```
# OrderRewrite_0
# arguments:
#    $a0: order
#    $a1: base
#    $a2: output
#
OrderRewrite_0:
    add     $t0, $0, $0         # i <- 0
    addi    $t1, $0, -1         # t1 <- sentinel
next:
    sll     $t2, $t0, 2         # t2 <- 4*i
    add     $t3, $a0, $t2       # t3 <- &(order[i])
    lw      $t4, 0($t3)         # t4 <- order[i]
    beq     $t4, $t1, done      # if order[i] == sentinel
    sll     $t5, $t4, 2          # t5 <- 4*order[i]
    add     $t6, $a1, $t5       # t6 <- &(base[order[i]])
    lw      $t7, 0($t6)         # t7 <- base[order[i]]
    add     $t5, $a2, $t2       # t5 <- &(base[i])
    sw      $t7, 0($t5)         # output <- base[order[i]]
    addi    $t0, $t0, 1         # i <- i+1
    j       next
done:
    jr      $ra
```

Figure 1: `OrderRewrite` — Optimization level 0.

Question 3 (20 points): The new MIPS processor also has store instructions with the same addressing mode and similar assembly syntax as the load instructions, in particular it has a store word indexed instruction `swi`. In this question we will study the performance of the subroutine `OrderRewrite` that receives three arguments that are the base address of three vectors: `order`, `base`, and `output`. The vector `order` contains a list of positions of the vector `base`. A sentinel value of `-1` signals the end of this list. `OrderRewrite` writes into the vector `output` the values found in the vector `base` in the order indicated by the vector `order`. Assume that the sentinel value `-1` appears in position N of the vector `order`. Whenever necessary, provide answers to the questions below in terms of N.

You will analyze two versions of `OrderRewrite`. The first version, called `OrderRewrite_0`, shown in Figure 1, was produced by a basic compiler that is unaware of the existence of the `lwi` and `swi` instructions. This compiler also does not perform sophisticated optimizations. The second version, `OrderRewrite_3`, shown in Figure 2, was produced by a newer optimizing compiler that is able to use `lwi` and `swi`. In this version the compiler also restructured the code to avoid the additional jump instruction at the end of the loop.

The average number of clock cycles needed to execute each of the instructions used in the two versions of `OrderRewrite` is given in Table 1.

a. (**4 points**) Complete the table below. To compute the CPI, assume that N is very large:

```
# OrderRewrite_3
# arguments:
#     $a0: order
#     $a1: base
#     $a2: output
#
OrderRewrite_3:
    addi    $t1, $0, -1         # tm <- sentinel
    add     $t0, $0, $0         # index <- 0
    lw      $t2, 0($a0)         # t2 <- order[0]
    beq     $t2, $t1, done      # if order[0] == sentinel
next:
    sll     $t3, $t2, 2         # t3 <- 4*(order[i])
    lwi     $t4, ($t3,0)($a1)   # t4 <- base[order[i]]
    swi     $t4, ($t0,0)($a2)   # output <- base[order[i]]
    addi    $t0, $t0, 4         # index <- index + 4
    lwi     $t2, ($t0,0)($a0)   # t2 <- order[i]
    bne     $t2, $t1, next      # if order[i] == sentinel
done:
    jr      $ra
```

Figure 2: `OrderRewrite` — Optimization level 3.

Instruction	Cycles	Instruction	Cycles	Instruction	Cycles
`add`	1	`lw`	4	`beq`	3
`addi`	1	`sw`	4	`bne`	3
`sll`	1	`swi`	4	`j`	2
`sub`	1	`lwi`	4	`jr`	2

Table 1: Average number of clocks to execute instructions of each type in both versions of the MIPS processor.

Subroutine	Number of Instructions Executed	Average Cycles per Instruction (CPI)
`OrderRewrite_0`	$11 \times N + 7$	2.09
`OrderRewrite_3`	$6 \times N + 5$	2.83

For number of instructions executed, we trace the code for each subroutine and find out how many instructions are executed for each value of N:

N	`OrderRewrite_0`	`OrderRewrite_3`
0	$2 + 4 + 1 = 7$	$4 + 1 = 5$
1	$2 + 11 + 4 + 1 = 18$	$4 + 6 + 1 = 11$
2	$2 + 22 + 4 + 1 = 27$	$4 + 12 + 1 = 17$
3	$2 + 33 + 4 + 1 = 40$	$4 + 18 + 1 = 23$
N	$2 + 11 \times N + 4 + 1 = 11 \times N + 7$	$4 + 6 \times N + 1 = 6 \times N + 5$

If N is very large, the only instructions that will matter are the instructions executed inside the loop. Therefore:

$$\begin{aligned}
\#\text{ of clock cycles}_0 &= (1+1+4+3+1+1+1+4+1+4+1+2)\times N = 23\times N \text{ cycles} \\
\text{CPI}_0 &= \frac{\#\text{ of clock cycles}_0}{\#\text{ of instructions}_0} = \frac{23\times N}{11\times N} = 2.09\frac{\text{cycles}}{\text{instruction}} \\
\#\text{ of clock cycles}_3 &= (1+4+4+1+4+3)\times N = 17\times N \text{ cycles} \\
\text{CPI}_3 &= \frac{\#\text{ of clock cycles}_3}{\#\text{ of instructions}_3} = \frac{17\times N}{6\times N} = 2.83\frac{\text{cycles}}{\text{instruction}}
\end{aligned}$$

b. (**4 points**) If both `OrderRewrite_0` and `OrderRewrite_3` are executed in the same processor and N is very large, which version is faster and by how much? Express your answer as a speedup (your answer should be in the form "`A` is k times faster than `B`").

Because the processor is the same, the clock frequency is the same. We can solve this question simply taking the ratio between the number of clock cycles executed inside the loop by each version:

$$\text{Speedup} = \frac{\text{Performance}_3}{\text{Performance}_0} = \frac{\#\text{ of cycles}_0}{\#\text{ of cycles}_3} = \frac{23\times N}{17\times N} = 1.35$$

Therefore `OrderRewrite_3` is 1.35 times faster than `OrderRewrite_0`.

c. (**4 points**) For a given execution of `OrderRewrite_3` $N = 1,000,000$ and it takes 9.375 milliseconds to execute the subroutine (1 millisecond = 0.001 second). What is the frequency of execution of this processor expressed in GHz (1 GHz = 10^9 Hz).

$$\begin{aligned}
\text{Execution Time}_3 &= \frac{\#\text{ of instructions}_3 \times \text{CPI}_3}{\text{Clock Frequency}} \\
\text{Clock Frequency} &= \frac{\#\text{ of instructions}_3 \times \text{CPI}_3}{\text{Execution Time}_3} \\
\text{Clock Frequency} &= \frac{6\times 10^6 \times 2.83}{9.375\times 10^{-3}} = 1.8\text{ GHz}
\end{aligned}$$

d. (**4 points**) Still assuming that both subroutines are running on the same processor, for which value of N will the execution of `OrderRewrite_0` take 9.375 milliseconds?

$$\begin{aligned}
\text{Execution Time}_0 &= \frac{\#\text{ of instructions}_0 \times \text{CPI}_0}{\text{Clock Frequency}} \\
9.375\times 10^{-3} &= \frac{11\times N\times 2.09}{1.8\times 10^9} \\
N &= \frac{9.375\times 10^{-3}\times 1.8\times 10^9}{11\times 2.09} = 734,015
\end{aligned}$$