**Instructions**

**Format:** This exam consists of 6 multiple choice questions for each of the 12 lectures in the course. You may have to do some calculations in order to determine the correct answer. The motivation for having so many questions is to get a more accurate assessment of your knowledge of the material.

Be sure to circle the answers on the page titled “answers page” or your answers will not be graded. If you choose not to answer a question do not circle a letter. Feel free to detach the answers page from the remainder of the exam and only turn it in.

**Grading:** To discourage guessing, each incorrect answer will be worth \(-\frac{1}{3}\) point, while each correct answer is worth +1 point. I.e., if you guess randomly, the expected score is 0.

**Material:** You are allowed a calculator and one double-sided, hand-written A4 sheet of notes during the exam.

Good luck!

---

**Note:** Please try to get a good grade on this exam. While everyone loves creating and grading exams, it does take a significant amount of time that could be (better?) spent improving the course.
0. Optional anonymous background questions
0.1. A B C
0.2. A B C
0.3. A B C
0.4. Program: _________________

1. ISA 1
1.1. A B C D
1.2. A B C D
1.3. A B C D
1.4. A B C D
1.5. A B C D
1.6. A B C D

2. ISA 2
2.1. A B C D
2.2. A B C D
2.3. A B C D
2.4. A B C D
2.5. A B C D
2.6. A B C D

3. Computer Arithmetic
3.1. A B C D
3.2. A B C D
3.3. A B C D
3.4. A B C D
3.5. A B C D
3.6. A B C D

4. Logic
4.1. A B C D
4.2. A B C D
4.3. A B C D
4.4. A B C D
4.5. A B C D
4.6. A B C D

5. Processor Control and Datapath
5.1. A B C D
5.2. A B C D
5.3. A B C D
5.4. A B C D
5.5. A B C D
5.6. A B C D

6. Pipelining
6.1. A B C D
6.2. A B C D
6.3. A B C D
6.4. A B C D
6.5. A B C D
6.6. A B C D

7. Hazards
7.1. A B C D
7.2. A B C D
7.3. A B C D
7.4. A B C D
7.5. A B C D
7.6. A B C D

8. Branch Prediction and Exceptions and Interrupts
8.1. A B C D
8.2. A B C D
8.3. A B C D
8.4. A B C D
8.5. A B C D
8.6. A B C D

9. Input/Output
9.1. A B C D
9.2. A B C D
9.3. A B C D
9.4. A B C D
9.5. A B C D
9.6. A B C D

10. Caches
10.1. A B C D
10.2. A B C D
10.3. A B C D
10.4. A B C D
10.5. A B C D
10.6. A B C D

11. Virtual Memory
11.1. A B C D
11.2. A B C D
11.3. A B C D
11.4. A B C D
11.5. A B C D
11.6. A B C D

12. Parallelism
12.1. A B C D
12.2. A B C D
12.3. A B C D
12.4. A B C D
12.5. A B C D
12.6. A B C D

Feel free to detach the remaining pages of the exam and turn in only your answers.
0. OPTIONAL anonymous background questions
(These are not graded and will not affect your grade on the exam. They are for me to be able to correlate what helps students prepare for the exam. Feel free to not answer these questions if you do not want to.)

1. How much of the book did you read?
   a. All
   b. Some
   c. None

2. How many of the practice sessions did you miss/turn in late?
   a. Several
   b. One
   c. None

3. How many of the previous years’ exams did you look at?
   a. All
   b. Some
   c. None

1. ISA 1

1. How many bytes of arbitrary data can a program store in the MIPS register file?
   a. $2^{32}$
   b. 124
   c. 32
   d. 31

2. Which of the following addresses are word-aligned?
   a. 1110 1011 0000 1110
   b. 0100 0011 0000 1010
   c. A and B
   d. Neither

3. The following assembly code should implement if (a==b-c) {a=a+2;}. What should the missing instruction be? (a is in R5, b is in R6, and c is in R7)
   ```
   sub   R2, R6, R7
   addi  R5, R5, 2
   __________
   skip:
   a. beq   R5, R2, skip
   b. beq   R6, R2, skip
   c. j     skip
   d. bne   R5, R2, skip
   ```

4. What address does the following code load into R4?
   ```
   lw   R2, 16(R0)
   sll  R2, R2, 16
   lw   R3, 20(R0)
   slr  R3, R3, 16
   or   R4, R2, R3
   ```
   a. 17
   b. 18
   c. 19
   d. 21

5. How many sequencing instructions do you need to implement an if-then-else in MIPS assembly?
   a. 0
6. How much space do branch labels (e.g., “skip:”) take up in the code?
   a. None
   b. 16 bits
   c. 4 bytes
   d. Depends on the length of the label

2. ISA 2

1. What size adder do we need to include the immediate into the next PC for a jump instruction?
   a. None
   b. 4 bit
   c. 26 bit
   d. 32 bit

2. How would you conditionally jump 1 million instructions forward in a program?
   a. Use a standard bne/beq instruction
   b. Use a standard j instruction
   c. Use a bne/beq instruction to control whether you execute a j instruction
   d. You can’t because conditional branches only have one constant

3. Which registers need to be saved in my_procedure?
   my_procedure:
   addi $t1, $a0, 4
   add $s0, $t1, $a1
   sub $a2, $s0, $a1
   jal fancy_math
   sub $v0, $t2, $v1
   jr $ra
   a. $s0, $ra
   b. $s0, $t1, $ra
   c. $s0, $t2, $ra
   d. $s0, $t1, $t2, $ra

4. We used the lui (load upper immediate) and the ori (or immediate) instructions to load a full 32-bit constant. What can you say about the sign extension of the immediate field?
   a. All immediates are sign-extended
   b. At least the ori instruction must not sign-extend its immediate
   c. Can’t conclude anything since sign-extending as part of an OR won’t matter
   d. This code example is too simple to tell us anything

5. Imagine a new MIPS instruction: bnei REG, CONST, ADDRESS which branches to ADDRESS if REG!=CONST in the same way that bne does. How many bits would this instruction require to encode? (You do not have to keep to the same instruction format as MIPS, but the decoding should be similar.)
   a. 16
   b. 32
   c. 48
   d. Depends on the size of the constants

6. Imagine we add a new MIPS instruction to the MIPS architecture discussed in class: bnei REG, CONST, ADDRESS which branches to ADDRESS if REG!=CONST in the same way that bne does. If we allow an 8 bit CONST, how far can the branch jump?
   a. +31 / -32 instructions
   b. +64 / -64 instructions
3. **Computer Arithmetic**

1. What is the value of the binary string 1101?
   a. -5
   b. -3
   c. 13
   d. Could be any of the above

2. Why do we use binary numbers in computers?
   a. They are faster than base 10 signals
   b. They take up less space than base 10 signals
   c. They are more immune to noise than base 10 signals
   d. All of the above

3. Which 8-bit unsigned fixed-point representation would be most appropriate for storing the speed of a car whose speedometer goes up to 200km/h?
   a. 8.0
   b. 7.1
   c. 6.2
   d. 5.3

4. Why can we get unexpected results from floating-point math when we do (big+small)-big?
   a. Floating point addition is inaccurate regardless of the size of the numbers
   b. Combining large and small numbers can lose precision
   c. Both A and B
   d. Neither

5. A serial multiplier trades off what for what vs. a parallel multiplier? (The serial multiplier is…)
   a. Lower precision, but faster
   b. Faster, but larger circuit area
   c. Slower, but smaller circuit area
   d. They are the same, just different ways to look at the circuit

6. When a floating-point number is normalized by shifting the mantissa to the left, what happens to the exponent?
   a. Increases
   b. Decreases
   c. Stays the same
   d. Depends on the number

4. **Logic**

1. What happens if you activate two rows in a memory array at the same time?
   a. You corrupt the data you are reading
   b. You can read out twice as much data
   c. You can read and write at the same time
   d. You can read and write at the same time as long as it is to different rows

2. Why is DRAM so much cheaper than SRAM?
   a. DRAM doesn’t require electricity to retain its values
   b. DRAM memory cells are much smaller than SRAM cells
   c. DRAM doesn’t require transistors to work
   d. All of the above

3. As circuits get smaller, the size of the DRAM cells is shrinking. This means there are fewer and fewer electrons to store each bit in the DRAM. What implications does this trend have?
   a. The DRAM will become less reliable since it is harder to know whether you read out a 1 or 0 if there are fewer electrons.
b. The DRAM will become slower since you have to wait longer to get enough of the electrons out to see if it is a 0 or 1.
c. The DRAM will be more sensitive to cosmic rays that leave random electrons on the chip.
d. All of the above

4. What changes from a simple latch when you have two latches back to back with opposite clock signals?

![Two latches back-to-back with opposite clock signals](image)

a. The latch behaves the same but it takes longer for the data to get through
b. The data is only updated when the clock goes from low to high
c. The data only goes through the latch when the clock is low
d. The latch can store a value when the clock is low

5. What is the current state in a counter circuit?

a. The adder
b. The next count
c. The current count
d. The clock

6. What would happen if you built a counter circuit without a state memory (e.g., latch or flip-flop)?

a. It would work without a clock
b. The signals would feedback around and count uncontrollably
c. You couldn’t calculate the next state
d. All of the above

5. Processor Control and Datapath

![Datapath diagram](image)

1. Which instructions need the connection shown in the datapath above with the thick line?

   a. jr
   b. jal
2. Which of the following do not need a clock signal in the datapath?
   a. Adders
   b. MUXes
   c. Sign-extender
   d. All of the above

3. What are the control signals for bne?
   a. RegWrite=0, ALUSrc=SignExtend, ALUOp=Sub, PCSrc=!Zero
   b. RegWrite=0, ALUSrc=Read data 2, ALUOp=Sub, PCSrc=Zero
   c. RegWrite=0, ALUSrc=Read data 2, ALUOp=Sub, PCSrc=!Zero
   d. RegWrite=0, ALUSrc=SignExtend, ALUOp=Sub, PCSrc=1

4. What is read out of Read data 2 in the register file for an I-format instruction?
   a. Nothing
   b. The immediate
   c. A register based on the immediate value
   d. The same register as Read data 1

5. Why was a single “branch” output from the control logic not enough to handle both bne and beq instructions?
   a. We need to know which one looks at the ALU’s Zero output
   b. We need to interpret the ALU’s Zero output differently for each
   c. We do different calculations in the ALU for each
   d. All of the above

6. Why is the path that determines the clock speed of a processor typically determined by the lw or sw instructions?
   a. They need to use the sign extension logic
   b. They use the ALU
   c. They use the memory
   d. They only read one register file output

6. Pipelining

   1. What’s the trick to making pipelines fast?
      a. Lots of stages
      b. Even stages
      c. High clock frequencies
      d. Keeping the pipeline full

   2. Does the pipelined processor execute instructions in an atomic and in-order manner?
3. What value will be in R2 after this code executes on the pipeline shown above?
   ```
   addi R2, R0, 4
   add  R3, R0, R4
   beq  R3, R4, skip
   addi R2, R0, 0
   skip:
   addi R2, R2, 2
   addi R2, R2, 4
   a. 4  
   b. 6  
   c. 10 
   d. 12 
   ```

4. How many NOPs need to be inserted in this code to make it execute correctly on the pipeline shown above? (You are allowed to re-arrange instructions.)
   ```
   iterate:
   sub  R7, R7, -1
   or   R6, R6, R4
   bne  R4, R5, iterate
   addi R5, R5, 1
   addi R6, R1, 2
   a. 0  
   b. 1  
   c. 2  
   d. 3  
   ```

5. What is the speedup for a processor with a 200ns clock frequency that is equally pipelined to 25 stages with pipeline latches that take 2ns each?
   a. 10x  
   b. 17x  
   c. 20x  
   d. 25x  

6. What data is not stored in the EX/MEM pipeline register?
   a. Register File 1 output  
   b. Register File 2 output  
   c. ALU result  
   d. Branch address  

7. Hazards
   1. Forwarding helps when?
      a. If the answer is somewhere else in the processor  
      b. If the answer hasn’t been computed yet  
      c. If multiple parts of the processor need the answer at the same time  
      d. All of the above  
   2. Hazards are a result of what?
      a. Needing to keep the semantics of atomic and sequential execution  
      b. Executing instructions in parallel  
      c. Executing instructions in pieces  
      d. All of the above
3. What forwarding paths are needed to handle this code correctly?
   \[
   \begin{align*}
   &\text{lw } R12, 0(R3) \\
   &\text{add } R2, R12, R3 \\
   &\text{add } R3, R14, R8 \\
   &\text{a. None} \\
   &\text{b. MEM-->EX} \\
   &\text{c. WB-->EX} \\
   &\text{d. Can’t forward here}
   \end{align*}
   \]

4. What is the CPI (cycles per instruction) for a program where 30% of all instructions are loads and \(1/3\) of the instructions following the load depend on it?
   \[
   \begin{align*}
   &\text{a. 1.01} \\
   &\text{b. 1.1} \\
   &\text{c. 1.3} \\
   &\text{d. 1.33}
   \end{align*}
   \]

5. Which of these does not address a structural hazard?
   \[
   \begin{align*}
   &\text{a. Data forwarding} \\
   &\text{b. Separate comparator for branch detection} \\
   &\text{c. Having separate instruction and data memories} \\
   &\text{d. Double-pumping the register file}
   \end{align*}
   \]

6. Was the decision to have a branch delay slot in the MIPS ISA a good one in retrospect?
   \[
   \begin{align*}
   &\text{a. Yes, it allowed the compilers to produce faster code} \\
   &\text{b. Yes, but it was hard to use and required hand-written code} \\
   &\text{c. No, because it was too hard to use so no one every filled it} \\
   &\text{d. No, because newer processors had different pipelines but had to support the ISA}
   \end{align*}
   \]

8. Branch Prediction and Exceptions and Interrupts

1. Predictors help when?
   \[
   \begin{align*}
   &\text{a. If the answer is somewhere else in the processor} \\
   &\text{b. If the answer hasn’t been computed yet} \\
   &\text{c. If multiple parts of the processor need the answer at the same time} \\
   &\text{d. All of the above}
   \end{align*}
   \]

2. What is the problem with predictors?
   \[
   \begin{align*}
   &\text{a. It’s hard to predict the future with any accuracy} \\
   &\text{b. We have to clean up if we make mistakes} \\
   &\text{c. Predicting the future takes as long as computing it} \\
   &\text{d. All of the above}
   \end{align*}
   \]

3. Why do we need a branch target buffer (BTB)?
   \[
   \begin{align*}
   &\text{a. To predict whether the branch is taken or not} \\
   &\text{b. To keep track of which branch the prediction is associated with} \\
   &\text{c. To avoid having to wait for the branch address calculation} \\
   &\text{d. To determine if the prediction is valid}
   \end{align*}
   \]

4. What percentage of the time will a 2-bit predictor that is initialized to strongly not-taken predict the if-branch correctly?
   \[
   \text{for (j=0; j<100,000,000; j++)} \{ \\
   \text{if (j % 2 == 1) \{ } \\
   \text{\hspace{1cm} // Do something} \\
   \text{\}} } \\
   \}
   \]
   \[
   \begin{align*}
   &\text{a. 0%} \\
   &\text{b. 1/100,000,000 (almost 0%)} \\
   &\text{c. 50%} \\
   &\text{d. 100%}
   \end{align*}
   \]
5. Why is BTFN (backwards taken, forwards not-taken) a reasonable idea for a predictor?
   a. It isn’t
   b. Loops jump backwards and you mostly take those branches
   c. Loops jump forwards and you mostly don’t take those branches
   d. Error checks are usually taken

6. If you have a branch predictor that keeps track of branch history based on the least significant bits of the PC and can store information for 16 separate branches, what happens when you have a program with the following behavior:
   branch at 0010 1011 1010 1100 always not taken
   branch at 0010 1011 1110 1100 always taken
   a. The program crashes
   b. The branch predictor will only predict for one of the branches
   c. The branch predictor will confuse branches and predict incorrectly
   d. All of the above

9. Input/Output
   1. How can a solid state (flash) disk have both lower latency and higher throughput than a traditional (spinning) hard disk?
      a. They have multiple read heads so they can read in parallel
      b. They have lower controller overhead so each read takes less time to start
      c. They have no moving parts so it doesn’t have to wait for the data to physically move to the right place to access
      d. All of the above

2. Why is the throughput enormously lower for reading small chunks of random data from a spinning hard disk than for reading large chunks?
   a. It takes a long time to move the read head between each small chunk
   b. The small chunks are stored on the inside track of the disk which is slower
   c. The controller has to process more to find many small chunks than one large chunk
   d. All of the above

3. Why are the wires on circuit boards often not straight?
   a. To try and fit more wires on the board
   b. To make them the same speed as other wires
   c. To save on copper
   d. To reduce their capacitance and inductance

4. Why does a serial link transmit both + and – versions of the signal?
   a. To save energy
   b. To increase speed
   c. To decrease noise
   d. All of the above

5. Why does polling give you the lowest latency for I/O?
   a. The transfer is handled by hardware so you don’t have to wait for the CPU
   b. There is no other code running so you can process the data quickly
   c. The hardware interrupts the current code to tell you when the I/O is ready
   d. All of the above

6. Why does DMA give you the highest throughput?
   a. The transfer is handled by hardware so you don’t have to wait for the CPU
   b. There is no other code running so you can process the data quickly
   c. The hardware interrupts the current code to tell you when the I/O is ready
   d. All of the above

10. Caches
    1. How do caches provide the illusion of a large and fast memory system?
       a. The make access to all data faster
b. They make access to the right data faster
   c. They make access to recently used data faster
   d. They make access to the least recently used data faster

2. Which of the following assumptions is not required for caches to be effective?
   a. Recently used data is likely to be used again
   b. The amount of data being used at any one time is small
   c. Data near recently used data is likely to be used soon
   d. Data is accessed in regular patterns

3. How many possible locations are there for storing the address 0100 0101 1110 0100 in a 64kB 4-way set-associative write back cache with 32 byte cache lines?
   a. 1
   b. 4
   c. 16
   d. 2048

4. Which has a lower average memory access time?
   #1: 32kB L1 cache 1 cycle + 256kB L2 cache 4 cycles (80% hit in the L1, 20% in the L2)
   #2: 64kB L1 cache 2 cycles + 128kB L2 cache 3 cycles (90% hit in the L1, 10% in the L2)
   a. #1
   b. #2
   c. Same
   d. Need to know something about the program

5. What percentage of the bits used for data in a 32kB (32,768 byte) direct-mapped write-back cache with a 64 byte cache line?
   a. 512/534
   b. 512/540
   c. 512/544
   d. 512/546

6. How did we use caches to solve the problem with an average of 1.33 memory accesses per instruction?
   a. Caches are faster, so it isn’t a problem
   b. Caches store data for recent instructions so we don’t need as many accesses
   c. We can have separate instruction and data caches to read both at the same time
   d. We have multiple levels of caches to make the cache seem larger

11. Virtual Memory

1. Which of these problems does virtual memory not solve?
   a. All applications having the same 32-bit address space
   b. Not having enough physical memory for what the ISA allows
   c. Providing fast access to recently used data
   d. Preventing applications from corrupting each other

2. What can virtual memory do for you if you have a 32-bit ISA and 8GB of physical memory?
   a. Nothing: a 32-bit ISA can only access 4GB of memory
   b. Not needed: a 32-bit ISA can have multiple programs access more than 4GB of memory without needing virtual memory
   c. A little: you can use the physical memory beyond 4GB instead of paging to disk for a single application
   d. A lot: you need virtual memory to have multiple 32-bit applications use more than 4GB of memory together

3. Why do we use pages to translate virtual addresses to physical addresses?
   a. Pages are faster than translating individual addresses
   b. Can address more physical memory using pages
   c. Can address more physical memory with a smaller TLB using pages
4. Which of the following pages would be good candidates to mark as copy-on-write instead of read-only?
   a. Compiled program code
   b. Data shared between multiple programs
   c. Library code shared between multiple programs
   d. All of the above
5. What size virtual memory page should you choose to address the most physical memory without a page fault on a system that supports the following TLB configuration:
   a. 64 4kB pages
   b. 16 2MB pages
   c. 4 1GB pages
   d. Page size doesn’t affect the amount of physical memory you can address without a page fault
6. Which cache and TLB arrangement does not provide memory protection between programs?
   a. Virtually Indexed, Virtually Tagged
   b. Virtually Indexed, Physically Tagged
   c. Physically Indexed, Physically Tagged
   d. All provide the same protection

12. Parallelism
1. What does a lock do?
   a. Protect data so only one processor can access it at a time
   b. Synchronize data so that different caches see the same value
   c. Provide a way for a program to check if another processor is using the data
   d. All of the above
2. Why did manufacturers start putting multiple cores on each chip in 2004?
   a. It gave users better performance
   b. They couldn’t make the clocks any faster
   c. It was too expensive to cool chips at faster speeds
   d. All of the above
3. If I have 10 processor cores and 90% of my program is perfectly parallel, in what percentage of the original time will my program run?
   a. 1.9%
   b. 10%
   c. 19%
   d. 90%
4. Why do we need atomic instructions for building a lock?
   a. Need to make sure that no one interrupts us from the time we see the lock is available to the time we have acquired it
   b. Need to avoid hazards in the pipeline to avoid synchronization issues with other instructions in our program
   c. Need to make sure data in the cache is shared with all other processors
   d. All of the above
5. Improper uses of locks results in what:
   a. Program crashes
   b. Incorrect results
   c. Deaths
   d. All of the above
6. Does having write-through caches for each processor mean you do not need to check for coherency between them?
   a. Yes, when data is changed it always goes back to the main memory
   b. Yes, when data is changed it always goes to all the caches
   c. No, when data is changed there may still be old values in other caches that are just reading it
   d. No, when data is changed it only updates the local cache

Make sure you circled the answers on the answer page. Answers on other pages will NOT be graded.