

Chapter 26

1.

```
I
dx      Thread 0
0
-1 1000 sub $1,%dx
-1 1001 test $0,%dx
-1 1002 jgte .top
-1 1003 halt
```

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.

```
2000      Thread 0
0
0 1000 I mov 2000, %ax
0 1001 add $1, %ax
1 1002 mov %ax, 2000
1 1003 sub $1, %bx
1 1004 test $0, %bx
1 1005 jgt .top
1 1006 halt
```

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.

```
1000 mov 2000, %ax
1001 add $1, %ax
1002 mov %ax, 2000
```

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.

2000	ax	Thread 0	Thread 1
0	1		
0	1	1000 test \$1, %ax	
0	1	1001 je .signaller	
1	1	1006 mov \$1, 2000	
1	1	1007 halt	
1	0	----- Halt;Switch -----	----- Halt;Switch -----
1	0		1000 test \$1, %ax
1	0		1001 je .signaller
1	0		1002 mov 2000, %cx
1	0		1003 test \$1, %cx
1	0		1004 jne .waiter
1	0		1005 halt

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.