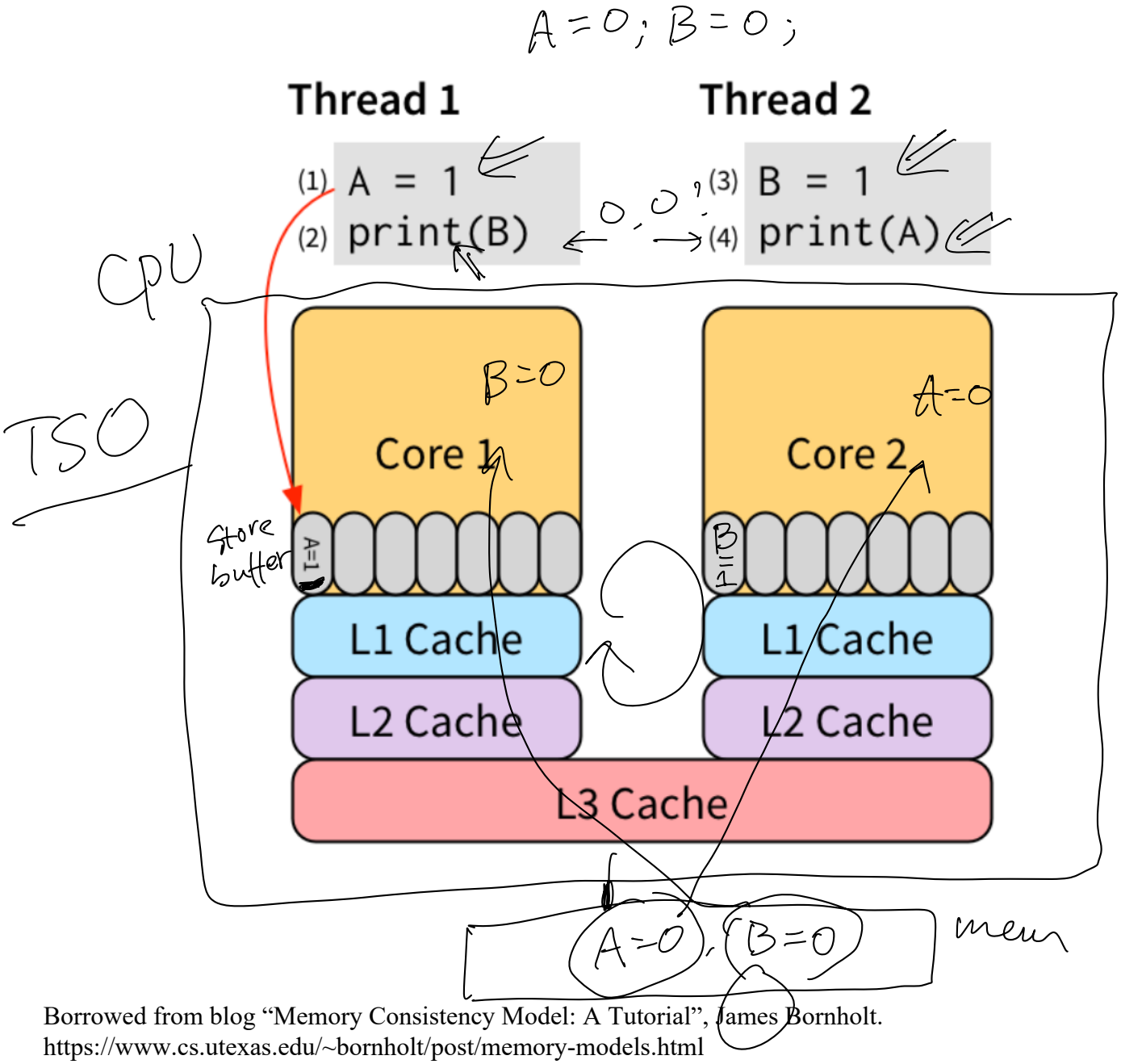


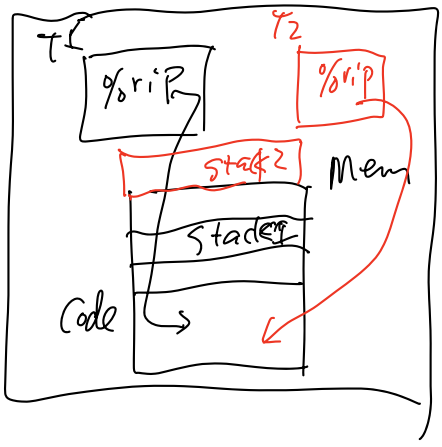
Borrowed from blog "Memory Consistency Model: A Tutorial", James Bornholt.
https://www.cs.utexas.edu/~bornholt/post/memory-models.html

Week 6.a
CS3650
02/12 2024
https://naizhengtan.github.io/24spring/

0. Processes vs. threads
1. Intro to concurrency
2. Managing concurrency
3. Mutexes

Q: What's the difference between processes and threads?

process

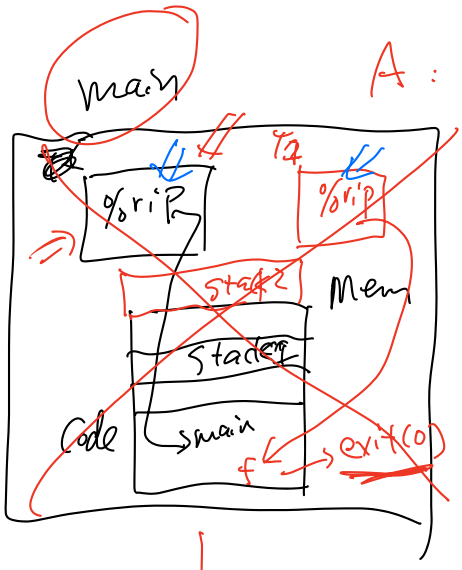


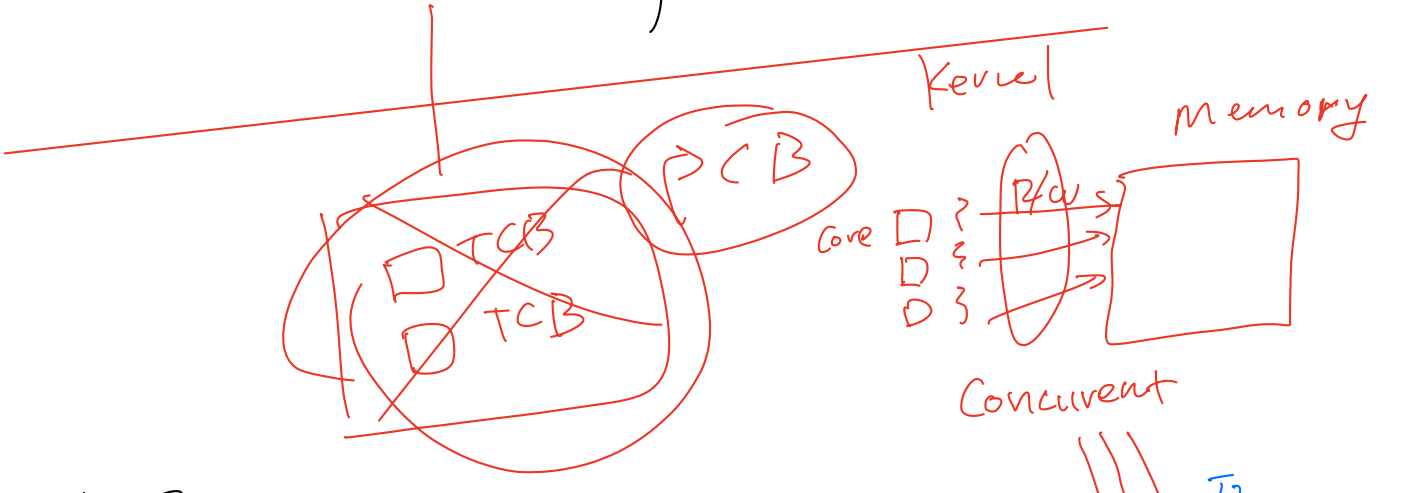
A:
this is f
this is main

B:
this is f

C:
this is main

D:
others





1. Intro to Concurrency

1. Example to illustrate interleavings: say that thread A executes f() and thread B executes g(). (Here, we are using the term "thread" abstractly. This example applies to any of the approaches that fall under the word "thread".)

a. [this is pseudocode]

```
int x;

int main(int argc, char** argv) {

    tid tid1 = thread_create(f, NULL);
    tid tid2 = thread_create(g, NULL);

    thread_join(tid1);
    thread_join(tid2);

    printf("%d\n", x);
}

void f() {
    x = 1;
    thread_exit();
}

void g() {
    x = 2;
    thread_exit();
}
```

What are possible values of x after A has executed f() and B has executed g()? In other words, what are possible outputs of the program above?

b. Same question as above, but f() and g() are now defined as follows

```
int x;
int y = 12;

f() { x = y + 1; }
g() { y = y * 2; }
```

What are the possible values of x?

c. Same question as above, but f() and g() are now defined as follows:

```
int x = 0;

f() { x = x + 1; }
g() { x = x + 2; }
```

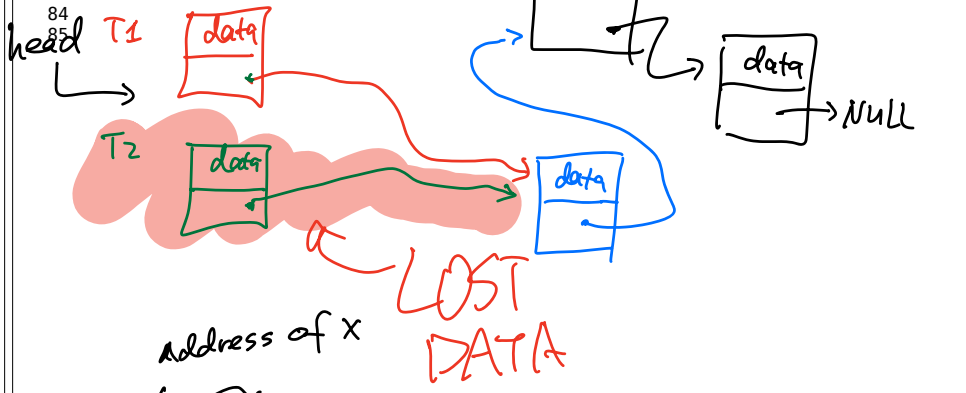
What are the possible values of x?

2. Linked list example

```
struct List_elem {
    int data;
    struct List_elem* next;
};

List_elem* head = 0;

insert(int data) {
    List_elem* l = new List_elem;
    l->data = data;
    l->next = head;
    head = l;
}
```

What happens if two threads execute insert() at once and we get the following interleaving?

```
thread 1: l->next = head
thread 2: l->next = head
thread 2: head = l;
thread 1: head = l;
```



address of x

```
movq 0x5000, %rbx    # load from address 0x5000 into register
addq $1, %rbx        # add 1 to the register's value
movq %rbx, 0x5000    # store back
```

```
movq 0x5000, %rbx
addq $2, %rbx
movq %rbx, 0x5000
```

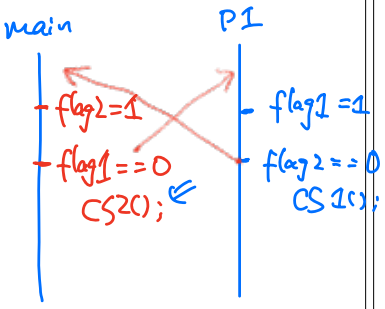
CS5600 Week06.a

1. Some concurrent programs. What is the point of these?

[From S.V. Adve and K. Gharachorloo, IEEE Computer, December 1996, 66-76. http://sadve.cs.illinois.edu/Publications/computer96.pdf]

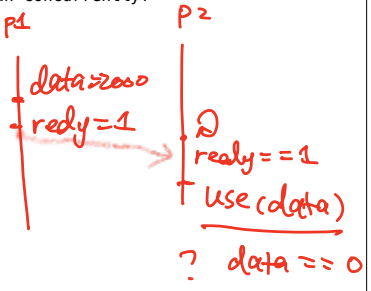
a. Can both "critical sections" run?

```
int flag1 = 0, flag2 = 0;

int main () {
    tid id = thread_create (p1, NULL);
    p2 (); thread_join (id);
}

void p1 (void *ignored) {
    flag1 = 1;
    if (!flag2) {
        critical_section_1 ();
    }
}

void p2 (void *ignored) {
    flag2 = 1;
    if (!flag1) {
        critical_section_2 ();
    }
}
```



b. Can use() be called with value 0, if p2 and p1 run concurrently?

```
int data = 0, ready = 0;

void p1 () {
    data = 2000;
    ready = 1;
}
int p2 () {
    while (!ready) {}
    use(data);
}
```

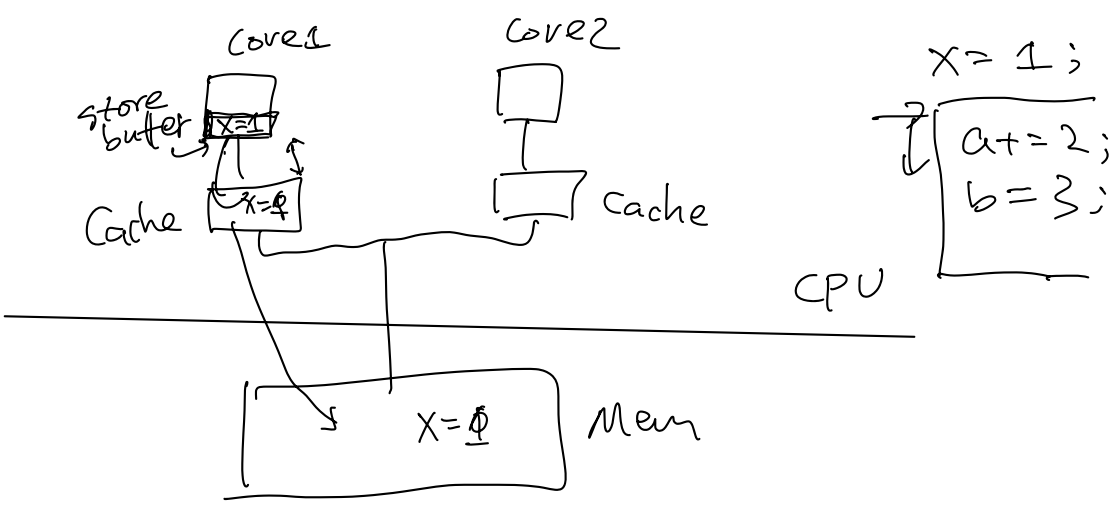


c. Can use() be called with value 0?

```
int a = 0, b = 0;

void p1 (void *ignored) { a = 1; }

void p2 (void *ignored) {
  if (a == 1)
    b = 1;
}

void p3 (void *ignored) {
  if (b == 1)
    use (a);
}
```


2. Protecting the linked list...

```
Mutex list_mutex;

insert(int data) {
  List_elem* l = new List_elem;
  l->data = data;

  acquire(&list_mutex);

  l->next = head;
  head = l;

  release(&list_mutex);
}
```



- Concurrency problem from Software

```
int x = 0;  // a global variable

void foo() {        //T1
  for (int i=0; i<100; i++) {
    x = 1;
    printf("%d", x);
  }
}

void bar() {        //T2
  x = 0;
}
```

X=0

1111111...

1101...

bars

1100...0

Could happen