Estructura de Computadores

El computador simple

Autores: David Guerrero. Isabel Gómez,Alberto Molina

Guión

- **El punto de partida: La calculadora**
- **Automatización en la ejecución y almacenamiento de programa (CS1)**
- **Almacenamiento de los datos y ampliación de modos de direccionamiento (CS2)**
- **Diversificación de instrucciones. Ejecución no secuencial (CS3)**

CS3

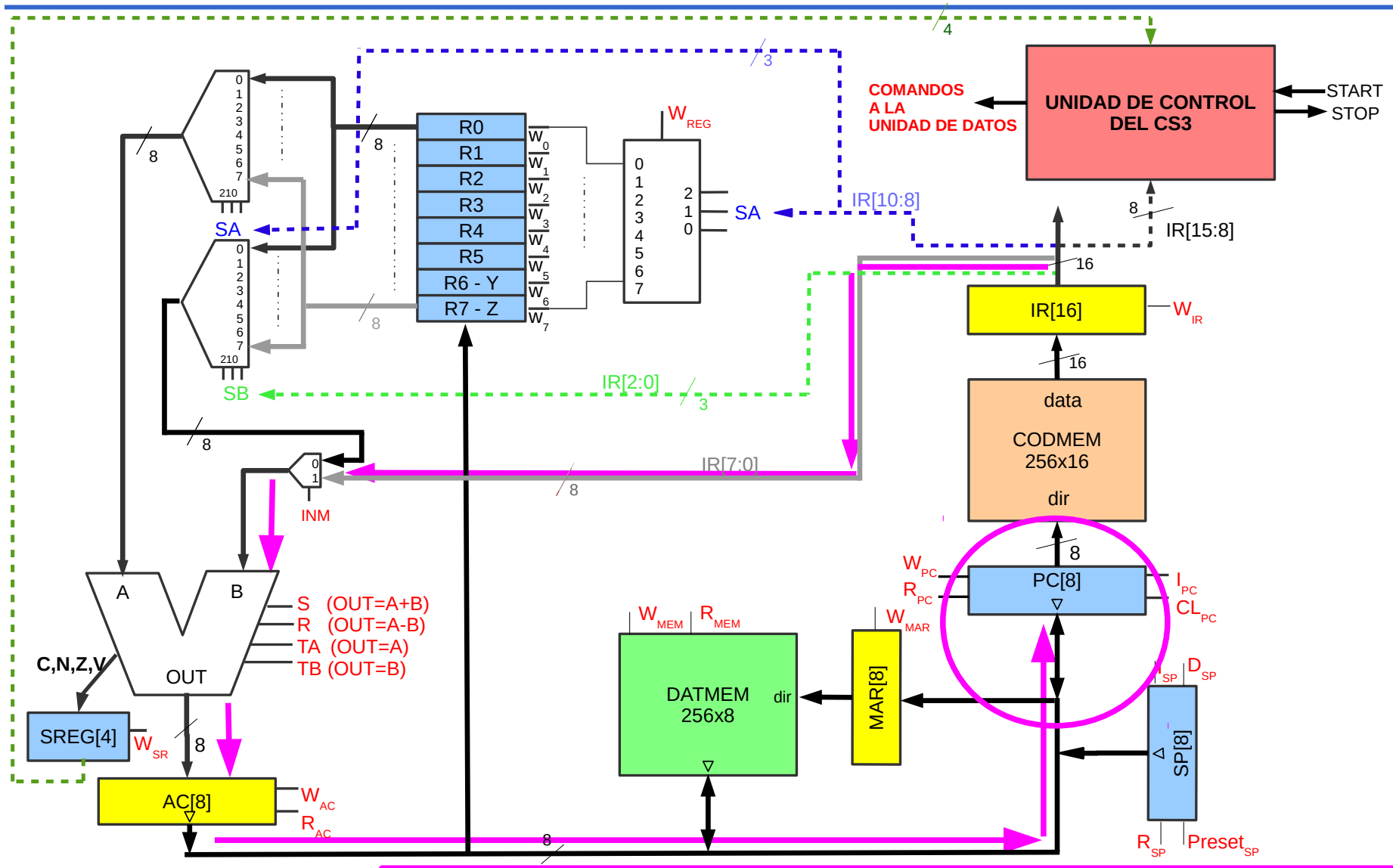
- El computador simple 2 presenta muchas deficiencias: imposibilidad de realizar saltos en la ejecución del programa, ausencia de variables de estado que informen del resultado de las operaciones, falta de comunicación con el exterior.
- Se propone una arquitectura pensada para solventar estas deficiencias: el CS3.
- Se ha procurado que la arquitectura del CS3 sea similar a la del microcontrolador que veremos en el tema siguiente, aunque muchísimo más simple.

CS3

Modelo de usuario: ISP

- Se pretende que el juego de instrucciones ensamblador del CS3 sea básicamente un subconjunto muy reducido del de la arquitectura AVR.
- Esto no implica que el formato del código máquina en ambos sistemas sea similar: El formato de instrucciones del CS3 es muchísimo más simple.

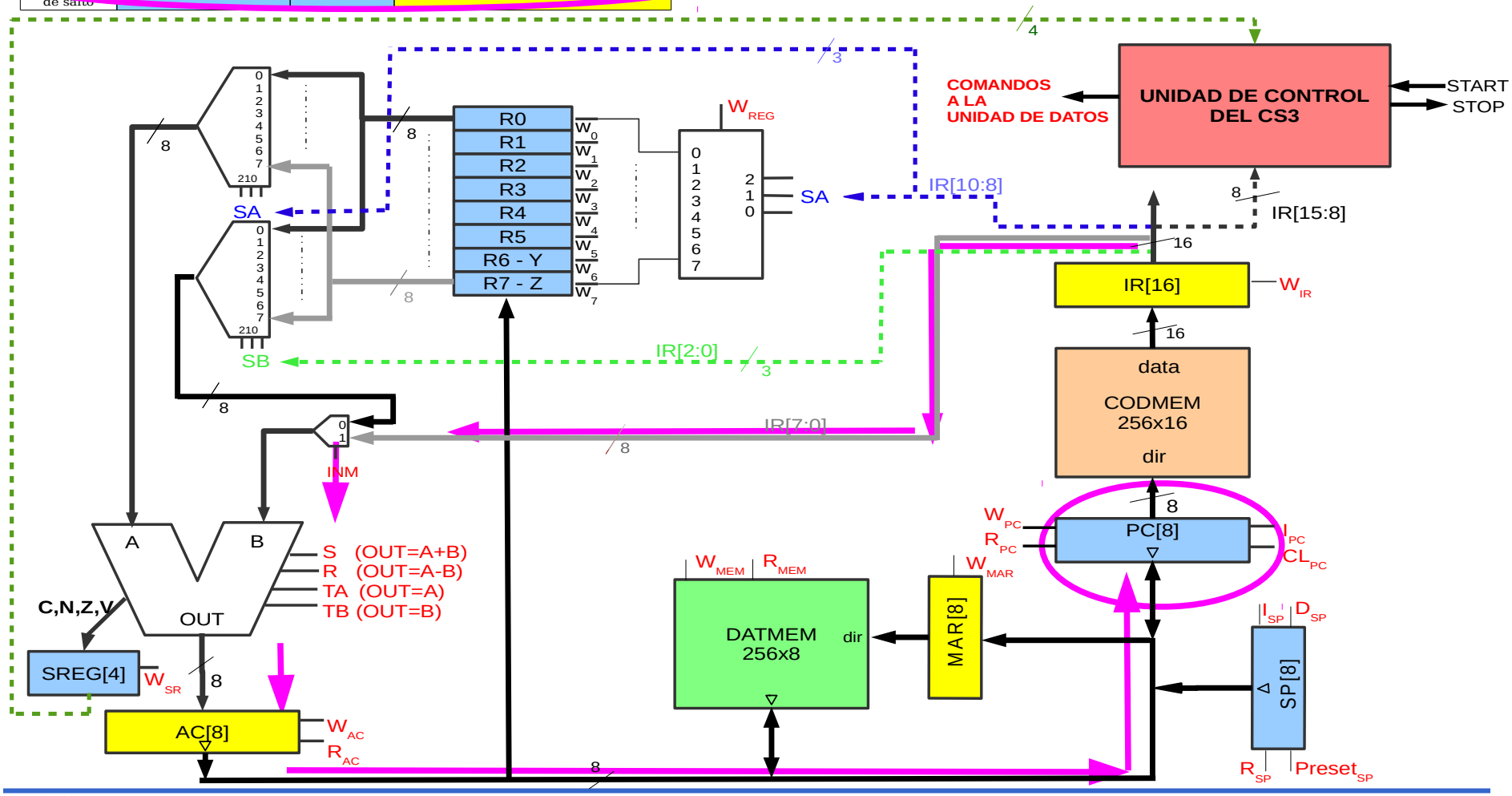
Modificaciones en la arquitectura y el formato de instrucción que nos permiten añadir instrucciones de salto

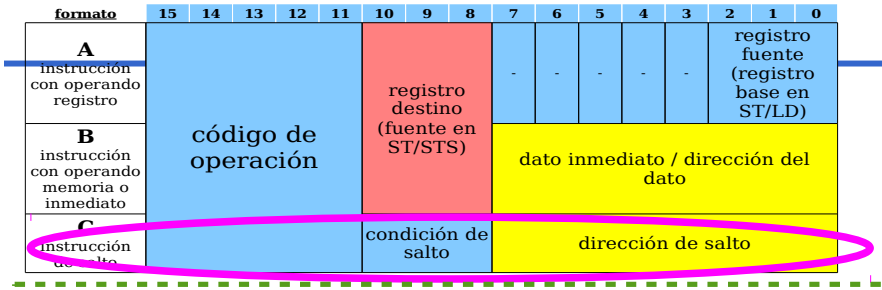


¿Cómo se hace el salto?: en el PC, ya no cambia siempre de forma secuencial, se debe actualizar a la dirección del salto.
¿Dónde está la dirección del salto?: guardada en el código de la instrucción.
El sistema debe de permitir guardar este dato en el PC. Necesitamos conectar el PC al bus para establecer un camino desde IR

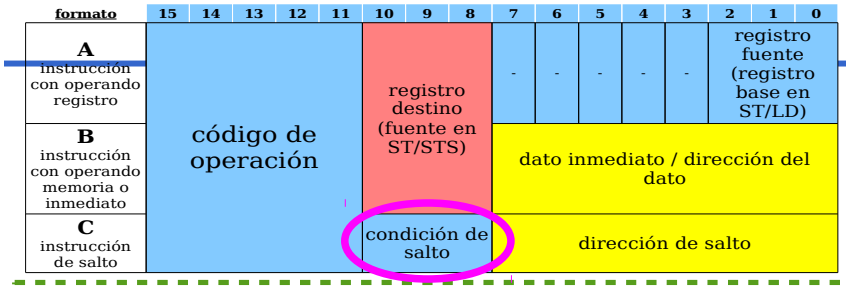
formato	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A instrucción con operando registro	código de operación					registro destino (fuente en ST/STS)			-	-	-	-	-	registro fuente (registro base en ST/LD)		
B instrucción con operando memoria o inmediato									dato inmediato / dirección del dato							
C instrucción de salto						condición de salto			dirección de salto							

Ha sido necesario añadir un nuevo formato que nos permitirá codificar todos los tipos de instrucciones de salto que explicaremos a continuación

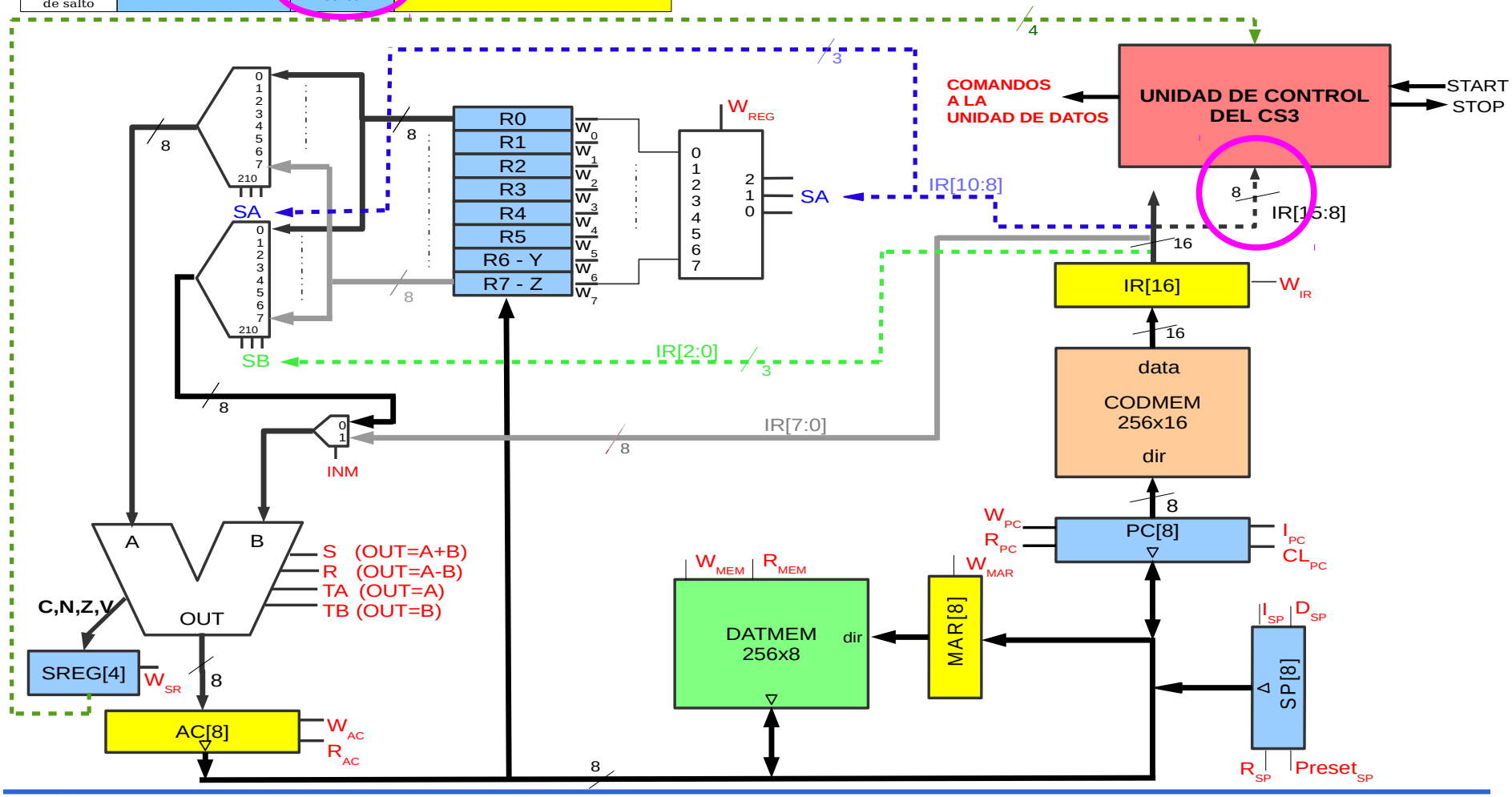




Salto incondicional.
Salta siempre, no se utiliza el campo condición de salto. Los 8 bits del IR se guardan en el PC y eso produce el salto.
La próxima instrucción que busque el sistema será en la dirección del salto

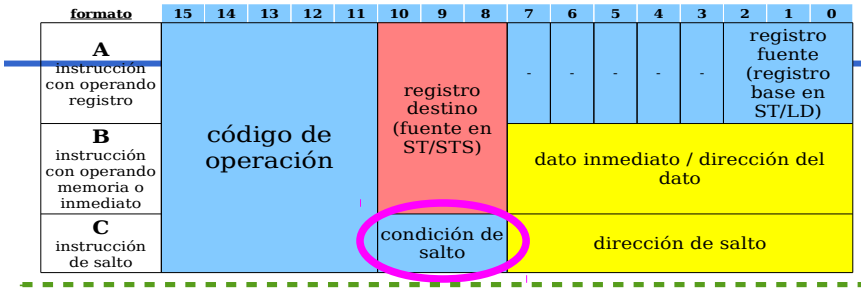


Salto condicional.
Salta si se cumple una condición codificada en la instrucción y que será evaluada por la unidad de control.



Salto condicional.

La condición para que se produzca un salto viene dada por el resultado de la operación que se realizó durante la ejecución de la instrucción anterior. ¿Fué el dato 0,positivo, desbordamiento?



Salto condicional.
Necesitamos una ALU que genere está información.
La información se guardará en los flags del registro de estado (SREG).
La complejidad de la ALU y el registro SREG son añadidos a la nueva arquitectura

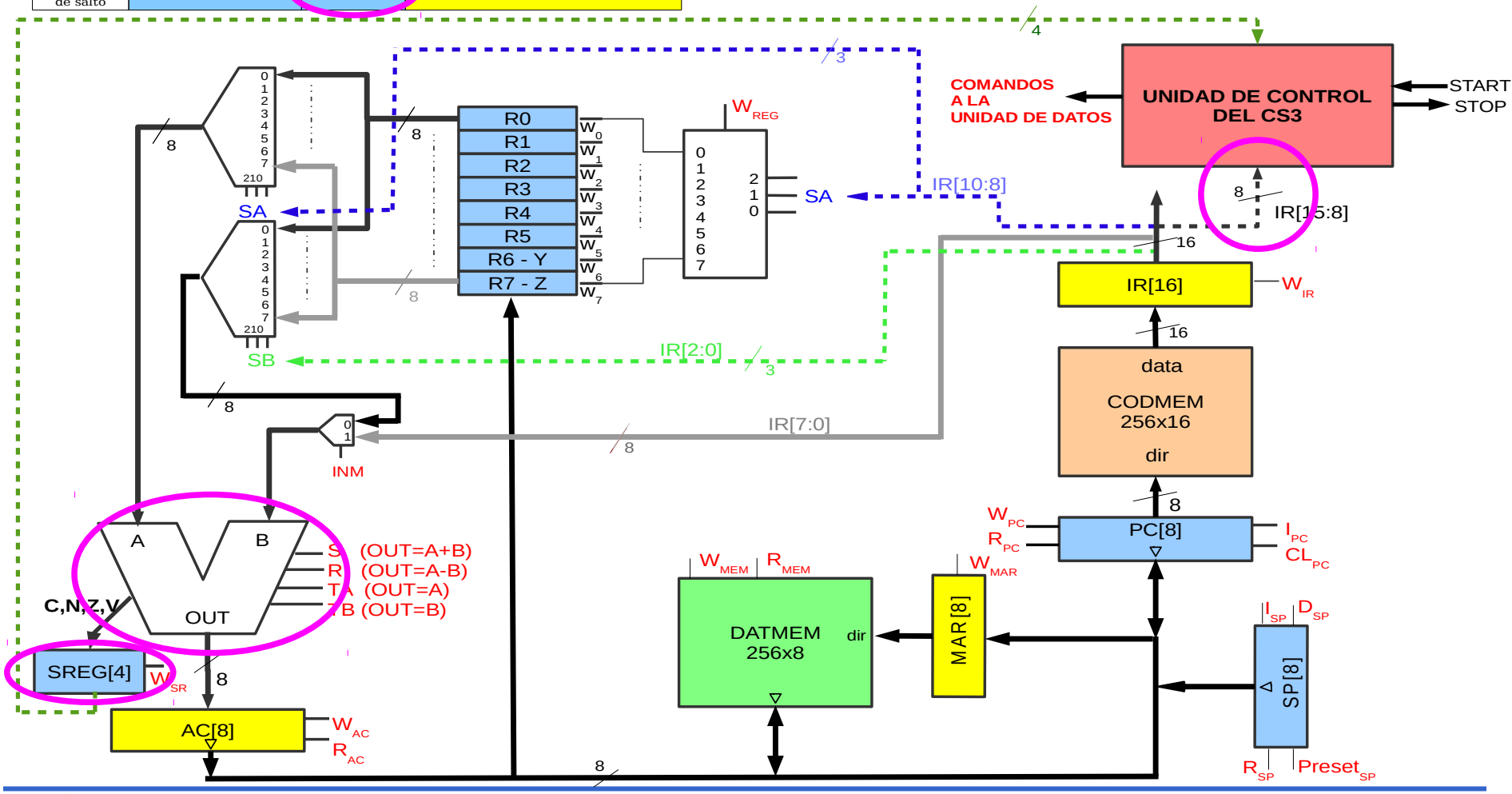
Información de lo que significan los códigos de condición

I_{10}	I_9	I_8	CONDICIÓN	nemónico(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 complemento a 2
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

El mnemónico de la instrucción de salto condicional será expuesto más adelante. Se trata ahora de entender la relación entre los bits del código de condición y el flag generado por la ALU y que se guarda en el registro de estado.

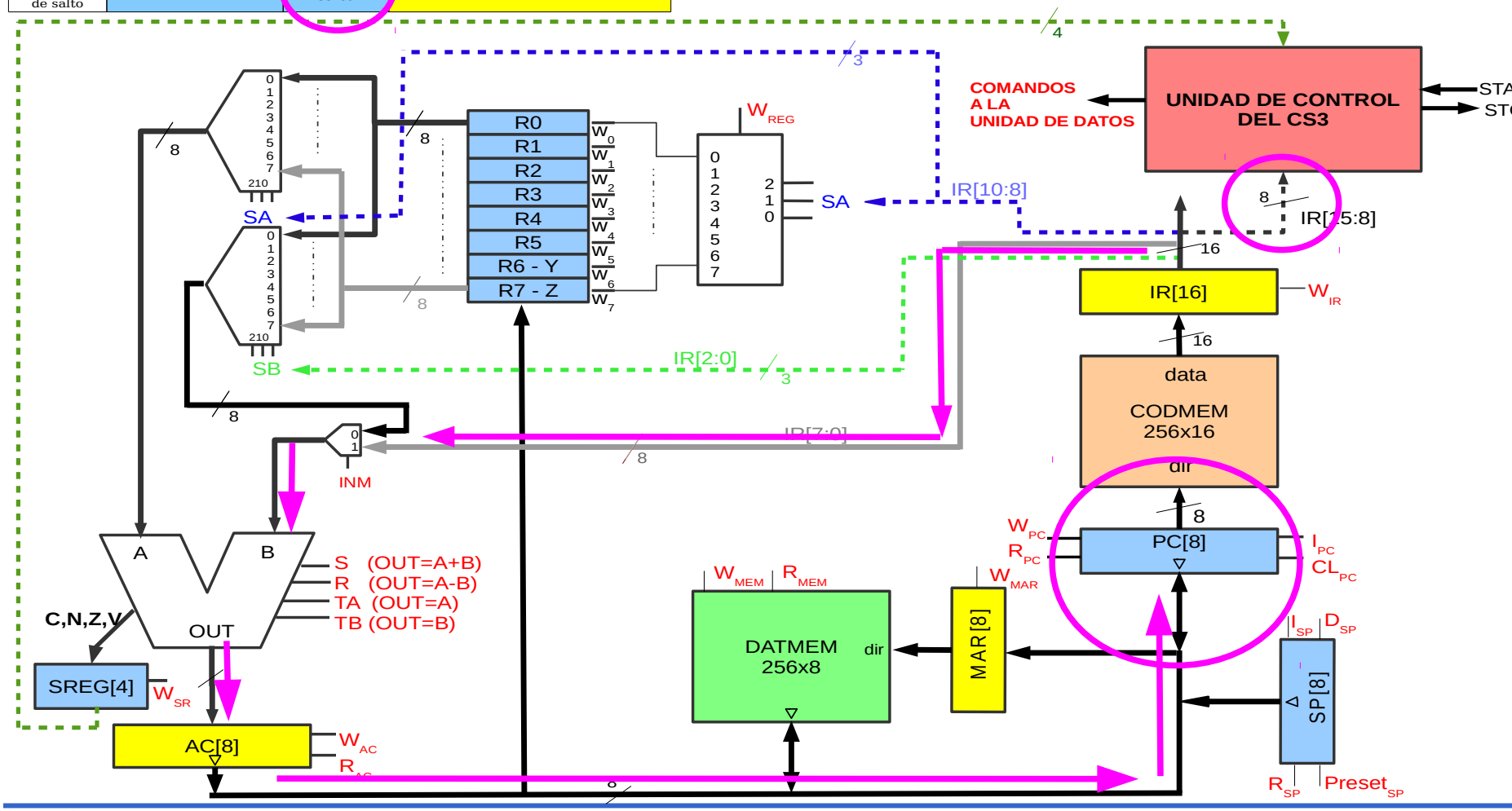
formato	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A instrucción con operando registro	código de operación					registro destino (fuente en ST/STS)			-	-	-	-	-	registro fuente (registro base en ST/LD)		
B instrucción con operando memoria o inmediato									dato inmediato / dirección del dato							
C instrucción de salto						condición de salto			dirección de salto							

Salto condicional.
No se cumple la condición: no se modifica el PC.



formato	15	14	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/STS)			-	-	-	-	-	registro fuente (registro base en ST/LD)		
B instrucción con operando memoria o inmediato									dato inmediato / dirección del dato							
C instrucción de salto						condición de salto			dirección de salto							

Salto condicional.
Se cumple la condición: el PC es modificado.



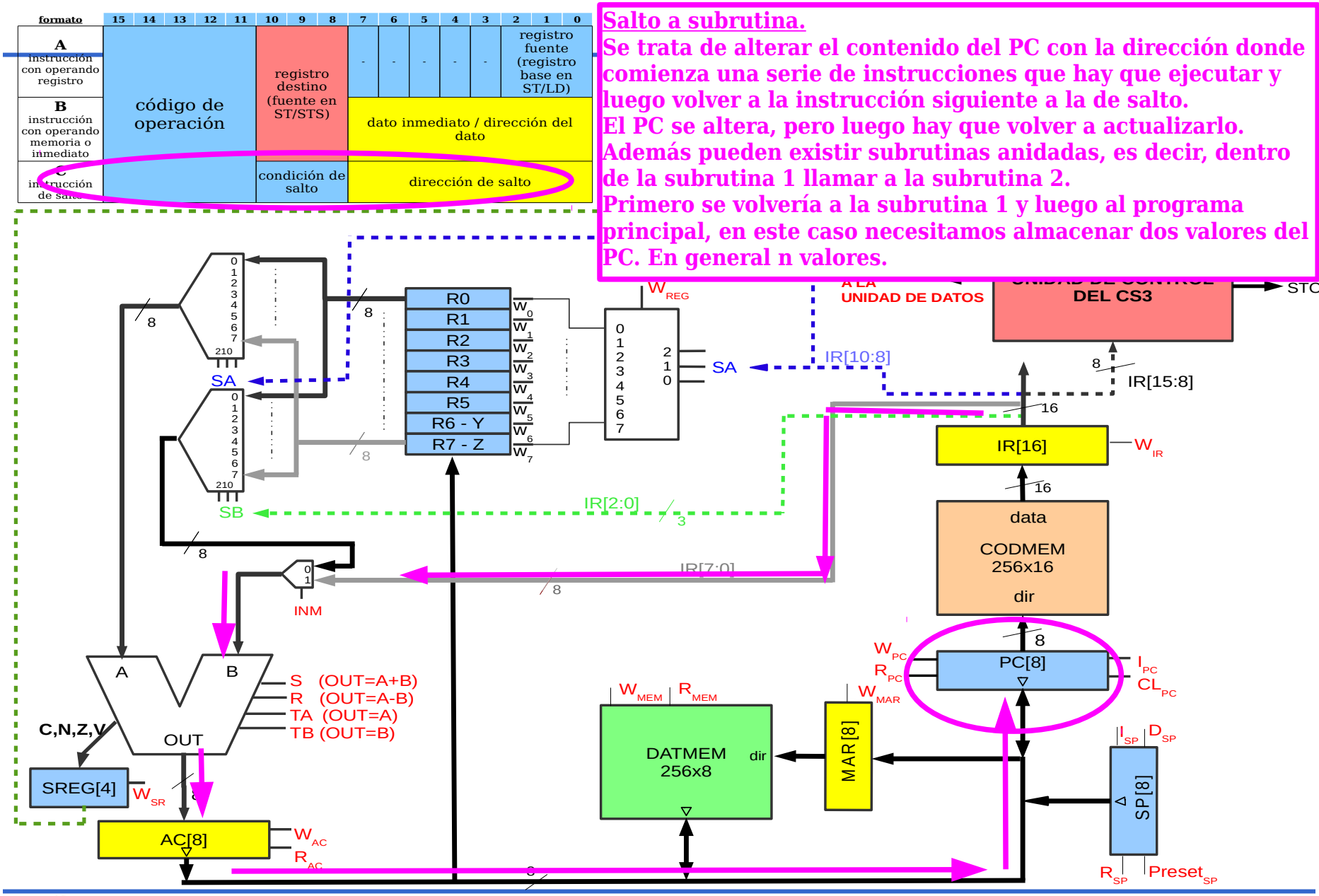
Salto a subrutina.

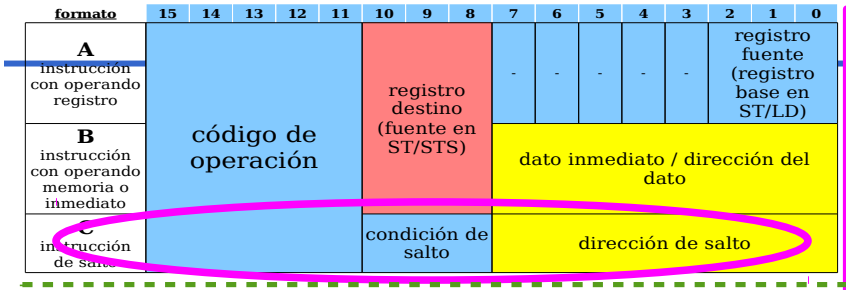
Se trata de alterar el contenido del PC con la dirección donde comienza una serie de instrucciones que hay que ejecutar y luego volver a la instrucción siguiente a la de salto.

El PC se altera, pero luego hay que volver a actualizarlo.

Además pueden existir subrutinas anidadas, es decir, dentro de la subrutina 1 llamar a la subrutina 2.

Primero se volvería a la subrutina 1 y luego al programa principal, en este caso necesitamos almacenar dos valores del PC. En general n valores.





Salto a subrutina.

Los sucesivos valores del PC se guardan en la memoria de datos del computador, pero no en cualquier sitio.

Se guardan en una zona denominada Pila, la justificación de este nombre será expuesta más adelante.

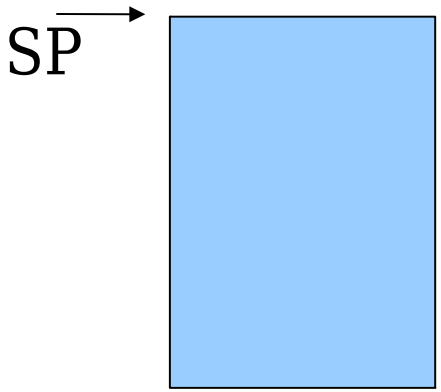
La dirección para acceder a la pila está guardada en un registro denominado puntero de pila y que hemos tenido que añadir a la nueva arquitectura CS3.

Estructura de la Pila

Antes de meter un nuevo dato el SP tiene guardada la dirección de la pila donde el dato se guardará.
Es como si se hiciera una pila de libros, el primero que quito es el último que he puesto.
El primer dato se guarda en una dirección determinada y el siguiente, en la dirección inmediatamente inferior. La pila crece hacia direcciones decrecientes de la memoria.

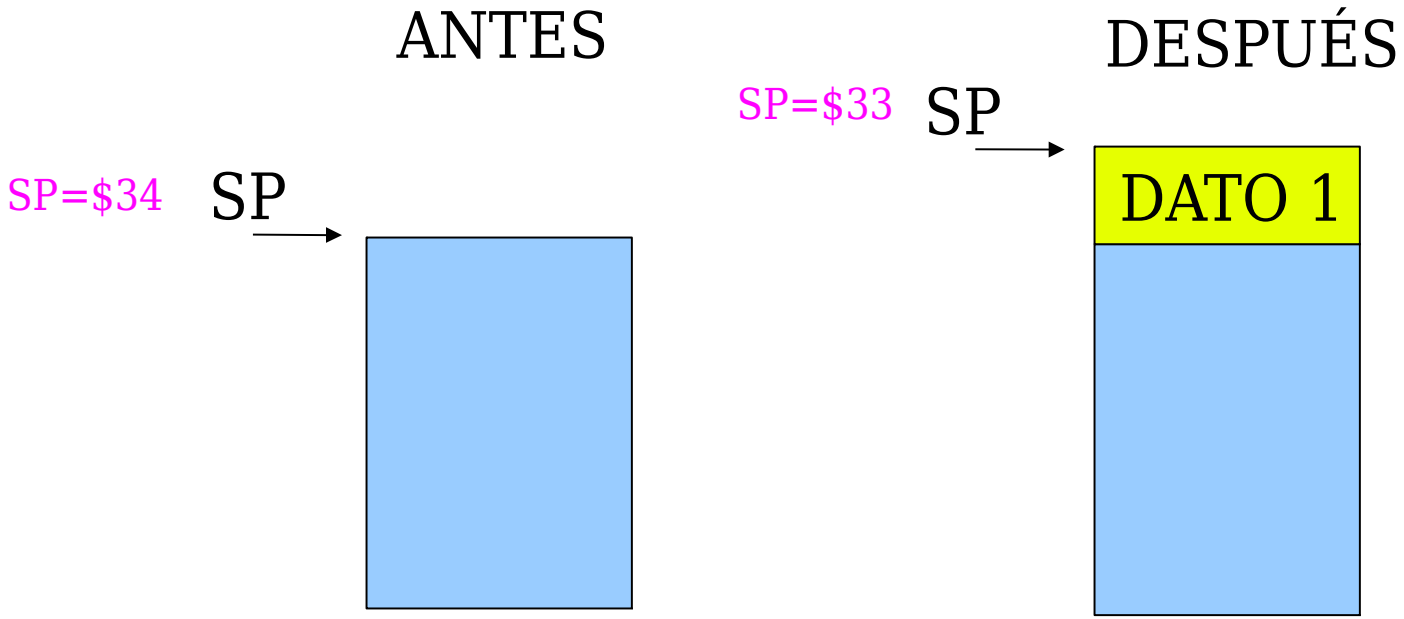
Supongamos SP=$34

Representación abstracta de la zona de la memoria de datos y los valores del SP del CS3



Estructura de la Pila

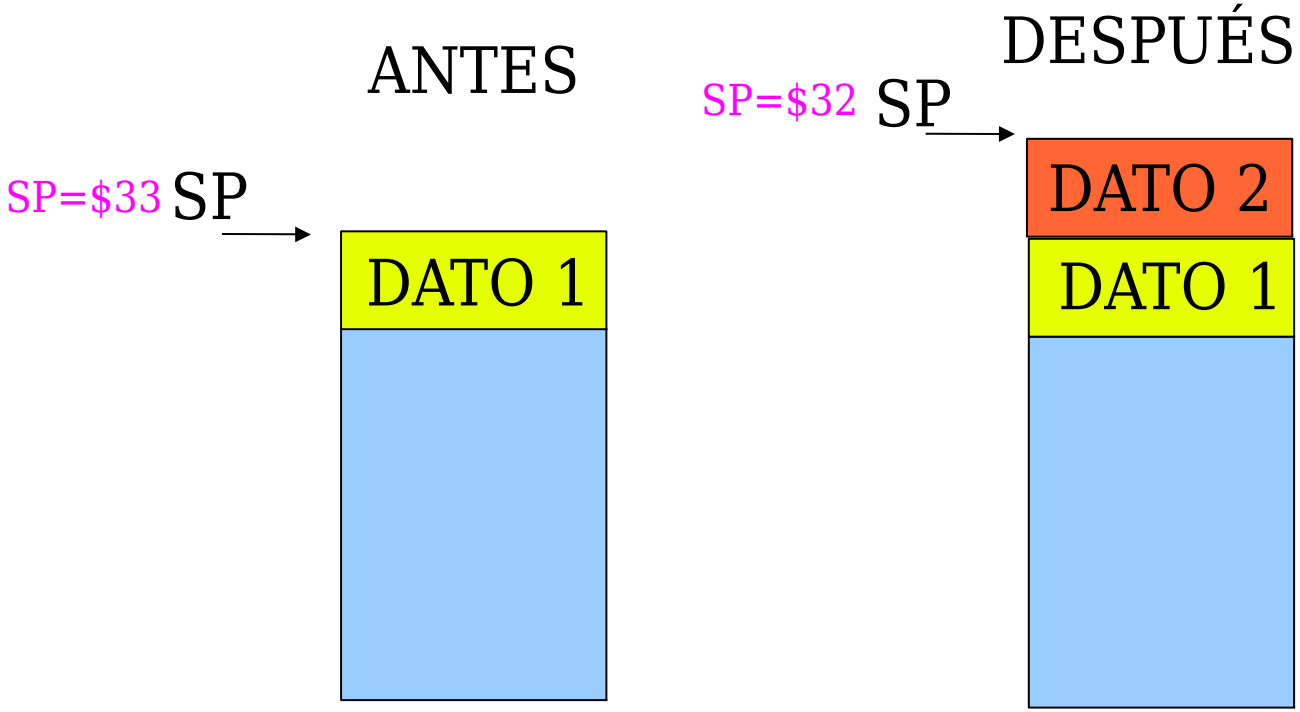
Representación abstracta de la zona de la memoria de datos y los valores del SP del CS3



Proceso de guardar datos en la pila. Operación de PUSH

Estructura de la Pila

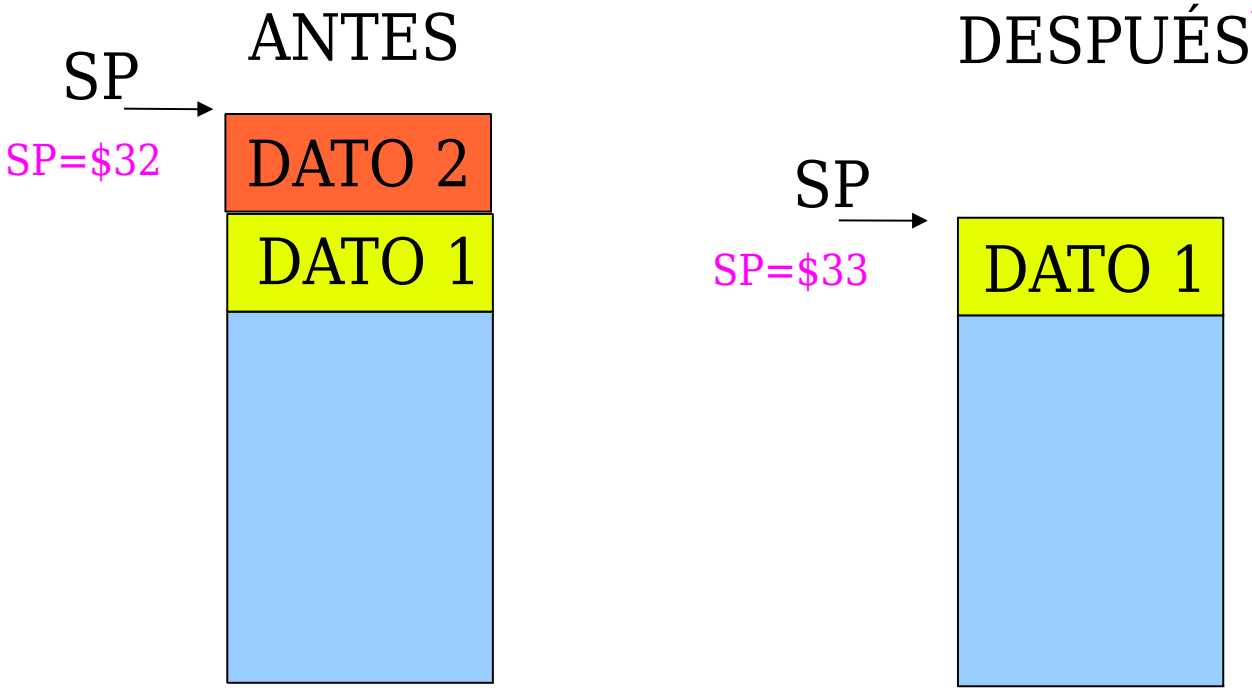
Representación abstracta de la zona de la memoria de datos y los valores del SP del CS3



Proceso de guardar datos en la pila. Operación de PUSH

Estructura de la Pila

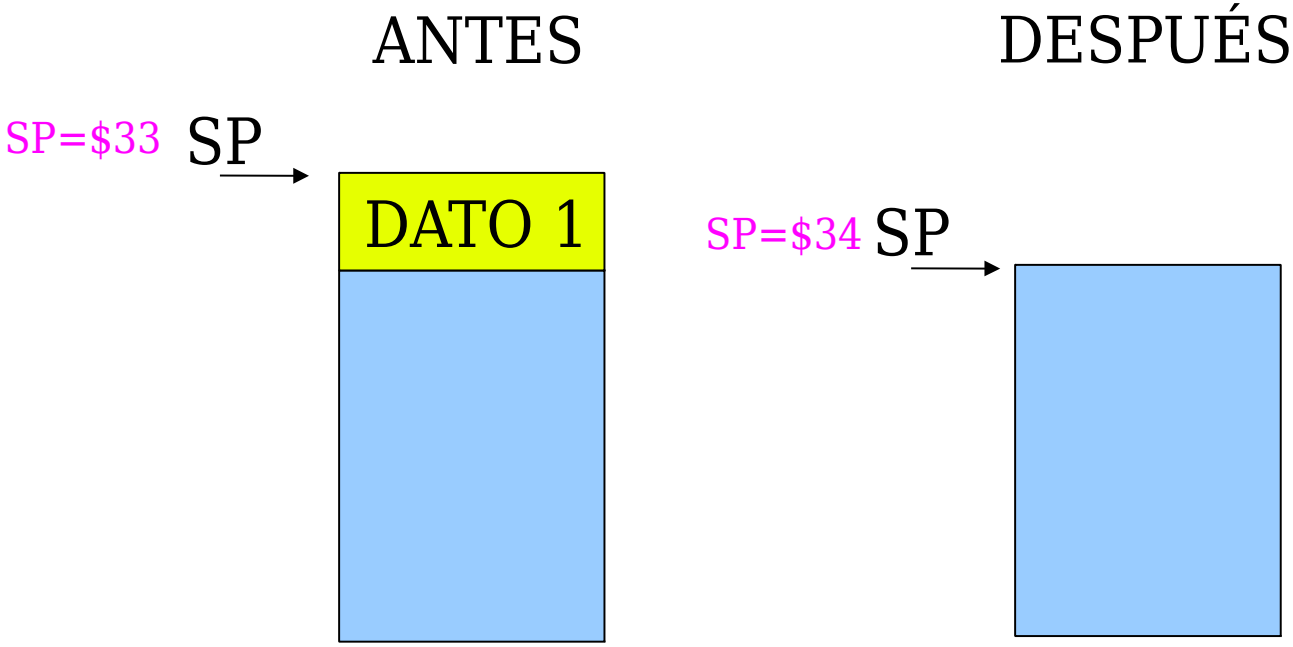
Representación abstracta de la zona de la memoria de datos y los valores del SP del CS3



Proceso de extracción de datos de la pila. Operación de PULL. El primero que se lee es el último que se escribió.

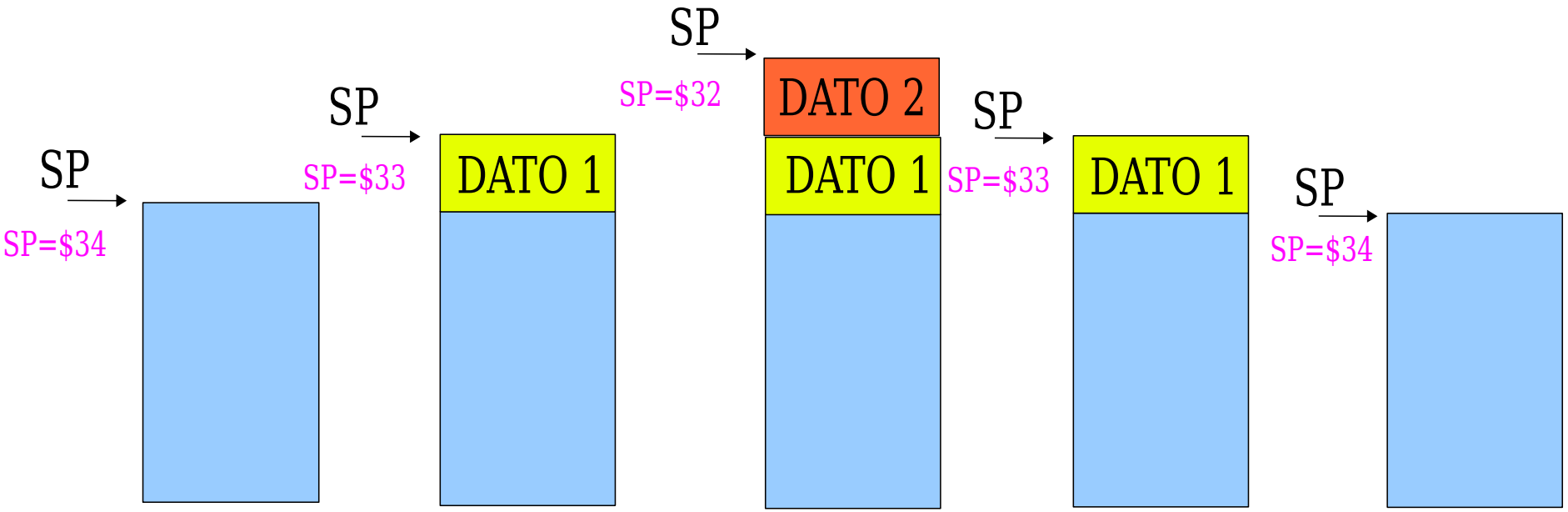
Estructura de la Pila

Representación abstracta de la zona de la memoria de datos y los valores del SP del CS3



Proceso de extracción de datos de la pila. Operación de PULL. El primero que Lee es el último que se escribió.

Estructura de la Pila

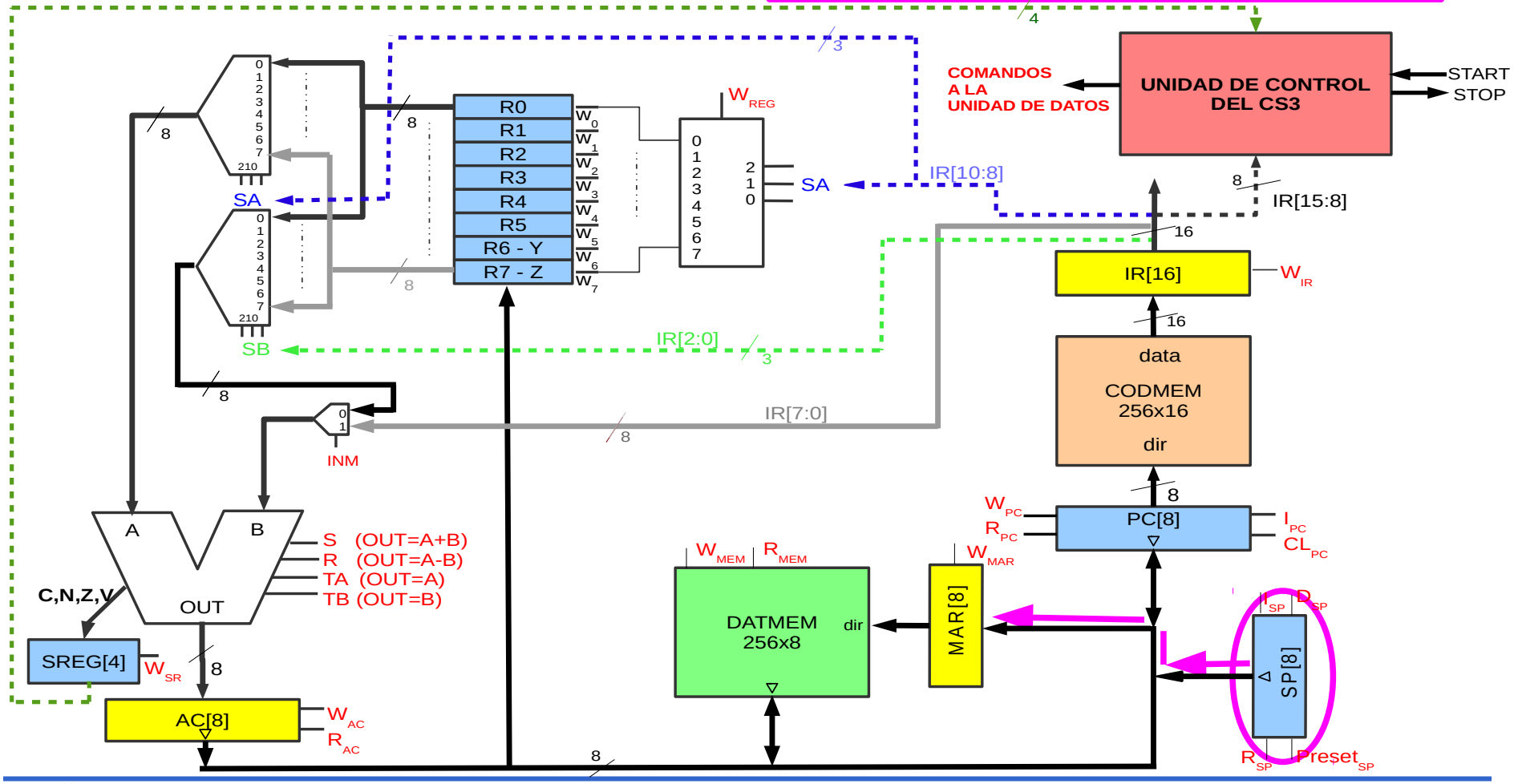


Resumen del proceso de dos operaciones de escritura en pila (PUSH) y otras dos de lectura de pila (PULL).
Los datos se leen en orden inverso a como se escribieron.

formato	15	14	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/STS)			-	-	-	-	-	registro fuente (registro base en ST/LD)		
B instrucción con operando memoria o inmediato									dato inmediato / dirección del dato							
C instrucción de salto						condición de salto			dirección de salto							

Retorno de subrutina.

Se recuperan lo valores del PC guardados para saber la dirección de salto.
La dirección de salto por lo tanto no se encuentra en el código de la instrucción.
No está asociado a ningún tipo de formato

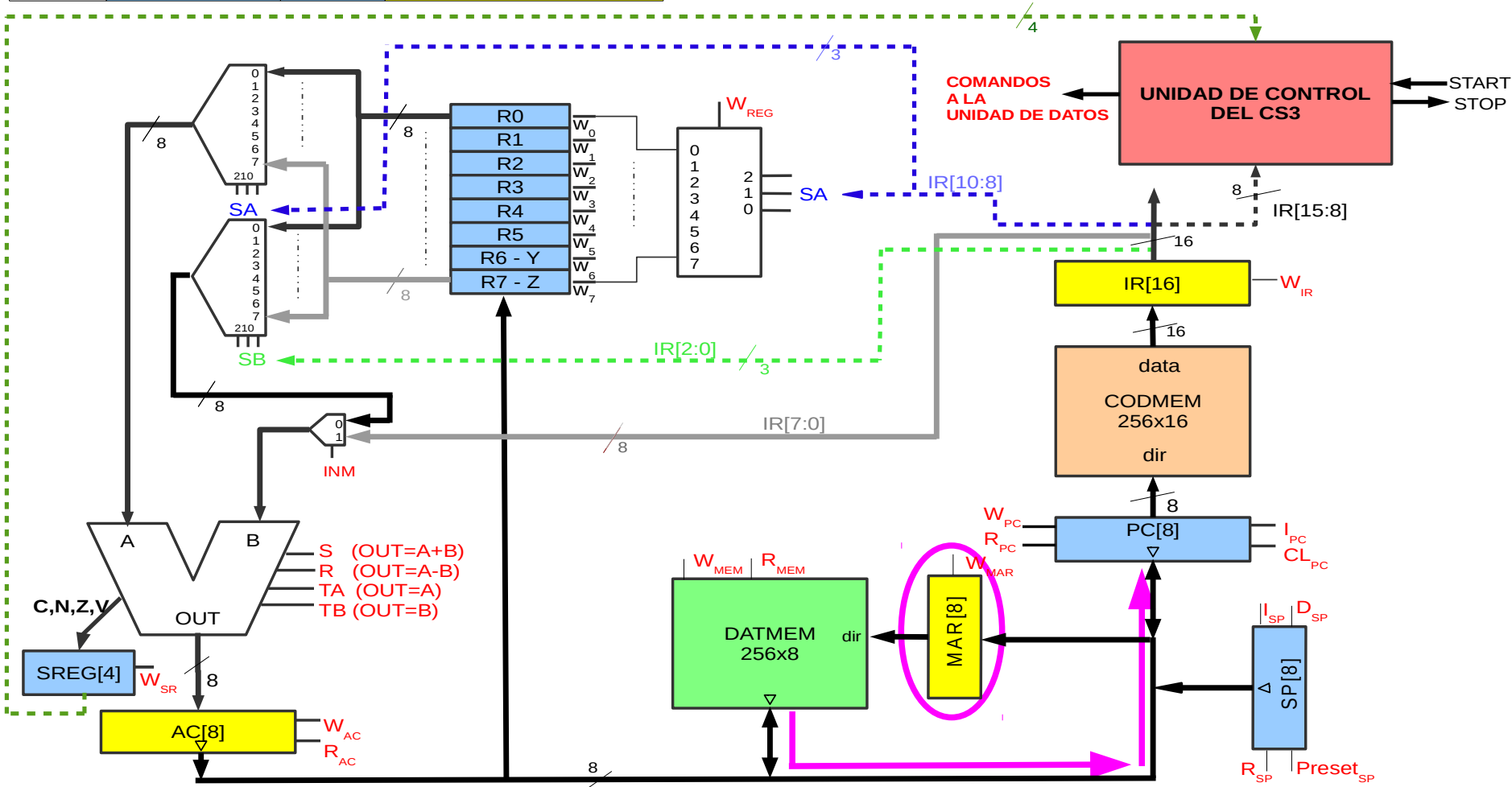


Descripción general: Incrementar SP y guardarlo en MAR
Esto permite el acceso a la dirección de la memoria de datos, cuyo contenido es la dirección de salto.

formato	15	14	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/STS)			-	-	-	-	-	registro fuente (registro base en ST/LD)		
B instrucción con operando memoria o inmediato									dato inmediato / dirección del dato							
C instrucción de salto						condición de salto			dirección de salto							

Retorno de subrutina.

Se recuperan lo valores del PC guardados para saber la dirección de salto.
La dirección de salto por lo tanto no se encuentra en el código de la instrucción.
No está asociado a ningún tipo de formato



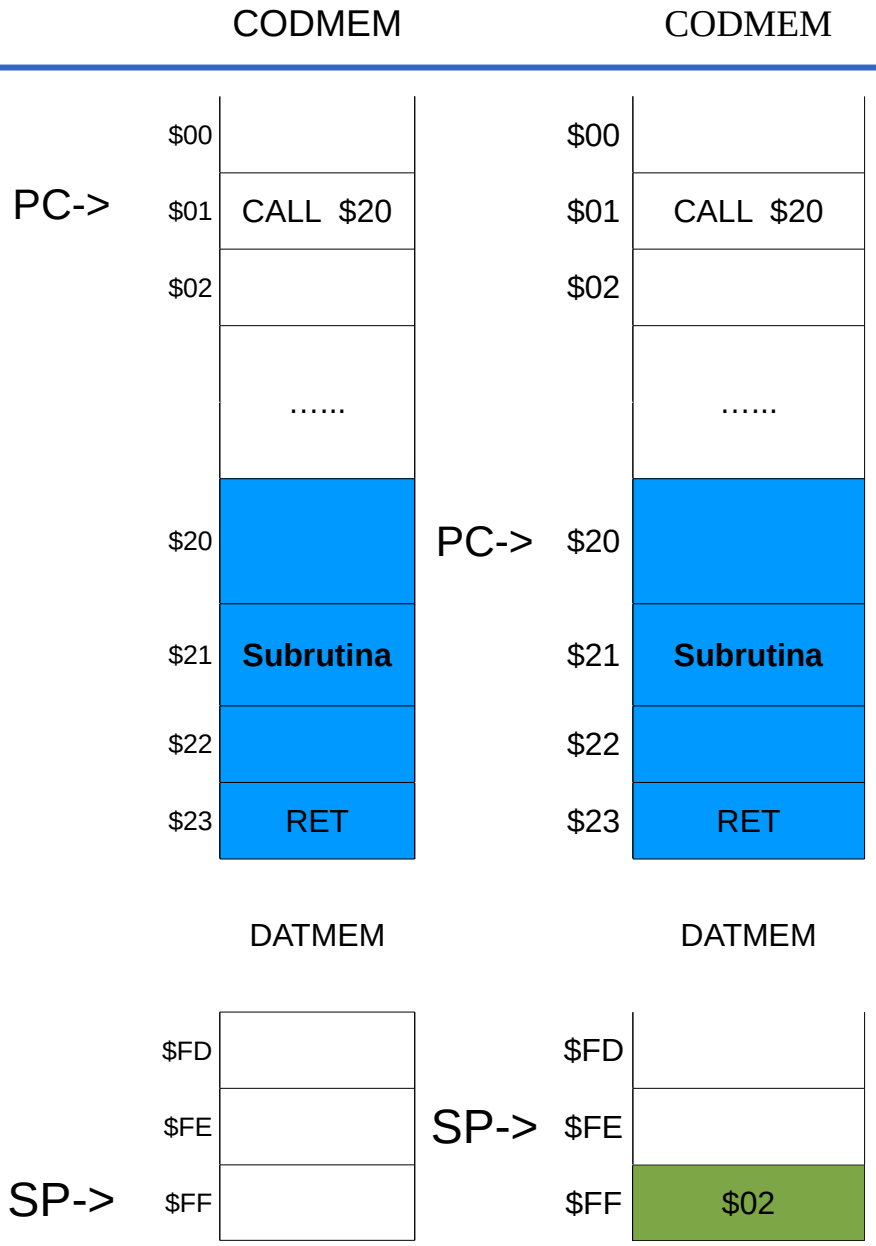
Descripción general:
Acceso a la pila recuperación del valor
Deseado del PC, vuelta a la instrucción siguiente a
La de salto a subrutina

Instrucciones de llamada y retorno de subrutina

- Una subrutina es una secuencia de instrucciones que realizan una cierta tarea a partir de unos datos de entrada, generando unos datos de salida
- Cuando un programa necesita realizar la tarea, hace una llamada a la subrutina, ésta se ejecuta y cuando termina, el programa se reanuda por donde se había quedado (retorno de subrutina).
- Una subrutina puede ser llamada todas las veces que el programa lo necesite
- Una subrutina puede llamar a su vez a otra subrutina: ejecución anidada de subrutinas
- Una subrutina puede llamarse a sí misma: subrutinas recursivas

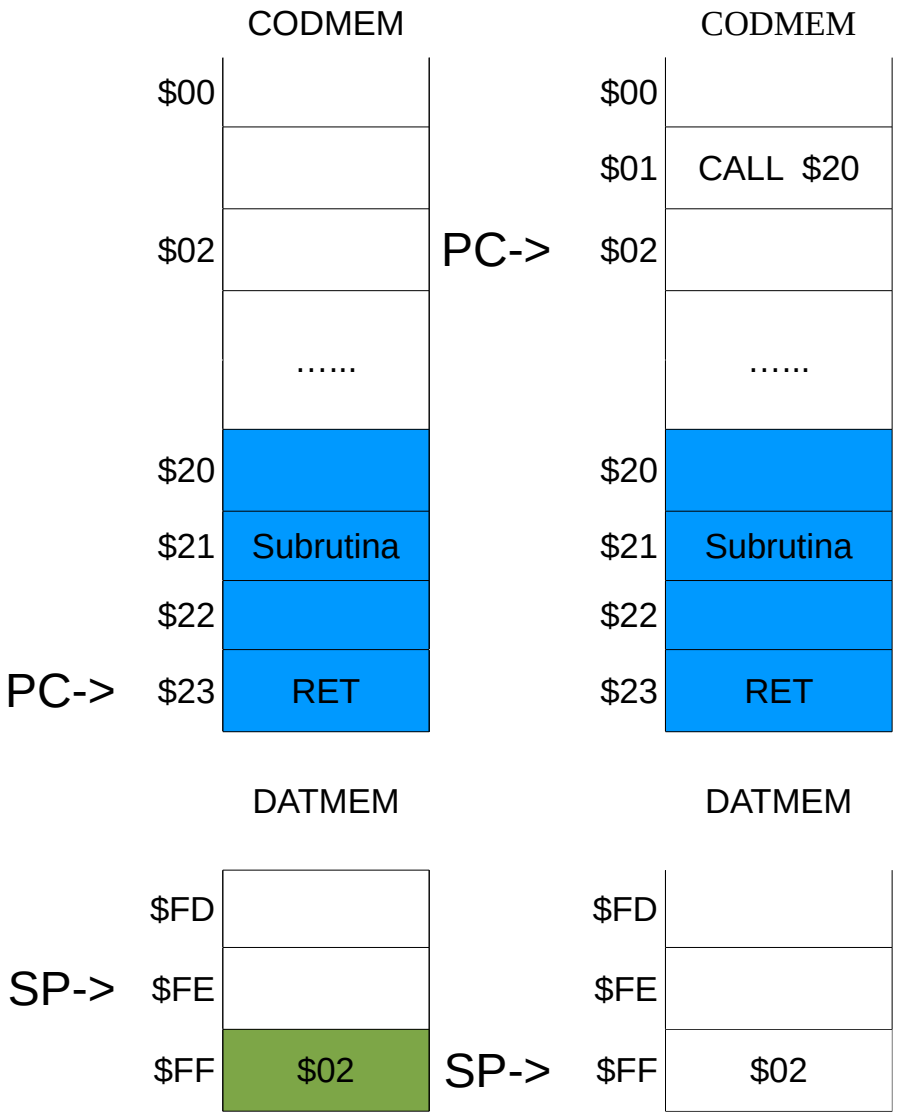
Instrucciones de llamada y retorno de subrutinas

- Situación inicial (a).
  El computador ejecuta instrucciones y llega a CALL $20.
- La ejecución de la instrucción CALL $20 hace que:
  - Se guarde en la pila el PC (que contiene $02)
  - Se “salte” a la dirección $20
- Observe el comportamiento del puntero de pila o registro SP (b).



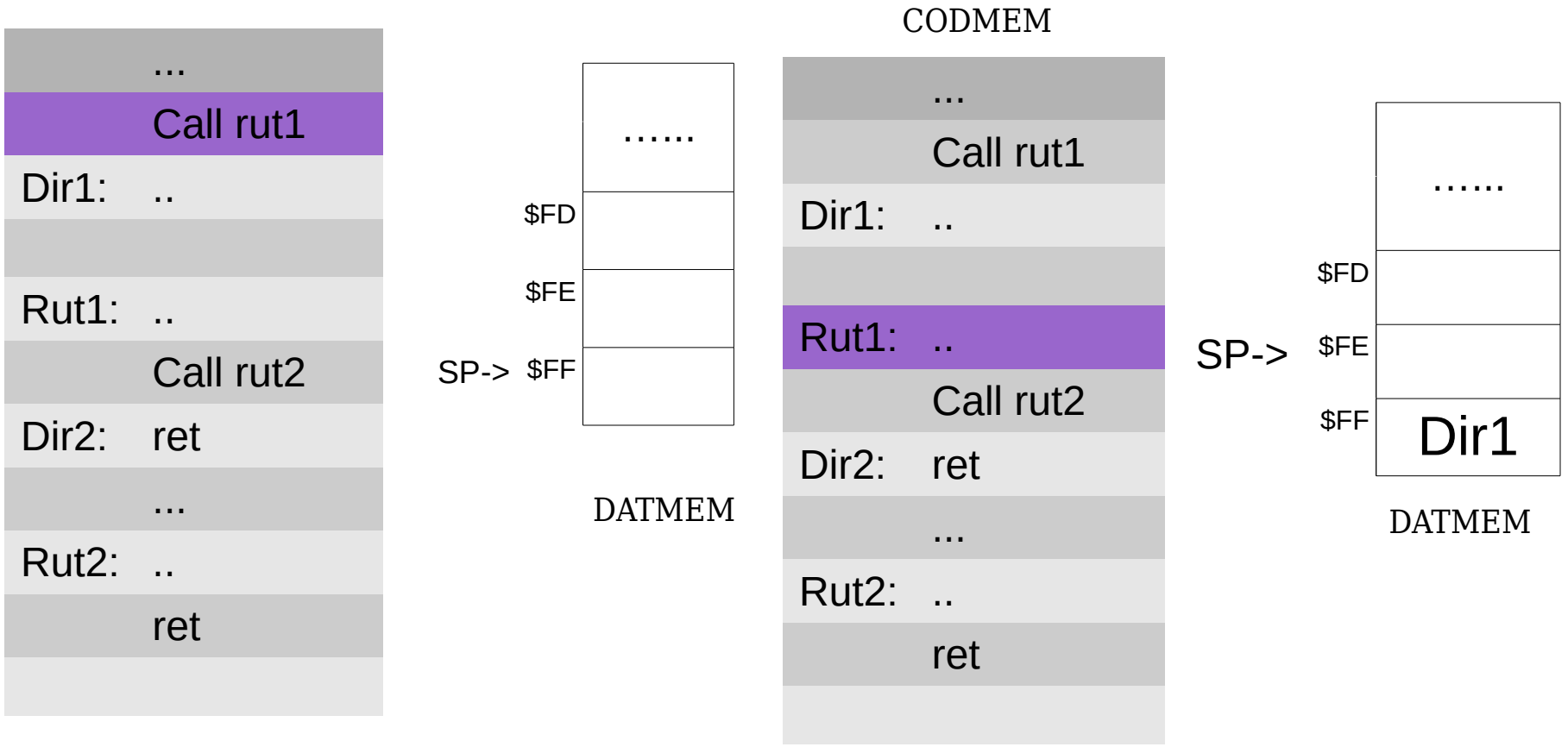
Instrucciones de llamada y retorno de subrutinas

- Se ejecutan las instrucciones de la subrutina hasta que se procede a ejecutar la última instrucción o RET (c)
- La ejecución de RET hace que se saque el dato que está en la cima de la pila y que se guarde sobre el PC (d)
- A continuación se sigue ejecutando instrucciones. En este caso a partir de la dirección $02.
- Observe que $02 sigue en la PILA pero ya no es accesible y en la próxima operación de PUSH será sobreescrito.



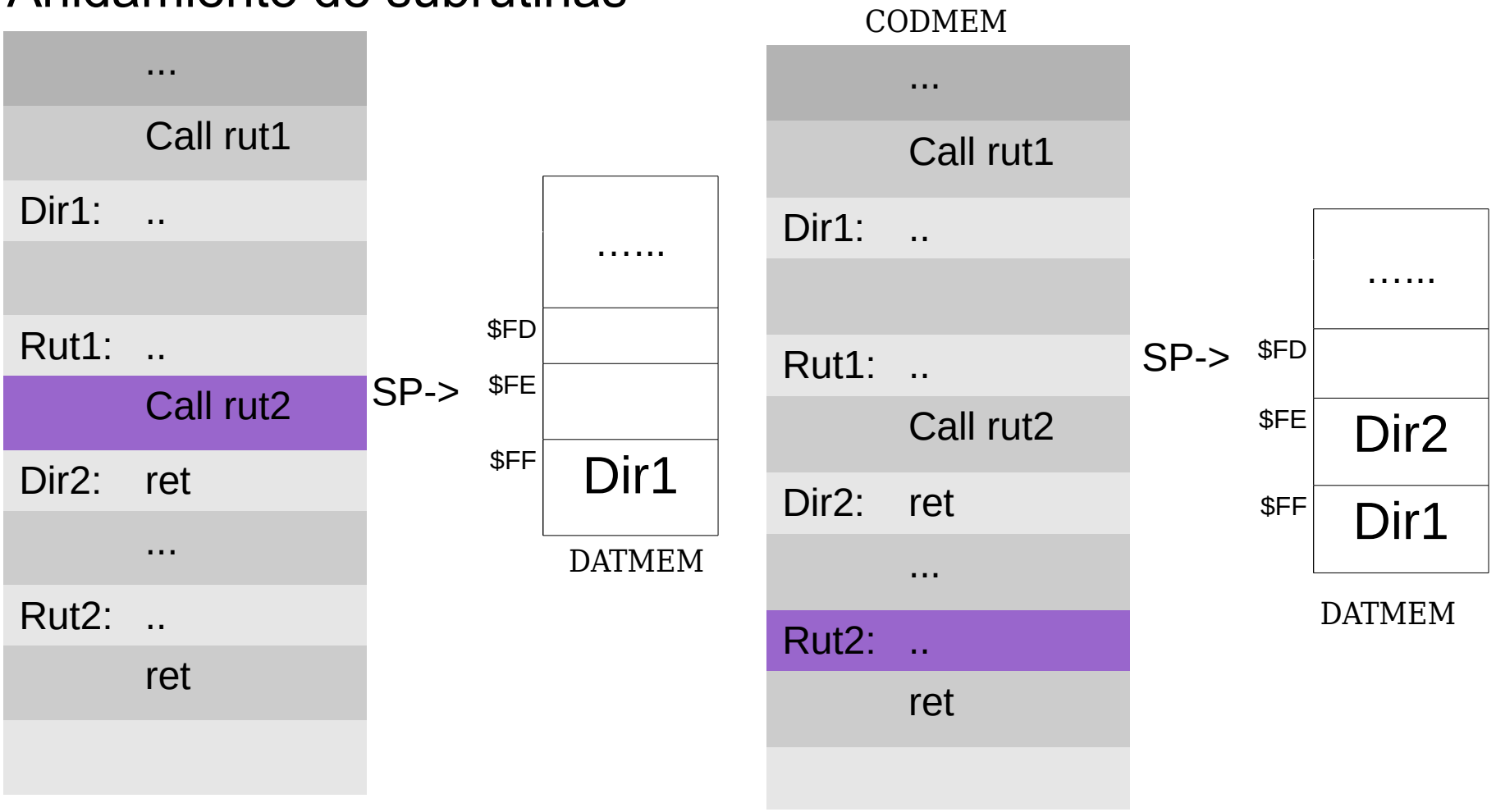
Instrucciones de llamada y retorno de subrutinas

- Anidamiento de subrutinas



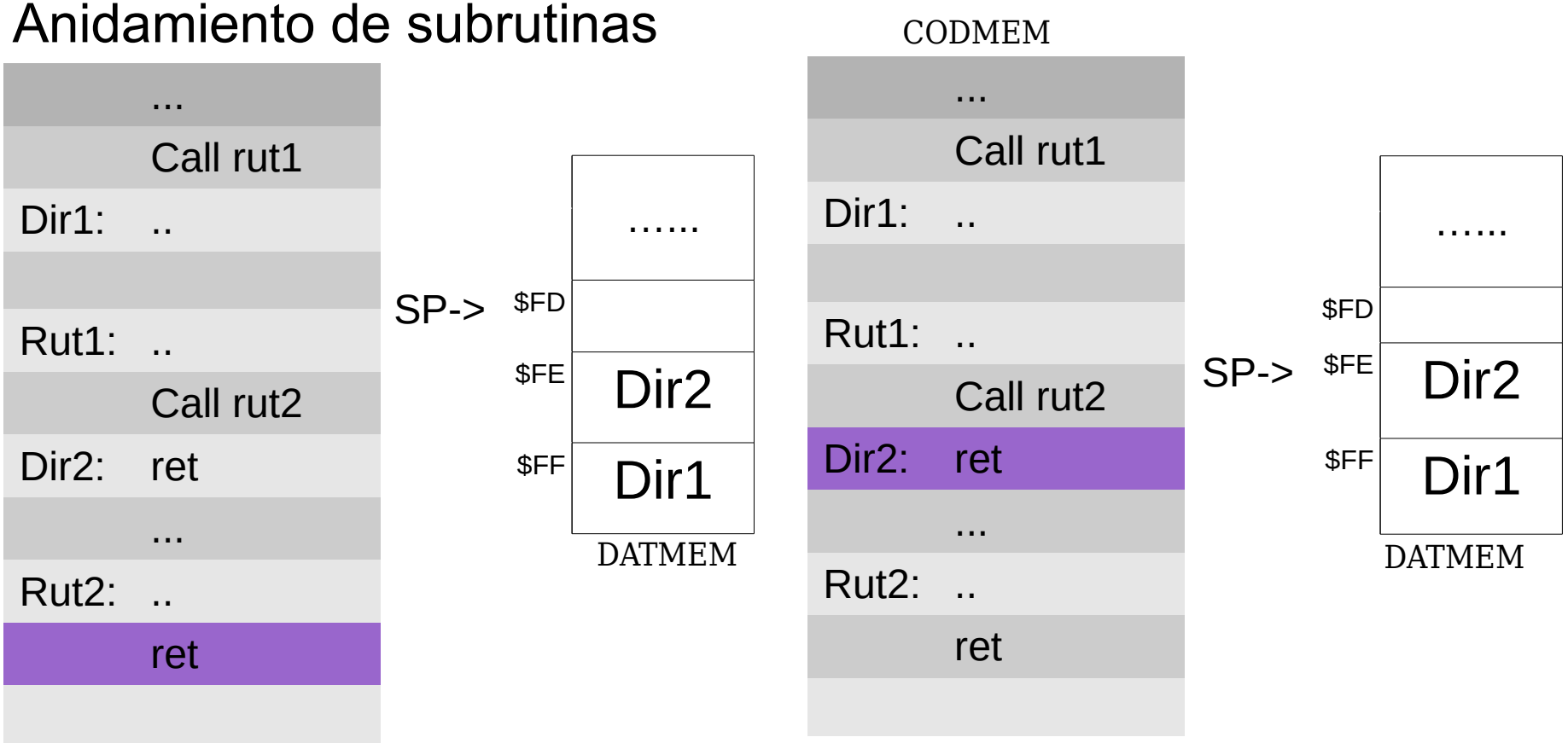
Instrucciones de llamada y retorno de subrutinas

Anidamiento de subrutinas



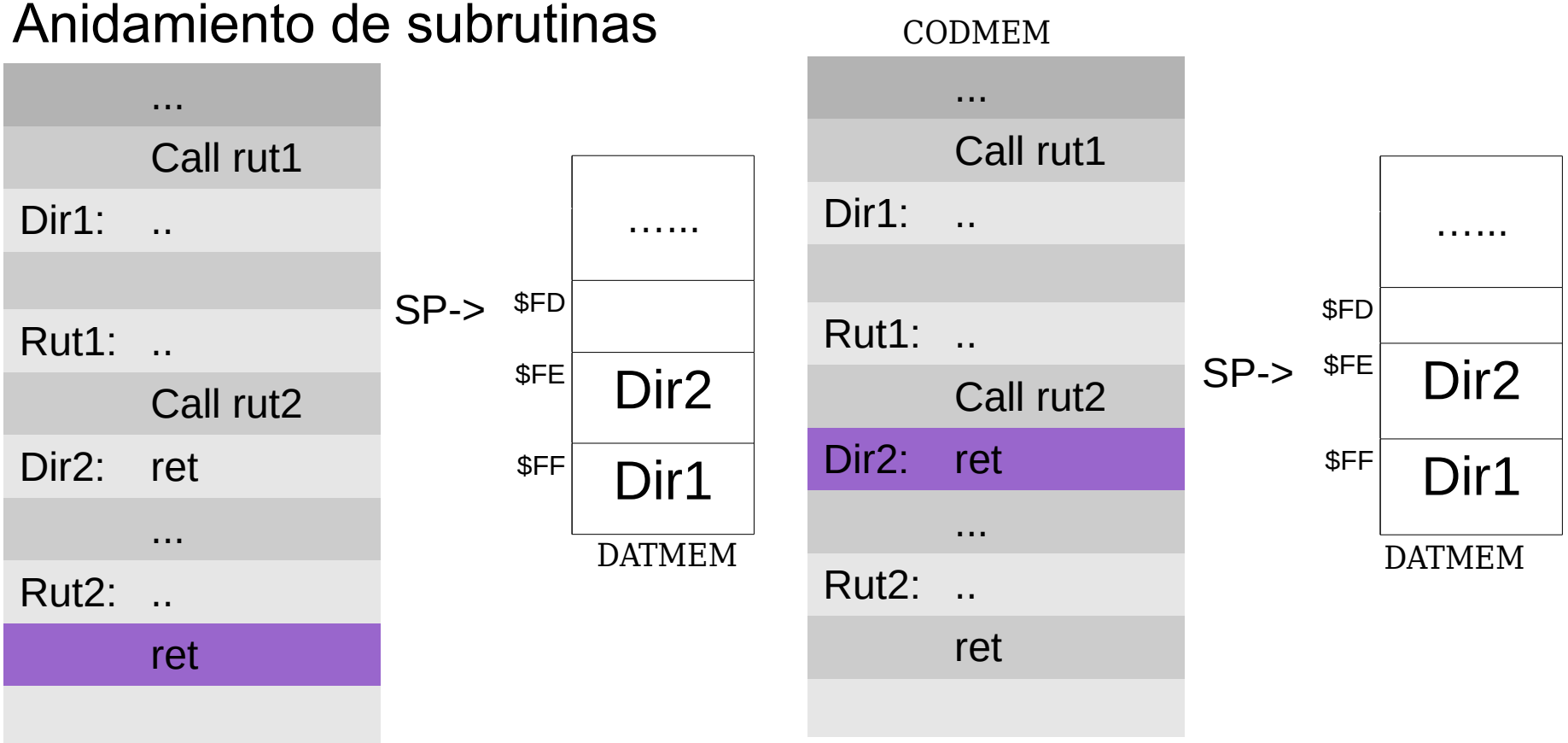
Instrucciones de llamada y retorno de subrutinas

Anidamiento de subrutinas



Instrucciones de llamada y retorno de subrutinas

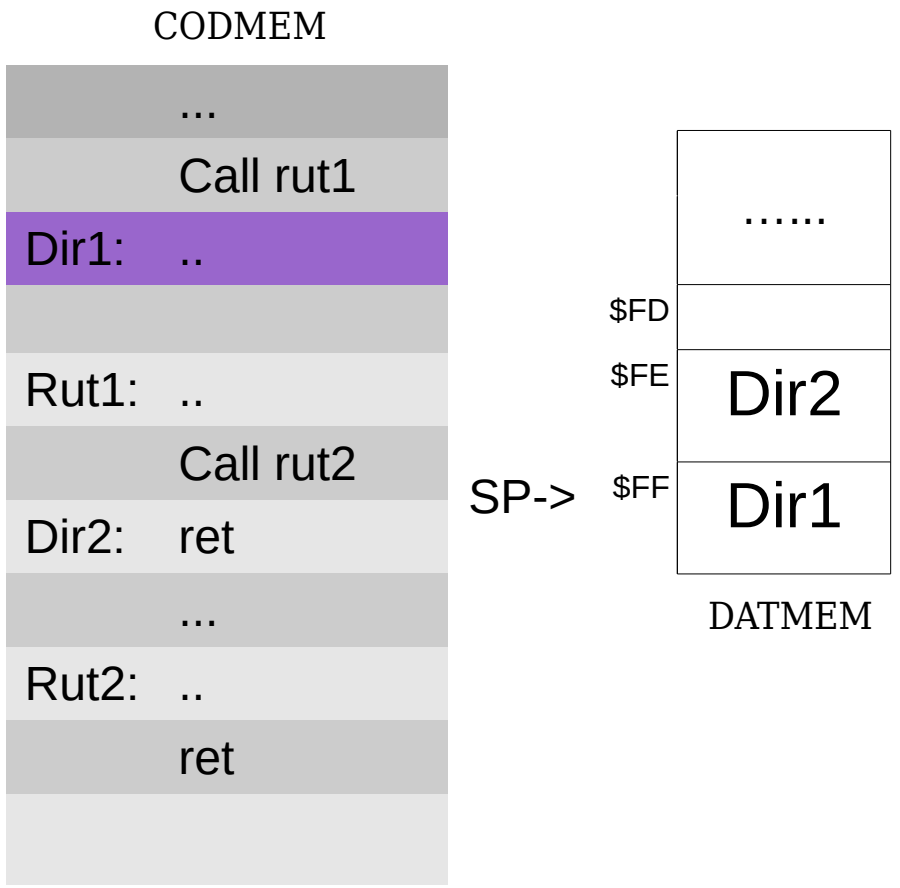
Anidamiento de subrutinas



Instrucciones de llamada y retorno de subrutinas

Anidamiento de subrutinas

- Observe que el orden en el que se deben extraer los contenidos del PC de la PILA debe ser en sentido inverso a como éstos se introdujeron en ella. (LIFO)



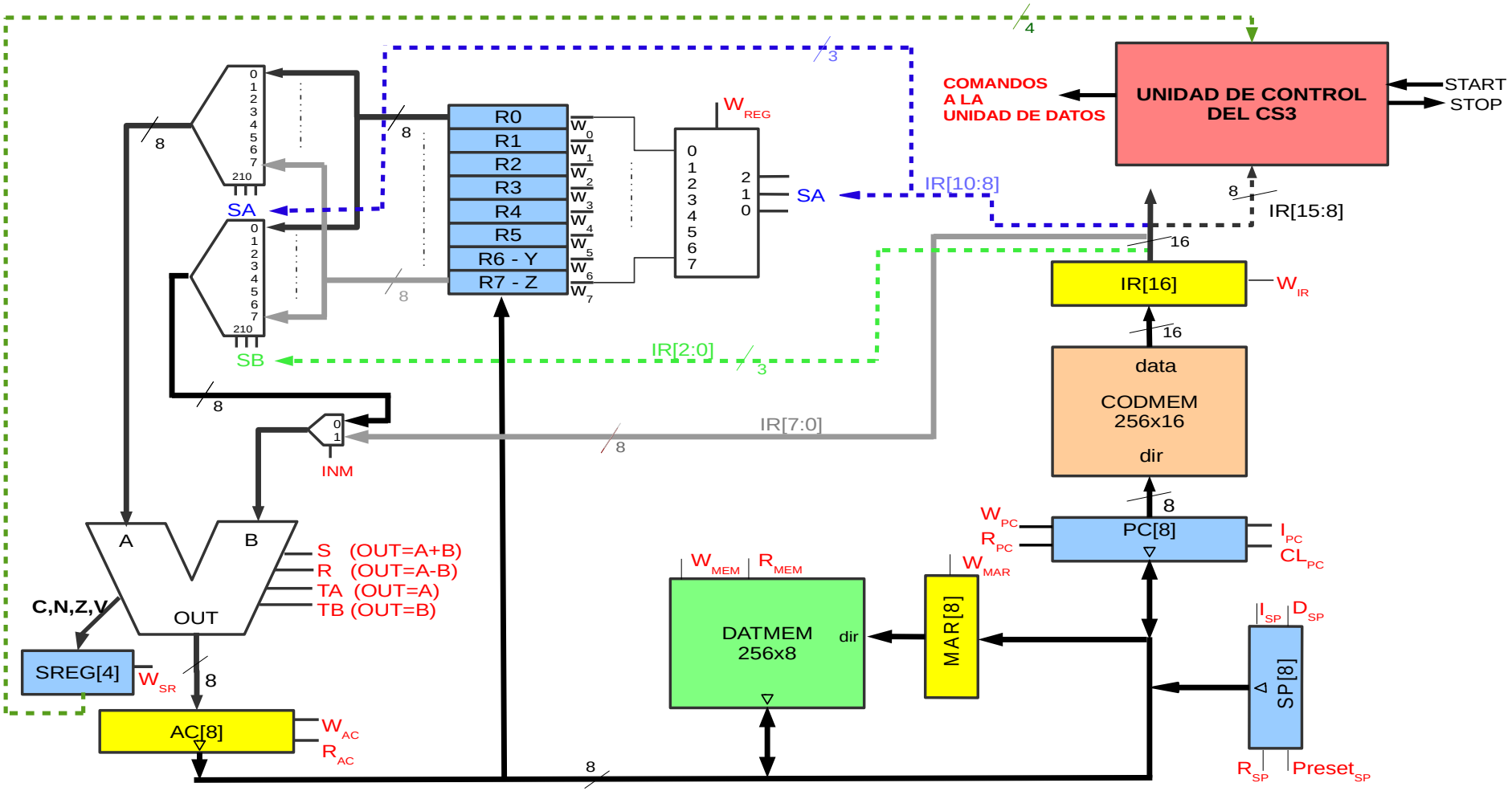
Registros visibles de la nueva arquitectura

- **Todos los registros visibles son de 8 bits**
- **Los de propósito general se etiquetan R0, R1, R2, R3, R4, R5, R6 y R7.**
- **Los de propósito específico son los siguientes:**
  - **PC o contador de programa:** 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.
  - **SREG o registro de estado:** Sus bits útiles se etiquetan Z (cero), V (desbordamiento), N (negativo) y C (cero). Indican, respectivamente, si la última instrucción aritmética/lógica generó un resultado con todos los bits a cero, un resultado no representable en complemento a 2, un resultado que es negativo al interpretarlo en complemento a 2 o si provocó carry/borrow.
  - **SP o puntero de pila:** Sirve para implementar la estructura de pila en memoria. Codifica la dirección de la primera posición libre. El puntero de pila se va decrementando a medida que se apilan datos y se incrementa al desapilarlos. Estos incrementos y decrementos se hacen de forma automatizada al realizar llamadas y retornos de subrutina.

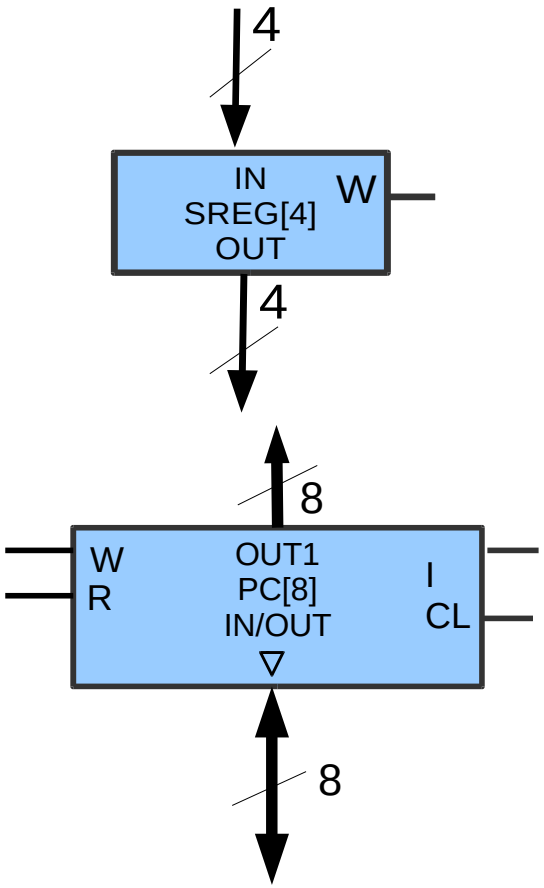
Registros ocultos

- **IR:** registro de 16 bits que sirve para almacenar la instrucción que está siendo ejecutada.
- **MAR:** Registro de 8 bits que direcciona la memoria de datos.
- **AC:** Registro de 8 bits para almacenar temporalmente el resultado de la ALU

Implementación del CS3



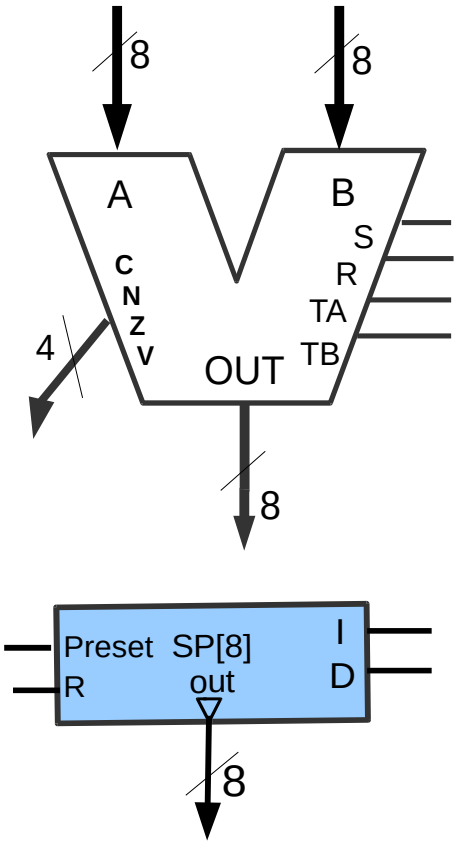
Descripción RT de los nuevos componentes



W	operación	OUT=
0	SREG ← SREG	SREG
1	SP ← IN	SREG

CL	W	R	I	operación	OUT1=	IN/OUT=
1	x	x	x	PC ← 0	PC	HI
0	1	X	x	PC ← IN/OUT	PC	DATO
0	0	1	X	PC ← PC	PC	PC
0	0	0	1	PC ← PC+1	PC	HI
0	0	0	0	PC ← PC	PC	HI

Descripción RT de los nuevos componentes



S	R	TA	TB	operación	C	N	Z	V
1	0	0	0	A+B	carry(A+B)	OUT_7	nor(out)	overflow(A+B)
0	1	0	0	A-B	borrow(A-B)	OUT_7	nor(out)	overflow(A-B)
0	0	1	0	A	-	-	-	-
0	0	0	1	B	-	-	-	-
Otras				prohibidas				

Preset	I	D	operación
0	0	0	SP ← SP
1	x	x	SP ← $FF
0	1	x	SP ← SP+1
0	0	1	SP ← SP-1

R	OUT=
0	HI
1	SP

Formatos:
Hay tres y comparten campos

<table>
<tr><th>formato</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/STS)</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td colspan="3">registro fuente (registro base en ST/LD)</td></tr>
<tr><td>B
instrucción con operando memoria o inmediato</td><td colspan="9">dato inmediato / dirección del dato</td></tr>
<tr><td>C
instrucción de salto</td><td colspan="3">condición de salto</td><td colspan="9">dirección de salto</td></tr>
</table>

Tabla de instrucciones

Asignación de códigos de operación: Facilita la decodificación

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 YoZ,Rfuente	MEM[YoZ]←Rfuente	----
0	0	0	0	1	LD	A	memoria	LD Rdestino,YoZ	Rdestino←MEM[YoZ]	----
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	Rdestino←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	1	0	SUB	A	aritmético/lógica	SUB Rdestino,Rfuente	Rdestino←Rdestino-Rfuente	****
0	1	0	1	1	CP	A	estado	CP Rdestino,Rfuente	Rdestino-Rfuente	****
0	1	1	1	1	MOV	A	movimiento de datos	MOV Rdestino,Rfuente	Rdestino←Rfuente	----
1	0	1	1	1	STOP	-	especial	STOP	lleva el procesador a espera	----
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	Rdestino-dato	****
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”

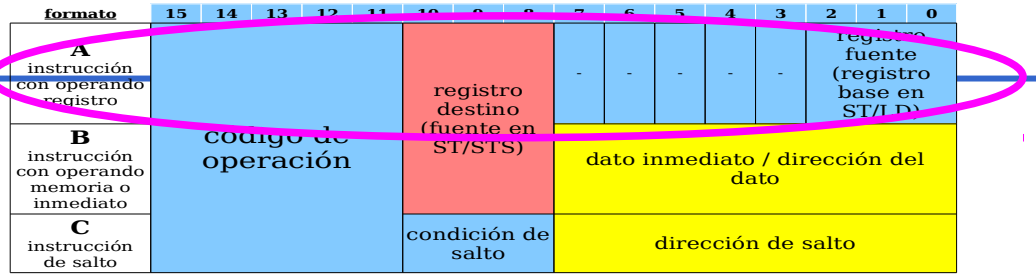
Códigos de condición de la instrucción de bifurcación condicional: BR*xx*

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

I_{10}	I_9	I_8	CONDICIÓN	nemónico(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 complemento a 2
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

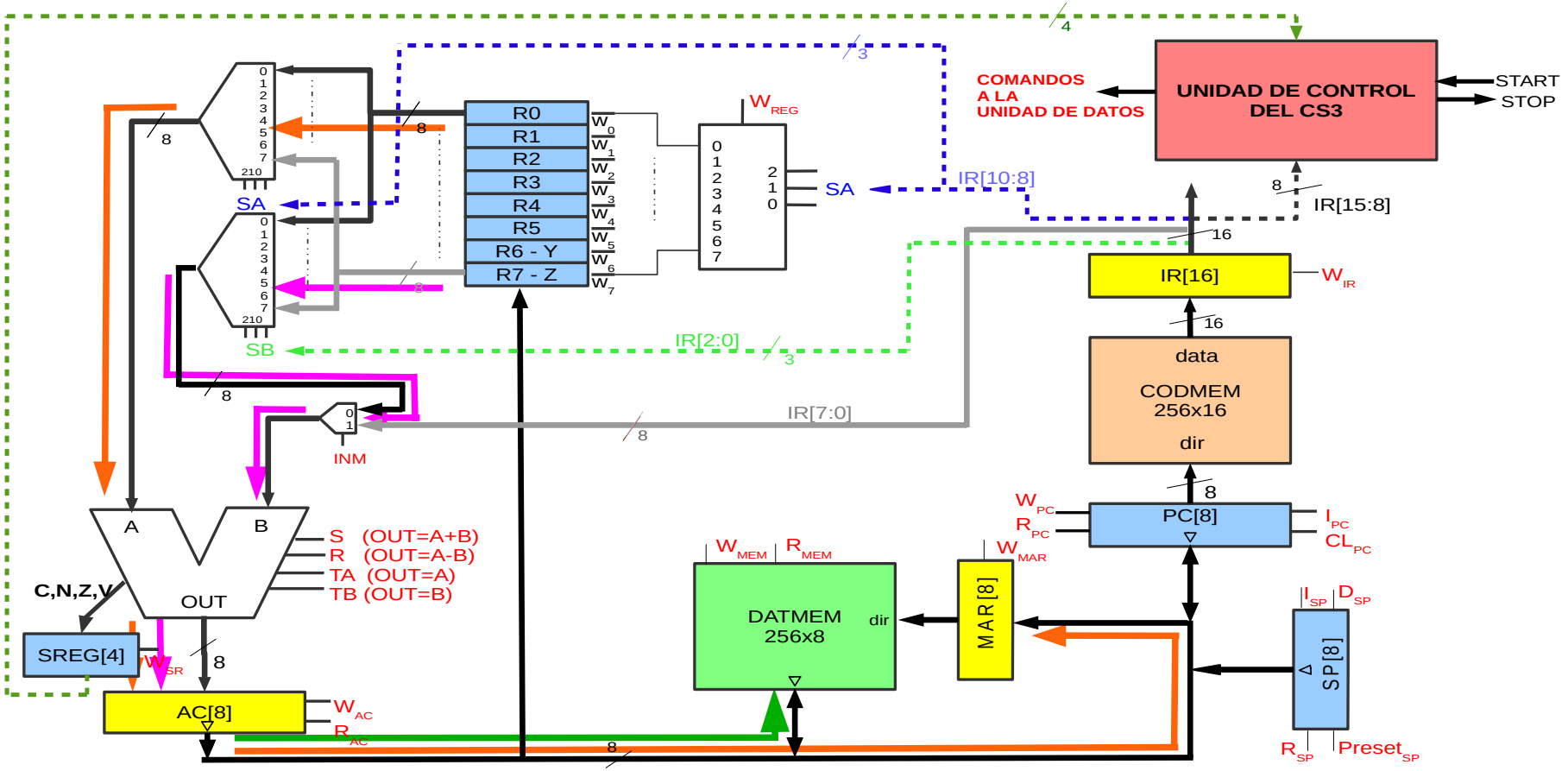
BRZS, BREQ, BRCS, BRLO, BRVS, BRLT

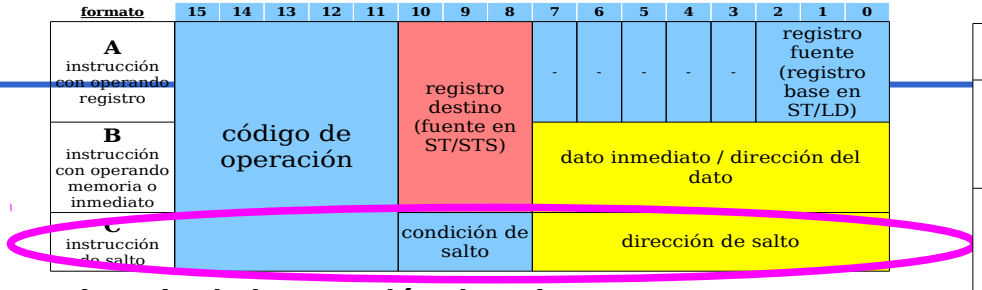
Descomposición de las instrucciones en microoperaciones para algunos casos



CICLO	MEM(YoZ) ← Rf	
1	AC ← R(IR_{2-0})	W_{AC}, TB
2	MAR ← AC AC ← R(IR_{10-8})	W_{MAR}, R_{AC} W_{AC}, TA
3	DATMEM(MAR) ← AC	WMEM, RAC

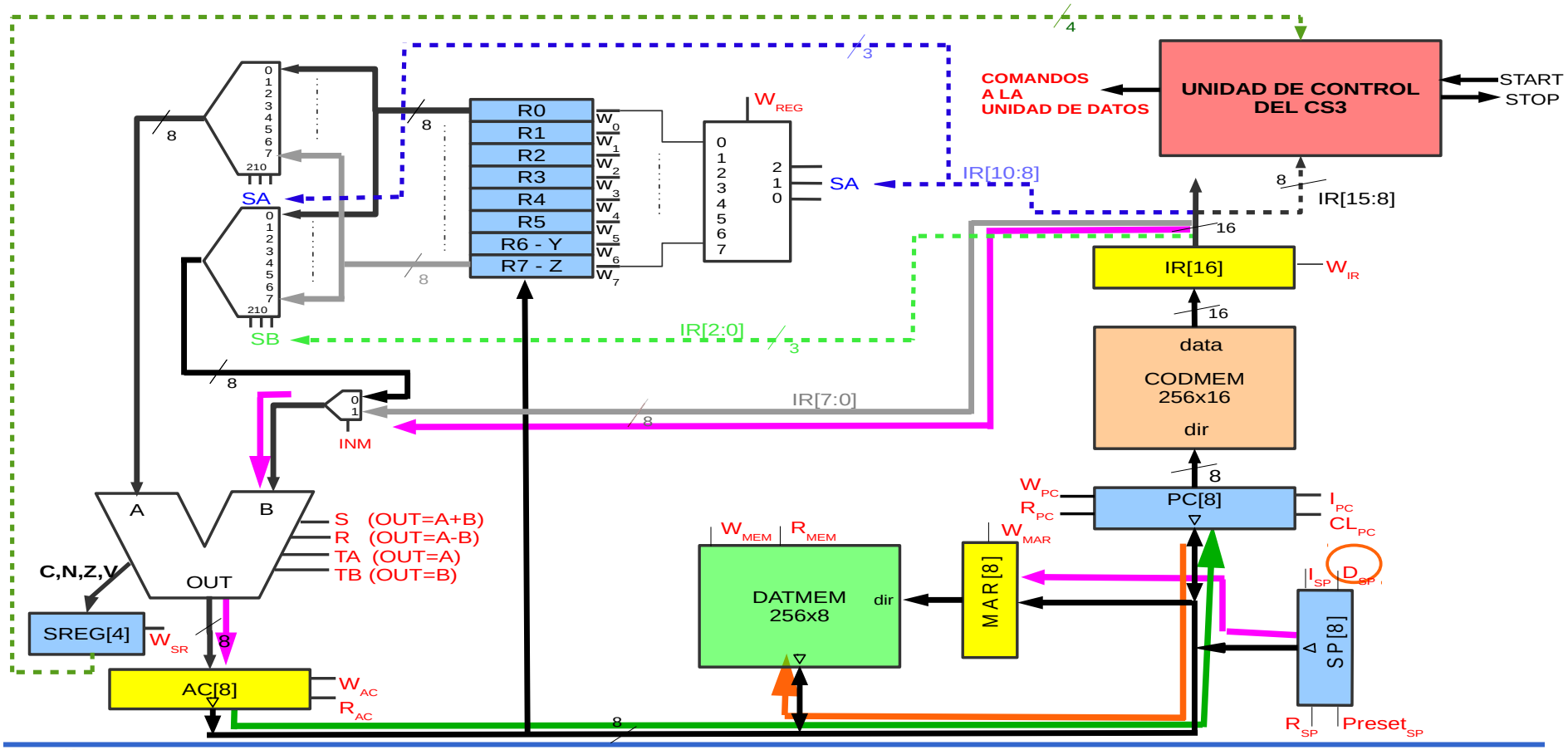
Ejemplo de instrucción de acceso a memoria para guardar dato en ella.
Direccionamiento indirecto de registro ST YoZ,Ri





CICLO	MEM(SP) ← PC, SP ← SP-1, PC ← dirección	
1	MAR ← SP AC ← IR[7:0]	W_{MAR}, R_{SP} W_{AC}, TB,INM
2	DATMEM(MAR) ← PC SP ← SP-1	W_{MEM}, R_{AC} D_{SP}
3	PC ← AC	W_{PC}, R_{AC}

Ejemplo de instrucción de salto
Direccionamiento absoluto CALL dirección

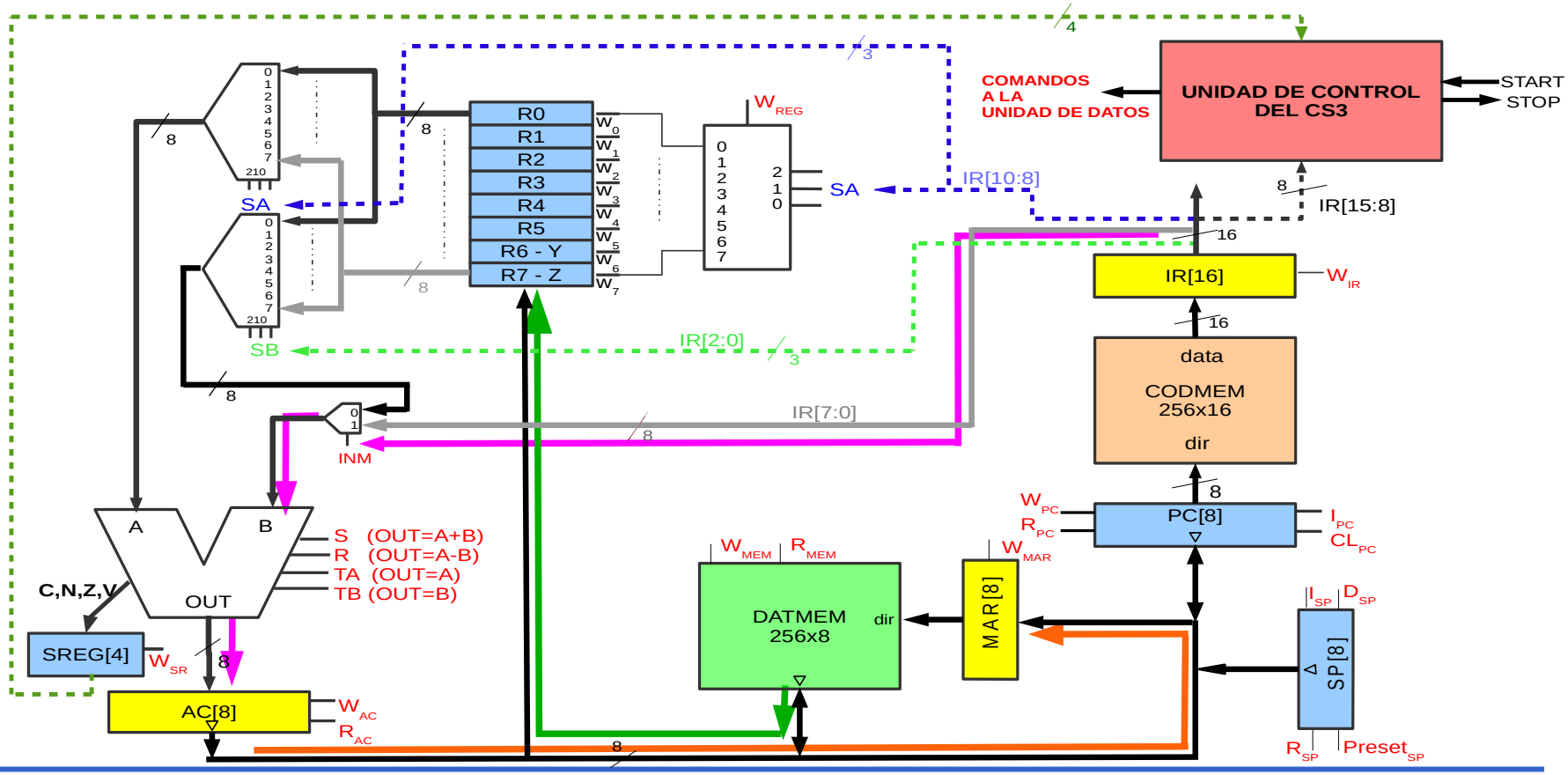


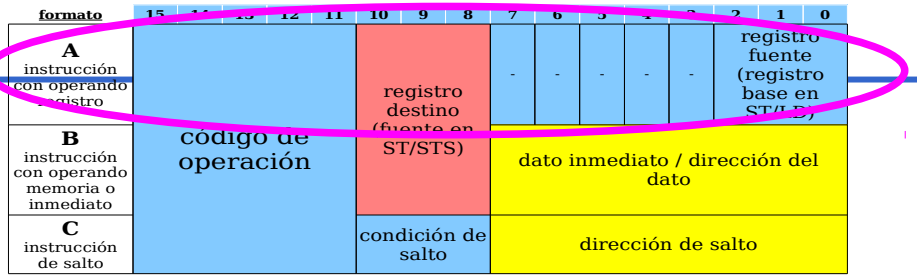
formato	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A instrucción con operando registro	código de operación					registro destino (fuente en ST/STS)			-	-	-	-	-	registro fuente (registro base en ST/LD)		
B instrucción con operando memoria o inmediato									dato inmediato / dirección del dato							
C instrucción de salto						condición de salto			dirección de salto							

CICLO	Ri ← DATMEM(dirección)	
1	AC ← IR[7:0]	W_{AC}, TB,INM
2	MAR ← AC	W_{MAR}, R_{AC}
3	$R(IR_{10-8}) \leftarrow DATAMEM(MAR)$	W_{REG}, R_{MEM}

Ejemplo de instrucción de transferencia de dato desde memoria

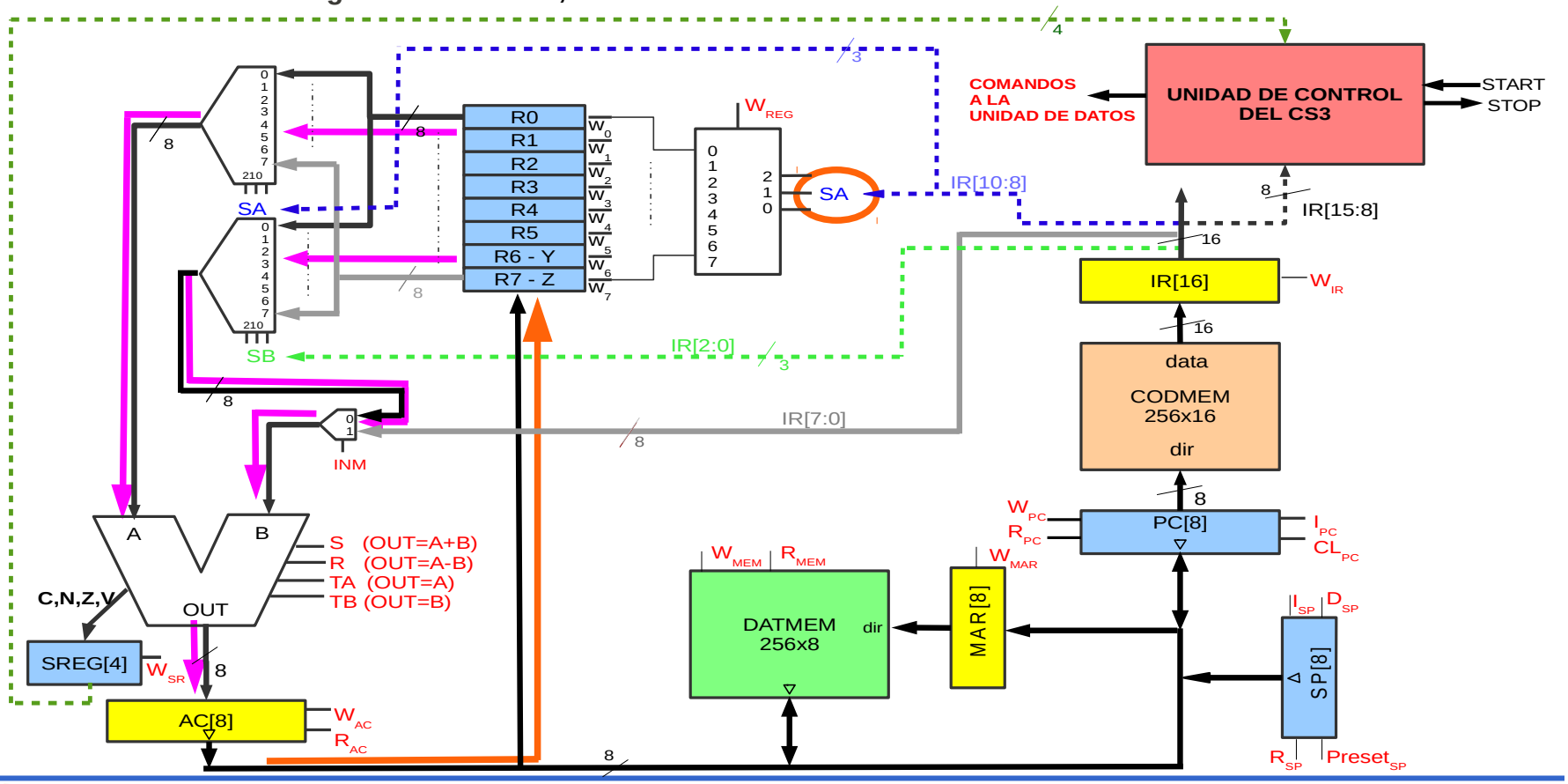
Direccionamiento absoluto **LDS Ri,dirección**





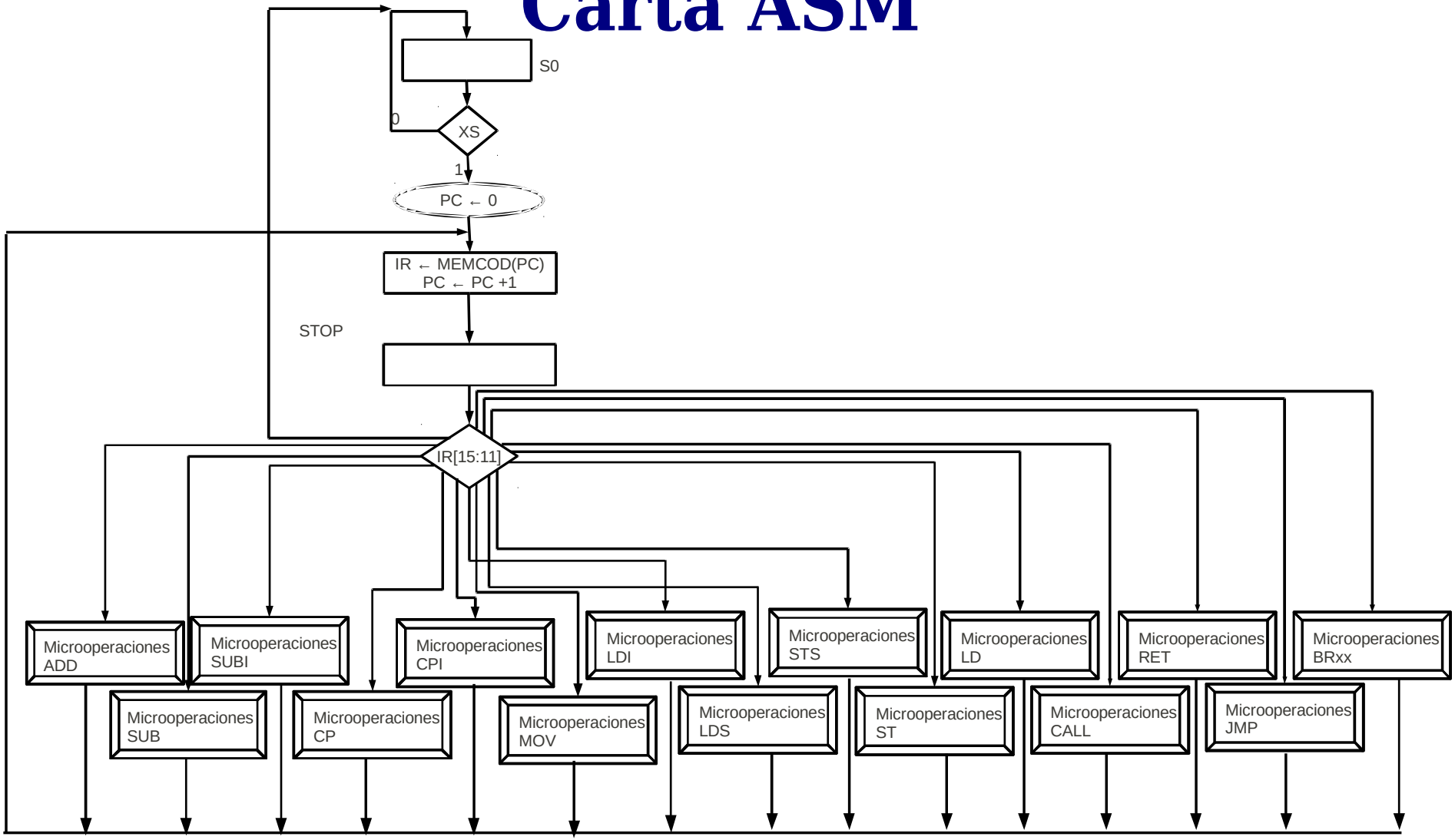
CICLO	Rd ← Rd+Rf	
1	AC ← $R(IR_{10-8})$ + $R(IR_{2-0})$ SREG ← INFO ESTADO	W_{AC}, S W_{SR}
2	$R(IR_{10-8})$ ← AC	W_{REG}, R_{Ac}

Ejemplo de instrucción de instrucción aritmética sobre pila de registros
Direccionamiento directo de registro ADD Rd,Rf



Programación

Carta ASM


S0
0
XS
1
PC ← 0
IR ← MEMCOD(PC)
PC ← PC +1
STOP
IR[15:11]
Microoperaciones ADD
Microoperaciones SUBI
Microoperaciones CPI
Microoperaciones LDI
Microoperaciones STS
Microoperaciones LD
Microoperaciones RET
Microoperaciones BRxx
Microoperaciones SUB
Microoperaciones CP
Microoperaciones MOV
Microoperaciones LDS
Microoperaciones ST
Microoperaciones CALL
Microoperaciones JMP

Lenguaje ensamblador CS3

El **lenguaje ensamblador** consta de:

- Instrucciones ejecutables: Son las desarrolladas para el CS3. Su sintaxis es la del mnemónico que se ha ido presentando.
- Directivas de ensamblado o pseudoinstrucciones: Son instrucciones que ayudan a escribir el programa y a documentarlo, pero que no causan código ejecutable.
- Conjunto de normas (léxicas, semánticas y sintácticas) que lo hacen operativo.

Directivas del ensamblador CS3

Las principales **normas y directivas de ensamblado del CS3** son:

- No distingue mayúsculas de minúsculas
- Comentario hasta final de línea ";" :

```
; Esto es un comentario: elimino ST (R2),R1
```

- Numerales: Formas de dar valores numéricos. Son: decimal con/sin signo (-27, 58); binario de 8 bits precedidos de 0b (0b11001111); hexadecimal de uno o dos dígitos precedidos de 0x (0xB, 0xF4).
- Etiquetas: Cadena alfanumérica que comienza por letra: p.ej. eti3. Pueden definirse con EQU o mediante identificador de línea de instrucción:

guarda: ST…; asociará a "guarda" el número de instrucción correspondiente a ST…

- .EQU <etiqueta>=<numeral>. Asocia un valor a una etiqueta:

```
.EQU n8=25          ;asocia el 25 a la etiqueta n8
```

Ejemplos de uso

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

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

Ejemplos de uso

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

```
MAX:        CP R1,R2
            BRLT R1MENOR
            MOV R0,R1
            RET
R1MENOR:    MOV R0,R2
            RET
```

Entrada/Salida

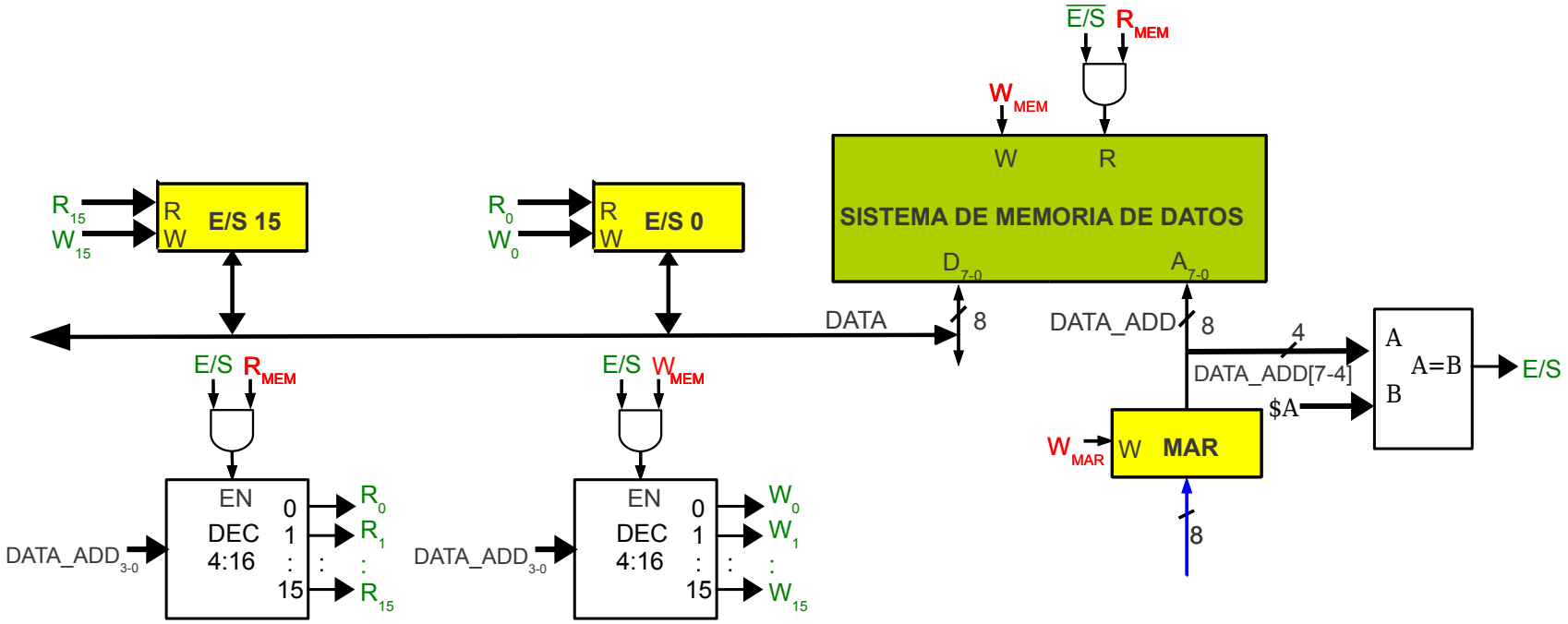
- El CS3 no ha permitido solucionar unas de las deficiencias del CS2 planteadas al principio de este tema: la incapacidad de comunicación con el exterior.
- Es decir, no posee capacidad de entrada salida.
- A continuación se expone un procedimiento que permitiría dotarlo de dicha capacidad.

Entrada/Salida

- Desde el punto de vista del programador la memoria es vista como un conjunto de bytes almacenados en direcciones concretas.
- Parte de ese rango de direcciones puede reservarse para que el procesador pueda acceder al exterior (entrada/salida)

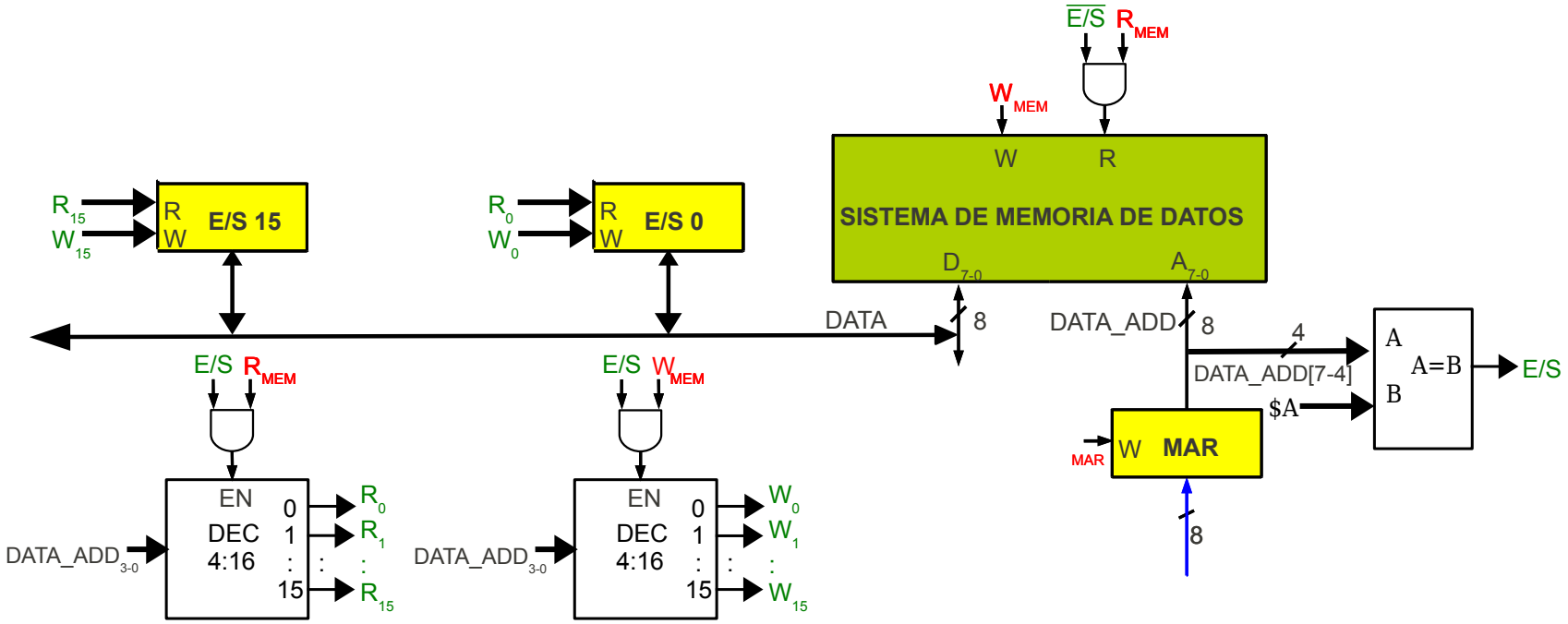
Entrada/Salida

- Reservaremos un subrango del espacio de direccionamiento (ej: $A0-$AF) para entrada/salida.
- Para ese subrango no se accede a la memoria de datos sino a los dispositivos de entrada/salida.



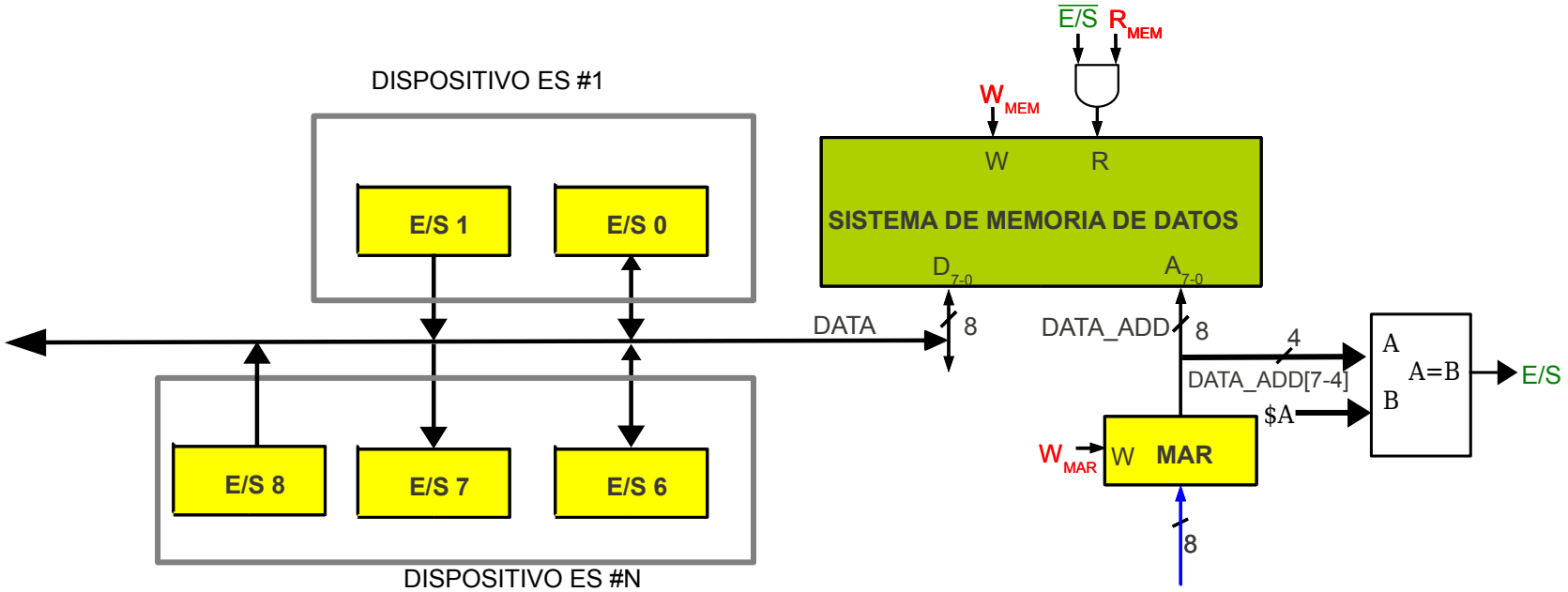
Entrada/Salida

- Los cuatro bits más significativos de DATA_ADD indican si la dirección pertenece al rango de entrada/salida .
- Para dicho rango, los bits restantes de DATA_ADD se utilizarán para seleccionar la posición de E/S concreta.



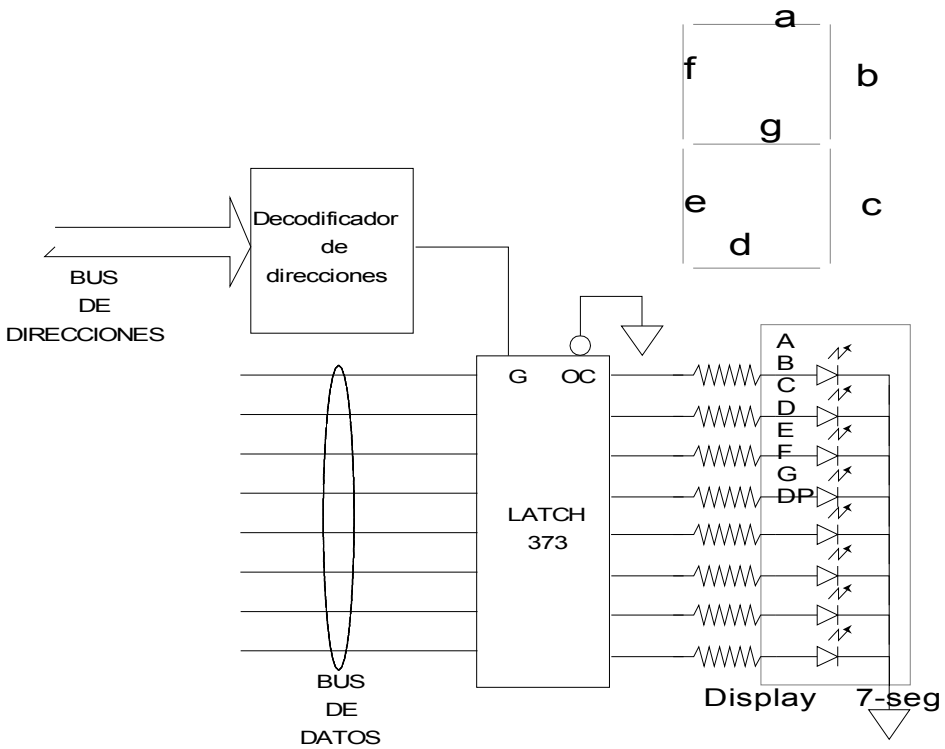
Entrada/Salida

- Las posiciones de E/S puede ser de lectura/escritura, de sólo lectura o de solo escritura
- Un dispositivo de E/S puede agrupar varias posiciones de E/S.



Entrada/Salida

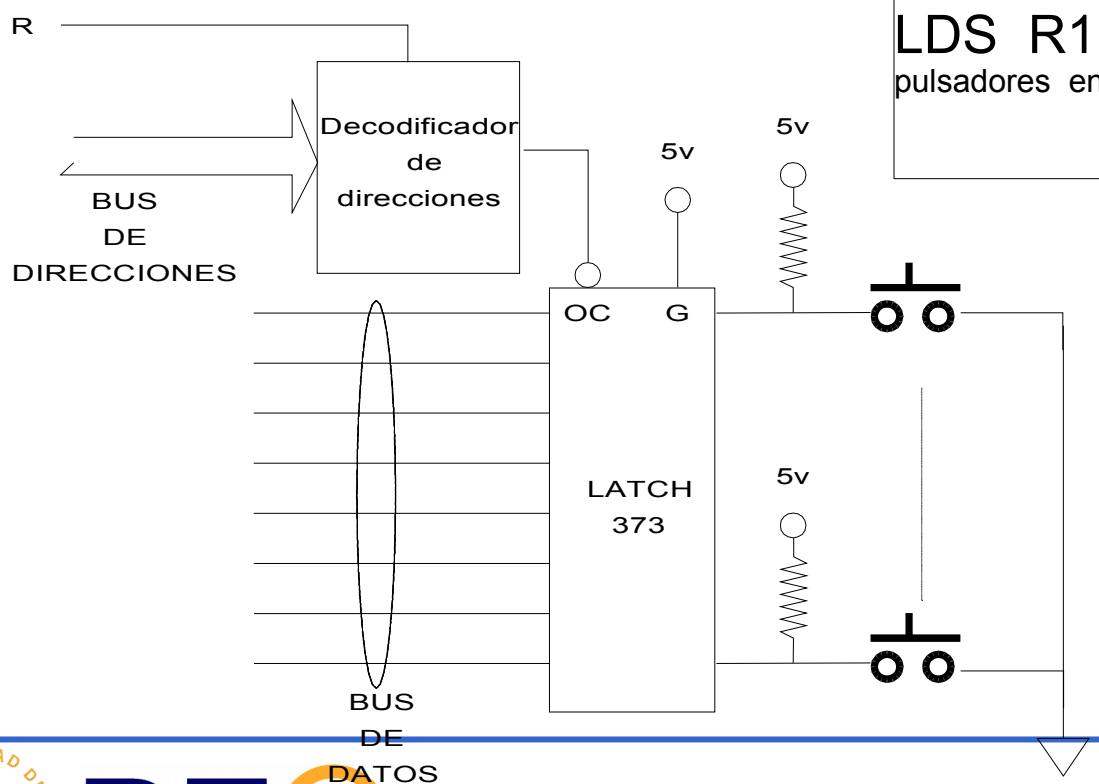
- Dispositivo de salida (display de 7 segmentos)




```
LDI  R1,$FF /*Código 7seg del número 8*/
STS $A0,R1  /*Muestra el número 8
en la dirección  del puerto de E/S*/
```

Entrada/Salida

- Dispositivo de entrada (pulsadores)




```
LDS R1, $A1 /*Almacena el estado de los pulsadores en la dirección del puerto de E/S */
```