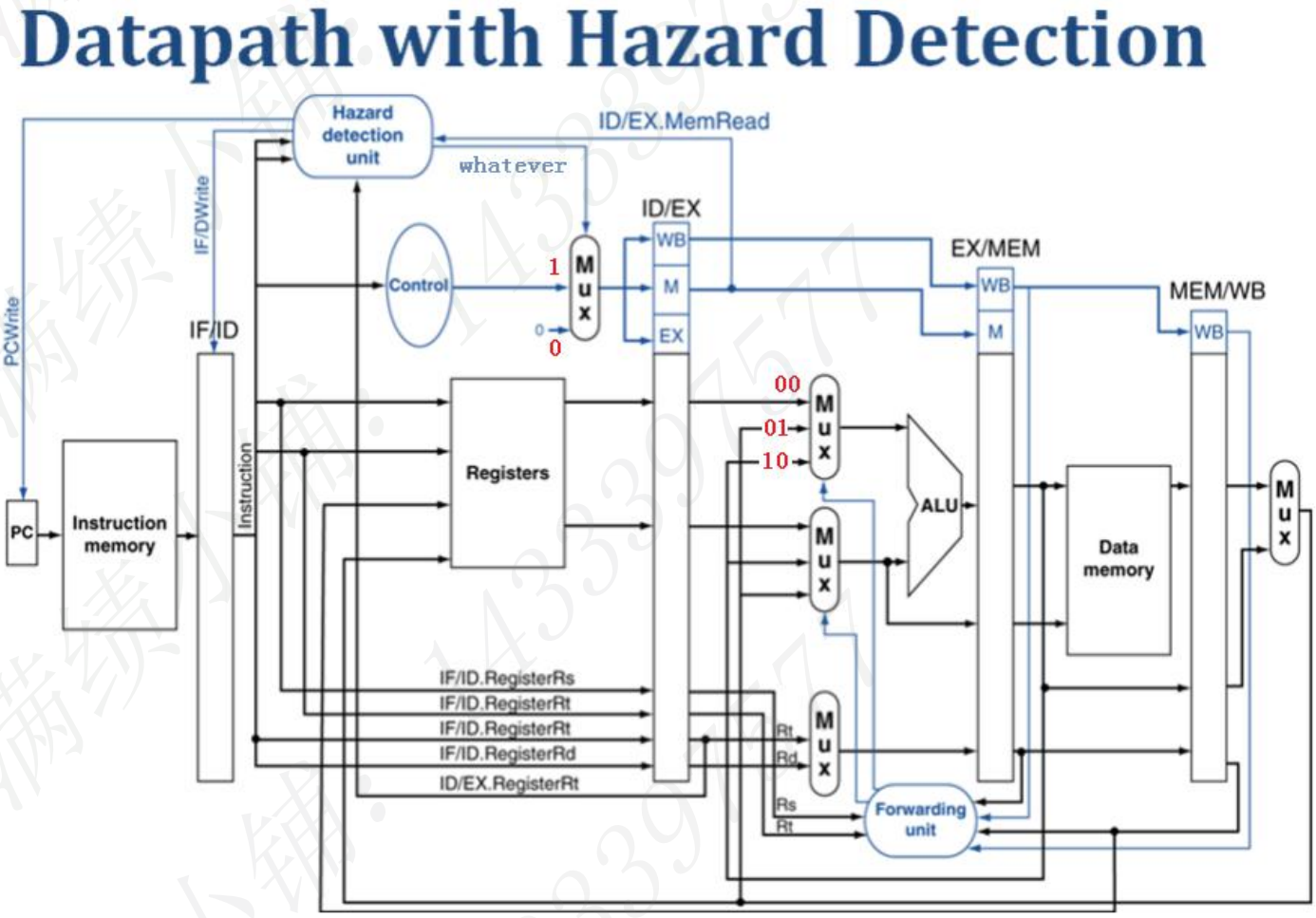
流水线示意图如下：



现有如下代码段：

Add $10,$0,$0

Addi $11,$0,100

lw $10,0($11)

Sw $10,-4($11)

Add $10,$11,$11

Addi $10,$11,-1

Beq $11,$10,exit；exit 在往下的第 89 条指令处

1. 画出在不带冒险处理机制的流水线上执行上述代码时的流水线图并用线条标明其中的数据冒险。

参考解答：

clk	1	2	3	4	5	6	7	8	9	10	11
Add $10,$0,$0	IF	ID	EX	MEM	WB						
Addi $11,$0,100		IF	ID	EX	MEM	WB					
lw $10,0($11)			IF	ID	EX	MEM	WB				
Sw $10,-4($11)				IF	ID	EX	MEM	WB			
Add $10,$11,$11					IF	ID	EX	MEM	WB		
Addi $10,$11,-1						IF	ID	EX	MEM	WB	
Beq $11,$10,exit							IF	ID	EX	MEM	WB

sw指令较特殊两个寄存器都是用来读的！！需考虑数据冒险！！

2. 画出在上述流水线上执行上述代码时的流水线图并分别用线条和方框标明其中利用的旁路以及被阻塞的指令。

参考解答：

clk	1	2	3	4	5	6	7	8	9	10	11	12
Add $10,$0,$0	IF	ID	EX	MEM	WB							
Addi $11,$0,100		IF	ID	EX	MEM	WB						
lw $10,0($11)			IF	ID	EX	MEM	WB					
Sw $10,-4($11)					IF	ID	EX	MEM	WB			
Add $10,$11,$11						IF	ID	EX	MEM	WB		
Addi $10,$11,-1							IF	ID	EX	MEM	WB	
Beq $11,$10,exit								IF	ID	EX	MEM	WB

说明：addi 和 sw 之间围绕$11 产生的的冒险因为 sw 的阻塞而消失；add 和 beq 之间围绕$10 产生的冒险因为 addi 和 beq 之间围绕$10 产生的冒险而消失。

3. 在上述流水线上执行上述代码时，在下表中说明冒险检测单元的输入和输出：

参考解答：

输入输出信号	位宽	Clk3	Clk4	Clk5	Clk6	Clk7	Clk8	Clk9
IF/ID.rs	5	0	11	11	11	11	11	11
IF/ID.rt	5	11	10	10	10	11	10	10
ID/EX.rt	5	0	11	10	10	10	11	10
ID/EX.MemRead	1	0	0	1	0	0	0	0
whatever	1	1	1	0	1	1	1	1
PCWrite	1	1	1	0	1	1	1	1
IF/IDWrite	1	1	1	0	1	1	1	1

分析如下：

I/O 信号	IF/ID.rs	IF/ID.rt	ID/EX.rt	ID/EX.MemRead	whatever	PCWrite	IF/IDWrite
位宽	5	5	5	1	1	1	1
Clk3	0	11	0	0	1	1	1
	当前指令 addi $11,$0,100　上条指令 add $10,$0,$0						
Clk4	11	10	11	0	1	1	1
	当前指令 lw $10,0($11)　上条指令 addi $11,$0,100						
Clk5	11	10	10	1	0	0	0
	当前指令 sw $10,-4($11)　上条指令 lw $10,0($11)						
Clk6	11	10	10	0	1	1	1
	当前指令 sw $10,-4($11)　上条指令 sw $10,-4($11)						
Clk7	11	11	10	0	1	1	1
	当前指令 add $10,$11,$11　上条指令 sw $10,-4($11)						
Clk8	11	10	11	0	1	1	1
	当前指令 addi $10,$11,-1　上条指令 add $10,$11,$11						
Clk9	11	10	10	0	1	1	1
	当前指令 beq $11,$10,exit　上条指令 addi $10,$11,-1						

说明：1）当利用 MemRead 来检测 lw-use 类型的冒险时，无需读数据存储器的指令的 MemRead 都必须为 0！

2）蓝色表示 sw 第一次解码执行（被阻塞），红色表示 sw 第二次解码执行（正常）；

3）有阻塞未必有冒险，比如 lw $10, 12($t0)和指令 addi $10, $t0, 100 顺序进入流水线时，这两条指令之间并不存在冒险，但是冒险检测单元会阻塞 addi 指令，这种“误解”不会影响程序的逻辑正确性：

I/O 信号	IF/ID.rs	IF/ID.rt	ID/EX.rt	ID/EX.MemRead	whatever	PCWrite	IF/IDWrite
位宽	5	5	5	1	1	1	1
	11	10	10	1	0	0	0
	当前指令 addi $10, $11, 100　　上条指令 lw $10, 12($11)						

4. 在上述流水线上执行上述代码时，在下表中说明转发单元的输入和输出：

参考解答：

输入输出信号	位宽	Clk5	Clk6	Clk7	Clk8	Clk9	Clk10
ID/EX.rs	5	11	11	11	11	11	11
ID/EX.rt	5	10	10	10	11	10	10
EX/MEM.rd	5	11	10	10/15	10/15	10	10
EX/MEM.RegWrite	1	1	1	0	0	1	1
MEM/WB.rd	5	10	11	10	10/15	10/15	10
MEM/WB.RegWrite	1	1	1	1	0	0	1
ForwardA	2	2(10)	1(01)	0(00)	0(00)	0(00)	0(00)
ForwardB	2	1(01)	2(10)	1(01)	0(00)	2(10)	2(10)

分析如下：

	当前指令（ID/EX）		上条指令（EX/MEM）		上上条指令（MEM/WB）		输出	输出
I/O 信号	rs	rt	rd	RegWrite	rd	RegWrite	ForwardA	ForwardB
位宽	5	5	5	1	5	1	2	2
Clk5	11	10	11	1	10	1	2（10）	1（01）
	Lw $10,0($11)		Addi $11,$0,100		Add $10,$0,$0			
	ForwardA：选择 addi 通过旁路转发过来的数据，有用转发 ForwardB：选择 add 通过旁路转发过来的数据，无用转发； 注意：lw 的寄存器$10 是用来写的！							
Clk6	11	10	10	1	11	1	1（01）	2（10）
	Sw $10,-4($11)		Lw $10,0($11)		Addi $11,$0,100			
	ForwardA：选择 addi 通过旁路转发过来的数据，无用转发 ForwardB：选择 lw 通过旁路转发过来的数据，无用转发 注意：sw 解码后被阻塞！							
Clk7	11	10	10/15	0	10	1	0（00）	1（01）
	Sw $10,-4($11)		Sw $10,-4($11)		Lw $10,0($11)			
	ForwardA：选择 sw 从寄存器读出的数据，无用转发 ForwardB：选择 lw 通过旁路转发过来的数据，有用转发							
Clk8	11	11	10/15	0	10/15	0	0（00）	0（00）
	Add $10,$11,$11		Sw $10,-4($11)		Sw $10,-4($11)			
	ForwardA：选择 add 从寄存器读出的数据，无用转发 ForwardB：选择 add 从寄存器读出的数据，无用转发							
Clk9	11	10	10	1	10/15	0	0（00）	2（10）

<table>
<tr><td rowspan="2"></td><td colspan="2">Addi $10,$11,-1</td><td colspan="2">Add $10,$11,$11</td><td colspan="2">Sw $10,-4($11)</td><td></td><td></td></tr>
<tr><td colspan="8">ForwardA：选择 addi 从寄存器读出的数据，无用转发
ForwardB：选择 add 通过旁路转发过来的数据，无用转发
注意：addi 的寄存器$10 是用来写的！</td></tr>
<tr><td rowspan="3">Clk10</td><td>11</td><td>10</td><td>10</td><td>1</td><td>10</td><td>1</td><td>0（00）</td><td>2（10）</td></tr>
<tr><td colspan="2">Beq $11,$10,exit</td><td colspan="2">Addi $10,$11,-1</td><td colspan="2">Add $10,$11,$11</td><td></td><td></td></tr>
<tr><td colspan="8">ForwardA：选择 beq 从寄存器读出的数据，无用转发
ForwardB：选择 addi 通过旁路转发过来的数据，有用转发</td></tr>
</table>

说明：1）蓝色表示第一次解码被阻塞的 sw 指令，红色表示第二次解码正常执行的 sw 指令；

2）有冒险一定有转发（必要时先阻塞），但有转发不一定有冒险（白转发）；无冒险也有转发（白转发）；

3）指令 sw $10,-4($11)的 rt 字段是 10，“rd”字段是 15（-4 的 16 位补码的高 4 位），sw 指令并不写寄存器，故表中会出现“10/15”，这跟 sw 指令的 RegDst 选择信号有关！