

Fibonacci sequence with delayed branch

In this exercise we will see

- how to write an assembly code with iteration (using branch instructions),
- the effect of (conditional) branch instructions on the pipeline activity,
- the importance of branch prediction and
- how the “delayed branch” technique works.

The Fibonacci sequence F is defined as $F(1) = F(2) = 1$ and for $n \geq 2$,

$$F(n+1) = F(n) + F(n-1)$$

i.e., the $(n+1)$ th value is given by the sum of the n th value and the $(n-1)$ th value.

1. Write an assembly program typical of RISC machines for computing the k th value $F(k)$, where k is a natural number greater than 2 loaded from a memory location M , and storing the result at memory location M .
2. Show the execution of your program on a pipelined processor when M contains the natural number 4. Assume that instructions are fetched from an onboard cache and that there is an instruction window register IW, where two fetched and decoded instructions can be stored. The pipeline stages are F(etch), D(ecode), R(egister read), E(xecute) and W(rite back). Explain where and why delay slots appear.
3. Repeat item 2, but assume that there is a branch prediction that always gets it right.
4. Repeat item 2 by applying the delayed branch technique.

Solution:

1. Here is a suitable code (other solutions exist):

```
1.  LOAD r2, M
2.  LOAD r0, #1
3.  LOAD r1, #1
4.  SUB r2, r2, #1
5.  ADD r3, r0, r1
6.  LOAD r0, r1
7.  LOAD r1, r3
8.  BNE 4, r2, #2    // jump to instruction 4 if r2 is not equal to 2
9.  STOR M, r3
```

where # indicates immediate addressing and BNE stands for “branch if not equal”.

2. We assume that data movement between the CPU and memory occurs in E and that loading a numerical value or a content of a register skips E. For conditional branch instructions, checking the condition happens in E and updating the PC in W. Instructions after the branch are not processed until the outcome of the branch is known. The sign **X** shows the delay caused by the conditional branch.

	F	D	IW	R	E	W	Comments
1	I1						
2	I2	I1					
3	I3	I2			I1		I1 skips R
4	I4	I3	I2			I1	I2 skips R,E, r2=4
5	I5	I4	I3			I2	I3 skips R,E
6	I6	I5		I4		I3	
7	I7	I6		I5	I4		
8	I8	I7		I6	I5	I4	r2=3
9	I9	I8	I7			I5	I6 skips E, W busy, r3 not ready
10		I9	I8	I7		I6	R busy
11			I9	I8		I7	I7 skips E
12			I9	X	I8		condition holds
13			I9	X		I8	PC updated
14	I4		I9	X			
15	I5	I4	I9	X			
16	I6	I5	I9	I4			
17	I7	I6	I9	I5	I4		
18	I8	I7	I9	I6	I5	I4	r2=2
19		I8	I7,I9			I5	r3 not ready
20			I8,I9	I7		I6	R busy
21			I9	I8		I7	
22			I9	X	I8		condition fails
23				I9			no need to update PC
24					I9		

3. Assuming that the branch prediction is correct we can get rid of some delay slots. We assume that branch prediction happens once the conditional branch instruction has been decoded.

	F	D	IW	R	E	W	Comments
1	I1						
2	I2	I1					
3	I3	I2			I1		I1 skips R
4	I4	I3	I2			I1	I2 skips R,E, r2=4
5	I5	I4	I3			I2	I3 skips R,E
6	I6	I5		I4		I3	
7	I7	I6		I5	I4		
8	I8	I7		I6	I5	I4	r2=3
9	I9	I8	I7			I5	I6 skips E, W busy, r3 not ready
10		I9	I8	I7		I6	R busy, I4 is predicted
11	I4	I9		I8		I7	I7 skips E
12	I5	I4	I9	X	I8		condition holds, prediction correct
13	I6	I5	I9	I4			
14	I7	I6	I9	I5	I4		
15	I8	I7	I9	I6	I5	I4	r2=2
16		I8	I7,I9			I5	r3 not ready
17			I8,I9	I7		I6	R busy, I9 predicted
18			I9	I8		I7	
19			I9	X	I8		condition fails, prediction correct
20				I9			
21					I9		

4. With delayed branch I7 and I8 are swapped and the effect of I8 is delayed so that I7 can finish. This can be done, because I7 and I8 are independent of each other.

```

1.  LOAD r2, M
2.  LOAD r0, #1
3.  LOAD r1, #1
4.  SUB r2, r2, #1
5.  ADD r3, r0, r1
6.  LOAD r0, r1
8.  BNE 4, r2, #2      // jump to instruction 4 if r2 is not equal to 2
7.  LOAD r1, r3
9.  STOR M, r3

```

	F	D	IW	R	E	W	Comments
1	I1						
2	I2	I1					
3	I3	I2			I1		I1 skips R
4	I4	I3	I2			I1	I2 skips R,E, r2=4
5	I5	I4	I3			I2	I3 skips R,E
6	I6	I5		I4		I3	
7	I8	I6		I5	I4		
8	I7	I8		I6	I5	I4	r2=3
9	I9	I7		I8		I5	I6 skips E, W busy
10		I9		I7	I8	I6	condition holds
11			I9			I8	I7 skips E, PC updated
12	I4		I9			I7	I7 fills delay slot
13	I5	I4	I9	X			
14	I6	I5	I9	I4			
15	I8	I6	I9	I5	I4		
16	I7	I8	I9	I6	I5	I4	r2=2
17		I7	I9	I8		I5	
18			I9	I7	I8	I6	condition fails
19			I9			I7	no need to update PC, r1 not ready
20				I9			
21					I9		