Odometer in a car

1 3 9

Dials tick as miles are driven
- Underlying quantity
- Readout of that quantity in turning dials
Odometer in a car

1 3 9

Dials tick as miles are driven
- Dials run from 0-9, ten values
- Rightmost counts 1s
- Next counts 10s
- Next counts 100s
- Etc.
Bodometer

1 0 0 0 1 0 1 1

Why bother with 2-9?
- We can do everything with 1s and 0s: bits!

Bodometer

1 0 0 0 1 0 1 1

Dials tick as miles are driven
- Underlying quantity – same as before
- Readout of that quantity in turning dials – new dials and new readout
Bodometer

1 0 0 0 1 0 1 1

Dials tick as miles are driven
• Dials run from 0-1, two values
• Rightmost counts 1s - $2^0$
• Next counts 2s - $2^1$
• Next counts 4s - $2^2$
• Etc.

Reading the readout

1 0 0 0 1 0 1 1
1×128
0×64
0×32
0×16
1×8
0×4
1×2
1×1
Reading the readout

```
0 0 0 1 0 1 1
128 + 8 + 2 + 1 = 139
```

Food for thought

Every number is its own pattern of bits. Every pattern of bits is its own number.

You spend a lot of time looking at decimal numerals
  • What if you’d spent that time looking at binary?
Food for thought

Overflow
• Eventually you count through all your numerals
• Wind up back at zero

Conversion to binary

“Loop”
• Set of instructions you apply over and over

Keep track of two things:
• Number to convert
• Power of two
Conversion to binary

Start with your number and
The largest power of two smaller than it

Example:
   number: 407
   power-of-two: 256

Conversion to binary

Step:
   If number is greater or equal to power-of-two,
   write 1 & subtract power-of-two from number
   otherwise write 0

Example:
   number: 407
   power-of-two: 256
   write 1; number becomes 407-256=151
Conversion to binary

Repeat:
   If power-of-two is 1 stop;
   Otherwise cut power-of-two in half & continue.

Example:
   Continue: number=151, power-of-two=128

Conversion to binary

All of example: That means
407,256,1 407 in binary is
151,128,1
23,64,0
23,32,0 11001011
23,16,1
7,8,0
7,4,1
3,2,1
1,1,1
Decimal Addition

\[
\begin{array}{c}
1 \\
1 & 1 & 5 \\
+ & 1 & 7 & 8 \\
\hline
2 & 9 & 3 \\
\end{array}
\]

Work right to left

Carry when needed

Binary Addition

Just the same:
- Work right to left
- Carry when needed

Only trick: add in binary
- 0+0+0 = 0
- 0+0+1 = 1
- 0+1+1 = 10
- 1+1+1 = 11
Binary addition

\[
\begin{array}{cccccc}
1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
+ & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
\hline
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
\end{array}
\]

Counter logic

<table>
<thead>
<tr>
<th>Two old bits: N</th>
<th>Two new bits: N+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Question

Old number is BC (B twos bit, C ones bit).
Call new ones bit F.
Which is true:

A. F = B.
B. F = not B.
C. F = C
D. F = not C.
E. F = not (B and C).

Counter in logic

New ones bit:
  0 if old ones bit was 1
  1 if old ones bit was 0
F = not C
Counter logic

Old number is BC (B twos bit, C ones bit).
Call new twos bit E.
Which is true:

A. $E = B$ or $C$
B. $E = (B \text{ and not } C) \text{ or (not } B \text{ and } C)$
C. $E = B \text{ and } C$
D. $E = \text{ not (B and C)}$
E. $E = (B \text{ and } C) \text{ or (not } B \text{ and } C)$