

EGM0017 (60h)

Fluxo e metodologias de projeto de Sistemas Embarcados

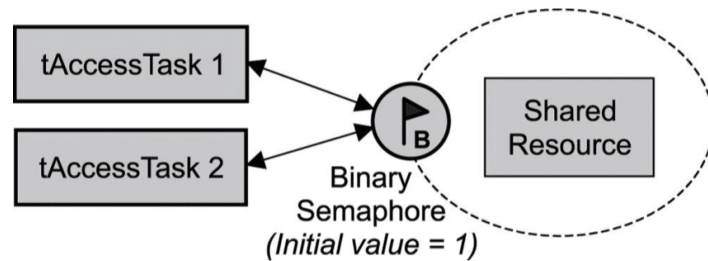
Prof. Josenalde Barbosa de Oliveira – UFRN

 josenalde.oliveira@ufrn.br

Programa de Pós-Graduação em Engenharia Mecatrônica

Sincronismo – semáforos (egpos e rtos)

- O semáforo binário tem funcionamento semelhante ao MUTEX
 - É inicializado com 1 (uma espécie de unsigned int, um tipo abstrato de dados)
 - Quando thread tenta obtê-lo checka se seu valor é ≥ 1 . Se for, obtém e decrementa 1
 - Se for igual a 0, não pode obter.
 - Se conseguir obter, ao concluir a tarefa, incrementa o semáforo em 1 unidade
- Um semáforo com qualquer valor inteiro positivo > 1 é do tipo CONTAGEM (counting)
 - Mais de uma thread acessando recurso compartilhado

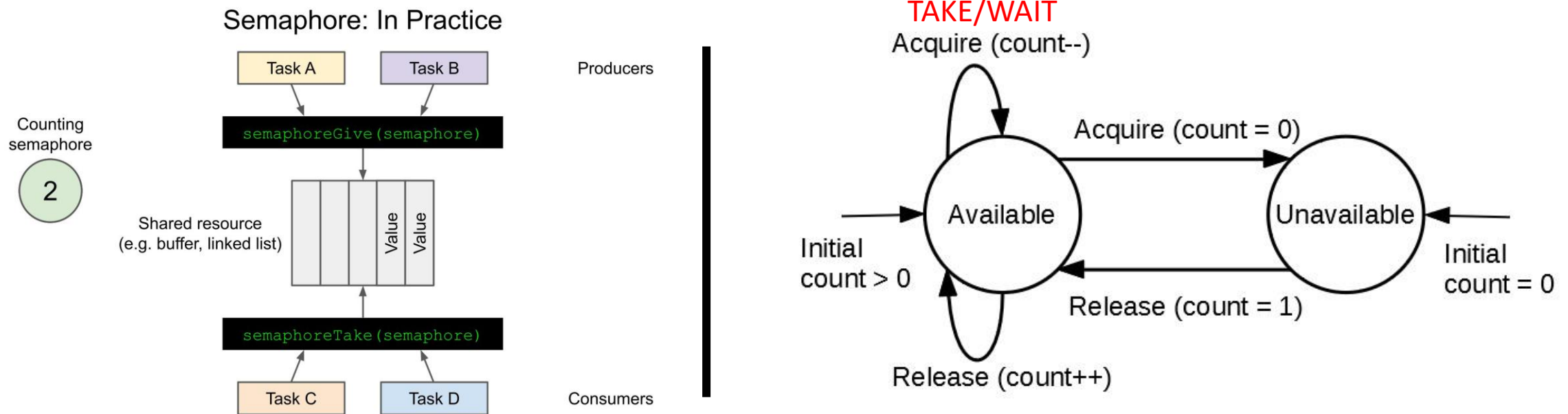


```
//antes da região crítica
sem_wait(&semaphore) // se valor==0, thread bloqueada
                    //Se valor !=0, decrementa e thread TAKE
// ao fim da execução na região crítica
sem_post(&semaphore) // incrementa (libera) GIVE
// outra thread esperando assume...
```



Edsger Dijkstra, 1930-2002

Sincronismo – semáforos (semaphore.h)

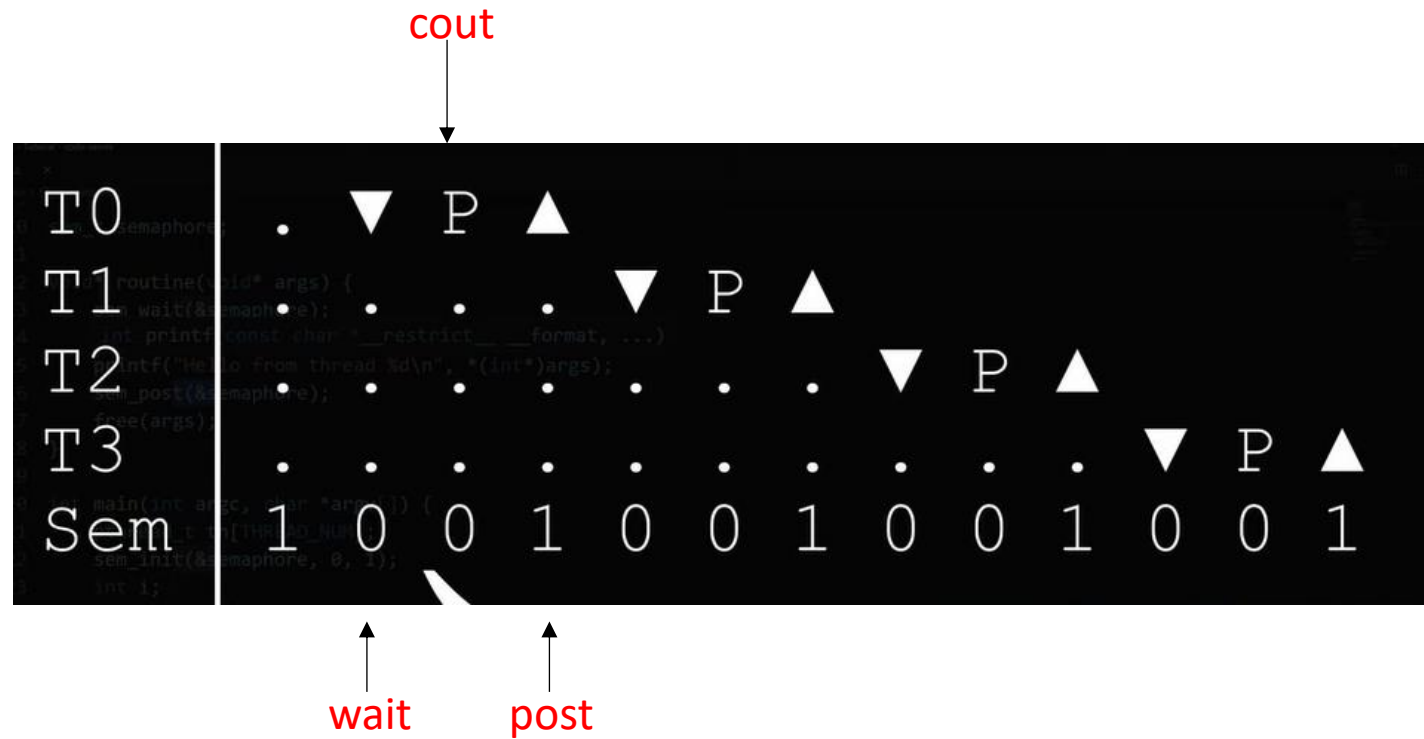


O tipo de sincronismo quando uma thread só pode seguir quando outra thread finaliza é chamado **produtor-consumidor**

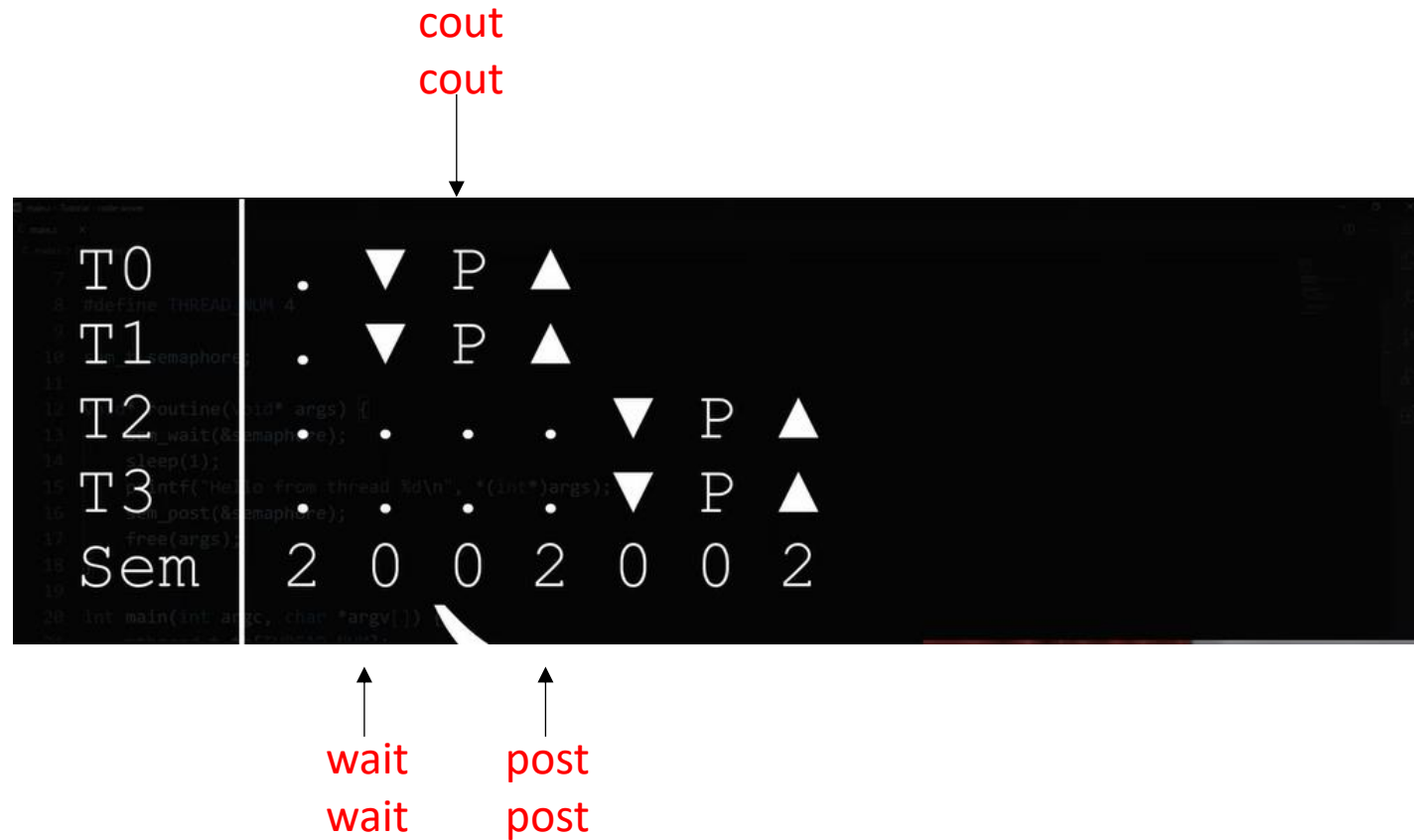
```
int sem_init(  
    sem_t* semaphore_p /* out */,  
    int shared /* in */, //Normalmente o segundo parâmetro é 0 para multithreading  
    unsigned initial_val /* in */);  
  
int sem_destroy(sem_t* semaphore_p /* in/out */);  
int sem_post(sem_t* semaphore_p /* in/out */);  
int sem_wait(sem_t* semaphore_p /* in/out */);
```

Exemplo

https://github.com/josenalde/flux-embedded-design/blob/main/src/semaphore_concept.cpp



Agora semáforo de contagem



Exemplo: fila de login

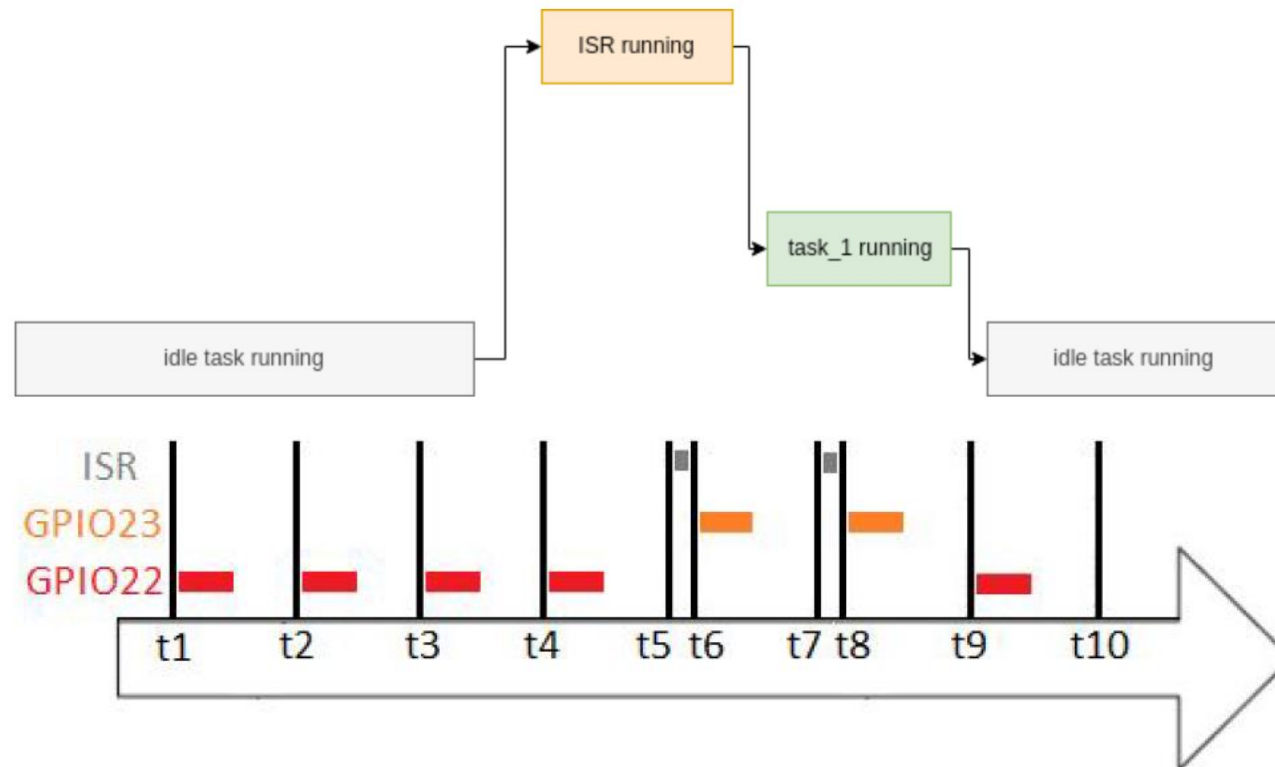
https://github.com/josenalde/flux-embedded-design/blob/main/src/semaphore_loginqueue.cpp

- Usuários aguardando login em server (games etc.)
- Menos recursos que demanda: gera fila, controlada por semáforos

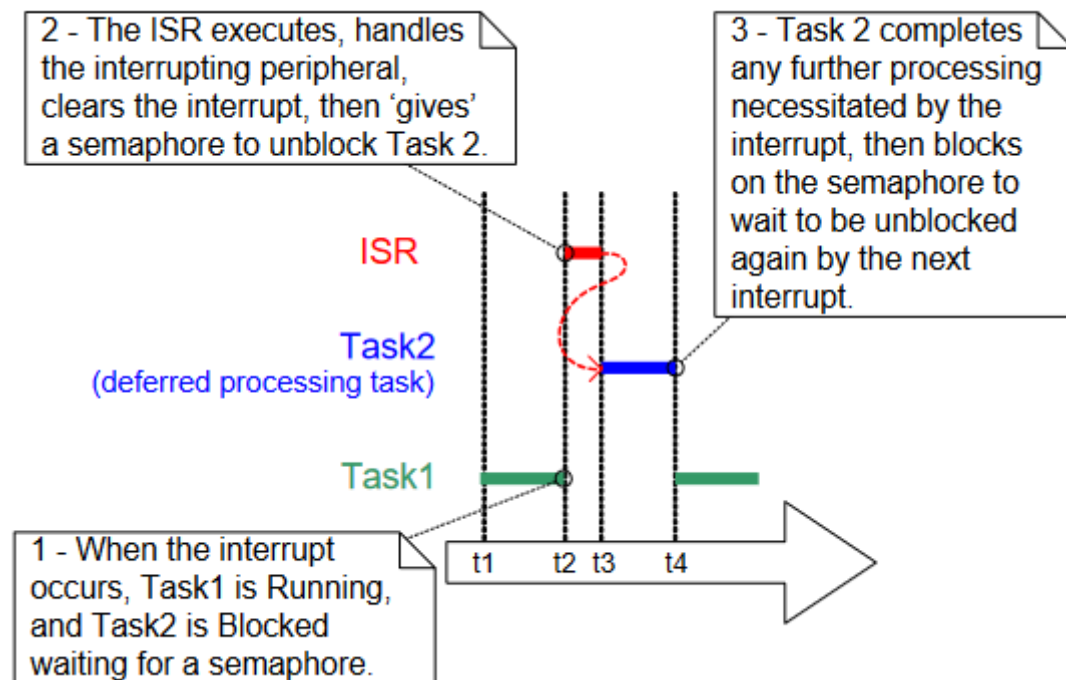
Exemplo: tratamento de interrupção no ESP32

https://github.com/josenalde/flux-embedded-design/blob/main/src/dih_semaphore_rtos/dih_semaphore_rtos.ino

- Objetivo: sincronizar TAREFA (task) com ISR, tornando a ISR a menor possível em termos de tempo de execução
- Fundamental para sistemas de tempo real, pois a ISR sempre tem a maior prioridade de qualquer outra TASK
- DIH (deferred interrupt handling)



Exemplo: tratamento de interrupção no ESP32



https://freertos.org/fr-content-src/uploads/2018/07/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf

Prof. Josenalde Oliveira – Fluxo e metodologias de projeto de sistemas embarcados

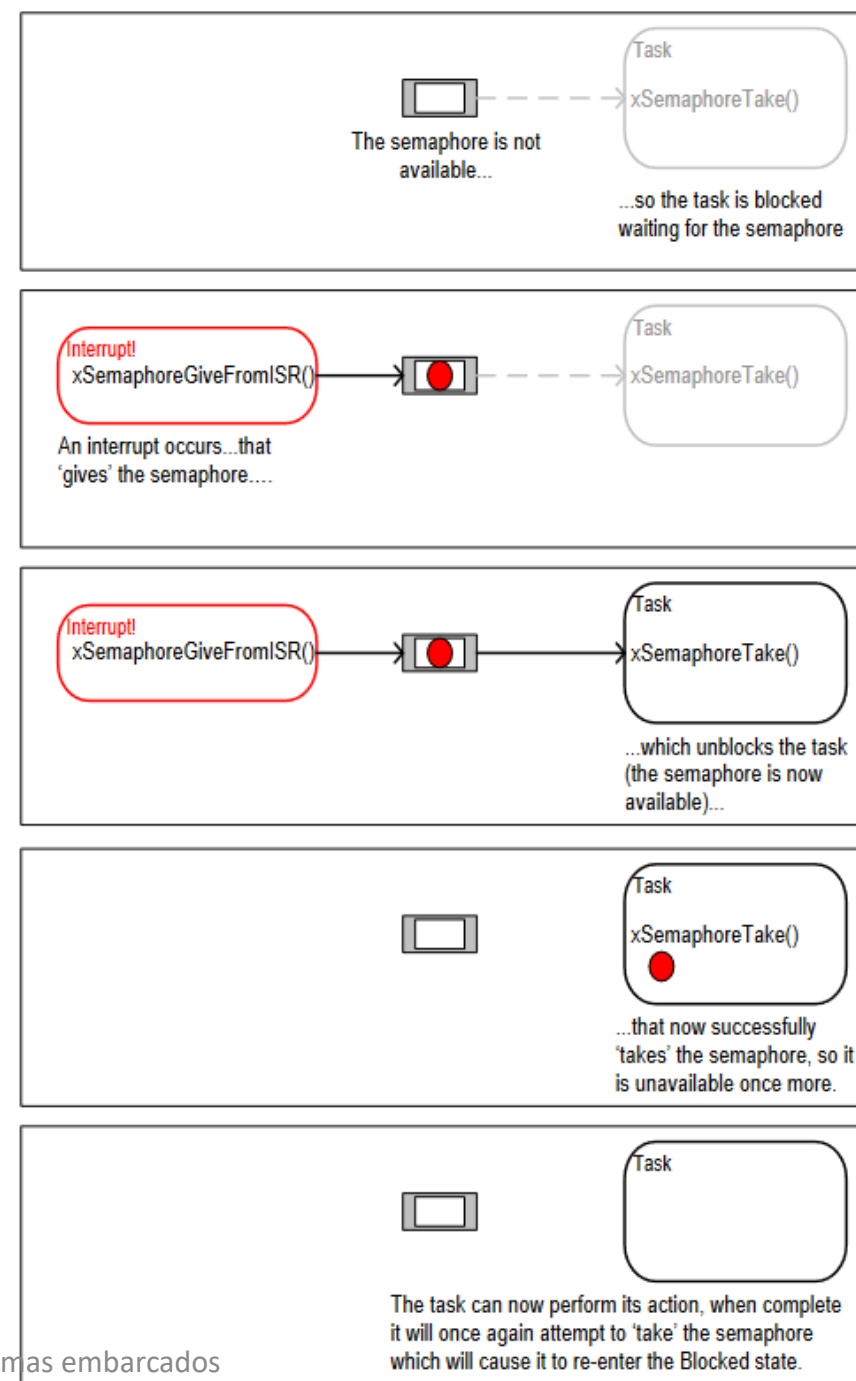


Figure 50. Using a binary semaphore to synchronize a task with an interrupt

Mas afinal uso mutexes ou semáforos binários?

Semáforos binários e mutexes são muito similares, mas tem algumas diferenças. Mutexes incluem um mecanismo de herança de prioridade, e semáforos binários não possuem. Isto torna os semáforos melhor para sincronização (entre tarefas e entre tarefas e interrupção), e mutexes a melhor escolha para implementar controle de região crítica (exclusão mútua)