Problem 1. Integrating a processor with a data cache

In this problem, we will consider a program executing on a processor connected to a data cache. The processor is a 5-stage pipelined TinyRV1 processor with **full bypassing** to resolve data hazards. The data cache is a **two-line direct-mapped** cache with 8B cache lines.

The program is calculating accumulated sum of an array.

```
int accu_sum( int src[], int size ) {
  int sum = 0;
  for ( int i = 0; i < size; i++ ) {
    sum += src[i]
    src[i] = sum;
  }
  return sum;
}
```

The simplified assembly instructions which correspond to the loop above are as shown below. Study the assembly instructions and make sure that you understand how it corresponds to the C++ program above. **For simplicity, assume the following**:

- `x1` : holds the accumulated sum of the `src[]` array
- `x2` : holds the base address of the `src[]` array
- `x3` : holds the size or loop count

```
loop:
 lw    x4, 0(x2)
 add   x1, x1, x4
 sw    x1, 0(x2)
 addi  x2, x2,  4
 addi  x3, x3, -1
 bne   x3, x0, loop
 opA
 opB
 ...
```

Part 1.A Cache Behavior

Assume the data cache is empty initially. We also assume the base address of the `src` array stored in `x2` is `0x2000`. The table below shows the first few dynamic memory transations. Fill in the table. What's the cache miss rate of this program? Can you think of any way to reduce cache miss rate? Will larger cache help?

				Set	
	tag	**idx**	**hit/miss**	**0**	**1**
`rd 0x2000`					
`wr 0x2000`					
`rd 0x2004`					
`wr 0x2004`					
`rd 0x2008`					
`wr 0x2008`					
`rd 0x200c`					
`wr 0x200c`					
`rd 0x2010`					
`wr 0x2010`					
`rd 0x2014`					
`wr 0x2014`					
`rd 0x2018`					

Part 1.B Pipeline Diagram Illustrating Cache Behavior

Now let's try to draw the pipeline diagram of the first two iterations of the loop. We also include the first instruction of the third iteration as shown in the instruction sequence.

Assume the miss penalty is two cycles. So on a cache miss, the processor stalls for **two extra** cycles in the M stage. Note that the cache is not pipelined, so please be aware of potential structual hazards in the M stage.

Draw arrows in your pipeline diagram to indicate any data or control hazards. Ignoring the startup overhead, what's the CPI of the first iteration? What's the CPI of the second iteration? What's the overall CPI if loop size is 64? If cache lines are 16B instead of 8B, what's the overall CPI?

lw x4, 0(x2)																																		
add x1, x1, x4																																		
sw x1, 0(x2)																																		
addi x2, x2, 4																																		
addi x3, x3, -1																																		
bne x3, x0, loop																																		
opA																																		
opB																																		
lw x4, 0(x2)																																		
add x1, x1, x4																																		
sw x1, 0(x2)																																		
addi x2, x2, 4																																		
addi x3, x3, -1																																		
bne x3, x0, loop																																		
opA																																		
opB																																		
lw x4, 0(x2)																																		