# **Guía de Estudio: Arquitectura de Computadoras 2025-2**

**Enfoque:** Implementación de un Procesador RISC Monociclo en VHDL  
**Temario Base:** Unidades I-V del programa de Arquitectura de Computadoras (ISC 2020).

## ****1. Conceptos Clave****

### ****1.1. Procesador RISC Monociclo****

* **Definición:** Ejecuta una instrucción por ciclo de reloj en 5 etapas (FETCH, DECODE, EXEC, MEM, WB).
* **Ventajas:** Simplicidad en el diseño, CPI = 1.
* **Desventajas:** Baja frecuencia de reloj (debido al critical path).

### ****1.1. Componentes Principales****

* **Data Path:**
  + **Registros:** Banco de registros (32 registros en RISC-V).
  + **ALU:** Operaciones aritméticas/lógicas.
  + **Memorias:** ROM (instrucciones), RAM (datos).
  + **Multiplexores:** Selección de operandos (ej: ALUSrc).
* **Unidad de Control (UC):** Genera señales para coordinar componentes.

### ****1.1.1 . Señales de Control****

| **Señal** | **Función** | **Valores típicos** |
| --- | --- | --- |
| RegWrite | Habilita escritura en banco de registros | 1: Escribir, 0: No |
| ALUSrc | Selecciona operando B (registro o inmediato) | 0: Registro, 1: Inmediato |
| MemtoReg | Selecciona dato a escribir en registro | 0: Salida ALU, 1: Memoria |
| Branch | Indica instrucción de salto condicional | 1: Branch activo |

### ****1.1. Procesador Monociclo****

* **Definición:** Ejecuta una instrucción completa en **un solo ciclo de reloj** (CPI = 1).
* **Componentes:**
  + **Data Path Único:** Rutas compartidas para todas las instrucciones.
  + **Unidad de Control (UC):** Genera señales en un único paso.
* **Ventajas:**
  + Simplicidad de diseño.
* **Desventajas:**
  + Baja frecuencia de reloj (critical path largo).
  + Ineficiencia en instrucciones con tiempos desiguales (ej: LW vs ADD).

### ****1.2. Procesador Multicíclico****

* **Definición:** Divide la ejecución en **múltiples ciclos** (CPI ≥ 1).
* **Características:**
  + **Etapas Flexibles:** Cada instrucción usa los ciclos que necesita.
  + **UC Finita (FSM):** Transiciones entre estados para cada instrucción.
* **Ventajas:**
  + Mejor balance entre hardware y rendimiento.
* **Desventajas:**
  + Mayor complejidad que el monociclo.

### ****1.3. Procesador Segmentado (Pipeline)****

* **Definición:** Solapa la ejecución de múltiples instrucciones en **etapas paralelas** (CPI ideal = 1).
* **Etapas Típicas (MIPS):**  
  FETCH → DECODE → EXEC → MEM → WB.
* **Hazards:**
  + **Estructurales:** Recursos compartidos (ej: memoria única).
  + **Datos (RAW):** Dependencias entre instrucciones (soluciones: forwarding, stalling).
  + **Control:** Saltos condicionales (soluciones: predicción estática/dinámica).

## ****2. Implementación en VHDL****

### ****2.1. Monociclo vs Multicíclico****

| **Aspecto** | **Monociclo** | **Multicíclico** |
| --- | --- | --- |
| **CPI** | MC | Variable (depende de la instrucción) |
| **Frecuencia** | Baja (ej: 100 MHz) | Moderada (ej: 200 MHz) |
| **Señales de Control** | Generadas en un paso | Máquina de estados (FSM) |

**Ejemplo UC Multicíclico (VHDL):**

type state\_type is (FETCH, DECODE, EXEC\_R, EXEC\_I, MEM, WB);

signal state : state\_type;

process(clk)

begin

if rising\_edge(clk) then

case state is

when FETCH =>

*-- Cargar instrucción*

state <= DECODE;

when DECODE =>

*-- Decodificar opcode*

if opcode = "0110011" then state <= EXEC\_R; *-- Tipo R*

else state <= EXEC\_I; end if;

when EXEC\_R =>

*-- Operación ALU*

state <= WB;

when WB =>

*-- Escribir en registro*

state <= FETCH;

end case;

end if;

end process;

### ****2.2. Pipeline: Manejo de Hazards****

Tipos de hazards y sus soluciones y manejos posibles. TEORIA.

**3. Métricas de Desempeño**

**3.1. Fórmulas Clave**

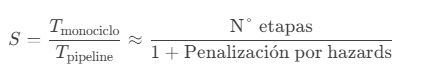
* **CPI (Ciclos por Instrucción):**

CPI=Instrucciones totalesCiclos totales​

* **Tiempo de Ejecución:**

*T*=N° instrucciones×CPI×Periodo de reloj

* **Speed-up (Pipeline vs Monociclo):**



**3.2. Ejemplo Comparativo**

| **Arquitectura** | **CPI** | **Frecuencia** | **Tiempo (100 inst.)** |
| --- | --- | --- | --- |
| Monociclo | MC | 100 MHz | 1000 ns |
| Pipeline (5 etapas) | 1.2\* | 300 MHz | 400 ns |
| \*Incluye penalización por hazards. |  |  |  |

**4. Ejercicios Resueltos**

**4.1. Ejercicio: Pipeline con Hazards**

**Código:**

ADD R1, R2, R3

LW R4, 0(R1)

SUB R5, R4, R6

**Problema:** Identificar hazards y soluciones.  
**Solución:**

* **RAW Hazard** (entre ADD y LW en R1): Usar *forwarding* de EX/MEM a ALU.
* **RAW Hazard** (entre LW y SUB en R4): Insertar *burbuja* (*stall*).

**4.2. Ejercicio: Extensión de Instrucciones**

**Instrucción Nueva:** MUL (Multiplicación).  
**Cambios en VHDL:**

1. **ALU:** Añadir operación de multiplicación.
2. **UC:** Agregar señal ALUOp = "11" para MUL.
3. **Data Path:** Ampliar el banco de registros para guardar resultados de 64 bits.

**5. Problemas Propuestos**

**5.1. Diseño Multicíclico**

Implementar en VHDL un procesador multicíclico que soporte:

* Instrucciones LW, SW, ADD, BEQ.
* Máquina de estados de 5 fases.

**5.2. Análisis de Pipeline**

Para un pipeline de 5 etapas con:

* 20% de branches (50% predicción correcta).
* Penalización por branch = 3 ciclos.  
  Calcular el **CPI efectivo**.

**4.1. Problema: Implementar XORI**

* **Instrucción:**

XORI R1, R2, 0xFF # R1 = R2 XOR 0xFF

* **Tareas:**
  1. Definir el formato de instrucción (tipo I).
  2. Modificar la ALU para soportar XOR.
  3. Actualizar la UC.

**4.2. Problema: Hazard de Datos**

* **Preguntas:**
  1. ¿Qué tipo de hazard ocurre?
  2. Que es la dependencia de datos?
  3. Que puede hacer tu compilador para ayudar?
  4. Propón una solución en VHDL (ej: *forwarding*).

**5. Recursos Adicionales**

* **Herramientas:**
  + **Xilinx Vivado**: Síntesis y simulación.
  + **ModelSim**: Simulación funcional. (Testbench VHDL)

**Bibliografía:**

* + *Computer Organization and Design (Patterson & Hennessy)*.
  + *Digital Design and Computer Architecture (Harris & Harris)*.

**Conclusión**

Esta guía integra:

1. **Arquitecturas** (\*\*\*Monociclo, \*Multicíclico, \*\*Pipeline).
2. **Implementación en VHDL** (UC, Data Path, y/o hazards).
3. **Métricas** (CPI, frecuencia, speed-up).
4. **ALU intermedia – avanzada en VHDL**
5. LIFO en VHDL
6. Interfaces en VHDL procedures, functions, components, port maps.
7. Tipos de instrucciones en básicas en RISC y caracterirtuicas del RISC

**ESTILO EXAMEN ETS**

* **30 a 40% conocimiento teorico, pero embebido en ejercicios**
* **60 a 70% programación en VHDL ejercicios en código escrito a mano.**