Concurrency

x86.py

① ./x86.py -p loop.s -t 1 -i 100 -R dx

No value given for dx register, so its zero.
After the first run, value becomes "-1" and halts.

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

② ./x86.py -p loop.s -t 2 -i 100 -a dx=3, dx=3 -R dx.

Thread 0 | Thread 1
dx = 3 | dx = 3

Presence multiple threads won't change anything here because

"interrupt switch" is set to 100, the no. of instructions for 1 value of loop is 3

dx	
3	
2	sub $1, %dx
2	test $0, %dx
2	jgte .top
1	
0	
-1	
	halt

13 instructions < 100 (interrupt interval)

Thread 1

same (since dx=3 for thread 1 too)

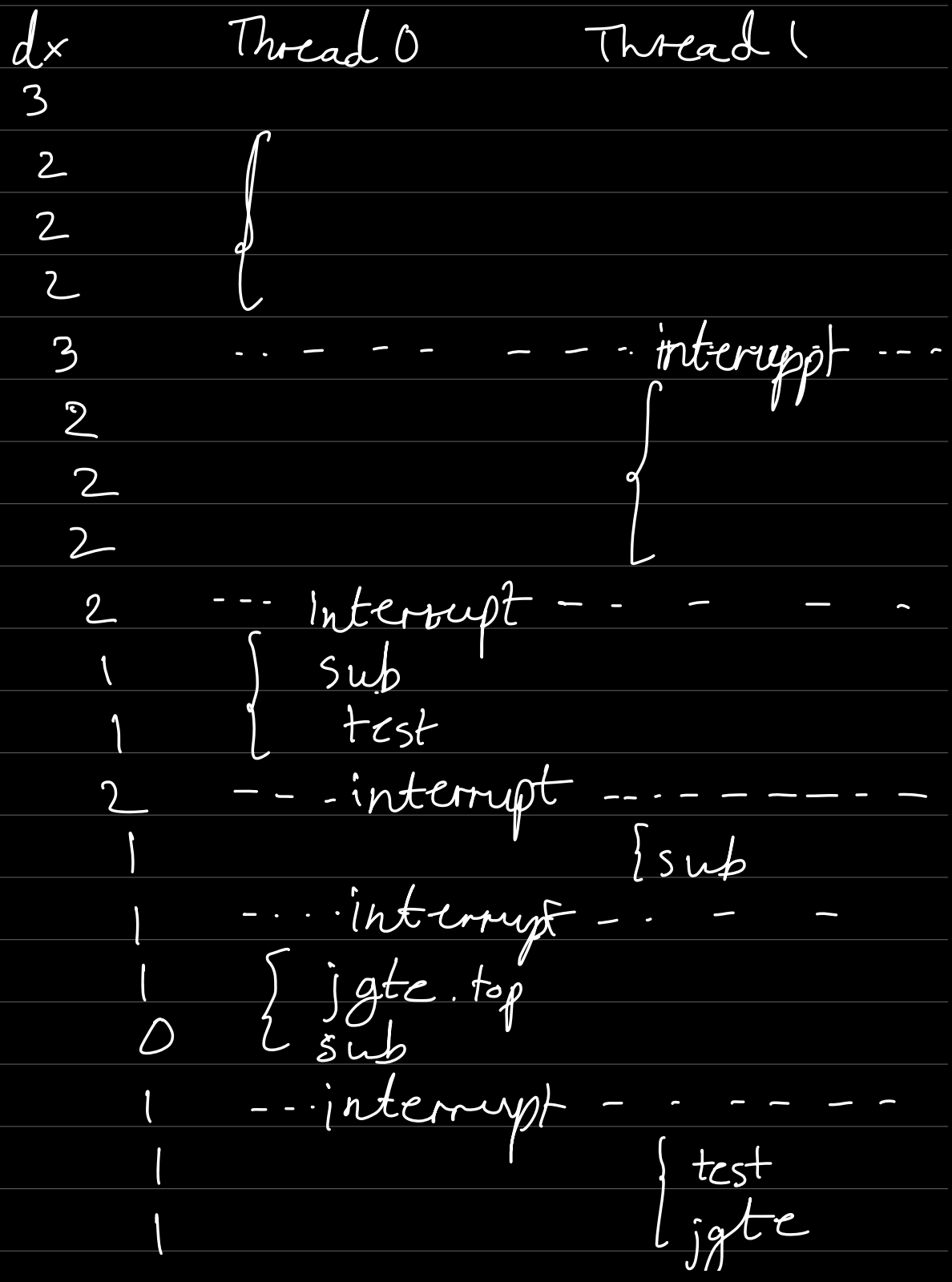
No race condition.

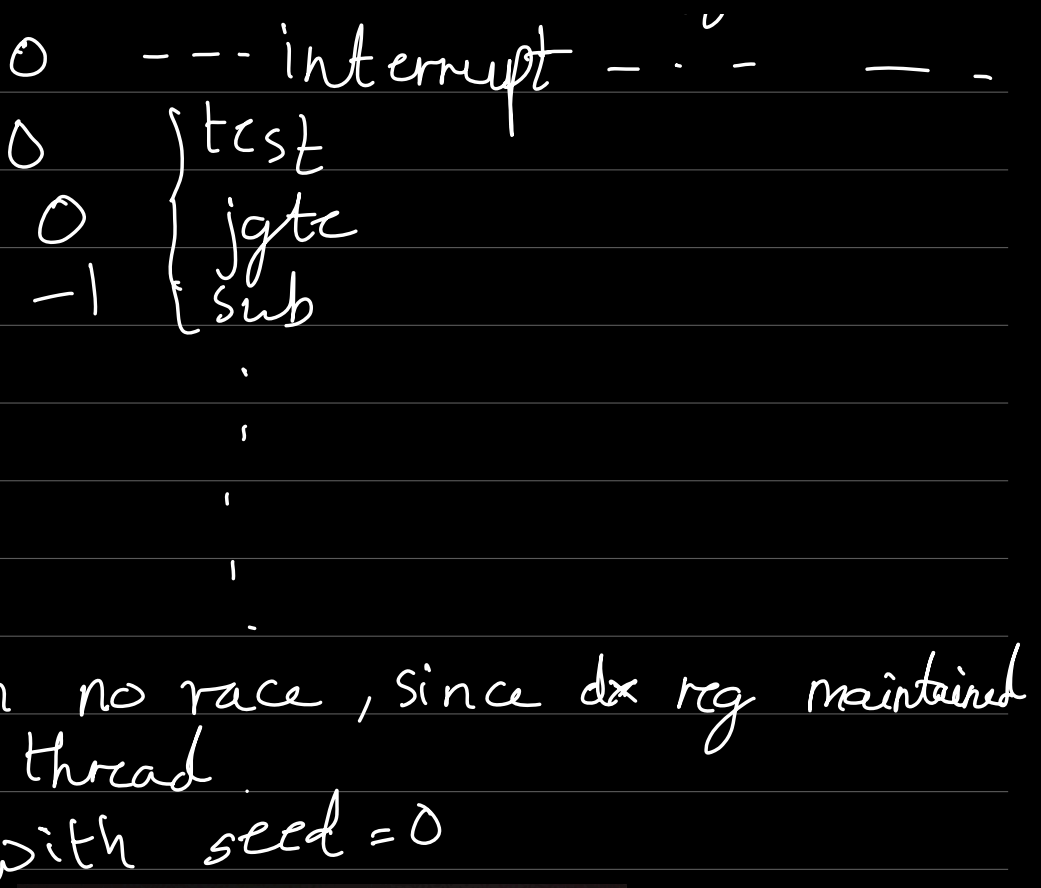
```
dx          Thread 0                Thread 1
 3
 2    1000 sub  $1,%dx
 2    1001 test $0,%dx
 2    1002 jgte .top
 1    1000 sub  $1,%dx
 1    1001 test $0,%dx
 1    1002 jgte .top
 0    1000 sub  $1,%dx
 0    1001 test $0,%dx
 0    1002 jgte .top
-1    1000 sub  $1,%dx
-1    1001 test $0,%dx
-1    1002 jgte .top
-1    1003 halt
 3    ----- Halt;Switch -----  ----- Halt;Switch -----
 2                             1000 sub  $1,%dx
 2                             1001 test $0,%dx
 2                             1002 jgte .top
 1                             1000 sub  $1,%dx
 1                             1001 test $0,%dx
 1                             1002 jgte .top
 0                             1000 sub  $1,%dx
 0                             1001 test $0,%dx
 0                             1002 jgte .top
-1                             1000 sub  $1,%dx
-1                             1001 test $0,%dx
-1                             1002 jgte .top
-1                             1003 halt
```

(3) ./x86.py -p loop.s -t 2 -i 3 -r -a dx=3,dx=3 -R dx

Now interrupt interval is low, so it will switch between threads

With different "dx" values.





Again no race, since dx reg maintained per thread.

with seed = 0

dx	Thread 0	Thread 1
3		
2	1000 sub $1,%dx	
2	1001 test $0,%dx	
2	1002 jgte .top	
3	------ Interrupt ------	------ Interrupt ------
2		1000 sub $1,%dx
2		1001 test $0,%dx
2		1002 jgte .top
2	------ Interrupt ------	------ Interrupt ------
1	1000 sub $1,%dx	
1	1001 test $0,%dx	
2	------ Interrupt ------	------ Interrupt ------
1		1000 sub $1,%dx
1	------ Interrupt ------	------ Interrupt ------
1	1002 jgte .top	
0	1000 sub $1,%dx	
1	------ Interrupt ------	------ Interrupt ------
1		1001 test $0,%dx
1		1002 jgte .top
0	------ Interrupt ------	------ Interrupt ------
0	1001 test $0,%dx	
0	1002 jgte .top	
-1	1000 sub $1,%dx	
1	------ Interrupt ------	------ Interrupt ------
0		1000 sub $1,%dx
-1	------ Interrupt ------	------ Interrupt ------
-1	1001 test $0,%dx	
-1	1002 jgte .top	
0	------ Interrupt ------	------ Interrupt ------
0		1001 test $0,%dx
0		1002 jgte .top
-1	------ Interrupt ------	------ Interrupt ------
-1	1003 halt	
0	----- Halt;Switch -----	----- Halt;Switch -----
-1		1000 sub $1,%dx
-1		1001 test $0,%dx
-1	------ Interrupt ------	------ Interrupt ------
-1		1002 jgte .top
-1		1003 halt

With seed = 1

```
dx          Thread 0                Thread 1
 3
 2   1000 sub  $1,%dx
 3   ------ Interrupt ------  ------ Interrupt ------
 2                           1000 sub  $1,%dx
 2                           1001 test $0,%dx
 2                           1002 jgte .top
 2   ------ Interrupt ------  ------ Interrupt ------
 2   1001 test $0,%dx
 2   1002 jgte .top
 1   1000 sub  $1,%dx
 2   ------ Interrupt ------  ------ Interrupt ------
 1                           1000 sub  $1,%dx
 1   ------ Interrupt ------  ------ Interrupt ------
 1   1001 test $0,%dx
 1   1002 jgte .top
 1   ------ Interrupt ------  ------ Interrupt ------
 1                           1001 test $0,%dx
 1                           1002 jgte .top
 1   ------ Interrupt ------  ------ Interrupt ------
 0   1000 sub  $1,%dx
 0   1001 test $0,%dx
 1   ------ Interrupt ------  ------ Interrupt ------
 0                           1000 sub  $1,%dx
 0                           1001 test $0,%dx
 0                           1002 jgte .top
 0   ------ Interrupt ------  ------ Interrupt ------
 0   1002 jgte .top
 0   ------ Interrupt ------  ------ Interrupt ------
-1                           1000 sub  $1,%dx
 0   ------ Interrupt ------  ------ Interrupt ------
-1   1000 sub  $1,%dx
-1   1001 test $0,%dx
-1   1002 jgte .top
-1   ------ Interrupt ------  ------ Interrupt ------
-1                           1001 test $0,%dx
-1                           1002 jgte .top
-1   ------ Interrupt ------  ------ Interrupt ------
-1   1003 halt
-1   ----- Halt;Switch -----  ----- Halt;Switch -----
-1                           1003 halt
```

(4) with looping-race-nolock.s and thread = 1

Addr 2000 Thread 0

```
0       mov 2000, %ax
0       add $1, %ax
1       mov %ax, 2000
1       sub $1, %bx   (-1)
1       test $0, %bx  (-1)
1       jgt .top      X
       halt
```

(5) `./x86.py -p looping-race-nolock.s -t 2 -a bx=3 -M 2000`

Each thread loops 3 times because "bx" register is not shared between threads unlike a specific addr location like 2000.

Default interrupt frequency = 50, so thread runs to completion before switching

Addr 2000	Thread 0	Thread 1
0		
0		
1		
	sub $1 %bx (2)	

test $0 %bx
jgt .top 2>0

2 bx(1)
3 bx(0)x

halt; switch

4
5
6

```
2000          Thread 0                Thread 1
   0
   0   1000 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   1000 mov 2000, %ax
   1   1001 add $1, %ax
   2   1002 mov %ax, 2000
   2   1003 sub  $1, %bx
   2   1004 test $0, %bx
   2   1005 jgt .top
   2   1000 mov 2000, %ax
   2   1001 add $1, %ax
   3   1002 mov %ax, 2000
   3   1003 sub  $1, %bx
   3   1004 test $0, %bx
   3   1005 jgt .top
   3   1006 halt
   3   ----- Halt;Switch -----  ----- Halt;Switch -----
   3                            1000 mov 2000, %ax
   3                            1001 add $1, %ax
   4                            1002 mov %ax, 2000
   4                            1003 sub  $1, %bx
   4                            1004 test $0, %bx
   4                            1005 jgt .top
   4                            1000 mov 2000, %ax
   4                            1001 add $1, %ax
   5                            1002 mov %ax, 2000
   5                            1003 sub  $1, %bx
   5                            1004 test $0, %bx
   5                            1005 jgt .top
   5                            1000 mov 2000, %ax
   5                            1001 add $1, %ax
   6                            1002 mov %ax, 2000
   6                            1003 sub  $1, %bx
   6                            1004 test $0, %bx
   6                            1005 jgt .top
   6                            1006 halt
```

(6) with seed 0

interleaving is fine, doesn't interleave the addition section but that doesn't really matter for the test instruction only that the final returned value at 2000 addr can be different if "add" instruction is interleaved but isn't in this run

Timing of the interrupt matter - Yes, with "2"/1 "add" and "mov" is interleaved which can cause other thread to run to a point which updates "2000" addr to "1" and then this thread does same "1", instead of "2" affecting "value" final value.

Interleaving the "sub" and "test" doesn't cause any issues because both threads have their own values.

```
2000          Thread 0                Thread 1
   0
   0   1000 mov 2000, %ax
   0   1001 add $1, %ax
   1   1002 mov %ax, 2000
   1   1003 sub  $1, %bx
   1   ------ Interrupt ------  ------ Interrupt ------
   1                           1000 mov 2000, %ax
   1                           1001 add $1, %ax
   2                           1002 mov %ax, 2000
   2                           1003 sub  $1, %bx
   2   ------ Interrupt ------  ------ Interrupt ------
   2   1004 test $0, %bx
   2   1005 jgt .top
   2   ------ Interrupt ------  ------ Interrupt ------
   2                           1004 test $0, %bx
   2                           1005 jgt .top
   2   ------ Interrupt ------  ------ Interrupt ------
   2   1006 halt
   2   ----- Halt;Switch -----  ----- Halt;Switch -----
   2                           1006 halt
```

seed=1, see with "1" interrupt interval change

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

seed = 2

```
2000          Thread 0                  Thread 1
   0
   0   1000 mov 2000, %ax
   0   1001 add $1, %ax
   1   1002 mov %ax, 2000
   1   1003 sub  $1, %bx
   1   ------ Interrupt ------  ------ Interrupt ------
   1                            1000 mov 2000, %ax
   1                            1001 add $1, %ax
   2                            1002 mov %ax, 2000
   2                            1003 sub  $1, %bx
   2   ------ Interrupt ------  ------ Interrupt ------
   2   1004 test $0, %bx
   2   ------ Interrupt ------  ------ Interrupt ------
   2                            1004 test $0, %bx
   2   ------ Interrupt ------  ------ Interrupt ------
   2   1005 jgt .top
   2   1006 halt
   2   ----- Halt;Switch -----  ----- Halt;Switch -----
   2                            1005 jgt .top
   2                            1006 halt
```

with i = 2, (interrupt interval)

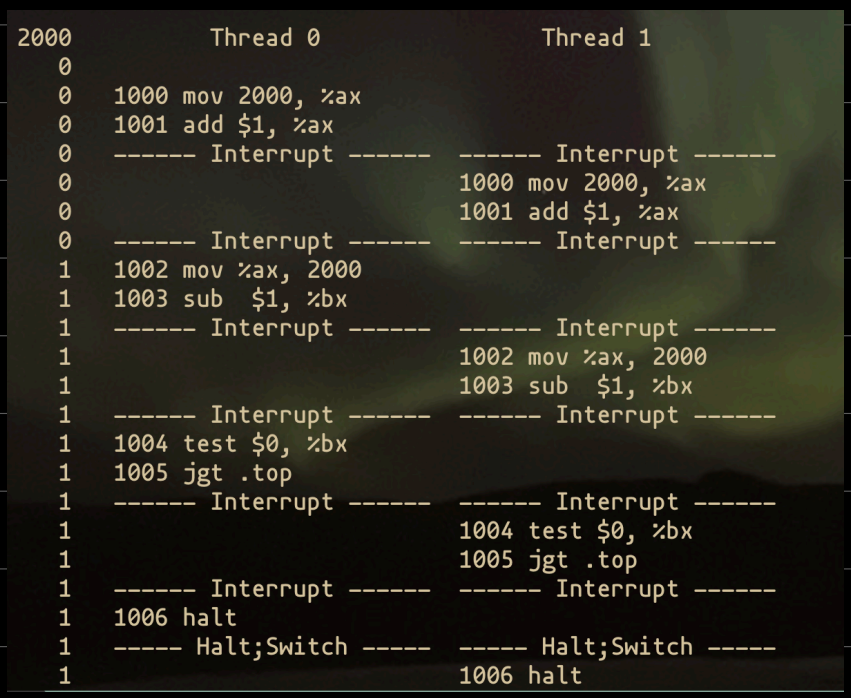
```
2000          Thread 0                  Thread 1
   0
   0   1000 mov 2000, %ax
   0   1001 add $1, %ax
   0   ------ Interrupt ------  ------ Interrupt ------
   0                            1000 mov 2000, %ax
   0                            1001 add $1, %ax
   0   ------ Interrupt ------  ------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

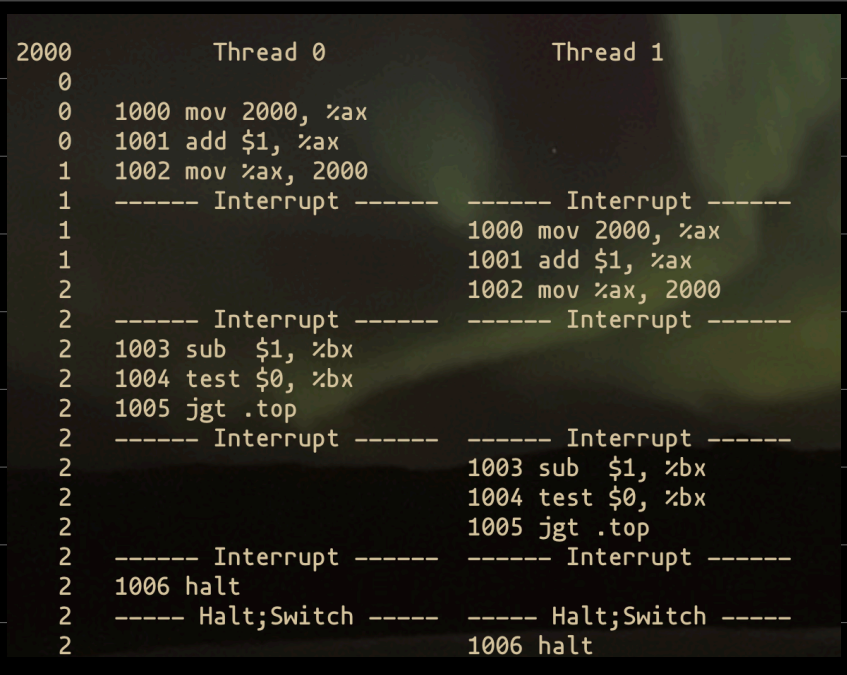
Critical section

```
.main
.top
# critical section
mov 2000, %ax  # get 'value' at address 2000
add $1, %ax    # increment it
mov %ax, 2000  # store it back

# see if we're still looping
sub  $1, %bx
test $0, %bx
jgt .top

halt
```

(7)

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


2000 Thread 0 Thread 1
0
0 1000 mov 2000, %ax
0 1001 add $1, %ax
0 ------ Interrupt ------ ------ Interrupt ------
0 1000 mov 2000, %ax
0 1001 add $1, %ax
0 ------ Interrupt ------ ------ Interrupt ------
1 1002 mov %ax, 2000
1 1003 sub $1, %bx
1 ------ Interrupt ------ ------ Interrupt ------
1 1002 mov %ax, 2000
1 1003 sub $1, %bx
1 ------ Interrupt ------ ------ Interrupt ------
1 1004 test $0, %bx
1 1005 jgt .top
1 ------ Interrupt ------ ------ Interrupt ------
1 1004 test $0, %bx
1 1005 jgt .top
1 ------ Interrupt ------ ------ Interrupt ------
1 1006 halt
1 ----- Halt;Switch ----- ----- Halt;Switch -----
1 1006 halt


2000 Thread 0 Thread 1
0
0 1000 mov 2000, %ax
0 1001 add $1, %ax
1 1002 mov %ax, 2000
1 ------ Interrupt ------ ------ Interrupt ------
1 1000 mov 2000, %ax
1 1001 add $1, %ax
2 1002 mov %ax, 2000
2 ------ Interrupt ------ ------ Interrupt ------
2 1003 sub $1, %bx
2 1004 test $0, %bx
2 1005 jgt .top
2 ------ Interrupt ------ ------ Interrupt ------
2 1003 sub $1, %bx
2 1004 test $0, %bx
2 1005 jgt .top
2 ------ Interrupt ------ ------ Interrupt ------
2 1006 halt
2 ----- Halt;Switch ----- ----- Halt;Switch -----
2 1006 halt

(8) i=4, due to alignment we get partial of expected value

```
149   ------ Interrupt ------  ------ Interrupt ------
149   1004 test $0, %bx
149   1005 jgt .top
149   1000 mov 2000, %ax
149   1001 add $1, %ax
149   ------ Interrupt ------  ------ Interrupt ------
149                            1004 test $0, %bx
149                            1005 jgt .top
149                            1000 mov 2000, %ax
149                            1001 add $1, %ax
149   ------ Interrupt ------  ------ Interrupt ------
150   1002 mov %ax, 2000
150   1003 sub  $1, %bx
150   1004 test $0, %bx
150   1005 jgt .top
150   ------ Interrupt ------  ------ Interrupt ------
150                            1002 mov %ax, 2000
150                            1003 sub  $1, %bx
150                            1004 test $0, %bx
150                            1005 jgt .top
150   ------ Interrupt ------  ------ Interrupt ------
150   1006 halt
150   ----- Halt;Switch -----  ----- Halt;Switch -----
150                            1006 halt
```

i = 3

```
198   ------ Interrupt ------  ------ Interrupt ------
198   1000 mov 2000, %ax
198   1001 add $1, %ax
199   1002 mov %ax, 2000
199   ------ Interrupt ------  ------ Interrupt ------
199                            1000 mov 2000, %ax
199                            1001 add $1, %ax
200                            1002 mov %ax, 2000
200   ------ Interrupt ------  ------ Interrupt ------
200   1003 sub  $1, %bx
200   1004 test $0, %bx
200   1005 jgt .top
200   ------ Interrupt ------  ------ Interrupt ------
200                            1003 sub  $1, %bx
200                            1004 test $0, %bx
200                            1005 jgt .top
200   ------ Interrupt ------  ------ Interrupt ------
200   1006 halt
200   ----- Halt;Switch -----  ----- Halt;Switch -----
200                            1006 halt
```

i = 1, 2

```
100   1002 mov %ax, 2000
100   1003 sub  $1, %bx
100   ------ Interrupt ------  ------ Interrupt ------
100                            1002 mov %ax, 2000
100                            1003 sub  $1, %bx
100   ------ Interrupt ------  ------ Interrupt ------
100   1004 test $0, %bx
100   1005 jgt .top
100   ------ Interrupt ------  ------ Interrupt ------
100                            1004 test $0, %bx
100                            1005 jgt .top
100   ------ Interrupt ------  ------ Interrupt ------
100   1006 halt
100   ----- Halt;Switch -----  ----- Halt;Switch -----
100                            1006 halt
```

(a)

2000	ax	Thread 0	Thread 1
0	1	test $1, %ax	
0	1	je .signaller	
1	1	mov $1, 2000	
1	1	halt	
1	0		test $1, %ax
1	0		je .signallerx
1	0		.waiter
1	0		mov 2000, %cx
1	0		test $1, %cx
1	0		jne .waiterx
1	0		halt

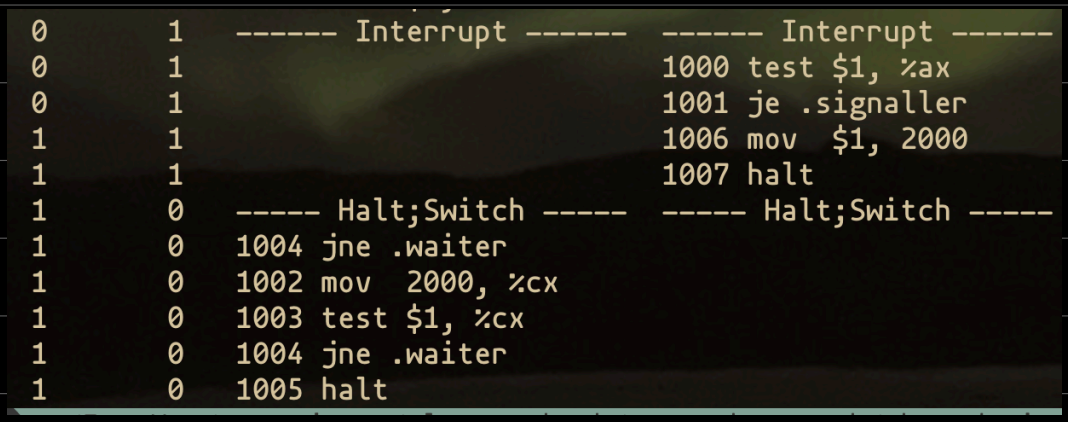
2000	ax	Thread 0	Thread 1
0	1		
0	1	1000 test $1, %ax	
0	1	1001 je .signaller	
1	1	1006 mov $1, 2000	
1	1	1007 halt	
1	0	----- Halt;Switch -----	----- Halt;Switch -----
1	0		1000 test $1, %ax
1	0		1001 je .signaller
1	0		1002 mov 2000, %cx
1	0		1003 test $1, %cx
1	0		1004 jne .waiter
1	0		1005 halt

(10) Switch ax inputs

2000	ax	ax=0 Thread 0	ax=1 Thread 1
0	0	test $1, %ax	
0	0	je .signaller X	
0	0	.waiter	
0	0	mov 2000, %cx	
0	0	test $1, %cx	
0	0	jne .waiter	
		loops again and again until interrupt frequency is matched	
0	1		test $1, %ax
			je .signaller
			.signaller
1	1		mov $1, 2000
1	1		halt
1	0	mov 2000 %cx	
1	0	test $1, %cx	
1	0	jne .waiter X	

1 0 halt

With i = 1000, also should be same result



```
0   1   ------ Interrupt ------   ------ Interrupt ------
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   1004 jne .waiter
1   0   1002 mov  2000, %cx
1   0   1003 test $1, %cx
1   0   1004 jne .waiter
1   0   1005 halt
```

Nope it isn't using the CPU efficiently.
"Busy waiting" instead of sleeping or yielding CPU time.