Conceptos de Arquitectura de Computadoras

Clase 4

Segmentación de Instrucciones

Repertorio sencillo de instrucciones

LW y SQ: load y store --> maquina de una direc. Llevan de un registro a memoria o al reves. Son las unicas que acceden a memoria principal

ADD, SUB, AND, OR --> entran operandos y salen un resultado distinto

BEQ --> como el JZ
SLT --> NUEVO

Instrucción	Pseudocódigo	Descripción
LW	LW RT, inmed(RS)	Carga registro RT desde memoria
SW	SW RT, inmed(RS)	Almacena en memoria desde registro RT
ADD	ADD RD, RS, RT	Suma palabras en registros RS y RT, resultado en RD
SUB	SUB RD, RS, RT	Resta palabras en registros RS y RT, resultado en RD
AND	AND RD, RS, RT	AND de palabras en registros RS y RT, resultado en RD
OR	OR RD, RS, RT	OR de palabras en registros RS y RT, resultado en RD
SLT	SLT RD, RS, RT	Pone 1 en RD si RS es menor o igual que RT
BEQ	BEQ RS, RT, destino	Salta a 'destino' si RS es igual a RT único salto condicional que vamos a usar

En todas las instrucciones salvo las primeras dos, vamos a utilizar TRES registros (corresponde a una máquina de 3 direcciones)

Casi todos utilizan modo de direc directo de registro. Salvo las que acceden a memoria --> modo de direc indirecto con desplazamiento

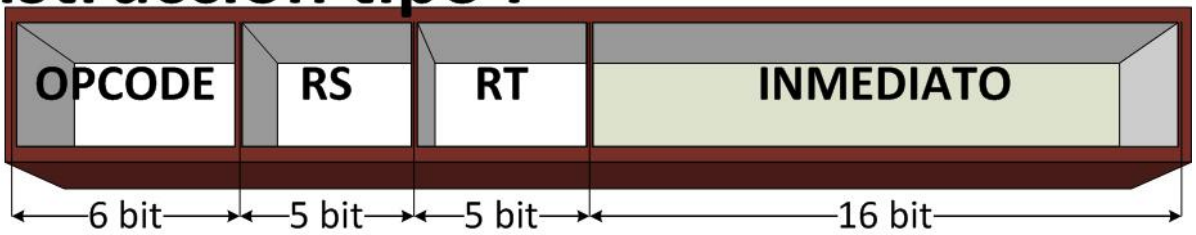
Este procesador tiene condiciones:
-> todas las operaciones hay que hacerlas en registros de la CPU
--> solo acceden a memoria LW y SW
--> dos direccionamientos, inmediato y de registro??

Formato de instrucción

TODAS las instrucciones tienen el mismo tamaño --> 32 bits!!!! Se dividen en función de que operación quiero hacer

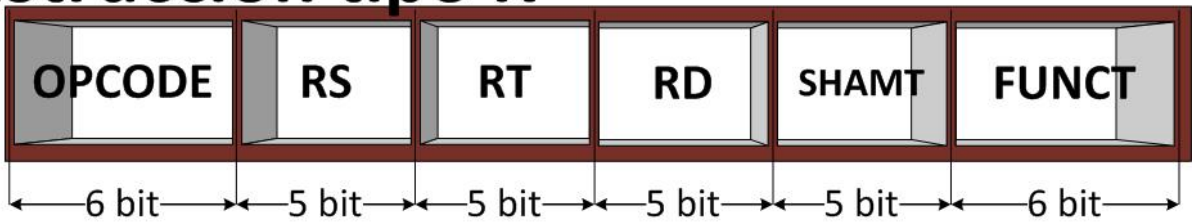
Instrucción tipo I

BEQ --> 2 registros y un valor.
LW y SW



Instrucción tipo R

todos los operandos son registros. RS, RT, RD



6 bits para el CodOp, 3 campos de 5 bits para registros. 1 campo de 5 bits para SHAMT. 1 campo de 6 bits para manejar la UC de la ALU

modo de direc

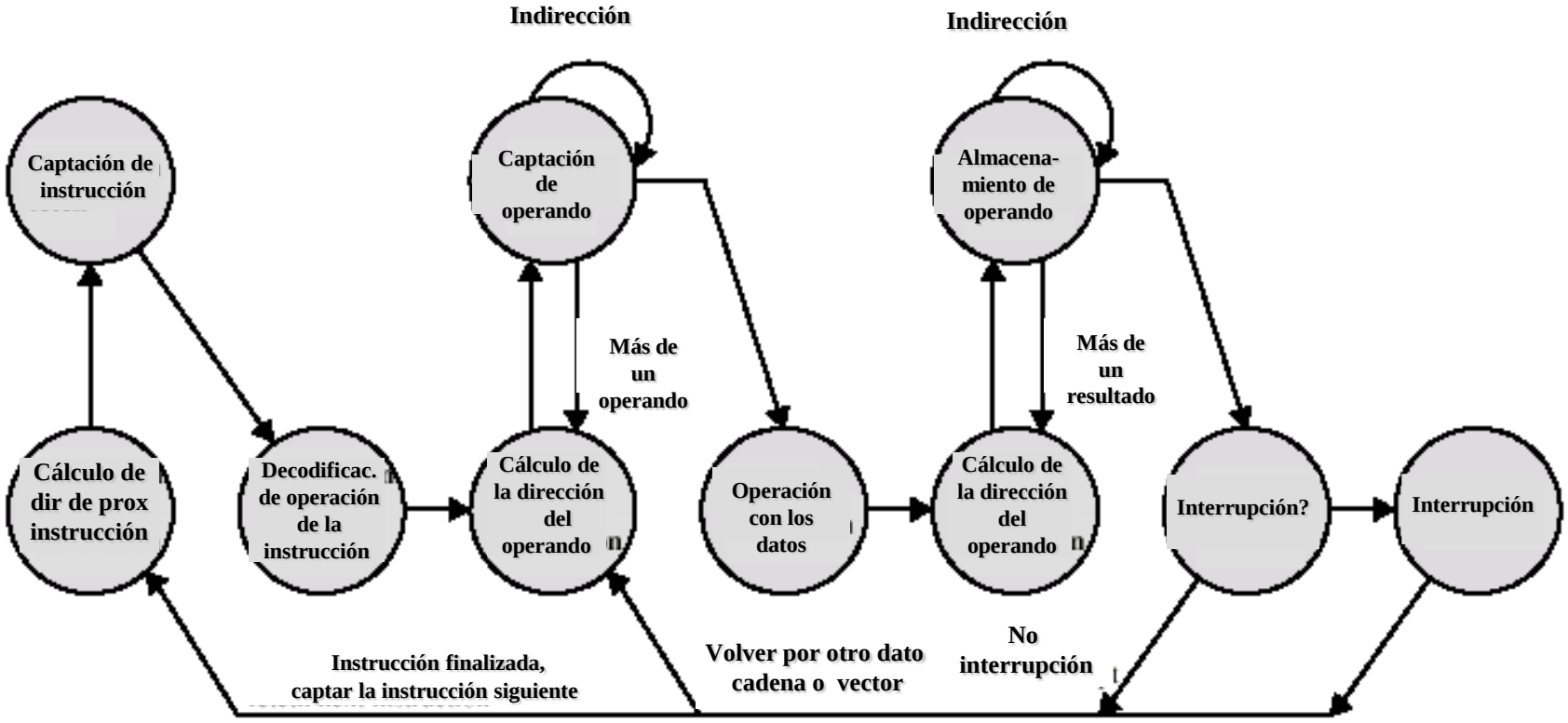
Instrucción tipo J

jmp

Dan la posibilidad de escribir un valor NUMERICO para sumarle al PC.
El PC tiene 32 bits por lo tanto se va a hacer la EXTENSION DE SIGNO para poder sumar bien

Diagrama de estados del ciclo de instrucción

varia como lo implementamos en hardware para sacarle el max provecho



Tareas a realizar por ciclo

tareas a realizar para poder cumplir el ciclo de instrucción anterior

- **Búsqueda (F, *Fetch*)** primera tarea a realizar para completar una instrucción
  accede a la memoria por la instrucción y simultáneamente incrementa el PC
  - Se accede a memoria por la instrucción
  - Se incrementa el PC
- **Decodificación (D, *Decode*)** decodificación de la operación para saber cuantos operandos se necesitan y y luego se realiza la búsqueda de operandos.
  - Se decodifica la instrucción, obteniendo operación a realizar en la ruta de datos
  - Se accede al banco de registros por el/los operando/s (si es necesario) en simultaneo se accede a los registros de la CPU y obtener el o los operandos
  - Se calcula el valor del operando inmediato con extensión de signo (si hace falta)
- **Ejecución (X, *Execute*)** realizar operación con los operandos del paso anterior. Se utiliza la ALU
  - Se ejecuta la operación en la ALU
- **Acceso a memoria (M, *Memory Access*)** se accede a mem para guardar el resultado, si es necesario
  - Si se requiere un acceso a memoria, se accede
- **Almacenamiento (W, *Writeback*)** a la CPU se le ponen los registros para no tener que acceder a memoria
  - Si se requiere volcar un resultado a un registro, se accede al banco de registros

M y W: son de almacenamiento pero puedo acceder para buscar un dato no para almacenar un resultado

pero buscar un operando en memoria no es lo recomendado entonces se intenta tener los operandos siempre en registro

Ruta de Datos en un ciclo

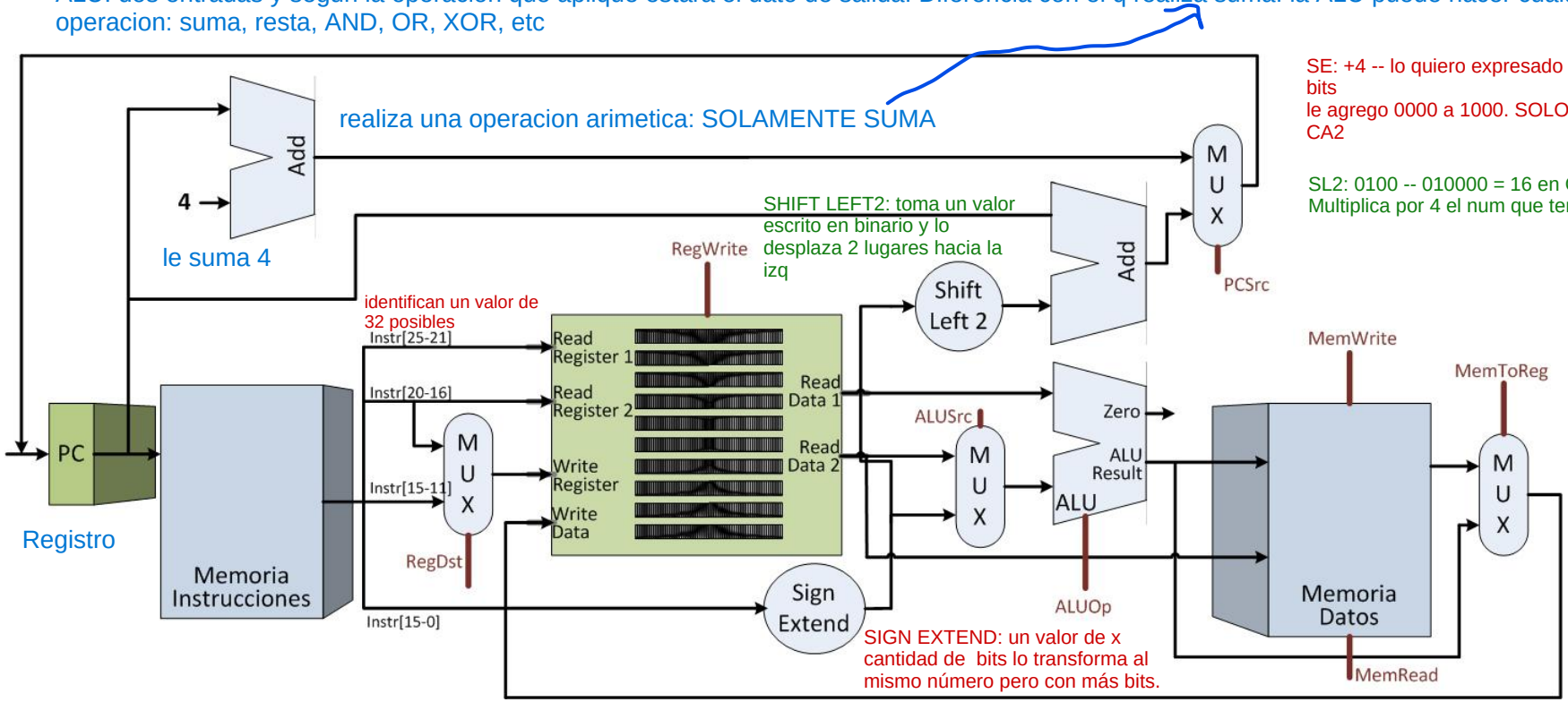
CAMBIOS QUE HACEMOS A LA RUTA DE DATOS VISTA ANTERIORMENTE

Se analiza de izq a derecha. Cuando se llega a la derecha terminó la instrucción

Banco de registro: puedo leer dos registros simultáneos y obtener sus dos valores de forma independiente. Estos dos datos pueden ir a la ALU.

En la ALU se calcula el resultado y se envía al banco de registros

ALU: dos entradas y segun la operación que aplique estará el dato de salida. Diferencia con el q realiza suma: la ALU puede hacer cualq operacion: suma, resta, AND, OR, XOR, etc

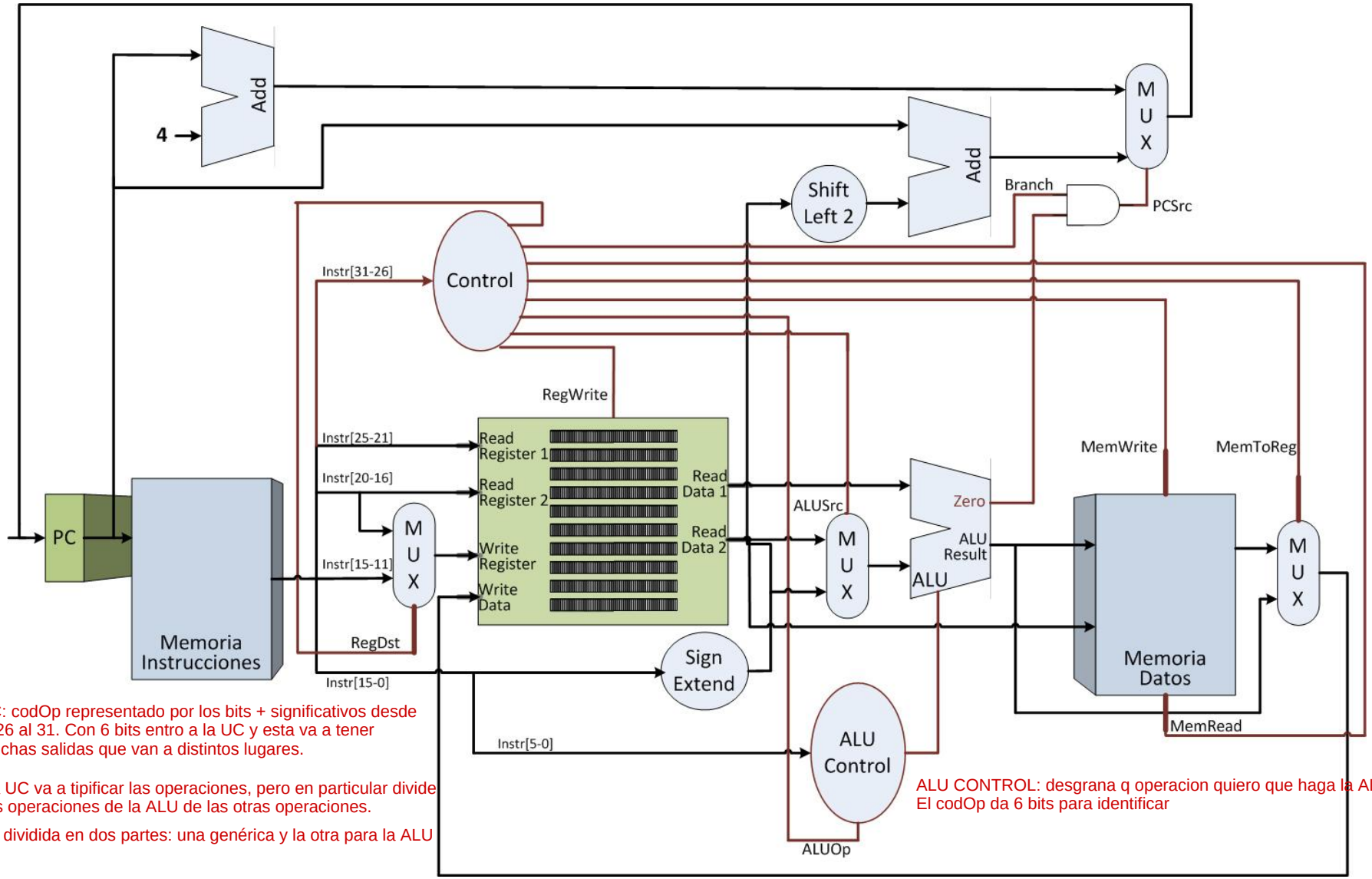


M U X: multiplexor circuito combinatorio que permite de varias entradas posibles tener una sola salida. La salida se elige a través de una señal de control.

Por qué le suma 4 al PC? ANTIGUAMENTE por cómo se tuvo que armar la relación o interacción de la CPU con la memoria. La mem solo tenía una sola manera de conectarse a través del bus de datos

Se va de a 4 bytes a buscar a memoria

Ruta de Datos y unidad de control



UC: codOp representado por los bits + significativos desde el 26 al 31. Con 6 bits entro a la UC y esta va a tener muchas salidas que van a distintos lugares.

La UC va a tipificar las operaciones, pero en particular divide las operaciones de la ALU de las otras operaciones.

UC dividida en dos partes: una genérica y la otra para la ALU

ALU CONTROL: desgrana q operacion quiero que haga la ALU. El codOp da 6 bits para identificar

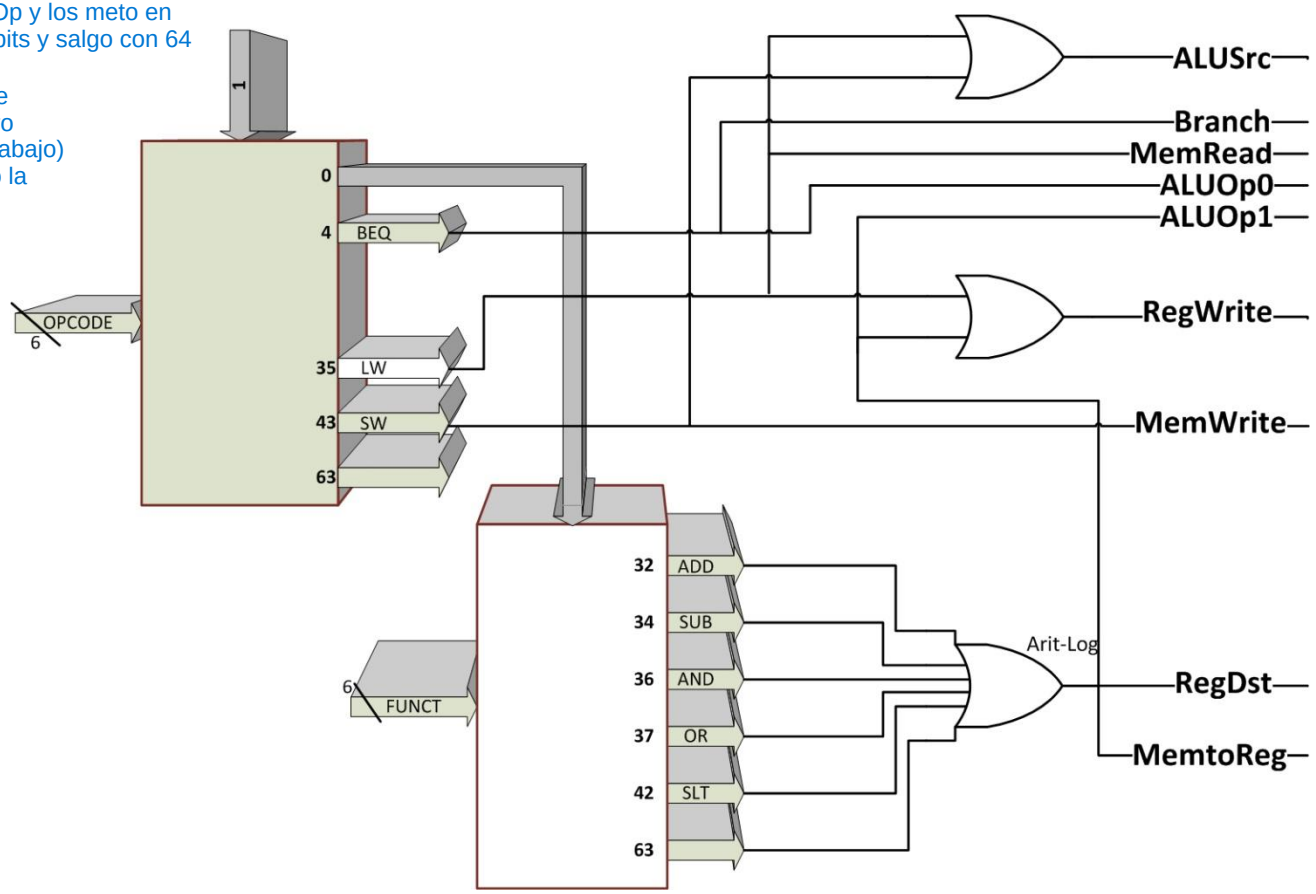
VER ESTO EN LIBROOOOOOOOOOOOOOO. No esta en el stallings

Unidad de control con decodificadores y puertas lógicas

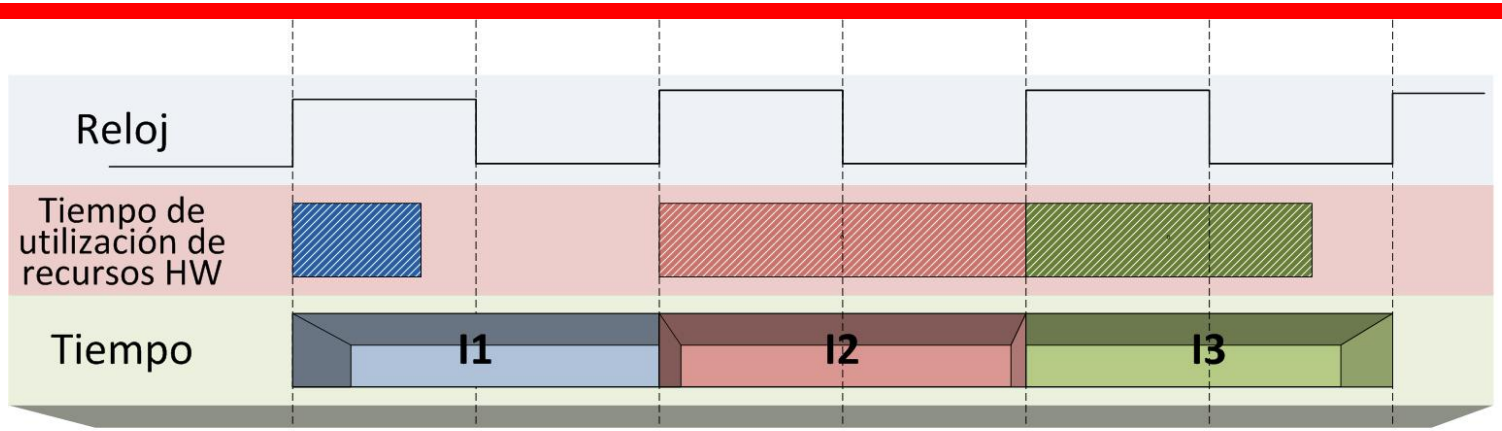
Tomo 6 bits del codOp y los meto en la caja. Entro con 6 bits y salgo con 64

Si los 6 bits son 0, se habilita entrada a otro decodificador (el de abajo) y con eso administro la ALU

Los bits que entran son distintos a los que salen



Comparación monociclo-multiciclo



Asumimos que las instrucciones tardan el mismo tiempo en Von Neumann

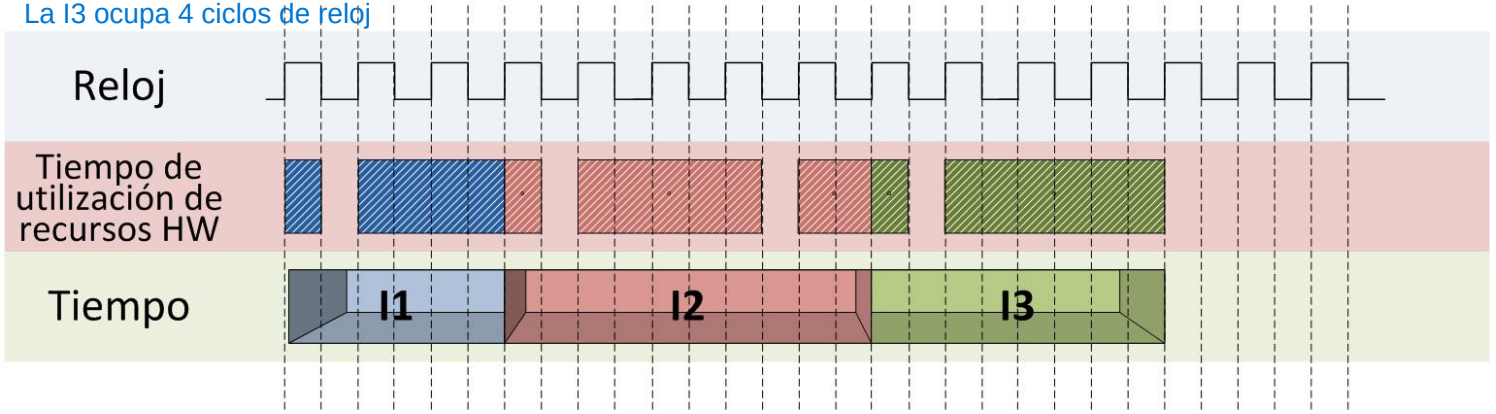
Cada instrucción tardaba 1 ciclo de reloj, pero desperdiciando hardware. Por lo tanto se agregó un hardware controlado por un reloj.

Cada instruccion tiene 5 fases para realizar para controlar lo que paso cuando no se utiliza el hardware.

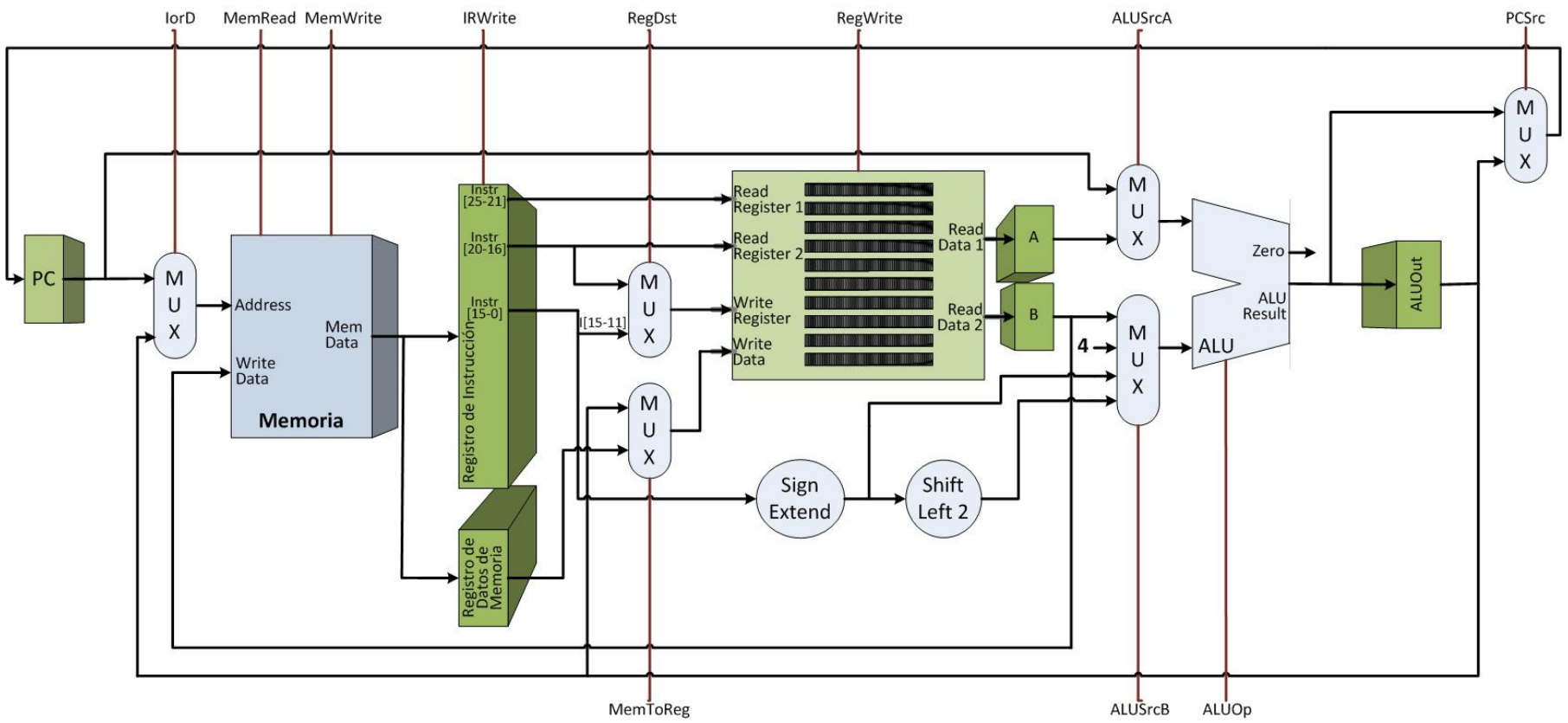
LA I1 ocupa 2 ciclos de reloj

LA I2 ocupa 5 ciclos de un reloj --> es decir ocupa todo el tiempo de la instrucción

La I3 ocupa 4 ciclos de reloj



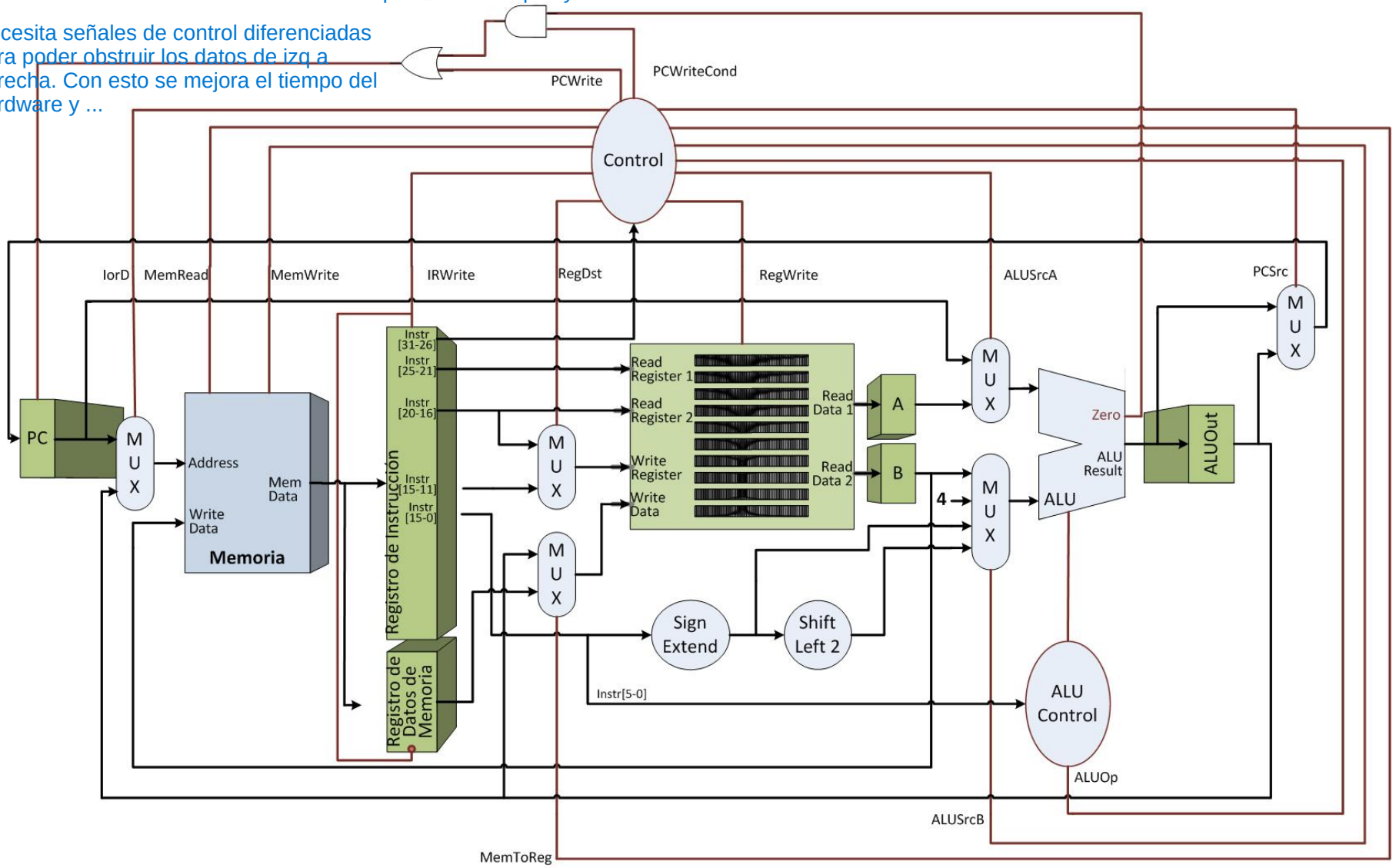
Ruta de datos multiciclo



Ruta de datos y control multiciclo

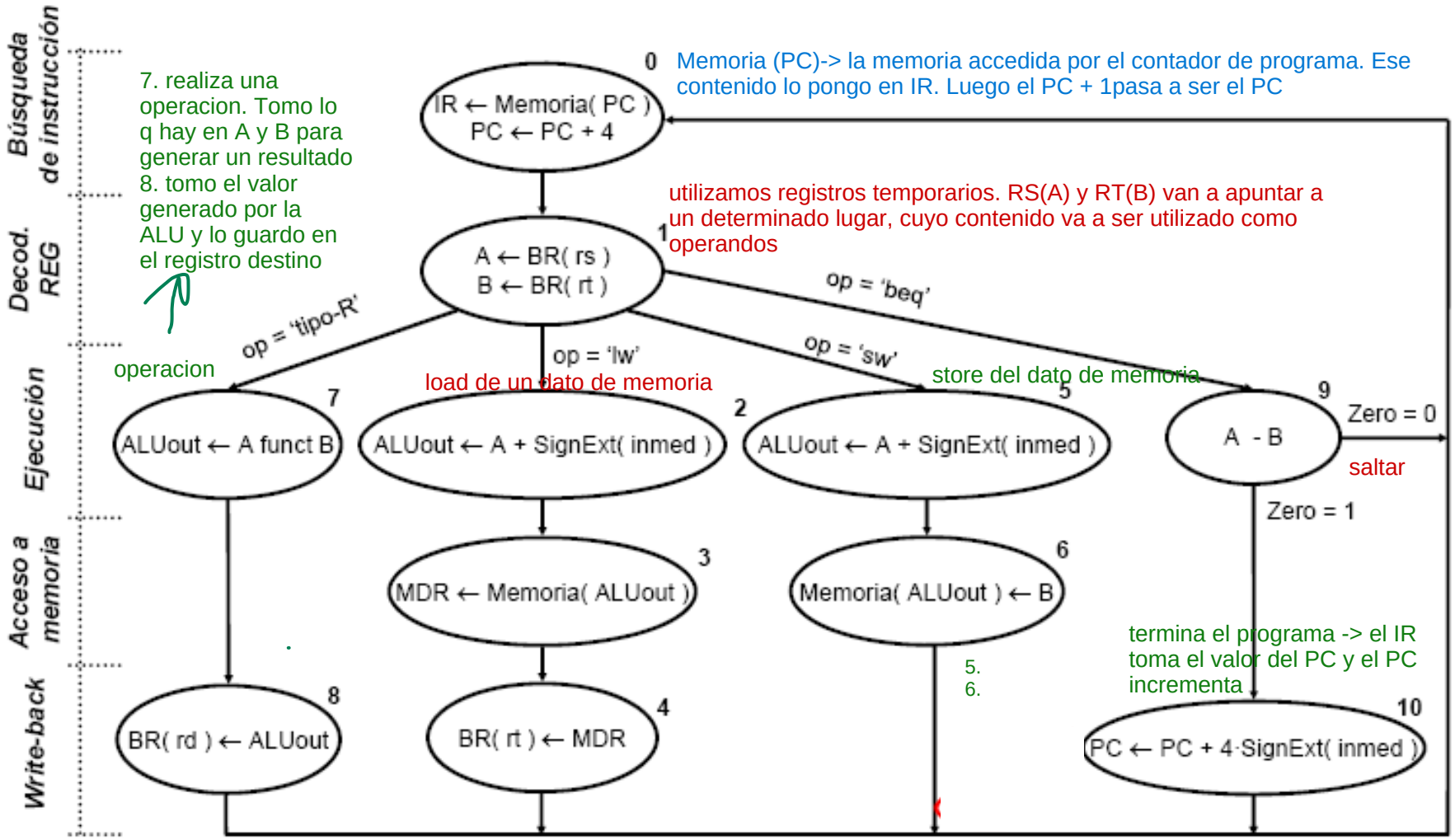
Admnistramos el acceso a este hardware a traves de una unidad de control que va a tomar datos del regidtro de instruccion y en virtud de ellos va a sacar las señales de control p todas las etapas y va a ir secuenciando

Necesita señales de control diferenciadas para poder obstruir los datos de izq a derecha. Con esto se mejora el tiempo del hardware y ...



CI -> se divide en 3 variantes (las de primer filmina)

Diagrama de estados del controlador



2. LA MAS LENTA. el valor de A + la extension de signo van a ser el valor de la ALU out
3. se accede a mem -> sacamos la direc de memoria y la mandamos al bus
4. tomamos lo que vino del bus de datos y colocarlo en el registro q indicaba la instruccion
SE UTILIZA TODO EL HARDWARE Y POR ESO ES LA MAS LENTA. MARCA LOS TIEMPOS DE TODAS LAS EJECUCIONES

Segmentación de cauce: Conceptos básicos

- La segmentación de cauce (*pipelining)* es una forma particularmente efectiva de organizar el hardware de la CPU para realizar más de una operación al mismo tiempo.
- Consiste en descomponer el proceso de ejecución de las instrucciones en fases o etapas que permitan una ejecución simultánea.
- Explota el paralelismo entre las instrucciones de un flujo secuencial.

Como se van a tratar las instrucciones para poder cumplirlas
Pipelining: FORMA de organizar el hardware q tenemos disponible en la CPU para que se pueda realizar mas de una tarea en simultaneo.

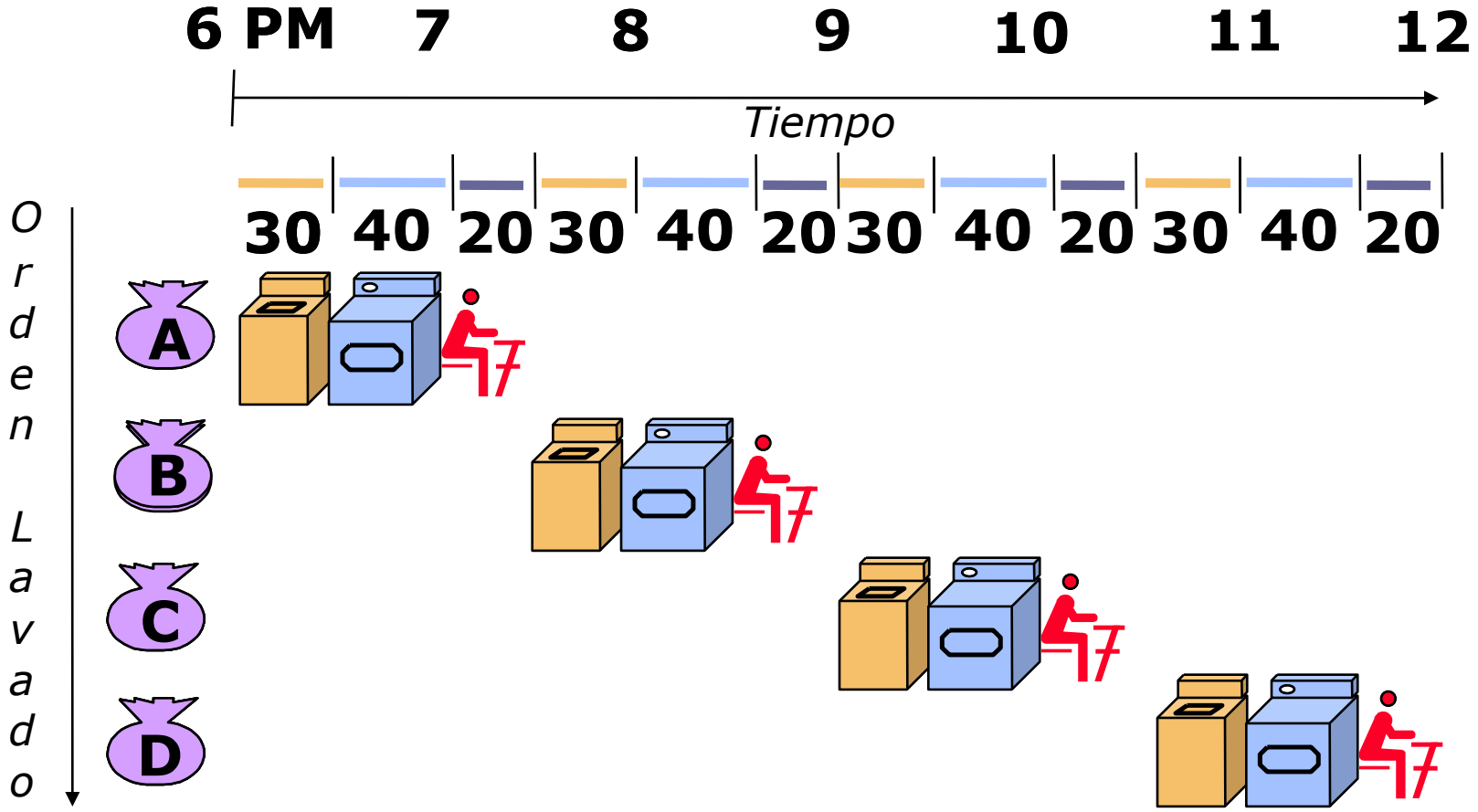
Permite agregar la PARALELIZACION de la ejecución de las instrucciones. Abandonamos la ejecución secuencial de instrucciones. Esto nos da otro salto de velocidad de la ejecución.

Ejemplo de estrategia (1)

- Similar a la línea de armado en una planta de manufactura.
- El producto pasa por varios estados en el proceso de producción.
- Por lo tanto, varios productos pueden ser manipulados simultáneamente (cada uno en estados distintos).
- Se puede comenzar el proceso nuevamente (entrada a la línea de producción) antes de que salga el producto final de la misma.

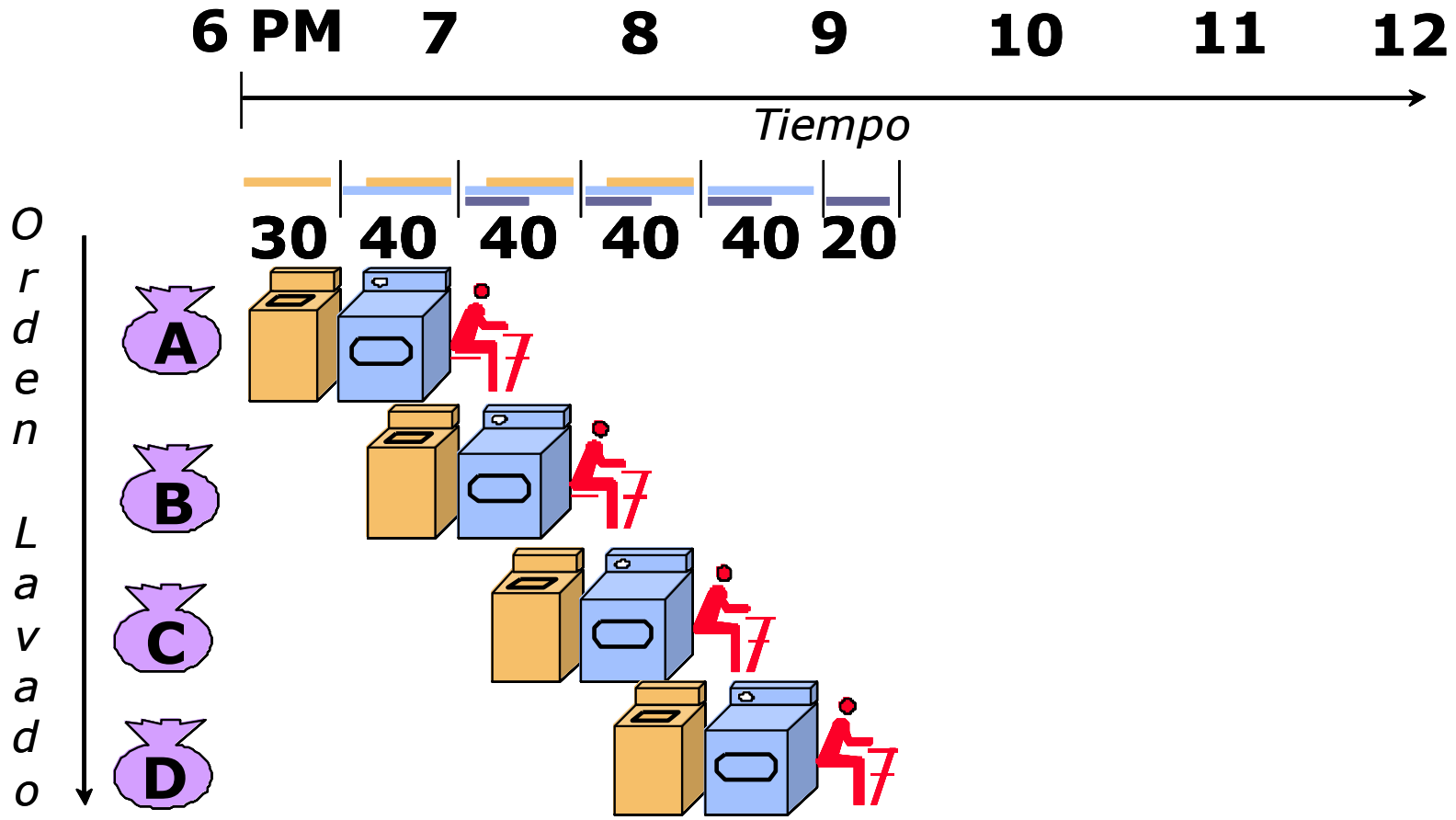
Ej. de estrategia (2)

Lavandería secuencial: ¡mal negocio!



Ej. de estrategia (3)

Lavandería segmentada: ¡buen negocio!



Características

- La segmentación es una técnica de mejora de prestaciones a nivel de diseño hardware.
- La segmentación es invisible al programador.
- Necesidad de uniformizar las etapas.
  - Al tiempo de la más lenta

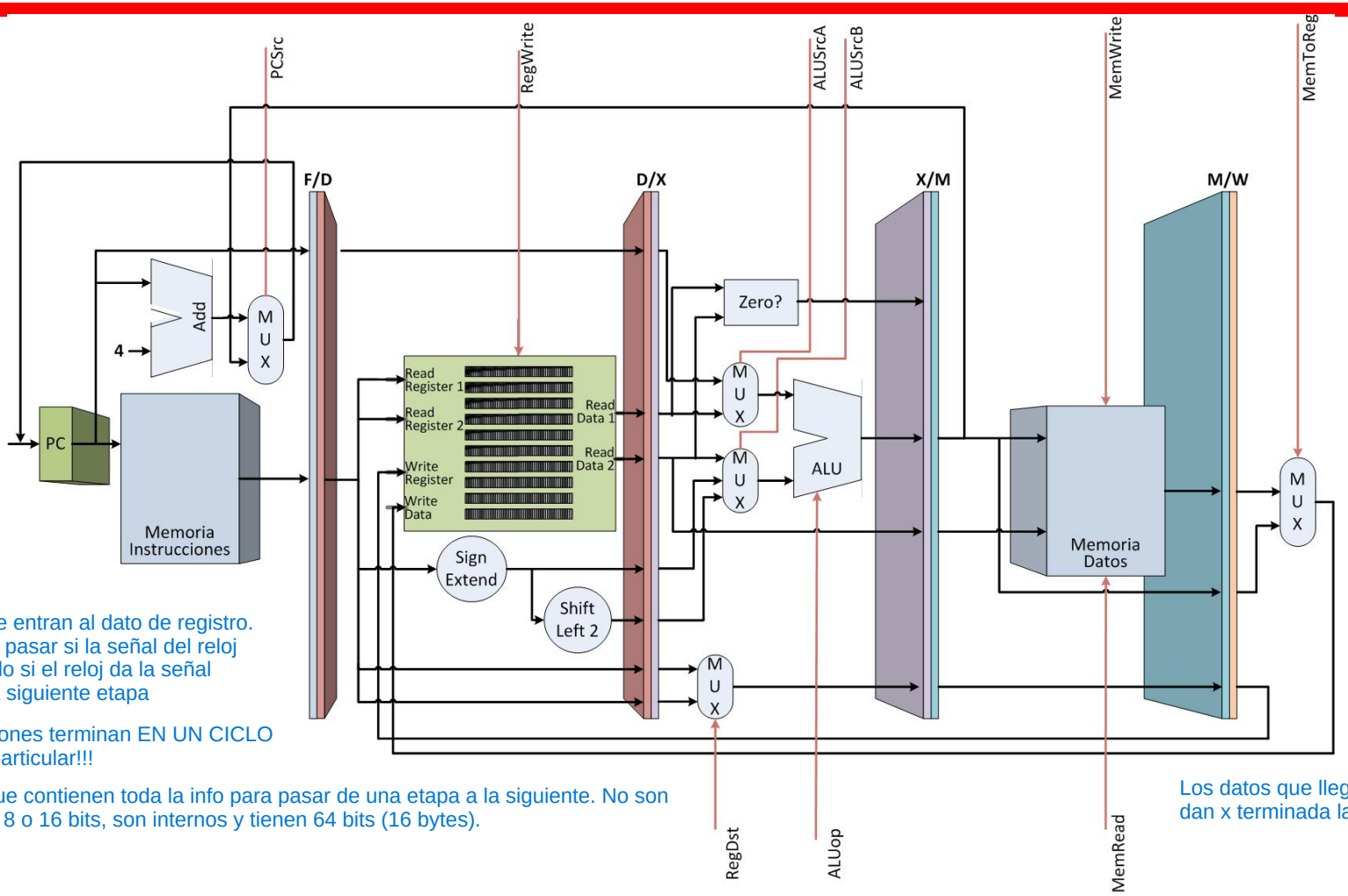
todas las etapas que tengamos vamos a tratar de que tengan aproximadamente el mismo tiempo

- El diseño de procesadores segmentados tiene gran dependencia del repertorio de instrucciones.

depende del repertorio de instrucciones que utilice la maquina

Ruta de datos segmentados

segmentación: separar etapas con registros de segmentación, si estos no están no existe la segmentación
Los registros q estaban en una etapa van a pasar a otra
Reloj trabaja sobre todas las etapas para que estas puedan trabajar una encima de la otra



D: Datos que entran al dato de registro. Van a poder pasar si la señal del reloj aparece. Solo si el reloj da la señal pasaran a la siguiente etapa

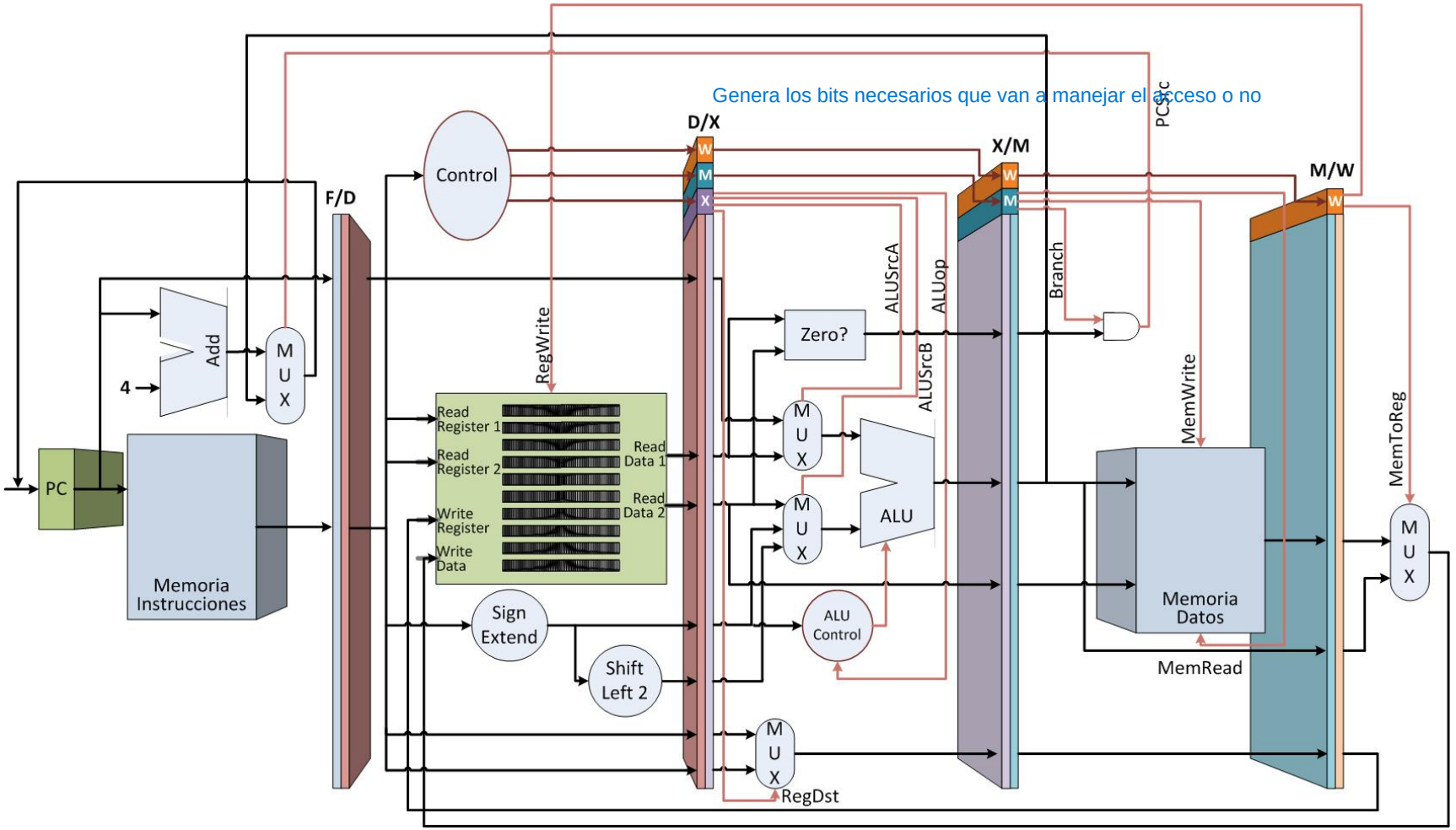
Las instrucciones terminan EN UN CICLO DE RELOJ particular!!!

Registros que contienen toda la info para pasar de una etapa a la siguiente. No son registros de 8 o 16 bits, son internos y tienen 64 bits (16 bytes).

Los datos que lleguen a lo naranja dan x terminada la instruccion.

Ruta de datos y control segmentado

Bits de la unidad de control -> se agregan a cada uno de los registros de segmento.
(los naranjas)



Prestaciones del cauce segmentado

Teórica: El máximo rendimiento es completar una instrucción con cada ciclo de reloj.

Si K es el número de etapas del cauce $\Rightarrow$

Vel. procesador segmentado = Vel. secuencial x K

La ejecucion es igual que antes. No gana en tiempo de ejecución

El incremento potencial de la segmentación del cauce es proporcional al número de etapas del cauce.

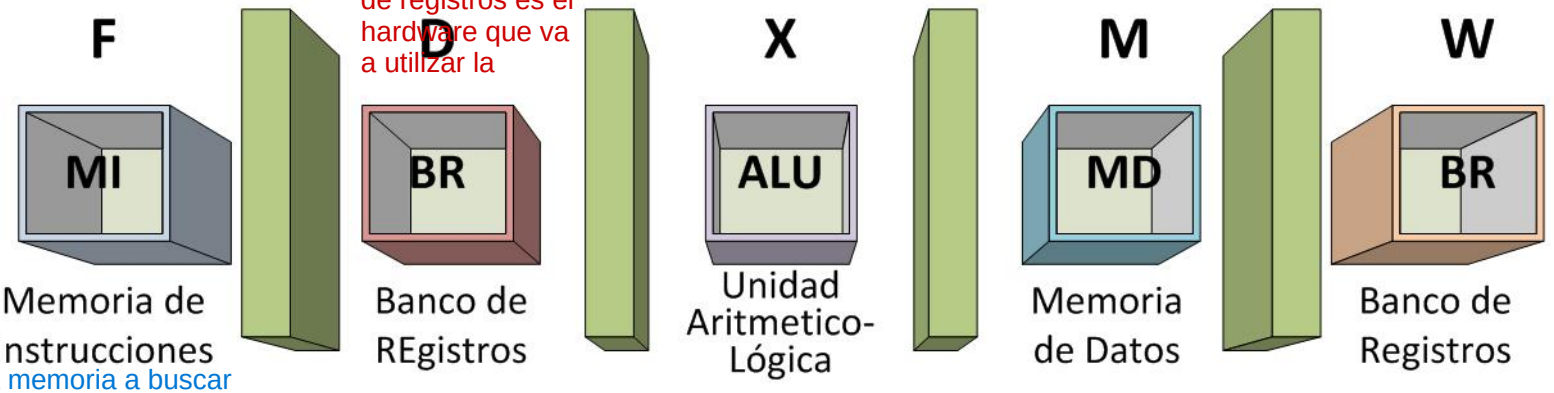
Incrementa la productividad (throughput), pero no reduce el tiempo de ejecución de la instrucción

Ejemplo de segmentación

cantidad de etapas - 1 = cantidad de registros que tengo que poner

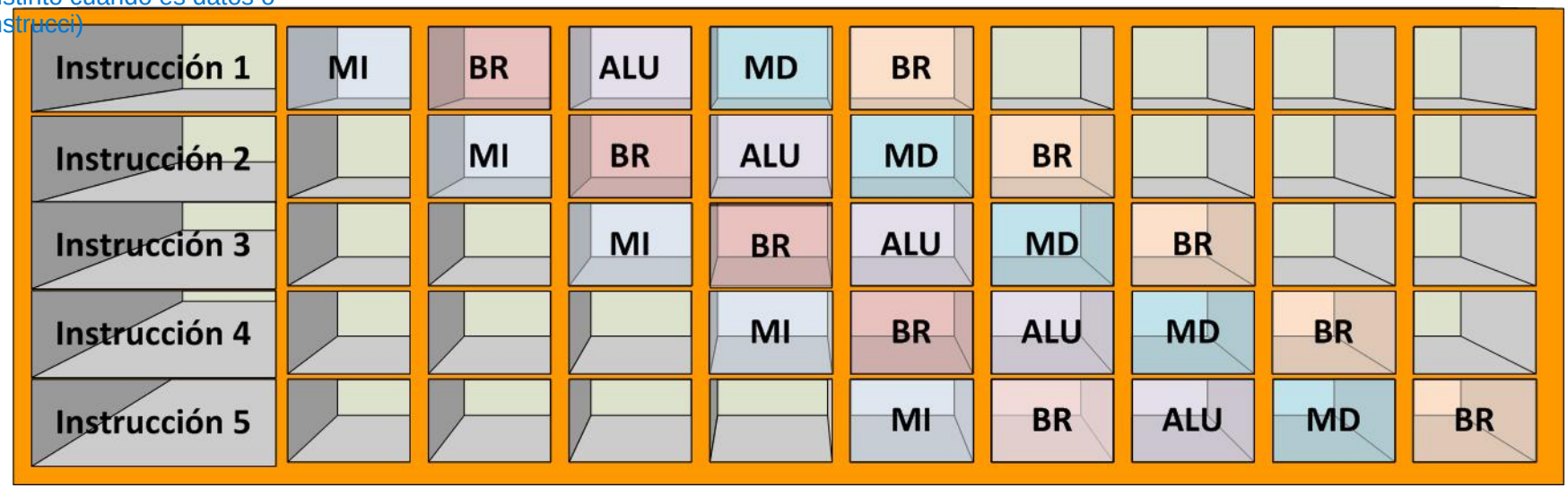
accede al banco de registros es el hardware que va a utilizar la

hacemos uso del banco de registros, para escribirlos



accede a memoria a buscar instrucciones (es la misma memoria pero lo aclaramos distinto cuando es datos o instrucci)

accedo para leer o acceder un dato



en cada ciclo de reloj permitimos que se vaya ingresando a la siguiente instruccion -> esto es la ganancia del procesador. Hasta que lleguemos al 5 ciclo de reloj no hay ninguna instruccion que haya terminado. A partir de la siguiente etapa, una instruccion fue terminando en cada ciclo de reloj!!!!

Análisis de la segmentación (1)

Suposiciones:

- Todas las tareas duran el mismo tiempo.
- Las instrucciones siempre pasan por todas las etapas.
- Todos las etapas pueden ser manejadas en paralelo.

Análisis de la segmentación (2)

Problemas:

- No todas las instrucciones necesitan todas las etapas.
  - SW RT, inmed(RS) ; no utiliza W
    - En MSX88: un MOV AX, mem ; no requiere X
- No todas las etapas pueden ser manejadas en paralelo.
  - F y M acceden a memoria

No se puede acceder a mem y al mismo tiempo estar guardando un dato porq tenemos UN solo bus

- No se tienen en cuenta los saltos de control.

Atascos de un cauce (stall)

stall: puede no salir ninguna instruccion porq se traba el cauce. Esto se da porque los elementos de una etapa no pueden pasar a la siguiente por diferentes motivos.

Se tienen que evitar

Situaciones que impiden a la siguiente instrucción que se ejecute en el ciclo que le corresponde.

- Estructurales
  - Provocados por conflictos por los recursos
- Por dependencia de datos
  - Ocurren cuando dos instrucciones se comunican por medio de un dato (ej.: una lo produce y la otra lo usa)
- Por dependencia de control
  - Ocurren cuando la ejecución de una instrucción depende de cómo se ejecute otra (ej.: un salto y los 2 posibles caminos)

porq vamos a querer utilizar el mismo hardware con etapas distintas. Ejemplo de fetch y acceso a memoria.

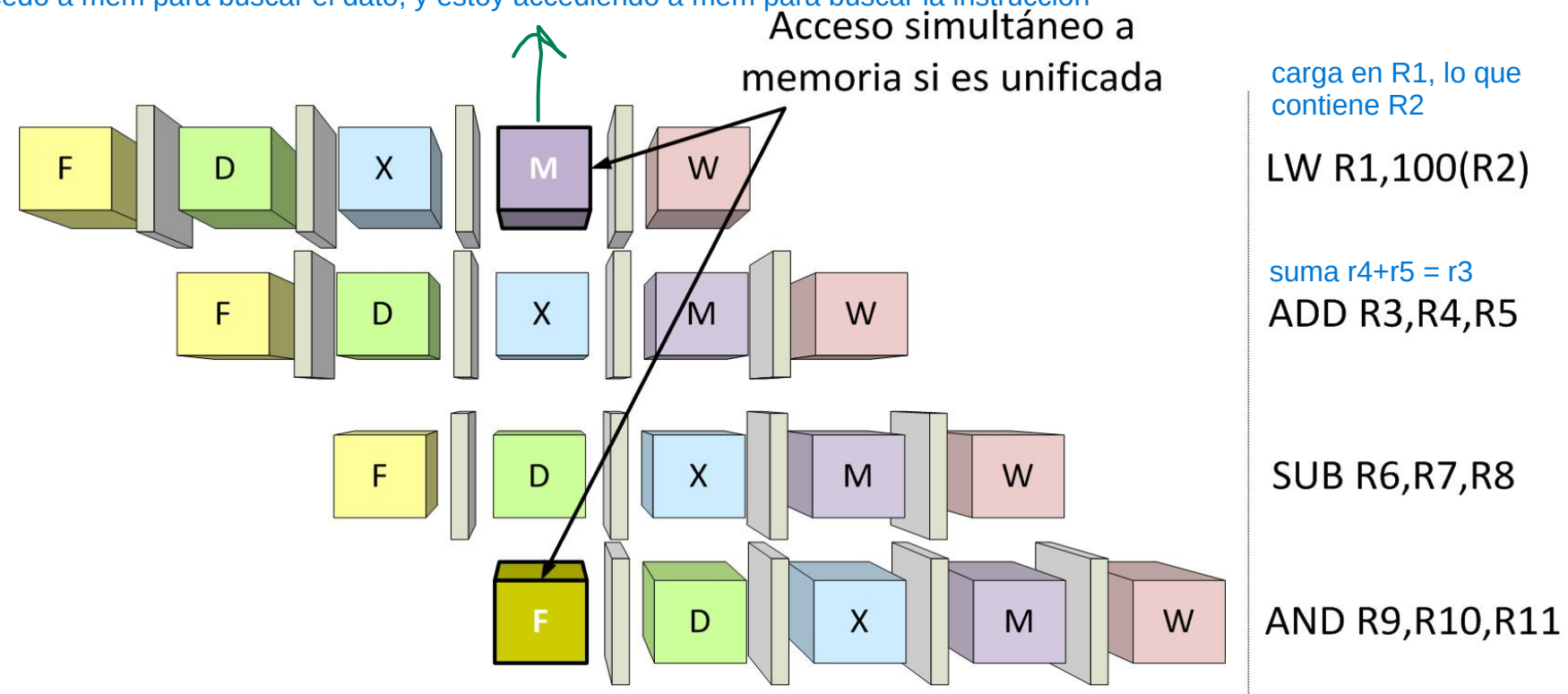
Dos instrucciones que escribiamos en secuencia, utilizan el mismo dato.
Ej: una insstruccion lee un dato en mem y otro lo toma p realizar una operacion

La ejecucion de una instruccion depende de como se haya ejecutado. Ej salto incondicional: hay que esperar a ver como se resuleve una cosa para poder continuar

Riesgos estructurales

Dos o mas instrucciones necesitan utilizar el mismo recurso hardware en el mismo ciclo.

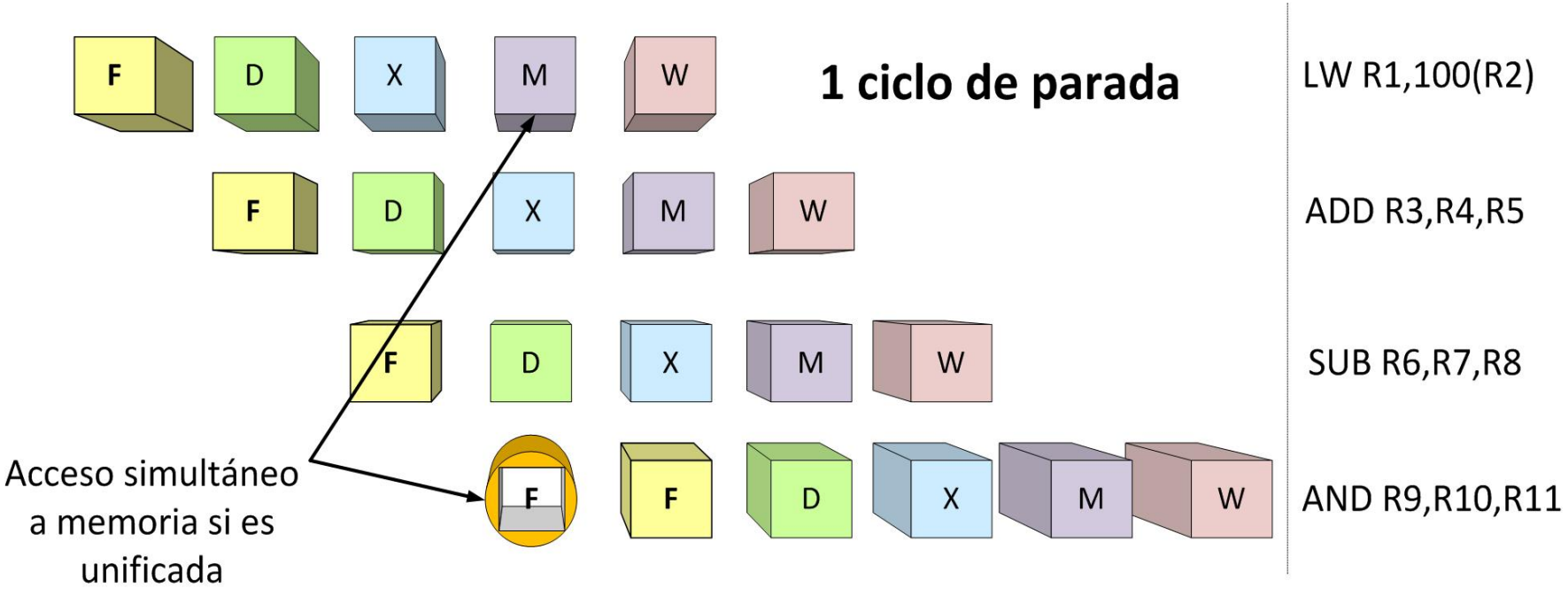
accedo a mem para buscar el dato, y estoy accediendo a mem para buscar la instruccion



Riesgos estructurales (2)

Resolución ante el riesgo:

Por lo tanto la ult instruccion qeu queria acceder a memoria, espera al proximo ciclo de reloj, por lo tanto va a haber un ciclo de reloj que no termine una instruccion



Riesgos por dependencias de datos

- Condición en la que los operandos fuente o destino de una instrucción no están disponibles en el momento en que se necesitan en una etapa determinada del cauce.

Tipos de dependencias de datos

read after write

- Lectura después de Escritura (RAW, dependencia verdadera)
  - una instrucción genera un dato que lee otra posterior

más común

- Escritura después de Escritura (WAW, dependencia en salida)

write after write

  - una instrucción escribe un dato después que otra posterior
  - sólo se da si se deja que las instrucciones se adelanten unas a otras

las dos son escrituras de un valor

No va a pasar porq no lo vamos a hacer

- Escritura después de Lectura (WAR, antidependencia)

write after read

  - una instrucción modifica un valor antes de que otra anterior que lo tiene que leer, lo lea
  - no se puede dar en nuestro cauce simple

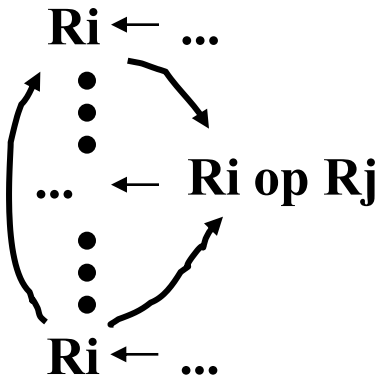
una instruccion termine antes q otra pero ademas modifica los valores. No va a pasar porq no lo vamos a hacer

Tipos de dependencias ...(2)

Secuencia de instruccion escrita en RTL, no hay operaciones pero si una secuencia de resultados posibles

se da en funcion de cuan cerca esten las dos instrucciones

ri= resultado



RAW — Dependencia real

para evitarlo, se pone un NOP

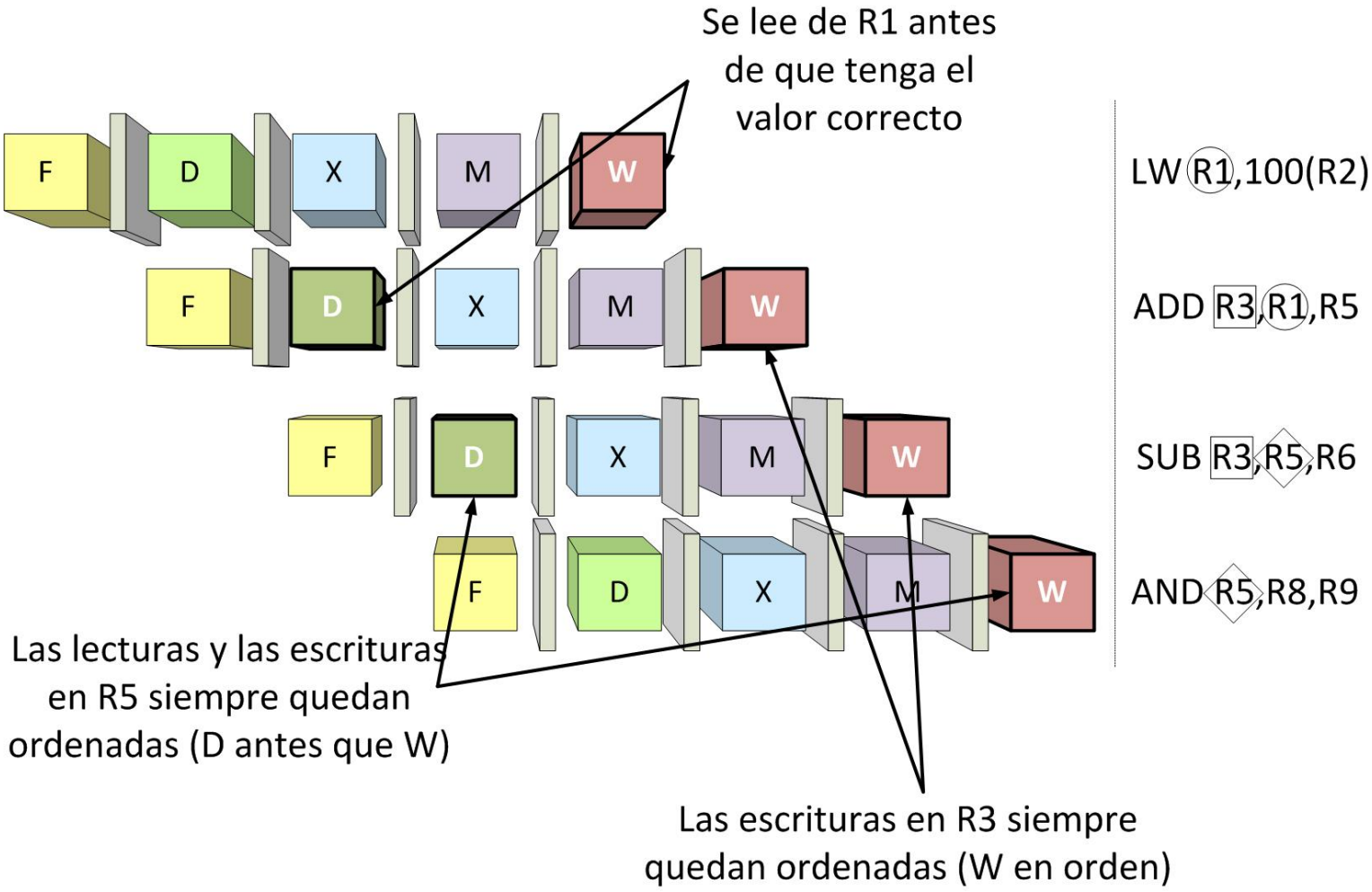
Dependencia de salida — WAW

operacion

WAR — Antidependencia

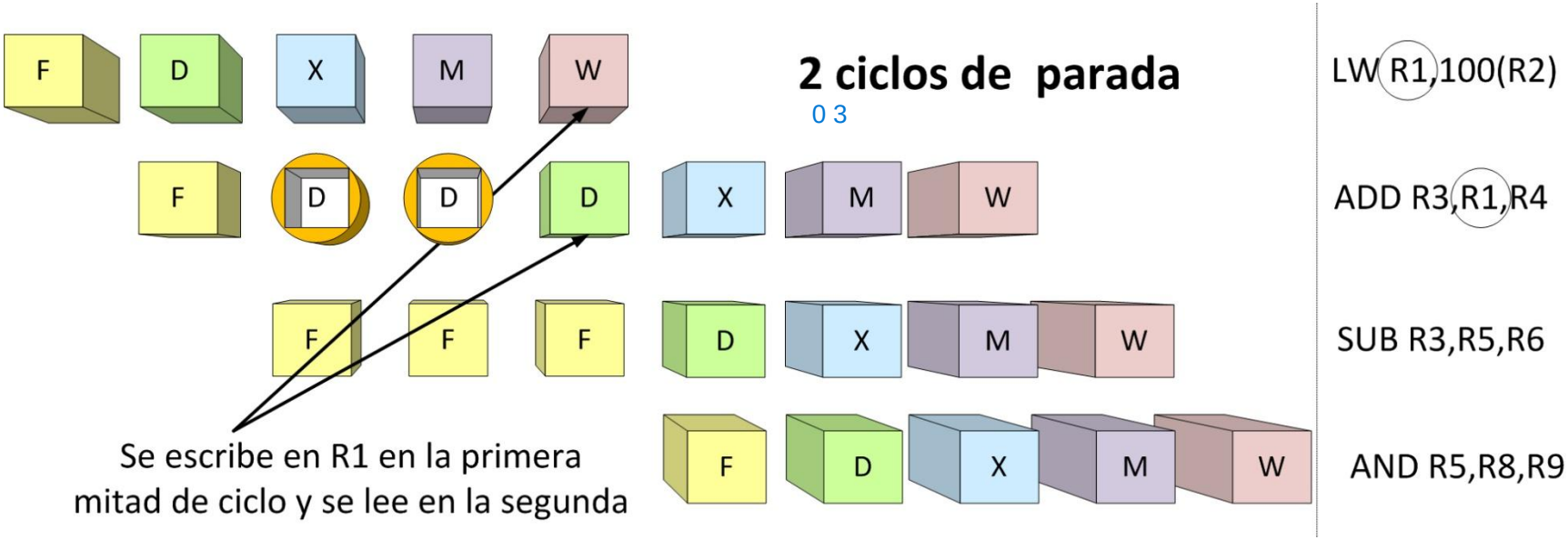
se genera un rsultado y lo pone en Ri

Riesgos por dep... datos (2)



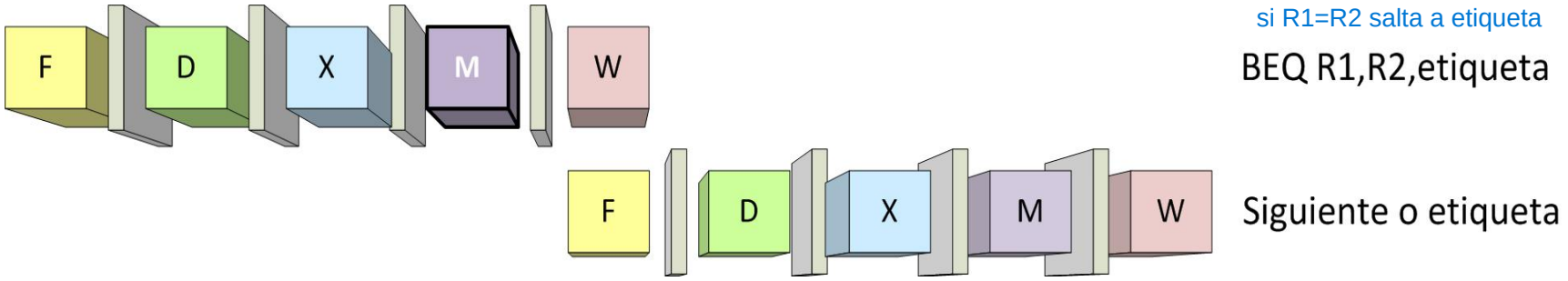
Riesgos por dep... datos (3)

Resolución ante el riesgo:



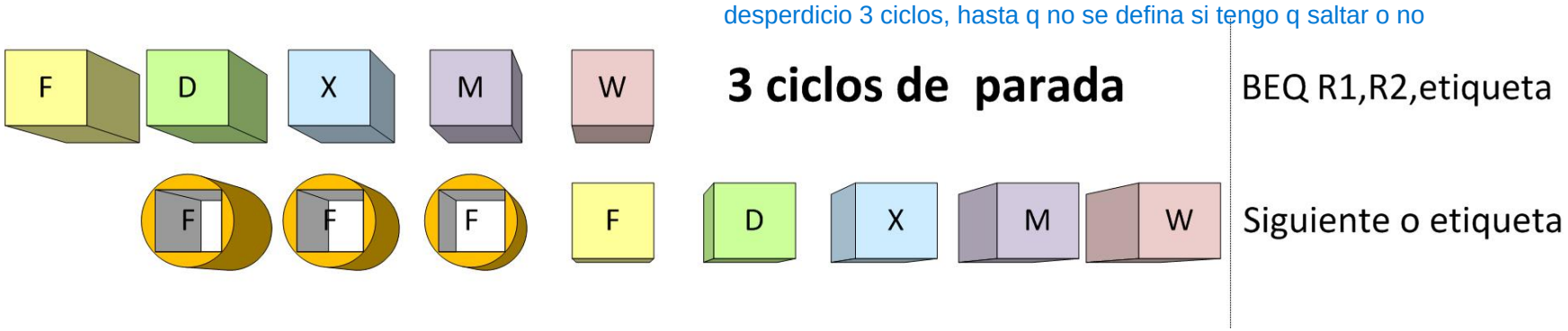
Riesgos de control (o de instrucciones)

Una instrucción que modifica el valor del PC no lo ha hecho cuando se tiene que comenzar la siguiente.



Riesgos de control (2)

Resolución ante el riesgo:



Lectura básica

- *Organización y Arquitectura de Computadores*, W. Stallings, Capítulo 11.