```
 1  module IDM(A, WD, clk, WE, out);
 2   input [7:0] WD;
 3   input[7:0] A;
 4   input clk, WE;
 5   output [15:0] out;
 6   reg [15:0] data[63:0]; // 2^6 memory slots
 7
 8   initial begin
 9      data[0]<=16'b10_001_01100001010;
10      data[1]<=16'b10_001_01100010101;
11      data[2]<=16'b10_001_01100011000;
12      data[3]<=16'b11_000_01000000000;
13
14
15      data[10]<=16'b11_000_010_00000001; // LDD R2, #1 ; load immediate numbers to the
    registers
16      data[11]<=16'b11_000_011_00000100; // LDD R3, #4
17      data[12]<=16'b11_000_100_00000011; // LDD R4, #3
18      data[13]<=16'b11_000_101_00000011; // LDD R5, #3
19      data[14]<=16'b11_000_111_00000011; // LDD R7, #3
20      data[15]<=16'b00_000_010_010_011_00; // ADD R2, R2, R3
21      data[16]<=16'b00_000_010_010_100_00;
22      data[17]<=16'b00_000_010_010_101_00;
23      data[18]<=16'b00_000_010_010_111_00;
24      data[19]<=16'b10_001_01100000001;
25
26      data[21]<=16'b11_00001000101010;
27      data[22]<=16'b00_00101000101000;
28      data[23]<=16'b10_00101100000010;
29
30      data[24]<=16'b11_00001000000000;
31      data[25]<=16'b11_00001100000000;
32      data[26]<=16'b11_00001000000111;
33      data[27]<=16'b11_00001100000111;
34      data[28]<=16'b01_01001001000000;
35      data[29]<=16'b01_01001001000000;
36      data[30]<=16'b01_01001001000000;
37      // to be continued
38
39   end
40
41   always @(posedge clk)begin
42    if(WE) data[A[7:0]]<= WD;
43   end
44
45   assign out = data[A[7:0]];
46  endmodule
47
```