

من سوالات ۶ و ۷ رو انتخاب کردم.

(سوال ۶)

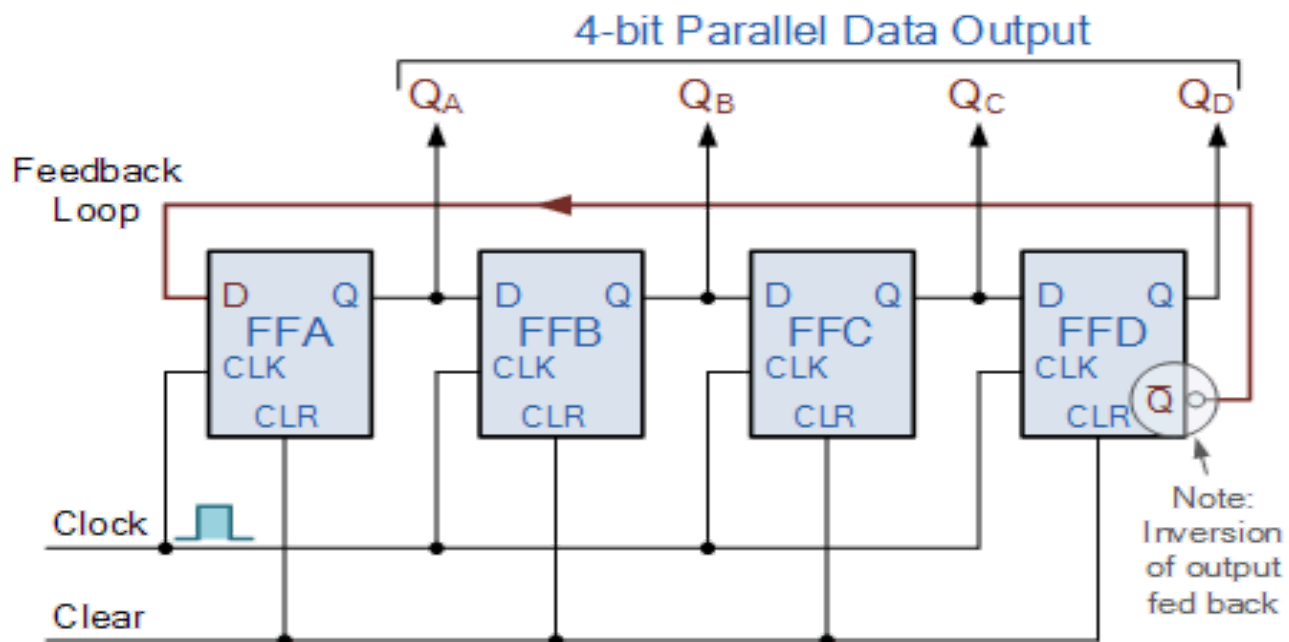
من ابتدا مدار D - flip flop را طراحی کردم که کد آن به شکل زیر است:

```

1 module D_FlipFlop(input D,clk,rst,output reg Q );
2     always @(posedge clk or negedge rst) begin
3         if(!rst)
4             Q<=0;
5         else
6             Q<=D;
7     end
8 endmodule

```

سپس با استفاده از آن شمارنده جانسون را طبق مدار زیر طراحی کردم:



کد این شمارنده به شکل زیر است:

```
9 module Johnson_counter #(parameter N=4)(input rst,clk,output [N-1:0]Q);
10     genvar ii;
11     generate
12         for (ii=0;ii<N;ii=ii+1) begin
13             if(ii==0) begin
14                 D_FlipFlop d(~Q[N-1],clk,rst,Q[ii]);
15             end
16             else begin
17                 D_FlipFlop d(Q[ii-1],clk,rst,Q[ii]);
18             end
19         end
20     endgenerate
21 endmodule
```

سپس طبق خواسته سوال برای آن تست بنچ نوشتم:

برای $N=4$:

```
21 endmodule
22 module TB;
23     parameter N=4;
24     reg rst;
25     reg clk;
26     wire [N-1:0]Q;
27     always begin
28         #5 clk=~clk;
29     end
30     Johnson_counter #(.N(N)) sample (rst,clk,Q);
31     initial begin
32         clk=0;
33         rst<=0;
34         #2 rst<=1;
35         #500 $finish;
36     end
37     initial begin
38         $monitor("%b", Q);
39     end
40     initial begin
41         $dumpfile("Johnson_counter.vcd");
42         $dumpvars(0,TB);
43     end
44 endmodule
```

که خروجی آن به شکل زیر است:

```
0000
0001
0011
0111
1111
1110
1100
1000
0000
0001
0011
0111
1111
1110
1100
1000
0000
0001
0011
0111
1111
1110
1100
1000
0000
0001
0011
0111
1111
1110
```

برای $N=8$:

```
00000000
00000001
00000011
00000111
00001111
00011111
00111111
01111111
11111111
11111110
11111100
11111000
11110000
11100000
11000000
10000000
00000000
00000001
00000011
00000111
00001111
00011111
01111111
11111111
11111110
11111100
11111000
11110000
11100000
11000000
```

برای $N=16$:

```
0000000000000000
0000000000000001
0000000000000011
0000000000000111
0000000000001111
0000000000011111
0000000001111111
0000000111111111
0000001111111111
0000011111111111
0000111111111111
0001111111111111
0011111111111111
0111111111111111
1111111111111111
1111111111111110
1111111111111100
1111111111111000
1111111111100000
1111111110000000
1111111100000000
1111111000000000
1111110000000000
1111100000000000
1111000000000000
1110000000000000
```

برای $N=32$:

```
VCD info: dumpfile Johnson_counter.vcd opened for output.
```

000000000000000000000000000000000000

[illegible]

000000000000000000000000000000000011

00000000000000000000000000000000111

000000000000000000000000000000001111

000000000000000000000000000000001111

00000000000000000000000000000000111111

0000000000000000000000000000111111

0000000000000000000000000000000011111111

0000000000000000000000000011111111

000000000000000000000000000011111111

0000000000000000000000001111111111

00000000000000000000111111111111

00000000000000000000111111111111

00000000000000000011111111111111

000000000000000000111111111111111

00000000000000001111111111111111

00000000000000001111111111111111

00000000000000001111111111111111

[illegible][illegible][illegible][illegible][illegible]

000000001111111111111111111111

[illegible][illegible][illegible]

کد این سوال در پیوست خدمتون تقدیم میشود.

سوال ۷)

برای این سوال ابتدا طبق خواسته های سوال من به ترتیب (از سمت چپ)
RegisterFile, ALU, Memory , Processor را طراحی کردم که کد آنها به همراه توضیحاتشان به شرح زیر است:

برای

برای RegisterFile من سیگنال های کنترلی برای اینکه شروع به نوشتن کنیم و از کدام رجیسترها بخوانیم و بنویسیم، برای کلاک،
تعریف کردم و همچنین ورودی و خروجی هایی برای نوشتن و خواندن دیتا.

```
1 module RegisterFile (  
2     input wire clk,  
3     input wire [1:0] read_reg1, read_reg2, write_reg1, write_reg2,  
4     input wire [511:0] write_data1, write_data2,  
5     input wire reg_write_enable1, reg_write_enable2,  
6     output reg [511:0] read_data1, read_data2  
7 );  
8     reg [511:0] registers [0:3];  
9     always @(posedge clk) begin  
10         if(reg_write_enable1)  
11             registers[write_reg1] <= write_data1;  
12         if(reg_write_enable2)  
13             registers[write_reg2] <= write_data2;  
14     end  
15     always @(*) begin  
16         read_data1 = registers[read_reg1];  
17         read_data2 = registers[read_reg2];  
18     end  
19 endmodule
```

برای ALU، سیگنال کلاک، دو سیگنال کنترلی برای مشخص کردن دستور و دو ورودی و دو خروجی برای ورودی دادن اطلاعات
و گرفتن خروجی عملیات تعریف کردم:

```
20 module ArithmeticUnit (  
21     input wire [511:0] A1, A2,  
22     input wire [1:0] operation, // 00: no-op, 01: addition, 10: multiplication  
23     output reg [511:0] A3, A4  
24 );  
25     always @(*) begin  
26         case (operation)  
27             2'b10: begin  
28                 // A4=512'b0;  
29                 {A4, A3} = A1 + A2;  
30             end  
31             2'b11: begin  
32                 {A4, A3} = A1 * A2;  
33             end  
34             default: begin  
35                 A3 = 0;  
36                 A4 = 0;  
37             end  
38         endcase  
39     end  
40 endmodule
```

\\
\\

برای Memory سیگنال کلاک، سیگنال فعال کردن توانایی نوشتن، ورودی دادن آدرس، ورودی دادن دیتا و خروجی گرفتن دیتا تعریف کردم:

```
41 module Memory (
42     input wire clk,
43     input wire mem_write_enable,
44     input wire [8:0] mem_addr,
45     input wire [511:0] mem_write_data,
46     output reg [511:0] mem_read_data
47 );
48     reg [31:0] memory [0:511];
49
50     integer i;
51
52     always @(posedge clk) begin
53         if (mem_write_enable) begin
54             for (i = 0; i < 16; i = i + 1) begin
55                 memory[mem_addr + i] <= mem_write_data[(i*32) +: 32];
56             end
57         end
58     end
59
60     always @(*) begin
61         for (i = 0; i < 16; i = i + 1) begin
62             mem_read_data[(i*32) +: 32] = memory[mem_addr + i];
63         end
64     end
65 endmodule
```

حالا میرسیم به اسمبل کردن این مدول ها و طراحی پردازنده:

من از بیرون سیگنال دستور، سیگنال برای اینکه روی کدام رجیسترها بنویسم و بخوانم، سیگنال آدرس، سیگنال فعال کردن نوشتن روی ورودی، سیگنال ورودی دادن دیتا برای نوشتن در حافظه، (توانایی نوشتن مستقیم بر روی رجیسترها نداریم)، سیگنال اینکه آیا باید دیتای روی حافظه از بیرون نوشته شود یا (control)، سیگنال فعال کردن نوشتن روی رجیسترها و همچنین چهار سیگنال برای اینکه ببینم نتایج عملیات ها موفقیت آمیز بوده یا نه تعریف کردم:


```

66 module Processor (
67     input wire clk,
68     input wire [1:0] instruction, // 00: load, 01: store, 10: add, 11: multiply
69     input wire [1:0] reg_select_read1, reg_select_read2, reg_select_writel, reg_select_write2,
70     input wire [8:0] mem_addr,
71     input wire [511:0] mem_write_data, // input of processor
72     input wire control,
73     input wire reg_write_enable1,
74     input wire reg_write_enable2,
75     output wire [511:0] A1, A2, A3, A4 // for checking content of A1, A2, A3, A4
76 );
77     wire [511:0] alu_result1, alu_result2;
78     wire [511:0] reg_read_data1, reg_read_data2;
79     wire [511:0] mem_read_data;
80     RegisterFile rf (
81         .clk(clk),
82         .read_reg1(reg_select_read1),
83         .read_reg2(reg_select_read2),
84         .write_reg1(reg_select_writel),
85         .write_reg2(reg_select_write2),
86         .write_data1((instruction == 2'b00) ? mem_read_data : alu_result1),
87         .write_data2((instruction == 2'b00) ? 512'bz : alu_result2),
88         .reg_write_enable1(reg_write_enable1),
89         .reg_write_enable2(reg_write_enable2),
90         .read_data1(reg_read_data1),
91         .read_data2(reg_read_data2)
92     );
93     ArithmeticUnit alu (
94         .A1(reg_read_data1),
95         .A2(reg_read_data2),
96         .operation(instruction[1:0]),
97         .A3(alu_result1),
98         .A4(alu_result2)
99     );
100     Memory mem (
101         .clk(clk),
102         .mem_write_enable(instruction == 2'b01),
103         .mem_addr(mem_addr),
104         .mem_write_data((control==1)? mem_write_data:reg_read_data1),
105         .mem_read_data(mem_read_data)
106     );
107     assign A1 = reg_read_data1;
108     assign A2 = reg_read_data2;
109     assign A3 = alu_result1;
110     assign A4 = alu_result2;
111
112 endmodule

```

حال میرسیم به تست بنچ نوشتن:

متأسفانه نشد که کل کد را در یک صفحه جا دهم. لطفاً به کد الصاقی مراجعه کنید:

خلاصه تست بنچی که نوشتیم اینه که ابتدا در دو خانه متفاوت حافظه دو دیتای متفاوت نوشته ام و سپس این دو دیتا را از حافظه روی رجیسترهای A1, A2 (که چون وریلاگ صفر بیس است، یک واحد از شماره رجیسترها کم میشود) سپس این دیتاها را به ALU میدهیم و سپس برای اینکه مطمئن شوم که پردازنده به درستی کار میکند این دو خروجی را روی حافظه نوشتیم و دوباره میخوانیم که کد ورودی و خروجی گرفتیم را میتوانیم در زیر ببینیم:

این بلاک برای TestBench1 است: جمع دو عدد ۲ و ۴ :

```
144 initial begin
145     reg_write_enable1 = 0;
146     reg_write_enable2 = 0;
147     clk = 0;
148     instruction = 2'b01;
149     mem_addr = 9'b0;
150     control = 1;
151     mem_write_data = 512'b10; // 2 in binary
152     #10
153     mem_addr = 9'b010000000; // 128 in binary
154     mem_write_data = 512'b100; // 4 in binary
155     #10
156     control = 0;
157     instruction = 2'b00;
158     mem_addr = 9'b0;
159     reg_select_writel = 2'b00;
160     reg_write_enable1 = 1;
161     #10
162     mem_addr = 9'b010000000;
163     reg_select_writel = 2'b01;
164     #10
165     reg_write_enable1 = 0;
166     reg_select_read1 = 2'b00;
167     reg_select_read2 = 2'b01;
168     #10
169     instruction = 2'b10; // add
170     #50
171     $display("A2= %d\n A1= %d", A2, A1);
172     $display("A4= %d\n A3= %d", A4, A3);
173     reg_select_writel=2'b10;
174     reg_select_write2=2'b11;
175     reg_write_enable1=1;
176     reg_write_enable2=1;
177     #10
178     instruction=2'b01;
179     reg_select_read1=2'b10;
180     mem_addr=9'b0;
181     #10
182     reg_select_read1=2'b11;
183     mem_addr = 9'b010000000;
184     #10
185     instruction = 2'b00;
186     mem_addr = 9'b0;
187     reg_select_writel = 2'b00;
188     reg_write_enable1 = 1;
189     #10
190     mem_addr = 9'b010000000;
191     reg_select_writel = 2'b01;
192     #10
193     reg_write_enable1 = 0;
194     reg_select_read1 = 2'b00;
195     reg_select_read2 = 2'b01;
196     #10
197     $display("A1= %d\n A2= %d", A1, A2);
198     #20 $finish;
199 end
```

entering them in simulation using question 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000

A2=	4
A1=	2
A4=	0
A3=	6
A1=	6
A2=	0

حالا من مدار را برای جمع -1 و دو به توان ۵۱۱ منهای یک حساب میکنم که همانطور که میبینید به درستی اورفلو را در A4 میریزد و مقدار درست را در A3 میریزد: که در ۵۱۲ بیت جا میشود و عدد صفر مینویسم که همانطور که میبینید برای این کار هم به درستی خروجی میدهد:

TestBench2:

```
232 initial begin
233     reg_write_enable1 = 0;
234     reg_write_enable2 = 0;
235     clk = 0;
236     instruction = 2'b01;
237     mem_addr = 9'b0;
238     control = 1;
239     mem_write_data = {1'b0, {511{1'b1}}};
240     #10
241     mem_addr = 9'b010000000;
242     mem_write_data = {512{1'b1}};
243     #10
244     control = 0;
245     instruction = 2'b00;
246     mem_addr = 9'b0;
247     reg_select_writel = 2'b00;
248     reg_write_enable1 = 1;
249     #10
250     mem_addr = 9'b010000000;
251     reg_select_writel = 2'b01;
252     #10
253     reg_write_enable1 = 0;
254     reg_select_read1 = 2'b00;
255     reg_select_read2 = 2'b01;
256     #10
257     instruction = 2'b10; // add
258     #50
259     $display("A2= %b\n A1= %b", A2, A1);
260     $display("A4= %b\n A3= %b", A4, A3);
261     #20 $finish;
262 end
263 endmodule
```

خروجی کد :

[illegible]

حالت مدار را برای جمع بزرگترین عدد مثبت و گزینه آن حساب میکنیم که به درستی عدد صفر را در A3 و اورفلو را در A4 میریزد:

TestBench 5

```

initial begin
    reg_write_enable1 = 0;
    reg_write_enable2 = 0;
    clk = 0;
    instruction = 2'b01;
    mem_addr = 9'b0;
    control = 1;
    mem_write_data = {1'b0,{511{1'b1}}};
#10
    mem_addr = 9'b010000000;
    mem_write_data = ~(1'b0,{511{1'b1}});
#10
    control = 0;
    instruction = 2'b00;
    mem_addr = 9'b0;
    reg_select_write1 = 2'b00;
    reg_write_enable1 = 1;
#10
    mem_addr = 9'b010000000;
    reg_select_write1 = 2'b01;
#10
    reg_write_enable1 = 0;
    reg_select_read1 = 2'b00;
    reg_select_read2 = 2'b01;
#10
    instruction = 2'b10; // add
#50
    $display("A2= %b\n A1= %b", A2, A1);
    $display("A4= %b\n A3= %b", A4, A3);
#20 $finish;
end
endmodule

```

خروجی:

[illegible]

حالا میخواهیم عملکرد مدار را برای ضرب دو عدد مورد بررسی قرار دهیم:

برای ضرب 10 و 100 :

برای فهم راحت تر اعداد را بر مبنای 10 مینویسم: TestBench3

```

295 initial begin
296     reg_write_enable1 = 0;
297     reg_write_enable2 = 0;
298     clk = 0;
299     instruction = 2'b01;
300     mem_addr = 9'b0;
301     control = 1;
302     mem_write_data = 512'd10;
303     #10
304     mem_addr = 9'b010000000;
305     mem_write_data = 512'd100;
306     #10
307     control = 0;
308     instruction = 2'b00;
309     mem_addr = 9'b0;
310     reg_select_writel = 2'b00;
311     reg_write_enable1 = 1;
312     #10
313     mem_addr = 9'b010000000;
314     reg_select_writel = 2'b01;
315     #10
316     reg_write_enable1 = 0;
317     reg_select_read1 = 2'b00;
318     reg_select_read2 = 2'b01;
319     #10
320     instruction = 2'b11; // multiply
321     #50
322     $display("A2= %d\n A1= %d", A2, A1);
323     $display("A4= %d\n A3= %d", A4, A3);
324     #20 $finish;
325 end

```

که همانطور که میبینیم به درستی حاصل که برابر ۱۰۰۰ است را در رجیستر A3 ریخته و A4 را چون ۱۰۰۰ در A3 جا میشد، با صفر مقداردهی کرده است.

```
A0=                                1  
A1=                               10  
A2=                              100  
A3=                             1000  
A4=                            10000
```

