Simple Example — Immediate Addressing

Consider the computation $(1 + 2) * 3$.

1. Write an assembly program for the above computation and store the result in register r1. You can assume that there are an unlimited number of regsiters available.

2. Describe what happens in the pipeline stages when various types of instructions (data-movement, data-processing) are processed on a five-stage pipeline: fetch IF, decode ID, register read RR, execute EX and write back WB.

3. Draw a diagram showing the in-order execution of the code and identify the delay slots.

Solution to Simple Example — Immediate Addressing

1. Here is the code:

```
I1  LOAD r1, #1
I2  LOAD r2, #2
I3  LOAD r3, #3
I4  ADD r1, r1, r2
I5  MUL r1, r1, r3
```

2. All instruction go through IF (instruction is loaded onto the CPU) and ID (control unit decodes the instruction). Using immediate addressing does not need reading registers or data transfer, so I1, I2 and I3 skip RR and EX. For these instructions data is written into the target register in WB. Arithmetic instructions need RR (the arguments of the operations are "picked up" from the registers, i.e., the contents of the registers are loaded into internal registers of the execution unit), EX (the operations are carried out) and WB (the results are written into the registers).

3. Here is the diagram showing delay slots (empty pipeline stages):

	IF	ID	RR	EX	WB	Comments
1	I1					
2	I2	I1				
3	I3	I2			I1	I1 skips RR and EX
4	I4	I3			I2	I2 skips RR and EX
5	I5	I4			I3	I3 skips RR and EX
6		I5	I4			
7		I5		I4		r1 not ready
8		I5			I4	r1 not ready
9			I5			
10				I5		
11					I5	

In cycles 7 and 8, I5 has to wait (r1 is not ready yet). This is called *data dependency* between I4 and I5 on r1.