1. RAM (Stack)

System

```
wire push,pop,reset;
singlePulser(push, btnU, targetClk); // push stack
singlePulser(pop, btnC, targetClk); // pop stack -> display top value in num3,num2 and stack size i
singlePulser(reset, btnD, targetClk); // reset

wire [3:0] num3,num2,num1,num0; // left to right
SinglePortRAM stack({num1,num0},{num3,num2},sw[7:0],pop,targetClk,push,reset);
```

SinglePortRAM สำหรับ implement Stack

```
reg [7:0] mem [255:0];

initial begin
    dout = 0;
    addr = 0;
end

always @(posedge clk) begin
    if(we) begin // write enable = write d_input into RAM
        mem[addr] = din;
        addr = addr+1;
    end
    if(reset || (oe&&addr==0)) begin // reset button or empty stack
        dout = 0;
        addr = 0;
    end
    if(oe && addr > 0) begin // read from stack to d_output
        addr = addr-1;
        dout = mem[addr];
        mem[addr] = 0;
    end
end
```

2. ROM (Display Binary to BCD)

System

```
reg [3:0] num3,num2,num1,num0; // left to right
reg [7:0] rom[2**5-1:0];
initial $readmemb("rom2.mem", rom);

always @(posedge targetClk)
    {num3,num2,num1,num0} = {8'b00000000, rom[sw[4:0]] };
```

rom2.mem อธิบายในข้อ 4

3. ROM (Calculator)

System

```
reg [3:0] num3,num2,num1,num0; // left to right
reg [15:0] rom[2**10-1:0];
initial $readmemb("rom4.mem", rom);

reg [1:0] mode;

always @(posedge targetClk && (btnU || btnL || btnD || btnR)) begin
    case({btnU, btnL, btnD, btnR})
        4'b1000: mode = 0; //plus
        4'b0100: mode = 1; //subtract
        4'b0010: mode = 2; //multiply
        4'b0001: mode = 3; //divide
    endcase
    {num3,num2, num1, num0} = rom[{mode,sw[7:0]}];
end
```

ใน rom4.mem มีข้อมูล 16 bits บอกตัวเลข BCD 4 ตัว โดย map ตามการคำนวณ 2 bits และตัวเลข 4 bits สองตัว รวมเป็น 10 bits

4. ROM for mapping 5-bit binary to BCD

ROM ใช้ map จากค่าในเลขฐานสองเป็นฐาน 10 (ที่ถูกเขียนด้วยฐาน 2 4 bit) สองตัว เช่น 10100 (20) จะได้เป็น 0010 0000 (20) โดยเก็บข้อมูลทั้งหมดไว้ใน file memory ขนาด 32*8 bits