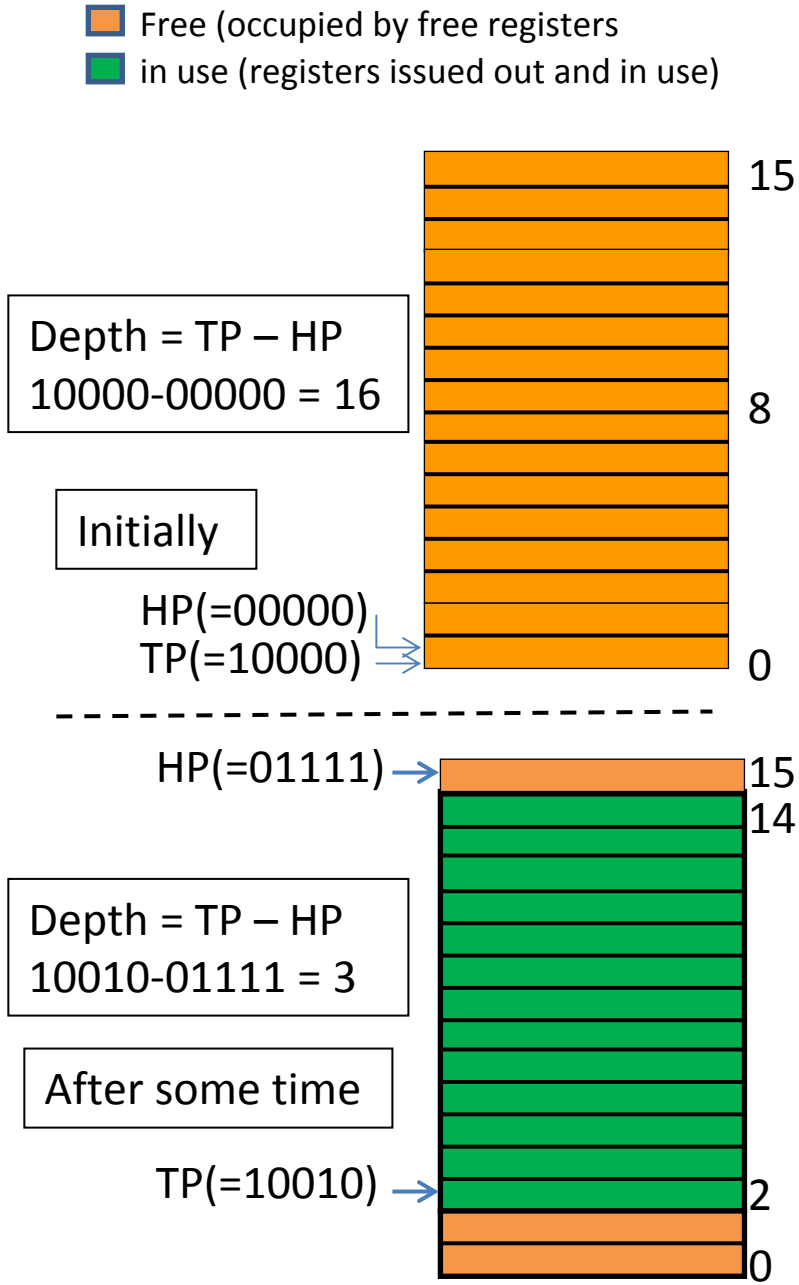
FREE REGISTER LIST

Free Register List

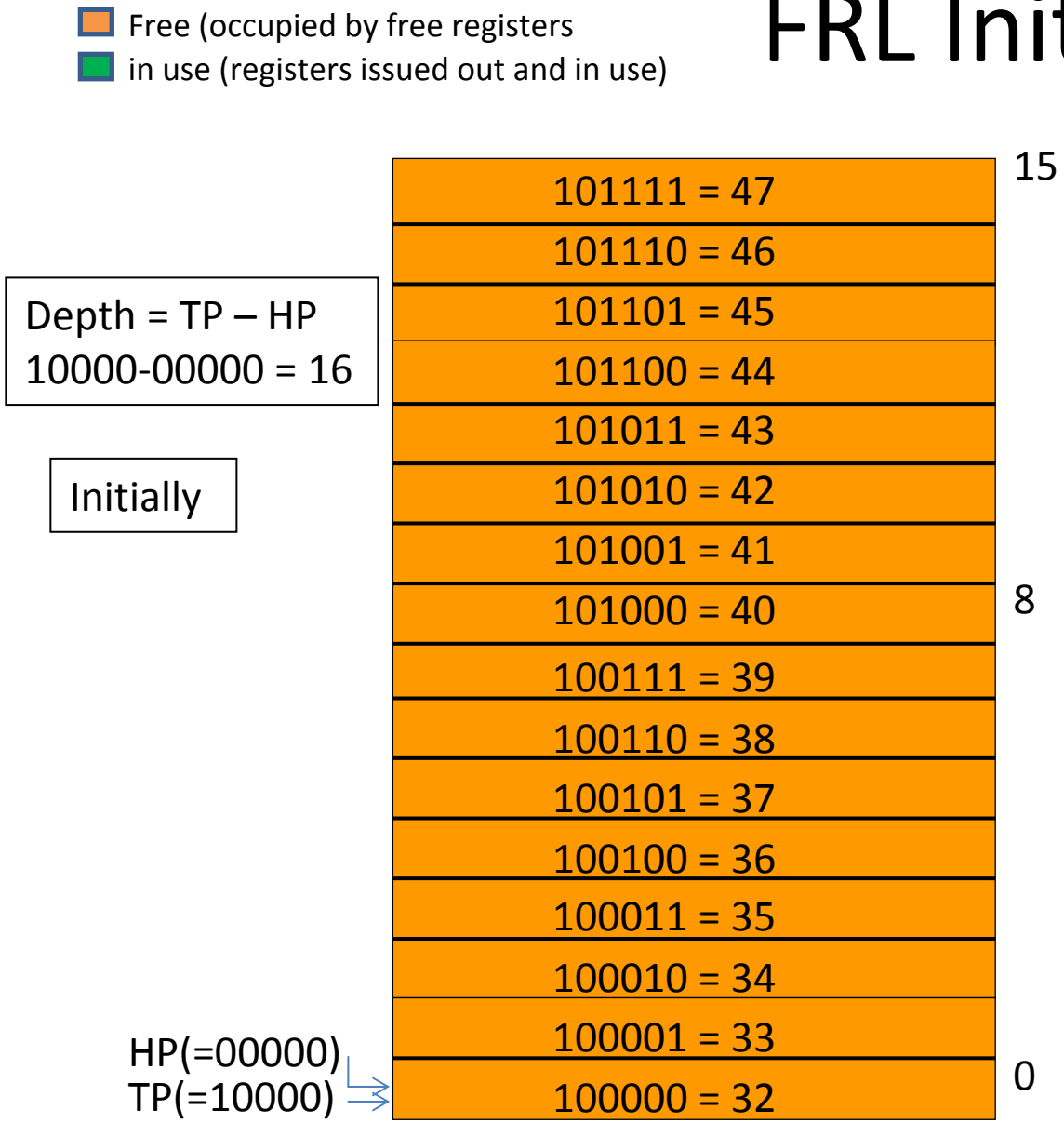
- The Free Register list has 16 locations each 6-bit wide and is a FIFO structure.
- The Read as well as Write pointers are 5-bit wide which can be realized as mod-32 counter.
- Initially the Read/Head pointer is set to 00000 where as the Write/Tail pointer is set to 10000.
- As in FIFO, we can use HP=TP to determine Empty signal and TP – HP = 10000 to infer FULL.

FRL



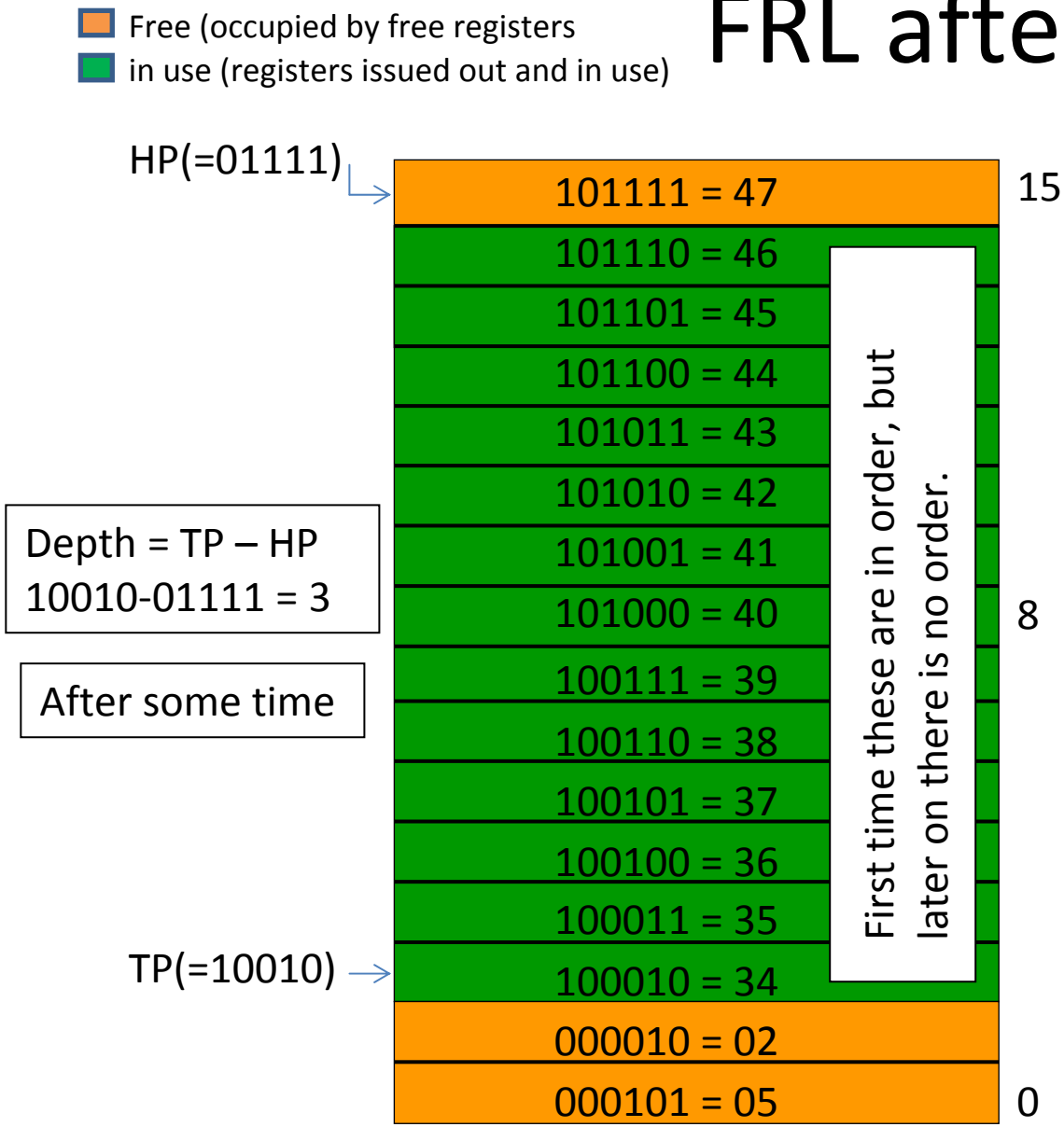
- FRL is a circular FIFO with 16 (=48-32) locations of 6 bit width referring to 16 of the 48 Physical Registers.
- Initially it is populated with 16 free registers (with PRF Ids 32 to 47 100000 to 101111). TP(WP) and HP(RP) are 5-bit pointers (not 4-bit). Initially, TP = 10000 and HP = 00000).
- As we dispatch, we use and assign registers from FRL by incrementing Head Pointer.
- As we Commit from Top of ROB, we free old (not new) physical registers to go back in FRL by incrementing Tail Pointer.

FRL Initial pre-fill



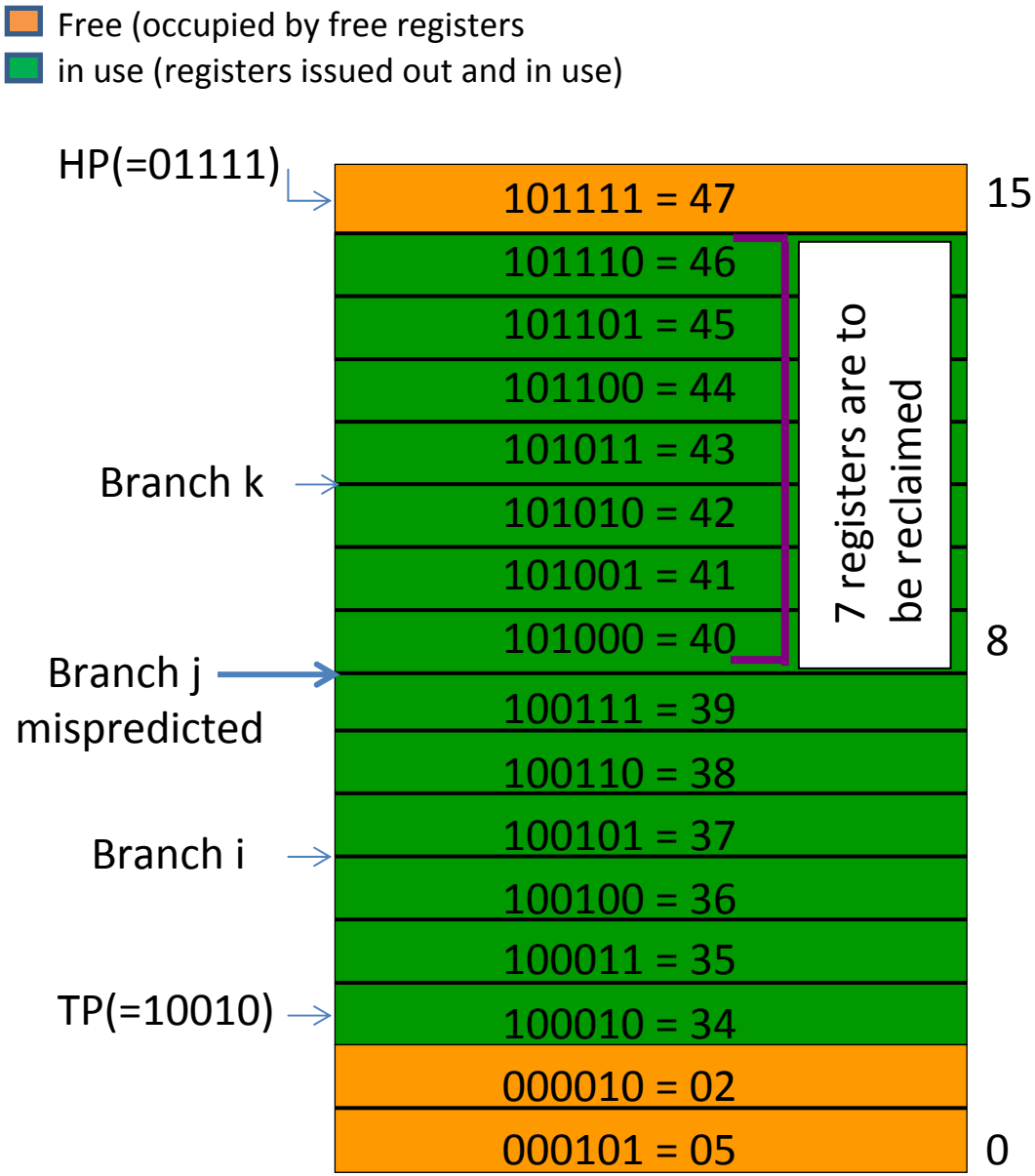
- Initial retirement mapping of architectural registers to physical registers is rather randomly chosen as $0 to $31 architectural to P0 to P31 physical. This is not necessary.
- The remaining free physical registers, 32 to 47, then shall be in FRL but can be placed in FRL in any order.
- We chose to fill them randomly in the order shown.

FRL after sometime



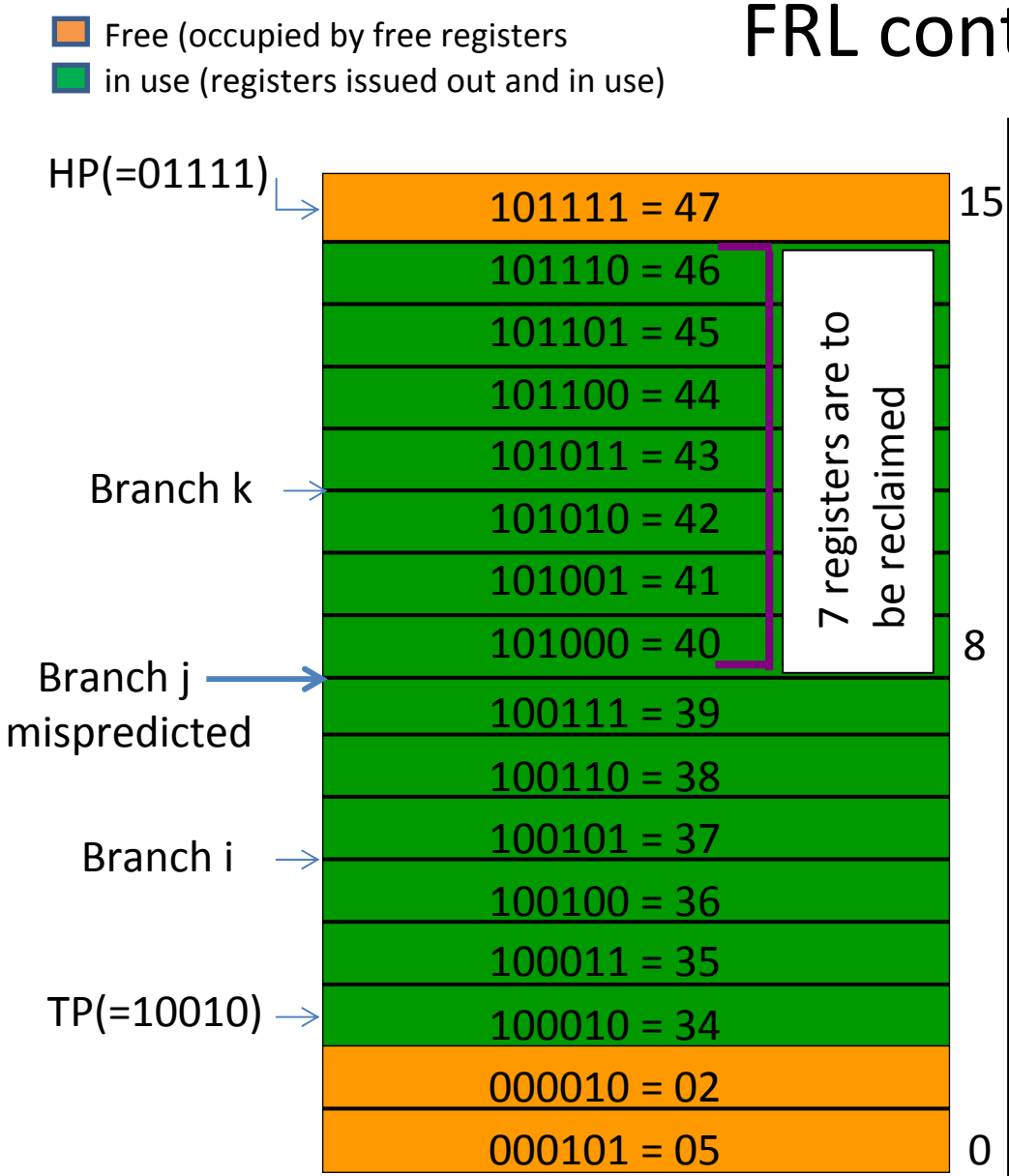
- 15 register writing instructions are dispatched
- Free physical registers, 32 through 46 are assigned to the destination registers of these 15 instructions.
- Hence HP is now pointing to location 15
- In the meanwhile the first two register-writing instructions have committed.
- They returned physical registers 5 and 2 as shown. These were the previous physical register mappings for the destination registers of the two instructions committed.

FRL restoring after misprediction



- Say branch j turns out to be mispredicted.
- Instructions, dispatched after this branch j, are called “wrong path instructions”. These need to be flushed.
- Seven physical registers have to be put back in the FRL.
- But we do not have 16 write ports on FRL to reclaim up to 16 registers.

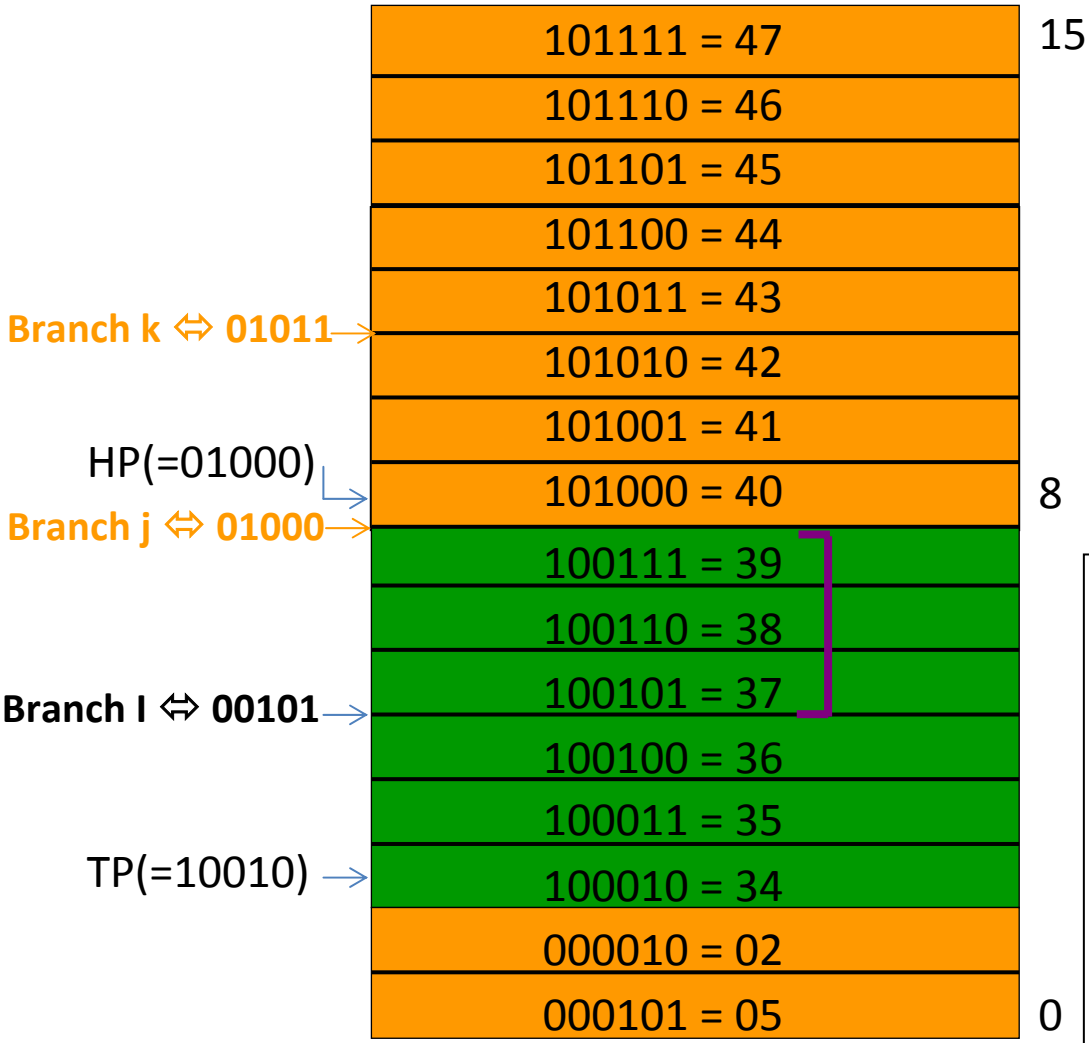
FRL contents remain in tact!



- After execution of a lot of instructions, the retirement RAT may be pointing to **any** subset of the 48 physical registers P0 to P47. The rest of the 16 **could be** in FRL.
- Out of the maximum 16 free registers, some can still be free in the FRL.
- The rest of the registers are allocated to the dispatched instructions. They were in FRL before allocation.
- Their IDs remain in tact in FRL as long as they are outstanding instructions in the backend.
- **So, reclaiming is easy. Just restore the HeadPointer HP!**

CFC maintains headpointer list!

Free (occupied by free registers
in use (registers issued out and in use)

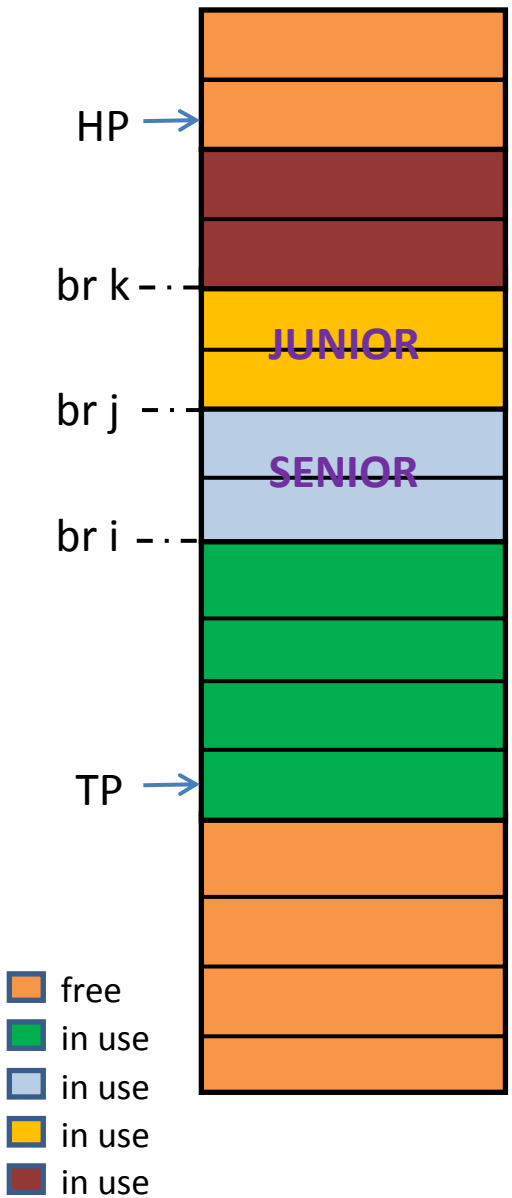


Branch ROB TAG	FRL Head Pointer
i	00101
j	01000
k	01011

These entries are also removed

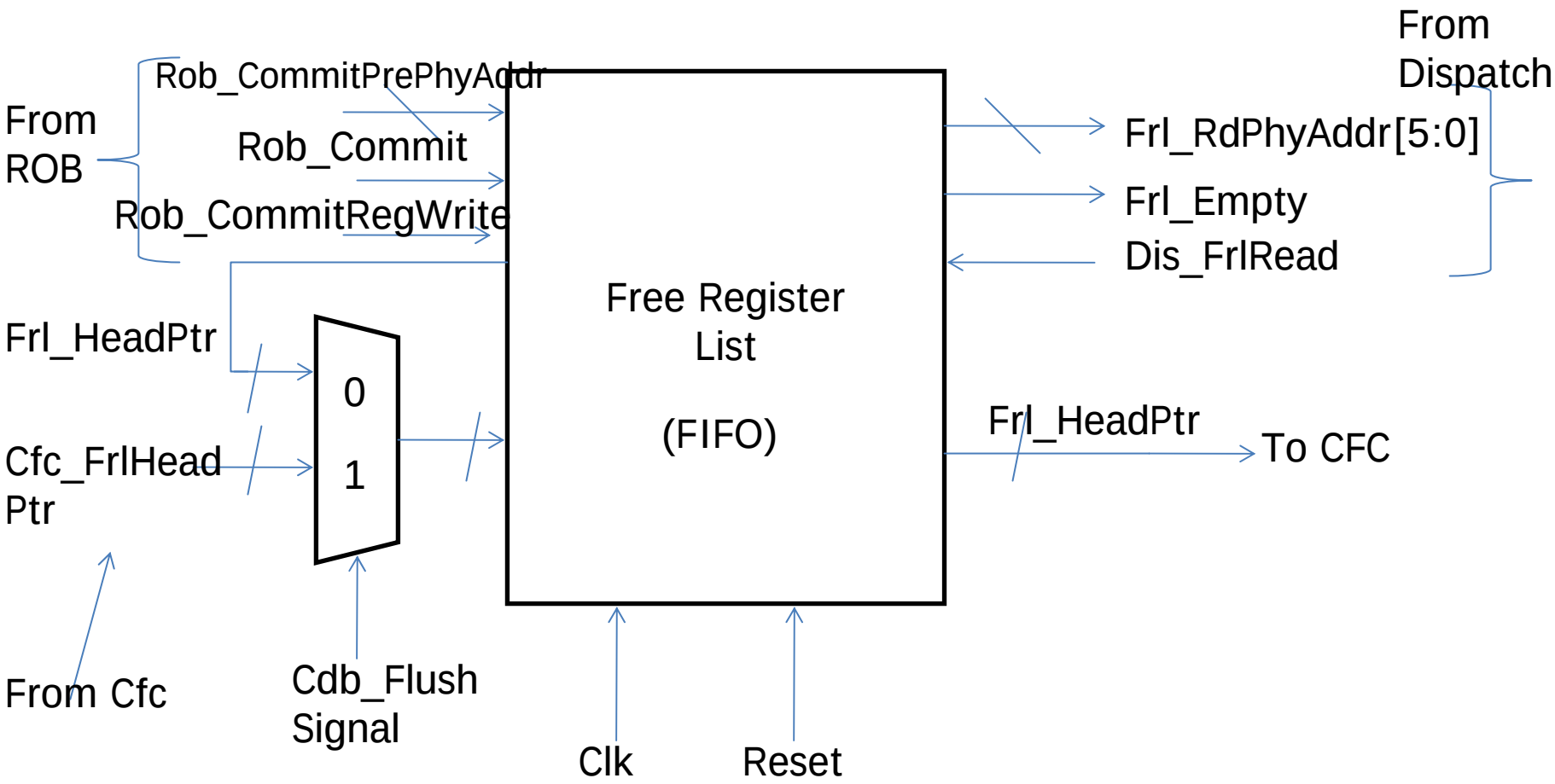
- **When branch j comes on CDB and announces "mispredicted", CDB looks up the above table gives FRL the head pointer value 01000. The FRL moves the head pointer as shown on the left side, there by reclaiming all 7 registers.**

FRL



- From the example on the side, if the branch j is mispredicted, then the head pointer which was stored (in the stack) corresponding to branch j is restored back into the head pointer. This is what we do in phase 1 of our design.
- In phase 2 of our design, some branches have check-points and some do not. If branch j is not check-pointed, we jump to the nearest check-pointed branch (in both ROB and FRL) and walk towards the mispredicted branch. We jump to branch i and walk forward (away from the top pointer in ROB emulating dispatch of intervening instructions and in FRL towards HP incrementing direction). Or jump to branch k and walk backward.
- While the above recovery is going on, the tail pointer can continue helping the registers which are freed on graduation of the instructions from the top of the ROB.

Free Register List



Free Register List

- FRL implementation does not require Almost Full and Full signals as we will not be *reclaiming* physical registers more than that are in-flight or in-use.

- FRL communicates with ROB as well as the Dispatch unit so as to determine whether the Head Pointer or Tail pointer or both need to be incremented or not.

- Signal **Dis_FrlRead** is used to increment the Read/Head pointer, where as the Signal **Rob_Commit** for a Reg write instruction i.e. **Rob_CommitRegWrite ='1'** is used to increment the Write/Tail Pointer.