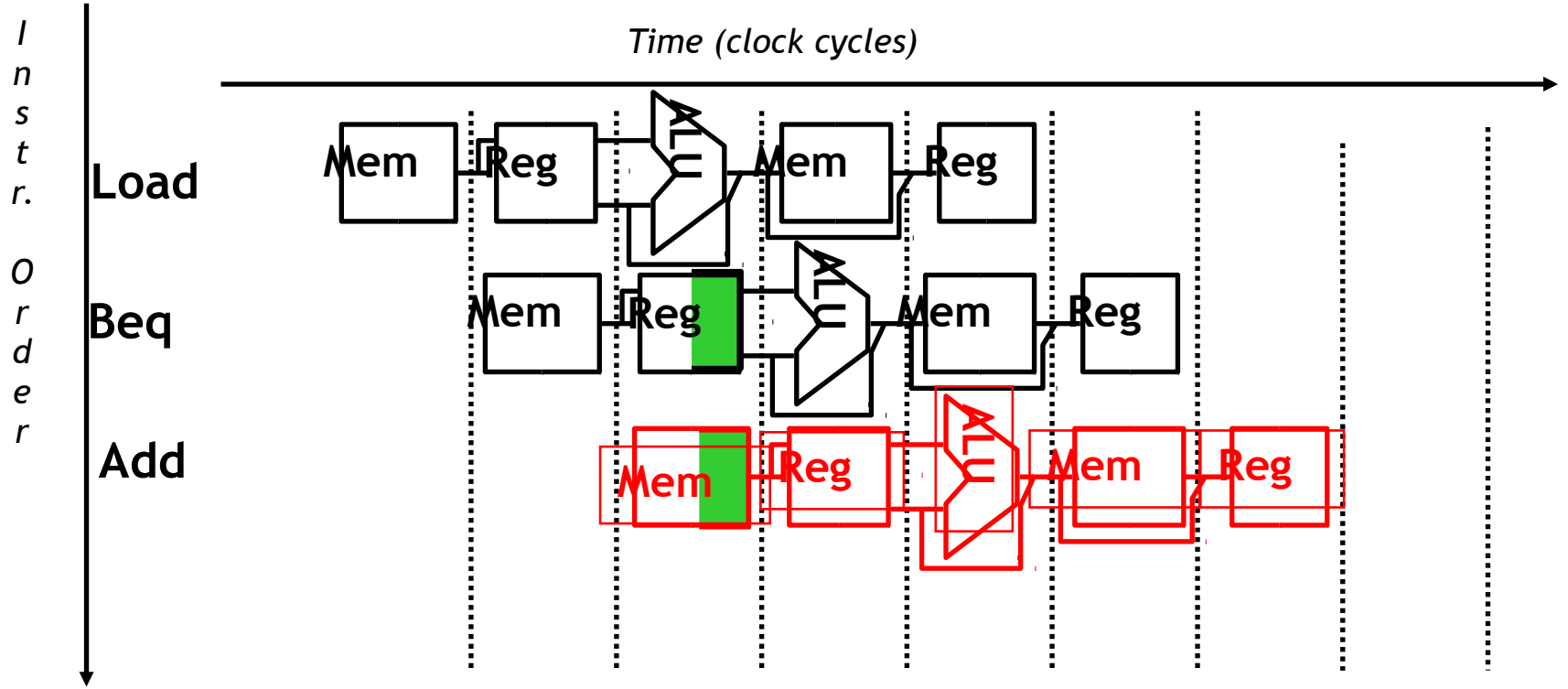
Control Hazards

Control hazards: attempt to make a decision before condition is evaluated

E.g., washing football uniforms and need to get proper detergent level; need to see after dryer before next load in

branch instructions

Control Hazards

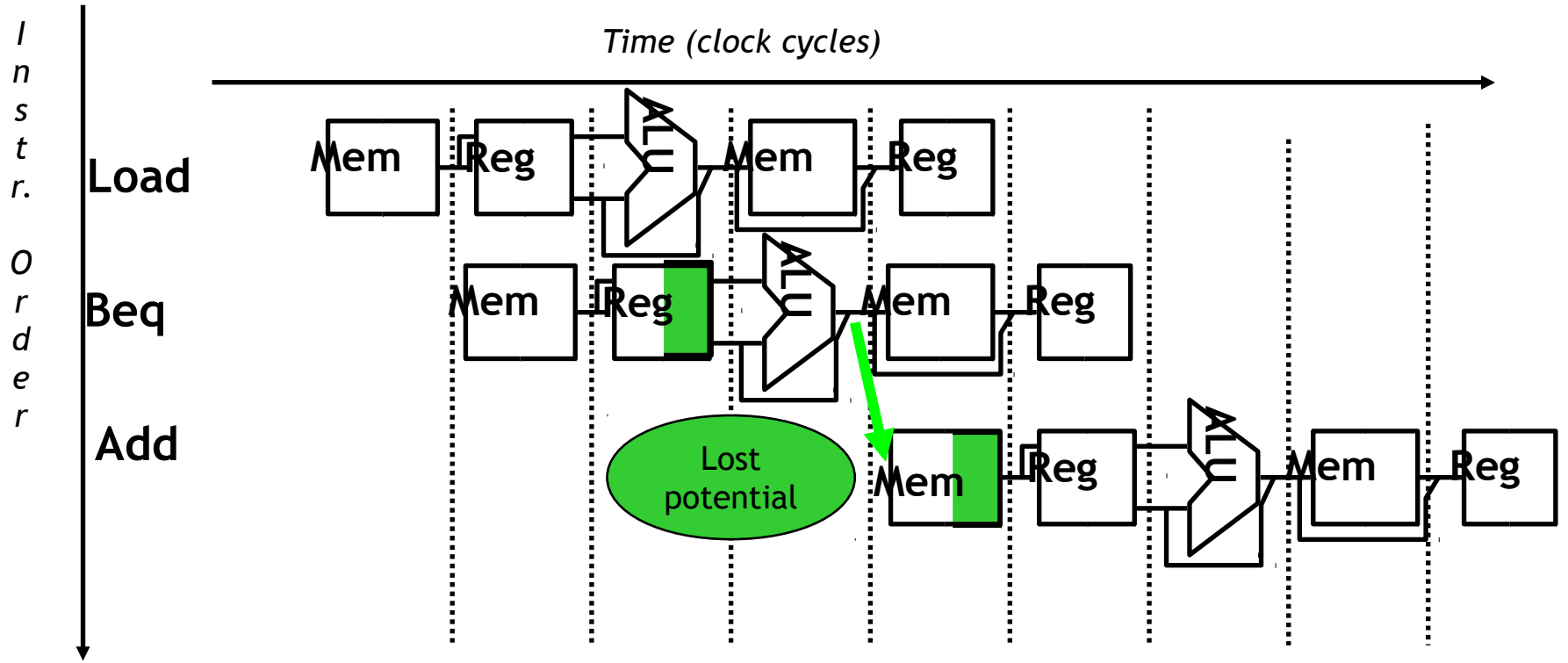


BEQ instruction: Decides which instruction to run next

We don't know the outcome of the BEQ until ALU is done

What do we fetch?

Control Hazard Solution #1: Stall



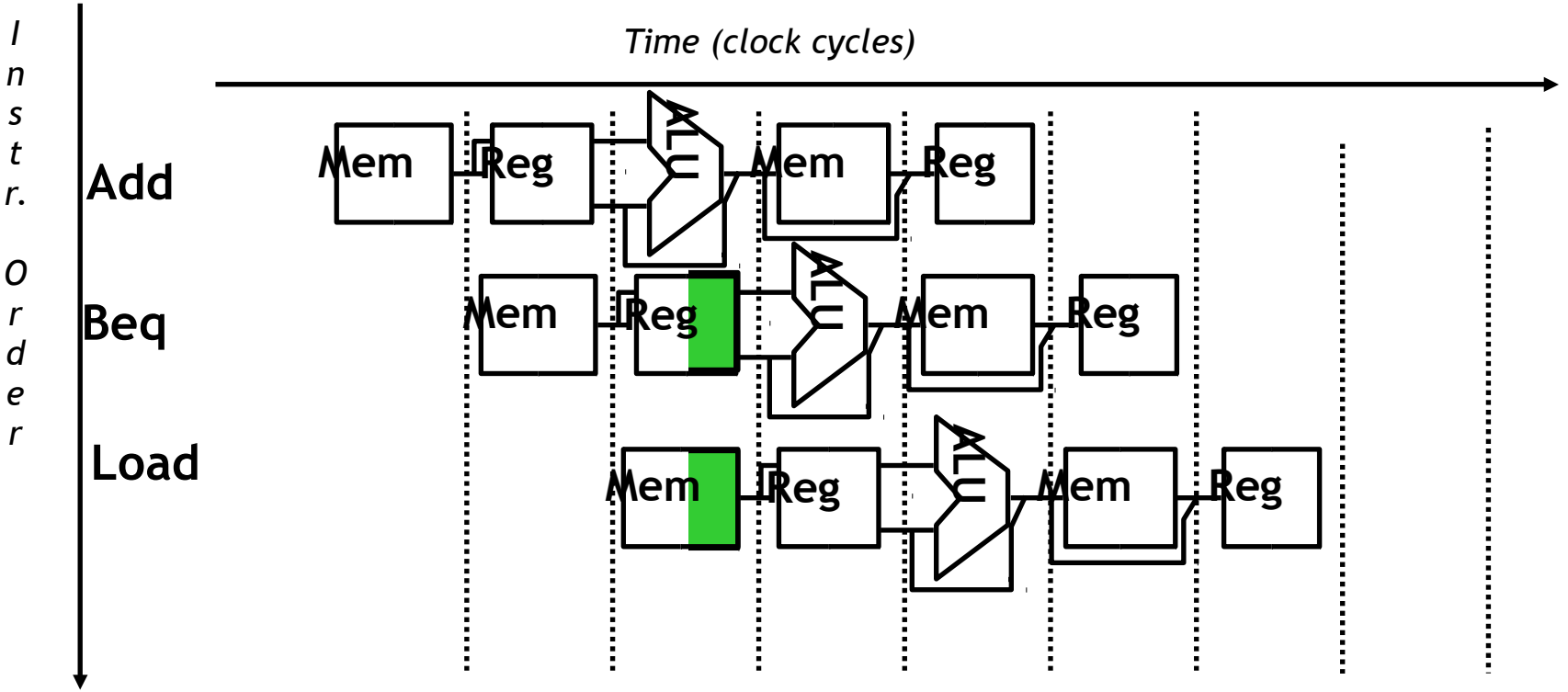
Stall: wait until decision is clear

Impact: 2 lost cycles (i.e. 3 clock cycles for Beq instruction above) => slow

Move decision to end of decode

save 1 cycle per branch, may stretch clock cycle

Control Hazard Solution #2: Predict



Predict: guess one direction then back up if wrong

Impact: 0 lost cycles per branch instruction if guess right, 1 if wrong (right ~ 50% of time)

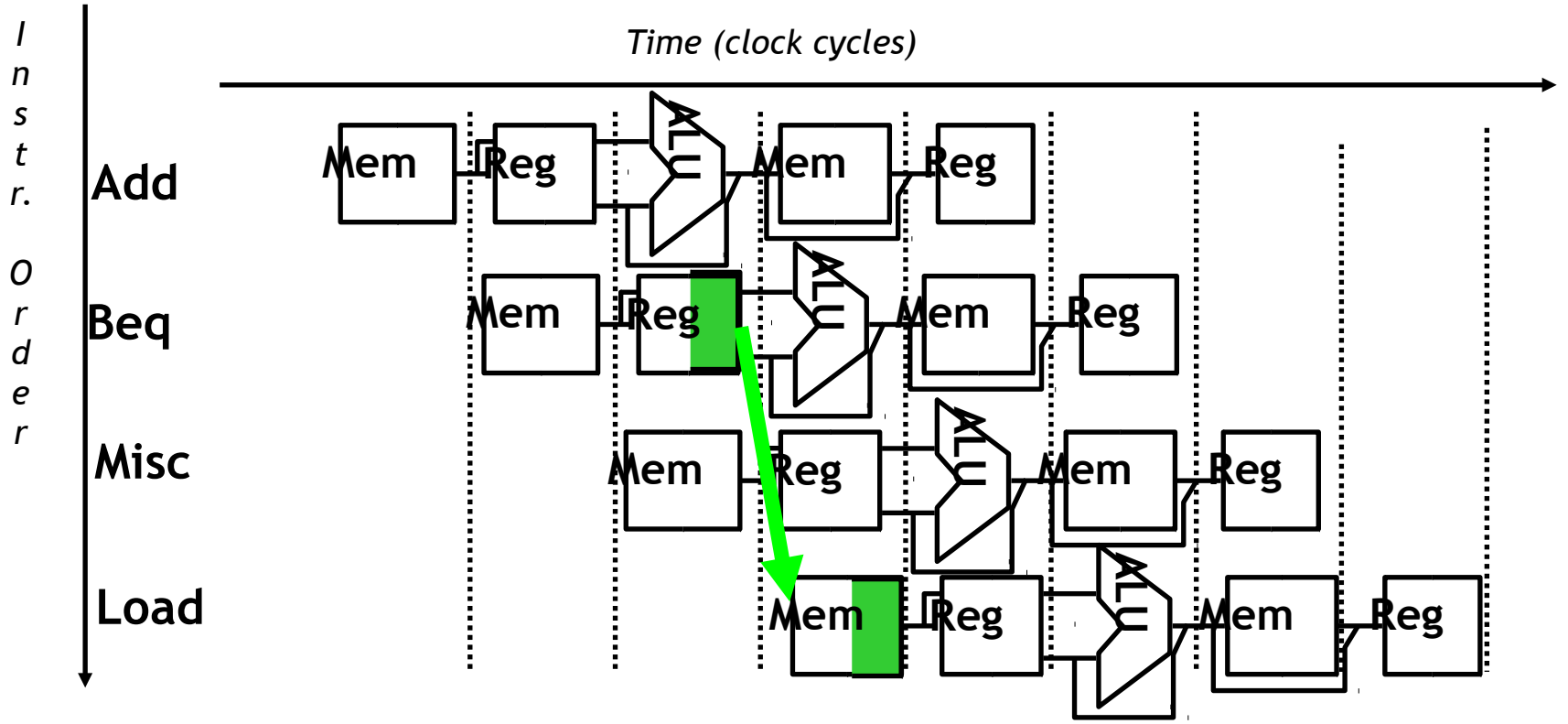
- Need to "Squash" and restart following instruction if wrong
- Produce CPI on branch of (1 *.5 + 2 * .5) = 1.5
- Total CPI might then be: 1.5 * .2 + 1 * .8 = 1.1 (20% branch)

More dynamic schemes: history of branch behavior (~90-99%)

Branch Prediction

- Goal: Predict which way the branch will go
  - Easiest: Forward branches not taken, backward branches taken (why?)
    - How does hardware know?
  - More complex: Keep track of previous behavior, predict based on that (> 90% accuracy)
  - Alternate: Let programmer (compiler) specify

Control Hazard Solution #3: Delayed Branch



Delayed Branch: Redefine branch behavior (takes place after next instruction)

Impact: 0 clock cycles per branch instruction if can find instruction to put in "slot" (~50% of time)

As launch more instruction per clock cycle, less useful

Loop Unrolling

```
for (i=0, s=0 ; i < 100 ; i++) {
     s += i*i; i++;
     s += i*i;
  }
```

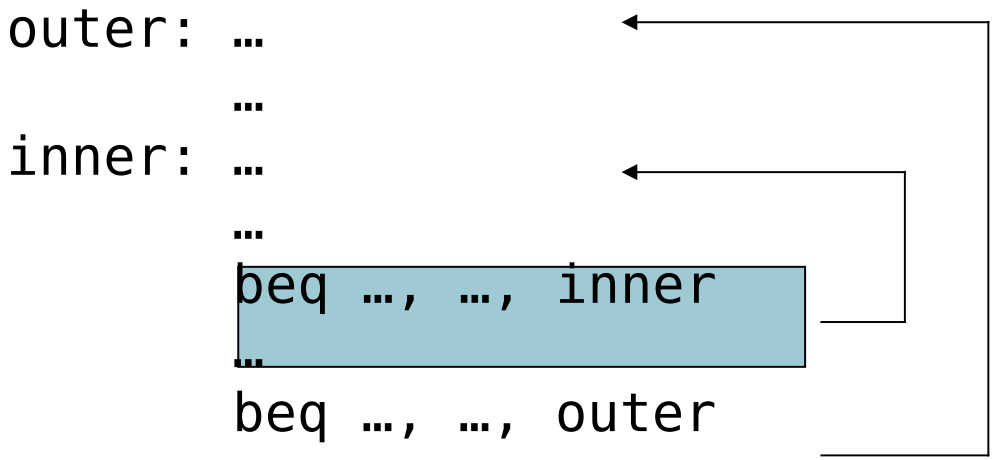
```
top:      mul temp, i, i   # pretend OK
          add s, s, temp
          addiu i, i, 1
          mul temp, i, i
          addiu i, i, 1
          sltiu b, i, 100
          beq b, 0, top
          add s, s, temp
```

How many cycles to run this? Where did we save?

8 cycles/iteration * 50 iterations = 400 cycles; CPI = 1

1-Bit Predictor: Shortcoming

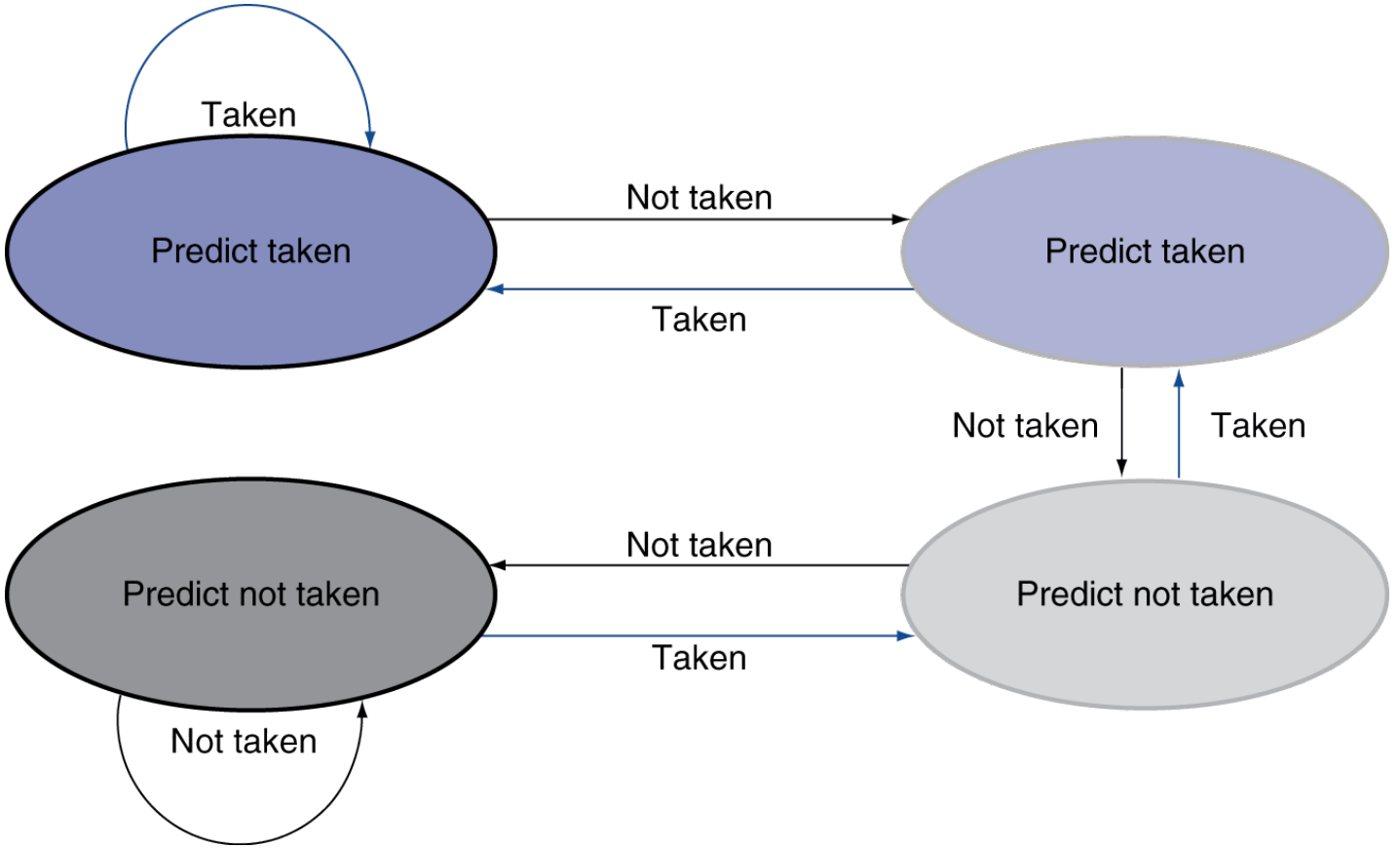
- Inner loop branches mispredicted twice!



  - Mispredict as taken on last iteration of inner loop
  - Then mispredict as not taken on first iteration of inner loop next time around

2-Bit Predictor

- Only change prediction on two successive mispredictions



Calculating the Branch Target

- Even with predictor, still need to calculate the target address
  - 1-cycle penalty for a taken branch
- Branch target buffer
  - Cache of target addresses
  - Indexed by PC when instruction fetched
    - If hit and instruction is branch predicted taken, can fetch target immediately