

Advanced Computer Architecture CMSC 611

Homework 3

Due in class Oct 17th, 2012

(Show your work to receive partial credit)

- 1) For the following code snippet list the data dependencies and rewrite the code to resolve name dependencies. (15 points)

```

Loop: LD    R2, 0(R7)

      Add   R1, R2, R3

      Sub   R4, R6, R1

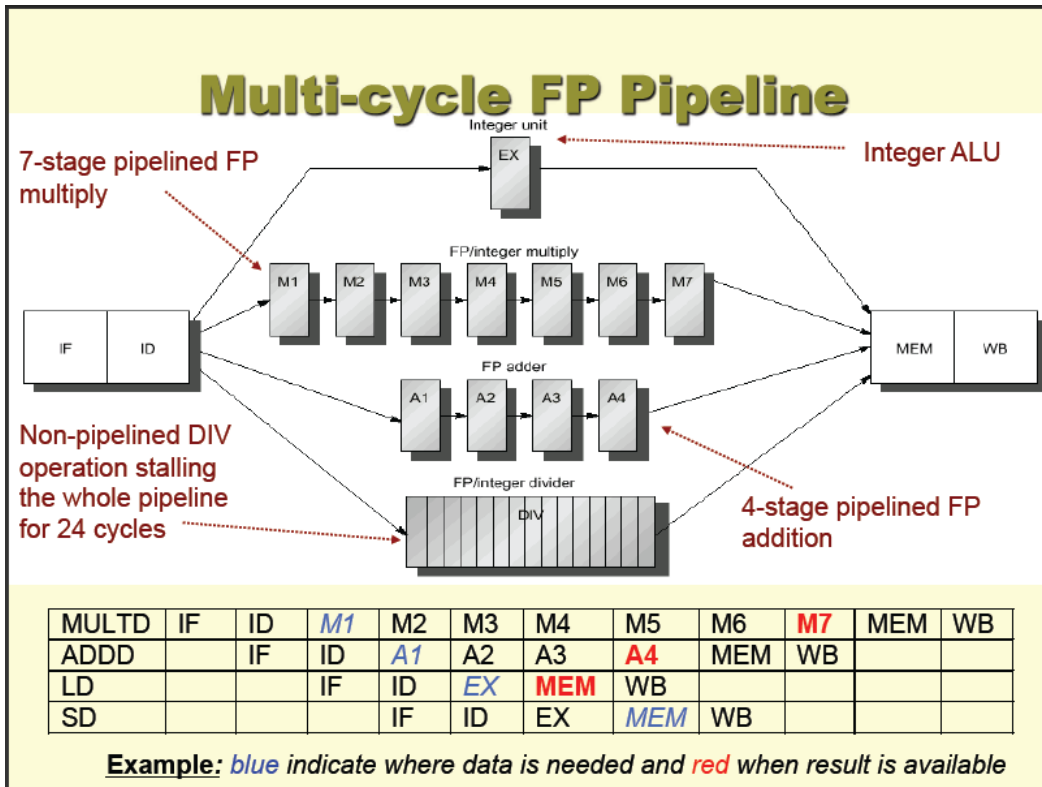
      Add   R2, R5, R6

      LD    R4, 32(R1)

      SD    36(R1), R2

      BEQ   R4, Loop
    
```

- 2) From the course slides. (Multi-cycle FP pipeline for MIPS) (45 points)



- Use the instruction latencies as indicated in the selected slide to first show all the stalls that is present in the following piece of code if the branch is not taken.
- Now unroll this loop as many times as needed and schedule the instructions to remove all the stalls. You may rename registers i.e. use new registers and/or change the immediate/offset values. You can ignore structural hazards as well.
- What is the speed up that you achieved after unrolling and scheduling?

```

Loop:      LD      F0, 0(r0)
           LD      F2, 0(r2)
           MULTD   F4, F0, F2
           LD      F6, 0(r4)
           ADDD    F8, F4, F6
           SD      0(r6), F8
           ADDI    r0, r0, #4
           ADDI    r2, r2, #4
           ADDI    r4, r4, #4
           ADDI    r6, r6, #4
           SUBI    r8, r8, #1
           BNEZ   r8, Loop

```

3) Tomasulo's Algorithm

Consider the following specifications.

(40 points)

FU type	cycles in EX	Number of FUs
Integer	1	1
FP adder	5	1
FP multiplier	8	1
FP Divide	24	1

Assume the following.

- Functional units are not pipelined.
- All stages except EX take one cycle to complete.
- No limit on reservation stations.
- There is no forwarding between functional units. Both integer and floating point results are communicated through the CDB.
- Memory accesses use the integer functional unit to perform effective address calculation.
- All loads and stores will access memory during the EX stage. Pipeline stage EX does both the effective address calculation and memory access for loads/stores.
- There are unlimited load/store buffers and an infinite instruction queue.

- Loads and stores take one cycle to execute. Loads and stores share a memory access unit.
- If an instruction is in the WR stage in cycle x , then an instruction that is waiting on the same functional unit (due to a structural hazard) can begin execution in cycle x , unless it needs to read the CDB, in which case it can only start executing on cycle $x + 1$.
- Only one instruction can write to the CDB in a clock cycle. Branches and stores do not need the CDB since they don't have WR stage.
- Whenever there is a conflict for a functional unit or the CDB, assume program order.
- When an instruction is done executing in its functional unit and is waiting for the CDB, it is still occupying the functional unit and its reservation station. (meaning no other instruction may enter).
- Treat the BNEZ instruction as an Integer instruction. Assume LD instruction after the BNEZ can be issued the cycle after BNEZ instruction is issued due to branch prediction

Fill in the execution profile for the code given in the table which includes the cycles that each instruction occupies in the IS, EX, and WR stages and comments to justify your answer such as type of hazards and the registers involved.

Instruction	IS	EX	WR	Comments
LD F0, 0(r0)	1	2	3	
ADDD F2, F0, F4	2			
MULTD F4, F2, F6				
ADDD F6, F8, F10				
DADDI r0,r0, #8				
LD F1, 0(r1)				
MULTD F1, F1, F8				
ADDD F6, F3, F5				
DADDI r1, r1, #8				