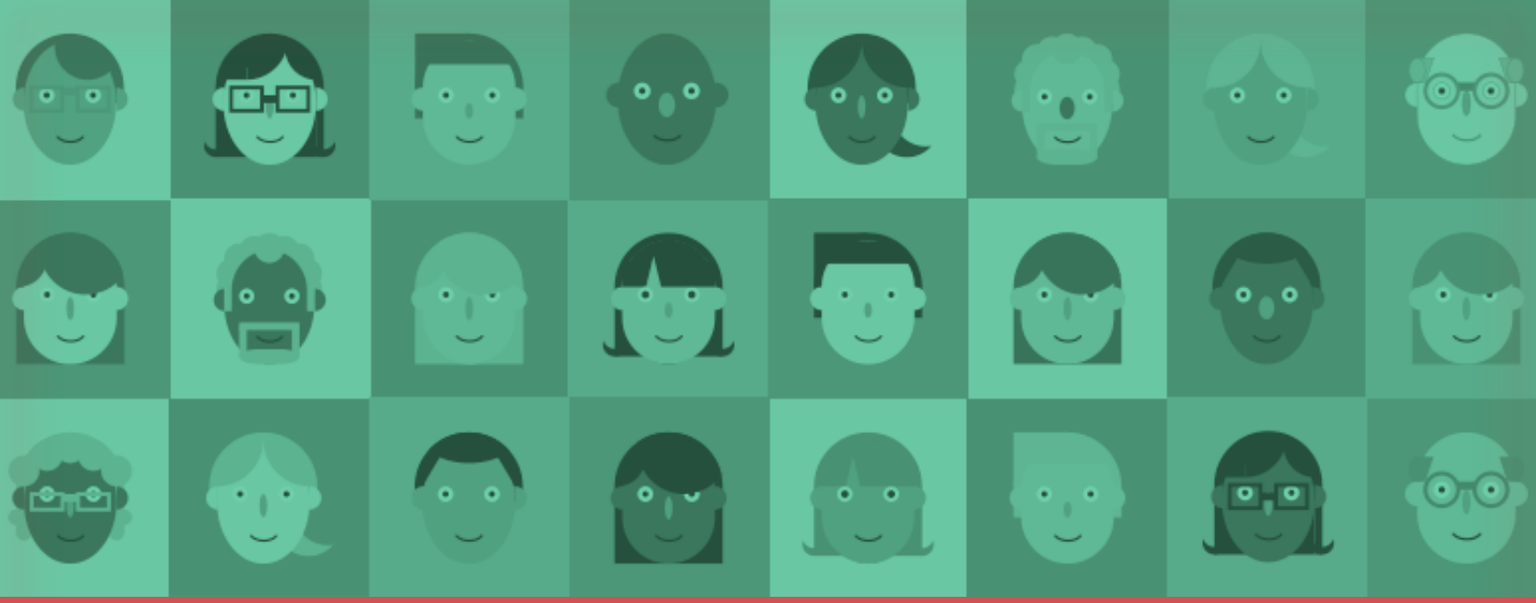


COMPUS

THE COMPUTER IS US!

1-4

THE STRATEGY GAME
WHERE WE ARE THE COMPUTER!

8-99

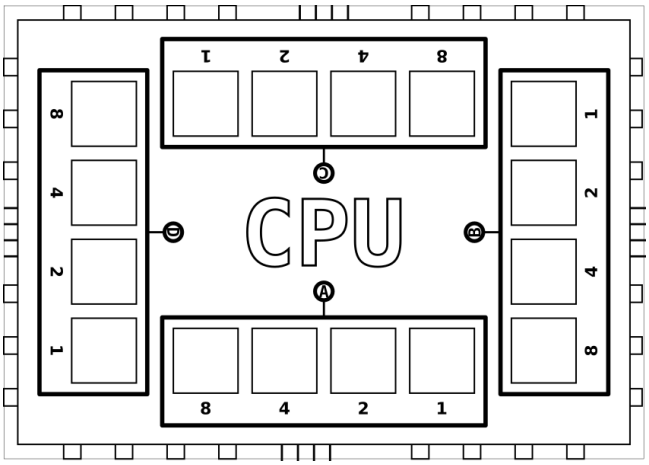
COMPUS

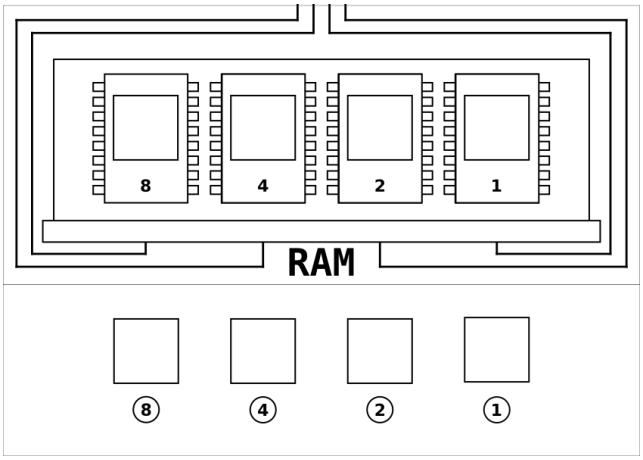
COMPUS es un juego para 1 a 4 personas, de 8 años en adelante.

En COMPUS cada participante simulará ser un programa informático dentro de una computadora y tratará de obtener el resultado deseado (una combinación de bits) antes que el resto.

Componentes

- 1 tablero central (CPU).
- 4 tableros individuales (RAM).
- 60 cartas de operación.
- 48/96 bits (0/1).
- 1 dado.





Preparación

Tanto el tablero central (CPU) como los tableros individuales (RAM) tiene dos caras, una para jugar con 4 bits y la otra con 8 bits. Para las primeras partidas sugerimos los tableros de 4 bits.

- En el tablero central (CPU), se coloca un bit a cero en todas las posiciones de los registros A, B, C y D.
- Cada participante elige un color, toma el tablero individual de ese color y coloca un bit a cero en todas las posiciones de su RAM (parte superior del tablero individual).
- Cada participante define la combinación de bits que será su resultado deseado de la siguiente manera: 1) lanza un bit al aire como si fuera una moneda; 2) coloca el resultado (cero o uno) en la casilla con un valor más bajo en la parte inferior de su tablero individual; 3) repite el proceso hasta completar todas las casillas de resultado. El resultado deseado cada participante permanecerá a la vista durante toda la partida.
- Se baraja el mazo de cartas de operación y se reparten 5 a cada participante. Las cartas de operación de cada participante se mantendrán en secreto para no desvelar su estrategia durante la partida.

La persona que saque el resultado que tenga más unos es la primera en jugar. La partida continúa, por turnos, hacia la izquierda de esa persona.

Jugar

El objetivo del juego es conseguir el resultado deseado en el registro A de la CPU. La primera persona que logre su objetivo gana la partida.

En cada turno, cada participante deberá hacer lo siguiente:

- Conocer cuánto tiempo de CPU tiene. Para ello tirará el dado y sabrá si dispone de 1, 2 o 3 unidades de tiempo (en un dado de 6 caras: 1-2 corresponde a 1 unidad de tiempo, 3-4 a 2 unidades de tiempo y 5-6 a 3 unidades de tiempo; en un dado de 4 caras: 1, 2, 3 corresponden a 1, 2, 3 unidades de tiempo y 4 implica repetir la tirada).
- Jugar tantas cartas de operación como unidades de tiempo se hayan conseguido en el paso anterior. No es obligatorio

jugarlas todas, es posible ceder el tiempo de CPU que no se desee usar.

- Tomar del mazo tantas cartas de operación como se hayan jugado (no podrán usarse hasta el siguiente turno).

Las cartas de operación realizan operaciones sobre los registros A, B, C y D de la CPU. Cualquier participante puede modificar los valores de todos los bits en los 4 registros de la CPU, pero no podrán leer o modificar los valores almacenados en la RAM del resto de participantes.

Estas son las operaciones que pueden realizarse utilizando cartas de operación:

NOT: negación lógica

Se aplica sobre un solo registro. Implica negar cada uno de sus bits, es decir, convertir los ceros en unos y viceversa. A efectos prácticos, esto supone dar la vuelta a todos los bits de un registro.

Ejemplo: si en el registro A se almacena el valor 1101, al hacer un NOT sobre este registro, el valor de A será 0010.

ROL: rotación hacia la izquierda

Se aplica sobre un solo registro. Implica desplazar cada uno de sus bits hacia la izquierda y colocar el bit que sobra por la izquierda en la posición más la derecha.

Ejemplo: si en el registro A se almacena el valor 0110, al hacer un ROL sobre este registro, el valor de A será 1100.

ROR: rotación hacia la derecha

Se aplica sobre un solo registro. Implica desplazar cada uno de sus bits hacia la derecha y colocar el bit que sobra por la derecha en la posición más la izquierda.

Ejemplo: si en el registro A se almacena el valor 0011, al hacer un ROR sobre este registro, el valor de A será 1001.

INC: incremento

Se aplica sobre un solo registro. Implica sumar 1 al valor total del registro. El valor total de un registro se calcula sumando los valores decimales que se indican debajo de cada bit (1, 2, 4, 8, etc.). Si todos los bits del registro están a uno, al sumarle 1,

se produce un desbordamiento que implica volver al valor 0 (todos los bits a cero).

Ejemplo: si en el registro A se almacena el valor 1011 (8 + 2 + 1 = 11 en decimal), al hacer un INC sobre este registro, el valor de A será 1100 (8 + 4 = 12 en decimal).

DEC: decremento

Se aplica sobre un solo registro. Implica restar 1 al valor total del registro. El valor total de un registro se calcula sumando los valores decimales que se indican debajo de cada bit (1, 2, 4, 8, etc.). Si todos los bits del registro están a cero, al restarle 1, se produce un desbordamiento que implica volver al valor máximo del registro (todos los bits a uno).

Ejemplo: si en el registro A se almacena el valor 1100 (8 + 4 = 12 en decimal), al hacer un DEC sobre este registro, el valor de A será 1011 (8 + 2 + 1 = 11 en decimal).

MOV: copia de bits

Se aplica sobre dos registros o un registro y la RAM (es la única operación que permite el uso de la RAM). Implica copiar cada uno de los bits de un lugar a otro. Podemos copiar bits de A a B, C o D y viceversa, o de A, B, C o D a nuestra RAM y viceversa. Es una operación muy útil para guardar un valor interesante para nuestra estrategia en nuestra RAM (ahí no podrá modificar nadie su valor) o para ganar la partida si nuestro resultado deseado se encuentra en un registro diferente a A y lo copiamos ahí.

Ejemplo: si en el registro A se almacena el valor 1110 y en el registro B el valor 0110, cuando hacemos un MOV de B a A, B mantiene su valor y A pasará a almacenar el valor 0110.

OR: disyunción lógica

Se aplica sobre dos registros. Implica realizar una suma lógica bit a bit de los dos registros y guardar el resultado en el primero de ellos. Para realizar una suma lógica bit a bit tomaremos el bit más hacia la izquierda de ambos registros y comprobaremos si al menos uno de los dos contiene un 1, si es así, el resultado para ese bit será 1; si ninguno de los dos es un 1, el resultado para ese bit será un 0. Continuamos con el siguiente bit de ambos registros hasta completar todos. Es importante recalcar que en un OR no hay acarreo o "llevadas": si ambos bits

son 1 el resultado será 1 pero no influirá en el cálculo del siguiente bit.

Ejemplo: si en el registro A se almacena el valor 1010 y en el registro B el valor 0110, cuando hacemos un OR de A y B, B mantiene su valor y A pasará a almacenar el valor 1110.

AND: conjunción lógica

Se aplica sobre dos registros. Implica realizar un producto lógico bit a bit de los dos registros y guardar el resultado en el primero de ellos. Para realizar un producto lógico bit a bit tomaremos el bit más hacia la izquierda de ambos registros y comprobaremos si ambos contienen un 1, si es así, el resultado para ese bit será 1; en caso contrario, el resultado para ese bit será un 0. Continuamos con el siguiente bit de ambos registros hasta completar todos.

Ejemplo: si en el registro A se almacena el valor 1010 y en el registro B el valor 0110, cuando hacemos un AND de A y B, B mantiene su valor y A pasará a almacenar el valor 0010.

XOR: disyunción exclusiva

Se aplica sobre dos registros. Implica realizar una disyunción exclusiva bit a bit de los dos registros y guardar el resultado en el primero de ellos. Para realizar una disyunción exclusiva bit a bit tomaremos el bit más hacia la izquierda de ambos registros y comprobaremos si ambos contienen un valor diferente (en uno hay un 1 y en el otro hay un 0 o viceversa), si es así, el resultado para ese bit será 1; en caso contrario, el resultado para ese bit será un 0. Continuamos con el siguiente bit de ambos registros hasta completar todos.

Ejemplo: si en el registro A se almacena el valor 1010 y en el registro B el valor 1110, cuando hacemos un XOR de A y B, B mantiene su valor y A pasará a almacenar el valor 0100.

Trucos

- No operar (NOP): en ocasiones nos interesará utilizar alguna de nuestras cartas sin cambiar el tablero, con la única intención de descartarnos de ella. En ese caso, es posible hacer operaciones que simulan un NOP (no operar), como por ejemplo: hacer un OR de A y A, hacer un AND de A y A, o hacer un MOV de A a A. También es posible descartarse de dos cartas

complementarias, como por ejemplo: hacer dos NOT de A, hacer un ROL y un ROR de A, o hacer un INC y un DEC de A.

- Reinicio de un registro: si alguna vez queremos reiniciar un registro (todos sus bits a cero), podemos hacer un XOR consigo mismo. Dado que XOR pone a 1 todos los bits diferentes y que un registro siempre es igual a sí mismo, el resultado será siempre cero.
- Multiplicaciones y divisiones: en muchas ocasiones, un ROL supondrá una división entre 2 del número que contiene el registro, y un ROR una multiplicación por 2. A veces es más fácil pensar en decimal para lograr nuestro objetivo.

Variantes

Complejidad

Para reducir la complejidad inicial del juego recomendamos comenzar con la versión de 4 bits del tablero. Asimismo, es posible facilitar las primeras partidas retirando algunas de las cartas de operación más complicadas como las operaciones matemáticas INC y DEC o las operaciones lógicas OR, AND y especialmente la operación XOR. Si hubiera grandes diferencias entre participantes, es posible asignar diferente tiempo de procesador a cada participante (por ejemplo: permitir siempre realizar 3 operaciones en cada turno al participante más novato).

Juego en solitario

La principal diferencia para el juego en solitario reside en que no habrá intromisiones a nuestra estrategia por parte de otras personas. Podemos jugar así y tratar de minimizar el número de cartas empleadas para lograr el resultado deseado, o bien tomar una carta del mazo al final de cada turno y aplicarla sobre el último registro utilizado (o los dos últimos registros en el caso de operaciones de dos registros) para simular las intromisiones en nuestra estrategia.

Implicaciones didácticas

Desde el punto de vista didáctico, COMPUS permite conocer muchos conceptos importantes sobre el funcionamiento interno de una computadora.

El primero de ellos tiene que ver con la arquitectura del sistema: las CPU o microprocesadores reales también disponen de registros

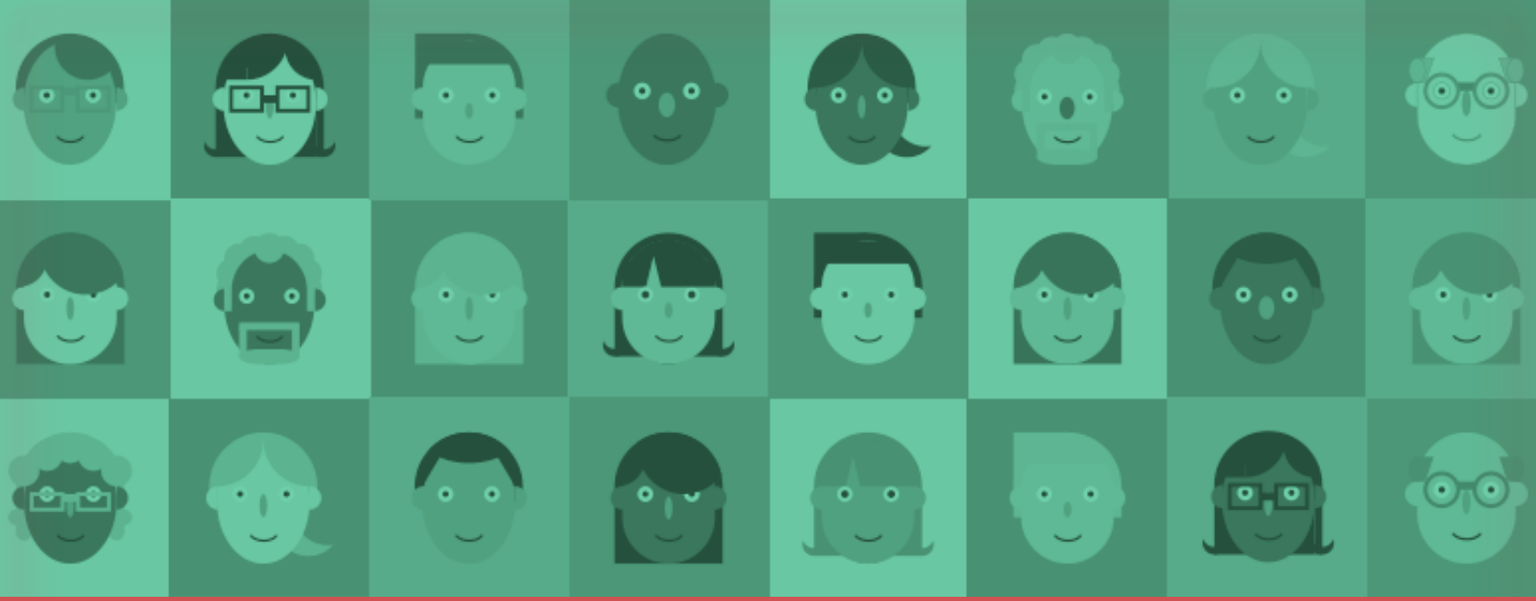
internos sobre los que se realizan múltiples operaciones aritmético-lógicas simples y son compartidos por todos los programas. En la arquitectura Intel, los registros generales también se llaman AX, BX, CX y DX, y el resultado de cada llamada al sistema suele devolverse en AX, tal y como se solicita a los jugadores de COMPUS. Los nombres y funcionalidades de las cartas de operación de COMPUS se corresponden fielmente con las operaciones lógicas y matemáticas con las que se programa una CPU a bajo nivel (lo que se conoce como código ensamblador).

A nivel lógico, los programas no comparten la memoria RAM en la que escriben sus datos, por lo que esa información está en un tablero aparte en COMPUS, no accesible al resto de jugadores. En un sistema multitarea que permite que varios programas corran sobre una misma CPU, el tiempo de procesamiento se asigna de forma similar a como ocurre en una partida de COMPUS (si bien no se concede aleatoriamente, sino en función de la prioridad de cada programa).

Teniendo todo esto en cuenta, quien juegue a COMPUS podrá aprender a contar en binario, sumar y restar 1 a un valor binario, realizar las principales operaciones lógicas en binario (NOT, OR, AND, XOR), conocer trucos para hacer multiplicaciones y divisiones por 2 de forma rápida y entender cómo comparten los recursos hardware los diferentes programas que se ejecutan al mismo tiempo en un sistema operativo.

Créditos

Última revisión: 22/01/2017



COMPUS

www.compus.es