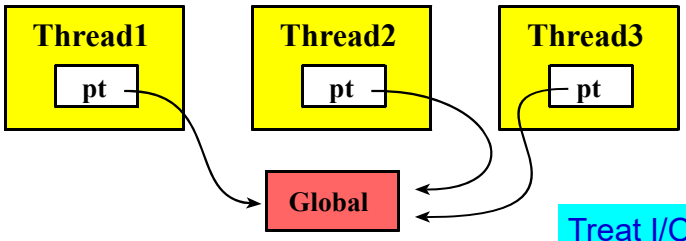
ECE445M/ECE380L.12
Embedded and Real-Time Systems/ Real-Time Operating Systems

Lecture 3:
Thread Communication & Synchronization

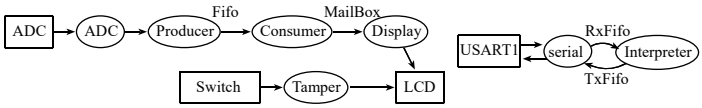
Thread Communication/Sharing



- Shared Globals
- Mailbox (Lab 2)
- FIFO queues (Lab 2)
- Message (Lab 6)

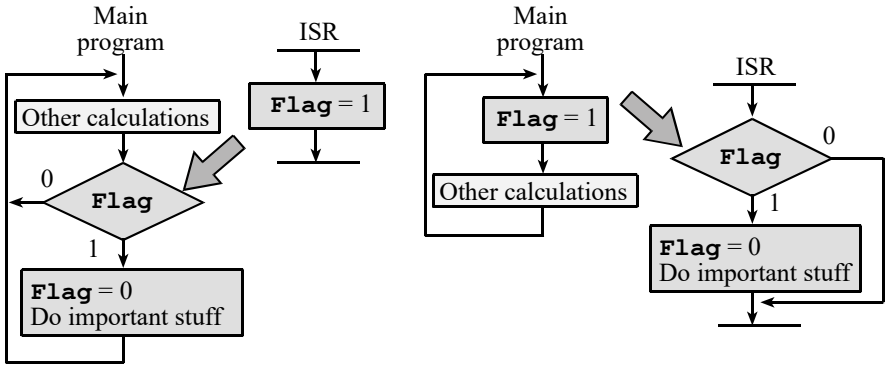
Treat I/O device registers like globals

Thread Communication

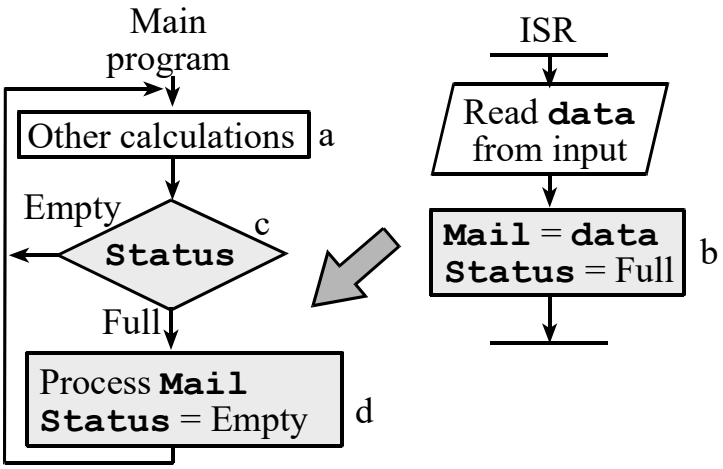


- Types
  - Data sharing (global variable)
  - Flag, Mailbox (one to one, unbuffered)
  - Pipes=FIFO (one to one, buffered, ordered)
  - Messages (many to many)
- Performance measures
  - Latency
  - Bandwidth
  - Error rate

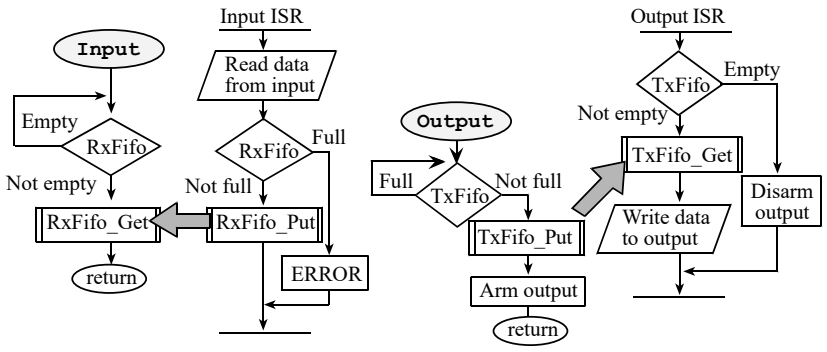
Flag



Mailbox



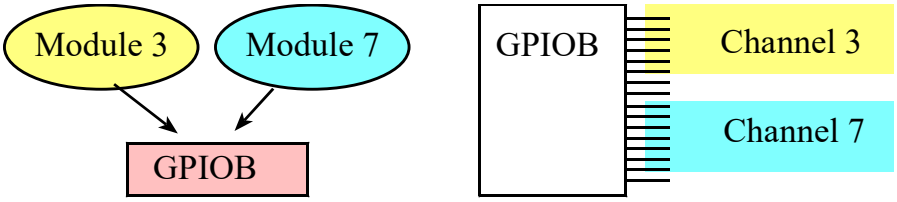
FIFO



UARTInts_4C123.zip

Race, Critical Section

- Two or more threads access the same global
  - Permanently allocated shared resource (memory, I/O port, …)
- At least one access is a write



Race Condition

- Timing bug
  - Result depends on the sequence of threads
    - E.g. two threads writing to the same global
- Hard to debug
  - Depends on specific order/interleaving
    - Non-deterministic (external events)
    - Hard to reproduce/stabilize (“Heisenbug”)
- Critical or non-critical
  - Final program output invalid?

Critical Section

- Non-atomic access sequence
  - Begins/ends with access to permanent resource
  - Involves at least one write
  - WR(+W), WW(+R/W), RR(+W)
- Load/store architecture
  - Read access creates two copies
    - Original copy in memory
    - Temporary copy in register
  - Write access changes official copy
  - Read-modify-write sequence: RMW(+W)

Thread-Safe, Reentrant

- Thread-safe code
  - No global resources
    - Variables in registers, stack
  - No critical section
    - No write access sequence
  - Mutual exclusion
    - Make accesses atomic (no preemption)
    - Prevent other threads from entering critical section
- Reentrant code
  - Multiple threads can (re-)enter same section
    - No non-atomic RMW, WW, WR sequence

Mutual Exclusion

- Disable all interrupts
  - Make atomic

Measure time with interrupts disabled
- Maximum time
- Total time

- Lock the scheduler
  - No other foreground threads can run
  - Background ISR will occur
- Mutex semaphore
  - Blocks other threads trying to access info
  - All nonrelated operations not delayed
    - Thread-safe, but not reentrant

Lecture 3

LDREX
STREX *Cortex-M3/M4F Instruction Set*, pg. 50

Thread Synchronization

- Sequential
- Rendezvous, Barrier
  - Fork/spawn & join
- Trigger, event flags
  - OR, AND
  - I/O event (e.g., I/O edge, RX, TX)
- Time
  - Periodic time triggered (e.g., TATOMIS)
  - Sleep

Lecture 3 J. Valvano, A. Gerstlauer ECE445M/ECE380L.12 12