Estructura de Computadores

(EdC)

Tema 3

El computador simple

Autores originales: David Guerrero. Isabel Gómez

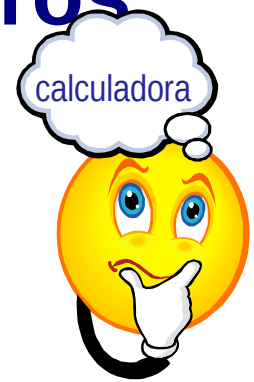
Personalización para ISW: Francisco Pérez

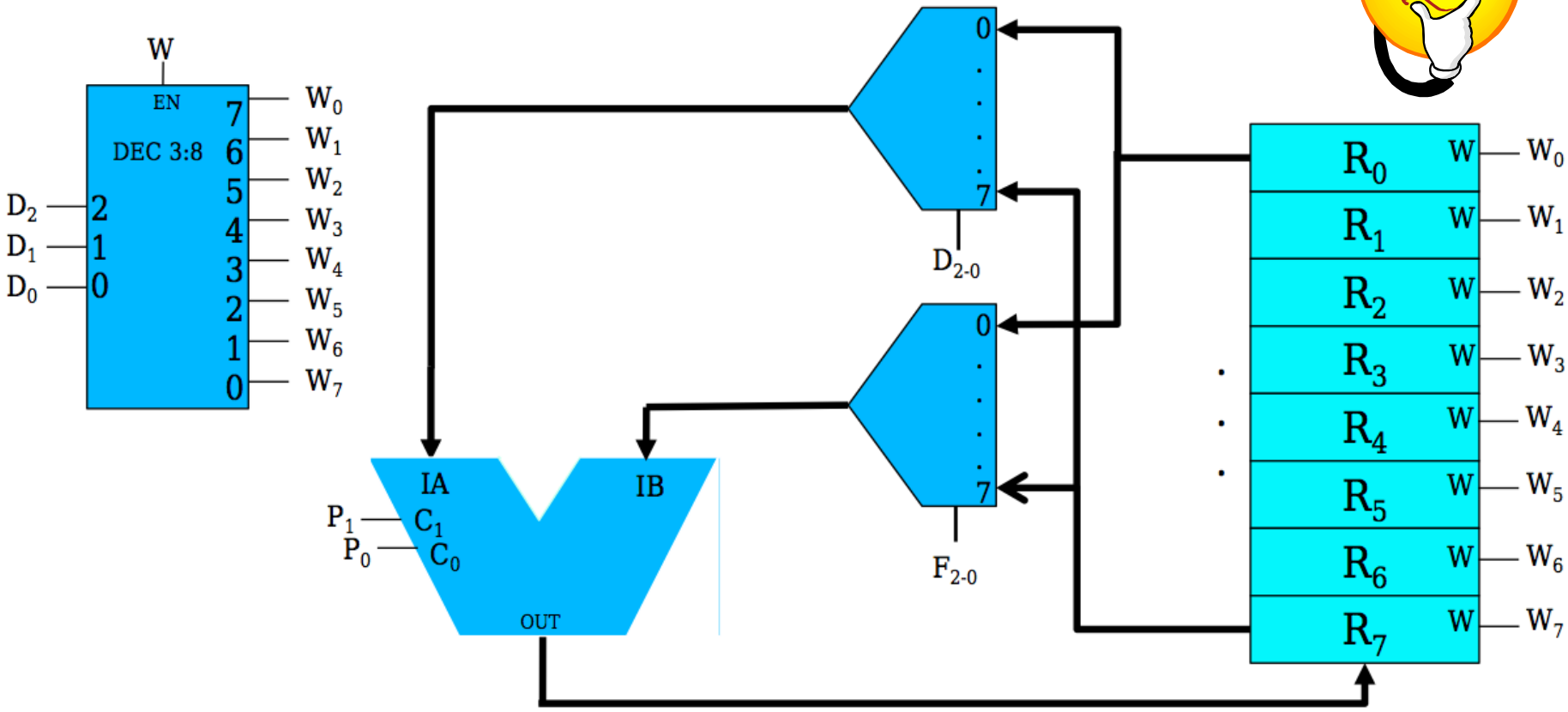
Adpatación del CS2010: Alberto J. Molina. Última modificación: 14/03/13

Índice

La calculadora simple de 8 registros

(Arquitectura de la Unidad de Datos)

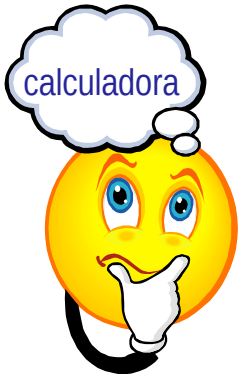




(Permite sumar, restar o transferir información entre dos registros)

La calculadora simple de 8 registros

- Partimos de la calculadora planteada en el tema anterior que es un sistema en un único paso.
- La calculadora ejecuta cualquier posibilidad de suma o resta entre sus registros así como el movimiento de datos entre los mismos.
- Las operaciones se ejecutan en un único ciclo de reloj.
  - En total, 3 ciclos:
    - S0: microop. de espera Xs
    - S1: microop. de ejecución
    - S2: microop. de FIN



Limitaciones de la calculadora simple

- No hay AUTOMATIZACIÓN EN LA EJECUCIÓN de instrucciones
- No hay PROGRAMA ALMACENADO (cada vez que se ejecuta una instrucción el usuario debe suministrar la siguiente).
- Por tanto, si queremos realizar una operación algo más compleja, debemos indicar, una a una, la secuencia de macroinstrucciones:

Por ejemplo, una posible secuencia de macroinstrucciones que debería dar el usuario (a nivel ISP) para
R0 ← 3R1-R2, sería:

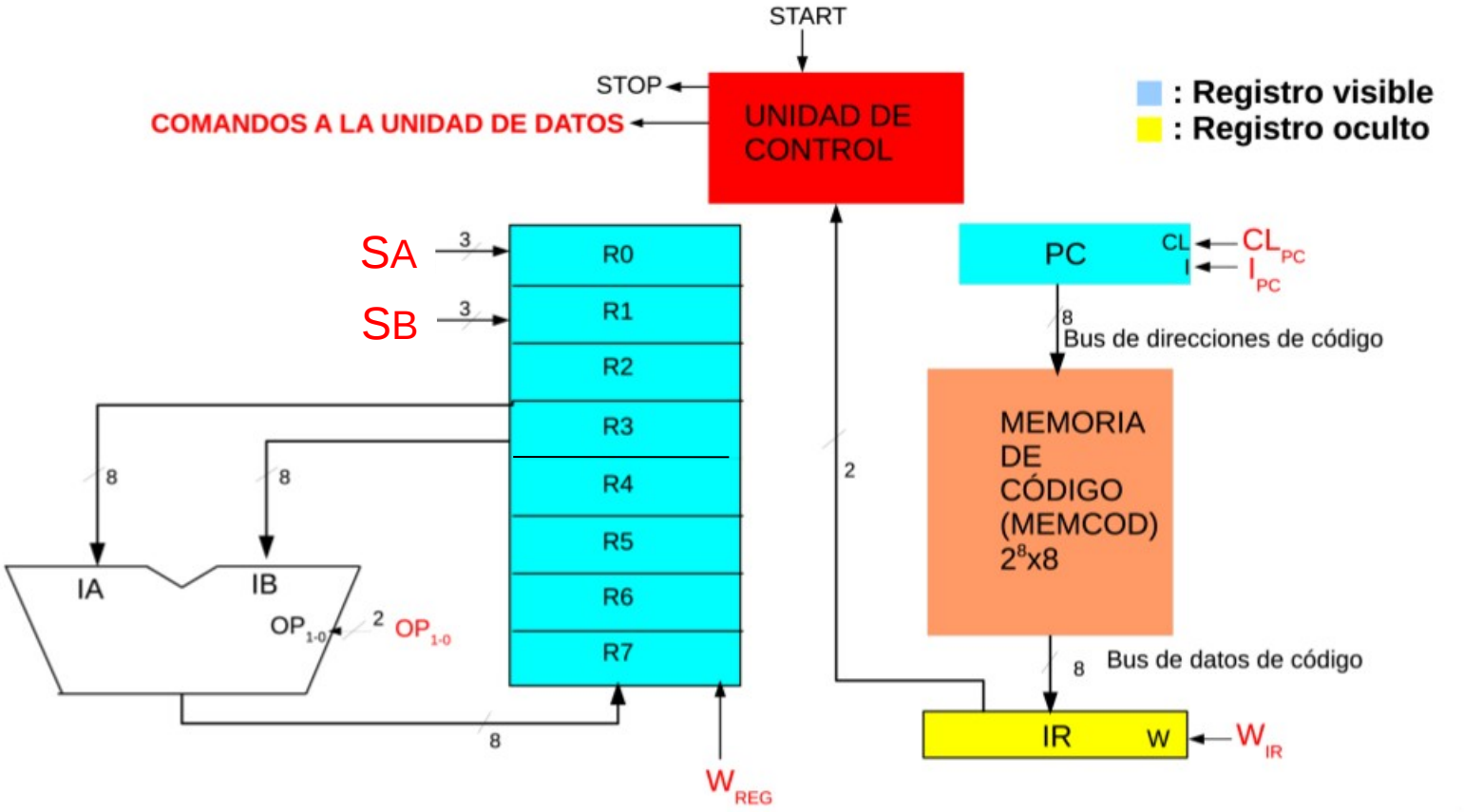
```
R0 ← R1
R0 ← R0+R1
R0 ← R0+R1
R0 ← R0–R2
```

Índice

Automatización en la ejecución (requerimientos)

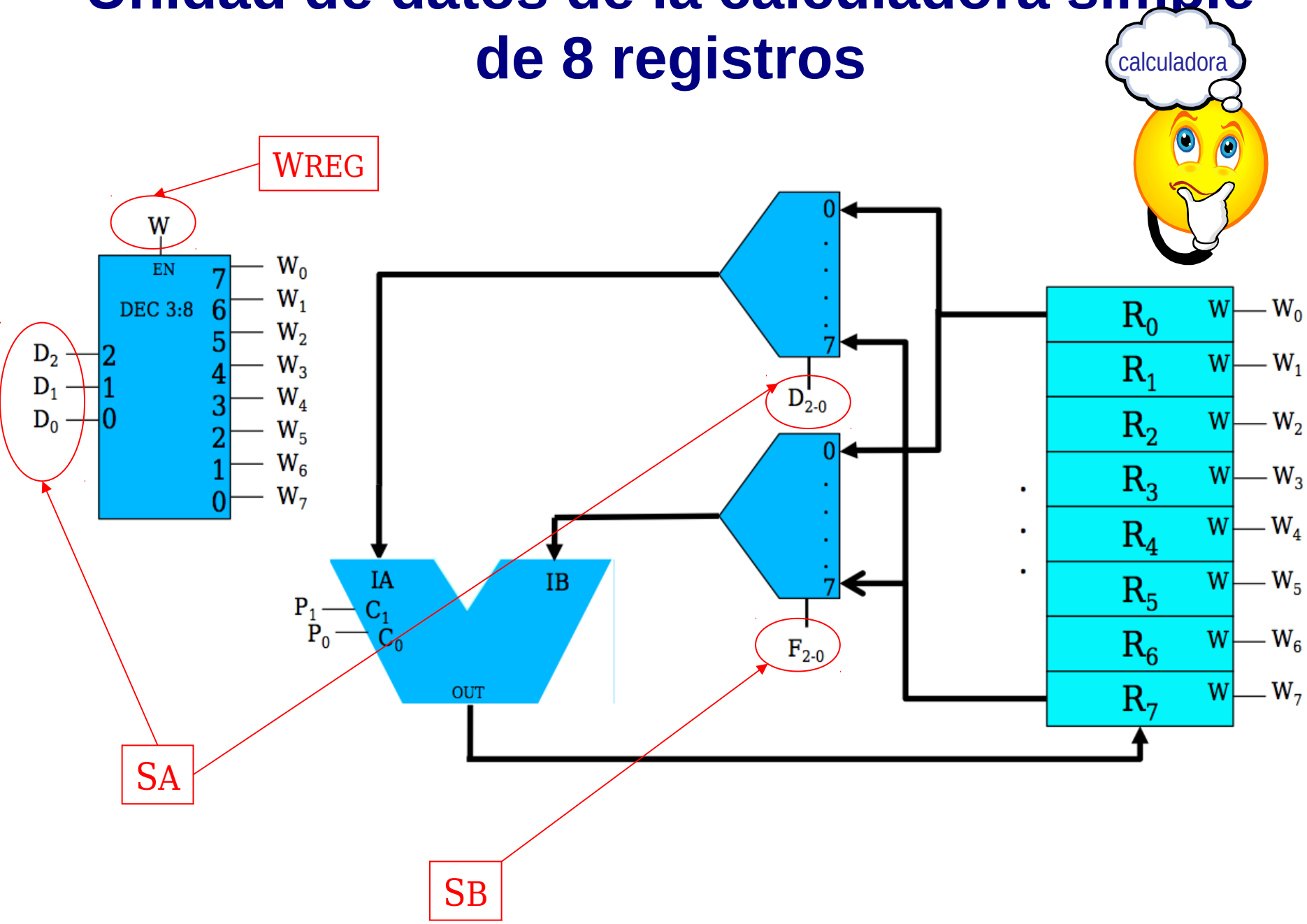
- Una memoria donde almacenar el programa (código)
- Una nueva unidad de control que “sepa” buscar instrucciones y ejecutarlas
- Un registro que apunte a la instrucción que ha de ejecutarse
  PC: Program Counter
- Un registro donde se almacene la instrucción que se está ejecutando en ese momento
  IR: Instruction Register

Arquitectura del computador CS1



- Se ha simplificado el dibujo del banco de registros (no se dibujan los MUX)
  - SA y SB: Selecciona registro de entrada en IA o IB (de la ALU)
  - WREG: Selecciona el registro en el que se escribe el resultado de la ALU
- El programa (secuencia de instrucciones) se almacena en la memoria (256x8)
- El PC apunta a la próxima instrucción a ejecutar
- El IR almacena la instrucción (8 bits) que se está ejecutando

Unidad de datos de la calculadora simple de 8 registros

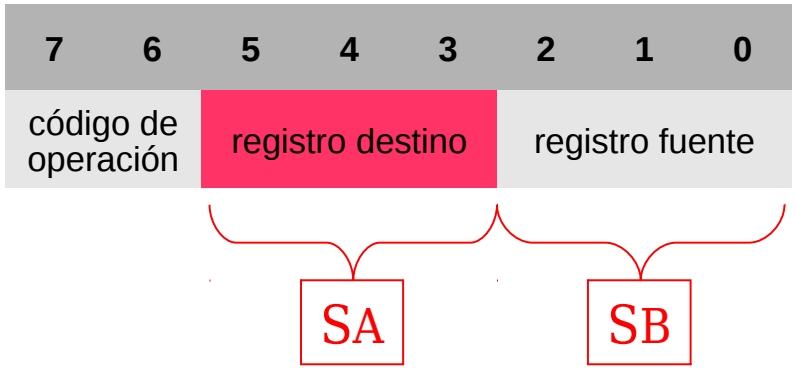


Aclaraciones sobre CS1

- Todas las instrucciones son de una palabra (1 byte).
- El programa a ejecutar debe almacenarse a partir de la dirección 0 de la memoria
- La ejecución del programa es lineal (sin bucles ni saltos).
- Instrucción en código máquina. Es el patrón de bits correspondiente a una instrucción.
- Formato de instrucción: indica cómo debe ser interpretada una instrucción en código máquina (código de operación y operandos).

Instrucciones del computador CS1

- Formato de todas las instrucciones del CS1



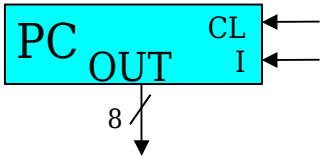
En CS1 los datos (operandos de las instrucciones) siempre están almacenados en el banco de registros.

- Conjunto de instrucciones: sólo 4 instrucciones (IR7 IR6)

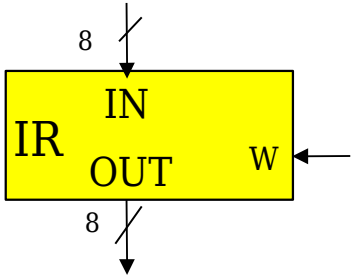
IR7 IR6	SINTAXIS	FUNCIÓN
0 0	ADD Rd,Rf	Rd ← Rd+Rf
1 0	SUB Rd,Rf	Rd ← Rd-Rf
0 1	MOV Rd,Rf	Rd ← Rf
1 1	STOP	NOP

En la sintaxis se han utilizado nemónicos que facilitan la tarea del programador.

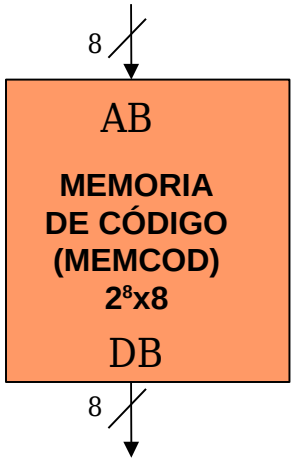
Descripción de los nuevos componentes del CS1



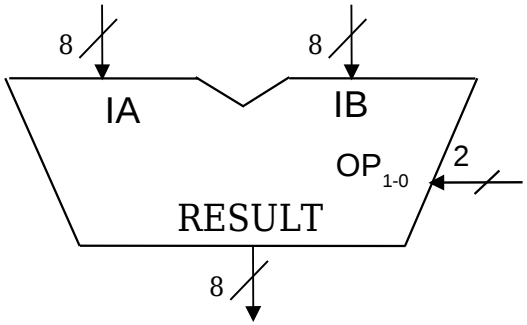
CL I	PC←	OUT=
00	PC	PC
01	PC+1	PC
10	0	PC
11	0	PC



W	IR←	OUT=
0	IR	IR
1	IN	IR



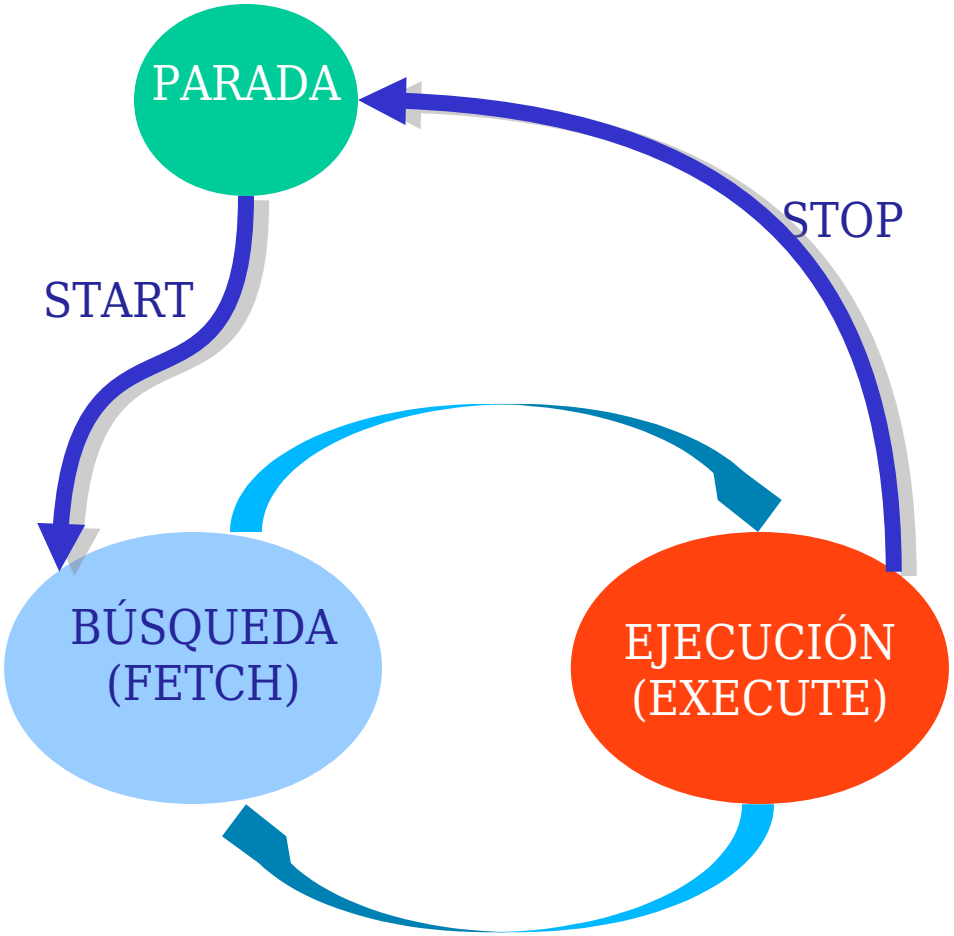
DB=MEMCOD[AB]



OP_1 OP_0	RESULT=
00	IA+IB
01	IA
10	IA-IB
11	IB

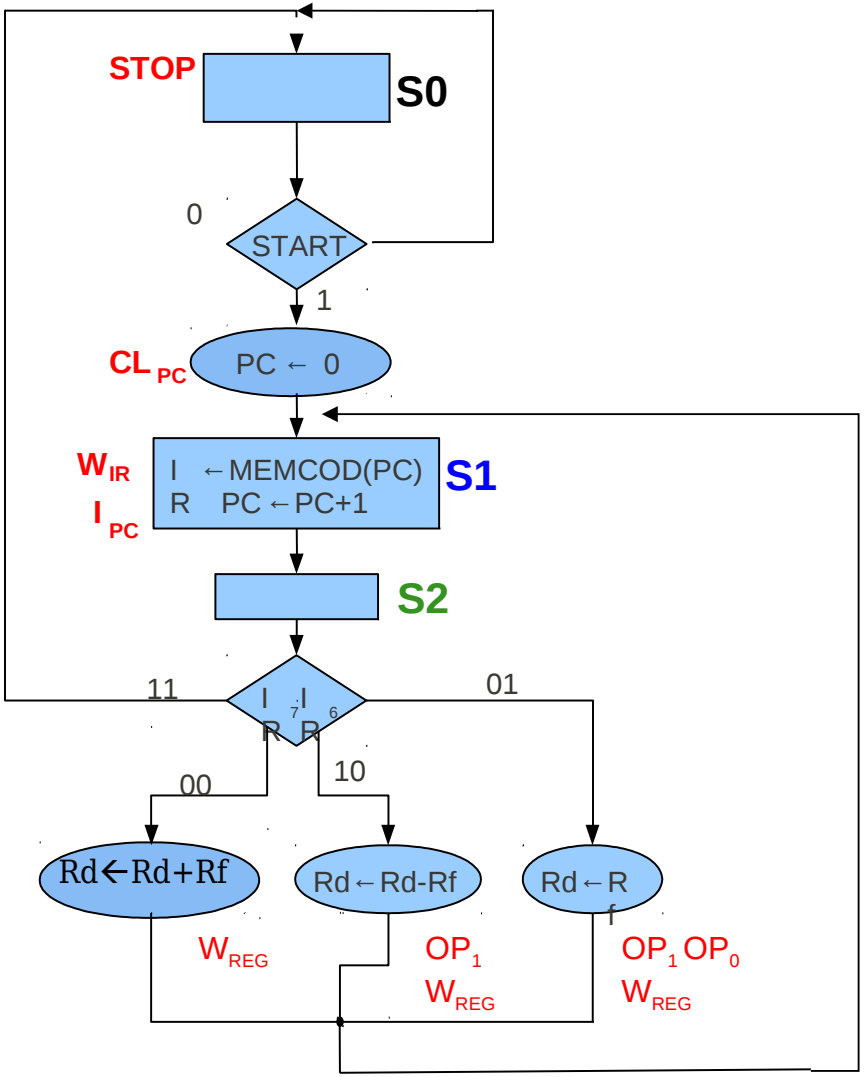
Diseño de la Unidad de Control del Computador Simple CS1

Controla la ejecución automática del programa almacenado en la memoria



Carta ASM del CS1

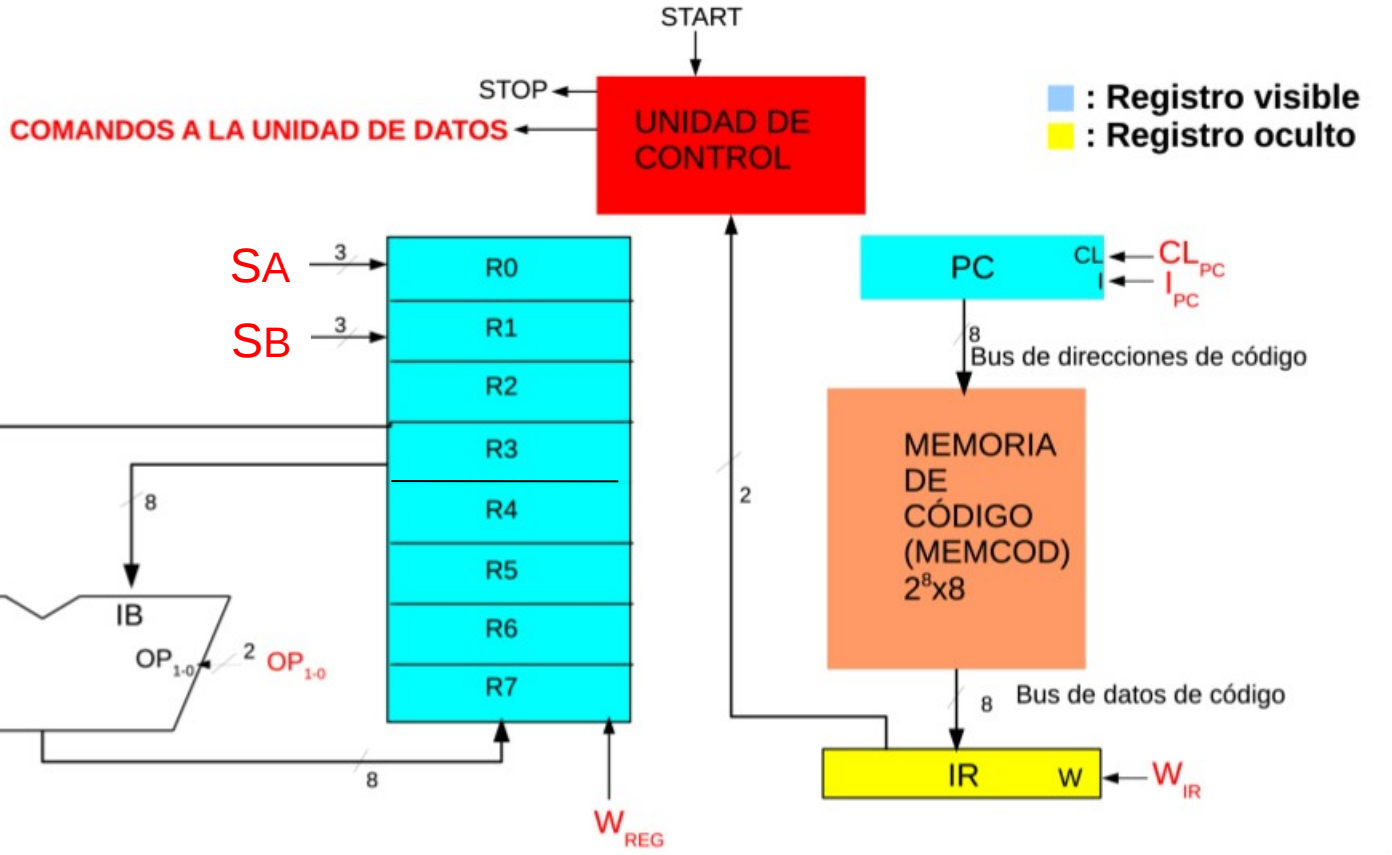
FETCH: S1
EXECUTE: S2

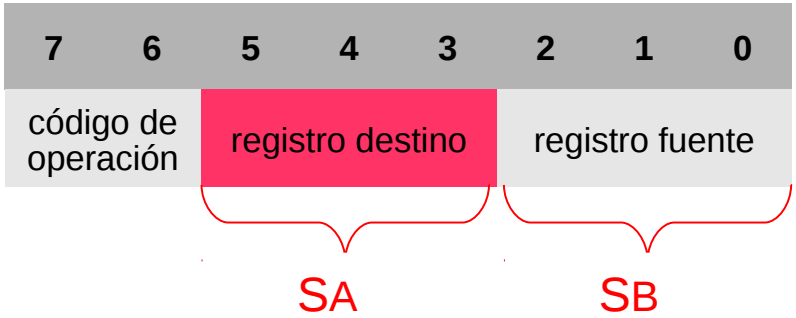


OP_1 OP_0	RESULT ←
00	IA+IB
01	IA
10	IA-IB
11	IB

IR7	IR6	SINTAXIS	FUNCIÓN
0	0	ADD Rd,Rf	Rd ← Rd+Rf
1	0	SUB Rd,Rf	Rd ← Rd-Rf
0	1	MOV Rd,Rf	Rd ← Rf
1	1	STOP	NOP

Arquitectura del computador CS1





Ejemplo de programación del CS1

Ejemplo:

Escribir un programa que realice la siguiente operación:

R6←3R4-2R1

Programa (ensamblador)

```
MOV R6,R4
SUB R6,R1
ADD R6,R6
ADD R6,R4
STOP
```

$Posición	contenido
$00	01 110 100
$01	10 110 001
$02	00 110 110
$03	00 110 100
$04	11 - - - - - -

Programa en memoria (Código máquina)

IR7	IR6	SINTAXIS	FUNCIÓN
0	0	ADD Rd,Rf	Rd←Rd+Rf
1	0	SUB Rd,Rf	Rd←Rd-Rf
0	1	MOV Rd,Rf	Rd←Rf
1	1	STOP	NOP

7	6	5	4	3	2	1	0
código de operación		registro destino			registro fuente		

Ejemplos de programación del CS1

Ejemplo 2:
Si inicialmente los contenidos de los registros son:
R1=$19, R4= $22 y R6=$43, indique sus contenidos tras cara instrucción

R6←3R4-2R1

Programa	R1	R4	R6
	$19	$22	$43
MOV R6,R4			
SUB R6,R1			
ADD R6,R6			
ADD R6,R4			
STOP			

Ejemplos de programación del CS1

Ejemplo 2:
Si inicialmente los contenidos de los registros son:
R1=$19, R4= $22 y R6=$43, indique sus contenidos tras cara instrucción

R6←3R4-2R1

Programa	R1	R4	R6
	$19	$22	$43
MOV R6,R4	$19	$22	$22
SUB R6,R1			
ADD R6,R6			
ADD R6,R4			
STOP			

Ejemplos de programación del CS1

Ejemplo 2:
Si inicialmente los contenidos de los registros son:
R1=$19, R4= $22 y R6=$43, indique sus contenidos tras cara instrucción

R6←3R4-2R1

Programa	R1	R4	R6
	$19	$22	$43
MOV R6,R4	$19	$22	$22
SUB R6,R1	$19	$22	$09
ADD R6,R6			
ADD R6,R4			
STOP			

Ejemplos de programación del CS1

Ejemplo 2:
Si inicialmente los contenidos de los registros son:
R1=$19, R4= $22 y R6=$43, indique sus contenidos tras cara instrucción

R6←3R4-2R1

Programa	R1	R4	R6
	$19	$22	$43
MOV R6,R4	$19	$22	$22
SUB R6,R1	$19	$22	$09
ADD R6,R6	$19	$22	$12
ADD R6,R4			
STOP			

Ejemplos de programación del CS1

Ejemplo 2:
Si inicialmente los contenidos de los registros son:
R1=$19, R4= $22 y R6=$43, indique sus contenidos tras cara instrucción

R6←3R4-2R1

Programa	R1	R4	R6
	$19	$22	$43
MOV R6,R4	$19	$22	$22
SUB R6,R1	$19	$22	$09
ADD R6,R6	$19	$22	$12
ADD R6,R4	$19	$22	$34
STOP			

Ejemplos de programación del CS1

Ejemplo 2:
Si inicialmente los contenidos de los registros son:
R1=$19, R4= $22 y R6=$43, indique sus contenidos tras cara instrucción

R6←3R4-2R1

Programa	R1	R4	R6
	$19	$22	$43
MOV R6,R4	$19	$22	$22
SUB R6,R1	$19	$22	$09
ADD R6,R6	$19	$22	$12
ADD R6,R4	$19	$22	$34
STOP	$19	$22	$34

Ejemplos de programación del CS1

Ejemplo 3:
Si inicialmente los contenidos de los registros son:
R1=$22, R4= $19 y R6=$43, indique sus contenidos tras cara instrucción

R6←3R4-2R1

Programa	R1	R4	R6
	$22	$19	$43
MOV R6,R4			
SUB R6,R1			
ADD R6,R6			
ADD R6,R4			
STOP			

Índice

Limitaciones y evolución del CS1

- El CS1 sólo opera con datos almacenados en el banco de registros.
- Es necesario aumentar la capacidad de almacenamiento
- Se propone una nueva arquitectura (CS2: Computador Simple 2), con los datos almacenados en una RAM
- Existen dos opciones para dotar al sistema de almacenamiento de datos:
  - Utilizar un único sistema de memoria para datos e instrucciones (Arquitectura von Neumann).
  - Utilizar sistemas de memoria distintos para datos e instrucciones (Arquitectura Harvard).

Arquitectura von Neumann vs Harvard

- En la **arquitectura Harvard**, las características de las memorias y los buses de interconexión de las mismas pueden diferir. Normalmente los datos requieren memoria de lectura y escritura
- En la **arquitectura de von Neuman** el sistema de memoria es único y, por lo tanto, tanto memoria como buses son únicos.
- El disponer de dos sistemas de memoria separados dota de eficiencia al sistema ya que normalmente es el acceso a memoria lo que enlentece su funcionamiento y en este caso se puede estar accediendo a instrucciones y datos simultáneamente.
- Las CPU modernas incorporan aspectos de ambas arquitecturas. La memoria cache interna a la CPU se separa en dos (datos e instrucciones), pero la memoria principal es única. Desde el punto de vista del programador se trata de una arquitectura de Von Neumann, pero desde el punto de vista del hardware es Harvard.

Arquitectura del CS2

- El CS2 dispondrá de una arquitectura Harvard.

- El conjunto de instrucciones debe ser ampliado ya que se requiere manejar los datos almacenados en la memoria.

Conjunto de Instrucciones (ISP) del CS2

CO	SINTAXIS	FUNCIÓN
000	ST (Rb),Rf	MEMDAT(Rb) ← Rf
001	LD Rd, (Rb)	Rd ← MEMDAT(Rb)
010	STS dir, Rf	MEMDAT(dir) ← Rf
011	LDS Rd,dir	Rd ← MEMDAT(dir)
100	ADD Rd,Rf	Rd ← Rd+Rf
110	SUB Rd,Rf	Rd ← Rd-Rf
101	MOV Rd,Rf	Rd ← Rf
111	STOP	NOP

- Las 4 primeras instrucciones son para intercambio de datos con la memoria: (ST, LD, STS, LDS)

- Ha sido necesario aumentar el número de bits del código de operación.

- Nuevas formas de acceso a los operandos (**modos de direccionamiento**)

Modos de direccionamiento del CS2

- Directo de registro Rx: *el dato* (“operando”) se encuentra *en un registro*.

```
ADD Rd,Rf
SUB Rd,Rf
MOV Rd, Rf
```

- Indirecto de registro (Rb): *la dirección memoria del dato* se encuentra en un registro (“*registro base*”)

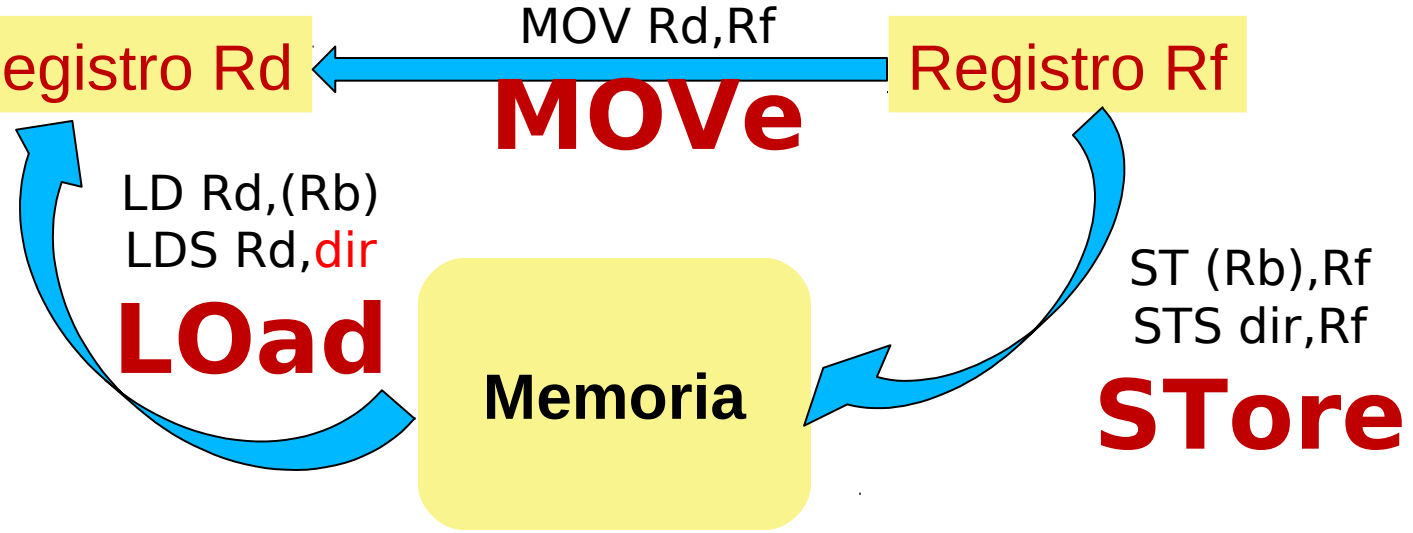
```
ST (Rb),Rf     ;STore
LD Rd,(Rb)     ;LoaD
MOV Rd, Rf     ;MOVe
```

- Directo de memoria (o absoluto) dir: *la dirección de memoria del dato* se suministra en *la propia instrucción*:

```
STS dir,Rf     ;STore Straight
LDS Rd,dir     ;LoaD Straight
```

En CS2 no existe direccionamiento inmediato (el dato no puede ser suministrado en la propia instrucción)

Conjunto de Instrucciones (ISP) del CS2

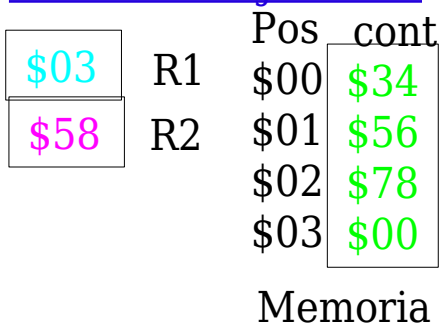


CO	SINTAXIS	FUNCIÓN	Direccionamiento
000	ST (Rb),Rf	MEMDAT(Rb) ← Rf	**Indirecto** de Registro
001	LD Rd, (Rb)	Rd ← MEMDAT(Rb)	**Indirecto** de Registro
010	STS dir, Rf	MEMDAT(dir) ← Rf	**Directo** de memoria
011	LDS Rd,dir	Rd ← MEMDAT(dir)	**Directo** de memoria
100	ADD Rd,Rf	Rd ← Rd+Rf	Directo de **Registro**
110	SUB Rd,Rf	Rd ← Rd-Rf	Directo de **Registro**
101	MOV Rd,Rf	Rd ← Rf	Directo de **Registro**
111	STOP	NOP	

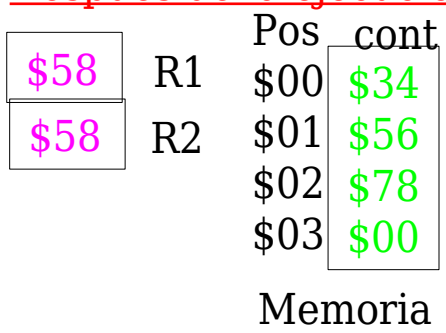
MOV Rd,Rf

MOV R1,R2

Antes de la ejecución



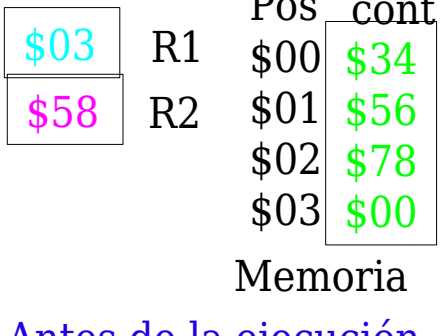
Después de la ejecución



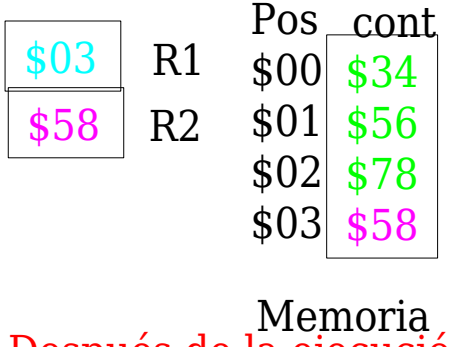
ST (Rb),Rf

ST (R1),R2

Antes de la ejecución



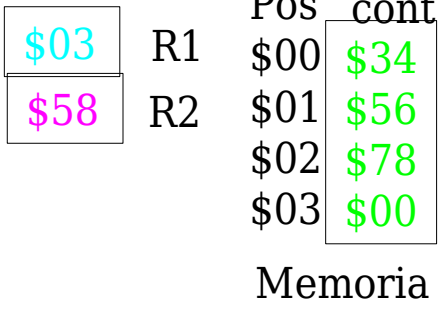
Después de la ejecución



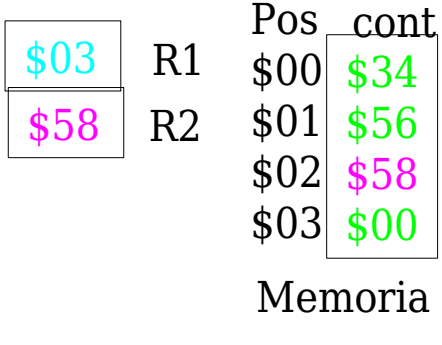
STS dir, Rf

STS $2,R2

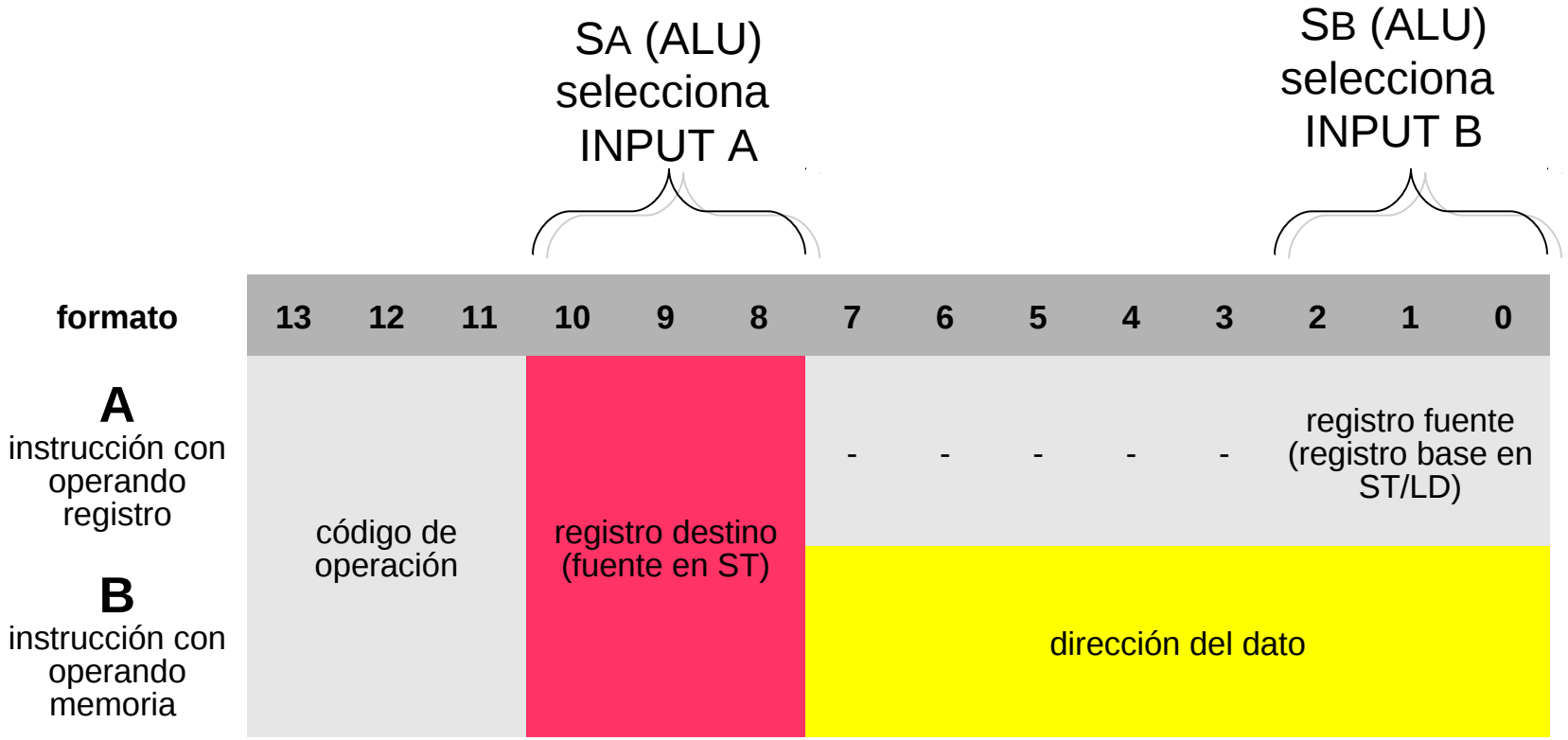
Antes de la ejecución



Después de la ejecución



Formato de instrucciones del CS2

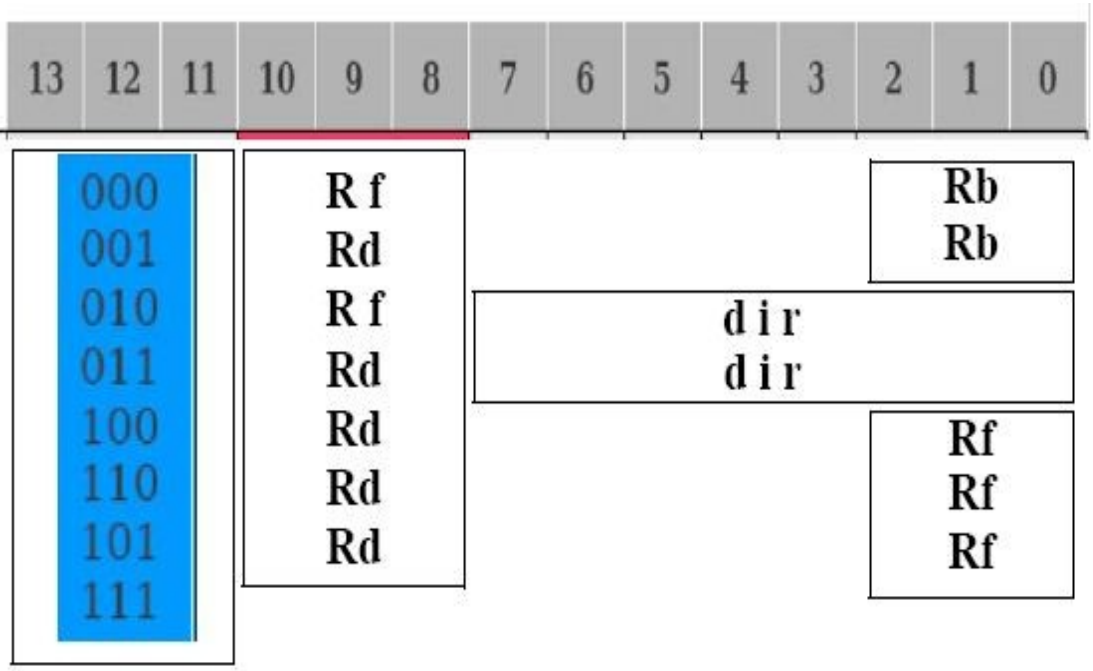


- Las instrucciones son de 14 bits y siguen ocupando una palabra de memoria.
- En el CS2 el ancho de la memoria de código será de 14 bits.
- El registro IR debe ser también de 14 bits

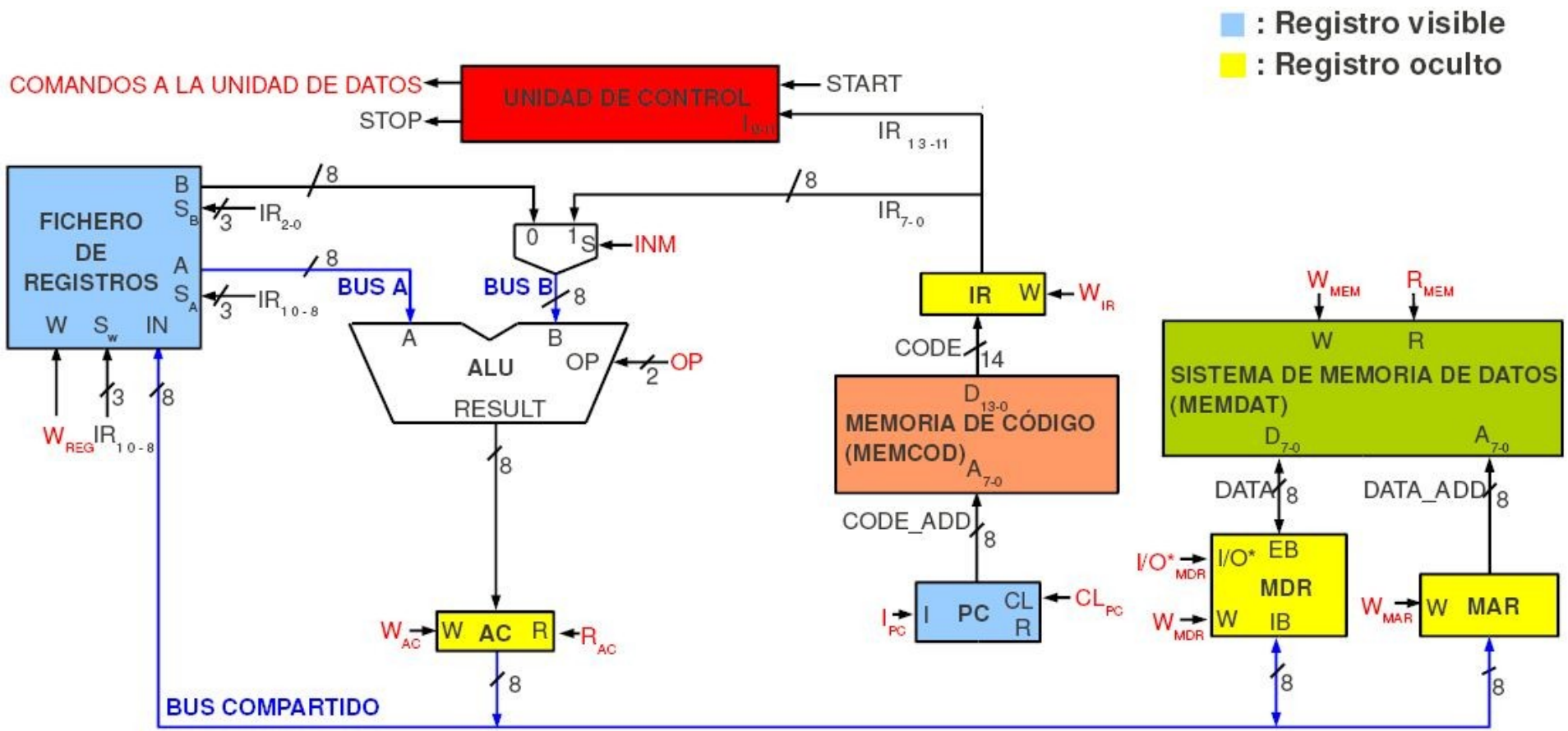
Formato de instrucciones del CS2

formato	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A instrucción con operando registro	código de operación			registro destino (fuente en ST)			-	-	-	-	-	registro fuente (registro base en ST/LD)		
B instrucción con operando memoria							dirección del dato							

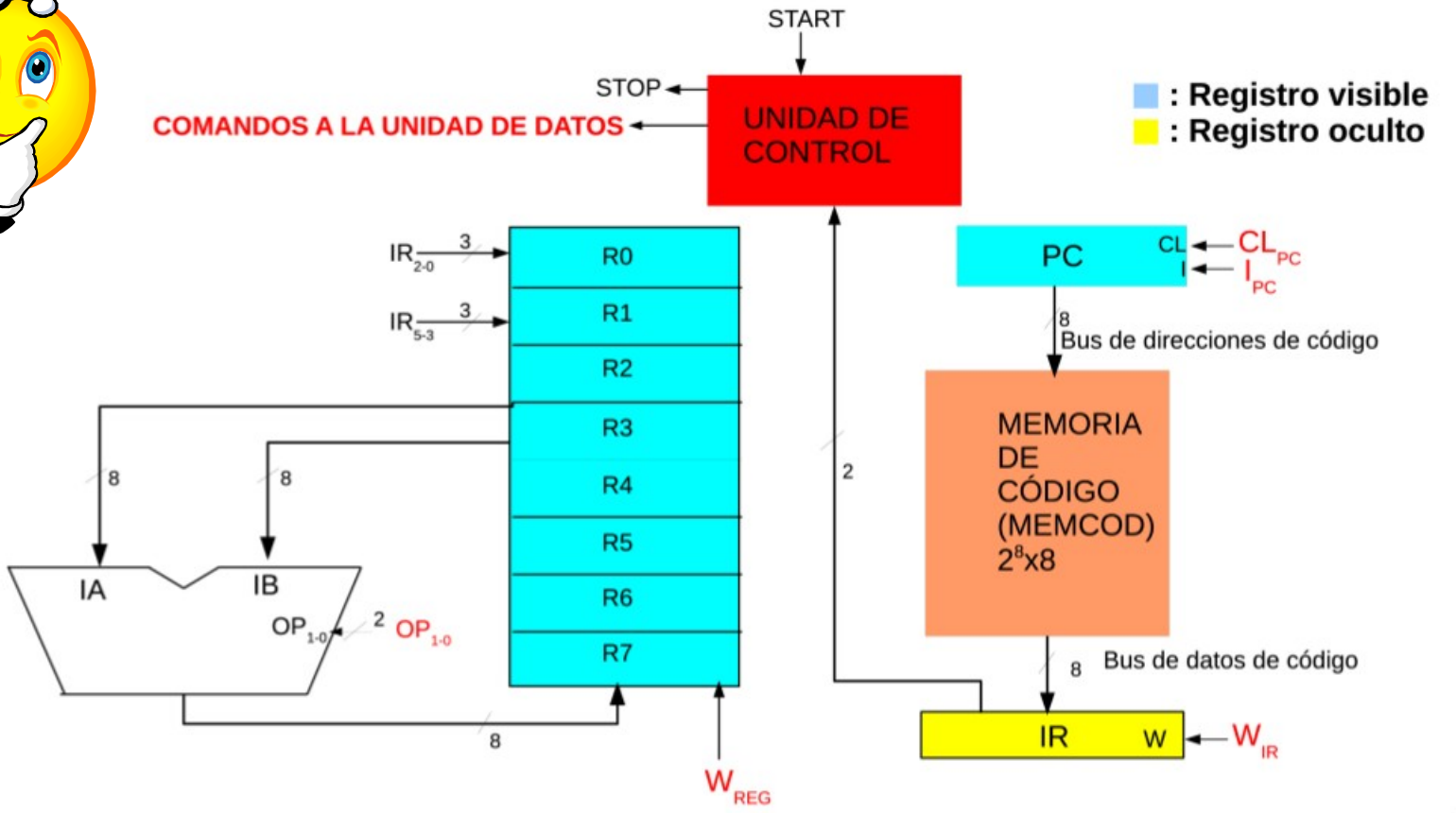
CO	SINTAXIS
000	ST (Rb),Rf
001	LD Rd, (Rb)
010	STS dir, Rf
011	LDS Rd,dir
100	ADD Rd,Rf
110	SUB Rd,Rf
101	MOV Rd,Rf
111	STOP



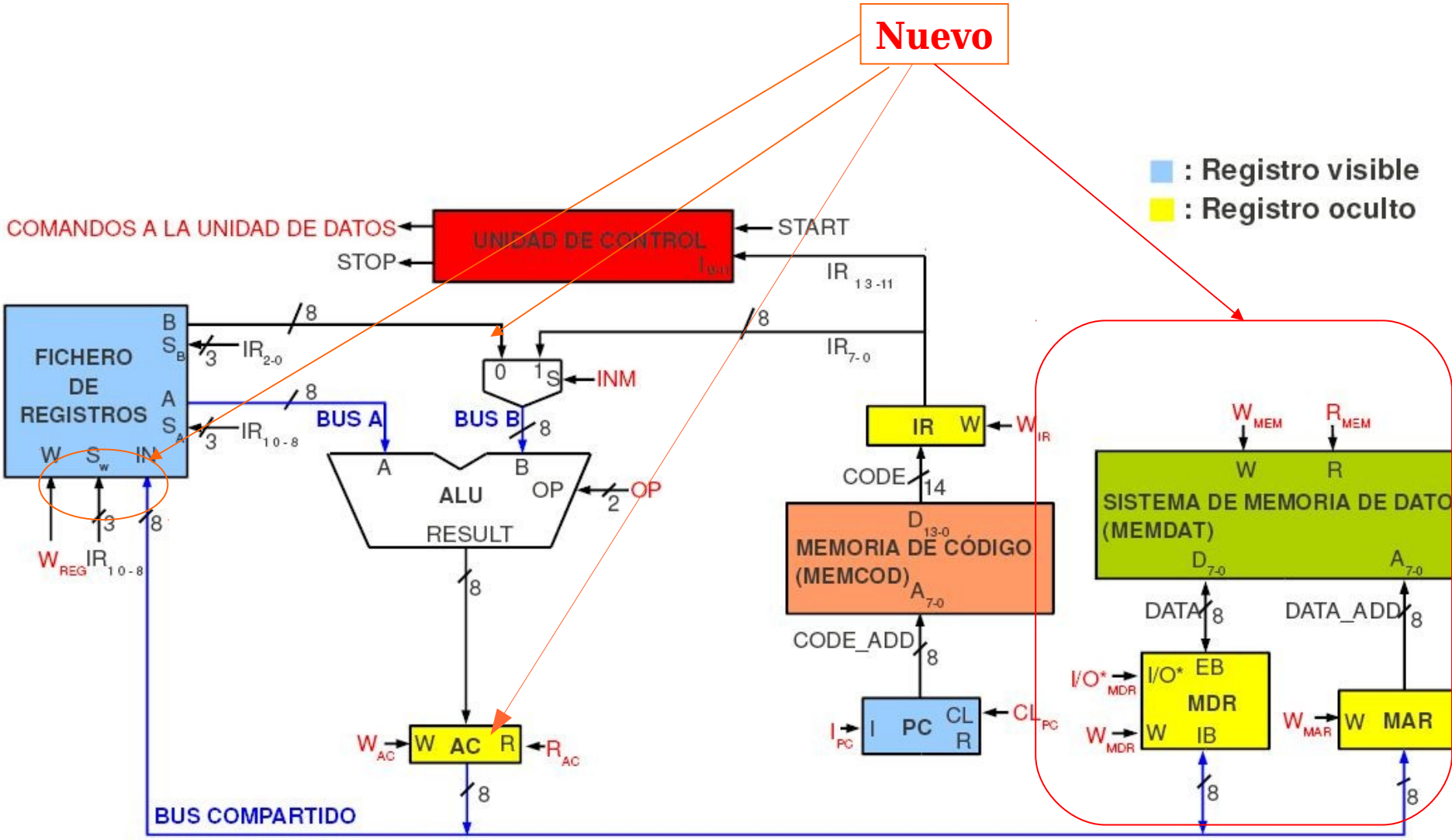
Arquitectura del CS2



Arquitectura del computador CS1

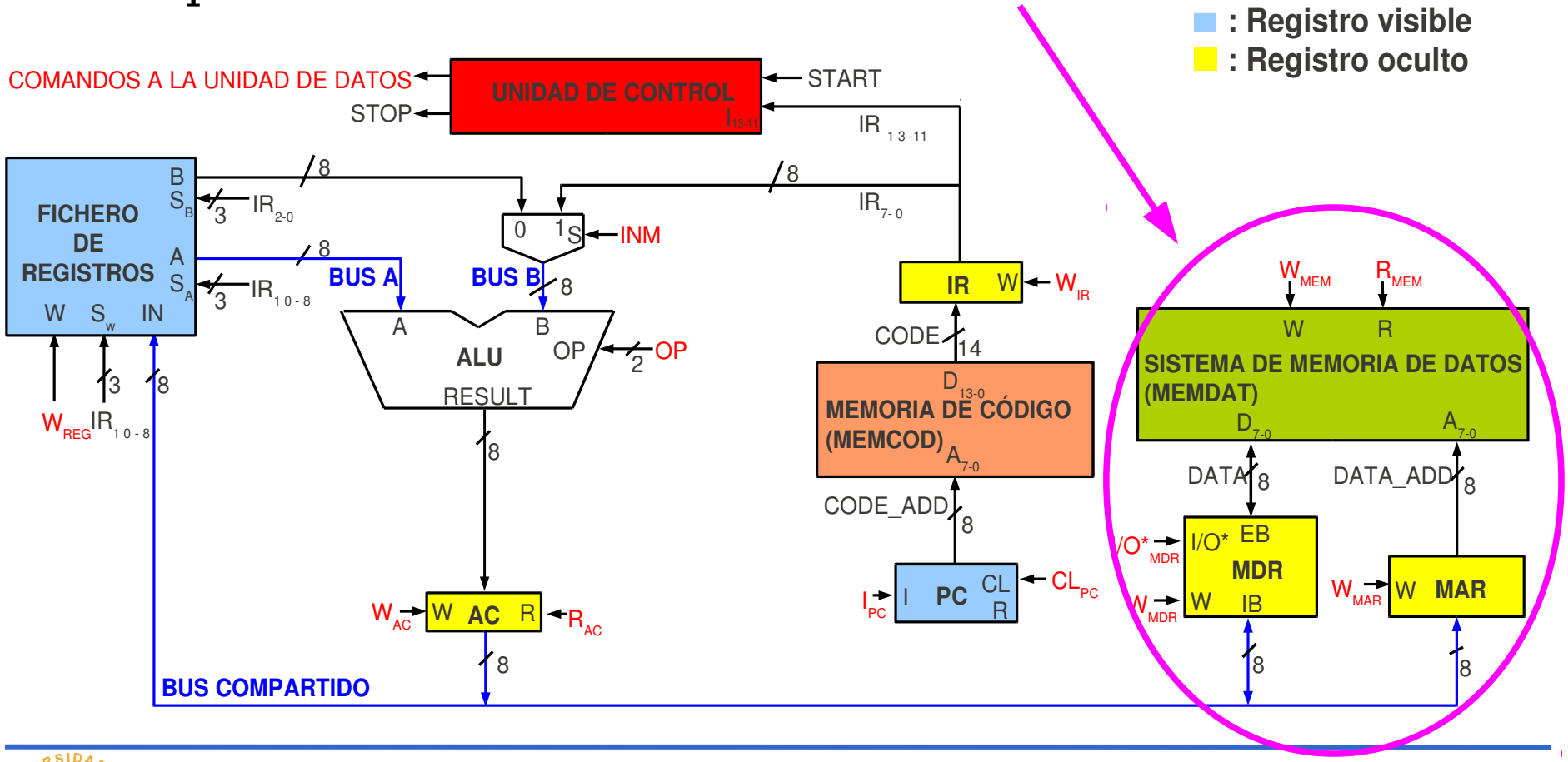


Arquitectura del CS2

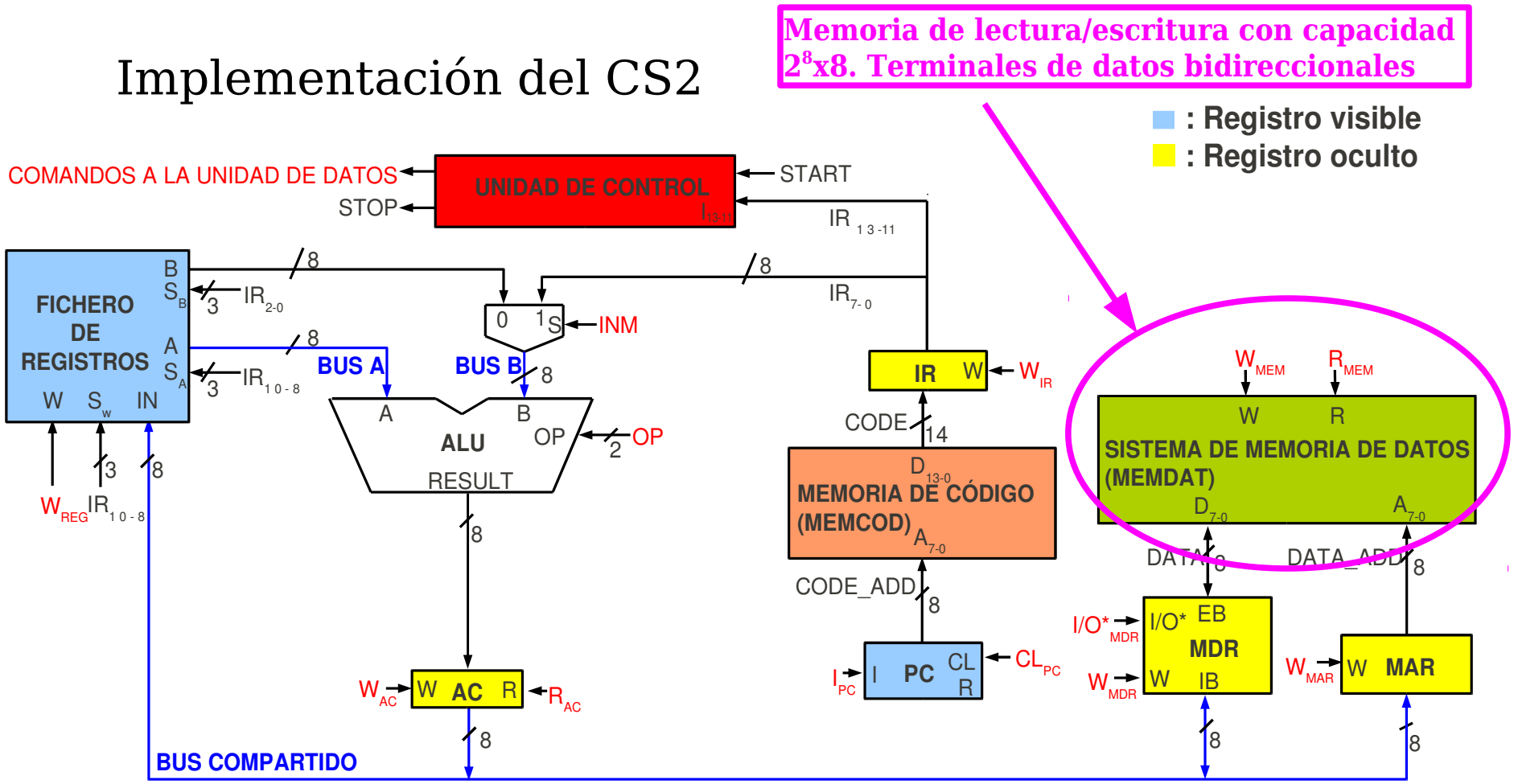


Implementación del CS2

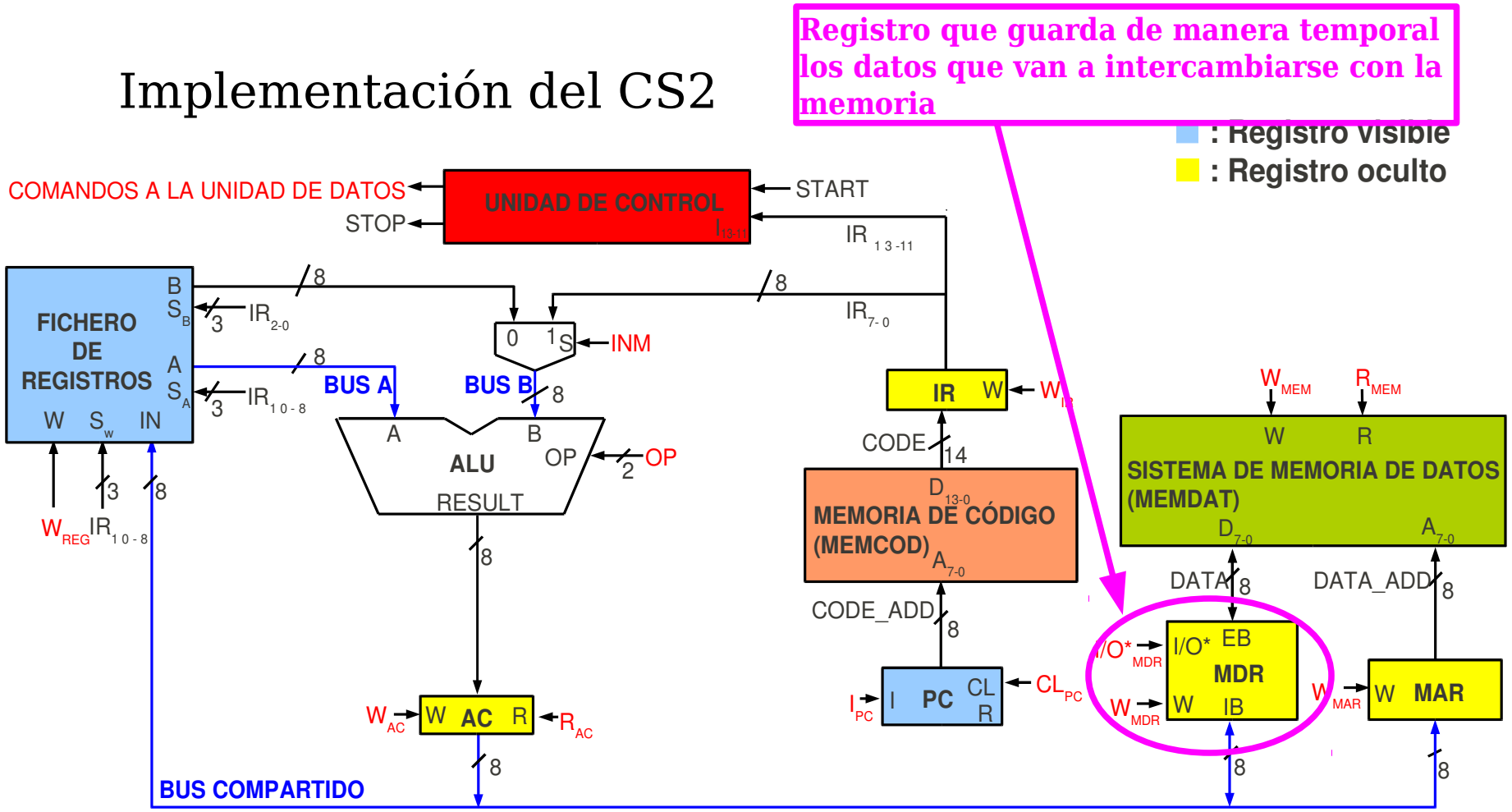
Parte añadida con objeto de guardar mas datos Además de los guardados en el Set de registros



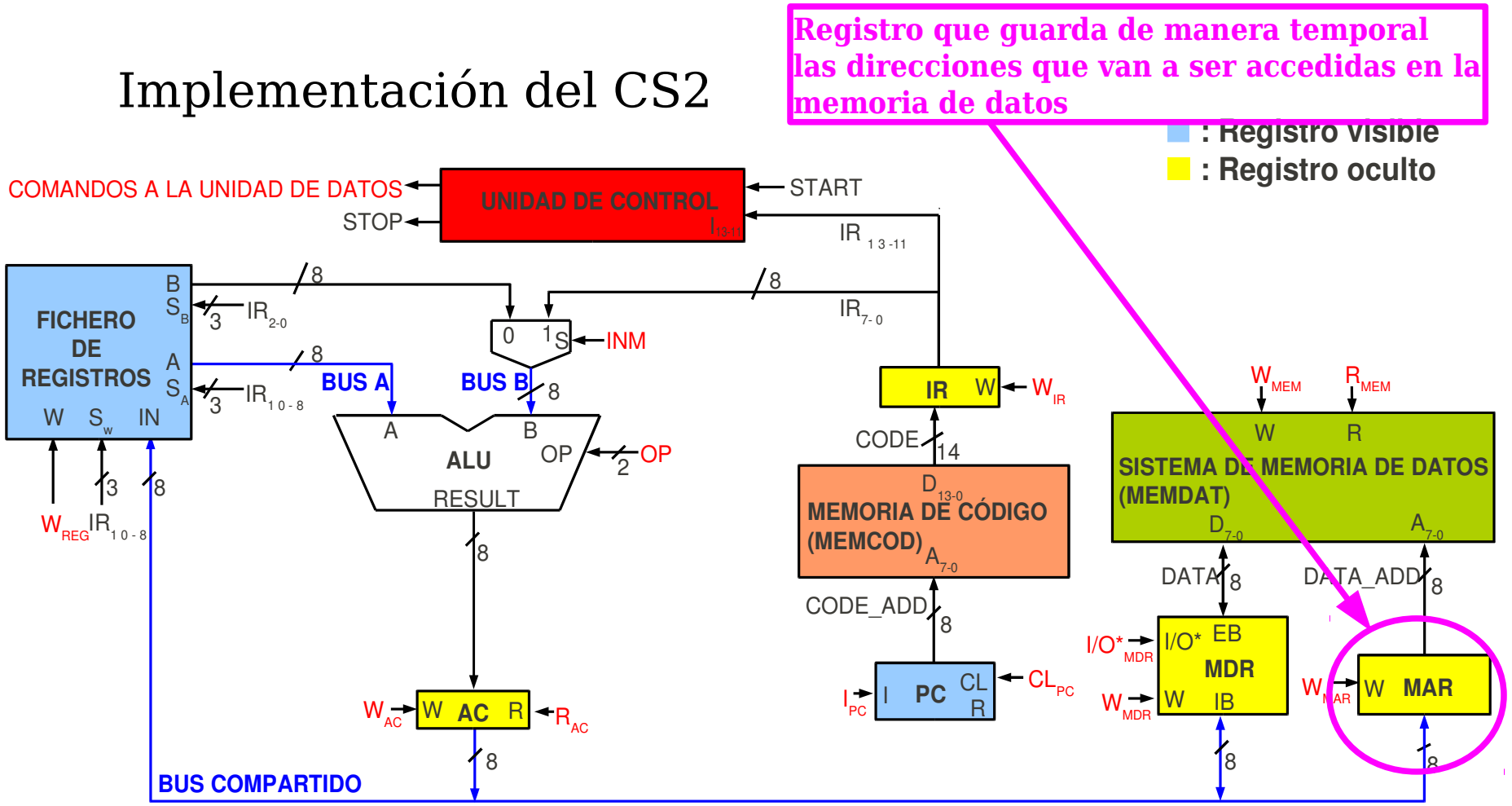
Implementación del CS2



Implementación del CS2

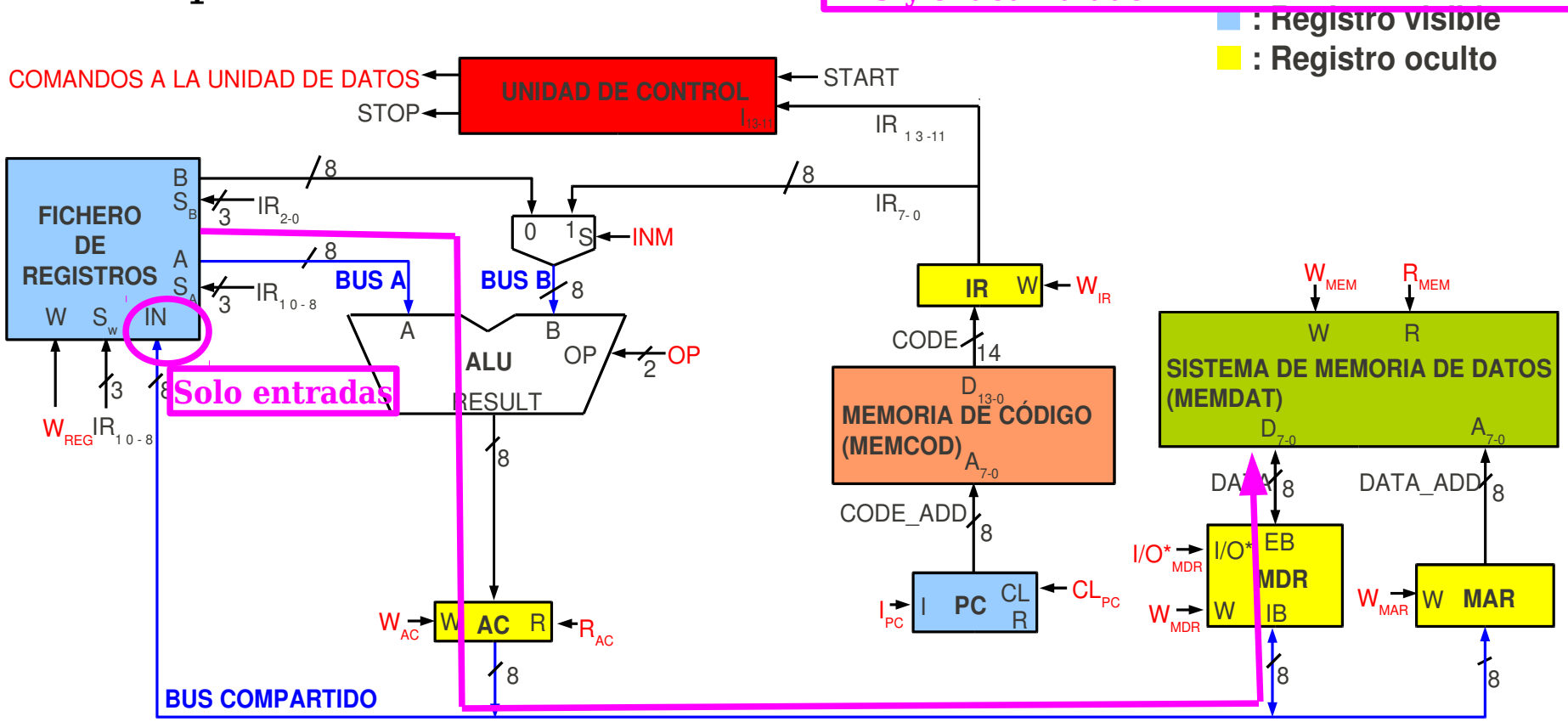


Implementación del CS2



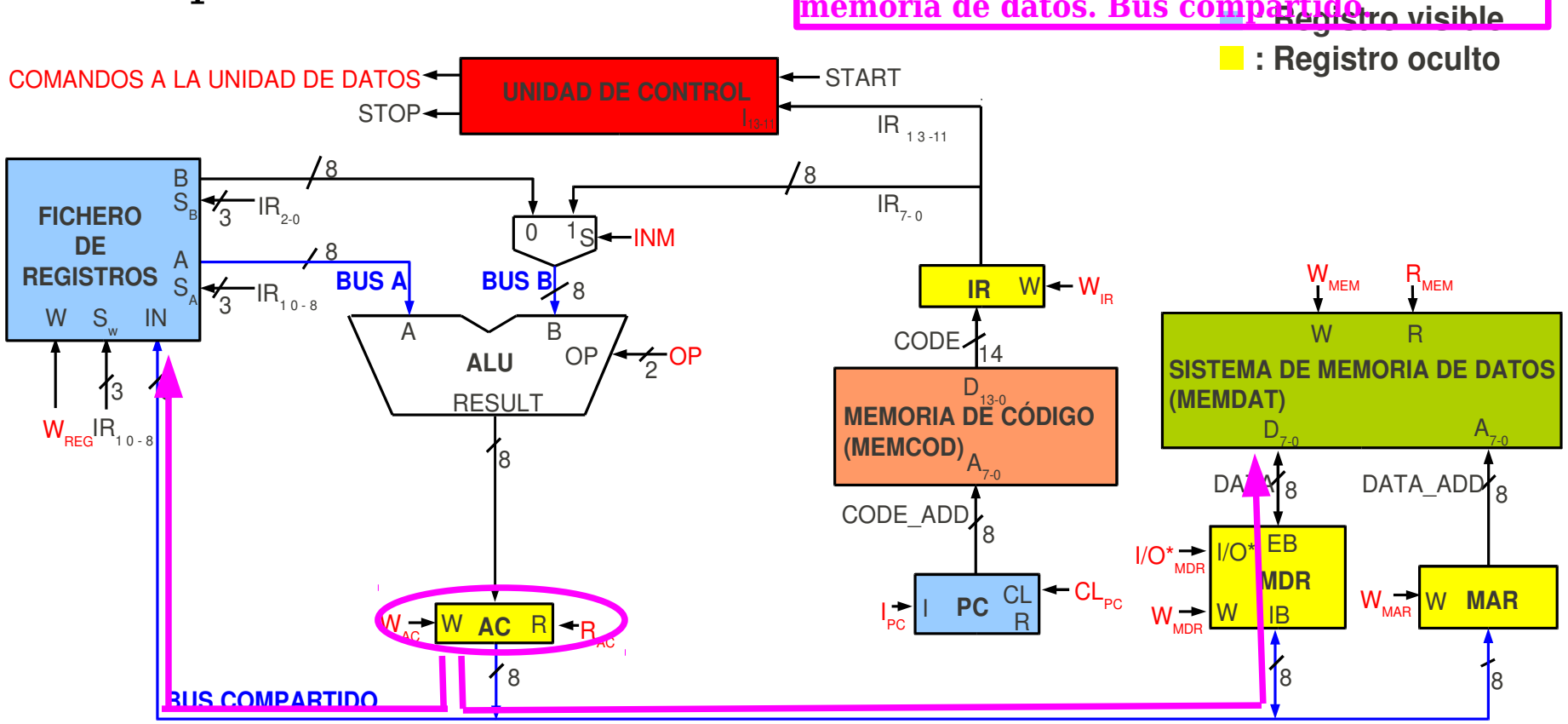
Implementación del CS2

El camino para intercambiar datos entre el set De registros y la memoria siempre pasa por la ALU y el acumulador



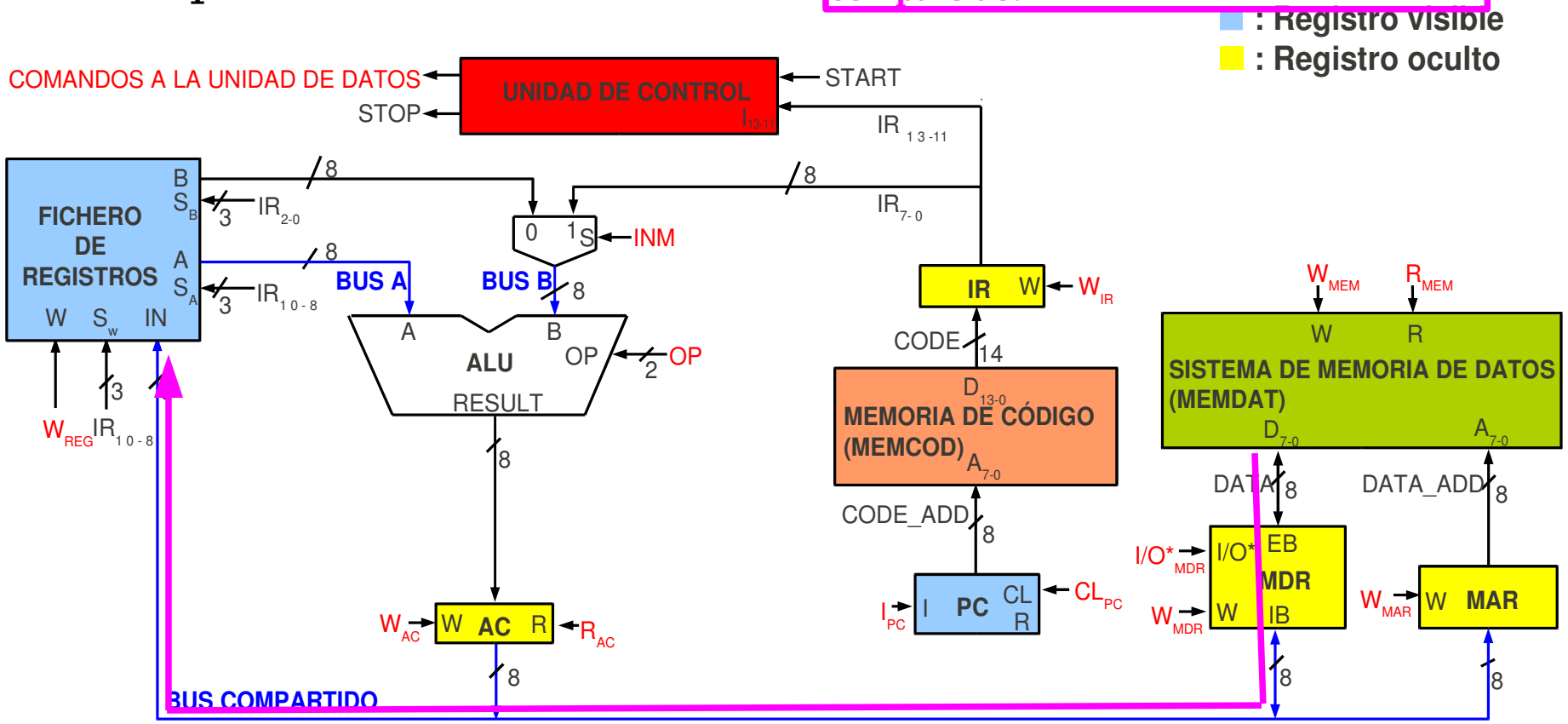
Implementación del CS2

Ha sido necesario añadir un Acumulador para conectar la salida de la ALU tanto con la pila de registros como con el sistema de memoria de datos. Bus compartido



Implementación del CS2

Camino para que los datos vayan de la memoria a los registros. Usa el bus compartido.

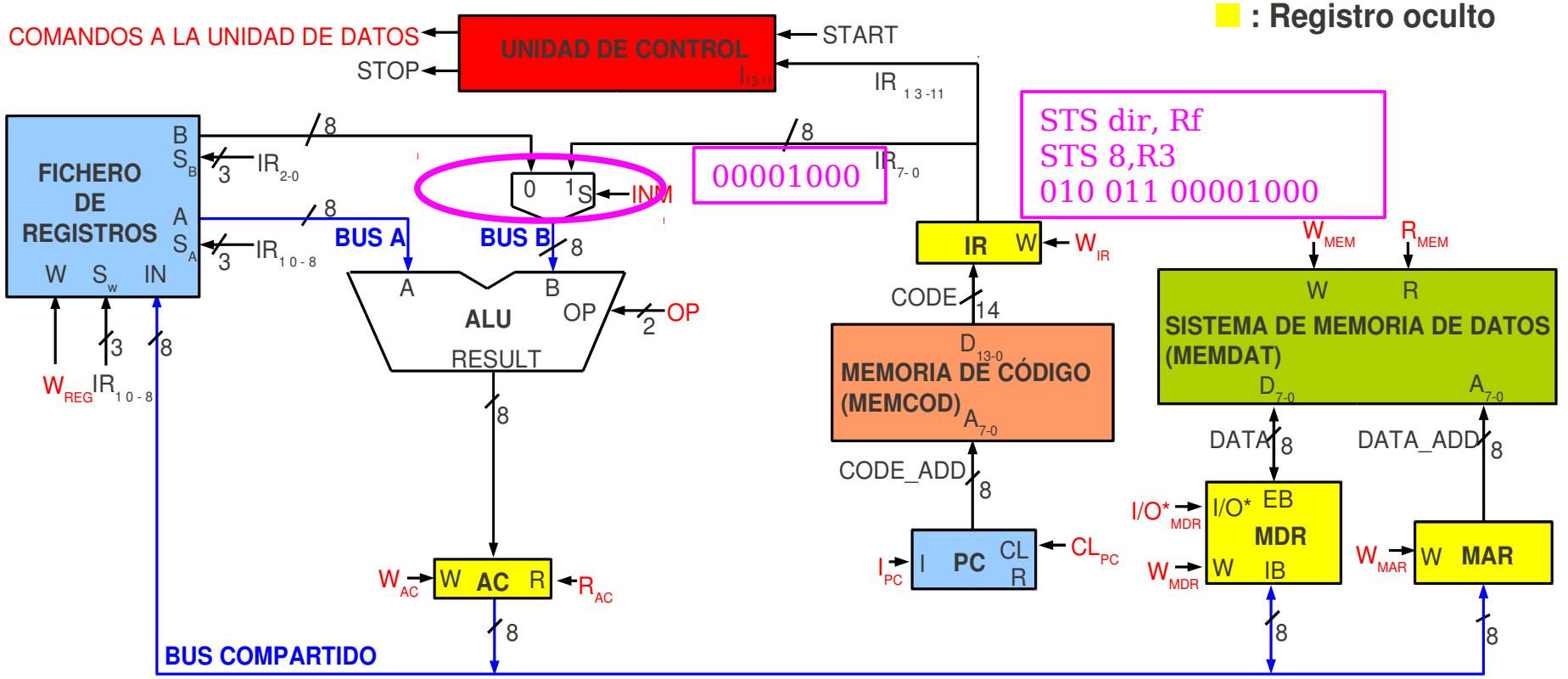


formato	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A	código de operación			registro destino (fuente en ST)			-	-	-	-	-	registro fuente (registro base en ST/LD)		
B							dirección del dato							

Direccionamiento absoluto. Necesidad del MUX a la entrada de la ALU

Implementación del CS2

: Registro visible
: Registro oculto

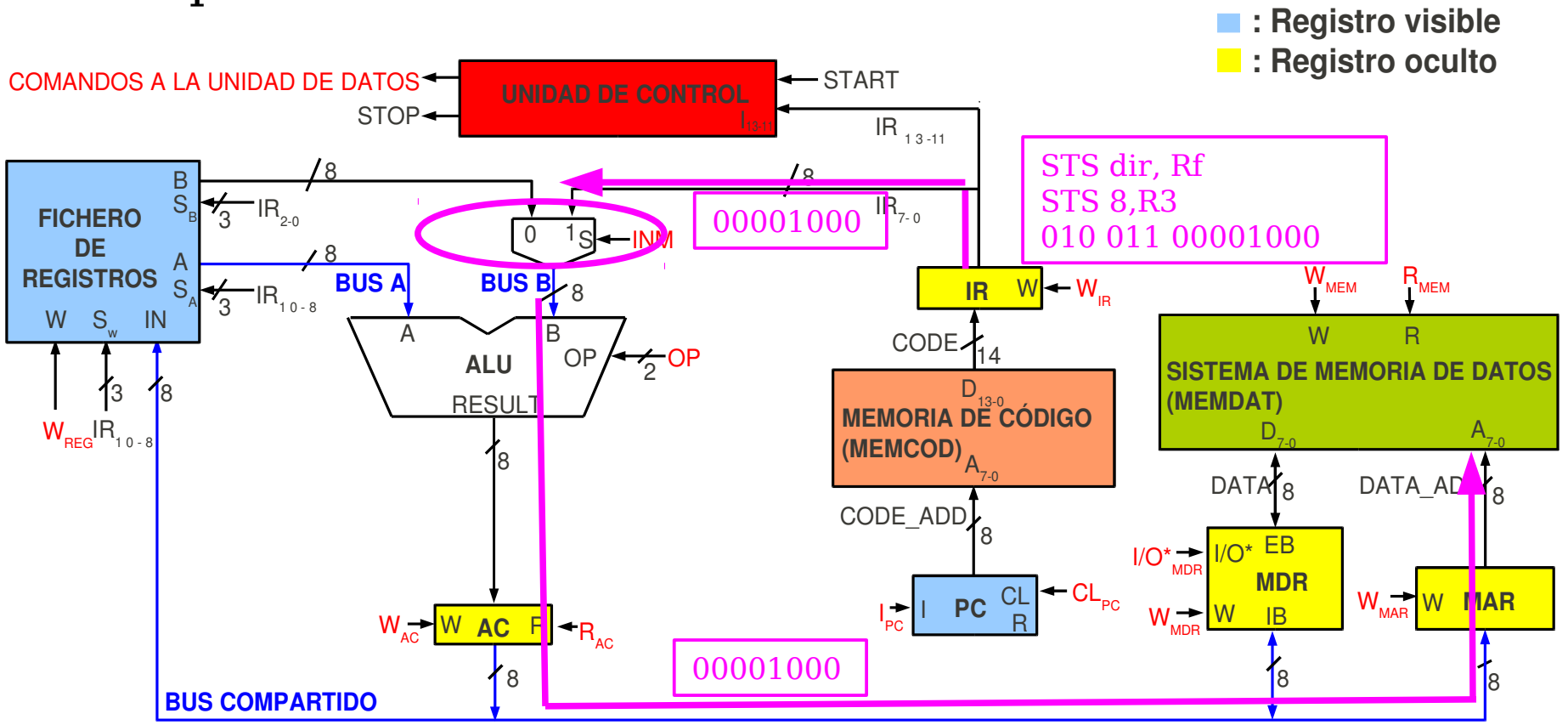


STS dir, Rf
STS 8,R3
010 011 00001000

formato	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A	código de operación			registro destino (fuente en ST)			-	-	-	-	-	registro fuente (registro base en ST/LD)		
B							dirección del dato							

Direccionamiento absoluto. Necesidad del MUX a la entrada de la ALU

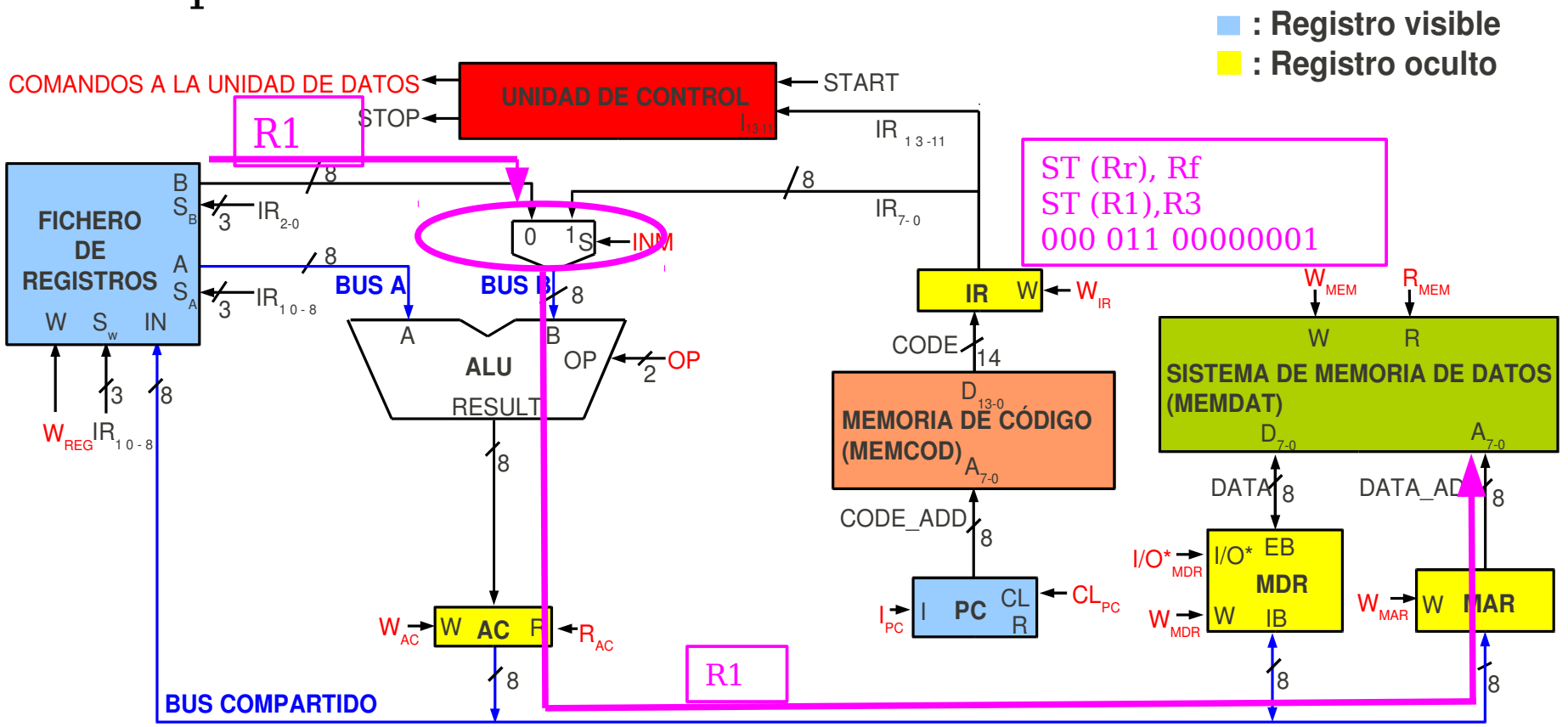
Implementación del CS2



formato	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A	código de operación			registro destino (fuente en ST)			-	-	-	-	-	registro fuente (registro base en ST/LD)		
B							dirección del dato							

Implementación del CS2

Direccionamiento indirecto. Necesidad del MUX a la entrada de la ALU

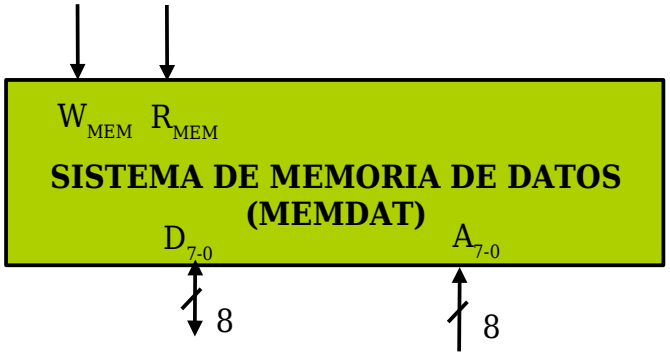


Arquitectura del CS2

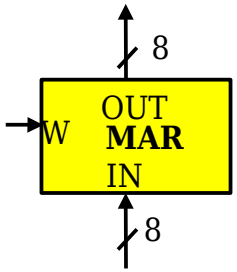
Modificaciones en la arquitectura del CS2 (todo lo que se ha añadido es para dotar al sistema de almacenamiento de datos en memoria).

- Sistema de memoria de datos.
- Registro de datos de la memoria (MDR).
- Registro de direcciones de la memoria (MAR).
- Acumulador (AC). Se ha añadido este registro a la salida de la ALU porque ésta debe compartir el bus.
- Multiplexor a la entrada de la ALU porque ésta debe transferir datos tanto de los registros como de los 8 bits menos significativos del registro IR a MAR en el caso de direccionamiento absoluto.

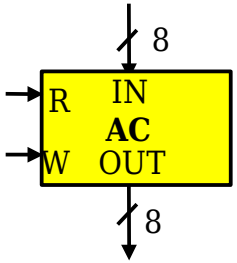
Descripción RT de los nuevos componentes del CS2



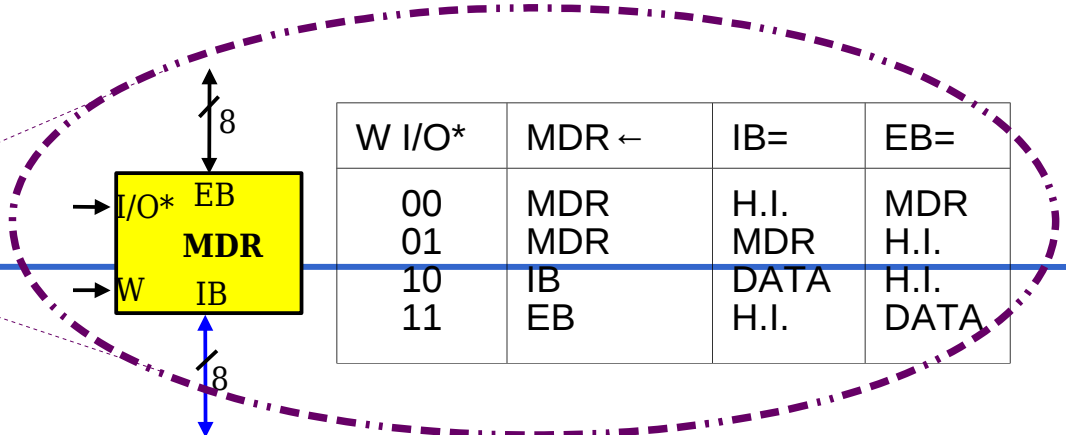
W_{MEM}	R_{MEM}	MEMDAT[A_{7-0}] ←	D_{7-0} =
0	0	MEMDAT[A_{7-0}]	H.I.
0	1	MEMDAT[A_{7-0}]	MEMDAT[A_{7-0}]
1	0	D_{7-0}	DATA
1	1	PROHIBIDO	----



W	MAR ←	OUT=
0	MAR	MAR
1	IN	MAR



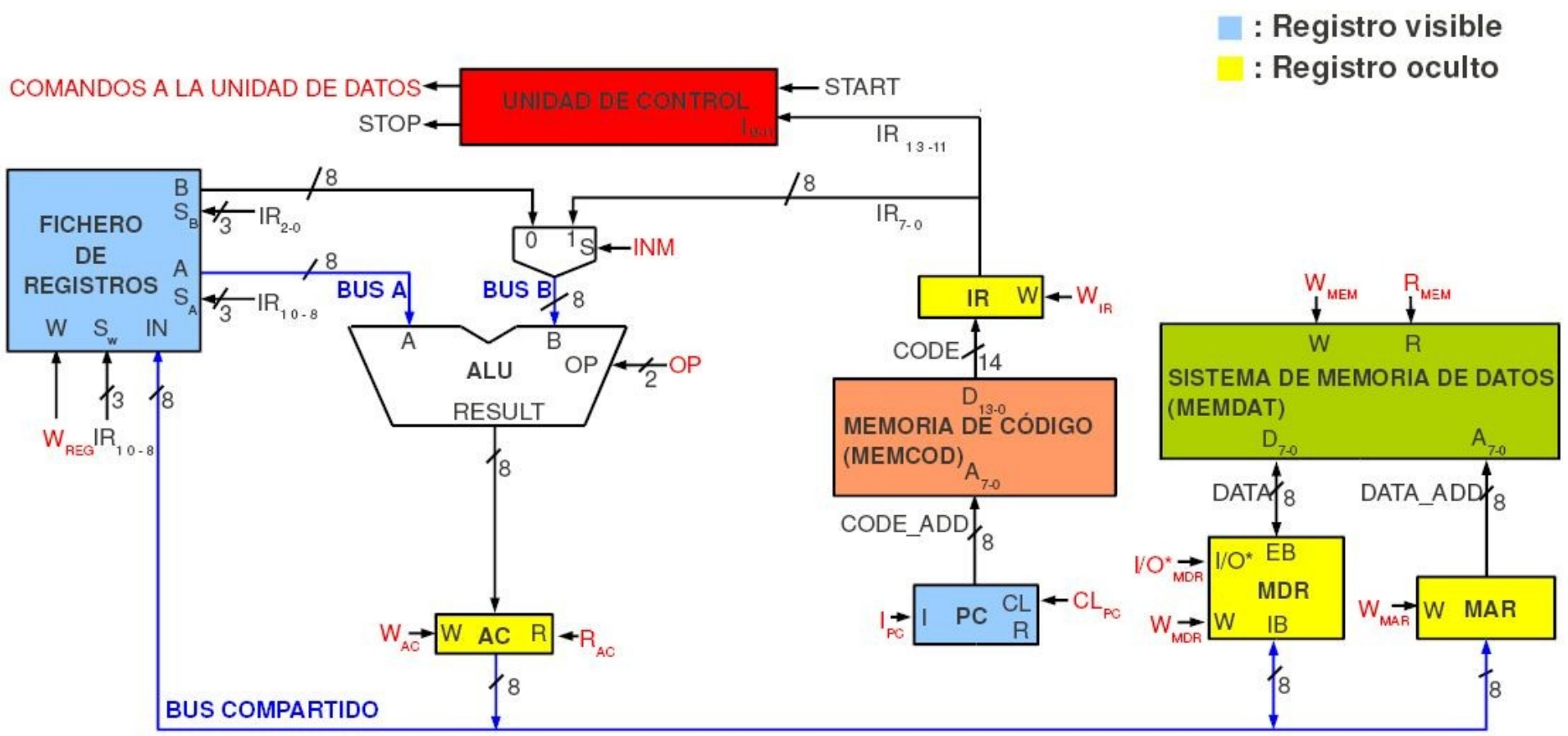
WR	AC ←	OUT=
00	AC	H.I.
01	AC	AC
10	IN	H.I.
11	IN	AC



W I/O*	MDR ←	IB=	EB=
00	MDR	H.I.	MDR
01	MDR	MDR	H.I.
10	IB	DATA	H.I.
11	EB	H.I.	DATA

Ejemplo de ejecución de instrucciones: ST(Rb),Rf

(MEMDAT(Rb) ← Rf)

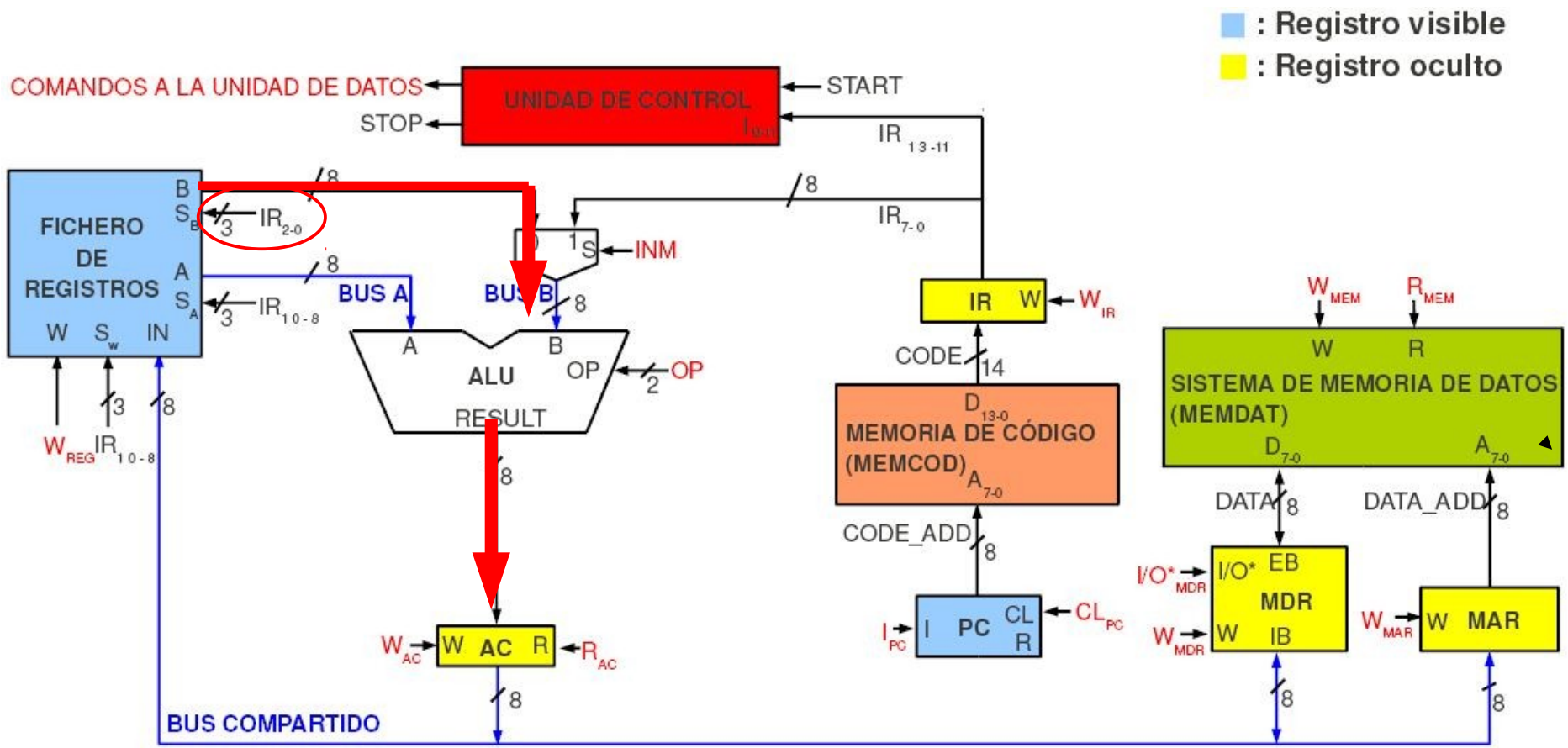


formato A	13	12	11	10	9	8	7	6	5	4	3	2	1	0
instrucción con operando registro	código de operación			registro fuente en ST			-	-	-	-	-	registro base en ST		

Los valores almacenados en IR deben corresponder a los registros **base** y **fuente**

Ejemplo de ejecución de instrucciones: ST(Rb),Rf

1. AC ← REG[IR_{2-0}] $OP_1OP_0W_{AC}$ (MEMDAT(Rb) ← Rf)



formato	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A instrucción con operando registro	código de operación			registro fuente en ST			-	-	-	-	-	registro base en ST		

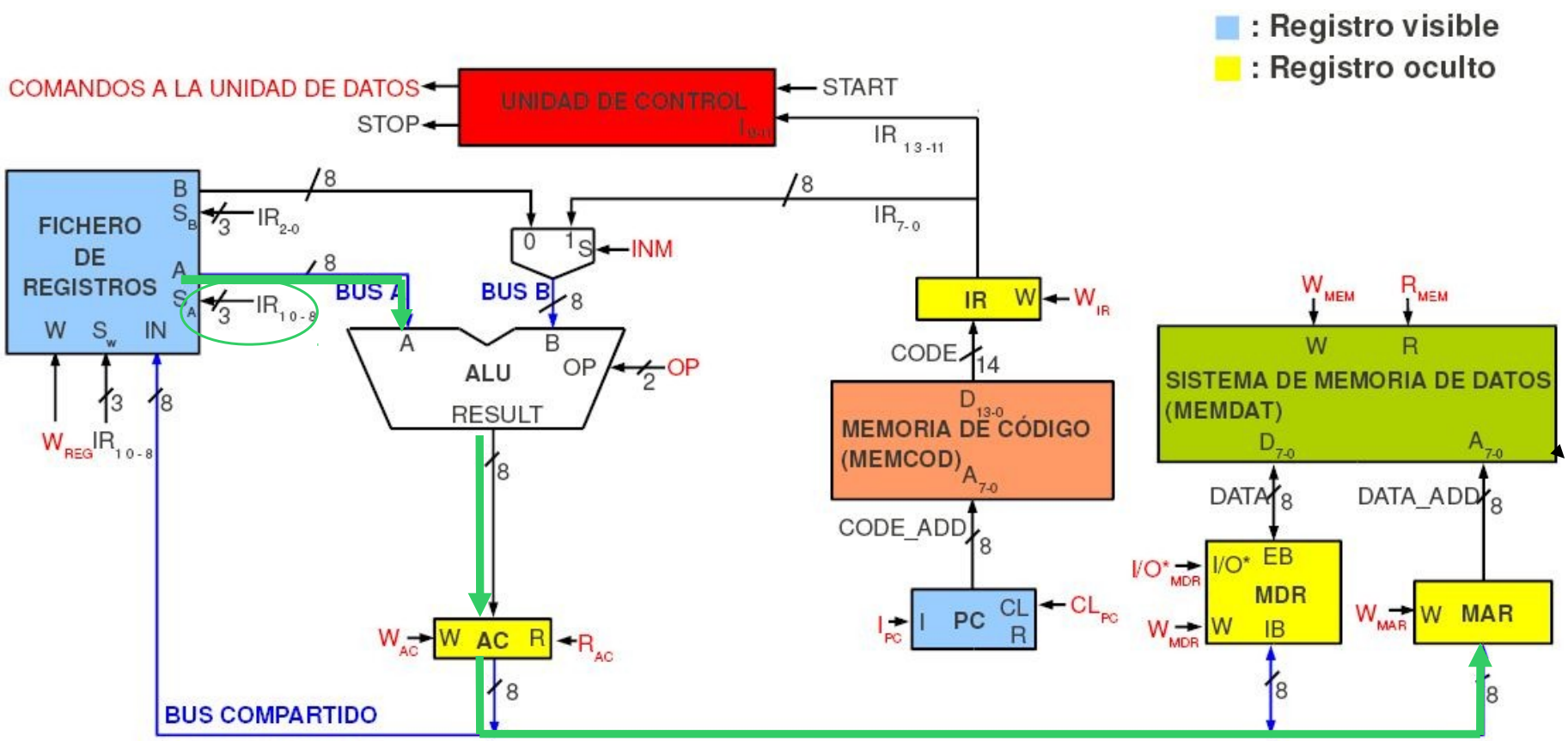
Los valores almacenados en IR deben corresponder a los registros **base** y **fuente**

Ejemplo de ejecución de instrucciones: ST(Rb),Rf

1. AC ← REG[IR_{2-0}] $OP_1 OP_0 W_{AC}$

2. MAR ← AC; AC ← REG[IR_{10-8}] $R_{AC} W_{MAR} OP_0 W_{AC}$

(MEMDAT(Rb) ← Rf)



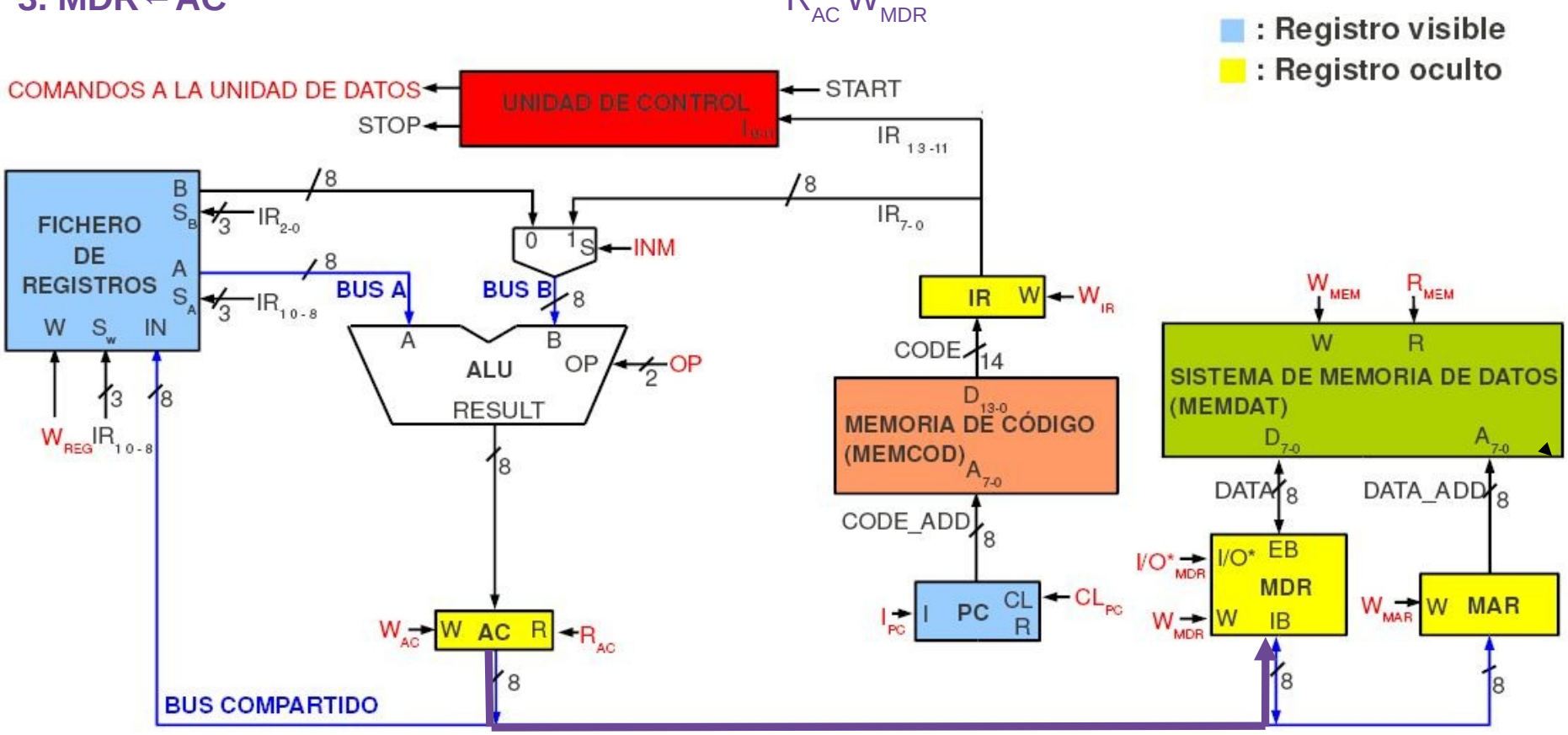
formato	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A instrucción con operando registro	código de operación			registro fuente en ST			-	-	-	-	-	registro base en ST		

Los valores almacenados en IR deben corresponder a los registros **base** y **fuente**

Ejemplo de ejecución de instrucciones: ST(Rb),Rf

(MEMDAT(Rb) ← Rf)

1. AC ← REG[IR_{2-0}] — $OP_1 OP_0 W_{AC}$
2. MAR ← AC; AC ← REG[IR_{10-8}] — $R_{AC} W_{MAR} OP_0 W_{AC}$
3. MDR ← AC — $R_{AC} W_{MDR}$

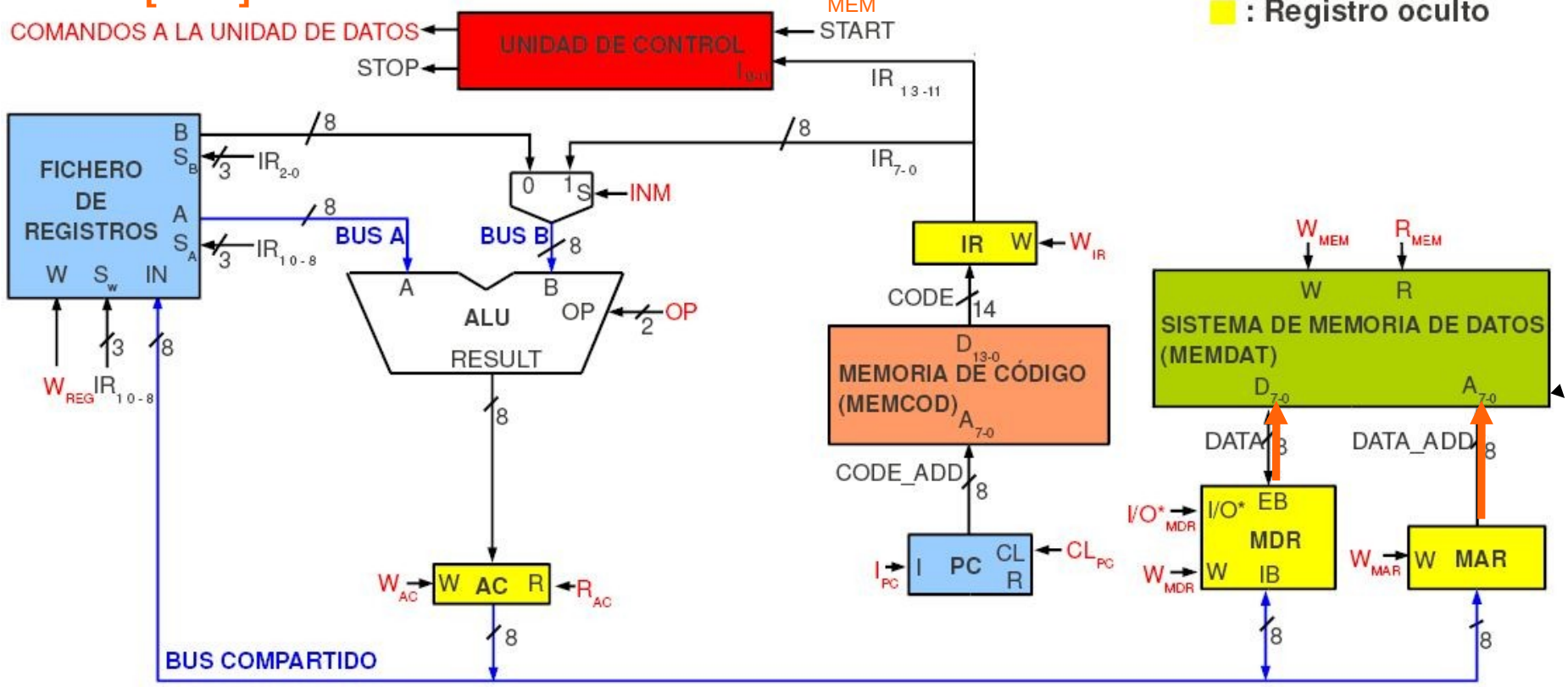


formato	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A instrucción con operando registro	código de operación			registro fuente en ST			-	-	-	-	-	registro base en ST		

Ejemplo de ejecución de instrucciones: ST(Rb),Rf

(MEMDAT(Rb) ← Rf)

1. AC ← REG[IR_{2-0}] — $OP_1 OP_0 W_{AC}$
2. MAR ← AC; AC ← REG[IR_{10-8}] — $R_{AC} W_{MAR} OP_0 W_{AC}$
3. MDR ← AC — $R_{AC} W_{MDR}$
4. MEM[MAR] ← MDR — W_{MEM}



formato	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A instrucción con operando registro	código de operación			registro fuente en ST			-	-	-	-	-	registro base en ST		

Los valores almacenados en IR deben corresponder a los registros **base** y **fuente**

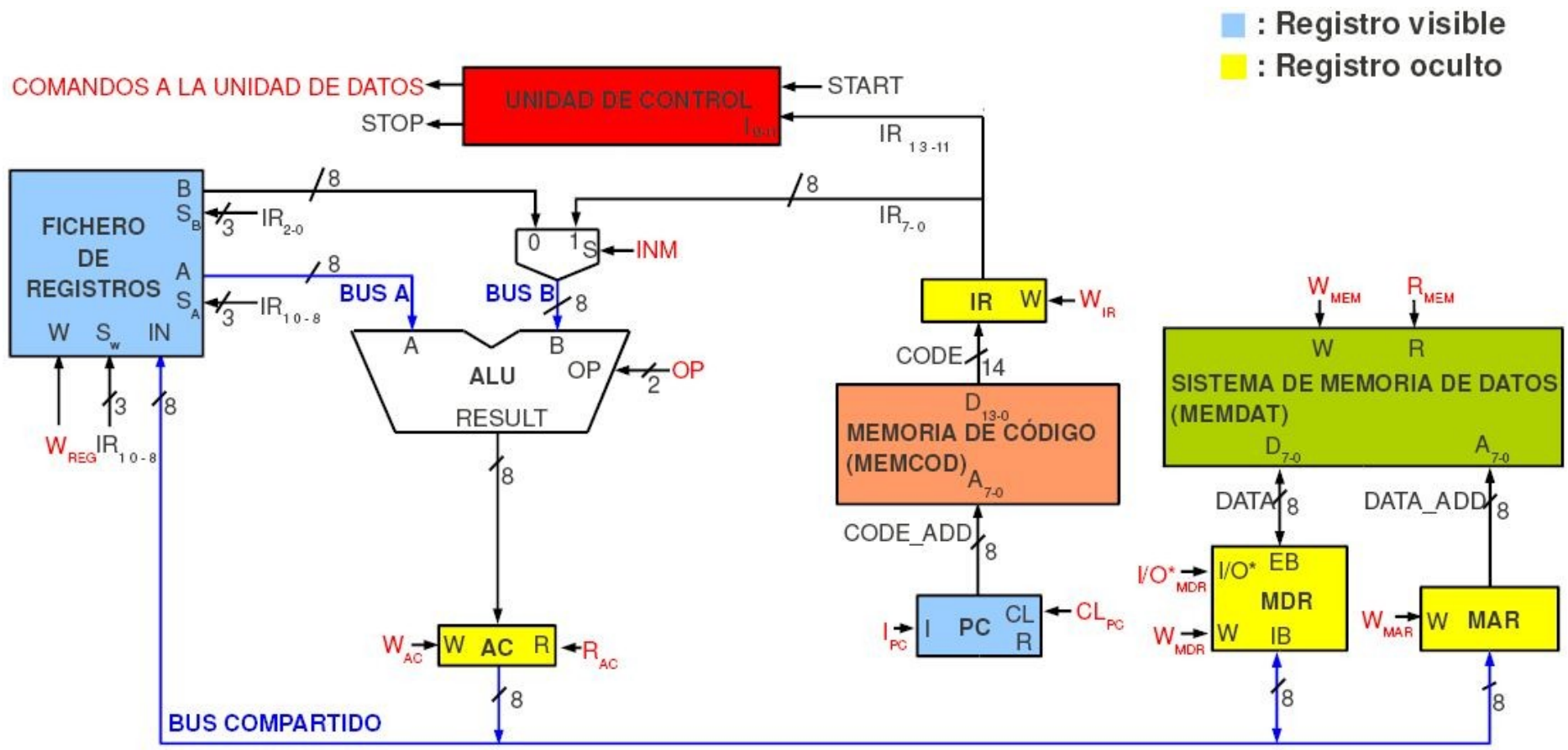
Descripción de las nuevas instrucciones del CS2

CO (IR_{13} IR_{12} IR_{11})	000	001	010	011
Instrucción	**ST (Rb),Rf**	**LD Rd,(Rb)**	**STS dir,Rf**	**LDS Rd,dir**
Micro 1	**AC ← Rb** $OP_1 OP_0 W_{AC}$	**AC ← Rb** $OP_1 OP_0 W_{AC}$	**AC ← dir** $OP_1 OP_0 W_{AC}$INM	**AC ← dir** $OP_1 OP_0 W_{AC}$INM
Micro 2	**MAR ← AC** $R_{AC} W_{MAR}$ **AC ← Rf** $OP_0 W_{AC}$	**MAR ← AC** $R_{AC} W_{MAR}$	**MAR ← AC** $R_{AC} W_{MAR}$ **AC ← Rf** $OP_0 W_{AC}$	**MAR ← AC** $R_{AC} W_{MAR}$
Micro 3	**MDR ← AC** $R_{AC} W_{MDR}$	**MDR ← MEM(MAR)** $R_{MEM} W_{MDR}$ I/O*$_{MDR}$	**MDR ← AC** $R_{AC} W_{MDR}$	**MDR ← MEM(MAR)** $R_{MEM} W_{MDR}$ I/O_{MDR}
Micro 4	**MEM(MAR) ← MDR** W_{MEM}	**RD ← MDR** W_{REG} I/O_{MDR}	**MEM(MAR) ← MDR** W_{MEM}	**RD ← MDR** W_{REG} I/O_{MDR}

Las otras 4 instrucciones tienen una microoperación más que en el CS1 debido al AC (ver carta ASM del apéndice)

Ejemplo de ejecución de instrucciones: LDS Rd,dir

(Rd ← MEMDAT(dir))

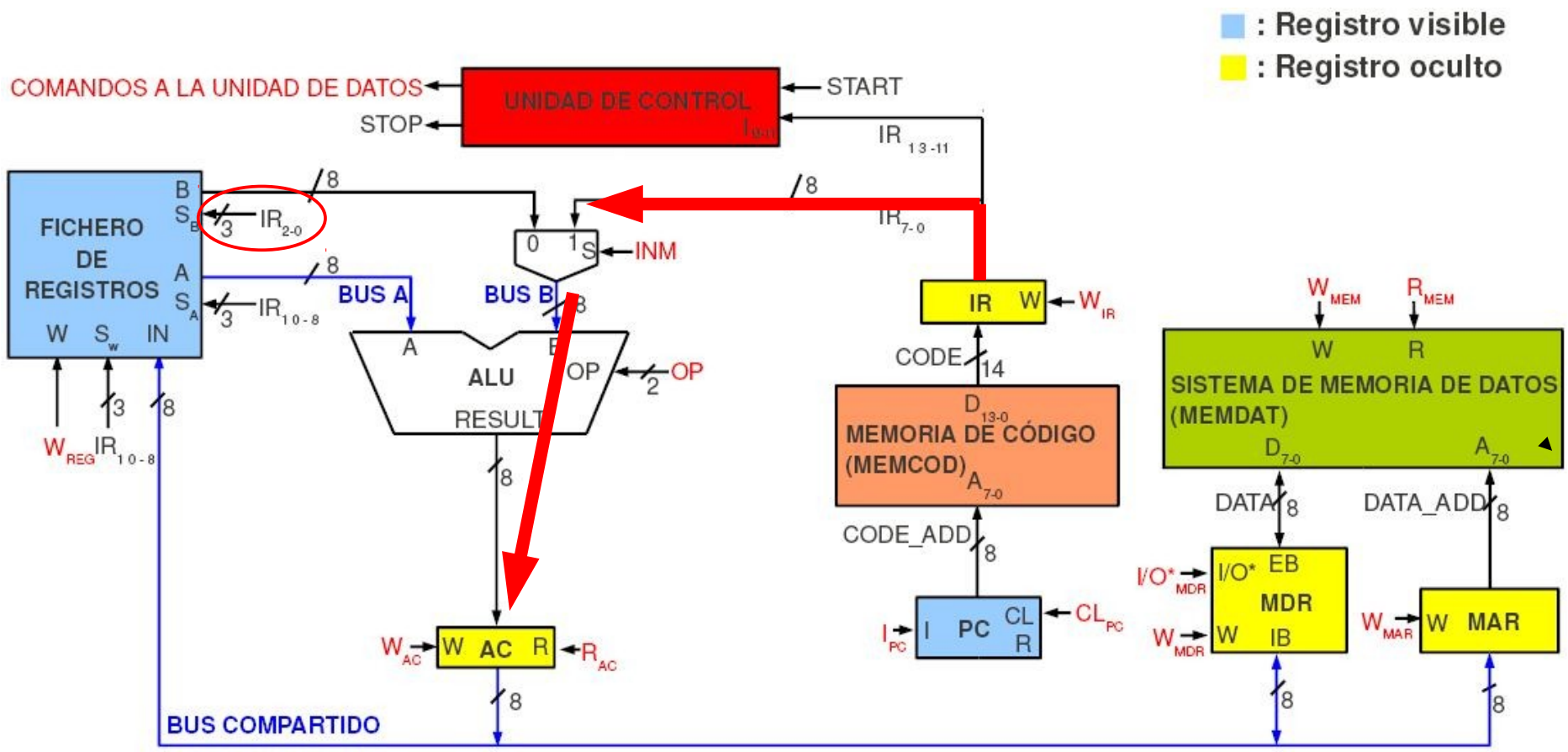


formato	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A instrucción con operando registro	código de operación			registro destino			Dirección del dato							

En IR debe aparecer la dirección del dato y el registro **destino**

Ejemplo de ejecución de instrucciones: LDS Rd,dir

1. AC ← IR_{7-0} $OP_1OP_0W_{AC}$ INM (Rd ← MEMDAT(dir))



formato	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A instrucción con operando registro	código de operación			registro destino			Dirección del dato							

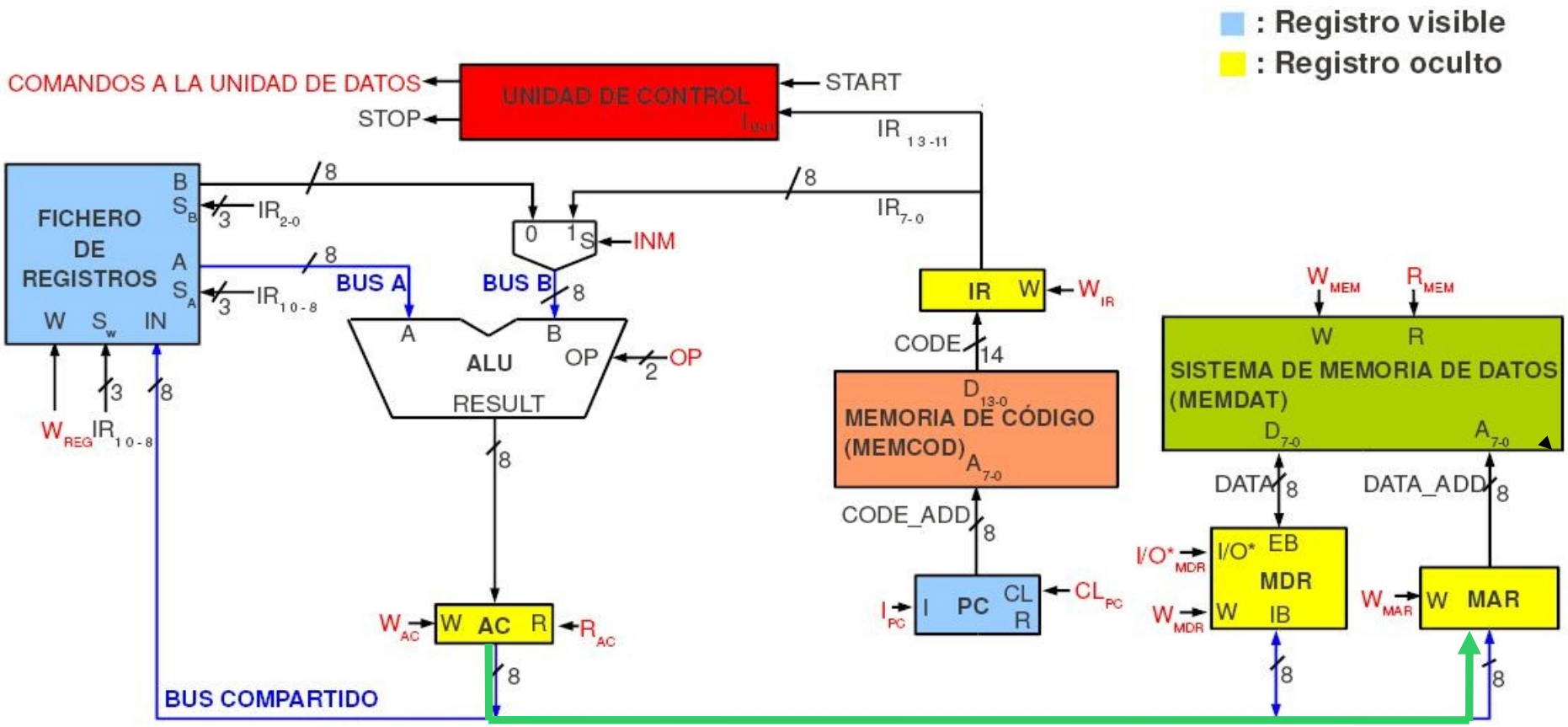
En IR debe aparecer la dirección del dato y el registro **destino**

Ejemplo de ejecución de instrucciones: LDS Rd,dir

1. AC ← IR_{7-0} $OP_1 OP_0 W_{AC}$ INM

2. MAR ← AC $R_{AC} W_{MAR}$

(Rd ← MEMDAT(dir))



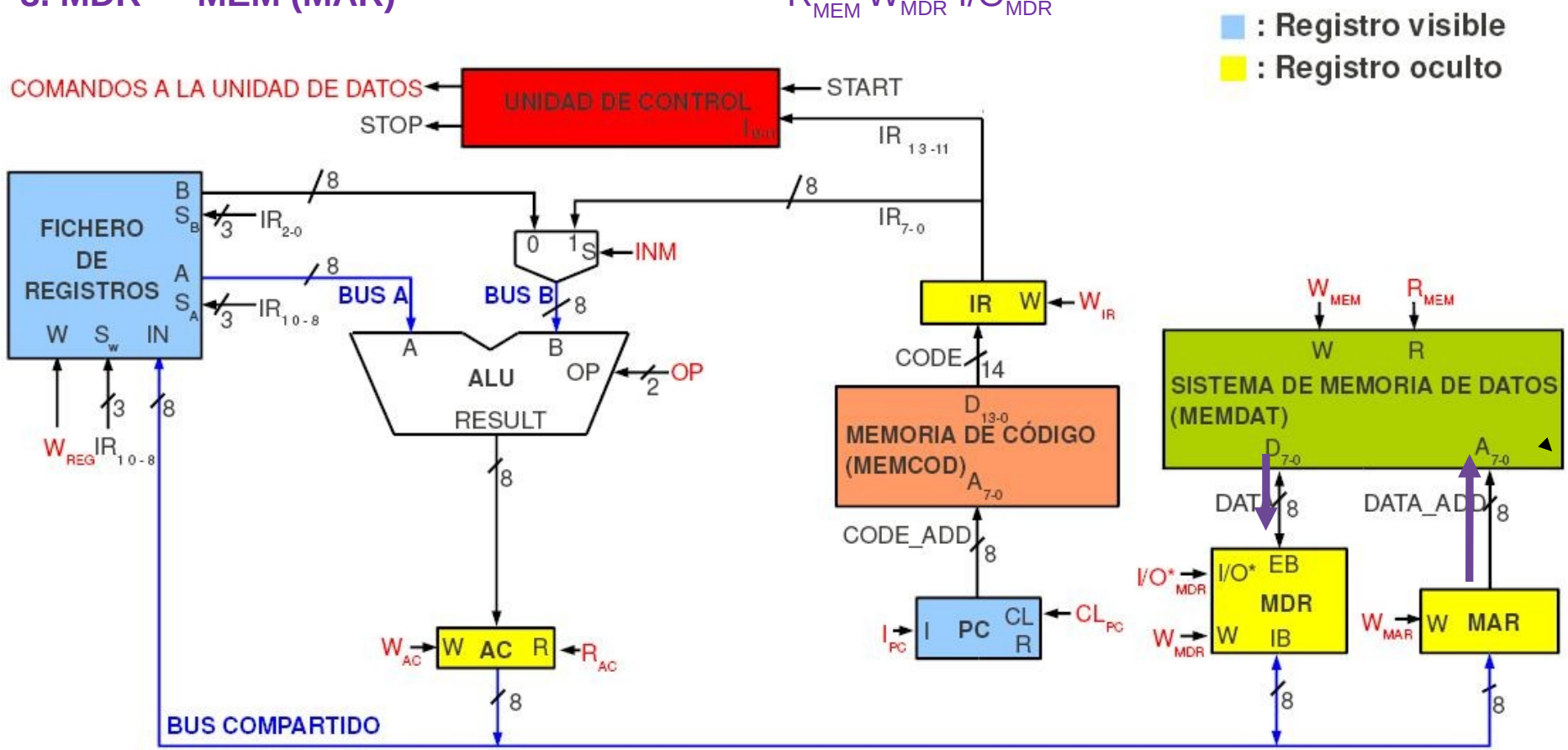
formato	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A instrucción con operando registro	código de operación			registro destino			Dirección del dato							

En IR debe aparecer la dirección del dato y el registro **destino**

Ejemplo de ejecución de instrucciones: LDS Rd,dir

1. AC ← IR_{7-0} — $OP_1 OP_0 W_{AC}$ INM
2. MAR ← AC — $R_{AC} W_{MAR}$
3. MDR ← MEM (MAR) — $R_{MEM} W_{MDR} I/O_{MDR}$

(Rd ← MEMDAT(dir))



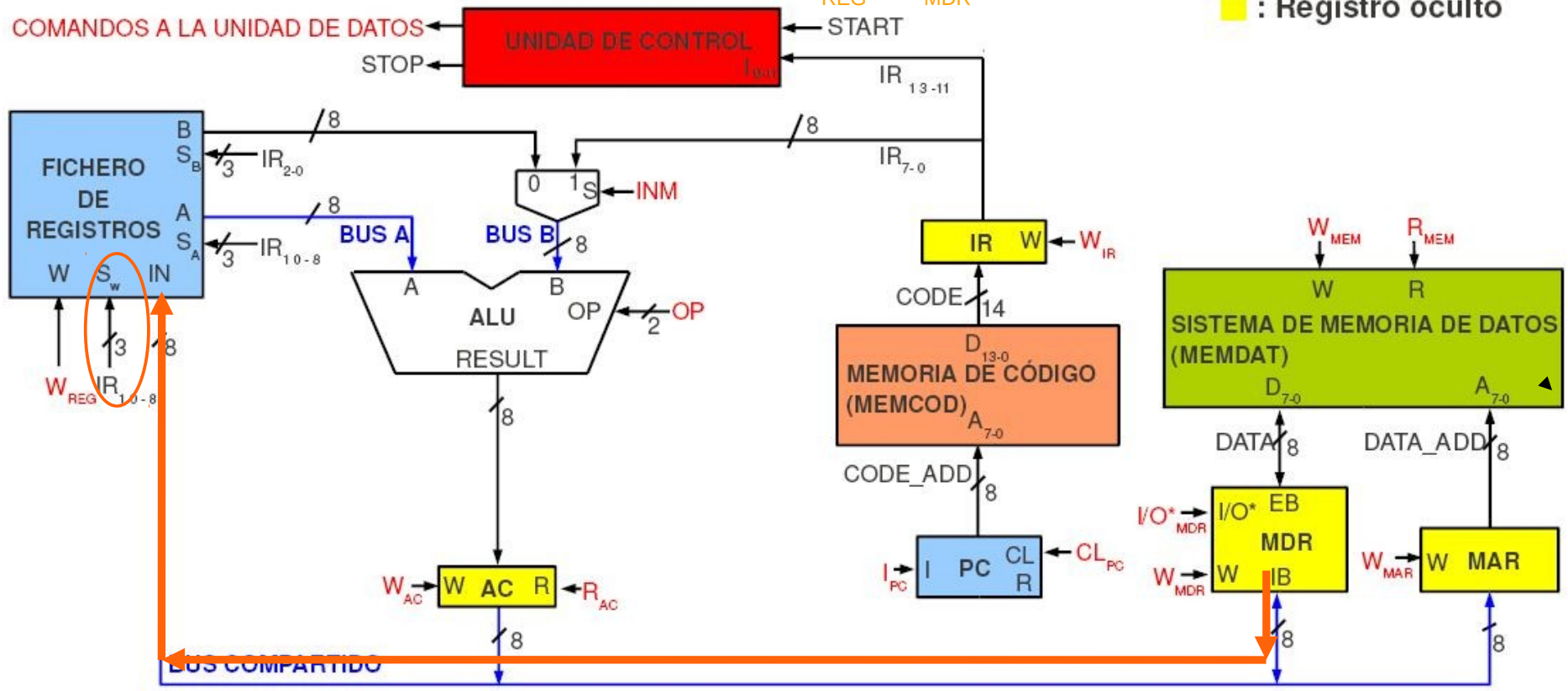
formato	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A instrucción con operando registro	código de operación			registro destino			Dirección del dato							

En IR debe aparecer la dirección del dato y el registro **destino**

Ejemplo de ejecución de instrucciones: LDS Rd,dir

1. AC ← IR_{7-0} — $OP_1 OP_0 W_{AC}$ INM
2. MAR ← AC — R_{AC} W_{MAR}
3. MDR ← MEM (MAR) — R_{MEM} W_{MDR} I/O_{MDR}
4. Rd ← MDR — W_{REG} I/O_{MDR}

(Rd ← MEMDAT(dir))

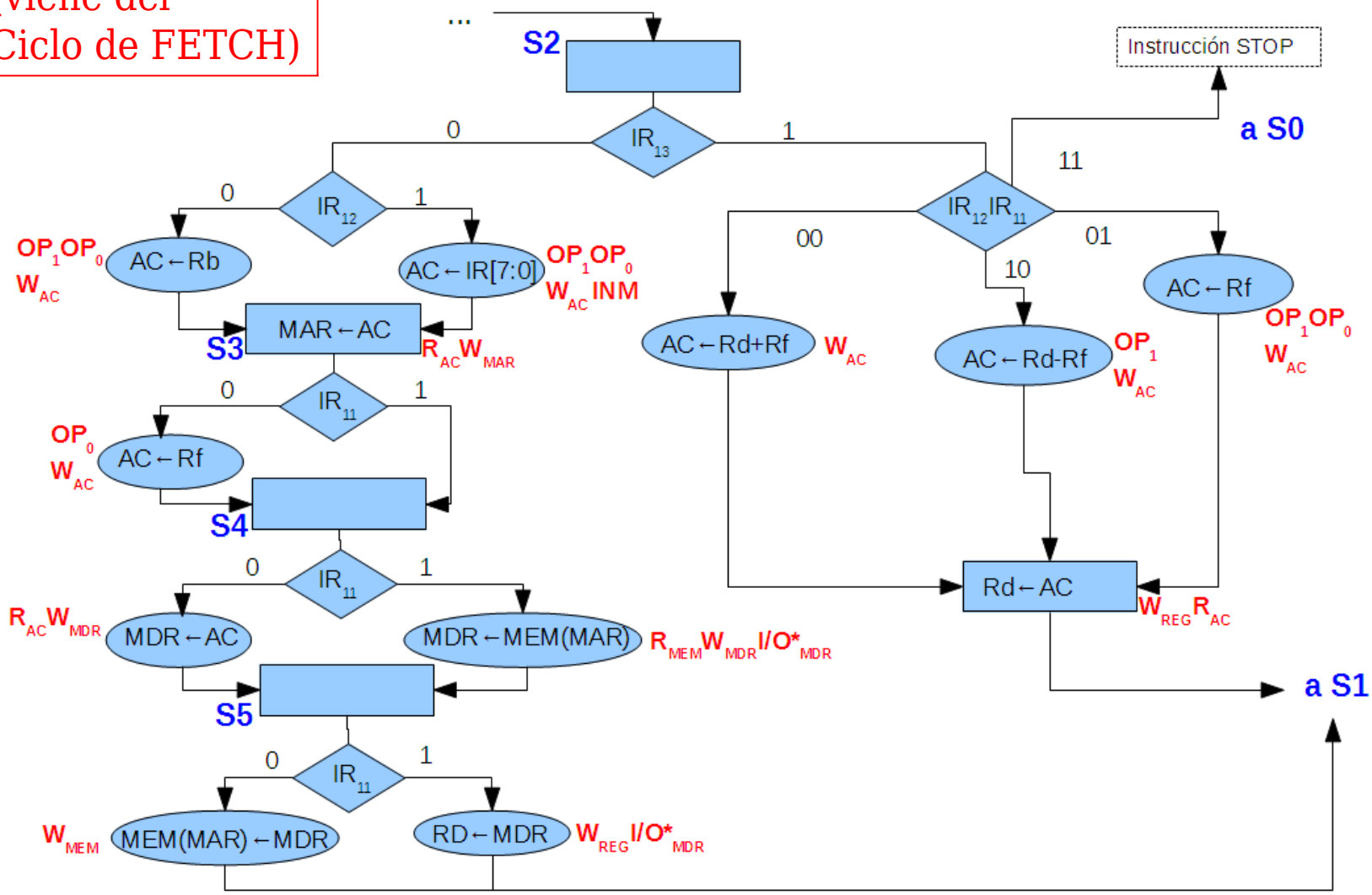


formato	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A instrucción con operando registro	código de operación			registro destino			Dirección del dato							

En IR debe aprecer la dirección del dato y el registro **destino**

Carta ASM del CS2 (ciclo EXECUTE)

(viene del Ciclo de FETCH)



Ejemplo de programación del Computador Simple CS2

Realizar un PROGRAMA que intercambie dos tablas de 4 datos que se encuentran almacenadas en la memoria y cuyas direcciones base están guardadas en los registros R0 y R1 respectivamente.
(El registro R2 tiene almacenado inicialmente el valor 1)

Programa

```
LD R3,(R0)
LD R4,(R1)
ST (R1),R3
ST (R0),R4
ADD R0,R2
ADD R1,R2
LD R3,(R0)
LD R4,(R1)
ST (R1),R3
ST (R0),R4
ADD R0,R2
ADD R1,R2
```

```
LD R3,(R0)
LD R4,(R1)
ST (R1),R3
ST (R0),R4
ADD R0,R2
ADD R1,R2
LD R3,(R0)
LD R4,(R1)
ST (R1),R3
ST (R0),R4
ADD R0,R2
ADD R1,R2
STOP
```

MEMORIA DE CÓDIGO

$Posición	contenido
$00	001 011-----000
$01	001 100-----001
$02	000 011-----001
$03	000 100-----000
$04	100 000-----010
$05	100 001-----010
…	…
$18	11 --- ---

MEMORIA DE DATOS

Pos	contenido
$00	.....
.....	
R0	DATO1TABLA1
R0+1	DATO2TABLA1
R0+2	DATO3TABLA1
R0+3	DATO4TABLA1
…	…
R1	DATO1TABLA2
R1+1	DATO2TABLA2
R1+2	DATO3TABLA2
R1+3	DATO4TABLA2
…	