***TEMARIO UNIDAD 3 - EXAMEN***

1. Clasificar los siguientes aspectos indicando si están relacionadas con la unidad de control microprogramada o cableada.
   1. Es más rápida. **cableada**
   2. Diseño simplificado. **microprogramada**
   3. Facilita la compatibilidad con máquinas anteriores de la misma familia. **microprogramada**
   4. Es adaptable a la incorporación de nuevas instrucciones. **microprogramada**
   5. Carece de memoria rom. **cableada**
   6. Permite disponer de varios juegos de instrucciones en una misma máquina. **microprogramada**
   7. Es necesario un entorno de desarrollo y compilación para los microprogramas. **Microprogramada**
2. Explicar cómo funciona internamente una unidad de control microprogramada.

**Actúa ejecutando microinstrucciones que definen la funcionalidad de la unidad de control. Se especifica mediante un microprograma, el cual consiste en una secuencia de instrucciones en un lenguaje de microprogramación. Almacena valores de control en la memoria como palabras y un contador genera instrucciones para ejecutar creando una colección de señales en cada pulso de reloj.**

**Es capaz de realizar secuenciamiento de microinstrucciones y generar señales de control para ejecutar cada microinstrucción.**

1. Explicar qué es un registro contador de programa. ¿Existen otros registros que almacenen direcciones de memoria? Si es así, pon un ejemplo.

**Es un registro especial que indica la ubicación de la siguiente instrucción a ejecutar**.

**Registros que existen: stack pointer, return adress, global adress.**

1. ¿Qué clase de unidad de control puede ser modificada sin cambiar el diseño de un procesador?
   1. Unidades de control cableadas y microprogramadas
   2. Unidad de control cableada
   3. **Unidad de control microprogramada**
   4. ninguna
2. Completar el código para realizar un sumador de 32 bits que incluya la generación de banderas ZNCV.

entity Adder is

Port (Input0: in STD\_LOGIC\_VECTOR(31 downto 0);

Input1: in STD\_LOGIC\_VECTOR(31 downto 0);

Result: out STD\_LOGIC\_VECTOR(31 downto 0);

ZNCV: out STD\_LOGIC\_VECTOR(31 downto 0));

end Adder;

architecture Behavioral od Adder is

begin

**-- Generamos las banderas**

**-- Bandera Zero: Se activa si el resultado es cero**

**zero <= '1' when (temp\_result(31 downto 0) = "00000000000000000000000000000000") else '0';**

**-- Bandera Negative: Se activa si el bit más significativo del resultado es 1 (número negativo)**

**negative <= temp\_result(31);**

**-- Bandera Carry: Se activa si hay un acarreo fuera del bit más significativo**

**carry\_out <= temp\_result(32);**

**-- Bandera Overflow: Se activa si hay un desbordamiento (el signo de los operandos es el mismo, pero el resultado tiene signo diferente)**

**overflow <= (Input0(31) = Input1(31)) and (temp\_result(31) /= Input0(31));**

**-- Asignamos las banderas a ZNCV**

**ZNCV <= zero & negative & carry\_out & overflow;**

**-- Realizamos la suma de 32 bits con un bit extra para el acarreo**

**temp\_result <= ("0" & Input0) + ("0" & Input1);**

**-- Asignamos el resultado**

**Result <= temp\_result(31 downto 0);**

end Behavioral;

1. Completar el diseño de banco de registros como el utilizado en el RISC-V (lectura asíncrona, escritura asíncrona con señal de habilitación y registro cero permanentemente en a cero).

entity RegisterFile is

Port ( clk : in STD\_LOGIC;

Adress1: in STD\_LOGIC\_VECTOR(4 downto 0);

Adress2: in STD\_LOGIC\_VECTOR(4 downto 0);

WriteData3: in STD\_LOGIC\_VECTOR(4 downto 0);

WriteEnable3: in STD\_LOGIC\_VECTOR;

ReadData1: out STD\_LOGIC\_VECTOR(31 downto 0);

ReadData2: OUT STD\_LOGIC\_VECTOR(31 downto 0));

end RegisterFile;

architecture Behavioral od RegisterFile is

type store\_regs is array (0 to 31) of std\_logic\_vector(31 downto 0);

signal regs: store\_regs :=(others => x”00000000”)

begin

**-- Lectura asíncrona de los registros**

**ReadData1 <= regs(to\_integer(unsigned(Adress1))); -- Leemos el primer registro**

**ReadData2 <= regs(to\_integer(unsigned(Adress2))); -- Leemos el segundo registro**

**-- Escritura asíncrona con habilitación de escritura**

**process(clk)**

**begin**

**if rising\_edge(clk) then**

**-- Verificamos si la habilitación de escritura es activa y la dirección no es el registro 0**

**if (WriteEnable3 = '1') and (to\_integer(unsigned(Adress1)) /= 0) then**

**regs(to\_integer(unsigned(Adress1))) <= WriteData3; -- Escribimos el dato en el registro correspondiente**

**end if;**

**end if;**

**end process;**

end Behavioral;