KONTROL UNITATEA

```
library IEEE;
use IEEE.std_logic_1164.all;


-- Deskribapen hau ASM algoritmoari dagokio.

entity KU_MAIN is
port (
clk,reset: in std_logic;
MEND, READYOUT, PAUSE, PLAY: in std_logic;
ENABLE, LD_K, COUNT: out std_logic
);
end KU_MAIN;

architecture arki of KU_MAIN is

type EGOERA is (e0,e1,e2,e3,e4);
signal UE,HE: EGOERA;

begin -- arkitekturaren definizioa

process (MEND, PLAY, PAUSE, READYOUT)
begin
case UE is
  when e0 => if PLAY='1' then HE<=e1;
                    else HE<=e0;
           end if;
  when e1 => if READYOUT='1' then HE<= e2;
                                 else HE<=e1;
              end if;
  when e2 => if MEND='0' and PAUSE='0' and READYOUT='0' then HE<= e3;
                    elsif MEND='0' and PAUSE='0' and READYOUT='1' then HE<= e2;
                              elsif MEND='0' and PAUSE='1' then HE<= e3;
                              elsif MEND='1' then HE<= e4;
             end if;
  when e3 => if PLAY='1' and READYOUT='1' then HE<=e2;
                    elsif PLAY='0' and READYOUT='1' then HE<=e3;
                    elsif READYOUT='0' then HE<=e3;
    end if;
  when e4 => HE<=e4;

end case;
end process;

-- prozesu sinkrono batean, uneko egoera (UE) erregistratzen da erloju-ertzero.
-- reset asinkronoa
process (clk,reset)
begin
```

```
if reset='1' then UE<=e0;
        elsif (clk'event and clk='1') then UE<=HE;
end if;
end process;
-- kontrol-seinale bakoitzeko, esleipen konkurrente bat
COUNT <= '1' when (UE=e2) else '0';
LD_K <= '1' when (UE=e0 and PLAY='1') else '0';
ENABLE <= '1' when (UE=e1 or UE=e2) else '0';

end arki; -- arkitekturaren bukaera
```

KONTROL UNITATEA ETA PROZESU UNITATEA

```
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.numeric_std.all;

-- Kronograma eskuz betetzeko erabili dugun ariketa (bigarrena).
-- Diseinuaren helburua: bi zenbaki osoen arteko biderketa egitea
-- Zenbakiak 2rako osagarrian adierazita daude.
-- Diseinua (ASM+PU) eta kronograma betetzeko eskema egelan daude eskura.
-- Deskribapen hau ASM algoritmoari dagokio.

entity MAIN is
port (
clk,reset: in std_logic;
ENABLE: out std_logic;
SAMPLE: out std_logic_vector (15 downto 0);
--MEND: in std_logic;
PLAY, PAUSE, READYOUT: in std_logic;
HELB: in std_logic_vector (15 downto 0)
);
end MAIN;

architecture arki_out of MAIN is

  --erazagupenak: osagaiak eta barne seinaleak
component KU_MAIN
port (
clk,reset: in std_logic;
MEND, READYOUT, PAUSE, PLAY: in std_logic;
ENABLE, LD_K, COUNT: out std_logic
);
end component;

component RAMSINC
port (
clk: in std_logic;
WE: in std_logic;
ADDR, DATIN: in std_logic_vector (15 downto 0);
DATOUT: out std_logic_vector (15 downto 0)
);
end component;

  --signal ENABLE, MEND, READYOUT, PAUSE, PLAY : std_logic;
  signal MEND: std_logic;
  signal LD_K, COUNT: std_logic;
  signal WE: std_logic;
  signal ADDR, DATIN: std_logic_vector (15 downto 0);
  signal DATOUT: std_logic_vector (15 downto 0);
  signal Q_KONT: unsigned (15 downto 0);

begin -- arkitekturaren hasiera
```

```
  ku: KU_MAIN
  port map (
    clk => clk,
    reset => reset, ENABLE => ENABLE,
    MEND => MEND, LD_K => LD_K, READYOUT => READYOUT, PAUSE => PAUSE, PLAY
=> PLAY, COUNT => COUNT
    );

  ram: RAMSINC
  port map (
           clk => clk,
           WE => WE, ADDR => ADDR, DATIN => DATIN, DATOUT => DATOUT
   );

    -- Kontagailua
    process(clk)
    begin
      if clk'event and clk='1' then
        if LD_K='1' then Q_KONT<="0000000000000000";
          elsif COUNT='1' then Q_KONT<= Q_KONT+1;
        end if;
      end if;
    end process;
   MEND<='1' when (Q_KONT="1111111111111111") else '0';


end arki_out;
```