When an instruction cannot be fetched be cause a load or store instruction is using the memory in that cycle. Cycles are represented from left to right, a for each instruction we show the pipeline stage it is in turning that cycle:

2424244444444444444

Inspection	Pipeline Stage	Lydes
Sw 116, 12 (16)	IF ID EX MEM WB	11
Iw r16, 8(rg)	IF ED EX MEM WB	
beg rsiry, label HASSure rsirry	IF ID EX MEM WB	OI CHA
add rsiring	THE TO EX MEM WE	16
sit rs, ris, ry	1 150 de 1931d a doub marist ID EX MEM	WB

We can not add NOPs to code to eliminate this hazard as NOPs need to be fetched just like any other instructions, so this hazard must be addressed with a hadware hazard detection unit in the processor.

I be discour of while los with a starting of the court of the

4.10-2 This change only somes one upde in an entire execution without data hazards (such as the one given). This cycle is saved because the last instruction finishes one cycle earlier (one less stage to go through). If there were data hazards from loads to other instruction the change would help eliminate some stall cycles.

Instructions	Cycles with 5	Cycles with 4	Speed up
Executed	Stages	Stages	The Landson
5	4+8=9	3+5=8	9/8 = 1.13

4.11.1	lw 11,0(1)	WB	1 01	: 1	-6-14	مادق	LELE	_ 1.00	144	
-	lw ri, olry	1	WFW	MR	ANDAA	0	alle		ALL STORY	and the same
	beg n, rollopp									the desired
idonaly		IF	+**							the man
	and riverz			IF						Le ignile
	(m 11,0(n)				IF	***	ID	E-X	WEW	
Ly	1w 1,0(1)					بسيالته	IF	ID	***	المراميا الم
	beg riro, loop			)		17 1	1 1	IF	***	6 118 111

In a particular cycle, dock cycle, a pipeline stage is not doing useful work if it is stalled or if the instruction going though that stage is not boing any useful work there. In the pipeline execution fingram from above, a stage is stalled if its name is not shown for a particular cycles, and stages in which the particular instruction is not doing useful work are marked in blue. Note that a beg instruction is doing useful work in the MEM stage, because it is determining the correct value of the

next instruction's PC in that stage.

Cycles per loop lycles in which all stages % of cycles in which all usety work Stage do usetu

tractions of cycles stalled when no towarding is used.

For the first instruction there will be two stall gives the to the dependencies of first and second next instruction. Therefore dependency will be one stall cycle with second next instruction.

I + (Ex to 1st enty + Ex to 2nd only + menony +0 1st) \* 2 +

(Ex to 2nd only) \* 1 (Menory to 2nd) \* 1 + other RAW dependences \* 1)

I + (0.35 \* 2) + (0.05 \* 1) + (0.1 \* 1) = (0.1 \* 1) = 1.96

Therefore, fraction of cycles stalled is 0.95/1.95 = 48.7 %.

U.122 tractions of cycles stalled when full torwarding is used.

With full forwarding only Read & write dependencies cause cycles to be stalled. There will be only one cycle stalled with memory stage of one instruction to next instruction.

Memory to 1st + memory to 2nd + other RAW dependencies

I + 0.2 + 0.1 + 0.1

= 3.4

Therefore, fraction of cycle stalled is 0.4/1.4 = 28.5%.

4.12.4 To calculate speed up:

Taking the computed results with no formating &

full forwarding:

Speed up with formating = 1.25 × 130ps = 253.5 ps

Speed up with formating = 1.20 × 180 ps = 180 ps

Takan speed up = 1+ (253.5-180)/180) = 1.4 ps

2 Ad 75 72.51	-			,	VI	11		1	ETGLES
					2	10	A	0	beg
nop			VI	Mc	1	4		1	
110 r3, 4(rs)			U	_/M		16	U	-	1100
10 12,0(12)				and the spills of	2	- Industrial		-	+-+
Marian San San San San San San San San San S		-	TI	11			-		+
or r3, rs, r3	100		-			-	+	-	
	1/4	4					-	-	
nop	1	1		-	-	_	-	_	
sio r3, o (rs)	000	1							
19 10	11	1							
MILES ME ME	DI								
	add rs, r2, r1 nop nop 1 10 r3, 4(rs) 1 10 r2, 0(r) nop 0 r3, rs, r3 nop nop sio r3, 0 (rs)	add rs, r2, r1  nop  nop  1 to r3, 4(rs)  1 to r2, 0(rs)  nop  or r3, rs, r3  nop  nop  sio, r3, o(rs)	a dd rs, r2, r1  nop  1ω r3, 4(rs)  1ω r2, 0(r2)  nop  or r3, rs, r3  nop  nop  sω r3, o(rs)	a dd rs, r2, r1  nop  nop  1ω r3, 4(rs)  ω r2, 0(r)  nop  or r3, rs, r3  nop  nop  sω r3, o (rs)	a dd rs, r2, r1  nop  nop  1ω r3, 4(rs)  ω r2, 0(r2)  nop  or r3, rs, r3  nop  nop  sω r3, 0 (rs)	add rs, r2, r1 nop nop 1 w r3, 4(rs) 1 w r2, 0(r2) nop or r3, rs, r3 nop nop sw r3, o(rs)	α δδ τς, τ 2, τ 1  nop  nop  1ω τ3, 4 (τς)  ω τ2, 0 (τ2)  nop  ον τ3, τς, τ3  nop  nop  νω τ3, ο (τς)	α δδ τς, τ 2, τ 1  nop  nop  1ω τ3, 4 (τς)  ω τ2, 0 (τ2)  nop  or τ3, τς, τ3  nop  nop  sw τ3, 0 (τς)	α dò rs, r 2, r 1  nop  nop  lω r3, y (rs)  ιω r2, 0 (r2)  nop  or r3, rs, r3  nop  nop  sω r3, o (rs)

4.132 Instruction can be moved up by swapping its place with another instruction, which has no teperatorcies such that some NOP slots can be filled with such instructions. Int he grene example, we have may make use of Rt to eliminate dependencies of WAW I WAR so that we can have more instructions to be moved up.

11: add r5, r2, r1

i3: 100 r2, o(r2) moved up to fill nop slot

nop

12; 10 +3, 4 LTS)

had to add on the nop hee, so thee

nop

is no performance gain.

14: or 13,15,13

is: 500, r3, 0(rs)

	- F E			1 1 1	0		1111111111111			EX	DEL			- June	- Li	
4. 14.1	Instructions	1	12	13	4	5	6	7	8	9	10	11	12	13	14	
	[w 12,0(n)	I	I	E	Me						_	1,2	LEE	2		11
		F	D	X	m	В		1						24		
	label 1: beg		I	I,	***	Ex	Me	W						10		
	r2, ro, label		F	0	*		m	B	-			للت		_11	_	
	2		6							-		110	غنب	2		-
	[w r3,						IF	ID	Ex	Me		none was their		. 26		
	0 (12)									~	В	1.2	1.23	M		- Green
	beg r3,		-	-				IF		*	6x	Me	W	فعد		
	ro, label					_	-	-	0	F		m	B	وكيف		
	laber 1: beg					L			I	*	ID	Ex		W		
	rz, ro, label							1	IF	*			m	B.	4	-
	2 (brach taken				-									-		
		-	-1		-		-	-	-			1-0	-	100	1	_
	500 11,								-		IF	120	Gx	me		
	0(2)								-					/"	6	

14.2	Instucions	1	2	3	14	5	6	17	18	9	],	,  ,	11.	114
-	(10 r2,0(n)	I	I	Ex	me	W	1		101		101	+	2 18	19
-		F	D	P.	m	B				7		1		
-	label1: beq r2,	-	I	I	*	Ex	Me	W		+	_	+	+	+-
-4	ro, label 2	1	F	0	*		m	B		1		1		
	lw r3,0(2)	-	-	IF	**1	ID	Ex	Me	WB					+
	beg r3, ro,						I	01	G <sub>X</sub>	Me	W B		1	
-	499 x11x81					-	-	IF	IO	Ex	Me	W		+
	label: begrz,	4		9					IF	IO	Бх	Me	W B	
	[ brance taken) 1 N r3, 0[r2] Sw r1,0[r2]									IF		-	Mem	WB -
	110112			1		1	1	1	1		JIF	ID	EX	me u

4.181

The first three execution cycles are IF, ID, and EX.
The branch outcomes are determined in the EX stage.

Number of stall cycles due to mis-presided branches: 3

Breakfown of dynamic instructions into BEQ = 25 % = 0.25

Accuracy of the presided branches with Always-Taken presider = 45% = 0.45
Accuracy of the mil-presided branches with Always-taken presider =

1.00-0.45 = 0.55

Threfore, Extra CPI the to mis-presides branches with Always-Taken presider= 3 + 0.55 + 0.25

= 0.4125