Mohammad Sharif
Professor Patra
CS 147 – 1

Homework 2

1. For a non-pipeline implementation of data and control path in following diagram for a processor implementing CS147DV show the control signal logic values at different phase of the processor executing the following instructions. You need to construct 10 tables similar to Table 1 (may use hexadecimal for multi-bus signal, leave blank if don't care). [5 pts]

   I. add r3, r2, r1
   II. srl r4, r3, 0x3a
   III. jr r20
   IV. muli r3, r4, 0xa5a5
   V. andi r3, r4, 0xa5a5
   VI. lui r5, 0x5a5a
   VII. bneq r2, r3, 0xa5a5; // r2 = 0x1; r3 = 0x1
   VIII. lw r24, r6, 0x5a5a
   IX. jal 0x2a55aa5
   X. push

add r3, r2, r1					
CTRL Sig Name	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
pc_load	0	0	0	0	1
pc_sel_1					1
pc_sel_2					0
pc_sel_3					1
mem_r	1	1	1	1	1
mem_w	0	0	0	0	0
r1_sel_1		0	0	0	0
reg_r	1	1	1	1	0
reg_w	0	0	0	0	1
wa_sel_1					0
wa_sel_2					
wa_sel_3					1
wd_sel_1					0
wd_sel_2					0
wd_sel_3					1
sp_load	0	0	0	0	0
op1_sel_1			0	0	0
op2_sel_1					
op2_sel_2					

op2_sel_3					
op2_sel_4			1	1	1
alu_oprn			add	add	add
ma_sel_1					
dmem_r	1	1	1	1	1
dmem_w	0	0	0	0	0
md_sel_1					

srl r4, r3, 0x3a					
CTRL Sig Name	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
pc_load	0	0	0	0	1
pc_sel_1					1
pc_sel_2					0
pc_sel_3					1
mem_r	1	1	1	1	1
mem_w	0	0	0	0	0
r1_sel_1		0	0	0	0
reg_r	1	1	1	1	0
reg_w	0	0	0	0	1
wa_sel_1					0
wa_sel_2					
wa_sel_3					1
wd_sel_1					0
wd_sel_2					0
wd_sel_3					1
sp_load	0	0	0	0	0
op1_sel_1			0	0	0
op2_sel_1			1	1	1
op2_sel_2					
op2_sel_3			1	1	1
op2_sel_4			0	0	0
alu_oprn			srl	srl	srl
ma_sel_1					
dmem_r	1	1	1	1	1
dmem_w	0	0	0	0	0
md_sel_1					

jr r20					
CTRL Sig Name	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
pc_load	0	0	0	0	1
pc_sel_1					0
pc_sel_2					0
pc_sel_3					1

mem_r	1	1	1	1	1
mem_w	0	0	0	0	0
r1_sel_1		0	0	0	0
reg_r	1	1	1	1	1
reg_w	0	0	0	0	0
wa_sel_1					
wa_sel_2					
wa_sel_3					
wd_sel_1					
wd_sel_2					
wd_sel_3					
sp_load	0	0	0	0	0
op1_sel_1					
op2_sel_1					
op2_sel_2					
op2_sel_3					
op2_sel_4					
alu_oprn					
ma_sel_1					
dmem_r	1	1	1	1	1
dmem_w	0	0	0	0	0
md_sel_1					

muli r3, r4, 0xa5a5					
CTRL Sig Name	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
pc_load	0	0	0	0	1
pc_sel_1					1
pc_sel_2					0
pc_sel_3					1
mem_r	1	1	1	1	1
mem_w	0	0	0	0	0
r1_sel_1		0	0	0	0
reg_r	1	1	1	1	0
reg_w	0	0	0	0	1
wa_sel_1					1
wa_sel_2					
wa_sel_3					1
wd_sel_1					0
wd_sel_2					0
wd_sel_3					1
sp_load	0	0	0	0	0
op1_sel_1			0	0	0
op2_sel_1					

op2_sel_2			1	1	1
op2_sel_3			0	0	0
op2_sel_4			0	0	0
alu_oprn			mul	mul	mul
ma_sel_1					
dmem_r	1	1	1	1	1
dmem_w	0	0	0	0	0
md_sel_1					

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

lui r5, 0x5a5a					
CTRL Sig Name	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
pc_load	0	0	0	0	1
pc_sel_1					1
pc_sel_2					0
pc_sel_3					1
mem_r	1	1	1	1	1
mem_w	0	0	0	0	0
r1_sel_1					
reg_r	1	1	1	1	0
reg_w	0	0	0	0	1
wa_sel_1					1
wa_sel_2					
wa_sel_3					1
wd_sel_1					
wd_sel_2					1
wd_sel_3					1
sp_load	0	0	0	0	0
op1_sel_1					
op2_sel_1					
op2_sel_2					
op2_sel_3					
op2_sel_4					
alu_oprn					
ma_sel_1					
dmem_r	1	1	1	1	1
dmem_w	0	0	0	0	0
md_sel_1					

bneq r2, r3, 0xa5a5; // r2 = 0x1; r3 = 0x1					
CTRL Sig Name	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
pc_load	0	0	0	0	1
pc_sel_1					1
pc_sel_2					0
pc_sel_3					1
mem_r	1	1	1	1	1
mem_w	0	0	0	0	0
r1_sel_1		0	0	0	0
reg_r	1	1	1	1	1
reg_w	0	0	0	0	0
wa_sel_1					

wa_sel_2					
wa_sel_3					
wd_sel_1					
wd_sel_2					
wd_sel_3					
sp_load	0	0	0	0	0
op1_sel_1			0	0	0
op2_sel_1					
op2_sel_2					
op2_sel_3					
op2_sel_4			1	1	1
alu_oprn			sub	sub	sub
ma_sel_1					
dmem_r	1	1	1	1	1
dmem_w	0	0	0	0	0
md_sel_1					

lw r24, r6, 0x5a5a					
CTRL Sig Name	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
pc_load	0	0	0	0	1
pc_sel_1					1
pc_sel_2					0
pc_sel_3					1
mem_r	1	1	1	1	1
mem_w	0	0	0	0	0
r1_sel_1		0	0	0	0
reg_r	1	1	1	1	0
reg_w	0	0	0	0	1
wa_sel_1					1
wa_sel_2					
wa_sel_3					1
wd_sel_1					1
wd_sel_2					0
wd_sel_3					1
sp_load	0	0	0	0	0
op1_sel_1			0	0	0
op2_sel_1					
op2_sel_2			1	1	1
op2_sel_3			0	0	0
op2_sel_4			0	0	0
alu_oprn			add	add	add

ma_sel_1				0	
dmem_r	1	1	1	1	1
dmem_w	0	0	0	0	0
md_sel_1				0	

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

push					
CTRL Sig Name	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
pc_load	0	0	0	0	1
pc_sel_1					1
pc_sel_2					0
pc_sel_3					1

mem_r	1	1	1	1	1
mem_w	0	0	0	0	0
r1_sel_1		1	1	1	1
reg_r	1	1	1	1	1
reg_w	0	0	0	0	0
wa_sel_1					
wa_sel_2					
wa_sel_3					
wd_sel_1					
wd_sel_2					
wd_sel_3					
sp_load	0	0	1	1	1
op1_sel_1			1	1	1
op2_sel_1			0	0	0
op2_sel_2					
op2_sel_3			1	1	1
op2_sel_4			0	0	0
alu_oprn			sub	sub	sub
ma_sel_1				1	
dmem_r	1	1	1	1	0
dmem_w	0	0	0	0	1
md_sel_1				1	

2. Regarding performance and CPI answer the following. **[1 pt]**

(a) Consider two processor P1 and P2 of the same instruction set. There are five classes of instructions in this ISA as in the following Table 2 with the corresponding CPI. P1 has clock rate of 5GHz and P2 has clock rate of 3GHz. Assuming equal distribution of each class of instructions in a benchmark program, what is the performance of P1 and P2 in terms of instructions per second?

Instructions Per Second = Clock Rate / CPI

Class	P1	P2 – Inst. Per Sec.
A	1	3
B	2	1
C	3	1
D	4	4
E	3	5
Average	2.6	2.8
Inst. Per. Sec	$5*10^9/2.6$ = **1,923,076,923.07692**	$3*10^9/2.8$ = **1,071,428,571.42857**

(b) If the A type of instruction happens in a program as thrice and type B instruction happens twice as C,D, and E type (C,D,E are equally distributed) which processor is faster and by how much?

P1 Clock Cycle = 200 ps
P2 Clock Cycle = 333 ps

P1 Avg. CPI = (3 + 4 + 3 + 4 + 3)/8 = 17/8 = 2.125
P2 Avg. CPI = (9 + 2 + 1 + 4 + 5)/58= 21/8 = 2.625

P1 Inst. Per Sec. = 200*2.125 = 425
P2 Inst. Per Sec. = 333*2.625 = 874.125

874.125/425= 2.06

P1 is 2.06 times faster than P2

3. Consider program P, which runs on a 2GHz machine M in 30 seconds. An optimization is done to P by compiler, replacing some instances of multiplying a value by 4 ($x \leftarrow x * 4$) with 2 instructions of adding x to x ($x \leftarrow x + x$; $x \leftarrow x + x$). Let's call the new optimized program as P'. The CPI of multiplication is 8 and CPI of add is 2. After recompiling the new program is now running in 20 seconds on machine M. How many multiplications were replaced by the compiler? **[1 pt]**

   Clock cycle of 500 ps

   The new program is 1.5 times faster than the first so:

   8 * 500 + 8 * 500 + 8 * 500 / 4 * 500 + 4 * 500 + 8 * 500 = 12000/8000 = 1.5

   Total instructions = (2 GHz * 30 sec)/8 CPI = 60,000,000,000 / 8 = 7,500,000,000

   2/3 of the multiplication instructions are replaced by the compiler since there's 7,500,000,000 instructions then 5,000,000,000 are replaced.

4. A computer architect needs to design the pipeline of a new micro-processor. An example workload of 10^8 instructions is used to design the pipeline. Each instruction takes 250ps to finish. How long does it take to execute this workload on a non-pipelined processor? The pipelined implementation has been done using 30 pipeline stages. Assuming a perfect pipeline (i.e. no hazard) how much speedup has been achieved compared to non-pipelined design? **[1 pt]**

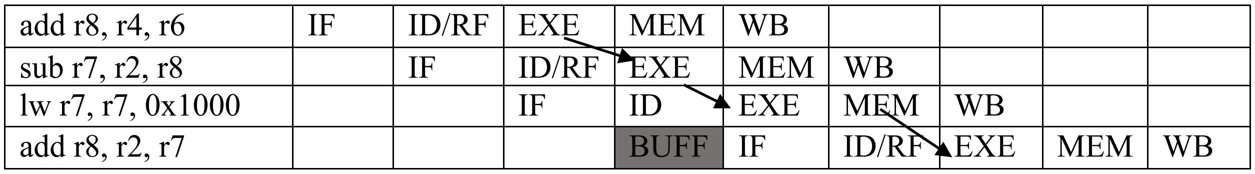
   Since there are 10^8 instructions and each of them execute in 250 ps:
   $10^8 * 250 * 10^{-12} = 250 * 10^{-4} =$ **0.025 seconds**

As we know for an n-stage pipeline $E_{\text{non pipeline}}/E_{\text{pipelined}} = n$ which means that $E_{\text{non pipeline}} / n = E_{\text{pipelined}}$:
0.025 / 30 = .000833 seconds = **8.33 * 10^{-4} seconds**

Speed up = 0.025 seconds / .000833 seconds = 30 times speedup

5. Show the forwarding path needed to execute the following four instructions in a 5-staged pipeline as discussed in the class. **[1 pt]**

```
add r8, r4, r6
sub r7, r2, r8
lw r7, r7, 0x1000
add r8, r2 r7
```



add r8, r4, r6	IF	ID/RF	EXE	MEM	WB				
sub r7, r2, r8		IF	ID/RF	EXE	MEM	WB			
lw r7, r7, 0x1000			IF	ID	EXE	MEM	WB		
add r8, r2, r7				BUFF	IF	ID/RF	EXE	MEM	WB

6. In a 5-staged pipelined processor, identify all the data dependencies in the following code. Which dependencies are the data hazards that will be resolved via forwarding? Which dependencies are data hazards that will cause a stall? **[1 pt]**

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

The following are all data hazards:
r3 because it isn't computed until the end of the execution stage for add but it needs to be used in sub.

r6 will result in a stall because lw can't have the data until after the MEM phase, but add needs it for execution.