Branching

Branches, being the most serious interruption to pipeline processing, have been studied extensively.

There are many mechanisms to reduce the affects of branching, since by assumption conditional execution and flow control are inevitable.

Branching elimination -- provide conditional operations.

Branch speedup -- reduce time for CC set or TIF.

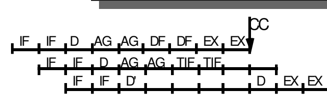
Branch prediction -- guess which way the branch goes.

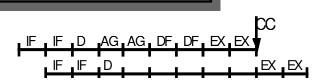
Branch target capture -- save the targets in a cache.

Branch Elimination

- Compilers emit a lot of short code sequences -- 23% of all runs are 1 or two instructions long.
- Many of these sequences are branching around instructions that are to be conditionally executed.
- Conditional instructions assist by making execution of an instruction dependent on a condition.
- Select (between two registers for assignment) is alternative.

Standard Control Flow	Conditional Instructions
`<inst. setting cc>`	`<inst. setting cc>`
`BCEQ *+2`	`MULC R1, R2, R4, EQ`
`MUL R1, R2, R4`	`<continue processing>`
`<continue processing>`	





Branch Speedup

Branch Delay = max{ TIF time, CC set time}

TIF time can be reduced by adding an address calculation ALU

-- TLB can be bypassed for those addresses in the same virtual page

79% of branch targets are to the same 4K page

Improving CC set time relies on compiler assistance --

Move CC set instructions earlier

Branch delay slots

Branch Predication

Guess the outcome and begin executing; flush if its wrong.
Multiple choices of predication strategies --

A fixed strategy does the same thing for all branches

A static strategy predicts the same thing based on op code

A dynamic strategy predicts based on program behavior

Perfect prediction converts conditional delay to unconditional delay.

Instruction Class	Instruction	Guessed Successful	Guessed Unsuccessful
Unc Branch	BR	Always*	Never
Branch on	BC	Never	Always
Condition	BCR	Never	Always
Loop Control	BCT	Always*	Never
Call/Return	BAL/BALR	Always*	Never
* Known successful branches			

Inst Class	Branch %	% Correct
BR	72.5*.2=14.5	14.5
BC	72.5*.8=58	58*.46=27
BCT	9.8	9.8*.91=9
BAL /R	17.7	17.7
	Total	68.2%

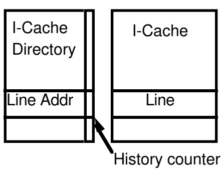
Guessing S on BC yields 72%

Dynamic Branch Prediction

Use a "saturating counter" to accumulate history of recent branches.

The counter is incremented when branch is taken, decremented otherwise: if half or more, guess "take"

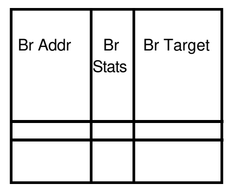
It takes very few bits per line



n	Compiler	Business	Scientific	OS
0	64.1	64.4	70.4	54.0
1	91.9	95.2	86.6	79.7
2	93.3	96.5	90.8	83.4
3	93.7	96.6	91.0	83.5
4	94.5	96.8	91.8	83.7
5	94.7	97.0	92.0	83.9

Branch Target Capture

Keep a cache of recent branch instructions, their branch stats and the target address.



Hit Ratio, LRU

n	Compiler	Business	Scientific	OS
1	0.031	0.121	0.158	0.012
2	0.075	0.150	0.223	0.066
4	0.185	0.212	0.272	0.154
8	0.298	0.247	0.402	0.259
16	0.369	0.315	0.636	0.275
32	0.514	0.412	0.693	0.301
64	0.634	0.624	0.812	0.435
128	0.769	0.832	0.953	0.545
256	0.888	0.929	0.970	0.615
512	0.967	0.968	0.993	0.672