Pipelining Review

2/4/2016

Section 3

Colin Schmidt

Agenda

- Iron Law
- Pipelining
- Branch Prediction
- Exceptions

Processor Performance

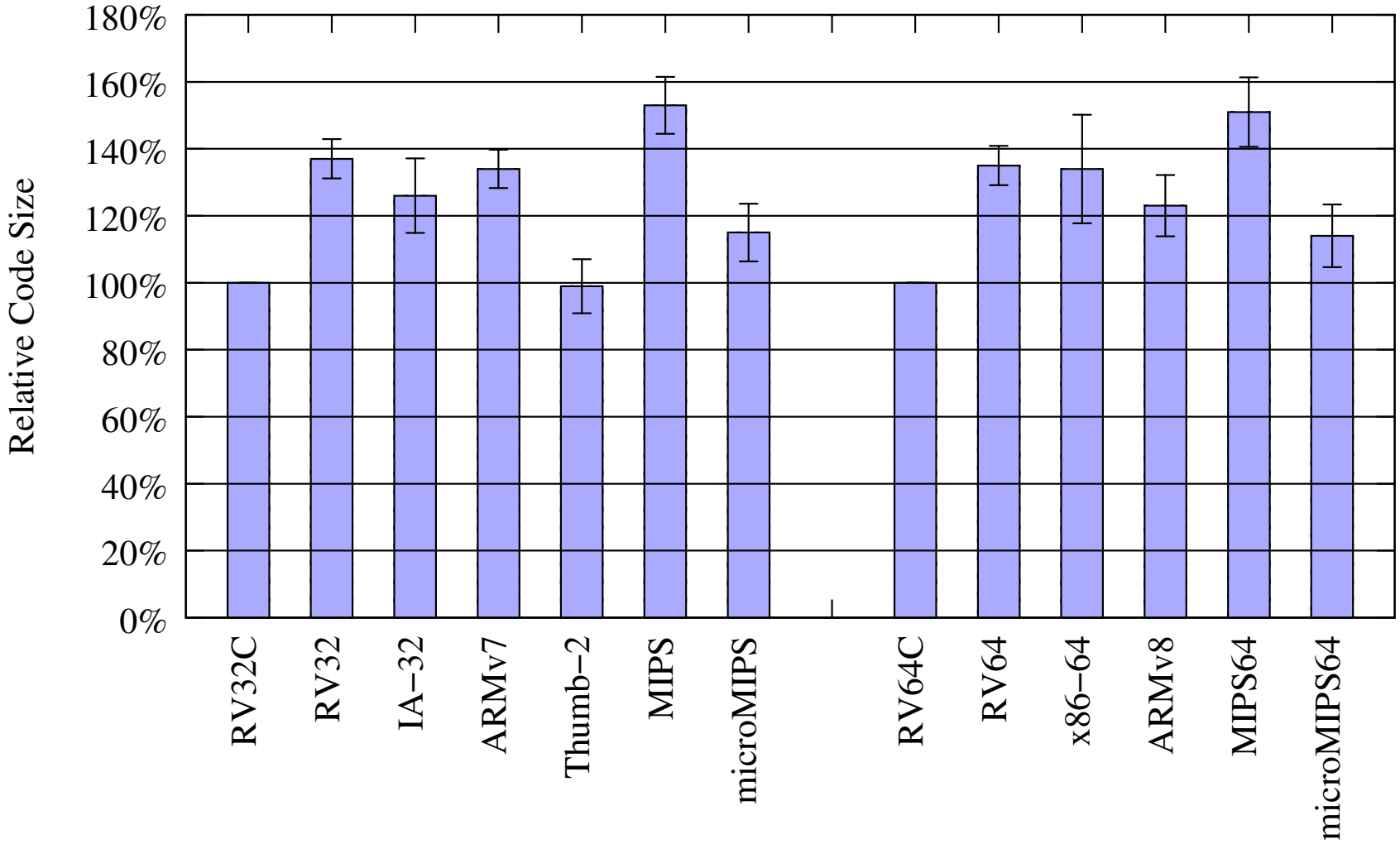
- Iron Law

$$\frac{\text{Time}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} * \frac{\text{Cycles}}{\text{Instruction}} * \frac{\text{Time}}{\text{Cycle}}$$

Instructions/Program

- Affected by
  - ISA
  - Compiler
  - Algorithm
  - Programmer

Static Code size by ISA SPEC CPU2006



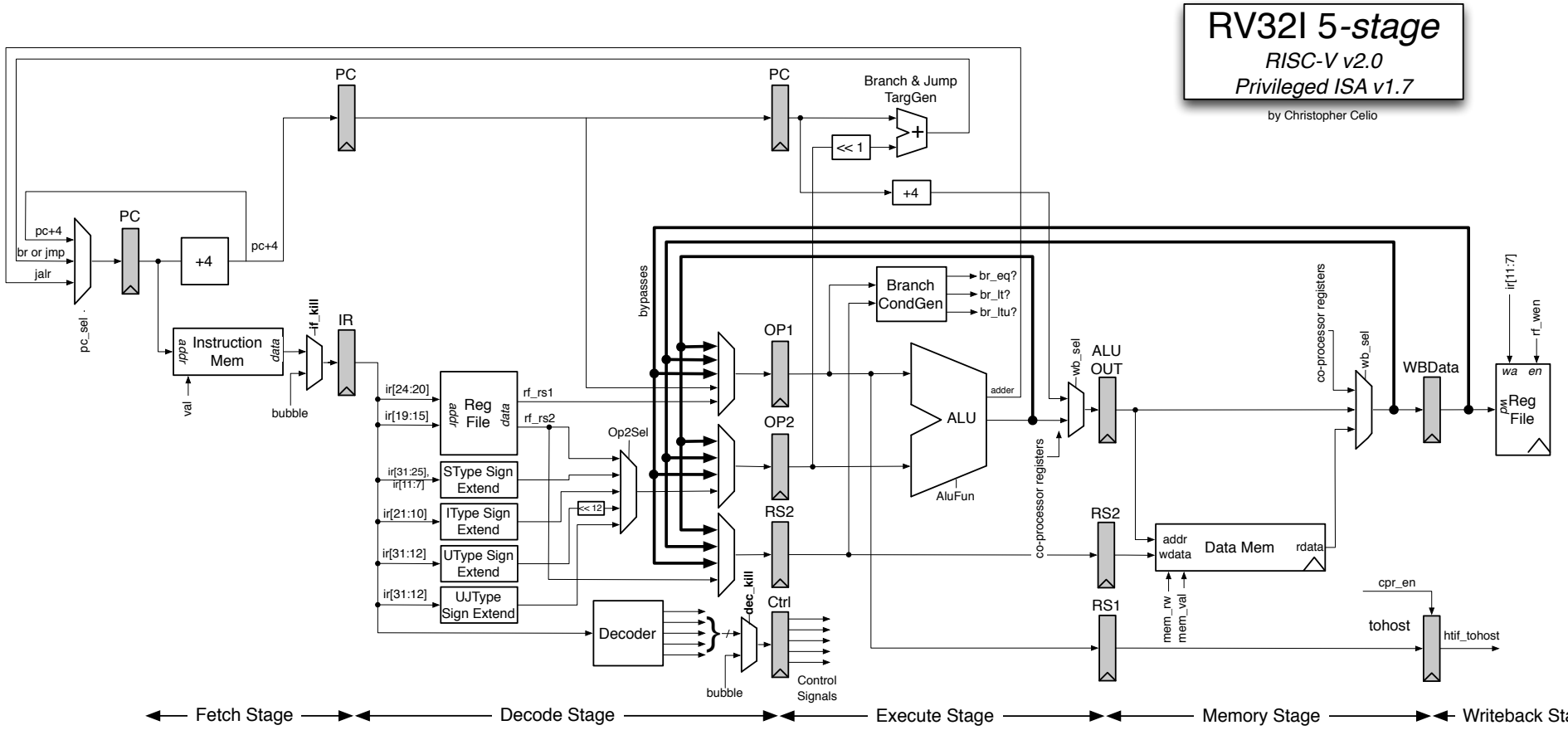
http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-1.pdf

CPI

- Ideal?
- Non-ideal
  - Structural Hazards
  - Data Hazards
  - Control Hazards
- Resolving
  - Stall/Interlock
  - Speculate
  - Bypass
  - Add hardware

Cycle Time

- Microarchitecture
  - Logic vs Memories
  - Control complexity
  - Datapath critical paths
    - FPU
- Technology 28nm, 45nm, etc.



time

(I_1) 096: ADD
(I_2) 100: BEQ +20
(I_3) 104: ADD
(I_4) 108:
(I_5) 300: ADD

Branch Delay Slot

- Why not have it
  - Expose microarchitecture in ISA
    - How many delay slots for 5 stage?
    - How about 10 stage? Or 3 stage?
    - What about an out of order superscalar machine?
  - Not filled often in practice
    - NOPs increase I-cache pressure
  - Branch predictors are really good

Branch/Jump Predicition

- Why?
  - Eliminate/mitigate control hazards
- How?
  - Spatial locality
  - Temporal locality

Bimodal(BHT) Predictor

- 2 bit saturating counter

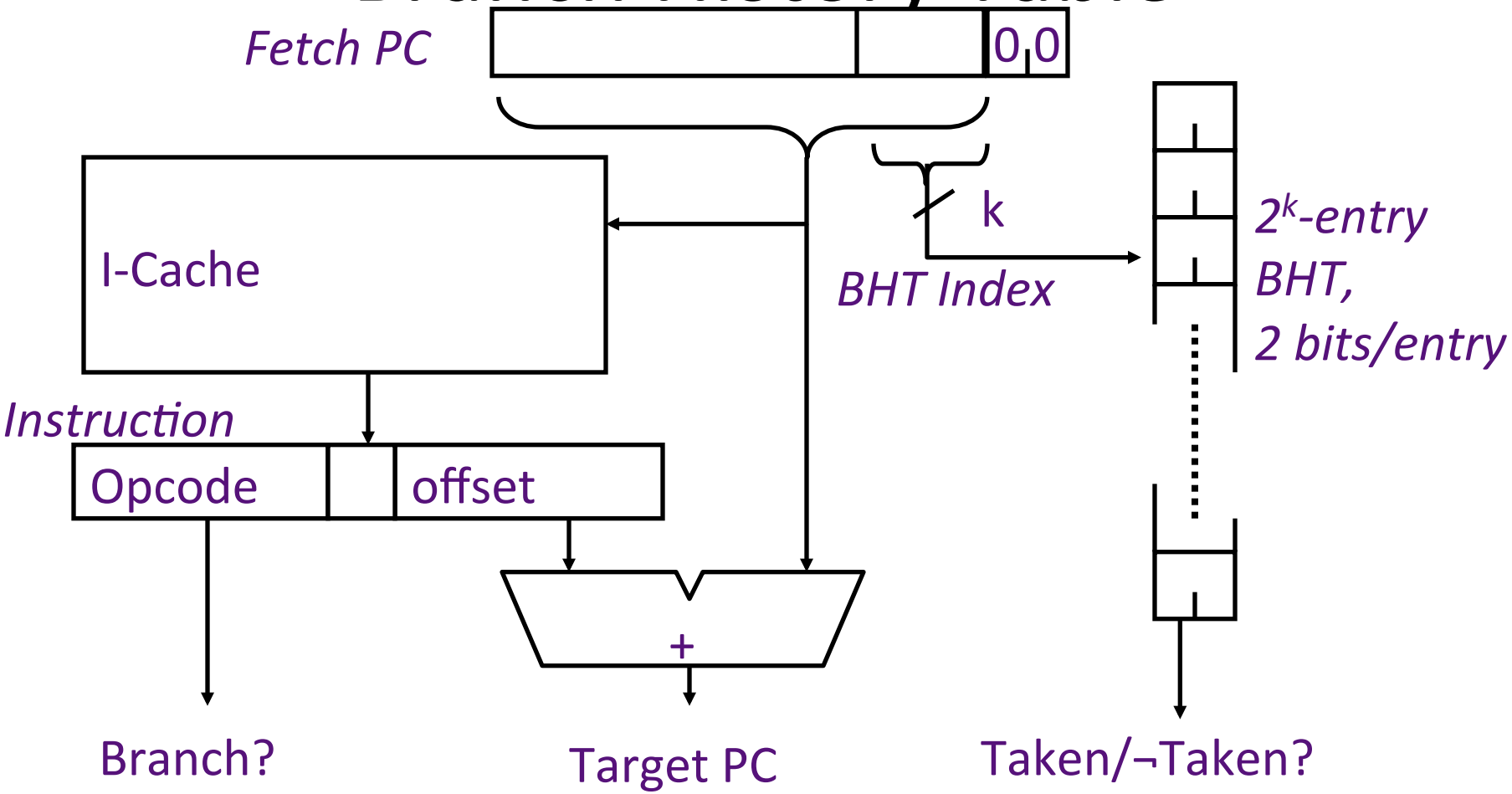
00 -> strongly not taken

01 -> weakly not taken

10 -> weakly taken

11 -> strongly taken

Branch History Table



4K-entry BHT, 2 bits/entry, ~80-90% correct predictions

Spatial(Global) vs Temporal(Local)

```
if(x < 7)
   …
if(x > 7)
   …
```

```
for(i=0;i<4;i++)
{…}
```

Spatial(Global) vs Temporal(Local)

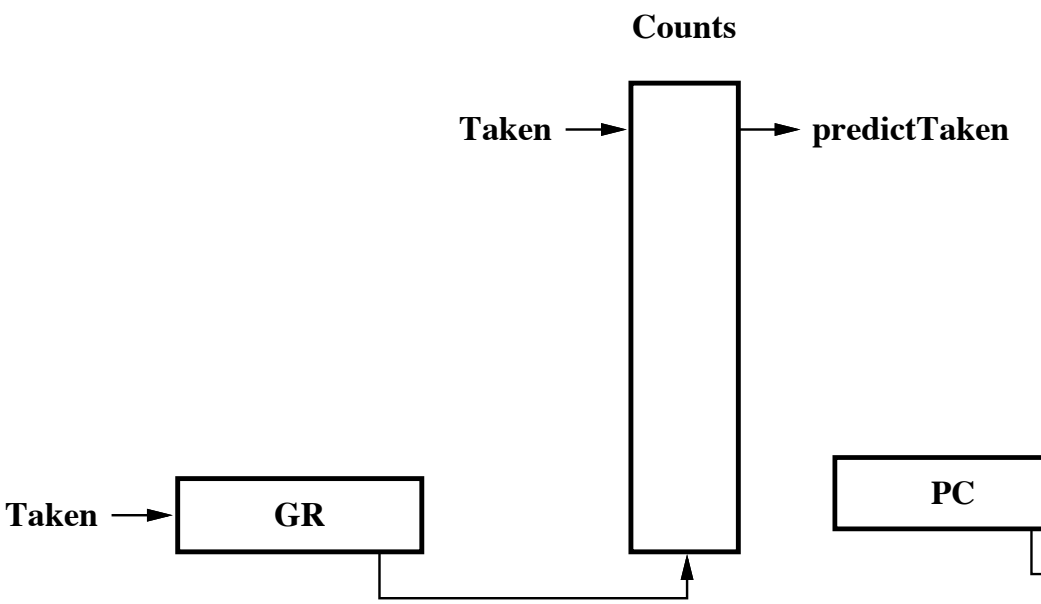


Figure 6: Global History Predictor Structure

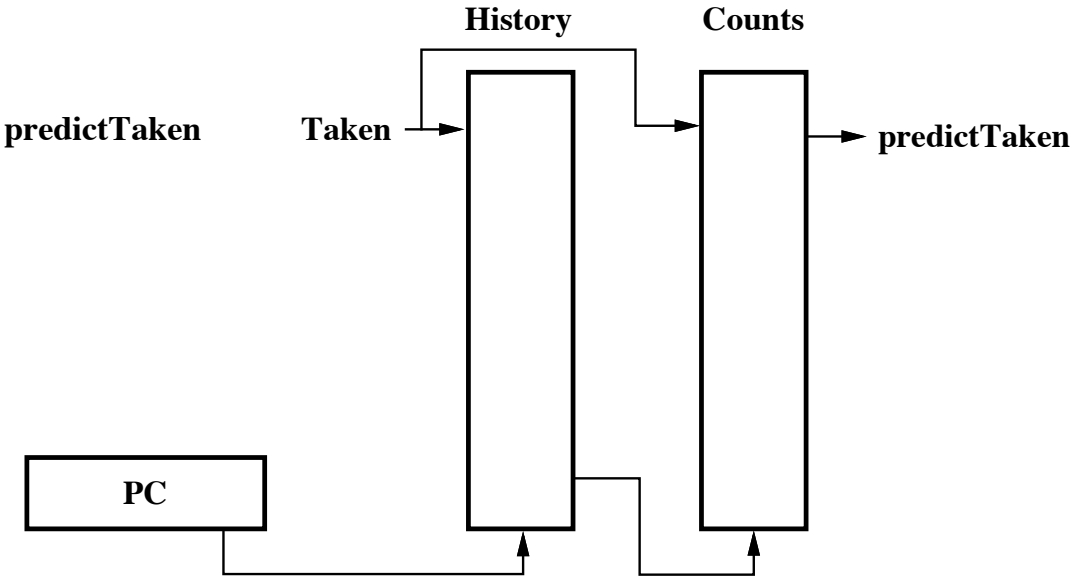


Figure 4: Local History Predictor Structure

Tradeoff

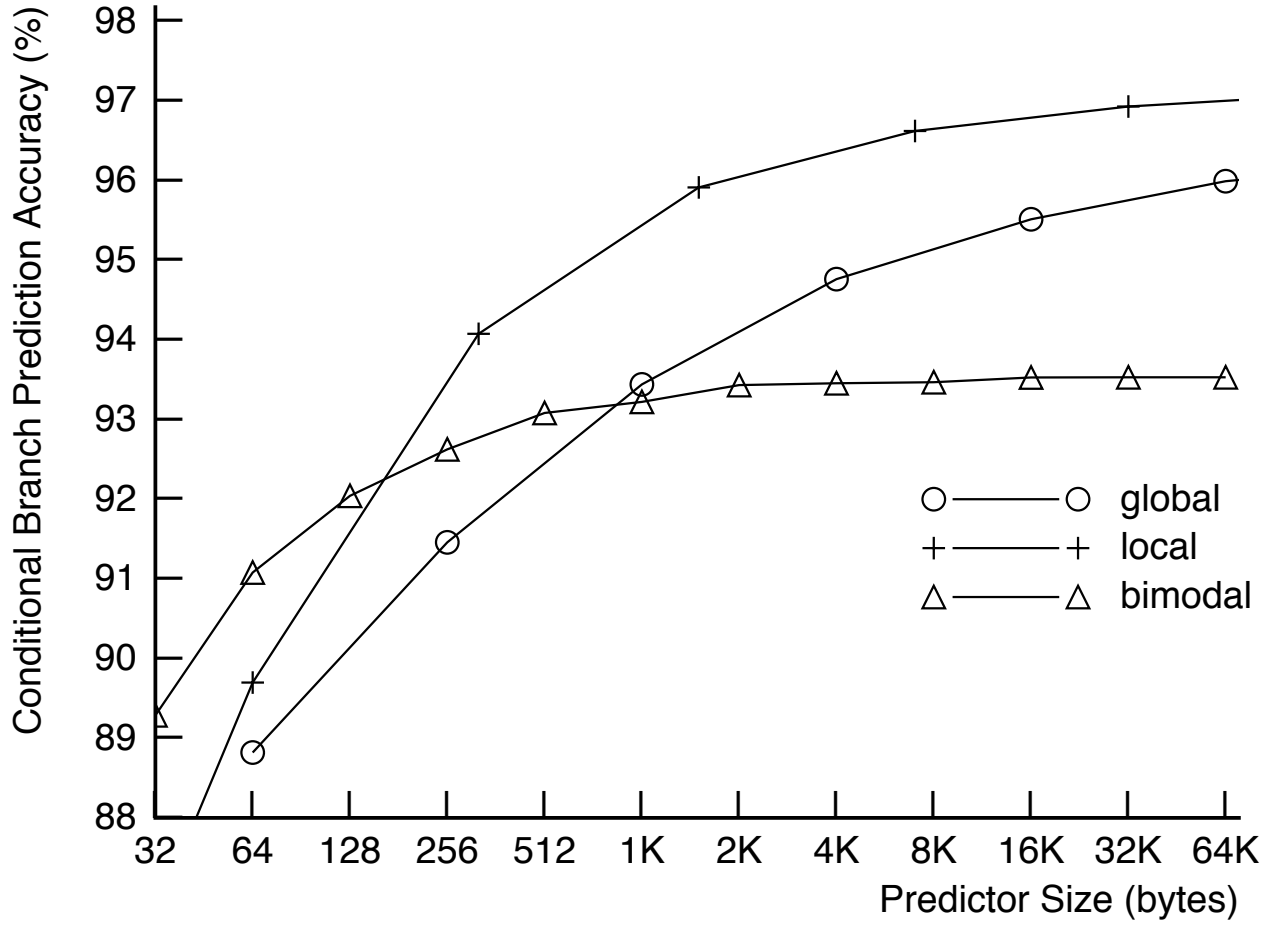
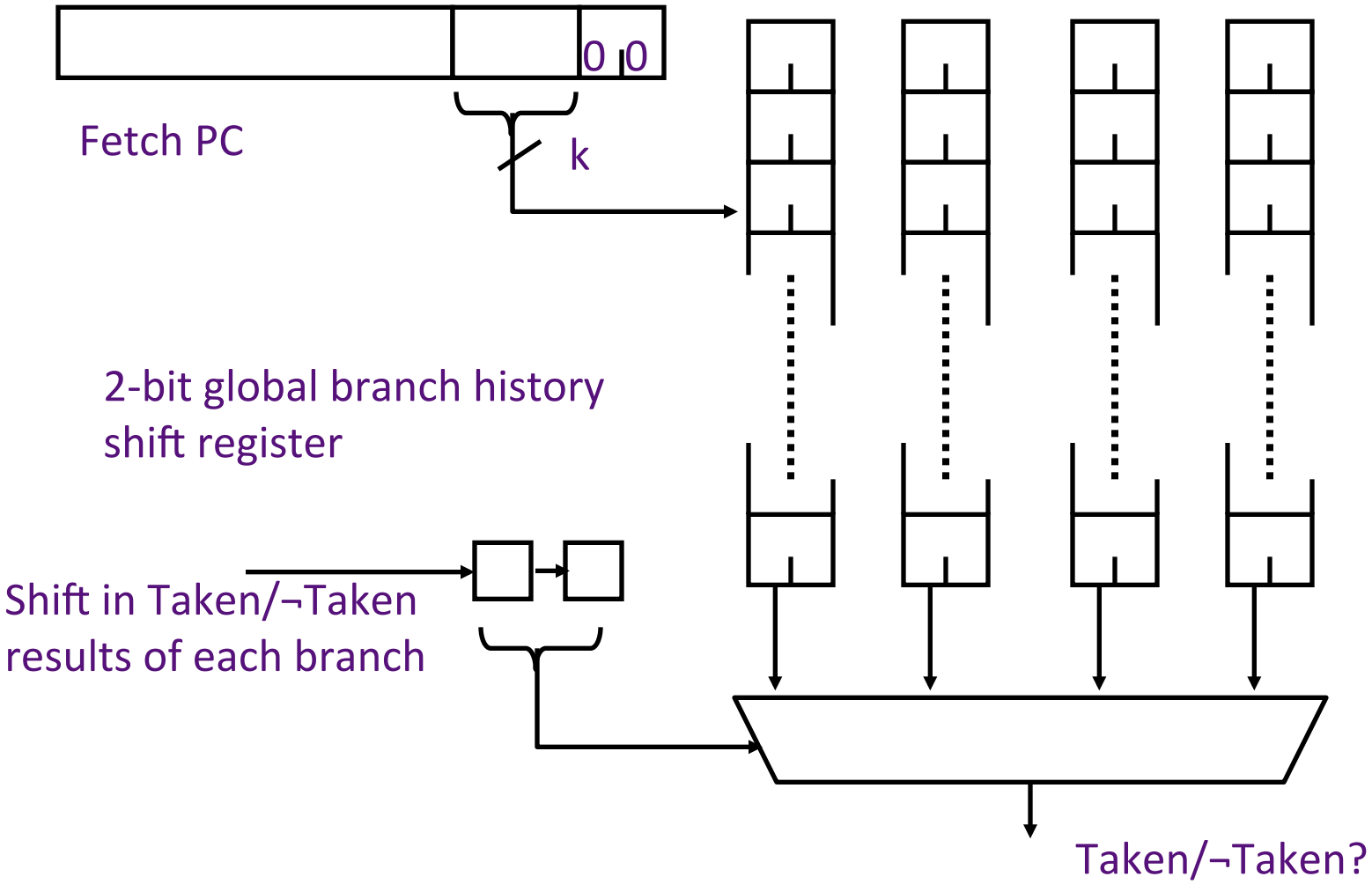


Figure 7: Global History Predictor Performance

http://www.hpl.hp.com/techreports/Compaq-DEC/WRL-TN-36.pdf

Two-Level Branch Predictor

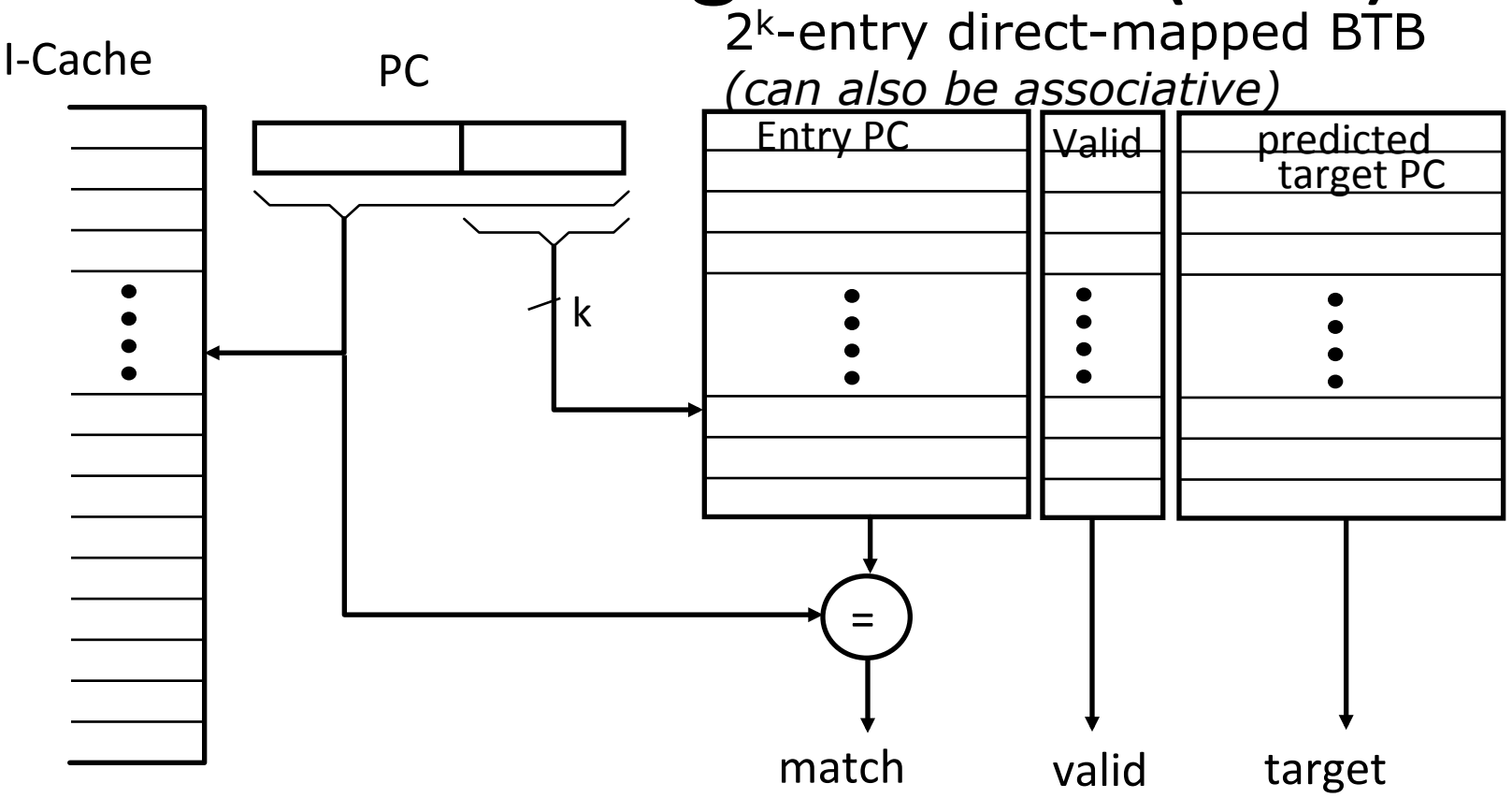
Pentium Pro uses the result from the last two branches to select one of the four sets of BHT bits (~95% correct)



What about Targets

- Taken/Not taken only part of the problem

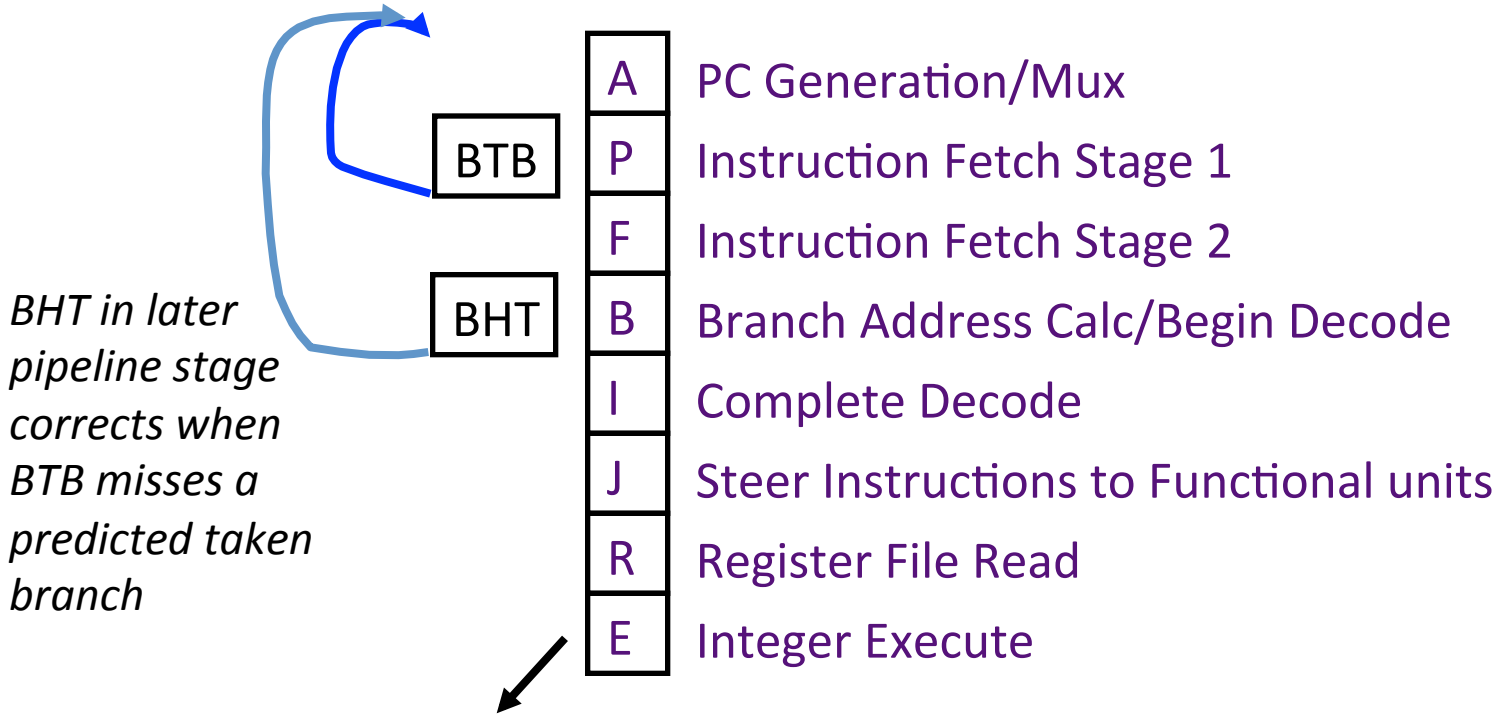
Branch Target Buffer (BTB)



- Keep both the branch PC and target PC in the BTB
- PC+4 is fetched if match fails
- Only *taken* branches and jumps held in BTB
- Next PC determined *before* branch fetched and decoded

Combining BTB and BHT

- BTB entries are considerably more expensive than BHT, but can redirect fetches at earlier stage in pipeline and can accelerate indirect branches (JR)
- BHT can hold many more entries and is more accurate



BTB/BHT only updated after branch resolves in E stage

Exceptions and Interrupts

- Interrupt
  - External asynchronous event (e.g. I/O)
  - Control transfer to supervisor
- Exception
  - Internal synchronous event (e.g. page fault)
  - May or may not transfer control
- Trap
  - Exception that forces control transfer

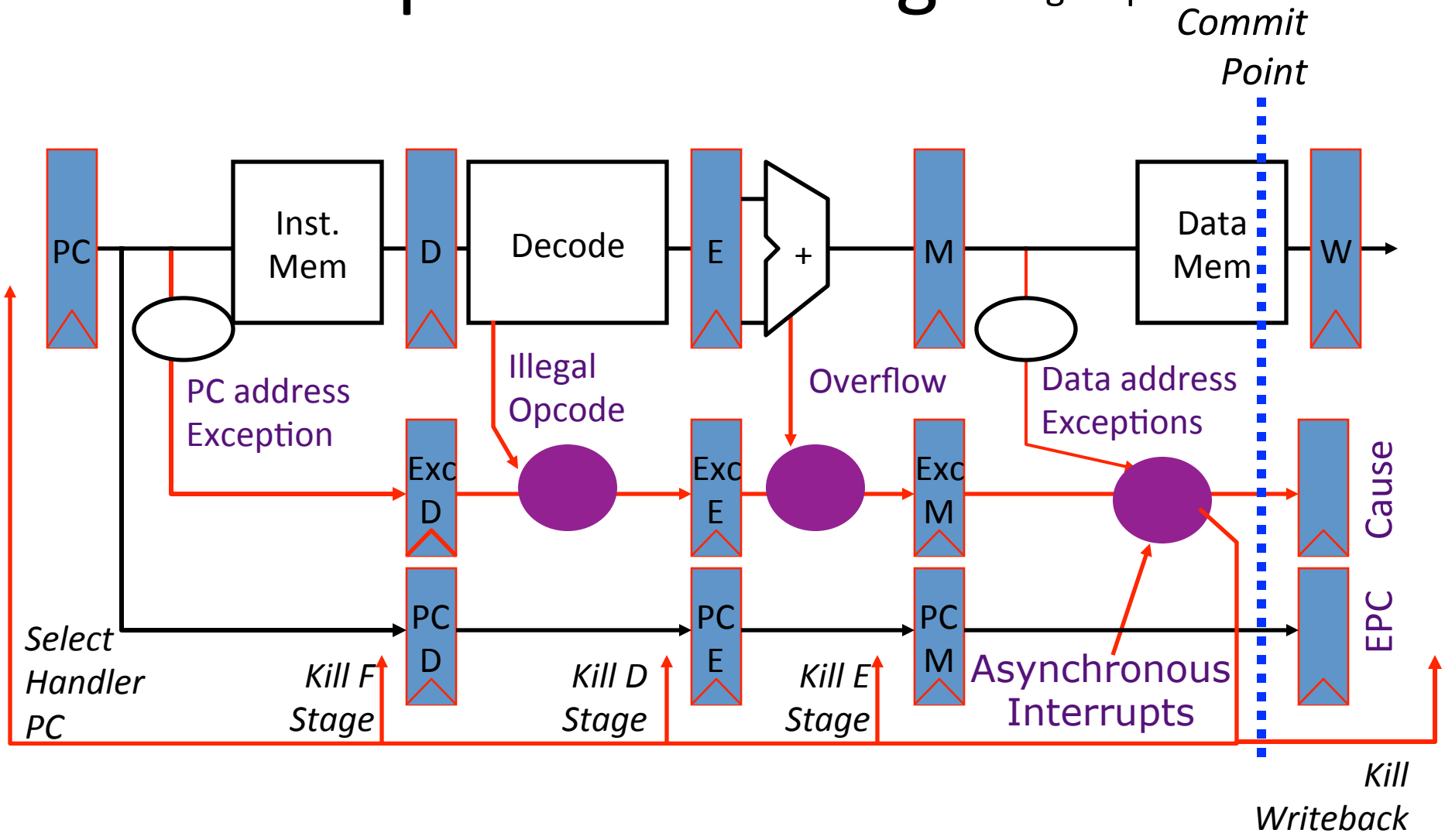
Precise vs Imprecise

- Precise exception/interrupts means that the processor looks as if it stopped exactly after one instruction and everything has been happening in program order
  - Pipelining not visible
  - Out of order execution not visible
  - Etc.
- Imprecise does not give this guarantee

Restartable Exceptions

- Subclass of imprecise
- Machine can save state and restart but the machine may be in an intermediate state
- Requires more state saving
- Support from OS to have large space to save state
- Easier/higher performance if state is complicated and long running

Exception Handling 5-Stage Pipeline



Questions