Floating point operations in MIPS

32 separate single precision FP registers in MIPS

   f0, f1, f2, ... f31,

Can also be used as 16 double precision registers

   f0, f2, f4, f30 (f0 means f0, f1 f2 means f2, f3)

These reside in a coprocessor C1 in the same package

Operations supported

add.s   $f2, $f4, $f6      # f2 = f4 + f6 (single precision)
add.d   $f2, $f4, $f6      # f2 = f4 + f6 (double precision)

(Also subtract, multiply, divide format are similar)

lwc1    $f1, 100($s2)       # f1 = M [s2 + 100] (32-bit load)
mtc1    $t0, $f0            # f0 = t0 (move to coprocessor 1)
mfc1    $t1, $f1            # t1 = f1 (move from coprocessor 1)
Sample program

Evaluation of a Polynomial \( a.x^2 + b.x + c \)

```plaintext
# $f0 --- x
# $f2 --- sum of terms

# Evaluate the quadratic
l.s $f2,a  # sum = a
mul.s $f2,$f2,$f0  # sum = ax

l.s $f4,b  # get b
add.s $f2,$f2,$f4  # sum = ax + b
mul.s $f2,$f2,$f0  # sum = (ax+b)x = ax^2 + bx

l.s $f4,c  # get c
add.s $f2,$f2,$f4  # sum = ax^2 + bx + c

.data
a: .float 1.0
b: .float 1.0
c: .float 1.0
```

Pseudo-instruction
**Sequential Circuits**

The output depends not only on the current inputs, but also on the past values of the inputs.

![An SR Latch](image)

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>̅Q</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0/1</td>
<td>1/0</td>
<td>Old state continues</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Set state</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Reset state</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Illegal inputs</td>
</tr>
</tbody>
</table>
A clocked D-latch

Clock is the enabler. If C=0, Q remains unchanged.
When C=1, then Q acquires the value of D. We will use it as a building block of sequential circuits.
There are some shortcomings of this simple circuit. An edge-triggered circuit (or a master-slave circuit) solves this problem (to be discussed in the class)

**Master-Slave D flip-flop**
The output Q acquires the value of the input D, only when one complete clock pulse is applied to the clock input.

**Register**

A 8-bit register is an array of 8 D-flip-flops.
Abstract view of a register

**Binary counter**

Counts 0, 1, 2, 3, ...

A toggle flip-flop (T) is a modulo-2 counter

A 4-bit counter (mod-16 counter)

4-bit adder
Observe how Q3 Q2 Q1 Q0 change when pulses are applied to the clock input

**State diagram of a 4-bit counter**

Here state = Q3Q2Q1Q0

Recall that the program counter is a 32-bit counter

**A shift register**
Shift (right)

With each pulse on the shift input, data moves by pulse to the right.