CSE352 Spring 2013 Homework Assignment #6 solutions
Due in class Friday May 31th 2013

Read Harris & Harris (1st edition) 7.5 - Pipelined Processor

Q1) From exercise 7.24 from Harris & Harris (1st edition)

Insn	1	2	3	4	5	6	7	8	9	10	11
add $s0, $t0, $t1	F	D	X	M	W						
sub $s1, $t2, $t3		F	D	X	M	W					
and $s2, $s0, $s1			F	D	X	M	W				
or $s3, $t4, $t5				F	D	X	M	W			
slt $s4, $s2, $s3					F	D	X	M	W		

On cycle 5, register s0 is being written to and registers t4 and t5 are being read.

Note that the ALU is being forwarded register values from the W stage and M stages in cycle 5 (referred to as WX and MX forwarding).

Q2)

A) Both add instructions are causing data hazard due to read-after-write dependencies between instruction 3 and 1, 3 and 2 as well as 6 and 4 and 6 and 5.

B) Here's the pipeline diagram (d* stands for data hazard, p* stands for pipeline hazard)

Insn	1	2	3	4	5	6	7	8	9	10	11	12	13
lw $t1, 0($t0)	F	D	X	M	W								
lw $t2, 4($t0)		F	D	X	M	W							
add $t3, $t1, $t2			F	D	d*	X	M	W					
sw $t3, 12($t0)				F	p*	D	X	M	W				
lw $t4, 8($t0)					p*	F	D	X	M	W			
add $t5, $t1, $t4							F	D	d*	X	M	W	
sw $t5, 16($t0)								F	p*	D	X	M	W

The WX bypass is used for:

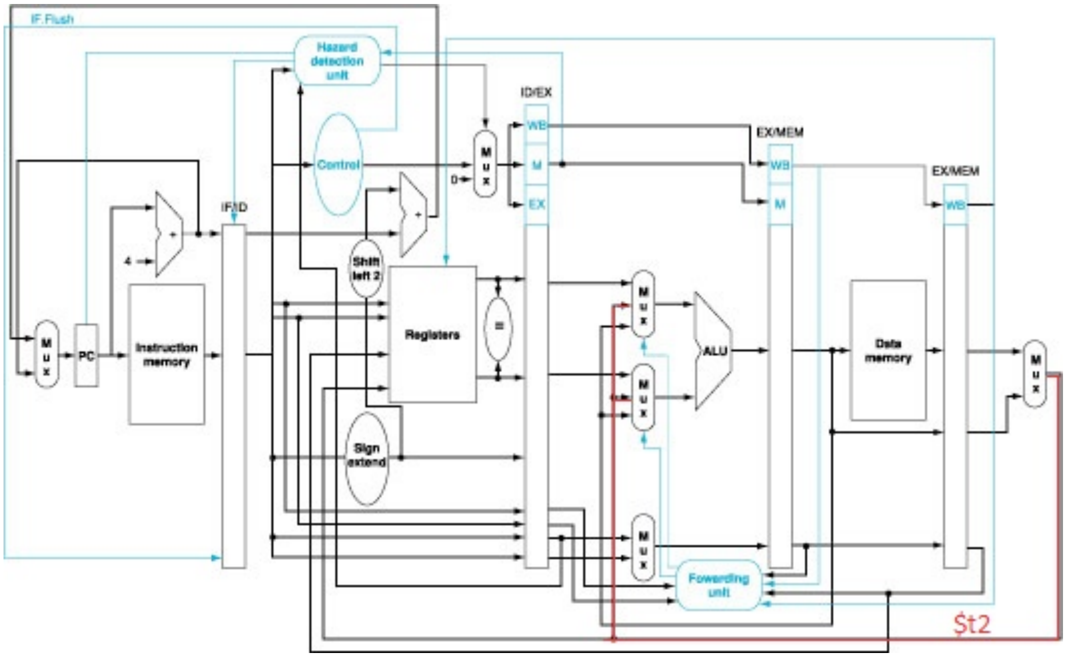
```
lw $t2, 4($t0) # Word size is 4 bytes in MIPS

add $t3, $t1, $t2
```

and

```
lw $t4, 8($t0)

add $t5, $t1, $t4
```



The WM bypass (we have to add it) is used for:

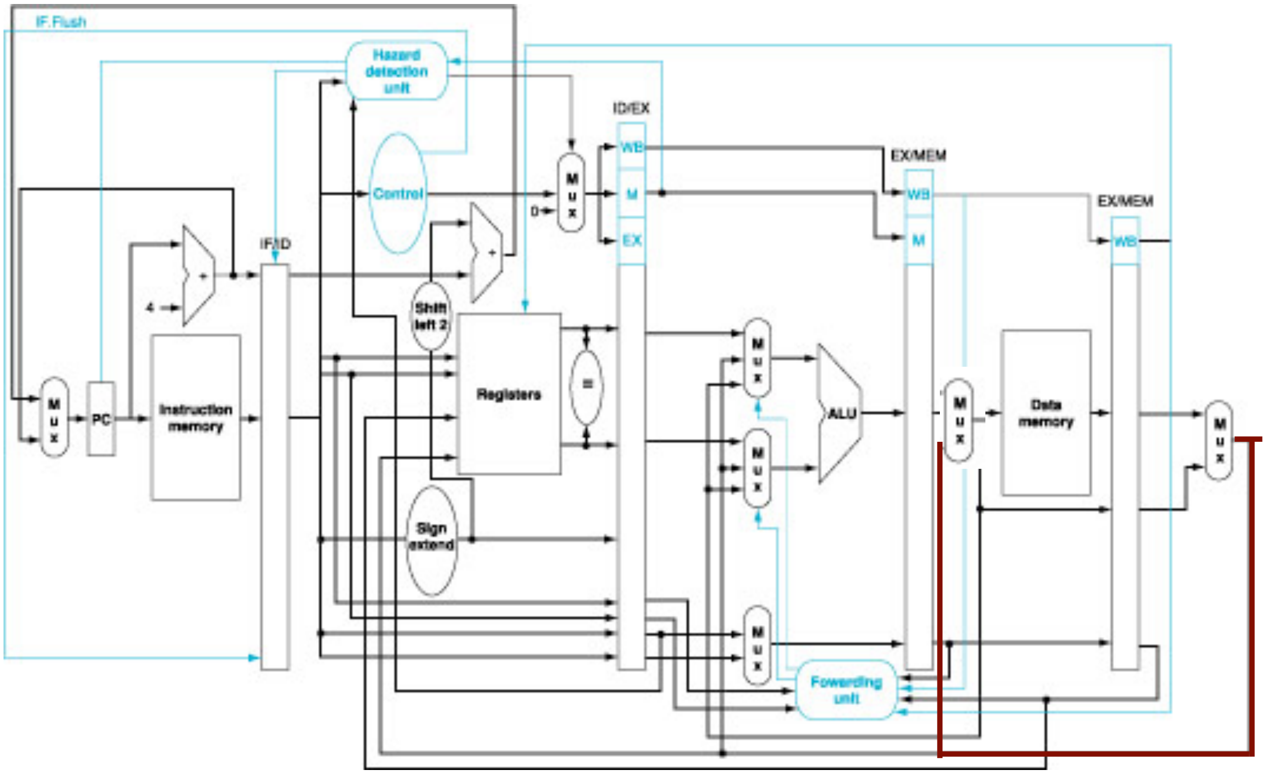
```
        add $t3, $t1, $t2

        sw $t3, 12($t0)
```

and

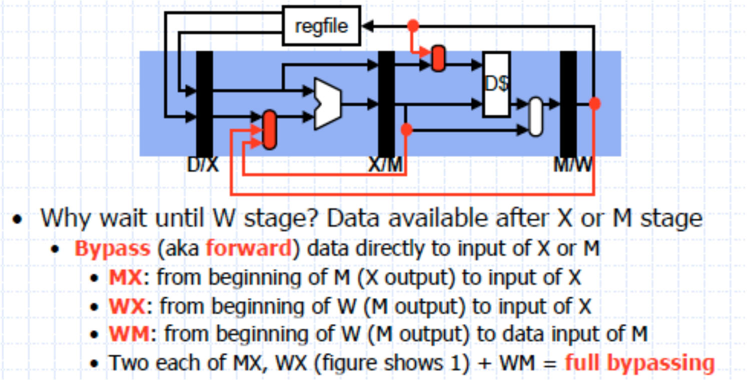
```
        add $t5, $t1, $t4

        sw $t5, 16($t0)
```



Note: alternatively you can assume no bypass, and instead insert stalls in the pipeline.

As an aside, here's a helpful summary of the forwarding paths in the 5-stage pipeline mips architecture:



- Why wait until W stage? Data available after X or M stage
  - Bypass (aka forward) data directly to input of X or M
    - MX: from beginning of M (X output) to input of X
    - WX: from beginning of W (M output) to input of X
    - WM: from beginning of W (M output) to data input of M
    - Two each of MX, WX (figure shows 1) + WM = full bypassing

C) The reordered sequence is:

```
lw $t1, 0($t0) # b is stored at 0 offset from $t0

lw $t2, 4($t0) # Word size is 4 bytes in MIPS

lw $t4, 8($t0)

add $t3, $t1, $t2

sw $t3, 12($t0)

add $t5, $t1, $t4

sw $t5, 16($t0)
```

***Note**: the following questions would have varied answers dependent on how you reordered the sequence.

Insn	1	2	3	4	5	6	7	8	9	10	11	12	13
lw $t1, 0($t0)	F	D	X	M	W								
lw $t2, 4($t0)		F	D	X	M	W							
lw $t4, 8($t0)			F	D	X	M	W						
add $t3, $t1, $t2				F	D	X	M	W					
sw $t3, 12($t0)					F	D	X	M	W				
add $t5, $t1, $t4						F	D	X	M	W			
sw $t5, 16($t0)							F	D	X	M	W		