About branches

Architects favorite problem

We’re not finished with the problems!

- When we decide to branch, other instructions are in the pipeline!

Simple solution

- Stalling for branch hazards!
  
  - Seems wasteful, particularly when the branch is not taken!
  - Makes all branches cost 4 cycles. If 30% of program is composed of branches, what is the penalty?

  Previous cycles = 1 cycles + 5
  New Cycles = 0.75 \( I_{cycles} \) + 0.25 \( I_{cycles} \) + 4 + 5
  New Cycles = 1.00 \( I_{cycles} \) + 5

See why this problem is interesting? Think about a 10 stage pipeline...
**Assume branch not taken**

Works fine, IF...

```
Program execution order (in instructions):

1 line 1
2 and 3, 4, 5
3 or 6, 7, 8
5 add 9, 10, 11
5 to 8, 9, 10
```

**Assuming branch not taken...**

- Has the same performance as stalling when you are wrong
- Performance depends on percentage of time your guess is right

**Branch prediction**

- Always assume branch not taken is too crude:
  - digital filters (loops take 95% of time)
  - dictionary search (sometimes taken)
- for some we want to predict taken, for others, not taken!
Branch prediction

- What about using history?

Branch prediction buffer

- addi $2, $0, 10
- addi $1, $1, 1
- beq $1, $2, loop

How many times will we go the wrong way if the branch is taken 10 times?
90% branches taken, but only 80% prediction

2-bit prediction scheme

Still branches

- Modern processor with deep pipelines try to predict the destination of the jump
- If prediction is correct, pipeline maintains performance!
- What about a wrong prediction?
Flush the pipeline!

Improving performance
- There is no religious rule saying we must branch immediately.
  - Branch delay slots!
  - Always execute instruction after the branch, regardless if the branch is taken or not.
  - Together with comparisons in the decode stage, solves the problem!
  - Nowadays, difficult to find instructions to fill all those slots

Still branch delay slots
- Add a branch delay slot
  - the next instruction is always executed
  - rely on compiler to fill the slot with something useful
  - what about deeper pipelines?
- Superscalar machines: start more than one instruction in the same cycle
Dynamic scheduling

- The hardware performs the “scheduling”
  - hardware tries to find instructions to execute
  - out of order execution is possible
  - speculative execution and dynamic branch prediction
- All modern processors are very complicated
  - DEC Alpha 21264: 9 stage pipeline, 6 instruction issue
  - PowerPC and Pentium: branch history table
  - Compiler technology important