Pipelined Instruction Execution
Part II: Performance

IF	ID	EXE	MEM	WB	
BNE					t
SUB	BNE				t+1
AND	SUB	BNE			t+2
idle	canceled	canceled	BNE		t+3
idle	idle	cancelled	canceled	BNE	t+4
LDR	idle	idle	cancelled	canceled	t+5
EOR	LDR	idle	idle	canceled	t+6
RSB	EOR	LDR	idle	idle	t+7

2

Performance Analysis

The number of cycles per instructions (CPI) is a standard measure of performance of microprocessors.

On an ideal pipelined unit, instructions execute at a rate of one per cycle. Thus, an ideal CPI is 1.

A simple way of calculating CPI is by determining the total number of cycles that take N instructions to execute (TC) and divide it by N.

Then, CPI = TC/N

53

Alternative Performance Analysis

The CPI can also be calculated in the following terms:

$CPI = CPI_{IDEAL} + CPI_{PENALTY} = 1 + CPI_{PENALTY}$

where $CPI_{PENALTY}$ is the contribution of the extra wasted cycles caused by pipeline interlocks.

In general, the $CPI_{PENALTY}$ is an aggregate of the wasted cycles introduced by structural, data and control hazards. Thus,

$CPI_{PENALTY} = Structural_{PENALTY} + Data_{PENALTY} + Control_{PENALTY}$

54

The Control Penalty

Every branch has the potential for generating wasted cycles due to miss predictions

We can define the control penalty as follows:

Let PB = % of instructions that are branches

BWC = average number of wasted cycles introduced by branch instructions.

Then,

$Control_{PENALTY} = PB \times BWC$

Typically $BWC = Prediction_{PENALTY}$

$Prediction_{PENALTY}$ = (% predictions not asserted) x (# of cycles wasted on a miss prediction)

55

Example Problem 1

The following piece of ARM code is executed 1000 times in a five stages pipeline with a Loop Buffer:

```
Back  ADD
      SUB
      BNE  Back
      ORR
      AND
```

If there are no structural or data hazards, determine the CPI for each of the following branch prediction schemes:

a) Normal ARM branch prediction (predict not taken)
b) Static, backward branches taken/forward branches not taken
c) 1 bit History Predictor
d) Two-bit Saturated Counter Predictor

56

Branch Penalty with a Loop Buffer

ARM code:

```
Back ADD
     SUB
     BNE  Back
     ORR
     AND
```

IF	ID	EXE	MEM	WB	
BNE					t
ORR	BNE				t+1
AND	ORR	BNE			t+2
ADD	canceled	canceled	BNE		t+3
SUB	ADD	canceled	canceled	BNE	t+4
BNE	SUB	ADD	canceled	canceled	t+5
ORR	BNE	SUB	ADD	canceled	t+6
AND	ORR	BNE	SUB	ADD	t+7

57

What we know

There are three instructions executed on each iteration of the loop, consuming one cycle each

N = (number of iterations) x (numbers of instructions executed per iteration)

N = 1000 x 3 = 3000

There are 2 additional cycles wasted each time the prediction is not asserted

58

Determining the Total Number of Wasted Cycles

Let

TC = total number of cycles that the 1000 iterations consume,

TIC = total number of cycles consumed by instructions

= (total numbers of iterations) x (number cycles consumed by instructions per iteration)

TWC = total number of wasted cycles

= (number of predictions not asserted) x (wasted cycles per prediction not asserted)

Then, TC = TIC + TWC

59

Normal ARM Branch

Since a branch not taken is predicted, the prediction is not asserted the first 999 iterations and asserted the last.

TIC = (total numbers of iterations) x (number cycles consumed by instructions per iteration)

TIC = 1000 x 3 = 3000

TWC = (number of predictions not asserted) x (wasted cycles per prediction not asserted)

TWC = 999 x 2 = 1998

TC = TIC + TWC = 3000 + 1998 = 4998

CPI = TC/N = 4998/(3000) = **1.67**

60

Static Prediction: Backward Branches Taken

Since a branch taken is predicted the prediction is asserted the first 999 iterations and not asserted the last.

TIC = 1000 x 3 = 3000

TWC = 1 x 2 = 2

TC = TIC + TWC = 3000 + 2 = 3002 cycles

CPI = TC/TI = 3002 /3000 = **1.00067**

61

1 Bit History Predictor

The behavior of the one bit history counter will be as follows:

Iteration	Counter Before	Prediction	Outcome of Prediction	Counter After
1	0	Not taken	Not asserted	1
2	1	Taken	Asserted	1
3	1	Taken	Asserted	1
4	1	Taken	Asserted	1
5	1	Taken	Asserted	1
999	1	Taken	Asserted	1
1000	1	Taken	Not asserted	0

62

1 Bit History Predictor

TIC = 1000 x 3 = 3000

TWC = 2 x 2 = 4

TC = TIC + TWC = 3000 + 4 = 3004 cycles

CPI = TC/TI = 3004 /3000 = **1.0013**

63

Two-bit Saturated Counter Predictor

The behavior of the saturated counter will be as follows:

Iteration	Counter Before	Prediction	Outcome of Prediction	Counter After
1	00	Not taken	Not asserted	01
2	01	Not taken	Not asserted	10
3	10	Taken	Asserted	11
4	11	Taken	Asserted	11
5	11	Taken	Asserted	11
999	11	Taken	Asserted	11
1000	11	Taken	Not asserted	10

Two-bit Saturated Counter Predictor

TIC = 1000 x 3 = 3000

TWC = 3 x 2 = 6

TC = TIC + TWC = 3000 + 6 = 3006 cycles

CPI = TC/TI = 3006 /3000 = **1.002**

Example Problem 1: Alternative Analysis

CPI = 1 + $CPI_{PENALTY}$

CPI = 1 + $Structural_{PENALTY}$ + $Data_{PENALTY}$ + $Control_{PENALTY}$

For this case there is no structural or data hazards. Then,

CPI = 1 + $Control_{PENALTY}$ = 1 + PB x BWC, where

PB = % of instructions that are branches

BWC = average number of wasted cycles introduced by branch instructions

Since the program is a loop of three instructions PB = 33% = .33

BWC = (% predictions not asserted) x (# of cycles wasted on a miss prediction)

Normal ARM Branch: Alternative Analysis

BWC = (% predictions not asserted) x (# of cycles wasted on a miss prediction)

BWC = (999/1000) x 2 = 1.998

CPI = 1 + PB x BWC = 1 + .33 x 1.998 = 1.67

Static Taken Prediction: Alternative Analysis

BWC = (% predictions not asserted) x (# of cycles wasted on a miss prediction)

BWC = (1/1000) x 2 = .002

CPI = 1 + PB x BWC = 1 + .33 x .002 = 1.00067

1 bit History Predictor: Alternative Analysis

BWC = (% predictions not asserted) x (# of cycles wasted on a miss prediction)

BWC = (2/1000) x 2 = .004

CPI = 1 + PB x BWC = 1 + .33 x .004 = 1.0013

Two-bit Saturated Counter Predictor: Alternative Analysis

BWC = (% predictions not asserted) x (# of cycles wasted on a miss prediction)

BWC = (3/1000) x 2 = .006

CPI = 1 + PB x BWC = 1 + .33 x .006 = 1.002

70

Summary of Prediction Schemes

	Predictions Not Asserted	Total Number Waisted Cycles	CPI
ARM Branch	999	1998	1.67
Static Predict Taken	1	2	1.00067
1 Bit History Predictor	2	4	1.0013
Two-bit Saturated Counter	3	6	1.002

- Although for this example the Two-bit Saturated Counter predictor did not result in better performance than the 1 bit history predictor and the static predictor, it is superior to them because it can better adapt to different branching outcome patterns.
- In comparison with the 1 bit History Predictor the Two-bit Saturated Counter will perform better in the long run if the loop is revisited.
  - *The 1-bit history will leave the loop predicting the next execution to be not taken (will be incorrect)*
  - *The two-bit saturated counter will leave the loop predicting the next execution to be taken (will be correct)*

71

Example Problem 2

For a particular ARM program branch instructions constitute 15% of all the instructions executed, and 60% of them are taken. Assume that the microprocessor has a typical 5 stage pipelined instruction execution unit and a memory latency of 4 cycles. Also assume that wasted cycles are only due control hazards.

a) Determine the CPI for the above case

b) Determine the CPI if the branch instruction is substituted with a delayed branch of one delay slot (the instructions following the branch that will always be executed) for which a useful instruction can be placed in the delay slot 75% of the time.

c) Determine the performance speedup resulting from replacing the original branch instruction with a delayed branch.

72

What we know

PB = % of instructions that are branches = 15% = .15

BWC = average number of wasted cycles introduced by branch instructions. Then,

$Control_{PENALTY}$ = PB x BWC

CPI = = 1 + $Control_{PENALTY}$ = 1 + PB x BWC = 1 + .15 x BWC

73

Solution of Part a

Since the ARM branch instruction predicts that the branch will not be taken, and 60% of the branches are taken, then, 60% (.6) of the predictions will not be asserted.

The penalty is the aggregate of the cycles wasted due to the 2 instructions cancelled and the 4 idle cycles due to the memory latency (total of 6).

BWC = (% predictions not asserted) x (# of cycles wasted on a miss prediction)

BWC = .6 x 6 = 3.6

CPI = 1 + .15 x BWC = 1 + .15 x 3.6 = 1 + .54 = **1.54**

74

Solution of Part b

Since a useful instruction cannot be placed on the delay slot 100% of the time, there will be a penalty due to the delayed branch even if the prediction is asserted.

Thus, BWC = $Delayed_{PENALTY}$ + $Prediction_{PENALTY}$

The delayed penalty will be one cycle for the 25% of the branches that cannot have a useful instruction in the delay slot. Then,

$Delayed_{PENALTY}$ = .25 x 1 = .25 cycles.

This time, every branch that miss the prediction introduces a penalty of 5 cycles (1 instruction canceled and the 4 idle cycles due to the memory latency). Since this happens for 60% of the branches,

$Prediction_{PENALTY}$ = .6 x 5 = 3 cycles.

Then, BWC = .25 + 3 = 3.25 cycles

CPI = 1 + .15 x BWC = 1 + .15 x 3.25 = **1.49**

75

Solution of Part c

Speedup = $CPI_{DELAYED\ BRANCH}$ - $CPI_{ARM\ BRANCH}$)/ $CPI_{ARM\ BRANCH}$

Speedup = (1.49- 1.54)/1.54 = - 3.2%

76

Lesson Outcomes

- Understand how a pipelined instruction execution unit works
- Know the three types of hazards and how are them detected
- Understand the basic mechanisms to reduce the penalty introduced by hazard interlocks
- Know how to analyze the performance of a processor in terms of CPI

77