68000 Binary Coded Decimal (BCD) Arithmetic

- Binary Coded Decimal (BCD) is a way to store decimal numbers in binary. This number representation uses 4 bits to store each digit from 0 to 9. For example:

$$1999_{10} = 0001\ 1001\ 1001\ 1001 \text{ in BCD}$$

- BCD wastes storage space since 4 bits are used to store 10 combinations rather than the maximum possible 16.
- BCD is often used in business applications and calculators.
- The 68000 instruction set includes three instructions that offer some support for BCD arithmetic:
  - ABCD Add BCD with extend
  - SBCD Subtract BCD with extend
  - NBCD Negate BCD
- BCD instructions use and affect the X-bit because they are intended to be used in chained calculations where arithmetic is done on strings of BCD digits.
  - For addition: the X-bit records the carry
  - For subtraction: the X-bit records the borrow

ABCD

Add Decimal with Extend
(M68000 Family)

ABCD

Operation: Source10 + Destination10 + X → Destination

Assembler Syntax: ABCD Dy,Dx
ABCD – (Ay), – (Ax)

Attributes: Size = (Byte)

Description: Adds the source operand to the destination operand along with the extend bit, and stores the result in the destination location. The addition is performed using binary-coded decimal arithmetic. The operands, which are packed binary-coded decimal numbers, can be addressed in two different ways:

1. Data Register to Data Register: The operands are contained in the data registers specified in the instruction.
2. Memory to Memory: The operands are addressed with the predecrement addressing mode using the address registers specified in the instruction.

This operation is a byte operation only.

ABCD

Add Decimal with Extend
(M68000 Family)

ABCD

Condition Codes:

X	N	Z	V	C
*	U	*	U	*

X — Set the same as the carry bit.
N — Undefined.
Z — Cleared if the result is nonzero; unchanged otherwise.
V — Undefined.
C — Set if a decimal carry was generated; cleared otherwise.

NOTE

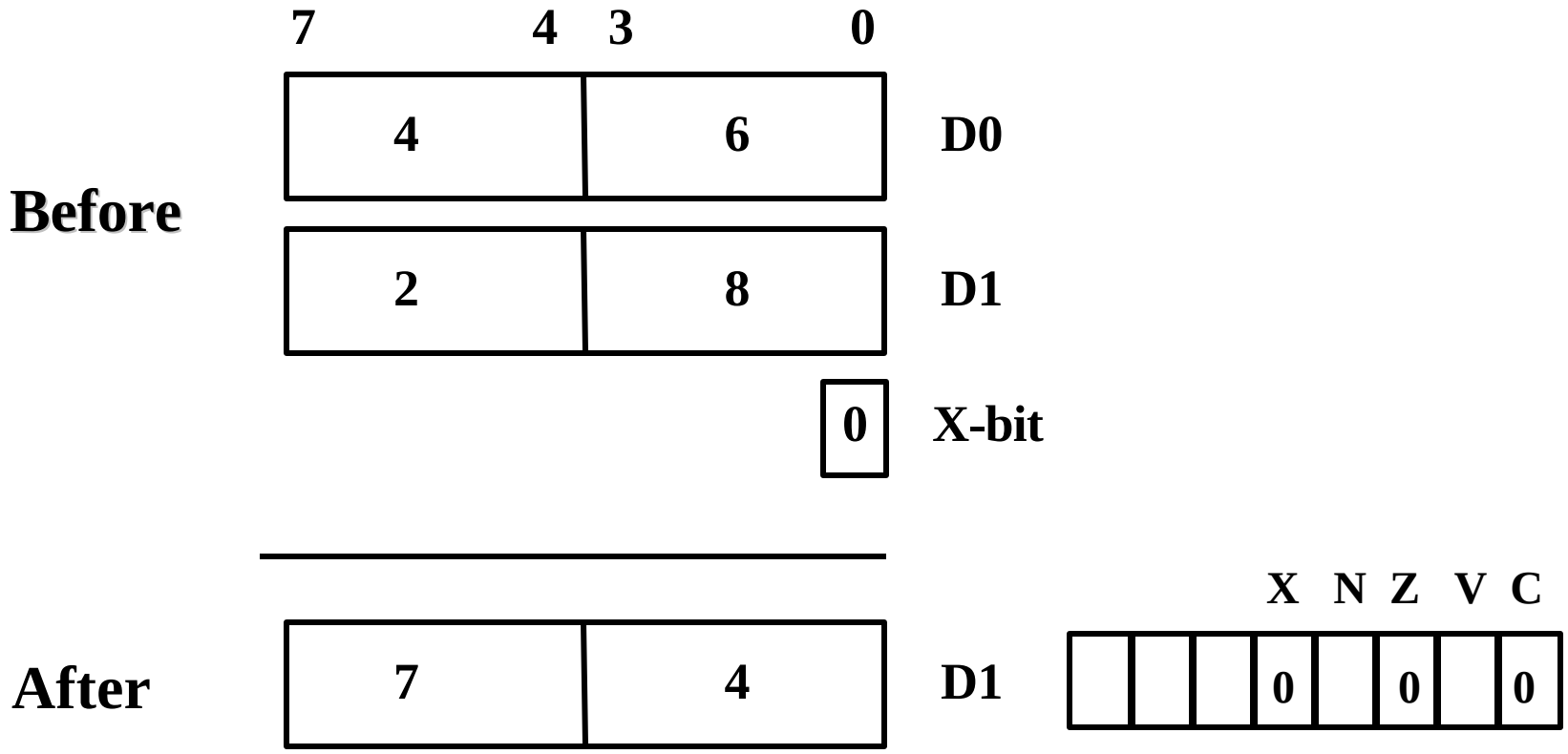
Normally, the Z condition code bit is set via programming before the start of an operation. This allows successful tests for zero results upon completion of multiple-precision operations.

Effect of ABCD

When X = 0 initially

ABCD D0,D1

Add D0 to D1 with the X-bit

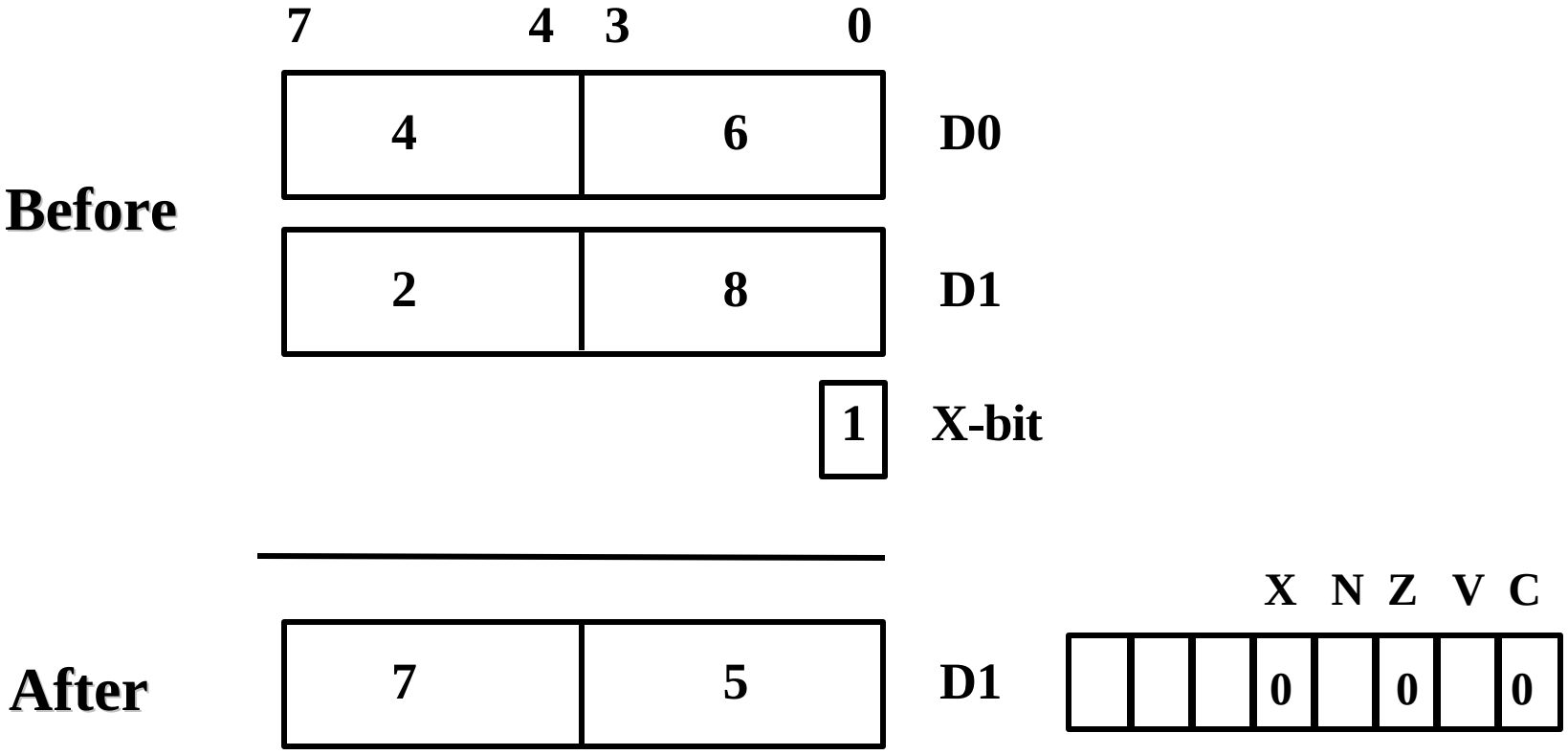


Effect of ABCD

When X = 1 initially

ABCD D0,D1

Add D0 to D1 with the X-bit



SBCD

Subtract Decimal with Extend
(M68000 Family)

SBCD

Operation: Destination10 – Source10 – X → Destination

Assembler Syntax: SBCD Dx,Dy
SBCD – (Ax), – (Ay)

Attributes: Size = (Byte)

Description: Subtracts the source operand and the extend bit from the destination operand and stores the result in the destination location. The subtraction is performed using binary-coded decimal arithmetic; the operands are packed binary-coded decimal numbers. The instruction has two modes:

1. Data register to data register—the data registers specified in the instruction contain the operands.
2. Memory to memory—the address registers specified in the instruction access the operands from memory using the predecrement addressing mode.

This operation is a byte operation only.

SBCD

Subtract Decimal with Extend
(M68000 Family)

SBCD

Condition Codes:

X	N	Z	V	C
*	U	*	U	*

X — Set the same as the carry bit.
N — Undefined.
Z — Cleared if the result is nonzero; unchanged otherwise.
V — Undefined.
C — Set if a borrow (decimal) is generated; cleared otherwise.

NOTE

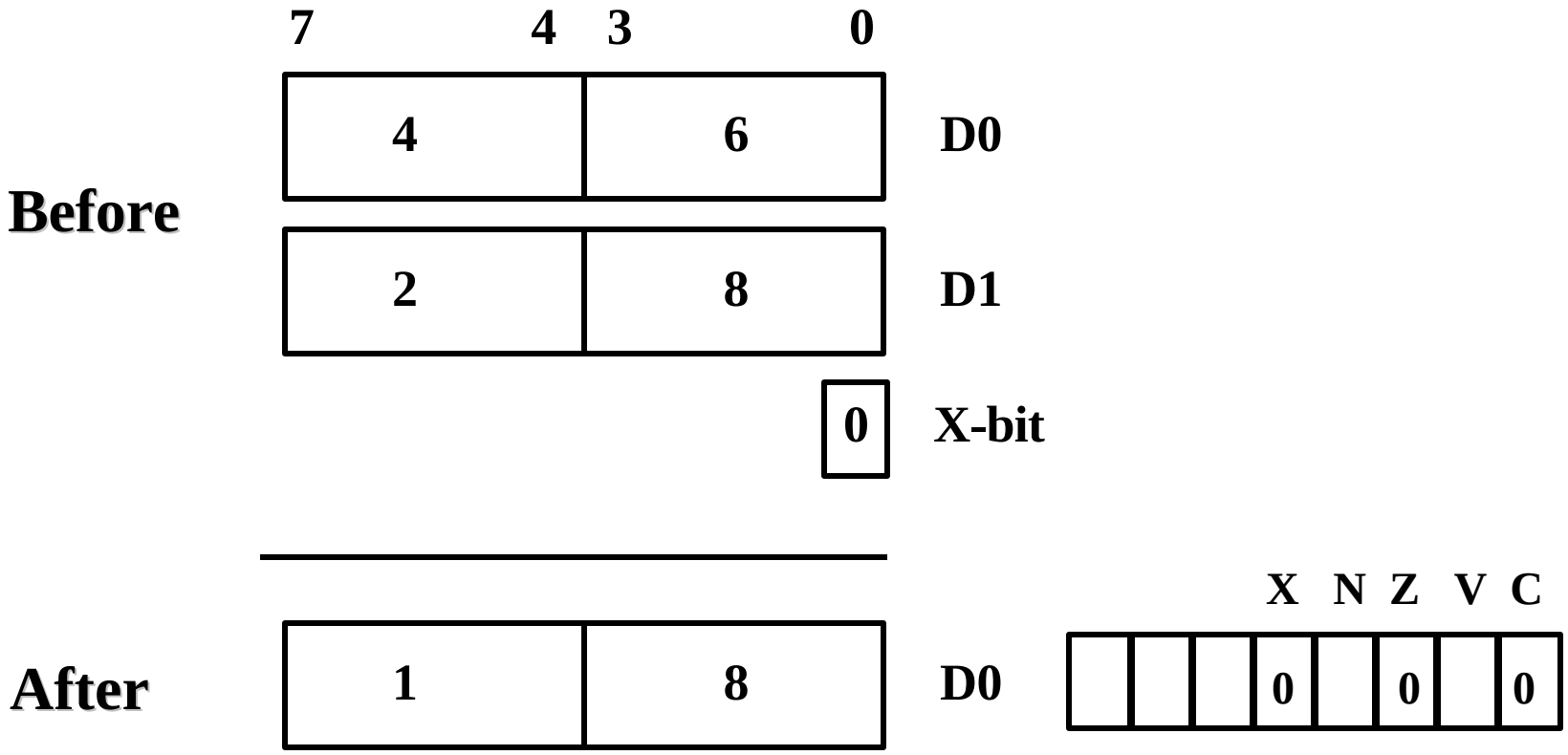
Normally the Z condition code bit is set via programming before the start of an operation. This allows successful tests for zero results upon completion of multiple-precision operations.

Effect of SBCD

When X = 0 initially

SBCD D1,D0

Subtract D1 from D0 with the X-bit

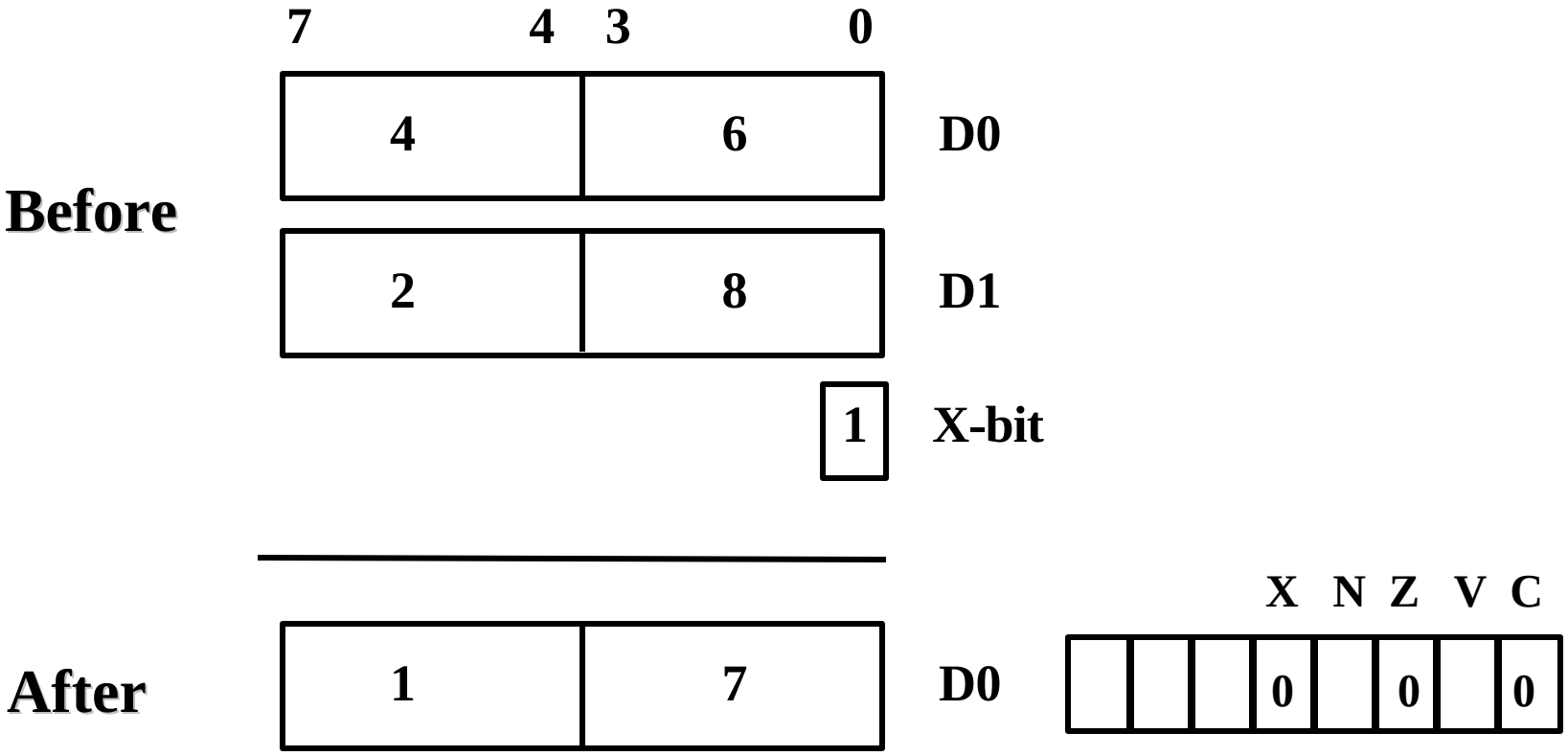


Effect of SBCD

When X = 1 initially

SBCD D1,D0

Subtract D1 from D0 with the X-bit



NBCD

Negate Decimal with Extend
(M68000 Family)

NBCD

Operation: $0 - \text{Destination}_{10} - X \rightarrow \text{Destination}$

Assembler Syntax: NBCD < ea >

Attributes: Size = (Byte)

Description: Subtracts the destination operand and the extend bit from zero. The operation is performed using binary-coded decimal arithmetic. The packed binary-coded decimal result is saved in the destination location. This instruction produces the tens complement of the destination if the extend bit is zero or the nines complement if the extend bit is one. This is a byte operation only.

Condition Codes:

X	N	Z	V	C
*	U	*	U	*

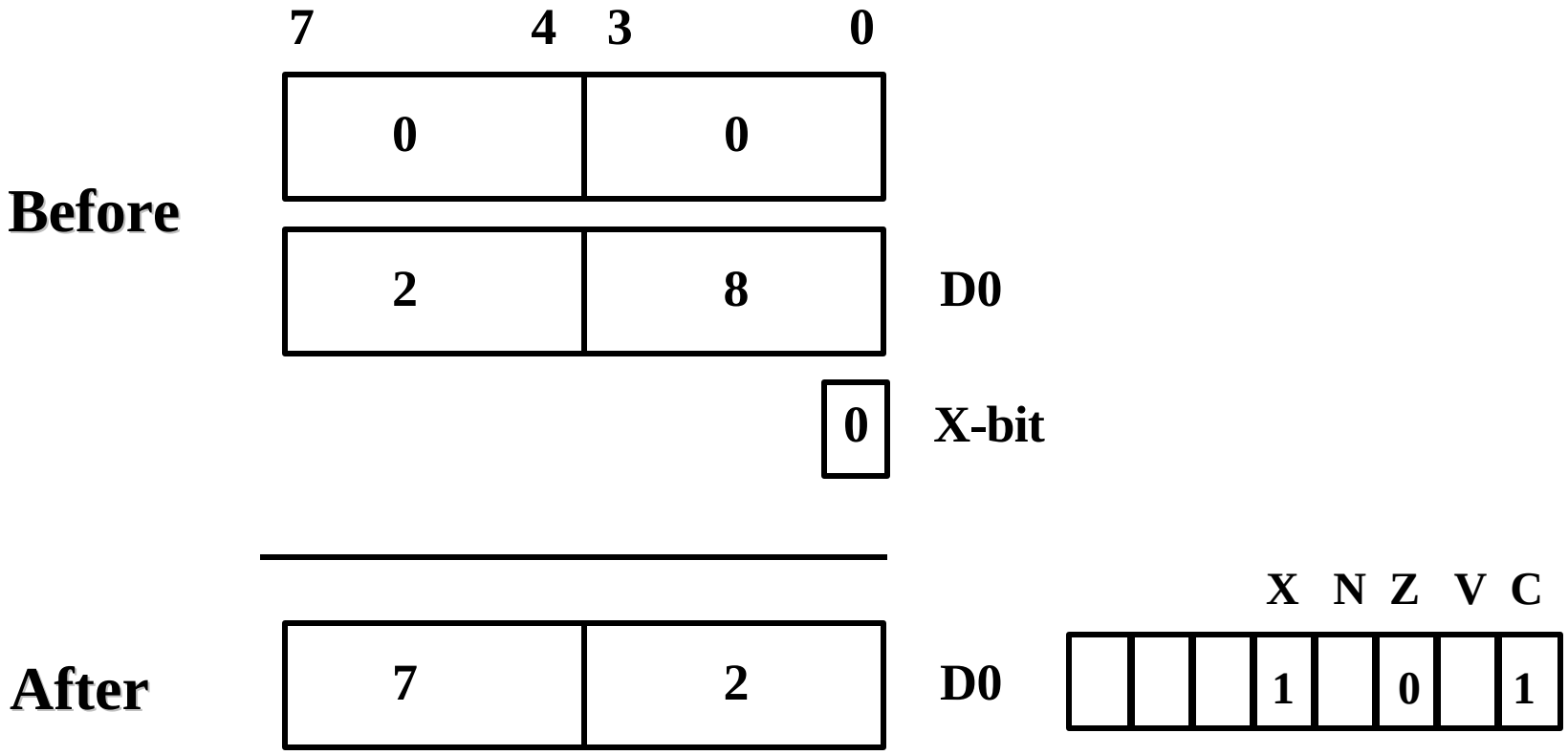
X — Set the same as the carry bit.
N — Undefined.
Z — Cleared if the result is nonzero; unchanged otherwise.
V — Undefined.
C — Set if a decimal borrow occurs; cleared otherwise.

Effect of NBCD

When X = 0 initially

NBCD D0

Subtract D0 from 0 with the X-bit

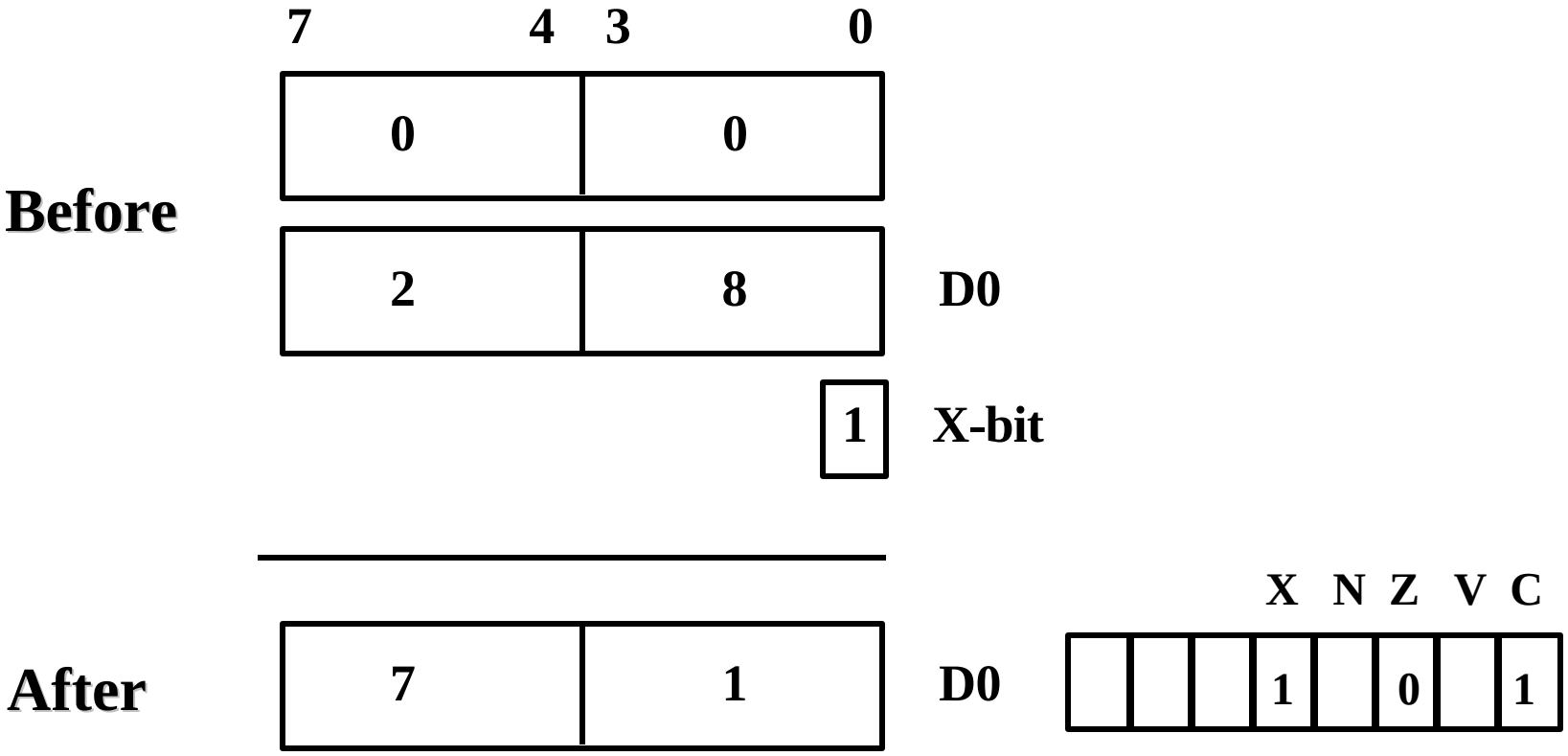


Effect of NBCD

When X = 1 initially

NBCD D0

Subtract D0 from 0 with the X-bit



BCD Addition Example

- **Two BCD strings each with 12 BCD digits (six bytes) and stored in memory starting at locations: String1, String2, are to be added together with the result to be stored in memory starting at String2**

```
        ORG     $1000
ADDBCD  MOVE.W  #5,D0       Loop counter, six bytes to be added
        ANDI    #$EF,CCR    Clear X-bit  in CCR
        LEA     String1+6,A0  A0 points at end of source string +1
        LEA     String2+6,A1  A0 points at end of destination string +1
LOOP    ABCD    -(A0),-(A1)   Add pair of digits with carry-in
        DBRA    D0,LOOP     Repeat until 12 digits are added
        RTS
          .
          .
String1 DS.B    6
String2 DS.B    6
```

DBRA used here because it does not affect the X-bit needed in BCD arithmetic

BCD Subtraction Example

- **Two BCD strings with 12 BCD digits (six bytes) each are stored in memory starting at locations String1, String2.**
- **String1 is to subtracted from String 2 with the result to be stored in memory starting at String2**

```
          ORG       $1000
SUBBCD    MOVE.W    #5,D0          Loop counter, six bytes to be added
          ANDI      #$EF,CCR       Clear X-bit in CCR
          LEA       String1+6,A0   A0 points at end of source string +1
          LEA       String2+6,A1   A0 points at end of destination string +1
LOOP      SBCD      -(A0),-(A1)    Subtract pair of digits with borrow
          DBRA      D0,LOOP        Repeat until 12 digits are added
          RTS
            .
            .
String1   DS.B      6
String2   DS.B      6
```

68000 Multiple-Precision Arithmetic

- **For numerical values, *precision* refers to the number of significant digits in the numerical value.**

  →If more precision is needed in a numerical value, more significant digits must be used to yield a more precise result.

- **The maximum single-precision operand length supported by the 68000 is 32 bits. Thus, values with greater length cannot be handled as a single arithmetic operand by the CPU.**
- **To extend the precision, several 32-bit operands can be used and considered mathematically as a single value.**
- **The 68000 offers three special instructions to facilitate addition, subtraction, and negation of multiple-precision integers:**

  - **ADDX ADD with eXtend**
  - **SUBX SUBtract with eXtend**
  - **NEGX NEGate with eXtend**

ADDX

Add Extended
(M68000 Family)

ADDX

Operation: Source + Destination + X → Destination

Assembler Syntax: ADDX Dy,Dx
ADDX – (Ay), – (Ax)

Attributes: Size = (Byte, Word, Long)

Description: Adds the source operand and the extend bit to the destination operand and stores the result in the destination location.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

X — Set the same as the carry bit.
N — Set if the result is negative; cleared otherwise.
Z — Cleared if the result is nonzero; unchanged otherwise.
V — Set if an overflow occurs; cleared otherwise.
C — Set if a carry is generated; cleared otherwise.

SUBX

Subtract with Extend
(M68000 Family)

SUBX

Operation: Destination – Source – X → Destination

Assembler Syntax: SUBX Dx,Dy
SUBX – (Ax), – (Ay)

Attributes: Size = (Byte, Word, Long)

Description: Subtracts the source operand and the extend bit from the destination operand and stores the result in the destination

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

X — Set to the value of the carry bit.
N — Set if the result is negative; cleared otherwise.
Z — Cleared if the result is nonzero; unchanged otherwise.
V — Set if an overflow occurs; cleared otherwise.
C — Set if a borrow occurs; cleared otherwise.

NEGX

Negate with Extend
(M68000 Family)

NEGX

Operation: 0 – Destination – X → Destination

Assembler Syntax: NEGX < ea >

Attributes: Size = (Byte, Word, Long)

Description: Subtracts the destination operand and the extend bit from zero. Stores the result in the destination location. The size of the operation is specified as byte, word, or long.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

X — Set the same as the carry bit.
N — Set if the result is negative; cleared otherwise.
Z — Cleared if the result is nonzero; unchanged otherwise.
V — Set if an overflow occurs; cleared otherwise.
C — Set if a borrow occurs; cleared otherwise.

Multiple-Precision Addition Example

- Two unsigned binary numbers each with 128 bits (16 bytes) and stored in memory starting at locations Num1, Num2 are to be added together with the result to be stored in memory starting at Num2

```
        ORG      $1000
MPADD   MOVE.W   #3,D0        Four long words to be added
        ANDI     #$EF,CCR      Clear X-bit  in CCR
        LEA      Num1,A0       A0 points at start of source
        ADDA     #16,A0        A0 points to end of source  + 1
        LEA      Num2,A1       A1 points at start of destination
        ADDA     #16,A1        A1 points to end of destination  + 1
LOOP    ADDX.L   -(A0),-(A1)   Add pair of long words with carry-in
        DBRA     D0,LOOP       Repeat until 4 long words are added
        RTS
          .
          .
Num1    DS.L     4
Num2    DS.L     4
```

DBRA is used here because it does not affect the X-bit needed in multiple-precision arithmetic

Multiple-Precision Subtraction Example

- Two unsigned binary numbers each with 128 bits (16 bytes) and stored in memory starting at locations Num1, Num2
- Num1 to be is to subtracted from Num2 with the result to be stored in memory starting at Num2

```
        ORG      $1000
MPADD   MOVE.W   #3,D0        Four long words to be added
        ANDI     #$EF,CCR      Clear X-bit  in CCR
        LEA      Num1,A0       A0 points at start of source
        ADDA     #16,A0        A0 points to end of source + 1
        LEA      Num2,A1       A1 points at start of destination
        ADDA     #16,A1        A1 points to end of destination + 1
LOOP    SUBX.L   -(A0),-(A1)   Subtract pair of long words with borrow
        DBRA     D0,LOOP       Repeat until 4 long words are subtracted
        RTS
          .
          .
Num1    DS.L     4
Num2    DS.L     4
```

DBRA is used here because it does not affect the X-bit needed in multiple-precision arithmetic