Integrating the Timing Analysis of Pipelining and Instruction Caching

by

Christopher A. Healy
David B. Whalley
Florida State University

Marion G. Harmon
Florida A & M University

Instruction Cache Categorizations

- Always Miss
- Always Hit
- First Miss
- First Hit

Each assembly instruction in the program has a worst-case categorization for every loop level in which it occurs.

C and Assembly Code

C Source Code

```
double Square(x)
double x;
{
  return x * x;
}
```

SPARC Instructions

```
inst 0: save   %sp,(-72),%sp
inst 1: st     %i0,[%sp+.4_x]
inst 2: st     %i1,[%sp+(.4_x+4)]
inst 3: ldd    [%sp+.4_x],%f2
inst 4: fmuld  %f2,%f2,%f0
inst 5: ret
inst 6: restore
```

After Processing Instructions 0,1,2,...,6

Pipeline Diagram

		stage						
		IF	ID	EX	FEX	CA	WB	FWB
cycle	1	**0**						
	...	...						
	10	**0**						
	11		**0**					
	12			**0**				
	13					**0**		
	14						**0**	

After Processing Instructions 0,1,2,...,6

Pipeline Diagram

	1						
	1						
	1						
	1						
15	1						
...	...						
20	**1**						
21		**1**					
22			**1**				
23					**1**		
24					**1**		

After Processing Instructions 0,1,2,...,6

Pipeline Diagram

	2						
		2					
			2				
			2				
25					2		
26					**2**		

After Processing Instructions 0,1,2,...,6

Pipeline Diagram

		stage						
		IF	ID	EX	FEX	CA	WB	FWB
cycle	1	**0**						
	...	...						
	10	0						
	11	1	**0**					
	12	1		**0**				
	13	1				**0**		
	14	1					**0**	
	15	1						
	...	...						
	20	1						
	21	2	1					
	22	3	2	1				
	23	4	3	2		1		
	24	4	3	2		1		
	25	5	4	3		2		
	26	5	4	3		2		
	27	5	4			3		
	28	5	4			3		
	29	**6**	5		**4**			**3**
	30		**6**		4			
	31			**6**	4			
	32				4	**6**		
	33				4		**6**	
	34				4			
	35				**4**			
	36							**4**

Worst-Case Loop Analysis Algorithm

- First *p* iterations
  - Find longest continue path, and identify first misses and first hits encountered.
  - Concatenate pipeline information onto end of loop's worst case union.
- Middle *n* - 1 - *p* iterations
  - All first hits and first misses have been encountered.
  - Replicate iteration *p* (*n* - 1 - *p*) times.
- Final iteration
  - Find longest exit path, and identify first misses and first hits encountered.
  - Concatenate pipeline information onto end of loop's worst case union.
  - Store WCET and union with the appropriate exit block.

Continue Path versus Exit Path

Continue Path Pipeline Diagram

		stage						
		IF	ID	EX	FEX	CA	WB	FWB
cycle	1	**6**						
	2	7	**6**					
	3	8	7	**6**				
	4	11	8			**6**		
	5	12	11	8			**6**	
	6	13	12	11		8		
	7	13	12	11		8		
	8	14	13	12		11		
	9	14	13	12		11		
	10	14	13			12		**11**
	11	15	14		**13**			12
	12	15	14		13			
	...	15	14		13			
	23	15	14		13			
	24	16	15		14			13
	25	17	16	15	14			
	...	17	16	15	14			
	79	17	16	15	**14**			
	80	18	17	16		15		**14**
	81	18	17	16		15		
	82	18	17	16		15		
	83	**19**	18	17		16		
	84		**19**			17	16	
	85			**19**			17	
	86					**19**		
	87						**19**	

Exit Path Pipeline Diagram

		stage						
		IF	ID	EX	FEX	CA	WB	FWB
cycle	1	**6**						
	2	7	**6**					
	3	**8**	7	**6**				
	4		**8**			**6**		
	5			**8**			**6**	
	6					8		
	7					**8**		

C Source Code

```
main()
{
  double a[5];
  int i;

  for (i = 0; ; i++)
  {
    if (i == 4)
      break;
    else
      a[i] = i / 2.0;
  }
}
```

First Miss Transition in Worst Case

Worst-Case Pipeline Diagram

		stage						
		IF	ID	EX	FEX	CA	WB	FWB
cycle	1	**13**						
	2	14	**13**					
	3	15	14	**13**				
	4	16	15	14		**13**		
	5	16	15	14		13		
	6	**17**	16	15		14		
	7		**17**			15	**14**	
	8			**17**			15	
	9					**17**		
	10						**17**	

- Instruction 16 in inner loop is a first miss in both loop levels.
- Treat as **hit** in the inner loop's pipeline analysis.
- In inner loop, **add** miss penalty to first iteration.
- **Subtract** miss penalty from inner loop time on each outer loop iteration after the first.
- One-cycle overestimation of WCET

Future Work

- Verify predictions with logic analyzer
- Wrap-around filling of cache lines
- Data caching
- More accurate calculations of loop iterations
- Retargetability
- Obtaining timing predictions more quickly
- Automatic detection of some data dependencies

Algorithm to Calculate Cache States

- input_state(top) = all invalid lines
- WHILE any change DO
- FOR each basic block instance B DO
- input_state(B) = NULL
- FOR each immed pred P of B DO
- input_state(B) += output_state(P)
- output_state(B) =
- (input_state(B) + prog_lines(B))
- - conf_lines(B)

Figure 4 : C and Assembly Code

C Source Code

```
main()
{
   int i, cnt = 0;
   double dcnt = 0.0;
   extern int incr;
   extern double dincr;

   for (i=0; i < 10; i++)
     if (i < 5)
        dcnt += dincr;
     else
        cnt += incr;
}
```

Assembly Code

```
Inst    Assembly Code
----    -------------------------
  0     mov   %g0,%o1
  1     sethi %hi(L01),%o0
  2     ldd   [%o0+%lo(L01)],%f2
  3     mov   %g0,%o2
  4     sethi %hi(_dincr),%o3
  5     sethi %hi(_incr),%o4
  6     cmp   %o2,5
  7 L8: bge,a L9
  8     ld    [%o4+%lo(_incr)],%o0
  9     ldd   [%o3+%lo(_dincr)],%f0
 10     ba    L6
 11     faddd %f2,%f0,%f2
 12 L9: add    %o1,%o0,%o1
 13 L6: add    %o2,1,%o2
 14     cmp   %o2,10
 15     bl,a  L8
 16     cmp   %o2,5
 17     retl
 18     nop
```

First Miss Transition in Best Case

Best-Case Pipeline Diagram

		stage						
		IF	ID	EX	FEX	CA	WB	FWB
cycle	1	**13**						
	2	14	**13**					
	3	15	14	**13**				
	4	16	15	14		**13**		
	5	16	15	14		13		
	6	16		15		14		
	7	16				15	**14**	
	8	16					15	
	9	16						
	...	...						
	13	16						
	14	**17**	16					
	15		**17**					
	16			**17**				
	17					**17**		
	18						**17**	

- Instruction 16 in inner loop is a first miss in both loop levels.
- Treat as a **miss** in inner loop's pipeline analysis.
- In inner loop, **subtract** miss penalty from each iteration after the first.
- In outer loop, **subtract** miss penalty from each iteration after the first.
- Underestimation in BCET of 1 cycle for each of outer loop's iterations after the first.