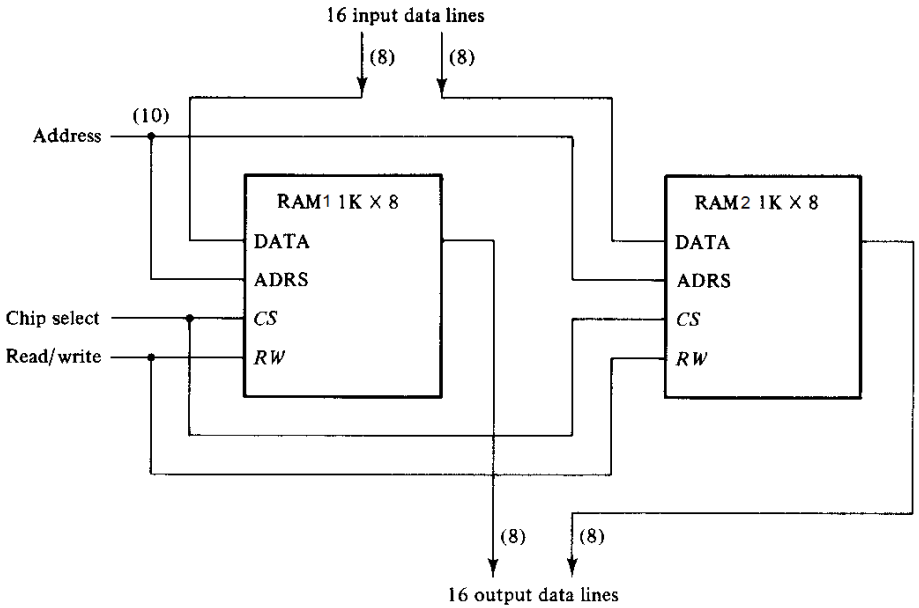
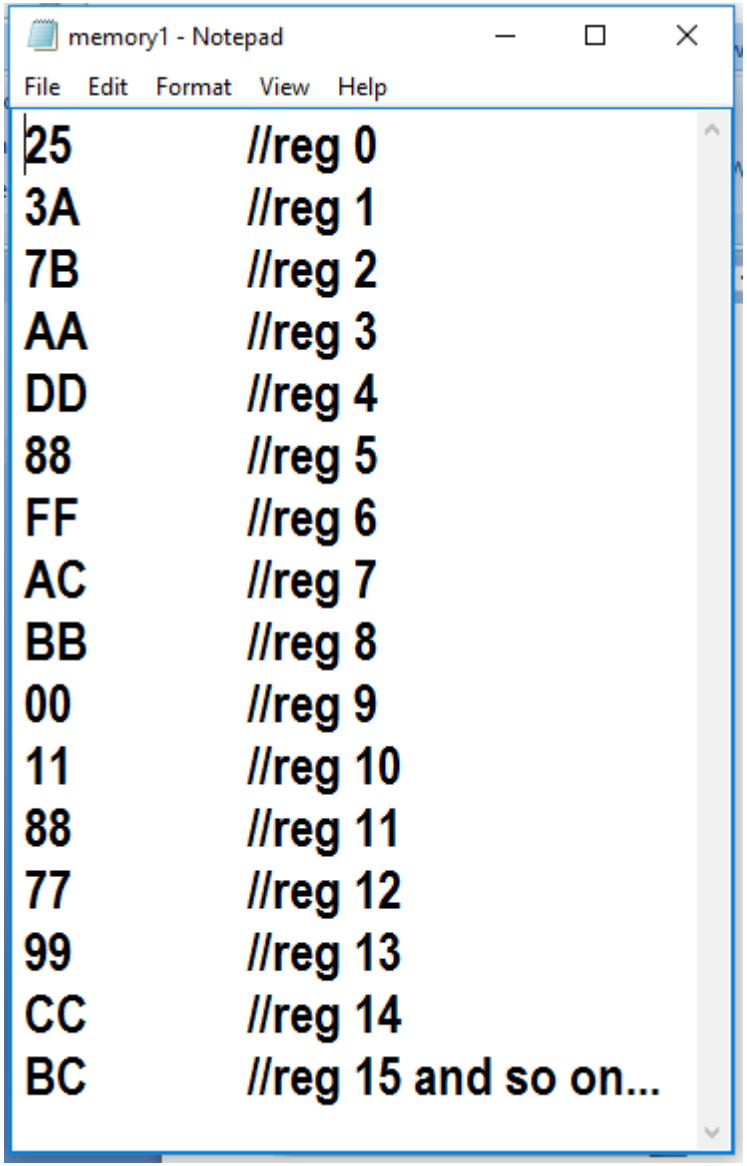


FIGURE
Block diagram of a 1K x 16 RAM.

Figure above shows the interconnection of two 1Kx8 RAM chips to form a 1Kx16 memory. The 16 input and output data lines are split between the two chips. Both receive the same 10-bit address and the common CS and RW control inputs.

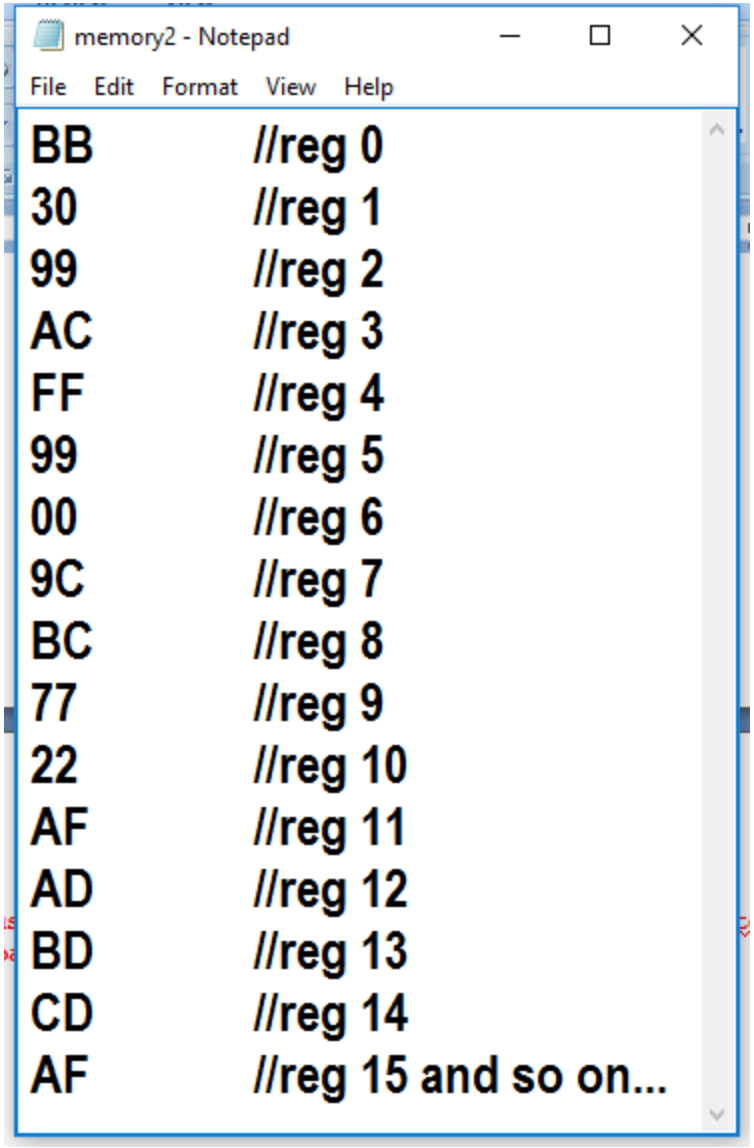
```
module Memory_1Kx16 (addr, CS, RW, idata, odata);

     input CS, RW;
     input [9:0] addr;
     input [15:0] idata;
     output [15:0] odata;

     wire [7:0] od1, od2;

     RAM1 rc1 (addr, CS, RW, idata [15:8], od1);
     RAM2 rc2 (addr, CS, RW, idata [7:0], od2);

     assign odata = (CS && RW)?{od1, od2}:16'bz;

endmodule

module RAM1 (addr, CS, RW, idata, odata);

     input CS, RW;
     input [9:0] addr;
     input [7:0] idata;
     output [7:0] odata;
     reg [7:0] d_out;
     reg [7:0] Mem1 [0:1023];

     assign odata = (CS && RW)?d_out:8'b0;

     always @(addr or idata or CS or RW)
               if (CS && !RW)
                    Mem1 [addr] = idata;
          always @(addr or CS or RW)
               if (CS && RW)
                    d_out = Mem1 [addr];

        initial
               $readmemh ("memory1.dat", Mem1);

endmodule
```

```
//memory1.dat. The data file must be in the same directory as design and testbench. Otherwise you
will need to specify the complete path.
```



```
module RAM2 (addr, CS, RW, idata, odata);

     input CS, RW;
     input [9:0] addr;
     input [7:0] idata;
     output [7:0] odata;
     reg [7:0] d_out;
     reg [7:0] Mem2 [0:1023];

     assign odata = (CS && RW)?d_out:8'b0;

     always @(addr or idata or CS or RW)
               if (CS && !RW)
                    Mem2 [addr] = idata;
          always @(addr or CS or RW)
               if (CS && RW)
                    d_out = Mem2 [addr];

        initial
               $readmemh ("memory2.dat", Mem2);

endmodule
```

```
//memory2.dat. The data file must be in the same directory as design and testbench. Otherwise you
will need to specify the complete path.
```



```
module tst_Memory_1Kx16 ();

      reg [9:0] address;
      reg CE, RW;
      reg [15:0] data_in;
      wire [15:0] data_out;

      Memory_1Kx16 m1 (
         .addr(address),
         .CS (CE),
         .RW(RW),
         .idata(data_in),
         .odata(data_out));

      initial begin
               $display("Reading from Memory (already initialized/populated using $readmemh)...");

                CE = 0;
                RW = 0;
                #5 address = 15;
                #5 RW = 1;
                #5 CE = 1;
                #10 address = 1;
                #10 address = 2;
                #10 address = 3;
                #10 address = 7;
                #10 address = 6;
                #10 address = 5;
                #10 address = 4;
                #10 address = 8;
                #10 address = 9;
                #10 address = 10;
                #10 address = 11;
                #10 address = 0;
                #10 address = 14;
                #10 address = 13;
                #10 address = 12;
```

```
            #500 $finish;
    end

endmodule
```