Experiment No 4

Indexing and Data Manipulation

Introduction:

In this experiment you will be introduced to data transfer and arithmetic instructions.
You will also deal with indexing, and array manipulation.

Objectives:

1- Basic arithmetic instructions
2- Use of the ASCII table.
3- Manipulation of arrays

ASCII code Table:

The ASCII table is a double entry table, which contains all alphanumeric characters and symbols. Each symbol, or character, has its own ASCII code. This code can either be in decimal or hexadecimal format. The code of a given character is found by concatenating the column number with the row number, if the code is to be expressed in hexadecimal. The row number is to be the least significant. For the same code to be expressed in decimal, the row number is added to the column number (See example given below).

As an example, the symbol' $' is at the intersection of row 4 and column 2, therefore its ASCII code is 24H. The decimal equivalent of this code can be found by adding 4 to 32, which yields 36.

The following tables show the ASCII codes (Table 4.1), and examples on the use of the ASCII table (Table 4.2), and how to calculate the ASCII codes for different characters and symbols.

DECIMAL VALUE	HEXA DECIMAL VALUE	0	16	32	48	64	80	96	112
		0	1	2	3	4	5	6	7
0	0	BLANK (NULL)	►	BLANK (SPACE)	0	@	P	‘	p
1	1	☺	◄	!	1	A	Q	a	q
2	2	☻	↕	"	2	B	R	b	r
3	3	♥	!!	#	3	C	S	c	s
4	4	♦	¶	$	4	D	T	d	t
5	5	♣	§	%	5	E	U	e	u
6	6	♠	▬	&	6	F	V	f	v
7	7	•	↨	’	7	G	W	g	w
8	8	◘	↑	(	8	H	X	h	x
9	9	○	↓	)	9	I	Y	i	y
10	A	◙	→	*	:	J	Z	j	z
11	B	♂	←	+	;	K	[	k	{
12	C	♀	∟	,	<	L	\	l	¦
13	D	♪	↔	—	=	M	]	m	}
14	E	♫	▲	.	>	N	^	n	~
15	F	☼	▼	/	?	O	_	o	△

DECIMAL VALUE	HEXA DECIMAL VALUE	128	144	160	176	192	208	224	240
		8	9	A	B	C	D	E	F
0	0	Ç	É	á	⋮	└	═	∝	≡
1	1	ü	æ	í	▒	┴	╤	β	±
2	2	é	Æ	ó	▓	┬	╥	Γ	≥
3	3	â	ô	ú	│	├	╙	π	≤
4	4	ä	ö	ñ	┤	─	╘	Σ	∫
5	5	à	ò	Ñ	╡	┼	╒	σ	⌠
6	6	å	û	a	╢	╞	╓	μ	÷
7	7	ç	ù	o	╖	╟	╫	τ	≈
8	8	ê	ÿ	¿	╕	╚	╪	Φ	°
9	9	ë	Ö	⌐	╣	╔	┘	Θ	•
10	A	è	Ü	¬	║	╩	┌	Ω	·
11	B	ï	¢	½	╗	╦	■	δ	√
12	C	î	£	¼	╝	╠	▐	∞	n
13	D	ì	¥	¡	╜	═	▌	φ	2
14	E	Ä	₧	«	╛	╬	▀	∈	■
15	F	Å	ƒ	»	┐	╧	α	∩	BLANK FF

Table 4.1: ASCII Table

Character	Column #	Row #	Code (H)	Code (10)
a	6	1	61	96+1=97
A	4	1	41	64 + 1 = 65
β	E	1	E1	224 + 1 = 225
%	2	5	25	32 + 5 = 37

Table 4.2: Examples on the use of the ASCII table

The DB and DW directives are respectively used to declare a variable of size byte or word. The following declaration defines a variable X of size byte and assigns it the value 10H.

```
X  DB      10H
```

Identically the following will define a variable of size word, and assigns it the value

```
13EFH:        Y              DW     13EFH
```

The DUP directive may be used to reserve more than one consecutive data item and initialize reserved items to the same value. For example, the instruction:

```
ByteArray          DB 100 DUP(0)
```

Instructs the assembler to reserve an array of 100 bytes, and initializes each byte to the value zero. If the "0" in the above declaration is replaced with "?", the assembler will not initialize the bytes of the array to any value.

To access the different elements of an array, we use one of the following addressing modes (See Experiment # 3).

- Based addressing mode.
- Indexed addressing mode.
- Based-Indexed addressing mode.

The Based-Indexed addressing mode may be used to access a two-dimensional array. Here are examples of each case.

```
Array1 DB 0,1,2,3,4,5,6,7,8,9
Array2 DB 10 DUP(0)
Array3 DB 11,12,13,21,22,23,31,32,33
RowSize EQU 3
```

Based addressing mode:

```
MOV BX, OFFSET Array1          ; Address Array1
MOV AL,[BX+4]                  ; Access 5th element of Array1
```

Indexed addressing mode:

```
MOV DI, OFFSET Array2          ; Address Array2
MOV [DI+6],AL                  ; Copy to 7th element of Array2
MOV SI,3
MOV Array2[SI],AL              ;Copy to 4th element of Array2
```

Based-Indexed addressing mode:

```
MOV BX, OFFSET Array3          ; Address Array3
MOV SI,1*RowSize               ; Beginning of 2nd row
MOV DI,2*RowSize               ; Beginning ofrd
MOV AL, [BX+SI+1]              ; Access 2nd element of  2nd row
MOV [BX+DI+2],AL               ; Access 3rd element of  3rd row
```

Remark:

Notice that row R, has index (R-1), and element n has index (n-1).

Pre Lab Work:

1. Study programs 4.1 and 4.2, and review the material related to indexing and data manipulation.
2. Write both programs and see how program 4.1 manipulates the variables in internal registers, and how program 4.2 uses memory for the same purpose.
3. Modify program 4.1 so that it adds two numbers of two digits each. Use only registers, and make sure to take care of the carry when adding the two most significant digits. Call this program 4.3.
   Note: In this case try to understand how the program reads the numbers and how it manipulates them. This will help you in writing your program.
   As a **hint**, one should know that numbers are given in decimal to the program.

4. Modify program 4.3 so that it can handle numbers of four digits. Use arrays in this case. Call this program 4.4.
5. Bring your work to the lab.

Lab Work:

1- Assemble, Link and Run program 1.
2- How many digits can you enter each time? Explain this.
3- What happens when the sum exceeds 9? Explain this.
4- Assemble, Link and Run program 2. Dress a table and show some inputs and outputs.
5- Repeat step 4 with program 3.
6- Show all your work to the instructor.
7- Submit all your work at the end of the lab session.

Lab Assignment:

Write a program that prompts the user to enter two numbers of 4 digits each. Then the program calculates the quotient and remainder of the division of the two numbers. The two numbers are entered as two arrays of size four (4).

```
     TITLE "PROGRAM 1 EXPERIMENT 4"

; This program reads two numbers from the keyboard and
; gives their sum. This program uses internal registers
; to store the variables.

.MODEL SMALL
.STACK  200
.DATA
       CRLF    DB 0DH,0AH,'$'
       PROMPT1 DB 'Enter the first  positive integer: ','$'
       PROMPT2 DB 'Enter the second positive integer: ','$'
       PROMPT3 DB 'The sum of the two numbers is: ','$'
.CODE
.STARTUP
LEA DX,PROMPT1   ;DISPLAY PROMPT1
       MOV AH,09H
       INT 21H

       MOV AH,01H      ;READ FIRST NUMBER
       INT 21H
       SUB AL,30H     ;Convert character to number
      MOV CL,AL  SAVE THE NUMBER IN CL

     MOV AH,01H      ;READ SECOND NUMBER
       INT 21H
       SUB AL,30H ;Convert character to number
   ADD AL,CL   ;PERFORM ADDITION AND SAVE
           RESULT IN CL
   MOV CL,AL
  ADD CL,30H ;CONVERT DIGIT TO CHARACTER


       LEA DX,CRLF     ;MOVE CURSOR TO NEXT LINE
       MOV AH,09H
       INT 21H
            LEA DX,PROMPT3       ;DISPLAY PROMPT3
       MOV AH,09H
       INT 21H

            MOV DL,CL
            MOV AH,02H
       INT 21H
.EXIT
END
```

Program 2:

```
        TITLE "PROGRAM 2 EXPERIMENT 4"

; This program reads two numbers from the keyboard and
; displays their sum. This program uses the memory to
; store the variables.

.MODEL SMALL
.STACK  200
.DATA
      CRLF                DB 0DH,0AH,'$'
      PROMPT1   DB 'Enter the first  positive integer: ','$'
      PROMPT2   DB 'Enter the second positive integer: ','$'
      PROMPT3   DB 'The sum of the two numbers is: ','$'
            NUM1          DB ?
            NUM2          DB ?
            RES           DB ?
.CODE
.STARTUP
      LEA DX,PROMPT1      ;DISPLAY PROMPT1
      MOV AH,09H
      INT 21H
      MOV AH,01H           ;READ FIRST NUMBER
      INT 21H
       SUB AL,30H          ;Convert character to number
       MOV NUM1,ALSAVE NUM1
       LEA DX,CRLF         ;MOVE CURSOR TO NEXT LINE

       MOV AH,09H
            INT 21H

       LEA DX,PROMPT2      ;DISPLAY PROMPT2
            MOV AH,09H
            INT 21H
```

```
        MOV AH,01H          ;READ SECOND NUMBER

            INT 21H
          SUB AL,30H          ;Convert character to number

        MOV NUM2,AL ;SAVE NUM2

        ADD AL,NUM1         ;PERFORM ADDITION

        MOV RES,AL          ;SAVE RESULT IN RES

        LEA DX,CRLF         ;MOVE CURSOR TO NEXT LINE
        MOV AH,09H
            INT 21H
LEA DX,PROMPT3              ;DISPLAY PROMPT3
            MOV AH,09H
            INT 21H
        ;DISPLAY SUM

        MOV DL,RES          ;RETREIVE RES FROM MEMORY
       ADD DL,30H           ;CONVERT DIGIT TO CHARACTER

             MOV AH,02H
       INT 21H
   .EXIT

   END
```