Author. Ignacio Pachés Giner

Introducción al Hardware de los computadores

Contenido

GOBIERNO DE ESPAÑA · MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE · SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES · Unión Europea · Fondo Social Europeo · El FSE invierte en tu futuro

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Author. Ignacio Pachés Giner

Introducción

Es difícil entender cómo funciona un computador. En los tutoriales y explicaciones de informática, muchas veces se dan por supuestos ya unos conocimientos previos cuando se trata algún tema. No hay apenas libros que supongan que el lector no tiene absolutamente idea de nada de informática.

En este libro se tiene en cuenta siempre que el lector no tenga ningún conocimiento previo y se van a tratar conceptos de forma incremental sin que haga falta acudir a otros libros o información externa para ser comprendidos. Esta intención no sale gratis, y es imposible que en un libro se explique todo. Por ello, se han aplicado algunas relajaciones y licencias:

- Se omitirán conceptos que supongan una complejidad importante respecto al protagonismo que tienen en la comprensión de la idea de computador. Si algo es muy largo o complejo y sirve poco, se deja fuera
- Se simplificará la realidad del computador, incurriendo en incorrecciones o falsedades parciales, si con ello se puede progresar con facilidad.
- Se utilizarán ejemplos aproximados de la vida real para entender ciertos conceptos.
- El lector deberá aceptar la valoración positiva o negativa que se indica sobre ciertos elementos, aunque no sea aparente y crea que no hay algo que mejorar. Si algo se dice que es "bueno" o "malo" debe creerse así.
- Es necesario que el lector se olvide de lo que pueda saber de antemano y que no intente encontrar aplicación o relación con la realidad a cada concepto que se va tratando. No sustituyas lo que aquí se cuenta con tus conocimientos hasta el final del tema.
- En muchas de las explicaciones se asume que alguien está usando el computador. Llamaremos "usuario" a ese alguien.
- Aunque se explicará todo, es inevitable tener que tratar algunos conceptos de forma retrasada o teniendo que dar mucho tiempo a conceptos previos.
- En ocasiones se hacen definiciones escuetas y difíciles de entender, que después se explican con ejemplos y detenidamente. No te desanimes si una frase es difícil de entender, a veces después viene la explicación.
- Muchas de las explicaciones están basadas en un ejemplo y el ejemplo esa dibujado. Es muy importante leer y ver el dibujo para asimilar los ejemplos.

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Author. Ignacio Pachés Giner

Tratamiento microscópico y macroscópico de los conceptos.

Imagínate que estás hablando por teléfono con un amigo y ocurre la siguiente conversación:

Amigo- "¿Qué hace tu hermano Pepe?"

Tú - "Está estudiando ASIR"

A- "Pero ¿Qué hace ahora?"

T- "Está en primero"

A- "Pero ¿Qué hace ahora?"

T- "Está en casa"

A- "Pero qué ¿Hace ahora ?"

T- "Está haciendo su cena"

A- …

Tú- "Está lavando un plato"

A- …

Tú- "Está fregándolo con el estropajo "

Tú- "Está moviendo la mano con el estropajo"

Tú- "Está activando el músculo del brazo que levanta el brazo. Tú-

"Está …

En esa conversación, la pregunta ha sido siempre la misma ("¿Qué hace tu hermano?"), y la respuesta ha ido cambiando, pero no respondiendo cosas diferentes, sino respondiendo sobre franjas temporales más pequeñas cada vez. La primera respuesta es **macroscópica** en el tiempo. En ella se responde por lo que lleva haciendo Pepe durante varios meses. Conforme avanza la conversación y las preguntas se repiten, las respuestas son cada vez más concretas, pero no porque den más información sobre lo que ya se ha respondido, sino porque comprenden espacios de tiempo más pequeños, son más **microscópicas** en el tiempo .

Seguramente si la última respuesta se hubiese dado al principio, el amigo estaría desconcertado con tu respuesta, y sólo alcanzaría a entenderlo todo, después de escuchar en orden inverso las respuestas.

En este tema se manifiestan **notables diferencias** en el tratamiento temporal de lo que se explica. Hay conceptos muy microscópicos y otros que son más macroscópicos . En todo momento hay que

GOBIERNO DE ESPAÑA · MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE · SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES · Unión Europea · Fondo Social Europeo · El FSE invierte en tu futuro

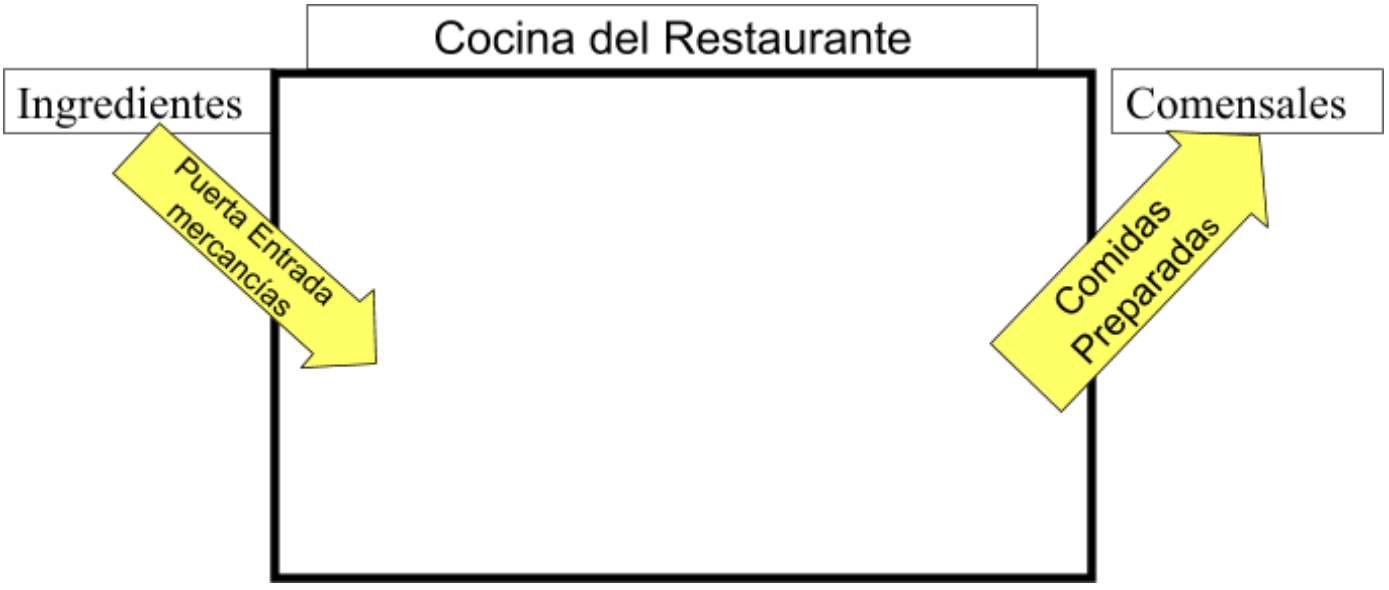
saber en qué nivel temporal está cada explicación y saber dónde situarse para no confundirse como podría ocurrir en la conversación de ejemplo anterior si no se responde en orden.

En el computador hay cosas que los humanos podemos apreciar claramente (por ejemplo, cuando guardamos un documento vemos que el disco tiene actividad durante unos instantes), otras que nos cuesta percibir (a veces el disco se activa pero no sabemos por qué), y otras en las que es imposible que nos demos cuenta de lo que ocurre. En este último caso, el nivel microscópico de los eventos que ocurren, hace que no podamos percibirlo los humanos.

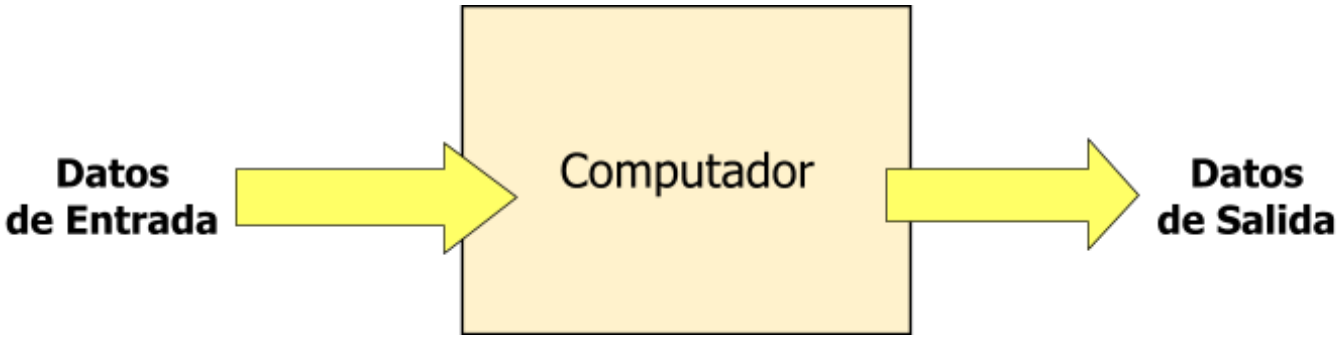
En este tema vamos a explicar las cosas a varios niveles temporales. Y además no vamos a ir de más a menos o al revés siempre. En ocasiones las percepciones macroscópicas tienen una explicación microscópica. Pero otras veces los eventos y funcionamiento microscópico no se manifiestan claramente y no hay que "rascar" para intentar corroborar todo lo que se cuenta con lo que puedes observar. Es importante que te situes en el nivel de detalle temporal de cada concepto para no desconcertante como podría pasarle al amigo que recibe la respuesta que no espera.

El computador como un restaurante

La mejor manera de entender qué es un computador, es observando cómo funciona la cocina de un restaurante, porque resulta ser una analogía sorprendentemente precisa y aproximada.



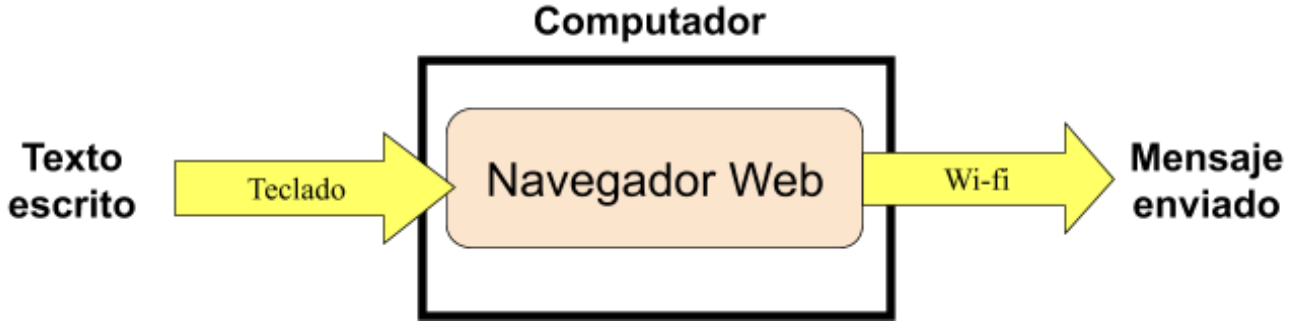
La cocina de un restaurante es un lugar al que llegan unos ingredientes o alimentos básicos y del que salen unos platos preparados. Esta definición tan simple, es válida para alguien que está fuera de la cocina y no sabe qué hay dentro ni cómo funciona, sólo ve lo que entra y lo que sale. Esta visión de una cocina es la que tienen los clientes del restaurante.



En un computador ocurre algo similar, pero lo que entran son datos y lo que se emite al exterior son datos también. Evidentemente, los datos que llegan al computador son captados desde el exterior y sufren algún tipo de transformación que da lugar a los datos que después el computador emite (igual que la comida se transforma en una cocina).

Cualquier actividad que el computador realice está dirigida a tomar datos de algún lado y emitir datos por otro lado. Podemos poner algunos ejemplos para validar esta afirmación.

1. Un usuario abre el ordenador para navegar por internet. El computador toma datos que le llegan por algún cable o señal inalámbrica, los procesa y transforma, y muestra una página web por la pantalla del ordenador como resultado.
2. Un usuario escucha música usando el ordenador. Los datos que llegan al usuario, se convierten en sonidos, pero siguen siendo datos que el computador emite al exterior. Y en este caso, los datos de entrada están almacenados en un disco de música (compact disc), o en alguna unidad de almacenamiento de datos.
3. Un usuario escribe un mensaje de correo electrónico y lo envía. En este caso, los datos no son emitidos desde el computador con la intención de llegar al usuario, sino que se emiten por algún medio para que lleguen lejos en internet. En este caso el ordenador capta los datos del usuario con el teclado.

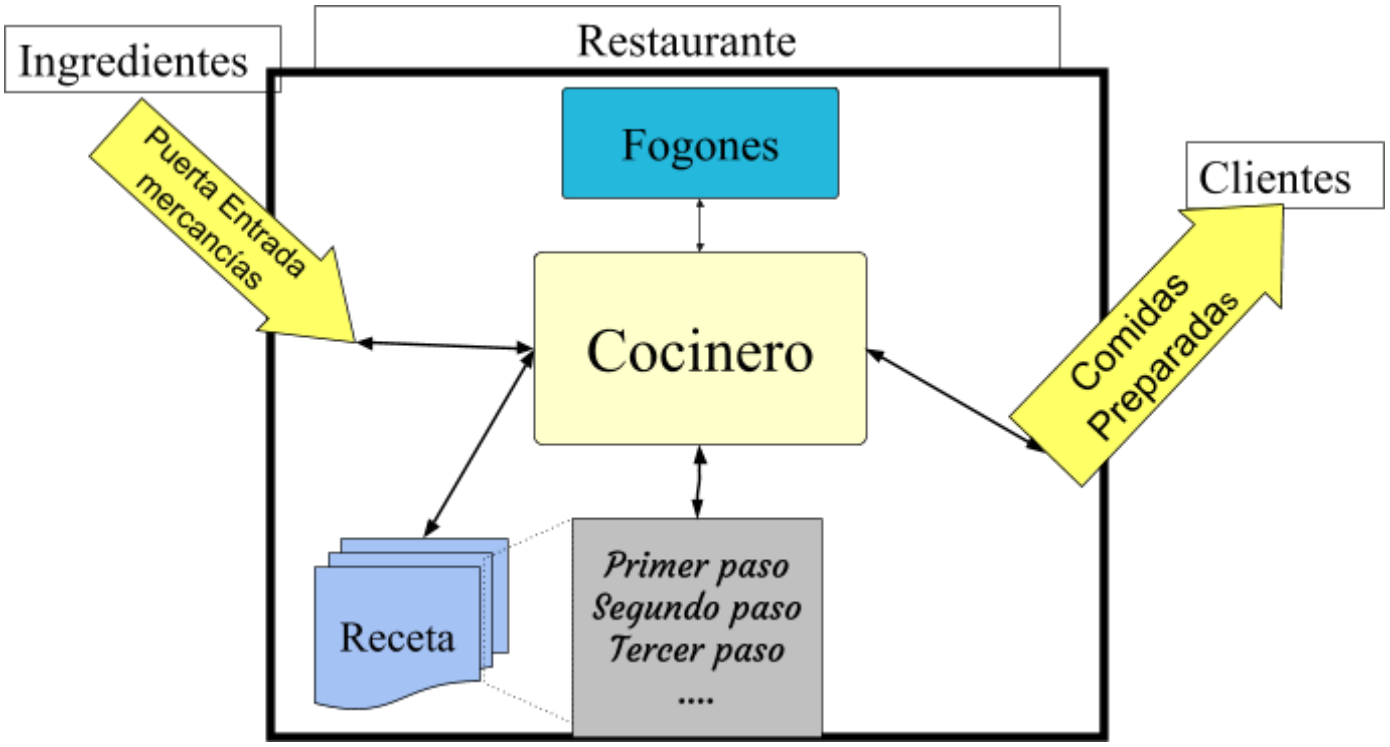


GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Author. Ignacio Pachés Giner

El interior de la cocina.

Hagamos un esquema de los elementos de la cocina y analicemos su funcionamiento



Encontramos los siguientes elementos:

- Una puerta de entrada por la que llegan las mercancías o ingredientes de los platos que la cocina prepara
- Un libro de recetas, donde están explicados los pasos para elaborar los platos
- Cada receta está compuesta de varios pasos simples.
- Unos fogones (también hornos, batidoras, máquinas de cortar, etc.). Allí es donde se transforma físicamente unos alimentos en otros, según el elemento o procedimiento empleado: freír, hervir, licuar, rayar, hornear, etc.
- El cocinero es el que actúa para que la cocina funcione.
- Una puerta o ventana de salida por el que los camareros recogen los platos y se los llevan al cliente.

GOBIERNO DE ESPAÑA · MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE · SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES · Unión Europea · Fondo Social Europeo · El FSE invierte en tu futuro

Author. Ignacio Pachés Giner

No creas que el cocinero es alguien que realiza una labor muy compleja y de gran sabiduría. Al revés, siempre realiza los mismos simples y "tontos" pasos:

1. Se dirige al libro de recetas, y concretamente , a un paso concreto de una receta concreta.
2. Lee el paso concreto, lo intenta entender y lo retiene en su mente.
3. Quizá la receta le indique que debe coger algún ingrediente o alimento para hacer algo con él. Lo siguiente pues, es buscar el alimento y tomarlo allá donde esté guardado
4. Ejecuta el paso que ha leído y espera que se termine totalmente dicho paso (a veces ha de esperar a que algo se termine de freír, por ejemplo)
5. Cuando termina, guarda o aparta lo que esté preparando y da por concluido el paso.
6. Vuelve al punto uno , pero el paso que buscará en la receta que está preparando es el paso siguiente al que acaba de ejecutar.

¿Cuál es el elemento más importante?

Es muy interesante hacerse esta pregunta. Está claro que todos los elementos son necesarios y si eliminamos alguno, ya no hay computador/restaurante que valga. Pero ¿Qué dirías que es lo más importante?

- ¿Los datos de entrada? Porque son la materia física que se transforma y de la cual se obtiene el resultado final.
- ¿Los platos ya preparados? Porque son lo que el cliente ha pedido y por el cliente es por lo que se monta un restaurante
- ¿El cocinero? Porque sin él nada funciona y no se obtendría ningún resultado o producto final
- ¿Los fogones e instrumentos? Porque es allí donde se pueden transformar unos alimentos en otros

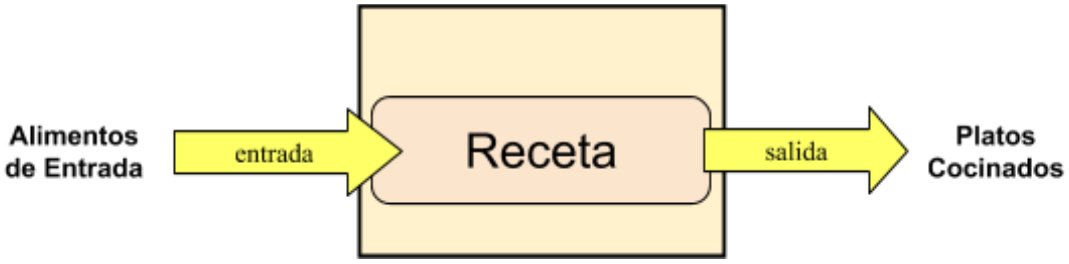
No, nada de eso es tan importante como las recetas . Las recetas son la sabiduría .
Todos los demás elementos están ahí para ejecutar los pasos de las recetas, para que se materialicen. Se puede cambiar de cocinero, de cocina, de fogones, etc. además siempre existirán tomates, patatas, carne, etc. listos para ser comprados y cocinados, pero si se ejecutan bien las recetas se le da carácter a un restaurante. Un restaurante se puede distinguir de otro por sus recetas. Sin recetas no hay nada

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea Fondo Social Europeo El FSE invierte en tu futuro

Author. Ignacio Pachés Giner

Simplifiquemos...

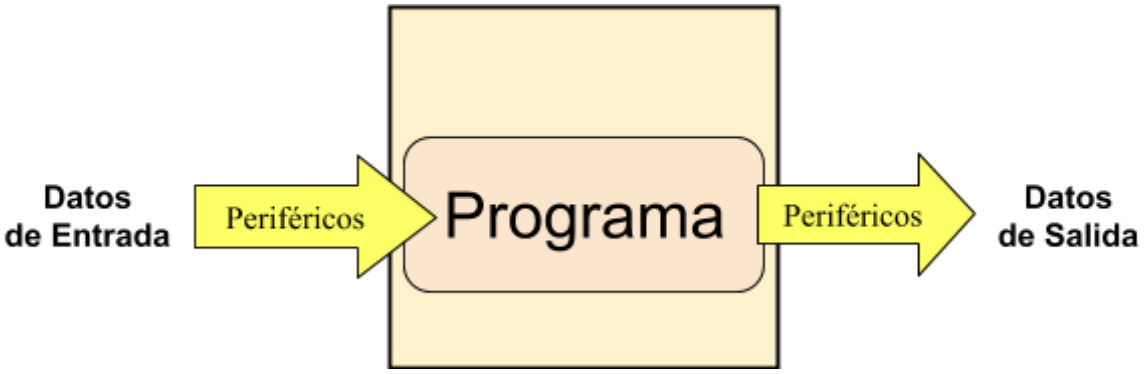
Por lo tanto, podemos ahora replantearnos con simplicidad la visión de una cocina acorde al siguiente diagrama que describe por sí sólo la función de una cocina:



Una cocina es "algo" que recibe unos alimentos de entrada, los procesa acorde a una receta y los ofrece al cliente.

¿Cuál es la función básica de un computador?

Un computador es "algo" que recibe unos datos de entrada, los procesa acorde a un programa y los emite al exterior (para que un usuario los reciba)



¿Y qué es un programa?

Un programa es una sucesión de instrucciones que indican cómo deben tratarse los datos… Casi es mejor que no te enteres ahora. Lo irás viendo conforme vayamos explicando el funcionamiento de las partes internas del computador. y un ¿Periférico?

Un periférico es un dispositivo electrónico capaz de captar cualquier dato del exterior y transmitirlo al interior del computador, o tomar un dato del computador y emitirlo al exterior en forma de imagen, sonido, movimiento, registro en un papel, en un disco, etc.

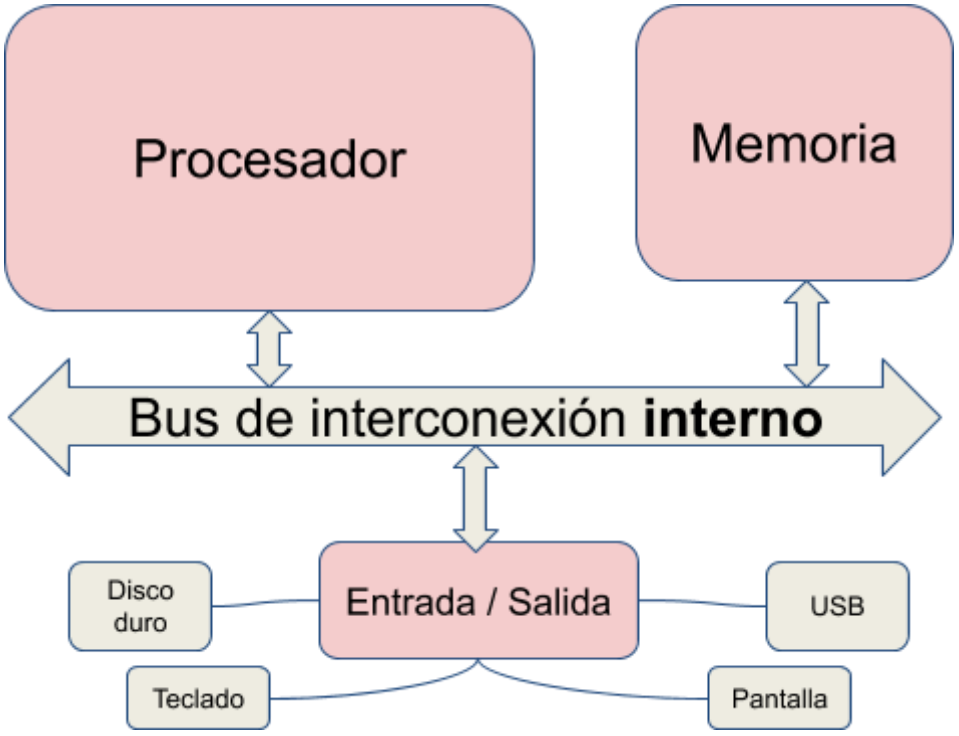
Author. Ignacio Pachés Giner

Partes internas de un computador

El diagrama anterior es una simplificación que muestra la función del computador, pero no sus componentes internos. Si abres un ordenador y lo examinas verás muchos "cacharros", placas, chips, etc. A veces varios elementos son similares o son partes de un componente mayor. Básicamente un computador tiene los siguientes elementos:

- "Procesador"
- "Memoria"
- Periféricos de entrada y de salida
- Un manojo de hilos que interconecta la memoria con la CPU y con los periféricos.

Vamos a representar estos elementos así:



Antes de complicar más las cosas, hemos de matizar el diagrama éste. Como verás, los diferentes dispositivos están conectados a una caja llamada "Entrada / Salida". Lo que conocemos como Disco duro, pantallas, teclados, etc. no son parte del computador, sino que son **periféricos externos** al computador en sí. Hoy en día, muchos de esos elementos están dentro de la caja que conocemos como "ordenador", pero en el fondo esto se hace así por una cuestión de fabricación, costes y rapidez. Si todos los computadores llevan un disco duro, ¿Por qué no fabricarlo ya metido en el ordenador?

Los periféricos son elementos muy complejos, pero no necesitamos tratarlos para entender el ordenador, por tanto los agruparemos bajo esa caja "E/S" y no los explicaremos en este tema y los consideraremos poco importantes. En muchos diagramas no dibujaremos ni a los periféricos ni la "Entrada/Salida" que representa su conexión con el computador. Sin embargo, no has de perder de vista que siempre están ahí conectados realmente.

Gobierno de España · Ministerio de Educación, Cultura y Deporte · Secretaría de Estado de Educación, Formación Profesional y Universidades · Unión Europea · Fondo Social Europeo · El FSE invierte en tu futuro

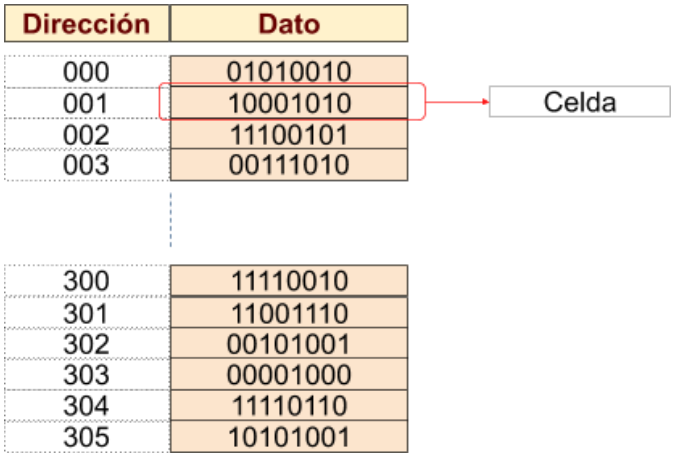
Author. Ignacio Pachés Giner

La memoria.

La memoria del computador es un dispositivo bastante simple. Se trata de un conjunto de celdas que pueden almacenar un dato cada una. El dato que una celda puede almacenar tiene 8 bits. (No confundas la memoria del ordenador con el disco duro, con un disco USB, con el "firmware" o con la "ROM"). Se la llama por cuestiones históricas "RAM"

Estructura de la memoria

Las celdas están situadas de forma que cada una tiene una dirección que se utiliza para distinguirlas del resto. El contenido de cada celda puede ser cualquiera (mientras quepa en 8 bits) pero cada celda tiene siempre la misma dirección. Esto se asemeja a los números de casa en una calle. Cada casa tiene siempre el mismo número, independientemente de quien viva en ella. Así, el cartero de correos usa este número para localizar la casa donde debe entregar algo.



En el diagrama anterior hemos escrito las direcciones en decimal y los contenidos en binario (con 0s y 1s como realmente se almacena en los circuitos). Simplemente, se quiere resaltar que el contenido de las celdas es de ocho bits (ocho dígitos binarios) y que -de momento- el significado de cada dato no nos interesa. En otras ocasiones no escribiremos el contenido en binario, sino que interpretaremos lo que ahí exista. Es decir, podremos suponer que en tal dirección hay almacenada una letra, número o dato inventado y los escribiremos en lugar de esos feos números en binario. Las direcciones de las celdas están en decimal para que nos sea fácil hacernos una idea de cómo está organizada la memoria

Cuando un usuario está usando un ordenador y está, por ejemplo, viendo una imagen. Los datos de la imagen están en la memoria en el momento en el que se están viendo. Si tuviesemos la capacidad de ver el contenido de la memoria con algún visor, sólo veríamos ceros y unos, pero podríamos localizar un bloque de celdas de memoria que están almacenando la imagen que el usuario está viendo en ese momento. En esas celdas está la informacioń de la imagen.

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Author. Ignacio Pachés Giner

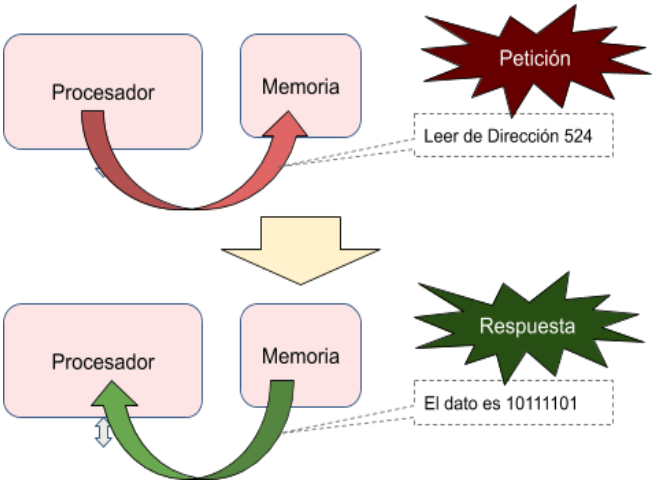
Operaciones con la memoria

La memoria es un dispositivo muy simple y muy poco inteligente, cuya función es obedecer al procesador y llevar a cabo las únicas dos operaciones que se le encargan:

- Lectura de una sola celda de memoria
- Escritura en una sola celda de memoria

Lectura

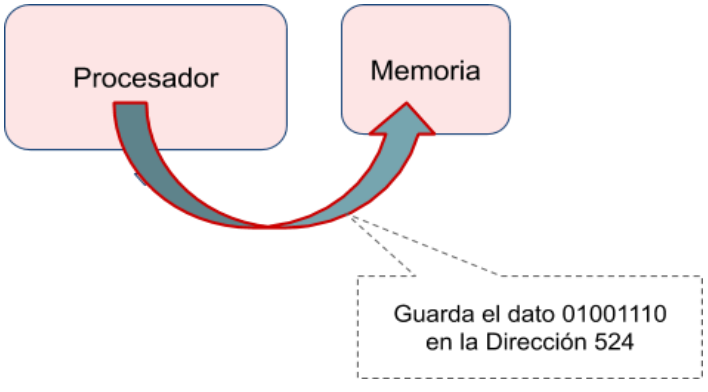
En la lectura, el procesador le transmite a la memoria una dirección y la orden para leer. La memoria devolverá el dato que estaba almacenado en esa dirección de memoria



Si te preguntas qué le ocurre a la celda de memoria después de su lectura es que ya estás pensando que la memoria es más compleja de lo que es. Una celda que es leída se queda con el mismo contenido. No se borra o cambia por ser leída.

Escritura

La escritura es una operación que ordena el procesador a la memoria y que no recibe ningún tipo de respuesta por parte de ésta. El procesador indica el dato a guardar y la dirección dónde guardarlo y la memoria escribe ese dato en la celda de esa dirección.



GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

¿Qué ocurre si la celda ya tenía un dato? ¿Qué le ocurre a ese dato? Son preguntas que la memoria ni se plantearía. La memoria obedece y punto. Lo que hubiese ahí antes, se pierde.

La memoria no puede realizar por sí misma ninguna de las siguientes acciones:

- Buscar un dato
- Indicar si una celda está libre u ocupada
- Indicar la última dirección leída o escrita.
- Rechazar una operación de lectura o escritura
- Almacenar un dato en la siguiente celda de memoria
- Borrar una celda de memoria
- Mover los datos

El procesador

Empezaremos a llamarlo también CPU o UCP. La mayoría de libros y explicaciones tratan al procesador como "un dispositivo compuesto de diferentes partes" pero realmente el procesador no debe verse como un dispositivo, sino como varios distintos e independientes que se fabrican juntos por conveniencia, y que por tanto se venden y diseñan juntos.

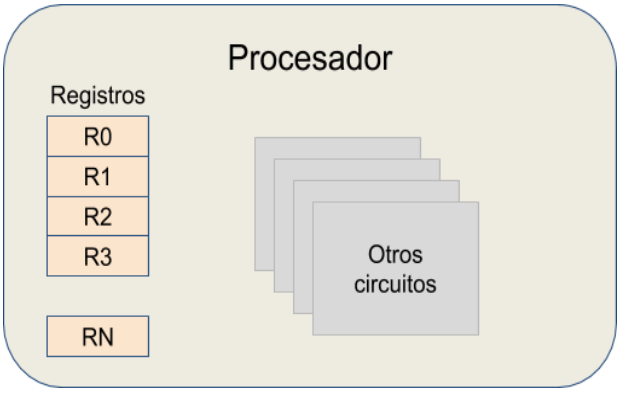
El procesador realmente está compuesto principalmente de:

- Una Unidad de control
- Un banco de registros
- Una unidad aritmético-lógica

El banco de registros

Se trata de una pequeña memoria compuesta de unos pocos registros, donde pueden guardarse datos de forma similar a como ocurre con la memoria. Los registros están también numerados o direccionados. Sin embargo, al ser pocos (en nuestros ejemplos, como mucho 8) reciben el nombre de "R0", "R1", etc. donde la "R" es por ser un "Registro". Recuerda: los registros están en el procesador.

Cuando un dato es leído de un registro, el contenido no cambia. Cuando se realiza una escritura, el contenido existente antes de la misma se pierde. Los registros podemos representarlos simplemente así:

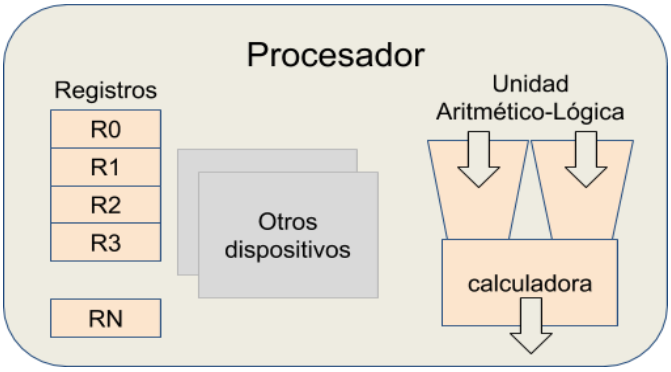


Author. Ignacio Pachés Giner

La unidad Aritmético-Lógica "ALU" o "ULA"

La unidad aritmético-lógica (UAL, ALU, ULA) es un circuito capaz de realizar cálculos matemáticos y lógicos sobre datos. Es como una calculadora que sabe sumar, restar, multiplicar, comparar, y otras operaciones que veremos más adelante.

La unidad aritmético lógica recibe uno o dos datos y una indicación de la operación a realizar. Con ello, es capaz de realizar un cálculo, obtener un resultado y devolverlo para que sea tratado o almacenado en otra parte (memoria o registros). La ALU no es la encargada de buscar datos en la memoria o en los registros, sino que algo debe "llevarle" los datos e indicarle la operación. Si ampliamos el dibujo del procesador que vamos descubriendo observaremos que el dibujo



El dibujo de la ALU hecho aquí es aproximado, e indica que tiene dos entradas para tomar dos números y realizar una operación cuyo resultado al exterior por una salida. Eso es lo que las flechas representan.

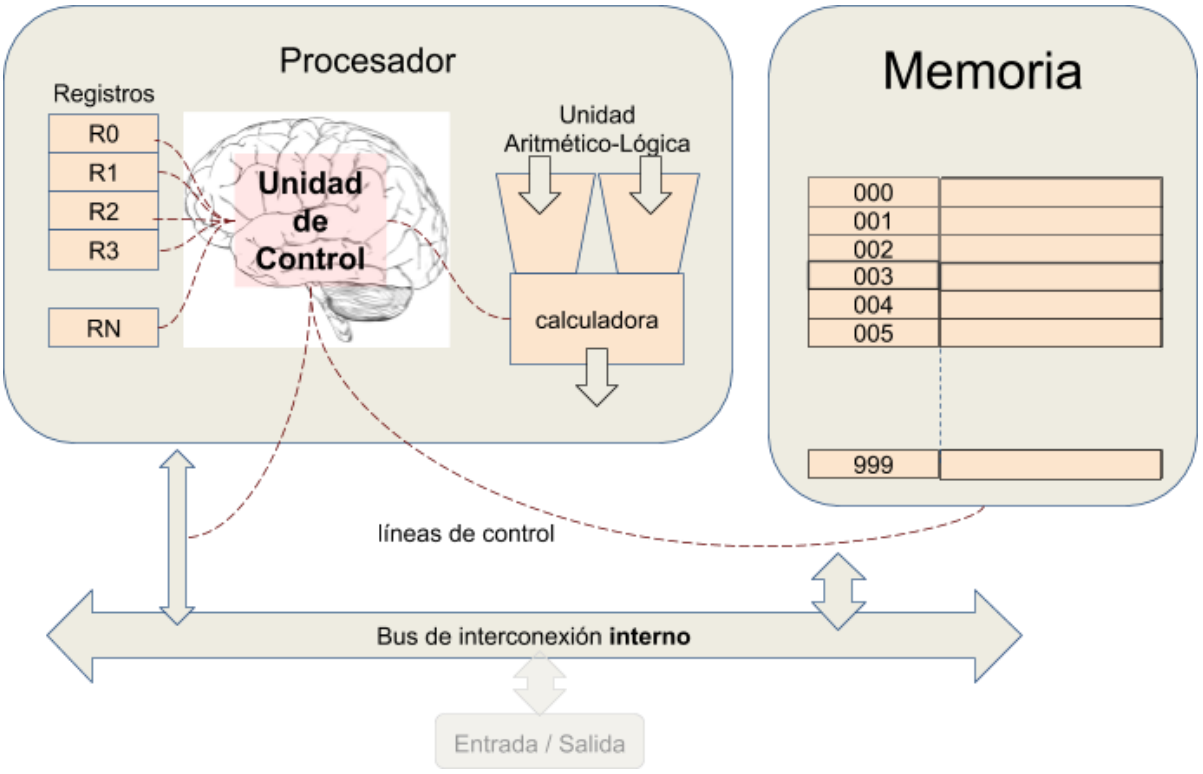
La unidad de control

La unidad de control ("UC") Lo gobierna todo. Ella está conectada con todos los dispositivos importantes del computador: registros, memoria, ULA. Ordena a los demás dispositivos que ejecuten las funciones para las que han sido pensados. Es como el cerebro de todo. La UC:

- Es la que le dice a la memoria que escriba un dato en una dirección. Y, por supuesto, le proporciona el dato y la dirección… y tiene sus motivos siempre
- Reclama un dato de una dirección a la memoria (lee de la memoria)
- Lee y escribe en cualquier registro del banco de registros
- Proporciona datos a la ULA y le indica qué operación matemática hay que realizar, recoge el resultado de ella.
- Todos los datos que lee o escribe puede tomarlos o dejarlos de cualquier otro lugar. Por ejemplo, puede tomar un dato de un registro y copiarlo a una posición de la memoria. ● … más funciones que iremos descubriendo.

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Author. Ignacio Pachés Giner



La pregunta clave es: ¿Cómo sabe la UC qué acciones ha de ordenar a cada momento a cada elemento? Ufff! esta pregunta es realmente difícil de contestar. Vamos a tomarnos un tiempo y recuperar el ejemplo de la cocina del restaurante. En ella, lo más parecido a la Unidad de control es el cocinero. Es el que "mueve" las cosas por allí. ¿Cómo sabe el cocinero qué es lo que hay que hacer en cada momento?... ahí está la clave. El cocinero no lo sabe , ¡ Lo lee del libro de recetas ! El cocinero hace lo que la receta le indica a cada paso. El cocinero sólo debe leer un paso de receta y hacer lo que allí se indica, y nada más. ¡Tú podrías ser cocinero, si supieses leer e interpretar el lenguaje de las recetas de "cocina"!

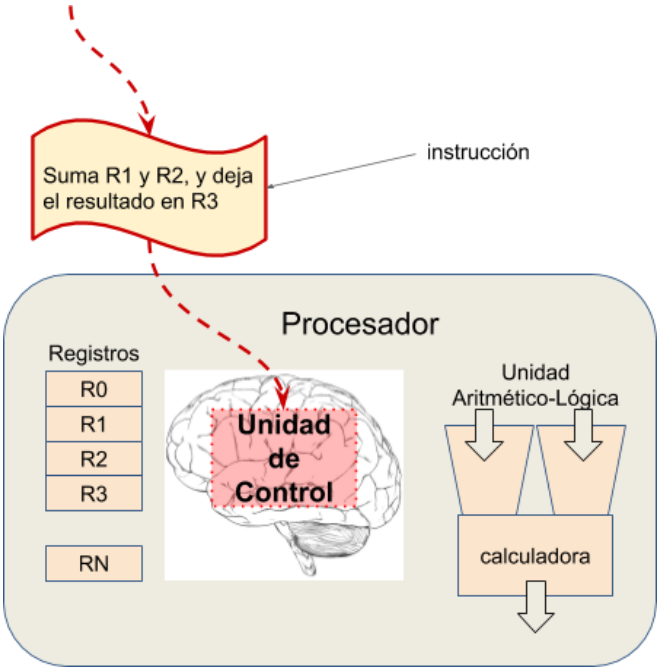
De la misma manera, la unidad de control también debe leer de algún lugar lo que hay que hacer y aplicar su poder de gobernar todo el computador para que se cumpla lo escrito. Decimos que "La Unidad lee, interpreta (entiende) y ejecuta **instrucciones**"

Las instrucciones

Una instrucción, en el ámbito de la informática que estamos tratando, es una indicación escrita que la Unidad de control puede leer, interpretar y que describe una acción muy simple a realizar, que debe ser obedecida y ejecutada. Por ejemplo, "*suma el contenido del registro R1, con el contenido de la celda de memoria 202 y guarda el resultado en el registro R5*". La UC cuando recibe una instrucción, la intenta entender y procede a "mover los hilos" necesarios para que se ejecute.

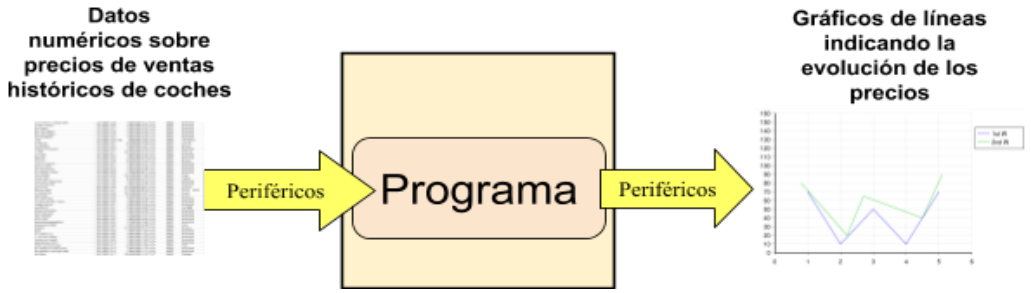
GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Author. Ignacio Pachés Giner



Las instrucciones describen algo muy simple, una operación a realizar muy concreta y que implica pocos movimientos de datos o cálculos. Con las recetas ocurre lo mismo: "echar sal a un huevo batido ". La idea es que muchos pasos de receta, bien ordenados consiguen un plato elaborado a partir de unos ingredientes simples. De la misma manera, muchas instrucciones, ejecutadas en el orden adecuado consiguen transformaciones de datos que aportan valor para el usuario.

Un bloque de instrucciones cuya ejecución es capaz de resolver un problema forma un programa



La pregunta ahora es ¿Dónde están las instrucciones? ¿Cómo llegan a la UC ordenadas? De momento, sólo es importante contestar a la primera pregunta. Las instrucciones están almacenadas en la propia memoria. Cada celda es una instrucción[1]. ¿Quién puso el programa en la memoria?... Ahora, no importa quién.

[1] Que una instrucción ocupe sólo una celda de memoria es mentira, pero vamos a creérnosla de momento.

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea Fondo Social Europeo El FSE invierte en tu futuro

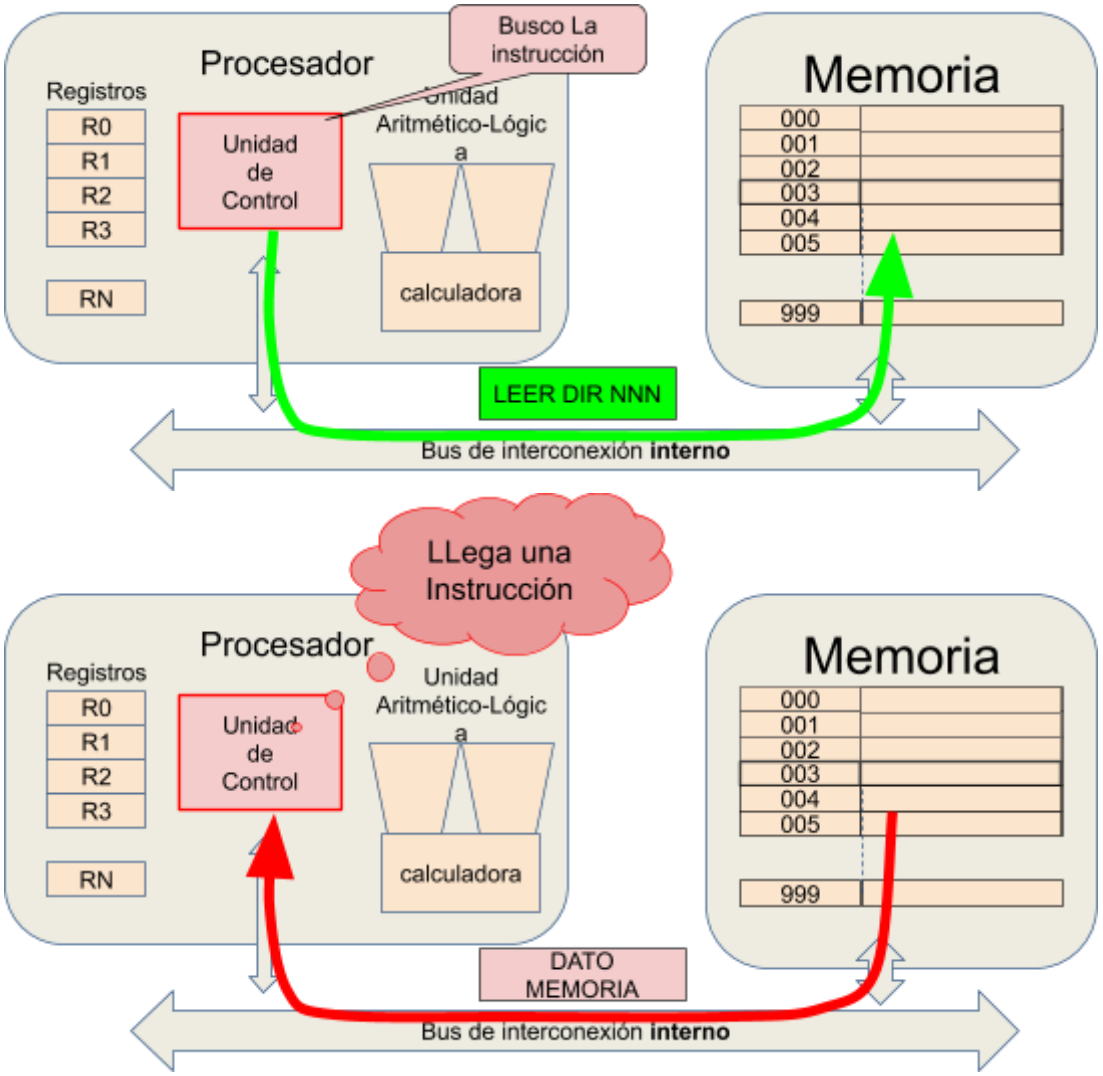
Author. Ignacio Pachés Giner

El ciclo de instrucción.

El procesador y concretamente la UC se dedican a ejecutar instrucciones. La manera en la que se ejecutan las instrucciones es siempre igual o casi igual en la mayoría de instrucciones. Se repite un ciclo de varias fases que consiguen ejecutar una instrucción. Explicaremos qué ocurre a cada fase:

Fase1 Búsqueda de la instrucción

La Unidad de control, sabe en todo momento cuál es (o dónde está) la siguiente instrucción a ejecutar. Por ello, lo primero que hace es leer de la memoria en la dirección sabida. Así, el contenido de la memoria llega a la Unidad de control. Recuerda que la memoria sólo obedece al procesador y no sabe dónde están las cosas



Fase 2. Decodificación de la instrucción

En la Unidad de control, se examina el dato recibido y se descifra su significado que es interpretado a sabiendas de que se trata de una instrucción. Los datos que llegan están en binario, pero cada combinación de datos binaria corresponde con una instrucción diferente.

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea Fondo Social Europeo El FSE invierte en tu futuro

Fase 3. Búsqueda de los operandos

Una vez que la unidad de control ha descubierto qué hay que hacer, se procede a buscar los datos que la instrucción indica que se han de tratar. Generalmente hay que hacer una o dos lecturas de la memoria o de los registros para "traerlos" hacia la UC o la ALU. Aunque las lecturas que se hacen de la memoria son igual que las de la fase 1 "búsqueda de la instrucción", los datos que llegan son tratados como operandos y no como instrucciones

Fase 4. Ejecución/Cálculo de la operación

La instrucción puede indicar la operación matemática a realizar con los datos obtenidos anteriormente, En esta fase se transportan los datos a la ALU y allí se realiza la operación matemática indicada por la instrucción.

Fase 5. Almacenamiento del resultado

Una vez obtenido un resultado del cálculo de la fase anterior, este resultado debe guardarse en el lugar indicado por la instrucción. Generalmente esto consiste en una escritura en la memoria o en un registro.

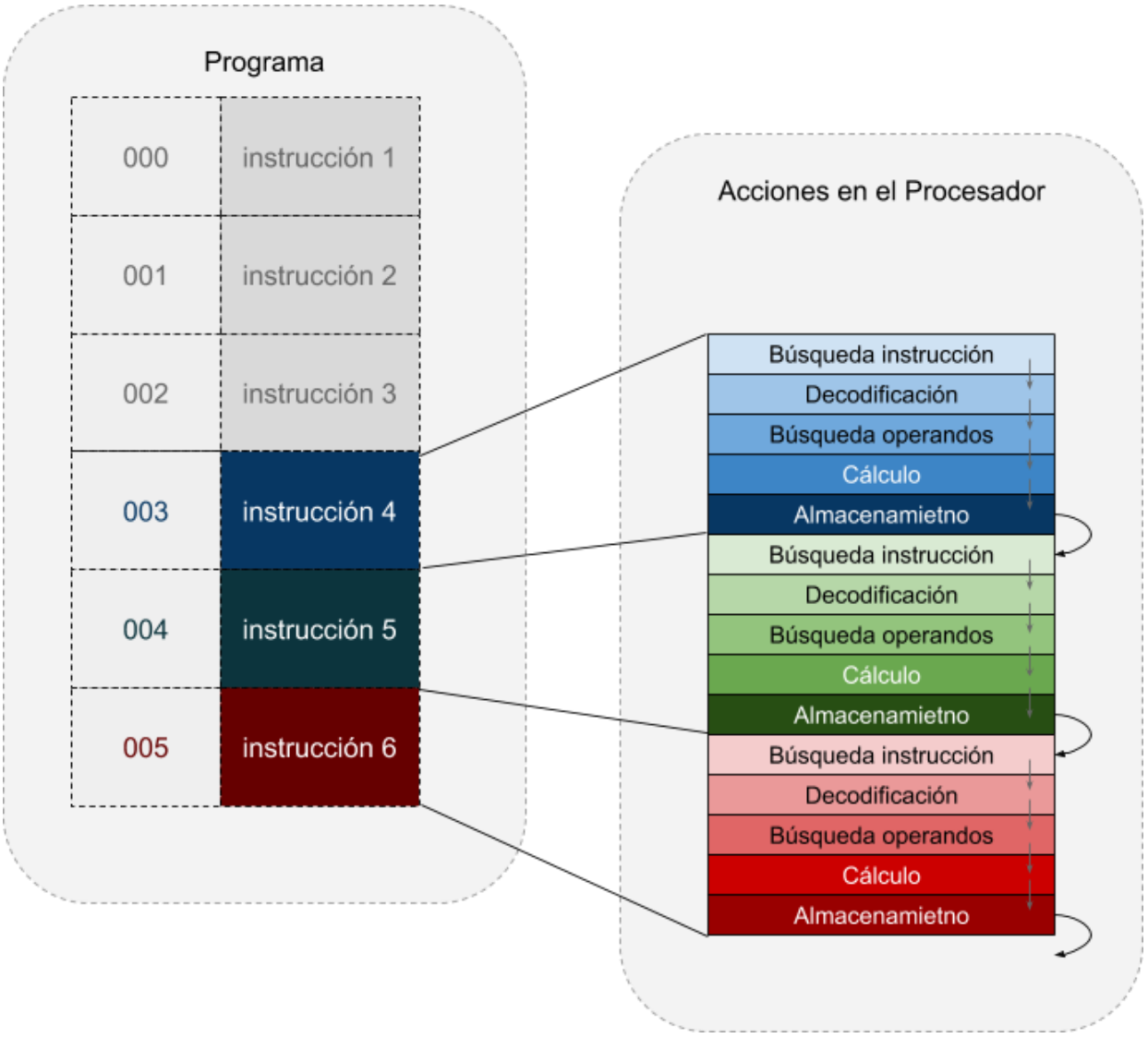
Algunos hechos sobre el ciclo de instrucción:

- No todas las instrucciones requieren todas las fases. Las únicas que son comunes a todas, son la primera y la segunda, todas las instrucciones realizan estas dos fases.
- Es posible que en la fase 3 "Búsqueda de operandos", se hagan hasta dos lecturas de memoria.
- La fase 4 "Cálculo de la operación" es la que realmente transforma los datos.
- Las instrucciones son muy simples y pese a ello tienen varias fases. Es imposible para un humano percibir cuándo se ejecuta una instrucción, y mucho menos percibir las diferentes fases.

Ejecución de un programa

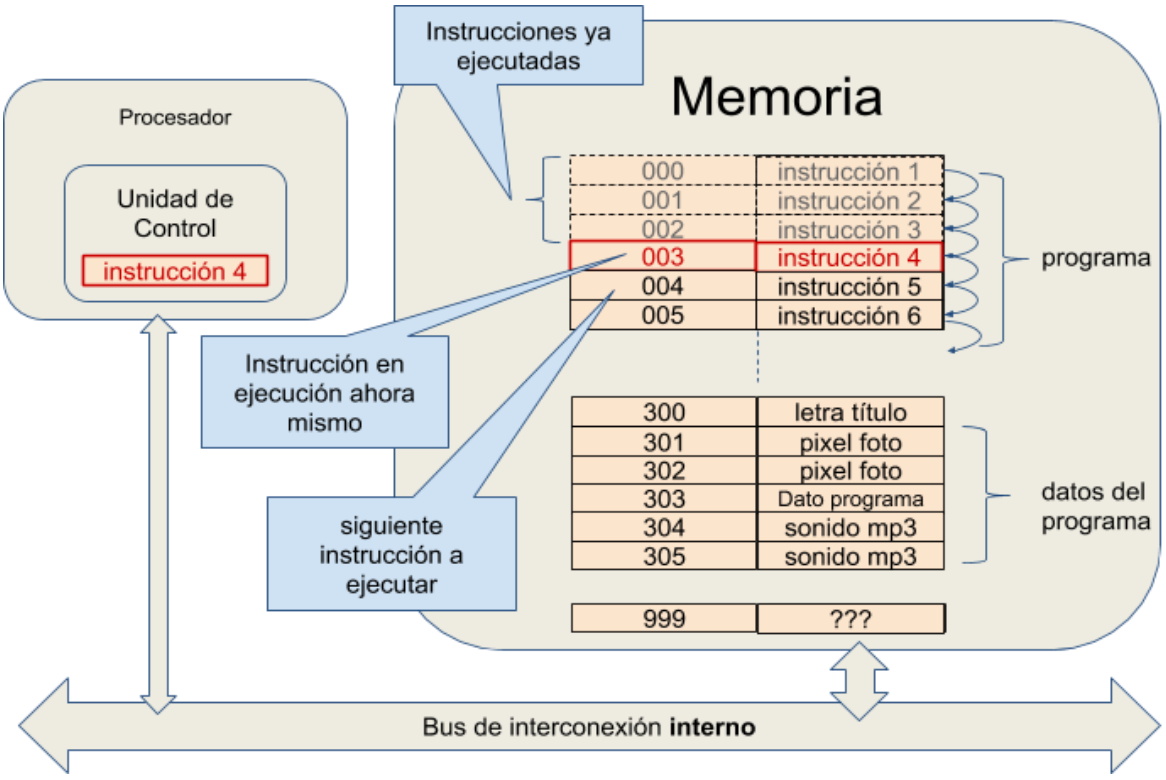
Ahora la pregunta es ¿Qué hace la UC cuando ha terminado de ejecutar una instrucción? Pues empieza de nuevo el ciclo, pero en la instrucción siguiente a la que acaba de terminar de ejecutar. Esto es, busca la instrucción que está en la celda a continuación de la que había leído en la fase 1 anteriormente.

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES Unión Europea Fondo Social Europeo El FSE invierte en tu futuro



Si aceleramos un poco en el tiempo la ejecución de las instrucciones, vemos que el programa va ejecutándose como una línea temporal que avanza a lo largo de la secuencia de instrucciones. Es igual que en una larga receta con muchos pasos. Si observas al camarero a cámara rápida, puedes ver cómo la receta "avanza". Podríamos decir que la receta está en preparación o realización. Aquí, podemos decir que el programa está en ejecución . Esta idea un tanto abstracta e intangible. Una posible representación del estado de un computador en un momento concreto que refleje esta idea es la siguiente:

Author. Ignacio Pachés Giner



El "avanzar" que observas conforme va ejecutándose una instrucción detrás de otra, se denomina técnicamente "hilo de ejecución".

Tipos de instrucciones

Quizá sea el momento de contestar otra pregunta: ¿Qué puede querer hacer una instrucción? ¿Qué tipos de instrucciones hay? ¿Qué puede indicar un paso de una receta?

Instrucciones de movimiento.

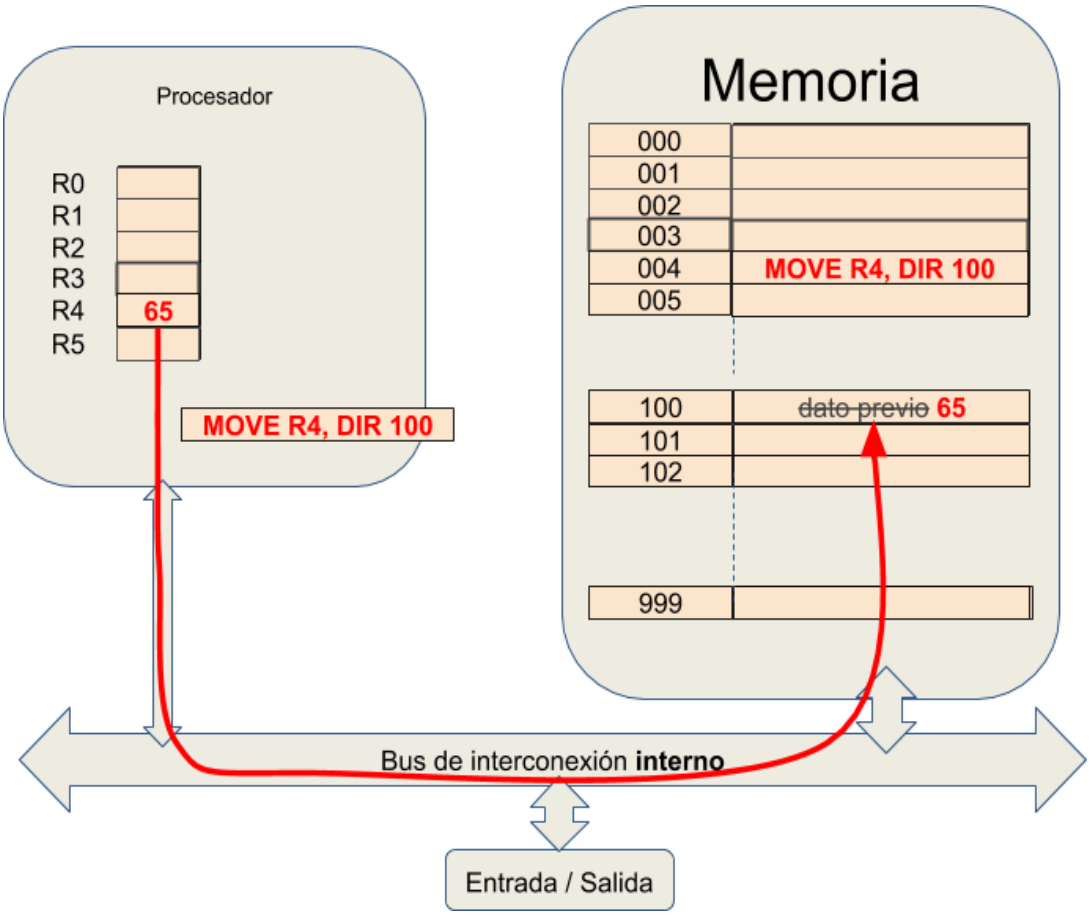
Las instrucciones de movimiento no realizan ningún cálculo ni operación matemática (no tienen fase 4). Tan sólo indican que algún dato debe copiarse de un lugar a otro lugar del computador. Aunque se trate de una copia, cuando se escribe un programa, se suele usar la palabra "MOVE" o "MOV" para escribir la instrucción. En la ella, se debe indicar dónde está el dato origen y dónde el lugar de destino, lo cual se indica después de la palabra MOVE/MOV. Las instrucciones de movimiento tienen la forma general:

```
MOV origen, destino
```

Por ejemplo, la instrucción

```
MOV R4, DIR 100
```

Copiará el dato que esté contenido en R4 a la celda de memoria cuya dirección es 100. Podemos poner un ejemplo más concreto, imaginando una situación como la que describe el siguiente diagrama, en el que el procesador está ejecutando esa instrucción cuando el contenido del registro R4 es el número 65. Debido a esa instrucción, ese número se copiará a la dirección 100 de la memoria y el dato previo se perderá. Observa que la instrucción ejecutada está en otra posición de la propia memoria, aunque ello realmente no importa para ver el efecto en sí de la propia instrucción



Los datos origen y los destinos pueden estar en la memoria, en algún registro y también en los periféricos. Por lo que podríamos hacer esta tabla:

Ejemplo	Ubicación del origen	Lugar de destino	Explicación
MOVE R1, R7	Registro R1 del procesador	Registro R7 del procesador	Se copia el contenido de R1 al registro R7
MOVE DIR100,R0	Dirección 100 de la memoria	Registro R0 del procesador	Se lee de la dirección 100 un dato y se escribe en el registro R0 de la memoria
MOVE DIR100, DIR20			
MOVE Teclado, R1	Teclado	Registro R1 del procesador	Se lee la tecla pulsada por el usuario y se guarda su código en el registro R1

Instrucciones aritméticas.

Estas instrucciones son las más completas (tienen todas las fases) y las más variadas. En ellas se indican:

- Uno o dos datos origen.
- Una operación
- Un lugar donde guardar el resultado.

La operación a realizar es lo que da nombre a la instrucción. Así, una instrucción que sume se llamará "ADD", y una que divida "DIV", etc. Después del nombre de instrucción vienen los operandos fuente y el destino, siguiendo la forma general

INSTRUCCIÓN Dato origen, dato origen, ubicación destino

Ejemplo:

ADD R1, DIR 209, R3

Explicación:

- Operación a realizar: viene indicado por la instrucción, se trata de una suma.
- Sumandos o datos origen: uno es el dato contenido en el registro R1 y el otro el contenido de la celda de memoria en la dirección 209,
- Lugar donde dejar el resultado: El registro R3 de la CPU.

Algunas operaciones aritméticas sólo actúan sobre un dato, y otras pueden tomar uno de los lugares origen como destino también.

Ejemplos:

ADD R1, R3

Suma el contenido de R1 a el contenido de R3, y deposita el resultado en R3. Se podría decir simplificadamente que esta instrucción "suma o añade R1 a R3".

ADD DIR 20, DIR 40

Suma el contenido de la celda de la dirección 20 al de la celda de la dirección 40 y se deposita el resultado en esta última celda en la dirección 40.

Podemos ver algunas combinaciones:

Ejemplo	Ubicación de los datos origen	Lugar de destino	Explicación
ADD R1, R7, DIR 20	Registro R1 y R7 del procesador	Dirección 20 de la memoria	Se suma R1 a R7 y el resultado se almacena en la celda de memoria con dirección 20
SUB R1, DIR20 , R3	Registro R1 y dirección 20 de Memoria	Registro R3 del procesador	Se resta *del registro R1* la *dirección 20* de la memoria y se guarda el resultado en R3
ADD DIR100, DIR20	Celdas en las posiciones 100 y 20	Celda en la posición 20	SE suman los contenidos de las celdas 100 y 20 de memoria y se guarda el resultado en la posición 20 de memoria
SUB R1, R1, R2	Los datos se toman del registro R1 tanto para el minuendo como para el sustraendo	Registro R2	A R1, se le resta él mismo. El resultado siempre será 0 y se almacenará en R2

SHL R1	R1	R1	Se desplazan los bits del registro R1 hacia la izquierda
SHR DIR 20, DIR 22	Dirección 20 de memoria	Dirección 22 de memoria	SE desplazan los bits de la dirección 20 de memoria y se deja el resultado en la dirección 22

Observa cómo se relaja el lenguaje. En vez de decir correctamente "Se resta, ***de la cantidad almacenada*** en el registro R1 ***la cantidad contenida*** en la celda 20 de memoria", se pasa a decir abreviadamente "se resta del registro R1, la celda 20" o "al registro R1 se le resta la celda 20". Y de ahí "se resta R1 menos la dirección 20 " A partir de aquí escribiremos simplificadamente la ubicación de los datos origen y destino.

Instrucciones de salto

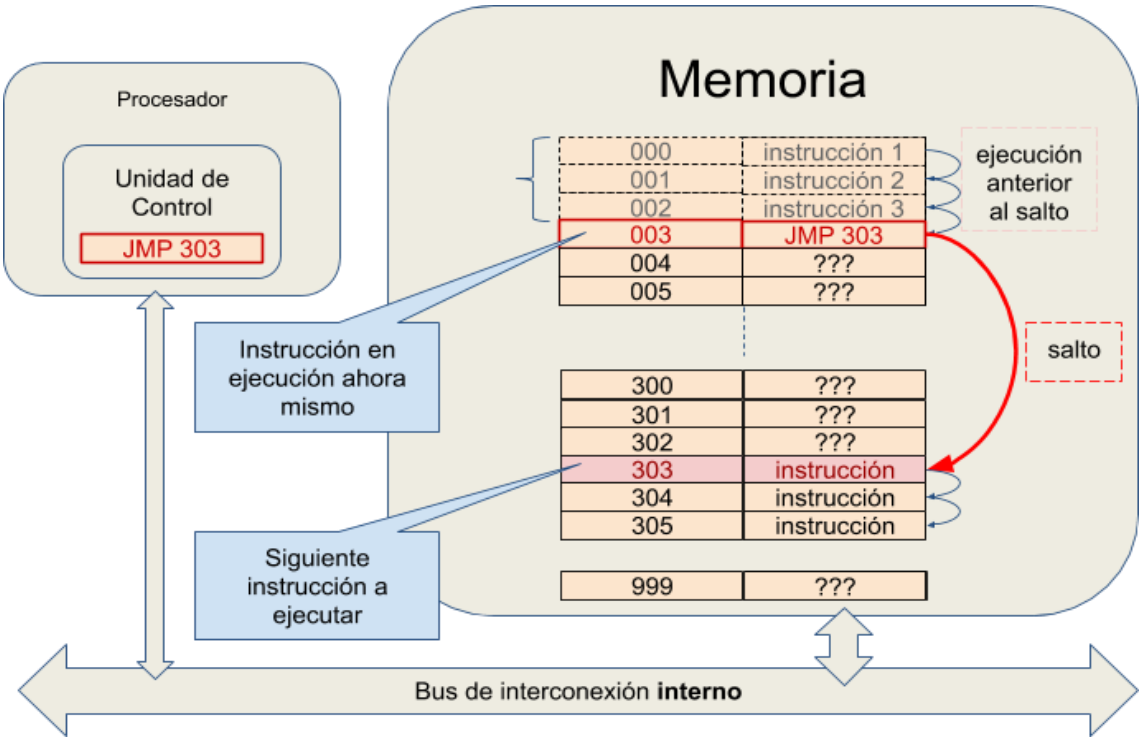
¿Recuerdas la idea asociada a la "Ejecución de un programa"? Cada instrucción se ejecuta después de la anterior, lo que provoca un avance a lo largo de una secuencia de instrucciones. Las instrucciones de salto están aquí para romper ese avance lineal y saltar a instrucciones ubicadas en otro punto de la memoria.

Para introducir las instrucciones de salto, vamos a tomar como ejemplo la instrucción de salto simple, llamada JMP (de JuMP en inglés) que tiene la forma

JMP DIR1002

Donde 1002 es una dirección de memoria de ejemplo. En esa instrucción la Unidad de Control asume que la siguiente instrucción a ejecutar está en la dirección 1002, y por tanto la que había en memoria a continuación de esta de salto que se está ejecutando ahora, ya no será la siguiente. Por eso decimos que es un "salto". Después del salto la ejecución continúa de manera normal en las posiciones siguientes a la que se ha saltado. Es decir, no se trata de un salto puntual, sino "definitivo" (o hasta que se encuentre otra instrucción de salto) y se continúa la ejecución en otro bloque de memoria distante.

Author. Ignacio Pachés Giner



Instrucciones de salto hay de **tres tipos**

1. Instrucción de salto simple como la que se ha puesto de ejemplo. También se le llama instrucción de salto incondicional y descubrirás ahora mismito por qué.

2. Instrucción de salto condicional, en la que el salto se realiza si y sólo **si se cumple alguna condición**. En caso de no cumplirse la condición, la instrucción no tiene ningún efecto y se continúa la ejecución normal en la instrucción en la siguiente posición de memoria.

3. Instrucción de salto con retorno. Es una instrucción de salto incondicional pero que guarda la dirección de memoria siguiente desde donde se realiza el salto. La UC recuerda el punto desde el que se ha hecho el salto porque más tarde habrá una instrucción de retorno que provocará un salto a dicha posición. Ahora, quizá te cueste entenderlo, más abajo hay un ejemplo

Instrucciones de salto condicional.

Las instrucciones de salto condicional que trataremos realizan un salto igual que las de salto incondicional pero sólo si se cumple una condición determinada. Es decir, no sabemos de antemano si se producirá el salto o no estipulado en la instrucción, todo depende de los datos que existan en el procesador o el estado en el que esté en el preciso momento en el que se ejecute la instrucción.

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea Fondo Social Europeo El FSE invierte en tu futuro

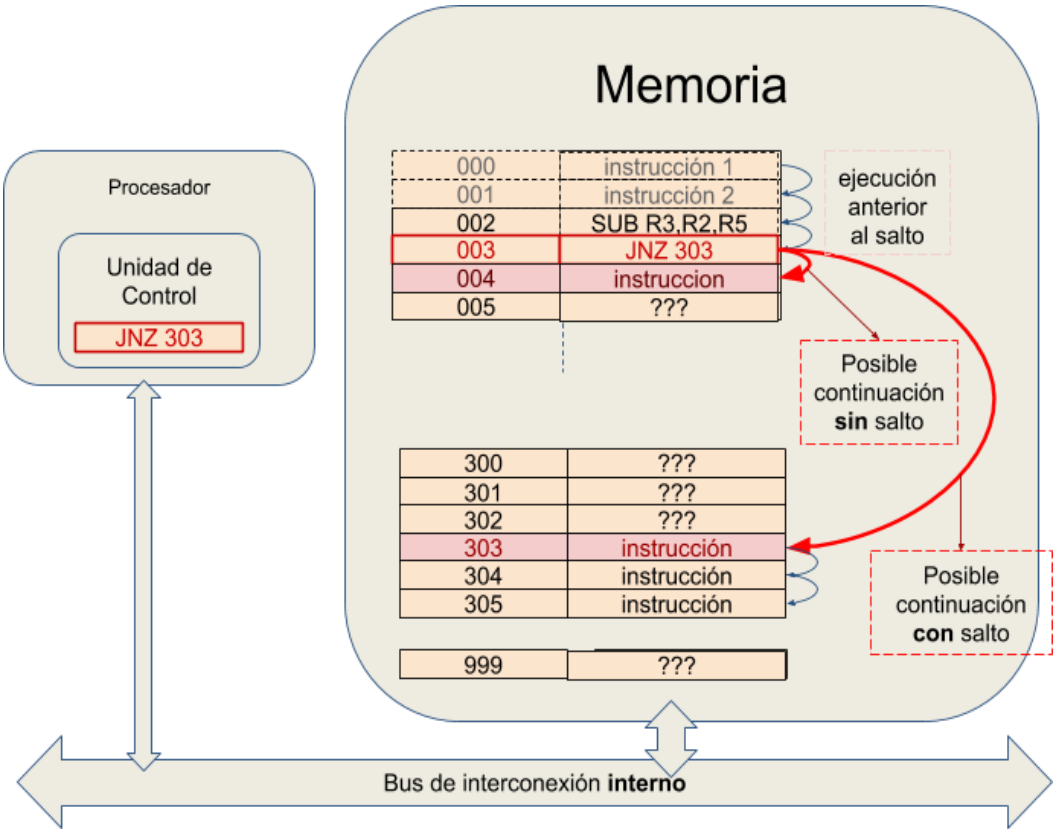
En nuestro estudio vamos a simplificar mucho estas instrucciones de salto y vamos a considerar que el salto depende de si el resultado de la última operación aritmético-lógica realizada en la ALU ha sido cero o distinto de cero.

Las instrucciones que usaremos reciben el nombre y forma de:

`JZE NNNN`	Se salta a la dirección NNNN si el resultado de la última operación aritmético-lógica ha sido 0
`JNZ NNNN`	Se salta a la dirección NNNN de memoria si el resultado de la última operación ha sido **distinto** de 0

En algún momento se realiza una operación como una suma o una multiplicación. Más tarde, una instrucción de salto condicional pregunta a la ALU si la última operación matemática que realizó dió cero, y se toma una decisión dependiendo de la respuesta y de la instrucción concreta de salto condicional usada.

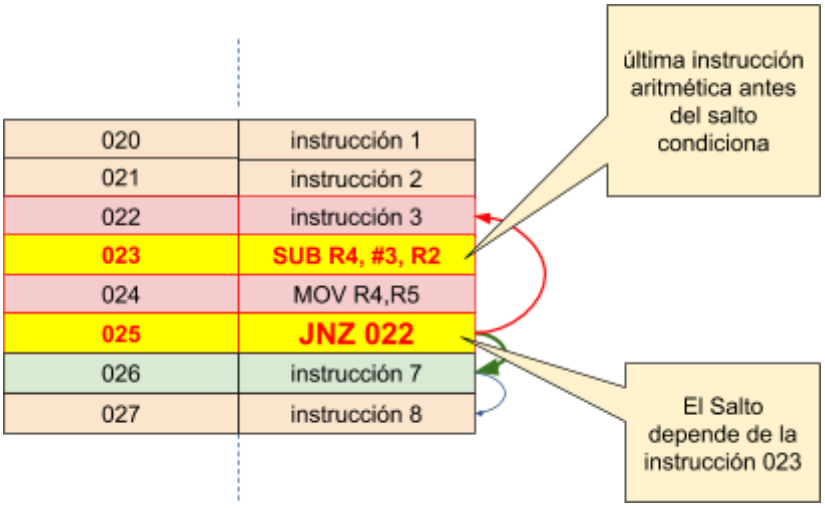
Veamos un ejemplo muy concreto:



En el ejemplo anterior, la instrucción en la dirección 002 instruye al procesador a realizar una resta. Esta resta se efectúa en la ALU y se registra en algún lugar si el resultado ha sido 0 o diferente de 0 (se registran más cosas realmente). Esa anotación sobre el resultado obtenido, perdura hasta la próxima operación aritmético-lógica. La siguiente instrucción en la dirección 003 es la instrucción de salto condicional a la

dirección 303. LA instrucción del ejemplo es "JNZ" que equivale a "salta si no es cero (Jump if Not Zero). Por lo que el salto se realizará si el resultado de la resta no ha sido 0. Todo lo que podemos decir ahora al ver el ejemplo es que el salto dependerá de los valores de R2 y R3 que es de dónde se toman los datos para la resta y determinan el resultado.

Las instrucciones de salto condicional no actúan solas, porque la condición para saltar, la establece alguna instrucción anterior, y concretamente la última instrucción anterior que provocó una operación aritmética.



Las instrucciones de salto condicional son claves para la construcción de programas que actúen de forma diferente según los datos. En el ejemplo del cocinero, puedes considerar que algún paso de receta pueda indicar algo así como "freír las albóndigas hasta que se empiece a dorar el rebozado exterior". En ese paso de receta se deja al cocinero la responsabilidad de decidir el tiempo de fritura en función de alguna inspección de los alimentos durante ese proceso. En algún momento, el cocinero mirará las albóndigas y observará el color, en función del color las sacará y continuará la receta por algún paso, o tomará la decisión de esperar un poco más (o hacer otra cosa mientras). Igualmente imagina que la receta indica "Corta unas manzanas a dados para mezclar con el queso. En caso de no tener manzanas, usa plátano" En este paso de receta, el ingrediente final se decide en el momento de su elaboración. El cocinero (que representa la UC) debe tomar una decisión en función de cómo esté la despensa del restaurante

De la misma manera, los programas se comportan de forma diferente según los datos recibidos. Un usuario eligiendo una opción dentro de un menú, está eligiendo el camino que debe tomar un programa para satisfacer su decisión. Por ello, el programa está preparado para examinar las decisiones del usuario y actuar según elija cada vez. Las instrucciones de salto condicional son la base para este comportamiento. En esta explicación que acaba de darse, se intenta relacionar un evento macroscópico con una técnica más microscópica, Pero no se intenta dar explicación a todas las incógnitas. Recuerda: no intentes entender más allá de lo que aquí se dice y quieras comprender ya todo el funcionamiento del computador, ve cogiendo ideas sueltas.

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Author. Ignacio Pachés Giner

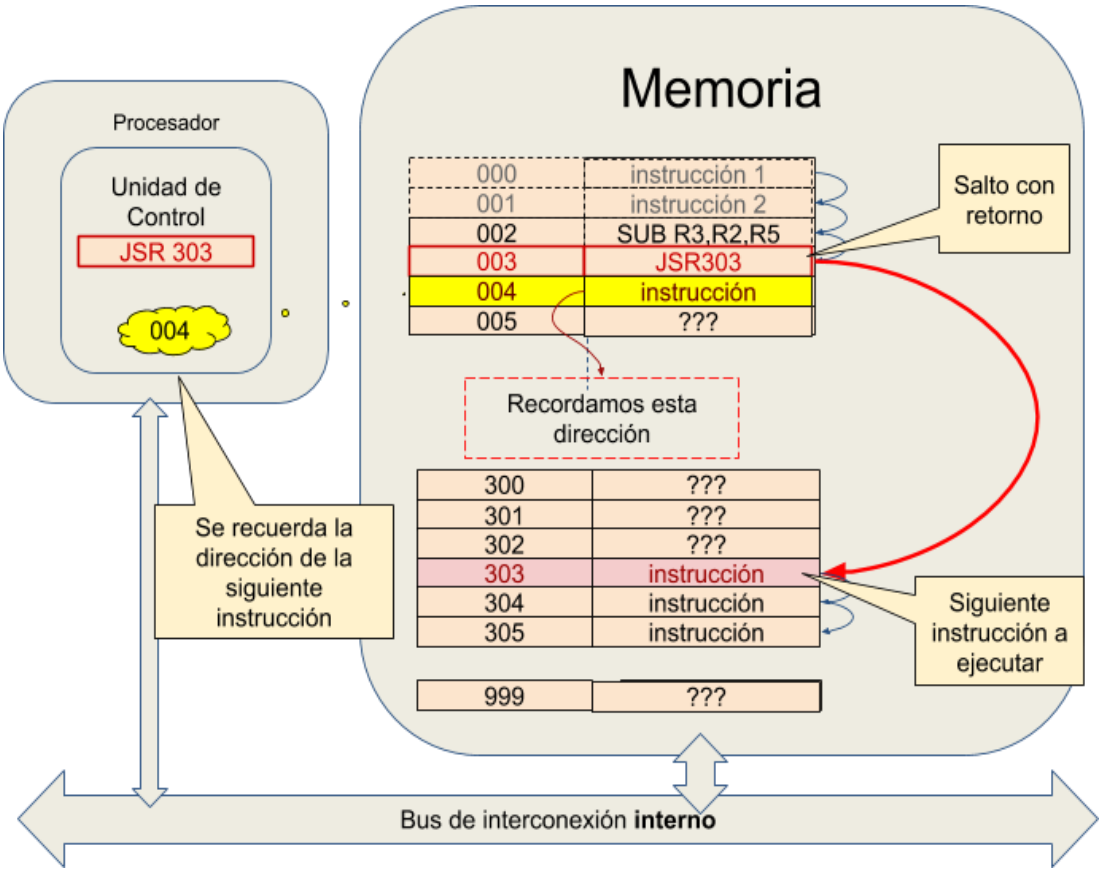
Instrucción de salto con retorno.

La instrucción de salto con retorno comprende realmente dos instrucciones:

- "JSR" (Jump to SubRoutine)
- "RET" (RETurn from subroutine).

Estas dos instrucciones están pensadas para ser usadas conjuntamente.

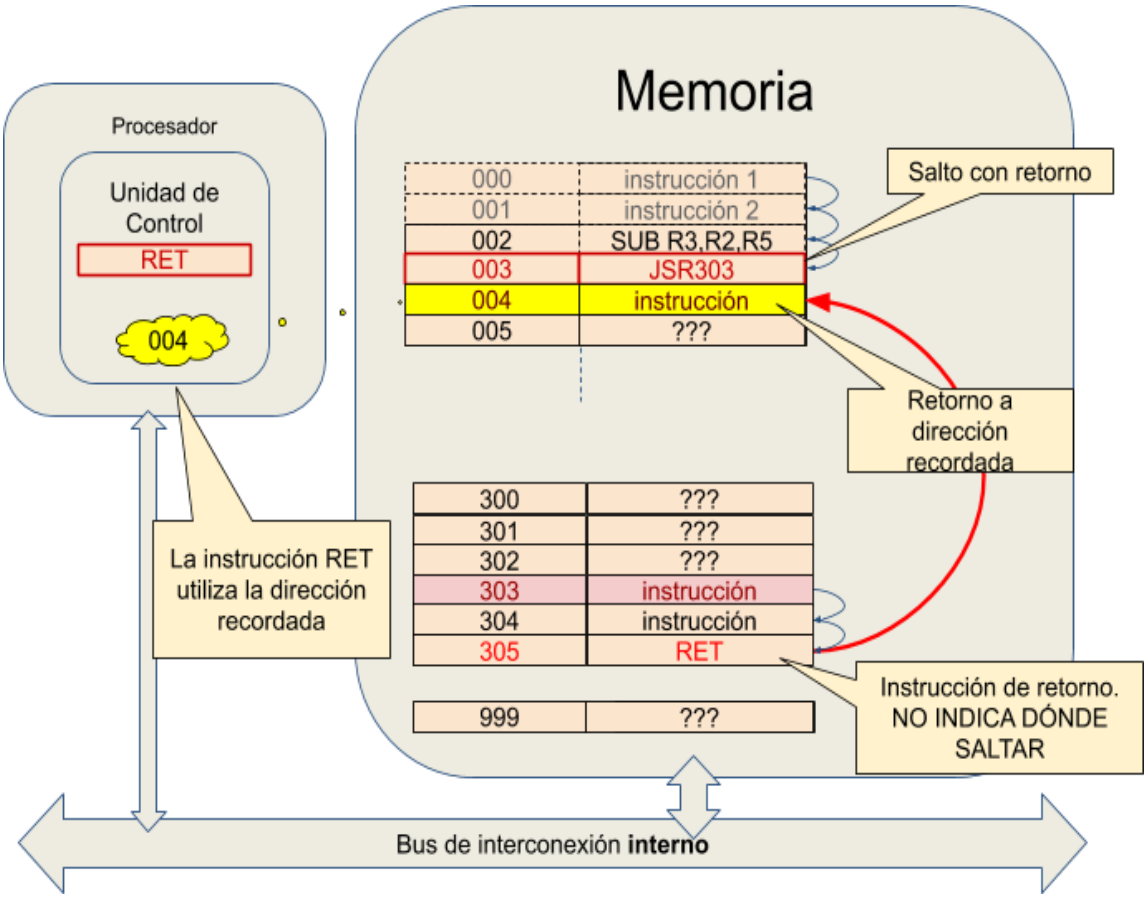
La instrucción JSR DIR**NNNN** realiza un salto incondicional - en principio normal e incondicional- a la dirección **NNNN** . La novedad, es que la UC recuerda la dirección siguiente a la JSR que provoca este salto. Es decir recuerda la dirección de la memoria donde está la instrucción que vendría a continuación si no se hubiese hecho ningún salto.



Después del salto se ejecuta la instrucción en la posición de memoria indicada por el salto y a continuación, se suceden las siguientes instrucciones de forma habitual.

En algún momento más tarde, se ejecutará la instrucción "RET", que también es una instrucción de salto, pero en la que no se indica ninguna dirección. RET es una instrucción que no lleva nada escrito junto a ella. Y ¿A dónde salta RET ? RET salta a la dirección que en la instrucción JSR se almacenó en la UC. Es

decir, regresamos a la instrucción siguiente a aquella desde la que se hizo el salto con JSR . Es como si *siguiéramos donde lo dejamos* .



Supón que el cocinero en muchos platos prepara una salsa especial. La receta de dicha salsa está escrita en una página concreta. En muchas de las recetas de los platos principales hay un paso que es "*consulte la página XX para realizar la salsa necesaria en este plato*". En la página de la receta de la salsa, al final de todos sus pasos hay una indicación "*Ahora ya tiene la salsa preparada, regrese a la página del plato que estaba preparando y continúe tal cual indique allí la receta en el paso siguiente al que se lo dejó*".

Es la misma idea, pero en programación e implica a estas dos instrucciones: "JSR" y "RET"

El bloque de instrucciones que se ejecuta como consecuencia de la instrucción de salto y comprende desde la dirección indicada por la instrucción JSR hasta la instrucción RET se denomina SubRutina, porque es como un trozo de programa algo aislado o separado del resto y es ejecutado mediante esta técnica.

Este par de instrucciones permiten:

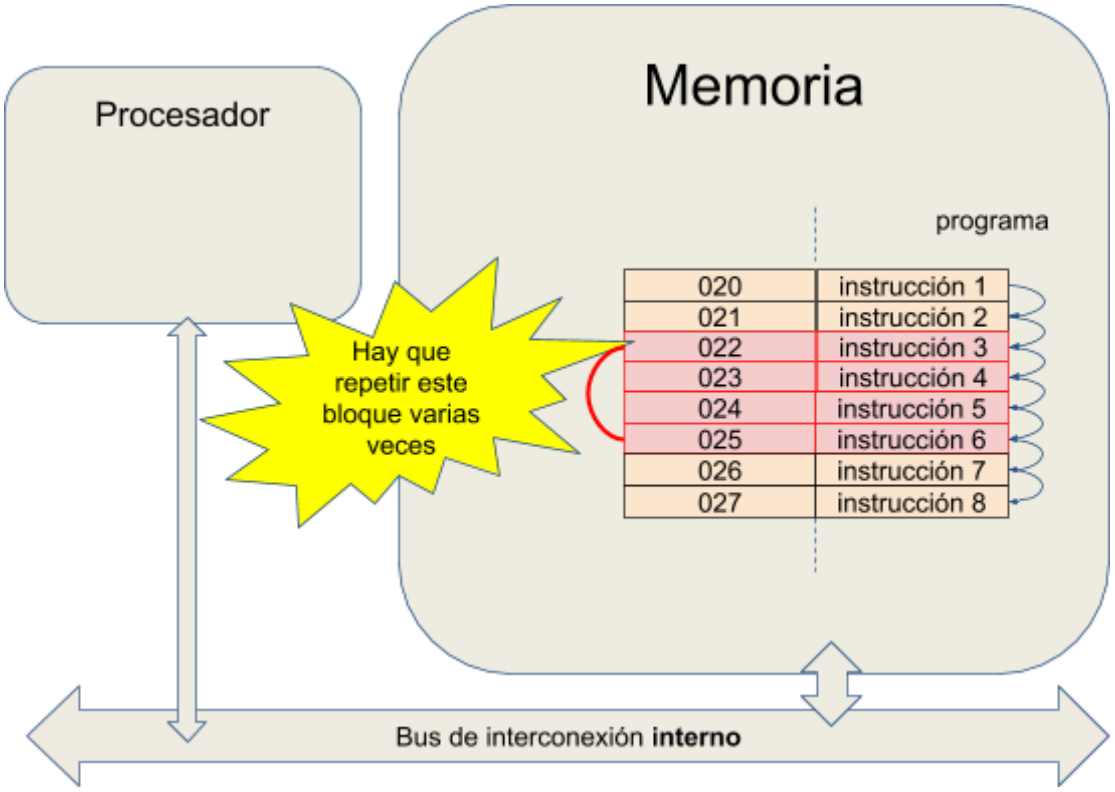
- Escribir un trozo de programa que es ejecutado en diferentes ocasiones y desde diferentes lugares.
- Permitir que un programador escriba un trozo de programa despreocupándose del resto del programa (es como si un cocinero escribe la receta de la salsa sin importarle en qué platos va a ser gastada)

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea Fondo Social Europeo El FSE invierte en tu futuro

NOTA: Realmente la dirección de retorno no se almacena directamente en el procesador. La dirección de retorno está en la memoria, pero la posición de la memoria donde está sí que está referenciada desde un registro.

Bucles

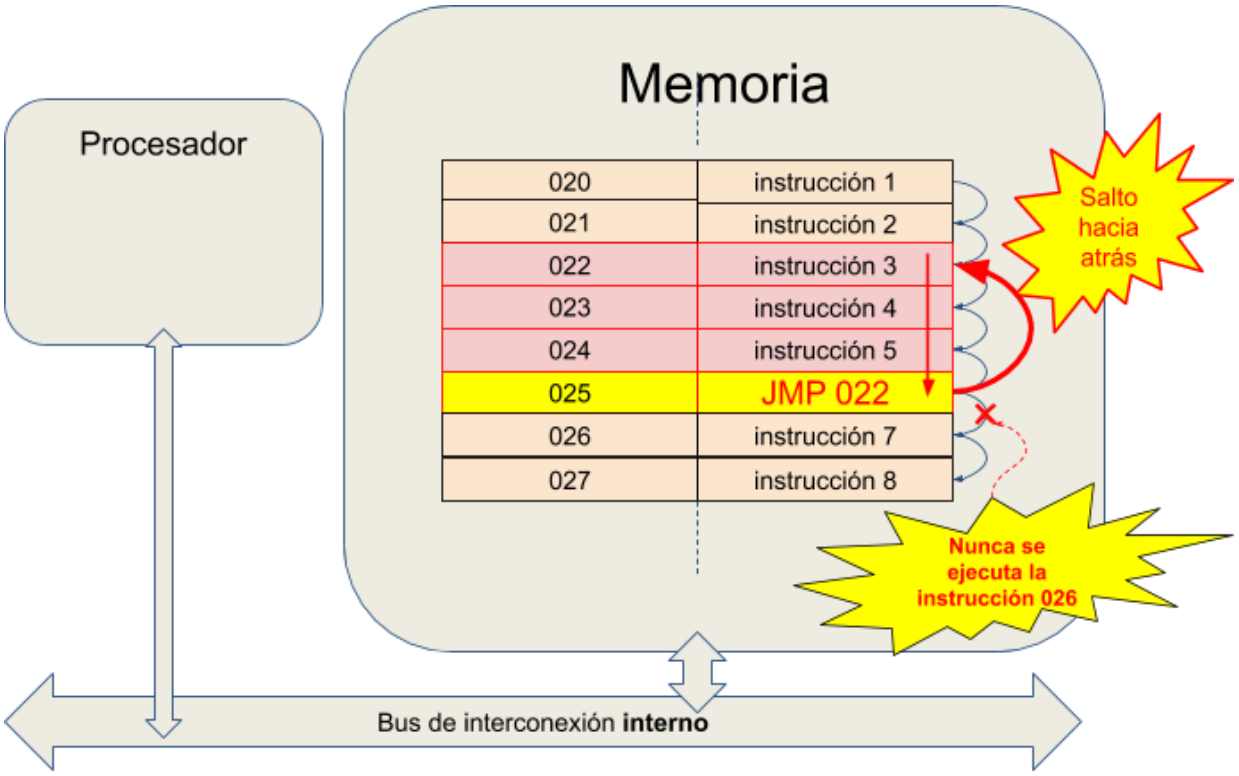
El "bucle" informático, se definiría como la repetición -varias veces- de la ejecución de un grupo o bloque de instrucciones consecutivas. En otras palabras, se trata de repetir un bloque de instrucciones, ejecutarlas todas una detrás de otras, pero varias veces, el bloque entero. No se trata de repetir varias veces UNA instrucción y después varias veces la siguiente, y así sucesivamente; sino de ejecutar una detrás de otra un bloque entero, y después de terminar con el bucle, volver al principio.



En el ejemplo anterior, la secuencia de instrucciones insrucción3 a instrucción6 se ejecutará varias veces antes de que se ejecute la instrucción 7 y siga todo con normalidad.

Conseguir que ocurra esto es tarea de las instrucciones de salto. Y la manera más inmediata (aunque no del todo correcta ahora mismo), es mediante un salto incondicional hacia atrás.

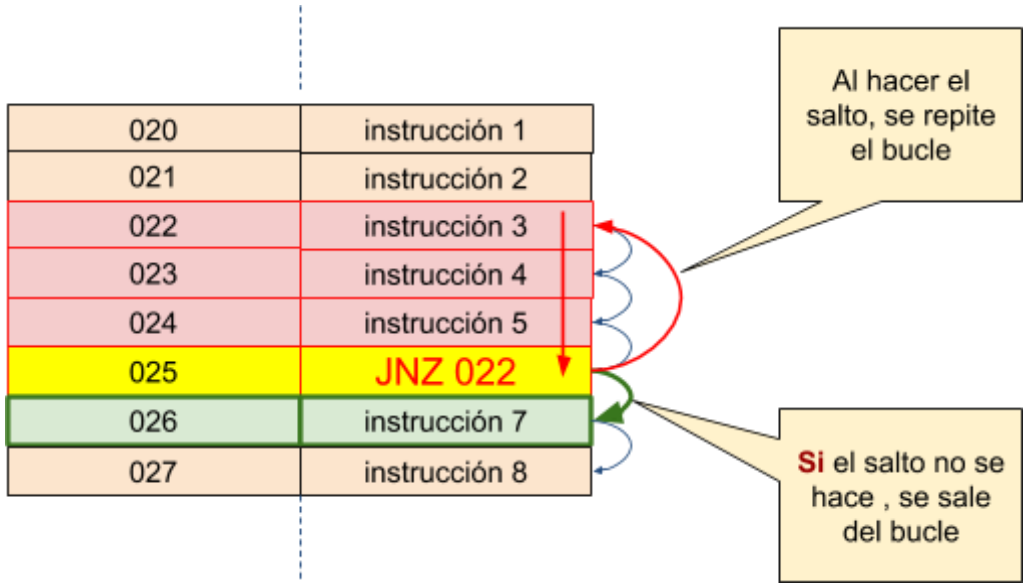
Author. Ignacio Pachés Giner



El bucle anterior del ejemplo, es un "bucle infinito" porque nunca termina, siempre se vuelve a ejecutar. La idea de un bucle en informática es que se repita el número necesario de veces y no infinitas veces. Date cuenta que la repetición ocurre como consecuencia del salto. Por ello, para no repetir eternamente, es posible usar una instrucción de salto condicional, ya que así, no siempre se salta y por tanto nos siempre se repite. La habilidad del programador es elegir la instrucción de salto adecuada al bucle que desea repetir.

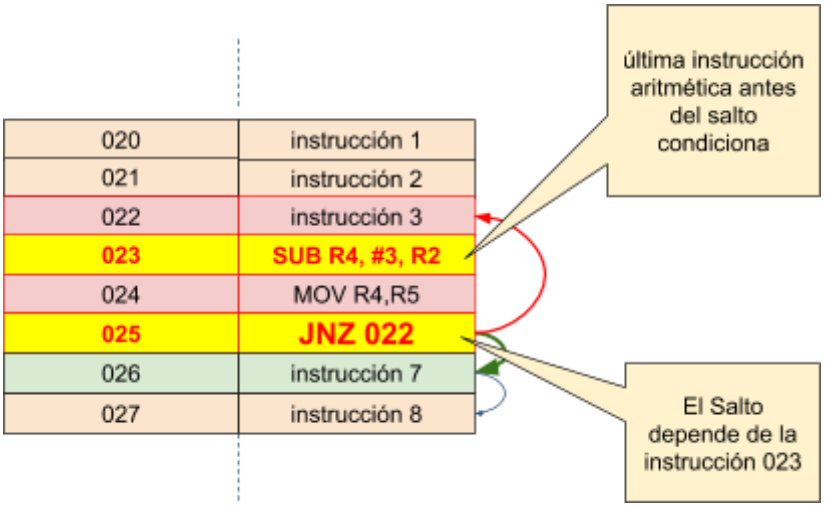
Repetir un bucle un número de veces.

El bucle infinito rara vez tiene sentido. Lo habitual es repetir un bloque unas cuantas veces y después seguir; Esto es, continuar con la siguiente instrucción después del salto hacia atrás en algún momento. Con una instrucción de salto incondicional como el ejemplo anterior (o siguiente) esto es imposible. Pero lo importante ahora es entender ese concepto de "salir del bucle" o "dejar de repetir el bucle"



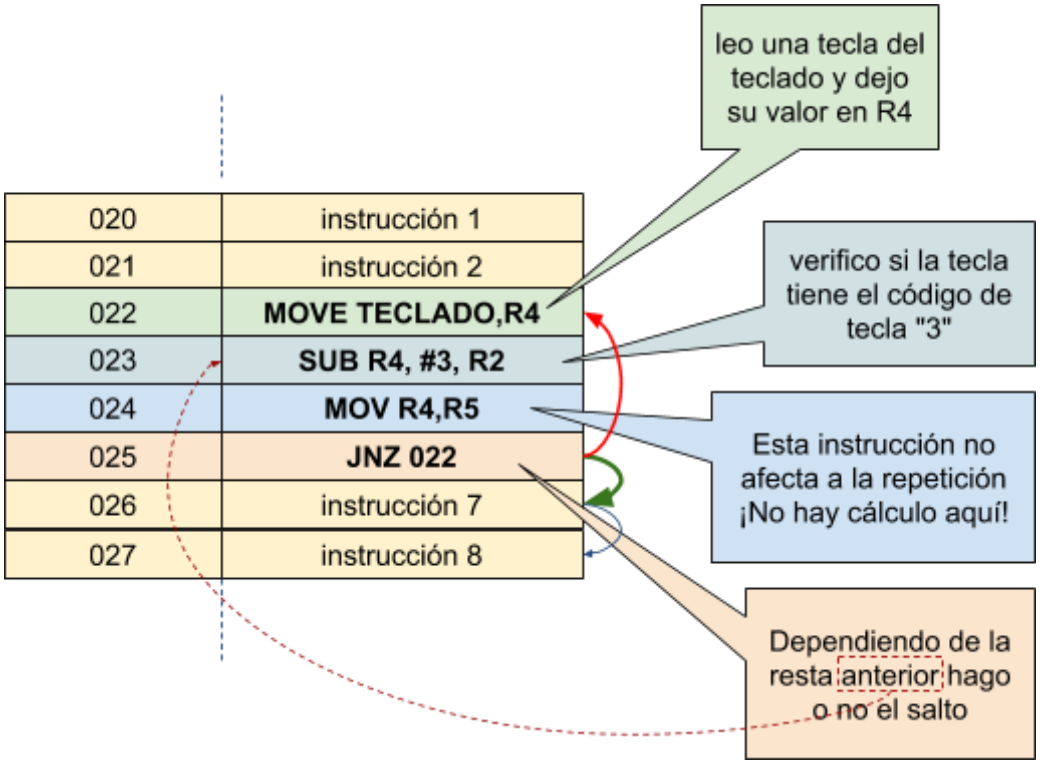
Para conseguir que alguna vez no se realice el salto y por tanto se cumpla esa idea de "salir del bucle" hemos de utilizar una instrucción de salto que a veces "falle"…. ¡Claro! Para eso están las instrucciones de salto condicional. Debemos razonar sobre el ejemplo anterior y sacar alguna conclusión:

- La instrucción de salto condicional realiza el salto dependiendo de alguna operación anterior. Es de suponer que en las instrucciones del bucle (instrucción 3,4,5) realizarán alguna operación aritmético-lógica. De no ser así… ¡El resultado de la condición será siempre el mismo ! y la instrucción de salto condicional siempre tendrá el mismo efecto
- Si en el bucle, hay varias instrucciones que realizan operaciones en la ALU, sólo la última es la que afecta a la condición que evalúa el salto condicional.
- Por tanto, la repetición del bucle entero está controlado por un par de instrucciones: la última que realiza una operación aritmético-lógica y la de salto condicional



- Cuando se ejecute la instrucción 7, se habrá salido del bucle y -en este ejemplo- ya no se volverá a saltar hacia atrás y repetir nada.
- En el ejemplo anterior, se puede adivinar que cuando R4 valga 3, se dejará de repetir el bucle.
- En el bucle, los datos con los que se comprueba la condición deben estar expuestos a cambios en algún momento. En el ejemplo anterior, podríamos decir que R4 es el registro que determina si se continúa repitiendo o no. Se espera que cambie de valor, de lo contrario, siempre se obtendrá el mismo resultado de la comprobación de la condición y estaremos como al principio: con un bucle infinito.

  R4 debe ser alterado en el propio bucle de alguna manera. Observa el siguiente ejemplo que después se explica:



El bucle anterior lee del teclado una tecla en la instrucción de la dirección 22. Cada tecla tiene un código asociado, que poco tiene que ver con el carácter que lleva impreso. Ese código se deposita en el registro R4 en la instrucción 022. En la siguiente instrucción se realiza una resta entre dicho registro y el número 3 (direccionado con modo inmediato). Una resta da de resultado 0 si los dos datos son iguales (minuendo = sustraendo). Aquí el resultado 0 o distinto de 0, es lo que determina repetir o no repetir una vez más el bucle. Veamos las dos posibilidades:

- Supón que R4 tiene un valor distinto de 3. Entonces la resta no da 0. Y ese resultado llega a ser el que influye en la instrucción de salto condicional que SÍ que efectúa el salto hacia atrás. Y por ello, se repite el bucle.
- Si el dato leído del teclado es 3, entonces la resta da de resultado 0 y ese resultado influye para que la instrucción en la dir 025 "JNZ 022" NO efectúe el salto y por tanto se salga del bucle.

Modos de direccionamiento

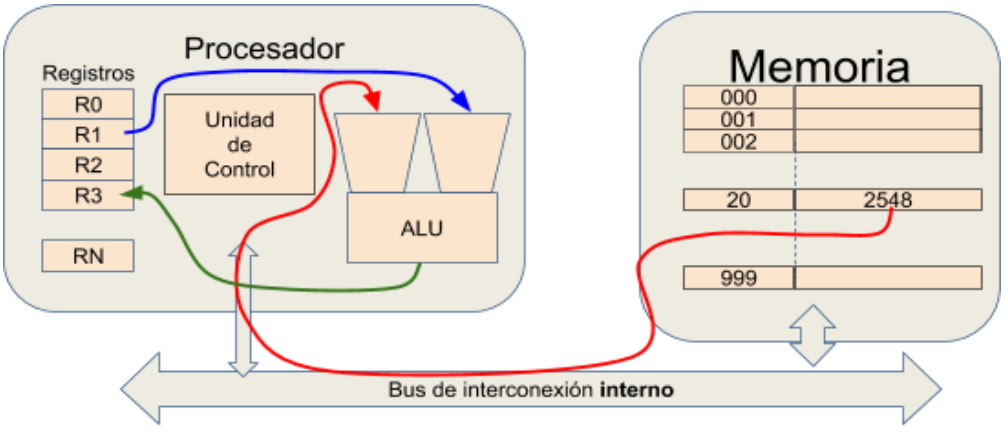
Los modos de direccionamiento son las maneras de especificar dónde están los datos que una instrucción referencia.

Modo directo a registro y memoria

Este modo ya lo hemos usado y lo conocemos por los ejemplos anteriores. Supongamos la instrucción vista antes:

SUB R1, DIR20 , R3

Esta instrucción lee del registro R1 un dato, y de la dirección 20 de la memoria otro dato. Esos dos datos viajan a la ALU, donde se realiza una resta y se obtiene un resultado. El resultado se almacenará en el registro R3.



SUB R1, DIR20 , R3

Lo que nos importa ahora es ver dos formas de especificar las ubicaciones de los datos:

- Uno de los datos necesarios para la resta está en R1, tal como dice la instrucción. Este es un modo de direccionamiento que indica que el dato está en el registro . Llamaremos a este modo "directo a registro".
- Se usa el mismo modo para indicar dónde almacenar el resultado. El modo "directo a registro" sirve, por tanto, para indicar también el lugar a guardar un resultado.
- El segundo dato necesario para realizar la resta está en la memoria. Tal como dice la instrucción, en la dirección 20 de memoria. Este es un modo de direccionamiento distinto al anterior. Aquí

indicamos una dirección de memoria de la que tomar el dato. Estamos usando el modo "directo a memoria".

Redefinamos los modos que hemos usados:

- "Modo directo a registro": La instrucción indica el número de registro de dónde tomar el dato
- "Modo directo a memoria" : La instrucción indica la dirección de memoria donde está el dato.

Pero hay más modos que explicaremos a continuación

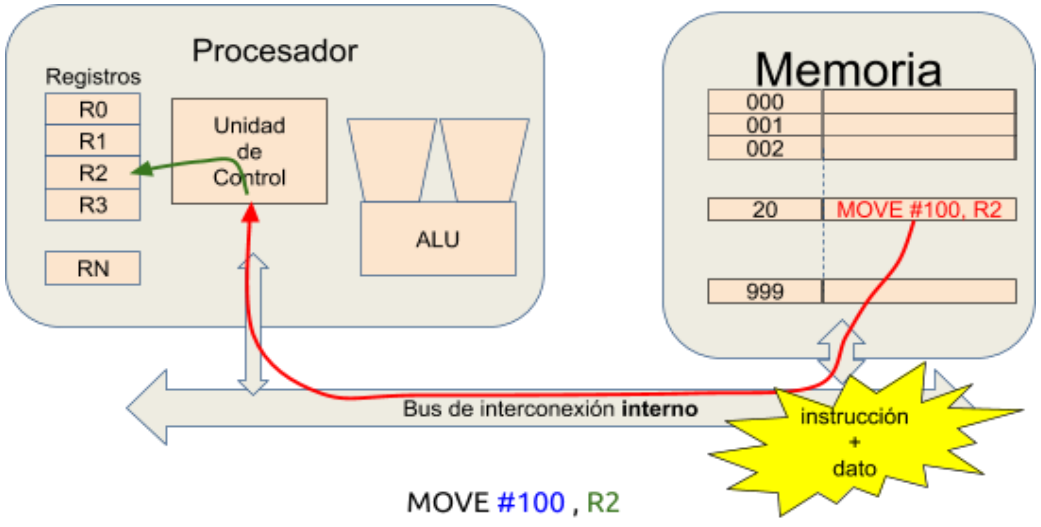
- "Modo inmediato": La instrucción contiene el dato, no hay que ir a buscarlo.
- "Modo indirecto a registro": Un registro contiene la dirección de memoria donde buscar el dato
- "Modo relativo": LA instrucción contiene un desplazamiento en memoria relativo a la instrucción actual … ¿Lo cuálo?

Veámoslos uno por uno.

Modo inmediato

El modo inmediato es un modo un tanto excepcional. En él, la instrucción ya proporciona el propio dato. Por tanto, este modo sólo puede usarse con datos de origen (el resultado de alguna operación ha de guardarse en algún lado y no se puede almacenar en la instrucción). Veamos un ejemplo para entender cómo usarlo:

MOVE #100, R2



El modo inmediato viene determinado por la almohadilla "#". En esta instrucción se indica que el dato a mover es directamente 100. No hay que ir a buscarlo a una celda de memoria o a un registro, es el número 100. En el ejemplo, el destino de ese dato es el registro R2. Con esta instrucción, sí sabemos totalmente

qué valor tendrá R2 después de ser ejecutada. Veamos otro ejemplo bastante habitual a la hora de hacer programas:

ADD R1,#1, R1

Esta instrucción suma el contenido de R1 al número 1, y deja el resultado en el propio registro R1. Lo que está haciendo simplemente es incrementar el registro en 1.

Date cuenta de que el modo inmediato no puede ser usado para especificar el destino.

ADD R1, DIR23, #25

Así mismo, el modo de direccionamiento inmediato es el que usan las instrucciones de salto, pero no hemos escrito el símbolo "#" para no confundir y seguiremos sin usarlo. Todas las siguientes instrucciones son la misma por tratarse de instrucciones de salto:

JMP #323

JMP 323

JMP DIR 323

El modo inmediato en la realidad está muy limitado, pues en los computadores, apenas se reserva espacio en bits para especificar datos dentro de una instrucción

Modo indirecto a registro

Este modo es un poco confuso al principio, porque especifica que el dato está en memoria, pero la instrucción no indica directamente en qué posición de la memoria (Retén esto en tu mente para forzar a entenderlo). La instrucción no sabe en qué celda de la memoria está el dato, pero es en la memoria igualmente.

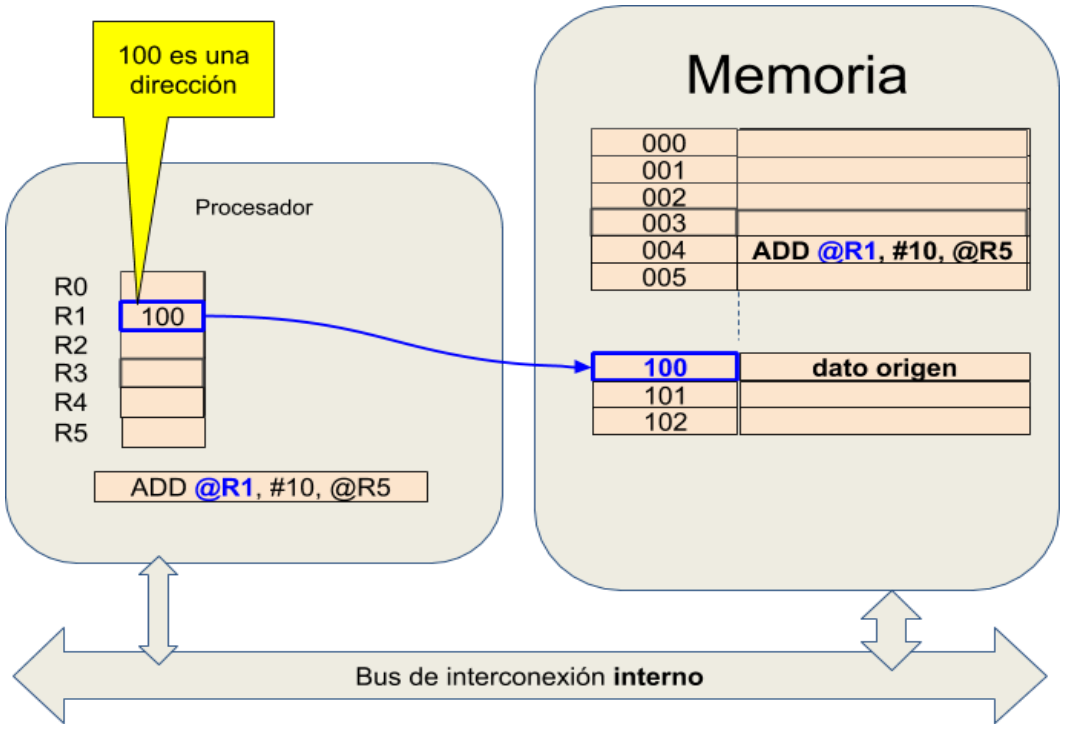
Por otra parte, el modo indirecto a registro, se llama así, porque usa un registro. En la instrucción se indica un registro. Y delante del registro se pone el caracter "arroba" o "@". Por ejemplo, en la siguiente instrucción se especifica un modo inmediato y dos modos indirectos a registro (gracias al símbolo de arroba "@") en los que aparecen el registro R1 y R5:

ADD @R1, #10, @R5

pero -tal como se ha dicho- el dato real a tomar no está en el registro está en la memoria. ¿Qué pinta el registro?

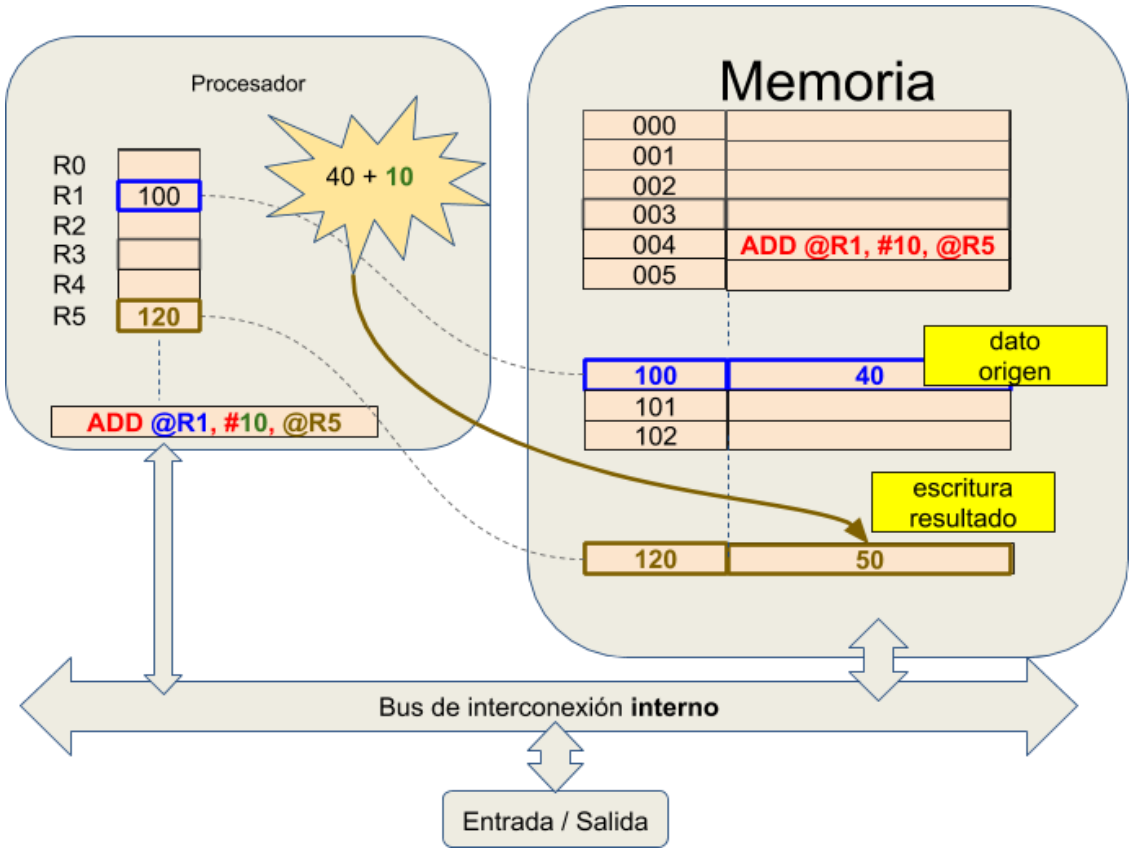
El contenido del registro se interpreta como la dirección de memoria donde está el dato. Es decir, la instrucción viene a decir "yo no sé dónde está el dato, lo sabe ese registro". En el caso de la instrucción anterior, para el primer operando (@R1), el siguiente esquema describe la situación.

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES Unión Europea Fondo Social Europeo El FSE invierte en tu futuro



El operando calificado como "@R1" está usando el modo indirecto a registro. En el ejemplo, el contenido de R1 es "100". Usando este modo de direccionamiento, ese 100 se interpreta como "la dirección 100 de memoria" y por tanto una vez leído el registro y descubierto este valor, la UC solicitará de la memoria el contenido de la celda 100.

Veamos el ejemplo completo con más datos de ejemplo de la instrucción anterior.



Esta instrucción del ejemplo es

ADD @R1, #10, @R5

Tal como aparece en el ejemplo, en la dirección 100 de memoria, hay un dato que corresponde a la cantidad 40. Este 40 es el dato que finalmente se envía a la ALU para realizar la suma.

El otro operando "#10" tiene un modo de direccionamiento inmediato correspondiente a la cantidad 10, no hay que leer nada de ningún sitio más, el dato venía con la instrucción. Por lo que la suma final, en este ejemplo es de 10 + 40. El resultado debe almacenarse en algún lugar

El destino de la suma, está especificado en la instrucción, otra vez usando el modo indirecto a registro "@R5". La suma va a guardarse en la dirección que esté anotada en el registro R5. En este caso, en R5, el dato es 120, luego es en la celda 120 donde se guarda el resultado en la última fase de la instrucción

Este modo puede usarse para especificar datos origen o destino.

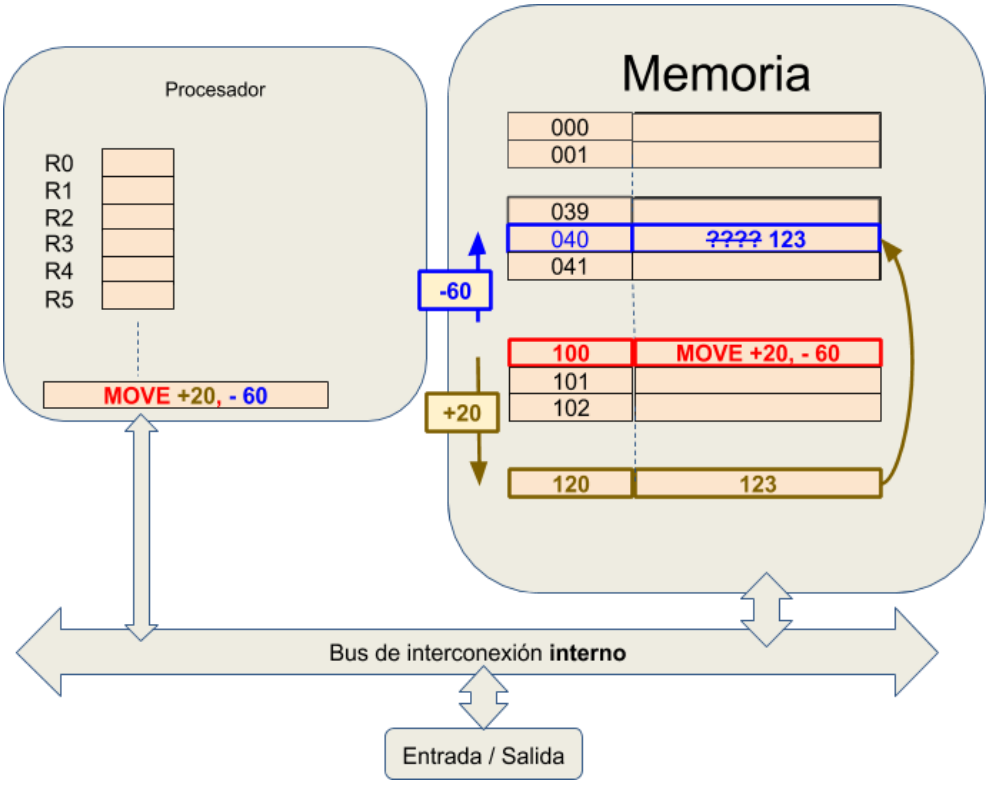
Author. Ignacio Pachés Giner

Modo relativo.

El modo relativo es un poco extraño pero fácil de entender. Otra vez estamos ante un modo que toma o deja los datos en memoria. Pero se llama *relativo* porque el lugar o la dirección de la celda de memoria depende de la celda de memoria donde está la instrucción que utiliza ese modo. Lo entenderemos mejor con un ejemplo en el que también veremos la manera de escribir estas direcciones:

MOV +20, -60

Lo que distingue al modo relativo son esos símbolos "+" y "-" antes de los números. Están indicando un desplazamiento en memoria respecto la posición de la instrucción. Es decir, para saber dónde está el dato origen, primero hemos de saber dónde está la propia instrucción que está ejecutándose. Vamos pues, a suponer una situación completa



Para entender este ejemplo, es muy importante darse cuenta de que la instrucción en ejecución está en la dirección 100. Esa dirección se toma como base para el direccionamiento relativo. En el caso del dato origen, que está calificado como "+20", se debe sumar 20 direcciones a la base de la instrucción. Esto es, 20 + 100 = 120. Por tanto, la dirección de memoria definitiva donde está el dato origen es 120. De forma análoga, el dato destino debe depositarse en una dirección que surge de tomar la dirección de la instrucción (100) y sumarle el desplazamiento indicado (-60), que es negativo (es como si restáramos). 100-60 = 40 . Por tanto, el dato ha de moverse (copiarse realmente) a la dirección 40

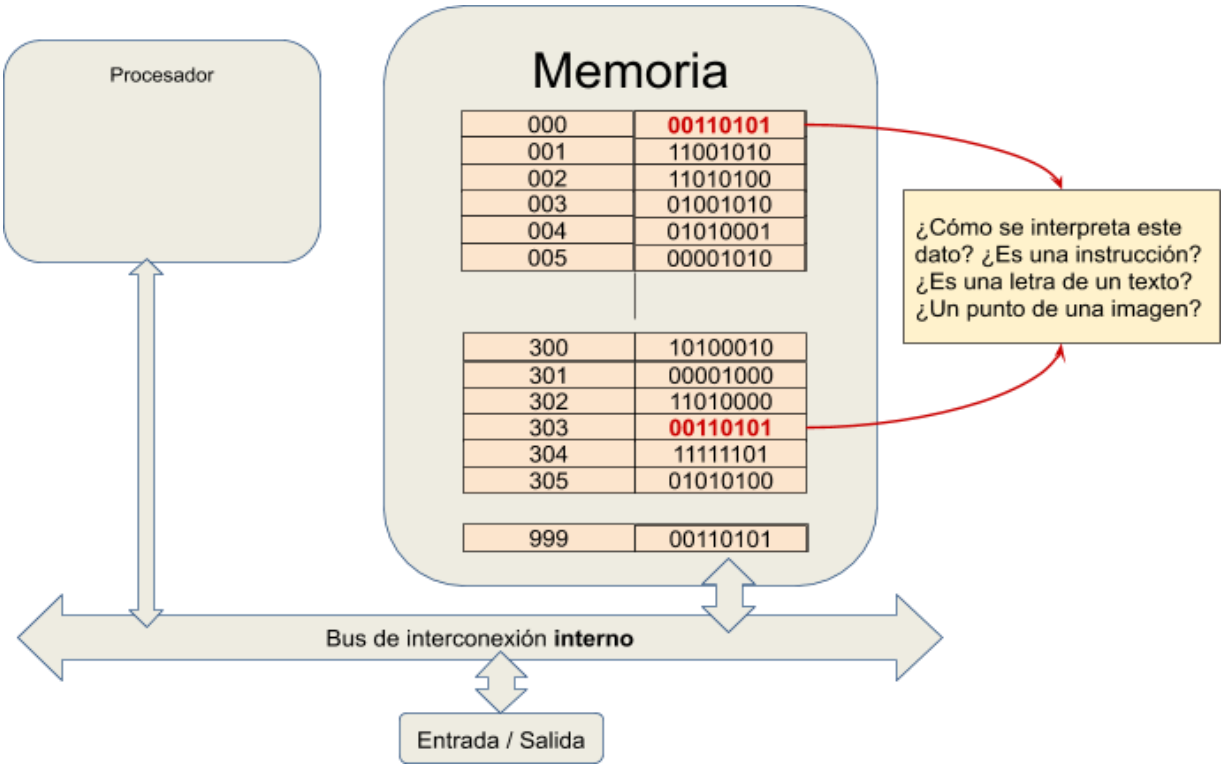
GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea Fondo Social Europeo El FSE invierte en tu futuro

Author. Ignacio Pachés Giner

Contenido de la memoria

Aunque ya se ha comentado acerca de la estructura de la memoria y su contenido ocasionalmente, es hora de reflexionar y aclarar algunas posibles dudas. La pregunta que hay que hacerse, es : "Cuando el ordenador está en funcionamiento ¿Cómo se distinguen las celdas que contienen datos de las que contienen instrucciones?

En principio no hay nada que lo diferencie. Si pudieses hacer una foto a la memoria del computador en un momento dado y ver el contenido de la memoria, no sabrías si los ceros y unos almacenados en una celda corresponden a un dato, una instrucción, etc. Dicho esto parece que haya un lío muy grande.

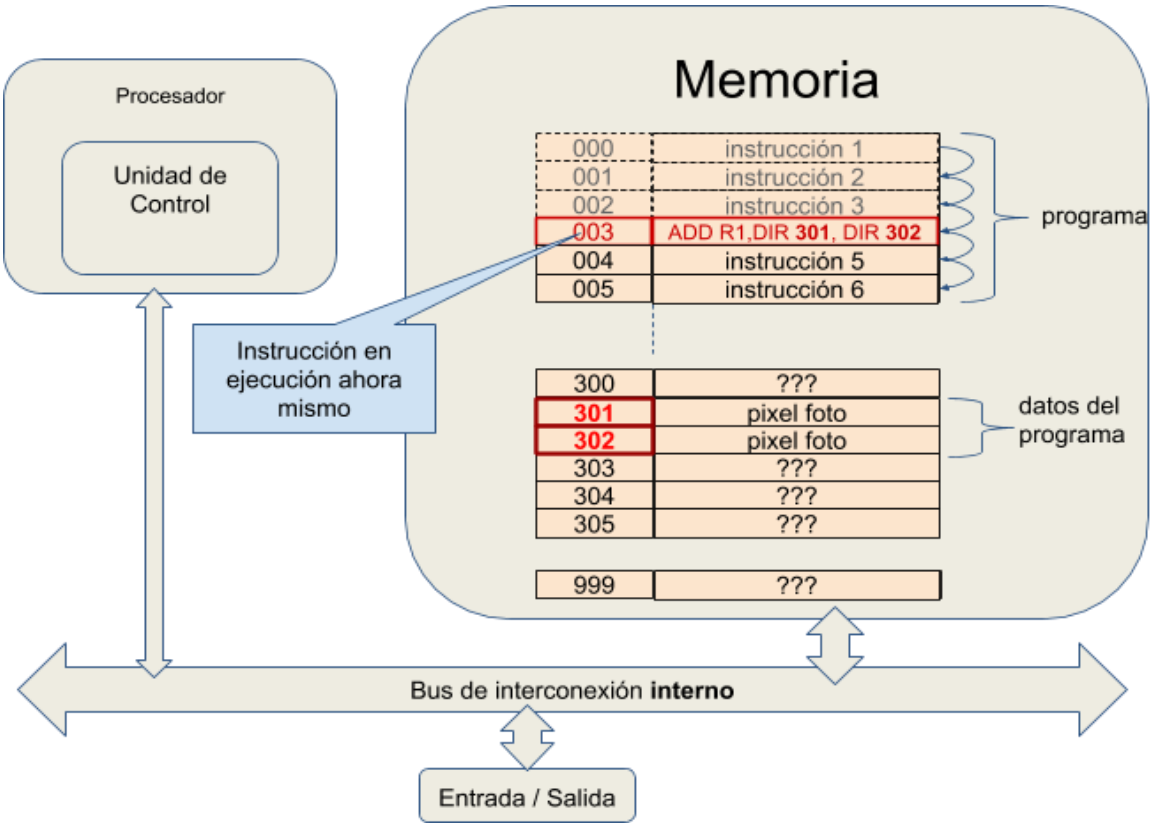


En este diagrama, se presenta una memoria en la que dos celdas diferentes y distantes tienen el mismo contenido en binario. ¿Significa eso que tienen la misma información ambas? Su contenido ¿Es el mismo en las dos? Por ejemplo, si una contiene un punto de una imagen que el usuario está viendo por su pantalla, ¿En la otra también se trata de un punto de una imagen?

La realidad es algo más ordenada. Supongamos que algo o alguien ha situado las instrucciones del programa en la memoria de forma correcta. Cuando éstas se ejecutan, van leyendo y escribiendo datos de la memoria. Esto datos estarán en algún otro lugar de la memoria. En esas celdas donde las instrucciones de un programa escriben o leen datos, es de esperar que el contenido sea interpretado como datos de un programa. Por ejemplo, supón que un programa de tratamiento de imágenes está situado en las direcciones de memoria de la 000 a la 005. En un momento determinado, se ejecuta la instrucción en la dirección 003, que lee un dato de la dirección 301 de memoria y escribe otro dato en la 302. Éstas dos últimas celdas corresponden a datos del programa, que, al ser de edición de imágenes, podemos sospechar que se tratan de información sobre la imagen que se está editando

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea Fondo Social Europeo El FSE invierte en tu futuro

Author. Ignacio Pachés Giner



Por otra parte, aunque no nos lo digan, si se ejecuta la instrucción en la dirección 003 y es una instrucción normal, entonces en la dirección 004 también habrá una instrucción porque después de una se ejecuta la siguiente (salvo saltos)

Decimos que "la ejecución de un programa va dando interpretación al contenido de la memoria"

Tipos de datos almacenables en memoria

Vamos a hacer un ejercicio de observación del mundo real e intentar razonar sobre todos los tipos posibles de datos que puede almacenar una memoria. Hasta ahora se ha visto que en una celda pueden haber instrucciones y también como ejemplo, hemos considerado que pueden haber datos de una fotografía (una fotografía son un montón de puntitos de color, cada puntito está almacenado en memoria)

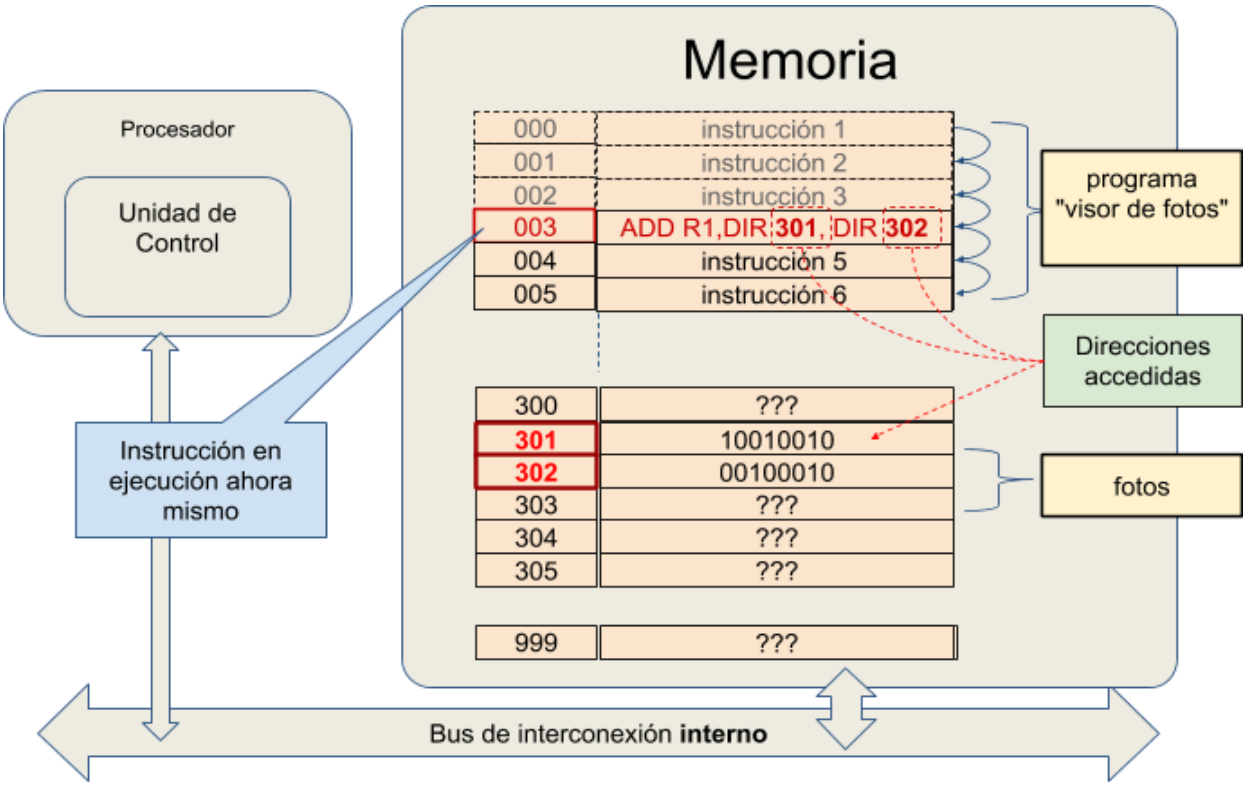
Observando los programas que utiliza un usuario normal en su casa, en algún momento, en una celda de memoria puede haber almacenado:

- Letras de un texto, presentación, hoja de cálculo
- Datos de un dibujo.
- Texto o datos de una página web
- Datos sobre sonidos que componen una canción.

Observando los programas que utiliza un usuario profesional en su trabajo, en algún momento, en una celda de memoria puede haber almacenado:

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea Fondo Social Europeo El FSE invierte en tu futuro

- Datos de un plano,
- Letras y símbolos en un diagrama
- Un Dato de una base de datos
- Datos de una factura, albarán, pedido, etc.



Hasta aquí es suficiente reflexión, conforme vayamos avanzando iremos descubriendo que en la memoria se pueden llegar a almacenar datos y otras cosas que ahora mismo no alcanzamos a comprender.

Tamaño de los datos y celdas de memoria.

Todas Las memorias tienen como unidad de almacenamiento básico el byte (8 bits). Es decir, cada celda es capaz de almacenar un byte, 8 bits, ni más ni menos. La mayoría de información que manipulamos en un ordenador, requiere muchas celdas de memoria para ser almacenado. Por ejemplo, en una fotografía, cada punto de la imagen requiere 3 celdas de memoria por cada punto. Si la imagen tiene una resolución de 2500x2000 píxeles (5MegaPixeles), el tamaño que ocupa en memoria esta imagen pequeña es de 15 millones de celdas (unos 15Mb) (otra cosa es lo que esa imagen pueda ocupar si se comprime a la hora de escribirla en un disco duro)

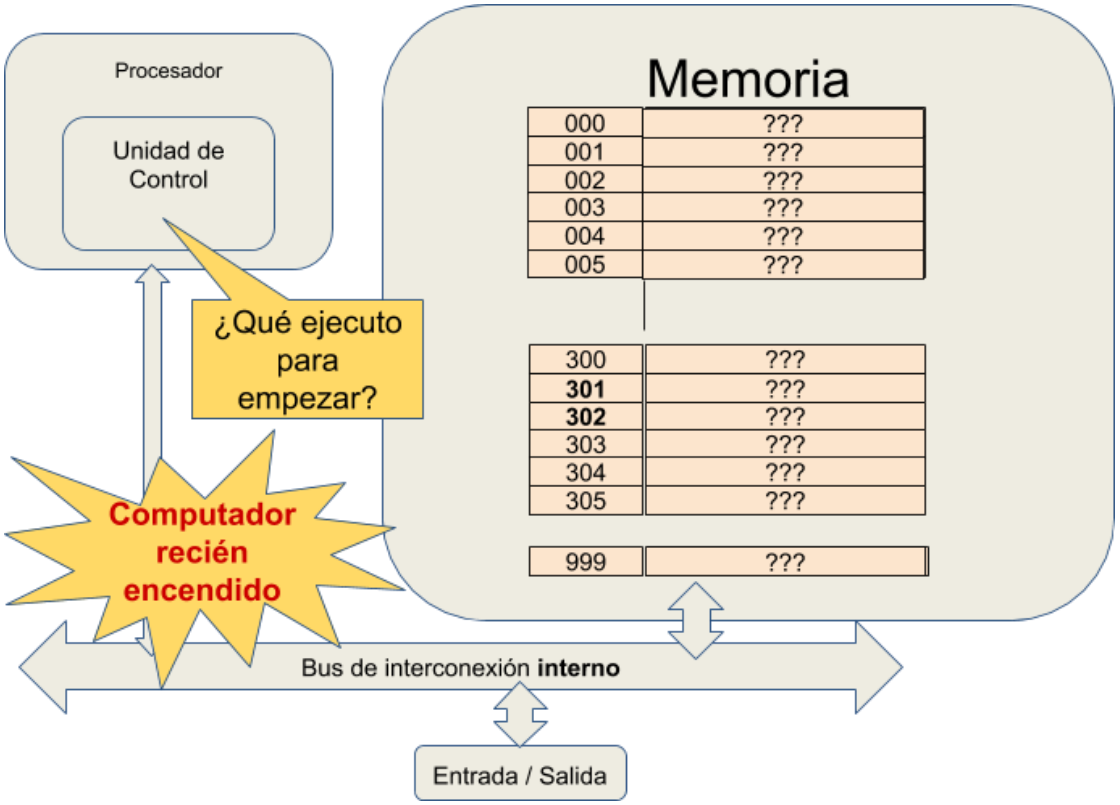
En este tutorial vamos a seguir ignorando este hecho y los datos e instrucciones tendrán el tamaño que más se ajuste a la explicación.

Tipos de memoria.

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea Fondo Social Europeo El FSE invierte en tu futuro

Author. Ignacio Pachés Giner

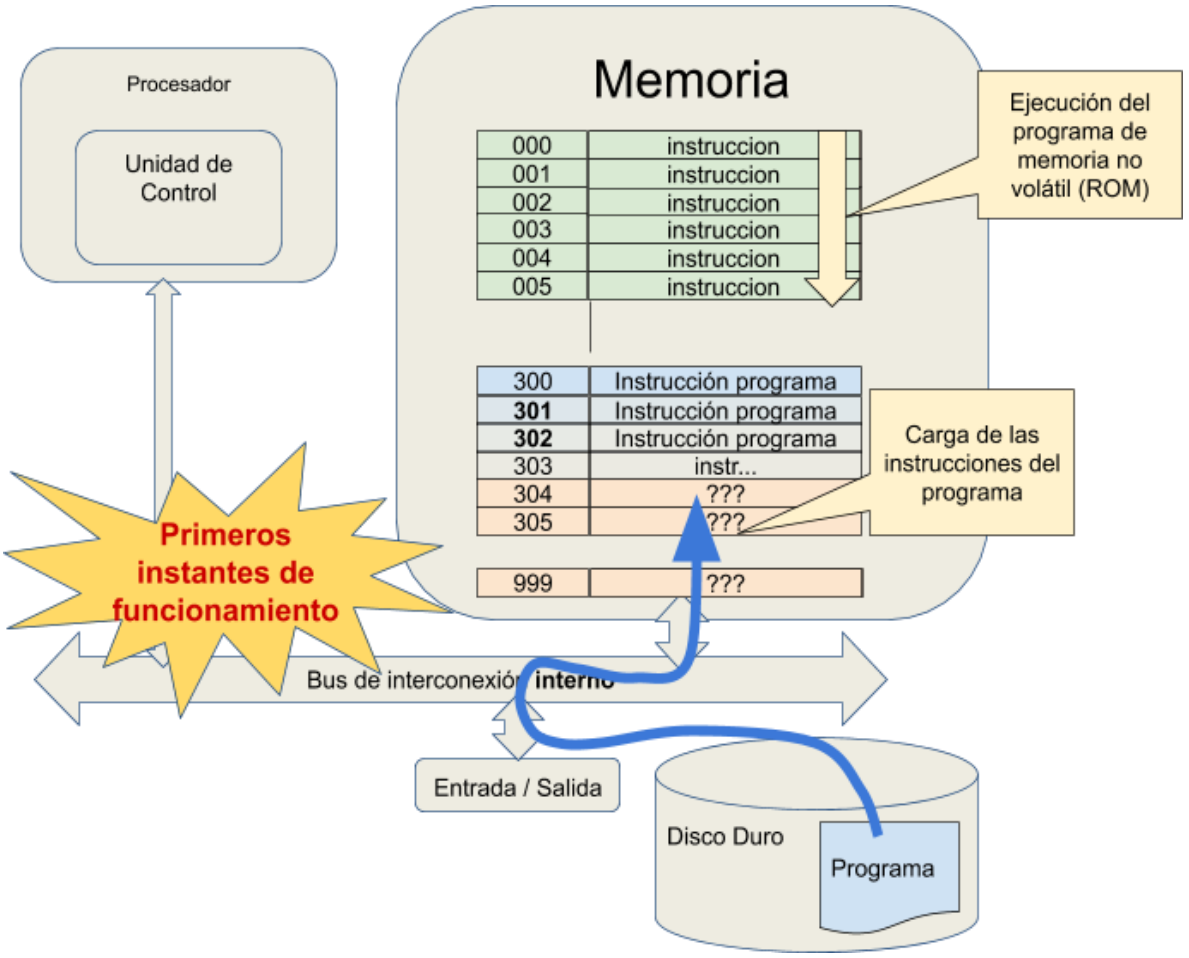
La memoria es un dispositivo totalmente electrónico que está -en principio- construido con una tecnología que no es capaz de retener los datos cuando se desconecta de la corriente. Es decir, un programa en ejecución y todos los datos se pierden al apagar el ordenador. Como puedes suponer, esto supone un impedimento total para fabricar un computador, porque al ponerse en marcha, el procesador no tendrá nada válido que ejecutar en la memoria.



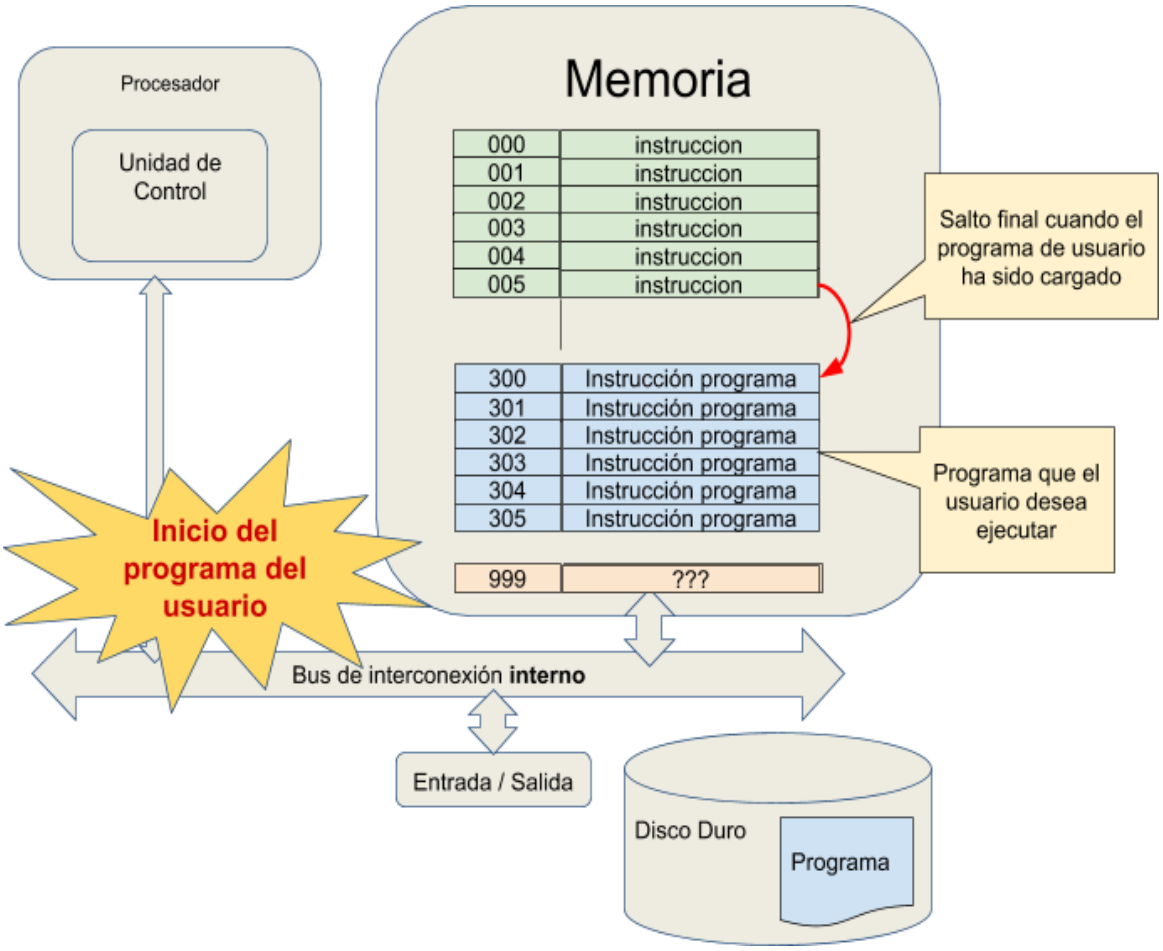
La solución consiste en fabricar algunas celdas de memoria de una manera que permita que su contenido no se pierda al apagar el ordenador. Hay diferentes tecnologías para fabricar este tipo de memoria "que no se borra", las más antiguas son memorias que se fabricaban con un contenido concreto y nunca más se podían cambiar. Las más recientes son memorias casi normales que retienen su contenido al desconectarse de la corriente eléctrica, pero pueden ser fácilmente escritas. En cualquier caso, es el fabricante del ordenador el que escribe el contenido.

Si construimos un computador de forma que el programa resida en un bloque de celdas que no se borre y el procesador, al iniciarse, empiece a ejecutar instrucciones por ese bloque, conseguiremos un ordenador que siempre que se enciende de nuevo ejecuta el mismo programa:

Gobierno de España · Ministerio de Educación, Cultura y Deporte · Secretaría de Estado de Educación, Formación Profesional y Universidades · Unión Europea · Fondo Social Europeo · El FSE invierte en tu futuro



De esta manera, el computador al iniciarse puede ejecutar un programa ya. Es de esperar que el programa cargado en la RAM no sea el que el usuario desea ejecutar. La misión de dicho programa "permanente", es cargar las instrucciones de otro programa en la memoria para permitir que se llegue a ejecutar a continuación.



La memoria volátil recibe el nombre de RAM (Random Access Memory) que es un pésimo nombre hoy en día para ese dispositivo. La memoria no volátil ha recibido diferentes nombres según la tecnología ha ido avanzando y permitiendo más funciones: ROM, EPROM, PROM, EEPROM, FLASH, Firmware.

Arranque, carga y ejecución de los programas.

En otro tema sobre introducción al sistema operativo, descubriremos importantes cambios y ampliaciones sobre este apartado.

Author. Ignacio Pachés Giner

Técnicas de aceleración del Computador

El procesador ejecuta instrucciones a un ritmo muy alto. Un humano no puede percibir que las instrucciones van ejecutándose una a una. De hecho, la medida o unidad habitual para medir el ritmo al que un procesador ejecuta instrucciones son los MIPS, que es Millones de Instrucciones Por Segundo. Pese a ello, siempre es conveniente acelerar la ejecución del programa y ser capaz de ejecutar más instrucciones por unidad de tiempo.

Hay que tener en cuenta que, en la ejecución de una instrucción, el elemento más lento entre CPU-memoria y entrada/salida, es el que determina el tiempo final que se necesitará para completar la instrucción. Estamos simplificando bastante el análisis del computador y ciñéndonos a instrucciones que trabajan con la memoria y el procesador únicamente. Y siendo así, hay que decir que, entre ellos, la memoria es el componente más lento y que por tanto supone un cuello de botella en principio.

Segmentación y ejecución superescalar

La segmentación consiste en diseñar un procesador para que pueda ejecutar las fases de las instrucciones independientemente unas de otras y encolar las operaciones básicas. Es decir, un procesador segmentado puede estar ejecutando una suma en la ALU, y tener simultáneamente una operación de suma esperando a que la ALU termine para inmediatamente realizar empezar la suma. Podemos pensar que las instrucciones se ponen en cola para realizar sus operaciones. Y el procesador va "despachando" esas peticiones.

Para que esto sea posible, los procesadores son superescalares. Esto es, son capaces de estar ejecutando una instrucción y al mismo tiempo estar también preparando la ejecución de la instrucción siguiente. Suponga el siguiente mini programa

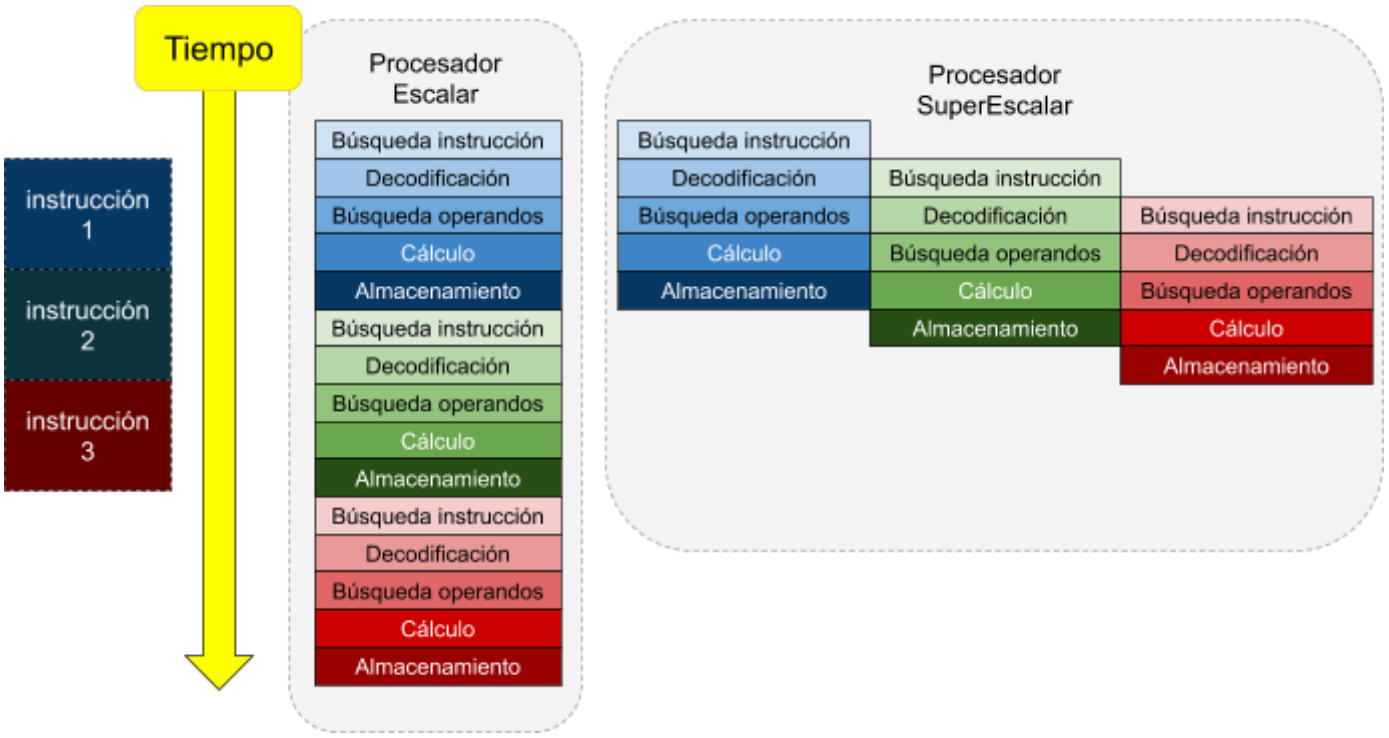
MULT R1,R2,R3

MOVE DIR 100, DIR 200

En ese par de instrucciones, primero se leerá la instrucción MULT, se decodificará (la UC sabrá entonces qué es lo que hay que hacer), dado que los datos están en los Registros y éstos son muy rápidos, instantáneamente se pasan los datos a la ALU para que haga la multiplicación. Sin embargo, esta multiplicación es muy lenta y requiere mucho tiempo. Un procesador superescalar, será capaz de empezar a leer de la memoria la siguiente instrucción (MOVE…) mientras está calculándose la multiplicación en la ALU de la anterior instrucción. Así mismo, la instrucción MOVE podría completarse incluso antes que termine la multiplicación de la instrucción anterior

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea Fondo Social Europeo El FSE invierte en tu futuro

Author. Ignacio Pachés Giner



La segmentación + ejecución superescalar es una técnica muy buena, porque no cambia la manera de programar, sin embargo, complica muchísimo el diseño de los procesadores y de la Unidad de Control que ha de asegurarse que una instrucción finalizando después de una posterior no rompe la lógica del programa. Es decir, el siguiente programa funcionaría mal si la segunda instrucción empieza antes que termine la primera

MULT R1,R2,R3

MOVE R3, DIR 200

En ese caso, la UC, sin ayuda de nadie debe darse cuenta de que la segunda instrucción debe esperar a que termine la primera.

Incremento del número de registros

Dado que los datos de la memoria tardan mucho tiempo en llegar a la U.C. en comparación con los datos almacenados en los registros, se plantea aumentar el número de registros de forma que no sea necesario guardar datos en la memoria por culpa de tener pocos registros.

Esta técnica tiene una gran desventaja que ha lastrado su expansión. Los programadores deben cambiar sus programas cuando aparece un modelo de procesador nuevo que tiene más registros, ya que el programa, cuando se escribe o diseña, sólo puede ejecutarse con los procesadores existentes en ese momento en el mercado.

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea Fondo Social Europeo El FSE invierte en tu futuro

Author. Ignacio Pachés Giner

Aumento de la frecuencia del reloj

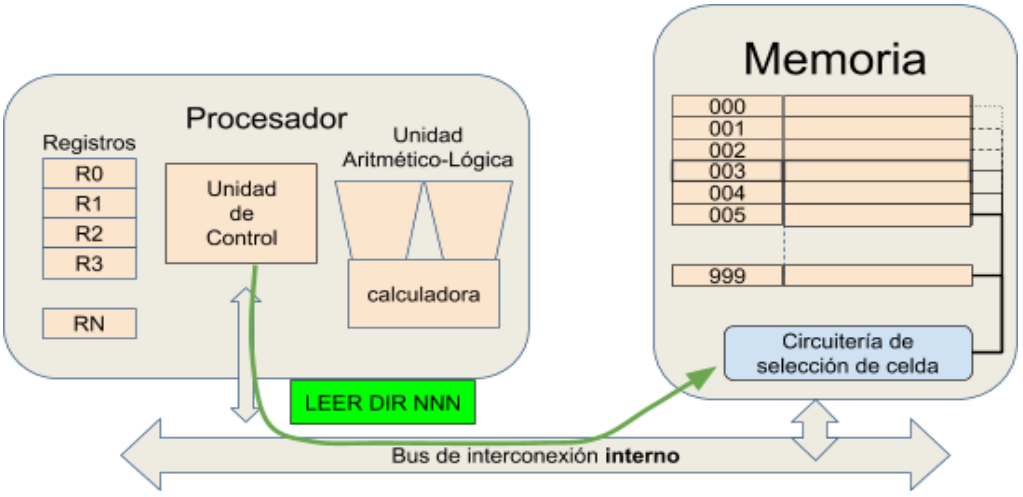
Toda la circuitería del procesador y del computador en general, va a un ritmo marcado por una señal eléctrica alternante. Es como los bombos que usaban en los barcos antiguamente para que todos los remeros fuesen al mismo ritmo y el barco avanzase con efectividad sobre el agua. Si había que huir del enemigo, el encargado de aporrear el bombo, aumentaba el ritmo para que los remeros, yendo al compás remasen más rápido.

Esta señal se llama "reloj" (¡Cuidado!, hay otras cosas a las que también se llaman reloj en el computador). Lo ideal es que el reloj lleve la frecuencia más alta posible, porque todo el ordenador irá más rápido. Sin embargo, hay límites físicos que si se superan provocan fallos, errores e incluso roturas en el computador. Durante mucho tiempo, los fabricantes mejoraban los procesos de fabricación de los procesadores para hacerlos capaces de funcionar a una frecuencia cada vez mayor. Desde hace unos 10 años, apenas se ha avanzado en la frecuencia admisible (en torno a 1.4 a 4 GHz para ordenadores personales normales), y las mejoras se consiguen con otras técnicas.

Memorias "caché"

La decodificación de la dirección de celda

Piensa en un restaurante con una carta muy pequeña de platos. El libro de recetas tiene pocas páginas y cuando el cocinero lo ha de consultar, apenas pierde tiempo buscando en su índice y localiza la página a la que acudir con rapidez. Por contra, con un libro de recetas muy muy grande, a cada paso de receta, el cocinero ha de perder mucho tiempo buscando en el índice y yendo a la página a consultar el siguiente paso de recetas. Un ejemplo similar ocurre con las oficinas de correos, cuánto más pequeño es un municipio, más fácilmente organizan el reparto. El tiempo que tarda una carta en llegar al destinatario no sólo es el que tarda el cartero en pasearse desde la oficina al domicilio. También hay que organizar las cartas que llegan a la oficina y repartirlas en sacos para que los distintos carteros se encarguen de diferentes barriadas. Ese tiempo es más pequeño cuantas menos calles hay.



GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea Fondo Social Europeo El FSE invierte en tu futuro

Este ejemplo es aproximado, pero describe el problema fundamental en la rapidez de respuesta de las memorias. La memoria también tarda un tiempo en averiguar dónde está la celda que una dirección le indica. La memoria tiene una especie de oficina de correos interna. En general, a mayor tamaño, mayor tiempo en responder electrónicamente a las peticiones de lectura o escritura.

Hay un fenómeno o consecuencia asociado al funcionamiento de la circuitería que localiza una celda y que resulta fundamental para entender lo que viene después: A la memoria le cuesta lo mismo encontrar una celda y activarla, que encontrar esa celda y todas las que le siguen a continuación en bloque. Es decir que, si queremos leer dos o más celdas consecutivas, sólo tarda mucho al encontrar la primera, el resto se pueden leer o escribir secuencialmente sin apenas esperas después de la primera. Al cartero le da igual si en una calle hay muchas cartas que repartir. Lo que más tiempo cuesta es todo lo que precede al momento en el que finalmente pisa la calle. Una vez allí puede repartir muchas cartas sin esfuerzo y rápidamente.

Como conclusión, podemos afirmar que cuesta lo mismo leer una celda solitaria que leer todo un bloque consecutivo de celdas.

La memoria como dispositivo alejado del procesador

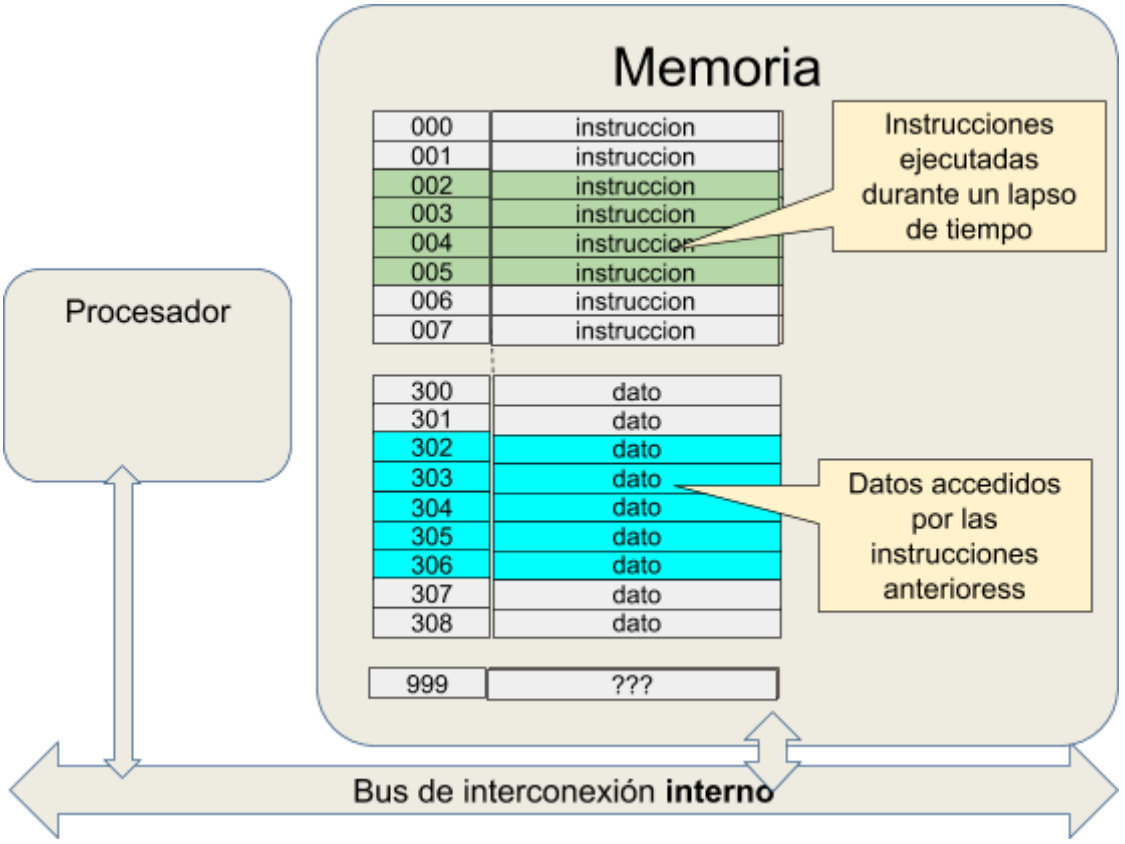
Supón ahora que el cocinero tiene un libro de recetas tan pequeño que lo puede llevar encima. Cada vez que consulta un paso, no tiene que desplazarse hasta el final de la cocina. Allá donde está, abre el recetario y consulta lo que tiene a mano. De la misma manera, la memoria es una placa con chips que está "lejos" de la CPU, en comparación con lo próximos que están fabricados, la unidad de control, los registros y la ALU

El principio de localidad.

Otro fenómeno que ocurre igual en el ejemplo del cocinero y del procesador, es que si observamos al cocinero durante un espacio de tiempo no muy grande, resulta que todas las consultas que hace al libro de recetas, son a la misma receta, que es la que corresponde al plato que está preparando en ese tiempo. En algún momento finalizará ese plato y pasará a preparar otro, pero en entonces, volverá a pasar tiempo mirando la página donde están los pasos de receta de ese nuevo plato.
Habrá habido sólo un momento en el que ha cambiado de mirar siempre una o dos páginas consecutivas, a mirar otra diferente y quedarse allí "un rato"

En el computador ocurre algo similar. Si observamos al procesador durante periodos de tiempos no muy grandes, lo más probable es que todas las instrucciones y datos a los que accede en memoria estén próximos a otras instrucciones y datos respectivamente. El caso más claro se puede imaginar si no hay instrucciones de salto de por medio: cada vez se ejecuta la siguiente instrucción y por tanto todas las instrucciones ejecutadas durante un ratito estarán juntas en un mismo bloque. Con los datos ocurre algo similar. Imagina un programa que está manipulando una foto. Todos los datos de la foto están en memoria en un bloque contiguo. Normalmente se trabaja y procesa la foto entera, lo cual significa que algún bucle recorre todas las celdas del bloque efectuando alguna operación.

A esto se le llama *principio de localidad* y se puede enunciar como el fenómeno por el que estadísticamente es probable encontrar que todas las instrucciones que se ejecutan durante un periodo de tiempo pequeño están juntas entre sí, así como los datos a los que acceden.



La memoria Caché.

La memoria caché es una solución a los problemas de lentitud de la memoria, aprovechando todo lo explicado antes:

- Lentitud en encontrar una celda, pero rapidez en encontrar las siguientes secuencialmente
- Lentitud al tener que viajar los datos por el bus entre la memoria y el procesador.
- Principio de localidad.

La idea básica es tener los bloques de celdas que están frecuentemente siendo accedidos más rápidamente disponibles para el procesador.

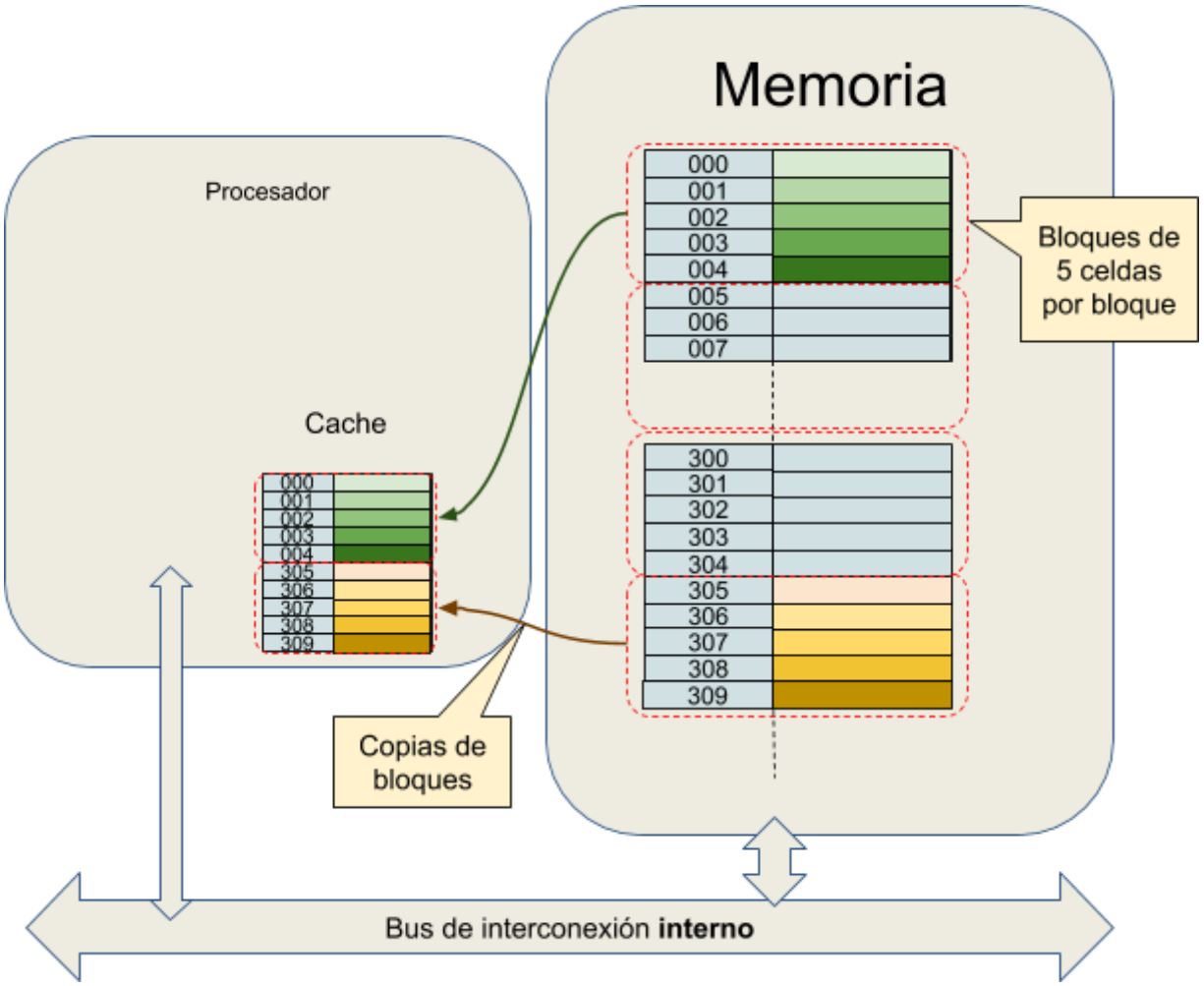
A los "*bloques de celdas consecutivas*" también les llamaremos "*páginas* ". En un computador las páginas tienen siempre el mismo tamaño (por ejemplo, 4Kbytes)

Author. Ignacio Pachés Giner

La caché es una memoria pequeña, rápida y organizada de forma similar a la RAM que:

- Es pequeña y por tanto más rápida en localizar la celda requerida.
- Se fabrica en el propio chip del procesador.
- Se fabrica con otra tecnología distinta a la memoria principal que es más rápida (por las dos razones anteriores).
- El procesador no lee de la memoria RAM que conocemos; sino que leerá de la memoria caché y por tanto funcionará más rápida la ejecución del programa.

- La memoria caché mantiene copias de las páginas de memoria que van siendo accedidas al ejecutar un programa. No mantiene una copia de todo, sino **de las páginas que están siendo accedidas en cada momento**. Ahí está la clave de la mejora.
- De la memoria principal (RAM) a la caché siempre se copian páginas (recuerda todas las páginas tienen el mismo número de celdas)
- Dado que la memoria caché es autónoma y transparente (ella hace las copias conforme ve que es necesario y nadie se entera), su instalación no afecta ni al programa, ni al programador, ni al procesador ni a la memoria. Es transparente para todos.



El funcionamiento es el siguiente:

1. El procesador emite una dirección y una señal de lectura hacia la memoria.
2. Esta señal pasa primero por la caché que tiene un registro de todos los bloques de celdas (páginas) que ha copiado.
3. Si la celda solicitada por el procesador pertenece a una página alojada en la cache, se entrega ese dato al procesador (que pensará que se lo ha entregado la memoria) directamente leído desde la caché

4. Si la celda solicitada no está copiada en la caché, hay que copiarla
    a. Se localiza o decide buscar un bloque de celdas libres en la cache. Seguramente no habrá ninguno libre
    b. Se determina qué bloque de celdas tiene una copia de un bloque de RAM que no haya sido utilizado en mucho tiempo, y se marca como "obsoleto"
    c. Se copia de la memoria la página solicitada encima de la página que se ha decidido sobreescribir.
5. Cuando esto termina, se entrega el dato solicitado originalmente al procesador

La caché se dice que es "**transparente**" a los programas, la memoria y a la CPU porque actúa de forma autónoma. Es una circuitería que una vez añadida y puesta en el computador realiza su trabajo con las siguientes características:

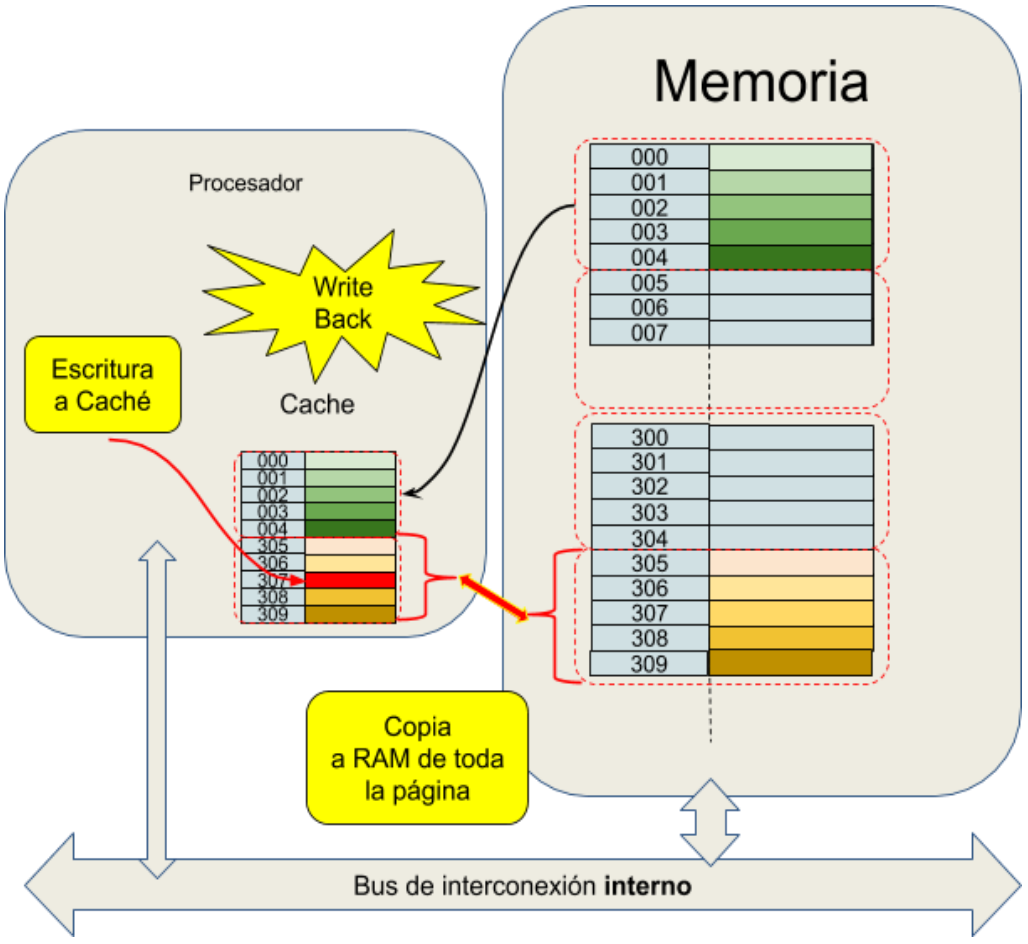
- Detecta automáticamente las lecturas o escrituras en memoria por parte de la CPU y los datos que se pretende escribir.
- Realiza lecturas de bloques de la memoria sin que esas lecturas se diferencien de las realizadas por la CPU.
- Los datos solicitados por la CPU son entregados por la caché y recibidos por la CPU de la misma manera que lo serían si fuesen entregados directamente por la memoria (salvo por la rapidez)
- No hay instrucciones del procesador capaces de interaccionar con la caché específicamente. Es decir, no se puede "hacer una lectura de los datos de la caché", se lee de la memoria, pero es la caché la que suministra el dato interponiéndose entre la CPU y la memoria.
- El programador puede hacer un programa y experimentar el rendimiento con diferentes computadores para medir el efecto de una u otra cache, pero el programa funcionará con o sin caché en todo, excepto en la rapidez.

Este sistema es muy efectivo y se usa mucho, pero hay que citar algunos problemas

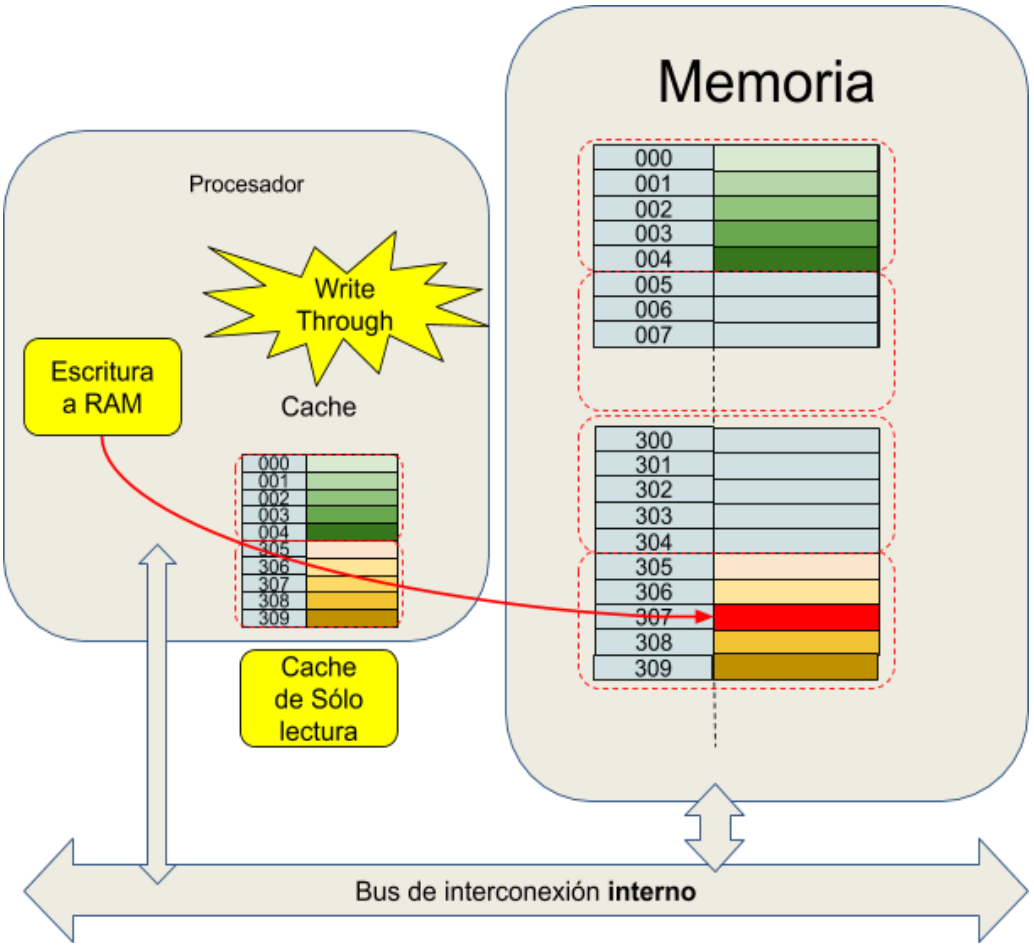
- La **fabricación** del procesador se complica.
- En algunos casos puede provocar una pérdida de rendimiento. Esto ocurre con programas que realizan frecuentes **saltos lejanos** y que obligan a leer todo un bloque de celdas cuando ocurre ese salto para traer a la caché.
    - A veces se está más tiempo copiando bloques que aprovechando la rapidez de la caché.

- Funciona muy bien en lectura, pero en **escritura** al tener que cambiar el dato en la memoria RAM

  (y no en la caché, que sólo guarda copias), se plantean varias soluciones y ninguna es perfecta.
  - write through. El dato se escribe en la caché (si está la dirección accedida) y en la memoria RAM
  - write back. El dato sólo se escribe en la caché y más tarde se copia la página de la caché a la memoria



Author. Ignacio Pachés Giner



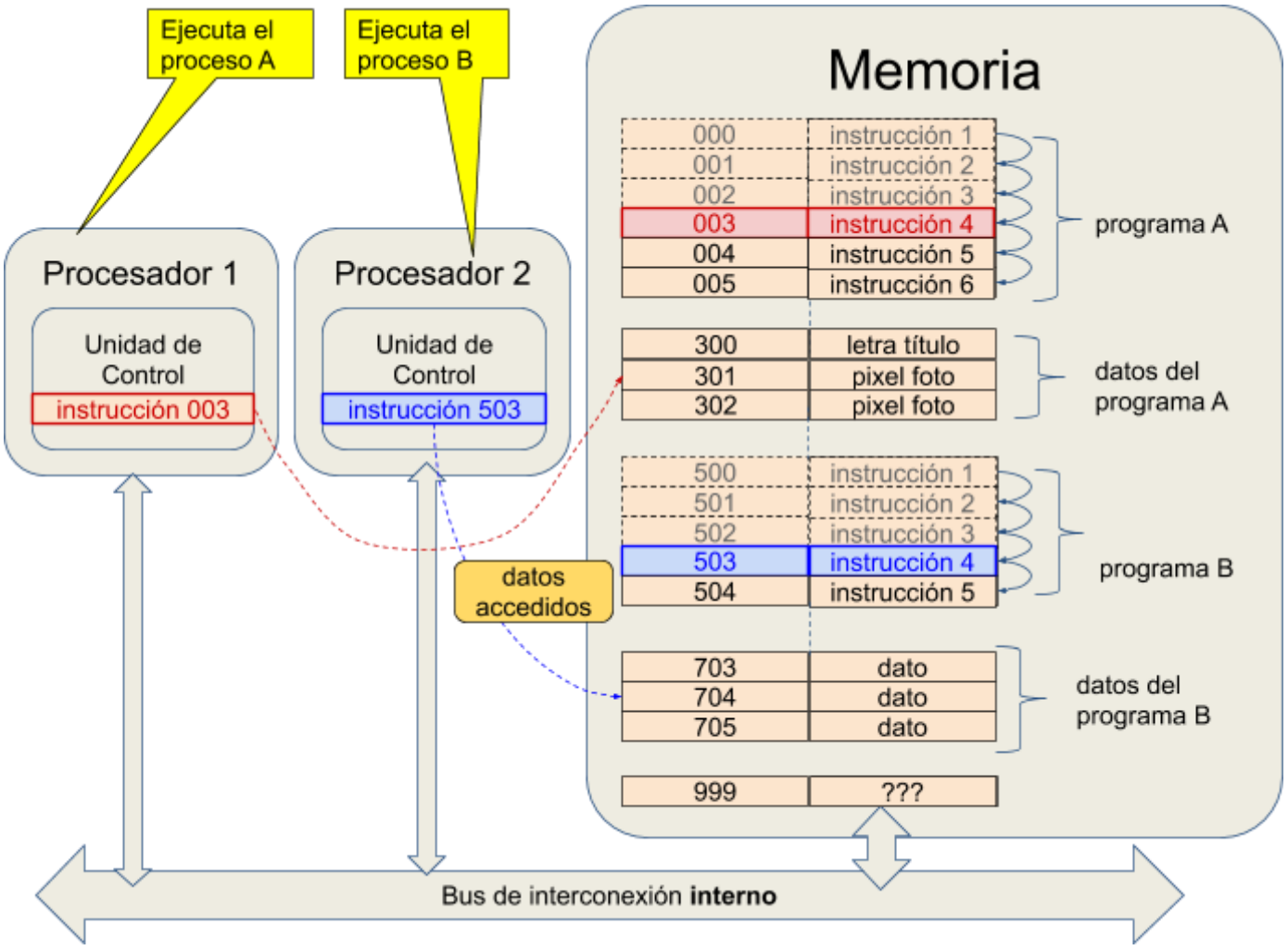
Multiprocesador /Multicore

Un computador multiprocesador dispone de varios procesadores pero sólo una memoria que es común y accesible por ambos indistintamente. Cada procesador puede escribir y leer de cualquier posición de memoria.

Cada procesador ejecuta instrucciones de un proceso (como si sólo estuviese ese procesador presente). Dado que distintos procesadores no coinciden en ejecutar instrucciones del mismo proceso, tampoco ocurre que los accesos a datos coinciden en las mismas zonas de datos.

Por ejemplo, puedes tener un proceso o programa para escuchar música accediendo a una canción almacenada en la memoria RAM, y por otra parte, un proceso para editar fotografías, accediendo a otra zona de memoria RAM diferente.

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea Fondo Social Europeo El FSE invierte en tu futuro



Aunque el multiprocesador no es lo único que permite que dos procesos funcionen a la vez (veremos en otro tema cómo), sí que acelera la ejecución general de un computador en el que se quieren hacer varias cosas a la vez

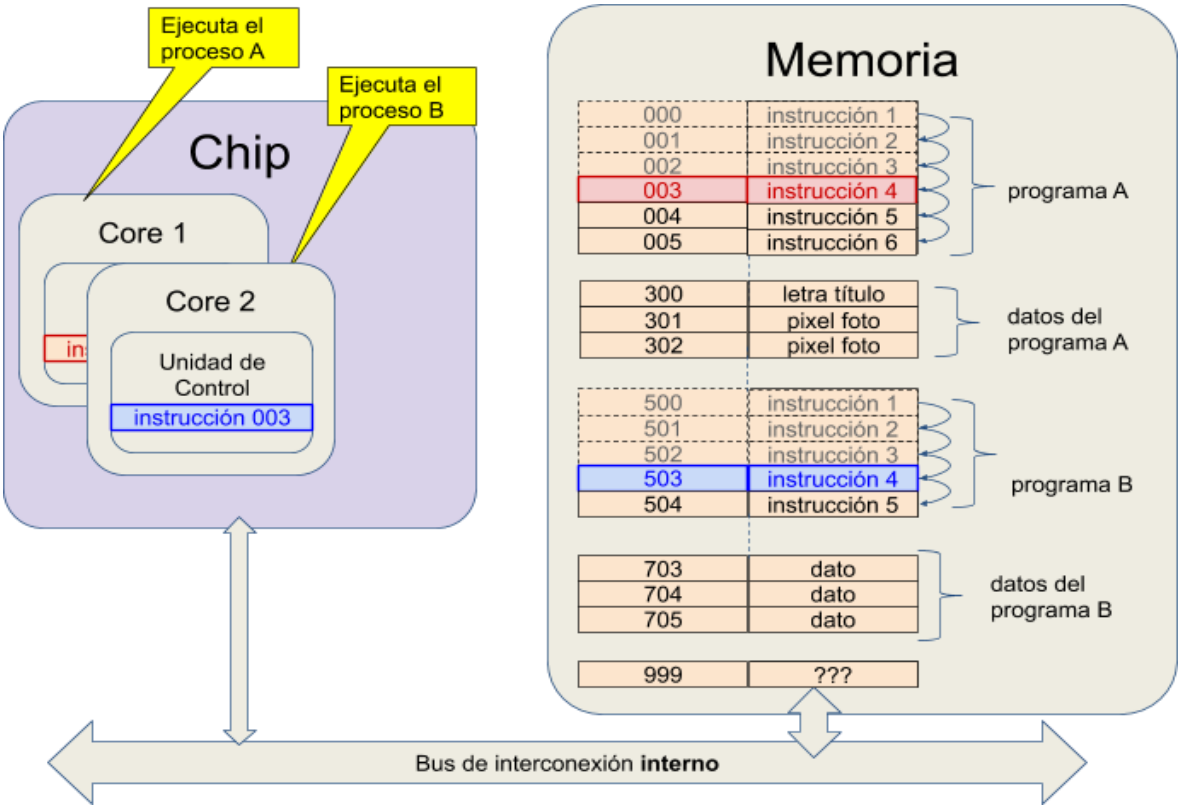
Multicore

"Multicore" es multiprocesador, pero con la diferencia de que los varios procesadores se fabrican juntos todos en el mismo "chip". Al estar varios procesadores dentro del mismo chip, surge una confusión con los nombres de forma que cuando se habla de procesadores modernos y multicore:

- Procesador o CPU: Es el circuito visible que se monta sobre la placa madre del ordenador y que internamente está compuesto de varias unidades equivalentes a lo que hemos ido viendo como procesadores.
- "core": Un core es una circuitería que es un procesador completo como el que se ha estudiado aquí.

Author. Ignacio Pachés Giner

Podemos decir que "un procesador moderno lleva varios cores", mientras que antiguamente "un procesador no tenía varios cores"



Después, aparecen otras pequeñas diferencias; por ejemplo, comparten una caché grande y cada cual tiene su propia caché, todo fabricado en el propio circuito. Otra diferencia es que los procesadores pueden ser diferentes . En algunos procesadores Quad Core (4 cores/procesadores) y OctaCore (8 cores), hay unos cuantos cores que funcionan a una frecuencia del reloj baja y consumen muy poca electricidad, generando poco calor, Para que cuando el sistema no tenga muchos programas prioritarios que ejecutar, se consuma poco. Pero Cuando se ejecuta un programa que requiere mucha rapidez de cálculo, entonces se activan y trabajan los procesadores / Cores de alta velocidad.

La palabra Multicore ha relegado a la palabra "multiprocesador" al significado de "diferentes chips de procesadores". Así, hoy en día, un computador multiprocesador, puede tener dos o más chips, cada uno con varios cores. Pero en el fondo un computador multicore y multiprocesador no se distinguen para el programador, sistema operativo o usuario.

El multicore y multiprocesador es como un restaurante con varias cocinas o una cocina con sitio para varios cocineros y sus cacharros. Cada cual trabaja en un plato distinto sin que unos se estorben a otros

GOBIERNO DE ESPAÑA MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES
Unión Europea Fondo Social Europeo El FSE invierte en tu futuro

Author. Ignacio Pachés Giner

Ejecución especulativa y predicción de saltos.

La ejecución especulativa se sitúa en el contexto en el que se combinan los siguientes elementos:

- Un bucle con una instrucción de salto condicional
- Un procesador superescalar (capaz de ejecutar varias instrucciones simultáneamente)

Opcionalmente:

- Multiprocesador

La idea básica de los procesadores superescalares es ejecutar varias instrucciones a la vez. Esto significa que el procesador se adelanta y empieza a leer una instrucción cuando todavía está ejecutando una muy anterior. Digamos que "unas instrucciones van empezando mientras otras están en curso y otras ya están acabando". Suponiendo que todo funciona bien y el plan es posible, hay un problema serio con las instrucciones de salto condicional. El procesador puede no estar seguro de si saltar o no hasta que no termine totalmente la instrucción anterior. Es decir que no puede adelantar trabajo ejecutando ninguna instrucción porque no sabe si saltar o no saltar y no sabe cierto cuál es la instrucción que vendrá después del salto.

Los procesadores con ejecución especulativa y predicción de salto, son capaces de recordar si ha habido un salto anteriormente y la esperanza de volver a repetir el mismo resultado (ten en cuenta que un bucle se repite muchas veces, pero sólo se sale una vez de él). Por tanto, el procesador puede tomar una decisión probabilísticamente favorable y adelantar la ejecución de las instrucciones existentes después del salto, o las instrucciones a las que se debería saltar; y cuando llegue el momento de evaluar la condición del salto confirmar que su decisión había sido la correcta y proseguir como si nada, o darse cuenta de que no había que haber ejecutado estas instrucciones y desechar los cambios habidos.

Dado que el procesador ha especulado sobre qué instrucciones sería más conveniente ejecutar, se dice que el procesador incorpora "ejecución especulativa"

Procesadores con instrucciones muy largas (VLIW)

Esta técnica es opuesta a la de ejecución especulativa, pero manteniendo la intención de ejecutar varias "operaciones" a la vez. En estos procesadores, las instrucciones ocupan muchos bytes (en todos los computadores modernos, las instrucciones ocupan como mínimo 4 bytes). Son tan largas las instrucciones (hasta 64 bits) que pueden indicar varias cosas a realizar a la vez. No es extraño encontrar una instrucción que indica que hay que hacer dos sumas, una multiplicación, una división etc. todo a la vez. Para codificar todo eso, las instrucciones necesitan muchos dígitos. Es como una receta que indica varios pasos simultáneos en la preparación de un plato; seguramente, ese paso de receta ocupe muchas líneas.

Para poder ejecutar con beneficio estas instrucciones, los procesadores tienen varias ALU y muchos registros. La idea es intentar emplear todas las ALU simultáneamente y así efectuar más cálculos por unidad de tiempo. Estos cálculos han de ser independientes unos de otros, porque se ejecutan a la vez.

GOBIERNO DE ESPAÑA · MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE · SECRETARÍA DE ESTADO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y UNIVERSIDADES · Unión Europea · Fondo Social Europeo · El FSE invierte en tu futuro

A cambio, estos procesadores no son superescalares. No pueden estar haciendo dos instrucciones simultáneamente, ya es bastante que una instrucción haga varias cosas a la vez. Esto tiene una implicación muy alta en la programación para estos procesadores, ya que hay que replantearse la forma de crear las instrucciones y aprovechar esta forma de trabajar por parte del procesador. Hay que cambiar, pues, la manera de programar.

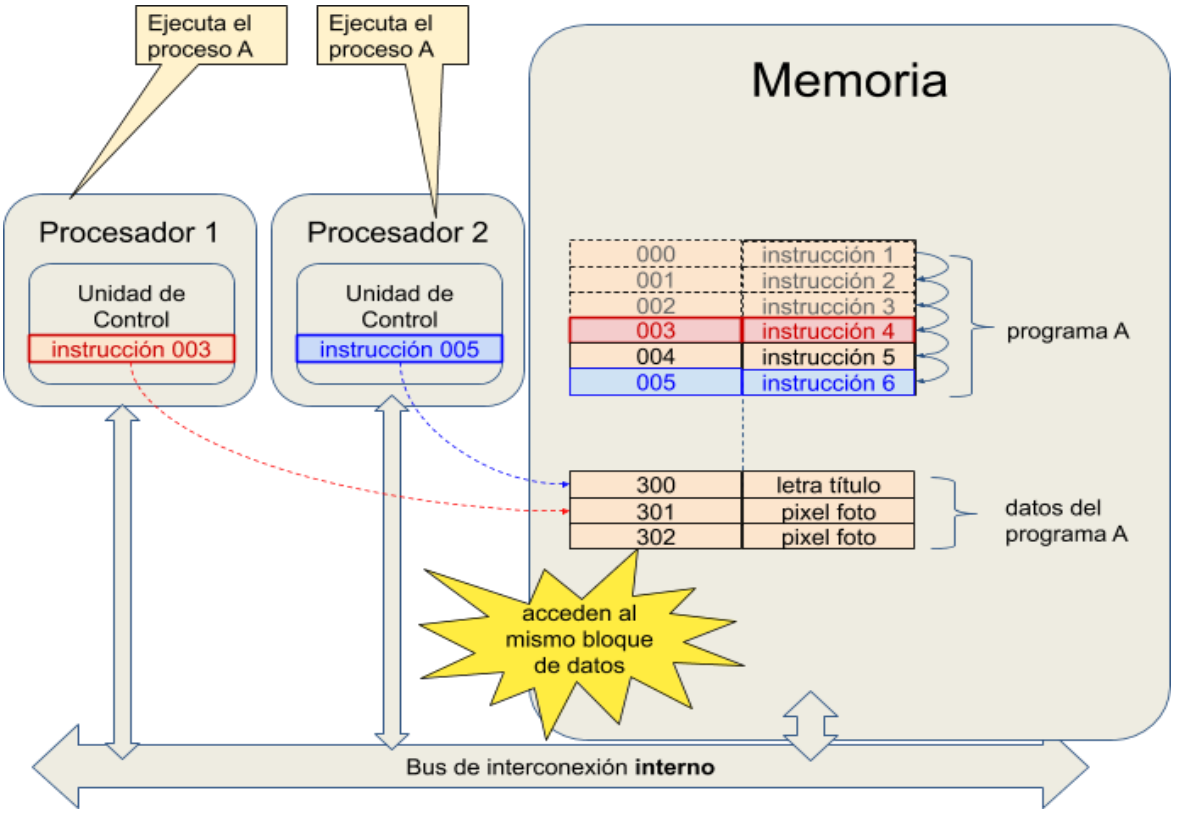
¿Son más eficientes que los procesadores superescalares? Vamos a dejar esa respuesta o discusión. Pero supongamos que no son más eficientes y aun así tienen una gran ventaja sobre los superescalares: son muy simples. La Unidad de control no ha de preocuparse de si puede ejecutar una instrucción junto con la siguiente o no. Aquí, ejecuta una instrucción cada vez y ya está. Menos complejidad en la UC significa menos calor desprendido y menos consumo. Ten en cuenta que tanto superescalares como VLIW, realizan varias operaciones a la vez, la diferencia es que, con los superescalares, además, la Unidad de Control se ha tenido que ocuparse de simultanear de alguna manera las instrucciones, mientras que en los procesadores VLIW, lo ha hecho el programador/compilador en el momento de hacer el programa.

Ejecución Multihilo

La ejecución multihilo no es una técnica para acelerar la ejecución de un computador. Sin embargo, es fácil entenderla por estar relacionada con algunos conceptos vistos aquí.

Para explicarla mejor, vamos a suponer un computador multicore o multiprocesador. Hasta ahora, en esos casos un procesador ejecutaba secuencialmente instrucciones de un proceso. Cada procesador ejecutaba instrucciones de procesos diferentes.

La idea con la ejecución multihilo, es que dos o más procesadores ejecuten instrucciones de un proceso en las cuales se accede a zonas de datos comunes



En el ejemplo del cocinero, se podría asimilar a la situación en la que uno o varios cocineros son capaces de preparar simultáneamente varios pasos de una receta que no necesitan ser elaborados secuencialmente o en orden. Por ejemplo, se debe preparar una salsa y una fritura para después unirlas en un plato. El cocinero podría ir preparando una salsa mientras está friéndose algo. En cierto momento hay dos pasos de receta que podrían estar a la vez en marcha. Si estamos hablando del mismo plato y para el mismo cliente, y un cocinero o dos consiguen hacer los dos pasos al mismo tiempo, entonces se trata de una preparación "multihilo" del plato.

A nivel macroscópico puedes percibir este fenómeno cuando en una página web hay varios anuncios que incluyen animaciones y todas ellas están simultáneamente animadas. Es razonable pensar que cada una es un hilo en ejecución todo dentro del mismo proceso (el navegador)

Gobierno de España · Ministerio de Educación, Cultura y Deporte · Secretaría de Estado de Educación, Formación Profesional y Universidades
Unión Europea · Fondo Social Europeo · El FSE invierte en tu futuro