Superscalar Problem

Consider the computation $((M1 * M2) + (M3 * M4))/(M5 + M6)$, where $M1, \ldots$ denote the content of memory locations.

1. Write an assembly program for this computation. Use a minimal number of registers.
2. Show the execution of your program on a superscalar processor. Assume that there is an instruction window where all the fetched and decoded instructions can be stored. The remaining pipeline stages are RR, EX and WB. Assume that there are four fetch units, two decode units, two units for reading the registers, two ALUs and two units writing to the registers.
3. Identify the dependencies in your code.
4. Remove the false dependencies by using register renaming.
5. Draw a diagram showing the execution of the modified code on the above superscalar processor using.

Solution:

1. Here is an assembly code satisfying the requirements:

```
 I1: LOAD r1, M1
 I2: LOAD r2, M2
 I3: MUL r1, r2
 I4: LOAD r2, M3
 I5: LOAD r3, M4
 I6: MUL r2, r3
 I7: ADD r1, r2
 I8: LOAD r2, M5
 I9: LOAD r3, M6
I10: ADD r2, r3
I11: DIV r1, r2
```

2. Here is a diagram showing the execution of the code. IW denotes the instruction window where fetched and decoded instructions are stored. We do not show the IF and ID decode stages and we start with the stage when I1 and I2 have just arrived into the IW (so that they are ready for further processing in the next cycle).

	IW	RR	EX	WB	Comments
0	I1,I2				
1	I3,I4		I1,I2		
2	I3,I5,I6		I4	I1,I2	r1,r2 not ready
3	I6,I7,I8	I3	I5		r2 in use, r1,r2,r3 not ready
4	I6,I7,I9,I10		I3,I8	I4,I5	
5	I6,I7,I10,I11	I6	I9	I3	r2 not ready
6	I7,I10,I11		I6	I9	r3 in use
7	I7,I10,I11			I6	
8	I10,I11	I7			
9	I10,I11		I7	I8	
10	I11	I10		I7	
11	I11		I10		
12	I11			I10	
13		I11			
14			I11		
15				I11	

3. Write–read (or true data) dependency: all the pairs of occurrences of a register ri where instruction Ij writes to ri and instruction Ik reads from ri with $j < k$. For instance, $I1 - I3$ on $r1$. Also $I2 - I3(r2)$, $I4 - I6(r2)$, $I5 - I6(r3)$, $I3 - I7(r1)$, $I6 - I7(r2)$, $I8 - I10(r2)$, $I9 - I10(r3)$, $I7 - I11(r1)$, $I10 - I11(r2)$.

Write–write (false or fake) dependency: all the pairs of occurrences of a register ri where instruction Ij writes to ri and instruction Ik also writes to ri with $j < k$. For instance, $I2 - I4$ on $r2$. Also $I1 - I3/I7/I11(r1)$, $I2 - I4/I6/I8/I10(r2)$, $I3 - I7/I11(r1)$, $I4 - I6/I8/I10(r2)$, $I5 - I9(r3)$, $I6 - I8/I10(r2)$, $I7 - I11(r1)$, $I8 - I10(r2)$.

Read–write (false or fake) dependency: all the pairs of occurrences of a register ri where instruction Ij reads from ri and instruction Ik writes to ri with $j < k$. For instance, $I3-I4$ on $r2$. Also $I3-I7/I11(r1)$, $I3-I4/I6/I8/I10(r2)$, $I6-I8/I10(r2)$, $I6 - I9(r3)$, $I7 - I11(r1)$, $I7 - I8/I10(r2)$.

4. We can rename $r2$ in $I4, I6, I7$ to $r2a$ and in $I8, I10, I11$ to $r2b$ (thus removing the write–write and read–write dependencies on $r2$) and also $r3$ in $I9, I10$ to $r3a$.

```
  I1: LOAD r1, M1
  I2: LOAD r2, M2
  I3: MUL r1, r2
 I4': LOAD r2a, M3
 I5': LOAD r3, M4
 I6': MUL r2a, r3
 I7': ADD r1, r2a
 I8': LOAD r2b, M5
 I9': LOAD r3a, M6
I10': ADD r2b, r3a
I11': DIV r1, r2b
```

5. Here is the diagram after register renaming.

	IW	RR	EX	WB	Comments
0	I1,I2				
1	I3,I4'		I1,I2		
2	I3,I5',I6'		I4'	I1,I2	r1,r2 not ready
3	I6',I7',I8'	I3	I5'	I4'	r2a,r3 not ready
4	I6',I7',I9',I10'		I3,I8'	I5'	r1,r2a not ready
5	I7',I10',I11'	I6'	I9'	I3,**I8'**	r2b,r3a not ready
6	I7',I10',I11'		I6'	I9'	r1,r2b not ready
7	I7',I11'	**I10'**		I6'	
8	I11'	I7'	I10'		
9	I11'		I7'	I10'	
10	I11'			I7'	
11		**I11'**			
12			I11'		
13				I11'	