Central Processing

Central Processing, or perhaps *Central* for short, is a collaborative game made to teach some fundamentals of binary logic and computer operations, intended to be fun even when the concepts are understood.

players: 2
duration: about 10 to 15 min per round
audience: general, 10+ years of age suggested

Minimal materials needed: a standard deck of 52 playing cards and 4 jokers*, a coin to flip, paper and a writing implement for short notes, the printed reference sheet.
Optional materials: access to this website to randomly generate 8-bit strings, a custom deck of cards.

*Note that many standard 54 card decks that contain only 2 jokers will actually have 56 cards, with two of them being decorative start and end cards. This deck will also work, once players decide which side of the decorative cards is the “1” and the “0” side.

About binary

Binary is a system of writing numbers in which each binary digit, or "bit," is either 0 or 1. So, for example, the number one is just written as 1, but the number two is written as 10, because you can’t increment the ones place any more.

In this game, we call a string of bits a register, which you'll represent as a row of eight cards, with 0 on one side and 1 on the other.

Computers work by doing various different operations on various different operands. Operands are what we call the input or output of an operation. For example, in the operation "add x to y," the operator is "add", and the operands are "x" and "y". In Central Processing, we'll focus on some of the most common binary operators, specifically, the ones named NOT, AND, OR, and XOR. It's easiest to understand them with the truth tables at the bottom of these rules, but here's a verbal explanation:

- NOT is the simplest operator. it has one operand, which is both an input and an output. it simply flips all the bits in this operand. In central processing, you apply the NOT operand on a register, and when doing so you flip all the cards in the register.
- AND is another operator, but it takes both input and output operands. . When you apply AND to an input operand and an output operand, every bit in the output operand will be 1 only if both corresponding input bits are 1. otherwise, it's 0.
- OR sets each output bit to 1 if, inclusively, the bit of the first or second operand is 1, i.e. if at least one of the input bits is 1.
- XOR sets each output bit to 1 if, exclusively, the bit in either the first or the second operand is 1, i.e. if the input bits differ.

Minimal setup (with regular playing cards and no internet):

1. Remove cards with values 1 (A) through 8. Separate out the hearts and spades and put them aside in a pile. These are called immediate operands, or *immediates* for short.
2. With the remaining diamonds and clubs of values 1 through 8, distribute one suit to either player, forming the player registers. Order them from 1 to 8 from right to left from the perspective of the respective player.
3. From the rest of the cards, remove the 4 jokers and 4 kings, and place them in the center, forming the central register.
4. Remaining should be all the cards of values 9 through Q. These are *operators*. Shuffle these with the immediates (hearts and spades with values 1-8), forming the draw pile.
5. Now we will generate and record our "goal state."
    - If you have an internet connection, you may follow step 4 in the custom setup.
    - Otherwise, pick up the 4 jokers and 4 kings that form the central register.
        - For an easier game, flip two of them face down.
        - For a harder game, flip four of them face down.
    - Then shuffle the cards without looking and place them in order on the table. Record this goal state as a string of 1s and 0s on your piece of paper and place it near the central register.
    - Note that the goal state will look different to each player depending on their orientation. If it reads 00000001 for one player, it will read 10000000 for the other.
6. Now, we will randomize the registers.
    - If you have an internet connection, you may use the results from step 5 in the custom setup.

    - Otherwise, for each card in the central register, flip a coin; if heads, place the card face up, and if tails, place it face down. Then do the same for both of the player registers.
7. Deal six cards to each player from the draw pile. Decide who goes first.

Custom setup (requires custom-printed cards, and internet access):

1. Some cards will have a 1 on one side, and a 0 on the other. These are *bits*. Distribute eight bits in a line in front of each player. Each line of bits is a player register.
2. Distribute the last eight bits in the center in a line. This is the central register.
3. Shuffle all remaining cards together, forming the draw pile.
4. Now we will find and record our "goal state." Navigate to https://imyxh.net/central_processing and use the randomizer tool.
    - If you wish to control your difficulty, select it from the dropdown menu and click "generate custom goal."
    - For a random difficulty, simply click the "randomize!" button.
    - Either way, read the string of eight bits to the right of "goal state" and write them down on a piece of paper near the central register.
    - Note that the goal state will look different to each player depending on their orientation. If it reads 00000001 for one player, it will read 10000000 for the other.
5. Now, we will randomize the registers.
    - Hit "randomize!" if you haven't already, and flip over cards in each of the three registers until they match what is shown on the randomizer.

8. Deal six cards to each player from the draw pile. Decide who goes first.

Mechanics

1. A *register* is a row of 8 bits (cards) ordered in a line, with each card having the value of one (face-up) or zero (face-down).
2. Each player has a register, and there is one register in the center. All the registers are randomized at the start of the game.
3. There is a draw pile for the player to draw from which contains *operators* (cards from 9-Q) and immediate operands or *immediates* (red cards from 1-8).
4. The operators are AND (9), OR (10), XOR (J), and NOT (Q), and their functions are explained in the operator table below. They are played on two registers, from one to another, to modify the latter register's bit values. They can also be played directly onto a register using an immediate card as an argument. (Note that NOT is an exception to these, and is only played on one register with no immediates.)
5. Immediates are used as operands; for example, a red 4 represents the string 11110111 (i.e. the only zero is the fourth one from the right). This replaces the "first register" for the operator, and so can be played directly onto a player register or the central register. See the chart listed below for the binary strings for each immediate.
6. On a player's turn, they may play as many cards from their hand as they would like. Playing cards consists of playing an operator and optionally an immediate at a time, following the operator table below. Players end their turn by drawing until they have six cards.
7. One of the two mechanics will be selected:

    a. <u>For beginner players:</u> players may, on their turn, also discard cards freely as they wish.
    b. <u>For advanced players:</u> players may not freely discard cards, but on their turn, the player may *push* a card to the *stack*, meaning they place one of the cards in their hand on the stack. The stack is created next to the draw and discard piles. When a player draws to conclude their turn, if they did *not* push to the stack, they can—for each card they draw—choose to *pop* from the stack, meaning draw the card from the top of the stack instead of the draw pile. This mechanic allows players to exchange cards with each other with some strategizing.
8. Players may talk freely and strategize together about their cards. They can talk about their cards, but they may not show each other their hand.
9. Both players win when the central register matches the goal state. Both players lose when all the cards have been drawn from the draw pile and there is no way to win.

Player's Turn:

1. You *must* do one or more of these actions in any order:
    a. Play an operator on two registers or use an immediate and operator on a register, or play a NOT on a register.
    b. <u>For beginner players</u>: discard as many cards as you please.
    c. <u>For advanced players</u>: push a card in your hand to the stack.
2. After all desired actions have been completed, draw from the draw pile and/or the stack (if playing with it) until you have 6 cards in your hand.

Additional Reminders:

- This game is cooperative, and so players may talk about their cards and their team strategy freely. They may not, however, show each other their hands.
- Players can decide who goes first after looking at their cards and discussing them.
- Players must take at least one action per turn.
- For the playing card version, note that bits are indexed from right to left.
- Players cannot push and pop from the stack in the same turn; in essence, if they push to the stack in their turn, they can only draw from the draw pile.
- Advanced players may wish to attempt multiple goal states in succession.

Operator table

9: **AND**

IN1	IN2	OUT
1	1	1
1	0	0
0	1	0
0	0	0

10: **OR**

IN1	IN2	OUT
1	1	1
1	0	1
0	1	1
0	0	0

J: **XOR**

IN1	IN2	OUT
1	1	0
1	0	1
0	1	1
0	0	0

Q: **NOT**
(use on one register, without an immediate)

IN	OUT
1	0
0	1

Immediates table

Black cards: (if n is the card number, then 2^{n-1} is 1, and the rest are 0)

- 1: 0000 0001
- 2: 0000 0010
- 3: 0000 0100
- 4: 0000 1000
- 5: 0001 0000
- 6: 0010 0000
- 7: 0100 0000

- 8: 1000 0000

Red cards: (if n is the card number, then 2^{n-1} is 0, and the rest are 1)

- 1: 1111 1110
- 2: 1111 1101
- 3: 1111 1011
- 4: 1111 0111
- 5: 1110 1111
- 6: 1101 1111
- 7: 1011 1111
- 8: 0111 1111