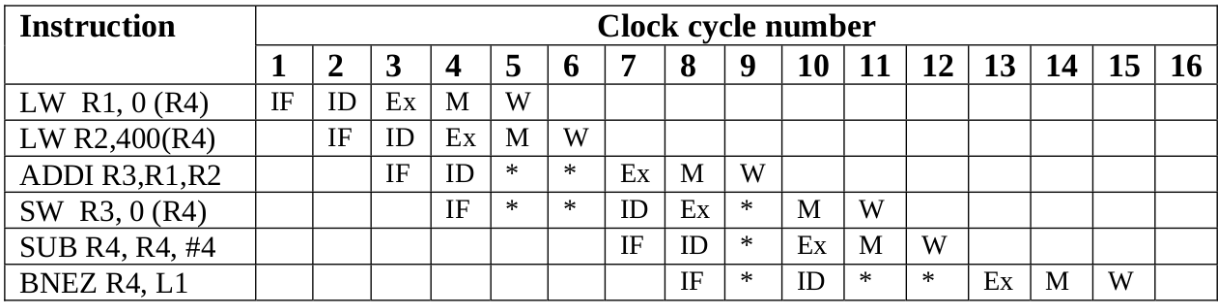
For all following questions we assume that:
a) Pipeline contains 5 stages: IF, ID, EX, M and W;
b) Each stage requires one clock cycle;
c) All memory references hit in cache;
d) Following program segment should be processed:

```
// 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 (non-pipelined) architecture.

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



Instruction	**Clock cycle number**															
	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,R2			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	

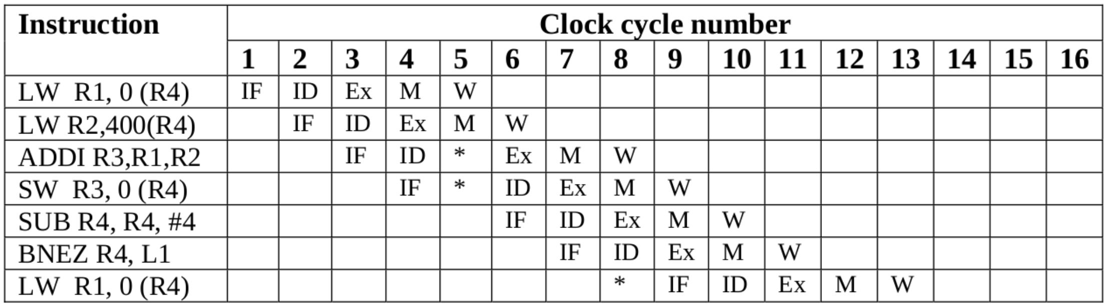
1. Two stall cycles (c.c. # 5 and 6) are caused by the delay of data in the register R2 for the ADDI
2. Same stall cycles in ID stage for the SW instruction are because ID stage circuits are busy for ADDI and becoming available only on 7-th c.c.
3. SUB can start only on 8-th c.c. because IF stage is busy with SW instruction.
4. 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 instructions).

5. 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.).

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.

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

Question 3
Calculate how many clock cycles will take execution of this segment on the simple pipeline with normal forwarding and bypassing when result of branch instruction (new PC content) is available after completion of the ID stage.



Instruction	**Clock cycle number**															
	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,R2			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					
LW R1, 0 (R4)								*	IF	ID	Ex	M	W			

1. Data (R2) for the ADDI is ready after “M” stage of the LW R2. During the “WB” stage the requested operand will be written to the R2 and operation register (e.g. Reg. A) of the ALU.
2. ID stage for the SW is delayed because it is busy with ADDI.
3. BNEZ can initiate IF stage of the LW R1, 0(R4) because new PC-content is ready after 8 c.c.

Number of cycles in the loop = 8 c.c.
Speedup of the pipelined processor with forwarding comparing with non-pipelined processor = 3005 c.c. / (1 c.c. + 400/4 x 8 c.c.) = 3005 / 801 = 3.75 times

Question 4
Schedule the segment instructions including branch-delay slot to get minimum processing time assuming that pipeline has normal forwarding and bypassing hardware. It is possible to reorder instructions and change position of loop label (L1) but not name of registers or op-code modification.

Instruction	Clock cycle number															
	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										
SUB R4, R4, #4			IF	ID	Ex	M	W									
ADDI R3,R1,R2				IF	ID	Ex	M	W								
BNEZ R4, L1					IF	ID	Ex	M	W							
SW R3, 4(R4)						IF	ID	Ex	M	W						
LW R1, 0 (R4)							IF	ID	Ex	M	W					

There are two time slots not used (stalls): during the 5-th and 8-th clock cycles. Thus, if we reschedule the SUB instruction after the LW R2 then we will fill the 5-th c.c. slot. SW instruction may be moved to the end of the loop (after BNEZ) to fill the 8-th c.c. time slot. Now, there will not be any stalls and the minimum number of clock.

Cycles for the segment processing will be = 6 c.c.
The maximum speedup comparing with non-pipelined processor
= 3005 / (1+ 6 x 100) = 5 times. It means that all stages of 5-stage pipeline are always busy (no stalls) during the task segment execution.