s

[**Question 1 1**](#_w34fn0j3xv07)

[**Question 2 3**](#_7hb4t3d24j47)

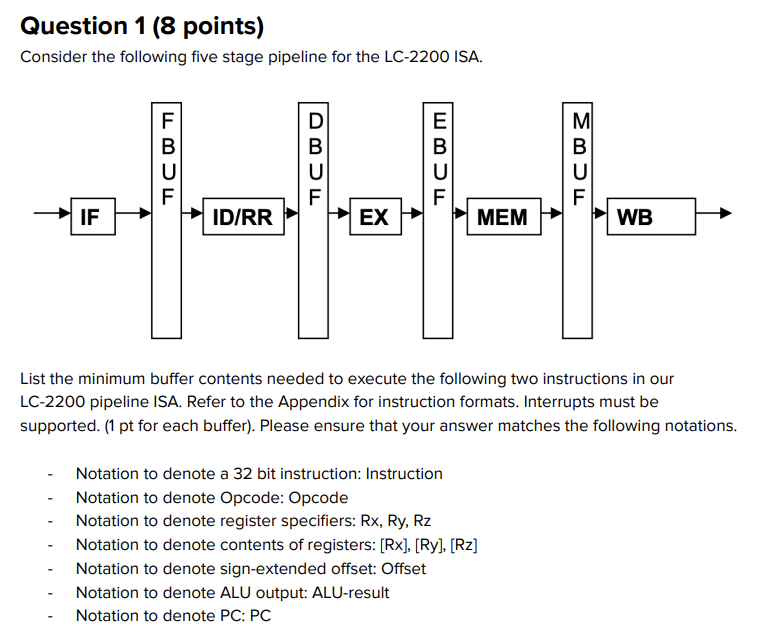
[**Question 3 4**](#_5y4z7svhhef6)

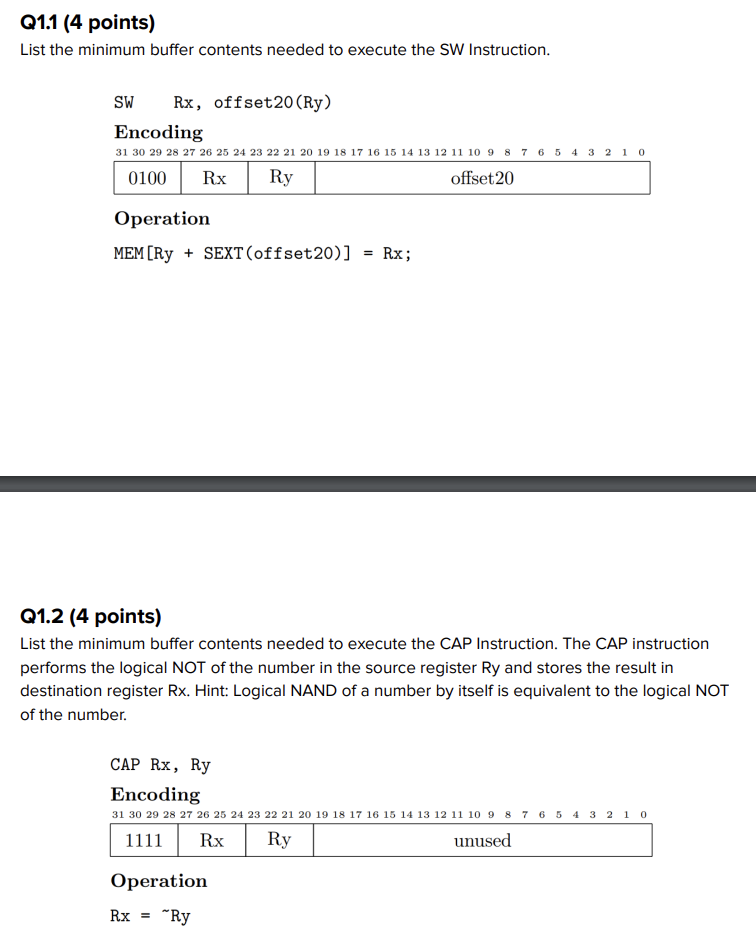
[**Question 4 6**](#_wt5l4it7wg86)

[**Question 5 8**](#_ivpwuvaam2ef)

[**Question 6 10**](#_2w9dyvem4g5s)

### Question 1

w



`**1.1**

FBUF: Instruction and PC

DBUF: [Rx], [Ry], opcode, PC, Offset,

EBUF: PC, opcode,[Rx], ALU Result

MBUF: PC, opcode

**1.2**

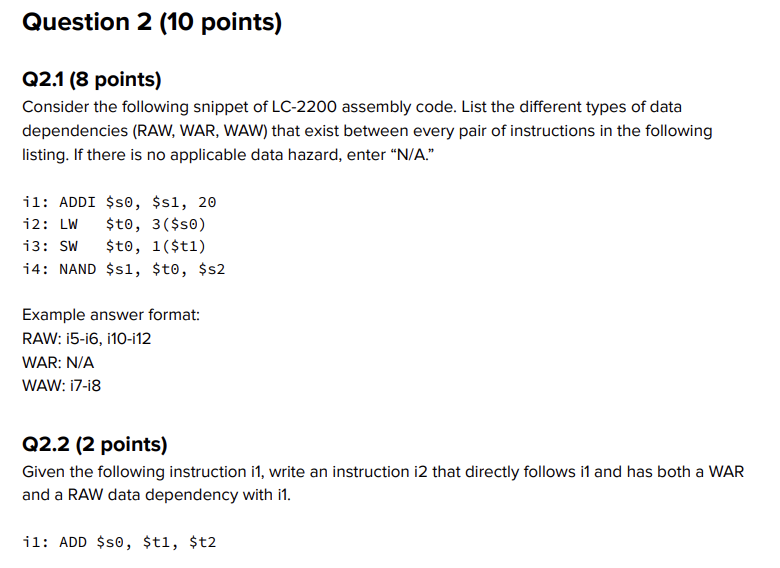
FBUF: Instruction and PC

DBUF: PC, opcode, Rx, [Ry]

EBUF: ALU Result, opcode, Rx, PC

MBUF: ALU Result, opcode, Rx, PC

### Question 2



**2.1**

|  | i1 | i2 | i3 | i4 |
| --- | --- | --- | --- | --- |
| i1 |  |  |  |  |
| i2 | RAW |  |  |  |
| i3 |  | RAW |  |  |
| i4 | WAR | RAW |  |  |

RAW: i1-i2, i2-i3, i2-i4

WAR: i1-i4

WAW: N/A

**2.2**

Given the following instruction:

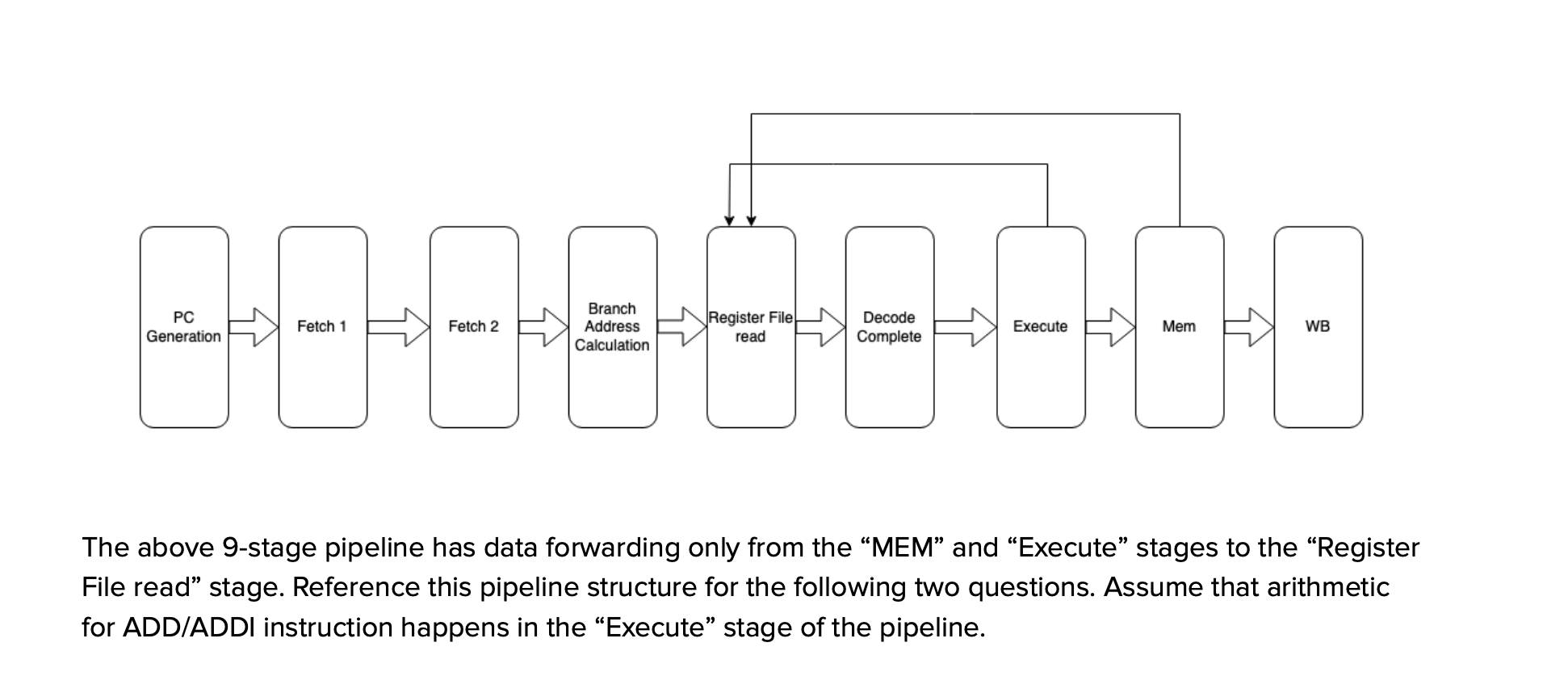
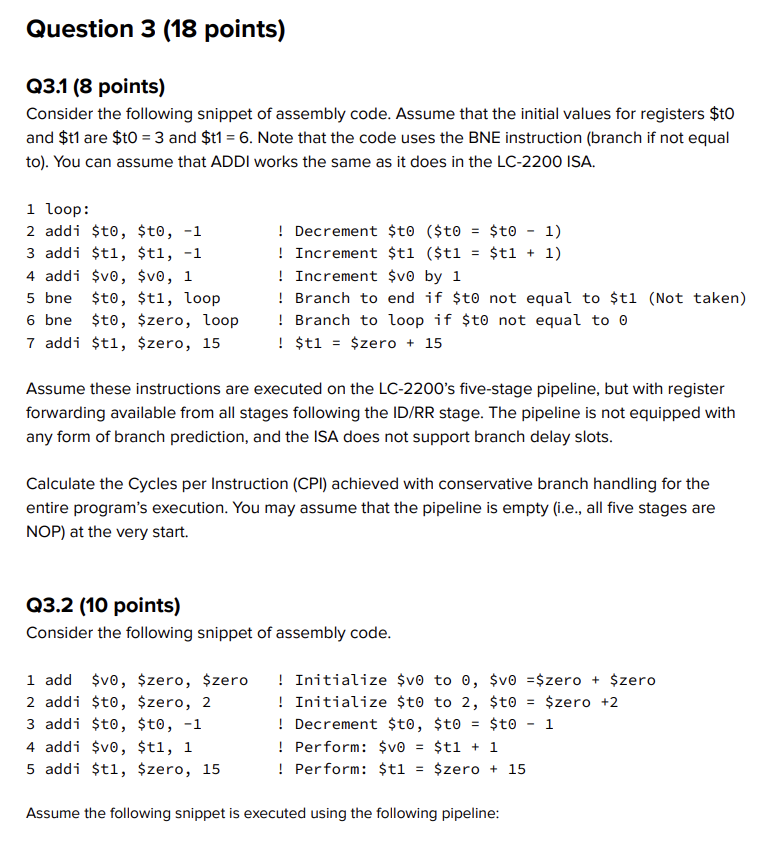
ADD $s0, $t1, $t2

ADD $t1, $s0, $t2

Both WAR (write after read) and RAW (read after write)

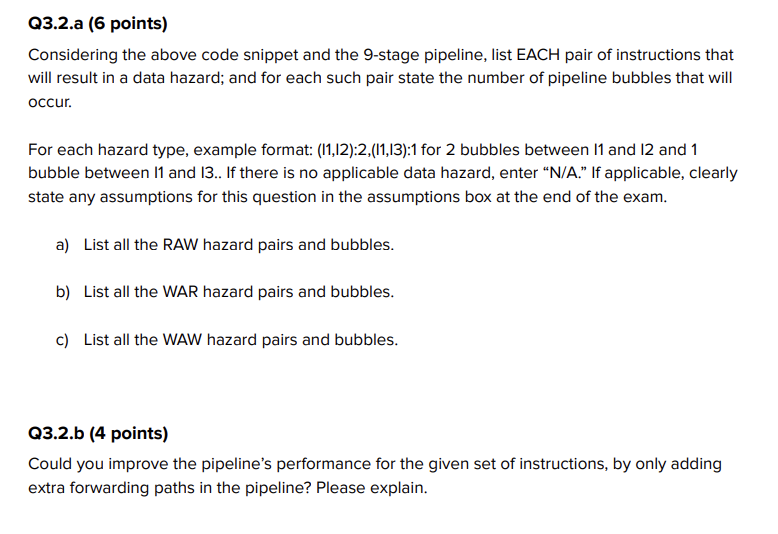
ni

### **Question 3**



5 instructions executed + 2 bubbles (2 instructions between taken bne and end) + 4 (setup)

CPI = cycles/instructions = 11/5



SEE CLARIFICATIONS FOR UPDATED PROBLEMS

1. RAW: i2-i3, 1 bubble

WAR: i4-i5

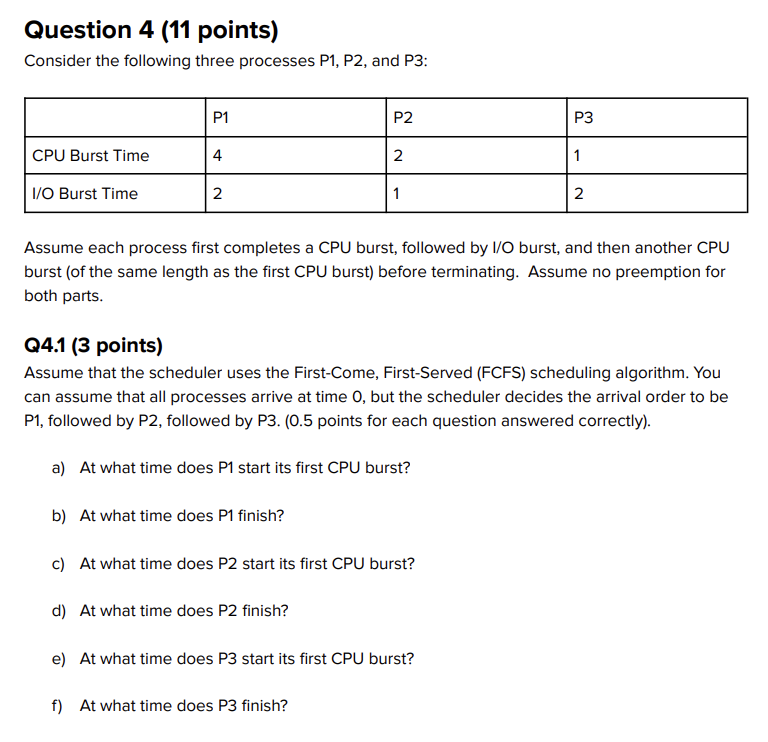
WAW: i1-i4, i2-i3

If the pipeline has priority data forwarding, no bubbles are created. WAW only necessitates bubbles if there is a following RAW, and in the above code there are no such sequences. With priority data forwarding, if such a situation were to ok occur, the bubble propagated by RAW would ensure the correct value is forwarded back to the register read (the closest value to register read would be the correct value to be forwarded back in this case once the bubble is propagated).

Without priority data forwarding, WAW would require bubbles if there was a subsequent RAW, as we would need the first write instruction to clear the pipeline before we allow the second write to proceed to prevent the first write from data forwarding back the incorrect value.

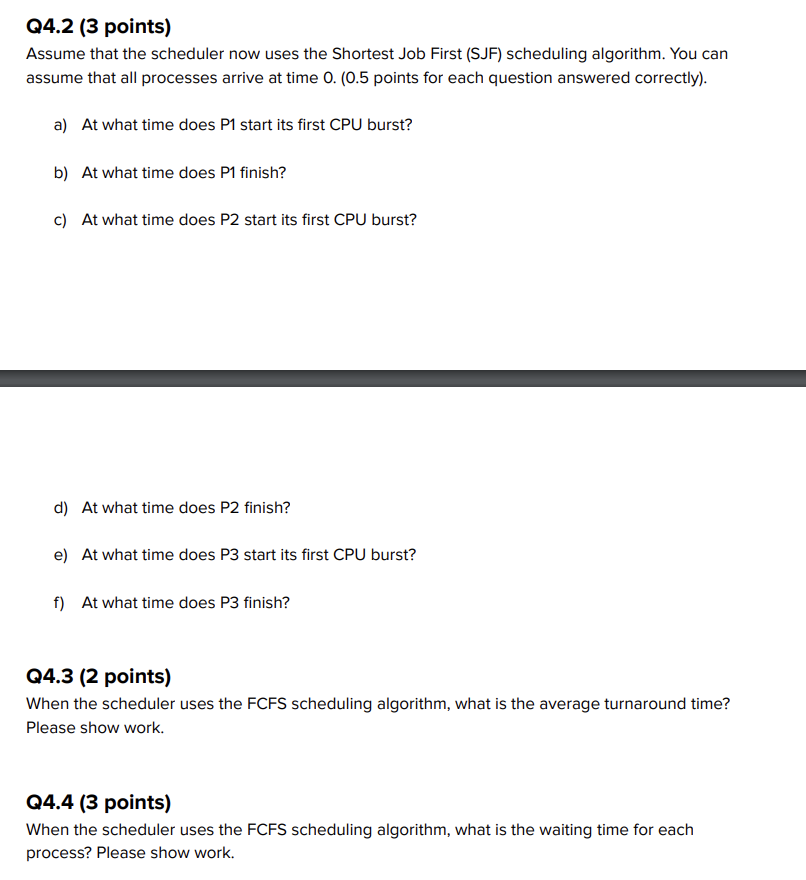
1. We could add a data forwarding path back from execute to decode complete to eliminate the need for RAW propagating bubbles. Note that this would require decode complete to have a mechanism to replace the value that it received from register file read with the updated value forwarded back to it from execute.

### Question 4



**4.1**

1. P1 starts: 0
2. P1 ends: 10
3. P2 starts: 4
4. P2 ends: 12
5. P3 starts: 12
6. P3 ends: 16



4.2

1. P1 starts: 6
2. P1 ends: 16
3. P2 starts: 1
4. P2 ends :6
5. P3 starts: 0
6. P3 ends: 4

4.3

Average turnaround time: (rt1+rt2 + rt3)/3 = (10+12+16)/3 = 38/3

4.4

Waiting time for each:

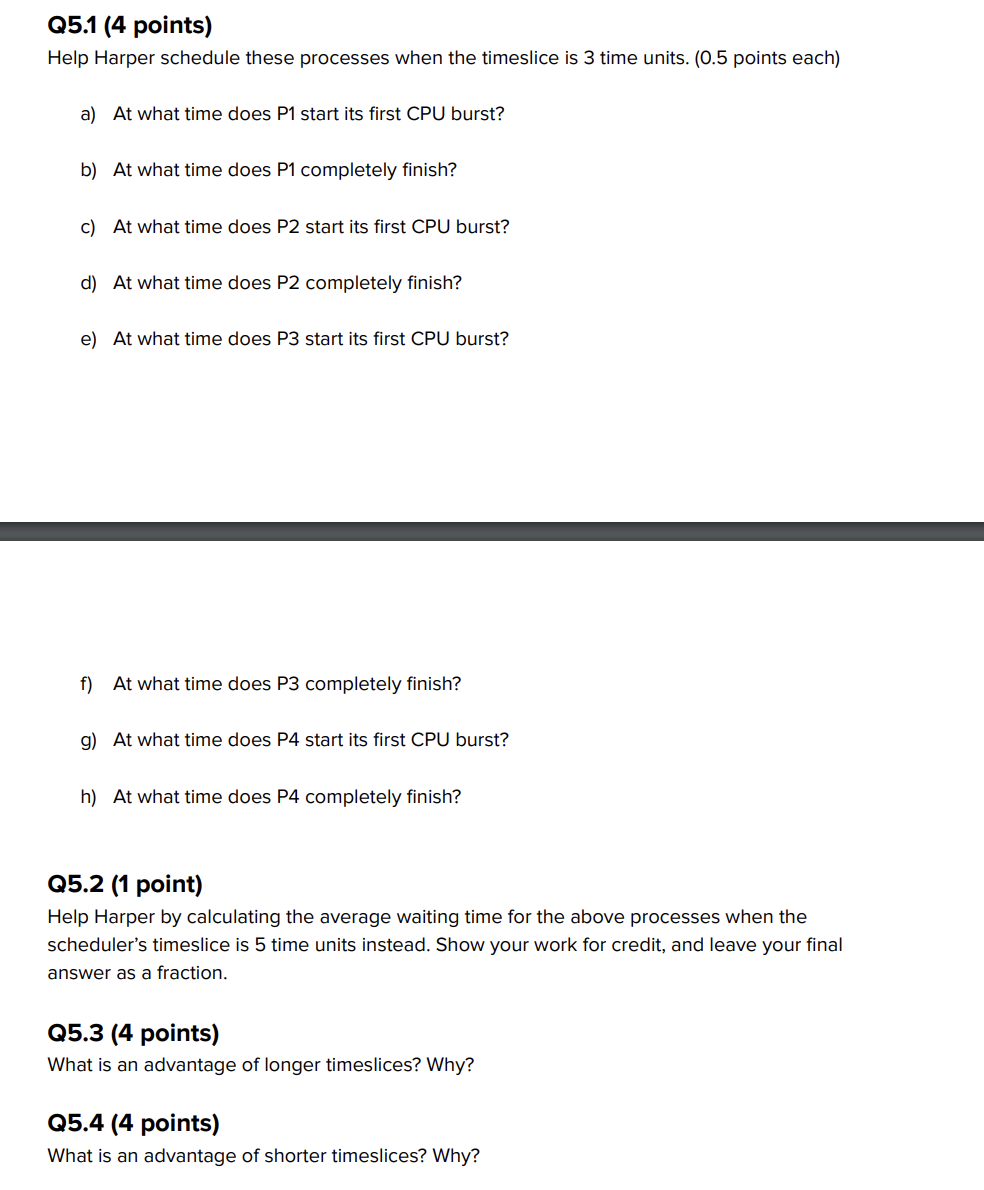
Waiting time p1: response time - execution

P1: 10 - 10 = 0

P2: 12 - 5 = 7

P3: 16 - 4 = 12

### Question 5



5.1

1. P1 starts: 0
2. P1 ends: 24
3. P2 starts: 3
4. P2 ends: 14
5. P3 starts: 5
6. P3 ends: 16
7. P4 starts: 7
8. P4 ends: 17

5.2

Wait Time = Response Time - Execution Time

P1 Wait Time: 22 - 13 = 9

P2 Wait Time: 17 - 6 = 11

P3 Wait Time: 19 - 5 = 14

P4 Wait Time: 20 - 4 = 16

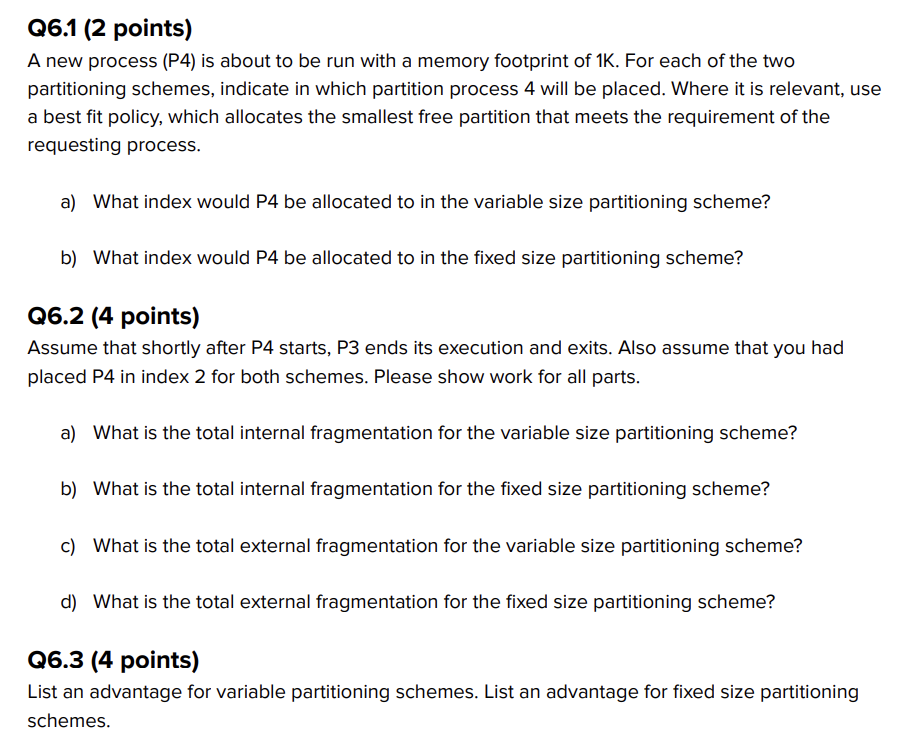
= (16 + 14 + 11 + 9) / 4 = 50/4 = 25/2 = 12.5

5.3

Longer time slices decrease the context switching overhead (the time to switch between processes) and decrease the wait time of longer jobs as they require fewer rounds/quanta to complete

5.4  
Shorter time slices reduce the waiting time of shorter jobs through reducing the convoy effect (series of short jobs waiting for the completion of a long job).

### Question 6



6.1

1. 4
2. 5

6.2

1. 0
2. 10
3. 0
4. 0

6.3

Fixed size partitioning is simple and variable partitioning eliminates internal fragmentation because it can change partition sizes to accommodate for new processes