Problem 4.3 (16 points)

Consider the following DLX code fragment to be executed on a simple 5-stage *pipelined* machine with separate instruction and data memories.

```
       ADD    R1,R0,R0      ; init counter to 0
Loop:  LW     R2,100(R1)    ; R2 is array element,
                            ; R1 has offset of highest unused array element
       LW     R3,500(R1)    ; R3 holds array element
       SUBI   R5,R3,R2      ; perform subtraction
       ADD    R5,R5,R4      ; perform addition of a constant
       SW     1000(R1),R5   ; store the result
       ADDI   R1,R1,#4      ; increment pointer
       SUBI   R5,R1,#400    ; check pointer
       BNEZ   R5,Loop       ; branch while not done
       STOP
```

Problem 4.3.1

Compute the CPI for one iteration of the loop, ignoring the BNEZ command in your analysis. Use this to estimate the total CPI for the loop again, ignoring the contribution of the branch.

Answer:

Mapping the cycles for the first loop, we get:

	Cycle										
Command	1	2	3	4	5	6	7	8	9	10	11
`ADD R1,R0,R0`	F	D	X	M	W						
`LW R2,100(R1)`		F	s	s	D	X	M	W			
`LW R3,500(R1)`					F	D	X	M	W		
`SUB R5,R3,R2`						F	s	s	D	X	M
`ADD R5,R5,R4`									F	s	s

	Cycle										
Command	12	13	14	15	16	17	18	19	20	21	22
`SUB R5,R3,R2`	W										
`ADD R5,R5,R4`	D	X	M	W							
`SW 1000(R1),R5`	F	s	s	D	X	M	W				
`ADDI R1,R1,#4`				F	D	X	M	W			
`SUBI R5,R1,#400`					F	s	s	D	X	M	W

The first loop begins at cycle 2 and completes after cycle 22, for a total of **21 cycles**. There are 7 instructions in the loop (excluding the branch), so the total CPI for the first loop is 3.

$\frac{21 cycles}{7 instructions} = 3$

Looking at this, one notices that the first instruction in the loop has two stalls while it waits for R1 to be read from memory. Subsequent executions of the loop do not need to stall for R1, so the second iteration will execute in **19 cycles**, giving a CPI of 2.71.

$\frac{19 cycles}{7 instructions} = 2.71$