

Arquitectura de computadoras



2020

Explicación Práctica 4

Conceptos generales

- ▶ La **segmentación de cauce** (*pipelining*) es una forma particularmente **efectiva** de organizar el hardware de la CPU para realizar más de una operación al mismo tiempo.
- ▶ Consiste en **descomponer el proceso de ejecución** de las instrucciones en **fases** (*etapas*) que permitan una ejecución simultánea.
- ▶ Explota el **paralelismo** entre las instrucciones de un flujo secuencial.

Conceptos generales

- **MSX88** → Conjunto **amplio** de instrucciones

ORG 1000H

NUM1	DW	5
NUM2	DW	12
NUM3	DB	1
RES	DW	?

ORG 2000H

MOV	AL,	NUM3
MOV	CX,	NUM2
MOV	BX,	OFFSET NUM1
ADD	CX,	AX
MOV	[BX],	CX
...		

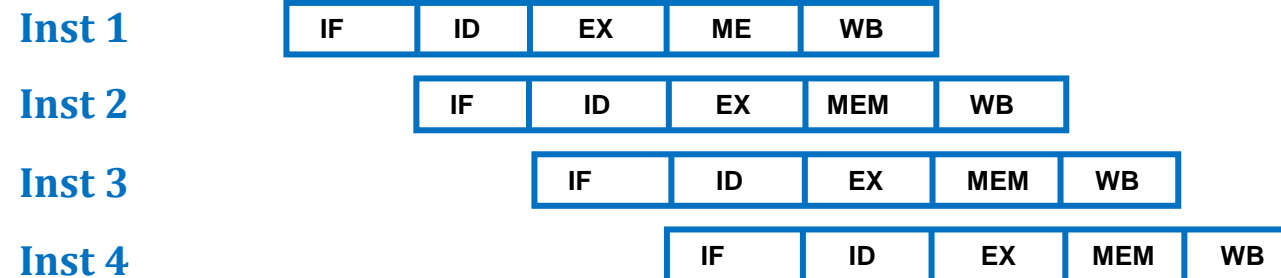
- **MIPS64** → Conjunto **reducido** de instrucciones

Conceptos generales

Ejecución secuencial (MSX88):



Ejecución segmentada (MIPS64):



Incrementa la productividad pero no reduce el tiempo de ejecución de la instrucción

Segmentación en MIPS64

Búsqueda (IF)

- ▶ Se accede a memoria por la instrucción
- ▶ Se incrementa el PC



Decodificación (ID)

- ▶ Se decodifica la instrucción
- ▶ Se accede al banco de registros por los operandos
- ▶ Se calcula el valor del operando inmediato
- ▶ Si es un salto, se calcula el destino y si se toma o no

Ejecución (EX)

- ▶ Si es una instrucción de proceso, se ejecuta en la ALU
- ▶ Si es un acceso a memoria, se calcula la dirección efectiva
- ▶ Si es un salto, se almacena el nuevo PC

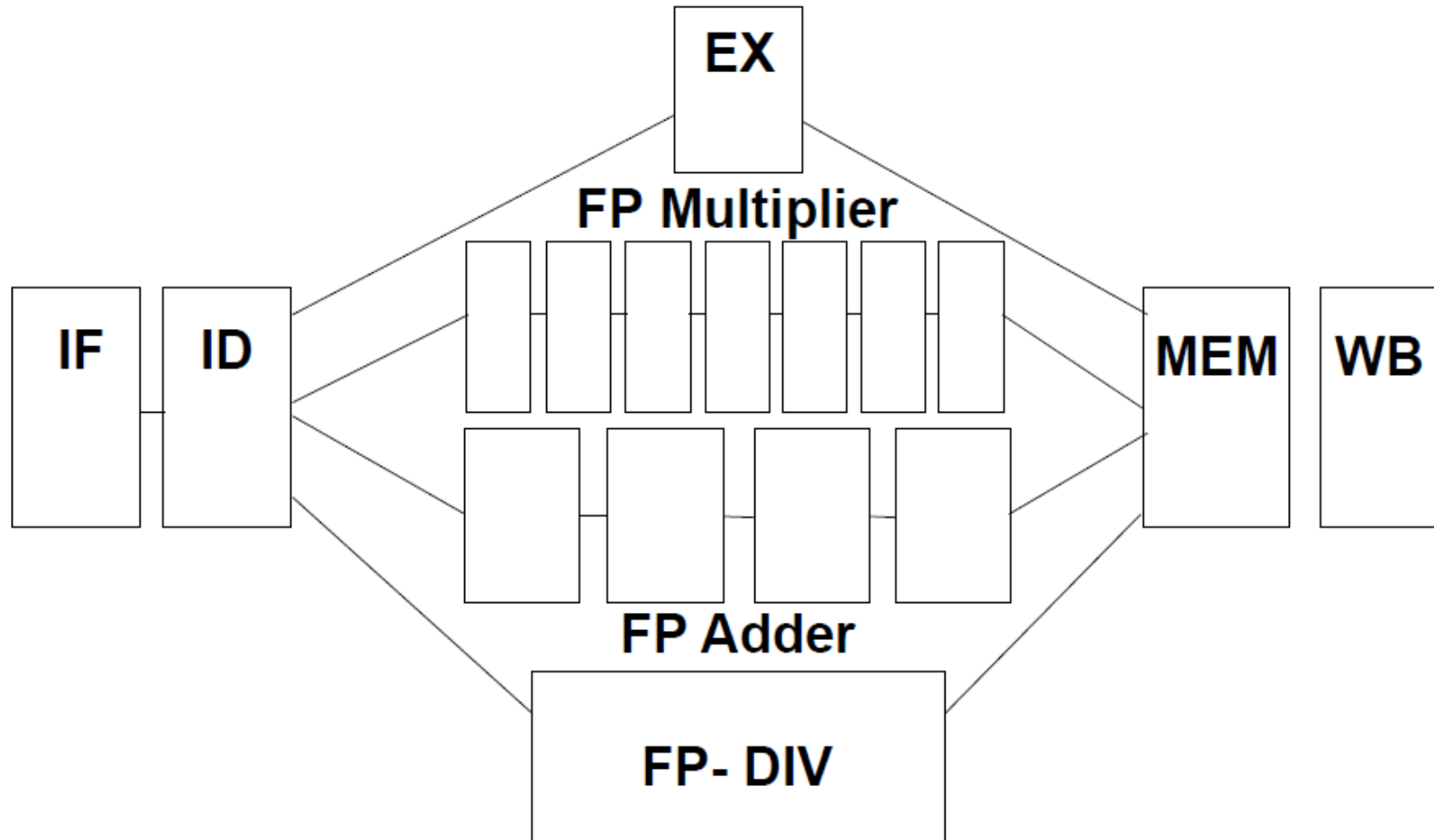
Acceso a memoria (MEM)

- ▶ Si es un acceso a memoria, se accede

Almacenamiento (WB)

- ▶ Se almacena el resultado (*si lo hay*) en el banco de registros

Segmentación en MIPS64



Análisis de la segmentación

Problemas

- No todas las instrucciones necesitan **todas** las etapas
- No todas las etapas pueden ser manejadas en **paralelo**.
(atascos estructurales o de dependencia de datos)
- No se tienen en cuenta los **saltos** de control (atascos de dependencia de control)

Atascos de un cauce (Stall)

Atascos: situaciones que impiden a la siguiente instrucción que se ejecute en el ciclo que le corresponde

► Estructurales

- Provocados por conflictos por los recursos

► Por dependencia de datos

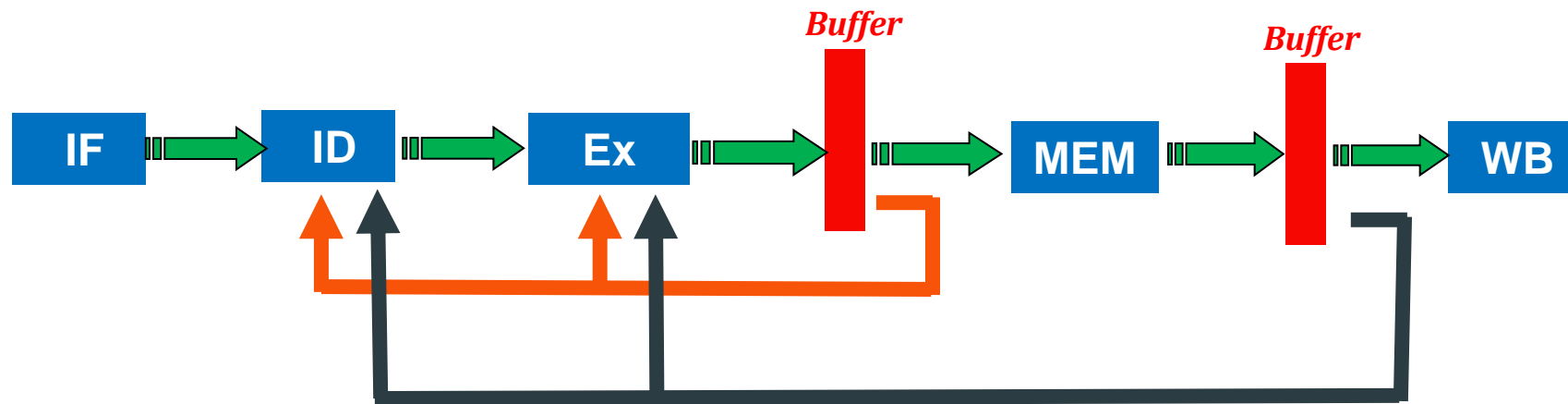
- Ocurren cuando dos instrucciones se comunican por medio de un dato (ej.: una lo produce y la otra lo usa)
 - **RAW:** lectura después de escritura
 - **WAR:** escritura después de lectura
 - **WAW:** escritura después de escritura

► Por dependencia de control

- Ocurren cuando la ejecución de una instrucción depende de cómo se ejecute otra (ej.: un salto y los 2 posibles caminos)

Adelantamiento de operandos (Forwarding)

Si el dato necesario está disponible a la salida de la ALU (E_i) se lleva a la entrada de la etapa correspondiente (E_{i+1}) sin esperar a la escritura (W_i).



WinMIPS64

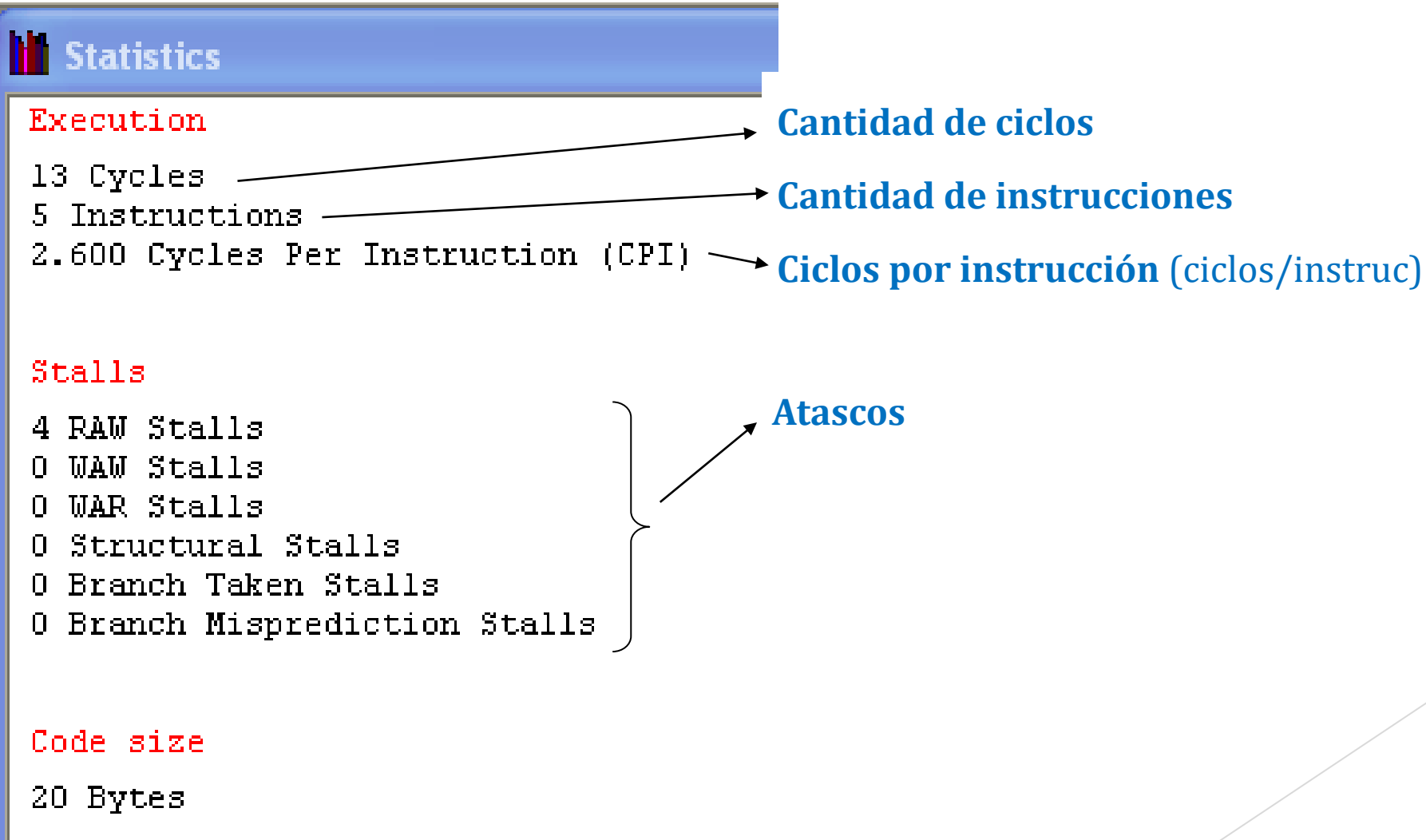
- ***F0 .. F31*** registros de 64 bits → ***punto flotante***
- ***R0 .. R31*** registros de 64 bits → ***enteros***
 - a) DADD R1, R2, R0
 - b) DADDI R3, R0, 5
 - c) DSUB R4, R4, R4
 - d) DADDI R5, R5, -1
 - e) XORI R6, R6, 0xFFFFFFFFFFFFFFFF

WinMIPS64

The image shows the WinMIPS64 MIPS64 Processor Simulator interface with several components and annotations:

- Instrucciones**: Points to the instruction list on the left, showing `ld r4,A(r0)`.
- Pipeline**: Points to the pipeline status window, which shows the instruction `ld` in the `IF` (Instruction Fetch) stage.
- Nombre de registros**: Points to the register names in the `Registers` window.
- valor de registros en hexadecimal**: Points to the hexadecimal values of the registers in the `Registers` window.
- Variables definidas en el programa**: Points to the `Variables` window, which lists variables `A`, `B`, and `C` with their values.
- Información de la ejecución**: Points to the `Execution` window, showing `0 Cycles` and `0 Instructions`.
- Recursos y avance de las instrucciones**: Points to the `Pipeline` window, which shows a diagram of the processor pipeline stages: `IF`, `ID`, `EX` (containing `Multiplier` and `FP Adder`), `MEM`, and `WB`.
- Código**: Points to the `Code` window, showing the assembly code starting with `0000 dc040000 ld r4,A(r0)`.
- Valor de las variables definidas en el programa (hexadecimal)**: Points to the `Data` window, which shows the memory layout with hexadecimal values.

WinMIPS64



The image shows a screenshot of the WinMIPS64 Statistics window. The window has a blue header bar with the title "Statistics" and a small icon of a bar chart. Below the header, the statistics are organized into three sections: "Execution", "Stalls", and "Code size".

Execution

- 13 Cycles → Cantidad de ciclos
- 5 Instructions → Cantidad de instrucciones
- 2.600 Cycles Per Instruction (CPI) → Ciclos por instrucción (ciclos/instruc)

Stalls

- 4 RAW Stalls
- 0 WAW Stalls
- 0 WAR Stalls
- 0 Structural Stalls
- 0 Branch Taken Stalls
- 0 Branch Misprediction Stalls

A bracket groups all the stall types, with an arrow pointing to the label "Atascos".

Code size

- 20 Bytes