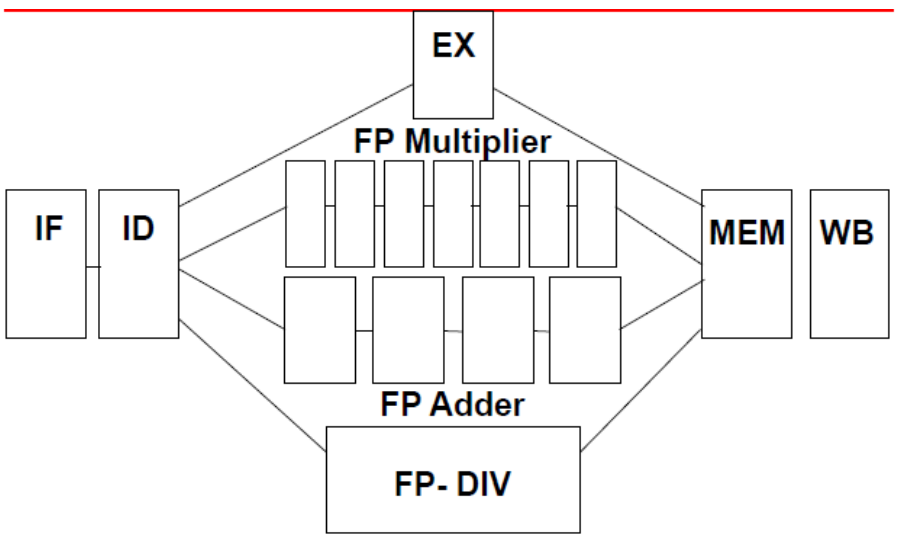
Segmentación de Cause

- Si las etapas son independientes podemos ejecutarlas en paralelo
- Mientras ejecutamos una etapa de una instrucción, ejecutamos otra etapa de otra instrucción.
- De esta manera usamos mejor la CPU, todas las etapas estan en funcionamiento en cada ciclo de CPU.
  - Cada instrucción tarda en ejecutarse 5 ciclos
  - 1 instrucción, 5 ciclos
  - 2 instrucciones, 6 ciclos
  - 3 instrucciones, 7 ciclos
  - 10 instrucciones, 14 ciclos
  - n cantidad de instrucciones se ejecutaran en $4 + n$ ciclos.
- A esta técnica se la llama Segmentación de Cause

Ciclo de Instruccion de WinMIPS64

- Busqueda (IF)
  - Se accede a la memoria buscando la instrucción
  - Se incrementa el PC
- Decodificación / Búsqueda de Operandos (ID)
  - Se decodifica la instrucción
  - Se accede al banco de registro por los operandos en la segunda mitad del ciclo. No puede avanzar si no estan disponibles.
  - Si es un salto condicional, se determina si hay que realizarlo o no
  - Si hay que ejecutar un salto, también se calcula el nuevo PC
- Ejecución / Dirección Efectiva (EX)
  - Si es una instrucción aritmetico/logica, se ejecuta en la ALU
  - Si es acceso a memoria, se calcula la dirección efectiva
- Acceso a Memoria (MEM)
  - Si es un acceso a memoria, se accede
- Almacenamiento (WB)
  - Se almacena el resultado (si existiese) en el banco de registros en la primera mitad del ciclo

- Cada etapa se ejecuta en un ciclo de reloj, salvo algunas operaciones aritmeticas de punto flotante, que tienen unidades de ejecuciión separadas
- Las etapas de cada uniad de ejecución dependen de la operación a realizar:
  - Suma (PF) se realiza en 4 ciclos, con 4 etapas
  - Multiplicación se realiza en 7 ciclos, con 7 etapas
  - División se realiza en 24 ciclos, 1 etapa
- Distintas unidades de ejecución pueden estar activas al mismo tiempo



Introducción

Suposiciones:

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

Algunos Problemas

Problemas:

- No todas las instrucciones necesitan todas las etapas
  - *SW RT, inmed(RS)* - no utiliza W
  - En VonSim *MOV AX, mem* no requiere EX
- No todas las etapas pueden ser manejadas en paralelo.
  - F y M acceden a memoria
- No se tienen en cuenta los saltos de control.

Atascos

Llamamos *atasco* a la situación que impide a una o mas instrucciones seguir su camino en el cauce.
Los atascos pueden ocurrir por 3 razones:

- Estructural
  - Provocados por conflictos con los recursos.
- Dependencia de Datos
  - Dos instrucciones se comunican por medio de un dato
- Dependencia de Control
  - La ejecución de una instrucción depende de cómo se ejecute otra

Si resolvemos con paradas del cauce, disminuye el rendimiento

Atasco por Dependencia 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.

- Lectura después de Escritura (RAW, dependencia verdadera)
  - una instrucción genera un dato que lee otra posterior
- Escritura después de Escritura (WAW, dependencia en salida)
  - una instrucción escribe un dato después que otra posterior
  - sólo se da si se deja que las instrucciones se adelanten unas a otras
- Escritura después de Lectura (WAR, antidependencia)
  - una instrucción modifica un valor antes de que otra anterior que lo tiene que leer lo lea
  - Es el que menos suele darse

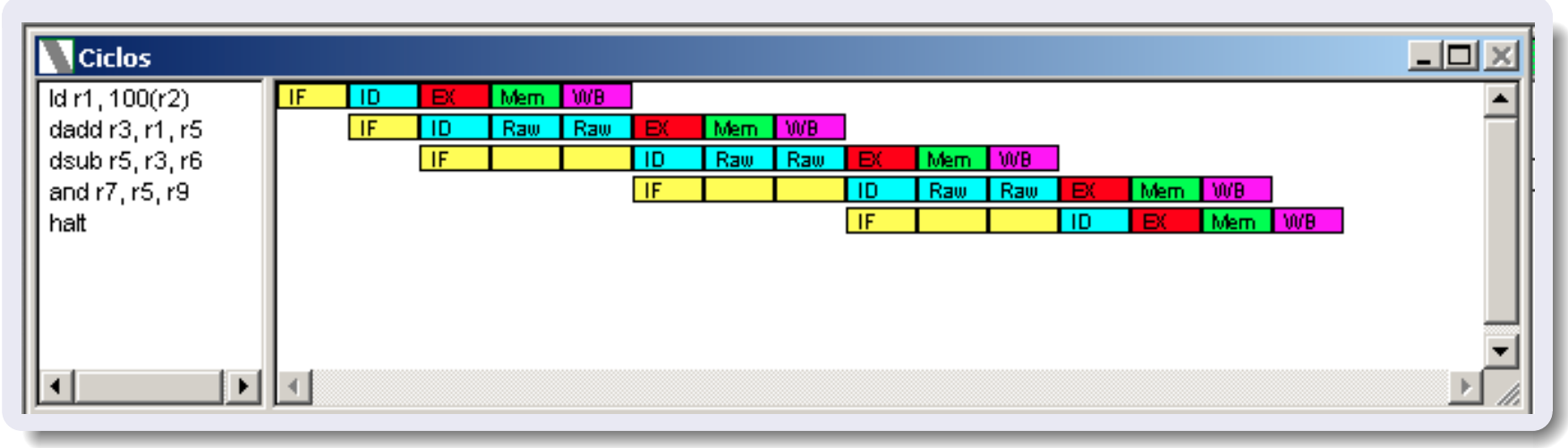
RAW

Una instrucción depende del resultado de otra instrucción que todavía no ha finalizado y debe esperar a que el resultado este disponible.

```
.code
LD   r1, 100(r2)
DADD r3, r1, r5
DSUB r5, r3, r6
AND  r7, r5, r9
```

RAW

Una instrucción depende del resultado de otra instrucción que todavía no ha finalizado y debe esperar a que el resultado este disponible.



General

- Se debe determinar cómo y cuando aparecen esos riesgos
- Hay dos tipos de soluciones
  - Software
    - Podemos agregar instrucciones *NOP*, o reordenar las instrucciones
  - Hardware
    - Adelantamiento de Operandos (Forwarding)

Agregar Instrucciones NOP

Cambiemos el ejemplo visto anteriormente

```
.code
ld r1, 100(r2)
nop
nop
dadd r3, r1, r5
nop
nop
dsub r5, r3, r6
nop
nop
and r7, r5, r9
halt
```

Ahora no hay atascos, pero tenemos más instrucciones

Reordenar instrucciones

Veamos otro ejemplo

```
.data
A:  .word 5
.code
ld r1, A(r0)
dadd r1, r1, r1
daddi r2, r0, 3
dadd r3, r0, r0
halt
```

Tenemos un atasco con el registro *r1*.

Reordenar instrucciones

Retrasando el *dadd* sobre el registro *r*1.

```
.data
A:  .word 5
.code
ld r1, A(r0)
daddi r2, r0, 3
dadd r3, r0, r0
dadd r1, r1, r1
halt
```

No hay atascos ahora.

Forwarding

Forwarding, Adelantamiento o Cortocircuito

- Consiste en pasar directamente el resultado obtenido con una instrucción a las instrucciones que lo necesitan como operando.
- Si el dato necesario está disponible antes se lleva a la entrada de la etapa correspondiente sin esperar a llegar a la etapa escritura de escritura del banco de registros.
- La idea es tener disponible el operando lo antes posible para no perder ciclos
- Usar esta técnica no implica la eliminación de todos los atascos.
- En WinMIPS64 podemos activarlo o desactivarlo en cualquier momento. Un cambio en esta configuración implica reiniciar la simulación.

Forwarding

Forwarding, Adelantamiento o Cortocircuito

- Cuando no existe Forwarding, una instrucción debe tener todos sus operandos (registros) disponibles en la etapa ID
- Cuando esta activo los registros pueden adelantarse una vez calculados o leidos a la etapa que los necesita. No siempre esa etapa es ID.
- Que instrucciones pueden adelantar un operando:
  - Si la instrucción es de lectura de memoria, puede adelantar el operando luego de ejecutar la etapa MEM.
  - Si es una instrucción aritmetico / lógica, luego de ejecutada la etapa EX.
- Si el operando se podia adelantar en EX, podrá también adelantarlo en MEM si una instrucción lo requiere.

Forwarding

Forwarding, Adelantamiento o Cortocircuito

- Cuando se requiere un operando:
  - Si la instrucción es de escritura de memoria, el operando a escribir, se necesita en MEM.
  - Si el operando se necesita para una operación aritmetico / lógica, se necesita en la etapa EX.
  - Si el operando se necesita para una instrucción de salto condicional, se necesita en la etapa ID.

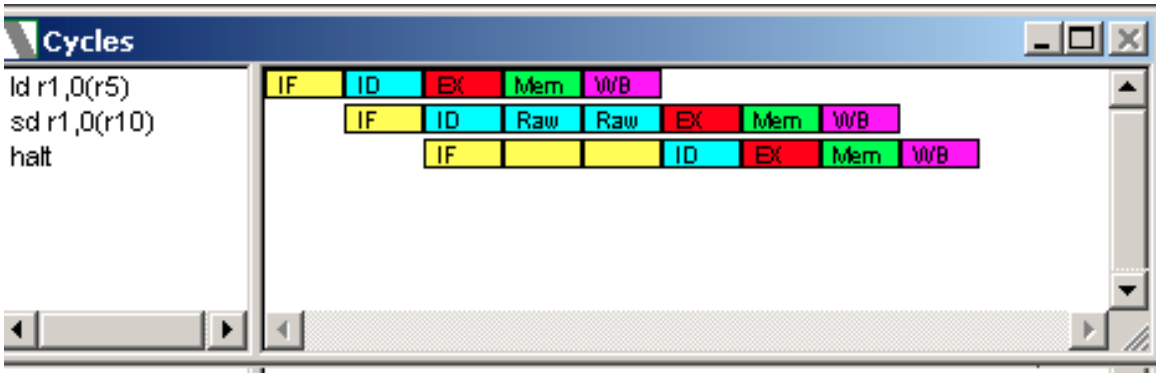
Forwarding

Forwarding, Adelantamiento o Cortocircuito

- Situaciones donde pueden ocurrir forwarding
  - Lectura seguida de Escritura
  - Lectura seguida de Aritmetico / Lógica
  - Lectura seguida de Salto Condicional
  - Aritmetica / Lógica seguida de Escritura
  - Aritmetica / Lógica seguida de Aritmetico / Logica
  - Aritmetica / Lógica seguida de Salto Condicional
  - Lectura seguida de Lectura
  - Aritmetica / Lógica seguida de Lectura

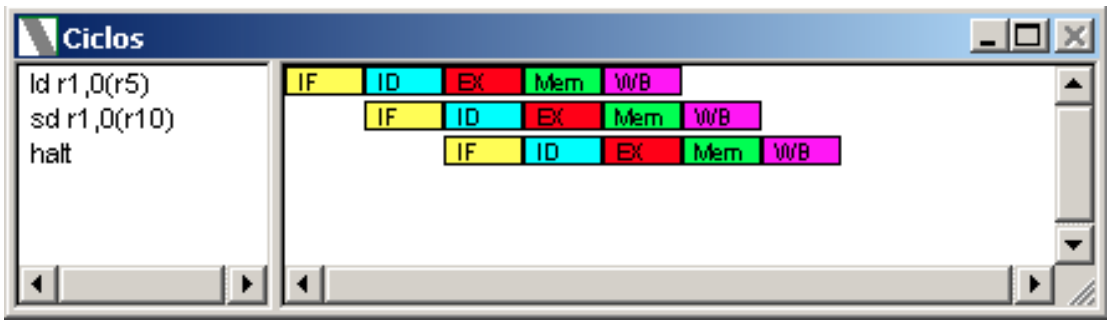
Forwarding - Lectura seguida de escritura

- Lectura seguida de Escritura
  - El operando a escribir en memoria viene de una instrucción de lectura anterior.
  - La instrucción *sd* debe esperar a que la instrucción *ld* se complete, esperando en *ID*



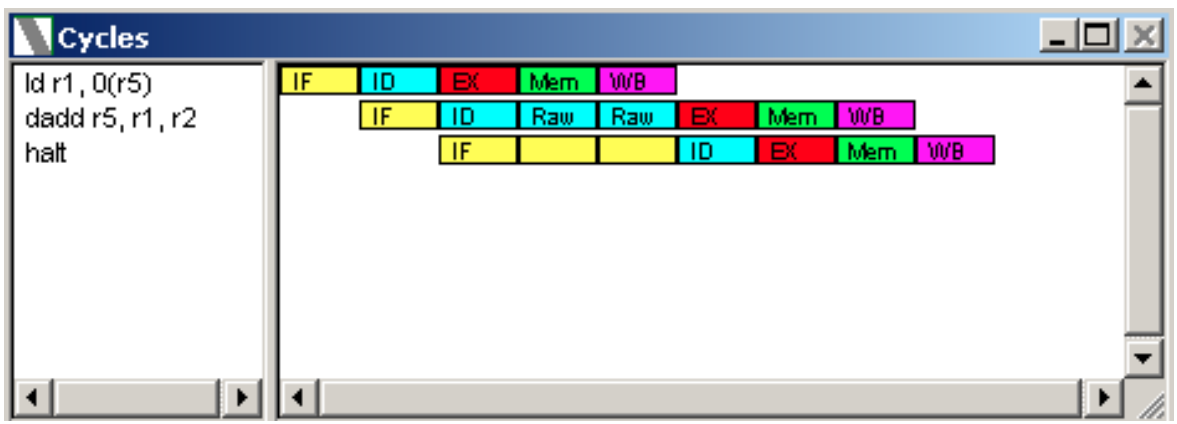
Forwarding - Lectura seguida de escritura

- Lectura seguida de Escritura
  - Con el adelantamiento activo, cuando la instrucción *ld* pasa la etapa *MEM*, ya puede adelantar el operando
  - La instrucción *sd* necesita el registro r1 en la etapa de *MEM*
  - Por lo tanto ya no hay atascos RAW.



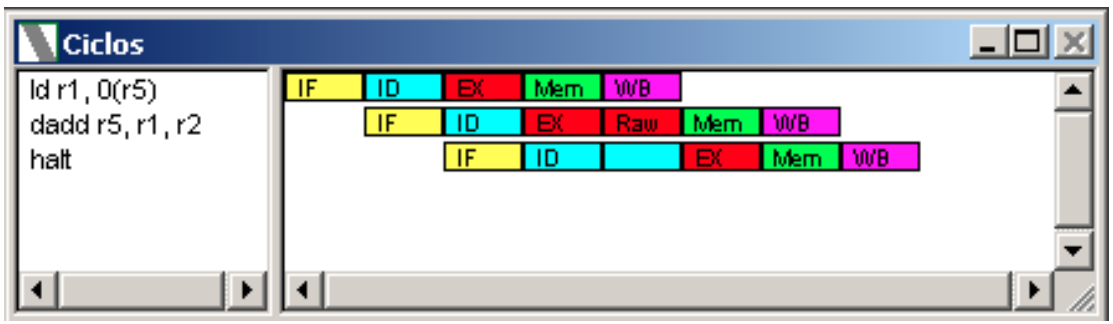
Forwarding - Lectura seguida de Aritmetico / Logica

- Lectura seguida de Aritmetico / Lógica
  - Uno de los operandos de una instrucción aritmetico lógico proviene de una lectura de memoria
  - La instrucción *daddi* debe esperar a que la instrucción *ld* se complete, esperando en *ID*



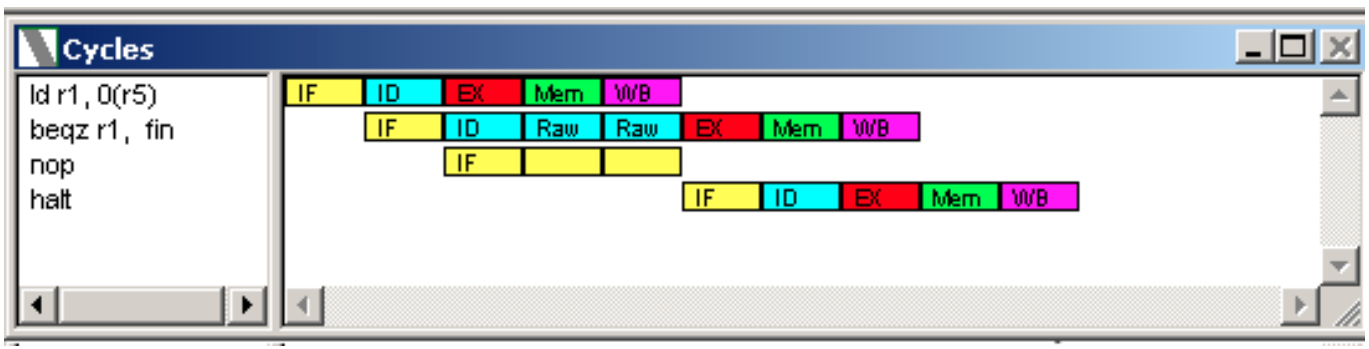
Forwarding - Lectura seguida de Aritmetico / Logica

- Lectura seguida de Aritmetico / Lógica
  - Con el adelantamiento activo, cuando la instrucción *ld* pasa la etapa *MEM*, ya puede adelantar el operando
  - La instrucción *daddi* necesita el registro r1 en la etapa de *EX*
  - Como *daddi* debe esperar en *EX* a que *ld* pase la etapa *MEM*, se produce un solo atasco RAW.



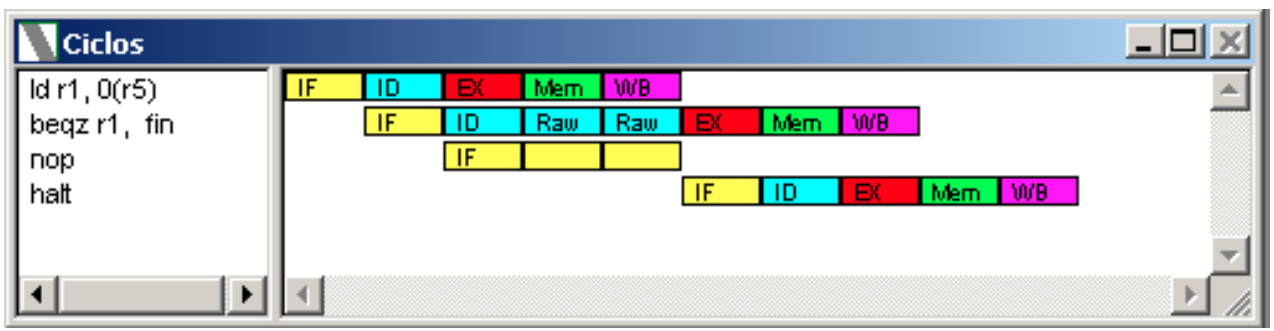
Forwarding - Lectura seguida de Salto Condicional

- Lectura seguida de Salto Condicional
  - Uno de los operandos de una instrucción de salto condicional proviene de una lectura de memoria
  - La instrucción *beqz* debe esperar a que la instrucción *ld* se complete, esperando en *ID*



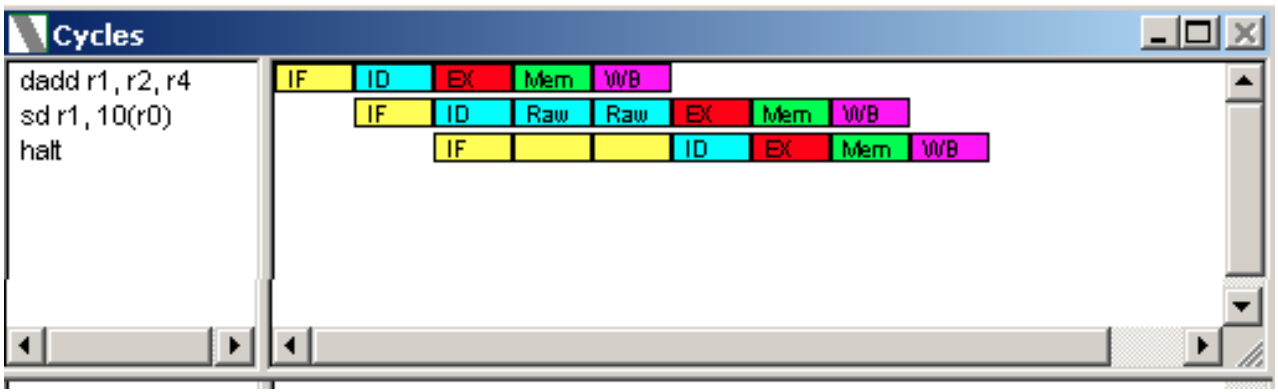
Forwarding - Lectura seguida de Salto Condicional

- Lectura seguida de Salto Condicional
  - Con el adelantamiento activo, cuando la instrucción *ld* pasa a la etapa *MEM*, ya puede adelantar el operando
  - La instrucción *beqz* necesita el registro r1 en la etapa de *ID*
  - Por lo tanto esta situación no cambia



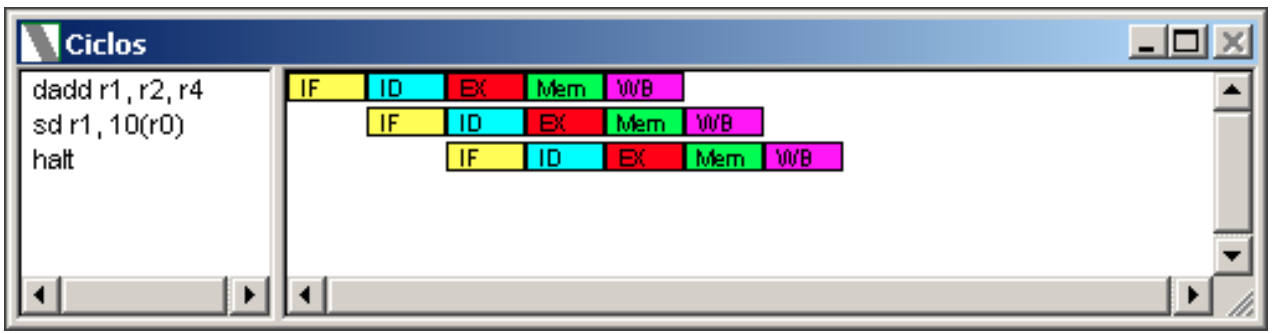
Forwarding - Aritmetica / Lógica seguida de Escritura

- Aritmetica / Lógica seguida de Escritura
  - El operando a escribir en memoria viene de una instrucción aritmetico lógica anterior.
  - La instrucción *sd* debe esperar a que la instrucción *daddi* se complete, esperando en *ID*



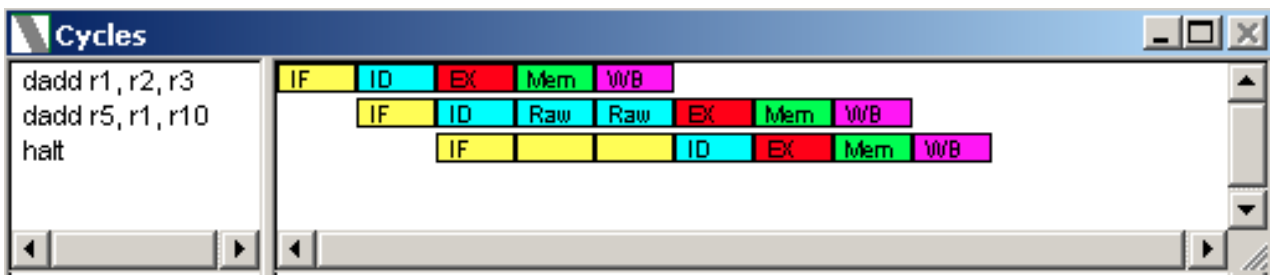
Forwarding - Aritmetica / Lógica seguida de Escritura

- Aritmetica / Lógica seguida de Escritura
  - Con el adelantamiento activo, cuando la instrucción *daddi* pasa la etapa *EX*, ya puede adelantar el operando
  - La instrucción *sd* necesita el registro r1 en la etapa de *MEM*
  - Por lo tanto ya no hay atascos RAW.



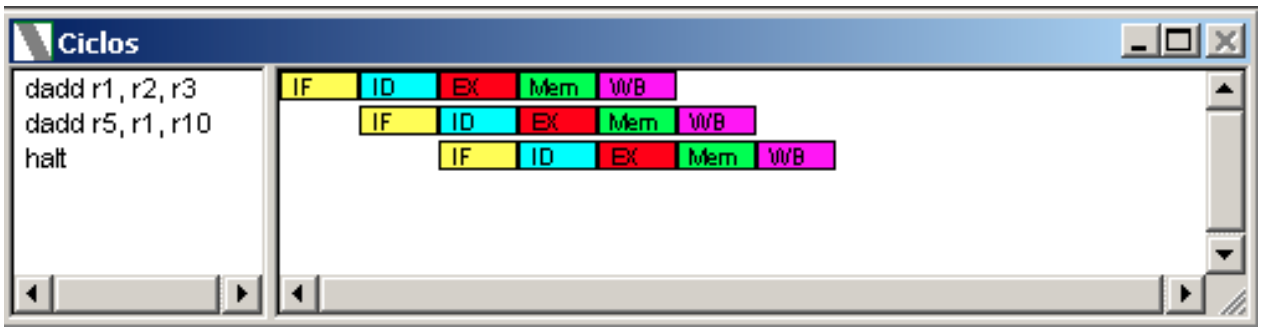
Forwarding - Aritmetica / Lógica seguida de Aritmetico / Logica

- Aritmetica / Lógica seguida de Aritmetico / Logica
  - Un operando de una instrucción aritmetico lógica viene de una instrucción aritmetico lógica anterior.
  - La segunda instrucción *daddi* debe esperar a que la primer instrucción *daddi* se complete, esperando en *ID*



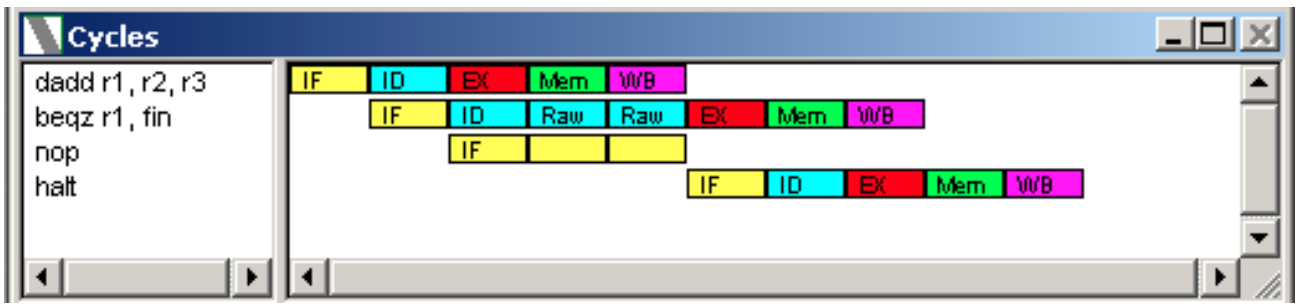
Forwarding - Aritmetica / Lógica seguida de Aritmetico / Logica

- Aritmetica / Lógica seguida de Aritmetico / Logica
  - Con el adelantamiento activo, cuando la primer instrucción *daddi* pasa la etapa *EX*, ya puede adelantar el operando
  - La segunda instrucción *daddi* necesita el registro r1 en la etapa de *EX*
  - Por lo tanto ya no hay atascos RAW.



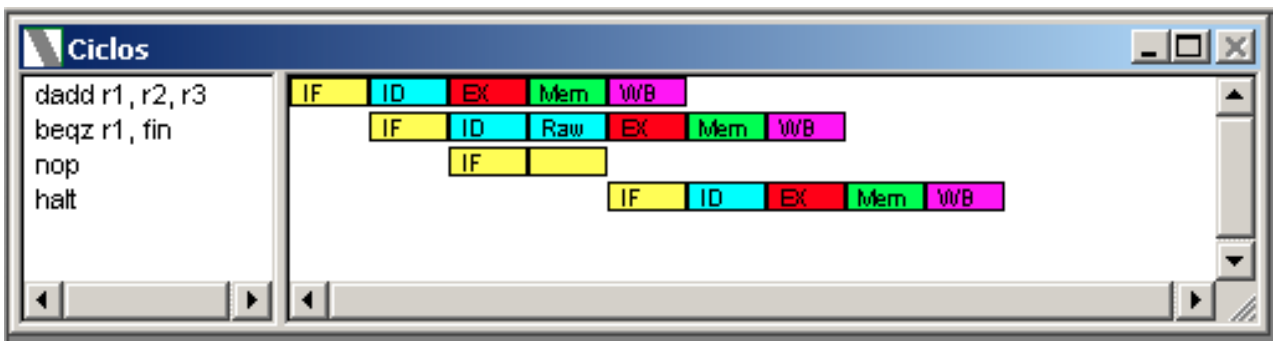
Forwarding - Aritmetica / Lógica seguida de Salto Condicional

- Aritmetica / Lógica seguida de Salto Condicional
  - Un operando de una instrucción de salto condicional viene de una instrucción aritmetico lógica anterior.
  - La instrucción *beqz* debe esperar a que la instrucción *daddi* se complete, esperando en *ID*



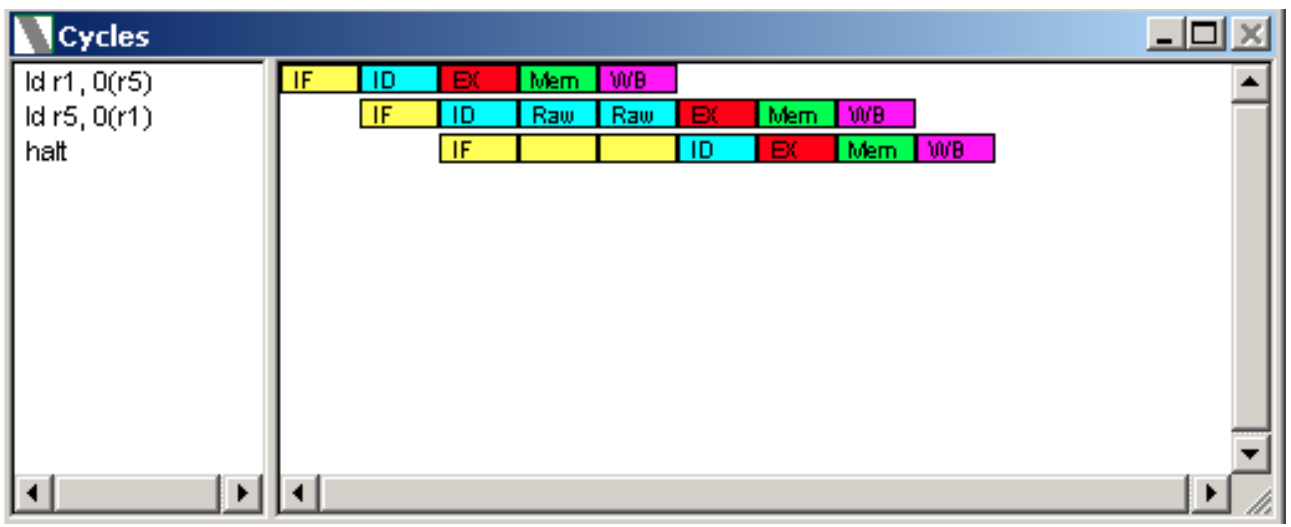
Forwarding - Aritmetica / Lógica seguida de Salto Condicional

- Aritmetica / Lógica seguida de Salto Condicional
  - Con el adelantamiento activo, cuando la instrucción *daddi* pasa la etapa *EX*, ya puede adelantar el operando
  - La instrucción *beqz* necesita el registro r1 en la etapa de *ID*
  - Como *beqz* debe esperar en *ID* a que *daddi* pase la etapa *EX*, se produce un solo atasco RAW.



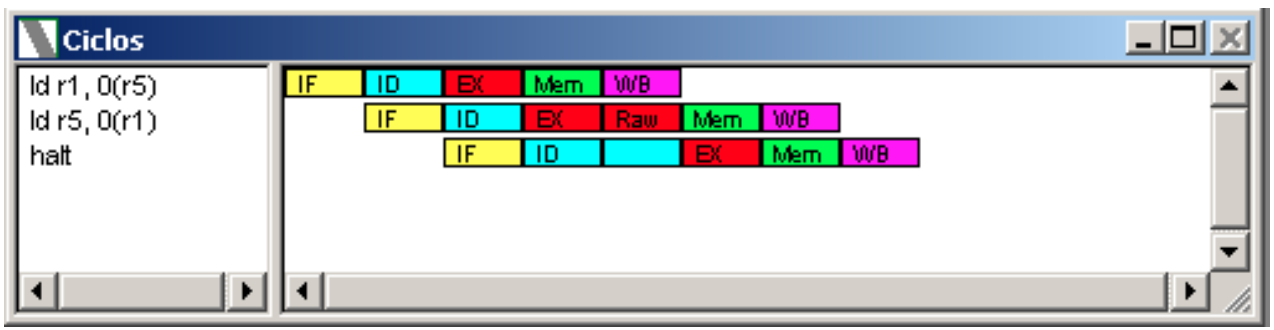
Forwarding - Lectura seguida de Lectura

- Lectura seguida de Lectura
  - El operando que calcula el desplazamiento en memoria de una escritura viene de una instrucción de lectura anterior.
  - La segunda instrucción *ld* debe esperar a que la primer instrucción *ld* se complete, esperando en *ID*



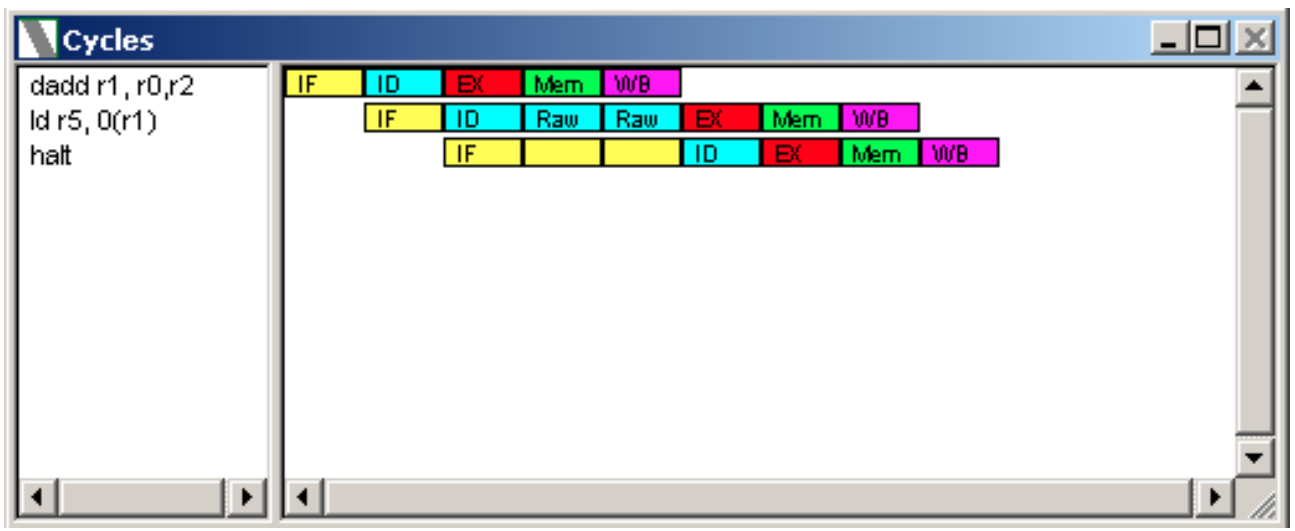
Forwarding - Lectura seguida de Lectura

- Lectura seguida de Lectura
  - Con el adelantamiento activo, cuando la primer instrucción *ld* pasa la etapa *MEM*, ya puede adelantar el operando
  - La segunda instrucción *ld* necesita acceder a r1 en la etapa *EX*
  - Como la segunda instrucción *ld* debe esperar en *EX* a que la primer instrucción *ld* pase la etapa *MEM*, se produce un solo atasco RAW.



Forwarding - Aritmetica / Lógica seguida de Lectura

- Aritmetica / Lógica seguida de Lectura
  - El operando que calcula el desplazamiento en memoria de una escritura viene de una instrucción aritmetico lógica anterior.
  - La instrucción *ld* debe esperar a que la instrucción *daddi* se complete, esperando en *ID*



Forwarding - Aritmetica / Lógica seguida de Lectura

- Aritmetica / Lógica seguida de Lectura
  - Con el adelantamiento activo, cuando la instrucción *daddi* pasa la etapa *EX*, ya puede adelantar el operando
  - La instrucción *ld* necesita acceder a r1 en la etapa *EX*
  - Por lo tanto ya no hay atascos RAW.