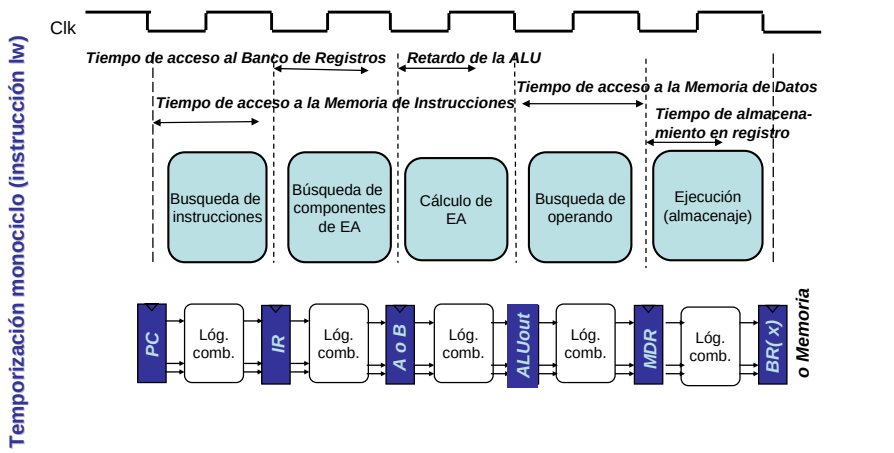
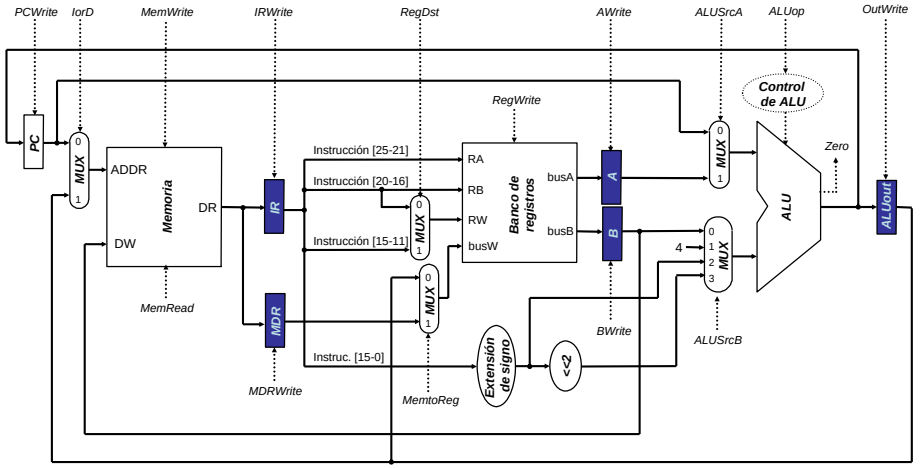
Procesador Segmentado

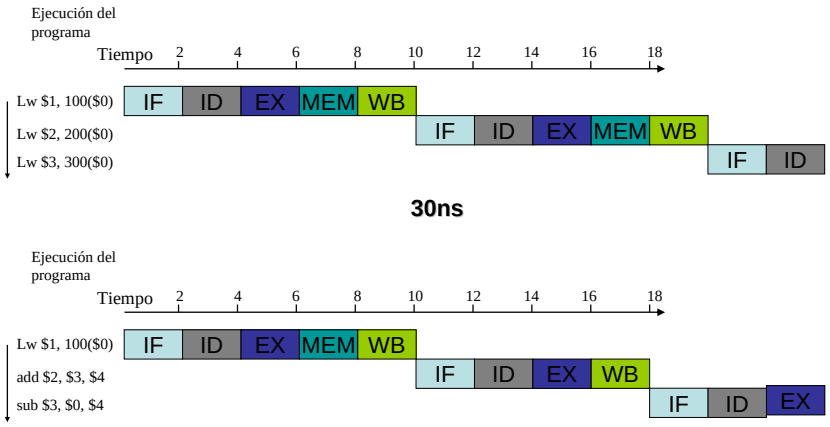
¿Qué es segmentación?



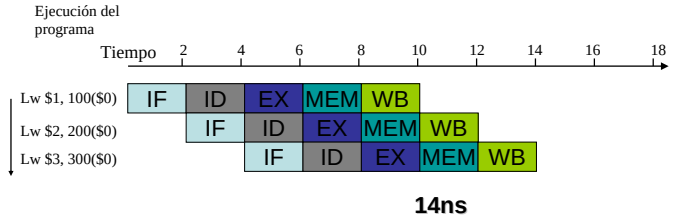
¿Qué es segmentación?

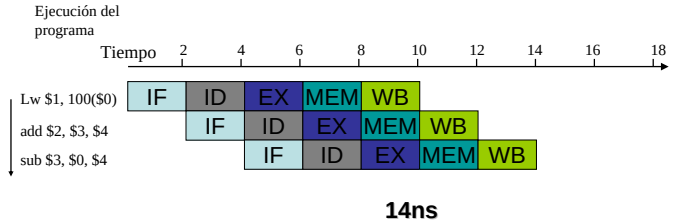


¿Qué es segmentación?



¿Qué es segmentación?

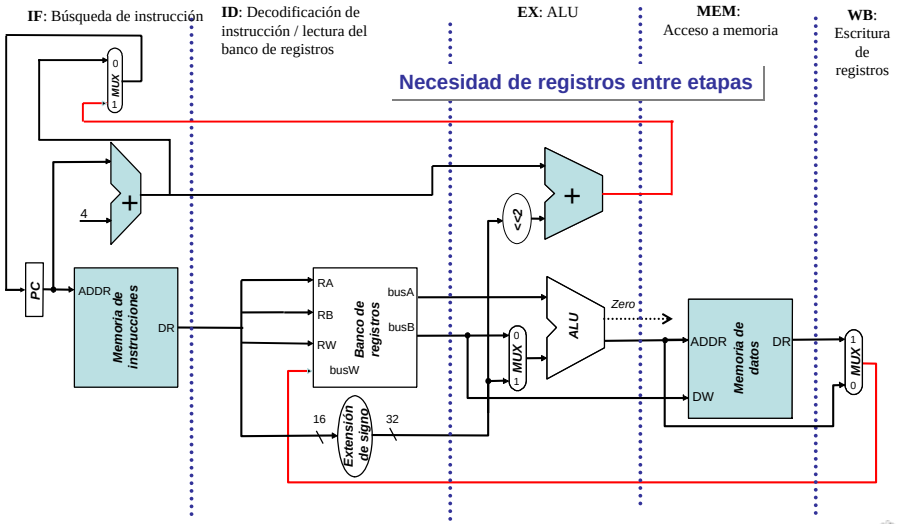




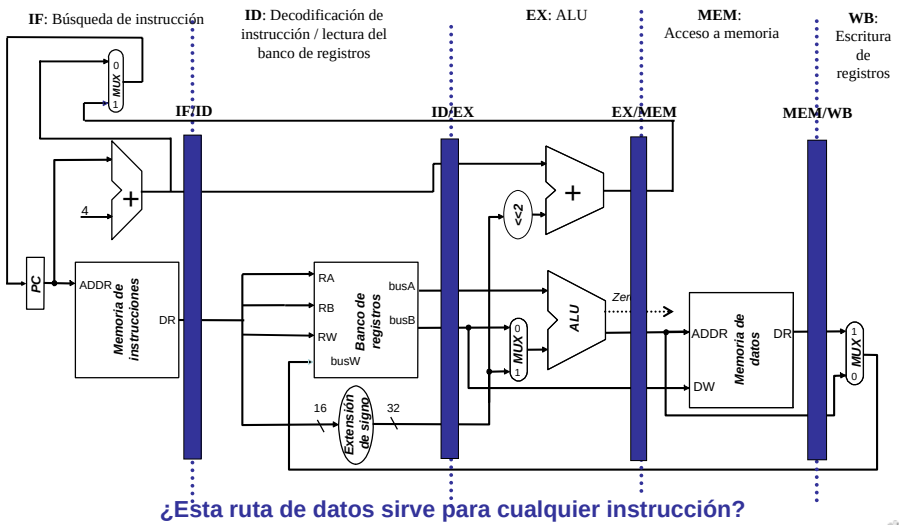
Introducción a la segmentación

- **¿Qué facilita la segmentación?**
  - Todas las instrucciones de igual longitud
  - Pocos formatos de instrucción
  - Operandos en memoria sólo en operaciones de carga y almacenamiento
- **¿Qué dificulta la segmentación?**
  - Conflictos estructurales
  - Conflictos de datos
  - Conflictos de control
  - Gestión de interrupciones
  - Ejecución fuera de orden

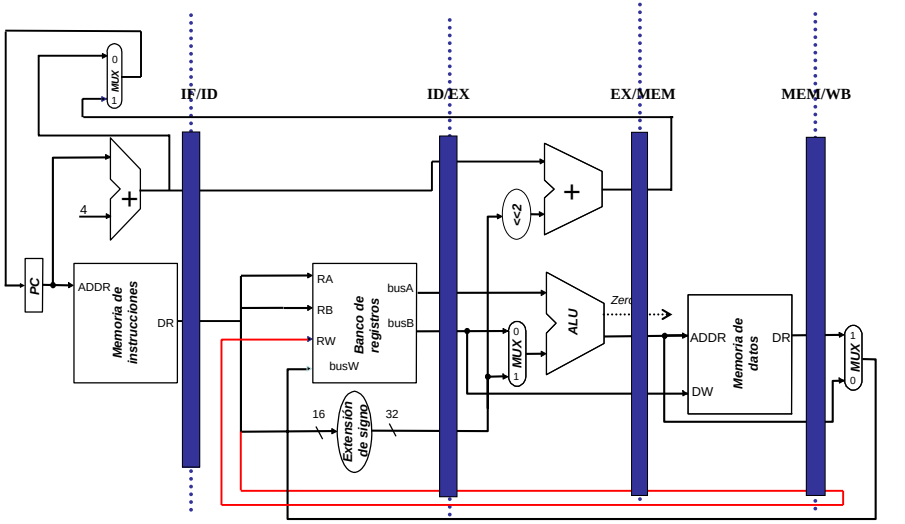
Introducción a la segmentación



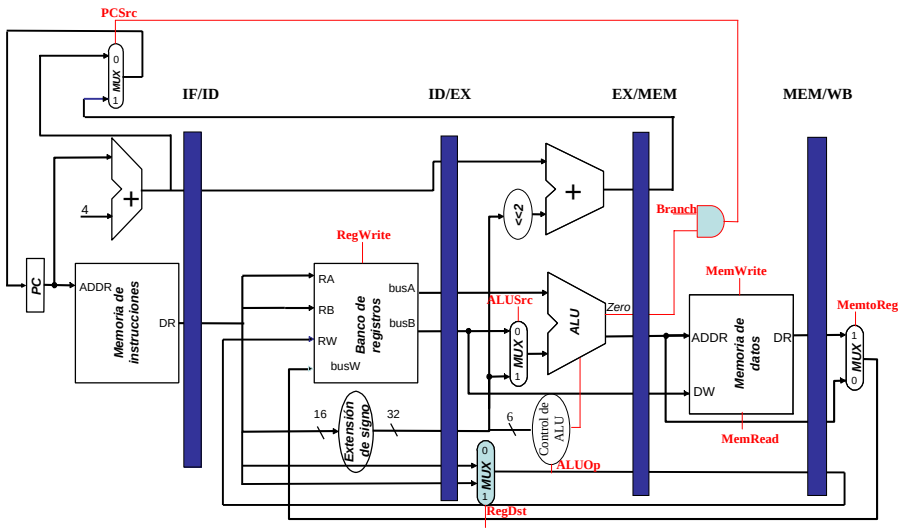
Ruta de datos segmentada



Ruta de datos segmentada



Ruta de datos segmentada



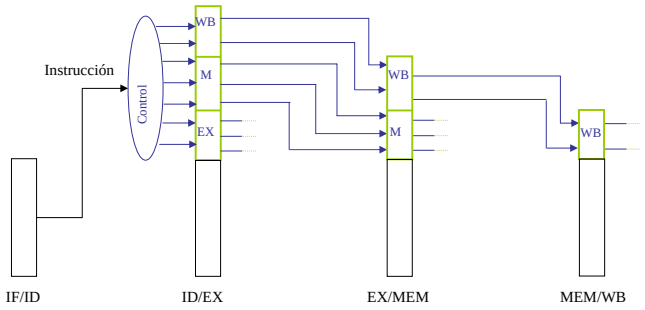
Ruta de datos segmentada

- Acciones a realizar:
  - Etapa IF: Búsqueda de la instrucción e incremento del PC
  - Etapa ID: Decodificación de la instrucción y búsqueda de operandos en los registros
  - Etapa EX: Ejecución
  - Etapa Mem: Acceso a la memoria de datos
  - Etapa WB: Escritura en el banco de registros.

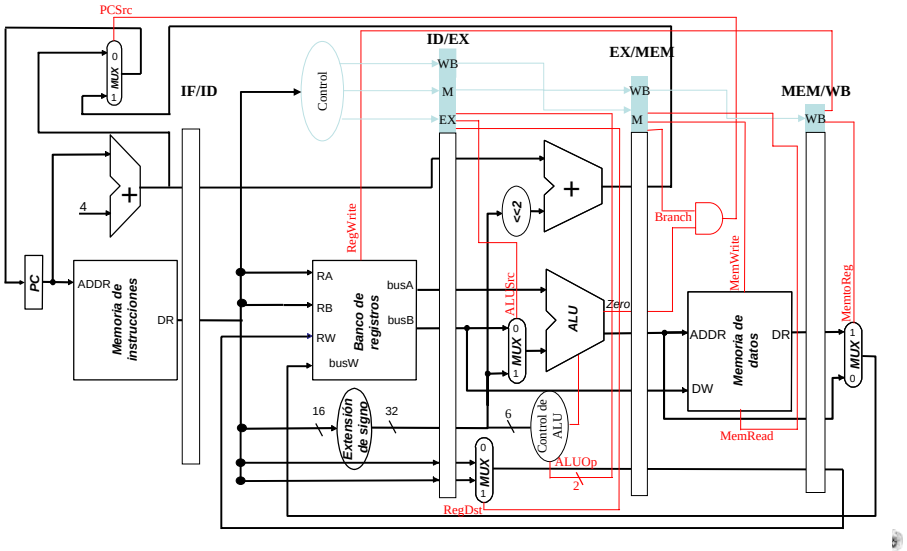
Instrucción	Líneas de control del estado de Ejecución/Cálculo de DE				Líneas de control del estado de acceso a memoria			Líneas de control del estado de WB	
	Reg Dst	ALU Op1	ALU Op0	ALU Src	Branch	Mem Read	Mem Write	Reg Write	Mem to Reg
Formato-R	1	1	0	0	0	0	0	1	0
Lw	0	0	0	1	0	1	0	1	1
Sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

Ruta de datos segmentada

Las señales de control se generan en la U.C. y se van pasando de una etapa a otra como si fuesen datos.



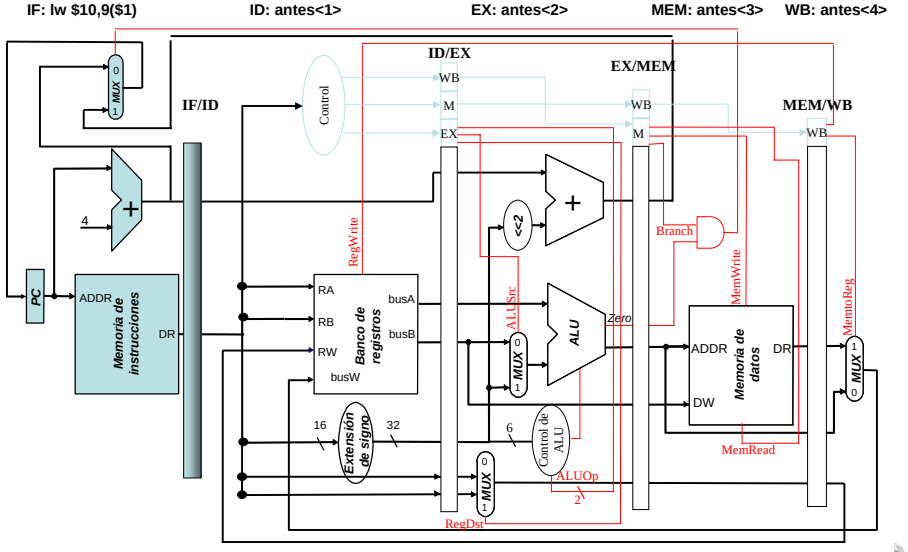
Control de la ruta de datos segmentada



Procesador Segmentado

Control de la ruta de datos segmentada

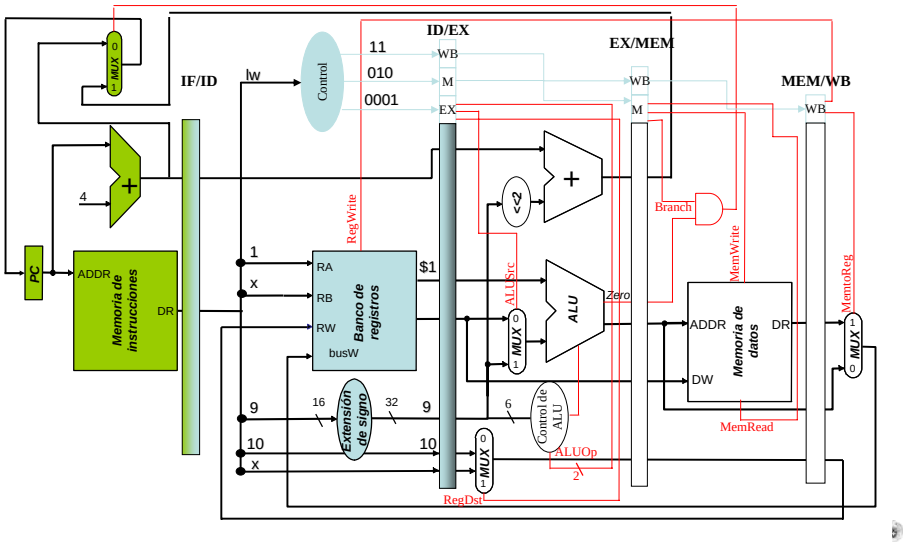
IF: lw $10,9($1) ID: antes<1> EX: antes<2> MEM: antes<3> WB: antes<4>



Procesador Segmentado

Control de la ruta de datos segmentada

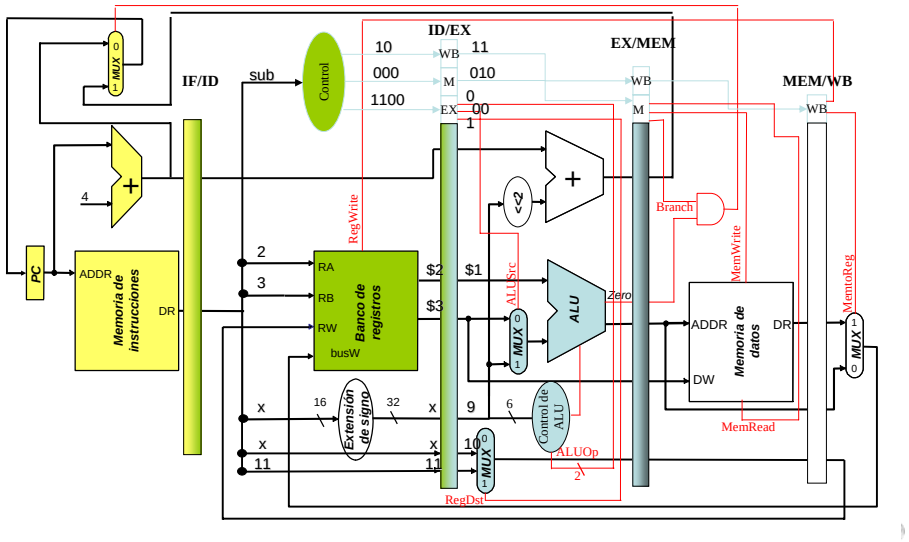
IF: sub $11,$2,$3 ID: lw $10,9($1) EX: antes<1> MEM: antes<2> WB: antes<3>



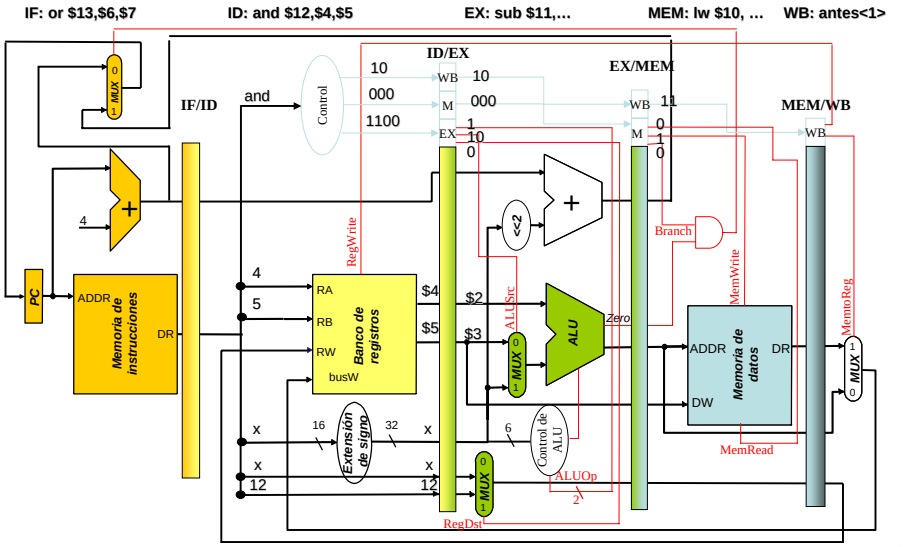
Procesador Segmentado

Control de la ruta de datos segmentada

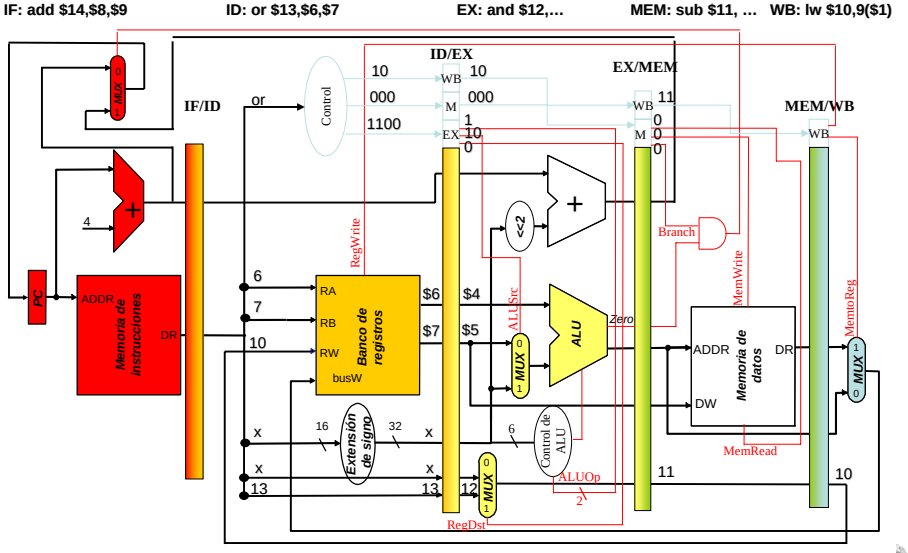
IF: and $12,$4,$5 ID: sub $11,$2,$3 EX: lw $10,... MEM: antes<1> WB: antes<2>



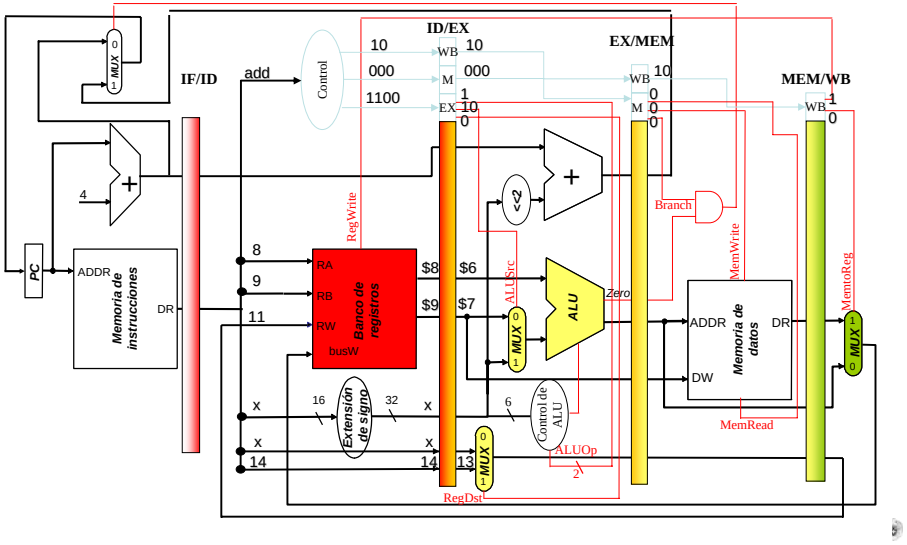
Control de la ruta de datos segmentada



Control de la ruta de datos segmentada



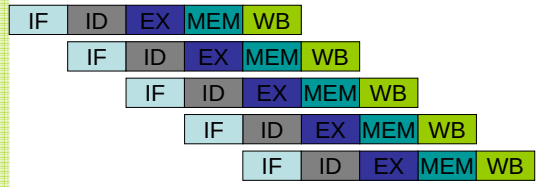
Control de la ruta de datos segmentada



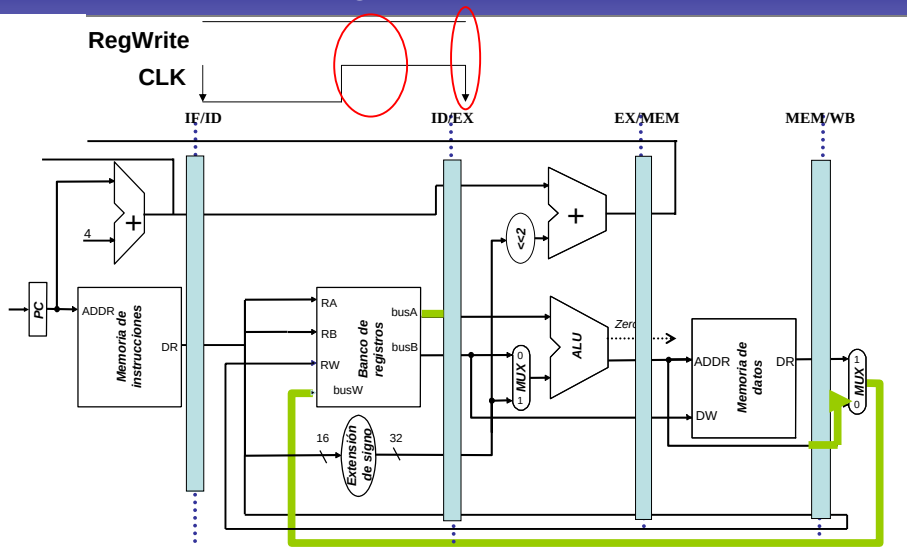
Ruta de datos segmentada

Ejemplo 1

add r1, r2 ,r3
sub r4, r1, r5
and r6, r1, r7
or r8, r1, r9
xor r10, r1, r11



Ruta de datos segmentada

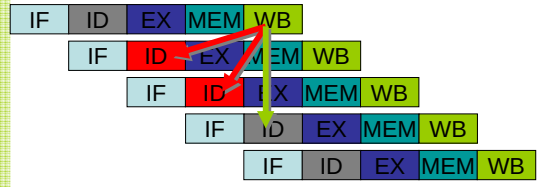


Ruta de datos segmentada

Ejemplo 1

add r1, r2 ,r3
sub r4, r1, r5
and r6, r1, r7
or r8, r1, r9
xor r10, r1, r11

El dato se guarda en el banco de registros en WB



Ruta de datos segmentada

Ejemplo 1

add r1, r2 ,r3
sub r4, r1, r5
and r6, r1, r7
or r8, r1, r9
xor r10, r1, r11

Ruta de datos segmentada

Ejemplo 2

```
.data 0
    a: .word 22
    b: .word 15
.text
main:
lw r2, 0(r3)
lw r3, 4(r3)
add r5, r6, r7
sub r4, r6, r7
sw r9, 8(r7)
trap 0
```

Ruta de datos segmentada

Ejemplo 3

```
.data 0
      a: .word 22
      b: .word 15
.text
main:
      lw r1, 0(r0)
      lw r2, 4(r0)
      add r3, r1, r2
      sub r3, r3, r1
      add r4, r1, r1
      sw 0(r0), r4
loop:       add r5, r6, r7
            sub r4, r5, r2
            sub r5, r7, r2
            subi r2, r2, 5
      bnez r2, loop
      addi r2, r2, 5
      trap 0
```


Conflictos

- Conflictos por escritura después de lectura

  sub $2, **$1**, $3
  add **$1**, $4, $5

  No existen en el MIPS

- Conflictos de escritura después de escritura

  sub **$2**, $1, $3
  add **$2**, $4, $5

  No existen en el MIPS

- Conflictos de lectura después de escritura (LDE)

  sub **$2**, $1, $3
  add $4, **$2**, $5
  or $6, $7, **$2**

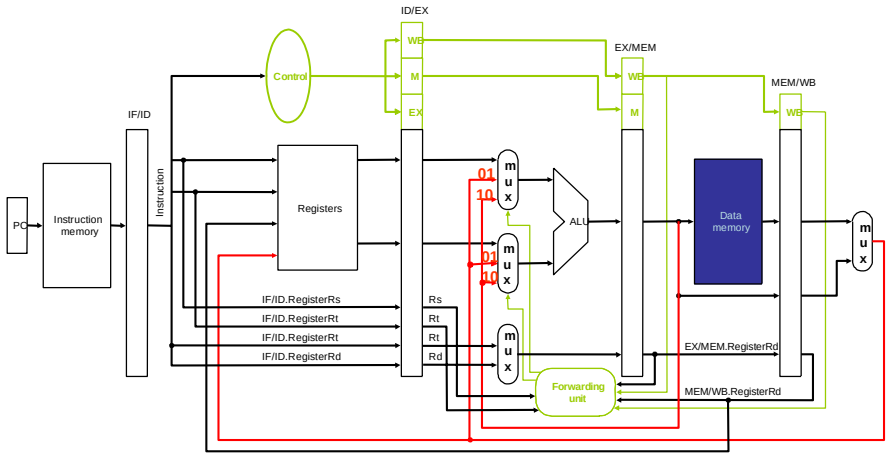
Conflicto

- Riesgo EX:
  - if (XM.RegWrite && (XM.Rd != 0) && (XM.Rd == DX.Rs)) {anticiparA = 10;}
  - if (XM.RegWrite && (XM.Rd != 0) && (XM.Rd == DX.Rt)) {anticiparB = 10;}
- Riesgo MEM:
  - if (MW.RegWrite && (MW.Rd != 0) && (XM.Rd != DX.Rs) && (MW.Rd == DX.Rs))

    {anticiparA = 01;}
  - if (MW.RegWrite && (MW.Rd != 0) && (XM.Rd != DX.Rt) && (MW.Rd == DX.Rt))

    {anticiparB = 01;}

MIPS con anticipación de datos



Ruta de datos segmentada

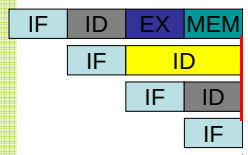
Ejemplo 3

```
add r1, r2 ,r3
sub r4, r1, r5
and r6, r1, r7
or r8, r1, r9
xor r10, r1, r11
```


Ruta de datos segmentada

Ejemplo 4

```
lw r1, 0(r0)
sub r4, r1, r5
and r6, r1, r7
or r8, r1, r9
xor r10, r1, r11
```

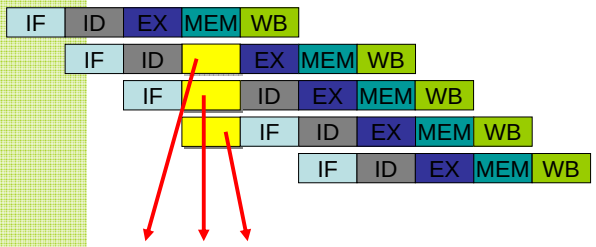


Interbloqueo de la segmentación
(burbuja o bloqueo del cauce)

Ruta de datos segmentada

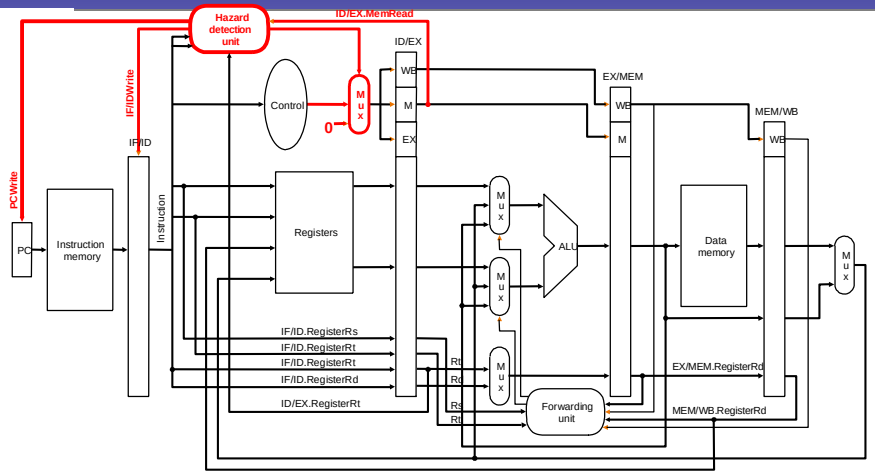
Ejemplo 4

```
lw r1, 0(r0)
sub r4, r1, r5
and r6, r1, r7
or r8, r1, r9
xor r10, r1, r11
```



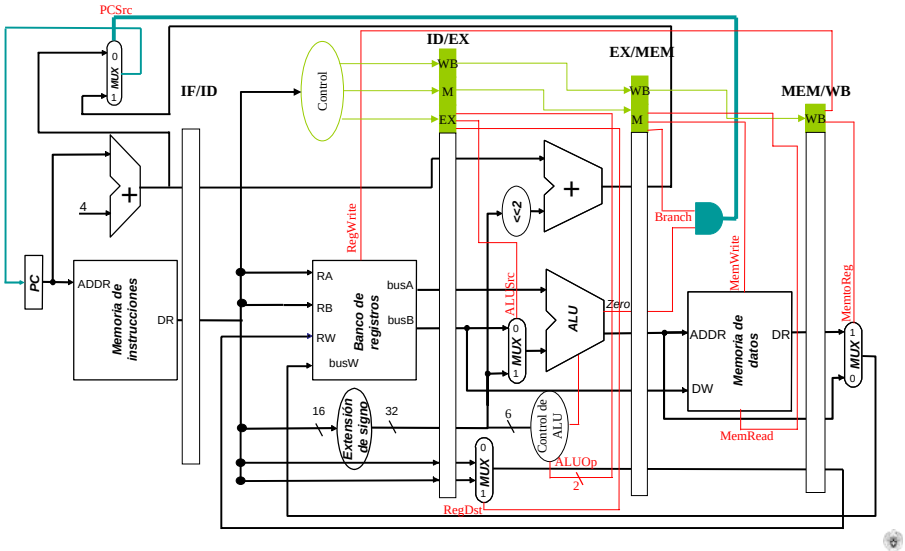
Bloqueo del cauce

Ruta de datos segmentada



Detección de conflicto de datos:
if (DX.MemRead && ((DX.Rt == FD.Rs) || (DX.Rt == FD.Rt))) { bloquear el *pipeline*;}

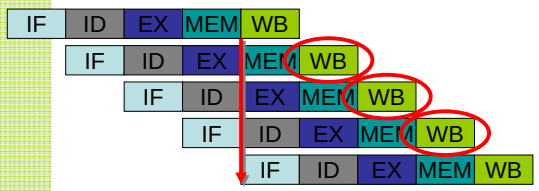
Dependencias de control/salto



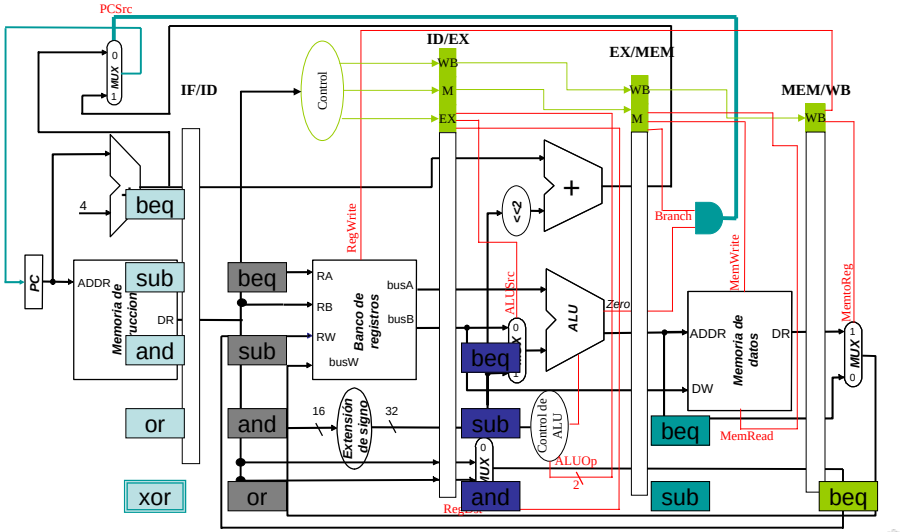
Dependencias de control/salto

Ejemplo 5

beq r1, r2, 4
sub r11, r1, r5
and r6, r1, r7
or r8, r1, r9
sw 0(r7), r8
xor r10, r1, r11



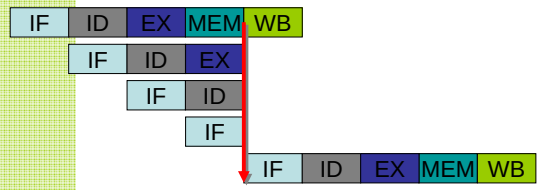
Dependencias de control/salto



Dependencias de control/salto

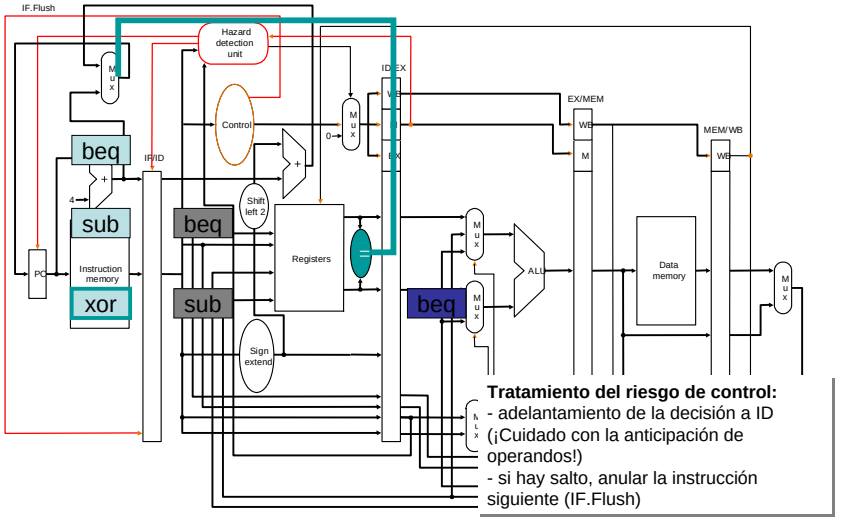
Ejemplo 5

beq r1, r2, 4
sub r11, r1, r5
and r6, r1, r7
or r8, r1, r9
sw 0(r7), r8
xor r10, r1, r11



Solo necesitamos limpiar los registros de segmentación
No se ha producido ningún write back

Dependencias de control/salto

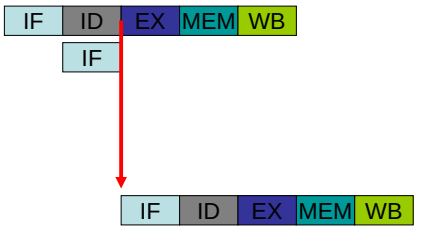


Tratamiento del riesgo de control:
- adelantamiento de la decisión a ID (¡Cuidado con la anticipación de operandos!)
- si hay salto, anular la instrucción siguiente (IF.Flush)

Dependencias de control/salto

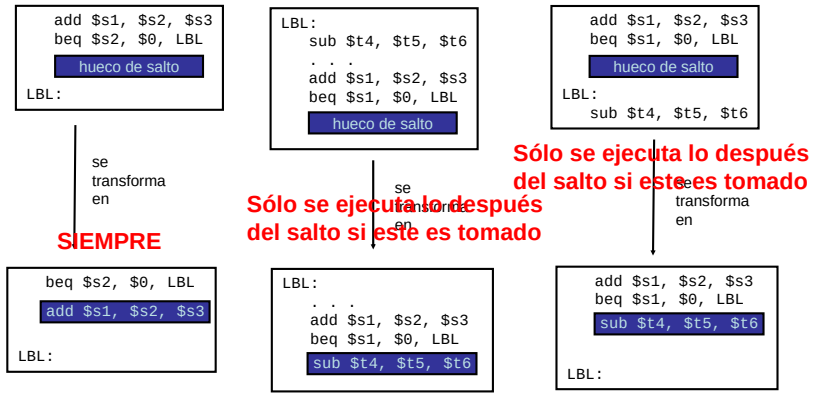
Ejemplo 5

```
beq r1, r2, 4
sub r11, r1, r5
and r6, r1, r7
or r8, r1, r9
sw 0(r7), r8
xor r10, r1, r11
```



Pasamos de nueve ciclos a 7
¿Se puede mejorar?

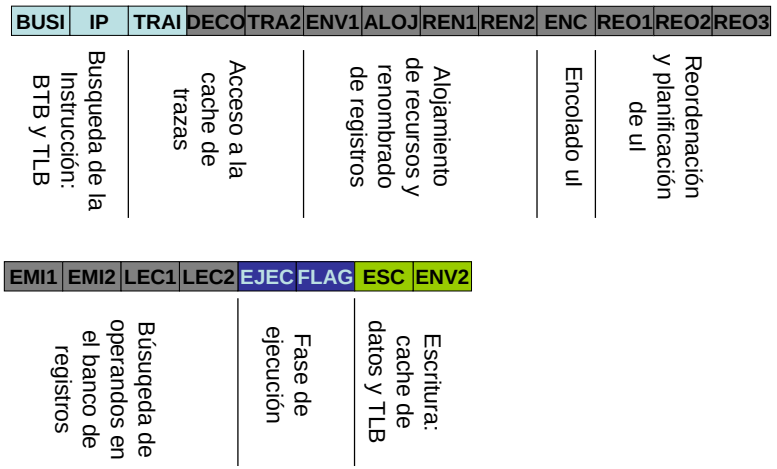
Gestión del hueco de salto retardado



Un salto retardado con un *pipeline* de un hueco siempre ejecuta la instrucción que hay después del salto, mientras que la segunda siguiente ya será dependiente del salto.
Es labor del compilador rellenar ese hueco con una instrucción válida y útil. Si no fuera posible encontrarla, se incluiría una NOP

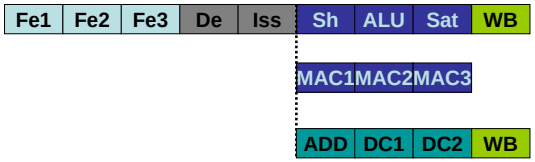
Segmentación Intel

Ejemplo para Pentium IV Prescott



Segmentación ARM

Ejemplo para el ARM1156T2F-S



Fe1: Se manda la dirección a memoria
Fe2: Se devuelve la instrucción
Fe3: Predicción de salto
De: Decodificación
Iss: Lectura de registros
Sh: Desplazamiento
ALU: Cálculo de la operación
Sat: Etapa del ‘pipe’ que permite la saturación de resultados en enteros
Wbex: Write back