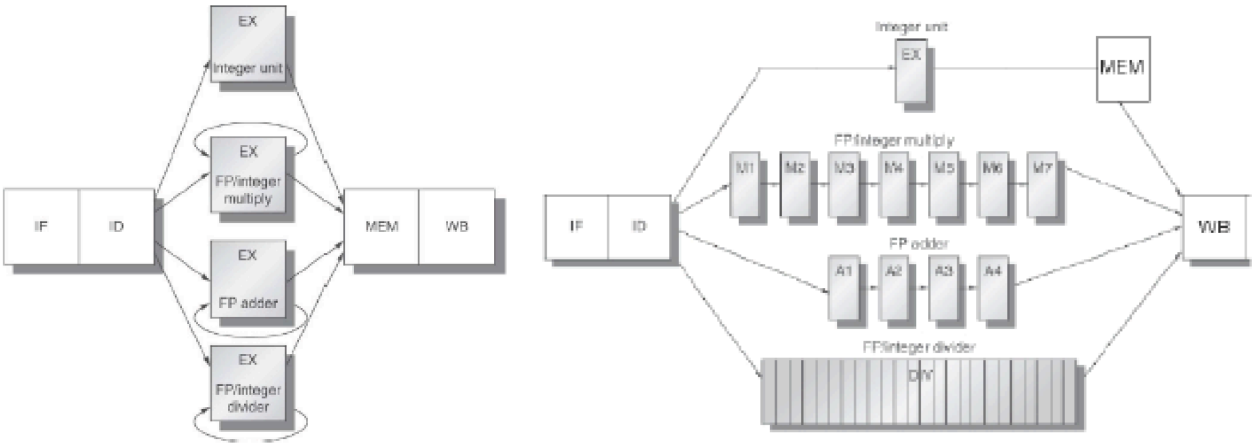
4DM4 - Computer Architecture - Assignment #2, 2022

Advanced Static Pipelining

Version 1c, Posted: Wed., Nov. 2 2022 (changes shown in red text.)
Due date: Tues, Nov. 8, 2022 (no late assignments accepted)
(Midterm: Tues, Nov. 15, 2022, 54:30-7:20pm, MDCL tentative)



(a) (b)

Fig 1. (a) Basic 5-stage pipeline, with 3 multi-cycle EX units.
(b) Basic 5-stage pipeline, with 3 multi-cycle EX units (detailed view).

Fig. 1 illustrates the basic 5-stage pipeline, along with several 'Multi-Cycle' EX units (see 4DM4 Lecture 11). These multi-cycle EX units perform FP/INT multiply, FP-ADD, and FP/INT division, and take several clock-cycles to compute their results. (In contrast, the INT EX unit takes 1 cc to compute its results.) Typically, the FP/INT-MULT and FP-ADD are fully pipelined - ie a new instruction can enter each pipeline, in each clock cycle. Typically, the FP-DIV is 'un-pipelined', ie a new FP-DIV instruction can enter the pipeline, only once the last FP-DIV instruction has left the pipeline. Assignment #2 will explore scheduling in multi-cycle pipelines.

ASSIGNMENT 2: The DAXPY Gaussian Elimination loop, on a 7-stage pipeline.

A.12 [20/22/22] <A.4, A.6> In this exercise, we will look at how a common vector loop runs on statically and dynamically scheduled versions of the MIPS pipeline. The loop is the so-called DAXPY loop (discussed extensively in Appendix G) and the central operation in Gaussian elimination. The loop implements the vector operation $Y = a \times X + Y$ for a vector of length 100. Here is the MIPS code for the loop:

```
foo:  L.D      F2,0(R1)      ;load X(i)
      MULT.D   F4,F2,F0      ;multiply a*X(i)
      L.D      F6,0(R2)      ;load Y(i)
      ADD.D    F6,F4,F6      ;add a*X(i) + Y(i)
      S.D      0(R2),F6      ;store Y(i)
      DADDUI   R1,R1,#8      ;increment X index
      DADDUI   R2,R2,#8      ;increment Y index
      DSGTUI   R3,R1,done    ;test if done
      BEQZ     R3,foo        ;loop if not done
```

Assumptions: Assume a basic 7 stage pipeline, as follows:
(1) IF is a 2-stage pipeline (stages F1, F2).
(2) ID is a 1-stage pipeline.
(3) Integer EX unit is 1-stage, with full forwarding of data to any stage.
(4) One FP-ADD unit, a 3-stage pipeline.
(5) One FP-MULT unit, a 6-stage pipeline.
(6) MEM is a 2-stage pipeline (stages M1, M2); WB is one-stage.
(7) Instructions that don't need the MEM stage can bypass MEM and go directly to WB.
(8) WB stage accepts <= 2 instructions per clock cycle (cc), and perform 2 writes per cc.
(9) Full forwarding of FP results, at the end of the cc in which they are produced.
(10) WB completes in 1st half of cc, D stage reads in 2nd half of cc (no forwarding needed here)
(11) 32 INT registers (R0 … R31), and 32 double-precision FP registers (F0…F31)
(12). The BNEZ instruction resolves in the ID stage. It does not need to use EX, MEM or WB stages. If an operand is not ready when the BNEZ enters the ID stage, we do not stretch the clock; the BNEZ will stall for 1 cc, and wait for the operand in the next cc.
(13) The No-op instructions only use stages F1 and F2. They do not need to use the EX, MEM or WB stages (ie hardware will detect the 'no-op' instructions and ignore them.)
(14) If you need to make any assumptions to proceed, please label each assumption (ie 'My-Assumption-#1') and highlight each assumption, so the TA can see it.

TYPICAL FP INSTRUCTIONS used in Q1:
- L.D is a double precision Load (8 bytes); S.D is a double precision Store,
- DADDUI is an integer ADD, DSGTUI is the integer Set-Greater-Than,
- 'done' is a immediate constant, and BEQZ is the Branch-Equal-Zero instruction.

Part (a): DAXPY Loop, No Unrolling, with No Scheduling
Use the excel tables provided on Avenue*. Create a regular timing diagram for one iteration of the DAXPY loop, as shown below. Label all data forwardings with a vertical line in the diagram, (or with 2 asterisks * or 2 double asterisks **), from the producing stage to the consuming stage of the data. Label all stalls. Please add a clear comment for every data forwarding, ie • *forward R0 (from EX* to *M) in cc 7*, below the table. Compute the clock cycles required per iteration of the loop. Compute the MFLOP rating, assuming a 3 GHz clock. (See 4DM4 Lecture 14.)

Regular Timing diagram format:

Instruction	Clock Cycle													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
loop: L.D F2, 0(R1)														
MULT.D F4, F2, F0														
L.D F6, 0(R2)														
ADD.D F6, F4, F6														
S.D 0(R2), F6														
DADDUI R1, R1, #8														
DADDUI R2, R2, #8														
DSGTUI R3, R1, done														
BEQZ R3, loop														
No-op (how many no-ops?)														

Part (b): DAXPY Loop, No Unrolling, with Scheduling

Repeat part (a), but schedule the code to reduce stalls (but do not unroll the loop). If you make any assumptions, highlight them so that the TA can see them.

For the following parts (c), (d) and (e), the regular timing-diagram format will result in large tables. Please use the following Compressed-Timing-Table format. It contains the same information. You can complete the regular timing diagram using the regular format to make sure you understand how it works, and then you can enter the results to the new compressed-table. Use the excel tables provided on Avenue*.

Compressed Timing Table format:

Instruction Slot #1	IF (F1,F2)	ID	EX (Int, FP)	MEM (M1,M2)	WB	Comment/Hazard
loop: L.D F2, 0(R1)						
MULT.D F4, F2, F0						
L.D F6, 0(R2)						
ADD.D F6, F4, F6						
S.D 0(R2), F6						
DADDUI R1, R1, #8						
DADDUI R2, R2, #8						
DSGTUI R3, R1, done						
BEQZ R3, loop						
No-Op (how many ?)						
loop: start next iteration						

Part (c): DAXPY Loop, With Unrolling, with no Scheduling

Repeat part (a), but unroll the loop to allow 4 iterations to occur in one new loop iteration, but do not schedule the code. Use the excel tables provided on Avenue*.

Part (d): DAXPY Loop, With Unrolling, and with Scheduling

Repeat part (c), but this time schedule the code. Use the excel tables provided on Avenue*.

Part (e): DAXPY Loop, With Unrolling and Scheduling. On Dual-Issue Machine

Consider a dual-issue machine, where 2 instructions can issue per clock cycle. A compressed timing-table for a 2-issue machine is shown below. The 1st column contains only INT instructions or LD, SD instructions. The 2nd column can contain INT or FP-arithmetic instructions.

Unroll the loop to allow 4 iterations to occur in one new loop iteration, and schedule the code for maximum performance. Use the same assumptions as before, but add these **NEW ASSUMPTIONS:**

- the 2 instructions per row issue together or stall together (ie they pass through the ID stage together)
- there are 2 INT EX units, each completes in 1 cc
- only 1 LD or SD instruction can enter the MEM pipeline per clock cycle

• we can perform 2 write-backs per clock cycle

Use the excel tables provided on Avenue. I will post excel diagrams for timing-tables on our web-site, so that you can edit these.

Instruction Slot #1	Instruction Slot #2	F1,F2	ID	EX	M1,M2	WB	Comment/Hazard
loop: L.D F2, 0(R1)							
MULT.D F4, F2, F0							
L.D F6, 0(R2)							
ADD.D F6, F4, F6							
S.D 0(R2), F6							
DADDUI R1, R1, #8							
DADDUI R2, R2, #8							
DSGTUI R3, R1, done							
BEQZ R3, loop							
No-op (how many ?)							

SUBMISSION FORMAT:

Please reserve a folder for Assignment #2, for your group, on Avenue. The folder will likely have a prefix 'CAT1" (which stands for CATEGORY-1 in Avenue.)

Please work in groups of 2. Please clearly state your group members on all documents.
It may help to include the text into your filenames, such as: (a) 'Group-X', where X is your group number, and (b) the text 'XX,YY', where XX,YY are the initials of your group members.

Put your excel files (or PDF files) into the folder. You can put all tables into one PDF file, if it is manageable. Alternatively, you can have many separate files, ie a separate file for parts (a), (b), (c), etc.

• Please label your files carefully, so that the TAs can find them easily. If you use different excel tables with different formats, or if your files are hard to find, then you may some loose marks (proportional to how much time the TAs spend trying to find and manage your files).