Lab 3

In this session, you are expected to write assembly language programs for x86 processor and use MASM Assembler to assemble the code.

In this lab, you are expected to know mov instructions, comparison instructions, unconditional Jump instruction and conditional jump instructions before you complete the lab tasks.

A brief description of the instructions is given below.

Comparison Instruction:

The comparison instruction (CMP) is a subtraction that changes only the flag bits; the destination operand never changes. A comparison is useful for checking the contents of a register or a memory location against another value. A CMP is normally followed by a conditional jump instruction, which tests the condition of the flag bits.

Some examples of CMP instructions are given below.

Assembly Language	*Operation*
CMP CL,BL	CL – BL
CMP AX,SP	AX – SP
CMP AX,2000H	AX – 2000H
CMP [DI],CH	CH subtracts from the byte contents of the data segment memory location addressed by DI
CMP CL,[BP]	The byte contents of the stack segment memory location addressed by BP subtracts from CL
CMP AH,TEMP	The byte contents of data segment memory location TEMP subtracts from AH
CMP DI,TEMP[BX]	The word contents of the data segment memory location addressed by TEMP plus BX subtracts from DI

Conditional jump Instructions:
Unconditional Jump instruction (JMP) changes the control flow of the instruction to the offset mentioned as a part of instruction. [You may refer to text book barry b brey for details regarding JMP instruction]

Conditional jump instructions are always short jumps in the 8086 through the 80286 microprocessors. This limits the range of the jump to within +127 bytes and -128 bytes from the location following the conditional jump. The conditional jump instructions test the following flag bits: sign (S), zero (Z), carry (C), parity (P), and overflow (0). If the condition under test is true, a branch to the label associated with the jump instruction occurs. If the condition is false, the next sequential step in the program executes.

Assembly Language	*Tested Condition*	*Operation*
JA	Z = 0 and C = 0	Jump if above
JAE	C = 0	Jump if above or equal
JB	C = 1	Jump if below
JBE	Z = 1 or C = 1	Jump if below or equal
JC	C = 1	Jump if carry
JE or JZ	Z = 1	Jump if equal or jump if zero
JG	Z = 0 and S = 0	Jump if greater than
JGE	S = 0	Jump if greater than or equal
JL	S != O	Jump if less than
JLE	Z = 1 or S != O	Jump if less than or equal
JNC	C = 0	Jump if no carry
JNE or JNZ	Z = 0	Jump if not equal or jump if not zero
JNO	O = 0	Jump if no overflow
JNS	S = 0	Jump if no sign (positive)
JNP or JPO	P = 0	Jump if no parity or jump if parity odd
JO	O = 1	Jump if overflow
JP or JPE	P = 1	Jump if parity or jump if parity even
JS	S = 1	Jump if sign (negative)
JCXZ	CX = 0	Jump if CX is zero

Flag register and the mapping of flag bits in MASM is as given below

FLAG	CLEAR (0) SYMBOL	SET (1) SYMBOL
Overflow Flag	NV (no overflow)	OV (overflow)
Direction Flag *	UP (up)	DN (down)
Interrupt Flag *	DI (disable interrupt)	EI (enable interrupts)
Sign Flag	PL (plus)	NG (negative)
Zero Flag	NZ (nonzero)	ZR (zero)
Auxiliary Carry Flag	NA (no auxiliary carry)	AC (auxiliary carry)
Parity Flag	PO (odd parity)	PE (even parity)
Carry Flag	NC (no carry)	CY (carry)

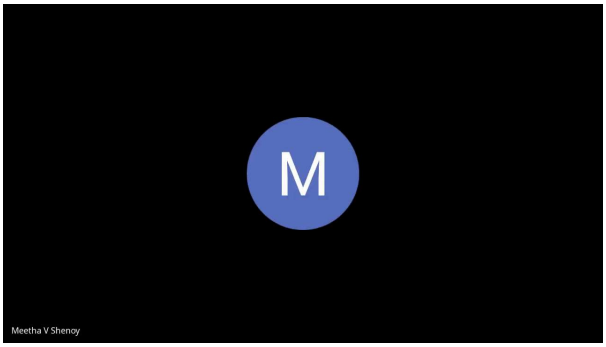
Sample code for practice.

Assume that a string is stored from a location labeled 'DATA1'.
e g: DATA1 db 'MicroProcessor'
The number of characters in the string is stored in the location labeled 'count'
Write an assembly language program (ALP) to count the no of small letters and capital letters in the string and store the count in two locations labelled 'sno' and 'cno' respectively.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Video demonstration is provided below:



In case the code is not working as expected, trace the instruction line by line- check how the flag bits are updated and debug the code.

Task for P1/P2 Lab sessions:

Task 1:

Assume that a string is stored from a location labeled 'DATA1'.

e g: DATA1 db 'MicroProcessoR'

The number of characters in the string is stored in the location labeled 'count'

Write an assembly language program (ALP) to inspect the string and change the capital letters in the string to small letters. [You may use ADD instruction]

Show the contents of the code, before and after running the code.

Task 2:

Write an ALP that will search for 10 byte locations in memory, whose starting location is labelled as DATA1 and replace any occurrence of 00 with FFH

Note: for hex numbers such as FFh etc, use 0FFh in code.

Last modified: Monday, 3 February 2025, 7:16 AM

◄ Lab-2

Jump to...

Lab4 ►