

ICS Homework 7

April 7, 2021

1 Organization

1.1

Consider the following function to copy the contents of one array to another:

```
1 void copy_array(long *src, long *dest, long n) {  
2     long i;  
3     for (i = 0; i < n; i++)  
4         dest[i] = src[i];  
5 }
```

Suppose `a` is an array of length 1000 initialized so that each element `a[i]` equals `i`.

1. What would the array become if call `copy_array(a+1, a, 999)`?
It will set each element `a[i]` to `i+1`, for $0 \leq i \leq 998$.
2. What would the array become if call `copy_array(a, a+1, 999)`?
It will set each element `a[i]` to 0, for $1 \leq i \leq 999$.
3. Our performance measurements indicate that the call in the first question has a CPE of 1.2, while the call in the second question has a CPE of 5.0. To what factor do you attribute this performance difference?
In the second case, the load of one iteration depends on the result of the store from the previous iteration. Thus, there is a write/read dependency between successive iterations.
4. What performance (CPE) would you expect for the call `copy_array(a, a, 999)`? Please explain your answer.
It will give a CPE of 1.2, the same as for the question 1, since there are no dependencies between stores and subsequent loads.

1.2

The assembly code generated for the compiled loop of `combine3` is shown below:

```
1 combine3:  
2 # data_t = float, OP = *  
3 # i in %rdx, data in %rax, dest in %rbp  
4 .L498:                                loop:
```


2 System Software

Non-local Jump

Consider the following program:

```
1  #include <setjump.h>
2  sigjmp_buf buf;
3  void handler(int sig) {
4      siglongjmp(buf, 1);
5  }
6  int main() {
7      if (!sigsetjmp(buf, 1)) {
8          Signal(SIGINT, handler);
9          Sio_puts("starting\n");
10     } else
11         Sio_puts("restarting\n");
12
13     while(1) {
14         Sleep(1); Sio_puts("processing...\n");
15     }
16     exit(0); /* Control never reaches here */
17 }
```

Please give an example output of the program when we launch it and press **Ctrl+C** several times.

ANS:

```
1  linux> ./a.out
2  starting
3  processing...
4  processing...
5  Ctrl+C
6  restarting
7  processing...
8  Ctrl+C
9  restarting
10 processing...
```