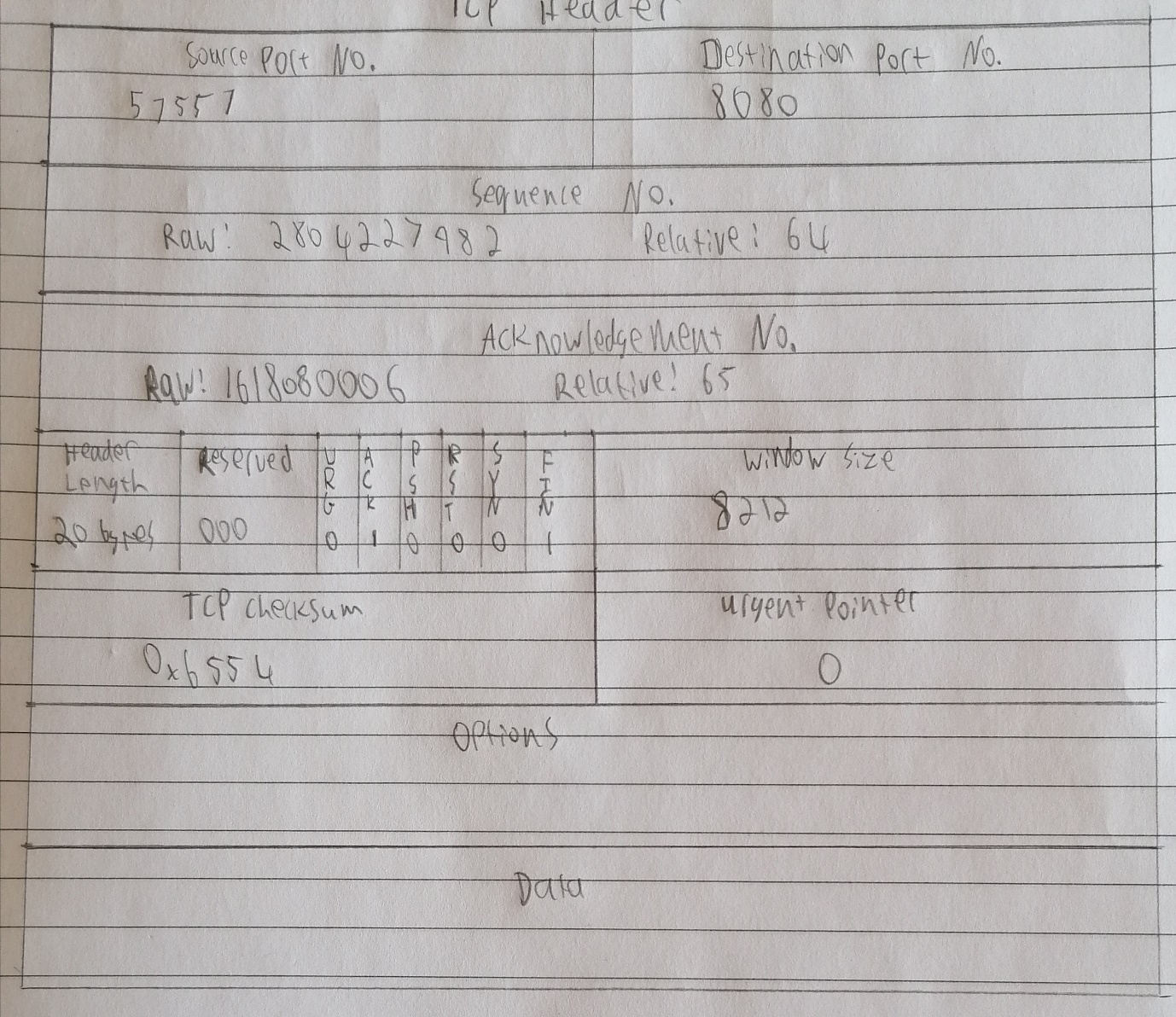
Question 1

**Source port**: this is a 16-bit field that specifies the port number of the sender.

**Destination port**: this is a 16-bit field that specifies the port number of the receiver.

**Sequence number**: the sequence number is a 32-bit field that indicates how much data is sent during the TCP session. When you establish a new TCP connection (3-way handshake) then the initial sequence number is a random 32-bit value. The receiver will use this sequence number and sends back an acknowledgment. Protocol analysers like Wireshark will often use a *relative sequence number of 0* since it’s easier to read than some high random number.

**Acknowledgment number**: this 32-bit field is used by the receiver to request the next TCP segment. This value will be the sequence number incremented by 1.

**DO**: this is the 4-bit data offset field, also known as the header length. It indicates the length of the TCP header so that we know where the actual data begins.

**RSV**: these are 3 bits for the reserved field. They are unused and are always set to 0.

**Flags**: there are 9 bits for flags, we also call them control bits. We use them to establish connections, send data and terminate connections:

**URG**: urgent pointer. When this bit is set, the data should be treated as priority over other data.

**ACK**: used for the acknowledgment.

**PSH**: this is the push function. This tells an application that the data should be transmitted immediately and that we don’t want to wait to fill the entire TCP segment.

**RST**: this resets the connection, when you receive this you have to terminate the connection right away. This is only used when there are unrecoverable errors and it’s not a normal way to finish the TCP connection.

**SYN**: we use this for the initial three-way handshake and it’s used to set the initial sequence number.

**FIN**: this finish bit is used to end the TCP connection. TCP is full duplex so both parties will have to use the FIN bit to end the connection. This is the normal method how we end a connection.

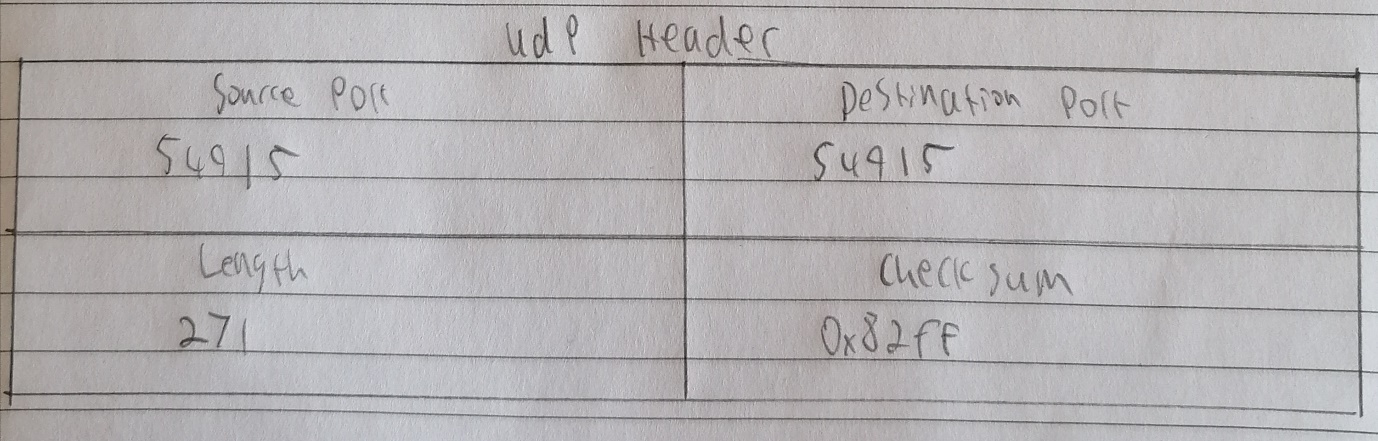
**Window**: the 16-bit window field specifies how many bytes the receiver is willing to receive. It is used so the receiver can tell the sender that it would like to receive more data than what it is currently receiving. It does so by specifying the number of bytes beyond the sequence number in the acknowledgment field.

**Checksum**: 16 bits are used for a checksum to check if the TCP header is OK or not.

**Urgent pointer**: these 16 bits are used when the URG bit has been set, the urgent pointer is used to indicate where the urgent data ends.

**Options**: this field is optional and can be anywhere between 0 and 320 bits

# Question 2



**Source port:** The port of the device sending the data. This field can be set to zero if the destination computer doesn’t need to reply to the sender.

**Destination port**: The port of the device receiving the data. UDP port numbers can be between 0 and 65,535.

**Length:** Specifies the number of bytes comprising the UDP header and the UDP payload data. The limit for the UDP length field is determined by the underlying IP protocol used to transmit the data.

**Checksum:** The checksum allows the receiving device to verify the integrity of the packet header and payload. It is optional in IPv4 but was made mandatory in IPv6.

# Question 3

c0 a8 00 65

c0 a8 00 0a

00 11 00 10

c2 f1 13 e7

00 10 00 00

01 00 01 00

00 0d 00 08

c0 a8 1100 0000 1010 1000

00 65 0000 0000 0110 0101

==>

c0 a8

1100 0001 0000 1101 (previous result)

1100 0000 1010 1000 (current value c0a8)

==>

00 0a

1000 0001 1011 0110 (previous result)

0000 0000 0000 1010 (current value 000a)

==>

00 11

1000 0001 1100 0000

0000 0000 0001 0001

==>

00 10

1000 0001 1101 0001

0000 0000 0001 0000

==>

c2 f1

1000 0001 1110 0001

1100 0010 1111 0001

==>

13 e7

0100 0100 1101 0011

0001 0011 1110 0111

==>

00 10

0101 1000 1011 1010

0000 0000 0001 0000

==>

01 00

0101 1000 1100 1010

0000 0001 0000 0000

==>

01 00

0101 1001 1100 1010

0000 0001 0000 0000

==>

00 0d

0101 1010 1100 1010

0000 0000 0000 1101

==>

00 08

0101 1010 1101 0111

0000 0000 0000 1000

==>

0101 1010 1101 1111

Reverse bits: 1010 0101 0010 0000

Checksum: a520

# Question 4

While streaming a video from Youtube, it seemed to use the TCP protocol.