

Introduction to Lab Packet Forward - Checksum

J. Nelson Amaral

Length of the Header (in words)

Used to determine how many halfwords to use to compute the Header Checksum

Byte Offset		0								1								2								3							
Bit Offset		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Word Offset	0	IP Version			Packet Header Length (# words)				Differentiated Services Code Point					ECN			Total Packet Length (# bytes)																
	1	Identification															Flags			Fragment Offset													
	2	Time To Live							Protocol							Header Checksum																	
	3	Source IP Address																															
	4	Destination IP Address																															
	5	Options (if Packet Header Length > 5)																															
	5 / 6+	Data (if Total Packet Length > Packet Header Length * 4)																															

What is the input to your assignment?

a_0 : The address of an IPv4 packet stored in memory

What does **checksum** do?

*Calculate and return the Header Checksum
of an IPv4 Packet*

Computing Header Checksum

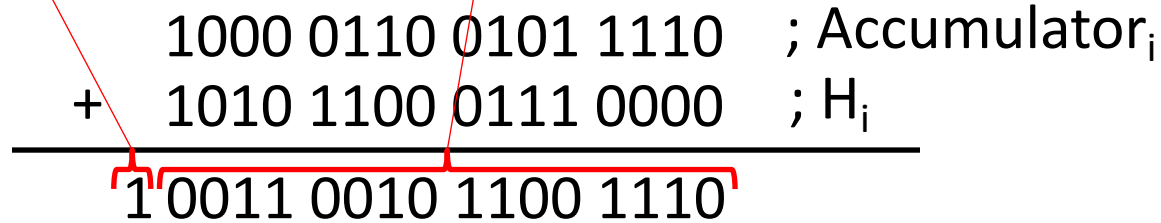
1. Break the packet's header into halfword (16 bit) values
2. Accumulator $\leftarrow 0$
3. **for each** halfword H_i in the header:
 1. (CarryOut, Sum) $\leftarrow$ Accumulator + H_i
 2. Accumulator $\leftarrow$ Sum + Carryout
4. Checksum $\leftarrow$ Logical complement of Accumulator

Note: you must skip the Header Checksum field when computing the checksum

Computing Header Checksum (Example)

CarryOut

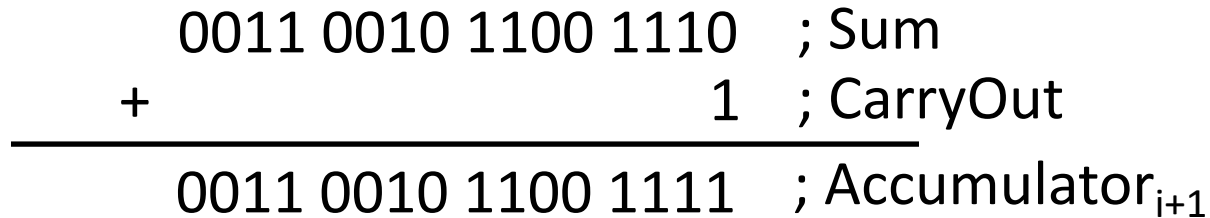
Sum



1000 0110 0101 1110 ; Accumulator_i
+ 1010 1100 0111 0000 ; H_i

1 0011 0010 1100 1110

The diagram shows a binary addition. The first column (leftmost) has a '1' in the top row and a '0' in the bottom row, with a '+' sign between them. A red bracket under the first column of the result row indicates a carry-out of 1. A red line connects the 'CarryOut' label to this bracket. A red bracket under the second column of the result row indicates the start of the sum. A red line connects the 'Sum' label to this bracket.



0011 0010 1100 1110 ; Sum
+ 1 ; CarryOut

0011 0010 1100 1111 ; Accumulator_{i+1}

The diagram shows a binary addition. The first column (leftmost) has a '0' in the top row and a '1' in the bottom row, with a '+' sign between them. A red bracket under the first column of the result row indicates a carry-in of 1. A red line connects the 'CarryOut' label from the previous step to this bracket. A red bracket under the second column of the result row indicates the start of the sum. A red line connects the 'Sum' label from the previous step to this bracket.

Important Details

- The standard for packets sent over a network is big-endian
- RISC-V is little-endian.
- Thus, your solution has to:
 - convert each halfword to little-endian before using it to compute the checksum
 - Return the checksum in big-endian byte order ie. Do **not** swap the bytes of the calculated checksum before returning the value

Halfword Endianness Conversion (Example)

1000 0110 0101 1110

Little Endian

Big Endian

You need to write three functions

- checksum
 - Argument:
 - `a0`: address of IP packet
 - Returns:
 - `a0`: calculated checksum, in lower halfword in big-endian byte order.

You need to write three functions

- `flipHalfwordBytes`

- Argument:

- `a0`: a value

- Returns:

- `a0`: the argument value with the order of two bytes of the lower halfword reversed

You need to write three functions

- `getHeaderLength`

- Argument:

- `a0`: address of an IP packet in memory

- Returns:

- `a0`: the value of the packet's *Packet Header Length* field in the lowest four bits