```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE IEEE.numeric_std.all;
--------------------------- Declaring the entity of RAM Circuit
--------------------------
entity data_memory is
port (
 clk,reset       : in std_logic;                            -- clock and reset inputs
 data_in         : in std_logic_Vector(7 downto 0);         -- external 8 bit input da
 input_data      : in std_logic_Vector(7 downto 0);         -- accumulator data input

 mem_write_en    : in std_logic;                            -- control signal input to
write(store) the accumulator data into RAM
 inc_address     : in std_logic;                            -- control signal input to
increment the address of external data input
 mem_read1       : in std_logic;                            -- control signal input to
load data from RAM to A Register
 mem_read2       : in std_logic;                            -- control signal input to
load data from RAM to B Register
 acc_bus         : in std_logic;                            -- control signal input to
load data from accumulator to A Register
 load_ext        : in std_logic;                            -- control signal input to
write external input data from input port to RAM
 mem_address_Breg: in std_logic_vector(2 downto 0);         -- address input register
for to read a data at particular location on RAM for A register
 mem_address_Areg: in std_logic_vector(1 downto 0);         -- address input register
for to read a data at particular location on RAM for B register
 data_out1       : out std_logic_Vector(7 downto 0);        -- 8-bit output for A
register
 data_out2       : out std_logic_Vector(7 downto 0));       -- 8-bit output for B
register
end data_memory;
---------------------------  the Architecure of RAM Circuit
-------------------------------
architecture Behavioral of data_memory is
constant rfill: std_logic := '0';                                  -- declaring
constant to fill the missing bit of mem_address_Areg
signal counter: unsigned(2 downto 0);
signal address_input: unsigned(2 downto 0);
type data_mem is array (0 to 7) of std_logic_vector(7 downto 0);  -- defining the
data memory array
signal RAM: data_mem :=(                                           -- building the RAM
data storage
"00000111", -- x
"00000101", -- y
"00000000", -- w
"00001010", -- z
"00010100", -- v
"00010101", -- f1
"00010111", -- temp
"00000000" ); -- f2
begin

---------------------- Sequential cicuit to Write external input or accumulator into
RAM --------------
process(clk)
begin
if(rising_edge(clk)) then                                  -- when rising edge of
clock is detected
if(mem_write_en='1') then                                  -- check if write signal is
set
ram(to_integer(unsigned(mem_address_Breg))) <= data_in;    -- if mem_write_en is set,
then Accumlator will be written into RAM based on the given operand address
elsif(load_ext ='1') then                                  -- check if load_ext is set
ram(to_integer(unsigned(address_input))) <=input_data; -- if load_ext is set, then
Accumlator will be written into RAM based on the given operand address
end if;
end if;
end process;
---------------------- Sequential cicuit to Load register A from data memory
--------------
process(clk,reset)
begin
if(reset = '1') then                                               -- check if active
reset is set
data_out1 <= x"00";                                                -- if active reset
is set, then Register A will be Zero
elsif(rising_edge(clk)) then                                       -- when rising edge
of clock is detected
if(mem_read1 ='1') then                                            -- check if
mem_read1 is set
data_out1 <= ram(to_integer(unsigned(mem_address_Areg & rfill))); -- if mem_read1 is
set, load data from program memory toward A register
elsif(acc_bus = '1') then
data_out1 <=data_in;
end if;
end if;
end process;
---------------------- Sequential cicuit to Load register B from data memory
--------------
process(clk,reset)
begin
```

```
if(reset='1') then                                                  -- check if active
reset is set
data_out2 <= x"00";                                                 -- if active reset
is set, then Register B will be Zero
elsif(rising_edge(clk)) then                                        -- when rising edge
of clock is detected
if(mem_read2 ='1') then                                             -- check if
mem_read2 is set
data_out2 <= ram(to_integer(unsigned(mem_address_Breg)));           -- if mem_read2 is
set, load data from program memory toward B register
end if;
end if;
end process;
-------------------------------------------------------------------------------------
------
process(clk,reset,counter)
begin
if(reset='1' or counter = "100") then
counter <= "000";
address_input<= "000";
elsif(rising_edge(clk)) then
if(inc_address ='1') then
counter <= counter + 1;
address_input <= unsigned(mem_address_Breg) + counter;
end if;
end if;
end process;
end Behavioral;
```