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

Source: 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 me 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 ; endereço 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.