Lecture 8: Branch Prediction, Dynamic ILP

• Topics: static speculation and branch prediction (Sections 2.3-2.6)
Correlating Predictors

• Basic branch prediction: maintain a 2-bit saturating counter for each entry (or use 10 branch PC bits to index into one of 1024 counters) – captures the recent “common case” for each branch

• Can we take advantage of additional information?
  ➢ If a branch recently went 01111, expect 0; if it recently went 11101, expect 1; can we have a separate counter for each case?
  ➢ If the previous branches went 01, expect 0; if the previous branches went 11, expect 1; can we have a separate counter for each case?

Hence, build correlating predictors
Global Predictor

A single register that keeps track of recent history for all branches

00110101

Branch PC

Table of 16K entries of 2-bit saturating counters

Also referred to as a two-level predictor
Local Predictor

Table of 16K entries of 2-bit saturating counters

Also a two-level predictor that only uses local histories at the first level

Branch PC

Use 6 bits of branch PC to index into local history table

Table of 64 entries of 14-bit histories for a single branch

10110111011001

14-bit history indexes into next level
Local/Global Predictors

• Instead of maintaining a counter for each branch to capture the common case,

→ Maintain a counter for each branch and surrounding pattern
→ If the surrounding pattern belongs to the branch being predicted, the predictor is referred to as a local predictor
→ If the surrounding pattern includes neighboring branches, the predictor is referred to as a global predictor
Tournament Predictors

• A local predictor might work well for some branches or programs, while a global predictor might work well for others.

• Provide one of each and maintain another predictor to identify which predictor is best for each branch.
Branch Target Prediction

• In addition to predicting the branch direction, we must also predict the branch target address

• Branch PC indexes into a predictor table; indirect branches might be problematic

• Most common indirect branch: return from a procedure – can be easily handled with a stack of return addresses
An Out-of-Order Processor Implementation

Branch prediction and instr fetch

Instr Fetch Queue

R1 ← R1+R2
R2 ← R1+R3
BEQZ R2
R3 ← R1+R2
R1 ← R3+R2

Decode & Rename

T1 ← R1+R2
T2 ← T1+R3
BEQZ T2
T4 ← T1+T2
T5 ← T4+T2

Reorder Buffer (ROB)

Instr 1  T1
Instr 2  T2
Instr 3  T3
Instr 4  T4
Instr 5  T5
Instr 6  T6

Register File R1-R32

ALU
ALU
ALU

Issue Queue (IQ)

Results written to ROB and tags broadcast to IQ

Results written to ROB and tags broadcast to IQ
Design Details - I

• Instructions enter the pipeline in order

• No need for branch delay slots if prediction happens in time

• Instructions leave the pipeline in order – all instructions that enter also get placed in the ROB – the process of an instruction leaving the ROB (in order) is called commit – an instruction commits only if it and all instructions before it have completed successfully (without an exception)

• To preserve precise exceptions, a result is written into the register file only when the instruction commits – until then, the result is saved in a temporary register in the ROB
Design Details - II

- Instructions get renamed and placed in the issue queue – some operands are available (T1-T6; R1-R32), while others are being produced by instructions in flight (T1-T6)

- As instructions finish, they write results into the ROB (T1-T6) and broadcast the operand tag (T1-T6) to the issue queue – instructions now know if their operands are ready

- When a ready instruction issues, it reads its operands from T1-T6 and R1-R32 and executes (out-of-order execution)

- Can you have WAW or WAR hazards? By using more names (T1-T6), name dependences can be avoided
Design Details - III

• If instr-3 raises an exception, wait until it reaches the top of the ROB – at this point, R1-R32 contain results for all instructions up to instr-3 – save registers, save PC of instr-3, and service the exception

• If branch is a mispredict, flush all instructions after the branch and start on the correct path – mispredicted instrs will not have updated registers (the branch cannot commit until it has completed and the flush happens as soon as the branch completes)

• Potential problems: ?
Title

• Bullet