

CS147, Sec 01
Homework II
Craig Huff

1. Table 1: Control Signal Table for I and II

Instruction: add r3, r2, r1						Instruction: slr r4, r3, 0x3a					
CTRL	IF	ID/RF	EXE	MEM	WB	CTRL	IF	ID/RF	EXE	MEM	WB
pc_load	0	0	0	0	1	pc_load	0	0	0	0	1
pc_sel_1					1	pc_sel_1					1
pc_sel_2					0	pc_sel_2					0
pc_sel_3					1	pc_sel_3					1
mem_r	1	1	1	1	1	mem_r	1	1	1	1	1
mem_w	0	0	0	0	0	mem_w	0	0	0	0	0
r1_sel_1		0	0	0	0	r1_sel_1		0	0	0	0
reg_r	1	1	1	1	0	reg_r	1	1	1	1	0
reg_w	0	0	0	0	1	reg_w	0	0	0	0	1
wa_sel_1					0	wa_sel_1					0
wa_sel_2						wa_sel_2					
wa_sel_3					1	wa_sel_3					1
wd_sel_1					0	wd_sel_1					0
wd_sel_2					0	wd_sel_2					0
wd_sel_3					1	wd_sel_3					1
sp_load	0	0	0	0	0	sp_load	0	0	0	0	0
op1_sel_1			0	0	0	op1_sel_1			0	0	0
op2_sel_1						op2_sel_1			1	1	1
op2_sel_2						op2_sel_2					
op2_sel_3						op2_sel_3			1	1	1
op2_sel_4			1	1	1	op2_sel_4			0	0	0
alu_oprn			add	add	add	alu_oprn			srl	sol	srl
ma_sel_1						ma_sel_1					
dmem_r	1	1	1	1	1	dmem_r	1	1	1	1	1
dmem_2	0	0	0	0	0	dmem_2	0	0	0	0	0
md_sel_1						md_sel_1					

Table 2: Control Signal Table for III and IV

Instruction: jr r20						Instruction: muli r3, r4, 0 x a5a5					
CTRL	IF	ID/RF	EXE	MEM	WB	CTRL	IF	ID/RF	EXE	MEM	WB
pc_load	0	0	0	0	1	pc_load	0	0	0	0	1
pc_sel_1					0	pc_sel_1					1
pc_sel_2					0	pc_sel_2					0
pc_sel_3					1	pc_sel_3					1
mem_r	1	1	1	1	1	mem_r	1	1	1	1	1
mem_w	0	0	0	0	0	mem_w	0	0	0	0	0
r1_sel_1		0	0	0	0	r1_sel_1					0
reg_r	1	1	1	1	1	reg_r	1	1	1	1	0
reg_w	0	0	0	0	0	reg_w	0	0	0	0	1
wa_sel_1						wa_sel_1					1
wa_sel_2						wa_sel_2					
wa_sel_3						wa_sel_3					1
wd_sel_1						wd_sel_1					0
wd_sel_2						wd_sel_2					0
wd_sel_3						wd_sel_3					1
sp_load	0	0	0	0	0	sp_load	0	0	0	0	0
op1_sel_1						op1_sel_1					
op2_sel_1						op2_sel_1					
op2_sel_2						op2_sel_2			1	1	1
op2_sel_3						op2_sel_3			0	0	0
op2_sel_4						op2_sel_4			0	0	0
alu_oprn						alu_oprn			mul	mul	mul
ma_sel_1						ma_sel_1					
dmem_r	1	1	1	1	1	dmem_r	1	1	1	1	1
dmem_2	0	0	0	0	0	dmem_2	0	0	0	0	0
md_sel_1						md_sel_1					

Table 3: Control Signal Table for V and VI

Instruction: andi r3, r4, 0xa5a5						Instruction: lui r5, 0xa5a5					
CTRL	IF	ID/RF	EXE	MEM	WB	CTRL	IF	ID/RF	EXE	MEM	WB
pc_load	0	0	0	0	1	pc_load	0	0	0	0	1
pc_sel_1					1	pc_sel_1					1
pc_sel_2					0	pc_sel_2					0
pc_sel_3					1	pc_sel_3					1
mem_r	1	1	1	1	1	mem_r	1	1	1	1	1
mem_w	0	0	0	0	0	mem_w	0	0	0	0	0
r1_sel_1		0	0	0	0	r1_sel_1					
reg_r	1	1	1	1	0	reg_r	1	1	1	1	0
reg_w	0	0	0	0	1	reg_w	0	0	0	0	1
wa_sel_1					1	wa_sel_1					1
wa_sel_2						wa_sel_2					
wa_sel_3					1	wa_sel_3					1
wd_sel_1					0	wd_sel_1					
wd_sel_2					0	wd_sel_2					1
wd_sel_3					1	wd_sel_3					1
sp_load	0	0	0	0	0	sp_load	0	0	0	0	0
op1_sel_1			0	0	0	op1_sel_1					
op2_sel_1						op2_sel_1					
op2_sel_2			0	0	0	op2_sel_2					
op2_sel_3			0	0	0	op2_sel_3					
op2_sel_4			0	0	0	op2_sel_4					
alu_oprn			and	and	and	alu_oprn					
ma_sel_1						ma_sel_1					
dmem_r	1	1	1	1	1	dmem_r	1	1	1	1	1
dmem_2	0	0	0	0	0	dmem_2	0	0	0	0	0
md_sel_1						md_sel_1					

Table 4: Control Signal Table for VII and VIII

Instruction: bneq r2, r3, 0xa5a5; // r2 = 0x1; r3 = 0x1						Instruction: lw r24, r6, 0x5a5a					
CTRL	IF	ID/RF	EXE	MEM	WB	CTRL	IF	ID/RF	EXE	MEM	WB
pc_load	0	0	0	0	1	pc_load	0	0	0	0	1
pc_sel_1					1	pc_sel_1					1
pc_sel_2					0	pc_sel_2					0
pc_sel_3					1	pc_sel_3					1
mem_r	1	1	1	1	1	mem_r	1	1	1	1	1
mem_w	0	0	0	0	0	mem_w	0	0	0	0	0
r1_sel_1		0	0	0	0	r1_sel_1		0	0	0	0
reg_r	1	1	1	1	1	reg_r	1	1	1	1	0
reg_w	0	0	0	0	0	reg_w	0	0	0	0	1
wa_sel_1						wa_sel_1					1
wa_sel_2						wa_sel_2					
wa_sel_3						wa_sel_3					1
wd_sel_1						wd_sel_1					1
wd_sel_2						wd_sel_2					0
wd_sel_3						wd_sel_3					1
sp_load	0	0	0	0	0	sp_load	0	0	0	0	0
op1_sel_1			0	0	0	op1_sel_1			0	0	0
op2_sel_1						op2_sel_1					
op2_sel_2						op2_sel_2			1	1	1
op2_sel_3						op2_sel_3			0	0	0
op2_sel_4			1	1	1	op2_sel_4			0	0	0
alu_oprn			sub	sub	sub	alu_oprn			add	add	add
ma_sel_1						ma_sel_1				0	
dmem_r	1	1	1	1	1	dmem_r	1	1	1	1	1
dmem_2	0	0	0	0	0	dmem_2	0	0	0	0	0
md_sel_1						md_sel_1				0	

Table 5: Control Signal Table for IX and X

Instruction: jal 0x2a55aa5						Instruction: push					
CTRL	IF	ID/RF	EXE	MEM	WB	CTRL	IF	ID/RF	EXE	MEM	WB
pc_load	0	0	0	0	1	pc_load	0	0	0	0	1
pc_sel_1						pc_sel_1					1
pc_sel_2						pc_sel_2					0
pc_sel_3					0	pc_sel_3					1
mem_r	1	1	1	1	1	mem_r	1	1	1	1	1
mem_w	0	0	0	0	0	mem_w	0	0	0	0	0
r1_sel_1						r1_sel_1		1	1	1	1
reg_r	1	1	1	1	0	reg_r	1	1	1	1	1
reg_w	0	0	0	0	1	reg_w	0	0	0	0	0
wa_sel_1						wa_sel_1					
wa_sel_2					1	wa_sel_2					
wa_sel_3					0	wa_sel_3					
wd_sel_1						wd_sel_1					
wd_sel_2						wd_sel_2					
wd_sel_3					0	wd_sel_3					
sp_load	0	0	0	0	0	sp_load	0	0	1	1	1
op1_sel_1						op1_sel_1			1	1	1
op2_sel_1						op2_sel_1			0	0	0
op2_sel_2						op2_sel_2					
op2_sel_3						op2_sel_3			1	1	1
op2_sel_4						op2_sel_4			0	0	0
alu_oprn						alu_oprn			sub	sub	sub
ma_sel_1						ma_sel_1				1	
dmem_r	1	1	1	1	1	dmem_r	1	1	1	1	0
dmem_2	0	0	0	0	0	dmem_2	0	0	0	0	1
md_sel_1						md_sel_1				1	

2. (a) *Performance comparison*

Performance of P1	Performance of P2
Average CPI = $17 / 5 = 2.6$	Average CPI = $14 / 3 = 4.6667$
Instructions per Second = 1,923,076,923.077	Instructions per Second = 1,071,428,571.429

(b) *Processor Speed Comparison (let n be the number of instructions)*

$$N_{p1} = 3n + 4n + 3n + 4n + 3n = 17n \quad N_{p2} = 9n + 2n + 1n + 4n + 5n = 21n$$

$$E_{p1} = \frac{17n}{5} = 3.4n \quad E_{p2} = \frac{21n}{3} = 7n$$

$$\frac{E_{p1}}{E_{p2}} = \frac{7n}{3.4n} = 2.05 \quad \textbf{P1 is 2.05 times faster than P2}$$

3. The program P will have an execution time (E) of 30 seconds. By using the equation for I (Number of Instructions) \times CPI \div Frequency.

$$30000000000 = \frac{I \times 8}{2.0}$$

$$I = \frac{30000000000 \times 2}{8.0}$$

$$I = 7,500,000,000 \text{ multiplication instructions}$$

When the program is recompiled and run again, the program P will have an execution time (E') of 20 seconds.

$$20000000000 = \frac{(I_1 \times 8)}{2.0}$$

$$(I_1 \times 8) = 40000000000$$

$$I_1 = 5,000,000,000$$

We replaced 5,000,000,000 of multiplications by the compiler.
Program P' replaces 2/3 of the multiplication instructions with additions

4. The workload of the non pipelined implementation will take $10^8 \times (250 \times 10^{12}) = .025$
The workload of the pipelined implementation can be found with

$$\frac{.025}{\text{Pipelined}} = 30$$

$$\text{Pipelined} = \frac{.025}{30}$$


$$\text{Pipelined implementation} = 0.0008333333333$$


5. Forwarding Path for 5-Stage Pipeline

Instruction									
add r8, r4, r6	IF	ID	EXE	MEM	WB				
sub r7, r2, r8		IF	ID	EXE	MEM	WB			
lw r7, r7, 0 x 1000			IF	ID	EXE	MEM	WB		
buffer				buffer	buffer	buffer	buffer	buffer	
add r8, r2, r7					IF	ID	EXE	MEM	WB

6. Data Hazards in a 5-Stage Pipeline

`add r3, r2, r4`
`sub r5, r1, r3`
`lw r6, r3, 0x2000`
`add r7, r3, r6`

 Data Hazard Will be solved in forwarding

 Data Hazard Will Cause a stall

r3 is a data hazard that will be fixed with forwarding. This is because r3 hasn't been computed by the add instruction by the time that it's called by the sub instruction.

r6 is a data hazard that will result in a stall. This is because the lw instruction won't have the data until the memory phase, but the add instruction needs r6 for execution.