La **unidad central de procesamiento** (**UCP**, de aquí en adelante, aunque también es común referirse a ella como CPU, por sus siglas en inglés: *Central Processing Unit*) contiene a la unidad aritmética y lógica (hace los cálculos) y a la unidad de control (define su orden y secuencia).

La función de la UCP es clara: ejecutar instrucciones. Pero para ello necesariamente deben cumplirse estas condiciones:

- que las instrucciones sean entendibles por la UCP
- que estén almacenadas en la memoria.

Para cumplir con la primera condición se vuelve imprescindible codificarlas de alguna forma, mediante números, para entonces construir circuitos electrónicos (Babbage fracasó en su intento de hacerlos con engranes) que se activen cuando reciban cada código numérico y efectúen la acción u operación codificada. Esto se hará más adelante.

Para la segunda condición, se abordará el problema del almacenamiento de números definiéndolo precisamente como la función de la memoria.

Para nuestros propósitos, la **memoria** será un conjunto de **celdas** (o casillas electrónicas) con las siguientes características:

- cada celda puede contener un (y sólo un) **valor** numérico
- cada celda tiene la propiedad de ser "direccionable", es decir, se puede distinguir una de otra por medio de un número unívoco llamado su **dirección**.

Esto implica que las celdas de la memoria deben estar organizadas para facilitar la localización de cualquiera de ellas con un esfuerzo mínimo. La forma más sencilla de lograrlo es organizándolas en forma de un conjunto numerado secuencialmente (también llamado **arreglo** o **vector**), para entonces referirse a las celdas por medio de su dirección. Se usará un **apuntador** para dirigirse a alguna celda (es decir, para especificar su dirección).

Así, el conjunto de celdas de memoria se puede ver como muestra el siguiente arreglo:

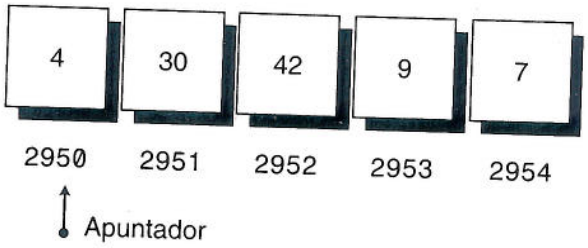

Figura 21 Arreglo de celdas de memoria

Cada celda tiene una dirección. Por ejemplo, la celda 2951 contiene un 30. Debe quedar clara la existencia de dos conceptos independientes: una cosa es la dirección de una celda y otra es su valor, aunque casualmente pudieran coincidir. Se dispone ya de una manera de almacenar (y recuperar) valores por medio de una dirección unívoca, como se verá a continuación, cuando se definan dos operaciones elementales que el procesador puede efectuar sobre la memoria: **leer** el contenido de una celda, y **escribir** un valor en una celda.

Operaciones sobre la memoria

Si se supone a la memoria de una computadora como una especie de almacén electrónico que funciona en forma autónoma del procesador, éstos deberán ser los pasos necesarios para efectuar las dos operaciones primitivas.

Para **leer**:

A) Decidir cuál celda se va a leer y proporcionar su dirección.
B) Esperar un tiempo fijo para que los circuitos de la memoria traigan el valor depositado en esa celda y lo dejen en otro registro especial (la celda no pierde ese valor; sólo se trae una copia del dato y no el dato mismo).
C) Recoger el dato y dar por terminada la operación de lectura.

Y para **escribir**:

A) Proporcionar a la memoria el dato que se desea depositar en una celda.
B) Proporcionar la dirección de la celda sobre la que se desea hacer la escritura del dato.
C) Esperar un tiempo fijo para que los circuitos de la memoria depositen el dato en la celda designada, para dar por terminada la operación de escritura. (Si la celda en cuestión tenía ya un valor, éste se pierde, pues es reemplazado por el nuevo.)

Necesidad de la codificación

Ahora debemos enfrentar el problema previamente expresado: cómo almacenar las instrucciones en la memoria; es decir, encontrar una forma de hacer caber las instrucciones en las celdas. Esto lleva necesariamente al concepto de **codificación**.

En efecto, si en las celdas de memoria sólo caben números, entonces habrá que convertir las instrucciones en números para poder emplearlas. Para codificarlas se debe considerar cuántas y cuáles habrá disponibles, así como el esquema de codificación por emplear.

El primer factor depende fundamentalmente de la capacidad de la unidad de control del procesador central para hacer operaciones; cuanto más compleja —y costosa— sea la unidad central de procesamiento, tanto mayor será el número de instrucciones diferentes que podrá efectuar. Igualmente, debe encontrarse un código adecuado para que a cada instrucción definida corresponda un, y sólo un, valor numérico. Para este propósito usaremos una especie de "diccionario electrónico" (que forma parte de la unidad de control: es lo que Babbage no pudo realizar con la tecnología mecánica de su tiempo), y que especificará, por ejemplo, los siguientes códigos numéricos. (Además, para nuestros fines pedagógicos escribimos a la izquierda una columna con el nombre de la instrucción, pero debe quedar claro que los circuitos de la máquina sólo están diseñados para reconocer los códigos, no sus nombres.)

Nombre de la instrucción	Código interno
SUMA	30
RESTA	33
.	.
.	.
.	.

De aquí en adelante se empleará el nombre **lenguaje de máquina** para referirse al código manejado por la unidad central de procesamiento de la computadora; los circuitos electrónicos de la unidad de control de la computadora reconocen y ejecutan precisamente las instrucciones codificadas en ese lenguaje de máquina.

Un primer programa

Ahora es posible escribir un primer programa completo, usando el modelo recién descrito. Se continuará con el problema de sumar 5 + 7.

Primera consideración: se requieren tres casillas, dos para los datos (5 y 7) y una para depositar el resultado. Se escogen las casillas 20, 21 y 22. (No hay ninguna razón especial para haberlas escogido; para nuestros fines, tres casillas cualesquiera son adecuadas.)

Segunda consideración: hay que definir con detalle las operaciones por efectuar y su orden, así como obtener una codificación adecuada (o sea, escribirlas en lenguaje de máquina).

Tercera consideración: hay que introducir todos los datos (e instrucciones) en la memoria.

Trabajando con nuestro programa de la página 77, se detecta la necesidad de inventar varias operaciones de la máquina. Se requiere, por lo pronto, una instrucción para llevar el contenido de una celda al **acumulador** (que es una celda especial, o **registro**, contenido

en la UCP); otra para hacer la suma, y otra para devolver el contenido del acumulador a una celda de la memoria.

Longitud de las instrucciones

La forma de la instrucción para llevar el contenido de una celda al acumulador es:

```
CARGA <dirección>
```

donde CARGA es el nombre dado a la instrucción, y la <dirección> indica la celda de memoria cuyo valor se desea llevar al acumulador. Cabe aclarar que CARGA es el **nombre simbólico** (o mnemónico) de la instrucción empleado para que las instrucciones sean legibles por nosotros, pero primero fue necesario asignarle un código numérico interno. Sin importar ahora cuál sea éste, ocupará el contenido de una celda de la memoria. De la misma manera, la dirección será un número que ocupará un lugar en otra celda. Esto significa que la instrucción CARGA ocupará dos celdas en la memoria: una para el código de la operación y la otra para la dirección a la que hace referencia. Es decir, su longitud es dos.

En términos generales, habrá instrucciones que ocupen una, dos y hasta tres o más celdas de memoria.

Las otras instrucciones requeridas son:

`GUARDA <dirección>`	Deposita el valor del acumulador en una celda de la memoria (ésta es la inversa de la anterior), y
`SUMA <dirección>`	Suma al acumulador el contenido de la celda de memoria descrita por la dirección.

Así que las instrucciones de nuestra máquina serán por lo pronto las siguientes:

Nombre de la instrucción	Código interno	Longitud de la instrucción
CARGA	20	2
GUARDA	02	2
RESTA	33	2
SUMA	30	2

(Los códigos internos para el lenguaje de máquina que se escogieron son arbitrarios aunque sí debe tenerse cuidado de usarlos consistentemente.)

Escribiremos el programa en forma tabular. En la parte izquierda del renglón se colocará el nombre simbólico de la instrucción, seguido de la dirección a la que haga referencia (si se da el caso); luego se describe brevemente el renglón (si se considera necesario) y, por último, se escribe su equivalente en el código interno que "entiende" el diccionario electrónico de la computadora.

Para lo que sigue, se supone que la celda 20 contiene un 5 y la celda 21 un 7, sin preocuparse por ahora de cómo se colocaron allí esos números.

He aquí el programa para sumar 5 + 7.

Instrucción	Dirección	Comentarios	Código
CARGA	20	Se coloca el primer número en el acumulador.	2020
SUMA	21	Se efectúa la suma.	3021
GUARDA	22	El resultado queda en la casilla 22.	0222
ALTO	--	Detiene la ejecución	70

De este simplísimo programa se puede aprender varias cosas. Fue necesario inventar una nueva instrucción (ALTO) para lograr que la secuencia —cuando se ejecute— llegue a un fin. Obsérvese que esta nueva instrucción ocupa una sola casilla de memoria, pues no es necesario que haga referencia a alguna dirección.

Otra cuestión importante es la aparición de dos programas: uno escrito en lenguaje simbólico y mnemónico (más fácil de reconocer para nosotros ya que es cercano al español), y otro —a la derecha— que está descrito en código numérico (el único que reconoce la computadora).

Se llamará **programa fuente** al primero y **programa objeto** al segundo. Esto es, el programa fuente es aquel que está escrito en un lenguaje similar al nuestro (pero inaccesible para la computadora), mientras que el programa objeto ya está traducido al código que la máquina reconoce. En este caso fue la misma persona quien escribió ambos programas; por lo general será tarea del programador escribir el programa fuente, y la propia computadora lo traducirá a lenguaje objeto.

El programa objeto, entonces, es

```
2020  3021  0222  70
```

que resulta por completo ininteligible para nosotros†.

Otro aspecto fundamental es entender que este programa suma cualquier par de números, siempre y cuando residan en las casillas 20 y 21. Esto es realmente importante, pues significa, ni más ni menos, que es una especie de "programa universal" para sumar dos números, sin importar cuáles sean. Claro que esto no resulta impresionante por ahora, pero si se piensa en un programa universal para escoger la mejor ruta en un mapa, u otro para resolver ecuaciones diferenciales en forma numérica, se apreciarán las ventajas de esta nueva herramienta.

Resta tan sólo introducir el programa objeto en la memoria de la computadora, para que pueda ejecutarse luego. Aquí es crucial elegir las casillas de la memoria que se utilizarán para almacenar el programa; esto es, en qué sección de la memoria se va a **cargar** el programa.

Se decidió hacerlo a partir de la celda 10 (se puede cargar a partir de cualquiera que esté desocupada, siempre que haya suficientes celdas secuenciales vacías).

Una vez cargado, el programa objeto se verá así:

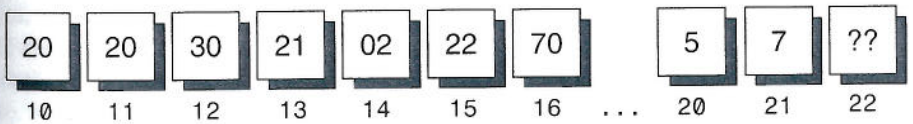

Figura 22 Programa objeto cargado en memoria

En cada celda hay un solo número. La celda 22 contiene uno no especificado todavía —el resultado de la suma—, que se obtendrá una vez ejecutado el programa.

En las dos primeras celdas hay un número 20, pero el inicial representa el código de la instrucción CARGA, mientras que el mismo 20 representa, en la segunda celda, la dirección 20. Obsérvese cómo una celda puede contener un mismo número que significará dos cosas diferentes, dependiendo del orden en que aparezcan con respecto al inicio.

Cuando se ha cargado el programa objeto a partir de la celda 10 de la memoria, debe encontrarse un procedimiento para lograr que la computadora comience a ejecutarlo y poder así obtener los resultados deseados.

Haremos a continuación un paréntesis para describir con cierto detalle cómo funciona la unidad de control del procesador central de la computadora.

(†) ...porque no somos máquinas.

La unidad de control

La función principal de la unidad de control de la UCP es dirigir la secuencia de pasos de modo que la computadora lleve a cabo un ciclo completo de ejecución de una instrucción, y hacer esto con todas las instrucciones de que conste el programa. Los pasos para ejecutar una instrucción cualquiera son los siguientes:

El ciclo de ejecución de la UCP

I. Ir a la memoria y extraer el código de la siguiente instrucción (que estará en la siguiente celda de memoria por leer). Esto se logra mediante un registro apuntador (que forma parte de los circuitos electrónicos de la unidad de control) conocido como **contador de programa** (CP). Este paso se llama ciclo de *fetch* en el uso computacional (*to fetch* significa traer, ir por).

II. Decodificar la instrucción recién leída (determinar de cuál se trata).

III. Ejecutar la instrucción.

IV. Prepararse para leer la siguiente casilla de memoria (es decir, actualizar el CP para que apunte a la celda que contiene la siguiente instrucción), y volver al paso I para continuar.

La unidad de control ejecutará este ciclo de cuatro "instrucciones alambradas" a una enorme velocidad†. Se les llama así porque no residen en memoria, ni fueron escritas por ningún programador, sino que la máquina las ejecuta directamente por medios electrónicos, y lo hará mientras esté funcionando (mientras esté encendida). La ejecución de estos cuatro pasos (que forman un ciclo repetitivo) en una computadora es a razón de millones de veces por segundo.

Es decir, la unidad de control es una realización electrónica del siguiente modelo digital:

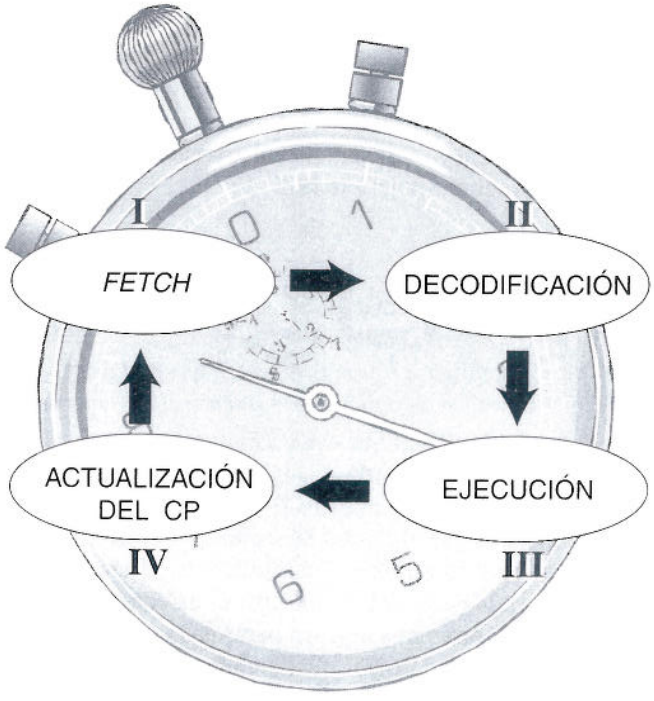

Figura 23 Modelo digital de la unidad de control

(†) Como se mencionó en el capítulo anterior, ésta es la diferencia principal entre **hardware** y **software**. El primer término denota todo lo referente a los circuitos electrónicos de una computadora, mientras que el segundo hace referencia a los programas, que no son parte física de la máquina, sino que residen en la memoria.

Se ha definido ya el modelo de von Neumann, y ahora se pondrá a funcionar sobre nuestro pequeño programa de ejemplo (que suponemos ya cargado en la memoria).

Pasos en la ejecución de un programa

Se describirán todos los pasos con detalle por única vez, para que el lector pueda estudiarlos con detenimiento hasta estar seguro de haberlos comprendido. Pedimos no seguir con la lectura si no cumple con esto.

0. En virtud de que el programa comienza a partir de la celda número 10, se debe indicar a la unidad de control que esa celda contiene la primera instrucción. Para ello hay que colocar el número 10 en el registro contador de programa (es decir, apuntar a esa celda). Esto se representará con CP ←10.

   (Obsérvese que este paso es externo, esto es, no forma parte del programa, sino que se tiene que hacer "desde afuera", para iniciar la operación de la computadora. Por eso se etiquetó como número cero.)

1. La unidad de control ejecutará el paso I e irá a la casilla 10 para leer su contenido, que es 20.
2. La unidad de control ejecuta el paso II, con lo que decodifica el 20 recién leído y determina que se trata de una operación CARGA. En este momento sucede algo de primordial importancia: como la instrucción 20 tiene una longitud de dos celdas, una para el código (20) y otra (la siguiente) para la dirección de la celda cuyo valor se cargará en el acumulador (que en este caso —de casualidad— también es 20), la máquina deberá ajustar internamente el valor del contador de programa para que apunte a esa celda siguiente (*i. e.*, la número 11).
3. La unidad de control ejecuta el paso III, con lo que efectivamente realizará la operación de carga. Para esto, la computadora debe ir a la celda 11 y extraer su contenido, pero ahora ya no lo considerará como instrucción, sino como dirección, por lo cual irá a la celda 20 para extraer el valor que contenga (que es 5).

En este momento hay que tener cuidado para que no haya confusión: el primer 20 (el de la celda 10) es la instrucción CARGA; el segundo 20 (el de la celda 11) es la dirección de la celda de memoria cuyo valor se desea cargar en el acumulador. Esta instrucción completa, 20 20, puede leerse de la siguiente manera: "cargar el acumulador con el valor que esté contenido en la celda cuya dirección aparece a la derecha de donde se está leyendo ahora". Aunque parece innecesariamente elaborado, es imprescindible tener muy claro esto, porque aquí reside la potencia del modelo de von Neumann.

4. La unidad de control ejecuta el paso IV, para luego reiniciar todo el ciclo. Obsérvese que se trata de una repetición ilimitada, que sólo terminará cuando se ejecute la instrucción ALTO. Por lo pronto, el contador de programa se hace igual a 12; esto es, apunta a la celda número 12. Internamente, los circuitos de la unidad de control le añadieron la longitud de la instrucción recién ejecutada (que midió 2).
5. Se ejecuta (por segunda vez) el paso I de la unidad de control. Como CP = 12, se leerá esa celda, que contiene un 30.
6. Se decodifica esa instrucción, que es SUMA, por lo que el CP se prepara para apuntar a la siguiente celda. (Recuérdese que la instrucción SUMA ocupa dos celdas: una para su código de operación y otra para la dirección de la celda cuyo contenido se sumará al acumulador.)
7. Se ejecuta la instrucción 30, con lo que se añade al acumulador el contenido de la celda 21 (la dirección 21 reside en la celda 13, que está inmediatamente a la derecha de la que se está ejecutando). Ahora el acumulador contendrá un 12 (o sea, 5 + 7).

8. El CP se actualiza para apuntar a la celda 14, en la cual (y no es casualidad, sino resultado del modelo) reside el código de la siguiente instrucción. Nuevamente en forma interna (paso IV) los circuitos de la unidad de control suman al contador de programa la longitud de la instrucción recién ejecutada.
9. Se lee la celda 14 y se extrae su contenido: 02.
10. Se decodifica la instrucción, (GUARDA), por lo que se examina la siguiente celda, que contiene la dirección de la celda en donde se guardará el contenido del acumulador.
11. Al ejecutarse esta instrucción se deposita el valor del acumulador (12) en la celda número 22, o sea, se deja el resultado de la suma en la celda que previamente se había separado para tal fin.
12. La unidad de control regresa al paso I, no sin antes actualizar el contador de programa con la longitud de la instrucción recién ejecutada, y ahora apunta a la celda 16, que es donde reside la siguiente instrucción.
13. Se lee la celda 16 y se extrae su contenido: 70.
14. Se decodifica esta instrucción, que es ALTO. No se examina el dato de la siguiente celda porque la instrucción 70 tiene una longitud de uno.
15. Se ejecuta esta instrucción, lo que detiene a la unidad de control y a la máquina. De esta manera se rompe el ciclo de los cuatro pasos.

El compromiso había sido que el lector seguiría estos dieciséis pasos con cuidado (con ayuda de la figura 22), por lo cual ya habrá aprendido varias cosas sobre el lenguaje básico de las computadoras, entre las que sobresalen las siguientes:

- Dado el contenido de una celda, la computadora no puede distinguir si se trata de una instrucción, un dato o una dirección.
- Debido a lo anterior, es responsabilidad de quien maneja la máquina indicarle cuál es la celda en donde comienza el programa (esto se hizo por medio del paso 0, que se describió como externo al programa). En algún capítulo posterior del libro se verá cómo la propia computadora, por medio del sistema operativo, será la que se encargue de esta tarea.
- Una vez que el contador de programa apunta a la celda que contiene la primera instrucción, el resto del proceso ocurre de manera automática e invisible para el usuario. Esto se debe a los ajustes internos que se hacen al CP (en cada paso IV) que, a su vez, dependen de la longitud de la instrucción que se acaba de ejecutar.

Ejemplos de programas en lenguaje de máquina

Por lo pronto, nuestra máquina de papel reconoce algunas pocas instrucciones (pues los circuitos electrónicos de su unidad de control así lo especifican), y sería deseable incrementar la cantidad y tipo de ellas, para disponer de más poder de cómputo.

En principio, la tarea es sencilla: basta con idear alguna instrucción y asignarle un código interno. Pero debe quedar claro que, en la vida real, cada nueva instrucción implica necesariamente la incorporación de nuevos circuitos especializados (y miles de nuevos microtransistores), por lo que existe un límite en la cantidad y complejidad de las instrucciones que un procesador puede realizar.

Evaluación de una fórmula

Escribiremos a continuación un programa para sumar tres números cualesquiera y restar otro número al total anterior. Es decir, se evaluará la fórmula R = A + B + C - D (concediendo que tal cosa le pueda ser de utilidad a alguien). Supondremos que los cuatro nú-

meros ya existen en memoria, en las celdas 770 a 773, y que el resultado va a quedar en la celda 774:

```
CARGA    770
SUMA     771
SUMA     772
RESTA    773
GUARDA   774
ALTO
```

La codificación de este programa fuente en lenguaje de nuestra máquina da como resultado el siguiente programa objeto equivalente:

```
20  770  30  771  30  772  33  773  02  774  70
```

Si se deseara posteriormente cargarlo a la memoria para ejecutarlo, habrá que seleccionar once celdas vacías secuenciales, que no incluyan de la 770 a la 774, porque ésas están reservadas para los datos.

NOTA: El resto de esta sección explora con mayor detalle algunas características del lenguaje de máquina, y podría incluso dejarse para una segunda lectura. Se trata del área específica de conocimiento AC19 de la subárea 3.3.1 (Arquitecturas y formas de procesamiento) de los Modelos Curriculares.

Modos de direccionamiento: programa para obtener un promedio

Analizaremos ahora cómo escribir un programa para obtener el promedio de cuatro números previamente almacenados en la memoria. Se considerarán únicamente operaciones sobre números enteros, por lo que el promedio será también un número entero. Para obtener un promedio hay que hacer una suma y luego una división. Pero nuestra máquina aún no puede dividir, por lo cual prestamente inventamos (y comenzaremos a analizar) la instrucción para lograrlo, que tomará el número que en ese momento esté en el acumulador y lo dividirá entre otro.

En general, las instrucciones aritméticas que el procesador puede realizar se efectúan sin punto decimal; es decir, sólo con enteros. Los circuitos para manejo de operaciones con punto flotante son tan complicados que no todos los procesadores los incluyen; en capítulos posteriores se habla más sobre esto.

Pero, ¿entre cuál número se hará la división de lo que contenga el acumulador? Existen dos posibilidades: entre el que resida en una celda de memoria cuya dirección se especifique, o bien entre un valor que a continuación se indique.

Es decir, puede haber dos tipos de instrucciones de división, dependiendo del modo en que hagan el acceso a su operando.

Modos de direccionamiento

Una tiene la siguiente especificación:

`DIV <dirección>`	Divide lo que contenga el acumulador entre el contenido de la celda de memoria descrita por la dirección.

Y la otra efectuará la división ya no entre el número contenido en la celda cuya dirección se especifica, sino simplemente entre el número que se indique inmediatamente a continuación:

`DIV-i <dato>`	Divide lo que contenga el acumulador entre el número que viene a continuación.

El nombre simbólico de la segunda variante contiene una letra "i", porque se trata de una división en modo inmediato. Ambas instrucciones tienen longitud 2: una celda para el código y la otra para la dirección del dato (en el primer caso), o para el dato mismo (en el segundo).

Algunas instrucciones tienen variantes de direccionamiento en modo normal o en modo inmediato (y otros más), dependiendo, como se acaba de ver en el ejemplo, de si el dato con el que operan está contenido en una celda de memoria cuya dirección se indica, o bien si se encuentra inmediatamente a continuación. Podría decirse que una operación en modo inmediato es menos general que la misma en modo normal, como se verá enseguida.

Por ejemplo, las instrucciones

```
CARGA     7428
SUMA      7429
GUARDA    7430
```

suman al número contenido en la celda 7428 el número que está en la celda 7429 y dejan el resultado en la celda 7430. Es decir, realizan la operación R = A + B , llamando "A", "B" y "R" a los contenidos de esas tres celdas, independientemente de cuáles sean sus valores.

Pero la variante

```
CARGA-i   18
SUMA-i    200
GUARDA    7430
```

únicamente hace la operación R = 18 + 200, con lo que siempre dejará el valor 218 en la celda que llamamos "R" (la 7430).

En general, las instrucciones en modo inmediato son menos interesantes que las normales, porque están "comprometidas" con los valores que también forman parte del programa, mientras que las otras trabajan sobre las direcciones de los valores, por lo que estos últimos no forman parte del programa, sino que pueden ser datos independientes.

Nuevas instrucciones

Podemos entonces suponer que el diccionario electrónico de nuestra máquina ha crecido (y también —claro— la complejidad de sus circuitos). Ahora se ve así, con nuevas instrucciones ya codificadas y alambradas:

Nombre de la instrucción	Código interno	Longitud de la instrucción
ALTO	70	1
CARGA	20	2
CARGA-i	21	2
DIV	38	2
DIV-i	39	2
GUARDA	02	2
MULT	36	2
MULT-i	37	2
RESTA	33	2
RESTA-i	34	2
SUMA	30	2
SUMA-i	31	2

Obsérvese que se añadió un CARGA en modo inmediato, para poder decir, por ejemplo,

```
CARGA-i <valor>
GUARDA <dirección>
```

y dejar un cierto valor preespecificado en alguna celda de memoria, para cubrir alguna necesidad posterior.