Announcements

- **Program #2**
  - On the web
Atomic Hardware

- **Atomic Instructions**
  - A single machine instruction
  - Executes without being stopped in the middle

- **Synchronization Instructions**
  - Ret = Test-and-set(m)
    - Rets gets the one bit value a memory location m
    - M is set to 1
  - Atomic-swap(a, b)
    - Temp <- a; a <- b; b <- temp;
    - a, b can be 1 bit, 8 bits, 16 bits, 32 bits, etc.
    - Often a is a register and b is a memory location
    - Emulate test-and-set with:
      - Reg = 1
      - Atomic-swap(reg, memAddress)
Implementing Semaphores

- **declaration**
  
  ```
  type semaphore = record
    value: integer = 1;
    L: FIFO list of process;
  end;
  ```

- **P(S):**
  
  ```
  S.value = S.value - 1
  if S.value < 0 then {
    add this process to S.L block;
  }
  ```

- **V(S):**
  
  ```
  S.value = S.value + 1
  if S.value <= 0 then {
    remove process P from S.L wakeup(P);
  }
  ```

*Can be neg, if so, indicates how many waiting*

*Bounded waiting!!*
Readers/Writers Problem

- **Data area shared by processors**
- **Some processes read data, others write data**
  - Any number of readers may simultaneously read the data
  - Only one writer at a time may write
  - If a writer is writing to the file, no reader may read it
- **Two of the possible approaches**
  - readers have priority or writers have priority
Readers have Priority

Semaphore wsem = 1, x = 1;
reader()
{
    repeat
        P(x);
        readcount = readcount + 1;
        if readcount = 1 then P (wsem);
        V(x);
        READUNIT;
        P(x);
        readcount = readcount - 1;
        if readcount = 0 V(wsem);
        V(x);
    forever
};

writer()
{
    repeat
        P(wsem);
        WRITEUNIT;
        V(wsem)
    forever
}
Comments on Reader Priority

- semaphores $x, wsem$ are initialized to 1
- note that readers have priority - a writer can gain access to the data only if there are no readers (i.e. when readcount is zero, signal(wsem) executes)
- possibility of starvation - writers may never gain access to data
Writers Have Priority

**reader**
repeat
  P(z);
  P(rsem);
  P(x);
    readcount++; if (readcount == 1) then
    readunit;
    V(x);
    V(rsem);
  V(z);
  readunit;
  P(x);
    readcount--; if readcount == 0 then
    V(rsem);
  V(x);
forever

**writer**
repeat
  P(y);
    writecount++:
      if writecount == 1 then
        P(rsem);
      V(y);
    P(wsem);
      writeunit
    V(wsem);
  P(y);
    writecount--;
      if (writecount == 0) then
        V(rsem);
      V(y);
forever;
Notes on readers/writers with writers getting priority

Semaphores $x, y, z, \text{wsem}, \text{rsem}$ are initialized to 1

Readers queue up on semaphore $z$; this way only a single reader
queues on $\text{rsem}$. When a writer
signals $\text{rsem}$, only a single
reader is allowed through

```plaintext
P(z);
P(\text{rsem});
P(x);
  readcount++;
  if (readcount==1) then
    P(\text{wsem});
V(x);
V(\text{rsem});
V(z);
```