جبرانی پایان ترم-علیرضا مفاخری - ۴۰۱۱۰۶۵۵۵

گزارش قسمت الف سوال یک:

بخش الف)

در این سوال ابتدا یک استک را تعریف می کنیم که به صورت آرایه دو بعدی است و برای آن یک پوینتر تعریف می کنیم که به خانه خالی در استک اشاره می کند (لزوما البته به خانه خالی اشاره نمی کند.)

در زیر نحوه پیاده سازی هر یک از اپراتور ها را شرح می دهیم.

عملگر ضرب :

```
//multiply
else if(opcode==3'b101)begin
    output_data=stack[pointer-1] * stack[pointer-2];
    checking_of=stack[pointer-1] * stack[pointer-2];
    extended_reg={{N{output_data[N-1]}},output_data};
    if(checking_of != extended_reg)begin
        overflow=1;
    end
    else
        overflow=0;
end
```

در اینجا دو خانه قبلی را ضرب می کنیم و حاصل را در خروجی می ریزیم .برای چک کردن اورفلو باید چک کنیم که حاصل در اِن بیت جا می شود یا خیر . پس یکبار حاصل را در یک رجیستر دو اِن بیتی می ریزیم و یکبار ریز مقدار خروجی را ساین اکستند می کنیم و در رجیستر دو اِن بیتی می ریزیم حال اگر این دو برابر شوند یعنی خروجی اورفلو نداشته و در غیر این صورت اورفلو داشته ایم.

عملگر جمع:

```
//addition
if(opcode==3'b100 && (pointer>=2))begin
    output_data=stack[pointer-1]+stack[pointer-2];
    if((stack[pointer-1][N-1]==stack[pointer-2][N-1]) && (output_data[N-1]!= stack[pointer-1][N-1]))begin
        overflow =1;
    end
    else
        overflow=0;
end
```

در این عملگر دو خانه قبلی استک را با هام جمع می کنیم و به خروجی می دهیم.و برای چک کردن اورفلو می دانیم در صورتی اورفلو داریم که از جمع مثبت و مثبت به منفی یا از منفی و منفی به مثبت برسیم .پس ابتدا چک می کنیم که ساین بیت دو خانه استک برابر باشد و با ساین بیت خروجی متفاوت باشد آنگاه اورفلو داریم.و در غیر این صورت نه.

عملگرpush:

```
//push
else if(opcode==3'b110)begin
    stack[pointer]=input_data//put input in the current place of stack that pointer point into that;
    pointer=pointer +1//inc pointer;
end
```

دقیقا مثل پوش کردن در خود استک است .ابتدا در جایی که پوینتر نشان می دهد می نویسیم و سپس پوینتر را به خانه بعدی می بریم.

عملگرpop:

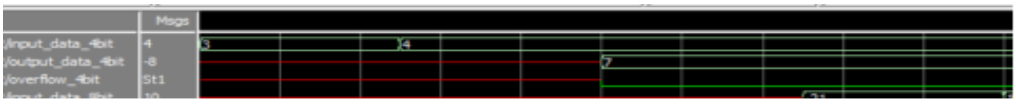
```
//pop
else if(opcode==3'111)begin
    pointer=pointer-1;//dec pointer
    output_data=stack[pointer]//pick the amount of stack in perevious place;
end
```

مثل مورد پوش کردن دقیقا مانند استک ابتدا یک واحد ازپوینتر کم می کنیم سپس جایی که پوینتر به ان اشاره می کند را در خروجی می ریزیم.

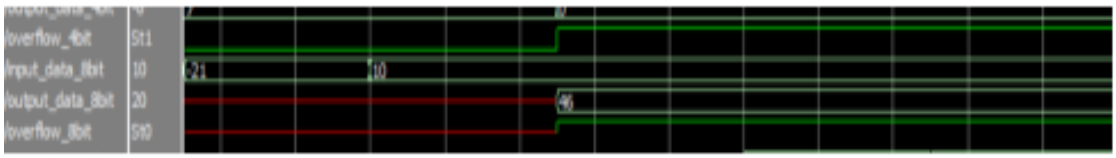
برای تست کردن نیز ۴ تا استک بیس اینیشیال میکنیم با ورودی و خروجی های ۳۲ و ۱۶ و ۸و ۴ بیتی و در هر کدام نیز اورفلو را چک می کنیم.

طبق کد های بالا در صورت عملگر نو اوپریشن هیچ اتفاقی نخواهد افتاد.

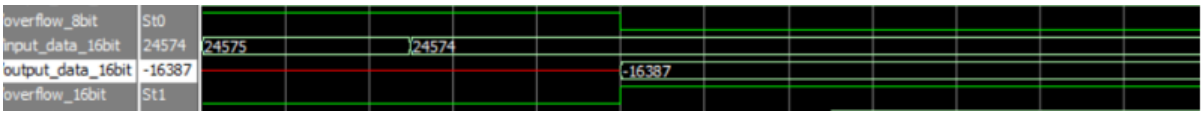
حال برای هر حالت ویو فرم را طبق تست بنچ بررسی می کنیم.



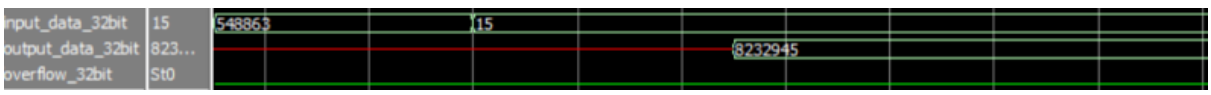
در ۴ بیتی میبینیم که جمع دو عدد ۳ و ۴ را به درستی انجام داده و اورفلو نیز رخ نداده پس بیت اورفلو صفر مانده است.



در ۸ بیتی میبینیم که ضرب ۲۱-در ۱۰ شده ۴۶ که یعنی اورفلو انجام شده و همانطور که مشخص است بیت اورفلو یک شده است.



در ۱۶ بیتی نیز جمغ دو عدد مثبت منفی شده پس اورفلو رخ داده و بیت اورفلو نیز یک شده است.



در اینجا نیز ضرب به صورت درست انجام شده بنابر این بیت اورفلو بدون تغییر صفر مانده است.

بخش ب)

در این بخش که کد آن به صورت زیر است

```
module postfix #((parameter N=8)(
  input signed [N-1:0] char,
  input wire check,
  input wire is_free,
  output reg signed [N-1:0] fianl_out
);
  reg [2:0] op_opcode;
  reg [2:0] num_opcode;
  wire overflow;
  wire signed [N-1:0] op_out;
  wire signed [N-1:0] num_out;
  reg signed [N-1:0] op_reg;
  reg signed [N-1:0] operatian;
  stack_base_alu #(N) num_stack(.input_data(op_reg), .opcode(num_opcode), .output_data(num_out),.overflow(overflow));
  stack_base_alu #(N) op_stack(.input_data(operatian), op_opcode, .output_data(op_out), .overflow(overflow));
  always @(*) begin
  if (check) begin
    if (char == 16'h002b) begin
      forever begin : loop
```

```
        op_opcode =3'b111;
        #10;
        op_opcode = 3'bxxx;

        if (op_out == 16'h0028) begin
          operatian = 16'h0028;
          op_opcode = 3'b110;
          #10;
          op_opcode = 3'bxxx;
          disable loop;

        end
        else if(no_op == 1) begin
        disable loop;
        end
        else begin
          if (op_out == 16'h002b) begin
            num_opcode = 3'b100; //sum
            #10;
            op_reg = num_out;
```

```
            num_opcode =3'b111;
             #10
            num_opcode =3'b111;
             #10
            num_opcode = 3'b110; // push
            #10;
            num_opcode = 3'bxxx;

          end
        end

    end
    operatian = 16'h002b;
    op_opcode = 3'b110;
    #10;
    op_opcode = 3'bxxx;

end

end
```

```
    else begin
      op_reg = char;
      num_opcode = 3'b110;
      #10;
      num_opcode = 3'bxxx;

      if (is_free) begin
        forever begin : loop1
          op_opcode =3'b111;
          #10;
          op_opcode = 3'bxxx;

          if(no_op == 1) begin
            disable loop1;
          end
          else begin
            if (op_out == 16'h002b) begin
              num_opcode = 3'b100; //sum
              #10;
              op_reg = num_out;
              num_opcode =3'b111;
               #10
```

```
            num_opcode =3'b111;
             #10
            num_opcode =3'b111;
             #10
            num_opcode = 3'b110; // push
            #10;
            num_opcode = 3'bxxx;

          end
        end

  end
   num_opcode =3'b111;
    #10;
    fianl_out = num_out;
```

یک استک برای عملگر ها و یک استک نیز برای اعداد درست می کنیم برای عملگر ها یک بیت ورودی چک داریم که وقتی یک است یعنی ورودی ما یکی از عملگر های +*()است.

برای پوش کردن هر عملگر در استک عملگر ها یک سری شرایط وجود دارد.برای پوش کردن * باید ان را مستقیم پوش کنیم و شرطی برای ان نداریم.

برای + شروع به پاپ کردن می کنیم تا استک خالی شود یا به) برسیم و در این میان اگر به عملگر + یا * رسیدیم دو عدد بالای استک اعداد را پاپ کرده عملگر مورد نظر را روی آنها انجام می دهیم و سپس حاصل را نیز در استک ذخیره می کنیم.و اگر به) رسیدیم صرفا آن را پوش می کنیم و برای (انقدر پاپ کردن را ادامه می دهیم تا به یک) برسیم.برای پایان نیز یک بیت ایز فری تعریف می کنیم که وقتی به آن رسیدیم می آییم و هر عملگر را پاپ کرده و با دو عدد بالای استک اعداد که انها را نیز پاپ می کنیم اجرا می کنیم و سپس می آییم نتیجه را درون استک اعداد پوش می کنیم و به این صورت عملیات را به صورت پسوندی اجرا میکنیم.برای اینکه مطمئن شویم که ایز فری یک شده می توانیم عبارت میانوندی را درون یک پرانتز قرار دهیم و سپس به ماژول خود ورودی دهیم که خروجی نهایی نیز تفاوتی نخواهد کرد.در حل این سوال فرض کردیم که عبارت میانوندی به صورت صحیح و درست پرانتزبندی شده و ایرادی از این باب ندارد.