

5. Threads

Chapter 5 objectives are to:

- Define a thread control block,
- Design and implement a preemptive thread scheduler,
- Design and implement spinlock semaphores,
- Design and implement block semaphores,
- Present applications that employ semaphores.

5.1. Multithreaded Preemptive Scheduler

thread
 execution of a software task
 has its own stack and registers
 light-weight process
 local variables are private
 threads cooperate to perform an overall function.

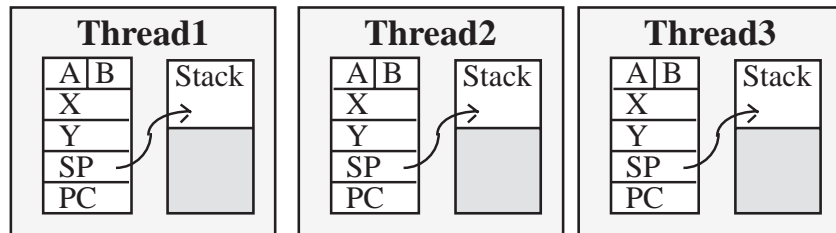


Figure 5.1. Each thread has its own registers and stack.

Since threads interact for a common goal, they do share resources such as global memory, and I/O devices.

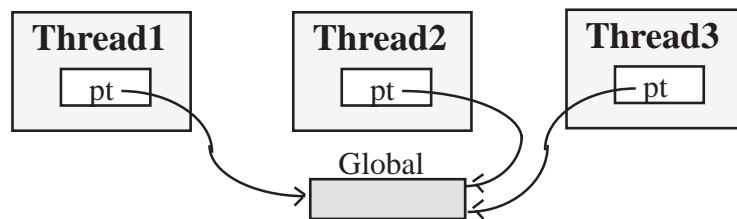


Figure 5.2. Threads share global memory and I/O ports.

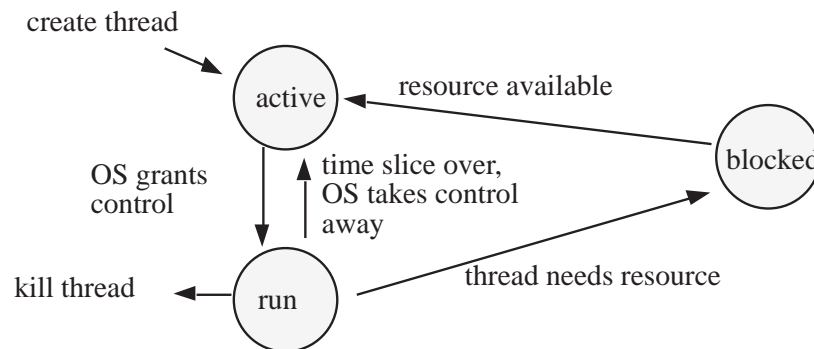


Figure 5.3. A thread can be in one of three states.

5.1.1. Round-Robin Scheduler

three statically-allocated threads

```
void main(void){
    PLL_Init();
    OS_AddThread(&Interpreter);
    OS_AddThread(&Consumer);
    OS_AddThread(&Math);
    SCI_Init(BR);
    OS_Launch(TIMESLICE); // doesn't return
}
```

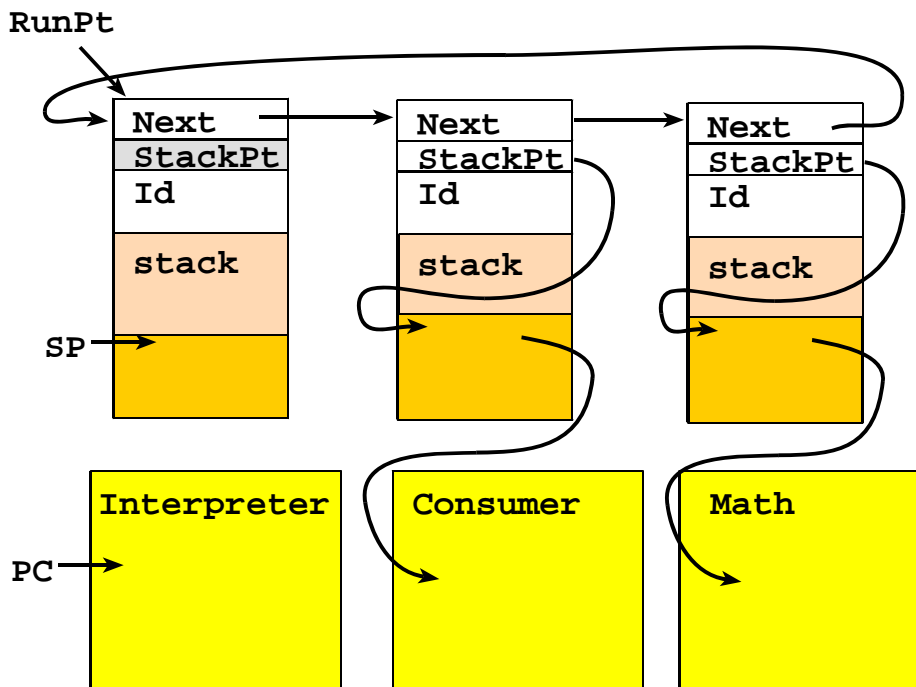


Figure 5.5. The circular linked list allows the scheduler to run all three threads equally.

Thread Control Block (TCB) information private to each thread.

- 1) a pointer so it can be chained into a linked list;
- 2) value of its stack pointer;
- 3) a stack area that includes local variables

While a thread is running, it uses the actual hardware registers, CCR,B,A,X,Y,PC,SP. In addition to these necessary components, the TCB might also contain:

- 4) Thread number, type, or name;
- 5) Age, or how long this thread has been active;
- 6) Priority;
- 7) Resources that this thread has been granted.

```

interrupt 11 void threadSwitchISR(void){
asm ldx RunPt
asm sts 2,x          // save StackPt
  RunPt = RunPt->Next;
  PTM = RunPt->Id;    // profiling
  TC3 = TCNT+8000;   // run for 1ms
  TFLG1 = 0x08;     // acknowledge
asm ldx RunPt
asm lds 2,x          // new SP
}

```

Output compare review

- 8 channels all using the same **TCNT**
- TCNT** rate controlled by **TMSK2**
- set **TIOS** bit to 1 for output compare
- set **TC3** to time for next interrupt

Output compare interrupt if

- Enabled, I=0 in the **CCR**
- Armed, C3I bit in **TIE** set
- Flag set, C3F bit in **TFLG1** set when **TCNT** equals **TC3**

Output compare interrupt acknowledge

- C3F cleared by writing a 1 to **TFLG1**

Open Lab17.c in Metrowerks

Show actual TCB, Thread switch software

Open Lab17.uc in TExaS

Run, and observe **PTM**, **PTT**

Interpreter thread (SCI input and SCI output)

I/O bound on typing rate

- t prints the time
- m prints the number of math calculations finished
- d prints the number of data points lost
- ? lists available commands

Producer/Consumer thread (ADC input, SCI output)

I/O bound on sampling rate

Real time data acquisition

Producer runs in background as a periodic OC interrupt

Consumer runs in the foreground calculating statistics

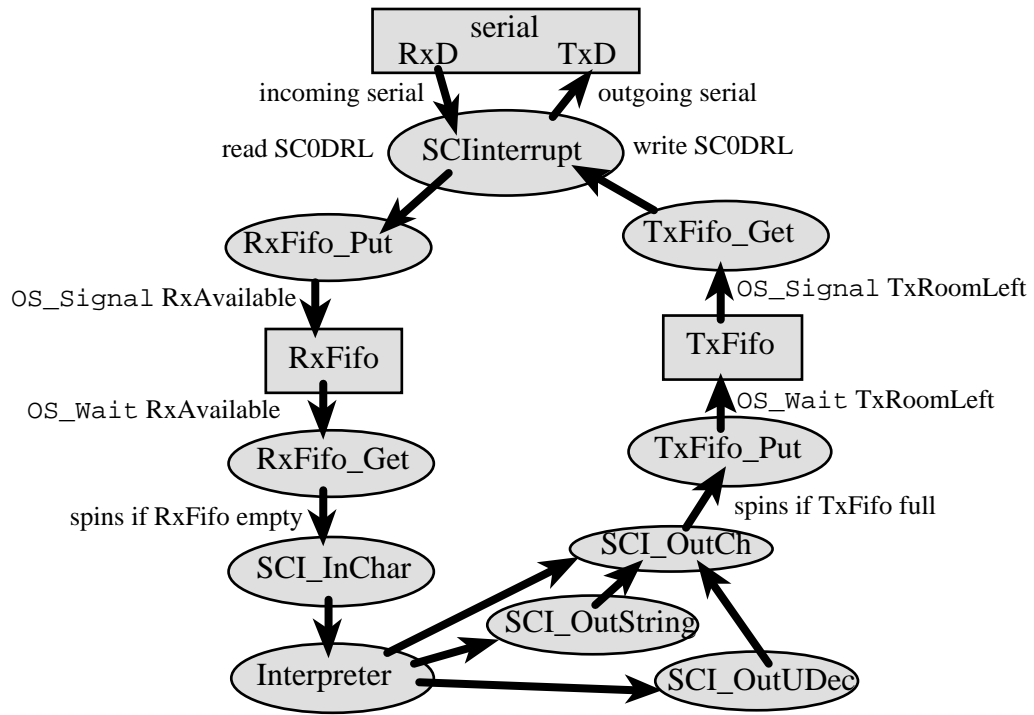
Consumer outputs results to SCI

DataFifo used to link background to foreground

Math thread (no input/output)

Calculates square roots

Draw call graph of Lab17



Data Flow graph of the SCI input/output for the interpreter