# Introducción a los sistemas digitales y los microprocesadores

© Luis Entrena, Celia López, Mario García,
Enrique San Millán

Universidad Carlos III de Madrid

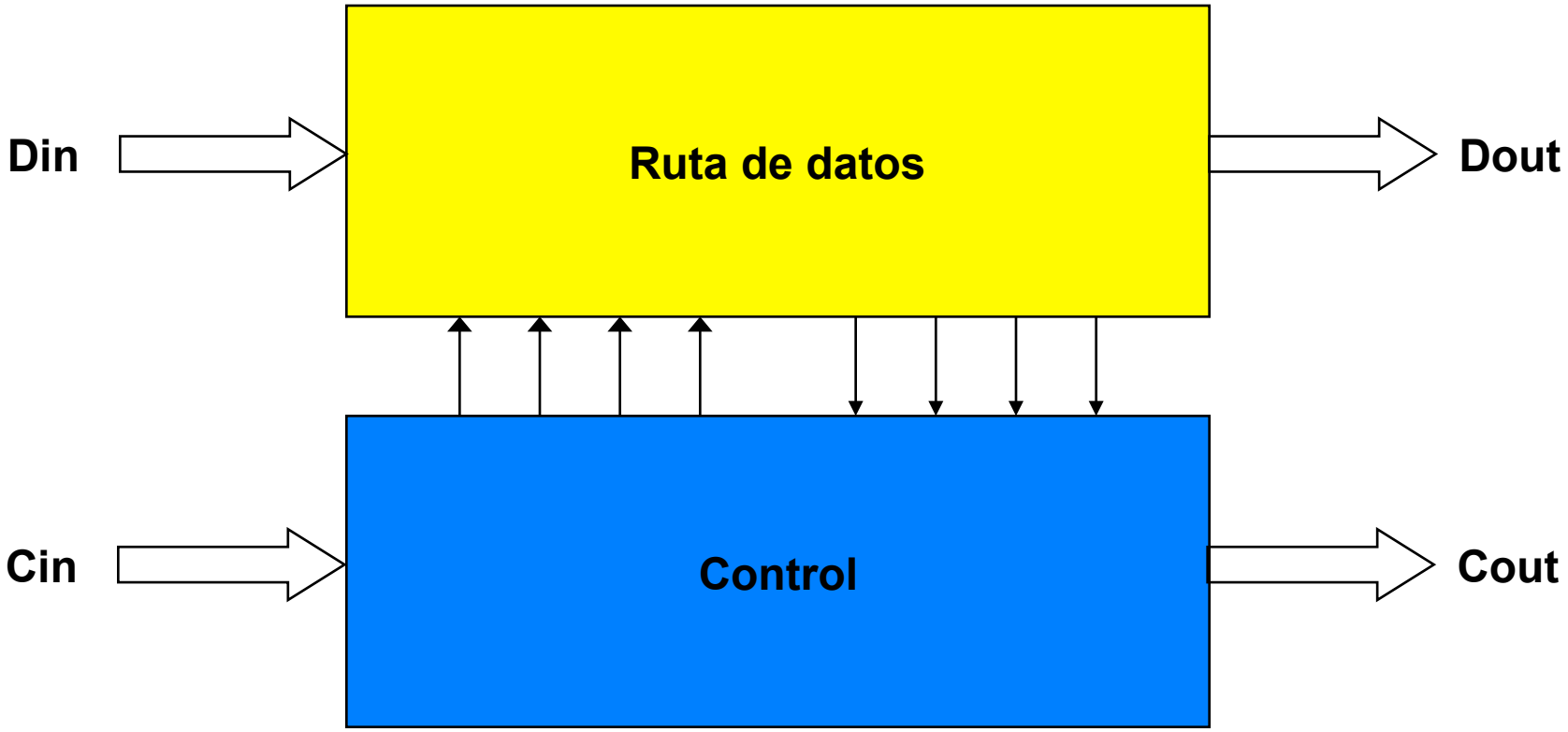
# Índice

- Estructura de un sistema digital
  - La ruta de datos
  - La unidad de control
- Estructura de un computador elemental
- Funcionamiento del computador elemental. Instrucciones

# Sistemas digitales

- Los sistemas digitales procesan información digital, de acuerdo con un algoritmo determinado
- Algoritmo: conjunto ordenado y finito de operaciones que permite hallar la solución a un problema
- Objetivo de la lección: introducción a los sistemas digitales

# Estructura de un sistema digital



# La ruta de datos

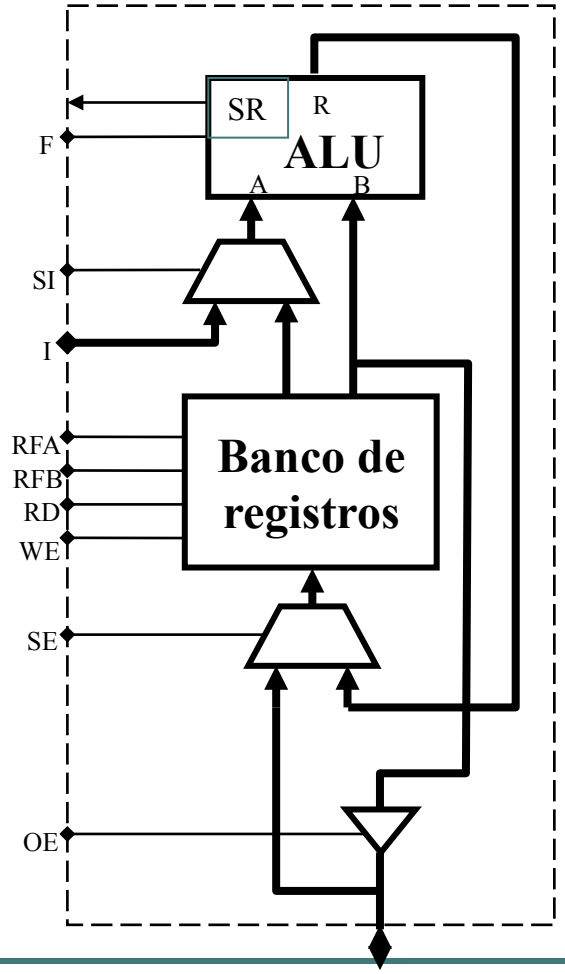
- Todo algoritmo se descompone en una serie de operaciones básicas:
  - Aritméticas
  - Lógicas
  - Desplazamientos y rotaciones
- La ruta de datos (“datapath”) es el conjunto de unidades funcionales que procesan los datos

# La ruta de datos

- Componentes típicos de la ruta de datos
  - ALUs: realizan las operaciones necesarias
  - Registros y memorias: almacenan datos temporales
  - Buses: conectan los elementos de la ruta de datos
  - Multiplexores: seleccionan los datos que se deben procesar en cada momento

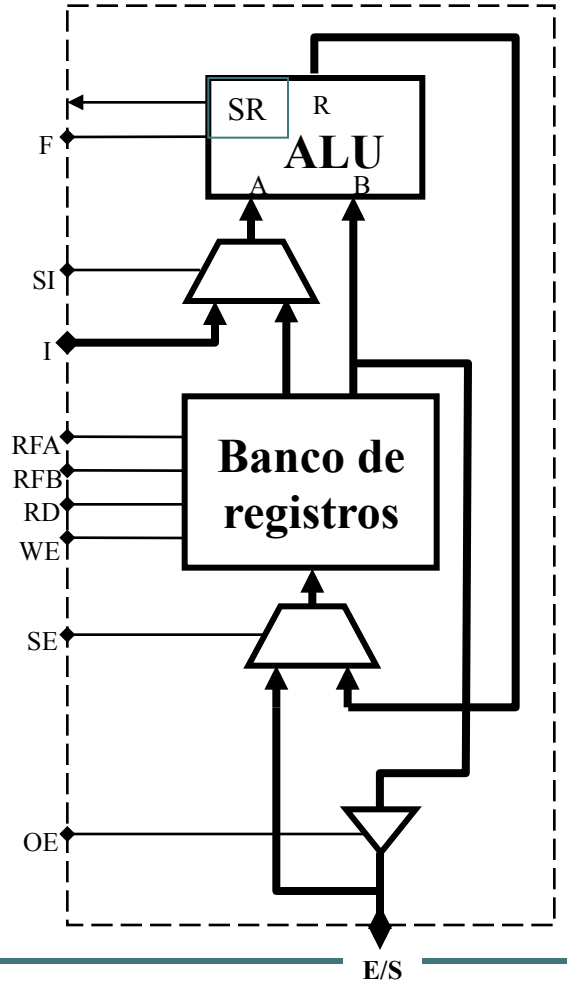
# Ejemplo de ruta de datos

- ALU realiza operaciones
  - F: Función de la ALU
- SR: Registro de estado. Indicadores (flags) de la operación realizada. Ejemplos:
  - C: Acarreo
  - O: Overflow
  - Z: Cero
  - S: Signo
- Operandos inmediatos a través de I, seleccionados con SI



# Ejemplo de ruta de datos

- Banco de registros (memoria de triple puerto) almacena valores intermedios
- Selección de registros
  - RFA: Registro Fuente A
  - RFB: Registro Fuente B
  - RD: Registro Destino
- E/S para conexión externa
  - SE selecciona la fuente para RD (externa o interna)
  - OE habilita la salida



# Unidad de control

- Determina la correcta secuenciación y utilización de las operaciones sobre los datos
- Elementos típicos:
  - Máquinas de estados
  - Contadores
  - Registros y memorias con datos de control, etc.

# Unidad de control

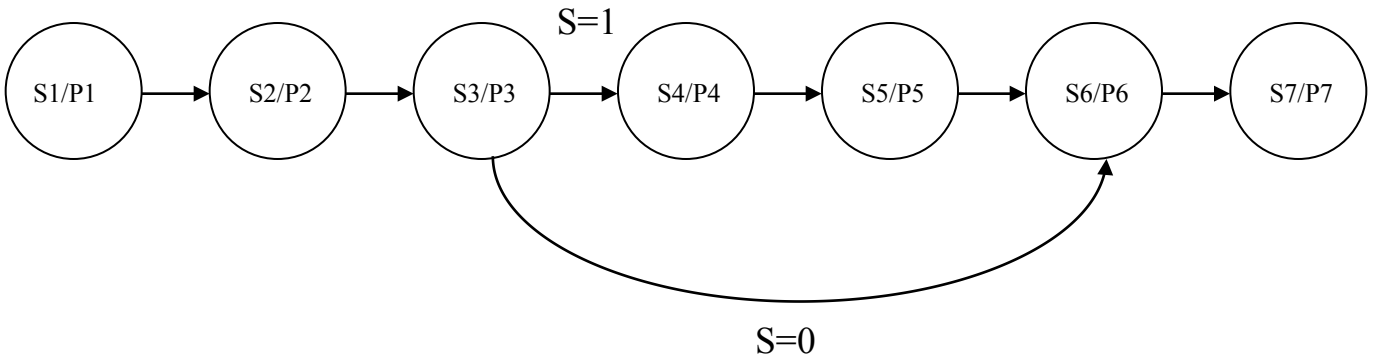
- Ejemplo: realizar la operación siguiente

  y = abs(x1 - x2)/2

- Pasos:
  - 1. Cargar x1 en el registro R1
  - 2. Cargar x2 en el registro R2
  - 3. Restar x2 de x1 y colocar el resultado en el registro R3 (R3 = R1 – R2).
    - Si el resultado es positivo, saltar al paso 6. En caso contrario, seguir con el paso 4.
  - 4. Complementar R3 (R3 = NOT R3)
  - 5. Incrementar R3 (R3 = R3 +1)
  - 6. Desplazar R3 a la derecha (R3 = R3/2)
  - 7. Enviar R3 a la salida

# Unidad de control

- Máquina de Estados de la unidad de control



# Unidad de control

- Salidas de la Máquina de Estados

| Paso | Función ALU (F) | RFA | SI | I | RFB | RD | WE | SE | OE |
|---|---|---|---|---|---|---|---|---|---|
| 1 | X | X | X | X | X | R1 | 0 | 1 | 0 |
| 2 | X | X | X | X | X | R2 | 0 | 1 | 0 |
| 3 | RESTA | R1 | 0 | X | R2 | R3 | 0 | 0 | 0 |
| 4 | NOT | R3 | 0 | X | X | R3 | 0 | 0 | 0 |
| 5 | SUMA | X | 1 | 1 | R3 | R3 | 0 | 0 | 0 |
| 6 | SHR | R3 | 0 | X | X | R3 | 0 | 0 | 0 |
| 7 | X | X | X | X | R3 | X | 1 | X | 1 |

# Diseño de sistemas digitales

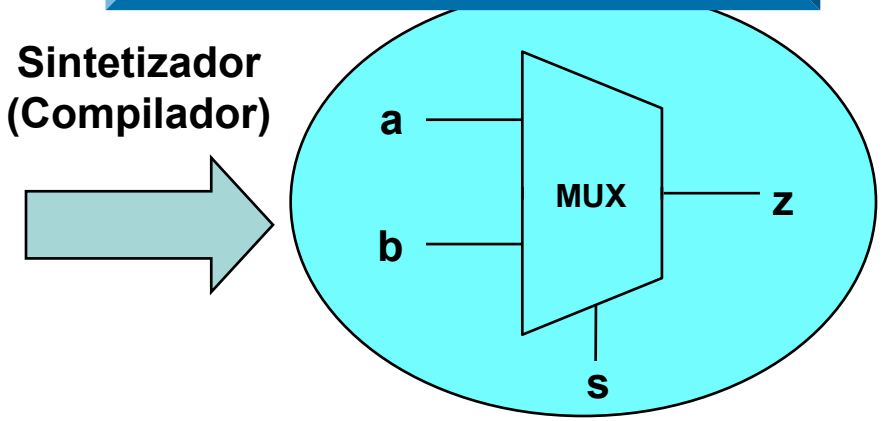
- El diseño con componentes tan básicos como las puertas lógicas no es práctico para diseñar sistemas digitales
- Diseño en el Nivel de Transferencia entre Registros (Register Transfer Level, RTL):
  - Componentes más abstractos: ALUs, registros o multiplexores
  - El sistema se describe como operaciones que se realizan entre datos almacenados en registros

# Lenguajes de Descripción de Hardware

- Los Lenguajes de Descripción de Hardware o HDLs (Hardware Description Languages) permiten diseñar un circuito digital desde un mayor nivel de abstracción
- Ejemplo (VHDL)

```
if s = '0' then
        z <= a;
else
        z <= b;
end if;
```

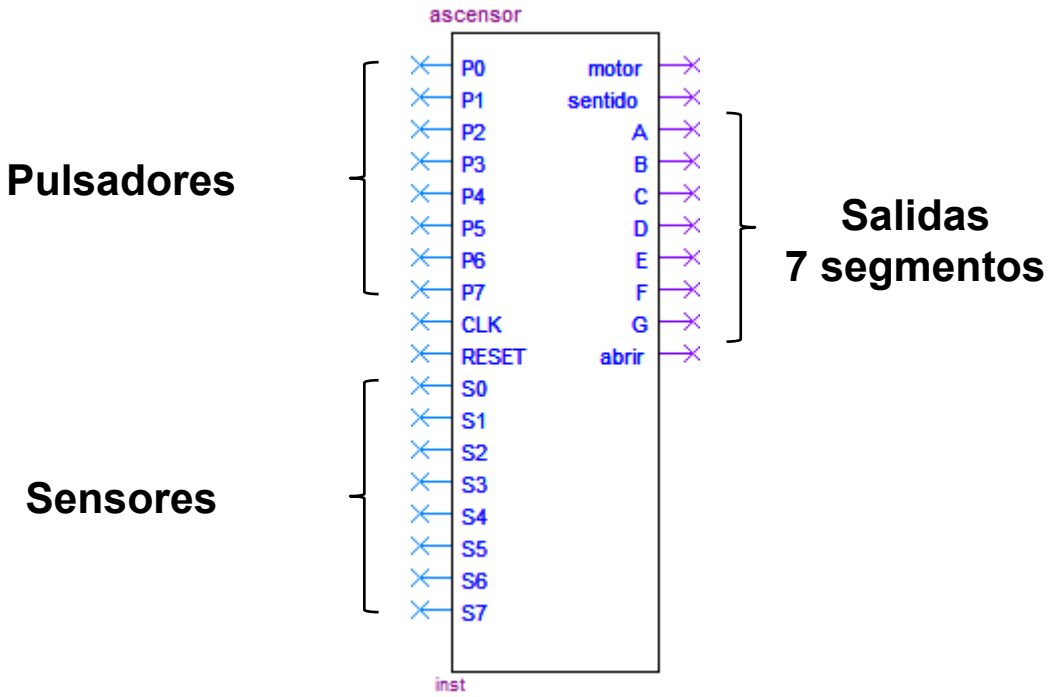
✓Asignatura 4º Curso
Diseño de Circuitos Integrados



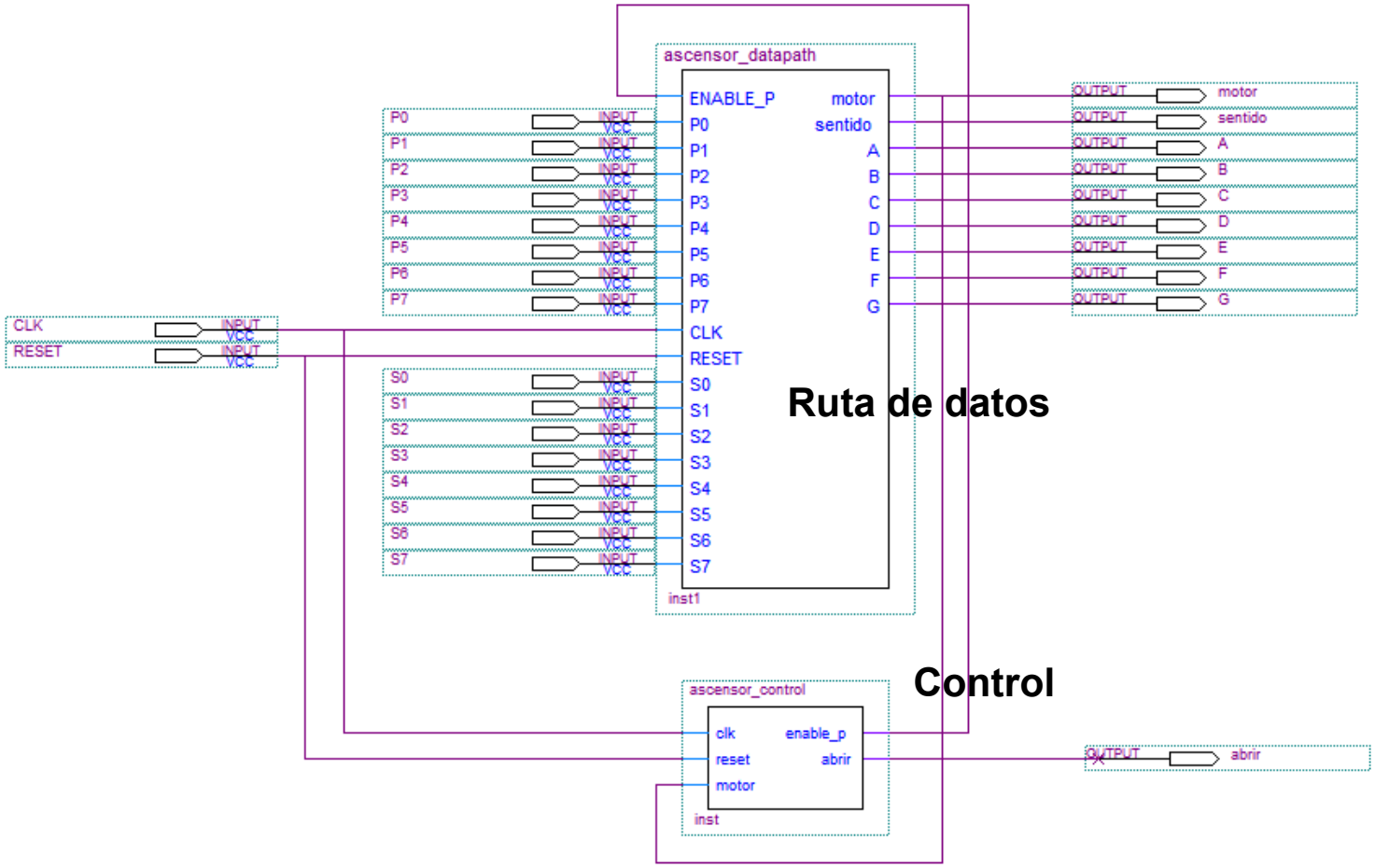
# Ejemplo completo: Controlador de un ascensor

- Especificaciones
  - 8 Plantas
  - Sin memoria: no se atienden nuevas peticiones hasta que el ascensor ha completado la petición en curso
  - Apertura de puertas temporizada: cuando el ascensor ha llegado a su destino debe abrir la puertas y esperar un tiempo dado (por simplicidad, supondremos que el tiempo de espera es 10 ciclos de reloj)
  - Visualización de planta mediante Display de 7 segmentos
- Entradas:
  - 8 Pulsadores de petición de destino (1 pulsador por planta)
  - 8 Sensores de paso por planta
- Salidas:
  - Motor: activar el motor
  - Sentido: ascendente (1) o descendente (0)
  - Abrir: Puertas abiertas (1) o cerradas (0)
  - Planta actual en código de 7 segmentos

# Ejemplo: Entradas y Salidas



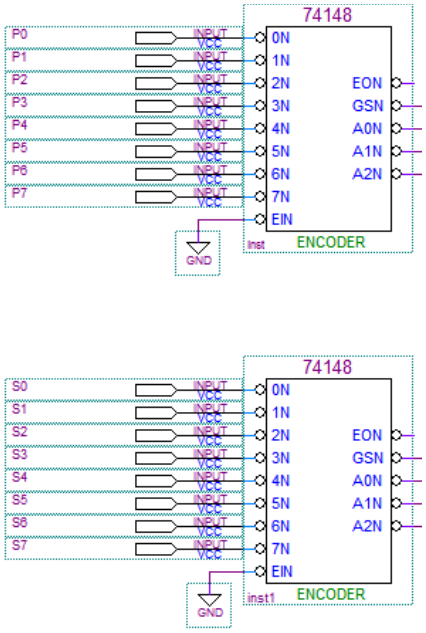
# Ejemplo: Descomposición



# Ejemplo: Descomposición

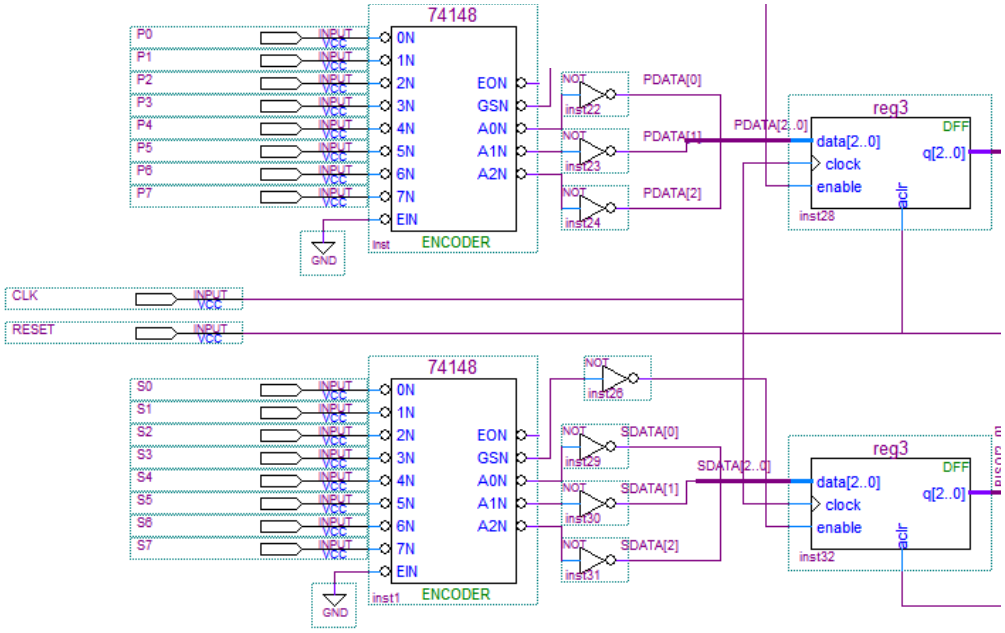
- Ruta de datos
  - Produce las señales de salida MOTOR , SENTIDO y visualizador de 7 segmentos, a partir de las entradas PULSADORES y SENSORES
- Control
  - Entradas:
    - MOTOR: indica cuando el ascensor está en movimiento
  - Salidas
    - ENABLE_P: Permite atender nuevas peticiones sólo cuando el ascensor está parado y las puertas están cerradas
    - ABRIR: Abre las puertas bajo control temporizado

# Ejemplo: Ruta de datos



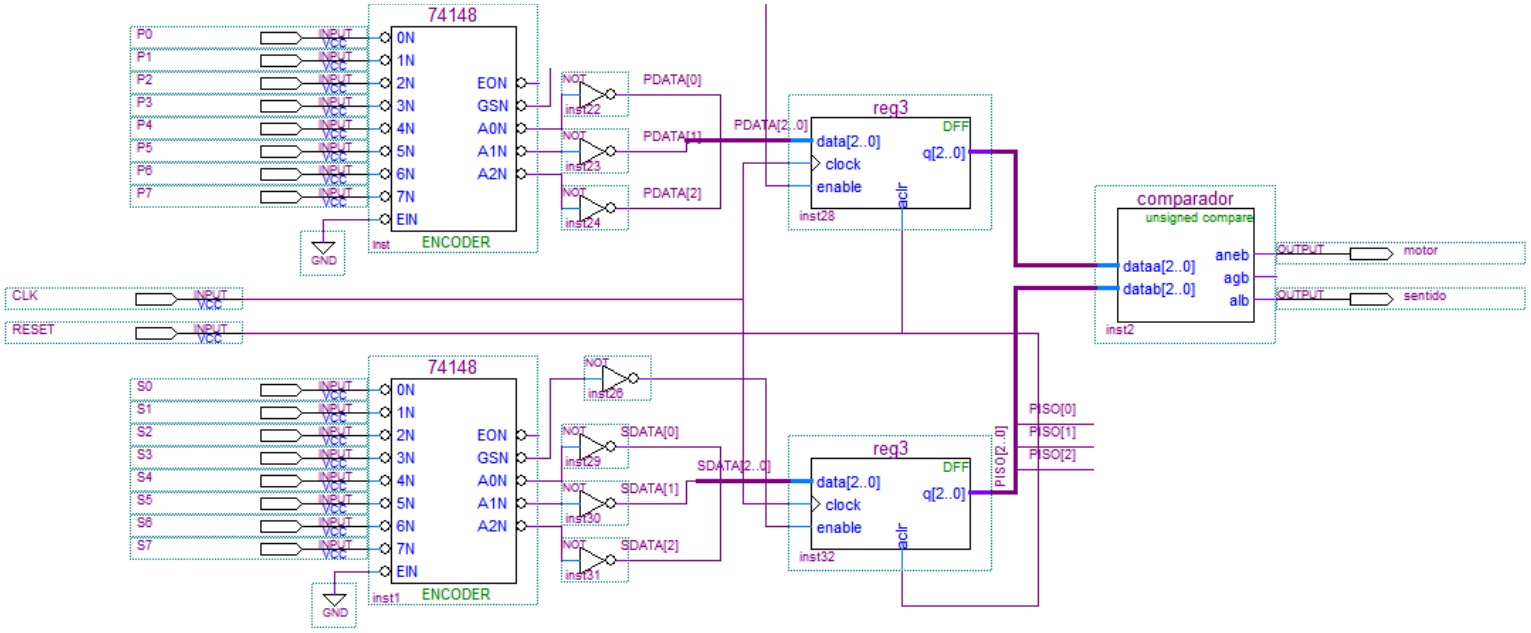

- Codificación de las entradas:
  - Utilizamos un codificador con prioridad, para el caso de que haya múltiples entradas activas
  - Las señales de grupo (GSN) nos sirven para saber si hay alguna entrada activa

# Ejemplo: Ruta de datos



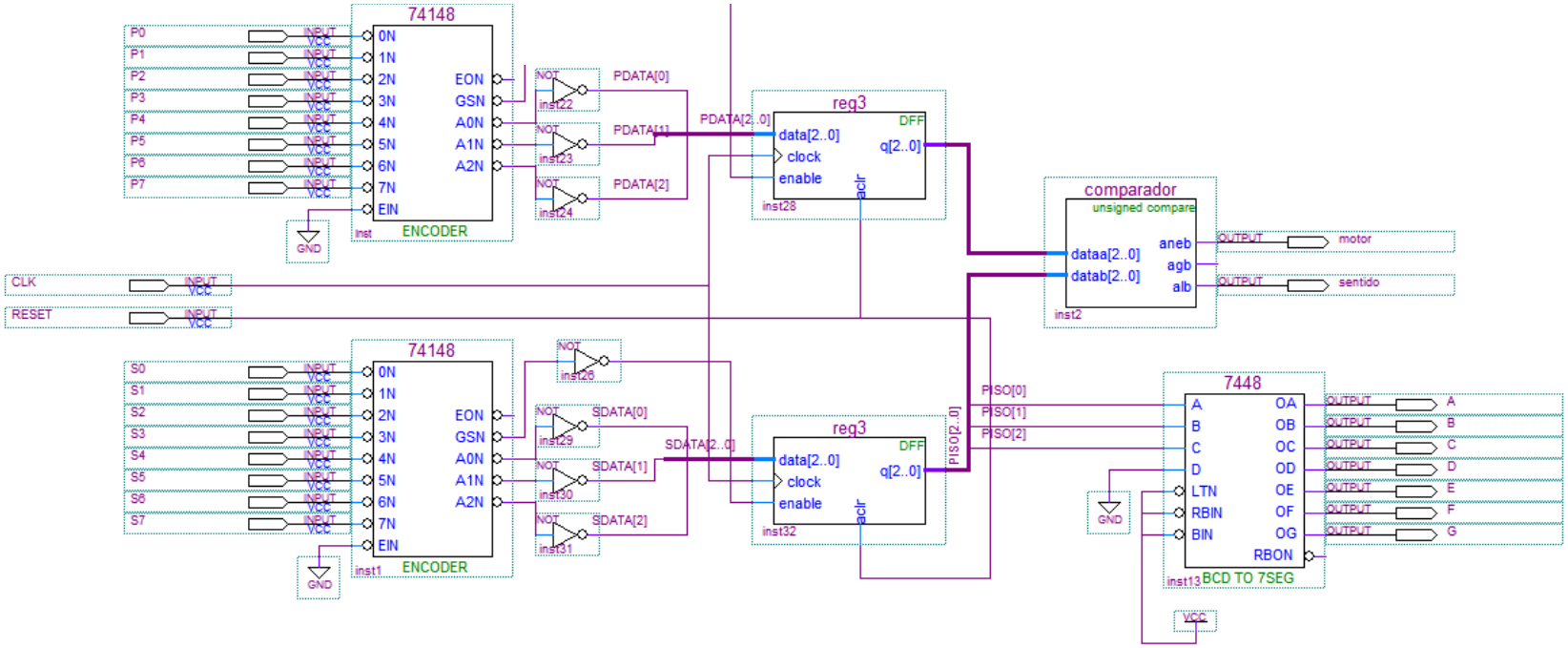

- Registros en las entradas codificadas:
  - Mantienen el valor cuando los pulsadores o los sensores se desactivan

# Ejemplo: Ruta de datos



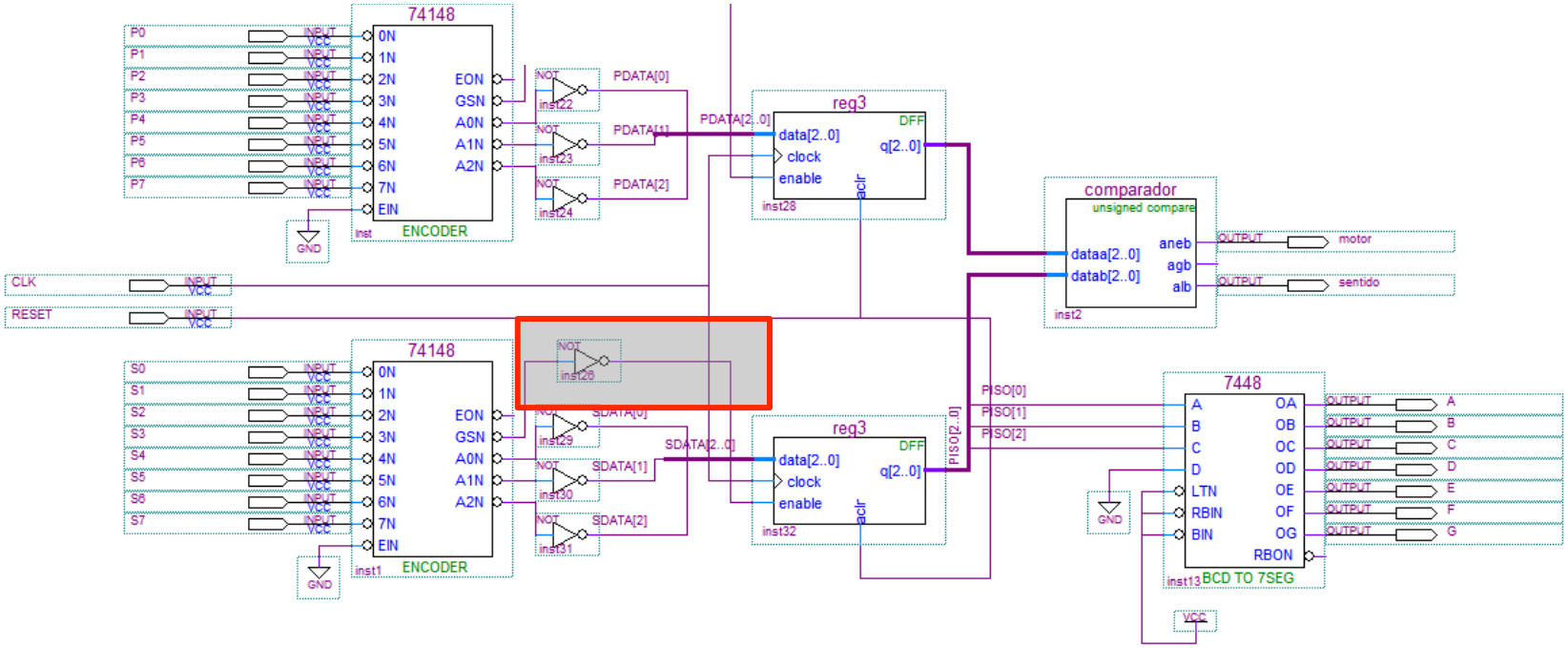

- Comparación:
  - Si el destino (PDATA) es distinto de la planta (SDATA), activamos el motor
  - Si PDATA < SDATA, el sentido es ascendente (1) y descendente (0) en caso contrario

# Ejemplo: Ruta de datos



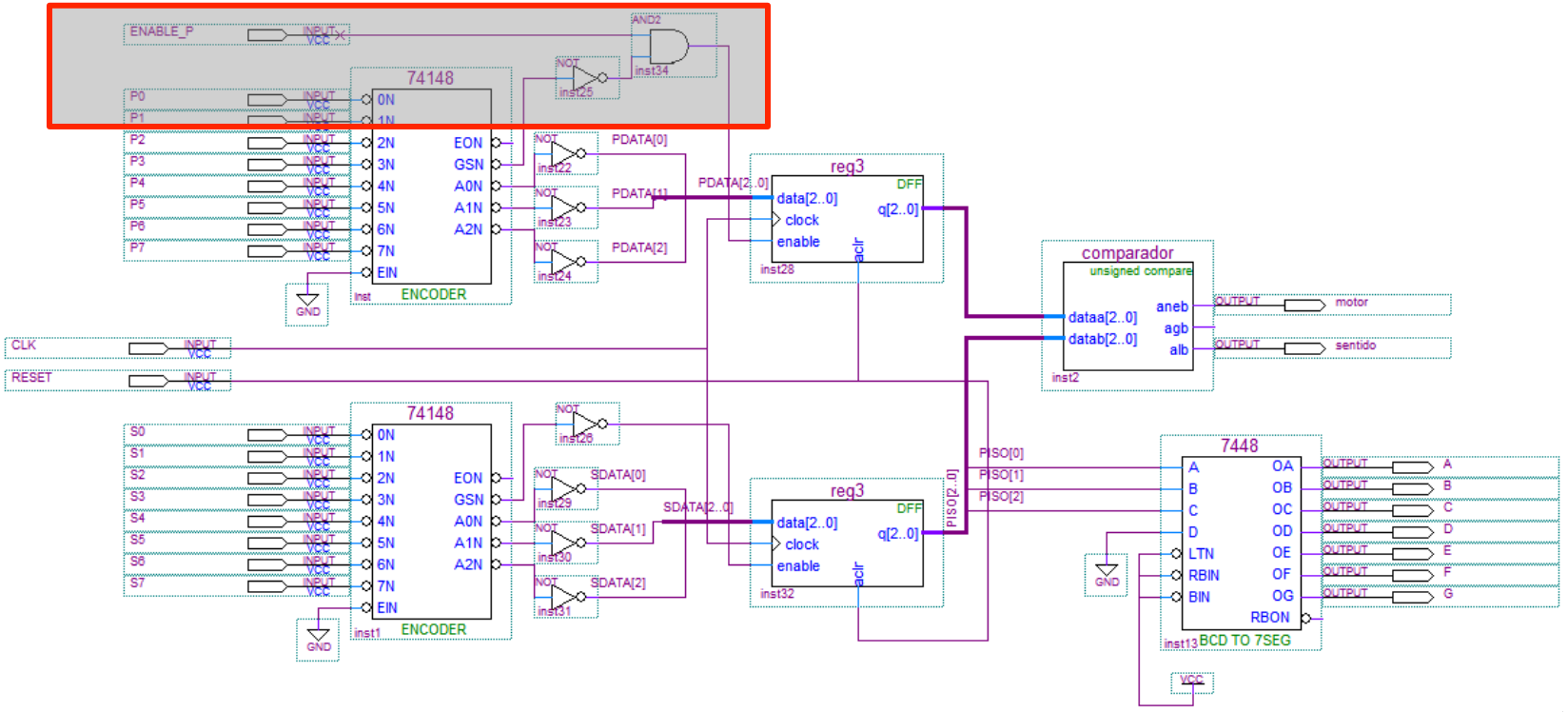

- Display de planta
  - La planta (PISO), se muestra al exterior en código de 7 segmentos

# Ejemplo: Ruta de datos



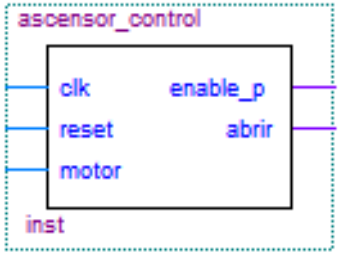

- Mantenimiento de la información de los sensores de planta
  - El registro de planta se habilita solo cuando algún sensor está activo; en caso contrario se mantiene la información
  - La señal de grupo del codificador se utiliza para este propósito

# Ejemplo: Ruta de datos



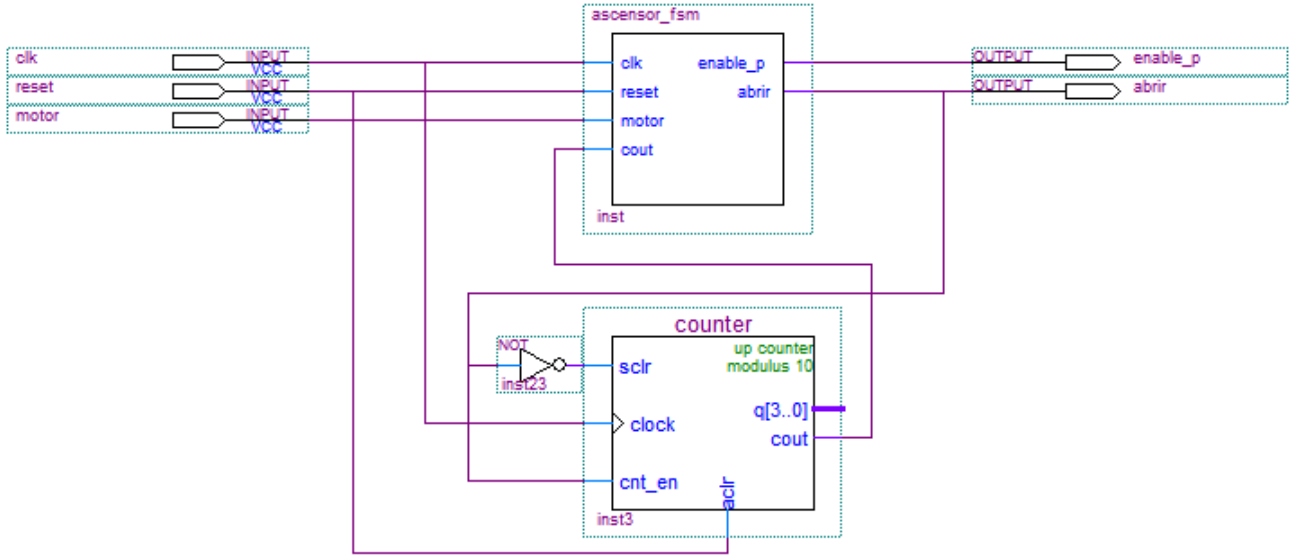

- Mantenimiento de la información del destino
  - Cuando el ascensor se está moviendo, el registro de destino debe bloquearse hasta completar el movimiento (señal de control ENABLE_P)
  - El registro de destino se habilita solo cuando algún sensor está activo (GSN) y el ascensor está parado (ENABLE_P)

# Ejemplo: Control



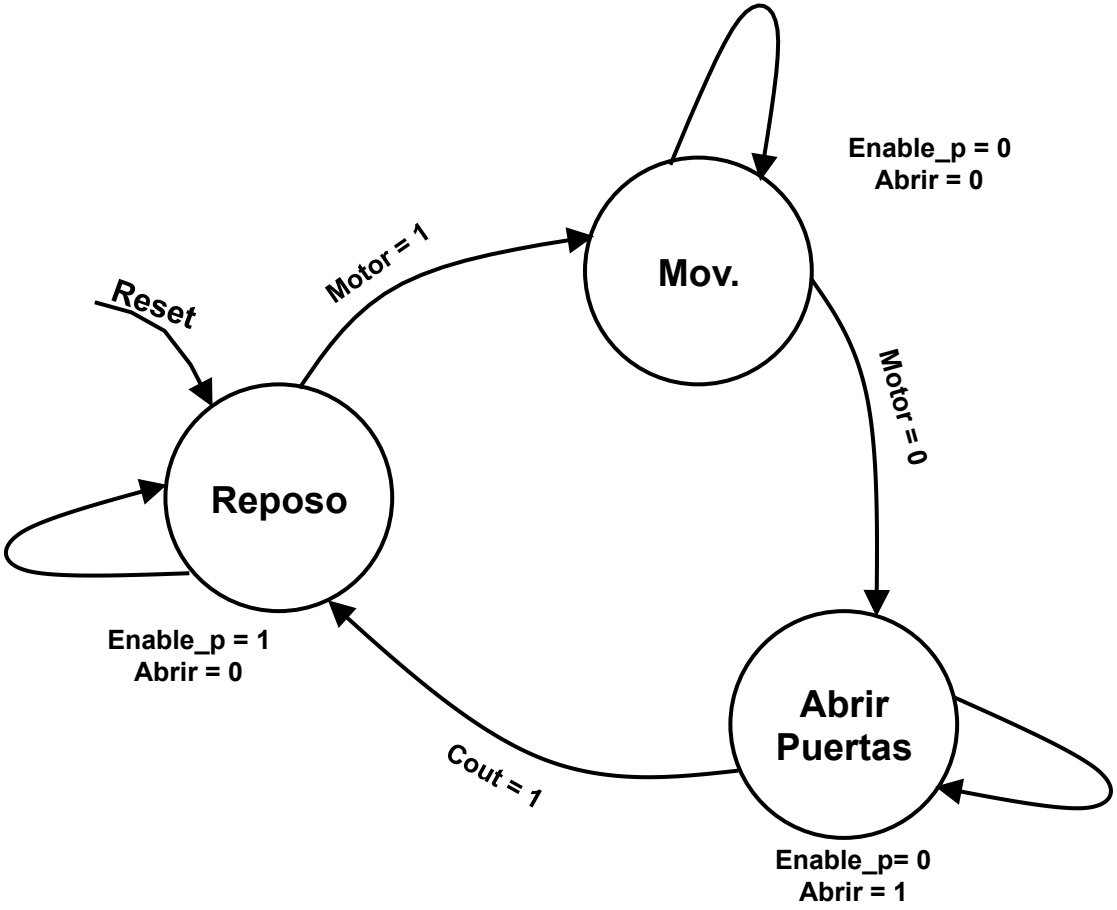

- Objetivos:
  - Bloquear la aceptación de nuevas peticiones (ENABLE_P = ‘0’) cuando el ascensor ha iniciado ya un movimiento (MOTOR = ‘1’)
  - Al llegar a destino, abrir las puertas durante el tiempo indicado
  - Finalmente, permitir la aceptación de nuevas peticiones (ENABLE_P = ‘1’)

# Ejemplo: Control




- El control se descompone en:
  - Una máquina de estados
  - Un contador-temporizador:
    - Se habilita la cuenta (CNT_EN) cuando se abren las puertas y se borra la cuenta (SCLR) en caso contrario
    - La salida COUT determina cuándo se llega al fin de cuenta

# Ejemplo: máquina de estados

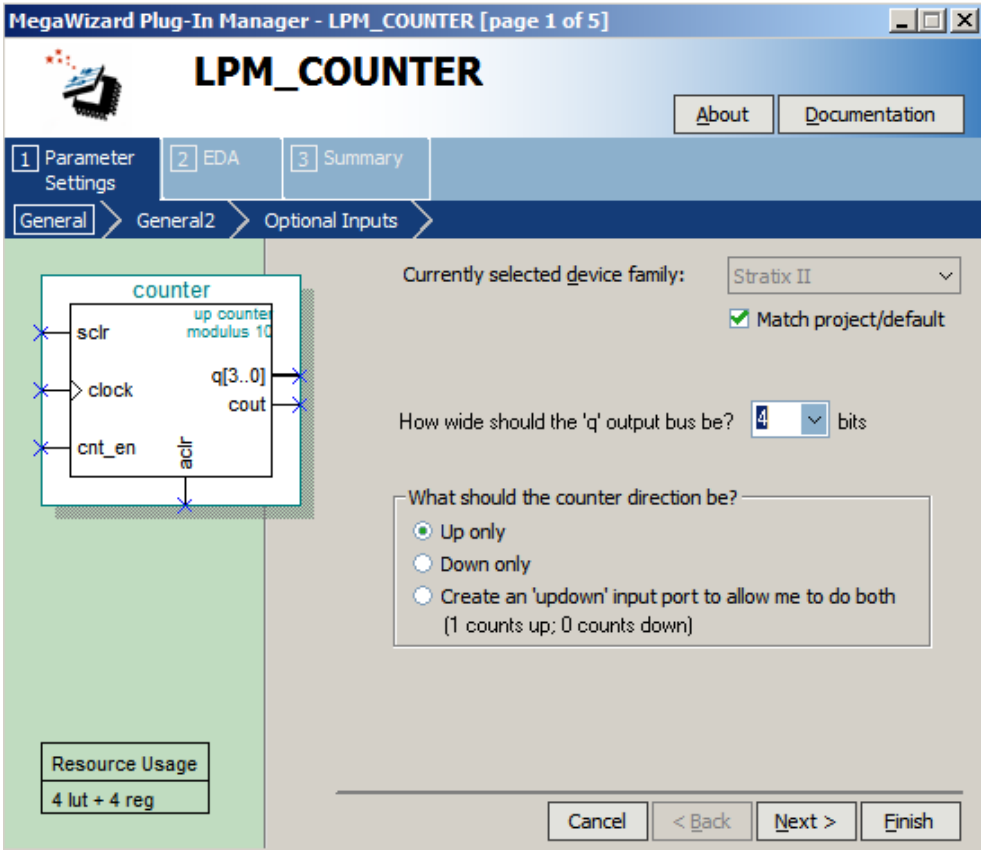


# Módulos utilizados

- Los módulos necesarios se pueden obtener de:
  - Bibliotecas de componentes primitivos (puertas y biestables)
    - NOT, AND2
  - Bibliotecas de fabricantes (Ejemplo: maxplus2)
    - Codificador 74148, Codificador BCD a 7 segmentos 7448
  - Bibliotecas de componentes parametrizables (Ejemplo: MegaWizard)
    - Reg3, comparador, contador
- O se pueden diseñar:
  - Diseño mediante captura de esquemas
  - Diseño con Lenguaje de Descripción de Hardware (VHDL)
    - Máquina de estados

# Módulos: componentes parametrizables

- ☑ Permiten realizar módulos comunes (registros, contadores, sumadores, memorias, etc.) con las opciones deseadas
- ☒ No son portables
- ☒ Sólo sirven para módulos muy generales



# Módulos: Diseño en VHDL

- ☑ Permite realizar cualquier diseño completo
- ☑ Portable
- ☑ Mejora productividad
- ☑ Orientado a describir la funcionalidad …
  - ¿qué quiero hacer?

  …en lugar de los componentes
  - ¿cómo lo hago?

```
architecture a of ascensor_fsm is
     type estados is (reposo, mov, puerta);
     signal actual: estados;
begin
     process(clk, reset)
     begin
          if reset = '1' then
               actual <= reposo;
          elsif clk'event and clk = '1' then
               case actual is
               when reposo =>
                         if motor = '1' then
                              actual <= mov;
                         end if;
               when mov =>
                         if motor = '0' then
                              actual <= puerta;
                         end if;
               when puerta =>
                         if cout = '1' then
                              actual <= reposo;
                         end if;
               end case;
          end if;
     end process;

     abrir <= '1' when actual = puerta else '0';
     enable_p <=    '1' when actual = reposo and motor = '0'
                    else '0';
end a;
```

# Implementación

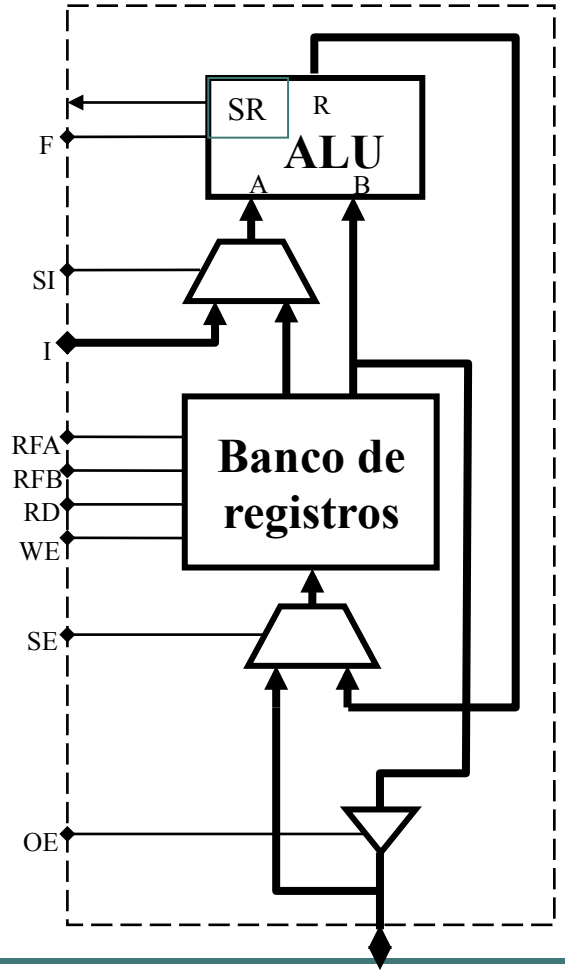
- Con componentes discretos
  - Placas voluminosas
  - Baja eficiencia
- En un circuito integrado (ASIC)
  - Caro, sólo aconsejable para elevados volúmenes de producción
- En un circuito programable (FPGA)
  - Fácil de implementar
  - Económico hasta para producciones altas
  - Actualizable

# Estructura de un computador elemental

- Computador: máquina de propósito general para el procesamiento de información
  - Orientado a realizar una amplia variedad de algoritmos
- Componentes:
  - Unidad de Proceso o Unidad Central de Proceso (CPU), que se encarga de realizar las operaciones necesarias
    - Ruta de datos general
    - Unidad de control programable (para poder realizar múltiples algoritmos)
  - Memoria, para el almacenamiento de información
  - Unidades de Entrada/Salida, para comunicarse con el exterior

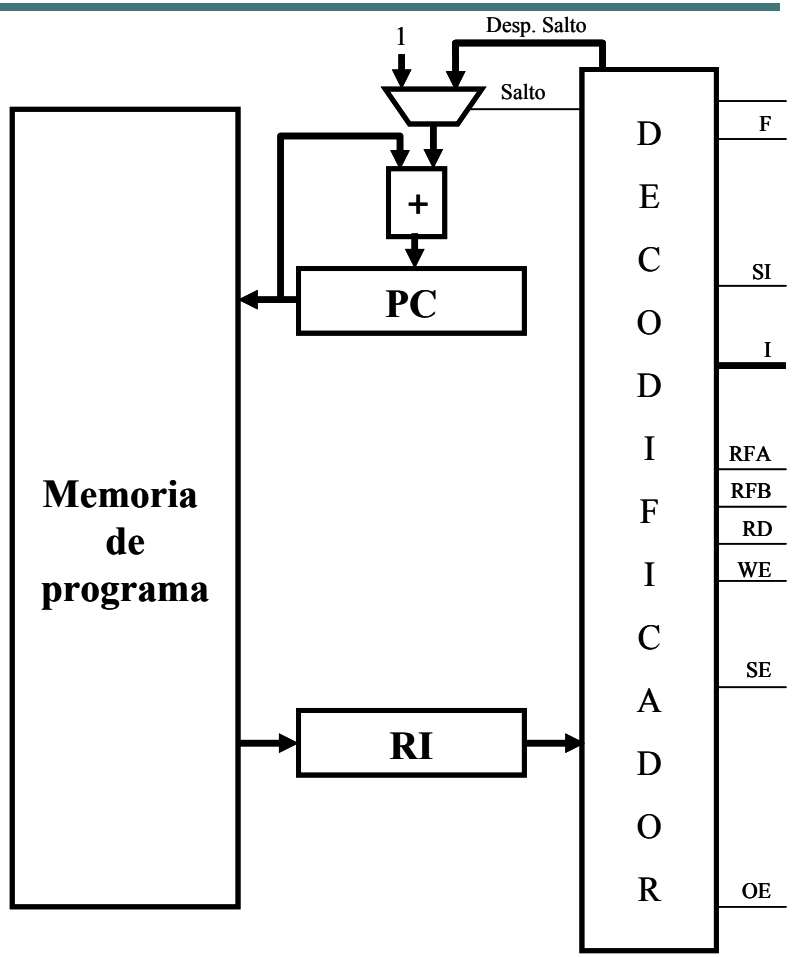
# Ruta de datos de propósito general

- Se puede diseñar una ruta de datos que pueda servir para diferentes algoritmos
- ¿Se puede diseñar también una unidad de control que pueda servir para diferentes algoritmos?

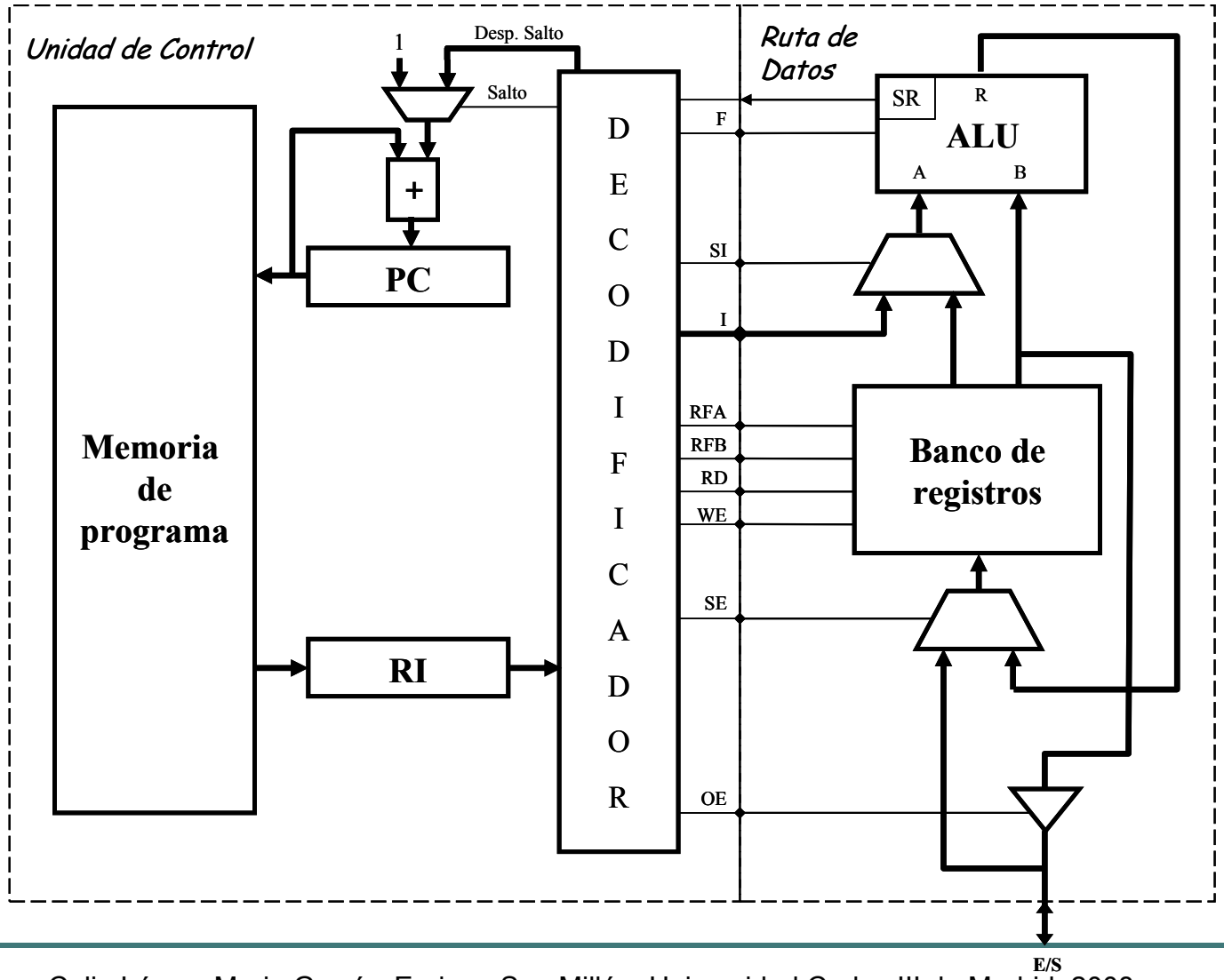


# Unidad de control programable

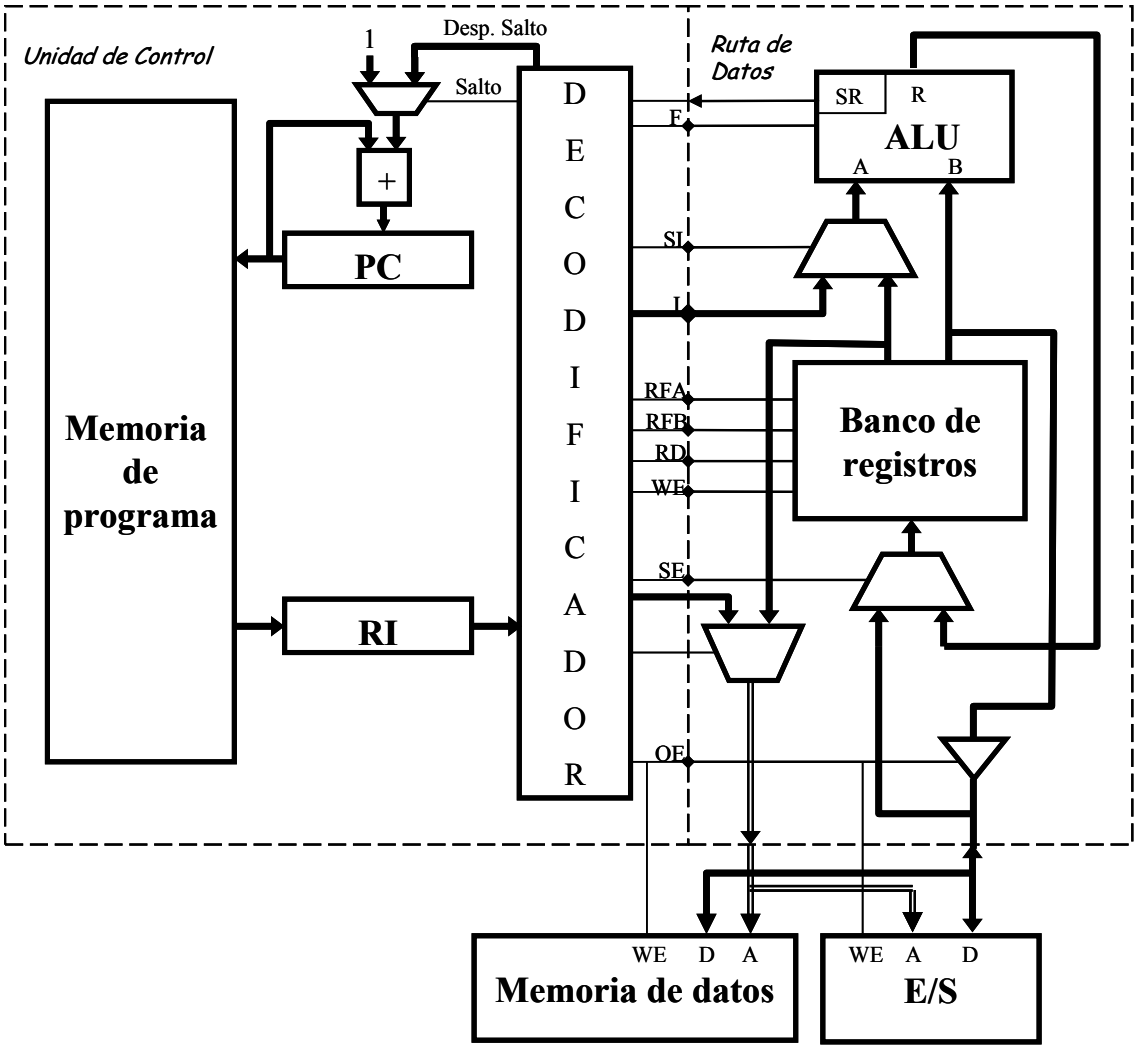
- Las palabras de control se almacenan en una memoria
  - Para ahorrar espacio, se codifican las palabras de control
  - Las palabras de control codificadas las llamamos *instrucciones*
- En cada ciclo de reloj se lee una instrucción (RI) y se decodifica
- La instrucción correspondiente a cada paso se determina con un contador (PC).
  - Los saltos en la secuencia se realizan con instrucciones, que pueden ser condicionados dinámicamente por SR



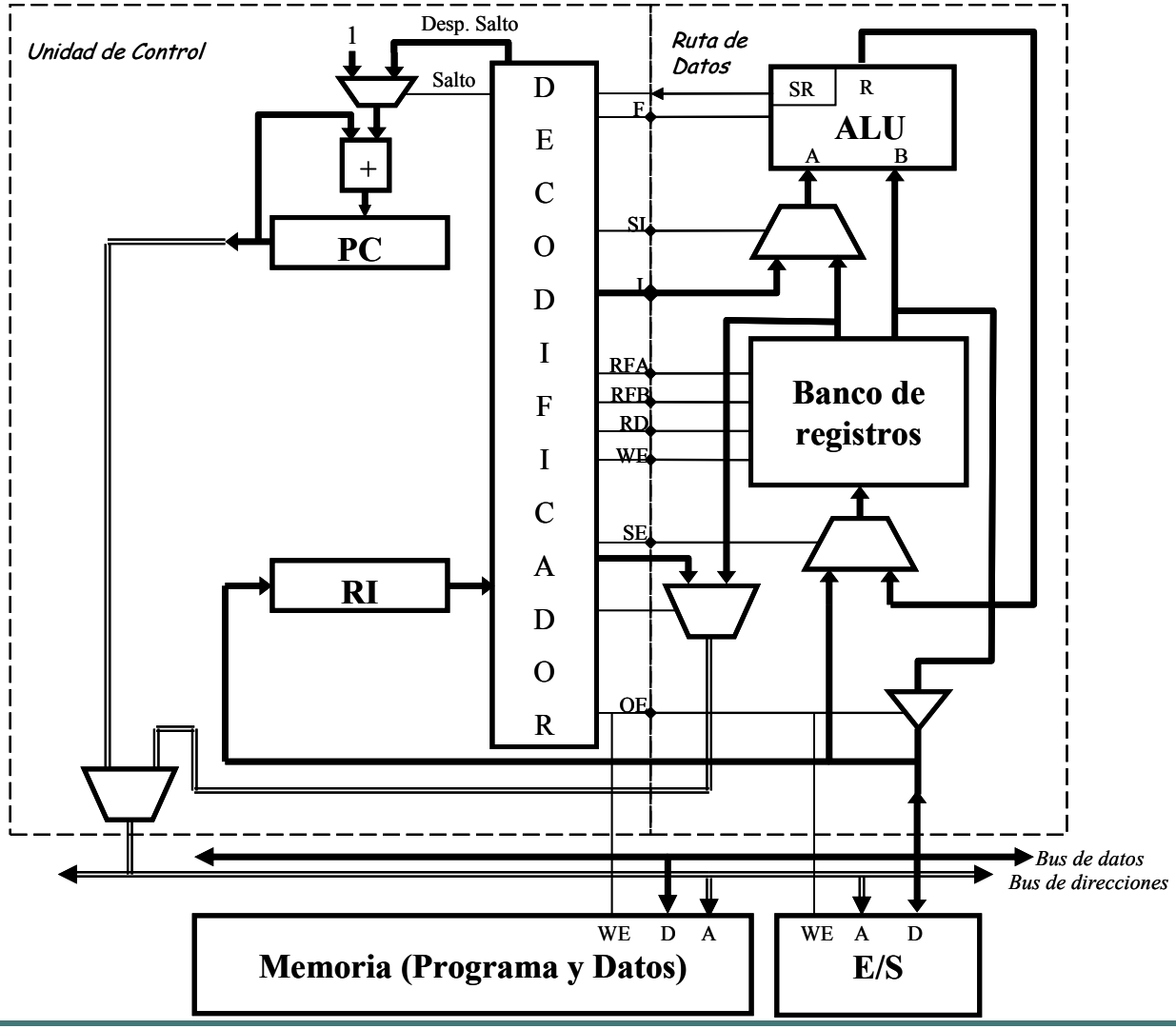
# Microprocesador



# Computador básico (Arquitectura Harvard)



# Computador básico (Arquitectura von Neuman)



# Funcionamiento del computador elemental. Instrucciones

- ¿Cómo se ejecutan las instrucciones?
  - Ciclo de instrucción
- ¿Qué tipos de instrucciones hay?
- ¿Cómo se codifica una instrucción?
  - Formato de instrucción
  - Modos de direccionamiento

# Ciclo de instrucción

- Cada instrucción se procesa típicamente en 2 fases:
  - Búsqueda de la instrucción: carga la instrucción en el IR
  - Ejecución de la instrucción: decodifica la instrucción y configura la ruta de datos para realizar la operación
- Cada una de estas fases puede realizarse en 1 ciclo de reloj o en varios, dependiendo de la complejidad del microprocesador

# Tipos de instrucciones

- Instrucciones de transferencia de datos
  - Transferencia de datos entre registros, entre un registro y la memoria, o entre un registro y un interfaz de E/S
- Instrucciones aritmético-lógicas
  - Realizan operaciones con la ALU: sumas, restas, desplazamientos, etc.
- Instrucciones de salto y bifurcación
  - Permiten realizar cambios en la secuencia de ejecución de las instrucciones
  - Modifican el PC
  - Ejemplos: saltos, llamadas a subrutina, etc.

# Formato de instrucción

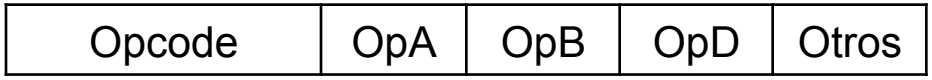
- Organización de la información de las instrucciones. Las instrucciones se codifican habitualmente por campos:
  - Código de operación (Opcode): indica la operación que se debe realizar
  - Operandos: indican los datos sobre los que se debe operar
- El número de operandos y el tamaño de los campos puede ser variable
- El tamaño de las instrucciones debe ser un múltiplo del ancho de palabra de la memoria

# Modos de direccionamiento

- Los operandos pueden indicarse de diversas formas, conocidas como modos de direccionamiento
- Algunos modos usuales:
  - *Inmediato*: el valor del operando se indica en la instrucción
  - *Directo por registro:* la instrucción indica un registro que contiene el operando
  - *Directo a memoria:* la instrucción indica una posición de memoria para el operando
  - *Indirecto*: la instrucción indica un registro que contiene la posición de memoria para el operando

# Ejemplo de formato de instrucción

- Para la arquitectura del ejemplo anterior, con los siguientes parámetros:
  - Código de operación: 5 bits
  - Banco de 8 registros (direccionamiento directo con 3 bits)
- Formato de instrucción



# Ejemplos de instrucciones

| Operación | Nemónico | Código de operación |
|---|---|---|
| Carga dato de memoria | LD | 0 0 0 0 0 |
| Almacena dato en memoria | ST | 0 0 0 0 1 |
| Suma | ADD | 0 1 0 0 0 |
| Resta | SUB | 0 1 0 0 1 |
| NOT | NOT | 0 1 1 0 0 |
| AND | AND | 0 1 1 0 1 |
| OR | OR | 0 1 1 1 0 |
| Desplazamiento | SHL, SHR | 0 1 1 1 1 |
| Salto incondicional | JMP | 1 0 0 0 0 |
| Llamada a subrutina | CALL | 1 0 1 0 0 |
| Retorno de subrutina | RET | 1 0 1 0 1 |
| . . . | . . . | . . . |

# Ejemplos de instrucciones

- Instrucción de una palabra

| Opcode | RFA | RFB | RD | Otros |
|---|---|---|---|---|
| 01000 | 001 | 010 | 011 | 00 |

**R3 = R1 + R2**

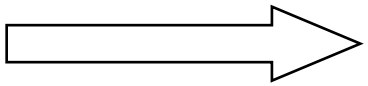
- Instrucción de dos palabras

| Opcode | RFA | RFB | RD | Otros |
|---|---|---|---|---|
| 00000 | 000 | 000 | 011 | 00 |
| Direccion | | | | |
| 1010 1011 1100 1101 | | | | |

**Carga en R3 el dato contenido en la posición de memoria ABCDh**

# Lenguaje ensamblador

- Los códigos de instrucción resultan muy poco manejables
- *Lenguaje ensamblador:* las instrucciones se especifican mediante nemónicos y los operandos mediante nombres simbólicos
- Programa ensamblador: traducen las instrucciones de un programa a su código correspondiente
- Ejemplo:

**ADD R1, R2, R3**  0100000101001100

# Conclusiones

- El diseño de sistemas digitales se realiza en el Nivel de Transferencia de Registros (RTL)
  - Se utilizan componentes más abstractos (sumadores, registros, multiplexores, etc.)
  - Los Lenguajes de Descripción de Hardware, como VHDL, para hacer el diseño más productivo
- El microprocesador es un sistema digital de propósito general
  - Ruta de datos de propósito general
  - Unidad de control programable: instrucciones

# Referencias


- “Principios de Diseño Digital”. D. Gajski. Ed. Prentice Hall