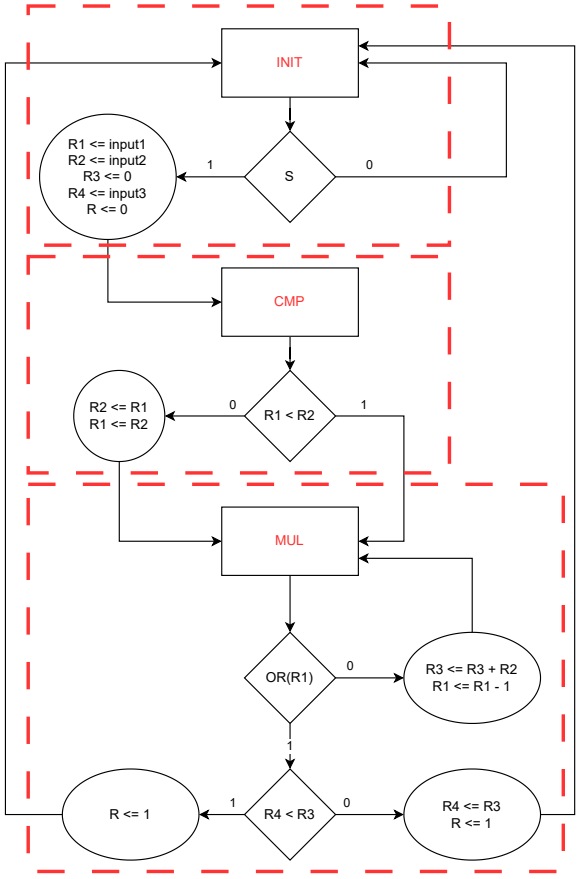
الف)



ب) واحدهای مورد نیاز برای این طراحی به صورت زیر هستند:

1) رجیسترهای ۳۲ بیتی برای R1, R2, R3, R4 با قابلیت reset, inc, dec
2) فلیپ‌فلاپ تک‌بیتی برای R = Ready
3) مقایسه کننده ۳۲ بیتی
4) جمع کننده ۳۲ بیتی
5) OR ۳۲ بیتی
6) MUX دو ورودی

پ) توصیف رفتاری این ماژول‌ها در وریلاگ را در زیر مشاهده می‌کنید:

1) رجیستر ۳۲ بیتی:

```
module reg32 (
    input wire clk,
    input wire reset,
    input wire inc,
    input wire dec,
    input wire [31:0] d,
    output reg [31:0] q
);
```

```
    always @(posedge clk or posedge reset) begin
        if (reset) begin
            q <= 32'b0;
        end else begin
            if (inc) begin
                q <= q + 1;
            end else if (dec) begin
                q <= q - 1;
            end else begin
                q <= d;
            end
        end
    end

endmodule
```

2) فلیپ‌فلاپ تک‌بیتی

```
module dff (
    input wire clk,
    input wire reset,
    input wire d,
    output reg q
);

    always @(posedge clk or posedge reset) begin
        if (reset) begin
            q <= 1'b0;
        end else begin
            q <= d;
        end
    end

endmodule
```

3) مقایسه کننده ۳۲ بیتی

```
module comparator32 (
    input wire [31:0] a,
    input wire [31:0] b,
    output wire gt,
    output wire eq,
    output wire lt
);

    assign gt = (a > b);
    assign eq = (a == b);
    assign lt = (a < b);

endmodule
```

4) جمع کننده ۳۲ بیتی

```
module adder32 (
    input wire [31:0] a,
    input wire [31:0] b,
    output wire [31:0] sum,
    output wire carry_out
);

    assign {carry_out, sum} = a + b;

endmodule
```

5) OR ۳۲ بیتی

```
module wide_or (
    input wire [31:0] in,
    output wire out
);

    assign out = |in;

endmodule
```

6) MUX ۳۲ بیتی

```
module mux32 (
    input wire [31:0] in0,
    input wire [31:0] in1,
    input wire sel,
    output wire [31:0] out
);

    assign out = sel ? in1 : in0;

endmodule
```

با توجه به این ماژول‌ها و ASM chart رسم شده در ابتدا، ماژول datapath به صورت زیر طراحی می‌شود:

```
`include "adder32.v"
`include "comparator32.v"
`include "dff.v"
`include "mux32.v"
`include "reg32.v"
`include "wide_or.v"
`include "control_unit.v"

module datapath(
    input clk,
    input start,
    input [31:0] A,
    input [31:0] B,
    input [31:0] C,
    output out_r,
    output [31:0] out
);

wire dec_r1, reset_r3;
```

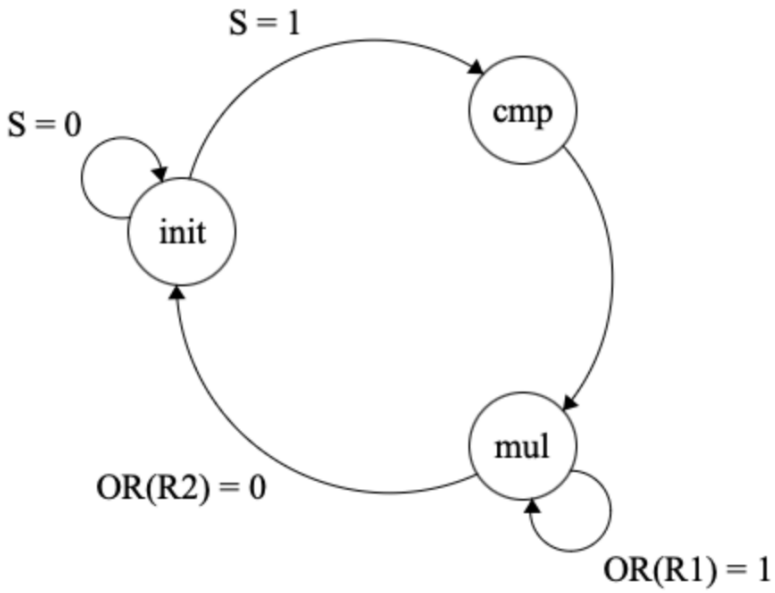
```
wire [31:0] R1, R2, R3, R4;
wire [31:0] out_mux_r1, out_mux_r2, out_mux_r4, out_adder;

reg32 reg_r1 (clk, 1'b0, 1'b0, dec_r1, out_mux_r1, R1);
reg32 reg_r2 (clk, 1'b0, 1'b0, 1'b0, out_mux_r2, R2);
reg32 reg_r3 (clk, reset_r3, 1'b0, 1'b0, out_adder, R3);
reg32 reg_r4 (clk, 1'b0, 1'b0, 1'b0, out_mux_r4, R4);

dff R (clk, 1'b0, in_r, out_r);

mux32 mux_r1 (A, R2, select_mux_r1, out_mux_r1);
mux32 mux_r2 (B, R1, select_mux_r2, out_mux_r2);
mux32 mux_r4 (C, R3, select_mux_r4, out_mux_r4);

adder32 adder (R2, R3, out_adder, cout);

comparator32 cmp1 (R1, R2, cmp1_gt, cmp1_e1, cmp1_lt);
comparator32 cmp2 (R3, R4, cmp2_gt, cmp2_e1, cmp2_lt);

wide_or or1 (R1, or_out);

control_unit cu (clk, start, or_out, cmp1_lt, cmp2_gt,
in_r, dec_r1, reset_r3, select_mux_r1, select_mux_r2, select_mux_r1);

assign out = R1;

endmodule
```

دقت کنید که برای طراحی این ماژول، نیاز به طراحی control unit را نیز داریم. به همین منظور، ماژول control unit را با الهام از اسلایدهای کلاس به صورت زیر طراحی می‌کنیم. در اینجا، بخش (ث) سوال نیز حل می‌شود:

```
module control_unit(
    input clk,
    input start,
    input or_out,
    input cmp1_lt,
    input cmp2_gt,
    output reg in_r,
    output reg dec_r1,
    output reg reset_r3,
    output reg select_mux_r1,
    output reg select_mux_r2,
    output reg select_mux_r4
);

reg [1:0] p_state, n_state;
localparam [1:0] init = 2'b00, cmp = 2'b01, mul = 2'b10;

always @(p_state or start or or_out or cmp1_lt or cmp2_gt) begin: combi
    dec_r1 = 0;
    reset_r3 = 0;
    select_mux_r1 = 0;
    select_mux_r2 = 0;
    select_mux_r4 = 0;

    n_state = init;
```

```
    case (p_state)
        init: begin
            if (start) begin
                n_state = cmp;
                in_r = 1'b0;
                select_mux_r1 = 1'b0;
                select_mux_r2 = 1'b0;
                select_mux_r4 = 1'b0;
                reset_r3 = 1'b1;
            end else begin
                n_state = init;
            end
        end

        cmp: begin
            n_state = mul;
            if (!cmp1_lt) begin
                select_mux_r1 = 1;
                select_mux_r2 = 1;
            end
        end

        mul: begin
            if (or_out) begin
                n_state = mul;
                dec_r1 = 1;
            end else begin
                n_state = cmp;
                if (cmp2_gt) begin
                    select_mux_r4 = 0;
                end else begin
                    select_mux_r4 = 1;
                end
            end
        end

        default: n_state = init;
    endcase
end

always @(posedge clk) begin: sequential
    p_state <= n_state;
end

endmodule
```

ت) نمودار حالت مدار به صورت زیر می‌باشد:



ج) در نهایت برای تست ماژول، از فایل testbench زیر استفاده می‌کنیم:

```
`timescale 1ns / 1ps

`include "datapath.v"

module testbench;

    reg clk;
    reg start;
    reg [31:0] A;
    reg [31:0] B;
    reg [31:0] C;

    wire out_r;
    wire [31:0] out;

    datapath uut (
        .clk(clk),
        .start(start),
        .A(A),
        .B(B),
        .C(C),
        .out_r(out_r),
        .out(out)
    );

    always #5 clk = ~clk;

    initial begin
```

```
        clk = 0;
        start = 0;
        A = 0;
        B = 0;
        C = 0;

        A = 32'd15;
        B = 32'd10;
        C = 32'd20;

        start = 1;
        #1000;
        $display(out); // test 1
        start = 0;

        #100;

        A = 32'd3;
        B = 32'd4;
        C = 32'd20;

        start = 1;
        #1000;
        $display(out); // test 2
        start = 0;

        #100;

        A = 32'd12;
        B = 32'd11;
        C = 32'd1000;

        start = 1;
        #1000;
        $display(out); // test 3
        start = 0;

        #100;

        A = 32'd24;
        B = 32'd1;
        C = 32'd25;

        start = 1;
        #1000;
        $display(out); // test 4
        start = 0;

        #100;

        A = 32'd5;
        B = 32'd5;
        C = 32'd20;

        start = 1;
        #1000;
        $display(out); // test 5
```

```
        start = 0;

        #100;

        A = 32'd19;
        B = 32'd28;
        C = 32'd10;

        start = 1;
        #1000;
        $display(out); // test 6
        start = 0;

        #100;


        $stop;
    end

endmodule
```

خروجی این تست بنچ با استفاده از کامپایلر iverilog به صورت زیر خواهد بود:

```
          20
          12
          132
          24
          20
          10
testbench.v:103: $stop called at 6600000 (1ps)
** VVP Stop(0) **
** Flushing output streams.
** Current simulation time is 6600000 ticks.
> finish
** Continue **
```