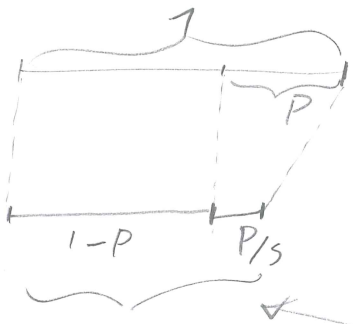
(3)

1.)

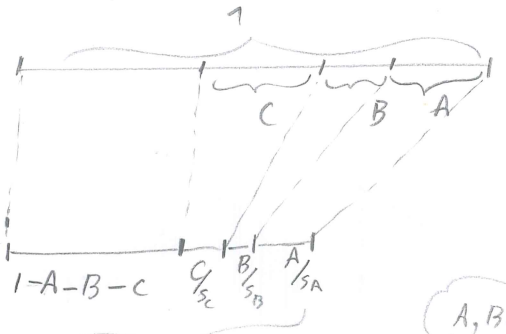


P in %

Normal formula:

$$\text{speedup} = \frac{1}{1-P+P/S}$$

multiple enhancements:



$$\text{speedup} = \frac{1}{1-A-B-C+\frac{A}{S_A}+\frac{B}{S_B}+\frac{C}{S_C}}$$

A, B, C in %

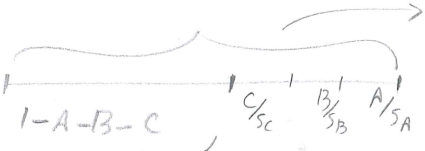
2.)

$$\text{speedup} = 10 = \frac{1}{1-.25-.25-C+\frac{.25}{30}+\frac{.25}{20}+\frac{C}{15}}$$

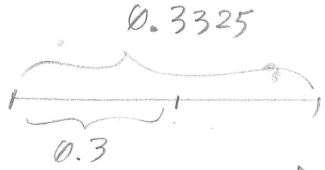
$$0.5-C+\frac{.25}{30}+\frac{.25}{20}+\frac{C}{15}=0.1 \quad \Rightarrow \quad .56C = 25.25 \qquad C = 0.45$$

C = 45%

3.)



→ in this case



$$\Rightarrow \quad \frac{1-A-B-C}{1-A-B-C+\frac{A}{S_A}+\frac{B}{S_B}+\frac{C}{S_C}} = \frac{1-.25-.35-.70}{1-.25-.35-.7+\frac{.25}{30}+\frac{.35}{20}+\frac{.70}{15}} = \frac{0.3}{0.3325}$$

= 90 %

4.) $Speedup_A = \frac{1}{1-.15+\frac{.15}{30}} = 1.17$

$Speedup_B = \frac{1}{1-.15+\frac{.15}{20}} = 1.17$

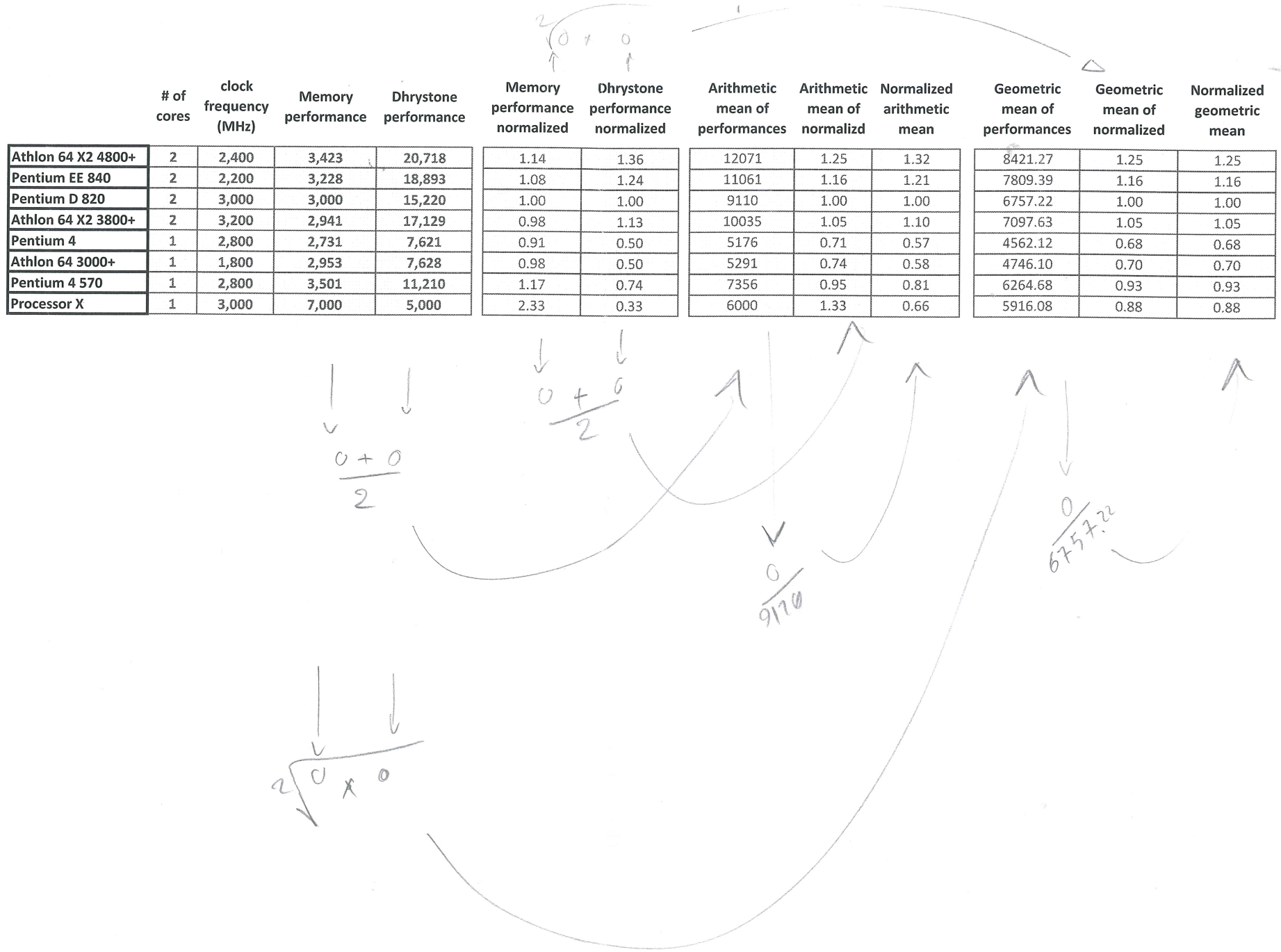
$Speedup_C = \frac{1}{1-0.7+\frac{0.7}{15}} = 2.88$ ✓

$Speedup_{AB} = \frac{1}{1-.15-.15+\frac{.15}{30}+\frac{.15}{20}} = 1.4$

$Speedup_{AC} = \frac{1}{1-.15-.7+\frac{.15}{30}+\frac{.7}{15}} = 4.96$ ✓

$Speedup_{BC} = \frac{1}{1-.15-.7+\frac{.15}{20}+\frac{.7}{15}} = 4.90$

	# of cores	clock frequency (MHz)	Memory performance	Dhrystone performance	Memory performance normalized	Dhrystone performance normalized	Arithmetic mean of performances	Arithmetic mean of normalizd	Normalized arithmetic mean	Geometric mean of performances	Geometric mean of normalized	Normalized geometric mean
Athlon 64 X2 4800+	2	2,400	3,423	20,718	1.14	1.36	12071	1.25	1.32	8421.27	1.25	1.25
Pentium EE 840	2	2,200	3,228	18,893	1.08	1.24	11061	1.16	1.21	7809.39	1.16	1.16
Pentium D 820	2	3,000	3,000	15,220	1.00	1.00	9110	1.00	1.00	6757.22	1.00	1.00
Athlon 64 X2 3800+	2	3,200	2,941	17,129	0.98	1.13	10035	1.05	1.10	7097.63	1.05	1.05
Pentium 4	1	2,800	2,731	7,621	0.91	0.50	5176	0.71	0.57	4562.12	0.68	0.68
Athlon 64 3000+	1	1,800	2,953	7,628	0.98	0.50	5291	0.74	0.58	4746.10	0.70	0.70
Pentium 4 570	1	2,800	3,501	11,210	1.17	0.74	7356	0.95	0.81	6264.68	0.93	0.93
Processor X	1	3,000	7,000	5,000	2.33	0.33	6000	1.33	0.66	5916.08	0.88	0.88



(4)

	1	2	3	4	5	6	7	8	9	10	11
1) ADD R0,R0,4	F	D	E1	E2	W						
2) LD R1, 0(R0)		F	D	E	M	W					
3) Mul R2,R1,R1			F	D	E1	E2	E3	E4	W		
4) ADD R2,R1,R2				F	D	E1	E2	W			
5) ADD R2,R2,R3					F	D	E1	E2	W		
6) SD R2,0(R0)						F	D	E	M	W	

problems:

② reads R0 before it's written by ①

③ reads R1 before it's written by ②

④ reads R2 before it's written by ③

④ writes R2 before it's written by ③

⑤ reads R2 before it's written by ④ or ③

⑤ & ③ write to R2 at the same time! (Same W stage)

⑥ reads R2 before written by ⑤

(6)

a)

RAW: 1,3 R1
2,3 R2
1,4 R1
2,4 R2
3,5 R3
4,5 R4
3,6 R3
4,6 R4
5

WAR: 1,3 R3
2,3 R3

WAR mem: 1,6

WAW: 4,5 R4

we'll see which leads to hazards in b)

b)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
(1)	F	D	E	M	W											
(2)		F	D	E	M	W										
(3)			F	D	//	//	E	M	W							
(4)				F	//	//	D	E	M	W						
(5)							F	D	//	//	E	M	W			
(6)								F	//	//	D	//	//	E	M	W

After 15 cycles all are finished (or 16, but 16 is WB for Hone which does nothing)

c) add forwarding from E & M stages to the D stage.

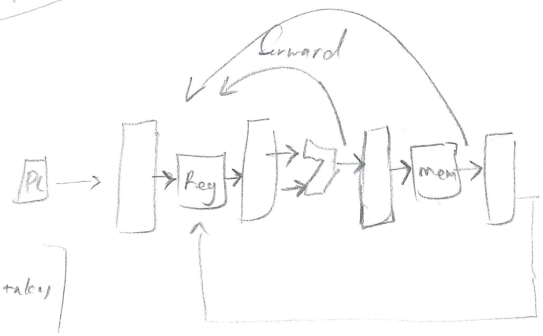
(for LD instructions, we need to stall as before, but only for 1 cycle)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
(1)	F	D	E	M	W											
(2)		F	D	E	M	W										
(3)			F	D	//	E	M	W								
(4)				F	//	D	E	M	W							
(5)						F	D	E	M	W						
(6)							F	D	E	M	W					

Finishes after 10 cycles

potentially can forward from M to E but then it takes too long, critical path will be through Mem & ALU

if done, then it takes 9 cycles



Iteration number	Instructions		Issues at clock cycle number	Executes at clock cycle number	Memory access at clock cycle number	Write CDB at clock cycle number	Comment
1	LD	R2,0(R1)	1	2	3	4	First issue
1	DADDIU	R2,R2,#1	1	5		6	Wait for LW
1	SD	R2,0(R1)	2	3	7		Wait for DADDIU
1	DADDIU	R1,R1,#8	2	3		4	Execute directly
1	BNE	R2,R3,LOOP	3	7			Wait for DADDIU
2	LD	R2,0(R1)	4	8	9	10	Wait for BNE
2	DADDIU	R2,R2,#1	4	11		12	Wait for LW
2	SD	R2,0(R1)	5	9	13		Wait for DADDIU
2	DADDIU	R1,R1,#8	5	8		9	Wait for BNE
2	BNE	R2,R3,LOOP	6	13			Wait for DADDIU
3	LD	R2,0(R1)	7	14	15	16	Wait for BNE
3	DADDIU	R2,R2,#1	7	17		18	Wait for LW
3	SD	R2,0(R1)	8	15	19		Wait for DADDIU
3	DADDIU	R1,R1,#8	8	14		15	Wait for BNE
3	BNE	R2,R3,LOOP	9	19			Wait for DADDIU

Figure 2.20 The time of issue, execution, and writing result for a dual-issue version of our pipeline *without* speculation. Note that the LD following the BNE cannot start execution earlier because it must wait until the branch outcome is determined. This type of program, with data-dependent branches that cannot be resolved earlier, shows the strength of speculation. Separate functional units for address calculation, ALU operations, and branch-condition evaluation allow multiple instructions to execute in the same cycle. Figure 2.21 shows this example with speculation,

the speculative processor executes in clock cycle 13, while it executes in clock cycle 19 on the nonspeculative pipeline. Because the completion rate on the nonspeculative pipeline is falling behind the issue rate rapidly, the nonspeculative pipeline will stall when a few more iterations are issued. The performance of the nonspeculative processor could be improved by allowing load instructions to complete effective address calculation before a branch is decided, but unless speculative memory accesses are allowed, this improvement will gain only 1 clock per iteration.

This example clearly shows how speculation can be advantageous when there are data-dependent branches, which otherwise would limit performance. This advantage depends, however, on accurate branch prediction. Incorrect speculation will not improve performance, but will, in fact, typically harm performance.

Iteration number	Instructions		Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD	R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU	R2,R2,#1	1	5		6	7	Wait for LW
1	SD	R2,0(R1)	2	3			7	Wait for DADDIU
1	DADDIU	R1,R1,#8	2	3		4	8	Commit in order
1	BNE	R2,R3,LOOP	3	7			8	Wait for DADDIU
2	LD	R2,0(R1)	4	5	6	7	9	No execute delay
2	DADDIU	R2,R2,#1	4	8		9	10	Wait for LW
2	SD	R2,0(R1)	5	6			10	Wait for DADDIU
2	DADDIU	R1,R1,#8	5	6		7	11	Commit in order
2	BNE	R2,R3,LOOP	6	10			11	Wait for DADDIU
3	LD	R2,0(R1)	7	8	9	10	12	Earliest possible
3	DADDIU	R2,R2,#1	7	11		12	13	Wait for LW
3	SD	R2,0(R1)	8	9			13	Wait for DADDIU
3	DADDIU	R1,R1,#8	8	9		10	14	Executes earlier
3	BNE	R2,R3,LOOP	9	13			14	Wait for DADDIU

Figure 2.21 The time of issue, execution, and writing result for a dual-issue version of our pipeline *with* speculation. Note that the LD following the BNE can start execution early because it is speculative.