Please write your name and Kerberos on your solutions.

The only material you may access for this exam is this pdf, the RISC-V ISA handout (https://6004.mit.edu/web/_static/fall20/resources/references/6004_isa_reference.pdf) and the quiz 2 answer template if you would like to use it (https://6004.mit.edu/web/_static/fall20/resources/quizzes/q2_answer_template.pdf).

Each subproblem is labelled with (Label: X), (e.g., Label: 2A_1). If you are writing your solutions on blank paper, please make sure you write down the label of each subproblem together with its solution and circle each answer. So, for example, if your computed value for problem 2A_1 is 5, you would write:

2A_1: 5

Problem 1. Honor Code Agreement (0 points)

We are using the honor system during this exam, and ask that you accept the following terms of this honor system:

1. You will not share the exam with anyone
2. You will not discuss the material on the exam with anyone until after solutions are released
3. You understand that the exam is closed book and that you may only use reference material provided with the exam

(label: 1) I will abide by the above terms (circle one): **Agree** / **Do Not Agree**
Problem 2. Combinational Circuits (15 points)

(A) (2 points) Consider a function, \( \text{mod}3 \), that takes an unsigned 2-bit input \( x \) and returns a 2-bit result which is equal to \( x \) modulo 3. Your result should be in the range of \( \{0, 1, 2\} \). Fill in the truth table below so that it describes the correct behavior of this function.

(label: 2A)

<table>
<thead>
<tr>
<th>( x[1] )</th>
<th>( x[0] )</th>
<th>( \text{out}[1] )</th>
<th>( \text{out}[0] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(B) (3 points) Implement the \( \text{mod}3 \) function in Minispec by filling in the function definition below. Your code should describe a circuit that when synthesized manually without optimizations results in at most 4 (one or two input) logic gates. Your solution can use the following gates: inverter, 2-input AND, OR, NAND, NOR, or XOR gates. (Hint: The bitwise logical operations in Minispec are: \( \sim \) (NOT), \& (AND), | (OR), ^ (XOR)).

(label: 2B)

```minispec
function Bit#(_2__) mod3 (Bit#(_2__) x);

    return __{x[1] & ~x[0], ~x[1] & x[0]} __________________________.;

endfunction
```

(C) (2 points) Manually synthesize your function into a combinational circuit.

(label: 2C)
(D) (8 points) Complete the truth table for the following Minispec function.

```minispec
function Bit#(3) f(Bit#(3) a);
    Bit#(3) ret = 3'b100;
    case {{a[2],a[0]}}
        0: ret = {1'b0, a[1]^a[0], 1'b1};
        1: ret = signExtend(a[1]) & ret;
        3: ret = {a[0], ~a[2:1]};
        default: ret = 3'b001;
    endcase
    return ret;
endfunction
```

(label: 2D)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Problem 3. Sequential Logic Timing (15 points)

Consider the sequential circuit below, which includes registers made of D flip flops, in addition to combinational logic. All registers share a common clock, which is not shown. The squares that begin with “R” denote registers. The stars that begin with “CL” denote combinational logic. The table below contains the propagation delay and contamination delay of the components.

(A) (4 points) What are the propagation ($t_{PD}$) and contamination delays ($t_{CD}$) of the circuit? Recall that these are measured from the rising edge of the clock to the output.

$t_{PD}$ and $t_{CD}$ are measured from rising edge of the clock to the output, so they equal the propagation delay and contamination delay of R3.

(label: 3A_1) $t_{PD}$ (ps): _____1_____

(label: 3A_2) $t_{CD}$ (ps): _____1_____

(B) (5 points) What is the minimum clock period we can use for this circuit to function properly?

$R1 \rightarrow R2$: $t_{PD,R1} + t_{PD,CL1} + t_{S,R2} = 2 + 3 + 1 = 6\text{ps}$

$R2 \rightarrow R3$: $t_{PD,R2} + t_{PD,CL2} + t_{S,R3} = 2 + 2 + 6 = 10\text{ps}$

$R3 \rightarrow R2$: $t_{PD,R3} + t_{PD,CL1} + t_{S,R2} = 1 + 3 + 1 = 5\text{ps}$

$t_{hold,R2} \leq t_{CD,R1} + t_{CD,CL1}$ $2 \leq 1 + 2$ Satisfied

$t_{hold,R3} \leq t_{CD,R2} + t_{CD,CL2}$ $1 \leq 2 + 1$ Satisfied

(label: 3B_1) Minimum clock period for correct operation (ps): _____10_____

6.004 Fall 2020 - 4 of 17 - Quiz #2
(C) (6 points) We would like to have a functioning circuit with a clock period of 5 picoseconds. The following table shows possible replacements for each component in the circuit along with the price for the replacement. What is the lowest amount of money we need to spend to achieve this goal? **Make sure that your new circuit obeys all timing constraints.** (A duplicate of the circuit diagram and original timing parameters are shown above for ease of reference).

<table>
<thead>
<tr>
<th>Component</th>
<th>( t_{PD} )</th>
<th>( t_{CD} )</th>
<th>( t_{SETUP} )</th>
<th>( t_{HOLD} )</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1-New</td>
<td>1 ps</td>
<td>1 ps</td>
<td>2 ps</td>
<td>1 ps</td>
<td>$1.50</td>
</tr>
<tr>
<td>R2-New</td>
<td>2 ps</td>
<td>2 ps</td>
<td>1 ps</td>
<td>1 ps</td>
<td>$2.00</td>
</tr>
<tr>
<td>R3-New</td>
<td>1 ps</td>
<td>1 ps</td>
<td>1 ps</td>
<td>4 ps</td>
<td>$5.00</td>
</tr>
<tr>
<td>CL1-New</td>
<td>2 ps</td>
<td>2 ps</td>
<td>N/A</td>
<td>N/A</td>
<td>$1.00</td>
</tr>
<tr>
<td>CL2-New</td>
<td>2 ps</td>
<td>2 ps</td>
<td>N/A</td>
<td>N/A</td>
<td>$4.00</td>
</tr>
</tbody>
</table>

First replace R3 with R3-New to get rid of 6 ps setup time.

**R2 → R3-New: \( t_{PD,R2} + t_{PD,CL2} + t_{S,R3-New} = 2 + 2 + 1 = 5ps \)**

Check if hold time of R3-New is satisfied:

\[ t_{hold,R3-New} \leq t_{CD,R2} + t_{CD,CL2} \] is 4 \( \leq 2 + 1 \) Not satisfied

Replace CL2 because \( t_{CD,R2-New} \) is the same as \( t_{CD,R2} \).

**R2 → R3-New: \( t_{PD,R2} + t_{PD,CL2-New} + t_{S,R3-New} = 2 + 2 + 1 = 5ps \)**

\[ t_{hold,R3-New} \leq t_{CD,R2} + t_{CD,CL2-New} \] is 4 \( \leq 2 + 2 \) Satisfied

Now, we need to fix the R1 \( \rightarrow \) R2 path:

**R1 → R2: \( t_{PD,R1} + t_{PD,CL1} + t_{S,R2} = 2 + 3 + 1 = 6ps \)**

Can either replace R1 or CL1. CL1-New is cheaper so replace CL1.

**R1 → R2: \( t_{PD,R1} + t_{PD,CL1-New} + t_{S,R2} = 2 + 2 + 1 = 5ps \)**

\[ t_{hold,R2} \leq t_{CD,R1} + t_{CD,CL1-New} \] 2 \( \leq 1 + 2 \) Still Satisfied

**R3-New → R2: \( t_{PD,R3-New} + t_{PD,CL1-New} + t_{S,R2} = 1 + 2 + 1 = 4ps \)**

(label: 3C_1) **Minimum money spent ($):** ____________________________10_________________________

(label: 3C_2) **Name(s) of replacement(s) purchased:** ____R3-New, CL1-New, CL2-New____
Problem 4. Finite State Machines and Sequential Circuits in Minispec (19 points)

Suppose we want to create a system that decides if the concatenation of its previous 2 single-bit inputs is a power of 2 (where the MSB is the input from 2 cycles ago and the LSB is from 1 cycle ago). If the previous 2 bits (prior to the current input) are a power-of-two the system outputs a 1, otherwise it outputs 0. Before any input is sent, assume the initial previous 2 bits are 2'b00.

A partial FSM diagram of this circuit is shown below:

*Before receiving any inputs the FSM is in state A.*

(A) (7 points) For this FSM to provide the correct answer, to what existing states must D transition to (A, B, C, or D), and what output does D give (0 or 1)?

*(label 4A_1) Current State = D, Input = 0, Next State = ___A_____

*(label 4A_2) Current State = D, Input = 1, Next State = ___B_____

*(label 4A_3) Current State = D, Output = ___1_____

(B) (2 points) Using the partial FSM, fill out the truth table below.

*(label 4B)*

<table>
<thead>
<tr>
<th>State</th>
<th>Input</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>D</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>Part_A</td>
<td>Part_A</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>Part_A</td>
<td></td>
</tr>
</tbody>
</table>
(C) (10 points) We now want to implement a different version of this is-power-of-2 sequential circuit in minispec. In this version, if the previous 5 bits are a power of 2 the module’s getOutput method will return \( \log_2(\text{previous 5 bits}) \), otherwise it will return -1.

To determine is-power-of-2, we will use the following identity for integers \( X \geq 0 \) and \( N > 0 \):

\[
X = 0 \text{ or } X = 2^N \implies X \& (X - 1) = 0
\]

The module has 2 registers:
- \texttt{prevBits}: contains the 5 bits to check for is-power-of-2 in the current cycle.
- \texttt{newOneIndex}: contains the index of the most recent “1” bit in \texttt{prevBits}. Hint: how does this relate to \( \log_2(\text{prevBits}) \)? (Note: this value is stored as a 4-bit 2’s complement value to support initializing it to -1).

**Fill in the minispec implementation of the described module:**

```minispec
(label 4C)
module PowTwo;
    Reg#(Bit#(5)) prevBits(0);
    Reg#(Bit#(4)) newOneIndex(-1);

    input Bit#(1) in;

    rule tick;
        prevBits <= ___{prevBits[3:0], in}________________; 
        if (in == 1) newOneIndex <= ___0________________; 
        else if (newOneIndex == 4) newOneIndex <=__-1____; 
        else if (newOneIndex != -1) 
            newOneIndex <= newOneIndex + 1;
    endrule

    method Bit#(4) getOutput();
        // fill in missing code for this method
        if ((prevBits & (prevBits - 1)) == 0) return newOneIndex;
        else return -1;
    endmethod
endmodule
```
Problem 5. Arithmetic Pipelines (16 points)

One day, you are given a mysterious module, named “Module A.” This module has three inputs, X, Y, and Z, and a single output, O. You are told that the circuit functions, but its throughput is too low. You decide to take a look and try to pipeline the circuit.

For each of the questions below, please create a valid K-stage pipeline of the given circuit. Each component in the circuit is annotated with its propagation delay in nanoseconds. Show your pipelining contours and place large black circles (●) on the signal arrows to indicate the placement of pipeline registers. Give the latency and throughput of each design, assuming ideal registers (tPD=0, tSETUP=0). Remember that our convention is to place a pipeline register on each output.

(A) (4 points) Show the maximum-throughput 2-stage pipeline using a minimal number of registers. What is the latency and throughput of the resulting circuit? Pay close attention to the direction of each arrow.

(label 5A_1) Latency (ns): _____18_____  
(label 5A_2) Throughput (ns⁻¹): _____1/9_____
(B) (4 points) Show the **maximum-throughput pipeline** using a minimal number of registers. What is the latency and throughput of the resulting circuit?

![Diagram](image)

(label 5B_1) Latency (ns): ____21_____

(label 5B_2) Throughput (ns\(^{-1}\)): ____1/7_____

---

6.004 Fall 2020 - 9 of 17 - Quiz #2
(C) Now, you are given a new module, “Module B.” This module has a single input, $U$, and a single output, $W$. You are also given the implementation of Module B, a 2-stage pipeline with registers denoted by the large black circles ($\bigcirc$). You want to connect modules A and B, so the output $O$ of Module A is connected to the input $U$ of Module B.

![Implementation of Module B](image)

(i) (2 points) Given the implementation of Module B above, what is the latency and throughput?

\[
\text{(label 5C_1_1) Latency (ns): } 22 \\
\text{(label 5C_1_2) Throughput (ns}^{-1}) : \frac{1}{11}
\]

(ii) (6 points) When connecting Module A to Module B, you have two options for Module A: your two-stage pipeline, or your maximum-throughput pipeline. If you want to maximize throughput, while minimizing latency and the number of registers used, which implementation of Module A would you use? What would the latency and throughput of the combined device be?

\[
\text{(label 5C_2_1) Module A implementation (circle one): } 2\text{-stage pipeline} \quad \text{Maximum-throughput pipeline}
\]

\[
\text{(label 5C_2_2) Latency (ns): } 44 \\
\text{(label 5C_2_3) Throughput (ns}^{-1}) : \frac{1}{11}
\]
Problem 6. Processor Implementation (15 points)

Ben Bitdiddle has written code, in RISC-V assembly, that repeatedly loads an address from memory and jumps to that address. Ben’s code has this code pattern repeated many times:

\[
\begin{align*}
\text{lw} & \ x2, \ 0(x1) \\
\text{jalr} & \ ra, \ 0(x2)
\end{align*}
\]

He wants his code to run efficiently, so he decides to change the processor implementation in order to execute the above code in one cycle. He wants to add the following instruction to the RISC-V ISA:

\[
\text{lwj} \ rd, \ \text{offset}(rs1)
\]

The behavior of the new \textit{lwj} instruction is to load data from memory based on the immediate and rs1. Then to jump to that location.

\[
\begin{align*}
\text{reg}[rd] & \leq pc + 4 \\
pc & \leq \text{mem}[\text{reg}[rs1] + \text{offset}]
\end{align*}
\]

The encoding is as follows:

\[
\begin{array}{ccccccc}
31 & \ldots & 20 & 19 & \ldots & 15 & 14 & \ldots & 12 & 11 & \ldots & 7 & 6 & \ldots & 0 \\
\text{imm}[11:0] & & rs1 & & 010 & & rd & & 0110100
\end{array}
\]

(A) (2 points) What is the 32-bit binary encoding of the following instruction? Provide your answer in hexadecimal notation.

\[
\text{lwj} \ ra, \ 4(x10)
\]

\[
\begin{array}{ccccccc}
31 & \ldots & 20 & 19 & \ldots & 15 & 14 & \ldots & 12 & 11 & \ldots & 7 & 6 & \ldots & 0 \\
0000000000100 & & 01010 & & 010 & & 00001 & & 0110100
\end{array}
\]

(label 6A) Encoding in hexadecimal (0x): \text{______004520B4______}
In the diagram below, Ben has modified the Decoder to recognize and output the correct values and signals for the new \texttt{lwj} instruction.

(B) (8 points) For each of the following signals, does the mux being controlled by that signal need an extra input to accommodate the new instruction? If so, indicate the name of the signal that needs to be added as an input to the mux. If not, indicate which existing input of the mux should be selected by the mux to make the instruction work properly.

(label 6B_1) BSEL: Needs new input? \hspace{1cm} YES \hspace{1cm} NO

(label 6B_2) New input/Existing mux input signal: __\texttt{imm} (1) __________

(label 6B_3) WDSEL: Needs new input? \hspace{1cm} YES \hspace{1cm} NO

(label 6B_4) New input/Existing mux input signal: __\texttt{PC + 4} (0) __________

(label 6B_5) WERFSEL: Needs new input? \hspace{1cm} YES \hspace{1cm} NO

(label 6B_6) New input/Existing mux input signal: __\texttt{1} __________

(label 6B_7) PCSEL: Needs new input? \hspace{1cm} YES \hspace{1cm} NO

(label 6B_8) New input/Existing mux input signal: __\texttt{dataResult} __________
(C) (3 points) Additionally, decide for each of the following control signals what their values should be when executing the lwj instruction. If the value of the signal doesn’t matter, then put N/A. The possible values for each signal are provided below.

**AluFunc**: Add, Sub, And, Or, Xor, Slt, Sltu, Sll, Srl, Sra

**BrFunc**: Eq, Neq, Lt, Ltu, Ge, Geu

**MWR**: Read, Write

(label 6C_1) AluFunc: _____Add__________

(label 6C_2) BrFunc: ____N/A__________

(label 6C_3) MWR: ___Read___________

(D) (2 points) Ben decides that he wants to add more functionality into lwj so that lwj can now implement a shift left by 2 of the calculated address (reg[rs1] + offset) before fetching this updated address from memory. Can this be done by only changing existing control signals? If not, describe the limitation that prevents it.

(label 6D)

This is not possible because the ALU is already being used for the address calculation so it cannot be used to perform the shift left operation.
Problem 7. Caches (20 points)

Dilvina and Saniel are analyzing 4.006 grade statistics and are performing some hefty calculations, so they suspect that a cache could improve their system's performance.

(A) (3 points) They are considering using a 2-way set-associative cache with a block size of 4 (i.e. 4 words per line). The cache can store a total of 64 words. Assume that addresses and data words are 32 bits wide. To properly make use of locality, which address bits should be used for the block offset, the cache index, and the tag field?

Address bits used for byte offset: A[1 : 0]

(label 7A_1) Address bits used for tag field: A[31 : 7]

(label 7A_2) Address bits used for block offset: A[3 : 2]

(label 7A_3) Address bits used for cache index: A[6 : 4]

Block size of 4 → 2 bits for block offset
64 words / 4 words per block = 16 blocks
16 blocks / 2 ways = 8 sets → 3 bits for cache index
32 – 3 – 2 – 2 = 25 bits of tag

(B) (2 points) If Dilvina and Saniel instead used a direct-mapped cache with the same total capacity (64 words) and same block size (4 words), how would the following parameters in their system change?

Change in # of cache lines (select one of the choices below):

(label 7B_1) UNCHANGED ... +1 ... -1 ... 2x ... 0.5x ... CAN’T TELL
Accepted Unchanged or 2x as solutions because 2-way cache also has ½ the number of sets as the direct mapped cache, but the number of cache lines is actually the same.

Change in # of bits in tag field (select one of the choices below):

(label 7B_2) UNCHANGED ... +1 ... -1 ... 2x ... 0.5x ... CAN’T TELL

Block size of 4 → 2 bits for block offset
64 words / 4 words per block = 16 blocks → 4 bits for cache line index
32 – 4 – 2 – 2 = 24 bits of tag
Ultimately, they decided that the 2-way set associative cache would probably have better performance for their application, so the remainder of the problem will be considering a 2-way set associative cache. Below is a snapshot of this cache during the execution of some unknown code. V is the valid bit and D is the dirty bit of each set.

<table>
<thead>
<tr>
<th>Way 0</th>
<th></th>
<th>V</th>
<th>D</th>
<th>Tag</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0x32</td>
<td>0x0A</td>
<td>0x1A</td>
<td>0x2A</td>
<td>0x3A</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0x32</td>
<td>0x4B</td>
<td>0x5B</td>
<td>0x6B</td>
<td>0x7B</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0x32</td>
<td>0x3C</td>
<td>0x2C</td>
<td>0x1C</td>
<td>0x0C</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0x32</td>
<td>0x7D</td>
<td>0x6D</td>
<td>0x5D</td>
<td>0x4D</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0x50</td>
<td>0x33</td>
<td>0x23</td>
<td>0x13</td>
<td>0x03</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0x50</td>
<td>0x44</td>
<td>0x34</td>
<td>0x24</td>
<td>0x14</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0x43</td>
<td>0x55</td>
<td>0x65</td>
<td>0x75</td>
<td>0x85</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0x66</td>
<td>0x66</td>
<td>0x76</td>
<td>0x86</td>
<td>0x96</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Way 1</th>
<th></th>
<th>V</th>
<th>D</th>
<th>Tag</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0x33</td>
<td>0x80</td>
<td>0x81</td>
<td>0x82</td>
<td>0x83</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0x33</td>
<td>0xB4</td>
<td>0xB5</td>
<td>0xB6</td>
<td>0xB7</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0x95</td>
<td>0xC3</td>
<td>0xC2</td>
<td>0xC1</td>
<td>0xC0</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0x95</td>
<td>0xD3</td>
<td>0xD4</td>
<td>0xD5</td>
<td>0xD6</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0x22</td>
<td>0x89</td>
<td>0x88</td>
<td>0x87</td>
<td>0x86</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0xA0</td>
<td>0x92</td>
<td>0x93</td>
<td>0x94</td>
<td>0x95</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0x37</td>
<td>0xF5</td>
<td>0xF6</td>
<td>0xF7</td>
<td>0xF8</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0x18</td>
<td>0xA7</td>
<td>0xA8</td>
<td>0xA9</td>
<td>0xAA</td>
</tr>
</tbody>
</table>

(C) (8 points) Would the following memory accesses result in a hit or a miss? If it results in a hit, specify what value is returned; if it is a miss, explain why in a few words or by showing your work.

32-Bit Byte Address: 0x4AB4

- **32-Bit Byte Address: 0x4AB4**

  **(label 7C_1) Line index: __3___**

  **(label 7C_2) Tag: 0x__95____**

  **(label 7C_3) Block offset: __1___**

  **(label 7C_4) Returned value if hit / Explanation if miss: _______________0xD4_______**

32-Bit Byte Address: 0x21E0

  **(label 7C_5) Line index: __6___**

  **(label 7C_6) Tag: 0x__43____**

  **(label 7C_7) Block offset: __0___**

  **(label 7C_8) Returned value if hit / Explanation if miss: __miss, valid bit is 0_______**
Dilvina and Saniel want to analyze the performance of this same cache on the following assembly program, which averages the quiz 2 scores for the 225 students in 4.006. The students' scores are stored in an array A whose base address is at 0x3000.

```assembly
// Assume the following registers are initialized:
// x1 = 0 (loop index)
// x2 = 225 (number of 4.006 students)
// x3 = 0x3000 (base address of array A)
// x6 = 0 (used for summing)

. = 0x0 // The following code starts at address 0x0
sum_loop:
    slli x4, x1, 2 // x4 = 4*i
    add x5, x4, x3 // x5 = address of A[i]
    lw x5, 0(x5) // x5 = A[i]
    add x6, x6, x5 // x6 = sum(A[i:0])
    addi x1, x1, 1 // increment i
    blt x1, x2, sum_loop // continue looping
divide_by_n:
    // divide by 225 here
```

Answer the following questions about the behavior of the cache during execution of the above code. Assume that the cache uses a least recently used (LRU) replacement policy, that the cache is initially empty, and that all cache lines in Way 0 are currently the least-recently used.

(D) (1 point) How many instruction fetches and data accesses occur per iteration of the loop?

(label 7D_1) Number of instruction fetches per loop iteration: ___6____

(label 7D_2) Number of data accesses per loop iteration: ___1____

(E) (2 points) What is the hit ratio for all memory accesses (both instruction fetches and data accesses) the first time through the loop?
First instruction misses, next 3 hit because block size is 4. Fifth instruction misses, sixth instruction hits. Data access is a miss.

(label 7E) First loop iteration hit ratio: ___4/7____

(F) (2 points) After the program has been running for many loop iterations, what is the steady-state hit ratio for all memory accesses (both instruction fetches and data accesses)?
All instructions hit. Data access misses once every 4 iterations. So 1 miss per for iterations or 27/28 hit rate.

(label 7F) Steady-state hit ratio: ___27/28____

(G) (2 points) Dilvina and Saniel want to use the cache above in their memory hierarchy between the CPU and main memory. The cache takes 5 cycles to determine if a memory access is a miss or a hit, and the main memory in their system takes an additional 140 cycles for each memory access that reaches it. What is the steady-state AMAT of their memory system during the test that they ran above?
AMAT = hit_time + (1-HR)(miss penalty)
    = 5 + (1/28)(140) =

(label 7G) AMAT: ___10_______

END OF QUIZ 2!