

[STORMRG]



How Do It *Know?*

The Basic Principles of Computers for Everyone

J. Clark Scott

PERO, ¿CÓMO LO SABE?

Los principios básicos de las
computadoras para todos

Por

J. Clark Scott

Derechos de autor © 2009 por John Clark Scott

Diseño informático incorporado aquí
Copyright © 2009 por John Clark Scott

Todos los derechos reservados

Publicado por John C. Scott, Oldsmar, FL 34677

ISBN 978-0-615-30376-5

buthowdoitknow.com

Arte de portada, fotografía y diseño de Alexander C.
Scott III
artbyalexscott.com

Impreso en los Estados Unidos de América

Primera edición: julio de 2009

10 9 8 7 6 5 4 3 2 1

Tabla de contenidos

[Tabla de contenido](#)
[Introducción](#)
[Solo los hechos](#)
[señora Speed](#)
[Idioma](#)
[Solo un poco ¿Qué](#)
[demonios...?](#)
[Diagramas de](#)
[variaciones](#)
[simples](#)
[Recuerda cuándo](#)
[¿Qué podemos hacer con un](#)
[poco? Una rosa con](#)
[cualquier otro nombre ,](#)
[ocho es suficiente](#)
[Códigos](#)
[Volver al Byte El](#)
[Bus Mágico](#)
[Más combinaciones de](#)
[puertas Primera mitad de](#)
[los números de la](#)
[computadora](#)
[Direcciones](#)
[La otra mitad de la computadora](#)
[más puertas](#)
[Jugando con los bytes](#)
[Las palancas de cambios](#)
[izquierda y derecha The](#)
[NOTter](#)
[El Y El Más](#)
[El Más](#)
[El Exclusivo ORer El](#)
[Adder](#)
[El comparador y la lógica](#)
[cero](#)
[La unidad aritmética y lógica](#)
[más del procesador](#)
[El reloj](#)
[Hacer algo útil paso a](#)
[paso](#)

Todo está bajo control
Hacer algo útil, revisado ¿Qué
sigue?

Las primeras instrucciones
de la gran invención
La instrucción aritmética o lógica
Las instrucciones de carga y
almacenamiento La instrucción de
datos
El segundo gran invento:
otra forma de saltar
El Tercer Gran Invento La
Instrucción de Banderas
Claras Ta Daa!
Algunas palabras más sobre
aritmética El mundo exterior
El teclado
La pantalla de
visualización Otro
código
La última palabra sobre
los códigos El disco
Disculpe, señora, eso
es todo lo que la
gente Programas de
hardware y software
Los idiomas del
sistema operativo
Los errores del
sistema de archivos
¿Enfermedades
informáticas?
Firmware
Botas
Digital vs.
Analógico Mentí:
una especie de
filosofía de
divulgación
completa

Introducción

El título de este libro es el remate de un viejo chiste que dice así:

Joe es un tipo muy agradable, pero siempre ha sido un poco lento. Entra en una tienda donde un vendedor está parado en una tribuna frente a un grupo de personas. El vendedor está lanzando el nuevo invento milagroso, la botella termo. Él está diciendo: "Mantiene caliente la comida caliente y fría la comida fría..." Joe piensa en esto un minuto, asombrado por este nuevo invento que es capaz de tomar una decisión sobre cuál de las dos cosas diferentes se supone que debe hacer dependiendo del tipo de comida que le pongas. No puede contener su curiosidad, está saltando arriba y abajo, agitando el brazo en el aire, diciendo "pero, pero, pero, pero..." Finalmente, suelta su pregunta candente: "¿Pero cómo lo sabe?"

Puede que te hayas reído o no de la broma, pero el punto es que Joe miró lo que podía hacer esta botella termo y decidió que debía ser capaz de detectar algo sobre su contenido y luego realizar una operación de calentamiento o enfriamiento en consecuencia. Pensó que debía contener un calentador y un refrigerador. No tenía idea del principio mucho más simple en el que realmente opera, que es que el calor siempre intenta moverse de un área más caliente a un área más fría, y todo lo que hace el termo es ralentizar este movimiento. Con el contenido frío, el calor exterior se ralentiza en su entrada, y con el contenido caliente, el calor se ralentiza en su salida. La botella no tiene que "saber" para cumplir su misión y no calienta ni enfría nada. Y eventualmente, el contenido, caliente o frío, termina a temperatura ambiente.

Pero el concepto de Joe de cómo funcionaba la botella era mucho más complicado que la verdad.

Entonces, la razón del título del libro es que cuando se trata de computadoras, la gente las mira, ve lo que pueden hacer e imagina todo tipo de cosas que deben estar en estas máquinas. O imaginan todo tipo de principios en los que deben basarse y, por lo tanto, de lo que pueden ser capaces. Las personas pueden asignar cualidades humanas a la máquina. Y no pocos se encuentran en situaciones en las que sienten que se están avergonzando a sí mismos, como

nuestro amigo de la broma, Joe.

Pero las computadoras son en realidad bastante fáciles de entender. Por supuesto, las computadoras tienen un mayor número de piezas que una botella termo, pero cada parte es extremadamente simple y todas funcionan según un principio muy simple y muy fácil de entender.

Con el termo, el principio es el del movimiento del calor. Esto es algo que podemos observar en la vida. Vemos cubitos de hielo derritiéndose cuando se sacan del congelador, y vemos que la comida caliente se enfría en la mesa cuando la familia llega tarde a la cena.

En la computadora, el principio sobre el que opera tiene que ver con la electricidad, pero eso no significa que sea difícil de entender. Si ha observado el hecho de que cuando enciende un interruptor de luz, se enciende una bombilla, y cuando apaga el interruptor, la luz se apaga, entonces ha observado el principio en el que funcionan las computadoras. Eso es todo lo que necesita saber sobre la electricidad para comprender las computadoras.

Solo los hechos, señora

Este libro no pretende ser principalmente un libro de texto. No hay problemas que hacer al final de cada capítulo. Su intención es simplemente desmitificar el tema de las computadoras para cualquiera que alguna vez se haya preguntado qué está pasando dentro de esa caja. Por supuesto, también es una introducción perfecta a las computadoras para una persona joven que finalmente obtendrá un doctorado en Ciencias de la Computación. Pero debe ser fácilmente comprensible para las amas de casa, las personas mayores y los niños que pueden leer bien. Debe ser comprensible para los plomeros y barrenderos. No requiere educación técnica previa. Solo requiere que pueda leer el idioma, puede encender y apagar una bombilla y puede hacer una suma muy simple del orden de $8 + 5 = 13$.

Este libro presenta los elementos esenciales completos que componen una computadora. Presenta cada pieza y parte, en el orden adecuado para que cada una tenga sentido y pueda ser entendida. Cada parte se explica completamente, y cada nueva palabra se define a fondo cuando se usa por primera vez. Cualquier intento de simplificar aún más el tema dejaría vacíos en el panorama general en los que alguien aún tendría que adivinar cómo funcionan las partes juntas, y simplemente nunca tendrías ese momento de "¡Ajá, lo entiendo!" que creo que pronto tendrás.

Este libro no es una versión "simplificada" de algún libro de texto universitario. Es una explicación completa de los principios básicos de las computadoras. Es un libro técnico, pero también lo es un libro de cocina y también lo es un manual de educación vial. Este libro comienza por el principio y define todos los elementos necesarios para comprender la máquina. No importa lo que alguien ya sepa sobre computadoras, esto completará las piezas faltantes y las unirá todas en algo que tenga sentido.

Incluso nuestro amigo, Joe, pudo entender este libro con un estudio diligente. Hay miles de palabras e ideas asociadas con el campo de las computadoras que hacen que todo el tema parezca un desastre. Pero los conceptos básicos que subyacen son simples.

En este libro, no habrá volúmenes de trivialidades sobre

la construcción o historia de las computadoras, solo lo esencial, ni más ni menos. Cada parte de la computadora tiene una función simple, y cuando están conectadas entre sí, terminas con una máquina útil llamada computadora.

No hay nada que memorizar en este libro. Cada capítulo está diseñado para darte una nueva idea que no tenías antes, o si es algo de lo que habías oído hablar anteriormente, siempre parecía confuso. Cada idea es muy simple, y una cosa lleva a la siguiente. Cada capítulo presenta una idea. Cada idea es simple y fácil de entender. Los capítulos posteriores presentan ideas que se basan en las ideas de los capítulos anteriores.

Si alguien escribiera un libro sobre cómo construir una casa, podría haber varios niveles de detalle. El libro más simple diría: "coloque los cimientos, levante las paredes, cubra con un techo, coloque plomería y electricidad, y listo". Eso no sería suficiente detalle para alguien que aún no tuviera algo de experiencia en el uso de un martillo y una sierra y la instalación de un grifo y el cableado de un interruptor de luz.

En el otro extremo del espectro estaría un libro que tuviera capítulos separados para cada tipo posible de cimientos, los diferentes tipos de tierra en los que podría tener que cavar, fórmulas para una docena de tipos diferentes de concreto, gráficos de condiciones climáticas que son óptimas para colocar cimientos, etc. Eso sería demasiada información. Habría tantos detalles, que lo que era realmente importante se perdería.

Este libro intenta dar los detalles suficientes para ver qué tiene en común cada computadora y cómo funcionan, no cómo construir la computadora más grande o mejor jamás fabricada. No se trata de ninguna marca específica de computadora. No se trata de cómo usar una computadora. Si fuera un libro sobre la construcción de una casa, probablemente describiría un plan simple para un cobertizo de jardín resistente con un fregadero y una bombilla desnuda, mostrando el tamaño y la forma de cada pieza de madera, dónde colocar todos los clavos, cómo colgar la puerta y cómo armar las tuberías de agua para que no goteen. No mostraría cómo construir algo tan complicado como una elegante escalera curva de roble.

Vamos a mostrar la parte simple que las computadoras

están hechos, y luego conectar un montón de ellos hasta que hayamos construido una computadora completa. Va a ser mucho más simple de lo que jamás imaginaste.

Velocidad

Las computadoras parecen misteriosas y mágicas. ¿Cómo pueden hacer lo que hacen? Juegan, hacen dibujos, "conocen" su calificación crediticia. Estas máquinas son capaces de hacer todo tipo de cosas extrañas y maravillosas. Sin embargo, son simples. Solo pueden hacer unas pocas cosas muy simples. Y solo pueden hacer una de estas cosas simples a la vez. Parecen estar haciendo cosas complejas, solo porque hacen una gran cantidad de cosas simples una tras otra en un pequeño período de tiempo. El resultado, como en un videojuego, es muy complejo en apariencia, pero en realidad, es muy simple, solo que muy, muy rápido.

Las computadoras están diseñadas para hacer una pequeña cantidad de cosas simples específicas y para hacer estas cosas rápidamente, una tras otra. Qué cosas simples se hacen, y en qué orden, determina qué tipo de tarea realiza la computadora en un momento dado, pero cualquier cosa que haga la computadora no consiste en nada fuera de sus capacidades limitadas.

Una vez que veas de qué se compone una computadora, te darás cuenta de cómo es que pueden hacer lo que hacen, exactamente de qué tipo de cosas son capaces y también, de qué no son capaces.

Entonces, el secreto de las computadoras no es que sean complejas, sino su velocidad. Veamos exactamente qué tan rápida es su velocidad.

Dado que las computadoras funcionan con electricidad, su velocidad está relacionada con la velocidad de la electricidad. Es posible que recuerde haber escuchado que la velocidad de la luz es de 186,000 millas por segundo. Eso es bastante rápido. La luz puede dar la vuelta a toda la tierra siete veces en un segundo, o de la Tierra a la Luna en aproximadamente un segundo y medio. Según los físicos, la electricidad tiene muchas propiedades en común con la luz, y su velocidad, cuando viaja en un cable, se reduce a aproximadamente la mitad de la velocidad de la luz. Pero aún así, dar la vuelta a la Tierra tres veces y media en un segundo es extremadamente rápido.

Como punto de comparación, imagina que es un día

caluroso y tienes un ventilador eléctrico sobre la mesa
soplando

aire fresco sobre ti. El ventilador gira tan rápido que las aspas son borrosas, pero solo gira unas 40 veces por segundo. Un punto en el borde de una de esas aspas solo viajará unos 150 pies en ese segundo, ese punto tardará 35 segundos en viajar solo una milla.

Dado que las aspas del ventilador ya son borrosas, puede ser difícil imaginar que vayan diez veces más rápido. Si lo hiciera, ese ventilador estaría dando una gran brisa. Y si pudieras hacer que fuera cien veces más rápido, es casi seguro que se autodestruiría, con aspas del ventilador que se romperían y se atascarían en el techo. Pero la electricidad que viaja en el mismo círculo daría unas cien millones de veces en un segundo, es decir, dos millones y medio de veces más rápido que las aspas del ventilador. Eso es rápido.

Un millón es un número muy grande. Si tomaras una hoja de papel grande de 40 pulgadas cuadradas y tomaras una regla y la colocaras en el borde superior, y dibujaras 25 puntos por pulgada a lo largo del borde superior del papel, tendrías que dibujar mil puntos para atravesar esa hoja de papel. Si luego movieras la regla hacia abajo en la página 1/25 de pulgada, y dibujaras otros mil puntos, y siguieras haciendo eso, tendrías que mover la regla hacia abajo en la página mil veces, cada vez dibujando mil puntos. Si pudieras completar una tarea tan aburrida, terminarías con una hoja de papel con un millón de puntos. Eso es un montón de puntos o mucho de cualquier cosa. Y solo para terminar el pensamiento, si pudieras encontrar mil personas que dibujaran cada una de estas hojas de millones de puntos, y apilaran esas mil hojas en una pila, entonces tendrías mil millones de puntos.

Ahora digamos que la electricidad que se mueve dentro de la computadora puede realizar una tarea simple viajando un pie. Eso significa que la computadora podría hacer 500 millones de cosas simples en un segundo. De nuevo a modo de comparación, el ventilador de la mesa girará durante 7 horas para dar la vuelta sólo un millón de veces y tardará seis meses completos en girar alrededor de 500 millones de veces.

Cuando se habla de la velocidad a la que la electricidad puede moverse entre las partes dentro de la computadora, algunas de las partes que se pueden ver están separadas por un pie, otras están más cerca, una pulgada, una

pulgada, un

décima de pulgada. Y dentro de estas partes hay una multitud de partes más que están muy cerca unas de otras, algunas a solo milésimas de pulgada de distancia. Y cuanto más corta sea la distancia que tiene que recorrer la electricidad, antes llegará.

No tiene sentido decir cuántas cosas hacen las computadoras de hoy en un solo segundo, porque eso fecharía este libro. Los fabricantes de computadoras continúan produciendo nuevas computadoras que van dos veces más rápido que las computadoras más rápidas de hace solo dos o tres años. Existe un límite teórico a la velocidad a la que pueden ir, pero los ingenieros siguen encontrando formas prácticas de eludir las teorías y hacer máquinas que vayan cada vez más rápido.

Durante todo este tiempo que las computadoras se han vuelto más rápidas, más pequeñas y más baratas, las cosas que hacen las computadoras realmente no han cambiado desde que se inventaron por primera vez en la década de 1940. Todavía hacen las mismas cosas simples, solo que más rápido, más barato, más confiable y en un paquete más pequeño.

Solo hay unas pocas secciones en una computadora, y todas están hechas del mismo tipo de piezas. Cada sección tiene una misión específica, y la combinación de estas partes en una máquina fue un invento verdaderamente maravilloso. Pero no es difícil de entender.

Idioma

En este libro, vamos a necesitar definir algunas palabras que se utilizan para describir las partes dentro de una computadora.

En algunas profesiones, especialmente la médica y legal, hay una tendencia a inventar muchas palabras nuevas, y a tomarlas de los antiguos idiomas griego y latino, y a hacerlas largas y difíciles de pronunciar.

En el mundo de las computadoras, parece que los inventores pioneros eran un tipo de personas menos formales. La mayoría de las palabras que usaron son palabras simples del lenguaje cotidiano, palabras que ya existían, pero que se usan de una manera nueva.

Algunas de las nuevas palabras son palabras que ya conocemos, utilizadas como una parte diferente del discurso, como un sustantivo existente que ahora se usa como verbo. Algunas de las palabras son acrónimos, las primeras letras de las palabras de una frase.

Cada palabra se describirá detalladamente cuando se use por primera vez. Y aunque hay miles de palabras y acrónimos en uso si se considera toda la industria informática, solo se necesitan alrededor de una docena o dos palabras para entender la computadora en sí. Probablemente hayas escuchado algunas de estas palabras antes, y hayas descubierto lo que significaban por cómo se usaron, pero ahora obtendrás las definiciones adecuadas y completas. En muchos casos, es posible que descubras que son más simples de lo que pensabas.

Solo un poco

¿Qué hay en una computadora? Te muestra imágenes fijas, imágenes en movimiento, música, tu chequera, cartas que has escrito, juega videojuegos, se comunica en todo el mundo y mucho más. Pero, ¿hay imágenes dentro de la computadora? Si sacaras un microscopio y supieras dónde buscar, ¿podrías encontrar pequeñas imágenes en algún lugar dentro de la computadora? ¿Verías "A" y "B" y "8" y "12" moviéndose allí en alguna parte?

La respuesta es no, no hay imágenes, números o letras en una computadora. Solo hay un tipo de cosa en una computadora. Hay una gran cantidad de este tipo de cosas, pero solo hay un tipo de cosas allí. Se llama bit.

Cuando lanzas una moneda en el aire y la dejas caer al suelo, terminará en el suelo en uno de los dos estados posibles, ya sea con la cara a la vista o la cola.

La luz de su sala de estar (suponiendo que tenga un interruptor y no un atenuador) puede estar encendida o apagada.

La cerradura de la puerta de entrada puede estar bloqueada o desbloqueada.

¿Qué tienen en común todas estas cosas? Todos son lugares que contienen una cosa que puede estar en uno de dos estados posibles. Esta es la definición de un bit.

Un bit es algún tipo de objeto físico que tiene un tamaño y una ubicación en el espacio, y tiene alguna cualidad sobre sí mismo, que en un momento dado puede estar en uno de dos estados posibles, y puede cambiar de un lado a otro entre esos dos estados.

Un trozo de arcilla no es un poco. Se puede moldear en una bola, un cubo, un panqueque, un anillo, un tronco, una cara o cualquier otra cosa que se te ocurra. Tiene un tamaño y una ubicación en el espacio, pero hay demasiados estados en los que puede estar para que se le llame un poco. Si tomaras ese trozo de arcilla, lo aplanaras, rayaras "sí" en un lado y "no" en el otro lado, y luego lo pusieras en un horno y lo cocieras hasta que estuviera duro, entonces podrías

poder llamarlo un poco. Podría sentarse en una mesa con el "si" o el "no" a la vista. Entonces solo tendría dos estados.

Probablemente hayas oído hablar de los bits antes en relación con las computadoras, y ahora sabes cuáles son. En una computadora, los bits no son como la moneda o la cerradura, son más como la luz. Es decir, los bits en una computadora son lugares que tienen electricidad o no. En una computadora, los bits son muy, muy pequeños y hay una gran cantidad de bits, pero eso es todo lo que hay allí.

Al igual que la luz de la sala de estar, la broca está encendida o apagada. En la sala de estar, hay electricidad en la pared que entra por el interruptor. Cuando enciende el interruptor, la electricidad va desde el interruptor, a través de los cables en la pared y el techo, hacia el enchufe de la luz y luego hacia la bombilla. Entonces, esta parte en la sala de estar mide varios pies de largo, incluye el interruptor, los cables, el enchufe y la bombilla. En una computadora, los bits son en su mayoría pequeños, en realidad microscópicos. Además, la parte de la computadora no tiene un interruptor mecánico en un extremo o una bombilla en el otro. Si quitaba la bombilla del enchufe de la sala de estar, el interruptor aún enviaría electricidad al enchufe cuando estuviera encendido, y todavía estaría un poco, simplemente no podría ver si estaba encendido o apagado al mirar una bombilla. Su computadora tiene algo parecido a interruptores, como las teclas del teclado, y algo parecido a bombillas, como los pequeños puntos en la pantalla, pero la mayoría de los bits están adentro y no se ven.

Esto es básicamente todo lo que hay en una computadora: bits. Hay montones y montones de ellos, y están dispuestos y conectados de varias maneras, que examinaremos en detalle a medida que avance el libro, pero esto es lo que hay dentro de todas las computadoras: bits. Un bit siempre está en uno de sus dos estados posibles, ya sea apagado o encendido, y cambian entre encendido y apagado cuando se les dice que lo hagan. Los bits de computadora no son como la moneda que tiene que voltearse físicamente para cambiar de un estado a otro. Los bits no cambian de forma o ubicación, no se ven diferentes, no se mueven ni giran ni se hacen

más grandes o más pequeños. Un bit de computadora es solo un lugar, si no hay electricidad en ese lugar, entonces el bit es

apagado. Cuando hay electricidad, entonces el bit está encendido.

Si desea cambiar una moneda de mostrar cara a mostrar cruz, debe moverla físicamente para darle la vuelta, lo que lleva cierta cantidad de tiempo. Debido a que lo único que tiene que moverse en un bit de computadora es la electricidad, cambiar su estado de apagado a encendido, o de encendido a apagado puede suceder mucho más rápido que cualquier cosa que tenga que moverse físicamente.

Como otro ejemplo, ¿recuerdas el salvaje oeste americano de las películas? Había pequeños pueblos separados por grandes distancias. Las ciudades más grandes tendrían una oficina de telégrafos. En esta oficina había un tipo que llevaba un sombrero divertido que tenía un interruptor de resorte llamado llave, y enviaba mensajes presionando esta tecla para encenderla y apagarla en ciertos patrones que representaban las letras del alfabeto.

Esa llave estaba conectada a una batería (sí, tenían baterías en ese entonces) y un cable que se colgaba a lo largo de postes hasta que llegaba a otra ciudad. La llave simplemente conectaba la batería al cable cuando se presionaba y desconectaba la batería cuando no se presionaba la tecla. En la otra ciudad había otra oficina de telégrafos, el cable entraba en esa oficina, el extremo estaba enrollado alrededor de una barra de hierro (que se convierte en un imán cuando hay electricidad en el cable), la varilla magnetizada atraía una pequeña barra de hierro sostenida cerca con un resorte y hacía un chasquido cada vez que se encendía la electricidad. El tipo de la oficina escuchó el patrón del clic y anotó las letras del mensaje. Podrían haber usado una bombilla en lugar del clicker, excepto que aún no se habían inventado las bombillas.

El punto de traer a colación este tema es que toda esta máquina de telégrafo, desde la tecla que se presiona en una ciudad, a través del largo cable que viaja a otra ciudad a muchas millas de distancia, hasta el clicker, todo este aparato comprende un solo bit. Es un lugar que puede tener o no electricidad, y se enciende y apaga según se le dice. Y este método de comunicación revolucionó el mundo de muchas maneras. Pero este invento tan importante de la década de 1840 consistió en nada más que un bit.

Así que espero que esto comience a simplificar el tema de

computadoras para ti. Solo hay una cosa dentro de las computadoras, bits. Muchos de ellos, sin duda, pero cuando entiendes las partes, entiendes lo que hay allí.

¿Qué...?

Imagina que es un día soleado y entras en una habitación con muchas ventanas abiertas. Notas que la luz del techo está encendida. Decides que esto es un desperdicio y vas a apagar la luz. Miras la pared al lado de la puerta y ves una placa de interruptores con dos interruptores. Así que asumes que el que está más cerca de la puerta es para la luz del techo. Pero luego notas que el interruptor ya está apagado. Y el otro interruptor también está apagado. Entonces piensas "bueno, tal vez alguien instaló el interruptor al revés", así que decides encender el interruptor de todos modos. Lo enciendes y lo apagas pero no pasa nada, la luz del techo permanece encendida. Entonces decides que debe ser el otro interruptor, y lo enciendes, apagas, enciendes, apagas. De nuevo no pasa nada, esa luz del techo sigue brillando hacia ti. Miras a tu alrededor, no hay otra puerta, no hay otros interruptores, no hay forma aparente de apagar esta maldita luz. Solo tiene que ser uno de estos dos interruptores, ¿quién construyó esta casa loca de todos modos? Así que agarras un interruptor con cada mano y comienzas a girarlos salvajemente. Entonces, de repente, notas que la luz del techo parpadea brevemente. Así que reduce la velocidad de encendido del interruptor y se detiene cuando la luz del techo está apagada. Ambos interruptores dicen "encendido" y la luz ahora está apagada. Apaga un interruptor, luego lo enciende, y la luz se enciende y luego se apaga. Esto es al revés. ¿Un apagado equivale a luz encendida? Entonces apagas el otro interruptor, luego lo enciendes, lo mismo, la luz se enciende y luego se apaga. ¿Qué diablos? De todos modos, finalmente descubres cómo funciona. Si ambos interruptores están encendidos, la luz se apaga. Si uno u otro o ambos interruptores están apagados, entonces la luz del techo está encendida. Un poco tonto, pero logras lo que pretendías, enciendes ambos interruptores, la luz se apaga y sales de esta loca habitación.

Ahora, ¿cuál es el propósito de esta pequeña historia sobre los extraños interruptores de luz? La respuesta es, que en este capítulo vamos a presentar la parte más básica de la que están hechas las computadoras. Esta parte funciona exactamente como el sistema de iluminación en esa extraña habitación.

Esta parte de la computadora es un dispositivo simple que tiene tres conexiones donde puede haber o no algunas

electricidad. Dos de esas conexiones son lugares donde se puede poner electricidad en el dispositivo, y la tercera conexión es un lugar donde la electricidad puede salir del dispositivo.

De las tres conexiones, dos de ellas se llaman "entradas", porque la electricidad se les puede enviar desde otro lugar. La tercera conexión se llama "salida" porque la electricidad puede salir de ella y luego enviarse a otro lugar.

Esta parte de la computadora es un dispositivo que hace algo con bits. Si tiene dos bits y conecta esos dos bits a las entradas, este dispositivo "mira" esos dos bits y "decide" si encender o apagar el bit de salida.

La forma en que "decide" es muy simple y siempre es la misma. Si ambas entradas están activadas, la salida estará apagada. Si una o ambas entradas están apagadas, la salida estará activada. Así es como funcionaba la habitación con los extraños interruptores de luz.

Recuerde que no hay nada más que bits dentro de la computadora. Este sencillo dispositivo es de donde vienen los bits y a dónde van. La "decisión" que toma este dispositivo es cómo se encienden y apagan los bits en una computadora.

Dos bits entran en el dispositivo y sale un bit. Dos bits provienen de otro lugar, son examinados por el dispositivo y se genera un nuevo tercer bit para que pueda ir a otro lugar.

Si ha sido más observador, es posible que se haya hecho esta pregunta: "cuando ambas entradas están apagadas, la salida está encendida, así que ... ¿Cómo se obtiene electricidad en la salida si ambas entradas están apagadas?" Bueno, esa es una excelente pregunta, y la excelente respuesta es que cada uno de estos dispositivos también está conectado a la alimentación. Como todos los electrodomésticos o lámparas de mesa de su casa, donde cada uno tiene un enchufe con dos clavijas, este dispositivo tiene un par de cables, uno de los cuales está conectado a un lugar donde la electricidad está siempre encendida y el otro está conectado a un lugar donde la electricidad siempre está apagada. De aquí proviene la electricidad para la salida. Cuando alguien construye una computadora, tiene que hacer todas esas conexiones de energía a cada una de esas partes para que funcione, pero

cuando estamos dibujando diagramas

de las piezas, cómo están conectadas y qué harán, no nos molestaremos en dibujar los cables de alimentación, simplemente desordenarían el dibujo. Se entiende que cada parte tiene su conexión de alimentación, y no nos preocupamos por eso. Solo entiende que está ahí, y no lo mencionaremos más por el resto del libro. Ni siquiera lo habría mencionado aquí, excepto que pensé que probablemente te harías esa pregunta tarde o temprano.

Ahora sé que dije que no tienes que entender mucho sobre electricidad para entender las computadoras. Aquí es tan complicado como parece. En realidad, hay media docena de partes electrónicas dentro de este dispositivo que lo hacen funcionar, pero no vamos a examinar esas partes en este libro. Alguien que tenga experiencia en electrónica podría mirar lo que hay allí, y en unos 30 segundos diría "Oh, sí, si ambas entradas están encendidas, la salida estará apagada, y para cualquier otra combinación la salida estará encendida, tal como dice el libro". Y luego esa persona podría seguir adelante y leer este libro sin tener que volver a pensar en lo que hay allí. Alguien que no sabe de electrónica se pierde esos pocos segundos de comprensión, pero este libro es igual para todos.

En el cableado normal de la casa, un interruptor enciende y apaga una luz. En la computadora, se necesitan dos interruptores, y es un poco al revés en el sentido de que ambos tienen que estar encendidos para apagar la luz. Pero si acepta el hecho de que se podría hacer algo que funcione de esta manera, puede comprender cómo funciona todo en la computadora.

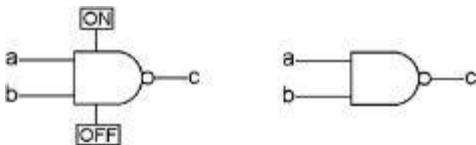
Este tipo de pieza de computadora es, de hecho, el ÚNICO tipo de pieza requerida para construir una computadora. Por supuesto, se necesitan muchos de ellos para construir una computadora completa, pero con suficientes, puede hacer cualquier tipo de computadora. Así que ahí lo tienes de nuevo, ¿ves lo simple que es una computadora? Está lleno de este pequeño tipo de cosas, muchas de ellas, sin duda, pero esto es todo lo que hay.

Ahora tenemos que darle un nombre a este dispositivo, esta cosa dentro de la computadora de la que están hechos los bits, se llama "puerta". No puedo encontrar una buena razón por la que se llama puerta, una puerta en una cerca deja pasar a la gente cuando está abierta y detiene a la

gente cuando está cerrada. Una computadora

gate genera un tercer bit a partir de otros dos bits, no se abre y se cierra ni se detiene ni deja pasar nada. El significado de este término informático "puerta" no parece encajar en el significado común de la palabra, pero lo siento, no inventé el nombre, así es como se llama. Te acostumbrarás. Al menos no es una palabra larga del griego antiguo.

En los próximos capítulos, vamos a mostrar cómo podemos hacer algo útil conectando varias puertas. Usaremos dibujos como los siguientes. La forma de 'D' con el pequeño círculo en su punta representa el dispositivo que hemos descrito, y las líneas representan los cables que entran y salen de él y que se conectan a otras partes de la computadora. La imagen de la izquierda muestra una puerta completa con sus cables de alimentación, pero como prometimos, no nos preocuparemos por ellos durante el resto de este libro. El dibujo de la derecha muestra todo lo que necesitamos:



Esta es una representación de una puerta. Los dos cables de la izquierda (a y b) son las entradas y el cable de la derecha

(c) es la salida. Los tres cables son bits, lo que significa que están encendidos o apagados. Cada bit de entrada proviene de otro lugar de la computadora y está encendido o apagado dependiendo de lo que esté sucediendo de donde vino, y luego esta puerta establece su salida en función de los estados de sus dos entradas.

A veces es útil hacer un pequeño gráfico que muestre cómo las diversas combinaciones de entrada crean la salida, así:

u	b	c

Apagado	Apagado	En
Apagado	En	En
En	Apagado	En
En	En	Apagado

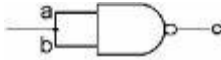
Cada línea muestra una posible combinación de las entradas y cuál será la salida en esas circunstancias.

Compare este pequeño gráfico con la experiencia con la habitación extraña con los dos interruptores de luz. Si un interruptor se llama 'a', el otro interruptor se llama 'b' y la luz del techo se llama 'c', entonces esta pequeña tabla describe completa y exactamente cómo funciona el equipo en esa habitación. La única forma de apagar esa luz es encender tanto el interruptor 'a' como el interruptor 'b'.

Variaciones simples

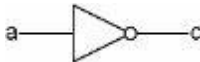
Como se mencionó, esta puerta es lo único que necesita para construir una computadora, pero necesita muchas de ellas, y deben conectarse de manera inteligente para poder hacer que hagan algo útil. Lo que vamos a hacer aquí es mostrar dos cosas sencillas que se hacen muchas veces dentro de cualquier ordenador.

Este primero es muy simple. Tome la puerta de arriba y tome los dos cables de entrada, 'a' y 'b', y átelos. Por lo tanto, 'a' y 'b' siempre serán lo mismo. Todavía se pueden cambiar de forma intermitente, pero 'a' y 'b' nunca pueden ser diferentes. 'A' y 'b' pueden estar encendidos o apagados. Por lo tanto, el gráfico de esta combinación solo tiene dos líneas, dos posibilidades:



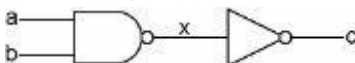
un	b	c
Apagado	Apagado	En
En	En	Apagado

En realidad, dado que las columnas 'a' y 'b' son lo mismo, en realidad solo hay una entrada y se puede dibujar simplemente así con un triángulo en lugar de la forma 'D'. Su gráfico también es muy sencillo:



un	c
Apagado	En
En	Apagado

Para nuestra segunda variación, combinemos uno de nuestros tipos originales de puerta con la nueva puerta que acabamos de inventar, así:



Y combinaremos los gráficos de cómo funcionan. La 'a', la 'b' y la 'x' son como la primera puerta, la 'x' y la 'c' son como la segunda puerta.

un	b	x	c
Apagado	Apagado	En	Apagado
Apagado	En	En	Apagado
En	Apagado	En	Apagado
En	En	Apagado	En

Esta combinación se usa con tanta frecuencia dentro de las computadoras, que está construida como una sola unidad, y el bit 'x' no está disponible para conectarse. Entonces, para que sea más fácil de entender, se dibuja como una sola unidad de esta manera:



La única diferencia entre esta imagen y la imagen de nuestra puerta original es que falta el pequeño círculo después de la gran 'D'.

Dado que no se usa 'x', el gráfico también se puede simplificar, y se ve así:

un	b	c
Apagado	Apagado	Apagado
Apagado	En	Apagado
En	Apagado	Apagado
En	En	En

La única diferencia entre este gráfico y el gráfico de nuestra puerta original es que cada elemento de la columna 'c' es lo opuesto a lo que era en el gráfico original.

Imagina que esta combinación de puertas se instaló en esa habitación con los dos interruptores de luz y la luz del techo. La única forma en que la luz podría estar encendida es si ambos interruptores estuvieran encendidos. Entonces, si entras allí y ves la luz encendida, y luego miras los interruptores, verás que ambos están encendidos. No importa qué interruptor decidieras que era para la luz, y lo apagaste, la luz se apagaría. Es posible que no se dé cuenta de que si apaga ambos y luego desea volver a encender la luz, no podría hacerlo con solo presionar un interruptor. Tendrías que pasar por el mismo experimento, accionando ambos interruptores hasta que se encendiera la luz, y encontrarías que un interruptor y el otro interruptor tendrían que estar encendidos para que la luz se

encienda.

Esta puerta de combinación podría describirse de esta manera: para que la salida esté encendida, una entrada Y la otra entrada deben estar encendidas. Por lo tanto, este tipo de puerta tiene un nombre, y en la tradición de la terminología informal inventada por la gente de la computadora, debido a que nos recuerda lo que significa la palabra Y, simplemente se llama "puerta Y".

Ahora, para completar algunos detalles omitidos a propósito anteriormente, la puerta original que vimos funciona como la puerta AND, excepto que la salida es lo opuesto, o el negativo de la puerta AND. Por lo tanto, se llama puerta Y negativa, o simplemente "puerta NAND" para abreviar.

La puerta simple que tenía ambas entradas unidas también tiene su propio nombre. La salida es siempre lo opuesto a la entrada, es decir, si la entrada está encendida, la salida no está encendida (apagada). Si la entrada está apagada, la salida no está apagada (encendida). La salida siempre NO es lo que es la entrada, por lo tanto, se llama "puerta NOT".

Observe la diferencia entre los diagramas de la puerta AND y la puerta NAND. Son los mismos, excepto que hay un pequeño círculo al comienzo de la salida de la puerta NAND. Lo que parece una letra grande 'D' significa hacer la función 'Y', lo que significa tomar medidas solo si ambas entradas están activadas, y el círculo pequeño significa cambiar a lo contrario. Entonces, una puerta AND está activada si ambas entradas están activadas, una puerta NAND está desactivada si ambas entradas están activadas. La puerta NOT comienza con un triángulo, lo que significa simplemente tomar la entrada y convertirla en una salida. El círculo significa entonces cambiar a lo contrario.

La puerta AND se usa mucho en las computadoras, y probablemente sea la más fácil de entender, pero primero miramos la puerta NAND por dos razones. La primera razón, y menos importante, es que la puerta NAND es la puerta más fácil de construir. Cuando tenga que construir una gran cantidad de puertas, será más barato y confiable si puede usar el tipo de puerta que sea más fácil de construir.

La segunda y muy importante razón por la que miramos primero la puerta NAND es esta: que todo en una

computadora que la convierte en una computadora, se puede hacer a partir de una o más puertas NAND. Ya hemos visto que la puerta NOT y la puerta AND se pueden hacer a partir de puertas NAND, y veremos algunas combinaciones más interesantes a medida que avancemos. Pero cada uno de ellos se basa en esto

pequeña cosa tonta llamada puerta NAND.

El problema en este capítulo ha sido que la puerta NAND es el bloque de construcción básico de las computadoras, pero la puerta AND es la primera puerta que tiene un nombre que tiene sentido. Así que miramos primero la puerta NAND y la puerta NOT sin darles nombres. Luego construimos una puerta AND, le dimos su nombre, y volvimos y nombramos las dos primeras.

Como nota sobre el lenguaje aquí, la palabra 'y' es una conjunción en inglés normal. Conecta dos cosas, como en "Me gustan los guisantes y las zanahorias". En las computadoras, usamos la palabra de dos maneras nuevas. Primero, es un adjetivo, una palabra que modifica un sustantivo. Cuando decimos "esta es una puerta Y", la palabra "puerta" es un sustantivo, y la palabra "Y" nos dice qué tipo de puerta es. Así es como se ha utilizado "Y" en este capítulo. "AND" también se usará como verbo, como en "let us AND these two bits". Veremos AND usado de esta manera más adelante en el libro.

Entonces, volviendo al tema de la simplicidad de este libro, hemos dicho que solo hay una cosa en las computadoras, bits. Y ahora vemos que los bits se construyen usando puertas, y todas las puertas bajan a la puerta NAND. Entonces, todo lo que tiene que saber para comprender las computadoras es este dispositivo muy simple, la puerta NAND. ¡No es broma! ¿Puedes entender esto? Entonces puedes entender toda la computadora.

Diagramas

Si quieres ver cómo funciona una máquina mecánica, la mejor manera de hacerlo es mirar dentro de ella, ver cómo se mueven las piezas mientras funciona, desmontarla, etc. La segunda mejor manera es estudiarlo a partir de un libro que tenga muchas imágenes que muestren las partes y cómo interactúan.

Una computadora también es una máquina, pero lo único que se mueve dentro de ella es la electricidad invisible y silenciosa. Es muy aburrido ver el interior de una computadora, no parece que esté sucediendo nada en absoluto.

La construcción real de las partes individuales de una computadora es un tema muy interesante, pero no vamos a cubrirlo más allá de decir lo siguiente: la técnica comienza con una oblea de cristal delgado y, en una serie de pasos, se somete a varios productos químicos, procesos fotográficos, calor y metal vaporizado. El resultado es algo llamado 'chip', que tiene millones de piezas electrónicas construidas en su superficie. El proceso incluye conectar las piezas en puertas y conectar las puertas en secciones completas de computadora. Luego, el chip se encierra en una pieza de plástico que tiene alfileres que salen. Varios de estos están conectados a una placa, y ahí tienes una computadora. La computadora que vamos a "construir" en este libro podría caber fácilmente en un chip de menos de un cuarto de pulgada cuadrada.

Pero el punto es que, a diferencia de una máquina mecánica, la estructura real de un chip es muy desordenada y difícil de seguir, y no se puede ver la electricidad de todos modos. Los diagramas que vimos en el capítulo anterior son la mejor manera de mostrar cómo funciona una computadora, por lo que será mejor que seamos bastante buenos para leerlos.

A lo largo del resto de este libro, vamos a construir nuevas piezas conectando varias puertas entre sí. Describiremos lo que hace la nueva pieza y luego le daremos un nombre y su propio símbolo. Entonces podemos conectar varias de esas nuevas partes en otra cosa que también recibe un nombre y un símbolo. Antes de que te des cuenta, habremos montado un ordenador completo.

Cada vez que haya un nuevo diagrama, el texto explicará

cuál es su propósito y cómo lo logran las partes, pero el lector realmente debe mirar el diagrama hasta que se pueda ver que las puertas realmente hacen lo que el libro dice que harán. Si esto se hace fielmente con cada uno, muy pronto verá exactamente cómo funciona una computadora.

Solo hay dos cosas en nuestros dibujos, hay partes que tienen entradas y salidas, y hay líneas, o cables, que conectan salidas y entradas.

Cuando la electricidad sale de la salida de una puerta, la electricidad viaja a través de todo el cable lo más rápido posible. Si la salida de una puerta está encendida, entonces la electricidad está encendida en el cable que está conectado a ella, hasta donde llega. Si la salida de una puerta está apagada, todo el cable está apagado. Supongo que podría considerar que la parte que sale de la puerta también incluye todo el cable.

Las entradas de las puertas no consumen la electricidad en el cable, por lo que una salida puede conectarse a la entrada de una o varias puertas.

Cuando los cables están conectados entre sí, esto se muestra con un punto donde se encuentran en el diagrama, y todos los cables que están conectados entre sí obtienen electricidad como si fueran un solo cable. Cuando los cables se cruzan en un diagrama sin un punto, significa que no hay conexión entre ellos, no se tocan, los dos bits están separados.

Siempre que haya una opción, los diagramas mostrarán la ruta de la electricidad moviéndose de izquierda a derecha, o desde la parte superior de la página hacia la parte inferior. Sin embargo, habrá muchas excepciones a esto, especialmente más adelante en el libro. Pero siempre puede saber en qué dirección se mueve la electricidad en un cable comenzando en una salida y siguiéndola hasta una entrada.

La mayoría de los diagramas del libro son muy fáciles de seguir. En algunos casos, también habrá uno de esos gráficos que muestre cuál será la salida para cada combinación posible de entradas. Si tiene problemas para seguir un diagrama, puede escribir los encendidos y apagados directamente en la página, o colocar monedas en la página y voltearlas para que cara signifique encendido y cruz signifique apagado.

Desafortunadamente, el diagrama en el siguiente capítulo es

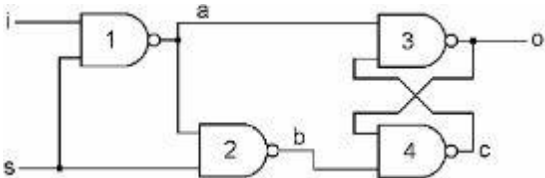
probablemente el más difícil de seguir en todo el libro, pero una vez que lo domines, serás un experto lector de diagramas.

Recuerda cuándo

Probablemente haya oído hablar de la memoria de computadora, y ahora vamos a ver exactamente qué es. Dado que lo único dentro de las computadoras son los bits, y lo único que les sucede a los bits es que se encienden o se apagan, entonces se deduce que lo único que una computadora puede "recordar" es si un bit estaba encendido o apagado. Ahora veremos cómo se logra eso.

El siguiente diagrama muestra un bit de memoria de computadora. Resulta ser uno de los trucos más ingeniosos que puedes hacer con unas pocas puertas. Examinaremos cómo funciona aquí en gran medida, y después de entenderlo, lo reemplazaremos con su propio símbolo y lo usaremos como un bloque de construcción para cosas más grandes y mejores.

Está hecho de solo cuatro puertas NAND, pero su cableado es algo especial. Aquí está:



Esta combinación en su conjunto tiene dos entradas y una salida. 'I' es donde ingresamos el bit que queremos recordar, y 'o' es la salida del bit recordado. 'S' es una entrada que le dice a estas puertas cuándo 'configurar' la memoria. También hay tres cables internos etiquetados como 'a', 'b' y 'c' que tendremos que mirar para ver cómo funcionan juntas estas partes. Trata de seguir esto cuidadosamente, una vez que veas que funciona, entenderás una de las cosas más importantes y más utilizadas en una computadora.

Para ver cómo funciona esto, comience con 's' activado e 'i' desactivado. Dado que 'i' y 's' van a la puerta 1, una entrada está desactivada, por lo que 'a' estará activada. Dado que 'a' y 's' van a la puerta 2, ambos

Las entradas están activadas y, por lo tanto, 'B' estará desactivada. Mirando la puerta 4, dado que 'b' está apagado, la salida de la puerta 4, 'c' estará encendida. Dado que 'c' y 'a' están encendidos, la salida de la puerta 3, 'o' estará apagada. 'O' vuelve a bajar a la puerta 4 proporcionando una segunda entrada de apagado, dejando 'c' todavía encendido. Lo importante a tener en cuenta aquí es que con 's' activada, 'o' termina igual que 'i'.

Ahora, con 's' todavía encendido, cambiemos 'i' a on. Dado que 'i' y 's' entran en la puerta 1, 'a' estará apagado. 'A' va a un lado de la puerta 2 y la puerta 3, por lo tanto, sus salidas 'o' y 'b' deben estar activadas. 'O' y 'b' ambos encendidos van a la puerta 4 y apagan 'c', que vuelve a subir a la puerta 3 proporcionándole una segunda entrada de apagado, dejando 'o' todavía encendido. Lo importante a tener en cuenta aquí es lo mismo que notamos en el párrafo anterior: que con 's' activada, 'o' termina igual que 'i'.

Hasta ahora, hemos visto que cuando 's' está activado, puede activar y desactivar 'i', y 'o' cambiará con él. 'O' se encenderá y apagará igual que 'i'. Con 's' activado, esta combinación no es más útil que un cable que conecte 'i' a 'o'.

Ahora veamos qué sucede cuando desactivamos la 's'. Mira la puerta 1. Cuando la 's' está apagada, la 'a' estará encendida sin importar lo que le hagas a la 'i'. Ahora puedes encender y apagar la 'i' y no pasará nada. Lo mismo ocurre con la puerta 2. 'A' puede estar activado, pero 's' está desactivado, por lo que 'b' solo puede estar activado. Tanto 'a' como 'b' están activados, y cambiar 'i' no hace nada. Ahora lo único que queda que importa, la gran pregunta es, ¿qué será 'o'?

Si 'i' y 'o' estaban encendidos antes de que 's' se apagara, la puerta 3 tenía ambas entradas apagadas y la puerta 4 tenía ambas entradas encendidas. Cuando 's' se dispara, 'a' se enciende, que es una entrada a la puerta 3. Pero la otra entrada está apagada, por lo que nada cambia, 'o' permanece encendida.

Si la 'i' y la 'o' estaban apagadas antes de que la 's' se apagara, la puerta

3 tenía ambas entradas encendidas y la puerta 4 tenía ambas entradas apagadas. Cuando 's' se apaga, 'b' se

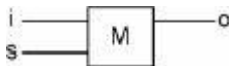
enciende, que es una entrada a la puerta 4. Pero la otra entrada está desactivada, por lo que nada cambia, 'c' permanece encendida y 'o' permanece apagada.

Entonces, la respuesta a la pregunta de qué le sucede a 'o' cuando 's' está apagado, es que permanece como estaba y ya no se ve afectada por 'i'.

Ahora, ¿qué tenemos aquí? Con la 's' activada, la 'o' hace lo que sea que haga la 'i'. Con la 's' apagada, la 'o' permanece como estaba y la 'i', en el último instante justo antes de que sonara la 's'. Ahora la 'i' puede cambiar, pero la 'o' permanece como estaba. Esta combinación de puertas se bloquea en la forma en que lo hizo la 'i' en un momento anterior. Así es como una combinación de cuatro puertas NAND puede "recordar". Esto es solo un bit de memoria, pero este es el componente básico de toda la memoria de la computadora. Todo lo que es la memoria de la computadora es una forma de preservar la forma en que se estableció un bit en algún momento.

Espero que hayas seguido los cables y los encendidos y apagados en este capítulo. Una vez que veas exactamente cómo funciona esto, sabrás que estas simples puertas NAND pueden crear un bit de memoria, y te aseguro que nunca más te lo preguntarás.

Ahora que sabemos cómo funciona esto, ya no necesitamos mirar ese complicado cableado interno de esta combinación. Hemos visto cómo funciona, y a partir de ahora, solo usaremos este diagrama para representarlo:



'I' es el bit de entrada que desea guardar. 'S' es la entrada que permite que 'i' entre en el bit de memoria cuando 's' está encendido, y lo bloquea en su lugar o lo 'establece' cuando 's' se apaga. 'O' es la salida de los datos actuales o guardados. 'M' significa Memoria. Bastante simple, ¿eh?

Volvamos a nuestra habitación con los divertidos interruptores de luz. Tenía una puerta NAND conectada a él. Quidemos la puerta NAND y reemplácela con este nuevo bit de memoria. Conectaremos el interruptor izquierdo al cable 'i', el interruptor derecho al cable 's' y la luz del techo al cable 'o'. Podríamos comenzar con todo igual, es decir, la luz está encendida, pero ambos interruptores están apagados. Eso significaría que en algún momento del pasado, tanto la 'i' como la 's' estaban activadas, y la 's' se apagaba primero, bloqueando el estado entonces de la 'i' en nuestro bit

de memoria, que luego sale en la 'o'. Entonces la 'i' podría haberse apagado sin afectar nada. Entonces, si entrábamos y decidíamos que queríamos apagar la luz, primero probábamos el interruptor 'i', lo encendíamos y apagamos, y

no pasaría nada. Luego probaríamos el interruptor 's'. Cuando lo encendemos, la luz se apaga. Ajá, decimos, el interruptor 's' controla la luz, ¡pero está instalado al revés! Entonces volvemos a apagar el interruptor 's', esperando que la luz vuelva a encenderse, pero la luz permanece apagada. Ahora los interruptores están en la misma posición que cuando entramos en la habitación, ambos están apagados, pero ahora la luz también está apagada, vaya que esto es confuso. Ahora no quiero especular sobre cuántas maldiciones se producirían antes de que alguien se diera cuenta de esto, pero al final encontrarían que cuando la 's' estaba encendida, la luz se encendía y apagaba con la 'i', y cuando la 's' estaba apagada, la luz permanecía como estaba justo antes de que la 's' se apagara.

¿Qué podemos hacer con un poco?

Ahora hemos descrito un poco, hemos mostrado cómo construir uno, cómo recordar a lo largo del tiempo en qué estado se encontraba un bit en un instante anterior en el tiempo, ¿y ahora qué? ¿Qué hacemos con él?

Dado que un bit en realidad no es más que la electricidad encendida o apagada, lo único real que podemos hacer con un bit es encender o apagar las luces, o las tostadoras o lo que sea.

Pero también podemos usar un poco para representar algo más en nuestras vidas. Podemos tomar un poco y conectarlo a una luz roja, y decir que cuando este bit está encendido, significa detenerse, y cuando este bit está apagado, puede irse. O si hay un poco en particular, quieres papas fritas con tu hamburguesa; Si está apagado, solo quieres la hamburguesa.

Esta es la acción de usar un código. ¿Qué es un código? Un código es algo que te dice lo que significa otra cosa. Cuando se supone que algo significa algo, en algún lugar alguien tiene que hacer una lista de todos los estados de la "cosa" y los significados asociados con cada uno de esos estados. Cuando se trata de un bit, dado que solo puede estar en dos estados diferentes, entonces un bit solo puede significar una de dos cosas. Un código para un bit solo necesitaría dos significados, y uno de esos significados estaría asociado con que el bit esté apagado, y el otro significado estaría asociado con el bit que está encendido.

Así es como se asigna significado a un bit. El bit no contiene ningún significado en sí mismo; No hay espacio en un poco para nada más que la presencia o ausencia de electricidad. El significado se asigna a un bit por algo externo al bit. No hay nada sobre el tráfico o las papas fritas en un momento, solo estamos diciendo que para este tramo en este lugar, conectado a una luz roja que cuelga sobre una intersección, cuando está encendido, debe detenerse, cuando está apagado, puede irse. Otra parte, en una caja registradora en un restaurante de comida rápida, significa poner papas fritas en la bolsa cuando la broca está encendida, o no papas fritas cuando está apagada.

Estos son dos casos de alguien que inventa un código simple de dos elementos. En un caso, el código es: bit on significa papas fritas,

Bit off significa que no hay papas fritas, en el otro caso, bit on significa ir, bit off significa detenerse. Estos dos bits son lo mismo, solo que se usan para diferentes propósitos, y alguien decide cuál será el significado de estos dos bits. El código está escrito en algún lugar de los libros de leyes o en el manual del gerente del restaurante, pero el código no está en el bit. El estado del bit simplemente le dice a alguien qué línea del código se supone que debe creer que es verdadera en el momento actual. Eso es lo que es un código.

Al igual que los espías que pasan mensajes usando un código secreto, el mensaje puede ser visto por otras personas, pero esas otras personas no tienen el código, por lo que no saben lo que significa el mensaje. Tal vez un espía tiene una maceta en el alféizar de la ventana delantera de su apartamento. Cuando la olla está en el lado izquierdo del alféizar, significa "Encuétrame en la estación de tren a la 1:30". Y cuando la maceta está en el lado derecho del alféizar, significa "No hay reunión hoy". Todos los días, el otro espía camina por la calle y mira hacia esa ventana para ver si necesita ir a la estación de tren hoy. Todos los demás que caminan por esa calle pueden ver este mensaje con la misma facilidad, pero no tienen el código, por lo que no significa nada para ellos. Luego, cuando los dos espías se encuentran, pueden pasar una hoja de papel que está escrita en otro código secreto.

Codifican y decodifican el mensaje utilizando un libro de códigos que no llevan cuando se encuentran. Entonces, si su mensaje es interceptado por alguien más, no significará nada para esa otra persona. Alguien que no tenga el libro de códigos no tendrá los significados adecuados para los símbolos en la hoja de papel.

Un bit de ordenador sigue siendo, y siempre será, nada más que un lugar donde hay o no hay electricidad, pero cuando nosotros, como sociedad de seres humanos, usamos un bit para un determinado propósito, le damos sentido al bit. Cuando conectamos un poco a un semáforo en rojo y lo colgamos sobre una intersección, y hacemos que las personas estudien los manuales del conductor antes de darles licencias de conducir, le hemos dado significado a ese poco. Rojo significa 'detenerse', no porque la parte sea capaz de hacer algo a un vehículo que viaja por la

carretera, sino porque nosotros, como personas, estamos de acuerdo en que rojo significa detenerse, y nosotros, al ver esa parte, detendremos nuestro automóvil para evitar ser atropellados por un automóvil que viaja por la calle transversal, y esperamos que todos los demás lo hagan

lo mismo para que podamos estar seguros de que nadie nos golpeará cuando sea nuestro turno de cruzar la intersección.

Así que hay muchas cosas que se pueden hacer con un poco. Puede indicar verdadero o falso, ir o detenerse. Un solo sí o un no puede ser algo importante, como en la respuesta a "¿Quieres casarte conmigo?" o un asunto cotidiano como "¿Quieres papas fritas con eso?"

Pero aún así, hay muchas cosas que no se pueden hacer con un bit, o parecen ser incompatibles con la idea de bits por completo. Puede haber muchos ejemplos de cosas de sí o no en la vida cotidiana, pero hay muchas más cosas que no son un simple sí o no.

En el caso del telégrafo, que indiscutiblemente era solo un bit, ¿cómo puede haber más de dos elementos en el código Morse? La respuesta es que la capacidad de enviar y recibir mensajes dependía de las habilidades y los recuerdos de los operadores en ambos extremos del cable. En el código Morse, si la tecla se presionaba por un tiempo muy corto, eso se llamaba "punto (.)", y si se presionaba por un tiempo un poco más largo, eso se llamaba "guión (-)". A cada letra del alfabeto se le asignó una combinación única de puntos o guiones, y ambos operadores estudiaron el código, lo memorizaron y practicaron su uso.

Por ejemplo, el código de la letra 'N' era el punto de guión (-.) y el código de la letra 'C' era el guión de punto

punto (-.-.). La longitud de los tiempos de encendido era diferente para formar puntos y rayas, y las longitudes de los tiempos de encendido eran diferentes para distinguir entre el tiempo que separa los puntos y los guiones dentro de una letra, el tiempo que separa las letras y el tiempo que separa las palabras.

Necesita un tiempo de inactividad más largo para evitar confundir una 'C' con dos 'N'. La persona receptora tenía que reconocerlos como patrones, es decir, tenía que escuchar y recordar la duración de varios tiempos de encendido y apagado hasta que reconociera una letra. El aparato telegráfico no tenía memoria en absoluto, nunca había ni una letra completa en el cable a la vez, los trozos de letras iban por el cable, para ser ensamblados en puntos y rayas en la mente del operador, luego en cartas, y luego en palabras y oraciones escritas en una

hoja de papel. Entonces, el bit del telégrafo logra más de dos significados al tener varios momentos individuales en los que puede haber encendidos o apagados.

Si una computadora se construyera según los principios del código Morse, solo tendría una bombilla encima que nos mostraría el código. Dado que preferimos ver letras, palabras y oraciones completas en la pantalla simultáneamente, necesitamos algo más que un solo bit y este código antiguo.

Incluso en los ejemplos utilizados en este capítulo, los semáforos reales en realidad tienen tres bits, uno para rojo, uno para amarillo y otro para verde. Si solo tuviera un poco, podría tener una luz roja en la intersección, y cuando estuviera encendida eso significaría detenerse, y cuando estuviera apagada eso significaría continuar. Pero cuando estaba apagado, es posible que se pregunte si realmente estaba apagado o si la bombilla se había quemado. Por lo tanto, usar tres bits es mucho más útil en este caso.

En el mundo real, ya hemos visto que las computadoras pueden contener letras, palabras, oraciones, libros enteros, así como números, imágenes, sonidos y más. Y, sin embargo, todo esto se reduce a nada más que pedazos.

Si queremos que la memoria de nuestra computadora pueda contener más de un encendido o apagado, o sí o no, tendremos que tener algo más que un bit.

Afortunadamente, podemos hacer algo mucho más útil simplemente usando varios bits juntos y luego inventando un código (o tal vez varios códigos) para asignarles algún significado útil.

Una rosa con cualquier otro nombre

Antes de continuar, vamos a introducir un cambio en lo que llamamos algo. Como sabemos, todos los bits de la computadora son lugares donde hay o no hay algo de electricidad. Llamamos a estos estados, "encendido" y "apagado", y eso es exactamente lo que son. Aunque estas son palabras cortas, hay lugares donde es mucho más fácil, claro y simple usar un solo símbolo para describir estos estados. Afortunadamente, no vamos a inventar nada complicado, solo vamos a usar dos símbolos que ya conoces bien, los números cero y uno. De aquí en adelante, cancelaremos 0 y llamaremos a 1. Y a veces todavía usaremos encendido y apagado.

Por lo tanto, el gráfico de nuestra puerta NAND se verá así:

un	b	c
0	0	1
0	1	1
1	0	1
1	1	0

Esto es muy fácil de entender, por supuesto, pero el punto que debe hacerse aquí es que las partes de la computadora no han cambiado, lo único que ha cambiado es lo que nosotros, como personas que miramos la máquina, lo llamamos. El hecho de que llamemos a un poco cero o uno, no significa que de repente hayan aparecido números y estén corriendo dentro de la computadora. Todavía no hay números (ni palabras, ni sonidos, ni imágenes) en una computadora, solo bits, exactamente como se describió anteriormente. Nosotros

podría haberlos llamado más y menos, sí y no, verdadero y falso, cara y cruz, algo y nada, norte y sur, o incluso Bert y Ernie. Pero cero y uno lo harán. Este es solo un código simple de dos artículos. Activado significa 1 y desactivado significa 0.

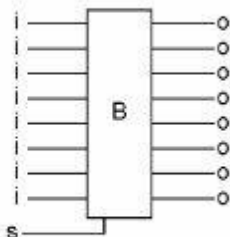
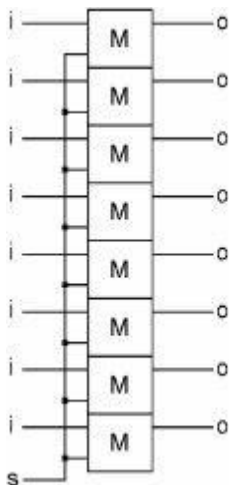
Como comentario aquí, parece haber una tendencia entre los fabricantes de electrodomésticos del mundo a reemplazar los términos obsoletos y anticuados de encendido y apagado con los modernos 0 y 1. En muchos interruptores de encendido ponen un 0 en la posición de apagado y un 1 en la posición de encendido. El primer lugar donde vi esto fue en una computadora personal, y pensé que era una linda novedad, estar en una computadora, pero ahora esta práctica se ha extendido a teléfonos celulares, cafeteras y tableros de automóviles. Pero creo que esto es un error. ¿Entiendes que el código podría haberse definido fácilmente como "apagado significa 1 y encendido significa 0"? La computadora funcionaría exactamente de la misma manera, solo cambiaría la impresión en los manuales técnicos que describen lo que está sucediendo dentro de la computadora.

Cuando ves uno de estos interruptores 0/1, tienes que traducirlo de nuevo de este código informático de uso común a lo que realmente significa, encendido o apagado. Entonces, ¿por qué molestarse? No desea encender su máquina de café '1', desea encender su java para que pueda obtener su java y despertarse ya. Imagínese poner estos símbolos en una máquina para hacer gofres en 1935. Nadie habría tenido idea de lo que significaba. Probablemente sea solo para que los fabricantes no tengan que imprimir interruptores en diferentes idiomas. O tal vez esta tendencia proviene de un deseo altruista de educar al público en el "hecho" moderno de que un 1 es lo mismo que on, pero no es un hecho, es un código arbitrario.

Ocho es suficiente

Para poder representar algo más que simples asuntos de sí/no, lo que vamos a hacer es apilar ocho bits en un solo paquete y usarlos como una sola unidad. Aquí hay un diagrama de cómo se hace. Hemos tomado ocho de nuestros bits de memoria, cada uno todavía tiene su propia entrada de datos 'i' y su propia salida 'o', pero hemos conectado las ocho entradas del conjunto 's' juntas.

Por lo tanto, cuando la 's' simple se enciende y luego se apaga nuevamente, estas ocho 'M' capturarán los estados de sus 'i' correspondientes al mismo tiempo. La imagen de la izquierda muestra las ocho 'M', la de la derecha es lo mismo, solo que un poco más simple.



Este ensamblado tiene un nombre; se llama byte, de ahí la "B" en el diagrama. Hay varias explicaciones contradictorias de dónde vino exactamente esta palabra, pero dado que suena igual que la palabra "morder", puedes pensar en ella como un bocado completo en comparación con una unidad más pequeña, un poco. Solo para mostrarle que los diseñadores de computadoras tienen sentido del humor, cuando usan cuatro bits como unidad, lo llaman mordisco. Así que puedes comer un poquito de pastel de cerezas, o comer un bocado o tomar un byte entero.

Cuando teníamos un poco, simplemente decíamos que su estado era 0 o 1. Ahora que tenemos un byte, escribiremos el contenido del byte así: 0000 0000, y puede ver por qué cambiamos de usar apagado / encendido

a 0/1. Ese

muestra el contenido de cada uno de los ocho bits, en este caso todos son ceros. El espacio en el medio está ahí para que sea un poco más fácil de leer. La mano izquierda

0 o 1 correspondería al bit superior de nuestro byte, y el 0 o 1 más a la derecha representaría el bit inferior.

Como ya sabrás, un bit tiene dos estados posibles en los que puede estar: encendido o apagado. Si tiene dos bits, hay cuatro estados posibles en los que pueden estar esos dos bits. ¿Recuerdas el gráfico que dibujamos para las entradas de la puerta NAND? Había cuatro líneas en el gráfico, una para cada combinación posible de los dos bits de entrada a la puerta, 0-0, 0-1, 1-0 y 1-1.

Observe que el orden de los bits sí importa, es decir, si mira dos bits y solo pregunta cuántos bits están encendidos, solo hay tres posibilidades: ningún bit encendido, un bit encendido o dos bits encendidos. Eso sería llamar a las combinaciones 1-0 y 0-1 lo mismo. Con el propósito de usar múltiples bits para implementar un código, definitivamente nos preocupamos por el orden de los bits en un byte. Cuando hay dos bits, queremos usar las cuatro posibilidades, por lo que tenemos que mantener los bits en orden.

¿Cuántas posibilidades diferentes hay cuando se utilizan ocho bits? Si todo lo que tienes es un bit, puede estar en uno de dos estados. Si agrega un segundo bit, el par tiene el doble de estados que antes porque el bit antiguo tiene sus dos estados mientras que el bit nuevo es unidireccional, y luego el bit antiguo tiene sus dos estados mientras que el nuevo bit es el contrario. Entonces, dos bits tienen cuatro estados. Cuando agrega un tercer bit, los dos primeros tienen cuatro estados con el nuevo bit desactivado y cuatro estados con el nuevo bit activado, para un total de ocho estados. Cada vez que agrega un poco, simplemente duplica el número de estados posibles. Cuatro bits tienen 16 estados, cinco tienen 32, seis tienen 64, siete tienen 128, ocho tienen 256, nueve tienen 512 estados, y así sucesivamente.

Vamos a tomar ocho bits y llamarlo byte. Dado que un bit es una cosa que tiene una ubicación en el espacio, que puede estar en uno de dos estados, entonces un byte es una cosa que tiene ocho ubicaciones separadas en el espacio, cada una de las cuales puede estar encendida o

apagada, que se mantienen en el mismo orden. El byte, tomado en su conjunto, es una ubicación en el espacio que puede estar en cualquiera de los 256 estados en un momento dado, y se puede hacer que cambie su estado con el tiempo.

Códigos

Un poco solo podría representar tipos de cosas de sí / no, pero ahora que tenemos 256 posibilidades, podemos buscar cosas en nuestras vidas que sean un poco más complicadas.

Una de las primeras cosas que podría encajar es el lenguaje escrito. Si miras en un libro y ves todos los diferentes tipos de símbolos que se utilizan para imprimir el libro, verás las 26 letras del alfabeto en mayúsculas y minúsculas. Luego están los números del 0 al 9, y hay signos de puntuación como puntos, comas, comillas, signos de interrogación, paréntesis y varios otros. Luego están los símbolos especiales como el signo 'at' (@), la moneda (\$), y más. Si sumas estos, 52 letras, 10 números, unas pocas docenas para la puntuación y los símbolos, obtienes algo así como 100 símbolos diferentes que pueden aparecer impresos en las páginas del libro promedio.

De aquí en adelante, usaremos la palabra 'carácter' para referirnos a uno de este tipo de cosas, una de las letras, números u otros símbolos que se usan en el lenguaje escrito. Un carácter puede ser una letra, un número, un signo de puntuación o cualquier otro tipo de símbolo.

Así que hemos escrito el lenguaje con unos 100 caracteres diferentes, y nuestro byte con 256 posibilidades, tal vez podamos representar el lenguaje con bytes. Veamos, ¿cómo se pone una 'A' en un byte? No hay nada inherente en un byte que lo asocie con un carácter, y no hay nada inherente en un carácter que tenga algo que ver con bits o bytes. El byte no contiene formas ni imágenes. Dividir un personaje en ocho partes no encuentra ningún fragmento.

La respuesta, como antes, es usar un código para asociar uno de los posibles estados del byte con algo que existe en el mundo real. La letra 'A' estará representada por un patrón particular de 1 y 0 en los bits de un byte. El byte tiene 256 estados posibles diferentes, por lo que alguien debe sentarse con lápiz y papel y enumerar las 256 combinaciones, y al lado de cada una, poner uno de los caracteres que quiere que represente ese patrón. Por supuesto, para cuando

Llega a la línea 101 más o menos, se quedará sin caracteres, por lo que puede agregar todo tipo de símbolos raramente utilizados que se le ocurran, o simplemente puede decir que el resto de las combinaciones no tendrán ningún significado en lo que respecta al lenguaje escrito.

Y así, en los primeros días de las computadoras, cada fabricante se sentó e inventó un código para representar el lenguaje escrito. En algún momento, las diferentes empresas se dieron cuenta de que sería beneficioso que todas usaran el mismo código, en caso de que alguna vez quisieran que las computadoras de su empresa pudieran comunicarse con otra marca. Así que formaron comités, celebraron reuniones e hicieron todo lo que tenían que hacer para llegar a un código en el que todos pudieran estar de acuerdo.

Hay varias versiones de este código diseñadas para diferentes propósitos, y todavía se celebran reuniones hoy en día para llegar a acuerdos sobre varios detalles esotéricos de las cosas. Pero no necesitamos preocuparnos por todo eso para ver cómo funciona una computadora. El código básico que se les ocurrió todavía está en uso hoy en día, y no conozco ninguna razón por la que sea necesario cambiarlo.

El código tiene un nombre, es el: Código Estándar Americano para el Intercambio de Información. Esto generalmente se abrevia como ASCII, pronunciado "aass-key". No necesitamos imprimir todo el código aquí, pero aquí hay un ejemplo. Estos son 20 de los códigos que se les ocurrieron, las primeras 10 letras del alfabeto en mayúsculas y minúsculas:

PARTE DE LA TABLA DE CÓDIGOS ASCII

Un	0100 0001	un	0110 0001
B	0100 0010	b	0110 0010
C	0100 0011	c	0110 0011
D	0100 0100	d	0110 0100
E	0100 0101	e	0110 0101
F	0100 0110	f	0110 0110

G	0100 0111	g	0110 0111
H	0100 1000	h	0110 1000

Yo	0100 1001	Yo	0110 1001
J	0100 1010	j	0110 1010

Cada código es único. Es interesante notar la forma en que organizaron los códigos para que los códigos para mayúsculas y minúsculas de la misma letra usen el mismo código excepto por un bit. El tercer bit de la izquierda está desactivado para todas las letras mayúsculas y activado para todas las letras minúsculas.

Si quisieras poner un mensaje en la pantalla de tu computadora que dijera "Hola Joe", necesitarías nueve bytes. El primer byte tendría el código para la "H" mayúscula, el segundo byte tendría el código para la "e" minúscula, el tercer y cuarto bytes tendrían el código para la "l" minúscula, el quinto byte tendría el código para la "o" minúscula, el sexto byte tendría el código para un espacio en blanco y los bytes siete, ocho y nueve contendrían los códigos para "J", "o" y "e".

Observe que incluso hay un código para un espacio en blanco (por cierto, es 0010 0000). Quizás se pregunte por qué debe haber un código para un espacio en blanco, pero eso solo demuestra lo tontas que son las computadoras. Realmente no contienen oraciones o palabras, solo hay una serie de bytes establecidos con los códigos de la tabla de códigos ASCII que representan los símbolos individuales que usamos en el lenguaje escrito. Y uno de esos "símbolos" es la falta de cualquier símbolo, llamado espacio, que usamos para separar palabras.

Ese espacio nos dice a nosotros, el lector, que este es el final de una palabra y el comienzo de otra. La computadora solo tiene bytes, cada uno de los cuales puede estar en uno de sus 256 estados. El estado en el que se encuentra actualmente un byte no significa nada para la computadora.

Así que tomemos un byte de memoria y establezcamos los bits en 0100 0101. Eso significa que hemos puesto la letra E en el byte, ¿verdad? Pozo... La verdad es que no. Hemos establecido el patrón que aparece junto a la letra E en la tabla de códigos ASCII, pero no hay nada inherente en el byte que tenga que ver con una 'E'. Si Thomas Edison hubiera estado probando ocho de sus nuevas bombillas experimentales y las hubiera tenido colocadas en una fila en un estante, y la primera, La tercera,

cuarta, quinta y séptima bombillas se habían quemado,
las bombillas restantes serían un byte con esto

patrón. Pero no había una sola persona sobre la faz de la Tierra que hubiera mirado esa fila de bombillas y hubiera pensado en la letra 'E', porque ASCII aún no se había inventado. La letra está representada por el código.

Lo único en el byte es el código.

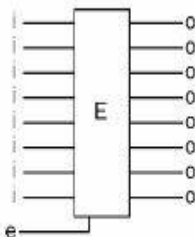
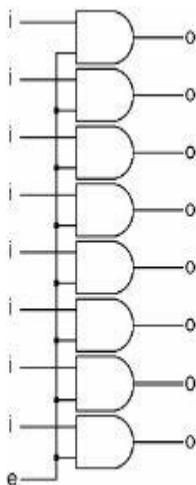
Ahí tienes el tema de los códigos. Un código informático es algo que le permite asociar cada uno de los 256 patrones posibles en un byte con otra cosa.

Otra nota de lenguaje aquí, a veces la palabra código se refiere a la lista completa de patrones y lo que representan, como en "Este mensaje fue escrito con un código secreto". A veces, el código solo se refiere a uno de los patrones, como en "¿Qué código hay en ese byte?" Será bastante obvio por el contexto de qué manera se está utilizando.

Volver al byte

¿Recuerdas el byte de memoria que dibujamos hace unos capítulos? Eran ocho bits de memoria con sus cables 's' conectados entre sí. Casi cada vez que necesitamos recordar un byte dentro de una computadora, también necesitamos una parte adicional que se conecta a la salida del byte. Esta parte adicional consta de ocho puertas Y.

Estas ocho puertas Y, juntas, se denominan "Facilitadoras". El dibujo de la izquierda muestra todas las partes, el dibujo de la derecha es una forma más sencilla de dibujarlo.

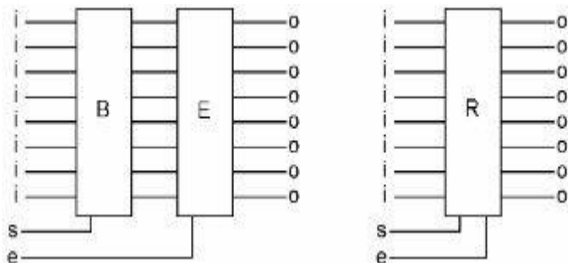


La segunda entrada de las ocho puertas AND están conectadas entre sí y se les da el nombre de 'habilitar' o 'e' para abreviar. Cuando 'e' está apagado, lo que sea que entre en el Habilitador no va más allá, porque el otro lado de cada puerta AND está apagado, por lo que las salidas de esas puertas estarán todas apagadas. Cuando 'e' está activado, las entradas pasan por el Enabler sin cambios a las salidas, 'o'.

Por cierto, cuando las puertas se usan para algo como esto, el nombre "puerta" comienza a tener sentido. Un Enabler permite un byte cuando el bit 'e' es 1 y detiene el byte cuando es 0. Entonces, 'e' está encendido es como abrir una puerta, y 'e' está apagado es como cerrar

una puerta.

Tomaremos nuestro byte y lo conectaremos a un habilitador, como se muestra en el dibujo de la izquierda. Para simplificar una vez más, podemos



dibujarlo como se muestra a la derecha.

Ahora tenemos una combinación que puede almacenar ocho bits. Los captura todos al mismo tiempo, y puede guardarlos para sí mismo o dejarlos salir para usarlos en otro lugar. Esta combinación de un Byte y un Enabler, tiene un nombre, se llama Registro, de ahí la 'R' en el dibujo.

Habrá algunos lugares en este libro donde hay registros que nunca necesitan que se apaguen sus salidas. En esos casos, dibujaremos un registro que solo tiene un bit 'set' y ningún bit 'enable'. Probablemente deberíamos referirnos a estos dispositivos como 'bytes', pero los llamaremos registros de todos modos.

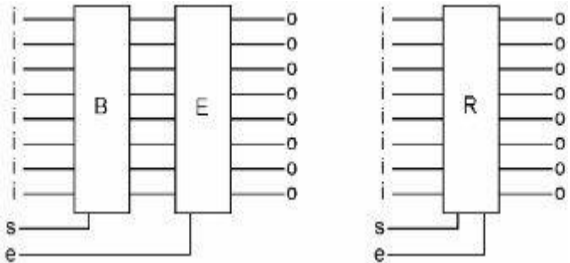
Registro simplemente significa un lugar para registrar algún tipo de información, como un registro de hotel donde todos los huéspedes se registran, o un registro de cheques donde escribe cada cheque que se escribe. En el caso de esta parte de la computadora, registra el estado de los ocho bits de entrada. Sin embargo, este registro es muy limitado, ya que solo puede contener un conjunto de valores; En el registro de un hotel hay una nueva línea para cada huésped. Cada vez que almacena un nuevo estado en un registro de computadora, se pierde el estado anterior de los ocho bits de memoria. Lo único que hay

allí es el valor guardado más recientemente.

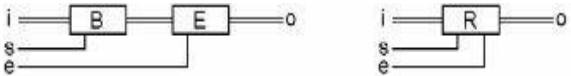
El autobús mágico

Hay muchos lugares en una computadora donde se necesitan ocho cables para conectar registros entre sí. Nuestro registro, por ejemplo, tiene ocho bits de memoria, cada uno de los cuales tiene una entrada y una salida. Para simplificar nuestros diagramas, reemplazaremos nuestros ocho cables con una línea doble.

Así que nuestro registro puede verse como uno de estos:



O bien, podemos simplificarlo y reemplazarlo con uno de estos:



Es exactamente lo mismo, solo que ahorraremos mucha tinta en nuestros dibujos, y serán más fáciles de entender.

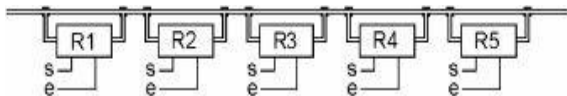
Cuando hay una conexión entre dos de estos haces de cables, un cable de cada paquete se conecta a un cable del otro paquete como se muestra en el diagrama de la izquierda. Pero lo simplificaremos y lo dibujaremos como el diagrama de la derecha.



Ahora, esta agrupación de ocho cables es tan común dentro de las computadoras que tiene un nombre. Se llama autobús. ¿Por qué se llama autobús? Bueno, probablemente tenga que ver con el antiguo término eléctrico 'buss', que significa una barra de metal utilizada como un cable muy grande en lugares como plantas generadoras de energía. Pero también hay una similitud interesante con el tipo de autobús que la gente usa para el transporte.

Un autobús es un vehículo que comúnmente viaja a lo largo de una ruta predeterminada y hace muchas paradas donde las personas suben o bajan. Comienzan en algún lugar y el autobús los lleva a otro lugar donde necesitan estar. En el mundo de las computadoras, un bus es simplemente un conjunto de ocho cables que van a varios lugares dentro de la computadora. Por supuesto, ocho es el número de cables necesarios para transportar un byte de información. Dentro de la computadora, el contenido de los bytes debe llegar desde donde están a otros lugares, por lo que el autobús va a todos estos lugares, y el diseño del registro permite que el contenido de cualquier byte seleccionado suba al autobús y se baje en un destino seleccionado.

En el siguiente ejemplo, tenemos un bus y hay cinco registros, cada uno de los cuales tiene su entrada y salida conectadas al mismo bus.



Si todos los bits 's' y 'e' están apagados, cada registro se establecerá como está y permanecerá así. Si desea copiar la información de R1 a R4, primero active el bit 'e' de R1. Los datos en R1 ahora estarán en el bus y estarán disponibles en las entradas de los cinco

registros. Si luego gira brevemente la parte 's'

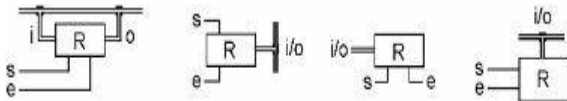
de R4 encendido y apagado, los datos en el bus se capturarán en R4. El byte ha sido copiado. Entonces, un autobús de computadora es un poco como el autobús que transporta personas. Hay una serie de paradas, y los bytes pueden llegar a donde deben ir.

Tenga en cuenta que podemos copiar cualquier byte en cualquier otro byte. Puede copiar R2 en R5 o R4 en R1. El autobús funciona en cualquier dirección. La electricidad que se pone en el bus cuando se habilita cualquier registro va lo más rápido posible a las entradas de todo lo demás en el bus. Incluso puede habilitar un registro en el bus y configurarlo en dos o más registros al mismo tiempo. Lo único que no desea hacer es habilitar las salidas de dos registros en el bus al mismo tiempo.

En términos de los tamaños de los bits, podría verlo de esta manera: cuando se enciende el bit 'e' de R1, los bits en R1 ahora se hacen más largos, son un espacio más grande porque ahora están conectados al bus, por lo que esos 8 bits ahora incluyen R1 y todo el bus. Cuando se enciende el bit 's' de R4, los bits R1 se hacen aún más grandes porque ahora incluyen R1, el bus y R4. Si algo en R1 cambiara de alguna manera en este momento, el autobús y R4 cambiarían inmediatamente con él. Cuando el bit 's' de R4 se apaga, R4 recupera su estado como un byte separado, y cuando el bit 'e' de R1 se apaga, el bus deja de ser parte de R1.

Así que esto es un autobús. Es un haz de ocho cables que normalmente va a muchos lugares.

Una cosa más sobre los registros: hay muchos lugares donde vamos a conectar la entrada y la salida de un registro al mismo bus, por lo que para simplificar aún más, podemos mostrar un paquete de cables etiquetados como 'E / S', que significa entrada y salida. Todos los siguientes son exactamente equivalentes en cuanto a cómo funcionan. La ubicación de los cables en el dibujo se puede ajustar para que esté lo más despejado posible.



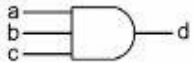
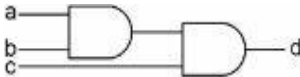
Otra nota de lenguaje: Un byte es una ubicación que puede estar en uno de los 256 estados. A veces hablamos de mover un byte de aquí para allá. Por definición, los bytes no se mueven dentro de la computadora. El byte solo se refiere a la ubicación, pero a veces, cuando alguien quiere referirse a la configuración actual del byte, y debería decir "copiemos el contenido de R1 en R4", simplifican y dicen "mover R1 a R4" o "mover este byte allí". Están usando la palabra byte para referirse al contenido del byte. Una vez más, el contexto suele dejar esto muy claro. En el ejemplo anterior de copiar el contenido de R1 en R4, es posible que escuche que se describe como "mover un byte de R1 a R4". Técnicamente, R1 y R4 son los bytes, que no se mueven, solo el contenido va de un lugar a otro.

Además, el contenido no sale del lugar de donde proviene. Cuando terminas de "mover" un byte, el byte "desde" no ha cambiado, no pierde lo que tenía. En el otro extremo, el patrón que originalmente estaba en el byte "to" ahora está "desaparecido", no fue a ninguna parte, solo fue escrito por la nueva información. El viejo patrón simplemente deja de existir. La nueva información es exactamente la misma que la que todavía está en el primer byte. El byte no se movió, todavía hay dos bytes en dos ubicaciones, pero la información del primer byte se ha copiado en el segundo byte.

Más combinaciones de puertas

Ahora vamos a mostrar solo dos combinaciones más, y luego podremos juntar lo que sabemos hasta ahora, para hacer la primera mitad de una computadora. Así que no te desanimes, solo un poco más y estaremos a mitad de camino a casa.

La primera combinación es muy simple. Es solo una puerta AND con más de dos entradas. Si conecta dos puertas AND como este diagrama de la izquierda, verá que para que 'd' esté activada, las tres entradas, 'a', 'b' y 'c' deben estar activadas. Entonces, esta combinación simplemente se puede dibujar como este diagrama a la



derecha:

Y el gráfico que muestra cómo funciona se ve así:

a	b	c	d
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0

1	1	0	0
1	1	1	1

Imagina reemplazar la entrada 'c' con otra puerta AND, entonces tendrías una puerta AND de cuatro entradas. A continuación, podría reemplazar cualquiera de las cuatro entradas con otra puerta AND y tener una puerta AND de cinco entradas. Esto se puede hacer tantas veces como sea necesario para lo que está haciendo.

A medida que agregue entradas, el gráfico necesitará más y más líneas. Cada vez que agrega otra entrada, duplica la cantidad de combinaciones que pueden tener las entradas. El gráfico que vimos para las dos entradas originales AND tenía cuatro líneas, una para cada posibilidad. La entrada de tres, directamente arriba, tiene ocho líneas. Una puerta de cuatro entradas Y tendrá 16 líneas, una entrada de cinco tendrá 32, etc. Sin embargo, en todos los casos, para una puerta AND, solo una combinación dará como resultado que la salida se encienda, es decir, la línea donde están activadas todas las entradas.

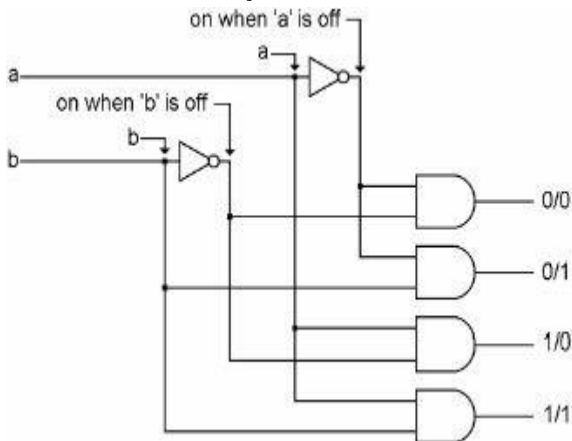
Aquí está la última combinación que necesitamos para hacer la primera mitad de una computadora. Esta combinación es diferente de todo lo que hemos visto hasta ahora, ya que tiene más salidas que entradas. Nuestro primer ejemplo tiene dos entradas y cuatro salidas. No es muy complicado, solo tiene dos puertas NOT y cuatro puertas AND.

En el diagrama a continuación, 'a' y 'b' son las entradas que vienen de la izquierda. Ambos están conectados a puertas NOT. Las puertas NOT generan lo contrario de sus entradas. Hay cuatro cables verticales que bajan por la página que provienen de 'a' y 'b' y los opuestos de 'a' y 'b'. Por lo tanto, para cada 'a' y 'b', hay dos cables que bajan por la página, donde uno de ellos estará encendido si su entrada está encendida y el otro estará encendido si su entrada está desactivada. Ahora colocamos cuatro

puertas AND a la derecha, y conectamos cada una a un par diferente de cables verticales de modo que cada puerta AND se encienda para una diferente de las cuatro combinaciones posibles de 'a' y 'b'. La parte superior

Y la puerta, etiquetada como "0/0", está conectada al cable que está encendido cuando 'a' está apagado, y al cable que está encendido cuando 'b' está apagado, y por lo tanto se enciende cuando 'a' y 'b' son 0. La siguiente puerta AND, "0/1" está conectada al cable que está encendido cuando 'a' está apagado, y 'b', por lo que se enciende cuando 'a' es 0 y 'b' es 1, etc.

Las entradas pueden estar activadas en cualquier combinación, ambos bits apagados, uno encendido, el otro encendido o ambos encendidos. Ninguno, uno o dos en adelante. Las salidas, sin embargo, siempre tendrán una y solo una salida activada y las otras tres desactivadas.



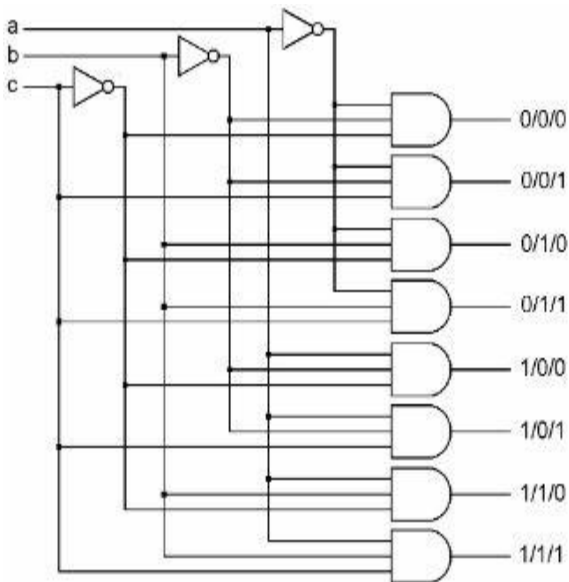
El que está encendido está determinado por los estados actuales de 'a' y 'b'.

un	b	0/0	0/1	1/0	1/1
----	---	-----	-----	-----	-----

0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Esta combinación se llama decodificador. El nombre significa que si considera los cuatro estados posibles de las dos entradas como un código, la salida le dice cuál de los códigos está actualmente en la entrada. Tal vez no sea un gran nombre, pero eso es lo que significó para alguien una vez, y el nombre se quedó. Este decodificador tiene dos entradas, lo que significa que puede haber cuatro combinaciones de los estados de las entradas, y hay cuatro salidas, una correspondiente a cada una de las posibles combinaciones de entrada.

Esto se puede ampliar. Si agregamos una tercera entrada, habría ocho combinaciones de entrada posibles, y si usamos ocho, tres puertas de entrada Y, podríamos construir un decodificador de tres entradas y ocho salidas. Del mismo modo, podríamos construir un decodificador de cuatro entradas y 16 salidas. Los decodificadores se nombran por el número de entradas "X" el número de salidas. Como 2X4, 3X8, 4X16, 5X32, 6X64, etc.

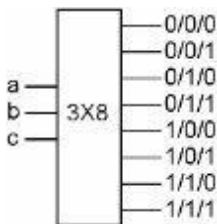


Nuevamente, simplificaremos nuestros dibujos, no mostraremos ninguna de las partes internas o el cableado, solo tendremos un cuadro con un nombre y las entradas y salidas que nos interesan. Hemos visto cómo las puertas NAND hacen puertas NOT y puertas AND, y luego las puertas NOT y las puertas AND hacen un decodificador. Es una caja llena de puertas NAND conectadas para hacer algo útil. Sabemos lo que hace,

uno y

Solo una de las salidas está siempre encendida, y cuál es, está determinada por el estado de las tres entradas.

Eso es todo lo que hace.



Primera mitad de la computadora

Construyamos algo con las piezas que tenemos hasta ahora. En realidad, ahora podemos construir la mitad de lo que hay en una computadora.

Primero, construyamos algo similar de madera (en nuestras mentes), luego volveremos y mostraremos cómo construir una versión de computadora que haga más o menos lo mismo.

Ya sabes, en un hotel, en la recepción, en la pared detrás del empleado, hay una serie de pequeños cubículos de madera, uno para cada habitación del hotel. Ahí es donde guardan las llaves de la habitación adicionales y los mensajes o el correo para los huéspedes. O puede que hayas visto una vieja película en la que alguien en una vieja oficina de correos estaba clasificando el correo. Se sienta en una mesa con una serie de cubículos en la parte de atrás. Tiene una pila de correo sin clasificar sobre la mesa, recoge uno a la vez, lee la dirección y coloca la carta en el cubículo apropiado.

Así que vamos a construir algunos cubículos. El nuestro será de tres pulgadas cuadradas, y habrá dieciséis cubículos de alto y dieciséis cubículos de ancho. Eso es un tamaño total de cuatro pies por cuatro pies, con un total de doscientos cincuenta y seis cubículos.

Ahora agregaremos algo que no tienen en la oficina de correos o en el hotel. Vamos a poner un gran panel de madera justo en frente de los cubículos que es dos veces más ancho que todo, y en el medio tiene una ranura vertical que es lo suficientemente grande como para exponer una columna de 16 cubículos. El panel tendrá ruedas en la parte inferior para que pueda deslizarse hacia la izquierda y hacia la derecha para exponer cualquiera de las columnas verticales de dieciséis cubículos a la vez y cubrir todas las demás columnas.

Tomemos otro panel de madera como el primero, pero gírelo hacia arriba para que sea el doble de alto que nuestros cubículos, y la ranura en el medio va de lado a lado. Este segundo panel se montará justo en frente del primero, en algo así como el marco de una ventana, para que pueda deslizarse hacia arriba y hacia abajo, exponiendo solo una fila de dieciséis cubículos a la vez.

Así que ahora tenemos una serie de 256 cubículos y dos paneles de madera ranurados frente a ellos que solo permiten

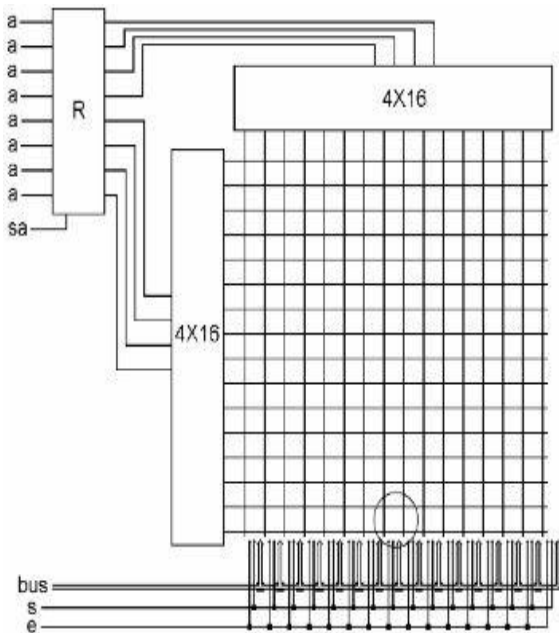
un cubículo a la vez para ser visible. En cada uno de estos cubículos, colocaremos una sola hoja de papel en la que escribiremos una de las posibles combinaciones de ocho ceros y unos.

Este dispositivo de cubículo tiene 256 lugares para almacenar algo. En un momento dado, podemos seleccionar uno y solo uno de esos lugares deslizando los paneles de madera de lado a lado o hacia arriba y hacia abajo. En el cubículo seleccionado, podemos meter la mano y tomar el trozo de papel y leerlo, o reemplazarlo por otro.

Ahora tomaremos las puertas, registros y decodificadores que hemos descrito, y haremos algo con ellos que haga más o menos lo mismo que nuestro dispositivo de cubículo. Esta cosa tendrá 256 lugares en los que almacenar algo, y podremos seleccionar uno y solo uno de esos lugares en un momento dado.

Haciendo referencia al diagrama a continuación, comenzamos con un solo registro. Su entrada 'a' es un bus que proviene de otro lugar en la computadora. Se coloca una combinación de bits en el bus y el bit 'sa' (conjunto a) va 1 y luego

0. Ese patrón de bits ahora se almacena en este registro, que es uno de esos registros cuya salida está siempre encendida. Los primeros cuatro bits de salida están conectados a un decodificador 4X16 y los otros cuatro bits de salida están conectados a otro decodificador 4X16. Las salidas de los dos decodificadores se presentan en un patrón de cuadrícula. Los cables no se tocan entre sí, pero hay 16 por 16, o 256 intersecciones aquí que usaremos pronto. Un decodificador, como se indicó, tiene una y solo una de sus salidas encendidas en cualquier momento, y el resto está apagado. Dado que tenemos dos decodificadores aquí, habrá un cable de rejilla horizontal y un cable de rejilla vertical. Por lo tanto, de estas 256 intersecciones, solo habrá una intersección donde estén los cables horizontales y verticales. La intersección cambiará cada vez que se cambie el valor en R, pero siempre habrá una en la que ambos cables estén encendidos mientras que los otros 255 tendrán solo uno encendido o ninguno.

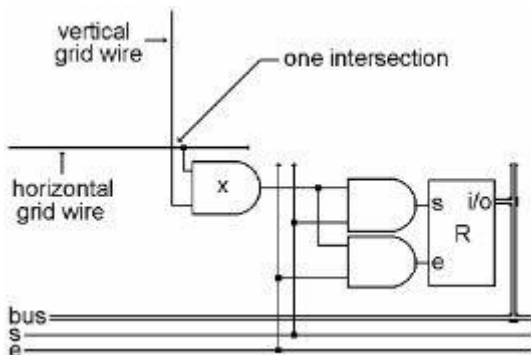


En la parte inferior de este diagrama hay un bus y un bit 's' y 'e', al igual que las conexiones que van a un registro. Como puede ver, van hacia arriba y hacia la cuadrícula. El diagrama no lo muestra, pero suben debajo

la cuadrícula hasta la parte superior, de modo que cada una de las 256 intersecciones tiene un autobús y un bit 's' y 'e' cerca.

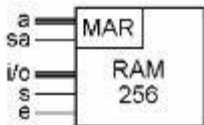
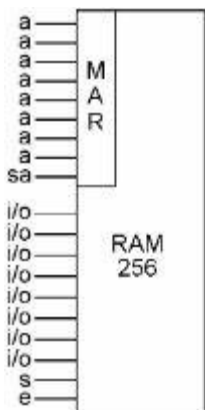
Hay un círculo en el diagrama de arriba, alrededor de una de las intersecciones de la cuadrícula. Lo que está en este círculo se magnifica en el diagrama a continuación, que muestra que hay tres puertas Y y un registro en cada una de las 256 intersecciones. Como podemos ver, hay una puerta AND 'x', conectada a un cable de rejilla vertical y al único cable de rejilla horizontal en esta intersección. Estas puertas 'x' son las únicas cosas conectadas a la red. El resto de las conexiones bajan al bus y a los bits 's' y 'e' en la parte inferior del diagrama. Recuerde que solo hay una intersección donde ambos cables de la red están encendidos. Por lo tanto, hay 256 de estas puertas 'x', pero solo una de ellas tiene su salida encendida en un momento dado.

La salida de esa puerta 'x' va a un lado de cada una de las dos puertas AND más. Estas dos puertas controlan el acceso al conjunto y permiten las entradas del registro en esa intersección. Entonces, cuando una puerta 'x' está apagada, los bits 's' y 'e' de ese registro no se pueden encender. Ese será el caso de 255 de estos registros, aquellos en los que la puerta 'x' está desactivada. Pero una intersección tiene su puerta 'x' activada, y su registro se puede configurar desde el bus, o su contenido se puede habilitar en el bus y enviar a otro lugar usando los bits 's' y 'e' en la parte inferior del diagrama.



Lo anterior es la memoria principal de la computadora. Es la mitad de lo que se necesita para construir una computadora. A veces se le llama con diferentes nombres, pero el nombre más correcto proviene del hecho de que puede seleccionar cualquiera de los 256 bytes en un momento, y luego puede seleccionar inmediatamente cualquier otro de los 256 bytes, y no importa dónde estaba el último, o dónde esté el siguiente, no hay ventaja o desventaja de velocidad en el orden en que seleccione los bytes. Debido a esta cualidad, este es un buen tipo de memoria para usar si desea poder acceder a los bytes de memoria en un orden aleatorio. Entonces, este tipo de memoria se llama "memoria de acceso aleatorio" o "RAM" para abreviar.

Esto es RAM. Utiliza 257 registros. Los registros 256 son ubicaciones de almacenamiento de memoria, un registro se usa para seleccionar una de las ubicaciones de almacenamiento y se denomina "Registro de direcciones de memoria" o "MAR" para abreviar. Ahora que sabemos lo que contiene, podemos hacer un diagrama simplificado como este, y una versión de bus aún más simple:



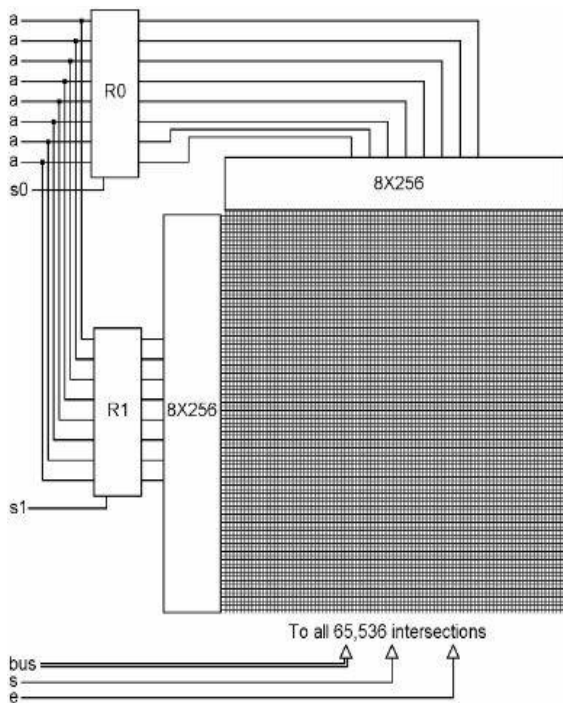
Esto es la mitad de una computadora. Una computadora tiene solo dos partes, y esta es una de ellas. Así que ahora sabes la mitad de lo que hay dentro de una computadora. Cada pieza está hecha de puertas NAND. Eso no fue muy difícil, ¿verdad?

Aquí hay un problema, y es que 256 bytes es un tamaño muy pequeño para la RAM de una computadora. Es posible que podamos salirnos con la nuestra en este libro, pero si desea una computadora real, necesitará una RAM con muchos más bytes para elegir.

Se puede construir una RAM más grande proporcionando dos registros que se utilizan para seleccionar una ubicación de almacenamiento de memoria. Esto permite el uso de decodificadores 8X256 y da como resultado una cuadrícula con 65,536 intersecciones y, por lo tanto, una RAM con 65,536 ubicaciones diferentes en las que almacenar algo.

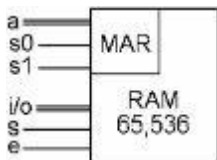
Aquí hay una idea de cómo se vería: (No te molestes

tratando de contar las líneas de la cuadrícula, solo era posible colocar aproximadamente la mitad de ellas en la página impresa).



Un bus transporta un byte a la vez, por lo que seleccionar una de las 65,536 ubicaciones de memoria de esta RAM sería un proceso de dos pasos. Primero, un byte tendría que colocarse en el bus 'a' y establecerse en R0, luego el segundo byte tendría que colocarse en el bus 'a' y establecerse en R1. Ahora puede acceder a la ubicación de memoria deseada con el bus y los bits 's' y 'e' en la parte inferior del dibujo.

Simplificando nuevamente, tenemos algo que se parece mucho a nuestra RAM de 256 bytes, solo tiene un bit de entrada más.



Para el resto de este libro, usaremos la RAM de 256 bytes solo para simplificar las cosas. Si quieres imaginar un ordenador con una memoria RAM más grande, cada vez que enviamos un byte al Registro de Direcciones de Memoria, todo lo que tienes que hacer es imaginar que envías dos bytes en su lugar.

Números

Vamos a volver al tema de los códigos por un momento. Anteriormente vimos un código llamado ASCII que se usa para representar el lenguaje escrito. Bueno, los números también se usan en el lenguaje escrito, por lo que hay códigos ASCII para los dígitos del cero al nueve. Anteriormente vimos 20 de los códigos ASCII para parte del alfabeto, aquí hay 10 más, los códigos para números en lenguaje escrito:

0	0011 0000
1	0011 0001
2	0011 0010
3	0011 0011
4	0011 0100
5	0011 0101
6	0011 0110
7	0011 0111
8	0011 1000
9	0011 1001

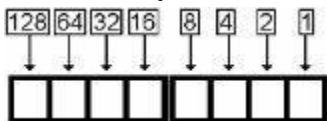
Este es un código muy útil, pero no todo lo que hacen las computadoras tiene que ver con el lenguaje escrito. Para otras tareas, hay otros códigos que se adaptan mejor a esas tareas. Cuando se trata de números, si usa ASCII, un byte puede ser cualquiera de los 10 dígitos del 0 al 9. Pero a veces hay un byte que siempre se usa para almacenar un número, y ese número nunca se imprimirá ni se mostrará en la pantalla. En este caso, podemos usar un código diferente que no desperdicie ninguno de sus posibles estados en letras del alfabeto o cualquier otra cosa que no sean números. Dado que un

byte tiene 256 estados posibles, puede

Haga que este código represente 256 números diferentes. Como queremos incluir cero, este código comienza en cero y sube hasta 255.

Ahora, ¿cómo está organizado este código? El ASCII anterior no se usa en absoluto; Este es un código completamente diferente. Este código no requirió ninguna reunión de comité para inventarlo porque es el código más simple y obvio que usan las computadoras. Es lo más parecido que hay a un código informático "natural".

Dado que este es un capítulo largo, aquí hay una vista previa de este código. Consiste en asignar un valor numérico a cada bit del byte. Para usar este código, simplemente active los bits que suman el número que desea



representar.

Para ver cómo funciona este código, por qué se usa en las computadoras y cómo se eligieron esos valores de bits, examinaremos el tema de los números fuera de las computadoras.

Hay tres sistemas numéricos con los que probablemente esté familiarizado y que podemos analizar. Tal como lo veo, estos tres sistemas se componen de dos ideas o elementos: primero, una lista de símbolos y, segundo, un método para usar esos símbolos.

Probablemente el sistema numérico más antiguo que existe es una cosa simple llamada County Marks. Tiene dos símbolos, "|" y "/". El método para usar estos símbolos es que escribas un "|" para cada una de las primeras cuatro cosas que estás contando, luego para la quinta marca, escribes una "/" en las primeras cuatro. Repites esto una y otra vez todo el tiempo que sea necesario y luego, cuando terminas, cuentas las calificaciones por grupos de cinco: 5, 10, 15, 20, etc. Este sistema es muy bueno para contar las cosas a medida que pasan, digamos tu rebaño de ovejas. A medida que cada animal pasa,

simplemente rasca una marca más, no tienes que tachar '6' y escribir '7'. Este sistema tiene otra ventaja en el sentido de que en realidad hay una marca para cada uno

cosa que se ha contado. Más adelante en el capítulo vamos a hacer algunas cosas interesantes con números que pueden resultar confusos, por lo que para mantener las cosas claras, haremos uso de este antiguo sistema.

¿Recuerdas los números romanos? Es un sistema numérico que también consta de dos elementos. El primer elemento son los símbolos, letras seleccionadas del alfabeto, 'I' para uno, 'V' para cinco, 'X' para diez, 'L' para cincuenta, 'C' para cien, 'D' para quinientos, 'M' para mil. El segundo elemento es un método que le permite representar números que no tienen un solo símbolo. El método romano dice que escribes varios símbolos, los más grandes primero, y los sumas, excepto cuando un símbolo más pequeño está a la izquierda de uno más grande, luego lo restas. Así que 'II' es dos (suma uno y uno) y 'IV' es cuatro (resta uno de cinco). Una de las cosas que hizo que este autor se sintiera muy feliz con la llegada del año 2000 fue el hecho de que los números romanos que representan el año se volvieron mucho más simples. 1999 fue 'MCMXCIX', tienes que hacer tres restas en tu cabeza solo para leer esa. 2000 fue simplemente 'MM'.

El sistema numérico normal que usamos hoy también consta de dos ideas, pero estas son dos ideas muy diferentes que nos llegaron a través de Arabia en lugar de Roma. La primera de estas ideas también trata sobre símbolos, en este caso 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9. Estos dígitos son símbolos que representan una cantidad. La segunda idea es un método al que estamos tan acostumbrados, que lo usamos instintivamente. Este método dice que si escribes un dígito, significa lo que dice. Si escribes dos dígitos uno al lado del otro, el de la derecha significa lo que dice, pero el de la izquierda significa diez veces lo que dice. Si escribes tres dígitos uno al lado del otro, el de la derecha significa lo que dice, el del medio significa diez veces lo que dice y el de la izquierda significa cien veces lo que dice. Cuando quieres expresar un número mayor que 9, lo haces usando varios dígitos, y usas este método que dice que el número de posiciones a la izquierda del primer dígito te dice cuántas veces lo multiplicas por diez antes de sumarlas. Entonces, si tienes 246 manzanas, eso significa que tienes doscientas manzanas más cuarenta manzanas más seis manzanas.

Entonces, ¿cómo funciona esto? Un número de cualquier cantidad se puede escribir con los dígitos del cero al nueve, pero cuando vas más allá de nueve, tienes que usar dos dígitos. Cuando superas los noventa y nueve, tienes que usar tres dígitos. Por encima de novecientos noventa y nueve, vas a cuatro dígitos, etc. Si está contando hacia arriba, los números en cualquiera de las posiciones dan vueltas y vueltas: cero a nueve, luego cero a nueve nuevamente, y así sucesivamente, y cada vez que pasa de nueve a cero, aumenta el dígito a la izquierda en 1. Por lo tanto, solo tiene diez símbolos, pero puede usar más de uno de ellos según sea necesario y sus posiciones entre sí especifican su valor total.

Hay algo extraño en esto en el sentido de que el sistema se basa en diez, pero no hay un símbolo único para diez. Por otro lado, hay algo correcto en esto: los símbolos del '0' al '9' forman diez símbolos diferentes. Si también tuviéramos un solo símbolo para diez, en realidad habría once símbolos diferentes. Así que cualquiera que pensara en esto era bastante inteligente.


Una de las nuevas ideas en este sistema árabe era tener un símbolo para el cero. Esto es útil si quieres decir que tienes cero manzanas, pero también es necesario mantener rectas las posiciones de los dígitos. Si tienes 50 manzanas o 107 manzanas, necesitas los ceros en los números para saber en qué posición está realmente cada dígito, para que puedas multiplicar por diez el número correcto de veces.



Ahora bien, estas dos ideas en el sistema numérico arábigo (los dígitos y el método) tienen una cosa en común. Ambos tienen el número diez asociado con ellos. Hay diez dígitos diferentes, y a medida que agrega dígitos al lado izquierdo de un número, cada posición vale diez veces más que la anterior.

En la escuela, cuando enseñan a los niños por primera vez sobre los números, dicen algo acerca de que nuestro sistema numérico se basa en el número diez, porque tenemos diez dedos. Así que aquí hay una pregunta extraña: ¿Qué pasaría si este sistema numérico hubiera sido inventado por perezosos de tres dedos? Solo tienen tres dedos en cada mano y no tienen pulgares. Habrían inventado un sistema numérico con solo seis dígitos: 0, 1, 2, 3, 4 y 5.

¿Podría funcionar esto? Si tuvieras ocho manzanas, ¿cómo lo escribirías? No hay un número '8' en este sistema. La respuesta es que, desde la primera idea, el

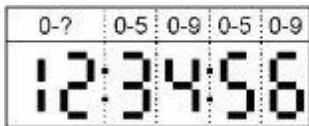
dígitos, se cambió para tener solo seis dígitos, entonces la segunda idea, el método, también tendría que cambiarse para que a medida que agregue posiciones a la izquierda, cada una tendría que multiplicarse por seises en lugar de decenas. Entonces este sistema funcionaría. Mientras cuentas tus manzanas, dirías "0, 1, 2, 3, 4, 5 ...". ¿Y luego qué? No hay '6' para el siguiente número. Bueno, según el método, cuando quieras ir más allá del dígito más alto, vuelves al '0' y agregas un '1' a la izquierda. OK, "0, 1, 2, 3, 4, 5, 10, 11, 12". Ahora has contado todas tus manzanas.

¿Qué significaría este '12'? Serían tantos: . Supongo que lo llamarías ocho, pero lo escribirías '12'. Muy extraño, pero funciona: 1 por seis más dos es igual a ocho manzanas, sigue el método árabe, pero se basa en seis en lugar de diez. Si continúas contando, cuando llegas a 15,

que es  (uno por seis más cinco), el siguiente número sería 20, pero el 2 significaría dos seises, o tantos: . Y 55 serían seguidos por 100. El '1' en esa tercera posición sería cuántos '36' había (seis por seis)

Este es un sistema numérico muy extraño, pero adivina qué, ya lo usas en tu vida cotidiana. Sí, piense en la forma en que escribimos el tiempo, o en el tipo de reloj que muestra los números en su cara. El dígito derecho de los minutos y segundos sigue nuestros números normales, 0-9, 0-9, una y otra vez. Pero el dígito izquierdo de los minutos y segundos solo va de 0 a 5. Después de 59 minutos, el reloj pasa a la siguiente hora y 00 minutos. Hay 60 minutos en una hora, numerados del 00 al 59. Esa posición izquierda nunca supera los 5. Esa posición utiliza el sistema numérico basado en seis símbolos (0-5). La parte de la hora del reloj dice cuántos 60 hay, aunque nunca verá un 60 en la esfera del reloj. Y estás tan acostumbrado a esto que no tienes que pensar en ello. Cuando el reloj marca la 1:30, sabes que está a medio camino entre la 1:00 y las 2:00, no tienes que hacer ningún cálculo en tu cabeza para averiguarlo. ¿Alguna vez has tenido que agregar tiempo? Si agrega 40 minutos y 40 minutos, obtienes 80 minutos, pero para escribir eso en horas y minutos, tienes que calcular cuántos 60 hay en 80, en este caso 1, luego calcular cuántos minutos hay más allá de 60, en este

Caso 20. Así que escribes 1:20. El 1 representa 60 minutos, agregue 20 y tendrá sus 80 nuevamente. Así que esto es bastante complicado, ¿dos sistemas numéricos diferentes en el mismo número! Pero ya lo has estado usando toda tu vida.



Las posiciones de las horas son aún más extrañas. En un reloj de 12 horas, se salta cero y va de 1 a 12 a.m., luego de 1 a 12 p.m. En un reloj de 24 horas, va de 00 a 23. No intentaremos analizarlos. Lo que queríamos decir es que ya está familiarizado con los sistemas numéricos basados en números distintos del diez.

Podrías inventar un sistema numérico para cualquier cantidad de dígitos, 10 o 6 como hemos visto anteriormente, o 3 o 14 o cualquier número que elijas. Pero la más simple sería si solo tuviera 2 dígitos, 0 y 1. ¿Cómo funcionaría este?

Bueno, contarías 0, 1 ... Y luego ya te quedas sin dígitos, así que vuelves a 0 y agregas 1 a la izquierda, haciendo que el siguiente número sea 10, luego 11, luego te quedas sin dígitos nuevamente, así que 100, luego 101, 110, 111 y luego 1000. Este sistema se basa en dos, por lo que solo hay dos dígitos, y a medida que agregas posiciones a la izquierda, cada uno vale dos veces más que el anterior. La posición derecha significa lo que dice, la siguiente a la izquierda significa dos veces lo que dice, la siguiente significa cuatro veces lo que dice, la siguiente significa ocho veces, etc. Cuando llegas a tener solo dos dígitos posibles, no tienes que hacer muchas multiplicaciones para calcular el valor total de una posición. En la posición que vale 'ocho', por ejemplo, solo puede haber un uno, que significa un 'ocho', o un cero, que significa 'sin ochos'.

Mientras estamos en eso, imaginemos un animal muy extraño con ocho dedos en cada mano. Ese animal habría inventado números basados en dieciséis. En su sistema,

podrían escribir de diez a quince cada uno con un

símbolo único. Solo cuando llegaban a los dieciséis volvían a 0 y necesitaban poner un 1 en la posición de la izquierda. Para ver cómo funcionaría esto, necesitamos seis nuevos símbolos, así que usamos las primeras seis letras del alfabeto. 'A' significará diez, 'B' significará once, 'C' significará doce, 'D' significará trece, 'E' significará catorce y 'F' significará quince. Solo después de usar los dieciséis símbolos en la posición correcta nos quedaremos sin símbolos, y el siguiente número será dieciséis, escrito '10' en este sistema. Si está familiarizado con el sistema de pesas de libras y onzas, es algo así como este sistema. Hay 16 onzas en una libra, por lo que sabe que 8 onzas es media libra. Agregar 9 onzas y 9 onzas sale a 1 libra 2 onzas.

Aquí hay una tabla que muestra cinco sistemas numéricos diferentes. La primera columna es el antiguo sistema de marcas de conteo para mantenerlo sensato.

Tally	0-9	0-5	0-1	0-F
.....	0	0	0	0
I	1	1	1	1
II	2	2	10	2
III	3	3	11	3
IIII	4	4	100	4
IIII I	5	5	101	5
IIII II	6	10	110	6
IIII III	7	11	111	7
IIII IIII	8	12	1000	8
IIII IIII I	9	13	1001	9
IIII IIII II	10	14	1010	A
IIII IIII III	11	15	1011	B
IIII IIII IIII	12	20	1100	C
IIII IIII IIII I	13	21	1101	D
IIII IIII IIII II	14	22	1110	E
IIII IIII IIII III	15	23	1111	F
IIII IIII IIII IIII	16	24	10000	10
IIII IIII IIII IIII I	17	25	10001	11
IIII IIII IIII IIII II	18	30	10010	12

Nuestros números normales del 0 al 9 se llaman sistema decimal, porque 'dec' significa diez en algún idioma antiguo. El 0-

5 se llamaría sistema senario, porque 'sen' significa seis en algún otro idioma antiguo. Esta noticia

sistema con solo 0 y 1 se llama sistema binario porque 'bi' significa dos, también debido a algún lenguaje antiguo. Este otro nuevo sistema, el sistema 0-F, se llamará sistema hexadecimal, porque 'hex' es otra palabra antigua que significa seis y 'dec' todavía significa diez, por lo que es el sistema seis más diez.

Otro método para nombrar diferentes sistemas numéricos es llamarlos por el número en el que se basan, como 'base 10' o 'base 2', etc. que significa decimal o binario, etc.

Pero observe que el número después de la palabra 'base' está escrito en el sistema decimal. '2' escrito en binario es '10', por lo que 'base 10' significaría binario si el '10' se escribiera en binario. De hecho, ¡cada sistema numérico sería 'base 10' si el '10' estuviera escrito en los números de ese sistema! Así que podríamos hablar de base 2, base 6, base 10 y base 16 si quisiéramos, siempre y cuando recordemos que esos números base están escritos en decimal. Si hablamos de binario, senario, decimal y hexadecimal, es lo mismo, solo que posiblemente un poco menos confuso.

Nuevamente, en nuestros números decimales normales, la posición más a la derecha es el número de unidades. La siguiente posición a la izquierda es el número de decenas, etc. Cada posición vale diez veces la anterior. En el sistema binario, la posición más a la derecha es también el número de unos, pero la siguiente posición a la izquierda es el número de 'dos', la siguiente a la izquierda es el número de 'cuatros', la siguiente es 'ochos'. Cada posición vale dos veces la cantidad a su derecha. Dado que cada posición tiene solo dos valores posibles, cero o uno, esto es algo que podríamos usar en un byte.

Este es el punto de este capítulo. El sistema numérico binario es una coincidencia "natural" con las capacidades de las piezas de la computadora. Podemos usarlo como código, con off representando cero y on representando uno, siguiendo el método del número arábigo con solo dos símbolos. En un byte, tenemos ocho bits. Cuando usamos este código, el bit de la derecha valdrá 1 cuando el bit esté encendido o 0 cuando esté apagado. El siguiente bit a la izquierda valdrá 2 cuando esté encendido, o 0 cuando esté apagado. El siguiente a la izquierda es 4, y así sucesivamente con 8, 16, 32, 64 y 128. En el orden en que

normalmente los vemos, los valores de los ocho bits se ven así: 128 64 32 16 8 4 2 1.

En este código, 0000 0001 significa uno, 0001 0000 significa dieciséis, 0001 0001 significa diecisiete (dieciséis más uno), 1111 1111 significa 255, etc. En un byte de ocho bits, podemos representar un número entre 0 y 255. Este código se llama "código numérico binario".

La computadora funciona bien con este arreglo, pero es molesto para las personas usarlo. Solo decir lo que hay en un byte es un problema. Si tienes 0000 0010, puedes llamarlo "cero cero cero cero cero cero binario" o puedes traducirlo mentalmente a decimal y llamarlo "dos", y eso es lo que generalmente se hace. En este libro, cuando se deletrea un número, como 'doce', significa 12 en nuestro sistema decimal. Un binario 0000 0100 se llamaría 'cuatro', porque eso es lo que resulta ser en decimal.

En realidad, en la industria informática, la gente a menudo usa hexadecimal (y simplemente lo llaman 'hex'). Si observa el gráfico anterior, puede ver que cuatro dígitos de binario se pueden expresar con un dígito hexadecimal. Si tiene un byte que contiene 0011 1100, puede traducirlo a 60 decimales o simplemente llamarlo "hexadecimal 3C". Ahora no te preocupes, no vamos a usar hexadecimal en este libro, pero es posible que hayas visto este tipo de números en alguna parte, y ahora sabes de qué se trataba.

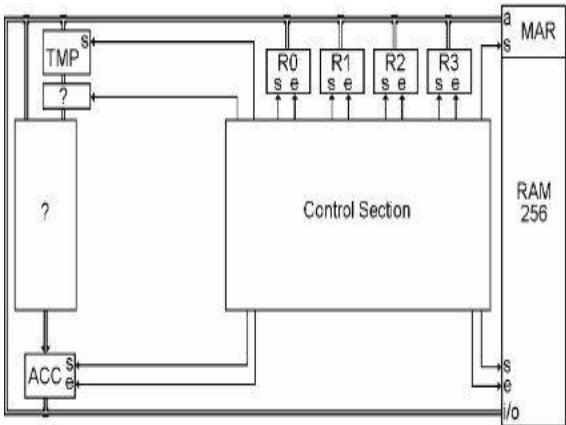
Direcciones

Ahora que tenemos el código numérico binario, podemos usarlo para varios propósitos en nuestra computadora. Uno de los primeros lugares en los que lo usaremos es en el Registro de direcciones de memoria. El patrón de bits que colocamos en este registro usará el código numérico binario. Los bits de este número en MAR luego seleccionan una de las 256 ubicaciones de almacenamiento de RAM. Se considera que el número en MAR es un número entre 0 y 255 y, por lo tanto, se puede considerar que cada uno de los 256 bytes de RAM tiene una dirección.

Esto es bastante simple, pero es necesario hacer un punto aquí sobre lo que se entiende exactamente por una dirección dentro de una computadora. En un vecindario de casas, cada casa tiene una dirección, como 125 Maple Street. Hay un letrero en la esquina que dice "Maple St." y en la casa están escritos los números "125". Esta es la forma en que normalmente pensamos en las direcciones. El punto a destacar aquí es que las casas y calles tienen números o nombres escritos en ellas. En la computadora, el byte no tiene ninguna información de identificación ni está contenido en él. Es simplemente el byte que se selecciona cuando se coloca ese número en el registro de direcciones de memoria. El byte se selecciona en virtud de dónde está, no por ningún otro factor que esté contenido en esa ubicación. Imagina un vecindario de casas que tuviera dieciséis calles y dieciséis casas en cada calle. Imagina que las calles no tienen letreros y las casas no tienen números escritos en ellas. Todavía podría encontrar cualquier casa específica si le dijeran, por ejemplo, que vaya a 'la cuarta casa en la calle séptima'. Esa casa todavía tiene una dirección, es decir, un método para localizarla, simplemente no tiene ninguna información de identificación en el lugar. Entonces, la dirección de una computadora es solo un número que hace que se seleccione un cierto byte cuando esa dirección se coloca en el registro de direcciones de memoria.

La otra mitad de la computadora

La otra mitad de la computadora también está hecha en última instancia de nada más que puertas NAND, y probablemente tenga menos partes totales que la RAM que hemos construido, pero no está diseñada de manera tan regular y repetitiva, por lo que tomará un poco más de tiempo explicarla. Llamaremos a esta mitad de la computadora la "Unidad Central de Procesamiento", o CPU para abreviar, porque hace algo con y para los bytes en la RAM. Los "procesa", y veremos qué significa eso en los próximos capítulos. Lo que es común a ambos lados de la



computadora es el bus.

Aquí están los inicios de la CPU. La RAM se muestra a la derecha y el bus hace un gran bucle entre las dos conexiones de bus en la RAM. La CPU comienza con seis

registros conectados al bus. Estos seis registros son todos los lugares que la CPU usará para "procesar" bytes. Eso no es tan complicado, ¿verdad?

El cuadro grande etiquetado como "Sección de control" en el medio del diagrama se examinará en detalle más adelante. Controla todos los bits de "configuración" y "habilitación" en la CPU y la RAM. Los cuadros con los signos de interrogación se explicarán inmediatamente después de este capítulo. Por ahora, vamos a ver dónde pueden ir los bytes dentro de la CPU.

R0, R1, R2 y R3 son registros que se utilizan como almacenamiento a corto plazo para los bytes necesarios en la CPU. Sus entradas y salidas están conectadas al bus. Se pueden utilizar para muchos propósitos diferentes, por lo que se conocen como "registros de propósito general".

El registro llamado 'TMP' significa temporal. Su entrada proviene del bus y su salida desciende a una y luego a la otra de las casillas con signos de interrogación. TMP tiene un bit 'set', pero no un bit 'enable' porque nunca tenemos una razón para apagar su salida.

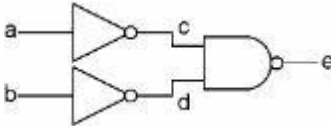
El último registro se llama acumulador, o ACC para abreviar. Esta es una palabra que proviene de los días de las viejas máquinas sumadoras mecánicas (antes de 1970). Supongo que significaba que a medida que sumaba una columna de números, "acumularía" un total acumulado. En una computadora, solo significa que almacena temporalmente un byte que proviene de esa gran casilla de interrogación. La salida de ACC se conecta a nuestro viejo amigo, el bus, para que pueda enviarse a otro lugar según sea necesario.

Los registros en la CPU y la RAM son los lugares de donde proviene y va el contenido de los bytes a medida que funciona la computadora. Todos los movimientos implican habilitar un registro en el bus y configurar el contenido del bus en otro registro.

Ahora veremos lo que hay en esos cuadros con los signos de interrogación.

Más puertas

Hemos usado puertas NAND, AND y NOT hasta ahora. Hay dos puertas combinadas más que debemos definir. El primero está construido así:



Todo lo que hace es NO las dos entradas a una de nuestras viejas puertas NAND. Aquí está el gráfico, que muestra los cables intermedios para que sea fácil de seguir.

a	b	c	d	e
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

En este caso, cuando ambas entradas están apagadas, la salida está apagada, pero si 'a' o 'b' está encendida, o ambas, entonces la salida estará encendida. Así que tiene otro nombre muy simple, se llama "puerta OR". En lugar de dibujar todas las partes, tiene su propio diagrama en forma de escudo. El diagrama y el gráfico



se ven así:

a	b	c
0	0	0

0	1	1
---	---	---

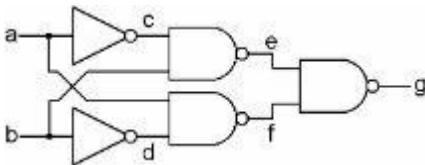
1	0	1
1	1	1

Al igual que la puerta AND, puede crear puertas OR con más de dos entradas. Simplemente agregue otra puerta OR en lugar de una de las entradas, y luego tendrá tres entradas, cualquiera de las cuales encenderá la salida. También como la puerta AND, cada vez que agregue una entrada, el número de líneas en el gráfico se duplicará. Con la puerta OR, solo la línea que tiene todas las entradas desactivadas tendrá la salida desactivada. El resto de las líneas mostrarán que la salida está activada.

La última puerta combinada que necesitamos aquí requiere cinco puertas para hacerla, pero lo que finalmente hace es bastante simple.

Similar a la puerta OR, la salida está encendida cuando cualquiera de las entradas está activada, pero en esta versión, la salida se vuelve a apagar si ambas entradas están activadas. Por lo tanto, se llama puerta OR exclusiva, o puerta XOR para abreviar. La salida está activada si la otra entrada está activada, exclusivamente. Solo si es OR, no si es AND. Otra forma de verlo es que la salida se enciende si una y solo una entrada está encendida.

Otra forma de verlo es que la salida está apagada si las entradas son las mismas y activada si las entradas son diferentes.



un	b	c	d	e	f	g
0	0	1	1	1	1	0

0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	1	0	0	1	1	0

El diagrama simplificado se parece a una puerta OR, pero tiene una línea curva doble en el lado de entrada. El diagrama y el gráfico se ven así:



a	b	c
0	0	0
0	1	1
1	0	1
1	1	0

Ahora tenemos cuatro tipos de puertas que toman dos entradas y hacen una salida. Son NAND, AND, OR y XOR. Aquí hay un gráfico que lo hace bastante simple:

a	b	NAND	AND	OR	XOR
0	0	1	0	0	0
0	1	1	0	1	1
1	0	1	0	1	1
1	1	0	1	1	0

Para las cuatro combinaciones de entrada posibles de 'a' y 'b', cada tipo de puerta tiene su propio conjunto de estados de salida, y los nombres de las puertas pueden ayudarlo a recordar cuál es cuál.

A pesar del hecho de que todo en la computadora está hecho de puertas NAND, ¡no vamos a usar ninguna puerta NAND por sí mismas en esta computadora! Ahora que los hemos usado para construir las puertas AND, OR, XOR y NOT, y el bit de memoria, hemos terminado con la puerta NAND. Gracias Sr. NAND gate, adiós por ahora.

Jugando con los bytes

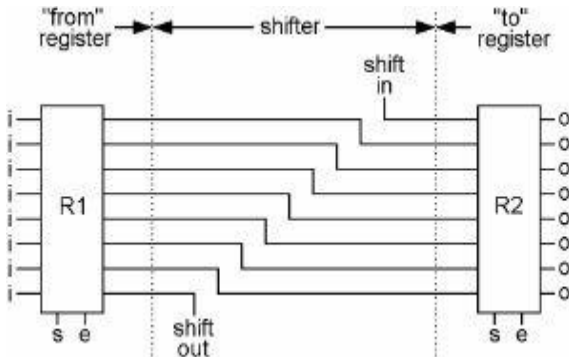
Las puertas individuales funcionan con brocas. Dos bits adentro, un poco afuera. Pero la RAM almacena y recupera un byte a la vez. Y el bus se mueve un byte a la vez. Aquí en la CPU, queremos poder trabajar en un byte completo a la vez. Queremos algunas 'puertas' que afecten a un byte completo. En el capítulo sobre el bus, vimos cómo se puede copiar el contenido de un byte de un registro a otro. Esto generalmente se conoce como mover un byte. Ahora vamos a ver algunas variaciones sobre esto.

Primero veremos tres formas en que podemos cambiar el contenido de un byte a medida que se mueve de un registro a otro. En segundo lugar, veremos cuatro formas en las que podemos tomar el contenido de dos bytes y hacer que interactúen entre sí para crear el contenido de un tercer byte.

Estas son todas las cosas que las computadoras realmente hacen con bytes. En última instancia, todas las cosas se reducen a estas siete operaciones.

Las palancas de cambios izquierda y derecha

La palanca de cambios es muy fácil de construir. No usa ninguna puerta en absoluto, solo conecta el autobús de manera un poco extraña. Se hace así:



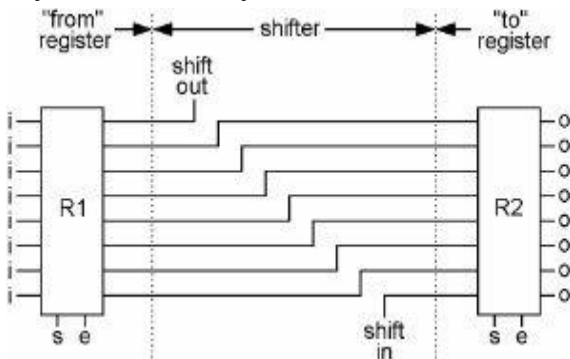
Esto muestra dos registros conectados por un desplazador derecho. La palanca de cambios son solo los cables entre los dos registros. Cuando se enciende el bit 'e' de R1 y se enciende y luego se apaga el bit 's' de R2, todos los bits de R1 se copian en R2, pero se desplazan sobre una posición. El que está en la parte inferior (desplazamiento hacia afuera) se puede conectar a algún otro bit en la computadora, pero a menudo se conecta de nuevo al que está en la parte superior (desplazamiento hacia adentro) y cuando se hace, el bit más a la derecha se envuelve alrededor del bit más a la izquierda en el otro extremo del byte.

Una palanca de cambios derecha cambiará 0100 0010 a 0010 0001.

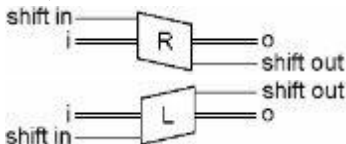
Si 'shift out' está conectado a 'shift in', un desplazamiento a la derecha cambiará 0001 1001 a 1000 1100

Una palanca de cambios izquierda cambiará 0100 0010 a 1000 0100. El

La palanca de cambios izquierda está conectada así:



Una vez más, tenemos versiones de bus de estos dibujos. Cada uno tiene un bus 'i' y 'o', y también un bit de entrada y salida, como este:



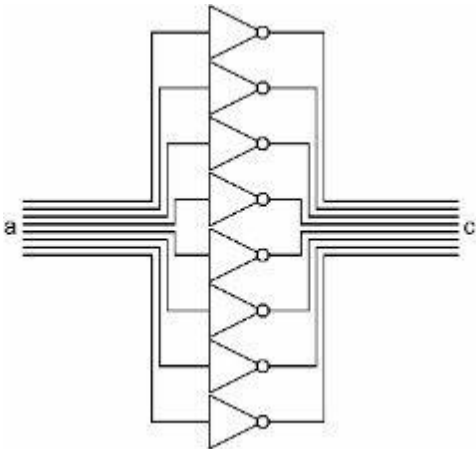
Ahora, ¿de qué sirve esto? Las mentes de los programadores han ideado todo tipo de cosas, pero aquí hay una interesante. Suponga que está utilizando el código numérico binario. Tiene el número 0000 0110 en R1. Eso resulta en el número decimal 6. Ahora cambie ese código a la izquierda a R2. R2 será entonces 0000 1100. Esto sale al número decimal 12. ¿Qué sabes, acabamos de

multiplicado el número por 2. Esta es la base de cómo se realiza la multiplicación en nuestra computadora. Cómo multiplicas por algo que no sea 2 se verá más adelante, pero así de simple es, solo cambia los bits. Esto es similar a algo que hacemos con los números decimales. Si desea multiplicar algo por diez, simplemente agregue un cero al lado derecho, cambiando efectivamente cada dígito a la izquierda una posición. En el sistema binario, esto solo da como resultado multiplicar por dos porque dos es en lo que se basa el sistema.

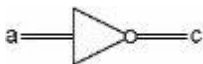
Así que esa es la palanca de cambios, sin puertas en absoluto.

El NOTter

Este dispositivo conecta dos registros con ocho puertas NOT. Cada bit se cambiará a su opuesto. Si comienza con 0110 1000, terminará con 1001 0111. Esta operación se utiliza para muchos propósitos, el primero es en algunas funciones aritméticas. Veremos exactamente cómo funciona esto poco después de describir algunas otras cosas. Otro nombre para una puerta NOT es un "inversor", porque hace lo contrario de lo que le das, lo pone boca abajo o lo invierte.

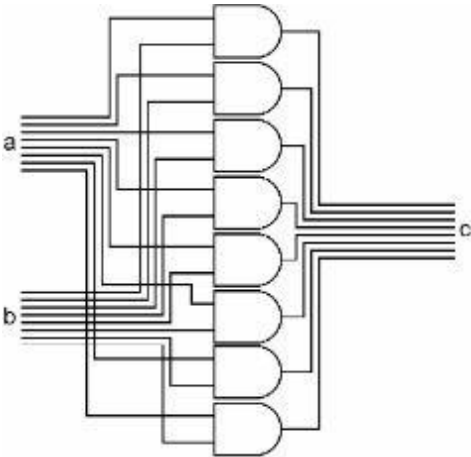


Dado que la entrada y la salida son ocho cables, simplificaremos usando nuestra imagen de tipo bus.



El ANDer

El ANDer toma dos bytes de entrada y AND cada bit de esos dos en un tercer byte. Como puede ver, los ocho bits del bus de entrada 'a' están conectados al lado superior de ocho puertas AND. Los ocho bits del bus de entrada 'b' están conectados al lado inferior de las mismas ocho puertas AND. Las salidas de las ocho puertas AND forman la salida de bus 'c' de este ensamblaje. Con esto, podemos Y dos bytes juntos para formar un tercer byte.



Hay muchos usos para esto. Por ejemplo, puede asegurarse de que un código de letra ASCII esté en mayúsculas. Si tiene el código de la letra 'e' en R0, 0110 0101, podría

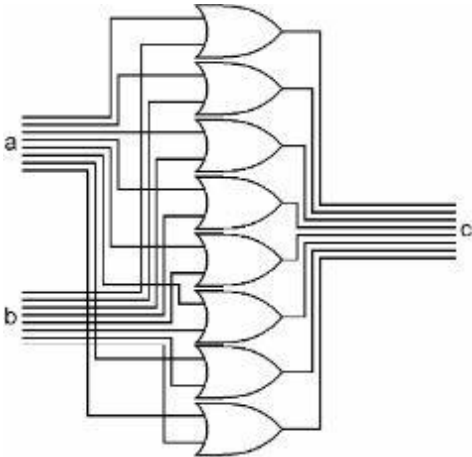
coloque el patrón de bits 1101 1111 en R1 y luego Y R1 y R0 y vuelva a poner la respuesta en R0. Todos los bits que estaban activados en R0 se copiarán de nuevo en R0 excepto el tercer bit. Ya sea que el tercer bit haya estado encendido o apagado antes, ahora estará apagado. R0 ahora contendrá 0100 0101, el código ASCII para 'E'. Esto funciona para todos los códigos de letras ASCII debido a la forma en que está diseñado ASCII.

Aquí hay una imagen de tipo bus más simple para el ANDer.



El ORer

El ORer toma dos bytes de entrada y OR cada bit de esos dos en un tercer byte. Como puede ver, los ocho bits del bus de entrada 'a' están conectados a la parte superior de ocho puertas OR. Los ocho bits del bus de entrada 'b' están conectados al lado inferior de las mismas ocho puertas OR. Las salidas de las ocho puertas OR son la salida de bus 'c' de este conjunto. Con esto, podemos OR dos bytes juntos para formar un tercer byte.



¿De qué sirve esto? Hay muchos, pero aquí hay uno de ellos. Digamos que tiene el código ASCII para la letra 'E' en R0, 0100 0101. Si quieres que esta carta sea

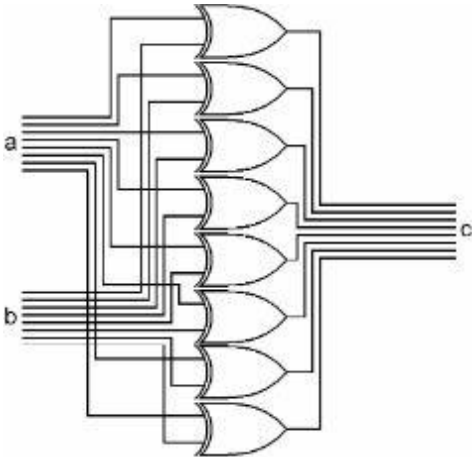
en minúsculas, podría poner el patrón de bits 0010 0000 en R1 y luego O R0 y R1 y volver a poner la respuesta en R0. ¿Qué pasará? Todos los bits de R0 se copiarán de nuevo en R0 como estaban, excepto que el tercer bit ahora estará encendido sin importar lo que haya sido. R0 ahora será 0110 0101, el código ASCII para 'e'. Esto funcionará sin importar qué código de letra ASCII haya en R0 debido a la forma en que se diseñó ASCII.

Aquí hay una imagen de tipo bus más simple para el ORer.



El ORer exclusivo

El XORer toma dos bytes de entrada, y los XOR cada bit de esos dos en un tercer byte. Como puede ver, los ocho bits del bus de entrada 'a' están conectados al lado superior de las mismas ocho puertas XOR. Los ocho bits del bus de entrada 'b' están conectados al lado inferior de las mismas ocho puertas XOR. Las salidas de las ocho puertas XOR son la salida de bus 'c' de este conjunto. Con esto, podemos XOR dos bytes juntos para formar un tercer byte.



¿De qué sirve esto? Una vez más, a las personas imaginativas se les han ocurrido muchos de los usos. Pero aquí está uno de ellos. Digamos que tiene un código en R1 y otro código en R2. Quieres

para averiguar si esos dos códigos son iguales. Así que XOR R1 y R2 en R1. Si R1 y R2 contenían los mismos patrones, entonces R1 ahora será todo ceros. No importa qué patrón de 0 y 1 haya en R1, si R2 contenía el mismo patrón, después de un XOR, R1 será todo ceros.

Dado que tanto las entradas como la salida son ocho cables, simplificaremos usando nuestra imagen de tipo



bus.

La víbora

Esta es una combinación de puertas que es sorprendentemente simple considerando lo que hace. En nuestro sistema numérico binario, tenemos números en el rango de 0 a 255 representados en un byte. Si piensas en cómo se hace la suma con dos de nuestros números decimales regulares, comienzas sumando los dos números en la columna de la derecha.

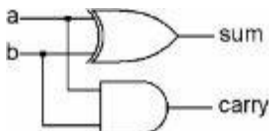
Dado que los dos números pueden estar entre 0 y 9, la suma de estos dos estará entre 0 y 18. Si la respuesta está entre 0 y 9, escríbela debajo de los dos números. Si la respuesta es del 10 al 18, escribe el dígito de la derecha y lleva el 1 para agregar a la siguiente columna a la izquierda.

$$\begin{array}{r} 2 \qquad \qquad 5 \\ +4 \qquad \qquad +7 \\ \hline 6 \qquad \qquad 12 \end{array}$$

En el sistema numérico binario, en realidad es mucho más simple. Si realiza el mismo tipo de suma en binario, los dos números en la columna de la derecha solo pueden ser 0 o 1. Por lo tanto, las únicas respuestas posibles para sumar la columna derecha de dos números binarios serán 0, 1 o 10 (cero, uno o dos). Si sumas 0+0, obtienes 0, 1+0 o 0+1 obtienes 1, 1+1 obtienes 0 en la columna de la derecha y llevas 1 a la columna de la izquierda.

$$\begin{array}{r} 1 \qquad \qquad 1 \\ +0 \qquad \qquad +1 \\ \hline 1 \qquad \qquad 10 \end{array}$$

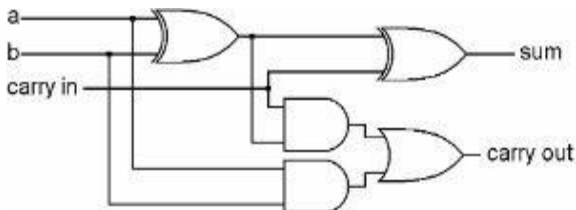
Las puertas que hemos descrito pueden hacer esto fácilmente. Un XOR de los dos bits nos dirá cuál debería ser la respuesta de la columna derecha, y un AND de los dos bits nos dirá si necesitamos llevar un 1 a la columna de la izquierda. Si un bit está encendido y el otro está apagado, es decir, estamos agregando un 1 y un 0, la respuesta para la columna de la derecha será 1. Si ambos números son 1, o ambos números son 0, la columna de la derecha será 0. La puerta AND se activa solo en el caso de que ambos números de entrada sean 1.



Así que hemos agregado la columna derecha fácilmente. Ahora queremos agregar la siguiente columna a la izquierda. Debería ser lo mismo, ¿verdad? Hay dos bits que podrían ser 0 o 1, pero esta vez también tenemos la posibilidad de un carry de la columna anterior. Así que no es lo mismo, esta vez realmente estamos sumando tres números, los dos bits de esta columna, más el posible acarreo de la columna anterior.

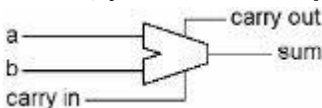
Llevar >	0	1	0	1 1
	00	01	10	011
	<u>+01</u>	<u>+01</u>	<u>+01</u>	<u>+011</u>
	01	10	11	110

Al sumar tres bits, las respuestas posibles son 0, 1, 10 u 11 (cero, uno, dos o tres). Es más complejo, pero no imposible. Aquí está la combinación que lo hace:



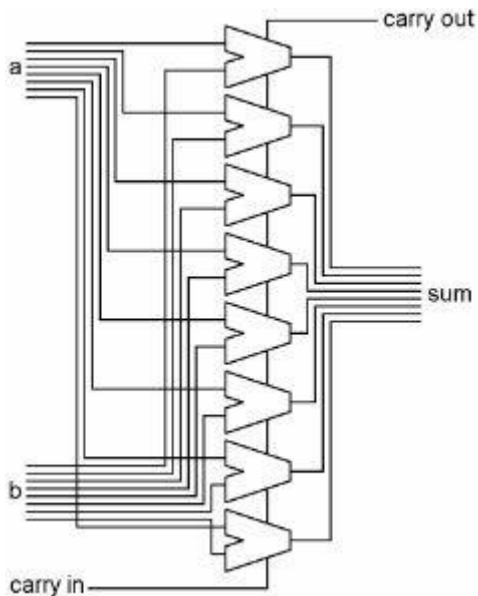
El XOR izquierdo nos dice si 'a' y 'b' son diferentes. Si lo están, y 'carry in' está desactivado, o si 'a' y 'b' son lo mismo y 'carry in' está activado, entonces el XOR derecho generará 1 como la suma de la columna actual. La puerta AND inferior se activará en 'llevar a cabo' si ambas entradas están activadas.

La puerta AND superior se activará si "llevar a cabo" y una de las entradas están activadas. Este es el ;Simplicidad de cómo las computadoras hacen sumas! Ahora que vemos que funciona, podemos hacer una imagen más

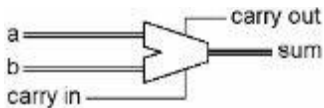


simple de ello:

Para hacer un sumador que sume dos bytes, necesitamos un sumador de un bit para cada bit de los bytes, con la salida de acarreo de cada bit conectada a la entrada de acarreo del siguiente. Observe que cada bit tiene un acarreo, incluso el primer bit (la columna de la derecha). Esto se usa cuando queremos sumar números que pueden ser mayores que 255.



Y la imagen simplificada con entradas y salidas de bus:



El bit de salida de acarreo de la columna más a la izquierda (superior) se encenderá si la suma de los dos números es mayor que 255, y este bit se utilizará en otra parte de la computadora.

Así es como las computadoras hacen sumas, solo cinco puertas por bit, ¡y la computadora puede hacer aritmética!

El comparador y el cero

Todas las cosas que hemos descrito anteriormente toman uno o dos bytes como entrada y generan un byte de salida. Los cambiadores y el sumador también generan un bit adicional de salida que está relacionado con su byte de salida. El comparador solo genera dos bits de salida, no un byte completo.

El comparador está integrado en el XORer porque puede hacer uso de las puertas que ya están allí. El XORer genera su byte de salida y el comparador genera sus dos bits. Estas dos funciones no están realmente relacionadas entre sí, simplemente resulta fácil construirlo de esta manera.

Lo que queremos que haga el comparador es averiguar si los dos bytes en el bus de entrada son exactamente iguales y, si no, si el del bus 'a' es más grande de acuerdo con el sistema numérico binario.

Lo igual es bastante simple. Las puertas XOR se apagan cuando las entradas son las mismas, por lo que si todas las puertas XOR están apagadas, entonces las entradas son iguales.

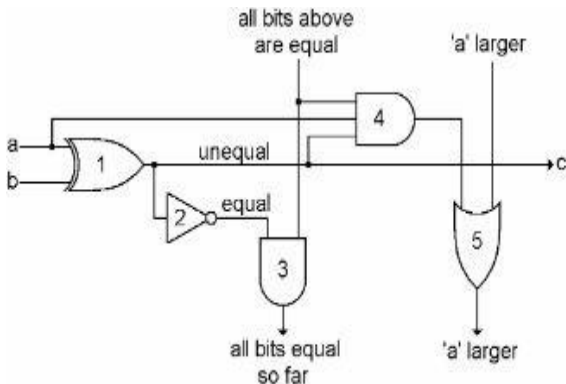
Determinar el mayor de dos números binarios es un poco más complicado. Tienes que comenzar con los dos bits superiores, y si uno está encendido y el otro está apagado, entonces el que está encendido es el número más grande. Si son iguales, entonces tienes que comprobar el siguiente par inferior de bits, etc., hasta que encuentres un par en el que sean diferentes. Pero una vez que encuentre un par que sea diferente, no querrá verificar más partes. Por ejemplo 0010 0000 (32) es mayor que 0001 1111 (31.) Los dos primeros bits son iguales en ambos bytes. El tercer bit está encendido en el primer byte y apagado en el segundo, y por lo tanto el primer byte es más grande. Aunque el resto de los bits están encendidos en el segundo byte, su total es menor que el bit que está encendido en el primer byte.

Eso es lo que queremos que suceda, y se necesitan cinco puertas por ocho posiciones para lograrlo. Dado que estamos comenzando con el XORer, agregaremos cuatro puertas más a cada posición como se muestra en este

diagrama. Recuerde que en el sumador, teníamos una broca de transporte que pasaba desde la posición más baja de la broca hasta la más alta. En

El comparador, tenemos dos bits que pasan de la posición de bits más alta a la más baja.

Aquí hay un poco del comparador. Puede ver la puerta XOR original, etiquetada como '1', conectada a un bit de cada bus de entrada a la izquierda y generando un bit para el bus de salida a la derecha.



Si la salida de la puerta 1 está activada, eso significa que 'a' y 'b' son diferentes o desiguales. Agregamos la puerta 2, que se encenderá cuando 'a' y 'b' sean iguales.

Si la puerta 2 está activada en todas las posiciones, entonces la puerta 3 también estará activada en todas las posiciones, y el bit que sale de la parte inferior nos dice que los dos bytes de entrada son iguales.

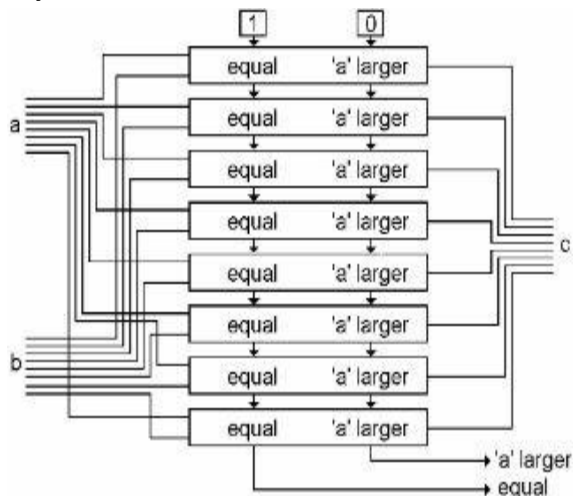
La puerta 4 se enciende si tres cosas son ciertas. 1) Los bits 'a' y 'b' son diferentes. 2) El bit 'a' es el que está encendido. 3) Todos los bits por encima de este punto han sido iguales. Cuando la puerta 4 se enciende, enciende la puerta 5, y eso enciende

todas las demás puertas 5 por debajo de este punto, y
por lo tanto el 'a

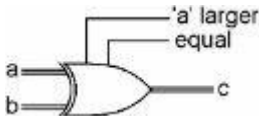
mayor salida del comparador.

Cuando el byte 'b' es el más grande, tanto el bit 'igual' como el bit 'a larger' estarán apagados.

Apila ocho de estos comparadores de bits como en el siguiente diagrama, con un '1' y un '0' conectados al superior para comenzar. Todavía tiene la función XOR que sale en 'c', y ahora los dos bits de comparación en la parte inferior.

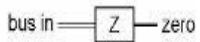
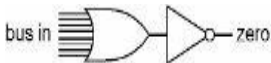


Simplificando de nuevo, volveremos al diagrama XOR de tipo bus, y sólo añadiremos los dos nuevos bits de salida del comparador.



Hay una cosa más que vamos a necesitar en nuestra computadora que nos da otro poco de información. Esta es una combinación de puerta simple que toma un byte completo como entrada y genera solo un bit como salida. El bit de salida se enciende cuando todos los bits del byte están apagados.

En otras palabras, el bit de salida nos dice cuándo el contenido del byte es todo ceros.



Es simplemente una puerta OR de ocho entradas y una puerta NOT. Cuando cualquiera de las entradas a la puerta OR está activada, su salida estará encendida y la salida de la puerta NOT estará apagada. Solo cuando las ocho entradas del OR estén apagadas y, por lo tanto, su salida esté apagada, la salida de la puerta NOT estará encendida. La representación de bus más simple se muestra a la derecha.

Lógica

El tema del pensamiento ha sido objeto de mucho estudio y especulación a través de los siglos. Había un hombre en la antigua Grecia llamado Aristóteles que trabajaba mucho en esta área. Debe haber conocido a mucha gente ilógica en su vida, porque inventó un tema que se suponía que ayudaría a la gente a pensar con más sensatez.

Una de sus ideas era que si tienes dos hechos, puedes derivar un tercer hecho de los dos primeros. En la escuela a veces dan pruebas que presentan dos hechos, luego te dan un tercer hecho y te preguntan si el tercer hecho es "lógico" basado en los dos primeros. Es posible que recuerde preguntas como:

Si Joe es mayor que Bill, Y

Fred es mayor que Joe,

Entonces Fred es mayor que Bill. ¿Verdadero o falso?

O

A los niños les
gustan los dulces.

Jane es una niña.

Por lo tanto, a Jane le gustan los dulces. ¿Verdadero o falso?

Aristóteles llamó a su estudio de este tipo de cosas "Lógica".

La única relevancia que esto tiene para nuestra discusión sobre las computadoras es esta palabra 'lógica'. La lógica de Aristóteles involucraba dos hechos que formaban un tercer hecho. Muchas de nuestras partes de computadora, como las puertas AND, toman dos bits y forman un tercer bit, u ocho puertas AND toman dos bytes y forman un tercer byte. Y así, las cosas que hacen estas puertas, han llegado a conocerse como lógica. Puede haber lógica AND y lógica OR y lógica XOR, pero el término general para todas ellas es lógica.

ANDing, ORing y XORing toman dos bytes para formar un tercero, por lo que se ajustan bastante bien a esta definición de lógica.

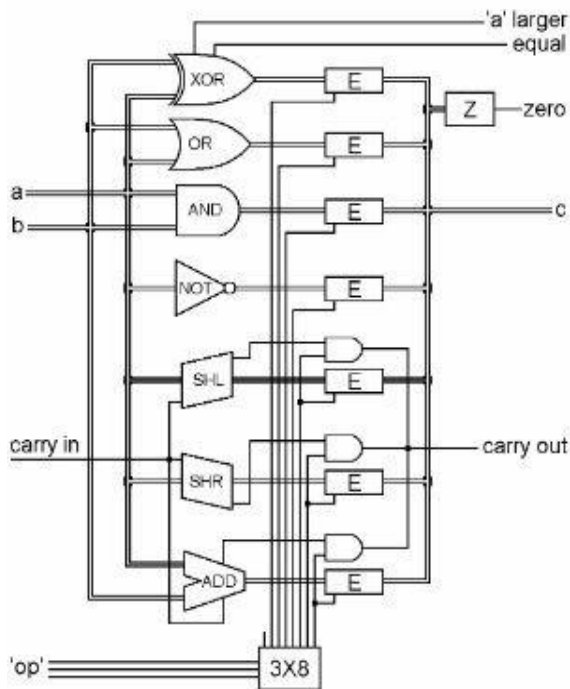
El desplazamiento y el NOTing también se conocen como lógica, aunque solo requieren un byte de entrada para generar su salida. El ADDer, aunque tiene dos entradas y

también es muy lógico, de alguna manera no se sabe que esté en la categoría de lógica, está en su propia categoría, aritmética.

Entonces, todas las formas que hemos descrito anteriormente de hacer cosas con bytes caen bajo el título de 'aritmética y lógica'.

La unidad aritmética y lógica

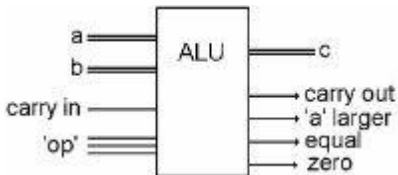
Ahora hemos construido siete dispositivos diferentes que pueden hacer aritmética o lógica en bytes de datos. Vamos a tomar estos siete dispositivos, juntarlos en una unidad y proporcionar un método para seleccionar cuál de estos dispositivos queremos usar en un momento dado. Esto se llama "Unidad Aritmética y Lógica" o "ALU" para abreviar.



Los siete dispositivos están conectados a la entrada 'a', los dispositivos que tienen dos entradas también están conectados a la entrada 'b'. Los siete dispositivos están conectados a las entradas en todo momento, pero cada salida está conectada a uno de esos habilitadores. Los cables que encienden los habilitadores están conectados a las salidas de un decodificador, por lo que solo un habilitador puede estar encendido en un momento dado. Siete de las salidas del decodificador permiten que un solo dispositivo continúe con la salida común, 'c'. La octava salida del decodificador se usa cuando no desea seleccionar ningún dispositivo en absoluto. Los tres cables de entrada al decodificador están etiquetados como 'op', porque eligen la 'operación' deseada.

La única pequeña complicación aquí son los bits de transporte de la víbora, y los bits de 'cambio hacia adentro' y 'cambio hacia afuera' de los cambios de marcha. Estos se usan de manera muy similar, por lo que de aquí en adelante nos referiremos a todos ellos como bits de transporte. El sumador y ambos cambiadores toman el carry como entrada y generan el carry como salida. Por lo tanto, las tres entradas de transporte están conectadas a una sola entrada ALU, y una de las tres salidas se selecciona junto con la salida de bus de su dispositivo. Mire la salida más a la derecha del decodificador 3X8 de arriba y vea que habilita tanto el bus sumador como el bit de transporte sumador.

¿Qué tenemos aquí? Es una caja que tiene dos entradas de bus, una salida de bus y otros cuatro bits de entrada y otros cuatro bits de salida. Tres de los bits de entrada seleccionan qué "operación" tendrá lugar entre los buses de entrada y salida. Nuevamente, ahora que sabemos qué contiene y cómo funciona, no necesitamos mostrar todas sus partes. Aquí hay una forma simplificada de



dibujarlo:

Observe que las tres entradas de un solo bit etiquetadas como "op",

anterior, puede tener ocho combinaciones diferentes. Siete de esas combinaciones seleccionan uno de los dispositivos descritos anteriormente. La octava combinación no selecciona ningún byte de salida, pero los bits 'a larger' e 'equal' siguen funcionando, como lo hacen en todo momento, por lo que este es el código a elegir si solo quieres hacer una comparación.

La combinación de bits en 'op' significa algo. Eso suena como otro código. Sí, aquí hay un código de tres bits que usaremos pronto.

000	AGREGAR	Agregar
001	SHR	Desplazar a la derecha
010	SHL	Desplazar a la izquierda
011	NO	No
100	Y	Y
101	O	O
110	XOR	Exclusivo O
111	CMP	Comparar

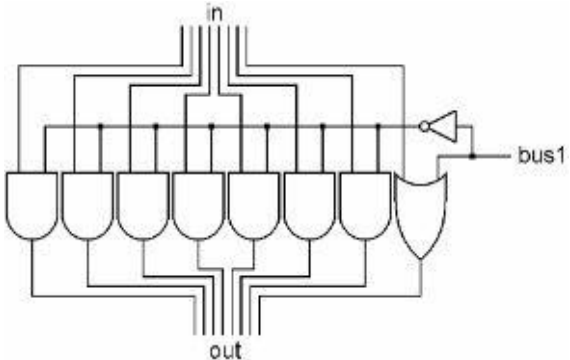
La Unidad de Aritmética y Lógica es el centro mismo, el corazón de la computadora. Aquí es donde ocurre toda la acción. Apuesto a que esto es mucho menos complicado de lo que pensabas.

Más del procesador

Hay un pequeño dispositivo más que necesitamos. Es algo muy simple, tiene una entrada de bus, una salida de bus y otro bit de entrada. Es muy similar a un habilitador. Siete de los bits pasan por puertas Y, y uno de ellos pasa por una puerta O. La entrada de un solo bit determina lo que sucede cuando un byte intenta pasar a través de este dispositivo.

Cuando el bit 'bus 1' está apagado, todos los bits del bus de entrada pasan al bus de salida sin cambios.

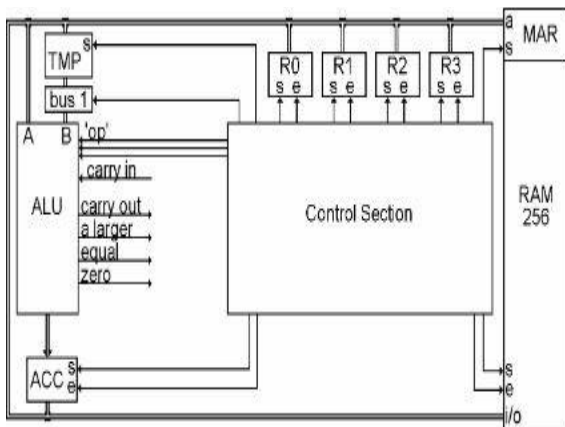
Cuando el bit 'bus 1' está encendido, el byte de entrada se ignora y el byte de salida será 0000 0001, que es el número 1 en binario. Llamaremos a este dispositivo un 'bus 1' porque colocará el número 1 en un bus cuando lo



necesitemos.

Ahora podemos poner este 'bus 1' y la ALU en la CPU. Cambiaremos dónde entran y salen los cables de la ALU para que se ajuste mejor a nuestro diagrama. Las entradas de bus vienen en la parte superior, la salida de bus sale en

la parte inferior y todos los bits de entrada y salida están a la derecha.



La salida de la ALU está conectada a ACC. ACC recibe y almacena temporalmente el resultado de la operación ALU más reciente. La salida de ACC se conecta al bus, por lo que su contenido se puede enviar a otro lugar según sea necesario.

Cuando queremos hacer una operación ALU de una entrada, tenemos que configurar los tres bits 'op' de la ALU en la operación deseada, habilitar el registro que queremos en el bus y establecer la respuesta en ACC.

Para una operación ALU de dos entradas, hay dos pasos. Primero habilitamos uno de los registros en el bus y lo configuramos en TMP. Luego habilitamos el segundo registro en el bus, elegimos la operación ALU y establecemos la respuesta en ACC.

Como puede ver, ahora podemos mover bytes de datos a y

desde la RAM, hacia y desde los registros, pasando por la ALU hasta el ACC, y desde allí, a un registro o RAM si activamos y desactivamos los bits de habilitación y configuración apropiados en el momento adecuado. Esto es lo que sucede dentro de las computadoras.

Eso no es tan complicado, ¿verdad?

Solo falta una cosa aquí, y tiene que ver con todos estos bits de control en los registros, ALU y RAM. La RAM tiene tres bits de control, uno para establecer MAR, otro para establecer el byte seleccionado actualmente, uno para habilitar la salida del byte seleccionado actualmente. Cada uno de los registros, R0, R1, R2, R3 y ACC tienen un bit establecido y un bit de habilitación, TMP solo tiene un bit establecido, el bus 1 tiene un bit de control y el ALU tiene esos tres bits 'op' que seleccionan la operación deseada.

Necesitamos algo que encienda y apague todos estos bits de control en los momentos apropiados para que podamos hacer algo que sea útil. Ahora es el momento de mirar en ese cuadro etiquetado como 'Sección de control'.

El reloj

Necesitamos encender y apagar los bits de control apropiados en los momentos apropiados. Veremos las partes apropiadas más adelante, primero veremos los momentos apropiados.

Aquí hay un nuevo tipo de dibujo, lo llamaremos gráfico. Muestra cómo cambia un bit con el tiempo. El tiempo comienza a la izquierda y avanza hacia la derecha. La línea en el gráfico tiene dos posiciones posibles, arriba significa que el bit está encendido y abajo significa que



el bit está apagado.

Este gráfico muestra el bit 'X' que se enciende y se apaga, se enciende y se apaga regularmente. Podría haber una escala de tiempo en la parte inferior para mostrar qué tan rápido está sucediendo esto. Si todo el ancho de la página representara un segundo, entonces el bit 'X' se encendería y apagaría unas ocho veces por segundo. Pero no necesitaremos una escala de tiempo en estos gráficos, ya que solo nos preocuparemos por el tiempo relativo entre dos o más bits. La velocidad en una computadora real será muy rápida, como el bit que se enciende y apaga mil millones de veces por segundo.

Cuando algo repite alguna acción regularmente, una de esas acciones, individualmente, se llama ciclo. El gráfico anterior muestra unos ocho ciclos. Puede decir que desde un momento en que el bit se enciende hasta la siguiente vez que el bit se enciende es un ciclo, o puede decir que desde la mitad del tiempo de apagado hasta la mitad del siguiente tiempo de apagado es el ciclo, siempre que el ciclo comience en un punto en el tiempo cuando el bit se encuentra en alguna etapa de su actividad, y continúa hasta que el bit vuelve a la misma etapa de la actividad. La palabra 'Ciclo' proviene de la palabra 'círculo', por lo que cuando el bit cierra el círculo, eso es un ciclo.

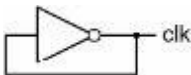
Hubo un científico que vivió en Alemania en la década de

1800 que hizo algunas de las primeras investigaciones que condujeron a la radio. Su nombre era Heinrich Hertz, y entre otras cosas,

estudió la electricidad que se encendía y apagaba muy rápidamente. Algunas décadas después de su muerte, se decidió usar su nombre para describir qué tan rápido se encendía y apagaba la electricidad, o cuántos ciclos ocurrían por segundo.

Por lo tanto, un hercio (o Hz para abreviar) significa que la electricidad se enciende y se apaga una vez por segundo. 500 Hz significa 500 veces por segundo. Para velocidades más rápidas, nos encontramos con esos idiomas antiguos nuevamente, y mil veces por segundo se llama kilohercios o kHz para abreviar. Encender y apagar un millón de veces por segundo se llama megahercio, o MHz para abreviar, y mil millones de veces se llama gigahercios, o GHz para abreviar.

Cada computadora tiene una parte especial. Todos los demás bits de una computadora provienen de algún lugar, se encienden y apagan mediante otros bits o interruptores. Esta broca especial se enciende y apaga por sí sola. Pero no hay nada misterioso en ello, simplemente se enciende y se apaga con mucha regularidad y muy rápidamente. Esta broca especial está construida



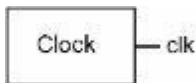
de manera muy simple, así:

Esto parece una tontería. ¿Simplemente conecte la salida de una puerta NOT a su entrada? ¿Qué hará esto? Bueno, si comienzas con la salida encendida, la electricidad viaja de regreso a la entrada, donde ingresa a la puerta que apaga la salida, que viaja de regreso a la entrada que enciende la salida. Sí, esta puerta se encenderá y apagará lo más rápido posible. En realidad, esto será demasiado rápido para usarse para cualquier cosa, por lo que se puede ralentizar simplemente alargando el cable que hace el bucle.



El diagrama simplificado muestra que este es el único

bit especial en la computadora que tiene una salida pero no tiene ninguna entrada.



Esta parte tiene un nombre. Se llama reloj. Ahora bien, solemos pensar en un reloj como una cosa con un dial y manecillas, o algunos números en una pantalla, y hemos visto tales relojes en la esquina de la pantalla de una computadora.

Desafortunadamente, alguien nombró este tipo de bit, un reloj, y el nombre se quedó con los pioneros de la computadora. Podría haberse llamado el ritmo del tambor o el marcador de ritmo o el corazón o la sección rítmica, pero lo llamaron reloj. Eso es lo que queremos decir cuando digamos reloj a lo largo del resto de este libro. Supongo que es un reloj que hace tictac, pero no tiene dial. Si queremos hablar sobre el tipo de reloj que le dice qué hora es, lo llamaremos 'reloj de la hora del día' o 'reloj TOD' para abreviar. Pero la palabra 'reloj' significará este tipo de broca.

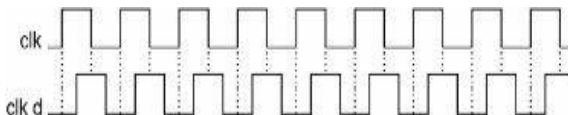
¿Qué tan rápido se enciende y apaga este reloj? En estos días es más de mil millones de veces por segundo, o varios gigahercios. Esta es una de las principales características que te comentan las empresas informáticas para mostrarte lo geniales que son sus ordenadores. Cuando ves computadoras a la venta, la velocidad que anuncian es la velocidad de su reloj. Cuanto más rápida es una computadora, más cara es, porque puede hacer más cosas en un segundo. Es la velocidad de este único bit que se enciende y apaga lo que establece el ritmo de toda la computadora.

Para mover datos a través del bus, primero necesitamos habilitar la salida de uno y solo un registro, de modo que su electricidad pueda viajar a través del bus a las entradas de otros registros. Luego, mientras los datos están en el bus, queremos activar y desactivar el bit establecido del registro de destino. Dado que el registro de destino captura el estado del bus en el instante en que se apaga el bit establecido, queremos asegurarnos de que se apague antes de apagar el bit de habilitación en el primer registro para asegurarnos de que no haya problemas.

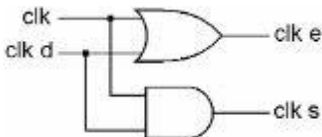
Primero conectemos un trozo de cable a la salida del reloj. Esto retrasará ligeramente la electricidad. Queremos que se retrase alrededor de un cuarto de ciclo.



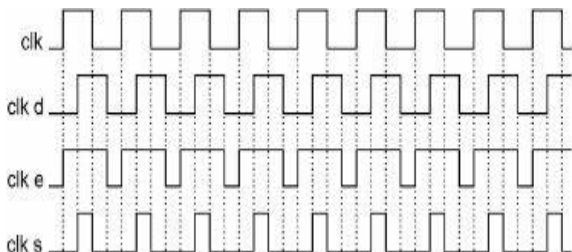
Si mostramos la salida de reloj original (clk) y la salida de reloj retardado (clk d) en un gráfico, se verán así:



Ahora vamos a hacer algo bastante simple. Tomaremos el reloj original y el reloj retrasado, y ambos AND them y OR them para crear dos nuevos bits, así:

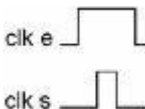


Uno de los nuevos bits está activado cuando 'clk' o 'clk d' están activados, y el otro nuevo bit está activado solo cuando tanto 'clk' como 'clk d' están activados. Aquí se muestra el gráfico de las entradas y salidas de las puertas AND y OR. Ambos todavía se encienden y apagan regularmente, pero uno de ellos está encendido por más tiempo que apagado, y el otro está apagado por más tiempo del que está encendido. La hora de encendido del segundo está justo en el medio de la hora de encendido del primero.



Observe que tienen nombres, 'clk e', que significa habilitación de reloj, y 'clk s', que significa conjunto de reloj. Y qué sabes, estos dos bits tienen el momento perfecto para mover un byte de datos de un registro, a través del bus y a otro registro. Simplemente conecte 'clk e' al bit de habilitación del registro 'from' y conecte 'clk s' al bit de conjunto de registro 'to'.

Aquí hay un solo ciclo de encendido / apagado de estos dos



bits.

Si observa el tiempo aquí, esto cumple con nuestros requisitos de necesidad de habilitar primero la salida de un registro y luego, después de que los datos tengan un poco de tiempo para viajar por el bus, encender y apagar el bit establecido del registro de destino antes de apagar el bit de habilitación en el primer registro.

Por supuesto, estos bits de reloj no se pueden conectar directamente a cada registro. Debe haber otras puertas intermedias, que solo permitan que un registro obtenga una habilitación en un momento dado, y solo los registros deseados reciban un conjunto. Pero todas las

habilitaciones y conjuntos en última instancia provienen de estas dos partes porque tienen el momento adecuado.

Dado que usaremos `clk`, `clk e` y `clk s` en toda la computadora, este es el diagrama que usaremos para mostrar el reloj:



Hacer algo útil

Digamos que queremos hacer algo útil, como sumar un número a otro número. Tenemos un número en R0, y hay otro número en R1 que queremos agregar al número en R0. El procesador que hemos construido hasta ahora tiene todas las conexiones para hacer esta adición, pero tomará más de un ciclo de reloj para hacerlo.

En el primer ciclo de reloj, podemos habilitar R1 en el bus y configurarlo en TMP.

En el segundo ciclo podemos habilitar R0 en el bus, configurar el ALU en ADD y establecer la respuesta en ACC.

En el tercer ciclo, podemos habilitar ACC en el bus y configurarlo en R0.

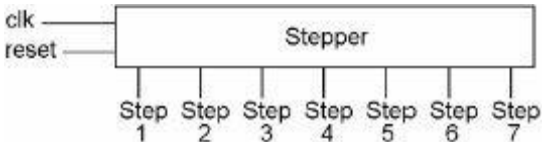
Ahora tenemos el valor anterior de R0, más R1 en R0. Quizás esto no parezca muy útil, pero es uno de los pequeños pasos que dan las computadoras. Muchos de estos pequeños pasos hacen que la computadora parezca capaz de hacer cosas muy complejas.

Así vemos que para que el procesador haga algo útil, se necesitan varios pasos. Necesita poder realizar acciones en una secuencia. Necesitamos otra pieza dentro de esta 'Sección de Control'.

Paso a paso

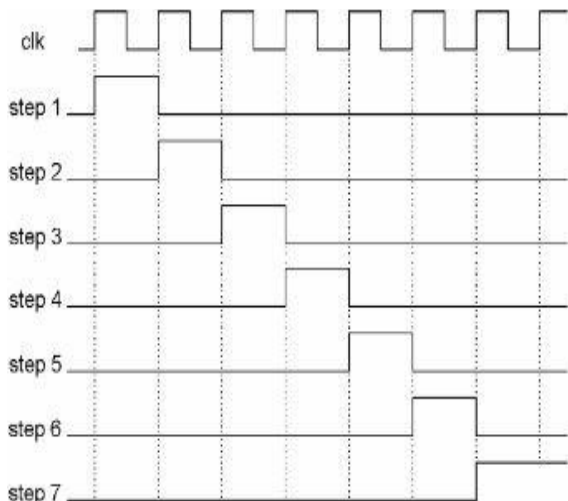
Este capítulo presenta una nueva parte llamada "Stepper". Primero, describiremos el paso a paso completo, mostrando exactamente lo que hace. Después de eso, veremos exactamente cómo se construye. Si confía en su autor lo suficiente como para creer que tal paso a paso se puede construir desde las puertas, y tiene tanta prisa que desea omitir la parte del capítulo de "cómo se construye", es posible que aún comprenda la computadora.

Aquí hay un paso completo.



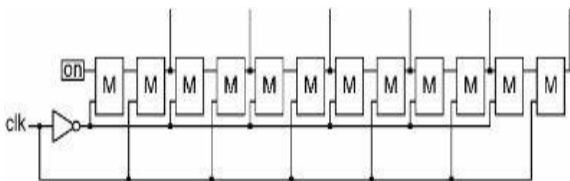
Tiene dos entradas. Uno se llama 'clk', porque aquí es donde conectamos un bit que se enciende y se apaga, como nuestro bit de reloj original. La otra entrada se llama 'reset', que se utiliza para devolver el paso a paso al paso uno. Para las salidas, tiene una serie de bits, cada uno de los cuales se encenderá durante un ciclo de reloj completo y luego se apagará, uno tras otro. La salida etiquetada como 'Paso 1' se enciende durante un ciclo de reloj, luego 'Paso 2' para el siguiente ciclo de reloj, etc. Se puede construir un paso a paso para que tenga tantos pasos como sea necesario para cualquier tarea en particular que desee realizar. En el caso de esta computadora que estamos construyendo, siete pasos son suficientes. Cuando el último paso (7) se enciende, permanece encendido y el paso a paso no hace nada más hasta que el bit de reinicio se enciende brevemente, momento en el que los pasos comienzan de nuevo comenzando con el 'Paso 1'.

Aquí hay un gráfico del bit 'clk' de entrada y las salidas de un paso a paso de siete pasos.



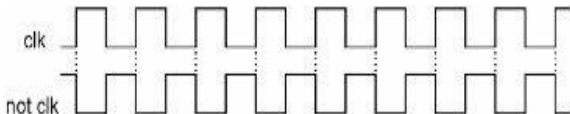
Así es como se construye el paso a paso. Se hace usando algunos de los mismos bits de memoria que usamos para hacer registros, pero están dispuestos de manera muy diferente. No vamos a almacenar nada en estos bits, los vamos a usar para crear una serie de pasos.

El paso a paso consta de varios bits de memoria conectados en una cadena, con la salida de uno conectada a la entrada del siguiente. Aquí hay un diagrama que muestra la mayor parte del paso a paso:



Primero mire la serie de bits de memoria 'M' como los que usamos anteriormente en el libro. En esta imagen, hay doce de ellos conectados entre sí, con la salida de uno conectado a la entrada del siguiente, hasta el final de la línea. La entrada al primer bit de la izquierda está conectada a un lugar donde la electricidad siempre está encendida, por lo que cuando se enciende el bit de ajuste de esa 'M', esa 'M' recibirá ese estado encendido y lo pasará a su salida.

Si observa los bits de conjunto de estas 'M', verá que los bits de conjunto de las 'M' pares están conectados a `clk`, y los bits de conjunto de las 'M' impares están conectados al mismo reloj después de que pasa por una puerta NOT. Este nuevo bit que se hace al pasar `clk` a través de una puerta NOT se puede llamar 'not clk', y podemos mostrar ambos en este gráfico:



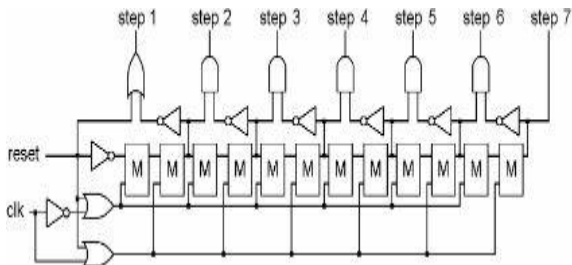
Entonces, ¿qué pasará con este montón de puertas? Si asume que todas las 'M' comienzan en el estado apagado y luego comienzan a 'clk' "marcar", esto es lo que hará.

La primera vez que se enciende 'clk', no pasará nada, porque el bit establecido de la primera 'M' está conectado a 'not clk', que está apagado cuando 'clk' está encendido. Cuando suena 'clk', 'not clk' se enciende y se encenderá la primera 'M', pero no pasará nada en la segunda 'M'

porque su bit 'set' está conectado a 'clk', que ahora está apagado. Cuando 'clk' vuelva a encenderse, la segunda 'M' ahora se encenderá. A medida que el reloj avanza, el "encendido" que ingresa al primer bit de memoria bajará por la línea, un bit por cada vez que el reloj se enciende y un bit por cada vez que el reloj se apague. Por lo tanto, se encienden dos bits para cada ciclo de reloj.

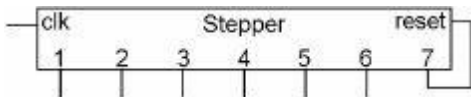
Ahora, pasando al diagrama paso completo a continuación, el paso 1 proviene de una puerta NOT conectada a la salida de la segunda 'M'. Dado que todas las 'M' comienzan, el paso 1 estará encendido hasta que se encienda la segunda 'M', momento en el que el paso 1 habrá terminado. Para los pasos restantes, cada uno durará desde el momento en que se enciende su lado izquierdo 'M' hasta el momento en que se enciende su lado derecho 'M'. Las puertas AND para los pasos 2-6 tienen ambas entradas activadas cuando la 'M' izquierda está encendida y la 'M' derecha está apagada. Si conectamos la salida de una 'M' y la NOT de la salida de una 'M' dos espacios más adelante a una puerta AND, su salida estará encendida durante un ciclo de reloj completo. Cada uno se enciende cuando se ha encendido su entrada izquierda, pero su entrada derecha aún no se ha encendido. Esto nos da una serie de bits que se encienden durante un ciclo de reloj y luego se apagan.

Lo único que falta aquí es que los bits 'M' se encienden y se quedan. Una vez que están todos encendidos, no hay más acción a pesar del continuo tictac del reloj. Así que necesitamos una forma de reiniciarlos todos para poder empezar de nuevo. Tenemos que tener una forma de apagar la entrada a la primera 'M' y luego encender todos los bits establecidos al mismo tiempo. Cuando eso sucede, el 'apagado' en la entrada de la primera 'M' viajará a través de todas las 'M' lo más rápido posible. Agregaremos una nueva entrada llamada 'reset', que logrará estas cosas.



Cuando activamos 'reset', hace que la entrada al primer bit 'M' sea un cero, y enciende todos los 'sets' al mismo tiempo para que el cero pueda viajar por la línea de 'M' muy rápidamente. El restablecimiento también se OR con el paso 1 para que el paso 1 se encienda inmediatamente. Ahora todos los bits están apagados y hemos comenzado otra secuencia. El restablecimiento solo debe activarse durante una fracción de un ciclo de reloj.

Para recapitular, este es un paso a paso. Tiene dos entradas: un reloj y un reinicio. Para las salidas, tiene una serie de bits, cada uno de los cuales se encenderá durante un ciclo de reloj. De hecho, podemos hacer esto tan largo como sea necesario, pero para los propósitos de este libro, un paso de siete etapas será suficiente. Solo habrá un paso en nuestra computadora, lo representaremos con este diagrama simplificado.



Hemos reubicado el bit de reinicio en el lado derecho del diagrama y lo hemos conectado al último paso (7) para que el paso a paso se reinicie automáticamente. Sin embargo, el paso 7 no estará encendido por mucho tiempo

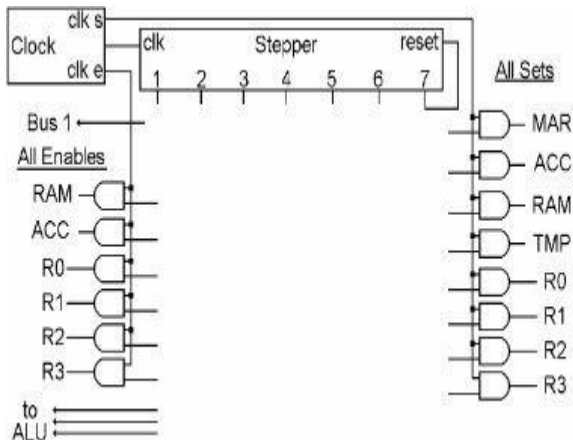
porque se cierra

tan pronto como el cero pueda atravesar la cadena de 'M'. Esto significa que el paso 7 no durará lo suficiente como para ser utilizado para una de nuestras transferencias de datos a través del bus. Todas las cosas que queremos lograr se llevarán a cabo en los pasos 1 al 6.

Todo está bajo control

Con nuestro reloj, tenemos un ritmo de tambor para hacer que las cosas funcionen. Cuenta con una salida básica, y dos más que están pensadas para facilitar el movimiento de los contenidos de los registros de uno a otro. Con el paso a paso, tenemos una serie de bits que se encienden uno tras otro, cada uno para un ciclo de reloj.

¿Recuerdas el diagrama de la CPU que vimos hace unos capítulos? Mostró el bus, la ALU, seis registros e incluso la otra mitad de la computadora (la RAM) conectados bastante bien. Al menos todas las conexiones de autobús estaban allí. Pero todos los registros, la RAM, el Bus 1 y el ALU están controlados por cables que provienen de esa misteriosa caja etiquetada como 'Sección de control' de la que aún no sabemos nada. Ahora es el momento de mirar dentro de esa caja.



Este dibujo es el comienzo de la sección de control de la computadora. En la parte superior están el reloj y el stepper. Luego, todos los bits de control de los registros y la RAM se han reunido aquí en un solo lugar, con todos los bits de 'habilitación' a la izquierda y todos los bits de 'conjunto' a la derecha. Luego hemos conectado la salida de una puerta AND a cada bit 'enable' y cada 'set'. Una entrada de cada puerta AND está conectada a 'clk e' para los 'habilita' a la izquierda, o 'clk s' para los 'conjuntos' a la derecha. Por lo tanto, si usamos la otra entrada de esas puertas AND para seleccionar cualquiera de esos registros, el bit 'enable' de todos los elementos de la izquierda nunca se encenderá excepto durante el tiempo 'clk e'. De manera similar, a la derecha, el bit 'set' de cualquiera de esos registros solo se activará durante el tiempo 'clk s'.

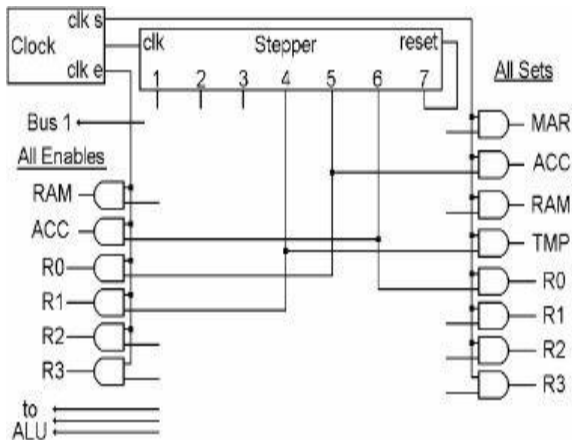
Esta es una especie de centralita. Todo lo que necesitamos

para

hacer que la computadora haga algo está aquí en un solo lugar. Todo lo que tenemos que hacer es conectar algunos bits de control a algunos pasos de manera inteligente, y sucederá algo útil.

Hacer algo útil, revisado

Ahora que tenemos el comienzo de nuestra sección de control, podemos agregar algunos cables y podremos hacer la adición simple que postulamos anteriormente, la de agregar R1 a R0.



Todo lo que tenemos que hacer para 'hacer algo útil', como agregar R1 a R0, es conectar algunos cables en el medio, como se muestra en este diagrama con los pasos cuatro, cinco y seis.

Cada paso hace que algo suceda con algunas de las partes que se muestran en el diagrama de la CPU. Cada paso está conectado a un 'enable' a la izquierda y un 'set' a la derecha y, por lo tanto, hace que una parte conecte su salida al bus y otra parte guarde lo que ahora aparece en su entrada. El paso cuatro está conectado a R1

'enable'

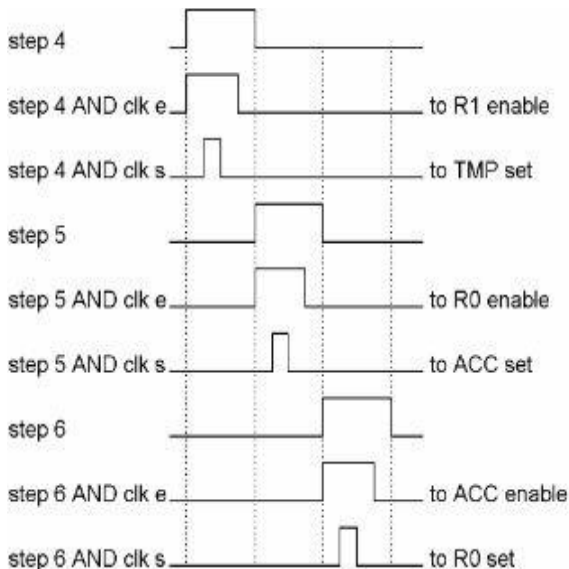
y TMP 'establecido'. El paso cinco está conectado a R0 'habilitar' y ACC 'establecer'. Los bits 'op' de ALU no necesitan ninguna conexión ya que el código 'op' para ADD es 000. El paso seis está conectado a ACC 'enable' y R0 'set'.

Durante el paso cuatro, R1 está habilitado y TMP está configurado. El contenido de R1 viaja a través del bus (en el diagrama de CPU) y TMP lo captura.

Durante el paso cinco, R0 está habilitado y ACC está configurado. Si quisiéramos hacer algo que no fuera ADD, este es el paso en el que activaríamos los bits de código 'op' ALU apropiados.

Durante el paso seis, ACC está habilitado y R0 está configurado.

Aquí hay un gráfico de los pasos, que muestra cuándo se habilita y configura cada registro.



R0 ahora contiene la suma del contenido original de R0 más R1.

Así es como la computadora hace que las cosas sucedan en un ballet estrictamente controlado de bits y bytes que se mueven dentro de la máquina.

En el paso siete, el paso a paso se restablece al paso 1, donde se repite el proceso. Por supuesto, no es muy útil simplemente

haga esta adición una y otra vez, incluso si comienza con el número 1 tanto en R0 como en R1, R0 llegará a 255 con bastante rapidez.

Si el reloj de nuestra computadora marca mil millones de veces por segundo, también conocido como un gigahercio, e incluso si usamos múltiples ciclos de reloj para "hacer algo útil" como esto, eso significa que la computadora puede hacer algo como esto cientos de millones de veces en un segundo.

Pero no queremos simplemente agregar R1 a R0 una y otra vez.

Quizás ahora que hemos agregado R1 a R0, queremos almacenar ese nuevo número en una dirección particular en la RAM, y R2 tiene esa dirección en él. Una vez más, nuestro procesador tiene todas las conexiones necesarias para hacer esto, y nuevamente tomará más de un ciclo de reloj para hacerlo. En el paso 4, podemos mover R2 a través del autobús a MAR. En el paso 5 podemos mover R0 a través del bus a la RAM. Eso es todo lo que se necesita, solo dos ciclos de reloj y listo.

El cableado para esta operación es más simple que el otro, solo dos habilitaciones y dos juegos.

¿Qué sigue?

Ahora, aquí hay una idea aterradora. Imagine que el trabajo que realiza un empleado en un restaurante de comida rápida se divide en sus elementos individuales. Camine hacia el mostrador, diga "¿Puedo tomar su pedido?", escuche la respuesta, presione el botón "hamburguesa con queso" en la caja registradora, etc. Ahora digamos que hay 256 o menos acciones individuales involucradas en el trabajo de trabajar en un establecimiento de este tipo. Luego podría inventar un código que asociaría uno de los estados de un byte con cada una de las actividades individuales de un empleado. Entonces podría expresar la secuencia de las acciones de un empleado como una secuencia de bytes.

Primero formamos una tabla de códigos. Escribimos algunos códigos en el lado izquierdo de la página. Luego decidimos qué queremos que signifiquen esos códigos y escribimos esos significados junto a los códigos. Ahora tenemos una lista de todas las acciones posibles que podría realizar un empleado y un código que representa cada una de ellas:

```
0000 0000 = Caminar hacia el mostrador
0000 0001 = Diga "¿Puedo tomar su pedido?"
0000 0010 = Escucha la respuesta
0000 0011 = Presione el botón de
hamburguesa con queso 0000 0100 = Presione
el botón de papas fritas.
0000 0101 = Presione el botón de
leche 0000 0110 = Presione el botón
total 0000 0111 = Recoger el dinero
0000 1000 = Dar el cambio al cliente 0000
1001 = Abrir una bolsa vacía
0000 1010 = Coloque una hamburguesa con
queso en la bolsa 0000 1011 = Coloque las
papas fritas en la bolsa
0000 1100 = Colocar un recipiente de leche en la
bolsa 0000 1101 = Entregar la bolsa al cliente
1000 0000 = Vaya al número de paso en los 6 bits de la
derecha. 0100 0000 = Si es "sí", vaya al número de paso
de la derecha
6 bits.
0001 0000 = Ir a casa.
```

Ahora, si queremos describir cómo se supone que debe actuar el empleado, escribimos una secuencia de eventos que debe seguir:

1. 0000 0000 = Camine hacia el mostrador.
2. 0000 0001 = Diga "¿Puedo tomar su pedido?"
3. 0100 0010 = Si el cliente no responde, vaya al paso 2.
4. 0000 0010 = Escuche la respuesta.
5. 0100 0111 = Si el cliente no dice hamburguesa con queso, vaya al paso 7.
6. 0000 0011 = Presione el botón de hamburguesa con queso.
7. 0100 1001 = Si el cliente no dice papas fritas, vaya al paso 9.
8. 0000 0100 = Presione el botón de papas fritas.
9. 0100 1011 = Si el cliente no dice leche, vaya al paso 11.
10. 0000 0101 = Presione el botón de leche.
11. 0100 1101 = Si el cliente dice que eso es todo, vaya al paso 13.
12. 1000 0100 = Vuelva al paso 4.
13. 0000 0110 = Pulse el botón de total.
14. 0000 0111 = Recoger el dinero.
15. 0000 1000 = Hacer cambio y dárselo al cliente.
16. 0000 1001 = Abrir una bolsa vacía.
17. 0101 0011 = Si el pedido no incluye hamburguesa con queso, vaya al paso 19.
18. 0000 1010 = Coloque una hamburguesa con queso en la bolsa.
19. 0101 0110 = Si el pedido no incluye papas fritas, vaya al paso 22.
21. 0000 1011 = Coloque las papas fritas en la bolsa.
22. 0101 1000 = Si el pedido no incluye leche, vaya al paso 24.
23. 0000 1100 = Coloque un recipiente de leche en la bolsa.
24. 0000 1101 = Entregar la bolsa al cliente.
25. 0101 1011 = Si es hora de salir, vaya al paso 27.
26. 1000 0001 = Volver al paso 1.
27. 0001 0000 = Ir a casa.

Espero que nadie intente hacer que los empleados de un restaurante de comida rápida aprendan un código como este. A la gente no le gusta ser tan mecanizada. Pero tal vez alguien intente dotar de personal a uno de estos restaurantes con robots algún día. En ese caso, los robots probablemente funcionarían mejor usando este tipo de código.

Y nuestra computadora podría ser capaz de "entender" un

código como este.

El primer gran invento

Lo que necesitamos es alguna forma de hacer diferentes operaciones de una secuencia paso a paso a la siguiente. ¿Cómo podríamos tenerlo conectado de una manera para una secuencia y luego de una manera diferente para la siguiente secuencia? La respuesta, por supuesto, es usar más puertas. El cableado para una operación se puede conectar o desconectar con puertas AND, y el cableado para una operación diferente se puede conectar o desconectar con algunas puertas AND más. Y podría haber una tercera y cuarta posibilidad o más. Siempre que solo una de esas operaciones esté conectada a la vez, esto funcionará bien. Ahora tenemos varias operaciones diferentes que se pueden realizar, pero ¿cómo se selecciona cuál se hará?

El título de este capítulo es "El primer gran invento", entonces, ¿cuál es el invento? El invento es que tendremos una serie de instrucciones en RAM que le dirán a la CPU qué hacer. Necesitamos tres cosas para que esto funcione.

La primera parte de la invención es que vamos a agregar otro registro a la CPU. Este registro se llamará "Registro de instrucciones" o "IR" para abreviar. Los bits de este registro "indicarán" a la CPU qué hacer. El IR obtiene su entrada del bus y su salida va a la sección de control de la CPU donde los bits seleccionan una de varias operaciones posibles.

La segunda parte de la invención es otro registro en la CPU llamado "Registro de direcciones de instrucción" o "IAR" para abreviar. Este registro tiene su entrada y salida conectadas al bus al igual que los registros de propósito general, pero este solo tiene un propósito, y es almacenar la dirección RAM de la siguiente instrucción que queremos mover al IR. Si el IAR contiene 0000 1010 (10 decimales), entonces la siguiente instrucción que se moverá al IR es el byte que reside en la dirección RAM diez.

La tercera parte de la invención es un cableado en la sección de control que utiliza el paso a paso para mover la "instrucción" deseada de la RAM al IR, añadir 1 a la dirección en el IAR y realizar la acción requerida por

el

instrucción que se ha puesto en la RI. Cuando se completa esa instrucción, el paso a paso comienza de nuevo, pero ahora se le ha agregado 1 al IAR, por lo que cuando obtenga esa instrucción de la RAM, será una instrucción diferente que se encontraba en la siguiente dirección de RAM.

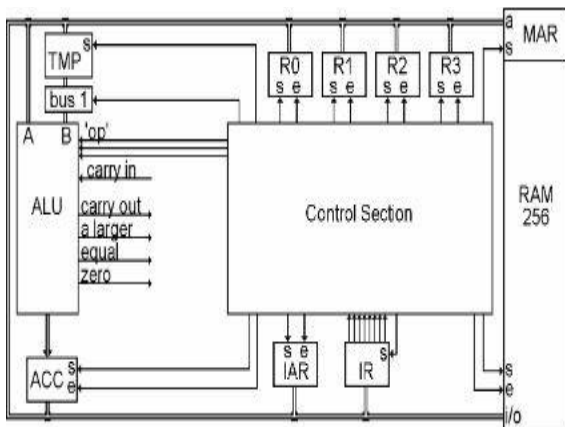
El resultado de estas tres partes es un gran invento. Esto es lo que nos permite hacer que la computadora haga muchas cosas diferentes. Nuestro bus, ALU, RAM y registros hacen posibles muchas combinaciones. El contenido del RI determinará qué registros se envían a dónde, y qué tipo de aritmética o lógica se hará sobre ellos. Lo único que tenemos que hacer es colocar una serie de bytes en la RAM que representan una serie de cosas que queremos hacer, una tras otra.

Esta serie de bytes que residen en la RAM que la CPU va a utilizar se llama "programa".

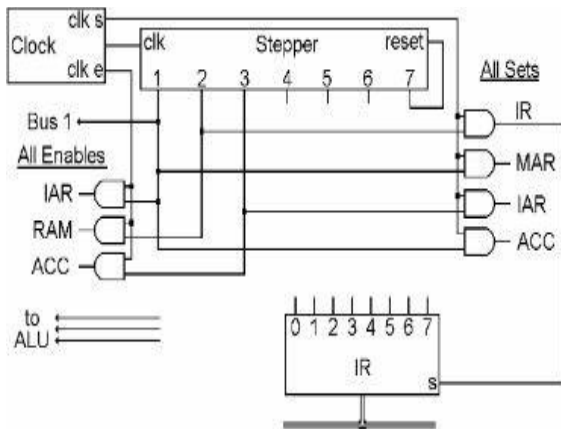
Lo básico que sucede aquí es que la CPU "obtiene" una instrucción de la RAM y luego "ejecuta" la instrucción. Luego busca el siguiente y lo ejecuta. Esto sucede una y otra vez, millones o miles de millones de veces por segundo. Esta es la simplicidad de lo que hace una computadora. Alguien pone un programa en la RAM, y ese programa, si está diseñado inteligentemente, hace que la computadora haga algo que la gente encuentra útil.

El paso a paso en esta computadora tiene siete pasos. El propósito del paso 7 es solo restablecer el paso a paso al paso 1. Por lo tanto, hay seis pasos durante los cuales la CPU hace pequeñas cosas. Cada paso dura un ciclo de reloj. Los seis pasos tomados en su conjunto se denominan "ciclo de instrucción". Se necesitan seis pasos para que la CPU realice todas las acciones necesarias para obtener y ejecutar una instrucción. Si asumimos que nuestro reloj marca un gigahercio, entonces nuestra computadora podrá ejecutar 166,666,666 instrucciones por segundo.

Aquí está la imagen de la CPU con los dos nuevos registros agregados. Allí están bajo la Sección de Control, conectados al bus. El IAR tiene un 'set' y 'enable', el IR solo tiene un 'set', al igual que TMP y MAR porque sus salidas no están conectadas al bus, por lo que nunca necesitamos apagarlas.



A continuación se muestra el cableado dentro de la Sección de Control que realiza la parte de "búsqueda" del ciclo de instrucción. Utiliza los tres primeros pasos del paso a paso y es el mismo para todos los tipos de instrucciones.



Los primeros tres pasos del paso a paso se muestran aquí y dan como resultado la "obtención" de la siguiente "instrucción" de la RAM. Luego, el resto de los pasos 'ejecutan' la 'instrucción'. Exactamente lo que se hará en los pasos 4, 5 y 6 está determinado por el contenido de la instrucción que se obtuvo. Luego, el paso a paso comienza de nuevo, obtiene la siguiente instrucción y la ejecuta.

La parte inferior de este diagrama incluye el registro de instrucciones. Observe que hemos dado números a los bits individuales del IR, 0 a la izquierda a 7 a la derecha. Pronto nos referiremos a los bits individuales.

Aquí están los detalles de exactamente cómo los pasos 1, 2 y 3 dan como resultado obtener una instrucción en nuestra pequeña computadora:

El paso 1 es el más complicado porque en realidad logramos dos cosas al mismo tiempo. Lo principal que queremos hacer es obtener la dirección en IAR a MAR. Esta es la dirección de la siguiente instrucción que queremos obtener de la RAM. Si observa el cable que sale del paso 1 del paso a paso, puede ver que dos de los lugares a los que está conectado son el 'habilitar' de IAR y el 'conjunto' de MAR. Por lo tanto, el contenido de IAR se colocará en el bus durante 'clk e' y se establecerá en MAR durante 'clk s'. En algún momento durante el ciclo de instrucción, necesitamos agregar 1 al valor en IAR, y dado que IAR ya está en el bus, también podríamos hacerlo ahora. Si no enviamos nada a los bits 'op' de la ALU, todos serán cero, y dado que 000 es el código para ADD, la ALU realizará una operación ADD en lo que sea que esté en sus dos entradas y presentará la respuesta a ACC. Una entrada proviene del bus, que tiene IAR durante este tiempo. Si también activamos el bit 'bus 1' durante el paso 1, la otra entrada a la ALU será un byte con el valor binario de 1. Si activamos el 'set' de ACC durante 'clk s', capturaremos la suma de IAR más 1 en ACC. ¡Esta es la dirección de la instrucción que queremos obtener después de que hayamos terminado con la actual!

El paso 2 habilita el byte seleccionado actualmente en la RAM en el bus y lo establece en IR. Esta es la instrucción que 'ejecutaremos' en los pasos 4, 5 y 6 de este ciclo de instrucción. En el diagrama, puede ver que el cable que viene del paso 2 está conectado a la 'habilitación' de RAM y al 'conjunto' de IR.

En el paso 3, debemos terminar de actualizar IAR. Le agregamos 1 en el paso 1, pero la respuesta sigue estando en ACC. Debe trasladarse a IAR antes del comienzo del próximo ciclo de instrucción. Así que puedes ver que el cable que sale del paso 3 está conectado a 'enable' de ACC y 'set' de IAR.

Para cuando llegamos al paso 4, la instrucción ya se ha movido de RAM a IR, y ahora los pasos 4, 5 y 6 pueden hacer lo que requiera el contenido de IR. Cuando se realiza esa operación y se restablece el paso a paso, la secuencia comenzará de nuevo, pero ahora se le ha agregado 1 a IAR, por lo que se obtendrá y ejecutará la instrucción en la siguiente dirección RAM.

Esta idea de poner una serie de instrucciones en la RAM y

hacer que la CPU los ejecute es un gran invento.

Instrucciones

Ahora tenemos este nuevo registro, llamado Registro de Instrucciones, que contiene un byte que le dirá a la Sección de Control qué hacer. Los patrones que se ponen en este registro tienen un significado. Suena como otro código, y de hecho, lo es. Este código se llamará "Código de instrucción".

Dado que estamos construyendo esta computadora desde cero, podemos inventar nuestro propio código de instrucciones. Tomaremos los 256 códigos diferentes que se pueden poner en el Registro de Instrucciones y decidiremos qué significarán. Luego tenemos que diseñar el cableado dentro de la unidad de control que hará que estas instrucciones hagan lo que dijimos que harían.

¿Recuerdas el código numérico binario? Dijimos que era lo más parecido a un código informático "natural" porque se basaba en el mismo método que usamos para nuestro sistema numérico normal. Luego estaba el código ASCII, que fue inventado por un grupo de personas en una reunión. No hay nada natural en ASCII en absoluto, fue justo lo que esas personas decidieron que sería.

Ahora tenemos el Código de Instrucción, que también será un código totalmente inventado, no tiene nada de natural. Se han inventado muchos códigos de instrucciones diferentes para muchos tipos diferentes de computadoras. No estudiaremos ninguno de ellos aquí, ni necesitará estudiar ninguno de ellos más adelante, a menos que vaya a seguir una carrera altamente técnica donde sea necesario. Pero todos los códigos de instrucción son similares, ya que son los que hacen que la computadora funcione. El único código de instrucciones en este libro será uno que inventemos para nuestra computadora simple. Lo más importante al inventar nuestro código de instrucciones será qué tan simple podemos hacer el cableado que hará que el código funcione.

¿Cuántas instrucciones diferentes podría haber? Dado que el registro de instrucciones es un byte, puede haber hasta 256 instrucciones diferentes. Afortunadamente, solo tendremos nueve tipos de instrucciones, y las 256 combinaciones entrarán en una de estas categorías.

Son bastante fáciles de describir.
Todas las instrucciones implican mover bytes a través del bus.

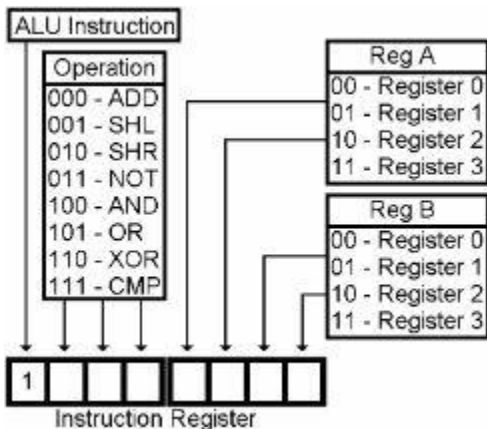
Las instrucciones harán que los bytes vayan hacia o desde la RAM, hacia o desde los registros y, a veces, a través de la ALU. En los siguientes capítulos, para cada tipo de instrucción, veremos los bits de esa instrucción, las puertas y el cableado necesarios para que funcione, y otro código útil que podemos usar para facilitar la escritura de programas.

La instrucción aritmética o lógica

Este primer tipo de instrucción es el tipo que usa la ALU como nuestra operación ADD anterior. Como recordarán, la ALU tiene ocho cosas que puede hacer, y para algunas de esas cosas usa dos bytes de entrada, para otras cosas solo usa un byte de entrada. Y en siete de esos casos, tiene un byte de salida.

Este tipo de instrucción elegirá una de las operaciones ALU y dos registros. Esta es la instrucción más versátil que puede hacer la computadora. En realidad, tiene 128 variaciones, ya que hay ocho operaciones y cuatro registros, y puedes elegir dos veces entre los cuatro registros. Eso es ocho veces cuatro por cuatro, o 128 formas posibles de usar esta instrucción. Por lo tanto, esta no es solo una instrucción, sino que es toda una clase de instrucciones que usan el mismo cableado para hacer el trabajo.

Aquí está el código de instrucción para la instrucción ALU. Si el primer bit en el registro de instrucciones es un 1, entonces esta es una instrucción ALU. Esa es la simplicidad de esto. Si el primer bit está encendido, los siguientes tres bits de la instrucción se envían a la ALU para decirle qué hacer, los siguientes dos bits eligen uno de los registros que se usarán y los dos últimos bits eligen el otro registro que se usará.



Por lo tanto, la Instrucción ALU (1), para sumar (000) el Registro 2 (10) y el Registro 3 (11), y colocar la respuesta en el Registro 3, sería: 1000 1011. Si colocaba este código (1000 1011) en la RAM en la dirección 10, y configuraba el IAR en 10, e iniciaba la computadora, obtendría el 1000 1011 de la dirección 10, lo colocaría en IR y luego el cableado en la sección de control haría la adición de R2 y R3.

Si elige una operación de entrada, como SHL, SHR o NOT, el byte provendrá de la Reg A, pasará por la ALU y la respuesta se colocará en la Reg B. Puede elegir ir de un registro a otro, como R1 a R3, o elegir volver de un registro al mismo, como R2 a R2. Cuando haga esto último, se reemplazará el contenido original del registro.

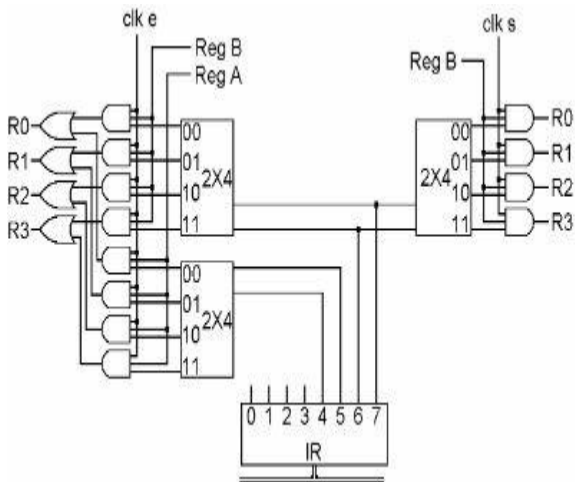
Para dos operaciones de entrada, se enviarán Reg A y Reg B

al ALU, y la respuesta se enviará a Reg B. Entonces, lo que sea que estuviera en Reg B, que fue una de las entradas para la operación, será reemplazado por la respuesta. También puede especificar el mismo registro para ambas entradas. Esto puede ser útil, por ejemplo, si desea poner todos los ceros en el Registro 1, solo XOR R1 con R1. No importa lo que haya en R1 para empezar, todas las comparaciones de bits serán iguales, lo que hace que la salida de todos los bits sea cero, que se vuelve a colocar en R1.

La operación CMP toma dos entradas y las compara para ver si son iguales y, si no, si la primera es mayor. Pero la operación CMP no almacena su byte de salida. No reemplaza el contenido de ninguno de los bytes de entrada.

El cableado en la unidad de control para la instrucción ALU es bastante simple, pero hay una cosa adicional que será utilizada por muchos tipos de instrucciones que debemos mirar primero. Esto tiene que ver con los registros. En "Doing Something Useful Revisited", usamos dos registros. Para usarlos, simplemente conectamos la puerta AND para cada registro al paso deseado del paso a paso.

Esto estaba bien, pero en la instrucción ALU, y en muchas otras, hay bits en el registro de instrucciones que especifican qué registro usar. Por lo tanto, no queremos conectarnos directamente a ningún registro, necesitamos poder conectarnos a cualquiera de los registros, pero dejar que los bits de la instrucción elijan exactamente cuál. Aquí está el cableado de la Sección de Control que lo hace:



Mira primero el lado derecho. Cuando queremos establecer un registro de propósito general, conectamos el paso adecuado a este cable que llamaremos 'Reg B'. Como puede ver, 'clk s' está conectado a las cuatro puertas AND. 'Reg B' también está conectado a las cuatro puertas AND. Pero estas cuatro puertas AND tienen cada una tres entradas. La tercera entrada a cada puerta AND proviene de un decodificador de 2x4. Recordará que una y solo una salida de un decodificador está encendida en un momento dado, por lo que solo se seleccionará un registro para que se encienda su bit 'set'. La entrada al decodificador proviene de los dos últimos bits del IR, por lo que determinan qué registro establecerá este cable etiquetado como 'Reg B'. Si miras hacia atrás en la tabla del

bits del código de instrucciones de ALU, muestra que los dos últimos bits de la instrucción son los que determinan qué registro desea usar para Reg B.

El lado izquierdo de la imagen es muy parecido al lado derecho, excepto que hay dos de todo. Recuerde que en una instrucción ALU como ADD, necesitamos habilitar dos registros, uno a la vez, para las entradas a la ALU. Los dos últimos bits de la instrucción también se usan para 'Reg B' a la izquierda, y puede ver que 'clk e', 'Reg B' y un decodificador se usan para habilitar un registro durante su paso adecuado. Los bits 4 y 5 del IR se utilizan para habilitar 'Reg A' durante su paso adecuado, utilizando un decodificador separado y un cable llamado 'Reg A'. Las salidas de estas dos estructuras se ORan juntas antes de ir a los bits de habilitación de registro reales. Nunca seleccionaremos 'Reg A' y 'Reg B' al mismo tiempo.

¿Qué sucede cuando la instrucción que se ha obtenido comienza con un 1? Eso significa que esta es una instrucción de ALU, y tenemos que hacer tres cosas. Primero queremos mover 'Reg B' a TMP. Luego queremos decirle a la ALU qué operación hacer, poner 'Reg A' en el bus y establecer la salida de la ALU en ACC. Luego queremos mover ACC a 'Reg B'.

El bit 0 del IR es el que determina si se trata de una instrucción ALU. Cuando el bit 0 está activado, las cosas que el bit 0 está conectado para hacer que ocurran todos los pasos de una instrucción ALU.

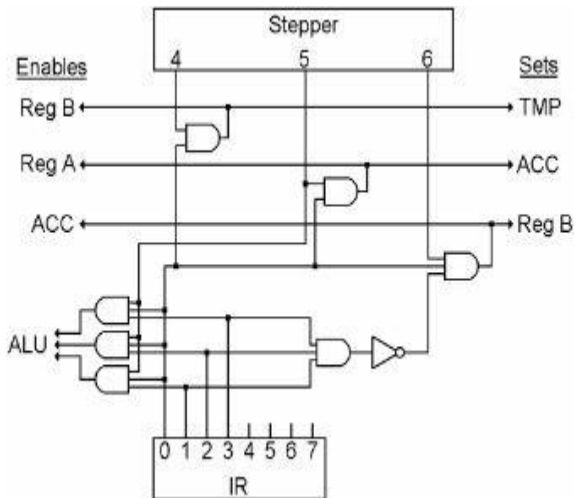
El siguiente diagrama muestra las ocho puertas y los cables que se agregan a la Sección de Control que hacen que los pasos 4, 5 y 6 de una instrucción ALU hacen lo que necesitamos que hagan.

En el diagrama a continuación, justo encima y a la izquierda del IR, hay tres puertas AND. Las salidas de estas puertas van a los tres cables 'op' en la ALU que le indican qué operación hacer. Cada una de estas tres puertas AND tiene tres entradas. Una entrada de cada puerta está conectada al bit 0 del IR. Una segunda entrada de cada puerta está conectada al paso 5 del stepper. La entrada restante de cada puerta está conectada a los bits 1, 2 y 3 del IR.

Por lo tanto, los tres cables que van a la ALU serán

000 en todo momento, excepto durante el paso 5, cuando el bit IR 0 resulta ser un 1. En ese momento, los cables que van a

el ALU será el mismo que los bits 1, 2 y 3 del IR.



El bit IR 0 continúa subiendo por el diagrama, gira a la derecha y está conectado a un lado de tres puertas AND más. Los otros lados de estas puertas están conectados a los pasos 4, 5 y 6.

La salida de la primera puerta se enciende durante el paso 4 y puede ver que va a dos lugares. A la izquierda, habilita 'Reg B' en el bus, y a la derecha, configura el bus en TMP. Este paso en realidad no es necesario para las operaciones SHL, SHR y NOT, pero no daña nada, y sería bastante complicado deshacerse de él, así que en aras de la simplicidad lo dejaremos así.

La segunda puerta se enciende durante el paso 5 (el mismo paso en el que el ALU recibe sus órdenes), y yendo a la izquierda hay un cable que habilita 'Reg A' en el bus. El ALU ahora tiene una entrada en TMP, la otra entrada en el bus y su funcionamiento especificado por esos tres cables 'op', por lo que a la derecha hay un cable que establece la respuesta en ACC.

La tercera puerta se enciende durante el paso 6. El cable que va a la izquierda habilita ACC en el bus, y el cable que va a la derecha establece el bus en 'Reg B'.

Solo hay una situación especial en una instrucción ALU, y es cuando la operación es CMP, código 111. Para una operación de comparación, no queremos almacenar ningún resultado en 'Reg B'. Por lo tanto, hay una puerta AND de tres entradas conectada a los bits IR 1, 2 y 3, que luego se conecta a una puerta NOT y luego a una tercera entrada en la puerta AND que realiza el paso 6 de la instrucción ALU. Entonces, cuando la operación es 111, se encenderá el primer AND, se activará el NOT y la salida de la puerta AND del Paso 6 no se encenderá.

Esta instrucción de ALU ya está hecha. El paso 7 restablece el paso a paso, que luego vuelve a realizar sus pasos, obteniendo la siguiente instrucción, etc, etc.

Vamos a inventar una cosa más aquí, y es una forma abreviada de escribir instrucciones de CPU en una hoja de papel. En el Código de Instrucción, 1000 1011 significa "Agregar R2 a R3", pero se necesita mucha práctica para que una persona mire 1000 1011 e inmediatamente piense en la suma y los registros. También se necesitaría mucha memorización para pensarlo al revés, es decir, si quisieras XOR dos registros, ¿cuál es el código de instrucciones para XOR? Sería más fácil escribir algo como ADD R2, R3 o XOR R1, R1.

Esta idea de usar una taquigrafía tiene un nombre, y se llama lenguaje informático. Entonces, junto con inventar un código de instrucción, también inventaremos un lenguaje informático que represente el código de instrucción. La instrucción de ALU da como resultado las primeras ocho palabras de nuestro nuevo idioma.

Idioma		Significado
AGREGAR	RA, RB	Agregue RA y RB y ponga respuestas

SHR	RA,RB	Cambie RA a la derecha y ponga ans
SHL	RA,RB	Shift RA Izquierda y coloque el botón
NO	RA,RB	No RA y poner la respuesta
Y	RA,RB	Y RA y RB y pon respuesta
O	RA,RB	O RA y RB y poner answe
XOR	RA,RB	Exclusivo OR RA y RB int
CMP	RA,RB	Comparar RA y RB

Cuando una persona quiere escribir un programa de computadora, puede escribirlo directamente en el código de instrucciones o usar un lenguaje de computadora. Por supuesto, si escribe un programa en un lenguaje informático, tendrá que traducirse al código de instrucción real antes de que pueda colocarse en la RAM y ejecutarse.

Las instrucciones de carga y almacenamiento

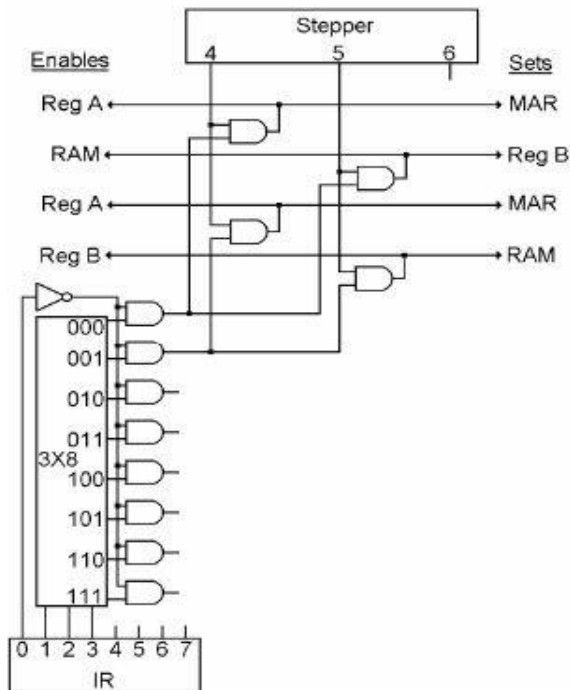
Las instrucciones de carga y almacenamiento son bastante simples. Mueven un byte entre la RAM y un registro. Son muy similares entre sí, por lo que los cubriremos en un capítulo.

Llegaremos a los detalles de estas instrucciones en un momento, pero primero necesitamos tener algo que nos diga cuándo tenemos una instrucción de carga o almacenamiento en el registro de instrucciones. Con la instrucción de ALU, todo lo que necesitábamos saber era que el bit 0 estaba encendido. El código para cualquier otro tipo de instrucción comienza con el bit 0 desactivado, por lo que si conectamos una puerta NOT al bit 0, cuando esa puerta NOT se enciende, eso nos dice que tenemos algún otro tipo de instrucción. En esta computadora, hay ocho tipos de instrucciones que no son instrucciones ALU, por lo que cuando bit

0 está desactivado, usaremos los siguientes tres bits del IR para decirnos exactamente qué tipo de instrucción tenemos.

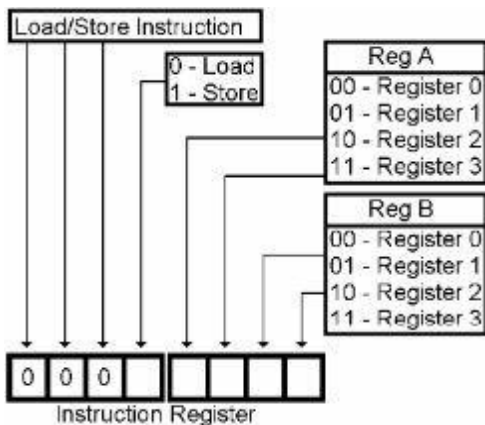
Los tres bits que fueron a la ALU en una instrucción ALU también van a un decodificador 3x8 aquí en la Sección de Control. Como recordará, una y solo una de las salidas de un decodificador está encendida en todo momento, por lo que tendremos puertas AND en las salidas para evitar que cualquier salida vaya a ninguna parte durante una instrucción ALU. Pero cuando no es una instrucción ALU, la única salida del decodificador que está encendida pasará a través de su puerta AND y, a su vez, se conectará a algunas puertas más que hacen que funcione la instrucción adecuada.

En el siguiente diagrama, puede ver los bits IR 1, 2 y 3 que entran en un decodificador que tiene ocho puertas AND en sus salidas. El bit IR 0 tiene una puerta NOT que va al otro lado de esas ocho puertas AND. Este decodificador sirve para el resto de instrucciones que tendrá nuestro ordenador.



Este capítulo trata sobre las instrucciones que usan las dos primeras salidas del decodificador, las que se encienden cuando el IR comienza con 0000 o 0001.

La primera instrucción mueve un byte de la RAM a un registro, esto se llama instrucción "Load". El otro hace lo mismo a la inversa, mueve un byte de un registro a la RAM y se llama instrucción "Store".



El código de instrucción para la instrucción de carga es 0000 y para la instrucción de tienda es 0001. Los cuatro bits restantes en ambos casos especifican dos registros, al igual que lo hizo la instrucción ALU, pero en este caso, un registro se usará para seleccionar una de las ubicaciones en la RAM y el otro registro se cargará o almacenará en esa ubicación de RAM.

El paso 4 es el mismo para ambas instrucciones. Uno de los

registers se selecciona mediante los bits IR 4 y 5 y se habilita en el bus. Luego, el bus se configura en MAR, seleccionando así una dirección en la RAM.

En el paso cinco, los bits IR 6 y 7 seleccionan otro de los registros de la CPU. Para la instrucción Load, la RAM se habilita en el bus y el bus se establece en el registro seleccionado. Para la instrucción Store, el registro seleccionado se habilita en el bus y el bus se establece en la RAM.

Cada una de estas instrucciones solo necesita dos pasos para completarse, el paso 6 no hará nada.

Aquí hay dos nuevas palabras para nuestro lenguaje informático:

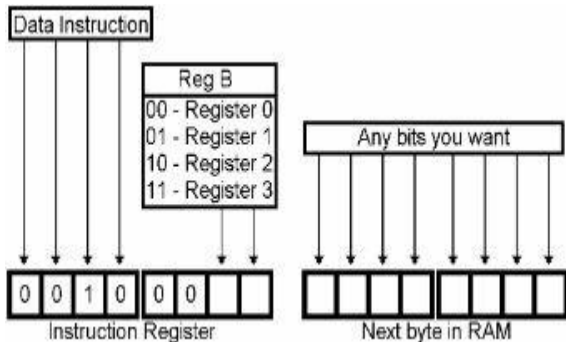
Idioma		Significado
LD	RA,RB	Cargar RB desde la dirección RAM en RA
C	RA,RB	Almacenar RB en la dirección RAM en RA

La instrucción de datos

Ahora bien, aquí hay una instrucción interesante. Todo lo que hace es cargar un byte de RAM en un registro como la instrucción de carga anterior. Sin embargo, lo que es diferente es de dónde obtendrá ese byte en la RAM.

En la instrucción de datos, los datos provienen de donde debería estar la siguiente instrucción. ¡Así que podría considerar que esta instrucción tiene en realidad dos bytes de largo! El primer byte es la instrucción y el siguiente byte son algunos datos que se colocarán en un registro. Estos datos son fáciles de encontrar, porque para cuando tenemos la instrucción en el IR, el IAR ya se ha actualizado, por lo que apunta directamente a este byte.

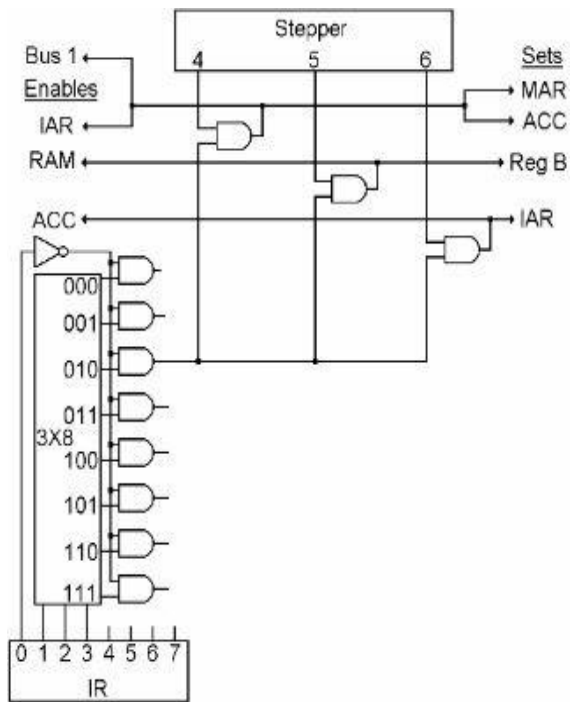
Aquí está el código de instrucción para la instrucción de datos. Los bits 0 a 3 son 0010. Los bits 4 y 5 no se utilizan. Los bits 6 y 7 seleccionan el registro que se cargará con los datos que están en el segundo byte.



Todo lo que esta instrucción debe hacer es, en el paso 4, enviar IAR a MAR y, en el paso 5, enviar RAM a la CPU

deseada

registro. Sin embargo, hay una cosa más que debe suceder. Dado que el segundo byte de la instrucción es solo datos que podrían ser cualquier cosa, no queremos ejecutar este segundo byte como una instrucción. Necesitamos agregar 1 al IAR por segunda vez para que omita este byte y apunte a la siguiente instrucción. Haremos esto de la misma manera que se hace en los pasos 1 y 3. En el paso 4, cuando enviemos IAR a MAR, aprovecharemos el hecho de que la ALU está calculando IAR más algo al mismo tiempo, activaremos el 'Bus 1' y estableceremos la respuesta en ACC. El paso 5 todavía mueve los datos a un registro, y en el paso 6 podemos mover ACC a IAR.



Aquí hay otra palabra nueva para nuestro lenguaje informático:

Idioma		Significado
DATOS	RB, xxxx xxxx	Cargue estos 8 bits en RB

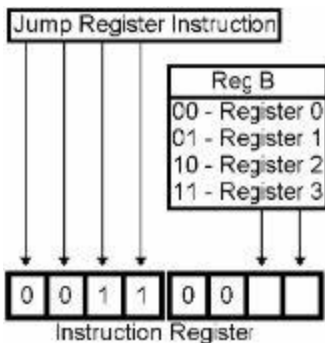
El segundo gran invento

El primer gran invento es esta idea de tener una cadena de instrucciones en la RAM que la CPU ejecuta una por una. Pero nuestro reloj es muy rápido y la cantidad de RAM que tenemos es limitada. ¿Qué sucederá, en mucho menos de un segundo, cuando hayamos ejecutado todas las instrucciones en RAM?

Afortunadamente, no tendremos que responder a esa pregunta, porque a alguien se le ocurrió otro tipo de instrucción que es tan importante que califica como el segundo gran invento necesario para permitir que la computadora haga lo que hace. Debido a la disposición versátil de nuestra CPU y su Sección de Control, es algo extremadamente simple hacer que esto funcione, pero su importancia no debe perderse debido a esta simplicidad.

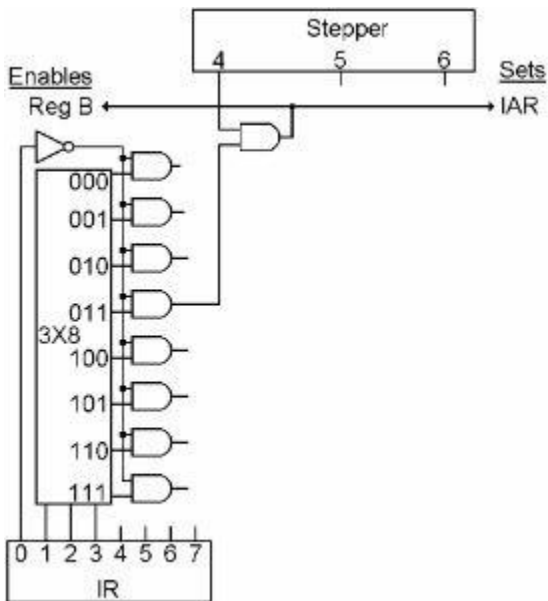
Este nuevo tipo de instrucción se llama instrucción de salto, y todo lo que hace es cambiar el contenido del IAR, cambiando así de qué parte de la RAM vendrán las instrucciones siguientes y posteriores.

El tipo exacto de instrucción de salto descrito en este capítulo se denomina instrucción de "registro de salto". Simplemente mueve el contenido de Reg B al IAR. Aquí está el código de instrucciones para ello:



La computadora está ejecutando una serie de instrucciones en RAM, una tras otra, y de repente una de esas instrucciones cambia el contenido del IAR. ¿Qué pasará entonces? La siguiente instrucción que se obtendrá no será la que siga a la última. Será el que esté en cualquier dirección RAM que se haya cargado en el IAR. Y continuará desde ese punto con el siguiente, etc. hasta que ejecute otra instrucción de salto.

El cableado para la instrucción Jump Register solo necesita un paso. En el paso 4, el registro seleccionado se habilita en el bus y se establece en el IAR, y eso es todo. Si quisiéramos acelerar nuestra CPU, podríamos usar el paso 5 para restablecer el paso a paso. Pero para mantener nuestro diagrama simple, no nos molestaremos con eso. Los pasos 5 y 6 no harán nada.



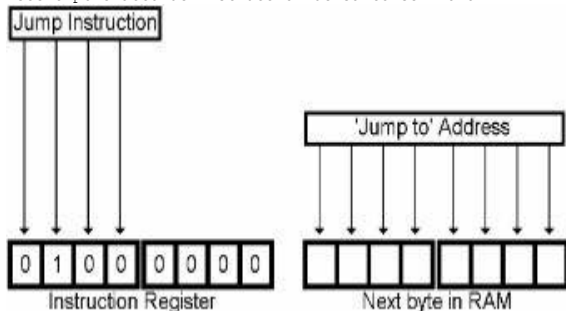
Aquí hay otra palabra nueva para nuestro lenguaje informático:

Idioma		Significado
JMPL	RB	Saltar a la dirección en RB

Otra forma de saltar

Este es otro tipo de instrucción de salto. Es similar a la instrucción Data en que utiliza dos bytes. Reemplaza el IAR con el byte que está en la RAM inmediatamente después del byte de instrucción, cambiando así de dónde provendrán las instrucciones siguientes y posteriores. Aquí está el código de instrucciones para ello. Los bits 4, 5, 6 y 7 no se utilizan en esta instrucción:

Este tipo exacto de instrucción de salto se llama

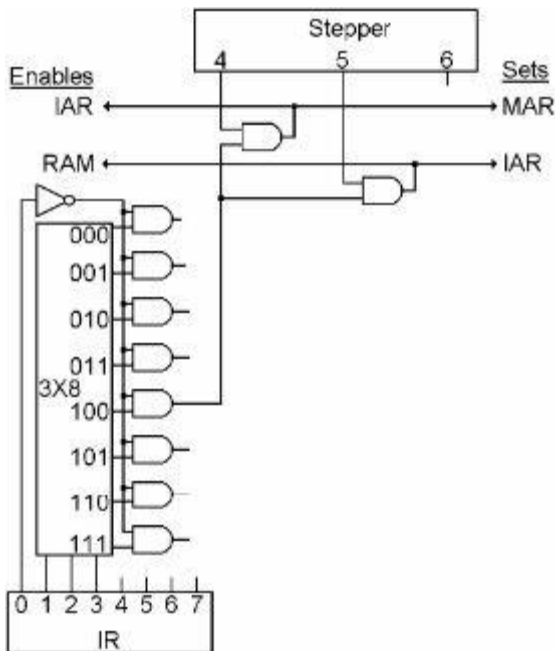


simplemente "salto". Es útil cuando conoce la dirección a la que va a querer saltar, cuando está escribiendo el programa. La instrucción de registro de salto es más útil cuando la dirección a la que desea saltar se calcula como el programa en ejecución, y puede que no siempre sea la misma.

Una de las cosas que puede hacer con una instrucción de salto es crear un bucle de instrucciones que se ejecuten una y otra vez. Puede tener una serie de cincuenta instrucciones en la RAM, y la última instrucción "salta" de nuevo a la primera.

Al igual que la instrucción Data, el IAR ya apunta al byte que necesitamos. A diferencia de la instrucción de datos, no necesitamos agregar 1 al IAR por segunda vez porque lo reemplazaremos de todos modos. Así que solo necesitamos dos pasos. En el paso 4, enviamos IAR a MAR. En el paso 5 movemos el byte de RAM seleccionado al IAR. El paso 6 no hará nada.

Aquí está el cableado que lo hace funcionar:



Aquí hay otra palabra nueva para nuestro lenguaje informático:

Idioma		Significado
--------	--	-------------

JMP	Addr	Saltar a la dirección en el siguiente byte
-----	------	--

El tercer gran invento

Aquí está el tercer y último invento que hace que una computadora sea una computadora.

Esto es como la instrucción de salto, pero a veces salta y a veces no. Por supuesto, saltar o no saltar son solo dos posibilidades, por lo que solo se necesita un poco para determinar cuál sucederá. Principalmente, lo que vamos a presentar en este capítulo es de dónde viene esa parte.

¿Recuerdas la parte de 'Carry' que sale y vuelve a entrar en el ALU? Esta parte proviene de la víbora o de una de las palancas de cambio. Si suma dos números que dan como resultado una cantidad superior a 255, se encenderá el bit de acarreo. Si desplaza a la izquierda un byte que tiene el bit izquierdo activado, o desplaza a la derecha un byte que tiene el bit derecho activado, estas situaciones también activarán el bit de realización de la ALU.

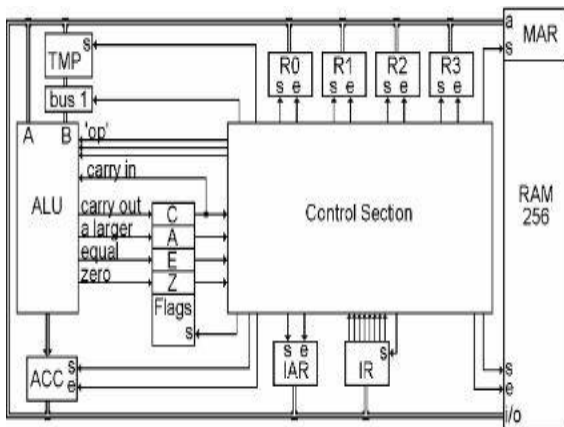
También hay un bit que nos dice si las dos entradas a la ALU son iguales, otro que nos dice si la entrada A es más grande y un bit más que nos dice si la salida de la ALU es todo ceros.

Estos bits son las únicas cosas para las que aún no hemos encontrado un hogar en la CPU. Estos cuatro bits se llamarán bits "Flag" y se utilizarán para tomar la decisión de una instrucción "Jump If" sobre si ejecutará la siguiente instrucción en RAM o saltará a alguna otra dirección.

Lo que estamos tratando de lograr es que la computadora pueda lograr es que primero ejecute una instrucción ALU y luego tenga una o más instrucciones de "Saltar si" a continuación. El "Saltar si" saltará o no dependiendo de algo que haya sucedido durante la instrucción de ALU.

Por supuesto, para cuando se ejecuta el "Saltar si", los resultados de la instrucción ALU desaparecen. Si regresa y observa los detalles de la instrucción de ALU, es solo durante el paso 5 que todas las entradas adecuadas ingresan a la ALU y sale la respuesta deseada. Es en este momento que la respuesta se establece en ACC. El tiempo es el mismo para los cuatro bits de Flag, solo son

válido durante el paso 5 de la instrucción ALU. Por lo tanto, necesitamos una forma de guardar el estado de los bits de Flag tal como estaban durante el paso 5 de la



instrucción ALU.

Aquí está el último registro que vamos a agregar a la CPU. Esto se llamará registro FLAG, y solo vamos a usar cuatro bits de él, uno para cada una de las banderas.

Los bits de bandera de la ALU están conectados a la entrada de este registro, y se configurarán durante el paso 5 de la instrucción ALU al igual que ACC y permanecerán configurados de esa manera hasta la próxima vez que se ejecute una instrucción ALU. Por lo tanto, si tiene una instrucción ALU seguida de una instrucción "Saltar si", los bits "Bandera" se pueden usar para "decidir" si saltar o no.

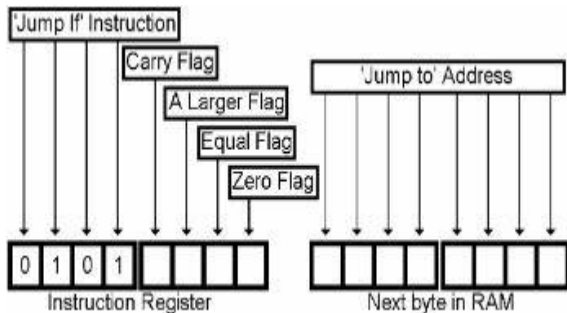
Cada ciclo de instrucción utiliza el ALU en el paso 1 para

agregar 1

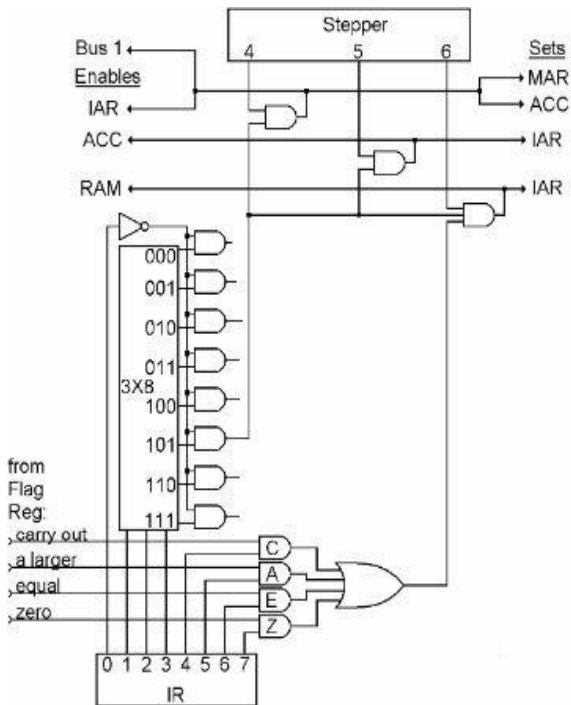
a la dirección de la siguiente instrucción, pero solo el paso 5 de la instrucción ALU tiene una conexión que establece los indicadores. (No mostramos esta conexión en el cableado para la instrucción ALU porque aún no habíamos introducido el Flag Reg, pero aparecerá en el diagrama de la Sección de Control completa).

Esta combinación de bits Flag y la instrucción Jump IF es el tercer y último gran invento que hace que las computadoras tal como las conocemos hoy funcionen.

Aquí está el código de instrucción para una instrucción 'Saltar si'. Los segundos cuatro bits de la instrucción le dicen a la CPU qué bandera o banderas deben verificarse. Pones un '1' en los bits de instrucción correspondientes a los indicadores que deseas probar. Si alguna de las banderas que pruebes está activada, el salto ocurrirá. Este arreglo nos da varias formas de decidir si saltar o no. Hay un segundo byte que contiene la dirección a la que se va a saltar, si se realiza el salto.



Aquí está el cableado en la sección de control que hace que la instrucción Jump If funcione.



El paso 4 mueve IAR a MAR para que estemos preparados para obtener el 'Saltar a la dirección' que usaremos SI saltamos. Pero como es posible que no saltemos, también necesitamos calcular la dirección de la siguiente instrucción en la RAM. Y así, el paso 4 también enciende el bus 1 y establece la respuesta en ACC.

En el paso 5, movemos ACC a IAR para que estemos listos para obtener la siguiente instrucción SI no saltamos.

El paso 6 es donde se toma la "decisión". Moveremos el segundo byte de la instrucción de RAM a IAR SI la tercera entrada a esa puerta AND está activada. Esa tercera entrada proviene de una puerta OR con cuatro entradas. Esas cuatro entradas provienen de los cuatro bits de bandera después de ser AND con los últimos cuatro bits de la instrucción Jump If en IR. Si, por ejemplo, hay un '1' en el bit 'Igual' de la instrucción, y el bit de bandera 'Igual' está activado, entonces se producirá el salto.

Aquí hay más palabras para nuestro lenguaje informático. 'J' significa Salto, 'C' significa Carry, 'A' significa que A es más grande, 'E' significa A es igual a B y 'Z' significa que la respuesta es todo ceros. Estas son las palabras del lenguaje que prueban una sola bandera:

Idioma		Significado
JC	Addr	Salta si el carry está encendido
JA	Addr	Salta si A es más grande que B
JE	Addr	Salta si A es igual a B
JZ	Addr	Salta si la respuesta es Cero

También puede probar más de un bit de bandera al mismo tiempo poniendo un 1 en más de uno de los cuatro bits. En realidad, dado que hay cuatro bits, hay 16 combinaciones posibles, pero la que tiene los cuatro bits apagados no es útil porque nunca saltará. En aras de la exhaustividad, aquí están el resto de las posibilidades:

Idioma		Significado
JCA	Addr	Salta si llevas o uno más grande
JCE	Addr	Salta si llevas o A es igual a B

JCZ	Addr	Saltar si el carry o la respuesta es cero
-----	------	---

JAE	Addr	Saltar si A es mayor o igual a B
JAZ	Addr	Saltar si A es más grande o la respuesta es
JEZ	Addr	Saltar si A es igual a B o la respuesta es Z
JCAE	Addr	Salta si llevas o un más grande o Equa
JCAZ	Addr	Saltar si es portador o uno más grande o cero
JCEZ	Addr	Salta si carry o A es igual a B o ze
JAEZ	Addr	Saltar si A es mayor o igual a B o
FÁCIL	Addr	Saltar si lleva, A más grande, Igual o

La instrucción Clear Flags

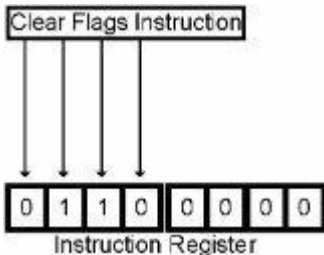
Hay un detalle molesto que debemos tener aquí. Cuando realiza una suma o un cambio, tiene la posibilidad de que la operación encienda la bandera de acarreo. Esto es necesario, lo usamos para la instrucción Jump If como en el capítulo anterior.

El indicador de acarreo también se utiliza como entrada para las operaciones de suma y desplazamiento. El propósito de esto es que pueda agregar números mayores de 255 y cambiar bits de un registro a otro.

El problema que surge es que si solo está agregando dos números de un solo byte, no le importa ningún Carry anterior, pero el Carry Flag aún puede establecerse a partir de una operación anterior. En ese caso, ¿puedes sumar 2+2 y obtener 5!

Las computadoras más grandes tienen varias formas de hacer esto, pero para nosotros, solo tendremos una instrucción de Clear Flags que debe usar antes de cualquier adición o cambio en el que un bit de transporte inesperado sería un problema.

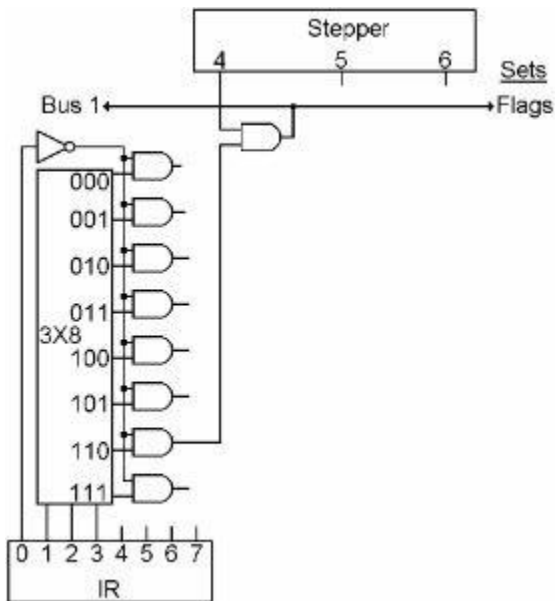
Aquí está el código de instrucciones para esta instrucción. Los bits 4, 5, 6 y 7 no se utilizan.



El cableado para esto es muy simple y un poco complicado. No habilitaremos nada en el bus, por lo que él y la entrada ALU 'A' serán todos ceros. Encenderemos el 'Bus

1'

por lo que la entrada 'B' es 0000 0001. No enviaremos una operación a la ALU, por lo que estará en modo ADD. El ALU, por lo tanto, sumará 0 y 1, y puede haber una entrada de acarreo. La respuesta entonces será 0000 0001 o 0000 0010. Pero no habrá salida de acarreo, la respuesta no es cero y B es más grande que A, por lo que 'igual' y 'A más grande' estarán apagados. "Configuramos" el registro de banderas en este momento mientras los cuatro bits de bandera están apagados.

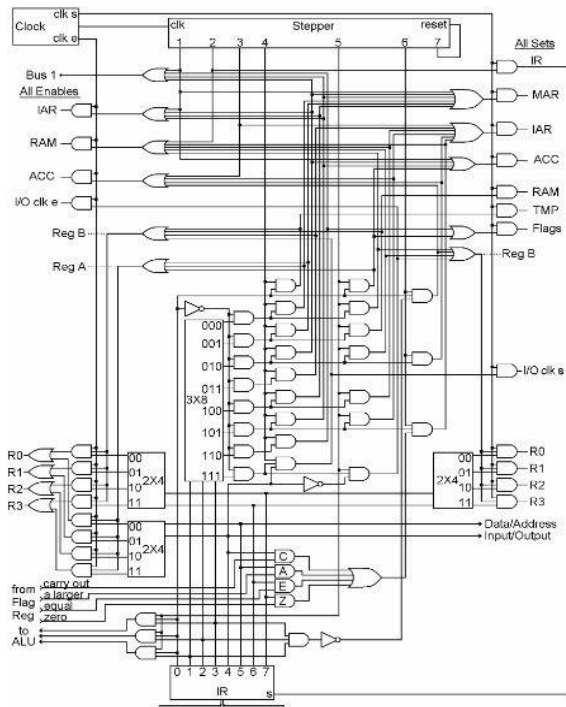


Aquí hay otra palabra para nuestro idioma.

Idioma		Significado
CLF		Borrar todas las banderas

¡Ta Daa!

Ahora hemos conectado la sección de control de nuestra CPU. Como resultado, podemos colocar una serie de instrucciones en la RAM, y el reloj, el paso a paso, el registro de instrucciones y el cableado buscarán y ejecutarán esas instrucciones. Aquí está toda la sección de control:



Sí, esto parece bastante complicado, pero ya hemos analizado cada parte. Lo único que tuvimos que agregar fueron algunas puertas OR porque la mayoría de los 'habilitadores' y 'conjuntos' necesitan múltiples conexiones. En realidad, tiene muchas menos partes que la RAM, pero eso era mucho más repetitivo. La mayor parte del desorden aquí es simplemente llevar los cables de un lugar a otro.

El byte que se coloca en el registro de instrucciones hace que se produzca una determinada actividad. Cada patrón posible provoca una actividad diferente. Por lo tanto, tenemos un código donde cada uno de los 256 códigos posibles representa una actividad específica diferente.

Como se mencionó, esto se llama Código de instrucciones. Otro nombre para esto es "lenguaje de máquina", porque este es el único lenguaje (código) que la máquina (computadora) "entiende". Le "dices" a la máquina qué hacer dándole una lista de órdenes que quieres que lleve a cabo. Pero tienes que hablar el único idioma que "entiende". Si lo alimenta con los patrones correctos de encendido y apagado del tamaño de un byte, puede hacer que haga algo que sea útil.

Aquí están todos los códigos de instrucción y nuestro lenguaje abreviado reunidos en un solo lugar.

Código de instrucciones		Idioma		Significado
1000	rarb	AGREGAR	RA,RB	Agregar
1001	rarb	SHR	RA,RB	Cambio R
1010	rarb	SHL	RA,RB	Mayús L
1011	rarb	NO	RA,RB	No
1100	rarb	Y	RA,RB	Y
1101	rarb	O	RA,RB	O
1110	rarb	XOR	RA,RB	Exclusivo
1111	rarb	CMP	RA,RB	Comparar
0000	rarb	LD	RA,RB	Cargar RB
0001	rarb	C	RA,RB	Tienda

				R
0010	00k xxxxxxxxxx	DATOS	RB,Datos	Cargar th
0011	00rb	JMPR	RB	Saltar a
0100	0000 xxxxxxxxxx	JMP	Addr	Saltar a

0101	Cáez xxxxxxxx	FÁCIL	Addr	Saltar si
0110	0000	CLF		Borrar un

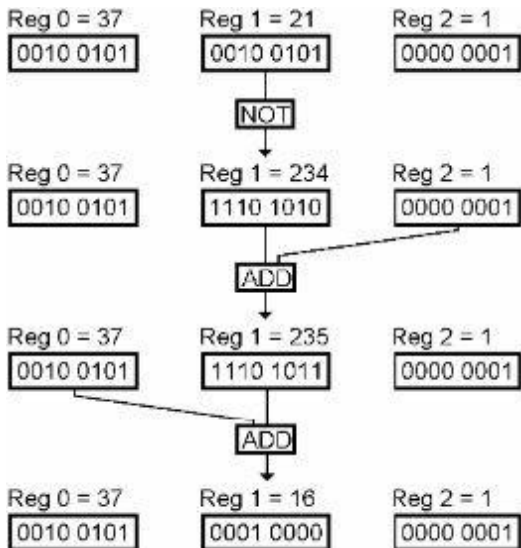
Lo creas o no, todo lo que has visto hacer a una computadora es simplemente el resultado de una CPU que ejecuta una larga serie de instrucciones como las anteriores.

Algunas palabras más sobre aritmética

No queremos dedicar mucho tiempo a este tema, pero lo único que hemos visto hasta ahora que parece aritmética es el sumador, por lo que veremos ejemplos simples de aritmética un poco más compleja. No para enseñarte cómo actuar como una computadora, sino solo para demostrarte que funciona.

Así es como se resta. Se hace con la víbora y las puertas NOT. Si desea restar R1 de R0, primero NO vuelva a R1 en sí mismo. Luego agrega 1 a R1, luego agrega R0 a R1.

Esto muestra un ejemplo de restar 21 de 37:



El último paso es sumar $37 + 235$, cuya respuesta debería ser 272. Pero un solo registro no puede contener un número mayor de 255. Por lo tanto, el sumador enciende su bit de transporte, y los ocho bits restantes de la respuesta son 0001 0000, que es 16, la respuesta correcta para 37 menos 21 .

¿Por qué NOTting y ADDing resultan en resta? ¿Por qué tienes que agregar 1 después de NOTting? ¿Por qué

ignoras?

¿la parte de transporte? No vamos a intentar responder a ninguna de estas preguntas en este libro. Estos son los detalles que impiden que muy pocos ingenieros duerman bien por la noche. Estas personas valientes estudian estos problemas y diseñan formas para que la gente común no tenga que entenderlos.

Así es como se hace la multiplicación. Cuando hacemos multiplicaciones con lápiz y papel en el sistema decimal, tienes que recordar tus tablas de multiplicar, ya sabes, 3 por 8 es igual a 24, 6 por 9 es igual a 54, etc.

En binario, la multiplicación es en realidad mucho más fácil que en decimal. 1 por 1 es igual a 1, y para cualquier otra combinación, ¡la respuesta es 0! ¡No podría ser mucho más simple que eso! Aquí hay un ejemplo de multiplicar 5 por 5 con lápiz y papel en binario.

$$\begin{array}{r} 00000101 \\ \times 00000101 \\ \hline 00000101 \\ 00000000 \\ 00000101 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \\ \hline 000000000011001 \end{array}$$

Si miras lo que está sucediendo aquí, si el dígito derecho del número inferior es un 1, pones el número superior en la respuesta. Luego, para cada dígito a la izquierda de eso, desplaza el número superior a la izquierda, y si el dígito inferior es un 1, agrega el número superior desplazado a la respuesta. Cuando pasas por los ocho bits del número inferior, estás

hecho.

Así que la multiplicación se logra con el sumador y los cambiantes. Es tan simple como eso. Puedes escribir un programa simple como este:

R0 contiene el número inferior, R1 contiene el número superior y R2 contendrá la respuesta. R3 se usa para saltar fuera del bucle después de pasarlo ocho veces.

Dirección de RAM	Instrucción		Comentarios
50	DATOS	R3,0000 0001	* Ponga '1' en R3
52	XOR	R2, R2	* Poner '0' en R2
53	CLF		* Borrar banderas
54	SHR	R0	* Un poco para Carr
55	JC	59	* Haz el ADD
57	JMP	61	* Omite el ADD
59	CLF		* Borrar banderas
60	AGREGAR	R1,R2	* AÑADIR esta línea
61	CLF		* Borrar banderas
62	SHL	R1	* Mucho más arriba por 2
63	SHL	R3	* Contador de turnos
64	JC	68	* Fuera si se hace
66	JMP	53	* Haz el siguiente paso
68	(Siguiendo instrucción en el programa)		

Vea lo que sucede con los registros a medida que este programa pasa por su bucle las primeras tres veces.

R0

R2

R1

R3

Al comienzo (después de 52):

0000 0101

0000 0101

0000 0000

0000 0001

Primera vez (después del 63):

0000 0010 0000 1010

0000 0101

0000 0010

Segunda vez (después del 63):

0000 0001 0001 0100

0000 0101

0000 0100

Tercera vez (después de 63):

0000 0000 0010 1000 0001 1001 0000 1000

Lo importante que ha sucedido aquí es que R1 se ha agregado a R2 dos veces. Sucedió la primera vez, cuando R1 contenía 0000 0101, y la tercera vez, después de que R1 se había desplazado a la izquierda dos veces y, por lo tanto, contenía 0001 0100. R2 ahora contiene 0001 1001 binario, que es $16+8+1$, o 25 decimal, que es la respuesta correcta para 5 por 5. El bucle se repetirá 5 veces más hasta que el bit en R3 se desplace a la bandera de acarreo, pero el total no aumentará porque no hay más 1 en R0.

Este programa se llevará a cabo ocho veces. Comenzamos con 0000 0001 en R3. Cerca del final del programa, R3 se desplaza a la izquierda. Las primeras siete veces, no habrá acarreo, por lo que el programa llegará al 'JMP 53' y volverá a la tercera instrucción del programa. La octava vez que R3 se desplaza a la izquierda, la única parte que está encendida se desplaza de R3 a la bandera de acarreo. Por lo tanto, el 'JC 68' saltará sobre el 'JMP 53' y continuará con las instrucciones que vengan después de esto.

El byte en R0 se desplaza hacia la derecha para probar qué bits están activados. El byte en R1 se desplaza hacia la izquierda para multiplicarlo por dos. Cuando había un bit en R0, agregas R1 a R2. Y eso es todo.

Una cosa que no abordamos en este ejemplo es lo que sucede si la respuesta de la multiplicación es más de 255. Si un programa de multiplicación multiplica dos números de un byte, debería ser capaz de manejar una respuesta de dos bytes. Eso se encargaría de dos números con los que podría comenzar. Esto se lograría con la bandera de acarreo y algunas instrucciones más de Jump If. No torturaremos al lector con los detalles.

Leer un programa como el anterior es una habilidad completamente diferente a leer los diagramas y gráficos que hemos visto hasta ahora en el libro. Espero que hayas podido seguirlo, pero no se espera que nadie se convierta en un experto en programas de lectura debido a este libro.

La división también se puede hacer por nuestra

computadora. Hay varias formas de hacerlo, y no vamos a hacerlo

examina cualquiera de ellos en detalle. Imagínese el siguiente método simple. Digamos que quieres dividir quince por tres. Si restas repetidamente tres de quince y cuentas el número de restas que puedes lograr antes de que se acaben las quince, esa cuenta será la respuesta. Como estos cinco pasos: (1) $15-3=12$, (2) $12-3=9$, (3) $9-3=6$, (4) $6-3=3$, (5) $3-3=0$. Esto es fácil convertido en un programa.

Las computadoras también tienen formas de manejar números negativos y números con puntos decimales. Los detalles son muy tediosos y estudiarlos no mejoraría nuestra comprensión de cómo funcionan las computadoras. Todavía se reduce a nada más que puertas NAND. Nuestra computadora simple podría hacer todas estas cosas con programas.

El mundo exterior

Lo que hemos descrito hasta ahora es toda la computadora. Tiene dos partes, la RAM y la CPU. Eso es todo lo que hay. Estas operaciones simples son las cosas más complicadas que puede hacer una computadora. La capacidad de ejecutar instrucciones, modificar bytes con la ALU, la capacidad de saltar de una parte del programa a otra y, lo más importante, la capacidad de saltar o no saltar en función del resultado de un cálculo. Esto es lo que una computadora es capaz de hacer. Estas son cosas simples, pero dado que funciona tan rápido, puede realizar una gran cantidad de estas operaciones que pueden resultar en algo que se ve impresionante.

Estas dos partes lo convierten en una computadora, pero si todo lo que la computadora pudiera hacer es ejecutar un programa y reorganizar los bytes en la RAM, nadie sabría lo que está haciendo. Así que hay una cosa más que la computadora necesita para ser útil, y es una forma de comunicarse con el mundo exterior.

Tratar con cualquier cosa fuera de la computadora se llama 'Entrada/Salida' o 'E/S' para abreviar. Salida significa datos que salen de la computadora; Entrada significa datos que ingresan a la computadora. Algunas cosas son solo de entrada, como un teclado, algunas cosas son solo de salida, como una pantalla de visualización, algunas cosas hacen tanto entrada como salida, como un disco.

Todo lo que necesitamos para E/S son unos pocos cables y una nueva instrucción.

Para los cables, todo lo que vamos a hacer es extender el bus de la CPU fuera de la computadora y agregar cuatro cables más para acompañarlo. Esta combinación de 12 cables se llamará bus de E/S. Todo lo que está conectado a la computadora está conectado a este bus de E/S.

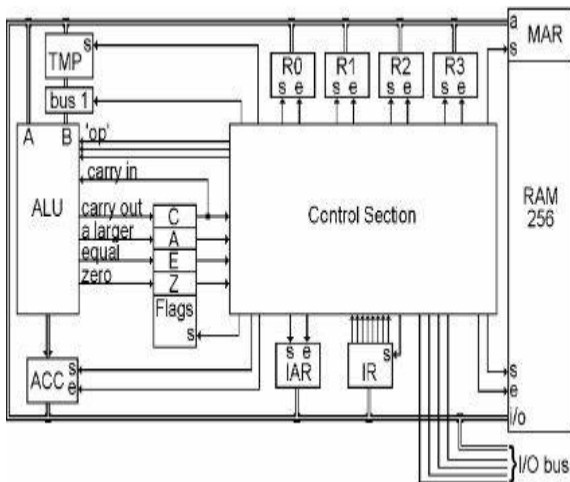
Los dispositivos que están conectados al bus de E/S se denominan "periféricos" porque no están dentro de la computadora, están fuera de la computadora, en su periferia (el área que la rodea).

Se puede conectar más de una cosa al bus de E/S, pero la computadora controla el proceso y solo una de estas

cosas está activa a la vez.

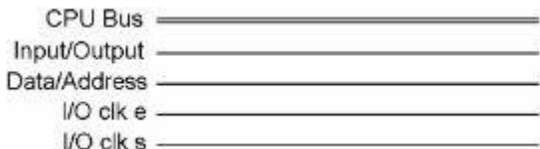
Cada cosa conectada al bus de E/S tiene que tener su propia dirección de E/S única. Esto no es lo mismo que las direcciones de los bytes en la RAM, es solo un 'número' que el periférico reconocerá cuando se coloque en el bus.

Así es como se ve el bus de E / S en la CPU, en la parte inferior derecha del dibujo.



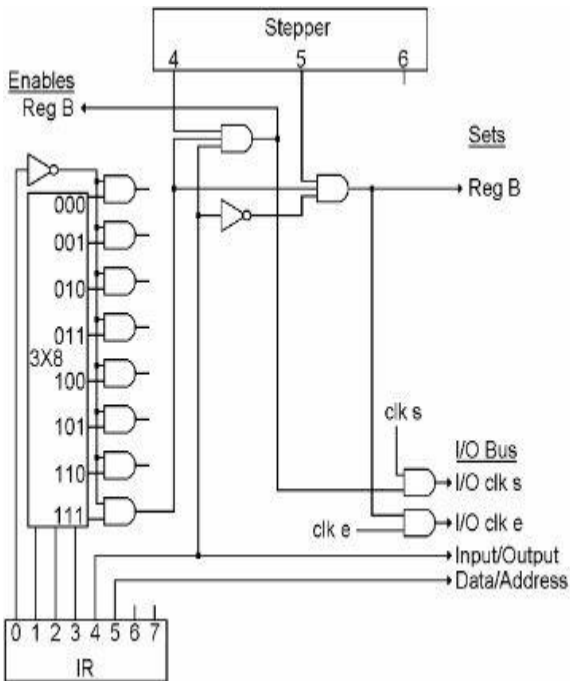
En el siguiente diagrama se muestran los cables del bus de E/S. El bus de CPU es el mismo paquete de ocho cables que va a todas partes. El cable 'Entrada/Salida' determina en qué dirección se moverán los datos en el bus de la CPU, ya sea hacia adentro o hacia afuera. El cable 'Datos/Dirección' nos dice si vamos a transferir un byte de datos, o un

Dirección de E/S que selecciona uno de los muchos dispositivos que se pueden conectar al bus de E/S. 'I/O Clk e' y 'I/O Clk s' se utilizan para habilitar y establecer registros para que los bytes se puedan mover



hacia adelante y hacia atrás.

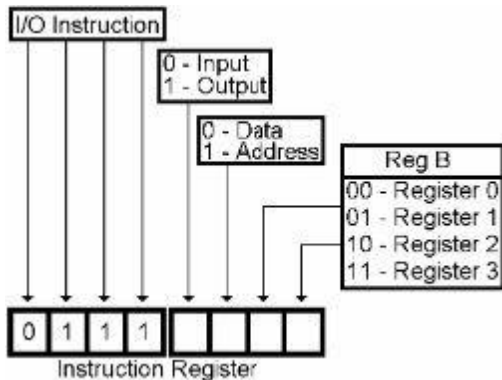
Aquí está el cableado de la sección de control para la nueva instrucción que controla el bus de E/S. Esto muestra de dónde provienen los cuatro nuevos cables para el bus de E/S. Están en la parte inferior derecha del dibujo. También se mostraron en el diagrama de la sección de control total unos capítulos atrás. Lo siento si eso fue confuso, pero tener ese diagrama en el libro una vez fue suficiente.



Los bits IR 4 y 5 se colocan en el bus de E/S en todo momento.

Para que la operación de E/S suceda, solo se necesita un paso. Para Salida, Reg B está habilitado y E/S Clk se activa y desactiva durante el paso 4. Los pasos 5 y 6 no hacen nada. Para Entrada, Clave de E/S está habilitada y Reg B se establece durante el paso 5. Los pasos 4 y 6 no hacen nada.

Aquí está el código de instrucción para la instrucción de E



/ S:

Esta instrucción se puede usar de cuatro maneras diferentes dependiendo de los bits IR 4 y 5, y por lo tanto hay cuatro palabras nuevas para nuestro idioma.

Idioma		Significado
EN	Datos, R B	Entrada de datos de E/S a RB
EN	Dirección, RB	Dirección de E/S de entrada a RB
FUERA	Datos, R B	Salida de RB a E/S como datos

FUERA	Dirección, RB	Salida RB a E/S como dirección
-------	---------------	--------------------------------

Cada dispositivo de E/S tiene sus propias características únicas, y

por lo tanto, necesita piezas y cableado únicos para conectarlo al bus de E / S. La colección de piezas que conecta el dispositivo al bus se denomina "adaptador de dispositivo". Cada tipo de adaptador tiene un nombre específico, como el "adaptador de teclado" o el "adaptador de disco".

El adaptador no hace nada a menos que su dirección aparezca en el bus. Cuando lo haga, el adaptador responderá a los comandos que le envíe el equipo.

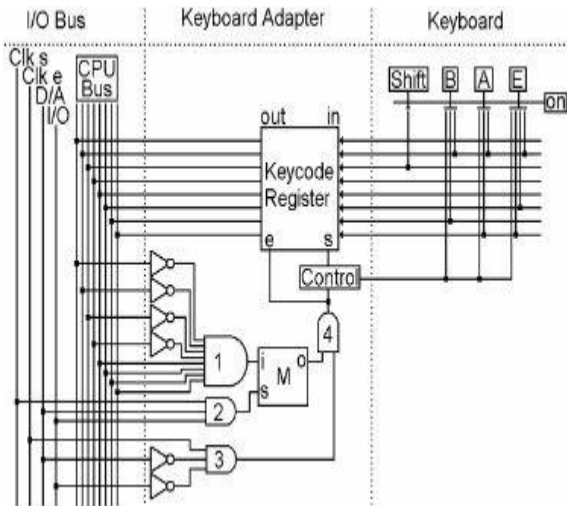
Con una instrucción 'OUT Addr', la computadora enciende el cable de dirección y coloca la dirección del dispositivo con el que desea hablar en el bus de la CPU. El periférico reconoce su dirección y cobra vida. Todos los demás periféricos tienen alguna otra dirección, por lo que no responderán.

No vamos a describir todas las puertas del sistema de E/S. En este momento, debería creer que los bytes de información se pueden transferir a través de un bus con unos pocos cables de control. El mensaje de este capítulo es solo la simplicidad del sistema de E/S. La CPU y la RAM son la computadora. Todo lo demás, discos, impresoras, teclados, el mouse, la pantalla, las cosas que hacen sonido, las cosas que se conectan a Internet, todas estas cosas son periféricos, y todo lo que son capaces de hacer es aceptar bytes de datos de la computadora o enviar bytes de datos a la computadora. Los adaptadores para diferentes dispositivos tienen diferentes capacidades, diferentes números de registros y diferentes requisitos en cuanto a lo que debe hacer el programa que se ejecuta en la CPU para operar el dispositivo correctamente. Pero no hacen nada más elegante que eso. La computadora controla el proceso con muy pocos comandos de E / S simples que son ejecutados por la CPU.

El teclado

Un teclado es uno de los periféricos más simples conectados al bus de E/S. Es un dispositivo de solo entrada y solo presenta un byte a la vez a la CPU.

El teclado tiene ocho cables en su interior, su propio pequeño autobús como se muestra a la derecha. Cuando presiona una tecla, simplemente conecta la electricidad a los cables necesarios para crear el código ASCII correspondiente a la tecla que se presionó. Ese pequeño cuadro que dice 'Control' también recibe una notificación cuando se presiona una tecla y establece el código ASCII en el registro de códigos clave.



Después de presionar una tecla, habrá un código ASCII esperando en el registro de códigos clave. Así es como la CPU introduce ese código en uno de sus registros.

Y la puerta #1 tiene ocho entradas. Están conectados al bus de la CPU, cuatro de ellos a través de puertas NOT. Por lo tanto, esta puerta AND se encenderá cada vez que el bus contenga 0000 1111. Esta es la dirección de E/S de este adaptador de teclado.

Y la puerta #2 se enciende solo durante el tiempo de 'clk s' de una instrucción OUT Addr. Opera la entrada 'set' de un bit de memoria. Si el bus contiene 0000 1111 en este momento, la entrada 'i' estará activada y el bit

de memória girará

en. Cuando este bit de memoria está encendido, significa que el adaptador de teclado está activo.

Y la puerta #3 se enciende durante el tiempo 'clk e' de una instrucción de datos IN. Si el bit de memoria está activado, la puerta AND #4 se activará y el registro de código clave se habilitará en el bus, que se establecerá en Reg B en la CPU.

Cada adaptador que está conectado al bus de E / S debe tener el tipo de circuito que vemos en las puertas # 1 y # 2 y el bit de memoria anterior. Cada adaptador tendrá una combinación diferente que enciende la puerta #1; esto es lo que permite a la CPU seleccionar cada adaptador individualmente.

Aquí hay un pequeño programa que mueve la pulsación de tecla actual a Reg 3 en la CPU.

Instrucción		Comentarios
Datos	R2,0000 1111	* Poner la dirección del teclado en
FUERA	Dirección,R2	* Seleccionar teclado
EN	Datos, R3	* Obtenga ASCII de la pulsación de teclas
XOR	R2, R2	* Borrar dirección en Reg 2
FUERA	Dirección,R2	* Anular la selección del teclado

Ese pequeño cuadro de 'Control' borra el registro de código clave después de que se haya enviado a la CPU.

El programa que se ejecuta en la CPU verificará el adaptador de teclado de forma regular, y si el byte que recibe es todo ceros, entonces no se ha presionado ninguna tecla. Si el byte tiene uno o más bits, entonces el programa hará lo que el programa haya sido diseñado para hacer con una pulsación de tecla en ese momento.

Una vez más, no vamos a pasar por todas las puertas del adaptador de teclado. Todos los adaptadores de dispositivos tienen los mismos tipos de circuitos para poder responder cuando se direccionan y enviar o recibir bytes de información según sea necesario. Pero no es más complicado que eso. Eso es todo lo que hacen los

dispositivos y adaptadores de E/S.

La pantalla de visualización

Las pantallas de visualización de televisión y computadora funcionan de la misma manera, la principal diferencia entre ellas es solo lo que muestran. En realidad, esto no es tecnología informática, porque no necesita una pantalla de visualización para tener una computadora, pero la mayoría de las computadoras tienen una pantalla, y la computadora pasa gran parte de su tiempo haciendo que la pantalla se vea como algo, por lo que necesitamos saber un poco sobre cómo funciona.

La televisión parece darte imágenes en movimiento con sonido. Las imágenes y el sonido se hacen por separado, y en este capítulo, solo nos interesa cómo funciona la imagen.

Lo primero que debe saber es que, aunque la imagen parece estar en movimiento, en realidad es una serie de imágenes fijas presentadas tan rápidamente que el ojo no lo nota. Probablemente ya lo sabías, pero aquí está lo siguiente. Has visto películas. Es una serie de imágenes. Para ver una película, pones la película en un proyector, que ilumina una imagen, luego mueve la película a la siguiente imagen, brilla luz a través de ella, etc. Por lo general, funciona a 24 imágenes por segundo, lo que es lo suficientemente rápido como para dar la ilusión de una imagen en constante movimiento.

La televisión va un poco más rápido, alrededor de 30 imágenes por segundo, pero hay otra diferencia mucho mayor entre el cine y la televisión. Con la película de película, cada imagen fija se muestra de una sola vez. Cada imagen está completa, cuando haces brillar la luz a través de ella, cada parte de la imagen aparece en la pantalla simultáneamente. La televisión no es capaz de hacer esto. No tiene una imagen completa para poner en la pantalla de una sola vez.

Todo lo que un televisor puede hacer en un instante en el tiempo, es iluminar un solo punto en la pantalla. Ilumina un punto, luego otro punto, luego otro, muy rápidamente hasta que se ha encendido una imagen completa de puntos.

El valor de los puntos de toda esta pantalla forma una imagen fija, por lo que tiene que iluminar todos los

puntos en una trigésima parte de un segundo y luego hacerlo todo de nuevo

con la siguiente imagen, etc. hasta que haya colocado 30 puntos de imagen en la pantalla en un segundo. Así que el televisor está muy ocupado iluminando puntos individuales, 30 veces el número de puntos en la pantalla, cada segundo.

Por lo general, el punto superior izquierdo se ilumina primero, luego el de la derecha, y así sucesivamente en la parte superior de la pantalla hasta la esquina superior derecha. Luego comienza con la segunda línea de puntos, recorriendo la pantalla nuevamente, la tercera línea, etc. hasta que haya escaneado toda la pantalla. El brillo de cada punto es alto o bajo, de modo que cada parte de la pantalla se ilumina con el brillo adecuado para que la pantalla se parezca a la imagen deseada.

En cualquier instante en el tiempo, la televisión solo está lidiando con un solo punto solitario en la pantalla. Entonces, con la televisión, hay dos ilusiones: la ilusión de movimiento que proviene de una serie de imágenes fijas, así como la ilusión de imágenes fijas completas que en realidad se dibujan un punto a la vez. Esta segunda ilusión se ve favorecida por el hecho de que está hecha la pantalla, cada punto solo se ilumina durante una pequeña fracción de segundo y comienza a desvanecerse de inmediato. Afortunadamente, sea lo que sea que esté hecha la pantalla que brilla, continúa brillando hasta cierto punto entre un momento en que se ilumina el punto y $1/30$ de segundo más tarde cuando ese mismo punto se ilumina nuevamente.

A simple vista, solo ves una imagen en movimiento, pero hay muchas cosas que suceden para que parezca así.

En una computadora, un solo punto en la pantalla se llama "elemento de imagen" o "píxel" para abreviar.

Las pantallas de las computadoras funcionan igual que los televisores. También tienen que escanear toda la pantalla 30 veces por segundo para iluminar cada píxel individual y, por lo tanto, hacer que aparezca una imagen. Incluso si el contenido de la pantalla no cambia, algo en la computadora tiene que escanear esa imagen inmutable en la pantalla 30 veces por segundo. Sin escaneo, sin imagen, así es como funciona.

No vamos a entrar en la misma cantidad de detalles aquí que hicimos con la CPU y la RAM, esos dos son los que la convierten en una computadora, pero si queremos saber

cómo nuestra computadora puede poner algo en la pantalla que podamos leer, debemos tener la idea básica de cómo

Obras.

En este capítulo veremos el tipo de pantalla más simple, el tipo que es en blanco y negro, y cuyos píxeles solo pueden estar completamente encendidos o completamente apagados. Este tipo de pantalla puede mostrar caracteres y el tipo de imágenes que están hechas de dibujos lineales. Más adelante en el libro veremos los pocos cambios simples que permiten que una pantalla muestre cosas como fotografías en color.

Las partes principales son tres. Primero está la computadora, hemos visto cómo funciona. Tiene un bus de E / S que puede mover bytes hacia y desde cosas fuera de la computadora. En segundo lugar está la pantalla. La pantalla es solo una gran cuadrícula de píxeles, cada uno de los cuales se puede seleccionar, uno a la vez, y mientras está seleccionado, se puede encender o no. El tercer elemento es el 'adaptador de pantalla'. El adaptador de pantalla está conectado al bus de E/S por un lado y a la pantalla por el otro.

El corazón de un adaptador de pantalla es algo de RAM. El adaptador de pantalla necesita su propia RAM para poder "recordar" qué píxeles deben estar encendidos y qué píxeles deben estar apagados. En el tipo de pantalla que vamos a describir aquí, debe haber un bit de RAM por cada píxel en la pantalla.

Para que la pantalla escanee cada píxel 30 veces por segundo, el adaptador de pantalla necesita su propio reloj que marque a una velocidad 30 veces mayor que la cantidad de píxeles en la pantalla. En cada tictac del reloj, se selecciona un píxel y se enciende o no por el bit correspondiente de la RAM.

Como ejemplo, usemos un tipo de pantalla antiguo. Es una pantalla en blanco y negro que muestra 320 píxeles en la pantalla y 200 píxeles hacia abajo. Eso equivale a 64,000 píxeles individuales en la pantalla. Cada píxel en la pantalla tiene una dirección única que consta de dos números, el primero es la posición izquierda-derecha u horizontal, y el otro es la posición arriba-abajo o vertical.

La dirección del píxel superior izquierdo es 0,0 y el píxel inferior derecho es 319,199

64,000 píxeles por 30 imágenes por segundo significa que el reloj de este adaptador de pantalla debe marcar

1,920,000 veces por segundo. Y dado que hay ocho bits en un byte, necesitaremos 8,000 bytes de RAM de pantalla para decirle a cada uno de los 64,000 píxeles de la pantalla si debe estar encendido o

apagado.

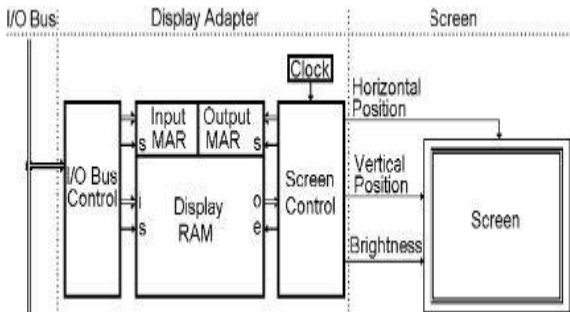
El adaptador de pantalla tiene un registro que establece la posición horizontal del píxel actual. El adaptador de pantalla agrega 1 a este registro en cada tictac del reloj. Comienza en cero, y cuando el número en él llega a 319, el siguiente paso lo restablece de nuevo a cero. Así que va de cero a 319 una y otra vez. También hay un registro que establece la posición vertical del píxel actual. Cada vez que el registro horizontal se restablece a cero, el adaptador de pantalla agrega 1 al registro vertical. Cuando el registro vertical llegue a 199, el siguiente paso lo restablecerá a cero. Entonces, como el registro horizontal va de cero a 319 200 veces, el registro vertical va de cero a 199 una vez.

El píxel de la pantalla seleccionado actualmente está controlado por estos registros, de modo que a medida que el registro horizontal pasa de

De 0 a 319, el píxel actual pasa por la pantalla una vez. Luego, al registro vertical se le agrega uno y el píxel actual se mueve hacia abajo hasta el primer píxel en la siguiente línea.

Por lo tanto, el reloj y los registros horizontal y vertical seleccionan cada píxel en la pantalla, uno a la vez, yendo de izquierda a derecha en una fila, luego seleccionando cada píxel en la siguiente fila hacia abajo, luego en la siguiente, etc. hasta que cada píxel en la pantalla se haya seleccionado una vez. Luego comienza todo de nuevo.

Al mismo tiempo, hay otro registro que contiene una dirección RAM de visualización. Este registro también se recorre, aunque solo necesitamos un nuevo byte por cada ocho píxeles. Los bits de cada byte, uno a la vez, se envían a la pantalla en ocho píxeles consecutivos para activarlos o desactivarlos. Después de cada ocho píxeles, el registro de direcciones RAM tiene 1 agregado. Para cuando se han pasado todos los píxeles, también se ha pasado por toda la RAM y se ha dibujado una imagen completa. Cuando los registros horizontal y vertical han alcanzado sus máximos y se restablecen a cero, la dirección RAM también se restablece a cero.



El adaptador de pantalla pasa la mayor parte de su tiempo pintando la pantalla. La única otra cosa que tiene que hacer es aceptar comandos del bus de E / S que cambiarán el contenido de la RAM del adaptador de pantalla. Cuando el programa que se ejecuta en la CPU necesita cambiar lo que está en la pantalla, usará el comando I/O OUT para seleccionar el adaptador de pantalla y luego enviará una dirección RAM del adaptador de pantalla y luego un byte de datos para almacenar en esa dirección. Luego, a medida que el adaptador continúe repintando la pantalla, los nuevos datos aparecerán en la pantalla en el lugar apropiado.

La RAM del adaptador de pantalla está construida de manera diferente a la RAM de nuestra computadora. Mantiene separadas las funciones de entrada y salida. Las entradas de todas las ubicaciones de almacenamiento están conectadas al bus de entrada y las salidas de todas las ubicaciones de almacenamiento están conectadas al bus de salida, pero el bus de entrada y el bus de salida se mantienen separados. Luego hay dos registros de direcciones de memoria separados, uno para entrada y otro para salida. El MAR de entrada tiene una cuadrícula que solo selecciona qué byte se 'establecerá' y el MAR de salida tiene una cuadrícula separada que solo selecciona qué byte se 'habilitará'.

Con esta configuración, la pantalla y la RAM de la pantalla se pueden escanear continuamente utilizando solo la salida MAR y

el bit de habilitación. Cuando se utiliza el bus de E/S para escribir en la RAM de la pantalla, sólo utiliza el MAR de entrada y el bit de ajuste.

Así es como el adaptador de pantalla crea una imagen en la pantalla. Debido a la forma en que funciona, existe una relación interesante entre qué bits en la RAM de la pantalla corresponden a qué píxeles en la pantalla. A medida que escanea los primeros ocho píxeles de la línea superior, utiliza los bits individuales del byte 0 de su RAM para activar o desactivar los píxeles. A medida que escanea los segundos ocho píxeles, utiliza los bits individuales del byte 1 de su RAM, etc.

Se necesitan 40 bytes de RAM para dibujar la primera línea, por lo que los últimos ocho píxeles, que están numerados del 312 al 319, provienen del byte 39 de RAM. La segunda fila usa el byte 40 para dibujar sus primeros 8 píxeles, etc.

Si quieres escribir letras y números en la pantalla, ¿cómo lo haces? Si coloca el código ASCII para 'A' en un byte en la RAM de la pantalla, solo obtendrá ocho píxeles seguidos donde uno está apagado, luego uno está encendido, luego cinco están apagados y el último está encendido. Así no es como debería verse una 'A'.

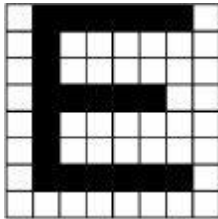
Hay una solución para esto, y se trata de...

Otro código

Cuando desee imprimir o mostrar lenguaje escrito, debe traducir el código ASCII a algo que pueda leer una persona en vivo. Tenemos un código, 0100 0101, que aparece en la tabla de códigos ASCII junto a la letra 'E'. Pero, ¿cómo convierte la computadora 0100 0101 en una 'E' legible?

Tenemos una pantalla de visualización, pero la pantalla es solo una cuadrícula de píxeles, no hay 'E' legibles por humanos en nada de lo que hemos descrito hasta ahora. Para que aparezca una 'E' en la pantalla, tiene que haber algo que haga esa forma que reconozcamos como una letra del alfabeto.

Por lo tanto, necesitamos otro código. Este código se trata realmente de pequeñas imágenes hechas de puntos. Para cada personaje que queremos poder dibujar en la pantalla, necesitamos una pequeña imagen de ese personaje. Si toma una cuadrícula de 8 píxeles de ancho y 8 píxeles de alto, podría decidir qué píxeles tenían que estar para hacer una pequeña imagen que se parezca al personaje que desea dibujar en la pantalla, como esta:



Si convierte esta imagen en encendidos y apagados, podría almacenarla en ocho bytes. Si hay 100 caracteres diferentes que desea poder mostrar en la pantalla, entonces necesitaría 100 pequeñas imágenes diferentes como esta, y requeriría 800 bytes de RAM para almacenarlas. Nuestra pequeña computadora solo tiene una memoria RAM de 256 bytes,

Así que este sería un buen momento para imaginar esa versión más grande que describimos anteriormente.

Estos 800 bytes son un tipo de código conocido como "fuente".

Si desea que un carácter aparezca en un lugar determinado de la pantalla, debe elegir la pequeña imagen correcta de la fuente y luego usar las instrucciones de E / S para copiar los ocho bytes de la imagen a los bytes adecuados en la RAM del adaptador de pantalla.

Si las imágenes de nuestra fuente están dispuestas en el mismo orden que la tabla de códigos ASCII, entonces podemos usar el valor numérico de un código ASCII para encontrar la imagen correspondiente dentro de la fuente. El código ASCII para 'E' es 0100 0101. Si aplica el código numérico binario al mismo patrón de unos y ceros, obtiene el número decimal

69. 'E' entonces, es el código 69 en ASCII, y la imagen de una 'E' será la imagen 69 dentro de la fuente. Como hay ocho bytes en cada imagen, multiplica el 69 por 8, y eso le dice que la imagen para 'E' serán los ocho bytes que comienzan en la dirección 552.

Ahora necesitamos saber dónde copiar estos bytes en la RAM de la pantalla. Digamos que queremos mostrar una 'E' en la parte superior izquierda de la pantalla. ¿Dónde están los bits que encienden los píxeles que nos interesan? Bueno, la primera línea es fácil, son los primeros ocho bits de la RAM de la pantalla, Dirección 0. Por lo tanto, usamos una serie de instrucciones OUT para copiar la dirección RAM 552 para mostrar la dirección RAM 0. Ahora, ¿dónde está la segunda línea en la RAM de la pantalla? La pantalla pinta los 320 bits de la fila superior antes de pasar a la segunda fila. Eso significa que usa 40 bytes en cada fila, por lo que la fila superior usa bytes 0-39. Eso significa que el segundo byte de la imagen de 'E' en la dirección RAM 553 debe escribirse en la dirección 40 en la RAM de la pantalla. De manera similar, los bytes tercero a octavo se escriben en los bytes 80, 120, 160, 200, 240 y 280. Cuando haya hecho todo eso, verá una 'E' completa en la pantalla. Si quisieras escribir una 'X' en la pantalla justo al lado de la 'E', ubicarías los ocho bytes en la fuente para 'X' y los copiarías en los bytes de RAM de la pantalla 1, 41, 81, 121, 161, 201, 241 y

281. Si necesita 27 'E' en su pantalla, simplemente copie la 'E' en su fuente en 27 lugares diferentes en el

pantalla RAM.

RAM Address	Byte Contents	Display RAM Address	Screen Pixels	Screen
552	01111110	000		
553	01000000	040		
554	01000000	080		
555	01111100	120		
556	01000000	160		
557	01000000	200		
558	01111110	240		
559	00000000	280		

Por supuesto, esto parece mucho trabajo solo para hacer que aparezca una sola letra en la pantalla. El programa que hace esto necesitaría un bucle de instrucciones que calcule las primeras direcciones 'de' y 'hasta', luego emita las instrucciones OUT apropiadas para copiar el primer byte a la RAM de la pantalla. Luego, el bucle se repetiría, actualizando ambas direcciones cada vez, hasta que los ocho bytes se hubieran copiado en los lugares apropiados. No vamos a escribir este programa, pero fácilmente podría ser un programa de 50 instrucciones que tiene que dar ocho vueltas antes de terminar. ¡Eso significa que podría tomar 400 ciclos de instrucción solo para poner un personaje en la pantalla! Si dibujó 1000 caracteres en la pantalla, eso podría tomar 400,000 ciclos de instrucción. Por otro lado, eso sigue siendo solo un cuarto del uno por ciento de lo que esta computadora puede hacer en un segundo.

Esto solo demuestra por qué las computadoras deben ser tan rápidas. Las cosas individuales que hacen son tan pequeñas que se necesita una gran cantidad de pasos para hacer algo.

La última palabra sobre los códigos

Hemos visto varios códigos utilizados en nuestra computadora. Cada uno fue diseñado para un propósito específico. Los mensajes codificados individuales se colocan en bytes, se mueven y se usan para hacer las cosas.

Los bytes no "saben" qué código se usó para elegir el patrón que contienen. No hay nada en el propio byte que te diga qué código se supone que es.

Ciertas partes de la computadora están construidas con varios códigos en mente. En la ALU, el sumador y el comparador están diseñados para tratar los bytes como si contuvieran valores codificados con el código numérico binario. También lo son el registro de direcciones de memoria y el registro de direcciones de instrucciones.

El registro de instrucciones está diseñado para tratar su contenido como si contuviera valores codificados con el código de instrucciones.

Los bits de RAM del adaptador de pantalla son solo encendidos o apagados para píxeles individuales. Las imágenes y las fuentes son cadenas de bytes que darán como resultado algo que una persona puede reconocer cuando se organiza, y los brillos se establecen, mediante el cableado de un adaptador de pantalla y una pantalla.

La tabla de códigos ASCII no aparece en ninguna parte dentro de la computadora porque no hay forma de representar una letra del alfabeto excepto mediante el uso de un código.

Los únicos lugares donde ASCII se convierte entre caracteres y el código del carácter, son en los periféricos. Cuando presiona 'E' en el teclado, obtiene el código ASCII para una 'E'. Cuando envía el código ASCII para una 'E' a una impresora, imprime la letra 'E'. Las personas que construyen estos periféricos tienen una tabla de códigos ASCII frente a ellos, y cuando construyen un teclado, el interruptor debajo del cuarto botón en la segunda fila, que tiene la letra 'E' impresa en él, está conectado a los cables de bus adecuados para producir el código que aparece junto a la letra 'E' en la tabla de códigos ASCII.

Una 'E' es la quinta letra de un alfabeto utilizado por las personas para representar sonidos y palabras en el proceso de escritura

su idioma hablado. Las únicas 'E' en la computadora son las del teclado y las que aparecen en la pantalla. Todas las 'E' que están en bytes son solo el código que aparece junto a la 'E' en una tabla de códigos ASCII. No son 'E', no hay forma de poner una 'E' en una computadora. Incluso si pones una imagen de una 'E' en una computadora, en realidad no es una 'E' hasta que se muestra en la pantalla. Ahí es cuando una persona puede mirarlo y decir "Eso es una E".

Los bytes son tontos. Solo contienen patrones de encendidos y apagados. Si un byte contiene 0100 0101 y lo envía a la impresora, imprimirá la letra 'E'. Si lo envía al Registro de instrucciones, la computadora ejecutará una instrucción de salto. Si lo envía al Registro de direcciones de memoria, seleccionará el byte número 69 de la RAM. Si lo envía a un lado de la víbora, agregará 69 a lo que esté al otro lado de la víbora. Si lo envía a la pantalla, activará y desactivará tres píxeles.

Cada una de estas piezas de la computadora está diseñada con un código en mente, pero una vez que se construye, la mente se va e incluso el código se va. Simplemente hace lo que fue diseñado para hacer.

No hay límite para los códigos que se pueden inventar y usar en una computadora. Los programadores inventan nuevos códigos todo el tiempo. Al igual que la caja registradora en el restaurante de comida rápida mencionado anteriormente, en algún lugar de esa máquina hay un poco que significa 'incluir papas fritas'.

El disco

La mayoría de las computadoras tienen un disco. Este es simplemente otro periférico que está conectado al bus de E/S. La misión del disco es muy simple; puede hacer dos cosas. Puede enviarle bytes, que almacenará, o puede decirle que devuelva algunos bytes, que se almacenaron anteriormente.

Hay dos razones por las que la mayoría de las computadoras tienen un disco. Primero, tienen la capacidad de almacenar una gran cantidad de bytes, muchas veces mayor que la RAM de la computadora. La CPU solo puede ejecutar programas que están en la RAM, solo puede manipular bytes que están en la RAM. Pero nunca hay suficiente RAM para almacenar todas las cosas que puede querer hacer con su computadora. Y así, un disco contendrá todo, y cuando quieras hacer una cosa, los bytes en el disco para esa cosa se copiarán en la RAM y se usarán. Luego, cuando desee hacer algo diferente, los bytes de la nueva actividad se copiarán del disco a la misma área de RAM que se usó para la primera actividad.

La segunda razón por la que las computadoras tienen discos es que los bytes almacenados en el disco no desaparecen cuando se apaga la alimentación. La RAM pierde su configuración cuando apaga la computadora, cuando la vuelve a encender, todos los bytes son 0000 0000, pero el disco conserva todo lo que se ha escrito en él.

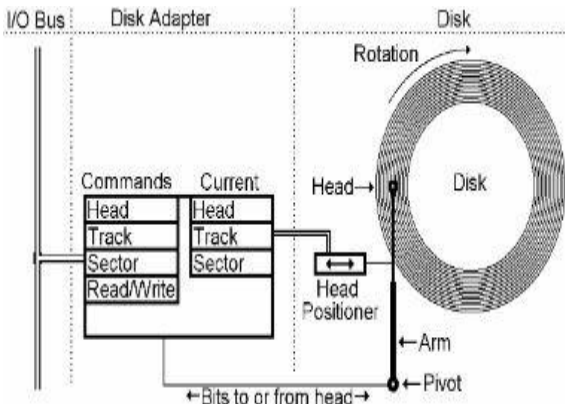
Hasta ahora, un bit de computadora se ha definido como un lugar donde hay o no electricidad. Pero antes de eso, lo definimos como un lugar que puede estar en uno de dos estados diferentes. En un disco, los bits eléctricos se transforman en lugares en la superficie del disco que se han magnetizado de una forma u otra. Dado que los imanes tienen polos norte y sur, el punto en el disco se puede magnetizar de norte a sur o de sur a norte. Una dirección representaría un cero, y la otra dirección, un uno. Una vez que un punto se magnetiza, permanece así a menos que el mismo punto se magnetice en la otra dirección. Apagar la alimentación no tiene ningún efecto sobre los puntos magnetizados.

Un disco, como su nombre lo indica, es una cosa redonda,

que gira rápidamente. Está recubierto con un material
que

se puede magnetizar fácilmente. ¿Recuerdas el telégrafo? En el extremo receptor, hay una pieza de metal con un alambre envuelto alrededor. Esa pieza de metal se convierte en un imán cuando la electricidad se mueve a través del cable. El disco tiene una versión diminuta de esto llamada "cabeza" montada en un brazo. El brazo sostiene la cabeza muy cerca de la superficie del disco giratorio, y el brazo puede balancearse hacia adelante y hacia atrás, de modo que la cabeza puede llegar a cualquier punto de la superficie del disco. Si pones electricidad a través de la cabeza, puede magnetizar la superficie del disco. Además, funciona al revés; Cuando la cabeza pasa sobre un área magnetizada, hace que aparezca electricidad en los cables que envuelven la cabeza. Por lo tanto, el cabezal puede escribir en el disco o leer lo que se ha escrito previamente en el disco. Los bits de los bytes se escriben uno tras otro en la superficie del disco.

La superficie del disco se divide en una serie de anillos, llamados pistas, muy cerca unas de otras. La cabeza puede moverse por la superficie y detenerse en cualquiera de las pistas. Cada pista circular generalmente se divide en partes cortas llamadas sectores. Dado que un disco tiene dos lados, generalmente ambos lados están recubiertos con el material magnético y hay una cabeza en cada lado.



En la RAM, cada byte tiene su propia dirección. En un disco, también hay una forma de localizar bytes, pero es muy diferente. Debe especificar qué encabezado, qué pista y en qué sector se encuentra un bloque de bytes. Ese es el tipo de "dirección" que tienen los datos en un disco, como "Cabeza 0, Pista 57, Sector 15". Y en esa dirección, no hay solo un byte, sino un bloque de bytes, generalmente varios miles. Para los ejemplos de nuestro libro, dado que nuestra RAM es tan pequeña, hablaremos de un disco que almacena bloques de 100 bytes.

Cuando se lee o escribe un disco, no hay forma de acceder a un byte individual en el bloque de bytes. Todo el bloque debe transferirse a la RAM, trabajarse en la RAM y luego todo el bloque debe volver a escribirse en el disco.

El disco gira rápidamente, más rápido que ese ventilador en su escritorio; Muchos discos populares giran 7200 veces por minuto, lo que

es 120 veces por segundo. Eso es bastante rápido, pero en comparación con la CPU, sigue siendo bastante lento. En el tiempo que el disco gira una vez, el reloj marcará más de ocho millones de veces y nuestra CPU ejecutará más de un millón de instrucciones.

El disco, como todos los periféricos, está conectado a su propio adaptador, que a su vez está conectado al bus de E/S. El adaptador de disco hace algunas cosas. Acepta comandos para seleccionar un cabezal, seleccionar una pista y seleccionar un sector. Acepta comandos para leer o escribir en el bloque de bytes en el encabezado, la pista y el sector seleccionados actualmente. También es probable que haya un comando en el que la CPU pueda verificar la posición actual del brazo y el disco.

El comando para seleccionar un cabezal se puede completar inmediatamente, pero cuando recibe un comando para seleccionar una pista, tiene que mover el cabezal a esa pista, lo que lleva mucho tiempo en términos de ciclos de instrucción. Cuando recibe un comando para seleccionar un sector, tiene que esperar a que ese sector gire hacia donde está la cabeza, lo que también lleva mucho tiempo en términos de ciclos de instrucción. Cuando la CPU ha determinado que el cabezal ha llegado a la pista y al sector deseados, se pueden ejecutar los comandos de E/S para leer o escribir, y se transferirá un byte a la vez a través del bus de E/S. Un programa que lee o escribe un bloque de bytes tiene que continuar el proceso hasta que se complete todo el bloque de bytes. Con nuestro sencillo sistema de E/S, los bytes individuales se mueven entre el disco y un registro de CPU. El programa que se está ejecutando tiene que mover estos bytes hacia o desde la RAM, generalmente en ubicaciones consecutivas.

Esto es todo lo que hace un disco. Probablemente haya usado una computadora que tenía un disco y no necesitaba saber nada sobre cabezales, pistas y sectores. Y eso es algo bueno, porque es bastante molesto tener que lidiar con un disco con ese nivel de detalle. Veremos cómo se usa normalmente un disco más adelante en el libro.

Otra nota de lenguaje: hay varias palabras que significan prácticamente lo mismo, pero por alguna razón ciertas palabras van con ciertas tecnologías.

Si quieres enviar una carta a alguien, primero la escribes en una hoja de papel, luego, cuando el

destinatario recibe la carta, la lee.

En los días de las grabadoras, comenzarías con una cinta en blanco. Luego grababas algo de música en la cinta. Cuando querías volver a escuchar la música, ponías la cinta.

Cuando se trata de discos de computadora, poner algo en el disco se llama escritura. Sacar algo del disco se llama lectura.

Poner algo en la RAM se llama escribir o almacenar.

Obtener algo de la RAM se llama leer o recuperar.

Poner algo en un registro de CPU generalmente se llama carga.

Poner música en un disco a veces se llama grabación, a veces grabación. Escuchar un disco todavía se llama reproducir, pero si lo está copiando en su computadora, entonces se llama ripeo.

Escribir, grabar, almacenar, cargar y grabar significan más o menos lo mismo. Leer, recuperar, tocar y rasgar también son muy similares. Significan las mismas cosas, es solo una diferencia de palabras.

Disculpe, señora

Hay otra cosa que la mayoría de las computadoras tienen como parte de su sistema de entrada / salida. Una computadora no necesita que uno de estos se llame computadora, por lo que no pasaremos por todas las puertas necesarias para construirla. Pero es algo muy común, por lo que describiremos cómo funciona.

Sabes que si mamá está en la cocina revolviendo una olla de sopa, y el pequeño Joey entra corriendo y dice "Quiero un vaso de leche", mamá dejará la cuchara, irá al gabinete, tomará un vaso, irá al refrigerador, servirá la leche, se la entregará a Joey y luego volverá a la estufa, Tome la cuchara y continúe revolviendo la sopa. La agitación de la sopa se interrumpió tomando un vaso de leche, y luego se reanudó la revuelta de la sopa.

Esta cosa que tienen la mayoría de las computadoras se llama "interrupción" y funciona de manera muy similar a lo que sucedió con mamá y Joey.

Una interrupción comienza con un cable más agregado al bus de E / S. Este cable es utilizado por ciertos adaptadores de dispositivos para que la CPU sepa que es un buen momento para que la CPU realice una operación de E / S, como justo después de que alguien presiona una tecla en el teclado. Cuando un adaptador de dispositivo activa el bit de interrupción, la próxima vez que el paso a paso vuelva al paso 1, el siguiente ciclo de instrucciones no realizará la búsqueda y ejecución habituales, sino que hará lo siguiente:

Paso 1	mover binario 0 a MAR
Paso 2	mover IAR a RAM
Paso 3	mover binario 1 a MAR
Paso 4	mover banderas a RAM
Paso 5	mover binario 2 a MAR
Paso 6	mover RAM a IAR
Paso 7	mover binario 3 a MAR
Paso 8	mover RAM a Banderas

El resultado de esta secuencia es que el IAR y los indicadores actuales se guardan en las direcciones RAM 0

y 1, y se reemplazan con el contenido de las direcciones de bytes RAM 2 y

3. Luego, la CPU vuelve a su búsqueda normal y se ejecuta

operación. ¡Pero el IAR ha sido reemplazado! Por lo tanto, la siguiente instrucción se obtendrá de cualquier dirección que estuviera en el byte 2 de RAM.

En otras palabras, lo que la CPU había estado haciendo se guarda y la CPU se envía a hacer otra cosa. Si al final de esta nueva actividad, el programa vuelve a colocar los bytes de RAM 0 y 1 en el IAR y los indicadores, la CPU continuará exactamente donde lo dejó, antes de que se interrumpiera.

Este sistema es muy útil para tratar operaciones de E/S. Sin interrupciones, el programa que se ejecuta en la CPU tendría que asegurarse de verificar todos los dispositivos en el bus de E / S de forma regular. Con las interrupciones, el programa puede hacer lo que esté diseñado para hacer, y el programa que se ocupa de cosas como la entrada del teclado será llamado automáticamente según sea necesario por el sistema de interrupción.

No hemos incluido esto en nuestra CPU porque solo haría que nuestro diagrama de cableado de la Sección de Control fuera demasiado grande. Necesitaría agregar lo siguiente: dos pasos más al paso a paso, cableado para realizar los 8 pasos anteriores en lugar del ciclo de instrucción normal, rutas para que el registro Flags llegue y salga del bus, un método para enviar un 0, 1, 2 o 3 binario a MAR, y una instrucción que restaure los bytes de RAM 0 y 1 al registro IAR y Flags.

Y ese es un sistema de interrupción. En lo que respecta al lenguaje, los diseñadores de computadoras tomaron un verbo existente, 'interrumpir', y lo usaron de tres maneras: es un verbo en "el teclado interrumpió el programa", es un adjetivo en "Este es el sistema de interrupción" y es un sustantivo en "la CPU ejecutó una interrupción".

Eso es todo amigos.

Sí, este es el final de nuestra descripción de una computadora. Esto es todo lo que hay. Todo lo que ve hacer una computadora es una larga concatenación de estas operaciones muy simples, las operaciones ADDing, NOTting, Shifting, ANDing, ORing, XORing de bytes, Sstore, Loading, Jumping y E / S, a través de la ejecución del código de instrucción desde la RAM. Esto es lo que hace que una computadora sea una computadora. Esta es la suma total de la inteligencia en una computadora. Este es todo el pensamiento que una computadora es capaz de hacer. Es una máquina que hace exactamente lo que está diseñada para hacer, y nada más. Como un martillo, es una herramienta ideada por el hombre para realizar tareas definidas por el hombre. Hace su tarea exactamente como fue diseñada. También como un martillo, si se lanza indiscriminadamente puede hacer algo impredecible y destructivo.

La variedad de cosas que se pueden hacer con la computadora está limitada solo por la imaginación y la inteligencia de las personas que crean los programas para que los ejecuten. Las personas que construyen las computadoras siguen haciéndolas más rápidas, más pequeñas, más baratas y más confiables.

Cuando pensamos en una computadora, probablemente pensamos en esa caja que se encuentra en un escritorio y tiene un teclado, un mouse, una pantalla y una impresora conectados. Pero las computadoras se usan en muchos lugares. Hay una computadora en su automóvil que controla el motor. Hay una computadora en tu teléfono celular. Hay una computadora en la mayoría de las cajas de televisión por cable o satélite. Las cosas que todos tienen en común son que todos tienen una CPU y RAM. Las diferencias están todas en los periféricos. Un teléfono celular tiene un teclado y una pantalla pequeños, un micrófono y un altavoz, y una radio bidireccional para periféricos. Su automóvil tiene varios sensores y controles en el motor y los diales del tablero para periféricos. La caja registradora de un restaurante de comida rápida tiene un teclado divertido, una pequeña pantalla y una pequeña impresora para recibos. Hay computadoras en algunos semáforos que cambian las luces según la hora del día y la

cantidad de tráfico que cruza los sensores incrustados en la carretera. Pero la CPU y la RAM lo convierten en una computadora, los periféricos pueden ser muy diferentes.

Para el resto del libro veremos temas diversos relacionados con la comprensión de cómo se usan las computadoras, algunas palabras interesantes relacionadas con las computadoras, algunas de sus debilidades y algunos otros cabos sueltos.

Hardware y software

Has oído hablar del hardware. Esa palabra ha existido durante mucho tiempo. Ha habido ferreterías durante un siglo o más. Creo que una ferretería originalmente vendía cosas que eran duras, como ollas y sartenes, destornilladores, palas, martillos, clavos, arados, etc. Quizás 'hardware' significaba cosas que estaban hechas de metal. Hoy en día, algunas ferreterías ya no venden ollas y sartenes, pero venden una gran variedad de cosas duras, como pernos y cortadoras de césped, también madera y muchas cosas blandas, como alfombras, papel tapiz, pintura, etc. Pero estas cosas blandas no se llaman software.

La palabra "software" se inventó en algún momento de los primeros días de la industria informática para diferenciar la computadora en sí del estado de los bits que contiene.

Software significa la forma en que los bits se activan o desactivan en una computadora en lugar de la computadora en sí. Recuerde que los bits pueden estar encendidos o apagados. El bit tiene una ubicación en el espacio, está hecho de algo, existe en el espacio, se puede ver. La parte es hardware. Si el bit está encendido o apagado es importante, pero no es una parte separada que se atornilla a la computadora, es lo que se puede cambiar en la computadora, lo que se puede moldear, es 'suave' en el sentido de que puede cambiar, pero no puedes recogerlo en tu mano por sí solo. Esta cosa se llama software.

Piense en una cinta de video en blanco. Luego graba una película en él. ¿Cuál es la diferencia entre la cinta de video en blanco y la misma cinta de video con una película? Se ve igual, pesa lo mismo, no se puede ver ninguna diferencia en la superficie de la cinta. Esa superficie está recubierta con partículas muy finas que se pueden magnetizar. En la cinta en blanco, toda la superficie de la cinta se magnetiza en direcciones aleatorias. Después de grabar la película en la cinta, algunos pequeños lugares de la cinta se magnetizan en una dirección y otros pequeños lugares se magnetizan en la otra dirección. No se agrega ni se quita nada de la cinta, es solo la forma en que se magnetizan las partículas magnéticas. Cuando pones la cinta en una

videograbadora, reproduce una película. La cinta es hardware, el patrón de las direcciones de magnetización en la cinta es software.

En una computadora, hay muchos bits. Como hemos visto, muchos bits deben configurarse de ciertas maneras para que la computadora haga algo útil. Los bits en la computadora siempre están ahí. Si desea que la computadora haga una determinada cosa, encienda o apague esos bits de acuerdo con el patrón que hará que la computadora haga lo que usted quiere que haga. Este patrón se llama software. No es algo físico, es solo el patrón en el que se establecen los bits.

Entonces, la diferencia entre hardware y software no es como el metal versus el caucho. Tanto el metal como el caucho son hardware en lo que respecta a la definición de la computadora. El hardware es algo que puedes recoger, ver, manejar.

El software es la forma en que se configura el hardware. Cuando compras software, se graba en algo, generalmente algún tipo de disco. El disco es hardware, el patrón específico grabado en ese disco es software. Otro disco puede parecerse a él, pero tener un software completamente diferente escrito en él.

Otra forma de ver la diferencia entre hardware y software es lo fácil que es enviarlo a distancia. Si tienes un jarrón que quieres enviar a tu tía Millie por su cumpleaños, tienes que empacar el jarrón en una caja y hacer que un camión lo lleve de tu casa a la casa de ella. Pero si quieres darle el regalo de música, puedes ir a la tienda, comprarle un disco y enviarlo por correo, pero también puedes comprarle un certificado de regalo en Internet, enviarle un correo electrónico y hacer que descargue la música. En ese caso, la música llegará a su casa sin que un camión tenga que ir allí. La música será transportada únicamente por el patrón de electricidad que llega a su casa a través de la conexión a Internet.

Otra forma de ver la diferencia entre hardware y software es lo fácil que es hacer una copia del artículo. Si tiene una cortadora de césped y desea una segunda cortadora de césped, no hay ninguna máquina que copie la cortadora de césped. Podrías fotografiar la cortadora de césped, pero solo tendrías una fotografía plana de una cortadora de césped. No se podía cortar el césped con la foto. Para obtener una segunda cortadora de césped real, tendría que volver a la fábrica de cortadoras de césped y construir otra de hierro, plástico y cuerda y cualquier otra cosa de la que estén hechas las cortadoras de césped. Esto es

hardware.

El software se puede copiar fácilmente por máquina. Todo lo que necesita es algo que pueda leer el disco o lo que sea que esté grabado, y algo más para escribirlo en un disco nuevo. El nuevo será igual que el original, hará las mismas cosas. Si el original es tu película favorita, la copia también será tu película favorita. Si el original es un programa que preparará sus documentos de impuestos, también lo hará la copia.

El software no es algo físico, es simplemente cómo se establecen las cosas físicas.

Con mucho, la definición más utilizada de "software" es referirse a un paquete de código de instrucciones informáticas. Creo que la forma en que obtuvo este nombre es que una vez que ha construido un dispositivo tan versátil como una computadora, hay muchas cosas diferentes que se pueden hacer para hacer. Pero cuando no hay instrucciones en él, no puede hacer nada. Por lo tanto, el software es una parte absolutamente necesaria de una computadora que está realizando alguna tarea. Es una parte vital de la máquina total, pero no es como cualquier otra parte de la máquina. No puedes pesarlo ni medirlo ni recogerlo con un par de alicates. Así que es parte del 'ware', pero no es hardware. Lo único que queda para llamarlo es 'software'.

Programas

Como se mencionó anteriormente, una serie de instrucciones en la RAM se denominan programa.

Los programas vienen en muchos tamaños. Generalmente, un programa es una pieza de software que tiene todo lo necesario para realizar una tarea específica. Un sistema sería algo más grande, compuesto por varios programas. Un programa puede estar compuesto por varias partes más pequeñas conocidas como "rutinas". Las rutinas, a su vez, pueden estar formadas por subrutinas.

No existen definiciones duras y rápidas que diferencien entre sistema, programa, rutina y subrutina. Programa es el término general para todos ellos, la única diferencia es su tamaño y la forma en que se utilizan.

Existe otra distinción entre dos tipos de programas que no está relacionada con su tamaño. La mayoría de las computadoras domésticas y comerciales tienen una serie de programas instalados en ellas. La mayoría de estos programas se utilizan para hacer algo que el propietario quiere hacer. Estos se denominan programas de aplicación porque están escritos para aplicar la computadora a un problema que debe resolverse. Hay un programa en la mayoría de las computadoras que no es una aplicación. Su trabajo es lidiar con la computadora en sí y ayudar a los programas de aplicación. Este programa que no es una aplicación se llama Sistema Operativo.

El sistema operativo

Un "sistema operativo", o "SO" para abreviar, es un programa grande que tiene muchas partes y varios objetivos.

Su primer trabajo es poner la computadora en funcionamiento cuando la enciende por primera vez.

Otro de sus trabajos es iniciar y finalizar programas de aplicación y dar tiempo a cada uno para que se ejecute. Es el "jefe" de todos los demás programas en esa computadora. Cuando hay más de un programa en la RAM, es el sistema operativo el que cambia entre ellos. Permite que un programa se ejecute durante una pequeña fracción de segundo, luego otro programa, luego otro programa. Si hay diez programas en la RAM, y cada uno se ejecuta durante una centésima de segundo a la vez, cada programa podría ejecutar millones de instrucciones en ese tiempo, varias veces por segundo. Parecería que los diez programas se ejecutaban simultáneamente porque cada uno puede hacer algo, más rápido de lo que el ojo puede ver.

Un sistema operativo también proporciona servicios a los programas de aplicación. Cuando un programa de aplicación necesita leer, escribir en el disco o dibujar letras en la pantalla, no tiene que hacer todas las complicadas instrucciones de E/S necesarias para realizar la tarea. El sistema operativo tiene una serie de pequeñas rutinas que mantiene en la RAM en todo momento para tales fines.

Todo lo que una aplicación necesita hacer para usar una de estas rutinas es cargar información en los registros y luego saltar a la dirección de la rutina del sistema operativo adecuada. Aquí hay un ejemplo de cómo se podría hacer. Digamos que quieres dibujar un personaje en la pantalla. Primero, coloque el código ASCII del carácter deseado en R0.

Luego coloque los números de fila y columna donde desea que aparezca en la pantalla en R1 y R2. Y aquí está la parte complicada: Pones la dirección de la siguiente instrucción de tu programa de aplicación, en R3. Ahora simplemente salta a la rutina del sistema operativo. La rutina se encargará de todos los detalles del dibujo del personaje en la pantalla, y luego su última instrucción será JMPR R3. Por lo tanto, estas rutinas se pueden "llamar" desde cualquier aplicación y, cuando terminen,

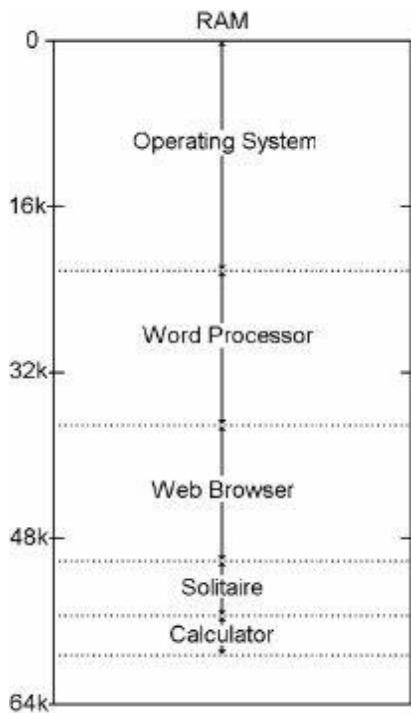
la rutina volverá a la siguiente instrucción

en la aplicación que lo llamó.

Hay varias razones para que el sistema operativo realice todas las funciones de E/S. Una es que facilita la escritura de programas de aplicación, el programador ni siquiera necesita saber cómo funcionan realmente los periféricos. Otra razón es que desperdiciaría mucha RAM si cada aplicación tuviera su propia copia de todas las rutinas de E / S. Una de las razones más importantes es que el sistema operativo puede verificar si se debe permitir que el programa haga lo que está pidiendo que haga. Esto es parte del otro trabajo del sistema operativo de ser el jefe.

El corazón del sistema operativo es básicamente un bucle de instrucciones que hace las siguientes preguntas: ¿Necesito ingresar algo? ¿Necesito generar algo? ¿Necesito dejar que se ejecute algún programa? Luego comienza de nuevo. Si la respuesta a todas estas preguntas es no, la CPU simplemente ejecuta las instrucciones en este bucle una y otra vez, millones de veces por segundo. Cuando hay algo que hacer, salta al principio del programa que se encarga de ello, y cuando se hace, vuelve a saltar a este bucle donde el sistema operativo "espera" a que haga otra cosa.

Aquí hay un diagrama de nuestra versión de RAM más grande, que muestra qué partes de la RAM podrían estar ocupadas por un sistema operativo y varios otros programas.



Dentro de la RAM de cada programa, está todo el código de instrucciones que hace que el programa funcione. Cada programa puede dividirse en su propio bucle principal y muchas rutinas que se utilizan para las diversas tareas que necesita realizar. Como se mencionó, el sistema operativo también tiene rutinas que pueden ser llamadas por otros programas.

Cada programa también utiliza parte de su "espacio de direcciones" para los datos en los que está trabajando. La calculadora, por ejemplo, debe tener unos pocos bytes donde almacena los números que el usuario ingresa en ella. El solitario necesita algunos bytes que especifiquen qué cartas están en qué posiciones. El procesador de textos necesita algo de RAM para todos los códigos ASCII que componen el documento en el que está trabajando. El sistema operativo también necesita bytes donde pueda almacenar fuentes, realizar un seguimiento de dónde se han cargado los programas de aplicación, recibir los datos que lee del disco y para muchos otros propósitos.

Y esto es lo que sucede dentro de su computadora promedio. Hay muchos programas y áreas de datos diferentes en la RAM. El sistema operativo salta a un programa, el programa salta a una rutina, la rutina salta a una subrutina. Cada programa trabaja con sus datos o calcula algo o realiza una operación de E/S. A medida que cada uno termina, salta de regreso a su lugar de origen. La CPU ejecuta una instrucción de un programa a la vez, y si se escriben de manera inteligente, cada programa hará su trabajo pieza por pieza, sin interferir con el resto.

Si nuestra computadora hubiera incluido un 'sistema de interrupción' como describimos hace unos capítulos, cada vez que alguien presionara una tecla en el teclado o moviera el mouse, habría una interrupción que llamaría a una parte del sistema operativo que determina qué dispositivo de E / S causó la interrupción, y luego llamaría a la rutina adecuada para encargarse de lo que sea. Cuando se hacía eso, la CPU continuaba con la siguiente instrucción de cualquier programa que se hubiera estado ejecutando cuando ocurría la interrupción.

Todo esto puede parecer muy complejo, con tantos millones y miles de millones de instrucciones que se ejecutan en un abrir y cerrar de ojos. Hay formas de

organizar programas y buenas prácticas de programación que pueden hacerlo mucho más comprensible. Un estudio de estos simplificaría el software de la misma manera que espero que este libro haya simplificado

el hardware. Pero ese sería el tema de otro libro completo.

Idiomas

Escribir programas es muy difícil de hacer cuando solo escribes unos y ceros, pero ese es el único código que la CPU 'entiende'.

¿Qué es un idioma? Un idioma hablado, como el inglés, es una forma de representar objetos, acciones e ideas con sonidos. Un lenguaje escrito es una forma de representar los sonidos de un idioma hablado con símbolos en papel. Suena como otro código, y un código que representa un código. ¡Simplemente no podemos alejarnos de estas cosas!

¿Recuerdas esa abreviatura que usamos cuando mirábamos el código de instrucciones de la CPU y el cableado en la Sección de Control? Bueno, eso es en realidad algo más que una herramienta útil que se inventó para este libro. Es un lenguaje informático. Aquí hay algunas líneas:

Código de instrucciones		Idioma		Significado
1000	rarb	AGREGAR	RA,RB	Agregar
1001	rarb	SHR	RA,RB	Desplazar a la derecha
1010	rarb	SHL	RA,RB	Desplazar a la izquierda
1011	rarb	NO	RA,RB	No

Un lenguaje informático es una forma de representar el código de instrucción. Su propósito es facilitar la escritura de programas informáticos.

Para usar este lenguaje, escribe el programa que desea con caracteres ASCII y lo guarda en un archivo. Luego carga un programa especial llamado 'compilador' en la RAM y salta a su primera instrucción. El compilador leerá el archivo ASCII, traducirá cada línea en el código de instrucción que representa y escribirá todos los bytes del código de instrucción en un segundo archivo. El segundo archivo puede cargarse en la RAM, y cuando la CPU salta a su primera instrucción, el programa que escribió en ASCII hará lo que pretendía que hiciera.

Por supuesto, cuando se inventaron las computadoras por

primera vez, todas las

los programas tenían que escribirse directamente en unos y ceros. Entonces alguien se cansó del tedio de programar de esa manera y decidió escribir el primer compilador. Luego, desde entonces, los programas se escribieron en este lenguaje más fácil y luego el compilador los tradujo al código de instrucción. Con el compilador original, incluso podría escribir un compilador mejor.

Entonces, para que exista un lenguaje informático, necesita dos cosas, un conjunto de palabras que componen el lenguaje (otro código) y un compilador que compila el lenguaje escrito en código de instrucción de computadora.

El lenguaje que hemos visto en este libro tiene solo unas 20 palabras. Cada palabra se correlaciona directamente con una de las instrucciones de las que es capaz esta computadora. Cada línea que escribes da como resultado una instrucción de computadora. Cuando escribe un programa de 87 líneas en este lenguaje, el archivo de código de instrucciones que genera el compilador tendrá 87 instrucciones.

Luego, alguien inventó un lenguaje de "nivel superior" donde una línea del lenguaje podría resultar en múltiples instrucciones de computadora. Por ejemplo, nuestra computadora no tiene una instrucción que haga resta. Pero el compilador podría diseñarse para que reconozca una nueva palabra en el lenguaje como 'SUB RA,RB' y luego genere tantas instrucciones de máquina como sean necesarias para que ocurra la resta. Si puedes descubrir cómo hacer algo elegante con 47 instrucciones, puedes tener una palabra en tu idioma que signifique esa cosa elegante.

Luego, alguien inventó un lenguaje de nivel aún más alto donde las palabras que componen el lenguaje ni siquiera se parecen a las instrucciones reales de la CPU. El compilador tiene mucho más trabajo por hacer, pero aún genera código de instrucciones que hace las cosas que significan las palabras en ese lenguaje. Algunas líneas de un lenguaje de nivel superior podrían verse así:

```
Saldo = 2,000 Tasa de
interés = .034
Imprimir "Hola Joe, tu interés este año es: $"
Imprimir Saldo X Tasa de Interés
```

El compilador de este lenguaje leería este programa de cuatro líneas y generaría un archivo que podría contener

fácilmente

cientos de bytes de código de instrucción. Cuando ese código de instrucción se cargaba en la RAM y se ejecutaba, imprimía:

Hola Joe, tu interés este año es: \$68

Escribir software en lenguajes de nivel superior puede resultar en hacer mucho más en un período de tiempo más corto, y el programador ya no necesita saber exactamente cómo funciona realmente la computadora.

Hay muchos lenguajes informáticos. Algunos lenguajes están diseñados para hacer trabajo científico, algunos están diseñados para fines comerciales, otros son de propósito más general.

Los idiomas de nivel inferior siguen siendo los mejores para ciertos propósitos.

El sistema de archivos

Como vimos anteriormente, la forma en que funciona realmente un disco es bastante extraña para la mayoría de las personas que usan una computadora.

Para facilitar las cosas, alguien inventó una idea llamada "archivo". Se supone que un archivo es similar al tipo de archivos en papel que la gente usa en las oficinas. Un archivo de papel es una hoja de cartón doblada por la mitad y colocada en un archivador. Esta carpeta tiene una pestaña donde puede escribir algún tipo de nombre para la carpeta, y luego puede poner una o varias hojas de papel en la carpeta.

Un archivo de computadora es una cadena de bytes que puede tener cualquier longitud, desde un byte hasta todos los bytes disponibles en el disco. Un archivo también tiene un nombre. Un disco puede tener muchos archivos, cada uno con su propio nombre.

Por supuesto, estos archivos son solo una idea. Para que un sistema de archivos funcione, el sistema operativo proporciona un montón de software que hace que el disco parezca un archivador en lugar de tener cabezas, pistas, sectores y bloques de bytes.

Este sistema de archivos proporciona a los programas de aplicación una forma fácil de utilizar el disco. Las aplicaciones pueden pedirle al sistema operativo que cree, lea, escriba o borre algo llamado archivo. Todo lo que la aplicación necesita saber es el nombre del archivo. Lo abres, solicitas bytes, le envías bytes, lo haces más grande o más pequeño, cierras el archivo.

El sistema operativo utiliza parte del disco para mantener una lista de nombres de archivo, junto con la longitud de cada archivo y la dirección del disco (cabeza, pista, sector) del primer sector de los datos. Si el archivo es más pequeño que un sector de disco, eso es todo lo que necesita, pero si el archivo es más grande que un sector, también hay una lista que contiene tantas direcciones de tipo disco como sea necesario para contener el archivo.

El programa de aplicación dice crear un archivo con el nombre "carta a Jane". Luego, el usuario escribe la carta a Jane y la guarda. El programa le dice al sistema

operativo dónde está la letra en la RAM y qué tan larga es, y el sistema operativo la escribe en el disco en el sector o sectores adecuados y actualiza la longitud del archivo y las listas necesarias de direcciones de tipo de disco.

Para usar el sistema de archivos, habrá algún tipo de reglas que el programa de aplicación debe seguir. Si desea escribir algunos bytes en el disco, deberá decirle al sistema operativo el nombre del archivo, la dirección RAM de los bytes que desea escribir y cuántos bytes escribir. Normalmente, colocaría toda esta información en una serie de bytes en algún lugar de la RAM y, a continuación, colocaría la dirección RAM del primer byte de esta información en uno de los registros y, a continuación, ejecutaría una instrucción Jump que salta a una rutina dentro del sistema operativo que escribe archivos en el disco. Todos los detalles son atendidos por esta rutina, que forma parte del sistema operativo.

Si le pide al sistema operativo que mire su disco, le mostrará una lista de todos los nombres de archivo y, por lo general, sus tamaños y la fecha y hora en que se escribieron por última vez.

Puedes almacenar todo tipo de cosas en archivos. Los archivos suelen tener nombres que se componen de dos partes separadas por un punto como "xxxx.aaa". La parte antes del punto es una especie de nombre como "carta a Jane", y la parte después del punto es una especie de tipo como "doc", que es la abreviatura de "documento". La parte anterior al punto le dice algo sobre lo que hay en el archivo.

La parte después del punto le indica qué tipo de datos contiene este archivo, en otras palabras, qué código utiliza.

El tipo de archivo le indica a usted y al sistema operativo qué código usan los datos del archivo. En un sistema operativo popular, ".txt" significa texto, lo que significa que el archivo contiene ASCII. Un ".bmp" significa BitMaP, que es una imagen. Un ".exe" significa ejecutable, lo que significa que es un programa y, por lo tanto, contiene un código de instrucciones.

Si le pregunta al sistema operativo qué programas están disponibles para ejecutar, le mostrará una lista de los archivos que terminan en ".exe". Si solicita una lista de imágenes que puede ver, le mostrará una lista de archivos que terminan con ".bmp".

Hay muchos tipos de archivos posibles, cualquier programa puede inventar su propio tipo y usar cualquier código o combinación de códigos.

Errores

La computadora es una máquina bastante compleja que hace una serie de cosas simples una tras otra muy rápidamente. ¿Qué tipo de cosas podrían salir mal aquí?

En los primeros días de la informática, cuando cada puerta de la computadora era relativamente costosa de construir, a veces había componentes que en realidad tenían partes móviles para hacer conexiones eléctricas. Dos piezas de metal tuvieron que tocarse para que la electricidad fuera a donde los constructores querían que fuera. A veces, cuando la máquina dejaba de funcionar correctamente, el tipo de reparación miraba dentro para averiguar qué estaba mal, y descubría que una araña se había arrastrado dentro de la máquina y se había quedado atrapada entre dos de estas piezas de metal que se suponía que se tocaban entre sí. Luego, cuando una pieza de metal se movía para tocar la otra, la araña estaba en el camino y no se tocaban. Por lo tanto, la electricidad no llegaría a donde tenía que ir y la máquina ya no funcionaría correctamente. El tipo de corrección eliminaría el error, limpiaría los contactos y informaría "Hubo un error en la computadora". Y literalmente se refería a un error.

Con el tiempo, cada vez que una computadora parecía estar funcionando incorrectamente, la gente decía que la computadora tenía un error. Hay dos clases principales de errores informáticos: hardware y software.

Un error de hardware en realidad significa que la computadora está rota. Esto podría ser tan grave como encender la computadora y se incendia, ya que hay un byte en la RAM donde un bit siempre está apagado.

Ahora, un bit en la RAM que se niega a cambiar puede ser un problema o no. Si el byte donde se encuentra ese bit de alguna manera nunca se usa, entonces la computadora funcionará bien. Si ese byte es parte de un lugar donde se almacena un nombre, entonces el nombre puede cambiarse de "Joe" a "Jod". Si ese byte tiene algunas instrucciones de programa, es posible que una instrucción XOR se cambie a una instrucción JMP. Luego, cuando el programa llegue a esa instrucción, no hará el XOR como se supone que debe hacerlo, sino que saltará a otro lugar y comenzará a ejecutar lo que sea que esté en

la nueva ubicación como si

fue una serie de instrucciones. El contenido de esos bytes determinará lo que sucederá a continuación, pero es casi seguro que será tan incorrecto como un tren que se sale de su vía.

Si se rompe una puerta en el paso a paso, por ejemplo, de modo que el paso 4 nunca se enciende, entonces la computadora no podrá funcionar en absoluto. Todavía podría obtener instrucciones en los pasos 1, 2 y 3, pero cada instrucción se ejecutaría incorrectamente. Ciertamente, el programa haría un desastre después de 'ejecutar' solo unas pocas instrucciones.

Los errores de software pueden tomar muchas formas, pero en última instancia, todos son errores del programador. Probablemente hay muchas más formas de escribir un programa incorrectamente que correctamente. Algunos errores solo crean algún tipo de resultado incorrecto y otros errores hacen que la computadora se "bloquee".

Una de mis historias favoritas de programadores estúpidos es esta: alguien compró un automóvil a crédito. Recibió un talonario de cupones con el préstamo, un cupón para enviar con cada pago. Pero cuando hizo su primer pago, accidentalmente usó el último cupón del libro en lugar del primero. Unas semanas más tarde, recibió un carta generada por computadora de la compañía de préstamos que dice: "Gracias por pagar su préstamo en su totalidad, la próxima vez que necesite un préstamo, utilícenos nuevamente". Obviamente, el programa simplemente verificaba el número de cupón y si era igual al cupón de número más alto en el libro, salte a la rutina para un préstamo pagado en su totalidad. Al menos debería haber verificado el saldo restante del préstamo antes de decidir que se pagó. Este es un error sutil, es posible que la compañía de préstamos no lo detecte hasta que audite sus libros meses después. La computadora hizo exactamente lo que se le dijo que hiciera, y la mayoría de las veces fue adecuado, pero el programa no fue escrito para anticipar todas las situaciones que a veces ocurren en el mundo real.

Uno de los peores errores de software es quedarse atascado en un bucle. El programa ejecuta una serie de instrucciones y luego vuelve al principio de la serie y la ejecuta una y otra vez. Por supuesto, los bucles se

usan todo el tiempo en la programación, pero se usan para hacer algo que tiene un número finito de pasos similares.

Puede repetirse hasta que se hayan movido 50 bytes a algún lugar, o seguir comprobando si el usuario presiona una tecla en el teclado. Pero la computadora saldrá del bucle en algún momento y continuará con su siguiente tarea. Pero si hay algún tipo de error de programación en el que hay un bucle que no tiene salida, la computadora parecerá estar completamente atascada. Esto a veces se llama estar "colgado", es posible que sea necesario apagar y reiniciar toda la computadora para salir del bucle y volver a funcionar de manera útil.

Hay todo tipo de errores que terminan con la CPU tratando de ejecutar algo más que código de instrucciones. Supongamos que tiene su programa en la dirección 10 a 150 y tiene algunos datos ASCII, como nombres y números de teléfono en las direcciones 151 a 210. Si el programa está escrito incorrectamente para que, bajo ciertas condiciones, salte a la dirección 180, simplemente continuará obteniendo y ejecutando los bytes a partir de la dirección 180. Si 180-189 se llenó con el ASCII para "Jane Smith", el "programa" ahora ejecutará basura completa, una serie de bytes que no fueron diseñados para ser código de instrucción. Puede ponerse en un bucle, o saltar hacia atrás en algún lugar del programa, o emitir el comando para borrar la unidad de disco. Y hará basura a su alta velocidad habitual. Si miraba los patrones en los bytes, podía ver lo que haría, pero podría ser casi cualquier cosa. Si el nombre en la dirección 180 fuera "Bill Jones", haría algo completamente diferente. Dado que no está diseñado para ser útil, lo más probable es que siga haciendo un desastre mayor con lo que está en la memoria hasta que la computadora tenga que apagarse para que se detenga.

Otro tipo de error podría ocurrir si un programa escribió accidentalmente "John Smith" en el lugar donde se almacenó una fuente. En ese caso, cada letra "E" que se dibujó en la pantalla a partir de entonces se vería así: 'E'

La computadora ejecuta cientos de millones de instrucciones cada segundo, y solo se necesita una instrucción incorrecta para detener todo en seco. Por lo tanto, el tema de programar computadoras de una manera que esté completamente "libre de errores" es algo

eso recibe mucha atención. Casi toda la programación se realiza con lenguajes, y los compiladores de estos lenguajes están diseñados para generar Código de Instrucción que evite los tipos de errores más graves, y para advertir al programador si se violan ciertas buenas prácticas de programación. Aún así, los compiladores pueden tener errores y nunca podrán detectar un error como el anterior con el préstamo del automóvil.

Como puede ver, la computadora y su software son cosas bastante frágiles. Cada puerta tiene que funcionar siempre, y cada instrucción que se ejecuta tiene que ser correcta. Cuando consideras todas las cosas que podrían salir mal, el alto porcentaje de cosas que normalmente salen bien es bastante impresionante.

¿Enfermedades informáticas?

Otro lugar donde se asignan características humanas a las computadoras es algo llamado virus informático. Esto implica que las computadoras pueden contraer una enfermedad y enfermarse. ¿Van a empezar a toser y estornudar? ¿Se resfriarán o tendrán varicela? ¿Qué es exactamente un virus informático?

Un virus informático es un programa escrito por alguien que quiere hacerle algo malo a usted y a su computadora. Es un programa que hará algún tipo de travesura a su computadora cuando se ejecute. La motivación de las personas que escriben programas de virus va desde el simple desafío técnico de ver si uno es capaz de hacerlo, hasta el deseo de derribar la economía de todo el mundo. En cualquier caso, las personas que hacen tales cosas no tienen en mente sus mejores intereses.

¿Cómo 'atrapa' una computadora un virus? Se debe colocar un programa de virus en su RAM y su computadora debe saltar al programa de virus y ejecutarlo. Cuando se ejecuta, localiza un archivo que ya está en su disco duro, que contiene un programa que su computadora ejecuta regularmente, como una parte del sistema operativo. Después de que el programa de virus localiza este archivo, copia el programa de virus al final de este archivo e inserta una instrucción de salto al principio del archivo que provoca un salto a donde está el programa de virus. Ahora tu computadora tiene un virus.

Cuando una computadora con un virus se está ejecutando, hace todas las cosas que se supone que debe hacer, pero cada vez que ejecuta el programa que contiene el virus, la instrucción de salto insertada hace que el programa de virus se ejecute en su lugar. Ahora, el virus generalmente hará algo simple, como verificar una fecha predeterminada, y si no coincide, el programa de virus volverá al comienzo del archivo donde aún existe el programa del sistema operativo.

Por lo tanto, su computadora parecerá totalmente normal, solo se están ejecutando algunas instrucciones adicionales durante sus operaciones regulares. El virus se considera inactivo en este momento. Pero cuando llega esa fecha, y el virus

'decide' hacer lo que esté en el resto de su programa, puede ser cualquier cosa. Cuando el programa de virus se está ejecutando, puede hacer cualquier daño que se le ocurra a la persona que lo escribió. Puede borrar archivos en su disco o enviarlos a otro lugar a través de Internet. Un virus humorístico, de vez en cuando, hacía que las letras de la pantalla parecieran soltarse y caer en una pila en la parte inferior de la pantalla.

Aquí hay un ejemplo de cómo contraer un virus. Digamos que tienes un amigo que encuentra una película divertida en Internet. Lo hace reír y cree que tú también lo disfrutarás, así que te envía el archivo de la película por correo electrónico. Recibes el archivo de película y lo reproduces, y lo disfrutas.

Hay dos cosas diferentes que podrían haber ocurrido aquí. Si su amigo le envió un archivo llamado "funny.mov" y su sistema operativo incluye un programa que reproduce archivos '.mov', entonces el sistema operativo cargará ese programa en la RAM y ese programa leerá las imágenes en el archivo "funny.mov" y las mostrará en su pantalla. Esto está bien, el programa que se ejecutó era algo que ya estaba en su computadora. El archivo "funny.mov" solo proporcionó una serie de imágenes que se mostraron en su pantalla.

Pero si su amigo le envió un archivo llamado "funny.exe", cuando le pida al sistema operativo que reproduzca la película, cargará "funny.exe" en la RAM y saltará a su primera instrucción. Ahora tiene un programa ejecutándose en su computadora que vino de otro lugar. Si se trata de un programa de virus, probablemente reproducirá la película por ti para que no sospeches nada, pero puede hacer cualquier otra cosa que quiera, en los archivos de tu disco mientras estás viendo la película. Probablemente se instalará y entrará en un estado inactivo durante días o semanas, y ni siquiera sabrá que su computadora está 'infectada'. Pero tarde o temprano cobrará vida y hará el daño para el que fue diseñado.

Este tipo de programa malicioso se llama virus porque la forma en que funciona es similar a la forma en que los virus reales infectan a los seres vivos. Un virus real es una cosa que es más pequeña que un animal unicelular. No califica del todo como vivo porque el virus por sí solo no puede reproducirse. Sin embargo, se reproducen

invadiendo una célula de algo que está vivo. Una vez en la célula, el virus utiliza los mecanismos de esa célula para hacer copias de sí mismo, que luego pueden continuar e infectar otras células.

El virus informático tampoco puede reproducirse ni hacer nada más por sí mismo. Necesita entrar en una computadora y, de alguna manera, ser ejecutado una vez por esa CPU. Cuando se ejecuta por primera vez, se inserta en algún lugar del sistema operativo para que a partir de entonces se ejecute de forma regular. Esas instrucciones harán cualquier daño que estén diseñadas para hacer a la computadora en la que se están ejecutando, y también generalmente harán algo que esté diseñado para propagar el virus a otras computadoras.

Firmware

Por supuesto, la RAM es una parte esencial de cualquier computadora. La capacidad de escribir bytes en la RAM y volver a leerlos es una parte integral de cómo funciona la máquina.

Pero en algunas computadoras, hay secciones de la RAM en las que solo se escribe cuando se inicia la computadora y, a partir de entonces, estas secciones permanecen sin cambios mientras la computadora funciona. Esto podría ser cierto en cualquier computadora que siempre ejecute el mismo programa. Quizás la mitad de la RAM se usa para contener el programa y la otra mitad de la RAM se usa para contener los datos en los que está trabajando el programa. La mitad con el programa tiene que cargarse en algún momento, pero después de eso, la CPU solo tiene que leer los bytes del programa para obtenerlos y ejecutarlos.

Cuando tiene este tipo de situación, puede construir la mitad de la RAM de su computadora de la manera normal, y con la otra mitad, omite las puertas NAND y simplemente conecta cada bit directamente a un encendido o apagado en el patrón de su programa.

Por supuesto, no puede escribir en la RAM precableada, pero puede leer desde ella sin problemas. A este tipo de RAM se le dio el nombre de memoria de solo lectura, o ROM para abreviar. Lo usas de la misma manera que usas la RAM, pero solo lees de él.

Hay dos ventajas de ROM. En los primeros días de las computadoras, cuando la RAM era muy cara, la ROM era mucho menos costosa que la RAM.

La otra ventaja es que ya no tiene que cargar el programa en la RAM cuando enciende la computadora por primera vez. Ya está allí en ROM, listo para ser ejecutado por la CPU.

El punto aquí es una nueva palabra. Dado que el software se llamó 'suave' porque es cambiabile, cuando se trata de ROM, todavía tiene un patrón en los bits, pero ya no son tan suaves. No puedes escribir en una ROM, no puedes cambiar los bits. Y así, este tipo de memoria llegó a conocerse como 'firmware'. Es software que está permanentemente escrito en el hardware.

Pero ese no es el final de la historia. La ROM descrita anteriormente tuvo que construirse de esa manera en la fábrica. Con el paso de los años, esta idea se mejoró y se hizo más fácil de usar.

El siguiente avance fue cuando alguien tuvo la brillante idea de hacer una ROM en la que cada bit se encendiera en la fábrica, pero había una forma de escribir con mucha potencia que podía quemar las conexiones individuales, cambiando los bits individuales a un apagado. Por lo tanto, esta ROM podría programarse después de salir de fábrica. Esto se llamó 'ROM programable' o 'PROM' para abreviar.

Luego, alguien descubrió cómo hacer un PROM que repararía todas esas conexiones rotas si se expusiera a la luz ultravioleta durante media hora. Esto se llamó 'Erasable PROM', o 'EPROM' para abreviar.

Luego, alguien descubrió cómo construir una EPROM que pudiera borrarse usando energía adicional en un cable especial integrado en la EPROM. Esto se llamó 'PROM borrable eléctricamente', o 'EEPROM' para abreviar. Un tipo particular de EEPROM tiene el nombre de 'memoria flash'.

Así que hay RAM, ROM, PROM, EPROM, EEPROM y Flash. Estos son todos los tipos de memoria de computadora. Lo que tienen en común es que todos permiten el acceso aleatorio.

Todos funcionan de la misma manera cuando se trata de abordar bytes y leer los datos que contienen. La gran diferencia es que la RAM pierde su configuración cuando se corta la energía. Cuando vuelve la alimentación, la RAM está llena de ceros. El resto de ellos todavía tienen sus datos después de apagarlos y volver a encenderlos.

Puede preguntarse entonces: "¿Por qué las computadoras no usan EEPROM para su RAM? Luego, el programa permanecería en la RAM cuando la computadora estuviera apagada". La respuesta es que se tarda mucho más en escribir en EEPROM que en RAM. Ralentizaría enormemente la computadora. Si alguien descubre cómo hacer una EEPROM que sea tan rápida y barata y use la misma o menos energía que la RAM, estoy seguro de que se hará.

Por cierto, la palabra ROM también se ha utilizado para referirse a cualquier tipo de almacenamiento que se establece permanentemente, como un disco pregrabado, como

en 'CD ROM', pero su definición original solo se aplicaba a algo que funcionaba igual que la RAM.

Botas

¿Qué tienen que ver las botas con las computadoras? Bueno, hay una vieja frase que dice "levántate por tus propios medios". Es una especie de broma, literalmente se refiere a las correas que se cosen en muchas botas que se usan para ayudar a ponerse las botas en los pies. La broma es que si llevas un par de botas de este tipo y quieres levantarte del suelo, en lugar de coger una escalera o subir una cuerda, puedes levantarte del suelo simplemente tirando lo suficientemente fuerte de esas botas. Por supuesto, esto solo funcionaría en una caricatura, pero la frase ha llegado a significar hacer algo cuando no hay una forma aparente de hacerlo, o hacer algo sin las herramientas que normalmente se usarían, o lograr algo por ti mismo sin la ayuda de nadie más.

En una computadora, hay un problema que es similar a la necesidad de despegar y no tener herramientas disponibles para lograrlo. Cuando una computadora está funcionando, la memoria está llena de programas que están haciendo algo, y cuando el operador de la computadora ingresa un comando para iniciar otro programa, el sistema operativo localiza el programa en el disco, lo carga en la memoria y salta a la primera instrucción del programa. Ahora ese programa se está ejecutando.

Pero cuando enciendes una computadora por primera vez, ¿cómo obtienes el sistema operativo en la memoria? Se necesita un programa que se ejecuta en la memoria para decirle a la unidad de disco que envíe algún código de instrucción, y el programa necesita escribir ese código en la memoria en un lugar apropiado, y luego saltar a su primera instrucción para ejecutar el nuevo programa. Pero cuando enciendes la computadora, cada byte en la memoria es todo ceros. No hay instrucciones en la memoria en absoluto. Esta es la situación imposible, necesitas un programa en la memoria para obtener un programa en la memoria, pero no hay nada allí. Entonces, para que la computadora se ponga en marcha en primer lugar, la computadora tiene que hacer algo imposible. ¡Tiene que levantarse por sí mismo!

Hace mucho tiempo, en los primeros días de las computadoras, la máquina tenía interruptores y botones en el panel frontal que permitían al operador ingresar

bytes de datos

directamente en los registros, y desde allí, en RAM. De esta manera, puede ingresar manualmente un programa corto y comenzar a ejecutarlo. Este programa, llamado "cargador de arranque", sería el programa más pequeño posible que podría escribir que le indicaría a la computadora que lea bytes de un periférico, los almacene en RAM y luego salte a la primera instrucción. Cuando se ejecuta el cargador de arranque, carga un programa mucho más grande en la memoria, como los comienzos de un sistema operativo, y luego la computadora se volverá utilizable.

Hoy en día, existen formas mucho más fáciles de cargar el primer programa en la computadora, de hecho, sucede automáticamente inmediatamente después de que se enciende la computadora. Pero este proceso aún ocurre, y el primer paso se llama "arranque" o "arranque" y solo significa obtener el primer programa en la memoria y comenzar a ejecutarlo.

La solución más común a este problema tiene tres partes. Primero, el IAR está diseñado para que cuando se encienda la alimentación por primera vez, en lugar de que todos sus bits sean cero, su último bit sea cero, pero el resto de sus bits serán unos. Por lo tanto, para nuestra pequeña computadora, la primera instrucción que se obtendrá estará en la dirección 1111 1110. En segundo lugar, algo así como los últimos 32 bytes de la RAM (235-256) serán ROM en su lugar, cableados con un programa simple que accede a la unidad de disco, selecciona head 0, track 0, sector 0, lee este sector en RAM y luego salta al primer byte de la misma. La tercera parte, entonces, es mejor que haya un programa escrito en ese primer sector del disco. Este sector, por cierto, se llama 'disco de arranque'.

Esta palabra 'boot' se ha convertido en un verbo en el lenguaje de la computadora. Significa cargar un programa en la RAM donde no hay programas. A veces la gente lo usa para significar cargar cualquier programa en la RAM, pero su significado original solo se aplicaba a cargar el primer programa en una RAM en blanco.

Digital vs. Analógico

Sin duda has escuchado estos términos. Parece que todo lo asociado con las computadoras es digital, y todo lo demás no lo es. Pero eso no está lo suficientemente cerca de la verdad.

Lo que quieren decir es bastante simple, pero de dónde vinieron y cómo terminaron en su uso actual no es tan sencillo.

La palabra 'digital' proviene de digit, que significa dedos de manos y pies en algunos idiomas antiguos, y dado que los dedos de las manos y los pies se han utilizado para contar, digital significa que tiene que ver con números. Hoy en día, los símbolos individuales que usamos para escribir números (0, 1, 2, 3, etc.) se llaman dígitos. En la computadora, representamos números con bits y bytes. Una de las cualidades de los bits y bytes es su naturaleza inequívoca. Un poco está encendido o apagado; no hay un área gris en el medio. Un byte siempre está en uno de sus 256 estados; No hay estado entre dos números como 123 y 124. El hecho de que estos estados cambien en pasos es a lo que nos referimos cuando decimos digital.

La palabra 'analógico' proviene del mismo lugar que 'analogía' y 'análogo', por lo que tiene que ver con la similitud entre dos cosas. En el mundo real, la mayoría de las cosas cambian gradual y continuamente, no en pasos. Una voz puede ser un grito o un susurro o absolutamente en cualquier punto intermedio. Cuando un teléfono convierte una voz en un equivalente eléctrico para que pueda viajar a través de un cable a otro teléfono, esa electricidad también puede variar en todas partes entre estar completamente encendida y completamente apagada.

El sonido y la electricidad son dos cosas muy diferentes, pero la esencia de la voz se ha duplicado con la electricidad. Dado que son similares en ese sentido, podemos decir que el patrón eléctrico es un 'análogo' de la voz. Aunque el significado de 'analógico' proviene de este factor de 'similitud', cuando haces un análogo, generalmente estás haciendo un análogo de algo que es continuamente variable. Esta idea de que algo es continuamente variable se ha convertido en la definición

de analógico cuando se compara lo digital y lo analógico.
Algo que es analógico puede estar en cualquier lugar dentro
del

todo de algún rango, no hay pasos.

Los medios digitales cambian por pasos y los medios analógicos cambian de manera continua y suave. Otra forma de decirlo es que digital significa que los elementos que componen un todo provienen de un número finito de opciones, mientras que analógico significa que una cosa está hecha de partes que se pueden seleccionar entre un número ilimitado de opciones. Algunos ejemplos no informáticos pueden ayudar a aclarar esto.

Si tiene una plataforma que está a tres pies sobre el piso, puede construir escaleras para que la gente suba a ella o una rampa. En la rampa, puedes subir a cualquier nivel entre el piso y la plataforma; En las escaleras, solo tienes tantas opciones como escalones. La rampa es analógica, las escaleras son digitales.

Digamos que quieres construir una pasarela en tu jardín. Tiene la opción de hacer la pasarela de concreto o de ladrillos. Si los ladrillos tienen tres pulgadas de ancho, entonces puede hacer una caminata de ladrillos de 30 pulgadas de ancho o 33 pulgadas de ancho, pero no 31 o 32. Si haces la caminata de concreto, puedes verterlo al ancho que desees. Los ladrillos son digitales, el hormigón es analógico.

Si tienes un libro antiguo y una pintura al óleo antigua, y quieres hacer una copia de cada uno, te resultará mucho más fácil hacer una copia del libro. Incluso si las páginas del libro están amarillentas, y las esquinas están con orejas de perro, y hay manchas de suciedad y agujeros de gusano en el interior, siempre que pueda leer todas las letras del libro, puede volver a escribir todo el texto, exactamente como lo pretendía el autor. Con la pintura al óleo, los colores originales pueden haberse desvanecido y estar oscurecidos por la suciedad. La ubicación exacta de cada cerda en cada pincelada, el grosor de la pintura en cada punto, la forma en que se mezclan los colores adyacentes, todo podría copiarse con gran detalle, pero inevitablemente habría algunas ligeras diferencias. Cada letra del libro proviene de una lista de un número específico de posibilidades; Las variaciones de colores de pintura y sus posiciones en el lienzo son ilimitadas. El libro es digital, la pintura es analógica.

Así que ahí tienes la diferencia entre analógico y digital. El mundo que nos rodea es mayoritariamente

analógico. La mayoría de las tecnologías antiguas eran analógicas, como el teléfono,

fonógrafo, radio, televisión, grabadoras y videocasetes. Sin embargo, por extraño que parezca, uno de los dispositivos más antiguos, el telégrafo, era digital. Ahora que la tecnología digital se ha vuelto altamente desarrollada y económica, los dispositivos analógicos están siendo reemplazados uno por uno con versiones digitales que logran las mismas cosas.

El sonido es algo analógico. Un teléfono antiguo es una máquina analógica que convierte el sonido analógico en un patrón eléctrico que es un análogo del sonido, que luego viaja a través de un cable a otro teléfono. Un nuevo teléfono digital toma el sonido analógico y lo convierte en un código digital. Luego, el código digital viaja a otro teléfono digital donde el código digital se convierte nuevamente en sonido analógico.

¿Por qué alguien se tomaría la molestia de inventar un teléfono digital cuando el teléfono analógico funcionaba bien? La respuesta, por supuesto, es que aunque el teléfono analógico funcionaba, no era perfecto. Cuando un patrón eléctrico analógico viaja largas distancias, le pueden pasar muchas cosas en el camino. Se hace cada vez más pequeño a medida que viaja, por lo que tiene que amplificarse, lo que introduce ruido, y cuando se acerca a otros equipos eléctricos, parte del patrón de los otros equipos puede mezclarse en la conversación. Cuanto más lejos llega el sonido, más ruido y distorsión se introducen. Cada cambio en el análogo de tu voz se convierte en parte del sonido que sale en el otro extremo.

Ingresa la tecnología digital al rescate. Cuando envía un código digital a largas distancias, los bits individuales están sujetos a los mismos tipos de distorsión y ruido, y cambian ligeramente. Sin embargo, no importa si un bit es solo 97% en lugar de 100%. La entrada de una puerta solo necesita "saber" si el bit está encendido o apagado, tiene que "decidir" entre esas dos opciones solamente. Mientras un bit esté a más de la mitad, la puerta en la que entra actuará exactamente de la misma manera que si el bit hubiera estado completamente encendido. Por lo tanto, el patrón digital al final es tan bueno como al principio, y cuando se vuelve a convertir a analógico, no hay ruido ni distorsión en absoluto, suena como si la persona estuviera justo al lado.

Hay ventajas y desventajas en cada método, pero en general, los beneficios de la tecnología digital superan con creces sus deficiencias.

Probablemente la mayor ventaja de lo digital tiene que ver con la realización de copias. Cuando haces una copia de algo como un disco de vinilo, puedes grabarlo en una grabadora, o supongo que incluso puedes conseguir todo el equipo para cortar un nuevo disco de vinilo. Pero habrá cierto grado de diferencia entre el original y la copia. En primer lugar, toda la maquinaria tiene limitaciones de precisión. Una copia de cualquier objeto físico puede ser muy cercana al original, pero nunca del todo exacta. En segundo lugar, si hay rasguños o partículas de polvo en el original, la copia tendrá duplicados de estos defectos. En tercer lugar, la fricción entre el disco y la aguja en realidad desgasta una pequeña cantidad de vinilo cada vez que lo reproduces. Si usa una grabadora, siempre se agrega un bajo nivel de "silbido" al sonido. Si haces una copia de una copia, y una copia de eso, etc., los cambios se harán cada vez más grandes en cada etapa.

Cuando se trata de algo que es digital, siempre que cada parte que estaba en el original también esté en la copia, obtenemos una copia exacta cada vez. Puedes hacer una copia de la copia, y una copia de aquella, etc., y cada una de ellas será exactamente igual que el original.

Lo digital es definitivamente el camino a seguir si desea poder hacer un número ilimitado de copias y preservar algo para siempre.

La computadora y los periféricos que hemos construido son completamente digitales hasta ahora. Y si todo lo que quisiéramos hacer con ellos fueran cosas digitales como la aritmética y el lenguaje escrito, podríamos dejarlo así. Sin embargo, si queremos que nuestro ordenador reproduzca música y trabaje con fotografías en color, hay una cosa más que debemos tener en cuenta.

Mentí, más o menos

Hay una pieza de hardware en una computadora que no está hecha completamente de puertas NAND. Esto no es realmente necesario para hacer que una computadora sea una computadora, pero la mayoría de las computadoras tienen algunas de ellas. Se utilizan para cambiar de algo que es analógico a algo que es digital, o de digital a analógico.

Los ojos y oídos humanos responden a cosas analógicas. Las cosas que escuchamos pueden ser fuertes o suaves, las cosas que vemos pueden ser brillantes u oscuras y ser de una multitud de colores.

La pantalla de visualización de la computadora que describimos anteriormente tenía 320 x 200 o 64.000 píxeles. Pero cada píxel solo tenía un bit para decirle qué hacer, estar encendido o apagado. Esto está bien para mostrar lenguaje escrito en la pantalla, o podría usarse para hacer dibujos lineales, cualquier cosa que solo tenga dos niveles de brillo. Pero todos hemos visto fotografías en pantallas de computadora.

En primer lugar, debe haber una manera de poner diferentes colores en la pantalla. Si sacas una lupa y miras una computadora o pantalla de televisión a color, verás que la pantalla en realidad está formada por pequeños puntos de tres colores diferentes, azul, rojo y verde.

Cada píxel tiene tres partes, una para cada color. Cuando el adaptador de pantalla escanea la pantalla, selecciona los tres colores de cada píxel al mismo tiempo.

Para que una computadora tenga una pantalla a color, necesita tener tres bits para cada píxel, por lo que tendría que tener tres veces la RAM para poder controlar los tres colores en cada píxel individualmente. Con tres bits, cada color podría estar completamente encendido o apagado y, por lo tanto, cada píxel tendría ocho estados posibles: negro, verde, rojo, azul, verde y rojo (amarillo), verde y azul (cian), azul y rojo (magenta) y verde, azul y rojo (blanco).

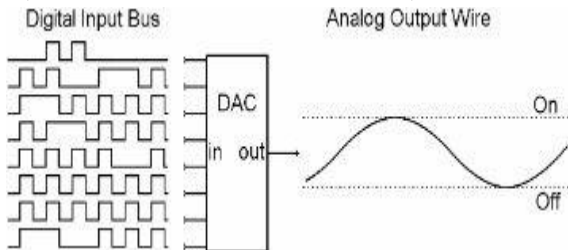
Pero esto todavía no es suficiente para mostrar una

fotografía. Para hacer eso, necesitamos poder controlar el brillo de cada color en todo el rango entre completamente encendido y completamente apagado. Para hacer esto, necesitamos un nuevo tipo de pieza que describiremos en breve, y necesitamos más bits en la RAM de la pantalla. En lugar de un bit para cada color en cada

píxel, podríamos tener un byte completo para cada color en cada píxel. Eso es tres bytes por píxel, para un total de 192,000 bytes de RAM solo para esta pequeña pantalla.

Con estos bytes, utilizando el código numérico binario, podría especificar 256 niveles de brillo para cada color en cada píxel. Esto equivaldría a 16.777.216 estados (o colores) diferentes para cada píxel. Esta es suficiente variedad para mostrar una fotografía razonablemente atractiva.

Para que esto funcione, un número que especifica 256 niveles diferentes de brillo, necesita algo llamado "convertidor digital a analógico" o "DAC" para abreviar. Un DAC tiene ocho entradas digitales y una salida analógica. La forma en que funciona es que está conectado para tratar la entrada como un número binario, y la salida tiene 256 niveles entre apagado y encendido. La salida tiene 256 gradaciones entre apagado y encendido, y llega al nivel que especifica el número de entrada. Si la entrada es un 128, la salida estará a la mitad. Para un 64, la salida será de un trimestre. Para 0, la salida



estará completamente apagada.

Para que esta pantalla a color funcione, el adaptador de pantalla necesita acceder a tres bytes a la vez, conectarlos a tres DAC y conectar las salidas de los DAC a los tres colores en el píxel actual que se está pintando. Así es como funciona una pantalla a color.

Cuando definimos 'analógico' en el último capítulo, dijimos

que era algo que era continuamente variable de completamente apagado a completamente encendido. Pero nuestro DAC realmente solo tiene 256 niveles diferentes en su salida 'analógica'. Está mucho más cerca de ser analógico que un poco, pero aún tiene pasos. Lo que está haciendo la computadora es aproximarse a una cosa analógica en pasos lo suficientemente pequeños como para engañar a la audiencia prevista. Cuando se trata del ojo, 256 niveles diferentes de brillo son suficientes.

Si algo requiere pasos más pequeños para engañar a la audiencia prevista, puede hacer un DAC que tenga 16 bits en el lado digital. Por lo tanto, puede presentar la entrada digital con un número entre 0 y 65535. El lado analógico solo puede variar de completamente apagado a completamente encendido, pero el tamaño de los pasos será mucho más pequeño ya que ahora hay 65536 de ellos.

Cuando se trata del oído, puede escuchar diferencias muy pequeñas, por lo que se requiere un DAC de 16 bits para un sonido de alta calidad.

Todos los sonidos, desde la música hasta el habla y los truenos, son vibraciones del aire. Varían en qué tan rápido vibra el aire y exactamente cómo vibra. El oído humano puede escuchar vibraciones desde aproximadamente 20 Hz en el extremo inferior hasta 20,000 Hz (20 kHz) en el extremo superior, por lo que este es el rango de vibraciones con el que están diseñadas las computadoras. Para que cualquier máquina electrónica produzca sonidos, existe un dispositivo llamado altavoz. Todo lo que hace un orador es moverse hacia adelante y hacia atrás en el aire, haciendo que el aire vibre. Si hace que el aire vibre exactamente de la misma manera que la cosa original que se grabó, sonará igual que la original.

Para almacenar un sonido en una computadora, la posición del altavoz se divide en 65536 posiciones posibles. Luego, un segundo se divide en 44,100 partes. En cada una de esas partes de un segundo, la posición deseada del altavoz se almacena como un número de dos bytes. Esta es información suficiente para reproducir sonido con muy alta calidad.

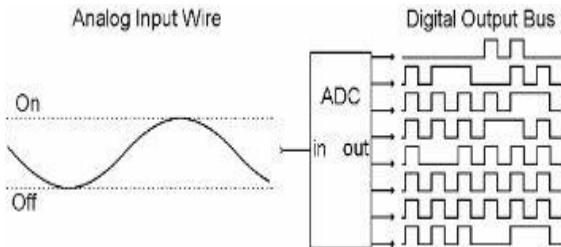
Para reproducir música estéreo de alta calidad, una computadora necesitaría un "periférico de sonido". Esto tendría dos DAC de 16 bits con sus salidas analógicas

conectadas a los altavoces. También tendría su propio reloj que marca 44,100 Hz. En cada tick, obtendría los siguientes dos números de dos bytes y los conectaría al lado digital del

Dacs.

En cuanto a la velocidad, esto sería 176,400 bytes por segundo. Ciertamente eso es rápido, pero recuerde que el reloj de nuestra computadora marca mil millones de veces por segundo. Eso significa que la computadora puede enviar cuatro bytes al periférico de sonido y ejecutar alrededor de 4000 instrucciones en alguna otra tarea antes de que necesite enviar las siguientes cuatro.

Para ir en sentido contrario, existe un "convertidor analógico a digital" o "ADC" para abreviar. Esto se usa para convertir el sonido de un micrófono en una serie de bytes, o para que una cámara convierta una imagen en una serie de bytes. La entrada tiene un cable que puede estar en cualquier lugar, desde el final hasta el final. El ADC convierte sus salidas en un número de 0 a 255 para un ADC de 8 bits o de 0 a 65,535 para un ADC de 16 bits. Este número representa cuánto está activada o desactivada la entrada. La mitad es 128 o 32,768, un cuarto es 64 o 16,384, etc. Este proceso es justo lo



contrario de lo que hace un DAC.

Los DAC y ADC no están hechos de puertas NAND, tienen partes electrónicas como las radios. Cómo hacen lo que hacen no es un tema apropiado para este libro. Entonces, ¿tal vez mentí cuando dije que todo en una computadora está hecho de puertas NAND? Bueno, en realidad no, porque los DAC y ADC solo se usan en ciertos tipos de

periféricos, no en

en la propia computadora.

Divulgación completa

Hemos construido una computadora muy pequeña aquí. Se trata de la computadora más pequeña que se podría inventar y que hace todo lo necesario para ser digna del nombre de computadora. No creo que nadie haya construido una computadora tan pequeña desde aproximadamente 1952, y nadie ha construido esta computadora exacta en el mundo real.

Si un verdadero diseñador de computadoras alguna vez leyera este libro, estoy seguro de que se estaría tirando de los pelos por todas las oportunidades que se han perdido aquí para hacer una máquina mejor. Pero, de nuevo, el objetivo ha sido ilustrar los principios informáticos de la manera más simple posible.

Esta es una computadora de ocho bits. Eso significa que los registros en el procesador son de ocho bits, el bus es de ocho bits y, en esta máquina, incluso el registro de direcciones de memoria es de ocho bits.

Con la mayoría de las computadoras que realmente se construyen, mientras que los bytes individuales en la RAM siguen siendo de 8 bits, todo lo demás se expande a 16 bits, 32 bits o 64 bits o una combinación de estos en diferentes partes de la máquina.

Nuestra RAM solo tiene 256 bytes, que es ridículamente pequeño, pero eso es todo lo que puede tener con un registro de direcciones de memoria de ocho bits. Si usas 16 bits, puedes tener 65.536 bytes de RAM (eso es 64kb), si usas 24 bits puedes tener 16mb, si usas 32 bits puedes tener 4 gigabytes de RAM.

Las computadoras reales tienen cosas que esta no tiene, pero no son capaces de hacer cosas que esta computadora no puede hacer.

En nuestra computadora, si desea desplazar un byte tres bits a la izquierda, pondría tres instrucciones de desplazamiento a la izquierda en su programa. En la mayoría de las computadoras reales, tienen cambiadores que cambiarán cualquier número de bits en una instrucción. Pero el resultado es el mismo, su byte termina luciendo igual en cualquier caso, la computadora real simplemente hace el trabajo más rápido.

En nuestra computadora, el sumador puede sumar dos números de ocho bits. Si desea sumar números de 16 bits, debe

emplear algún software para hacerlo. En la mayoría de las computadoras, el sumador puede agregar números de 16 o 32 bits en una instrucción. Nuevamente, los resultados son los mismos, uno es más rápido que el otro.

El paso a paso en nuestra computadora es una simplificación de algo que tienen la mayoría de las computadoras, llamado 'máquina de estado'. Las máquinas de estado proporcionan pasos, pero inician la siguiente instrucción lo antes posible, hacen lo necesario para un sistema de interrupción, pueden crear instrucciones más complejas, etc. Como todo lo que necesitábamos eran seis pasos consecutivos, construimos algo más simple e inventamos el término 'stepper'.

Así que sí, nuestra computadora es una computadora simple, pequeña y relativamente lenta, pero puede hacer todo lo que pueden hacer las máquinas más complicadas. Las cosas que hacen que una máquina más grande sea más grande, están diseñadas para hacer el trabajo más rápido, hacerlo en menos ciclos de reloj, hacer la misma tarea con menos instrucciones, operar en varios bytes al mismo tiempo. Pero la naturaleza de lo que hacen las máquinas es exactamente la misma. Cada tarea que pueden hacer se reduce a cambiar, AND, ORing, XORing, ADDing y NOTing bytes. No hay otros tipos de operaciones sofisticadas que se hayan dejado fuera de este libro.

En una máquina más grande, puede hacer sumas, restas, multiplicaciones y divisiones en una sola instrucción. Esto se debe a que tienen una gran cantidad de puertas dispuestas en cosas como un 'multiplicador de hardware'. No hay razón para mostrarle los detalles de cómo construir uno de estos, es un trabajo muy complicado para las pocas personas que necesitan construir uno. Es comprensible, y en última instancia, todo se reduce a las puertas NAND como todo lo demás. Pero hemos visto cómo hacer todas las operaciones matemáticas que hay con solo un sumador, una palanca de cambios, NO puertas y algún software. El multiplicador de hardware llega más rápido, pero los resultados son exactamente los mismos.

Las máquinas más grandes tienen más registros, los registros son cada uno de varios bytes, tienen sumadores que pueden sumar tres números al mismo tiempo, pero aún así las instrucciones se reducen a las mismas operaciones simples. Su comprensión de las computadoras

no es pequeña porque hemos visto una computadora pequeña.

Filosofía

¿Por qué tenemos un capítulo llamado "Filosofía" en un libro sobre computadoras? Lo único en este libro que se acerca a ser una pregunta filosófica es su título, "¿Pero cómo lo sabe?" Intentaremos responder a esta pregunta un poco más adelante.

Este libro ha sido sobre las computadoras que tenemos hoy. Pero, ¿qué pasa con el futuro? A medida que las computadoras y el software continúan avanzando, ¿qué tan pronto, si es que alguna vez, llegará el día en que haya robots computarizados que caminen y hablen y se vean y actúen como personas? ¿Llegará el día en que tengamos que decidir si damos o no a estos robots los mismos derechos legales que a las personas? ¿Las computadoras eventualmente se apoderarán del mundo y reemplazarán a las personas por completo?

Para responder a este tipo de preguntas, la gente a menudo se refiere a una pregunta importante que ha estado pendiente en el campo de la filosofía durante muchos años.

La pregunta es, si el hombre está compuesto únicamente del cuerpo estructural que podemos ver y diseccionar, o si hay un componente espiritual integral en cada ser humano que explica las cualidades de conciencia, amor, honor, felicidad, dolor, etc.

Esa pregunta está mucho más allá del alcance de este libro, y sigue siendo respondida de manera poco convincente a pesar de que muchos libros argumentan cada punto de vista. Hay personas en las ciencias que dicen que estamos en camino de construir computadoras conscientes, y sucederá. Hay personas en las humanidades que dicen que es imposible porque no se puede fabricar un espíritu. Cada lado ha sido incapaz de influir en el otro.

Si definimos el cerebro como ese trozo de carne gris de aspecto extraño encerrado por el cráneo, y definimos la mente como lo que sea que sea responsable de la conciencia, la memoria, la creatividad, el pensamiento y todo lo demás que notamos que sucede en nuestras cabezas, entonces podemos replantear la gran pregunta filosófica como: "¿Son el cerebro y la mente una y la misma cosa?"

Luego, cuando se trata de la pregunta sobre la construcción

de un robot humano convincente, habría dos

Posibilidades.

Si el cerebro y la mente son la misma cosa, es posible que no puedas construir una persona sintética hoy, pero a medida que pasó el tiempo, eventualmente podrías comprender cada estructura y función en el cerebro, y construir algo de igual complejidad que generaría una verdadera conciencia, y que realmente debería actuar como cualquier otra persona.

Si el cerebro y la mente no son lo mismo, entonces construir un compañero robot siempre se tratará de simular la humanidad, no de construir algo de igual calidad y valor.

Replantear la pregunta no hace que sea más fácil de responder, pero esta idea de separar lo que sabemos sobre las mentes de lo que sabemos sobre los cerebros puede ser útil.

Al principio, dijimos que íbamos a mostrar cómo funcionan las computadoras para que pudiéramos ver lo que eran capaces de hacer y también lo que no eran capaces de hacer. Vamos a tomar lo que sabemos sobre los cerebros y lo que sabemos sobre las mentes y comparar cada uno individualmente con nuestro nuevo conocimiento sobre las computadoras. Al hacerlo, podemos buscar diferencias y similitudes, y es posible que podamos responder algunas preguntas menos controvertidas.

Las computadoras hacen ciertas cosas con gran facilidad, como sumar columnas de números. Una computadora puede hacer millones de sumas en un solo segundo. La mente apenas puede recordar dos números al mismo tiempo, y mucho menos sumarlos sin lápiz y papel.

La mente parece tener la capacidad de mirar y considerar cantidades relativamente grandes de datos al mismo tiempo. Cuando pienso en mi gato favorito, puedo volver a experimentar ver cómo se ve, escuchar los sonidos de su ronroneo y maullidos, sentir la suavidad de su pelaje y su peso cuando lo levantan. Estas son algunas de las formas en que conozco a mi mascota.

¿Qué significaría para nuestra computadora pensar en un gato? Podría tener imágenes del gato y sonidos del gato codificados en archivos en un disco giratorio o en RAM.

¿Ese pensamiento? Si ejecutara los bytes de estos archivos uno por uno a través de la ALU, ¿estaría pensando? Si pusieras la imagen en la pantalla, ¿sería eso pensar? Si reprodujeras los sonidos en los altavoces, ¿sería eso pensar?

Los sonidos y las imágenes codificados en la computadora son solo patrones de bytes que se encuentran donde están. No se parecen a nada ni suenan como nada a menos que se envíen a los periféricos para los que fueron diseñados.

Y si se envían a la pantalla y a los altavoces, la computadora no los ve ni los escucha. Por supuesto, su computadora podría tener una cámara apuntando a la pantalla y un micrófono escuchando los sonidos, pero la computadora aún no vería una imagen ni escucharía un sonido, solo recopilaría más cadenas de bytes muy similares a las que se envían a la pantalla y los altavoces en primer lugar.

Podría haber programas que realicen operaciones matemáticas en los archivos de imagen para descubrir patrones y almacenar los resultados de estos cálculos en otros archivos. Puede haber archivos que relacionen un archivo de imagen con otros archivos de imagen similares, y las imágenes con los sonidos, etc., creando más archivos.

Pero no importa cuánta programación se aplique a los archivos de imagen, hay algo que la mente puede hacer para lo que la computadora simplemente no tiene ninguna facilidad.

La mente puede considerar la totalidad de algo al mismo tiempo. Puedes pensar en todo el gato a la vez. Es algo así como la diferencia entre la película, la película y la pantalla de televisión. La película tiene imágenes completas, la pantalla del televisor solo tiene un píxel a la vez.

Se podría decir que tu mente funciona tan rápido que no notas los detalles, se integra en un todo al igual que los píxeles se integran en una imagen completa. Pero, ¿qué significa la integración? Y cuando está integrado, ¿qué es y dónde está? ¿Y qué mira el todo integrado?

Acabamos de ver todo lo que hay en una computadora. El equipo se mueve un byte a la vez a través del bus. Lo más elegante que hace es agregar dos bytes en uno. Todo lo

demás que "hace" no es más que el simple almacenamiento de bytes. Un byte almacenado no hace nada más que mantener su propia configuración actual.

Una computadora simplemente no tiene ninguna instalación que integre los elementos de una imagen en cualquier otra cosa, ningún lugar para almacenar esa otra cosa y nada con qué mirarla.

No estoy diciendo que no se pueda construir algo que realizara estas funciones, solo digo que las computadoras tal como las conocemos hoy en día no incluyen actualmente ningún dispositivo de este tipo.

Aquí hay otra pregunta. Si un cerebro funciona como una computadora, entonces necesita tener un programa para que la CPU funcione. ¿De dónde vendría este programa?

Aunque el cerebro tiene billones de células, todo el cuerpo humano comienza con un óvulo fertilizado. Entonces, cualquier programa que tenga el cerebro tendría que estar presente en esta única célula, presumiblemente en el ADN.

Los científicos ahora han decodificado toda la secuencia de ADN de los humanos. El ADN es interesante porque es una larga cadena de solo cuatro tipos de cosas. ¡Es digital! Muchas de las piezas de esta cadena se utilizan para hacer que se produzcan reacciones químicas para producir proteínas, etc., pero la mayoría se llama "ADN basura" porque nadie sabe cuál es su propósito. Pero incluso si considera que la totalidad del ADN está dedicado al software de computadora, entonces podría haber alrededor de mil millones de instrucciones en este programa. Ahora bien, eso es mucho software, pero la computadora doméstica promedio probablemente tenga esa cantidad de software cargado en su disco duro, y eso no sería suficiente para ejecutar a un ser humano.

Algunos han dicho que la computadora humana se programa a sí misma. Como programador, no puedo imaginar cómo funcionaría esto. Si bien es cierto que un programa puede acumular datos y modificar la forma en que funciona en función de los datos recopilados, esto no es lo mismo que escribir un nuevo programa. Si alguien alguna vez escribe este programa que pueda escribir cualquier programa nuevo que necesite, habrá una gran cantidad de programadores de computadoras sin trabajo para siempre.

Luego están los tipos de errores que cometen las

computadoras frente a los que cometen las personas. Si una computadora se atasca en un bucle, parece que se ha detenido por completo.

¿Alguna vez has visto a una persona que camina por la calle dejar de trabajar de repente? Todas las funciones simplemente cesan. La persona simplemente se caía hasta que de alguna manera su computadora se reiniciaba. Las personas colapsan de vez en cuando, pero generalmente es porque alguna otra parte se rompió, como tener un ataque cardíaco, y puedes ver que la persona reconoce el dolor mientras la derriba. Pero si la computadora humana se atascara en un bucle, habría una pérdida instantánea de conciencia y el cuerpo simplemente caería completamente flácido sin lucha. Nunca he visto eso, pero si el cerebro funcionara como una computadora, esperarías verlo con bastante regularidad.

Luego está la cuestión de la velocidad. Como hemos visto, una computadora simple puede hacer mil millones de cosas en un segundo. Cuando se trata del cerebro, tiene nervios que tienen cierta similitud con los cables de las computadoras. Los nervios también pueden transportar electricidad de un lugar a otro. En una computadora, los cables salen de las puertas y entran en otras puertas. En el cerebro, los nervios están conectados entre sí por "sinapsis". Estas sinapsis son espacios entre nervios donde la electricidad en un nervio crea una reacción química, que luego hace que el siguiente nervio cree su propia electricidad. Estas reacciones químicas son dolorosamente lentas.

Nadie ha demostrado que estos nervios estén conectados de manera similar a los cables de una computadora, pero su falta de velocidad hace que sea muy poco probable que sirva de mucho, incluso si las conexiones fueran similares. Después de que la electricidad viaja rápidamente a través de la célula nerviosa, llega a la sinapsis, donde la reacción química tarda aproximadamente una quinienta de segundo en completarse. Eso significa que nuestra computadora simple construida a partir de puertas NAND podría hacer dos millones de cosas en el mismo tiempo que solo una computadora construida a partir de nervios y sinapsis.

Otra área donde la diferencia entre la mente y las computadoras es bastante obvia, es en el área del reconocimiento de rostros. La mente es muy buena en

eso. Si entras en una fiesta con cincuenta personas presentes, sabrás en cuestión de segundos si estás entre un

grupo de amigos o de extraños. Se ha investigado mucho sobre cómo las personas logran esta hazaña y se ha descubierto mucha información muy interesante. También hay mucha especulación y hay muchas teorías fascinantes sobre los principios y mecanismos subyacentes. Pero las estructuras y funciones completas y exactas no han sido descubiertas.

Si le da a una computadora un archivo de imagen de una persona y luego le da el mismo archivo nuevamente, puede comparar los dos archivos byte por byte y ver que cada byte en un archivo es exactamente igual al byte correspondiente en el otro archivo. Pero si le da a la computadora dos imágenes de la misma persona que fueron tomadas en diferentes momentos, o desde diferentes ángulos, o con diferente iluminación, o a diferentes edades, entonces los bytes de los dos archivos no coincidirán byte por byte. Para la computadora, determinar que estos dos archivos representan a la misma persona es una tarea enorme. Tiene que ejecutar programas muy complejos que realizan funciones matemáticas avanzadas en los archivos para encontrar patrones en ellos, luego averiguar cómo se verían esos patrones desde diferentes ángulos, luego comparar esas cosas con todas las demás caras que haya almacenado en su disco, elegir la coincidencia más cercana y luego determinar si está lo suficientemente cerca como para ser la persona o simplemente alguien que se parece.

El punto es que las computadoras tienen un método para tratar con imágenes basado en los principios en los que funcionan las computadoras. El uso de estos principios por sí solo aún no ha producido computadoras o software que puedan reconocer una cara con la velocidad y precisión de cualquier persona común.

El reconocimiento de voz por parte de las computadoras es otra tecnología que ha recorrido un largo camino, pero tiene mucho más por recorrer para rivalizar con lo que la mente hace fácilmente.

Entonces, al comparar una computadora con un cerebro, simplemente no parece muy probable que operen con los mismos principios. El cerebro es muy lento, no hay ningún lugar para obtener el software para ejecutarlo y no vemos los tipos de problemas que esperaríamos con

los errores de software de computadora.

Al comparar una computadora con la mente, la computadora es

Es mucho mejor en matemáticas, pero la mente es mejor para tratar con rostros y voces, y puede contemplar la totalidad de alguna entidad que ha experimentado previamente.

Los libros y películas de ciencia ficción están llenos de máquinas que leen mentes o implantan ideas en ellas, naves espaciales con computadoras parlantes incorporadas y robots y androides realistas. Estas máquinas tienen diferentes capacidades y algunas de las tramas tratan sobre el robot que lucha con la conciencia, la autorrealización, las emociones, etc. Estas máquinas parecen sentirse menos que completas porque son solo máquinas y quieren desesperadamente convertirse en completamente humanas. Es una especie de versión adulta del clásico infantil "Pinocho", la historia sobre una marioneta que quiere convertirse en un niño de verdad.

Pero, ¿sería posible construir tales máquinas con una versión enormemente ampliada de la tecnología que usamos para construir nuestra computadora simple?

El optimismo es una gran cosa, y no debe ser aplastado, pero un problema no será susceptible de solución si está utilizando una metodología o tecnología que no está a la altura de ese problema. En el campo de la medicina, algunas enfermedades han sido eliminadas por los antibióticos, otras pueden prevenirse mediante inoculaciones, pero otras aún afectan a la humanidad a pesar de la mejor atención y décadas de investigación. Y ni siquiera veamos temas como la política. Tal vez todo lo que se necesita es más tiempo, pero también hay que considerar la posibilidad de que estos problemas sean irresolubles o que la investigación haya estado buscando la respuesta en los lugares equivocados.

Como ejemplo, muchas visiones del futuro han incluido a personas que viajan en autos voladores. En realidad, se han construido varios tipos de autos voladores. Pero son caros, ineficientes, ruidosos y muy peligrosos. Trabajan con los mismos principios básicos que los helicópteros. Si dos autos voladores tienen algún tipo de accidente menor, todos morirán cuando ambos autos se estrellen contra la Tierra. Por lo tanto, la tecnología de aviación actual simplemente no dará como resultado un automóvil volador satisfactorio. A menos que alguien invente un dispositivo antigravedad barato y confiable,

no habrá un mercado masivo para los autos voladores y el tráfico en las carreteras no se aliviará.

Si quieres construir una máquina que funcione como una persona, ciertamente la mejor manera de hacerlo sería averiguar cómo funciona la persona y luego construir una máquina que funcione con los mismos principios, tenga partes que hagan las mismas cosas y esté conectada de la misma manera que una persona.

Cuando Thomas Edison inventó el fonógrafo, estaba tratando el tema del sonido. El sonido es una vibración del aire. Así que inventó un aparato que capturaba las vibraciones en el aire y las transformaba en una ranura vibratoria en la superficie de un cilindro de cera. El sonido podría recrearse transfiriendo las vibraciones en el surco al aire. El punto es que, para recrear el sonido, descubrió cómo funcionaba el sonido y luego hizo una máquina que funcionaba según el mismo principio. El sonido es una vibración, el surco de un fonógrafo es una vibración.

Se han realizado muchas investigaciones sobre el tema de lo que motiva a las personas. Se han realizado muchas investigaciones sobre el tema de cómo hacer que las computadoras hagan las cosas que hace la gente. Se han descubierto muchas cosas y se han inventado muchas cosas. No quiero minimizar el trabajo realizado ni los resultados logrados en estas áreas.

Pero hay muchas cosas que aún no se han descubierto o inventado.

Se han diseccionado muchos cerebros muertos y se han estudiado y clasificado sus partes. El cerebro contiene células nerviosas que mueven la electricidad de un lugar a otro. Esta es una similitud entre cerebros y computadoras. Pero la investigación sobre el funcionamiento real de los cerebros humanos vivos es necesariamente limitada. La mayoría de las observaciones se han realizado durante cirugías que fueron necesarias por accidente o enfermedad. Se han hecho muchas observaciones de cambios en el comportamiento después de que una lesión o enfermedad ha desactivado ciertas partes del cerebro. A partir de esta investigación, se ha podido asociar ciertas funciones con ciertas áreas del cerebro.

Pero nadie ha descubierto un bus, un reloj, ningún registro, una ALU o RAM. El mecanismo exacto de la memoria en el cerebro sigue siendo un misterio. Se ha demostrado que los nervios desarrollan nuevas conexiones

con el tiempo, y se supone que este es el mecanismo de aprendizaje, pero nadie lo ha sido

Capaz de decir que este nervio en particular hace esta función exacta, como podemos hacer con los cables individuales en una computadora.

Todo lo que entra en una computadora se convierte en un código u otro. El teclado genera un byte de ASCII por pulsación de tecla, un micrófono genera 44.100 números binarios por segundo, una cámara a color genera tres números binarios por píxel, 30 veces por segundo, y así sucesivamente. Nadie ha aislado el uso de códigos como ASCII, números binarios, fuentes o un código de instrucción en el cerebro. Pueden estar allí, pero no han sido aislados. Nadie ha rastreado un pensamiento o localizado un recuerdo de la misma manera que podríamos seguir el funcionamiento de un programa en una computadora.

Se asume ampliamente que el cerebro funciona de una manera mucho más dispersa que una sola computadora, que hay miles o miles de millones de elementos informáticos que cooperan y comparten el trabajo. Pero tales elementos aún no han sido localizados. En el mundo de la informática, esta idea se llama "procesamiento paralelo" y se han construido computadoras con docenas o cientos de CPU. Pero estas computadoras aún no han resultado en un sustituto humano.

Piensa en todo como un rompecabezas. La forma en que trabaja la gente es un lado del rompecabezas. Hacer que las computadoras hagan cosas que la gente hace es el otro lado del rompecabezas. Las piezas del rompecabezas se están ensamblando en ambos lados. El problema es que a medida que se avanza en ambos lados, parece cada vez más que estos son dos acertijos diferentes, no se unen en el medio. No están convergiendo en una sola imagen.

Los investigadores son muy conscientes de estos desarrollos. Pero cuando se trata de la cultura pop, la gente oye hablar de nuevos inventos todo el tiempo, y ve el futuro retratado en las películas de ciencia ficción, y la conclusión lógica parece ser que la investigación continuará resolviendo los problemas uno por uno hasta que en 10, 20 o 30 años tengamos nuestros amigos electromecánicos. En el siglo pasado conquistamos la electricidad, el vuelo, los viajes espaciales, la química, la energía nuclear, etc. Entonces, ¿por qué no el cerebro y / o la mente?

La investigación, sin embargo, aún se encuentra en la etapa en la que cada vez que se encuentra una nueva respuesta, se crea más de una nueva pregunta.

Así que parece que de cualquier manera que lo miremos, ni el cerebro ni la mente funcionan con los mismos principios que las computadoras tal como las conocemos. Digo "tal como los conocemos" porque en el futuro se puede inventar algún otro tipo de ordenador. Pero todas las computadoras que tenemos hoy en día entran en la definición de 'Computadoras digitales de programa almacenado', y todos los principios sobre los que operan se han presentado en este libro.

Aún así, nada de esto "prueba" que nunca se pueda construir un humano sintético, solo significa que los principios informáticos presentados en este libro no son suficientes para el trabajo. Algún tipo de dispositivo completamente diferente que funcione con un conjunto de principios completamente diferente podría hacerlo. Pero no podemos comentar sobre un dispositivo de este tipo hasta que alguien invente uno.

Volviendo a una pregunta más simple, ¿recuerdas a Joe y la botella termo? Pensó que el termo tenía algún tipo de sensor de temperatura, y un calentador y un enfriador en el interior. Pero incluso si hubiera tenido toda esa maquinaria, todavía no "sabía" qué hacer, solo sería un dispositivo mecánico que encendía el calentador o enfriador dependiendo de la temperatura de la bebida colocada en él.

Un par de tijeras es un dispositivo que realiza una función cuando se le hace hacerlo. Pones un dedo y un pulgar en los agujeros y aprietas. Las hojas en el otro extremo de las tijeras se mueven juntas y cortan un poco de papel o tela o lo que sea que hayas colocado en su camino. ¿Las tijeras "saben" cómo cortar formas de papel o cómo hacer un vestido de tela? Por supuesto que no, simplemente hacen lo que se les dice.

Del mismo modo, las puertas NAND no "saben" lo que están haciendo, simplemente reaccionan a la electricidad o la falta de electricidad colocada en sus entradas. Si una puerta no sabe nada, entonces no importa cuántas de ellas conectes, si una de ellas sabe absolutamente cero, un millón de ellas también sabrán cero.

Usamos muchas palabras que dan características humanas a nuestras computadoras. Decimos que "sabe" cosas. Decimos que "recuerda" cosas. Decimos que "ve" y "entiende". Incluso algo tan simple como un adaptador de dispositivo

"escucha" su dirección para que aparezca en la E/S

autobús, o una instrucción de salto "decide" qué hacer. No hay nada de malo en esto siempre que sepamos la verdad del asunto.

Ahora que sabemos lo que hay en una computadora y cómo funciona, creo que es bastante obvio que la respuesta a la pregunta "¿Pero cómo lo sabe?" es simplemente "¡No sabe nada!"