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.
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. What address does the following code load?
   addi R3, R0, 12
   addi R2, R3, 12
   addi R1, R0, 4
   lw R1, 12(R2)
   a. 4
   b. 12
   c. 24
   d. 36
   The first line puts the constant 12 into R3
   The second line adds 12 to the value in R3 and stores the result in R2 (So R2 now contains 24)
   The third line puts the constant 4 into R1
   The fourth line offsets the content of R2 by 12, and thus loads the address 24 + 12=36.

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

3. In MIPS, what would happen if you try to write into the R0 register?
   a. Your program might crash
   b. Your operating system might crash
   c. The content of R0 is changed
   d. Nothing, this register always contains 0
   Answer: D.

4. In MIPS, what special meaning does the “exit:” label has compared to other labels?
   a. It has no special meaning
   b. It is used in branch instructions to jump to the end of a loop
   c. It is used in branch instructions to jump to the end of the program
   d. It marks the end of the current function
Labels only have the semantic you give them. You are free to choose whatever name you want.

5. If R4 contains an unsigned integer, what does the instruction “sll R4, R4, 2” do?
   a. It multiplies the content of R4 by 4
   b. It shifts the bits in R4 2 positions to the left
   c. A and B
   d. Neither A nor B

   sll stands for Shift Left Logical. It shifts the bit pattern in R4 N positions to the left. When working with unsigned integer, shifting the bit pattern N positions to the left is equivalent to multiplying by $2^N$. Therefore, shifting R4 2 positions to the left is also equivalent to multiply it by 4.

6. Which of the following will NOT copy the contents of R2 to R1?
   a. addi R1, R2, 0
   b. sll R1, R2, 0
   c. sw R1, R0(12)
   d. lw R2, R0(12)

The sw/lw combination stores R1 into memory and then loads it into R2. This is equivalent to copying R1 to R2, not R2 to R1.

2. ISA 2

1. Which statement is NOT true?
   a. J can only jump within its 256 MB ($2^{23}$) regions
   b. BEQ can sometimes jump (branch) further ahead than J can
   c. Either J or BEQ can always jump further than JR
   d. BEQ and BNE have the same restrictions for the jump target

   JR can specify a full 32 bit address, and have thus no restrictions on where to jump

2. main() uses t0, t1, t2, s0, B() uses t4, s3, s4, C() uses t1, t2, t3, t4, s0, s5. How many words on the stack are needed when main() calls B() and B() calls C()?
   a. 8
   b. 9
   c. 11
   d. 17

   main() is only a caller, so it saves: t0, t1, t2. B is a caller and a callee, so it needs to save: t4, s3, s4, ra; and C is only a callee, so it saves: s0, s5 = 9 total

3. Which of the following is NOT a valid statement?
   a. beq R1, R5, exit
   b. add RA,R0,T1
   c. sub R0,R0,R0
   d. addi T1,T2,63000

   The immediate is a signed 16 bit integer, so this value can not be represented (max is $2^{15} - 1$ or 32767).

4. What is written to $RA$ when the JR instruction is executed?
   a. Nothing
   b. PC
   c. PC + 4
   d. The previous value of the $RA$ register

   Nothing. However, the value is read.

5. What is a disadvantage of a load-store-register machine compared to a memory-register machines?
   a. More complex instruction decoder
b. Cannot be pipelined

c. **Less dense code**

d. Denser code

You generally need more instructions to achieve the same thing with a load-store isa.

6. When calling another procedure, a certain convention must be followed. Why?
   a. To minimize the stack size
   b. To avoid corrupting data from other programs.
   c. **Procedures do not know which registers their caller and callees use**
   d. There may be an arithmetic overflow, which must be checked beforehand

   *If you knew exactly which registers would and wouldn’t be used you wouldn’t need a convention, but then you couldn’t have libraries either.*

3. **Computer Arithmetic**

   1. Will the results be correct if you use an ALU designed for 32-bit two’s complement addition for 32-bit unsigned integers?
      a. **Yes, however the overflow flag is calculated differently, and may be incorrect**
      b. No, if MSB is 1 for any of the inputs, then the answer will be incorrect
      c. Yes, the same algorithm is used, regardless whether the input happen to be signed or not
      d. No, all signed numbers are subjected to the operation NOT(input) + 1 before an addition, which would produce garbage in the unsigned case

   *Two’s compliment uses the same logic as unsigned except for overflow.*

   2. Is it possible to complete hardware binary multiplication of two 32 bit numbers in a single clock cycle?
      a. **Yes**
      b. No, it would require too much power
      c. No, but it will likely be possible in the future, as chips get smaller
      d. No, you need registers to store partial products, which requires extra cycles

   *Yes it is possible. However the propagation delay would be long and it would require a fair amount of chip area, compared to an iterative solution.*

   3. For a 7-bit normalized floating-point format: \((-1)^S \times (FFFF) \times (2^{EE})\) where FFFF is unsigned and EE is two’s complement, what is the smallest positive value you can represent?
      a. 0.5
      b. **0.25**
      c. 0.125
      d. 0.28125

   \((0 \ 0000 \ 10) = 1*1.0*2^{-2} (=\frac{1}{4}).\)

   4. Compute 1010 - 1110 for 4-bit two's complement numbers?
      a. Overflow
      b. +4
      c. +7
      d. **-4**

   *1010-1110 is -6- -2 = -4, which can be represented as 1100.*

   5. What is the range of an 8-bit signed magnitude binary number?
      a. -64 to 64
      b. -64 to 63
      c. -128 to 127
      d. **-127 to 127**

   *1 bit for the sign, and 7 bits for magnitude (0 ... 2^7 - 1=0 ... 127)*

   6. Which of the following statements is true?
a. A non-normalized floating-point number can represent the same number in several ways.

b. Signed-magnitude has a wider range than two's complement numbers.
c. To get the absolute value of a two's complement number, you remove MSB and negate the remaining bits.
d. You can detect arithmetic overflow on unsigned numbers by comparing carry in and carry out of the MSB.

A, but not if the floating point number is normalized.

4. Logic

1. These two circuits do the same thing. If the inputs to all gates change at the same time, which of them will be faster?
   a. X output: Circuit A, Y output: Circuit A
   b. X output: Circuit B, Y output: Circuit A
   c. X output: Both the same, Y output: Circuit A
   d. X output: Both the same, Y output: Circuit B

   The circuits do the same thing for the X output so they will be the same speed (on AND gate). For the Y output, circuit B’s longest path is an inverter->AND->OR, while circuit A just has one OR gate. As a result circuit A has the shorter path and will be faster.

2. Why are DRAMs slower than SRAMs?
   a. They only have small transistors to generate the output
   b. They only have a small amount of charge to output
   c. They have longer wires for the output
   d. They aren’t

   DRAMs store their values in small capacitors, which means they only have a very small amount of charge to use to output their value. SRAMs have small transistors to generate the output, but those can still push out a lot more charge than DRAMs because they are attached to the power supply. In both cases the effect of the wire length is the same.

3. What does the decoder do in a memory?
   a. Take in the row to read and produce the address
   b. Take in the address and choose the row to read
   c. Take in the address and select the columns to output
   d. Take in the columns and produce the address

   The decoder takes the address and selects the row to read. We then use a mux to choose which of the columns to output.

4. Why are binary circuits more immune to noise than analog circuit?
   a. Binary values are further apart in terms of voltage than analog values, so they are harder to confuse
   b. Binary circuits generate un-polluted outputs (e.g., 1 or 0 out) after each circuit, so the noise doesn’t propagate through
   c. Binary circuits interpret values close to a 1 or 0 as a 1 or 0 even if they are not
5. In the above circuit, an adder takes 10ps, a multiplier 50ps, a latch 10ps, and a mux 1ps. What will happen if you run the clock at 25ps for the above circuit?
   a. It will always produce the wrong results
   b. It will always produce the right results
   c. It will only produce the right results if A=0
   d. It will only produce the right results if A=1

   The A=0 path (adder->MUX->latch) needs 21ps, and the clock is 25ps so that is not too fast. The A=1 path (multiplier->MUX->latch) needs 61ps, which is longer than the 25ps clock, so it will not produce the right value in time.

6. In the above latch each inverter takes 5ps. How much time before the clock rises does the data need to be ready such that when the clock rises there will be a stable signal to the second part of the latch, and how much time will it take after the clock rises before the new data is available on the output?
   a. 5ps before, 5ps after
   b. 10ps before, 5ps after
   c. 5ps before, 10ps after
   d. 10ps before, 10ps after

   The data needs to go around both inverters in the input to make sure it is stable so that when the clock changes it will not change. (Think what would happen if the clock changed before it went around both input inverters: the lower inverter would still be outputting the old value so it wouldn’t change as we expect.) However, the output will change after going through just one inverter. The second output inverter is only needed when the clock goes low again to keep the value there.
4. **Processor Control and Datapath**

1. How many inputs can a 6-bit multiplexer with 4 control bits switch between?
   a. 4  
   b. 6  
   **c. 16**  
   d. 64
   
   4 control bits means the MUX can choose between $2^4 = 16$ possible inputs. Each input has 6 wires.

2. If we were to replace the sign-extend module with a “zero-extend”, how would this affect I-type instructions?
   a. Not at all, the sign-extend is not used for I-type instructions  
   b. Not at all, the operand is already two's complement encoded  
   **c. If the operand value is negative, then it changes value**  
   d. If the operand value is positive, then it will become negative
   
   Adding 0 to msb of a negative turns it into a different, positive value. Positive values are not affected.

3. There is a hardware error in our processor chip. As a result none of the I-type instructions work as they should (the other types work fine). Where could the error be?
   a. **The ALU mux input control signal (ALUSrc) is stuck at zero (always chooses Reg File input)**  
   b. The sign-extend module sometimes flips LSB, so that a 1 becomes 0, and vice versa  
   c. MemToRg mux input control signal is stuck at one (always choose ALU output)  
   d. All of the above faults could be responsible for the error
   
   All I-type instructions require the data from the sign-extended immediate field. Changing this MUX would prevent that data from getting to the ALU.

4. Which of these statements is NOT true?
   a. There are no hazards in a single-cycle processor  
   b. Single-cycle processors have no use for a branch predictor  
   c. The ALU generally does not need a clock input
5. A single-cycle processor has three types of instructions: A takes 1ns, B takes 2ns, and C takes 10ns. A program with an instruction mix of 25% A, 70% B, and 5% C will run at what clock speed?
   a. 1000 MHz (1 / 1ns)
   b. 465 MHz (1/2.15ns)
   c. 333 MHz (1/3ns)
   d. 100 MHz (1/10ns)

   The slowest instruction path dictates the clock speed.

6. What are the control signals for BEQ?
   a. RegWrite=0, ALUSrc=Read Data 2, ALUOp=Sub, PCSrc=Zero
   b. RegWrite=0, ALUSrc=SignExtend, ALUOp=Sub, PCSrc=!Zero
   c. RegWrite=1, ALUSrc=Read Data 2", ALUOp=Sub, PCSrc=!Zero
   d. RegWrite=0, ALUSrc=SignExtend, ALUOp=Add, PCSrc=Zero

   BEQ does a subtraction so it needs to take data from the register file for both inputs and does not write back to the register file. The PC src comes from the Zero output of the ALU to branch if the results are equal.
Pipelining allows overlapping execution of multiple instructions with the same circuitry. The circuitry is usually divided up into stages, including instruction decoding, arithmetic, and register fetching stages, wherein each stage processes one instruction at a time.

3. What is the role of pipeline registers?
   a. They store the results of each stage so that the next stage can use them
   b. They allow the register file to store more values
   c. They reduce the number of NOPs we need to insert in the pipeline when executing a branch instruction
   d. All of the above
   *Pipeline registers store partial results from each pipeline stage and make them available to the next stage at the next clock cycle.*

4. A single-cycle processor takes 100ns per instruction. It is perfectly pipelined into 10 stages with pipeline registers that take 2ns and can be kept 60% full. What change in instruction throughput can you expect compared to the single-cycle processor?
   a. 60% slowdown
   b. 60% speedup
   c. 5x speedup
   d. 10x speedup
   *If perfectly pipelined the processor would have 10x the throughput, but each instruction takes 12ns (10 for the stage plus 2 for the pipeline register) so the pipeline will finish an instruction every 12ns, which is an increase of 100ns/12ns = 8.33x. However, if the pipeline is only 60% full, the throughput speedup will only be 60% of that, or 5x.*

5. A single-cycle processor takes 100ns per instruction. It is perfectly pipelined into 10 stages with pipeline registers that take 2ns and can be kept 60% full. What change in instruction latency can you expect compared to the single-cycle processor?
   a. 20% speedup
   b. 20% slowdown
   c. 60% speedup
   d. 60% slowdown
   *Each stage takes 12ns (10 for the stage + 2 for the pipeline register), and there are 10 of them, so an instruction will spend 120ns in total to complete, while it took 100ns on the single-cycle processor. This is a 20% slowdown in latency. The pipeline occupation plays no role in this.*

6. What is the advantage of a multi-cycle CPU compared to a single-cycle CPU?
   a. The clock speed is no longer limited by the slowest instruction path
   b. The CPU can analyze the lengths of the instructions currently in the pipeline and adjust its clock speed accordingly
   c. Each instruction can take a different number of cycles to execute
   d. A and C
   *In a single-cycle CPU, the clock speed is limited by the slowest instruction path. In a multi-cycle CPU, each instruction can take a different amount of cycles, which means that "slow" instructions can use more than one cycles to complete.*

7. Hazards
   1. What is needed to resolve the following hazard?
      and R1,R2,R1
      ld R2,0(R3)
      add R2,R4,R1
      a. Forward from MEM to EX
      b. Forward from WB to EX
      c. A and B are both required
2. A 5-stage MIPS pipeline has a double-pumped register file and forwarding from WB to EX but not from MEM to EX. How many NOPs will you need to add to make this code work?
add R10, R12, R13
sub R12, R13, R11
add R13, R12, R11
   a. 3
   b. 2
   c. 1
   d. 0
You need a single NOP to use the forwarding for the R12 dependency.

3. An 8-stage pipeline processor requires NOPs to avoid hazards. The compiler is aware of this and inserts appropriate NOP instructions into the code. What would happen if the processed code now is run on a single-cycle processor?
   a. The program would execute correctly
   b. Conditional branches would not work correctly, due to the zero output from the ALU
   c. The code would not work correctly because the NOPs are in the wrong place
   d. The code would not work correctly because there are too many NOPs
   Everything would execute correctly since a single-cycle processor does not have any hazards, however the NOPs would make the program run slower than necessary.

4. What kind of hazard would an add instruction have a in pipeline that executes two instructions at a time (dual-issue) that it does not have in our standard (single-issue) pipeline?
   a. Structural hazard
   b. Data hazard
   c. Control hazard
   d. None
   If two instructions can be executed at the same time you may need two adders to do them both. This is a structural hazard. The regular pipeline already had data hazards for the add instruction, and adds do not have control hazards.

5. Which of the following is true about hazards?
   a. All hazards can be solved with bubbles
   b. The use of bubbles breaks the ISA promise of "Atomic execution" of instruction
   c. Using the branch delay slot ensures the promise of sequential execution is kept
   d. There are still control hazards in single-cycle processor
   Delaying instructions long enough that the processor is effectively not pipelined will always solve a hazard, but will hurt performance.

6. In a certain program run on a 5-stage pipeline processor with double-pumped register file but no forwarding, 50% of all instructions are loads and 10% of the instructions directly following the loads depend on it. What is the slowdown of the program due to these dependencies?
   a. 1.0x (no slowdown)
   b. 1.05x
   c. 1.1x
   d. 1.2x
   You will need 2 bubbles to handle the data dependency, so the dependent instructions will have a throughput of ½ IPC, or 3 cycles per instruction. 
   $0.5 \times 0.10 \times 3 + (1 - 0.5 \times 0.10) \times 1 = 1.1$. 
   
   d. Nothing, there is no hazard
   The third instruction needs R1 in the EX stage, but the results from the first instruction is still in the WB stage, so we need to forward from WB to EX.
8. Branch Prediction and Exceptions and Interrupts

1. Why are branch predictors useful?
   a. They allow us to only execute the needed instructions when they are correct
   b. They allow us to squash instructions that should never have entered the pipeline
   c. They enable forwarding between pipeline stages to avoid branch dependencies
   d. All of the above

   Branch predictors try to predict the outcome of branches. Without a branch predictor, the CPU would either have to stall until the outcome of the branch is known, or start executing instructions and “squash” them if they should not have been executed.

2. What is the advantage of 2-bit branch predictors compared to 1-bit predictors?
   a. They change decision more slowly
   b. They learn faster
   c. They always make better predictions
   d. They use less logic

   When in the state “strongly taken” or “strongly not taken”, it takes two consecutive mispredictions to change the next prediction. This is useful to reduce the number of mispredictions in the inner loop of two nested loops.

3. A program contains 1 million instructions, of which 15% are branches. If we run it on a CPU with a 20 cycles branch penalty and a branch predictor that is 85% accurate, how many cycles will be wasted due to mispredictions?
   a. 100 000 cycles
   b. 200 000 cycles
   c. **450 000 cycles**
   d. 1 000 000 cycles

   This program contains 150 000 branches. 15% of them will be mispredicted, and each misprediction costs 20 cycles. 150 000 * 0.15 * 20 = 450 000 cycles wasted.

4. How would the performance change if we ran the program described in the previous question on a CPU with a 30 cycles branch penalty but with a branch predictor that is 90% accurate?
   a. Need more information
   b. The performance would improve
   c. The performance would decrease
   d. **The performance would remain the same**

   The program still contains 150 000 branches. 10% of them will be mispredicted, and each misprediction costs 30 cycles. 150 000 * 0.10 * 30 = 450 000 cycles wasted, so the performance remains the same compared to the previous predictor.

5. Consider the following code:
   ```c
   for (i = 0; i < 1000000; ++i)
   {
     for (j = 0; j < 1000; ++j)
     {
       m[i][j] = 0
     } // branch j-loop
   } // branch i-loop
   ``

   How many mispredictions would we see if we ran this program on a CPU with a 1-bit branch predictor (initialized to “taken”)?
   a. 0
   b. 1 000 001
   c. **2 000 000**
   d. 1 000 005
j-loop: wrong once in first j-loop, then twice per j-loop: 1 999 999 mispredictions
i-loop: wrong once at the very end: 1 misprediction.

6. What if we ran the same code on a CPU with a 2-bit branch predictor? (initialized to “strongly taken”)?
   a. 0
   b. 1 000 001
   c. 2 000 000
   d. 1 000 005

9. Input/Output
   1. Why have busses become a problem in modern systems?
      a. Hard to make small wires
      b. Hard to connect many wires
      c. Hard to find space for long wires
      d. Hard to keep the data on the wires synchronized
      The wires all have slightly different resistance and capacitance so they run at different speeds and are hard to keep synchronized.
   2. What is the benefit of DMA?
      a. Faster to detect that data is ready than polling
      b. Easier to program than interrupts
      c. Processor can do other things at the same time
      d. All of the above
      DMA really just allows the processor to do other things while the transfer is going on.
   3. How would you access a device if you wanted the lowest possible latency?
      a. Polling
      b. Interrupt
      c. I/O instructions
      d. DMA
      Polling is the lowest latency, but highest CPU usage, way to access. Because your processor is constantly checking you will see if data is ready soonest. I/O instructions are one way to do polling, but you could also use them in an interrupt. DMA is just a way to transfer data, not to find out if the data is ready.
   4. Why do we not use Flash memory instead of DRAM?
      a. Too expensive
      b. Too slow
      c. Doesn’t keep data without power
      d. Can’t store enough data
      Flash memory is cheaper than DRAM and keeps its data with no power, but it is still much slower than DRAM.
   5. How does Ethernet share a single serial connection?
      a. Each device is assigned different times to transmit
      b. Each device checks for data corruption and re-tries if needed
      c. Uses encryption to keep data safe from corruption
      d. Actually has separate wire for each device
      Ethernet looks to see if the checksum for each packet has been corrupted by another device trying to use the same wire and, if so, retries.
   6. How can memory-mapped IO accesses and virtual memory interact?
      a. Virtual memory allows more I/O devices per application than the ISA’s address space
b. Page protection can limit which programs can access I/O devices directly.
c. Copy-on-write is used to get new data each time you read from the I/O address.
d. All of the above.

Page protection can prevent programs from directly accessing memory-mapped I/O devices by preventing them from accessing the addresses that talk to those devices. Virtual memory doesn’t allow more I/O devices than the ISA address space since you still need addresses in the address space to access each one. You get new data each time you read from an I/O address because the device provides new data, not because of a copy-on-write.

10. Caches

1. A program is reading every word (4 bytes) consecutively of a very large (4GB) array. Which of the following cache configurations can achieve a hit ratio of more than 50%? Assume LRU replacement policy?
   a. Cache size: 16 B. Block size 4 B, Direct mapped
   b. Cache size: 8 MB. Block size 8 B, 2-way associative
   c. **Cache size: 2 KB. Block size 32 B. Fully associative**
   d. All of the above

None of these caches are large, so they will not be able to fit all the data. Since the data is being read sequentially, if we have a large enough cache block size, then we can reuse the same block repeatedly. A 32B block will be reused 8 times for sequential words, giving a hit ratio of 7/8=87.5%.

2. A program is reading a 4x4 word matrix in "the wrong way", e.g., addresses 0, 4*4, 8*4, .... What is the lowest associativity that will have no conflict misses for a 32B LRU cache with a block size of 8 B?
   a. **2-way**
   b. 4-way
   c. It is impossible to fully avoid conflict misses with this access pattern
   d. There are no conflict misses with this memory pattern

A full matrix column can now fill up the cache exactly.

3. What data is evicted on a miss in a LRU cache?
   a. The data that has been in the cache for the longest time
   b. The data that has most recently been installed in the cache
   c. **The data that has not been used for the longest time**
   d. Misses do not evict data

LRU=Least Recently Used.

4. Which of the following statements are NOT true about caches?
   a. Caches are faster than the DRAM used for main memory
   b. Large cache lines improves spatial locality
   c. **High associativity decreases the number of misses when you first access data**
   d. LRU is not always the best replacement policy

Increasing associativity does not reduce the number of misses when you first access data because the data will still not be in the cache (first time). A larger cache block might help by pulling in other data for you.

5. What data will be in a 4-entry, fully-associative, LRU cache with one word per line after the following memory accesses? 0, 1, 2, 3, 4, 5, 3, 2, 1?
   a. 1 3 4 5
   b. 1 2 3 5
   c. 1 2 4 5
   d. 0 3 4 5
In a fully associative LRU cache the most recently used data is always resident in the cache.

6. How many sets does a 4kB, 4-way set associative cache with an 8-byte line size have?
   a. 4
   b. 64
   c. 128
   d. 512
   \[4096 \text{ B} / 8 \text{ byte lines} / 4 \text{ ways per set} = 128 \text{ sets}.\]

11. Virtual Memory
   1. What does the TLB do?
      a. It caches recently-used data
      b. **It caches recently-used address translations**
      c. It caches recently-used pages
      d. It caches recently-used instructions
      **The TLB caches the recently-used virtual-to-physical address translations.**
   2. What would happen if two programs tried to read from the same virtual address?
      a. The two programs would likely crash
      b. One of them will read the correct data while the other would read random data
      c. They would read the same data
      d. **They would read their own data**
      Virtual memory gives the illusion to each program that it has sole access to the whole address space. If two addresses read from the same virtual address, they will read their own data, since they actually map to different physical addresses (assuming the addresses are valid in both programs).
   3. What are the benefits of virtual memory?
      a. It increases security by isolating each process’ memory
      b. It frees the applications from having to manage a shared memory
      c. It allows an application to use more memory than physically available
      d. **All of the above**
      Virtual memory gives the illusion to each program that it has sole access to the whole address space. The advantages are multiple: It increases security by preventing an application from accessing another program’s memory. It also frees applications from having to manage a shared physical memory (this burden is on the OS instead). Finally, each applications has access to the whole virtual address space, even if there is not enough physical memory available. In this case, the OS will use secondary storage transparently.
   4. What can cause a page fault?
      a. Trying to read a virtual address not present in the TLB
      b. Trying to read from a page that is not in main memory but that exists on the disk
      c. Trying to read an address that has no page in the main memory and no page on the disk
      d. **B and C**
      A fault occurs when the data is not in memory, either because it has been swapped to disk or it has not yet been read into memory or had a page allocated. If the page table entry is not in the TLB it is a TLB miss.
   5. What is true if you have more physical address bits than your virtual address bits?
      a. **The only way to use all the physical memory is to run multiple programs**
      b. You cannot address all the virtual memory from one program
      c. You can address more memory from one program than you have physical memory
      d. All of the above
The amount of physical memory that a process can address is limited by the number of virtual address bits. For example, on a system with 32-bits virtual addresses, each process can address at most $2^{32}$ bytes. It means that the rest of the physical memory cannot be addressed by a single process. However, by running multiple programs, we can make use of all the physical memory available.

6. For a system with 4MB ($2^{22}$ bytes) pages and 32GB ($2^{35}$ bytes) of physical memory, figure out which bits of a 64 bits virtual address are used for the Virtual Page Number (goes into the TLB for translation), the Page Offset and how many bits of Physical Addresses you have?

   a. 22 bits for PO, 42 bits for VPN. 35 bits physical addresses
   b. 42 bits for PO, 35 bits for VPN. 22 bits for physical addresses
   c. 35 bits for PO, 42 bits for VPN. 22 bits for physical addresses
   d. 22 bits for PO, 35 bits for VPN. 42 bits physical addresses

   22 bits are needed as offset within a page. The 42 bits left will be used as Virtual Page Number.
   If we have 32GB of physical memory, then we need at least 35-bits long physical addresses to address all of it.

12. Parallelism

   1. A program takes 200s to run on a single CPU, and 35% of it can be run in parallel. How long will it take to run the same program on a multicore CPU with 8 cores?

      a. 112s
      b. 120s
      c. 138s
      d. 145s

      Amdahl’s law gives us a maximum speedup of $1 / (0.65 + 1 / 8 * 0.35) = 1.44x$, which means the program will complete in 138s instead of 200s.

   2. Which of the following designs to improve performance does not require multiple PCs?

      a. Simultaneous multithreading (issuing instructions from multiple programs into the pipeline at the same time)
      b. Multicore processors (having multiple processors on one chip)
      c. Dual-issue processors (issuing multiple instructions from the same program into the pipeline at the same time)
      d. All of them require multiple PCs

      Dual-issue processors issue two instructions each cycle, so the PC simply counts up by 8. The others require multiple PCs to keep track of which instruction in each program is next.

   3. Why do we need locks?

      a. To synchronize accesses to the same data by different processors
      b. To make sure only one thread can use the pipeline at the same time
      c. To avoid page faults
      d. None of the above

      Locks ensure that data accessed by different processors or programs are properly synchronized.

   4. 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 20% parallel?

      a. Run on the CPU cores
      b. Run on the GPU cores
      c. Same
d. Need more information

On the GPU, Amdahl’s law gives us a speedup of 1.24x, thus the average speed of the system over the whole execution is 26GFlops.

On the CPU, Amdahl’s law gives us a speedup of 1.18x, thus the average speed of the system over the whole execution is 66GFlops.

5. How much faster can my program run if I have 9000 cores and 10% of my program cannot be parallelized?
   a. 9000x faster
   b. 4500x faster
   c. 5x faster
   **d. 10x faster**

   Using Amdahl’s law, we get a speedup of \( \frac{1}{0.10 + \frac{1}{9000} \times 0.90} = 10x \).

   Another way to answer this question is to say that by having infinite parallelism, 90% of this program will be done in 0 unit of time. Only 10% of it must be done serially, and thus the total execution time is divided by 10.

6. An image processing algorithm divides an image in 4 equally-sized parts and works on each part on a separate CPU in parallel. Each processor completes its work in 10 seconds, except one that completes in 12 seconds. If we run this program on a CPU with 8 cores, what is its execution time?
   a. 2 seconds
   b. 10 seconds
   c. **12 seconds**
   d. 5.25 seconds

   As long as we have at least as many cores as the program has threads, the execution time of this program will be the execution time of the slowest thread.

   Moreover, having more cores than threads is of no help here (4 cores would just idle).

---

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