Exam

ⓘ This is a preview of the published version of the quiz

Started: Dec 7 at 4:05pm

Quiz Instructions

Open book. No communicating with anyone else. **You must work on your own**!

Only submit the quiz **when you are done** with all the questions! Do submit **after** you are done.

Contacting us and Announcements

If you have questions about the exam, please contact CH (**chang.hyun.park@it.uu.se (mailto:chang.hyun.park@it.uu.se)**) and Anirban (**anirban.nag@it.uu.se (mailto:anirban.nag@it.uu.se)**). **Email both of us (mailto:chang.hyun.park@it.uu.se,anirban.nag@it.uu.se?subject=Question-DARK-Exams)** so we can handle questions more promptly.
CH will be unavailable between 8:30-9:15. During the other times, we'll try to answer ASAP.

If we have any announcements to make regarding the exam, we'll send you a message through Studium.

About the Exam

The total number of points for the exam is 120 points. The 7 additional points will be awarded as the bonus points students earned throughout the course. There are 20 questions.

Grading protocol: We are *planning* to set the thresholds as follows (points out of a total of 120 pts):
5: 100 pts and above
4: 85 pts and above
[illegible] pts and above
(Thresholds may be adjusted.)

Good luck!

Question 1 **10 pts**

Write the MIPS assembly code for finding the greatest common divisor using the following algorithm. (Two parts to this question. See below)

Given two arguments a and b, which are both positive integers, subtract the larger value by the smaller value until the two values are equal. When equal, that value is the greatest common divisor.
The following example shows how the algorithm works.

```
gcd (48, 18) :
  (48-18, 18): (30, 18)
  (30-18, 18): (12, 18)
  (12, 18-12): (12, 6)
  (12-6, 6): (6, 6)
gcd = 6
```

Your code should provide a function called gcd, that receives two arguments (both will be positive integers) and returns the gcd of the two arguments.

Note: Your MIPS assembly code must not use division nor modulo, and must only use subtraction.
You may use the Assembly instructions you used in the Lab 1, or you may refer to the table on page 3 of the **ISA-2 lecture slide (https://uppsala.instructure.com/courses/39372/files/1624512?wrap=1)** **(https://uppsala.instructure.com/courses/39372/files/1624512/download?download_frd=1)** .

1. [2 points] Write the pseudo-code for the gcd algorithm above. (Feel free to use a high-level language such as C or python to represent your pseudo-code)

2. [8 points] Use the MIPS assembly to write the gcd function as instructed.

Edit View Insert Format Tools Table

12pt Paragraph

p 0 words

Question 2

6 pts

Assume we want to add a new branch equal immediate instruction (beqi). This instruction would compare the value stored in a register with an immediate value. If the comparison results in equality, the CPU will branch.
Assume that we have a 32-register ISA (MIPS has 32-registers), and the 6-bits are reserved for the opcode, and the instructions are always 32-bits wide.

> **A.** If the immediate that is used for comparison with the value in the register is allocated 6 bits, how far (number of instructions) can the branch instruction jump? (Hint: Answer will be a range)
> **B.** If the branch can jump $\left[-2^{13},\ 2^{13}-1\right]$ instructions, what is the min/max value of the immediate used for comparison with the register value?

Edit View Insert Format Tools Table

12pt Paragraph

p 0 words

Question 3

4 pts

For the following program, list the contents of the stack when the program is running the 2nd instruction (line 30) of the `double` function. The program will start from the first line. Show the contents for the first encounter of line 30, if line 30 is encountered multiple times.
List the contents from the lowest address to the highest address.

```
 1 main:
 2     addi $t4, $zero, 8
 3     addi $sp, $sp, -4
 4     sw   $t4, 0($sp)
 5     addi $t5, $zero, 4
 6     add  $a0, $t4, $zero
 7     add  $a1, $t5, $zero
 8     jal  manipulate
 9     lw $t4, 0($sp)
10     addi $sp, $sp, 4
11     beq $t4, $v0, main
12     j Exit
13 manipulate:
14     add  $t2, $a0, $a1
15     sub  $t3, $a0, $a1
16     addi $sp, $sp, -8
17     sw   $t2, 0($sp)
18     sw   $ra, 4($sp)
19     add  $a0, $t2, $zero
20     add  $a1, $t3, $zero
21     jal  double
22     lw   $t2, 0($sp)
23     lw   $ra, 4($sp)
24     addi $sp, $sp, 8
25     add  $t4, $v0, $t2
26     sub  $v0, $v1, $t4
27     jr $ra
28 double:
29     add $v0,$a0,$a0
30     add $v1,$a1,$a1
31     jr $ra
```

Edit View Insert Format Tools Table

12pt Paragraph

p 0 words </>

Question 4

4 pts

For an 8-bit normalized floating-point format: $(-1)^{S}*$ (1.FFFF) * $2^{(EEE - 4)}$ where FFFF and EEE are unsigned, the number is written as follows from MSB to LSB: SEEEFFFF.

Design an algorithm that checks whether a floating-point number *x* is greater than, equal to, or less than another floating-point number *y*. Write the algorithm by comparing their floating-point representations bitwise from left to right, stopping as soon as the first differing bit is encountered.

Do you see why arranging the sign, exponent, or mantissa bits in a different way would have made such a comparison algorithm more complicated? (No answer needed)

Edit View Insert Format Tools Table

12pt Paragraph

p 0 words </>

Question 5

6 pts

You are designing an instruction set (ISA) for a processor which will only run machine learning applications. The processor will be used in edge devices (e.g., traffic cameras, headphones, etc) and should consume minimum chip area and power. The applications have the following characteristics: (i) All input/output values are in the range [-3, 3], and (ii) they desire a numeric precision of 0.03125.

1. [2 points] Which numeric format would you use for your ISA (floating-point or fixed-point)? Why?

2. [2 points] How many bits are needed (minimum) to represent the numbers?

3. [2 points] How are the bits allocated for various components of the numeric format? (If floating-point, then how many sign bits, how many mantissa bits, and how many exponent bits. If fixed-point, then how many bits before and after the radix point).

Edit View Insert Format Tools Table

12pt Paragraph

p 0 words

Question 6

3 pts

You are designing a logic circuit for an insulin pump that administers insulin to control blood glucose levels in diabetic patients. In every cycle, the circuit checks: (I) whether glucose level is above a T threshold (condition A), and (II) whether time since the last dose was administered is more than C cycles (condition B). If both conditions are met, a shot of insulin is administered (output O).

1. [1 point] Does the circuit include combinational logic, sequential logic, or both? (Remember, your circuit includes evaluating each of the conditions as well)

2. [2 points] Draw a truth table with the two conditions (A, B) and the output (O).

Question 7

8 pts

To realize the above logic circuit for administering insulin dosage, you need a bunch of logic components.

Here is a list of components that you might need:

1. AND gate (Inputs: AND_In1 and AND_In2; Output: AND_Out)

2. Comparator1 (Inputs: Comp1_In1 and Comp1_In2; Output: Comp1_Out)

3. Comparator2 (Inputs: Comp2_In1 and Comp2_In2; Output: Comp2_Out)

4. Counter (Inputs: Clock, Count_In, and Reset; Output: Count_Out). The Reset signal resets the counter to zero. The counter adds 1 to Count_In and outputs the result to Count_Out every cycle.

Inputs to your circuit are: Current blood glucose level (G), glucose threshold (T), and cycles to wait after a dose (C).

Connect the components to get your desired logic circuit. Write your answer in the following format: "X connects to Y". Write one connection per line.

Question 8

4 pts

1. [2 points] Assume that a classmate of yours asked the following question. State why your classmate is wrong, and answer their question.

> Why would you even need the instruction memory? Why not decode the value from the PC register directly to get the opcode, the register numbers, immediate values and more?

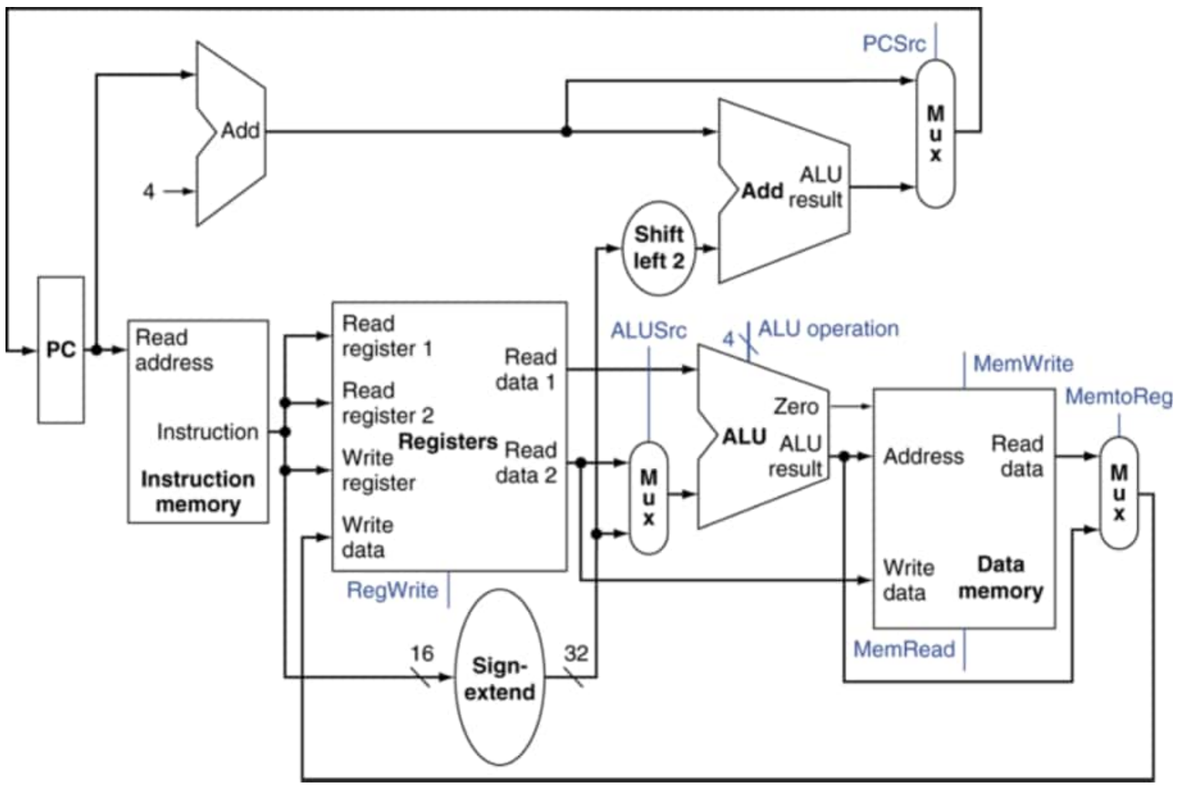
2. [2 points] This time your classmate asks you a question about the branch adder. Correct your classmate.

> Isn't the branch target adder redundant? Why can't we just use the ALU to calculate our branch target and send the result to the PC?

Question 9

6 pts

Given the basic data-control path diagram below, explain how you would implement the `jalr` instruction. (Jump to the instruction whose address is in the register `rs`. Save the address of the next instruction in the register `rd`)



A. What are the new data path wires that are needed? Where would the datapath wires connect to?
B. List all the control signals that will be required, and state what each control signal controls.

Question 10

9 pts

An unpipelined processor takes 10 ns to work on one instruction. I was able to convert the circuits into 10 sequential pipeline stages with a pipeline register delay of 0.2ns. The stages have the following lengths: 1.3ns; 0.7ns; 0.8ns; 1.2ns; 0.7ns; 1.4ns; 0.7ns; 0.4ns; 1.5ns; 1.3ns. Answer the following, assuming that there are no stalls in the pipeline.

1. [1.5 points] What are the cycle times in each of the processors?
2. [1 point] What are the IPCs (Instructions Per Cycle) in each of the processors (averaged across millions of instructions and assuming no pipeline hazards)?
3. [1.5 points] How long does it take to finish one instruction in each of the processors (in nano-seconds)?
4. [2 points] What is the throughput speedup provided by the 10-stage pipeline?
5. [3 points] If I was able to build a magical 1000-stage pipeline, where each stage took an equal amount of time, what throughput speedup would I get? (Assume each stage also has an additional 0.2ns pipeline register delay)

Question 11

3 pts

Consider a program with branch instructions, where a branch is taken 70% of the time. Assume a single branch delay slot. How many useless instructions are executed on average per branch instruction in each of the following cases? (Average useless instructions can be fractional)
1. [1 point] The compiler can't find anything to put in the branch delay slot and hence puts NOPs.

2. [1 point] The compiler puts an instruction before the branch into the branch delay slot.

3. [1 point] The compiler puts an instruction from the branch "taken" path into the branch delay slot.

Question 12

9 pts

Consider the following in-order pipeline with the three different multi-cycle instructions:

Instruction Type	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6	Stage 7	Stage 8	Stage 9

Integer Add	IF	DEC	RR	RR	IntAdd	WB	WB		
Load/Store	IF	DEC	RR	RR	EffAdd	DM	DM	WB	WB
Floating Point Add	IF	DEC	RR	RR	FPAdd	FPAdd	FPAdd	WB	WB

The initial part of the pipeline involves instruction fetch (IF), decode(DEC), and two register-read (RR) stages. The decode stages will force an instruction to remain in the DEC stage until it is safe to proceed. After the 2nd RR stage, Integer-adds go through IntAdd, and then two register writeback (WB) stages. Loads and stores go through EffAdd (where the load/store address is calculated), then two data-memory (DM) stages, and finally two WB stages. Floating-point adds go through three FPAdd stages and then two WBstages. What are the stall cycles introduced between the following pairs of successive instructions with and without data forwarding? (You need not show pipeline diagrams).

1. [1.5 + 1.5 points] A floating-point Add, providing data for a store.
2. [1.5 + 1.5 points] A load, providing address for a store.
3. [1.5 + 1.5 points] Int-add, providing address for a load.

Hints:

Register-reads complete at the end of the second RR stage. Writebacks complete at the end of the second WB stage.

Floating-point add instruction produces a result at the end of the third FPAdd stage.

Store instruction consumes data at the beginning of the first DM stage.

Load instruction produces data at the end of the second DM stage.

Load/Store instruction consumes address at the beginning of the EffAdd stage to calculate the effective address.

Integer Add instruction produces a result at the end of the IntAdd stage.

Question 13

7 pts

1. [5 points] Assuming the 5-stage pipelined processor we learned in the course, which stage would you place the branch predictor? Justify your answer. Make sure to include how many bubbles or nops cycles are required on a branch prediction hit. Also include how your branch predictor would connect to the datapath (Where would the output of your branch predictor send the data to?)

2. [2 points] If your branch predictor miss-predicts, how many instructions after the miss-predicted branch will need to be squashed (prevent instruction from finishing). Assume that there are no branch delay slots. Explain your answer.

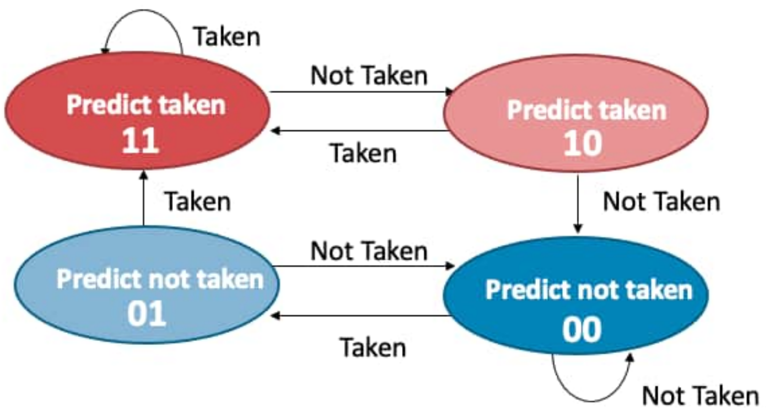
Question 14

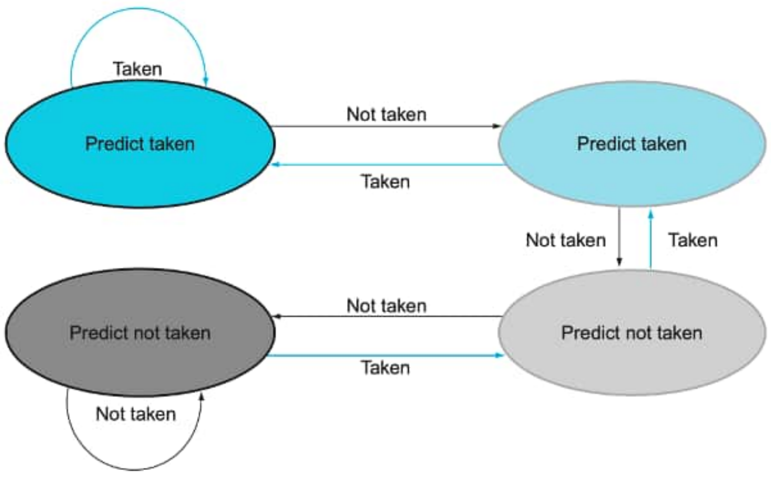
3 pts

For the code below, calculate the accuracy of the two branches (for and if statements) for the following branch predictors:

```
for (i = 0; i < 1,000,000; i++) {
  if (i % 5 < 3) {
    do_a(); // Branch Not Taken
  } else {
    do_b(); // Branch Taken
  }
}
```

- 1-bit branch predictor
- 2-bit branch predictor from the lecture slide (image on left)
- 2-bit branch predictor from the textbook (image on right)





the 1-bit predictor is initialized to not-taken, the 2-bit predictor on the left as strongly not taken (00, dark blue), and the 2-bit predictor on the right as strongly not taken (dark grey, bottom left).

Question 15

10 pts

1. [2 points] Explain how the latency and the bandwidth of your computer network (LAN, Wifi) affect your web-browsing experience. (Explain for both latency and bandwidth).

2. [2 points] With I/O, what is the benefit of polling over interrupt? What is the benefit of interrupt over polling?

3. [2 points] As network and storage devices improve, the latency of the devices are becoming increasingly important. Which method of I/O completion (polling or interrupt) will be more advantageous for lower latency I/O? Explain why the completion method will improve latency.

4. [4 points] Once the network device receives a packet from the network, the packet must be copied onto the main memory before the packet can be processed by the CPU. Explain how you could use DMA and interrupts to copy the packet into main memory. How can using DMA improve the system performance?

Question 16

7 pts

Consider a processor using 2 cache levels. Level-1 cache is a 32KB direct-mapped cache with 16B blocks used for both instructions and data. Level-2 is a 1MB, 4-way set-associative cache with 64B cache lines. Assume that the processor can address up to 4GB of main memory.

1. [3 points] Find the size of tag, index, and offset bits for level-1 and level-2 caches.

2. [4 points] Compute the total number of data bits and tag bits (overhead) for each level.

Question 17

5 pts

Consider a 2-way set-associative cache with two sets. When running a program on this machine, the following access pattern is observed.

A, D, B, D, C, E, E, A, F, B, F, C, D, A, E, F, B, C

Assume that blocks A, B, and C are mapped to set 1, and blocks D, E, and F are mapped to set 2.

1. [3 points] Compute the hit rates for the program when the LRU replacement policy is used.

2. [2 points] Which four cache blocks are present in the cache at the end?

Question 18

6 pts

1. [4 points] Assume that our computer is a 32-bit system (much like the one we learned during the lecture. Assume that the size of a page is 4KB.

- Program A: Allocates 4GB of memory from the beginning of the address space (starting at address 0)

- Program B: Allocates 1 page (4KB) at the very beginning and at the very end of the address space.

Compare the size of a singe level page table and a 2-level page table for both program A and B. Explain how you calculated the size.

2. [2 points] Explain the benefit of a VIPT cache over a PIPT cache. How does VIPT cache provide that benefit?

Question 19

4 pts

Assume a system has 16KB pages and the designers are planning to implement a 8-way cache.

- What is the maximum size of an L1 cache if it is going to be organized as a VIPT cache?
- Assuming that the cache block size is 64B, how many sets will the cache have?

Question 20

6 pts

As an architect, you have two options to upgrade your company's first-generation uni-core single-issue processor.

Option 1: Add 3 additional cores (total 4 cores). On average, 50% of an application code can be parallelized across 4 cores (providing speedup of 4x for the parallel part).

Option 2: Make it a dual-issue pipeline that can issue load/store instructions alongside other instructions. On average, 33% of an application code is load/store instructions. Assume all load/store instructions can be issued in parallel alongside other instructions.

Which option would you choose to improve the performance of your second-generation processor (compare speed-ups)? Assume IPC (Instructions Per Cycle) is 1 in the first generation processor.

Question 21

7 pts

End of exam ;) This question is only used by the grader to assign you with bonus points you earned throughout the course.

Not saved Submit Quiz