Split/Join – Modifies the stack to support divergent control flow. When a warp diverges, entries in the stack track the control flow and the corresponding threads. In a divergence, a set of threads A takes the branch, and a complementary set of threads B falls through. At the point of reconvergence, all threads, C, resume normal execution.

When a split is encountered, entries corresponding to B and C and pushed to the stack. The current active set of threads are modified into A. When a join is reached, the entry corresponding to B are popped off the stack and executes from the instruction after the initial split. When a join is encountered the second time, the entry corresponding to C is popped and all threads resume execution at the instruction after join.

Example:

- No control flow – Warp execution is split into two groups, threads where pred=1 execute from _L1, then, and threads where pred=0 execute from L1. The join instruction is encountered twice, once per group of threads.

  ```
  @pred ? split
  _L1
  …
  join
  ```

- Control flow – conditional jump instruction must depend on same predicate register as split, or another predicate register with the same values across a warp. The group of threads that execute the jump instruction will eventually return to the join instruction

  ```
  @pred ? split
  …
  @pred ? jmpi OFFSET
  …
  join
  ```

- Nested split/join – after the initial split, one group of threads may encounter another split

  ```
  @pred ? split
  …
  @pred ? jmpi OFFSET
  …
  @pred2 ? split
  @pred2 ? jmpi OFFSET2
  …
  join
  ..
  join
  ```