COMPUTER ORGANIZATION (IS F242)

LECT 28: MIPS ARCHITECTURE

Synchronization

- Two processors sharing an area of memory
  - P1 writes, then P2 reads
  - Data race if P1 and P2 don't synchronize
    - Result depends on order of accesses
- Hardware support required
  - Atomic read/write memory operation
  - No other access to the location allowed between the read and write
- Could be a single instruction or an atomic pair of instructions

Synchronization in other architectures

- Single instruction to achieve synchronization
  - TestAndSet Instruction
    - Pass a memory address
    - Modify the Mem[passed address] to TRUE
    - Returns the Mem[passed address] (before modifying)
  - Swap Instruction
    - Pass two memory addresses
    - Swap the value of these two memory addresses

TestAndSet Instruction

- Definition:

```
boolean TestAndSet (boolean *target)
{
   boolean rv = *target;
  *target = TRUE;
   return rv:
}
```

- Solution: Shared boolean variable lock initialized to false.

```
do {
      while ( TestAndSet (&lock )) ;// do nothing
      //    critical section
      lock = FALSE;
     //      remainder section
      } while (TRUE);
```

Swap Instruction

- Definition:

```
void Swap (boolean *a, boolean *b)
{     boolean temp = *a;
       *a = *b;
       *b = temp:
}
```

- Shared Boolean variable lock initialized to FALSE; Each process has a local Boolean variable key
- Solution:

```
do {    key = TRUE;
        while ( key == TRUE)
              Swap (&lock, &key );
        //    critical section
         lock = FALSE;
       //      remainder section
 } while (TRUE);
```

Synchronization in MIPS

- Load linked: `ll rt, offset(rs)`
- Store conditional: `sc rt, offset(rs)`
  - Succeeds if location not changed since the `ll`
    - Returns 1 in rt
  - Fails if location is changed
    - Returns 0 in rt
- Example: atomic swap (to test/set lock variable)

```
try: add $t0,$zero,$s4 ;copy exchange value
     ll  $t1,0($s1)    ;load linked
     sc  $t0,0($s1)    ;store conditional
     beq $t0,$zero,try ;branch store fails
     add $s4,$zero,$t1 ;put load value in $s4
```

Assembler Pseudoinstructions

- Most assembler instructions represent machine instructions one-to-one
- Pseudoinstructions: figments of the assembler's imagination

```
move $t0, $t1     → add $t0, $zero, $t1
blt $t0, $t1, L   → slt $at, $t0, $t1
                    bne $at, $zero, L
```

  - $at (register 1): assembler temporary

Loading a Program

- Load from image file on disk into memory
  1. Read header to determine segment sizes
  2. Create virtual address space
  3. Copy text and initialized data into memory
     - Or set page table entries so they can be faulted in
  4. Set up arguments on stack
  5. Initialize registers (including $sp, $fp, $gp)
  6. Jump to startup routine
     - Copies arguments to $a0, … and calls main
     - When main returns, do exit syscall