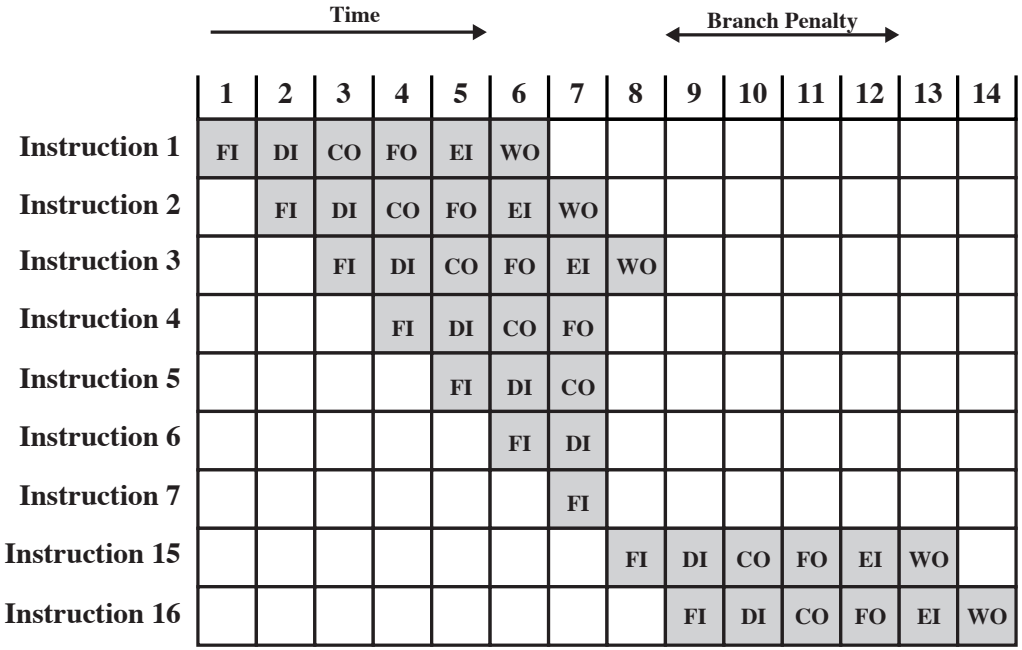
Preventing Stalls: 2

Branch or Control Hazards



A plot of the effect of a conditional branch at instruction 3 on the pipeline.
On cycle 8 when the address is known the pipeline is emptied (flushed).
Pipe starts refilling and there are 5 slots when no instruction completes

Mitigating Branch Hazards

Methods to mitigated the effect

- Multiple streams
- Prefetch branch target
- Loop buffer
- Branch Prediction
- Delayed Branch

Multiple streams

Duplicate the pipeline.
Process branch taken and branch not taken

If you have two pipelines, then they will both need access to registers, cache and memory.
Contention or increasing number of functional units

What happens when another branch enters one of the pipelines.

Using space for the units, and power for operation.

Can we get the same effect more efficiently

Prefetch target buffer

The branch target is fetched before it is clear that it is necessary.

Meanwhile the instruction following the branch enters the pipeline.

For branch not taken the pipeline operates at full efficiency.

If the branch is taken, then the branch instruction has already been fetched, reducing the length of the stall

Loop buffer

The *instruction fetch* part of the pipeline contains a small buffer of high speed memory which contains the last few instructions.

For branch taken the hardware takes the instruction from the buffer (if it is there) ie back to the start of the loop

- The loop buffer allows for pre-fetching instructions and thus there will be instructions in the buffer ahead of the current one. These are available with no memory access overhead.
- IF – THEN – ELSE constructs. Where the branch target may only be a few instructions ahead it will be pre-fetched
- If the buffer is large enough to contain the whole loop, then they will only be fetched once.

Called loop buffer, but with other advantages.

Delayed Branch

Address	Instruction
100	LDA
101	ADD
102	JMP 106
103	MUL

Unoptimised

Here 103 will be in the pipeline and the pipeline will need to be cleared before proceeding. Adding to the complication of the circuit

Address	Instruction
100	LDA
101	ADD
102	JMP 106
103	NOOP
104	MUL

NOOP

Here the addition of a NOOP instruction means that the pipeline does not need to be cleared

Pipe Effic

Delayed Branch

Address	Instruction	
100	LDA	
101	JMP	106
102	ADD	
103	MUL	

Here the branch command is pulled back and executed earlier.
The command which precedes the jump is now after the the jump.

It can finish executing as the jump command is executed and the command following the jump is being fetched.

This must be done more carefully if the branch is conditional.

The execution following the branch is referred to as the delay slot

Delay slot

Branch Prediction

This means saying ahead of time whether a branch will be taken or not.

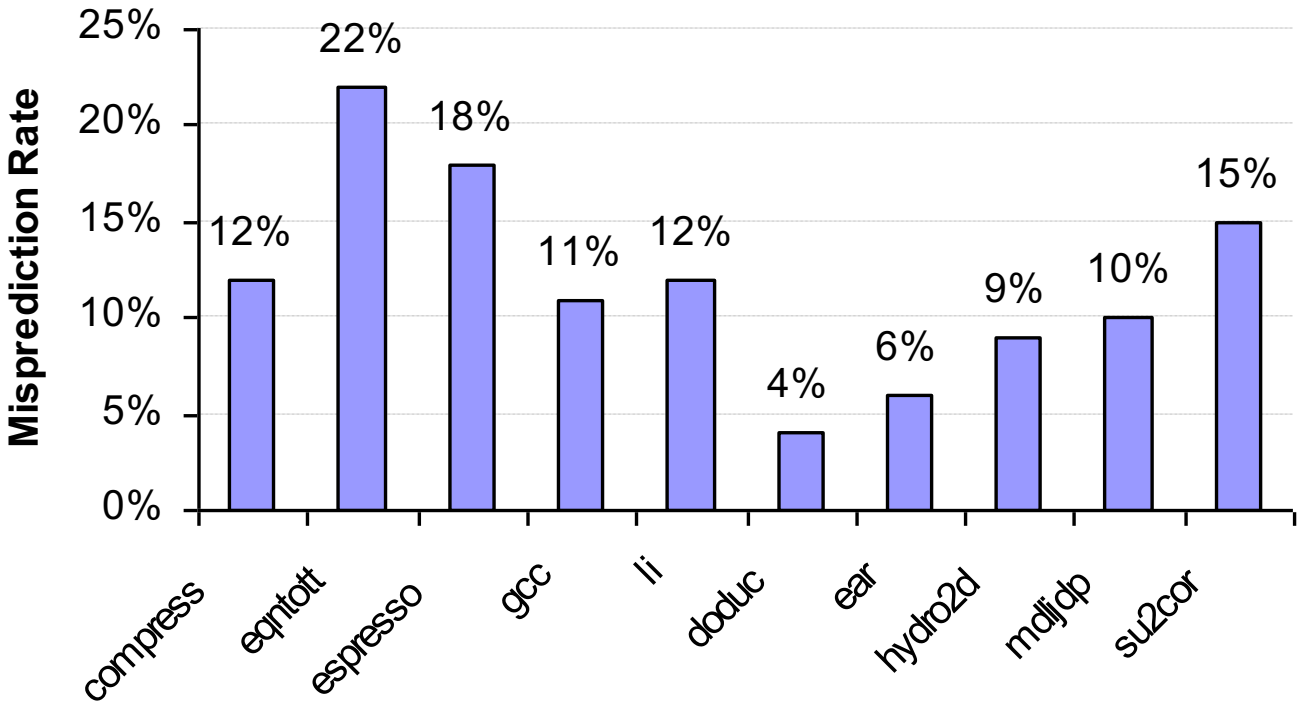
Turns out to be possible to guess the correct answer a significant percentage of the time.

If we consider a loop …

The branch at the end of the loop is taken every time that the system executes the loop.
Only at the end of the loop is the branch not taken.

Branch Prediction

Reduces stalls, only if the prediction is correct.
Simplest is predict branch taken



Some spec programs and failure rate for branch taken. – static, compile time

Can collect data during execution and modify prediction.

Dynamic Branch Prediction

Depends on underlying regularity of code and data.
Dynamic is better than static.

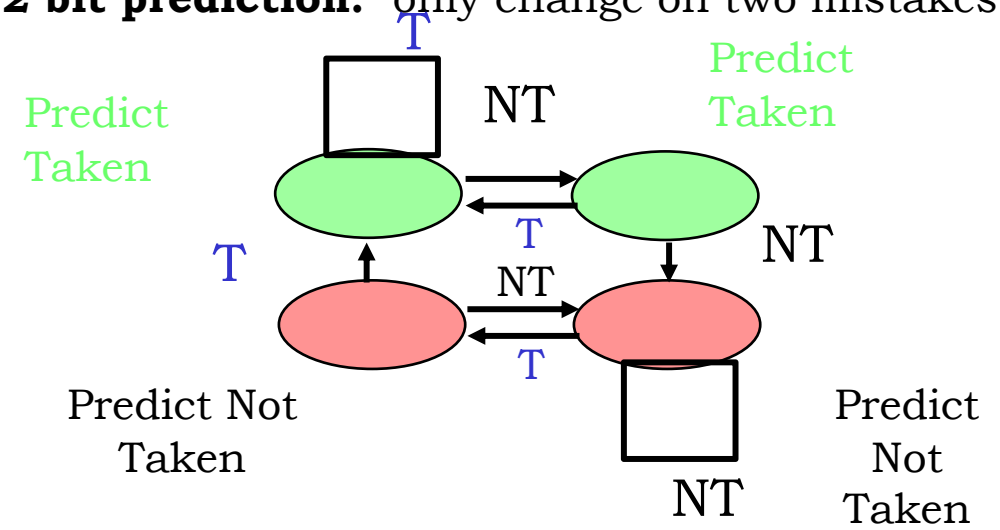
Performance is a function of the accuracy and the cost of starting down the wrong route.

Weather tomorrow, same as today.

Simplest is a Branch History Table (BHT) – do the same as last time. 1bit

A loop will have 2 mis-directions – first and last
Average is only 9 (seems small??)

2 bit prediction: only change on two mistakes



Finite state machine

Return address predictors

One target for improvement is predicting indirect jumps, whose address varies at run time.

The return address from a procedure/method accounts for the vast majority of indirect jumps.

Spec95 benchmarks show 15% of branches are returns. Java and other OO languages use methods and shorter code runs. Even more advantage in optimising method returns.

BTB will nearly always get this wrong – calls from a single site typically not temporally clustered. (Repeated calls from a loop may be)

Solution : return address stack.
Push on call
Pop on recall.

Approaching 100% accuracy depending on call depth and stack size.

Branch History Table

Branch history table: small cache associated with the instruction fetch.

Each entry contains the address of the branch instruction; history bit(s) recording the state of that instruction; information about the target instruction – either the target address or the actual instruction.

It means a target instruction can be retrieved without memory access and the decision on whether to take or not to take the branch is also rapidly available