

ETC-Tema-2.pdf



TurbApuntPatata



Estructura de Computadores



2º Grado en Ingeniería Informática



Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Valencia

BBVA

1/6
Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

Ábrete la Cuenta Online de BBVA y
llévate 1 año de **Wuolah PRO**

cómo??



Ventajas Cuenta Online de BBVA

0€
Sin comisión de administración o
mantenimiento de cuenta.
(0 % TIN 0 % TAE)

0€
Sin comisión por emisión y
mantenimiento de Tarjeta
Aqua débito.

0
Sin necesidad de domiciliar
nómina o recibos.

Las ventajas de **WUOLAH PRO**

✖ Di adiós a la publi en
los apuntes y en la web

✖ Descarga carpetas
completas de un tirón

✖ Acumula tickets
para los sorteos

cómo???



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

ESTRUCTURA DE COMPUTADORES

Tema 2

ventajas

PRO



Di adiós a la publi en los apuntes y en la web



Acumula tickets para los sorteos



Descarga carpetas completas

estudia sin publi
WUOLAH PRO

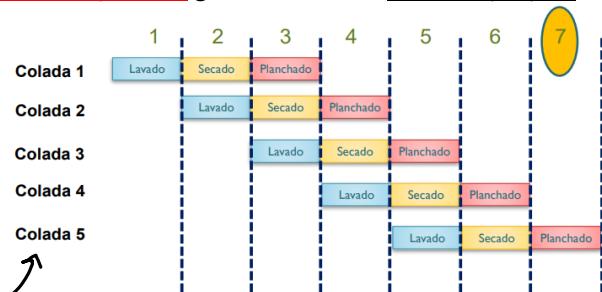


TEMA 2

Segmentación

Descomposición de un sistema que ejecuta un determinado proceso en **varias etapas**, de manera que: Cada etapa se ocupa de una parte del proceso global utilizando recursos propios, el **proceso global** requiere la aplicación ordenada de todas las etapas y todas las etapas trabajan simultáneamente (cada una en un proceso distinto)

Su Objetivo es **Aumentar el paralelismo** (temporal) de los procesos y, por tanto, aumentar la productividad



Tiempo de procesar n datos (5 coladas) en k etapas (3 etapas): $T_{seg} = k + n - 1$

Proceso de Segmentación

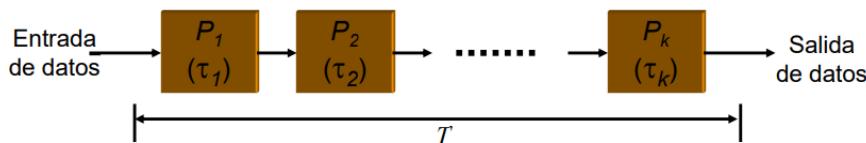
Entrada

Cada segmento es un **Operador lógico P** que opera sobre *datos de entrada*. P se considera constituido por k etapas P_i que operan de manera secuencial sobre los datos.

Consideraciones temporales

- Retardo del circuito (Total, suma de cada etapa): T
- Cada etapa P_i tiene necesidades temporales (retardo) distintas: τ_i

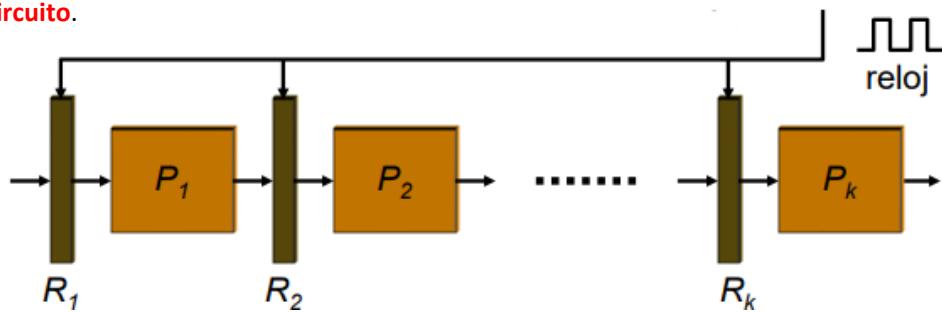
$$T = \sum_{i=1}^k \tau_i$$



Salida (circuito segmentado)

Si el circuito tiene k etapas se dice que tiene una **profundidad de segmentación "k"**

Registro de etapa (R_i, staging latch): Cada etapa va precedida de un **registro de etapa** cuyo tiempo es T_r . Cada registro escribe lo que le diga el proceso que le preceda, todos los registros de etapa **se actualizan en el mismo flanco de reloj**. Cada Registro recoge los datos producidos por la etapa $i-1$ y los suministra datos a la etapa i . Los registros de etapa **incrementan el coste del circuito**.



2



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

Ábrete la Cuenta Online de BBVA y llévate 1 año de Wuolah PRO



cómo??

Las ventajas de Wuolah PRO



Di adiós a la publi en
los apuntes y en la web



Descarga carpetas
completas de un tirón



Acumula tickets
para los sorteos

Ventajas Cuenta Online de BBVA

0€

Sin comisión de administración o
mantenimiento de cuenta.
(0 % TIN 0 % TAE)

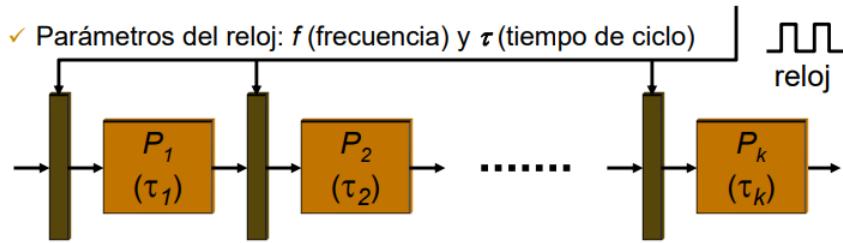
0€

Sin comisión por emisión y
mantenimiento de Tarjeta
Aqua débito.

0

Sin necesidad de domiciliar
nómina o recibos.

Sincronización del circuito



Parámetros de reloj

- f (frecuencia)
- T (tiempo total)
- τ_i (tiempo de ciclo, los cortitos)
- T_r (Tiempo de registro)

El retardo de los **registros TR** debe considerarse en el tiempo de ciclo, por lo que “ τ ” será el mayor $\tau_i + T_r$ (El tiempo máximo que tardará en hacer una etapa)

El más lento marca el ritmo de la segmentación

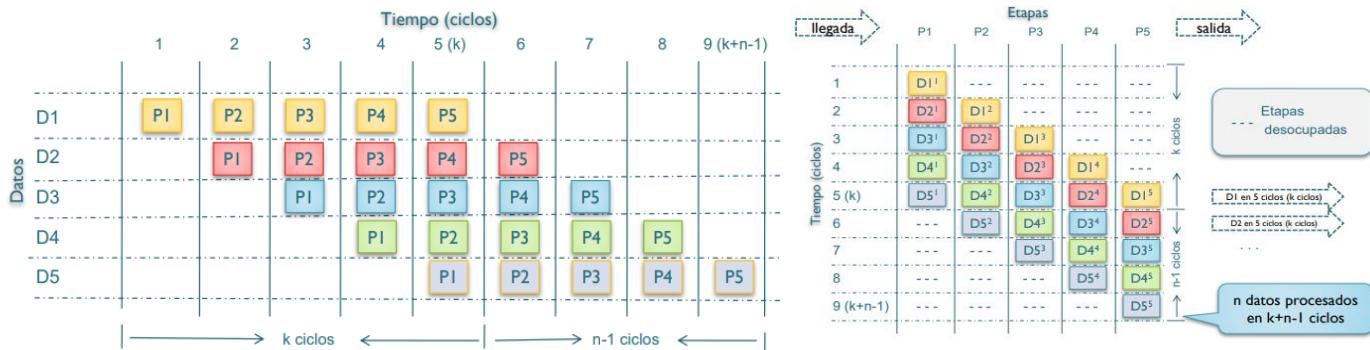
$$\tau = \max\{\tau_i\} + T_r$$

$$f = \frac{1}{\tau}$$

Límite: El “límite” de segmentar es cuando te pasas segmentando e incluso el T_r es más grande que $\max\{\tau_i\}$. En caso de ser una sola operación la segmentación penaliza el costo T

$$T \leq k \times \tau$$

DIAGRAMAS ETAPAS/TIEMPO



Prestaciones de la Segmentación

Productividad: trabajo realizado por unidad de tiempo (**instrucciones por segundo**)

$$X_{seg}(n) = \frac{n}{T_{seg}(n)} = \frac{n}{(n+k-1)\tau} \text{ Datos/tiempo} \quad X_{no\,seg}(n) = \frac{n}{n \cdot T} = \frac{1}{T} = f \text{ no seg}$$

Límite Teórico: con la n suficientemente grande se considera una operación por ciclo:

$$X(\infty) = \frac{1}{\tau} = f \text{ segmentada}$$

Aceleración: Ganancia de velocidad respecto al circuito no segmentado

$$S(n) = \frac{T_{ns}(n)}{T_{seg}(n)} = \frac{n \cdot T}{(n+k-1) \cdot \tau} \quad S(\infty) = \frac{T}{\tau}$$

La aceleración ideal es el número de etapas "k"

Ganancia teórica máxima

En el **caso ideal**, cuando $T = k \cdot \tau$, la ganancia sería $S = k$ (**número de etapas**). Ello se cumpliría sólo si **todas las etapas** tuvieran **igual retardo** (τ_i) y el **retardo de los registros de segmentación** (Tr) fuera **igual a cero**

En principio, y con suficientes datos, a menor τ mayor aceleración

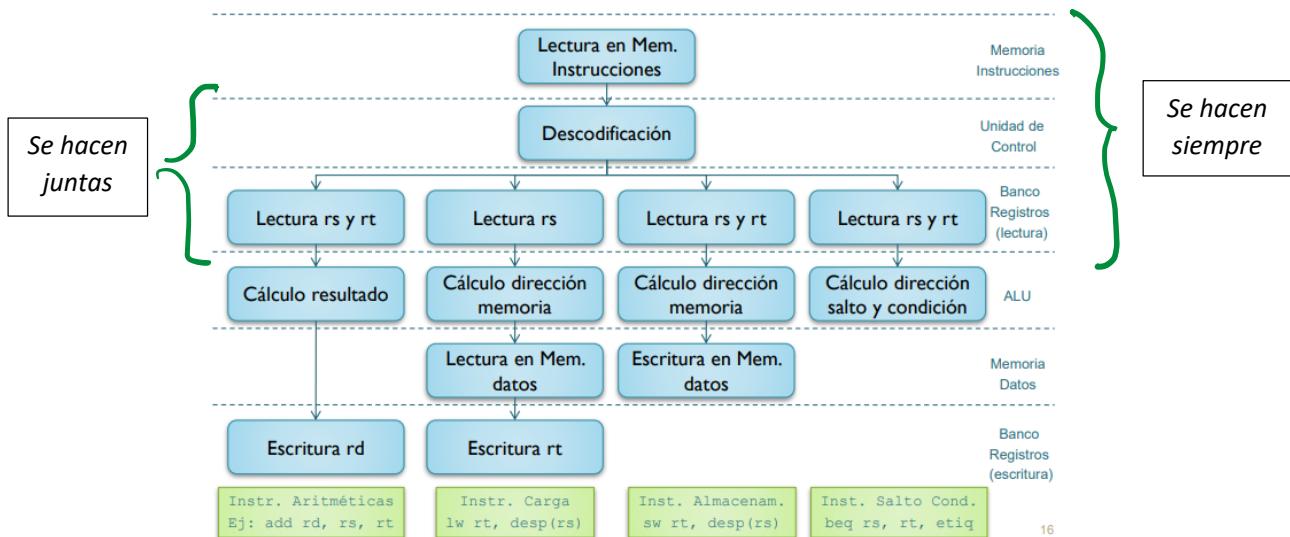
La frecuencia de reloj

Tiempo de ciclo de reloj: $T_{seg} = \max(\tau_i) + Tr$

El período mínimo de reloj: $\tau = \max(\tau_i) + Tr$

Incrementar la productividad (1/ τ): Se reducirá el **retardo máximo de etapa** (**aumentando k y manteniendo los retardos equilibrados**) y el **retardo de los registros**

Segmentación de la Ruta de Datos



Etapas comunes a todas las instrucciones

LI: Etapa de **lectura de instrucción** (e incremento del PC)

DI: Etapa de **decodificación de instrucción** (y lectura de registros)

Etapas que dependen del tipo de instrucción

EX: Etapa de **ejecución** → **Instrucciones de cálculo del resultado, Load y Store:** cálculo dirección de memoria **Instrucciones de salto:** **Cálculo** de la **dirección de salto y de la condición de salto**

M: Etapa de **memoria** → **Load y Store:** acceso a la memoria, **Instruc. de cálculo y salto:** **nada**

ER: Etapa de **escritura de registro** → **Instrucciones de cálculo y Load :** escritura del registro **Store e instrucciones de salto:** **nada**

cómo??
↗



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

ventajas

PRO



Di adiós a la publi en los apuntes y en la web



Acumula tickets para los sorteos



Descarga carpetas completas

estudia sin publi
WUOLAH PRO

Los registros de segmentación

Hacen falta cuatro, cada uno está estructurado en subregistros (tiene mini registros para almacenar los datos antes de pasárselos)

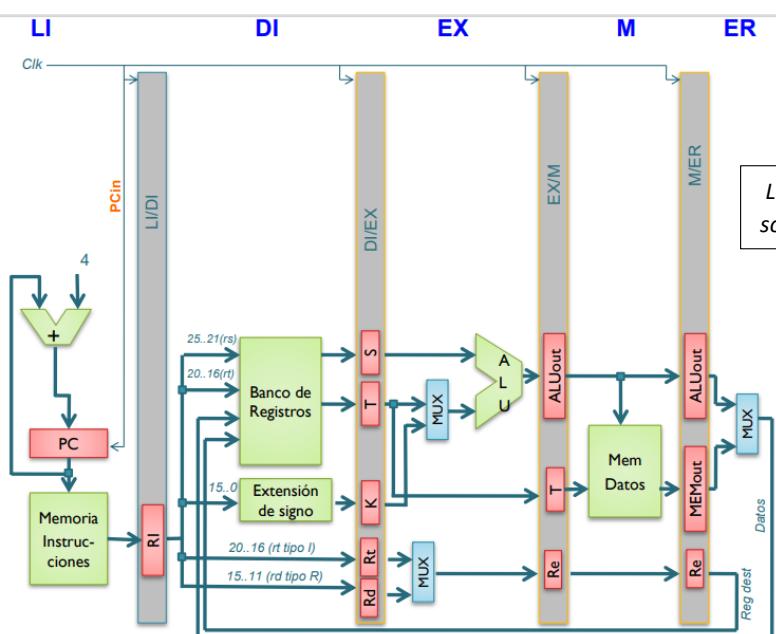
Nomenclatura: Nombre de cada registro de etapa: el de las etapas que separa

Ej: registro LI/DI

Subregistros: Los registros de memoria que se hayan dentro de cada registro de segmentación: reg.subreg:
Ej.: LI/DI.RI

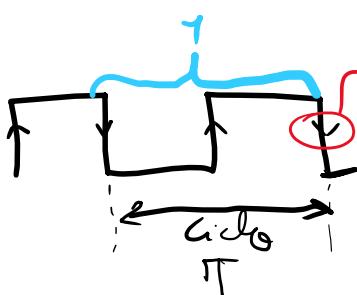
Rango de bits de un subregistro: Se hace con subíndices →

Ej.: LI/DI.RI31..26



Este modelo no se nos pedirán que lo modifiquemos en el examen

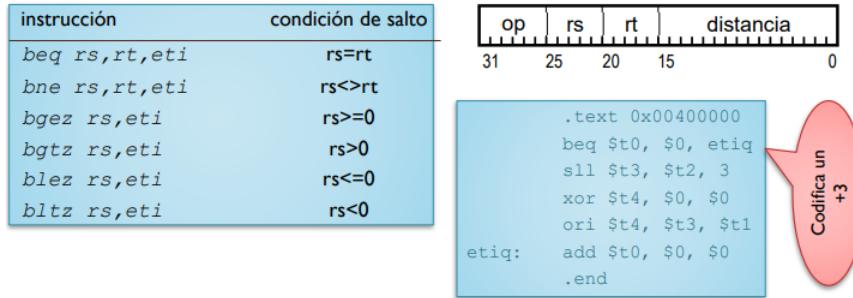
En la timeline de los ciclos de reloj se escribirá en el flanco de bajada, justo al final de ciclo. Cuando empiece el siguiente ciclo procesará lo que tenga que hacer y de igual forma escribirá en los registros de segmentación al final del ciclo. Todos a la vez



Soporte Instrucciones de Salto

En el MIPS R2000 tenemos **seis instrucciones de salto condicional**, todas del formato I. El **Direccionamiento relativo** al PC se hace como **distancia de salto**, se **codifica** (en complemento a 2) el **número de palabras** entre la instrucción siguiente y la **instrucción de salto**

$$\text{Distancia} = \frac{\text{Dirección objetivo} - \text{Dirección siguiente}}{4}$$

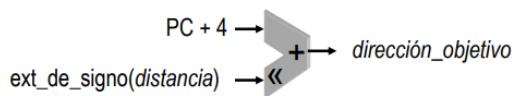


Calculo Dirección de Salto

También Hay que calcular la dirección absoluta de salto:

$$\text{dirección_objetivo} = (\text{PC} + 4) + \text{ext_de_signo}(\text{distancia}) * 4$$

El cálculo se puede hacer en EX con un sumador específico adicional, para eso hay que transmitir el valor $\text{PC} + 4$ desde LI hasta EX y Si la condición se cumple, la etapa M **escribe dirección objetivo en el CP**



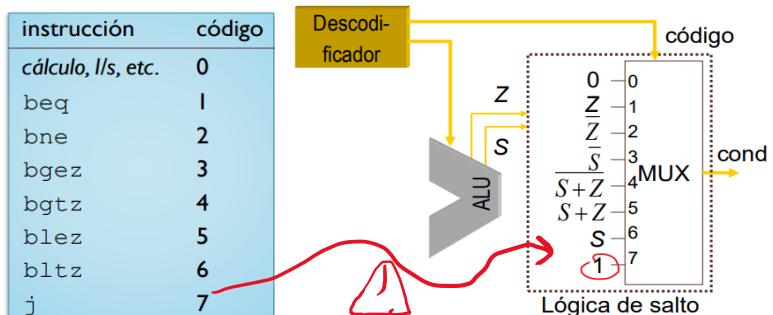
Evaluación de la condición: La ALU ha de tener la operación identidad: $\text{ALUout} = A$ que ha de suministrar dos indicadores: Z (resultado igual a zero) y S (bit de signo del resultado)

instrucción	condición	op. ALU	COND
<code>beq</code>	$a=b$	resta	Z
<code>bne</code>	$a \neq b$	resta	Z
<code>bgez</code>	$a \geq 0$	identidad	S
<code>bgtz</code>	$a > 0$	identidad	$\bar{S} \cdot \bar{Z} = \bar{S} + Z$
<code>blez</code>	$a \leq 0$	identidad	$S + Z$
<code>bltz</code>	$a < 0$	identidad	S

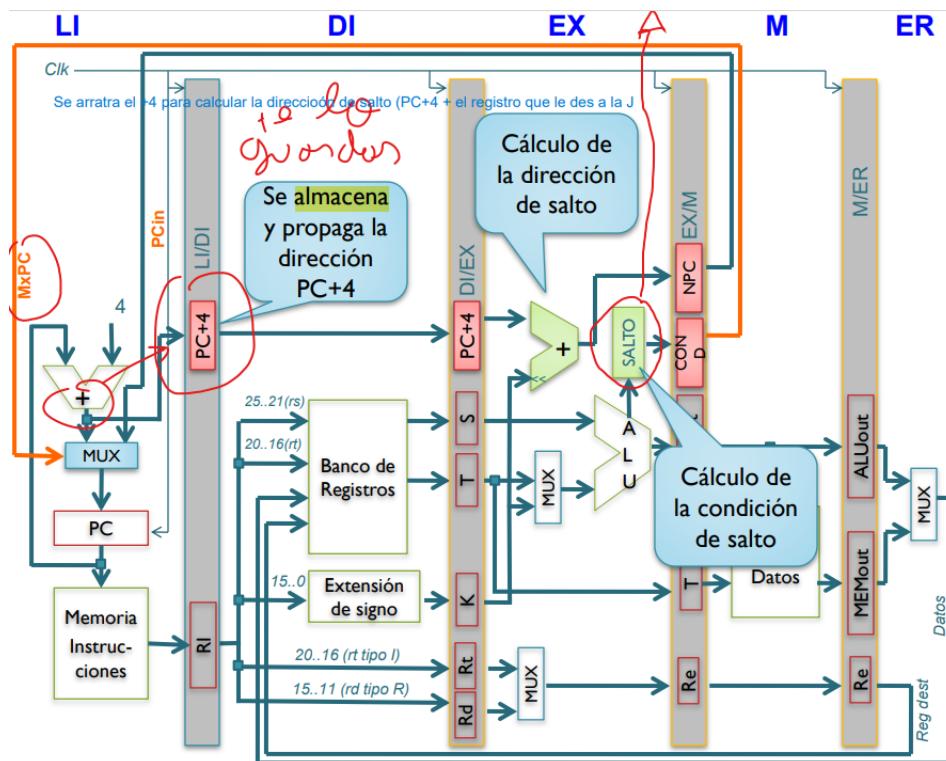
(bits de la ALU)

Control básico de la bifurcación

Un nuevo registro **EX/M.Cond** (un bit): **indicará si hay bifurcación efectiva** Implementando un MUX: El descodificador de instrucción, en la etapa DI, **calcula la posición del MUX** para cada instrucción y en función de esta pone una operación o otra



El Mux de antes



Señales de control

Cada etapa debe de ser autónoma pero hay que tener en cuenta que: Las primeras etapas (LI y DI) procesan instrucciones no descodificadas por lo que sus señales de control serán las mismas durante todos los ciclos → La etapa DI se encargará de calcular las señales de control de las etapas posteriores y las transferirá a las etapas siguientes

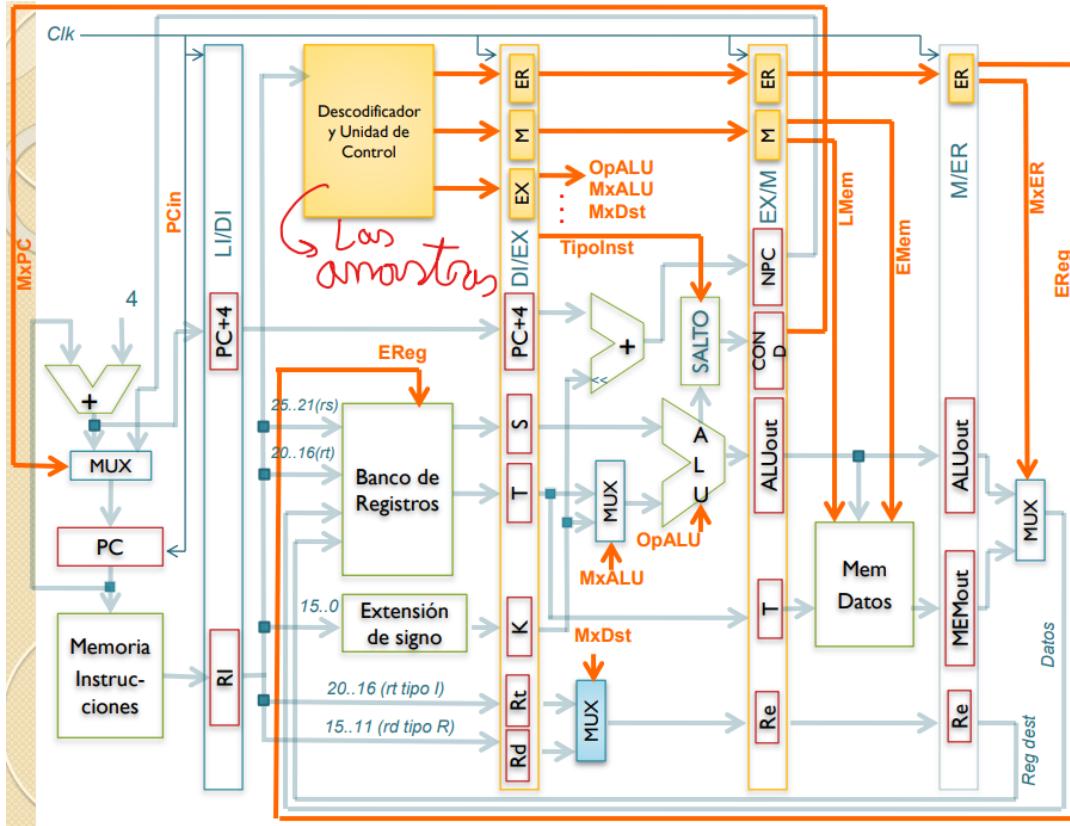
Las señales de control coinciden con las de la ruta no segmentada dependen exclusivamente de la instrucción ejecutada y no de la ruta de datos

Unidad de Control: Señales de control

Instrucción	Form	Código Op.	Func.	Tipoinst	Banco Registros		OpALU	LMem	EMem	Multiplexores Configuración Ruta de Datos			
					LReg	EReg				MxPC	MxALU	MxDst	MxER
add rd, rs, rt	R	000000	100000	0	1	1	010	0	0	COND	0	1	0
sub rd, rs, rt	R	000000	100010	0	1	1	110	0	0	COND	0	1	0
and rd, rs, rt	R	000000	100100	0	1	1	000	0	0	COND	0	1	0
or rd, rs, rt	R	000000	100101	0	1	1	001	0	0	COND	0	1	0
lw rt, desp(rs)	I	100011		0	1	1	010	1	0	COND	1	0	1
sw rt, desp(rs)	I	101011		0	1	0	010	0	1	COND	1	X	X
beq rs, rt, etiq	I	000100		1	1	0	110	0	0	COND	0	X	X
bne rs, rt, etiq	I	000101		2	1	0	110	0	0	COND	0	X	X
bgez rs, etiq	I	000001		3	1	0	111	0	0	COND	0	X	X
bgtz rs, etiq	I	000111		4	1	0	111	0	0	COND	0	X	X
blez rs, etiq	I	000110		5	1	0	111	0	0	COND	0	X	X
bltz rs, etiq	I	000001		6	1	0	111	0	0	COND	0	X	X
j target	J	000010		7	1	0	XXX	0	0	X	X	X	X



RUTA DE DATOS FINAL CON TODAS LAS LINEAS DE CONTROL



Prestaciones del Procesador Segmentado

Ecuación del tiempo de ejecución de un programa en un procesador en régimen estacionario (no tiene en cuenta el tiempo de llenado de las etapas)

$$T = I \cdot CPI \cdot tc$$

T: tiempo de ejecución **total** del programa

I: número de instrucciones que se ejecutan

CPI: número medio de ciclos por instrucción

tc: tiempo de ciclo del reloj del procesador

CPI

Es el índice que se utiliza para cuantificar las prestaciones del procesador y **Representa el número medio de ciclos por instrucción**

$$CPI = \frac{Ciclos - 4}{I}$$

CPI > 1: el pipeline **necesita k-1 ciclos** para llegar a la **última etapa**; ciclos > I

$$\text{Ciclos} = x, \text{Instrucciones} = I, CPI = X/I \rightarrow \frac{Ciclos - (k-1)}{I} = \text{"Número de etapas - 1"}/\text{Instru}$$

5 - 1 = 4 esto es cuando el pipeline (ruta datos) se llena → **empezamos a producir resultados**.

CPI IDEAL = 1: cero ciclos de paro dejando la "cota inferior del CPI" esto no nunca es así?? ns

cómo???



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

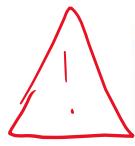
BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

ventajas

PRO



Dí adiós a la publ en los apuntes y en la web



Conflictos

De manera realista las producciones no son ideales, esto sucede porque hay cosas que causan que no se puede usar el 100% de la ruta de datos, lo causan los "conflictos". Son situaciones producidas por la segmentación del procesador, en las que la ejecución de una o más instrucciones no debe avanzar. Hay 3 tipos de conflictos

Estructurales (NO)

De Datos

De Control

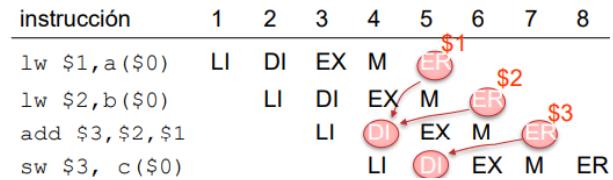
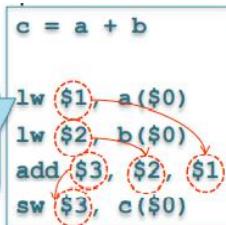
Como soluciones tenemos:

1. **Modificar el Software** añadiendo instrucciones vacías (Intruc NOP)
2. **Crear ciclos de parada** haciendo que le programa se pare

Conflictos de datos

Esto sucede cuando **una instrucción requiere leer los datos de un registro de memoria, pero una instrucción anterior va a reescribirlo, sin embargo, aún no ha terminado**, por lo que si se ejecutase sin más **leería un valor incorrecto**

Existe dependencia entre la instrucción 1 y la 3 por el registro \$1



Cuando \$1 se escribe la add ya ha leído algo incorrecto, como mínimo ha de leer en el next ciclo

AYUDA SOLUCIÓN: Para ganar 1 ciclo se puede hacer que ER escriba a mitad de ciclo (subida), en vez de a final de ciclo (bajada). Así a mitad ciclo ya está hecho y el que lee lo tiene. ¡Si están en la misma etapa esta okey!



Solución Modificar el Software

La idea es que a la hora de compilar las instrucciones si hay dependencia entre ellas se añadan instrucciones vacías.

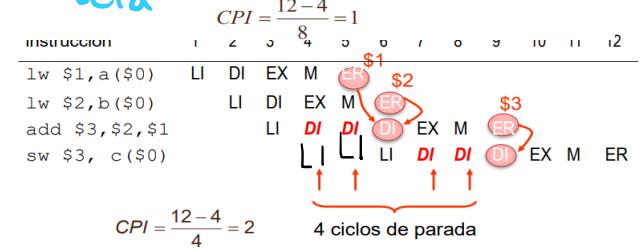
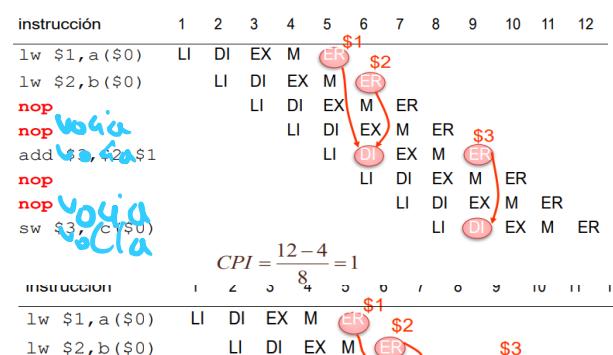
Esta solución arregla las cosas, pero te cuesta **2 ciclos +**, esto se hace desde compilación.

Solución Modificar Ruta de Datos

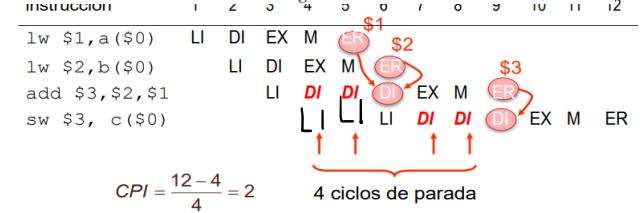
La idea es hacer que cuando se detecte la dependencia, la ruta de datos, haga que las instrucciones se repitan hasta que la deseada termine y se pueda leer el registro

- $T_{nop} = T = I \cdot CPI \cdot T_{ciclo} = 8 \cdot (12 - 4)/8 \cdot tc = 8 \cdot tc$
- $T_{parada} = T = 4 \cdot (12 - 4)/4 \cdot T_{ciclo} = 8 \cdot tc$

Comparando los dos tiempos vemos que nos quedamos igual, solo cambia el "a qué" le das más trabajo, al procesador al compilador...



$$CPI = \frac{12 - 4}{8} = 1$$



$$CPI = \frac{12 - 4}{4} = 2$$

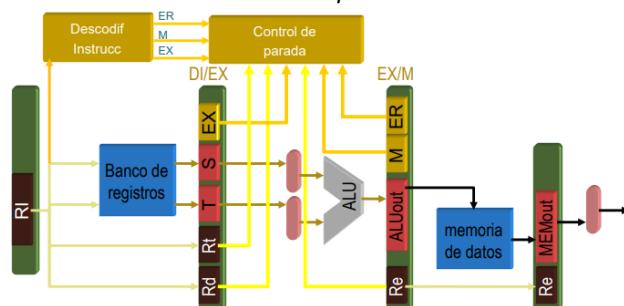
4 ciclos de parada

Curioso con los ~~LW~~ y los SW

Detección y Solución de la dependencia

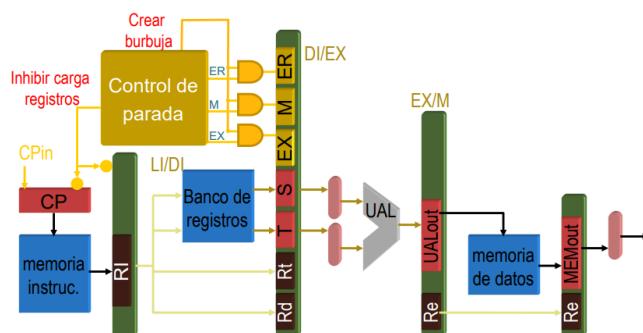
Detector

Comparar donde va a escribir las partes **EX** y **M** con las del **Di** Tanto el Rt como el Rd. En alto nivel sería: **If(LI/DI.RI.RS == DI/EX.Rd) || (LI/DI.RI.RS == Ex/M.Re)** + iguales pero con los "rd"
Comparar 2 registros con una OR exclusiva. Le pasamos al CONTROL DE PARA los registros



Solución

Se hace que **NO se escriba en los registros de segmentación** haciendo así que las **instrucciones se repitan**. Esto crea que las siguientes etapas “estén vacías”, para evitar eso se mandan instrucciones vacías mediante **burbujas**.



Valoración de las dos soluciones vistas

Semejanzas

- El número de ciclos en tiempo de ejecución es el mismo: *En el ejemplo: 12 ciclos*
 - El compilador puede ayudar a bajar el tiempo de ejecución mediante reordenación de código

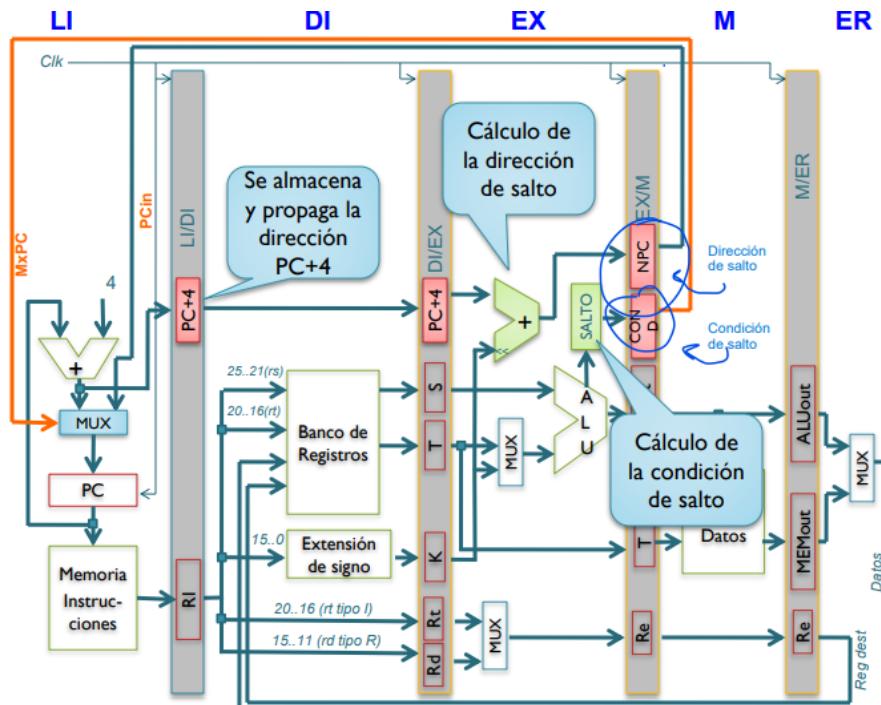
Diferencias

- La inserción de instrucciones inútiles incrementa el factor CPI
 - Los ciclos de parada incrementan el factor CPI
 - La ruta de datos segmentada es compatible binaria con la no segmentada (la ejecución en ambos procesadores produciría los mismos resultados)
 - La complejidad de la lógica de inserción de ciclos de parada podría alargar el retardo de las etapas y habría que bajar la frecuencia del reloj

Conflictos de Control

Las instrucciones de salto Son instrucciones que rompen la secuencia lineal de ejecución de los programas y Pueden ser de 3 Tipos: **Salto incondicional (jump)**, **Salto condicional o bifurcación (branch)** y **Llamada y retorno de subprograma (call/jump&link y retorno)**

También existen 3 **Modos de direccionamiento**: *Absolute, Relativo a PC e Indirecto*



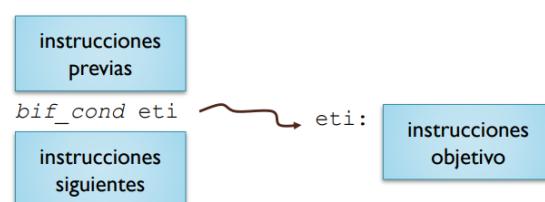
Conceptos

Instrucción objetivo (target): la instrucción destinataria del salto

Bifurcaciones: *saltan si se cumple una condición*

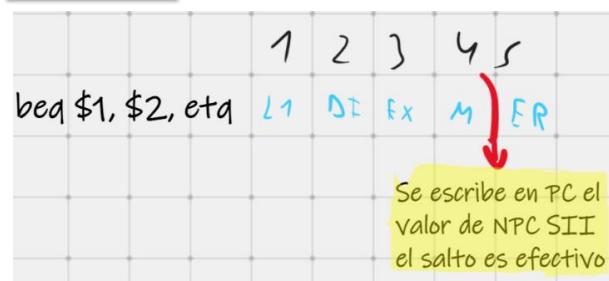
- **Salto Efectivo:** Si bifurcan, diremos que el salto es efectivo (**taken**)
- **Salto NO Efectivo:** En caso contrario, diremos que el salto es no efectivo (**not taken**)

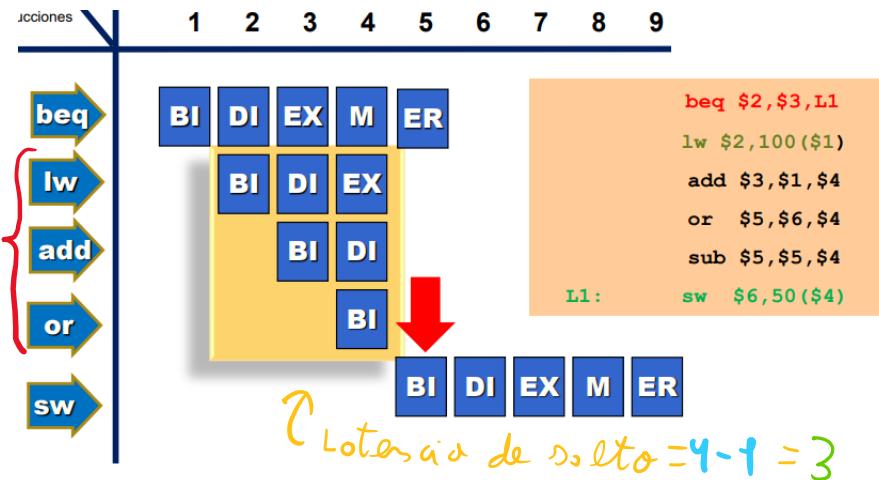
Una instrucción de salto condicional relaciona **tres grupos de instrucciones**:



Saber cuándo saltar BEQ

Las instrucciones BEQ no hacen nada en la segmentación ER, y saben si se va a hacer el salto tras 4 ciclos, “en M”, si el salto es efectivo se cambia el pc sinó no





Latencia del Salto

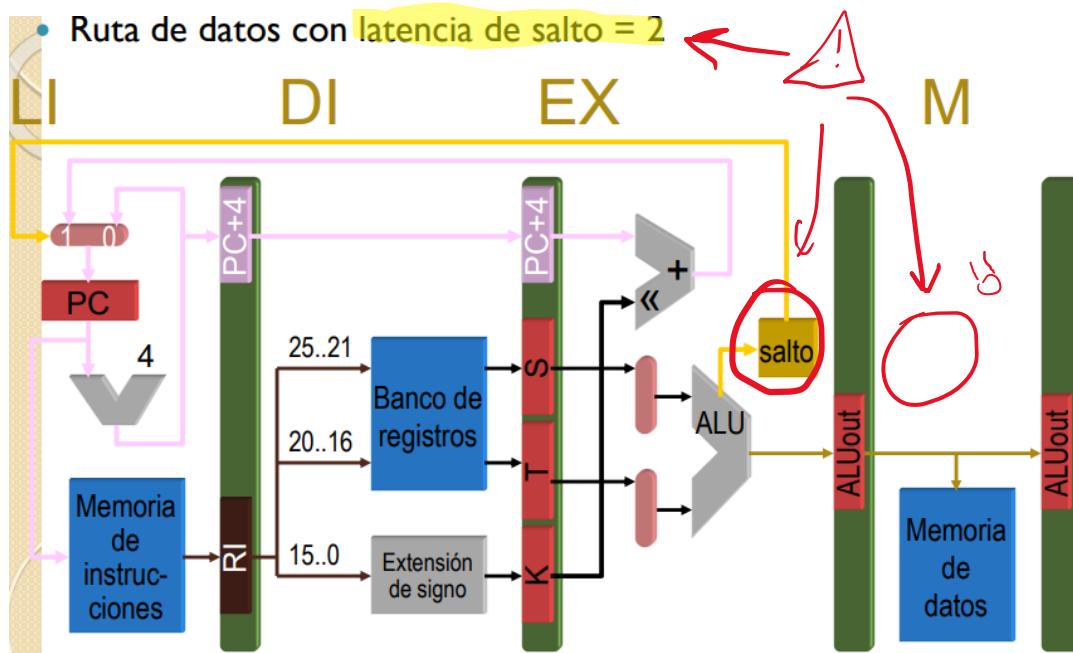
Número de ciclos desperdiciados en caso de que el salto se produzca. Ciclos que perdemos mientras se calcula la condición de salto, como **Número de instrucciones “erróneas”** en el procesador si el salto se produce. Se calcula (desde que se empieza a ejecutar la instrucción)

$$\text{Latencia de salto} = \text{número de etapa en la que el salto es efectivo} - 1$$

Bueno en las diapos pone eso, “menos 1” pero te diría que es “etapa en la que salta” – “etapa en la que empieza” que tiene más sentido xd

Mejorar Latencia

Cuanto más temprana sea la etapa en que se actualiza el CP, menor latencia y por tanto menor penalización. También, para reducir el conflicto, se puede **modificar el diseño de la ruta de datos** y **avanzar el momento de la escritura del PC**.



cómo???



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

ventajas

PRO



Di adiós a la publ en los apuntes y en la web



Acumula tickets para los sorteos



Descarga carpetas completas

estudia sin publi

WUOLAH PRO

Tratamiento de los Conflictos de Control

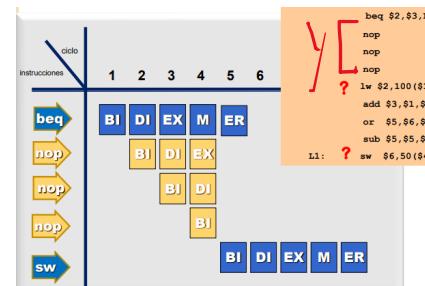
Soluciones conservativas o de urgencia

Hardware: El decodificador **inserta ciclos de parada** al detectar un salto

Software: El compilador inserta **NOP**

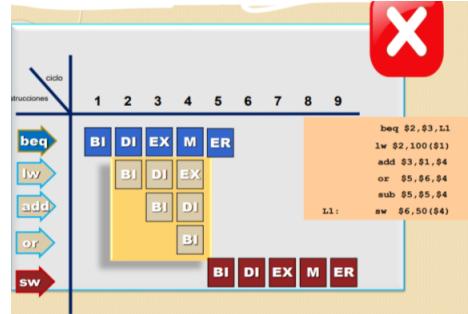
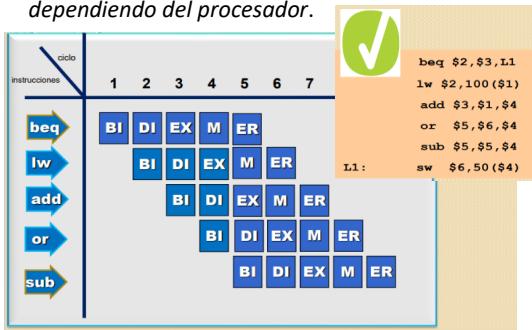
Soluciones avanzadas: *predicción*

Predicción fija: Predict-not taken y predict taken



Predicción: Salto NO efectivo

Se asume que el salto se evaluará a "not taken" por lo que en la ejecución NO saltará y se seguirán ejecutando las instrucciones de debajo. En el que caso se acertar la predicción todo bien y todo correcto: **No se pierden ciclos**. En caso de que se falle, habremos empezado a ejecutar 3 instrucciones que no se acabarán, **por lo que habremos perdido 2 o 3 ciclos dependiendo del procesador**.



RESUMEN

Solución 1

- Poner 3 NOPs: meter instr. vacías

Solución 1

- PNT (Perdict not taken): Mediante estadística tomar decisiones sobre qué hacer: si ejecutar las instrucciones siguientes (y no saltar) o no hacerlo

Solución 2

- Detener lecturas: 3 ciclos parada

EN LOS EJERCICIOS TE TIENEN QUE DECIR

- Si el salto es efectivo o no
- La latencia de salto
- Como se procesan: NOPs, Parada...

A PARTIR DE AQUÍ EL TEMA NO HACE FALTA, SOLO LEER QUE ES TEORÍA RARA :)

Seguidme en wuolah “TurboApuntesPat” hay apuntes de todas las asignaturas, no te arrepentirás juju. Sígueme en twitter @[MaikPatataOtaku](#)
;) Y en Instagram también juju @[potatproductions](#) :p