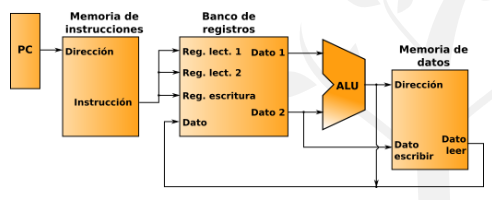
**Sección de procesamianto y control**

**Ejecución de instrucciones**

* Se siguen los siguientes pasos:
  + Captura: Se accede a la memoria de instrucciones y se lee la instrucción
  + Decodificación: Se decodifica la instrucción y se accede a los operandos, accediendo a memoria si es necesario
  + Ejecución: se realiza la operación necesaria, posiblemente usando la ALU
  + Escritura de resultados: se almacenan los resultados en memoria
* Todo este proceso se produce en **un ciclo de reloj**. Suponemos que cada instrucción se ejecuta en un ciclo (**instrucciones monociclo**)

**Camino de datos**

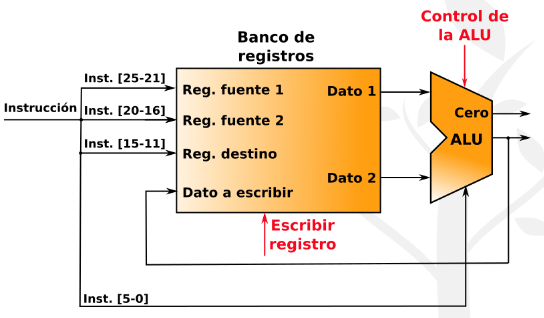


* **Contador de programa (PC)**: Apunta a la dirección de la próxima instrucción que ejecutar. Aumenta en cada ciclo de reloj (por 4 si no hay saltos)

**Incorporación de nuevas instrucciones**

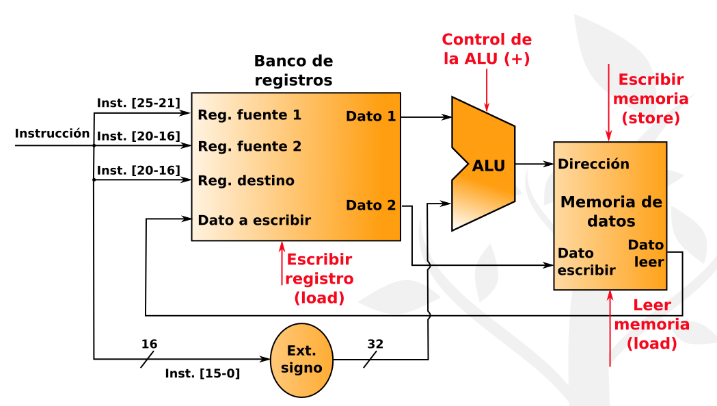
* Añadir nuevas instrucciones puede implicar añadir hardware adicional o modificar la unidad de control.
* Se debe intentar aprovechar el hardware existente.

**Instrucciones aritmético-lógicas**



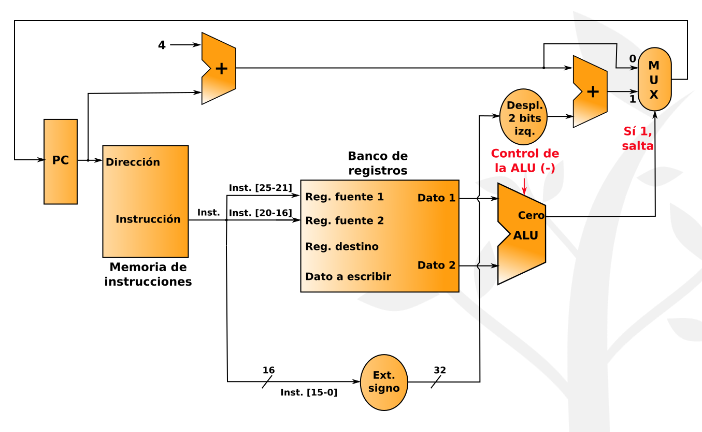
* Los tres registros se almacenan en el banco de registros, que devuelve los dos datos con los que operar
* Los datos se envían a la ALU, junto con el código de la instrucción
* El dato a escribir (resultado) se devuelve al banco de registros

**Carga/almacenamiento**



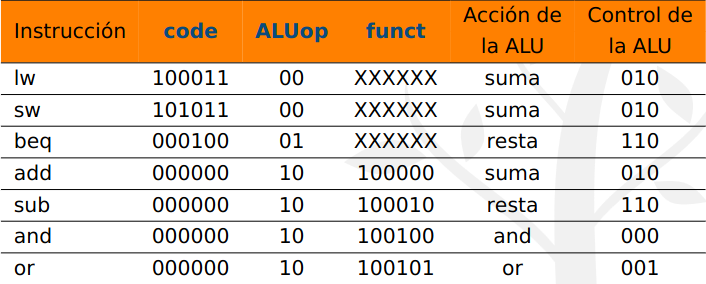
* Los tres registros se guardan en el banco de registros, que devuelve los datos que contienen.
* Si hay desplazamiento, se suma con una ALU
* Se accede a la dirección resultante mediante la memoria de datos, que:
  + Si es una operación de escribir, recibe el dato a escribir
  + Si es una operación de leer, devuelve el dato contenido.

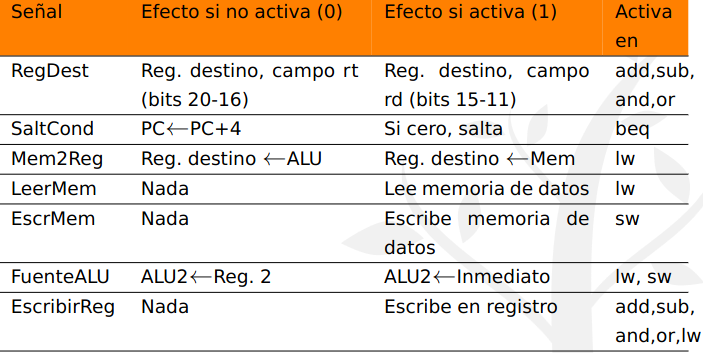
**Salto condicional**



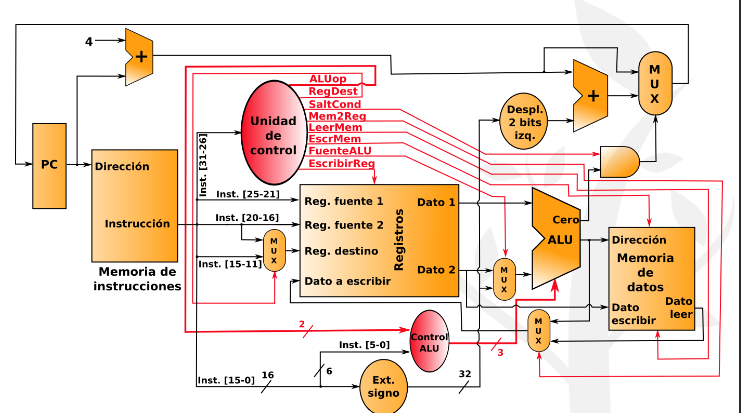
* Los registros se aportan al banco de registros, que devuelve los dos datos que comparar
* Para la condición se utiliza un multiplexor: se restan y si el resultado es igual a 1 (significa que no son iguales) el resultado será distinto de 0 y se salta. Si el resultado es 0, se mantiene el mismo PC pero se le suma 4 (para avanzar a la siguiente instrucción)
* La dirección de la etiqueta se multiplica por 4 (despl. 2 bits izq.) antes de sumarla al PC
* El resultado se guarda en el PC, realizando el salto si es el caso.
* nota: la arquitectura de la imagen completa no permite salto incondicional. Para incorporar esta función, sería necesario un multiplexor adicional que comprobase si SaltIncond==1.

**Unidad de control**

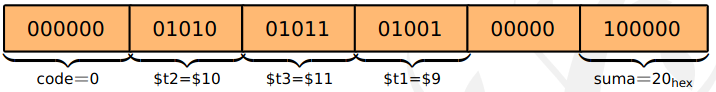
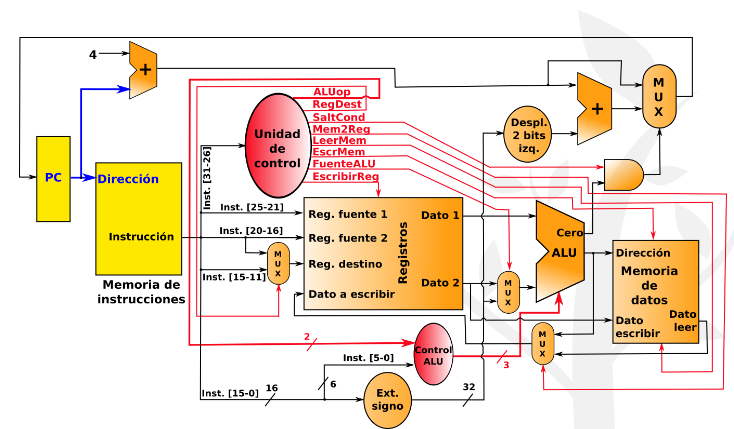
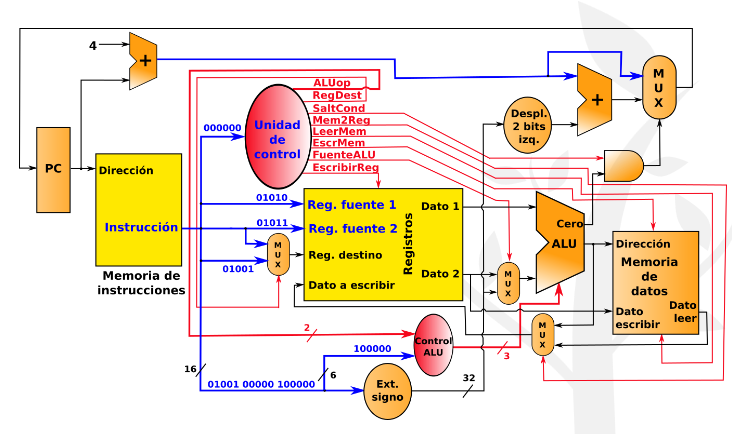
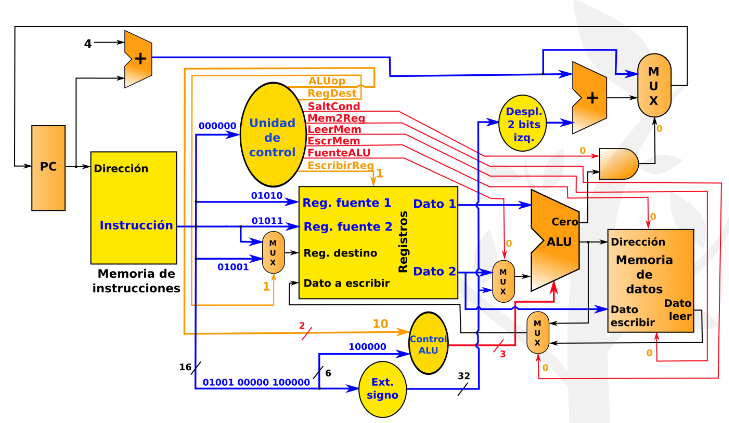
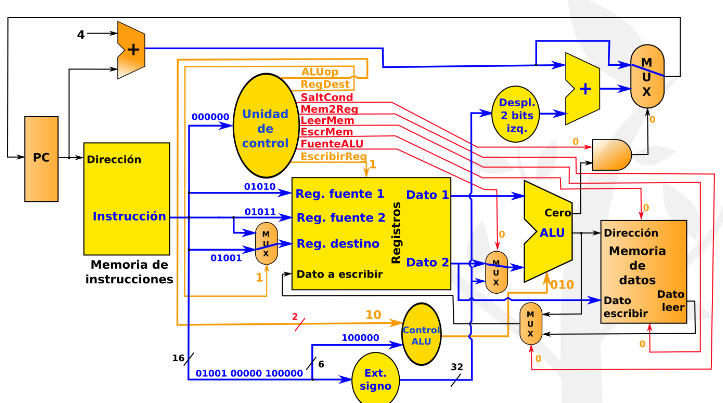
* La **Unidad de control** (UC) recibe los 6 bits de code, y genera:
  + 2 bits de **ALUop** que se pasan a la UCALU (en el esquema, Control ALU), y le indican la operación que debe realizar.
  + 
  + 7 bits separados de señal de control:



* Las instrucciones aritméticas (add, sub, and, or) además tienen código de función (funct) que recibe directamente la UCALU
* Camino de datos + unidad de control:



**Ejemplo (suma)**

* Instrucción: **add $t1,$t2,$t3**
* 
* Ciclo 1: se pasa la dirección del PC actual, para leer la instrucción.
* 
* Ciclo 2: Se leen los registros de fuente que se utilizaarán, se pasa el code a la UC, el funct a la UCALU e non sei o que ao ext. signo
* 
* Ciclo 3: Se accede a la memoria de instrucciones para leer los datos que sumar. La UC genera los bits de ALUop y pone a 1 los de RegDest y EscribirReg.
* 
* Ciclo 4: Se lee el registro de destino. La ALU recibe el código de la operación a realizar (suma)
* Ciclo 5: La ALU calcula el resultado de la suma, que se almacena en el registro de destino.
* 