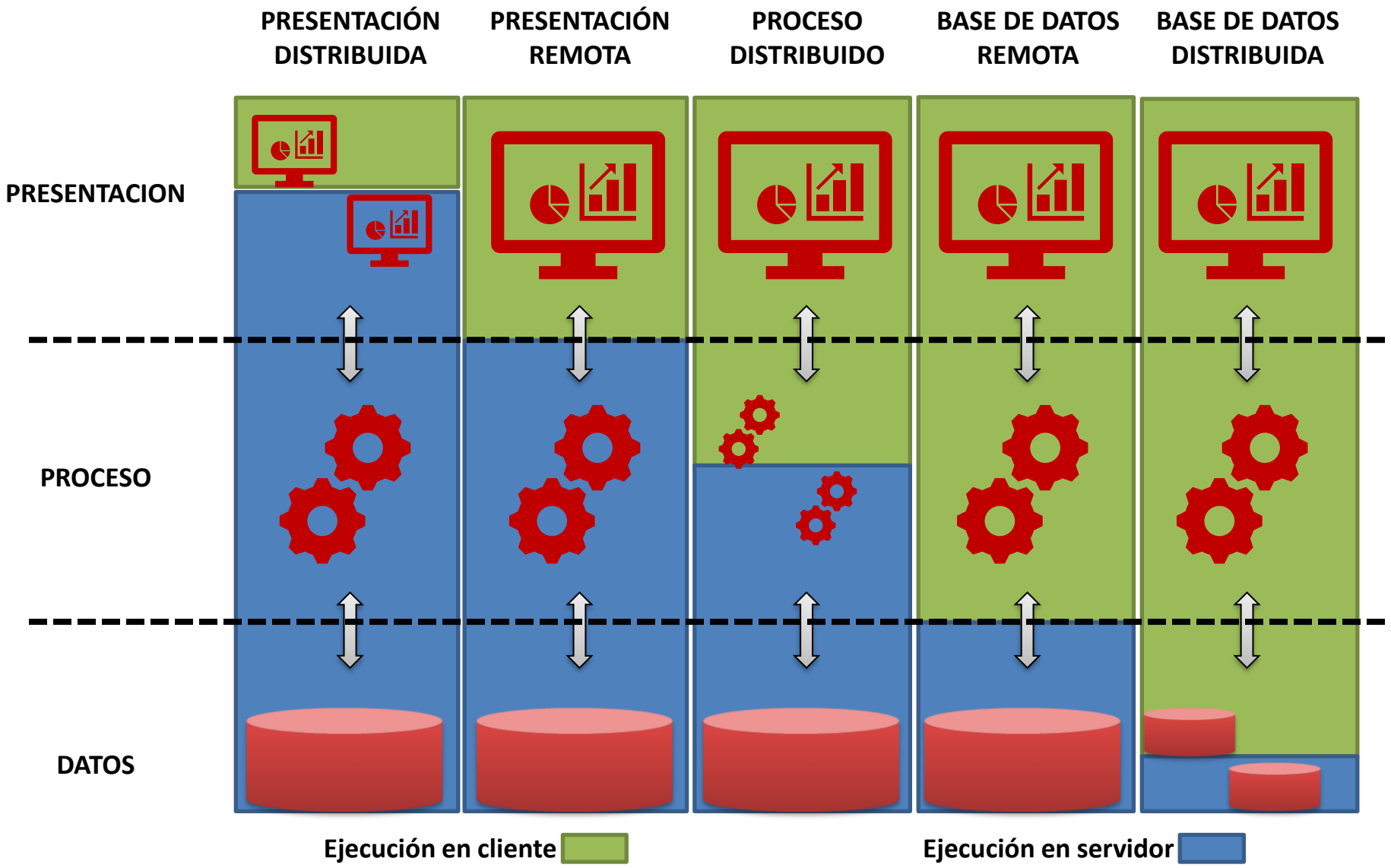
DWEC

Desarrollo Web en Entorno Cliente

ARQUITECTURA CLIENTE SERVIDOR

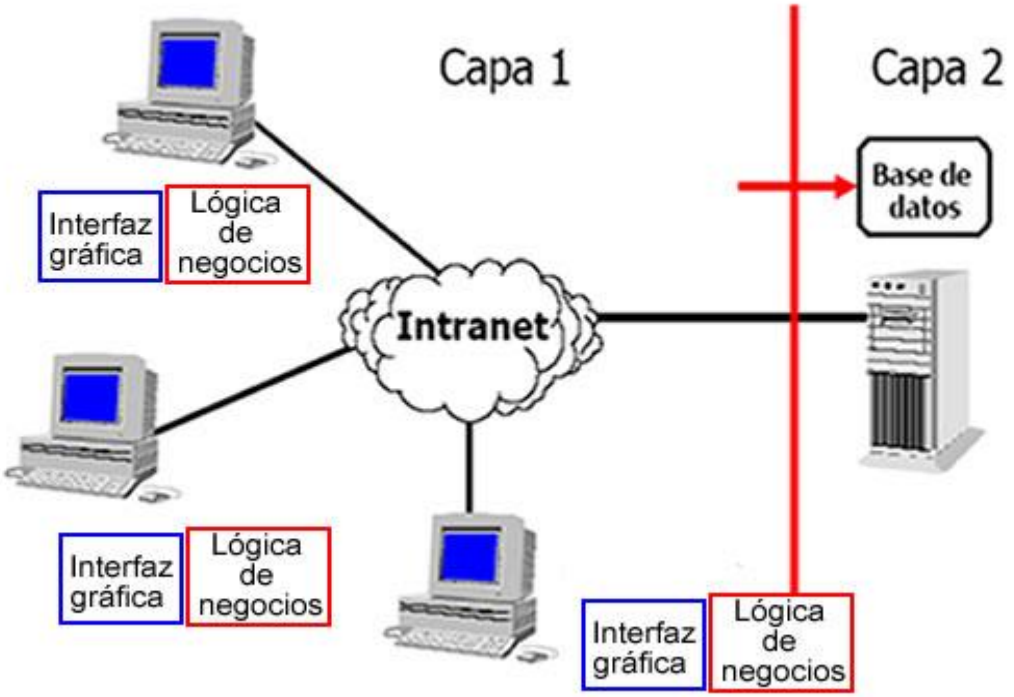
IES Severo Ochoa
2º DAW

TIPOS DE ARQUITECTURAS CLIENTE SERVIDOR



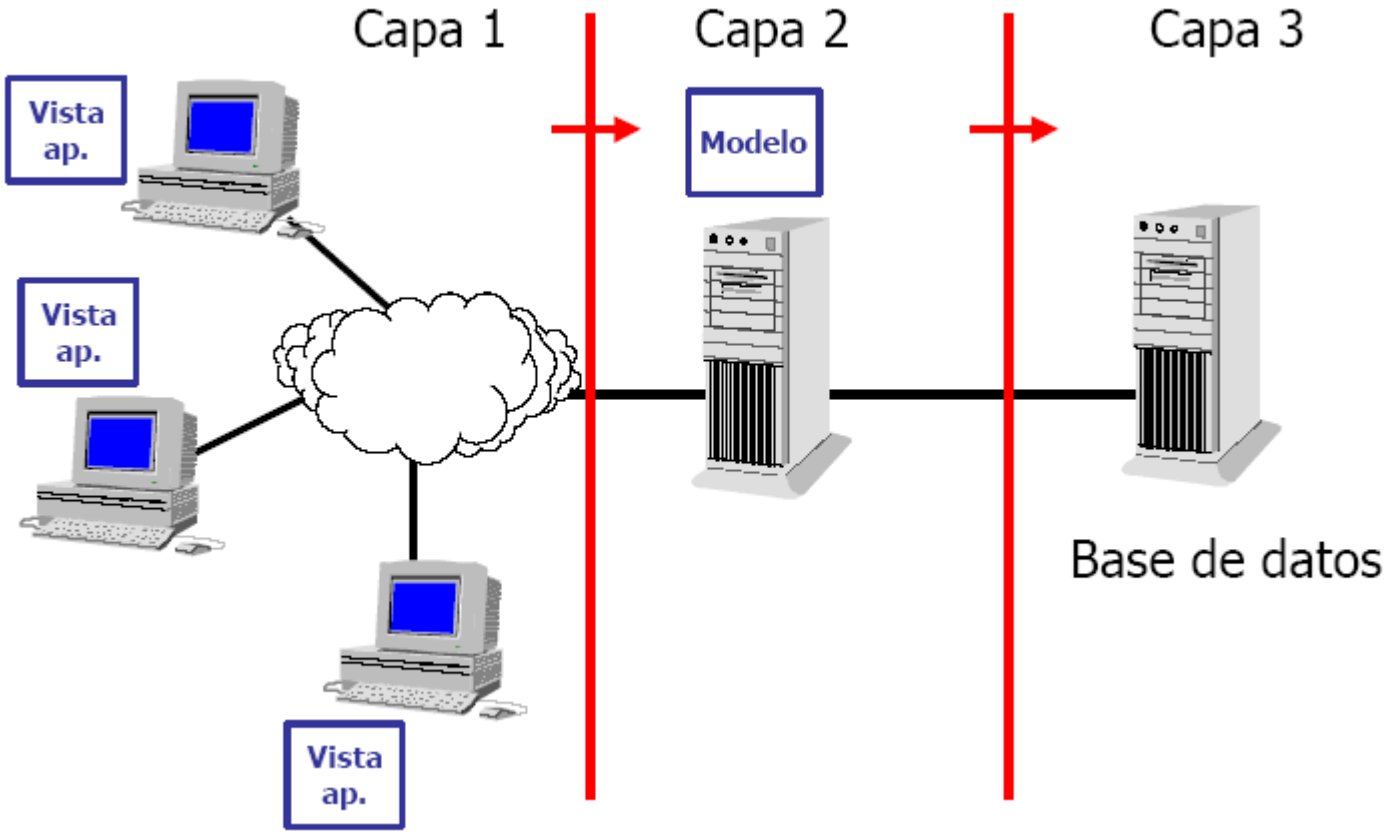
TIPOS DE ARQUITECTURAS CLIENTE SERVIDOR

MODELOS DE 2 CAPAS



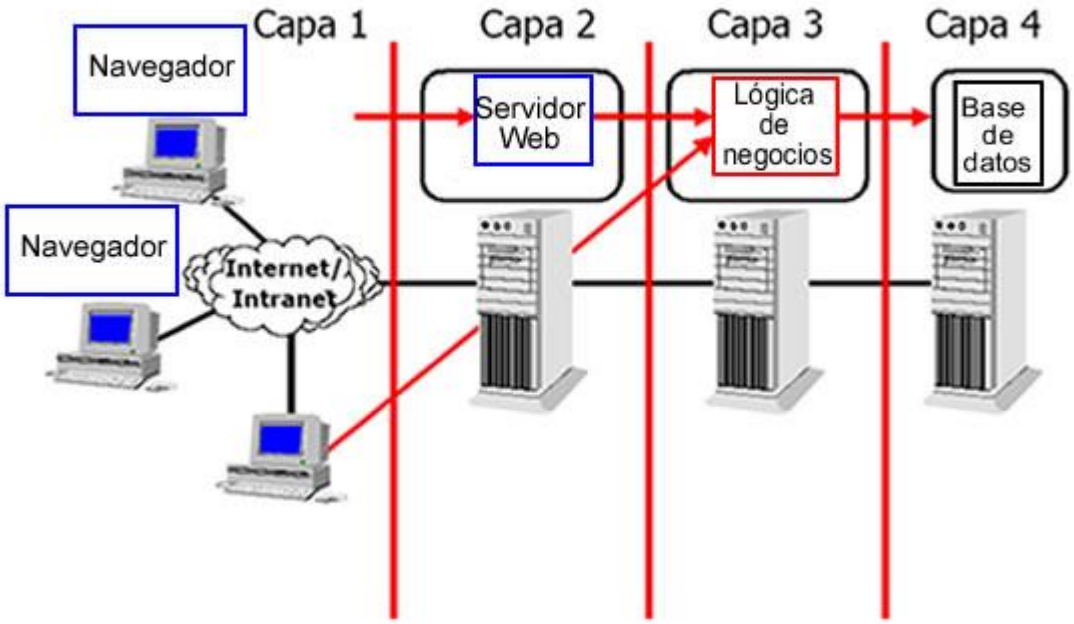
TIPOS DE ARQUITECTURAS CLIENTE SERVIDOR

MODELOS DE 3 CAPAS



TIPOS DE ARQUITECTURAS CLIENTE SERVIDOR

MODELOS DE N CAPAS



DWEC

Desarrollo Web en Entorno Cliente

CODIFICACIÓN DE DISTINTOS TIPOS DE NÚMEROS EN BINARIO

IES Severo Ochoa
2º DAW

BINARIO PURO

USO DE BINARIO PURO: Números naturales (0,1, 2…)

Se usa el binario puro para los números naturales.

Dado un nº binario se puede saber su valor decimal aplicando el **TEOREMA FUNDAMENTAL DE LA NUMERACIÓN:**

$$Vd = \sum_{i=0}^{n-1} b_i * 2^i$$

"El Valor decimal (Vd) de un número binario es igual al sumatorio desde i=0 hasta n del bit de la posición i por la base 2 elevado a la posición i"

Ej.: si tengo el nº binario 10010011 vemos que tendríamos $b_7 b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0$ donde $b_7 = 1$; $b_6 = 0$; $b_5 = 0$; $b_4 = 1$; $b_3 = 0$; $b_2 = 0$; $b_1 = 1$; $b_0 = 1$;

Ahora calcularíamos: $1 * 2^0 + 1 * 2^1 + 0 * 2^2 + 0 * 2^3 + 1 * 2^4 + 0 * 2^5 + 0 * 2^6 + 1 * 2^7$ = 1+2+16+128 =**147**

Como los números binarios crecen muy rápido en longitud, se usa a veces otras bases: Octal y Hexadecimal

Siempre va a existir la limitación del nº de bits de que se disponga. Si se disponen de N bits, se tendrán 2^N combinaciones (o números) distintos.

BINARIO PURO

USO DE BINARIO PURO: Tabla binario, octal, hexadecimal, decimal

BINARIO	OCTAL	HEXADECIMAL	DECIMAL
0000	00	0	0
0001	01	1	1
0010	02	2	2
0011	03	3	3
0100	04	4	4
0101	05	5	5
0110	06	6	6
0111	07	7	7
1000	10	8	8
1001	11	9	9
1010	12	A	10
1011	13	B	11
1100	14	C	12
1101	15	D	13
1110	16	E	14
1111	17	F	15
10000	20	10	16

NÚMEROS CON SIGNO

Representación en SIGNO MAGNITUD

En Signo-Magnitud se usa uno de los bits no como parte del número, si no como signo.

Por ejemplo, en el siguiente caso, vamos a suponer una representación de 8 bits, con signo en el bit más significativo $b_S b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0$

Los bits restantes (de 0 a 6) serán el número en binario puro y el primero será el bit de signo. Lo que se hace es que si bs=0, todo el número será positivo y si bs=1, todo el número será negativo.

Por ejemplo, la cadena de bits 10010011

donde $b_s = 1; por\ tanto\ el\ número\ es\ negativo$

Y la magnitud del número $b_6 = 0; b_5 = 0; b_4 = 1; b_3 = 0; b_2 = 0; b_1 = 1; b_0 = 1;$

Ahora calcularíamos: $1 * 2^0 + 1 * 2^1 + 0 * 2^2 + 0 * 2^3 + 1 * 2^4 + 0 * 2^5 + 0 * 2^6$ = 1+2+16 =19

Pero como todo el nº es negativo, el resultado real es **-19**

El problema con ***signo-magnitud*** es que las operaciones matemáticas son complicadas, y existe una doble representación del 0 (ej.: 00000000 y 10000000)

NÚMEROS CON SIGNO

Representación en EXCESO ZETA

Para evitar algunos problemas, como la doble representación del 0, se usa la codificación en exceso Z.

La clave en este sistema de representación es que yo voy a transformar los números antes de guardarlos o representarlos. Por ejemplo, quiero representar el nº **X**, pues bien, no guardo el **X** directamente, si no que le aplico una función matemática (que básicamente es sumarle un valor fijo **Z**). Y lo que de verdad guardaré será un nº **Y**, que es el resultado de hacer la siguiente operación: **Y= X+ Z**

El valor de **Z** va a depender del nº de bits que tenga: Si tengo n bits, $\boldsymbol{Z = 2^{(n-1)}}$

Por ejemplo, si tengo que el nº de bits es 8: n = 8 bits => $\boldsymbol{Z = 2^{(8-1)} = 2^7 = 128}$

Ejemplo 1
Si tengo una cadena de bits (por ejemplo 10010011) y me dicen que está almacenada en exceso Z hago lo siguiente, calculo su valor (como si fuese binario puro) y me da **147**. Pero 147 no es el nº de verdad que quería guardar si no que es el nº después de haberle sumado Z. Por tanto, si a 147, le resto Z me dará el nº original 147 – 128 = 19, en positivo.

Ejemplo 2
Me dicen que tengo que almacenar el nº -10 en exceso Z de 8 bits. No se puede almacenar -10 tal cuál, hay que sumarle Z. El resultado de -10 + 128 =118, y ahora se pasa 118 a binario => **0 1 1 1 0 1 1 0**

Con **Exceso Z** las operaciones matemáticas siguen siendo complicadas, pero ahora ya no aparece la doble representación cuando quiero almacenar el nº 0.

NÚMEROS CON SIGNO

Representación en COMPLEMENTO A 2 (C2)

Complemento a 2 o C2 es la forma más común de representar números con signo. El 99% de los ordenadores usan esta codificación. Al igual que en los casos anteriores, vamos a estar limitados por el nº de bits que dispongamos. Un número en complemento a dos podrá ser positivo o negativo.

Positivo: Para representar un nº positivo, se pone simplemente en binario puro, teniendo en cuenta que su bit más significativo (el de más a la izquierda) debe ser 0. Los números positivos en C2 empiezan por 0 y los negativos por 1.

Negativo: Se hace una operación matemática (después se verá que esta operación tiene *truco* y simplifica los calculos) y si quiero almacenar un número –X, se hará la siguiente operación $\mathbf{2^N - |X| = Y}$ y será este número **Y** el que guarde.

Por ejemplo, si tengo 8 bits y quiero representar:

a) El nº positivo 5, la representación será: 00000101 (binario puro y 0 a la izq.)

b) El nº negativo -5, la representación será: $\mathbf{2^8 - |-5| = 2^8 - 5 = 256 - 5 = 251}$ que en binario es 11111011

El *truco* indicado anteriormente es que esta operación matemática se puede hacer de una forma más sencilla, si quiero saber cómo es la forma negativa en C2 de un número empiezo con el nº en positivo, por ejemplo el 5:

0 0 0 0 0 1 0 1

1 1 1 1 1 0 1 0

Se hace la *inversión* de los bits, esto es, los ceros se cambian por unos y los unos por ceros

Y al resultado final se le suma un 1

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

NÚMEROS CON SIGNO

Representación en COMPLEMENTO A 2 (C2) *Continuación…*

¿Qué ventaja tiene C2 para que sea el más usado?

En C2 también desaparece la doble representación del 0, pero la principal ventaja es que convierte todas las sumas y restas en sumas, y por extensión, el resto de operaciones matemáticas se pueden reducir a sólo sumas, lo cual simplifica (y abarata) la electrónica interna de la cpu.

Ejemplos (todos con 8 bits)

A. Restar 8 – 5. Es lo mismo que 8 + (-5). Por lo tanto lo primero es pasar -5 a C2 y después se suman ambos números

8: 00001000 =>
5: 00000101 => (inversión) 11111010 => + 1 =>

```
  00001000
  11111011 +
 100000011
```

Este bit se desecha porque es que hace 9 y sólo puede haber 8

el resultado es 3

Vemos que el resultado da un 1 a la izq. pero ese sería el bit 9, y sólo hay 8 bits, por lo que se deshecha

B. Restar 5 – 8. Es lo mismo que 5 + (-8). Por lo tanto lo primero es pasar -8 a C2 y después se suman ambos números

5: 00000101 =>
8: 00001000 => (inversión) 11110111 => + 1 =>

```
  00000101
  11111000 +
  11111101
```

el resultado es la representación en complemento a 2 de -3

Vemos que el resultado da un número negativo, ya que empieza por 1. Calculamos que número es ese y en binario puro es el 253, que resulta de 256 – 3 =253. Por tanto, la operación me ha dado un resultado ya correcto en complemento a 2.

NÚMEROS CON DECIMALES

Representación en COMA FIJA

Simplemente se decide que una posición de los bits irá la coma, los de la izquierda serán la parte entera y los de la derecha la parte decimal. Por ejemplo, el valor de los siguientes números, todos ellos representados con 8 bits:

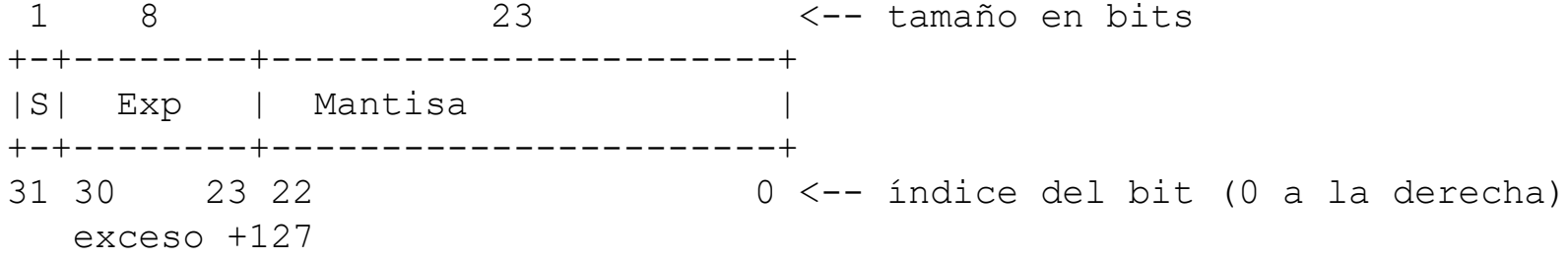
$10101{,}110 = 1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-2} + 0\times2^{-3} = 21{,}75_{10}$
$01001{,}011 = 0\times2^4 + 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2} + 1\times2^{-3} = 9{,}375_{10}$

De los 8 bits hemos fijado y reservado 5 para la parte entera y 3 para la fraccionaria. En los anteriores ejemplos la coma está fija y sirve para separar la parte entera de la parte fraccionaria.

Representación en COMA FLOTANTE

Es como la notación científica. El estándar es el IEEE754.

Coma flotante 32 bits



NÚMEROS CON DECIMALES

Representación en COMA FLOTANTE - EJEMPLO

Codifiquemos el número decimal -118,625 usando el sistema IEEE coma flotante. Necesitamos obtener el signo, el exponente y la fracción. Dado que es un número negativo, el bit de signo es "1".

- Primero, escribimos el número (sin signo, es decir 118,625) usando notación binaria. El resultado es 1110110,101.
- Ahora, movamos la coma decimal a la izquierda, dejando sólo un 1 a su izquierda.
- 1110110,101=1,110110101·26 Esto es un número en coma flotante normalizado.
- El significante es la parte a la derecha de la coma decimal, rellenada con ceros a la derecha hasta que obtengamos todos los 23 bits. Es decir 11011010100000000000000.
- El exponente es 6, pero necesitamos convertirlo a binario y desplazarlo (de forma que el exponente más negativo es 0, y todos los exponentes son solamente números binarios no negativos). Para el formato IEEE coma flotante, el exceso es 127, así es que 6 + 127 = 133. En binario, esto se escribe como 10000101.

Poniendo todo junto:

```
 1     8                23              <-- Tamaño en bits
+-+--------+-----------------------+
|S|  Exp   |     Significante      |
|1|10000101|11011010100000000000000|
+-+--------+-----------------------+
31 30    23 22                     0   <-- índice del bit (0 a la derecha)
   exceso +127
```

CARACTERES

REPRESENTACIÓN DE CARACTERES y TEXTO

A cada elemento del alfabeto, signos, exclamaciones, símbolos, etc. hay que darle una determinada codificación en binario. Existen varios tipos de codificaciones posibles. Las más utilizadas:

- **ASCII (American Standard Code for Information Interchange —Código Estándar Estadounidense para el Intercambio de Información)**
  - 8 bits, hasta 256 caracteres, páginas de códigos para distintos países

- **UNICODE**
  - Unicode incluye todos los caracteres de uso común en la actualidad. La versión 11.0 contiene 137374 caracteres provenientes de alfabetos, sistemas ideográficos y colecciones de símbolos (matemáticos, técnicos, musicales, iconos...). La cifra crece con cada versión.

CARACTERES

TABLA ASCII

Caracteres ASCII de control		
00	NULL	(carácter nulo)
01	SOH	(inicio encabezado)
02	STX	(inicio texto)
03	ETX	(fin de texto)
04	EOT	(fin transmisión)
05	ENQ	(consulta)
06	ACK	(reconocimiento)
07	BEL	(timbre)
08	BS	(retroceso)
09	HT	(tab horizontal)
10	LF	(nueva línea)
11	VT	(tab vertical)
12	FF	(nueva página)
13	CR	(retorno de carro)
14	SO	(desplaza afuera)
15	SI	(desplaza adentro)
16	DLE	(esc.vínculo datos)
17	DC1	(control disp. 1)
18	DC2	(control disp. 2)
19	DC3	(control disp. 3)
20	DC4	(control disp. 4)
21	NAK	(conf. negativa)
22	SYN	(inactividad sínc)
23	ETB	(fin bloque trans)
24	CAN	(cancelar)
25	EM	(fin del medio)
26	SUB	(sustitución)
27	ESC	(escape)
28	FS	(sep. archivos)
29	GS	(sep. grupos)
30	RS	(sep. registros)
31	US	(sep. unidades)
127	DEL	(suprimir)

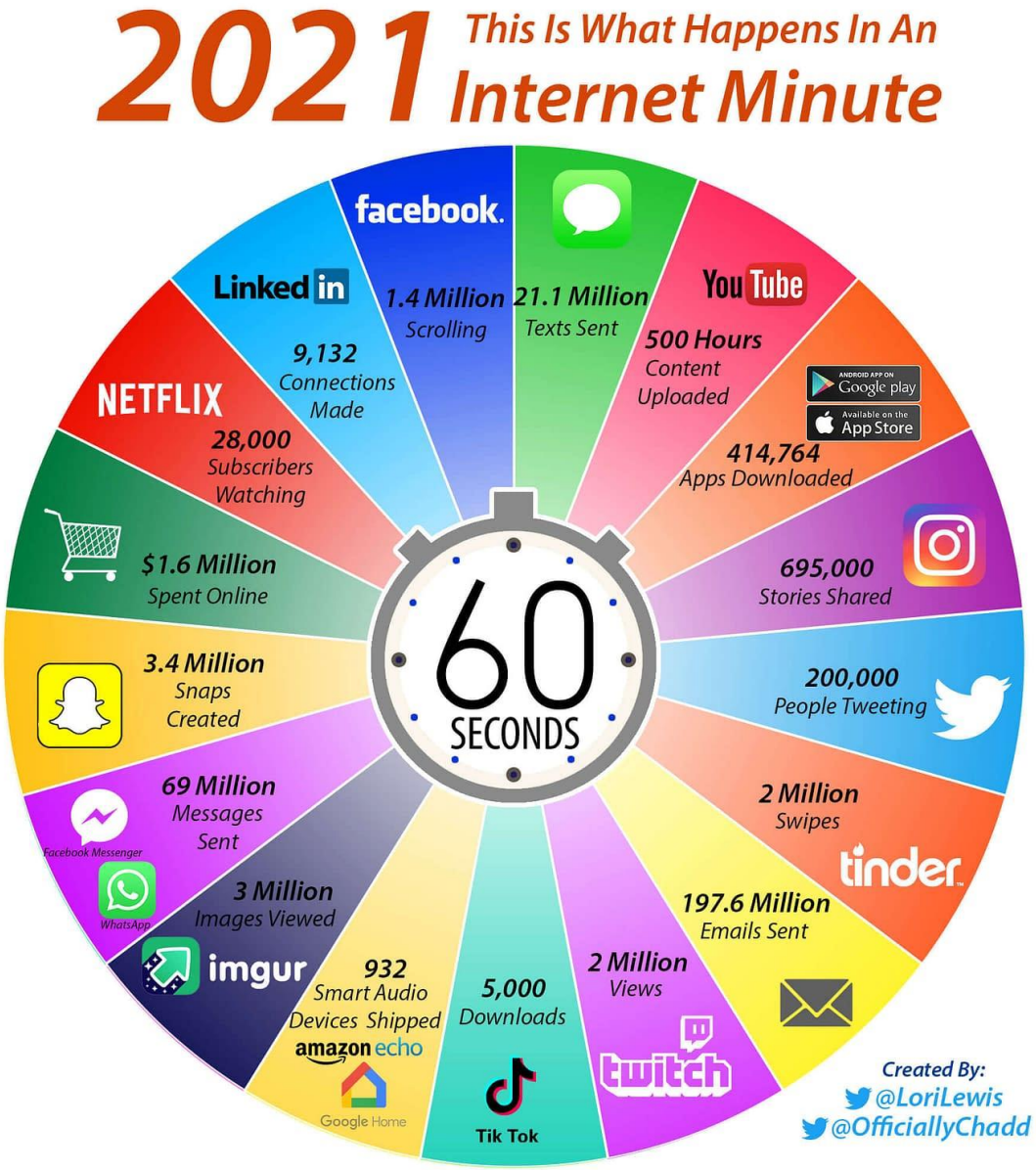
Caracteres ASCII imprimibles					
32	espacio	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

ASCII extendido							
128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

MAGNITUDES INFORMACIÓN DIGITAL

MAGNITUD	SÍMBOLO	EQUIVALENCIA	
Bit	b	Un 1 o un 0	Mínima cantidad de información
Nibble		4 bits	Cuarteto, nº hexadecimal
Byte	B	8 bits	Octeto, un caracter
Kilobyte	KB	1024 B	4 KB una página de texto
Megabyte	MB	1024 KB	1 minuto de música en MP3
Gigabyte	GB	1024 MB	40 fotos de muy alta calidad (profesional)
Terabyte	TB	1024 GB	250 películas en Full HD
Petabyte	PB	1024 TB	Estimación cerebro humano 2,5 PB
Exabyte	EB	1024 PB	Saber de la humanidad 500 EB
Zettabyte	ZB	1024 EB	Peso de Internet 10 ZB (estimación 2018)
Yottabyte	YB	1024 YB	Tráfico de Internet 19 YB (estimación 2018)

MAGNITUDES INFORMACIÓN DIGITAL



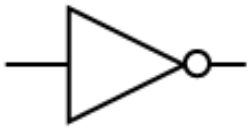
DWEC

Desarrollo Web en Entorno Cliente

OPERACIONES LÓGICAS

IES Severo Ochoa
2º DAW

OPERACIONES LÓGICAS



NOT

Input	Output
I	F
0	1
1	0

AND

Inputs		Output
A	B	F
0	0	0
1	0	0
0	1	0
1	1	1

NAND

Inputs		Output
A	B	F
0	0	1
1	0	1
0	1	1
1	1	0

OR

Inputs		Output
A	B	F
0	0	0
1	0	1
0	1	1
1	1	1

NOR

Inputs		Output
A	B	F
0	0	1
1	0	0
0	1	0
1	1	0

EXCLUSIVE OR

Inputs		Output
A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

EXCLUSIVE NOR

Inputs		Output
A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

OPERACIONES LÓGICAS

Diferencias entre operadores lógicos y operadores de bit

Hay que distinguir entre operadores lógicos y operadores de bits (ejemplo en C y C++)

```
int a=5;
int b=8;
if (a&&b)
        printf (“verdadero”);
else
        printf (“falso”);
```

```
int a=5;
int b=8;
if (a&b)
        printf (“verdadero”);
else
        printf (“falso”);
```

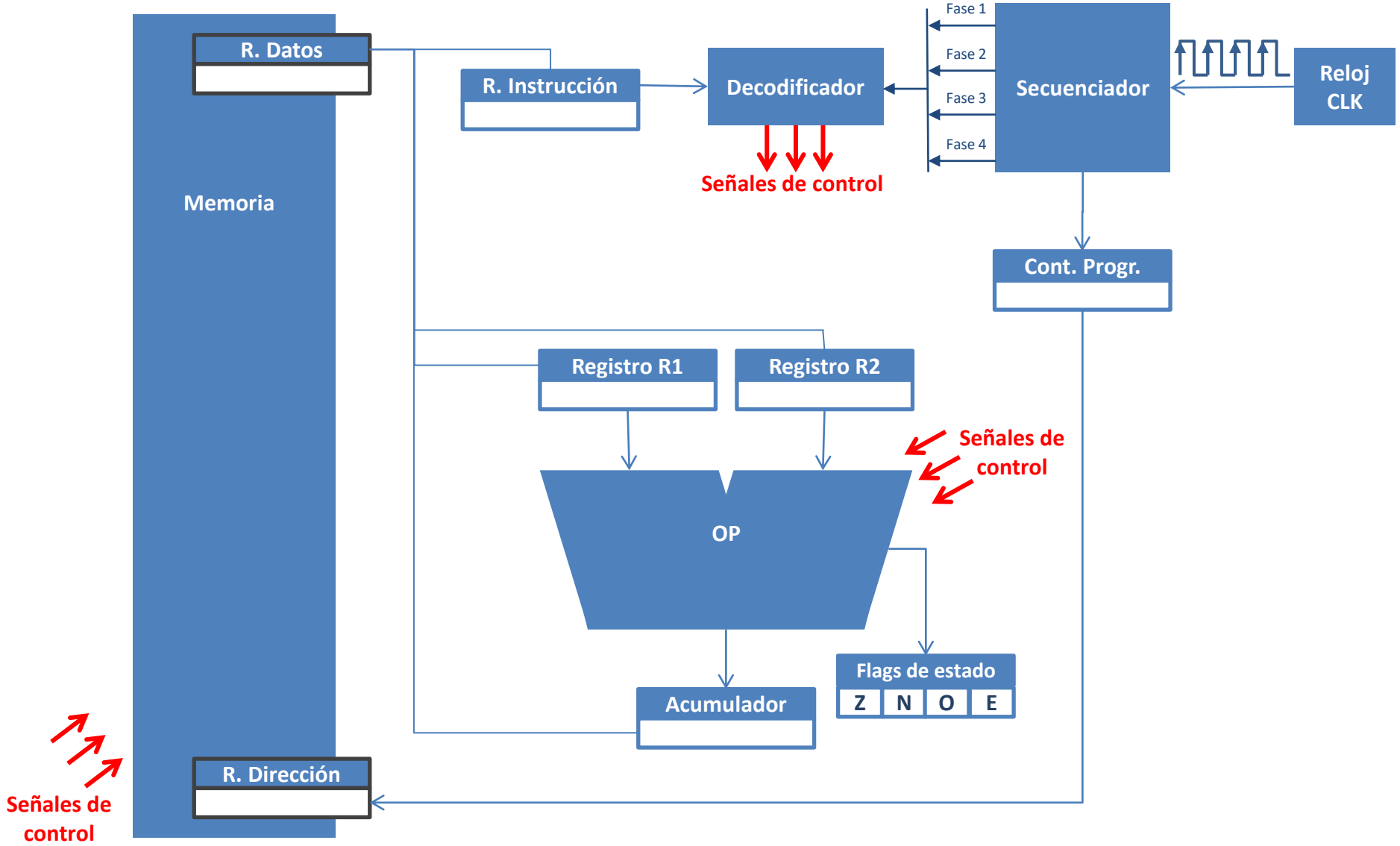
DWEC

Desarrollo Web en Entorno Cliente

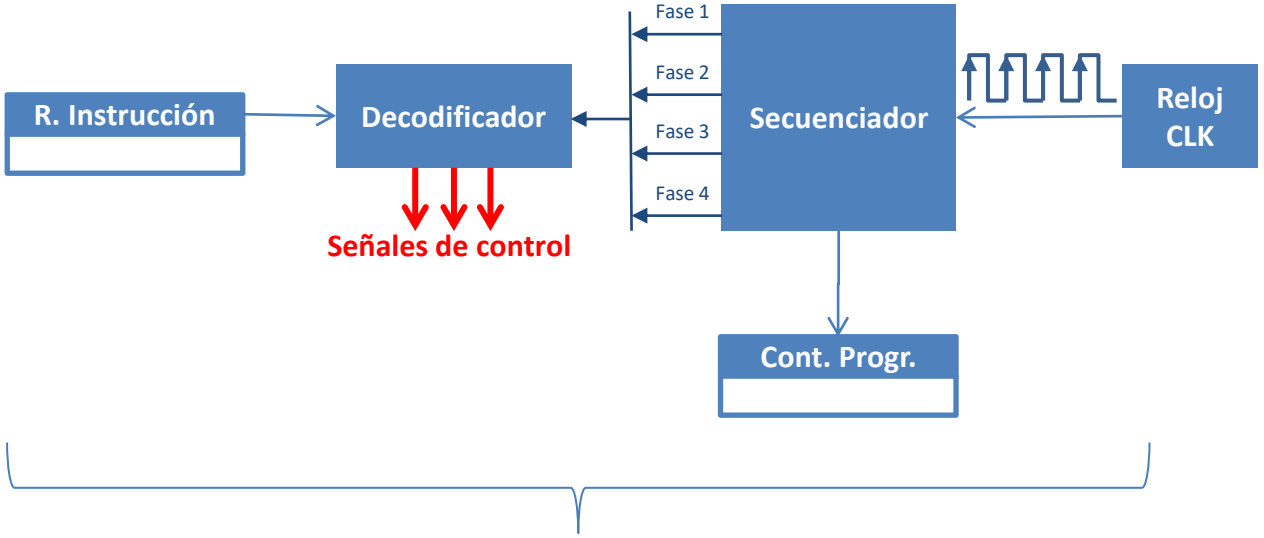
FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

IES Severo Ochoa
2º DAW

FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL



FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL



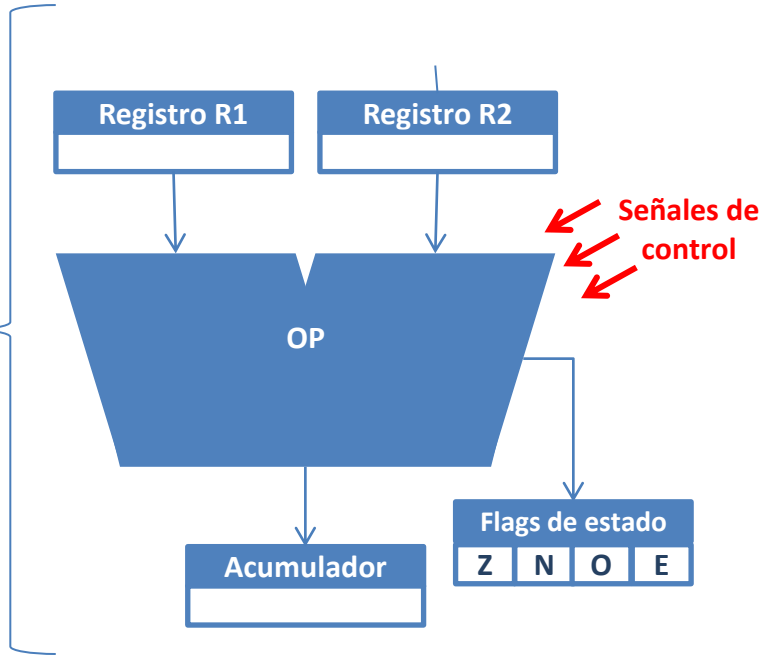
CPU: UNIDAD DE CONTROL

La Unidad de Control o UC es la parte “pensante” del ordenador; es como el director de la orquesta. Se encarga de traer de la memoria RAM las instrucciones necesarias para la ejecución de los programas.

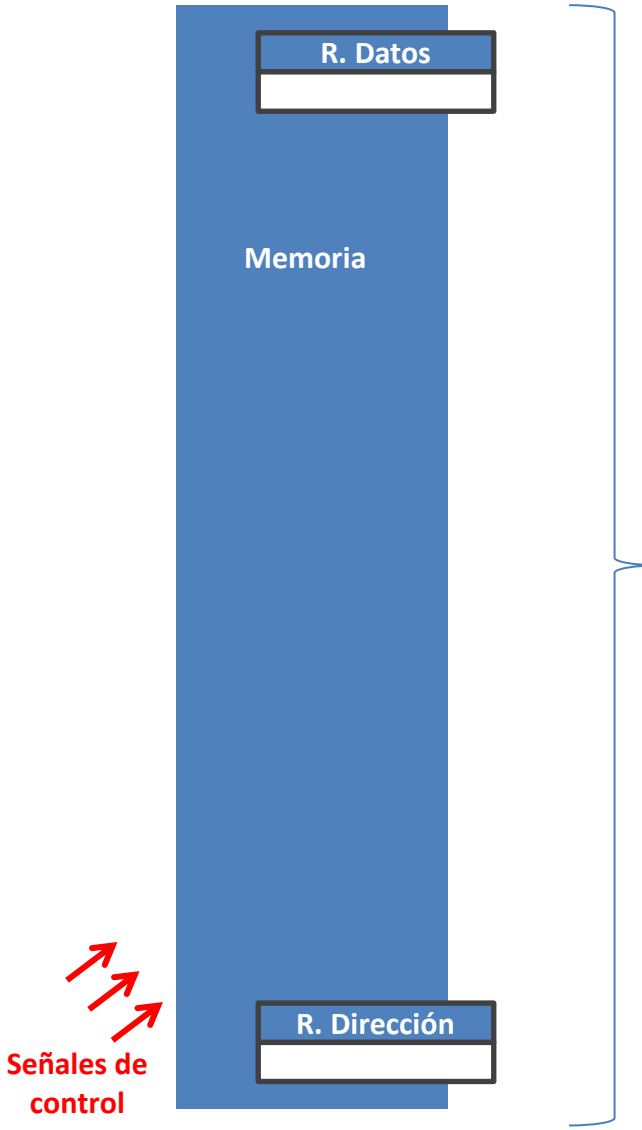
FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

CPU: UNIDAD ARITMETICO LOGICA

La Unidad Aritmético Lógica (o ALU) es la parte de CPU encargada de realizar operaciones aritméticas (sumas, restas, divisiones, multiplicaciones y en general operaciones matemáticas) y operaciones lógicas (operaciones booleanas tales como comparaciones, saltos, si-entonces-sino



FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL



MEMORIA RAM

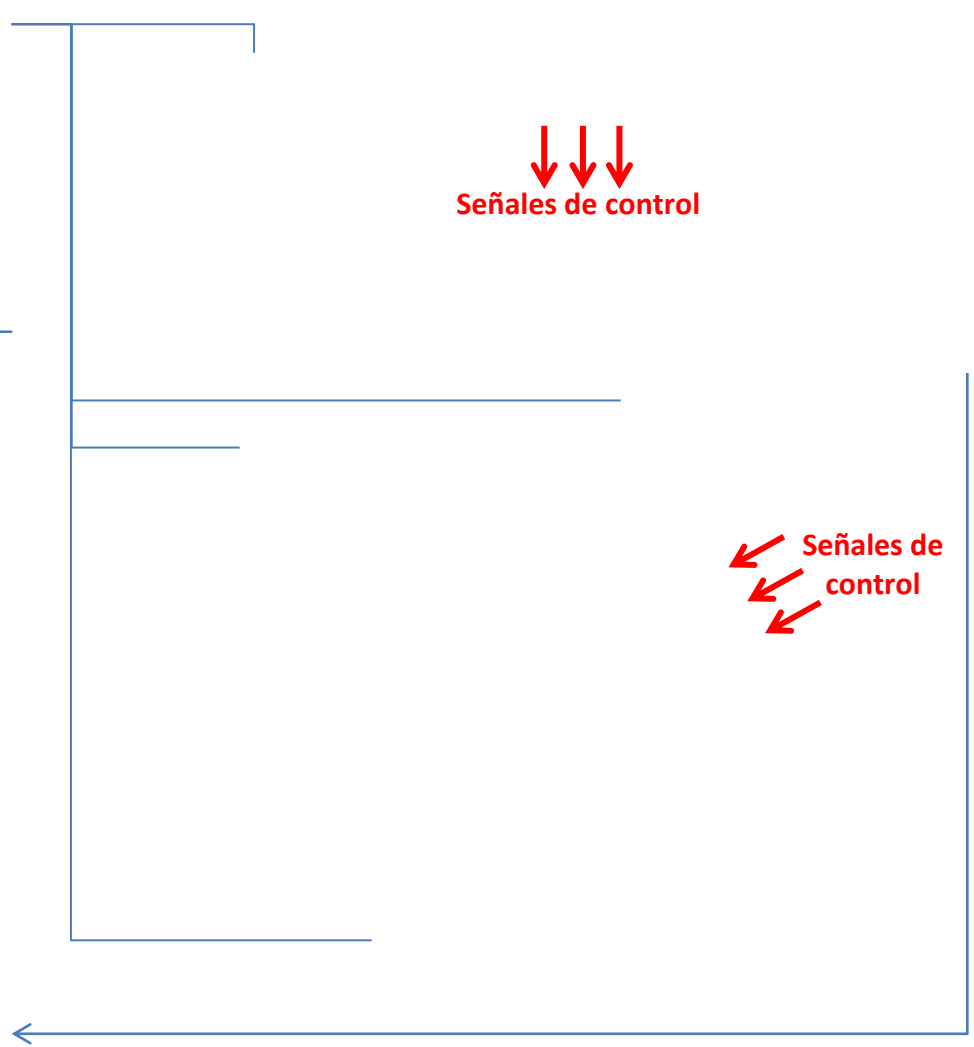
La RAM es el componente que almacena la información del ordenador mientras este se encuentra encendido.

Se podría considerar como un gran número (millones o miles de millones) de celdillas, cada una amacena un valor concreto.

FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

BUSES
de datos y control

En arquitectura de computadores, el bus es un sistema digital que transfiere datos entre los componentes de una computadora. Está formado por cables o pistas en un circuito impreso, dispositivos como resistores y condensadores, además de circuitos integrados.



DWEC

Desarrollo Web en Entorno Cliente

EJECUCIÓN DE UN PROGRAMA

IES Severo Ochoa
2º DAW

FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

EJECUCIÓN DE UN PROGRAMA

Vamos a ejecutar un programa muy simple que va a sumar 2+3 y guardar el resultado en memoria. No se puede ejecutar directamente esta orden, así que habrá que dividirla en pasos más simples. El programa quedaría de la siguiente forma:

Instrucción 1: PONER 2 R1
Instrucción 2: PONER 3 R2
Instrucción 3: SUMAR
Instrucción 4: PONER Ac mX

FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

EJECUCIÓN DE UN PROGRAMA

Realmente las instrucciones están en código máquina (1's y 0's) que de verdad es lo único que entiende el ordenador:

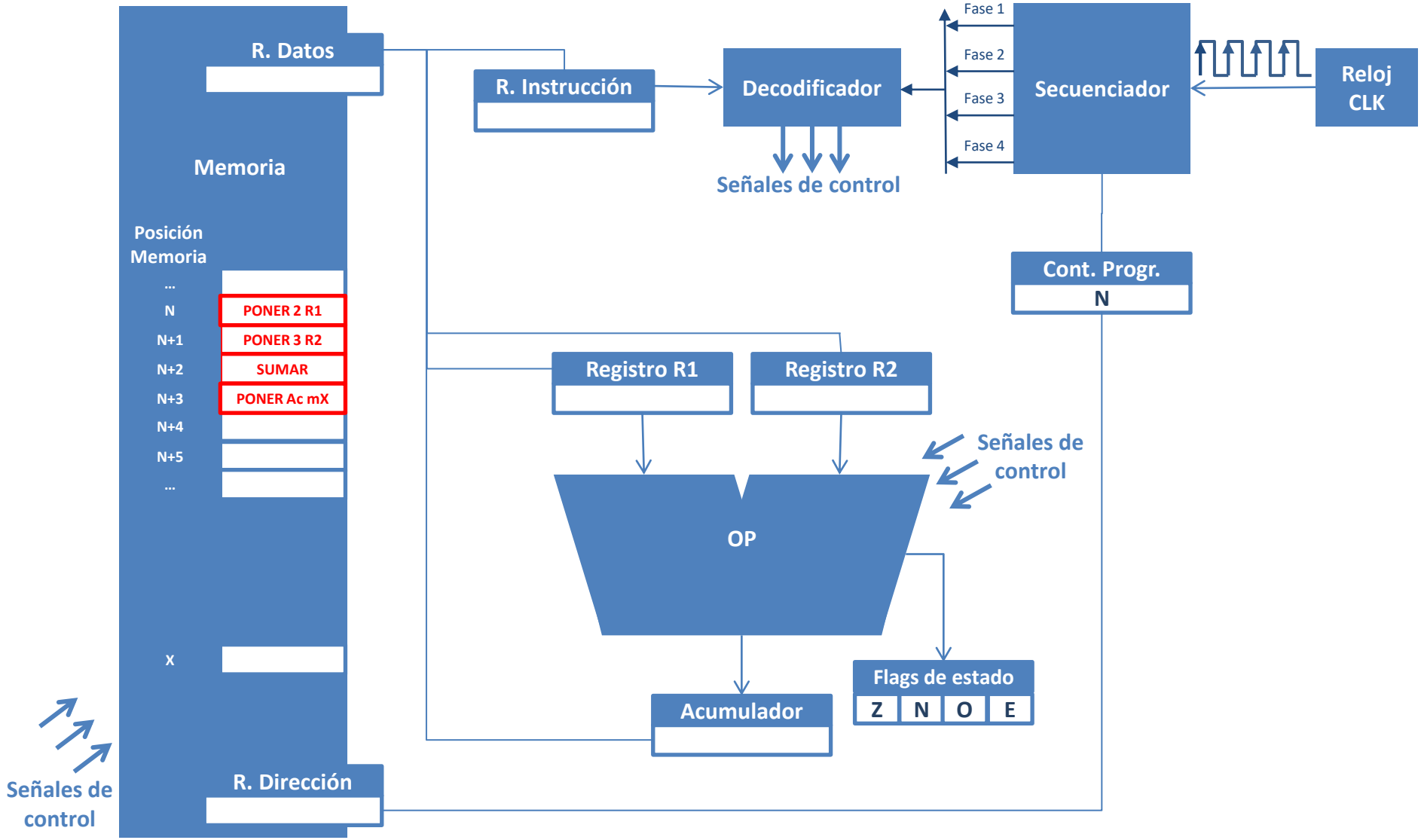
Instrucción 1: 011011	00000010	01
Instrucción 2: 011011	00000011	10
Instrucción 3: 101100	00000000	00
Instrucción 4: 011010	XXXXXXXX	00

Aunque se suele usar ensamblador, que son códigos nemotécnicos que un programa traduce directamente a código máquina:

Instrucción 1: MOV 2 R1
Instrucción 2: MOV 3 R2
Instrucción 3: ADD
Instrucción 4: MOV Ac mX

FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

SITUACIÓN ANTES DE EMPEZAR A EJECUTAR EL PROGRAMA: El programa se encuentra cargado en memoria.



FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

EJECUCIÓN DE INSTRUCCIÓN 1

Instrucción 1: PONER 2 R1

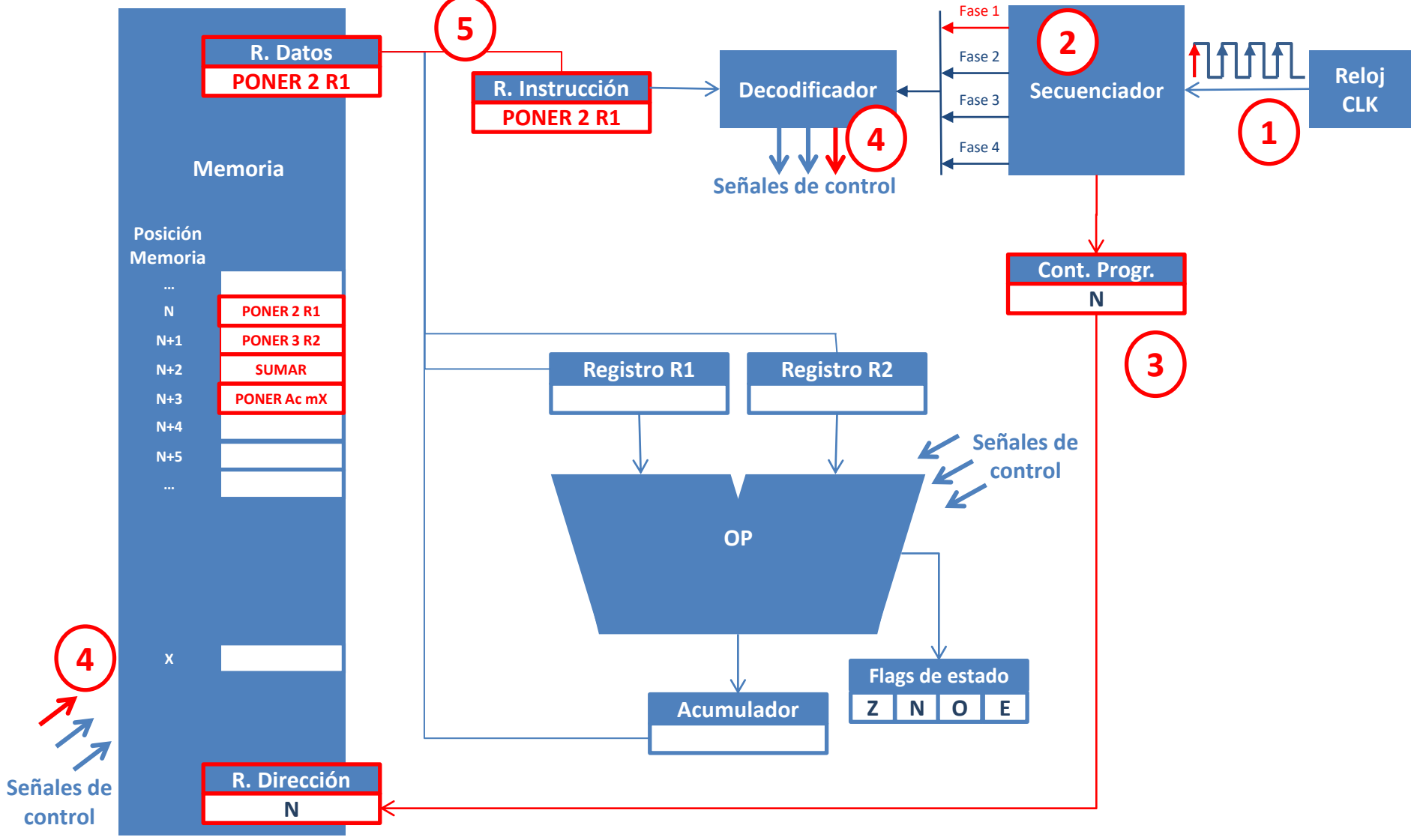
Instrucción 2: PONER 3 R2

Instrucción 3: SUMAR

Instrucción 4: PONER Ac mX

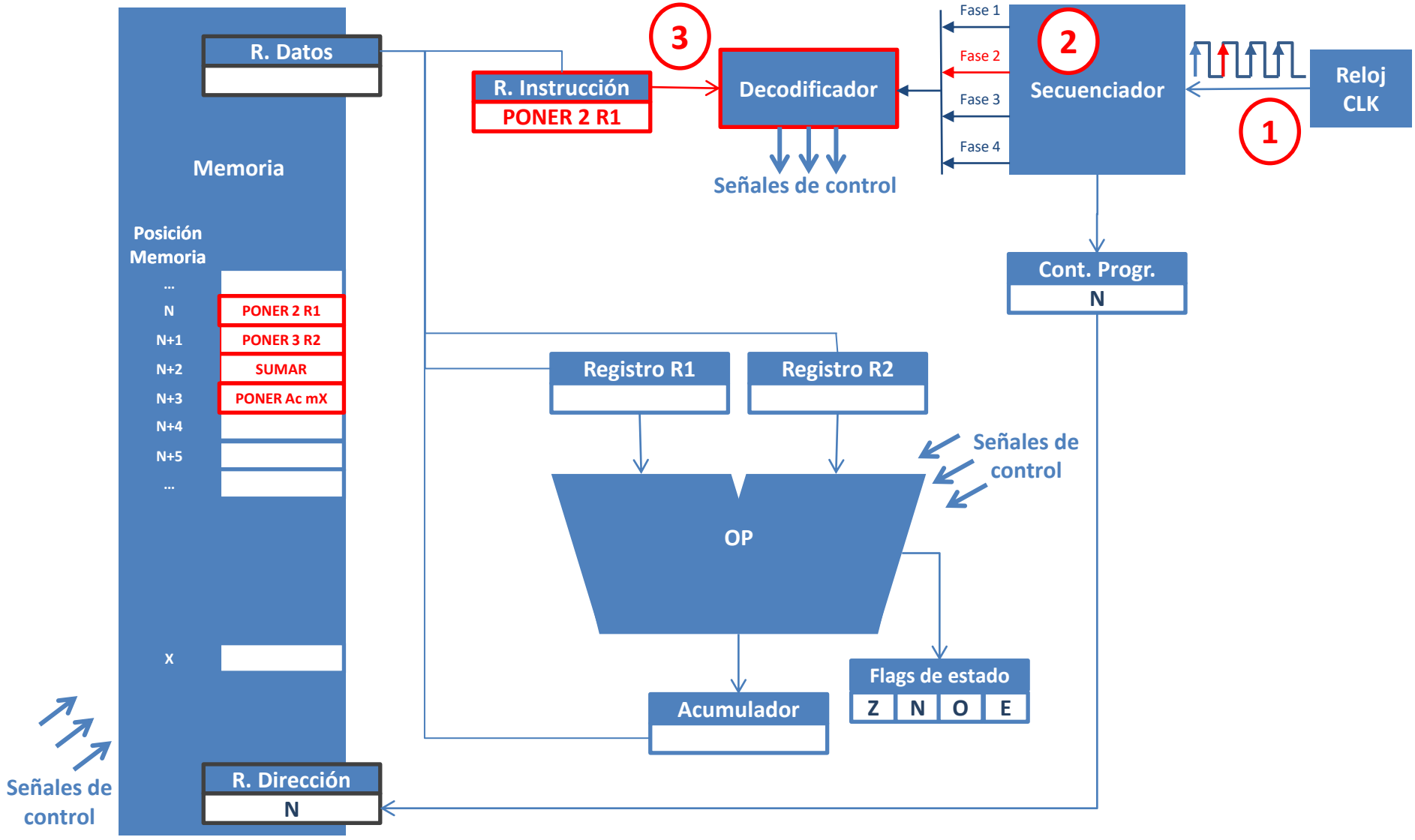
FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

FASE 1: BUSQUEDA DE INSTRUCION: El reloj genera un pulso, el secuenciador genera fase 1, el contador de programa vuelca su contenido en el R.Dirección de memoria, el decodificador da la ordena se seleccionar celda N, el contenido se lleva al registro de instrucción.



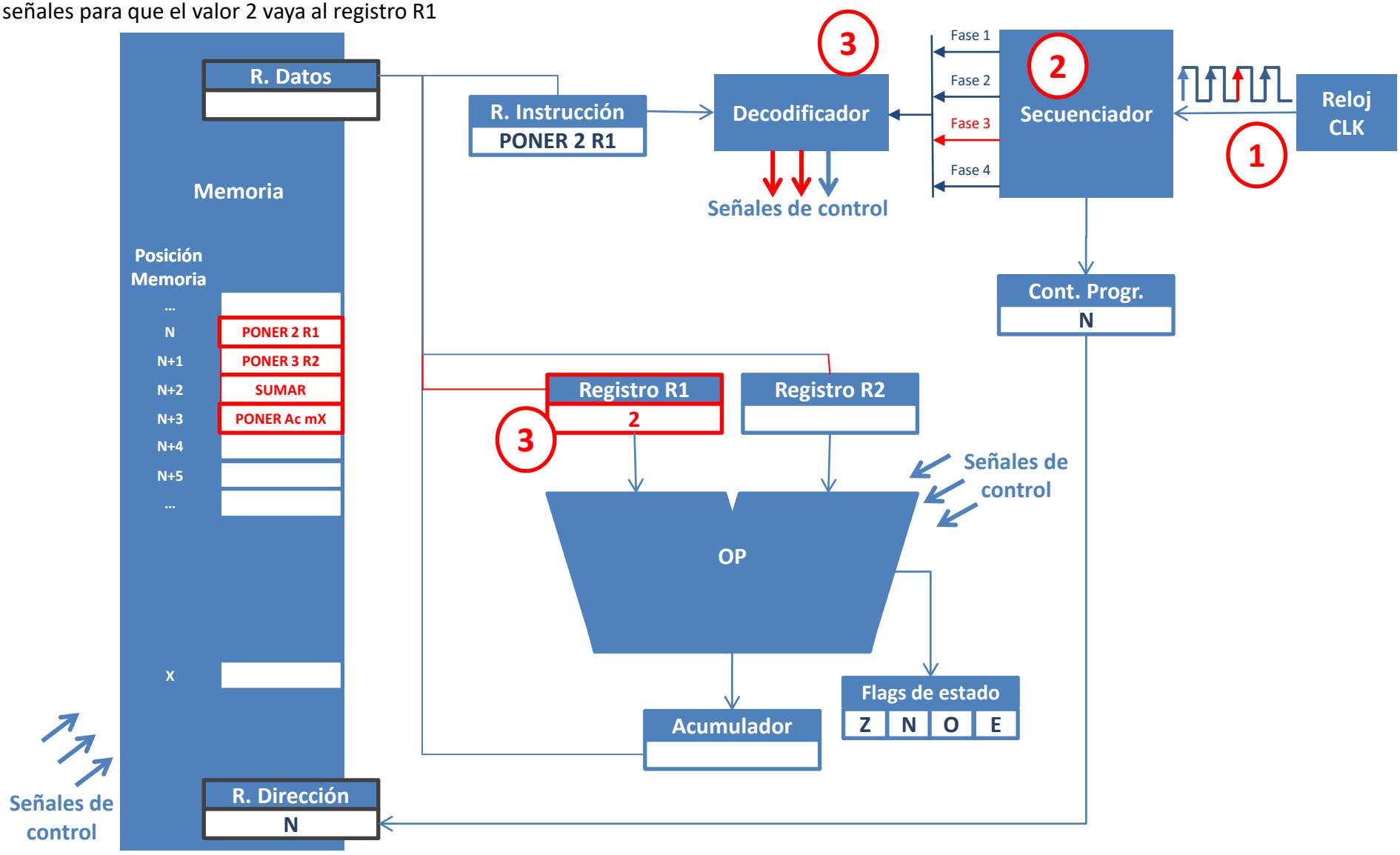
FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

FASE 2: DECODIFICACION DE INSTRUCCIÓN: El reloj genera otro pulso y el secuenciador se pone en fase 2. El decodificador accede al registro de instrucción y decodifica de que instrucción se trata.



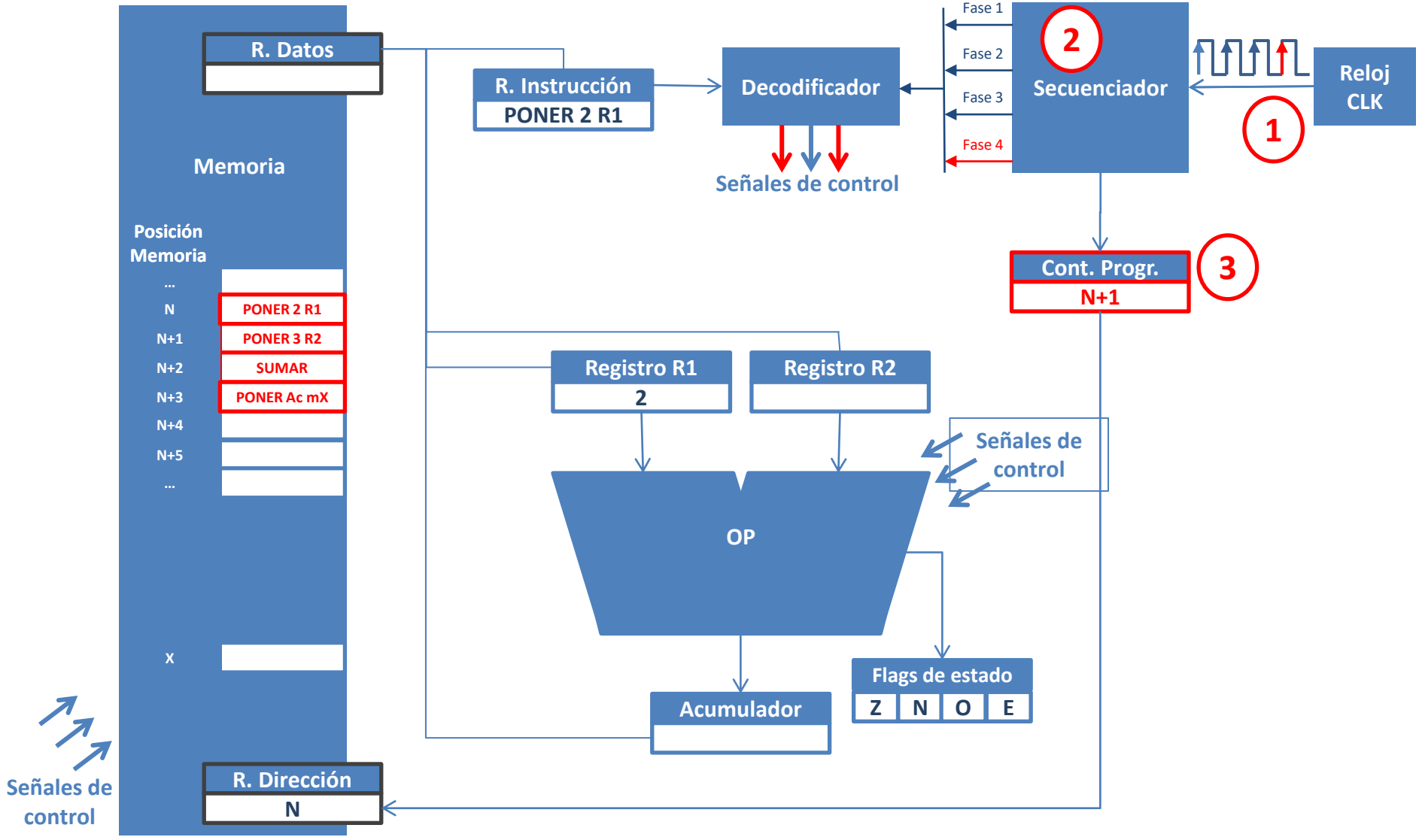
FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

FASE 3: EJECUCIÓN DE INSTRUCCIÓN: El reloj genera otro pulso y el secuenciador se pone en fase 3. El decodificador activa las señales para que el valor 2 vaya al registro R1



FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

FASE 4: SIGUIENTE INSTRUCCIÓN Y ALMACENAMIENTO DE RESULTADOS: El reloj genera el siguiente pulso y el secuenciador se pone en fase 4. Se incrementar en 1 el contador de programa.



FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

EJECUCIÓN DE INSTRUCCIÓN 2

Instrucción 1: PONER 2 R1

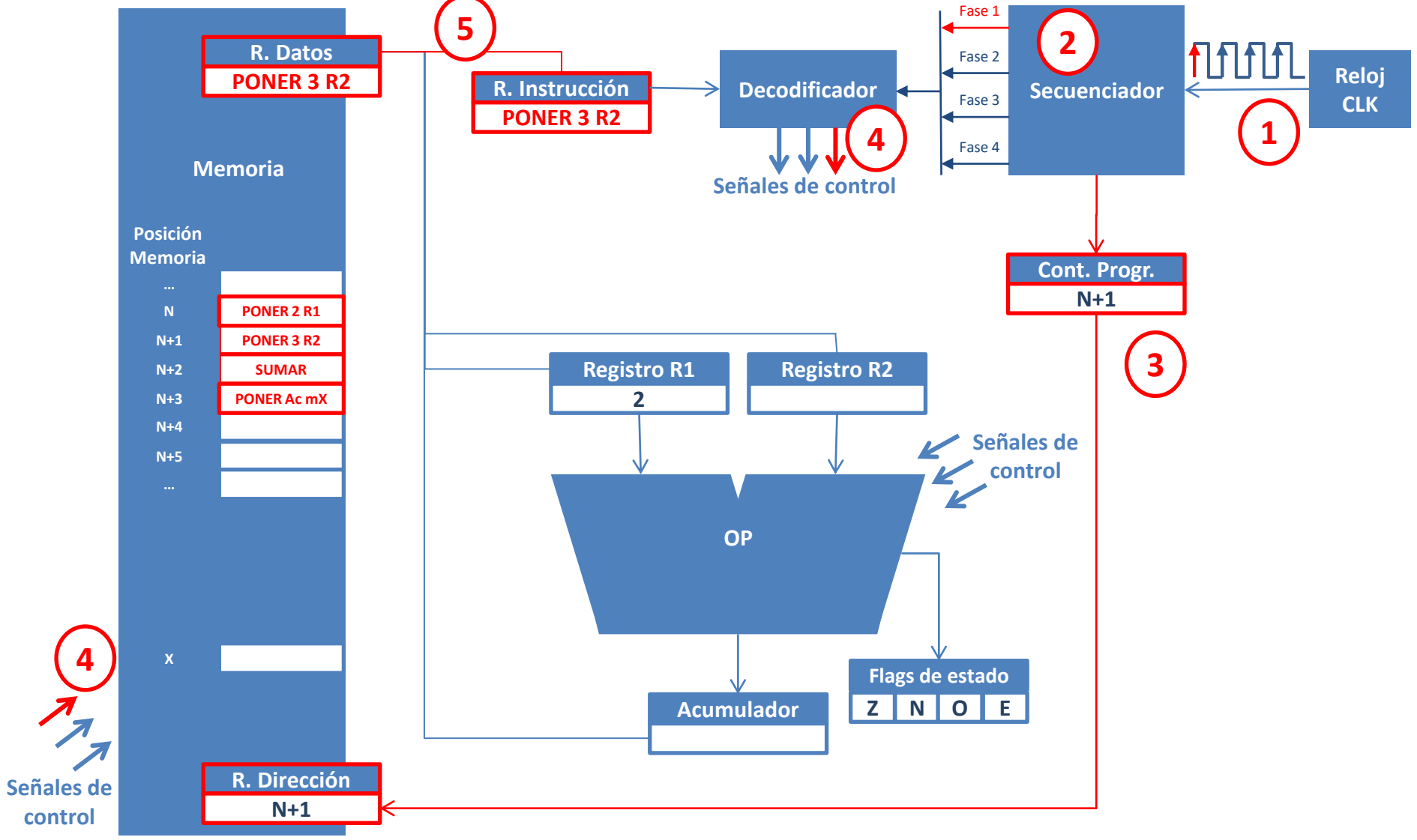
Instrucción 2: PONER 3 R2

Instrucción 3: SUMAR

Instrucción 4: PONER Ac mX

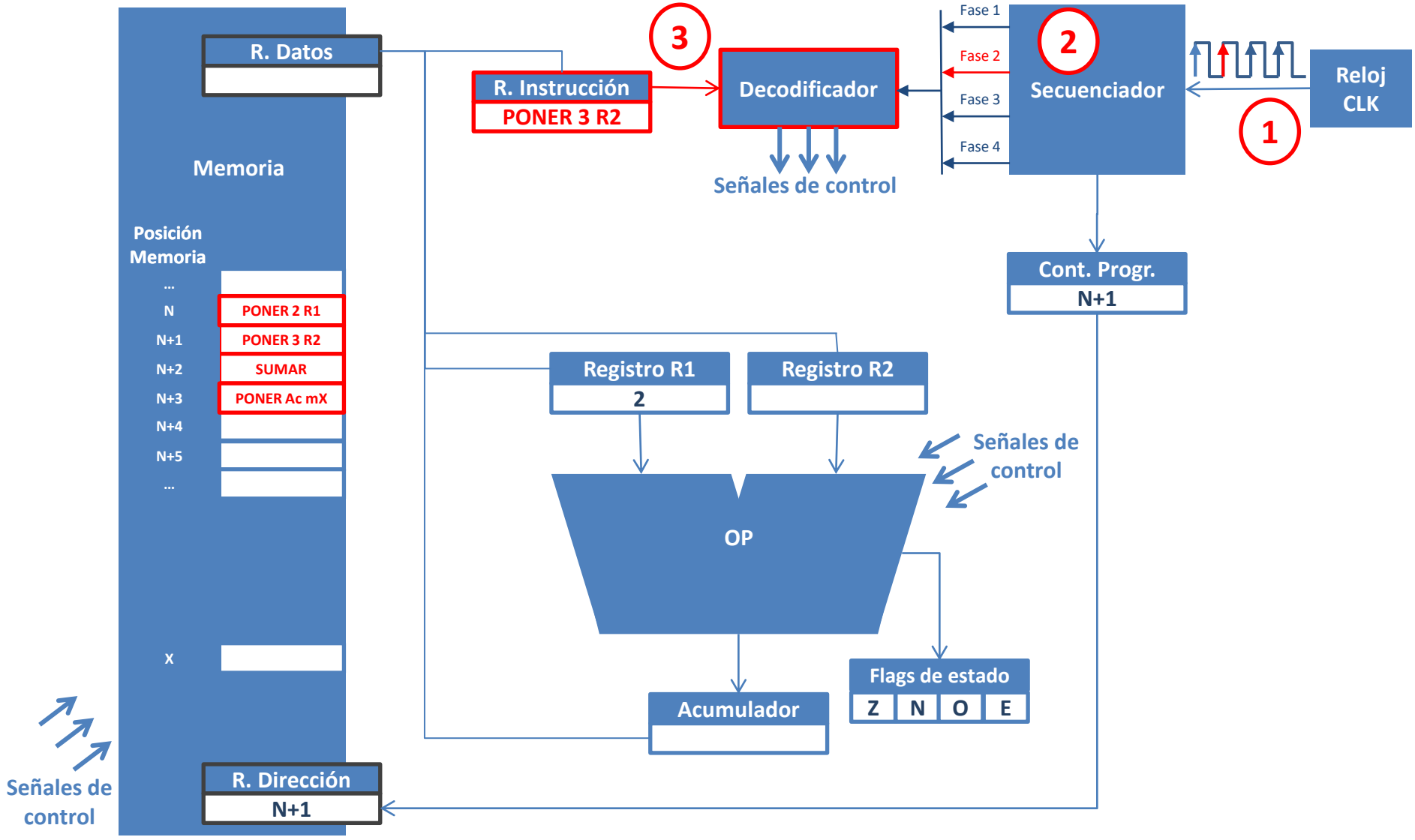
FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

FASE 1: BUSQUEDA DE INSTRUCION: El reloj genera un pulso, el secuenciador genera fase 1, el contador de programa vuelca su contenido en el R.Dirección de memoria, el decodificador da la ordena se seleccionar celda N+1, el contenido se lleva al registro de instrucción.



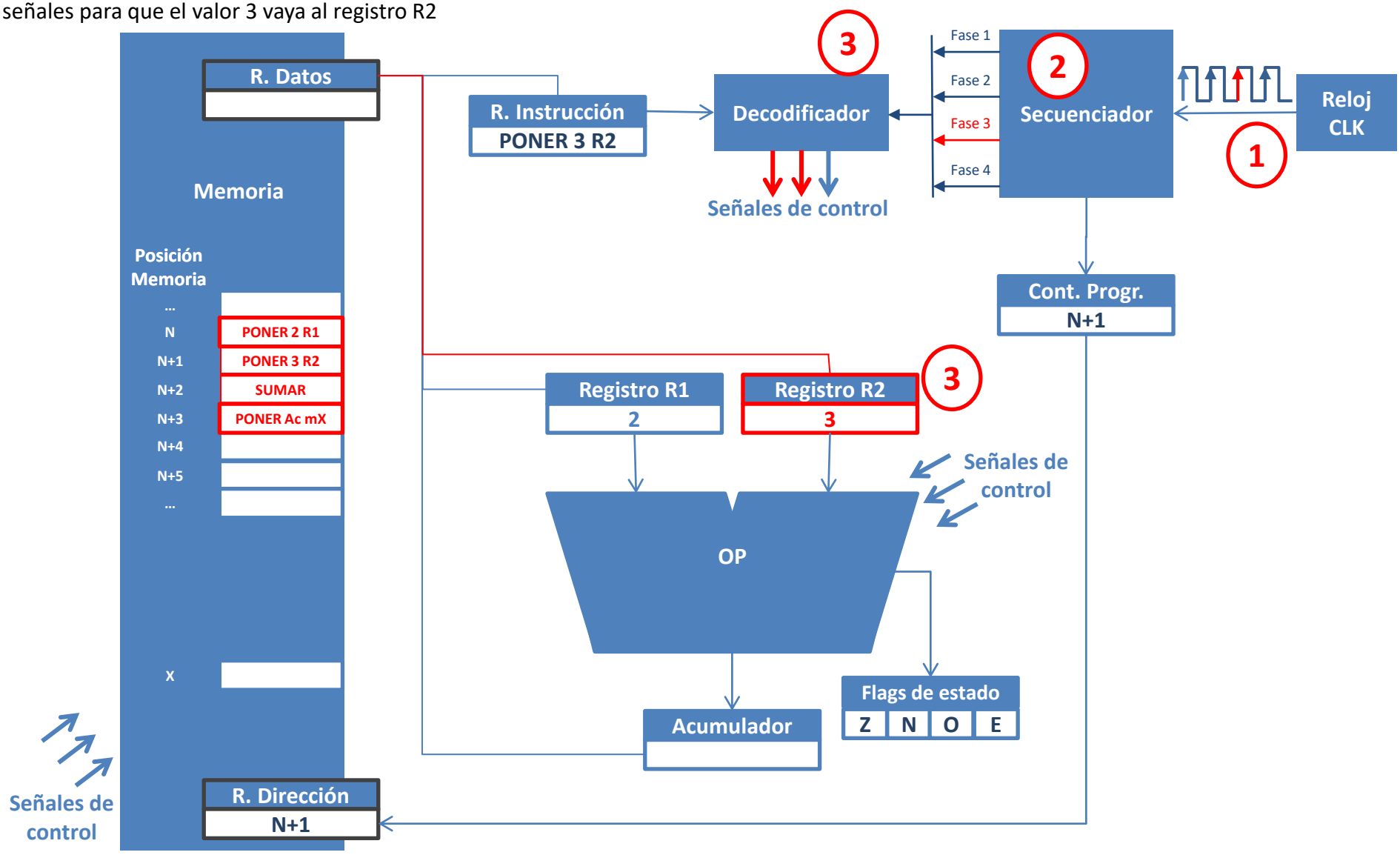
FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

FASE 2: DECODIFICACION DE INSTRUCCIÓN: El reloj genera otro pulso y el secuenciador se pone en fase 2. El decodificador accede al registro de instrucción y decodifica de que instrucción se trata



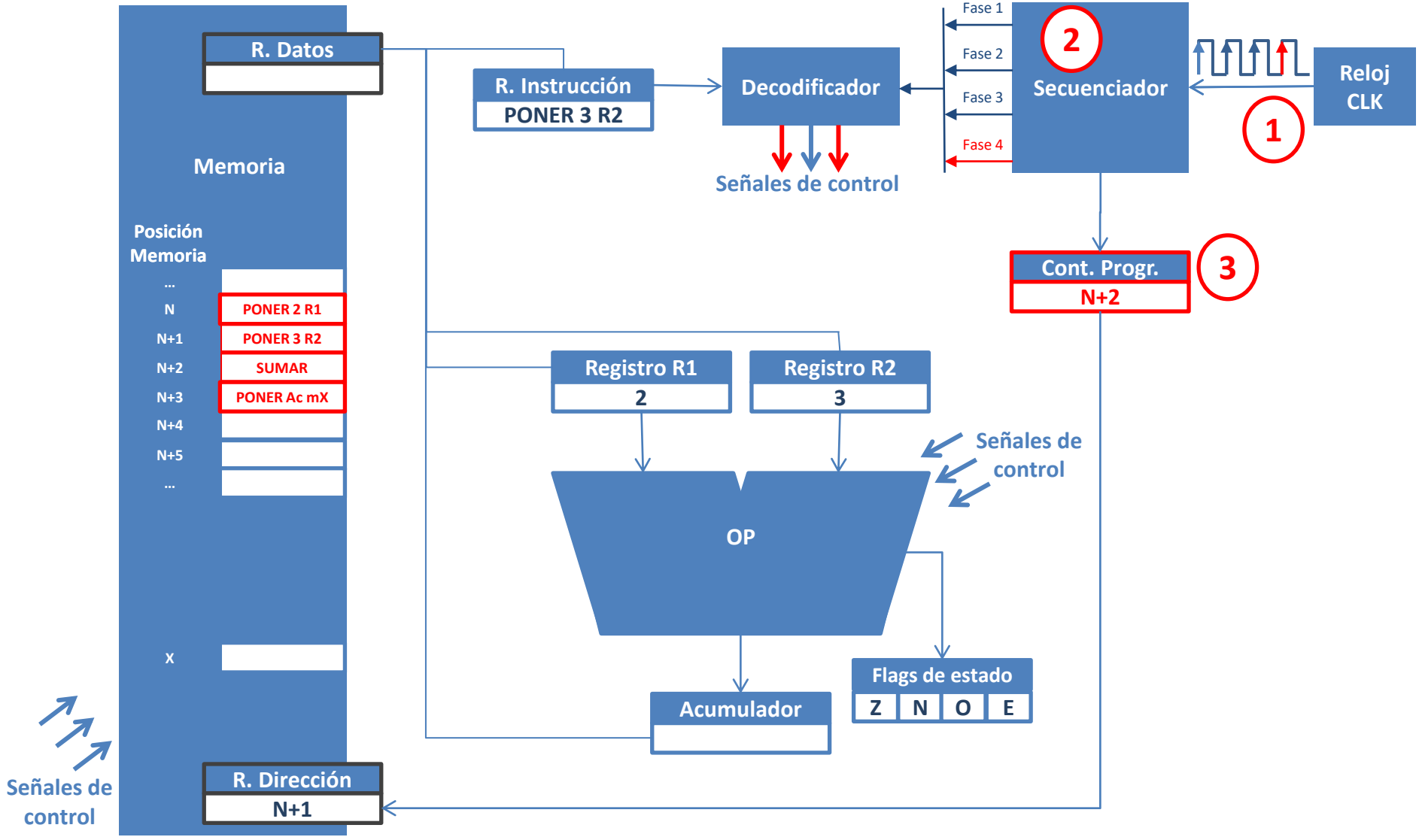
FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

FASE 3: EJECUCIÓN DE INSTRUCCIÓN: El reloj genera otro pulso y el secuenciador se pone en fase 3. El decodificador activa las señales para que el valor 3 vaya al registro R2



FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

FASE 4: SIGUIENTE INSTRUCCIÓN Y ALMACENAMIENTO DE RESULTADOS: El reloj genera el siguiente pulso y el secuenciador se pone en fase 4. Se incrementar en 1 el contador de programa.



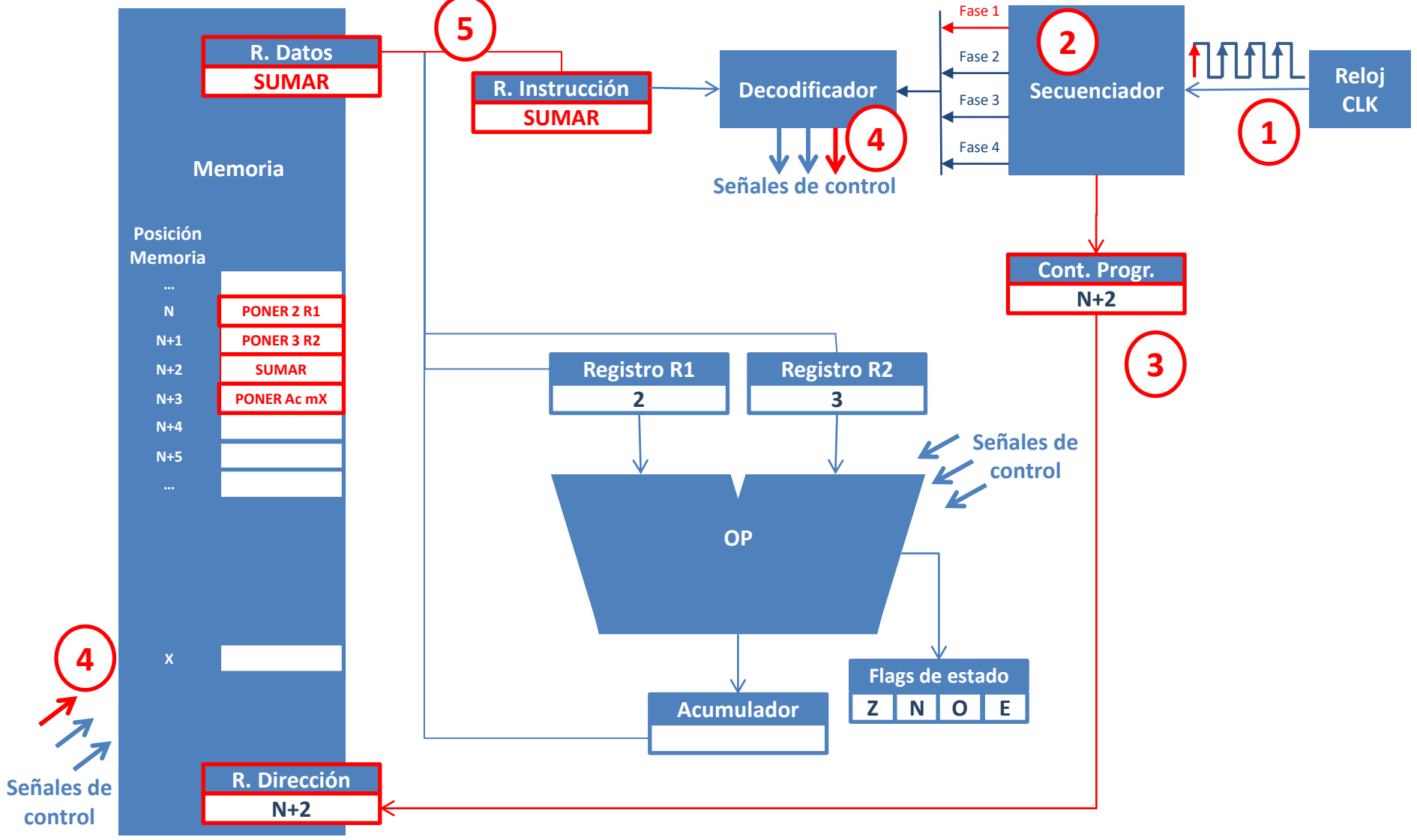
FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

EJECUCIÓN DE INSTRUCCIÓN 3

Instrucción 1: PONER 2 R1
Instrucción 2: PONER 3 R2
Instrucción 3: SUMAR
Instrucción 4: PONER Ac mX

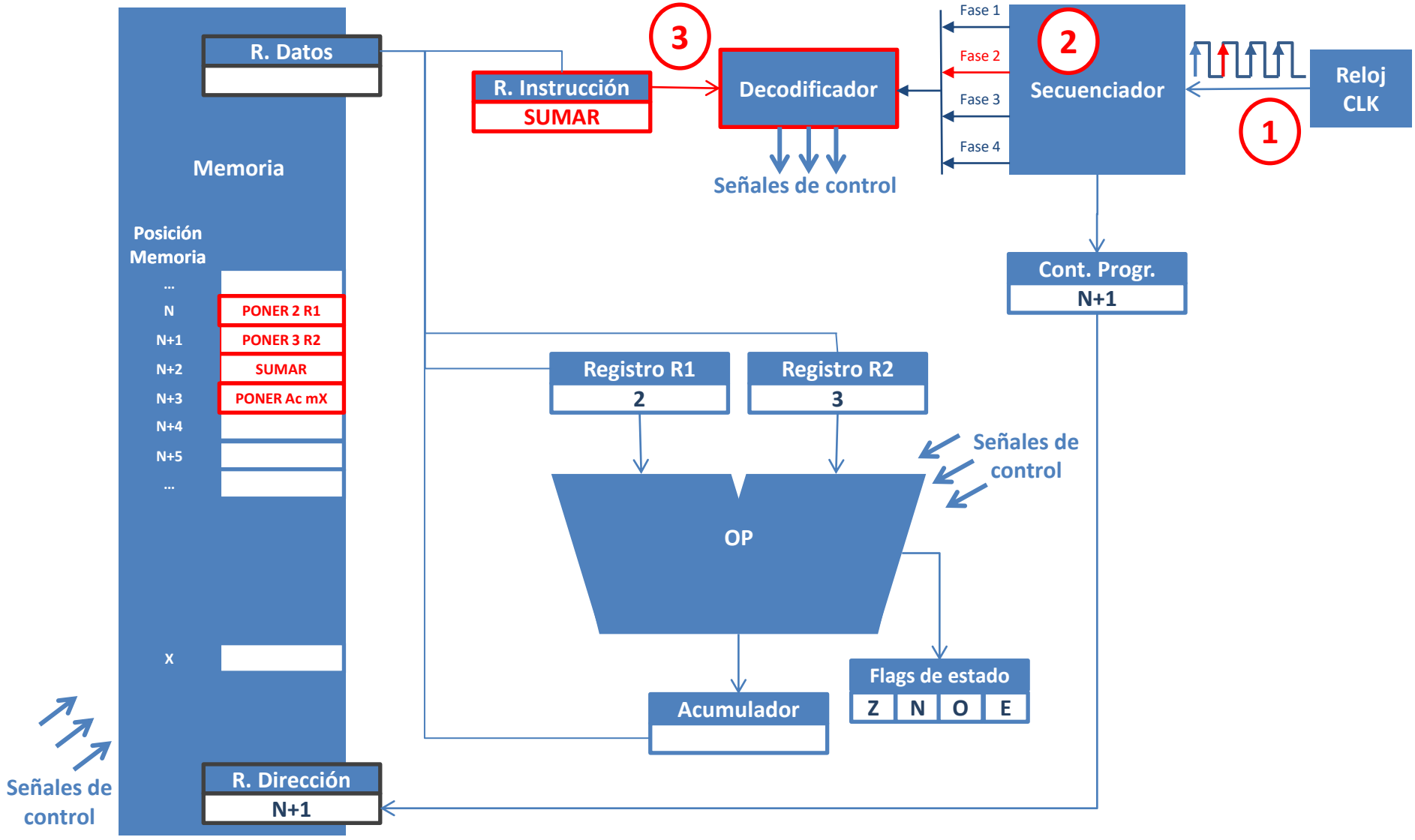
FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

FASE 1: BUSQUEDA DE INSTRUCION: El reloj genera un pulso, el secuenciador genera fase 1, el contador de programa vuelca su contenido en el R.Dirección de memoria, el decodificador da la ordena se seleccionar celda N+2, el contenido se lleva al registro de instrucción.



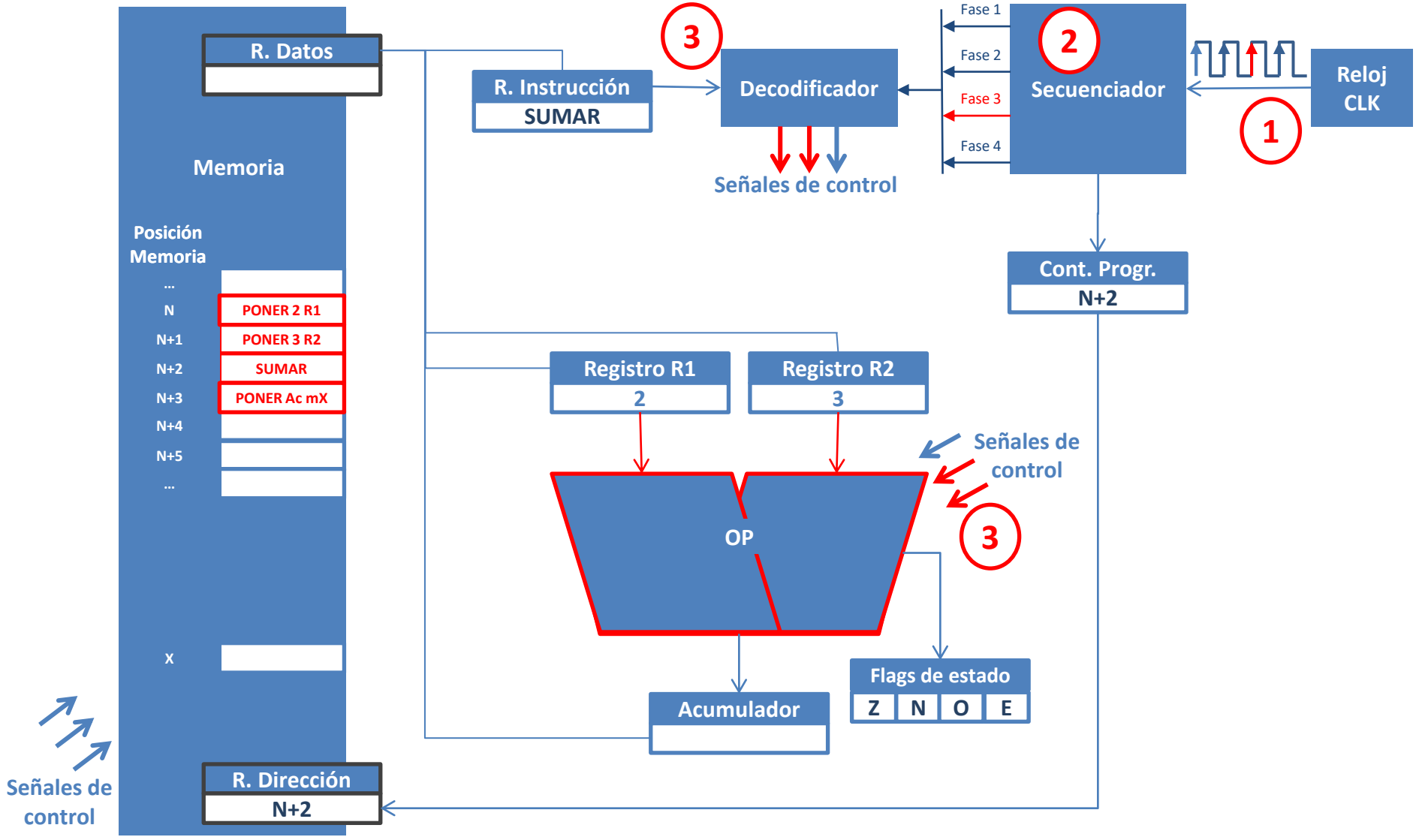
FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

FASE 2: DECODIFICACION DE INSTRUCCIÓN: El reloj genera otro pulso y el secuenciador se pone en fase 2. El decodificador accede al registro de instrucción y decodifica de que instrucción se trata



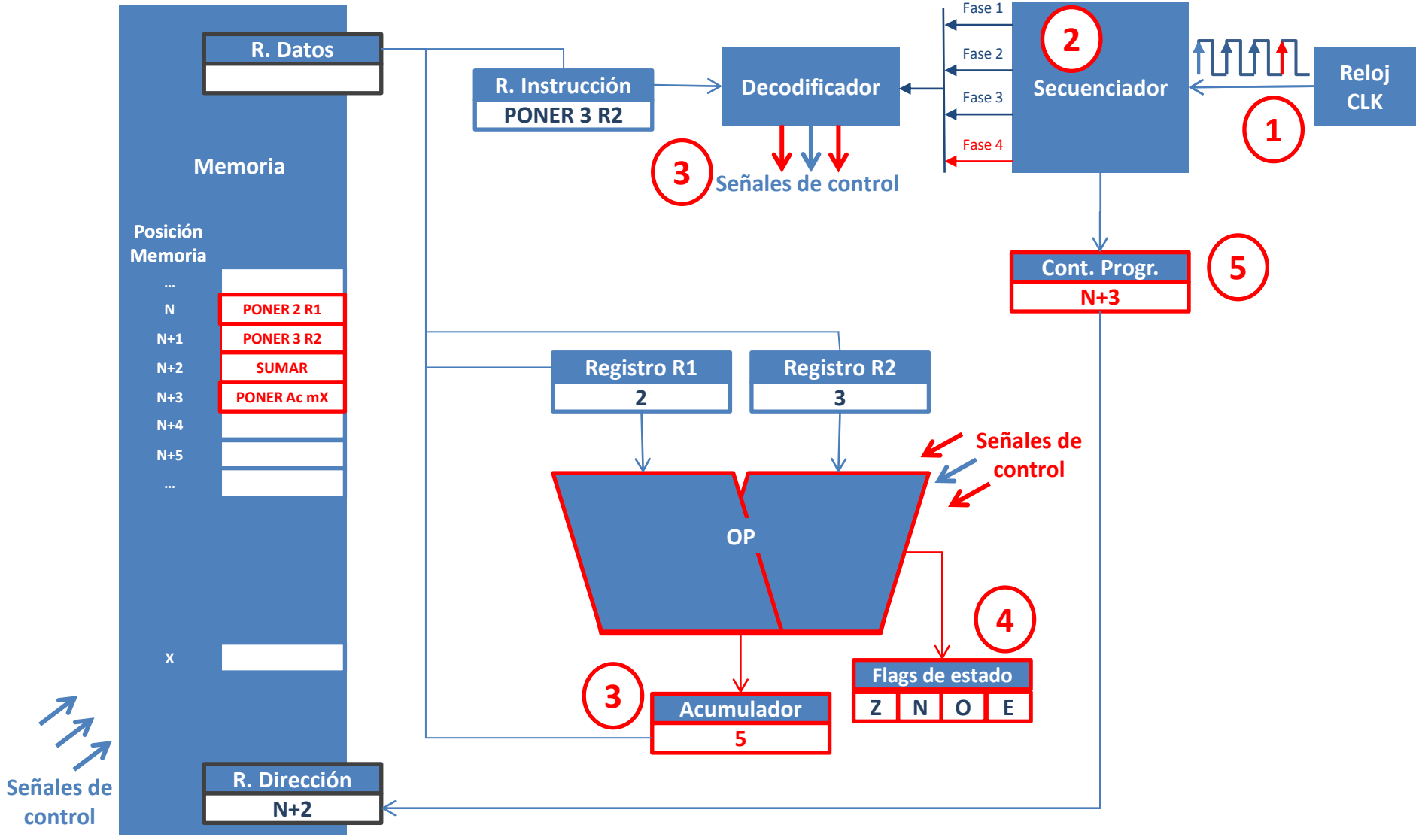
FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

FASE 3: EJECUCIÓN DE INSTRUCCIÓN: El reloj genera otro pulso y el secuenciador se pone en fase 3. El decodificador activa las señales para que la ALU haga la suma



FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

FASE 4: SIGUIENTE INSTRUCCIÓN Y ALMACENAMIENTO DE RESULTADOS: El reloj genera el siguiente pulso y el secuenciador se pone en fase 4. Se pasa el resultado al Acumulador, se actualizan los flags y se incrementar en 1 el contador de programa.



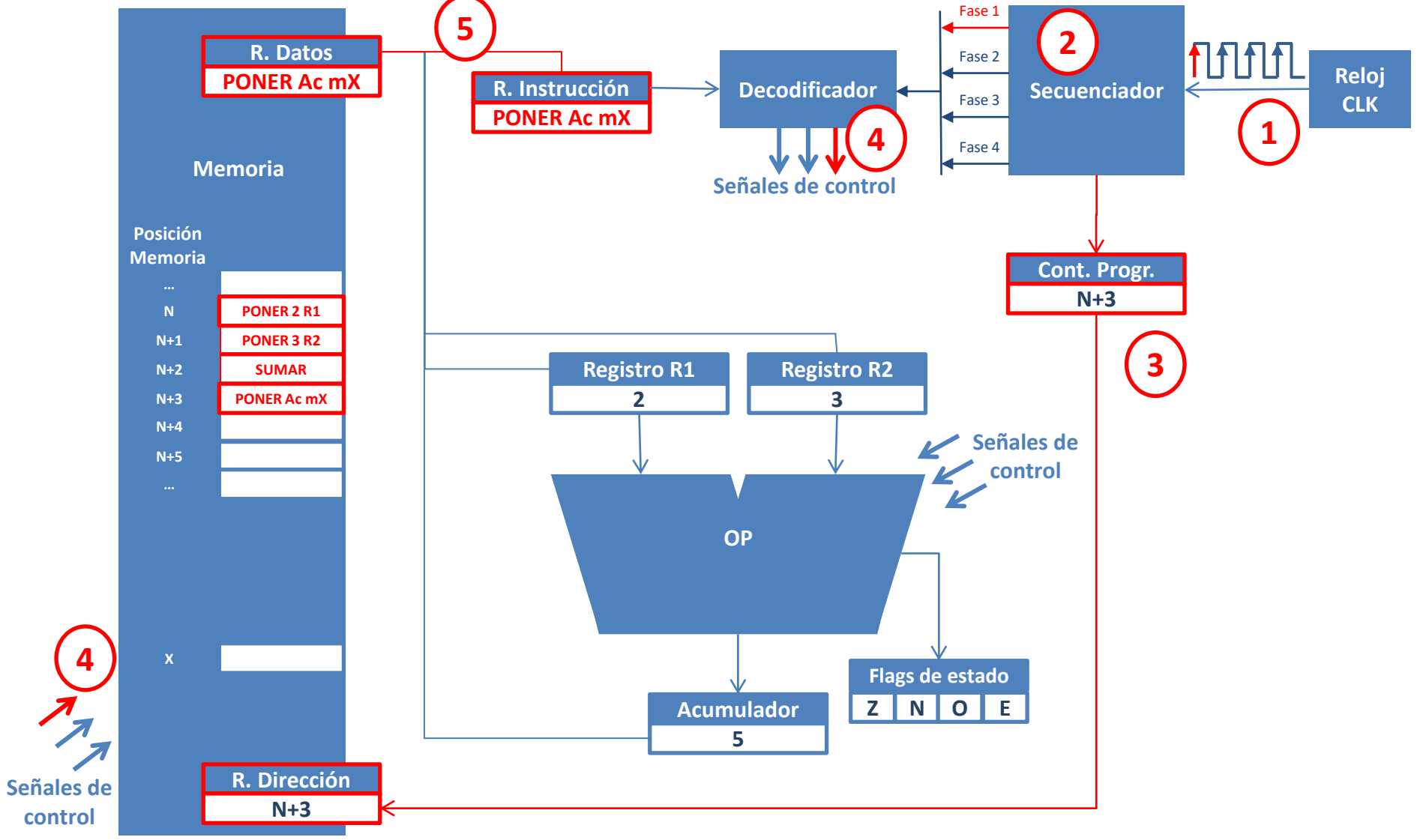
FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

EJECUCIÓN DE INSTRUCCIÓN 3

Instrucción 1: PONER 2 R1
Instrucción 2: PONER 3 R2
Instrucción 3: SUMAR
Instrucción 4: PONER Ac mX

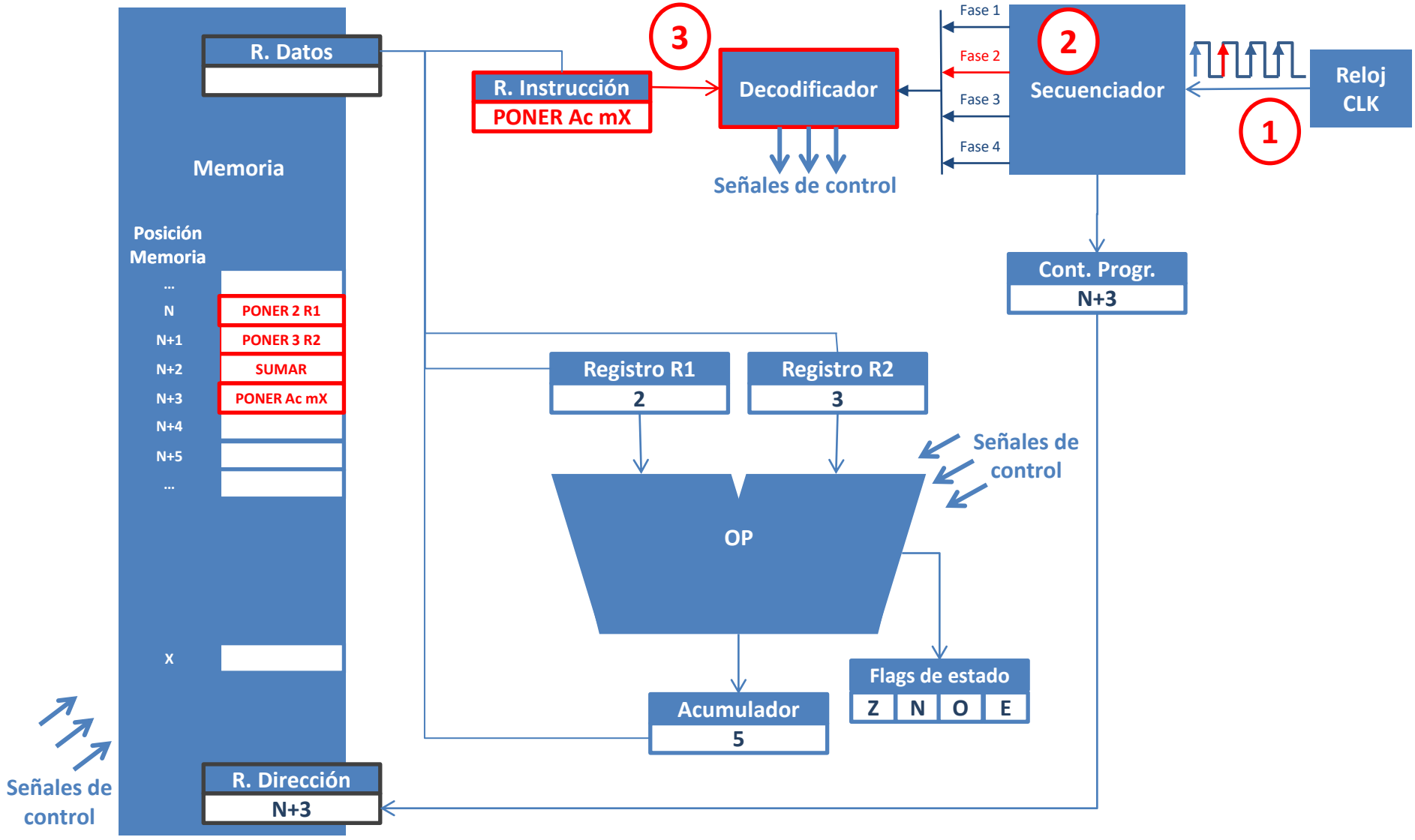
FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

FASE 1: BUSQUEDA DE INSTRUCION: El reloj genera un pulso, el secuenciador genera fase 1, el contador de programa vuelca su contenido en el R.Dirección de memoria, el decodificador da la ordena se seleccionar celda N+2, el contenido se lleva al registro de instrucción.



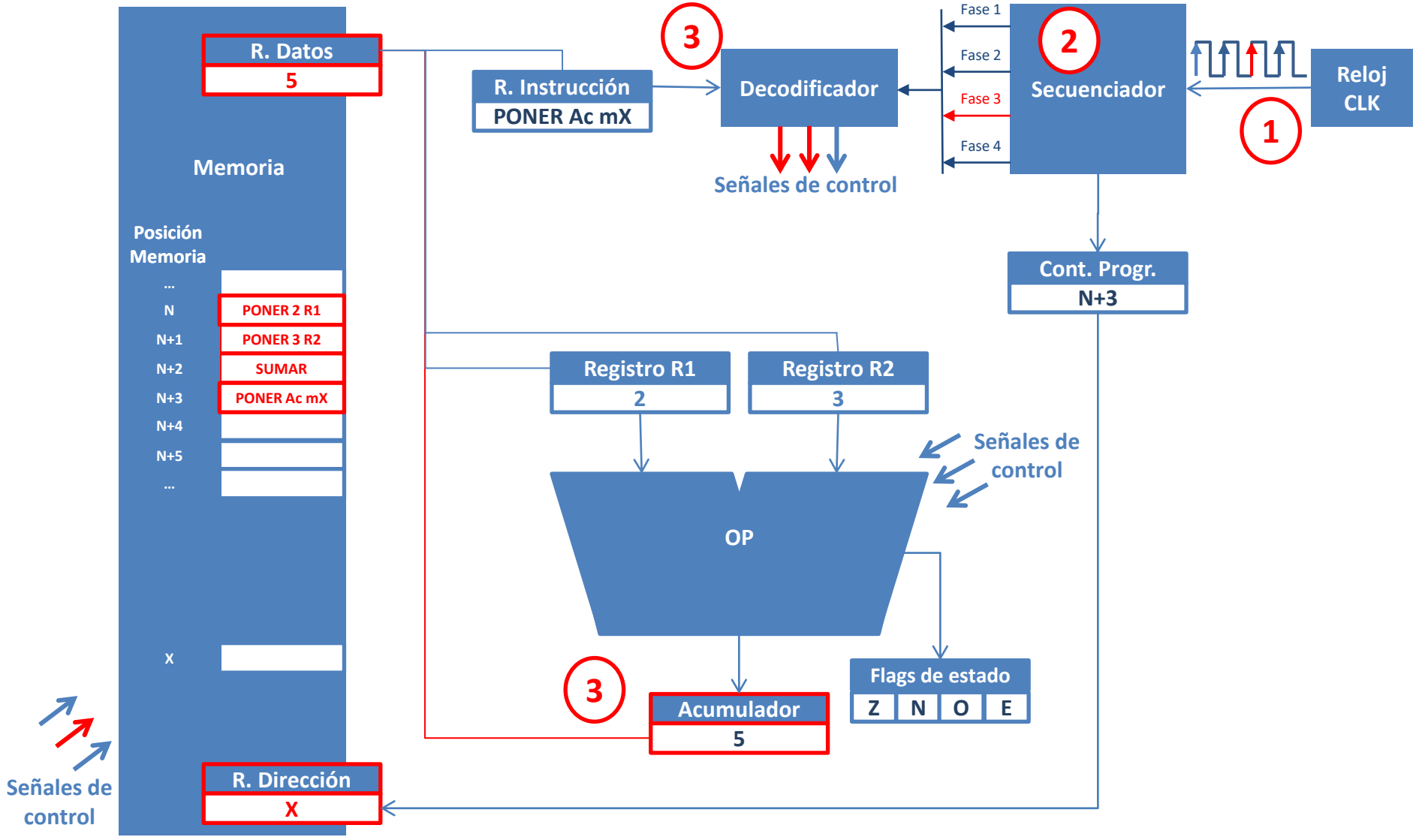
FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

FASE 2: DECODIFICACION DE INSTRUCCIÓN: El reloj genera otro pulso y el secuenciador se pone en fase 2. El decodificador accede al registro de instrucción y decodifica de que instrucción se trata



FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

FASE 3: EJECUCIÓN DE INSTRUCCIÓN: El reloj genera otro pulso y el secuenciador se pone en fase 3. El decodificador activa las señales para que el contenido del acumulador pase a la posición de Memoria X



FUNCIONAMIENTO DEL COMPUTADOR ELEMENTAL

FASE 4: SIGUIENTE INSTRUCCIÓN Y ALMACENAMIENTO DE RESULTADOS: El reloj genera el siguiente pulso y el secuenciador se pone en fase 4. Se actualiza la memoria y se incrementar en 1 el contador de programa.