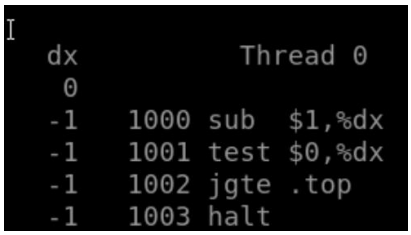
Chapter 26

1.



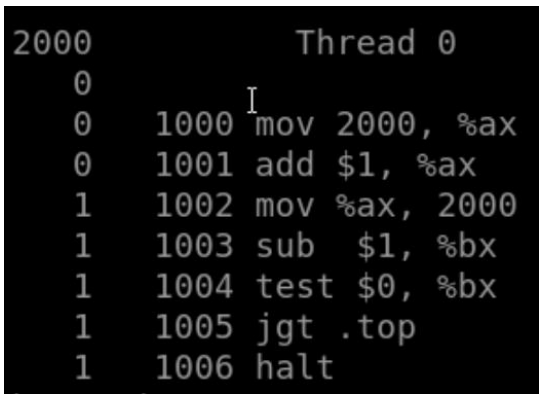
2.

No race occurs in this code, because interrupt interval is 100 instructions.

3.

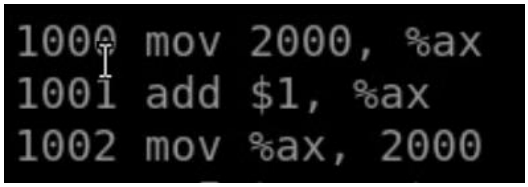
Yes, if interrupt frequency is less than 13 in this case, 1 thread will be interrupted without being halted. However this somehow doesn't cause a race condition on %dx.

4.



5. Each loop runs 3 times because of register %bx is initialized to 3.
The final value of address 2000 is 6, updated separately by 2 threads.

6. Time of interruption (4) doesn't matter.
Threads are safe when time of interruption >= 3 in this case.
The critical section is as below.

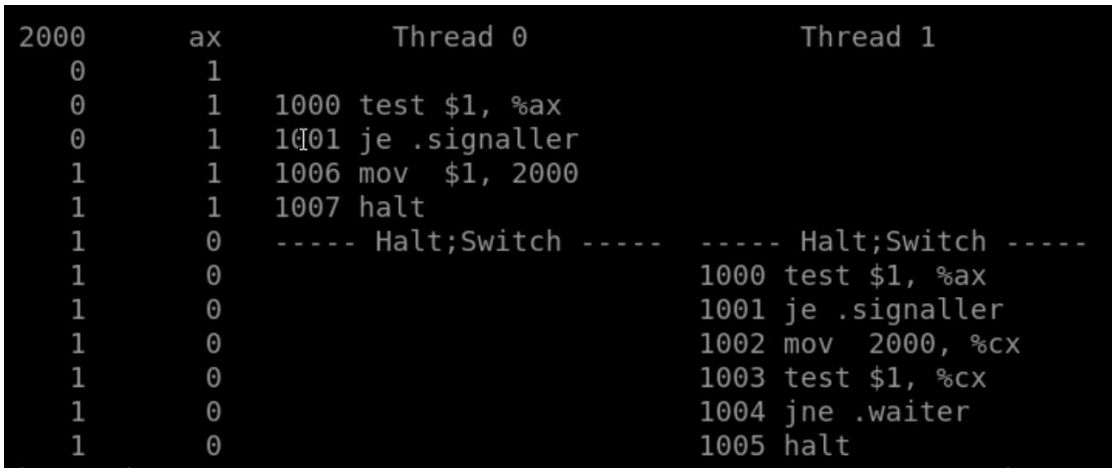


7. -i 1 gives a final value of 1.
When interrupt time >=3, the program gives correct answer (no race condition).

8.
For example, when running -a bx=5, the loop in each threads are supposed to run 5 times, and value to be 10.

When interval >=15, it is always correct.
Interval -i 3, 6, 9 or 12 is surprising, because it happens to give the correct result while it doesn't fall into the range of intervals without race conditions.

9.



First, the data of address 2000 is updated by thread 0, so it becomes 1. Then the data address 2000 is loaded on %cx.

10.
Thread 0 is entering an infinite loop, thread 1 helps thread 0 to stop by updating the data on address 2000 thus %cx.

The longer the interval the thread 0 stays longer in the infinite loop.