Estructura de Computadores

El computador simple

Autores: David Guerrero. Alberto Molina e Isabel Gómez

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)**

CS2

- Es necesario aumentar la capacidad de almacenamiento de datos del sistema y que no quede únicamente limitada a sus 8 registros.
- Para ello se ampliará la arquitectura anterior y al nuevo sistema le llamaremos el computador simple 2 (CS2).
- Existen dos opciones para dotar al sistema de almacenamiento de datos:
  - Utilizar un único sistema de memoria para datos e instrucciones lo que se denomina arquitectura Von Neumann.
  - Utilizar sistemas de memoria distintos para datos e instrucciones, lo que es denominado arquitectura Harvard.

CS2

- 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.

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)

CO	SINTAXIS	FUNCIÓN
00000	ST Y o Z,Rf	MEMDAT(Y o Z) ← Rf
00001	LD Rd,Y o Z	Rd ← MEMDAT(Y o Z)
00010	STS dir,Rf	MEMDAT(dir) ← Rf
00011	LDS Rd,dir	Rd ← MEMDAT(dir)
11111	LDI Rd,dato	Rd ← dato
01000	ADD Rd,Rf	Rd ← Rd+Rf
01010	SUB Rd,Rf	Rd ← Rd-Rf
11010	SUBI Rd,dato	Rd ← Rd-dato
01111	MOV Rd,Rf	Rd ← Rf
10111	STOP	NOP

- Las 4 primeras instrucciones son para intercambio de datos con la memoria.
- Es necesario aumentar los bits del código de operación.
- Se han añadido nuevas formas de acceso a los operandos (modos de direccionamiento).

Conjunto de instrucciones (ISP)

CO	SINTAXIS	FUNCIÓN
00000	ST Y o Z,Rf	MEMDAT(Y o Z) ← Rf
00001	LD Rd,Y o Z	Rd ← MEMDAT(Y o Z)
00010	STS dir,Rf	MEMDAT(dir) ← Rf
00011	LDS Rd,dir	Rd ← MEMDAT(dir)
11111	LDI Rd,dato	Rd ← dato
01000	ADD Rd,Rf	Rd ← Rd+Rf
01010	SUB Rd,Rf	Rd ← Rd-Rf
11010	SUBI Rd,dato	Rd ← Rd-dato
01111	MOV Rd,Rf	Rd ← Rf
10111	STOP	NOP

Arquitectura load/store
No se opera contra la Memoria.
Solo se intercambian datos con ella en dos sentidos:

- **memoria->registros**
- **registros->memoria**

Conjunto de instrucciones (ISP)

CO	SINTAXIS	FUNCIÓN
00000	ST Y o Z,Rf	MEMDAT(Y o Z) ← Rf
00001	LD Rd, Y o Z	Rd ← MEMDAT(Y o Z)
00010	STS dir, Rf	MEMDAT(dir) ← Rf
00011	LDS Rd,dir	Rd ← MEMDAT(dir)
11111	LDI Rd,dato	Rd ← dato
01000	ADD Rd,Rf	Rd ← Rd+Rf
01010	SUB Rd,Rf	Rd ← Rd-Rf
11010	SUBI Rd,dato	Rd ← Rd-dato
01111	MOV Rd,Rf	Rd ← Rf
10111	STOP	NOP

Las operaciones de suma y resta se realizan sólo con datos guardados en registros

Conjunto de instrucciones (ISP)

CO	SINTAXIS	FUNCIÓN
00000	ST Y o Z,Rf	MEMDAT(Y o Z)←Rf
00001	LD Rd, Y o Z	Rd←MEMDAT(Y o Z)
00010	STS dir, Rf	MEMDAT(dir)←Rf
00011	LDS Rd,dir	Rd←MEMDAT(dir)
11111	LDI Rd,dato	Rd←dato
01000	ADD Rd,Rf	Rd←Rd+Rf
01010	SUB Rd,Rf	Rd←Rd-Rf
11010	SUBI Rd,dato	Rd←Rd-dato
01111	MOV Rd,Rf	Rd←Rf
10111	STOP	NOP

Se ha aumentado el número de bits del código de operación. Pero no se utilizan todas las combinaciones posibles que serían 32.

Conjunto de instrucciones (ISP)

CO	SINTAXIS	FUNCIÓN
00000	ST Y o Z,Rf	MEMDAT(Y o Z) ← Rf
00001	LD Rd,Y o Z	Rd ← MEMDAT(Y o Z)
00010	STS dir,Rf	MEMDAT(dir) ← Rf
00011	LDS Rd,dir	Rd ← MEMDAT(dir)
11111	LDI Rd,dato	Rd ← dato
01000	ADD Rd,Rf	Rd ← Rd+Rf
01010	SUB Rd,Rf	Rd ← Rd-Rf
11010	SUBI Rd,dato	Rd ← Rd-dato
01111	MOV Rd,Rf	Rd ← Rf
10111	STOP	NOP

Formas de direccionar
Directamente con el valor de la dirección de la memoria de datos
Con un dato guardado en un registro que tiene el valor de la dirección
Contenido del registro
Con el dato que esta guardado en la propia instrucción

Modos de direccionamiento

- Directo de registro: `MOV Rd,Rf`
  - El dato se encuentra en un registro.
- Indirecto de registro: `ST Y o Z,Rf`
  - El dato se encuentra en una posición de memoria cuya dirección está almacenada en un registro denominado “registro base”. (En este caso dos de los 8 que tiene la arquitectura, R6 o Y; R7 o Z).
- Absoluto: `STS dir,Rf`
  - El dato se encuentra en una posición de memoria cuya dirección se da como operando.
- Inmediato: `SUBI Rd,dato`
  - El dato se encuentra en la propia instrucción.

MOV Rd,Rf

MOV R6,R2

Antes de la ejecución

		Pos	cont
$03	R6	$00	$34
$58	R2	$01	$56
		$02	$78
		$03	$00
		Memoria	

Después de la ejecución

		Pos	cont
$58	R6	$00	$34
$58	R2	$01	$56
		$02	$78
		$03	$00
		Memoria	

ST Y o Z,Rf

ST Y,R2

Antes de la ejecución

		Pos	cont
$03	R6	$00	$34
$58	R2	$01	$56
		$02	$78
		$03	$00
		Memoria	

Después de la ejecución

		Pos	cont
$03	R6	$00	$34
$58	R2	$01	$56
		$02	$78
		$03	$58
		Memoria	

STS dir,Rf

STS $2,R2

Antes de la ejecución

		Pos	cont
$03	R6	$00	$34
$58	R2	$01	$56
		$02	$78
		$03	$00
		Memoria	

Después de la ejecución

		Pos	cont
$03	R6	$00	$34
$58	R2	$01	$56
		$02	$58
		$03	$00
		Memoria	

SUBI Rd,dato

SUBI R6,7

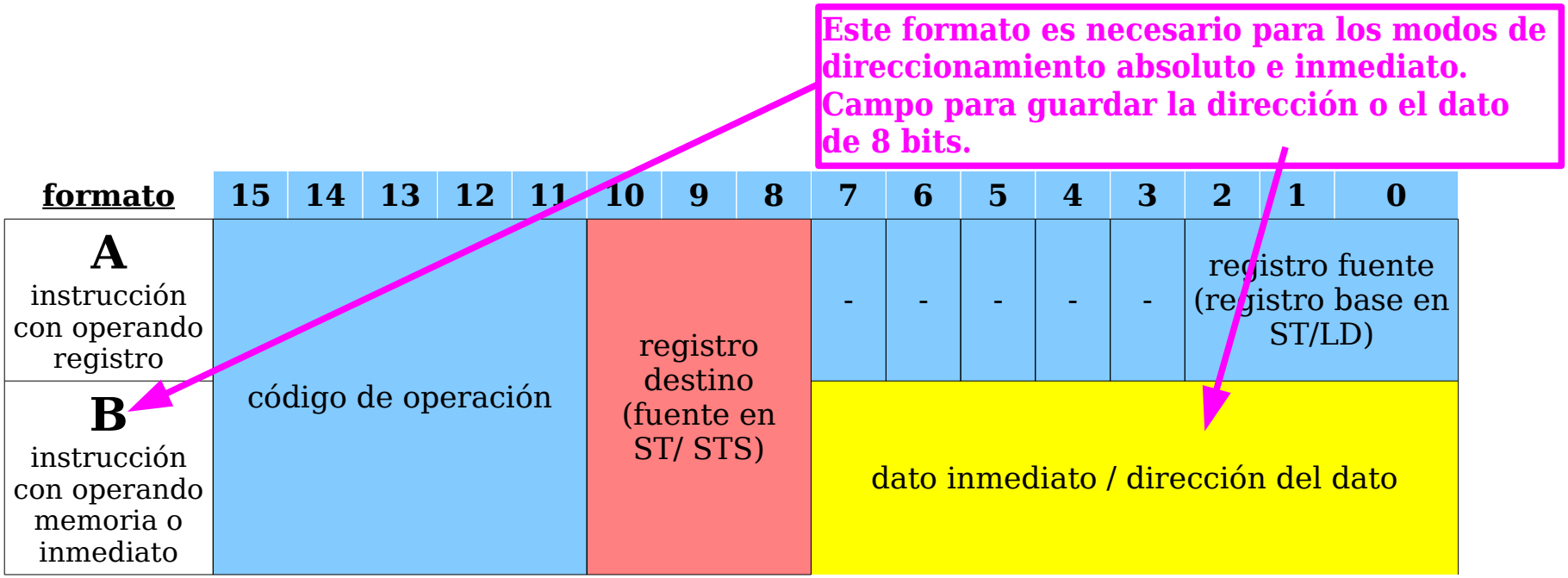
Antes de la ejecución

		Pos	cont
$03	R6	$00	$34
$58	R2	$01	$56
		$02	$78
		$03	$00
		Memoria	

Después de la ejecución

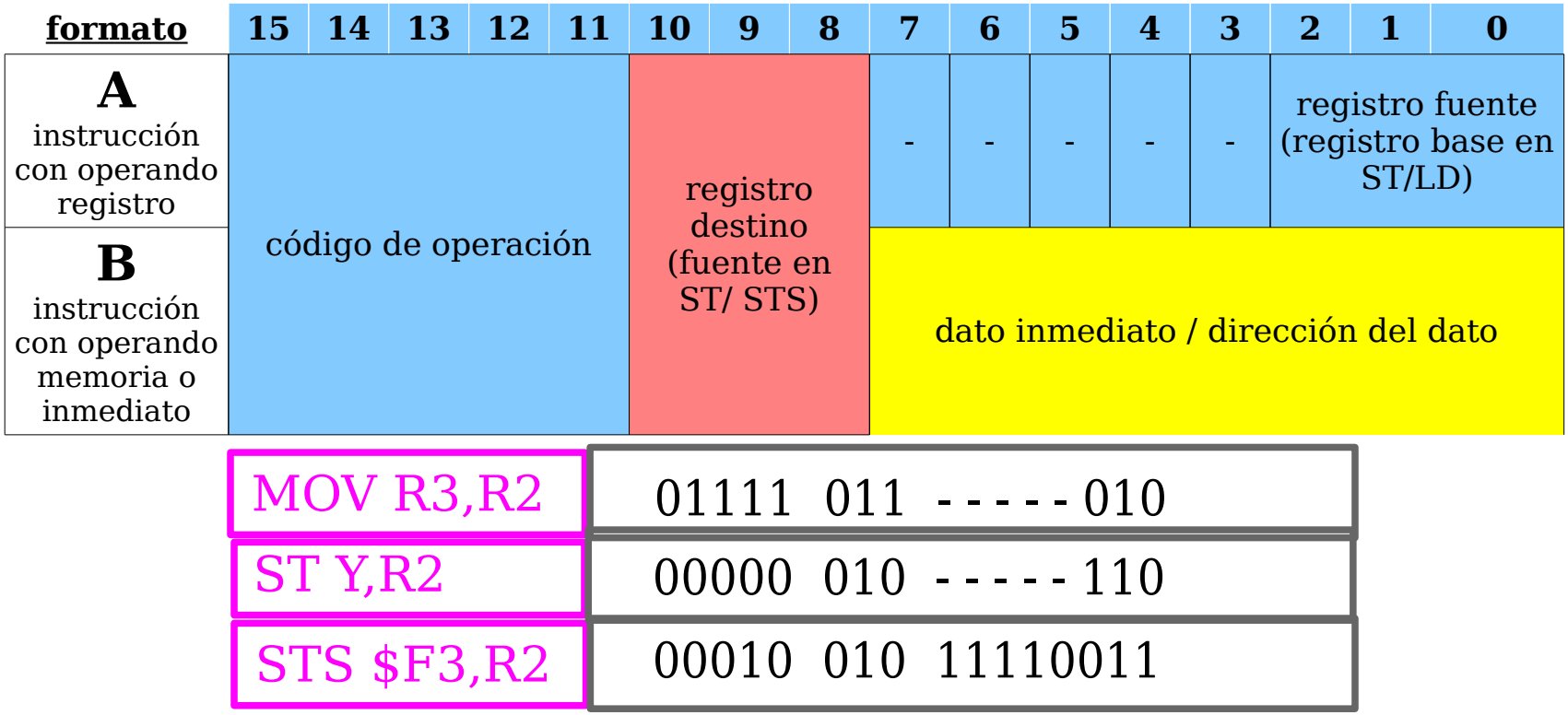
		Pos	cont
$FC	R6	$00	$34
$58	R2	$01	$56
		$02	$78
		$03	$00
		Memoria	

Formato de instrucción



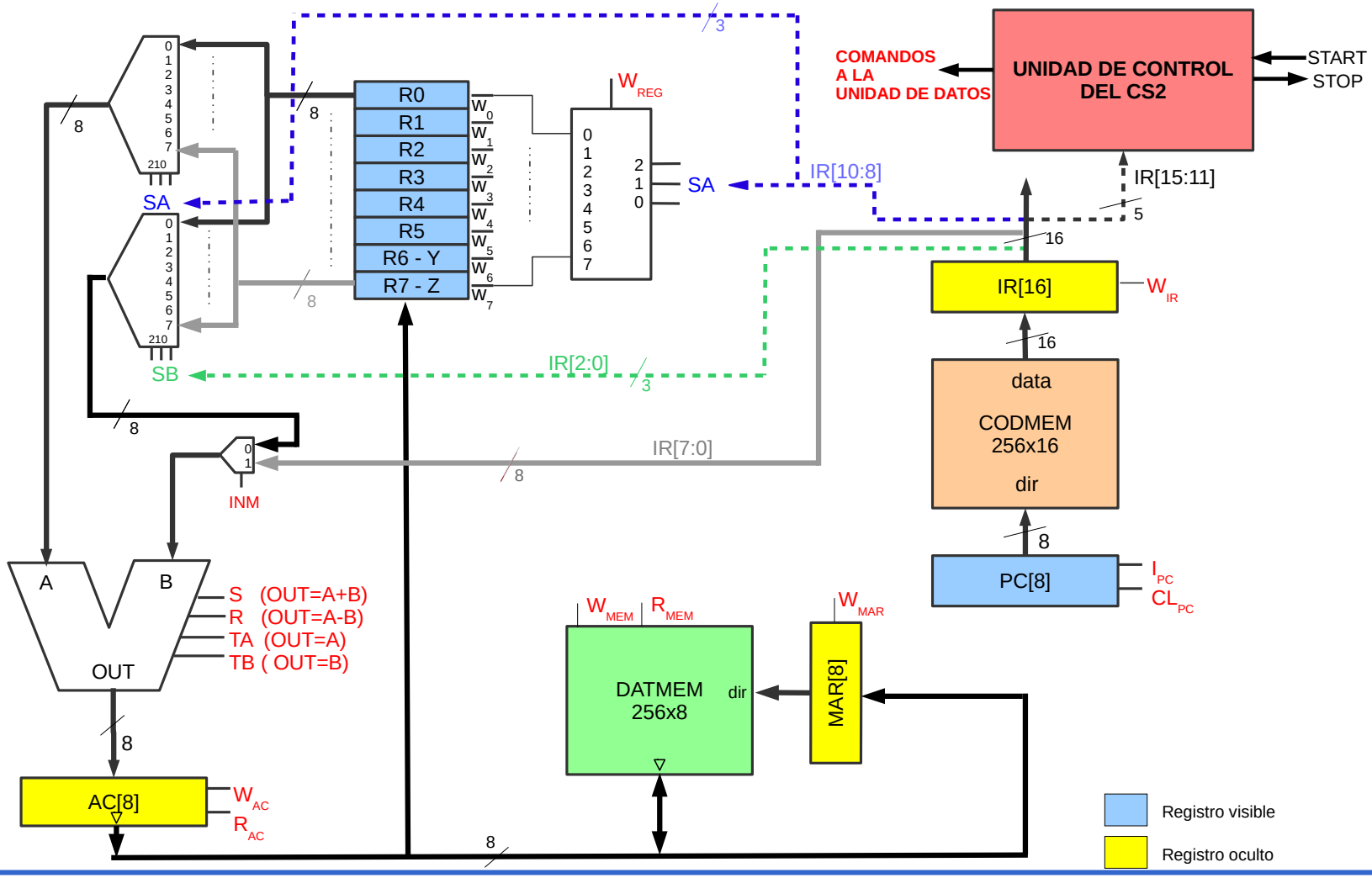
Las instrucciones son de 16 bits y siguen ocupando una palabra de memoria. En el CS2 el ancho de la memoria de código será de 16 bits. También será necesario que el registro IR sea de este tamaño.

Formato de instrucción

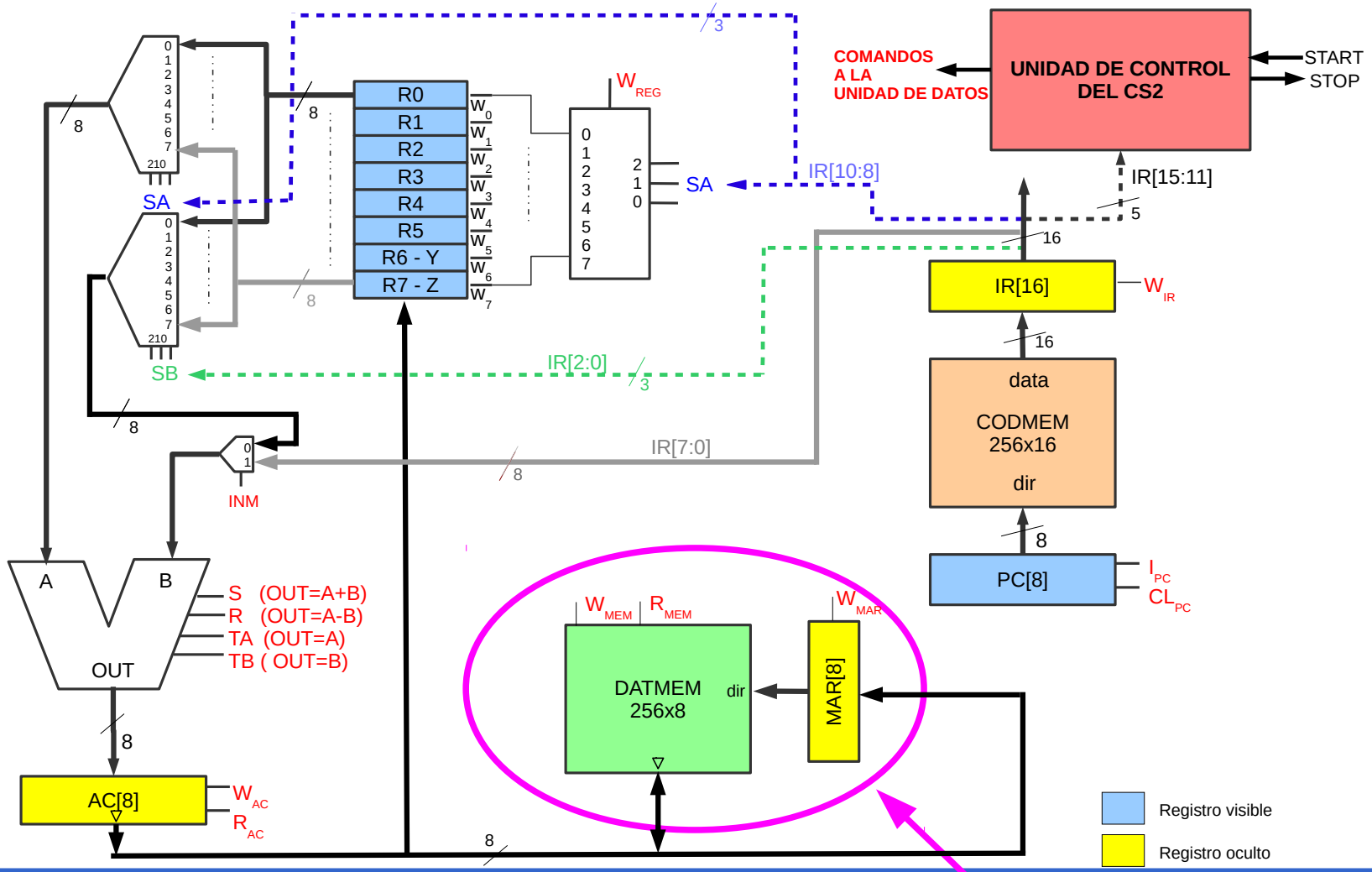


MOV R3,R2	01111 011 - - - - - 010
ST Y,R2	00000 010 - - - - - 110
STS $F3,R2	00010 010 11110011

CS2

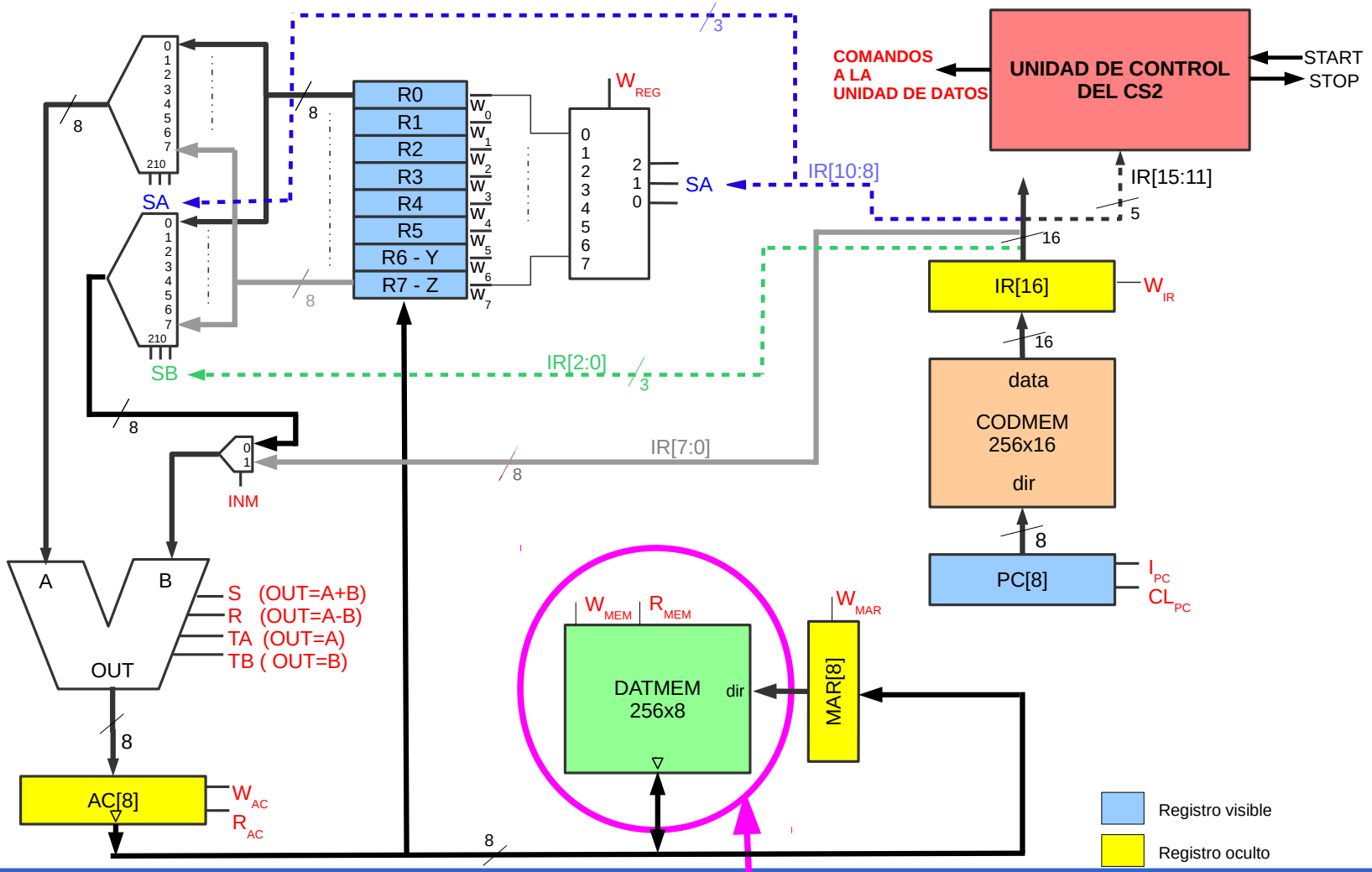

COMANDOS
A LA
UNIDAD DE DATOS
UNIDAD DE CONTROL
DEL CS2
START
STOP
IR[10:8]
IR[15:11]
IR[16]
W IR
data
CODMEM
256x16
dir
PC[8]
I PC
CL PC
R0
R1
R2
R3
R4
R5
R6 - Y
R7 - Z
W REG
SA
SB
IR[2:0]
IR[7:0]
INM
A
B
OUT
S (OUT=A+B)
R (OUT=A-B)
TA (OUT=A)
TB (OUT=B)
AC[8]
W AC
R AC
W MEM
R MEM
DATMEM
256x8
dir
W MAR
MAR[8]
Registro visible
Registro oculto

CS2



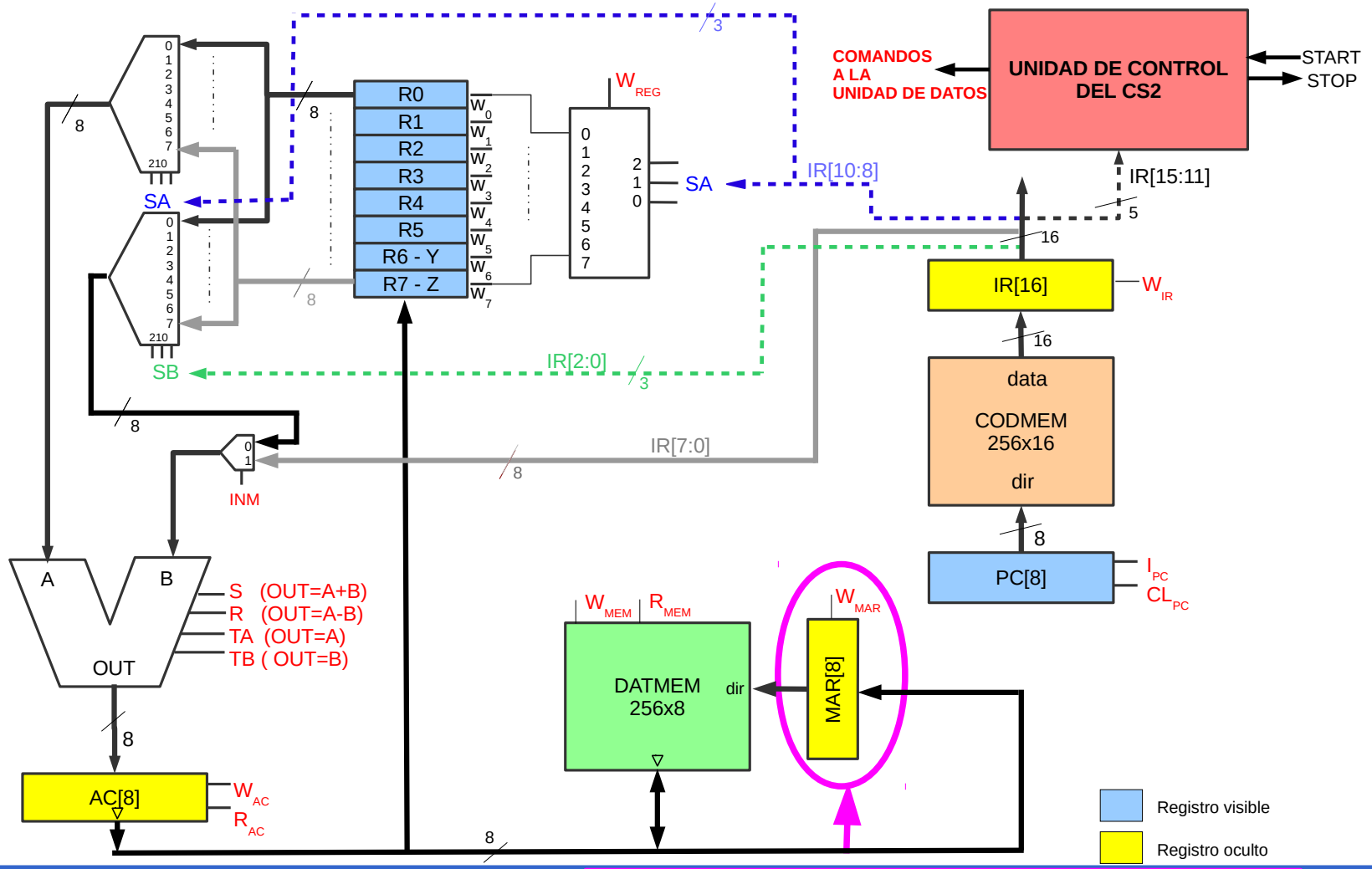
Parte añadida con objeto de guardar más datos además de los guardados en la pila de registros

CS2



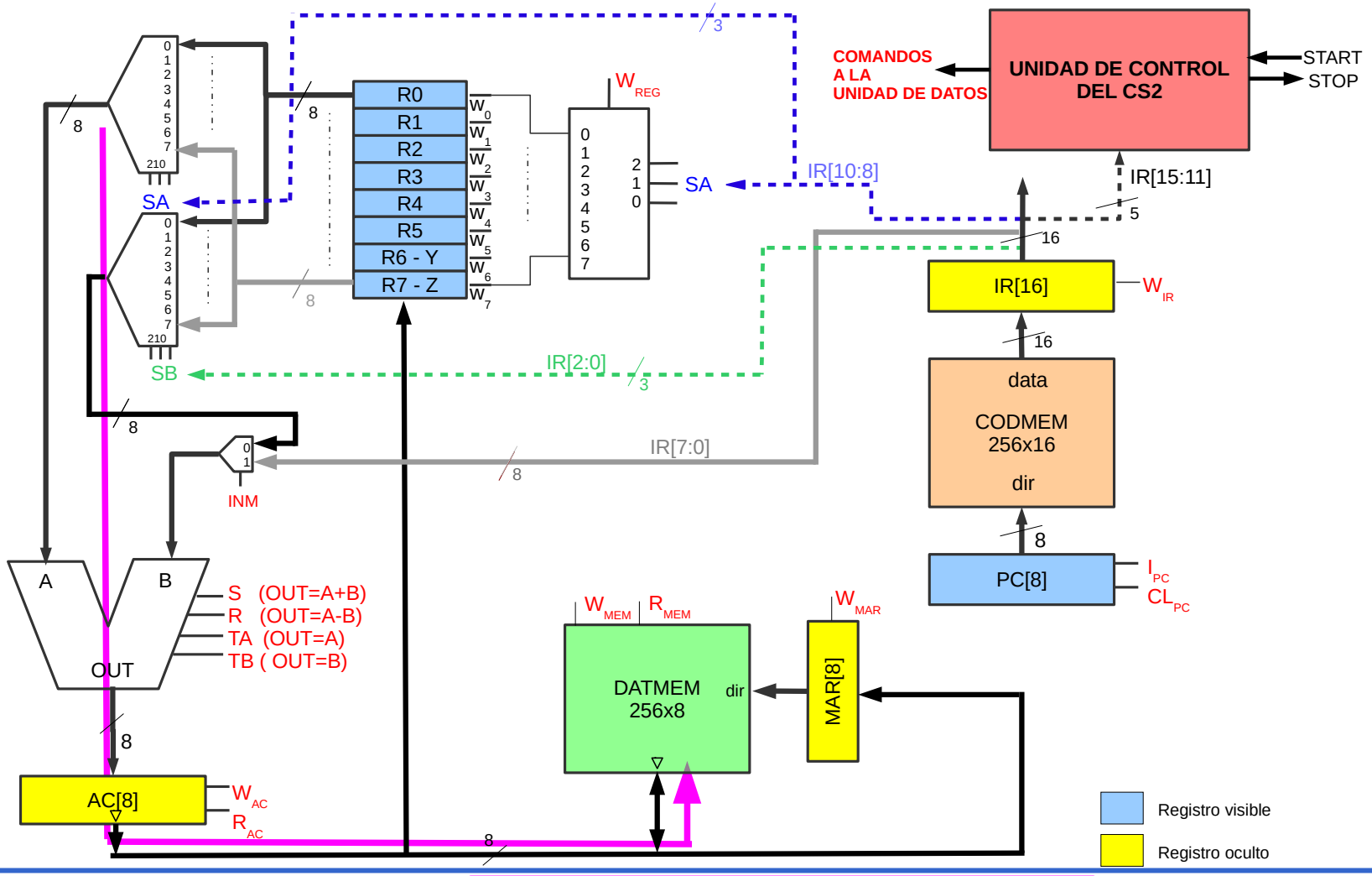
Memoria de lectura/escritura con capacidad 2^8x8. Terminales de datos bidireccionales

CS2



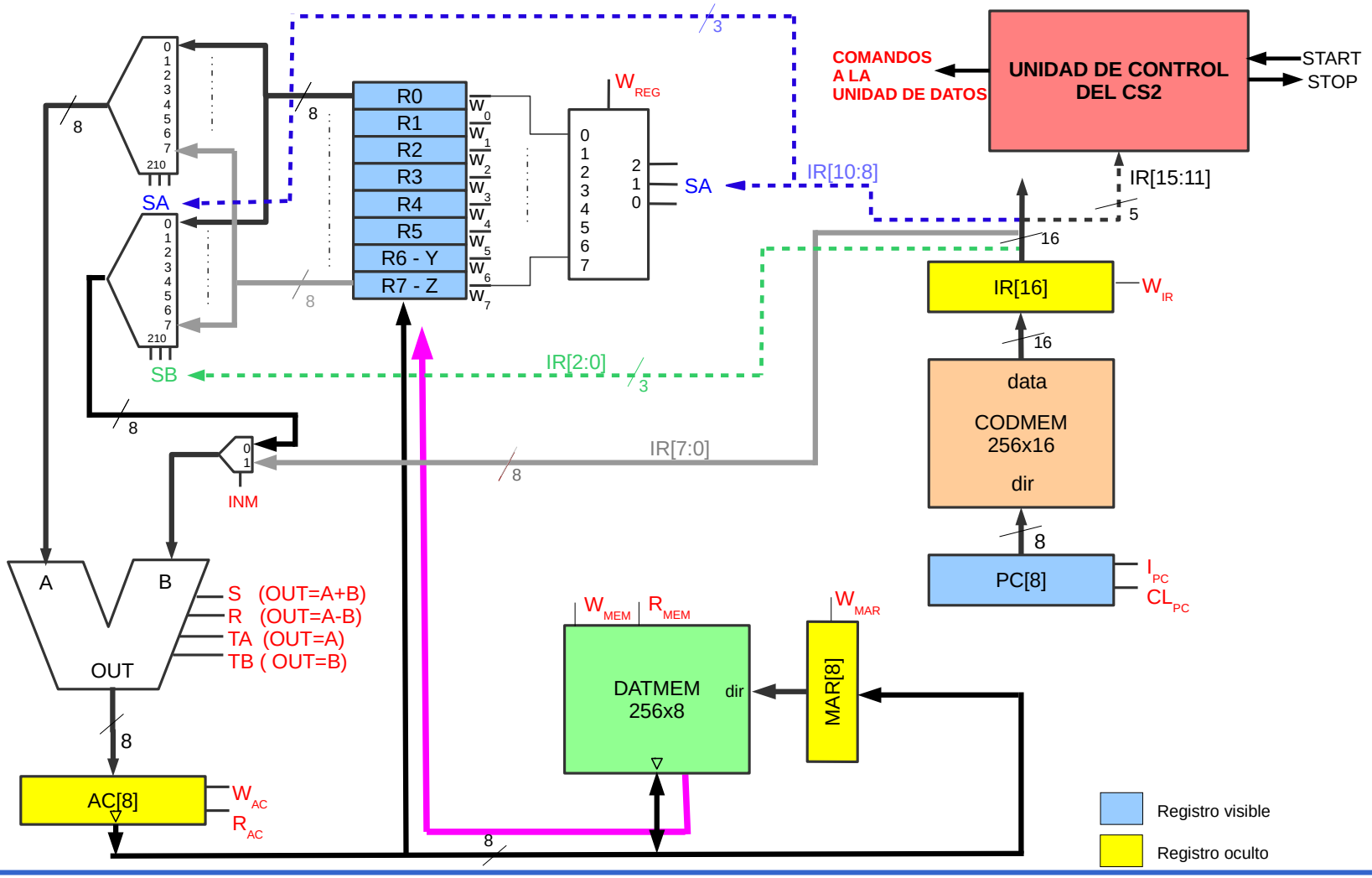
Registro que guarda de manera temporal las direcciones que van a ser accedidas en la memoria de datos

CS2



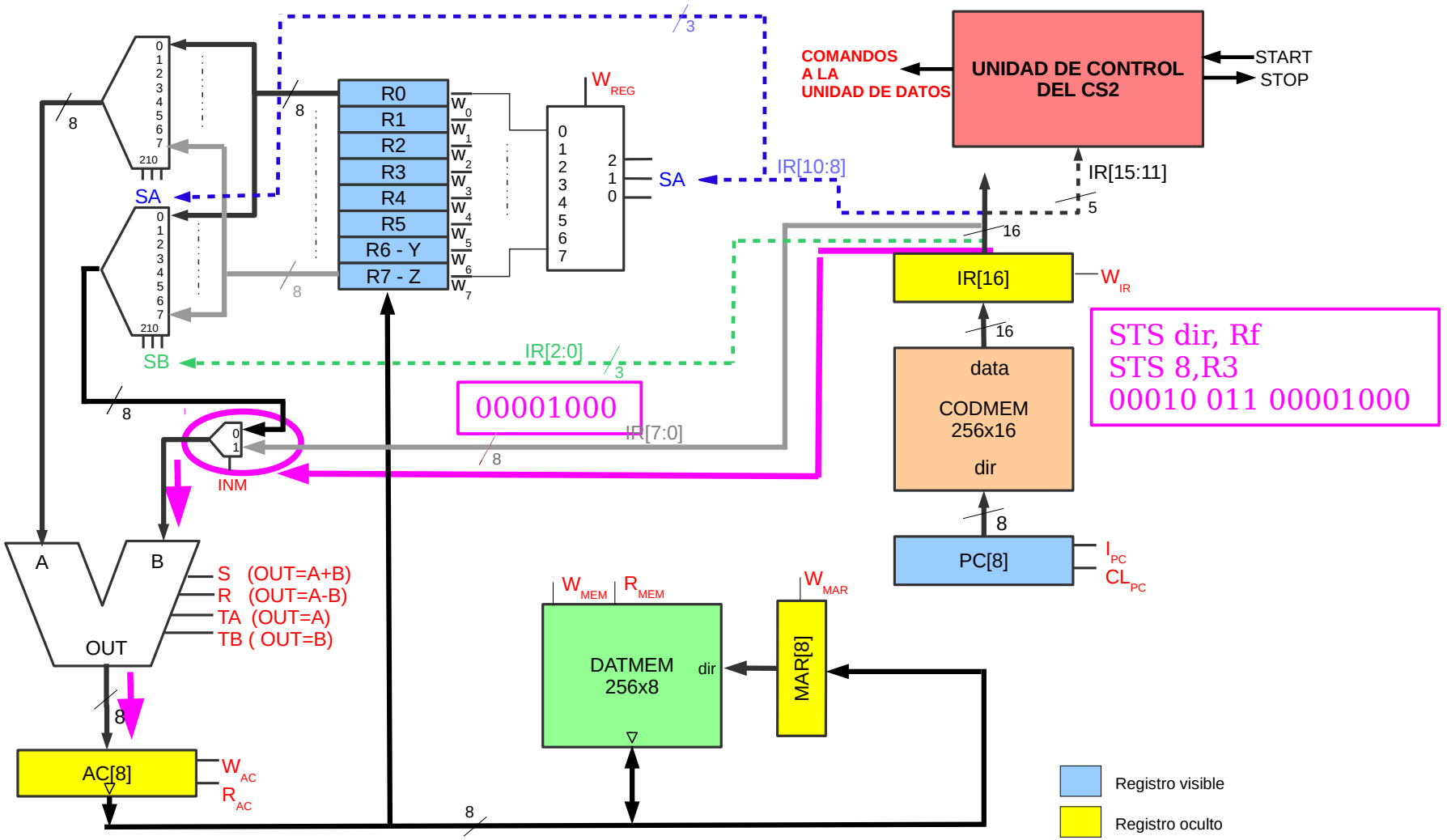
El camino para mover datos desde los registros siempre pasa por la ALU y el acumulador

CS2

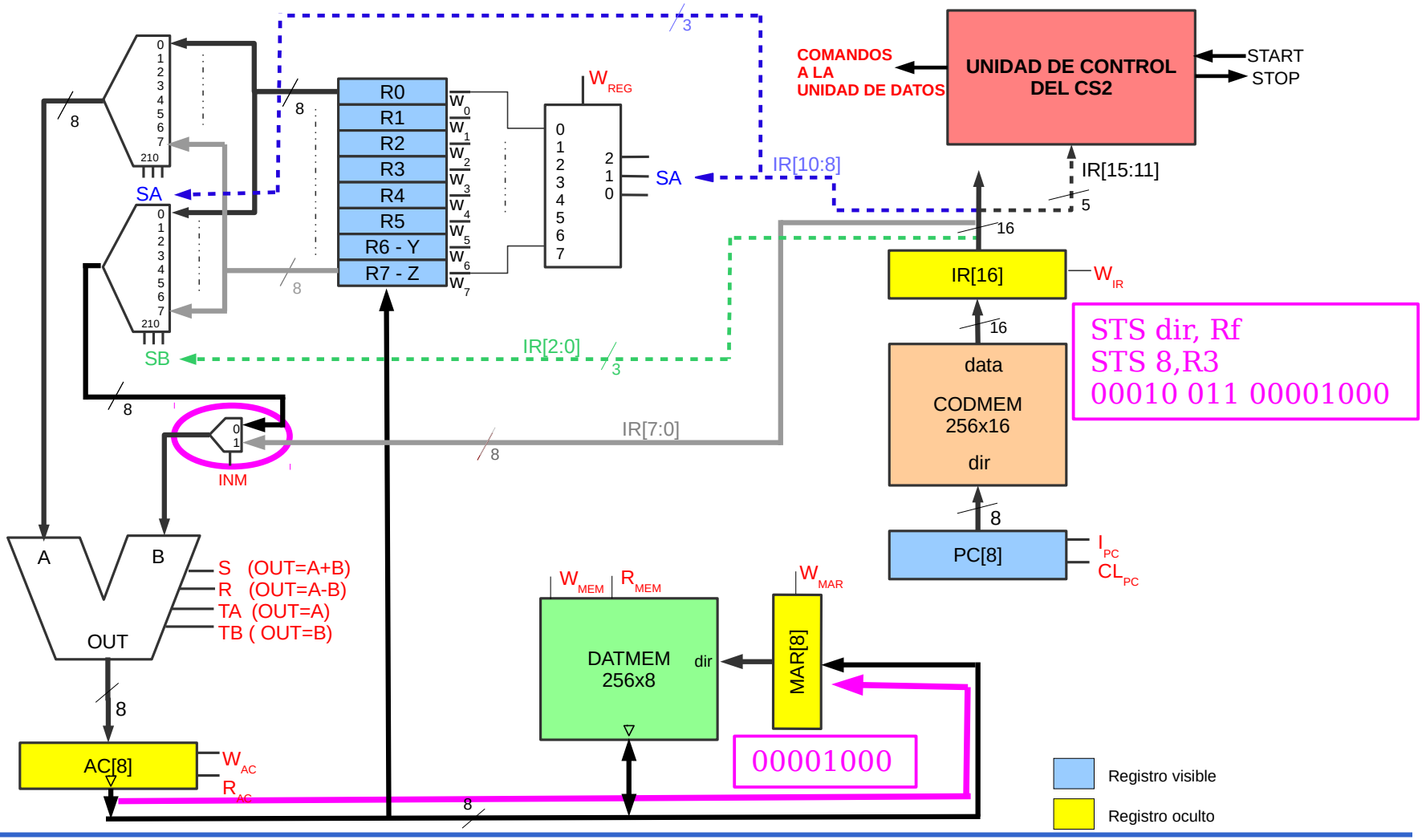


Camino para que los datos vayan de la memoria a los registros. Usa el bus compartido.

CS2


COMANDOS A LA UNIDAD DE DATOS
UNIDAD DE CONTROL DEL CS2
START
STOP
IR[15:11]
IR[10:8]
IR[2:0]
IR[7:0]
R0
R1
R2
R3
R4
R5
R6 - Y
R7 - Z
W_REG
SA
SB
IR[16]
W_IR
data
CODMEM
256x16
dir
00001000
STS dir, Rf
STS 8,R3
00010 011 00001000
INM
A
B
OUT
S (OUT=A+B)
R (OUT=A-B)
TA (OUT=A)
TB (OUT=B)
AC[8]
W_AC
R_AC
PC[8]
I_PC
CL_PC
W_MEM
R_MEM
W_MAR
DATMEM
256x8
dir
MAR[8]
Registro visible
Registro oculto

CS2


COMANDOS
A LA
UNIDAD DE DATOS
UNIDAD DE CONTROL
DEL CS2
START
STOP
IR[15:11]
IR[10:8]
IR[2:0]
IR[7:0]
R0
R1
R2
R3
R4
R5
R6 - Y
R7 - Z
W_REG
SA
SB
INM
IR[16]
W_IR
data
CODMEM
256x16
dir
PC[8]
I_PC
CL_PC
STS dir, Rf
STS 8,R3
00010 011 00001000
A
B
OUT
S (OUT=A+B)
R (OUT=A-B)
TA (OUT=A)
TB (OUT=B)
AC[8]
W_AC
R_AC
W_MEM
R_MEM
DATMEM
256x8
dir
W_MAR
MAR[8]
00001000
Registro visible
Registro oculto

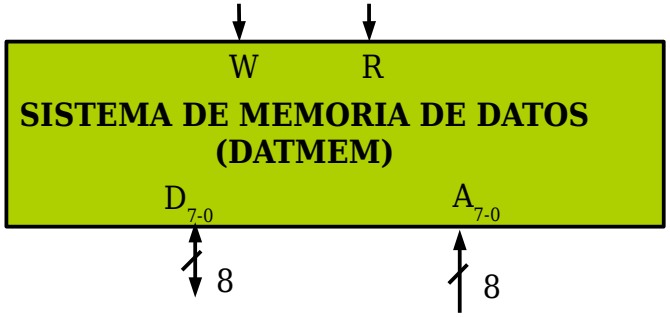
CS2

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

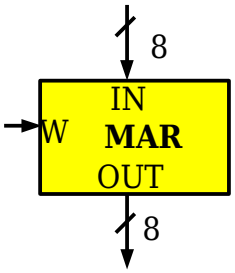
- Sistema de memoria de datos.
- Registro de direcciones de la memoria (MAR).
- Acumulador (AC). Se ha añadido este registro a la salida de la ALU porque esta debe compartir el bus.
- Multiplexor a la entrada de la ALU porque esta debe transferir datos tanto de los registros como de los 8 bits menos significativos del registro IR a MAR en el caso de direccionamientos absoluto e indirecto.
- Aumento del tamaño de palabra en la memoria de código
- Aumento del tamaño del registro IR

CS2

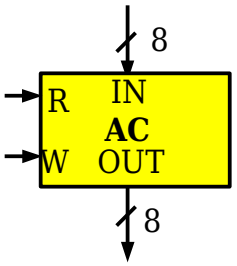
Descripción RT de los nuevos componentes



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



W	MAR ←	OUT:=
0	MAR	MAR
1	IN	MAR

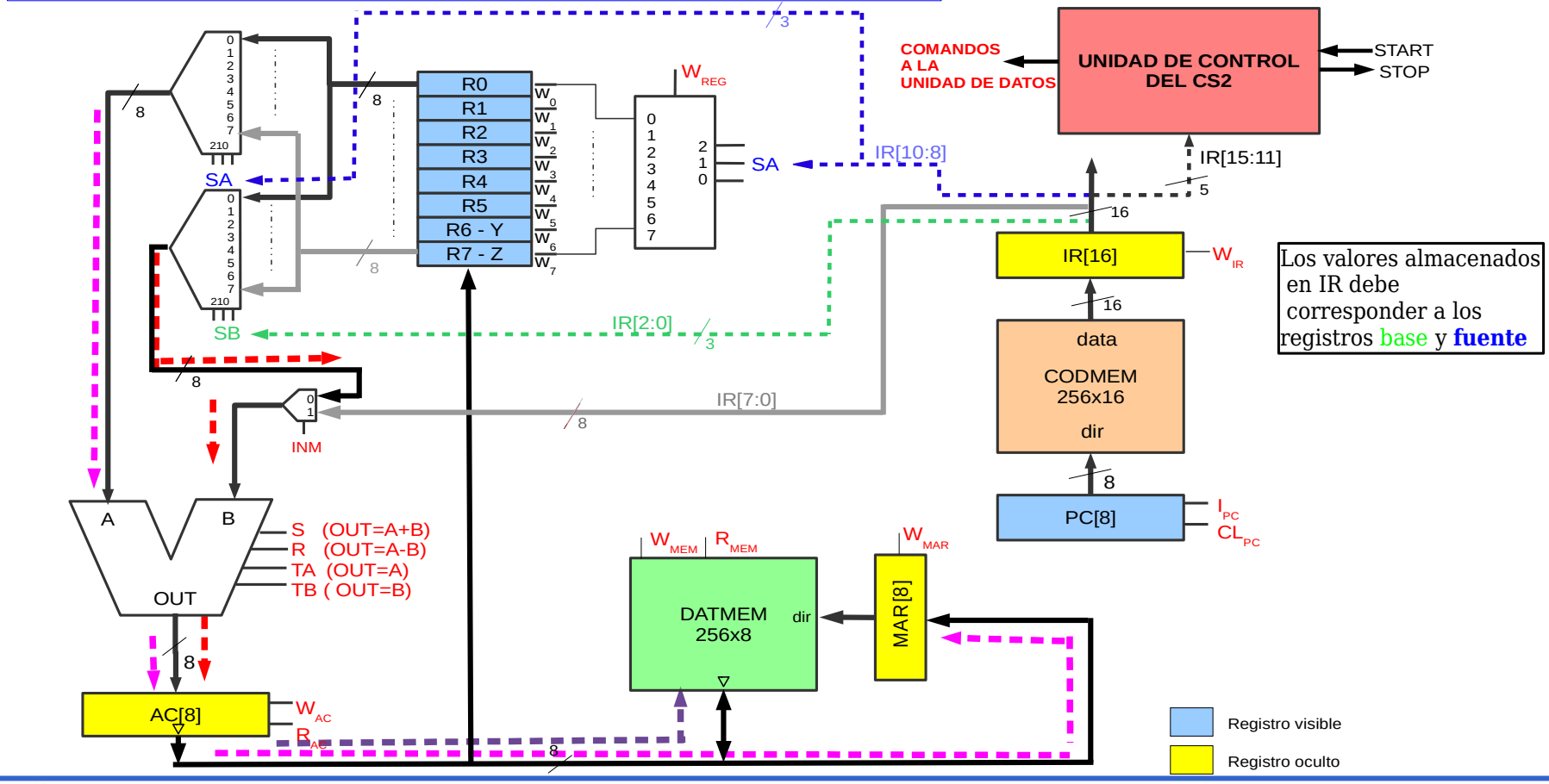


WR	AC ←	OUT:=
00	AC	H.I.
01	AC	AC
10	IN	H.I.
11	IN	AC

CS2

ST YoZ,Rf

1. AC←R[IR$_{2-0}$]	TB W_{AC}
2. MAR←AC; AC←R[IR$_{10-8}$]	R_{AC} W_{MAR} TA W_{AC}
3. DATMEM[MAR]←AC	R_{AC} W_{MEM}



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

CS2

Descripción de las microoperaciones de las instrucciones de intercambio de datos con la memoria

CO (I_{15} I_{14} I_{13} I_{12} I_{11})	00000	00001	00010	00011
Instrucción	**ST YoZ,Rf**	**LD Rd,YoZ**	**STS dir,Rf**	**LDS Rd,dir**
Micro 1	AC←R[Sb] (YoZ) TB W_{AC}	AC←R[Sb] (YoZ) TB W_{AC}	AC←IR_{7-0} TB W_{AC}INM	AC←IR_{7-0} TB W_{AC}INM
Micro 2	MAR←AC R_{AC} W_{MAR} AC←R[Sa] TA W_{AC}	MAR←AC R_{AC} W_{MAR}	MAR←AC R_{AC} W_{MAR} AC←R[Sa] TA W_{AC}	MAR←AC R_{AC} W_{MAR}
Micro 3	MEM(MAR)←AC R_{AC} W_{MEM}	R[Sa]←MEM(MAR) R_{MEM} W_{REG}	MEM(MAR)←AC R_{AC} W_{MEM}	R[Sa]←MEM(MAR) R_{MEM} W_{REG}

Nota: R[Sa] es $R[IR_{10-8}]$
R[Sb] es $R[IR_{2-0}]$

CS2

Descripción de las microoperaciones de las instrucciones cuyos operandos están en registros

CO (I_{15} I_{14} I_{13} I_{12} I_{11})	01000	01010	01111
Instrucción	**ADD Rd,Rf**	**SUB Rd,Rf**	**MOV Rd,Rf**
Micro 1	AC←R[Sa]+R[Sb] S W_{AC}	AC←R[Sa]-R[Sb] R W_{AC}	AC←R[Sb] TB W_{AC}
Micro 2	R[Sa]←AC R_{AC} W_{REG}	R[Sa]←AC R_{AC} W_{REG}	R[Sa]←AC R_{AC} W_{REG}

Nota: R[Sa] es R[IR_{10-8}]
R[Sb] es R[IR_{2-0}]

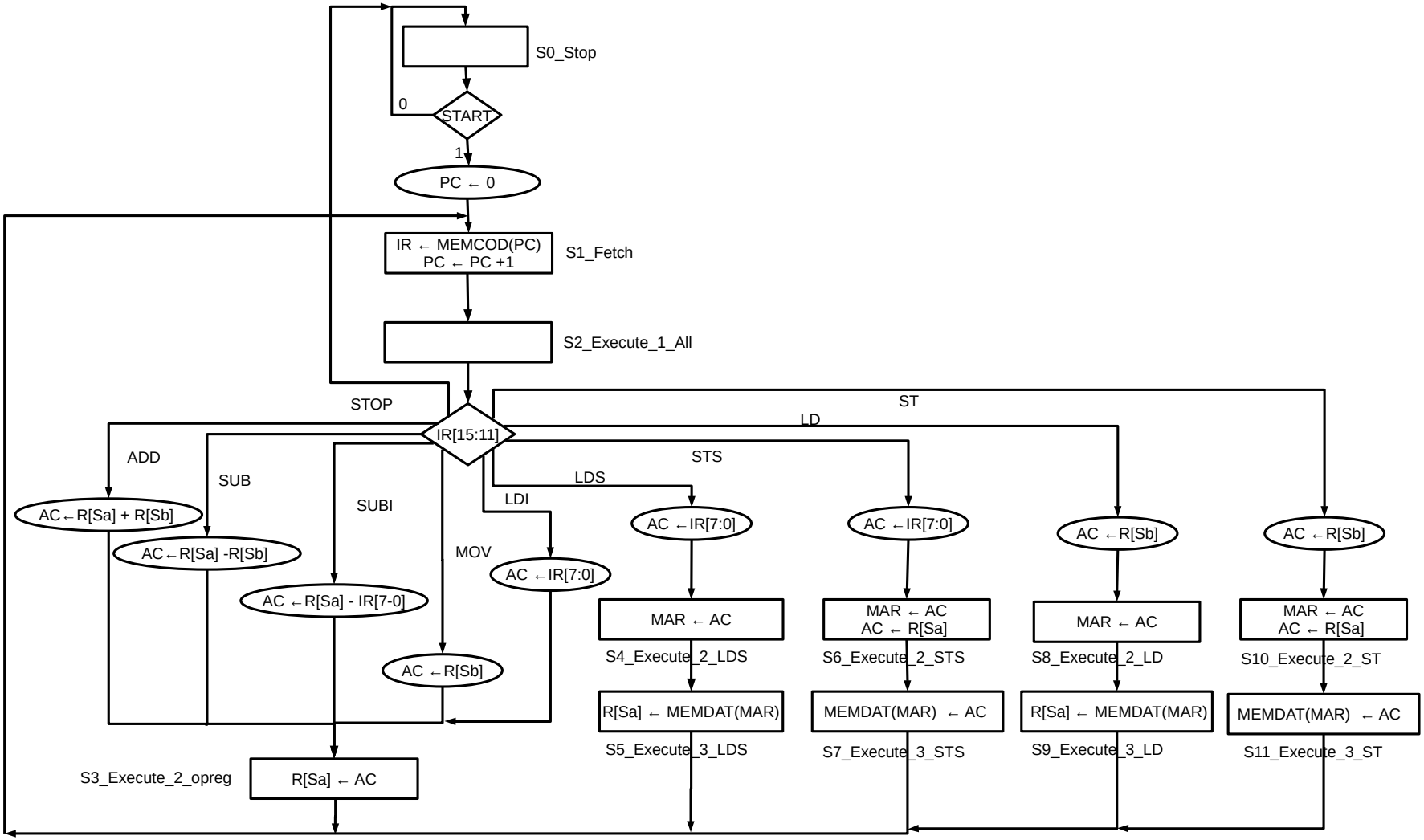
CS2

Descripción de las microoperaciones de las instrucciones que utilizan modo inmediato

CO (I_{15} I_{14} I_{13} I_{12} I_{11})	11111	11010
Instrucción	**LDI Rd,dato**	**SUBI Rd,dato**
Micro 1	AC←IR_{7-0} TB INM W_{AC}	AC←R[Sa]-IR_{7-0} R INM W_{AC}
Micro 2	R[Sa]←AC R_{AC} W_{REG}	R[Sa]←AC R_{AC} W_{REG}

Nota: R[Sa] es R[IR_{10-8}]
R[Sb] es R[IR_{2-0}]

Carta ASM


S0_Stop
0
START
1
PC ← 0
IR ← MEMCOD(PC)
PC ← PC +1
S1_Fetch
S2_Execute_1_All
STOP
IR[15:11]
ST
LD
STS
LDS
LDI
MOV
ADD
SUB
SUBI
AC ← R[Sa] + R[Sb]
AC ← R[Sa] -R[Sb]
AC ← R[Sa] - IR[7-0]
AC ← R[Sb]
AC ← IR[7:0]
AC ← IR[7:0]
AC ← IR[7:0]
AC ← R[Sb]
AC ← R[Sb]
MAR ← AC
S4_Execute_2_LDS
MAR ← AC
AC ← R[Sa]
S6_Execute_2_STS
MAR ← AC
S8_Execute_2_LD
MAR ← AC
AC ← R[Sa]
S10_Execute_2_ST
R[Sa] ← MEMDAT(MAR)
S5_Execute_3_LDS
MEMDAT(MAR) ← AC
S7_Execute_3_STS
R[Sa] ← MEMDAT(MAR)
S9_Execute_3_LD
MEMDAT(MAR) ← AC
S11_Execute_3_ST
S3_Execute_2_opreg
R[Sa] ← AC

Ejemplo de uso

Realizar un programa que intercambie dos tablas de 4 datos que se encuentran almacenadas en las direcciones $F0 y $FA de la memoria de datos.

Programa

```
LDI R2,1
LDI Y,$F0
LDI Z,$FA
LD R3,Y
LD R4,Z
ST Z,R3
ST Y,R4
ADD Y,R2
ADD Z,R2
LD R3,Y
LD R4,Z
ST Z,R3
ST Y,R4
ADD Y,R2
ADD Z,R2
LD R3,Y
LD R4,Z
ST Z,R3
ST Y,R4
STOP
```

MEMORIA DE CÓDIGO

Pos	contenido
$00	11111 01000000001
$01	11111 11011110000
$02	11111 11111111010
$03	00001 011 - - - - -110
$04	00001 100 - - - - -111
$05	00000 011 - - - - -111
$06	00000 100 - - - - -110
$07	01000 110 - - - - -010
$08	01000 111 - - - - -010
…	…
$19	10111 - - - - - - - - - -

MEMORIA DE DATOS

Pos	contenido
$00	…..
…..	
$F0	DATO1TABLA1
$F1	DATO2TABLA1
$F2	DATO3TABLA1
$F3	DATO4TABLA1
…	…
$FA	DATO1TABLA2
$FB	DATO2TABLA2
$FC	DATO3TABLA2
$FD	DATO4TABLA2
…	