Student Name: Süleyman Gölbol
Student Number: 1801042656

CSE 331

Computer Organizations

Homework 1

Due Date 04/11/2022 Friday 17:00

1. A compiler designer wants to improve the performance of a machine for one specific program. The program has the following properties:

	R-type ($x10^6$)	I-Type ($x10^6$)	J-Type ($x10^6$)
Program instructions	50	30	20

	R-type	I-Type	J-Type
Required Cycles	2	4	3

Assume you can improve only one type with 50%. Which type do you prefer for improvement and how many times can you improve the whole program in the end?

Solution:

Solution 1:

Performance is inversely proportional with clock cycle.

Inverse Proportion Formula:

$$\frac{Execution\ Time_1}{Execution\ Time_2} = \frac{Required\ Cycle_1}{Required\ Cycle_2} = \frac{Performance_2}{Performance_1}$$

So, $\frac{4}{Required\ Cycle_2} = \frac{150}{100}$, which means new required cycle is **2.667**.

Because of total cycles of I type is larger than others, improving it 50% makes bigger difference.

$$Total\ Cycles = 2 \cdot 50 \cdot 10^6 + 2.667 \cdot 30 \cdot 10^6 + 3 \cdot 20 \cdot 10^6 = 240 \cdot 10^6$$

At the end, better improvement we get on whole program is;

$\frac{280 \cdot 10^6}{240 \cdot 10^6} = 1.16$ times improvement on whole program.

Solution 2:

Amdahl's Law screenshot is from class presentations.

Execution time after improvement =

$$\frac{\text{Execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected}$$

$$x = \frac{(280-160)}{\%50} + 160 \cdot 10^6 = \frac{(280-160) \cdot 10^6}{\frac{3}{2}} + 160 \cdot 10^6$$

$x = 240 \cdot 10^6$ (Execution time after improvement).

Because of improvement is inversely proportional with execution time,

$\frac{280 \cdot 10^6}{240 \cdot 10^6} = 1.16$ times improvement on whole program.

2. In this part you will write an assembly program on MARS for finding and printing all divisible sum pairs as explained below:

Given an array of integers and a positive integer k, determine the number of (i, j) pairs where $i < j$ and $ar[i]$ + $ar[j]$ is divisible by k.

Example

$ar = [1, 2, 3, 4, 5, 6]$
$k = 5$

Three pairs meet the criteria: $[1, 4]$, $[2, 3]$, and $[4, 6]$.

Function Description

Complete the divisibleSumPairs function in the editor below.

divisibleSumPairs has the following parameter(s):

- int n: the length of array ar
- int ar[n]: an array of integers
- int k: the integer divisor

Returns

- int: the number of pairs

Input Format

The first line contains 2 space-separated integers, n and k.
The second line contains n space-separated integers, each a value of $arr[i]$.

Constraints

- $2 \leq n \leq 100$
- $1 \leq k \leq 100$
- $1 \leq ar[i] \leq 100$

Sample Input

```
STDIN          Function
-----          --------
6 3            n = 6, k = 3
1 3 2 6 1 2     ar = [1, 3, 2, 6, 1, 2]
```

Explanation

Here are the 5 valid pairs when $k = 3$:

- $(0, 2) \rightarrow ar[0] + ar[2] = 1 + 2 = 3$
- $(0, 5) \rightarrow ar[0] + ar[5] = 1 + 2 = 3$
- $(1, 3) \rightarrow ar[1] + ar[3] = 3 + 6 = 9$
- $(2, 4) \rightarrow ar[2] + ar[4] = 2 + 1 = 3$
- $(4, 5) \rightarrow ar[4] + ar[5] = 1 + 2 = 3$

Solution Report:

1801042656_hw1.asm file is my solution file. Screenshot in right is terminal output. All inputs should be taken in different lines. (Separate them by enter button)
First it takes n, then k, then array elements.
After calculation, it prints total number of pairs.

```
Enter size of array (n): 3
Enter k: 2

n is : 3, k is : 2
Enter array elements (seperated by enter): 3
2
1
Array elements are:
ar[0] + ar[2] = 4
Total Number of Pairs: 1
```