CS 61C Spring 2026 — Yan Final

Solutions last updated: Tuesday, May 26, 2026

Print Your Name: _____

Print Your Student ID: _____

Print the Name and Student ID of the person to your left: _____

Print the Name and Student ID of the person to your right: _____

Print the Name and Student ID of the person in front of you: _____

Print the Name and Student ID of the person behind you: _____

You have 170 minutes. There are 9 questions of varying credit. (100 points total)

Question:	1	2	3	4	5	6	7	8	9	Total
Points:	14	10	12	15	10	11	12	16	0	100

For questions with **circular bubbles**, you may select only one choice.

- ○ Unselected option (Completely unfilled)
- ◎ Don't do this (it will be graded as incorrect)
- ● Only one selected option (completely filled)

For questions with **square checkboxes**, you may select one or more choices.

- ■ You can select
- ■ multiple squares
- ☑ (Don't do this)

Anything you write outside the answer boxes or you ~~cross out~~ will not be graded. If you write multiple answers, your answer is ambiguous, or the bubble/checkbox is not entirely filled in, we will grade the worst interpretation. For coding questions with blanks, you may write at most one statement per blank and you may not use more blanks than provided.

If an answer requires hex input, you must only use capitalized letters (`0xB0BACAFE` instead of `0xb0bacafe`). For hex and binary, please include prefixes in your answers unless otherwise specified, and do not truncate any leading 0's. For all other bases, do not add any prefixes or suffixes.

As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others. I will follow the rules of this exam.

Acknowledge that you have read and agree to the honor code above and sign your name below:

_____

Q1 *61Chaos: Potpourri*  **(14 points)**

Suppose we run the code on the right on a 64-bit little-endian machine. Assume `sizeof(short) = 2`. What is the output of each print statement?

```
1 int8_t y = 0x8A;
2 printf("%d", y);
3 int64_t val = 0x0102030405060708;
4 printf("0x%x", *((short *) &val) + 2));
```

Q1.1 (1 point) Print statement on Line 2:

`-118`

Q1.2 (1 point) Print statement on Line 4:

`0x304`

Consider a 16-bit floating point system SEEEEEMMMMMMMMMM (1 sign bit, 5 exponent bits, 10 mantissa bits) with a bias of −16. All other properties of IEEE 754 apply (bias, denormalized numbers, ∞, NaNs, etc.).

Q1.3 (2 points) Express the decimal −13.75 in this floating point system. Write your answer in hexadecimal.

`0xCEE0`

Q1.4 (2 points) What is the difference between the smallest positive normalized number and the largest positive denormalized number? Write your answer as a power of 2. If your answer is 2^x, you should write 2^x, not x.

2^{-25}

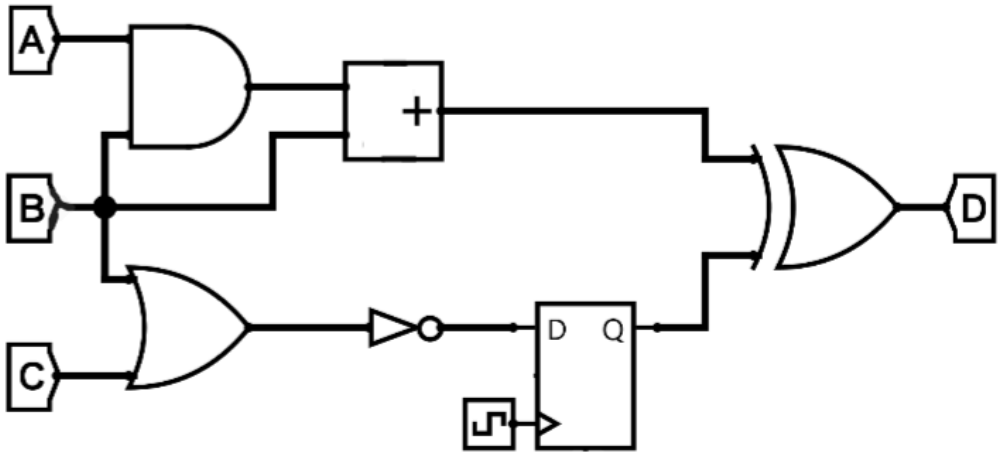
Q1.5 (2 points) Translate `0x00B92223` to a RISC-V instruction. Express immediates in decimal and use the corresponding register names instead of numbers (i.e. `a5` instead of `x15`).

`sw a1 4(s2)`

Q1.6 (2 points) Suppose the sequential component of a program accounts for 20% of the total execution time. If we want the overall program to achieve a 4x speedup, what must be the speedup of the parallelizable portion? For full credit, express your answer in its most simplified form.

16× speedup

Consider the following circuit diagram and timings. **Assume each input and output tunnel is connected to a register.**



$t_{\texttt{AND}} = 30$ ns
$t_{\texttt{OR}} = 25$ ns
$t_{\texttt{XOR}} = 25$ ns
$t_{\texttt{NOT}} = 8$ ns
$t_{\texttt{ADDER}} = 15$ ns
$t_{\texttt{CLK-TO-Q}} = 7$ ns
$t_{\texttt{SETUP}} = 5$ ns

Q1.7 (2 points) What is the minimum allowable clock period?

82 ns

Q1.8 (2 points) What is the maximum allowable hold time?

32 ns

Q2 *61Characters* (10 points)

We prompted a Generative AI agent to write the (functional) code below. The reference for `strchr` from the C `<string.h>` library follows.

The `unravel` function unravels a pattern string into individual "strand" strings (Figure 1). A `pattern` has the special character `'/'` to indicate a "cut" in a strand, meaning the remainder of the `pattern` should start from a new strand.

Read the code first, then answer the questions on the following page.

```
"/abcd/efg/lm"
      ↓
      ""
   "abcd"
    "efg"
    "lm"
```

Figure 1: (top) `pattern`; (bottom) `ret.strands`.

```
1  typedef struct {
2      char **strands;
3      int count;
4  } strands_t;
5
6  strands_t unravel(char *pattern) {
7      strands_t ret = {NULL, 0};
8      if (!pattern) return ret;
9
10     char *next_slash;
11     while ((next_slash = strchr(pattern, '/')) != NULL) {
12         ret.strands = realloc(ret.strands, sizeof(char *) * (ret.count + 1));
13
14         int len = (int)(next_slash - pattern);
15         ret.strands[ret.count] = malloc(len + 1);
16         strncpy(ret.strands[ret.count], pattern, len);
17         ret.strands[ret.count][len] = '\0';
18
19         ret.count++;
20         pattern = next_slash + 1;
21     }
22
23     ret.strands = realloc(ret.strands, sizeof(char *) * (ret.count + 1));
24     int final_len = (int)strlen(pattern);
25     ret.strands[ret.count] = malloc(final_len + 1);
26     strcpy(ret.strands[ret.count], pattern);
27     ret.count++;
28
29     return ret;
30 }
```

```
char *strchr(const char *s, int c);
```
Find the first occurrence of character `c` (the `int` is converted to a `unsigned char`) in the string `s`. Return a pointer to the matched character or `NULL` if the character is not found. The terminating null byte is considered part of the string. If `c` is specified as `'\0'`, return a pointer to the terminator.

Q2.1 (1 point) Right before the function returns, where is `ret.strands` located? ○ text ○ data ○ heap ● stack

Q2.2 (1 point) Right before the function returns, where is `*ret.strands` located? ○ text ○ data ● heap ○ stack

Q2.3 – Q2.5: From K&R: "In C, any assignment...is an expression and has a value, which is the value of the left hand side after the assignment." Move the assignment statement out of the loop conditional.

(2 points) First, modify Lines 10 and 11 and rewrite them below.

```
... // (Line 9 omitted)

char *next_slash = strchr(pattern, '/');  // Line 10
                   Q2.3

while (next_slash) { // Line 11
       Q2.4
```

Q2.5 (2 points) Second, fill in the blanks: Insert after Line Number 20 the statement:

```
next_slash = strchr(pattern, '/');
```

Q2.6 (2 points) In Lines 12 and 23, explain why `realloc` is needed to implement `unravel`'s functionality. Additionally, justify the choice of arguments to `realloc`. Answer in at most 20 words.

Lines 15/25 use ret.count as an index, so we must allocate one more element. Elements are strings/char pointers.

Q2.7 (2 points) The description of `strncpy` from the `man` pages:

> `char *strncpy(char *dest, const char *src, size_t n);`
> Copies the string pointed to by `src` to the buffer pointed to by `dest`. At most `n` bytes of `src` are copied. **Warning**: If there is no null byte among the first `n` bytes of `src`, the string placed in `dest` will not be null-terminated. If the length of `src` is less than `n`, `strncpy()` writes additional null bytes to `dest` to ensure that a total of `n` bytes are written.

Which of the below edits maintain the functionality of Lines 16 and 17?

- ■ Rewrite Line 17 to `*(*(ret.strands + ret.count) + len) = '\0';`
- ☐ Rewrite Line 17 to `(*(*ret.strands + ret.count) + len) = '\0';`
- ☐ Rewrite Line 17 to `*(*ret.strands + ret.count + len) = '\0';`
- ☐ Replace Lines 16 and 17 with `strncpy(ret.strands[ret.count], pattern, len+1);`
- ○ None of the above

Q3 *61Compare* ■ **(12 points)**

(12 points) Complete the implementation of the `square_indicators` RISC-V function.

The `square_indicators` function takes in an array of integers, and modifies their value **in place**. The value should be a 1 if the integer at that position is a perfect square, and a 0 if it is not.

Before	`[1, 6, 7, 56, 25, 81, 13, 9]`
After	`[1, 0, 0, 0, 1, 1, 0, 1]`

`square_indicators`		
Arguments	`a0`	A pointer to the start of an array of integers.
	`a1`	Number of elements in the array
Return value	`None`	

You have access to the `is_square` function below. You may not assume anything about its implementation, other than that it follows calling convention and behaves as follows:

`is_square`		
Arguments	`a0`	An integer
Return value	`a0`	1 if the integer is a square, otherwise 0

You may only use the registers:
`x0`, `a0`, `a1`, `s0`, `sp`, and `ra`.

```
1  square_indicators:
2      addi sp sp -8
           Q3.1
3      sw s0 0(sp)
4      sw ra 4(sp)
           Q3.2
5      mv s0 a0
6
7  loop:
8      beq a1 x0 done
             Q3.3
9      lw a0 0(s0)
           Q3.4
10     addi sp sp -4
11     sw a1 0(sp)
           Q3.5
12     jal ra is_square
         Q3.6
13     lw a1 0(sp)
           Q3.7
14     addi sp sp 4
15     sw a0 0(s0)
           Q3.8
16     addi s0 s0 4
           Q3.9
17     addi a1 a1 -1
18     j loop
19
20 done:
21     lw s0 0(sp)
           Q3.10
22     lw ra 4(sp)
           Q3.11
23     addi sp sp 8
           Q3.12
24     jr ra
```

Q4 *61Cmov* 💡 **(15 points)**

We would like to incorporate a instruction, `cmov rd rs1 rs2` ("conditional move") with the functionality described to the right.

```
if (R[rs1] != 0) {
  R[rd] = R[rs2]
}
```

Q4.1 (1 point) The code on the right uses `cmov`. (*Note*: For branch and jump instructions, the signed offset can be directly specified as a number, e.g., executing `j -8` adds −8 to the PC.)

After running the code below, what is register `s0`? Express your answer in decimal.

```
1  li   s0 715
2  li   s1 496
3  beq  s0 s1 12
4  cmov s0 s0 s1
5  j    -8
6  jr   ra
```

496

The `mystery` function on the right takes in 3 inputs, `a0`, `a1`, and `a2`, and returns a single value in `a0`.

Q4.2 (1 point) If the function is called with `R[a0] = 13, R[a1] = 76, R[a2] = 32`, what is the final value in register `a0`?

```
1  mystery:
2    slt  t1 a0 a1
3    cmov a0 t1 a1
4    slt  t1 a0 a2
5    cmov a0 t1 a2
6    jal  ra
```

76

Q4.3 (1 point) In 10 words or less, describe the behavior of `mystery`.

Return the maximum of the three arguments

(2 points) We can implement `cmov` by translating it into two consecutive existing RISC-V instructions. Write the instructions and arguments required to match the functionality of `cmov`. You may only use the following registers in your answer: `x0` and `cmov` parameters (`rd`, `rs1`, `rs2`).

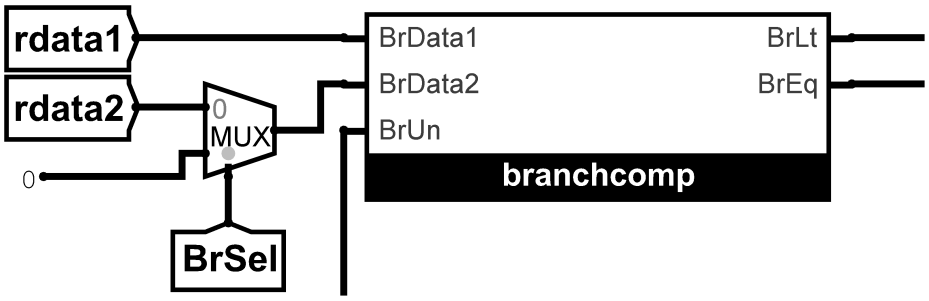
```
1  beq rs1 x0 8 # Instruction 1
       Q4.4
2  add rd rs2 x0 # Instruction 2
       Q4.5
```

Q4.6 (1 point) Which of the below stages would be responsible for this translation of `cmov`? Assume static linking.

○ Compiler ● Assembler ○ Linker ○ Loader

We can also implement `cmov` as a new instruction by incorporating it into our datapath. Refer to the **single-cycle datapath** on the reference card, which is the standard RISC-V CPU used in Project 3. To do this, `cmov` requires two changes described in Q4.7 (on this page) and Q4.8 (on the next page).

Q4.7 (5 points) A new MUX is added to the second input of the branch comparator with a new control signal, `BrSel`. When `BrSel = 0`, the MUX selects `rdata2`; when `BrSel = 1`, the MUX selects a constant 32-bit `0` value, as pictured below.

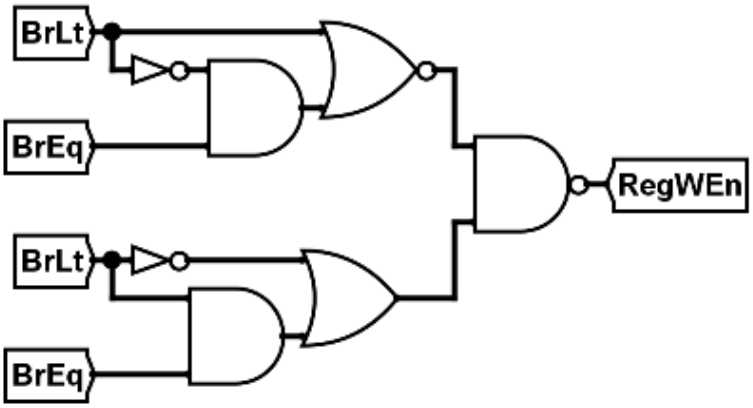


For the instruction `cmov`, what are the control signals used? If a control signal doesn't matter, select "Don't Care".

`PCSel`	● 0 ○ 1 ○ Don't Care
`ImmSel`	○ I-Type ○ B-Type ○ J-Type ○ S-Type ○ U-Type ● Don't Care
`BrSel` (new)	○ 0 ● 1 ○ Don't Care
`BrUn`	○ 0 (signed) ○ 1 (unsigned) ● Don't Care
`ASel`	○ 0 ○ 1 ● Don't Care
`BSel`	● 0 ○ 1 ○ Don't Care
`ALUSel`	○ `add` ○ `xor` ○ `and` ○ `mulhu` ● `bsel` ○ `sll` ○ `srl` ○ `mul` ○ `sub` ○ Don't Care ○ `slt` ○ `or` ○ `mulh` ○ `sra`
`MemRW`	● Read ○ Write ○ Don't Care
`WBSel`	○ 0 ● 1 ○ 2 ○ Don't Care
`RegWEn`	(see next page)

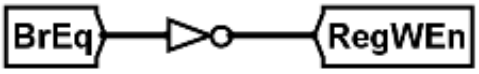
Q4.8 (4 points) The `cmov` instruction requires modifications to the implementation of the `RegWEn` control signal. Which of the following circuit(s) are valid modified circuit implementations using `BrEq` and/or `BrLt` as inputs? The `RegWEn` circuits here apply to the `cmov` instruction only.

☐ A

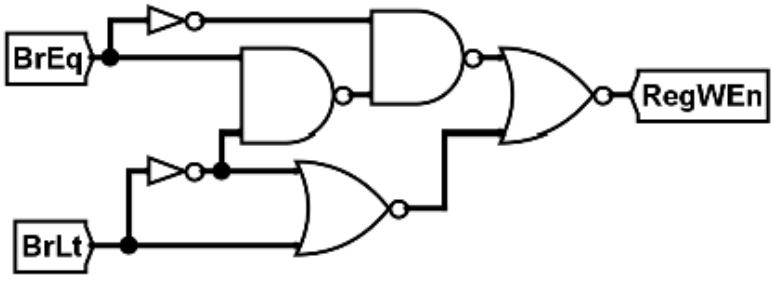


☐ B

■ C



■ D



○ None

Q5 *61Cya* (10 points)

Q5.1 (1 point) Structural hazards due to memory access can be completely eliminated by having separate instruction and data memories.

- ● True
- ○ False

Q5.2 (1 point) Suppose that in a new 5-stage pipelined datapath, the branch comparator is moved to the ID stage, and the PCSel control signal is updated at the beginning of the following clock cycle. The control hazard penalty (number of stalls) for a mispredicted branch in this new datapath is _____ that of the 5-stage datapath on the reference card.

- ○ greater than
- ○ equal to
- ● less than

Solution: Q5.1: **True.** By providing separate hardware for each functional unit (like separate IMEM and DMEM), we ensure that multiple instructions can access necessary resources simultaneously without conflict.

Q5.2: **less than.** The 5-stage pipelined datapath in CS 61C resolves branches in the MEM stage, not the EX stage. Resolving the branch earlier (in EX instead of MEM) means fewer instructions need to be flushed when a branch is mispredicted, decreasing the control hazard penalty.

The following diagram Figure 2, illustrates all **potential** forwarding paths. *Unless explicitly stated, assume no forwarding is available. For the remainder of this question, you will identify pipeline hazards. It may be helpful use the waterfall diagrams in the back of the reference card, but will NOT be graded.*

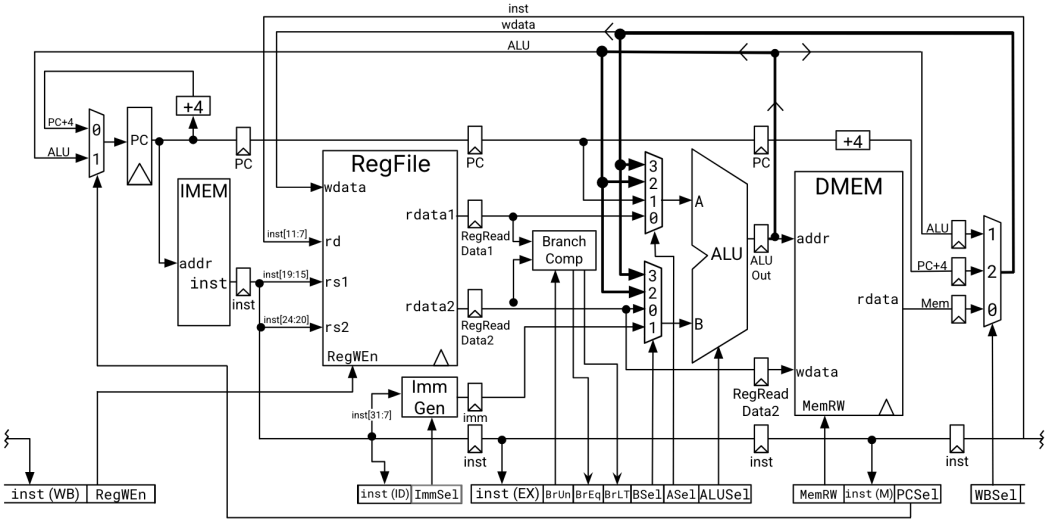


Figure 2: 5-Stage datapath diagram with MEM ⟶ EX and WB ⟶ EX forwarding paths

Q5.3 (2 points) Suppose we implement both forwarding paths in Figure 2.

```
1  slli t0 a0 2
2  xor  t1 a0 a1
3  addi t0 t0 4
```

During the cycle where `addi t0 t0 4` is in the EX stage, which data source must ASel select to resolve the hazard without stalling?

- ○ 0: the value from the Register File
- ○ 1: the Program Counter
- ○ 2: the forwarded value from the MEM pipeline register
- ● 3: the forwarded value from the WB pipeline register

Solution: The `slli` instruction (which writes to `t0`) is 2 instructions ahead of `addi` (which reads `t0`). When `addi` is in EX, `slli` is in WB, so we forward from the WB pipeline register.

Inst/cycles	1	2	3	4	5	6	7	8	9	10
`slli t0 a0 2`	IF	ID	EX	MEM	WB					
`xor t1 a0 a1`		IF	ID	EX	MEM	WB				
`addi t0 t0 4`			IF	ID	EX	MEM	WB			

Note: At Cycle 5, `slli` is in WB and `addi` is in EX. The value of `t0` is forwarded from WB to EX.

Q5.4 (1.5 points) Suppose we implement the RISC-V 5-stage pipelined datapath on the reference card.

```
1 lw   t0 0(s0)
2 addi t1 t0 4
3 sw   s0 4(t1)
```

Which of the hardware mechanisms must be implemented to resolve the data hazards in the code above with the minimum number of stalls?

- ■ MEM ⟶ EX forwarding
- ■ WB ⟶ EX forwarding
- ☐ Write-Before-Read Regfile
- ◯ None

Solution: The `lw` → `addi` dependency requires a single load-use stall. Because the stall moves the `addi` EX stage to Cycle 5, the data from `lw` is forwarded using the WB → EX path.

The `addi` → `sw` dependency is resolved with MEM → EX forwarding, as the `addi` result from its EX stage (Cycle 5) is forwarded to the `sw` EX stage (Cycle 6). No further stalls are required.

Inst / Cycle	1	2	3	4	5	6	7	8
`lw t0, 0(s0)`	IF	ID	EX	MEM	WB			
`addi t1, t0, 4`		IF	ID	ID	EX	MEM	WB	
`sw s0, 4(t1)`			IF	IF	ID	EX	MEM	WB

Note: `lw` WB → `addi` EX (Cycle 5) and `addi` MEM → `sw` EX (Cycle 6).

Solution: A, E, C, B, F, D – By interleaving independent loads between dependent uses, we reduce stalls. The reordering A, E, C, B, F, D places independent instructions between dependent ones to give load values time to propagate.

Q5.5 (1.5 points) In Project 3 we pipelined the CPU along the dashed line in Figure 3. What possible hazards can occur in this 2-stage pipeline?

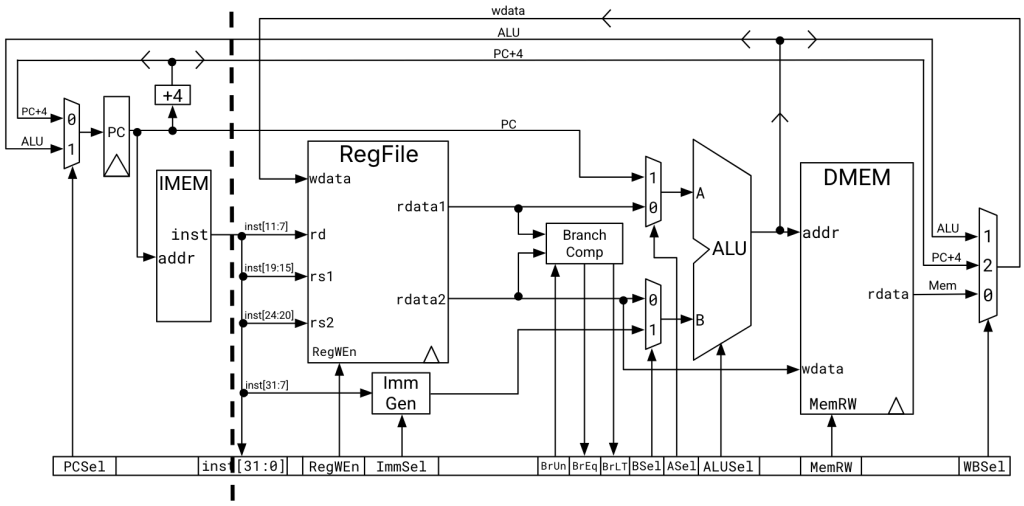


Figure 3: 2-Stage Pipeline Diagram

■ Control Hazards ☐ Data Hazards ☐ Structural Hazards ○ None

Solution: Control Hazards exist because branches/jumps are resolved in Stage 2, but the next instruction is already being fetched in Stage 1, requiring a flush. Data Hazards are avoided because instructions are Decoded, Executed, and Written Back in a single stage (Stage 2), so the next instruction always sees the updated Register File. Structural Hazards are eliminated by using separate IMEM and DMEM.

Q5.6 (3 points) **Instruction scheduling**: Suppose we implement the RISC-V 5-stage datapath on the reference card. Assume the RegFile does not support write-then-read. Reorder the instructions on the right to minimize pipeline stalls while ensuring the final architectural state (register and memory values) remains identical to the original sequence.

```
1  lw t1 0(s0)
2  sw t1 0(s1)
3  lw t2 0(t1)
4  sw t2 4(s1)
5  lw t3 4(s0)
6  sw t3 8(s1)
```

Fill in the grid below. The answer choices "First, Second, Third, etc." represent execution order of your reordered instructions. The first and fifth instruction are filled in for you.

First	● Line 1	○ Line 2	○ Line 3	○ Line 4	○ Line 5	○ Line 6
Second	○ Line 1	○ Line 2	○ Line 3	○ Line 4	● Line 5	○ Line 6
Third	○ Line 1	○ Line 2	● Line 3	○ Line 4	○ Line 5	○ Line 6
Fourth	○ Line 1	● Line 2	○ Line 3	○ Line 4	○ Line 5	○ Line 6

(Question 5 continued…)

Fifth	○ Line 1	○ Line 2	○ Line 3	○ Line 4	○ Line 5	● Line 6
Sixth	○ Line 1	○ Line 2	○ Line 3	● Line 4	○ Line 5	○ Line 6

Q6 *61Cache or Venmo* **(11 points)**

Q6.1 (1.5 points) Suppose that **Computer A** has **1 MiB** memory and a single L1 data cache, **Cache A**, which is is a write-back, 128-byte direct-mapped cache with 16-byte blocks.

What is the tag-index-offset breakdown for **Cache A**?

Tag: 13 bit(s)	Index: 3 bit(s)	Offset: 4 bit(s)

Q6.2 (1.5 points) Now Suppose **Computer B** has **1 MiB** memory and a single L1 data cache, **Cache B**, which is a write-back, 128-byte 2-way associative cache with 16-byte blocks.

What is the tag-index-offset breakdown for **Cache B**?

Tag: 14 bit(s)	Index: 2 bit(s)	Offset: 4 bit(s)

In the code below, assume that `sizeof(int) = 4`.

```
1 int data[12];   // starts at address 0x01000
2 int coeff[12];  // starts at address 0x01400
3
4 for (int i = 2; i < 10; i++) {
5     data[i] += coeff[i] * (data[i-1] - data[i-2]);
6 }
```

Q6.3 (3 points) Suppose we ran this code on Computers A and B. Fill in the blank: Assuming caches start cold, the overall cache hit rate for Cache A is _____ that for Cache B.

- ○ higher than
- ● lower than
- ○ comparable to

Select a choice, then justify in no more than 20 words.

Higher. Cache B's 2-way associativity eliminates the conflict misses in Cache A where data and coeff elements map to the same index, allowing both to coexist and improving the hit rate.

Q6.4 (1 point) Suppose a system on **Computer C** has a two-level cache hierarchy:

- **L1 Cache:** 95% hit rate, 2-cycle hit time
- **L2 Cache:** 80% local hit rate, 15-cycle hit time
- **Main memory:** 125-cycle miss penalty

Calculate the AMAT in cycles.

4 cycles

Q6.5 (4 points) Suppose **Computer C** has **4 GiB** memory and a single L1 data cache, **Cache C**, which is a write-back, 1KiB direct-mapped cache with 64-byte blocks. We design a longer program as a benchmark workload and run it on Computer C. We observe a large amount of cache misses, both compulsory and non-compulsory.

We are interested in lowering our AMAT on this workload. Consider how the following modifications to **Cache C** impact the overall hardware complexity and each component of AMAT (Hit Time, Miss Penalty, Miss Rate) for this workload.

Modification	**Hit Time**	**Miss Penalty**	**Miss Rate**	**Hardware Complexity**
1. Change cache size to 4 MiB (same block size, same associativity)	● Increase ○ Decrease ○ Minimal/ no effect	○ Increase ○ Decrease ● Minimal/ no effect	○ Increase ● Decrease ○ Minimal/ no effect	● Increase ○ Decrease ○ Minimal/ no effect
2. Change to fully associative with block size to 1 byte (same cache size)	● Increase ○ Decrease ○ Minimal/ no effect	○ Increase ● Decrease ● Minimal/ no effect	● Increase ○ Decrease ○ Minimal/ no effect	● Increase ○ Decrease ○ Minimal/ no effect

Solution:

Modification 1: Increase cache size to 4 MiB (same block size, same associativity)

- **Hit Time (Increase):** A larger cache has more sets, which requires a larger index decoder and longer wires to access the SRAM. This increases the time to look up and return a block on a hit.
- **Miss Penalty (Minimal/no effect):** Miss penalty is determined by the time to fetch a block from the next level of memory. Block size is unchanged, so the number of bytes transferred per miss is the same.
- **Miss Rate (Decrease):** A larger cache holds more blocks, reducing capacity misses. More of the working set fits in the cache at once.
- **Hardware Complexity (Increase):** More SRAM cells, larger tag arrays, and a larger index decoder are required.

Modification 2: Change to fully associative with block size of 1 byte (same cache size)

- **Hit Time (Increase):** A fully associative cache compares the tag against every line in parallel. With a 1-byte block size, the cache contains the maximum number of lines, requiring the maximum number of parallel comparators. This significantly increases hit time.
- **Miss Penalty (Decrease or Minimal/no effect):** If miss penalty is modeled as proportional to block size, transferring 1 byte is faster than transferring a full block (decrease). However, most of the miss penalty comes from DRAM access latency rather than transfer time, so the penalty may not change significantly (minimal/no effect).
- **Miss Rate (Increase):** A 1-byte block size eliminates spatial locality. Each access to a nearby byte becomes a separate miss instead of hitting on a block already brought in. Full associativity removes conflict misses, but the loss of spatial locality dominates, so miss rate increases overall.
- **Hardware Complexity (Increase):** Fully associative lookup requires one comparator per cache line, and a 1-byte block size maximizes the number of lines. Each byte also requires its own tag, and the tag is much larger than 1 byte, so tag storage exceeds data storage.

Q7 *61Cache-ually VM* 🐐 **(12 points)**

Suppose our virtual memory system has a 1 GiB virtual memory space and 4096-byte pages. The physical page number (PPN) is 16 bits wide.

Q7.1 (0.5 points) What is the size of the physical address space? Express your answer in integer bytes with the largest possible IEC prefix, (e.g., 777 PiB).

256 MiB

Q7.2 (1 point) How many bits are in the virtual page number (VPN) and offset?

VPN: 18 bit(s) | PPN: 16 bits(s) | Offset: 12 bit(s)

A single page table entry (PTE) on our virtual memory system is 4 bytes, partioned as below.

1 Valid Bit	15 Status Bits	16 PPN Bits

Q7.3 (1 point) What is the size of a page table on our system? Express your answer in integer bytes with the largest possible IEC prefix, (e.g., 777 PiB).

1 MiB

Q7.4 (0.5 points) Given your answer to the previous part, does a page table fit on a single page?

○ Yes ● No ○ Depends

Q7.5 (3 points) The C function `get_ppn` takes a 32-bit page table entry `pte` (formatted above) and extracts the PPN if the valid bit is set (1), and `-1` otherwise. For example, `get_ppn(0xB055CAFE)` returns `0x0000CAFE` and `get_ppn(0x7890ABCD)` returns `0xFFFFFFFF` (`-1`).

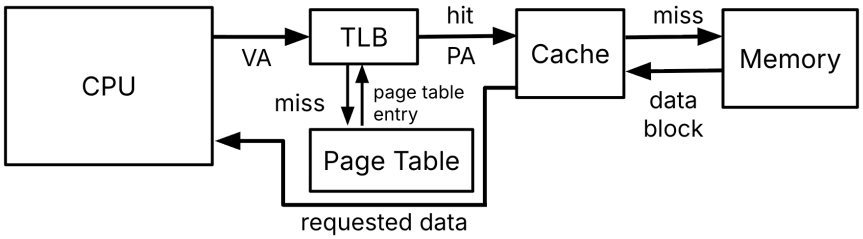
Write **one bitwise expression** that involves only bitwise operations (`&`, `|`, `~`, `^`, `<<`, `>>`), parenthesis, the variable `pte`, and hexadecimal or decimal constants.

Hint: The argument `pte` is signed for your convenience.

```
uint32_t get_ppn(int32_t pte) {

  int32_t ppn = (pte & 0xFFFF) | ~((int32_t)pte >> 31);
                                 Q7.6
  return (uint32_t) ppn;
}
```

Consider a memory hierarchy that combines caches and virtual memory. On a memory access, virtual addresses (VAs) are translated to physical addresses (PAs) before the single L1 cache is addressed. During address translation, assume the translation lookaside buffer (TLB) is checked first, then the page table.



Q7.7 (3 points) Which are the following statements are true?

- ■ The page table is located in physical memory.
- ☐ The TLB stores a copy of the cache's least recently used block.
- ■ A page fault can occur when the corresponding PTE is invalid.
- ☐ Up to two page faults can occur on any given memory access.
- ■ AMAT is now the combined time for address translation and data access.
- ☐ To improve AMAT, fix pages and cache blocks to the same size.
- ○ None of the above

Q7.8 (3 points) Which of the below combinations of events are possible across the TLB, the virtual memory system, and cache?

Complete the below table; some entries are filled out for you.

	TLB	Page	Cache	Possible?
1	Hit	In memory	Hit	● Possible ○ Impossible
2	Hit	In memory	Miss	● Possible ○ Impossible
3	Hit	Not in memory	Hit	○ Possible ● Impossible
4	Hit	Not in memory	Miss	○ Possible ● Impossible
5	Miss	In memory	Hit	● Possible ○ Impossible
6	Miss	In memory	Miss	● Possible ○ Impossible
7	Miss	Not in memory	Hit	○ Possible ● Impossible
8	Miss	Not in memory	Miss	● Possible ○ Impossible

Solution:

- Impossible: TLB hit while page is not in memory. Cannot have a translation in TLB is page is not present in memory.
- Impossible: Cache Hit while page is in memory. Data cannot be in cache if page is not in memory.
- On TLB misses, it is possible for caches to have the necessary data as long as the page is also in memory.

Q8 *61CascAbel* **(16 points)**

Complete the `count_fruit` function on the following page using SIMD vector instructions to speed up computation.

As an extension to Project 1, the `count_fruit` function counts the total number of fruits, represented by the character `'*'`, on the game board `board`. The `game_t` struct from Project 1 is shown to the right.

```
1 struct game_t {
2   unsigned int num_rows;
3   char** board;
4   unsigned int num_snakes;
5   snake_t* snakes;
6 };
```

You have access to the following SIMD operations. A single vector is a 256-bit vector register capable of holding thirty-two 8-bit characters. **You may use at most one operation per blank.**

- `vector vec_load(char *A)`: Loads 32 characters at memory address `A` into a vector.
- `vector vec_setchar(char c)`: Creates a vector where every element is `c`.
- `vector vec_cmpeq(vector A, vector B)`: Computes `A == B` element-wise. Element-wise result is `0xFF` if true, `0x00` otherwise.
- `vector vec_add(vector A, vector B)`: Computes `A + B` element-wise.
- `uint32_t vec_movemask(vector A)`: Take the most significant bit of each 8-bit element in the vector, and stores the result into a single `uint32_t`.

Line 11 calls the `popcount()` function from the midterm. This function, `int popcount(uint32_t val)`, returns the number of set bits (`1`s) in a 32-bit integer `val`. The usage has been provided for you in the following code.

(The rest of this page is blank.)

(8 points) Complete the **count_fruit** function as described on the previous page.

```
1  uint32_t count_fruit(game_t* game){
2    uint32_t total = 0;
3    vector v_fruit = vec_setchar('*');
                      Q8.1
4    for (unsigned int r = 0; r < game->num_rows; r++) {
                                  Q8.2
5      char* row_ptr = game->board[r];
6      int row_len = strlen(row_ptr);
7      for (int c = 0; c <= (row_len / 32 * 32) - 32; c += 32) {
                                  Q8.3
8        vector data = vec_load(&row_ptr[c]);
                       Q8.4
9        vector match = vec_cmpeq(data, v_fruit);
                        Q8.5
10       uint32_t top_bits = vec_movemask(match);
                             Q8.6
11       total += popcount(top_bits);
12     }
13     for (int c = (row_len / 32 * 32) - 32; c < row_len; c++) {
                    Q8.7                          Q8.8
14       if (row_ptr[c] == '*') {
                  Q8.9
15         total++;
             Q8.10
16       }
17     }
18   }
19   return total;
20 }
```

Q8.11 (2 points) We now want to speed up the performance of **count_fruit** with thread level parallelism (OpenMP). We make one modification to our code above to produce code that is safe under parallel execution and that produces deterministic results.

To do so, fill in the blanks: Insert after [Line Number 3] the statement:

```
#pragma omp parallel for reduction (+:total)
```

We would like to count the number of odd numbers in the first 10 numbers. The (silly but functional) serial implementation is shown right.

```
int total = 0;
for (int i = 0; i < 10; i++){
    total += (i % 2);
}
```

Consider Code Block A:

```
1  omp_set_num_threads(2);
2  int total = 0;
3  #pragma omp parallel
4  for (int i = 0; i < 10; i++){
5      total += (i % 2);
6  }
```

Q8.12 (2 points) What are the minimum and maximum values possible of `total` after running this code?

Min: 1	Max: 10

Q8.13 (1 point) Which statement best describes the code execution?

- ○ Correct result, slower than serial
- ○ Correct result, faster than serial
- ● Sometimes incorrect result
- ○ Always incorrect result

Consider Code Block B:

```
1  omp_set_num_threads(4);
2  int totals[] = {0, 0, 0, 0};
3  #pragma omp parallel
4  {
5      int tid = omp_get_thread_num();
6      for (int i = tid; i < 10; i += 4) {
7          totals[tid] += i % 2;
8      }
9  }
10 int total = totals[0] + totals[1] + totals[2] + totlas[3];
```

Q8.14 (2 points) What are the minimum and maximum values possible of `total` after running this code?

Min: 5	Max: 5

Q8.15 (1 point) Which statement best describes the code execution?

- ● Correct result, slower or comparable to serial
- ○ Correct result, faster than serial
- ○ Sometimes incorrect result
- ○ Always incorrect result

Solution: Cache coherency is the biggest slowdown here! Parallelizing a 10 iteration for loop is often not worth the overhead.

NOTE: During the exam, the number of threads was set to 2 while the size of the array was 4. The intended problem had 4 threads with an array of size 4, but points were given for multiple solutions due to the confusion.

Q9 *61Celebration* 🎉 **(0 points)**

These questions will not be assigned credit. Feel free to leave them blank.

Q9.1 What cool stuff are you doing this summer? Draw a picture :)

Q9.2 If there's anything else you want us to know, or you feel like there was an ambiguity in the exam, please put it in the box below.

For ambiguities, you must qualify your answer and provide an answer for both interpretations. For example, "if the question is asking about A, then my answer is X, but if the question is asking about B, then my answer is Y". You will only receive credit if it is a genuine ambiguity and both of your answers are correct. We will only look at ambiguities if you request a regrade.