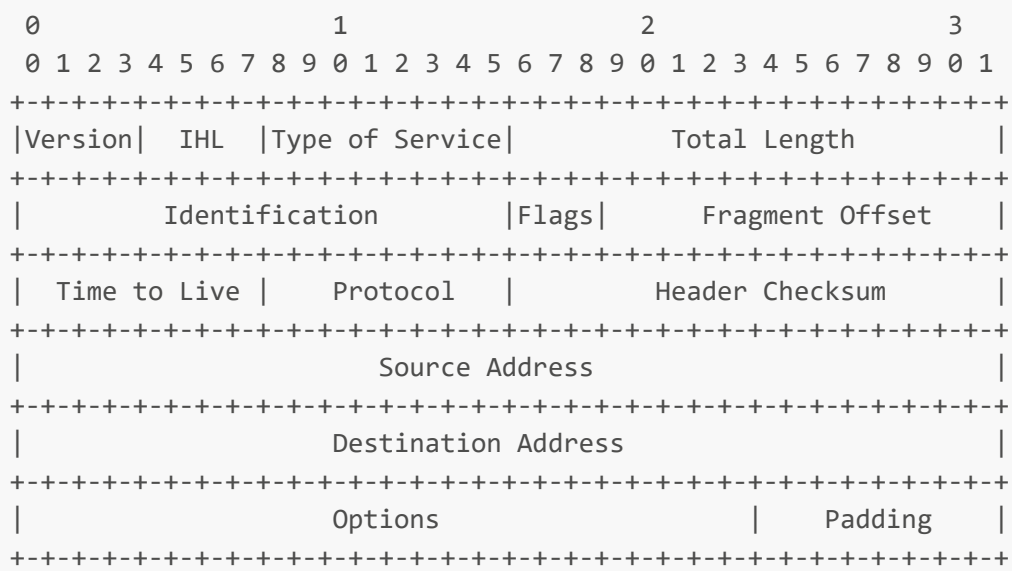
Q1 - Checksum

The Internet checksum, also called the IPv4 header checksum is a checksum used in version 4 of the Internet Protocol (IPv4) to detect corruption in the header of IPv4 packets. It is carried in the IP packet header, and represents the 16-bit result of summation of the header words.

The contents of an IPv4 header are shown below, with each tick mark representing one bit position. For simplication, here we only consider the first 20 bytes of an IPv4 header (i.e., ignoring `Options` and `Padding` fields).



The checksum calculation is defined in RFC 791:

> The checksum field is the 16-bit ones' complement of the ones' complement sum of all 16-bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero.

Take the following IPv4 header for example (values in hexadecimal representation):

```
4500 0073 0000 4000 4011 0000(checksum) c0a8 0001 c0a8 00c7
```

We first calculate the sum of each 16 bit value within the header, note that checksum field in the header is `0000`:

```
4500 + 0073 + 0000 + 4000 + 4011 + c0a8 + 0001 + c0a8 + 00c7 = 2479c
```

Note that the checksum should be 16-bit, but the calculation result is larger than 16-bit. To handle this situation, we truncate the carry `2` and add it back to the sum (wrap-around):

```
2 + 479c = 479e
```

Finally, we take the ones' complement of this result as the checksum:

```
ffff - 479e = b861
```

In this exercise, given a IPv4 header, please calculate the checksum of the corresponding header. The output should be printed in hexadecimal notation (you do not need to omit the 0's). We guarantee that wrap-around operation happens at most once, all alphabetical characters are in lower case, and checksum field in input is always 0.

Sample Input:

```
45000073000040004011000c0a80001c0a800c7
```

Sample Output:

```
0x0000b861
```

The input will be given in a `.txt` (需要确定文件名) file, hence you will need to use Mars and its file I/O functions. Below is a code snippet that reads `test.txt`, and stores its content to memory space marked by `buffer`. You can find more details in Mars help manual.

```
.data
    filename: .asciiz "test.txt"
    fill: .space 3
    buffer: .space 400000

.text
main:
    # open file
    li $v0, 13
    la $a0, filename
    li $a1, 0 # 0 for read
    li $a2, 0
    syscall
    move $s6, $v0 # save file descriptor

    # read file
    li $v0,14
    move $a0,$s6 # load file descriptor
    la $a1, buffer # save read content to buffer space
    li $a2, 40 # reads 40 ascii chars
    syscall
```

Hint:

After reading the input file, its content will be stored as a sequence of ASCII chars. You could use a loop to convert groups of four chars to 16-bit integers while adding them together.