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!

Note: Please take this exam seriously and pass it so you do not have to take future makeup exams! While it is essential to give students appropriate chances to demonstrate their knowledge, producing and grading an exam consumes significant teaching resources that could be spent otherwise improving the course. For example, this exam will cost the IT department roughly 10,000SEK in teaching time to prepare, administer, and grade. For that much effort we could have revised a whole lab.
0. Optional anonymous background questions
0.1. A B C
0.2. A B C
0.3. A B C D
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.

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

2. How many practice exams did you do?
   a. None
   b. One
   c. Several

3. How much time did you spend studying for this exam?
   a. None
   b. <2 hours
   c. 2-8 hours
   d. >8 hours

1. **ISA 1**

1. Which of the following addresses is word-aligned?
   a. 0110 1010 0110 1000 0110 1001 1010 0110 1000 1100 1100
   b. 0000 0100 1000 1011 0110 1000 1000 1100 1100 1100
   c. 0000 0010 1000 1010 0110 1000 1110 1111 1111
   d. 0000 0000 0000 1010 0110 1000 1010 1110 1110

2. Why do we need both a register read and an ALU operation to write to memory?
   a. We don’t
   b. The register provides the write data and the ALU calculates the address
   c. The ALU always does something regardless of whether we use the results
   d. The register provides the address and the ALU proves the write data

3. What address does the following code load?
   ```
   addi R9, R0, 24
   addi R1, R9, 12
   addi R9, R0, 12
   lw R9, 12(R1)
   ```
   a. 12
   b. 24
   c. 36
   d. 48

4. Why do branch labels (e.g., “skip:”) not take up any space in the program code?
   a. We do not need a constant to jump to a label
   b. The label is included in instruction after the label
   c. We just use the label to know what constant to put in the branch/jump
   d. They do take up space!

5. What value should be put in X to load the word starting at address 25 into R8?
   ```
   lw R2, 24(R0)
   sll R2, R2, 8
   lw R3, 28(R0)
   sll R3, R3, X
   or R8, R2, R3
   ```
   a. 8
   b. 16
   c. 24
   d. 32
6. Why would we ever execute an add with R0 as a source register?
   a. To move data between registers
   b. To create a NOP
   c. A and B
   d. We wouldn’t: it adds 0 so it is pointless

2. ISA 2
1. Why is the instruction “bne R9, 1, loop” not possible in the MIPS ISA encoding?
   a. Because R-type instructions only have enough immediate field bits for one constant
   b. Because I-type instructions don’t have enough immediate field bits for two constant
   c. Because I-type instructions need a constant offset, not a string like “loop”
   d. Because J-type instructions don’t have any bits for registers

2. How would you load the value 9 into the PC?
   a. lui $pc, 0
   b. addi $pc, $r0, 9
      jr $pc
   c. lui $pc, 9
   d. addi $ra, $r0, 9
      jr $ra

3. What happens with the immediate for branch instructions, and why?
   a. It is shifted right by 8 bits because the lower byte is not necessary
   b. It is shifted right because instruction addresses are always multiples of 4
   c. It is shifted left because instructions are word-aligned
   d. It is not modified

4. How can the processor know if an instruction is I-type and not R-type?
   a. It looks at the opcode
   b. It looks if there is an immediate field
   c. It looks if the destination register is in the right place
   d. All of the above

5. This code will never finish. Why?
   main:
      addi  R1, R0, 4
      addi  R2, R0, 0
      jal   sum_from_next_address
      j     finish
   sum_from_next_address:
      addi  R1, R1, 4
      jal   load_data
      add   R2, R2, R3
      jr
   load_data:
      lw     R3, 0(R1)
      jr
   a. It doesn’t save temporary and saved registers
   b. It doesn’t save the return address register
   c. It doesn’t follow the register convention for argument and return registers
   d. All of the above

6. Why do we decrement the stack pointer when we store data to the stack?
   a. Because the ISA defines it that way
   b. Because memory addresses go down
   c. To avoid running out of space if we increment it
   d. All of the above
3. Computer Arithmetic

1. If we changed our 32-bit processor to have 256 registers, have the same size instructions, and keep similar I-, R-, and J-formats, how many bits would we waste if we loaded a 32-bit constant?
   a. 0
   b. 2
   c. 4
   d. 8

2. Which of the following statements is not true about two’s compliment notation?
   a. We can easily determine if the number is negative by looking at the MSB
   b. Addition is a non-inverted addition with carry set to 1
   c. Subtraction inverts one input and does an addition with carry in set to 1
   d. There is only one zero

3. For an 8-bit floating-point format: \((-1)^S*(FFFF)*(2^{EEE})\) where FFFF is unsigned and EEE is two’s complement, what is the largest positive value you can represent?
   a. 125
   b. 31.75
   c. 15.875
   d. 7.5

4. Which of these is not a valid interpretation of the bits 1010 according to the representations we’ve discussed?
   a. -6
   b. -2
   c. 2.75
   d. 10

5. How are binary numbers distributed along the number line for the fixed-point number format?
   a. Linearly
   b. Exponentially
   c. Depends on the binary point location
   d. Depends on the exponent

6. Which is not a benefit of floating point over fixed-point?
   a. Floating point can represent larger numbers
   b. Floating point can represent smaller numbers
   c. Floating point addition is simpler
   d. Floating point does not require more bits

4. Logic

1. How many output bits does an adder have if it is adding two 8-bit numbers?
   a. 5
   b. 8
   c. 9
   d. Depends on whether the numbers are two’s compliment or fixed-point

2. \((!A \text{ AND } B) \text{ OR } (!B \text{ AND } A)\) is the equivalent of which of the following?
   a. A AND B
   b. A OR B
   c. A XOR B
   d. A XNOR B

3. Why are transparent latches worse than edge-triggered flipflops?
   a. They are slower. They look if the clock signal is high or low the whole time, instead of just looking at the edge of the clock
b. They don’t avoid feedback loops as they allow the input to pass directly through to the output

c. They don’t allow the input to pass directly through to the output, so they slow down the circuit

d. They are bigger because they need more than two inverters

4. Which one of these circuits updates its output whenever the clock goes high?

a.

b.

c.

d.

5. Why is DRAM slower than SRAM?

a. There is a lot more of it, so it takes longer to get data from it to the processor

b. It’s built of capacitors which have less charge and take longer to produce output

c. It’s built of feedback loops which need external power to output their data

d. The wires are longer so it takes longer to charge up the wire and read the output

6. How would the output of this circuit change if we removed the inverters at the end of each feedback loop? (One is labeled A, the other is directly before the Out. Ignore the B label.)

a. The output would be inverted, but faster

b. The output would be inverted, but slower

c. The output would not change, but be faster

d. The output would not change, but be slower
5. Processor Control and Datapath

1. Which part of the processor is storing state?
   a. PC, registers, ALU
   b. PC, registers, ALU, instruction memory
   c. PC, registers, ALU, instruction memory, sign extension
   d. PC, registers, instruction memory

2. Why did our single-cycle processor design have two memories?
   a. It doesn’t, we just treat them as two different memories to understand better
   b. Different parts of the instruction access the instruction memory and the data memory
   c. Some instructions need to access the instruction memory and the data memory at the same time
   d. We don’t have a cache so it is faster if we have two memories

3. What do we need to add to the processor shown above to support the jr instruction?
   a. Another read and another write inputs to the Register File
   b. Path from the PC to the Register File and another ALU function to the ALU control
   c. Path from the Register File to the PC
   d. All of the above

4. Which type of branch and jump instructions are supported by the datapath shown above?
   a. j
   b. beq, j
   c. bne, j
   d. beq, bne, j

5. What would happen if the RegDst MUX was removed from the datapath and it’s 1 input was wired in directly?
   a. No instructions could write back to the register file
   b. I-type instructions could not write back to the register file
   c. I-type instructions would write back to the wrong register
   d. The processor would be slower due to hazards

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
e. All of the above

6. Pipelining

1. What is the key thing that makes pipeline processors faster?
   a. They can execute instructions in parallel
   b. Instructions can take different numbers of cycles
   c. Instructions are split into smaller steps
   d. All of the above

2. What value will be in R9 after this code finishes executing on the pipeline shown above?
   R1=1, R4=1, R9=1
   beq R1, R4, skip
   addi R9, R0, 0
   addi R9, R9, 2
   addi R9, R9, 4
   skip:
   add R0, R0, 0
   end:
   a. 0
   b. 1
   c. 6
   d. 12

3. What value will be in R9 after this code finishes executing on the pipeline shown above?
   R1=1, R4=1, R9=1
   beq R1, R4, skip
   addi R9, R0, 0
   skip:
   addi R9, R9, 2
   addi R9, R9, 4
   add R0, R0, 0
   end:
   a. 0
   b. 1
   c. 6
   d. 12

4. A processor takes 280ns for the longest instruction. The processor is pipelined with 14 (equal) stages using pipeline registers that take 10ns. What percentage of the resulting cycle time is used for computation?
   a. 33%
5. In the pipeline above, why do we need to store the ALU’s Zero output in the EX/MEM stage pipeline register? (Note the wire that would answer this question is not shown in the diagram.)
   a. We don’t
   b. We need it in the MEM stage
   c. We need it in the EX stage
   d. We need it in the IF stage

6. Does the pipelined processor keep the ISA’s promise of in-order and atomic execution?
   a. Atomic: yes, In-order: yes
   b. Atomic: yes, In-order: no
   c. Atomic: no, In-order: yes
   d. Atomic: no, In-order: no

7. Hazards

1. Double-pumping the register file fixes which of the following problems?
   a. Needing to read the register file early for branches
   b. Needing to get data to the register file early for all dependent instructions
   c. Needing to read and write the register file at the same time
   d. All of the above

2. What is needed to resolve the following hazard?
   add R14, R1, R13
   lw R1, 4(R14)
   add R14, R15, R15
   addi R14, R12, 0
   a. Forwarding from MEM to EX
   b. Forwarding from WB to MEM
   c. Can’t resolve, need to stall
   d. There is no hazard to resolve

3. What is needed to resolve the following hazard?
   lw R1, 4(R14)
   add R14, R1, R13
   add R14, R15, R15
   addi R14, R12, 0
   a. Forwarding from MEM to EX
   b. Forwarding from WB to MEM
   c. Can’t resolve, need to stall
   d. There is no hazard to resolve

4. Which of the following options contain only structural hazards?
   a. Calculating the branch condition and address at the same time; reading and writing the RF at the same time; writing the data read by the previous instruction.
   b. Reading the instructions and data at the same time; writing the data read by the previous instruction.
   c. Calculating the branch condition and address at the same time; reading the instructions and data at the same time; writing the data read by the previous instruction.
   d. Calculating the branch condition and address at the same time; reading and writing the RF at the same time; reading instructions and data at the same time.

5. A processor is ideally pipelined to 10 stages, but due to hazards, 10% of the instructions are NOPs. What is the speedup over the original processor?
   a. 10%
   b. 90%
   c. 9x
6. Which of these does not address a structural hazard?
   a. Having independent data and instruction memories
   b. Reading and writing the register file in the same cycle
   c. Extra comparator for branch decisions
   d. Forwarding from a later stage

8. Branch Prediction and Exceptions and Interrupts
   1. What happens if you have fewer entries in your branch predictor than you have branches in your program?
      a. The branch predictor can’t make a prediction so it will turn off
      b. The branch predictor will always predict incorrectly
      c. The branch predictor will be very likely to predict less accurately
      d. You won’t be able to run your program
   2. Which processor would you rather have, assuming the other characteristics of the processor are the same? Processor A with a 100 cycle branch misprediction penalty and a 99.9% accurate branch predictor or Processor B with a 10 cycle branch misprediction penalty and a 99% accurate branch predictor?
      a. A
      b. B
      c. They’re both just as good
      d. Need more information
   3. Which processor would you rather have, assuming the other characteristics of the processor are the same, and the pipelining speedup is only limited by branch mispredictions? Processor A with a 100 cycle pipeline, a 100 cycle branch misprediction penalty and a 99.9% accurate branch predictor or Processor B with a 10 cycle pipeline, a 10 cycle branch misprediction penalty and a 99% accurate branch predictor?
      a. A
      b. B
      c. They’re both just as good
      d. Need more information
   4. For which kind of application would a BTFN branch predictor work best?
      a. Applications with plenty of code that jumps forward over error handling code
      b. Applications with plenty of loops that are usually taken backwards
      c. Applications with loops that start out being taken all the time and then switch to not being taken later in the program
      d. All of the above
   5. Why is a 2-bit predictor often better than a 1-bit predictor?
      a. It learns faster
      b. It changes more slowly
      c. It uses less logic
      d. It predicts more accurately
   6. If an interrupt occurs which causes the operating system to switch to another program. Whose responsibility is it to save the current program’s registers to the stack?
      a. The current program
      b. The caller
      c. Don’t need to
      d. The operating system

9. Input/Output
   1. Why would you choose to use interrupts instead of polling?
      a. Polling has lower latency than interrupts
      b. Polling uses up less processor time
c. Interrupts use up less processor time
d. Interrupts have lower latency

2. Which of the following is true when it comes to making busses fast?
   a. Wires interfere with their neighbors
   b. Wires of different lengths take different amounts of time
   c. To send more bits at the same time you need more wires in a bus
   d. All of the above

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

4. What does Ethernet not do to avoid errors?
   a. Detect data collisions
   b. Use separate wires for each sender
   c. Retry sending the data
   d. Use a checksum

5. How do DMA and memory-mapping interact?
   a. DMA cannot be used if the device is memory-mapped
   b. DMA uses the device’s interrupt to move data
   c. DMA moves data by accessing the device directly
   d. DMA moves data using the device’s memory-mapped addresses

6. How does DMA make our computers faster?
   a. It transfers data faster than the CPU can
   b. It avoids the overhead of interrupts when data is ready
   c. It frees up the CPU from having to do the transfer
   d. All of the above

10. Caches

1. Why do we want caches?
   a. DRAM is slow
   b. Disks are slower than DRAM
   c. Virtual memory requires it
   d. All of the above

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

3. What data will be in a 4-entry, direct-mapped, cache with a least-recently-used replacement
   policy and one byte per line after the following memory accesses? address: 3, 2, 1, 0, 4, 5, 3, 1, 2
   a. 1, 2, 3, 5
   b. 1, 2, 3, 4
   c. 1, 3, 4, 5
   d. 1, 2, 4, 5

4. What data will be in a 4-entry, fully-associative, cache with a least-recently-used
   replacement policy and one byte per line after the following memory accesses? address: 3, 2, 1, 0, 4, 5, 3, 1, 2
   a. 1, 2, 3, 5
   b. 1, 2, 3, 4
5. Caches give programmers the illusion of what?
   a. Atomic load/store instructions in a pipeline
   b. A full 32-bit address space for each application
   c. A large and fast memory
   d. All of the above
   
6. What is the difference between a direct-mapped cache with a least-recently-used (LRU) replacement policy and a direct-mapped cache with a most-recently-used (MRU) replacement policy?
   a. The MRU policy will work less well, leading to a lower hit ratio
   b. The MRU policy will work less well, leading to a higher hit ratio
   c. The MRU policy will effectively have only one entry that gets used
   d. The caches will behave the same

11. Virtual Memory

1. How much data can a program access without a TLB miss on a machine with a 64 entry, 8-way set associative TLB where each page is 4kB, and a 16kB cache with a 64 byte line size.
   a. 16kB
   b. 256kB
   c. 1MB
   d. Don’t have enough information

2. What happens before the actual write to DRAM when a program with the following page table tries to write to address 14?
   VA→PA on disk access bits
   2→8 0 read/copy on write
   1→9 0 read/write
   14→7 0 read/write
   16→14 1 read/write
   13→6 1 read only
   a. VA→PA translation, then the data is loaded from the disk
   b. VA→PA translation, then a copy on write
   c. VA→PA translation, then a memory protection exception is generated
   d. VA→PA translation

3. Writing to which address with the following page table would result in a page being loaded from disk?
   VA→PA on disk access bits
   2→8 0 read/copy on write
   1→9 0 read/write
   14→7 0 read/write
   16→14 1 read/write
   13→6 1 read only
   a. 8
   b. 13
   c. 14
   d. 16

4. What can virtual memory do for you if you have an ISA with a 64-bit address space and 16GB of physical memory?
   a. Nothing: a 64-bit address space can address a lot of memory already so it is not useful
   b. Not needed: a 64-bit address space can have multiple programs access more than 16GB of memory without needing virtual memory
c. A little: you can use the physical memory beyond 16GB instead of paging to disk for a single application
   d. A lot: you need virtual memory to keep multiple applications from crashing each other.

5. Why is a Virtually Tagged, Physically Indexed cache bad?
   a. Slow: have to do the translation before you can access the cache.
   b. Insecure: no way to prevent different programs from sharing data in the cache.
   c. Both A and B
   d. It’s not: it is both fast and secure

6. The TLB operates like a cache for page table entries. Which of the following does a TLB have that a data cache does not?
   a. A replacement policy
   b. Dirty bits
   c. Permission bits
   d. Tag bits

12. Parallelism

1. Why do we do parallelism?
   a. To reduce power
   b. To get better performance
   c. To save power
   d. To get better performance, save power and overcome Moore’s law

2. What is a lock?
   a. It is a mechanism to protect data so only one processor can access it at a time
   b. It is a mechanism to let a program check if another processor is using the data
   c. It is a mechanism to synchronize data so that different caches see the same value
   d. All of the above

3. How much faster can my program run if I have 9000 cores and 20% of the program cannot be parallelized?
   a. 5x
   b. 1800x
   c. 7200x
   d. 9000x

4. Which of the following used parallelism to improve performance?
   a. Associative caches
   b. Address and branch calculations
   c. Register file access
   d. All of the above

5. The Intel Haswell Processor (CPU) with the Iris Pro 5100 integrated Graphics Processor (GPU) has the following characteristics: 40 GPU cores that can each execute 20.8B floating point operations per second per core and 4 CPU cores that can each execute 56B floating point operations per second per core. Which is better for a program that is 50% parallel?
   a. Run on the CPU cores
   b. Run on the GPU cores
   c. Same
   d. Need more information

6. The Intel Haswell Processor (CPU) with the Iris Pro 5100 integrated Graphics Processor (GPU) has the following characteristics: 40 GPU cores that can each execute 20.8B floating point operations per second per core and 4 CPU cores that can each execute 56B floating point operations per second per core. How much faster is running the parallel part on the GPU and the serial part on the CPU than running the whole thing on just the GPU?
   a. 4% faster
b. 1.4x faster
c. 2.8x faster
d. Need more information

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