Evaluating A Pipeline Design

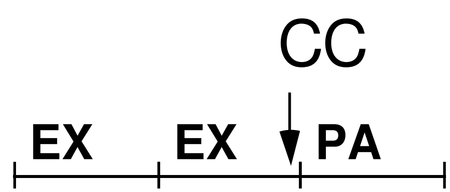
A pipeline will stall when a there are dependencies among the instructions, but how much will be the delay?

The major forms of dependence are:

- Branch or procedural dependency.
- Data conflict due to the unavailability of an operand.
- Run-on instructions force a completion ordering.
- Resource conflicts occur when multiple instructions need a unit simultaneously.

Processor Assumptions

1) At most one instruction is decoded at a time.
2) No out-of-order decode or putaway.
3) "Time is lost forever."
4) Only resources to execute one instruction sequence.
5) Functional instructions set the CC after EX cycle(s).



Evaluation Rules!

- Form timing template.
- Create the relative timing templates of two successive instructions, * and *+1.
- For each delay type (branch, data, run-on) figure the delay in the second instruction, where delay is the clock cycles that execution is deferred.
- The weighted occurrence of this delay is added to the minimum execution rate to determine the delay for that instruction type.
- Summarize the effects for all delay classes.

Effects Of Delays On Pipelines

Branches

- BC (in-line) -- D awaits the cycle following CC set.
- BC (target) -- D awaits the cycle following CC set and fetch of target instruction.
- BR -- D awaits the fetch of the target instruction.

Data

- Address dependence -- AG awaits completion of EX of prior instruction.
- Execution dependence -- EX awaits completion of prior instruction (PA)

Run-on -- the following instruction's EX does not begin until after the PA so they complete in order.

Resource -- only one instruction at a time can use a give resource

Case Study

Assume a variant of the Amdahl V/8

Multiphase operations -- seperate the use of a resource into phases, with at most one instruction in a stage

- 3 phase cache -- access directory, access array, X-mit
- Cache reference is a 3-cycle operation.

Cache Access:	3 cycles
Execution	2 cycles
Address Generate	1 cycle
Putaway	0 (combined with EX)

Ideal Case

*	IF	IF	IF	D	AG	DF	DF	DF	EX	EX	
*+1		IF	IF	IF	D	AG	DF	DF	DF	EX	EX

Features:

10 cycles, mostly determined by memory speed

2 memory references per cycle

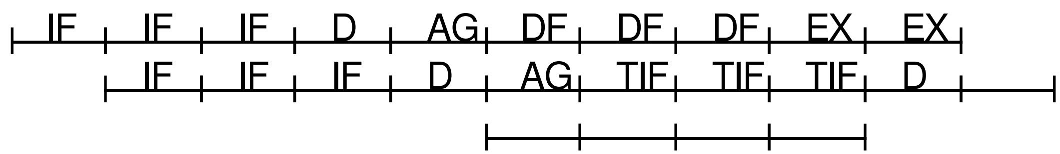
L/S architecture will have fewer memory references per cycle, on average.

One Issue Per Cycle

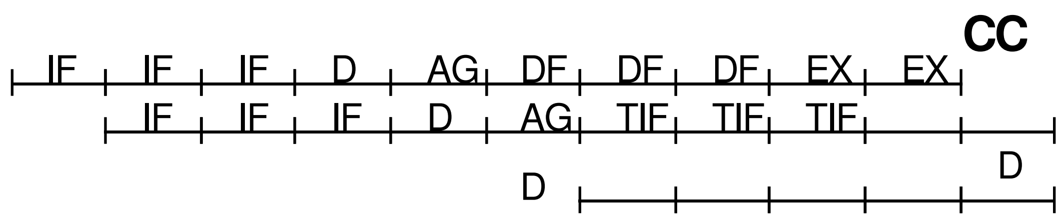
IF	IF	IF	D	AG	DF	DF	DF	EX	EX									
	IF	IF	IF	D	AG	DF	DF	DF	EX	EX								
		IF	IF	IF	D	AG	DF	DF	DF	EX	EX							
			IF	IF	IF	D	AG	DF	DF	DF	EX	EX						
				IF	IF	IF	D	AG	DF	DF	DF	EX	EX					
					IF	IF	IF	D	AG	DF	DF	DF	EX	EX				
						IF	IF	IF	D	AG	DF	DF	DF	EX	EX			
							IF	IF	IF	D	AG	DF	DF	DF	EX	EX		
								IF	IF	IF	D	AG	DF	DF	DF	EX	EX	
									IF	IF	IF	D	AG	DF	DF	DF	EX	EX

Branch Dependencies

Unconditional branch 4-cycle penalty



Conditional branch 5-cycle penalty



Note that the target instruction fetch comes at the point in the pipeline when the data fetch would normally have occurred, so the memory is available.

Data Dependencies

Address dependencies 5-cycle penalty

IF	IF	IF	D	AG	DF	DF	DF	EX	EX	
	IF	IF	IF	D						AG

Store dependency 6-cycle penalty

IF	IF	IF	D	AG	DF	DF	DF	EX	EX						
	IF	IF	IF	D	AG					DS	DS	DS			
		IF	IF	IF	D	AG							DF	DF	DF

Optimize by removing value from register

IF	IF	IF	D	AG	DF	DF	DF	EX	EX				
	IF	IF	IF	D	AG					R→SR	DS	DS	DS
		IF	IF	IF	D	AG					DF	DF	DF

Run-on Instructions

IF	IF	IF	D	AG	DF	DF	DF	EX	EX	EX	EX	EX	EX		
	IF	IF	IF	D	AG	DF	DF	DF						EX	EX

The penalty lasts for as many cycles as are required to free the execution unit that is being monopolized by the run-on insruction.