ICS Homework 7

April 6, 2023

1 Organization

1.1

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

```c
void copy_array(long *src, long *dest, long n) {
    long i;
    for (i = 0; i < n; i++)
        dest[i] = src[i];
}
```

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 of part a has a CPE of 1.2, while the call of part b 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.

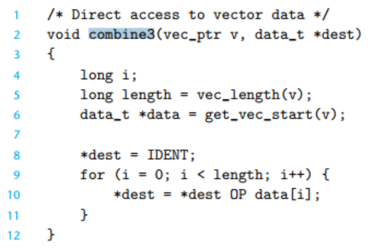
4. What performance (CPE) would you expect for the call `copy_array(a, a, 999)`? Please explain your answer.

   It will give a CPE of 1.2, the same as for part a, since there are no dependencies between stores and subsequent loads.

1.2

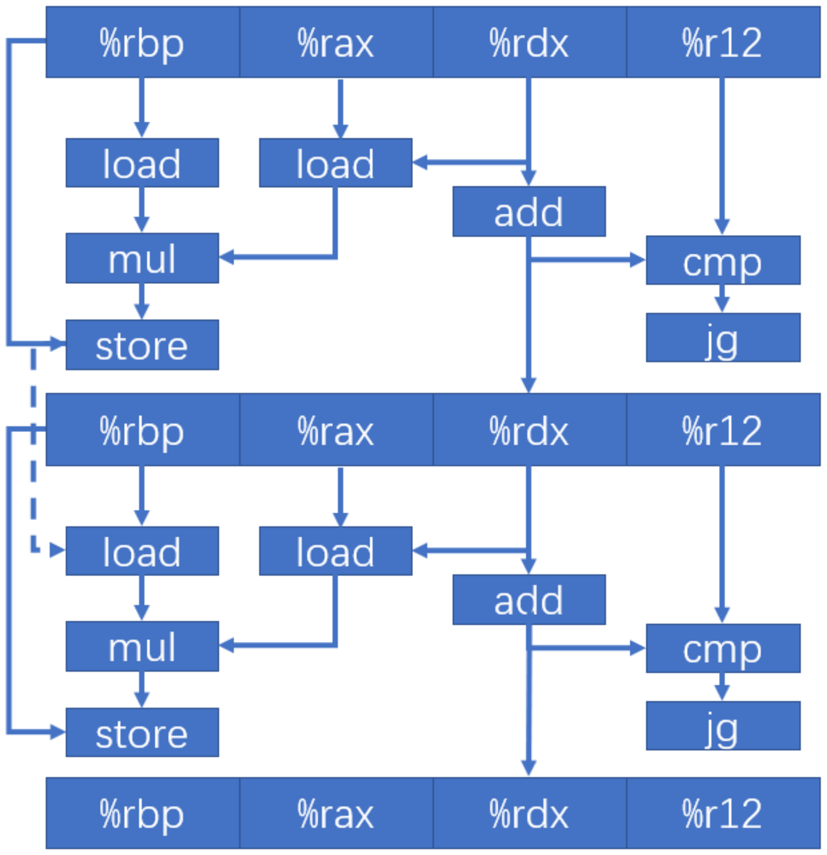
The assembly code generated for the compiled loop of combine3 is shown below:

```
combine3:
# data_t = float, OP = *
# i in %rdx, data in %rax, dest in %rbp
.L498:                          loop:
 5  movss (%rbp), %xmm0              Read from dest
 6  mulss (%rax, %rdx, 4), %xmm0     Multiply
 7  movss %xmm0, (%rbp)              Store at dest
 8  addq $1, %rdx                    Increment i
 9  cmpq %rdx, %r12                  Compare i: limit
10  jg .L498                         If >, goto loop
```



```
1   /* Direct access to vector data */
2   void combine3(vec_ptr v, data_t *dest)
3   {
4       long i;
5       long length = vec_length(v);
6       data_t *data = get_vec_start(v);
7
8       *dest = IDENT;
9       for (i = 0; i < length; i++) {
10          *dest = *dest OP data[i];
11      }
12  }
```

Illustrate the code above with data-flow graph like figure 5.14(a) or (b) in CSAPP. You can use ”store” to identify operation in line 4.



2 System Software

2.1 Signal

Recall our second problem in exe-5. Please use `sigsuspend` to fix potential bugs in the example code.

ANS: Change line 39 41 to:

```
  while(!child_exit)
    sigsuspend(&prev_one)
```

Change line 32 33 to:

```
  sigsuspend(&prev_one)
```

2.2 Non-local Jump

Consider the following program:

```
#include <setjump.h>
sigjmp_buf buf;
void handler(int sig) {
  siglongjmp(buf, 1);
}
int main() {
  if (!sigsetjmp(buf, 1)) {
    Signal(SIGINT, handler);
    Sio_puts("starting\n");
  } else
    Sio_puts("restarting\n");

  while(1) {
    Sleep(1); Sio_puts("processing...\n");
  }
  exit(0); /* Control never reaches here */
}
```

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

ANS:

```
linux> ./a.out
starting
processing...
processing...
Ctrl+C
restarting
processing...
Ctrl+C
restarting
processing...
```