NOTE: It is my policy to give a failing grade in the course to any student who either gives or receives aid on any exam or quiz.

INSTRUCTIONS: Except where indicated otherwise, circle the letter of the one best answer for multiple choice questions. Fill in the blanks for fill in the blank questions. Follow the instructions for other questions.

1. How many ports does the MIPS register file have?
   A. One: for both reading and writing.
   B. Two: one for reading and one for writing.
   C. Three: one for reading and two for writing.
   D. Four: two for reading and two for writing.
   E. Three: two for reading and one for writing.

2. How many bits are associated with each port of the MIPS register file?
   A. Five: for the register number.
   B. Five: for the register data.
   C. Thirty-two: for the register number.
   D. Thirty-two: for the register data.
   E. Thirty-seven: five for the register number plus thirty-two for the register data.

3. What is special about MIPS register number zero?
   A. It is automatically used as the link register for jump and link instructions.
   B. The assembler reserves the right to change its contents when expanding pseudo-instructions into real instructions, if necessary.
   C. It is used as a stack pointer for push and pop instructions.
   D. It is used as a frame pointer for activation records during nested function calls.
   E. Reading from it always produces the value zero.

4. How were the slide switches used in testing the read operations of the MIPS Register File as implemented on the DE1?
   A. They selected whether the results would be displayed in the LEDs or in the seven-segment displays.
   B. They were used to select two register numbers for reading from.
   C. They were used to select which byte of the read operation to change.
   D. They were used generate the clock pulses needed to move from one instruction to the next.
   E. They were used to supply the function code that told the Register File what operation to perform.

5. What was the problem with testing the write operation on the DE1 implementation of the MIPS Register File, and how was it solved?
   A. There was no way to write into register zero, so it had to be simulated in software.
   B. Writing to register zero sometimes gave inconsistent results, so each write operation had to be performed twice to insure reliability.
   C. There were not enough switches to enter data for an entire register, so the test data had to be set up in multiple steps before the whole register could be written.
   D. There were not enough flip-flops on the DE1 FPGA to hold the entire register file, so it had to be split across two devices, which meant that the data had to be written twice.
   E. The DE1 clock was too slow to write data into all the registers at once, so a clock-doubler had to be added to the design.

6. *Jump* and *Jump and Link* instructions have a six-bit op-code and a 26-bit jump address field. How is the full 32-bit Jump Address calculated?
   A. The jump address field is sign-extended six bits to the left.
   B. The jump address field is zero-extended six bits to the left.
   C. The leftmost four bits of the PC are concatenated with the jump address field, and two bits of zero are concatenated to the end of that.
   D. The leftmost six bits of the PC are concatenated with the jump address field.
   E. The jump address is ANDed with Register 0 to produce the jump address.

7. Circles the letters of *all* the uses for I-format instructions:
   A. For jump instructions
   B. For instructions with one operand contained in the instruction
   C. For *lw* and *sw* instructions
   D. For branch instructions
   E. For instructions that operate on two registers and save the result in a third register

8. How is a branch target address calculated?
   A. Sign-extend bits 15:0 of the instruction, append two bits of zeros, and add the result to the PC.
   B. Zero extend bits 15:0 of the instruction, append two bits of ones, and subtract the result from the PC.
   C. Sign extend the instruction and append two bits of zeros.
   D. Add the immediate operand to register zero, and sign extend the result.
   E. Concatenate the leftmost 16 bits of the PC with the rightmost 16 bits of the instruction.

9. What arithemetic operation is performed by the ALU for branch instructions?
   A. The ALU is not used for branch instructions.
   B. The ALU subtracts one register operand from the other one to see if they are different.
   C. The ALU adds the branch target address to the link register.
   D. The ALU adds the branch target address to the jump address.
   E. The ALU performs the SLT function to determine the branch target address.

10. What determines whether a branch instruction actually branches or not?
    A. Whether the operand in memory is greater than the register operand or not.
    B. Whether register zero contains zero or not.
    C. Whether there is anything in the link register or not.
    D. Whether it is an even or odd numbered clock cycle.
    E. The Z bit.

11. How is the effective address calculated for *lw* and *sw* instructions?
    A. By converting the byte address to a word address and adding it to the PC.
    B. By putting the immediate operand into register 0.
    C. By sign-extending bits 15:0 of the instruction, and adding that to the contents of the register whose number is in bits 25:21 of the instruction.
    D. By using the *shamt* field of the instruction to determine where to branch.
    E. By adding register zero to the jump address.

12. What is the purpose of the *lui* instruction?
    A. It works with the Huey and Dewey instructions.
    B. It is used to under-increment latches.
    C. It makes it possible to load an arbitrary 32-bit value into a register.
    D. It converts ASCII characters from lower case to upper case.
    E. It is used to fill pipeline bubbles when there is nothing else to be done.

13. Why is there a separate adder for calculating PC+4 in the single-cycle CPU design?
    A. It speeds up the design.
    B. Because there is no way to add a constant to a register using the ALU.
    C. Because it eliminates the need for a full ALU in the datapath.
    D. Because the entire instruction has to be performed within a single clock cycle, and the main ALU cannot do more than one calculation per cycle.
    E. This is a trick question: there is no separate adder for calculating PC+4 in the single-cycle design; that is a feature of the multi-cycle design.

14. Why is it necessary to provide a mechanism for using either the *rt* or the *rd* field of instructions as the register number for writing data to the register file?
    A. Because *add* instructions use *rt*, but *sub* instructions use *rd.*
    B. Because I-Format instructions don't have an *rd* field.
    C. Because branch instructions need both *rt* and *rd* for saving the results.
    D. Because R-Format instructions don't have an *rt* field.
    E. It makes the instruction set more interesting.

15. What is the mechanism for using either the *rt* or *rd* field of instructions as the register number for writing data to the register file?
   A. There are separate instruction caches for the two types of instructions.
   B. There are separate register files for the two types of instructions.
   C. There are separate memories for the two types of instructions.
   D. The contents of register zero are used to determine which field to use.
   E. There is a 5×2 multiplexer controlled by the *RegDst* control signal.

16. How long would it take to execute one million instructions on a single-cycle processor with a 1 GHz clock?
   A. 1 minute
   B. 1 second
   C. 1 millisecond
   D. 1 microsecond
   E. 1 nanosecond

17. What determines the clock period of a single-cycle processor?
   A. The number of registers in the register file
   B. The number of ALUs
   C. The number of CPUs
   D. The number of DE1s
   E. The propagation delays for *lw* instructions

18. Fill in this table for the single-cycle design. One row is done for you. *Hint:* this tests whether you know which cells to leave blank.

Instruction Class	Instruction Memory	Register Read	ALU Operation	Data Memory	Register Write	Total
R-type						
Load word	160	20	60	160	20	420
Store word						
Branch						
Jump						

19. What is the unit of measure for these numbers?
   A. Seconds
   B. Milliseconds
   C. Microseconds
   D. Nanoseconds
   E. Picoseconds

20. What is the average time per instruction for this single-cycle design? __________

21. Assume a program has an instruction mix of 40% R-Type, 20% Load word, 20% Store word, 10% Branch, and 10% Jump instructions. If the clock period could be adjusted to match the values in the Total column, what would be the average clock period for the program? __________

22. How much faster would the variable-clock design be than the fixed-clock design? (You do not have to divide it out.) _________

23. What is Key0 used for in the DE1 implementation of the single-cycle design?
   A. It is not used
   B. The CPU clock
   C. To select which register to read
   D. To select which register to write
   E. To select which seven-segment display to use

24. Complete the following table for the multi-cycle design:

Instruction Class	Number of States
R-type	
Load word	
Store word	
Branch	
Jump	

25. What would be the average CPI for the multi-cycle design given the same instruction mix as Question 21? __________

26. If the clock period for the multi-cycle design is 160 picoseconds, what would be the average time per instruction? __________

27. Which is faster, and by how much: the variable-cycle design or the multi-cycle design? Answer below:

28. Why is there an Instruction Register in the multi-cycle design, but not the single-cycle design?
 A. Because it is all right for the instruction to change during the single-cycle design.
 B. Because the memory may need to be read from twice in one instruction in the multi-cycle design, and the instruction must not be overwritten by the second read.
 C. It makes the ALU faster.
 D. It eliminates the need to calculate the Branch Target Address.
 E. It allows the Register File to do simultaneous reads and writes.

29. What is the purpose of the Control Unit?
 A. It coordinates the inputs from the control signals and generates the clock synchronization pipeline register's alternate latch mechanism.
 B. It regulates the passive activity of the multi-function operational shift-add butterfly network.
 C. It manages the integration of the substitute activity generator with the regulated hypermedia generator.
 D. It integrates the pipeline stall detection network with the cache coherence processor in order to minimize races, glitches, and priority inversions.
 E. It generates the control signals for the datapath.

30. What would be the clock period of a pipelined processor with five perfectly-balanced stages compared to a single-cycle implementation of the same instruction set?
 A. They would be the same.
 B. The pipeline's period would be five times longer.
 C. The single-cycle's period would be five times longer.
 D. 5 GHz
 E. 5 nanoseconds

31. How many clock cycles does it take an instruction to go through a 5-stage pipeline?
 A. 1
 B. 2
 C. 3
 D. 4
 E. 5

32. What is the ideal number of clock cycles per instruction for a long-running program?
    A. 1
    B. 2
    C. 3
    D. 4
    E. 5

33. What information is held in the pipeline register between the Instruction Fetch and the Instruction Decode stages of the pipeline design in the textbook (the IF/ID register)? *Hint: there are 64 bits.* Answer below:

34. What information is in the last pipeline register (MEM/WB). *Hint: There are 71 bits, two of which are the RegWrite and MemtoReg control signals and none of which were in the IF/ID register in the previous question.* Answer below.

35. What is the name for the problem in a pipeline design when an instruction needs the result of a previous instruction to use as an operand, but that result has not been written back into the register file yet?
    A. The register write delay loop problem
    B. Control hazard
    C. Data hazard
    D. Dukes of hazard
    E. Syntax error

36. What technique can be used to reduce the problem in Question 35?
    A. Branch prediction
    B. Speculative execution
    C. Code optimization
    D. Register forwarding
    E. Split cache design