Representation of numbers in binary:

(Ford & Topp call this the expanded form representation of a number)

$10_2 = 1\times2^1 + 0\times2^0 = 2_{10}$

$11_2 = 1\times2^1 + 1\times2^0 = 2 + 1 = 3_{10}$

How can you convert numbers from decimal to binary?

Subtraction of powers method:

Example: Convert $N = 217_{10}$ to $(D_7D_6D_5D_4D_3D_2D_1D_0)_2$

You can represent up to 256 using eight bits.

Want $N = 217_{10} = D_7\times2^7 + D_6\times2^6 + D_5\times2^5 + D_4\times2^4 + D_3\times2^3 + D_2\times2^2 + D_1\times2^1 + D_0\times2^0 = (D_7D_6D_5D_4D_3D_2D_1D_0)_2$

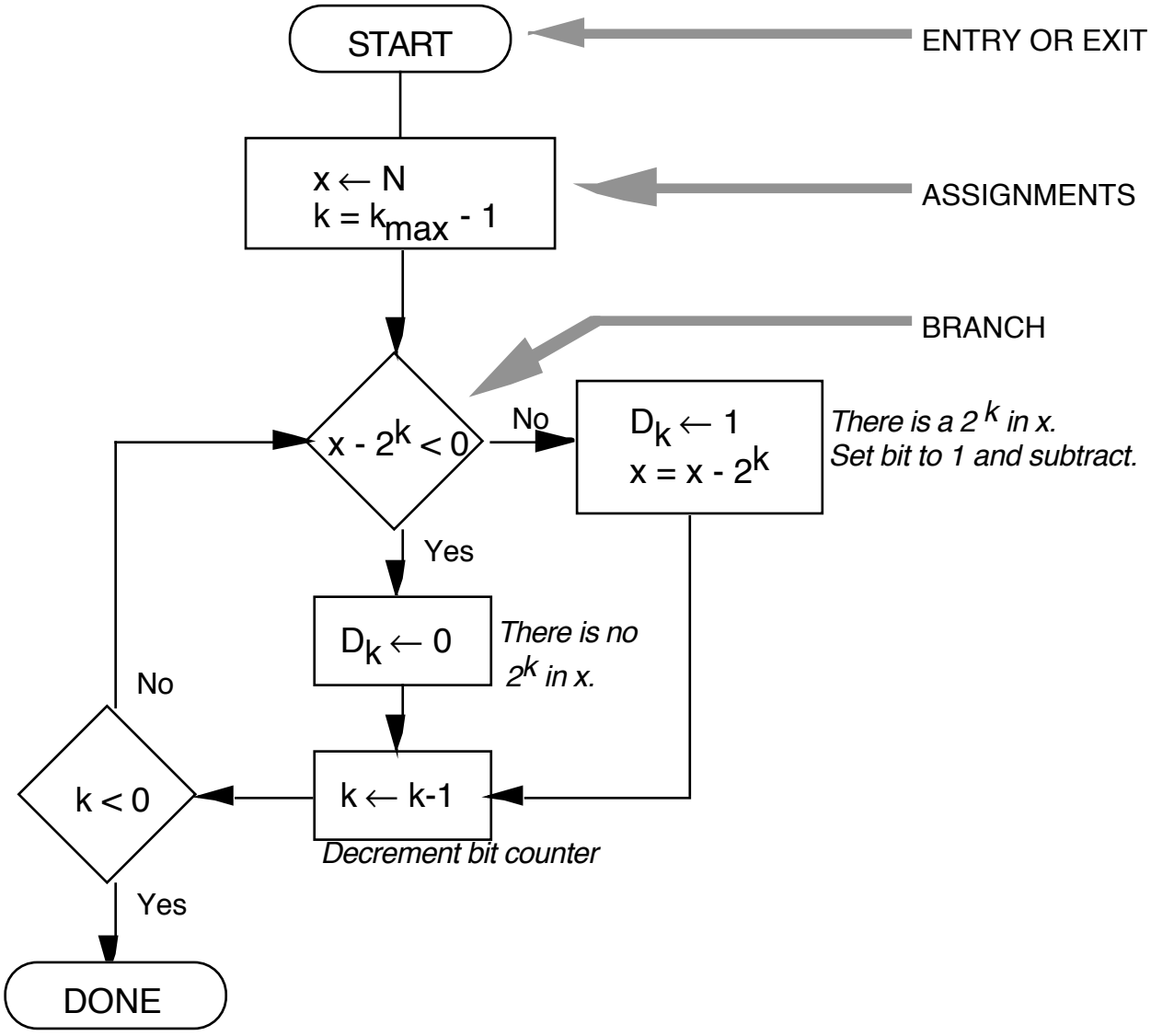
Test each bit (starting from the most significant)

$217_{10} - 2^7$	$= 217_{10} - 128_{10} = 89_{10} > 0$	$\rightarrow D_7=1$
$89_{10} - 2^6$	$= 89_{10} - 64_{10} = 25_{10} > 0$	$\rightarrow D_6=1$
$25_{10} - 2^5$	$= 25_{10} - 32_{10} < 0$	$\rightarrow D_5=0$
$25_{10} - 2^4$	$= 25_{10} - 16_{10} = 9_{10} > 0$	$\rightarrow D_4=1$
$9_{10} - 2^3$	$= 9_{10} - 8_{10} = 1_{10} > 0$	$\rightarrow D_3=1$
$1_{10} - 2^2$	$= 1_{10} - 4_{10} < 0$	$\rightarrow D_2=0$
$1_{10} - 2^1$	$= 1_{10} - 2_{10} = < 0$	$\rightarrow D_1=0$
$1_{10} - 2^0$	$= 1_{10} - 1_{10} = 0$	$\rightarrow D_0=1$

Therefore, $217_{10} = 11011001_2$

FLOWCHART (representing the subtraction of powers method)

N = STARTING NUMBER (the number to convert)
k_{max} = # of binary digits (8)



In pseudo code the same algorithm can be documented as:

```
x = number_tobe_conv
k = number_digits - 1
(* number_digits = 8 do loop starts at 7 *)

WHILE k≥0 do
      BEGIN
      if (x-2^k)<0 THEN d(k)=0
          ELSE
          BEGIN
          d(k) = 1
          x = x-2^k
          END
      k=k-1
      END
```

trace loop:
setup:
x=217
k=7
looping:
k=7
    $217-2^7=89$
    d(7)=1
    x=89
k=6
    $89-2^6=25$
    d(6)=1
    x=25
k=5
    $25-2^5=-7$
    d(5)=0
k=4
    $25-2^4=9$
    d(4)=1
    x=9
k=3
    $9-2^3=1$
    d(3)=1
    x=1
k=2
    $1-2^2=-3$
    d(2)=0
k=1
    $1-2^1=-2$
    d(1)=0
k=0
    $1-2^0=0$
    d(0)=1
    x=0

Even Odd decimal-to-binary conversion method:

Example:
Convert N = 217_{10} to $(D_7D_6D_5D_4D_3D_2D_1D_0)_2$

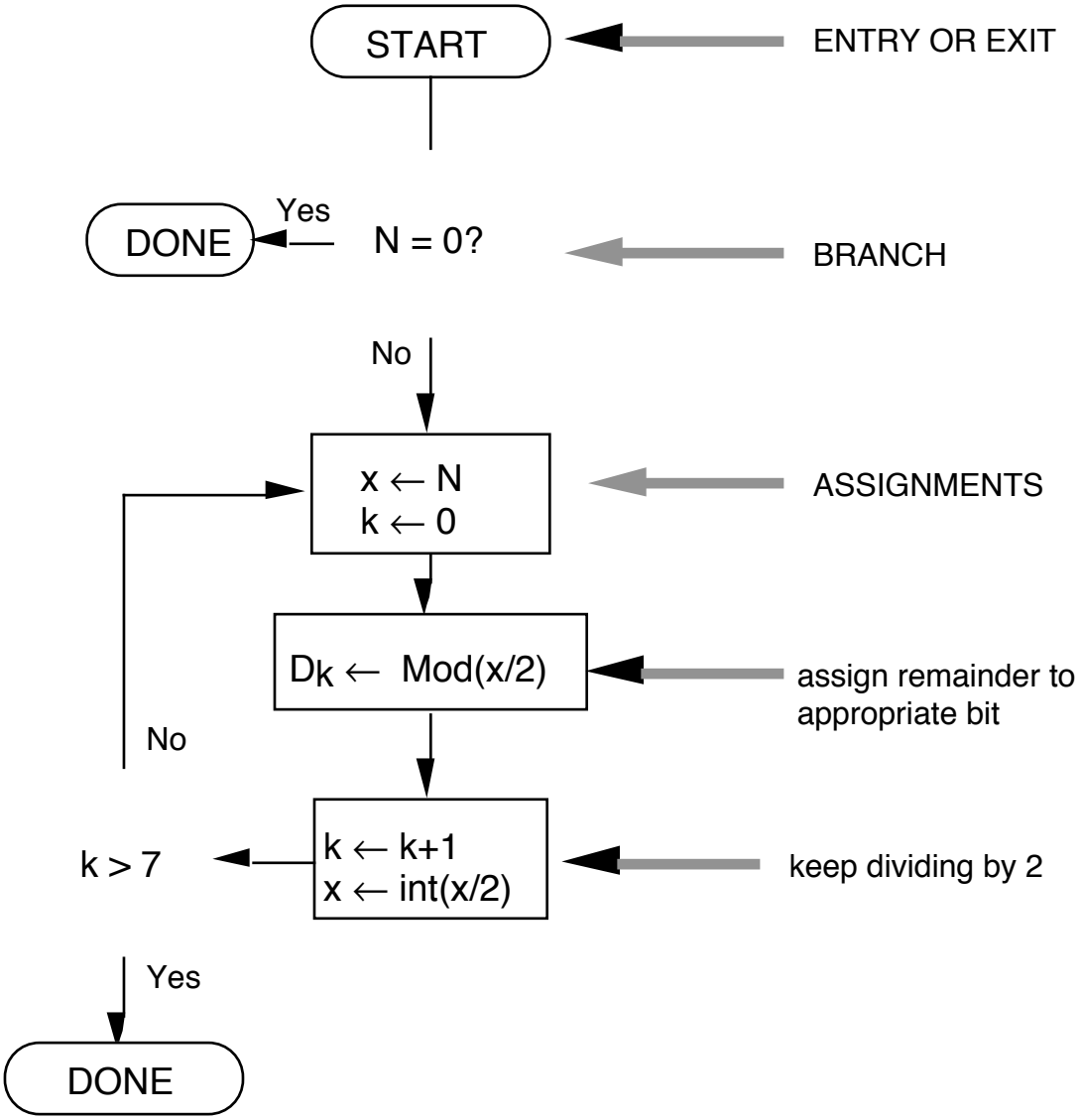
Test each bit (starting from the least significant)

$217_{10}/2$	= 108 with remainder=1	$\rightarrow D_0=1$
$108_{10}/2$	= 54 with remainder=0	$\rightarrow D_1=0$
$54_{10}/2$	= 27 with remainder=0	$\rightarrow D_2=0$
$27_{10}/2$	= 13 with remainder=1	$\rightarrow D_3=1$
$13_{10}/2$	= 6 with remainder=1	$\rightarrow D_4=1$
$6_{10}/2$	= 3 with remainder=0	$\rightarrow D_5=0$
$3_{10}/2$	= 1 with remainder=1	$\rightarrow D_6=1$
$1_{10}/2$	= 0 with remainder=1	$\rightarrow D_7=1$

Therefore, $217_{10} = 11011001_2$

FLOWCHART (representing the even-odd method)

N = STARTING NUMBER
kmax = # of binary digits (8)



In pseudo code the same algorithm can be documented as:

```
N = number_tobe_conv
x = quotient

IF N≥0 THEN
      BEGIN
      x=N
      k=0
      WHILE k<8 DO
          BEGIN
          d(k) = mod(x,2)
          x = int(x/2)
          k=k+1
          END
      END
```

trace loop:	setup:
	x=217
	k=0
looping:	
k=1	
d(0) = mod(217,2) = 1	
x=int(217,2) = 108	odd
k=2	
d(1) = mod(108,2) = 0	
x=int(108,2) = 54	even
k=3	
d(1) = mod(54,2) = 0	
x=int(54,2) = 27	even
k=4	
d(1) = mod(27,2) = 1	
x=int(27,2) = 13	odd
k=5	
d(1) = mod(13,2) = 1	
x=int(13,2) = 6	odd
k=6	
d(1) = mod(6,2) = 0	

```
    x=int(6,2) = 3                  even
k=7
    d(1) = mod(3,2) = 1
    x=int(3,2) = 1                  odd
k=8
    d(1) = mod(1,2) = 1
    x=int(1,2) = 0                  odd
```

How about fractions?

No one said we couldn't have negative powers of two!

$$\begin{aligned} 0.1101_2 &= 0\times2^0 + 1\times2^{-1} + 1\times2^{-2} + 0\times2^{-3} + 1\times2^{-4} \\ &= 1\times0.5 + 1\times.25 + 0\times.125 + 1\times.0625 \\ &= .5 + .25 + 0 + .0625 \\ &= 0.8125_{10} \end{aligned}$$

Binary $\rightarrow$ decimal conversion is always exact

Decimal $\rightarrow$ binary conversions are NOT always exact

Example:

0.42357_{10} @ $0.011011000..._2$ (binary numbers typically do not terminate)

Example: (using subtraction of powers conversion):

$0.42357_{10} - 2^{-1} = .42357-.5 < 0 \Rightarrow d_{-1}=0$

$0.42357_{10} - 2^{-2} = .42357-.25=.17357 \Rightarrow d_{-2}=1$

$0.17357_{10} - 2^{-3} = .17357-.125=.04857 \Rightarrow d_{-3}=1$

$0.04857_{10} - 2^{-4} = .04857-.0625 < 0 \Rightarrow d_{-4}=0$

$0.04857_{10} - 2^{-5} = .04857-.03125 = 0.01732 \Rightarrow d_{-5}=1$

$0.01732_{10} - 2^{-6} = .01732-.015625 = 0.001695 \Rightarrow d_{-6}=1$

$0.001695_{10} - 2^{-7} < 0 \Rightarrow d_{-7}=0$

$0.001695_{10} - 2^{-8} < 0 \Rightarrow d_{-8}=0$

$0.001695_{10} - 2^{-9} < 0 \quad \Rightarrow \quad d_{-9}=0$