Segmentación del MIPS

Arquitectura de Computadoras I
2024

Prof. Dr. Martín Vázquez

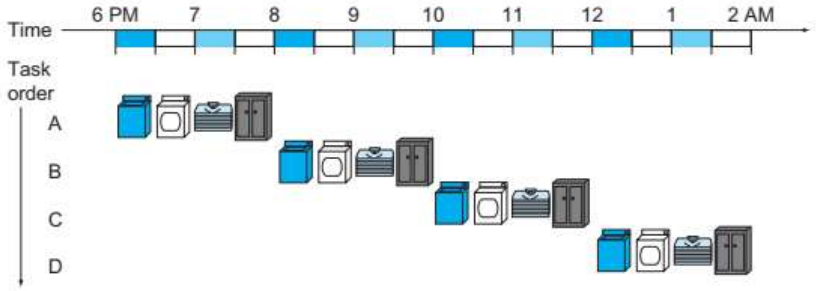
Segmentación
Introducción

Técnica utilizada para optimizar el tiempo de ejecución de procesos que se realizan mediante la repetición de una secuencia básica de pasos

- Separar el proceso en etapas y ejecutar cada una en un recurso independiente
- Mejorar la productividad, cantidad de procesos terminados por unidad de tiempo
- Cuando una etapa termina, el recurso liberado puede ejecutar la misma etapa del siguiente proceso

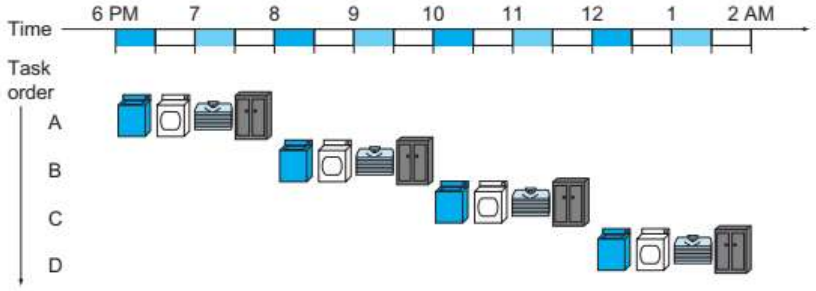
Segmentación
Introducción

- **Funcionamiento secuencial**
  - usar lavarropa
  - usar secarropa
  - planchar ropa
  - guardar ropa



Segmentación
Introducción

- **Funcionamiento secuencial**



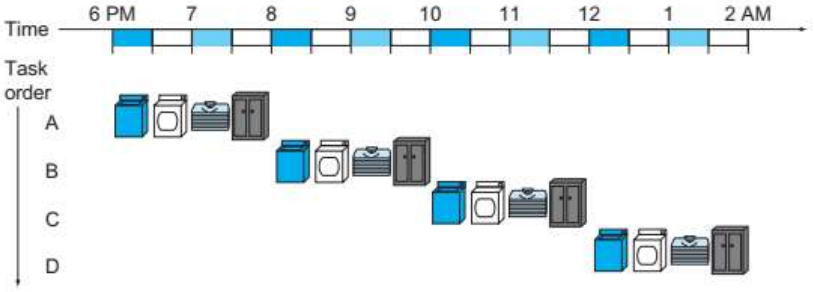
Cada proceso completo de lavado tarda 2 hs (cada tarea de ½ hora)

Tiempo de realizar *M* procesos secuenciales

$T_P(M) = T_P(1).M = 2.M$ hs

Segmentación
Introducción

- **Funcionamiento secuencial**

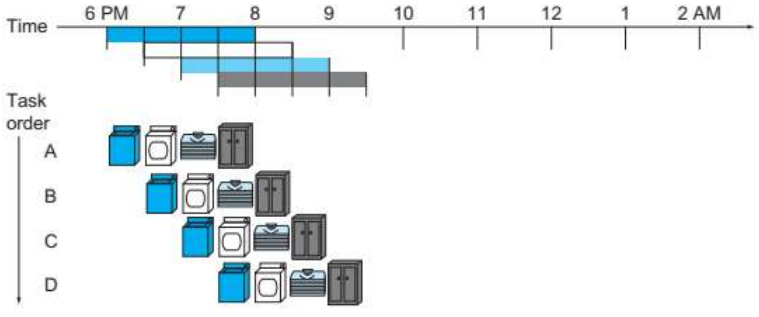


$T_P(M) = 2.M$ hs

completar 4 procesos de lavado tarda 8 hs

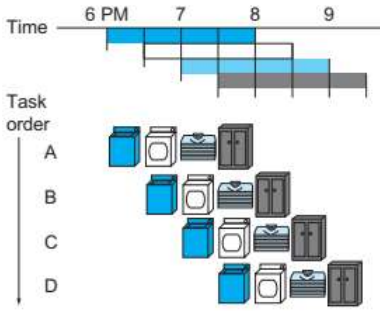
Segmentación
Introducción

- **Funcionamiento segmentado**



Segmentación
Introducción

- **Funcionamiento segmentado**

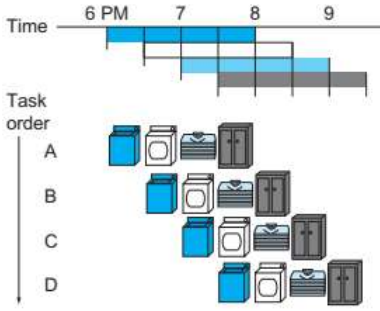


Para *M* procesos

el primero se completa en 2 hs
los siguientes en ½ hora

Segmentación
Introducción

- **Funcionamiento segmentado**



Para *M* procesos

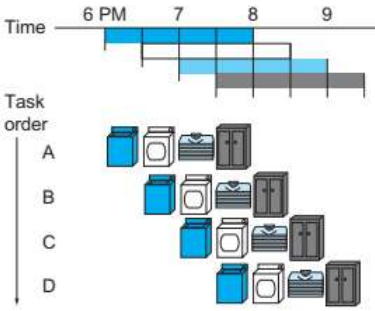
el primero se completa en 2 hs
los siguientes en ½ hora

Tiempo de realizar *M* procesos segmentados

$$T_S(M) = T_S(1) + (M\text{-}1).T_{ETAPA} \text{ hs}$$

Segmentación
Introducción

- **Funcionamiento segmentado**



completar 4 procesos de lavado tarda 3 ½ hs

$T_S(M) = T_S(1) + (M\text{-}1).T_{ETAPA}$ hs

Segmentación
Introducción

- Para 4 procesos de lavado

Secuencial **8 hs** → **Segmentado** **3½ hs**

- Para 10 procesos de lavado

Secuencial **20 hs** → **Segmentado** **6½ hs**

Segmentación
Introducción

- Para 4 procesos de lavado

Secuencial 8 hs → Segmentado 3½ hs ← Aceleración = 2.28

- Para 10 procesos de lavado

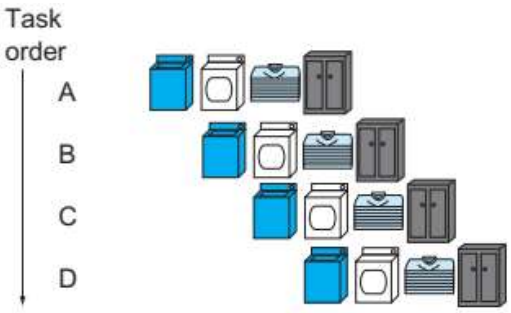
Secuencial 20 hs → Segmentado 6½ hs ← Aceleración = 3.07

La aceleración aumenta en la medida que aumenta la cantidad de procesos completos de lavado

Segmentación
Introducción

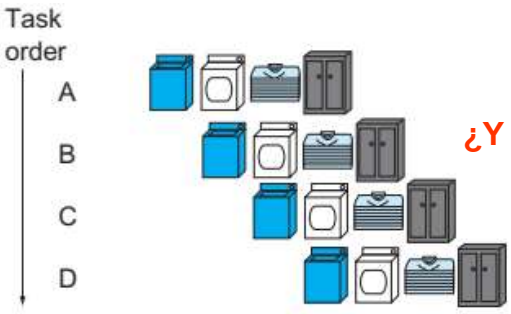
- Para un solo proceso de lavado **NO** hay aceleración
  - $T_P(1) = T_S(1)$
- Para *M* procesos de lavado con *M* tendiendo a infinito
  - $T_s(M) \rightarrow M.T_{ETAPA}$
  - Aceleración máxima → $T_P(1)/T_{ETAPA}$
- Para el ejemplo del lavado de ropa
  - Aceleración máxima → 2 / ½ = 4
- **La aceleración máxima es igual a la cantidad de etapas del proceso segmentado**

Segmentación
Introducción



- Las etapas tardan lo mismo: ½ hs.
- Cada proceso de lavado completo lleva 2 hs
- Primer proceso completo sale después de 2 hs
- Una vez en régimen, el resto de los procesos salen cada ½ hora

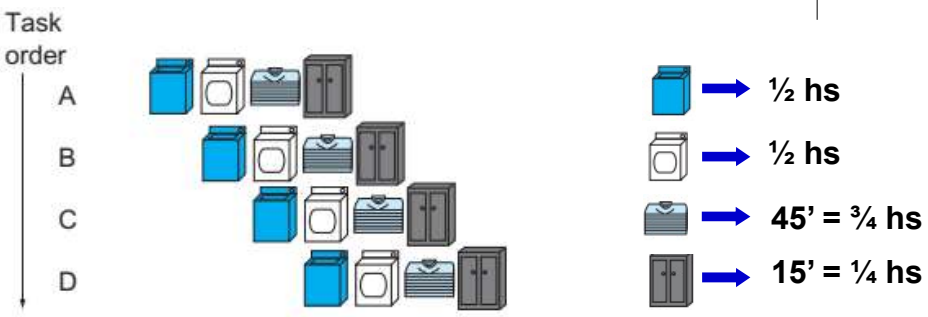
Segmentación
Introducción



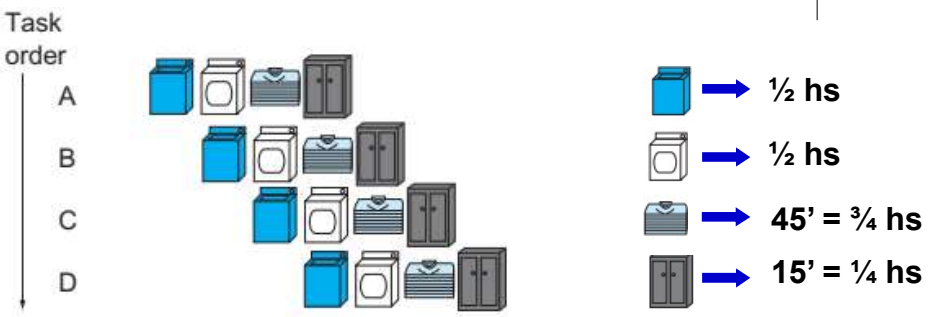
¿Y si las etapas no tardan el mismo tiempo?

- Las etapas tardan lo mismo: ½ hs.
- Cada proceso de lavado completo lleva 2 hs
- Primer proceso completo sale despúes de 2 hs
- Una vez en régimen, el resto de los procesos salen cada ½ hora

Segmentación
Introducción



Segmentación
Introducción



- La etapa más lenta es la que determina la segmentación
- Una vez en régimen, el resto de los procesos salen cada 45 minutos
- Aceleración máxima es 2.66 (antes era de 4)
- **La aceleración ya no es igual a la cantidad de etapas de segmentación**

Segmentación
Introducción

- La **latencia** es el tiempo que tarda en salir un proceso completo
- En ejemplo de filmina anterior
  - **Latencia** = 3x45+15 minutos = 150 minutos ($\mathbf{T_S(1)}$)
  - La latencia original del proceso sin segmentar es de 120 minutos ($\mathbf{T_P(1)}$)

Segmentación
Introducción

- La **latencia** es el tiempo que tarda en salir un proceso completo
- En ejemplo de filmina anterior
  - **Latencia** = 3x45+15 minutos = 150 minutos ($\mathbf{T_S(1)}$)
  - La latencia original del proceso sin segmentar es de 120 minutos ($\mathbf{T_P(1)}$)

- Analice un proceso segmentado en 5 etapas con:

  E1 = 30', *E2* = 30', *E3* = 45', *E4* = 15' y *E5* = 30'

¿Cuál es la latencia?

Segmentación
Introducción

- La **latencia** es el tiempo que tarda en salir un proceso completo
- En ejemplo de filmina anterior
  - **Latencia** = 3x45+15 minutos = 150 minutos (**$T_S(1)$**)
  - La latencia original del proceso sin segmentar es de 120 minutos (**$T_P(1)$**)
- Analice un proceso segmentado en 5 etapas con:

  E1 = 30', *E2* = 30', *E3* = 45', *E4* = 15' y *E5* = 30'

¿Cuál es la latencia?

Latencia = 3x45+2x30 min. = 195 min.

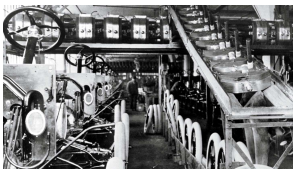
Segmentación
Introducción

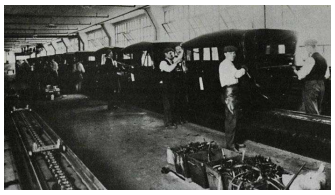
- La **latencia** de un proceso segmentado es siempre mayor o igual a la **latencia** de un proceso sin segmentar.
- ***Throughput*** es la cantidad de procesos completos que se realizan por unidad de tiempo
- Considerando *N* la cantidad de etapas y T_{ETAPA} el tiempo de etapa más lenta
  - **Latencia** = $N.T_{ETAPA}$ (llevado a etapas que duren lo mismo)
  - ***Throughput*** = $1/T_{ETAPA}$

Segmentación
Introducción

- **Henry Ford** (1863-1947)
  - aplica la teoría del sistema de Taylor para la producción del Ford T

Fabricación del Modelo T (1908)





Segmentación
Introducción

- **Henry Ford** (1863-1947)

  - Popularizó la **producción en cadena** o **producción en serie**
  - Todos los operarios realizan simultáneamente una parte del trabajo sobre diferentes automóviles
  - El flujo de piezas es continuo
  - El número de operaciones (etapas) es reducido
  - Las tareas se planean de manera que puedan tardar aproximadamente el mismo tiempo
  - La velocidad máxima de producción (*throughput*) queda limitada por la velocidad de la etapa más lenta

Segmentación
Introducción

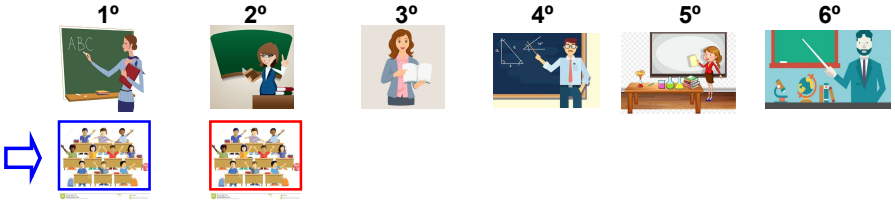
- El actual sistema educativo en la Argentina posee una producción en serie de egresados
- Es un “sistema industrializado”
- Los procesos completos son las promociones de alumnos formados y egresados
- Los docentes realizan una parte del trabajo sobre una promoción de alumnos
- En la escuela primaria (estructura 6 años), existen 6 etapas

Segmentación
Introducción

1º 2º 3º 4º 5º 6º

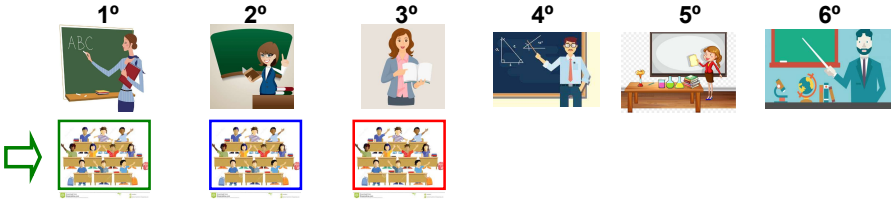
Promoción 2014

Segmentación
Introducción



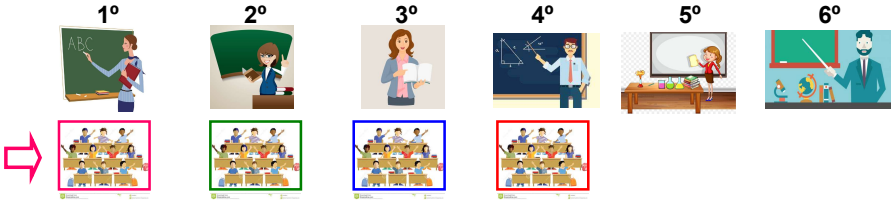
Promoción 2014
Promoción 2015

Segmentación
Introducción



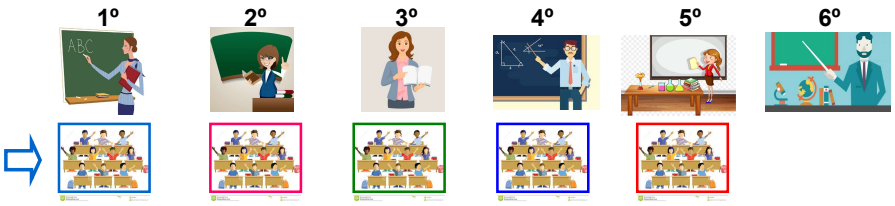
Promoción 2014
Promoción 2015
Promoción 2016

Segmentación
Introducción



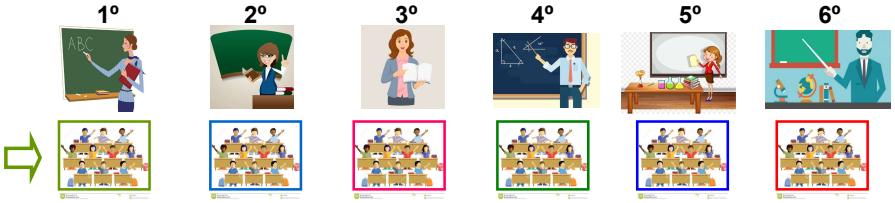
Promoción 2014
Promoción 2015
Promoción 2016
Promoción 2017

Segmentación
Introducción



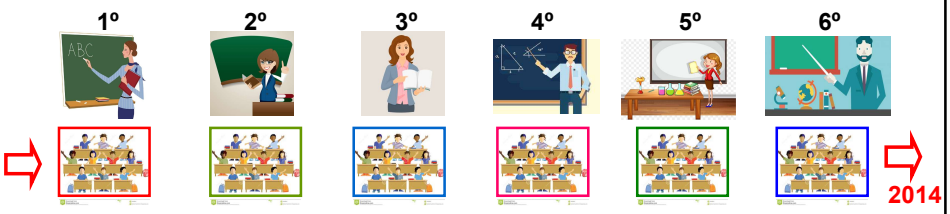
Promoción 2014
Promoción 2015
Promoción 2016
Promoción 2017
Promoción 2018

Segmentación
Introducción



Promoción 2014
Promoción 2015
Promoción 2016
Promoción 2017
Promoción 2018
Promoción 2019

Segmentación
Introducción



Promoción 2015
Promoción 2016
Promoción 2017
Promoción 2018
Promoción 2019
Promoción 2020

Segmentación
Introducción

- Todas las etapas tardan lo mismo (un año) y trabajan en paralelo
- Una vez que el sistema se encuentra en régimen con todas las aulas llenas. Todos los años se obtiene una promoción de alumnos egresados y formados
- La **latencia** de un alumno o promoción de alumnos en egresar es de 6 años
- El ***troughput*** es una promoción completa de egresados por año

Pipeline

- En diseño de circuitos digitales, la segmentación de procesos es conocida como la técnica del ***pipeline***
- Una “tubería” de procesos conectados en cadena

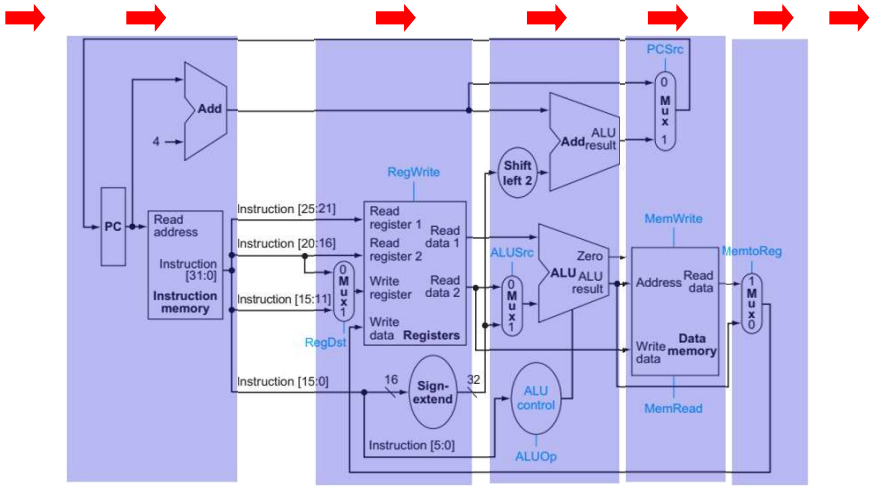
El flujo de entrada es el mismo al flujo de la salida

No puede entrar nada a la tubería sin salir algo del otro lado

La velocidad del flujo de entrada no depende de la longitud de la tubería

Microprocesador MIPS
Pipeline

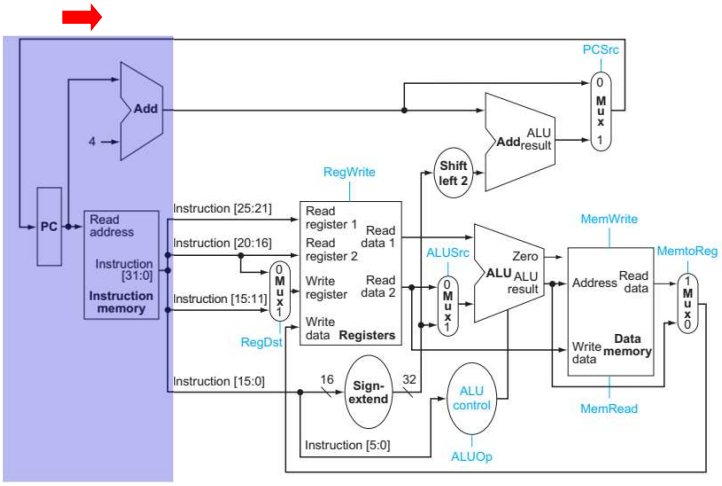
- Observar el procesador como una “tubería” de instrucciones



Microprocesador MIPS
Pipeline

- **lw** $1, 134($2)

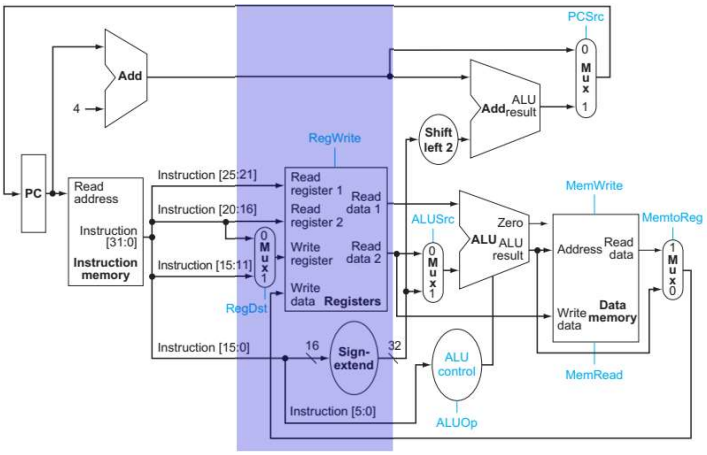
Obtener instrucción lw
Acceso a memoria de instrucciones



Microprocesador MIPS
Pipeline

- **lw** $1, 134($2)

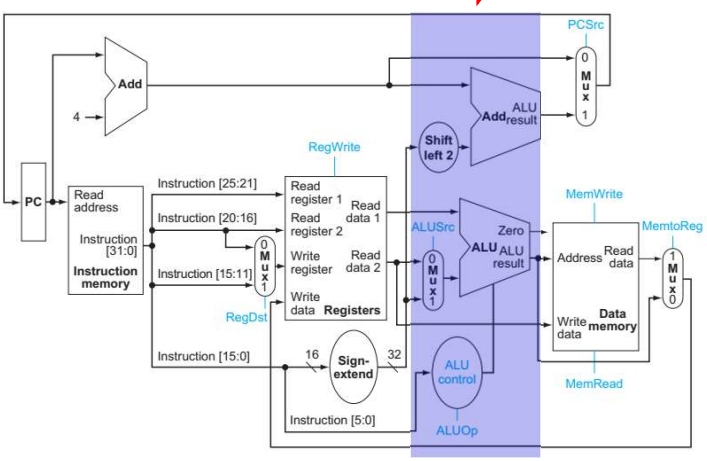
Decodificar instrucción lw
Lectura de operando $2



Microprocesador MIPS
Pipeline

- **lw** $1, 134($2)

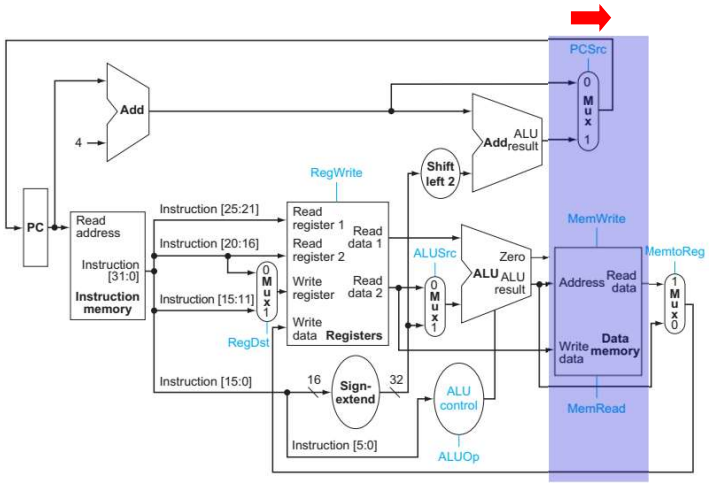
Calcula dirección de acceso a memoria



Microprocesador MIPS
Pipeline

- **lw** $1, 134($2)

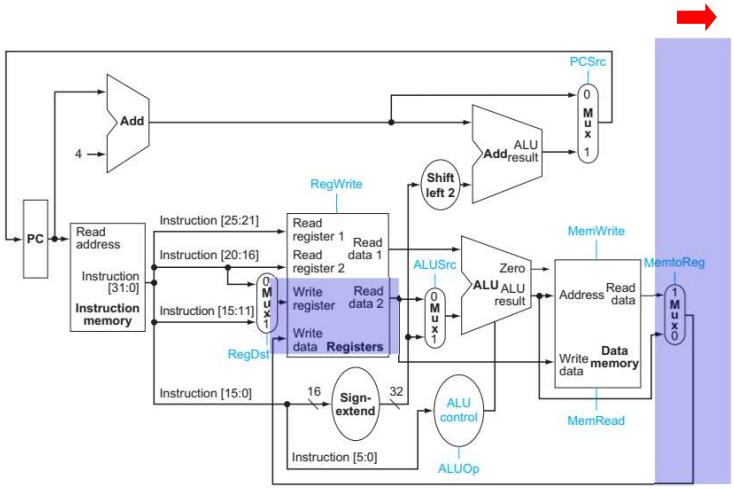
Obtener dato
Acceso a memoria de datos



Microprocesador MIPS
Pipeline

- **lw** $1, $2(134)

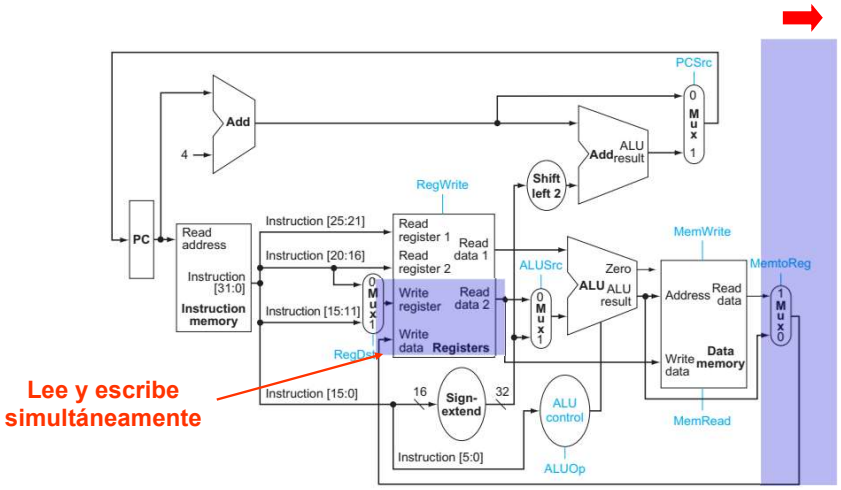
Escritura del resultado en $1



Microprocesador MIPS
Pipeline

- **lw** $1, 134($2)

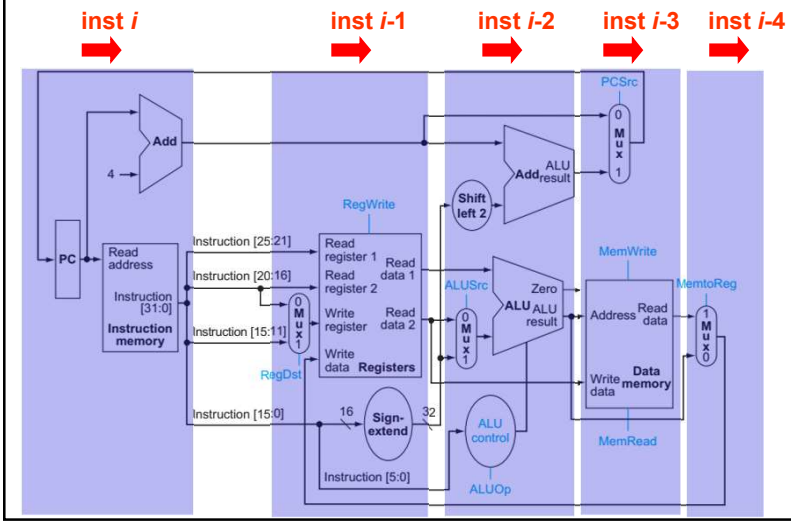
Escritura del resultado en $1



Lee y escribe simultáneamente

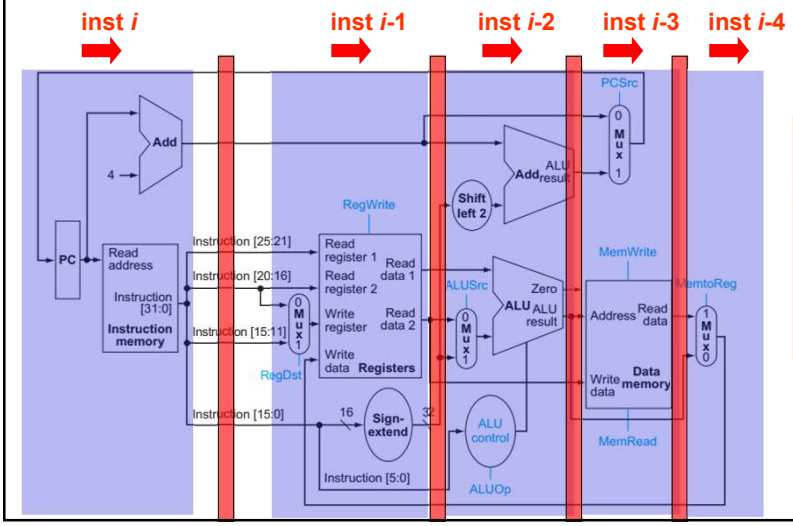
Microprocesador MIPS
Pipeline

- "Tubería" de instrucciones



Microprocesador MIPS
Pipeline

- "Tubería" de instrucciones



Difícil de implementar

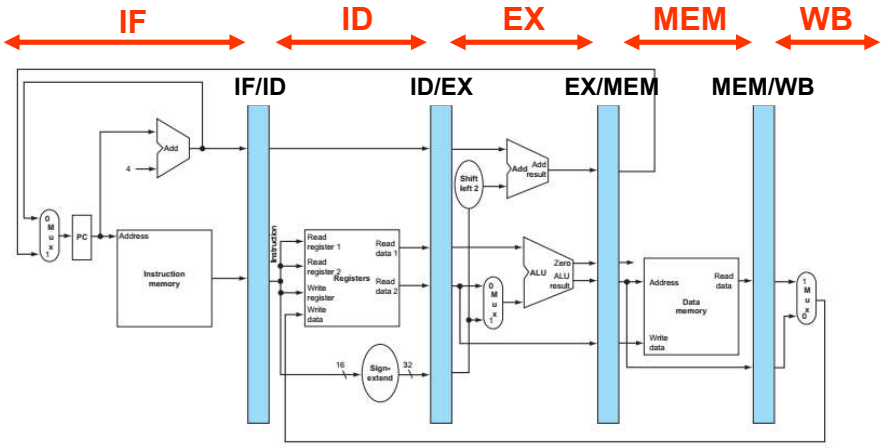
Necesita registros de *pipelining*

Microprocesador MIPS
Pipeline

- División del *datapath* del MIPS en 5 etapas de *pipelining*
  - **IF** (*Instruction Fetch*): obtiene la instrucción desde la memoria de programa
  - **ID** (*Instruction Decode*): lectura de registros mientras decodifica la instrucción
  - **EX** (*Execute*): ejecuta los operandos o calcula la dirección efectiva de memoria
  - **MEM** (*Memory*): acceso a memoria o escrbir en PC dirección de salto
  - **WB** (*Write Back*): escribe el resultado en un registro

Microprocesador MIPS
Pipeline

- Diseño con 5 etapas



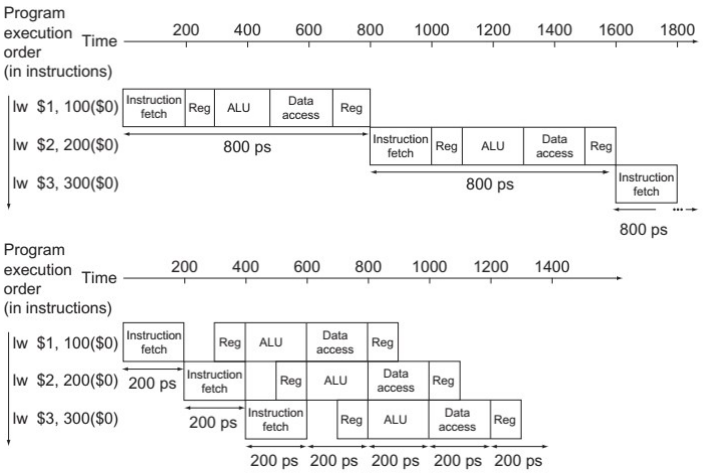
Microprocesador MIPS
Pipeline

- Ciclo único vs. Segmentado. Ejemplo

Instruction class	Instruction fetch	Register read	ALU operation	Data access	Register write	Total time
Load word (lw)	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
Store word (sw)	200 ps	100 ps	200 ps	200 ps		700 ps
R-format (add, sub, AND, OR, slt)	200 ps	100 ps	200 ps		100 ps	600 ps
Branch (beq)	200 ps	100 ps	200 ps			500 ps

Microprocesador MIPS
Pipeline

- Ciclo único vs. Segmentado. Ejemplo

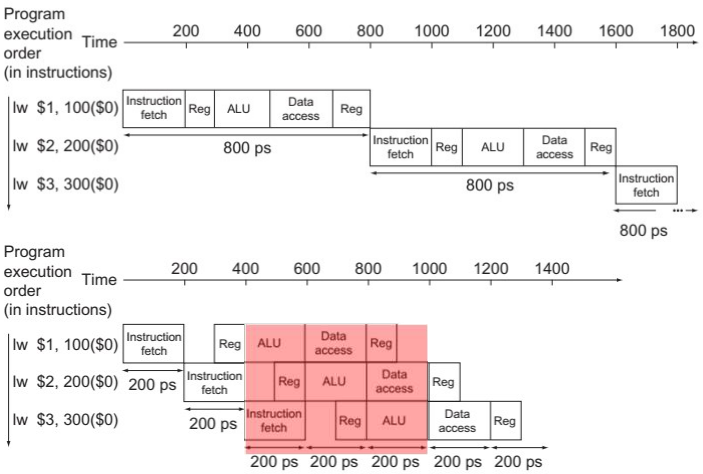


Microprocesador MIPS
Pipeline

- Ciclo único vs. Segmentado. Ejemplo

Se ejecutan tres instrucciones en paralelo

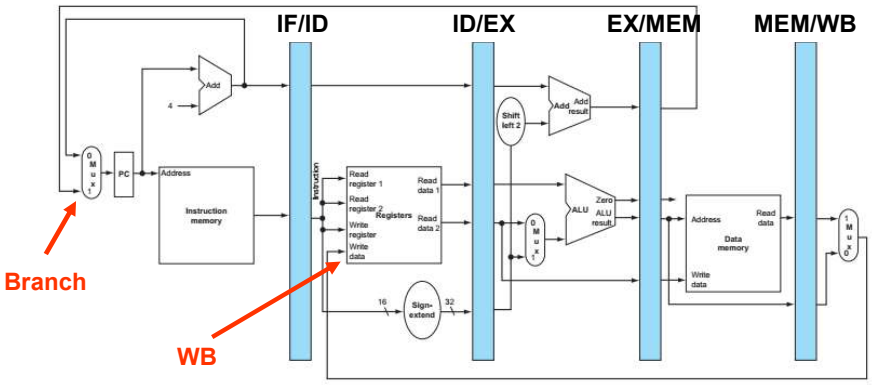
Las unidades funcionales ejecutan instrucciones diferentes



Microprocesador MIPS
Pipeline

- Algunas cosas a tener en cuenta

datos que van de derecha a izquierda

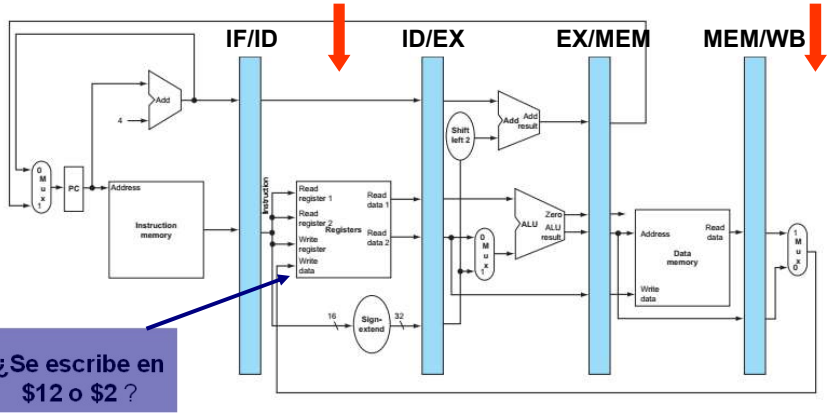


Microprocesador MIPS
Pipeline

- Ejemplo

add $12, $5, $3

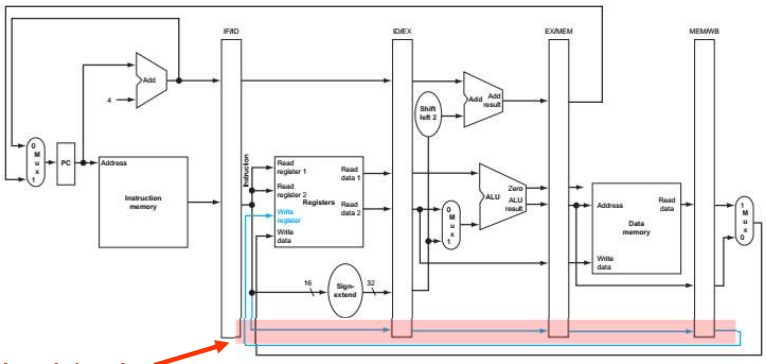
lw $2, 325($5)



¿Se escribe en $12 o $2 ?

Microprocesador MIPS
Pipeline

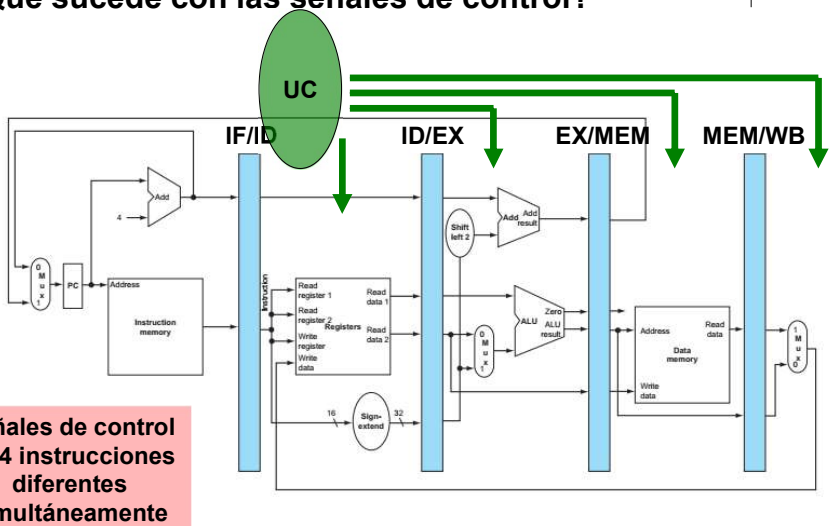
- Solución para determinar el registro de escritura



El número del registro de escritura se pasa por los registros de *pipe*

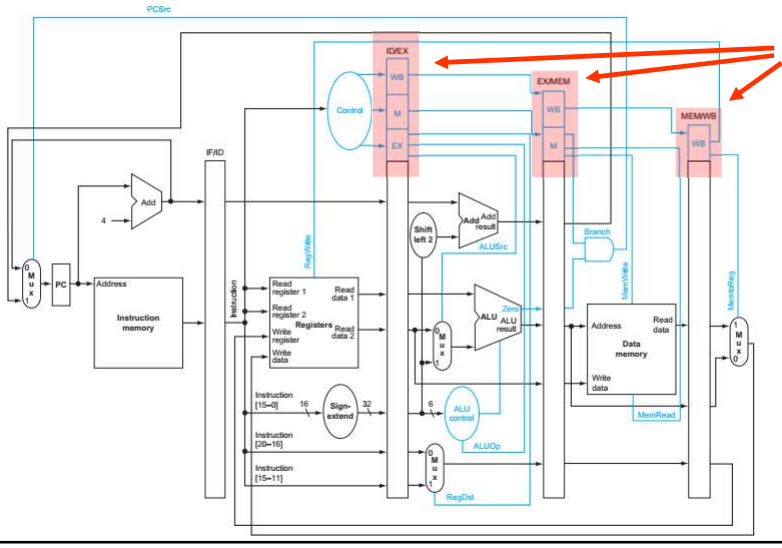
Microprocesador MIPS
Pipeline

- ¿Qué sucede con las señales de control?



señales de control de 4 instrucciones diferentes simultáneamente

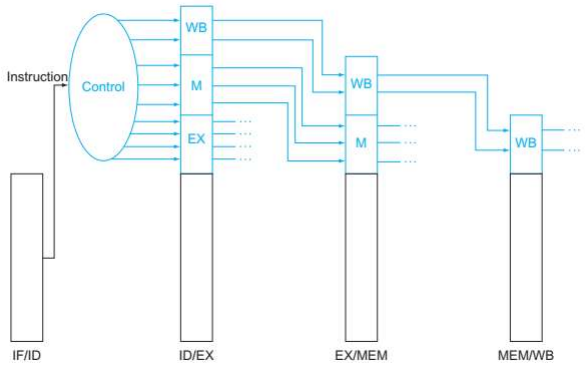
Microprocesador MIPS
Pipeline



Retraso de señales de control

Microprocesador MIPS
Pipeline

- Las señales de control se determinan en la etapa de decodificación (ID)
- Los valores de las señales de control son los mismos que los calculados en el control del microprocesador uniciclo



Microprocesador MIPS
Pipeline - Riesgos

- En el microprocesador escalar segmentado idealmente el CPI es igual a uno
- Existen situaciones en el *pipelining* en las que las instrucciones no pueden ejecutarse en el siguiente ciclo de reloj
- En esos casos el rendimiento del procesador disminuye, el CPI>1
- Existen tres tipos de **RIESGOS**
  - Estructurales
  - Dependencia de datos
  - Control

Microprocesador MIPS
Pipeline - Riesgos

- **Riesgos**
  - Estructurales:
    - se intenta utilizar el mismo recurso hardware por dos instrucciones diferentes al mismo tiempo
    - el hardware impide cierta combinación de operaciones
  - Dependencia de datos:
    - se intenta usar el dato antes que esté disponible
    - el operando de una instrucción depende del resultado de una operación precedentes que aún no se ha obtenido
  - Control
    - se intenta tomar una decisión antes de evaluarse la condición
    - si se salta, las instrucciones posteriores no deben terminar su ejecución

Microprocesador MIPS
Pipeline - Riesgos

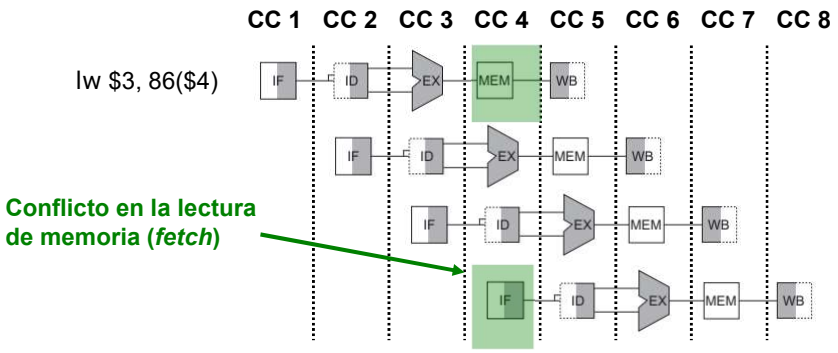
- La forma más sencilla de solucionar un riesgo es detener el *pipeline* hasta que el riesgo desaparezca
  - las instrucciones que preceden a la causante del riesgo pueden continuar
  - la instrucción que causa el riesgo y las siguientes no continúan su ejecución hasta que el riesgo no exista
- El control del *pipeline* debe
  - detectar las causas del riesgo
  - decidir acciones que resuelvan el riesgo

Microprocesador MIPS
Pipeline - Riesgos

- Riesgo Estructural
  - Sucede con accesos simultáneos a memorias, registros o unidades funcionales
  - Se soluciona
    - agregando esperas
    - duplicar recursos
  - Ejemplo: con utilización de arquitectura Von Neumann
    - conflicto estructural entre instrucción lw y el *fetch* de nueva instrucción (tres instrucciones después)
    - soluciona utilizando memoria de programa y datos separadas (Harvard)

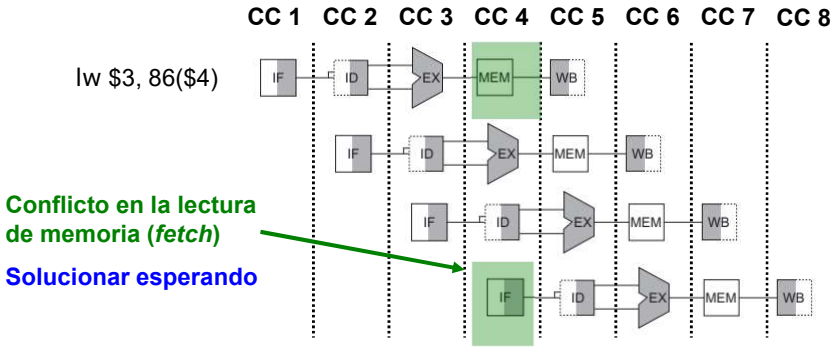
Microprocesador MIPS
Pipeline - Riesgos

- Riesgo Estructural.
  - Arquitectura Von Neumann



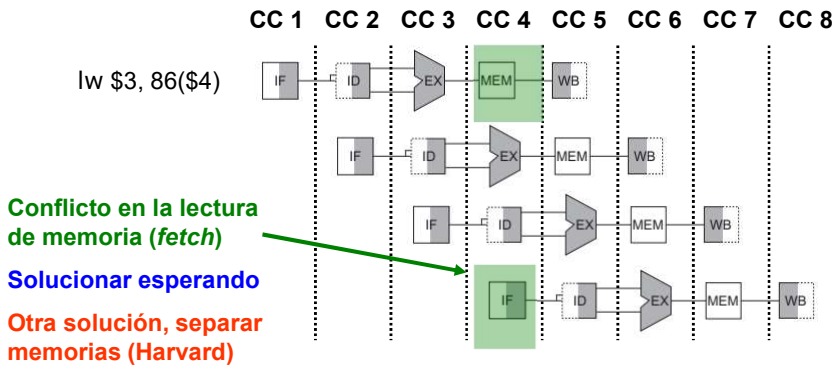
Microprocesador MIPS
Pipeline - Riesgos

- Riesgo Estructural.
  - Arquitectura Von Neumann



Microprocesador MIPS
Pipeline - Riesgos

- Riesgo Estructural.
  - Arquitectura Von Neumann



Microprocesador MIPS
Pipeline - Riesgos

- Entre dos instrucciones pueden existir tres tipos de dependencias
  - **RAW** (*Read After Write*), **WAW** (*Write After Write*), **WAR** (*Write After Read*)

Dependencia RAW	Dependencia WAW	Dependencia WAR
add **$2**, $5, $6	and **$12**, $5, $6	or $9, **$15**, $16
sub $5, $1, **$2**	div **$12**, $1, $2	addi **$15**, $1, 67
lw $3, 35(**$2**)		

Microprocesador MIPS
Pipeline - Riesgos

- Riesgos por dependencia de datos
  - **RAW** (*Read After Write*): Una instrucción posterior intenta leer una fuente antes que una instrucción anterior la haya determinado
  - **WAW** (*Write After Write*): Una instrucción posterior intenta modificar el destino antes que la instrucción anterior lo haya hecho (se modifica el orden de la escritura)
  - **WAR** (*Write After Read*): Una instrucción posterior intenta modificar su destino antes que la instrucción anterior lo haya leído como fuente

Microprocesador MIPS
Pipeline - Riesgos

- Riesgos por dependencia de datos
  - En procesadores escalares segmentados (MIPS), los conflictos se producen con dependencias de datos RAW

Dependencia RAW	Dependencia WAW	Dependencia WAR
add **$2**, $5, $6	and **$12**, $5, $6	or $9, **$15**, $16
sub $5, $1, **$2**	div **$12**, $1, $2	addi **$15**, $1, 67
lw $3, 35(**$2**)		

Microprocesador MIPS
Pipeline - Riesgos

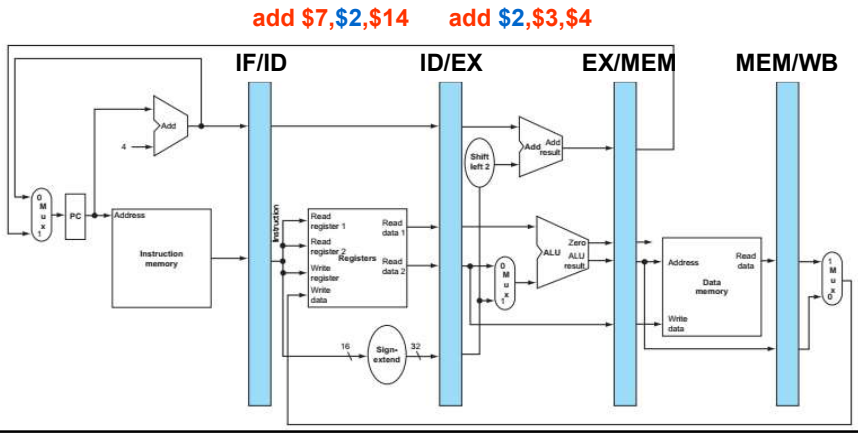
- Riesgos por dependencia de datos
  - En procesadores escalares segmentados (MIPS), los conflictos se producen con dependencias de datos RAW

Dependencia RAW	Dependencia WAW	Dependencia WAR
add **$2**, $5, $6	and **$12**, $5, $6	or $9, **$15**, $16
sub $5, $1, **$2**	div **$12**, $1, $2	addi **$15**, $1, 67
lw $3, 35(**$2**)		

¿Cuál sería el alcance de conflicto por dependencia de datos RAW que ocasiona una escritura?

Microprocesador MIPS
Pipeline - Riesgos

- Riesgos por dependencia de datos

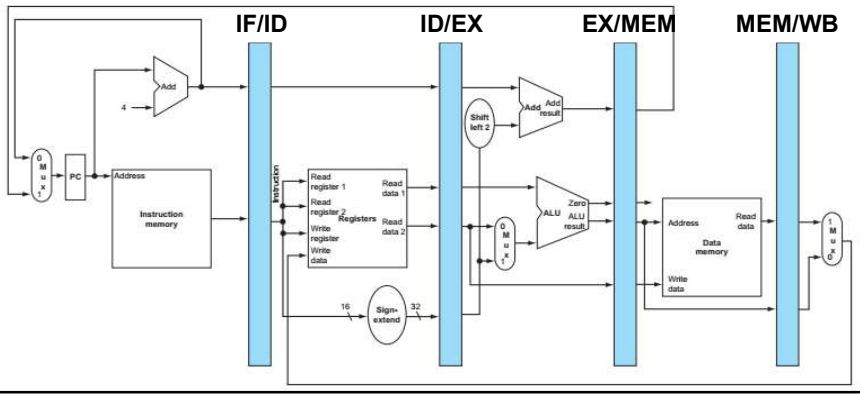


Microprocesador MIPS
Pipeline - Riesgos

- Riesgos por dependencia de datos

Conflicto RAW al leer $2

add $7,$2,$14 add $2,$3,$4

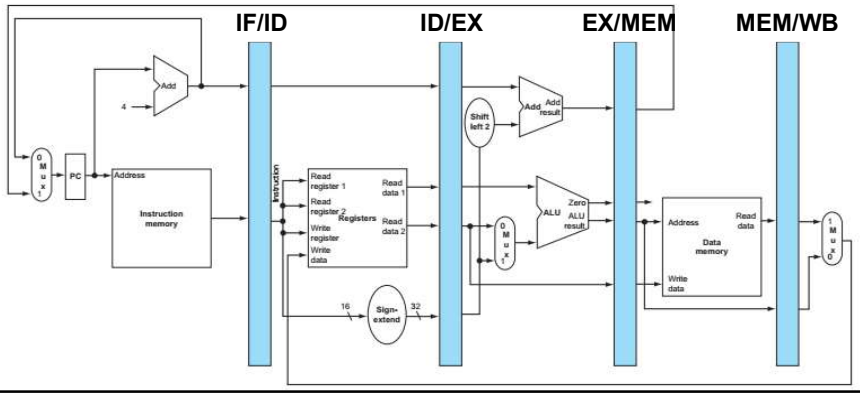


Microprocesador MIPS
Pipeline - Riesgos

- Riesgos por dependencia de datos

Conflicto RAW al leer $2

or $6,$9,$2 add $2,$3,$4

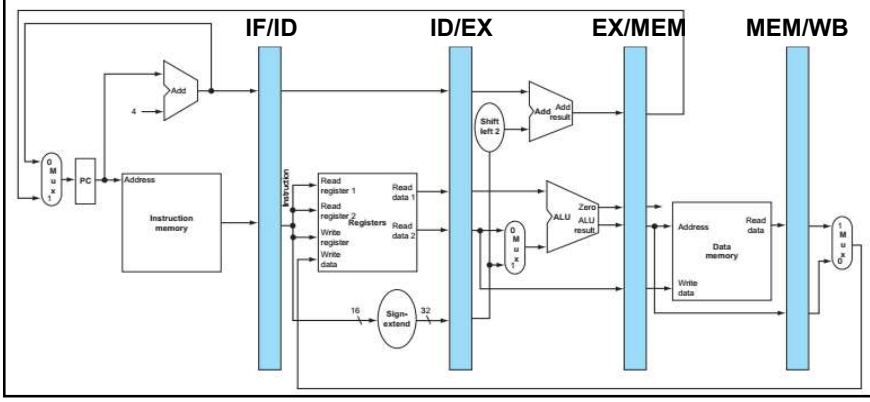


Microprocesador MIPS
Pipeline - Riesgos

- Riesgos por dependencia de datos

¿Existe conflicto?

sub $8,$5,$2 add $2,$3,$4

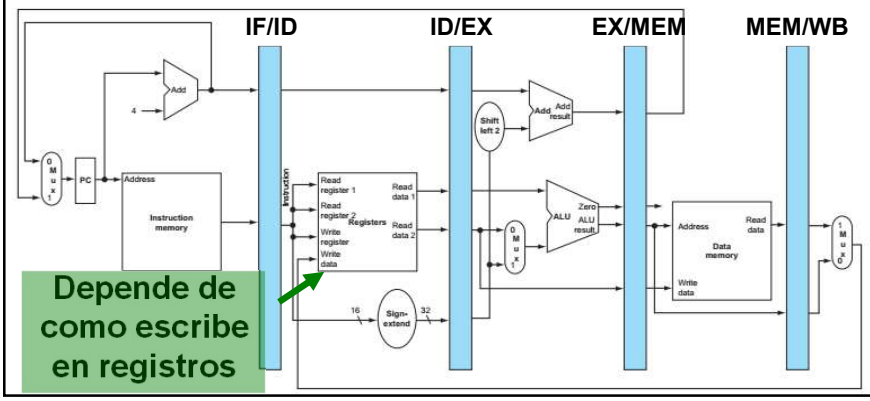


Microprocesador MIPS
Pipeline - Riesgos

- Riesgos por dependencia de datos

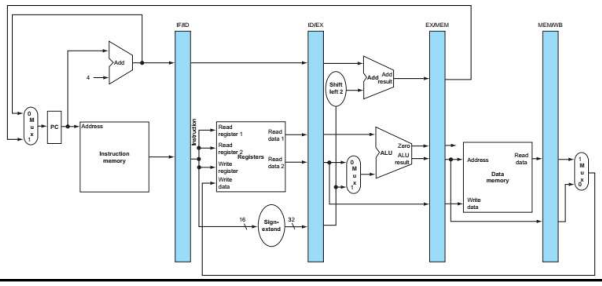
¿Existe conflicto?

sub $8,$5,$2 add $2,$3,$4

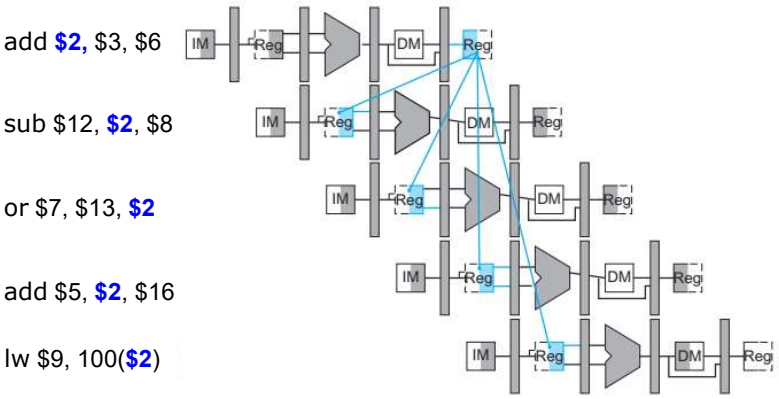


Microprocesador MIPS
Pipeline - Registros

- Escritura en banco de registros
  - en flanco ascendente la instrucción lw escribe en el ciclo 6
    - el alcance de riesgo por dependencia RAW es de tres instrucciones
  - en flanco descendente la instrucción lw escribe en el ciclo 5
    - el alcance de riesgo por dependencia RAW es de dos instrucciones
    - en el estado alto del ciclo se debe resolver el mux



Microprocesador MIPS
Pipeline – Riesgos por dependencias RAW



Microprocesador MIPS
Pipeline – Riesgos por dependencias RAW

i-ésimo ciclo

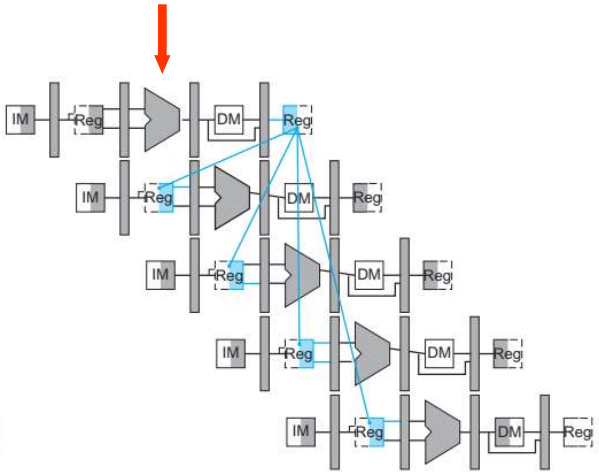
EX add **$2,** $3, $6

ID sub $12, **$2**, $8

IF or $7, $13, **$2**

add $5, **$2**, $16

lw $9, 100(**$2**)



Microprocesador MIPS
Pipeline – Riesgos por dependencias RAW

i-ésimo ciclo

El valor que corresponde de $2 no se encuentra almacenado

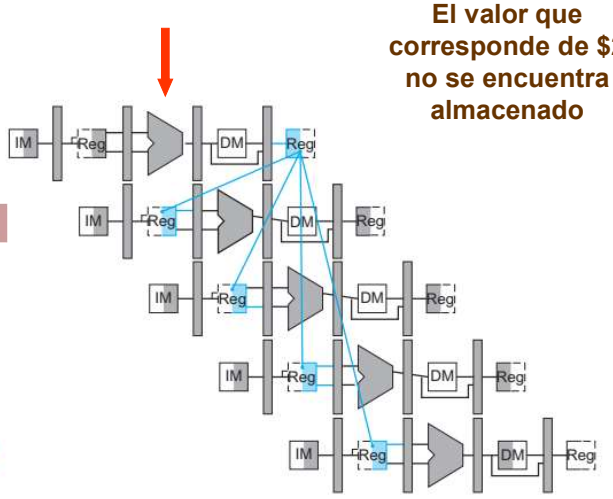
EX add **$2,** $3, $6

ID sub $12, **$2**, $8

IF or $7, $13, **$2**

add $5, **$2**, $16

lw $9, 100(**$2**)



Microprocesador MIPS
Pipeline – Riesgos por dependencias RAW

(*i*+1)-ésimo ciclo

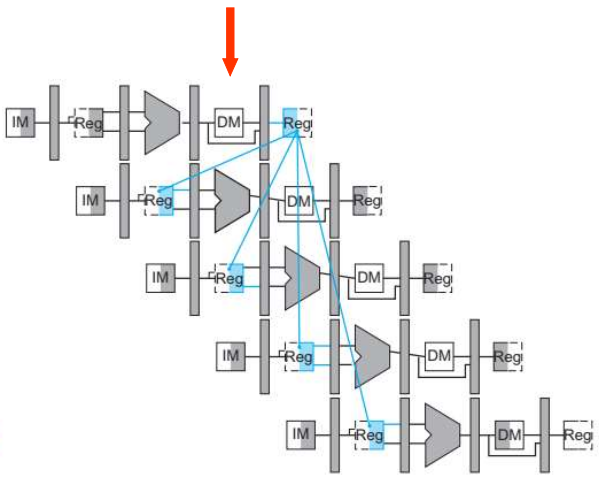
MEM add **$2,** $3, $6

EX sub $12, **$2**, $8

ID or $7, $13, **$2**

IF add $5, **$2**, $16

lw $9, 100(**$2**)



Microprocesador MIPS
Pipeline – Riesgos por dependencias RAW

(*i*+1)-ésimo ciclo

El valor que corresponde de $2 no se encuentra almacenado

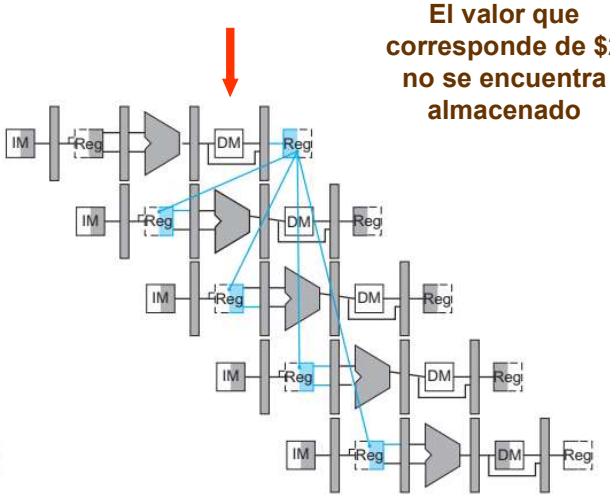
MEM add **$2,** $3, $6

EX sub $12, **$2**, $8

ID or $7, $13, **$2**

IF add $5, **$2**, $16

lw $9, 100(**$2**)



Microprocesador MIPS
Pipeline – Riesgos por dependencias RAW

(*i*+2)-ésimo ciclo

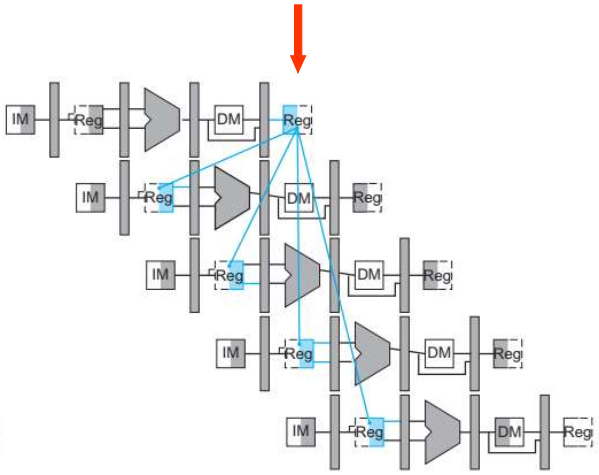
WB add **$2,** $3, $6

MEM sub $12, **$2**, $8

EX or $7, $13, **$2**

ID add $5, **$2**, $16

IF lw $9, 100(**$2**)



Microprocesador MIPS
Pipeline – Riesgos por dependencias RAW

(*i*+2)-ésimo ciclo

El valor que corresponde de $2 se encuentra almacenado en el flanco descendente

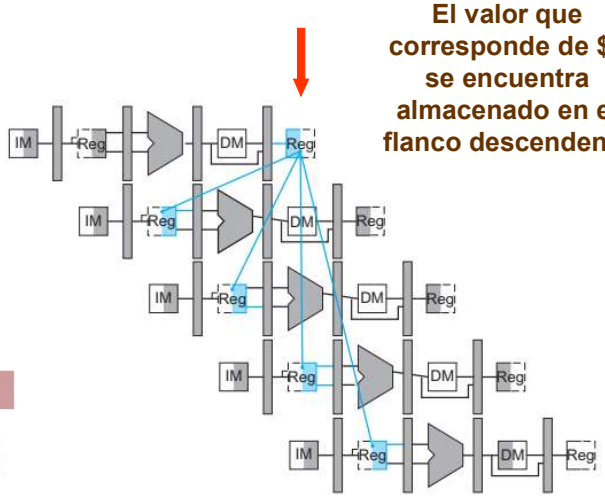
WB add **$2,** $3, $6

MEM sub $12, **$2**, $8

EX or $7, $13, **$2**

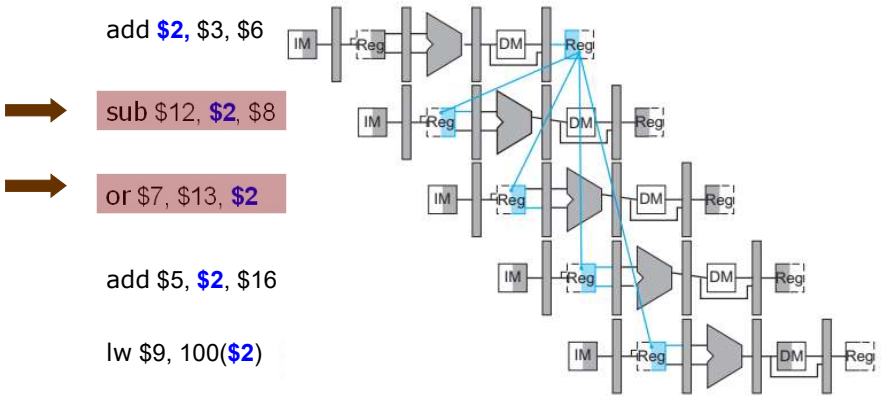
ID add $5, **$2**, $16

IF lw $9, 100(**$2**)



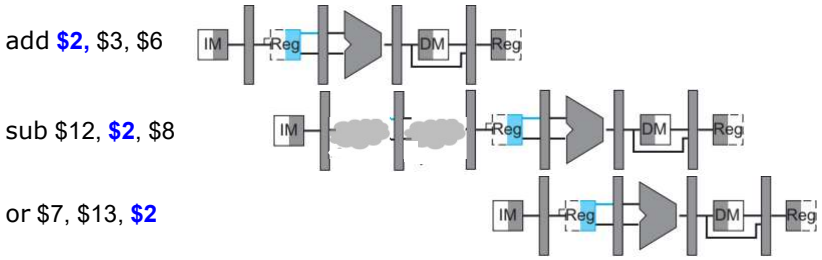
Microprocesador MIPS
Pipeline – Riesgos por dependencias RAW

instrucciones con conflictos por dependencia de datos RAW



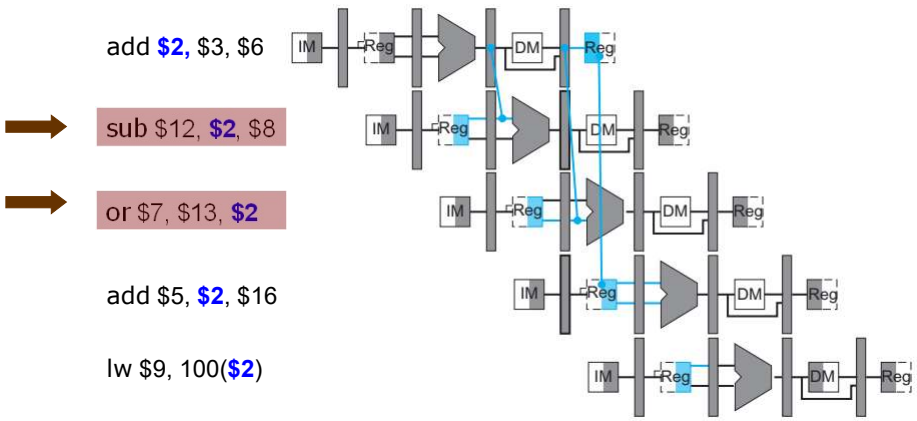
Microprocesador MIPS
Pipeline – Riesgos por dependencias RAW

- Una forma de solucionarlo es esperando o deteniendo el *pipelining*
  - incrementa el CPI



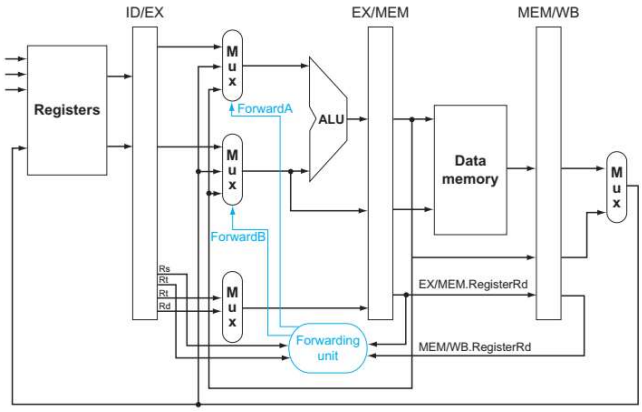
Microprocesador MIPS
Pipeline – Riesgos por dependencias RAW

- Otra manera de solucionarlo es mediante adelantamiento de datos



Microprocesador MIPS
Pipeline – Riesgos por dependencias RAW

- Unidad de adelantamiento de datos (*Forwarding*)
  - Un bloque combinacional para detectar el riesgo
  - Multiplexores para adelantar los datos



Microprocesador MIPS
Pipeline – Riesgos por dependencias RAW

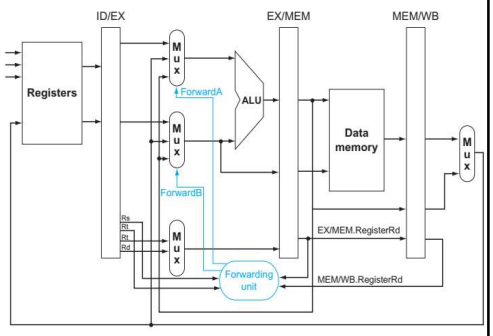
- Unidad de adelantamiento de datos (*Forwarding*)

EX/MEM.RegisterRd= ID/EX.RegisterRs

EX/MEM.RegisterRd= ID/EX.RegisterRt

MEM/WB.RegisterRd= ID/EX.RegisterRs

MEM/WB.RegisterRd= ID/EX.RegisterRt



Unidad de adelantamiento considera 4 condiciones para establecer los controles de los multiplexores (*ForwardA* y *ForwardB*)

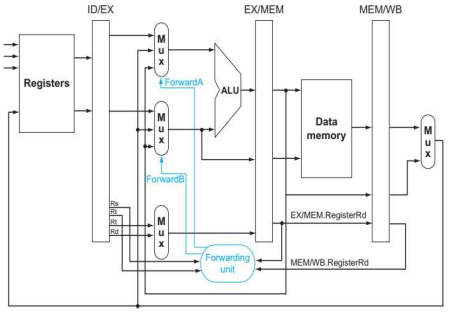
Microprocesador MIPS
Pipeline – Riesgos por dependencias RAW

- Unidad de adelantamiento de datos (*Forwarding*)

Riesgo en EX

if EX/MEM.RegWrite **and**
EX/MEM.RegisterRd<>0 **and**
EX/MEM.RegisterRd=ID/EX.RegisterRs)
then
ForwardA = 10

if EX/MEM.RegWrite **and**
EX/MEM.RegisterRd<>0 **and**
EX/MEM.RegisterRd=ID/EX.RegisterRt)
then
ForwardB = 10



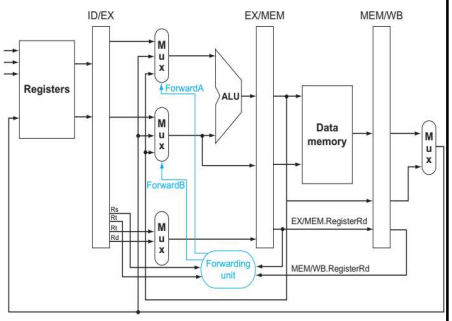
Microprocesador MIPS
Pipeline – Riesgos por dependencias RAW

- Unidad de adelantamiento de datos (*Forwarding*)

Riesgo en MEM

if MEM/WB.RegWrite **and**
MEM/WB.RegisterRd<>0 **and**
not (EX/MEM.RegWrite **and**
EX/MEM.RegisterRd=ID/EX.RegisterRs)
and
MEM/WB.RegisterRd=ID/EX.RegisterRs
then
ForwardA = 01

if MEM/WB.RegWrite **and**
MEM/WB.RegisterRd<>0 **and**
not (EX/MEM.RegWrite **and**
EX/MEM.RegisterRd=ID/EX.RegisterRt)
and
MEM/WB.RegisterRd=ID/EX.RegisterRt
then
ForwardB = 01



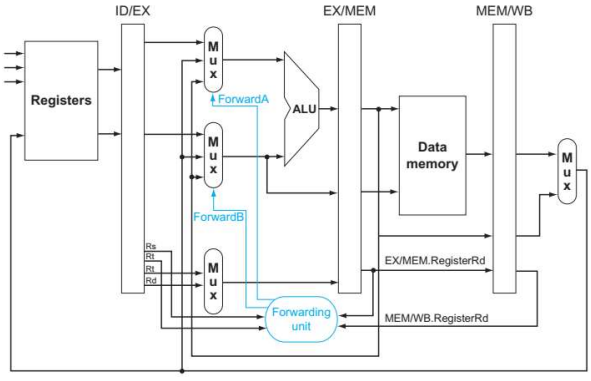
Microprocesador MIPS
Pipeline – Riesgos por dependencias RAW

- Riesgo insalvable con Unidad de adelantamiento

lw **$2**, 100($4)

add $5, **$2**, $4

Hasta que no accede a memoria, no se sabe el contenido que tendrá $2



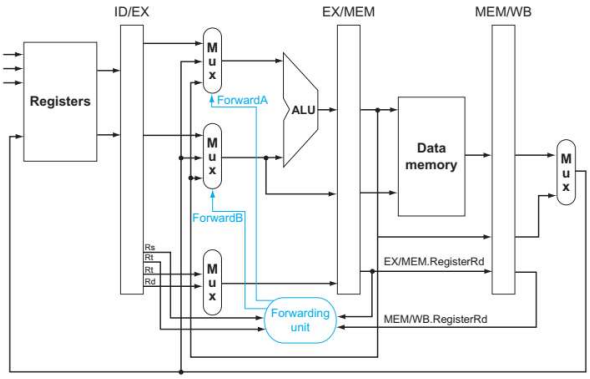
Microprocesador MIPS
Pipeline – Riesgos por dependencias RAW

- Riesgo insalvable con Unidad de adelantamiento

lw **$2**, 100($4)

add $5, **$2**, $4

Hasta que no accede a memoria, no se sabe el contenido que tendrá $2



Se debe bloquear el *pipeline* un ciclo o que el compilador agregue una instrucción **nop** luego del **lw**

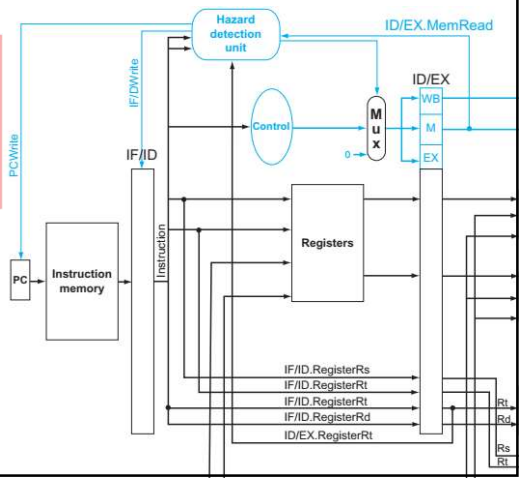
Microprocesador MIPS
Pipeline – Riesgos por dependencias RAW

- Unidad de detección de riesgo (*Hazard Detection Unit*)
  - para los riesgos insalvables con *Forwarding Unit*

if ID/EX.MemRead **and**
(ID/EX.RegisterRt=IF/ID.RegisterRs
or
(ID/EX.RegisterRt=IF/ID.RegisterRt)
then
bloqueo de *pipeline*

Bloqueo del *pipeline*:

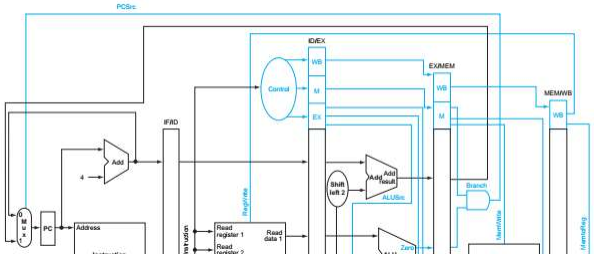
- **PCWrite** = 0
- **IF/IDWrite** = 0
- **MuxNOP** = 1



Microprocesador MIPS
Pipeline - Riesgos

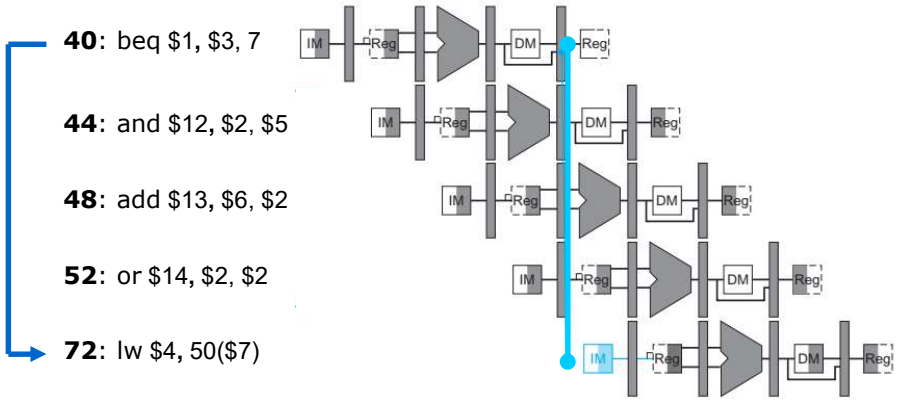
- Riesgos de Control
  - producido por las instrucciones de salto
  - la condición de salto se determina en la etapa EXE
  - en la cuarta etapa MEM, se actualiza el PC con la dirección de salto efectiva
  - pérdida de tres ciclos en el caso que se cumpla la condición

Posible mejora puede ser adelantar el cálculo de la dirección y condición en la segunda etapa



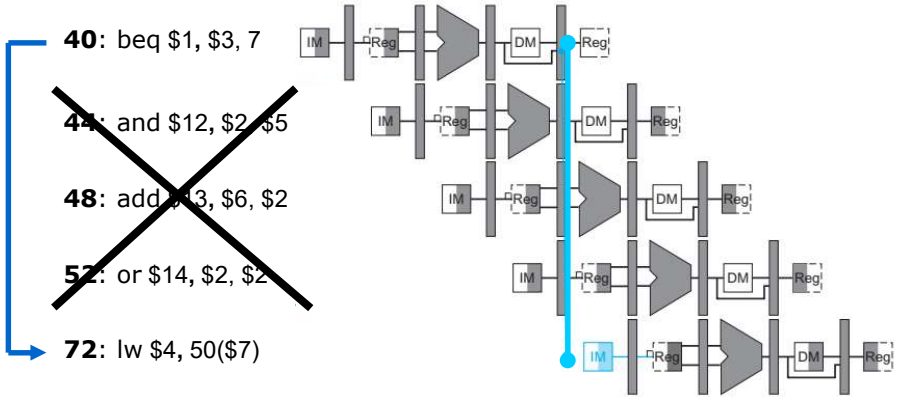
Microprocesador MIPS
Pipeline - Riesgos

- Riesgos de Control
  - sin la mejora de adelantar cálculos para la instrucción de salto se pierden tres ciclos



Microprocesador MIPS
Pipeline - Riesgos

- Riesgos de Control
  - sin la mejora de adelantar cálculos para la instrucción de salto se pierden tres ciclos



Microprocesador MIPS
Pipeline – Riesgos de Control

Qué hacer con siguientes instrucciones al salto condicional

- Salto retardado, las instrucciones posteriores siempre se ejecutan
  - El compilador rellena con instrucciones válidas los huecos del retardo
- Esperar hasta que la dirección y condición de salto estén definidas
- Asumir que no se da la condición del salto.
  - En caso de error se vacía el *pipeline*
- Reducir o adelantar el cálculo de la dirección y condición de salto
  - Agrega lógica adicional en etapa que realiza el cálculo.
  - Posible adelantamiento de datos involucrados para el cálculo de la condición
- Predicción dinámica del salto
  - Se ejecuta especulativamente, en caso de error se vacía el *pipeline*
  - Implementación basada en buffer de predicción y/o máquinas de estados

Microprocesador MIPS
Pipeline – Riesgos de Control

Qué hacer con siguientes instrucciones al salto condicional

- **Salto retardado, las instrucciones posteriores siempre se ejecutan**
  - **El compilador rellena con instrucciones válidas los huecos del retardo**
- Esperar hasta que la dirección y condición de salto estén definidas
- Asumir que no se da la condición del salto.
  - En caso de error se vacía el *pipeline*
- Reducir o adelantar el cálculo de la dirección y condición de salto
  - Agrega lógica adicional en etapa que realiza el cálculo.
  - Posible adelantamiento de datos involucrados para el cálculo de la condición
- Predicción dinámica del salto
  - Se ejecuta especulativamente, en caso de error se vacía el *pipeline*
  - Implementación basada en buffer de predicción y/o máquinas de estados

Microprocesador MIPS
Pipeline – Riesgos de Control

Qué hacer con siguientes instrucciones al salto condicional

- Salto retardado, las instrucciones posteriores siempre se ejecutan
  - El compilador rellena con instrucciones válidas los huecos del retardo

- **Esperar hasta que la dirección y condición de salto estén definidas**

- Asumir que no se da la condición del salto.
  - En caso de error se vacía el *pipeline*
- Reducir o adelantar el cálculo de la dirección y condición de salto
  - Agrega lógica adicional en etapa que realiza el cálculo.
  - Posible adelantamiento de datos involucrados para el cálculo de la condición
- Predicción dinámica del salto
  - Se ejecuta especulativamente, en caso de error se vacía el *pipeline*
  - Implementación basada en buffer de predicción y/o máquinas de estados

Microprocesador MIPS
Pipeline – Riesgos de Control

Qué hacer con siguientes instrucciones al salto condicional

- Salto retardado, las instrucciones posteriores siempre se ejecutan
  - El compilador rellena con instrucciones válidas los huecos del retardo
- Esperar hasta que la dirección y condición de salto estén definidas
- **Asumir que no se da la condición del salto.**
  - **En caso de error se vacía el *pipeline***
- Reducir o adelantar el cálculo de la dirección y condición de salto
  - Agrega lógica adicional en etapa que realiza el cálculo.
  - Posible adelantamiento de datos involucrados para el cálculo de la condición
- Predicción dinámica del salto
  - Se ejecuta especulativamente, en caso de error se vacía el *pipeline*
  - Implementación basada en buffer de predicción y/o máquinas de estados

Microprocesador MIPS
Pipeline – Riesgos de Control

Qué hacer con siguientes instrucciones al salto condicional

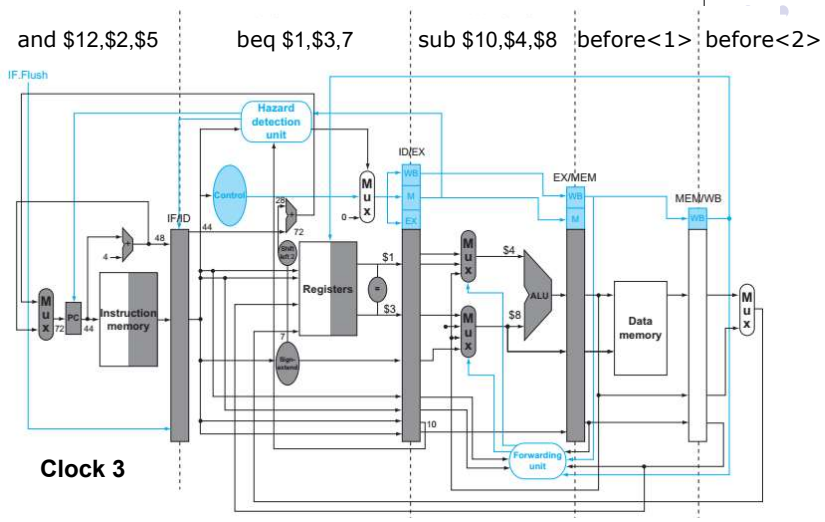
- Salto retardado, las instrucciones posteriores siempre se ejecutan
  - El compilador rellena con instrucciones válidas los huecos del retardo
- Esperar hasta que la dirección y condición de salto estén definidas
- Asumir que no se da la condición del salto.
  - En caso de error se vacía el *pipeline*

- **Reducir o adelantar el cálculo de la dirección y condición de salto**
  - **Agrega lógica adicional en etapa que realiza el cálculo.**
  - **Posible adelantamiento de datos involucrados para el cálculo de la condición**

- Predicción dinámica del salto
  - Se ejecuta especulativamente, en caso de error se vacía el *pipeline*
  - Implementación basada en buffer de predicción y/o máquinas de estados

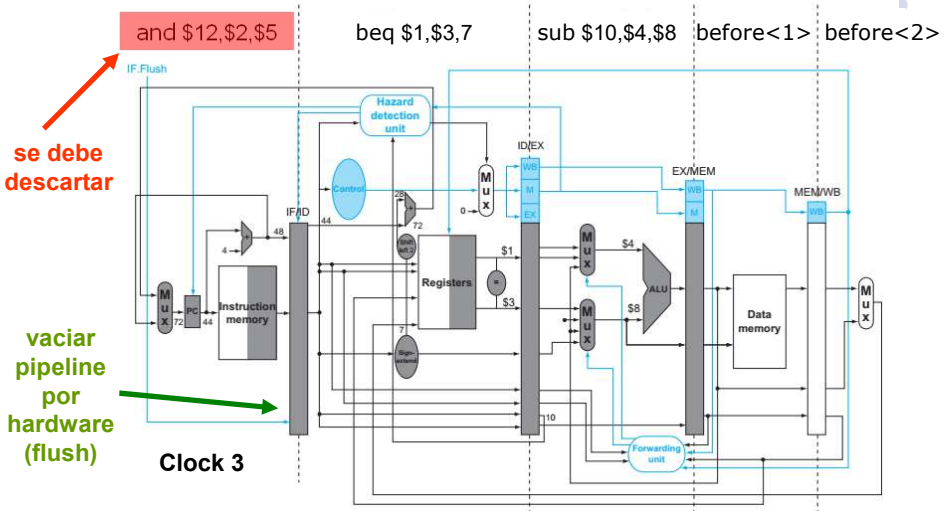
Microprocesador MIPS

Pipeline – Riesgos de Control



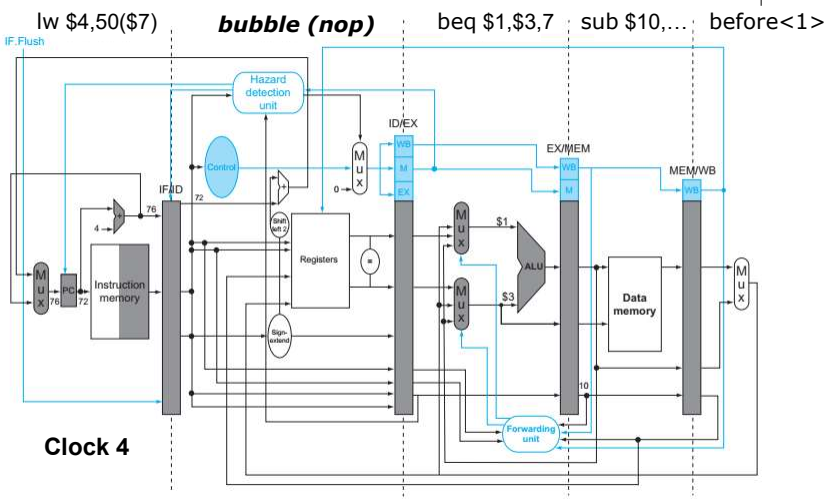
Microprocesador MIPS

Pipeline – Riesgos de Control



Se carga la instrucción nop **en el registro de segmentación IF/ID**

Microprocesador MIPS
Pipeline – Riesgos de Control



Microprocesador MIPS
Pipeline – Riesgos de Control

Qué hacer con siguientes instrucciones al salto condicional

- Salto retardado, las instrucciones posteriores siempre se ejecutan
  - El compilador rellena con instrucciones válidas los huecos del retardo
- Esperar hasta que la dirección y condición de salto estén definidas
- Asumir que no se da la condición del salto.
  - En caso de error se vacía el *pipeline*
- Reducir o adelantar el cálculo de la dirección y condición de salto
  - Agrega lógica adicional en etapa que realiza el cálculo.
  - Posible adelantamiento de datos involucrados para el cálculo de la condición
- **Predicción dinámica del salto**
  - **Se ejecuta especulativamente, en caso de error se vacía el *pipeline***
  - **Implementación basada en buffer de predicción y/o máquinas de estados**

Pipelining y Paralelismo

- El *pipelining* explota el paralelismo a nivel instrucción (ILP – *instruction-level paralellism*)
- En el MIPS escalar con *pipeline*
  - se trata de lograr que el CPI=1, es difícil debido a los **Riesgos**
  - las instrucciones son “lanzadas” (*launched*) una por ciclo de reloj
- Como mejora. Replicar componentes del procesador para “lanzar” más de una instrucción por ciclo de reloj (*Multiple Issue*)
  - posibilita CPI<1
  - *Multiple Issue* estático
    - decisiones tomadas en compilación: VLIW (*Very Long Instruction Word*)
  - *Multiple Issue* dinámico
    - decisiones tomadas durante la ejecución: Superescalares