Име: _____ ФН: _____ Група: ____

Теоретични задачи за специалност КН, първи поток, 06.06.2021 г.:

**Задача 1,** (15 точки)

Всеки от процесите `P`, `Q` и `R` изпълнява поредица от три инструкции:

```
process P          process Q          process R
  p_1                q_1                r_1
  p_2                q_2                r_2
  p_3                q_3                r_3
```

Осигурете чрез семафори синхронизация на `P`, `Q` и `R` така, че да се изпълнят едновременно следните изисквания:

(а) Някоя от инструкциите `p_2` и `q_2` да се изпълни преди `r_2`.

(б) Ако инструкция `p_2` се изпълни преди `r_2`, то `q_2` да се изпълни след `r_2`.

(в) Ако инструкция `q_2` се изпълни преди `r_2`, то `p_2` да се изпълни след `r_2`.

**Задача 2,** (15 точки)

Каква задача решава инструментът spinlock (активно изчакване)?

Опишете хардуерните инструменти, необходими за реализацията на spinlock.

В кои ситуации не бива да се ползва spinlock?

## Примерни решения

**Задача 1.** За синхронизация използваме семафори `f` и `u`, инициализираме ги така:

```
semaphore f, u
f.init(1)
u.init(0)
```

Добавяме в кода на процесите `P`, `Q` и `R` синхронизиращи инструкции:

```
process P              process Q              process R
  p_1                    q_1                    r_1
  f.wait()               f.wait()               u.wait()
  p_2                    q_2                    r_2
  u.signal()             u.signal()             f.signal()
  p_3                    q_3                    r_3
```

Инструкция `r_2` може да се изпълни след като семафорът `u`, който в началото е блокиран, получи сигнал. Това става единствено след изпълнението на някоя от инструкциите `p_2` и `q_2`. Така осигуряваме изпълнението на условие (а).

Броячът на семафора `f` в началото е 1, само един от процесите `P` и `Q` ще премине реда си `f.wait()` и ще го нулира, другият процес ще чака сигнал. Това става само след изпълнението на ред `f.signal()` от процеса `R`, след изпълнение на инструкция `r_2`. Така осигуряваме изпълнението на условия (б) и (в).

Ако процесът `P` пръв достигне инструкцията `f.wait()`, ще се изпълни предпоставката на условие (б), редът на изпълнение на интересните инструкции ще е `p_2, r_2, q_2`.

Ако процесът `Q` пръв достигне инструкцията `f.wait()`, ще се изпълни предпоставката на условие (в), редът на изпълнение на интересните инструкции ще е `q_2, r_2, p_2`.

**Задача 2.** Spinlock се ползва за достъп до споделени данни в режим на взаимно изключване. Така се предотвратява разрушаването им, когато паралелно работещи процеси временно нарушат структурата им и създават условия за race condition.

Реализацията на spinlock използва байт(бит) lock, разположен в споделената памет. Стойността на lock показва дали паметта се ползва (lock=1) или е свободна (lock=0).

Преди изпълнението на критичната секция макросът spin_lock(lock) проверява и сменя с 1 стойността на lock в цикъл, докато завари стойност 0. Това става със специална атомарна инструкция, примерно test_and_set.

След изпълнението на критичната секция макросът spin_unlock(lock) сменя с 0 стойността на lock, така освобождава споделената памет.

Преди заемането на ресурса се забраняват прекъсванията, а след освобождаването се разрешават, с цел по-бързо изпълнение на критичната секция и защита от второ влизане в spin_lock(lock).

Операциите по манипулиране на прекъсванията, както и test_and_set са специфични хардуерни инструменти, без които не може да се направи удобна и бърза реализация на spinlock.

Spinlock не бива да се ползва при защита на критична секция, която може да продължи дълго време или да извика отново spin_lock(lock).