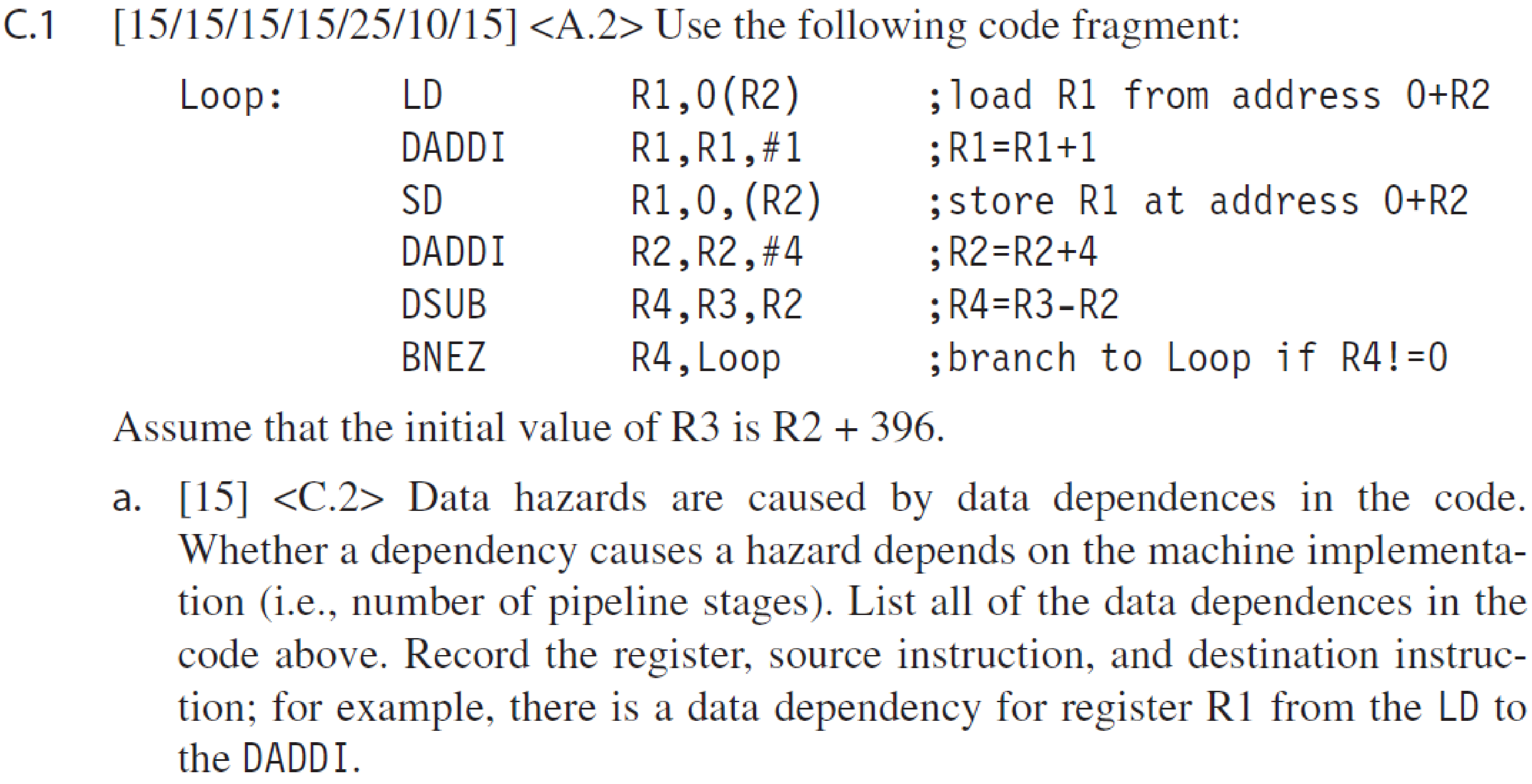
Computer Architecture (Spring 2020)

Pipelining

Dr. Duo Liu (刘铎)
Office: Main Building 0626
Email: liuduo@cqu.edu.cn

Example



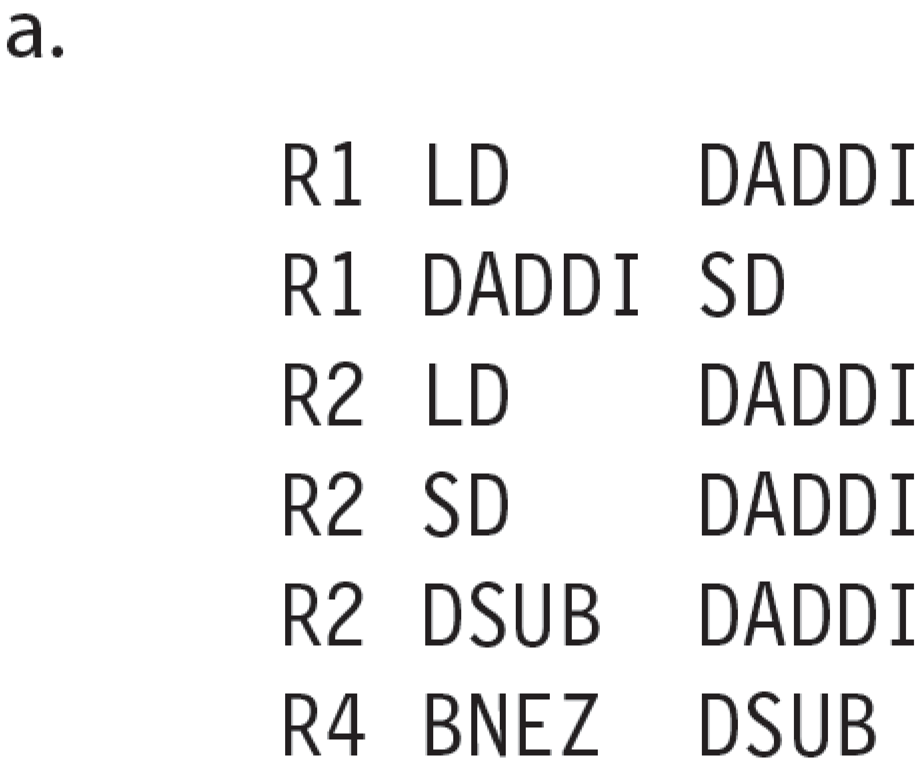
C.1 [15/15/15/15/25/10/15] <A.2> Use the following code fragment:

```
Loop:   LD      R1,0(R2)     ;load R1 from address 0+R2
        DADDI   R1,R1,#1     ;R1=R1+1
        SD      R1,0,(R2)    ;store R1 at address 0+R2
        DADDI   R2,R2,#4     ;R2=R2+4
        DSUB    R4,R3,R2     ;R4=R3-R2
        BNEZ    R4,Loop      ;branch to Loop if R4!=0
```

Assume that the initial value of R3 is R2 + 396.

a. [15] <C.2> Data hazards are caused by data dependences in the code. Whether a dependency causes a hazard depends on the machine implementation (i.e., number of pipeline stages). List all of the data dependences in the code above. Record the register, source instruction, and destination instruction; for example, there is a data dependency for register R1 from the LD to the DADDI.

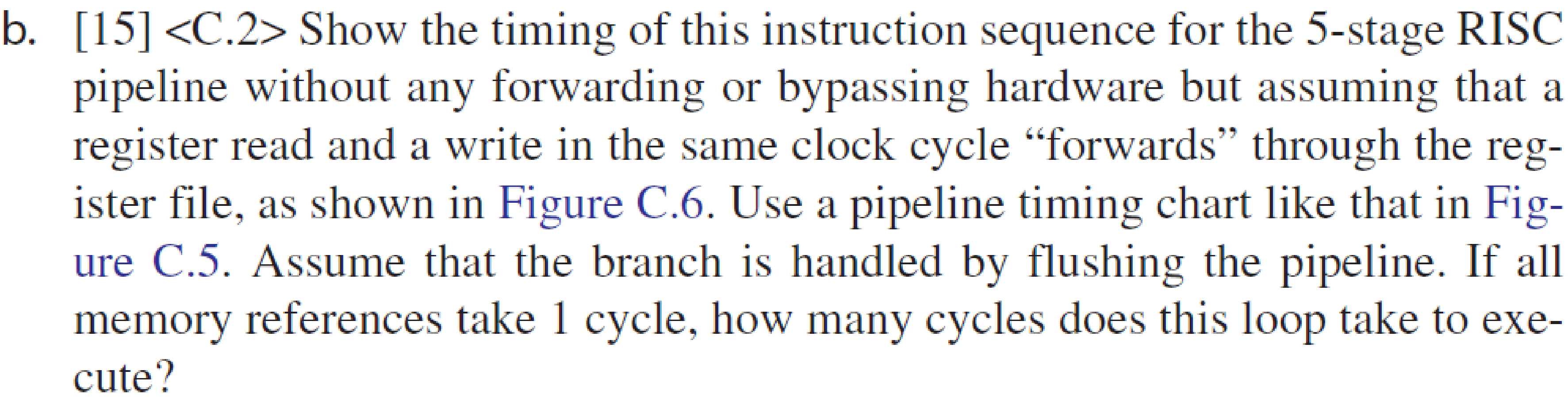
Example



a.

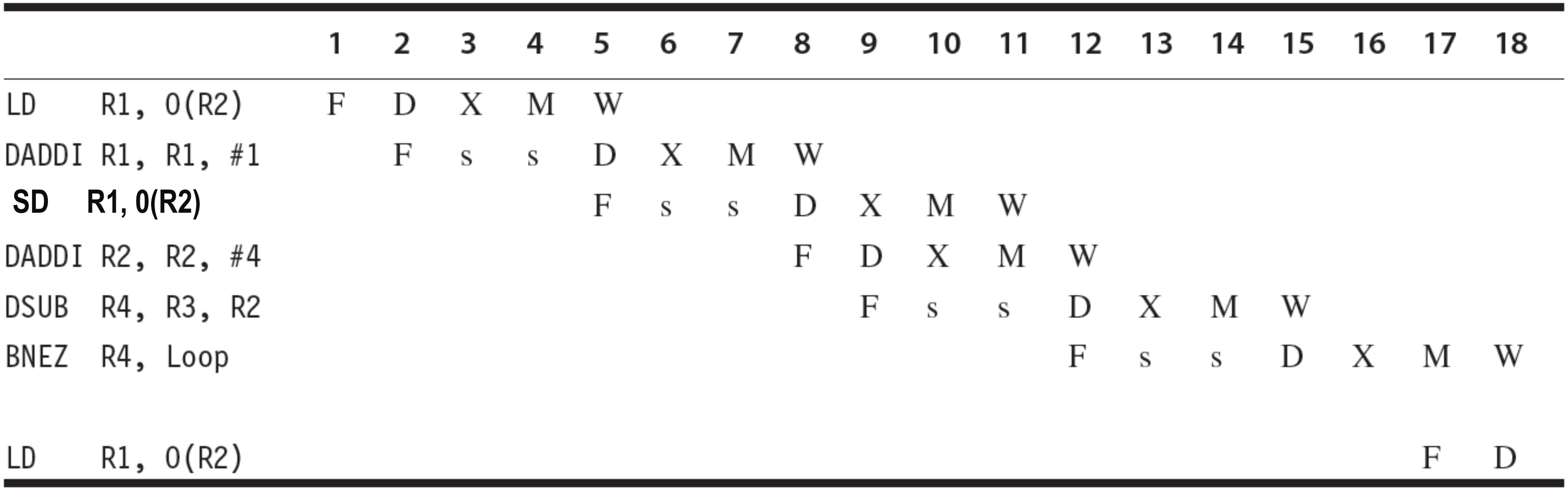
R1	LD	DADDI
R1	DADDI	SD
R2	LD	DADDI
R2	SD	DADDI
R2	DSUB	DADDI
R4	BNEZ	DSUB

Example

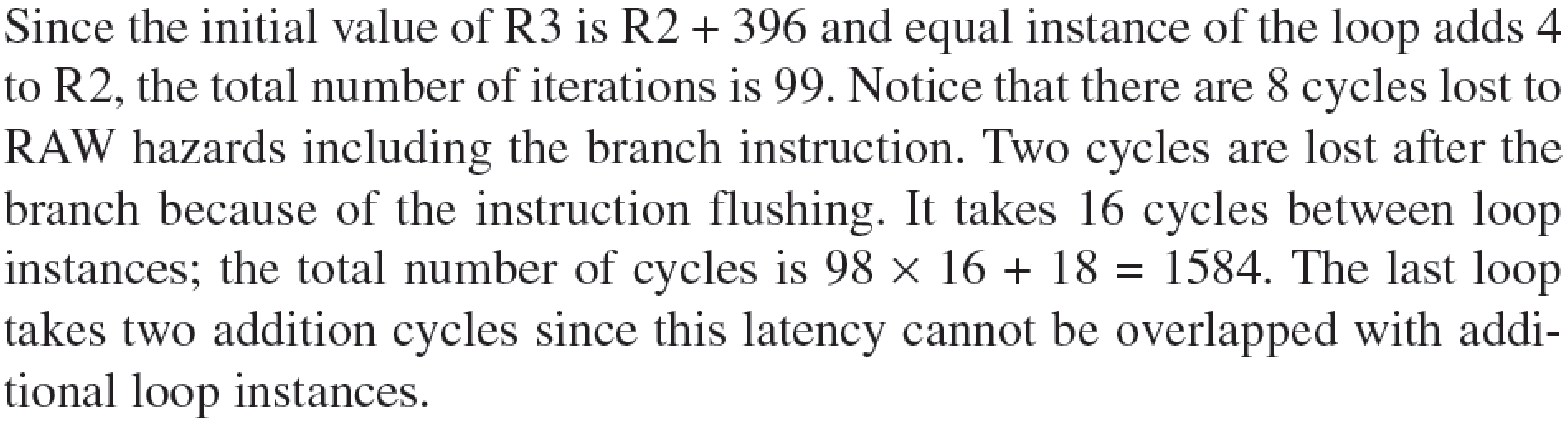


b. [15] <C.2> Show the timing of this instruction sequence for the 5-stage RISC pipeline without any forwarding or bypassing hardware but assuming that a register read and a write in the same clock cycle "forwards" through the register file, as shown in Figure C.6. Use a pipeline timing chart like that in Figure C.5. Assume that the branch is handled by flushing the pipeline. If all memory references take 1 cycle, how many cycles does this loop take to execute?

Example

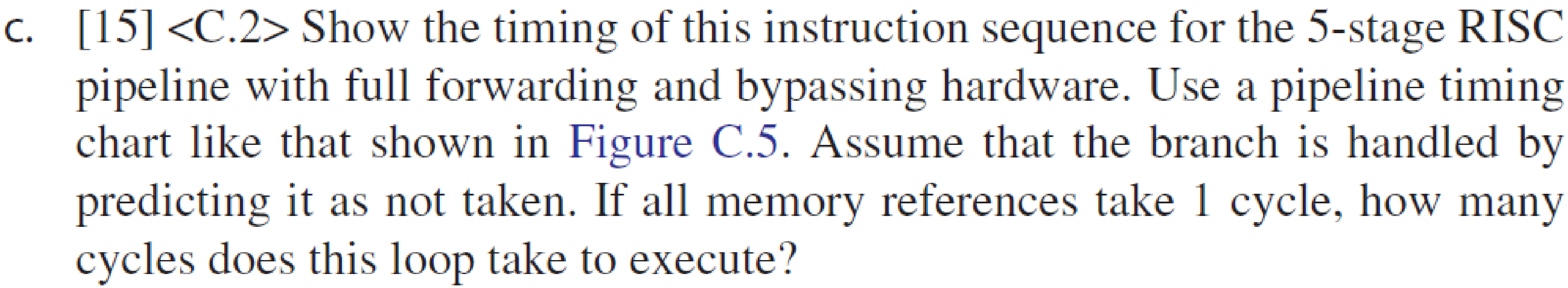


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
LD R1, 0(R2)	F	D	X	M	W													
DADDI R1, R1, #1		F	s	s	D	X	M	W										
SD R1, 0(R2)					F	s	s	D	X	M	W							
DADDI R2, R2, #4								F	D	X	M	W						
DSUB R4, R3, R2									F	s	s	D	X	M	W			
BNEZ R4, Loop												F	s	s	D	X	M	W
LD R1, 0(R2)																	F	D



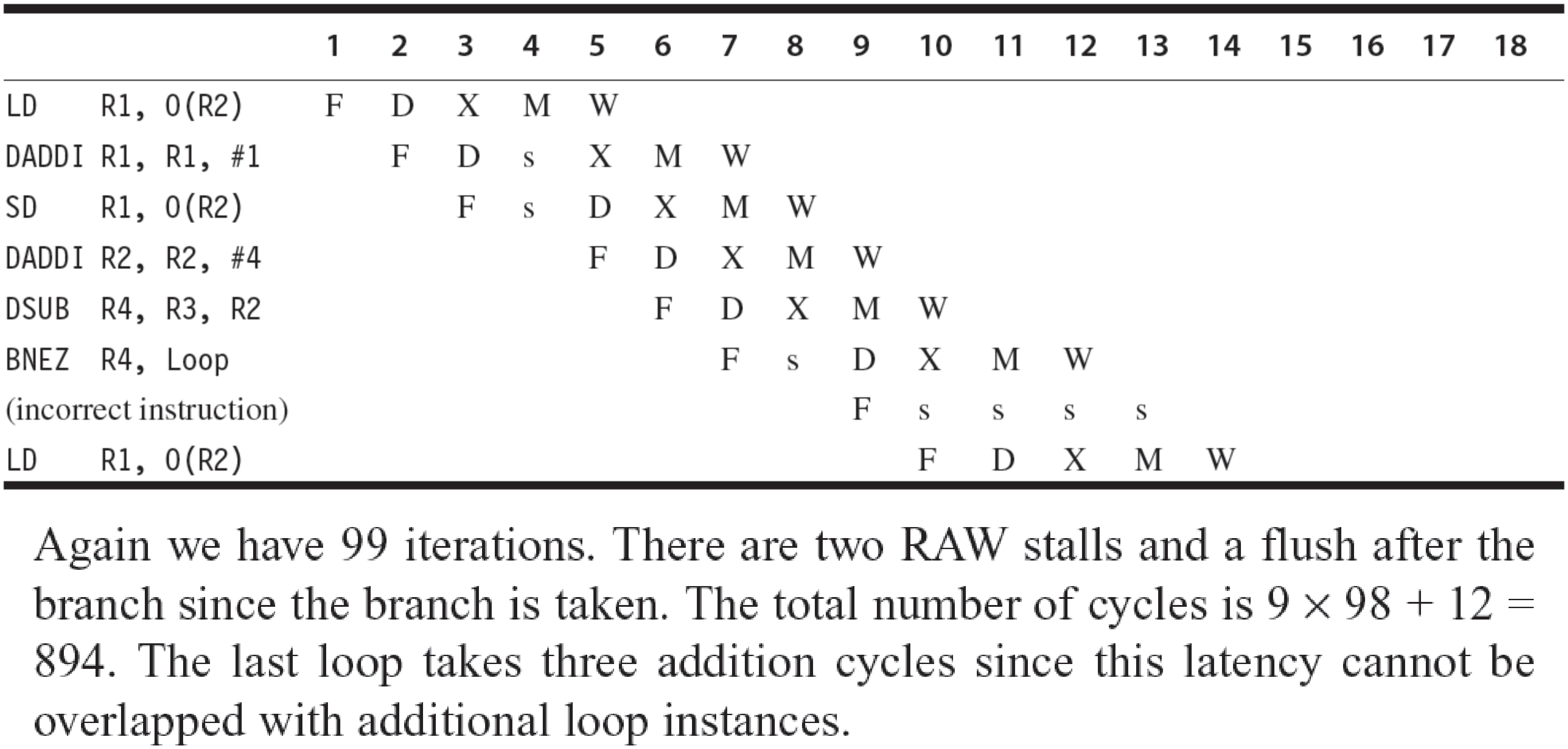
Since the initial value of R3 is R2 + 396 and equal instance of the loop adds 4 to R2, the total number of iterations is 99. Notice that there are 8 cycles lost to RAW hazards including the branch instruction. Two cycles are lost after the branch because of the instruction flushing. It takes 16 cycles between loop instances; the total number of cycles is 98 × 16 + 18 = 1584. The last loop takes two addition cycles since this latency cannot be overlapped with additional loop instances.

Example



c. [15] <C.2> Show the timing of this instruction sequence for the 5-stage RISC pipeline with full forwarding and bypassing hardware. Use a pipeline timing chart like that shown in Figure C.5. Assume that the branch is handled by predicting it as not taken. If all memory references take 1 cycle, how many cycles does this loop take to execute?

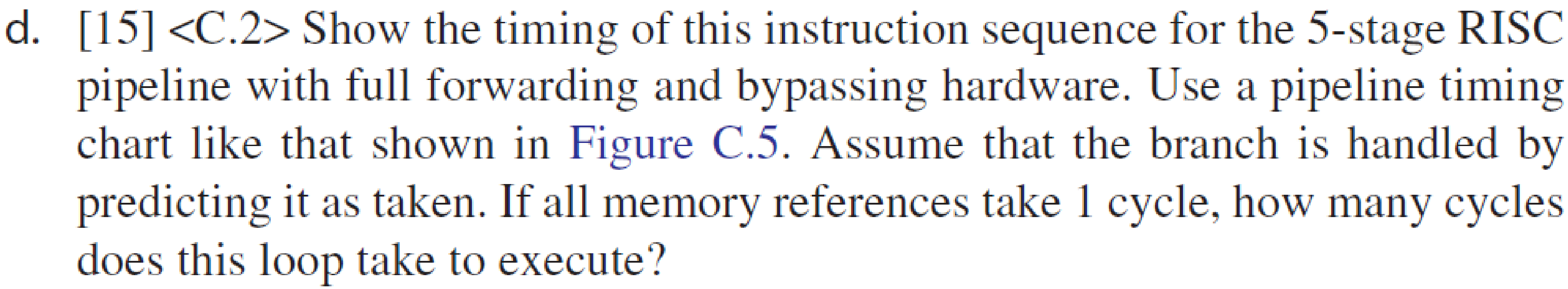
Example



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
`LD R1, 0(R2)`	F	D	X	M	W													
`DADDI R1, R1, #1`		F	D	s	X	M	W											
`SD R1, 0(R2)`			F	s	D	X	M	W										
`DADDI R2, R2, #4`					F	D	X	M	W									
`DSUB R4, R3, R2`						F	D	X	M	W								
`BNEZ R4, Loop`							F	s	D	X	M	W						
(incorrect instruction)									F	s	s	s	s					
`LD R1, 0(R2)`										F	D	X	M	W				

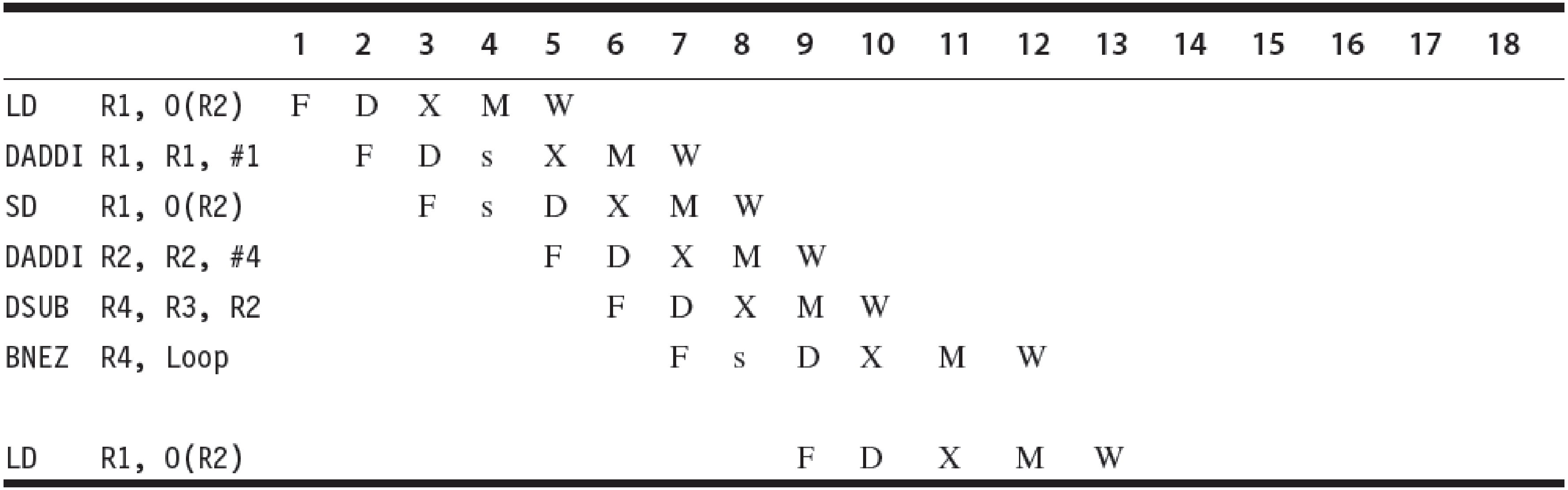
Again we have 99 iterations. There are two RAW stalls and a flush after the branch since the branch is taken. The total number of cycles is 9 × 98 + 12 = 894. The last loop takes three addition cycles since this latency cannot be overlapped with additional loop instances.

Example

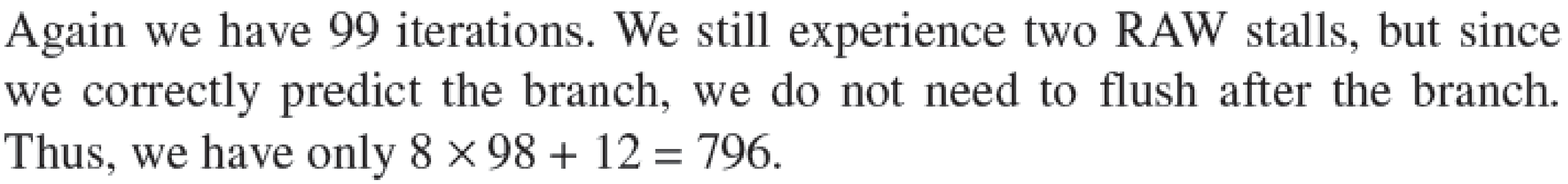


d. [15] <C.2> Show the timing of this instruction sequence for the 5-stage RISC pipeline with full forwarding and bypassing hardware. Use a pipeline timing chart like that shown in Figure C.5. Assume that the branch is handled by predicting it as taken. If all memory references take 1 cycle, how many cycles does this loop take to execute?

Example



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
LD R1, 0(R2)	F	D	X	M	W													
DADDI R1, R1, #1		F	D	s	X	M	W											
SD R1, 0(R2)			F	s	D	X	M	W										
DADDI R2, R2, #4					F	D	X	M	W									
DSUB R4, R3, R2						F	D	X	M	W								
BNEZ R4, Loop							F	s	D	X	M	W						
LD R1, 0(R2)									F	D	X	M	W					



Again we have 99 iterations. We still experience two RAW stalls, but since we correctly predict the branch, we do not need to flush after the branch. Thus, we have only 8 × 98 + 12 = 796.

Quiz #3

1. Ture or False
   - [T/F] The throughput of a pipeline is decided by the fast pipeline stage.
   - [T/F] The performance of a machine X is improved for a program P when the execution time of P in X decreases

2. Suppose an enhancement on machine M1 makes instructions of type A 10 times faster, and that when executing a program P, machine M1 exec utes type A instructions 40% of the time. What is the overall speedup of program P because of this enhancement?

3. Choose one of A, B, C, D. As pipeline depth increases, the latency to pr ocess a single instruction:
   A. increases
   B. decreases
   C. stays the same
   D. could increase, decrease, or stay the same, depending on ....

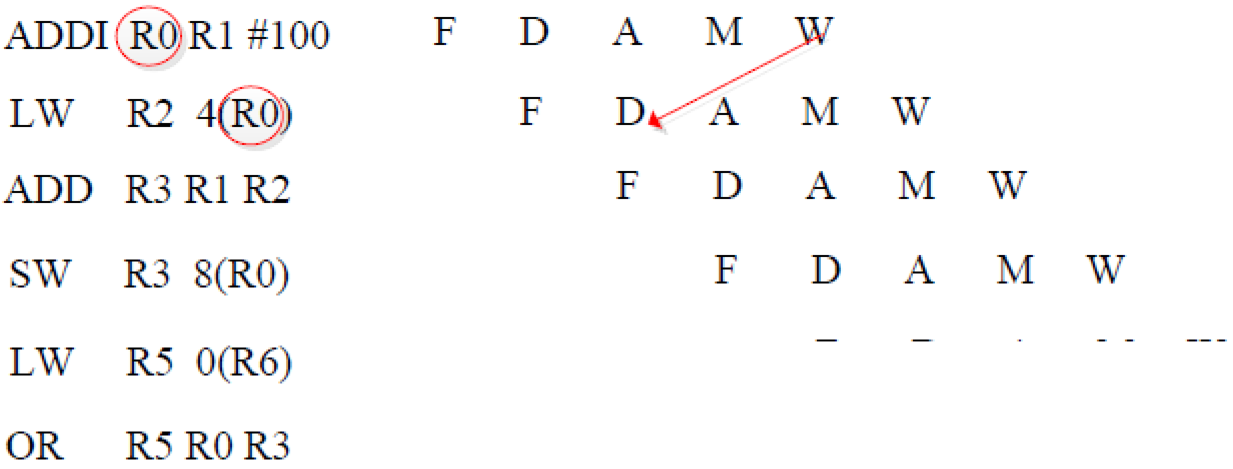
   (b) Explain your reason (in no more than 30 words):

Quiz #3

Spot all data dependencies (including ones that do not lead to stalls). Draw arrows from the stages where data is made available, directed to where it is needed.

(a) Circle the involved registers in the instructions. Assume no forwarding. One dependency has been drawn for you. How many stalls will we have to add to the pipeline to resolve the hazards?

(b) If allow forwarding, how many stalls will we have to add to the pipeline to resolve the hazards?



ADDI R0 R1 #100	F	D	A	M	W			
LW R2 4(R0)		F	D	A	M	W		
ADD R3 R1 R2			F	D	A	M	W	
SW R3 8(R0)				F	D	A	M	W
LW R5 0(R6)								
OR R5 R0 R3								