



TECNICATURA SUPERIOR EN

Telecomunicaciones

Electrónica Microcontrolada

Circuito Digital. UPPG, Unidad de Procesamiento
de Propósito General

Índice

- **Introducción:**
 - Unidades de Control/Proceso (Específicas).
 - Un Ejemplo: Máximo Común Divisor.
- **El procesador de propósito general; definición.**
- **Unidad de Proceso de PPG:**
 - Banco de Registros.
 - ALU.
 - Palabra de Control.
- **Entrada/Salida.**
- **Memoria.**
- **Unidad de Control de PPG:**
 - Secuenciamiento de Instrucciones.
 - Formato de Instrucciones.

Introducción

- **Dificultades de los Circuitos Lógicos Combinacionales:**
 - Sumador de dos números de 16 bits ($n = 16$).
 - Tabla de Verdad inviable: 65.536 filas. (¿Por qué?).
 - ¿Quién la dibuja?
 - Implementación con Not, And y Or inviable. (¿Cuántas puertas de cada tipo aproximadamente?).
 - Implementación con ROM, posible, escritura por programa.

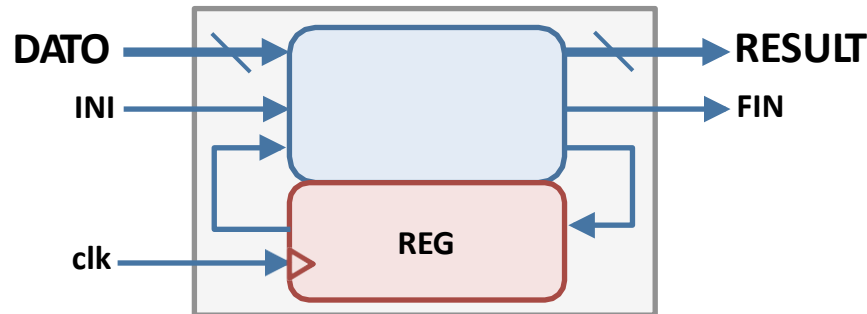
No diseñaremos los **CLC que procesan palabras de n bits** (para n típicos de 8, 16, 32 ó 64) con los métodos sistemáticos que hemos visto.

Como en el caso del sumador visto en prácticas, haremos **diseños ad-hoc con bloques multinivel**, aplicando conocimientos, inteligencia, experiencia, etc.



Introducción

- **Dificultades de los Circuitos Lógicos Secuenciales:**
 - Los CLS, en general, requieren muchos estados y los métodos de síntesis vistos anteriormente resultan inviables.
- **Un Ejemplo:**
 - Diseñar un CLS que realice la suma de una secuencia de 3 números naturales codificados en binario con 2 bits cada uno.
 - Los números llegan al sistema por la entrada DATO a razón de un número por ciclo, comenzando por el ciclo en el que la señal de entrada INI vale 1.
 - Una vez terminado el cálculo, el resultado estará disponible en la salida RESULT durante 1 ciclo, en el cual el circuito pondrá la salida FIN a 1.

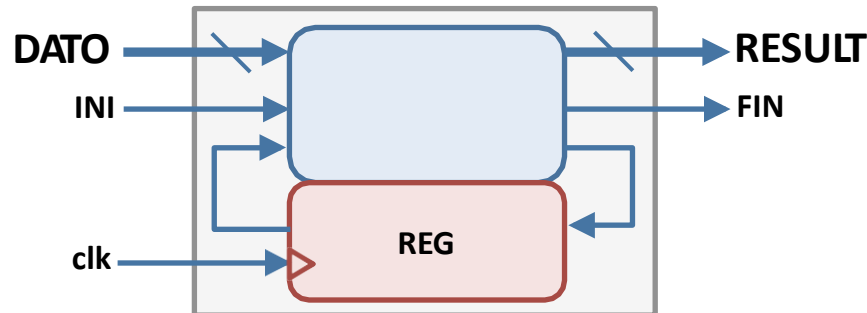


Introducción

- **Dificultades de los Circuitos Lógicos Secuenciales:**

- ¿Alguien se atreve a diseñar el grafo de estados del CLC que sume 5 números de 8 bits?
- La complejidad de un sistema como éste crece exponencialmente con n (número de bits para codificar los números). Nuestros métodos de síntesis no valen.

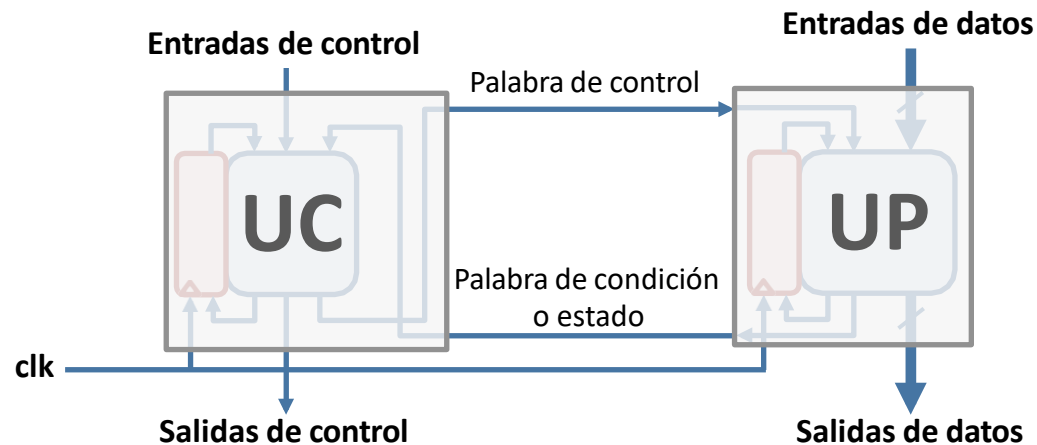
Dado que tenemos dos tipos de señales con características y funciones distintas (DATO–RESULT; INI–FIN), crearemos dos subsistemas que manejen cada grupo de manera independiente y se comuniquen entre ellos.



Introducción

- **Diseño de Sistemas Complejos:**

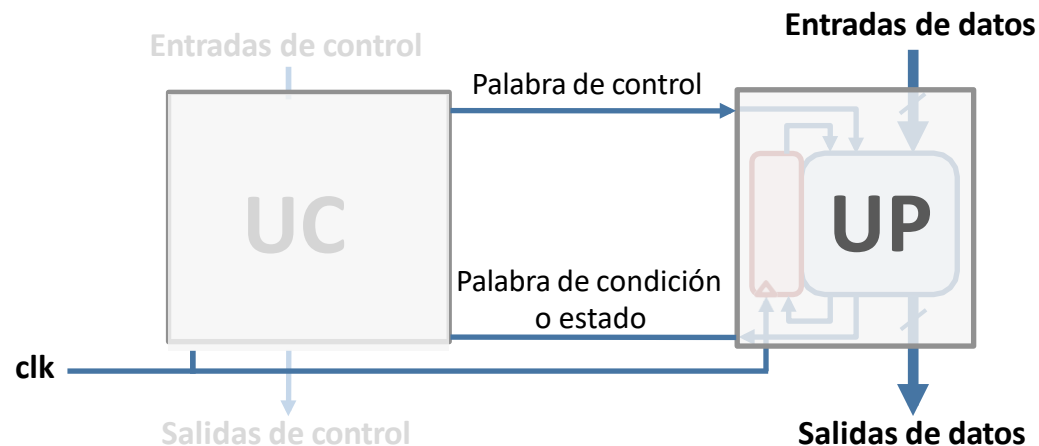
- A efectos de diseño, un sistema lógico secuencial complejo se descompone en dos subsistemas (ambos secuenciales): **Unidad de Proceso (UP)** o camino de datos (datapath) y **Unidad de Control (UC)**.
- La UP almacena y transforma (opera) los datos hasta obtener los resultados.
- La UC controla las operaciones que se realizan en la UP y su secuenciamiento correcto.



Introducción

- **Unidad de Proceso:**

- La Unidad de Proceso es un circuito secuencial que diseñaremos a nivel de bloques mediante un diseño ad-hoc.
- Diseñaremos una UP formada por elementos de almacenamiento (registros, bancos de registros, memorias) interconectados entre sí a través de bloques combinacionales (sumadores, restadores, desplazadores aritméticos, incrementadores, comparadores, ALUs, codificadores, decodificadores, multiplexores...) y puertas lógicas.

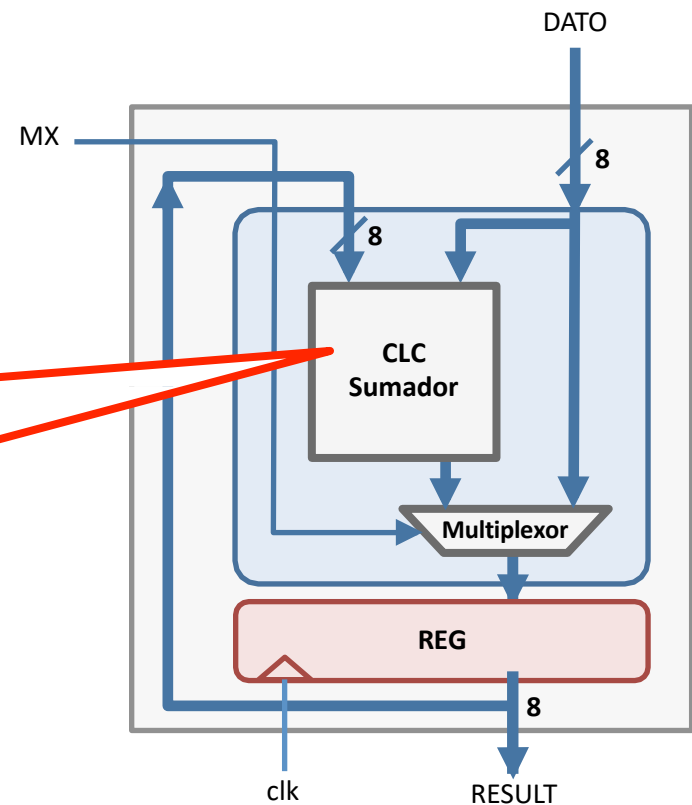


Introducción

- **Unidad de Proceso:**

- Ejemplo: Unidad de Proceso de un sumador de 4 números.

Un sumador es suficiente para sumar 4 números, ya que estos llegan uno por ciclo y podemos reutilizar el sumador (poniendo más de 1 sumador no reducimos el tiempo necesario para realizar las 4 sumas).



Introducción

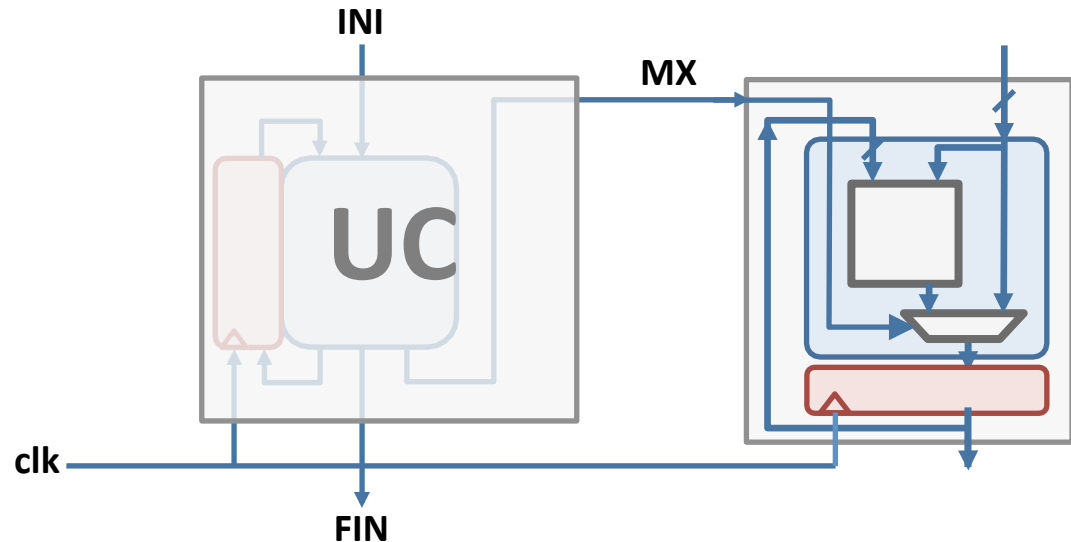
- **Unidad de Control:**

- La UC también es un CLS, de complejidad más reducida que la UP.
- Se especifica mediante un grafo de estados, según el modelo Moore:
 - Las salidas de un autómata de Moore (palabra de control que gobierna la UP y salidas de control de la UC) son función únicamente del estado actual.
 - El estado siguiente es función de las Entradas de control, de la palabra de control que llega de la UP y del estado actual.

Solo 1 bit de entrada: INI.

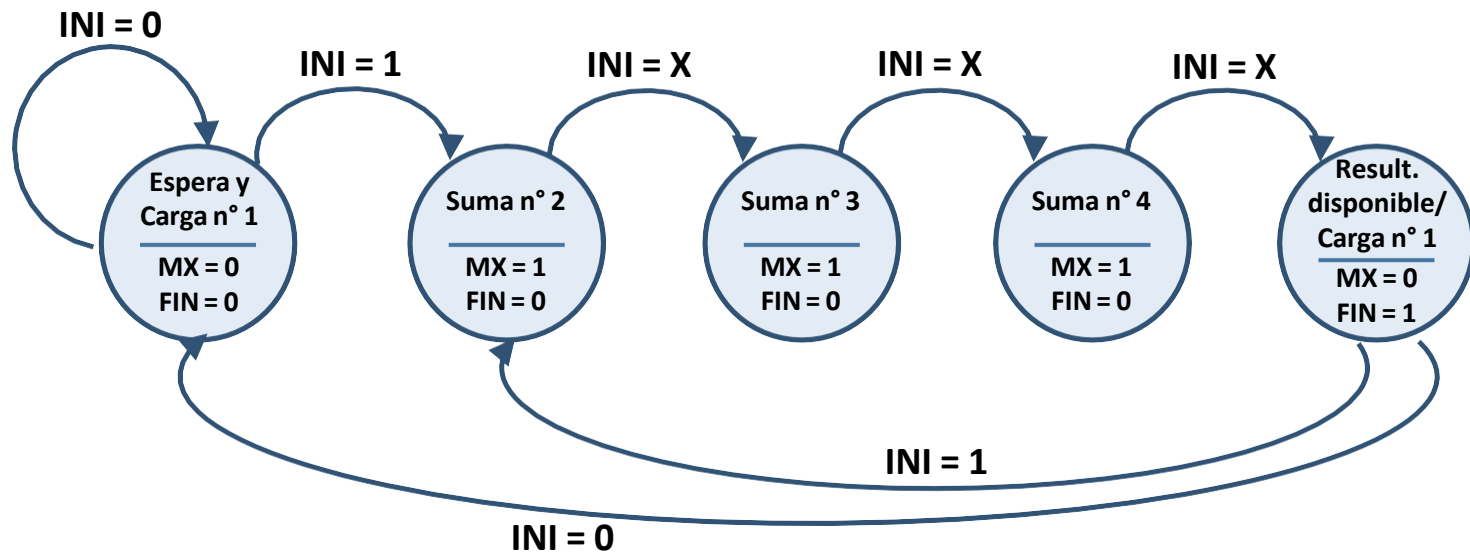
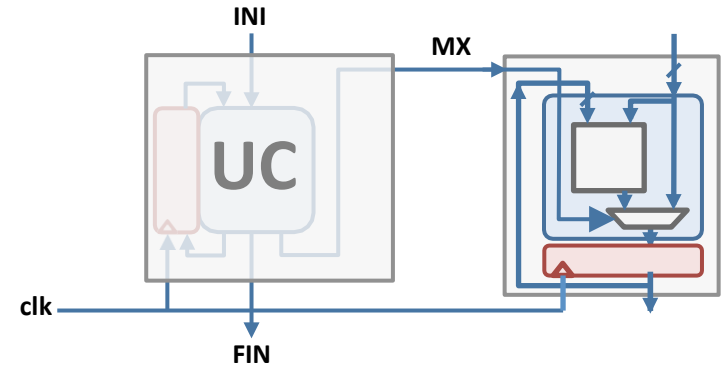
Dos salidas: Bit FIN y bit MX (palabra de control).

No necesitamos condiciones de la UP para generar el secuenciamiento correcto de las palabras de control.



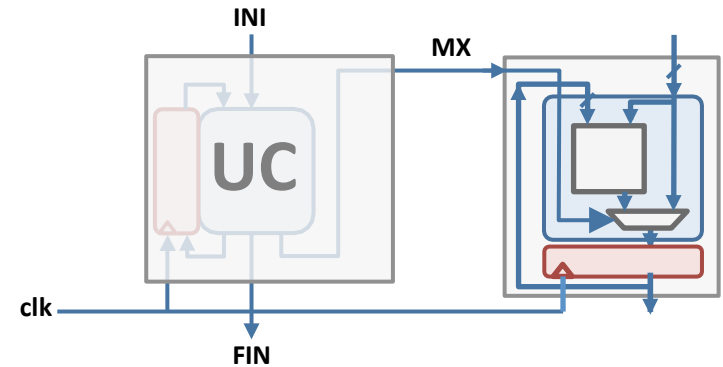
Introducción

- **Unidad de Control:**
 - El grafo de estados:



Introducción

- **Cronograma:**
 - Tiempo de ciclo suficiente.



	ciclo 1	ciclo 2	ciclo 3	ciclo 4	ciclo 5	ciclo 6
CLK						
INI	0	1	0	0	0	0
DATO	23	5	12	18	X	X
MX	0	1	1	1	0	0
Ent. Reg.	23	28	40	58	X	X
Sal. Reg.		23	28	40	58	X
FIN	0	0	0	0	1	0

Introducción

- **Ejemplo: Máximo Común Divisor:**
 - Diseñar, utilizando el algoritmo de Euclides, un circuito capaz de calcular el Máximo Común Divisor de dos números enteros X e Y codificados en complemento a 2 con 16 bits cada uno. Los números a sumar llegan al sistema cuando la entrada INI vale 1, y el resultado en la salida será válido cuando la salida FIN se ponga a 1.

Algoritmo de Euclides para MCD(X, Y):

A: = X; B: = Y;

mientras (A \neq B) **hacer:**

si (A > B) **entonces** A: = A - B;

en caso contrario B: = B - A;

fin-mientras

MCD: = A.

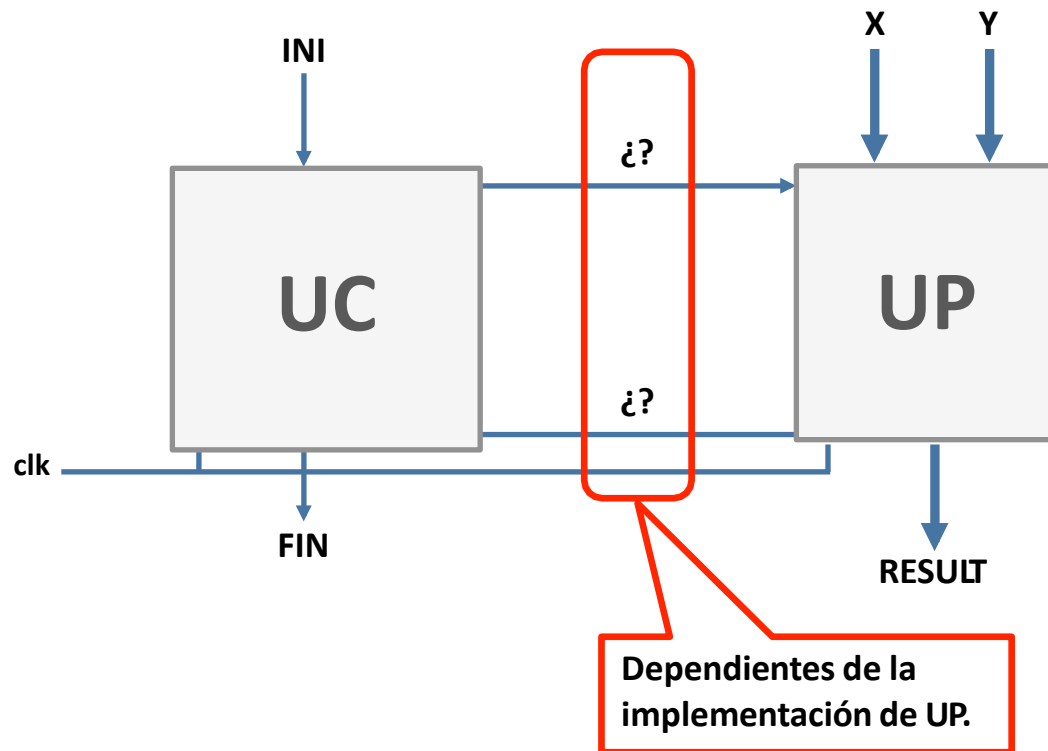
Introducción

- **Ejemplo: Máximo Común Divisor:**
 - Paso 1: definir las entradas/salidas de la UC y UP.



Introducción

- **Ejemplo: Máximo Común Divisor:**
 - Paso 1: definir las entradas/salidas de la UC y UP.



Introducción

- **Ejemplo: Máximo Común Divisor:**
 - Paso 2: diseñar la Unidad de Proceso.

Algoritmo de Euclides para $MCD(X, Y)$:

$A := X; B := Y;$

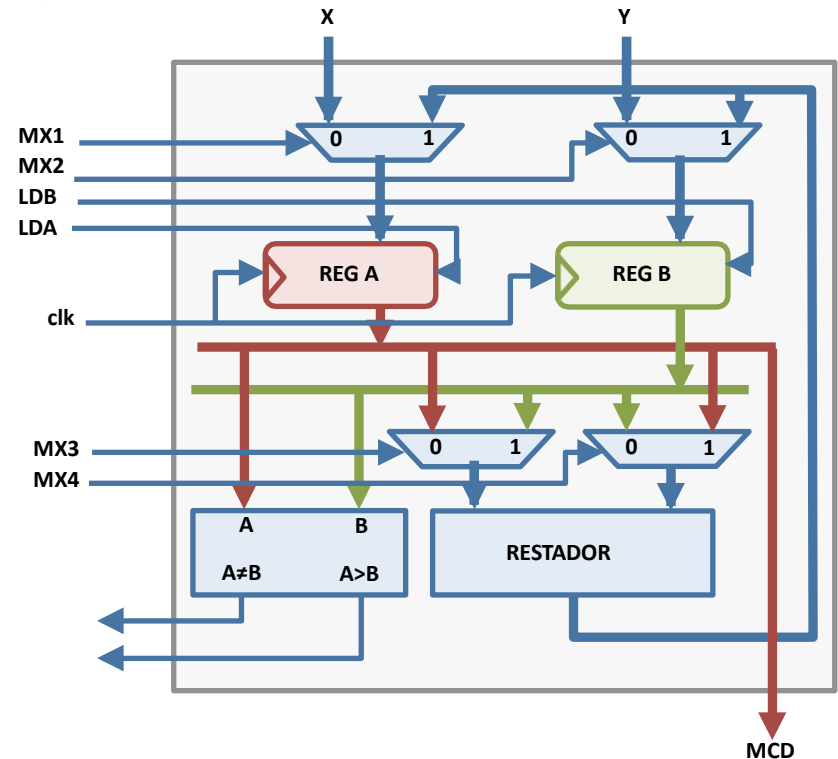
mientras $(A \neq B)$ **hacer:**

si $(A > B)$ **entonces** $A := A - B;$

en caso contrario $B := B - A;$

fin-mientras

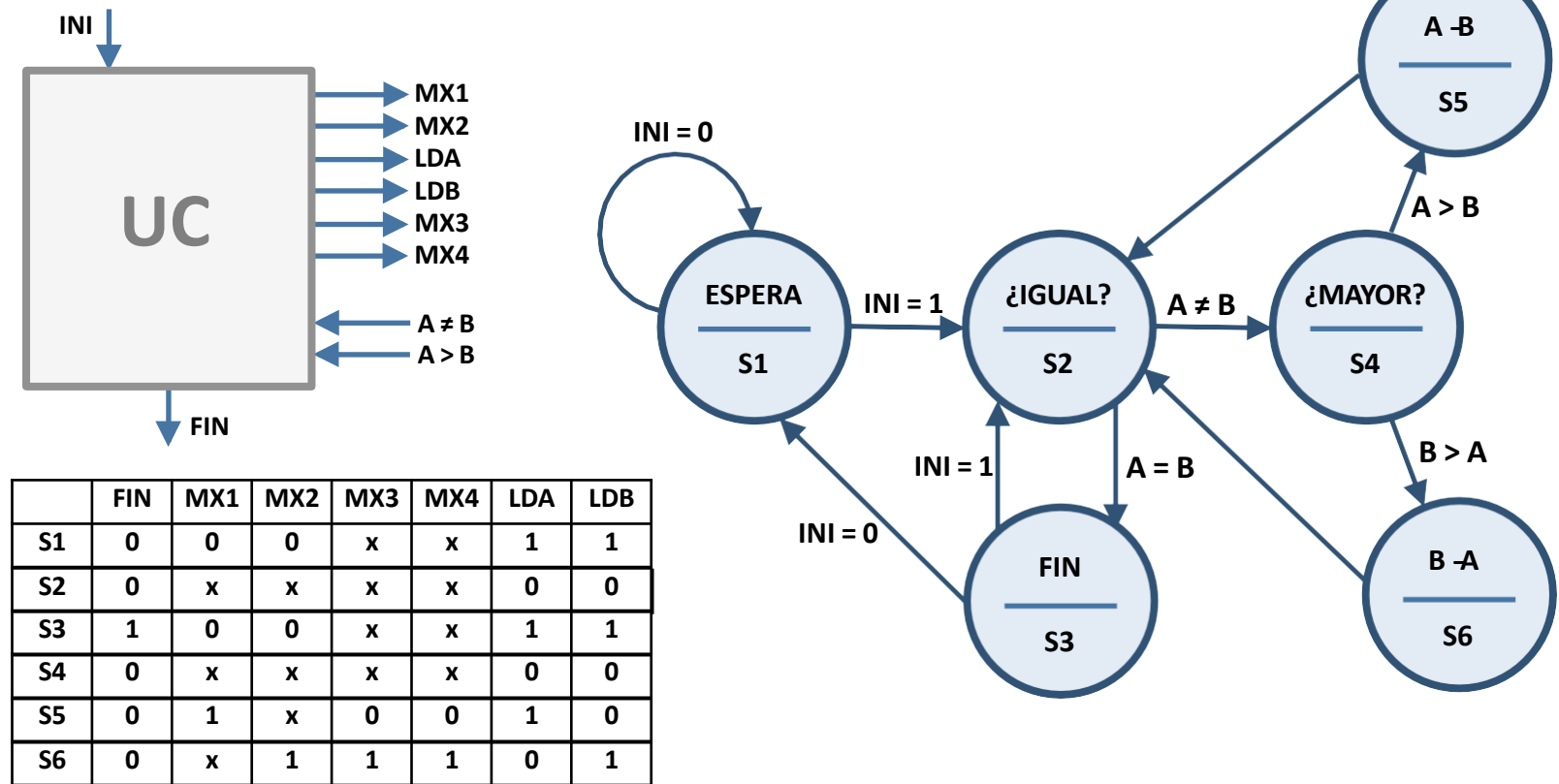
$MCD := A.$



Introducción

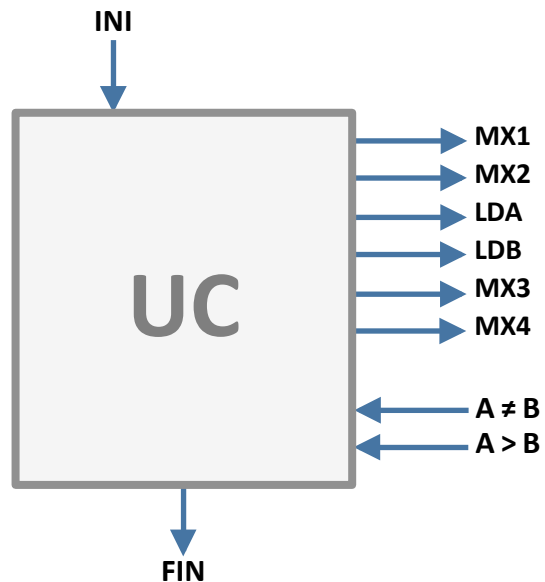
- **Ejemplo: Máximo Común Divisor**

- Paso 3: implementar el grafo de estados de la UC.



Introducción

- **Ejercicio:**
 - Dado el grafo de estados del MCD, implementar su UC con una ROM y conectar dicho circuito a la UP diseñada.



Índice

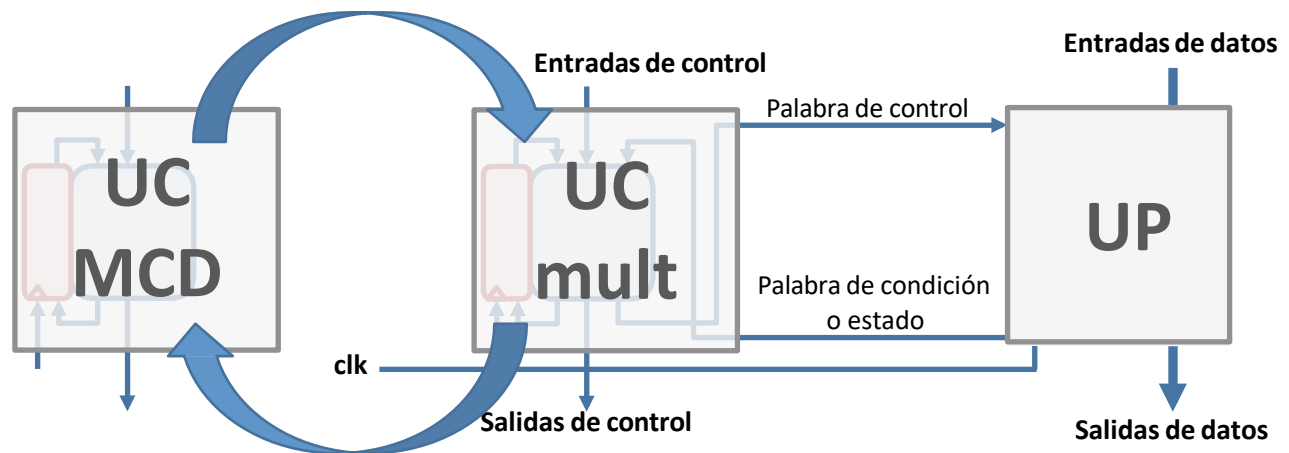
- **Introducción:**
 - Unidades de Control/Proceso (Específicas).
 - Un Ejemplo: Máximo Común Divisor.
- **El procesador de propósito general; definición.**
- **Unidad de Proceso de PPG:**
 - Banco de Registros.
 - ALU.
 - Palabra de Control.
- **Entrada/Salida.**
- **Memoria.**
- **Unidad de Control de PPG:**
 - Secuenciamiento de Instrucciones.
 - Formato de Instrucciones.

Procesador de Propósito General

- El núcleo de un computador es un **procesador de propósito general**: un mismo circuito que sirve para resolver diferentes problemas.
- Únicamente se cambia la secuencia de órdenes (el **programa**) que se encuentra almacenado en memoria.
- Pasos:
 - Definiremos una Unidad de Proceso General (UPG) que puede servir para resolver diferentes problemas con sólo cambiar la unidad de control (específica para cada problema).
 - Transformaremos la UC hasta obtener un diseño genérico “programable”, generando las señales de control adecuadas para la UPG diseñada.
- Objetivo final: un procesador de propósito general (tanto UP como UC).

Unidad de Proceso General (UPG)

- A una **UP general** podremos conectar diferentes Unidades de control, de tal forma que pueda realizar las diferentes tareas sin necesidad de ser modificada.
- Una **UP general** debe tener un conjunto de **registros**, una o varias **unidades funcionales** (circuitos combinacionales que realizan diferentes cálculos) y una **interconexión** que se pueda modificar (multiplexores, demultiplexores) dinámicamente en cada ciclo.



Unidad de Proceso General (UPG)

- Registros + Unidades Funcionales + Interconexión

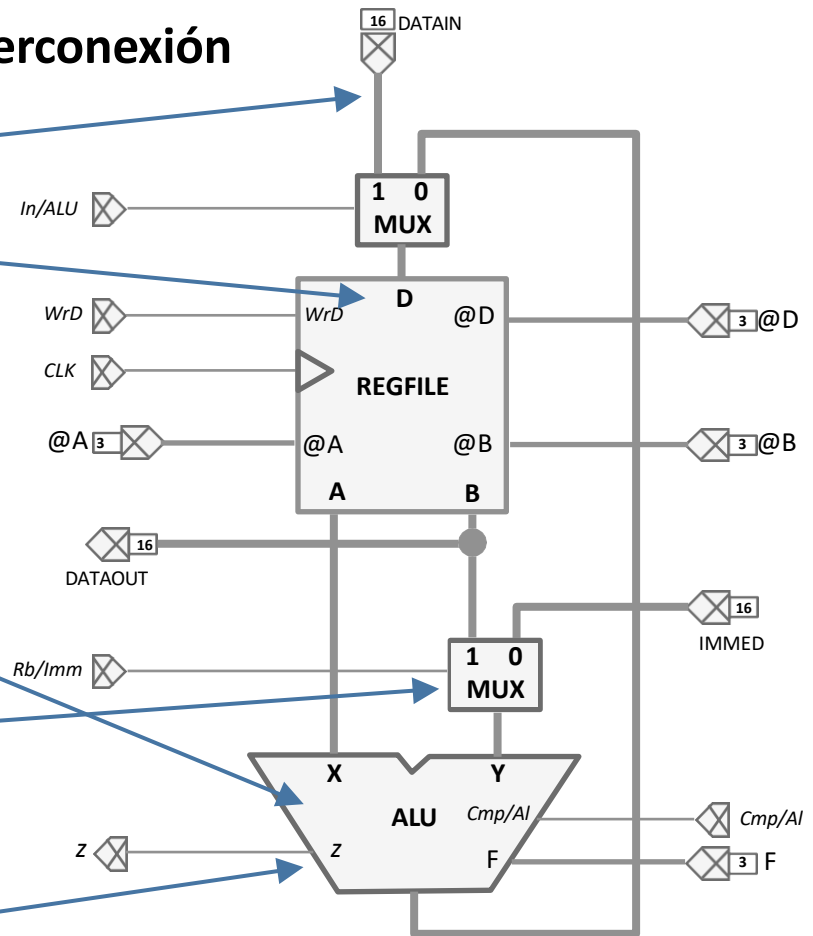
Ancho de los buses: 16 bits.

Banco de registros: 8 registros de 16 bits de tamaño cada uno (Como son 8, 3 bits de dirección @).

Unidad Aritmético--Lógica: 8 posibles operaciones aritméticas y 5 de Comparación (Cmp/Al y F determinan la operación).

Las operaciones de la ALU se llevan a cabo sobre dos registros ó un registro y una constante (**IMMED**).

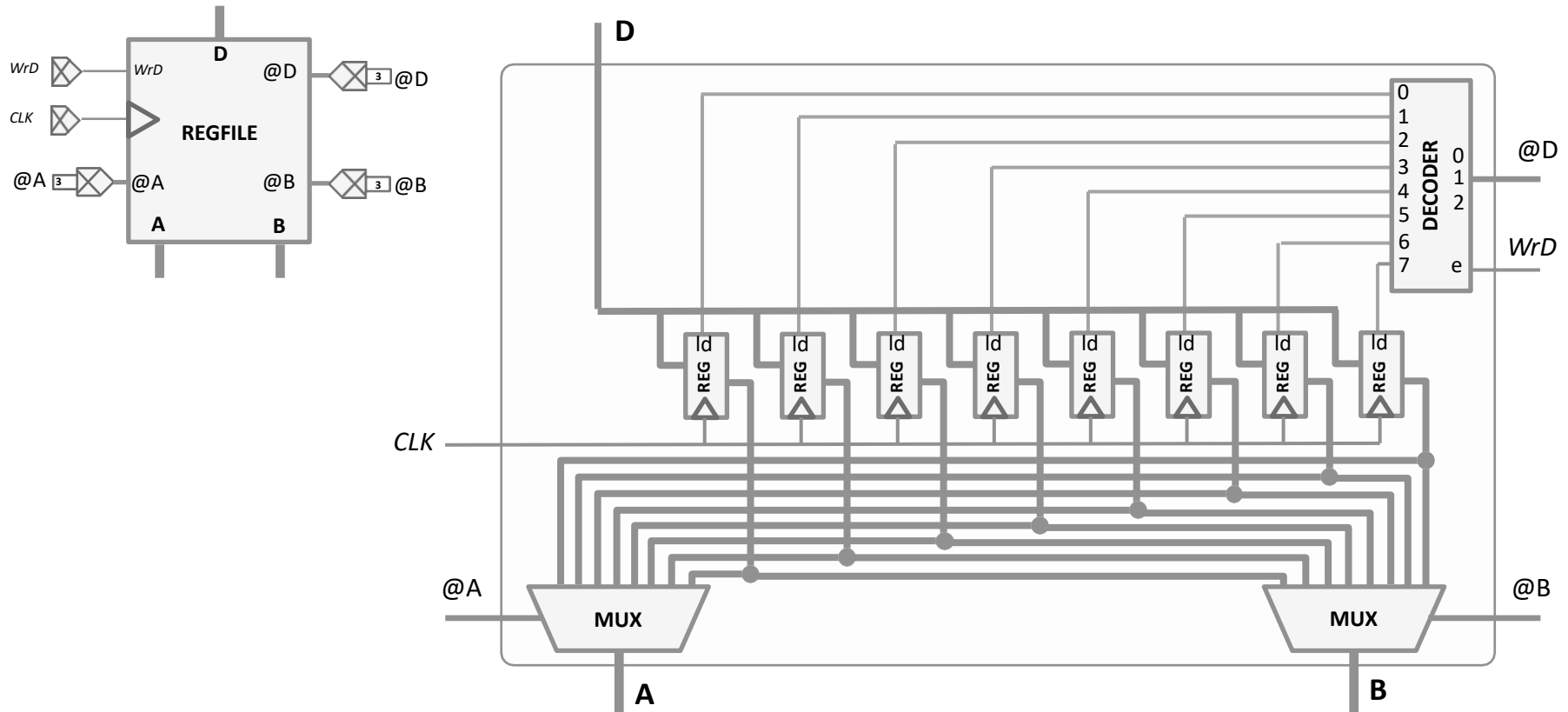
El **bit z** vale 1 cuando los 16 bits de salida de la ALU valen 0 (en caso contrario vale 0).



Unidad de Proceso General (UPG)

- Banco de Registros:**

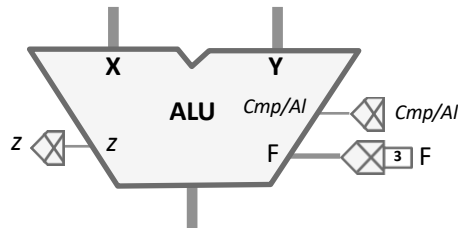
- 8 Regs de 16 bits, con 1 puerto de escritura (D) y dos de lectura (A y B).



Unidad de Proceso General (UPG)

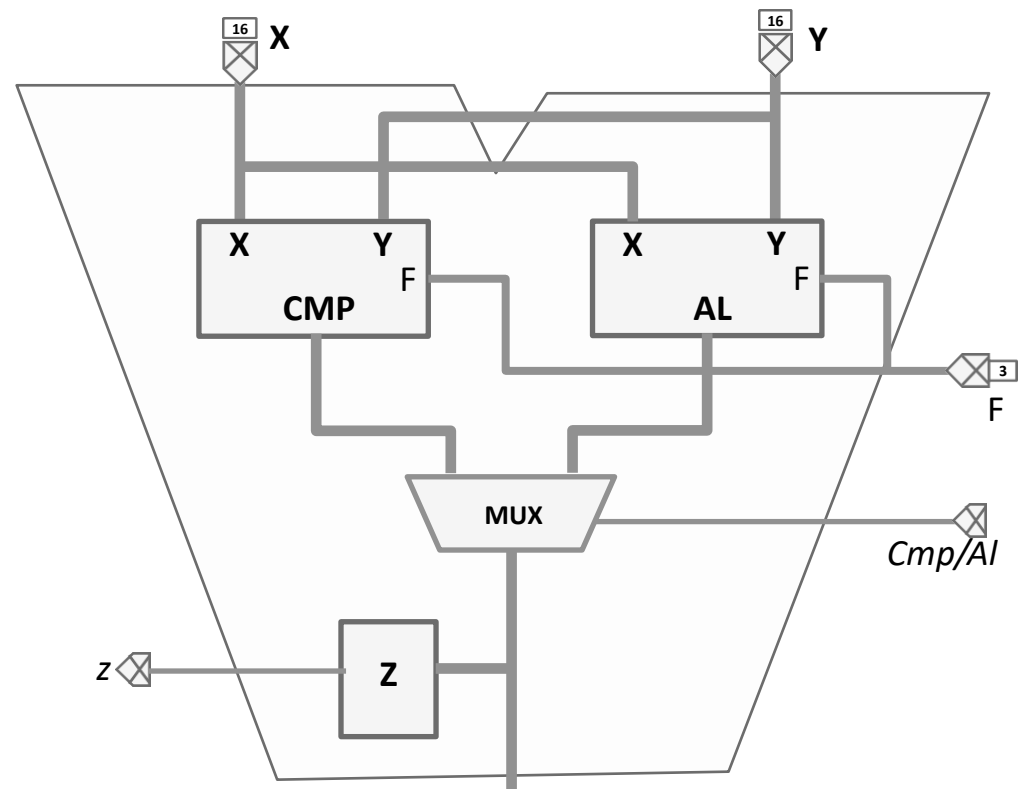
- Unidad Aritmético--Lógica:**

- Datos de 16 bits, 13 Operaciones (4 Aritméticas, 4 Lógicas, 5 Comparación).



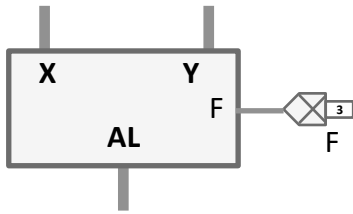
El tipo de operación es seleccionado con los 3 bits de la entrada F y con la entrada Cmp/Al:

OP	Cmp/Al=1	Cmp/Al=0
000	CMPLT(X,Y)	AND(X,Y)
001	CMPLT(X,Y)	OR(X,Y)
010	—	XOR(X,Y)
011	CMPEQ(X,Y)	NOT(X)
100	CMPLTU(X,Y)	ADD(X,Y)
101	CMPLTU(X,Y)	SUB(X,Y)
110	—	SHA(X,Y)
111	—	SHL(X,Y)

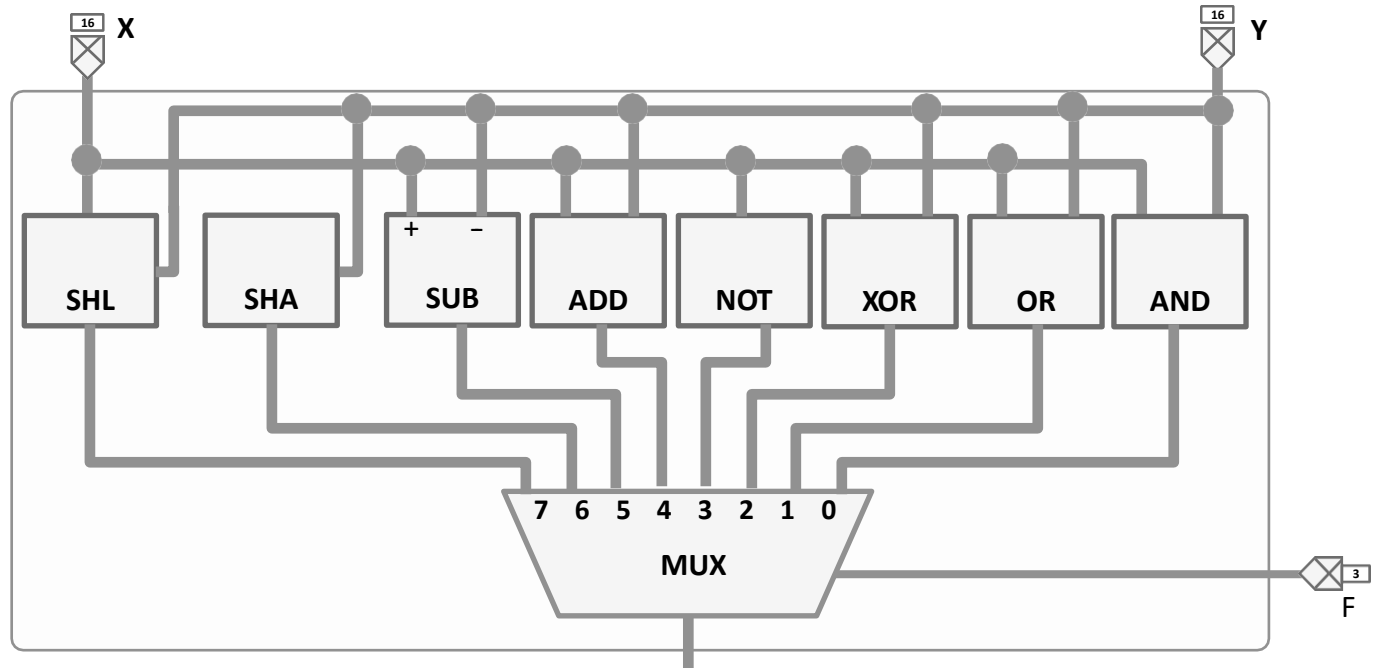


Unidad de Proceso General (UPG)

- **Unidad Aritmético--Lógica: Operaciones Aritméticas/Lógicas:**
 - La salida de la ALU es el valor de la función seleccionada por F, aplicado a X e Y.



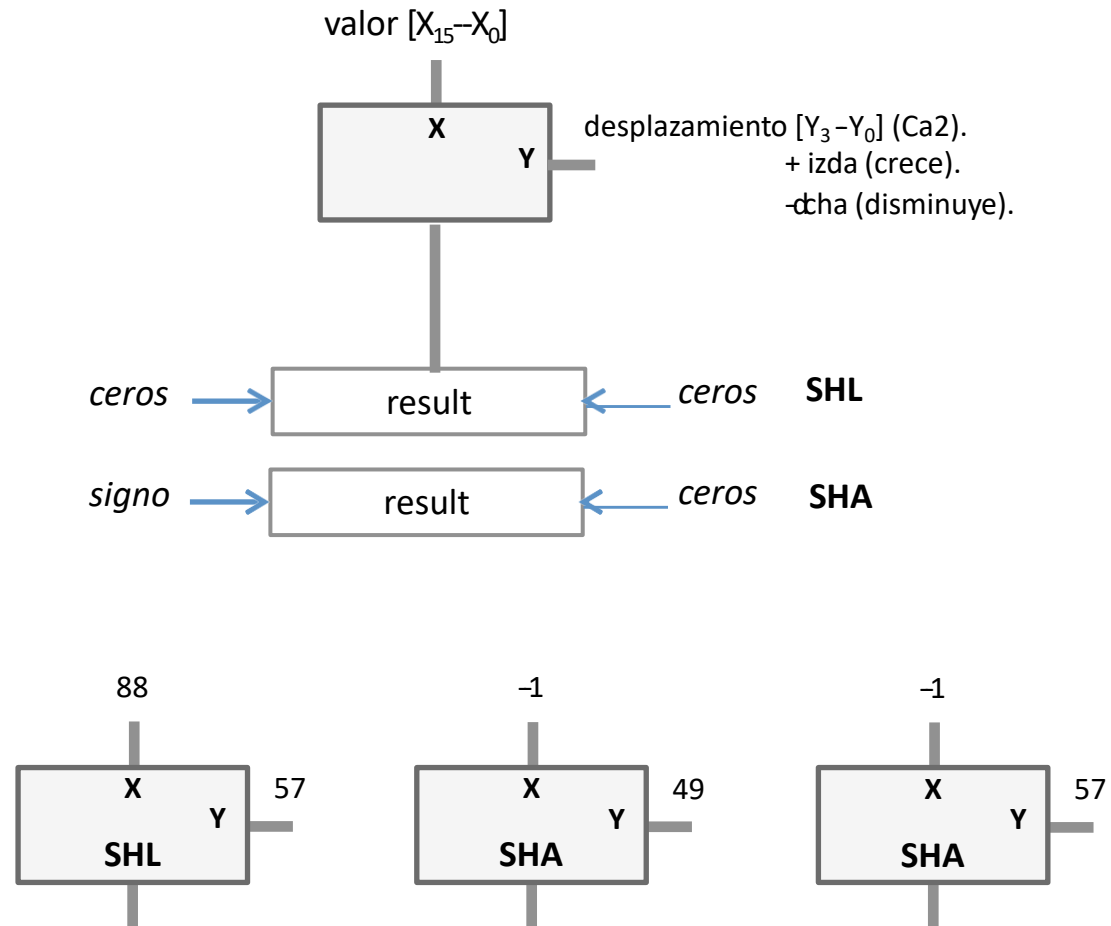
F	W
000	AND(X,Y)
001	OR(X,Y)
010	XOR(X,Y)
011	NOT(X)
100	ADD(X,Y)
101	SUB(X,Y)
110	SHA(X,Y)
111	SHL(X,Y)



Unidad de Proceso General (UPG)

- Operación Shift.

F	W
000	AND(X,Y)
001	OR(X,Y)
010	XOR(X,Y)
011	NOT(X)
100	ADD(X,Y)
101	SUB(X,Y)
110	SHA(X,Y)
111	SHL(X,Y)

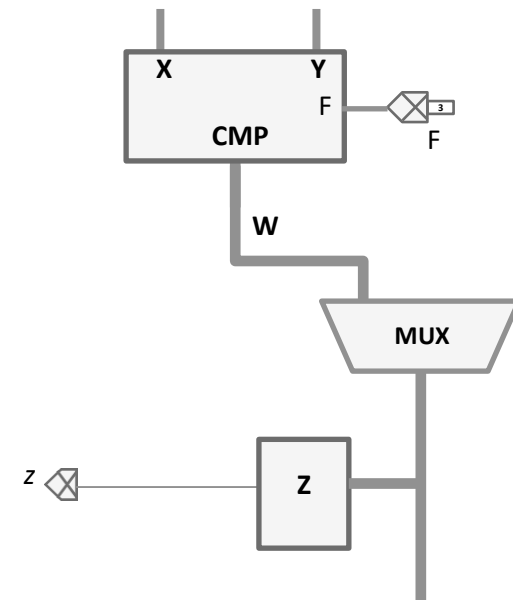


Unidad de Proceso General (UPG)

- Unidad Aritmético--Lógica: Comparaciones:**

- Salida de CMP: True ó False (codificable con 1 bit).
- Utilizo el bit 0: TRUE: $W(b_0) = 1$ FALSE: $W(b_0) = 0$. El resto de bits de salida a 0 ($W(b_i) = 0$).

OP	Cmp/Al = 1	Cmp/Al = 0
000	CMPLT(X,Y)	Less Than (<i>signed</i>): if ($X < Y$) $W=1$; else $W=0$;
001	CMPLE(X,Y)	Less Than or Equal (<i>signed</i>): if ($X \leq Y$) $W=1$; else $W=0$;
010	—	—
011	CMPEQ(X,Y)	Equal: if ($X == Y$) $W=1$; else $W=0$;
100	CMPLTU(X,Y)	Less Than (<i>unsigned</i>): if ($X < Y$) $W=1$; else $W=0$;
101	CMPLEU(X,Y)	Less Than or Equal (<i>unsigned</i>): if ($X \leq Y$) $W=1$; else $W=0$;
110	—	—
111	—	—



El valor de z indicará a la Unidad de Control el resultado de nuestras comparaciones.



Unidad de Proceso General (UPG)

- **Posibles acciones (en 1 ciclo) de la Unidad General de Proceso:**

1. Operaciones con 2 registros: leer dos registros, operarlos con alguna de las 12 funciones de 2 operandos de la ALU y dejar el resultado en otro registro al final del ciclo. Ejemplos:

$R6 = R3 + R5$ (ADD R6, R3, R5).

if ($R1 < R5$) then $R3 = 1$ else $R3 = 0$ (CMPLTU R3, R1, R5).

* El segundo operando puede ser un número codificado en complemento a 2 con 16 bits que entra por IMMED:

$R7 = R1 - 1$ (ADDI R7, R1, -1).

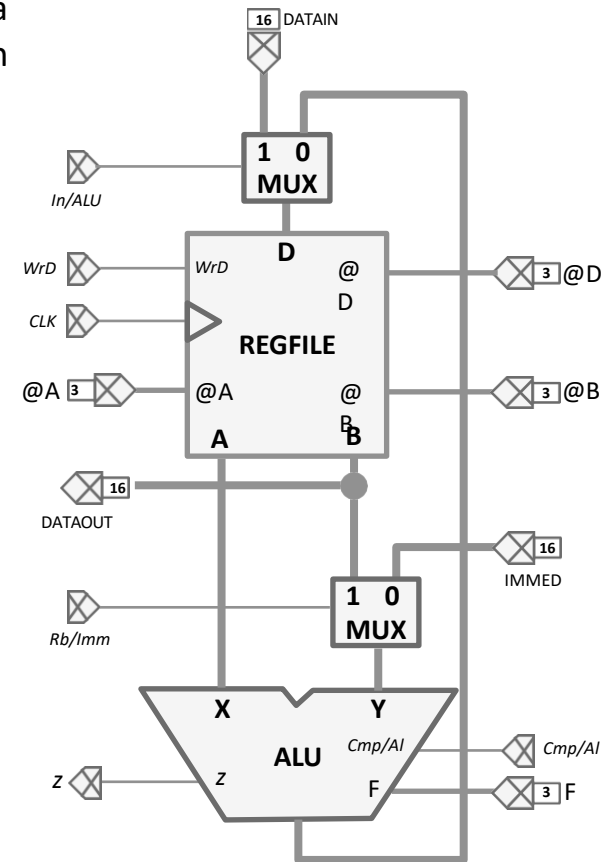
2. Operaciones con 1 registro: leer un registro por el bus A, operarlo con la función NOT:

$R4 = !R2$ (NOT R4, R2).

3. Entrada de Datos: escribir un registro, al final del ciclo, con la información presente durante ese ciclo en DATAIN:

$R2 = \text{valor de DATAIN}$ (IN R2).

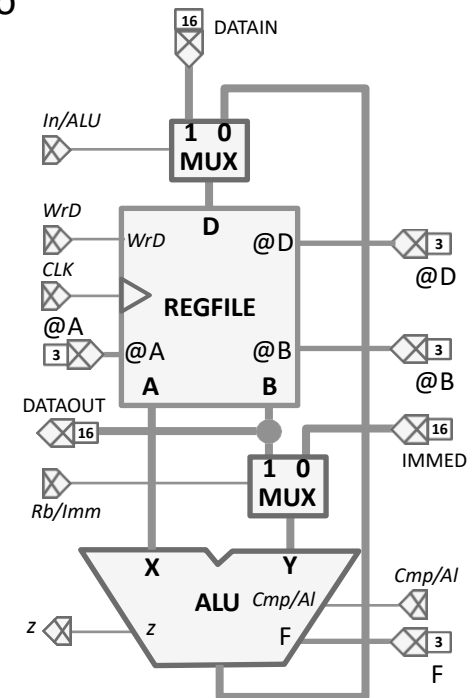
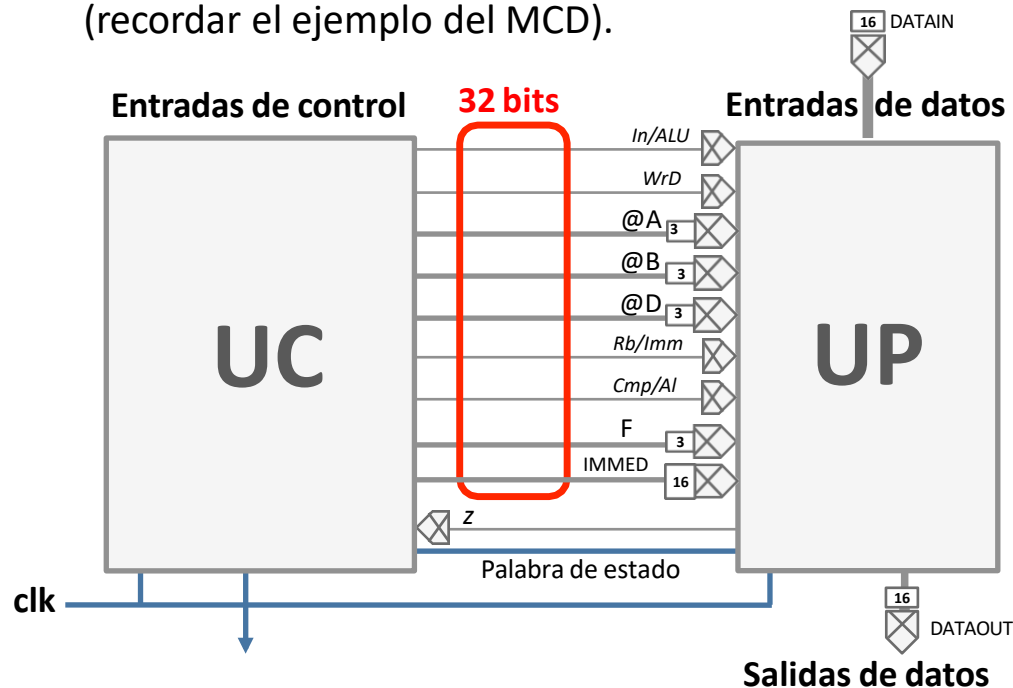
4. Salida de Datos: contenido de un registro presente en DATAOUT:
valor de DATAOUT = R4 (OUT R4).



Unidad de Proceso General (UPG)

- **La Palabra de Control:**

- No perder la perspectiva. Seguimos trabajando con el modelo utilizado al comienzo del tema (Unidad de Control \leftrightarrow Unidad de Proceso).
- Para que la Unidad de Proceso realice una acción en un ciclo determinado, hay que proporcionar a la UP la palabra de control adecuada durante ese ciclo (recordar el ejemplo del MCD).



Unidad de Proceso General (UPG)

- **Mnemotécnicos y Palabras de Control:**

- Interpretar las palabras de control desde el punto de vista «humano» es complejo.
 - Ejemplo: ¿qué hace la palabra de control **0111001101001101xxxxxxxxxxxxxxxx**?
- Utilizaremos un sistema de representación más fácil de interpretar (mnemotécnicos).
- El formato de dichos mnemotécnicos será el siguiente:

1. Operaciones con 2 registros:

Operación	Reg. Destino	Reg. Origen	Reg. Origen
ADD	R6,	R3,	R5

2. Operaciones con 1 registro y 1 inmediato:

Operación	Reg. Destino	Reg. Origen	Inmediato
ADD	R6,	R3,	2

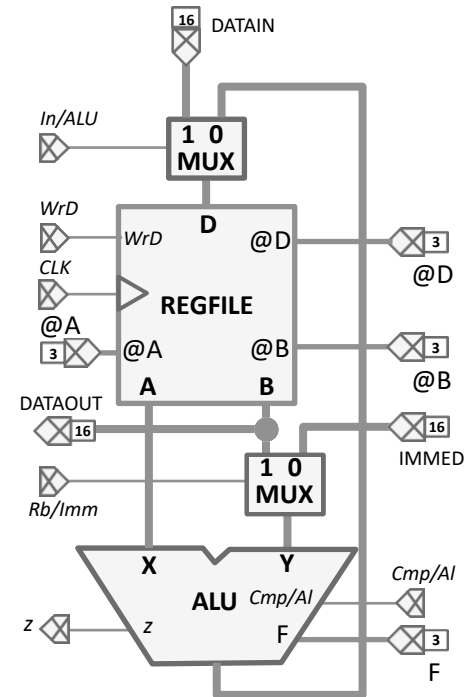
3. Operaciones con 1 registro:

Operación	Reg. Destino	Reg. Origen
NOT	R6,	R3

4. Entrada / Salida de datos:

Operación	Registro
IN	R6

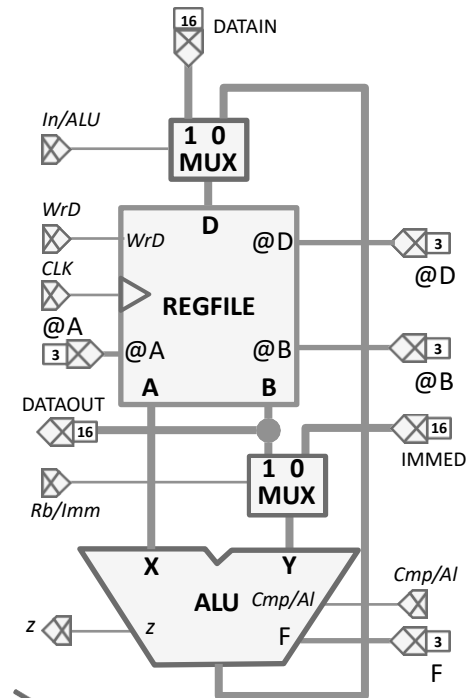
Estos mnemotécnicos también se denominan **Instrucciones**.



Unidad de Proceso General (UPG)

- **Mnemotécnicos y Palabras de Control**

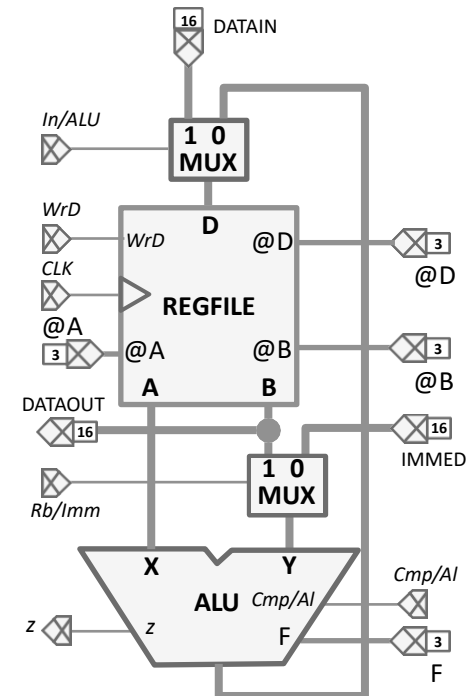
- A partir de un mnemotécnico debemos ser capaces de generar la palabra de control (y viceversa).
- Ejemplo: Para que la UPG haga NOT R4, R2 la palabra de control será:

[illegible]

Unidad de Proceso General (UPG)

- **Ejercicio:**
 - Obtener las palabras de control de la UPG para las siguientes instrucciones:
 ADD R6, R3, R5
 CMPLTU R3, R1, R5
 ADDI R7, R1, -1
 IN R2
 OUT R4

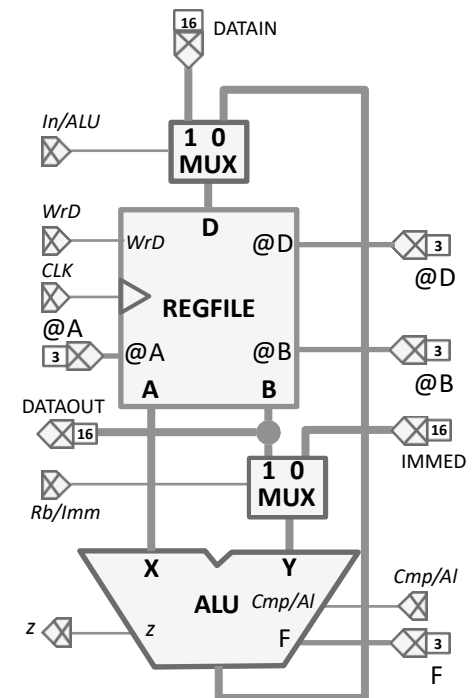
Mnemotécnico	In/ALU	WrD	@D			@A			Cmp/Al	F			RB/I	@B			IMMED $b_{15}b_{14}b_{13}...b_1b_0$
			b_2	b_1	b_0	b_2	b_1	b_0		b_2	b_1	b_0		b_2	b_1	b_0	



Unidad de Proceso General (UPG)

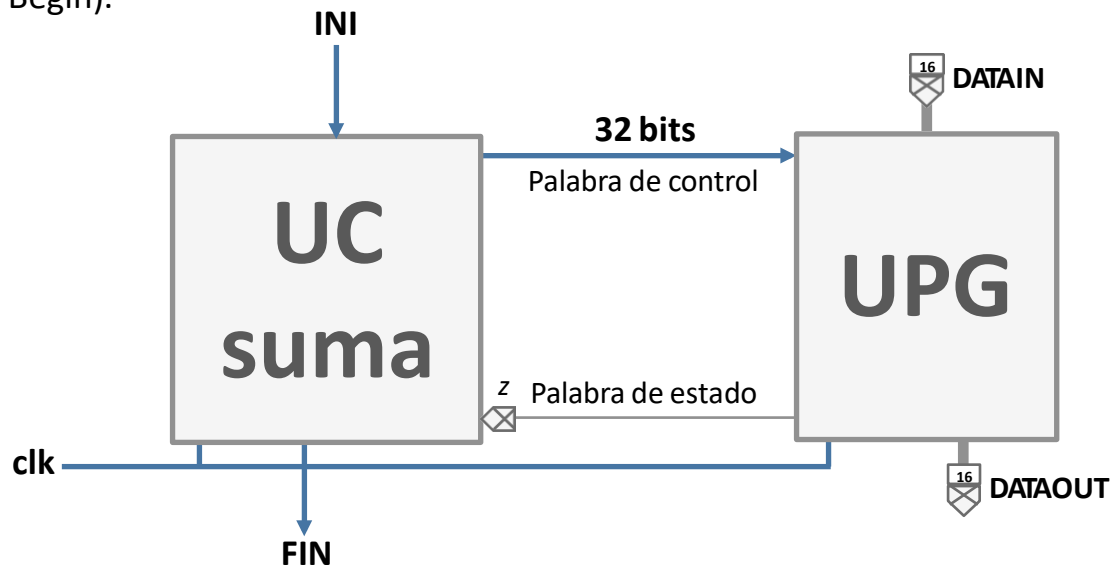
- **Observaciones:**

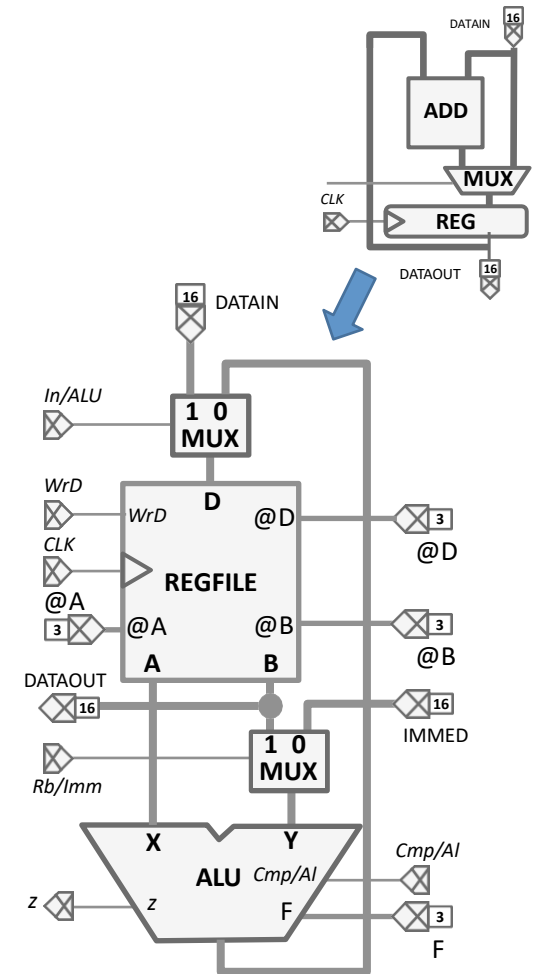
- De los 4 tipos de acciones solo en una (OUT) no es necesario escribir al final del ciclo en el banco de registros.
- En ocasiones puede resultar interesante que la ALU opere sin escribir el resultado. Por ejemplo, la acción `ANDI ¬R3, 0x8000` indicará si el bit 15 de R3 vale 0 si z vale 1.
- Algunas operaciones pueden realizarse de forma simultánea. Por ejemplo, `SUB R1, R2, R3` y `OUT R2`. (La mayoría no pueden. Ej.: `ADD R6, R3, R5` y `IN R2`).



Unidad de Proceso General (UPG)

- **Ejemplo de uso de la UPG:**
 - Sumar, con la UPG, 4 números que lleguen por la entrada DATAIN.
 - La Unidad de Proceso ya está diseñada, hay que diseñar la Unidad de Control.
 - La UC debe generar en cada ciclo:
 - La palabra de control (32 bits) y las salidas de control (bit END).
 - El estado siguiente en función de: Estado Actual, palabra de estado (bit z) y entradas de control (bit Begin).

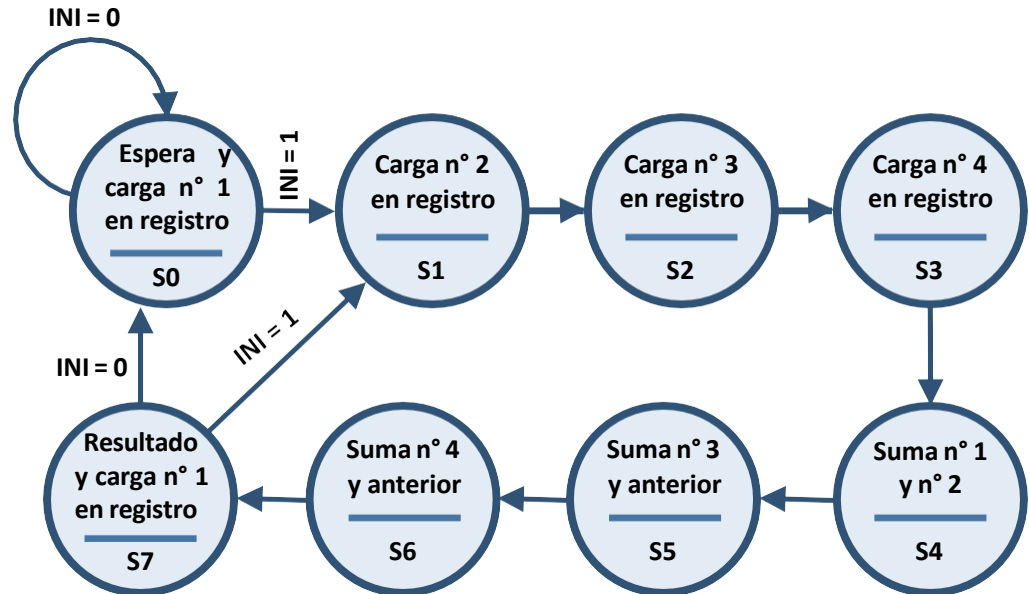




Unidad de Proceso General (UPG)

- Ejemplo de uso de la UPG:
 - Sumar, con la UPG, 4 números que lleguen por DATAIN.
 - Grafo de Estados:

	Palabra de Control	FIN
S0	IN R1	0
S1	IN R2	0
S2	IN R3	0
S3	IN R4	0
S4	ADD R5, R1, R2	0
S5	ADD R5, R3, R5	0
S6	ADD R5, R4, R5	0
S7	OUT R5 // IN R1	1



Unidad de Proceso General (UPG)

- Ejercicio 1:**

- Dadas las instrucciones de la UC necesarias para llevar a cabo la suma de 4 números (Diapositiva anterior), Determina la Palabra de Control de S1, S5 y S7.

Mnemotécnico	In/ALU	WrD	@D			@A			Cmp/A	F			RB/I	@B			IMMED $b_{15}b_{14}b_{13}...b_1b_0$
			b_2	b_1	b_0	b_2	b_1	b_0		b_2	b_1	b_0		b_2	b_1	b_0	

- Ejercicio 2:**

- Dado el grafo de estados obtenido para la suma de 4 números con una UPG, obtener la implementación de la UC mediante una ROM con el número de entradas y salidas que consideres adecuado.

Unidad de Proceso General (UPG)

- **Ejercicio 3:**

- Suponiendo que el estado (contenido de sus registros) de la UPG al inicio de un ciclo es: $R0 = 0/R1 = 830/R2 = 3456/R3 = 16/R4 = R5 = 4096/R6 = 1/R7 = 234$. Determinar cómo se modifican sus registros en los siguientes casos:

```
AND R3, R1, R5
ADD R1, R2, R3
ADDI R4, R7, -1
OUT R5 // IN R6
CMPEQ -, R3, R2
SUBI -, R2, 1
```

- **Ejercicio 4:**

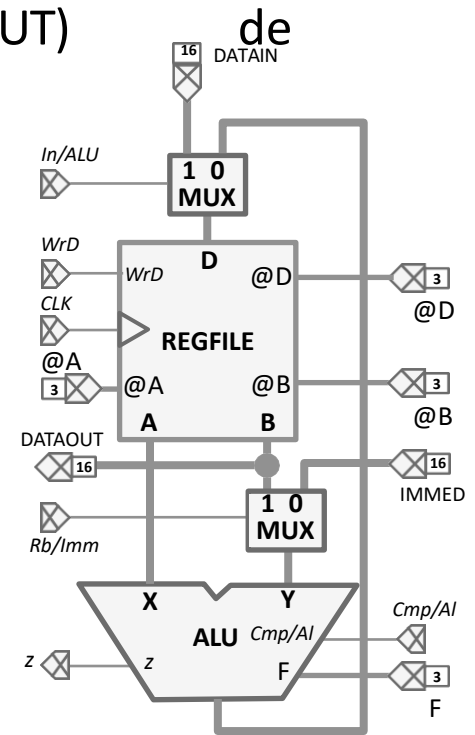
- Diseñar, utilizando el algoritmo de Euclides, un circuito capaz de calcular el Máximo Común Divisor de dos números enteros X e Y codificados en complemento a 2 con 16 bits cada uno. Los números a sumar llegan al sistema cuando la entrada INI vale 1, y el resultado en la salida será válido cuando la salida FIN se ponga a 1.

Índice

- **Introducción:**
 - Unidades de Control/Proceso (Específicas).
 - Un Ejemplo: Máximo Común Divisor.
- **El procesador de propósito general; definición.**
- **Unidad de Proceso de PPG:**
 - Banco de Registros.
 - ALU.
 - Palabra de Control.
- **Entrada/Salida.**
- **Memoria.**
- **Unidad de Control de PPG:**
 - Secuenciamiento de Instrucciones.
 - Formato de Instrucciones.

Entrada / Salida

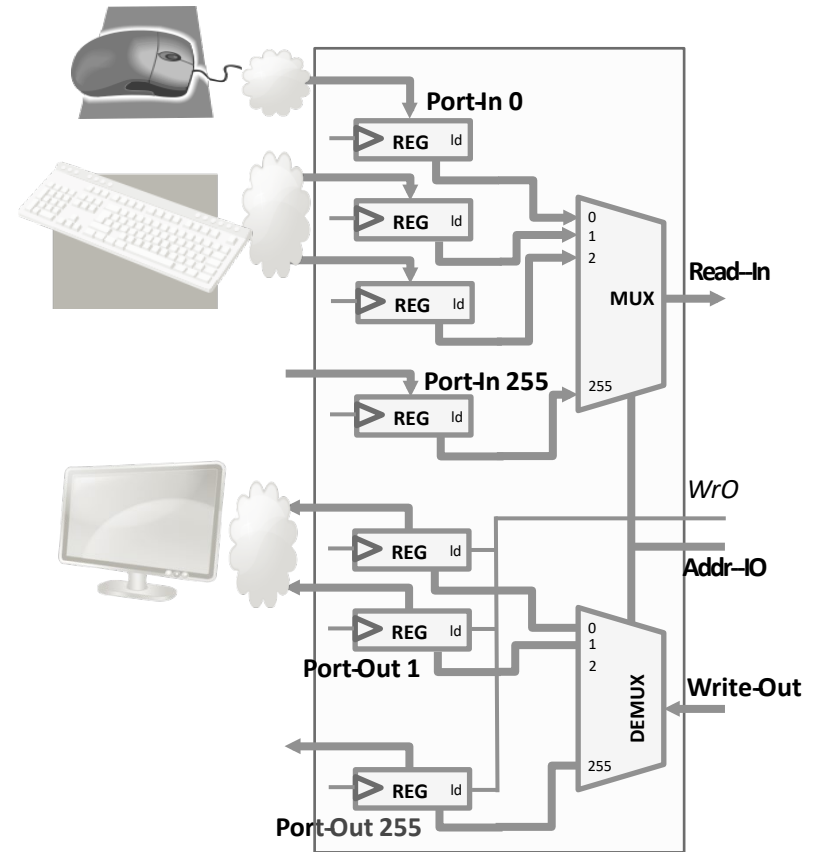
- El objetivo es **introducir y extraer datos** del procesador (**teclado, ratón, pantalla**). ¿De dónde viene DATAIN?, ¿dónde va DATAOUT?
- Hay que buscar una forma de conectar dispositivos muy distintos al único Bus de entrada (DATAIN) y al único Bus de salida (DATAOUT) de nuestro procesador.
- Los periféricos se comunican con el procesador a través de registros. Añadiremos un banco de registros adicional que se conectará a DATAIN y DATAOUT.
- El número de registros de E/S será mucho mayor que los del banco de registros del procesador.



Entrada / Salida

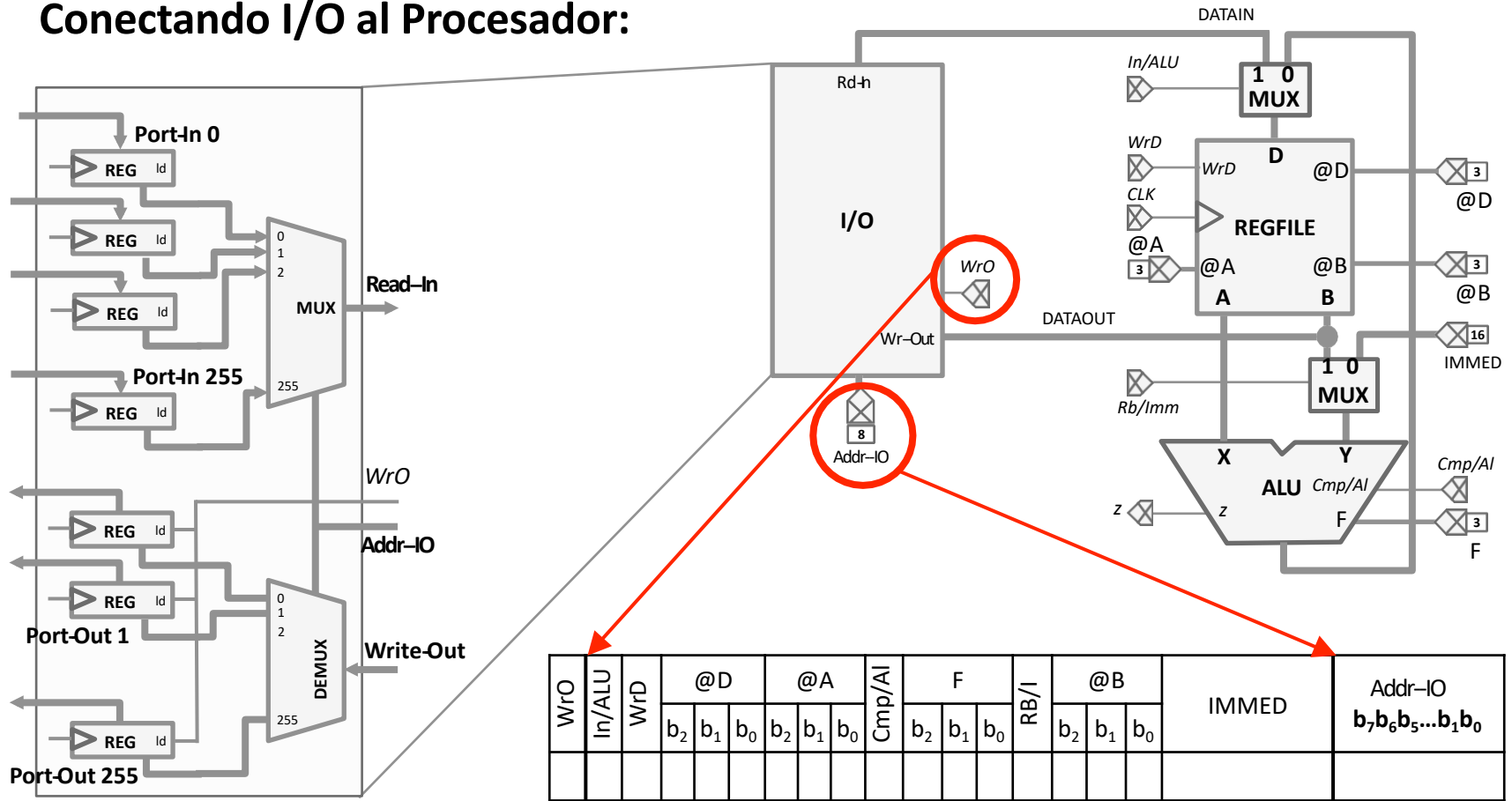
- **Nuevo Banco de Registros:**

- Dos tipos de Registros, de entrada y de salida.
- Los periféricos se conectan a los registros, de acuerdo a su tipo (Teclado--PortIn, Pantalla--PortOut).
- Los periféricos escriben y leen los datos de dichos registros
- Utilizamos un único bus de direccionamiento (Addr-IO), por lo que el procesador no puede leer y escribir de I/O en el mismo ciclo.



Entrada / Salida

- Conectando I/O al Procesador:



La palabra de control cambia (crece)

Entrada / Salida

- **Conectando I/O al Procesador:**

- La nueva palabra de control tendrá 41 bits. Ahora, la unidad de control tiene que generar 9 bits más (8 Addr–IO y 1 para WrO).
- Ahora, en las instrucciones (mnemotécnico) empleadas para introducir y extraer datos del procesador, habrá que especificar el Registro IO (Addr–IO) del que queremos leer o al que queremos escribir.
- Ejemplo:

Antes: IN Ri → Ahora: IN Ri, AddrPortIn.

Antes: OUT Ri →Ahora: OUT AddrPortOut, Ri.

- **Ejercicio:**

- Determina la palabra de control para las siguientes instrucciones: IN R3, 12 / OUT 0x0F, R5.

[illegible]

Índice

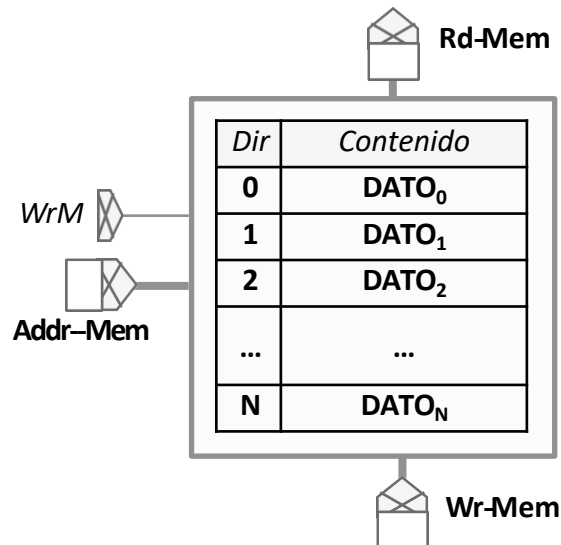
- **Introducción:**
 - Unidades de Control/Proceso (Específicas).
 - Un Ejemplo: Máximo Común Divisor.
- **El procesador de propósito general; definición.**
- **Unidad de Proceso de PPG:**
 - Banco de Registros.
 - ALU.
 - Palabra de Control.
- **Entrada/Salida.**
- **Memoria.**
- **Unidad de Control de PPG:**
 - Secuenciamiento de Instrucciones.
 - Formato de Instrucciones.

Memoria

- El número de registros de nuestro procesador es 8.
- ¿Por qué no se ha puesto un banco de registros mayor?
 - Máxima en diseño de procesadores: Pequeño → Simple → Rápido. Un banco de registros pequeño reduce el tiempo de propagación de dicho circuito, permitiendo operar a nuestro procesador con tiempos de ciclo más cortos (mayor frecuencia).
- ¿Qué pasa si necesito operar con más de 8 datos?
 - Emplearemos una memoria de mucho mayor tamaño (pero más lenta) que llamaremos **Memoria Principal**.

- Conceptualmente:

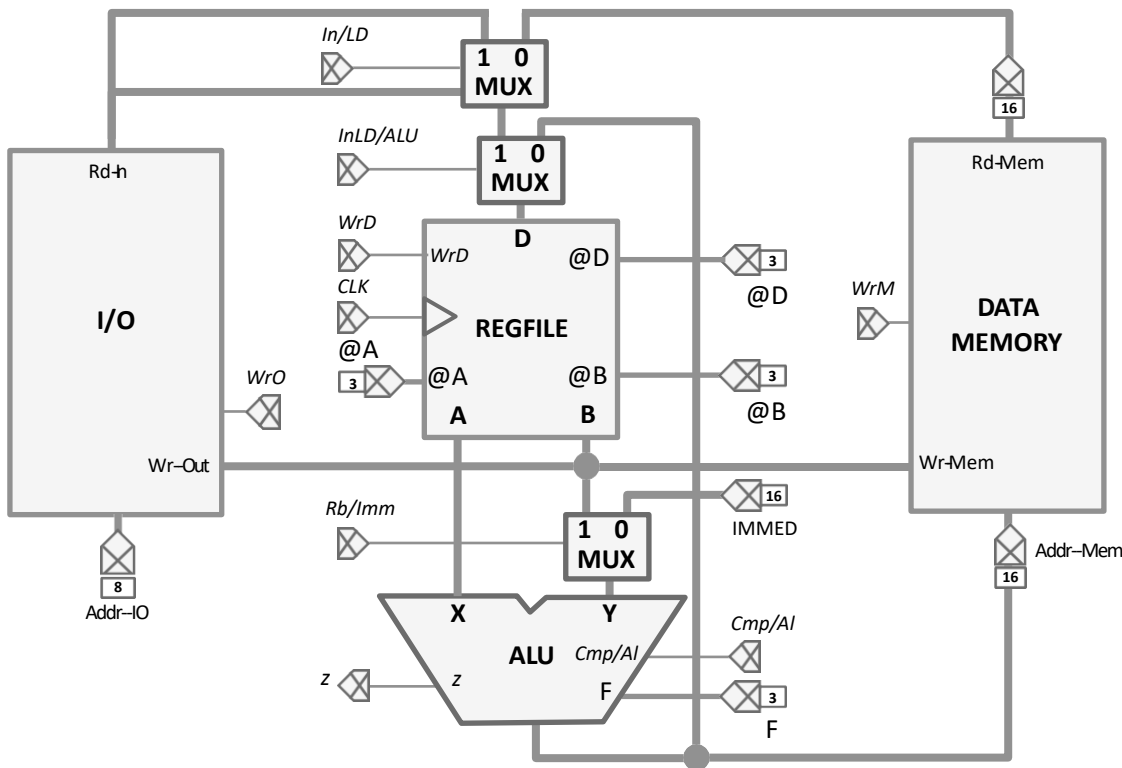
Utilizaremos los datos del procesador (16 bits) para determinar la dirección de la memoria a la que queremos acceder.
¿Cuántas posibles direcciones?



Puesto que el procesador trabaja con datos de 16 bits, ese será el tamaño de cada dato de Memoria (DATO_i: 16 bits).

Memoria

- Conectando la Memoria al Procesador:



La escritura en memoria se hará desde el mismo punto que se utiliza para IO, La salida B del Banco de Registros.

La dirección de Lectura/ Escritura en Memoria será generada por la ALU.

Para la lectura necesitamos añadir un Mux adicional, para determinar si el dato proviene de Entrada/Salida ó de Memoria.

¿Cómo afectan estos cambios a las palabras de control?

Memoria

- **Conectando la Memoria al Procesador:**

- La palabra de control vuelve a crecer. Puesto que aparecen dos señales de control nuevas (In/LD y WrM), pasamos de 41 a 43 bits.

In/LD	WrM	WrO	In/ALU	WrD	@D			@A			Cmp/Al	F			RB/I	@B			IMMED	Addr-IO $b_7b_6b_5...b_1b_0$
					b_2	b_1	b_0	b_2	b_1	b_0		b_2	b_1	b_0		b_2	b_1	b_0		

- Además de las instrucciones para Leer/Escribir de IO (IN y OUT), necesitamos dos nuevas instrucciones para hacerlo desde Memoria: **LOAD** y **STORE**.
- El formato es el siguiente:
 - **LOAD R2, 0x4001(R6):**
 - Dirección que queremos leer: Contenido de R6 + IMMED(0x4001).
 - En qué registro almacenamos la lectura de memoria?: R2.
 - **STORE 0x0001(R4), R7:**
 - Dirección en la que queremos escribir: Contenido de R4 + IMMED (0x0001).
 - Qué queremos escribir en esa dirección?: El contenido de R7.

Memoria

- **Ejercicio:**

- Determina la palabra de control para las siguientes instrucciones:

LOAD R2, 0x4001(R6)

IN R2 0x4001

[illegible]

STORE 0x0001(R4), R7

OUT 0x0001, R7

[illegible][illegible][illegible]

¡Muchas gracias!