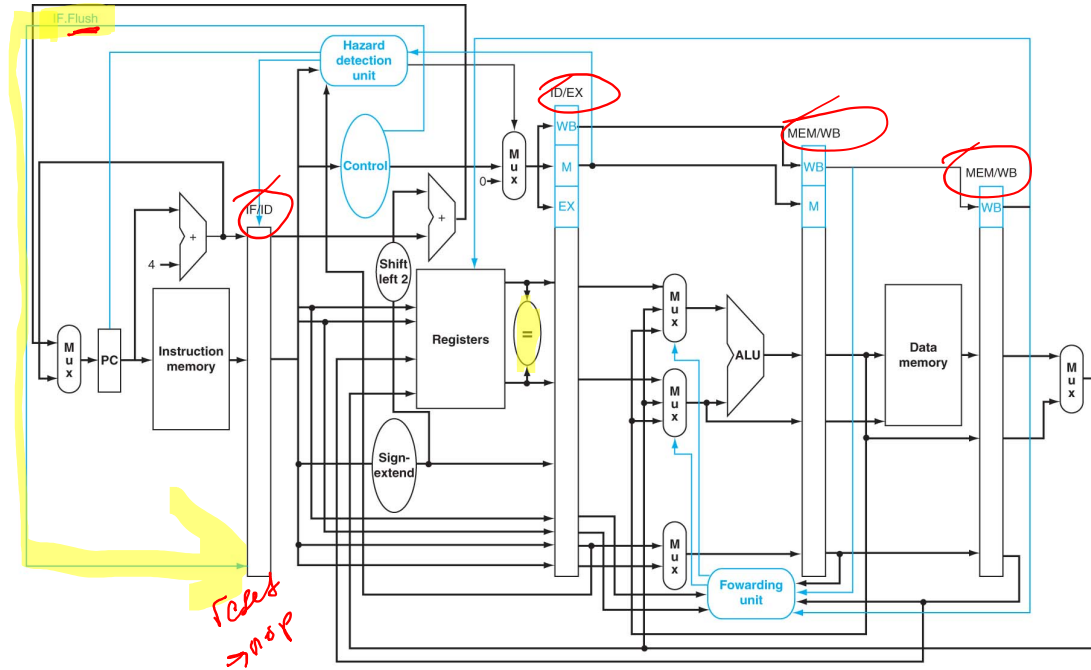


Announcements

- Midterm is next week **on Tuesday**
 - 90 minute slot (for an 80 minute exam)
 - Available 8am-8pm US/Pacific time
 - Need to start by 6:30pm!
 - Forward progress only!
 - Monday class will be a review session
 - There is also a review session hosted by Prof. Tullsen at 6:30pm tonight
- Midterm covers material through HW4
 - I.e. through data hazards (does not include control hazards, or prediction)
- Next week Wednesday is Veteran's Day (UCSD holiday)

The Pipeline with flushing for taken branches

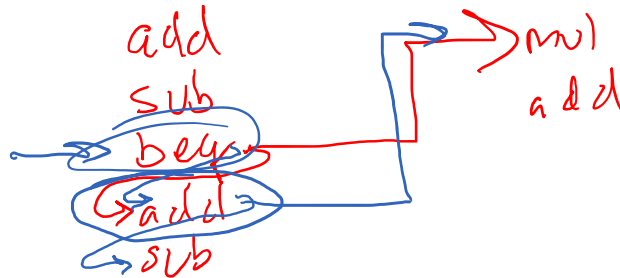


- Notice the IF/ID flush line added.

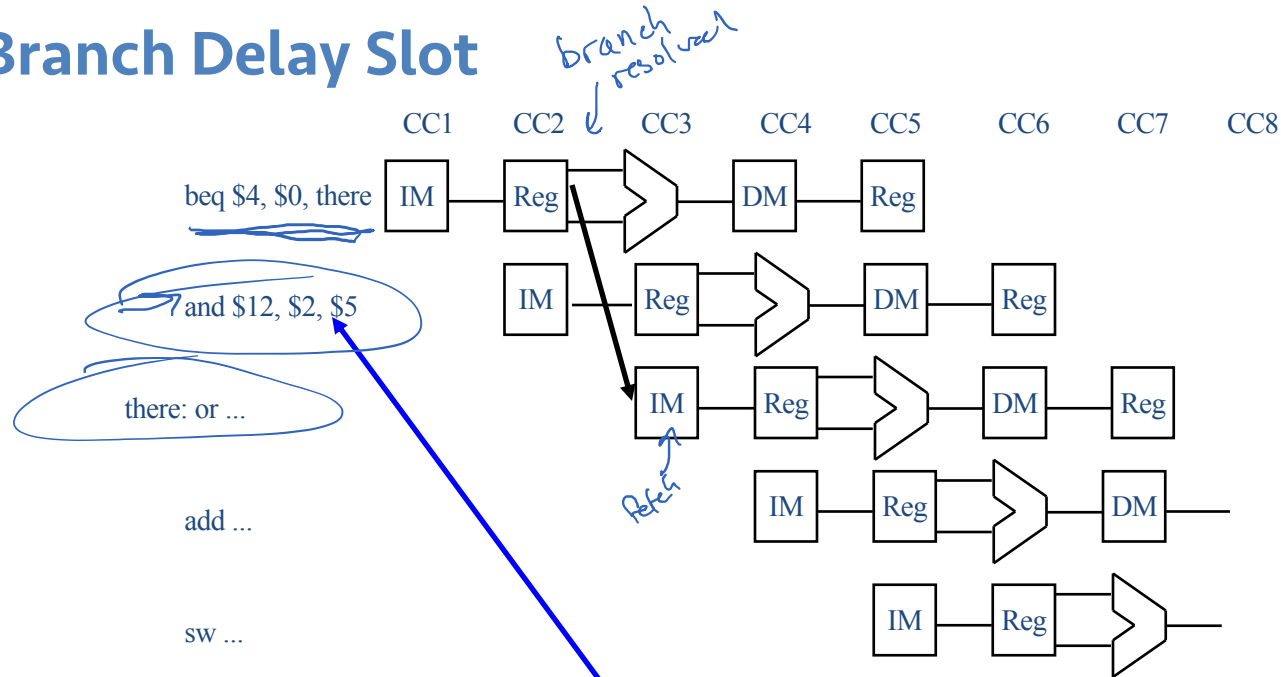
Eliminating the Branch Stall

A cute idea, but not one used by any modern core

- There's no rule that says we have to see the effect of the branch immediately. Why not wait an extra instruction before branching?
- The original SPARC and MIPS processors each used a single *branch delay slot* to eliminate single-cycle stalls after branches.
- The instruction after a conditional branch is *always executed* in those machines, regardless of whether the branch is taken or not!

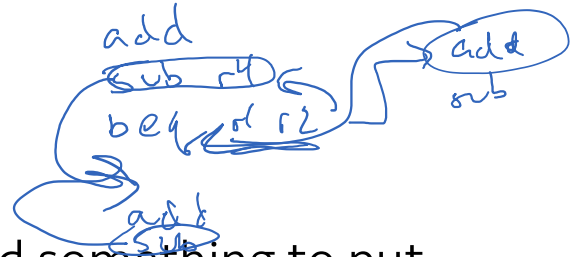


Branch Delay Slot



Branch delay slot instruction (next instruction after a branch) is executed even if the branch is taken.

Filling the branch delay slot



- The branch delay slot is only useful if you can find something to put there.
- If you can't find anything, you must put a nop to ensure correctness.
- Where do we find instructions to fill the branch delay slot?

after {
- not-taken
- taken
-

Branch Delay Slots

Branch Delay Slots

- This works great for this implementation of the architecture, but becomes a permanent part of the ISA.

Branch Delay Slots

- This works great for this implementation of the architecture, but becomes a permanent part of the ISA.
- What about the MIPS R10000, which has a 5-cycle branch penalty, and executes 4 instructions per cycle??

Branch Delay Slots

- This works great for this implementation of the architecture, but becomes a permanent part of the ISA.
- What about the MIPS R10000, which has a 5-cycle branch penalty, and executes 4 instructions per cycle??
- What about the Pentium 4, which has a 21-cycle branch penalty and executes up to 3 instructions per cycle???

Early resolution of branch + branch delay slot

- Worked well for MIPS R2000 (the 5-stage pipeline MIPS)
- Early resolution doesn't scale well to modern architectures
 - Better to always have execute happen in execute
 - Forwarding into branch instruction?
- Branch delay slot
 - Doesn't solve the problem in modern pipelines
 - Still in ISA, so have to make it work even though it doesn't provide any significant advantage.
 - Violates important general principal – (unless you really only want a single generation of your product) do not expose current technology limitations to the ISA.

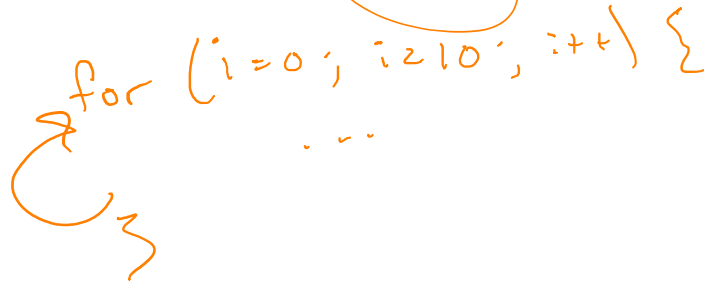
Okay, then...

- What do we do in modern architectures???

Branch Prediction

- Always assuming a branch is not taken is a crude form of *branch prediction*.
- What about loops that are *taken* 95% of the time?
 - we would like the option of assuming *not taken* for some branches, and assuming *taken* for others, depending on ???

for (i=0; i<10; i++) {
 ...
}

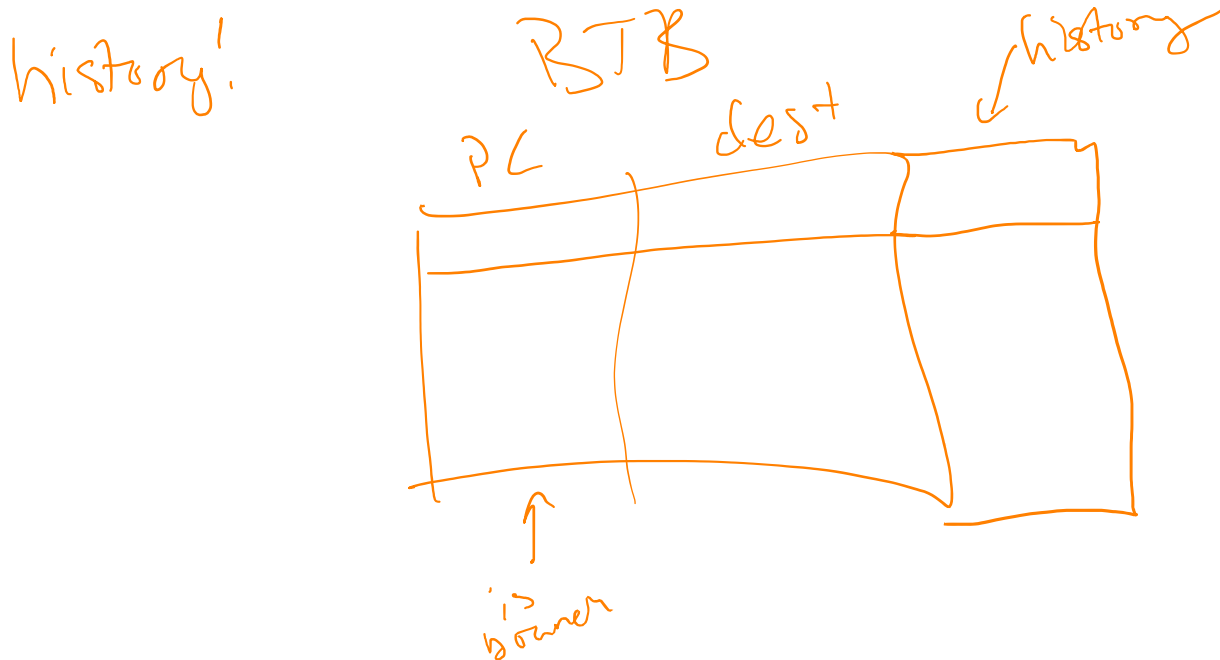


Branch Prediction

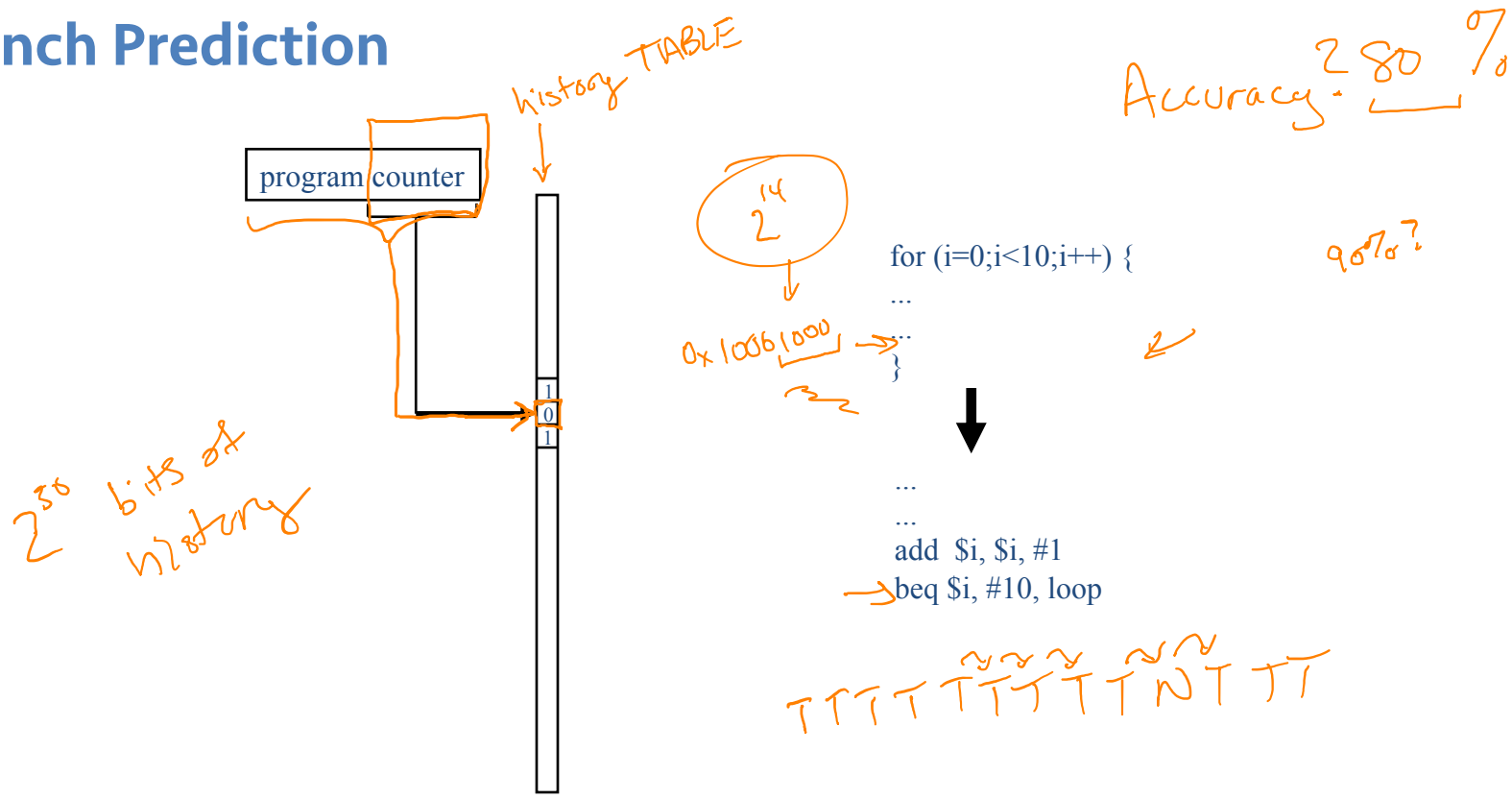
- Historically, two broad classes of branch predictors:
- Static predictors – for branch B, always make the same prediction.
- Dynamic predictors – for branch B, make a new prediction every time the branch is fetched.
- Tradeoffs?
- Modern CPUs all have sophisticated dynamic branch prediction.

Dynamic Branch Prediction

- What information is available to make an intelligent prediction?



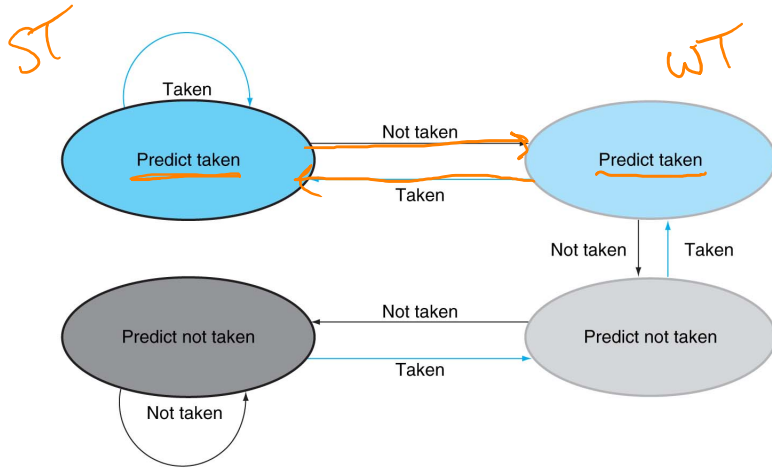
Branch Prediction



Two-bit predictors give better loop prediction

Acc 90%

TTTT N + TTT
↑ ↑
 ↓



```
for (i=0; i<10; i++) {  
  ...  
  ...  
}
```



```
...  
...  
add $i, $i, #1  
beq $i, #10, loop
```

This state machine also referred to as a *saturating counter* – it counts down (on *not takens*) to 00 or up (on *takens*) to 11, but does not wrap around.