Explain your ROM for mapping 5-bit binary to 2-digit BCDs (or 2x8 bits seven segment displays depending on your design in Exercise.2).

เลือกใช้วิธีการแปลงจาก 5-bit binary เป็น 2 digit BCDs

```
1  module rom(
2      output wire [7:0]d,
3      input wire [4:0]addr,
4      input wire clk
5  );
6
7  parameter bits = 5;
8  reg [7:0] rom[0:(2**bits)-1];
9  // NOTE: To infer combinational logic instead of a ROM, use
10 // (* synthesis, logic_block *)
11 initial $readmemb("rom.data", rom);
12 assign d = rom[addr];
13 //always @(posedge clk) $display("rom %b", d);
14
15 endmodule
```

เมื่อเริ่มต้นทำงาน โปรแกรมจะไปดึงข้อมูลจาก memory ด้วยคำสั่ง readmemb ซึ่งจะอ่านข้อมูลเป็น binary และจะเก็บข้อมูลไว้ที่ reg rom

เมื่อเรารับ address เข้ามาที่มี 5 บิต คำสั่ง assign จะทำการต่อ wire d ไปที่ reg rom ที่ index นั้น จากบรรทัดที่ 12

เนื่องจาก address มีทั้งหมด 5 บิต ดังนั้น ข้อมูลจึงที่มากสุดที่ 32 บรรทัด บรรทัดละ 8 บิต

ข้อมูลใน rom.data

```
1  00000000 //addr = 0
2  00000001 //addr = 1
3  00000010 //addr = 2
4  00000011 //addr = 3
5  00000100 //addr = 4
6  00000101 //addr = 5
7  00000110 //addr = 6
8  00000111 //addr = 7
9  00001000 //addr = 8
10 00001001 //addr = 9
11 00010000 //addr = 10
12 00010001 //addr = 11
13 00010010 //addr = 12
```

```
00010011 //addr = 13
00010100 //addr = 14
00010101 //addr = 15
00010110 //addr = 16
00010111 //addr = 17
00011000 //addr = 18
00011001 //addr = 19
00100000 //addr = 20
00100001 //addr = 21
00100010 //addr = 22
00100011 //addr = 23
00100100 //addr = 24
00100101 //addr = 25
00100110 //addr = 26
00100111 //addr = 27
00101000 //addr = 28
00101001 //addr = 29
00110000 //addr = 30
00110001 //addr = 31
```