ADVANCED PIPELINING FOR INTEGER OPERATION

Example

Assume that we have…

- a) Pipeline contains 5 stages: IF, ID, EX, M and W;
- b) Each stage requires one clock cycle;
- c) All memory references hit in cache;

The following program should be processed

```
// ADD TWO INTEGER ARRAYS
      LW R4 # 400
L1:   LW R1, 0 (R4) ; Load first operand
      LW R2, 400 (R4) ; Load second operand
      ADDI R3, R1, R2 ; Add operands
      SW R3, 0 (R4)   ; Store result
      SUB R4, R4, #4 ; Calculate address of
                     ;next element
      BNEZ R4, L1     ; Loop if (R4) != 0
```

Question 1

- Calculate how many clock cycles will take execution of this segment on the regular (nonpipelined) architecture. Show calculations:

Solution 1

Number of cycles

= [Initial instruction + (Number of instructions in the loop L1) x number of loop cycles] x number of clock cycles / instruction (CPI)

Solution 1 continued

Number of cycles

= [1 + (6) x 400/4] x 5 c.c. = 3005 c.c.

Question 2

- Calculate how many clock cycles will take execution of this segment on the simple pipeline without forwarding or bypassing when result of the branch instruction (new PC content) is available after WB stage.

Solution

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LW R1, 0 (R4)	IF	ID	Ex	M	W											
LW R2, 400 (R4)		IF	ID	Ex	M	W										
ADDI R3, R1,			IF	ID	*	*										
SW R3, 0 (R4)				IF	*	*										
SUB R4, R4, #4																
BNEZ R4, L1																

Why stall at cycle 5 and 6

- Two stall cycles (c.c. # 5 and 6) are caused by the delay of data in the register R2 for the ADDI
- Same stall cycles in ID stage for the SW instruction are because ID stage circuits are busy for ADDI and becoming available only

Solution

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LW R1, 0 (R4)	IF	ID	Ex	M	W											
LW R2, 400 (R4)		IF	ID	Ex	M	W										
ADDI R3, R1,			IF	ID	*	*	Ex	M								
SW R3, 0 (R4)				IF	*	*	ID	Ex								
SUB R4, R4, #4							IF	ID								
BNEZ R4, L1								IF								

Why SUB start at 8^{th} cc

- SUB can start only on 8-th c.c. because IF stage is busy with SW instruction

Solution

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LW R1, 0 (R4)	IF	ID	Ex	M	W											
LW R2, 400 (R4)		IF	ID	Ex	M	W										
ADDI R3, R1,			IF	ID	*	*	Ex	M	W							
SW R3, 0 (R4)				IF	*	*	ID	Ex	*	M						
SUB R4, R4, #4							IF	ID	*	Ex						
BNEZ R4, L1								IF	*	ID						

Why one c.c. stall

- One c.c. stall in the pipeline happens because the content of R3 (for SW) is not ready. However, “Ex” stage can be executed for SW instruction. This becomes possible because during the “Ex” stage the address in memory is calculated (only for Load or Store instruction

Solution

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LW R1, 0 (R4)	IF	ID	Ex	M	W											
LW R2, 400 (R4)		IF	ID	Ex	M	W										
ADDI R3, R1,			IF	ID	*	*	Ex	M	W							
SW R3, 0 (R4)				IF	*	*	ID	Ex	*	M	W					
SUB R4, R4, #4							IF	ID	*	Ex	M	W				
BNEZ R4, L1								IF	*	ID	*	*	Ex	M	W	

Why two stalls at c.c. 11 and 12

- Two stall cycles (c.c. # 11 and 12) in BNEZ are coming from the delay of updating the R4. New content of R4 becomes available only after 12 c.c. Thus, the content of PC is updated on W-stage of BNEZ (after15 c.c.).

Solution (number of cycles)

Number of cycles in the loop = 15 c.c.

Number of clock cycles for segment execution on pipelined processor

= 1 c.c. (IF stage of the initial instruction) + (Number of clock cycles in the loop L1) x Number of loop cycles = 1 + 15 x 400/4

= 1501 c.c.

Solution (speed up)

Speedup of the pipelined processor comparing with non-pipelined processor

= Number of Clock cycles for the segment execution on non-pipelined processor / Number of Clock cycles for the segment execution on simple pipelined processor

= 3005 c.c. / 1501 = 2 times

Credit to

- http://www.ee.ryerson.ca/~lkirisch/coe818/midfin/Solutions_COE818_MT.pdf