

O objetivo desta atividade é permitir que o aluno seja capaz de criar um programa em linguagem assembly que manipule caracteres ASCII adequadamente.

### – Tabela ASCII

Observe a tabela ASCII a seguir:

**Tabela ASCII**

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0 000	000	<b>NUL</b> (null)	32	20 040	00000010	&#32;	<b>Space</b>	64	40 100	000000100	&#64;	<b>A</b>	96	60 140	000000100	&#96;	<b>^</b>
1	1 001	001	<b>SOH</b> (start of heading)	33	21 041	000000101	&#33;	<b>!</b>	65	41 101	000000101	&#65;	<b>A</b>	97	61 141	000000101	&#97;	<b>a</b>
2	2 002	002	<b>STX</b> (start of text)	34	22 042	000000110	&#34;	<b>"</b>	66	42 102	000000110	&#66;	<b>B</b>	98	62 142	000000110	&#98;	<b>b</b>
3	3 003	003	<b>ETX</b> (end of text)	35	23 043	000000111	&#35;	<b>#</b>	67	43 103	000000111	&#67;	<b>C</b>	99	63 143	000000111	&#99;	<b>c</b>
4	4 004	004	<b>EOT</b> (end of transmission)	36	24 044	0000001100	&#36;	<b>\$</b>	68	44 104	0000001100	&#68;	<b>D</b>	100	64 144	0000001100	&#100;	<b>d</b>
5	5 005	005	<b>ENQ</b> (enquiry)	37	25 045	0000001101	&#37;	<b>%</b>	69	45 105	0000001101	&#69;	<b>E</b>	101	65 145	0000001101	&#101;	<b>e</b>
6	6 006	006	<b>ACK</b> (acknowledge)	38	26 046	0000001110	&#38;	<b>&amp;</b>	70	46 106	0000001110	&#70;	<b>F</b>	102	66 146	0000001110	&#102;	<b>f</b>
7	7 007	007	<b>BEL</b> (bell)	39	27 047	0000001111	&#39;	<b>'</b>	71	47 107	0000001111	&#71;	<b>G</b>	103	67 147	0000001111	&#103;	<b>g</b>
8	8 010	010	<b>BS</b> (backspace)	40	28 050	0000001000	&#40;	<b>(</b>	72	48 110	0000001000	&#72;	<b>H</b>	104	68 150	0000001000	&#104;	<b>h</b>
9	9 011	011	<b>TAB</b> (horizontal tab)	41	29 051	0000001001	&#41;	<b>)</b>	73	49 111	0000001001	&#73;	<b>I</b>	105	69 151	0000001001	&#105;	<b>i</b>
10	A 012	012	<b>LF</b> (NL line feed, new line)	42	2A 052	0000001010	&#42;	<b>*</b>	74	4A 112	0000001010	&#74;	<b>J</b>	106	6A 152	0000001010	&#106;	<b>j</b>
11	B 013	013	<b>VT</b> (vertical tab)	43	2B 053	0000001011	&#43;	<b>+</b>	75	4B 113	0000001011	&#75;	<b>K</b>	107	6B 153	0000001011	&#107;	<b>k</b>
12	C 014	014	<b>FF</b> (NP form feed, new page)	44	2C 054	00000010100	&#44;	<b>,</b>	76	4C 114	00000010100	&#76;	<b>L</b>	108	6C 154	00000010100	&#108;	<b>l</b>
13	D 015	015	<b>CR</b> (carriage return)	45	2D 055	00000010101	&#45;	<b>-</b>	77	4D 115	00000010101	&#77;	<b>M</b>	109	6D 155	00000010101	&#109;	<b>m</b>
14	E 016	016	<b>SO</b> (shift out)	46	2E 056	00000010110	&#46;	<b>.</b>	78	4E 116	00000010110	&#78;	<b>N</b>	110	6E 156	00000010110	&#110;	<b>n</b>
15	F 017	017	<b>SI</b> (shift in)	47	2F 057	00000010111	&#47;	<b>/</b>	79	4F 117	00000010111	&#79;	<b>O</b>	111	6F 157	00000010111	&#111;	<b>o</b>
16	10 020	020	<b>DLE</b> (data link escape)	48	30 060	00000011000	&#48;	<b>0</b>	80	50 120	00000011000	&#80;	<b>P</b>	112	70 160	00000011000	&#112;	<b>p</b>
17	11 021	021	<b>DC1</b> (device control 1)	49	31 061	00000011001	&#49;	<b>1</b>	81	51 121	00000011001	&#81;	<b>Q</b>	113	71 161	00000011001	&#113;	<b>q</b>
18	12 022	022	<b>DC2</b> (device control 2)	50	32 062	00000011010	&#50;	<b>2</b>	82	52 122	00000011010	&#82;	<b>R</b>	114	72 162	00000011010	&#114;	<b>r</b>
19	13 023	023	<b>DC3</b> (device control 3)	51	33 063	00000011011	&#51;	<b>3</b>	83	53 123	00000011011	&#83;	<b>S</b>	115	73 163	00000011011	&#115;	<b>s</b>
20	14 024	024	<b>DC4</b> (device control 4)	52	34 064	000000110100	&#52;	<b>4</b>	84	54 124	000000110100	&#84;	<b>T</b>	116	74 164	000000110100	&#116;	<b>t</b>
21	15 025	025	<b>NAK</b> (negative acknowledge)	53	35 065	000000110101	&#53;	<b>5</b>	85	55 125	000000110101	&#85;	<b>U</b>	117	75 165	000000110101	&#117;	<b>u</b>
22	16 026	026	<b>SYN</b> (synchronous idle)	54	36 066	000000110110	&#54;	<b>6</b>	86	56 126	000000110110	&#86;	<b>V</b>	118	76 166	000000110110	&#118;	<b>v</b>
23	17 027	027	<b>ETB</b> (end of trans. block)	55	37 067	000000110111	&#55;	<b>7</b>	87	57 127	000000110111	&#87;	<b>W</b>	119	77 167	000000110111	&#119;	<b>w</b>
24	18 030	030	<b>CAN</b> (cancel)	56	38 070	0000001101100	&#56;	<b>8</b>	88	58 130	0000001101100	&#88;	<b>X</b>	120	78 170	0000001101100	&#120;	<b>x</b>
25	19 031	031	<b>EM</b> (end of medium)	57	39 071	0000001101101	&#57;	<b>9</b>	89	59 131	0000001101101	&#89;	<b>Y</b>	121	79 171	0000001101101	&#121;	<b>y</b>
26	1A 032	032	<b>SUB</b> (substitute)	58	3A 072	0000001101110	&#58;	<b>:</b>	90	5A 132	0000001101110	&#90;	<b>Z</b>	122	7A 172	0000001101110	&#122;	<b>z</b>
27	1B 033	033	<b>ESC</b> (escape)	59	3B 073	0000001101111	&#59;	<b>;</b>	91	5B 133	0000001101111	&#91;	<b>[</b>	123	7B 173	0000001101111	&#123;	<b>{</b>
28	1C 034	034	<b>FS</b> (file separator)	60	3C 074	00000011011000	&#60;	<b>&lt;</b>	92	5C 134	00000011011000	&#92;	<b>\</b>	124	7C 174	00000011011000	&#124;	<b> </b>
29	1D 035	035	<b>GS</b> (group separator)	61	3D 075	00000011011001	&#61;	<b>=</b>	93	5D 135	00000011011001	&#93;	<b>]</b>	125	7D 175	00000011011001	&#125;	<b>}</b>
30	1E 036	036	<b>RS</b> (record separator)	62	3E 076	00000011011010	&#62;	<b>&gt;</b>	94	5E 136	00000011011010	&#94;	<b>^</b>	126	7E 176	00000011011010	&#126;	<b>~</b>
31	1F 037	037	<b>US</b> (unit separator)	63	3F 077	00000011011011	&#63;	<b>?</b>	95	5F 137	00000011011011	&#95;	<b>-</b>	127	7F 177	00000011011011	&#127;	<b>DEL</b>

Source: [www.LookupTables.com](http://www.LookupTables.com)

A tabela ASCII tem a função de mapear **caracteres** a **códigos de 8 bits** (geralmente representados em seu valor binário, decimal ou hexadecimal).

Na tabela acima podemos ver, por exemplo, que o código ASCII do caractere “A” corresponde ao valor **65** (em decimal), ou **41h** (em hexadecimal), que corresponde aos 8 bits **01000001b**.

Da mesma forma, podemos observar que o código ASCII do caractere “a” corresponde ao valor **97** (em decimal), ou **61h** (em hexadecimal), que corresponde aos 8 bits **01100001b**.

Já o código ASCII do caractere “1” corresponde ao valor **49** (em decimal), ou **31h** (em hexadecimal), que corresponde aos 8 bits **00110001b**.

Observe que nos programas em linguagem assembly:

- **caracteres** são representados entre **aspas simples** ou **aspas duplas**. Ex: ‘A’ ou “A”
- valores **decimais** são representados por números que podem opcionalmente ser terminados com a **letra D (ou d)**. Ex: 65 ou 65D ou 65d
- valores **hexadecimais** são representados por números sempre terminados com a **letra H (ou h)**. Ex: 41H ou 41h

- valores **binários** são representados por números sempre terminados com a **letra B (ou b)**. Ex: 01000001B ou 01000001b

### **Parte 1 - Programa: ATIV02\_1.asm**

Faça um programa em Assembly x86, que leia uma letra minúscula e a transforme em letra maiúscula.

O programa deve ter as seguintes mensagens:

Digite uma letra minúscula: a

A letra maiúscula correspondente eh: A

### **Parte 2 – DEBUGGER - - Programa: ATIV02\_2.asm**

Usando o programa abaixo, use o debugger para preencher a tabela abaixo, com os valores correspondentes após a execução de cada instrução.

	AX	BX	CX	DX	SI	DI	DS	CS	IP
MOV AX,@DATA									
MOV DS,AX									
MOV AH,09									
MOV DX, OFFSET MSG									
INT 21H									
MOV AH,4CH									
INT 21H									

```

title exemplo
.model small
.data
    msg db 'ola bem vindo',13,10,'$'
.code
main proc
    mov ax,@data ; inicializar DS com endereco do segmento de dados
    mov ds,ax

    mov ah,09      ; funcao 09 impirme um string
    mov dx, offset msg ; endereco incial do string
    int 21h        ; executa a funcao 09

    mov ah,4ch      ; fim do programa
    int 21h        ; retorno ao SO
main endp
end main

```

### **Parte 3- Programa: ATIV2\_3.asm**

Crie um programa em linguagem assembly chamado **ATIV2\_3.asm** que exibe uma mensagem na tela solicitando ao usuário que digite um primeiro número (de 0 a 9), lê o caractere digitado

do teclado, exibe uma mensagem na linha seguinte solicitando ao usuário que digite um segundo número (de 0 a 9), lê o caractere digitado do teclado, exibe uma mensagem na linha seguinte informando qual o valor da soma do primeiro com o segundo número e exibe o caractere contendo o resultado da soma.

**OBS:** A soma dos dois números nunca deve ultrapassar o valor 9, ou seja, o usuário sempre deve digitar dois números cuja soma seja menor ou igual a 9.

**Exemplo:**

Digite um primeiro numero: **2**

Digite um segundo numero: **5**

A soma dos dois numeros eh: **7**

**ENTREGA**

Cada aluno deve postar os arquivos **ATIV2\_1.asm**, **ATIV2\_2.asm**, **ATIV2\_3.asm** e **arquivo pdf com a tabela CANVAS**.