2. Arquitetura do microprocessador 8086

2.1 A família Intel *iapx86* (ou 80X86)

Processador	Co-proc.	Ano de introdução	No. de bits	No. de transistores	Velocidade (MHz)
4004	-	1971	4	2.205	-
8008	-	1972	8	3.300	-
8080	-	1974	8	4.500	-
8085	-	1978	8	6.200	-
8086	8087	1978	16	29.000	4,77 a 10
8088	8087	1979	8 (16)	29.000	4,77 a 10
186	8087	1982	16	100.000	8 a 16
188	8087	1982	8 (16)	100.000	8 a 16
286	287	1982	16	134.000	8 a 12,5
386 (368 DX)	387	1985	32	375.000	16 a 40
386 SX	387 SX	1988	16 (32)	375.000	16, 20
486 (486 DX)	-	1989	32	1.200.000	25, 33, 50
486 SX	487 SX	1991	32	1.200.000	20, 25
486 DX2	-	1992	32	?	66
486 DX4	-	1993	32	?	99
Pentium	-	1993	32/64	3.100.000	60, 66, 90
P6 (?)	-	1995 - 96	? 64	? 22.000.000	? 120, 150

- O primeiro IBM PC foi construído com o 8088 (versão de 8 bits do 8086).
- Todo software produzido para a família *iapx86* mantem compatibilidade com os futuros microprocessadores.

2.2 Organização dos microprocessadores 8086 e 8088

(aqui há uma figura colada: referir à apostila, disponível no Xerox)

Configuração interna do 8086

- barramento de endereços é comum com o de dados: 20 bits
- endereços: 20 bits, 2^{20} = 1.048.576 combinações = 1 MByte (1 MB)
- dados: utiliza somente 16 bits do barramento comum
- barramento de controle: 16 bits independentes do barramento comum

8088 -> versão do 8086 que se conecta externamente a um barramento de dados de 8 bits, processando internamente 16 bits.

O 8086 divide-se internamente em duas unidades.

Execution Unit (EU) - unidade de execução:

- UAL - realiza operações aritméticas de +, -, X, / e operações lógicas AND, OR, NOT, XOR;
- contem registradores para armazenamento temporário durante as operações, que são **endereçados por nome**.

BUS Interface Unit (BIU) - unidade de interface de barramento:

- faz a comunicação de dados entre a EU e o meio externo (memória, E/S);
- controla a transmissão de sinais de endereços, dados e controle;
- controla a sequência de busca e execução de instruções;
- mecanismo de ***pre-fetch***: busca até 6 instruções futuras deixando-as na fila de instruções (*instruction queue*) -> aumento de velocidade.

Registradores: elementos de memória muito rápida dentro da CPU.

- de dados, ou de propósito geral
- de endereços (segmentos, apontadores e índices)
- sinalizadores de estado e controle (**FLAGS**)

Registradores de dados:

AX, BX, CX e DX

- são todos registradores de **16 bits**
- utilizados nas operações aritméticas e lógicas
- podem ser usados como registradores de 16 ou 8 bits

AH e **AL**	8 registradores de 8 bits cada
BH e **BL**	
CH e **CL**	"H" -> byte alto ou superior
DH e **DL**	"L" -> byte baixo ou inferior

AX (acumulador) -> utilizado como acumulador em operações aritméticas e lógicas; em instruções de E/S, ajuste decimal, conversão, etc

BX (base) -> usado como registrador de **BASE** para referenciar posições de memória;
BX armazena o endereço BASE de uma tabela ou vetor de dados, a partir do qual outras posições são obtidas adicionando-se um valor de deslocamento (***offset***).

CX (contador) -> utilizado em operações iterativas e repetitivas para contar bits, bytes ou palavras, podendo ser incrementado ou decrementado; **CL** funciona como um contador de 8 bits.

DX (dados) -> utilizado em operações de multiplicação para armazenar parte de um produto de 32 bits, ou em operações de divisão, para armazenar o resto;
utilizado em operações de E/S para especificar o endereço de

uma porta de E/S.

Registradores de segmento:

CS, DS, SS e ES

- são todos registradores de 16 bits
- o endereçamento no 8086 é diferenciado para:
  - código de programa (instruções)
  - dados
  - pilhas

- **segmento:** é um bloco de memória de 64 KBytes, endereçável.

- durante a execução de um programa no 8086, há **4 segmentos ativos**:

segmento de código:	endereçado por	**CS**
segmento de dados:	"	**DS**
segmento de pilha:	"	**SS** (stack segment)
segmento extra:	"	**ES**

Registrador apontador de instrução:

IP (*instruction pointer*)

- utilizado em conjunto com **CS** para localizar a posição, dentro do segmento de código corrente, da próxima instrução a ser executada;

- **IP** é automaticamente incrementado em função do número de bytes da instrução executada.

Registradores apontador de pilha e de índice:

Armazenam valores de deslocamento de endereços (***offset***), a fim de acessar regiões da memória muito utilizadas:

- pilha,
- blocos de dados,
- *arrays* e *strings*.

Podem ser utilizados em operações aritméticas e lógicas, possibilitando que os valores de deslocamento sejam resultados de computações anteriores.

- **SP** (*stack pointer* - **apontador de pilha**) é utilizado em conjunto com SS, para acessar a área de pilha na memória; aponta para o topo da pilha.

- **BP** (*base pointer* - **apontador de base**) é o ponteiro que, em conjunto com SS, permite acesso de dados dentro do segmento de pilha.

- **SI** (*source index* - **índice fonte**) usado como registrador índice em alguns modos de endereçamento indireto, em conjunto com **DS**.

- **DI** (*destination index* - **índice destino**) similar ao SI, atuando em conjunto com **ES**.

Obs: **SI** e **DI** facilitam a movimentação de dados sequenciados entre posições fonte (indicado por SI) e posições destino (indicado por DI).

Registrador de sinalizadores (FLAGS):

- indica o estado do microprocessador durante a execução de cada instrução;
- conjunto de bits individuais, cada qual indicando alguma propriedade;
- subdividem-se em: **FLAGS** da estado (*status*) e **FLAGS** de controle.
- organização:

  1 registrador de 16 bits
  6 FLAGS de estado
  3 FLAGS de controle
  7 bits não utilizados (sem função)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	**DF**	**IF**	**TF**	**SF**	**ZF**		**AF**		**PF**		**CF**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	**DF**	**IF**	**TF**	**SF**	**ZF**		**AF**		**PF**		**CF**
											#		#		#

Flags de estado:

- **CF - Flag de Carry**

  CF = 1 -> após instruções de soma que geram "vai um" após instruções de subtração que não geram "emprestimo" ("empresta um");

  CF = 0 -> caso contrário.

- **PF - Flag de paridade**

  PF = 1 -> caso o byte inferior do resultado de alguma operação aritmética ou lógica apresentar um número **par** de "1's";

  PF = 0 -> caso contrário (número **impar**).

- **AF - Flag de Carry Auxiliar:** utilizado em instruções com números BCD

  AF = 1 -> caso exista o "vai um" do bit 3 para o bit 4 de uma adição caso não exista "empréstimo" do bit 4 para o bit 3 numa subtração;

  AF = 0 -> caso contrário.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	**DF**	**IF**	**TF**	**SF**	**ZF**		**AF**		**PF**		**CF**
				#				#	#						

- **ZF - Flag de Zero**

  ZF = 1 -> caso o resultado da última operação aritmética ou lógica seja igual a zero;

  ZF = 0 -> caso contrário.

- **SF - Flag de Sinal:** utilizado para indicar se o número resultado é positivo ou negativo em termos da aritmética em **Complemento de 2** (se não ocorrer erro de transbordamento - *overflow*).

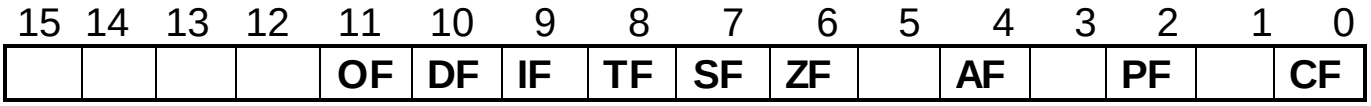
  SF = 1 -> número negativo;

  SF = 0 -> número positivo.

- **OF - Flag de Overflow** (erro de transbordamento).

  OF = 1 -> qualquer operação que produza *overflow*;

  OF = 0 -> caso contrário.



\# \# \#

Flags de controle

- **TF - Flag de Trap** (armadilha)

  TF = 1 -> após a execução da próxima instrução, ocorrerá uma interrupção; a própria interrupção faz TF = 0;

  TF = 0 -> caso contrário

- **IF - Flag de Interrupção**

  IF = 1 -> habilita a ocorrência de interrupções;

  IF = 0 -> inibe interrupções tipo INT externas.

- **DF - Flag de Direção:** usado para indicar a direção em que as operações com *strings* são realizadas.

  DF = 1 -> decremento do endereço de memória (DOWN);

  DF = 0 -> incremento do endereço de memória (UP).

Os registradores do 8086 (visão geral):

Registradores de dados

AH	**AL**	**AX**
BH	**BL**	**BX**
CH	**CL**	**CX**
DH	**DL**	**DX**

Registradores de segmentos

CS
DS
SS
ES

Registradores índices e apontadores

SI
DI
SP
BP
IP

Registrador de sinalizadores

FLAGS

- O registrador **IP** corresponde ao Contador de Programa - PC
- Todos os registradores são de 16 bits
- **AH** -> byte alto de AX; **AL** byte baixo de AX; ambos de 8 bits

Gerenciamento de memória por segmentação

- O **8086** possui 20 bits para acessar posições de **memória física**
- 2^{20} = 1.048.576 bytes (**1 Mbyte**) posições endereçáveis
- Exemplos de endereços:

0000 0000 0000 0000 0000b	->	00000h	
0000 0000 0000 0000 0001b	->	00001h	
0000 0000 0000 0000 0010b	->	00002h	-> 5 dígitos hexa
0000 0000 0000 0000 0011b	->	00003h	
....			
1111 1111 1111 1111 1111b	->	FFFFFh	

- O 8086 opera internamente com 16 bits
- **Problema**: Como gerar endereços com 20 bits?
- **Solução**: Utilizar a idéia de **segmentação de memória**!

Segmento de memória:

- bloco de 64 Kbytes de posições de memória consecutivas
- 2^{16} = 65.536 bytes (64 Kbytes)
- Segmento de memória é identificado por um número de segmento
- Uma **posição de memória** é especificada pelo **número de segmento** e por um **deslocamento** (***offset***) em relação ao início do segmento

Formato de **endereço lógico** -> **segmento:offset**

Exemplo de endereçamento

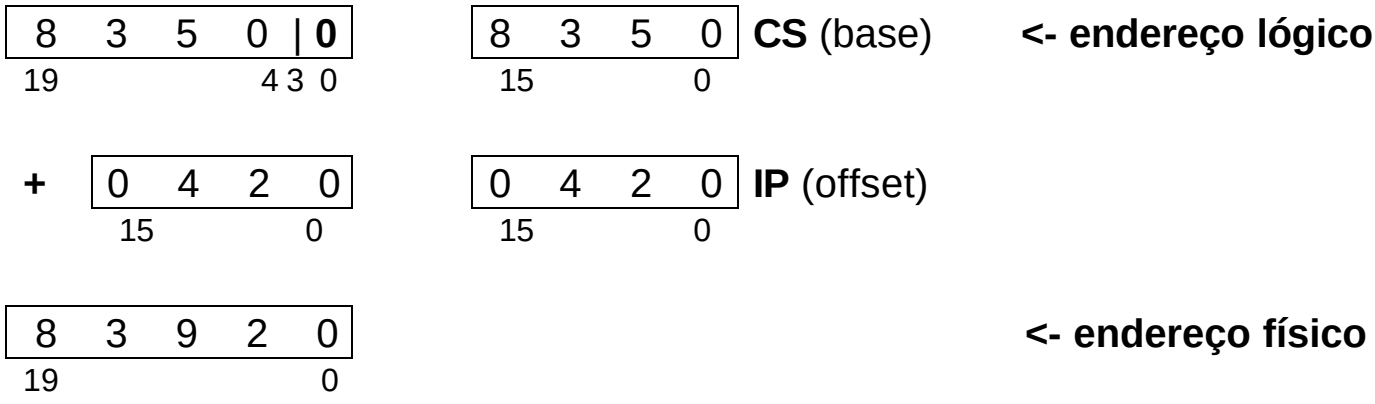
Dado o endereço lógico: 8350:0420h

reconhece-se: segmento no. 8350h
deslocamento 0420h

o endereço físico vale:

	8350**0**h	-> desloca-se 1 casa hexa (4 casas binárias)
+	0420h	-> soma-se o deslocamento
	83920h	-> endereço físico resultante (20 bits)

Graficamente:



para a memória

Tipos de referência à memória	Identificador de segmento	Identificador alternativo	Offset
Busca de instrução	CS	-	IP
Operação com pilha	SS	-	SP, BP
Variável (dado)	DS	CS, SS, ES	*
Fonte para instrução com *string*	DS	CS, SS, ES	SI
Destino para instrução com *string*	ES	-	DI

- O identificador de **segmento** (base) aponta para uma **região** da memória;
- O **offset** aponta para um local **dentro** deste segmento;
- O offset é aquele que **aparece nos programas** como o endereço dos dados, rótulos e endereços de instruções;
- O identificador de segmento aparece somente quando um **novo segmento** precisa ser especificado;
- **Segmentação** é um esquema muito útil para gerar **códigos relocáveis**;
- A maioria das **variáveis** está localizada no **segmento de dados**; podem também estar localizadas em outros segmentos;

- Endereços lógicos diferentes podem representar o mesmo endereço físico;

ex: base 028Ch
offset 0003h endereço físico -> 028C3h

base 0287h
offset 0053h endereço físico -> 028C3h.

Interrupção

Ocorrência eventual, durante a execução de um processamento pelo computador, que deve ser **prontamente** atendida, causando a **suspensão** do processamento em curso para o atendimento da "*chamada*".

Tipos de interrupções do 8086:

- Causadas pela ocorrência de **eventos "catastróficos"**:
  - falta de energia
  - erro de memória
  - erro de paridade em comunicações
  - este tipo de interrupção **não pode ser inibida**

- Causadas pela ação de **dispositivos externos** (periféricos):
  - podem ser **habilitadas** ou **inibidas**

- Causadas pelo **próprio programa** em curso:
  - erro de divisão
  - erro de transbordamento (***overflow***)
  - **TRAP** - útil para depuração de um programa
  - **BREAKPOINT** - colocado em pontos estratégicos do programa para permitir processamento especial

-Interrupção **RESET**
  - permite a **inicialização** do microprocessador, via ***hardware***.

2.3 Organização de um PC (*personal computer*)

Sistema operacional (SO): tipicamente **DOS** (*Disk Operating System*)

Funções:

- ler e executar comandos digitados pelo usuário no teclado;
- realizar operações de E/S;
- gerar mensagens de erro, quando necessário;
- gerenciar a memória (primária e secundária) e outros recursos.

DOS consiste em vários programas (rotinas):

- COMMAND.COM -> que responde aos comandos do usuário;
- outras rotinas de serviço.

Em operação normal, muitas rotinas do **DOS** não estão carregadas na memória, para economizar espaço.

BIOS (***Basic Input/Output System***): rotinas que realizam E/S para o PC.

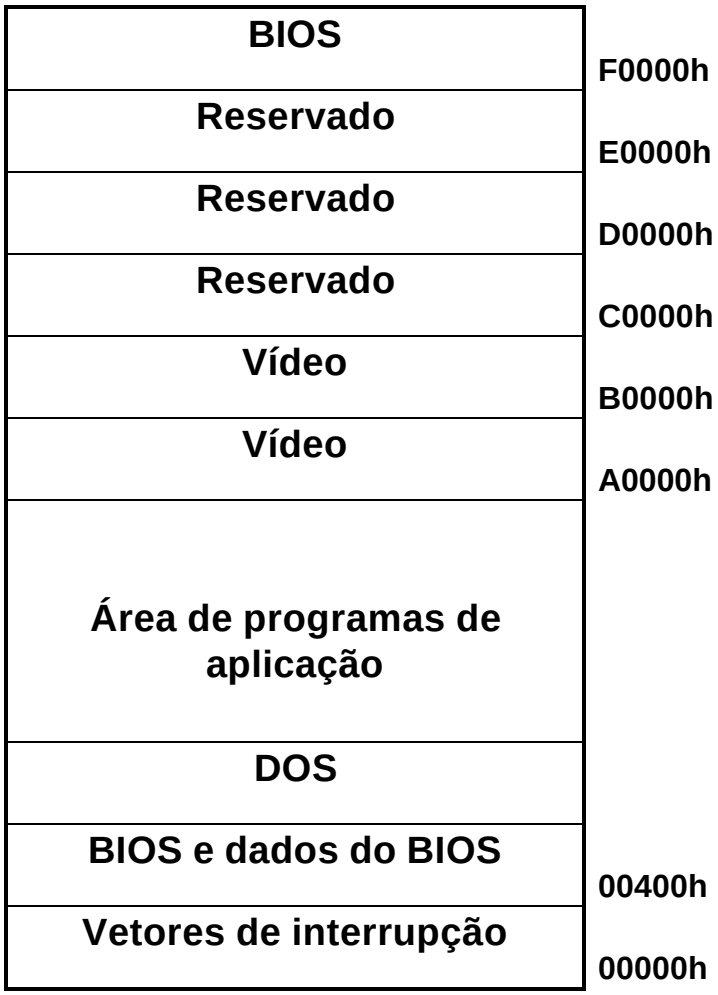
- são extremamente relacionadas à configuração do hardware;
- realizam a carga do **DOS** na memória
- realizam **verificação funcional** dos circuitos do PC
- os endereços das rotinas (**vetores de interrupção**) estão armazenados em endereços começando em **0000h**; o **DOS** também possui os endereços de suas rotinas armazenados nesta região;
- responsável por carregar o **programa de *boot*** (durante a inicialização)

Obs: Em **FFFF0h** há algumas poucas instruções de inicialização (***boot***), gravadas em memória **ROM**.

Organização de memória

A ocupação de memória feita pelo **DOS** (1024 Kbytes = 1 Mbyte) considera:

- 640 Kbytes para programas (10 segmentos disjuntos de 64 Kbytes)
- 384 Kbytes reservados para memória de vídeo, BIOS, etc.



Na área de programas do **DOS**, existem áreas para:

- os vetores de interrupção (1 Kbyte -> endereços **00000h** a **003FFh**)
- o próprio BIOS e seus dados
- o próprio DOS.

Na área restante, podem ser carregados os programas do usuário.