## Ain Shams University, Faculty of Engineering

## Computer and Systems Engineering Department

## COMPUTER ORGANIZAION II

## Midterm Examination - November 2017

#### Time: 1.5 Hours الإجابة فى الورقة المرفقة Total: 45 Points

**For each of the following multiple choice questions, select ONLY the ONE correct answer. Mark your choice in the answer sheet.**

أختار فقط الإجابة الصحيحة الوحيدة لكل من الأسئلة متعددة الاختيارات الآتية في ورقة الإجابة.

1. Using N pipeline stages in a processor, it is extremely easy to achieve a speed up of N relative to single cycle implementation.

|  |  |  |  |
| --- | --- | --- | --- |
| 1. True | 1. False | 1. Blank | 1. Blank |

1. If A=1’b1, B=2’b01, C=2’b00, then Y ={{4{A}},{2{B}},{C}} equals

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 10’b1111010100 | 1. 10'b0010101111 | 1. 4'd420 | 1. none of the previous |

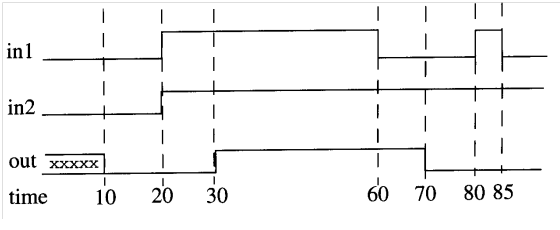
1. Let m1(R,A,B,C,D,E,F,G) be a Verilog module, taking the signals A,B,C,D,E,F,G as inputs, and giving the signal R as output. Because you were asleep during simulation, you forgot to initialize the value of D in your initial block, while you initialized all the other inputs perfectly. When you tried to simulate the module and view the value of R , you found that it takes a value of either 1'b1 or 1'bx. What can the logic function that module m1 performs be ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. R=((A|F)^ (B & C |H))|D | 1. R=((D|F)^ (D & !C |H))|H | 1. R = A^B^C^D | 1. D & ~D |

1. Is your choice in the previous question the only possible choice that can produce this behavior (in the whole world) ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Yes | 1. No | 1. Blank | 1. Blank |

1. Consider the following waveform. what could be the verilog code producing such result ?



|  |  |  |  |
| --- | --- | --- | --- |
| 1. **assign #40 out = in1&in2** | 1. **assign out = in1&in2** | 1. **assign #20 out = in1&in2** | 1. **assign #10 out = in1&in2** |

1. In continuous assignment, the left hand side must be of type

|  |  |  |  |
| --- | --- | --- | --- |
| 1. reg | 1. wire | 1. integer | 1. genvar |

1. A reg datatype can be used in verilog modules as

|  |  |  |  |
| --- | --- | --- | --- |
| 1. input only | 1. output only | 1. both of them | 1. none of them |

1. A wire datatype can be used in verilog modules as

|  |  |  |  |
| --- | --- | --- | --- |
| 1. input only | 1. output only | 1. both of them | 1. none of them |

1. consider the following verilog line :

**reg [31:0] M [0:31]**

Let M[i] = i for 0 < i < 32 (as initialization)

What is the value of M[30][3] ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 3 | 1. 1 | 1. 0 | 1. 30 |

1. Now suppose you have the following verilog module , what will be the result of running this program ?

**module A\_Trials();**

**reg[31:0] M [0:31];**

**reg [0:31] Q [31:0];**

**integer i;**

**initial**

**begin**

**for( i = 0 ; i <32 ; i = i+1)**

**begin**

**M[i] = i;**

**Q[31-i] = i ;**

**end**

**#10**

**$monitor("%d", M[5][5:0]+ Q[5][26:31]);**

**end**

**endmodule**

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Error | 1. 5 | 1. 15 | 1. 31 |

1. Suppose you created a verilog 5-bit Alu module that is capable of performing signed operations and supports overflow. you want to test overflow in addition by adding two inputs (a,b). what are possible values for (a,b) to actually have the overflow signal asserted ?

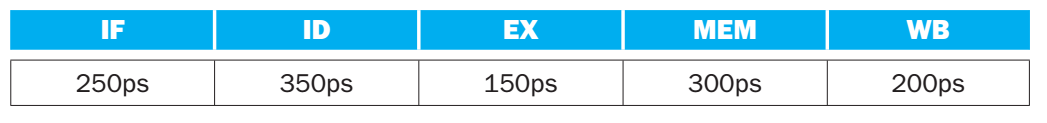
|  |  |  |  |
| --- | --- | --- | --- |
| 1. (10,6) | 1. (14,1) | 1. (16,10) | 1. all of them |

1. For the answer you chose in the previous question, what would be the Alu output when printed as a signed number ?

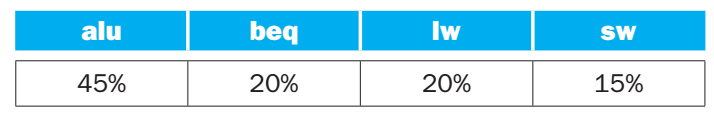
|  |  |  |  |
| --- | --- | --- | --- |
| 1. 26 | 1. -1 | 1. -15 | 1. none of them |

1. A single cycle processor has a clock cycle of 1GHz frequency. What is the maximum operating frequency we can achieve if we split it to a five-stage processor ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 5 GHz | 1. 3 GHz | 1. (1/5.0) GHz | 1. 1 GHz |

1. Suppose we have a pipelined processor, where the stages of the pipeline have the following latencies.  
   

Also assume the instructions executed by the processor are broken down as follows :



What is the clock cycle time in a single-cycle processor (in ps) ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 350 | 1. 1000 | 1. 1250 | 1. 1300 |

1. Referring to tables in Q(14), What is the clock cycle time in a pipelined processor (in ps) ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 350 | 1. 1000 | 1. 1250 | 1. 1300 |

1. Referring to tables in Q(14), what is the total latency of the LW instruction in a pipelined implementation ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 350 | 1. 1250 | 1. 1750 | 1. 300 |

1. Referring to tables in Q(14), what is the total latency of the LW instruction in a Non-pipelined implementation ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 1250 | 1. 1750 | 1. 1500 | 1. 1000 |

1. Referring to tables in Q(14), if we can split one stage into two stages with equal latencies, what stage should we choose to improve the overall performance of the processor ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. ID | 1. MEM | 1. EX | 1. WB |

1. What would the new clock cycle time after the splitting in Q (18) ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 350 | 1. 1250 | 1. 300 | 1. 1750 |

1. What would be the latency of the LW instruction after the splitting in Q(18) ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 1500 | 1. 1800 | 1. 350 | 1. 300 |

1. Imagine a program with millions of instructions with no stalls or hazards, what is the overall speedup achieved after the splitting chosen in Q(18) over the original 5-stage pipelined performance ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 1750/1500 | 1. 1750/350 | 1. 1800/350 | 1. none of them |

1. Referring to tables in Q(14), Assuming there is no stalls or hazards, what is the utilization of the write port of the data memory ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 20% | 1. 15% | 1. 35% | 1. 45% |

1. Referring to tables in Q(14), Assuming there is no stalls or hazards, what is the utilization of the first read data port of the register file ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 45% | 1. 65% | 1. 80% | 1. 100% |

1. Referring to tables in Q(14), Instead of a single-cycle organization, we can use a multi-cycle organization where each instruction takes multiple cycles but one instruction finishes before another is fetched. In this organization, an instruction only goes through stages it actually needs (e.g., ST only takes 4 cycles because it does not need the WB stage). Also suppose the branch instruction finishes in the memory stage. How many cycles would a program of 1000 instructions take on average if it follows the frequencies provided in the tables ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 4000 | 1. 4200 | 1. 4400 | 1. none of them |

1. What is the clock cycle period of the configuration in Q(24) ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 300 | 1. 350 | 1. 1250 | 1. 1750 |

1. What is speedup achieved for the configuration in Q(24) over the pipelined implementation ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 4.2 | 1. 42 | 1. .42 | 1. none of them |

Consider the following code snippet. What will be printed on the screen ?

**module A\_Trials();**

**reg [3:0] A,B,C,D,E;**

**initial**

**begin**

**A = 5;**

**B = 3;**

**C = 4;**

**D = 5;**

**E = 4;**

**#10**

**A<=B;**

**B<=A;**

**C<= A+B;**

**E <= E+1;**

**D <= A+B+C+E;**

**$monitor("%d",D);**

**end**

**endmodule**

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 21 | 1. 16 | 1. 0 | 1. 5 |

1. Conisder the following code snippet, what is the value printed on the screen ?  
    **module A\_Trials();**

**reg [3:0] A,B,C,D,E;**

**initial**

**begin**

**A = 5;**

**B = 3;**

**C = 4;**

**D = 5;**

**E = 4;**

**#10**

**A = B;**

**B = A;**

**C = A+B;**

**E = E+1;**

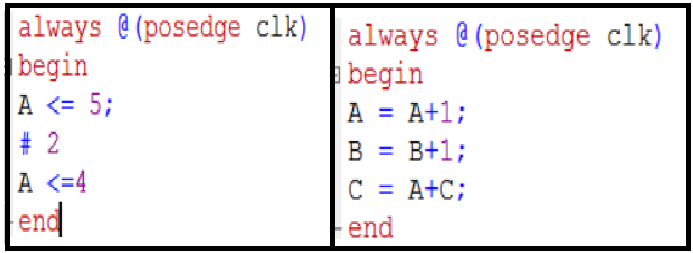
**D = A+B+C+E;**

**$monitor("%d",D);**

**end**

**endmodule**

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 21 | 1. 17 | 1. 1 | 1. none of them |

Please Read Very Carefully. Verilog code is said to be synthesizable if it can be "burnt" on real hardware (FPGA) and work as expected without problems as an IC. Sometimes, you write verilog code that works perfectly fine on simulators (like modelsim) but then raise errors when you try to synthesize it on an FPGA. That may happen for many reasons. One example causing this would be changing a "reg" value multiple times in the same clock cycle. Another example would be using the blocking assignment operator "=" instead of non-blocking assignment operator " <= " to give a value to a "reg", because the blocking assignment causes the code to execute sequentially while in reality hardware executes in parallel. So, the blocking assignment operator has to be used in simulation or for testing only but mustn't be used in synthesizable hardware modules. The following screenshot has examples of non-synthesizable codes due to the previously mentioned reasons:   


Now, Do you want an extra mark for reading very carefully?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Yes | 1. No | 1. Who Cares ? | 1. Noneof the previous |

1. During work on the ALU project, you hear that one of your friends is struggling to implement one of the alu operations. You then offer to help with a big, confident smile on your face, telling him/her that he/she should use a code similar to the following to implement the operation.  
   module A\_Trials();

**reg[31:0] data;**

**integer i = 0;**

**integer shift\_amount = 5;**

**initial**

**begin**

**data = 32'b1111\_1111\_0011\_0011\_1100\_1100\_1010\_1010;**

**for(i = 0 ; i < 31-shift\_amount; i = i + 1)**

**data[i] = data[i+shift\_amount];**

**$monitor("%b",data);**

**end**

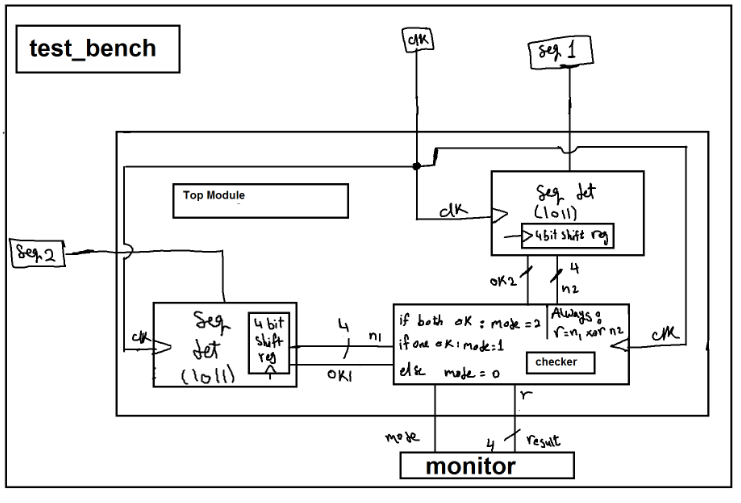
**endmodule**  
What is the operation that your friend was struggling to implement ?

|  |  |
| --- | --- |
| 1. shift right arithmetic | 1. shift right logical |
| 1. shift left arithmetic | 1. shift left logical |

1. Referring to Q(29), Is the code you provided in Q(30) synthesizable ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Yes | 1. No | 1. Blank | 1. Blank |

1. Consider The Following Design :



Consider the wire named "n1" that comes out of the sequence detector module and goes into the checker module. How should this signal be defined when you are coding your "top\_module" module ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. output wire [3:0] n1 | 1. output reg[3:0]n1 | 1. wire [3:0] n1 | 1. input reg [3:0] n1 |

1. Referring to Q(32), How should the signal "n1" be defined when you are coding your sequence detector "seq\_det" module ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. output wire[3:0]n1 | 1. output reg[3:0]n1 | 1. reg n1 | 1. wire n1 |

1. Referring to Q(32), How should the signal "n1" be defined when you are coding your shift register module ?

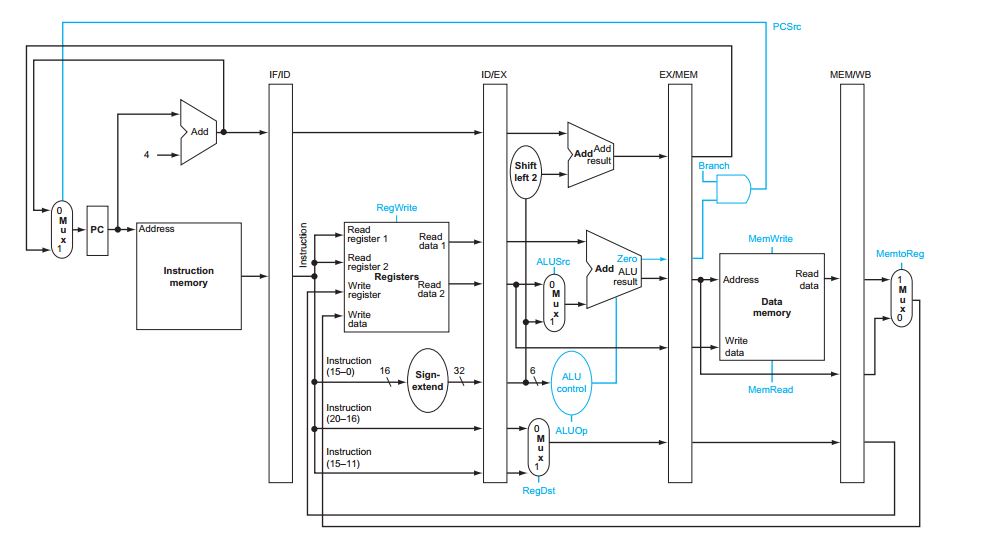
|  |  |  |  |
| --- | --- | --- | --- |
| 1. output wire[3:0]n1 | 1. output reg[3:0]n1 | 1. output reg n1 | 1. wire n1 |

1. Regarding Mips processor, how many registers are there in the register file ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 32 | 1. 64 | 1. 5 | 1. 10 |

1. Regarding Mips processor, how many bits do we need to give a unique address to each register in the register file ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 5 | 1. 64 | 1. 6 | 1. 32 |

1. Consider the MIPS processor shown in the following screenshot: 

In which stage should the control unit of the pipelined processor be placed ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. IF Stage | 1. ID Stage | 1. EX Stage | 1. MEM Stage |

1. In the figure of Q(37), there exists one ALU and one Adder in the EX stage. For what instruction is the upper ADDER necessary ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. add | 1. lw | 1. beq | 1. ori |

1. Consider the following instruction sequence and Pipelined Mips processor :  
   add $s1, $s2, $s3  
   lw $s5, 4($s4)  
   sub $s6, $s7, $s8  
   At clock cycle (4), which instruction is using the execution stage ?

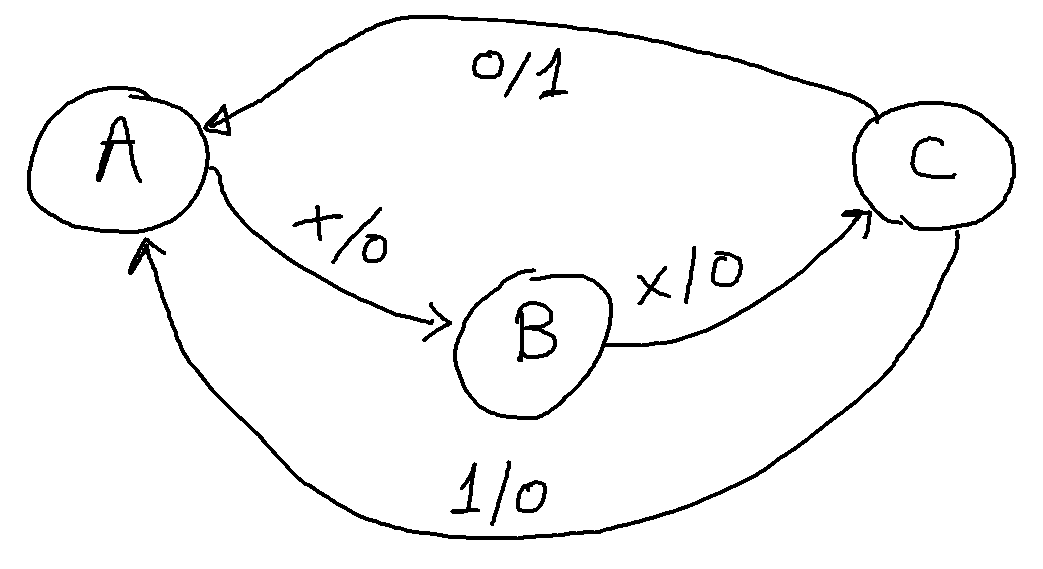
|  |  |  |  |
| --- | --- | --- | --- |
| 1. add | 1. lw | 1. sub | 1. none |

1. For the figure shown in Q(37), from where does the signal "RegWrite" directly come to the register file ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Control Unit | 1. MEM/WB pipeline register | 1. IF/ID pipeline register | 1. EX/MEM pipeline register |

1. For the figure shown in Q(37), at what stage of the pipeline can the signal "RegWrite" be activated for a single instruction ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. EX | 1. ID | 1. MEM | 1. WB |

1. Consider the following state machine (Sequence Detector)  
   

What is the sequence that it detects ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Even numbers less than 7 | 1. Odd numbers less than 7 | 1. xx1 | 1. xxx |

1. In verilog, if you use a variable without first defining it (as wire or reg), what will happen ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Error | 1. will be considered wire | 1. will be considered reg | 1. none of them |

1. You are simulating your circuit and find out that a signal you are monitoring has a value of "4'bzzz1". what could (most possibly of the available choices) be the cause of this behavior ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. you declared the signal twice | 1. floating [unconnected] wires | 1. un-initialized signal | 1. none of them |

1. You are simulating your circuit and find out that a signal you are monitoring has a value of "4'bxxxx". what could (most possibly of the available choices) be the cause of this behavior ?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. You declared he signal twice | 1. floating [unconnected] wires | 1. un-initialized signal | 1. none of them |

**E N D**