Instructions:

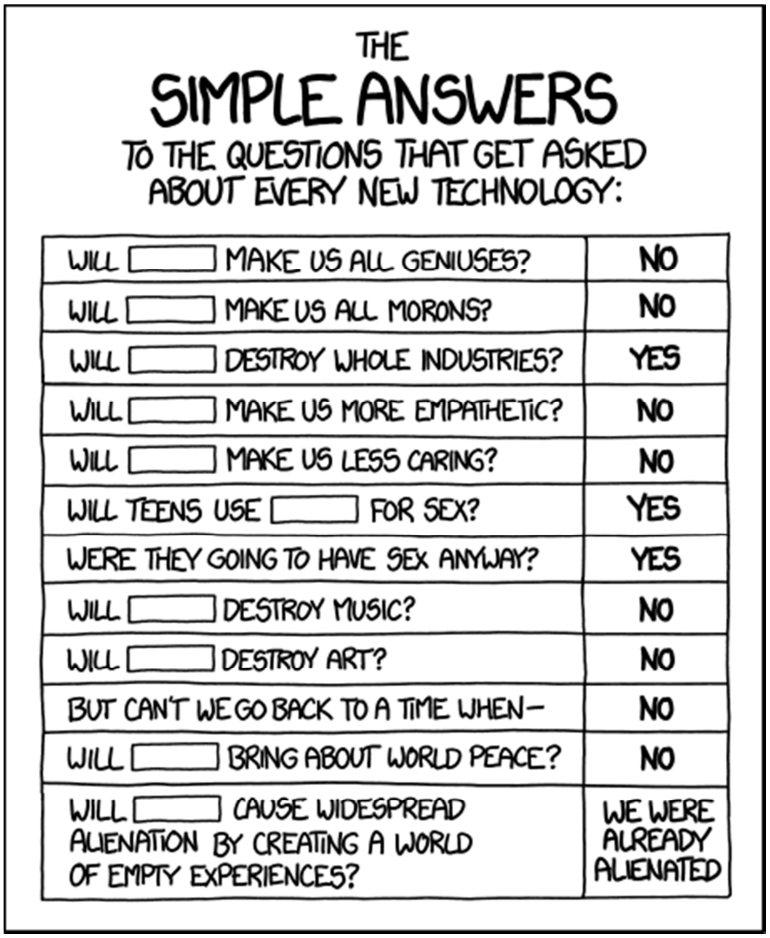
- Print your name in the space provided below.
- This examination is closed book and closed notes, aside from the permitted one-page formula sheet. No calculators or other computing devices may be used. The use of any such device will be interpreted as an indication that you are finished with the test and your test form will be collected immediately.
- Answer each question in the space provided. If you need to continue an answer onto the back of a page, clearly indicate that and label the continuation with the question number.
- If you want partial credit, justify your answers, even when justification is not explicitly required.
- There are 10 questions, some with multiple parts, priced as marked. The maximum score is 100.
- When you have completed the test, sign the pledge at the bottom of this page and turn in the test.
- If you brought a fact sheet to the test, write your name on it and turn it in with the test.
- Note that either failing to return this test, or discussing its content with a student who has not taken it is a violation of the Honor Code.

Do not start the test until instructed to do so!

Name Solution
printed

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

_____
signed


THE
SIMPLE ANSWERS
TO THE QUESTIONS THAT GET ASKED ABOUT EVERY NEW TECHNOLOGY:
WILL [ ] MAKE US ALL GENIUSES? NO
WILL [ ] MAKE US ALL MORONS? NO
WILL [ ] DESTROY WHOLE INDUSTRIES? YES
WILL [ ] MAKE US MORE EMPATHETIC? NO
WILL [ ] MAKE US LESS CARING? NO
WILL TEENS USE [ ] FOR SEX? YES
WERE THEY GOING TO HAVE SEX ANYWAY? YES
WILL [ ] DESTROY MUSIC? NO
WILL [ ] DESTROY ART? NO
BUT CAN'T WE GO BACK TO A TIME WHEN— NO
WILL [ ] BRING ABOUT WORLD PEACE? NO
WILL [ ] CAUSE WIDESPREAD ALIENATION BY CREATING A WORLD OF EMPTY EXPERIENCES? WE WERE ALREADY ALIENATED

xkcd.com

1. [10 points] Consider the following statement about the MIPS32 datapath design:

> The adoption of a pipelined datapath design will increase the total time (latency) to execute certain instructions, and but will still decrease the total time required to execute a typical sequence of instructions.

If the statement is true, explain why. If the statement is false, explain why adopting a pipelined datapath does improve overall performance.

As we saw in class, the introduction of the MIPS32 pipeline would actually increase the total latency for every supported instruction, so the statement above is true in that regard.

With the pipelined design, we would (ideally) complete one instruction per much shorter clock cycle.

The effect is that we improve the throughput.

---

2. [12 points] This question relates to the execution of MIPS32 instructions on a pipelined datapath, with both forwarding and hazard detection. Complete the following sequence of assembly instructions so that completing the execution of the instruction that's currently in the ID stage would require forwarding from both the EX/MEM interstage buffer and the MEM/WB interstage buffer (but no stall in either case):

```
add $t2, $s0, $s1

add $t1, $s2, $s3

sw $t0, 12($t2)        # currently in the ID stage
```

In your answer above, what must be forwarded from the EX/MEM interstage buffer?

The value that the second add instruction will write to $t1.

In your answer above, what must be forwarded from the MEM/WB interstage buffer?

The value that the first add instruction will write to $t2.

3. [8 points] The Hazard Detection unit was added to the pipeline design in order to deal with what specific scenario? Give an example and explain it.

The purpose of the Hazard Detection unit is to stall the reading instruction when the writing instruction is lw and is once cycle ahead of the reading instruction:

```
lw   $t0, ($t1)
add  $t3, $t0, $t2
```

4. [8 points] The Forwarding Unit takes (as input) the write-to register numbers from the instruction that's in the EX stage and from the instruction that's in the MEM stage. But the Hazard Detection unit does not take the write-to register number for the instruction that's in the MEM stage. Explain.

The load-use hazard described above occurs only in situations where the writing lw instruction is one cycle ahead of the reading instruction.

Therefore, the register the instruction that's two cycles ahead is irrelevant (for this issue).

---

5. [10 points] Why can't the given MIPS32 pipeline design (with forwarding and hazard detection) correctly execute the following sequence of instructions? Or can it?

```
        lw   $s0, 0($t1)        # 1
        lw   $s1, 4($t1)        # 2
        sub  $s7, $s0, $s1      # 3
        beq  $s7, $zero, skip   # 4
        sw   $s7, 0($s2)        # 5
skip:                           # 6
        add  $s0, $s1, $zero    # 7
        lw   $s1, 8($t1)        # 8
```

The problem is the occurrence of the beq instruction (#6).

With the given pipeline design, we do not decide whether to take the branch until the beq reaches the MEM stage, where we set the control signal for the MUX and choose the next value for the PC.

The given pipeline design has no way to deal with this delay. It will fetch some instruction when beq moves into EX, and another when beq moves into MEM, but those may not be the correct instructions to execute next.

6. [10 points] Was it logically necessary to add interstage buffers to the pipeline design? If yes, explain why. "They store stuff" is not an acceptable answer. If it was not logically necessary, what was the rationale for adding them?

In order for the pipeline to execute instructions correctly, the control signals relevant to each instruction must move forward, stage by stage, along with the instruction.

The interstage buffers operate like registers in that they only update state on a clock tick.

Therefore, they allow us to synchronize each instruction with its control signals, register numbers and computed values as it moves down the pipeline.

So, they ARE logically necessary.

---

7. [10 points] Refer to the diagram of the MIPS32 pipeline with forwarding and hazard detection.

Consider the value of the Write register number, which is set in the EX stage of the pipeline. That value is passed forward via the EX/MEM and MEM/WB interstage buffers. But the value is then sent back to the ID stage, bypassing the interstage buffers.

Why is the signal NOT sent back to the ID stage via the interstage buffers? Be precise and direct.

Any instruction that writes to a register (R-type and lw) does not do so until that instruction reaches the WB stage of the pipeline.

Since the write must occur during the (first half of the) clock cycle the instruction is in WB, we must send the RegWrite signal (and write register number and write data) to the register file in the ID stage.

Interstage buffers update their state on clock "ticks".

If any of those were sent "backwards" via the interstage buffers, the write could not occur during the correct clock cycle, even assuming forwarding logic and hazard logic and interstage buffer sizes were updated to support this.

8. [12 points] A system has 2^{36} bytes of DRAM. The system has a single level of cache memory, organized in 2^8 sets each holding 2^7 blocks, with a block size of 2^6 bytes.

Recall that we number bits of an address from low to high, starting at 0.

Which bits of a DRAM address **A** would be used to determine the set number to which that address would be mapped?

$A_{13}A_{12}A_{11}A_{10}A_9A_8A_7A_6$

Which bits of a DRAM address **A** would be used for the tag field?

$A_{35}...A_{14}$

How many bytes of user (DRAM) data can the cache hold? Express your answer as a power of 2.

$2^8 \times 2^7 \times 2^6 = 2^{21}$

For a given physical address, in how many places could the corresponding block of DRAM be stored in the cache?

The address determines the unique set number to which the data must be stored; however, it could be placed in ANY of the lines within that set:

2^7

---

9. [8 points] Give an example of C code that exhibits spatial locality, and explain why.

There are many examples; convincing ones will involve traversing an array in a stride-1 pattern, and comment on that in the explanation.

10. This question is about the effect of associativity on cache implementation.

a) [2 points] Which cache design approach implies that for each DRAM address there is no restriction on where the corresponding block of data could be stored in the cache?

This is true for a fully-associative cache (one in which there is a single set).

b) [6 points] Describe the logical conditions that determine whether a particular cache line (block) contains the data matching a particular DRAM address.

A match occurs precisely if and only if the cache line's Valid bit is 1 and the tag stored in the cache line equals the tag portion of the DRAM address.

c) [4 points] Some cache designs imply that there are several (or many) locations where the block of data corresponding to a DRAM address could be stored (in the cache). How can the cache be implemented so that the cost of searching each of those possible locations for a match is minimized? Be complete.

A cache match occurs when the valid bit in a cache line is set AND the address tag matches the tag in the cache line.

We can check each of the lines in a given set in parallel by simply supplying for each line equality hardware for the tags and an AND gate to check that both conditions hold.

The key is simply that we can replicate the necessary hardware (at some cost in space) and perform all the checks simultaneously.