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 -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!
0. **Optional anonymous background questions**

0.1. A B C
0.2. A B C
0.3. A B C

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. If a word is 8 bytes, how many word addresses are there in a 32-bit byte-addressable computer?
   a. 32
   b. $2^{32}$
   c. $(2^{32})-8$
   d. $(2^{32})/8$

2. What is the minimum number of simultaneous reads and writes needed for a register file to work with the MIPS ISA?
   a. 0 writes, 2 reads
   b. 1 write, 2 reads
   c. 1 write, 3 reads
   d. 2 writes, 3 reads

3. How does the register file change after executing “addi R0, R1, 1”?
   a. R0 has R1+1
   b. R0 has R0+R1+1
   c. R1 has R0+1
   d. None of the above

4. The instruction “lui R1, 350” does what?
   a. Loads the value at address 350 in to R1
   b. Loads the value 350 into R1
   c. Loads the value 350*2^16 into R1
   d. Stores the value 350 into the address in R1

5. How many temporary registers (not counting the input and output registers) are needed to compute R5=(R4-R3)+R2?
   a. 0
   b. 1
   c. 2
   d. 3

6. What is required to load any unaligned memory address in MIPS?
   a. A regular load word instruction
   b. Two load words, a shift, and an OR
   c. A load word, a load byte, a shift, and an OR
   d. You can’t load data from unaligned memory addresses
2. ISA 2

1. What does the “jal” instruction do?
   a. First saves the address of the next instruction in a user-specified register, and then jumps to the given address
   b. First jumps to the given address, and then saves the address of the next instruction in the $ra register
   c. First saves the address of the next instruction in the $ra register, and then jumps to the given address
   d. First jumps to the given address, and then saves the address of the next instruction in a user-specified register

2. Why is the instruction “bne R2, 5, loop” not possible in the MIPS ISA?
   a. Because I-type instructions don’t have enough immediate field bits for two constants
   b. Because R-type instructions only have enough immediate field bits for one constant
   c. Because J-type instructions don’t have any bits for registers
   d. Because I-type instructions need a constant offset, not a string like “loop”

3. A J-type instruction has 6-bits for the opcode and 26-bits of immediate field. It works by:
   a. Using the immediate as a signed offset to the current PC (imm <<2 + PC)
   b. Using the immediate as a signed offset to the next PC (imm<<2 + PC+4)
   c. Replacing the lower bits of the next PC directly (imm <<2 overwrites bits 2 to 28 of PC+4)
   d. Directly using the immediate value for the new PC (next PC is the immediate value)

4. What is the value of $ra right after instructions 8, 24, and 34?
   a. 8, 24, 34
   b. 8, 24, 24
   c. 12, 28, 12
   d. 12, 28, 28

5. For question 4 above, what should the value of $ra be right after instructions 8, 24, and 34 such that the program executes instruction 12 after finishing process and add?
   a. 8, 24, 34
   b. 8, 24, 24
   c. 12, 28, 12
   d. 12, 28, 28

6. How does the code need to be modified to make it exit correctly?
   a. Store $ra on the stack before instruction 8 and restore it before instruction 12
   b. Store $ra on the stack before instruction 24 and restore it before instruction 28
   c. Store $ra on the stack before instructions 28 and 34, and restore it before instructions 32 and 36
   d. Use the caller/callee register saving conventions for R1, R2, R3, and R4

3. Computer Arithmetic

1. Compute 0011 – 1011 for 4-bit signed magnitude numbers.
1. A XOR B is the equivalent of which of the following?
   a. \((A \ AND \ !B) \ OR \ (!A \ AND \ B)\)
   b. \((!A \ AND \ !B) \ OR \ (A \ AND \ B)\)
   c. \((A \ AND \ !B) \ AND \ (!A \ AND \ B)\)
   d. \((!A \ AND \ !B) \ AND \ (A \ AND \ B)\)

2. How many control bits are needed to control a Multiplexor with 8 inputs?
   a. 2
   b. 3
   c. 8
   d. \(2^8\)

3. The combinational logic does what in a sequential circuit?
   a. Store the current value
   b. Store the next value
   c. Calculate the current value
   d. Calculate the next value

4. What limits the speed of a sequential circuit?
   a. The slowest latch
   b. The slowest path through the combinational logic
   c. The slowest latch + the slowest path through the combinational logic
5. Which is faster, and why? DRAM or SRAM?
   a. DRAM: each bit is stored in a powered feedback loop so it can be read quickly
   b. DRAM: each bit is stored in a passive capacitor so it can be read quickly
   c. SRAM: each bit is stored in a powered feedback loop so it can be read quickly
   d. SRAM: each bit is stored in a passive capacitor so it can be read quickly

6. Which one of these circuits updates its output only when the clock goes from 1 to 0?
   a. ![Circuit A]
   b. ![Circuit B]
   c. ![Circuit C]
   d. ![Circuit D]
c. Check zero and branch type
d. Always set if beq

3. What needs to be added to the processor shown above to support the jr instruction?
   a. Path from the Register File to the PC
   b. Path from the PC to the Register File
   c. Another write input to the Register File
   d. All of the above

4. What is the shortest clock cycle time for the processor above if the instruction memory takes 5ns, the registers take 1ns to read or write, sign-extension takes 0.5ns, the ALU takes 2ns, the data memory takes 10ns, and the branch computation takes 2ns after the sign extension?
   a. 8ns
   b. 18ns
   c. 19ns
   d. 21ns

5. Which element in the processor is not a state element?
   a. Instruction memory
   b. Registers
   c. ALU
   d. Data memory

6. Why do we not have hazards in the single-cycle processor?
   a. Every instruction finishes before the next one starts so the register file always has the right data
   b. No instructions are executing in parallel so there are no conflicts between instructions
   c. Each instruction has access to the whole processor while it is executing so there is no fighting for resources
   d. All of the above

6. Pipelining

1. What speedup will we get if we divide each instruction in a single-cycle processor into 8 pieces and run the clock 8 times faster?
   a. 1/8th the speed
   b. Same speed
   c. 8x faster
   d. Depends on how many stalls we need to handle hazards

2. A single-cycle processor takes 200ns per instruction. If we pipeline it to 20 stages, and each pipeline register has 2ns delay, how much faster can it run?
   a. Same speed
   b. 14.3 x faster
   c. 16.7 x faster
   d. 20.0 x faster

3. In the previous question, how long does each instruction now take?
   a. 12ns
   b. 14ns
   c. 200ns
   d. 240ns

4. Why do we need to store Register File output 2’s data in the pipeline register for the MEM stage?
   a. Needed for the store instruction’s data
   b. Needed for the store instruction’s address
   c. Needed for writeback later in the WB stage
   d. We don’t need to store it
5. A single-cycle processor has three types of instructions that take different amounts of time and make up different amounts of the program. (30% Type 1=600ps, 30% Type 2=700ps, and 40% Type 3=800ps.) How much faster or slower is a pipelined processor with a 200ps clock, assuming no latch overhead and no hazards?
   a. 0.3x
   b. 1.0x
   c. 3.6x
   d. 4.0x

6. A 6-stage pipeline processor runs at 6x the clock speed of a single-cycle processor. Assume no hazards and no overhead. How much faster can the pipelined processor execute 6 instructions than the single-cycle version?
   a. 1.0x
   b. 3.3x
   c. 6.0x
   d. 36.0x

7. Hazards
   1. Which one of these is not a hazard
      a. Data hazard
      b. Control hazard
      c. Pipeline Register hazard
      d. Structural hazard
   2. Double-pumping the register file fixed what kind of hazard?
      a. Data hazard
      b. Control hazard
      c. Pipeline Register hazard
      d. Structural hazard
   3. How many hazards does the code below have for the standard 5-stage MIPS pipeline without any forwarding logic?
      ```
      add r1, r2, r3
      add r4, r2, r2
      ld r5, 0(r4)
      add r4, r5, r4
      ```
      a. 0
      b. 1
      c. 2
      d. 3
   4. How many bubbles are needed in the pipeline to correctly execute the following code assuming no forwarding?
      ```
      add r1, r2, r5
      sub r2, r2, r3
      add r3, r5, r5
      ld r4, 0(r1)
      beq r1, r2, done
      add r1, r2, r3
      sub r2, r2, r4
      ```
      a. 0
      b. 1
      c. 2
      d. 3
   5. What is the CPI if 20% of the instructions generate results used by the next instruction and there is no forwarding? (Assume there are no other hazards in the pipeline.)
      a. 0.6
b. 1.0
c. 1.2
d. 1.4

6. Why couldn’t we move the branch computation logic even further forward in the pipeline?
   a. We need to read the register file
   b. We can’t put four ALUs in one stage
   c. We need to put the instruction in the pipeline register
   d. We can’t update the PC while reading the instruction memory

8. Branch Prediction and Exceptions and Interrupts
   1. What can we do to avoid the penalty of the branch delay slot?
      a. Double-pump the register file
      b. Re-order instructions
      c. Put in a bubble
      d. Move the branch logic forward
   2. What is the slowdown due to a branch predictor that has 90% accuracy on the Intel
      Nehalem processor with a 17 cycle branch misprediction cost and a normal CPI of 1.0 for a
      program with 25% branches?
      a. 0.4
      b. 1.4
      c. 3.8
      d. 4.8
   3. How many mispredictions does the BTFNT predictor make for this loop?
      \[ k=0 \]
      \[ \text{loop:} \]
      \[ k++ \]
      \[ \text{branch (} k\geq 10 \text{) to skip} \]
      \[ j++ \]
      \[ \text{skip:} \]
      \[ \text{branch if (} k\leq 1000 \text{) to loop} \]
      a. 1
      b. 11
      c. 991
      d. 1010
   4. Why do we need a valid bit but no dirty bit in the branch target buffer?
      a. Branch targets can be invalid, but we never write to the BTB
      b. Branch targets can be invalid, but we never write the BTB back to DRAM
      c. Branch targets can not be invalid, so we always write them back
      d. We don’t: we need both a dirty bit and a valid bit in the BTB
   5. Was the decision to have a branch delay slot in MIPS a good one?
      a. Yes, because it enabled early MIPS processors to be easier to build
      b. Yes, because compiler technology can easily fill it
      c. No, because newer MIPS processors have to pretend to have just one
      d. No, because compilers can’t reliably fill it
   6. How many exceptions could you have at the same time in a 5-stage MIPS pipeline?
      a. 2
      b. 3
      c. 4
      d. 5

9. Input/Output
   1. Why don’t we use DRAM for data storage instead of disks and flash?
      a. Too expensive
1. What is the problem that caches try to solve?
   a. ISA promised a full 32-bit address space
   b. We want a large and fast memory
   c. We want a cheap and reliable memory
   d. All of the above

2. Caches work by doing what?
   a. Moving the current data into faster memories
   b. Combining slower Flash and faster DRAM
   c. Combining slower hard disk and faster DRAM
   d. Making SRAM as cheap as DRAM

3. An N-line, direct-mapped, write-through cache needs what in addition to the data?
   a. N valid bits, N dirty bits, N tags, N comparators
   b. N valid bits, N tags, N comparators
   c. N valid bits, N dirty bits, N tags, 1 comparator
   d. N valid bits, N tags, 1 comparator

4. A processor with a nominal CPI of 1.0 has one cache for all memory accesses. It takes 1 cycle for a hit and 100 cycles for a miss. If 33% of the instructions are memory accesses and the cache has a miss ratio of 10%, what is the average memory access time per instruction?
   a. 3.3
   b. 3.6
   c. 13.3
5. Which addresses are present in a LRU, 2-way, set associative, write-through, cache with 1 byte per line and 4 entries after the following addresses are accessed in this order?
   1, 2, 3, 4, 5, 2, 6
   a. 2, 4, 5, 6
   b. 2, 3, 5, 6
   c. 3, 4, 5, 6
   d. 1, 4, 5, 6

6. What percentage of the data in a cache with a 64-byte line is used by the following code?
   int data[100]; // each int is a word
   for (int i=0; i<96; i+=4) {
     data[i]++;
     data[i+3]--;
   }
   a. 100%
   b. 75%
   c. 50%
   d. 25%

11. Virtual Memory

   1. What is not a benefit of having a virtual address to physical address translation?
      a. Memory protection
      b. Using disk as memory
      c. Faster cache accesses
      d. Using fragmented memory

   2. If each page table entry maps one word and requires 4 bytes, how large is the whole page table for a 32-bit machine?
      a. 4 bytes
      b. 4kB
      c. 4MB
      d. 4GB

   3. If a machine does not translate 13 bits of the virtual address, how large are its pages?
      a. 4kB
      b. 8kB
      c. 2MB
      d. 2GB

   4. Do page tables need a dirty bit?
      a. No, they don’t store data, just VA-to-PA mappings
      b. No, they are read only
      c. Yes, the bit indicates if the data is on disk
      d. Yes, the bit indicates if the page has been changed since it was loaded from disk

   5. What is the problem with a virtually indexed, virtually tagged cache?
      a. Have to access the TLB before the cache lookup
      b. Cache size is limited by the page size
      c. Need extra bits to tell which processes owns each line
      d. Processes can not share data

   6. What happens when process 1 loads address 10 given the page tables below?

<table>
<thead>
<tr>
<th>process</th>
<th>VA</th>
<th>PA</th>
<th>valid</th>
<th>dirty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
   a. It loads physical address 7 from DRAM
b. It loads physical address 10 from DRAM

c. It causes a page fault and the OS loads the data for physical address 7 from disk before returning it to the program

d. It causes an protection violation and the OS kills it

**12. Parallelism**

1. Why can’t we just keep increasing frequency to get better performance?
   a. Increasing frequency increases power
   b. We can’t afford to cool chips beyond about 300W
   c. Voltage isn’t going down so power goes up
   d. All of the above

2. If I have a Intel Xeon Phi processor with 62 cores, what is the best speedup I could get from a program that is 99% parallelizable?
   a. 38.5
   b. 58.4
   c. 61.4
   d. 62.0

3. What do you **not** need to add to the pipeline to support executing load/store instructions in parallel with other instructions?
   a. Extra ALU
   b. Extra data memory ports
   c. Extra register file ports
   d. Extra instruction memory ports

4. Which of the following does **not** use parallelism to improve performance?
   a. Associative caches
   b. Register File
   c. ALUs
   d. None: they all use parallelism

5. What does adding instruction level parallelism to the processor **not** do?
   a. Make the processor more complicated
   b. Reduce hazards
   c. Reduce CPI
   d. Simplify the compiler’s job

6. The cache for processor 2 wants to load an address. What does the cache for processor 1 need to check to see if it needs to write back data and tell processor 2’s cache to wait for it?
   a. If it has the line (tag and valid) and it is dirty
   b. If it has the line (tag and valid) and it is not dirty
   c. If it has the line (tag and valid)
   d. If it has the dirty line (tag and dirty)

---

**Make sure you circled the answers on the answer page.**

**Answers on other pages will NOT be graded.**