Towards a More Efficient Trace Cache

Amit Saha

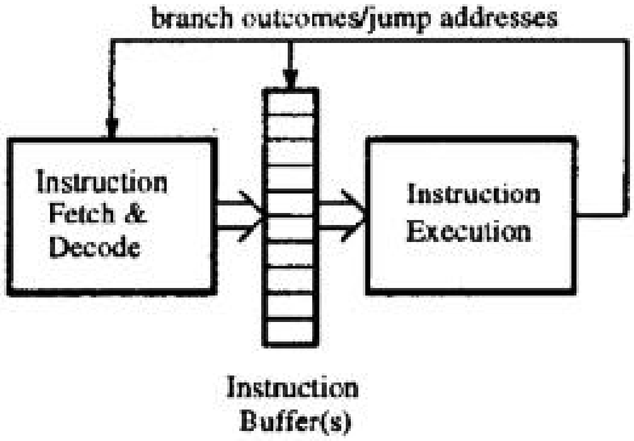
Jerry Yen

Rajnish Kumar

ELEC/COMP 525   April 24, 2001

Motivation

- Exploiting ILP
- Current limitations of instruction fetch mechanisms



From: Trace Cache: a Low Latency Approach to High Bandwidth Instruction Fetching by Rotenberg, et al. 1996

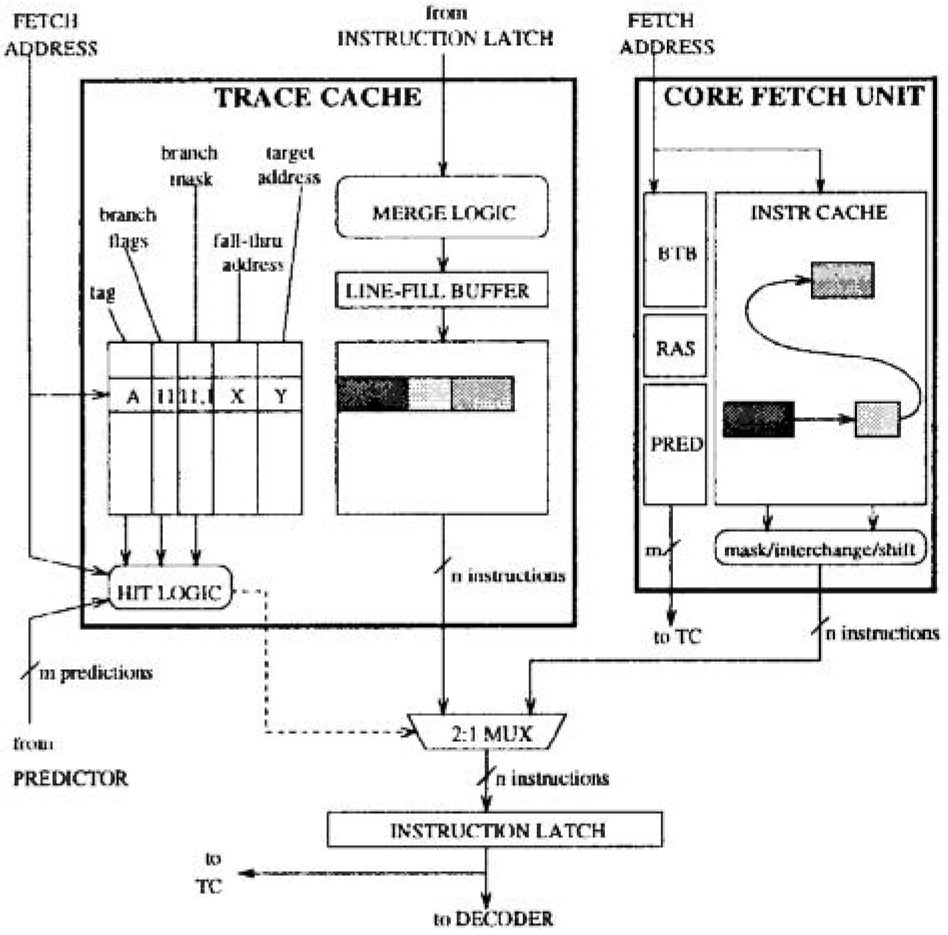
Hypothesis

Trace cache implemented by:

- Giving weights to entries based on past use and future usage prediction (branch prediction) *and*
- Using the weights for the line fill and replacement buffer logic

will enhance processor performance

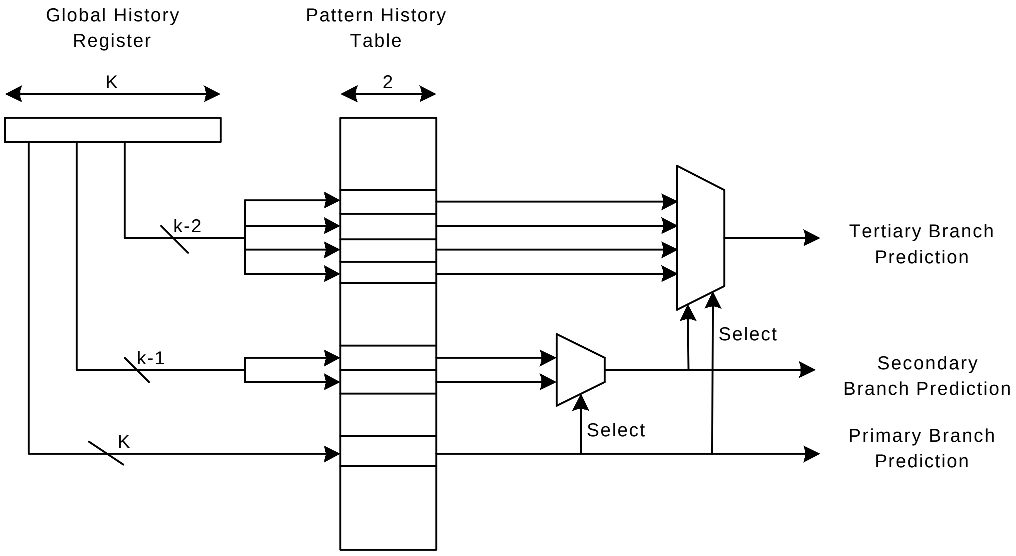
Architecture



- Trace Cache
  - 1024 or 32 entries
  - Max 3 blocks per entry
  - Max 16 instruction per entry

Branch Predictor

- Two Level Adaptive Branch Predictor


Global History Register
Pattern History Table
K
2
k-2
k-1
K
Select
Select
Tertiary Branch Prediction
Secondary Branch Prediction
Primary Branch Prediction

Weight Parameters

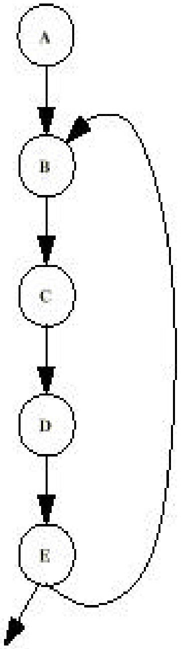
- Number of basic blocks
- Non-contiguity of the line
  - Zero-count in branch-prediction values
- Expected future use
  - 2-bit hit counter
  - Active-window-size field

Implementation

- Separate fields for different parameters
- Total weight of trace cache line is sum of
  - Basic_block_count weight
  - Branch prediction values mapped to weights
  - Number of hits in last *x* number of cycles
    - *x* is active_window_size.

Redundancy in Trace-Cache

- Line-fill-buffer logic changed :
  - If a block is the point of multiple entry, like B here start a new trace cache line with B.



Possible segments

ABC

DEB

CDE

BCD

EBC

Implementation

Example :

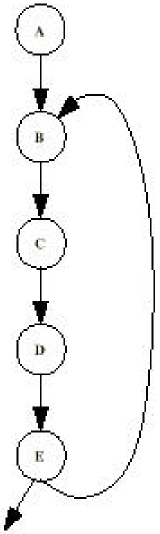
[ABC] → [ABC, DE]

↓

[ABC, DE, BCD]

↓

[BCD]



Possible segments

ABC

DEB

CDE

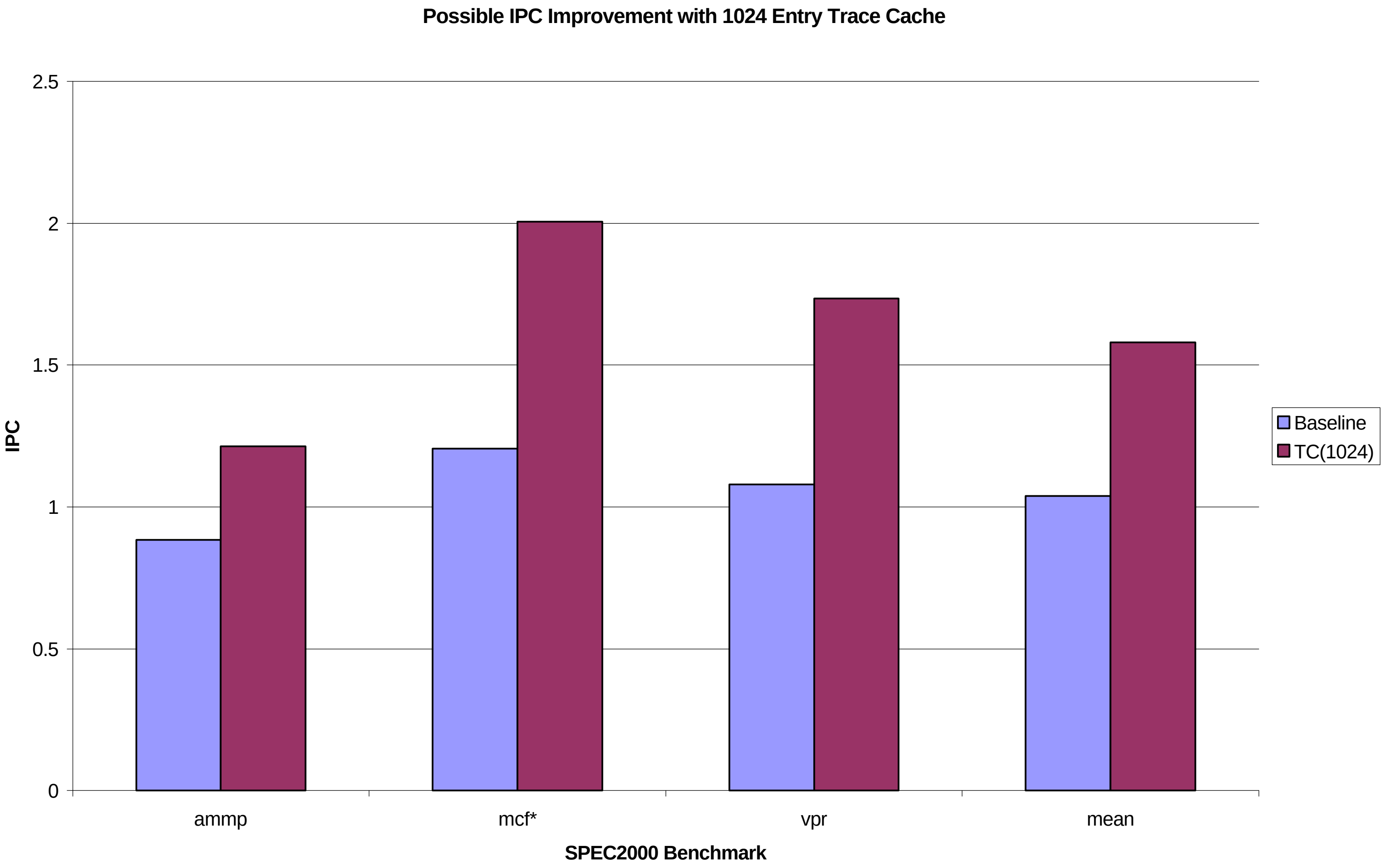
BCD

EBC

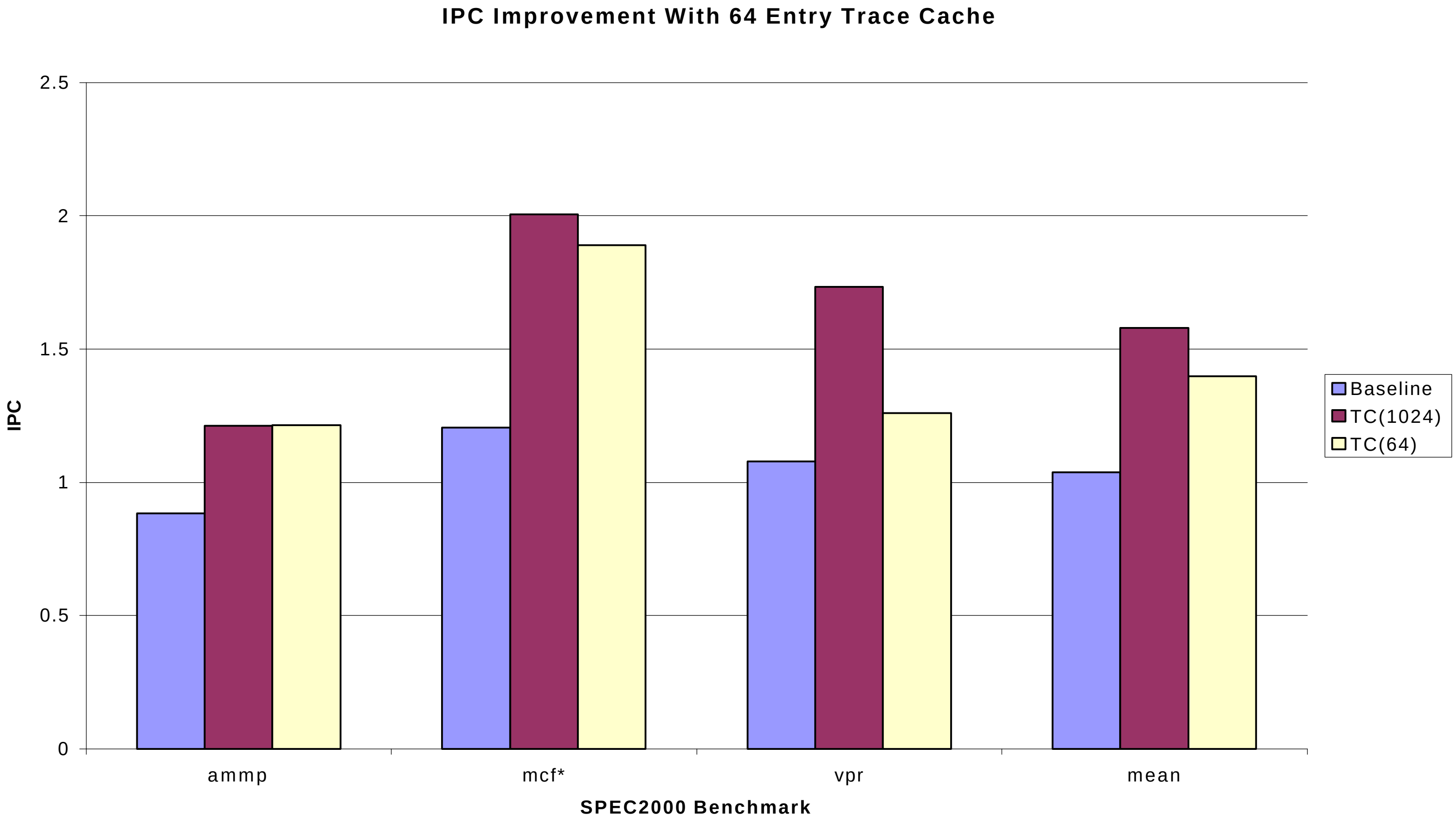
Methodology

- Baseline
  - Increased execution resources
- Baseline with TC
- Baseline with modified TC
- Unmodified Trace Cache
  - LRU replacement policy

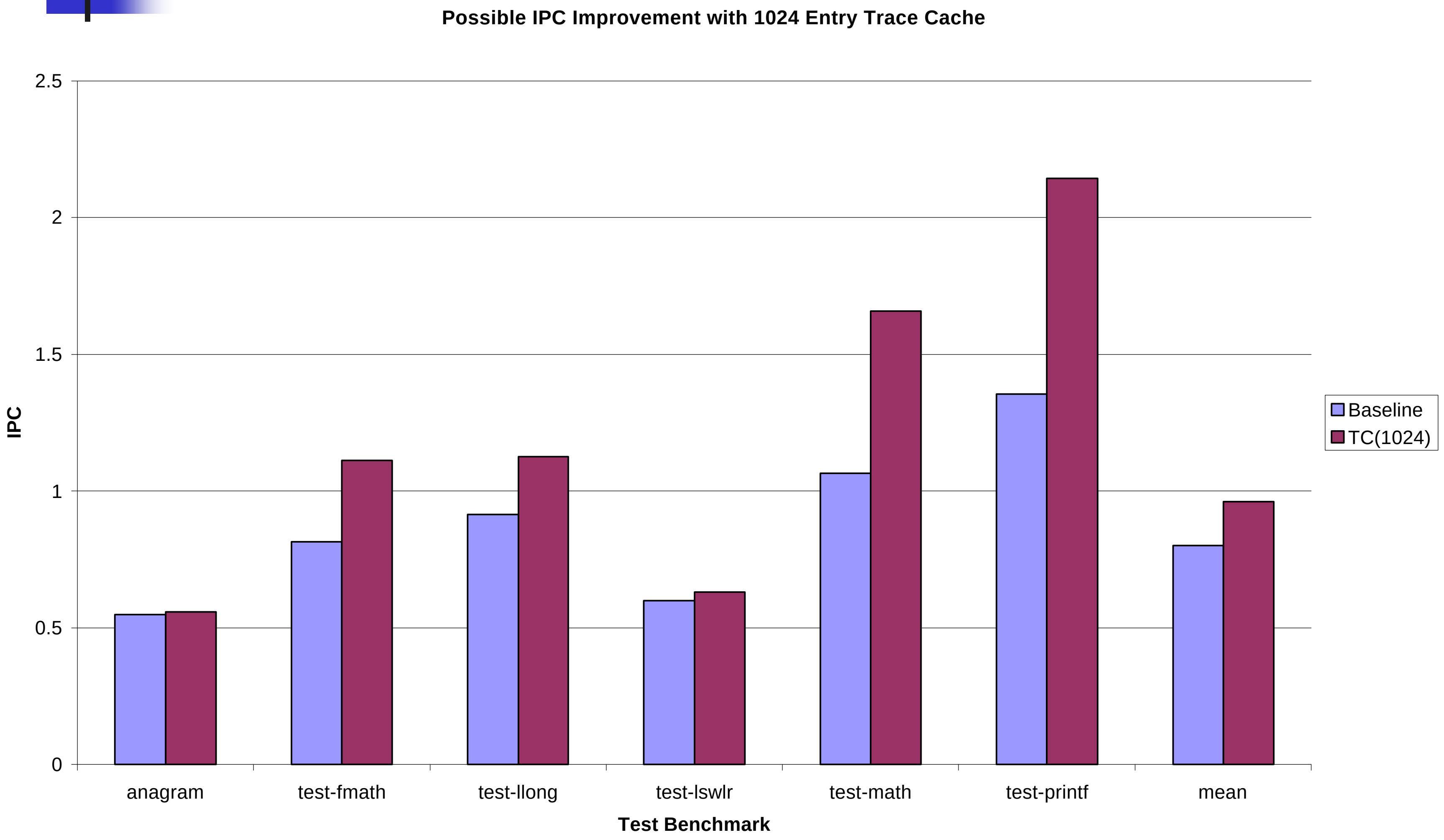
Ideal case


Possible IPC Improvement with 1024 Entry Trace Cache
2.5
2
1.5
1
0.5
0
IPC
ammp
mcf*
vpr
mean
SPEC2000 Benchmark
Baseline
TC(1024)

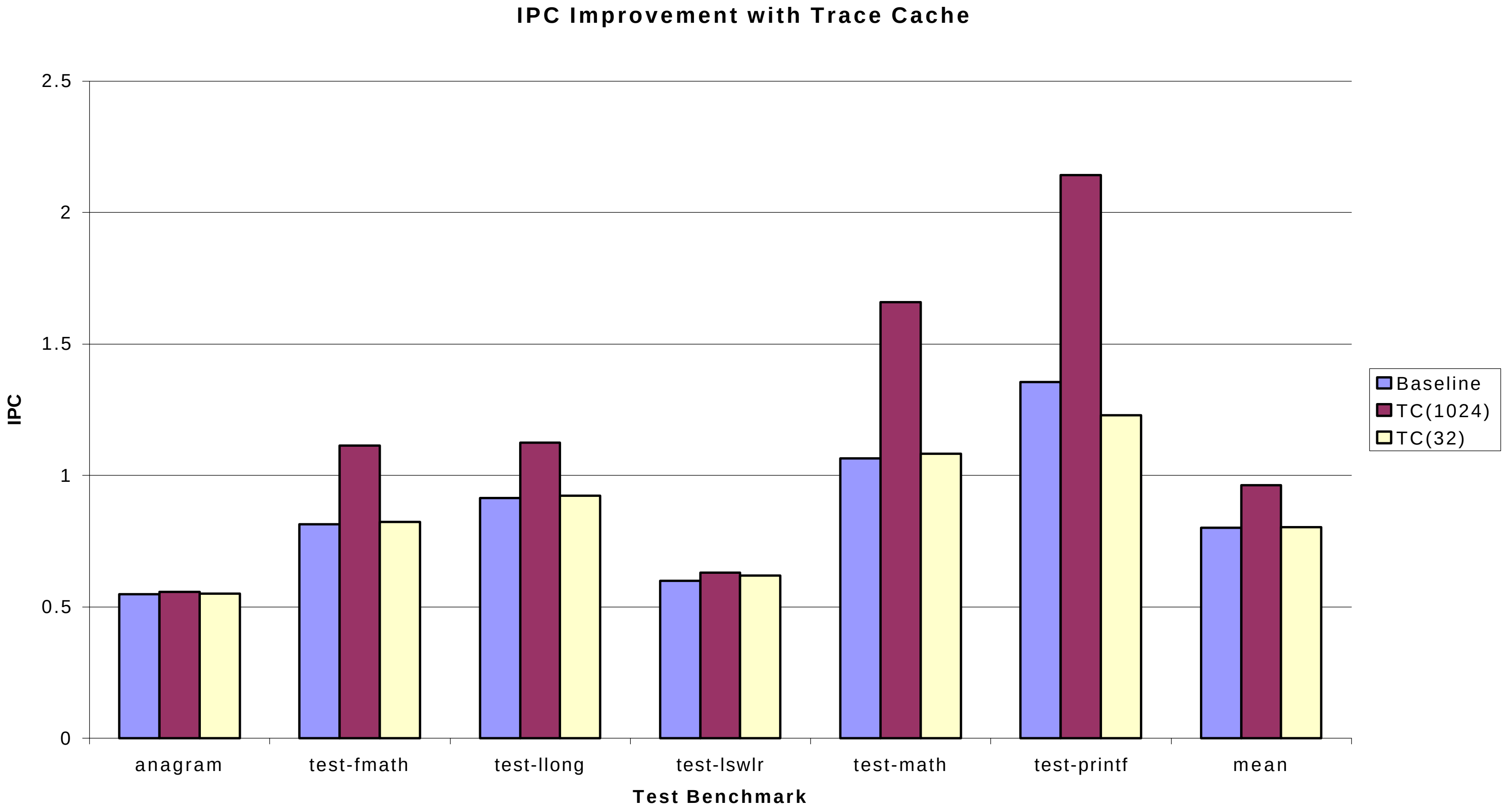
Small sized trace cache



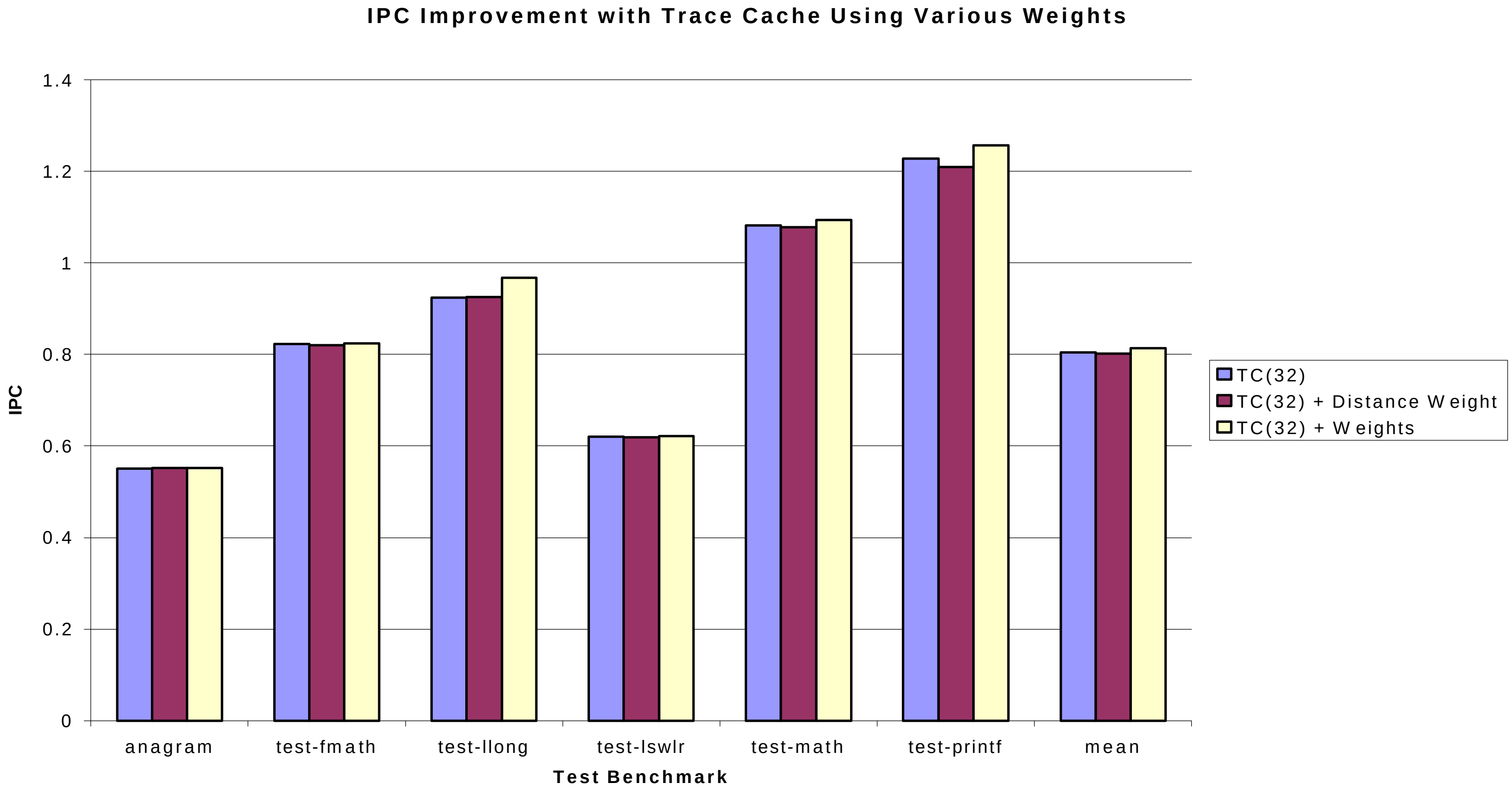
./tests/ benchmarks ideal case



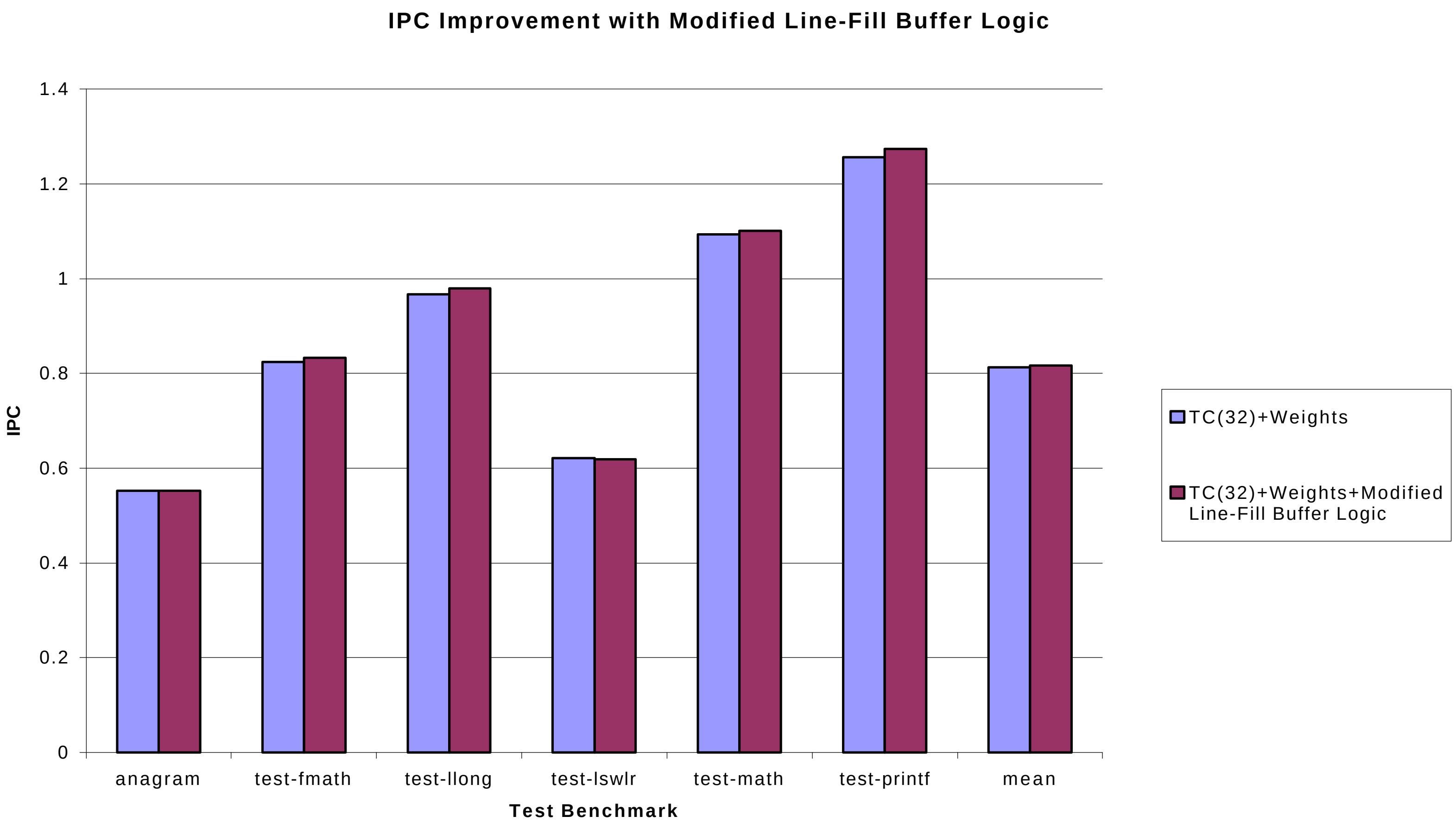
Small sized trace cache



Various weights used



Modified lfb logic



In conclusion

- Fetch Q is the bottleneck
- Hypothesis partially valid
  - Better results for Spec2000 ?
- Better combination of proposed weights ?
- New weights ?
- Same weights to work across multiple benchmarks ?

Learning experience

- Difficult to increase IPC beyond what a base trace cache offers.
- How to proceed with such research projects
- Why man-months are so important in architecture research ?