Homework of Chapter 1

Exercise 1.4

Problem

1.4 As an alternative to the approach outlined in the preceding problem, we can use C's bitwise operators to implement the tree-structured global sum. In order to see how this works, it helps to write down the binary (base 2) representation of each of the core ranks, and note the pairings during each stage:

	Stages		
Cores	*1*	*2*	*3*
$0_{10} = 000_2$	$1_{10} = 001_2$	$2_{10} = 010_2$	$4_{10} = 100_2$
$1_{10} = 001_2$	$0_{10} = 000_2$	×	×
$2_{10} = 010_2$	$3_{10} = 011_2$	$0_{10} = 000_2$	×
$3_{10} = 011_2$	$2_{10} = 010_2$	×	×
$4_{10} = 100_2$	$5_{10} = 101_2$	$6_{10} = 110_2$	$0_{10} = 000_2$
$5_{10} = 101_2$	$4_{10} = 100_2$	×	×
$6_{10} = 110_2$	$7_{10} = 111_2$	$4_{10} = 100_2$	×
$7_{10} = 111_2$	$6_{10} = 110_2$	×	×

From the table we see that during the first stage each core is paired with the core whose rank differs in the rightmost or first bit. During the second stage cores that continue are paired with the core whose rank differs in the second bit, and during the third stage cores are paired with the core whose rank differs in the third bit. Thus, if we have a binary value `bitmask` that is 001_2 for the first stage, 010_2 for the second, and 100_2 for the third, we can get the rank of the core we're paired with by "inverting" the bit in our rank that is nonzero in `bitmask`. This can be done using the bitwise exclusive or $\wedge$ operator.

Implement this algorithm in pseudo-code using the bitwise exclusive or and the left-shift operator.

Answer

Pseudo Code

```
int bitmask = 1;
int total_cores = bin(111);
int my_rank;
int my_value;

while (bitmask < total_cores){
    int partener = my_rank ^ bitmask;
    if (my_rank % bitmask != 0){
        send(my_value,  partener);
```

```
            return;
        }
        my_value += receive(partener);
        bitmask << 1;
    }
```

Exercise 1.6

Problem

1.6 Derive formulas for the number of receives and additions that core 0 carries out using
- **a.** the original pseudo-code for a global sum, and
- **b.** the tree-structured global sum.

Make a table showing the numbers of receives and additions carried out by core 0 when the two sums are used with $2, 4, 8, \ldots, 1024$ cores.

Answer

Suppose we have p processors.

If we use the original global sum, it's obviously that the core 0 will receive values sent by each core. So core 0 will receive p - 1 and operate p - 1 times additions.

And if we use tree structure, other cores will do the same things as core 0 at the same time. Let's consider a structure of a perfect binary tree with p child node at the last layer. The number of layer of this tree represents the times that core 0 will operate. So core 0 will receive only $\log_2 p$ and operate $\log_2 p$ times additions.

So we can conclude a table below:

TYPE	NUM OF RECEIVES	NUM OF ADDITIONS
Original	p - 1	p - 1
Tree Structure	$\log_2 p$	$\log_2 p$

ORIGINAL	NUM OF RECEIVES	NUM OF ADDITIONS
2	1	1
4	3	3
8	7	7
16	15	15

ORIGINAL	NUM OF RECEIVES	NUM OF ADDITIONS
32	31	31
64	63	63
128	127	127
256	255	255
512	511	511
1024	1023	1023

TREE STRUCTURE	NUM OF RECEIVES	NUM OF ADDITIONS
2	1	1
4	2	2
8	3	3
16	4	4
32	5	5
64	6	6
128	7	7
256	8	8
512	9	9
1024	10	10