Name and Section: ______________________________

1 Packet Decode Preparation

- This **packet decode** is provided to help reinforce encapsulation and gain a better understanding of the IP header as well as and fragmentation
- Provided below are the packet headers for an Ethernet frame, IPv4 datagram and an ICMP message. You do not need to memorize the structure, however, you should have a complete understanding of what each field represents and how to decode the given fields/frames if given the structure and the corresponding bytes in hexadecimal form.

Figure 1: Ethernet II Frame

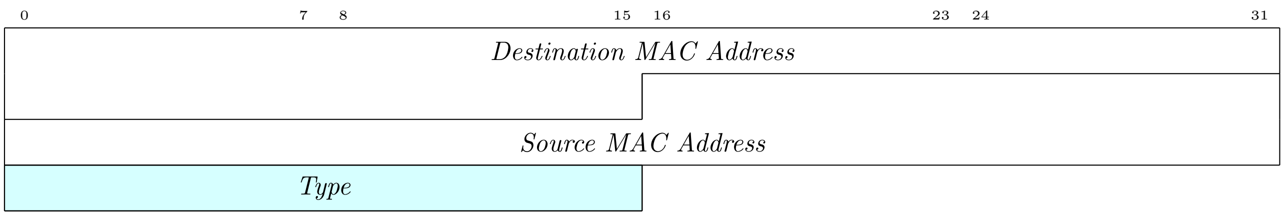


Figure 2: IPv4 Packet

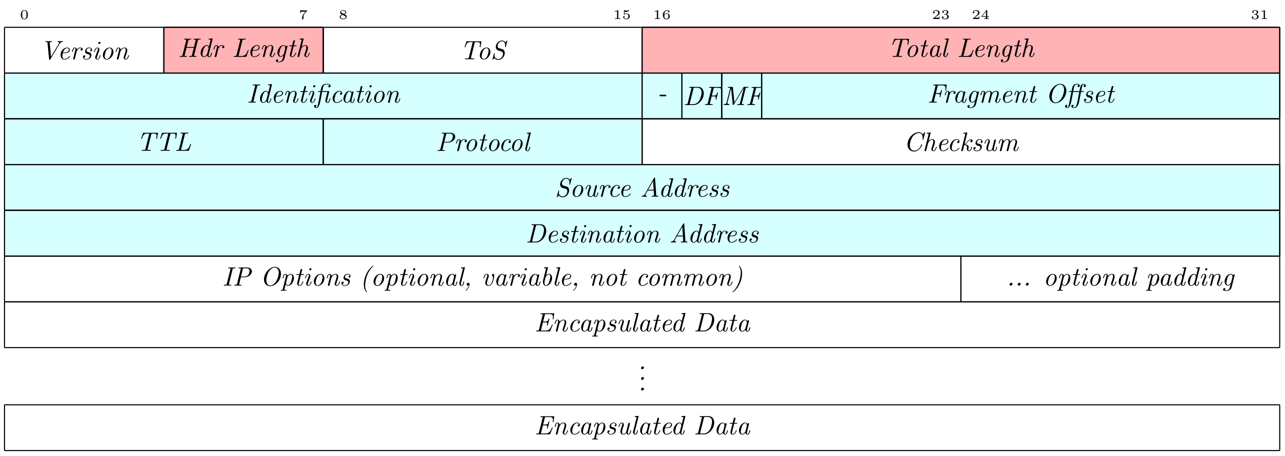
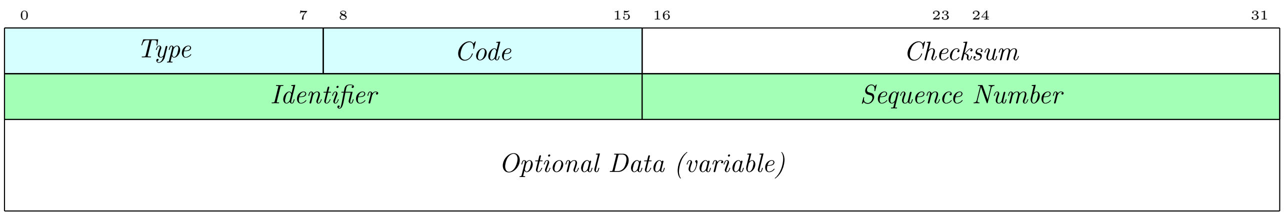


Figure 3: ICMP Echo Request and Reply Messages



2 Packet Decode Assignment

For the following questions you are provided a blank header that you must fill in based on the bytes provided. For all highlighted fields, you should additionally translate from hexadecimal to what the hex value actually represents, just as Wireshark does for you automatically. For example 'Type Field = 0x01 = ICMP'. Layer 3 addresses should be displayed in dotted decimal notation. Fields marked in green should be decoded in Little Endian. After decoding the fields marked red, clearly annotate on the right of the figure where the header ends.

All frames begin with a 14 byte Ethernet II header.

1.
```
00 20 78 e1 5a 80 00 00 65 10 22 1b 08 00 45 00
05 dc 2d 00 20 00 20 01 33 bc 0a 01 00 01 0a 01
00 63 08 00 2d 42 01 00 06 00 61 62 63 64 65 66
67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76
...
75 76 77 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d
6e 6f 70 71 72 73 74 75 76 77
```

Figure 4: Ethernet II Frame

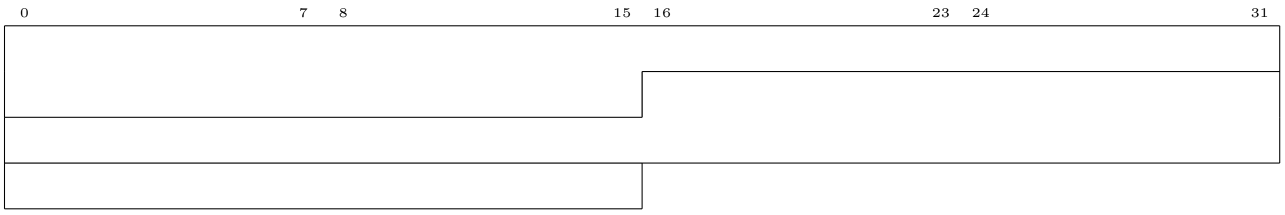


Figure 6: IP v4 Packet

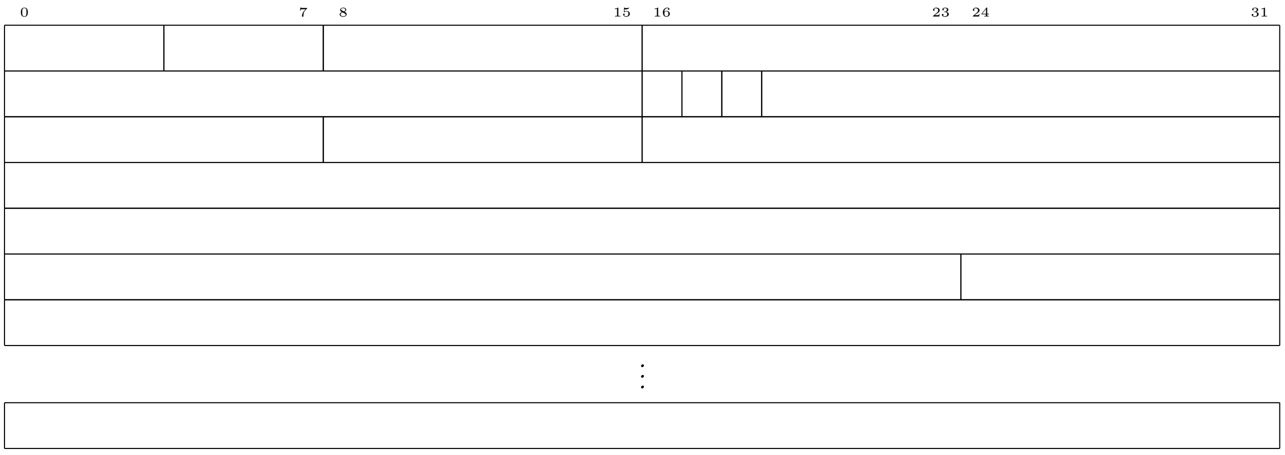
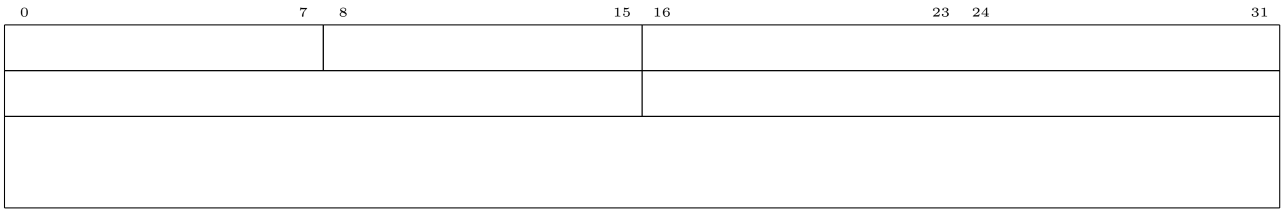


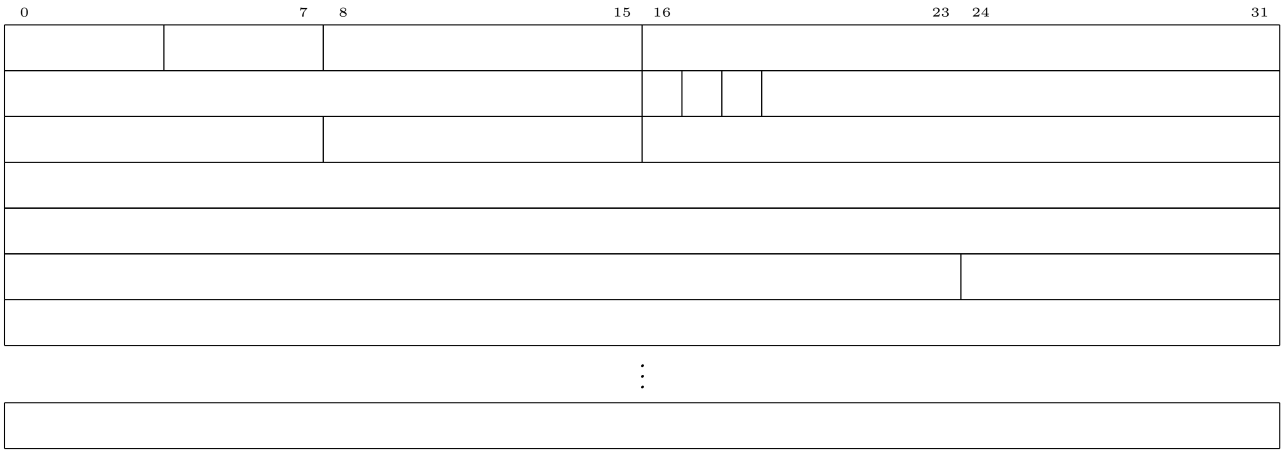
Figure 8: ICMP Echo Request and Reply Messages



2. For the purpose of this question the ethernet header in red can be skipped (It is the same as the first packet...).

```
00 20 78 e1 5a 80 00 00 65 10 22 1b 08 00 45 00
05 dc 2d 00 20 b9 20 01 33 03 0a 01 00 01 0a 01
00 63 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e
6f 70 71 72 73 74 75 76 77 61 62 63 64 65 66 67
...
66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75
76 77 61 62 63 64 65 66 67 68
```

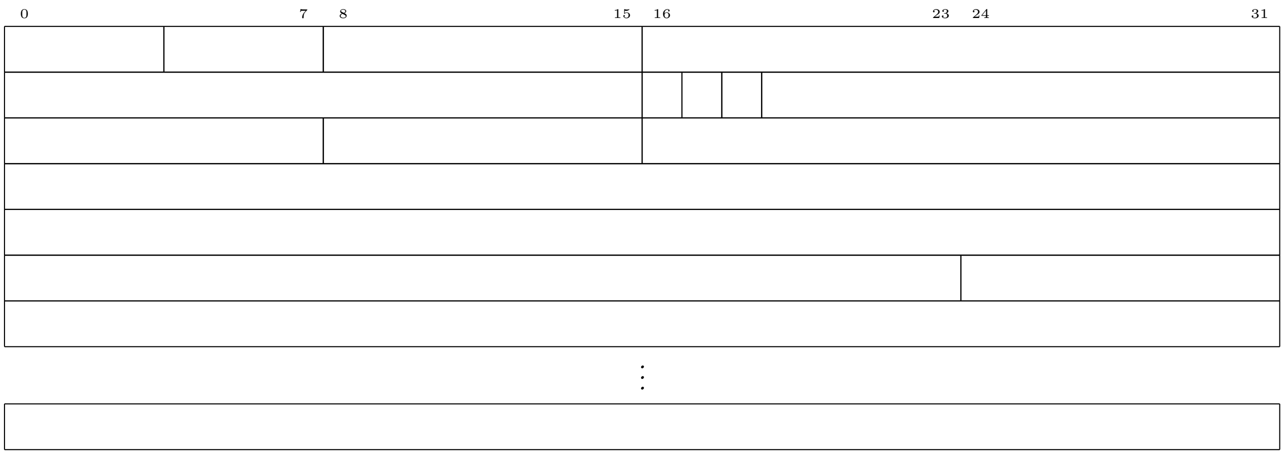
Figure 10: IP v4 Packet



3. For the purpose of this question the ethernet header in red can be skipped.

```
00 20 78 e1 5a 80 00 00 65 10 22 1b 08 00 45 00
05 dc 2d 00 21 72 20 01 32 4a 0a 01 00 01 0a 01
00 63 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76
77 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
...
6e 6f 70 71 72 73 74 75 76 77 61 62 63 64 65 66
67 68 69 6a 6b 6c 6d 6e 6f 70
```

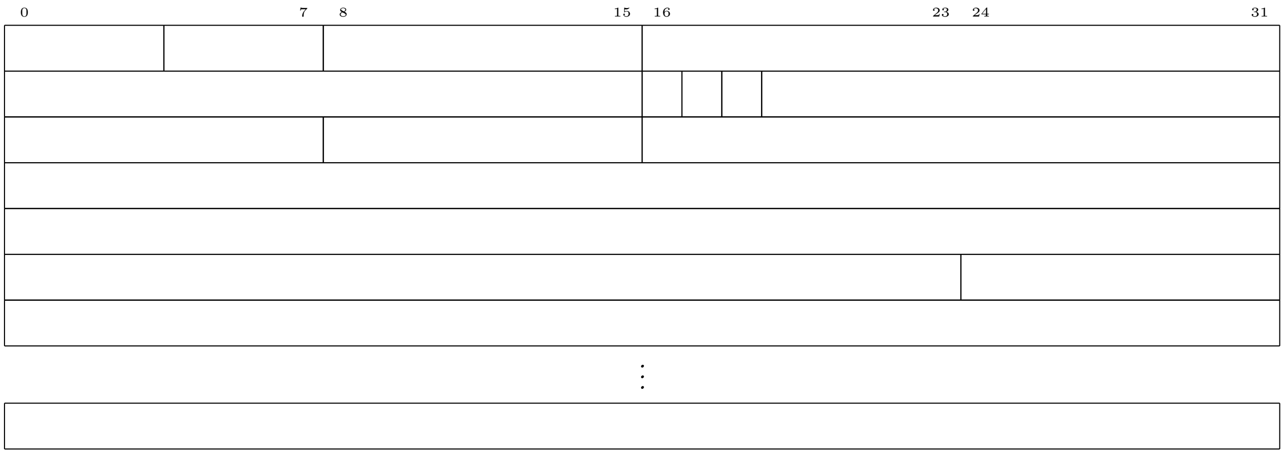
Figure 12: IP v4 Packet



4. For the purpose of this question the ethernet header in red can be skipped.

```
00 20 78 e1 5a 80 00 00 65 10 22 1b 08 00 45 00
04 a4 2d 00 02 2b 20 01 52 c9 0a 01 00 01 0a 01
00 63 71 72 73 74 75 76 77 61 62 63 64 65 66 67
68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77
...
71 72 73 74 75 76 77 61 62 63 64 65 66 67 68 69
6a 6b
```

Figure 14: IP v4 Packet



Review Questions

1. How did we know that data encapsulated within the IPv4 datagram was ICMP?

2. How do we know what type of ICMP message this is? Why didn't we need to use the code field for this message type? What kind of ICMP message would have a Type value of 3? 11?

3. How do you know that these fragments were part of the same initial packet?

4. How do you know which packet was the first fragment?

5. How do you know which packet was the last fragment?

6. What was the total length of the original data in the ICMP message, not including any headers.

7. Why are the TTL values the same for all of the fragments?