Implement loop unrolling in RISC-V using Ripes:

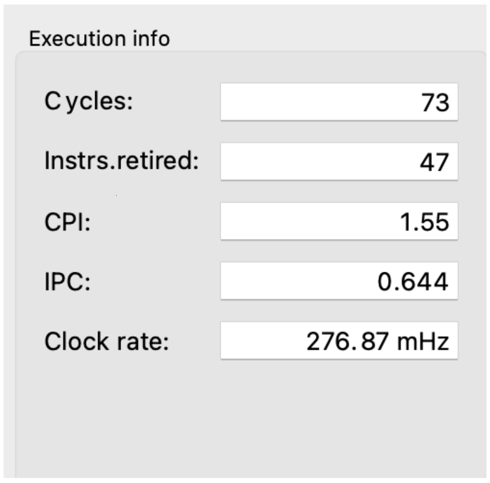
Problem Statement: Sum of all the elements in an array

1. Without Unrolling:

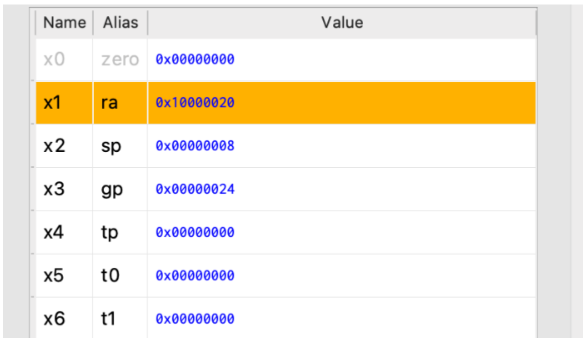
Code:

```
# RISC-V Assembly code without loop unrolling for 5-stage pipeline
.data
array:  .word 1, 2, 3, 4, 5, 6, 7, 8
size:  .word 8
sum:   .word 0

.text
  # Load address of array into x1
  la x1, array

  # Load size of array into x4
  lw x4, size

  # Initialize sum to 0
  li x3, 0

  # Loop to calculate sum
  loop:
    lw x2, 0(x1)        # Load array element
    add x3, x3, x2       # Add to sum
    addi x1, x1, 4       # Move to next element
    addi x4, x4, -1      # Decrement counter
    bne x4,x0,loop       # loop condition
    sw x3,0(x1)         #sum
    nop
```

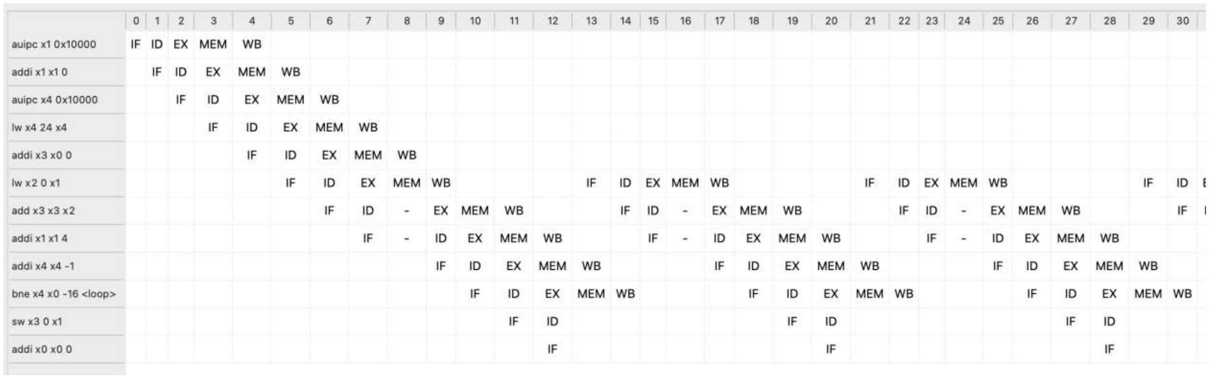
Output:


Execution info

Cycles:	73
Instrs.retired:	47
CPI:	1.55
IPC:	0.644
Clock rate:	276.87 mHz



Name	Alias	Value
x0	zero	0x00000000
x1	ra	0x10000020
x2	sp	0x00000008
x3	gp	0x00000024
x4	tp	0x00000000
x5	t0	0x00000000
x6	t1	0x00000000



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
auipc x1 0x10000	IF	ID	EX	MEM	WB																										
addi x1 x1 0		IF	ID	EX	MEM	WB																									
auipc x4 0x10000			IF	ID	EX	MEM	WB																								
lw x4 24 x4				IF	ID	EX	MEM	WB																							
addi x3 x0 0					IF	ID	EX	MEM	WB																						
lw x2 0 x1						IF	ID	EX	MEM	WB				IF	ID	EX	MEM	WB				IF	ID	EX	MEM	WB				IF	ID
add x3 x3 x2							IF	ID	-	EX	MEM	WB			IF	ID	-	EX	MEM	WB			IF	ID	-	EX	MEM	WB			IF
addi x1 x1 4								IF	-	ID	EX	MEM	WB			IF	-	ID	EX	MEM	WB			IF	-	ID	EX	MEM	WB		
addi x4 x4 -1										IF	ID	EX	MEM	WB				IF	ID	EX	MEM	WB				IF	ID	EX	MEM	WB	
bne x4 x0 -16 <loop>											IF	ID	EX	MEM	WB				IF	ID	EX	MEM	WB				IF	ID	EX	MEM	WB
sw x3 0 x1												IF	ID							IF	ID							IF	ID		
addi x0 x0 0													IF								IF								IF		

* Stalls are observed

1.With Unrolling and Scheduling:

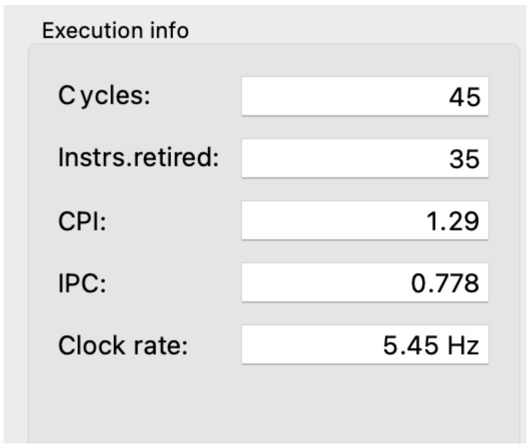
Loop unrolling is the optimization technique of replicating loop iterations to reduce loop overhead, while instruction scheduling reorders instructions to minimize stalls and maximize resource utilization in assembly programs.

Code:

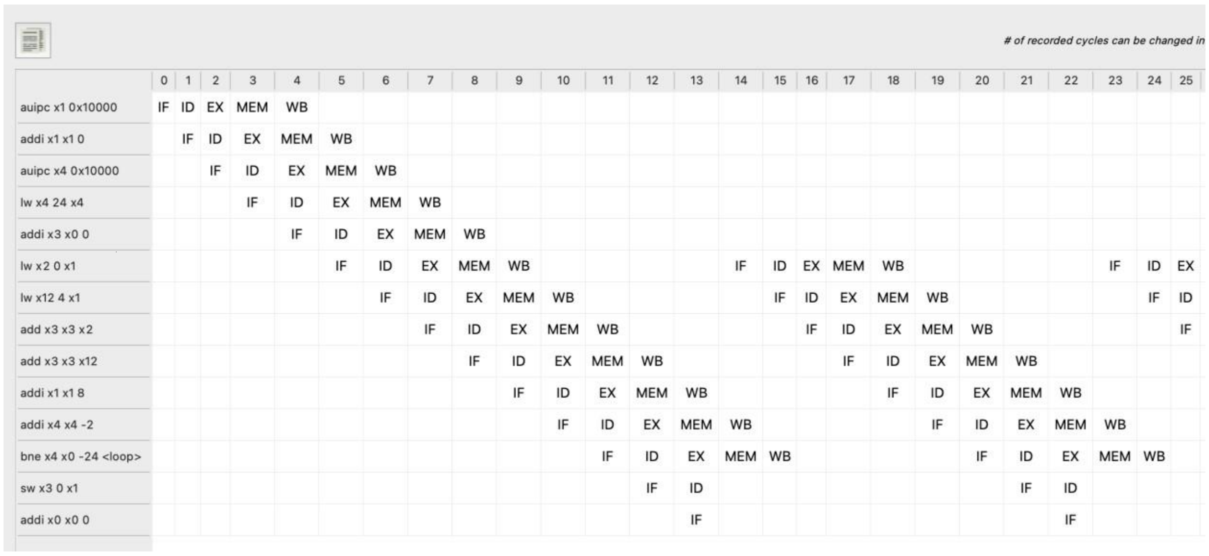
```
# RISC-V Assembly code with loop unrolling for 5-stage pipeline

.data
array:  .word 1, 2, 3, 4, 5, 6, 7, 8
size:  .word 8
sum:   .word 0

.text
   # Load address of array into x1
   la x1, array

   # Load size of array into x4
```

```
lw x4, size

# Initialize sum to 0
li x3, 0

# Loop Unrolling
loop:

   # Load array element and add to sum (unrolled iteration 1)
   # Load next array element and add to sum (unrolled iteration 2)

   lw x2, 0(x1)        # Load array element
   lw x12, 4(x1)        # Load array element

   add x3, x3, x2       # Add to sum
   add x3, x3, x12       # Add to sum

   addi x1, x1, 8       # Move to next element
   #addi x1, x1, 4       # Move to next element

   # Decrement counter by 2
   addi x4, x4, -2      # Decrement counter
   bne x4,x0,loop       # loop condition


   # Store the final sum in memory
   sw x3,0(x1)        #sum
   nop
```

Output:



```
Execution info
Cycles:              45
Instrs.retired:      35
CPI:               1.29
IPC:              0.778
Clock rate:     5.45 Hz
```



# of recorded cycles can be changed in

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
auipc x1 0x10000	IF	ID	EX	MEM	WB																					
addi x1 x1 0		IF	ID	EX	MEM	WB																				
auipc x4 0x10000			IF	ID	EX	MEM	WB																			
lw x4 24 x4				IF	ID	EX	MEM	WB																		
addi x3 x0 0					IF	ID	EX	MEM	WB																	
lw x2 0 x1						IF	ID	EX	MEM	WB					IF	ID	EX	MEM	WB					IF	ID	EX
lw x12 4 x1							IF	ID	EX	MEM	WB					IF	ID	EX	MEM	WB					IF	ID
add x3 x3 x2								IF	ID	EX	MEM	WB					IF	ID	EX	MEM	WB					IF
add x3 x3 x12									IF	ID	EX	MEM	WB					IF	ID	EX	MEM	WB				
addi x1 x1 8										IF	ID	EX	MEM	WB					IF	ID	EX	MEM	WB			
addi x4 x4 -2											IF	ID	EX	MEM	WB					IF	ID	EX	MEM	WB		
bne x4 x0 -24 <loop>												IF	ID	EX	MEM	WB					IF	ID	EX	MEM	WB	
sw x3 0 x1													IF	ID								IF	ID			
addi x0 x0 0														IF									IF			

*No stalls are observed

Inference:

In the absence of loop unrolling and instruction scheduling, stalls occurred during code execution, resulting in higher CPI and increased cycle count compared to optimized versions. However, by implementing loop unrolling and instruction scheduling techniques, all stalls were effectively eliminated, leading to enhanced code optimization and more efficient execution. This underscores the importance of employing these optimization strategies to minimize stalls, decrease CPI, and reduce cycle count, ultimately improving overall code performance and execution efficiency.