BIU coloca o conteúdo do IP (que é somado ao registrador CS) no barramento para efetuar a busca de instrução;
2. O registrador IP é incrementado (aponta para a próxima instrução);
3. A instrução lida é passada para a fila;
4. A EU pega a primeira instrução da fila;
5. Enquanto a EU executa esta instrução a BIU faz uma nova busca de instrução para preencher a fila. Se a instrução a ser executada pela EU for muito demorada a BIU preenche toda a fila.

Há 2 situações em que não são aproveitadas as instruções contidas na fila. São elas:

- Na execução de instruções de desvio. Neste caso a fila é descartada (ou seja, é sobreescrita);
- Quando a instrução faz referência à memória.

### Definições:

- ⇒ **Pipelining**: introdução de paralelismo para executar programas de natureza sequencial. OBS: Esta tecnologia é utilizada no projeto de processadores RISC.
- ⇒ **Periférico**: qualquer equipamento ou dispositivo que provê à CPU comunicação com o resto do sistema.
- ⇒ **Interface**: Conjunto de componentes capaz de controlar um dispositivo (*device*). OBS: uma placa de expansão pode conter uma ou mais interfaces.
- ⇒ **IP** (*Instruction Pointer*): tem a mesma função do PC (*Program Counter*) no 8085.
- ⇒ **Registrador de Segmento**: registrador que armazena o endereço base. Deve ser somado ao endereço de deslocamento (ou *offset*, ou ainda endereço lógico) para obter o endereço físico (ou endereço absoluto). No 8088 os segmentos definem blocos de 64Kbytes de memória. Há 4 registradores de segmento:
  - \* Code Segment (CS): área destinada a código;
  - \* Data Segment (DS): área destinada a dados;
  - \* Extra Segment (ES): área extra destinada a dados;
  - \* Stack Segment (SS): área destinada a armazenar endereços de retorno de rotinas de interrupções e de sub-rotinas;

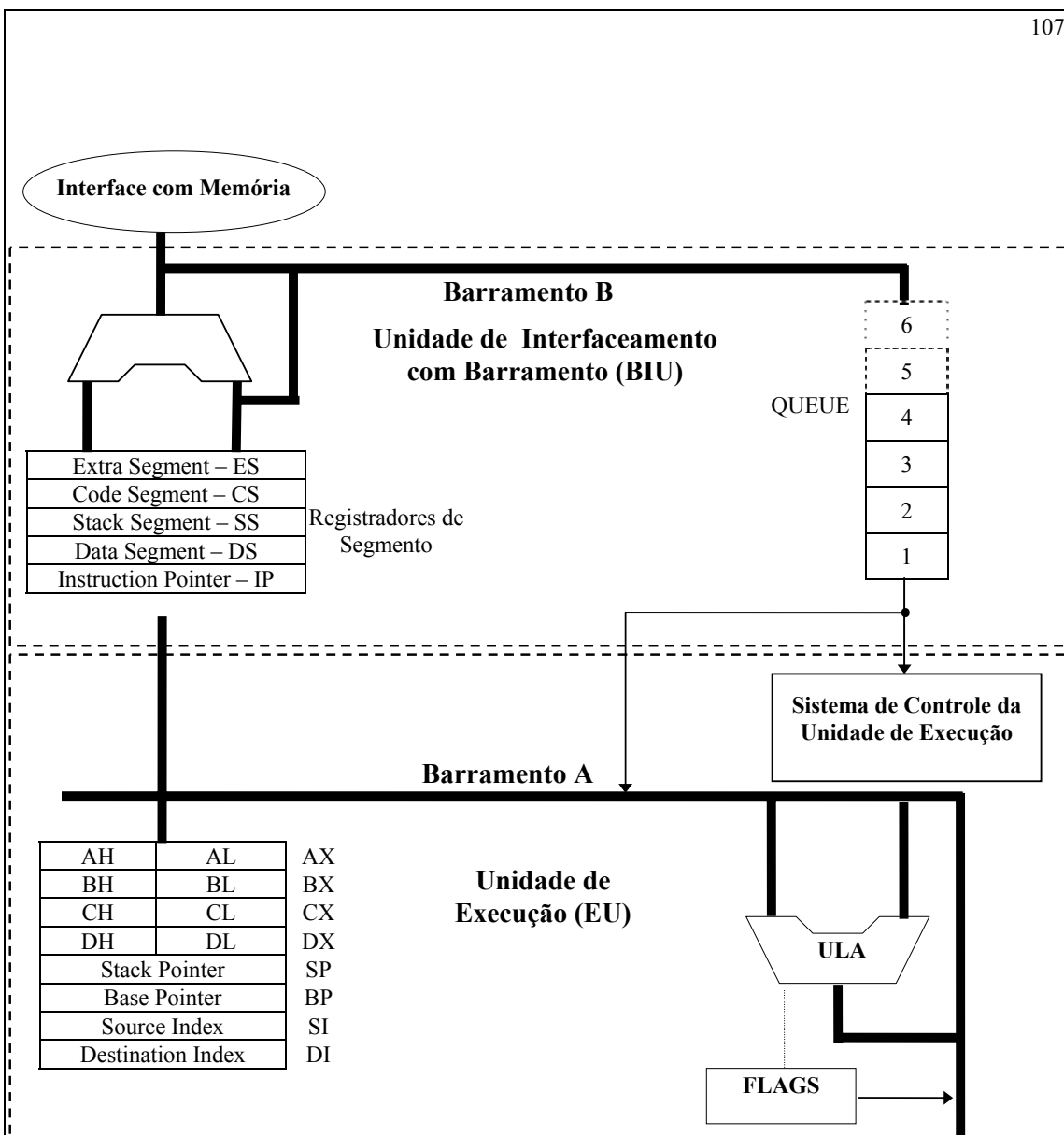


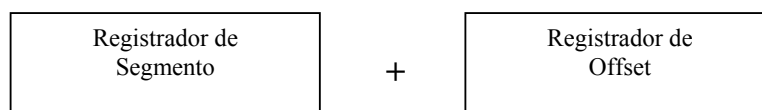
Diagrama de blocos do 8088/8086

### Vantagens da Utilização de Memória Segmentada

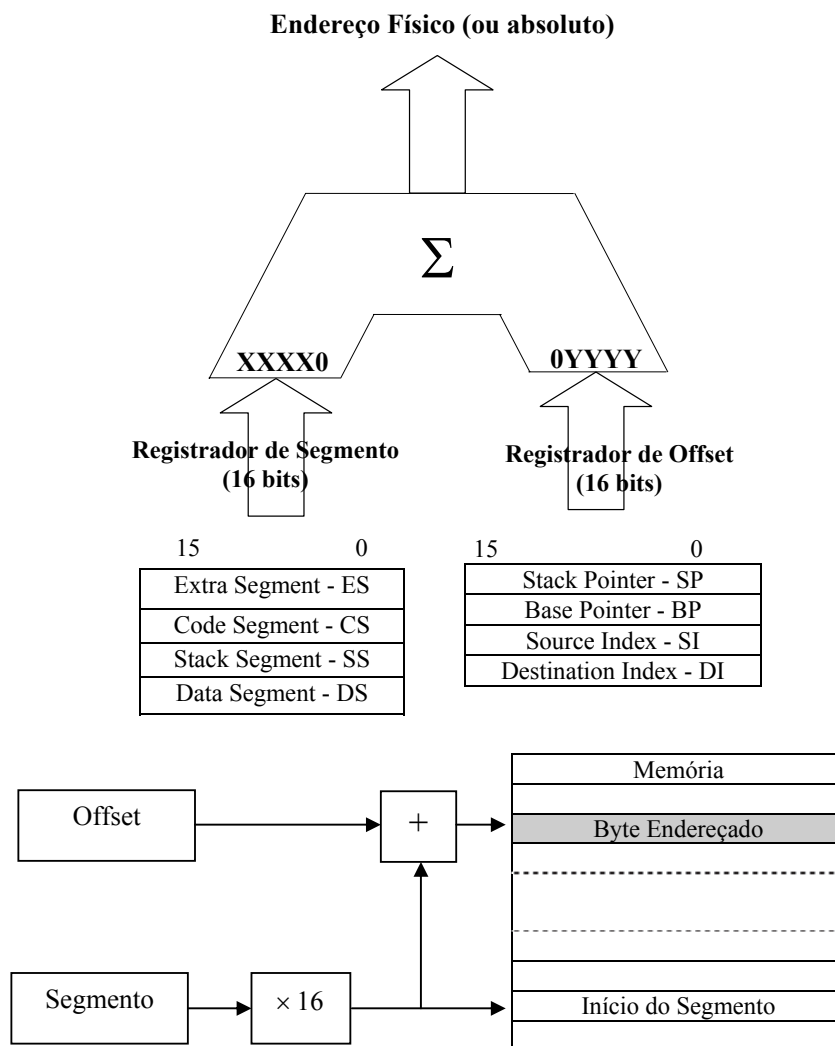
- Por haver uma área específica para armazenamento de código e outras áreas para armazenamento de dados, pode-se trabalhar com tipos diferentes de conjuntos de dados (por exemplo, em um ambiente multitarefa onde um programa atende várias entradas de dados);
- Programas que referenciam endereços lógicos (0000 a FFFF no caso do 8088) podem ser carregados em qualquer espaço (físico) da memória (00000 a FFFFF): possibilita a realocação de programas.

### SEGMENTAÇÃO

Consiste em combinar 2 registradores de 16 bits para gerar um endereço de memória de 20 bits



Endereço Físico = (Conteúdo do Registrador de Segmento)  $\times$  16 + (Conteúdo do Registrador de Offset)



Exemplo: SEGMENTO = 2000H; OFFSET = 2000H

Representação: 2000h:2000h

Endereço Físico = 20000 + 02000h = 22000h

Se o segmento for 4000h, tem-se:

Representação: 4000h:2000h e Endereço Físico = 40000h + 02000h = 42000h

### 9.3 Os Registradores do 8088

O 8088 tem 14 registradores de 16 bits, que podem ser classificados em grupos:

- ⇒ grupo de registradores de dados (4), (*Data register*);
- ⇒ grupo de registradores apontadores (*Pointer*) (2) e índices (*Index*) (2);

- ⇒ grupo de registradores de segmento (4) (*Segment register*);
- ⇒ um registrador apontador de instruções (*Instruction Pointer register*);
- ⇒ um registrador de flags (*Flags register*).

### 9.3.1 Registradores de Dados

Os Registradores de Dados são também chamados Registradores de Propósito Geral. São 4 registradores de 16 bits de uso geral, normalmente utilizados pelo conjunto de instruções para realizar operações lógicas e aritméticas. Podem também ser usados como registradores de 8 bits para instruções envolvendo 1 byte.

Registradores de Dados					
	15	8	7	0	
AX	AH		AL		Accumulator
BX	BH		BL		Base
CX	CH		CL		Counter
DX	DH		DL		Data

**AX** – Acumulador primário – Todas as operações de I/O são realizadas através deste registrador. Operações que utilizam dados imediatos necessitam de menos memória quando feitos através de **AX**. Algumas operações com '*strings*' e instruções aritméticas pedem o uso deste registrador. Geralmente é usado pelos compiladores como hospedeiro para valores retornados de subrotinas.

**BX** – Registrador Base – Parece com registrador **HL** do 8085. É o único registrador de finalidade geral que pode ser utilizado no cálculo de endereço de memória. Todas as referências à memória que usam esse registrador no cálculo do endereço usam o registrador **DS**, como segmento '*default*'.

**CX** – Contador – Parece com o registrador **BC** do 8085. É decrementado durante operações com '*loops*' e '*strings*'. Tipicamente, é usado para controlar o número de repetições do '*loop*'. Também é usado para rotações e deslocamentos de vários bits.

**DX** – Endereçador de I/O e Registrador de Dados – Parece com o registrador **DE** do 8085. Recebe o nome de registrador de dados, principalmente por força dos mnemônicos. Em algumas operações de I/O, fornece o endereço, coisa que nenhum outro registrador pode fazer. Também é usado em operações aritméticas, incluindo multiplicação e divisão, com o resultado a 32 bits. Pode ser usado por compiladores, juntamente com **AX**, para retornar valores de subrotinas.

### 9.3.2 Registradores Apontadores e Índices

Este grupo de 4 registradores é tipicamente usado para gerar endereços de memória efetivo (nome dado à porção offset do endereço físico). Apenas são usados em 16 bits. Podem ainda ser utilizados em operações aritméticas e lógicas para gerar novos endereços efetivos.

<i>Registradores Apontadores e Índices</i>		
	15	0
Stack Pointer	SP	
Base Pointer	BP	
Source Index	SI	
Destination Index	DI	

**SP** – Ponteiro de Pilha – Parece com o registrador **SP** do 8085. Armazena o offset do endereço do topo da pilha. Todas as referências ao **SP**, por definição, usam o registrador **SS**.

**BP** – Ponteiro da Base – Permite acessar dados no segmento da pilha. Tipicamente é usado para acessar parâmetros que foram passados pela pilha.

**SI** e **DI** – Registradores de Indexação – São usados para acessar dados na memória de dados. São extensivamente usados nas operações com '*strings*'. Podem ser usados como operandos em todas as operações lógicas e aritméticas de 16 bits.

No conjunto de instruções do 8088 nem todos os registradores são especificados. Em muitos casos, uma instrução pode usar um só registrador ou conjunto de registradores específicos. Para outras instruções, os registradores têm uso implícito. A tabela a seguir lista as operações que implicam no uso específico (implícito) de um determinado registrador.

Registradores	Operações
AX	multiplicação de word (16 bits), divisão de word, I/O de word
AL	multiplicação de byte, divisão de byte, I/O de byte, aritmética decimal
BX	referência de memória (semelhante ao reg. par HL do 8085)
CX	operação de strings, loops
CL	deslocamento, rotação
DX	multiplicação de word, divisão de word, end. indireto de I/O (0-65535)
SP	operação de stack
SI	operação de string (origem)
DI	operação de string (destino)

### 9.3.3 Registradores de Segmento

#### *Registradores de Segmento*

	15	0
Code Segment	CS	
Data Segment	DS	
Stack Segment	SS	
Extra Segment	ES	

Estes 4 registradores de 16 bits são utilizados para alocar segmentos de 64 kB de memória.

**CS** – Segmento de Código – Para a busca (*'fetch'*) de cada instrução, o offset, definido por **IP**, é adicionado ao endereço base definido por **CS** para o endereço da instrução.

**DS** – Segmento de Dados – Todo acesso a dados usa este registrador como referência, mas existem 3 exceções: (a) endereços para acessos à pilha são calculados usando o registrador de segmento de pilha (**SS**); (b) endereços para acessos a dados que usam o **BP** são calculados usando o **SS** e (c) operações com 'strings', que usam o **DI** no cálculo do endereço, são feitas usando **ES**.

**SS** – Segmento de Pilha – Todos os acessos a dados que usam os registradores **SP** ou **BP** tomam como referência o registrador de segmento de pilha (**SS**).

**ES** – Segmento Extra – Operações com 'strings', que usam **DI** para calcular o endereço, são feitas usando o registrador **ES** para definir o segmento.

A tabela a seguir sintetiza o uso dos registradores de segmento como *'default'*.

#### *Uso dos registradores de segmento*

Tipos de Referência à Memória	Segmento Base (default)	Alternativo	Offset
Instrução de busca	CS	nenhum	IP
Operação de pilha	SS	nenhum	SP
Variável	DS	CS, ES, SS	endereço efetivo
Fonte string	DS	CS, ES, SS	SI
Destino string	ES	nenhum	DI
BP usado como reg. pointer	SS	CS, DS, ES	endereço efetivo

BX usado como reg. pointer	DS	CS, SS, ES	endereço efetivo
----------------------------	----	------------	------------------

O registrador de segmento *default* é selecionado pelo hardware do 8088. Em alguns casos é possível não levar em conta o registrador default e especificar um registrador de segmento diferente (ver exemplo abaixo).

#### **EXEMPLO:**

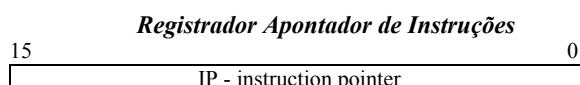
A instrução **MOV [BX],CX** move o conteúdo de CX para a posição de memória calculada utilizando o registrador de segmento **DS** (segmento default para dados) e o registrador de dados **BX**. Mas, se escrevermos **MOV CS:[BX],CX** forçamos o uso do registrador de segmento **CS** para o cálculo da posição de memória destino (que será **CS:BX**).

Da mesma forma, a instrução **MOV AX,[BP]** acessa posição de memória diferente da instrução **MOV AX,ES:[BP]**.

Notar que não há registrador de segmento e registrador de offset alternativos para a busca de instruções (que deve sempre ser o par **CS:IP**) e para operações de pilha (que deve sempre ser o par **SS:SP**).

### 9.3.4 Apontador de Instruções

Este registrador contém o endereço offset da próxima instrução a ser executada pelo microprocessador. Tem a mesma função do PC utilizado no 8085.



### 3.5 - O Registrador de Flags

É um registrador de 16 bits mas apenas 9 bits são usados. Seis deles são bits de status que refletem os resultados de operações aritméticas e lógicas.

Os outros três são bits de controle. O conjunto de instruções do 8088 possui instruções específicas setar e resetar seus flags (que será visto mais adiante).

#### **Registrador de Flags = Registrador de Estado do Programa (PSW)**

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

#### **Flags de Status**

**C** – Flag de carry – reflete o ‘vai um’ do bit mais significativo, nas operações aritméticas (de 8 ou 16 bits). Ele também é modificado por algumas instruções de rotação e deslocamento.

Obs.: As operações de subtração usam aritmética em complemento dois e, por isso, o ‘carry’ é invertido e passa a funcionar como ‘borrow’. Se, após uma operação de subtração, obtém-se  $C = 1$ , isso indica que não houve ‘borrow’, mas  $C=0$ , indica que houve ‘borrow’.

**P** – Flag de Paridade – indica a paridade (par), dos 8 bits menos significativos, do resultado da operação realizada.

$P = 1 \rightarrow$  número par de ‘1’ nos 8 bits menos significativos

$P = 0 \rightarrow$  número ímpar de ‘1’ nos 8 bits menos significativos

**A** – Flag Auxiliar de Carry – reflete o ‘vai um’ do bit 3, em uma operação de 8 bits.

**Z** – Flag de Zero – indica se uma operação teve zero como resultado.

$Z = 1 \rightarrow$  se o resultado da operação for igual a zero



$Z = 0 \rightarrow$  se o resultado da operação for diferente de zero

S – Flag de Sinal – é igual ao bit de mais alta ordem do resultado de uma operação aritmética.

$S = 0 \rightarrow$  resultado positivo

$S = 1 \rightarrow$  resultado negativo

O – Flag de Overflow – seu conteúdo é obtido através de uma operação XOR do ‘carry in’ com o ‘carry out’ do bit de mais alta ordem do resultado de uma operação aritmética. Ele indica um ‘overflow’ de magnitude, em aritmética binária com sinal. Indica que o resultado é muito grande para o campo destino.

### Flags de Controle

T – Flag de Trap (armadilha) – usada para a depuração de programas. Coloca o 8086 no modo passo a passo. Após cada instrução uma interrupção é gerada automaticamente.

I – Flag de Interrupção – habilita ou desabilita a interrupção externa (pedida pelo pino INTR). Ao contrário do 8085, onde as interrupções RST 7.5, RST 6.5 e RST 5.5 podem ser habilitadas/desabilitadas individualmente, no 8086 todas são habilitadas ou desabilitadas ao mesmo tempo. A habilitação/desabilitação individual pode ser feita através do controlador de interrupção 8259.

$I = 1 \rightarrow$  interrupção habilitada

$I = 0 \rightarrow$  interrupção desabilitada

D – Flag de Direção – determina se as operações com ‘strings’ vão incrementar ou decrementar os registradores de indexação (SI e DI).

$D = 1 \rightarrow$  os registradores SI e DI serão decrementados, ou seja, a ‘string’ será acessada a partir do endereço mais alto em direção ao mais baixo.

$D = 0 \rightarrow$  os registradores SI e DI serão incrementados, ou seja, a ‘string’ será acessada a partir do endereço mais baixo em direção ao mais alto.

### EXEMPLO:

Descreva o estado dos flags após a execução da instrução **ADD AL,1** (Adiciona 1 ao conteúdo do registrador de 8 bits **AL**), sabendo que inicialmente  $AL = 7Fh$ .

Resultado: AL terá o valor 80h ( $7Fh + 01h$ ) e seus flags serão:

$C \rightarrow 0$

$P \rightarrow 0$

$A \rightarrow 1$

$Z \rightarrow 0$

$S \rightarrow 1$

$O \rightarrow 1$

## 9.4 A Pinagem do 8088

O microprocessador 8088 é um CI com 40 pinos. Na figura a seguir alguns pinos apresentados tem duas definições. O 8088 tem dois modos de operação que são selecionados pelo pino 33 ( $MN / \overline{MX}$ ). Quando este pino é ligado um nível alto, o  $\mu P$  fica no modo mínimo e é compatível com o 8085, podendo substituir diretamente este  $\mu P$ .

Vcc	A15	A16/ S3	A17/ S4	A18/ S5	A19/ S6	Vcc	MN/ MX	RD	RQ/ GT0	RQ/ GT1	lock	S2	S1	S0	QS0	QS1	Test	RDY	RST
40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Gnd A14 A13 A12 A11 A10 A9 A8 AD7 AD6 AD5 AD4 AD3 AD2 AD1 AD0 NMI INTR CLK GND

No PC, o 8088 é usado no modo máximo ( $MN/\overline{MX}=0$ ). As definições a seguir são relativas a este modo.

- (a) **AD0-AD7 (9 -16):** transmitem os 8 bits menos significativos dos endereços multiplexados com o byte de dados. Os bits de endereço (A0-A7) são apresentados no início do ciclo (T1) e os dados a partir de T2. Ambos sinais são bufferizados para fazerem parte do barramento do sistema.
- (b) **A8-A15 (2-8 e 39):** é a outra parte dos bits de endereços. Não são multiplexados (no 8086 são), permanecendo estáveis durante grande parte do ciclo de acesso a barramento. Após bufferização fazem parte do barramento do sistema.
- (c) **A16/S3 - A19/S6 (35-38):** sinais de status e endereços multiplexados. No início do ciclo de barramento estão presentes os bits dos endereços A16 a A19. Durante o restante do ciclo (a partir de T2) informa o status interno do 8088.
  - S6 = 0 (permanece em nível baixo no modo máximo);
  - S5 indica o status do flag de interrupção (se = 1 a interrupção está habilitada);
  - S3 e S4 são decodificados para indicar qual registrador de segmento está sendo usado no momento (ver tabela abaixo). Não são usados no PC-XT da IBM.

S3	S4	Reg. Segmento
0	0	Dados Extra (ES)
0	1	Pilha (SS)
1	0	Código (CS)
1	1	Dados (DS)

- (d) **Clock (19):** sinal de entrada responsável pela geração da temporização no  $\mu P$ . No projeto do PC este sinal é gerado pelo CI 8284A, tendo frequência de 4,77Mhz e ciclo de trabalho (*duty cycle*) de 33%.
- (e)  **$\overline{RQ}/\overline{GT_0}$  (Request / Grant) (31):** sinal bidirecional, usado por outros controladores de barramento local para requerer o uso do barramento (faz o papel dos pinos HOLD e HLDA do 8085). No PC este sinal é ligado ao co-processador 8087.
- (f)  **$\overline{RQ}/\overline{GT_1}$  (30):** idem ao anterior, com grau de prioridade menor. No PC não é usado.
- (g) **Lock (29):** é ativado pela instrução LOCK e permanece ativo até o fim da próxima instrução. É usado para indicar a outros controladores de barramento que estes não devem tentar obter o controle do barramento durante a execução da instrução seguinte à LOCK. O PC não é projetado para multi-controladores e portanto, este pino não é usado.
- (h) **NMI (17):** sinal de entrada, usado para gerar uma interrupção não mascarável no  $\mu P$ . No PC este sinal é mascarado externamente por uma lógica programável (um FlipFlop). É equivalente ao pino TRAP do 8085.
- (i) **INTR (18):** entrada para solicitação de interrupção mascarável. No PC este pino é ligado ao CI 8259A que expande esta entrada de 1 para 8, adicionando, dentre outras funções, níveis de prioridade.
- (j) **Ready (22):** sinal de entrada usado para inserir estados de espera no ciclo de barramento, estendendo-os. No PC este sinal vem do chip do clock 8284A que o sincroniza com o clock do sistema.
- (l) **Reset (21):** usado para reiniciar o  $\mu P$ . No PC este sinal vem do chip de clock 8284A que recebe uma entrada da fonte do sistema. A fonte envia um sinal chamado "Power Good" indicando que os níveis de tensão estão apropriados e o sinal Reset pode ser então removido para iniciar o  $\mu P$ . A situação dos registradores do 8088 após o RESET é a seguinte:

- Reg. IP = 0000 e reg. CS = FFFF resultando no end. de início de programa = FFFF0h;
- Registradores DS, SS, ES e IP zerados;
- Registrador Flags zerado.

(m) **QSO, QS1 (24, 25) - Queue Status:** sinais de saída que fornecem o status da fila de instrução (queue) do 8088. No PC são ligados ao co-processador de modo que este possa acompanhar o status da fila 8088.

(n) **TEST (23):** pino de entrada, utilizado em conjunto com a instrução WAIT (não confundir com estado de espera, "wait state"). Se após a instrução WAIT o pino  $\overline{TEST}$  estiver em nível alto, o 8088 espera em estado inativo até este pino obter sinal baixo. No PC é ligado a um pino de saída do 8087.

(o) **S0, S1, S2 (26 - 28) (status):** apresentam informações de status sobre o tipo de ciclo de barramento que está sendo desenvolvido. Este status é válido no início de cada ciclo de barramento (durante o período T1). No PC estes pinos são ligados ao 8288, chip controlador de barramento, onde são decodificados. Os sinais de saída decodificados no 8288 formam as linhas de controle do sistema. Os sinais de controle gerados pelo 8288 a partir das informações de status são:  $\overline{IOR}$ ,  $\overline{IOW}$ ,  $\overline{MemR}$ ,  $\overline{MemW}$  e ALE. Os bits de status podem ser vistos na tabela abaixo.

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Tipo de ciclo no BUS
0	0	0	Reconhecimento de interrupção ( $\overline{INTA}$ )
0	0	1	Leitura I/O ( $\overline{IOR}$ )
0	1	0	Escrita I/O ( $\overline{IOW}$ )
0	1	1	Halt
1	0	0	Busca de instrução
1	0	1	Leitura memória ( $\overline{MemR}$ )
1	1	0	Escrita memória ( $\overline{MemW}$ )
1	1	1	Passivo

## 9.5 Ciclos de Barramento do 8086

O 8086 COMUNICA-SE COM AMBIENTE EXTERNO ATRAVÉS DE UM SISTEMA DE BARRAMENTOS. ASSIM, EXISTEM CICLOS DE BARRAMENTO PARA BUSCAR ("FETCH") INSTRUÇÕES E PARA TRANSFERIR DADOS (ESCREVER OU LER). AS FIGURAS A SEGUIR (ZELENOVSKY, PÁG. 22) MOSTRAM OS CICLOS DE LEITURA E ESCRITA, RESPECTIVAMENTE.

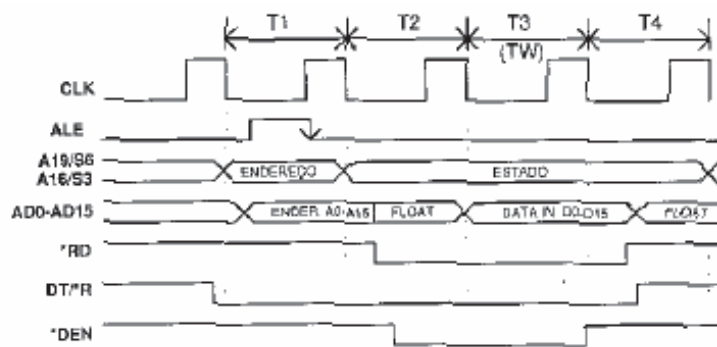


Figura 2.4. Ciclo de barramento para a leitura de dados ou instrução.

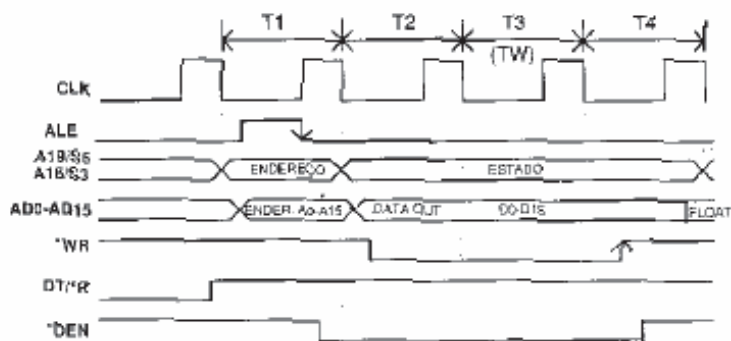


Figura 2.5. Ciclo de barramento para a escrita de dados.

T1 a T4 → períodos de relógio

ALE – Address Latch Enable – O endereço só é reconhecido depois que este sinal vai para nível alto, mas o endereço só é efetivamente usado quando este sinal volta para o nível lógico baixo. Durante o ciclo T2 os sinais de endereço são removidos.

AD0 – AD15 → Barramento de dados. No ciclo de leitura esse barramento é colocado em estado de alta impedância no estado T2, em preparação para a leitura de dados. No ciclo de escrita a CPU coloca os dados no barramento no estado T2.

\*RD → Ativo baixo. Indica leitura de dados da memória ou de um dispositivo de entrada e saída. Ativado em T2.

\*WR → Ativo baixo. Indica escrita de dados na memória ou em um dispositivo de entrada e saída. Ativado em T2.

DT/\*R → Indica a direção da transferência de dados.

\*DEN → Ativo baixo. Data Enable. Indica que vai haver tráfego de dados no barramento.

A19/S6, A16/S3 → Barramento de dados durante ciclo T1. Barramento de sinais de estado de T2 a T4.

A17/S4	A16/S3	Significado
0	0	Segmento Extra
0	1	Segmento de Pilha
1	0	Segmento de Código (ou nenhum)
1	1	Segmento de Dados
S5 = IF	Estado de habilitação da interrupção	
S6 = 0	CPU 8086 tem controle do barramento	

TW → Wait State – Tempo de espera. No período de leitura a CPU lê o dado do barramento no período T3, após o dispositivo lido entregar esses dados. Se o dispositivo não conseguir entregar os dados até o final do ciclo T3, o pino READY do 8086 é colocado em nível lógico baixo, fazendo com que a CPU fique estado de espera, repetindo do estado T3 quantas vezes forem necessárias, enquanto aguarda o dispositivo entregar os dados. No período de escrita pode acontecer o mesmo. A CPU está pronta para entregar os dados, mas o dispositivo não está pronto para recebê-los. O pino READY é colocado em nível baixo, gerando períodos de espera.

## 9.6 Endereçamento de Memória

Já foi explicado numa seção anterior como é feita a composição do endereço físico do 8088. Uma das características desse microprocessador é a sua capacidade de endereçar mais que 64 kbytes de memória. O 8088 possui 20 bits de endereços, permitindo ter uma memória física de 1.048.576 bytes (ou 1MB).

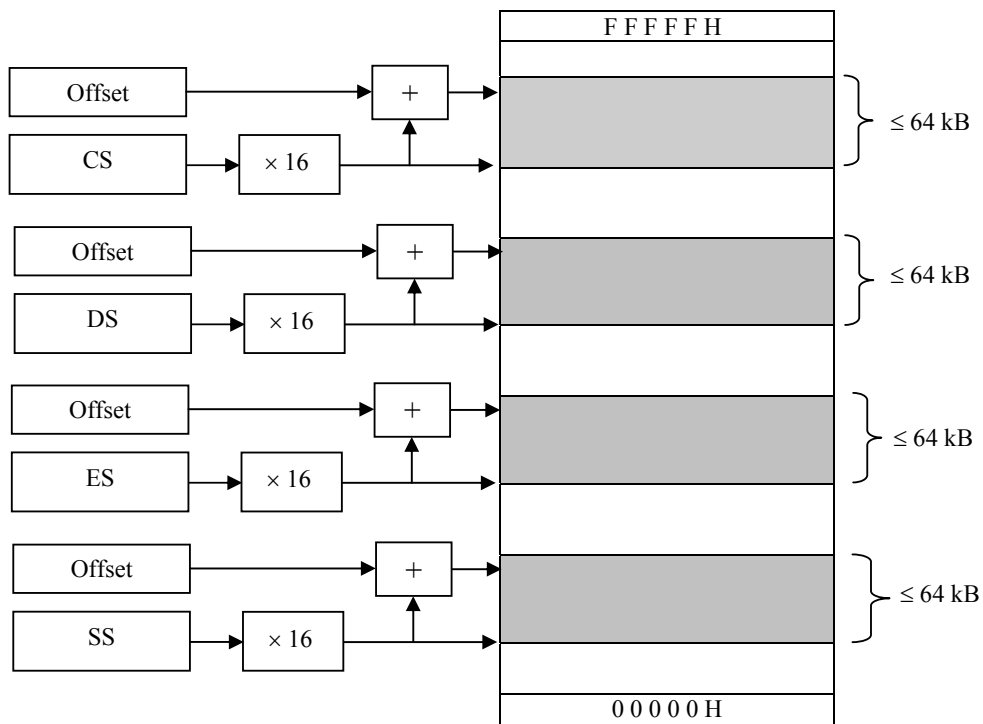
A geração de 20 bits, já explicada, é feita manipulando o conteúdo de registradores especiais chamados *Registradores de Segmento*. O valor carregado no registrador de segmento é usado para localizar, no espaço de 1MB, a região de 64kB onde as instruções do 8088 irão operar. Um outro registrador (que pode ser um reg. apontador, um reg. índice ou o BX) pode especificar 64kB dentro de um segmento. O endereço de memória físico será formado pelo deslocamento do conteúdo do registrador de segmento de 4 bits para a esquerda (acrescentado-se 4 bits 0 à direita), somado a outro valor de 16 bits (acrescentado-se 4 bits 0 à esquerda), resultando num endereço físico com 20 bits. É interessante observar que diferentes combinações de endereços lógicos (offsets) e registradores de segmento podem oferecer o mesmo endereço físico, como exemplificado a seguir:

**EXEMPLO:** supondo que **SS=8F00h**, **SP=21F1h**, **CS=9020h**, **IP=0FF1h**, os endereços físicos para um acesso à pilha e para a busca da próxima instrução são dados por:

SS = 8F00h	→	8F000	
SP = 21F1h	→	+ 021F1	
<b>endereço físico</b>	→	911F1h	←
CS = 9020h	→	90200	
IP = 0FF1h	→	+ 00FF1	
<b>endereço físico</b>	→	911F1h	←

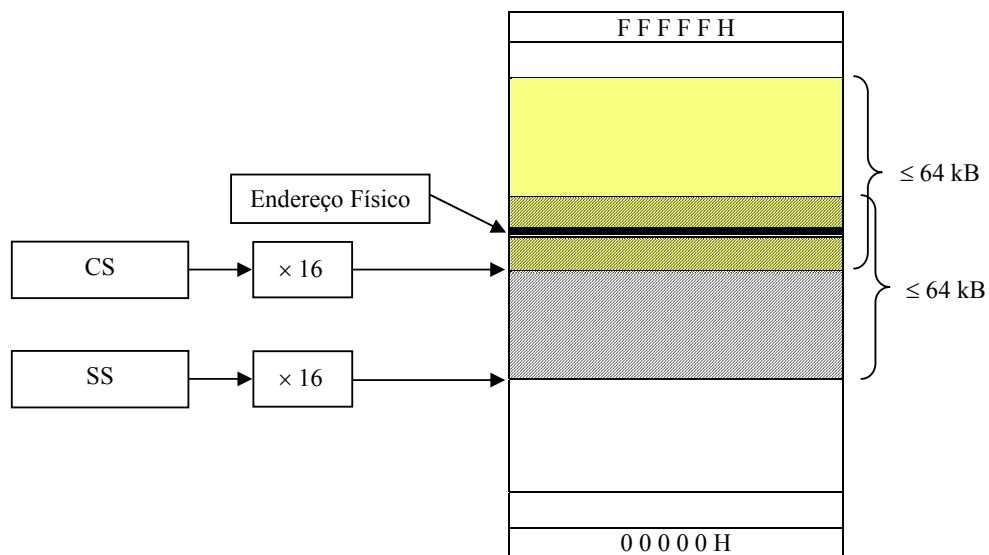
Na realidade os quatro registradores de segmento no 8088 (CS, DS, SS e ES) podem assumir o mesmo valor, ou valores próximos, ou seja, podem se sobrepor parcial ou totalmente. O programador deve tomar cuidado para que isto apenas ocorra se for estritamente necessário.

Estes 4 registradores podem apontar para qualquer região de 64 kB, no espaço de 1 MB. Uma vez setados, haverá uma região de 64kB para o código (CS), 64kB para dados (DS), 64kB para pilha (SS) e 64kB para dados extra (ES). Em qualquer momento que o programa necessite manipular a memória física, fora das atuais regiões de 64 kB, ele deve manipular o registrador de segmento apropriado (alterando-o através de instruções de transferência de dados. As figuras a seguir ilustram as regiões definidas pelos registradores de segmento sem e com sobreposição.



### MULTIPLICIDADE DE ENDEREÇOS

Um mesmo endereço pode ser acessado usando diferentes registradores de segmentos e diferentes registradores de offset



Exemplo: Endereço Físico = 10020 h

Possibilidades de pares SEGMENTO:OFFSET

$$1000h:0020h = 1000h * 10h + 20h$$

$$1001h:0010h = 1001h * 10h + 10h$$

$$1002h:0000h = 1002h * 10h + 00h$$

Um segmento tem até 64 kBytes

Como o segmento é multiplicado por 16 na formação do endereço



O espaço mínimo entre dois segmentos consecutivos é 16 bytes



No intervalo de 64 kBytes há  $63536 \div 16 = 4096$  possibilidades diferentes de endereçar a mesma posição física de memória.

## 9.7 Linguagem de Programação ASSEMBLY do 8086

Conceito de tipo → Cada símbolo (nome de variável, endereço, constante) tem um determinado tipo.

Tipos da linguagem assembly ASM-86:

**BYTE PTR** → referencia uma variável de 1 byte

**WORD PTR** → referencia uma variável de 1 word

**DWORD PTR** → referencia uma variável de 2 words

**NEAR PTR** → referencia o endereço de destino de uma instrução de desvio do tipo *near*

**FAR PTR** → referencia o endereço de destino de uma instrução de desvio do tipo *far*

**NUMBER** → constante de 16 bits

Obs.: *near* → altera somente IP (intra segmento)

*far* → altera IP e CS (inter segmento)

Exemplos:

**MOV AX, [BX]** → copia o word endereçado por **BX** para **AX** → o tipo do símbolo usado já está implícito (uma vez que **AX** é de 16 bits), não havendo necessidade de informar ao assembly.

**INC [BX]** → é necessário informar se deve ser incrementado o **BYTE** de offset **BX** ou o **WORD** de offset **BX**:

**INC BYTE PTR [BX]** → incrementa o byte cujo offset é o valor contido em **BX**

**INC WORD PTR [BX]** → incrementa word que se encontra nos endereços **BX** e **BX+1**.

### FORMATO DAS INSTRUÇÕES DO 8086

Instruções compactas → otimização do uso da memória e da velocidade de leitura das instruções



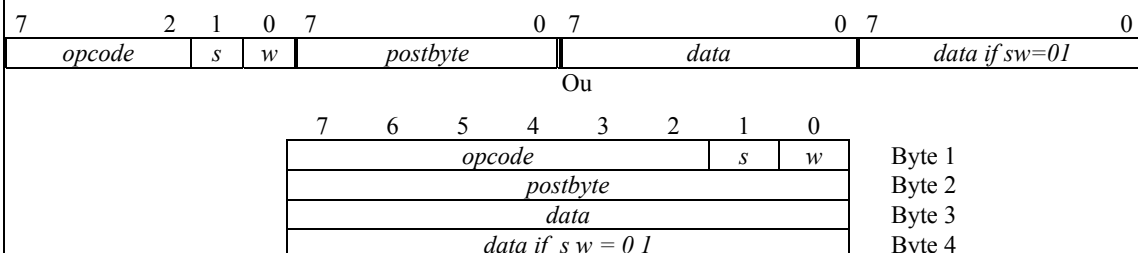
codificação mais complexa do que a do 8085

instruções com comprimento de 1 a 6 bytes

instruções não precisam ocupar exatamente 1 byte

bits restantes podem ser usados

Formato das instruções de transferência de dados:



*opcode* → código da operação

$w \rightarrow$  se  $w = 0 \Rightarrow$  instrução manipula byte; se  $w = 1 \Rightarrow$  instrução manipula word

*postbyte*:

7	6	5	4	3	2	1	0
mod		reg			r/m		

Se  $mod = 11 \Rightarrow$  o operando da instrução é um registrador, que é identificado em “r/m”

Se a instrução envolver dois registradores o campo “reg” identifica o segundo registrador.

## 9.8 Modos de Endereçamento e Endereços de Memória Efetivo

Como observado anteriormente, um endereço físico (20 bits) é constituído de duas partes:

- um segmento ou valor base;
- um offset ou endereço efetivo.

Um Endereço Efetivo (**EA**: *Effective Address*) é a parte offset do endereço físico. Ele é formado pela soma de um deslocamento (um número) com um registrador base (**BX** ou **BP**) e com um registrador índice (**SI** ou **DI**) (exemplo: **MOV AX, [BP+SI+4]**). A referência de memória com omissão de qualquer uma destas partes é também chamada de Endereço Efetivo (por exemplo, a parte offset do exemplo anterior poderia ser **[BP+4]** ou **[BP+SI]**).

Uma das vantagens de se escrever instruções com endereços efetivos é que um endereço de memória pode ser modificado baseado em condições do programa. Isto é particularmente útil quando se trabalha com tabelas (ou matrizes). Por exemplo, **BP** pode ser setado para apontar para a base (o início) da tabela e **SI** (que armazena o resultado de um cálculo) localiza um elemento na tabela.

A tabela abaixo sumariza os modos de endereçamento do 8088 e os registradores que estão disponíveis em cada modo.

Modo Endereçamento	Exemplo Mnemônico	Registrador de Segmento acessado	Operação
Imediato	MOV AX, 1000H	Code $\rightarrow$ CS	AX $\leftarrow$ 1000H
Registrador	MOV DX, CX	-	DX $\leftarrow$ CX
Direto	MOV AH, [1000H]	Data $\rightarrow$ DS	AH $\leftarrow$ [1000H]
Indireto por Registro (BX, BP, SI, DI)	MOV AX, [SI]	Data $\rightarrow$ DS	AL $\leftarrow$ [SI]; AH $\leftarrow$ [SI+1]
Indexado (SI ou DI)	MOV AX, [SI+6]	Data $\rightarrow$ DS	AL $\leftarrow$ [SI+6]; AH $\leftarrow$ [SI+7]
Baseado (BP ou BX)	MOV AX, [BP+2]	Stack $\rightarrow$ SS	AL $\leftarrow$ [BP+2]; AH $\leftarrow$ [BP+3]
Baseado e Indexado	MOV AX, [BX+SI]	Data $\rightarrow$ DS	AL $\leftarrow$ [BX+SI]; AH $\leftarrow$ [BX+SI+1]
Baseado e Indexado com deslocamento	MOV AX, [BX+SI+5]	Stack $\rightarrow$ SS	AL $\leftarrow$ [BX+SI+5]; AH $\leftarrow$ [BX+SI+6]
Strings	MOVS	Extra, Dado $\rightarrow$ ES, DS	[ES:DI] $\leftarrow$ [DS:SI] se DF = 0 então: SI $\leftarrow$ SI+1 e DI $\leftarrow$ DI+1 se DF = 1 então: SI $\leftarrow$ SI-1 e DI $\leftarrow$ DI-1

### ENDEREÇAMENTO POR REGISTRADOR

Operando está em um registrador (de 16 bits AX a DI ou de 8 bits AH a DL)

Ex.: MOV AX, BX  $\rightarrow$  copia conteúdo de BX para AX

CMP AL, DL  $\rightarrow$  compara conteúdos de AL e DL (e seta flags)

### ENDEREÇAMENTO IMEDIATO



Operando faz parte da própria instrução. Usado para atribuir valores de 8 ou 16 bits a registradores

Ex.: MOV AX, 1000H → faz AX = 1000H

CMP SI, 0000H → compara conteúdo de SI com 0000H (e seta flags)

Obs.: Modo imediato não pode ser usado para os registradores de segmento, nem para as instruções PUSH

### ENDEREÇAMENTO ABSOLUTO OU DIRETO

O operando está na memória. A instrução contém o offset do endereço do operando (o segmento é DS).

Ex.: MOV AX, [1000H] → copia para AX o conteúdo localizado nos offsets 1000H e 1001 H, ou seja, AX = 1234 H.

Endereço	Dado
1000H	34H
1001H	12H
1002H	25H

### ENDEREÇAMENTO INDIRETO

O operando está na memória. A instrução, ao invés de conter o offset do endereço, como no caso direto, contém um registrador, que contém o offset do endereço do operando. Os registradores de offset podem ser BX, BP, DI ou SI (o segmento é DS).

Ex.: MOV AX, [BX] → copia para AX o conteúdo localizado no offset dado em BX. Se BX = 1000h, então AX = 1234h

Endereço	Dado
1000H	34H
1001H	12H
1002H	25H

Exemplo de aplicação: Tabelas, onde o endereço de leitura da tabela é dado através do registrador de offset, que é incrementado para varredura da tabela.

### ENDEREÇAMENTO INDEXADO

Uma constante, denominada base, e um registrador de índice são usados. A base e o índice são somados para a obtenção do offset do endereço. Os registradores de índice podem ser BX, BP, DI ou SI

Ex.: MOV AX, 0100H[BX] → copia para AX o conteúdo localizado no offset dado por BX + 0100H. Se BX = 1000h, então BX + 0100H = 1100H e AX = 1234h

Endereço	Dado
1100H	34H
1101H	12H
1102H	25H

Exemplo de aplicação: Tabelas, onde a posição inicial da tabela pode ser dada através da base; o índice pode então ser variado para varredura da tabela.

## ENDEREÇAMENTO BASEADO

É o contrário do endereçamento indexado, ou seja, o conteúdo de um registrador é copiado para uma posição variável indicada por um registrador e uma constante. Registradores usados: BX, BP, DI ou SI

Ex.: MOV [BX + 0100H], AX → copia conteúdo de AX para o endereço BX + 0100H. Se BX = 1000H, então BX + 0100H = 1100H. Se AX = 1234H, então o valor 34H será armazenado em 1100H e o valor 12H será armazenado em 1101H.

Endereço	Dado
1100H	34H
1101H	12H
1102H	xxH

Exemplo de aplicação: Tabelas, onde a posição inicial da tabela pode ser dada através da base; o índice pode então ser variado para varredura da tabela.

## ENDEREÇAMENTO RELATIVO

As instruções de desvio e chamadas de subrotina do tipo near ou short somam ou subtraem um valor de deslocamento para o desvio, ao invés de usar o endereço longo.

Offset de destino = offset da instrução seguinte + deslocamento

## 9.9 Interrupções do 8086

### 9.9.1 Estrutura de interrupção do 8086:

Enquanto o 8085 possui apenas 5 canais de interrupção, o 8086 é capaz de tratar de 256 interrupções diferentes, numeradas de 00h a FFh (ou 0 a 255 decimal). Ao contrário do 8085, a localização de cada interrupção não vem pré-definida no microprocessador; existe uma tabela, mostrada a seguir, com os vetores de interrupção, onde o endereço físico de cada interrupção é colocado.

Número da Interrup.	Endereço do vetor de interrupção	Endereço do tratador da interrupção	Endereço efetivo do vetor de interrupção	Endereço do tratador da interrupção
0	0000h:0000h	IP low h	0000h:0000h	CS:IP
	0000h:0001h	IP high h		
	0000h:0002h	CS low h		
	0000h:0003h	CS high h		
1	0000h:0004h	IP low h	0000h:0004h	CS:IP
	0000h:0005h	IP high h		
	0000h:0006h	CS low h		
	0000h:0007h	CS high h		
2	0000h:0008h	IP low h	0000h:0008h	CS:IP
	0000h:0009h	IP high h		
	0000h:000Ah	CS low h		
	0000h:000Bh	CS high h		
<i>i</i>	0000h:4 <i>i</i>	IP low h	0000h:4 <i>i</i>	CS:IP
	0000h:(4 <i>i</i> +1)	IP high h		
	0000h:(4 <i>i</i> +2)	CS low h		
	0000h:(4 <i>i</i> +3)	CS high h		
255	0000h:03FCh	IP low h	0000h:03FCh	CS:IP

	0000h:03FDh	IP high h		
	0000h:03FEh	CS low h		
	0000h:03FFh	CS high h		

Obs.: O endereço efetivo da interrupção **INT  $n$**  é: **0000h:4n**

### 9.9.2 Interrupções Pré-definidas, reservada ou exceções

O 8086 possui três tipos de interrupção: Interrupções Pré-definidas, reservadas ou exceções, Interrupções por Hardware e Interrupções por Software.

As interrupções de 0 a 31 (ou 00h a 1Fh) são reservadas. Elas têm finalidades específicas para o 8086. Dentre as reservadas 5 interrupções já são pré-definidas no 8086:

Número da Interrup.	Endereço do vetor	Finalidade
00 H	0000h:0000h	Divisão por zero. É chamada sempre que o divisor de uma operação de divisão for zero.
01 H	0000h:0004h	Execução passo a passo. É chamada sempre que a Flag de Trap (T) estiver setada. Facilita a depuração de um programa.
02 H	0000h:0008h	NMI = Interrupção Não-Mascarável. Equivale à TRAP do 8085. Pino 17 do 8086.
03 H	0000h:000Ch	Breakpoints. Permite a execução de programas para depuração até um endereço especificado pelo programador.
04 H	0000h:0010h	Overflow. É chamada sempre que a Flag de Overflow estiver setada, indicando que o conteúdo do resultado não cabe no registrador de destino.
05 H – 1F H	0000H:0014H até 0000H:007C H	Interrupções que não são pré-definidas, mas são reservadas para o 8086. As diversas versões de sistemas usando o 8086 usam essas interrupções para finalidades específicas, mas não necessariamente idênticas entre os sistemas.

### 9.9.3 Interrupções por Hardware

São interrupções solicitadas através do pino INTR (pino 18). Nesse pino é conectado um controlador de interrupção que amplia para 8 os pedidos de interrupção. São interrupções mascaráveis através do bit IF do registrador de flags. A sequência de atendimento de uma interrupção por hardware é:

- 8086 envia primeiro pulso \*INTA para periférico, após receber pedido de interrupção. Após esse sinal reconhecimento de pedido de interrupção o periférico prepara-se para enviar o número da interrupção
- 8086 envia segundo pulso \*INTA para periférico, tendo como resposta do periférico o número da interrupção colocada no barramento de dados
- 8086 salva PSW na pilha
- Apaga flag TF (Trap – passo a passo) e, se for reservada, também IF (Interrupt Flag – desabilita interrupções por INTR)
- Salva CS (segmento) e IP (offset) na pilha
- Carrega novo IP a partir do endereço  $4*nn$
- Carrega novo CS a partir do endereço  $4*nn + 2$

### 9.9.4 Interrupções por Software

São interrupções solicitadas através da instrução **INT  $nn$** , onde  **$nn$**  deve estar entre 32 e 255 (as interrupções de 0 a 31 são reservadas). As interrupções geradas pela instrução **INT  $nn$**  não são mascaráveis. A sequência de atendimento de uma interrupção por software e também de uma interrupção reservada é:

- Salva PSW na pilha

- Apaga flag TF (Trap – passo a passo) e, se for reservada, também IF (Interrupt Flag – desabilita interrupções por INTR)
- Salva CS (segmento) e IP (offset) na pilha
- Carrega novo IP a partir do endereço  $4*nn$
- Carrega novo CS a partir do endereço  $4*nn + 2$

Embora existam 256 interrupções por software, cada uma delas pode ter até 256 sub-funções diferentes, ou seja, através das interrupções por software, pode-se executar  $256*256 = 65.536$  funções diferentes. Normalmente o registrador **AH** é usado para passar para a interrupção qual a função a ser executada. A tabela a seguir exemplifica algumas dessas funções.

Interrupção	Finalidade “macro”	Subfunção	Registradores	Descrição da subfunção
INT 10 H	Serviços de vídeo (É uma função do BIOS)	00 H	AH = 00 H	Seta o modo vídeo
			AL = modo	AL = 03 $\Rightarrow$ VGA 80 x 25
		01 H	AH = 01 H	Seta tamanho do cursor
			CH: bit 7 = 0	
			Bits 6 e 5	00 $\rightarrow$ normal; 01 $\rightarrow$ invisível
			Bits 4 a 0	Linha mais alta do caracter para o cursor
			CL: bits 4 a 0	Linha mais baixa do caracter para o cursor
		02 H	AH = 02 H	Seta posição do cursor
			BH = 0 a 3	Número da página de vídeo
			DH = linha	00 H $\rightarrow$ linha superior
			DL = coluna	00 H $\rightarrow$ coluna da esquerda
INT 21 H	Serviços Gerais (É uma função do DOS)	01 H	AH = 01 H	Lê caractere da entrada padrão, com eco.
			AL	Retorna o código ASCII do caractere lido
		02 H	AH = 02 H	Escreve caractere na saída padrão
			DL	Contém o código ASCII do caractere
		08 H	AH = 08 H	Lê caractere da entrada padrão, sem eco
			AL	Retorna o código ASCII do caractere lido
		09 H	AH = 09 H	Escreve na saída padrão textos terminados em “\$”
			DS:DX	Endereço de início do texto
		25 H	AH = 25 H	Seta vetor de interrupção
			AL	Número do vetor de interrupção a ser modificado
			DS:DX	Novo valor para o vetor a ser modificado
		31 H	AH = 31 H	Encerra programa e deixa residente (TSR)
			AL	Código de retorno
			DX	Número de parágrafos que devem ficar residentes
		4C H	AH = 4C H	Encerra programa
			AL	Código de retorno

## 9.10 Conjunto de Instruções do 8088/86

Há mais de 3000 opcodes distintos se considerados os vários modos de endereçamento. Abaixo estão listadas algumas instruções relativas a cada grupo funcional.

### 9.10.1 Instruções de Transferência de Dados

- movimentação de dados entre dois registradores,
- entre um registrador e posição de memória,
- entre registradores e portas [I/O].

**FORMATO:** *destino, fonte*

Formas genéricas possíveis:

MOV REG, memória	Copia conteúdo da posição de memória para o Registrador indicado
MOV memória, REG	Copia conteúdo do registrador indicado para a posição de memória
MOV REG, REG	Copia conteúdo de um registrador para outro
MOV memória, imediato	Carrega posição de memória com valor indicado
MOV REG, immediate	Carrega registrador com valor indicado

Formas genéricas possíveis para os registradores de segmento:

MOV SREG, memória	Copia conteúdo da posição de memória para o Registrador de segmento indicado
MOV memória, SREG	Copia conteúdo do registrador de segmento indicado para a posição de memória
MOV SREG, REG	Copia conteúdo de um registrador comum para um de segmento
MOV REG, SREG	Copia conteúdo de um registrador de segmento para um comum

Podem ser movimentados byte ou word, sendo que os registradores Ponteiros (BP e SP), Índices (SI e DI) e de Segmento (CS, DS, SS, ES) apenas podem ser acessados como word (16 bits).

**DEFINIÇÃO DE POSIÇÃO DE MEMÓRIA:** Ao escrever programas é conveniente dar nomes a posições de memória: para isto utiliza-se pseudo-instruções.

MEMBY DB ? ; variável MEMBY (do tipo "byte") com valor indefinido

MEMBY DB 3A H ; variável MEMBY com valor definido e igual a 3A H

MEMWO DW ? ; variável MEMWO (do tipo "word") com valor indefinido

MEMWO DW 21AB H ; variável MEMWO com valor definido e igual a 21AB H

Exemplos de mnemônicos com instruções de transferência

Mnemônico	Descrição
MOV AX, 0100 H	Carrega registrador de 16 bits AX com valor 0100 h
MOV DL, 23 H	Carrega registrador de 8 bits DL com valor 23 h
MOV CL, 'char'	Carrega registrador de 8 bits com código ASCII do caractere
MOV BP, A8	Carrega registrador de 16 bits BP com valor 00A8 h
MOV DS, AX	Registrador de segmento DS é carregado com conteúdo de AX
MOV AX, BX	Carrega AX com conteúdo de BX
MOV AL, BL	Carrega registrador de 8 bits AL com conteúdo de BL
MOV AX, CS	Carrega registrador AX com conteúdo do segmento CS
MOV AX, [BX]	Carrega AX com conteúdo do endereço de offset BX
MOV [BP], CL	Carrega posição de offset BP com valor contido em CL
MOV AH, [SI]	Carrega AH com valor da posição cujo offset é SI
MOV AX, [1000 H]	Faz $AL \leftarrow [1000 H]$ e $AH \leftarrow [1001 H]$
MOV CL, MEMBY	Carrega CL com posição da variável de 1 byte MEMBY
MOV CX, MEMWO	Carrega CX com posição da variável de 1 word MEMWO
MOV CX, [MEMWO]	Carrega CX com posição da variável de 1 word MEMWO
MOV BX, [BX+2]	Carrega BX com conteúdo da posição BX+2 e BX+2+1
MOV [BP+2], BH	Carrega posição BP+2 com conteúdo do registrador BH
MOV AX, [SI-6]	$AL \leftarrow [SI - 6]$ e $AH \leftarrow [SI - 5]$
MOV AX, [BX+SI]	$AL \leftarrow [BX + SI]$ e $AH \leftarrow [BX + SI + 1]$
MOV BX, [BP+SI+3]	$BL \leftarrow [BP + SI + 3]$ e $BH \leftarrow [BP + SI + 3 + 1]$
MOV BX, MEMBY[SI]	O mesmo que MOV BX, [SI+MEMBY]

Observações.:

- Um registrador de segmento **não** pode ser carregado usando o modo imediato. Exemplos **incorretos**: MOV CS, 1000, MOV DS, 3A00.
- Somente o registrador BX e os ponteiros e índices podem referenciar memória. Exemplos incorretos: MOV AX, [DX], MOV CX, [AX]
- Destino e fonte não podem especificar posições de memória ao mesmo tempo. Exemplo incorreto: MOV [BX], [SI]

#### Instruções Especiais de Transferência de Dados:

Mnemônico	Descrição
XCHG AX, BX	permuta registradores AX e BX $\rightarrow (AX \leftrightarrow BX)$
XCHG AL, CL	Permuta registradores AL e CL
XCHG DX, [SI]	Permuta DX com o conteúdo da posição de memória endereçada por SI. Neste exemplo $DL \leftrightarrow [SI]$ e $DH \leftrightarrow [SI + 1]$
LAHF	$AH \leftarrow \text{Flags low}$
SAHF	$\text{Flags low} \leftarrow AH$
IN AH, 26	IN direto
IN AL, DX	IN indireto para as 65536 portas (0 - 65535 ou 0000 - FFFF).
OUT 26, AX	Aplica-se as mesmas regras da instrução IN.
OUT DX, AX	
LEA BX, MEMBY	Load Effective Address: carrega endereço efetivo. No exemplo, BX é carregado com o endereço efetivo de MEMBY (notar a diferença desta com a instrução "MOV BX, MEMBY")
LEA BX, [1000]	$BX \leftarrow 1000$
LDS BX, dword ptr [SI]	Load Pointer using DS. No exemplo $BX \leftarrow [SI+1:SI]$ , $DS \leftarrow [SI+3:SI+2]$ . Esta instrução é útil quando é preciso estabelecer novo endereço absoluto (composição endereço base + endereço de offset)
LES BX, dword ptr [SI]	Load Pointer using ES. No exemplo $BX \leftarrow [SI+1:SI]$ , $ES \leftarrow [SI+3:SI+2]$
XLAT	$AL \leftarrow [BX+AL]$ . Carrega registrador AL com o conteúdo (byte) da tabela iniciada em [BX] e com offset AL

Obs.:

- (a) A instrução IN somente pode ser usada para as primeiras 256 portas (0 a 255). Portas c/ endereço > 255 devem utilizar somente reg. DX (referência de I/O). Exemplo não permitido: IN AX, 3400.
- (b) A instrução IN deve utilizar somente o registrador AX. Exemplo não permitido: IN BL,DX.

**Exemplo 1:** O programa a seguir demonstra o uso de instruções de transferência de dados. O caractere "A" é escrito diretamente na memória de vídeo.

Mnemônico	Descrição
#MAKE COM#	Diretiva para o compilador gerar um arquivo ".com"
ORG 100h	Diretiva que indica o endereço inicial do programa: 0100 H
MOV AX, 0B800h	Carrega registrador AX com valor B800 H
MOV DS, AX	Copia valor de AX para DS, definindo segmento de dados
MOV CL, 'A'	Carrega CL com o código ASCII do caractere 'A', isto é, 41 H
MOV CH, 01011111b	Carrega CH com o valor binário 01011111 b = 5F H
MOV BX, 15Eh	Carrega registrador BX com valor 015E H
MOV [BX], CX	Copia conteúdo de CX na posição DS:BX, ou seja, B800:015E
HLT	Pára programa

**Exemplo 2:** Programa que demonstra o uso de variáveis

Mnemônico	Descrição
#MAKE COM#	Diretiva para o compilador gerar um arquivo ".com"
ORG 100h	Diretiva que indica o endereço inicial do programa: 0100 H

MOV AL, var1	Carrega registrador de 8 bits AL com valor 7 (variável var1)
MOV BX, var2	Carrega registrador BX de 16 bits com valor 1234 H (var2)
RET	Pára o programa
var1 DB 7	Define variável var1 como byte de valor 7 decimal
var2 DW 1234h	Define variável var2 como Word de valor 1234 hexadecimal

### 9.10.2 Instruções de Strings

- instruções para movimentação de blocos de dados (ou seja, vários bytes).
- reg. de seg. DS é utilizado como base e o reg. índice SI (*Source Index*) como offset para o cálculo do endereço fonte. Simbologia: **DS:SI**. (**OBS**: aceita registrador de segmento alternativo, por exemplo, CS, SS e ES).
- reg. de seg. ES é utilizado como base e o reg. índice DI (*Destination Index*) como offset para o cálculo do endereço destino. Simbologia: **ES:DI** (**OBS**: não aceita registrador de segmento alternativo. Deverá ser sempre ES).
- o flag DF (*Direction Flag*) é consultado nestas operações.

#### Instruções para movimentação de “strings”

Mnemônico	Descrição
MOVS BYTE PTR ES:[DI],[SI]	Copia bloco de dados de uma região para outra da memória. Origem: região DS:SI (a origem é sempre do segmento DS) Destino: região ES:DI (o destino é ES, mas poderia ser CS, DS ou SS)
MOVS WORD PTR ES:[DI],[SI]	Idêntico ao anterior, mas a movimentação é um Word por vez.
STOS BYTE PTR ES:[DI], AL	Armazena o byte em AL no endereço ES:DI. (Poderia ser CS, DS ou SS) ES:[DI] ← AL. Se D = 0 incrementa DI; se D = 1 decrementa DI
STOS WORD PTR ES:[DI], AX	Armazena o word em AX no endereço ES:DI. (Poderia ser CS, DS ou SS) ES:[DI] ← AL. Se D = 0 incrementa DI; se D = 1 decrementa DI
LODS BYTE PTR AL, ES:[DI]	Carrega AL com byte da posição ES:DI AL ← DS:[SI]. Se D = 0 incrementa SI; Se D = 1 decrementa SI
LODS WORD PTR AX, ES:[DI]	Carrega AX com word da posição ES:DI AL ← DS:[SI]. Se D = 0 incrementa SI; Se D = 1 decrementa SI
CMPS BYTE PTR ES:[DI],[SI]	Compara bloco de bytes de duas regiões de memória
CMPS WORD PTR ES:[DI],[SI]	Compara bloco de word de duas regiões de memória

#### Instruções para movimentação de “strings”

Mnemônico	Descrição
STOSB	Armazena o byte em AL no endereço padrão ES:DI. ES:[DI] ← AL. Se D = 0 incrementa DI; se D = 1 decrementa DI
STOSW	Armazena o word em AX no endereço padrão ES:DI. ES:[DI] ← AL ; ES:[DI + 1] ← AH. Se D = 0 DI ← DI + 2; Se D = 1 DI ← DI -2
LODSB	Copia em AL o byte localizado no endereço padrão DS:SI. AL ← DS:[SI]. Se DF=0 incrementa SI; Se DF=1 decrementa SI
LODSW	Copia em AX o word localizado no endereço padrão DS:SI. AL ← DS:[SI]; AH ← DS:[SI+1]. Se DF=0 SI ← SI+2; Se DF=1 SI ← SI -2
MOVSB	Copia bytes da região de origem padrão (DS:SI) para a região de destino padrão (ES:DI). Sendo a origem e o destino as regiões padrões (ao contrário da instrução MOVS), a instrução não precisa de argumentos. ES:[DI] ← DS:[SI]. Se D = 0, incrementa SI e DI; Se D = 1, decrementa SI e DI.
MOVSW	Copia words da região de origem padrão (DS:SI) para a região de destino padrão (ES:DI).

	Sendo a origem e o destino as regiões padrões (ao contrário da instrução MOVS), a instrução não precisa de argumentos. $ES:[DI] \leftarrow DS:[SI]; ES:[DI+1] \leftarrow DS:[SI+1]$ . Se $D = 0$ , $DI \leftarrow DI + 2$ e $SI \leftarrow SI + 2$ ; Se $D = 1$ , $DI \leftarrow DI - 2$ e $SI \leftarrow SI - 2$
CMPSB	Compara os bytes das posições DS:SI e ES:DI e atualiza Flags. $DS:[SI] - ES:[DI]$ .
CMPSW	Compara os words das posições DS:SI e ES:DI e atualiza Flags. $DS:[SI+1:SI] - ES:[DI+1:DI]$ .

### Uso do Prefixo REP (Repeat)

Precedendo as instruções de transferência de strings com "REP" faz com que estas instruções sejam repetidas o número de vezes igual ao conteúdo do registrador CX.

#### EXEMPLO 1: Uso de MOVSB e REP

Mnemônico	Descrição
#make_COM#	Diretiva para o compilador
ORG 100h	Diretiva para o compilador → programa começa na posição 100h
LEA SI, a1	Offset SI assume o endereço do primeiro valor da variável a1.
LEA DI, a2	Offset DI assume o endereço da variável a2.
MOV CX, 5	Registrador CX recebe valor 5
REP MOVSB	Instrução MOVSB é repetida até o registrador CX alcançar 0.
RET	Encerra programa
a1 DB 1,2,3,4,5	Variáveis
a2 DB 5 DUP(0)	A variável a2 terá 5 valores zero, conforme declarado em 5 DUP(0).

#### EXEMPLO 2: Um bloco de memória de 10 bytes com caractere "A3" é armazenado na memória, a partir do endereço físico E0000 H.

Mnemônico	Descrição
#make_COM#	Diretiva para o compilador
ORG 100h	Diretiva para o compilador → programa começa na posição 100h
MOV AX, 0E000 H	$AX = E000\text{ h}$
MOV ES, AX	Segmento especial $ES = AX = E000\text{ h}$
MOV DI, 0000	$DI = 0000\text{ h} \rightarrow ES:DI = E0000 + 00000 = E0000\text{ h}$
MOV AL, 0A3 H	Byte a ser armazenado é colocado em AL
CLD	clear direction flag (resseta flag "direção": modo auto-incremento)
MOV CX, 10	Faz $CX = 10 \rightarrow$ serão armazenado 10 bytes a partir de E0000 h
REP STOSB	Repete instrução 10 vezes
HLT	Pára programa

#### EXEMPLO 3: Copiar um bloco de memória de 1000 bytes do endereço físico A1000H para o endereço físico E1000 H.

Mnemônico	Descrição
MOV AX, 0A000 H	$AX = A000\text{ H} \rightarrow$ valor que será passado para registrador de segmento DS
MOV DS, AX	Segmento de dados $DS = AX = A000\text{ h}$
MOV SI, 1000 H	$SI = 1000\text{ h} \rightarrow DS:SI = E0000 + 00000 = A1000\text{ h}$
MOV AX, 0E000 H	$AX = E000\text{ h} \rightarrow$ valor que será passado para registrador de segmento ES
MOV ES, AX	Segmento especial $ES = AX = E000\text{ h}$
MOV SI, 1000 H	Registrador de offset $SI = 1000\text{ h} \rightarrow DS:SI = A1000\text{ H}$
MOV DI, 1000 H	Registrador de offset $DI = 1000\text{ h} \rightarrow ES:DI = E1000\text{ H}$
CLD	clear direction flag (resseta flag "direção": modo auto-incremento)
MOV CX, 03E7 H	Faz $CX = 03E7\text{ h} \rightarrow$ serão armazenado 1000 bytes a partir de E0000 h
REP MOVSB	Repete instrução 1000 vezes → serão copiados 1000 bytes



### 9.10.3 Instruções Lógicas

- referem-se a funções de lógica booleanas;
- cada instrução é realizada bit a bit;
- Há também instruções de rotação e deslocamento;
- Há 5 instruções booleanas: NOT, AND, OR, XOR, TEST.

Mnemônico	Descrição
NOT BX	BX é complementado (seus bits são invertidos)
NOT BYTE PTR [SI]	Byte apontado por SI é complementado
NOT WORD PTR [SI]	Word apontado por SI é complementado
AND CX, DX	Função AND entre CX e DX
AND BL, BYTE PTR [SI]	Função AND entre BL e conteúdo do Byte apontado por SI
AND AX, 8000 H	Função AND entre AX e 8000 H
OR CX, DX	Função OR entre CX e DX
OR BL, BYTE PTR [SI]	
OR AX, 8000	
XOR CL, DH	Função XOR entre CL e DH
XOR BX, WORD PTR [SI]	Função XOR entre BX e o Word apontado por SI
XOR AX, 8000	
TEST CX, DX	Semelhante ao AND, apenas não altera os operandos. É utilizado quando se deseja testar vários bits: se o teste do 1º bit fracassa, pode-se testar o 2º bit.
TEST AX, 3000	

**EXEMPLO:** Determine o estado do registrador AL e dos flags após a sequência de instruções:

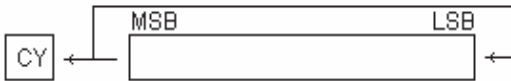
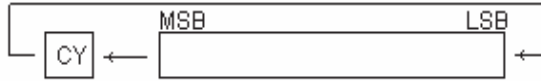
```
MOV AL, 6D
MOV BH, 40
AND AL, BL
```

Resultado das Flags:

C = 0 → não houve transporte do bit 7 para o 8 na operação de 8 bits executada  
 Z = 1 → o resultado da operação é zero  
 S = 0 → o bit mais significativo após a operação é zero (número positivo)  
 O = 0 → não houve “overflow” na operação.  
 P = 1 → há um número par de bits “1” nos 8 primeiros bits, após a operação (número de 1s = 0)  
 A = 0 → não houve transporte do bit 3 para o bit  
 I = 1 → interrupção desabilitada  
 D = 0 → os registradores SI e DI serão incrementados nas operações com “string”

#### Instruções de Deslocamento e de Rotação

SHL AL, 1 SHL BX, CL	Deslocar à esquerda uma vez. Formato: SHL <i>destino, contador</i> Deslocar à esquerda CL vezes. <div style="text-align: center;"> <p><b>Instrução SHL</b></p> </div>
SHR AX, 1 SHR BL, CL	Deslocar à direita uma vez. Deslocar à direita CL vezes <div style="text-align: center;"> <p><b>Instrução SHR</b></p> </div>

ROL AL, 1 ROR CX, CL	Rodar à esquerda. Rodar à direita CL vezes.  <b>Instrução ROL</b>
RCL AL, 1	rodar a esquerda através do CY  <b>Instrução RCL</b>
RCR DL, CL	Rodar à direita através do CY CL vezes.

#### 9.10.4 Instruções Aritméticas

Mnemônico	Descrição
ADD SI, DX	soma de registradores
ADD BYTE PTR [BX], CH	
ADD DI, 6000	
ADC SI, DX	soma de registradores com carry
ADC BYTE PTR [BX], CL	
SUB AX, BX	subtração de registradores
SUB BL, 34	subtração de cte imediata
SBB AX, CX	subtração de registradores com borrow
SBB BL, 34	subtração de cte imediata com borrow
DAA	ajuste decimal para adição
DAS	ajuste decimal para subtração
AAA	ajuste ASCII para adição
AAS	ajuste ASCII para subtração
INC CL	incremento de registrador
INC WORD PTR [SI]	incremento de byte apontado por SI
DEC CX	decremento de registrador
DEC WORD PTR [SI]	
NEG DX	complemento de 2 de registrador
NEG WORD PTR [SI]	
CMP BL, BH	comparação entre dois registradores, BL e BH
CMP [BX], CX	comparação entre word apontado por BX e CX
MUL BL	multiplicação de byte: $AX \leftarrow AL * BL$
MUL CX	multiplicação de word: $DX:AX \leftarrow AX * CX$
IMUL BL	multiplicação de número com sinal
DIV BL	divisão de byte: $AX \leftarrow AL / BL$ (AL: quociente; AH: resto)
DIV CX	divisão de word: $DX:AX \leftarrow AX / CX$ (AX: quoc.; DX: resto)
IDIV BL	divisão de número com sinal

#### 9.10.5 Instruções de Desvio

Mnemônico	Descrição
Desvio Incondicional	
JMP [BX]	$IP \leftarrow [BX+1:BX]$
JMP BX	$IP \leftarrow BX$
JMP DWORD PTR [BX];	$IP \leftarrow [BX+1:BX] ; CS \leftarrow [BX+3:BX+2]$
Desvio condicional	

JNC LABEL	desvia para LABEL se flag carry = 0 (se não há carry)
JS LABEL	desvia para LABEL se MSB =1 (flag de sinal =1)
LOOP LABEL	decrementa CX e desvia para LABEL se CX ≠ 0
LOOPE LABEL	decrementa CX e desvia para LABEL se CX ≠ 0 e ZF = 1
PUSH e POP → Só pode ser aplicado a registrador de 16 bits	
PUSH CX	salva registrador CX na pilha
PUSH [DI+2]	salva posição de memória apontada por DI+2 e DI+3
POP DS	Recupera DS da pilha
PUSHF	Salva registrador de Flags na pilha
POPF	Recupera registrador de Flags da pilha
CALL e RETURN	
CALL DELAY	chamada de sub-rotina "DELAY"
CALL [BX]	chamada de sub-rotina iniciada pelo conteúdo de memória apontado por BX: IP ← [BX+1:BX]
CALL BX	chamada de sub-rotina iniciada pelo BX: IP ← BX
RET	retorno de sub-rotina
IRET	retorno de rotina de interrupção (restaura CS e IP)
Interrupções por Software	
INT <i>nn</i>	Guarda na pilha o endereço de retorno (CS:IP), antes de chamar a interrupção <i>nn</i> .

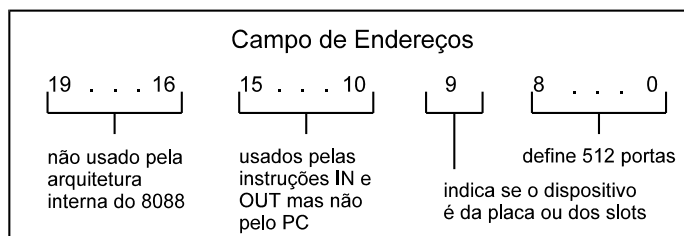
### 9.10.6 Instruções de Controle

Mnemônico	Descrição
STI	seta flag Interrupção
CLI	reseta flag Interrupção
STC	seta carry flag
CLC	reseta carry flag
CMC	complementa carry flag
STD	seta flag direção
CLD	reseta flag direção
HLT	para a CPU
WAIT	para a CPU até o pino test ficar ativo
NOP	sem operação (utilizado para gerar delay)
LOCK <i>instruction</i>	coloca Pino LOCK em "0" durante a execução da próxima instrução ( <i>instruction</i> )

## 9.11 Características de I/O do PC-XT

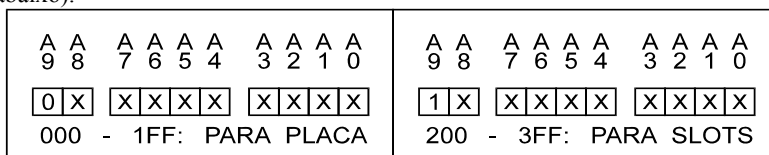
### 9.11.1 Divisão dos Endereços dos Dispositivos de I/O do PC-XT

Apesar de possuir 20 bits de endereço, a arquitetura do 8088 suporta apenas 64KB (65.536) endereços de dispositivos de I/O pois os bits de endereçamento A16 a A19 não são utilizados. A configuração usada no PC também não usa toda esta capacidade. Somente os 10 bits menos significativos do barramento de endereços são utilizados. Como  $2^{10} = 1024$ , este é o número de endereços de I/O disponíveis no PC (0000h a 03FFh). A figura abaixo mostra a divisão de endereços no PC.



O bit A9 tem um significado especial no projeto do PC: quando está inativo (0), os dados não podem ser recebidos (pelo micro) vindo dos slots. São habilitados apenas dados vindos de dispositivos de I/O instalados na própria placa mãe (*mother board*, *main board* ou *system board*). Quando ativo (1), são habilitados a recepção apenas de dados provenientes dos slots.

Tem-se assim 512 portas para uso interno do PC e 512 portas para uso de dispositivos conectados nos slots (ver figura abaixo).



Esta divisão aplica-se apenas à instrução IN (instrução de entrada de I/O). Para a instrução OUT (instrução de saída de I/O), todos os 1024 endereços podem ser utilizados nos slots.

Em sintonia com a restrição da instrução IN, o PC usa a primeira metade dos endereços para os dispositivos da placa principal (placa mãe) e a metade restante para os dispositivos dos slots (ver figura abaixo).

Endereço	Quantidade de Endereços	Função
0000 h 01FF h	512	Dispositivos da Placa Principal
0200 h 03FF h	512	Dispositivos dos Slots
0400 h FFFF h	64.512	Não usado no projeto do PC

A seguir serão apresentados os mapas de endereços de I/O do PC-XT. Deve-se observar que novas placas (por exemplo, placas de som ou de rede) poderão decodificar endereços que atualmente estão livres.

### 9.11.2 Endereços dos Dispositivos da Placa Principal (Placa Mãe)

Dispositivos de I/O residentes na placa principal fazem o serviço de interrupção, transferência de dados (DMA), contagem, etc. Na figura abaixo pode ser visto o mapa de endereçamento de I/O para a placa principal.

Endereço	Quantidade de Endereços	Função
0000 h 001F h	32	Chip de DMA (8239)
0020 h 003F h	32	Chip de interrupção (8259)
0040 h 005F h	32	Chip temporizador (8253)
0060 h 007F h	32	Chip PPI (8255)
0080 h 009F h	32	Registrador de páginas DMA
00A0 h 00BF h	32	Bit de interrupção NMI
00C0 h	320	Não codificado ou usado

01FF h

### 9.11.3 Endereços dos Dispositivos dos Slots

O conhecimento destes endereços torna-se a base de qualquer projeto de interface, já que representam o ponto de entrada e saída de dados (ver figura na página 16).

Nem todos os 512 endereços disponíveis nos slots podem ser utilizados para interfaceamento pois vários endereços são previamente reservados para dispositivos comuns ao sistema. Os dispositivos de I/O que normalmente são conectados nos slots são joysticks, portas paralelas (impressoras), portas seriais (RS-232), adaptador gráfico (CGA, MDA, EGA, VGA), acionador de disquetes e de disco rígido. Para evitar problemas de incompatibilidade de placa de interface entre diferentes micros deve-se evitar a duplicação desses endereços.

O modo mais simples de decodificar um endereço para uma porta de I/O ou um grupo de endereços para projetos de interface é inspecionar o mapa de endereçamento e encontrar um bloco de endereços não utilizado, e então, construir um circuito decodificador adequado. O endereço decodificado deve ser acionado logicamente com os sinais de barramento  $\overline{\text{IOR}}$  (leitura de I/O) ou  $\overline{\text{IOW}}$  (escrita de I/O) para gerar entradas ou saídas de dados, respectivamente.

Outro sinal importante é o AEN (*Address ENable*), gerado pelo controlador de DMA. Quando ativo (1), o DMA assume o controle dos barramentos e o microprocessador é desabilitado. Portanto, tem a função de desabilitar a decodificação de endereços de I/O durante o ciclo de DMA: deve fazer parte da lógica de decodificação. Como mostrado mais à frente, a faixa de endereços de 0300h à 031Fh está livre para uso em placas de interface.

### 9.11.4 O Slot do PC (ISA 8 bits)

O slot do PC apresenta todos os sinais de interesse para a interface com qualquer dispositivo de I/O. Os sinais disponíveis nos slots são os sinais de endereçamento, de dados, clock, tensões, etc. A configuração dos sinais no slot ISA 8 bits pode ser visualizada na figura da página 17. Os sinais mais utilizados são dados a seguir:

**Alimentação:** Servem para alimentar os CI's digitais e analógicos). São eles:

<b>GND</b>	(B1, B10, B31)
<b>+5 V</b> (B3, B29)	$I_{\text{máx}} = 700 \text{ mA}$
<b>- 5 V</b> (B5)	$I_{\text{máx}} = 30 \text{ mA}$
<b>+12 V</b> (B9)	$I_{\text{máx}} = 100 \text{ mA}$
<b>- 12 V</b> (B7)	$I_{\text{máx}} = 50 \text{ mA}$

**Sinais de Controle de I/O:**

Servem para controlar o acesso aos CI's de I/O). Alguns principais sinais de controle são:

**IOW** (B13) - habilita escrita nos periféricos de I/O quando em nível lógico baixo;

**IOR** (B14) - habilita leitura dos periféricos pelo microprocessador quando em nível lógico baixo;

**RESET DRV** (B2) - sinal de reset utilizado para ressetar portas programáveis da placa de interface;

**AEN** (A11): utilizado na lógica de decodificação de endereços para indicar quando está ocorrendo um DMA;

**CLOCK** (B20): sinal de clock fornecido pela placa principal, frequência = 4,77 MHz;

**DADOS:** de D0 à D7 (A9 - A2): sinais provenientes do barramento de dados;

**ENDEREÇOS:** de A0 à A9 (A31 - A22): sinais provenientes do barramento de endereços.

Todos os demais sinais não são usados frequentemente, pois apenas placas de interface mais sofisticadas necessitam destes outros recursos.

Endereço	Quantidade de Endereços	Função
0200 h 020F h	16	Adaptador de Jogos
0210 h 021F h	16	Unidades de Expansão

0220 h 026F h	80	Reservada ou Não Documentada
0270 h 027F h	16	Porta Paralela ou Impressora # 2
0280 h 02EF h	112	Reservada ou Não Documentada
02F0 h 02FF h	16	Porta Serial #2
0300 h 031F h	32	Placa de Prototipo
0320 h 032F h	16	Adaptador de Disco Rígido
0330 h 036F h	64	Não Documentada
0370 h 037F h	16	Impressora Paralela Principal
0380 h 03AF h	48	Reservada ou Não Documentada
03B0 h 03BF h	16	Placa de Vídeo Monocromática
03C0 h 03CF h	16	Placa de Vídeo EGA
03D0 h 03DF h	16	Placa de Vídeo CGA
03E0 h 03EF h	16	Reservado
03F0 h 03FF h	16	Disco Flexível e Porta Serial

## 9.12 Decodificação de Endereços

A decodificação de endereços de I/O é semelhante ao mapeamento de memória e é utilizada em qualquer placa de interface. Para acessar periféricos de I/O, o PC-XT utiliza-se apenas 10 bits de endereços, de A0 a A9. Pode, portanto endereçar 1024 endereços diferentes. Decodificar um endereço ou grupo de endereços é gerar um único sinal, geralmente baixo, a partir dos sinais de endereço e controle. Este sinal é chamado de habilitação e é ligado nos CI's periféricos no pino CS (*Chip Select*) ou CE (*Chip Enable*). Para gerar este sinal pode-se utilizar CI's decodificadores e portas lógicas mais simples, decodificando os sinais de controle ( IOR, IOW e AEN) e os sinais de Endereços 99(A0 até A9). O decodificador 74LS138 é um dos decodificadores usados nesse processo. Um decodificador é 74LS154, cuja principal diferença do 74LS138 é possuir dois sinais de habilitação e quatro sinais de seleção, gerando, portanto 16 saídas diferentes.

## 9.13 Exemplos Gerais

A maioria dos exemplos dados a seguir são exemplos do emulador do 8086 chamado **Emu8086 v2.58**. Eles podem ser verificados através desse emulador.

1. Adiciona o conteúdo de dois registradores

Mnemônico	Descrição
#make BIN#	Diretiva do compilador para gerar um arquivo “.bin”
MOV AX, 5	Registrador AX = 0005 (decimal)
MOV BX, 10	Registrador BX = 0010 (decimal)
ADD AX, BX	Faz $AX \leftarrow AX + BX \rightarrow AX = 0005\text{ h} + 000A\text{ h} = 000F\text{ h}$ (15 dec)
SUB AX,1	Faz $AX \leftarrow AX - 0001\text{ h} \rightarrow AX = 000F - 0001 = 000E\text{ h}$ (14 dec)
HLT	Pára programa

2. Calcula a soma dos elementos do vetor V1 e armazena o resultado na variável V2.

Rótulo	Mnemônico	Descrição
--------	-----------	-----------

	#make BIN#	Diretiva do compilador para gerar um arquivo “.bin”
	MOV CX, 5	Número de elementos CE = 5 (decimal)
	MOV AL, 0	Registrador AL registrará a soma dos elementos (valor inicial = 0)
	MOV BX, 0	BX é o indexador. Valor inicial = 0.
Next:	ADD AL, V1[BX]	Faz $AL \leftarrow AL + [V1 + BX]$ . Conteúdo da posição V1 + BX.
	MOV V1[BX], BL	Modifica o conteúdo da posição V1 + BX com o valor de BL
	INC BX	Incrementa BX
	LOOP Next	Retorna para “Next” até o contador CX = 0. Decrementa CX automaticamente
	MOV V2, AL	Armazena conteúdo de AL na variável V2
	HLT	Pára programa
	V1 DB 4, 3, 2, 1, 0	Valores do vetor V1
	V2 DB 0	Valor inicial da variável V2

3. Carrega registradores com valores em notação binária, hexadecimal e octal.

Rótulo	Mnemônico	Descrição
	#make BIN#	Diretiva do compilador para gerar um arquivo “.bin”
	MOV AL, 00000101b	Carrega AL com valor binário correspondente a 5
	MOV BL, 0Ah	Carrega registrador com o hexadecimal de 10
	MOV CL, 10o	Carrega CL com o valor octal que corresponde a 8
	ADD AL, BL	Adiciona o conteúdo de BL ao conteúdo de AL ( $5 + 10 = 15$ )
	SUB AL, CL	Subtrai o conteúdo de CL do conteúdo de AL ( $15 - 8 = 7$ )
	HLT	

## 9.14 Problemas Propostos

1. Rodar o programa “int21” do Emu8086 e verificar a entrada de textos pelo usuário e a impressão do mesmo no vídeo.
2. Rodar o programa “Advanced\_io” do Emu8086 e verificar o uso do display de 7-segmentos.
3. Criar um programa para fazer uma contagem hexadecimal crescente de 00 a FFh e mostrar a contagem no vídeo e no display de 7-segmentos.
4. Repetir o programa anterior para uma contagem hexadecimal de 0000h até FFFF h, ininterrupta.
5. Criar um programa para fazer uma contagem decimal crescente de 0 a 50, ininterrupta. Mostrar resultado no display de 7-segmentos.
6. Repetir o programa anterior para uma contagem decimal de 0 a 50000.
7. Criar um programa para fazer uma contagem hexadecimal decrescente de FFh a 00 ininterrupta e mostrar a contagem no vídeo e no display de 7-segmentos.
8. Criar um programa para fazer uma contagem decimal decrescente ininterrupta de 50000 a 0. Mostrar no display de 7-segmentos.
9. Fazer um programa que simula um cronômetro decrescente que conta de 20:00 min até 00:00. Mostrar o resultado no vídeo, no formato apresentado.
10. Fazer um programa que simula um relógio no formato hh:mm:ss. Mostrar resultado no vídeo.
11. Criar um programa para mostrar três diferentes frases no vídeo, em linhas diferentes.
12. Fazer um programa para somar os elementos de dois vetores de 5 valores, cada. O resultado deve ser mostrado no vídeo.

13. Fazer um programa para ler dois valores pelo teclado, adiciona-los e mostrar o resultado no display de 7-segmentos.
14. Adapte o programa “stepper\_motor” do Emu8086 para que o motor fique girando 10 vezes para um lado e 10 vezes para o outro lado, ininterruptamente.
15. Adapte o programa “traffic\_lights” do Emu8086 de forma a simular um cruzamento simples, ou seja, permissão para seguir em frente ou virar à direita para as duas avenidas (sinal de dois tempos).
16. Adapte o programa “traffic\_lights” do Emu8086 de forma a simular um cruzamento onde os dois sentidos da avenida horizontal tenha permissão para seguir em frente, virar à esquerda ou virar à direita e os veículos da avenida vertical tenham permissão apenas para seguir em frente ou virar à direita (sinal de três tempos).
17. Adapte o programa “traffic\_lights” do Emu8086 de forma a simular um cruzamento onde os veículos de todas as pistas possam seguir em frente, virar à direita ou virar à esquerda (sinal de quatro tempos).

### 9.15 Referências Bibliográficas

- [1] ZILLER, Roberto M., ‘Microprocessadores: Conceitos Importantes’, Edição do autor, Florianópolis, SC, Brasil, 2000.
- [2] ZELENOVSKY, Ricardo, MENDONÇA, Alexandre, ‘PC: Um Guia Prático de Hardware e Interfaceamento’, Interciência, Rio de Janeiro, RJ, 1996.
- [3] Tutorial do Emulador emu8086 – <http://www.emu8086.com>