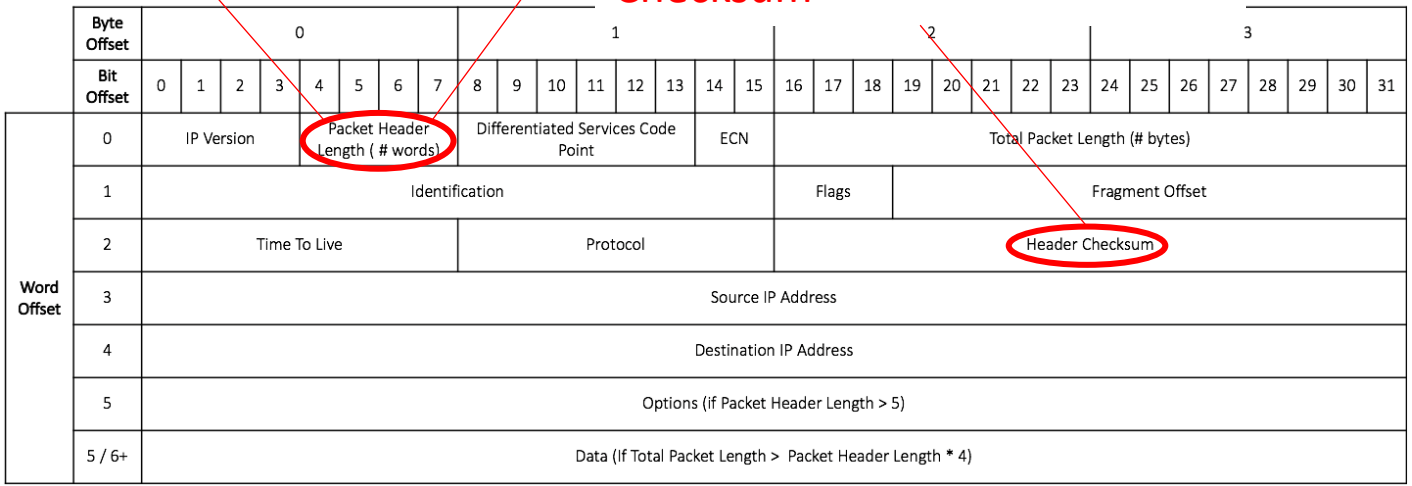
Introduction to Lab Packet Forward - Checksum

J. Nelson Amaral

Layout of an IP Packet

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?

`a0`: The address of an IPv4 packet stored in memory

What does `checksum` do?

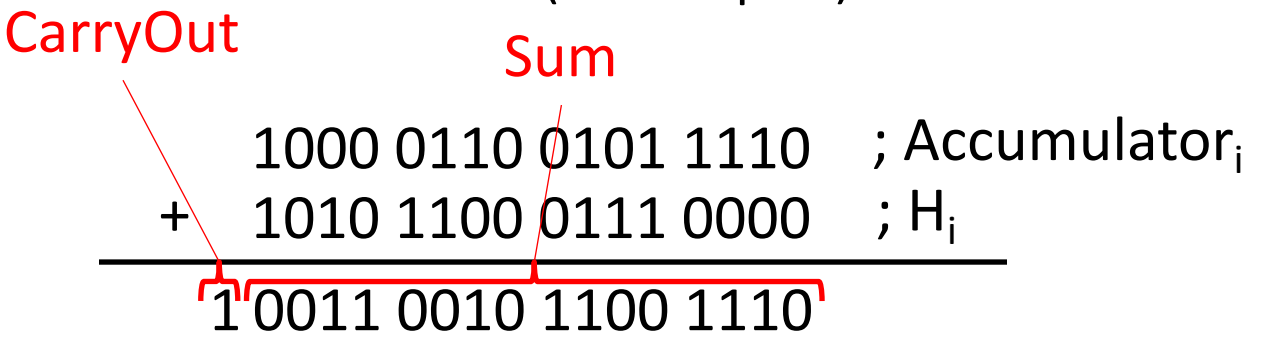
Calculate and return the Header Checksum of an IPv4 Packet

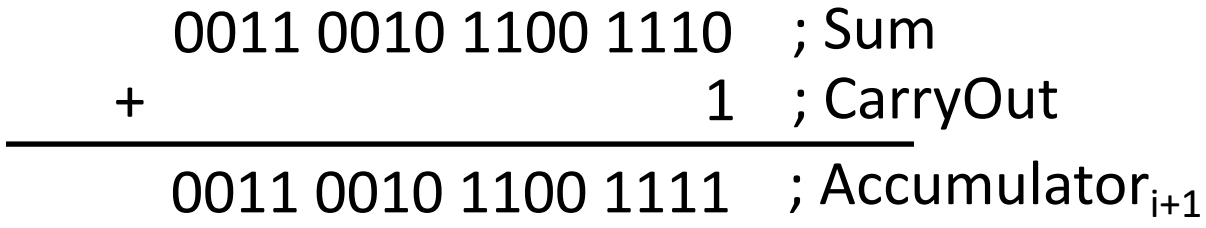
Computing Header Checksum

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

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

Computing Header Checksum (Example)





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 Endianess Conversion (Example)

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:` addres of an IP packet in memory
  - Returns:
    - `a0`: the value of the packet's *Packet Header Length* field in the lowest four bits