Packed Decimal Arithmetic, DAA, DAS

Packed decimal arithmetic is a way of representing and performing arithmetic on decimal numbers using a binary format.

Packed decimal numbers store two decimal digits per byte, with each digit represented by 4 bits.

This makes packed decimal arithmetic a more efficient way to store and manipulate decimal numbers than traditional binary arithmetic, which stores each decimal digit as a separate byte.

Packed decimal arithmetic is often used in financial and business applications, where it is important to be able to perform accurate calculations on large numbers.

It is also used in some scientific and engineering applications, where it is necessary to perform calculations on numbers with a high degree of precision.

There are two main advantages to using packed decimal arithmetic:

Efficiency: Packed decimal numbers require less storage space than traditional binary numbers. This is because packed decimal numbers store two decimal digits per byte, while traditional binary numbers store each decimal digit as a separate byte.

Accuracy: Packed decimal arithmetic can be used to perform calculations with a high degree of accuracy. This is because packed decimal numbers store two decimal digits per byte, which allows for more precise calculations than traditional binary arithmetic.

However, there is one main disadvantage to using packed decimal arithmetic:

Performance: Packed decimal arithmetic can be slower than traditional binary arithmetic. This is because packed decimal arithmetic requires additional instructions to convert packed decimal numbers to and from binary numbers, which is necessary for performing arithmetic operations. Overall, packed decimal arithmetic is a powerful tool for performing arithmetic on decimal numbers. It is especially useful for financial and business applications, where it is important to be able to perform accurate calculations on large numbers.

Here are some examples of packed decimal numbers:

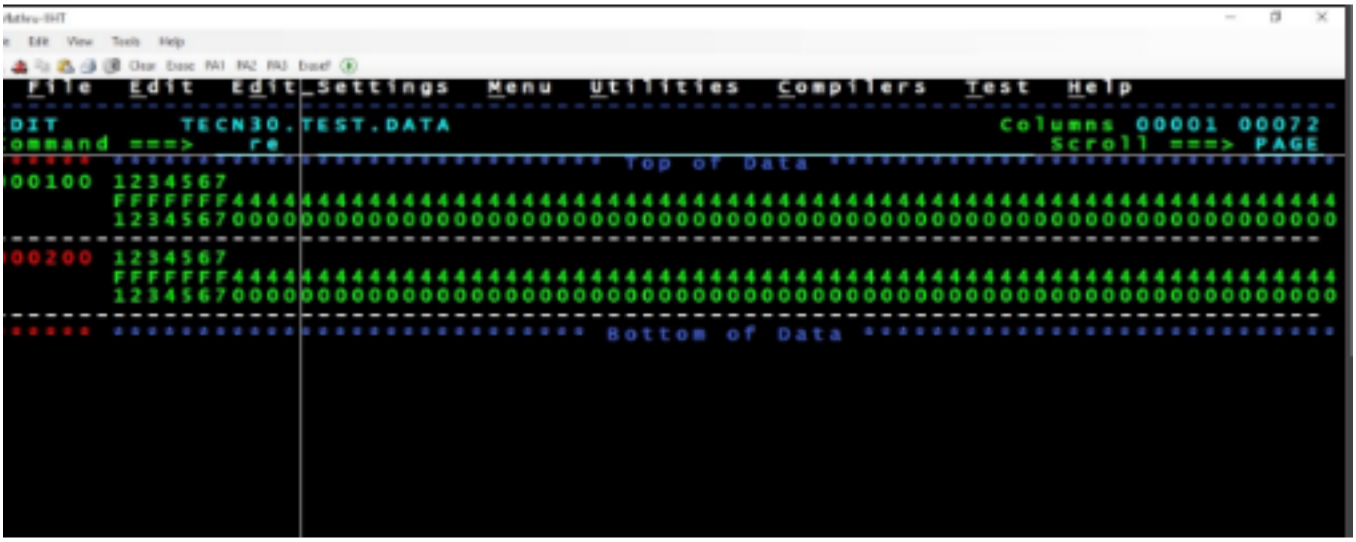
```
bcd1 QWORD 2345673928737285h    ; 2,345,673,928,737,285 decimal

bcd2 DWORD 12345678h             ; 12,345,678 decimal

bcd3 DWORD 08723654h             ; 8,723,654 decimal

bcd4 WORD 9345h                 ; 9,345 decimal

bcd5 WORD 0237h                 ; 237 decimal

bcd6 BYTE 34h                   ; 34 decimal
```

The following two instructions are used to adjust the result of an addition or subtraction operation on packed decimals:

- **DAA (decimal adjust after addition)**
- **DAS (decimal adjust after subtraction)**

These instructions are necessary because packed decimal numbers store two decimal digits per byte. After an addition or subtraction operation, it is possible that the result will be a three-digit number. The DAA and DAS instructions adjust the result to ensure that it is a valid two-digit packed decimal number.

Packed decimal arithmetic can be used to perform all of the basic arithmetic operations, including addition, subtraction, multiplication, and division. However, there are no specific instructions for multiplication and division of packed decimal numbers. This means that packed decimal numbers must be unpacked before these operations can be performed, and then repacked after the operations are complete.



Despite this disadvantage, packed decimal arithmetic is a powerful tool for performing arithmetic on decimal numbers. It is especially useful for financial and business applications, where it is important to be able to perform accurate calculations on large numbers.

==

DAA (decimal adjust after addition)

==

The DAA (decimal adjust after addition) instruction is used to convert the binary sum produced by the ADD or ADC instruction in the AL register to packed decimal format.

This is necessary because packed decimal numbers store two decimal digits per byte. After an addition operation, it is possible that the result will be a three-digit number.

The DAA instruction adjusts the result to ensure that it is a valid two-digit packed decimal number.

Here is an example of how to use the DAA instruction:

```
mov al, 35h
add al, 48h ; AL = 7Dh
daa ; AL = 83h (adjusted result)
```

In this example, the ADD instruction adds the packed decimal numbers

35 and 48. The result is 7Dh, which is a three-digit binary number.

The DAA instruction adjusts the result to 83h, which is the packed decimal representation of the sum of 35 and 48.

The DAA instruction can be used to perform packed decimal addition on any number of digits.

However, it is important to note that the sum variable must contain space for one more digit than the operands. This is because the DAA instruction can generate a carry digit.

The following program adds two 16-bit packed decimal integers and stores the sum in a packed doubleword:

```
; Packed Decimal Example
(AddPacked.asm)
; Demonstrate packed decimal addition.
INCLUDE Irvine32.inc
.data
packed_1 WORD 4536h
packed_2 WORD 7207h
sum DWORD ?
.code
main PROC
    ; Initialize sum and index.
    mov sum, 0
    xor esi, esi

    ; Add low bytes and handle carry.
    add al, BYTE PTR packed_1[esi]
    daa
    mov BYTE PTR sum[esi], al

    ; Add high bytes and include carry.
    inc esi
    add al, BYTE PTR packed_1[esi]
    adc al, BYTE PTR packed_2[esi]
    daa
    mov BYTE PTR sum[esi], al

    ; Add final carry, if any.
    inc esi
    adc al, 0
    mov BYTE PTR sum[esi], al

    ; Display the sum in hexadecimal.
    mov eax, sum
    call WriteHex
```

```
    call Crlf
    exit
main ENDP
END main
```

This program uses a loop to add the two packed decimal integers one digit at a time. The DAA instruction is used to adjust the result of each addition operation.

The sum variable is a packed doubleword, which is large enough to store the sum of two 16-bit packed decimal integers.

The DAA instruction is a powerful tool for performing packed decimal arithmetic. It is easy to use and can be used to implement a variety of arithmetic algorithms.

===

DAS (decimal adjust after subtraction)

===

The DAS (decimal adjust after subtraction) instruction is used to convert the binary result of a SUB or SBB instruction in the AL register to packed decimal format.

This is necessary because packed decimal numbers store two decimal digits per byte.

After a subtraction operation, it is possible that the result will be a negative three-digit number.

The DAS instruction adjusts the result to ensure that it is a valid two-digit packed decimal number.

Here is an example of how to use the DAS instruction:

```
mov bl, 48h
mov al, 85h
sub al, bl ; AL = 3Dh
das ; AL = 37h (adjusted result)
```

In this example, the SUB instruction subtracts the packed decimal numbers 85 and 48. The result is 3Dh, which is a negative three-digit binary number.

The DAS instruction adjusts the result to 37h, which is the packed decimal representation of the difference of 85 and 48.

The DAS instruction can be used to perform packed decimal subtraction on any number of digits.

However, it is important to note that the result variable must contain space for one more digit than the operands.

This is because the DAS instruction can generate a borrow digit.

Here is a pseudocode implementation of the DAS instruction:

```
DAS(AL):
  if AL < 10:
    return AL
  else:
    AL -= 10
    AH += 1
    if AH >= 10:
      AH -= 10
      CF = 1
    else:
      CF = 0
    return AL
```

Explanation:

The DAS instruction begins by checking if the value in the AL register (the low decimal digit) is less than 10. If it is, it means there's no need for adjustment, and it returns AL as it is.

If AL is greater than or equal to 10, it means there's a carry or overflow in the low digit. To correct this:

Subtract 10 from AL, effectively "borrowing" from the low digit.
Increment AH (the high digit) to account for the borrow from AL.

Check if AH itself requires adjustment. If AH is now greater than or equal to 10, it means there's a carry in the high digit as well.

If AH needs adjustment, subtract 10 from AH to bring it within the valid range.

Finally, set the Carry Flag (CF) to 1 to indicate that there was a carry or borrow operation.

If AH does not require adjustment, set CF to 0 to indicate that no carry occurred.

In summary, the DAS instruction ensures that after a subtraction operation, the AL and AH registers contain valid packed decimal digits, taking into account any borrows or carries to maintain the integrity of the packed decimal representation.

It is a crucial instruction in packed decimal arithmetic, commonly used in financial and decimal data processing applications.