HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

OPERATING SYSTEM

Lab 6

SYNCHRONIZATION

Giảng viên hướng dẫn: Hoàng Lê Hải Thanh

Group 7: Trần Minh Trí - 1910637

Trần Quốc Vinh - 1915953

Võ Văn Hiền - 2020023

Phạm Duy Quang - 2011899

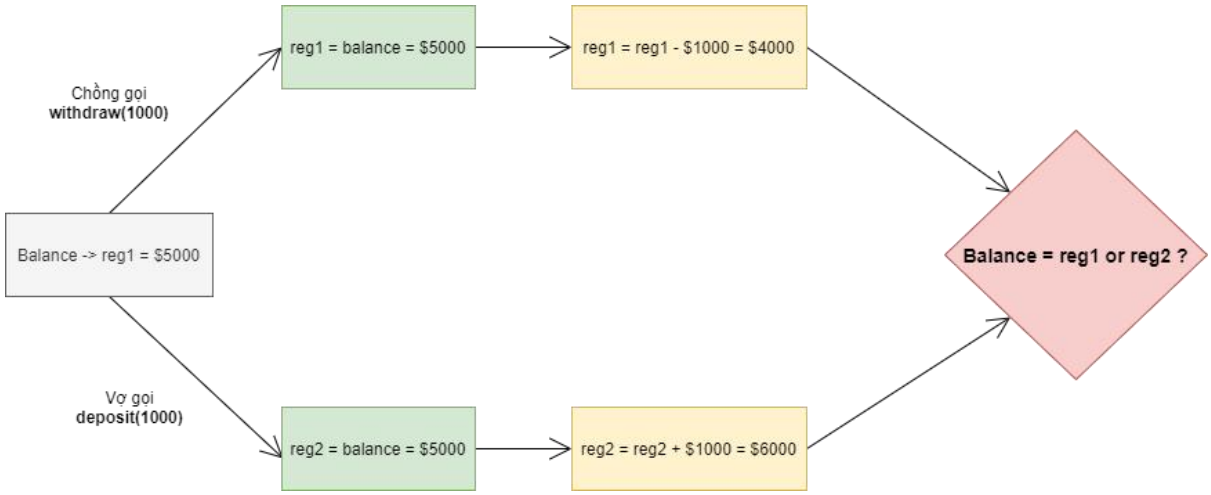
-10 /2021 -

Ta thấy rằng race condition xảy ra không phải xảy ra giữa người chồng và người vợ mà là do sự bất đồng bộ giữa withdraw() và deposit() (nạp tiền và rút tiền).

Sự bất đồng bộ sẽ được minh họa qua ví dụ sau đây:

- Đặt 1 biến Balance = $5000 là số tiền cân bằng hiện tại trong tài khoản ngân hàng.
- Khi có 1 trong hai vợ chồng gọi withdraw() hoặc deposit(), ta giả sử chồng gọi withdraw(1000) ; vợ gọi deposit(5000)

Lúc này ta sẽ có race condition như hình sau:



Vậy vấn đề ở đây là sau khi 2 phương thức được gọi cùng lúc biến Balance sẽ không biết được nó nên có giá trị nào.

=> **Để giải quyết vấn đề này, ta có thể dùng biện pháp "Lock" trên biến Balance hay bất kỳ vùng nhớ dùng chung nào khác của 2 thao tác này:**

- Mỗi phương thức trước khi tiếp cận và thay đổi trạng thái của biến đó thành "locked".
- Nếu một phương thức khác muốn tiếp cận và thay đổi giá trị của Balance mà xác nhận được biến Balanced đang bị "locked" thì phải đợi cho đến khi phương thức đang "lock" biến kết thúc.

Lưu ý: Ta nên hiện thực thực 1 cách nào đó để hệ thống nhận ra biến Balance bị khóa quá lâu. Khi đó nó sẽ tự động "unlock" biến này và kết thức phương thức đang khóa biến đó.

Điều này giúp tránh rơi vào tình trạng *deadlock*.

Mã giả đề xuất cho giải pháp xử lý race condition trên:

+) Withdrawl(value)

```
If BalanceState is locked, then wait
else Lock(BalanceState) // đưa gán cho biến trạng thái của Balance là locked
```

Assign (Balance - value) to Balance

UnLock (BalanceLock) // gán cho biến trạng thái của Balance là unlocked

end withdrawl