EXPERIMENT ELEVEN:
LOOKUP TABLES

INTRODUCTION

Lookup tables are used for data conversion and also to access data that are in tabular form. Tabular data includes both numeric and alphabetic data in the form of character strings as well as in binary form. This experiment uses lookup tables to illustrate conversion from one code to another and also to lookup character string data. It also illustrates how a jump table accesses various software as required by some applications.

OBJECTIVES

1. Use a lookup table to convert from one numeric code to another.
2. Locate and display character string data located in a lookup table.
3. Use a lookup to obtain jump or call addresses for computed jumps or calls.
4. Search a table for information.

PROCEDURE

A lookup table is a group of data organized so it is easily accessed without searching. An example lookup table may contain information such as 7 segment code for numeric displays. Or it may contain EBCDIC (extended binary coded decimal interchange code) that is converted to ASCII through a lookup table. Neither of these codes can be converted by using a numeric technique such as conversion from ASCII to BCD (subtract 30H).

Suppose that alphabetic data must be encrypted as a different code. A lookup table could be used to store the encryption codes. Such a lookup table and a procedure that converts ASCII to the encrypted code appears in Example 11-1. Note that the ASCII-coded character in AL is converted to an entry from the table by using the XLAT (translate) instruction. The XLAT instruction adds the contents of AL to the contents of BX to form an address in the table. It then transfers a copy of the data at that address to the AL register. This software uses two encryption tables: the UTAB table encrypts uppercase letters and the LTAB table encrypts lowercase letters. The letters stored in the tables can be changed for different encryption codes. They could also be randomized occasionally for additional security. Of course this example may only be practical for developing anagrams or word scrambles for the newspaper. The XLAT instruction uses a segment override prefix to override the default segment (DS).
Example 11-1

**UTAB** DB 'MNBVCXZLKJHGFDSAPOIUYTREWQ'
**LTAB** DB 'bgtnhmjukilopvfrcdexswzaq'

**ENCRP** PROC NEAR ; encrypts AL to code in UTAB and LTAB
**.IFDEF**
**.IFDEF**
**MOV** BX, OFFSET UTAB
**XLAT** CS:UTAB
**RETURN**
**.ENDIF**

**.IFDEF**
**MOV** BX, OFFSET LTAB
**XLAT** CS:LTAB
**RETURN**
**.ENDIF**

**ENCRP** ENDP

**STEP 1:** Develop a lookup table that contains the uppercase letters A through Z and develop a procedure that tests AL for a lowercase letter. Next develop a procedure that examines AL and if it contains a lowercase letter use the lookup table to convert it to uppercase. Next write a program that reads a key from the keyboard using DOS INT 21H function 7 (no echo) and converts lowercase letters to uppercase letters and uppercase letters to lowercase letters using the procedure and lookup table developed here.

The lookup table presented thus far uses **XLAT** to convert from one 8 bit code to another. Suppose that a lookup table converts a single 8 bit code into a 16 bit code. Such a table appears in Example 11-2. This lookup table contains the squares of the numbers between 0 and 25. This example also lists a procedure that squares the number in AL and returns with it's square in AX via the lookup table. Although this can also be accomplished with a multiply instruction. The lookup table technique is much faster with most microprocessors. It can also be adapted to hold other numeric information.

Example 11-2

**STAB** DW 0,1,4,9,16,25,36,49,64,81,100
**STAB** DW 121,144,169,196,225,256,289,324,361,400
**STAB** DW 441,484,529,576,625

**SQAL** PROC NEAR
**MOV** AH, 0 ; clear AH
**MOV** SI, OFFSET STAB ; address STAB
**ADD** AX, AX ; double AX to address word entry
**ADD** SI, AX ; add AX to SI to generate address
**MOV** AX, CS:[SI] ; get square of AL
**RETURN**

**SQAL** ENDP
**STEP 2:** Develop a lookup table that contains the number 0 through 9 raised to the fourth power. For example, 2 raised to the fourth power is $2^4 = 2 \times 2 \times 2 \times 2 = 16$. Using the macro developed in Experiment 10 (_NUM 10) to display data in decimal format, develop a program that displays the numbers 0 through 9 raised to the fourth power. The video display must contain the numbers 0 through 9 in the leftmost column with an equal sign and the number raised to the fourth power next to each one.

Another, and probably more useful lookup table is one that stores ASCII-coded character strings. Example 11-3 lists a lookup table that contains the ASCII-coded character string for the days of the week. It also contains a procedure that reads the date from DOS and displays the day of the week. Notice how the lookup table (DTAB) contains the addresses of the character strings for Sunday through Saturday.

**Example 11-3**

```asm
CODE SEGMENT 'code' ; indicate start of code segment
ASSUME CS:CODE

DTAB DW D0,D1,D2,D3,D4,D5,D6
DO DB 'Sunday$'
D1 DB 'Monday$'
D2 DB 'Tuesday$'
D3 DB 'Wednesday$'
D4 DB 'Thursday$'
D$ DB 'Friday$'
D6 DB 'Saturday$

MAIN PROC FAR

    MOV AX,CS ; make DS = CS
    MOV DS,AX
    MOV AH,2AH ; get date
    INT 21H
    MOV SI,OFFSET DTAB
    MOV AH,0
    ADD AX,AX
    ADD SI,AX
    MOV DX,[SI] ; get string address
    MOV AH,9 ; display day of week
    INT 21H
    MOV AX,4C00H ; exit to DOS
    INT 21H

MAIN ENDP

CODE ENDS

END MAIN
```

**STEP 3:** Enter the program from Example 11-3, assemble it and execute it. You should see today's day displayed.
STEP 4: In addition to obtaining the day of the week, DOS INT 21H, function 2AH also returns the month. Modify the program in Example 11-3 so it displays the day of the week and the month as Monday, March. This will require a second lookup table to store the names of the 12 months. Be careful, because the first month is returned from function 2AH as 1 not zero.

Another way to use a lookup table, is to store addresses other than those of character strings in the table. These other addresses may be used with a JMP or CALL instruction to access different sections of a program or different procedures.

The program listed in Example 11-4 asks if the user wishes to exit to DOS or continues to ask the same question. The location jumped to in this program is accessed via a lookup table that contains the addresses for the start of the program or the point at which the exit to DOS occurs. Note how the .IF statement is used to test for a valid character of 0 or 1. Also notice that the labels MAIN1 and EXIT contain double colons (::) instead of the customary single colon (:). A label is local if it contains a single colon and global if it contains a double colon. A local label is local to the procedure it is used in or the segment it is used in. In this case the label MAIN2: is local only to the MAIN procedure and not to the CODE segment. Because of this the labels MAIN1 and EXIT are made global so they can be used from the CODE segment to access memory locations MAIN1 and EXIT in the LTAB statement.

Example 11-4

```
CODE SEGMENT 'code' ;indicate start of code segment
ASSUME CS:CODE ;identify CODE as CS

LTAB DW MAIN1,EXIT
MES DB 13,10
DB 'Enter 0 to continue and 1 to exit to DOS: $'

MAIN PROC FAR ;start main procedure
MOV AX,CS ;make DS = CS ,
MOV DS,AX

MAIN1::
MOV AH,9 ;display MES
MOV DX,OFFSET MES
INT 21H

MAIN2:
MOV AH,7 ;read without echo
INT 21H

.sub AL < '0' || AL > '1'
JMP MAIN2
.endif

SUB AL,30H ;convert to binary
MOV BX,OFFSET LTAB
MOV AH,0
ADD AX,AX ;double AX
ADD BX,AX
```
STEP 5: Using a jump type lookup table, write a program that displays your name if a 1 is typed, displays your address if a 2 is typed, and displays your telephone number if a 3 is typed. If a 0 is typed, the program exits to DOS. After displaying your name, etc. the program should redisplay the menu. Make sure that the program accesses four different subprograms to accomplish these four tasks. If any other character is typed, it must be ignored. The prompt and menu displayed by the program must appear as follows:

```
Main Menu
0 - exit to DOS
1 - name
2 - address
3 - telephone number

Enter choice:
```

The jump lookup table is accessed by the `JMP WORD PTR [BX]` instruction in Example 11-4. This same type of program can be modified to call various procedures by changing the `JMP` instruction to a `CALL` instruction. If the `CALL` instruction is used, don't forget that a return will eventually return to the step following the call. This does not happen with the `JMP` instruction.

All the table lookup techniques discussed thus far have used direct lookup. The next portion of this experiment details tables searches. A table search is accomplished by using either the `SCAS` instruction to search for a byte, word, or double word entry, or the `CMPS` instruction to search for a multiple byte entry. The type of search is dictated by the data stored in the table.

Suppose that the data encrypted back in Example 11-1 must be decrypted. This can be accomplished by using the `SCASB` instruction to search through an uppercase and a lowercase table to locate an entry. When the entry is found, its relative position in the table is found in the DI register. Recall that `SCASB` searches an area of memory in the extra segment address by the DI register. See Example 11-5 for a procedure that decrypts the information encrypted by Example 11-1.

Example 11-5

```
UTAB DB 'MNBVCXZLKJHGDSAPQIOUYTREWQ'
LTAB DB 'bgtnymjukilopvfrcdswzaq'
```
This example uses the `SCASB` instruction to search the table. The `SCASB` instruction modifies the contents of the DI register which points to the byte following the character found in one of the tables. Notice how the number in DI is converted to either an uppercase or lowercase letter by subtracting the start of the table and then by adding either a 40H or 60H to the value in DI. Also notice how the `REPNE` prefix is used to repeat the `SCASB` instruction while a not-equal results from the comparison performed by `SCASB`.

**STEP 6:** Using the `SCASB` instruction (it addresses data in the extra segment addressed by DI), write a program that searches through memory beginning at location F800:0000 for the letter M. Your program is to display the offset address of this location using the `_NUM 16` macro developed in Experiment 10.

The offset address is: ____________________

**QUESTIONS**

11-1. Using the XLAT instruction, develop a table lookup procedure that converts the contents of AL (0-F) into the ASCII-coded characters for 0-9 and A-F.

11-2. Using the XLAT instruction, develop a table lookup procedure that converts AL from the ASCII-coded characters 30H-39H and 41H-46H into the numbers 0-9 and letters A-F in AL. Note that this table contains many unused memory locations that can be bypassed using the DUP directive to store duplicates.
11-3. Using the technique developed in Example 11-2, develop a lookup table procedure that converts AL from hexadecimal into two ASCII-coded digits stored in the table. For example, the first entry in the table, representing 00H in AL should contain a 3030H, the second entry representing a 01H should contain a 3031H, and so forth. Because this table is very large, only store entries for the values 00H through 0FH.

11-4. In Example 11-2, what is the purpose and operation of the MOV AX, CS:[SI] instruction?

11-5. Using the technique developed in Example 11-3, write a program that displays 8 different phases of your own choosing whenever any key is pressed on the keyboard. Note that a random number, which generates a number between 0 and 7 in CL, is obtained with the following procedure:

```
RAND PROC NEAR
INC CL
MOV AH, 6
MOV DL, -1
INT 21H
JZ RAND
AND CL, 7
RET
RAND ENDP
```

11-6. What is the purpose of the .IF AL < '0' || AL > '1' statement in Example 11-4?


11-8. Using the SCASB instruction write a procedure that skips over spaces found in the character string addresses by ES:DI. Note that this string is no more than 256 bytes in length.

11-9. Explain how Example 11-5 converts the encrypted uppercase letter B in AL into 43H (C).

11-10. What is the purpose of the REPNE prefix used with the SCASB instruction in Example 11-5?