4DM4 Ungraded Assignment #3

DAXPY Loop on a Multi-Threaded Processor (v1)

Out: Wednesday, Nov. 23, 2022
(Solutions will be taken up in a class lecture to be announced.)

In 4DM4 Assignment #2, we looked at the DAXPY loop in a static-scheduled linear pipeline. (DAXPY stands for '*Double-Precision A*X Plus Y'* (a central loop in Gaussian elimination).

In Assignment #3, we will look at variations of the DAXPY loop in (a) a single-core (processor) using Dynamic-Scheduling and multiple-issue, and in (b) a MULTI-THREADED core (processor) using Dynamic-Scheduling and multiple-issue, like the INTEL CORE i7 processor.

Here is a modified version of the basic DAXPY loop, with 8 instructions:

X(i) = a*X(i)+Y(i)

loop:	LD F0,0(R1)	Load X(i)
	MULTD F0,F0,F31	Multiply a*X(i) (a is in F31)
	LD F1,0(R2)	Load Y(i)
	ADDD F0,F0,F1	add a*X(i) + Y(i)
	SD 0(R2), F0	store Y(i)
	SUBI R1,R1,#8	decrement X address
	SUBI R2,R2,#8	decrement Y address
	BGEZ R1,loop	loop if not done

We will use some new 'Timing Table' formats for this assignment. Blank timing-tables for Q1, Q2 and Q3 are attached at the end of this assignment, and will be available on the class web-site.

In this assignment, all branches always resolve quickly. Specifically, the branches are resolved before any speculative instructions finish execution. The speculative instructions therefore become non-speculative before they finish, so they can perform the normal Write-Back (WB), and they ignore the COMMIT stage.

Question #1: DAXPY Loop on a Single-Threaded Dynamically-Scheduled Machine

Assumptions:
- one FP-ADD, 2-stage pipeline,
- one FP-MULD, 4-stage pipeline.
- one FP-DIVD, 10 stages, non-pipelined
- many INT-ALU units, 1 stage, to compute INT results
- 3 MEM units, each a 2-stage pipeline, for Loads and Stores
- WB over the CDB takes 1 cc, and many WBs can occur in 1 cc

- LD instructions can bypass the MEM-RS, when a MEM unit is available (always, in this case)
- SD instructions with a TAG for an operand must wait in the MEM-RS

• 3 shared Reservation-Stations (RS), as follows:
• one FP-RS, for ADDD and MULD instructions, with capacity = 8 instructions
• one FP-RS for DIVD instructions, with capacity = 1 instruction
• one MEM-RS with capacity = 16 instructions (for SD instructions)

• FAST-FORWARDING for INT results; INT results are forwarded at the end of the cc in which they are produced, in 0 clock-cycles.

• SLOW-FORWARDING for FP results; FP forwarding occurs during WB, and results are broadcasted over the CBD, which takes 1 cc. FP results can be used in the next clock cycle after the WB unit completes.

• an 8-ISSUE machine, with '**FULL-ISSUES**'; 8 instructions in 1 loop iteration must issue together, or stall together; (we do not allow '**PARTIAL ISSUES'** of a loop, with less than 8 instructions; this simplifies our tables.)

• BRANCHES: Cancelling branches, with PREDICT-TAKEN, with perfect branch prediction
• SPECULATIVE execution; the COMMIT stage takes 1 cc (when necessary)

• 4 GHz clock rate

• Q1A: Complete an execution table for Q1A for several iterations of the DAXPY loop, on a SINGLE-THREADed machine. In the last 3 columns labelled 'RS occupancy', we enter 2 times for each instructions that must wait in a RS, the (Entry,Exit) times.
Entry-time: Assume instructions enter the RS during the issue cc (if they enter the RS).
Exit-time: The exit-time is the last cc in which the instruction resides in the RS, before moving into a FU at the next rising edge of the clock cycle.

• For a TAG, show 3 fields (Register, Instruction, WB clock cycle), as we did in the class notes, if you prefer completeness. However, the Instruction field is not necessary; it is shown for completeness. For a TAG, you can show only 2 fields (Register, WB clock cycle), if you prefer to write less.

• Q1B: Will this loop reach a 'STEADY-STATE' ? If so, explain it. If not, explain it.

• Q1C: Can you compute or estimate the sustained performance, in INT instructions per second, and in FLOPS per second?

• (HINT: Completing the tables for Q1 can take 1/2 an hour. Completing the tables for Q2 can take 1/2 an hour. To save time, you can have one group member complete Q1, and have a second group member complete Q2, and you can both work together to complete Q3)

							RS occupancy		
Thread #1	Issue	EX	MEM	WB	TAG wrtten	TAGs read	MEM	ADDD, MULD	DIVD
LD F0, 0(R1)									
MULD F0,F0,F31									
LD F1,0(R2)									
ADDD F0,F0,F1									
SD F0,0(R1)									
SUBI R1,#8									
SUBI R2,#8									
BGEZ R1, loop									
						RS OCCUPANCY =>			

FIG 1: Execution table for Q1A.

Question #2: Modified DAXDY Loop on a Single-Threaded Dynamically-Scheduled Machine

Consider a new loop called DAXDY, which stands for '*Double-Precision A*X Divided by Y*'. Here is the basic DAXDY loop :

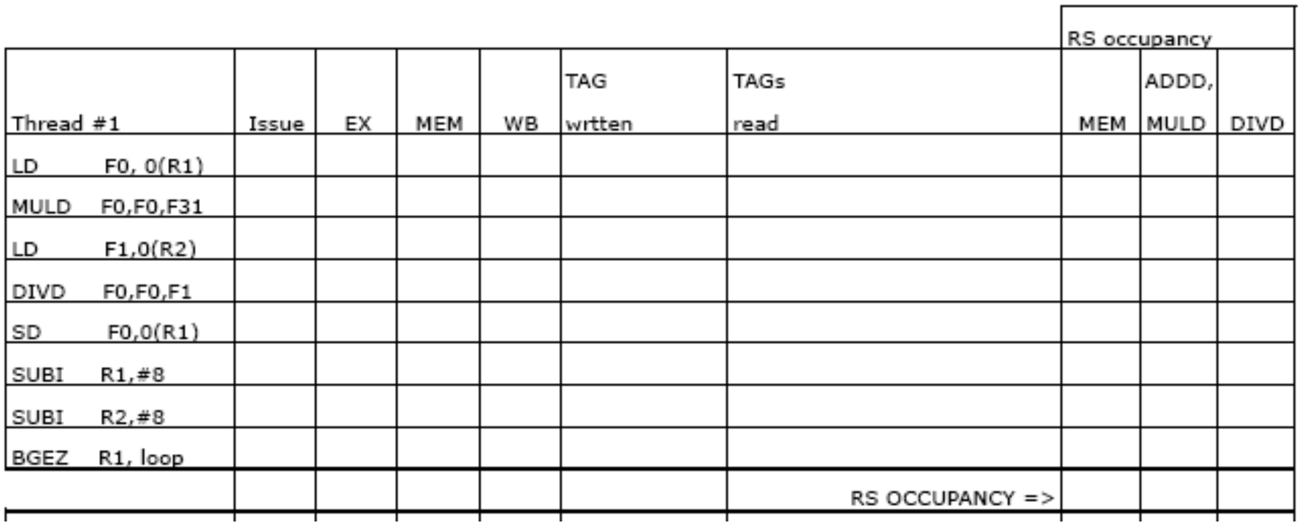
X(i) = a*X(i)/Y(i):

loop:	LD F0,0(R1)	Load X(i)
	MULTD F0,F0,F31	Multiply a*X(i)
	LD F1,0(R2)	Load Y(i)
	DIVD F0,F0,F1	compute a*X(i) /Y(i)
	SD 0(R2), F0	store Y(i)
	SUBI R1,R1,#8	decrement X address
	SUBI R2,R2,#8	decrement Y address
	BGEZ R1,loop	loop if not done

• Q2A: Complete the execution table for several iterations of the DAXDY loop, on a SINGLE-THREADed machine.

• Q2B: Will this loop reach a 'STEADY-STATE' ? If so, explain it. If not, explain it.

• Q2C: Can you compute or estimate the sustained performance, in INT instructions per second, and in FLOPS per second?



							RS occupancy		
Thread #1	Issue	EX	MEM	WB	TAG wrtten	TAGs read	MEM	ADDD, MULD	DIVD
LD F0, 0(R1)									
MULD F0,F0,F31									
LD F1,0(R2)									
DIVD F0,F0,F1									
SD F0,0(R1)									
SUBI R1,#8									
SUBI R2,#8									
BGEZ R1, loop									
						RS OCCUPANCY =>			

FIG. Execution table for Q2A.

<u>Question #3 : Two Loops on a Double-Threaded Dynamically-Scheduled Machine</u>

The goal of this question is to determine how the performance changes due to simultaneous MULTI-THREADING. Consider a Multi-Threaded processor, which supports 2 threads. The first thread is the DAXPY loop in Q1, which runs forever. The second thread is the DAXDY loop in Q2, which runs forever.

<u>Additional Assumptions:</u>
• The controller can switches threads every clock cycle, in a round-robin order, when the threads are not stalled and can issue.
• If a thread is stalled, it is skipped in the round-robin order.
• If no thread is ready to issue, then no instructions will issue, which is equivalent to a wasted bundle or a stalled bundle, since no new issues occur.
• RS sizes remains unchanged.

We'll use 2 execution tables in Q3. One table shows the execution-times for Thread 1, the other shows the execution-times for thread 2. When a thread issues in a clock cycle, complete the table for one loop iteration for that thread. For that clock cycle, show the RS occupancy for both threads, so that the reader can how full all the RSs are, after that clock cycle. During one clock cycle, the number of instructions from thread 1 waiting in the RSs are explicitly shown in the row 'RS OCCUPANCY (Thread 1)'. The number of instructions from thread 2 waiting in the RS are explicitly shown in the row 'RS OCCUPANCY (Thread 2)'.

• Q3A: Complete the execution table for several clock cycles, with enough iterations of each loop so that you can answer the rest of this question <u>with reasonable confidence</u>. We will accept reasonable arguments, based upon observations of several iterations.

• Q3B: Will this system reach a STEADY-STATE ? If so, explain it. If not, explain it.

• Q3C: Can you compute or estimate the sustained performance, in INT instructions per second, and in FLOPS per second?

NOTE: You may need to make some reasonable assumptions for any questions. Your ability to make reasonable assumptions in the design of a processor is important in this course.

WHAT TO SUBMIT:

This assignment is “ungraded”, for several reasons: (1) There is a CUPE strike in 2022, so most of our TAs are not available. (2) Even if all of our TAs were available, with 3 TAs for a class of over 100 students, as we would not have enough TAs to grade a 3rd assignment.

The prof will take up the solutions to this assignment in a class lecture, and you can learn the material by trying to do this assignment, before seeing the solutions.

Here are some tables for Q3:

							RS occupancy		
Thread #1	Issue	EX	MEM	WB	TAG written	TAGs read	MEM	ADDD, MULD	DIVD
LD F0, 0(R1)									
MULD F0,F0,F31									
LD F1,0(R2)									
ADDD F0,F0,F1									
SD F0,0(R1)									
SUBI R1,#8									
SUBI R2,#8									
BGEZ R1, loop									
						RS OCCUPANCY (Thread 1) =>			
						RS OCCUPANCY (Thread 2) =>			

							RS occupancy		
Thread #2	Issue	EX	MEM	WB	TAG written	TAGs read	MEM	ADDD, MULD	DIVD
LD F0, 0(R1)									
MULD F0,F0,F31									
LD F1,0(R2)									
DIVD F0,F0,F1									
SD F0,0(R1)									
SUBI R1,#8									
SUBI R2,#8									
BGEZ R1, loop									
						RS OCCUPANCY (Thread 1) =>			
						RS OCCUPANCY (Thread 2) =>			

FIG. Execution Tables for Threads 1 and 2, in Q3.