13) CPI = cycles / instr

```
void Hadamard(int *A, int *B, int N){
    for(i=0; i<N; i++){
        A[i] = A[i] * B[i];
    }
}
```

```
7  HadamardNaive:
8          add     t0, zero, zero  # i <- 0
9  loop:   bge     t0, a2, done    # if i >= N goto done
10         slli    t1, t0, 2       # t1 <- 4*i
11         add     t2, a0, t1      # t2 <- Address(A[i])
12         lw      t3, 0(t2)       # t3 <- A[i]
13         slli    t4, t0, 2       # t4 <- 4*i
14         add     t5, a1, t4      # t5 <- Address(B[i])
15         lw      t6, 0(t5)       # t6 <- B[i]
16         mul     t7, t3*t6       # t7 <- A[i]*B[i]
17         slli    t8, t0, 2       # t8 <- 4*i
18         add     t9, a0, t8      # t9 <- Address(B[i])
19         sw      t7, 0(t9)       # A[i] <- A[i]*B[i]
20         addi    t0, t0, 1       # i <- i+1
21         jal     zero, loop
22 done:   jalr zero, ra, 0
```

(a) Naive Version

31 = 2 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 8 + 1 + 4 + 1 + 1 + 2

```
24 HadamardOpt:
25         slli    t0, a2, 2       # t0 <- 4*N
26         add     t1, a0, t0      # t1 <- Address(A[N])
27         bge     zero, a2, done  # if(0<=N) done
28 loop:   lw      t2, 0(a0)       # t2 <- A[i]
29         lw      t3, 0(a1)       # t3 <- B[i]
30         mul     t4, t2, t3      # t4 <- A[i]*B[i]
31         sw      t4, 0(a0)       # A[i] <- A[i]*B[i]
32         addi    a0, a0, 4       # pA++
33         addi    a1, a1, 4       # pB++
34         blt     a0, t1, loop    # if pA < Address(A[N])
35 done:   jalr zero, ra, 0
```

(b) Optimized Version

24 = 4 + 1 + 8 + 4 + 1 + 1 + 2

$CPI = \frac{31 \text{ cycles}}{13 \text{ instr}}$

≈ 2.384

≈ 2.38

$CPI = \frac{24}{7}$

≈ 3.428

≈ 3.43

15) $\text{Cycles}_{\text{Naive}} > \text{Cycles}_{\text{Optimized}}$

→ $\text{Naive} = 31 \frac{\text{cycles}}{\text{iteration}}$

$\text{Optimized} = 24 \frac{\text{cycles}}{\text{iteration}}$

→ So, optimized is faster since it takes less cycles for 1 iteration than naive.

→ $\frac{31}{24} \approx 1.291$

$\approx (1.29)\times$ faster