

Code Converter

Experiment #7

By:

Pedro Cabrera

Submitted to:

Dr. V. Rajaravivarma

CET-2123C Fundamentals of Microprocessors

Department of Electronics Engineering Technology

Valencia College – West Campus

3/2/2024

INTRODUCTION

In this report, we explore the Binary-Coded Decimal (BCD) system, often overshadowed by more prevalent numerical systems in computing. Despite its lesser status, understanding BCD is crucial due to its application in legacy systems. Through an examination of conversion techniques such as the double dabble and isolation method, this report will elucidate the process of converting binary to BCD and vice versa. The aim is to demonstrate these methods' effectiveness and efficiency, particularly for students and practitioners dealing with legacy computing systems.

PARTS LIST:

- Windows based computer
- Visual Studio Code 2022

DISCUSSION:

The lab manual delves into numerical systems in computing, highlighting the Binary-Coded Decimal (BCD) system's importance in legacy systems. Through the lab exercises, it becomes apparent that while BCD is not as prevalent today, it plays a vital role in understanding older technologies. The manual guides us through the double dabble method for binary to BCD conversion, revealing the complexity of conversion processes due to the non-integer power relationship between binary and decimal systems. Additionally, it introduces the isolation method, a more efficient technique for BCD to binary conversion, illustrating its fewer required steps. The manual provides visual aids, such as tables and flowcharts, to clarify these methods and underscores the importance of not assuming the reader's familiarity with technical jargon or concepts.

```

main.asm
1  .MODEL FLAT
2  .DATA
3  BCD DW 6789H
4  .CODE
5  main PROC
6      ;----- CLEAR CONTENTS OF REGISTERS -----
7      xor eax,eax
8      xor ebx,ebx
9      xor ecx,ecx
10     xor edx,edx
11     ;----- BEGIN MAIN PROCEDURE HERE -----
12
13     mov dx,BCD
14     mov ch,4
15
16     X10: shl ax,1
17     mov bx,ax
18     mov cl,2
19     shl ax,cl
20     add ax,bx
21
22     mov cl,4
23     rol dx,cl
24     mov bx,dx
25     and bx,0Fh
26     add ax,bx
27
28     dec ch
29     jnz X10
30
31     ; At this point, AX contains the binary equivalent of 6789 BCD,
32     ; which is 1A85h (convert this to decimal to see the result)
33
34     ;----- END MAIN PROCEDURE HERE -----
35     ret
36 main ENDP
37 END
38

```

(Figure 1) Convert BDC to Binary.

```
EAX = 00001A85
EBX = 00000009
ECX = 00000004
EDX = 00006789
ESI = 00231005
EDI = 00231005
EIP = 00231042
ESP = 0033F7E0
EBP = 0033F7EC
EFL = 00000246

OV = 0 UP = 0 EI = 1
PL = 0 ZR = 1 AC = 0
PE = 1 CY = 0
```

(Figure 2)

This is a program for converting a Binary-Coded Decimal (BCD) number into its binary equivalent. It initializes the BCD value (6789) and clears the contents of the registers to avoid unexpected behavior. The main procedure uses a loop that shifts and adds the value to process each digit of the BCD. It performs bitwise operations to manipulate the bits accordingly, ensuring the conversion is done correctly. The result is stored in the AX register, which, at the end of the procedure, contains the binary equivalent of the given BCD number.

VALIDATION OF DATA:

Q1 *Can you create a program for binary to BCD using the double dabble method?*

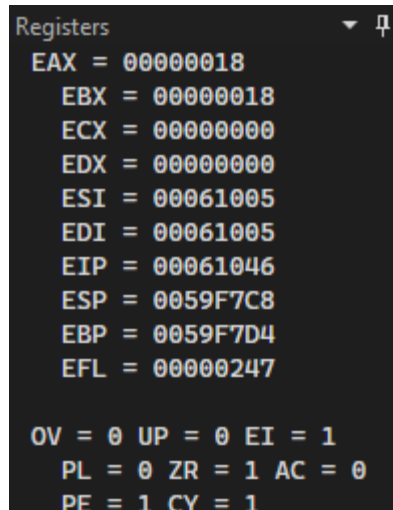
While attempting to create a program for converting binary to BCD using the double dabble method, I encountered several challenges that prevented me from completing a fully functional version. Issues arose with the register displays not showing the correct results, among other technical difficulties. Typically, the double dabble algorithm involves shifting bits left, checking each nibble for values greater than 4, and then adding 3 to those nibbles as necessary, all while iterating through each bit of the binary number. This method is efficient for converting binary numbers into their BCD equivalent without using division or multiplication, making it suitable for low-power or simple hardware implementations where processing resources are limited.

```

main.asm
1  .MODEL FLAT
2  .DATA
3  binaryValue DW 012h ; Example binary value to convert
4
5  .CODE
6  main PROC
7      ;----- CLEAR CONTENTS OF REGISTERS -----
8      xor eax,eax
9      xor ebx,ebx
10     xor ecx,ecx
11     xor edx,edx
12     ;----- BEGIN MAIN PROCEDURE HERE -----
13
14     ; Load the binary number into AX
15     mov ax, binaryValue
16
17     ; CX will count the shifts, 16 for a 16-bit register
18     mov cx, 16
19
20     ; Clear BX which will be used to accumulate the BCD result
21     xor bx, bx
22
23     DoubleDabble:
24         ; Shift left through BX to make room for the next bit
25         shl bx, 1
26         ; Shift the next bit out of AX into the carry flag
27         rcl ax, 1
28         ; Add the carry flag to BX
29         adc bx, 0
30
31         ; Adjust each nibble in BX if it is greater than 4
32         ; First nibble
33         cmp bl, 9
34         jbe NoAdjust1
35         add bl, 6
36
37     NoAdjust1:
38         ; Second nibble
39         cmp bh, 9
40         jbe NoAdjust2
41         add bh, 6
42
43     NoAdjust2:
44         dec cx
45         jnz DoubleDabble
46
47     ; Store the BCD result in AX
48     mov ax, bx
49
50
51     ret
52
53 main ENDP
54 END main

```

This was my latest attempt. This works; however, not all the time.



```
Registers
EAX = 00000018
EBX = 00000018
ECX = 00000000
EDX = 00000000
ESI = 00061005
EDI = 00061005
EIP = 00061046
ESP = 0059F7C8
EBP = 0059F7D4
EFL = 00000247

OV = 0 UP = 0 EI = 1
PL = 0 ZR = 1 AC = 0
PE = 1 CY = 1
```

Registers displaying 18 which is in fact 12h decimal, to BCD.

Q2 What is the reverse-double-dabble algorithm?

The reverse-double-dabble algorithm is essentially the inverse process of the double-dabble algorithm. While the double dabble is used to convert binary to BCD (Binary-Coded Decimal), the reverse-double-dabble algorithm would logically be used to convert BCD to binary. This would involve a series of operations that reverse the steps taken in the double-dabble method, adjusting and shifting BCD-encoded digits back into a binary number format. The specific details of the algorithm would depend on the implementation, focusing on efficiently reversing the BCD encoding process.

CONCLUSION:

In conclusion, the laboratory exercise on the conversion between binary and BCD representations utilizing the double dabble method has been instrumental in enhancing both theoretical comprehension and practical skill sets. The exploration of both the double dabble and reverse-double-dabble techniques is crucial for understanding the interactions between different number systems. However, in real-world applications, my preference leans towards converting BCD to decimal and subsequently from decimal to binary. This preference stems from the direct correlation between decimal numbers and their BCD counterparts, where each decimal digit is directly represented in binary. This method is more intuitive and generally involves simpler arithmetic operations, making it a preferable choice for practical conversions.