Estructura de Computadores
(EdC-IC)

Tema 3
El computador simple

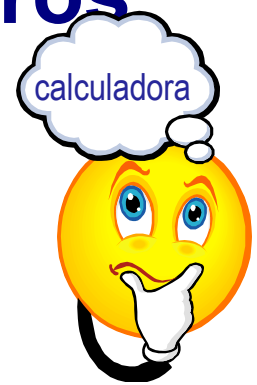
Autores originales: David Guerrero. Isabel Gómez

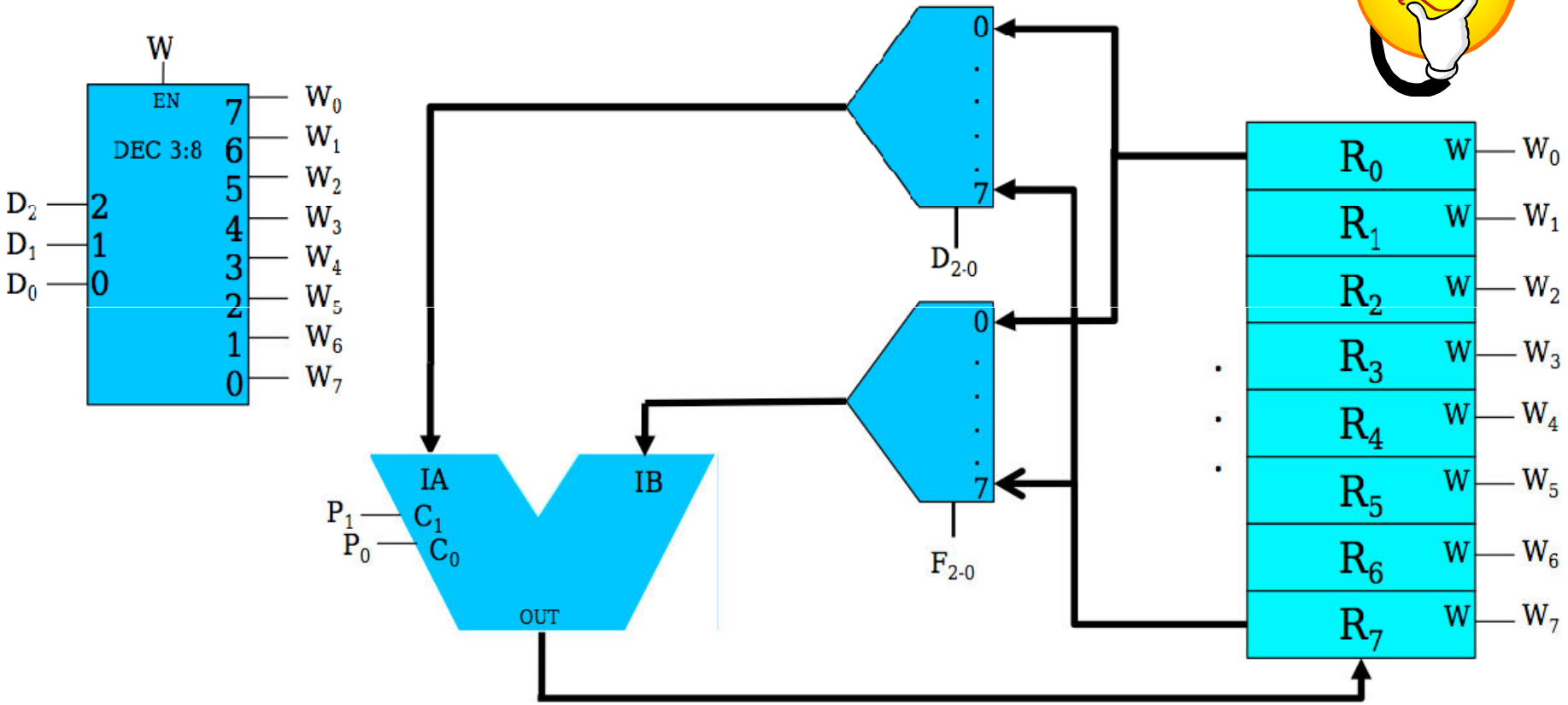
Personalización para ISW e IC: Francisco Pérez

Índice

La calculadora simple de 8 registros

(Arquitectura de la Unidad de Datos)

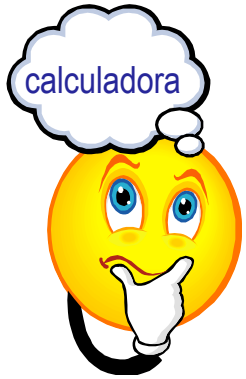




(Permite sumar, restar o transferir información entre dos registros)

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:

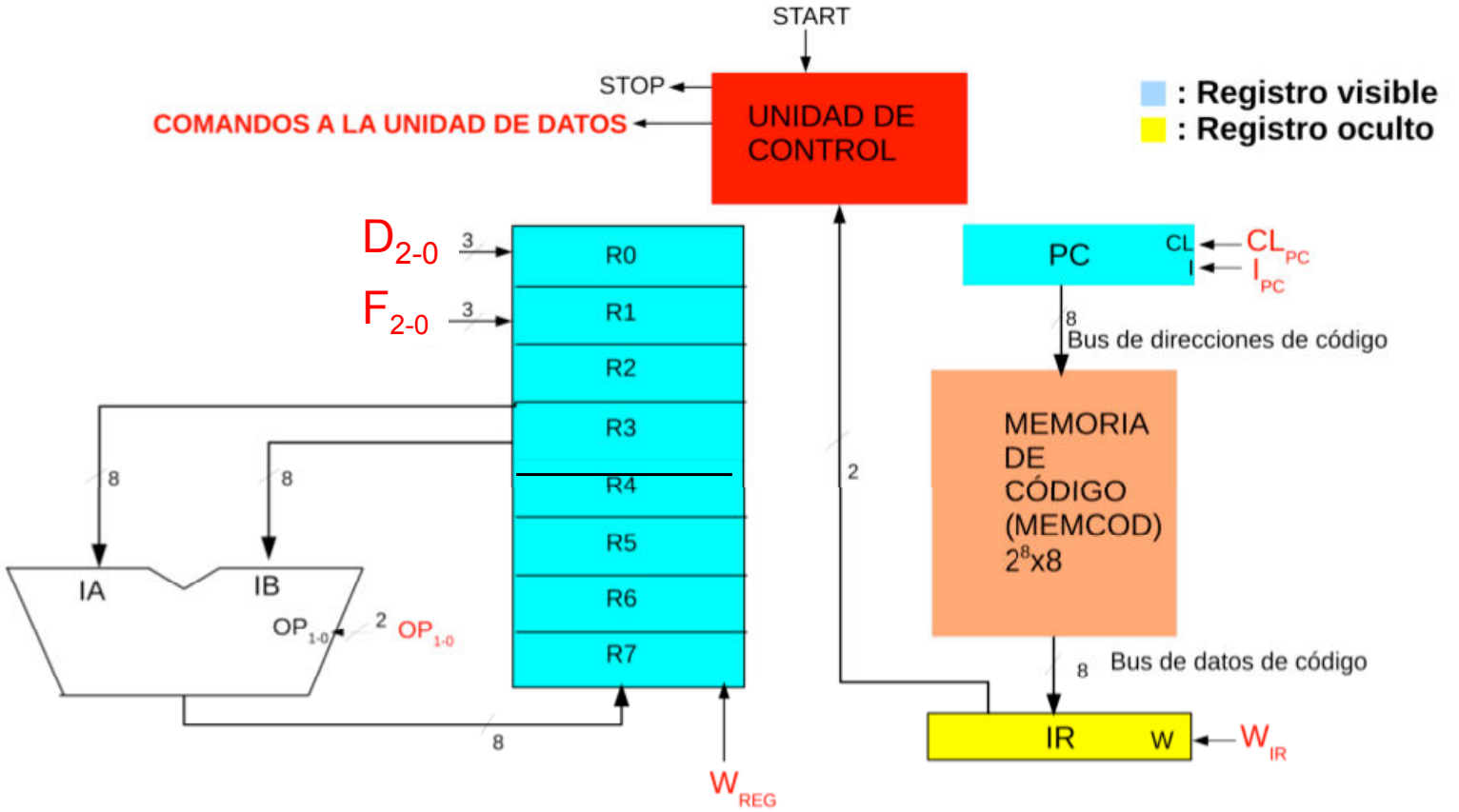
```
R0 ← R1
R0 ← R0+R1
R0 ← R0+R1
R0 ← R0-R2
```

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)
  - $D_{2\text{-}0}$: Selecciona registro de entrada en IA (de la ALU)
  - $F_{2\text{-}0}$: Selecciona registro de entrada en IB (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

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

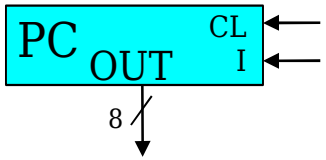
En CS1 los datos (operandos de las instrucciones) siempre están almacenados en el banco de registros.

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

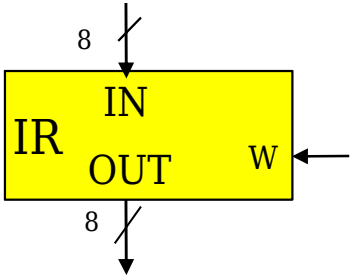
IR7 IR6	SINTAXIS	FUNCIÓN
0 0	ADD Rd,Rf	Rd←Rd+Rf
1 0	SUB Rd,Rf	Rd←Rd-Rf
0 1	MOV Rd,Rf	Rd←Rf
1 1	STOP	NOP

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

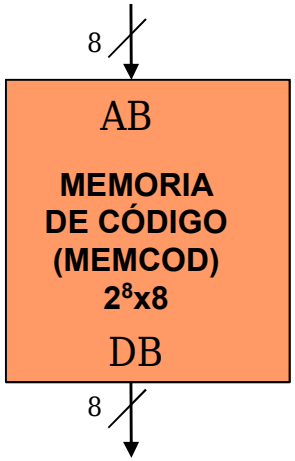
Descripción RT de los nuevos componentes del CS1



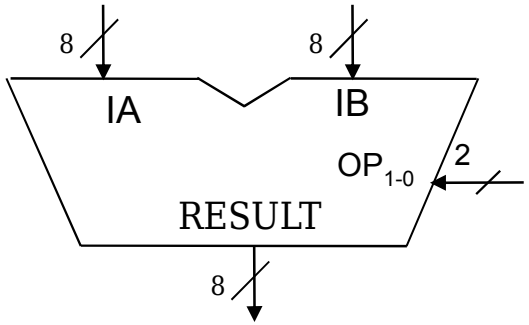
CL I	PC←	OUT:=
00	PC	PC
01	PC+1	PC
10	0	PC
11	0	PC



W	IR←	OUT:=
0	IR	IR
1	IN	IR



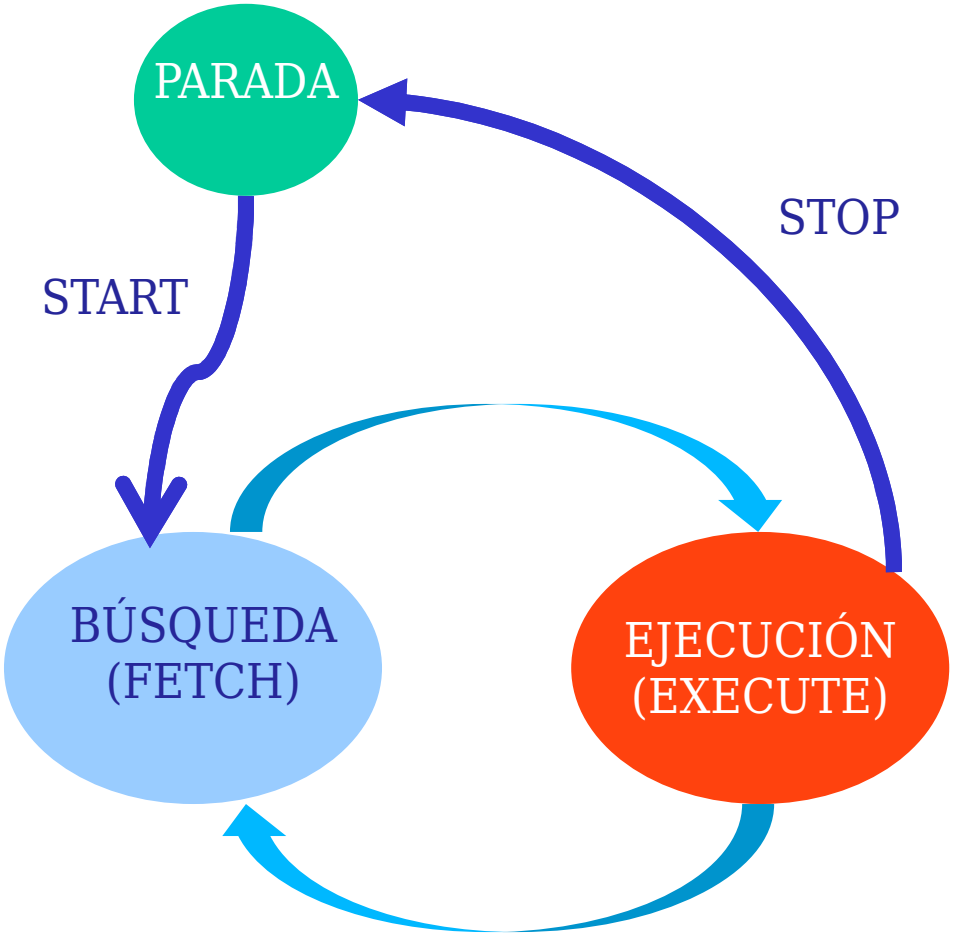
DB=MEMCOD[AB]



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

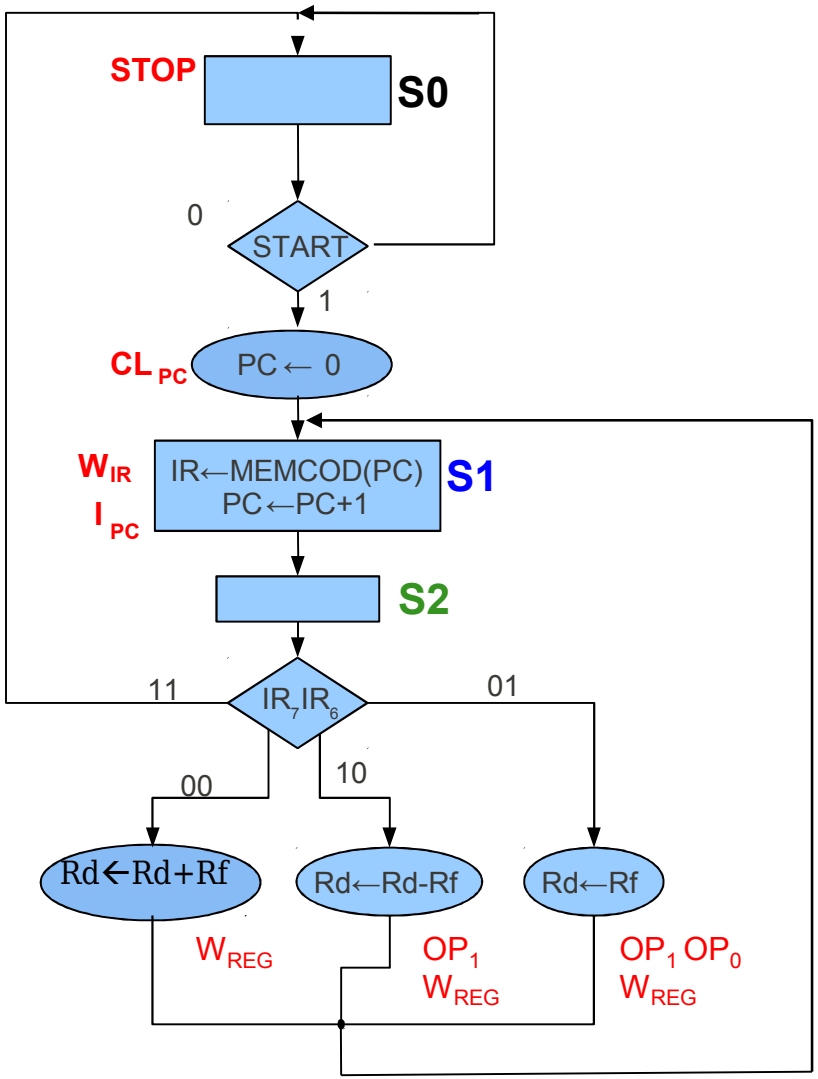
Diseño de la Unidad de Control del Computador Simple CS1

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

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

Ejemplo de programación del CS1

Ejemplo:

Escribir un programa que realice la siguiente operación:

R6←3R4-2R1

Programa con mnemónicos

```
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

1. Limitaciones de la calculadora simple
2. El Computador Simple 1 (CS1)

   (concepto de Programa almacenado en memoria)
3. El Computador Simple 2 (CS2)

   (memoria de datos y memoria de programa)
4. El Computador Simple CS2010

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*”)

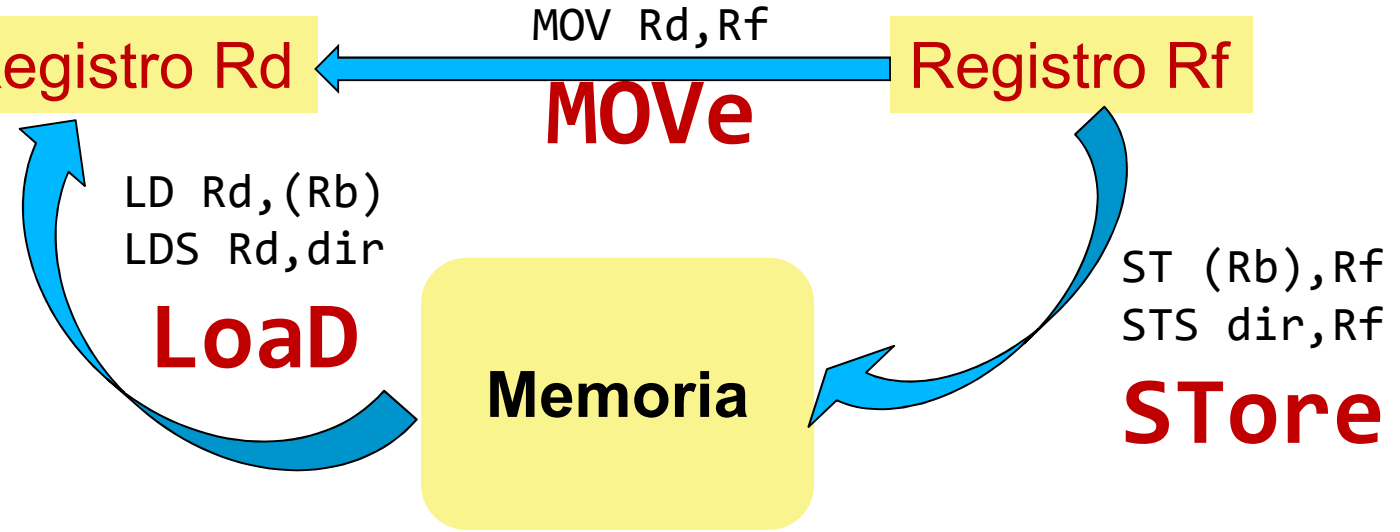
```
ST (Rb),Rf   ;STore
LD Rd,(Rb)   ;LoaD
```

- 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
```

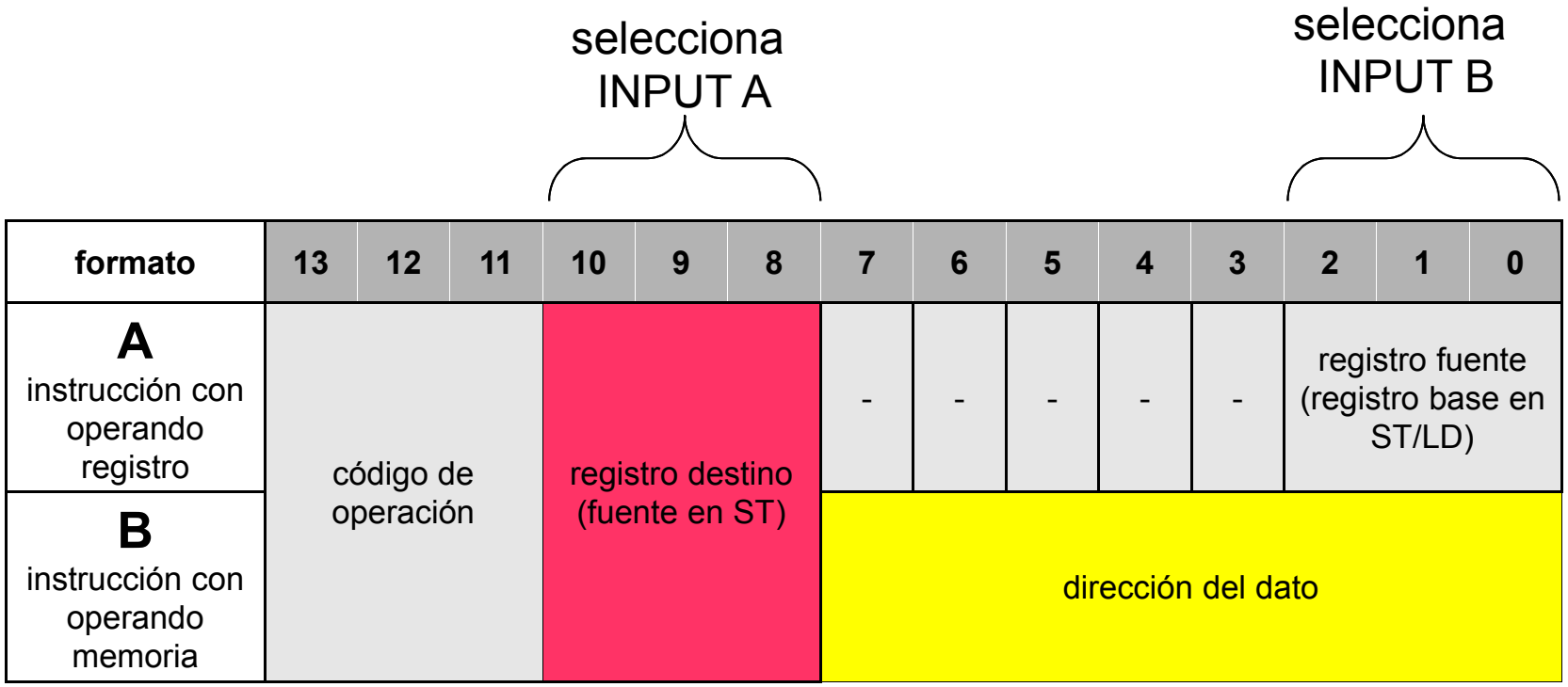
En CS2 no existe direccionamiento inmediato (el dato no puede ser suministrado en la propia instrucción)

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	----------

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

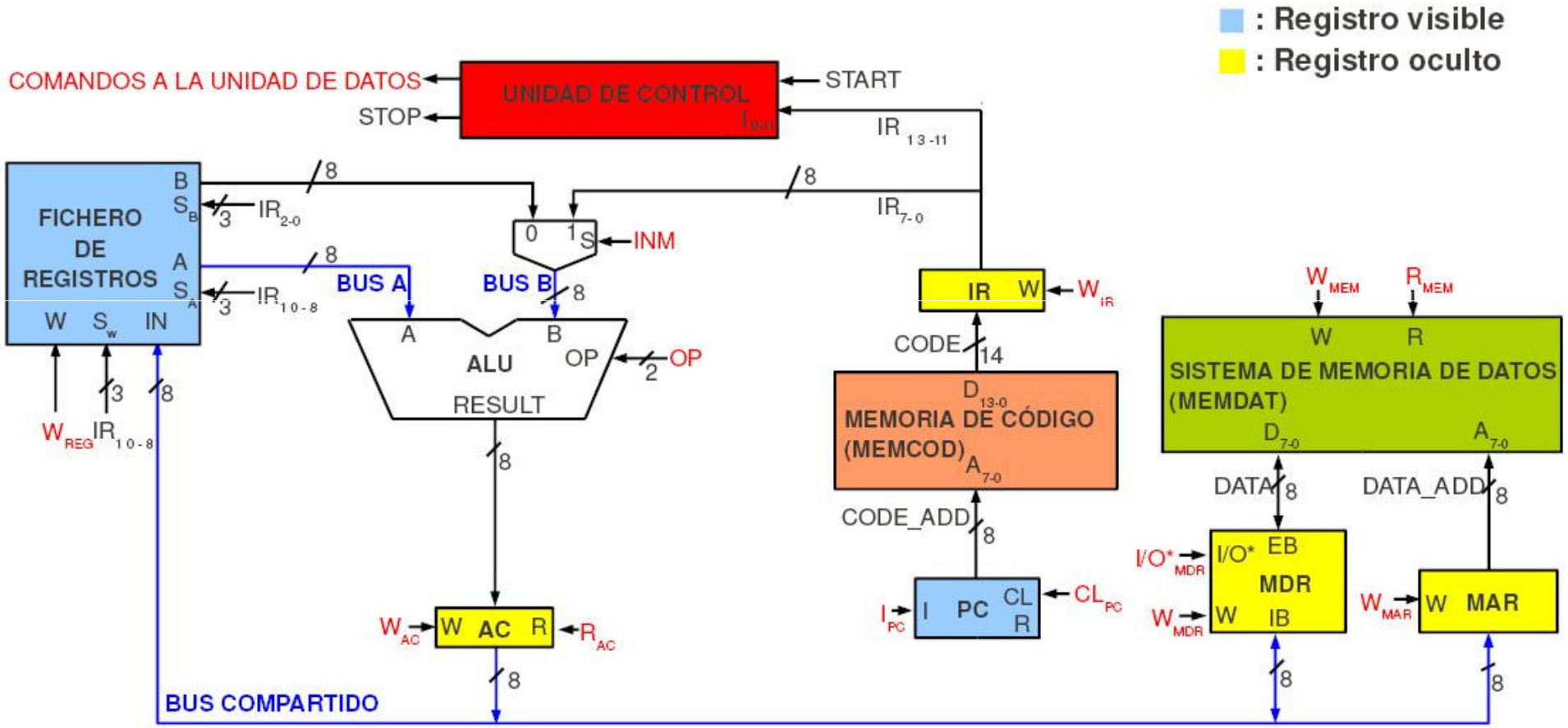
Formato de instrucciones del CS2

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							

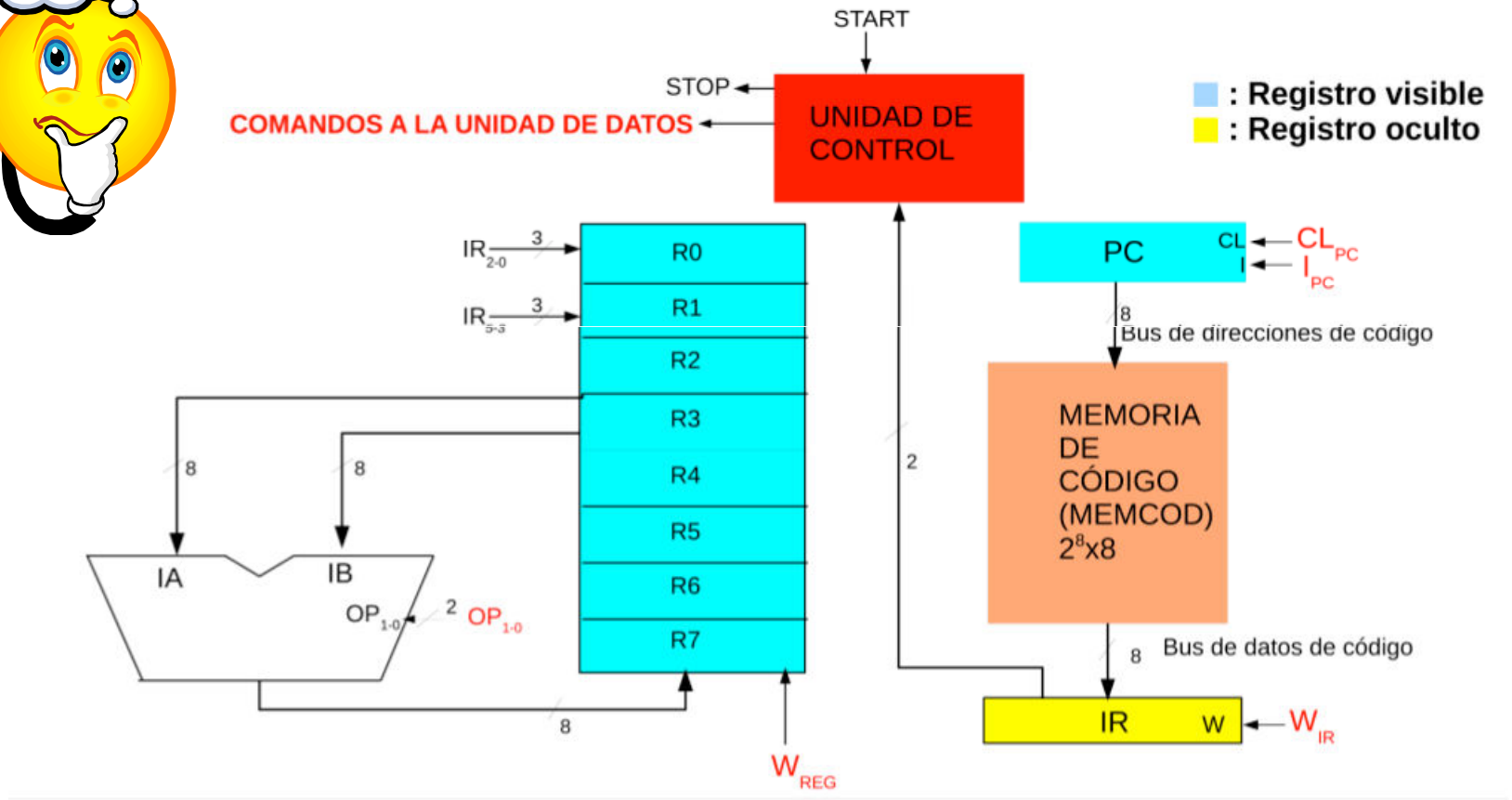
CO	SINTAXIS
000	ST (Rb),Rf
001	LD Rd, (Rb)
010	STS dir, Rf
011	LDS Rd,dir
100	ADD Rd,Rf
110	SUB Rd,Rf
101	MOV Rd,Rf
111	STOP

13	12	11	10	9	8	7	6	5	4	3	2	1	0
000			R f								Rb		
001			Rd								Rb		
010			R f										
011			Rd										
100			Rd										
110			Rd										
101			Rd								Rf		
111											Rf		
							d i r						
							d i r						
											Rf		

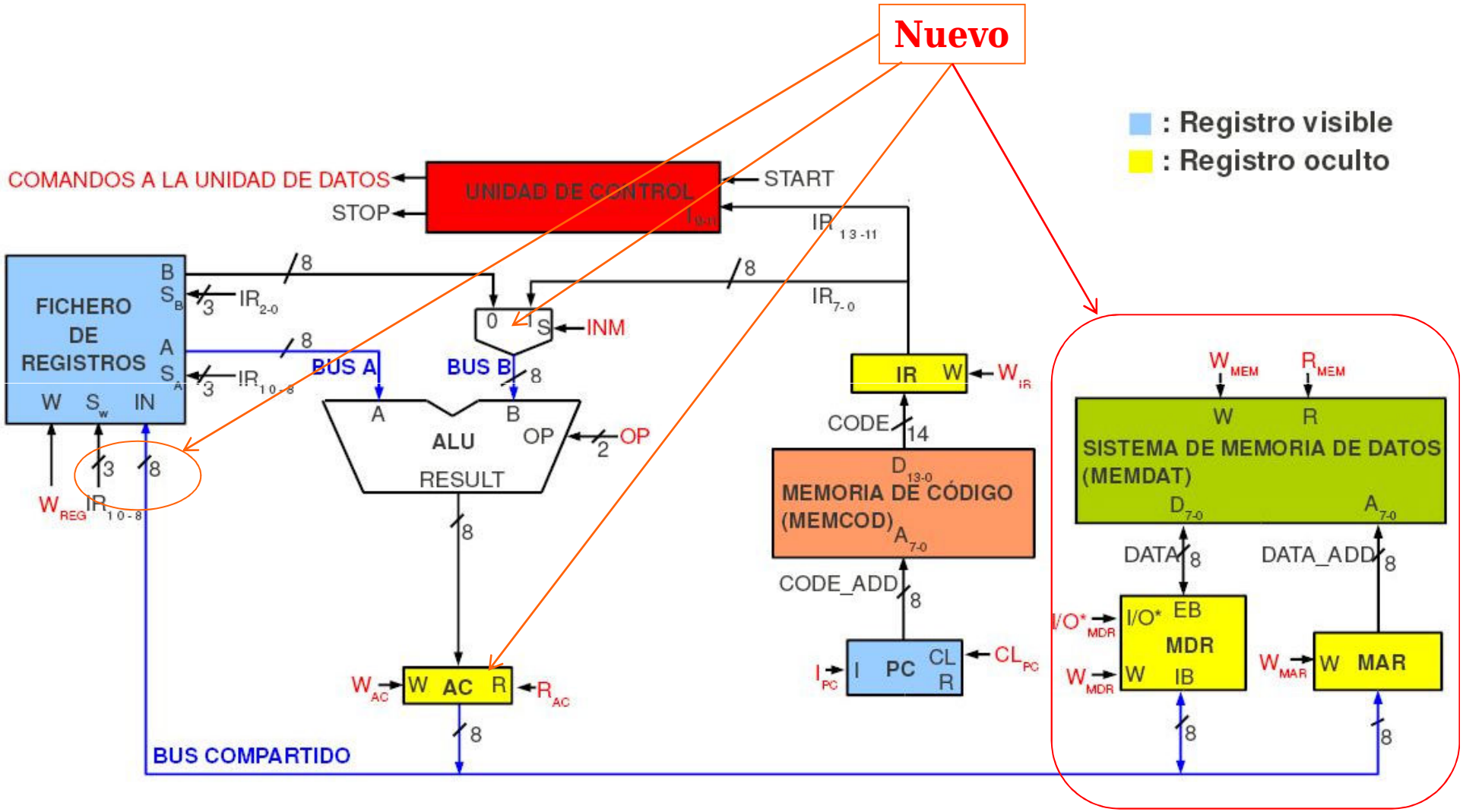
Arquitectura del CS2



Arquitectura del computador CS1



Arquitectura del CS2

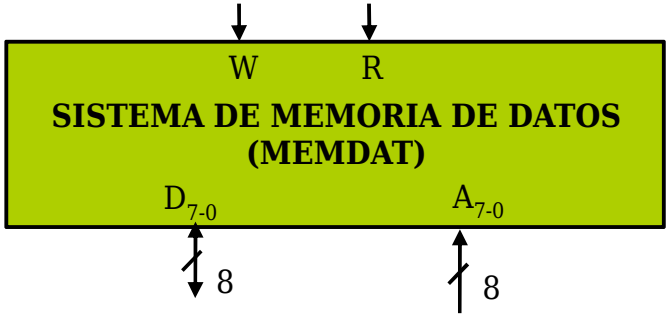


Arquitectura del CS2

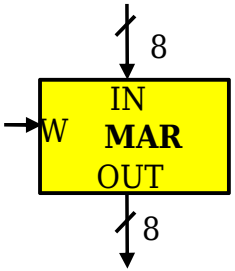
Modificaciones en la arquitectura del CS2 (todo lo que se ha añadido es para dotar al sistema de almacenamiento de datos en memoria).

- 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 esta debe compartir el bus.
- Multiplexor a la entrada de la ALU porque esta 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.

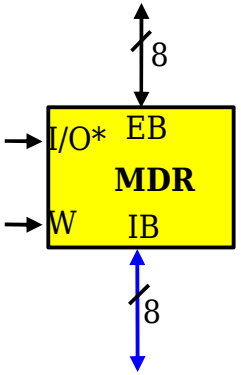
Descripción RT de los nuevos componentes del CS2



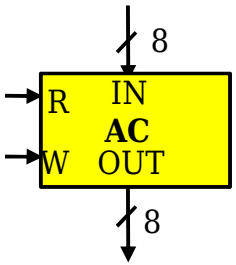
W R	MEMDAT[A_{7-0}]←	D_{7-0}:=
00	MEMDAT[A_{7-0}]	H.I.
01	MEMDAT[A_{7-0}]	MEMDAT[A_{7-0}]
10	D_{7-0}	DATA
11	PROHIBIDO	----



W	MAR ←	OUT:=
0	MAR	MAR
1	IN	MAR



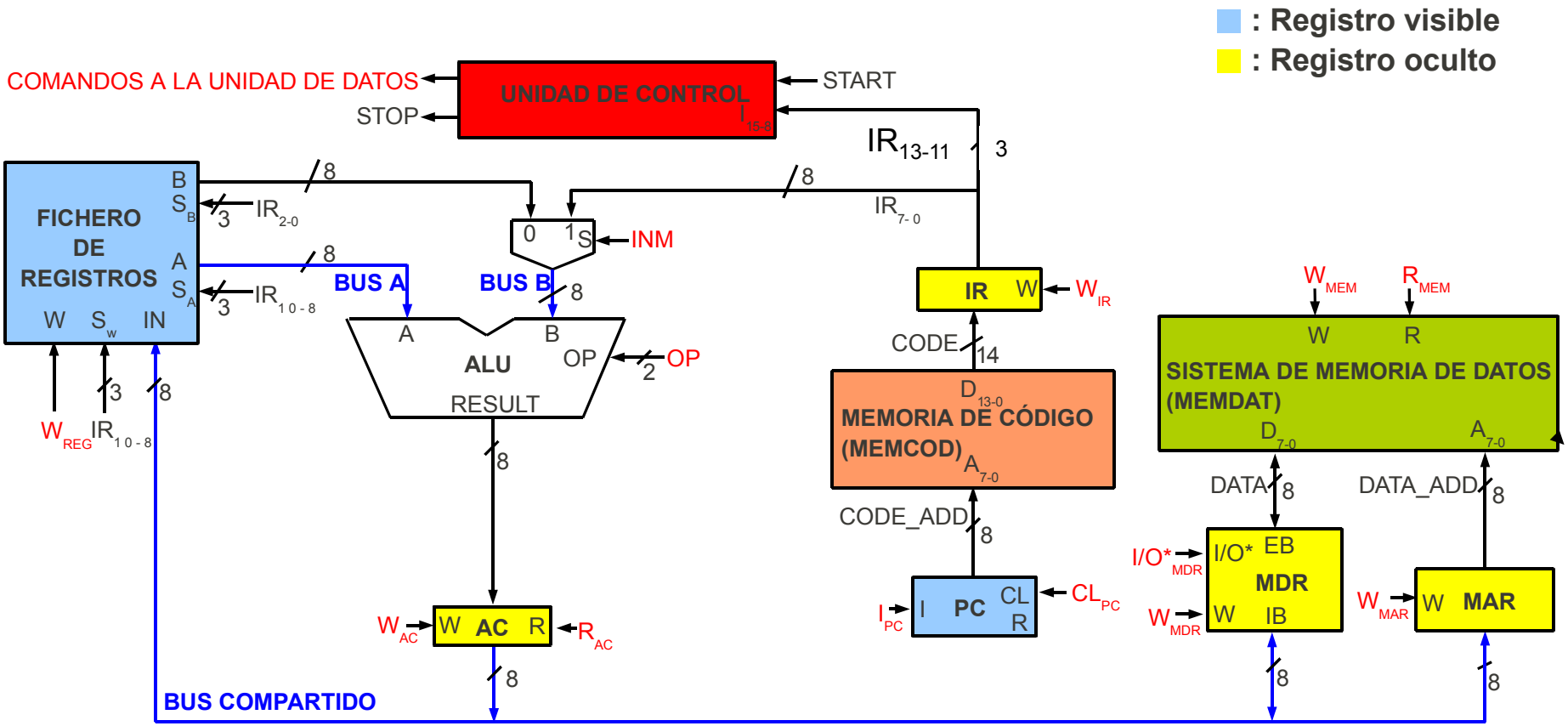
W I/O*	MDR←	IB:=	EB:=
00	MDR	H.I.	MDR
01	MDR	MDR	H.I.
10	IB	IB	H.I.
11	EB	H.I.	EB



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

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

(MEMDAT(Rb) ← Rf)

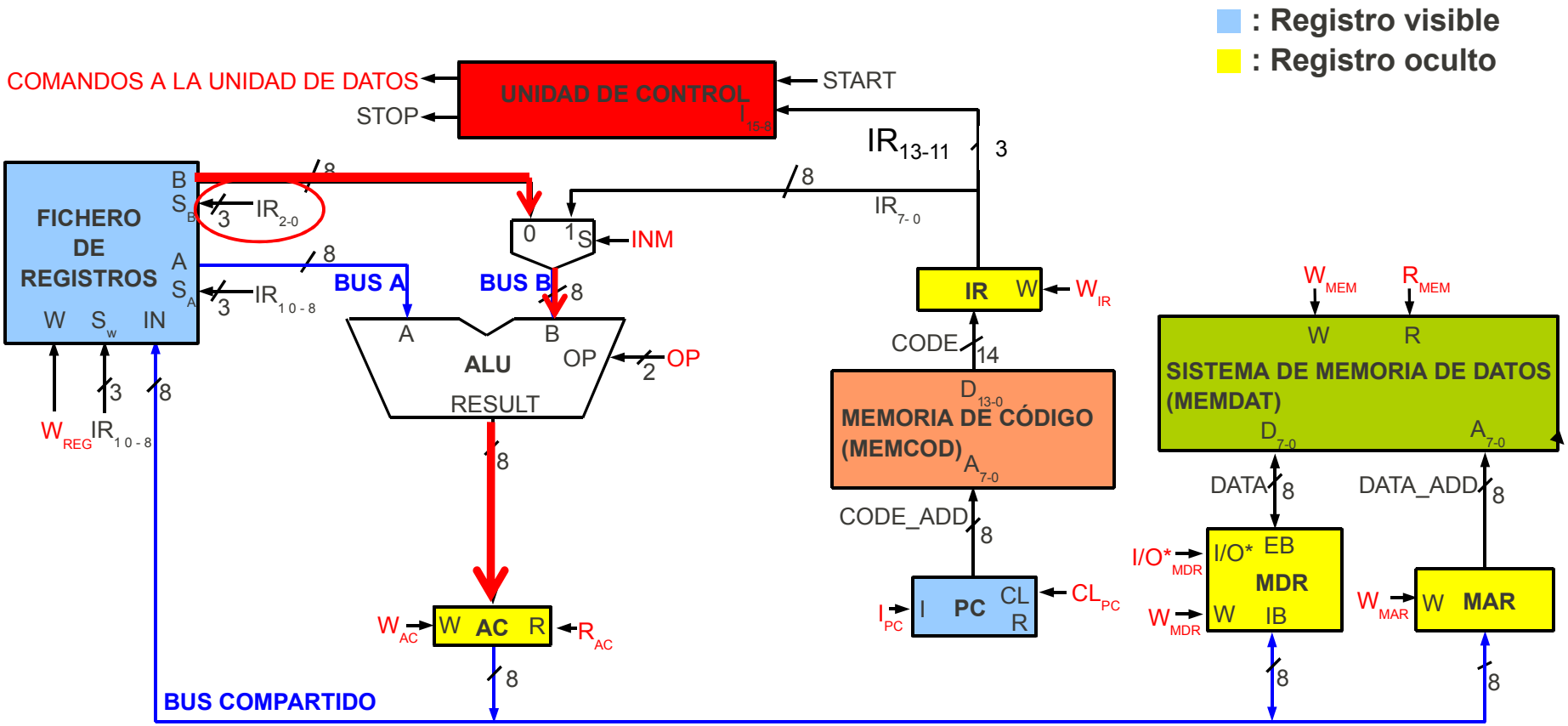


formato	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A instrucción con operando registro	código de operación			registro fuente en ST			-	-	-	-	-	registro base en ST		

Los valores almacenados en IR debe corresponder a los registros **base** y **destino**

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		

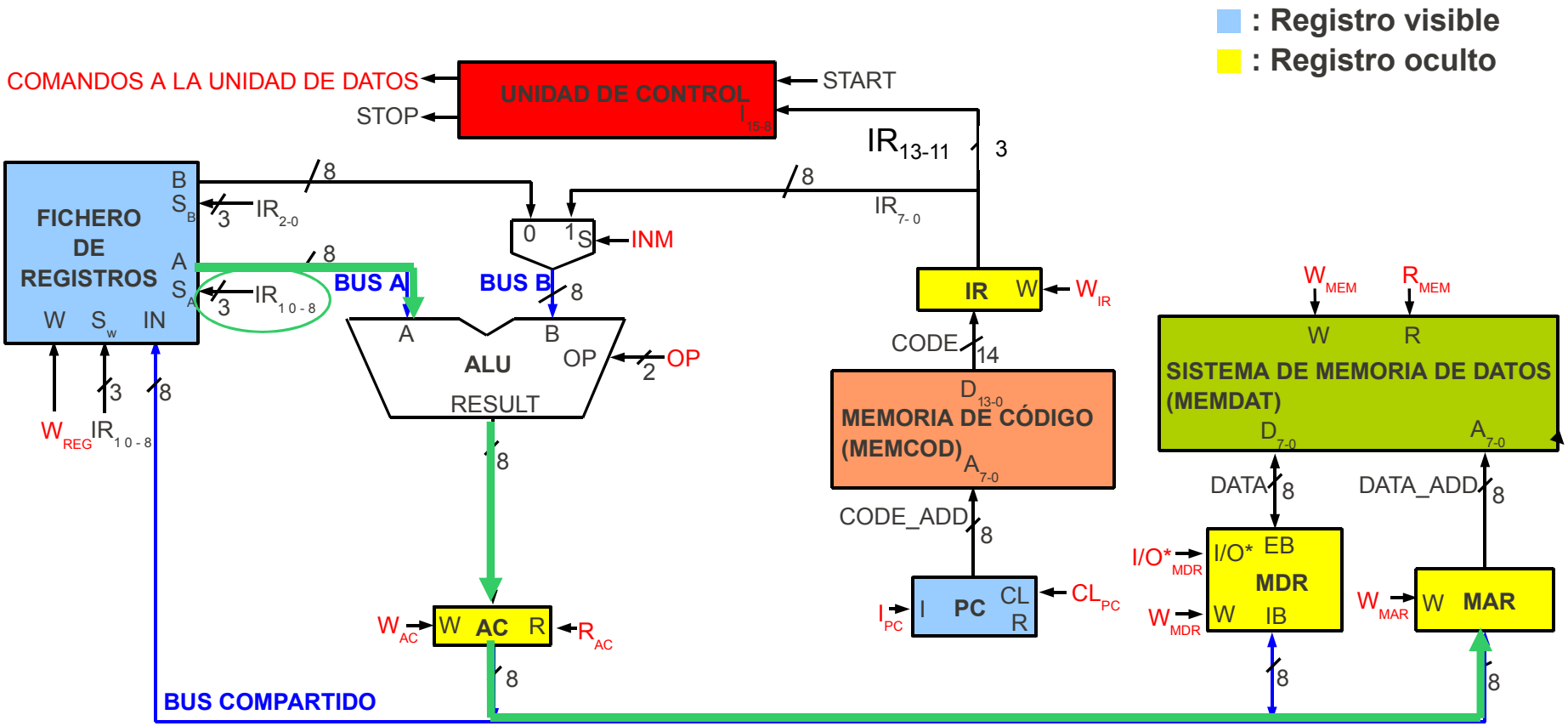
Los valores almacenados en IR debe corresponder a los registros **base** y **destino**

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		

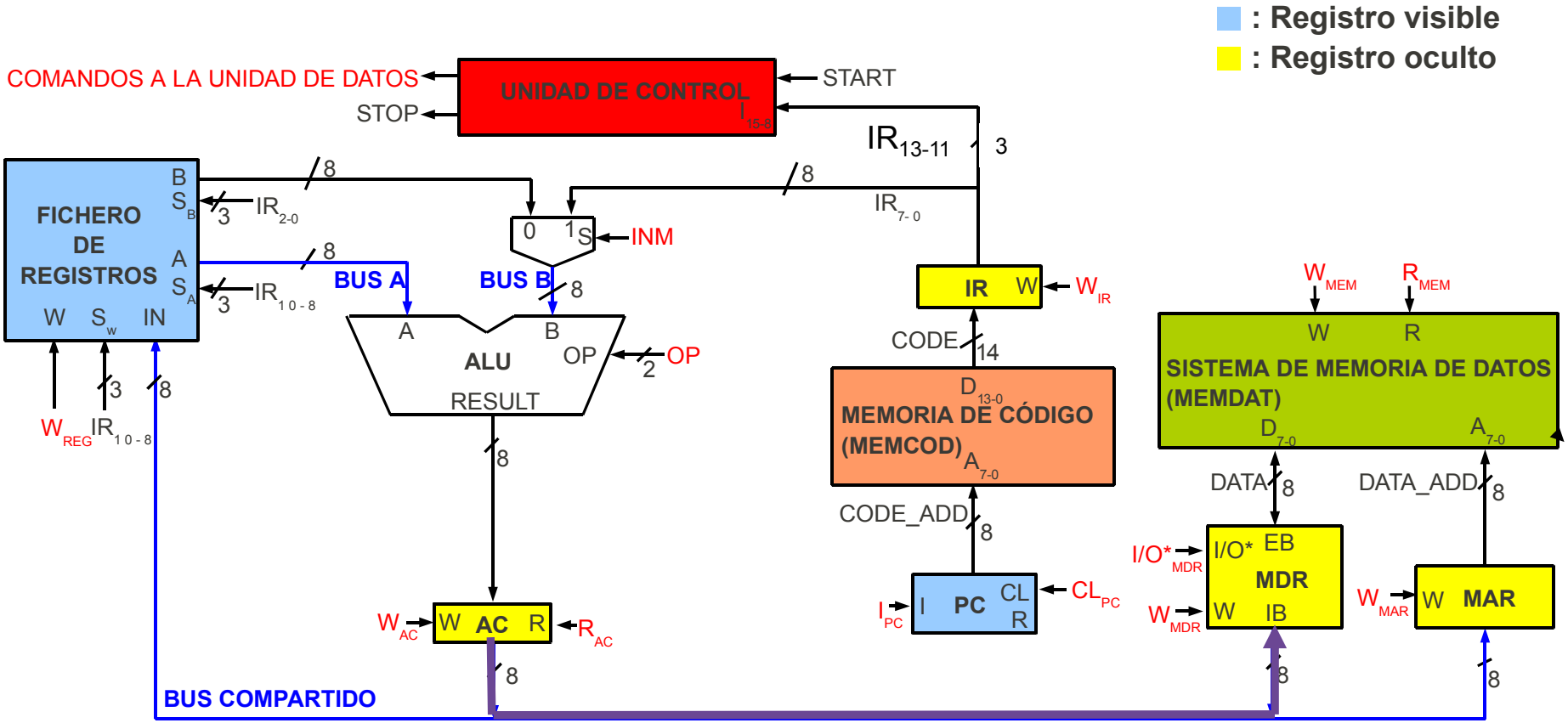
Los valores almacenados en IR debe corresponder a los registros **base** y **destino**

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

1. AC←REG[IR_{2-0}]
2. MAR←AC; AC←REG[IR_{10-8}]
3. MDR←AC

$OP_1 OP_0 W_{AC}$
$R_{AC}\ W_{MAR}\ OP_0\ W_{AC}$
$R_{AC}\ W_{MDR}$

(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 debe corresponder a los registros **base** y **destino**

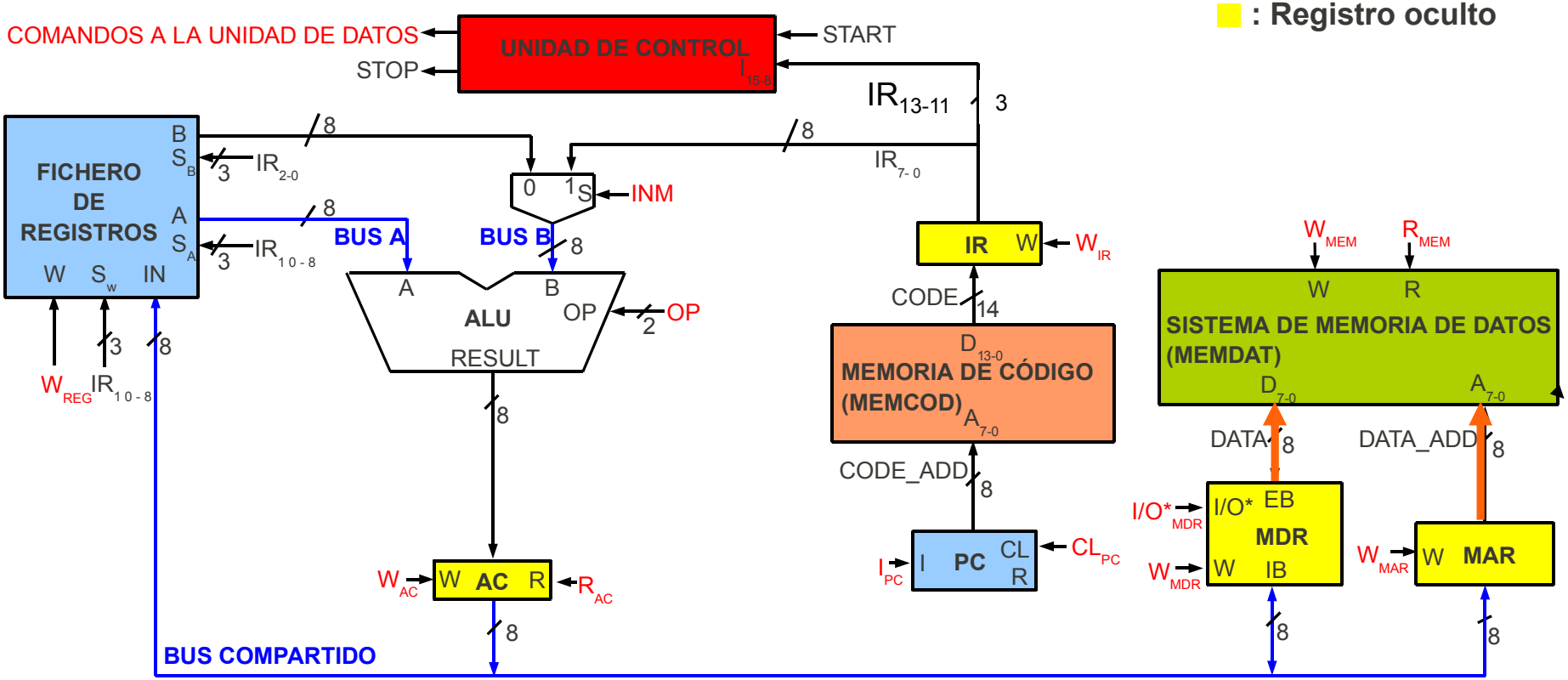
Ejemplo de ejecución de instrucciones: ST(Rb),Rf

1. AC←REG[IR_{2-0}]
2. MAR←AC; AC←REG[IR_{10-8}]
3. MDR←AC
4. MEM[MAR]←MDR

$OP_1 OP_0 W_{AC}$
$R_{AC} W_{MAR} OP_0 W_{AC}$
$R_{AC} W_{MDR}$
W_{MEM}

(MEMDAT(Rb) ← Rf)

: Registro visible
: Registro oculto



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 debe corresponder a los registros **base** y **destino**

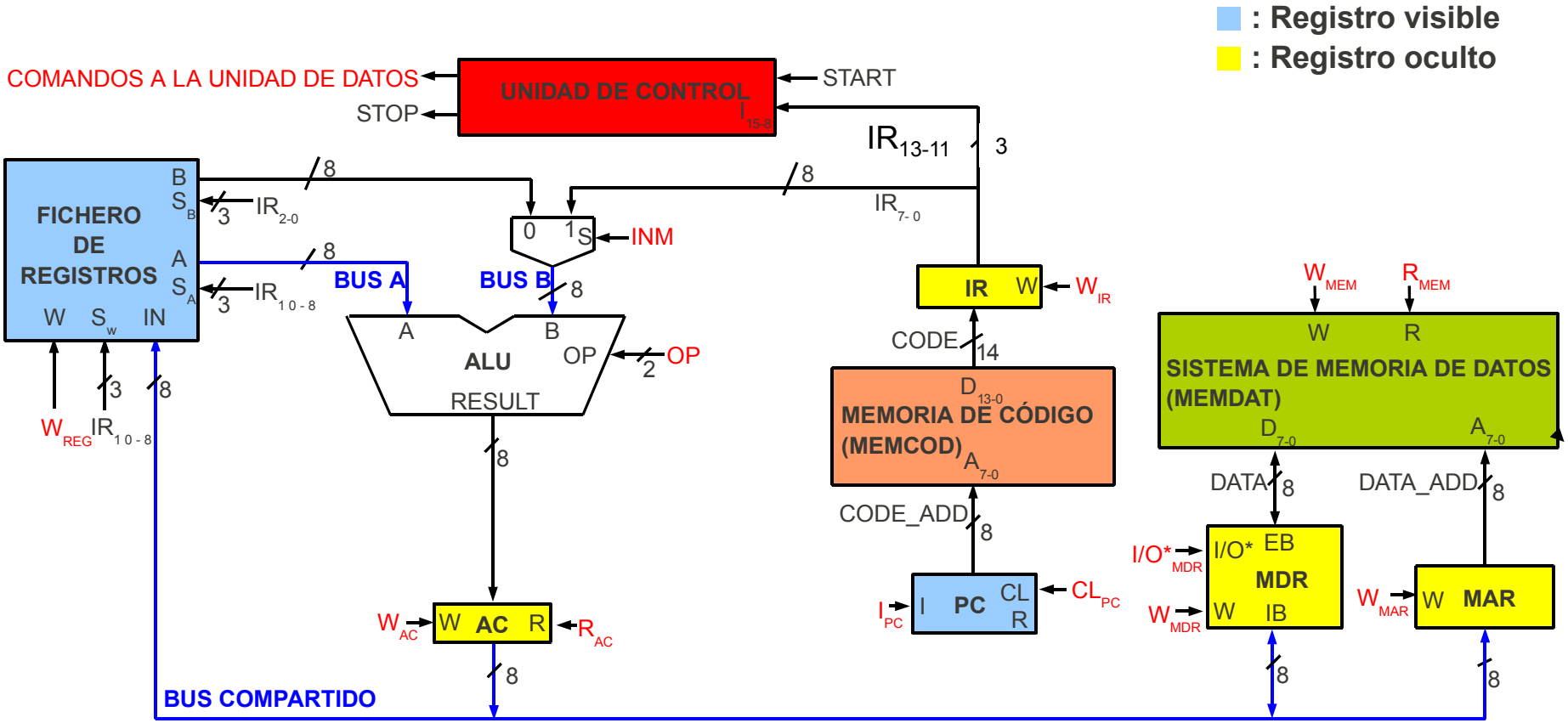
Descripción de las nuevas instrucciones del CS2

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** $OP_1OP_0W_{AC}$	**AC←Rb** OP_1OP_0 W_{AC}	**AC←dir** $OP_1OP_0W_{AC}$INM	**AC←dir** $OP_1OP_0W_{AC}$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

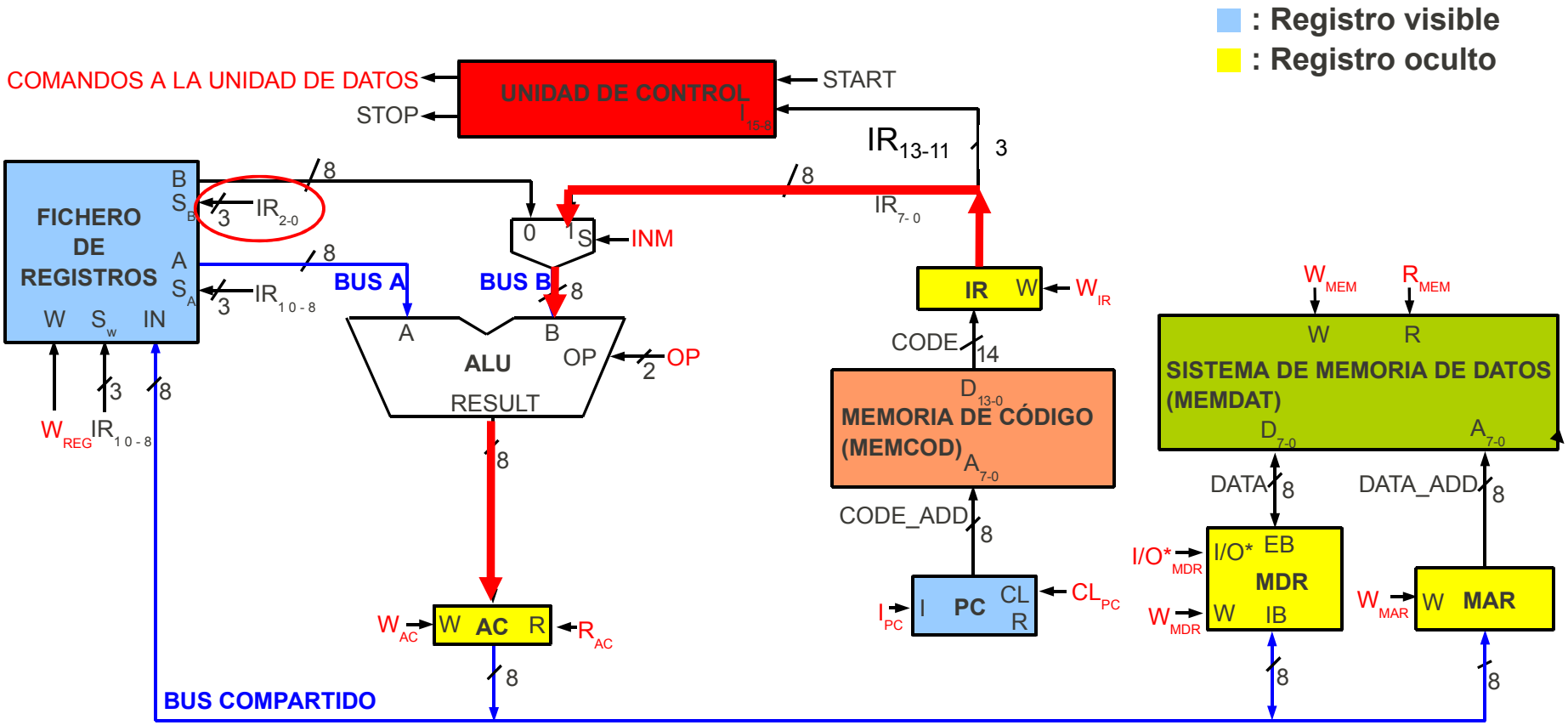
(Rd ← MEMDAT(dir))



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							

Ejemplo de ejecución de instrucciones: LDS Rd,dir

1. AC←REG[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							

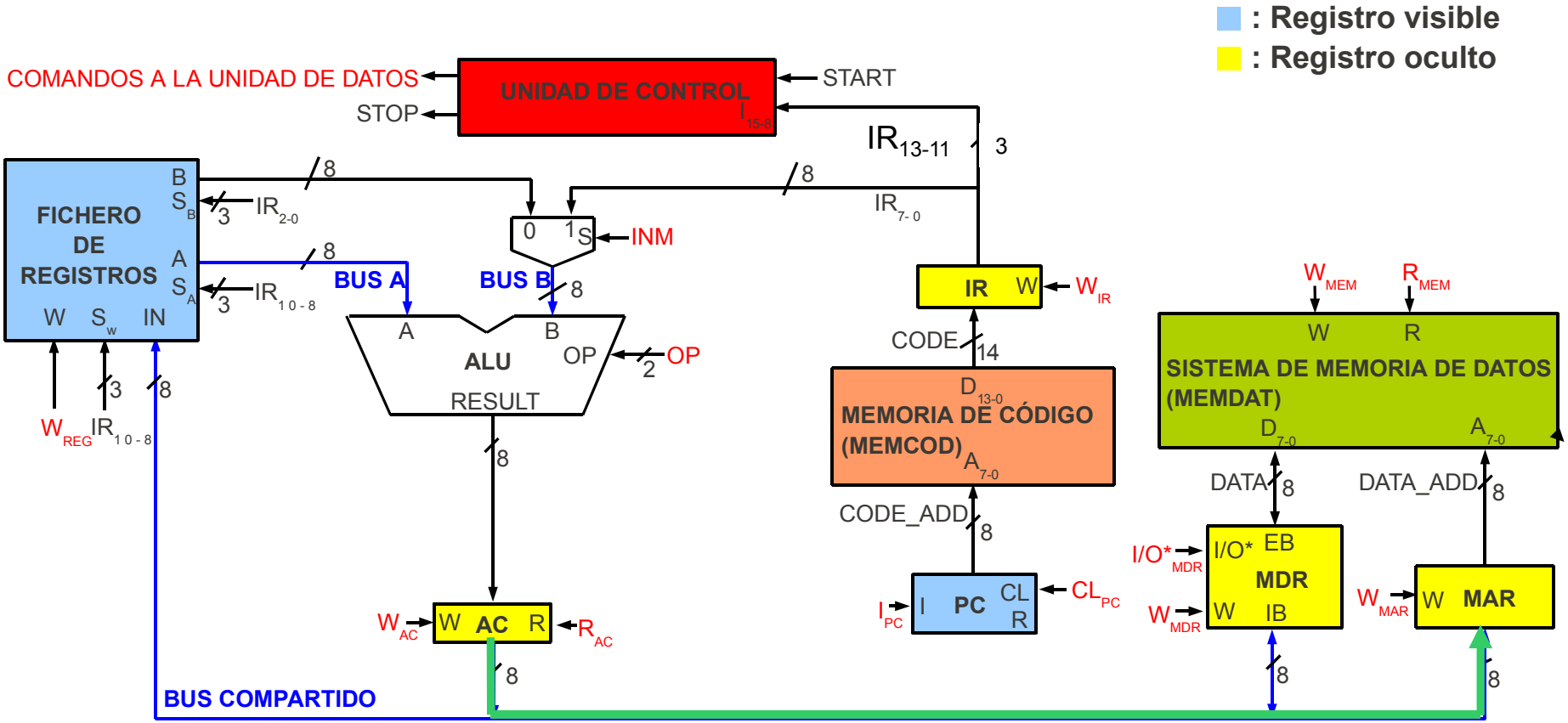
En IR debe aprecer la dirección del dato y el registro **destino**

Ejemplo de ejecución de instrucciones: LDS Rd,dir

1. AC←REG[IR_{7-0}]
2. MAR←AC

$OP_1OP_0W_{AC}$ INM
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 aprecer la dirección del dato y el registro **destino**

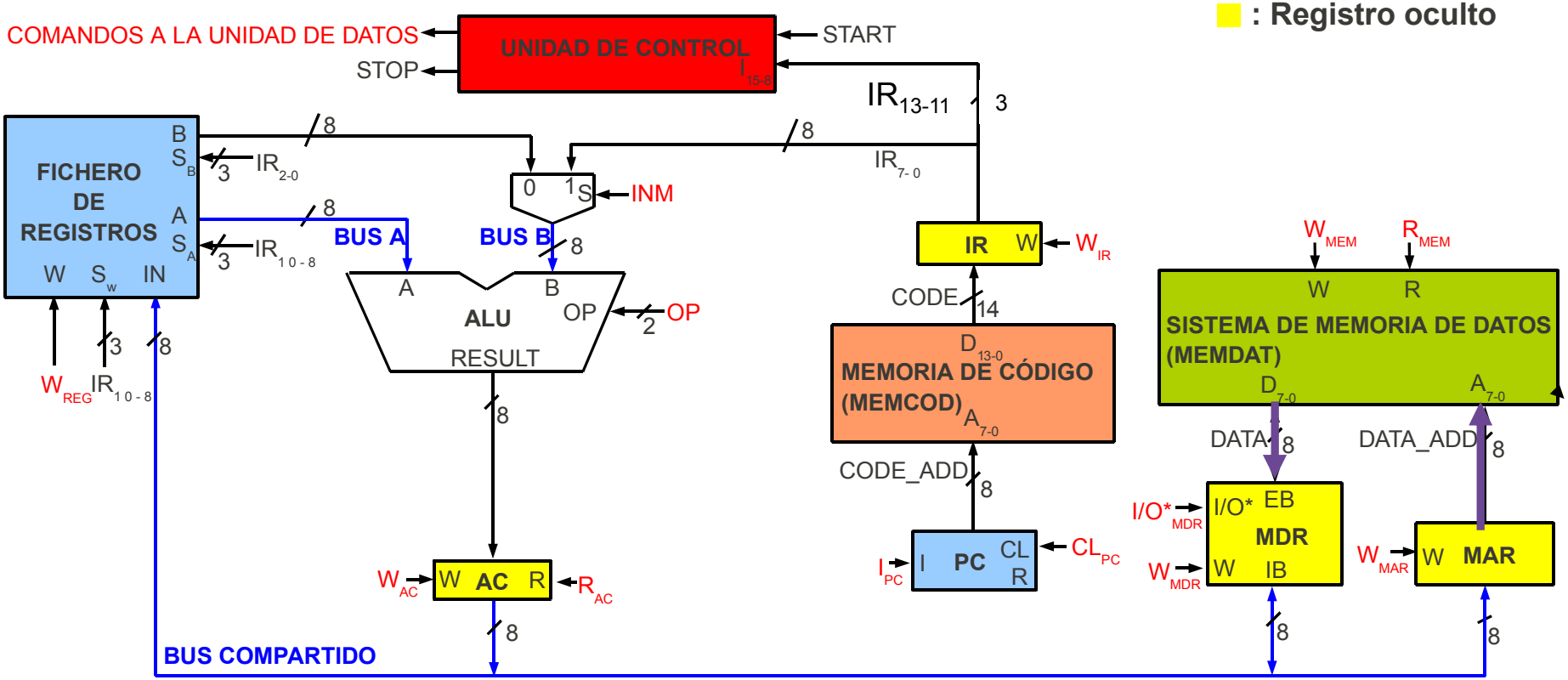
Ejemplo de ejecución de instrucciones: LDS Rd,dir

1. AC←REG[IR_{7-0}]
2. MAR←AC
3. MDR ← MEM (MAR)

$OP_1 OP_0 W_{AC}$ INM
R_{AC} W_{MAR}
R_{MEM} W_{MDR} I/O_{MDR}

(Rd ← MEMDAT(dir))

: Registro visible
: Registro oculto



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**

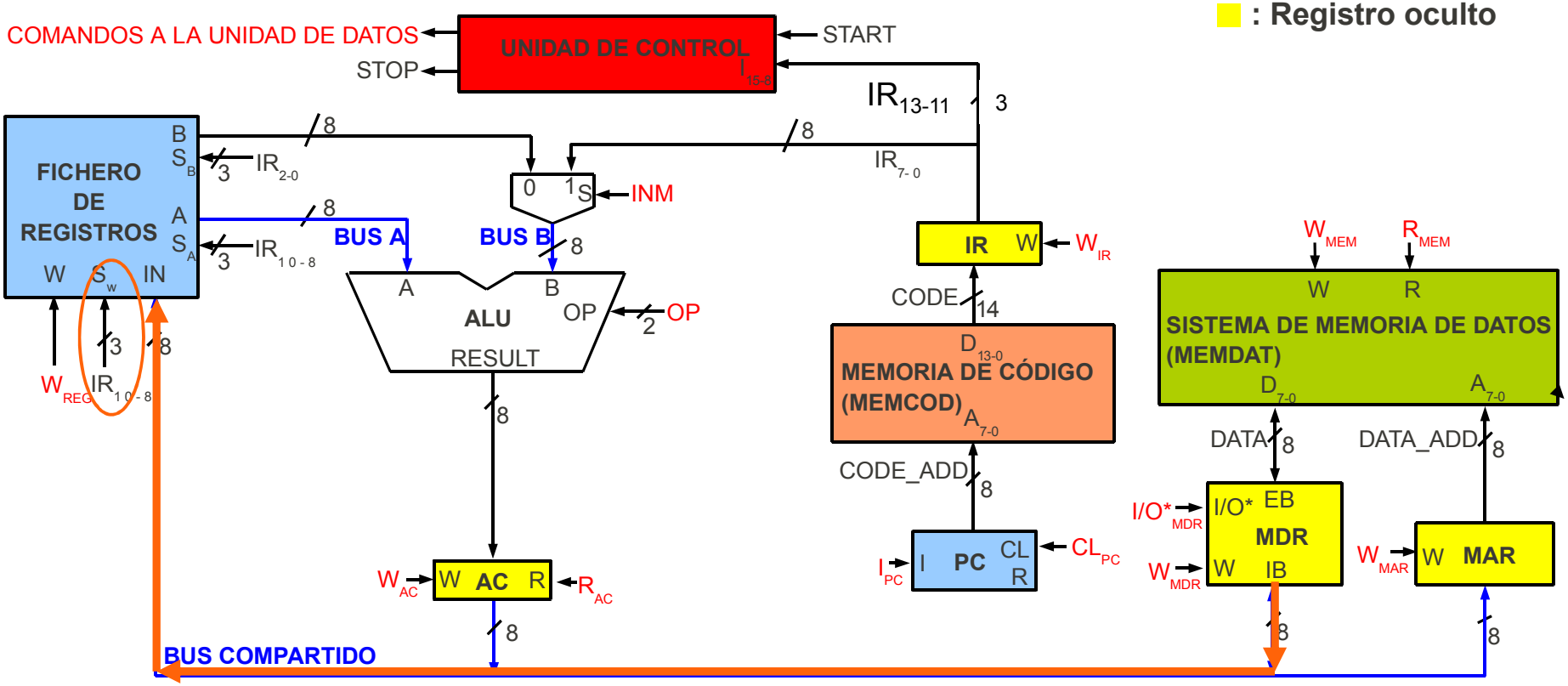
Ejemplo de ejecución de instrucciones: LDS Rd,dir

1. AC←REG[IR_{7-0}]
2. MAR←AC
3. MDR ← MEM (MAR)
4. Rd ← MDR

$OP_1 OP_0 W_{AC}$ INM
R_{AC} W_{MAR}
R_{MEM} W_{MDR} I/O_{MDR}
W_{REG} I/O_{MDR}

(Rd ← MEMDAT(dir))

: Registro visible
: Registro oculto

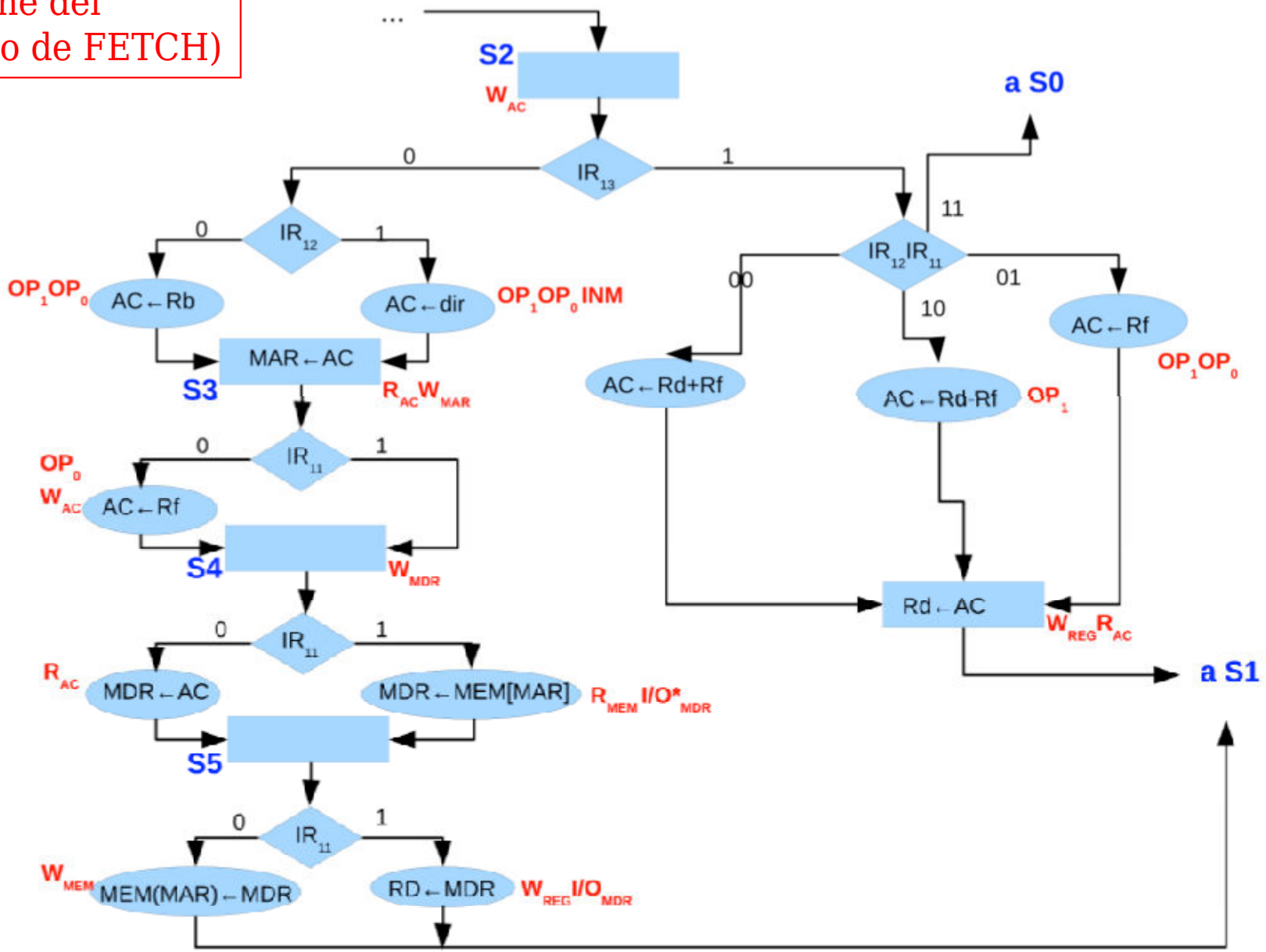


formato	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A instrucción con operando registro	código de operación			registro destino			Dirección del dato							

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

Carta ASM del CS2 (ciclo EXECUTE)

(viene del
Ciclo de FETCH)



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** $OP_1OP_0W_{AC}$	**AC←Rb** OP_1OP_0 W_{AC}	**AC←dir** $OP_1OP_0W_{AC}$INM	**AC←dir** $OP_1OP_0W_{AC}$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 programación del Computador Simple CS2

Realizar una subrutina 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
...	

Índice

1. Limitaciones de la calculadora simple
2. El Computador Simple 1 (CS1)

   (concepto de Programa almacenado en memoria)
3. El Computador Simple 2 (CS2)

   (memoria de datos y memoria de programa)
4. El Computador Simple CS2010

   (ampliación del conjunto de instrucciones)

Limitaciones del computador CS2

- El computador simple 2 presenta muchas limitaciones:
  - Imposibilidad de realizar saltos en la ejecución del programa
  - Ausencia de variables de estado que informen del resultado de las operaciones,
  - No permite direccionamiento inmediato
  - etc
- **Se propone** una nueva arquitectura pensada para solventar estas deficiencias: **el CS2010**.

- CS2010 es un computador "académico", pero compatible en juego de instrucciones con un microcontrolador real (AVR)

Modelo de programador del CS2010

- 8 Registros de propósito general: R0, R1, R2, R3, R4, R5, R6 y R7 (para suministrar datos a la ALU).
- 3 Registros de propósito específico:
  - **PC (Program counter)**:

    Contiene la dirección de la próxima instrucción que se ejecutará. Se inicializa a cero y se va incrementando a medida que se ejecutan las instrucciones.
  - **SR (Status Register):**

    Dan información sobre el tipo de resultado de la última operación realizada en la ALU: Z (cero), V (desbordamiento), N (negativo) y C (carry/borrow).
  - **SP (Stack Pointer):** Puntero de pila (memoria LIFO).

    SP apunta hacia la primera posición libre de la pila.

    PUSH (escribir en la pila): MEM(SP) ← REG; SP← SP-1

    POP ó PULL (leer de la pila): SP←SP+1; REG ← [MEM(SP)]

    En CS2010 sólo las instrucciones de subrutinas (CALL, RET) usan SP

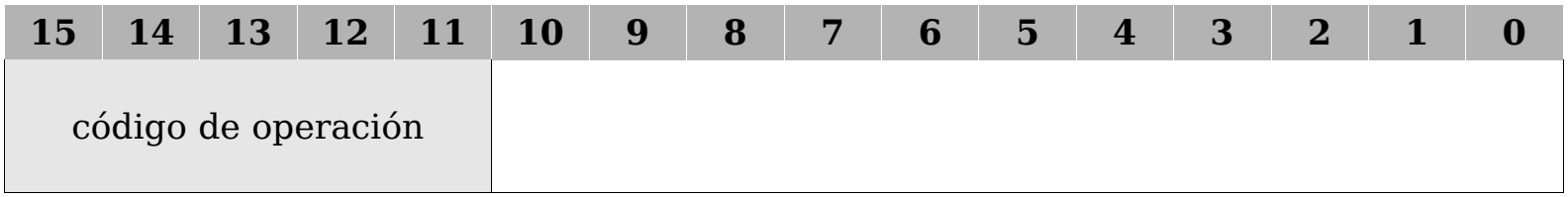
Formato de instrucciones del CS2010

- El juego de instrucciones (ISP) (mnemónicos del ensamblador) del CS2010 es un subconjunto del de la arquitectura AVR.

  Los programas en ensamblador del CS2010 pueden reensamblarse y ejecutarse en el AVR

  No es compatible a nivel de código máquina

- Instrucciones de 16 bits, con códigos de operación de 5 bits (21 tipos de instrucción diferentes)



Formato de instrucciones del CS2010

Formatos: Solo hay tres y comparten campos

<table>
<tr><th><u>formato</u></th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>A
instrucción con operando registro</td><td colspan="5" rowspan="3">código de operación</td><td colspan="3" rowspan="2">registro destino (fuente en ST)</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td colspan="4">registro fuente (registro base en ST/LD)</td></tr>
<tr><td>B
instrucción con operando memoria o inmediato</td><td colspan="9" rowspan="2">dato inmediato / dirección del dato</td></tr>
<tr><td>C
instrucción de salto</td><td colspan="3">condición de salto</td></tr>
<tr><td colspan="9"></td><td colspan="8">dirección de salto</td></tr>
</table>

Códigos de operación CS2010

Bits del código de operación					NEMÓNICO	FORMATO	TIPO	SINTAXIS	EFECTO[1]	VCNZ[2]
15	14	13	12	11						
0	0	0	0	0	ST	A	memoria	ST (Rbase),Rfuente	MEM[Rbase]←Rfuente	----
0	0	0	0	1	LD	A	memoria	LD Rdestino,(Rbase)	Rfuente←MEM[Rbase]	----
0	0	0	1	0	STS	B	memoria	STS dirección,Rfuente	MEM[dirección]←Rfuente	----
0	0	0	1	1	LDS	B	memoria	LDS Rdestino,dirección	Rfuente←MEM[dirección]	----
0	0	1	0	0	CALL	C	salto	CALL dirección	MEM[SP]←PC,SP←SP-1, PC←dirección	----
0	0	1	0	1	RET	-	salto	RET	PC←MEM[SP+1],SP←SP+1	----
0	0	1	1	0	BR*xx*	C	salto	BR*xx* dirección	*xx*:PC←dirección	----
0	0	1	1	1	JMP	C	salto	JMP dirección	PC←dirección	----
0	1	0	0	0	ADD	A	aritmético/lógica	ADD Rdestino,Rfuente	Rdestino←Rdestino+Rfuente	****
0	1	0	0	1	-	-	-	-	no documentado	UUUU
0	1	0	1	0	SUB	A	aritmético/lógica	SUB Rdestino,Rfuente	Rdestino←Rdestino-Rfuente	****
0	1	0	1	1	CP	A	estado	CP Rdestino,Rfuente	NOP	****
0	1	1	0	0	-	-	-	-	no documentado	UUUU
0	1	1	0	1	-	-	-	-	no documentado	UUUU
0	1	1	1	0	-	-	-	-	no documentado	UUUU
0	1	1	1	1	MOV	A	movimiento de datos	MOV Rdestino,Rfuente	Rdestino←Rdestino	----
1	0	0	0	0	-	-	-	-	no documentado	UUUU
1	0	0	0	1	-	-	-	-	no documentado	UUUU
1	0	0	1	0	CLC	-	estado	CLC	NOP	---*
1	0	0	1	1	SEC	-	estado	SEC	NOP	---*
1	0	1	0	0	ROR	A o B	desplazamiento	ROR Rdestino	Rdestino←SHR(Rdestino,C)	****
1	0	1	0	1	ROL	A o B	desplazamiento	ROL Rdestino	Rdestino←SHL(Rdestino,C)	****
1	0	1	1	0	-	-	-	-	no documentado	UUUU
1	0	1	1	1	STOP	-	especial	STOP	lleva el procesador a espera	----
1	1	0	0	0	ADDI	B	aritmético/lógica	ADDI Rdestino,dato	Rdestino←Rdestino+dato	****
1	1	0	0	1	-	-	-	-	no documentado	UUUU
1	1	0	1	0	SUBI	B	aritmético/lógica	SUBI Rdestino,dato	Rdestino←Rdestino-dato	****
1	1	0	1	1	CPI	B	estado	CPI Rdestino,dato	NOP	****
1	1	1	0	0	-	-	-	-	no documentado	UUUU
1	1	1	0	1	-	-	-	-	no documentado	UUUU
1	1	1	1	0	-	-	-	-	no documentado	UUUU
1	1	1	1	1	LDI	B	movimiento de datos	LDI Rdestino,dato	Rdestino←dato	----

[1] (sin tener en cuenta el registro de estado y el incremento del PC)

[2] El caracter '-' denota “no modificado”, '*' denota “modificado de forma definida”, 'U' denota “no documentado”

Códigos de operación CS2010

Bits del código de operación					NEMÓNICO	FORMATO	TIPO	SINTAXIS	EFECTO[1]	VCNZ[2]
15	14	13	12	11						
0	0	0	0	0	ST	A	memoria	ST (Rbase),Rfuente	MEM[Rbase]←Rfuente	----
0	0	0	0	1	LD	A	memoria	LD Rdestino,(Rbase)	Rfuente←MEM[Rbase]	----
0	0	0	1	0	STS	B	memoria	STS dirección,Rfuente	MEM[dirección]←Rfuente	----
0	0	0	1	1	LDS	B	memoria	LDS Rdestino,dirección	Rfuente←MEM[dirección]	----
0	0	1	0	0	CALL	C	salto	CALL dirección	MEM[SP]←PC,SP←SP-1, PC←dirección	----
0	0	1	0	1	RET	-	salto	RET	PC←MEM[SP+1],SP←SP+1	----
0	0	1	1	0	BR*xx*	C	salto	BR*xx* dirección	*xx*:PC←dirección	----
0	0	1	1	1	JMP	C	salto	JMP dirección	PC←dirección	----
0	1	0	0	0	ADD	A	aritmético/lógica	ADD Rdestino,Rfuente	Rdestino←Rdestino+Rfuente	****
0	1	0	0	1	-	-	-	-	no documentado	UUUU
0	1	0	1	0	SUB	A	aritmético/lógica	SUB Rdestino,Rfuente	Rdestino←Rdestino-Rfuente	****
0	1	0	1	1	CP	A	estado	CP Rdestino,Rfuente	NOP	****
0	1	1	0	0	-	-	-	-	no documentado	UUUU
0	1	1	0	1	-	-	-	-	no documentado	UUUU
0	1	1	1	0	-	-	-	-	no documentado	UUUU
0	1	1	1	1	MOV	A	movimiento de datos	MOV Rdestino,Rfuente	Rdestino←Rdestino	----
1	0	0	0	0	-	-	-	-	no documentado	UUUU
1	0	0	0	1	-	-	-	-	no documentado	UUUU
1	0	0	1	0	CLC	-	estado	CLC	NOP	---*
1	0	0	1	1	SEC	-	estado	SEC	NOP	---*
1	0	1	0	0	ROR	A o B	desplazamiento	ROR Rdestino	Rdestino←SHR(Rdestino,C)	****
1	0	1	0	1	ROL	A o B	desplazamiento	ROL Rdestino	Rdestino←SHL(Rdestino,C)	****
1	0	1	1	0	-	-	-	-	no documentado	UUUU
1	0	1	1	1	STOP	-	especial	STOP	lleva el procesador a espera	----
1	1	0	0	0	ADDI	B	aritmético/lógica	ADDI Rdestino,dato	Rdestino←Rdestino+dato	****
1	1	0	0	1	-	-	-	-	no documentado	UUUU
1	1	0	1	0	SUBI	B	aritmético/lógica	SUBI Rdestino,dato	Rdestino←Rdestino-dato	****
1	1	0	1	1	CPI	B	estado	CPI Rdestino,dato	NOP	****
1	1	1	0	0	-	-	-	-	no documentado	UUUU
1	1	1	0	1	-	-	-	-	no documentado	UUUU
1	1	1	1	0	-	-	-	-	no documentado	UUUU
1	1	1	1	1	LDI	B	movimiento de datos	LDI Rdestino,dato	Rdestino←dato	----

[1] (sin tener en cuenta el registro de estado y el incremento del PC)

[2] El caracter '-' denota “no modificado”, '*' denota “modificado de forma definida”, 'U' denota “no documentado”

Códigos de condición de la instrucción de salto condicional BRXX *(Branch if…)*

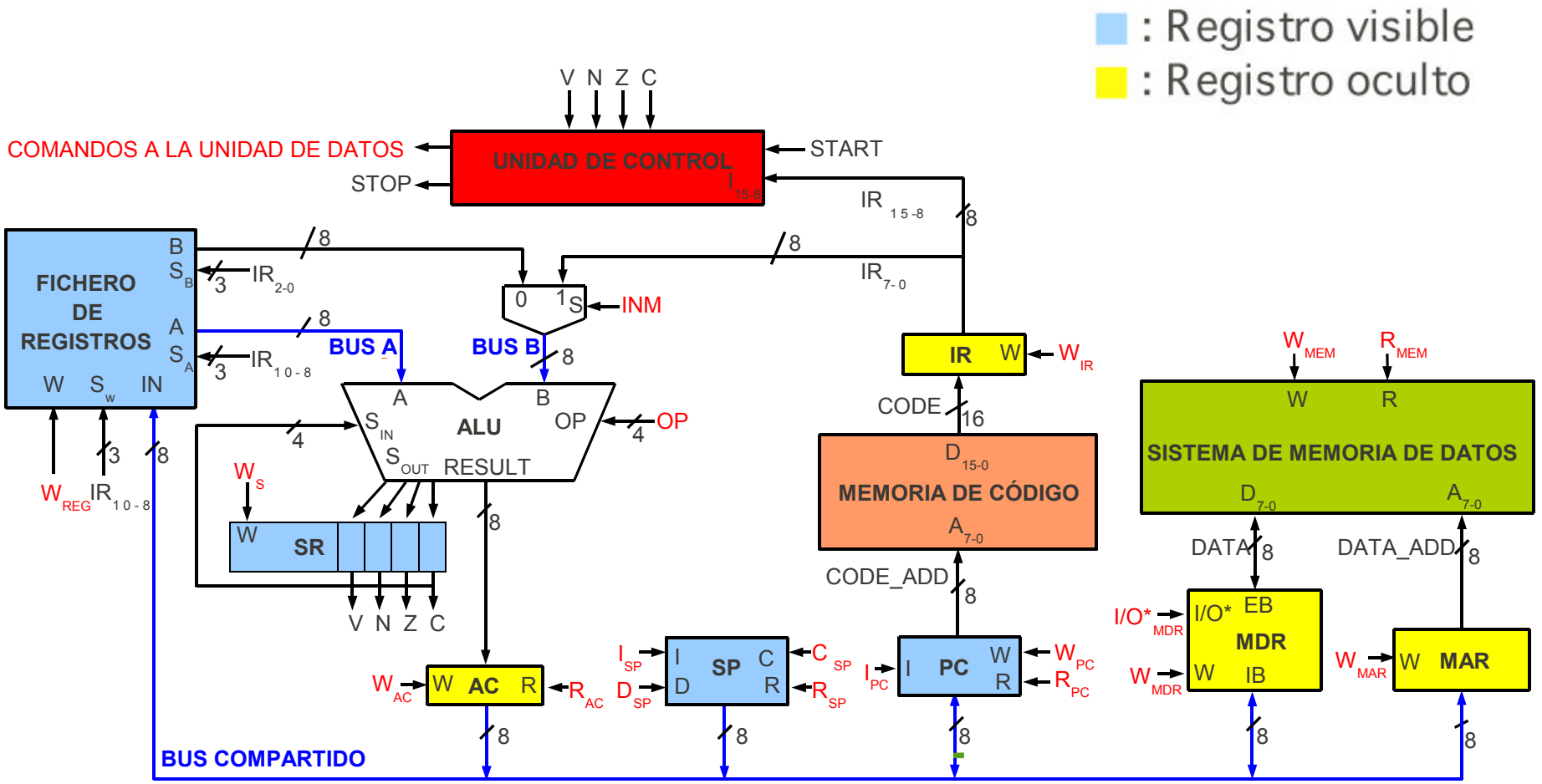
Los bits $I_{10}I_9I_8$ codifican la condición de salto *xx*.

I_{10}	I_9	I_8	CONDICIÓN	nmónico(s) de la condición	notas
0	0	0	Z	ZS, EQ	será cierta justo tras realizar la resta A-B si y solo si A=B
0	0	1	C	CS, LO	será cierta justo tras realizar la resta A-B si y solo si A<B asumiendo notación base 2 sin signo
0	1	0	V	VS	será cierta si y solo si el dato recién calculado no es representable en notación base 2 sin signo
0	1	1	N xor V	LT	será cierta justo tras realizar la resta A-B si y solo si A<B asumiendo notación complemento a 2
1	-	-	?	-	estas condiciones no están definidas y no deben utilizarse

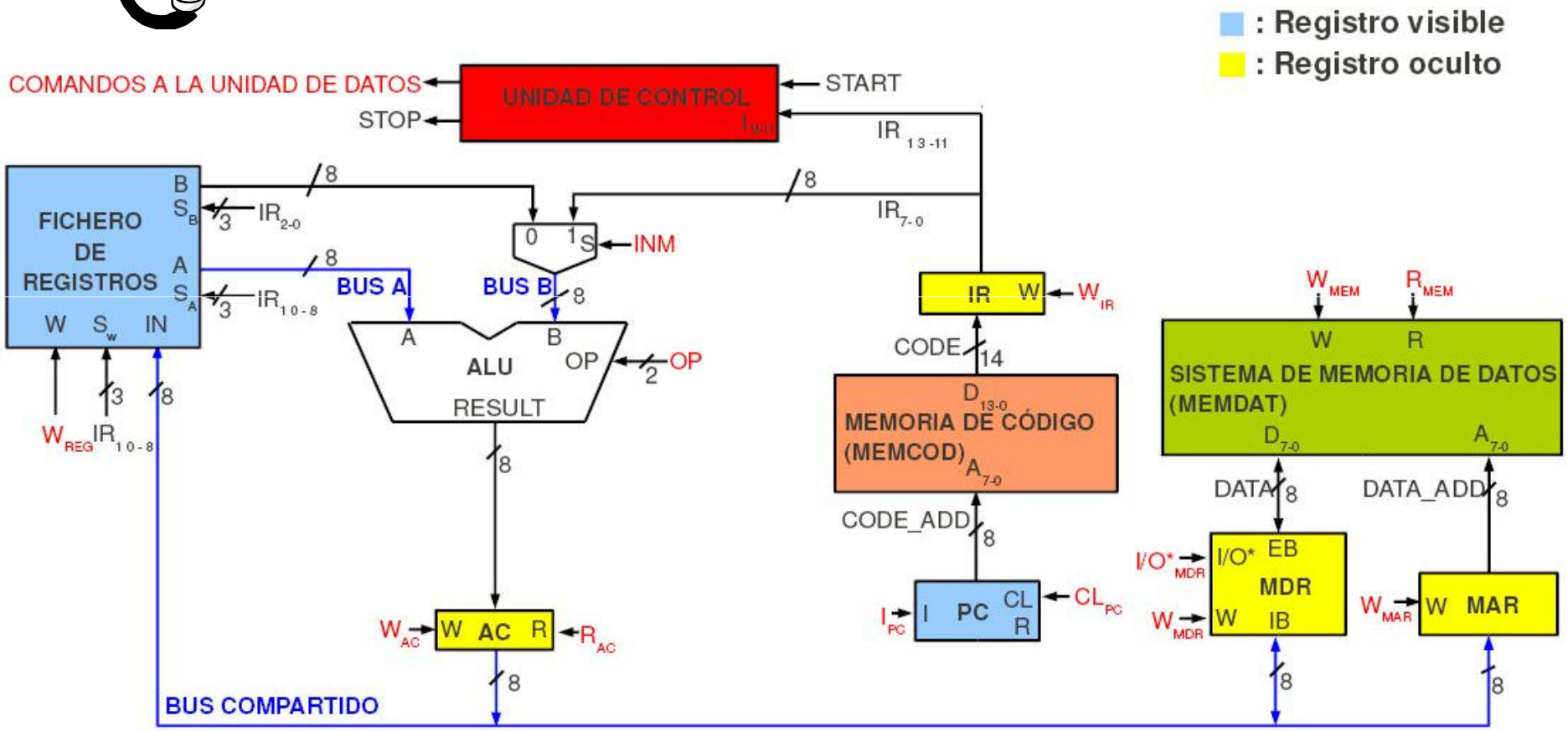
NOTAS:

- La condición Z con el mnemónico “ZS” no existe en AVR, por lo que BRZS no podrá ser ensamblada en AVR
- La condición V con el mnemónico “VS” es para números CON signo

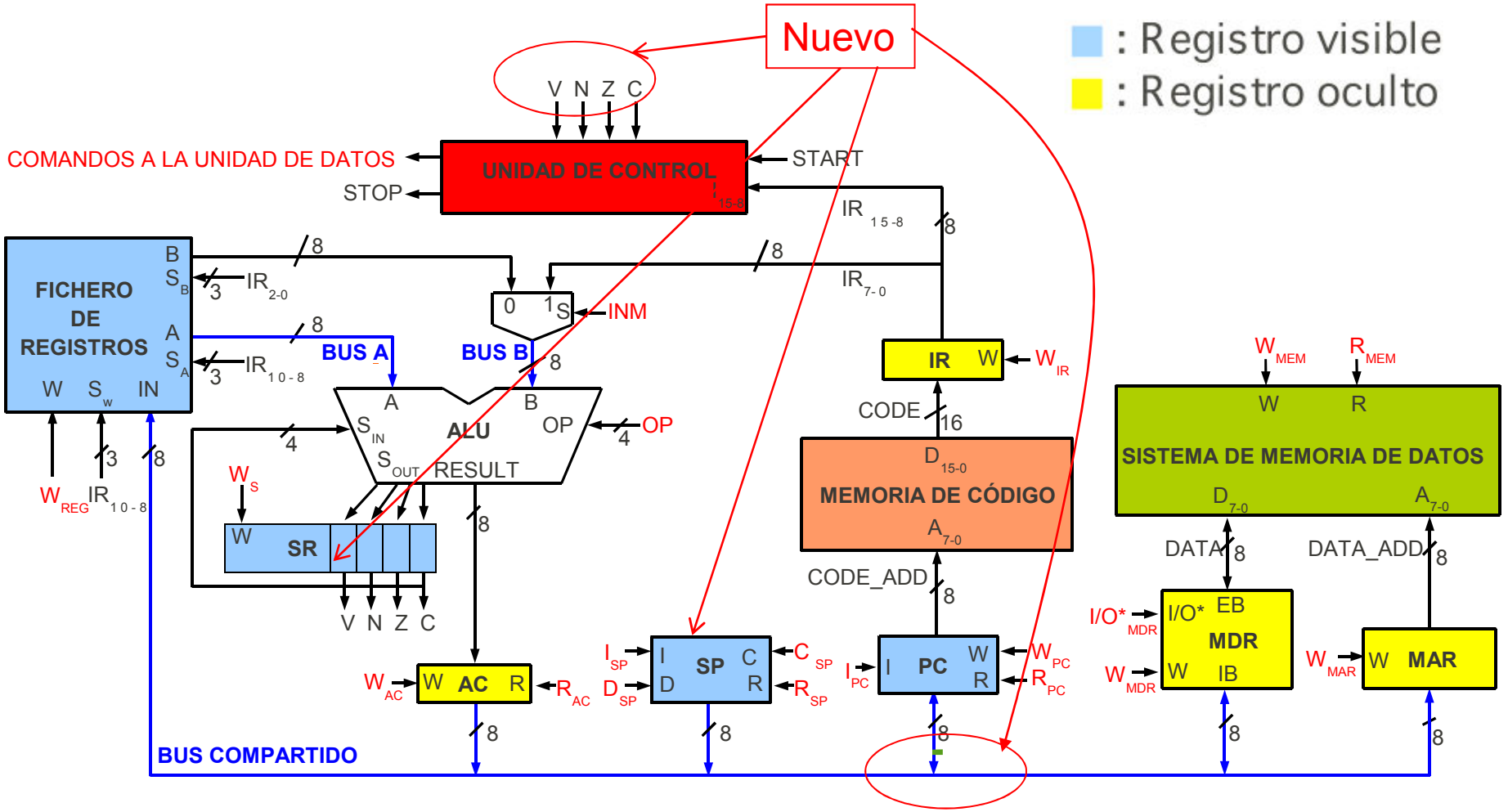
Arquitectura propuesta para el CS2010



Arquitectura del CS2



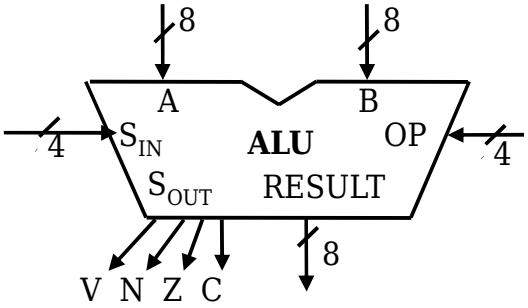
Arquitectura propuesta para el CS2010



Registros “ocultos al programador” de la arquitectura del CS2010

- **IR[16] (Instruction Register):** sirve para almacenar la instrucción que está siendo ejecutada.
- **MDR[8] (Memory Data Register):** Almacenar los datos que se intercambian con la memoria de datos.
- **MAR[8] (Memory Address Register):** Sirve para direccionar la memoria de datos
- **AC[8]:** Almacena el resultado de la opración realizada por la ALU

Descripción RT de los nuevos registros del Computador Simple CS2010 (i)

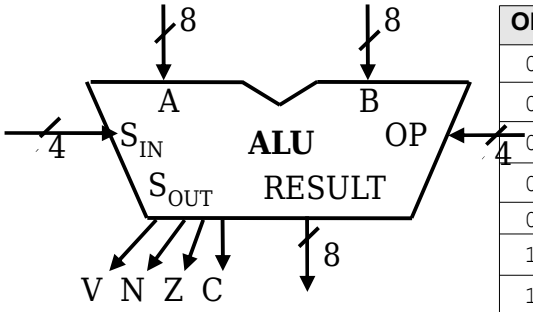


C_{in} XOR A_7

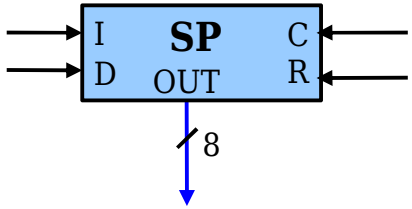
OP_3	OP_2	OP_1	OP_0	RESULT=	V_{OUT}=	N_{OUT}=	Z_{OUT}=	C_{OUT}=
0	0	-	0	-	V_{IN}	N_{IN}	Z_{IN}	0
0	0	1	1	-	V_{IN}	N_{IN}	Z_{IN}	1
0	1	0	0	SHR(A,C_{IN})	C_{IN} EXOR A_0	$RESULT_7$	NOT $OR_{i=0}^{7}(RESULT_i)$	A_0
0	1	0	1	SHL(A,C_{IN})	A_7 EXOR A_6	$RESULT_7$	NOT $OR_{i=0}^{7}(RESULT_i)$	A_7
0	1	1	-	A	-	-	-	-
1	0	0	-	(A + B) mod 2^8	overflow(A+B)	$RESULT_7$	NOT $OR_{i=0}^{7}(RESULT_i)$	carry(A+B)
1	0	1	-	(A - B) mod 2^8	underflow(A-B)	$RESULT_7$	NOT $OR_{i=0}^{7}(RESULT_i)$	borrow(A-B)
1	1	-	-	B	-	-	-	-

Mod 2^8. Corregir “underflow y explicar “overflow”

Descripción RT de los nuevos registros del Computador Simple CS2010 (ii)



OP_3	OP_2	OP_1	OP_0	RESULT=	V_{OUT}=	N_{OUT}=	Z_{OUT}=	C_{OUT}=
0	0	-	0	-	V_{IN}	N_{IN}	Z_{IN}	0
0	0	1	1	-	V_{IN}	N_{IN}	Z_{IN}	1
0	1	0	0	SHR(A,C_{IN})	C_{IN} EXOR A_0	$RESULT_7$	NOT $OR_{i=0}^{7}(RESULT_i)$	A_0
0	1	0	1	SHL(A,C_{IN})	A_7 EXOR A_6	$RESULT_7$	NOT $OR_{i=0}^{7}(RESULT_i)$	A_7
0	1	1	-	A	-	-	-	-
1	0	0	-	(A + B) mod 2□	overflow(A+B)	$RESULT_7$	NOT $OR_{i=0}^{7}(RESULT_i)$	carry(A+B)
1	0	1	-	(A - B) mod 2□	underflow(A-B)	$RESULT_7$	NOT $OR_{i=0}^{7}(RESULT_i)$	borrow(A-B)
1	1	-	-	B	-	-	-	-

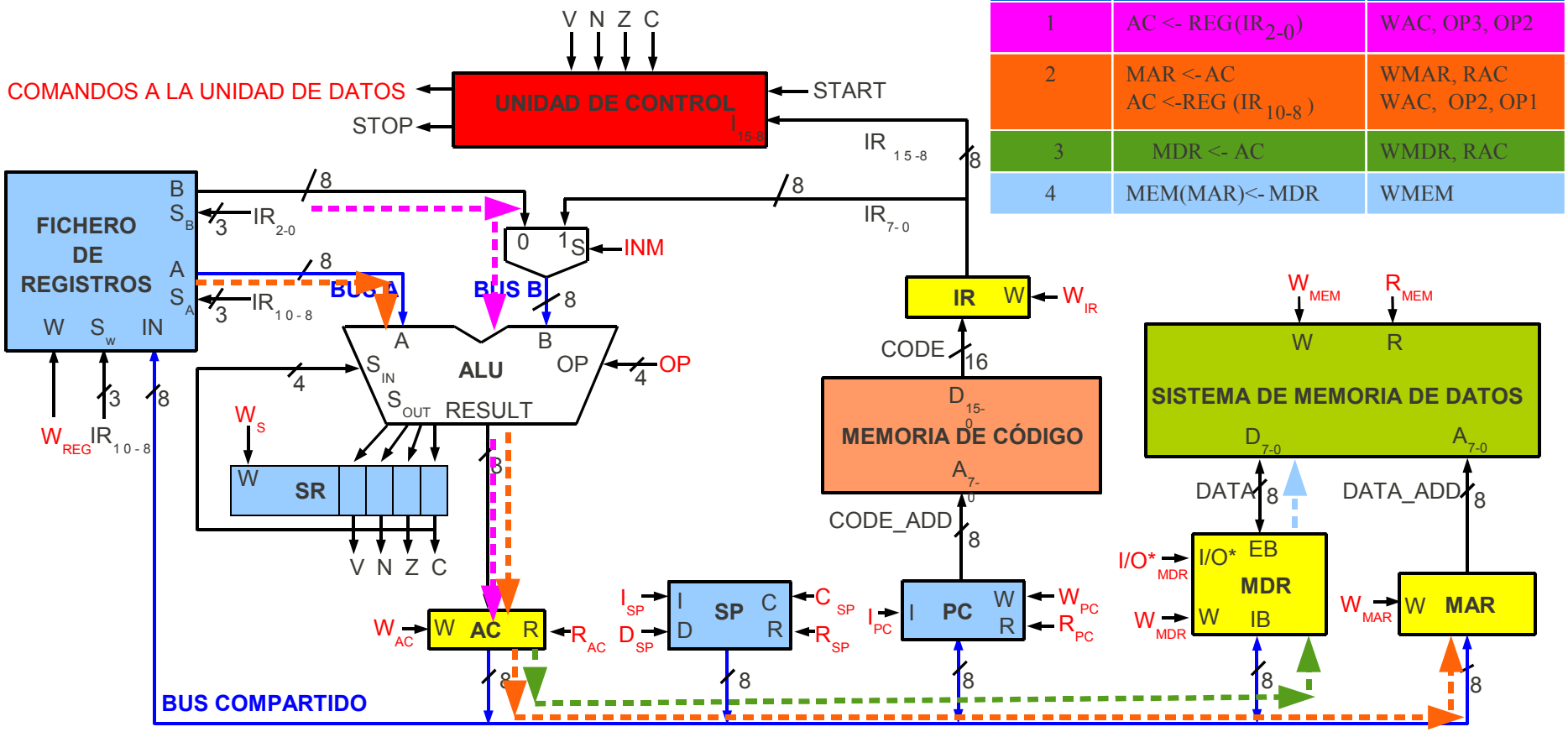


I D C R	SP ←	OUT:=
0 0 0 0	SP	HI
0 0 0 1	SP	SP
0 0 1 0	0	HI
0 1 0 0	SP-1	HI
1 0 0 0	SP+1	HI
OTRAS	PROHIBIDAS	

- La descripción de SR es idéntica a la del MAR

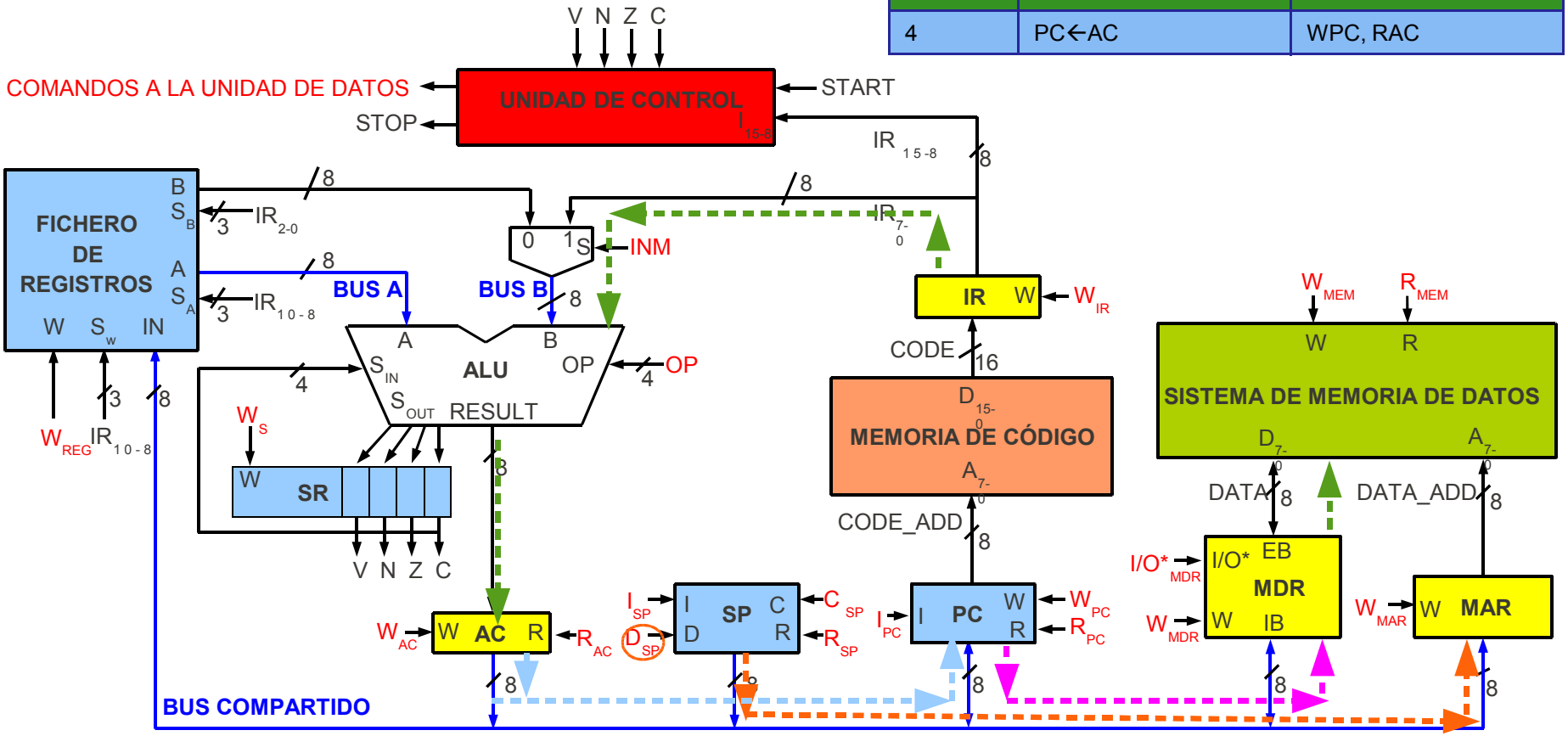
Ejemplo de secuencia de microoperaciones de la instrucción ST (Rj),Ri *(Store in memory)*

CICLO	MEM(Rj) <- Ri	
1	AC <- REG(IR_{2-0})	WAC, OP3, OP2
2	MAR <- AC AC <-REG (IR_{10-8})	WMAR, RAC WAC, OP2, OP1
3	MDR <- AC	WMDR, RAC
4	MEM(MAR)<- MDR	WMEM



Ejemplo de secuencia de microoperaciones de la instrucción CALL dirección *(Llamada a subrutina)*

CICLO	MEM(SP) ← PC; SP← SP-1; PC← dirección	
1	MDR ←PC	WMDR, RPC
2	MAR ← SP, SP←SP-1	WMAR, RSP, DSP
3	AC←IR 7-0 MEM(MAR)←MDR	INM, OP3, OP2, WAC WMEM
4	PC←AC	WPC, RAC



Nota

La descripción RT del resto de las instrucciones se encuentra en un documento anexo a este.

El alumno debe trabajar con ese documento.

Ejemplo de programación en ensamblador del CS2010

Ejemplo 1. *Escriba una subrutina para el cálculo de la multiplicación mediante el algoritmo de sumas sucesivas.*

```
; Ejemplo 1:
; multiplica los datos de los registros R1 y R2
; y devuelve el resultado en R0 (truncando a 8 bits)
; sumo el dato de R1 tantas veces como diga R2

MULT:      LDI R0,0   ;inicializa R0
           CPI R2,0
           BREQ RETORNA
BUCLE:     ADD R0,R1
           SUBI R2,1
           BREQ RETORNA
           JMP BUCLE
RETORNA:   RET
```

Ejemplo de programación en ensamblador del CS2010

Ejemplo 2. *Escriba una subrutina que devuelva el mayor de dos números (escritos en complemento a 2)*

```
; Ejemplo 2:
; devuelve en R0 el mayor de los datos almacenados
; en los registros R1 y R2

MAX:        MOV R0,R1
            SUB R0,R2
            BRLT R2MAYOR
            MOV R0,R1
            RET
R2MAYOR:    MOV R0,R2
            RET
```

Ejemplo de programación en ensamblador del CS2010

Ejemplo 3. *Escriba una subrutina que escriba, en orden descendente, los números del 100 al 1 en una tabla situada a partir de la posición de memoria 123*

```
; Ejemplo 3:
; Rellena una tabla en orden descendente, del 100 a 1
; la tabla empieza en la dirección 123

CUENTA:    LDI R0,100 ;primer dato de la tabla
           LDI R1,123
BUCLE:     ST R0,(R1) ;R1 es el Registro Base
           ADDI R1,1
           SUBI R0,1
           BREQ RETORNA
           JMP BUCLE
RETORNA:   RET
```

Índice

1. Limitaciones de la calculadora simple
2. El Computador Simple 1 (CS1)

   (concepto de Programa almacenado en memoria)
3. El Computador Simple 2 (CS2)

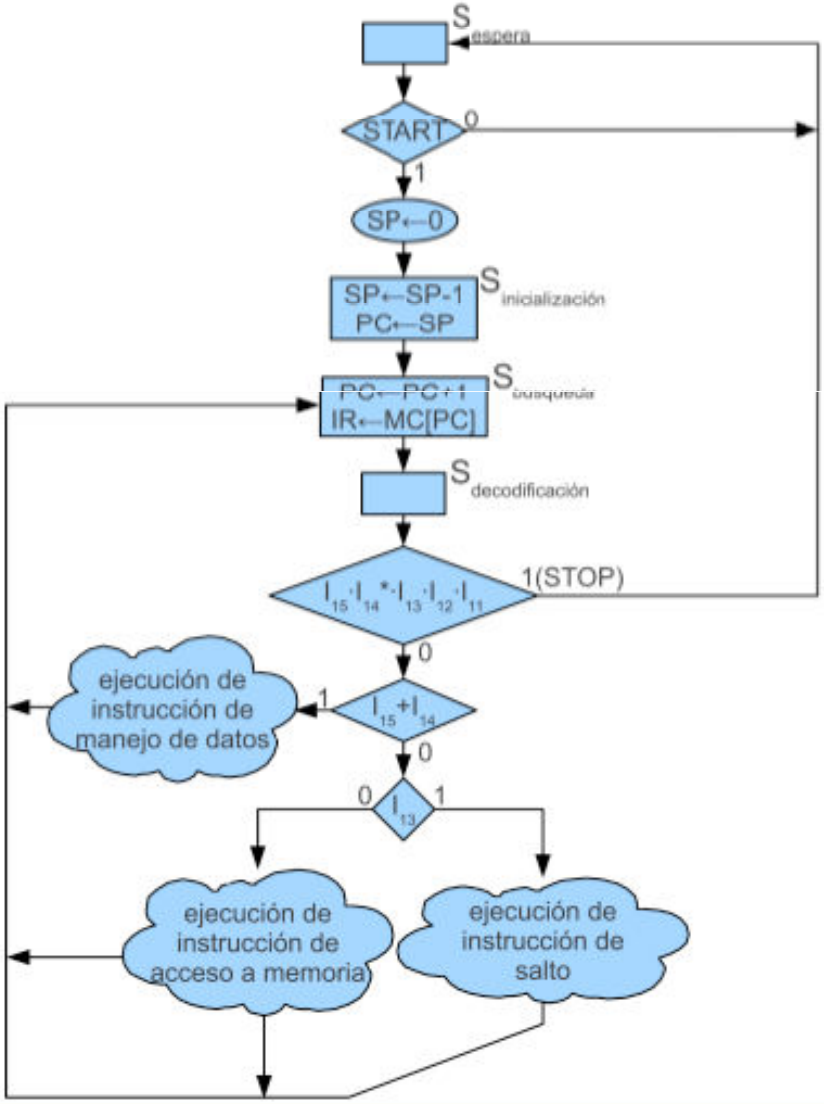
   (memoria de datos y memoria de programa)
4. El Computador Simple CS2010

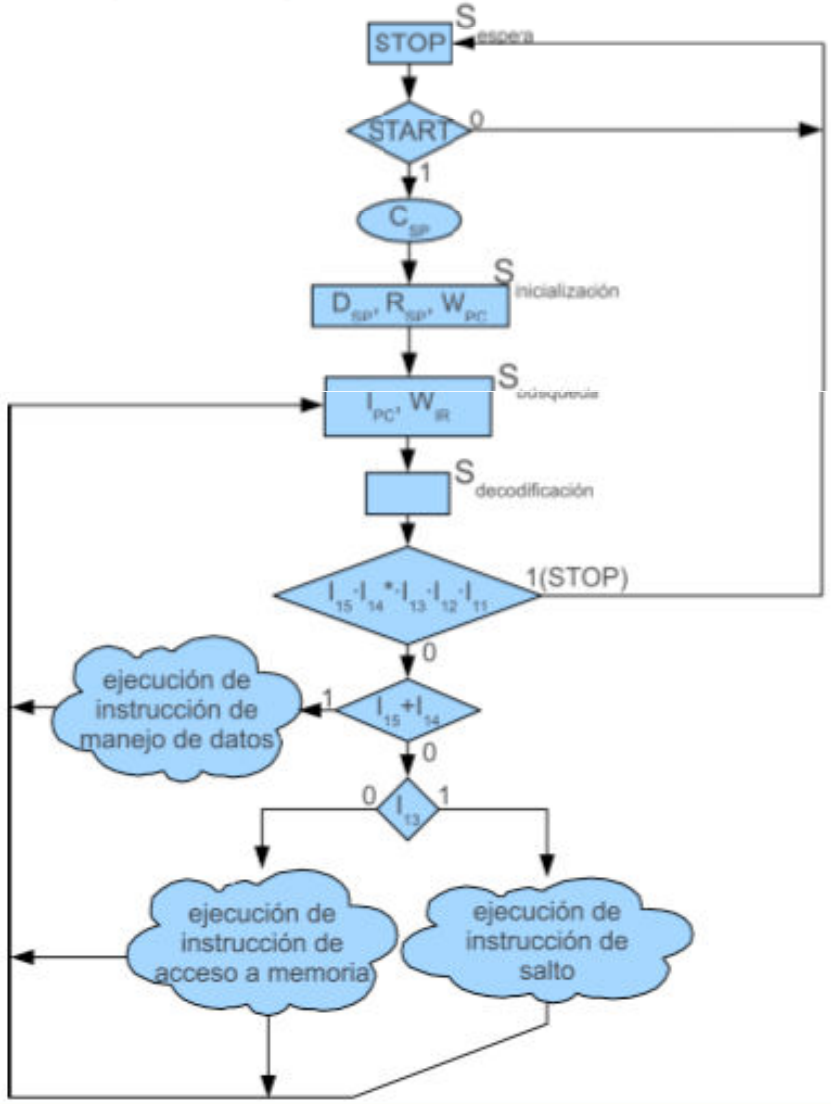
   (ampliación del conjunto de instrucciones)

Apéndice: Cartas ASM del Computador Simple CS2010

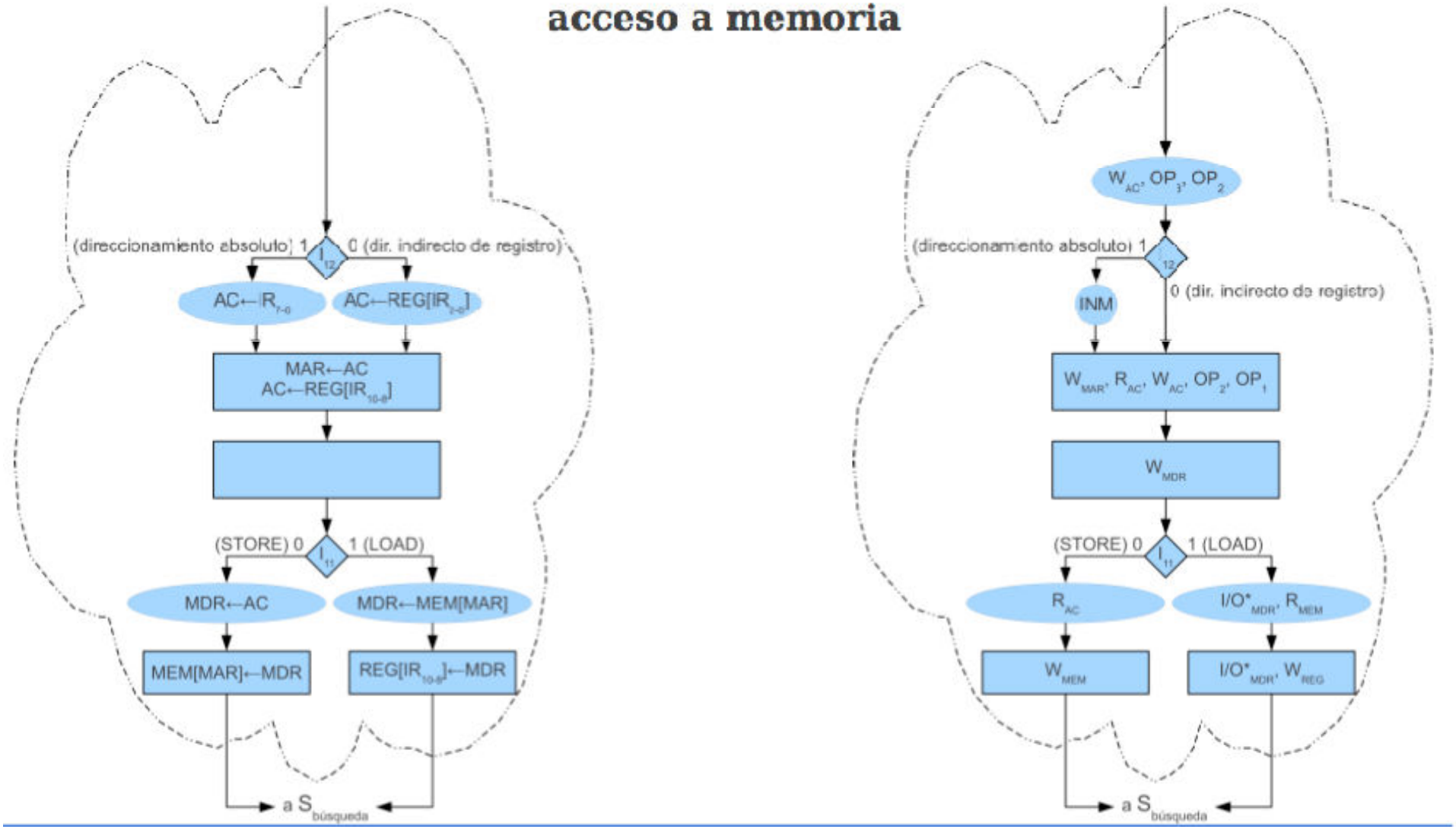
Cartas ASM del CS2010

Cartas ASM de datos y código

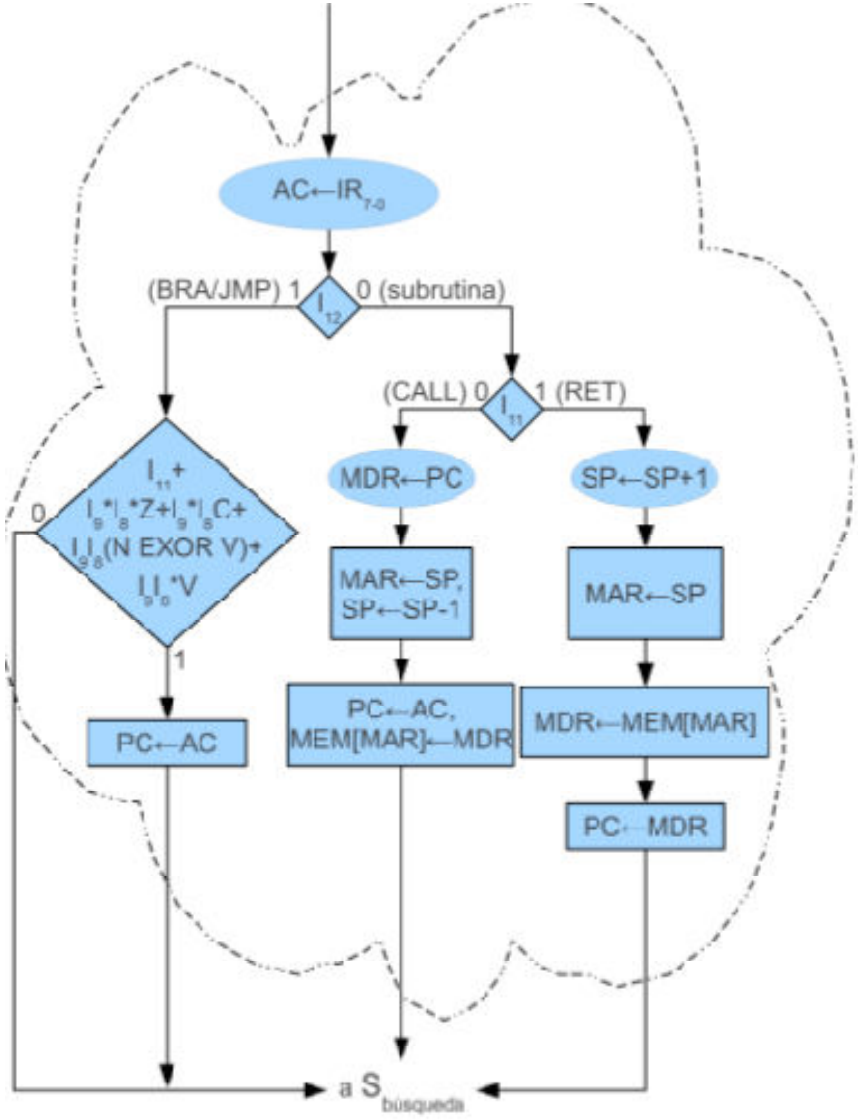




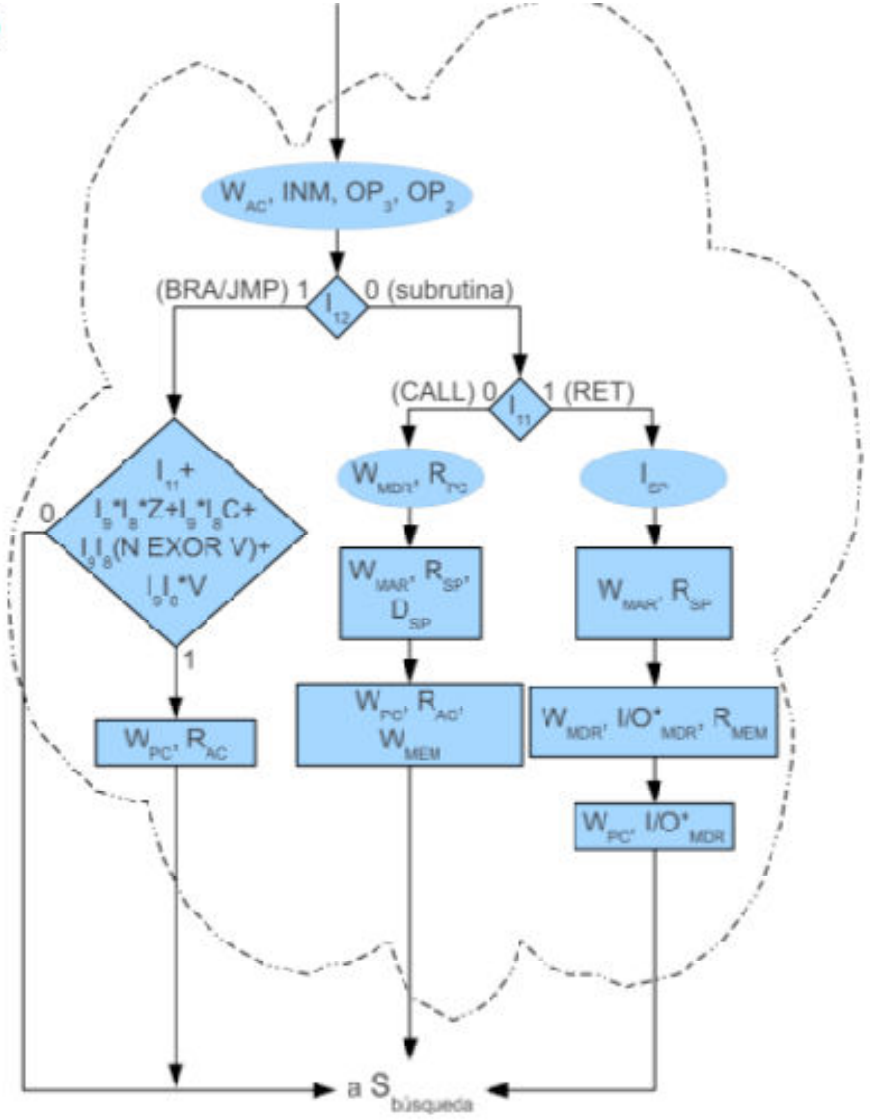
Cartas ASM del CS2010 (ii)



Cartas ASM del CS2010 (iii)

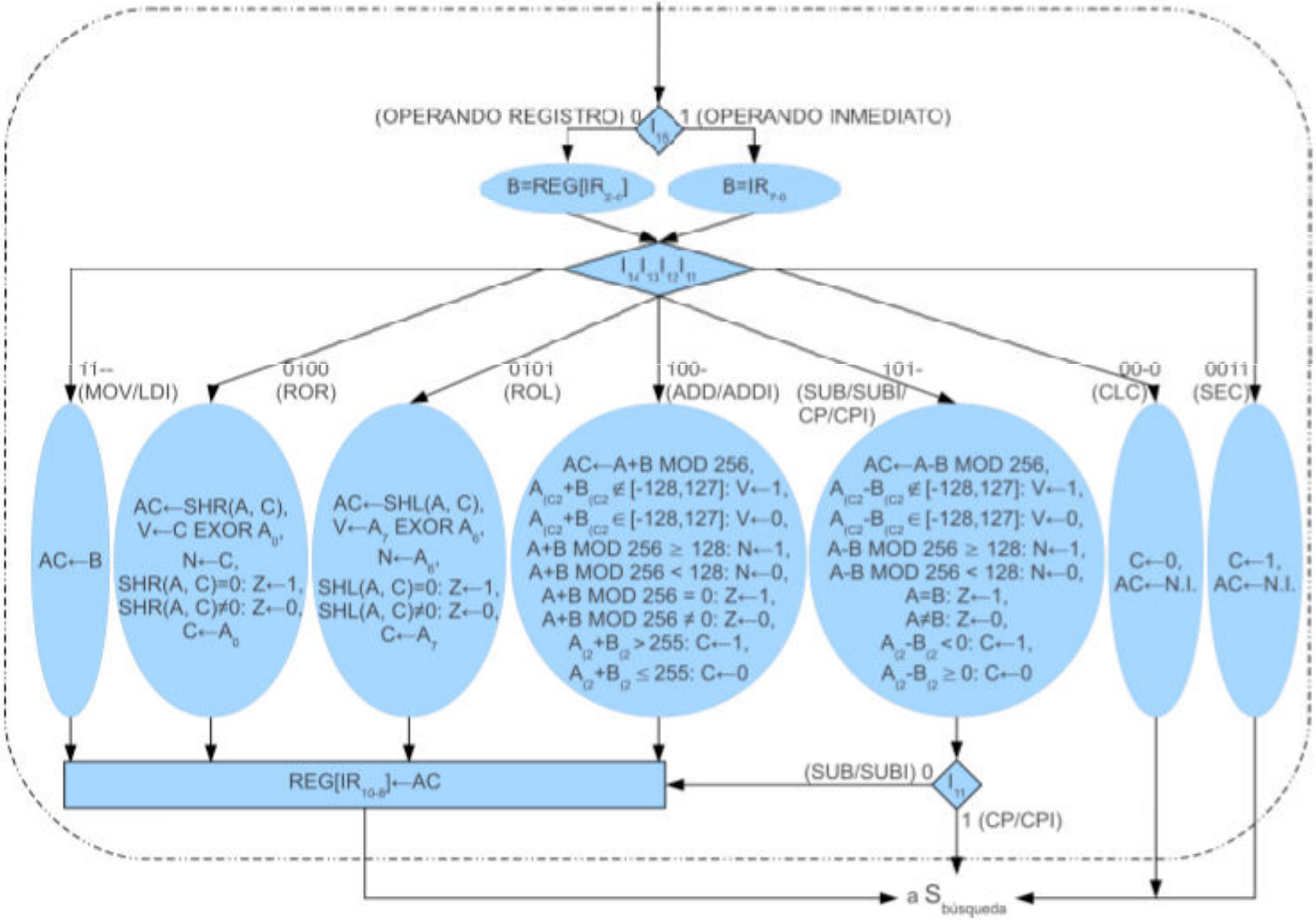


salto



Cartas ASM del CS2010 (iv)

manejo de datos (trozo de la carta de datos)



Cartas ASM del CS2010 (v)