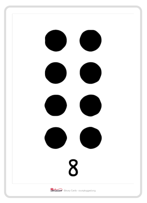
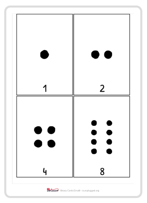
How binary digits work

- **Duration:** 45 minutes
- **Ages 8 to 10:** Lesson 1

Printables



Binary Cards
One set for class demonstration.



Binary Cards (Small)
One set of cards per student.

Classroom resources

- Paper
- Pens
- Whiteboard
- Whiteboard pens

⊖ Learning outcomes

Students will be able to:

- Add numbers to a given amount.
  Mathematics: Numeracy
- Argue that 0’s and 1’s are still a correct way to explain what is stored in the computer.
  Computational Thinking: Abstraction
- Explain how understanding how binary numbers increase supports your knowledge of place value.
  Mathematics: Numeracy
- Explain the logic of why the right-hand bit needs to represent a one.
  Computational Thinking: Logic
- Explain why we can use any two different states or things to represent binary; it doesn’t have to just be 0’s and 1’s.
  Computational Thinking: Abstraction
- Identify even and odd numbers by explaining why the most right number is different to the others.
  Mathematics: Numeracy
- Justify why there aren’t actual 0’s and 1’s zooming around inside a computer.
  Computational Thinking: Abstraction
- Perform a demonstration of how the binary number system works by converting any decimal number into a binary number.
  Computational Thinking: Algorithmic Thinking

Key questions

- What different number systems do we know about? (Answers might include: Roman Numerals; Tally marks; Number bases like binary, octal and hexadecimal; Language based systems like Chinese or Ancient Egyptian.)
- Why do we normally use 10 digits? (Probably because we have 10 fingers, plus it's a fairly efficient way to write things compared with, say, tally marks.)
- Why do we have different number systems? (They are convenient for different things e.g. tally marks are easy if you are counting; Roman numerals can be useful for making a number look more mysterious or harder to read.)

Lesson starter

⊕ See teaching this in action

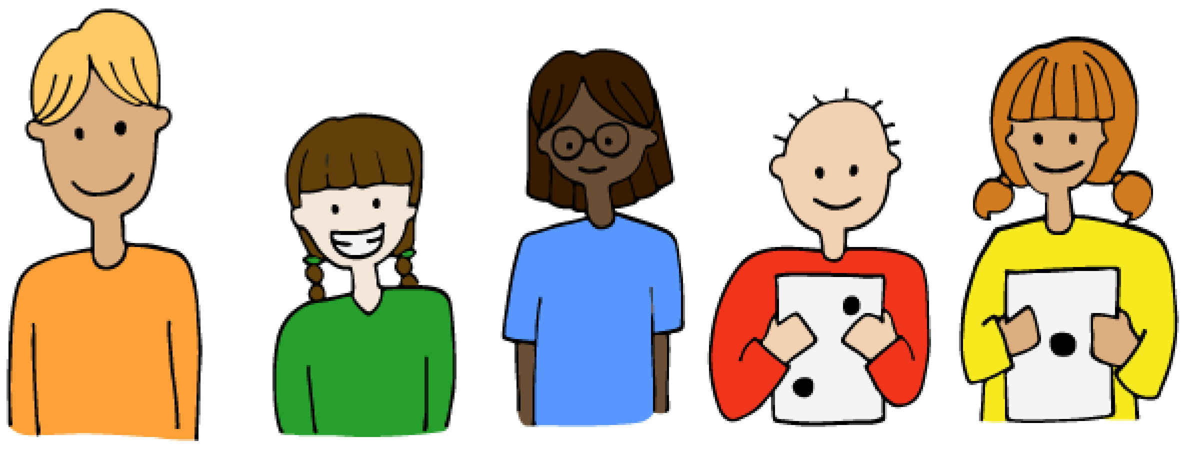
⊖ Note from the authors

We've noticed that once students understand how the binary number system works, they have many questions and are excited to explore the concepts outlined in this lesson further. We've added a lot of information into this lesson, however, it is not our intention that you will teach and cover all the concepts, but that you have at your fingertips the information you need when your students express an interest in learning more.

⊖ Notes on resources

There is also an online interactive version of the binary cards here, from the Computer Science Field Guide, but it is preferable to work with physical cards.

1. Hold the first 5 cards (1, 2, 4, 8 and 16 dots), but don't let students see the dots. Ask for 5 students to volunteer to be "bits", and have them stand in a line in front of the class.
2. Hand out the 1-dot card to the person on the right. Explain that they are one "bit" (binary digit), and can be on or off, black or white, 0 or 1 dots. The only rule is that their card is either completely visible, or not visible (i.e. flipped over). Hand out the second card to the second person from the right. Point out that this card has either 2 dots (visible), or none (upside down).

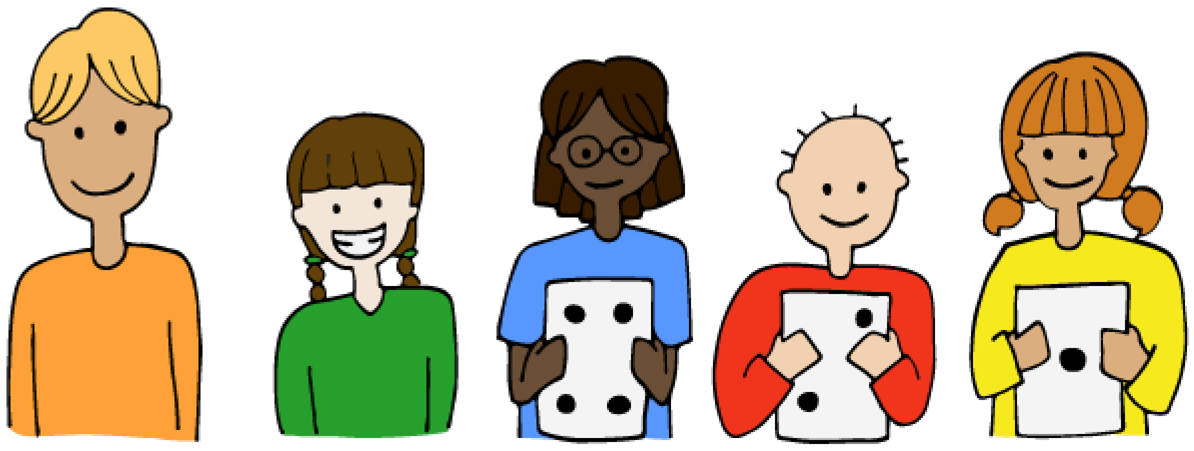


3. Ask the class what the number of dots on the next card will be. Get them to explain why they think that.

⊖ Teaching observations

Students will usually suggest it should be three. If they suggest 4, they have probably done the activity before (or have seen the cards you are holding!) If they suggest the wrong number, don't correct them, but continue without comment, so that they can construct the rule for themselves.

4. Silently give out the four-dot card, and let them try to see the pattern.



⊖ Teaching observations

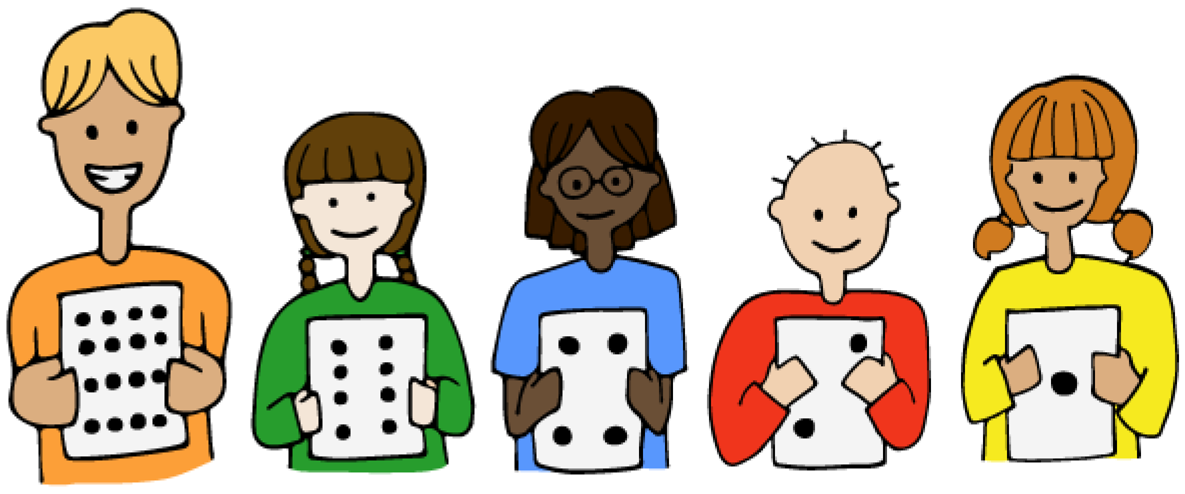
Usually some students will complain that you've missed out the three, but simply indicate that you haven't made a mistake. This gives them the opportunity to try to construct the pattern for themselves.

5. Ask what the next card is, and why.

⊖ Teaching observations

At this point it is common for students to guess that it is 6 (since it follows the numbers 2 and 4). However, if you let them think about it a little more, some will usually come up with 8, and those students should be able to convince the others that they are correct (there are several ways a student could explain this e.g. that each card is double the previous one, or that if you take two of a card, you get the next one)

6. Students should be able to work out the fifth card (16 dots) without help:



7. How many dots would the next card have if we carried on to the left? (32) The next...? (There's no need to have students hold these cards, as they won't be used in the next part of the activity, but you can show them to confirm that they are correct).

8. Continue with 64 and 128 dots.

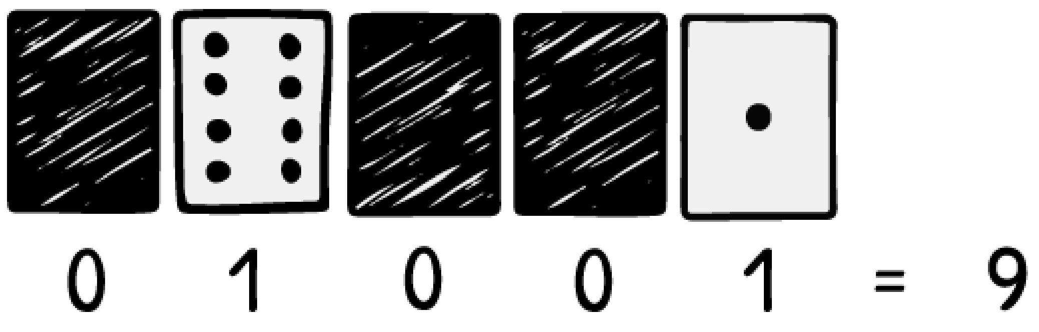
⊖ **Teaching observations**

At 128 dots there would be 8 cards. This is 8 bits, which is commonly referred to as a byte. It may be distracting to bring this up at this point, but some students may already be familiar with the idea that 8 bits is a byte, and make that observation. However, in the meantime, we'll work with a 5-bit representation, which isn't as useful as a whole byte, but a good size for teaching. (A byte is a convenient grouping of bits, and usually computer storage is based around bytes rather than individual bits; it's just the same as eggs being sold as a dozen; they could be sold individually, but groups of a dozen are usually more convenient for everyone concerned.)

A common mistake is to hand out the cards from left to right, but it's convention in number representation that the least significant value is on the right, and this is an important idea for students to take away from this activity.

Lesson activities

1. Remind the students that the rule is that a card either has the dots fully visible, or none of them are visible. If we can turn cards on and off by showing the front and back of the card, how would we show exactly 9 dots? Begin by asking if they want the 16 card (they should observe that it has too many dots), then the 8 card (they will likely reason that without it there aren't enough dots left), then 4, 2 and 1. Without being given any rules other than each card being visible or not, students will usually come up with the following representation.



⊖ **Mathematical links**

Base 10 (our counting system) has 10 digits, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. When we count in base 10, we count from 0 to 9 and then run out of digits. So we need to add another column; we put a 1 in that column and start counting again from 0. This makes the number 10, we then repeat that process until the tens column is 9 and the ones column is 9 (making 99); from there we then add another column. Hence we have the familiar place value system that can be shown something like this:

100,000s | 10,000s | 1,000s | 100s |10s | 1

Note: Use the appropriate place value example based on what you have already taught in your class; this is an extended example.

Base 2 (binary) follows the same logic, except it moves a lot quicker to the “next” place value, because there are only two digits, 0 and 1. The binary place values look like this:

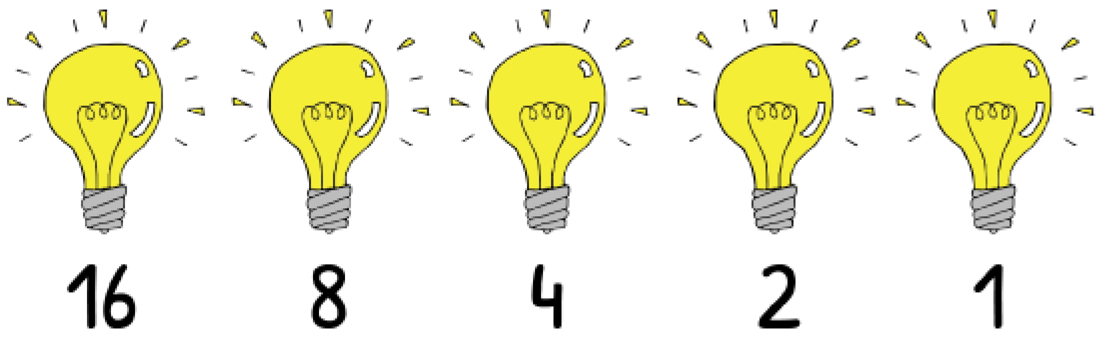
32 | 16 | 8 | 4 | 2 | 1 |

Sometimes students confuse the order of digits in a binary representation. To support students to understand the correct ordering of binary digits, ask the question: If I was going to give you $435.00 which number are you most interested in? Is it the 4 or the 5? Why is that? It's the same for binary code, the lowest value (least significant digit) is on the far right, whereas the most significant digit is on the far left.

2. Now ask "How would you make the number 21?" (Again, start by asking if they want the 16 card, then the 8 card, and so on from left to right).

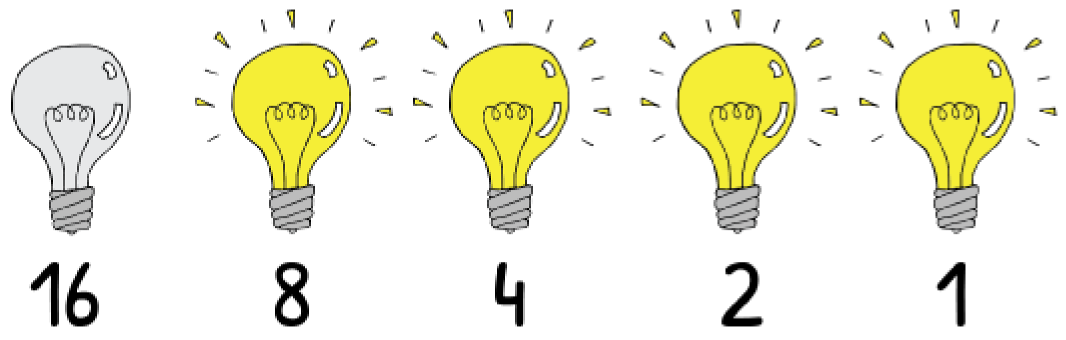
3. This is an algorithm for converting numbers to a binary representation. Let's think through the steps to do this together.

   a. Start with all the numbers switched to on (dots showing).



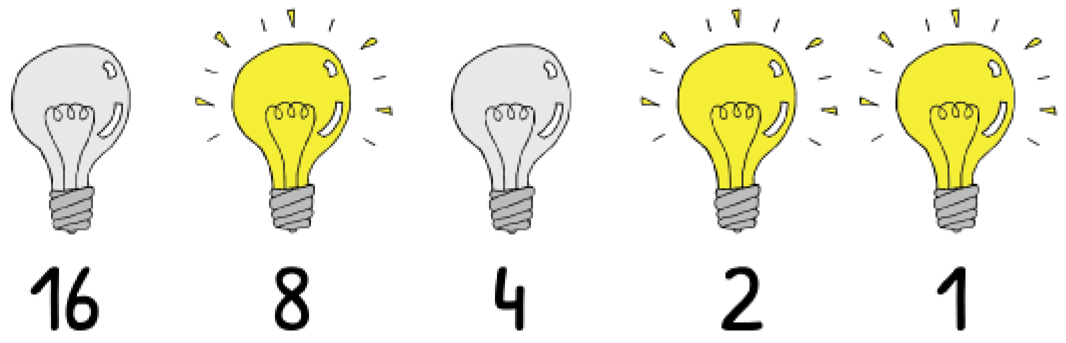
   b. Consider representing the number 10

   c. Does 16 fit into 10? No - so turn it off



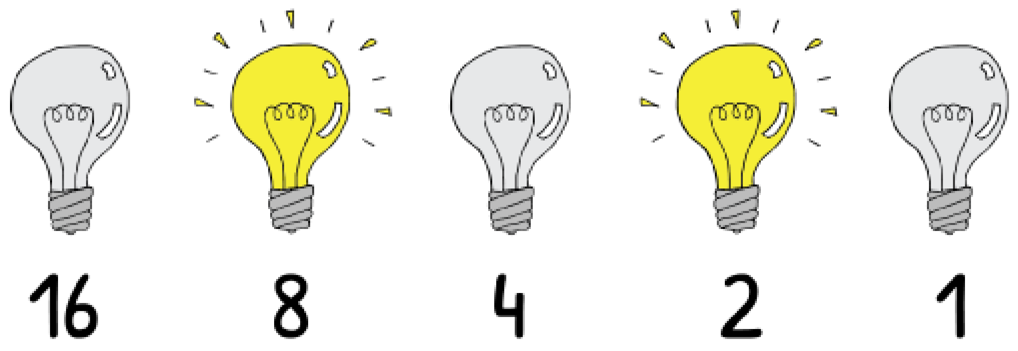
   d. Does 8 fit into 10? Yes - so keep it on. How many are left over? (2)

   e. Does 4 fit into 2? No - so turn it off



   f. Does 2 fit into 2? Yes - so keep it on. How many are left over? (None)

   g. So turn off 1.



Applying what we have just learnt

- Group students into pairs.
- Give each pair a set of the smaller binary cards (either 5 or 6 cards, depending on the range of numbers they are comfortable with).
- Starting with just 5 cards, have them practise the algorithm (deciding about each card from left to right) for numbers such as 20, 15, and 8.

1. Explain to students that we're working with just two digits, so they are called binary digits. They are so common that we have a short name for them: write "binary digit" on a piece of paper, then rip off the "bi" at the start, and the "t" at the end, put it together and ask what the combined word ("bit") spells. This is the short name for a binary digit, so the 5 cards that they have are actually 5 bits.

2. Now let's count from the smallest number we can make up to the highest number:

   a. What is the smallest number? (they may suggest 1, then realise that it's 0).

3. Get the number zero displayed on the cards (i.e. no dots showing).

4. Now count up 1, 2, 3, 4 .... (each pair should work out these numbers between them).

5. Once they start to get into a routine, ask: how often are we seeing the 1-dot card? (every second time, which is every odd number)

   a. What other patterns are we seeing? (some may observe that the 2-dot card flips on every second count, the 4-dot on every 4th and so on; so the 16 dot card doesn't do much!)

6. Continue until all the cards are switched to "on" and have counted to 31. What happens next? (We have to add a new card.) How many dots on it? (32) What do we have to do to the other 5 cards when we get to 32? (we have to turn them all off)

7. Let's explore this further ...

   a. So when I have two bits I can make a maximum of? (3)

   b. I add another bit and that has how many dots on it? (4)

   c. I turn off the first two bits to make 4 right?

   d. Now let's turn on all three bits, so now we have how many? (7)

   e. I add another bit and that has how many dots on it? (8)

   f. Repeat until a pattern is recognised that the number on the next card to the left is one more than the total number of dots on all the cards to the right (e.g. there are 15 dots on the 8, 4, 2 and 1 cards, so the next card to the left is 16). This makes it easy to work out the number if all the bits are switched on - double the left-hand card, and subtract 1.

   g. How many different numbers can I make with two bits? (4; often students will say 3 because they haven't counted 0)

   h. Let's add the next bit; how many different numbers can we make now? (8, again 7 will often be given as an answer first)

   i. Repeat until a pattern is recognised that each time we add another bit we can now represent twice as many numbers.

Teaching observations

A concept that students may struggle with here is that the number of values is one more than the maximum value (e.g. from 0 to 7, there are 8 different numbers). The same observation occurs with the number of digits in conventional decimal numbers; the largest digit is 9, but there are 10 possible digits (counting 0). This is sometimes called the fencepost problem (the number of fence posts is one more than the number of gaps between them), and it comes up a lot in computing.

Lesson Reflection

- Would this activity work if we used white and cream cards? Why? Why not? (In principle you could use these, but it wouldn't be a good idea. We are looking for the answer that they are not contrasting colours, therefore it would be difficult to see if it is actually on or off. This hints why computers use easily distinguished physical representations.)

- What are some contrasting symbols or ways that we can show on and off in binary?

  - (Ideas could include holding the cards up high, or down low; simply holding up a hand; sitting down or standing up; or using a different representation such as lights that are on or off.)
- Computers are cheaper and easier to build if they represent data with just two contrasting values, which we represent as the numbers 0 and 1. What else could we use to represent two opposites in writing? (Perhaps a cross or tick; happy or sad face; or any other pair of symbols.)

- Extending this idea, the numbers might be represented by a voltage that is either close to 5 volts, or close to 0 volts. The circuitry is built so that anything less than about 2.5 volts counts as 0 and anything over 2.5 volts counts as 1. Like the contrasting colours of the cards, this is very easy to recognise. We could have had 10 colours of cards to represent the digits from 0 to 10, and we could have ten voltage ranges (0 to 0.5, 0.5 to 1.0 and so on), but it's way more complicated to build fast and accurate circuitry for this.

Seeing the Computational Thinking connections

Throughout the lessons there are links to computational thinking. Below we've noted some general links that apply to this content.

Teaching computational thinking through CSUnplugged activities supports students to learn how to describe a problem, identify what are the important details they need to solve this problem, break it down into small logical steps so that they can then create a process which solves the problem, and then evaluate this process. These skills are transferable to any other curriculum area, but are particularly relevant to developing digital systems and solving problems using the capabilities of computers.

These Computational Thinking concepts are all connected to each other and support each other, but it's important to note that not all aspects of Computational Thinking happen in every unit or lesson. We've highlighted the important connections for you to observe your students in action. For more background information on what our definition of Computational Thinking is see our notes about computational thinking.

⊖ Algorithmic thinking

We used an algorithm in this lesson to convert a decimal number to a binary one. This is an algorithm because it is a step-by-step process that will always give the right solution for any input you give it as long as the process is followed exactly.

Here's an algorithm for working out which dot cards should be showing, written in text:

- Find out the number of dots that is to be displayed. (We'll refer to this as the "number of dots remaining", which initially is the total number to be displayed.)
- For each card, from the left to the right (i.e. 16, 8, 4, 2 then 1):
    - If the number of dots on the card is more than the number of dots remaining:
        - Hide the card
    - Otherwise:
        - Show the card
        - Subtract the number of dots on the card from the number of dots remaining

Note that this algorithm (working from right to left) works very well with the cards, but if you look up computer programs for doing this, you may encounter a different one that works from right to left. It's usual to have multiple algorithms that achieve the same thing.

Examples of what you could look for:

Which students are methodical when they convert between decimal and binary? Which ones start with the leftmost card and move one card at a time to the right, rather than choosing cards at random and flipping them on and off until they get the right number?

⊖ Abstraction

Binary number representation (just using 0 and 1) is an abstraction that hides the complexity of the electronics and hardware inside a computer that store data. Abstraction helps us simplify things because we can ignore the details we don't currently need to know.

In this case the details we can ignore include: Computers use physical devices like electronic circuits and voltages in circuits to store and move data, and there are many complex physics and mathematical theories that make this work.

We don't need to understand how these circuits work to use data and represent things using binary. Using binary is an Abstraction of these circuits and allows us to represent numbers as being made out of bits (0s and 1s), to understand data and work out problems without having to think about what is happening 'underneath the hood' of the computer.

Another use of abstraction is considering what is needed to represent any given digit in binary. The answer is all you need are two different things. These things can be anything! Two different colours, two different animals, two different symbols etc. As long as there are two of them, and they are different, you can use these to represent any number, using binary, the same way a computer uses electricity to represent data.

We can use binary digits to represent any type of data stored on a computer. When we represent other forms of data (such as letters, images, and sound) we also use abstraction because we hide the details of all the binary numbers underneath and just look at the whole piece of data. All forms of data end up being represented as numbers (which in turn are really just combinations of bits) - for text we have a number for each letter, for images we use a number for each colour, and so on. We are using multiple layers of abstraction! For example, a familiar form of abstraction is that the month "October" could be represented by the number ten, which in turn is represented by the bits 01010, and if these are stored as voltages in computer memory, it is ultimately "low, high, low, high, low" for the voltages.

Examples of what you could look for:

Who are the students that demonstrate converting and representing binary numbers using things other than “1’s and 0’s”, “black and white”, and “off and on” (for example using :) and :(, or using people standing up or sitting down). If you are able to interchange terms like "black" and "white" with 0 and 1 without students being concerned about the difference, they are exercising abstraction.

⊖ Decomposition

An example of decomposition is breaking the conversion of the number to binary into one bit at a time. The questions "Should this be 1 or 0" for each of the dot cards is decomposing the problem to a series of questions.

Examples of what you could look for:

Which students recognise that it is important to start with the leftmost card and only consider one bit at a time? Which students focus on each individual bit at a time, rather than being overwhelmed by trying to work them all out in one go?

⊖ Generalising and patterns

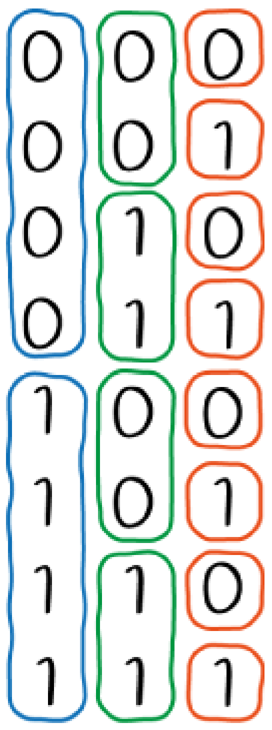
Recognising patterns in the way the binary number system works helps give us a deeper understanding of the concepts involved, and assists us in generalising these concepts and patterns so that we can apply them to other problems.

At a simple level, we started with the numbers 1, 2, and 4, and students generalised that to doubling values. The exercise used 5-bit numbers, but students should be able to generalise that to 8-bit numbers, or larger.

The algorithm for converting a decimal number to a binary one follows a pattern that can be generalised to solve the problem of giving change when someone pays by cash. For binary numbers you start with the largest bit always turn a bit on if you need it, just like when you’re giving change you start with the largest denomination and then always take a coin (or note) whenever you need it. Jargon note: This is called a greedy algorithm - it takes as much as it can each time!

⊕ Mathematical links

When counting upwards in binary, there is a pattern for how often particular cards flip. The 1st bit (with 1 dot) turns over every time, the 2nd (with 2 dots) turns for every second number, the 3rd (with 4 dots) turns for every 4th... Is there a pattern like this when we count in decimal numbers?



If you have 5 of the cards and all are visible, you will have the number 31, which is 1 less than the value of the next card, 32. Is this pattern always true?

The amount of numbers you can represent with a certain number of bits is the same as the value of the next bit that can be added. For example, using 4 cards (1, 2, 4, 8) you can represent 16 different numbers (0-15), and the next card in the sequence is the number 16. Each time we add the next card we also double the amount of different numbers we can represent.

Working with these patterns is valuable for working out the relationship between the number of bits being used and the power of what they can represent.

Explain one or more of the following patterns:

- That with a certain number of cards you can make the same amount of different numbers as the number of dots that would be on the next card to be added on the left (remember that 0 is a number).
- When you are counting upwards: the first card (1 dot) turns over every time, the second card (2 dots) turns every two times, the third (4 dots), every four times, and the fourth (8 dots), every eight times...
- That when all the cards you have are visible it will add up to the next binary card number minus 1.

Examples of what you could look for:

Which students recognised quickly that each card was doubling the number of dots? Can students see the similarities between this and multiplying place values by 10 when they are using the decimal system?

Which students easily understand the patterns of cards flipping when counting with binary numbers?

⊖ Logic

Logical thinking means using rules you already know and using logic to deduce more rules and information from these. Once we know what number each of the binary cards represents then we can use this knowledge to figure out how to represent other numbers with the cards. If you memorise how to represent the numbers we can make with 5 cards, does that mean you understand how to represent any number with any number of bits? It doesn’t, but you can understand how to do that if you understand the logic behind how these numbers with the 5 cards are made.

A good example of logical thinking in binary numbers is the reasoning for why each bit "has to" have a particular value (e.g. it has to be 1, or it has to be 0) to represent a given number. This in turn leads to understanding that there is only one representation for each number.

Examples of what you could look for:

Do students explicitly explain that the right-most bit needs to be a one because it is the only odd number and therefore is needed so that we are able to make any, and all, odd numbers? Without it we could only make even numbers.

Are students able to explain that each card "has to" be up the way it is for a given number e.g. the 16-dot card is needed for the number 19 because without it you only have 15 dots remaining to its right (not enough); but the 16 card isn't needed for the number 9 because it would give too many dots?

⊖ Evaluation

An example of evaluation is working out how many different values can be represented by a given number of bits (e.g. 5 bits can represent 32 different values), and vice versa (to represent 1000 different values, you need at least 10 bits).

Examples of what you could look for:

Can a student work out the range possible with 4 bits? (16)

6 bits? (64)

8 bits? (256)

If we add one more bit to a representation, how much does that increase the range? (it doubles it)

If we add two more bits to a representation, how much does that increase the range? (it is four times as much)

How many bits do we need to represent 1000 different values? (10 is sufficient)