

Chapter 7: Microarchitecture

Pipelined Performance

Pipelined Processor Performance Example

- **SPECINT2000 benchmark:**
 - 25% loads
 - 10% stores
 - 13% branches
 - 52% R-type
- **Suppose:**
 - 40% of loads used by next instruction
 - 50% of branches mispredicted
- **What is the average CPI?** (Ideally it's 1, but...)
 - CPI
 - Load CPI = 1 when not stalling, 2 when stalling
So, $\text{CPI}_{lw} = 0.6 \times 1 + 0.4 \times 2 = 1.4$
 - Branch CPI = 1 when branch not taken, 3 when mispredicting
So, $\text{CPI}_{beq} = 0.5 \times 1 + 0.5 \times 3 = 2$

$$\text{Average CPI} = (0.25)(1.4) + (0.1)(1) + (0.13)(2) + (0.52)(1) = \mathbf{1.23}$$

Branch CPI Calculation

$$\begin{aligned}\text{Branch CPI} &= \% \text{ of correctly predicted branches} \times 1\text{cc} \\ &\quad + \% \text{ of mispredicted branches} \times 3\text{cc} \\ &= 0.5 \times 1 + 0.5 \times 3\end{aligned}$$

$$\text{Branch CPI} = 2$$

Example 7.9 PIPELINED PROCESSOR CPI

The SPECINT2000 benchmark considered in Example 7.4 consists of approximately 25% loads, 10% stores, 11% branches, 2% jumps, and 52% R- or I-type ALU instructions. Assume that 40% of the loads are immediately followed by an instruction that uses the result, requiring a stall, and that 50% of the branches are taken (mispredicted), requiring two instructions to be flushed. Ignore other hazards. Compute the average CPI of the pipelined processor.

Solution The average CPI is the weighted sum over each instruction of the CPI for that instruction multiplied by the fraction of time that instruction is used. Loads take one clock cycle when there is no dependency and two cycles when the processor must stall for a dependency, so they have a CPI of $(0.6)(1) + (0.4)(2) = 1.4$. Branches take one clock cycle when they are predicted properly and three when they are not, so they have a CPI of $(0.5)(1) + (0.5)(3) = 2$. Jumps take three clock cycles (CPI = 3). All other instructions have a CPI of 1. Hence, for this benchmark, the average CPI = $(0.25)(1.4) + (0.1)(1) + (0.11)(2) + (0.02)(3) + (0.52)(1) = 1.25$.

Pipelined Performance Example

Program with 100 billion instructions

$$\begin{aligned}\text{Program Execution Time} &= (\# \text{ instructions}) \times \text{CPI} \times T_c \\ &= (100 \times 10^9)(1.23)(350 \times 10^{-12}) \\ &= \mathbf{43 \text{ seconds}}\end{aligned}$$

Processor Performance Comparison

Processor	Execution Time (seconds)	Speedup (single-cycle as baseline)
Single-cycle	75	1
Multicycle	155	0.5
Pipelined	43	1.7

Branch Prediction

- Ideal pipelined processor: $CPI = 1$
- Branch misprediction increases CPI
- **Static branch prediction:**
 - Check direction of branch (forward or backward)
 - If backward, predict taken Unrealistic!
 - Else, predict not taken Unrealistic!
- **Dynamic branch prediction:**
 - Keep **history** of last several hundred (or thousand) branches in *branch target buffer*, record:
 - Branch destination
 - Whether branch was taken

CPI Calculation Without Branch Prediction

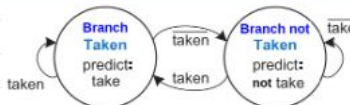
loop	Iteration	Branch	Next Instruction	Misprediction	Flushed Instr
1	1st	Taken	sub	yes	sub,sub
	2nd	Taken	sub	yes	sub,sub
	3rd	Taken	sub	yes	sub,sub
	4th	Taken	sub	yes	sub,sub
	5th	Taken	sub	yes	sub,sub
	6th	Taken	sub	yes	sub,sub
	7th	Not Taken	sub	no	-

Branch CPI = % of correctly predicted branches x 1cc
+ % of mispredicted branches x 3cc
= $\frac{1}{7} \times 1 + \frac{6}{7} \times 3$

Branch CPI = 2.71

Dynamic 1-bit Branch Prediction

- 1-bit branch predictor
- Starting State: Branch not Taken
- Loop 1 is transient as state changing is settling down.
Loop 2 onwards is the one more deterministic for CPI calculation

addi s0,zero,1	Branch Target Buffer								
addi s1,zero,1	bne s0,t0, while	BTA	State (1-bit)						
addi t0,zero, 128	Starting State		Branch Not Taken						
	loop	Iteration	Prediction in Fetch Stage	Correction in Execute Stage	State (1-bit)	Next Instruction	Misprediction	Flushed Instr	
while:	1	1st	predict: not take	taken	BranchTaken	sub	yes	sub,sub	
		2nd	predict: take	taken	BranchTaken	slli	no	-	
		3rd	predict: take	taken	BranchTaken	slli	no	-	
		4th	predict: take	taken	BranchTaken	slli	no	-	
		5th	predict: take	taken	BranchTaken	slli	no	-	
		6th	predict: take	taken	BranchTaken	slli	no	-	
		7th	predict: take	not taken	BranchNotTaken	slli	yes	slli/addi	
	2	1st	predict: not take	taken	BranchTaken	sub	yes	sub, sub	
2nd		predict: take	taken	BranchTaken	slli	no	-		
3rd		predict: take	taken	BranchTaken	slli	no	-		
4th		predict: take	taken	BranchTaken	slli	no	-		
5th		predict: take	taken	BranchTaken	slli	no	-		
6th		predict: take	taken	BranchTaken	slli	no	-		
7th		predict: take	not taken	BranchNotTaken	slli	yes	slli/addi		
sub ...									
sub ...									

Dynamic 1-bit Branch Prediction

- 1-bit branch predictor
- Starting State: Branch Taken
- Loop 1 is transient as state changing is settling down.
Loop 2 onwards is the one more deterministic for CPI calculation

addi s0,zero,1	Branch Target Buffer							
bne s0,t0, while	BTA	State (1-bit)						
addi s1,zero,1								
addi t0,zero, 128								
while: slli s0,s0,1 addi, s1,s1,1 bne s0,t0,while sub ... sub ...	Starting State		Branch Taken					
	loop	Iteration	Prediction in Fetch Stage	Correction in Execute Stage	State (1-bit)	Next Instruction	Misprediction	Flushed Instr
	1	1st	predict: take	taken	BranchTaken	slli	no	-
		2nd	predict: take	taken	BranchTaken	slli	no	-
		3rd	predict: take	taken	BranchTaken	slli	no	-
		4th	predict: take	taken	BranchTaken	slli	no	-
		5th	predict: take	taken	BranchTaken	slli	no	-
		6th	predict: take	taken	BranchTaken	slli	no	-
		7th	predict: take	not taken	BranchNotTaken	slli	yes	slli/addi
	2	1st	predict: not take	taken	BranchTaken	sub	yes	sub, sub
		2nd	predict: take	taken	BranchTaken	slli	no	-
		3rd	predict: take	taken	BranchTaken	slli	no	-
		4th	predict: take	taken	BranchTaken	slli	no	-
		5th	predict: take	taken	BranchTaken	slli	no	-
		6th	predict: take	taken	BranchTaken	slli	no	-
7th		predict: take	not taken	BranchNotTaken	slli	yes	slli/addi	

CPI Calculation with 1-bit Branch Prediction

- Let's consider stable loops ie Loop 2 onwards
- Any starting state in Loop 1 will not affect

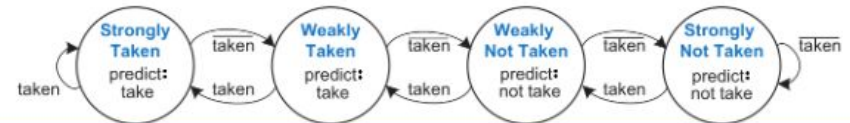
Branch CPI = % of mispredicted branches x 3cc
+ **% of correctly predicted branches x 1cc**

$$\begin{aligned}\text{Branch CPI} &= 2/7 \times 3\text{cc} + 5/7 \times 1 \\ &= \mathbf{1.57}\end{aligned}$$

Dynamic 2-bit Branch Prediction

- 2-bit branch predictor
- Starting State: Weakly Not Taken or Weakly Taken

Branch Target Buffer							
addi s0,z0	bne s0,t0, while	BTA	State (2-bit)				
addi s1,z0	Starting State		Weakly Not Taken				
addi t0,z0	loop	Iteration	Prediction in Fetch Stage	Correction in Execute Stage	State (2-bit)	Next Instruction	Misprediction
while:	1	1st	predict: not take	taken	WeaklyTaken	sub	yes
slli s0,s0		2nd	predict: take	taken	StronglyTaken	slli	no
addi, s1,s0		3rd	predict: take	taken	StronglyTaken	slli	no
bne s0,t0		4th	predict: take	taken	StronglyTaken	slli	no
sub ...		5th	predict: take	taken	StronglyTaken	slli	no
sub ...		6th	predict: take	taken	StronglyTaken	slli	no
		7th	predict: take	not taken	WeaklyTaken	slli	yes
	2	1st	predict: take	taken	StronglyTaken	slli	NO
		2nd	predict: take	taken	StronglyTaken	slli	no
		3rd	predict: take	taken	StronglyTaken	slli	no
		4th	predict: take	taken	StronglyTaken	slli	no
		5th	predict: take	taken	StronglyTaken	slli	no
		6th	predict: take	taken	StronglyTaken	slli	no
		7th	predict: take	not taken	WeaklyTaken	slli	yes



CPI Calculation with 2-bit Branch Prediction

- Let's consider stable loops ie Loop 2 onwards
- Any starting state in Loop 1 will not affect

Branch CPI = % of mispredicted branches x 3cc
+ % of **correctly predicted branches** x 1cc

$$\begin{aligned}\text{Branch CPI} &= 1/7 \times 3cc + 6/7 \times 1 \\ &= \mathbf{1.28}\end{aligned}$$

Total CPI for do-while loop with 2-bit Branch Prediction

Total CPI =

% of R/I-Type Instr. x CPI of R-Type Instr. + % of B-Type Instr. x CPI of B-Type Instr.

$$\begin{aligned} \text{\% of R/I-Type Instr.} &= \frac{\left[\underset{\text{L}}{2} \times \underset{\text{L}}{3} \underset{\text{I}}{\text{addi}} + \underset{\text{L}}{2} \times \underset{\text{I}}{7} \underset{\text{I}}{\text{sub}} \right] + \left[\underset{\text{L}}{2} \times \underset{\text{I}}{7} \underset{\text{I}}{\text{slli}} + \underset{\text{L}}{1} \underset{\text{I}}{\text{addi}} \right]}{\left[\underset{\text{L}}{2} \times \underset{\text{L}}{3} \underset{\text{I}}{\text{addi}} + \underset{\text{L}}{2} \times \underset{\text{I}}{7} \underset{\text{I}}{\text{sub}} \right] + \left[\underset{\text{L}}{2} \times \underset{\text{I}}{7} \underset{\text{I}}{\text{slli}} + \underset{\text{L}}{1} \underset{\text{I}}{\text{addi}} \right] + \left[\underset{\text{L}}{2} \times \underset{\text{I}}{7} \underset{\text{I}}{\text{bne}} \right]} \\ &= 38/(38+14)=0.73 \end{aligned}$$

$$\begin{aligned} \text{\% of B-Type Instr.} &= \frac{\left[\underset{\text{L}}{2} \times \underset{\text{I}}{7} \underset{\text{I}}{\text{bne}} \right]}{\left[\underset{\text{L}}{2} \times \underset{\text{L}}{3} \underset{\text{I}}{\text{addi}} + \underset{\text{L}}{2} \times \underset{\text{I}}{7} \underset{\text{I}}{\text{sub}} \right] + \left[\underset{\text{L}}{2} \times \underset{\text{I}}{7} \underset{\text{I}}{\text{slli}} + \underset{\text{L}}{1} \underset{\text{I}}{\text{addi}} \right] + \left[\underset{\text{L}}{2} \times \underset{\text{I}}{7} \underset{\text{I}}{\text{bne}} \right]} \\ &= 14/(38+14)=0.269 \end{aligned}$$

Total CPI =

% of R/I-Type Instr. x CPI of R/I-Type Instr. + % of B-Type Instr. x CPI of B-Type Instr.

$$0.73 \times 1 + 0.269 \times 1.28 = \mathbf{1.1}$$

```
addi s0,zero,1
addi s1,zero,1
addi t0,zero, 128

while:
slli s0,s0,1
addi, s1,s1,1
bne s0,t0,while

sub ...
sub
```

Total CPI for do-while loop with 1-bit Branch Prediction

Total CPI =

% of R/I-Type Instr. x CPI of R-Type Instr. + % of B-Type Instr. x CPI of B-Type Instr.

$$\% \text{ of R/I-Type Instr.} = \frac{\left[\underset{\text{L}}{2} \times \underset{\text{L}}{3} \text{ addi} + \underset{\text{L}}{2} \times \underset{\text{I}}{7} \times (1 \text{ slli} + 1 \text{ addi}) \right]}{\left[\underset{\text{L}}{2} \times \underset{\text{L}}{3} \text{ addi} + \underset{\text{L}}{2} \times \underset{\text{I}}{7} \times (1 \text{ slli} + 1 \text{ addi}) \right] + \left[\underset{\text{L}}{2} \times \underset{\text{I}}{7} \times (1 \text{ bne}) \right]}$$

$$\% \text{ of B-Type Instr.} = \frac{\left[\underset{\text{L}}{2} \times \underset{\text{I}}{7} \times (1 \text{ bne}) \right]}{\left[\underset{\text{L}}{2} \times \underset{\text{L}}{3} \text{ addi} + \underset{\text{L}}{2} \times \underset{\text{I}}{7} \times (1 \text{ slli} + 1 \text{ addi}) \right] + \left[\underset{\text{L}}{2} \times \underset{\text{I}}{7} \times (1 \text{ bne}) \right]}$$

Total CPI =

% of R/I-Type Instr. x CPI of R/I-Type Instr. + % of B-Type Instr. x CPI of B-Type Instr.

$$0.73 \times 1 + 0.269 \times 1.57 = \mathbf{1.19}$$

Total CPI =

$$\% \text{ of R/I-Type Instr.} \times \text{CPI of R-Type Instr.} + \% \text{ of B-Type Instr.} \times \text{CPI of B-Type Instr.}$$

$$\% \text{ of R/I-Type Instr.} = \frac{[2 \times (3 \text{ add} + 2 \text{ sub}) + 2 \times 7 \times (1 \text{ slli} + 1 \text{ addi})]}{[2 \times (3 \text{ add} + 2 \text{ sub}) + 2 \times 7 \times (1 \text{ slli} + 1 \text{ addi})] + [2 \times 7 \times (1 \text{ bne})]}$$

$$\% \text{ of B-Type Instr.} = \frac{2 \times 7 \times (1 \text{ bne})}{[2 \times (3 \text{ add} + 2 \text{ sub}) + 2 \times 7 \times (1 \text{ slli} + 1 \text{ addi})] + [2 \times 7 \times (1 \text{ bne})]}$$

Total CPI =

$$\% \text{ of R/I-Type Instr.} \times \text{CPI of R/I-Type Instr.} + \% \text{ of B-Type Instr.} \times \text{CPI of B-Type Instr.}$$

$$0.73 \times 1 + 0.269 \times 2.71 = 1.53$$

Pipelined Performance Example

Program with 52 instructions

without BP

$$\begin{aligned}\text{Program Execution Time} &= (\# \text{ instructions}) \times \text{CPI} \times T_c \\ &= (52)(1.53)(350 \times 10^{-12}) \\ &= \mathbf{27.84 \text{ nano seconds}}\end{aligned}$$

with 1-bit BP

$$\begin{aligned}\text{Program Execution Time} &= (\# \text{ instructions}) \times \text{CPI} \times T_c \\ &= (52)(1.19)(350 \times 10^{-12}) \\ &= \mathbf{21.65 \text{ nano seconds}}\end{aligned}$$

with 2-bit BP

$$\begin{aligned}\text{Program Execution Time} &= (\# \text{ instructions}) \times \text{CPI} \times T_c \\ &= (52)(1.1)(350 \times 10^{-12}) \\ &= \mathbf{20.02 \text{ nano seconds}}\end{aligned}$$

