Problem 1: Complete the design below. (Note: Sfifo = Searchable Fifo)

module mkBypassRFile(BypassRFile);
    RFile    rf <- mkRFile;
    SFifo#(1, RIdxData#(RIndx, Data))
            bypass <- mkBypassSFifo;
    rule move;
        begin let x = bypass.first;
            rf[x.index] <= x.data;
            bypass.deq end;

endrule

method Action wr(RIndx rindx, Data data);
    if (rindx!0) bypass.enq(
        RIdxData{index:rindx, data:data});
endmethod

method Data rd(RIndx rindx) =

    return (!bypass.search1(rindx)) ? rf.rd1(rindx)
        : bypass.read1(rindx);
endmodule

typedef struct {RIndx index; Data data}
RIdxData deriving (Bits);
**Problem 2:** Can any pipeline stages fire concurrently if the FIFOs do not permit concurrent enq and deq?

![Diagram of pipeline stages](image)

**Sol:** Alternate stages in the pipeline can still fire concurrently
Problem 3: When is this rule enabled?

```
rule sync-pipeline;
    if (inQ.notEmpty)
        begin
            sReg1 <= f0(inQ.first);
            inQ.deq;
            sReg1v <= Valid  
        end
    else
        begin
            sReg1v <= Invalid;
            sReg2 <= f1(sReg1);
            sReg2v <= sReg1v;
            if (sReg2v == Valid)
                outQ.enq(f2(sReg2));
        end
endrule
```

<table>
<thead>
<tr>
<th>inQ</th>
<th>sReg1v</th>
<th>sReg2v</th>
<th>outQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>NE</td>
<td>V</td>
<td>V</td>
<td>NF</td>
</tr>
<tr>
<td>NE</td>
<td>V</td>
<td>V</td>
<td>F</td>
</tr>
<tr>
<td>NE</td>
<td>V</td>
<td>I</td>
<td>NF</td>
</tr>
<tr>
<td>NE</td>
<td>V</td>
<td>I</td>
<td>F</td>
</tr>
<tr>
<td>NE</td>
<td>I</td>
<td>V</td>
<td>NF</td>
</tr>
<tr>
<td>NE</td>
<td>I</td>
<td>V</td>
<td>F</td>
</tr>
<tr>
<td>NE</td>
<td>I</td>
<td>I</td>
<td>NF</td>
</tr>
<tr>
<td>NE</td>
<td>I</td>
<td>I</td>
<td>F</td>
</tr>
</tbody>
</table>
Limitations of registers

- Using the register primitive no *communication* can take place in the same clock cycle between
  - two methods or
  - two rules or
  - a rule and a method

**EHRs to the rescue ...**

Ephemeral History Register (EHR): a primitive element to remedy this problem

---

**Ephemeral History Register (EHR)**
Dan Rosenband [MEMOCODE’04]

- $r[1]$ returns:
  - the current state if $w[0]$ is not enabled
  - the value being written if $w[0]$ is enabled
- $w[1]$ has higher priority than $w[0]$