FUNCTIONALITY OF THE PRELOADING MEMORY FILE

I will explain the functionality of this sample code. You have to customize this based on your requirement. Make sure to know the operation of the keywords like TEXT, line, character, READ_MODE, end file, readline, read, character'pos. This code is used to read the data from the .h file into the VHDL code. Here, we are storing the values into the pre defined arrays of memory bits.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use STD.TEXTIO.all;
use IEEE.NUMERIC_STD_UNSIGNED.all;
```

This is the library declaration. use STD.TEXTIO.all; is the library which supports the file handling operations like FILE_OPEN, TEXT, end file, readline, read etc.

```
entity imem is -- instruction memory
port(a: in STD_LOGIC_VECTOR(5 downto 0);
rd: out STD_LOGIC_VECTOR(31 downto 0));
end;
```

Input and Output Ports declaration in Entity.

```
architecture behave of imem is
begin
process is
file mem_file: TEXT;
variable L: line;
variable ch: character;
variable i, index, result: integer;
type ramtype is array (63 downto 0) of STD_LOGIC_VECTOR(31 downto 0);
variable mem: ramtype;
begin
```

Variables of the type TEXT, line, character is declared and memory array is declared.

```
-- initialize memory from file

for i in 0 to 63 loop -- set all contents low
mem(i) := (others => '0');
end loop;
```

index := 0;
Memory Initialization to zero initially using for loop

```
FILE_OPEN (mem_file, "memfile.dat", READ_MODE);
while not endfile(mem_file) loop
readline(mem_file, L);
result := 0;
```

Opening the file and reading the line until the last line is read.

```
for i in 1 to 8 loop
        read (L, ch);
        if '0' <= ch and ch <= '9' then
        result := character'pos(ch) - character'pos('0');
        elsif 'a' <= ch and ch <= 'f' then
        result := character'pos(ch) - character'pos('a')+10;
        else report "Format error on line" & integer'
        image(index) severity error;
        end if;
        mem(index)(35-i*4 downto 32-i*4) := to_std_logic_vector(result,4);
        end loop;
        index := index + 1;
end loop;
```

reading the character in the line one by one and checking whether the character falls between 0 and f in hexadecimal including 0 and f. Please make sure to give the appropriate hexadecimal value in the imem.h file. Hexadecimal value should be equivalent to the 32 bits value. Therefore, make a 32 bit value and convert that value to the hexadecimal value. Enter that hexadecimal value in the imem.h file. If you have the 4 digits of hexadecimal value, then make sure to fill the rest of the four digits with zeros or else you will be reading the erroneous value.

```
-- read memory
loop
rd <= mem(to_integer(a));
wait on a;
end loop;
end process;
end;
```

This part is about reading the corresponding memory based on the address pointed based on the input port a.
