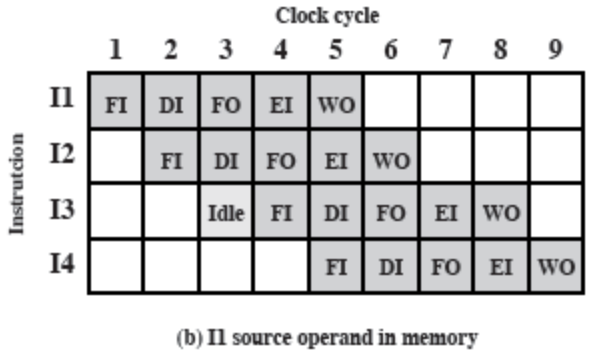
CS430 Computer Architecture

Spring 2015

More Pipeline Hazards

- pipeline hazard (pipeline bubble) – occurs when some portion of the pipeline must stall because execution cannot continue
- resource hazard (structural hazard)



(b) I1 source operand in memory

Cycle 3 assumes FO is from memory so FI of I3 must be delayed. All other operands are assumed to be in registers

Data Hazards

- data hazard – conflict in the access of an operand location
  - RAW (Read After Write) – "true dependency" instruction i+1 needs the result written by instruction i. The hazard occurs if instruction i+1 reads BEFORE instruction i writes
  - WAR (Write After Read) – "antidependency" instruction i reads from a location and instruction i+1 writes to the same location. The hazard occurs if instruction i+1 writes BEFORE instruction i has read from the same location
  - WAW (Write After Write) – "output dependency" instructions i and i+1 write to the same location. The hazard occurs if the write occurs in reverse order

Control Hazard

- control hazard (branch hazard) – pipeline makes the wrong decision on a branch prediction
  - can cause greater performance loss that a data hazard
- branch may or may not change the PC
  - taken - branch changes PC to target
  - not taken (untaken) - execution falls through

Control Hazard

- Dealing with Branches
  - Stall Until Target Known
  - Multiple Streams
  - Prefetch branch target
  - Loop Buffer
  - Branch Prediction
  - Delayed Branch

Stall Until Target Known Aggressive

```
branch instr (i)      IF ID EX M WB
instr i+1                IF IF ID EX M  WB
instr i+2                      IF ID EX M WB
```

Aggressive means during ID

- decode instr
- read registers
- do equality test on registers for possible branch
- sign extend offset field if needed
- compute branch target if needed

The second IF of instr i+1 might be redundant but the branch penalty is 1 cycle

Multiple Streams

- Replicate the initial portions of the pipeline for the target instruction and the instruction following the branch
- Disadvantages:
  - Contention delays for access to registers and memory
  - Additional branch instructions may enter the pipeline
- IBM 370/168 & IBM 3033 have two or more pipeline streams

Prefetch Branch Target

- The target is prefetched in addition to the instruction following the branch.
- If the branch is taken the target is fetched.
- IBM 360/91 uses this approach

Loop Buffer

- Very high speed memory maintained by the instruction fetch stage
- Contains n most recently fetched instructions
- If the branch is taken, the loop buffer is first checked

Loop Buffer

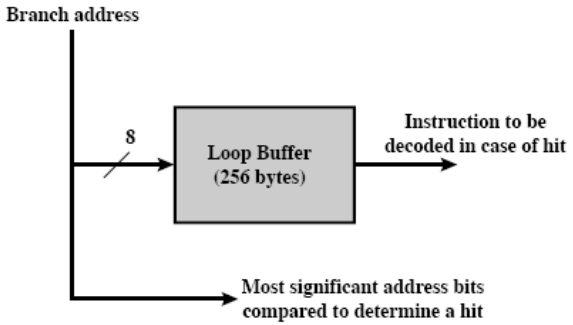


Figure 12.15 Loop Buffer

- The least significant 8-bits are used to index the buffer
- The remaining bits are the tag to see if the branch target is actually in the buffer

Loop Buffer

- Assuming the Loop Buffer works with Physical Addresses
  1. What is the branch address if the jump is taken?
  2. What is the branch address if the jump is not taken?
  3. What is the address index?
  4. What is the tag?

```
13CF:0100 B80000        MOV     AX,0000
13CF:0103 BB0000        MOV     BX,0000
13CF:0106 40            INC     AX
13CF:0107 01C3          ADD     BX,AX
13CF:0109 3D0A00        CMP     AX,000A
13CF:010C 75F8          JNZ     0106
13CF:010E 90            NOP
```

Loop Buffer

- Advantages
  1. Useful for dealing with code iteration as the instructions will be in the loop after the first iteration
  2. If the branch is a few instructions ahead, which is the case with several IF-THEN type statements, the target is already in the buffer.
- CDC computers, CRAY-1

Branch Prediction

- Predict never taken
- Predict always taken
- Predict by opcode
- Taken/Not Taken switch
- Branch history table

Predict Never Taken

- Predict never taken (static approach used by 68020, VAX 11/780)
- Note: If the instruction following the branch would cause a page fault or protection violation, the next instruction is not fetched.
- Studies analyzing program behavior show that conditional branches are taken more than 50% of the time.
- It depends on whether we are branching forward or backward:
  - backward: 90% probability that it is taken
  - forward: IFs 50% probability taken

Predict

- Always Taken (static)
- By opcode (static)
  - Prefetch decision is based on the branch's opcode
  - In some cases, success rates are as high as 75%

Taken/Not Taken Switch