Arquitectura de Computadores

Tema 3
Segmentación (Pipelining)

Índice

Pipelining

- Técnica de implementación por la que varias instrucciones se superponen en ejecución
- Se aprovecha el paralelismo inherente en la instrucción
- Es la técnica más importante en la actualidad para mejorar el rendimiento de un procesador
- Consiste en dividir la ruta de datos en etapas
  - Operan en paralelo
  - Cada etapa ejecuta una instrucción
    - Pipe stage
    - Pipe segment
    - Etapa
    - Segmento

Rendimiento de un pipe-line

- ¿Cuanto tarda una instrucción en recorrer el pipe?
  - N° de Etapas
  - Ciclo del procesador
- Ciclo del procesador
  - Tiempo que tarda una instrucción en pasar de una etapa a otra
    - Normalmente un ciclo
    - A veces dos ciclos
    - Raramente + de 2 ciclos
      - (Sin operaciones segmentadas)
- El diseñador debe buscar el equilibrio entre la longitud de las etapas
- Rendimiento Ideal
  - Tiempo por instrucción (pipeline)=Tiempo por instrucción/(N° Etapas pipeline)

Ventajas

- Reducción de CPI
  - Se pasa por las mismas etapas
  - La ejecución de la siguiente instrucción comienza antes
  - No ideal
    - Paradas
    - Riesgos
- Reducción del ciclo de Reloj
  - Equilibrio en la duración
  - Mayor posibilidad de optimización de cada etapa
- No es visible al programador

Instrucciones RISC

- Características básicas de un ISA RISC
  - Todas las operaciones de datos operan entre registros
  - Reducción de las operaciones intermedias por no acceder a la memoria
  - Solo Load/Store acceden a memoria
  - Pocas instrucciones
  - Pocos formatos
- MIPS64
  - 32 Registros de propósito general
    - R0 = 0
  - 3 clases de instrucciones
    - ALU
    - Load / Store
    - Branch and Jump

Fases de la Instrucción

1. IF
   - Instruction Fetch: Búsqueda de la Instrucción
     - (PC)→Adress Mem→(Mem (PC))→RI
2. ID
   - Intruction Decoding: Decodificación de la Instrucción
     - Lectura de Registros
3. EX
   - Execution: Ejecución de la instrucción
     - ALU
     - Cálculo de la dirección efectiva
     - Operaciones Reg – Reg
     - Opereciones Reg – Inmediato
4. MEM
   - Memory Access: Acceso de Memoria
5. WB
   - Write Back : Escritura
     - Escritura de resultados

CPI

- CPI: Ciclos por Instrucción
  - Branch
    - 2 ciclos
    - Frecuencia 12 %
  - Store / Load
    - 4 ciclos
    - Frecuencia 10%
  - Resto
    - 5 ciclos
    - Frecuencia 78 %
- CPI = 4.5 = 2*0.12 + 4*0.10 + 5*0.78
  - Ciclos promedio por instrucción

Pipelining

- Idea: Comenzar una instrucción en cada ciclo

Nº Instr	Ciclo de Reloj								
	1	2	3	4	5	6	7	8	9
i	IF	ID	EX	MEM	WB				
i+1		IF	ID	EX	MEM	WB			
i+2			IF	ID	EX	MEM	WB		
i+3				IF	ID	EX	MEM	WB	
i+4					IF	ID	EX	MEM	WB

- Aspectos a considerar
  - Que pasa en cada ciclo
  - Está ocupado el camino de datos?
  - Está ocupada la parte que quiero usar?
  - Hay recursos?
  - Ocupación de los buses?
- Soluciones básicas
  - + Hardware
  - Modificaciones al control
  - Incremento de registros
    - Información del pipeline

Aspectos básicos del Rendimiento

- Pipe
  - Mejora el Rendimiento
    - Reduce el tiempo de ejecución total
    - Tiempo de cpu de cada instrucción no se modifica
  - Problemas
    - Desequilibrio entre etapas e instrucciones
    - Clock Skew
    - Sobrecarga
- Ejemplo

	sin pipe	con pipe
Tciclo	1 ns	1.2 ns
CPI	4.5	1 (ideal)
Speed-up	4.5/1.2 = 3.75	

Índice

1. Segmentación
2. Riesgos (Hazards)
3. Forwarding
4. Implementación del pipelining
5. Tratamiento de las interrupciones
6. Complicaciones del Repertorio de Instrucciones
7. Extensiones para Multiciclo

Riesgos (Hazards)

- El mayor problema de la segmentación
- 3 tipos
  - Estructurales
    - Conflicto de Recursos
  - Datos
    - Dependencia con instrucciones anteriores
  - Control
    - Decisión en los saltos y bifurcaciones
- Hazard:
  - Situación que impide la ejecución de la siguiente instrucción
  - Provoca una parada en el pipe
    - Se ejecuta lo que quedaba
    - Se elimina lo posterior
      - Degrada el Rendimiento

Degradación del Rendimiento

- Rdto del Pipe con Paradas (Stalls)

$$Speed-Up = \frac{\overline{TPI_{\text{sin pipe}}}}{\overline{TPI_{\text{con pipe}}}} = \frac{\overline{TPI}}{\overline{TPI_p}} = \frac{CPI \times T_{ciclo}}{CPI_p \times T_{ciclo_p}}$$

$$CPI_p = CPI_{ideal} + CPPI$$

$$CPI_{ideal} = 1$$

$CPPI$ = Ciclos de Parada del Pipe por Instrucción

- Suponiendo Equilibrio Completo y sobrecarga nula

$$Speed-Up = \frac{CPI}{1+CPPI}$$

- Si CPI = n° de etapas del pipe (segmentos) para todas las instrucciones

$$Speed-Up = \frac{N^{o}\,Segmentos}{1+CPPI}$$

Degradación del Rendimiento

$$Speed-Up = \frac{CPI}{CPI_p} \times \frac{T_{ciclo}}{T_{ciclo_p}} = \frac{1}{1+CPPI} \times \frac{T_{ciclo}}{T_{ciclo_p}}$$

$$Speed-Up = \frac{1}{1+CPPI} \times N^{o}\,Etapas_{pipe}$$

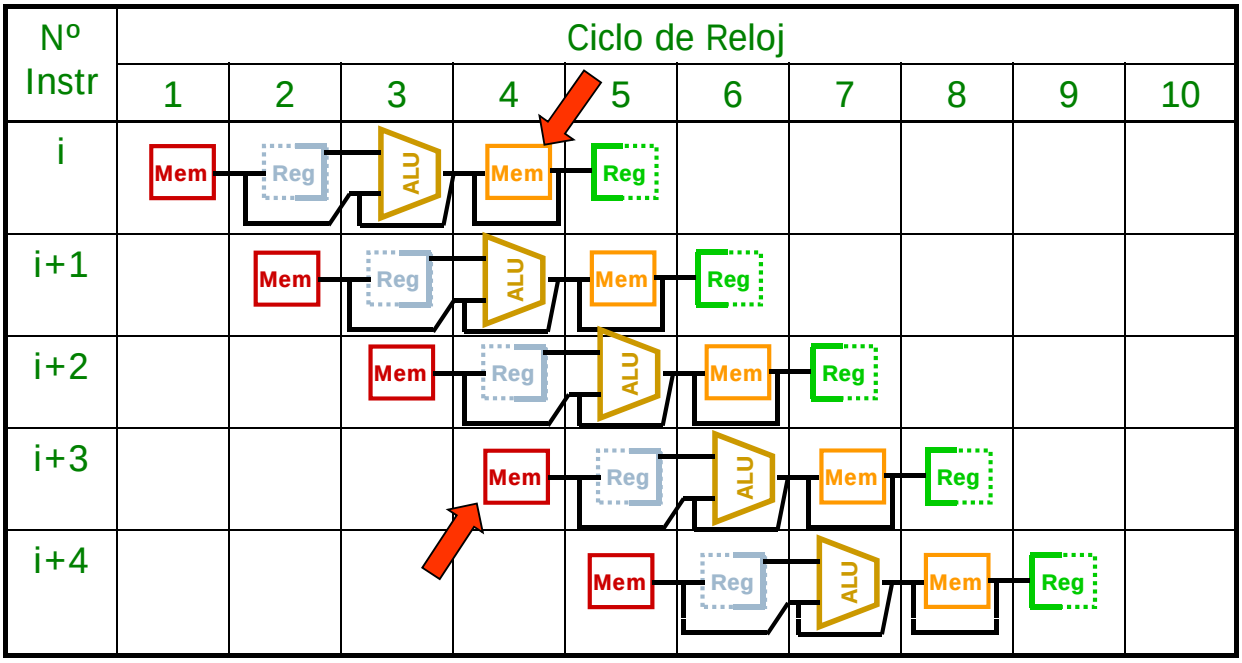
Hazars estructurales

- Una combinación de instrucciones no se puede ejecutar por un conflicto de recursos Hardware
  - Implica la parada de la instrucción
  - Se crea una "burbuja"
    - Se crea un hueco en el pipe STOP
    - Ejemplo con memoria única de datos e instrucciones

Nº Instr	Ciclo de Reloj									
	1	2	3	4	5	6	7	8	9	10
i	IF	ID	EX	MEM	WB					
i+1		IF	ID	EX	MEM	WB				
i+2			IF	ID	EX	MEM	WB			
i+3				STOP	STOP	STOP	IF	ID	EX	MEM
i+4										

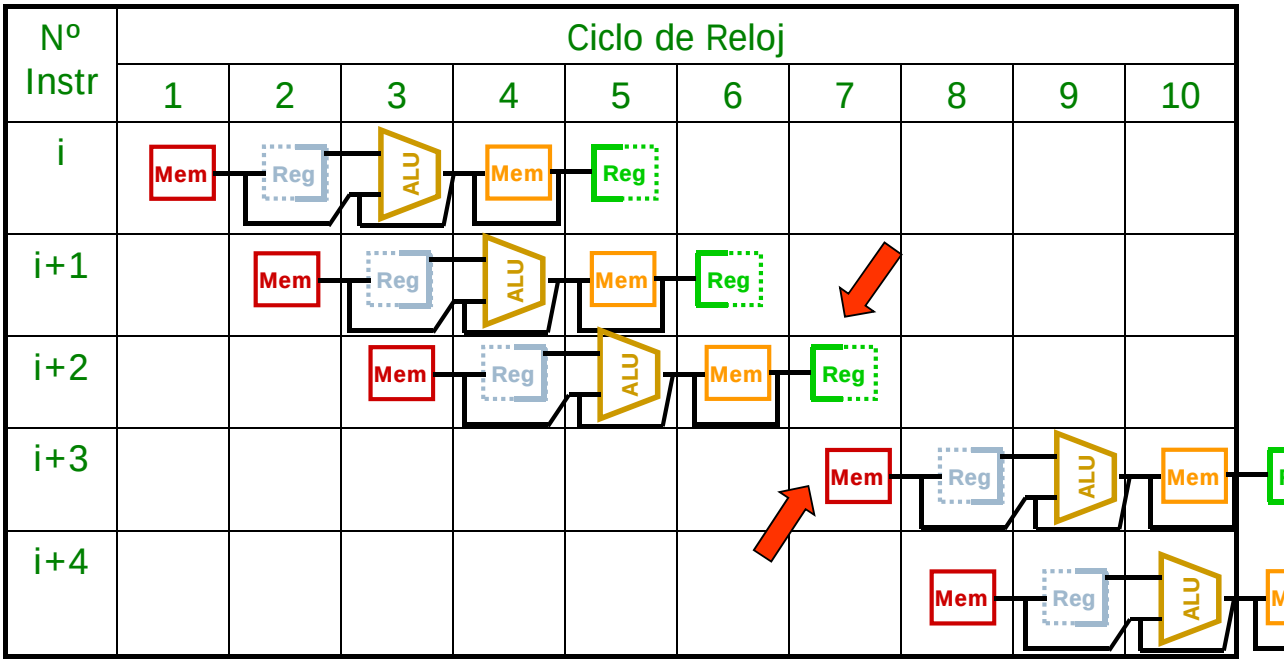
Riesgos estructurales

- Ejemplo Memoria única

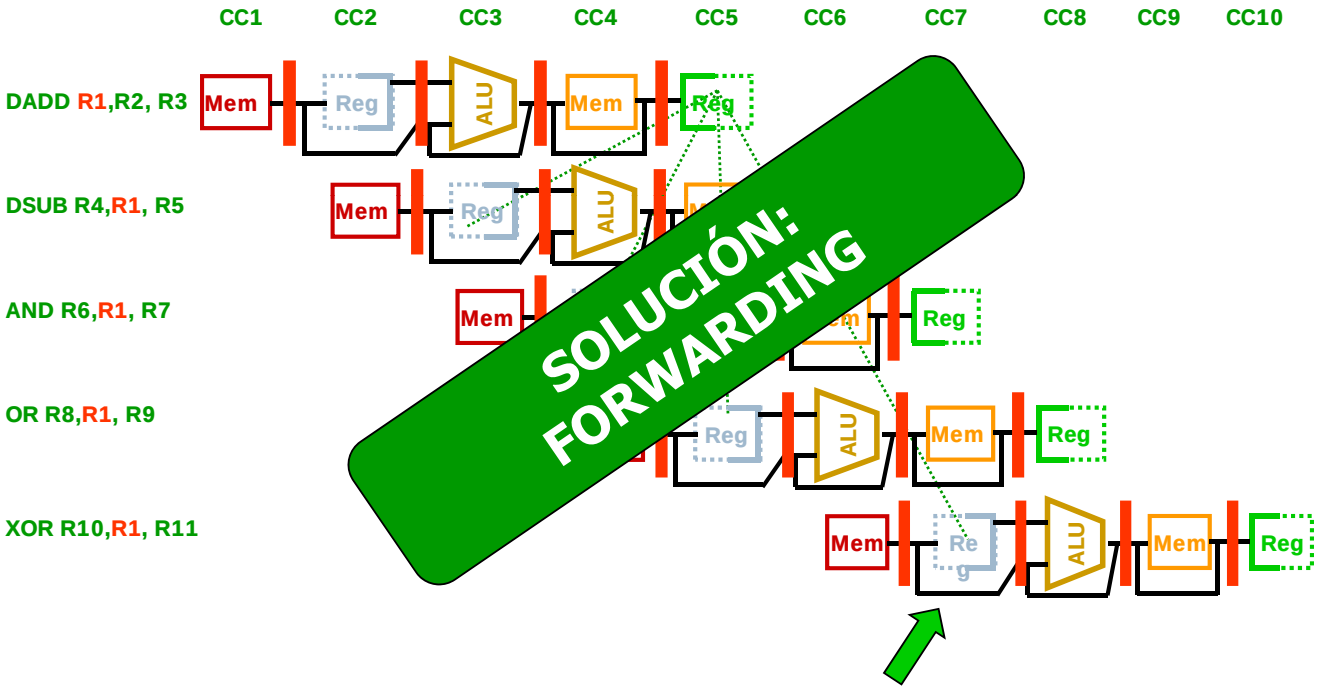


Riesgos estructurales

- Ejemplo Memoria única



Hazards de datos



Hazards de Control

- Si un salto cambia el PC
  - Salto tomado (taken branch)
    - PC no cambia hasta después de la fase de ID
    - Solución Básica: Repetir IF de la instrucción siguiente al BRANCH
    - Es una parada pero evita la propagación

Nº Instr	Ciclo de Reloj									
	1	2	3	4	5	6	7	8	9	10
i	IF	ID	EX	MEM	WB					
BRANCH		IF	ID	EX	MEM	WB				
i+2			IF	IF	ID	EX	MEM	WB		
i+3					IF	ID	EX	MEM	WB	
i+4						IF	ID	EX	MEM	WB

Reducción de las penalizaciones por salto

- Soluciones simples en tiempo de compilación
  1. FLUSH or FREEZE
     - Repetir o borrar cualquier instrucción después del salto hasta conocer el destino del mismo
       - Simple a nivel HW y SW
  2. ALL NOT TAKEN
     - Tratar todos los saltos como no tomados
     - Se continúa la ejecución y si se toma se ignoran las fases correspondientes
       - Algo más complejo

ALL NOT TAKEN

Nº Instr	Ciclo de Reloj								
	1	2	3	4	5	6	7	8	9
Salto no Tomado (i)	IF	ID	EX	MEM	WB				
i+1		IF	ID	EX	MEM	WB			
i+2			IF	ID	EX	MEM	WB		
i+3				IF	ID	EX	MEM	WB	
i+4					IF	ID	EX	MEM	WB

Nº Instr	Ciclo de Reloj								
	1	2	3	4	5	6	7	8	9
Salto Tomado (i)	IF	ID	EX	MEM	WB				
i+1		IF	IDLE	IDLE	IDLE	IDLE			
Branch Target			IF	ID	EX	MEM	WB		
Branch Target + 1				IF	ID	EX	MEM	WB	
Branch Target + 2					IF	ID	EX	MEM	WB

Reducción de las penalizaciones por salto

3. ALL TAKEN
   - Tratar todos los saltos como tomados
   - No tiene sentido para este pipeline
4. BRANCH DELAY SLOT
   - Se retarda una instrucción el salto
   - El compilador debe buscar que el sucesor secuencial sea siempre válido

Branch instruction
Sucesor secuencial I
Objetivo si el salto es tomado

Salto Retardado

Nº Instr	Ciclo de Reloj								
	1	2	3	4	5	6	7	8	9
Salto no Tomado (i)	IF	ID	EX	MEM	WB				
Branch Delay (i+1)		IF	ID	EX	MEM	WB			
i+2			IF	ID	EX	MEM	WB		
i+3				IF	ID	EX	MEM	WB	
i+4					IF	ID	EX	MEM	WB

Nº Instr	Ciclo de Reloj								
	1	2	3	4	5	6	7	8	9
Salto Tomado (i)	IF	ID	EX	MEM	WB				
Branch Delay (i+1)		IF	ID	EX	MEM	WB			
Branch Target			IF	ID	EX	MEM	WB		
Branch Target + 1				IF	ID	EX	MEM	WB	
Branch Target + 2					IF	ID	EX	MEM	WB

Efectos en el Rendimiento

- Aspectos a considerar
  - Frecuencia de los Saltos
  - Penalización por saltos
  - Tipo de saltos
    - Condicionales
    - No condicionales
      - Los más frecuentes

$$Speed - Up_{pipeline} = \frac{\text{Nº Etapas}}{1 + \text{Ciclos de parada por saltos}}$$

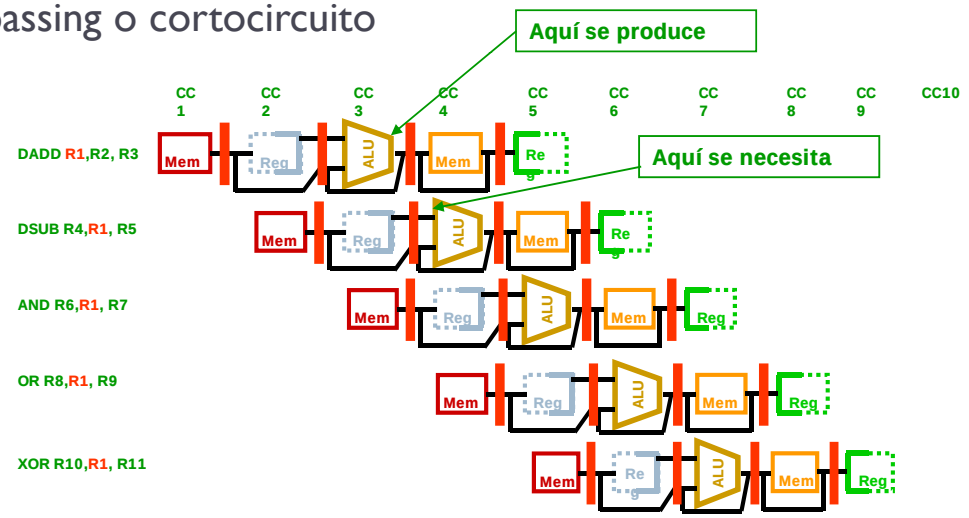
$$= \frac{\text{Nº Etapas}}{1 + frec_{saltos} \times Penal_{saltos}}$$

Índice

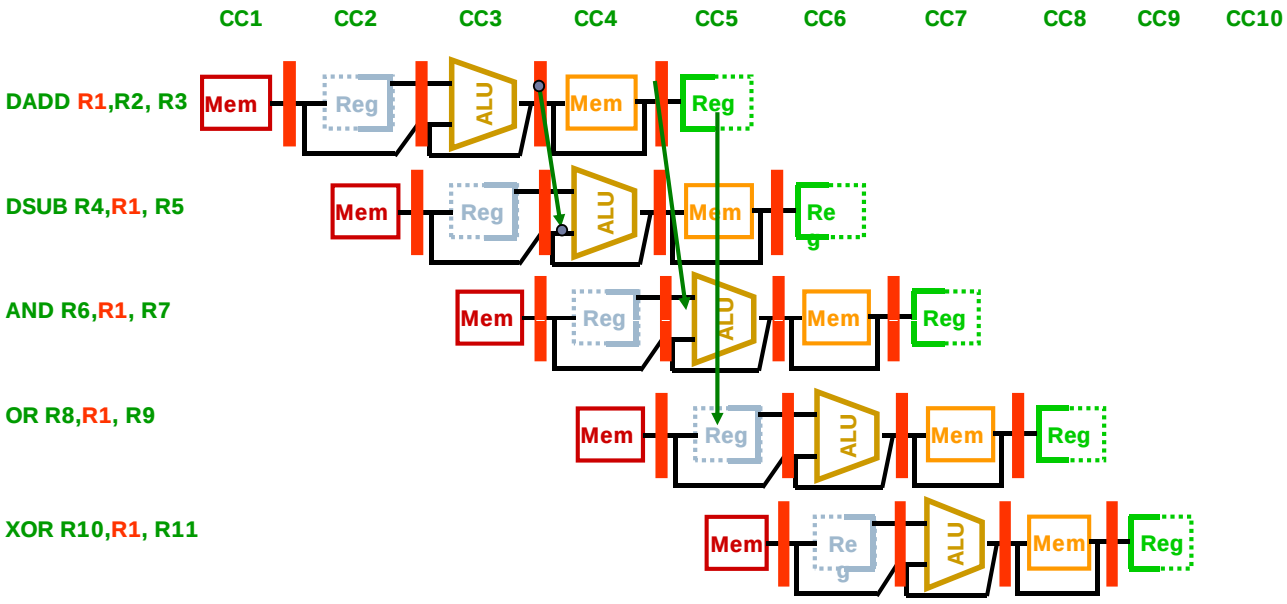
1. Segmentación
2. Riesgos (Hazards)
3. Forwarding
4. Implementación del pipelining
5. Tratamiento de las interrupciones
6. Complicaciones del Repertorio de Instrucciones
7. Extensiones para Multiciclo

Forwarding

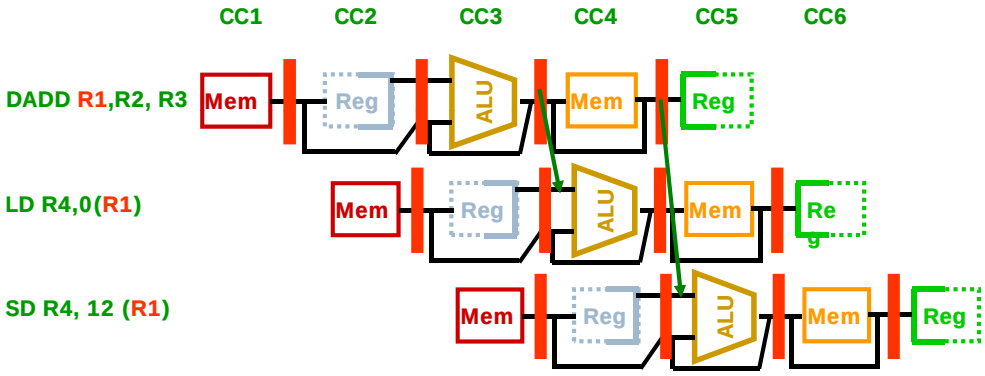
- El resultado estás disponible después de EXE
  - Podemos adelantar el resultado
  - Bypassing o cortocircuito



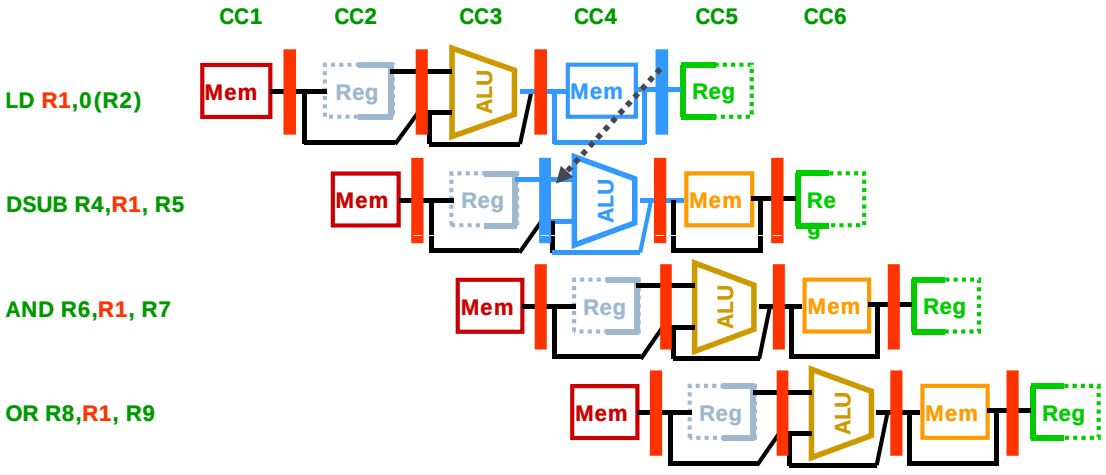
Forwarding



Forwarding



Algunos Riesgos no se solucionan

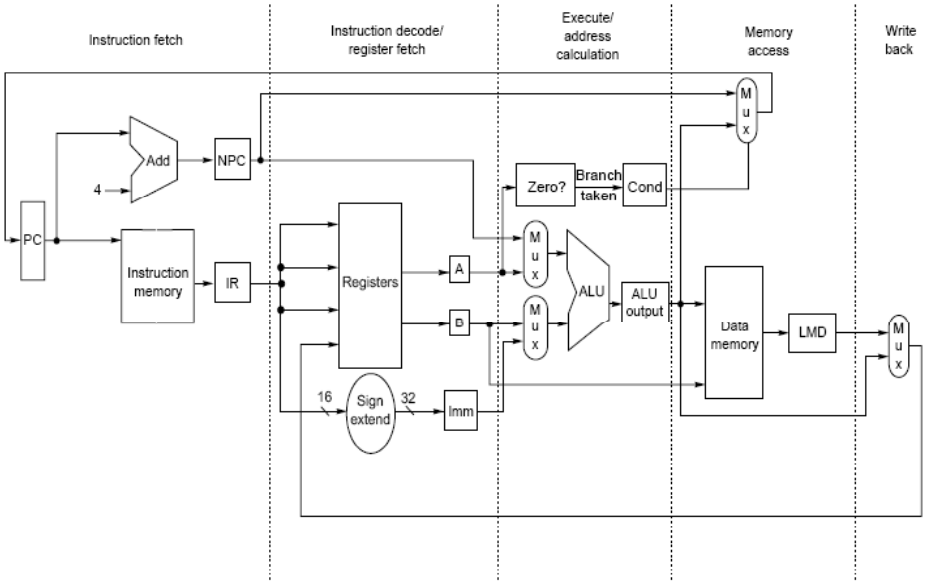


Índice

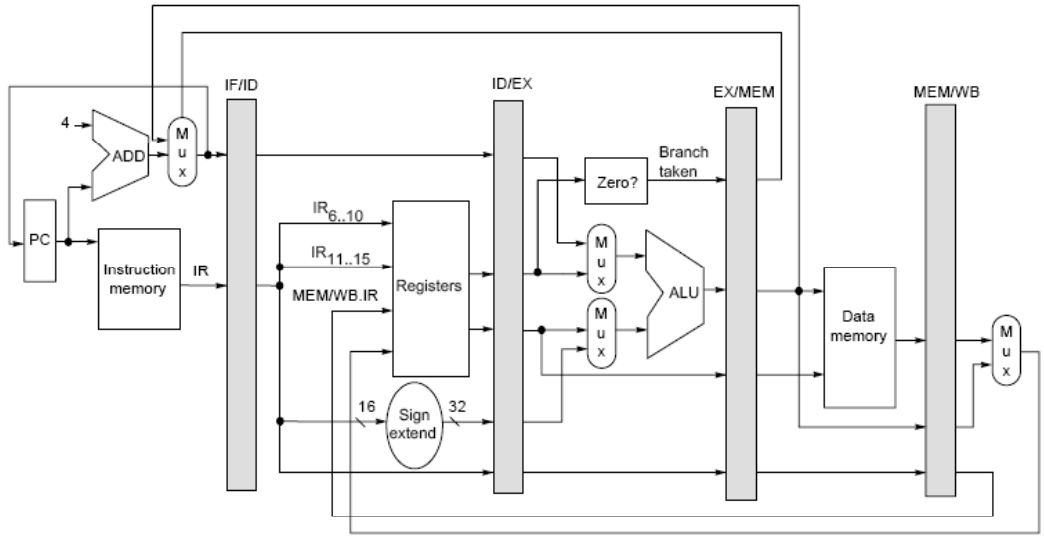
1. Segmentación
2. Riesgos (Hazards)
3. Forwarding
4. Implementación del pipelining
   1. Modificaciones al cauce no segmentado
   2. Implementación del control
   3. Implementación del forwarding
   4. Dificultades
5. Tratamiento de las interrupciones
6. Complicaciones del Repertorio de Instrucciones
7. Extensiones para Multiciclo

Implementación del Pipelining

- Se parte de la implementación del MIPS
  - Registros entre etapas



Implementación del Pipelining



Eventos en cada etapa del pipe

Stage	Any instruction		
IF	IF/ID.IR ← Mem[PC]; IF/ID.NPC,PC ← (if ((EX/MEM.opcode == branch) & EX/MEM.cond){EX/MEM. ALUOutput} else {PC+4});		
ID	ID/EX.A ← Regs[IF/ID.IR[rs]]; ID/EX.B ← Regs[IF/ID.IR[rt]]; ID/EX.NPC ← IF/ID.NPC; ID/EX.IR ← IF/ID.IR; ID/EX.Imm ← sign extend(IF/ID.IR[immediate field]);		
	ALU instruction	**Load or store instruction**	**Branch instruction**
EX	EX/MEM.IR ← ID/EX.IR; EX/MEM.ALUOutput← ID/EX.A *func* ID/EX.B; or EX/MEM.ALUOutput ← ID/EX.A *op* ID/EX.Imm; EX/MEM.cond ← 0;	EX/MEM.IR← ID/EX.IR EX/MEM.ALUOutput ← ID/EX.A + ID/EX.Imm; EX/MEM.cond ← 0; EX/MEM.B← ID/EX.B;	 EX/MEM.ALUOutput ← ID/EX.NPC+ID/EX.Imm; EX/MEM.cond ← (ID/EX.A == 0);
MEM	MEM/WB.IR ← EX/MEM.IR; MEM/WB.ALUOutput ← EX/MEM.ALUOutput;	MEM/WB.IR ← EX/MEM.IR; MEM/WB.LMD ← Mem[EX/MEM.ALUOutput]; or Mem[EX/MEM.ALUOutput] ← EX/MEM.B;	
WB	Regs[MEM/WB.IR[rd]] ← MEM/WB.ALUOutput; or Regs[MEM/WB.IR[rt]] ← MEM/WB.ALUOutput;	For load only: Regs[MEM/WB.IR[rt]] ← MEM/WB.LMD;	

Implementación del control

- Detección y resolución de riesgos
  - Comparación de registros

Situation	Example code sequence	Action

	LD **R1**,45(R2)
	ADDD R5,R6,R7
	SUBD R8,R6,R7
	OR R9,R6,R7	

	LD **R1**,45(R2)
	ADDD R5,**R1**,R7
	SUBD R8,R6,R7
	OR R9,R6,R7	

	LD **R1**,45(R2)
	ADDD R5,R6,R7
	SUBD R8,**R1**,R7
	OR R9,R6,R7	

	LD **R1**,45(R2)
	ADDD R5,R6,R7
	SUBD R8,R6,R7
	OR R9,**R1**,R7	

Implementación del control

- Detección y resolución de riesgos
  - Comparación de registros

Situation	Example code sequence	Action

No dependence	LD **R1**,45(R2)
	ADDD R5,R6,R7
	SUBD R8,R6,R7
	OR R9,R6,R7
		No hazard possible because no dependence exists on R1 in the immediately following three instructions.

Dependence requiring stall	LD **R1**,45(R2)
	ADDD R5,**R1**,R7
	SUBD R8,R6,R7
	OR R9,R6,R7
		Comparators detect the use of R1 in the ADD and stall the ADD (and SUB and OR) before the ADD begins EX.

Dependence overcome by forwarding	LD **R1**,45(R2)
	ADDD R5,R6,R7
	SUBD R8,**R1**,R7
	OR R9,R6,R7
		Comparators detect use of R1 in SUB and forward result of load to ALU in time for SUB to begin EX.

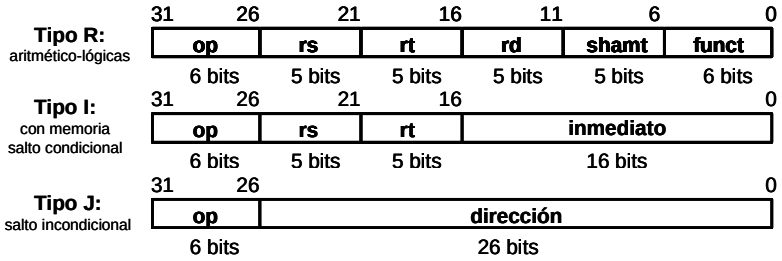
Dependence with accesses in order	LD **R1**,45(R2)
	ADDD R5,R6,R7
	SUBD R8,R6,R7
	OR R9,**R1**,R7
		No action required because the read of R1 by OR occurs in the second half of the ID phase, while the write of the loaded data occurred in the first half.

Implementación

- Detección de las paradas por Load en ID
  - Requiere 3 comparaciones

Opcode field of ID/EX ($ID/EX.IR_{0..5}$)	Opcode field of IF/ID ($IF/ID.IR_{0..5}$)	Matching operand fields
Load	Register-register ALU	ID/EX.IR [rt] == IF/ID.IR [rs]
Load	Register-register ALU	ID/EX.IR [rt] == IF/ID.IR [rt]
Load	Load, store, ALU immediate, or branch	ID/EX.IR [rt] == IF/ID.IR [rs]

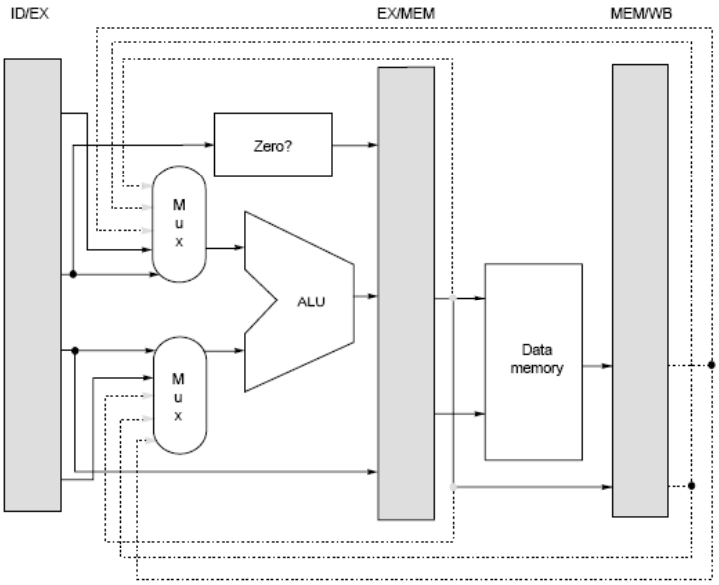
FIGURE A.21 The logic to detect the need for load interlocks during the ID stage of an instruction requires three comparisons. Lines 1 and 2 of the table test whether the load destination register is one of the source registers for a register-register operation in ID. Line 3 of the table determines if the load destination register is a source for a load or store effective address, an ALU immediate, or a branch test. Remember that the IF/ID register holds the state of the instruction in ID, which potentially uses the load result, while ID/EX holds the state of the instruction in EX, which is the potential load instruction.



Implementación del Forwarding

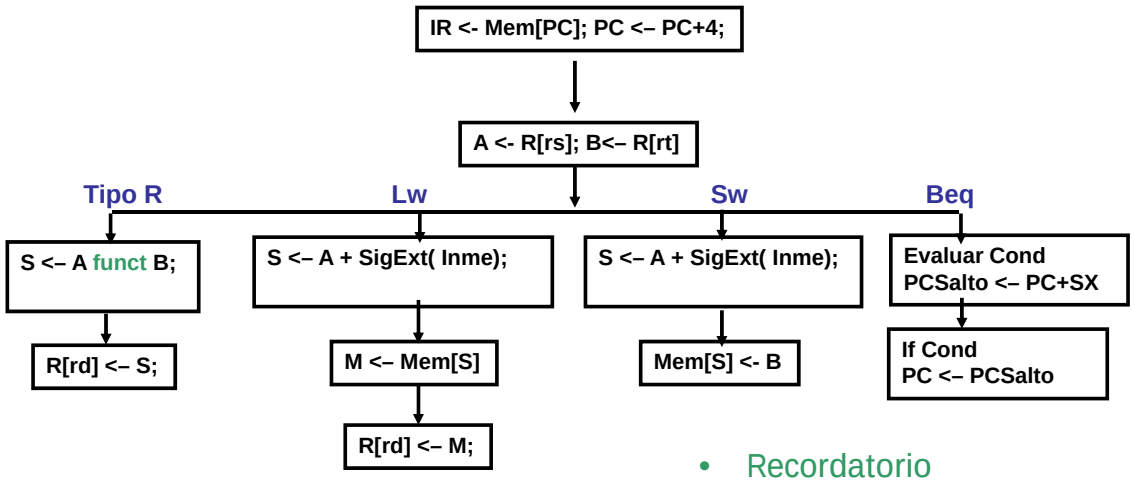
Pipeline register containing source instruction	Opcode of source instruction	Pipeline register containing destination instruction	Opcode of destination instruction	Destination of the forwarded result	Comparison (if equal then forward)
EX/MEM	Register-register ALU	ID/EX	Register-register ALU, ALU immediate, load, store, branch	Top ALU input	EX/MEM.IR [rd] == ID/EX.IR [rs]
EX/MEM	Register-register ALU	ID/EX	Register-register ALU	Bottom ALU input	EX/MEM.IR [rd] == ID/EX.IR [rt]
MEM/WB	Register-register ALU	ID/EX	Register-register ALU, ALU immediate, load, store, branch	Top ALU input	MEM/WB.IR [rd] == ID/EX.IR [rs]
MEM/WB	Register-register ALU	ID/EX	Register-register ALU	Bottom ALU input	MEM/WB.IR [rd] == ID/EX.IR [rt]
EX/MEM	ALU immediate	ID/EX	Register-register ALU, ALU immediate, load, store, branch	Top ALU input	EX/MEM.IR [rt] == ID/EX.IR [rs]
EX/MEM	ALU immediate	ID/EX	Register-register ALU	Bottom ALU input	EX/MEM.IR [rt] == ID/EX.IR [rt]
MEM/WB	ALU immediate	ID/EX	Register-register ALU, ALU immediate, load, store, branch	Top ALU input	MEM/WB.IR [rt] == ID/EX.IR [rs]
MEM/WB	ALU immediate	ID/EX	Register-register ALU	Bottom ALU input	MEM/WB.IR [rt] == ID/EX.IR [rt]
MEM/WB	Load	ID/EX	Register-register ALU, ALU immediate, load, store, branch	Top ALU input	MEM/WB.IR [rt] == ID/EX.IR [rs]
MEM/WB	Load	ID/EX	Register-register ALU	Bottom ALU input	MEM/WB.IR [rt] == ID/EX.IR [rt]

Implementación del Forwarding



Diseño del control

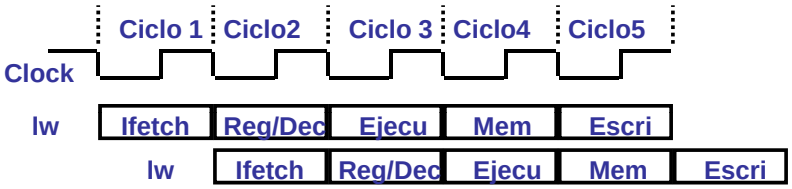
- Ejecución de instrucciones: diagrama RT



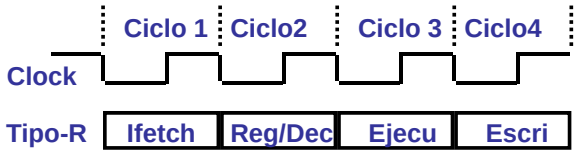
- Recordatorio
  - ✓ rs = Ins[25-21]
  - ✓ rt = Ins[20-16]
  - ✓ rd = Ins[15-11]
  - ✓ Inme = Ins[15-0]

Diseño del control

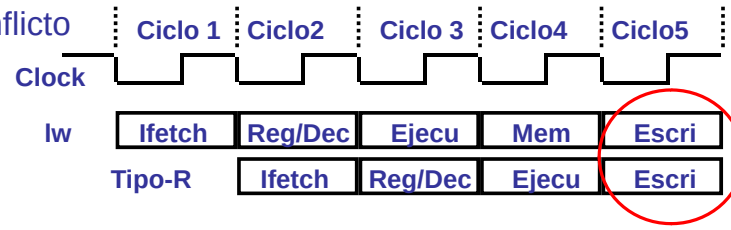
- Las instruciones: Load (lw) y tipo-R
  - ✓ Load. Tiene 5 fases y cada una utiliza una de las etapas



  - ✓ Tipo-R. Tiene 4 fases y no usa la memoria de datos



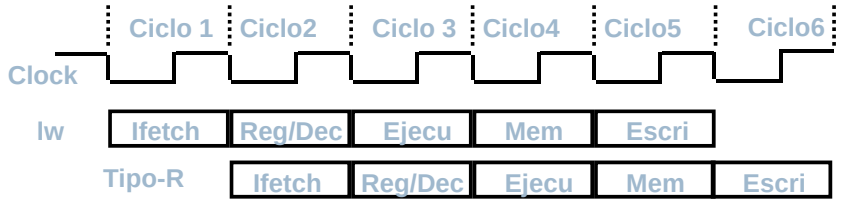
  - ✓ Conflicto



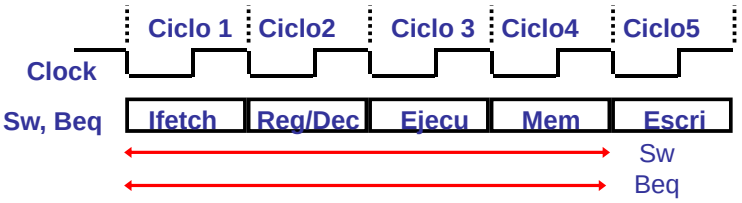
Solo una puerta en el bloque de registros

Diseño del control

- Importante. ¿ como evitarlo ?
  - ✓ Cada etapa del procesador puede usarse en cada ciclo por una sola instrucción.
  - ✓ Cada instrucción debe usar cada etapa del procesador en la misma fase de ejecución.
- Solución.
  - ✓ Introducir una etapa (Mem) que no hace nada.

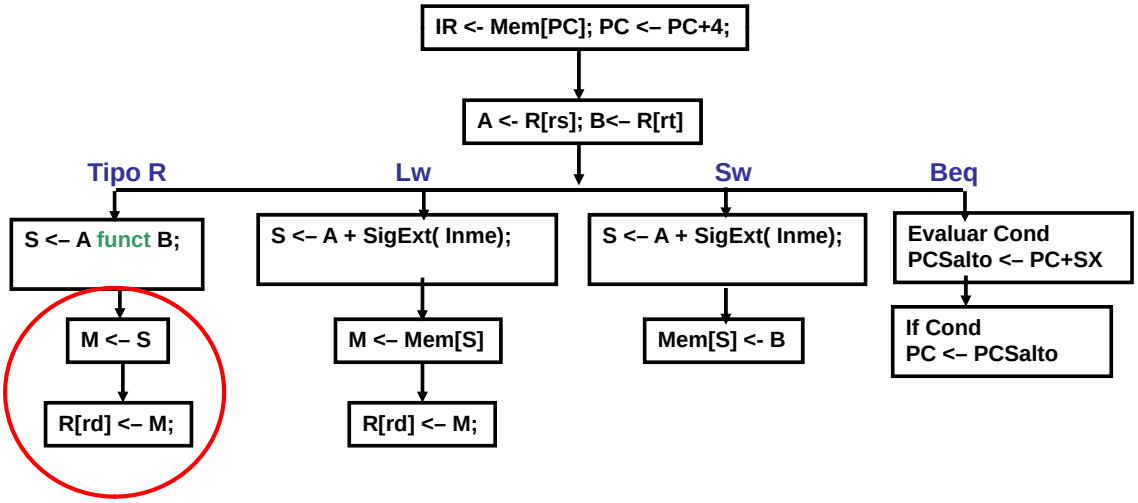


  - Store 4 etapas con actividad. Branch cuatro etapas activas

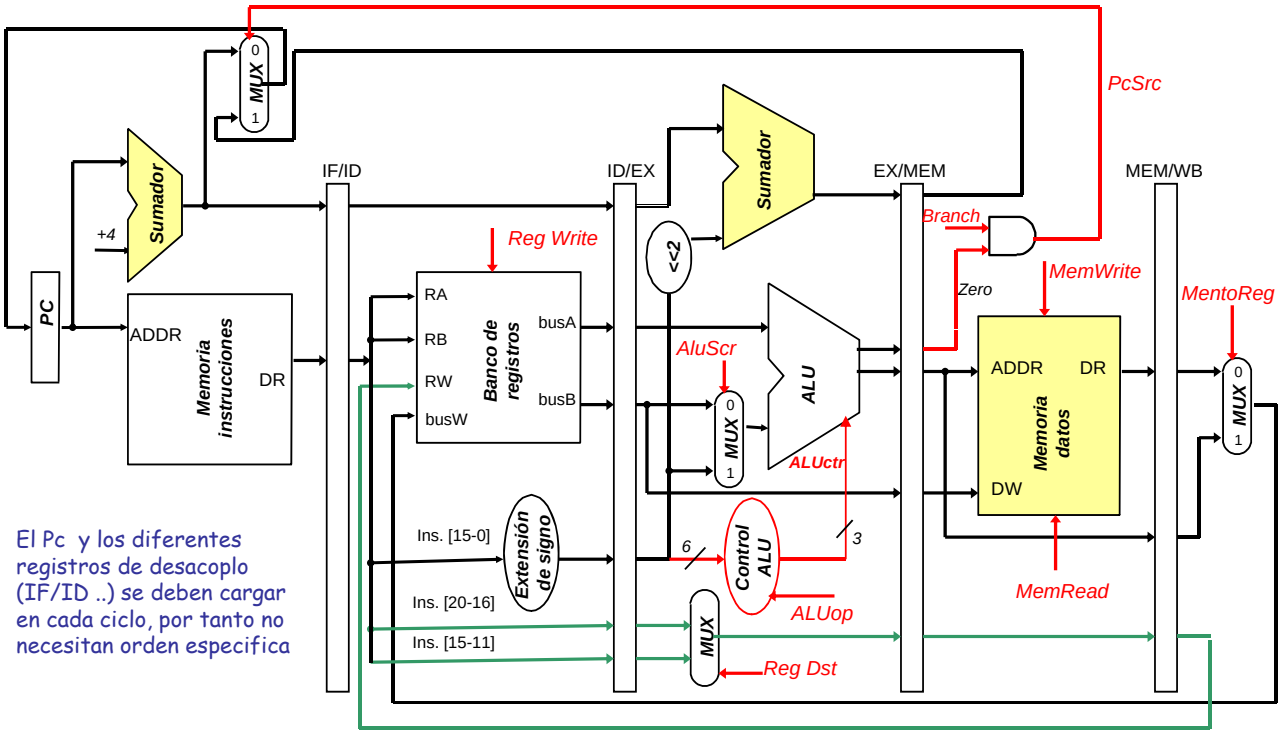


Diseño del control

- Ejecución de instrucciones: Diagrama RT modificado



Ruta de datos del DLX con las ordenes de control necesarias



El Pc y los diferentes registros de desacoplo (IF/ID ..) se deben cargar en cada ciclo, por tanto no necesitan orden especifica

Señales de control

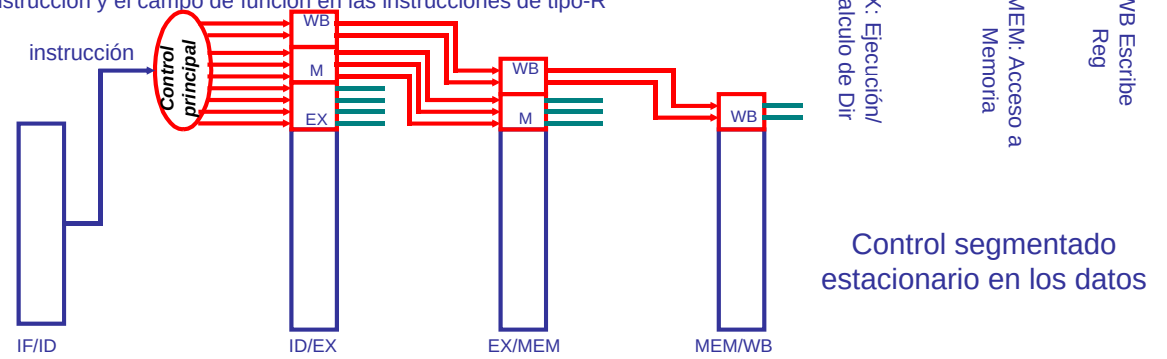
Control de la ALU

op	funct	ALUop	ALUctr
100011 (lw)	XXXXXX	00	010
101011 (sw)		00	010
000100 (beq)		01	110
000000 (tipo-R)	100000 (add)	10	010
	100010 (sub)	10	110
	100100 (and)	10	000
	100101 (or)	10	001
	101010 (slt)	10	111

Control principal

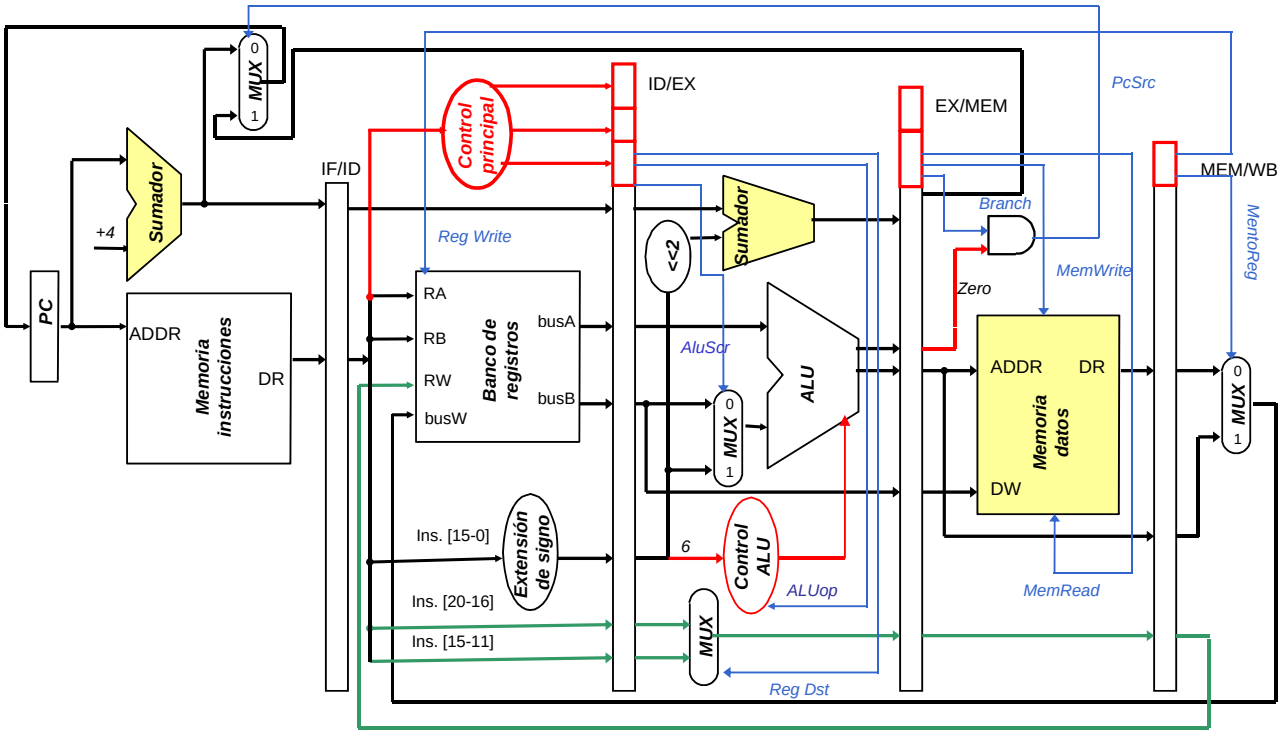
op	RegDst	ALUSrc	ALUop	MemRead	MemWrite	Branch	RegWrite	MemtoReg
100011 (lw)	0	1	00	1	0	0	1	0
101011 (sw)	X	1	00	0	1	0	0	X
000100 (beq)	X	0	01	0	0	1	0	X
000000 (tipo-R)	1	0	10	0	0	0	1	1

El control de la alu se determina por ALUop que depende del tipo de Instrucción y el campo de función en las instrucciones de tipo-R



Control segmentado estacionario en los datos

Ruta de datos del DLX con las ordenes de control y control estacionario en los datos

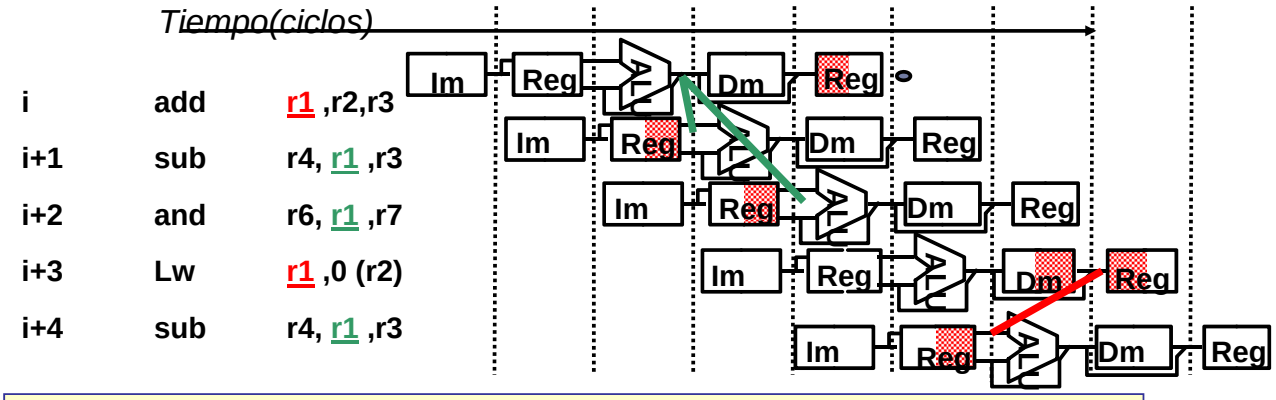


Diseño del control

- ¿Qué facilita el control segmentado?
  - Todas las instrucciones con igual duración
  - Pocos formatos diferentes de instrucción
  - Accesos a memoria solo en load y store
- ¿Qué dificulta el control segmentado?
  - Riesgos estructurales. Conflictos de recursos
  - Riesgos de datos. Solo LDE
  - Riesgos de control
- El diseño anterior no tenía en cuenta los riesgos de datos y los riesgos de control las eliminaba con saltos retardados
  - Implementación del cortocircuito
  - Caso del load
  - Mejora en el comportamiento de los saltos.

Diseño del control con riesgos

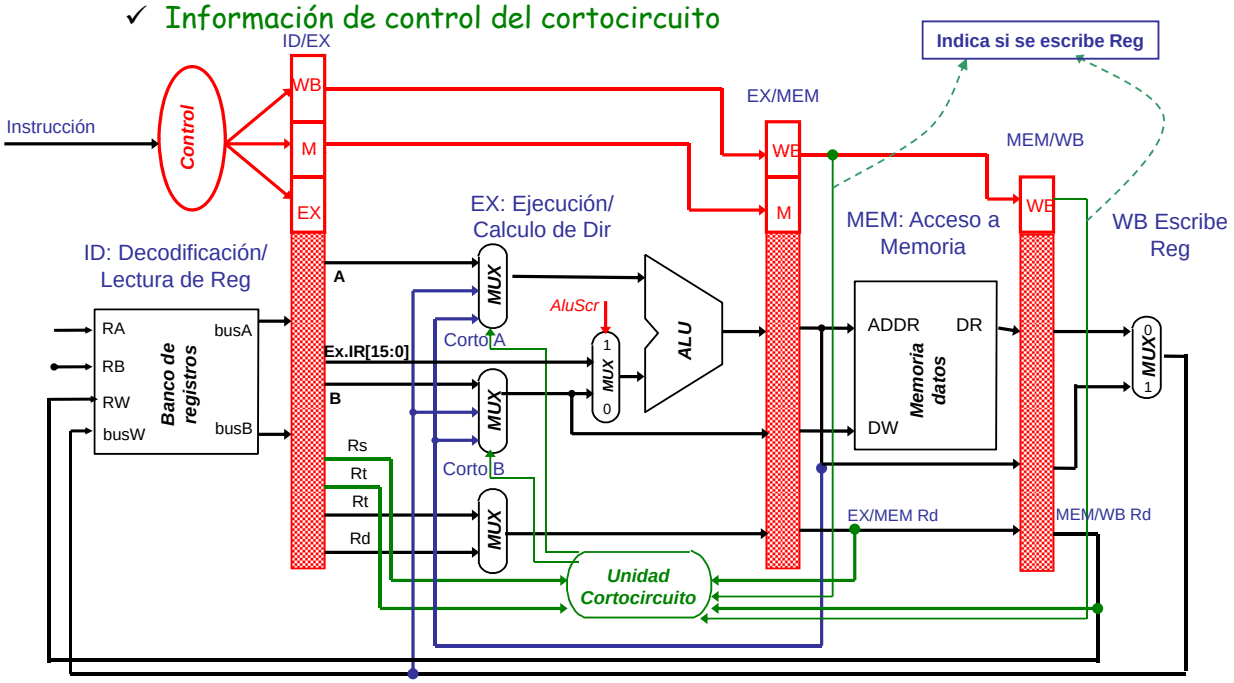
- Riesgos de datos $R(i) \cap D(i+?) \neq \phi$ LDE (RAW).
  - Cortocircuito. Enviar el dato a las etapas que lo necesitan, cuanto esta listo



Dos caminos de datos: Desde salida ALU (EX/MEM) a entrada ALU
Desde la salida de la memoria (MEM/WB) a entrada ALU
Información necesaria: Registro a escribir en ultima etapa (Rd en Tipo-R y Rt en Lw)
Registros que se leen en segunda etapa (Rs y Rt)

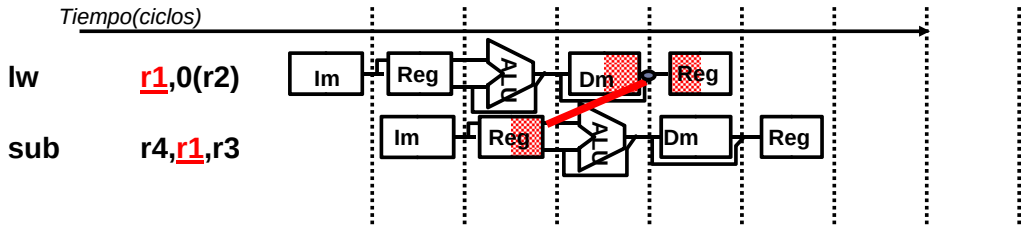
Diseñando el control con riesgos LDE: Implementación del cortocircuito

- ✓ Cortocircuitos de datos
- ✓ Información de control del cortocircuito

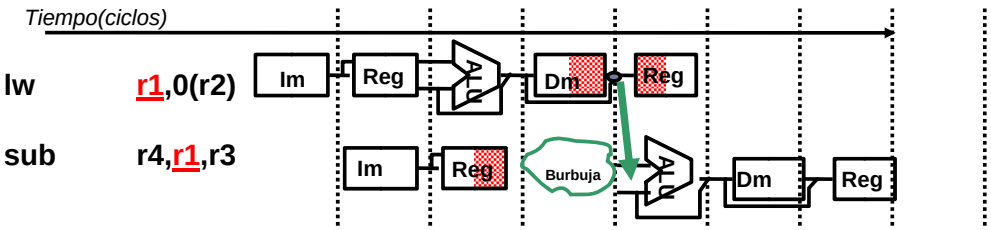


Diseño del control con riesgos

- Riesgos de datos LDE: Caso del load

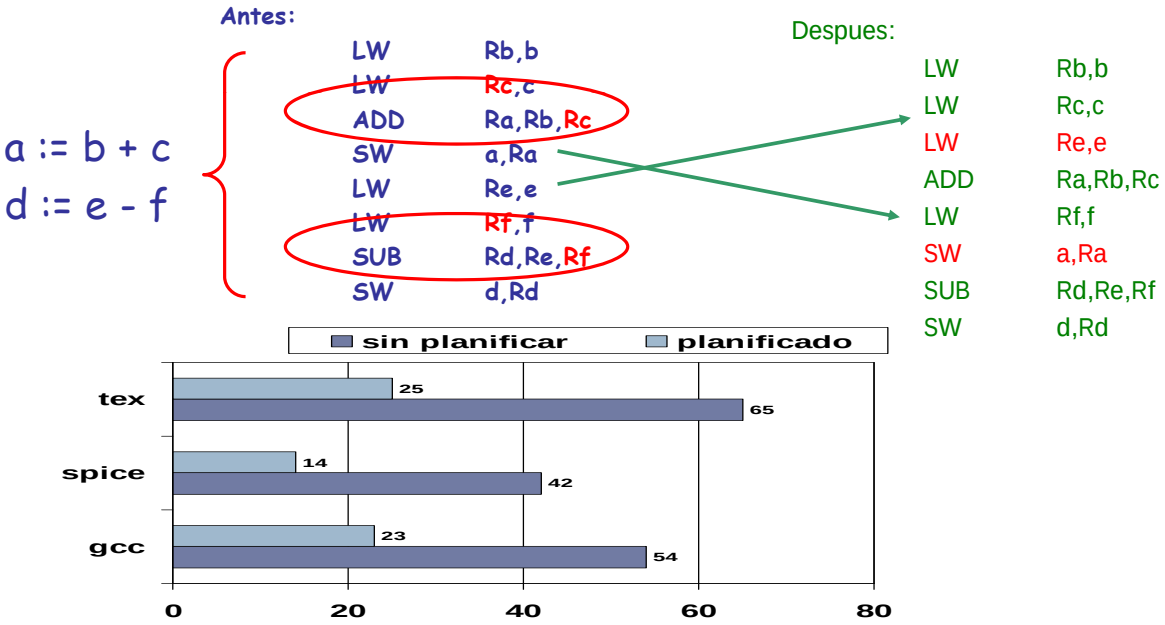


- Se debe esperar un ciclo a pesar del cortocircuito



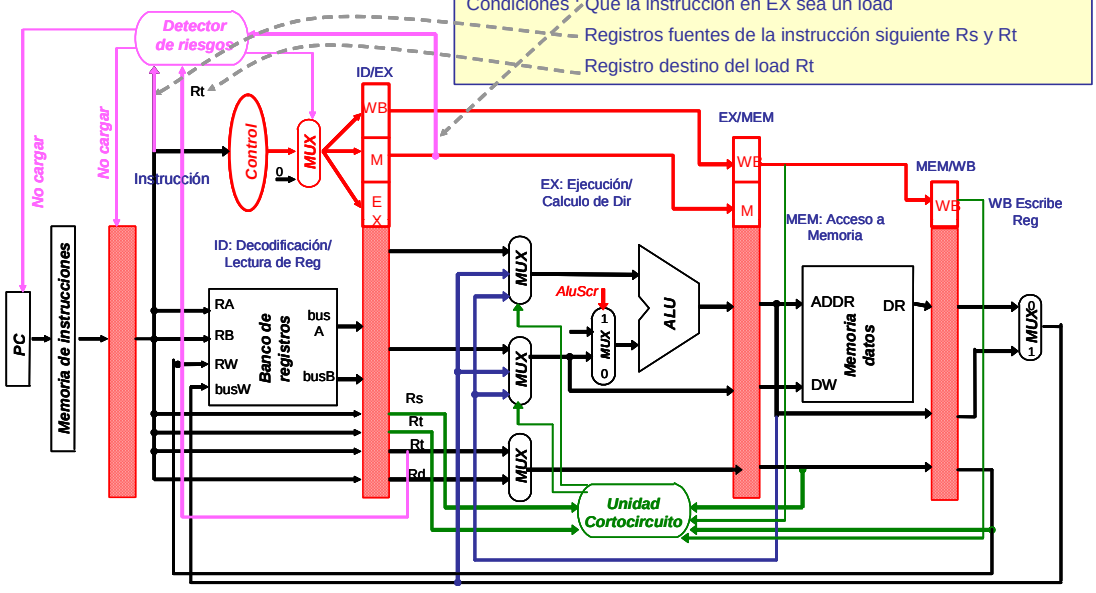
Diseño del control con Riesgos LDE: Caso del load

- Solución SW :
  - Anticipar el Load en la planificación que hace el compilador



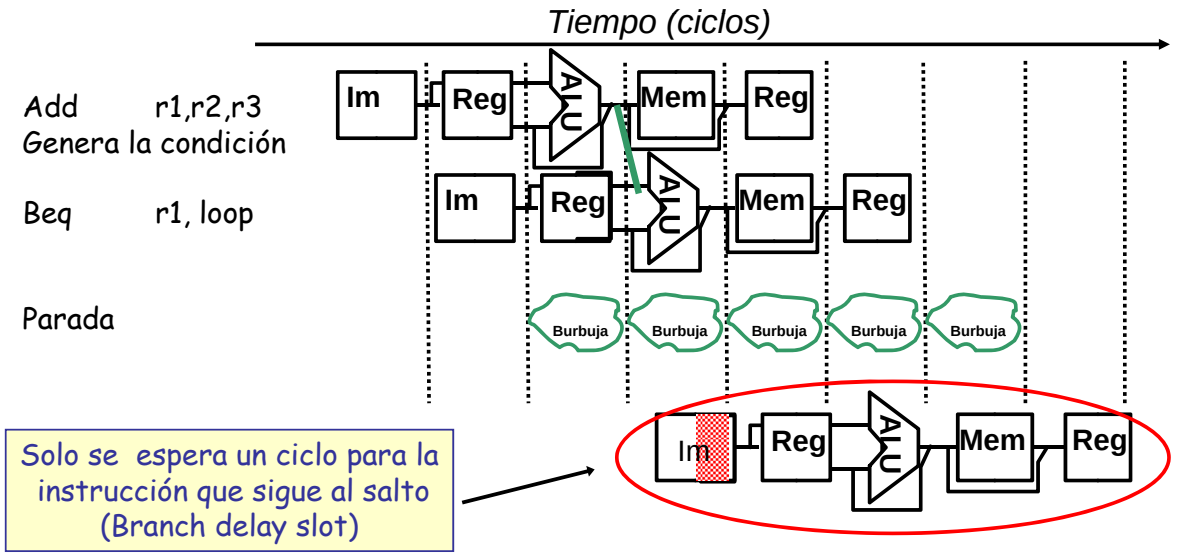
Solución HW: Detección de riesgos y parada del procesador un ciclo

Detector de riesgos
Debe actuar en ID impidiendo la carga y lectura de una nueva instrucción
Condiciones : Que la instrucción en EX sea un load
Registros fuentes de la instrucción siguiente Rs y Rt
Registro destino del load Rt



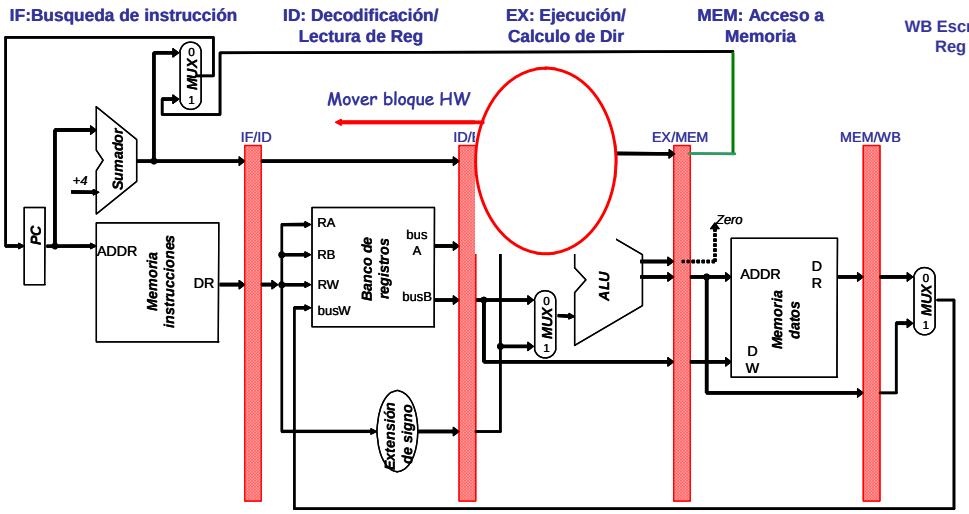
Diseño del control con riesgos

- Riesgos de control
  - ✓ Solución: Desplazar el calculo de la dirección y la evaluación de la condición a la etapa anterior



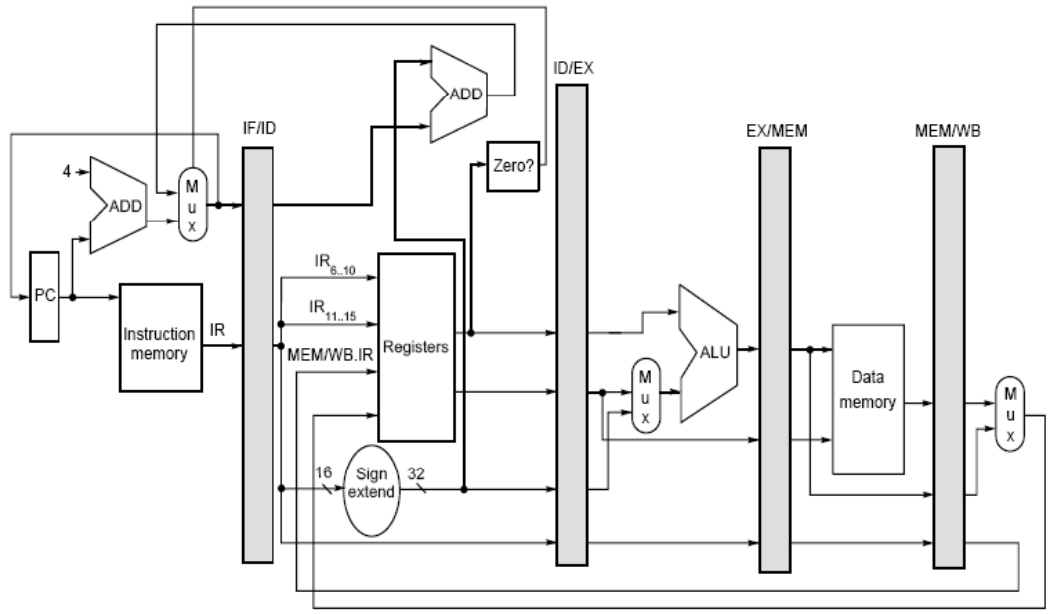
Diseño del control con riesgos

- Mejora del comportamiento de los saltos. Solo un ciclo de parada
  - Calculo de la dirección. Operandos disponibles (Pc y desplazamiento)
  - Calculo de la condición. Unidad de detección de cero



Reducción de las paradas por Riesgos de control

- Mover el cálculo de la condición de salto a la etapa anterior



Reducción de las paradas por Riesgos de control

- Mover el cálculo de la condición de salto a la etapa anterior

Pipe stage	Branch instruction
IF	IF/ID.IR ← Mem[PC]; **IF/ID.NPC,PC ← (if ((IF/ID.opcode == branch) & (Regs[IF/ID.IR$_{6..10}$] op 0)) {IF/ID.NPC + (IF/ID.IR$_{16}$)^{16}##IF/ID.IR$_{16..31}$} else {PC+4});**
ID	ID/EX.A ← Regs[IF/ID.IR$_{6..10}$]; ID/EX.B ← Regs[IF/ID.IR$_{11..15}$]; ID/EX.IR ← IF/ID.IR; ID/EX.Imm ← (IF/ID.IR$_{16}$)^{16}##IF/ID.IR$_{16..31}$
EX	
MEM	
WB	

Dificultades en la implementación de pipelines

- Tratar de las excepciones
- Paradas de la ejecución
- Reinicio tras las paradas
- Complicaciones del repertorio de instrucciones

Índice

1. Segmentación
2. Riesgos (Hazards)
3. Forwarding
4. Implementación del pipelining
5. Tratamiento de las interrupciones
6. Complicaciones del Repertorio de Instrucciones
7. Extensiones para Multiciclo

Tratamiento de Interrupciones

- Situaciones en las que se cambia el orden natural de la ejecución
- Tipos de Excepciones
  - Peticiones de dispositivos de E/S
  - Solicitud de utilización del SO
  - Realización de trazas
  - Breakpoints
  - Overflows de las unidades funcionales de enteros
  - Situaciones anómalas de punto flotante
  - Fallo de página
  - Desalinamiento de memoria
  - Violación de la protección de memoria
  - Instrucciones no definidas
  - Problema de Hardware
  - Problema de potencia

Caracterización de las Excepciones

- Por su solicitud
  - Síncronas
  - Asíncronas
- Quién las solicita
  - Solicitadas por el usuario
  - Forzadas
- Por su prioridad
  - Enmascarables o no
- Cuando se producen
  - Dentro o fuera de la ejecución de una instrucción
  - Al final o dentro de un programa

Clasificación

Tipo de Excepción	Síncronas o Asíncronas	Usuario o Forzada	Enmascarable?	Dentro o entre Instrucciones	Reinicio o Finalización
Petición E/S	Asíncronas	Forzada	No	Entre	Reinicio
Solicitud SO	Síncronas	Usuario	No	Entre	Reinicio
Trazas	Síncronas	Usuario	Sí	Entre	Reinicio
Breakpoints	Síncronas	Usuario	Sí	Entre	Reinicio
Overflows INT	Síncronas	Forzada	Sí	Dentro	Reinicio
PF	Síncronas	Forzada	Sí	Dentro	Reinicio
Fallo de página	Síncronas	Forzada	No	Dentro	Reinicio
Desalinamiento	Síncronas	Forzada	Sí	Dentro	Reinicio
Protección Mem.	Síncronas	Forzada	No	Dentro	Reinicio
I. no definidas	Síncronas	Forzada	No	Dentro	Finalización
Problema de Hw	Asíncronas	Forzada	No	Dentro	Finalización
Problema Pot.	Asíncronas	Forzada	No	Dentro	Finalización

Parada y reinicio de la ejecución

- Las excepciones más difíciles ocurren dentro de las instrucciones (fases EX o MEM)
  - Deben ser reiniciables
- Ejemplo:
  - Fallo de página virtual
    - Debe ser reiniciable
    - Requiere la intervención de otro proceso (SO)
    - Salvar el PC
      - Branch: reevaluación de la condición
      - No Branch: Normal
- Que puede hacer el control del pipe
  - Hacer IF trap a la instrucción
  - Apagar escrituras para la instrucción que provoca la excepción y las siguientes
  - Salvar el PC

Utilización de saltos retardados

- No se puede recrear el estado del procesador con un único PC
  - Las instrucciones en el pipe pueden no estar relacionadas secuencialmente
  - Podemos necesitar tantos PC´s como longitud del salto +1
- Excepciones precisas
  - Se puede parar el pipe
  - Las instrucciones justo antes se terminan
  - Continuan las siguientes desde el principio
- Excepciones en PF
  - A veces escriben el resultado antes de que aparezca la excepción
    - Eliminar operandos
    - Finalización fuera de orden

Interrupciones Precisas

- 2 Modos de funcionamiento
  - Precisas
  - No precisas
- Asegurar siempre aunque sea por Software
  - Fácil en Integer
  - Difícil en FP

Pipeline stage	Problem exceptions occurring
IF	Page fault on instruction fetch; misaligned memory access; memory-protection violation
ID	Undefined or illegal opcode
EX	Arithmetic exception
MEM	Page fault on data fetch; misaligned memory access; memory-protection violation
WB	None

Excepciones en el MIPS

- Pueden ocurrir fuera de orden
- Vector de estado
  - Se chequea después de MEM
  - Se manejan en orden de ejecución
  - Deshabilitación de las escrituras

Etapa	Problema
IF	Fallo de Página en Instrucción, desaliniamiento, violación de protección
ID	Opcode no definido o ilegal
EX	Excepción aritmética
MEM	Fallo de Página en Datos, desaliniamiento, violación de protección
WB	Ninguna

Índice

1. Segmentación
2. Riesgos (Hazards)
3. Forwarding
4. Implementación del pipelining
5. Tratamiento de las interrupciones
6. Complicaciones del Repertorio de Instrucciones
7. Extensiones para Multiciclo

Complicaciones del Repertorio de Instrucciones

- Algunos procesos pueden ocasionar excepciones entre medias de una instrucción
  - Autoincremento de la dirección en un IA-32 puede actualizar los registros en medio de la ejecución
  - Hace falta Hardware para gestionarlos
- Actualización de la memoria durante la ejecución
  - VAX; IBM360
  - Se puede solucionar
    - Se pueden utilizar bancos de registros temporales, para guardar trabajos parcialmente completados
    - Se añaden bits de estado para saber si está completado o no

Complicaciones del Repertorio de Instrucciones

- Códigos de condición incluídos en la instrucción
  - Dificultades de planificación de retardos del pipeline
  - Decidir cuando es fija la condción de salto
- Operaciones Multi-ciclo

MOVL R1, R2 ;	Movimiento entre registros
ADDL3 42(R1),56(R1)+,@(R1);	Suma posiciones de memoria
SUBL2 R2,R3 ;	Resta entre registros
MOVC3 @R1[R2],74(R2), R3;	Mover un carácter

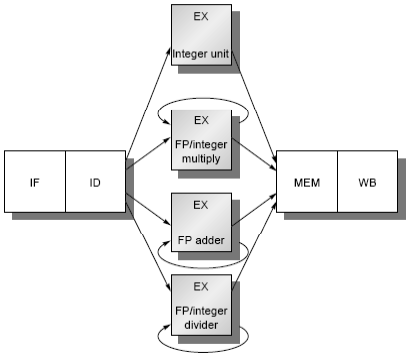
  - Solución : Hacer pipelining a nivel de microinstrucción

Índice

1. Segmentación
2. Riesgos (Hazards)
3. Forwarding
4. Implementación del pipelining
5. Tratamiento de las interrupciones
6. Complicaciones del Repertorio de Instrucciones
7. Extensiones para Multiciclo

Operaciones Multi-ciclo

- Operaciones de un solo ciclo
  - Mucho Hardware
  - Reloj muy lento
- Operaciones en Punto Flotante
  - Las fases de ejecución se puede repetir tantas veces como sea necesario
  - Puede haber varias unidades funcionales
  - Ejemplo con 4 unidades funcionales
    - Enteros: ALU, Branches, LD y Store
    - Multiplicación: Entera y FP
    - Sumador FP
    - División: Entera y PF



Operaciones Multi-ciclo

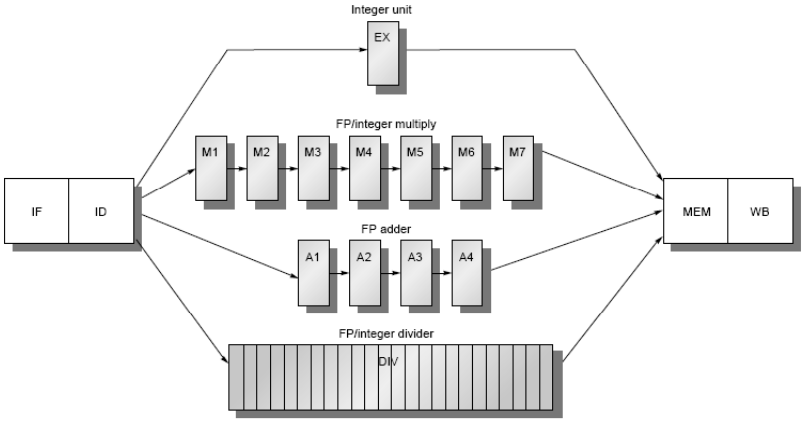
- Permitir un pipe similar al hecho con MIPS
- Latencia de las Ufs
  - N° de ciclos que hay que esperar para poder utilizar el resultado
- Intervalo de inicialización
  - N° ciclos que hay que dejar pasar para lanzar dos intrucciones del mismo tipo
- Ejemplo

UF	Latencia	I. Incialización
IALU	0	1
Dmem	1	1
FP add	3	1
FP MUL	6	1
FP DIV	24	25

  - EL resultado de IALU se puede usar en el siguiente ciclo
  - Sol: Reducir Lógica → Aumenta el n° de etapas → II → Relojes + rápidos → Aumento de la latencia

Operaciones Multi-ciclo

- Solo puede haber activa una operación
- Mayor Latencia => riesgos de LDE y sus paradas
- Nuevos registros entre etapas y modificaciones a la conexión
- Información de control solo con una etapa



Hazards y Forwarding para pipelines de latencia mayores

- Aspectos a considerar
  - Riesgos estructurales
    - La división no está totalmente segmentada
  - Número de escrituras en Registros
    - Puede ser mayor que 1 en un ciclo
  - Riesgos de Escritura después de Escritura
    - No se alcanza al etapa de WB en orden
  - Terminación fuera de orden
    - No Riesgos de Escritura después de lectura
    - La lectura de registros se realiza siempre en ID
  - Incremento de la latencia
    - Puede provocar paradas por Riesgos de Lectura después de escritura
    - RAW / LDE

RAW y Escrituras simultaneas

	Clock cycle number																
Instruction	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
L.D F4,0(R2)	IF	ID	EX	MEM	WB												
MULT.D F0,F4,F6		IF	ID	stall	M1	M2	M3	M4	M5	M6	M7	MEM	WB				
ADD.D F2,F0,F8			IF	stall	ID	stall	stall	stall	stall	stall	stall	A1	A2	A3	A4	MEM	
S.D 0(R2),F2					IF	stall	stall	stall	stall	stall	stall	ID	EX	stall	stall	stall	MEM

	Clock cycle number										
Instruction	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
MULT,D F0,F4,F6	IF	ID	M1	M2	M3	M4	M5	M6	M7	MEM	WB
...		IF	ID	EX	MEM	WB					
...			IF	ID	EX	MEM	WB				
ADD,D F2,F4,F6				IF	ID	A1	A2	A3	A4	MEM	WB
...					IF	ID	EX	MEM	WB		
...						IF	ID	EX	MEM	WB	
L,D F2,0(R2)							IF	ID	EX	MEM	WB

Operaciones Multi-ciclo

- 2 soluciones
  - Registro de desplazamiento + Stall
    - Realizar la comprobación en ID
  - Parada de las instrucciones en conflicto después de EXE
    - Necesidad de Prioridades de Accesso
    - Lógica de Chequeo
    - Paso a atrás de memoria
    - Dejar pasar la de mayor latencia
- Riesgos WAW
  - Si el load va un ciclo antes
  - Useless instruction

Implementación

- Lógica de Riesgos y lanzamiento para el pipeline FP
  - Suponemos que se realiza la detección en ID
    - Mismos conceptos que en MIPS
      - Forwarding en:
        - EX/MEM
        - A4/MEM
        - M7/MEM
        - D/MEM
        - MEM/WB
    - Necesitamos 3 comprobaciones

1. Riesgos estructurales
   - Esperar hasta que la UF necesaria esté libre
   - Asegurarse de que el puerto de escritura esté libre
2. RAW
   - Esperar hasta que los registros fuente estén disponibles
3. WAW
   - Determinar si alguna instrucción en A1,...,A4, D y M1,.....,M7 tienen el mismo registro de destino

Influencia en las Excepciones

- Mantenimiento de las excepciones precisas
  DIV.D F0,F2,F4
  ADD.D F10,F10,F8
  SUB.D F12,F12,F14
- No hay dependencias pero DIV acabará + tarde
- Terminamos fuera de orden
- Problema excepción en medio de DIV
  - Si ha terminado ADD puede no recuperarse porque F10 se destruye y no podemos volver atrás
  - Aproximaciones
    - Ignorarlos (Problema con la MV)
      - 60's y 70's
    - Dos modos de ejecución (Impreciso y preciso). 1operación de PF cada vez
      - 21064, 21164, Power1, Power2, MIPS R8000
    - Almacenar Resultados Parciales
      - Comparadores y MUX
      - Registros de Historia
      - Registro de futuro
    - Permitir algunas condiciones de imprecisión

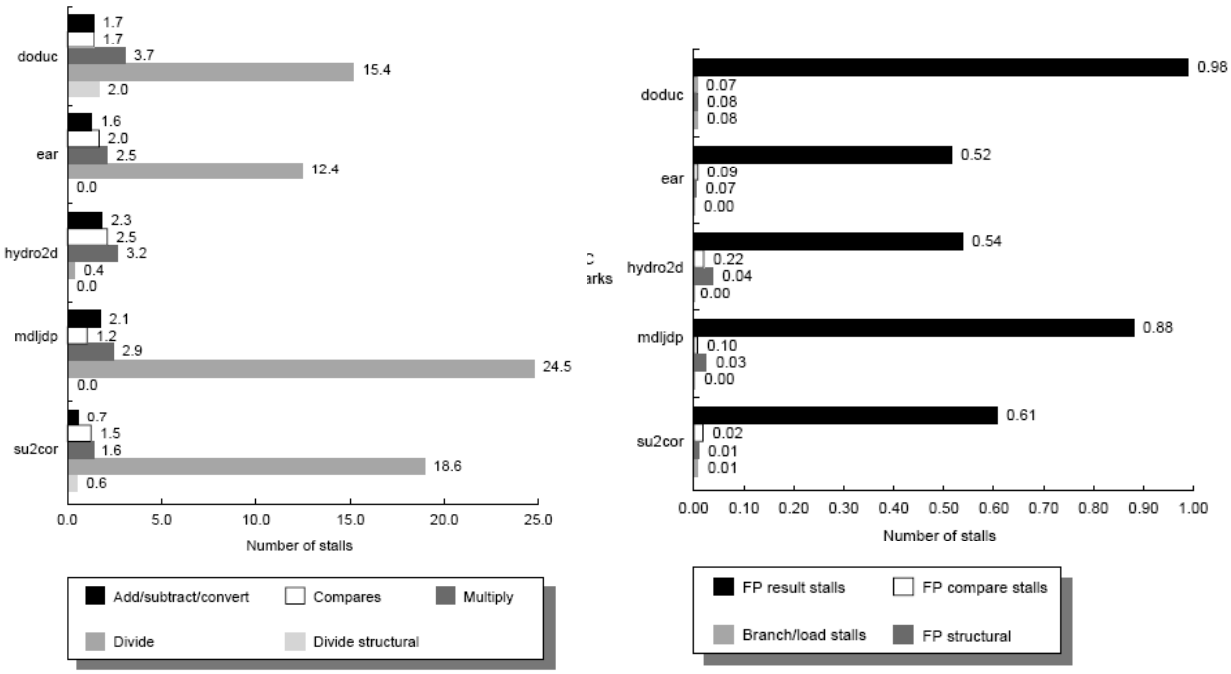
Influencia en las Excepciones

- Ejemplo

Instrucción 1;	Causa la interrupción
Instrucción 2;	No completada
.....	No completada
Instrucción n-1;	No completada
Instrucción n;	Completada

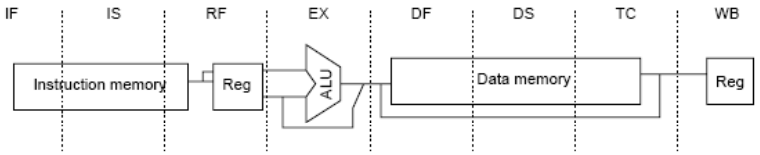
- Se almacena el PC para I2 a In-1 y luego se continúa con In
  - Variación ejecutar solo las instrucciones en PF que haya entre I2 e In-1
- Técnicas Híbridas
  - Permitir el lanzamiento para continuar
  - Solo en caso de que se hayan terminado las anteriores sin excepciones

Paradas por operaciones en PF



Ejemplo: MIPS R4000 Pipeline

- 8 etapas
  - Incremento en el Forwarding
  - Incremento de retardos por load y branch
  - MI está activa en la etapa RF para comprobación de etiquetas

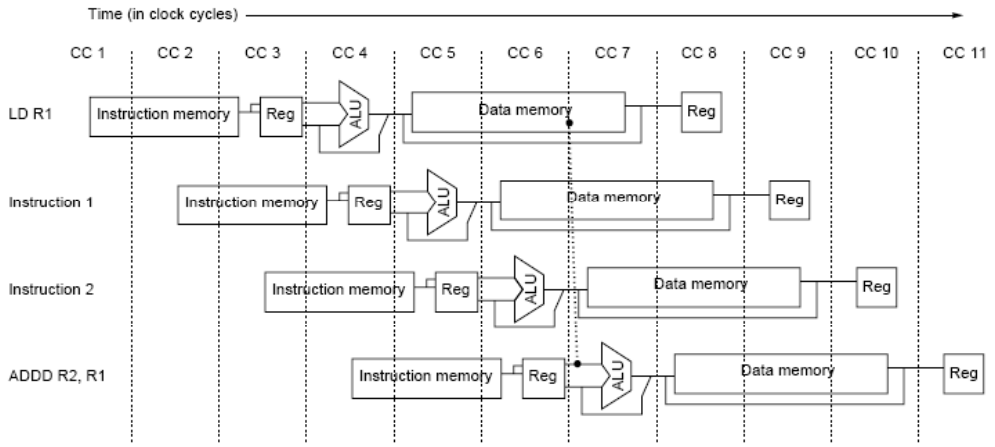


  - IF: PRIMERA PARTE DEL FETCH
  - IS: SEGUNDA PARTE DEL FETCH
  - RF: DECODIFICACIÓN Y FETCH DE REGISTROS
  - EX: EJECUCIÓN
  - DF: DATA FETCH PRIMERA PARTE
  - DS: DATA FETCH SEGUNDA PARTE
  - TC: COMPARACIÓN DE ETIQUETAS
  - WB: ESCRITURA

MIPS R4000 Pipeline

- IF: PRIMERA PARTE DEL FETCH
  - Selección del PC
  - Inicio del acceso a la cache de Instrucciones
- IS: SEGUNDA PARTE DEL FETCH
  - Finalización del acceso a la Cache de instrucciones
- RF: DECODIFICACIÓN Y FETCH DE REGISTROS
  - Decodificación de la instrucción
  - Comprobación de Riesgos
  - Detección de acierto en acceso a la Cache de Instrucciones
- EX: EJECUCIÓN
  - Cálculo de la Dirección efectiva
  - Operaciones Aritmético Lógica
  - Evaluación de la condición de salto
  - Cálculo de la dirección de salto
- DF: DATA FETCH PRIMERA PARTE
  - Primera parte del acceso a la caché de datos
- DS: DATA FETCH SEGUNDA PARTE
  - Finalización del acceso a la caché de datos
- TC: COMPARACIÓN DE ETIQUETAS
  - Comprobación de etiquetas
  - Detección de acierto en acceso a la Cache de Datos
- WB: ESCRITURA
  - Escritura de resultados para operaciones de Load y ALU

Incremento en la penalización



	Clock number								
Instruction number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
LD R1,...	IF	IS	RF	EX	DF	DS	TC	WB	
ADDD R2,R1,...		IF	IS	RF	stall	stall	EX	DF	DS
SUBD R3,R1,...			IF	IS	stall	stall	RF	EX	DF
OR R4,R1,...				IF	stall	stall	IS	RF	EX

Operaciones en PF

- Existen 3 unidades funcionales en PF
  - División en PF
  - Multiplicación en PF
  - Suma en PF
    - También se utiliza al final de las operaciones de mul y div
  - Duración desde 2 ciclos a 112
  - Segmentado en 8 etapas
    - Combinadas en diferente orden
      - A: +FP suma de mantisa
      - D: %FP Etapa de división
      - E: *FP Chequeo de excepciones
      - M: *FP Primera etapa de Mul
      - N: *FP Segunda Etapa de Mul
      - R: +FP Redondeo
      - S: +FP Desplazamientp
      - U: Desempaquetado de números en PF

Latencias e Intervalos de Inicialización

Instrucción	Fases que utiliza										Lat	II
Suma y Resta	U	S+A	A+R	R+S							4	3
Multiplicación	U	E+M	M	M	M	N	N+A	R			8	4
División	U	A	R	D^{28}	D+A	D+R	D+A	D+R	A	R	36	35
Raíz cuadrada	U	E	$(A+R)_{108}$	A	R						112	111
Negación	U	S									2	1
Valor absoluto	U	S									2	1
Comparación FP	U	A	R								3	2

Conflictos entre operaciones en PF

- MUL, ADD

Oper	Lanzar/Parar	0	1	2	3	4	5	6	7	8	9
Mul	Lanzar	U	E+M	M	M	M	N	N+A	R		
Add	Lanzar		U	S+A	A+R	R+S					
	Lanzar			U	S+A	A+R	R+S				
	Parar				U	S+A	A+R	R+S			
	Parar					U	S+A	A+R	R+S		
	Lanzar						U	S+A	A+R	R+S	
	Lanzar							U	S+A	A+R	R+S

Conflictos entre operaciones en PF

- ADD , MUL

Oper	Lanzar/Parar	0	1	2	3	4	5	6	7	8	9
Add	Lanzar	U	S+A	A+R	R+S						
Mul	Lanzar		U	E+M	M	M	M	N	N+A	R	
	Lanzar			U	E+M	M	M	M	N	N+A	R

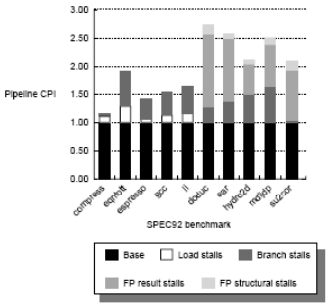
Conflictos entre operaciones en PF

- Ejercicio DIV, ADD

Solución

Oper	Lanzar/Parar	25	26	27	28	29	30	31	32	33	34	35	36
DIV	Lanzar	D	D	D	D	D	D+A	D+R	D+A	D+R	A	R	
Add	Lanzar		U	S+A	A+R	R+S							
	Lanzar			U	S+A	A+R	R+S						
	Parar				U	S+A	A+R	R+S					
	Parar					U	S+A	A+R	R+S				
	Parar						U	S+A	A+R	R+S			
	Parar							U	S+A	A+R	R+S		
	Parar								U	S+A	A+R	R+S	
	Lanzar									U	S+A	A+R	R+S

Ejemplo Rendimiento R4000



Benchmark	Pipeline CPI	Load stalls	Branch stalls	FP result stalls	FP structural stalls
compress	1.20	0.14	0.06	0.00	0.00
eqntott	1.88	0.27	0.61	0.00	0.00
espresso	1.42	0.07	0.35	0.00	0.00
gcc	1.56	0.13	0.43	0.00	0.00
li	1.64	0.18	0.46	0.00	0.00
Integer average	1.54	0.16	0.38	0.00	0.00
doduc	2.84	0.01	0.22	1.39	0.22
mdljdp2	2.66	0.01	0.31	1.20	0.15
ear	2.17	0.00	0.46	0.59	0.12
hydro2d	2.53	0.00	0.62	0.75	0.17
su2cor	2.18	0.02	0.07	0.84	0.26
FP average	2.48	0.01	0.33	0.95	0.18
Overall average	2.00	0.10	0.36	0.46	0.09

Bibliografía

- Apéndice A de [HePa06]
- Standard Performance Evaluation Corporation.
  - http://www.spec.org
- MIPS R4000 Microprocessor User's Manual
  - cag.csail.mit.edu/raw/documents/R4400_Uman_book_Ed2.pdf
- Simulador WinDLX
  - http://cs.uns.edu.ar/~jechaiz/arquitectura/windlx/windlx.html