

---

# Estructura de Computadores

## *El computador simple*

---

Autores: David Guerrero. Isabel Gómez

Usted es libre de copiar, distribuir y comunicar públicamente la obra y de hacer obras derivadas siempre que se cite la fuente y se respeten las condiciones de la licencia Attribution-Share alike de Creative Commons.

Texto completo de la licencia: <http://creativecommons.org/licenses/by-nc-sa/3.0/es/>



---

# Guión

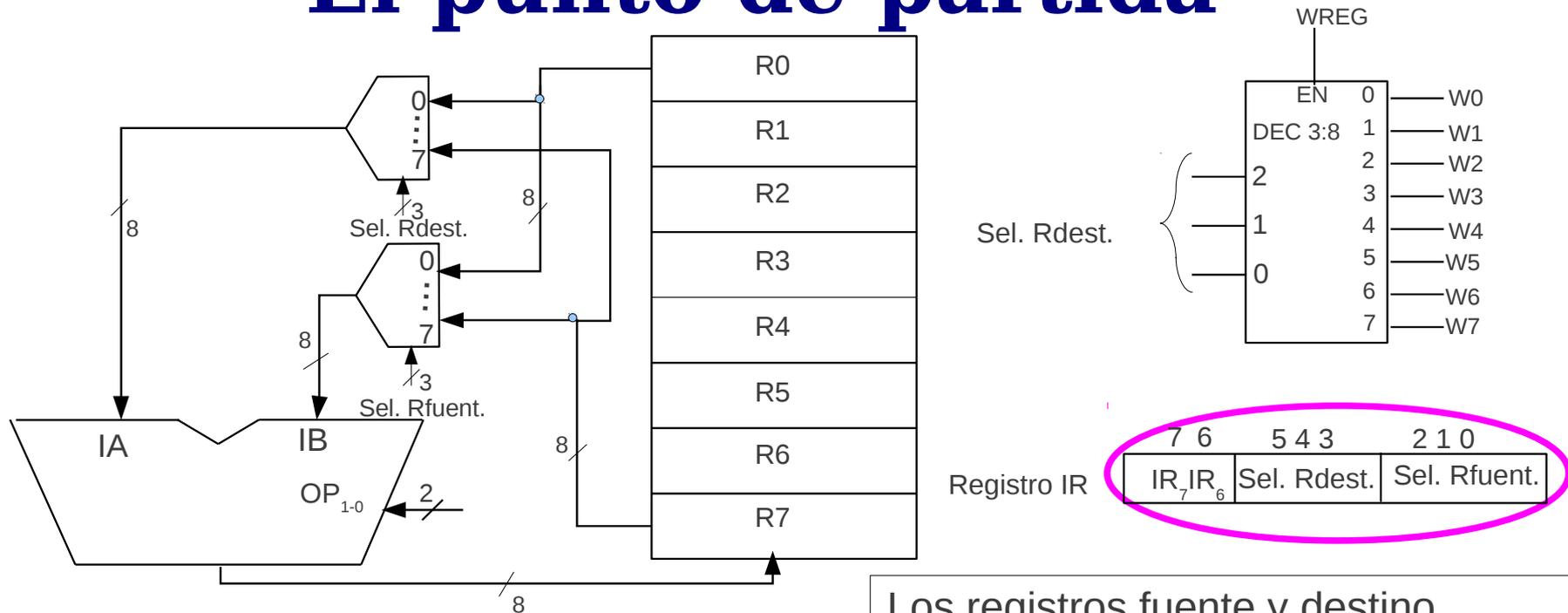
- ▶ **El punto de partida: La calculadora**
- ▶ **Automatización en la ejecución**
- ▶ **Almacenamiento de los datos**
- ▶ **Diversificación de instrucciones**
- ▶ **Una posible implementación**
- ▶ **Ejemplos de uso**

---

# El punto de partida

- ▶ 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 realizan en un único ciclo de reloj.

# El punto de partida



Los registros fuente y destino pueden ser cualquiera de los 8 pertenecientes a la arquitectura

IR <sub>7</sub> IR <sub>6</sub>	Operaciones
00	Rdest ← Rdest+Rfuent
10	Rdest ← Rdest-Rfuent
01	Rdest ← Rfuent

**Registro adicional que no posee la calculadora**  
**Es donde se guarda la información que pone el usuario desde el exterior. Información sobre la operación a realizar y los datos implicados**

# El punto de partida

Se propone la realización de un instrucción más compleja:  $R0 \leftarrow 3R1 - R2$

Secuencia de instrucciones a nivel ISP:

Instrucción 1.  $R0 \leftarrow R1$

Instrucción 2.  $R0 \leftarrow R0 - R2$

Instrucción 3.  $R0 \leftarrow R0 + R1$

Instrucción 4.  $R0 \leftarrow R0 + R1$

Las instrucciones deben ser las de la calculadora.

Resta siempre en el mismo sentido y sólo registros destino y fuente implicados.

En este ejemplo se actualizaría 4 veces el registro IR.

## Ventajas:

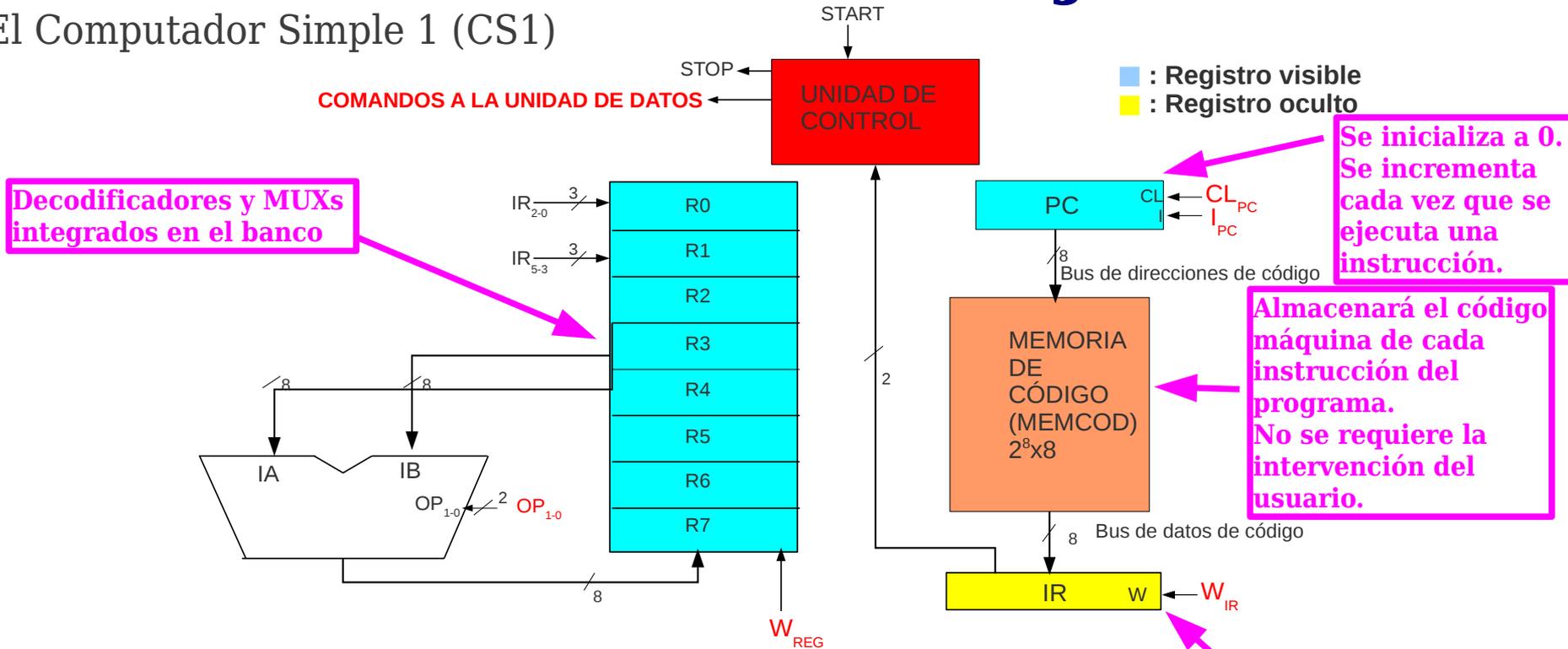
- Podemos resolver problemas más complejos que los resueltos por las instrucciones primitivas mediante una secuencia adecuada de las mismas.
- El usuario del sistema no necesita ser especialista en la electrónica del sistema.

## Deficiencias:

- No hay AUTOMATIZACIÓN EN LA EJECUCIÓN del programa: para ejecutar cada instrucción el usuario debe proporcionar el valor de xs y esperar fin para cada una de las cuatro instrucciones.
- No hay PROGRAMA ALMACENADO: cada vez que se ejecuta una instrucción el usuario debe suministrar la siguiente.

# Automatización en la ejecución

El Computador Simple 1 (CS1)



- Se ha simplificado el dibujo del banco de registros.
- La memoria permite el almacenamiento de un programa.
- Se ha añadido el registro PC (contador del programa) que permite el acceso a las instrucciones almacenadas en la memoria.
- El registro IR almacena la instrucción que el sistema está ejecutando.
- Se distingue entre registros visibles y no y visibles (ocultos): Los visibles aparecen en la descripción de las macrooperaciones.

Contendrá el código máquina de la instrucción que se esté ejecutando en cada momento.

---

# Automatización en la ejecución

- ▶ En el CS1 todas las instrucciones ocupan una posición de la memoria de código. Esto es una simplificación ya que en muchos sistemas reales son de longitud variable.
- ▶ Dada la simplicidad de la arquitectura, los programas siempre están almacenados a partir de la dirección 0 de la memoria y la ejecución es lineal.
- ▶ Instrucción en código máquina: Es el patrón de bits correspondiente a una instrucción.
- ▶ Formato de instrucción: Indica como se divide el código máquina de dicha instrucción en distintos campos (código de operación y operandos).

# Automatización en la ejecución

## ► Formato de instrucción del CS1

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

En el CS1 los datos siempre están almacenados en sus 8 registros y por tanto los operandos están codificados por 6 bits. Los tres primeros identifican al registro destino y los tres últimos al fuente.

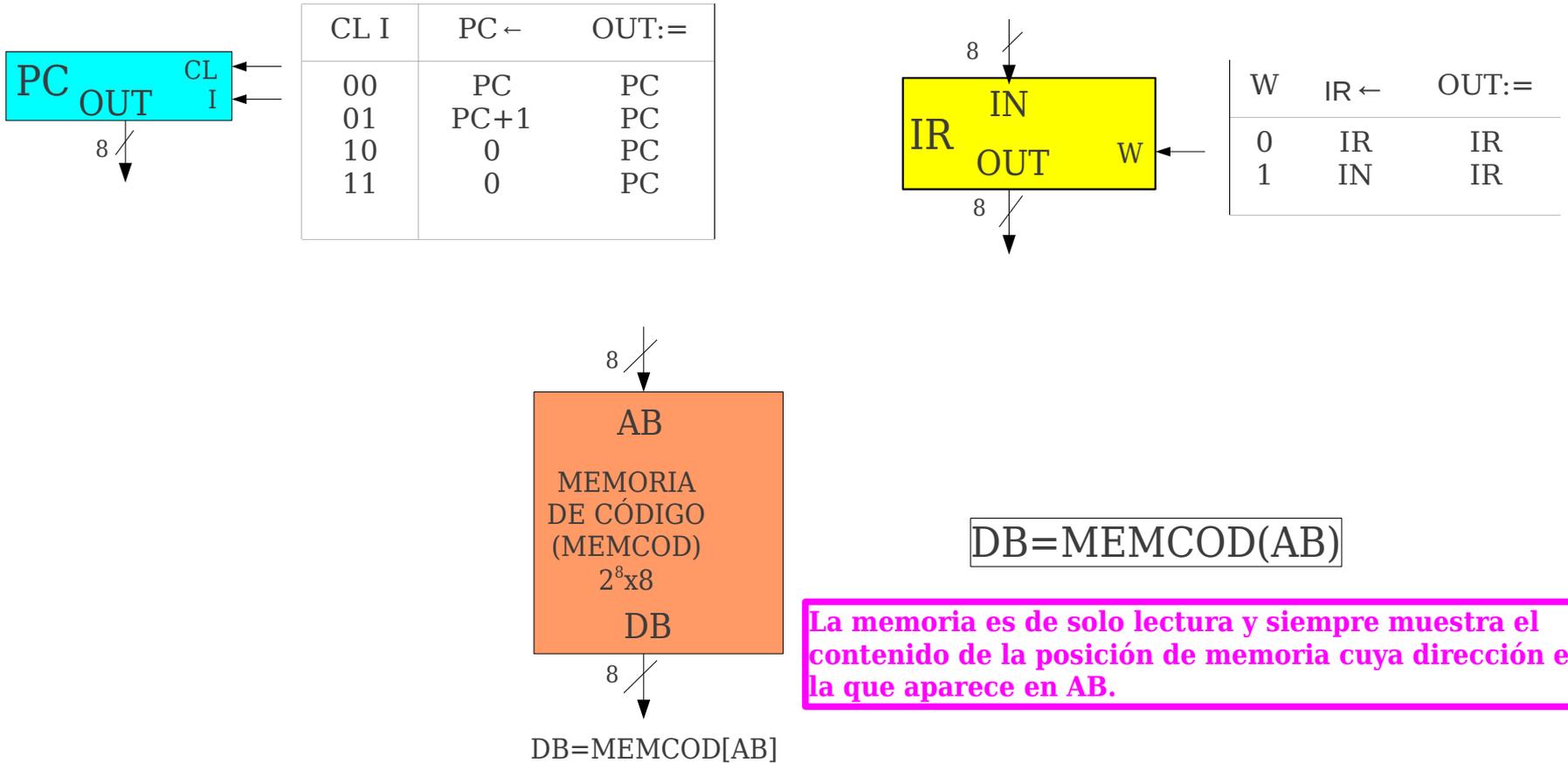
## ► Instrucciones. Se definen 4 instrucciones ya que se dispone de dos bits de código de operación.

CO: IR7 IR6	SINTAXIS	FUNCIÓN
00	ADD Rd,Rf	$Rd \leftarrow Rd + Rf$
10	SUB Rd,Rf	$Rd \leftarrow Rd - Rf$
01	MOV Rd,Rf	$Rd \leftarrow Rf$
11	STOP	NOP

*En la sintaxis se han utilizado nemónicos que facilitan La tarea al programador.*

# Automatización en la ejecución

## Descripción RT de los nuevos componentes

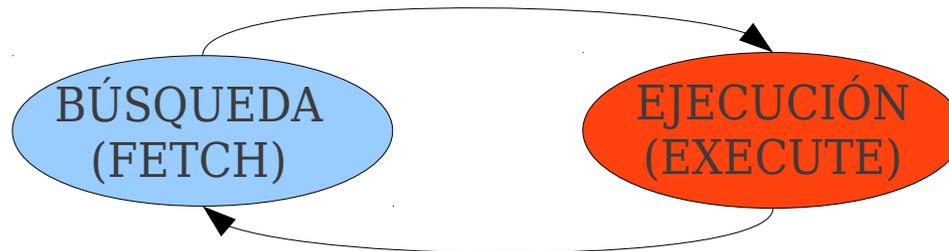


# Automatización en la ejecución

► Diseño de la unidad de control: permite la ejecución automática del programa almacenado en la memoria.

► Ciclo de instrucción

La búsqueda consiste en la lectura del código máquina de la instrucción que se va a ejecutar.



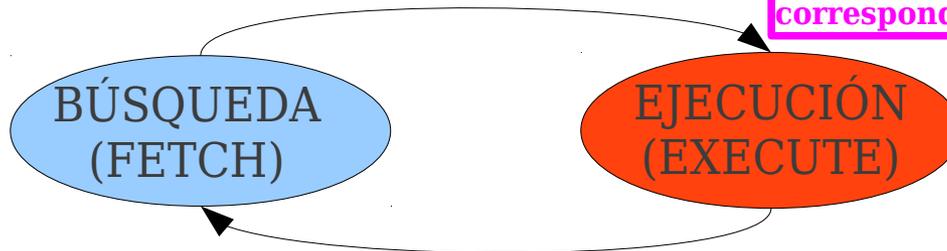
► La carta ASM del CS1 implementa los ciclos de todas las instrucciones del sistema.

C.O: IR7 IR6	SINTAXIS	FUNCIÓN
00	ADD Rd,Rf	$Rd \leftarrow Rd+Rf$
10	SUB Rd,Rf	$Rd \leftarrow Rd-Rf$
01	MOV Rd,Rf	$Rd \leftarrow Rf$
11	STOP	NOP

# Automatización en la ejecución

► Diseño de la unidad de control: permite la ejecución automática del programa almacenado en la memoria.

► Ciclo de instrucción



Durante la fase de ejecución la unidad de control ordena a la unidad de datos realizar la secuencia de microoperaciones requerida para realizar la instrucción. Para ello examina los bits del código máquina correspondientes al código de operación.

► La carta ASM del CS1 implementa los ciclos de todas las instrucciones del sistema.

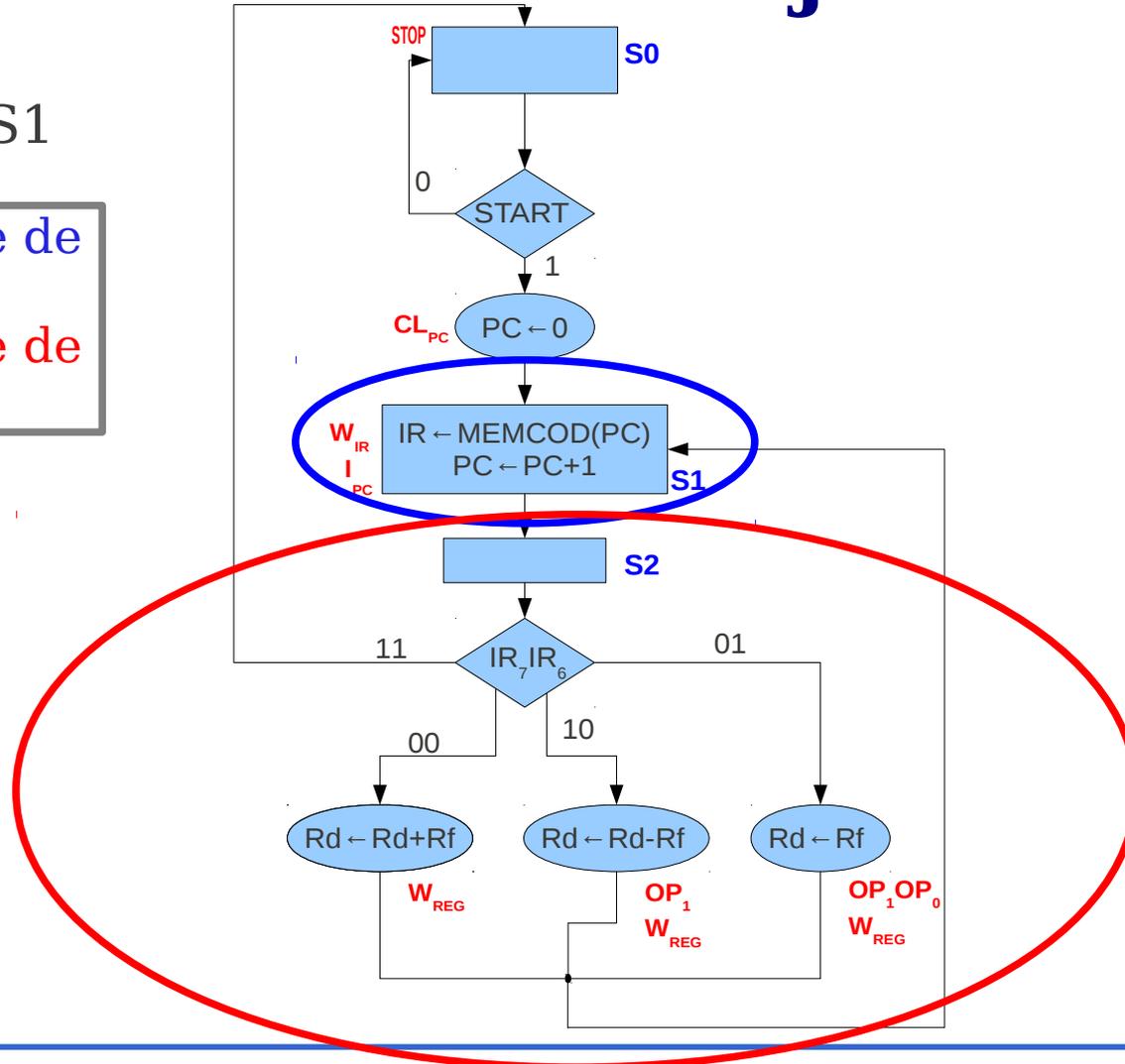
C.O: IR7 IR6	SINTAXIS	FUNCIÓN
00	ADD Rd,Rf	$Rd \leftarrow Rd+Rf$
10	SUB Rd,Rf	$Rd \leftarrow Rd-Rf$
01	MOV Rd,Rf	$Rd \leftarrow Rf$
11	STOP	NOP

Los bits IR7 e IR6 son entradas a la Unidad de Control

# Automatización en la ejecución

Carta ASM del CS1

Estados de la fase de  
Búsqueda: S1.  
Estados de la fase de  
Ejecución: S2



# Automatización en la ejecución

Ejemplo de uso del CS1: Escribir un programa que realice la siguiente operación:  $R6 \leftarrow 3R4 - 2R1$

## Programa

```
MOVE 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 --- ---

MEMORIA DE CÓDIGO