BOX A. Questions 1, 2, and 3 refer to the following Y86_64 code. We have numbered the lines so you can refer to them in your answers:

```
1. irmovq     $3, %rsi
2. irmovq     $2, %rdi
3. addq       %rsi, %rdi
4. irmovq     $5, %rsi
5. irmovq     $4, %r8
6. subq       %r8, %rsi
7. rrmovq     %rsi, %r9
8. addq       %rdi, %r9
```

Question 1

Identify the RAW data dependencies within the code in **BOX A**. For each dependence, indicate the instructions involved, the register involved, and the type of dependence.

Instruction 1	Instruction 2	Register
2	3	%rdi
1	3	rsi
4	6	rsi
6	7	rsi
3	8	rdi
7	8	r9
5	6	r8

BOX A. Questions 1, 2, and 3 refer to the following Y86_64 code. We have numbered the lines so you can refer to them in your answers:

```
1. irmovq     $3, %rsi
2. irmovq     $2, %rdi
3. addq       %rsi, %rdi
4. irmovq     $5, %rsi
5. irmovq     $4, %r8
6. subq       %r8, %rsi
7. rrmovq     %rsi, %r9
8. addq       %rdi, %r9
```

Question 2

Based on the 5-stage processor design that we discussed in class *without data forwarding*, fill in the pipeline time diagram for the code in **BOX A**. Assume that instructions stall until their required values are written into the register file. Remember that writing to the register file during the write-back stage occurs in the first half of a clock cycle and the reading from the register file in the decode stage happens during the second half of the clock cycle.
(Use labels F, D, E, M, and W for the fetch, decode, execute, memory, and writeback stages.)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
`irmovq $3, %rsi`	F	D	E	M	W															
`irmovq $2, %rdi`		F	D	E	M	W														
`addq %rsi, %rdi`			F	D	D	D	E	M	W											
`irmovq $5, %rsi`				F	F	F	D	E	M	W										
`irmovq $4, %r8`							F	D	E	M	W									
`subq %r8, %rsi`								F	D	D	D	E	M	W						
`rrmovq %rsi, %r9`									F	F	F	D	D	D	E	M	W			
`addq %rdi, %r9`												F	F	F	D	D	D	E	M	W

BOX A. Questions 1, 2, and 3 refer to the following Y86_64 code. We have numbered the lines so you can refer to them in your answers:

```
1. irmovq     $3, %rsi
2. irmovq     $2, %rdi
3. addq       %rsi, %rdi
4. irmovq     $5, %rsi
5. irmovq     $4, %r8
6. subq       %r8, %rsi
7. rrmovq     %rsi, %r9
8. addq       %rdi, %r9
```

<u>Question 3</u>

Based on the PIPE processor design we discussed in class *with data forwarding*, fill in the pipeline time diagram for the code in **BOX A**. When data is forwarded, draw an arrow between the two stages involved and indicate what register's value is being forwarded.
(Use labels F, D, E, M, and W for the fetch, decode, execute, memory, and writeback stages.)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
`irmovq $3, %rsi`	F	D	E	M	W				M->E	rsi				
`irmovq $2, %rdi`		F	D	E	M	W			E->E	rdi				
`addq %rsi, %rdi`			F	D	E	M	W							
`irmovq $5, %rsi`				F	D	E	M	W		M->E	rsi			
`irmovq $4, %r8`					F	D	E	M	W			E->E	r8	
`subq %r8, %rsi`						F	D	E	M	W		E->E	rsi	
`rrmovq %rsi, %r9`							F	D	E	M	W		E->E	r9
`addq %rdi, %r9`								F	D	E	M	W		

BOX A. Questions 4, 5, and 6 refer to the following Y86_64 code. We have numbered the lines so you can refer to them in your answers:

```
1. mrmovq     (%rsp), %rsi
2. addq       %rsi, %rsi
3. irmovq     $3, %rdi
4. addq       %rsi, %rdi
5. mrmovq     8(%rsp), %r8
6. addq       %rdi, %rsi
7. addq       %r8, %rsi
```

Question 4

Identify the RAW data dependencies within the code in **BOX A**. For each dependence, indicate the instructions involved, the register involved, and the type of dependence.

Instruction 1	Instruction 2	Register
1	2	%rsi
2	6	rsi
6	7	rsi
2	4	rsi
3	4	rdi
4	6	rdi
5	7	r8

Question 5
Based on the 5-stage processor design that we discussed in class *without data forwarding*, fill in the pipeline time diagram for the code in **BOX A**. Assume that instructions stall until their required values are written into the register file. Remember that writing to the register file during the write-back stage occurs in the first half of a clock cycle and the reading from the register file in the decode stage happens during the second half of the clock cycle.
(Use labels F, D, E, M, and W for the fetch, decode, execute, memory, and writeback stages.)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
`mrmovq (%rsp), %rsi`	F	D	E	M	W													
`addq %rsi, %rsi`		F	D	D	D	E	M	W										
`irmovq $3, %rdi`			F	F	F	D	E	M	W									
`addq %rsi, %rdi`						F	D	D	D	E	M	W						
`mrmovq 8(%rsp), %r8`							F	F	F	D	E	M	W					
`addq %rdi, %rsi`										F	D	D	E	M	W			
`addq %r8, %rsi`											F	F	D	D	D	E	M	W

Question 6
Based on the PIPE processor design we discussed in class *with data forwarding*, fill in the pipeline time diagram for the code in **BOX A**. When data is forwarded, draw an arrow between the two stages involved and indicate what register's value is being forwarded.
(Use labels F, D, E, M, and W for the fetch, decode, execute, memory, and writeback stages.)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
`mrmovq (%rsp), %rsi`	F	D	E	M	W			M->E	rsi					
`addq %rsi, %rsi`		F	D	E	E	M	W		M->E	rsi				
`irmovq $3, %rdi`			F	D	D	E	M	W		E->E	rdi			
`addq %rsi, %rdi`				F	F	D	E	M	W		M->E	rdi		
`mrmovq 8(%rsp), %r8`						F	D	E	M	W			M->E	r8
`addq %rdi, %rsi`							F	D	E	M	W		E->E	rsi
`addq %r8, %rsi`								F	D	E	M	W		