Remember: The 2 separate threads have 2 separate program counters, register sets (each with its own %eax) and stack.
The 2 separate threads SHARE a common virtual memory. For this problem, the one virtual memory location you need to work with has address 0x804a018. That location implements the global static variable NNN. Both threads will execute the C code NNN = NNN + 1; concurrently.

```
%eax: garbage
mov 0x804a018,%eax
%eax: 0x804a018: 7
%eax: garbage
add $0x1, %eax
%eax: 0x804a018: 7
%eax: garbage
mov %eax, 0x804a108
%eax: 0x804a018: 0x804a018:
%eax: garbage

(This is a "good" interleaving.)
```

```
mov 0x804a018,%eax
%eax: 0x804a018: 7
%eax: garbage
add $0x1, %eax
%eax: 0x804a018: 7
%eax: garbage
mov %eax, 0x804a108
0x804a018: 0x804a018:
%eax: 0x804a018: 0x804a018:
%eax: garbage
```

```
%eax: garbage
mov 0x804a018,%eax
%eax: 0x804a018: 7
%eax: garbage
add $0x1, %eax
%eax: 0x804a018: 7
%eax: garbage
mov %eax, 0x804a108
%eax: 0x804a018: 0x804a018:
%eax: garbage
```

(This is a "bad" interleaving.)

```
mov 0x804a018,%eax
%eax: 0x804a018: 7
%eax: garbage
add $0x1, %eax
%eax: 0x804a018: 7
%eax: garbage
add $0x1, %eax
%eax: 0x804a018: 7
%eax: garbage
mov %eax, 0x804a108
%eax: 0x804a018: 0x804a018:
%eax: garbage
```

What is bad about the result below?

```
mov %eax, 0x804a108
%eax: 0x804a018: 7
%eax: garbage
```