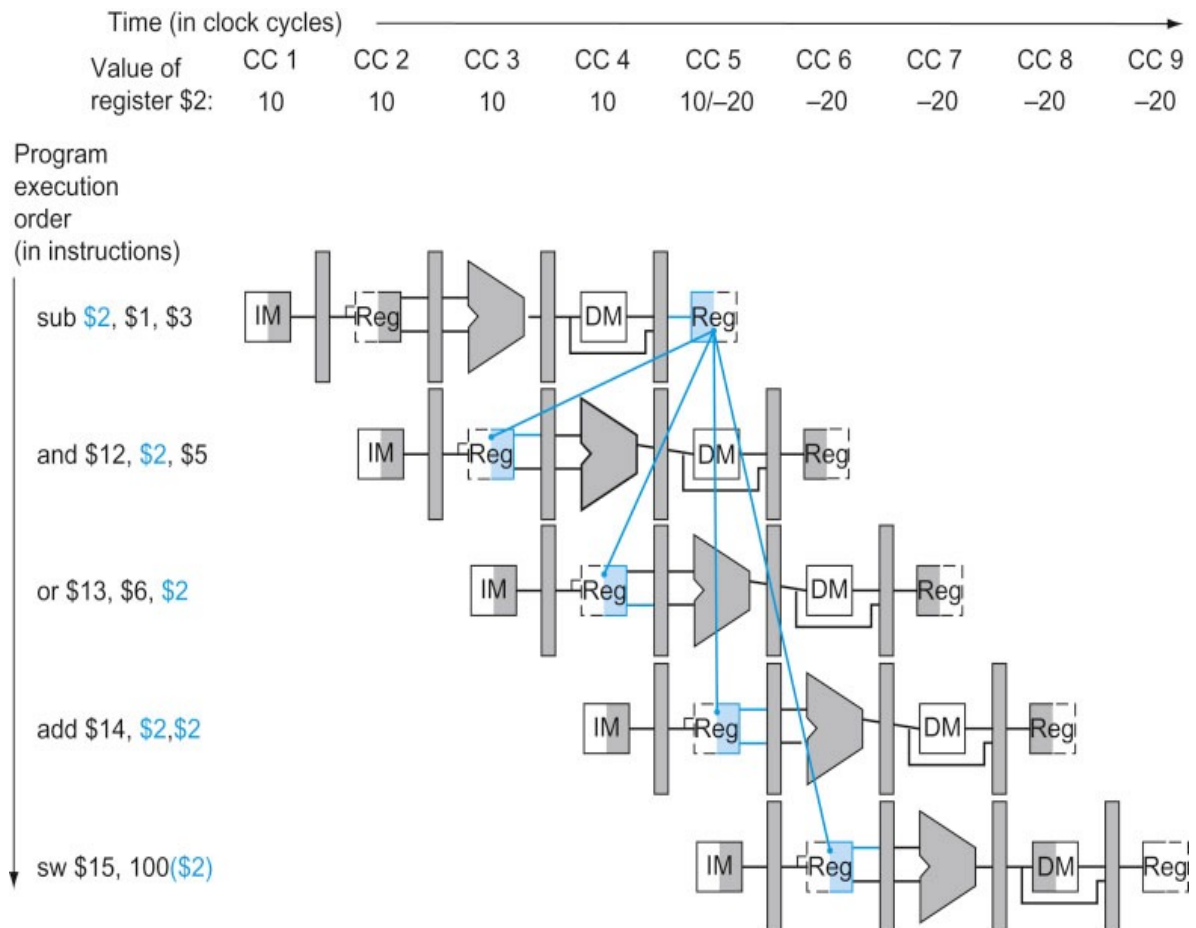
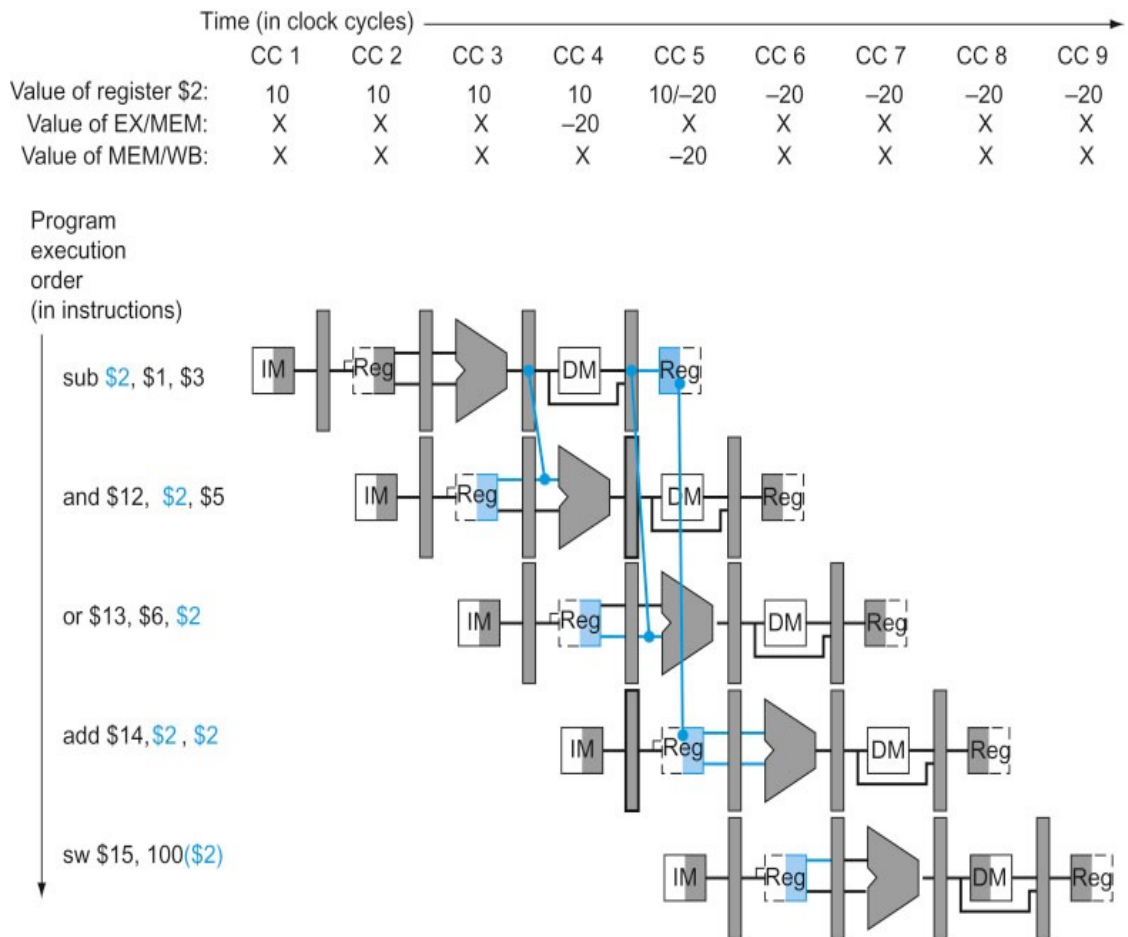


Dependencias de datos

sub \$2, \$1, \$3
 and \$12, \$2, \$5
 or \$13, \$6, \$2
 add \$14, \$2, \$2
 sw \$15, 100(\$2)



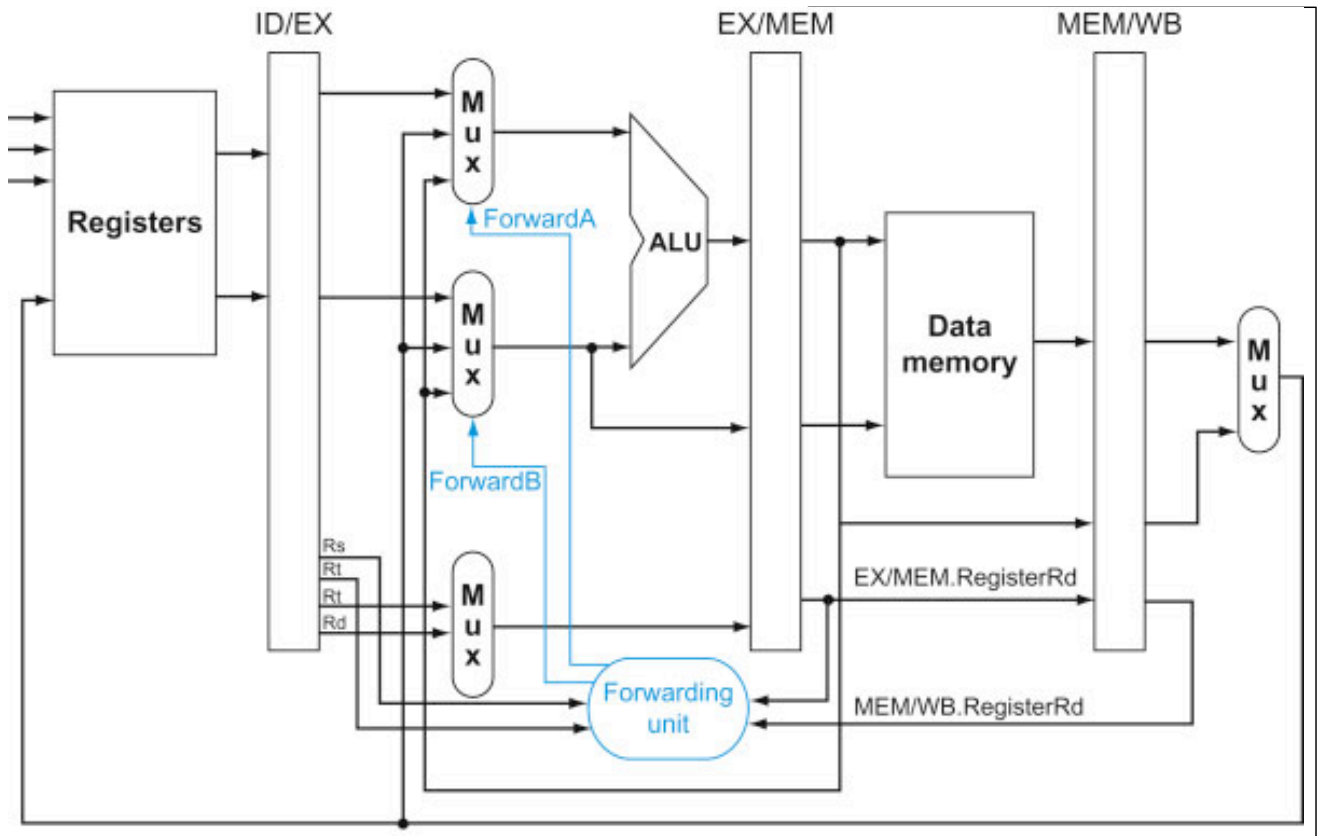


7/9/16

Guillermo Aguirre

14

Nuevo hardware para adelantamiento



7/9/16

Guillermo Aguirre

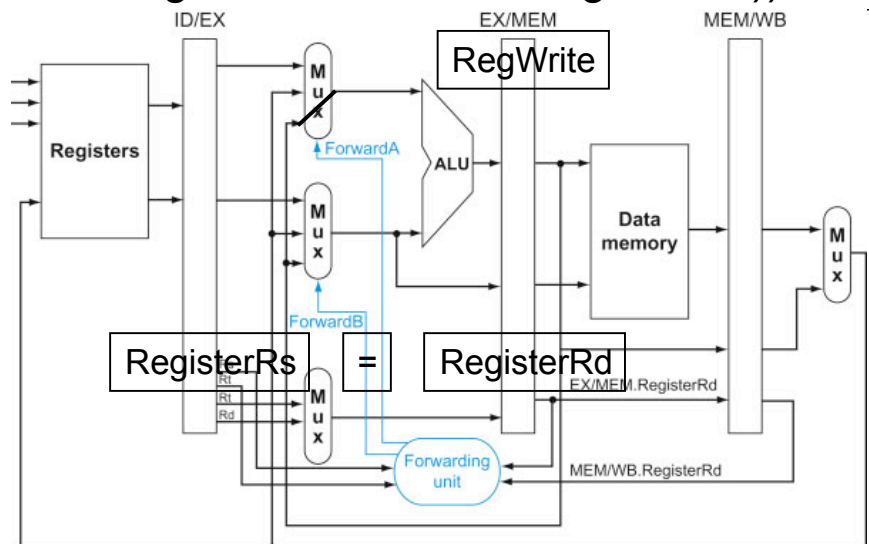
15

Control de los multiplexores para adelantamiento

Mux control	Source	Explanation
ForwardA = 00	ID/EX	The first ALU operand comes from the register file.
ForwardA = 10	EX/MEM	The first ALU operand is forwarded from the prior ALU result.
ForwardA = 01	MEM/WB	The first ALU operand is forwarded from data memory or an earlier ALU result.
ForwardB = 00	ID/EX	The second ALU operand comes from the register file.
ForwardB = 10	EX/MEM	The second ALU operand is forwarded from the prior ALU result.
ForwardB = 01	MEM/WB	The second ALU operand is forwarded from data memory or an earlier ALU result.

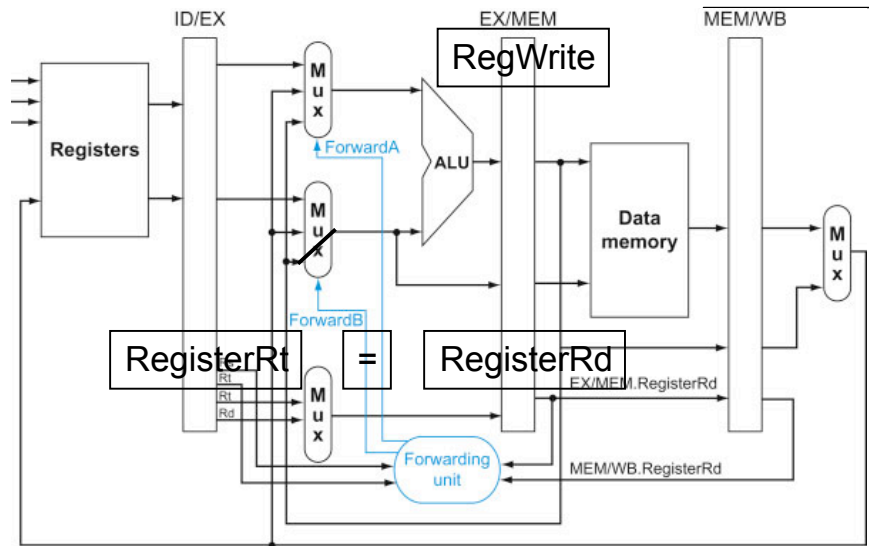
Condiciones para detectar riesgos. Señales de control. Riesgo con EX

if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd \neq 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRs)) ForwardA=10



Condiciones para detectar riesgos. Señales de control. Riesgo con EX

if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd \neq 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRt)) ForwardB=10



7/9/16

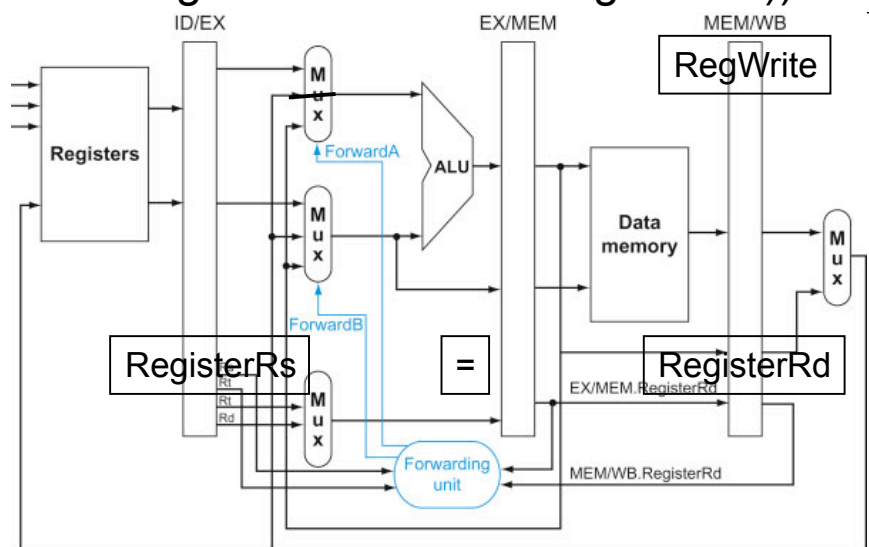


Guillermo Aguirre

18

Condiciones para detectar riesgos. Señales de control. Riesgo con MEM

if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd \neq 0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRs)) ForwardA=01



7/9/16

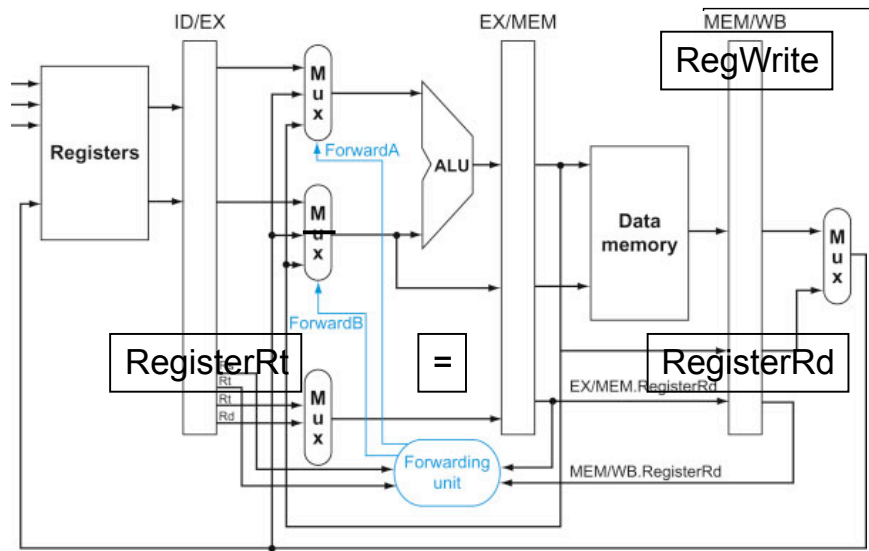


Guillermo Aguirre

19

Condiciones para detectar riesgos. Señales de control. Riesgo con MEM

if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd \neq 0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRt)) ForwardB=01



7/9/16



Guillermo Aguirre

20

Conflicto: sumar un vector de números

if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd \neq 0)
and not(EX/MEM.RegWrite and (EX/MEM.RegisterRd \neq 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRs))
and (MEM/WB.RegisterRd = ID/EX.RegisterRs)) ForwardA=01

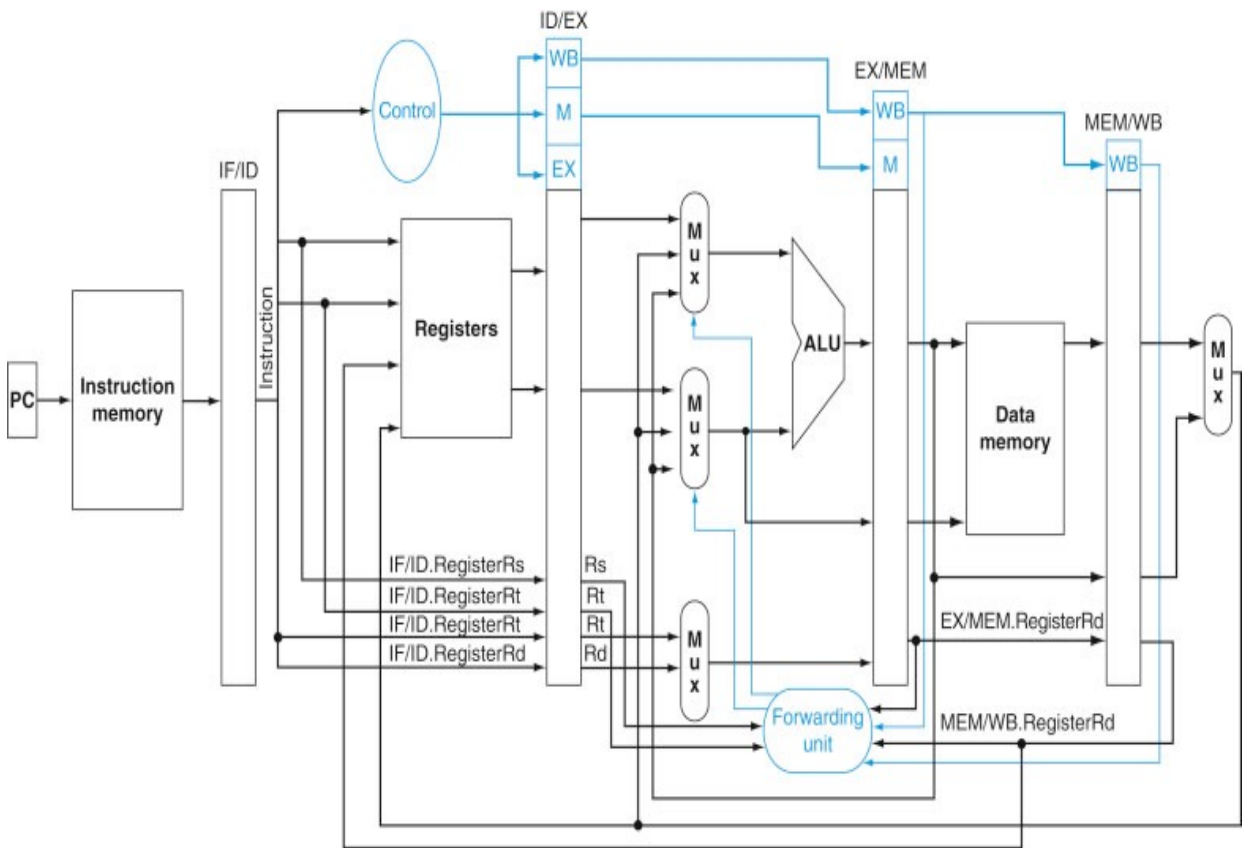
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd \neq 0)
and not(EX/MEM.RegWrite and (EX/MEM.RegisterRd \neq 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRt))
and (MEM/WB.RegisterRd = ID/EX.RegisterRt)) ForwardB=01

7/9/16

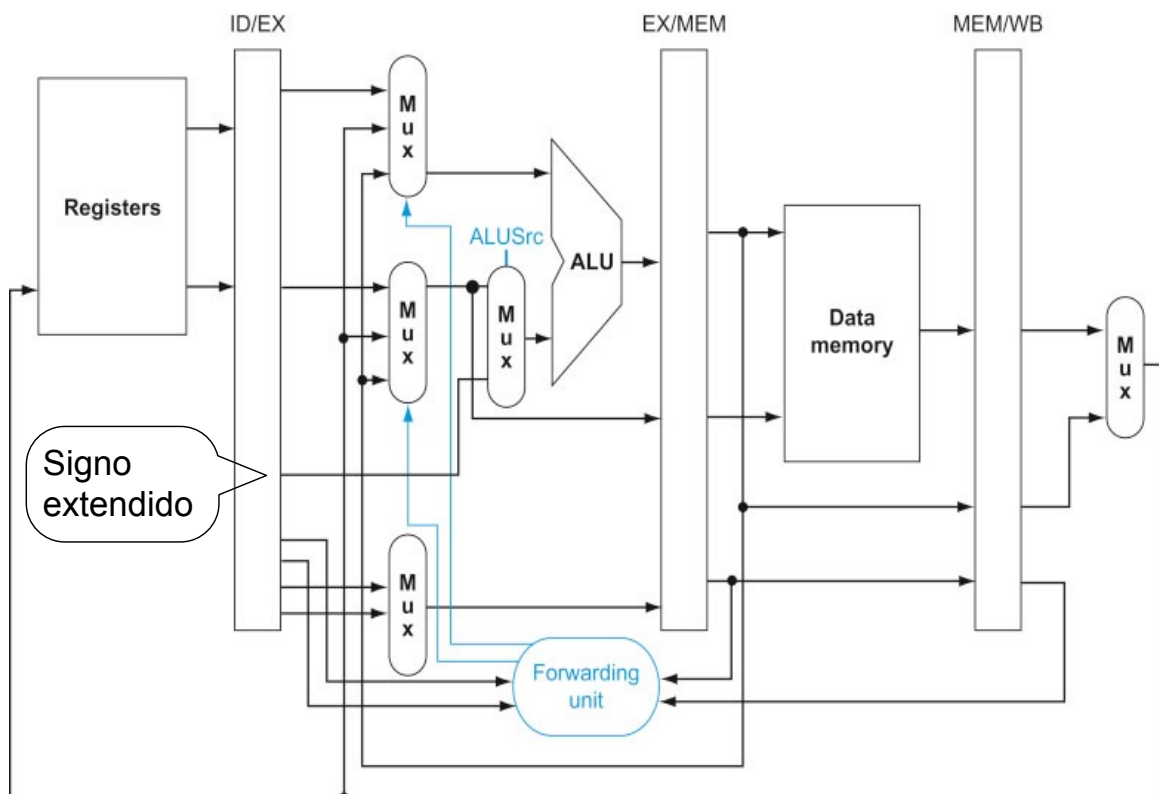


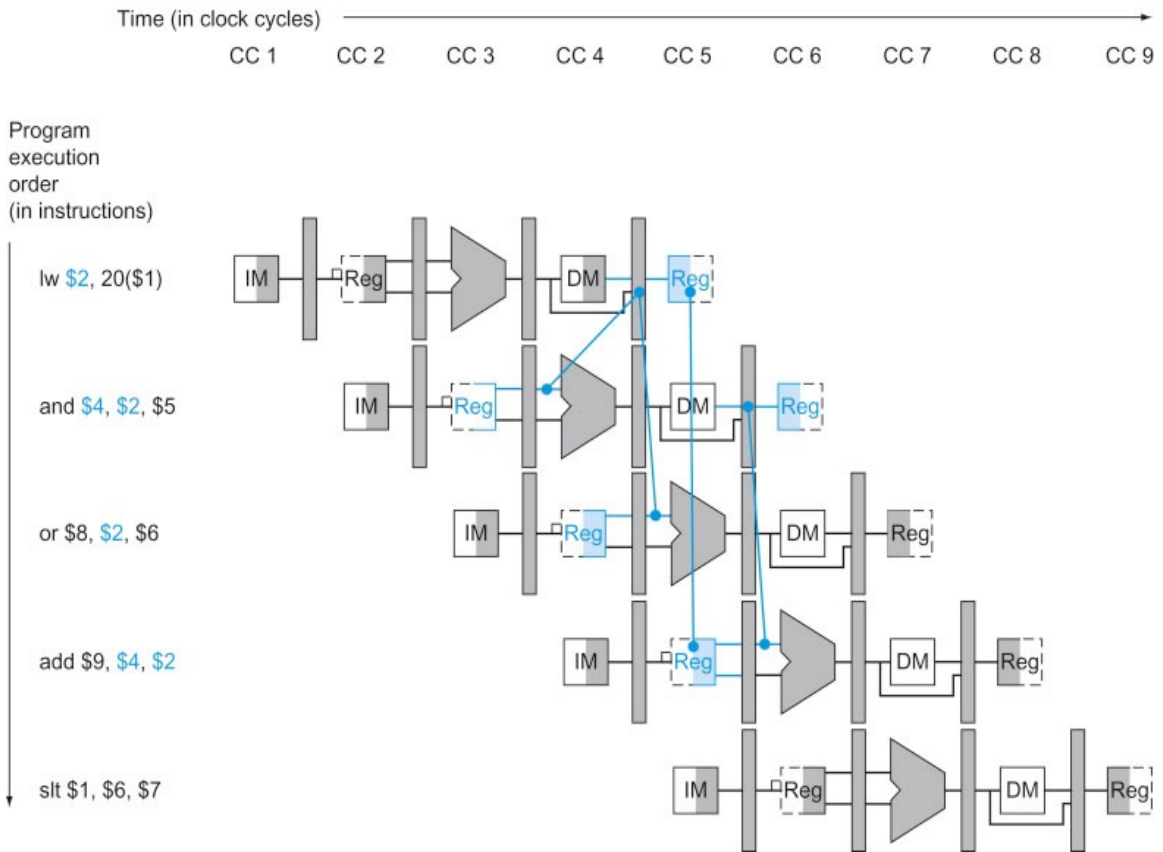
Guillermo Aguirre

21



Detalle de opciones en la entrada de la ALU





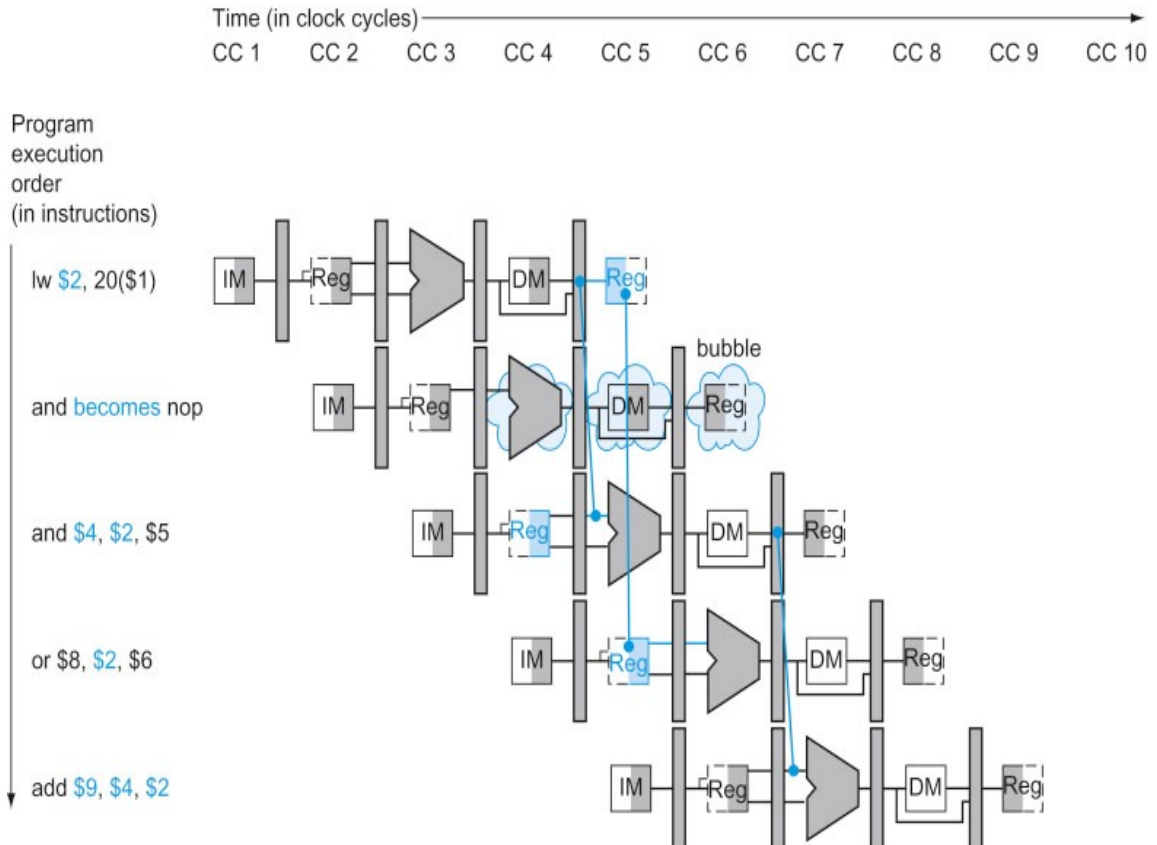
7/9/16



Guillermo Aguirre

24

Riesgo load-use



7/9/16



Guillermo Aguirre

25

Detección de riesgo load-use

if (ID/EX.MemRead and
((ID/EX.RegisterRt = IF/ID.RegisterRs) or
(ID/EX.RegisterRt = IF/ID.RegisterRt)))
atasco

No se modifican ni PC ni el registro IF/ID

Se crea una burbuja en EX. Una nop

Una nop son todos ceros

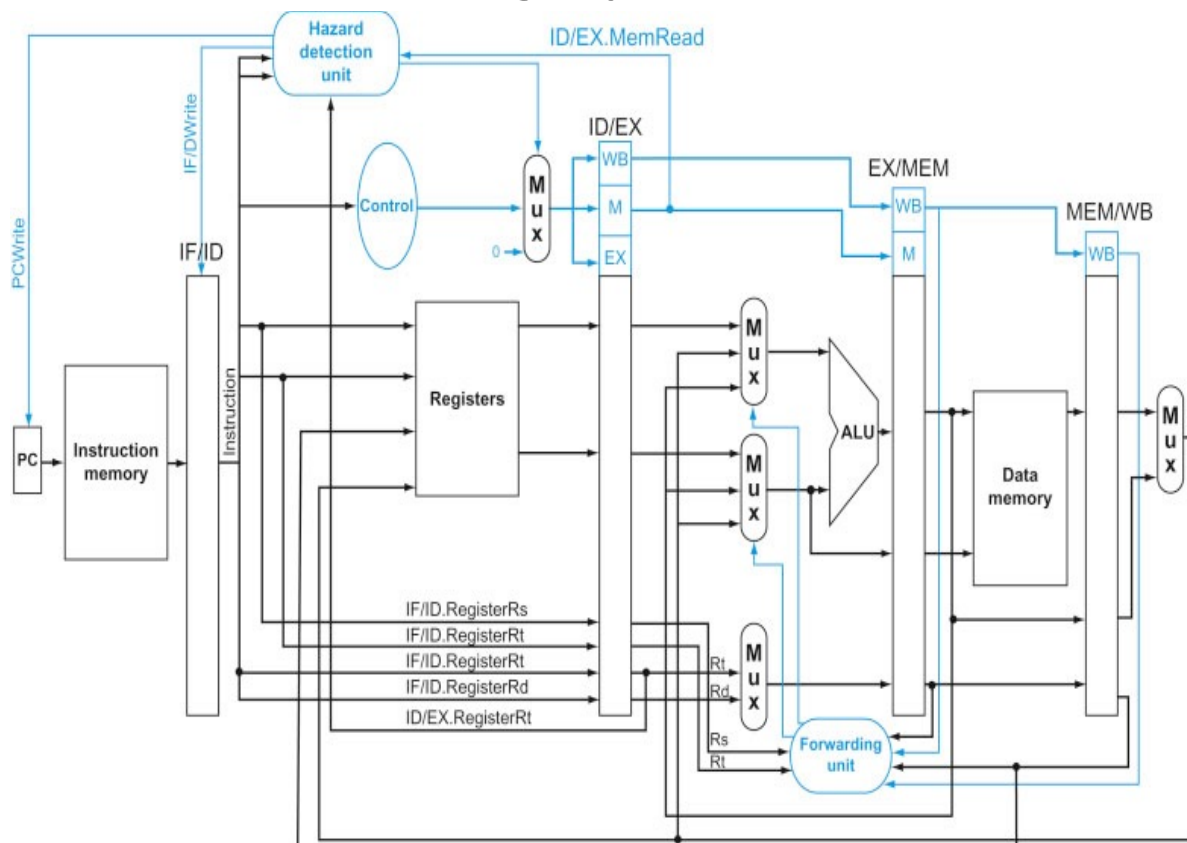
Las instrucciones posteriores se demoran 1 ciclo

7/9/16

 Guillermo Aguirre

26

Unidades de riesgo y de adelantamiento

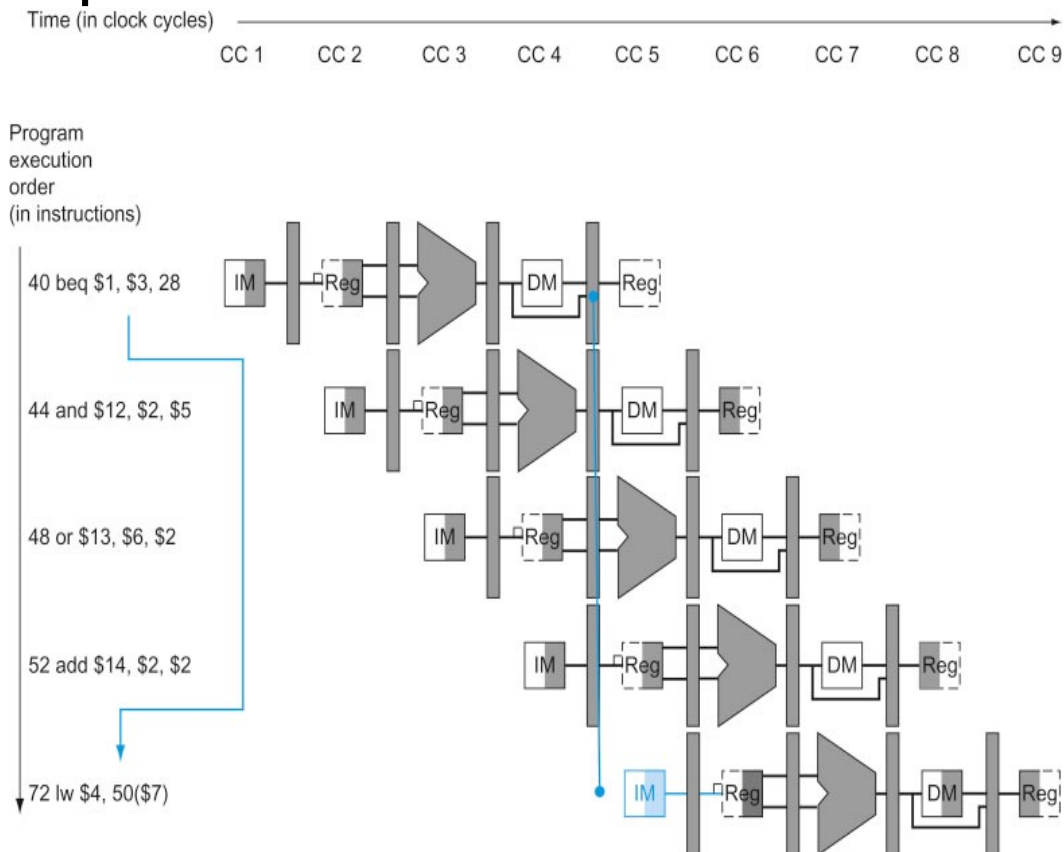


7/9/16

 Guillermo Aguirre

27

Impacto de los saltos condicionales



7/9/16



Guillermo Aguirre

28

Riesgos de control

Atascar hasta completar el salto es ineficiente.

Predecir como salto no tomado:

Acierto: sin costo

Fallo: descartar (flush) instrucciones

Reduciendo las demoras por saltos.

Adelantar la ejecución elimina menos instrucciones.

Colocar el sumador para saltos en ID.

Una simple comparación no requiere una ALU.

Se requiere adelantamiento.

7/9/16



Guillermo Aguirre

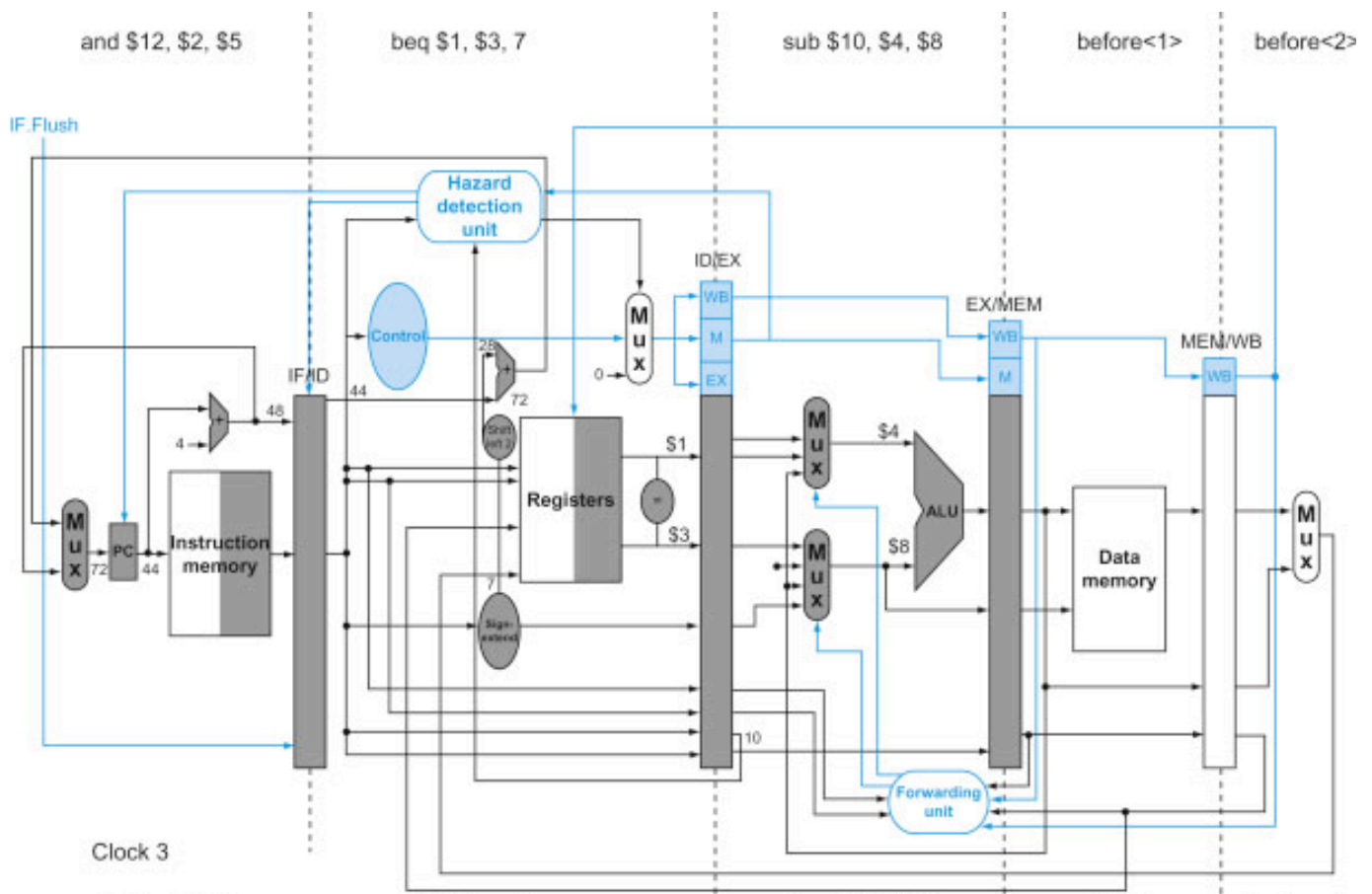
29

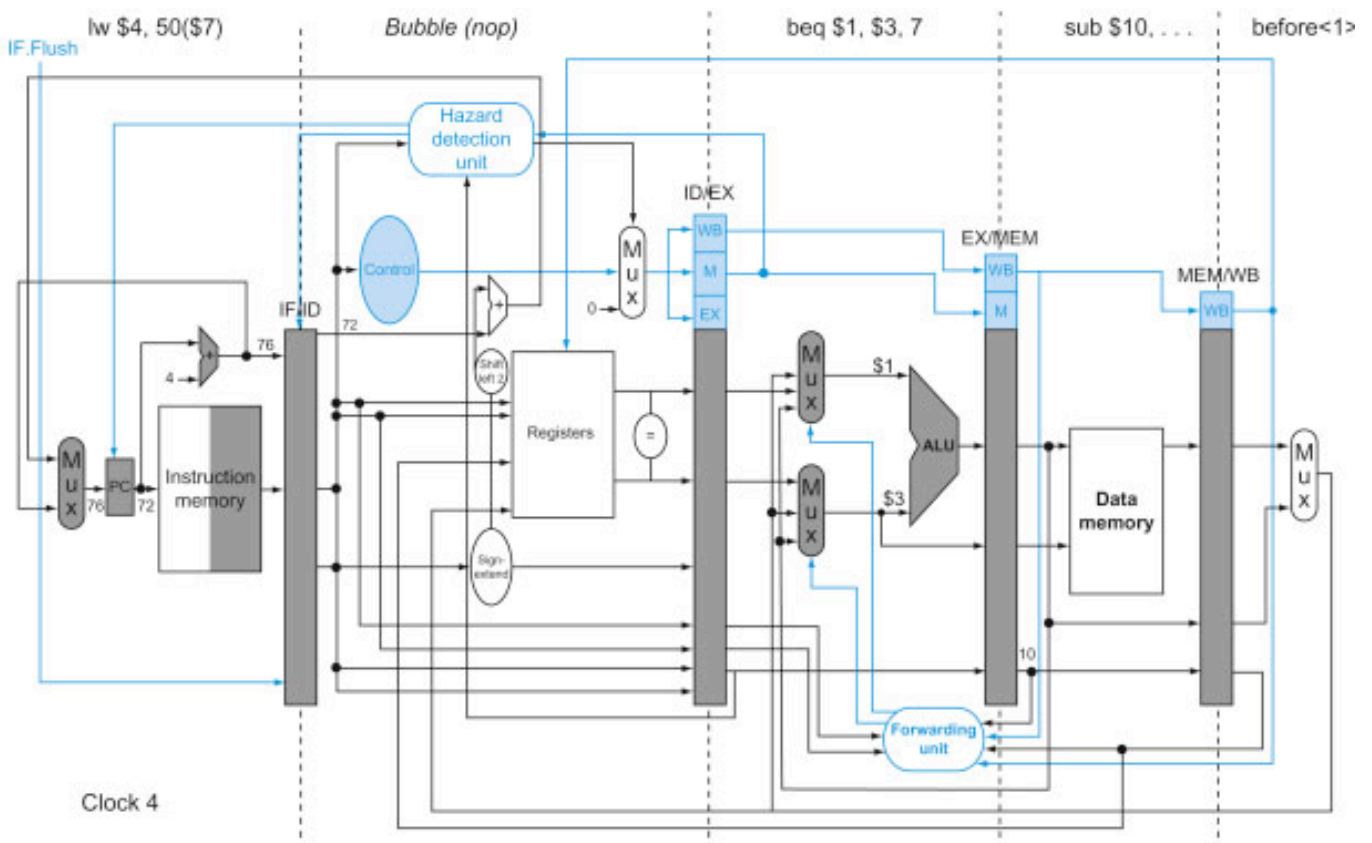
Optimización del salto tomado

```

36  sub $10, $4, $8
40  beq $1, $3, 7
44  and $12, $2, $5
48  or  $13, $2, $6
52  add $14, $4, $7
...
72  lw  $4, 50($7)
    
```


Salto relativo al PC a 72 ($40+4+7*4$)





Clock 4

7/9/16

 Guillermo Aguirre

32

¿Qué vimos?

Dependencia de datos

Adelantamiento: circuitos y controles

Riesgo load-use

Riesgos de control

Salto no tomado

Penalidad

7/9/16

 Guillermo Aguirre

33