Tema 1:Introducción

Introducción a la programación I

Ana Isabel Sierra de las Heras

Marcos Novalbos

Francisco Javier Garcia Algarra

Rodrigo Alonso Solaguren-Beascoa

Alfonso Castro Escudero

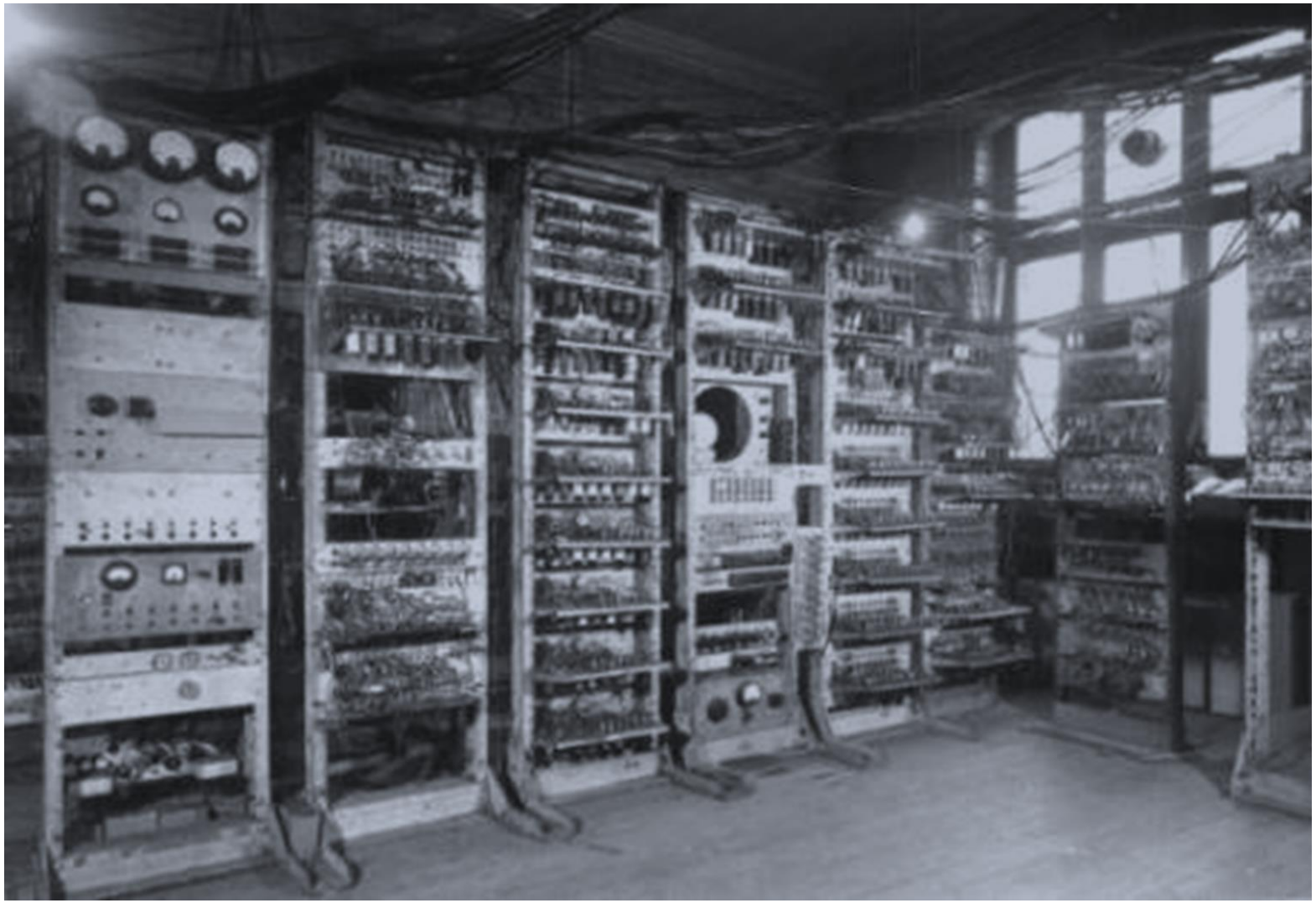
“Es más fácil viajar en un avión, incluso pilotarlo, que entender por qué puede volar“

John Von Neumann

Historia de la Informática

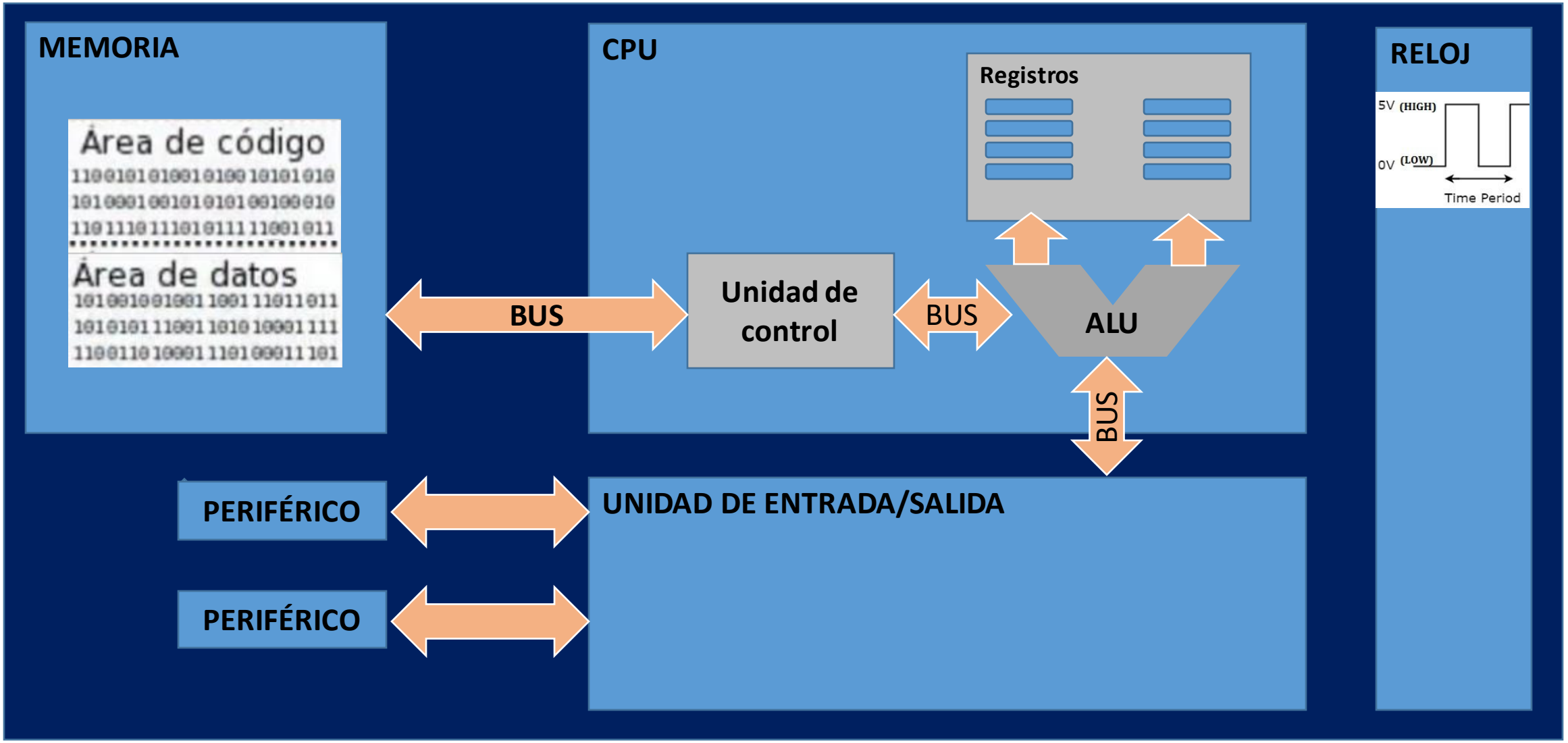
Arquitectura Von Neumann. Definición

- La arquitectura de Von Neumman es un diseño que usa una memoria para almacenar instrucciones y datos.
- Los elementos que la componen son:
  - CPU
    - ALU
    - REGISTROS
    - UNIDAD DE CONTROL
  - MEMORIA
  - PERIFÉRICOS
  - UNIDAD DE ENTRADA Y SALIDA
  - REJOJ
- En la actualidad las computadoras se basan en el modelo de Von Neumann



Historia de la Informática

Arquitectura Von Neumann.



Hay 10 tipos de personas, las que entienden binario y las que no...

- El sistema binario es un sistema de numeración de base 2, eso implica que únicamente trabaja con 0 y 1.

Binario	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Binario	Decimal
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Ejemplo:

$11=1*2^3 + 0*2^2 + 1*2^1 + 1*2^0$

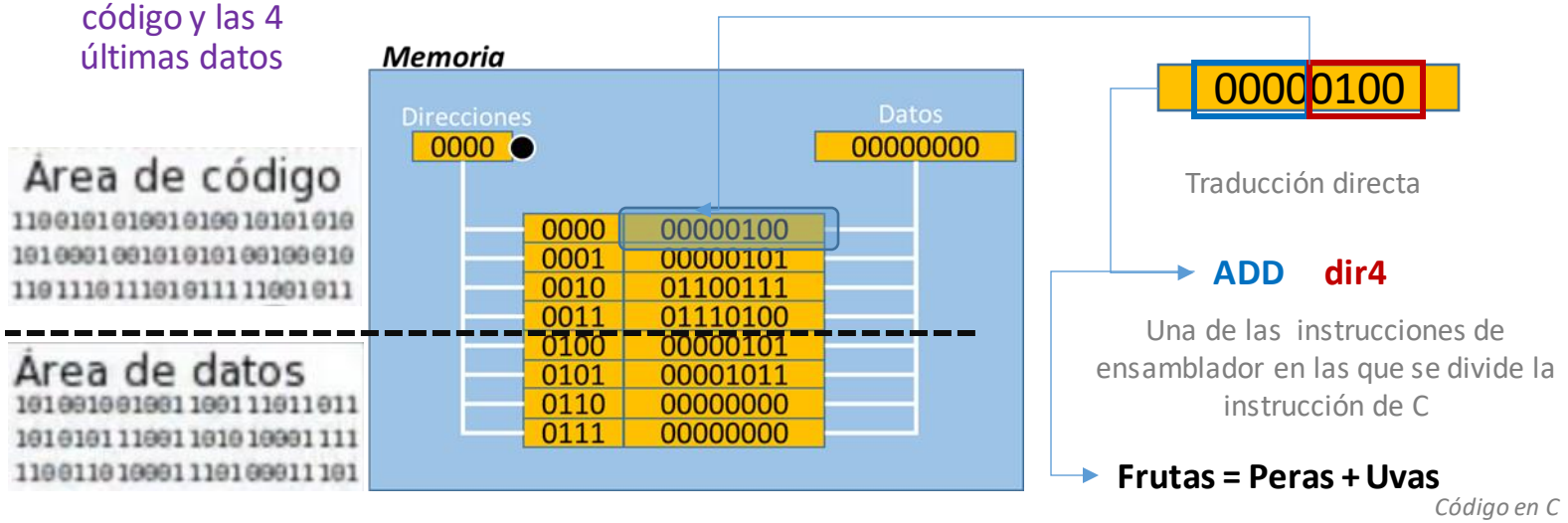
Historia de la Informática

Arquitectura Von Neumann. Ejemplo de funcionamiento

CODIFICACIÓN DE LA INFORMACIÓN

- Los circuitos digitales de los que están formados los elementos de un ordenador trabajan con señales que sólo toman dos valores: encendido-apagado o tensión alta-tensión baja, a estos dos estados diferenciados se les asignan los valores binarios 0 y 1. Por tanto, dentro de un ordenador, todo discurre en forma de dígitos binarios o bits.
- El proceso de establecer la correspondencia entre las magnitudes que conocemos del mundo real y las magnitudes binarias (0s y 1s) con los que trabaja el ordenador se le conoce como "**codificación**".

Las 4 primeras posiciones de memoria son código y las 4 últimas datos



¿Qué significan los mismos valores en un registro de datos?

00000100

Si la dirección de memoria es un número entero almacena el número 4

Introducción

Arquitectura Von Neumann. Ejemplo de funcionamiento

SUMAR 5 + 11
y el resultado
dejarlo en una
posición de
memoria

Las instrucciones en un pseudolenguaje ensamblador para este programa sería:

```
ADD    dir4
ADD    dir5
STORE dir7 Acumulador
```

Historia de la Informática

Arquitectura Von Neumann. Ejemplo de funcionamiento

Fase de búsqueda de la instrucción

La unidad de control envía una micro-orden para transferir el contenido del contador de programa al registro de direcciones

El contador de programa aumenta en uno, por lo que su contenido será la dirección de la siguiente instrucción a ejecutar

Se selecciona la posición de memoria que indica el **registro de direcciones** y se realiza una lectura en la memoria

Se deposita en el **registro de datos** la instrucción a ejecutar

Se realiza el traslado de la información contenida en el registro de datos a**l registro de instrucciones**, donde se almacenará

El **decodificador** procede a la interpretación de la instrucción que serán los 4 primeros bits, es decir, interpreta el código de operación

Fase de ejecución de la instrucción

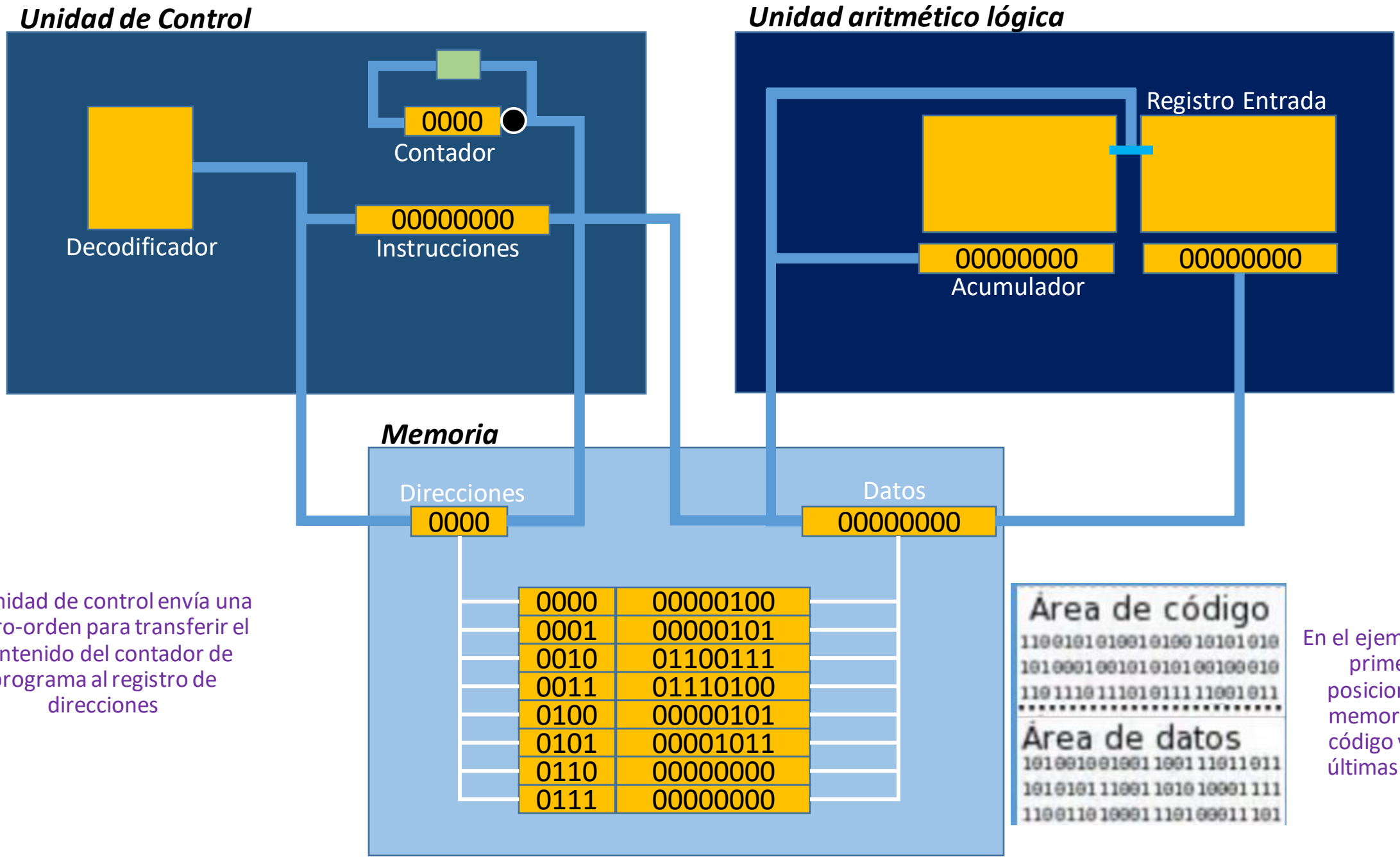
El registro de instrucciones envía los 4 últimos bits al Registro de direcciones

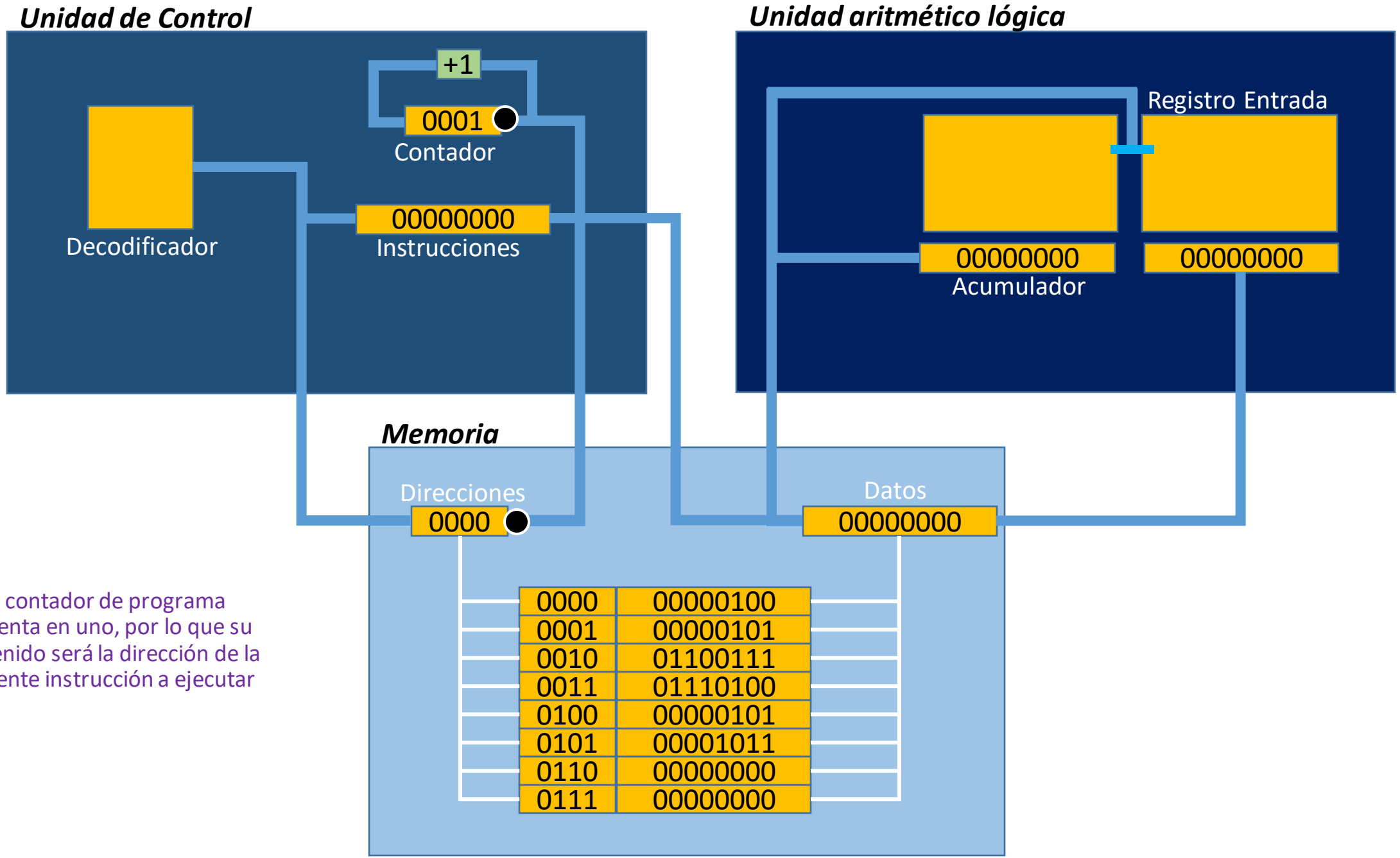
El registro de direcciones busca en la menoría la celda correspondiente y procede a la lectura del dato

La información es enviada al Registro de datos

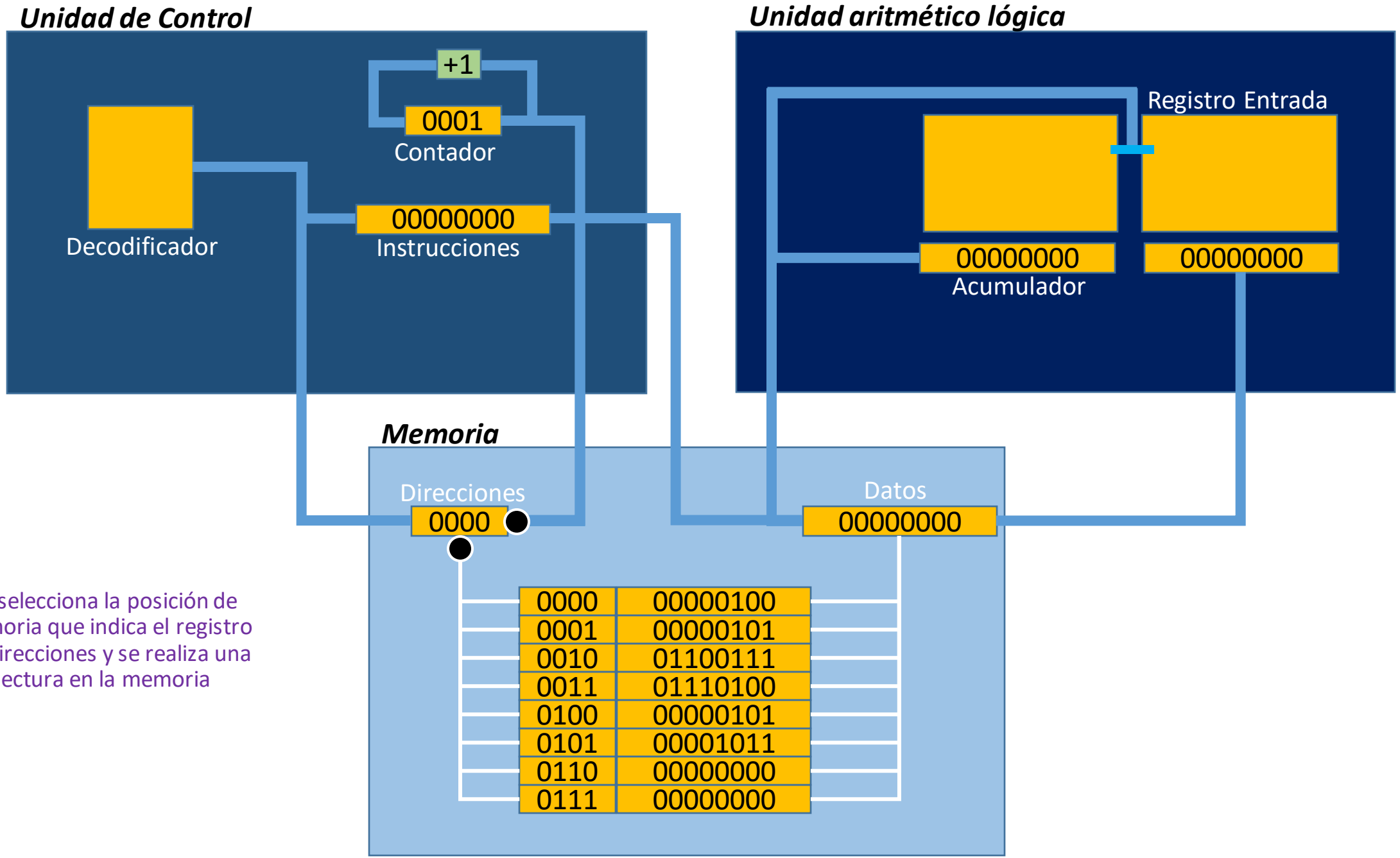
El registro de datos envía la información al registro de entrada

El circuito operacional realiza la operación con el registro acumulador y el registro de entrada y lo almacena de nuevo en el Registro Acumulador

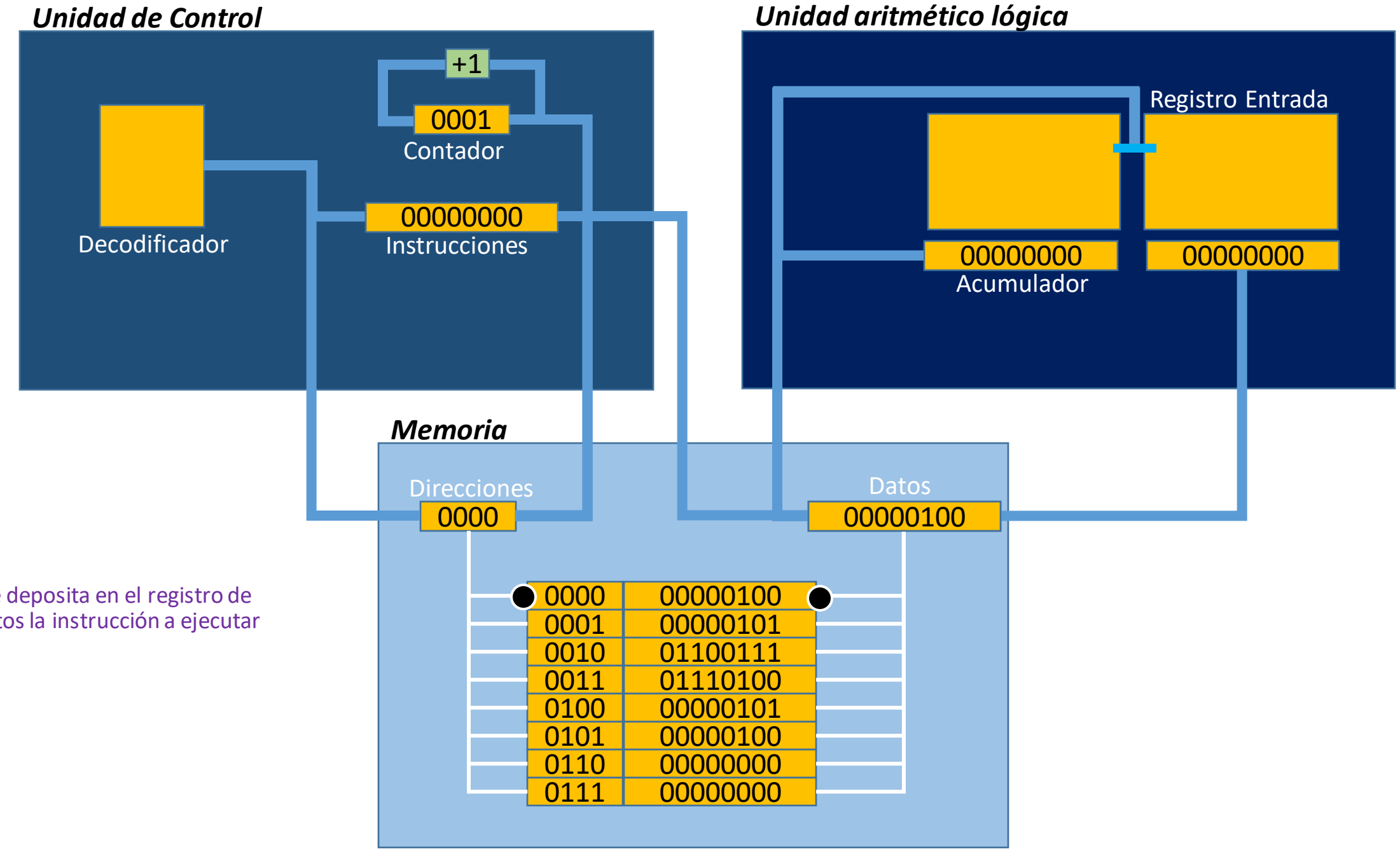

Unidad de Control
Unidad aritmético lógica
0000
Contador
Decodificador
00000000
Instrucciones
Registro Entrada
00000000
00000000
Acumulador
Memoria
Direcciones
0000
Datos
00000000
0000 00000100
0001 00000101
0010 01100111
0011 01110100
0100 00000101
0101 00001011
0110 00000000
0111 00000000
Área de código
Área de datos
La unidad de control envía una micro-orden para transferir el contenido del contador de programa al registro de direcciones
En el ejemplo las 4 primeras posiciones de memoria son código y las 4 últimas datos



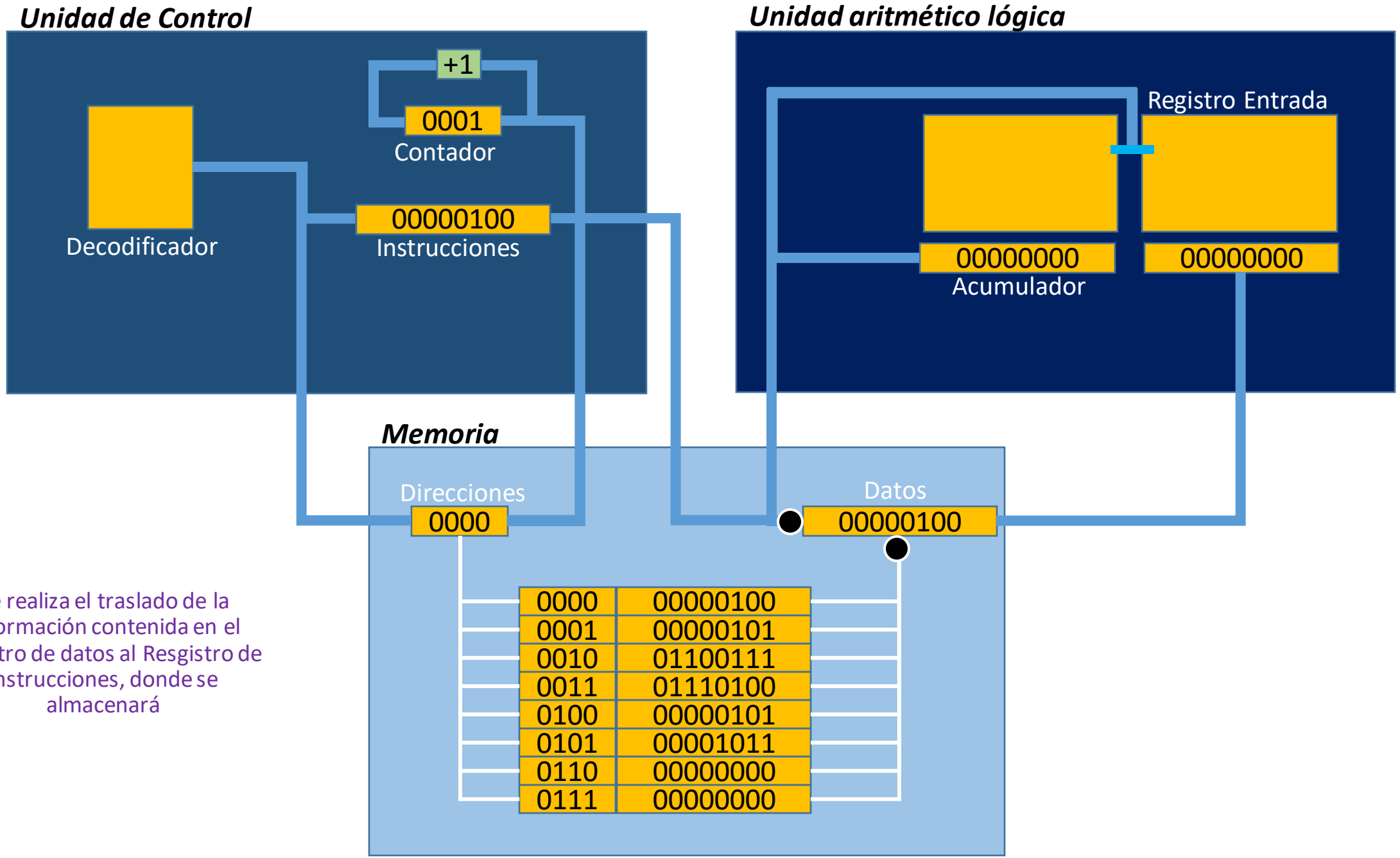
El contador de programa aumenta en uno, por lo que su contenido será la dirección de la siguiente instrucción a ejecutar



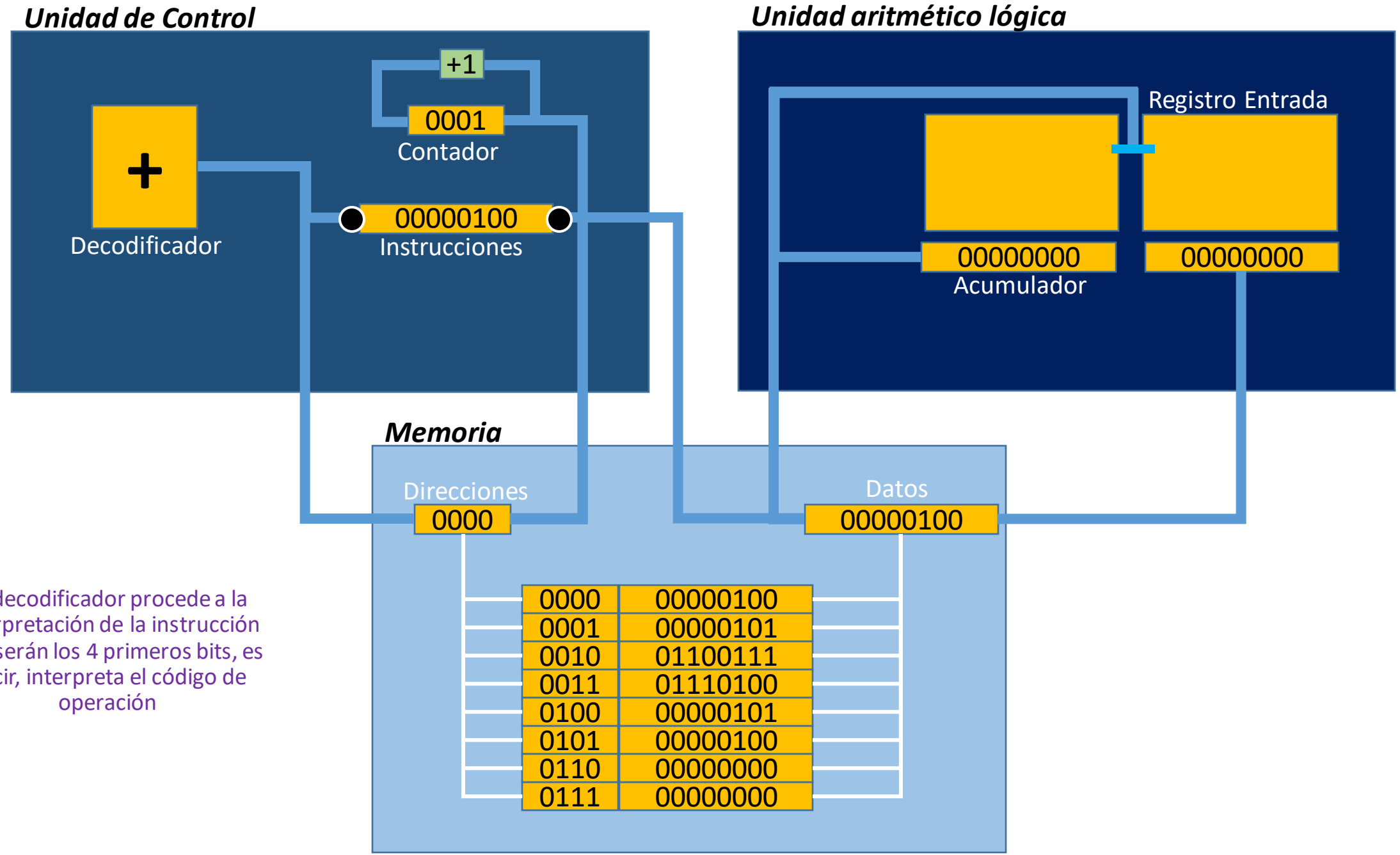
Se selecciona la posición de memoria que indica el registro de direcciones y se realiza una lectura en la memoria



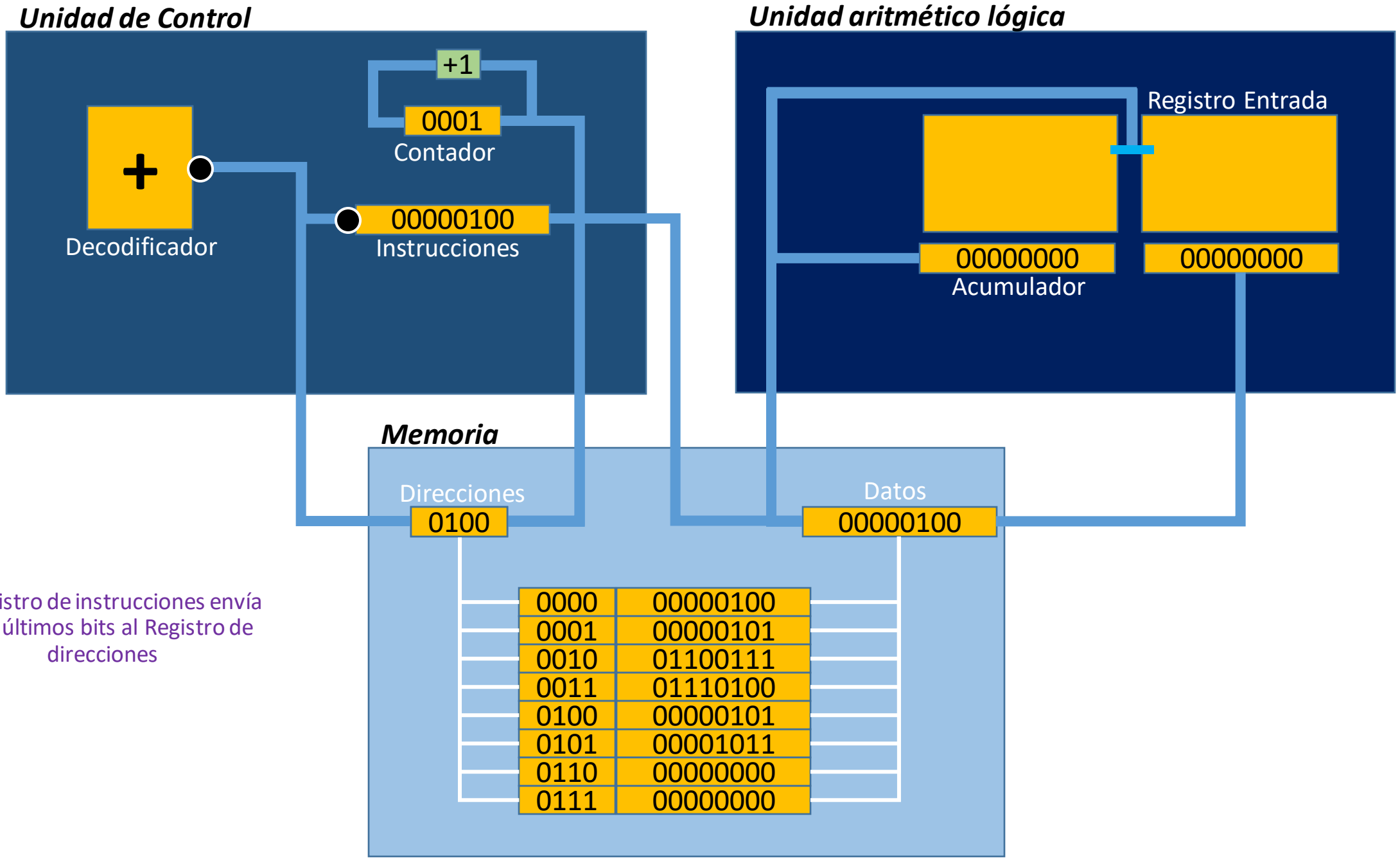
Se deposita en el registro de datos la instrucción a ejecutar



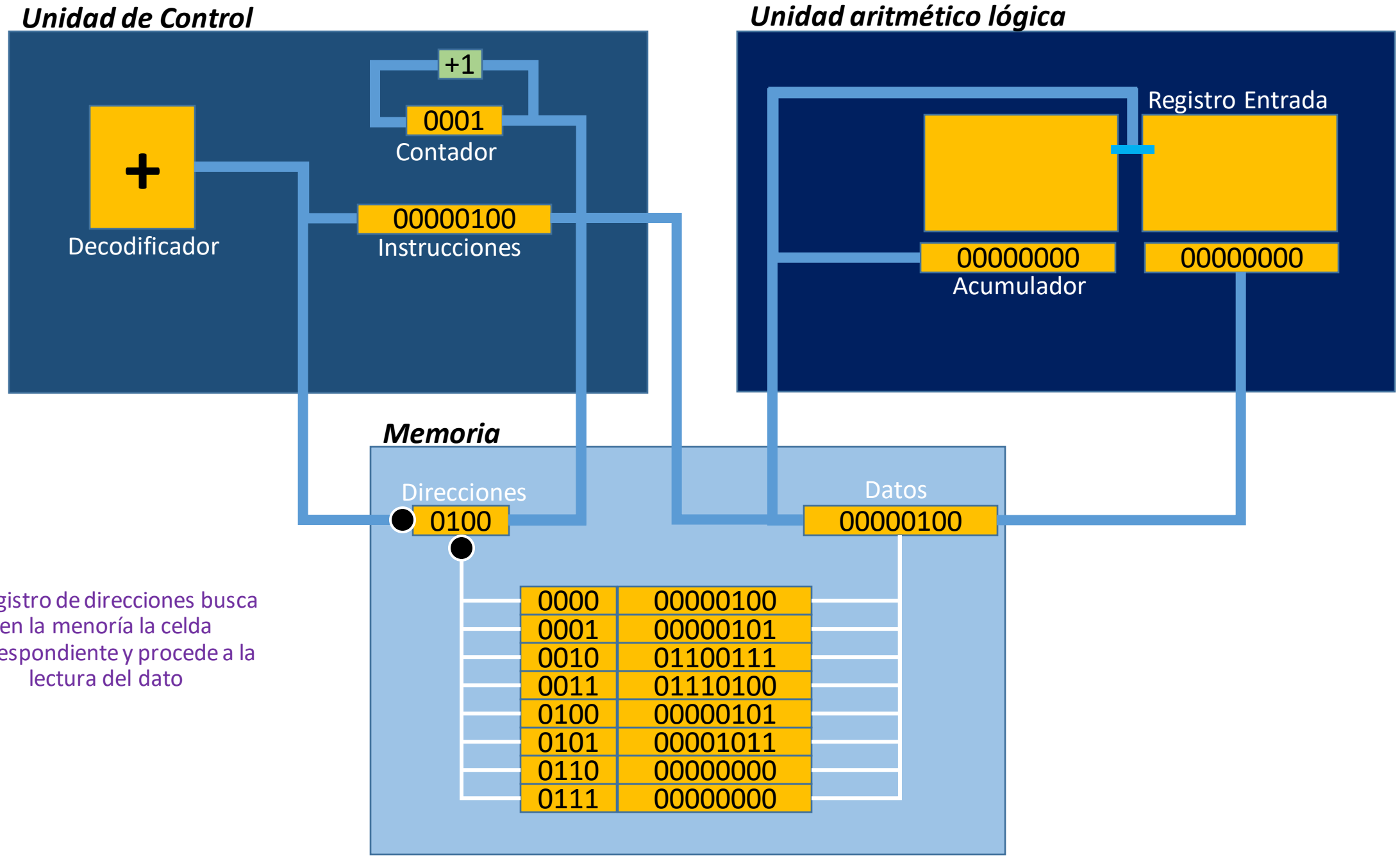
Se realiza el traslado de la información contenida en el Registro de datos al Resgistro de instrucciones, donde se almacenará



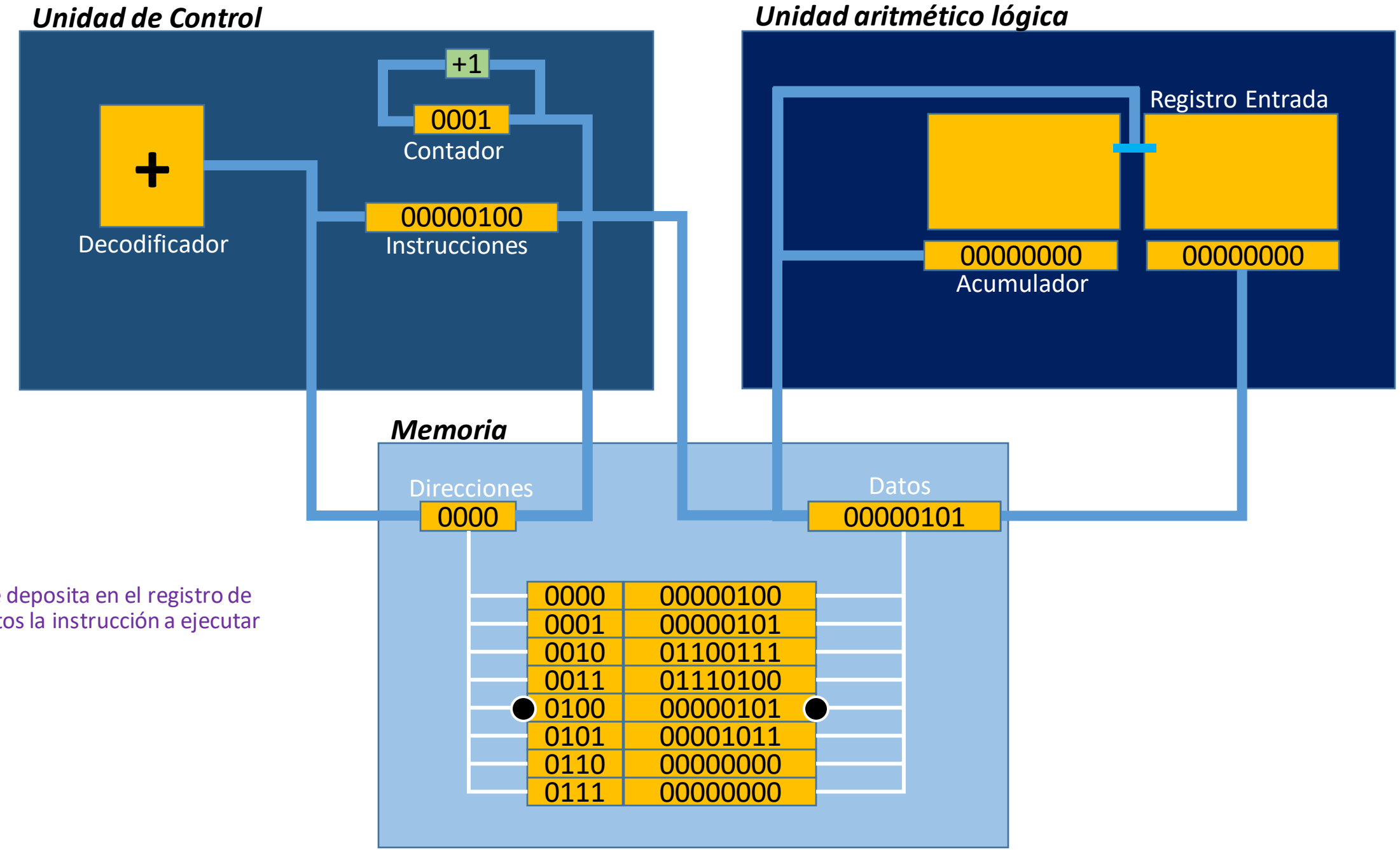
El decodificador procede a la interpretación de la instrucción que serán los 4 primeros bits, es decir, interpreta el código de operación



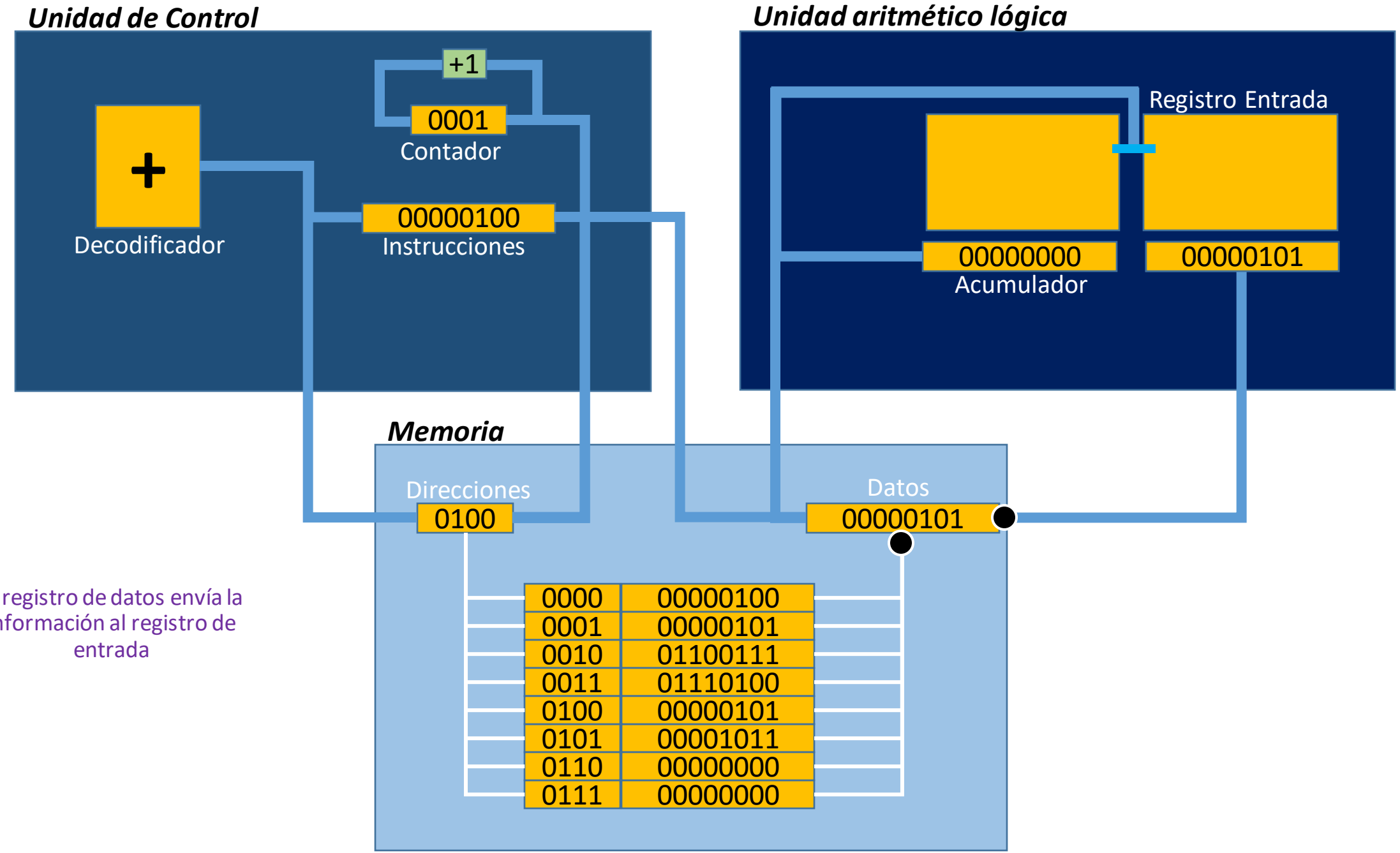
El registro de instrucciones envía los 4 últimos bits al Registro de direcciones



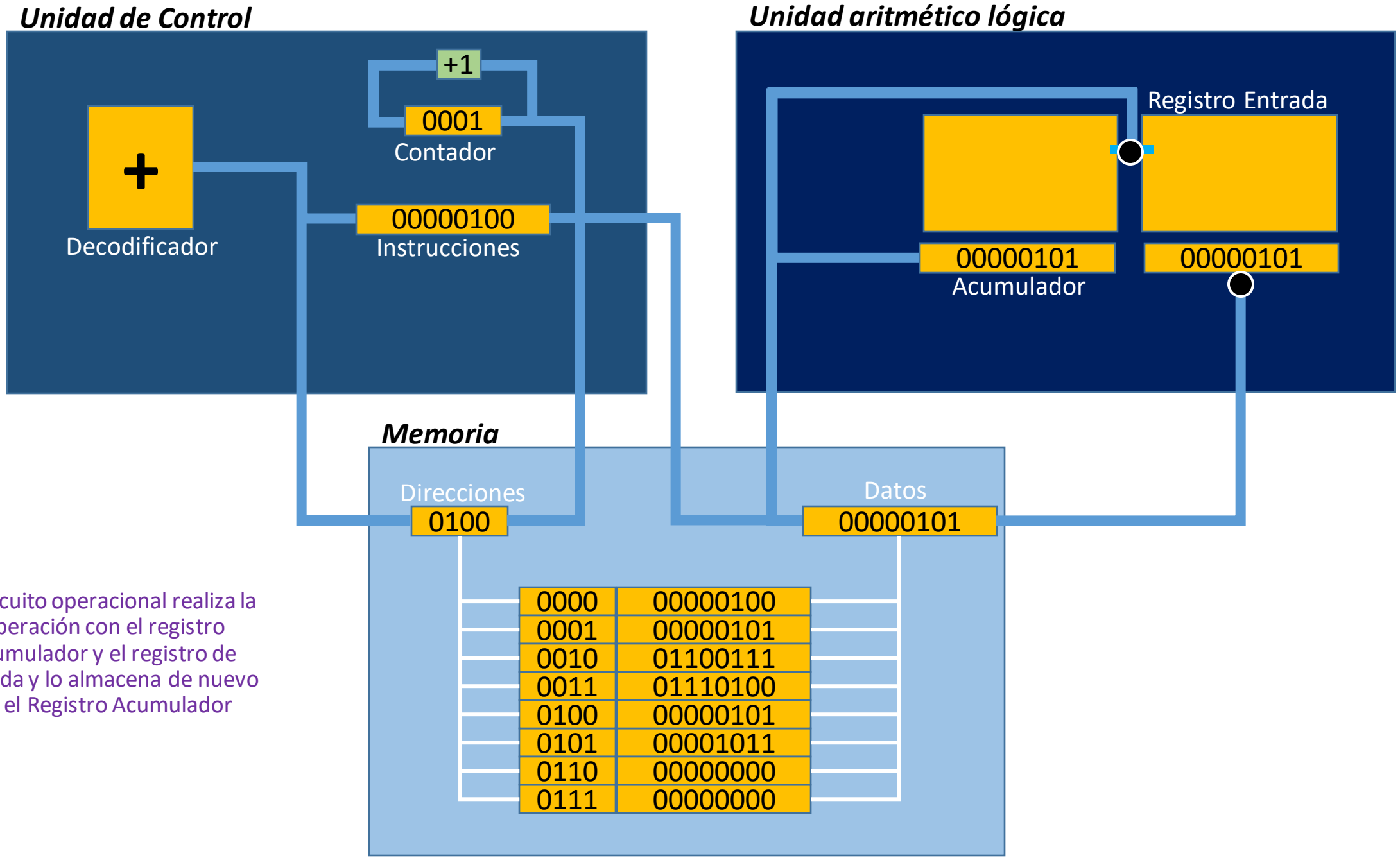
El registro de direcciones busca en la menoría la celda correspondiente y procede a la lectura del dato



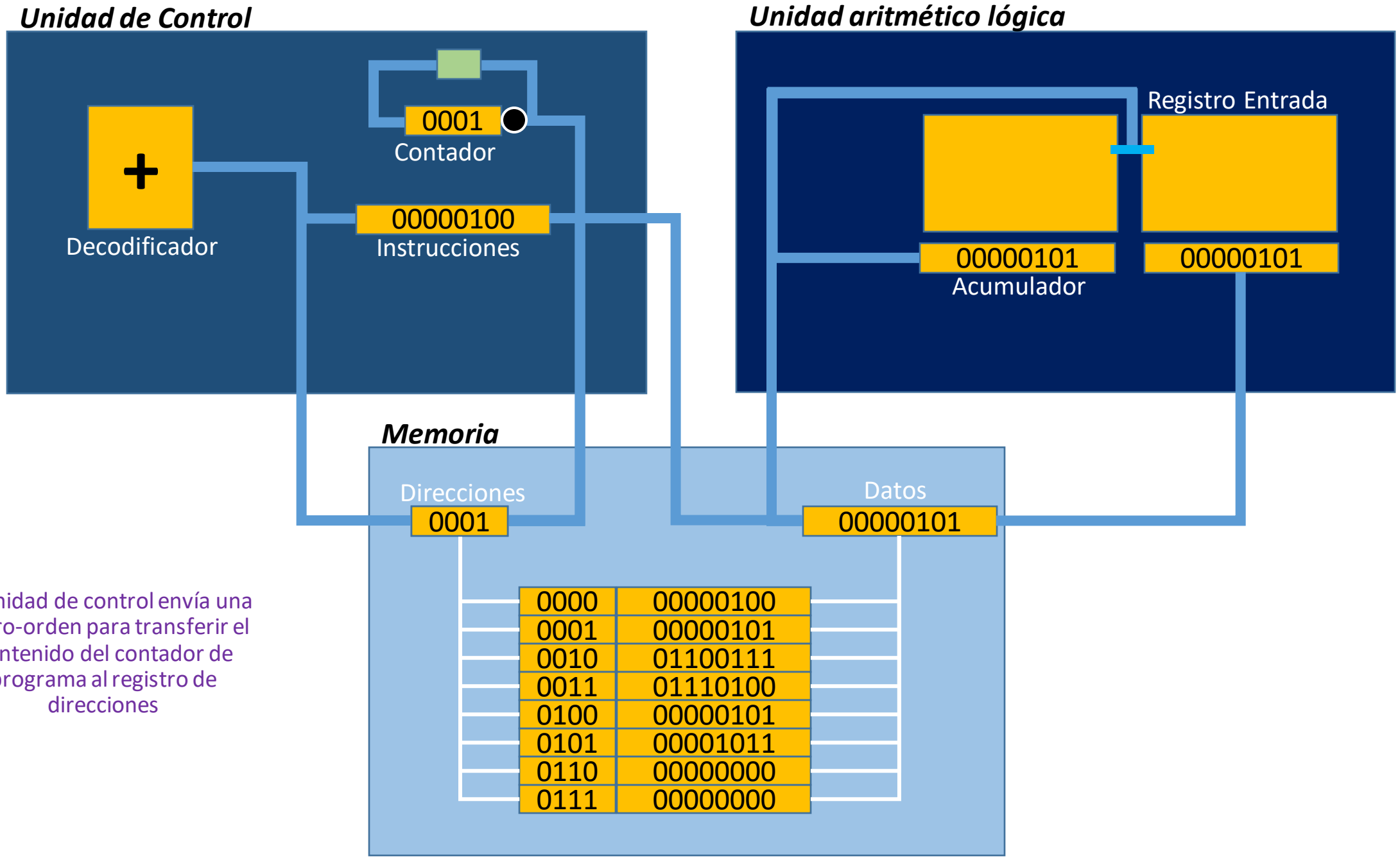
Se deposita en el registro de datos la instrucción a ejecutar



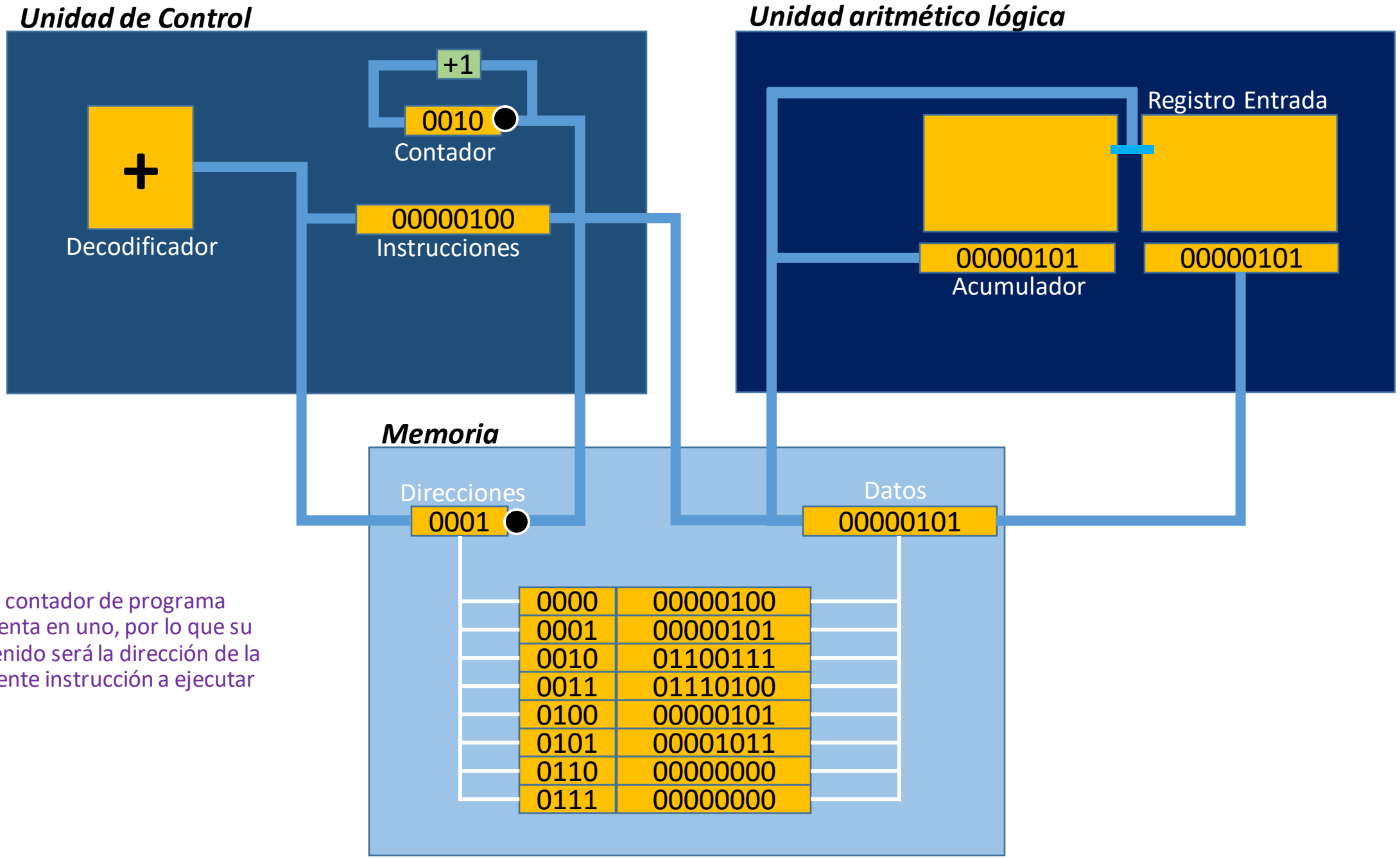
El registro de datos envía la información al registro de entrada



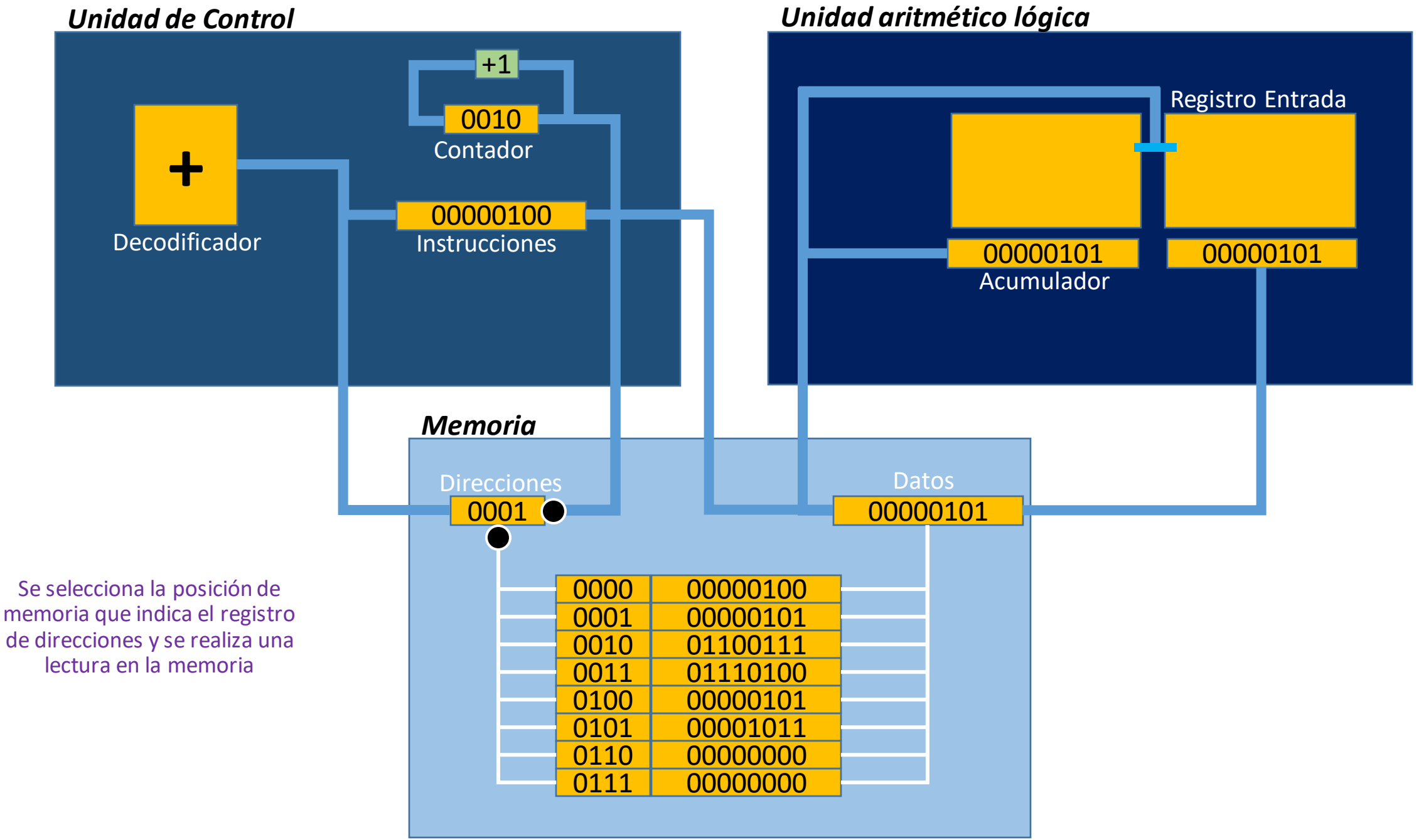
El circuito operacional realiza la operación con el registro acumulador y el registro de entrada y lo almacena de nuevo en el Registro Acumulador



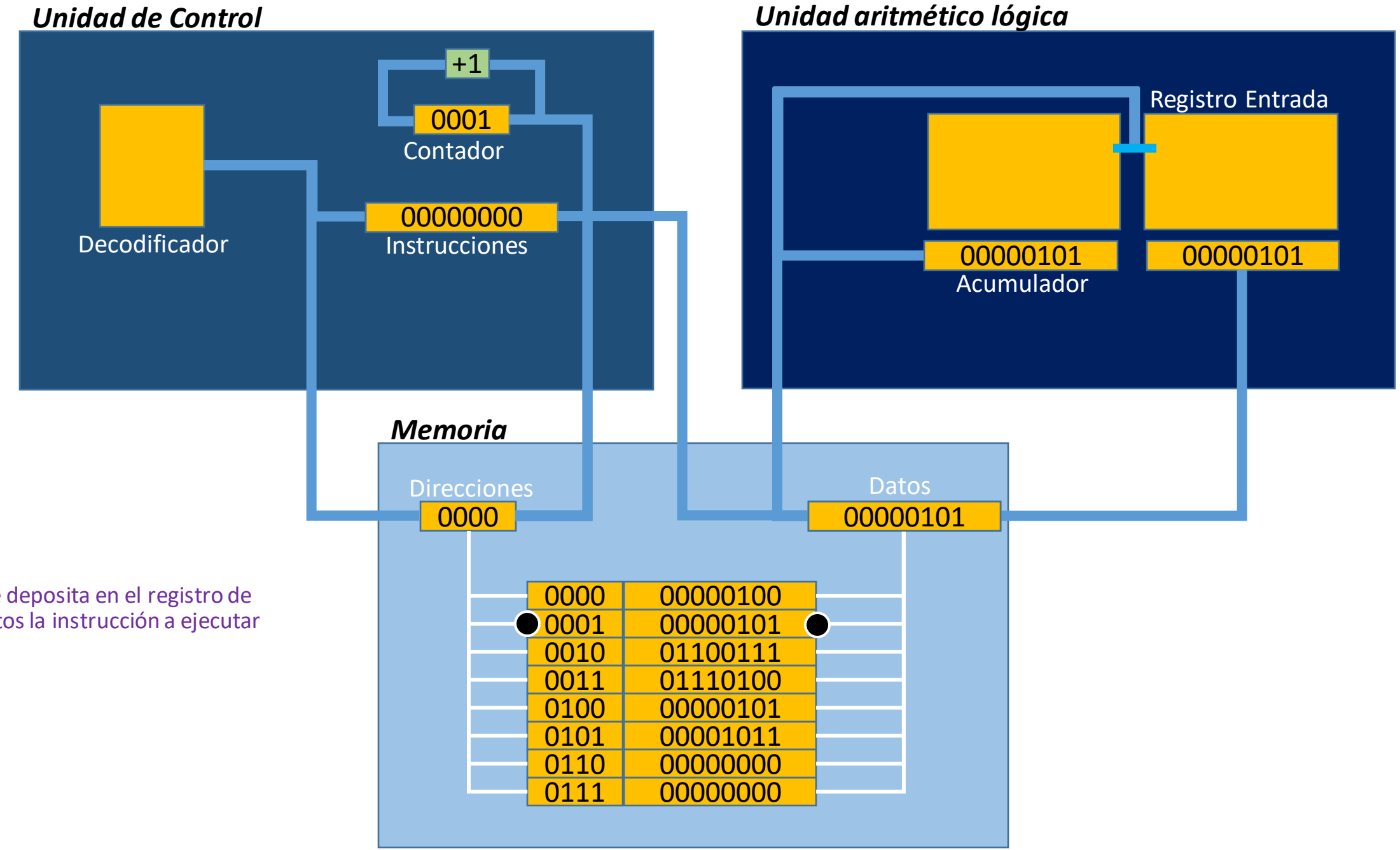
La unidad de control envía una micro-orden para transferir el contenido del contador de programa al registro de direcciones



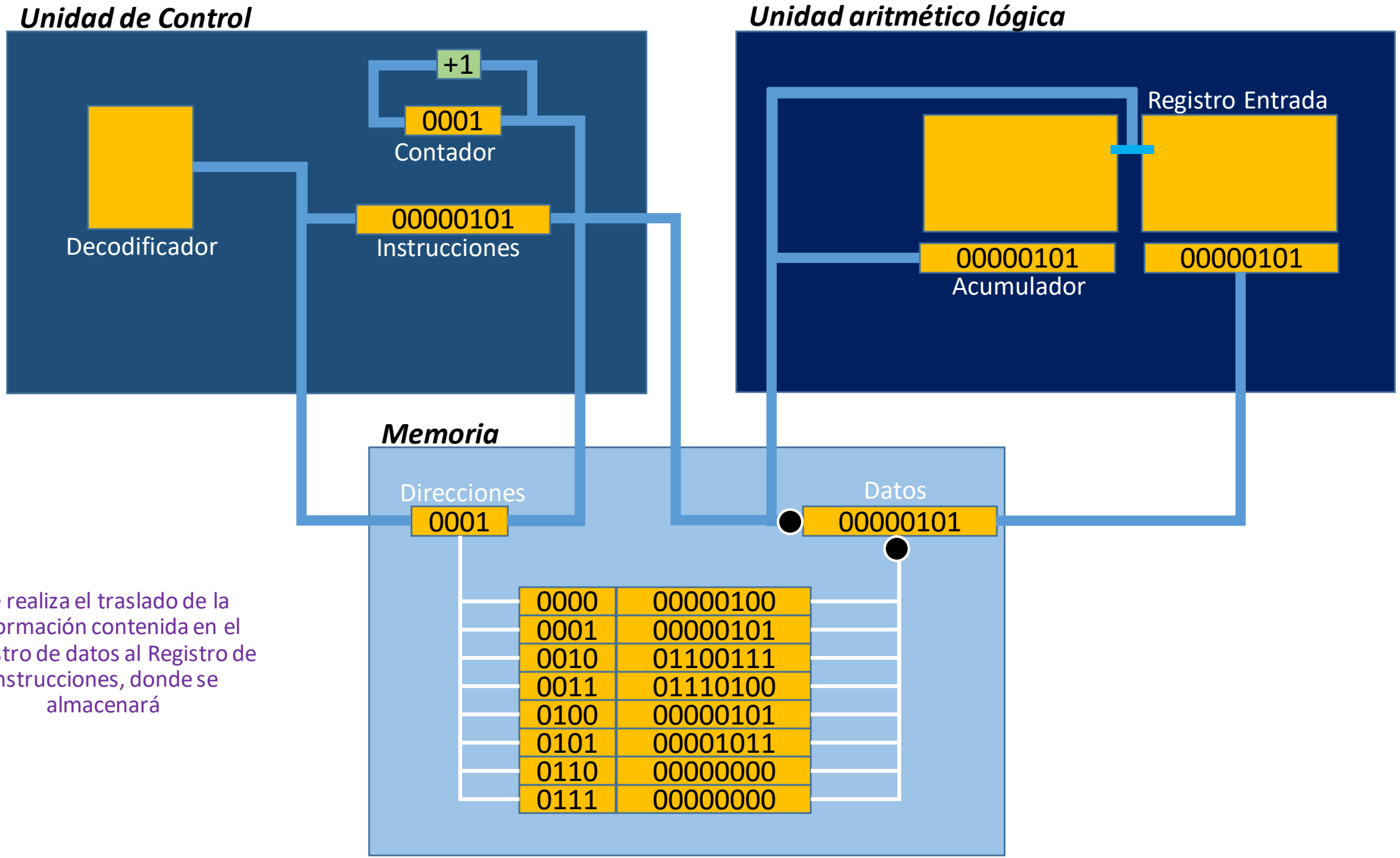
El contador de programa aumenta en uno, por lo que su contenido será la dirección de la siguiente instrucción a ejecutar



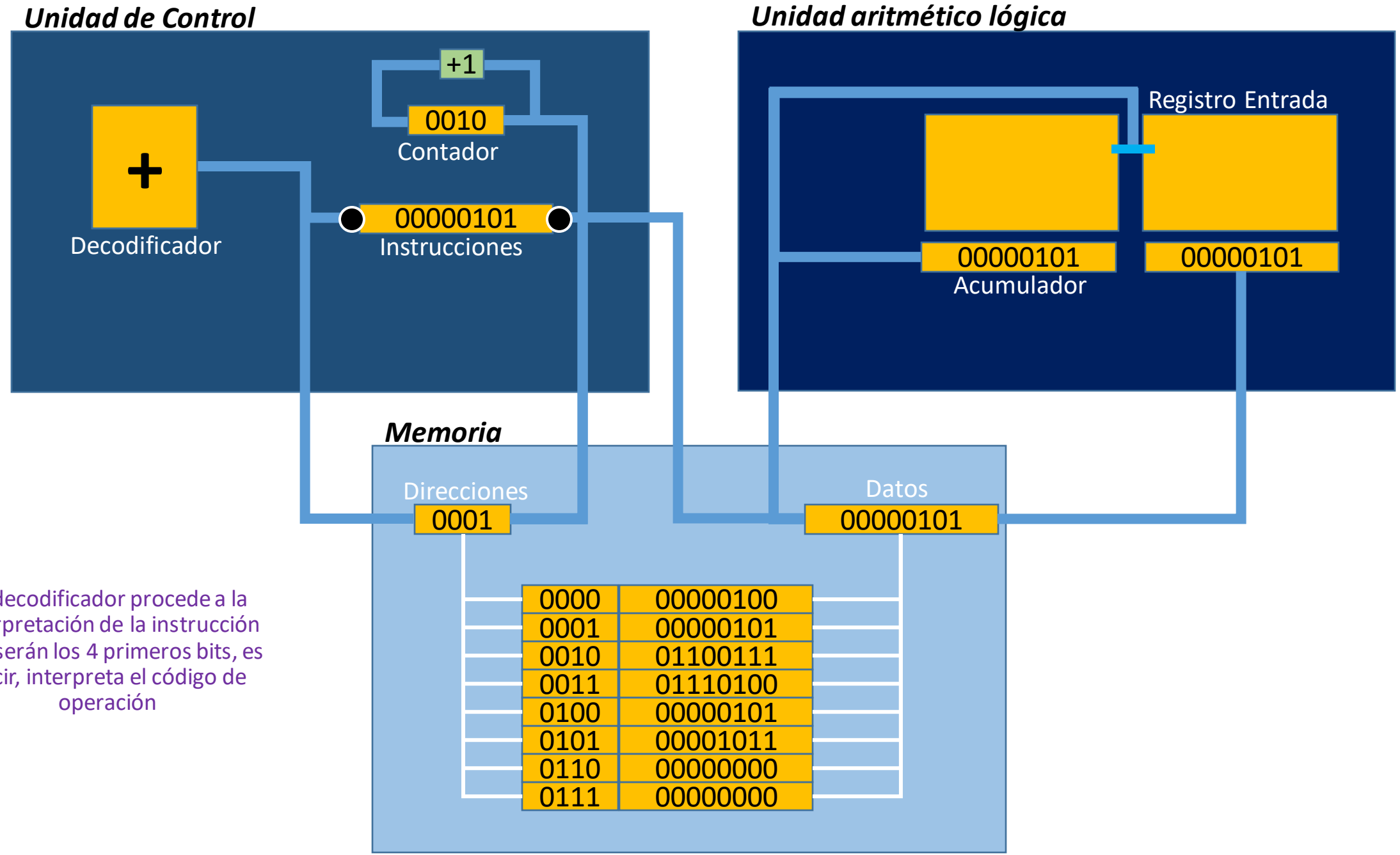
Se selecciona la posición de memoria que indica el registro de direcciones y se realiza una lectura en la memoria



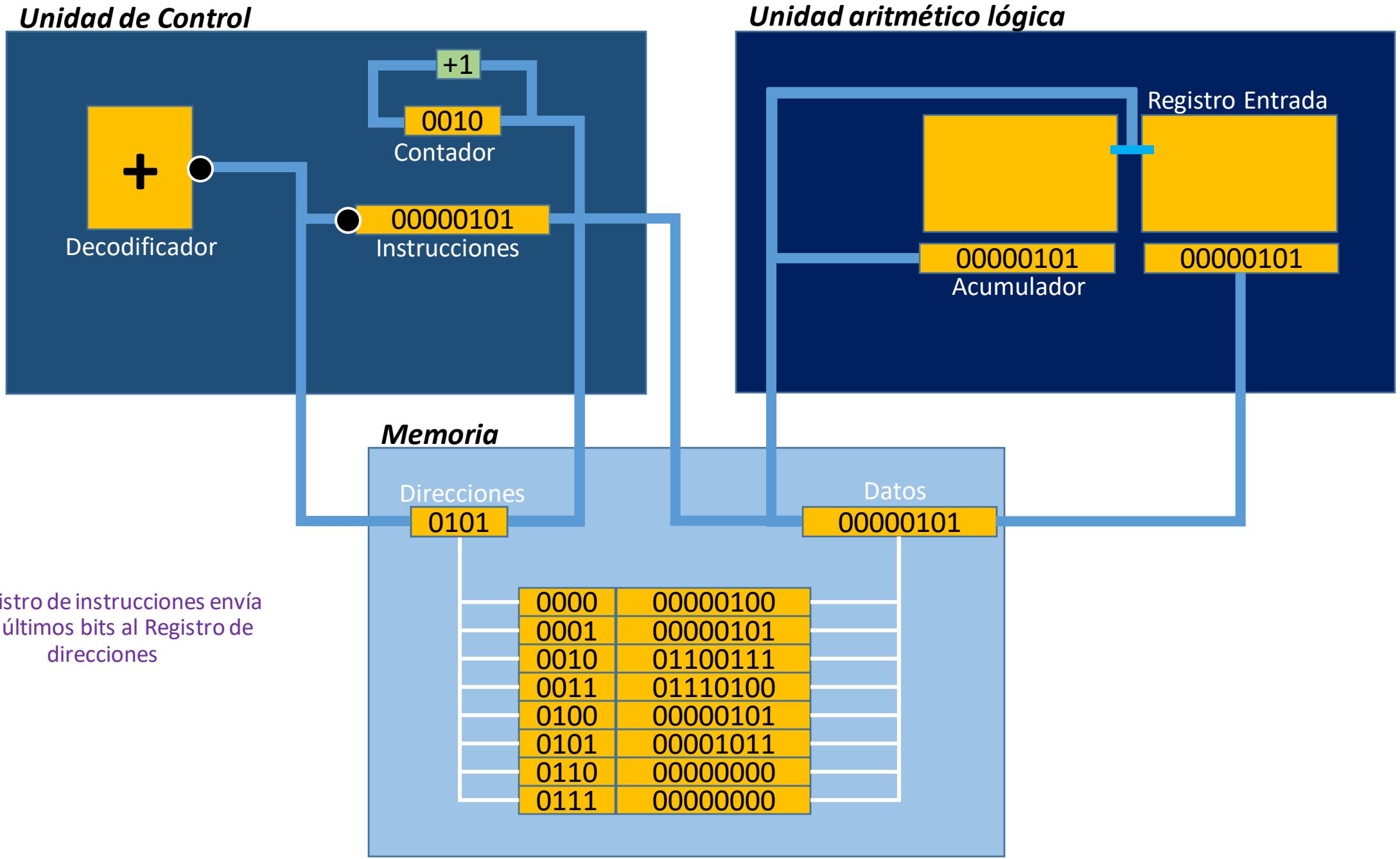
Se deposita en el registro de datos la instrucción a ejecutar



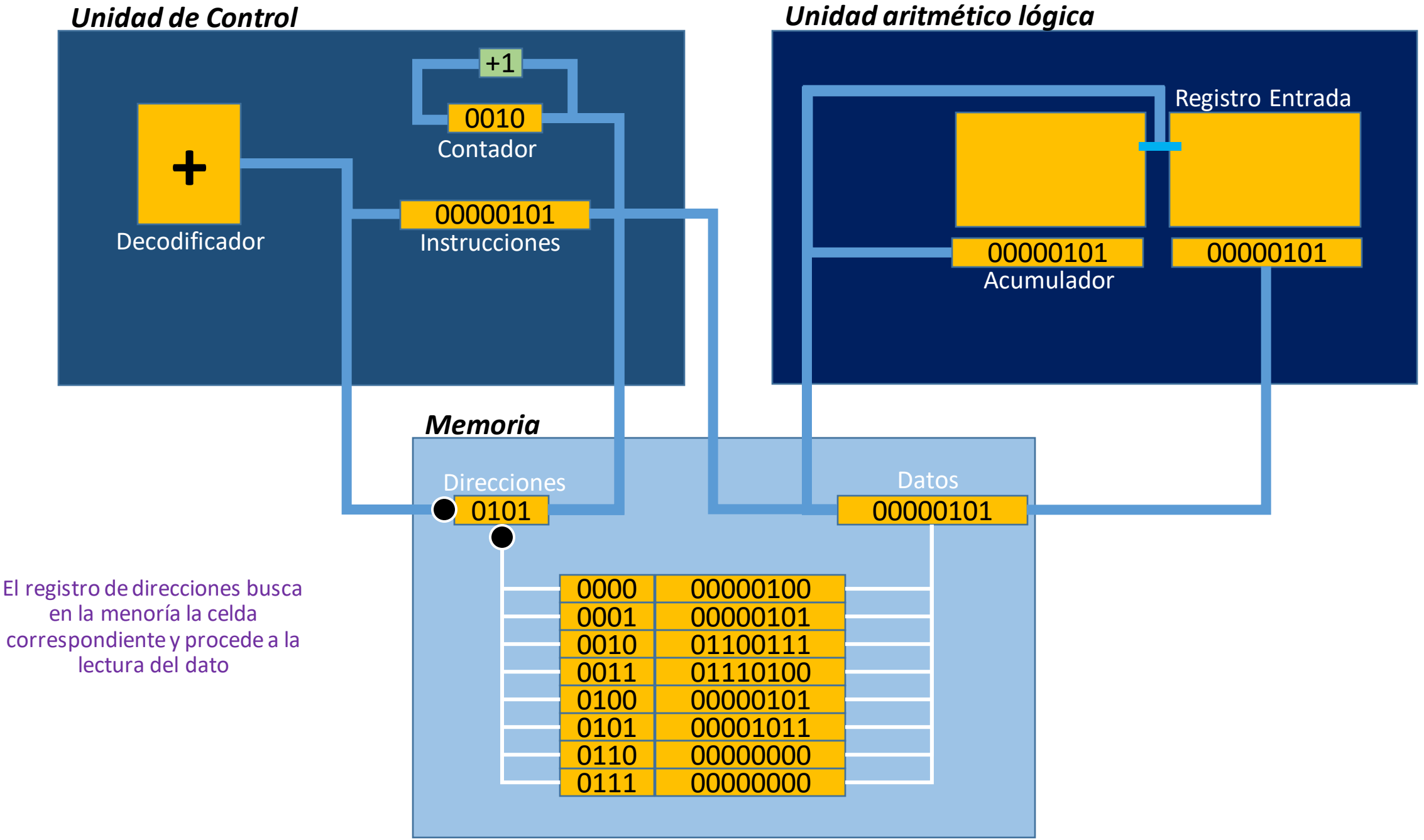
Se realiza el traslado de la información contenida en el Registro de datos al Registro de instrucciones, donde se almacenará



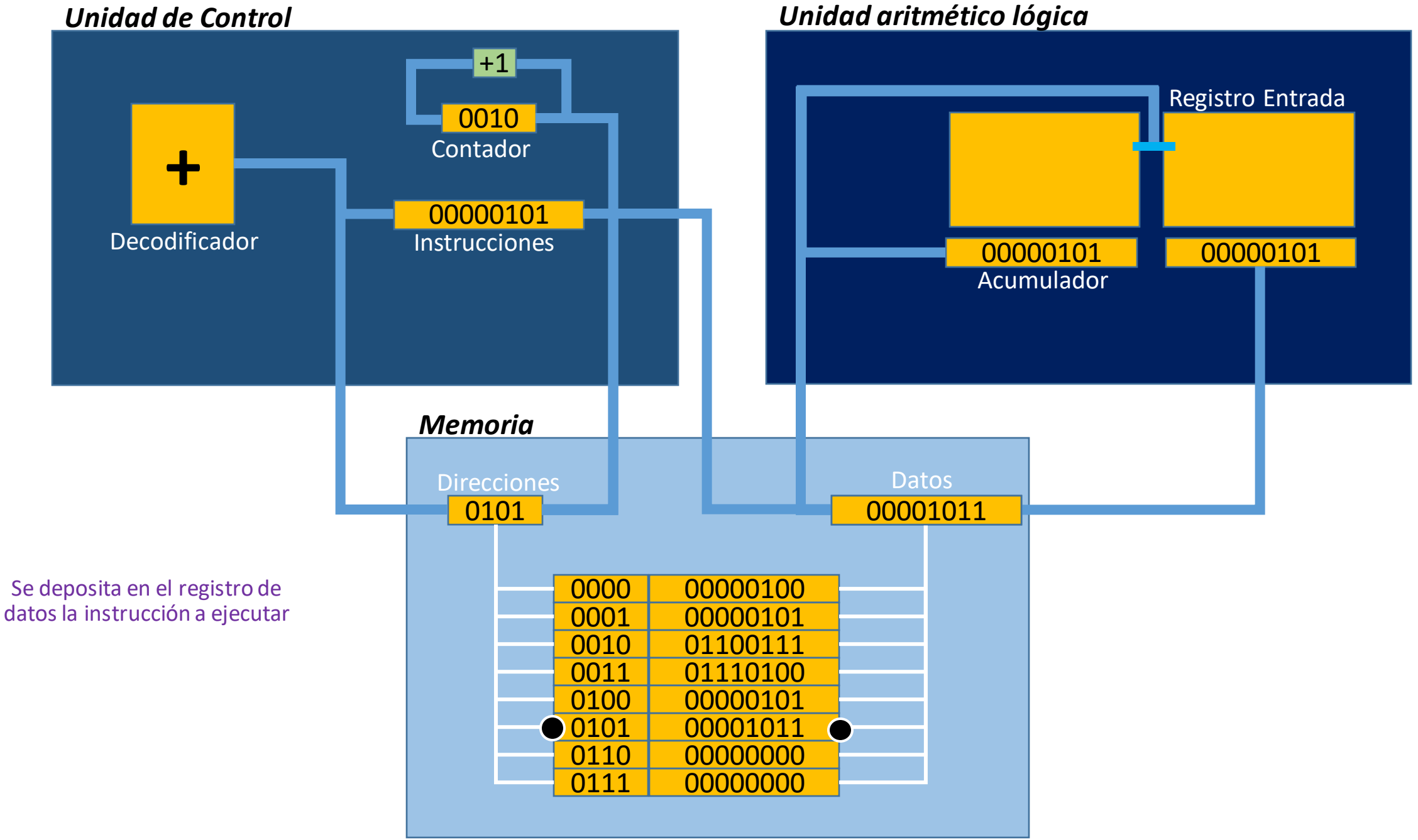
El decodificador procede a la interpretación de la instrucción que serán los 4 primeros bits, es decir, interpreta el código de operación



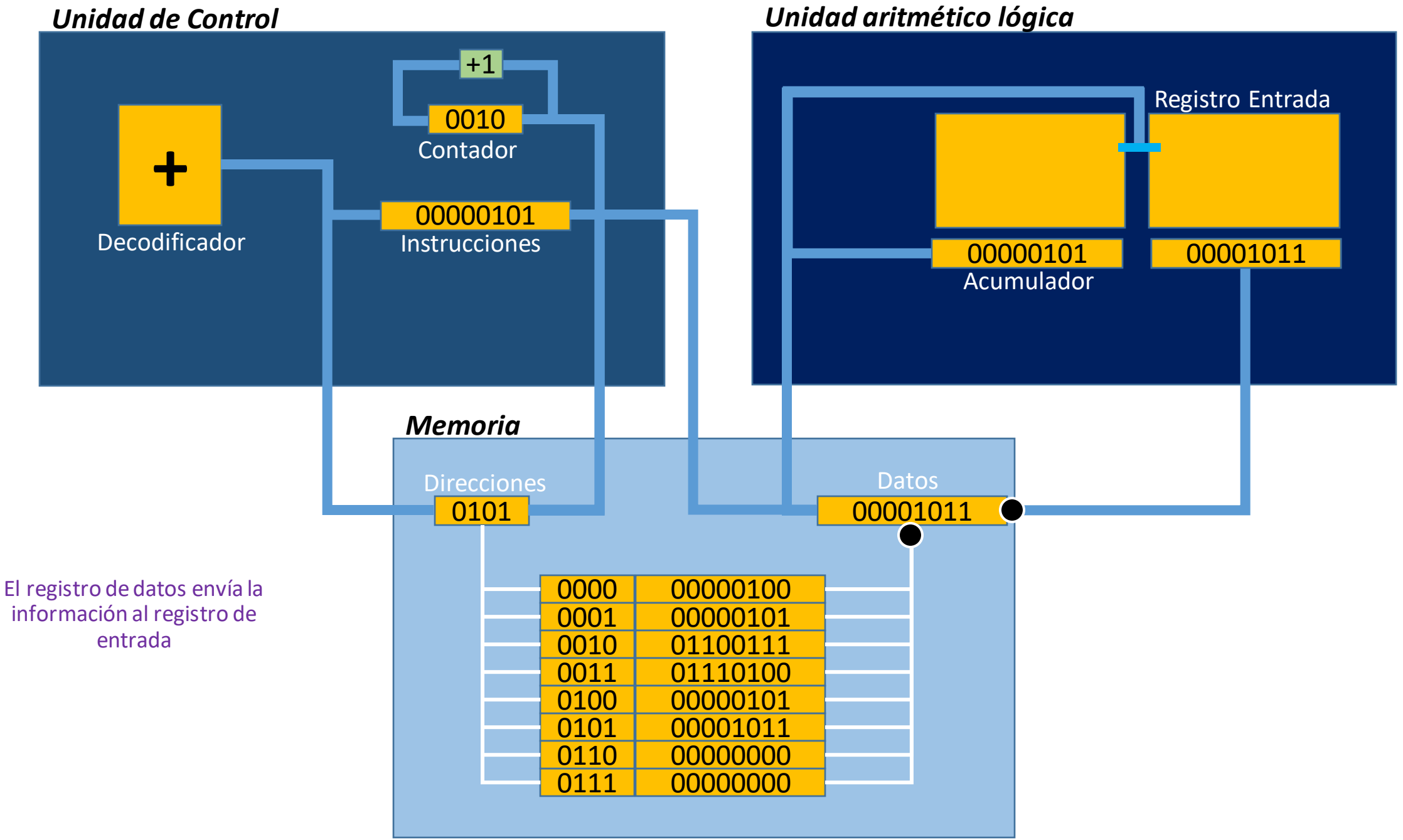
El registro de instrucciones envía los 4 últimos bits al Registro de direcciones



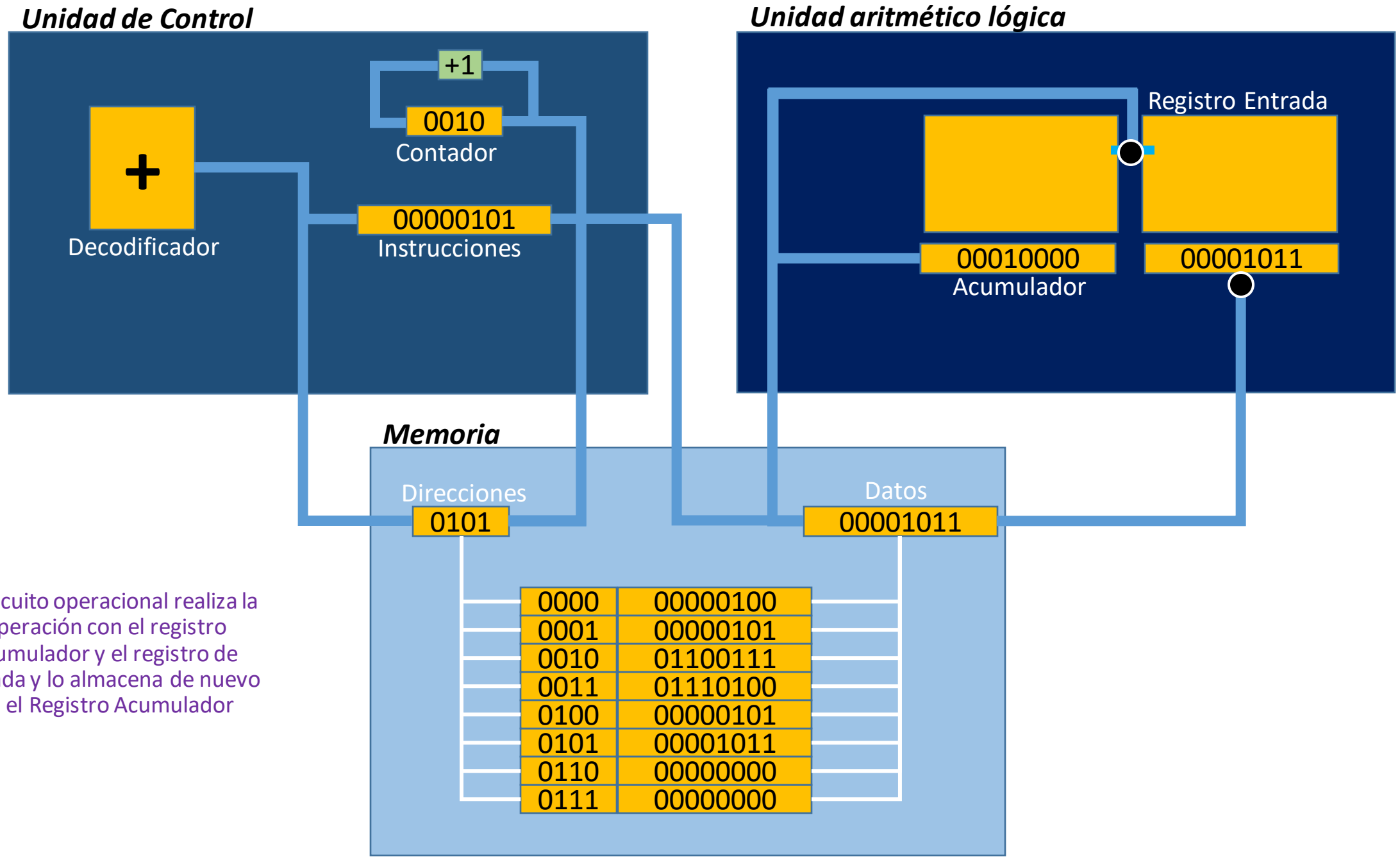
El registro de direcciones busca en la menoría la celda correspondiente y procede a la lectura del dato



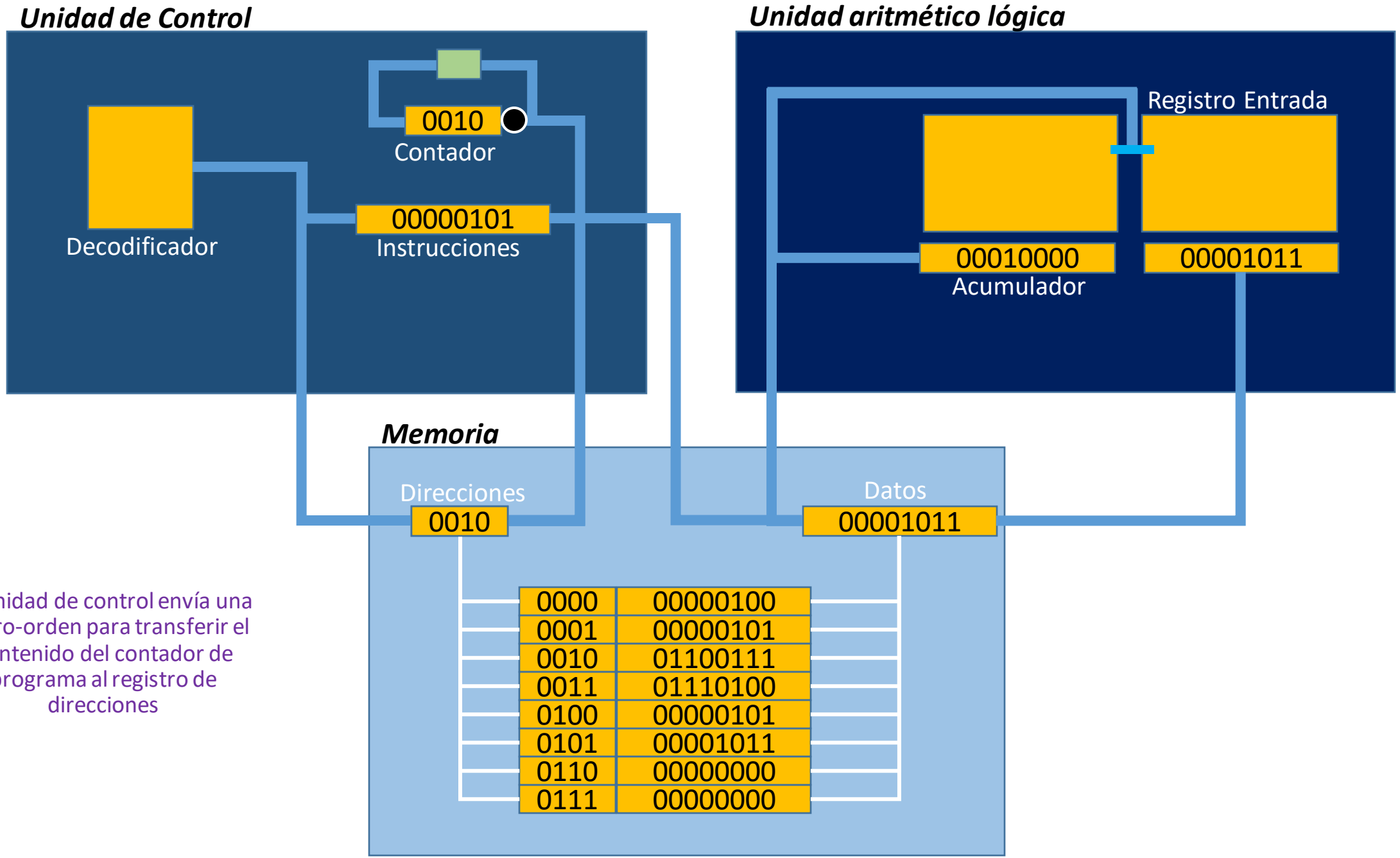
Se deposita en el registro de datos la instrucción a ejecutar



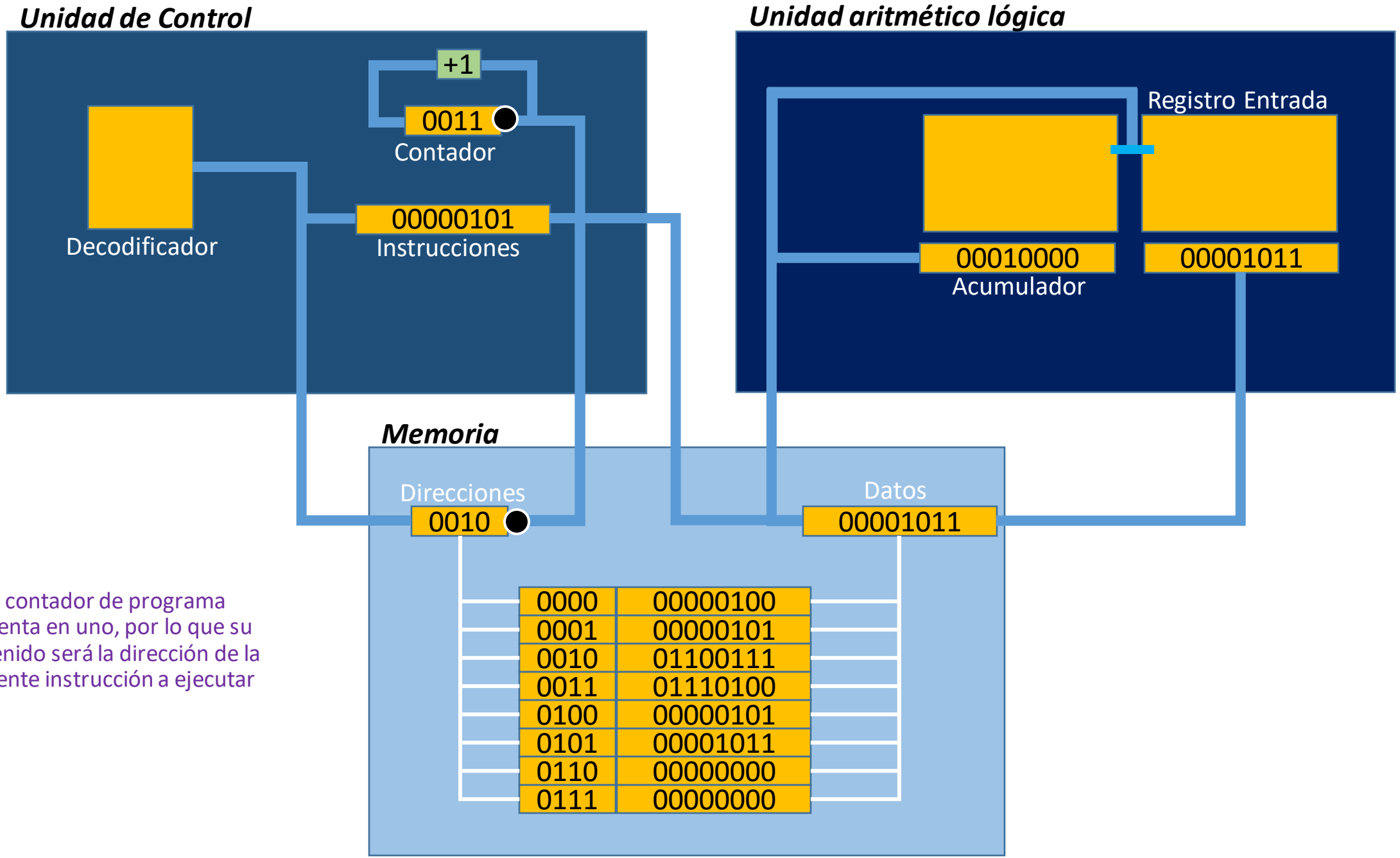
El registro de datos envía la información al registro de entrada



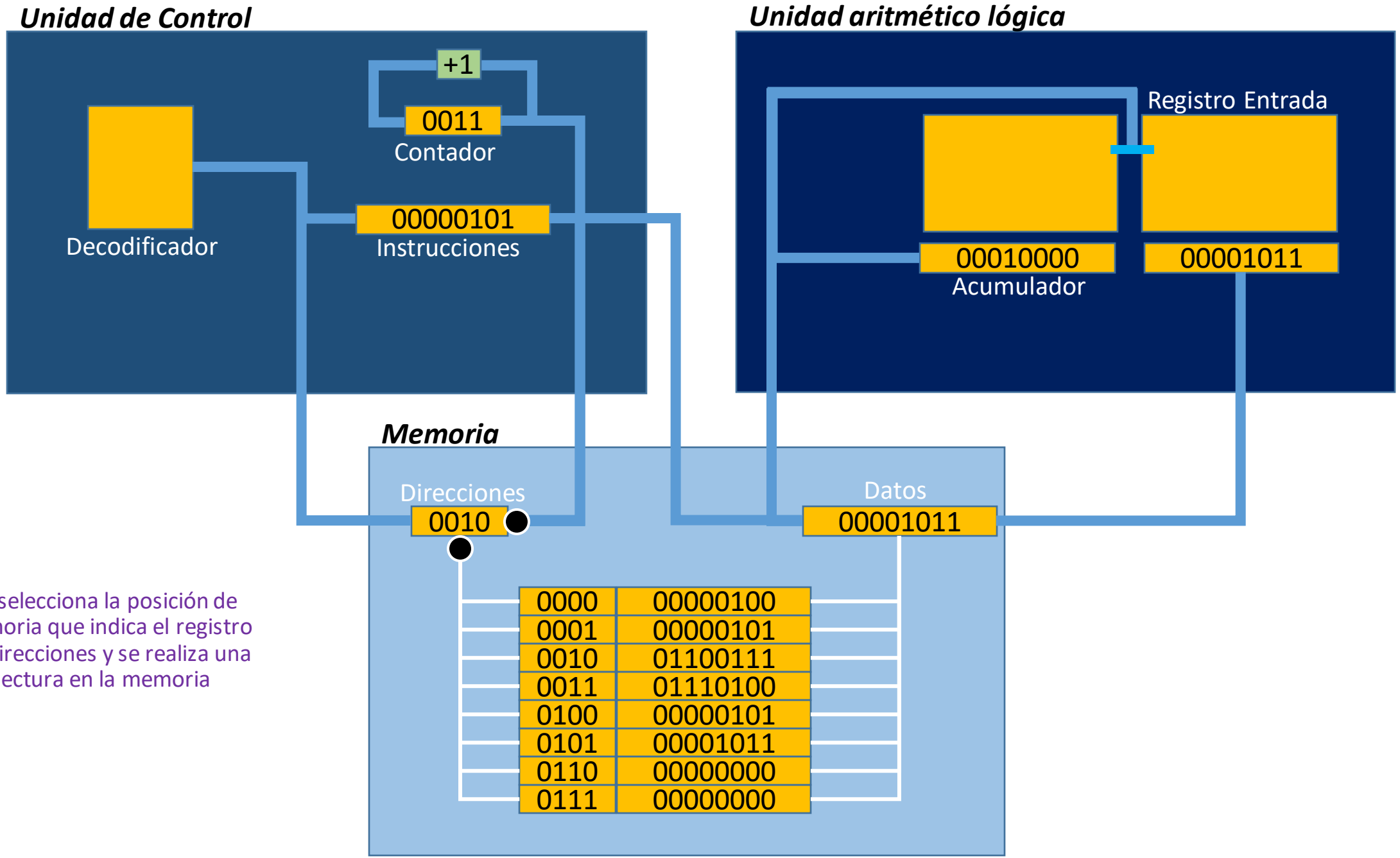
El circuito operacional realiza la operación con el registro acumulador y el registro de entrada y lo almacena de nuevo en el Registro Acumulador



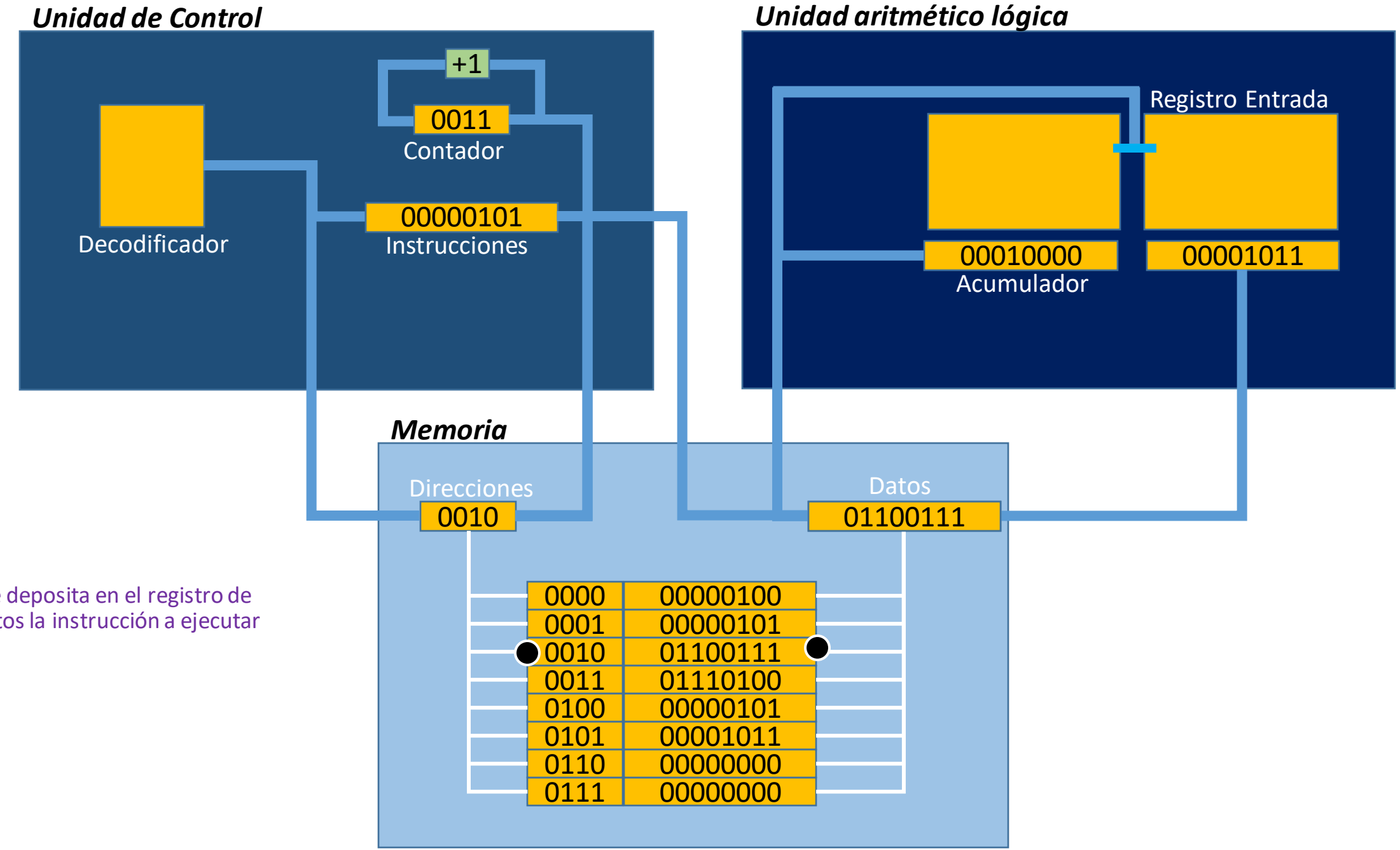
La unidad de control envía una micro-orden para transferir el contenido del contador de programa al registro de direcciones



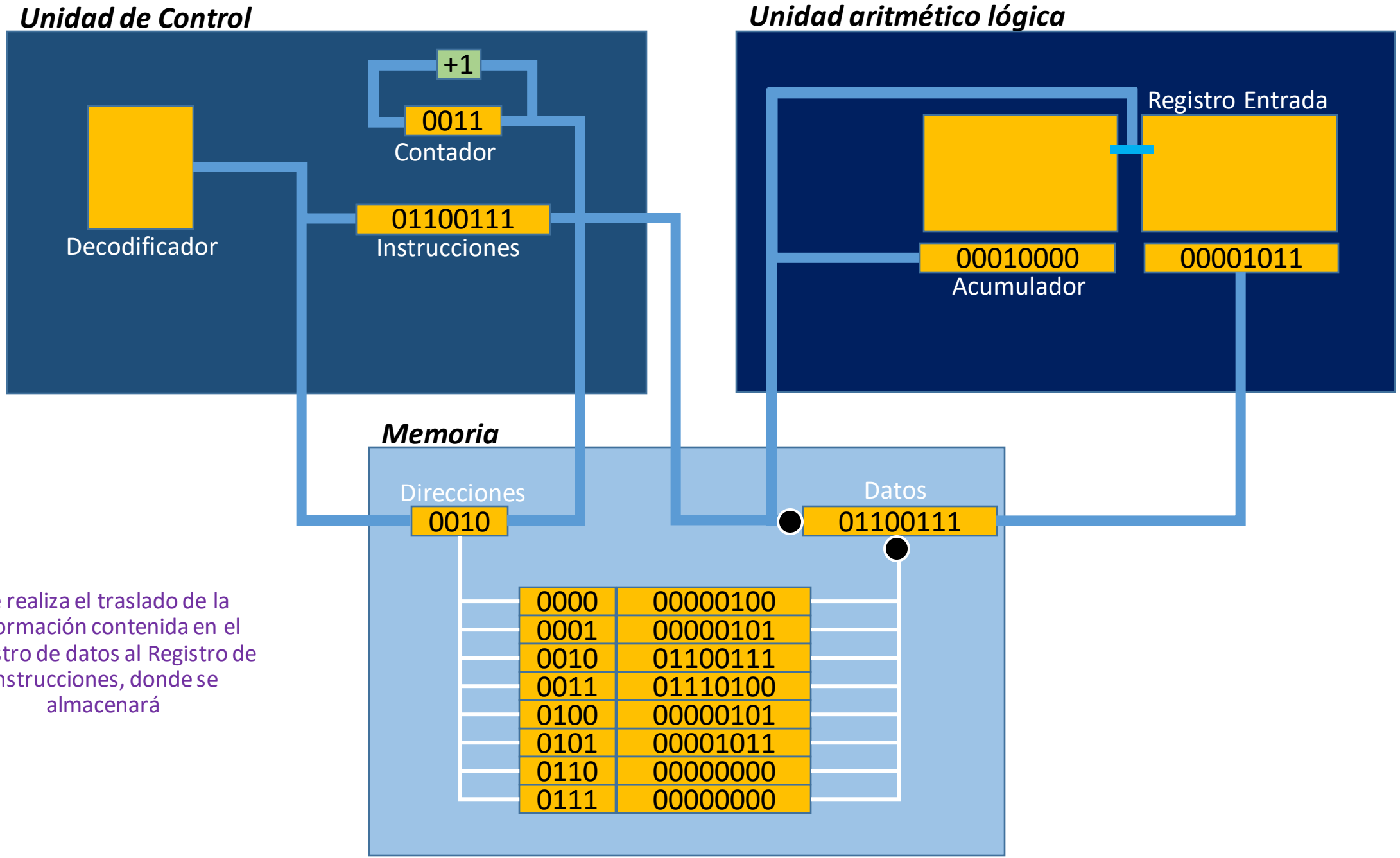
El contador de programa aumenta en uno, por lo que su contenido será la dirección de la siguiente instrucción a ejecutar



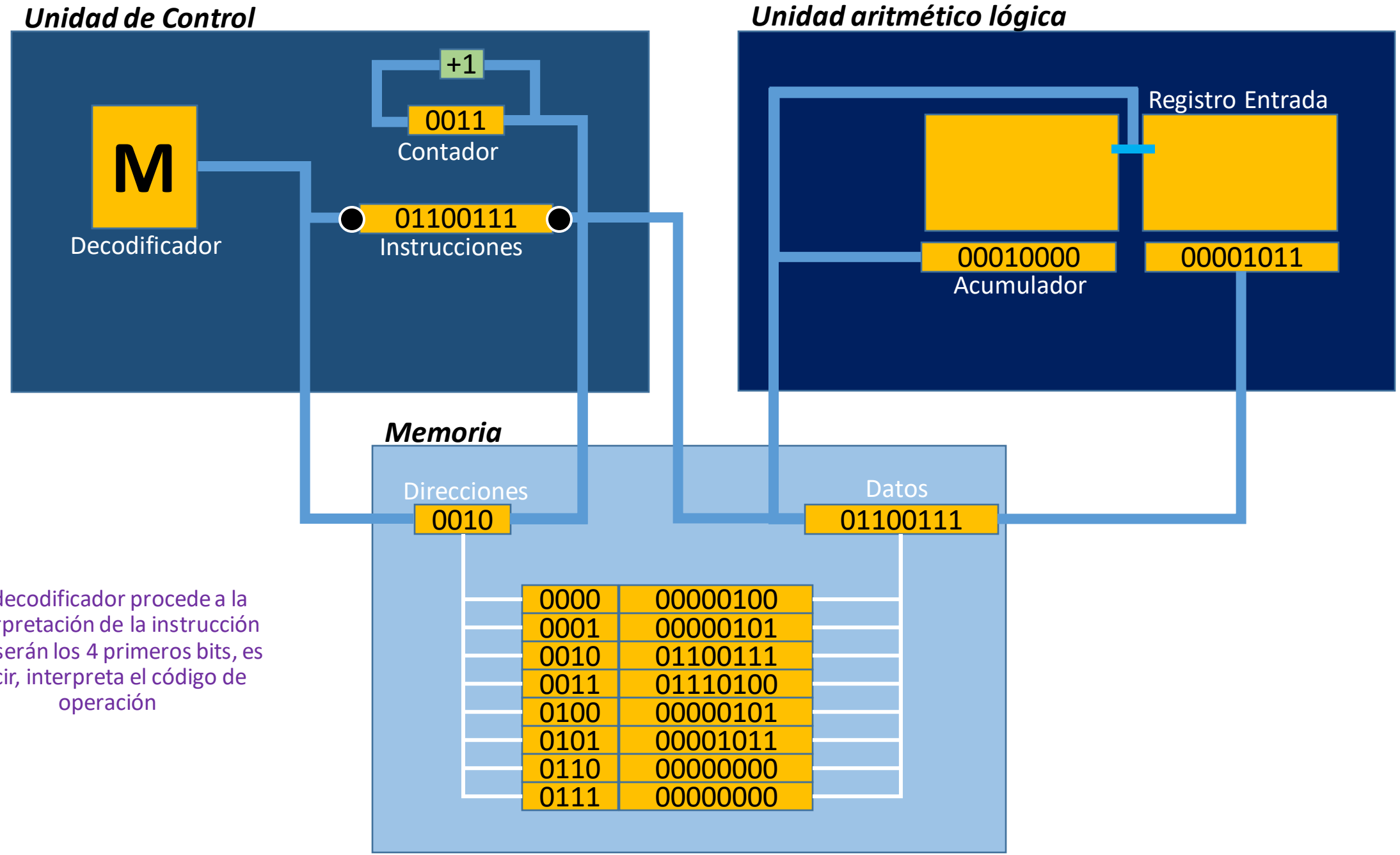
Se selecciona la posición de memoria que indica el registro de direcciones y se realiza una lectura en la memoria



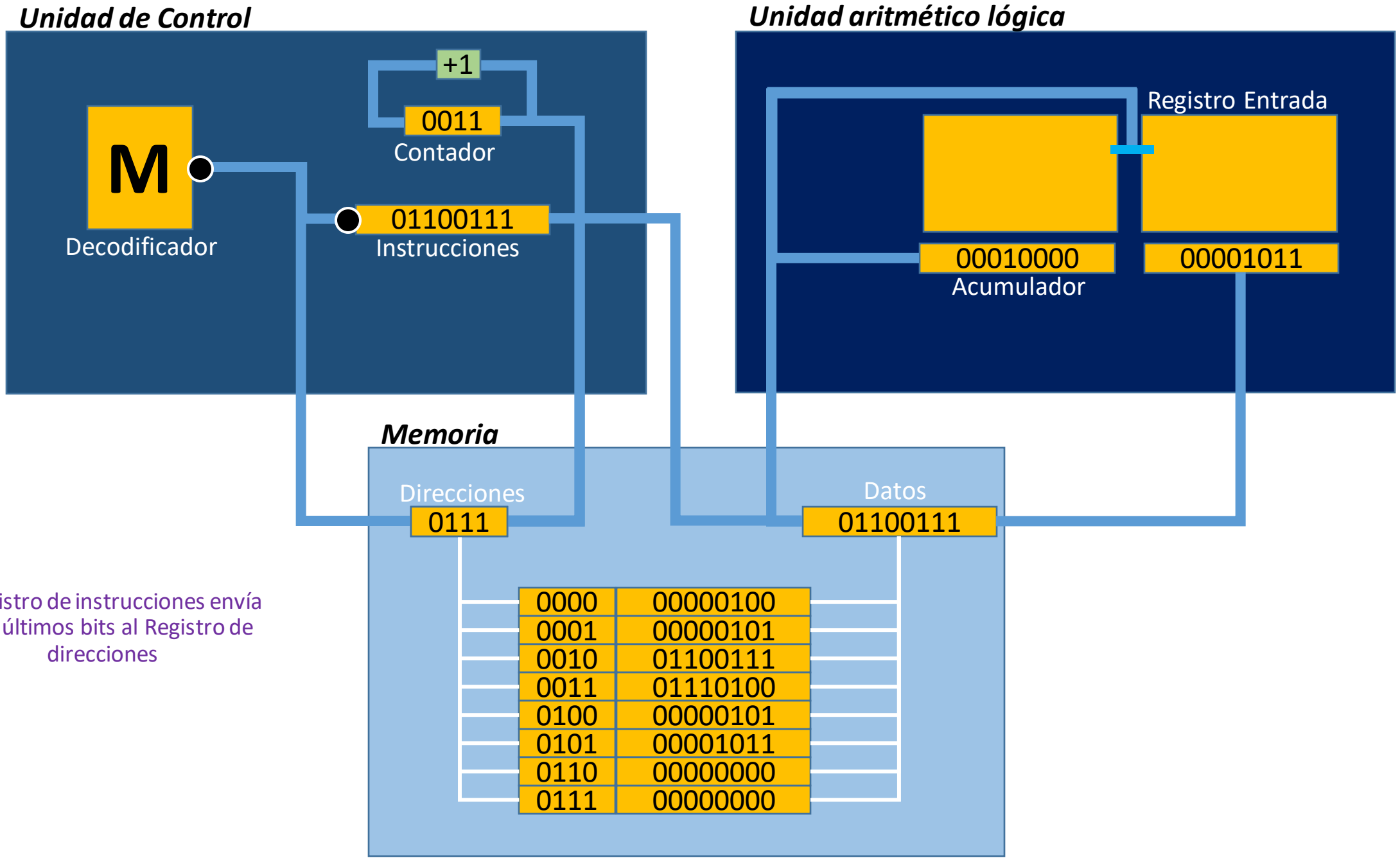
Se deposita en el registro de datos la instrucción a ejecutar



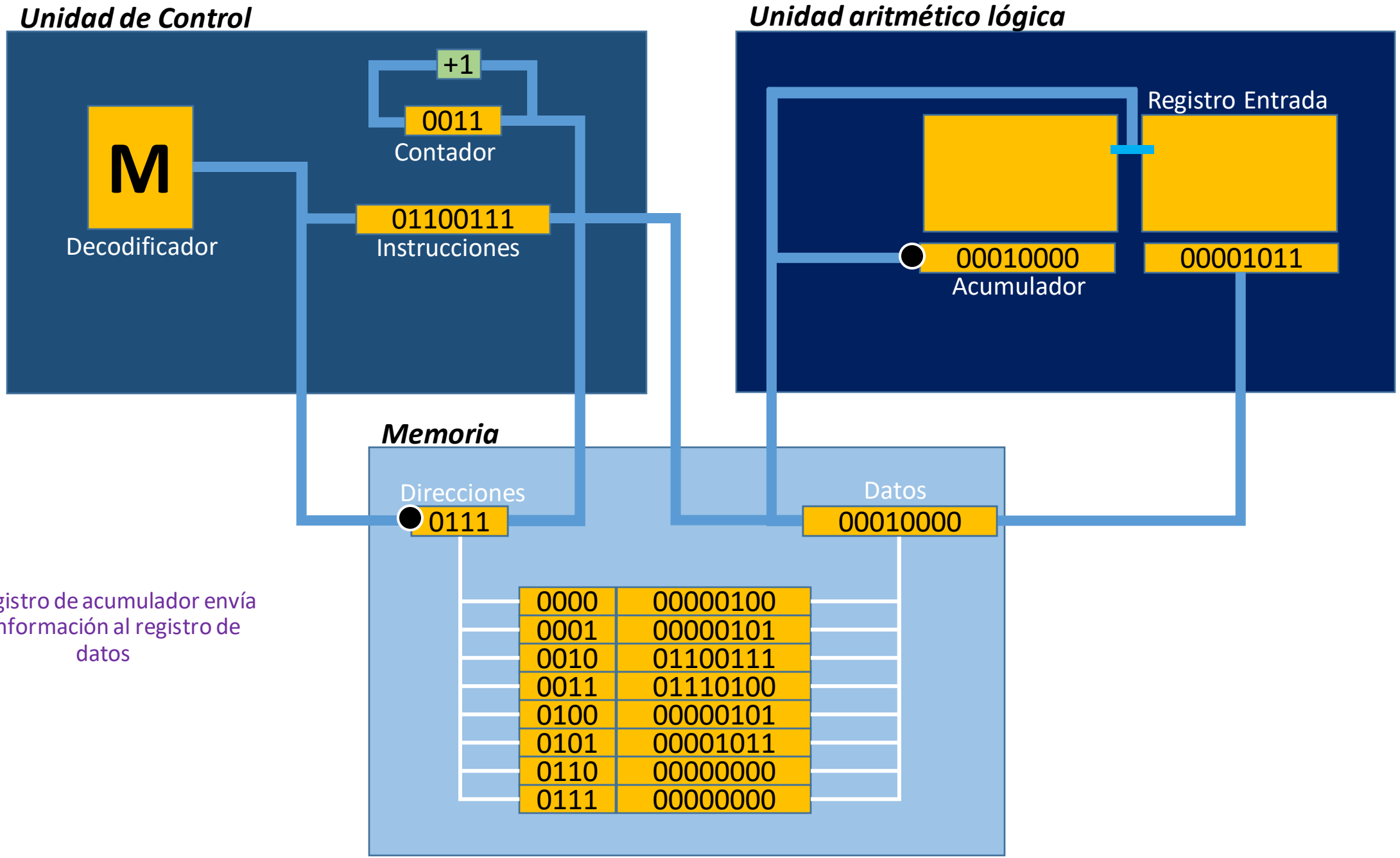
Se realiza el traslado de la información contenida en el Registro de datos al Registro de instrucciones, donde se almacenará



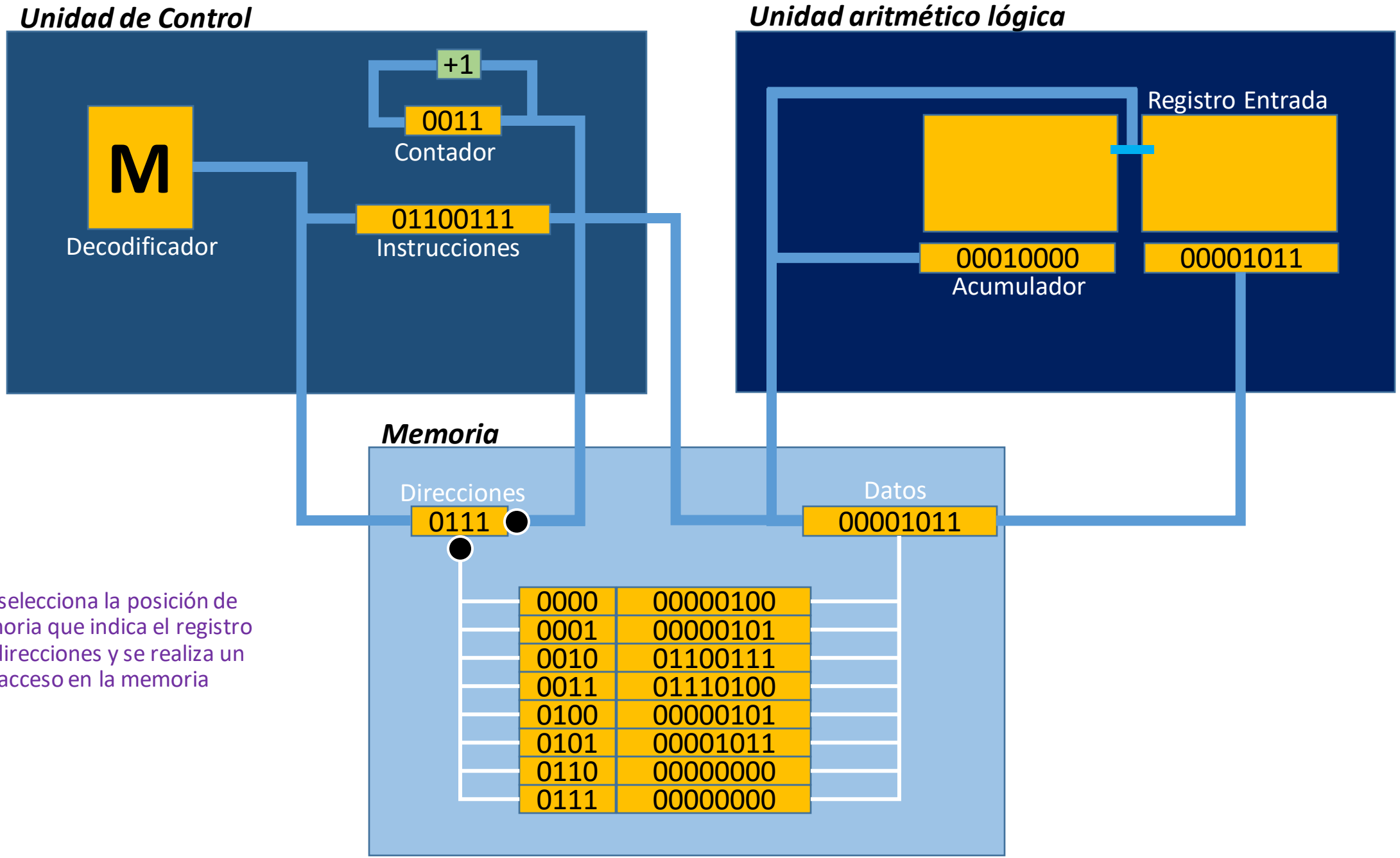
El decodificador procede a la interpretación de la instrucción que serán los 4 primeros bits, es decir, interpreta el código de operación



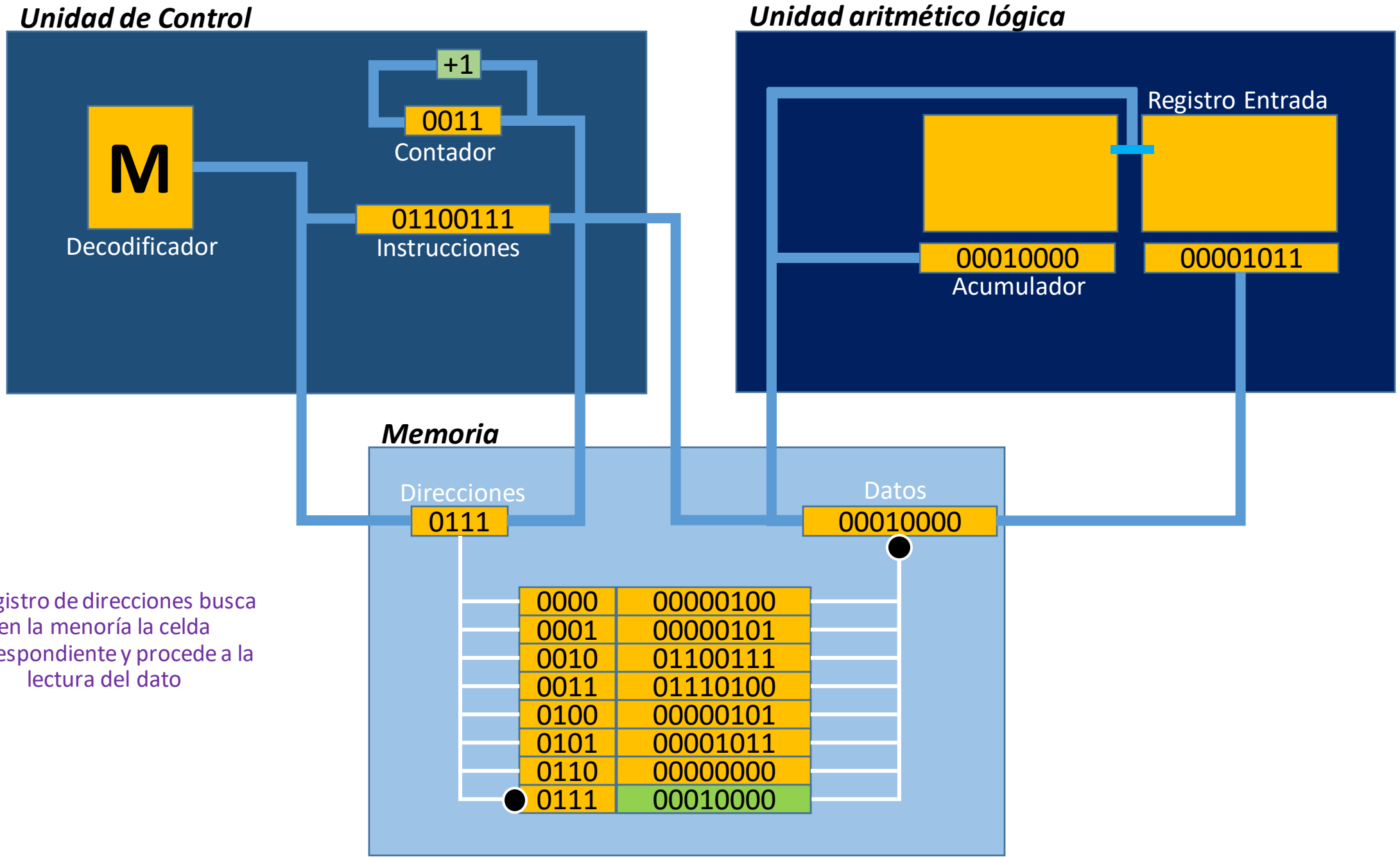
El registro de instrucciones envía los 4 últimos bits al Registro de direcciones



El registro de acumulador envía la información al registro de datos



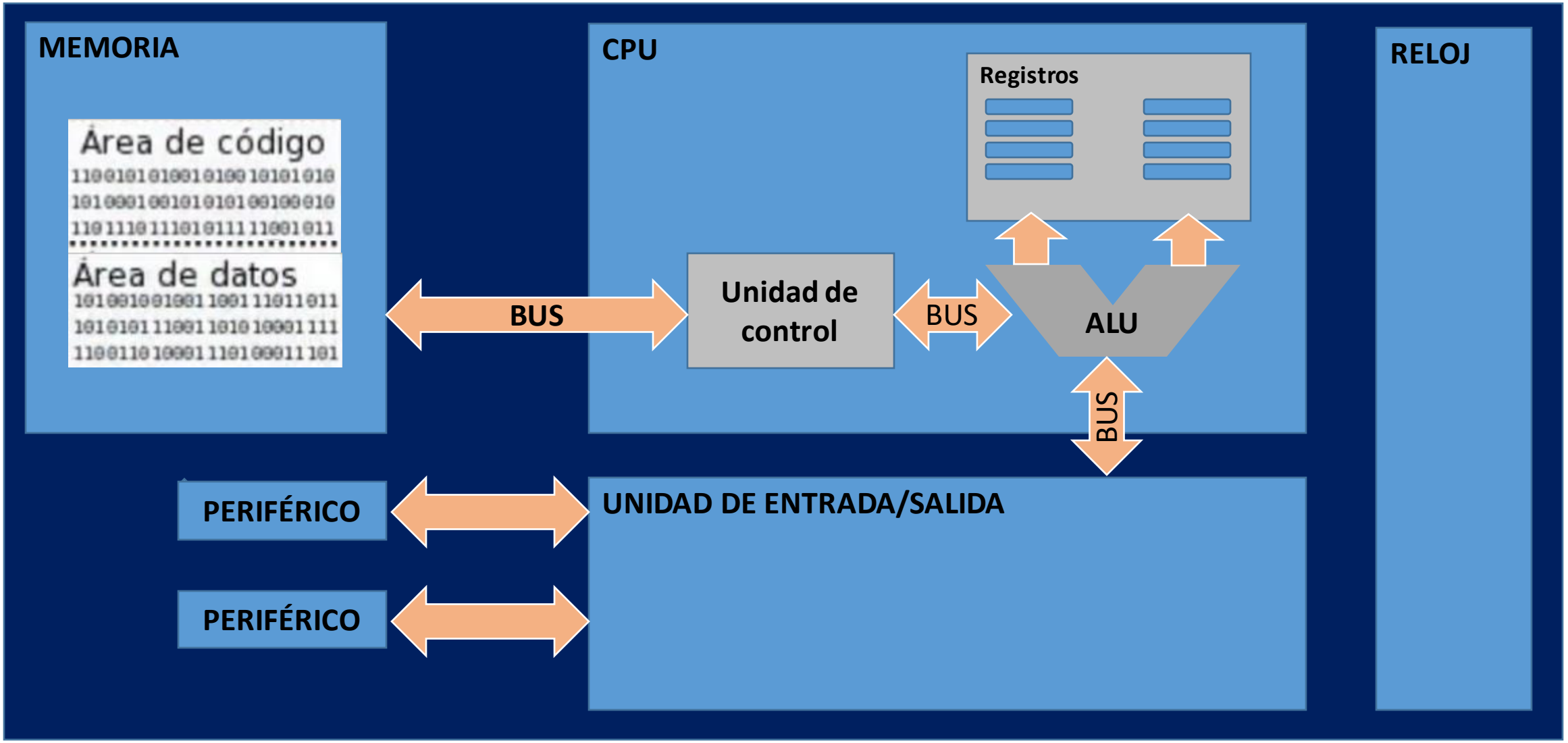
Se selecciona la posición de memoria que indica el registro de direcciones y se realiza un acceso en la memoria



El registro de direcciones busca en la menoría la celda correspondiente y procede a la lectura del dato

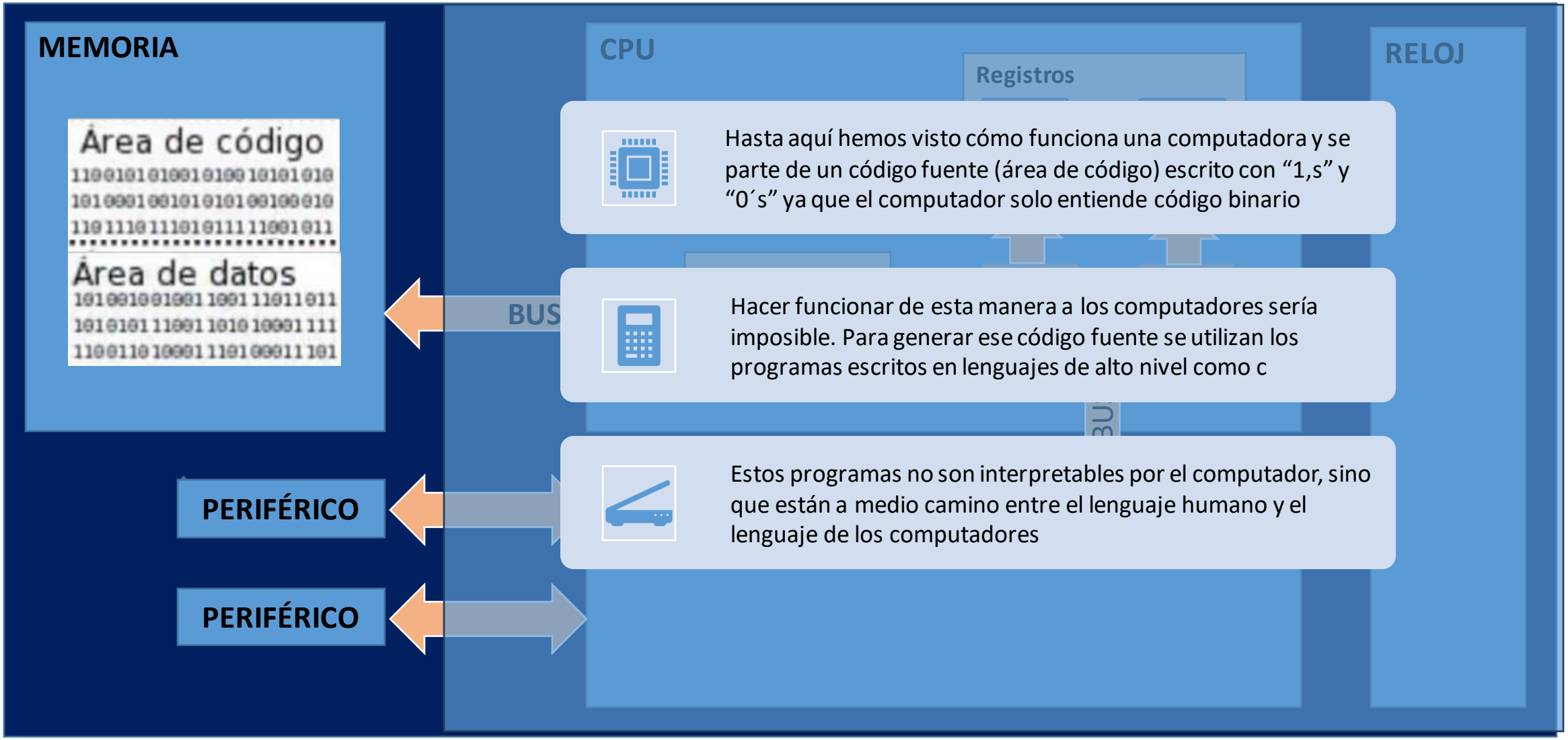
Historia de la Informática

Arquitectura Von Neumann.



Historia de la Informática

Arquitectura Von Neumann.



La Historia de la informática
Los lenguajes de programación: lenguaje C

Nosotros aprenderemos a programar en C!!

- El primer lenguaje de un nivel algo más elevado que el código máquina es el **ensamblador**. Es un lenguaje de bajo nivel. Hace una representación simbólica de los códigos máquina especifico de cada arquitectura.
- En los cincuenta, surgen los tres primeros lenguajes de programación modernos, cuyos descendientes aún continúan siendo utilizados, son:
  - FORTRAN (1955), creado por John Backus (IBM).
  - LISP (1958), creado por John McCarthy (MIT).
  - COBOL (1959), creado por el Short Range Committee, altamente influenciado por Grace Hopper.
- C es un lenguaje de programación de propósito general originalmente desarrollado por Dennis Ritchie y con la colaboración de Brian Kernighan, entre 1969 y 1972 en los Laboratorios Bell.
- C es el lenguaje utilizado en el desarrollo de los Kernels de los sistemas operativos GNU/Linux, Windows y MacOS.
- Es ampliamente usado en dispositivos embebidos.
- C es el "inspirador" de muchos lenguajes de alto nivel disponibles en la actualidad: Perl, PHP, Python y Ruby,..
- Además, es un lenguaje de referencia dentro del mundo de la ingeniería y es comúnmente enseñado en los primeros cursos de estos estudios y nunca ha dejado de ser utilizado.

La Historia de la informática

Los lenguajes de programación: lenguaje C

Hasta aquí hemos visto cómo funciona una computadora. Todos está basado en elcódigo fuente (área de código) escrito con “1,s” y “0´s” ya que el computador solo entiende código binario

Hacer funcionar de esta manera a los computadores sería muy complicado. Para generar ese código fuente se utilizan los programas escritos en lenguajes de alto nivel como c

Estos programas no son interpretables por el computador, sino que están a medio camino entre Encinas lenguaje humano y el lenguaje de los computadores

Lenguaje de alto nivel

```c
#include<stdio.h>
main()
{
    int i;

    /* function returning the max between two numbers */
    int max(int num1, int num2) {

       /* local variable declaration */
       int result;

       if (num1 > num2)
          result = num1;
       else
          result = num2;

       return result;
    }


//    for (i=0;i<10;i++)
//       printf("Hello World! %d\n",i);
    printf("%d",max(3,8));
}
```

Compilador

Programa traductor

Código maquina

```
0000: 4B C7 E0 9B 0D 98 2F FD B7 48 84 B2
000C: 6A F9 47 5E 05 09 FE E6 60 F0 E6 06
0018: 7A B4 83 3A 53 EB 96 B5 55 E9 EB 69
0024: 54 09 BC 11 FB 6C FF 59 15 E7 A1 C9
0030: DB DE 35 BC 0B 08 45 A7 94 60 B8 18
003C: 22 5B E8 50 65 FB E2 91 DC F1 4B 3F
0048: 9B AF B7 9F 31 35 C1 08 16 A3 35 D0
0054: 23 D5 EB B2 B0 B9 12 A3 00 E9 14 CD
0060: 89 B0 64 23 6A E4 78 8E 80 1C 42 38
006C: E9 2F 44 C2 77 AC 5A CA B4 B2 7B DB
```

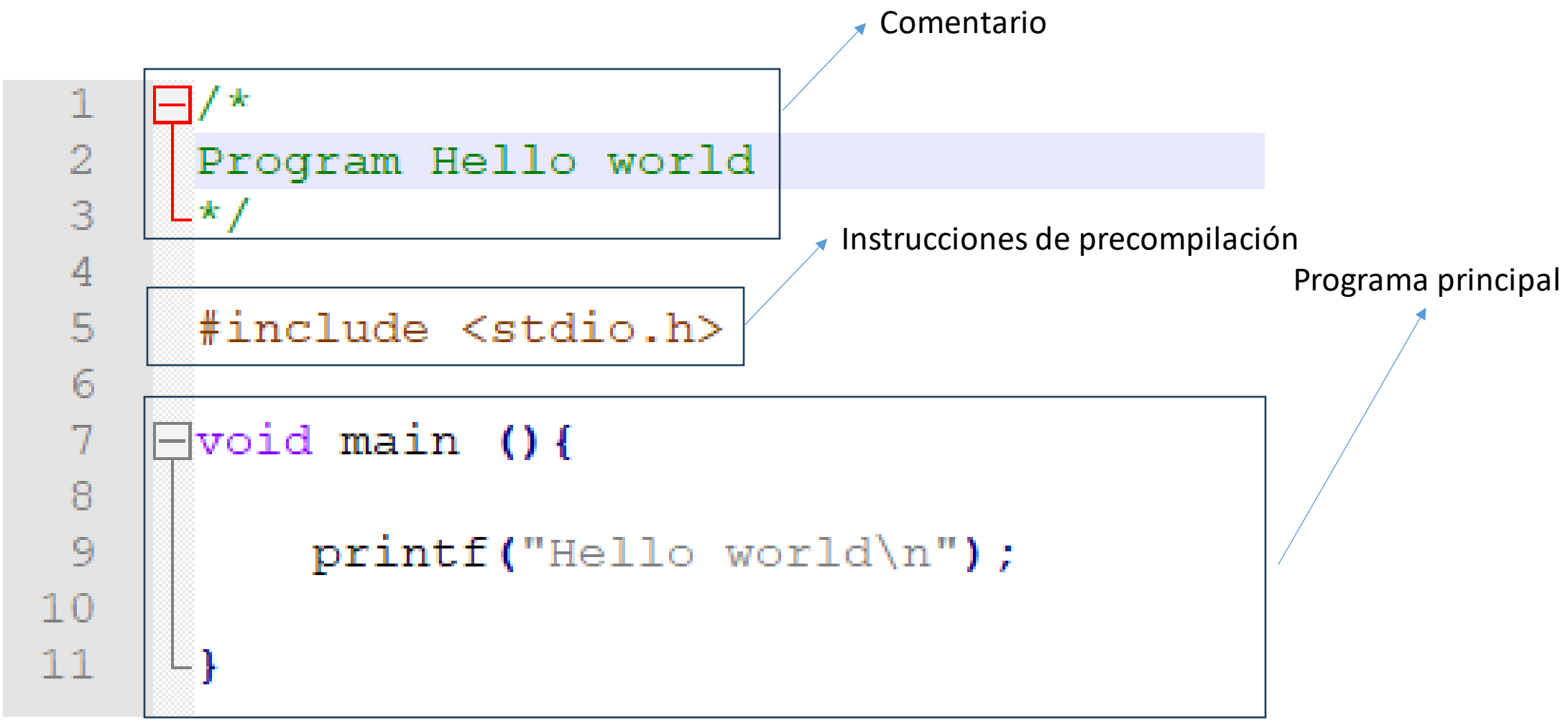
Introducción

Los lenguajes de programación: lenguaje C

- La creación de un ejecutable de un programa en C sigue los siguientes 3 pasos:
  - Generación de las líneas de código (construcción o codificación del programa).
  - Compilación del programa
  - Enlazado del programa con las funciones de biblioteca que necesita
- Herramientas
  - Entorno Cygwin
    - Descarga de paquete preparado
    - Compilador basado en gcc
  - Editor Notepad ++


```
$ gcc HelloWorld.c -o HelloWorld.exe
```

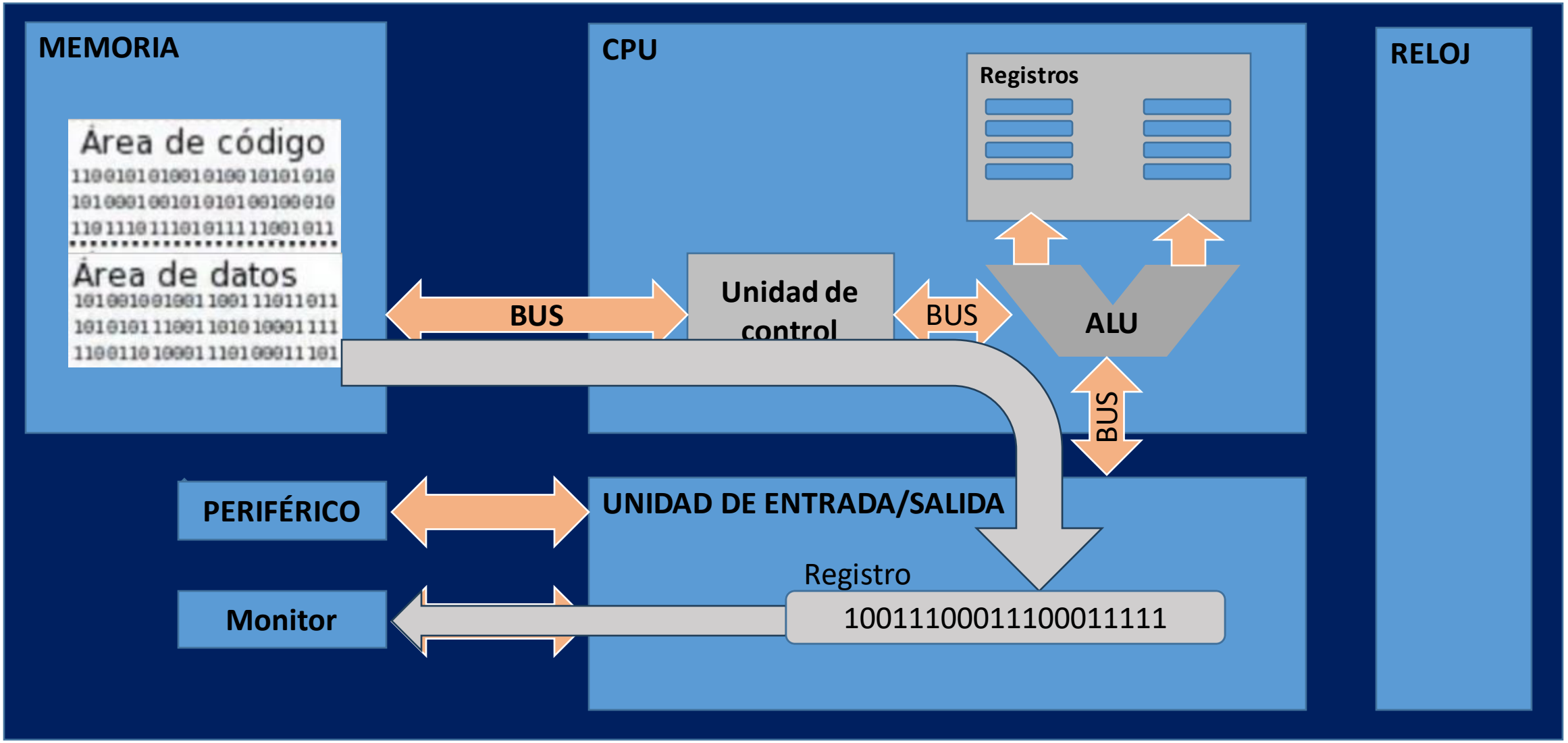
El primer programa en C



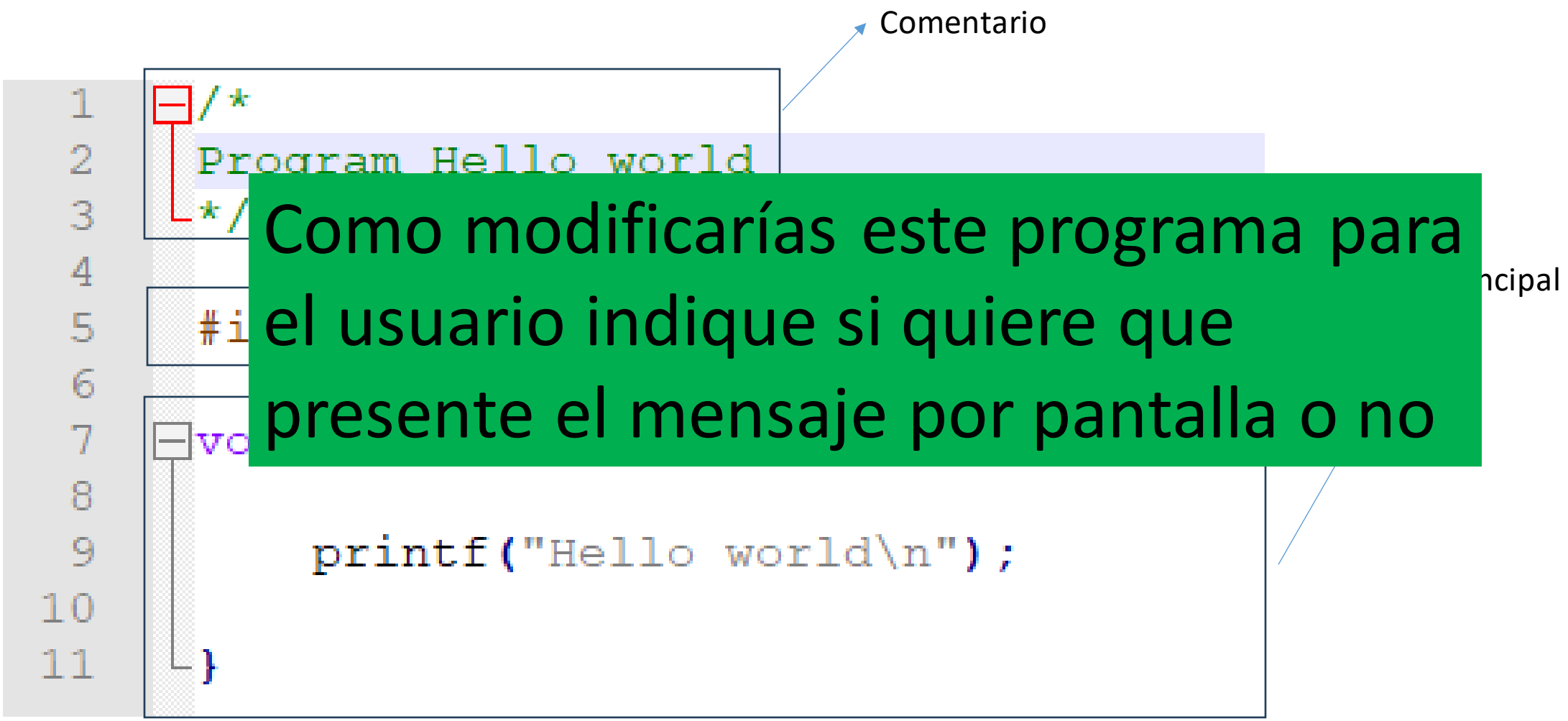
Historia de la Informática

printf(“Ingresa un caracter”);

Mi primer programa.



El primer programa en C



Como modificarías este programa para el usuario indique si quiere que presente el mensaje por pantalla o no

El primer programa en C

```c
/*
Program Hello world avanzado
*/

#include <stdio.h>

void main (){
   char caracter;

     printf("Ingresa un carácter ('S' para imprimir 'Hello world'): ");
     scanf("%c", &caracter);

     if (caracter == 'S' || caracter == 's') {
         printf("Hello world\n");
     } else {
         printf("Carácter no válido. No se imprime 'Hello world'\n");
     }
}
```

Instrucciones de precompilación

Programa principal

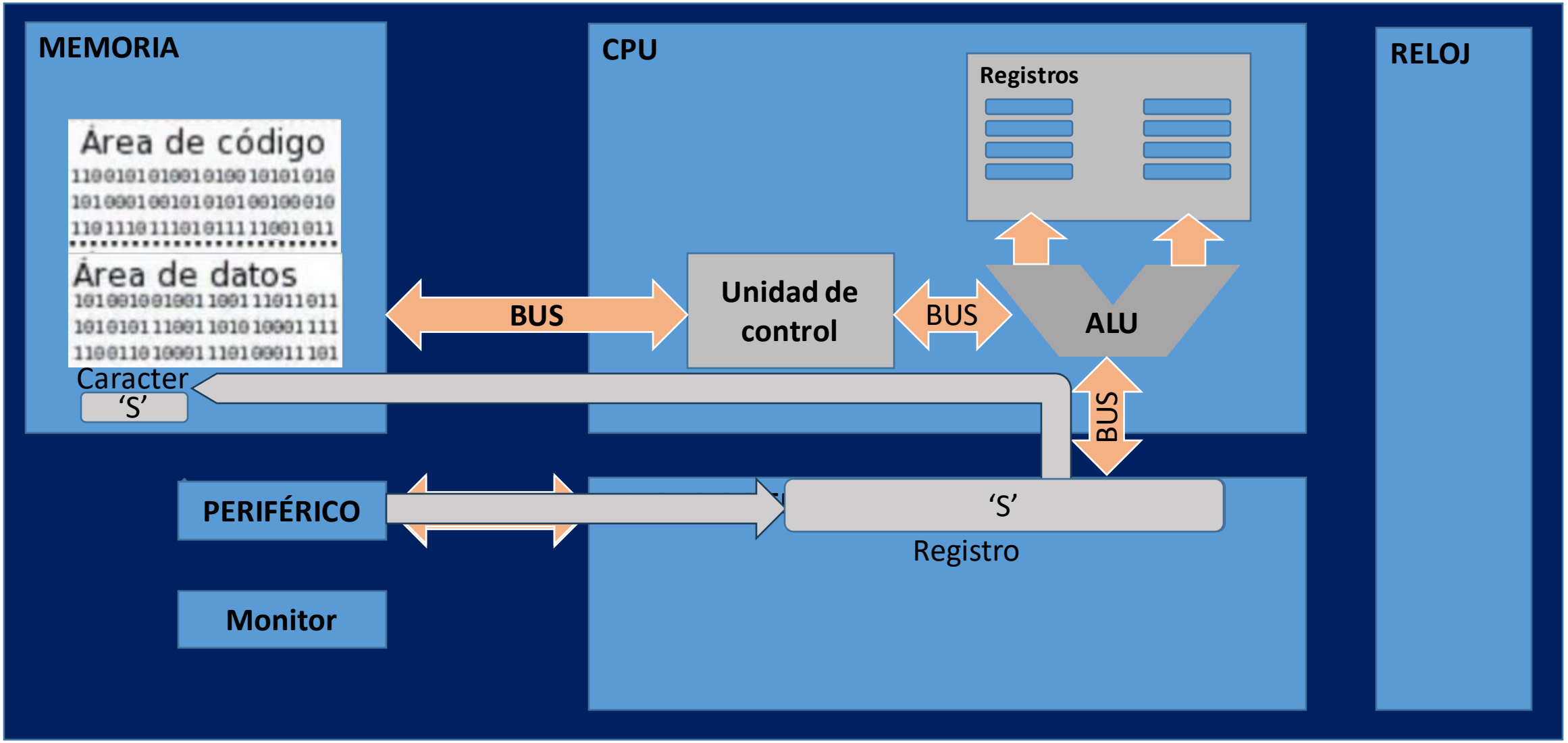
Historia de la Informática

Mi primer programa.

```
scanf("%c",&Caracter);
scanf("%d",&Entero);
```

Para caracteres

Para enteeos



Historia de la Informática

Mi primer programa.

Instrucciones de precompilación

- En este caso la línea comienza por `#include`
- Esto avisa al compilador que tiene que cargar la librería de funciones <stdio.h> (librería estándar de entrada y salida)
  - Antes de nosotros otros programadores han creado programas para que otros los usen y los han almacenado en librerías.
  - Este es el caso de los programas printf y scanf que envían información al periférico monitor y recibe información del periférico teclado.

Programa principal

- Zona de declaración de variables `char Caracter;`
  - Se reservan unas posiciones de memoria para almacenar una variable.
  - La palabra char determina el tipo de variable.
  - En este caso es una variable de tipo carácter. En caso de tener un número entero se utiliza el tipo int (`int Entero;`)
- Cuerpo del main
  - Está formado por el conjunto de instrucciones que siguen un flujo de programas .
  - En este caso es un flujo condicional Si ocurre algo ejecuto alguna acción y si no ejecuto otra acción. (`if … else`)

Historia de la Informática

Mi primer programa. Estructuras de control

- **Las estructuras de control sirven para especificar el orden en que se ejecutarán las distintas instrucciones de un algoritmo.**

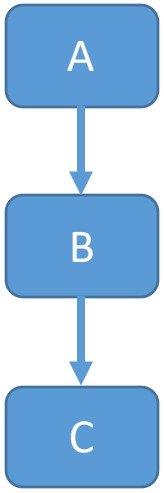
- Este orden de ejecución determina el flujo de control del programa.

- La programación estructurada hace los programas más fáciles de escribir, verificar, leer y mantener, utilizando un número limitado de estructuras de control que minimizan la complejidad de los problemas.

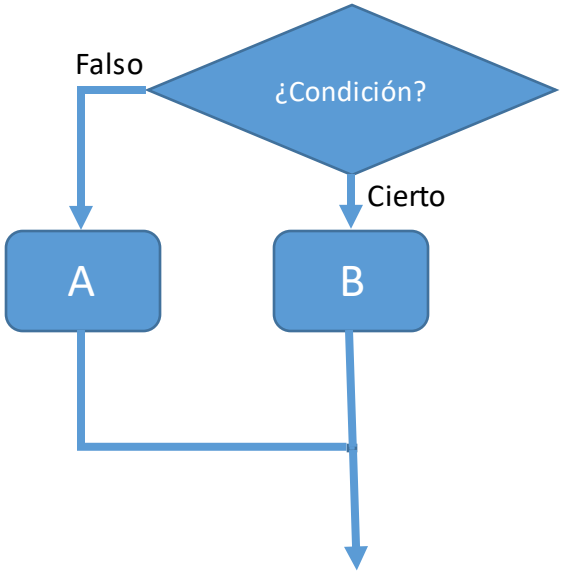
Historia de la Informática

Mi primer programa. Estructuras de control

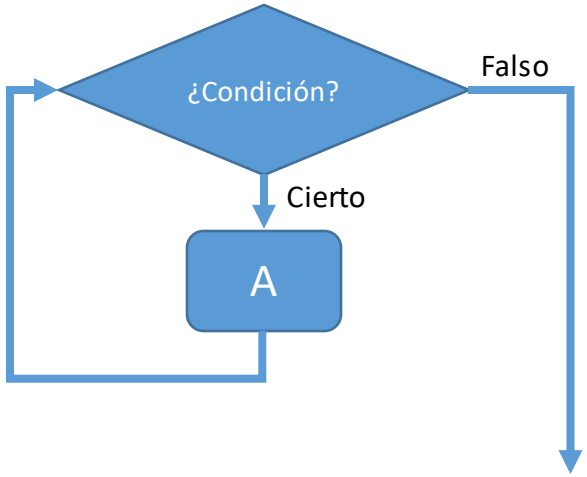
Secuencial



Alternativa o selección



Iteración o Repetitiva



Historia de la Informática

Mi primer programa. Estructuras de control. If con dos alternativas

```
if (caracter == 'S' || caracter == 's') {
    printf("Hello world\n");
} else {
    printf("Carácter no válido. No se imprime 'Hello world'\n");
}
```

- La sentencia **if puede tener dos alternativas** y tiene la siguiente sintaxis:

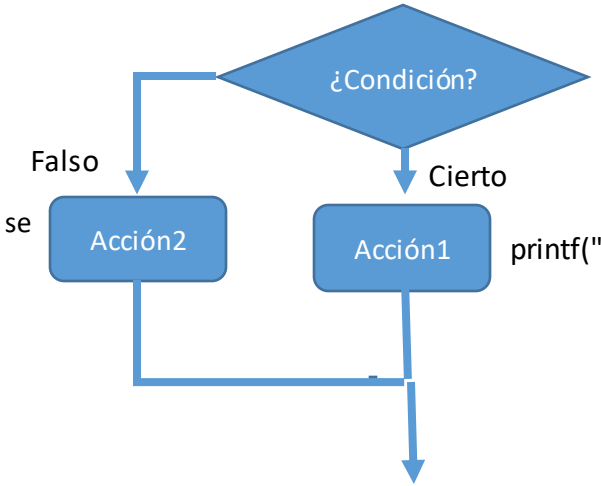
```
if (Condidición)
        Acción1;
else
        Acción2;
```

- Si la `condición` es verdadera se ejecuta la `acción1`, en caso contrario se ejecuta la `acción2`.

- Si la `condición` es una expresión lógica

| Operador lógico OR. Es cierto si se cumple una u otra de las condiciones.|

(caracter == 'S' || caracter == 's')



Historia de la Informática

Mi primer programa. Estructuras de control. Condiciones

Operadores relacionales

```
Expresión_a operador_relacional expresión_b
```

Operador	Significado	Ejemplo
==	Igual a	a == b
!=	No igual a	a != b
>	Mayor que	a > b
<	Menor que	a < b
>=	Mayor o igual que	a >= b
<=	Menor o igual que	a <= b

Operadores lógicos

```
Operando_a operador_relacional Operando_b
```

Símbolo	Significado	Descripción
!	not	Produce falso si su operando es verdadero y viceversa
&&	and	Produce verdadero sólo si ambos operandos son verdaderos. Si cualquiera de los operandos es falso produce falso
||	or	Produce verdadero si cualquiera de los operandos es verdadero y falso solo si ambos operandos son falsos.

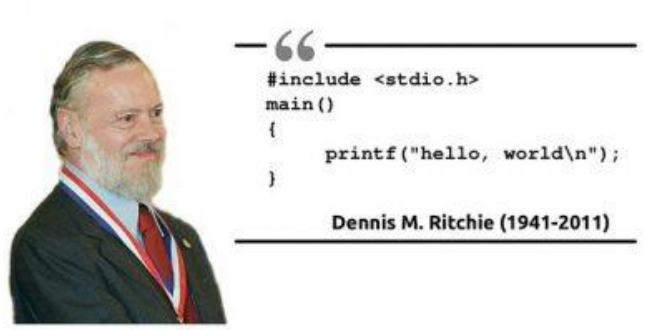
History of Computer Science

C respetuoso con el medio ambiente!

Table 4: Normalized global results for Energy, Time, and Memory

Total					
	Energy (J)		Time (ms)		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

https://haslab.github.io/SAFER/scp21.pdf



https://www.youtube.com/watch?v=g3jOJfrOknA

https://www.youtube.com/watch?v=CYvJPra7Ebk

Historia de la Informática
Ejercicios

- Ejercicio 1:
  - Programa de calificación:
    - El usuario ingresa una nota numérica entre 0 y 9, y el programa debe imprimir la calificación correspondiente, siguiendo el siguiente criterio: suspenso menor a 5, aprobado mayor o igual a 5 y menor a 9 y sobresaliente mayor). Utilizando una estructura if-else
- Ejercicio 2:
  - Programa de comparación de tres números:
    - El usuario ingresa tres números enteros entre el 0 y el 9, y el programa debe determinar e imprimir cuál de los tres números es el mayor, utilizando una estructura if-else:

Introducción
Ejercicios

- Ejercicio 3:
  - Programa de suma o resta:
    - Pedir al usuario que ingrese 2 números y si el primero es mayor o igual que el segundo realiza la resta del primero menos el segundo. En caso contrario realiza la suma. Por último, presenta por pantalla el resultado de la operación que se haya realizado.

Asignatura/Tema