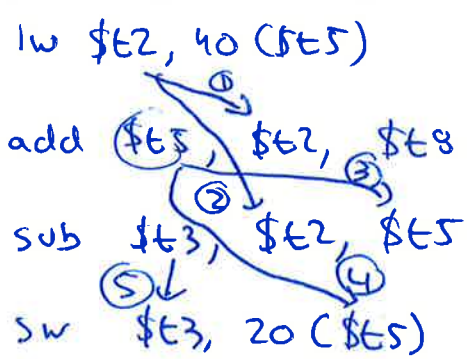
Consider the following RISC-V code sequence

```
lw  $t2, 40($t5)
add $t5, $t2, $t8
sub $t3, $t2, $t5
sw $t3, 20($t5)
```

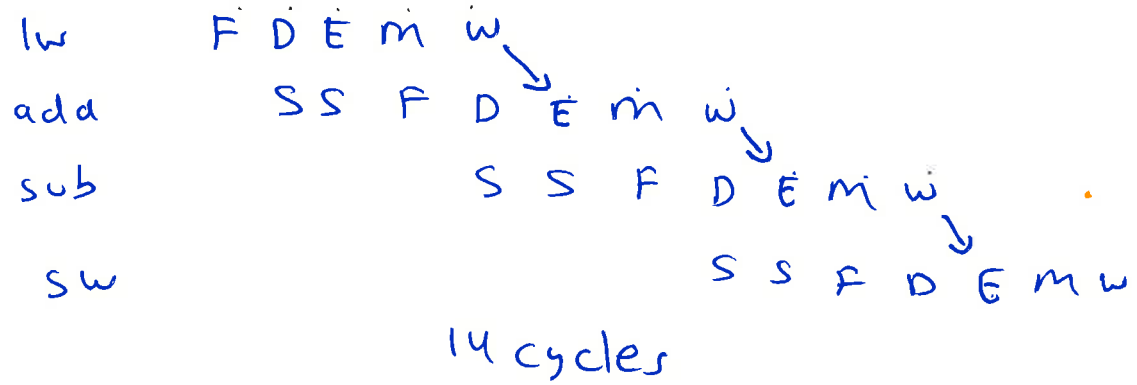
a) *Assuming no forwarding*, identify all pipeline hazards between pairs of instructions



1) $t2 lw → add
2) $t2 lw → sub
3) $t5 add → sub
4) $t5 add → sw
5) $t3 sub → sw

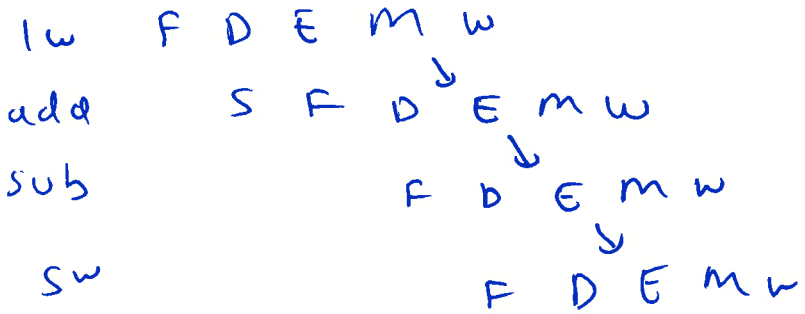
Also Valid

b) *Assuming no forwarding*, insert *stalls* as needed to overcome these hazards. How many clock cycles are needed to finish these instructions?



14 cycles

c) Assuming we use forwarding, insert stalls as needed to overcome these hazards. *How many clock cycles are needed to finish executing these instructions*?

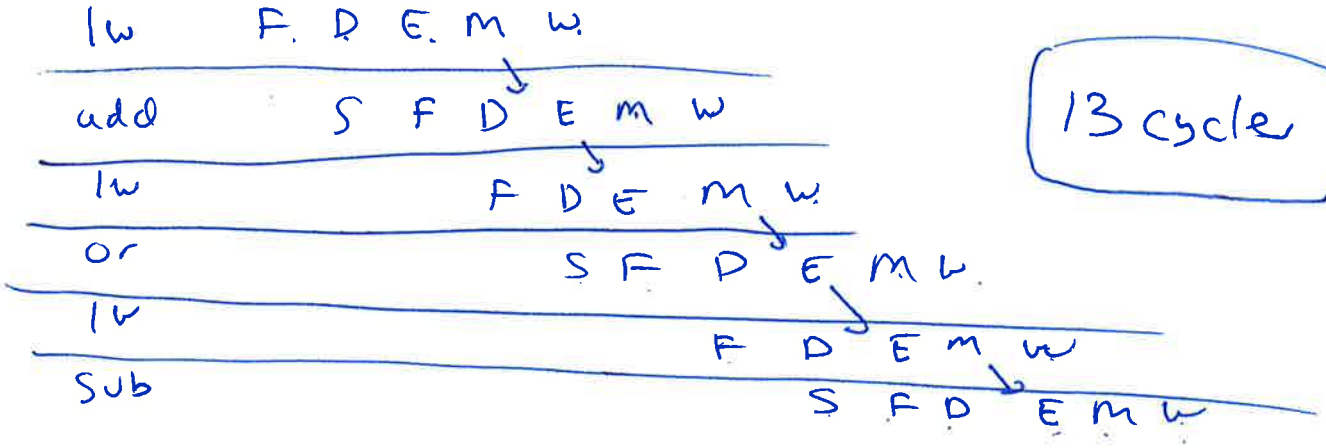


9 cycles

Consider the following RISC-V code sequence

```
1  lw  $x18, 40($x5)
2  add $x22, $x18, $x22
3  lw  $x19, 40($x6)
4  or $x23, $x18, $x19
5  lw  $x24, 40($x7)
6  sub $x22, $x24, $x23
```

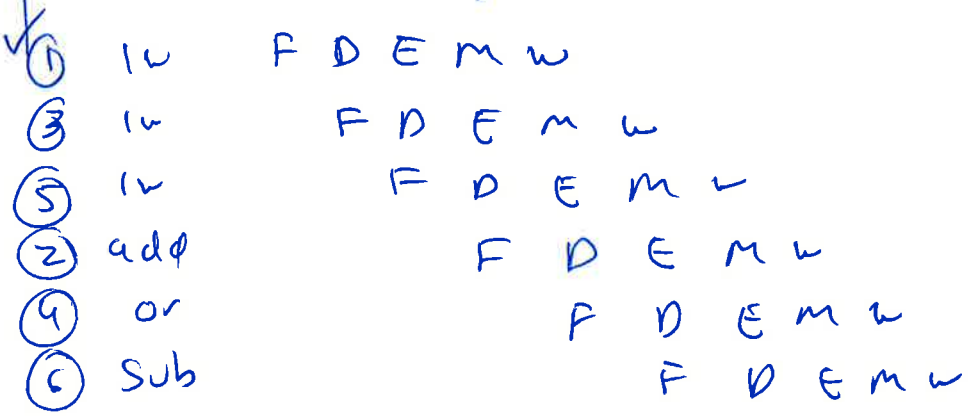
a) Assuming we use forwarding, insert stalls as needed to overcome these hazards. How many clock cycles are needed to finish executing these instructions?



13 cycle

b) Rearrange the instructions to minimize the need for stalls. How many clock cycles are needed to finish executing these instructions?

~~lw $x18, 40(x5)~~
~~add x22, x18, x22~~



10 cycle