Software Transactional Memory (STM)

Introduction

```
class Account {
  double balance;
  void debit(double amount){
     balance -= amount;
  }
  void credit(double amount){
     balance += amount;
  }
  void transfer(Account from, Account to, double amount){
        lock(from);
        lock(to);

        from.debit(amount);
        to.credit(amount);

        release(to);
        release(from);
  }
}
```

Deadlock

```
class Account {
  double balance;
  void debit(double amount){
     balance -= amount;
  }
  void credit(double amount){
     balance += amount;
  }
  void transfer(Account from, Account to, double amount){
        lock(from);
        lock(to);

        from.debit(amount);
        to.credit(amount);

        release(to);
        release(from);
  }
}
```

Process 1	Process 2
transfer (a, b)	transfer (b, a)

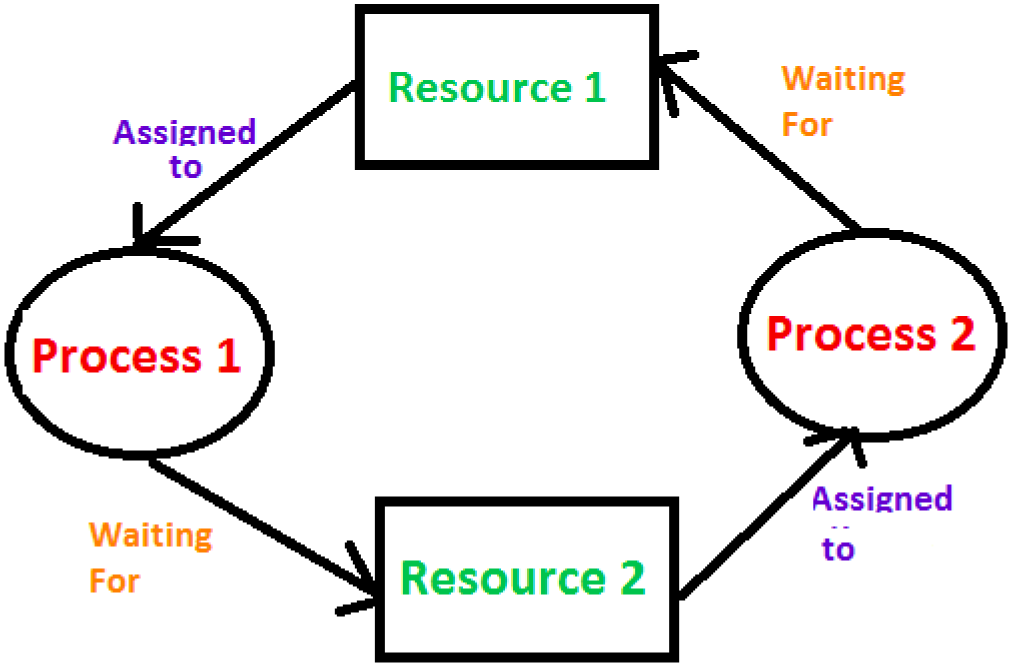
?

Deadlock

```
class Account {
  double balance;
  void debit(double amount){
     balance -= amount;
  }
  void credit(double amount){
     balance += amount;
  }
  void transfer(Account from, Account to, double amount){
        lock(from);
        lock(to);

        from.debit(amount);
        to.credit(amount);

        release(to);
        release(from);
  }
}
```



Deadlock

Deadlock

```
class Account {
  double balance;
  void debit(double amount){
     balance -= amount;
  }
  void credit(double amount){
     balance += amount;
  }
  void transfer(Account from, Account to, double amount){
        lock(from);
        lock(to);

        from.debit(amount);
        to.credit(amount);

        release(to);
        release(from);
  }
}
```

Thread 1	Thread 2
transfer (a, b)	transfer (b, a)

?

Software Transactional Memory (STM)

```
lock()
  a.x = t1
  a.y = t2
  if (a.z == 0) {
    a.x = 0
    a.z = t3
  }
release()
```

```
tmTXBegin()
tmWr(&a.x, t1)
tmWr(&a.y, t2)
if (tmRd(&a.z) != 0) {
  tmWr(&a.x, 0)
  tmWr(&a.z, t3)
}
tmTXCommit()
```

The STM API (a simple view)

- **begin()** returns ok

- **read()** returns a value or abort
- **write()** returns ok or abort

- **commit()** returns ok or abort
- **abort()** returns ok

Software Transactional Memory (STM)

- Transactions should respect the ACID property (atomicity, consistency, isolation and durability)
    - Appear as a single operation (no inconsistency)
    - Writes should be visible from outside only after commit
    - Should not interfere with other running transactions