Question 1: (0 points)
Bank of Questions

Performance Analysis (V01, V09, V0B, V11)

The following questions study the RISC-V assembly code for the `FindMax` procedure shown in Figure 1. For simplicity, the RISC-V code for storing and restoring callee-saved registers to and from the stack are omitted.

```
1  # FindMax(Square, N, M)
2  # Input Parameters
3  #   a0: Square is the address of first element of 2D matrix
4  #   a1: N is the number of rows in Square
5  #   a2: M is the number of columns in Square
6  # Return Value:
7  #   a0: value of maximum element in Square
8  #
9  0x1FFF FFAC FindMax:    li      s0, -1              # max <- -1
10 0x1FFF FFB0             add     s1, zero, zero      # i <- 0
11 0x1FFF FFB4 NextRow:    slt     t6, s1, a1          # if i<N then t6 <- 1
12 0x1FFF FFB8             beq     t6, zero, Return    # if i>=N Return
13 0x1FFF FFBC             add     s2, zero, zero      # j <- 0
14 0x1FFF FFC0 NextColumn: slt     t6, s2, a2          # if j<M then t6 <- 1
15 0x1FFF FFC4             beq     t6, zero, RowDone   # if j>=M RowDone
16 0x1FFF FFC8             mul     t1, s1, a2          # t1 <- i*M
17 0x1FFF FFCC             add     t2, t1, s2          # t2 <- i*M+j
18 0x1FFF FFD0             slli    t3, t2, 2           # t3 <- 4*(i*M+j)
19 0x1FFF FFD4             add     t4, a0, t3          # t4 <- &(Square[i][j])
20 0x1FFF FFD8             lw      t5, 0(t4)           # t5 <- Square[i][j]
21 0x1FFF FFDC             slt     t6, s0, t5          # if(max < Square[i][j]) then t6 <- 1
22 0x1FFF FFE0             beq     t6, zero, NoChange
23 0x1FFF FFE4             add     s0, t5, zero        # max <- Square[i][j]
24 0x1FFF FFE8 NoChange:   addi    s2, s2, 1           # j <- j+1
25 0x1FFF FFEC             jal     zero, NextColumn
26 0x1FFF FFF0 RowDone:    addi    s1, s1, 1           # i <- i+1
27 0x1FFF FFF4             jal     zero, NextRow
28 0x1FFF FFF8 Return:     add     a0, s0, zero        # a0 <- max
29 0x1FFF FFFC             jalr    zero, ra, 0
```

Figure 1: RISC-V Assembly code for `FindMax` procedure.

Question 2: (4 points)
Consider the following invocation of the procedure `FindMax`

```
lui     a0, 0x002
li      a1, 0x1F4
li      a2, 0x3E8
call    FindMax
```

What are the values, expressed in decimal, of the parameters `N` and `M` for this call to `FindMax`?

Solution: We simply have to convert the hexadecimal values given into decimal

`N = 0x1F4` $= 16^2 + 15 \times 16 + 4 = 256 + 240 + 4 = 500$

`M = 0x3E8=` $3 * 16^2 + 14 * 16 + 8 = 768 + 224 + 8 = 1000$

Question 3: (4 points)

In a given invocation of `FindMax`, `N = 10000` and `M = 5000` and the condition for the branch in line 22 is true 50% of the time. How many instructions are executed by this call?

Solution: To solve this question, we need to analyze the assembly code to determine how many times each instruction is executed:

- Instructions in lines 9, 10, 28, and 29 are not inside any loop and therefore each is executed once.
- Instructions in lines 11, 12, 13, 26, and 27 are executed by the outer loop but are not executed by the inner loop. Thus each of these instructions is executed N times.
- Instructions in lines 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, and 25 are executed once for each iteration of the inner loop. Therefore these instructions are executed $N \times M$ times.
- Once the last iteration of the inner loop executes the jump instruction at line 25, instructions at lines 14 and 15 are executed one more time. This happens N times.
- Similarly, when the last time that the jump instruction at line 27 is executed, the instructions at lines 11 and 12 are executed to get out of the outer loop. Thus there are two more instructions executed.
- The instruction in line 23 is only executed when the branch in line 22 is not taken, therefore it is executed 50% of the times that the inner loop is executed. Thus, this instruction is executed $0.5 \times N \times M$.

The number of instructions executed by `FindMax`, for this call, is given by:

$$\begin{aligned} \#\text{ of instructions} &= 6 + N \times (7 + 11.5 \times M) \\ &= 6 + 7 \times N + 11.5 \times N \times M \end{aligned}$$

Thus, for the specific execution, we have

$$\begin{aligned} \#\text{ of instructions} &= 6 + 7 \times 10000 + 11.5 \times 10000 \times 5000 \\ &= 575,070,006 \end{aligned}$$

Question 4: (4 points)

Several executions of programs that are similar to `FindMax` have been used to determine the number of clock cycles executed by each type of instructions in the RISC-V processor that is executing `FindMax`. It was determined that the following instructions take one cycle each: `li`, `slt`, `add`, `slli`, `addi`. The `mul` instruction takes five cycles. Branch instructions take four cycles each, the jump instructions `jal` and `jalr` take two cycles each, and a load-word instruction takes ten cycles. How many clock cycles are necessary to execute an invocation of `FindMax` with `N = 10000` and `M = 5000` described above?

Solution: The code for `FindMax` executes double-nested loop. The outmost loop starts in line 11 and the jump instruction that returns to the start of the loop is at line 26. A similar reasoning as explained in the answer of the item above.

The number of cycles required to execute `FindMax` is:

Lines	Cycles	# Times Executed	Total Cycles
9, 10	$1+1$	1	2
11, 12	$1+4$	$N+1$	$5N+5$
13, 26, 27	$1+1+2$	N	$4N$
14, 15	$1+4$	$N(M+1)$	$5NM+5N$
16, 17, 18, 19, 20, 21, 22, 24, 25	$5+1+1+1+10+1+4+1+2$	NM	$26NM$
23	1	$0.5NM$	$0.5NM$
28, 29	$1+2$	1	3

$$\#\text{ of clock cycles} = 10 + 14N + 31.5NM$$

For the specific invocation:

$$\begin{aligned} \#\text{ofclockcycles} &= 10 + 14 \times 10000 + 31.5 \times 10000 \times 5000 \\ &= 1,575,140,010 \end{aligned}$$

Question 5: (4 points)

What is the average number of clocks per instruction (CPI) for the invocation of `FindMax` with `N = 10000` and `M = 5000` described above?

Solution:

$$\text{CPI} = \frac{\text{Number of Clock Cycles}}{\text{Number of Instructions}} = \frac{1,575,140,010}{575,070,006} = 2.7 \frac{\text{Clock Cycles}}{\text{Instruction}}$$

Question 6: (4 points)

If the invocation of `FindMax` with `N = 10000` and `M = 5000` described above is executing in a RISC-V processor running with a clock frequency of 4 GHz, how long does it take to execute `FindMax`?

Solution:

$$\text{Clock Cycle} = \frac{1}{4 \times 10^9 Hz} = 0.25 \times 10^{-9} s = 0.25\ ns$$

$$\text{Time} = \text{Number of Clock Cycles} \times 0.25 \times 10^{-9} s = 1.575 \times 10^9 \times 0.25 \times 10^{-9} = 0.39\ s$$