Example — Memory Addressing

Consider the computation $((M1 * M2) + M3) - M4$ where each Mi indicates the content of a memory location.

1. Write an assembly program for the above computation using minimum number of registers.

2. Show the delay slots when the code is executed on a five-stage pipeline. Assume that instructions and data have to be loaded from main memory. Assume that there is an instruction window IW, where fetched and decoded instructions can be stored until they can be further processed (out-of-order if possible).

3. Repeat the previous item, but now assume that instructions and data are loaded from onboard caches.

Solution to Example — Memory Addressing

1. Here is the code:

```
I1  LOAD r1, M1
I2  LOAD r2, M2
I3  MUL r1, r1, r2
I4  LOAD r2, M3
I5  ADD r1, r1, r2
I6  LOAD r2, M4
I7  SUB r1, r1, r2
```

2. Here is the diagram showing delay slots:

	IF	ID	IW	RR	EX	WB	Comments
1	I1						
2	I2	I1					
3		I2			I1		I1 skips RR, data bus busy
4					I2	I1	I2 skips RR, data bus busy
5	I3					I2	
6	I4	I3					
7	I5	I4		I3			
8	I6	I5	I4		I3		I4 skips RR
9		I6	I5		I4	I3	r2 not ready, data bus busy
10			I5		I6	I4	data bus busy
11			I7	I5			
12			I7		I5	I6	
13			I7			I5	r1 not ready
14				I7			
15					I7		
16						I7	

In cycles 3 and 4, I3 cannot be fetched (the data bus is busy with loading M1 and M2), in cycles 9 and 10, I7 cannot be fetched (the data bus is busy with loading M3 and M4). In cycle 10, I6 can execute, since r2 is updated only in WB (in cycle 12), and I5 has read r2 in cycle 11. In cycles 9 and 10, I5 has to wait (r2 is not ready yet), and in cycle 12, I7 has to wait (r1 is not ready yet).

3. Using onboard caches speeds up the execution.

	IF	ID	IW	RR	EX	WB	Comments
1	I1						
2	I2	I1					
3	I3	I2				I1	I1 skips RR,EX
4	I4	I3				I2	I2 skips RR,EX
5	I5	I4		I3			
6	I6	I5			I3	I4	I4 skips RR,EX
7	I7	I6	I5			I3	r1 not ready
8		I7	I6	I5			r2 in use
9			I7		I5	I6	I6 skips RR,EX, r1,r2 not ready
10			I7			I5	
11				I7			
12					I7		
13						I7	