.Fill in the blanks in the below modules

;module ShiftLeftBy2 (out, in)

;input [1:0] in

;output [3:0] out

; { in[1:0], 2'b00 } = assign #4 out

endmodule

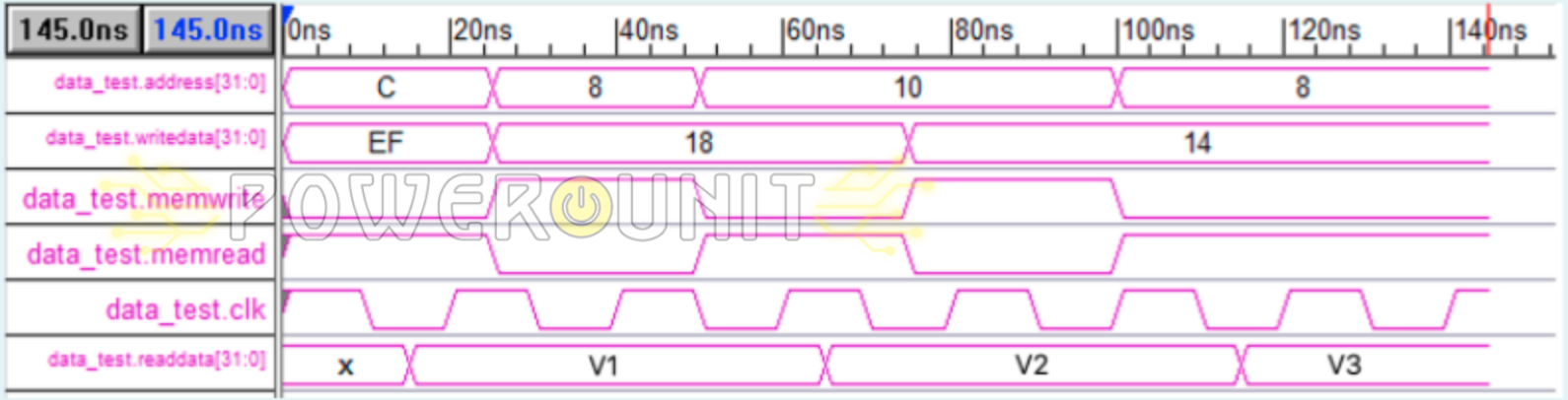
;module ShiftRightBy2 (out, in)

;input [3:0] in

;output [3:0] out

; { b00, in[3:2]'2 } = assign #4 out

endmodule

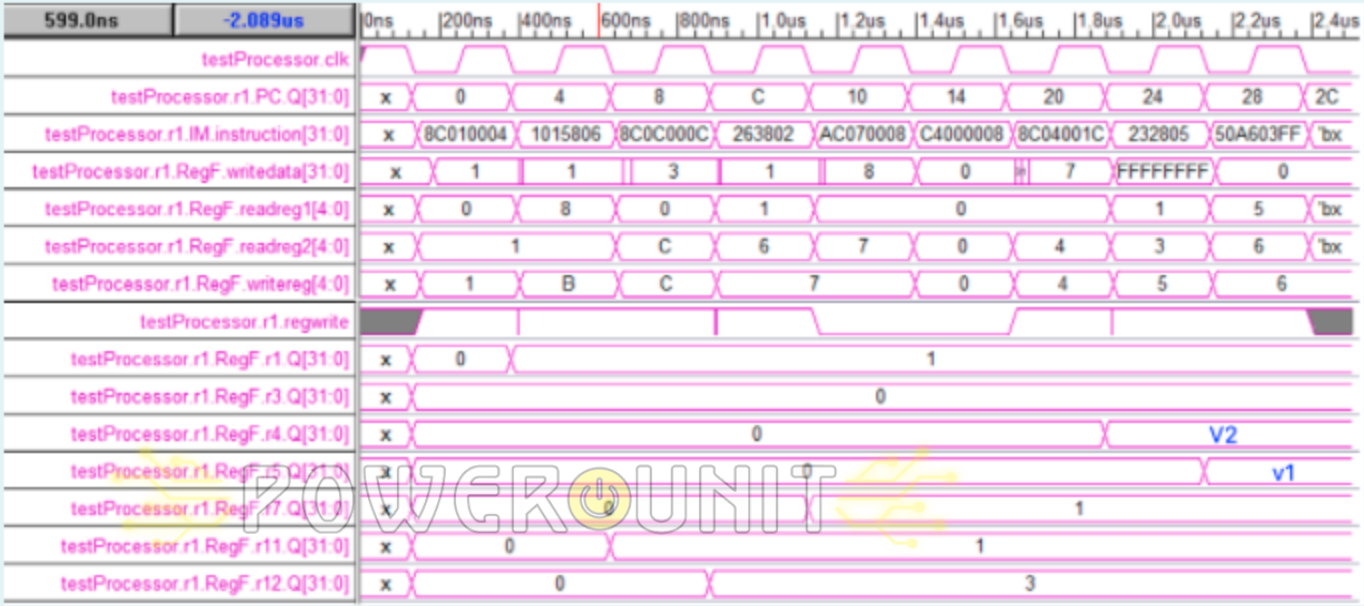


Given the above timing diagram for the data memory module in experiment 4, answer the below questions.

What is the value of v1 ? 3

?What is the value of v3 18

?What is the clock cycle time 20



Given the above timing diagram of the single-cycle processor in experiment 6, answer the below questions.

Note: if needed, save the timing diagram to your computer so you can zoom and see the full timing diagram with better clarity .

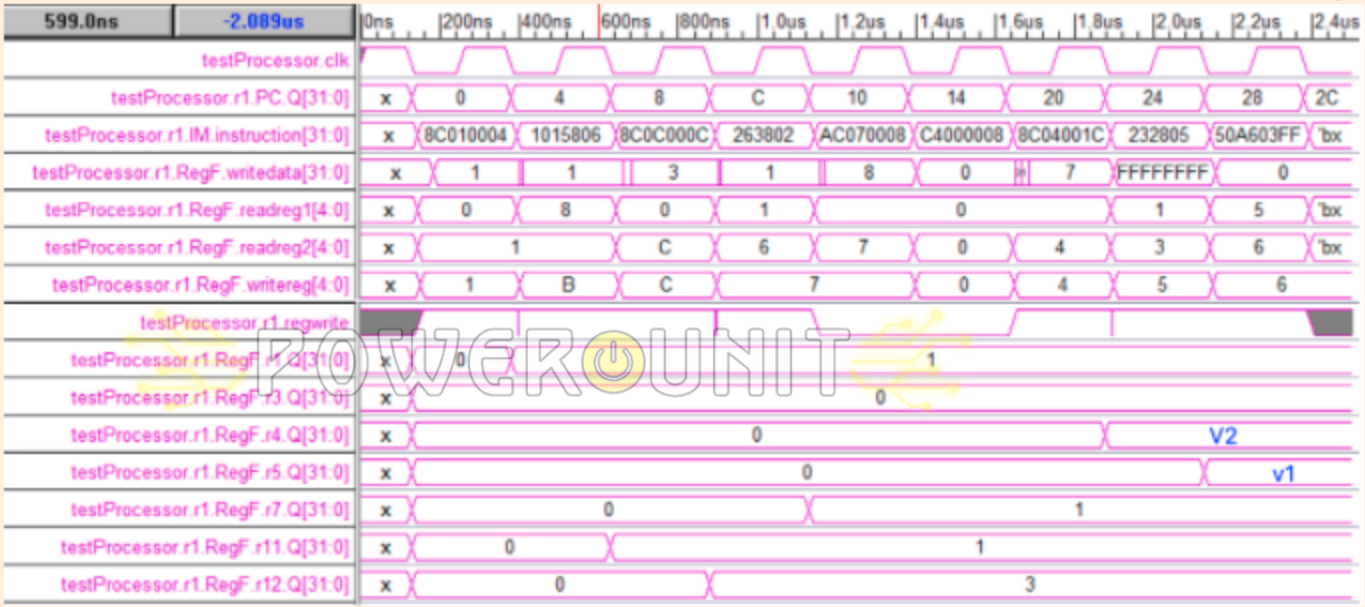
What is the number of jump/branch instructions ? 3

What is the number of instructions that write to register R8 ? 0

7 ? What is the value of V2

What is the number of instructions that read register R5 ? 1

الإجابة الصحيحة هي:



Given the above timing diagram of the single-cycle processor in experiment 6, answer the below questions.

Note: if needed, save the timing diagram to your computer so you can zoom and see the full timing diagram with better clarity .

What is the number of jump/branch instructions ? [1]

What is the number of instructions that write to register R8 ? [0]

What is the value of V2 ? [7]

What is the number of instructions that read register R5 ? [1]

We want to implement a 2 bit ALU module that supports only two operations: ADD (when m=0) and SUB (when m=1).

Fill the blanks in the below module to complete the code.


```
;module ALU_2bit (Result, A, B, m)
;input m
;input [1:0] A, B
;output [1:0] Result
;wire  x, y , z, w
XORgate (out, in1, in0)//
; XORgate x0 ( x , [ m ] , B[0] )
; XORgate x1 ( y , [ m ] , B[1] )
FullAdder (Sum, Cout, Cin, a, b) //
;FullAdder fa0 ( [ Result[0] ] , [ z ] , [ m ] , A[0] , [ x ] )
;FullAdder fa1 ( [ Result[1] ] , w , z , A[1] , [ y ] )
endmodule
```

```
;module Xorgate2 (out, in1, in0)
;input in1, in0
;out out
;assign #2 out = in1 ^ in0
endmodule

;module FullAdder (sum, cout, a, b, cin)
;input a, b, cin
;out sum, cout
;assign #3 sum = a ^ b ^ cin
; assign #2 cout = a & b | a & cin | b & cin
endmodule
```

?Assuming we want to use the above xor and full adder modules to implement a 4-bit adder/subtractor circuit, what is the delay

7 ○

8 ○

9 ○

15 ○

10 ○

11 ○

14 ○

17 ◉

12 ○

16 ○

الإجابة الصحيحة هي:

11

Answer the below short answer questions.

Q1. Is the below statement true or false?

The propagation delay of the Control unit implemented in experiment 5 is bigger than the propagation delay of the register file implemented in experiment 3. false

b10'2 ? Q2. What is the output of the below Verilog module if the input is in = 3'b100

```
;module circuit (out, in)
;input [2:0] in
;output [1:0] out
; assign out[1] = in[1] | ~in[0] & in[2]
;assign out[0] = ~in[0] ^ in[2]
endmodule
```

Q3. Assume the delay for an AND gate is 1, and the delay for an XNOR gate is 2. What is the delay of the below module? 4


```
module circuit (out, a, b);
    input [2:0] a,b;
    output out;
    wire [3:0] w ;
    XNORgate x0 ( w[0], a[0], b[0] ) ;
    XNORgate x1 ( w[1], a[1], b[1] ) ;
    XNORgate x2 ( w[2], a[2], b[2] ) ;
    ANDgate  a0 (  w[3], w[0], w[1]) ;
    ANDgate  a1 (  out, w[2], w[3]) ;
endmodule
```

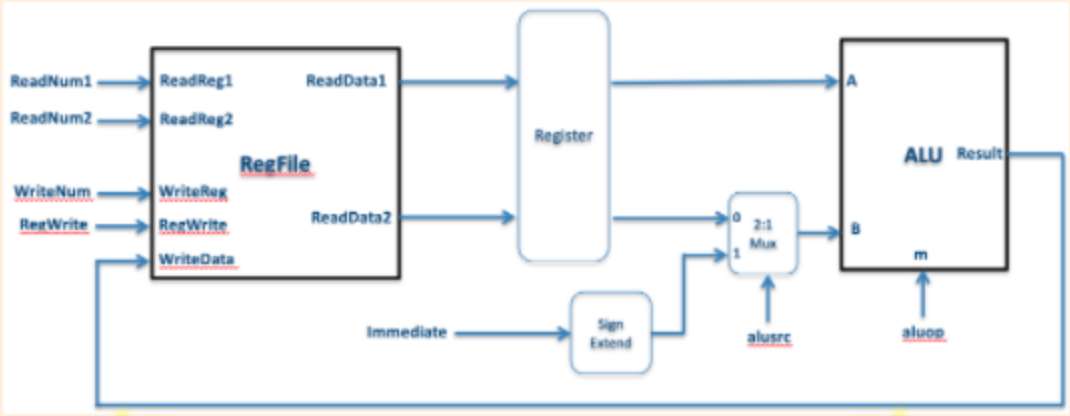
opcode	function	W
000000	100000	0
000000	100010	0
000000	100100	1
000000	100110	1
000000	101000	0
000000	101010	0
000000	101100	1
000000	101110	1
100010	------	1
100001	------	0

Given the above truth table, what is the equation of W ?

- opcode[5] & function[2] | ~opcode[5] & opcode[1]
- ~opcode[5] & function[2] | opcode[5] & opcode[0]
- opcode[5] & ~function[2] | ~opcode[5] & opcode[0]
- opcode[5] & function[2] | ~opcode[5] & opcode[0]
- ~opcode[5] & ~function[2] | opcode[5] & opcode[0]
- opcode[5] & ~function[2] | ~opcode[5] & opcode[1]
- **~opcode[5] & function[2] | opcode[5] & opcode[1]**
- ~opcode[5] & ~function[2] | opcode[5] & opcode[1]

The correct answer is:

Fill in the blanks in the below code.




```
module circuit (immediate, ReadNum1, ReadNum2, WriteNum, RegWrite, alusrc, aluop, clk, reset, enable);
    input [15:0] immediate;
    input [4:0] ReadNum1, ReadNum2, WriteNum;
    input [2:0] aluop;
    input alusrc, clk, reset, enable, RegWrite;
    wire [31:0] w1, w2, w3, w4, w5, w6, w7;
    //RegFile ( ReadData1, ReadData2, ReadReg1, ReadReg2, WriteData, WriteReg, RegWrite, clk, reset)
    RegFile RF ( w1,  [w3] , ReadNum1, ReadNum2,  [w4]  , WriteNum, RegWrite, clk, reset );
    //SignExtend( out, in )
    SignExtend se (  [w7]  , immediate ) ;
   //Register ( Q, D, clk, reset, enable )
    Register reg ( {w2, w5} , {w1, w3}, clk, reset, enable );
    //Mux2to1_32bit( out, s, i1, i0 )
    Mux2to1_32bit mux1 ( w6 , alusrc , w7 , w5 );
    // ALU ( Result, A, B, m )
    ALU alu (  [w4]  ,  [w2]  ,  [w6]  , aluop ) ;
endmodule
```

In this question, you need to use Verilogger pro to write a test module and generate a timing diagram for a given circuit by following these steps:

step 1: download this compressed folder (midterm.zip) from this link:

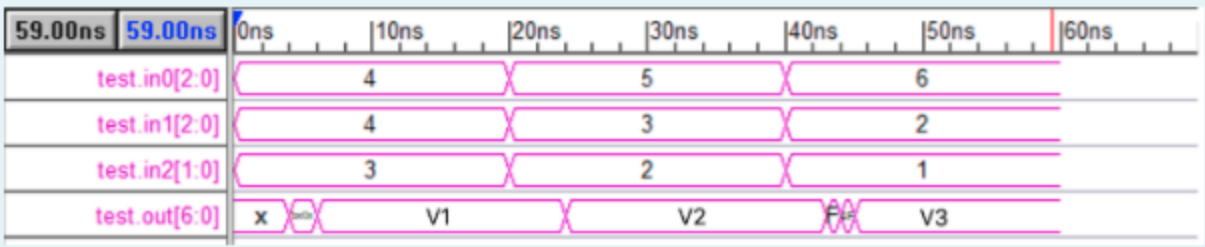
https://drive.google.com/uc?id=17VXyKrb4f1WGb_f6VZ2CxA_rkGLeKBpU&export=download

step 2: unzip the folder and open the Verilogger project I already provided

step 3: add the verilog files circuit.v and test.v to the project

step 4: note that the top-level module ***circuit (in0, in1, in3, out)*** is already provided in circuit.v (therefore, do not make any modifications in circuit.v)

step 5: in test.v, write a test module for ***circuit*** module to generate the below timing diagram



step 6: Fill-in the blanks in below to write the values of V1, V2 and V3

V1 = 30

V2 = 38

V3 = 4F