Daniel McDonough (dmcdonough)
3/29/19
CS4515
HW3

[25] <3.7, 3.8> In this exercise, you will explore performance trade-offs between three processors that each employ different types of multithreading (MT). Each of these processors is superscalar, uses in-order pipelines, requires a fixed three-cycle stall following all loads and branches, and has identical L1 caches. Instructions from the same thread issued in the same cycle are read in program order and must not contain any data or control dependences.

- *Processor A is a superscalar SMT architecture, capable of issuing up to two instructions per cycle from two threads.*
- *Processor B is a fine-grained MT architecture, capable of issuing up to four instructions per cycle from a single thread and switches threads on any pipeline stall.*
- *Processor C is a coarse-grained MT architecture, capable of issuing up to eight instructions per cycle from a single thread and switches threads on an L1 cache miss.*

Our application is a list searcher, which scans a region of memory for a specific value stored in R9 between the address range specified in R16 and R17. It is parallelized by evenly dividing the search space into four equal-sized contiguous blocks and assigning one search thread to each block (yielding four threads). Most of each thread's runtime is spent in the following unrolled

loop body:

```
loop: lw x1,0(x16)
lw x2,8(x16)
lw x3,16(x16)
lw x4,24(x16)
lw x5,32(x16)
lw x6,40(x16)
lw x7,48(x16)
lw x8,56(x16)
beq x9,x1,match0
beq x9,x2,match1
beq x9,x3,match2
beq x9,x4,match3
beq x9,x5,match4
beq x9,x6,match5
beq x9,x7,match6
beq x9,x8,match7
DADDIU x16,x16,#64
blt x16,x17,loop
```

Assume the following:

- *A barrier is used to ensure that all threads begin simultaneously.*
- *The first L1 cache miss occurs after two iterations of the loop.*
- *None of the BEQAL branches is taken.*
- *The BLT is always taken.*

All three processors schedule threads in a round-robin fashion.
Determine how many cycles are required for each processor to complete the first two iterations of the loop.

First Two Cycles Unscheduled:

Clock Cycle	Instruction
1	DADDIU R4,R1
2	LW F2,R1
3	STALL
4	Multiply F4,F2,F0
5	LW F6,R2
6	STALL
7	ADD F6,F4,F6
8	STALL
9	STALL
10	STALL
11	SD F6,R2
12	DADDIU R1,R1
13	DADDIU R2,R2
14	DSLTU R3,R1,R4
15	STALL
16	BNEZ R3,loop

Processor A:

Here there are many stall cycles because of the name and data dependences required for the next instruction and the lack of parallelism. SMT architecture allows separate threads to achieve parallelism. Using separate threads we can do two cycles in two separate threads and then combine them. This causes the original 16 clock cycle to become a **10 clock cycle loop**

First Two Cycles Simultaneously:

Clock Cycle	Instruction
1	DADDIU R4,R1
2	LW F2,R1
3	LW F6,R2
4	Multiply F4,F2,F0
5	DADDIU R1,R1
6	DADDIU R2,R2
6	DSLTU R3,R1,R4
6	STALL
6	STALL

7	ADD F6,F4,F6
8	STALL
9	STALL
10	BNEZ R3,loop

Processor B:

Using a fine-grained multi-threading scheme, we can issue instructions from a different thread each cycle, but all the instructions issued together in a cycle must come from the same thread. Here to remove all stalls the loop must be unrolled 3 times. Here it takes **19 clock cycles**

First Two Cycles Fine-grained:

Clock Cycle	Instruction
1	DADDIU R4,R1
2	LW F2,R1
3	LW F6,R2
4	Multiply F4,F2,F0
5	LW F2,R1
6	LW F10 R2
7	Multiply F8,F2,F0
8	LW F2,R1
9	LW F14,R2
10	Multiply F12,F2,F0
11	Add F6,F4,F6
12	DADDIU F10,F8,F10
13	Add F10,F8,F10
14	DADDIU
15	DSLTU
16	ADD
17	S.D F6,R2
18	S.D F10,R2
19	BNEZ

Processor C:

In "coarse-grain multi-threading" we can issue instructions from one thread for many cycles and then switch to issuing from another thread only when the first thread gets a cache miss or some other long-latency operation. We can loop unroll two cycles to get **8 cycles** for the first two iterations.

Clock Cycle	Memory Reference 1	Memory Reference 2	Operation 1	Operation 2	Branch
1	LW F1,R1	LW F2,R2			
2	LW F3,R1	LW F4,R2	Multiply F1,F2,F0	Multiply F1,F2,F0	
3			Multiply F3,F4,F0	Multiply F3,F4,F0	DADDIU
4					DADDIU
5			ADD	ADD	
6	S.D	S.D	ADD	ADD	
7	S.D	S.D			
8					BNEZ R3,loop