6.175: Constructive Computer Architecture

Tutorial 5

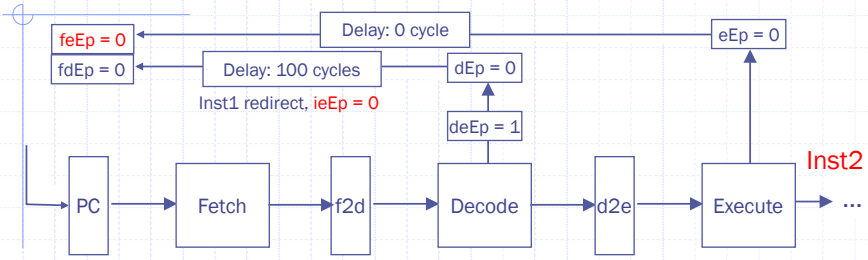
Epochs, Debugging, and Caches

Quan Nguyen
(Troubled by the two biggest problems in computer science... and Comic Sans)

Agenda

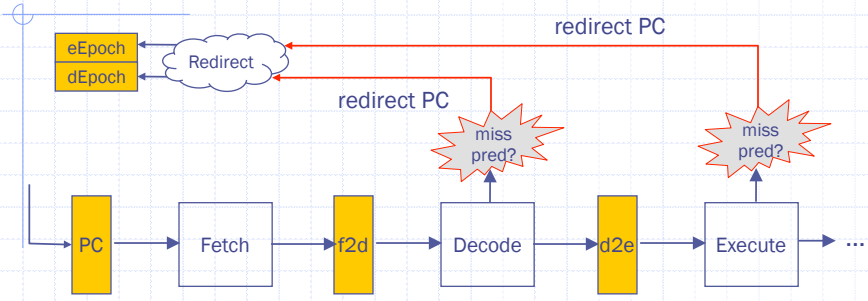
- Epochs: a review
- Debugging your processor ft. Piazza
- Caches: a primer

Review: 1-bit Distributed Epochs



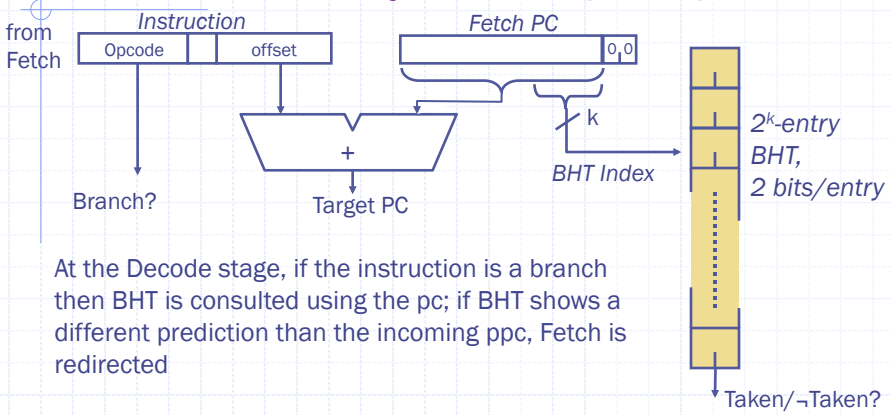
- Decode redirects Inst1 (ieEp = idEp = 0)
- Execute redirects Inst1
- Correct-path Inst2 (ieEp = 1, idEp = 0) issues
- Execute redirects Inst2
- Inst1 redirect arrives at Fetch (ieEp == feEp)
  - change PC to a wrong value

Review: Unbounded Global Epochs



- Both Decode and Execute can redirect the PC
  - Execute redirect should never be overruled
- Global epoch for each redirecting stage
  - eEpoch: incremented when redirect from Execute takes effect
  - dEpoch: incremented when redirect from Decode takes effect
  - Initially set all epochs to 0

Review: Branch History Table (BHT)

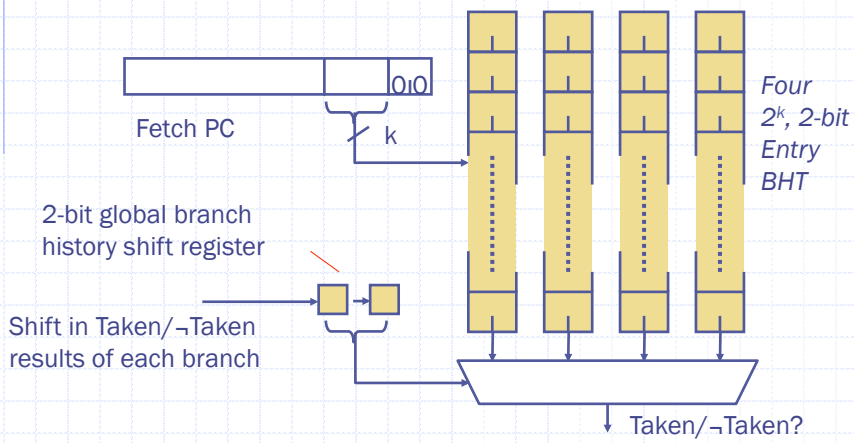


At the Decode stage, if the instruction is a branch then BHT is consulted using the pc; if BHT shows a different prediction than the incoming ppc, Fetch is redirected

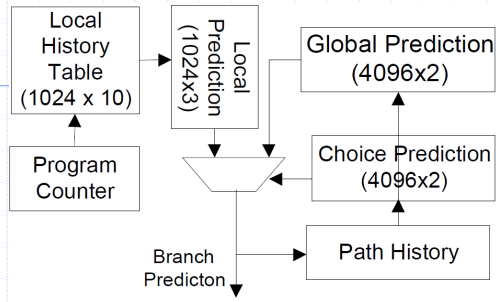
4K-entry BHT, 2 bits/entry, ~80-90% correct direction predictions

Review: 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)



Review: Tournament Predictor



"The Alpha 21264 Microprocessor Architecture"

- 10-bit PC: index 1024 x 10-bit local history table
- 10-bit local history
  - Index 1024 x 3-bit BHT: prediction 1
- 12-bit global history
  - Index 4096 x 2-bit BHT: prediction 2
  - Index 4096 x 2-bit BHT: select between predictions 1, 2

Debugging Your Processor

Unsupported Instruction

- Processor initialized? (csrf.started)
- What could be redirecting your PC?
  - Faulty branch address calculation?
  - BTB? (Lab 6 hint!)
  - Bad instruction? (unlikely for this course)

Processor Hangs

- Rules conflict?
  - Check schedule (option "`-show-schedule`")
  - Use $display() statements to diagnose
- Did you size pipeline FIFOs correctly?
- Are FIFOs being drained?

Incorrect Behavior

- Which test fails?
- Where is it in the dump?
- Do you have a log?
  - If your rules don't fire, temporarily make simpler rules

Demo: our code

src/TwoStage.bsv

```
rule doFetch (csrf.started);
  // fetch
  Data inst = iMem.req(pcReg[0]);
  // Addr predPc = btb.predPc(pcReg[0]);
  Addr predPc = pcReg[0]; // always predict PC to be next PC
  ...
endrule
```

```
[qmn@vlsifarm] $ ./run_asm.sh twostage
  ...
-- assembly test: lw --
ERROR: Executing unsupported instruction at pc: 00001000.
Exiting
^C
```

Demo: the log

scemi/sim/logs/lw.log

```
  ...
Cycle 5 ------------------------------------------------
Fetch: PC = 00000208, inst = 0000a183, expanded = lw r 3 = [r 1 0x0]
Execute finds misprediction: PC = 00000208
Fetch: Mispredict, redirected by Execute

Cycle 6 ------------------------------------------------
Fetch: PC = 00001000, inst = 00ff00ff, expanded = unsupport 0x00ff00ff
Execute: Kill instruction
```


Demo: the dump

programs/assembly/build/assembly/dump/lw.dump

```
Disassembly of section .text:

00000200 <_start>:
 200:   000010b7                lui     x1, 0x1
 204:   00008093                mv      x1, x1
 208:   0000a183                lw      x31, 0(x1) # 1000
   ...

Disassembly of section .data:

00001000 <begin_signature>:
    1000:       00ff                    0xff
    1002:       00ff                    0xff
```

Demo: back to the code

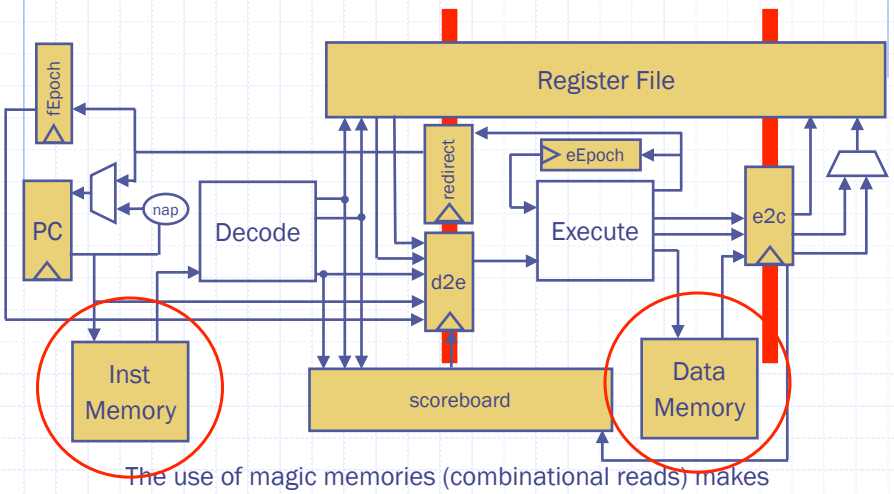
src/TwoStage.bsv

```
rule doExecute (csrf.started);
  if (eInst.mispredict) begin
    $display("Execute finds misprediction: PC = %x", f2e.pc);
    exeRedirect[0] <= Valid(ExeRedirect{
      pc: f2e.pc,
      nextPc: eInst.addr
    });
  end
endrule
```

- What sets eInst.addr?
- What happens to exeRedirect[0]?

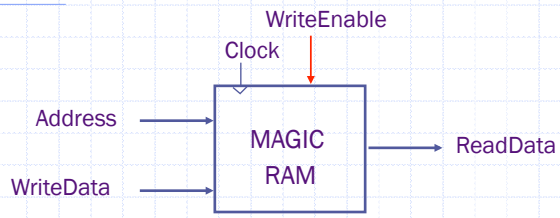
Caches

Multistage Pipeline



The use of magic memories (combinational reads) makes these designs unrealistic

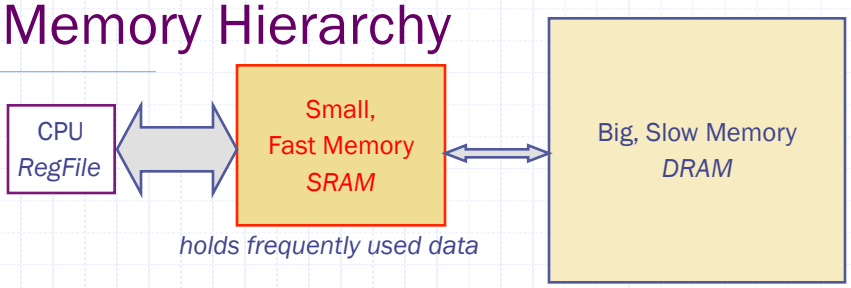
Magic Memory Model



- Reads and writes are always completed in one cycle
  - a Read can be done any time (i.e. combinational)
  - If enabled, a Write is performed at the rising clock edge
  - *(the write address and data must be stable at the clock edge)*

In a real DRAM the data will be available several cycles after the address is supplied

Memory Hierarchy



holds frequently used data

size:	RegFile << SRAM << DRAM	
latency:	RegFile << SRAM << DRAM	why?
bandwidth:	on-chip >> off-chip	

On a data access:

hit (data ∈ fast memory) ⇒ low latency access

miss (data ∉ fast memory) ⇒ long latency access *(DRAM)*

Two biggest problems in CS

- Cache invalidation
  - How to inform caches of stale data
- Naming things
- Off-by-one errors