

AC

Ziffer 1

$$\begin{array}{ll} X_{(2)} = 10010100 & 148_{(10)} \\ X_{(8)} = 377 & 255_{(10)} \\ X_{(16)} = 100 & 256_{(10)} \end{array}$$

$$\begin{array}{ll} X_{(10)} = 77 & 1001101_{(2)} \\ X_{(8)} = 316 & 11001110_{(2)} \\ X_{(16)} = FF & 11111111_{(2)} \end{array}$$

$$\begin{array}{ll} X_{(2)} = 1000001 & 101_{(8)} \\ X_{(10)} = 15 & 17_{(8)} \\ X_{(16)} = 16 & 26_{(8)} \end{array}$$

$$\begin{array}{ll} X_{(2)} = 10001 & 11_{(16)} \\ X_{(8)} = 33 & 1B_{(16)} \\ X_{(10)} = 46 & 2E_{(16)} \end{array}$$

$$\begin{array}{l} X_{(2)} = 1001_{(2)} + 111_{(2)} = 10000 \\ X_{(2)} = 1001_{(2)} + 11111_{(2)} = 101100 \\ X_{(2)} = 1101_{(2)} + 1111_{(2)} = 11116 \end{array}$$

$$\begin{array}{l} X_{(8)} = 15_{(8)} + 13_{(8)} = 30 \\ X_{(8)} = 15_{(8)} + 73_{(8)} = 88 \\ X_{(8)} = 17_{(8)} + 43_{(8)} = 62 \end{array}$$

$$\begin{array}{l} X_{(16)} = 1A_{(16)} + 1D_{(16)} = 37 \\ X_{(16)} = A1_{(16)} + EF_{(16)} = 190 \\ X_{(16)} = 1AF_{(16)} + 1D_{(16)} = 1CC \end{array}$$

$$\begin{array}{l} X_{(256)} = 192.168.0.0_{(256)} + 128_{(10)} = 192.168.0.128 \\ X_{(256)} = 192.168.0.0_{(256)} + 254_{(10)} = 192.168.0.254 \\ X_{(256)} = 192.168.0.0_{(256)} + 256_{(10)} = 192.168.1.0 \\ X_{(256)} = 192.168.0.0_{(256)} + 512_{(10)} = 192.168.2.0 \end{array}$$

última

(1)

Código binário	Código binário natural	Sinal e valor absoluto	Complementos para 1	Complementos para 2
0000	0	0	1	0
0001	1	1	2	1
0010	2	2	3	2
0011	3	3	4	3
0100	4	4	5	4
0101	5	5	6	5
0110	6	6	7	6
0111	7	7	-1	7
1000	8	-0	-7	-8
1001	9	-1	-6	-7
1010	10	-2	-5	-6
1011	11	-3	-4	-5
1100	12	-4	-3	-4
1101	13	-5	-2	-3
1110	14	-6	-1	-2
1111	15	-7	0	-1

01110

(2)

Nº de Bits	Código binário natural		Sinal e valor absoluto		Complementos para 1		Complementos para 2	
	Máximo	Mínimo	Máximo	Mínimo	Máximo	Mínimo	Máximo	Mínimo
5	11111	00000	01111	11111	01111	10000	01111	10000
7	1111111	0000000	0111111	1111111	0111111	1000000	0111111	1000000
8	11111111	00000000	01111111	11111111	01111111	10000000	01111111	10000000

(3) 3 bits

Nº Inteiro	Sinal e valor absoluto	Complementos para 1	Complementos para 2
+3	011	011	011
+2	010	010	010
+1	001	001	001
0	000	111/000	000
-1	110	110	111
-2	101	101	110
-3	011	100	101
-4	Não Representável.		

(4)

Nº Inteiro	Nº mínimo de bits		
	Sinal e valor absoluto	Complementos para 1	Complementos para 2
-64	8	8	7
+128	9	9	8
-16	6	6	5
+12	5	5	4
-250	9	9	8

*em complemento para 1 trocou os 0 por 1 e os 1 por 0
0/1 determina-se os valores positivos, negativos
0 - positivante*

(5)

5) Efectue, em complementos para 1 e complementos para 2, com 6 bits, as somas algébricas seguintes:

$$\text{a)} (-24) + (-7)$$

$$(-31) + (+15)$$

$$\text{b)} (-23) + (-9)$$

$$(+13) + (+20)$$

$$\text{c)} (+18) + (-23)$$

$$(-3) + (+30)$$

a) $-24_{(10)} = 11000_{(2)}$

$c_1 \hookrightarrow 100111$

$$\begin{array}{r} 100111 \\ + 111000 \\ \hline 1011111 \\ + \quad \quad \quad 1 \\ \hline 100000 \end{array}$$

$-51_{(10)}$

 c_2

$$+_{(10)} = 111_{(2)}$$

$c_1 \hookrightarrow 111\ 000$

$$011111_{(2)} = 31_{(10)}$$

 c_2

$$-24_{(10)} = 101000$$

$$-+_{(10)} = 111001$$

$$101000$$

$$111001$$

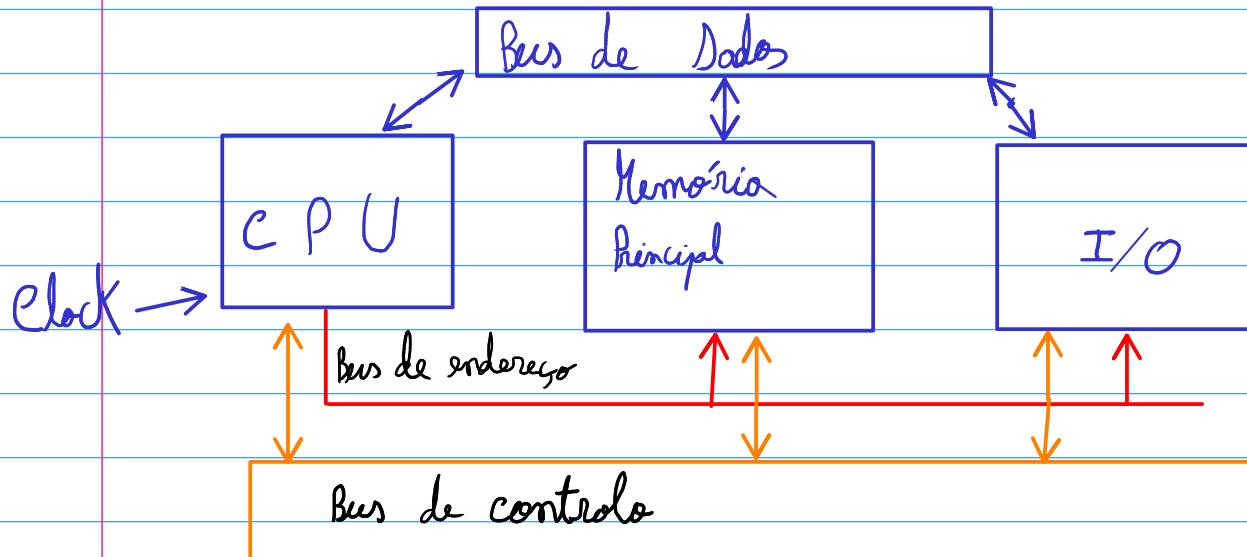
$$\cancel{X}100000$$

$$\downarrow \hookrightarrow 011111_{(2)}$$

$$-31_{(10)}$$

$$31_{(10)}$$

Modelo de arquitetura de Von Neumann



*Clock → Define a frequência de operação do sistema.
É utilizado para efetuar a sincronização das operações*

*CPU → Unidade central de processamento (central processing unit)
É o "cérebro" do computador. Lê instruções de memória, executa instruções, lê dados e escreve resultados.*

*Memória Principal → É capaz de armazenar informação binária, normalmente organizada em colunas de 8 bits! Armazena dados, instruções e resultados.
Dividida entre a RAM e a ROM*

Unidades I/O → Utilizadas para estabelecer uma comunicação com o mundo exterior, como teclado, rato, etc.

Bus de Sistema → conjunto de linhas (ligações) que transportam informação. Permite a comunicação entre o eCPU, Memória e Periféricos

Bus de Dados → conjunto de ligações físicas que transportam informação entre o eCPU, Memória e Periféricos. A largura do Bus de Dados é dada pelo nº de linhas do bus, ou nº de bits do microprocessador.

Bus de Endereços → conjunto de ligações físicas que transportam o endereço das células de memória ou dos portos de I/O (unidirecional). A largura do bus de endereço é dada pelo nº de linhas do bus ou pelo nº de bits, e define a capacidade de endereçamento.

Bus de controlo → Contém os sinais necessários para uma correta implementação do protocolo de comunicação.

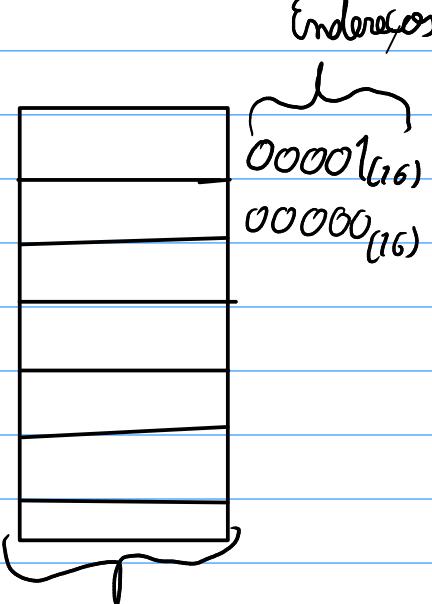
Armazenamento da Informação Digital Binária

A organização da memória é feita em células.

As células normalmente têm 8 bits

Cada célula tem 1 endereço

Os bytes menos significativos não armazenados
em endereços menos significativos.



- byte - 8 bits
- Word - 16 bits
- dWord - 32 bits

Estrutura interna do CPU (microprocessador) 8086

Registros de uso genérico

Ax registro de 16 bits

AH registro de 8 bits (8 bits + significativo)

AL registro de 8 bits (8 bits - significativo)

Registro acumulador

Implicito em algumas instruções

Bx registro de 16 bits

BH registro de 8 bits (8 bits + significativo)

BL registro de 8 bits (8 bits - significativo)

Registro de base

Normalmente utilizado para endereçar variáveis em memória

Cx registro de 16 bits

CH registro de 8 bits (8 bits + significativo)

CL registro de 8 bits (8 bits - significativo)

Registro acumulador

Implicito em algumas instruções como contador

Dx registro de 16 bits

DH registro de 8 bits (8 bits + significativo)

DL registro de 8 bits (8 bits - significativo)

Registro de Dados

Utilizado em algumas operações aritméticas

Utilizado em instruções de I/O

Registo de endereçamento

SP registo de 16 bits

Stack Pointer

Utilizado para referenciar variáveis na pilha do sistema

BP registo de 16 bits

Base Pointer

Utilizado para referenciar parâmetros e variáveis locais em subrotinas

Registos de indexação

SI registo de 16 bits

Source Index

DI registo de 16 bits

Destination Index

Registos Especiais

Registo apontador de instrução

Instruction Pointer (IP)

registo de 16 bits

contém o endereço da próxima instrução a ser executada

Registo de Segmentos

Neste processador existem 4 segmentos:

CS - "Code Segment" - Segmento de Código - Armazena as instruções do programa

ES: [IP] - Ponteiro para a próxima instrução a ser executada

DS - "Data Segment" - Segmento de Dados
Armazena dados/resultados relativos a variáveis
DS: (deslocamento)

SS: "Stack Segment" - Segmento de Pilha

Suporte à programação estruturada; Parâm. de Parâmetros

SS: (deslocamento)

ES - "Extra Segment" - Segmento extra

Segmento auxiliar, utilizado por exemplo na manipulação de algoritmos endereços de caractres.

Segmentos de Memória

Endereço lógico

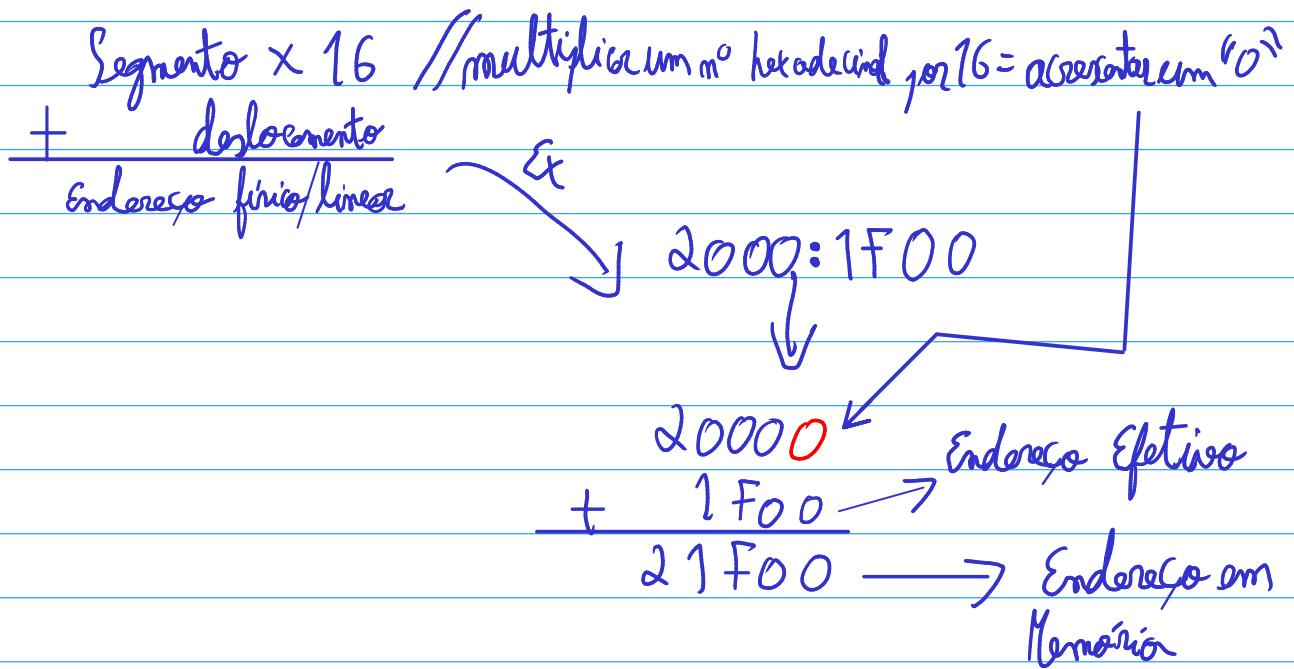
O endereço lógico pode ter os seguintes formatos:

- (segmento): [(deslocamento)]
- (segmento): [(offset)] -
- (segmento): [(displacement)]

endereço efetivo

Endereço físico/linear

O endereço físico/linear pode ser determinado da seguinte forma:



ES - Extra Segment

Início

52B90₍₁₆₎

52B90₍₁₆₎

+ FFFF₍₁₆₎

Fim 62B8F₍₁₆₎

SS - Stack Segment

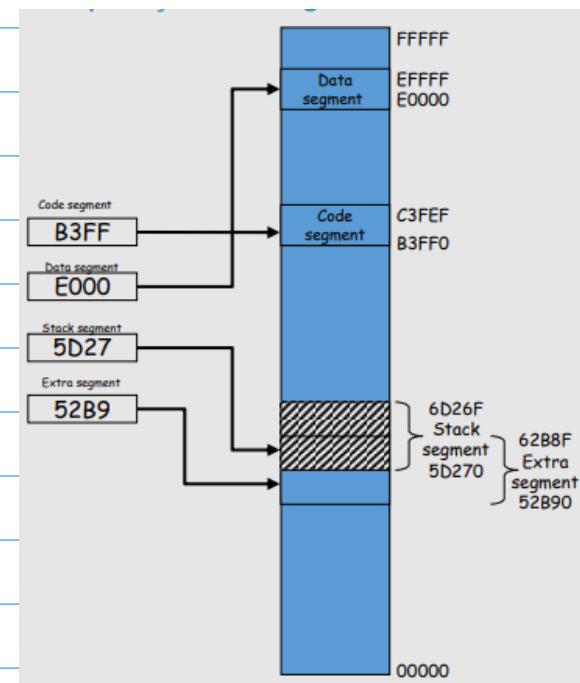
Início

5D270₍₁₆₎

5D270₍₁₆₎

+ FFFF₍₁₆₎

Fim 6D26F₍₁₆₎



Ciclo do Processador - fetch

O CPU procura as instruções na memória principal uma por uma.

Ciclo do Processador - decode

O CPU decodifica a instrução presente no program counter.

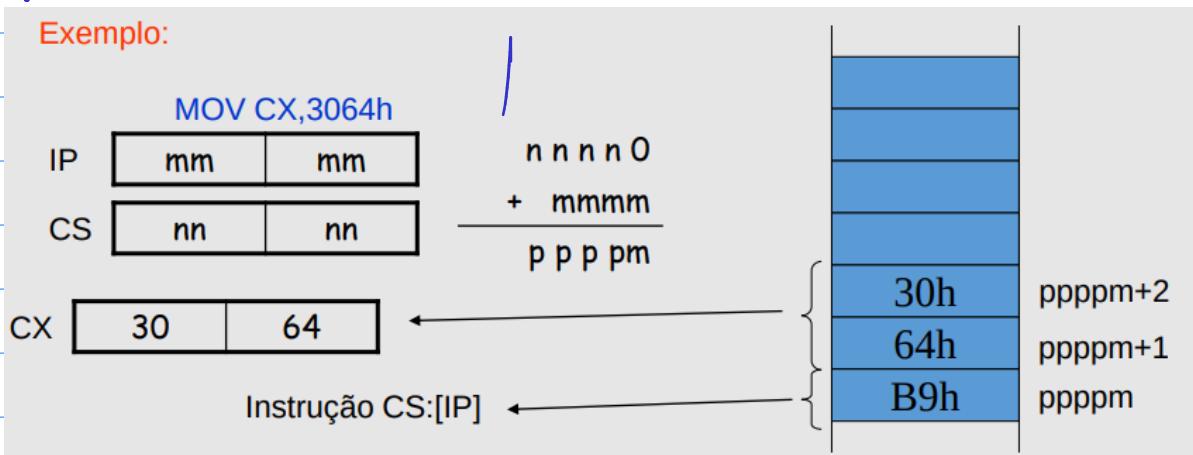
Ciclo do Processador - execute

O CPU executa a instrução anteriormente decodificada

Modos de endereçamento no 8086

Endereçamento imediato

Este tipo de endereçamento acontece quando o operando está contido na própria instrução.



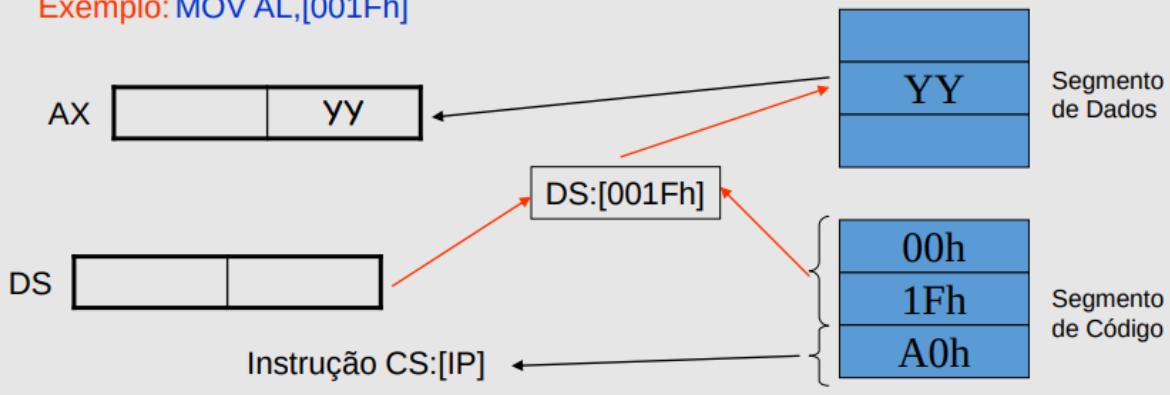
Endereçamento directo

Este tipo de endereçamento acontece quando a instrução fornece o endereço em memória do operando

Endereçamento directo

Este tipo de endereçamento acontece quando a instrução fornece o endereço em memória do operando

Exemplo: MOV AL,[001Fh]



Endereçamento Indireto

Este tipo de endereçamento acontece quando o endereço do operando é obtido por:

Registo de base + deslocamento: $[BX + disp]$ ou $[BX]disp$ ou $disp[BX]$ ou...

Registo de indexação: $[SI]$ ou $[DI]$

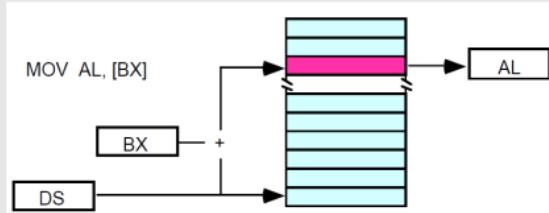
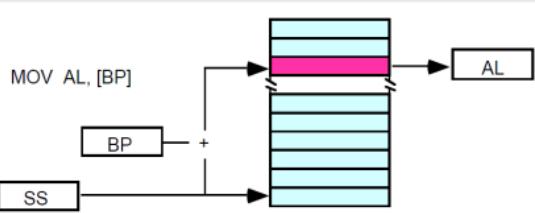
Registo de indexação + deslocamento: $[SI + disp]$ ou $[DI + disp]$ ou...

Registo de base + registo de indexação: $[BX + SI + disp]$ ou...

Endereçamento Indirecto

MOV AL, [BX]
MOV AL, [BP]
MOV AL, [SI]
MOV AL, [DI]

- [BX], [SI] e [DI] usam o segment DS por defeito
- [BP] usa o segment SS por defeito



Instruções Aritméticas

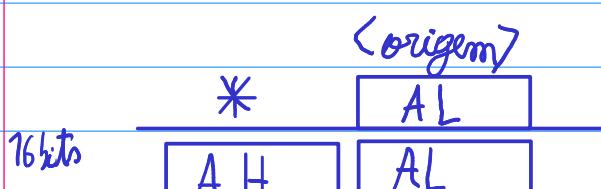
Instrução	Descrição	Bits de estado(Flags) afectados
ADD a, b	$a \leftarrow a + b$	Z, C, O, N
ADC a, b	$a \leftarrow a + b + C$	Z, C, O, N
NEG a	$a \leftarrow -a$	Z, C, O, N
SUB a, b	$a \leftarrow a - b$	Z, C, O, N
SBB a, b	$a \leftarrow a - b - C$	Z, C, O, N
MUL b	$ax \leftarrow al * b \} \{ dx:ax \leftarrow ax * b$	Z, C, O, N
DIV b	$ax \leftarrow ax / b \} \{ dx:ax \leftarrow dx:ax / b$	Z, C, O, N
INC a	$a \leftarrow a + 1$	Z, O, N
DEC a	$a \leftarrow a - 1$	Z, O, N

Diferença entre MUL e IMUL

MUL é apenas a números inteiros > 0

IMUL é para todos os números inteiros

Multiplicação de 8 bits



Multiplicação de 16 bits



Tipos de instruções de acordo com a função que realizam

AND	a, b	$a_i \leftarrow a_i \wedge b_i \ (i \in 0..N-1)$
OR	a, b	$a_i \leftarrow a_i \vee b_i \ (i \in 0..N-1)$
XOR	a, b	$a_i \leftarrow a_i \oplus b_i \ (i \in 0..N-1)$
NOT	a	$a_i \leftarrow \overline{a_i} \ (i \in 0..N-1)$

Operadores Lógicos bit a bit

AND // multiplicação

Sintaxe: AND < destino > < origem >

<destino>	$\rightarrow 10010011$
<origem>	$\xrightarrow{\text{AND}} 00100111$
<destino>	$\rightarrow 00000011$

Neste exemplo vemos que na operação AND, o 0 é o elemento aboriente, e o 1 é o elemento neutro

OR

Sintaxe: OR < destino > < origem > // soma

<destino>	$\rightarrow 10010011$
<origem>	$\xrightarrow{\text{OR}} 00100100$
<destino>	$\rightarrow 10110111$

Neste exemplo vemos que na operação OR o 1 é o elemento aboriente, e o 0 é o elemento neutro

XOR

Sintaxe: XOR <destino> <origem> // quando não os 2 iguais é 0 senão é 1

<destino>	→ 10 111 011
<origem>	→ 00 001 111
<destino>	→ 01 001 110

Neste exemplo vemos que na operação XOR o 1 complementa, e o 0 é o elemento neutro

NOT

Sintaxe: NOT <destino> // inverter os bits 1→0 e 0→1

<destino>	→ 10 110 101
<destino>	→ 01 001 010

Neste exemplo vemos que na operação XOR o 1 passa a 0 e o 0 passa a 1

P - 2/3/2023

6) Considere os seguintes códigos binários:

a) $1011_{(2)}$

b) $1BA_{(16)}$

c) $10_{(16)}$

d) $657_{(8)}$

e) $FF_{(16)}$

Determine os números inteiros representados em complemento para 2.

a)

$$\begin{array}{cccc} -2^3 & 2^7 & 2^0 \\ 1 & 0 & 1 & 1 \end{array}$$

$$-2 + 0 + 2 + 1 = -5$$

b)

$$1BA_{(16)} = 110111010_{(2)}$$

$$\begin{array}{cccccccccc} -2^8 & 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \end{array}$$
$$-2^8 + 2^7 + 32 + 16 + 8 + 4 + 2 + 0 = -70$$

$$c) \quad 10_{(16)}$$

$$10_{(16)} = 00010000_{(2)}$$

$$\begin{array}{r} \text{-} 2^4 \\ \hline 00010000 \end{array}$$

$$\text{-} 2^4 = f76$$

$$d) \quad 657_{(8)} = \begin{array}{r} -2^8 + 2^6 + 2^4 + 2^3 + 2^2 + 2^0 \\ \hline 110101111_{(2)} \\ -256 + 64 + 16 + 8 + 4 + 1 \end{array}$$

$$e) \quad FF_{(16)} = 11111111_{(2)}$$

$$= 1$$

7)

7) Considere os seguintes códigos binários de 4 bits:

$$a) 1100_{(2)}$$

$$b) 7_{(16)}$$

$$c) 10_{(8)}$$

$$d) 0000_{(2)}$$

$$a) \quad 1100_{(2)} \longrightarrow 11111100_{(2)}$$

\uparrow
4 bits

$$b) \quad 7_{(16)} = 0111 \rightarrow 00000111$$

$$c) \quad 10_{(8)} = 1000 \rightarrow 11111000$$

d)

$$0000 \xrightarrow{(2)} 00000000$$

8)

8) Considere os seguintes códigos binários de 8 bits:

a) D1₍₁₆₎

b) FA₍₁₆₎

c) 0B₍₁₆₎

d) C9₍₁₆₎

a)

$$D1_{(16)} = 11010001 = -47, \text{ Não Representável}$$

$$b) FA_{(16)} = 11111010 = -6 \quad \text{Representável}$$

$$c) 0B_{(16)} = 00001011 = +11 \quad \text{Não Representável}$$

$$d) C9_{(16)} = 11001001 = -55 \quad \text{Não Representável}$$

considerando ES = A020₁₆, BX = FF20₁₆

Enderço:

$$\text{En: } [BX] + 20$$

Enderço efetivo (EE)

$$EE = [BX] + 20 = FF20_{16} + 74 = FF34_{16}$$

Enderço Logico :

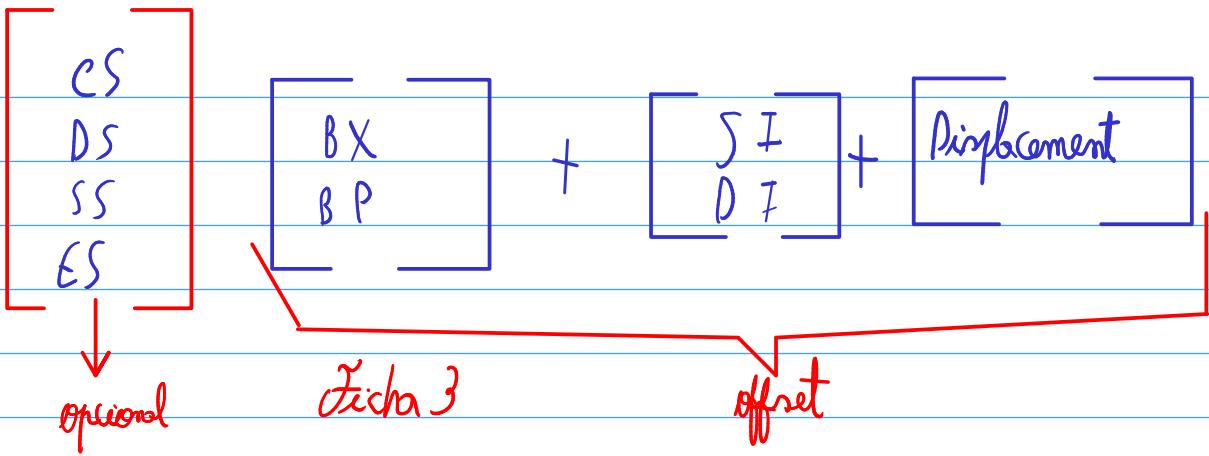
$$A020_{16} : FF34_{16}$$

Enderço em memória :

$$\begin{aligned} EM &= \text{segmento} \times 16_{16} + \text{offset} = \\ &= A0200 + FF34 = B0134_{16} \end{aligned}$$

$$\begin{array}{r} 2 \\ + F \\ \hline 11 \end{array}$$

$$\begin{array}{r} A0200 \\ + FF34 \\ \hline B0134 \end{array}$$



1) Supondo que CS=1000h, DS=1200h, SS=0F00h, ES=14F0h, IP=0010h, BX=120Eh, BP=340Fh, SI=A000h e DI=3000h.

- a) Determine o endereço em memória (endereço físico) da próxima instrução a ser executada.
- b) Determine o endereço efectivo e o endereço em memória dos seguintes operandos:

[2000h]	ES:[2000h]	
[BX]	[BP]	ES:[BX]
[BX][2000h]	[BX+2]	ES:[BX][40h]
[BP][2000h]	[BP+10]	ES:[BP][40h]
[SI]	[DI]	ES:[SI]
[SI][0100h]	[DI+20]	ES:[SI+10h]
[BX][SI]	[BX+DI]	ES:[BX+SI]
[BX][SI][0100h]	[BX+DI+32]	ES:[BX+SI+32]

a) CS [IP]

CS $\Rightarrow 1000_{(16)}$
IP $\Rightarrow 0010_{(16)}$

$$CS \times 10_{(16)} + IP = 1000 + 0010 = 10010_{(16)}$$

\hookrightarrow é o endereço em memória

b)

EE $\Rightarrow 2000_{(16)}$

$$EH \Rightarrow DS = 2000_{(16)} = DS \times 10 + 2000 = 2000 \times 10 + 2000 = 24000_{(16)}$$

$$ES: [2000]_{16}$$

$$SF \rightarrow 2000_{16}$$

$$EM \rightarrow ES = 2000_{16} = ES \times 10 + 2000 = 14F0 \times 10 + 2000 = 14F00 + 2000 = 16F00_{16}$$

$$SE \rightarrow 120E_{16}$$

$$EM \rightarrow 1200 \times 10 + 120E = 12000 + 120E - 120E_{16}$$

$$SE \rightarrow 340F_{16}$$

$$EM \rightarrow SS: BF = SS \times BP = 0F00 \times 10 + 340F = 0F000 + 340F = 340F_{16}$$

$$\begin{array}{r} 0F00 \\ + 340F \\ \hline 1240F \end{array}$$

$$SE \rightarrow 120E + 2000_{16} = 320E_{16}$$

$$EM \rightarrow 1200 + 120E + 2000 = 2520E_{16}$$

$$SE \rightarrow 340E_{16} + 2000 = 540F_{16}$$

$$EM \rightarrow 0F00_{16} \times 10 + 540F_{16} = 0F000 + 540F = 1440F$$

$$\begin{array}{r} 0F00 \\ + 540F \\ \hline 1440F \end{array}$$

$[DP+10]$

$10_{(10)} \rightarrow t_{(16)}$

$$EE \rightarrow 390F+A = 3479_{(16)}$$

1 A
390 F
<hr/>
3479

$$EM \rightarrow SS: 3979 = 0F000 + 3979 = 12979$$

0 F000
3979
<hr/>
12979

$[BX+DI+12]$

$32_{(10)} = 20_{(16)}$

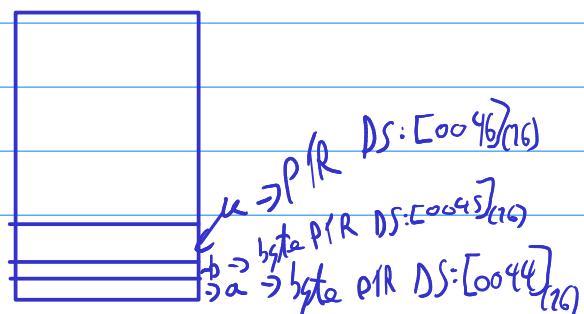
$$EE = 100E + 3000 + 10 = 922E_{16}$$

$$EM = DS \div [BX+DI+12] = 1000 \times 10 + 922E = 1000 + 922E = 1620E_{16}$$

1 - 6/3/2022

Acerca a variação global

char a, b;
int x, y;



C:

Assembly

a = a
b = a
a = b

MOV a, 0
MOV b, a
MOV AL, b
MOV a, AL

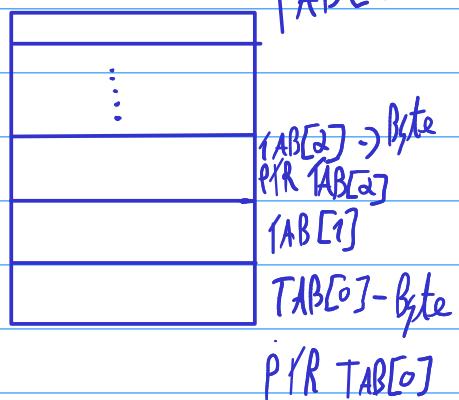
x = -700

y = -70

Labels

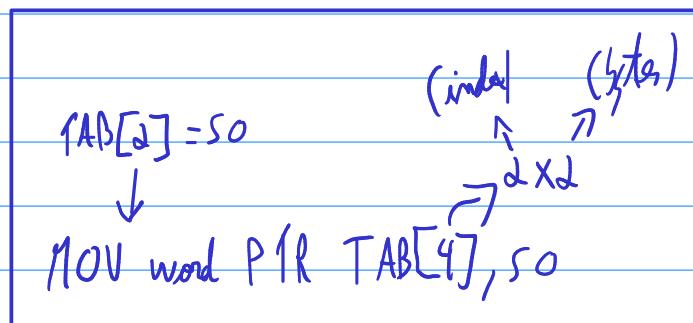
unsigned char TAB[10]

unsigned int i;



int TAB[10];
unsigned int i;

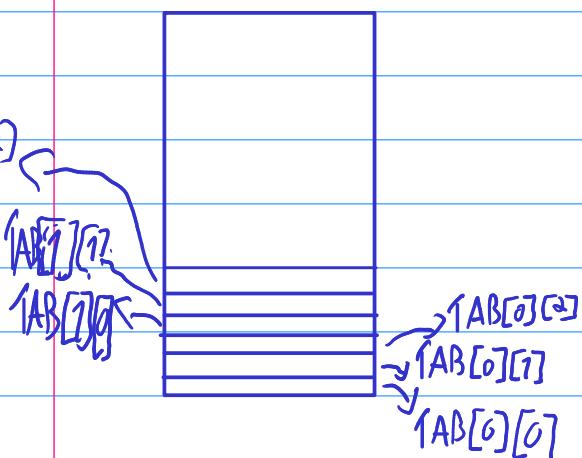
TAB[0] = 50



$i = 3$
 $\text{TAB}[i] = 50;$

MOV $i, 3$
MOV AX, 2
MUL i $\rightarrow ax = ax \times i$
MOV Word PTR TAB[BX], 50

unsigned char TAB[4][3];
unsigned int i, j;

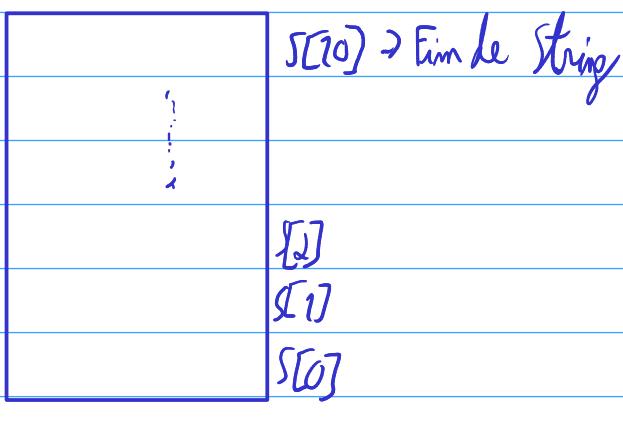


MOV Byte PTR TAB [1x3x1] [0x1]
↑ N° Elementos por linea
↓ N° linea
↓ N° columna

$i = 1, j = 2;$
 $\text{TAB}[0][j] = 50;$

MOV i, 1
MOV j, 2
MOV AX, 1 tamano de cada elemento
MOV BX, 3 n° elementos p/línea
MUL BX AX - AX \times BX
MUL i AX - AX \times i
MOV BX, AX
MOV AX, 1 tamano de cada elemento
MUL j
ADD DX, AX
MOV Byte PTR TAB[BX], 50

Labels - String

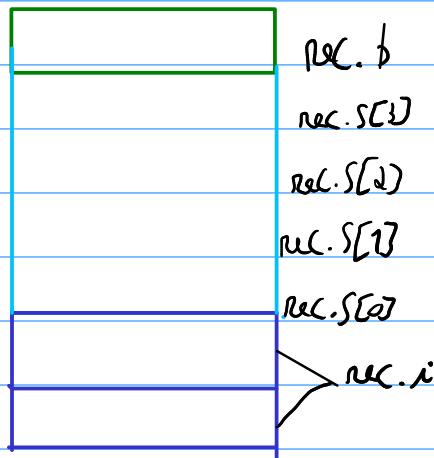


$\text{stcgs}(S, "a") \rightarrow \text{MOV PTR PTR S[0], "a"$
 $\text{MOV Byte PTR S[1], 0}$
 ↗
 const 0

Registers

struct {

int i; // 2 bytes
 char S[4] → 4 bytes } + 5 bytes
 char; → 1 byte
 } rec.



$\text{rec.i} = 20$

$\text{stcgs}(\text{rec.s}, "AC");$

$\text{MOV WORD PTR rec.i, 20};$
 or
 $\text{MOV WORD PTR rec[0], 20}$
 $\text{MOV Byte PTR rec.s, 'A'}$
 $\text{MOV Byte PTR rec.s[1], 'C'}$
 $\text{MOV Byte PTR rec.s[2], 0}$

Abel de los.

-

-

-

-

Globo de Expresión auditiva

Abel

Caro

1 - q3/dad2

Ejercicio 3

(2)

```
const int cte=2;  
unsigned char x,y;  
int xx,yy,zz;
```

a) x=24; MOV X,24
 y=x; MOV AL,X
 . MOV Y,AL

b) x=32; MOV X,32
 y=x+cte; ADD X,cte
 . MOV =AL,X
 . MOV =Y,AL

c) xx=100; MOV XX,100
 yy=200; MOV YY,200
 xx=xx+yy; ADD AX,YY → ADD XX,AX

d) xx=10; MOV XX,10
 yy=20; MOV YY,20
 zz=xx; MOV AX,XX
 xx=yy; MOV ZZ,AX
 yy=zz; MOV AX,YY
 . MOV XX,AX
 . MOV AX,ZZ
 . MOV YY,AX

(3)

```
unsigned char tabbyte[4];
int tabint[3];
unsigned int i, j;
```

a)

```
tabbyte[0]=15;
tabbyte[1]= tabbyte[0]+2;
tabbyte[2]= tabbyte[1]+2;
```

MOV byte P/R tabbyte, 15
MOV AL, tabbyte
ADD AL, 2
MOV tabbyte[1], AL

b)

```
i=0; j=2;
tabint[i]=15;
tabint[i]= tabint[i]+10;
tabint[j]= tabint[i]+20;
```

MOV i, 0 → MOV j, 2
MOV AX, 2
MUL i
MOV BX, AX
MOV word P/R tabint[BX], 15
ADD word P/R tabint[BX], 10
MOV CX, tabint[BX]
ADD CX, 20
MOV AX, 2
MUL j
MOV BX, AX
MOV tabint[BX], CX

(4)

```
int tabint[3][4];
unsigned int i, j;
```

a)

```
tabint[0][1]=15;
tabint[1][2]=20;
```

0x4x2+1x2
↑
MOV word P/R tabint[2], 15
MOV P/R tabint[12], 20
1x4x2+2x2 C

b) `i=2; j=1;` `MOV i, 2`
`tabint[i][j]=50;` `MOV j, 1`

`MOV ax, 2`
`MOV bx, 4`
`MUL bx`
`MUL i`
`MOV bx, ax`
`MOV ax, 2`
`MUL j`
`ADD BX, Ax`
`MOV word P/R tabint[BX], 50`

(7)

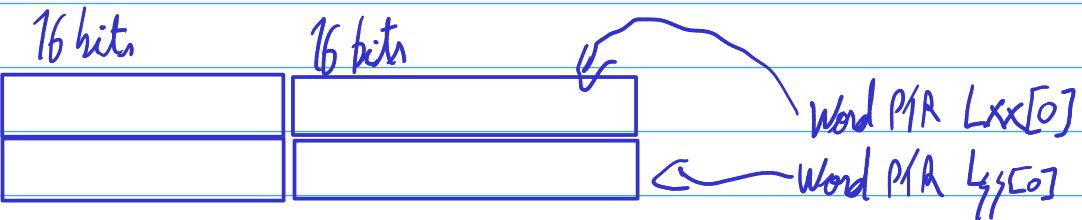
7) Considere as seguintes declarações em C:

```
typedef struct
{
    int i;
    char c;
} rectype;
rectype tabrec[4];
unsigned int i, j;
```

`i=0; j=2;` `MOV i, 0`
`tabRec[i].i=15;` `MOV j, 2`
`tabRec[i].c='A' ;` `MOV ax, 3`
`tabRec[j].i=25;` `MUL i`
`tabRec[j].c='B' ;` `MOV BX, Ax`
`MOV word P/R tabrec[BX], 15`
`MOV byte P/R tabrec[BX][2], 'A'`

Adição de longs

long Lxx, Lyy, Lzz



Soma de 2 longs: $(Lxx = Lyy + Lzz)$

$\text{MOV Ax, Word P/R Lyy}$
 $\text{ADD Ax, Word P/R Lyy}$
 $\text{MOV Word P/R Lxx, Ax}$

} Soma 16 bits Significativos

ADD com ←
 Lyy
 $\text{MOV Ax, Word P/R Lyy[2]}$
 $\text{ADC Ax, Word P/R Lyy[2]}$
 $\text{MOV Word P/R Lxx[2], Ax}$

} Soma com carry, 16 bits + Significativos

Invenção

$$Lxx = Lxx + 1;$$

~~INC Word P/R Lxx~~

~~ADC Word P/R Lxx[2], 0~~

$\text{ADD Word P/R Lxx, 1}$
 $\text{ADC Word P/R Lxx[2], 0}$

Subtração

$$Lxx = Lxx - Lyy$$

MOV Ax Word P/R Lyy
SUB Word P/R Lxx, Ax

MOV

;

Decrementação

$S_{xx} = S_{xx} - 1$

Dec S_{xx}

$FF = FF - 1$

Dec FF

$L_{xx} = L_{xx} - 1$

Conversão de tipos de dados

8 bits \rightarrow 16 bits

MOV AL, X

MOV AH, 0

MOV XX, AX

Existem instruções para fazer tais como:

CBW ← Convert Byte to Word

CWD ← Convert Word to double

Sxx = Sxx;

MOV AL, Sx
CBW
MOV Sxx, Ax

Lxx = Sxx;

MOV Ax, Sxx
CWD

MOV word PTR [Lxx], Ax
MOV word PTR [Lxx+4], Dx

Exercice

MOV AL, 0010h
CBW

AL=0010 0010
Ax=0022h

MOV AL, F0h
CBW

AL=1111 0000

FFFF

MOV Ax, 3922h
CWD

Multiplicação

Instrução para multiplicação de números >0:

MUL

Instrução para multiplicação de números $>0 \wedge <0$:

IMUL

Unsigned char

$x = y \times z;$

MOV AL, y

MUL z

MOV x, AL

char

$sx = sy \times sz;$

MOV AL, sy

IMUL sz

MOV sx, AL

Divisão

Para números >0

DIV

Para números $>0 \wedge <0$

IDIV

;

8) Considere as seguintes instruções de atribuição em C.

~~1~~ X=y+z;

~~4~~ X=y-z-w;

~~7~~ X=-y;

~~2~~ w=x % x;

~~3~~ y=y+1;

~~2~~ X=y+z+w;

~~5~~ X=x- (y+z) ;

~~8~~ X=(x+y) * (y+w) ;

~~11~~ x=((y+x) * (x-w)) / 10;

~~14~~ x=y-1;

~~3~~ X=y-z;

~~6~~ X=x+y*z;

~~9~~ X=y / x;

~~12~~ X=(x*y) % 15;

~~15~~ X=(y-x)-1;

Codifique em assembly 8086 cada uma das instruções anteriores, considerando:

a) x,y,z,w variáveis do tipo unsigned char;

b) x,y,z,w variáveis do tipo int.

a)

MOV AL, ~~5~~

ADD AL, 2

MOV X, AL

11	12
MOV AL, 4	MOV AX
ADD AL, X	INC
MOV BL, X	MOV BL, 15
SUB BL, W	DIV BL
MUL BL	MOV X, AH
WOR BL, 10	
DIV BL	
MOV X, AL	

Opcode	Ax(h)	Bx(h)	Carry
MOV AC, 20	?? 20	????	
ADD AC, 20	61 30	?? ??	
MOV BX, 200		0200	
MOV Word PTR [BX], 20			1
SUB Word PTR [BX], 10			
MOV AH, 89	8920		
SBP AX, 1001	8930-1=892F		0
SBP AX, 601	8940-9=893D		0

$$Ax = 8920$$

$$Bx = 200$$

1) Considere as seguintes declarações em C.

unsigned char x,y; char sx,sy; unsigned int xx; int sxx,syy;

- 1 $xx = xx + y;$
- 2 $xx = x * y;$
- 3 $x = xx / x;$

MOV BL, y

MOV BH, 0

ADD XX, BX

MOV AL, X

MUL L

MOV XX, AX

MOV AX, XX

MOV X

MOV X, AL

- 4 $sxx = syy + sy;$
- 5 $sxx = sx * sy;$
- 6 $sx = sxx \% sx;$

MOV AL, SY

CBW

ADD AX, SYY

MOV SXX, AX

MOV AL, SYY

IDIV SX

MOV SXX, AX

MOV AX, SXX

IDIV SX

MOV SX, AH

2) Considere as seguintes declarações em C.

long xx, yy;

Codifique em assembly 8086 cada uma das instruções seguintes:

- 1 $xx = xx + yy;$

- 2 $xx = yy - xx;$

MOV AX, Word P/R YY
ADD Word P/R XX, AX

MOV AX, Word P/R YY.
SUB AX, Word P/R XX
MOV Word P/R XX, AX

3) Utilizando os operadores lógicos AND, OR, XOR e NOT, codifique em assembly 8086 as seguintes operações:

- a) complementar os bits b₇, b₆, b₅, b₄ da variável A; $\text{XOR}(\text{XOR byte P/R A }) \text{ 0XF0})$
b) colocar a 1 os bits b₅, b₂, b₀ da variável A; $\text{OR}(\text{A}, 37 / 0610 } 0101)$
c) colocar a 0 os bits b₆, b₃, b₁ da variável A; $\text{AND}(\text{A}, 181 / 0111 } 0101)$
d) complementar todos os bits da variável A; $\text{NOT}(\text{A})$
considerando a variável A do tipo char.

4) Considerando:

unsigned int a,b; int x,y,z; long lxx;

Não utilizando as instruções de multiplicação e divisão (MUL, IMUL, DIV, IDIV) codifique as intruções:

- a) y= 10*x; b) y= 7*x; c) z=12*x;
y= x / 2; b=a / 8; x=x / 16;
b= a % 2; b= a % 16; lxx=lxx / 2;
lxx=lxx * 2; lxx=lxx*4; lxx=lxx / 2;

a) MOV Ax, x
SAL Ax, 3
ADD Ax, x
ADD Ax, x
MOV s, Ax

b) MOV Ax, x
SAL Ax, 3
SUB Ax, x
MOV s, Ax

c) MOV Ax, x
SAL Ax, 4
MOV BX, x
SAC BX, 2
SUB AX, BX
MOV Z, Ax

d) MOV Ax, x
SAR Ax, 7
MOV s, Ax

e) MOV Ax, a
SHR Ax, 3
MOV b, Ax

f) SAR x, 4